Univerzitet Singidunum



Ivana Štrumberger
Nebojša Bačanin-Džakula

KLAUD RAČUNARSTVO

Beograd, 2021.

UNIVERZITET SINGIDUNUM

Ivana Štrumberger
Nebojša Bačanin-Džakula

KLAUD RAČUNARSTVO

DRUGO IZDANJE

Beograd, 2021.

KLAUD RAČUNARSTVO

Autori:

dr Ivana Štrumberger
dr Nebojša Bačanin-Džakula

Recenzenti

dr Mladen Veinović
dr Milan Tuba
dr Miodrag Živković
dr Boško Nikolić

Izdavač:

Univerzitet Singidunum
Danijelova 32
www.singidunum.ac.rs

Za izdavača:
dr Milovan Stanišić

Priprema za štampu:
Miloš Višnjić

Dizajn korica:
Aleksandar Mihajlović, MA

Godina izdanja:
2021.

Tiraž:
1300

Štampa:
Birograf, Beograd

ISBN: 978-86-7912-772-3

Copyright:
© 2021. Univerzitet Singidunum
Izdavač zadržava sva prava.
Reprodukcija pojedinih delova ili celine ove publikacije nije dozvoljena.

Sadržaj

Spisak slika XIII
Lista tabela XX
Listing naredbi XXI
Uvodna razmatranja autora XXIV

DEO I KONVERGIRANA INFRASTRUKTURA I VIRTUELIZACIJA 3

1. TRADICIONALNA *IT* INFRASTRUKTURA 5
    1.1. Klijentski uređaji 9
    1.2. Računarska mreža 12
        1.2.1. *Ethernet* mreža 17
        1.2.2. Bežična mreža 19
        1.2.3. Povezivanje na Internet 21
        1.2.4. VoIP sistem 22
    1.3. Serveri 23
        1.3.1. Računski centar 26
    Pitanja za razmatranje 28

2. KONVERGIRANA I HIPER-KONVERGIRANA *IT* INFRASTRUKTURA 29
    2.1. Kratak uvod u konvergiranu infrastrukturu 30
    2.2. Glavni nedostaci tradicionalne *IT* infrastrukture.....31
        2.2.1. Troškovi tradicionalne infrastrukture.....32
    2.3. Prevazilaženje barijera za implementiranje konvergirane infrastrukture.....34
    2.4. Konvergirana infrastruktura.....35
        2.4.1. Implmentacija mehanizma otpornosti na greške u konvergiranoj infrastrukturi pomoću virtuelizacije servera.....37
    2.5. Hiper-konvergirana infrastruktura.....42
    Pitanja za razmatranje.....47

3. TEHNOLOGIJA VIRTUELIZACIJE....48
3.1. Virtuelizacija servera....51
3.1.1. Hipervizori i glavne tehnike virtuelizacije servera....51
3.1.2. Softveri za virtuelizaciju servera....54
3.1.3. Prednosti virtuelizacije servera kroz konsolidaciju....57
3.2. Virtuelizacija klijenata....58
3.2.1. Prednosti virtuelizacije klijenata....60
3.3. Virtuelizacija mreža....61
3.4. Virtuelizacija skladišta....61
3.5. Prednosti virtuelizacije....63
3.6. Automatizacija....64
Pitanja za razmatranje....65

DEO II KLAUD RAČUNARSTVO....68

4. UVOD U KLAUD RAČUNARSTVO....70
4.1. Definisanje i pojam klaud računarstva....70
4.2. Razvoj klaud računarstva....75
4.2.1. Glavne transformacije u računarstvu....75
4.2.2. Evolucija klaud računarstva....80
4.2.3. Klaud računarstvo vs. virtuelizacija....84
4.2.4. Tržište klaud računarstva danas....85
4.3. Koncept klaud računarstva....86
4.3.1. Prednosti korišćenja koncepta klaud računarstva....89
4.3.2. Oprema u prostorijama kompanije vs. klaud računarstvo....91
Pitanja za razmatranje....93

5. TIPOVI KLAUD RAČUNARSTVA PREMA MODELU USLUGA....94
5.1. Softver kao usluga....96
5.2. Infrastruktura kao usluga....99
5.3. Platforma kao usluga....103
5.4. Razlike između SaaS, IaaS i PaaS....107
5.5. Komunikacije kao usluga....111
5.6. *Desktop* kao usluga....112
5.7. Poslovni proces kao usluga....113
5.8. Skladište podataka kao usluga....115
5.9. Bilo šta kao usluga....117
Pitanja za razmatranje....118

6. TIPOVI KLAUD RAČUNARSTVA PREMA MODELU ISPORUKE....119
6.1. Javni klaud....120
6.2. Privatni klaud....123
6.3. Zajednički klaud....127
6.4. Hibridni klaud....128
6.5. Izbor klaud okruženja....129
Pitanja za razmatranje....133

DEO III MIGRACIJA RESURSA NA KLAUD....137

7. IZVRŠAVANJE MIGRACIJE NA KLAUD....139
7.1. Planiranje procesa migracije....139
7.1.1. Upravljanje promenama....140
7.1.2. Informisanje svih zainteresovanih strana o promenama....141
7.1.3. Definisanje realnog vremenskog okvira za sprovođenje migracije....142
7.1.4. Izrada dokumentacije i praćenje procedura....143
7.1.5. Klaud radni tok....144
7.1.6. Klaud automatizacija....145
7.1.7. Klaud alati i sistemi za upravljanje....146
7.2. Modeli implementacije klaud okruženja....149
7.3. Implementacija i migracija računarske mreže na klaud....153
7.3.1. Klaud mrežni protokoli....153
7.3.2. Konfigurisanje klaud mreže....158
7.3.3. Virtuelne privatne mreže....158
7.3.4. Sistemi za detekciju i sprečavanje upada u klaud sistem....160
7.3.5. Klaud demilitarizovane zone....161
7.3.6. *VxLAN* implementacija na klaudu....163
7.3.7. Adresiranje klaud mreže....164
7.3.8. Segmentacija bloka adresa....165
7.4. Benčmark testovi....166
7.5. Sporazum o nivou korišćenja klaud usluga....167
7.6. Migracija na klaud i uštede u troškovima i potrošenoj energiji....167
Pitanja za razmatranje....169
8. USAGLAŠAVANJE FIZIČKIH RESURSA SA VIRTUELNIM RESURSIMA NA KLAUDU....170
8.1. Raspoloživi hardverski resursi u klaud okruženju....170
8.1.1. Fizički i virtuelni procesori....171
8.1.2. Fizička i virtuelna memorija....171

8.1.3. Upravljanje virtuelnom memorijom....172
8.1.4. Prekoračenje memorije....172
8.1.5. Tehnologija višenitnog izvršavanja....173
8.1.6. Optimizacija iskorišćenosti procesora primenom *Intel-VT* i *AMD-V* tehnologija....175
8.2. Visok nivo dostupnosti klaud resursa i oporavak od katastrofe....176
Pitanja za razmatranje....178

9. KLAUD SKLADIŠTA PODATAKA....179
9.1. Tipovi klaud skladišta podataka....180
9.1.1. *DAS* skladište podataka....180
9.1.2. *NAS* skladište podataka....181
9.1.3. *SAN* skladište podataka....182
9.1.4. Objektno skladište podataka....183
9.2. Dodavanje klaud skladišta....185
9.2.1. Debelo dodavanje skladišta....186
9.2.2. Tanko dodavanje....186
9.3. Prekoračenje skladišta....187
9.4. Migracija sa jedne fizičke mašine na drugu....187
9.5. Šifrovanje podataka....188
9.6. Pristup podacima u klaud skladištu pomoću tokena....189
9.7. Slojevita arhitektura klaud skladišta....190
9.8. Upravljanje i zaštita podataka na klaudu....192
9.8.1. Visoka raspoloživost klaud podataka i otpornost na otkaze....192
9.8.2. Replikacija klaud podataka u i izvan regiona....193
9.8.3. Korišćenje ogledala sajta i kloniranje podataka....194
9.9. *RAID* tehnologija....194
9.9.1. *RAID* 0....196
9.9.2. *RAID* 1....198
9.9.3. *RAID* 5....200
9.9.4. *RAID* 0+1....201
9.9.5. *RAID* 1+0....202
9.9.6. *RAID* 5+0....204
9.9.7. Ostali *RAID* nivoi....205
9.10. Sigurnost klaud skladišta....206
9.10.1. Kontrola pristupa klaud skladištu....207
9.10.2. Zavaravanje....208
9.10.3. *SAN*, zoniranje i *LUN* maskiranje....208

9.11. Pristup klaud skladištu podataka....210
Pitanja za razmatranje....212
10. MIGRACIJA SERVERA NA KLAUD....213
10.1. Vrste migracije servera....214
10.1.1. Migracija sa fizičke mašine na virtuelnu mašinu....215
10.1.2. Migracija sa virtuelne mašine na virtuelnu mašinu....216
10.1.3. Migracija sa fizičke mašine na fizičku mašinu....217
10.1.4. Migracija sa virtuelne mašine na fizičku mašinu....218
10.2. Migracija skladišta podataka....219
10.3. *Online* i *offline* migracija server....220
10.4. Format fajlova virtuelne mašine i virtuelnog diska....221
10.5. Portabilnost aplikacija....222
10.6. Dodatna razmatranja prilikom migracije aplikacija....223
10.7. Infrastruktura za sprovođenje migracije....223
10.7.1. Kapacite računarske mreže....223
10.7.2. Vreme prekida u radu sistema....224
10.7.3. Optimalno vreme za izvršavanje migracije....224
10.7.4. Zakonodavna i legalna pitanja....225
10.7.5. Lokalne vremenske zone....225
Pitanja za razmatranje....226
11. UPRAVLJANJE KORISNICIMA I INFRASTRUKTURNI SERVISI....227
11.1. *RBAC* model....227
11.2. Autentifikacija i autorizacija....229
11.3. Federacije i *SSO* sistemi....230
11.4. Infrastrukturni klaud servisi....232
11.4.1. *DNS* servis....232
11.4.2. *DHCP* servis....233
11.4.3. Servisi za sertifikate....234
11.4.4. Servisi za balansiranje opterećenja....234
11.4.5. Servis za autentifikaciju pomoću više faktora sigurnosti....235
11.4.6. *Firewall* servisi....236
Pitanja za razmatranje....237
12. OPORAVAK OD KATASTROFE I KONTINUITET POSLOVANJA....238
12.1. Odgovornosti i mogućnosti provajdera klaud usluge....240
12.1.1. *RPO* i *RTO*....240
12.1.2. Politike i smernice korporativnog klijenta klaud usluga....241

12.1.3. Politike i smernice provajdera klaud usluga....242
12.1.4. Računarska mreža i oporavak od katastrofe....243
12.1.5. Ograničenja na strani klaud provajdera....244
12.2. Modeli i tehnike za oporavak od katastrofe....244
12.2.1. Kreiranje ogledala sajta....245
12.2.2. Replikacija podataka....249
12.2.3. Transfer fajlova i arhiviranje podataka....251
12.2.4. Ponude treće strane....253
12.3. Kontinuitet poslovanja....254
12.3.1. Uspostavljanje plana kontinuiteta poslovanja....255
12.3.2. Određivanje lokacija za alternativne sajtove....255
12.3.3. Određivanje kontinuiteta poslovnih operacija....256
12.3.4. Računarska mreža i kontinuitet poslovanja....256
12.3.5. Implementacija rubnih sajtova....256
Pitanja za razmatranje....257

DEO IV SIGURNOST U KLAUD OKRUŽENJU....261

13. POJAM, KLASIFIKACIJA I PRIORITIZACIJA RIZIKA NA KLAUDU....263
13.1. Procena rizika....264
13.1.1. Prioritizacija rizika....266
13.1.2. Kvalitativna i kvantitativna procena rizika....267
13.2. Terminologija rizika....269
13.3. Strategije koje se preduzimaju na osnovu identifikovanih i procenjenih rizika....272
13.3.1. Izbegavanje rizika....272
13.3.2. Transfer rizika....273
13.3.3. Ublažavanje rizika....273
13.3.4. Odvraćanje od rizika....274
13.3.5. Prihvatanje rizika....274
Pitanja za razmatranje....275
14. GLAVNI CILJEVI RAČUNARSKE SIGURNOSTI I SIGURNOSNE KONTROLE U KLAUD OKRUŽENJU....276
14.1. Ciljevi računarske sigurnosti....277
14.1.1. Cilj poverljivosti....277
14.1.2. Cilj integriteta....279
14.1.3. Cilj dostupnosti....283

14.1.4. Bezbednost....286
14.1.5. Slojevita sigurnost....287
14.2. Sigurnosne kontrole....288
14.2.1. Klasifikacija sigurnosnih kontrola na osnovu načina implementacije....289
14.2.2. Klasifikacija sigurnosnih kontrola na osnovu ciljeva....292
Pitanja za razmatranje....297

15. USAGLAŠAVANJE SA ZAHTEVIMA KLAUD SIGURNOSTI....298
15.1. Usaglašavanje na strani klaud provajdera....299
15.2. Uspostavljanje sigurnosne politike u organizaciji koja koristi klaud usluge....300
15.3. Izbor i primena sigurnosnih politika za klaud operacije....301
15.4. Primeri regulatornih zahteva....302
Pitanja za razmatranje....304

16. TEHNOLOGIJE KLAUD SIGURNOSTI....305
16.1. Zaštita podataka na klaudu....306
16.2. Tehnologije za zaštitu podataka....307
16.2.1. *IPsec*....307
16.2.2. *SSL*....310
16.2.3. Šifarski algoritmi....311
16.3. Infrastruktura javnog ključa....315
16.4. Sigurnost udaljenog pristupa na klaudu....318
16.4.1. *L2TP* protocol....320
16.4.2. *PPTP* protocol....320
16.4.3. *SSTP* protocol....321
16.4.4. GRE protocol....322
16.5. Automatizacija klaud sigurnosti....322
16.5.1. Alati za automatizaciju na klaudu....323
16.5.2. Pristup alatima za automatizaciju klaud provajdera....325
16.6. Kontrola pristupa na klaudu....327
16.6.1. Pristup klaud objektima....327
16.6.2. Modeli kontrole pristupa u klaud okruženju....389
16.7. Klaud modeli usluge i sigurnost....334
16.8. Klaud modeli isporuke i sigurnost....336
16.9. Implementacija klaud sigurnosti....337
16.9.1. Klasifikacija podataka....337

16.9.2. Segmentacija klauda....339
16.9.3. Segmentacija skladišta....340
16.9.4. Segmentacija resursa koji vrše izračunavanja....340
16.9.5. Implementacija sistema za šifrovanje podataka....341
16.9.6. Sistemi za autentifikaciju pomoću više faktora sigurnosti....342
Pitanja za razmatranje....344

DEO V PRAKTIČNE IMPLEMENTACIJE....349

17. INSTALIRANJE I KONFIGURISANJE VIRTUELNIH MAŠINA....351
17.1. *Oracle VirtualBox*....351
17.2. *VirtualBox* virtuelizacija mreže....355
17.2.1. Mod preslikavanje mrežne adrese....357
17.2.2. Mod spojene mreže....359
17.2.3. Mod interne mreže....360
17.2.4. Mod mreže samo sa *host* računarom....362
17.2.5. Mod preslikavanja mrežne adrese sa prosleđivanjem portova....364
17.3. Implementiranje i konfigurisanje virtuelnih mašina u *Oracle VirtualBox* okruženju....366
17.3.1. Implementacija *gitsrv* virtuelne mašine....367
17.3.2. Implementacija *gitclient* i *gitclient2* virtuelnih mašina....386

18. IMPLEMENTACIJA GIT SISTEMA U VIRTUELNOM OKRUŽENJU....399
18.1. Uvod u *Git* i osnovna terminologija....400
18.1.1. Sistemi za kontrolu verzije....400
18.1.2. Kratka istorija *Git*-a....404
18.1.3. Prednosti *Git*-a....405
18.1.4. Osnovna *Git* terminologija....406
18.2. Implementacija *Git* okruženja....411
18.2.1. Instaliranje *Git* server....413
18.2.2. Instaliranje *Git* klijenta....414
18.2.3. Generisanje *RSA* ključeva i dodavanje udaljenog repozitorijuma....418
18.3. Osnovne Git operacije....424
18.3.1. Operacije *add, commit, push* i *pull*....424
18.3.2. *Git* naredbe za pregled i izmenu promena....432
18.3.3. *Git stash* operacija....437
18.3.4. *Git mv* operacija....440
18.3.5. *Git rm* operacija....444

18.3.6. Ispravljanje grešaka....446
18.3.7. Pomeranje *HEAD* pokazivača primenom *git reset* komande....447
18.4. Rad sa *online* repozitorijumima....450

19. RASPOREĐIVANJE POSLOVA U KLAUD OKRUŽENJU....453
19.1. Algoritmi za raspoređivanje poslova....456
19.2. Pojam optimizacije....457
19.3. Model raspoređivanja resursa sa jednom funkcijom cilja....459
19.4. Model raspoređivanja resursa sa više funkcija cilja....462
Pitanja za razmatranje....465

20. OPTIMIZACIONI ALGORITMI....466
20.1. Metaheuristike....468
20.2. Metaheuristike koje su inspirisane prirodom....471
20.3. Metaheuristike inteligencije rojeva....473
20.4. Algoritam optimizacije jatom kitova....479
Pitanja za razmatranje....483

21. CLOUDSIM PLATFORMA....485
21.1. CloudSim simulacija....488
Pitanja za razmatranje....500

22. SIMULACIJE METAHEURISTIKE INTELIGENCIJE ROJEVA ZA PROBLEM RASPOREĐIVANJA POSLOVA U KLAUD OKRUŽENJU....501
22.1. Prikaz hibridizovanog WOA algoritma....501
22.2. Inicijalizacija populacije i WOA-AEFS pseudo kod....505
22.3. Simulacije sa pravim skupom podataka....507
22.4. Simulacije sa veštačkim skupom podataka....513
Pitanja za razmatranje....533

Literatura....523

Spisak slika

Slika 1-1: Tradicionalna IT infrastruktura....7
Slika 1-2: Mreža ravnopravnih računara....7
Slika 1-3: Klijent-server organizacija....8
Slika 1-4: HP desktop računari različitih faktora oblika....9
Slika 1-5: HP računar tipa sve u jedan....10
Slika 1-6: HP Slate 7 tablet računar....11
Slika 1-7: LAN mreža....13
Slika 1-8: Kampus mreža....14
Slika 1-9: Širokopojasna mreža....14
Slika 1-10: Topologija čvor i linkovi....15
Slika 1-11: Potpuno zamršena topologija....16
Slika 1-12: Potpuno zamršena topologija....16
Slika 1-13: Zaštićeni bakarni kabl (levo) i nezaštićeni (desno)....18
Slika 1-14: Konfiguracija WLAN mreže....20
Slika 2-1: Tradicionalna IT infrastruktura....31
Slika 2-2: Tranzicija od tradicionalnog IT-ja ka konvergiranom okruženju....36
Slika 2-3: Tehnologija uparivanja mrežnih kartica....38
Slika 2-4: Klaster za otpornost na greške....41
Slika 2-5: Klaster za balansiranje opterećenja....41
Slika 2-6: Tradicionalni računski centar....42
Slika 2-7: Konvergirani računski centar....43
Slika 2-8: Hiper-konvergirani računski centar....44
Slika 2-9: Skalabilnost hiper-konvergiranog računskog centra....44
Slika 2-10: Softverski definisani računski centar....46
Slika 3-1: Hipervizor tip 1....52
Slika 3-2: Hipervizor tip 2....53
Slika 3-3: Arhitektura VMware ESXi hipervizora....55
Slika 3-4: Screenshot - vSphere Client interfejs....56
Slika 3-5: Virtuelizacija desktop okruženja....59
Slika 4-1: Koncept klaud računarstva prema Gartner-u....72
Slika 4-2: Glavne transformacije u računarstvu....76
Slika 4-3: Pristup bazi podataka koja se nalazi na mainframe računaru....76
Slika 4-4: Koncept desktop računarstva....77
Slika 4-5: Koncept klijent/server računarstva....78

Slika 4-6: Koncept mrežnog računarstva....79
Slika 4-7: Koncept deljenja vremena....82
Slika 4-8: Pristup sa udaljene lokacije u klaud računarstvu....88
Slika 4-9: Oprema u prostorijama kompanije....91
Slika 4-10: Oprema u prostorijama klaud provajdera....91
Slika 4-11: Virtuelni računski centri dele zajednički fizički hardver....92
Slika 4-12: Konzumiranje raznih klaud usluga od strane korporativnih korisnika....92
Slika 5-1: Tipovi klaud računarstva prema modelu usluga....95
Slika 5-2: Modeli usluge i funkcionalnosti....95
Slika 5-3: SaaS aplikacije....97
Slika 5-4: Pristup klijenata klaud aplikacijama....98
Slika 5-5: Klaud aplikacije kompanije Salesforce....98
Slika 5-6: Osnovne IaaS usluge....100
Slika 5-7: Migracija lokalnog računskog centra na klaud....101
Slika 5-8: Microsoft Azure platforma....102
Slika 5-9: PaaS model....103
Slika 5-10: PaaS alati....104
Slika 5-11: Google App Engine....106
Slika 5-12: Razlikovanje modela klaud računarstva na osnovu korisnika usluga, raspoloživosti usluga i razloga za korišćenje usluga....107
Slika 5-13: Razlika između SaaS, IaaS i PaaS modela na osnovu podele nadležnosti i odgovornosti klijenta i klaud provajdera....108
Slika 5-14: Razlika između SaaS, IaaS i PaaS modela na osnovu tipa korisnika...110
Slika 5-15: Komunikacije kao usluga....111
Slika 5-16: Primena tehnologije virtuelizacije u VDI modelu....112
Slika 5-17: Infrastruktura virtuelnog desktopa....113
Slika 5-18: Poslovni proces kao usluga....114
Slika 5-19: Korišćenje klaud skladišta za potrebe korporativnih aplikacija i servisa....115
Slika 5-20: Klaud skladište podataka....116
Slika 5-21: Bilo šta kao usluga....117
Slika 6-1: Tipovi klaud računarstva prema modelu isporuke....120
Slika 6-2: Primer korišćenja servisa javnog klauda....120
Slika 6-3: Pristup infrastrukturi javnog klauda....122
Slika 6-4: Model privatnog klauda....123
Slika 6-5: Tipovi okruženja privatnog klauda....124
Slika 6-6: Microsoft privatni klaud....126
Slika 6-7: Zajednički klaud....127
Slika 6-8: Hibridni klaud....129
Slika 6-9: Slučajevi korišćenja klaud modela....132

Slika 7-1: Sistemi za upravljanje mrežom na klaudu....147
Slika 7-2: Glavne funkcije alata za upravljanje klaud mrežom....147
Slika 7-3: Zajedničke usluge javnog klauda....150
Slika 7-4: Model implementacije privatnog klauda....150
Slika 7-5: Model implementacije zajedničkog klauda....151
Slika 7-6: Povezivanje privatnog i javnog klauda jedne organizacije....152
Slika 7-7: ISO/OSI referentni model i TCP/IP stek protokola....154
Slika 7-8: Prikaz komunikacije između lokalnog klijenta i klaud Web servera..155
Slika 7-9: Pristup klaud servisima sa udaljene lokacije korišćenjem VPN servisa....159
Slika 7-10: Povezivanje privatnih klaud okruženja preko VPN koncentratora..159
Slika 7-11: Primer upotrebe IDS uređaja u klaud okruženju....160
Slika 7-12: Primer upotrebe IPS uređaja na klaudu....161
Slika 7-13: Primer DMZ konfiguracija na klaudu....162
Slika 7-14: Proces VxLAN enkapsulacije....164
Slika 8-1: Koncept prekoračenja memorije....173
Slika 8-2: Screenshot - Windows servisa za merenje performansi sistema za i7 procesor sedme generacije koji podržava višenitno izvršavanje....174
Slika 8-3: Radni režim aktivan/pasivan Web servera....176
Slika 9-1: DAS skladište podataka....181
Slika 9-2: NAS skladište podataka....182
Slika 9-3: SAN skladište podataka....183
Slika 9-4: Slojevita arhitektura klaud skladišta....191
Slika 9-5: RAID 0 nivo povezivanja diskova....197
Slika 9-6: RAID 0 tehnologija....197
Slika 9-7: RAID 1 nivo povezivanja diskova....199
Slika 9-8: RAID 1 tehnologija....199
Slika 9-9: RAID 5 tehnologija sa tri diska....201
Slika 9-10: RAID 5 tehnologija sa četiri diska....201
Slika 9-11: RAID 0+1 nivo....202
Slika 9-12: RAID 1+0 nivo....203
Slika 9-13: RAID 5+0 implementacija....204
Slika 9-14: RAID 6 nivo....205
Slika 9-15: Uporedni prikaz RAID nivoa....206
Slika 9-16: SAN zoniranje....209
Slika 10-1: Tip migracije P2V....215
Slika 10-2: Tip migracije V2V....216
Slika 10-3: Tip migracije P2P....217
Slika 10-4: Tip migracije V2P....218

Slika 10-5: Migracija skladišta podataka na klaud....220
Slika 11-1: Klaud DNS servis....232
Slika 11-2: Klaud DHCP servis....233
Slika 11-3: Sistemi za balansiranje opterećenja na klaudu....235
Slika 11-4: Klaud firewall između javnog Interneta i privatne korporativne mreže....236
Slika 11-5: Klaud firewall između javnog Interneta i klaud mreže....236
Slika 12-1: Implementacija jednog mehanizma za oporavak od katastrofe..239
Slika 12-2: Model vrućeg sajta....246
Slika 12-3: Arhitektura mlakog sajta....247
Slika 12-4: Model hladnog sajta....248
Slika 12-5: Proces replikacije podataka na klaudu....249
Slika 12-6: Proces sinhrone replikacije na klaudu....250
Slika 12-7: Proces asinhrone replikacije na klaudu....251
Slika 14-1: Piramida slojevite sigurnosti....288
Slika 16-1: Prikaz IPsec tunela....309
Slika 16-2: Simetrični šifarski algoritam....312
Slika 16-3: Asimetrični šifarski algoritam....313
Slika 16-4: Pojednostavljena arhitektura PKI sistem....316
Slika 16-5: Prikaz jedne VPN implementacije....319
Slika 16-6: Screenshot - Oracle Integration Cloud dashboard....326
Slika 16-7: IaaS model sigurnosti....334
Slika 16-8: PaaS model sigurnosti....335
Slika 16-9: SaaS model sigurnosti....335
Slika 16-10: Klasifikacija podataka na klaud skladištu....338
Slika 16-11: Hardverski token....342
Slika 16-12: Login forma za autentifikaciju pomoću hardverskih tokena....343
Slika 16-13: Google Authenticator softverski token....343
Slika 17-1: NAT mrežni mod....358
Slika 17-2: Mod spojene mreže....359
Slika 17-3: Mod interne mreže....361
Slika 17-4: Mod mreže samo sa host računarom....363
Slika 17-5: Screenshot - Oracle VirtualBox Host Network Manager interfejs....363
Slika 17-6: Screenshot - Oracle VirtualBox Port Forwarding Rules interfejs....366
Slika 17-7: Screenshot - pokretanje wizarda za kreiranje nove virtuelne mašine....367
Slika 17-8: Screenshot - izbor tipa i verzije operativnog sistema....368
Slika 17-9: Screenshot - definisanje količine RAM memorije....369
Slika 17-10: Screenshot - kreiranje virtuelnog diska (1)....370
Slika 17-11: Screenshot - kreiranje virtuelnog diska (2)....370

Slika 17-12: Screenshot - kreiranje virtuelnog diska (3).....371
Slika 17-13: Screenshot - kreiranje virtuelnog diska (4).....371
Slika 17-14: Screenshot - Oracle VM VirutalBox Manager.....372
Slika 17-15: Screenshot - prozor sa podešavanjima virtuelne mašine.....372
Slika 17-16: Screenshot - izbor virtuelnog instalacionog medijuma (ISO fajl)..373
Slika 17-17: Screenshot - prozor sa podešavanjima mrežnih adaptera 374
Slika 17-18: Screenshot - grupa opcija za podešavanje prvog mrežnog adaptera.....375
Slika 17-19: Screenshot - podešavanje pravila za prosleđivanje portova na prvom mrežnom adapteru.....375
Slika 17-20: Screenshot - podešavanje drugog mrežnog adaptera.....376
Slika 17-21: Screenshot - pokretanje virtuelne mašine.....377
Slika 17-22: Screenshot - pokretanje instalacije operativnog sistema Ubuntu Server.....377
Slika 17-23: Screenshot - Ubuntu Server instalacija (mrežni adapter).....378
Slika 17-24: Screenshot - Ubuntu Server instalacija (Profile setup).....379
Slika 17-25: Screenshot - Logovanje na Linux bash.....379
Slika 17-26: Screenshot - VirtualBox Guest Additions instalacioni medijum....380
Slika 17-27: Screenshot - pristup virtuelnoj mašini gitsrv preko PuTTY klijenta....385
Slika 17-28: Screenshot - kreiranje virtuelne mašine gitclient.....386
Slika 17-29: Screenshot - definisanje pravila za prosleđivanje portova mašine gitclient.....387
Slika 17-30: Screenshot - Ubuntu Desktop Show Applications ikonica.....388
Slika 17-31: Screenshot - Ubuntu Desktop Terminal ikonica.....388
Slika 17-32: Screenshot - VirtualBox Guest Additions instalacioni medijum....389
Slika 17-33: Screenshot - pokretanje procesa instalacije VirtualBox Guest Additions.....390
Slika 17-34: Screenshot - omogućavanje funkcionalnosti VirtualBox Guest Additions.....390
Slika 17-35: Screenshot - kloniranje virtuelne mašine gitclient.....391
Slika 17-36: Screenshot - kloniranje virtuelne mašine gitclient (opcija Full cone).....392
Slika 17-37: Screenshot - konfigurisanje pravila za prosleđivanje portova na gitclient2.....392
Slika 17-38: Screenshot - pristup mašini gitclient preko PuTTY klijenta.....397
Slika 18-1: Arhitektura LVCS softvera.....401
Slika 18-2: Arhitektura CVCS softvera.....402
Slika 18-3: Arhitektura DVCS softvera.....403
Slika 18-4: Git proces rada.....407
Slika 18-5: Ilustracija Git operacija add, commit, push i pull.....424

Slika 18-6: Resetovanje HEAD pokazivača....449
Slika 18-7: Screenshot – Online GitHub repozitorijum....450
Slika 18-8: Screenshot – Online GitHub repozitorijum (sinhronizacija)....452
Slika 19-1: Model sistema raspoređivanja poslova u klaud okruženju....454
Slika 21-1: Dijagram klasa i arhitekture CloudSim-a....486
Slika 21-2: Provera Java Development Kit....489
Slika 21-3: Kreiranje novog projekta....489
Slika 21-4: Java Project....490
Slika 21-5: Koraci kreiranja Java projekta....490
Slika 21-6: Preuzimanje biblioteke....491
Slika 21-7: Konfigurisanje putanje....491
Slika 21-8: Dodavanje eksternih JAR file-ova....492
Slika 21-9: Selektovanje .jar file-ova....492
Slika 21-10: Referenced Libraries....493
Slika 21-11: CloudSim projekat....493
Slika 21-12: Kreiranje paketa....494
Slika 21-13: Kreiranje naziva paketa....494
Slika 21-14: CloudSim putanja do primera....495
Slika 21-15: Primeri klaud okruženja....495
Slika 21-16: Selektovanje primera....496
Slika 21-17: Dodavanje primera....496
Slika 21-18: CloudSim primer 1 linije koda....497
Slika 21-19: Pokretanje koda....497
Slika 21-20: Izvršavanje primera 1 CloudSim....498
Slika 21-21: Linije koda primer 2 CloudSim....498
Slika 21-22: Izvršenje primera 2 CloudSim....499
Slika 22-1: Konfiguracija host računara u CloudSim-u u simulacijama WOA-AEFS metaheuristike sa pravim skupom podataka....508
Slika 22-2: Konfiguracija virtuelnih mašina u CloudSim-u u simulacijama WOA-AEFS metaheuristike sa pravi skupom podataka....508
Slika 22-3: Komparativna analiza WOA-AEFS vs. WOA korišćenjem NASA iPSC realnih podataka za mali i veliki broj poslova....509
Slika 22-4: Grafikon brzine konvergencije WOA-AEFS vs. WOA korišćenjem NASA iPSCrealnih podataka za 200 zahteva....510
Slika 22-5: Komparativna analiza WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih podataka sa manjim brojem poslova....501
Slika 22-6: Komparativna analiza WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih podataka sa većim brojem poslova....511

Slika 22-7: Poboljšanja WOA-AEFS algoritma u odnosu na druge algoritme u simulacijama sa NASA iPSC realnim podacima.....511
Slika 22-8: Komparativna analiza DI indikatora u simulacijama sa NASA iPSC realnim skupom podataka.....512
Slika 22-9: Komparativna analiza CPU vremena u simulacijama sa NASA iPSC realnim skupom podataka.....513
Slika 22-10: Karakteristike virtuelnih mašina, host računara I data centara u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam).....514
Slika 22-11: Karakteristike poslova u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam).....514
Slika 22-12: Vrednosti kontrolnih parametara WOA-AEFS metaheuristike.....515
Slika 22-13: Komparativna analiza makespan funkcije cilja-WOA-AEFS vs. druge tehnike korišćenjem veštačkog skupa podataka (stopa dolaska 10).....516
Slika 22-14: Komparativna analiza makespan funkcije cilja-WOA-AEFS vs. druge algoritme korišćenjem veštačkog skupa podataka (stopa pristizanja 40).....517
Slika 22-15: Komparativna analiza troškova sa 200 zahteva i promenljivim rokom izvršavanja zadataka-WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka.....518
Slika 22-16: Komparativna analiza troškova sa 400 zahteva i promenljivim rokom zvršavanja zadataka-WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka.....518
Slika 22-17: Komparativna analiza troškova sa 500 zahteva i promenljivim rokom izvršavanja zadataka-WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog kupa podataka.....519
Slika 22-18: Komparativna analiza narušavanja roka sa 200 zahteva-WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 520
Slika 22-19: Komparativna analiza narušavanja roka sa 400 zahteva-WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 520
Slika 22-20: Komparativna analiza narušavanja roka sa 500 zahteva-WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka 521

Lista tabela

Tabela 1-1: Standardi za bakarni kabl....18
Tabela 1-2: IEEE 802.11 standardi....19
Tabela 1-3: Protokoli za deljenje fajlova....24
Tabela 2-1: Vrste uparivanja mrežnih adaptera....39
Tabela 5-1: Razlike između SaaS, IaaS i PaaS modela na osnovu načina isporuke....109
Tabela 5-2: Razlike između SaaS, IaaS i PaaS modela na osnovu scenarija korišćenja....109
Tabela 7-1: Pregled protokola....157
Tabela 7-2: Blokovi privatnih IP adresa....165
Tabela 11-1: Korisničke uloge na klaudu....228
Tabela 13-1: Preporučena lista vrednosti verovatnoće dešavanja rizika....269
Tabela 18-1: Specifikacije Git servera....411
Tabela 18-2: Specifikacije prvog Git klijenta....412
Tabela 18-3: Specifikacije drugog Git klijenta....412
Tabela 18-4: Git promenljive okruženja....415

Listing naredbi

Listing 17-1: Montiranje instalacionog medijuma VirtualBox Guest Additions..381
Listing 17-2: Sadržaj foldera /mnt.....381
Listing 17-3: Instaliranje biblioteka za pokretanje VirutalBox Guest Additions...381
Listing 17-4: Pokretanje instalacije VirutalBox Guest Additions.....381
Listing 17-5: Izvršavanje bash komande ifconfig.....382
Listing 17-6: Izmena fajla etc/netplan/50-cloud-init.yaml.....383
Listing 17-7: IP konfiguracija mrežnih adaptera nakon izmene.....384
Listing 17-8: SSH daemon.....384
Listing 17-9: Instalacija biblioteka za VirutalBox Guest Additions.....389
Listing 17-10: Promena naziva klonirane mašine na nivou operativnog sistema.....393
Listing 17-11: Izmene fajla /etc/hosts.....393
Listing 17-12: Kreiranje korisnika ivana i dodavanje korisnika u sudo grupu.....394
Listing 17-13: Izvršavanje ifconfig komande na mašini gitclient.....394
Listing 17-14: Izmene fajla /etc/netplan/01-network-manager-all.yaml.....395
Listing 17-15: IP konfiguracija mrežnih adaptera mašine gitclient nakon izmena.....396
Listing 17-16: Rezultat izvršavanja ping komande.....396
Listing 17-17: Instaliranje SSH servera.....397
Listing 17-18: Rezultat izvršavanja ifconfig naredbe (gitclient2).....398
Listing 18-1: Primeri add i commit operacija.....408
Listing 18-2: Primer čitanja vrednosti HEAD pokazivača.....410
Listing 18-3: Primer čitanja config fajla.....410
Listing 18-4: Instaliranje Git serverske komponente.....413
Listing 18-5: Inicijalizovanje udaljenog Git repozitorijuma.....414
Listing 18-6: Instaliranje Git-a na mašini gitclient.....414
Listing 18-7: Podešavanje vrednosti Git varijabli okruženja na mašini gitclient....416
Listing 18-8: Podešavanje vrednosti Git varijabli okruženja na mašini gitclient2.....416
Listing 18-9: Inicijalizovanje lokalnog Git repozitorujuma na mašini gitclient....417
Listing 18-10: Inicijalizovanje lokalnog Git repozitorujuma na mašini gitclient2..417
Listing 18-11: Generisanje RSA ključeva korisnika nebojsa na mašini gitclient..418
Listing 18-12: Generisanje RSA ključeva korisnika ivana na mašini gitclient2....419

Listing 18-13: Kopiranje javnog RSA ključa sa mašine gitclient na mašinu gitsrv....420
Listing 18-14: Uspostavljanje SSH konekcije sa mašine gitclient ka mašini gitsrv....420
Listing 18-15: Kopiranje javnog RSA ključa sa mašine gitclient2 na mašinu gitsrv....421
Listing 18-16: Dodavanje udaljenog Git repozitorijuma na mašinu gitclient...421
Listing 18-17: Kreiranje README fajla i dodavanje na udaljeni repozitorijum....422
Listing 18-18: Kloniranje udaljenog Git repozitorijuma....423
Listing 18-19: Korak 1 – Git operacije add, commit, push i pull....425
Listing 18-20: Korak 2 – Git operacije add, commit, push i pull....426
Listing 18-21: Korak 3 – Git operacije add, commit, push i pull....427
Listing 18-22: Korak 4 – Git operacije add, commit, push i pull....427
Listing 18-23: Korak 5 – Git operacije add, commit, push i pull....428
Listing 18-24: Korak 6 – Git operacije add, commit, push i pull....429
Listing 18-25: Korak 7 – Git operacije add, commit, push i pull (1)....429
Listing 18-26: Korak 7 – Git operacije add, commit, push i pull (2)....430
Listing 18-27: Korak 8 – Git operacije add, commit, push i pull....431
Listing 18-28: Korak 9 – Git operacije add, commit, push i pull....432
Listing 18-29: Izmene fajla Hello.java....432
Listing 18-30: Dodavanje izmena fajla Hello.java u faznu oblast....433
Listing 18-31: Git show i git log naredbe....434
Listing 18-32: Izvršavanje git diff naredbe....435
Listing 18-33: Izvršavanje operacije git commit sa amend argumentom....436
Listing 18-34: Snimanje promena iz repozitorijuma mašine gitclient2 na udaljeni repozitorijum....436
Listing 18-35: Sinhronizacija lokalnog repozitorijuma mašine gitclient sa udaljenim repozitorijumom....437
Listing 18-36: Izvršavanje git stash operacije....438
Listing 18-37: Git stash pop naredba....439
Listing 18-38: Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 sa udaljenim repozitorijumom (git stash)....439
Listing 18-39: Demonstriranje git mv operacije za premeštanje fajlova....440
Listing 18-40: Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 saudaljenim repozitorijumom (git mv operacija za premeštanje fajlova)....441
Listing 18-41: Demonstriranje git mv operacije za preimenovanje fajlova....442
Listing 18-42: Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 sa udaljenim repozitorijumom (git mv operacija za preimenovanje fajlova)....443

Listing 18-43: Demonstriranje git rm operacije za brisanje fajlova.....444

Listing 18-44: Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 saudaljenim repozitorijumom (git rm operacija za brisanje fajlova).....445
Listing 18-45: Prikaz komande git checkout za vraćanje fajla na prethodnu verziju.....446
Listing 18-46: Prikaz komande git checkout za povraćaj obranisanih fajlova..446
Listing 18-47: Poništavanje promene nad fajlovima koji su dodati u faznu oblast.....447
Listing 18-48: Prikaz commit ID poslednje promene.....448
Listing 18-49: Primeri upotrebe komandi git reset i git log.....448
Listing 18-50: Kloniranje udaljenog GitHub repozitorijuma.....451
Listing 18-51: Sinhronizacija sa GitHub repozitorijumom.....451

Uvodna razmatranja autora

Udžbenik „Klaud računarstvo“ namenjen je pre svega studentima Fakulteta za informatiku i računarstvo i Tehničkog fakulteta Univerziteta Singidunum kao pomoć u savladavanju silabusa kurseva iz oblasti klaud računarstva, koji se slušaju na studijskim modulma (smerovima) Informacione tehnologije, Informatika i računarstvo i Softversko inženjerstvo.

Međutim, ovaj udžbenik je dobar izvor informacija i za sve ostale koji imaju nameru da se upoznaju sa ovom aktuelnom tehnologijom, njenim funkcionalnim principima i primenom.

Mnogi termini i pojmovi, kako iz oblasti računarstva, tako i iz klaud računarstva (eng. *cloud computing*) nastali su iz engleskog jezika. Pretragom raspoložive stručne literature iz ovog domena na srpskom jeziku vidi se da su neki autori engleske termine i pojmove prevodili na srpski jezik, dok su drugi koristili izvorne nazive pojmova na engleskom jeziku. S obzirom da u pojedinim slučajevima ne postoji odgovarajući prevod engleskih termina na srpski jezik, kao i da prevod određenih termina zvuči rogobatno i neprirodno, u ovom udžbeniku, u pojedinim slučajevima, korišćeni su engleski termini, ili engleske skraćenice, koji su u tekstu formatirani kurzivom. Osim toga, nazivi pojmova na engleskom jeziku korišćeni su i na onim mestima gde je njihova primena široko prihvaćena u srpskom jeziku.

U svetu računarstva popularizovani su termini informacione tehnologije (eng. *Information Technology – IT*) i informaciono-komunikacione tehnologije (eng. *Information Communication Technology – ICT*). Međutim, s obzirom da su informacione i komunikacione tehnologije konvergirale (danas teško može da se napravi razlika između komunikacionih i informacionih tehnologija, kao što na primer pametni telefon spada u grupu informacionih, ali i komunikacionih tehnologija), u ovom udžbeniku korišćen je termin *IT*. S druge strane, termin računarske tehnologije se takođe koristi, kako u stranoj, tako i u domaćoj literaturi i već duži niz godina u stručnoj javnosti vodi se rasprava o tome da li treba koristiti termin *IT* ili računarske tehnologije.

Bez obzira što između ova dva pojma postoji razlika, kako bi autori ostali neutralni, u ovom udžbeniku termini *IT* i računarske tehnologije korišćeni su kao sinonimi. Tako na primer za ukazivanje na skup informacionih ili računarskih resursa korišćen je termin računarski resursi, ali i termin informacioni resursi.

Oblast računarstva (eng. *Computer Science*) obuhvata veliki broj domena, a klaud računarstvo je samo jedna tehnologija koja pripada bogatom svetu računarstva. Prema jednom viđenju, a ima ih mnogo, pojam klaud računarstvo odnosi se na isporuku računarskih resursa na zahtev preko Interneta, ili neke druge računarske mreže korišćenjem modela plaćanja na osnovu nivoa ostvarene potrošnje.

Glavni cilj ovog udžbenika je upoznavanje čitalaca sa osnovama klaud računarstva. Tekst je propraćen primerima i praktičnim simulacijama, koji imaju za cilj da čitaoce što više približe ovoj aktuelnoj oblasti, koja će svakako u budućnosti da ima još veći uticaj, kako na razvoj računarstva, tako i na sve druge domene iz oblasti ekonomije, finansija, biznisa, itd. Primeri su koncipirani da budu jednostavni, ali su u isto vreme realni i interesantni.

Praktične implementacije i simulacije realnog klaud i virtuelizovanog okruženja date su u petom, poslednjem delu udžbenika. Primarni cilj ovih simulacija je da osposobe studente i druge čitaoce za samostalni rad na implementiranju servisa klaud računarstva uz pomoć solidno postavljene osnove.

Kao dodatak drugog izdanja ovog udžbenika autori su obuhvatili i nekoliko problema i izazova koje treba rešiti u klaud okruženju, kao i prikaz mogućih tehnika, metoda i algoritama za savladavanje prepreka u komunikaciji i razmeni zahteva, resursa i paketa sa i u okviru klaud sistema.

Obzirom da je jedan od najčešćih problema u klaud računarstvu izazov raspoređivanja poslova i balansiranja opterećenja, u udžbeniku je poseban akcenat dat na prikazu metoda za njegovo rešavanje. Bez obzira što je veliki broj tehnika i metoda za prevazilaženje ovog problema dostupan u savremenoj literaturi, u užbeniku je posebno istaknuto njegovo rešavanje pomoću metaheuristika inteligencije rojeva, koje spadaju u domen prirodom inspirisanih algoritama. Ovakva strategija je implikacija dve činjenice: prvo, metaheuristike rojeva su se pokazale kao veoma efikasne metode i rešavanju problema raspoređivanja poslova i balansiranja opterećenja u klaud okruženju i drugo, autori imaju bogato naučno i praktično iskustvo u radu sa ovom grupom optimizacionih metoda.

Problem raspoređivanja na klaudu odnosi se na definisanje rasporeda izvršavanja zadataka na ograničenom skupu raspoloživih resursa uzimajući pritom u obzir potencijalna ograničenja i funkciju cilja koju je potrebno optimizovati.

Za potrebe testiranja, prikazano je rešavanje dve instance (modela) raspoređivanja poslova na klaudu primenom metaheuristike inteligencije rojeva. Prvi model, koji pripada grupi jednokriterijumske optimizacije, uzima u obzir minimizaciju vremena izvršavanja svih zadataka na klaudu, dok drugi, višekriterijumski model, uzima u obzir minimizaciju vremena izvršavanja svih zadataka na klaudu i budžeta, tj. troškova za izvršavanje zahteva krajnjih korisnika.

Simulacije (testiranja) su izvršene u okruženju CloudSim simulatora i algoritam je testiran sa skupom veštačkih podataka, generisanih u okviru CloudSim platforme, i realnih podataka, koji su preuzeti iz globalno dostupne benčmark baze.

Udžbenik obiluje slikama, zato što autori smatraju da primenom vizuelnih tehnika učenja, studenti lakše mogu da savladaju priloženo gradivo.

Udžbenik se sastoji iz pet delova, gde se svaki deo sastoji iz nekoliko glava. Svaki deo predstavlja logičku celinu, koja je neophodna za izgradnju šire slike klaud računarstva. Izlaganje materijala u svakom delu vrši se od poznatog ka nepoznatog i od konkretnog ka apstraktnom.

Na kraju svake glave data je lista pitanja. Preporučuje se da studenti, nakon čitanja svake glave, pažljivo pročitaju pitanja i da razmisle o odgovorima. Ovakav način rada pomoći će im u efikasnijem savladavanju priloženog materijala.

Važnije definicije i pojmovi istaknuti su tako što su uokvireni plavim dijalogom za tekst.

Prema uobičajenoj praksi pisanja udžbenika, knjiga i naučnih radova iz oblasti računarstva, za prikaz programskih naredbi korišćen je font sa znakovima iste širine (eng. *monospaced font*), gde su logičke celine naredbi prikazane u posebnom okviru za tekst u formi isečka koda (eng. *code snippet*).

Zahvalnost: veliku zahvalnost dugujemo svima koji su pročitali delove ovog udžbenika i koji su svojim sugestijama i smernicama doprineli da završna verzija bude bolja.

Mišljenja i sugestije: drugo izdanje udžbenika „Klaud računarstvo“ predviđeno je studentima i drugim čitaocima koje ova oblast interesuje. Za bilo kakva zapažanja o knjizi, kao i pitanja možete da se obratite putem email na istrumberger@singidunum.ac.rs ili nbacanin@singidunum.ac.rs.

septembar, 2021.

dr Ivana Štrumberger

dr Nebojša Bačanin-Džakula



KONVERGIRANA INFRASTRUKTURA I VIRTUELIZACIJA

DEO I

KONVERGIRANA INFRASTRUKTURA I VIRTUELIZACIJA



U prvom delu udžbenika predstavljeni su koncepti konvergirane infrastrukture, hiper-konvergirane infrastrukture i tehnologije virtuelizacije kao vodećih tehnologija za razvoj i implementiranje servisa klaud računarstva.

Većina organizacija svoje potrebe za računarskim resursima zadovoljava korišćenjem tradicionalne troslojne računarske arhitekture, koja se sastoji od nezavisnih jedinica za izračunavanje, skladišnih i mrežnih jedinica. Ovakva arhitektura još uvek predstavlja izbor većine korporativnih entiteta.

Međutim, u uslovima savremenog, turbulentnog okruženja, kada su zahtevi za promenama češći i intenzivniji, ova tehnologija je skupa za izgradnju, teška za upravljanje i nezavidnih mogućnosti skaliranja. Tradicionalna troslojna arhitektura ne može da odgovori dovoljno brzo na zahteve poslovnog okruženja u kome korporacija posluje i onemogućava implementiranje proaktivne strategije konkurentnosti.

Nedostatak agilnosti tradicionalne troslojne infrastrukture doveo je do toga da se mnoge organizacije okreću javnom klaudu kako bi zadovoljile svoje informacione potrebe. Međutim, korišćenjem servisa javnog klauda, organizacije se plaše gubitka kontrole nad svojim računarskim resursima, nekontrolisanog rasta troškova iznajmljivanja resursa i sigurnosti svojih vrednih podataka, koji se nalaze na skladištima klaud provajdera.

Na sreću, razvoj modernog hardvera doveo je do zastarevanja tradicionalne troslojne platforme, tako što su serveri, skladišta podataka i mreže uključene u jedan *x86* server u rešenjima konvergirane i hiper-konvergirane infrastrukture. Ovakva rešenja inkorporiraju virtuelne komponente primenom tehnologije virtuelizacije i kreiranju softverski definisane računske centre.

Konvergirana i hiper-konvergirana infrastruktura i tehnologija virtuelizacije predstavljaju vodeće tehnologije u razvoju klaud računarstva. Primenom ovih tehnologija kompanije mogu na relativno lak i troškovno efikasan način da implementiraju svoje privatno klaud okruženje.

S druge strane, klaud provajderi koriste konvergirane tehnologije za razvoj javnog i hibridnog klaud okruženja i mogu da ponude svoje usluge individualnim i korporativnim klijentima po značajno nižim troškovima.

Zbog toga što se kombinovanjem konvergirane i hiper-konvergirane infrastrukture sa tehnologijim virtuelizacije kreira efikasna razvojna platforma za implementiranje servisa klaud računarstva i s pogledom na činjenicu da ove tehnologije imaju suštinski uticaj na razvoj klaud računarstva u celini, ceo jedan deo udžbenika je posvećen ovim tehnologijama.

Međutim, za potpunije i sveobuhvatnije razumevanje navednih koncepata, potrebno je razumevanje tradicionalne infrastrukture, i zbog toga priča u ovom delu udžbenika počinje odatle.

1. TRADICIONALNA IT INFRASTRUKTURA

Većina poslovnih sistema danas koristi informacione tehnologije (eng. *Information Technology – IT*) kako bi zadovoljila brojne poslovne potrebe.

Među najznačajnijim poslovnim potrebama koje *IT* zadovoljava izdvajaju se sledeće:

- povećanje efikasnosti obavljanja poslovnih procesa,
- poboljšanje komunikacije sa klijentima i
- proširenje dometa poslovanja.

Način na koji se *IT* primenjuje u poslovanju zavisi od prirode poslova, industrijskog segmenta i korporativne kulture. *IT* poslovni analitičar (eng. *IT Business Analyst – IT BA*) približava *IT* poslovanju organizacionih jedinica (eng. *Organizational Units – OU*) kako bi se stvorili preduslovi za generisanje boljih poslovnih rezultata. *IT* poslovni analitičar je pojedinac u organizaciji koji ima ulogu posrednika između *IT*-ja i poslovnih korisnika. Ovaj pojedinac poznaje i *IT* i poslovne procese i koristi svoje znanje kako bi preporučio dobra rešenja svakodnevnih (operativnih) i strateških problema. *IT* poslovni analitičar prepoznaje poslovne potrebe i radi zajedno sa *IT* odeljenjem kako bi se transformisale identifikovane potrebe u dobra poslovna rešenja koja se baziraju na tehnologiji.

Savremeni poslovni sistemi moraju brzo da se razvijaju da bi ostvarili i održavali konkurentsku prednost. Nažalost, u mnogim slučajevima IT predstavlja barijeru inovativnih aktivnosti i umesto da bude pokretač, on često koči proces inovacija. U mnogim organizacijama sa tradicionalnom organizacijom *IT* infrastrukture, *IT* operacije se izvode u okviru nezavisnih „*IT* silosa" (eng. *IT silos*). Implementiranje promena u takvim okruženjima uglavnom zahteva proces odobravanja koji se sprovodi na više nivoa, značajnu kapitalnu investiciju i izražene aktivnosti koordinacije između *IT* silosa.

> *IT* silos može da se definiše kao skup *IT* resursa koji poseduje i kontroliše određena organizaciona jedinica, ili odeljenje u organizaciji.

Još jedan problem koji se javlja kada su resursi poput računara, mreža i skladišta (eng. *storage*) organizovani u forme *IT* silosa jeste taj što su kapaciteti nekih resursa previše korišćeni, dok su drugi nedovoljno korišćeni u vidu viška procesorske snage, memorijskog i skladišnog kapaciteta.

U organizacijama sa tradicionalnom organizacijom *IT* resursa, *IT* infrastruktura je u početku mala, a potom raste tako što se dodaju *IT* resursi kada za to nastane potreba. Ovaj fenomen je u literaturi poznat pod nazivom *IT* bujanje (eng. *IT sprawl*). Fenomen *IT* bujanja navodi se kao glavni uzrok brojnih neefikasnosti, između ostalog i sledećih:

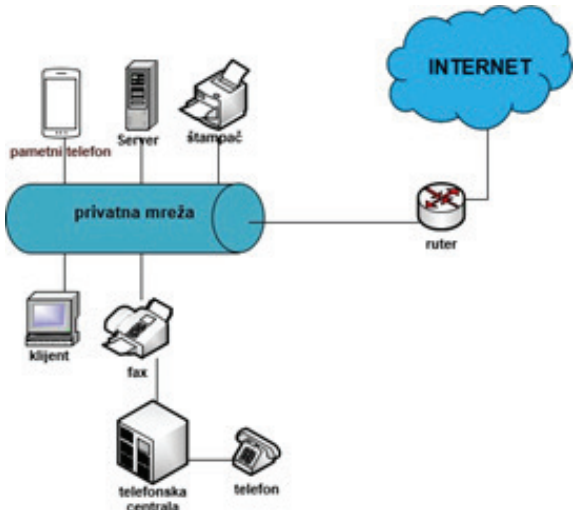
- resursi se nedovoljno ili previše koriste,
- preveliko opterećenje menadžmenta u donošenju upravljačkih odluka vezanih za korišćenje resursa,
- povećana potrošnja električne energije i
- povećani zahtevi za rashladnim sistemima računarskih komponenti i opreme.

Navedene neefikasnosti povećavaju ukupne troškova posedovanja opreme (eng. *Total Cost of Ownership – TCO*) i povećavaju troškove u vidu novca i vremena za izvođenje svakodnevnih aktivnosti održavanja opreme, što ima za posledicu da se sve manje novca i vremena koristi za implementiranje novih tehnoloških rešenja koja omogućavaju poslovnom sistemu da raste i da jača i unapređuje konkurentnu prednost. Ukupni troškovi posedovanja računarske opreme predstavljaju zbirne troškove kupovine, implementacije i održavanja opreme ili *IT* rešenja. Ovi troškovi uključuju i troškove električne energije, okruženja gde je implementirana *IT* oprema, podrške, obuke, kao i mnoge druge. Na osnovu izvedenih studija, 70% *IT* budžeta koristi se za održanje postojeće opreme, dok se samo 30% koristi za implementaciju novih tehnoloških rešenja i platformi.

Tradicionalno *IT* okruženje je okruženje u kom se serveri i mrežna infrastruktura nalaze isključivo u prostorijama kompanije (eng. *on-premises*). Grafički prikaz tradicionalne infrastrukture dat je na Slici 1-1.

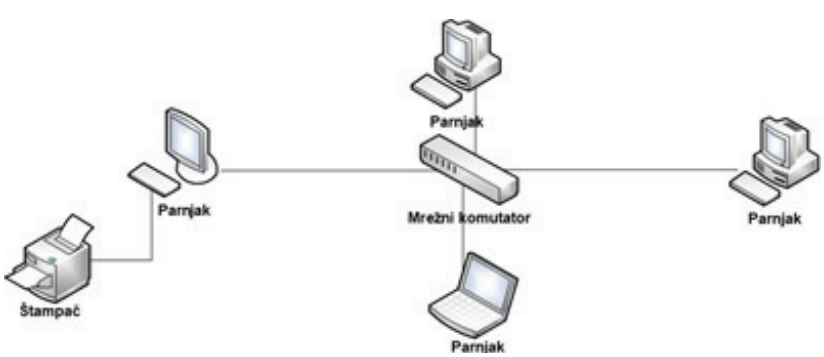
U većini savremenih okruženja, tradicionalna infrastruktura u najmanju ruku obuhvata privatnu mrežu koja se koristi u kompaniji i konekciju ka Internetu. Telefonska *IP* (*Internet Protocol*) centrala (eng. *Private Branch Exchange – PBX*) je u najvećem broju slučajeva implementirana kao zasebna mreža, dok klasična telefonija može da se integriše u računarsku mrežu. Računarska mreža se definiše kao skup povezanih uređaja i medijuma preko kojih uređaji međusobno komuniciraju. Telefonska *IP* centrala je uređaj koji omogućava da više korisnika koristi jednu ili više linija za uspostavljanje telefonskih poziva.

Slika 1-1: *Tradicionalna IT infrastruktura*



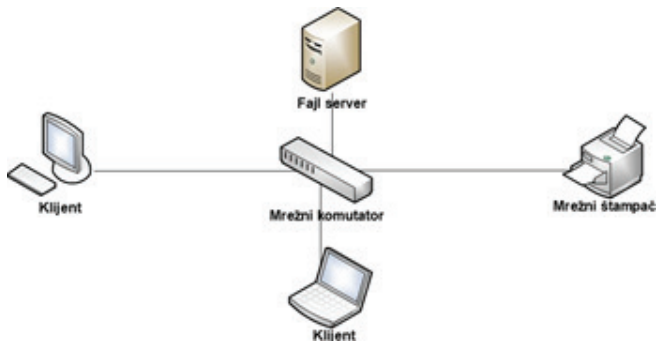
Na osnovu alokacije resursa, tradicionalna *IT* infrastruktura može biti organizovana u formi mreže ravnopravnih računara (eng. *peer to peer – P2P*) ili u formi klijent-server organizacije. Organizacija u formi ravnopravnih računara omogućava povezanim uređajima da dele (eng. *share*) resurse jedan sa drugima. Resurs je opšti pojam i on može da bude bilo koji objekat koji može da se deli, kao što su na primer štampači i fajlovi. U mreži ravnopravnih računara, kako sam naziv implicira, svi računari su ravnopravni. Primer jedne ovakve organizacije dat je na Slici 1-2, gde svaki parnjak (eng. *peer*) deli fajlove sa svog lokalnog hard diska sa drugim parnjacima na mreži. Ovaka tip organizacije resursa najčešće se sreće u kućnim okruženjima i u okruženjima manjih organizacija.

Slika 1-2: *Mreža ravnopravnih računara*



Druga vrsta organizovanja računarske mreže u tradicionalnoj *IT* infrastrukturi, model klijent-server, najčešće se koristi u korporativnim okruženjima. U ovakvom okruženju klijent uređaji koriste usluge servera. Zbog činjenice da su svi resursi locirani na jednom ili više servera, administriranje ovakvog okruženja je jednostavnije od administriranja mreže ravnopravnih računara, gde se svakim računarom posebno upravlja, nezavisno od drugih računara u istoj mreži. Primer ovog okruženja dat je na Slici 1-3, gde namenski fajl server (eng. *dedicated file server*) obezbeđuje klijentima pristup deljenim fajliovima (eng. *share files*), a mrežni štampač je neposredno preko mreže dostupan klijentima na korišćenje.

Slika 1-3: *Klijent-server organizacija*



Za implementaciju kako tradicionalne, tako i konvergirane infrastrukture (o ovoj infrastrukturi biće više reči u nastavku), neophodne su sledeće *IT* komponente:

- klijentski uređaji,
- računarska mreža sa konekcijom ka Internetu i
- serveri i računski centri.

Da bi se razumeo koncept konvergirane infrastrukture, neophodno je poznavanje navedenih komponenti. Zbog toga će u nastavku biti prikazane osnovne informacije o svakoj od nabrojanih komponenti.

1.1. KLIJENTSKI UREĐAJI

Svakako je jedan od glavnih resursa *IT* infrastrukture klijentski uređaj. Klijent predstavlja bilo koji računarski resurs koji neposredno koristi krajnji korisnik (eng. *end user*). Osnovne komponente koje čine jedan klijentski uređaj (računarski sistem):

- procesor,
- memorija,
- skladišni uređaj,
- sistem za napajanje i hlađenje i
- operativni sistem.

Do sada, najzastupljeniji klijentski uređaji su bili stoni računari (eng. *desktop computer*) i lap top računari (eng. *laptop*). Međutim, u poslednjoj deceniji, mnogi drugi uređaji koji su dobili ulogu klijenata počeli su da se koriste, kao što su pametni telefoni (eng. *smart phone*), tablet računari (eng. *tablet computers*) i tanki klijenti (eng. *thin client*). Svaki od navednih uređaja ima svoje prednosti i nedostatke.

Desktop računar je klijent koji se koristi na fiksnoj lokaciji. Mnogo godina su desktop računari bili najzastupljeniji u korporacijama, a i danas oni zauzimaju značajno mesto među poslovnim korisnicima. Ovi računari se proizvode u različitim faktorima oblika (eng. *form factor*). Faktor oblika je kategorizacija koja opisuje dimenzije i oblik računara, uređaja, ili računarske komponente. Na Slici 1-4 prikazani su *HP* (*Hawlett Packard*) *desktop* računari u tri različita faktora oblika.

Slika 1-4: *HP desktop računari različitih faktora oblika*



Izvor: (slika preuzeta sa: https://www8.hp.com)

Sve u jedan personalni računar (eng. *All in One Personal Computer – All in One PC*) je model *desktop* računara kome je većina komponenti ugrađena u monitor, a kao posebne komponente izdvojeni su samo tastatura i miš. Ovaj tip računara zauzima vrlo malo prostora, a kao glavni nedostatak navodi se ograničena mogućnost proširenja, tj. dodavanja novih komponenti. Jedan *HP* model sve u jedan prikazan je na Slici 1-5.

Slika 1-5: *HP računar tipa sve u jedan*



Izvor: (slika preuzeta sa: https://www8.hp.com)

Laptop računari su prenosivi računarski sistemi. Njih pre svega koriste mobilni korisnici čija je priroda posla takva da nisu vezani za fiksnu lokaciju. Kao i *desktop* računari, laptop računari su dostupni u nekoliko faktora oblika. Kao posebna kategorija ovih računara izdvajaju se mrežni računari (eng. *netbooks*). Ovi računari imaju slabije performanse i manje dimenzije od klasičnih laptop računara, a samim tim i lakši. U njih se uglavnom ugrađuje slabiji procesor koji koristi manje električne energije, kao što je *Intel Atom*.

Tablet računar je mali uređaj koji poseduje ekran osetljiv na dodir (eng. *touchscreen*) umesto tastature. Tableti se uglavnom koriste za pretraživanje Interneta, a mogu da podrže većinu aplikacija za obavljanje rutinskih i svakodnevnih poslovnih zadataka. *HP Slate 2* model prikazan je na Slici 1-6.

Slika 1-6: *HP Slate 7 tablet računar*



Izvor: (slika preuzeta sa: https://www.bhphotovideo.com)

Pametni telefon je uređaj koji se drži u ruci i koji omogućava visok stepen mobilnosti i konektivnosti. Koristi se uglavnom kao sekundarni uređaj, a kada su zahtevi za procesorskom snagom mali, i kao primarni uređaj krajnjeg korisnika. Ovi uređaji omogućavaju konekciju ka Internetu preko *3G* i *4G* mreža mobilnih mreža. Ovi uređaji takođe mogu da vrše funkciju i bežične pristupne tačke (eng. *Wireless Access Points – WAP*). Pametni telefoni osim navedenog podržavaju i *bluetooth* konekcije koje se uglavnom koriste za sinhronizaciju podataka između različitih uređaja i povezivanje perifernih uređaja kao što su slušalice i zvučnici.

Mobilna mreža treće generacije (*3G*) je bezična širokopojasna mreža (eng. *Wireless Wide Area Network – WWAN*) koja podržava propusni opseg od bar 200Kb/s. S druge strane, mobilna mreža četvrte generacije (*4G*) podržava propusni opseg od 100 Mb/s kada je korisnik u pokretu (na primer kada je korisnik u vozilu) i 1Gb/s kada se korisnik kreće sporo, ili kada je u mestu. Pametni telefoni uglavnom koriste procesore koji troše malo električne energije kao što su *ARM* procesori. Zbog manje potrošnje električne energije, ovi procesori se slabo zagrevaju. Među najpopularnijim pametnim telefonima su telefoni kompanije *Apple* koji koriste *iOS* operativni sistem, i telefoni koji koriste *Android, Blackberry OS* i *Windows Phone* operativne sisteme.

Zbog činjenice da većina zaposlenih poseduje sopstveni klijentski uređaj, kao što su tableti i pametni telefoni, veliki broj kompanija promoviše politiku „ponesi sopstveni uređaj" (eng. *Bring Your Own Device – BYOD*). Ova politika dozvoljava korisnicima da se povezuju za korporativnu mrežu preko svojih privatnih uređaja. Prednosti koje ova politka donosi odnosi se na smanjivanje kapitalnih investicija za nabavku opreme, a najveći nedostatak je smanjivanje bezbednosti.

Tanki klijent je „osiromašeni" računar koji poseduje ograničenu memoriju, procesorsku moć, i kapacitet za skladištenje. On je dizajniran tako da može samo da pokrene aplikaciju za virtualizaciju *desktop*-a, a korisniku se čini kao da je njegovo radno okruženje (eng. *desktop environment*) na lokalnom računaru, dok se ono nalazi na udaljenom serveru. O ovoj, kao i o drugim tehnikama virtualizacije, dato je više u Glavi 3. Tanki klijenti su dostupni u *desktop* i mobilnom faktoru oblika.

1.2. RAČUNARSKA MREŽA

Sledeća važna komponenta korporativnog *IT* okruženja je računarska mreža. Računarska mreža je skup komponenti, hardverske i softverske opreme koja obezbeđuje i omogućava prenos podataka i informacija u formi paketa, kako bi računari i ljudi međusobno komunicirali i razmenjivali poruke.

U literaturi je poznato više taksonomija mreža, a prema jednoj, gde se u obzir uzima tehnologija, mreže se dele na:

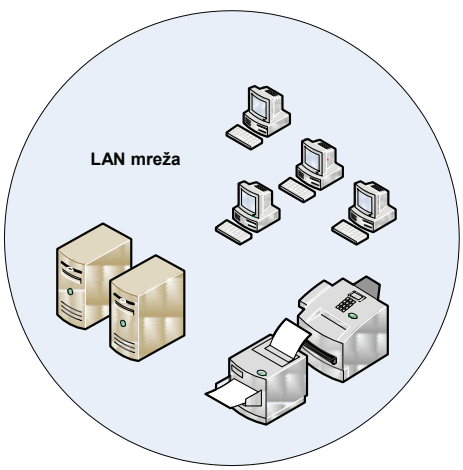
- javna komutirana telefonska mreža (eng. *Public Switched Telephone Network – PSTN*). Ova mreža se koristi za uspostavljanje poziva korišćenjem tradicionalnog telefona,
- ćelijska mreža (eng. *cellular network)*. Ova mreža se koristi za uspostavljanje poziva i tranfer podataka pomoću mobilnih telefona i
- Internet. Globalna računarska mreža koja se temelji na *WWW* (*World Wide Web*) tehnologiji.

Prema geografskoj oblasti koju pokriva, računarske mreže se dele na lokalne, kampus, gradske i širokopojasne mrežne.

U korporativnom okruženju najčešće se koristi lokalna računarska mreža (eng. *Local Area Network – LAN*). Engleska skraćenca *LAN* je usvojena u srpskom jeziku za imenovanje ove vrste mreže, pa će ova skraćenica biti korišćena i u ovom udžebniku.

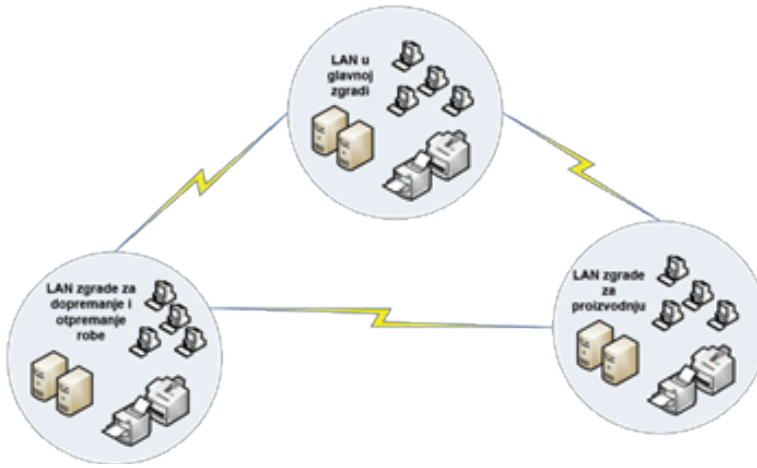
LAN je privatna mreža koju koriste zaposleni u kompaniji za razmenu i transfer računarskih resursa, pristup podacima i komunikaciju. *LAN* mreža se najčešće ograničava na računare koji se fizički nalaze na jednoj lokaciji, kao što je na primer jedna zgrada. *LAN* najčešće koristi dve tehnologije referentnog modela *ISO/OSI* (*International Organization for Standardization/Open Systems Interconnection*) sloja veze (eng. *data link layer*) za prenos podataka, i to *Eternet* (*IEEE* 802.3) i bežičnu (eng. *Wireless Fidelity - IEEE* 802.11), ili kombinaciju ove dve. Grafički prikaz tipične *LAN* mreže dat je na Slici 1-7.

Slika 1-7: *LAN mreža*



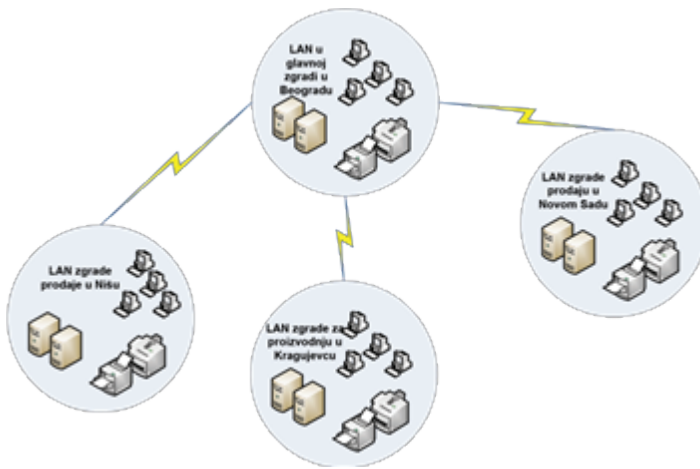
Kompanija čije se prostorije nalaze u nekoliko udaljenih zgrada koristi kampus računarsku mrežu (eng. *Campus Area Network – CAN*). Na primer, kompanija koja poseduje glavnu zgradu, zgradu gde se realizuju proizvodne aktivnosti i zgradu za otpremanje i primanje robe, može svoju računarsku mrežu da realizuje u formi kampus mreže. Ovaj tip korporativnih mreža prikazan je na Slici 1-8.

Slika 1-8: *Kampus mreža*



Gradska računarska mreža (eng. *Metropolitan Area Network – MAN*) je slična kampus mreži s tim što su lokacije (zgrade) locirane u različitim delovima jednog grada. Konačno, ako je kompanija velika i ako poslovne aktivnosti obavlja u više različitih gradova, pa i država, njena mreža je organizovana u formi širokopojasne mreže (eng. *Wide Area Network – WAN*). Vizuelna reprezentacija ovog tipa računarske mreže data je na Slici 1-9.

Slika 1-9: *Širokopojasna mreža*

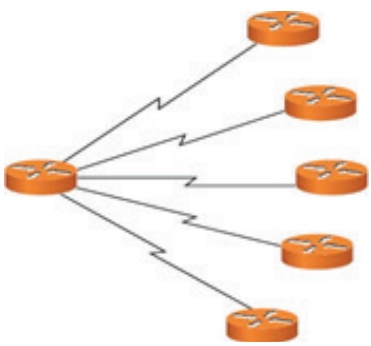


Kada se govori o širokopojasnim korporativnim mrežama, potrebno je istaći da postoje tri najčešće korišćene topologije:

- Topologija čvor i linkovi,
- potpuno zamršena topologija i
- delimično zamršena topologija.

Kada se povezuje više lokacija, gde se jedna lokacija može diferencirati (razlikovati) od drugih kao glavna, onda se koristi topologija čvor i linkovi (eng. *hub and spoke*), gde linkovi (veze) sa svih udaljenih lokacija (eng. *spoke*), idu ka glavnoj lokaciji (eng. *hub*). Ova topologija je slična topologiji zvezde koja se koristi u topološkim rešenjima za *LAN* mreže (Slika 1-10).

Slika 1-10: *Topologija čvor i linkovi*

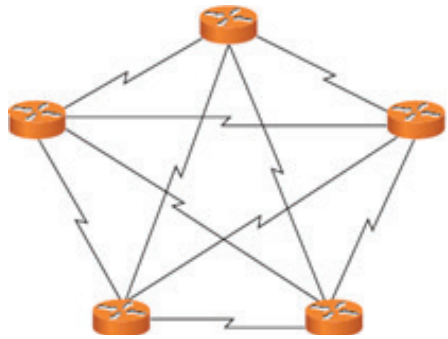


S druge strane, kod potpuno zamršene topologije (eng. *full mesh*), sve lokacije su međusobno povezane (Slika 1-11). U ovom slučaju postoji više redudantnih putanja između svake dve tačke. Ukupan broj linkova računa se po formuli:

$$\omega = n(n-1),$$

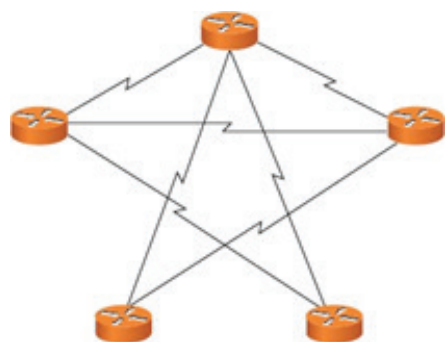
gde je ω ukupan broj linkova, a n broj lokacija.

Slika 1-11: *Potpuno zamršena topologija*



Delimično zamršena topologija (eng. *partial-mesh*) je hibrid prethodne dve. U slučaju ove topologije postoji redudantnost kod povezivanja značajnijih lokacija kako bi se obezbedila otpornost u slučaju otkaza jedne putanje (Slika 1-12). Kod drugih, manje značajnih povezivanja, ne postoje redudantne putanje u cilju smanjivanja složenosti i cene, kako implementacije, tako i održavanja.

Slika 1-12: *Potpuno zamršena topologija*



1.2.1. *Ethernet* mreža

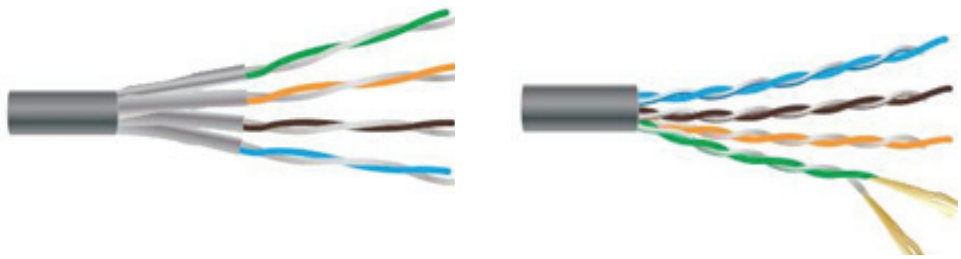
Kao što je gore pomenuto, prema tehnologiji koja se koristi za prenos podataka na drugom sloju *ISO/OSI* referentnog modela, lokalne mreže mogu da budu *Ethernet* (*IEEE* 802.3 stadnard), bežične (*IEEE* 802.11 standard) ili kombinacija ove dve tehnologije. *IEEE* (*Institute of Electrical and Electronics Engineers*) je organizacija koja donosi standarde iz oblasti računarskih tehnologija. Savremena *Ethernet* mreža se sastoji od jednog ili više komutatora (eng. *switch*), mrežnih kartica (eng. *Network Interface Card – NIC*) na svakom uređaju koji učestvuju u komunikaciji i kablovima koji povezuju mrežne kartice uređaja sa određenim portom komutatora.

Za povezivanje uređaja u *Ethernet* mreži najčešće se koristi bakarni kabl sa upredenim paricama (eng. *twisted pair*). Ovaj kabl se povezuje pomoću *RJ45* (*Registered Jack 45*) priključka na mrežnoj kartici. Zbog činjenice da se ovaj kabl najčešće koristi, mnoge matične ploče računara imaju fabrički ugrađeni *RJ45* port. Jedna od mana bakarnog kabla jeste ta što ovaj tip kabla nije otporan na elektromagnetno zračenje (eng. *Electromagnetic Interference – EMI*). Ovakvu vrstu zračenja generišu kablovi za električnu energiju i uređaji koji koriste mnogo električne energije, kao što su na primer proizvodna oprema i mikrotalasne rerne.

Postoje dva tipa bakarnog kabla, i to nezaštićeni kabl sa upredenim paricama (eng. *Unshielded Twisted Pair – UTP*) i zaštićeni kabl sa upredenim paricama (eng. *Shielded Twisted Pair – STP*). Razlika između ove dve kategorije je u tome što je prva manje otporna na elektromagnetno zračenje, dok druga kategorija kablova poseduje omotač koji štiti kabl od elektromagnetnog zračenja. Još jedna mana bakarnog kabla je opadanje jačine signala sa povećavanjem dužine kabla, i zbog toga je preporučena maksimalna dužina bakarnog kabla 100 metara. Zbog relativno niske cene uz zadovoljavajuće performanse, u korporativnim mrežama najčešće se koristi nezaštićeni kabl sa upredenim paricama.

Kabl sa upredenim paricama sastoji se iz 4 para žica. Da bi se bolje definisali pinovi i kodiranje na osnovu boje za svaku žicu, razvijen je standard *TIA/EIA-568* (*Telecommunications Industry Association/Electronic Industries Alliance*). Prva iteracija ovog standarda objavljena je 1991. godine pod nazivom *TIA/EIA-568-A*, dok je ažurirana verzija ovog standarda objavljena je 2001. godine pod nazivom *TIA/EIA- 568-B*. Kodovi boja za nezaštićeni i zaštićeni kabl prikazani su na Slici 1-13.

Slika 1-13: *Zaštićeni bakarni kabl (levo) i nezaštićeni (desno)*



U Tabeli 1-1 prikazani su najčešće korišćeni standardi za bakarni kabl.

Tabela 1-1: *Standardi za bakarni kabl*

Standard	Brzina prenosa podataka	Kategorija kabla
10BaseT	10Mbps	Cat3
100BaseTX	100Mbps	Cat5
1000BaseT	1Gbps	Cat5e, Cat6
10GBaseT	10Gbps	Cat6e

Druga vrsta kabla koja se najčešće koristi u *Ethernet* mrežama su optički kablovi. Optički kablovi se prave od stakla ili od plastičnog vlakna. Zbog činjenice da optički kablovi prenose podatke pomoću snopova svetlosti, oni su otporni na elektromagnetno zračenje. Takođe, ovi kablovi mogu da prenose signal preko većih razdaljina i zbog toga su svoju primenu našli i u širokopojasnim mrežama.

Osnovne komponente koje su potrebne za implementaciju optičkih kablova su:

- optički predajnik koji pretvara električni signal u svetlosni snop,
- optičko vlakno koje služi kao prenosni medijum i
- optički prijemnik koji pretvara svetlosni snop u električni signal.

Postoje dve vrste optičkih vlakana: jednomodna (eng. *singlemode)* i višemodna (eng. *multimode*). U slučaju prve vrste, snop svetlosti ulazi u vlakno pod određenim uglom, dok višemodni tip koristi *LED* (*Light Emitting Diode*) za emitovanje svetlosti. Jednomodna vlakna su bolja od višemodnih zato što koriste laser kao izvor svetlosti. Kod višemodnih vlakana se javlja modalno rasprašivanje koje se dešava zbog toga što *LED* svetlosni signali ulaze pod različitim uglovima, zbog čega se svi signali ne završavaju na drugom kraju vlakna u isto vreme.

Optički kablovi su raspoloživi u brzinama od 1Gbps, 10Gbps, 40Gbps i 100Gbps. Razdaljina koju podržavaju je između 300m i 10km. Glavna mana optičkih kablova je cena koja je značajno veća od cene bakarnog kabla. S druge strane, za povezivanje optičkim kablom, uređaji moraju da imaju posebne adaptere.

U optičkim mrežama se koriste četiri topa konektora: *LC* (*Lucent Connector*), *SC* (*Standard Connector*), *ST* (*Straight Tip*) i *MTRJ* (*Media Termination Recommended Jack*).

Mnoge kompanije u svojoj *IT* infrastrukturi koriste i bakarne i optičke kablove. Optički kablovi se najčešće koriste za implementaciju mrežnih okosnica (eng. *backbone*) i za međusobno povezivanje lokalnih mreža, dok se bakarni kablovi koriste za priključivanje klijentskih uređaja na *LAN*.

1.2.2. Bežična mreža

Bežične mreže koriste radio talase kao medijum za prenos podataka. Ove mreže se nazivaju bežične lokalne računarske mreže (eng. *Wireless LAN – WLAN*) i zasnivaju se na *IEEE* 802.11 standardima. *IEEE* 802.11 specifikacija omogućava uređajima da budu usaglašeni sa više od jednog standarda. Svaki standard za bežične mreže definiše frekvencije na kojoj rade uređaji i maksimalnu teoretsku brzinu prenosa podataka. Pregled *IEEE* 802.11 standarda dat je u Tabeli 1-2.

Tabela 1-2: *IEEE 802.11 standardi*

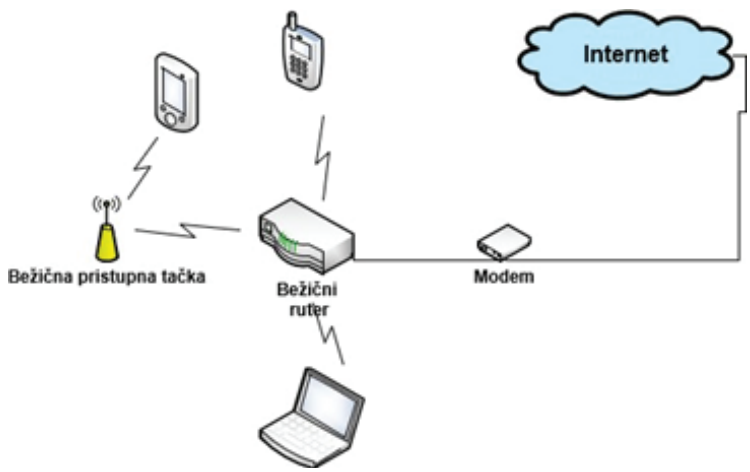
Standard	Frekvencija	Maksimalna brzina prenosa
802.11a	5 MHz	54Mbps
802.11b	2.4 MHz	11Mbps
802.11g	2.4 MHz	54Mbps
802.11n	5 MHz/2.4 MHz	do 600Mbps
802.11ac (draft)	5 MHz	do 1Gbps

Od navedenih standarda, potrebno je posebno da se istakne *IEEE* 802.11n standard koji podržava frekvencije od 5MHz i 2.4MHz. Ova tehnologija je poznata i pod nazivom *MIMO* (*Multiple Inputs Multiple Outputs*) zato što brzinu od 600Mbps ostvaruje pomoću više antena. Standard 802.11ac je u vreme pisanja ovog užbenika još uvek u fazi nacrta.

U početku su 802.11 mreže služile samo za realizaciju privatnih mreža u kući, ili u kancelariji. Međutim, poslednjih godina su javne 802.11 mreže postale popularne kao bežična priključna mesta (eng. *Wireless Hotsposts*). Većina 802.11 konfiguracija se baziraju na bezičnoj pristupnoj tački (eng. *Wireless Access Point – WAP*). Na pristupnu tačku povezuju se svi bežični klijenti (eng. *hosts*) koji na taj način međusobno komuniciraju.

Većina korporacija koristi hibridnu konfiguraciju računarske mreže tako što koriste i 802.3 i 802.11 standarde. Serveri su zbog veće potrebe za pozudanom konekcijom vezani žično korišćenjem *Ethernet* tehnologije, dok su manje važni klijenti i drugi resursi vezani bežično. Konfiguracija *WLAN* mreže sa bežičnom pristupnom tačkom je prikazana na Slici 1-14.

Slika 1-14: *Konfiguracija WLAN mreže*



1.2.3. Povezivanje na Internet

Pouzdana konekcija na Internet sa dovoljnim propusnim opsegom (eng. *bandwidth*) je ključna za nesmetano obavljanje poslovnih operacija. Prilikom izbora Internet konekcije potrebno je da se utvrdi da li je troškovno efikasnije da se koristi sistem plaćanja na osnovu ostvarenog saobraćaja (eng. *postpaid*), ili da se koristi režim neograničenog prenosa podataka na mesečnom nivou (eng. *flat rate*). Neki od najčešćih načina na koji se korporacije povezuju na Internet su:

- T ili E nosilac,
- DSL,
- kablovski modem,
- optičkim vezama i
- satelitska veza.

T veza se zasniva na *T1* „deblima". *T1* deblo je kolo za prenos podataka koje funkcioniše brzinom od 1.544 Mbps. *T1* veze se koriste u Severnoj Americi, Japanu i Južnoj Koreji. Poslovni subjekti najčešće kombinuju više *T1* „debala" i tako kreiraju logičku konekciju. Tako je na primer *T3* veza sastavljena od tri *T1* konekcije.

E veza se temelji na „deblima" brzine 2.048 Mbps. Ove veze se najčešće koriste u evropskim zemljama.

Digitalna pretplatnička linija (eng. *Digital Subscriber Line – DSL*) obezbeđuje Internet konekciju slanjem digitalnih signala putem kablova sa upredenim paricama ili optičkim telefonskim kablovima. Postoje dva osnovna tipa digitalnih pretplatničkih linija:

- asimetrična digitalna pretplatnička linija (eng. *Asymmetric Digital Subscriber Line – ADSL*) koja obezbeđuje veći propusni opseg prilikom preuzimanja podataka (eng. *download*) nego kada se podaci šalju (eng. *upload*) i
- simetrična digitalna pretplatnička linija (eng. *Symmetric Digital Subscriber Line – SDSL*) kod koje je propusni opseg prilikom preuzimanja podataka isti kao i kod slanja podataka.

Brzina ovakvih Internet konekcija uglavnom zavisi od infrastrukturne opreme telekomunikacionih provajdera.

Kablovski modem za konekciju na Internet koristi mrežu operatera kablovske televizije. Teoretski, konekcije putem kablovskog modema podržavaju veće brzine od digitalne pretplatničke linije. Brzine dostižu i do 200Mbps. Ove veze sa Internetom su popularne i kod nas, a njihova cena je iz godine u godinu sve prisupačnija krajnjim korisnicima.

U slučaju korišćenja optičkih veza ka Internetu, koristi se uređaj pod nazivom konverter medijuma (eng. *Media Converter*). Ovaj uređaj vrši konverziju optičkog signala u signal za prenos u *LAN* mreži.

Satelitske veze se uglavnom koriste za područja koja su udaljena i gde ne postoji infrastruktura neophodna da podrži druge vidove Internet konekcija. Kod satelitske veze, signal se šalje od satelita na površini Zemlje do satelita u Zemljinoj orbiti. Zbog ovakvog načina komunikacije satelitske veze „pate" od kašnjenja (eng. *latency*).

1.2.4. VoIP sistem

Kada se govori o mrežnoj infrastrukturi potrebno je spomenuti i *VoIP* (eng. *Voice over IP*) rešenje. Mnoge kompanije prelaze na ovo rešenje umesto standardnog *PBX* rešenja. Kod *VoIP* sistema telefonski pozivi se šalju i primaju putem *Ethernet* ili Internet mreža.

Neke od prednosti *VoIP* sistema u odnosu na klasični telefonski sistem uključuju:

- niži operativni troškovi,
- niži ukupni troškovi posedovanja opreme i
- *VoIP* sistem podržava više funkcionalnosti nego klasični telefonski sistem.

Još jedna značajna prednost ovog sistema jeste ta što korisnik nije više ograničen na jedan telefonski priključak, već korisnik može da telefonira sa bilo koje lokacije gde se nalazi njegov *IP* telefon.

VoIP sistem može da se implementira na brojne načine. Najjednostavniji način jeste da se instalira softver na PC računaru kao što je *Skype* ili *VoIP Discount*. Drugi način implementacije je putem pretplate kod provajdera *VoIP* usluga i povezivanje putem podržanog *IP* telefona ili analognog telefonskog priključka (eng. *Analog Telephone Adapter – ATA*). *VoIP* sistem može da se integriše sa softverom za unificirane komunikacije putem poruka, ili sa serverom za unificirane komunikacije (eng. *Unified Communications Server – UC Server*).

Međutim, pored pobrojanih prednosti, *VoIP* rešenje ima i nekoliko nedostataka:

- sistem ima veći broj kritičnih tačaka (eng. *points of failure*) nego klasični telefonski sistem,
- prilikom promene provajdera *VoIP* sistema, verovatno je potrebno da se promeni i komunikaciona oprema i
- povećano korišćenje propusnog opsega lokalne mreže ili veze sa Internetom.

1.3. SERVERI

Server pre svega mora da bude jak računar koji može da obavlja veliki broj operacija u pozadini (eng. *background*). Serveri imaju jake procesore tipa *Intel Xeon* ili *AMD Opteron* i izvršavaju veliki broj simultanih procesa. Serveri takođe mogu da imaju veliki broj uloga (eng. *roles*), a neke od njih su:

- obrada procesa logovanja korisnika,
- slanje, primanje i skladištenje *email* poruka,
- deljenje fajlova i štampača,
- *hostovanje* platforme za *Web* sajtove i
- čuvanje podataka i obezbeđivanje pristupa podacima.

U skladu s navedenim ulogama, serveri se u savremenim korporativnim okruženjima dele na sledeće kategorije:

- serveri za deljenje fajlova (eng. *file sharing servers*),
- serveri za štampanje (eng. *printing servers*),
- serveri infrastrukture (eng. *infrastructure servers*),
- serveri baza podataka (eng. *database servers*),
- serveri za razmenu poruka (eng. *messaging servers*),
- *Web* serveri (eng. *web server*),
- terminal serveri (eng. *terminal servers*) i
- serveri za kolaboraciju (eng. *collaboration servers*).

Serveri za deljenje fajlova se u najmanju ruku sastoje od uređaja za skladištenje podataka, protokola za deljenje fajlova (eng. *file sharing protocol*), i od skupa pravila za pristup podacima (eng. *access rules*) koja određuju ko može i kada može da pristupa, čita i modifikuje deljene fajlove. Najčešće korišćeni protokoli za deljenje fajlova dati su u Tabeli 1-3.

Primarni parametri koji se uzimaju u obzir prilikom nabavke fajl servera su kapacitet, pouzdanost i brzina uređaja za skladištenje podataka. Uređaj za skladištenje koji je priključen na mrežu (eng. *Network Attached Storage – NAS*) je alternativa fajl serveru koja se često koristi.

Server za štampanje čuva softver i drajvere za štampače koji su povezani lokalno ili putem mreže. Ovaj tip servera upravlja poslovima štampe svih povezanih štampača. Kada se određeni posao distribuira na štampanje, server stavlja taj posao u red čekanja (eng. *queue*), a zatim ga delegira određenom štampaču na realizaciju.

Tabela 1-3: *Protokoli za deljenje fajlova*

Protokol	Opis
SMB (Server Message Block)	Siguran protokol koji služi za deljenje fajlova između *Windows* računara. *SMB* je takođe poznat pod nazivom *CIFS* (*Common Internet File System*)
NFS (Network File System)	Siguran protokol koji služi za deljenje fajlova između *Linux* klijenata
FTP (File Transfer Protocol)	Industrijski standard za transfer fajlova u formi čistog (eng. *plain*) teksta
SFTP (Secure File Transfer Protocol)	Industrijski standard za transfer fajlova putem šifrovanog kanala

Iako male organizacije uglavnom koriste stoni računar kao server za štampu, deljenje štampača putem serverskog operativnog sistema omogućava brojne prednosti, a neke od njih su:

- centralizovana kontrola pristupa štampačima i poslovima štampe,
- mogućnost logičkog grupisanja štampača i
- mogućnost određivanja rasporeda raspoloživosti štampača.

Serveri infrastrukture rade „iza scene“ obezbeđujući usluge koje omogućavaju funkcionisanje svih delova mreže, kao što su dodeljivanje *IP* (*Internet Protocol*) adresa, razrešavanje simboličkih imena računara, autentifikacija uređaja i slično. Prema funkcijama koje obavljaju, moguće je izvršiti podelu infrastrukturnih servera na sledeći način:

- *DHCP* (*Dynamic Host Configuration Protocol*) serveri dodeljuju *IP* konfiguracije računarima koji koriste *TCP/IP* (*Transmission Control Protocol/ Internet Protocol*) set protokola,
- *DNS* (*Domain Name System*) serveri koji razrešavaju potpuno kvalifikovano ime domena (eng. *Fully-qualified Domain Name – FQDN*) u *IP* adresu,
- *WINS* (*Windows Internet Naming Services*) serveri koji razrešavaju *NetBIOS* (*Network Basic Input Output System*) imena u *IP* adrese,
- serveri direktorijuma (eng. *Directory Services Servers*) upravljaju direktorijumskom bazom podataka koja čuva podatke o svim resursima na mreži,
- serveri za autentifikaciju (eng. *Authentication Servers*) proveravaju da li su računari i resursi ono što oni „tvrde" da jesu i vrše proces verifikacije. Ovi serveri koriste protokole za autentifikaciju, od kojih su najpoznatiji *NTLM* (*Network Lan Manager*), *Kerberos* i *RADIUS* (*Remote Access Dial-in User Service*) i
- *proxy* serveri koji optimizuju pristup Internetu tako što keširaju (eng. *caching*) sadržaj i vrše filtriranje *Web* saobraćaja.

Serveri baza podataka skladište podatke i omogućavaju konkurentne konekcije na bazu podataka. Ovi serveri najčešće skladište baze podataka za obradu transakcija u realnom vremenu (eng. *Online Transaction Processing - OLTP*) i baze podataka za podršku procesu odlučivanju (eng. *Decision Support System - DSS*). Baze podataka za obradu transakcija u realnom vremenu obavljaju veliki broj operacija, kao što su dodavanje novih podataka, brisanje i modifikovanje postojećih podataka. Kao primeri ovih baza podataka mogu da se navedu baze podataka koje rade u pozadini sajtova za elektronsku trgovinu (eng. *e-Commerce*) i baze koje čuvaju podatke o proizvodima u skladištima i magacinima. S druge strane, baze podataka za podršku procesu odlučivanju služe prvenstveno za operacije čitanja. Podaci se čitaju iz baze pomoću *SQL* (*Structured Query Language*) naredbi.

Serveri za razmenu poruka upravljaju skladištenjem i prenosom poruka koje korisnici razmenjuju. Najčešći tip servera za razmenu poruka je email server (eng. *Email Server*).

Web server je bilo koji server koji opslužuje *HTTP* (*HyperText Transfer Protocol*) ili *HTTPS* (*HyperText Transfer Protocol Secure*) zahteve. Iako mnogi ljudi vezuju *Web* servere za Internet, mnoge kompanije implementiraju ovaj tip servera u okviru svog Intraneta kako bi omogućile korisnicima internu razmenu informacija i sadržaja.

Terminal server omogućava korisnicima da izvršavaju i interaguju sa aplikacijama koje se nalaze na virtuelnom *desktopu*. Server procesira sve aktivnosti vezane za aplikaciju, a korisnički interfejs prenosi rezultate obrade do korisnika putem računarske mreže.

Server za kolaboraciju omogućava da više korisnika zajedno rade na dokumentu, ili jednom projektu. Servisi ovog servera obuhvataju konkurentna ažuriranja sadržaja, proveru dokumenata i kreiranje verzija (eng. *versioning*). Implementacija jednog takvog servera prikazana je u Glavi 18.

1.3.1. Računski centar

Računski centar (eng. *data center*) je zgrada ili prostorija u kojoj se nalaze serveri i druga *IT* oprema kompanije.

Na ovim lokacijama, gde je „srce" infrastrukture jedne korporacije, najčešće se skladište deljeni podaci i usluge koje *IT* sistem pruža.

Prema jednom stanovištu, svaki računski centar treba da obezbedi sledeće:

- odgovarajući prostor za čuvanje opreme,
- dovoljno električne energije za rad opreme,
- kontrola okruženja da bi se toplota koju generiše oprema na odgovrajući način rasipala i da bi se oprema hladila do optimalne radne temperature,
- mehanizme fizičke sigurnosne kontrole kako bi se onemogućio neautorizovan pristup opremi i podacima i
- Internet konekcija koja će zadovoljiti potrebe za brzim prenosom podataka i pravovremenim obavljanjem svih aktivnosti.

Ako se posmatra fizička lokacija računskog centra (da li se računski centar nalazi u prostorijama kompanije, ili na nekoj drugoj lokaciji), računski centar može da bude organizovan u nekoj od sledećih formi:

- kolokacioni računski centar,
- namensko hostovanje,
- hostovanje kojim se upravlja,
- deljeno hostovanje i
- servisi klaud računarstva.

U okruženju kolokacionog računskog centra (eng. *colocated data center*), kompanija pružalac kolokacione usluge iznajmljuje prostor za čuvanje *IT* opreme računskog centra organizacijama klijentima. Serveri, skladišni uređaji, mrežna oprema i softver koji je instaliran u kolokacionom centru je u vlasništvu organizacije klijenta, koji se naziva pretplatnik (eng. *subscriber*). Ova opcija je dobra u slučaju kada kompanija nema raspoloživi prostor gde bi implementirala opremu i Internet konekciju, ali je potrebno da ima vlasništvo i kontrolu nad opremom.

Namensko hostovanje (eng. *dedicated hosting*) je slično rešenje kao i prethodno s razlikom da kompanija koja pruža ovu uslugu zajedno sa prostorom, sistemima za hlađenje i izvorima električne energije iznajmljuje i servere, skladišne uređaje i mrežnu infrastrukturu.

Hostovanje kojim se upravlja (eng. *managed hosting*) obuhvata sve kao i model namenskog hostovanja, s tim da kompanija pružalac usluge pruža i usluge upravljanja i kontrolisanja računskog centra i obezbeđuje neophodne resurse za ove aktivnosti. Ova opcija je atraktivna za kompanije koje nemaju kompetentno *IT* osoblje, ili ga nemaju u dovoljnom broju, pa im je potrebna pomoć treće strane.

Deljeno hostovanje (eng. *shared hosting*) je model koji opisuje scenario kada se više korporativnih *Web* sajtova nalazi na istom serveru. Ovaj model je pogodan u slučajevima kada *Web* sajtovi nemaju veliki broj poseta i kada za normalno funkcionisanje poslovnih operacija nije neophodan namenski server.

S obzirom da je ceo udžbenik posvećen klaud računarstvu, ovaj koncept je detaljnije prikazan u nastavku.

PITANJA ZA RAZMATRANJE



1. Objasniti strukture modela organizacije *IT* infrastrukture. Dati primere.
2. Navesti prednosti i nedostatke klijentskih uređaja, sagledavajući osnovne komponente koje ih sačinjavaju.
3. Šta je računarska mreža? Navesti i objasniti podele računarskih mreža na osnovu geografske oblasti koju pokrivaju.
4. Objasniti razliku između *UTP* i *STP* kabla. Koji tip kabla se koristi za povezivanje uređaja u *Ethernet* mreži i zašto?
5. Navesti razlike između tradicionalnih ožičenih mrežnih infrastruktura i bežičnih mrežnih infrastruktura.
6. Objasniti pojam Interneta. Navesti načine na koji organizacije mogu da se povezuju na Internet. Objasniti svaki način konekcije.
7. Šta je *VoIP*? Koji tip mreže se koristi u slučaju korišćenja *VoIP* sistema?
8. Navesti kategorije servera koje mogu da se implementiraju u korporativnim okruženjima. Dati primere.
9. Šta je računski centar? Koje forme organizovanja računarskih centara postoje? Objasniti.

2. KONVERGIRANA I HIPER-KONVERGIRANA *IT* INFRASTRUKTURA

Promene u tehnologiji i u poslovnim potrebama primorale su mnoge kompanije da se okrenu mnogo agilnijoj i troškovno efikasnijoj *IT* infrastrukturi koja omogućava da računarske tehnologije bolje, pravovremeno i efikasnije reaguju na brze i nepredvidive promene poslovnih potreba i pravno-regluativnog okruženja u kome kompanija posluje.

Agilnija i efikasnija *IT* infrastruktura u literaturi i stručnim krugovima poznata je pod nazivom konvergirana infrastruktura (eng. *Converged Infrastructure – CI*) i predstavlja novi fleksibilniji način organizovanja računarskih resursa. Hiperkonvergirana infrastruktura (eng. *Hyper-Converged Infrastructure – HCI*), koja je nastala poslednjih godina, ide još jedan korak dalje u odnosu na konvergirane sisteme, o čemu će biti reči na kraju glave.

> Konvergirana infrastruktura može da se definiše kao *IT* infrastruktura koja koristi najbolje prakse iz računarstva u procesu izbora serverskih, mrežnih i resursa za skladištenje podataka koji se alociraju (eng. *allocate*) na zahtev (eng. *on-demand*) raznim operativnim okruženjima i aplikacijama.

U ovoj glavi prvo su opisani preduslovi koji su neophodni za implementaciju konvergirane infrastrukture, kao što su zajedničke komponente, konsolidacija uređaja i upravljački softver. Zatim su prikazani problemi i barijere koje usporavaju implementaciju ovog novog koncepta. Poseban osvrt dat je na troškove tradicionalne *IT* infrastrukture, kako bi čitaoci mogli bolje da razumeju prednosti koje konvergirana i hiperkonvergirana infrastrukture omogućavaju. Konačno, na kraju ove glave opisan je konvergirani i hiperkonvergirani pristup organizovanja i korišćenja *IT* resursa, uz naglasak na smanjivanju troškova, koji su u uslovima korišćenja tradicionalne *IT* infrastrukture veliki.

2.1. KRATAK UVOD U KONVERGIRANU INFRASTRUKTURU

U svrhu razumljivog uvoda u koncept konvergirane infrastrukture, postavljen je jedan hipotetički scenario. Kompanija *X* koristi tradicionalnu *IT* infrastrukturu. Ova infrastruktura zadovoljava poslovne potrebe, ali ne ispunjava zahteve za efikasnim i efektivnim korišćenjem *IT* resursa. Neki serveri, posebno fajl serveri se koriste sa svega 20% kapaciteta. Serveri koji vrše druge uloge se koriste u punom kapacitetu samo kada se dostigne najveći nivo saobraćaja na mreži (eng. *peek network traffic*). Ako ova hipotetička kompanija želi da izvrši bilo kakve modifikacije na *IT* infrastrukturi, proces promena će biti skup i spor.

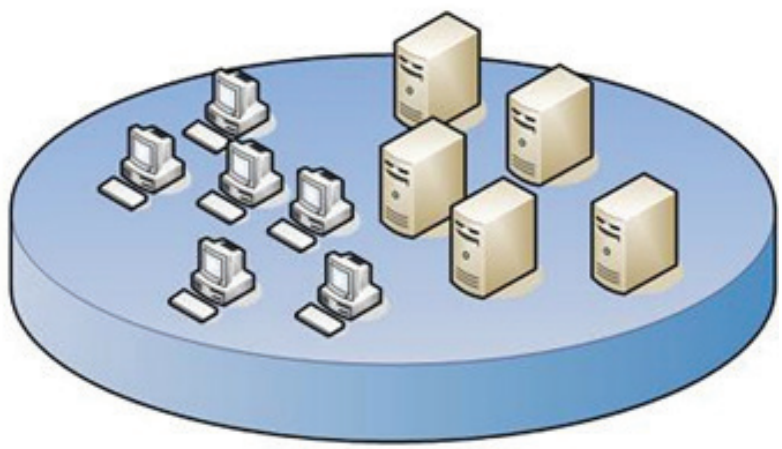
Rešenje konvergirane infrastrukture je dizajnirano pre svega sa ciljem da spreči „bujanje" računskog centra (eng. *data center sprawl*), da se poveća procenat iskorišćenosti raspoloživog hardvera, da se smanje prekomerne nabavke *IT* opreme i da se predudpredi stvaranje informacionih silosa (eng. *information silos*), ili informacionih ostrva (eng. *information islands*). „Bujanje" računskog centra je scenario kada dolazi do naglog povećanja *IT* uređaja, gde je procenat iskorišćenosti kapaciteta većine uređaja mali. Informacioni silosi su skupovi *IT* opreme koje poseduje određena poslovna jedinica, ili odeljenje jedne kompanije.

Konvergirana infrastruktura objedinjuje servere, skladišni prostor, mrežnu opremu i ostale *IT* resurse u jednu integrisanu celinu, tako da se njima upravlja kao jednim skupom resursa, što je mnogo efektivnije i troškovno efikasnije nego u slučaju tradicionalne *IT* infrastrukture, gde se svaki resurs posmatra nezavisno od ostalih resursa. Ovako postavljena infrastruktura formira bolju osnovu za virtuelizaciju i automatizaciju, kao ključnih elemenata za implementiranje modela usluga klaud računarstva (eng. *cloud delivery mode*l).

2.2. GLAVNI NEDOSTACI TRADICIONALNE IT INFRASTRUKTURE

Kod mnogih organizacija koje koriste tradicionalnu *IT* infrastrukturu, infrastrukturne komponente se grupišu, i u vlasništvu su organizacionih odeljenja ili divizija (Slika 2-1). Svaka organizaciona divizija poseduje sopstvenu računarsku opremu na kojoj se isključivo izvršavaju aplikacije i platforme za potrebe te divizije. Ova politika uglavnom vodi ka povećanim troškovima održavanja i nedovoljnoj iskorišćenosti opreme zato što se oprema ne deli između odeljenja. U ovakvim uslovima često dolazi do nekontrolisanog povećavanja količine *IT* opreme, tj. do *IT* „bujanja", što implicira još veće obaveze upravljačkih i administrativnih aktivnosti.

Slika 2-1: *Tradicionalna IT infrastruktura*



Jedan od glavnih razloga neiskorišćenosti *IT* opreme u tradicionalnoj organizaciji *IT* resursa jeste taj što kada se menja konfiguracija opreme u pogledu veličine i snage hardverskih platformi, potrebno je da se predvidi potreban kapacitet opreme, a te prognoze često nisu tačne. *Vendori* aplikacija uglavnom preporučuju jače platforme nego što je potrebno kako bi se postigle bolje operativne performanse. Ove preporuke smanjuju rizik od zastoja, ali s druge strane često vode ka predimenzioniranju serverskih rešenja.

S druge strane, *IT* tim u organizaciji često povećava minimalne hardverske zahteve kako bi predupredio rizik da će jednog dana biti potreban veći nivo hardverskog kapaciteta.

Praksa je pokazala da *IT* obično dodaje 20% - 30% na definisane minimalne zahteve hardverskih platformi. Sve navedene prakse i aktivnosti vode ka povećanim kapitalnim investicijama kada se vrši nabavka opreme.

2.2.1. Troškovi tradicionalne infrastrukture

Troškovi su uvek jedni od najvećih prepreka svake promene. Ova činjenica važi i za rešenja konvergirane infrastrukture. Da bi se jasnije videle prednosti konvergirane infrastrukture, potrebno je razumeti troškove koji nastaju usled tradicionalog vođenja *IT* resursa. Implementiranjem konvergirane infrastrukture, najveće uštede se ostvaruju u oblasti hardvera, objekata za skladištenje opreme i upravljanja infrastrukturom. Zbog toga su u ovom odeljku opisani navedeni troškovi tradicionalne *IT* organizacije.

Troškovi hardvera

Kada se govori o troškovima hardvera, uglavnom se misli na troškove nabavki serverskih i klijentskih platformi. Međutim, ovo nisu jedini hardverski troškovi. Treba imati u vidu i troškove rasta računarske mreže, održavanja računarske opreme, troškove moguće nadogradnje opreme (eng. *upgrades*), kao i troškove zamene zastarele opreme.

Dva najčešća scenarija koji povećavaju troškove hardvera su prekomerna i nedovoljna nabavka hardvera (eng. *over-buying i under-buying*). Prekomerna nabavka se odnosi na scenario kada se kupuju računarske komponente koje imaju više kapaciteta nego što je potrebno, pa su takvi kapaciteti uglavnom neiskorišćeni. Nedovoljna nabavka je situacija kada računarske komponente jedva zadovoljavaju minimalne uslove (eng. *minimum requirements*) operativnih potreba. U obe situacije se troši više nego što je potrebno.

Takođe, često se kupuju jake serverske platforme da bi se opslužio saobraćaj u periodima kada je najveća iskorišćenost resursa. U ovom slučaju, takav server je 90% operativnog vremena neiskorišćen pošto je tada saobraćaj nižeg intenziteta. U tradicionalnoj infrastrukturi, dinamička alokacija resursa prema potrebama je teško izvodljiva, a u većini slučajeva i nemoguća.

Kupovina klijentskih računara koji zadovoljavaju samo minimalne uslove takođe može da vodi ka dugoročnom povećanju troškova. U vremenskim periodima kada je saobraćaj intenziviran, takvi sistemi imaju problem sa performansama, a često dolazi i do pada (eng. *system crash*), što ugrožava izvršavanja kritičnih poslovnih operacija.

Troškovi objekata za skladištenje opreme

Ova kategorija troškova je često „skrivena" u organizacionom budžetu iza opštih troškova. Međutim, bez obzira na njihovu skrivenu prirodu, ovi troškovi su značajni i obuhvataju sledeće komponente:

- električnu energiju. Računarski sistemi ma koliko bili energentski efikasni, ipak troše električnu energiju. Tako na primer server čak i kada ništa ne radi (eng. *idle*) ipak troši ovaj značajni resurs,
- hlađenje. Za osetljivu računarsku opremu kao što su serveri potrebna je sofisticirana oprema za hlađenje (eng. *cooling equipment*). Sa rastom obima aktivnosti raste i broj servera, a samim tim i troškovi sistema za hlađenje i
- prostor. Za računarsku opremu je potreban prostor, i sa rastom obima opreme, raste i prostor koji treba da se obezbedi za čuvanje te opreme. Taj prostor predstavlja oportunitetni trošak zbog toga što su sredstva izdvojena za opremu mogla da budu alocirana za druge potrebe.

Troškovi upravljanja

Troškovi upravljanja i održavanja *IT* infrastrukture se stalno povećavaju. Što je veća i složenija infrastruktura, potreban je veći broj ljudi koji će njome upravljati. Svaka promena u infrastrukturi implicira trošak za njeno rekonfigurisanje. Često postoji i skriveni trošak koji nastaje zbog utrošenog vremena za implementiranje i testiranje promena, kao i zbog potrebe za ispravljanjem grešaka.

2.3. PREVAZILAŽENJE BARIJERA ZA IMPLEMENTIRANJE KONVERGIRANE INFRASTRUKTURE

Upravljačka struktura u organizaciji, kao i *IT* odeljenje, su ponekada protiv implementacije savremenih tehnoloških rešenja kao što je konvergirana infrastruktura. Kao najčešći razlog navodi se činjenica od većina kompanija ima ograničen budžet za obuku i trening svog stručnog osoblja, pa ne mogu da obezbede ni obuku svog *IT* osoblja i zato se oslanjaju na partnere i kompanije koje pružaju usluge održavanja i koje im pomažu u planiranju, implementaciji i upravljanju svojim starim *IT* okruženjem (eng. *legacy IT environments*). U ovakvim okruženjima kompanije za administriranje sistemom najčešće koriste svoj mali *IT* tim zajedno sa spoljnim konsultantima. Normalno je da se u takvim uslovima *IT* brine za svoj položaj, a samim tim se i protivi tehnološkim promenama.

Mnoge kompanije, među kojima je i *HP* pružaju konsultantske i usluge podrške svojim klijentima za implementaciju konvergirane infrastrukture u svojim okruženjima. *HP* usluge implementacije konvergirane infrastrukture (eng. *HP Converged Infrastructure Implementation Services*) omogućavaju klijentima da u potpunosti implementiraju, integrišu i optimizuju konvergiranu infrastrukturu pomoću *HP* konvergiranih sistema (eng. *HP Converged Systems*). *HP* konvergirani sistemi se isporučuju po principu „ključ u ruke", kao sistemi koji su već testirani i analizirani u realnim proizvodnim okruženjima (eng. *production environments*). Praksa je pokazala da ovako opisane konsultantske usluge spoljnih partnera uspešno ruše barijere za postavku konvergirane infrastrukture.

2.4. KONVERGIRANA INFRASTRUKTURA

Za bolje razumevanje pojma konvergirane infrastrukture, najbolje je pogledati praktičan primer iz prakse kompanije *HP*, kao jednog od pionira uvođenja ovog novog koncepta organizovanja računarskih resursa.

HP konvergirana infrastruktura nudi način za ubrzavanje procesa kreiranja poslovnih vrednosti tako što pomaže korporacijama u prevazilaženju nefleksibilnosti i visokih troškova računskih centara, aplikacija, i *IT* „bujanja". Ovo se postiže kreiranjem nacrta „računarske mreže budućnosti" koja integriše servere, skladišta, mrežne komponente, sigurnost, sisteme za snabdevanje električnom energijom, sisteme hlađenja i prostorija u deljeni skup interoperabilnih resursa. Svim navedenim resursima se upravlja kroz zajedničku upravljačku platformu. Navedeni pristup pomaže korporacijama da ubrzaju proces isporuke aplikacija i sprovođenja poslovnih inovacija na takav način da je moguće predvideti troškove uz efikasno korišćenje svih *IT* resursa, prostorija za skladištenje opreme i zaposlenih.

> Konvergirana infrastruktura je efektivno pakovanje procesa, koji vrše isporuku ponude dobavljača u formi hardverskog steka, koji je unapred testiran i validiran od strane proizvođača. Konvergirani sistemi se sastoje iz servera, skladišta i mrežnog hardvera i obično se isporučuju kao je jedan rek (eng. *rack*), ili kao jedan proizvod.

Primenom koncepta konvergirane infrastrukture dobija se definitivno više nego što se dobija ulaganjem u klasičan *IT*. Konvergirana infrastruktura utiče na sledeće komponente:

- *hardver*. U uslovima konvergirane infrastrukture podudaranje između nabavke hardvera sa stvarnim hardverskim potrebama je daleko bolje nego u tradicionalnim sistemima, uz mogućnost dinamičkog alociranja računarskih resursa u cilju zadovoljavanja novonastalih poslovnih potreba. Konvergirane tehnologije omogućavaju da se uradi više korišćenjem manjeg broja fizičkih servera nego u slučaju tradicionalnog *IT*-ja,
- *prostorije za skladištenje IT opreme*. Manji broj fizičkih servera direktno utiče na manje zahteve u pogledu infrastrukture. Iako je u kratkom roku potrebno investirati u novu opremu, u dugom roku ovaj sistem stvara velike uštede i

- *upravljanje*. Konvergirana infrastruktura omogućava centralizovano upravljanje *IT* resursima, a u određenim slučajevima i automatizaciju ovih aktivnosti. Ovo direktno smanjuje troškove i napor koji se ulaže u upravljanje *IT* komponentama.

Implementacija konvergirane infrastrukture je cilj koji treba da se dostigne, ali i dugotrajan proces. *HP* definiše tranziciju od tradicionalnog do konvergiranog *IT*-ja kroz petofazni proces koji je prikazan na Slici 2-2.

Slika 2-2: *Tranzicija od tradicionalnog IT-ja ka konvergiranom okruženju*



U prvoj fazi vrši se standardizacija *IT* operacija i konsolidacija aplikacija. U ovoj fazi prednosti nove infrastrukture postaju uočljivije, pošto složenost upravljanja *IT* infrastrukturom opada. S druge strane, standardizacija je potreban uslov za sprovođenje automatizacije u drugoj fazi.

Druga faza obuhvata procese virtuelizacije i automatizacije. Poslednje tri faze se odnose na razvoj klasične *IT* arhitekture ka arhitekturi orjentisanoj ka servisima (eng. *service-oriented architecture*), kao što su usluge koje se isporučuju putem klaud računarstva.

Međutim, bez obzira što je konvergirana infrastruktura sofisticiranija i efikasnija tehnološka platforma od tradicionalne *IT* infrastrukture, konvergirani sistemi nisu optimalni za sve kompanije. Prilikom donošenja odluke o tome da li korporacija treba da implementira konvergiranu infrastrukturu, potrebno je postaviti i dati odgovore na sledeća pitanja:

- da li postojeća tehnološka platforma zadovoljava zahteve *IT*-ja i poslovanja?
- ako se poslovanje prebaci na novu platformu sada, ili u bliskoj budućnosti, koji je najbolji rezultat koji može da se postigne, i kako postojeće iskustvo može da se integriše sa novom platformom?
- kako će *IT* da izbegne rizike koji se javljaju tokom tranzicije. Koliko dugo će proces tranzicije da traje?
- koliki će biti troškovi, a koliki povraćaj od investicije u konvergiranu infrastrukturu?

Potrebno je još napomenuti da fazni proces migracije sa tradicionalne na konvergiranu infrastrukturu počinje u onoj organizaconoj jedinici gde za to postoji najveća potreba. Izbor početne organizacione jedinice po pravilu bira *IT* odeljenje kompanije.

2.4.1. Implmentacija mehanizma otpornosti na greške u konvergiranoj infrastrukturi pomoću virtuelizacije servera

Jedan od ključnih preduslova za uspostavljanje održive konvergirane infrastrukture je da se obezbedi pouzdanost (eng. *reliability*) i raspoloživost (eng. *availability*) računarskih resursa. Konvergirana infrastruktura sama po sebi implicira da se sistem oslanja na manji broj fizičkih servera, a time i na manji broj potencijalnih domena otkaza (eng. *failure domains*). U ovom slučaju negativne posledice zastoja fizičkog servera (eng. *downtime*) imaju još veći efekat, zato što svi virtuelni serveri koji se *hostuju* na istom fizičkom serveru koji je otkazao neće funkcionisati. Za više informacija o virtuelizaciji servera pogledati Glavu 3.

Domen otkaza je po definiciji skup usluga koje neće funkcionisati ukoliko određena komponenta otkaže.

Tako na primer, ako su četiri virtuelna servera *hostovana* na jednom fizičkom serveru kome je otkazala matična ploča, onda će i sve usluge koje pružaju virtuelni serveri da budu nedostupne. Ako bi s druge strane četiri virtuelne mašine bile raspoređene na dva fizička servera, tada bi u slučaju otkaza jedne fizičke mašine samo polovina usluga bila nedostupno.

Takođe, uska grla (eng. *bottlenecks*) u performansama sistema i mrežnim komunikacijama u konvergiranoj infrastrukturi mogu biti još „skuplja" i frustrirajuća nego u tradicionalnoj konfiguraciji.

Na osnovu navedenog, dolazi se do zaključka da je u konvergiranoj infrastrukturi jedan od prioriteta da se osigura otpornost na greške (eng. *fault tolerance*), visoka raspoloživost i pouzdanost infrastrukture. Dve tehnologije koje pomažu u ostvarivanju ovih ciljeva su uparivanje mrežnih kartica (eng. NIC *teaming*) i klasterovanje (eng. *clustering*).

Tehnologijom uparivanja mrežnih kartica koja se u tehnološkoj terminologiji još sreće pod nazivima agregacija linkova (eng. *link aggregation*) i kreiranje timova mrežnih kartica, postiže se veća propusna moć mrežnih komunikacija. Ova tehnologija omogućava da dva mrežna adaptera rade zajedno tako što dele opterećenje na komunikacionom linku i na taj način udvostručuju raspoloživi propusni opseg mreže.

Mnogi *vendori*, uključujući i *HP* nude softverska rešenja koja podržavaju tehnologiju uparivanja mrežnih kartica. Na Slici 2-3 data je grafička reprezentacija ove tehnologije na primeru veze između servera i mrežnog komutatora. Kao napomena navodi se da mrežni adapteri koji se povezuju na ovaj način moraju da budu od istog proizvođača, a u većini slučajeva treba da budu i istog modela.

Za konfigurisanje timova mrežnih kartica, mrežnim administratorima je na raspolaganju nekoliko opcija (modova). Raspoloživi modovi uparivanja prikazani su u Tabeli 2-1.

Slika 2-3: *Tehnologija uparivanja mrežnih kartica*

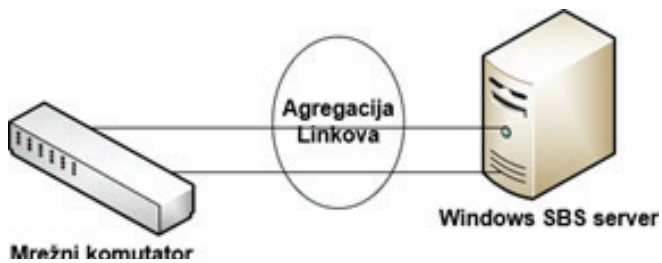


Tabela 2-1: *Vrste uparivanja mrežnih adaptera*

Mod uparivanja	Opis
Automatski	Sistem automatski bira tip uparivanja u zavisnosti od konfiguracije mrežnih adaptera.
802.3ad dinamički sa otpornošću na otkaze (eng. *dynamic with fault tolerance*)	Mrežni komutator postavlja članove tima u odgovarajući zbirni kanal (eng. *trunk channel*). Opterećenje poslatih paketa se automatski balansira pomoću drajvera koji kontroliše uparivanje. Opterećenje pri primanju paketa balansira komutator.
Balansiranje tereta pomoću komutatora sa otpornošću na greške (eng. *Switched-assisted Load Balancing with Fault Tolerance - SLB*)	Opterećenje poslatih paketa se automatski balansira putem drajvera koji kontroliše uparivanje. Opterećenje pri primanja paketa balansira komutator. Ne postoji eksplicitno određen primarni adapter.
Balansiranje tereta poslatog saobraćaja sa otpornošću na greške (eng. *Transmit Load Balancing with Fault Tolerance - TLB*)	Opterećenje poslatih paketa se automatski balansira putem drajvera koji kontroliše uparivanje. Sve pakete koje ne pripadaju grupi *IP* paketa šalje primarni mrežni adapter. Proces primanja paketa izvršava primarni adapter.
Balansiranje tereta poslatog saobraćaja sa otpornošću na greške i određivanje željenog redosleda (eng. *Transmit Load Balancing with Fault Tolerance and Preference Ord*)	Opterećenje poslatih paketa se automatski balansira putem drajvera koji kontroliše uparivanje. Ne *IP* pakete šalje primarni adapter. Pakete prima primarni adapter. Administrator može da podesi prioritet svakog uređaja tako što diferencira primarni od sekundarnog adaptera
Otpornost na greške samo za mreže (eng. *Network Fault Tolerance Only – NFT*)	Ne vrši se balansiranje opterećenja. Primarni adapter i šalje i prima pakete. Sekundarni adapter je pasivan i preuzima aktivnu ulogu samo u slučaju otkaza primarnog adaptera.
Otpornost na greške samo za mreže sa određivanjem željenog redosleda (eng. *Network Fault Tolerance Only with Preference Order*)	Ne vrši se balansiranje opterećenja. Primarni adapter i šalje i prima pakete. Sekundarni adapter je pasivan i preuzima aktivnu ulogu samo u slučaju otkaza primarnog adaptera. Administrator može da podesi prioritet svakog uređaja tako što diferencira primarni od sekundarnog adaptera.

Primenom tehnologije timova mrežnih kartica, pored određivanja tipa uparivanja, moguće je definisati i:

- metod balansiranja opterećenja koji se primenjuje,
- fizičke *MAC* (eng. *Medium Access Control*) adrese mrežnih adaptera koji su članovi tima i
- informacije o pripadnosti adaptera određenoj *VLAN* (*Virutal Local Area Network*) mreži.

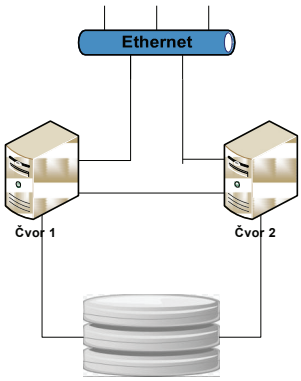
Tehnologija klasterovanja omogućava automatizovanu podršku za otpornost na greške, ubrzavanje performansi izračunavanja i balansiranje opterećenja izračunavanja između više servera. *Microsoft* serverski operativni sistemi podržavaju dve tehnike klasterovanja:

- klaster za otpornosti na greške (eng. *failover cluster*) i
- klaster za balansiranje mrežnog opterećenja (eng. *Network Load Balancing – NLB*).

Svaki od navedenih tipova klasterovanja ima specifičnu ulogu u konvergiranoj infrastrukturi. Najveći nivo raspoloživosti servera može da se postigne postavljanjem klastera za otpornost na greške. Ovaj klaster se obično sastoji od dva ili više servera koji se nazivaju čvorovi (eng. *nodes*) i koji dele zajedničko skladište podataka (Slika 2-4).

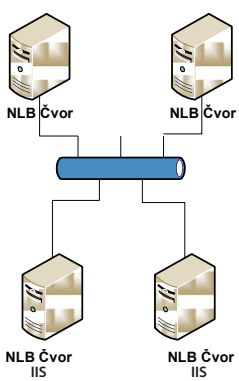
Zajedničko skladište može da se implementira kao direktno priključeno skladište (eng. *Directly Attached Storage – DAS*), mrežno skladište (eng. *Network Attached Storage – NAS*) i u formi skladišne mreže (eng. *Storage Area Network – SAN*). Serveri međusobno komuniciraju jednosmernom transmisijom (eng. *unicast*) i na svakih n sekundi svaki server proverava da li je drugi server operativan, tj. da li čuje njegov „otkucaj srca" (eng. *hearthbeat*). Ovaj tip klastera se obično koristi za podršku aplikacija za komunikaciju u realnom vremenu, servera baza podataka, kritičnih fajl servera i servera za štampanje.

Slika 2-4: *Klaster za otpornost na greške*



Klaster za balansiranje mrežnog opterećenja služi za obezbeđivanje veće raspoloživosti i performanse mrežnih aplikacija (Slika 2-5). Najčešće se koristi za podršku u radu *Web, FTP* i *VPN* servera i servera za distribuciju medijskog sadržaja (eng. *media streaming servers*). Svi serveri koji učestvuju u ovom klasteru formiraju virtuelni klaster. *Windows Server* platforma omogućava kreiranje klastera sa maksimalnih 32 čvora. Na svakom čvoru se nalazi kopija aplikacije koja se izvršava u klasteru. Klasteru se pristupa preko jedinstvene *IP* adrese klastera, ali svaki čvor u klasteru ima i svoju jedinstvenu privatnu *IP* adresu. Ovaj tip klastera raspoređuje zahteve klijenata pojedinim serverima na osnovu trenutnog opterećenja. Ako neki od članova klastera otkaže, onda se opterećenje distribuira preostalim operativnim članovima klastera.

Slika 2-5: *Klaster za balansiranje opterećenja*



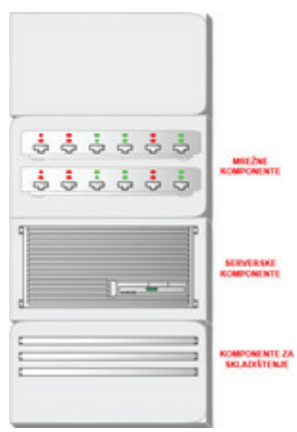
2.5. HIPER-KONVERGIRANA INFRASTRUKTURA

Svi principi koji se primenjuju u konvergiranoj infrastrukturi primenjuju se i u hiper-konvergiranoj instrastrukturi, samo što hiper-konvergirana infrastruktura ide korak dalje i predstavlja viši nivo razvoja ove relativno nove tehnologije.

Hiper-konvergirana infrastruktura (eng. *Hyper-Converged Infrastructure – HCI*) je tehnologija novijeg datuma koju su promovisali poznati vendori klaud rešenja, među kojima se posebno ističe *Nutanix*. Hiper-konvergirana infrastruktura predstavlja novi način „pakovanja" infrastrukture računskog centra i njene osnovne karakteristike su troškovna efikasnost, jednostavnost upravljanja računarskim resursima i visoka skalabilnost.

Hiper-konvergirana infrastruktura najbolje može da se razume ako se pogleda njena evolucija od tradicionalne *IT* infrastrukture. Tradicionalni računski centri sastoje se od nezavisnih serverskih komponenti, mrežnih komponenti i komponenti za skladištenje podataka, gde se sistemi različitih proizvođača usaglašavanju sa industrijskim standardima (Slika 2-6). Organizacije čiji su računarski resursi organizovani u formi tradicionalne infrastrukture mnogo vremena troše na testiranje kompatibilnosti i interoperabilnosti između komponenti različitih proizvođača u cilju kreiranja pouzdanog i robusnog sistema.

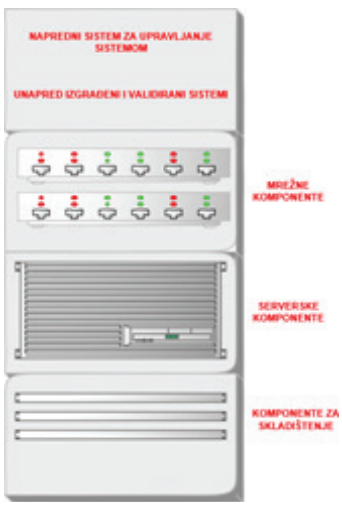
Slika 2-6: *Tradicionalni računski centar*



Kao što je već i navedeno, jedan od načina za prevazilaženje nedostataka koje ispoljava tradicionalna *IT* infrastruktura jeste implementiranje konvergirane infrastrukture. U konvergiranim sistemima jedan ili više proizvođača definšu i validiraju dizajn sistema i na taj način isporučuju unapred testirane i robusne sisteme koji su spremni za rad. Na ovaj način se vreme izgradnje sistema u značajnoj meri smanjuje.

Sistemi konvergirane infrastrukture se isporučuju sa naprednim sistemom za upravljanje (eng. *advanced management system*) koji olakšava proces upravljanja sistemom, ali i samo podešavanje sistemskih parametara. Kao implikacija ovoga proističe značajno ubrzavanje implementacije aplikacija i servisa uz dodatnu prednost da proizvođači nezavisnih komponenti zajedno rade na obezbeđivanju podrške klijentima za održavanje konvergiranih sistema. Konvergirani računski centar prikazan je na Slici 2-7.

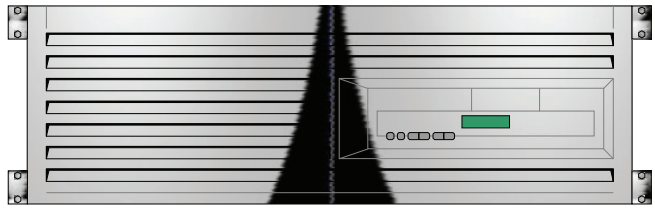
Slika 2-7: *Konvergirani računski centar*



Hiperkonvergirani sistemi idu još jedan korak dalje u odnosu na konvergirane sisteme. Ovi sistemi „pakuju“ servere, mrežne resurse i skladišta u jedno kućište (Slika 2-8). Sve tri navedene komponente se virtuelizuju pomoću softvera i na taj način se dobija jedinstven sistem koji je usklađen sa industrijskim standardima (o tehnologiji virtuelizacije će biti više reči u Glavi 3). Kompanije koje koriste hiperkonvergiranu infrastrukturu beleže značajan pad troškova nabavke i upravljanja računarskim resursima.

Slika 2-8: *Hiper-konvergirani računski centar*

VIRTUELIZOVANE KOMPONENTE U JEDNOM KUĆIŠTU

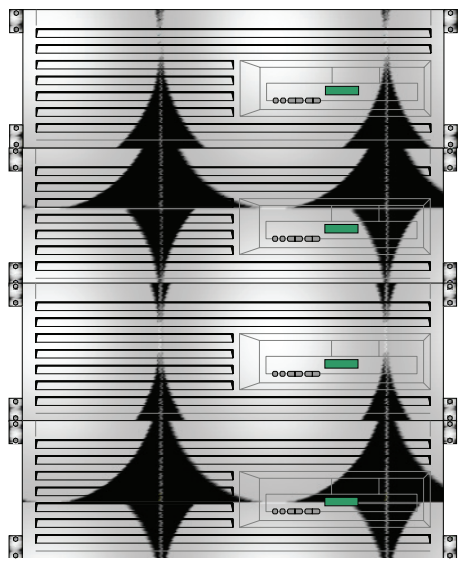


Hiperkonvergirani sistemi su dizajnirani na takav način da lako mogu da se proširuju, što znači da ako je potrebno da se skalira ovakav sistem, jednostavno se dodaju nova kućišta ili čvorovi (Slika 2-9). U ekspertskim krugovima iz oblasti računarskog hardvera smatra se da je hiperkonvergirana infrastruktura „fenomen" u razvoju, čiji je napredak motivisan jednostavnošću, fleksibilnošću i niskim troškovima.

Među najpoznatijim predstavnicima rešenja hiperkonvergirane infrastrukture ističu se *VMware, Nutanix, Cisco, Hitachi, Dell EMC* i *Scale Computing*.

Slika 2-9: *Skalabilnost hiper-konvergiranog računskog centra*

VIRTUELIZOVANE KOMPONENTE U JEDNOM KUĆIŠTU



Laka proširivost
Visoka skalabilnos

Jaki procesori, velika količina *RAM* memorije, napredne tehnologije za skladištenje podataka, interni *SSD* (*Solid State Drive*) diskovi konvergiraju u *x86* server koji predstavlja jedinstveni hiper-konvergirani sistem. Na ovaj način kompanije štede novac, jer ne moraju da kupuju vlasničke računarske komponente (eng. *proprietary computer components*). S druge strane u jednom organizovanom i integrisanom okruženju upravljanje je mnogo jednostavnije i zahteva mnogo niži nivo ekspertize u obavljanju svakodnevnih operacija, kao što je na primer podešavanje hardvera. Na ovaj način, organizacija koja svoj računski centar organizuje u formi hiper-konvergirane infrastrukture postaje agilnija i mnogo brže reaguje na poslovne promene i izazove.

Hiper-konvergirani sistemi integrišu virtuelizaciju servera, skladišta i mreža u formi jedinstvene hardverske ponude. U literaturi se računski centar koji implementira hiper- konvergiranu infrastrukturu često sreće pod nazivom softverski definisani računski centar (eng. *Software Defined Data Center – SDDC*). Ovakvi računski centri osim virtuelizacije servera svoj rad baziraju i na softverski definisanim skladištima (eng. *Software Defined Storage – SDD*) i softverski definisanim mrežama (eng. *Software Defined Network – SDN*). Prikaz jednog ovakvog računskog centra dat je na Slici 2-10.

Softverski definisani računski centar je mnogo fleksibilniji i agilniji od drugih modela. Tako na primer, u slučaju povećanog mrežnog saobraćaja i zahteva klijenata, dovoljno je samo da se implementira još jedna instanca virtuelne mašine koja će obraditi povećani broj zahteva za korišćenjem računarskih resursa.

U hiper-konvergiranim sistemima mrežne usluge se i dalje isporučuju eksterno pomoću namenskih komutatora (eng. *dedicated switches*), ali se većina saobraćaja između internih virtuelnih mašina izvršava preko softverski definisanih mreža na nivou hipervizora. U hiper-konvergiranim sistemima usluge skladišta se isporučuju preko hiperzvizora koji se izvršava u čvoru, ili u virtuelnoj mašni u čvoru. Otpornost se postiže replikacijom podataka na više čvorova.

Slika 2-10: *Softverski definisani računski centar*



Hiper-konvergirani sistemi još više smanjuju administrativne i operativne troškove konvergiranih infrastruktura. Implementacija ovakvih sistema je mnogo lakša od implementacije konvergiranih i bazira se na „uključi i radi" (eng. p*lug and play*) principu uz minimalnu potrebu za dodatnim konfigurisanjem.

Potrebno je da se napomene da se u svetskoj naučnoj i stručnoj javnosti vodi debata o razlikama, sličnostima, prednostima i nedostacima konvergiranih i hiper- konvergiranih sistema, gde često postoje oprečna mišljenja. Ovakva situacija ne čudi, jer je ova tehnologija tek u prvoj deceniji svog razvoja.

Zbog svoje prirode i prednosti u odnosu na klasičnu organizaciju *IT* resursa, rešenja konvergirane i hiper-konvergirane infrastrukture potpomognuta tehnologijom virtuelizacije predstavljaju osnovu za razvoj klaud računarstva. Kako klaud provajderi, tako i druge kompanije u implementaciji svog privatnog klaud okruženja često koriste ove tehnologije već dugi niz godina.



PITANJA ZA RAZMATRANJE

1. Objasniti pojam konvergirane *IT* infrastrukture. Zbog čega su se kompanije okrenule prema ovakvoj *IT* infrastrukturi? Objasniti i navesti razloge.
2. Koji su glavni nedostaci tradicionalne *IT* infrastrukture?
3. Na koje komponente utiče konvergirana *IT* infrastruktura? Objasniti.
4. Navesti i objasniti faze tranzicije *IT* infrastrukture prema definiciji kompanije *HP*.
5. Objasniti tehnologije klasterovanja. Zašto su ove tehnologije bitne?
6. Objasniti pojam hiper-konvergirane *IT* infrastrukture.
7. Navesti razlike između konvergirane i hiper-konvergirane *IT* infrastrukture.

3. TEHNOLOGIJA VIRTUELIZACIJE

Virtuelizacija (eng. *virtualization technology*) je oprobana i dokazana tehnologija koja ubrzano transformiše *IT* koncept u celini i menja način na koji ljudi koriste računare i *IT* tehnologije. Virtuelizacija je efikasan alat i način da se ostvare uštede u hardveru. Ova tehnologija se koristi i za potrebe razvoja i testiranja, kada zaposleni u *IT* odeljenjima kompanija mogu da konfigurišu i probaju nova tehnološka rešenja u izolovanom, virtuelizovanom okruženju, a da pritom ne utiču na produkcionu mrežu i servere.

Pretragom raspoložive literature vidi se da postoje brojne definicije pojma virtuelizacije. U nastavku će biti navedene samo neke od njih.

> Pojam virtuelizacije odnosi se na tehnologije koje obezbeđuju nivo apstrakcije između računarskog hardvera i softvera koji se na njemu izvršava.

Na ovaj način virtuelizacija daje logički pogled na računarske resurse, za razliku od klasičnog fizičkog pogleda, što dalje između ostalog omogućava i sledeće: operativni sistem „vidi" grupu fizičkih servera kao jedan skup računarskih resursa i više operativnih sistema može simultano (pseudo-paralelno) da se izvršava na jednoj fizičkoj mašini (serveru).

Virtuelizacija podrazumeva enkapsulaciju i apstrakciju računarskih resursa, kada se oni koriste na način koji odgovara određenoj primeni. Tehnologije virtuelizacije osim toga što omogućavaju bolju iskorišćenost *IT* infrastrukture, tako što se isti hardverski resursi mogu pseudo-paralelno koristiti u različitim sistemima, značajno doprinose i bezbednosti i pouzdanosti računarskih sistema.

Jedna od definicija pojma virtuelizacije koja se najčešće koristi navodi da se virtuelizacijom omogućava pokretanje više različitih instanci operativnih sistema koji dele zajedničke hardverske resurse. U kontekstu ove definicije najviše korisnika koristi tehnologiju virtuelizacije, na primer pokretanje operativnog sistema *GNU/Linux* u *Windows* okruženju.

Glavna snaga virtuelizacije leži u particionisanju (eng. *partitioning*), kada se jedan fizički server deli na više logičkih servera. Kada je fizički server podeljen, onda na svakom logičkom serveru mogu nezavisno da se izvršavaju operativni sistem i aplikacije.

Početkom devedesetih godina prošlog veka, virtuelizacija se uglavnom koristila za kreiranje različitih okruženja krajnjih korisnika (eng. *end-user environment*) na jednom *mainframe* hardveru. Tako na primer, ako bi *IT* administrator hteo da testira kako se novi softver izvršava na *Windows NT* ili *GNU/Linux* platformama, on bi testiranje vršio u okviru izolovanih okruženja kranjih korisnika.

Međutim, sa razvojem *x86* arhitektura i jeftinih *PC* platformi, virtuelizacija je dobila potpuno nove primene. Sa razlogom zasluge za ponovno „rođenje" virtuelizacije mogu da se pripišu kompaniji *VMware*, koja je razvila prvi hipervizor za *x86* arhitekture još početkom devedesetih godina prošlog veka i na taj način postavila osnovu za eksponencijalni razvoj tehnologija virtuelizacije i za široki dijapazon njenih primena.

Prema jednom stanovištu virtuelizacija obuhvata četiri oblasti:

- virtuelizacija servera (eng. *server virtualization*),
- virtuelizacija klijenata (eng. *client virtualization*),
- virtuelizacija mreža (eng. *network virtuelization*) i
- virtuelizacija skladišta (eng. *storage virtuelization*).

Virtuelizacija danas igra ključnu ulogu u konvergiranoj infrastrukturi i klaud računarstvu. Tako na primer, pre primene virtuelizacije, mnogi računski centri su koristili servere i skladišne uređaje koji su radili sa samo 10%, ili čak i manje kapaciteta. Primenom tehnologija virtuelizacije vrši se transformacija resursa računskog centra tako što se resursi alociraju na ona mesta, i u ono vreme kada su najviše potrebni. Na osnovu iskustava iz prakse, procenjeno je da virtuelizacija povećava iskorišćenost hardvera sa 10% - 15% na čak 70% - 80%.

Implementacija virtuelizacije u mrežno okruženje donosi mnoge prednosti, od kojih su najvažnije sledeće:

- proširivanje mogućnosti hardvera kroz efikasniju upotrebu hardverskih resursa,
- olakšano održavanje velikih klaster sistema (eng. *cluster systems*), kao što su farme servera,
- kontrola troškova kroz pojednostavljeno upravljanje infrastrukturom, putem konsolidacije platformi sa različitim operativnim okruženjima na manji broj fizičkih računara,
- omogućavanje izvršavanja različitih okruženja operativnog sistema zbog kompatibilnosti aplikacija na jednoj hardverskoj platformi i
- povećana sigurnost i pouzdanost mrežnih servera.

Tradicionalne infrastrukture su obično bazirane oko proizvodno-centričnog modela i oko softvera određenog vendora koji je instaliran i koji se održava u lokalu. Virtuelizacija otvara model deljenih usluga (eng. *shared-services model*) gde se svi troškovi dele između klijenata koji konzumiraju usluge. Korišćenjem modela deljenih usluga, može da se:

- ubrza standardizacija,
- smanje kapitalni troškovi;
- smanje operativni troškovi i
- ubrzaju i poboljšaju poslovni rezultati.

Najčešća implementacija virtuelizacije u malim i srednjim poslovnim sistemima (eng. *small medium business – SMB*) obuhvata virtuelizaciju računskog centra (servera) i klijenata. Oba pristupa obećavaju u pogledu smanjenja, kako troškova nabavke, tako i operativnih troškova uz istovremeno veći stepen kontrole nad *IT* okruženjem.

U nastavku su detaljno opisane gore navedene četiri oblasti virtuelizacije. Zbog najveće važnosti u rešenjima konvergirane arhitekture i klaud računarstva, posebna pažnja je posvećena virtuelizaciji servera. Na kraju ove glave taksativno su navedene i obrazložene sve prednosti koje tehnologija virtuelizacije omogućava.

3.1. VIRTUELIZACIJA SERVERA

Virtuelizacija servera je primarna komponenta koja se koristi u izradnji konvergiranog okruženja, koje se najčešće koristi u klaud računarstvu. U ovako postavljenom okruženju, virtuelne mašine (eng. *virtual machines – VMs*) se izvršavaju na računaru domaćinu (eng. *host computer*) i takvo okruženje omogućava da više servera koegzistira na jednoj hardverskoj platformi.

Virtuelne mašine imaju pristup hardverskim komponentama domaćina kao što su mrežne kartice (eng. *Network Interface Cards – NICs*) za pristup lokalnoj i *SAN* (*Storage Area Network*) mrežama. Virtuelna mašina je po definiciji računar u okviru računara koji se izvršava na isti način kao da je u pitanju fizički odvojen računar. Računar domaćin je hardver na kome se virtuelna mašina izvršava.

Virtuelne mašine su odvojene kako međusobno, tako i od računa domaćina, ali svi učesnici dele zajedničku hardversku platformu. Ne samo da više virtuelnih mašina može da se izvršava na istom domaćinu, već su podržani i različiti operativni sistemi i njihove verzije. Na ovaj način, u kontekstu servera, jedan domaćin može da izvršava veliki broj uloga na mreži.

3.1.1. Hipervizori i glavne tehnike virtuelizacije servera

Ključni element virtuelizacije je hipervizor (eng. *hypervisor*), koji se još naziva menadžer virtuelnih mašina (eng. *Virtual Machine Manager – VMM*). Hipervizor je komponenta koja je zadužena za kreiranje i izvršavanje virtuelnih mašina. Dva od najpoznatijih hipervizora su *Microsoft Hyper-V* i *VMware vSphere*. Paket *VMware vSphere*, koji obuhvata *ESXi* hipervizor, će kasnije biti detaljnije opisana sa fokusom na integraciju u konvergiranu infrastrukturu.

U najopštijem smislu, hipervizori se dele u dve grupe:

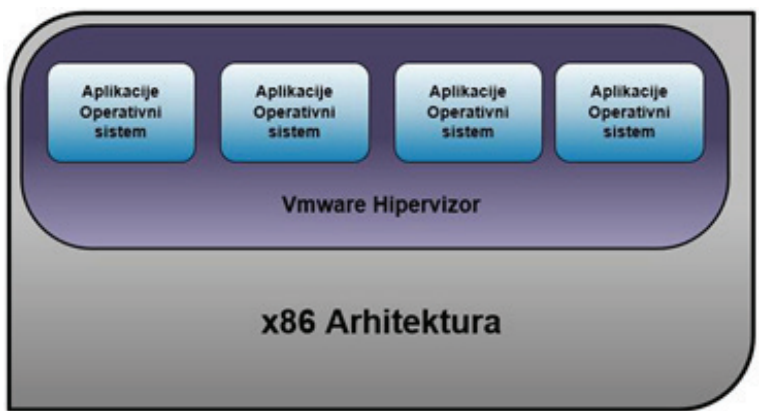
- hipervizor tip 1 (eng. *type 1 hypervisor*), koji je takođe poznat u literaturi i pod nazivima urođeni hipervizor (eng. *native hypervisor*) i goli hipervizor (eng. *baremetal hypervisor*) i
- hipervizor tip 2 (eng. *type 2 hypervisor*), koji se drugačije naziva ugošćeni hipervizor (eng. *hosted hypervisor*).

Hipervizor tip 1 se izvršava direktno na hardveru sistema. Ovaj hipervizor prikazan na Slici 3-1. Na slici je prikazana arhitektura *VMware* hipervizora. Urođeni hipervizor upravlja gostujućom mašinom koja se nalazi na nivou iznad njega. Na taj način hipervizor omogućava da gostujuća mašina ima direktan i neposredan pristup hardveru fizičkog računara.

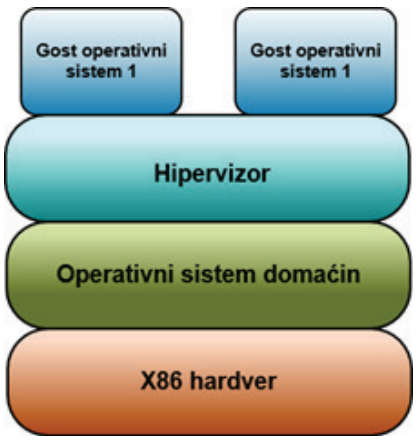
Tip 2 hipervizor se izvršava kao aplikacija iznad operativnog sistema domaćina. Operativni sistem domaćina predstavlja dodatni nivo između hadvera i hipervizora. Dakle, u slučaju ugošćenog hipervizora, gostujući operativni sistem se nalazi na jednom nivou više u odnosu na hardver, nego što je to slučaj kod hipervizora tip 1. Hipervizori tip 2 podržavaju brojne operativne sisteme, kao što su *Microsoft Windows, GNU/Linux* i *macOS*. *Oracle VirtualBox* i *VMware Workstation* su najpoznatiji predstavnici ovog tipa hipervizora.

Arhitektura hipervizora tip 2 data je na Slici 3-2.

Slika 3-1: *Hipervizor tip 1*



Slika 3-2: *Hipervizor tip 2*



Još jedna podela hipervizora odnosi se na način na koji hipervizori opslužuju zahteve gostujućih mašina za hardverskim resursima. Prema ovom kriterijumu, tehnologije hipervizora dele se u tri kategorije:

- potpuna virtuelizacija (eng. *full virtualization*);
- paravirtuelizacija (eng. *paravirtuelization*) i
- hardverski potpomognuta virtuelizacija (eng. *hardware-assisted virtualization*).

Kod tehnologije potpune virtuelizacije, hipervizor prevodi zahteve koji su dati u binarnom formatu za hardverskim resursima i prosleđuje ih direktno hardverskoj platformi. Glavna prednost ove tehnologije je u tome što gostujući operativni sistem „ne zna" da je virtuelizovan, pa ga nije potrebno modifikovati. Dakle, bilo koji operativni sistem može da radi u okruženju potpune virtuelizacije.

U slučaju korišćenja paravirtuelizacije, gostujući operativni sistem „zna" da je virtuelizovan, i on komunicira isključivo sa posebnim interfejsom sloja virtuelizacije. Glavni nedostatak ove tehnike jeste taj što je gostujući operativni sistem potrebno adaptirati kako bi mogao da se izvršava u virtuelnom okruženjom i zbog toga je pre korišćenja potrebno izvršiti modifikaciju operativnih sistema.

Hardverski potpomognuta virtuelizacija može da se implementira samo na računarima koji u svom procesoru imaju ugrađenu podršku za virtuelizaciju. Primer ovakvih podrška je *VT* (*Virtualization Technology*) set instrukcija kompanije *Intel* i *AMD-V* (*AMD Virtualization*) set instrukcija kompanije *AMD*. Ovaj vid virtuelizacije je najbrži i hipervizor šalje zahteve gostujućeg operativnog sistema direktno hardveru bez prethodne konverzije.

3.1.2. Softveri za virtuelizaciju servera

Glavna podela softvera za virtuelizaciju odnosi se na kriterijum dostupnosti softvera, pa je na osnovu toga glavna podela izvršena na vlasničke (eng. *proprietary*) i softvere otvorenog koda (eng. *open source software*).

Microsoft Hyper-V i *VMware vSphere* su neki od najpoznatijih primeri vlasničkih, tj. komercijalnih hipervizora. Kod ove vrste hipervizora, proizvođač ima potpunu kontrolu nad izvornim kodom hipervizora.

S druge strane, u slučaju hipervizora otvorenog koda, izvorni kod je dostupan svakome ko želi da ga modifikuje i personalizuje u skladu sa svojim potrebama. Čak više, tako modifikovani kod je moguće dalje distribuirati. Jedan od najpoznatijih predstavnika iz ove kategorije je svakako *Xen* paket alata za virtuelizaciju, a najpopularnija komponenta ovog paketa je *Xen* hipervizor. Takođe, nekoliko komercijalnih implementacija Xen proizvoda je dostupno, kao što je *Citrix XenServer*. Osnovna verzija ovog softvera je dostupna besplatno, dok za verziju sa dodatnim funckionalnostima i karakteristikama mora da se plati.

Još jedna implementacija *Xen*-a je *Oracle VM Server* za *x86* mašine. Kao i u prethodnom slučaju, osnovna verzija ovog softvera je besplatna, dok za unapređenu verziju softvera mora da se izdvoje finansijska sredstva. *Red Hat Enterprise Linux* u svom osnovnom paketu alata ima ugrađen *Xen* hipervizor.

Iako u tehničkom smislu nije hipervizor, kernel bazirana virtuelna mašina (eng. *Kernel-based Virutal Machine – KVM*) je mašina za virtuelizaciju koja je implementirana jedan sloj iznad *Linux kernel*-a. *KVM* obezbeđuje skoro iste funkcionalnosti kao pravi hipervizor, kao što je virtuelizacija hardvera računara domaćina. *KVM* je popularan na nekim distribucijama *Linux*-a zato što pruža podršku za veliki broj gostujućih operativnih sistema.

VMware vSphere

VMware je već jako dugo jedan od vodećih proizvođača tehnologija za virtuelizaciju (eng. *Virtuelization Technology – VT*). *VMware* nudi široku paletu proizvoda za virtuelizaciju koji zadovoljavaju razne potrebe. Neke od softverskih alata *VMware*-a:

- virtuelizacija servera i računskih centara (eng. *server and data center virtualization*),
- virtuelizacija *desktopa* (eng. *desktop virtualization*),

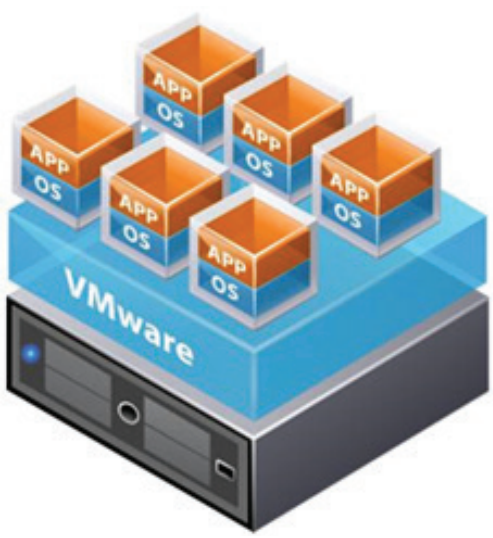
- računarstvo u oblaku (eng. *cloud computing*),
- platforma za aplikacije (eng. *application platform*),
- rešenja za male i srednje organizacije (eng. *SMB solutions*) i
- rešenja za *Apple Macintosh* (eng. *Apple Macintosh solutions*).

Karakteristike određenog rešenja varijaju od proizvoda do proizvoda i od platforme do platforme. Potrebno je naglasiti da *VMware* pokriva sve tri vodeće operativne platforme: *Windows, GNU/Linux* i *macOS*. Prilikom izbora konkretnog proizvoda, možda je najbolji način da se koristi *VMware* rešenje za izbor proizvoda (eng. *VMware product selector*). Ova aplikacija potencijalnom korisniku postavlja jednostavna pitanja u vezi mrežnog okruženja i potreba za virtuelizacijom, i kasnije koristi odgovore na postavljena pitanja za generisanje optimalnog predloga *VMware* softverskog proizvoda.

VMware vSphere je svakako najpopularniji *VMware* softverski paket za virtuelizaciju servera. Ovaj paket između ostalog obuhvata *ESXi* hiperzivor i *vCenter* server. *ESXi* je popularan hipervizor za dinamički i automatizovani računski centar. Ovaj hipervizor kao i *Microsoft Hyper-V* koristi tehniku potpune virtuelizacije za generisanje virtuelnog okruženja.

Arhitektura *ESXi* hipervizora prikazana je na Slici 3-3.

Slika 3-3: *Arhitektura VMware ESXi hipervizora*



Izvor: (slika preuzeta sa: https://www.vmware.com).

Glavne karakteristike *VMWare ESXi* hipervizora su:

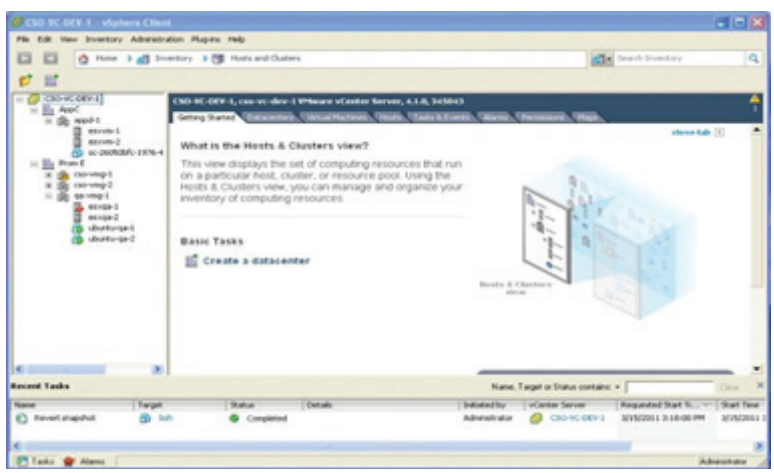
- podrška za standardne i distribuirane komutatore, uparivanje mrežnih karata (eng. *NIC teaming*) i virtuelne lokalne mreže,
- korišćenje *VMware vStorage VMFS* (*Virtual Machine File System*) za skladištenje fajlova virtuelnih mašina,
- funkcija *vMotion* za migraciju virtuelnih mašina s jednog na drugog fizičkog domaćina,
- pristup pomoću *vSphere* klijent komponente,
- upravljanje pomoću konzole koja podseća na *BIOS* (*Basic Input/Output System*) ili pomoću *vCLI* (*Virtual Command Line Interface*) i
- velika sigurnost primenom otiska softvera (eng. *software footprint*) koji zauzima samo 32MB skladišnog prostora.

Za pristup ESXi domaćinu na raspolaganju je nekoliko interfejsa:

- *vSphere Client*,
- *vSphere CLI*,
- *vSphere API/SDK (Application Programming Interface/Software Development Kit*) i
- *CIM* (*Common Information Model*).

Među pobrojanim, najčešća opcija koja se koristi za kreiranje i upravljanje virtuelnim mašinama kako na *ESXi* domaćinima, tako i na *VMware vCenter* rešenju za virtuelizaciju servera, je *vSphere Client*, čiji je *screenshot* prikazan na Slici 3-4.

Slika 3-4: *Screenshot - vSphere Client interfejs*



Potrebno je još da se istakne komponenta *VMware vCenter* koja omogućava da se sa više *ESXi* domaćina simultano upravlja, a koja između ostalog podržava i druge napredne funkcije, kao što su:

- *Storage vMotion* - mogućnost migracije virtuelne mašine sa jednog skladišta na drugo,
- distribuirano raspoređivanje resursa (eng. *Distributed Resource Scheduler - DRS*) - automatsko balansiranje tereta klastera *ESXi* domaćina pomoću *vMotion* funkcionalnosti i
- visoka dostupnost (eng. *Hight Availability - HA*) - u slučaju da jedan server u klasteru otkaže, aktivnosti svih virtuelnih mašina će automatski (bez manuelne intervencije) da se nastave na drugom serveru.

3.1.3. Prednosti virtuelizacije servera kroz konsolidaciju

Jedna od najvećih prednosti virtuelizacije servera jeste što se svi servisi konsoliduju na manji broj fizičkih servera. Ovo u velikoj meri utiče na troškove hardvera i održavanje opreme koji se značajno smanjuju implementacijom ovakvog rešenja.

Konsolidacija servera takođe smanjuje troškove upravljanja i troškove administriranja servera i mreže. Jedan od razloga snižavanja ovih troškova jeste dostupnost brojnih upravljačkih alata koji omogućavaju centralizovano upravljanje i administriranje virtuelnih okruženja. S druge strane, savremeni serveri su dizajnirani na način da se lako integrišu u virtuelno okruženje. Konsolidacija takođe omogućava bolju iskorišćenost fizičkog prostora gde se nalazi računski centar, kao i optimizaciju potrošnje električne energije.

S druge strane, osim snižavanja pomenutih *IT* troškova, kreiranjem konvergirane infrastrukture kroz virtuelizaciju konsolidacijom servera, snižavaju se i drugi poslovni troškovi. Virtualizacija povećava fleksibilnost infrastrukture i omogućava pravovremeno reagovanje na promene u okruženju. Pod uslovom da postoji fizički računar sa dovoljno resursa, implementacija virtuelnog okruženja, a posebno nove virtuelne mašine je brz i jednostavan proces. Osim toga, resursi koje dele računari mogu da se dinamički alociraju po potrebi. Sve navedene prednosti virtuelizacije impliciraju manje „uzaludno potrošenih" (eng. *wasted*) *IT* resursa, dok se prinos na investicije povećava.

Kao dodatna poslovna korist navodi se da je tranzicija na virtuelno okruženje potpuno transparentna krajnjim korisnicima sistema, dok se obavljanje poslovnih aktivnosti u uslovima tranzicije odvija nesmetno.

Zbog svih navedenih prednosti, konsolidacija i virtuelizacija servera se masovno primenjuju u klaud okruženjima.

3.2. VIRTUELIZACIJA KLIJENATA

U konceptualnom smislu, virtuelizacija klijenata, ili kako je neki nazivaju *desktop* virtuelizacija, je slična virtuelizaciji servera. U ovom slučaju, klijent se izvršava kao virtuelna mašina na serveru domaćinu.

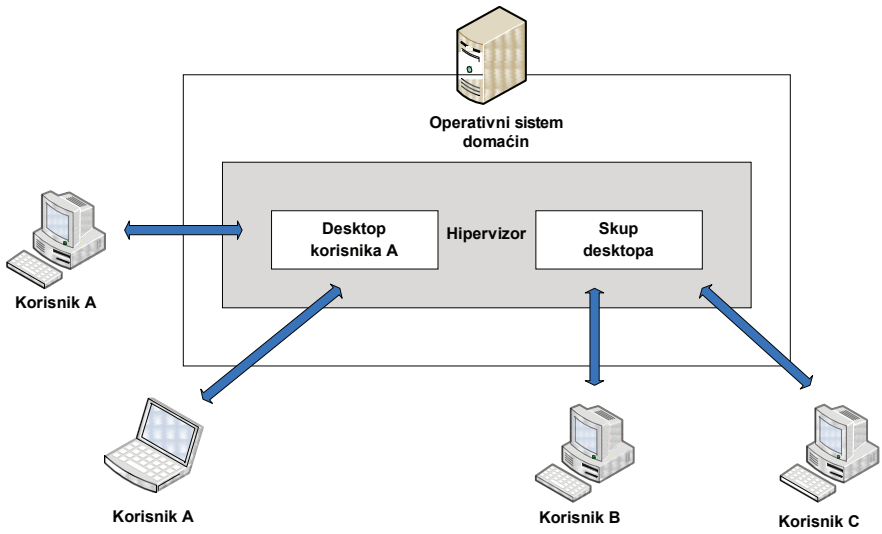
Zbog značaja za klaud računarstvo i konvergiranu infrastrukturu ovde će fokus da bude na infrastrukturi virtuelnog *desktopa* (eng. *Virtual Desktop Infrastructure – VDI*). Infrastruktura virtuelnog *desktopa* je vid virtuelizacije klijenata gde se virtuelne mašine kreiraju i čuvaju na serveru. Korisnici se povezuju na virtuelne mašine preko mreže. U literaturi se ovakva arhitektura naziva još i upravljiva desktop virtuelizacija (eng. *managed desktop virtualization*).

Pomoću tehnologije virtuelnog *desktopa* mogu da se naprave skupovi *desktop* okruženja (eng. *virtual desktop pools*) koji se koriste od strane više korisnika istovremeno, a takođe mogu da se naprave i lična *desktop* okruženja (eng. *personal desktop environments*) koja koriste individualni korisnici. U ovakvom okruženju korisnici se povezuju na virtuelne mašine putem standardnih mrežnih konekcija kao što su na primer udaljene konekcije korisnika (eng. *remote user connections*). Infrastru- ktura virtuelnog *desktopa* je prikazana na Slici 3-5.

Infrastruktura virtuelnog desktopa u značajnoj meri olakšava procese upravljanja softverom, nadogradnje softvera (eng. *software upgrades*) i podrške korisnicima zato što se sve virtuelne mašine otpremaju na jedan server. Drugim rečima, ništa se ne nalazi na klijent računarima, već se sve nalazi na serveru.

Ova tehnologija omogućava visok nivo fleksibilnosti u izboru klijentskih računara. Korisnicima nisu potrebni jaki računari da bi izvršavali zahtevne aplikacije. Korisnicima čak više ne treba ni klasični *PC* računar, već je dovoljan bilo koji kompatibilan uređaj koji podržava mrežno povezivanje, kao što su stoni računari slabijih performansi, tablet računari, pametni uređaji i tanki klijenti. Ova tehnologija sve više pospešuje primenu politike „ponesi svoj sopstveni uređaj" (eng. *Bring Your Own Device – BYOD*), koja se sve češće primenjuje u savremenim organizacijama.

Slika 3-5: *Virtuelizacija desktop okruženja*



Iako virtuelizacija klijenata, a posebno infrastruktura virtuelnog *desktopa*, može da zadovolji brojne zahteve okruženja, postoje scenariji kada je ipak potrebna implementacija klasičnog operativnog sistema na standardnom stonom računaru. Neki od ovakvih scenarija obuhvataju sledeće:

- potreba za izvršavanjem aplikacija koje zahtevaju grafički procesor, kao što je *3D* renderovanje,
- aplikacije koje zahtevaju mnogo memorije i procesorske snage,
- korišćenje specijalizovanog hardvera i
- postoji stalna potreba za korišćenjem računara, čak i u situacijama kada je nemoguće uspostaviti vezu sa serverom.

3.2.1. Prednosti virtuelizacije klijenata

Virtuelizacija klijenata ima podršku za klasične korisnike, ali takođe pruža i uslove za rad mobilnih korisnika. Korisnici često imaju potrebu da pristupe sistemima kao što su softver za kolaboraciju, e-učenje, unificirane komunikacije i razmenu poruka u realnom vremenu sa bilo koje lokacije i sa bilo kog uređaja.

Implementacija rešenja virtuelizacije klijenata kao integralnog dela klaud računarstva i konvergirane infrastrukture omogućava sledeće prednosti:

- povećana produktivnost radnika,
- niži troškovi opreme za *desktop* računare,
- važne korporativne informacije ostaju bezbedne iza zaštitnog zida korporativne mreže, čak i u slučajevima krađe uređaja krajnjih korisnika,
- mogućnost „zaključavanja" podataka u računskom centru, gde je sigurnost podataka veća nego na lokalnom disku,
- predupređenje mogućnosti da se osetljivi podaci preuzmu na lokalno skladište klijentskih uređaja,
- konzistentno i unificirano radno okruženje,
- poboljšanje integriteta podataka zato što se podaci skladište na serveru, a pravljenje rezervnih kopija vrše administratori, a ne obični korisnici,
- centralizovano upravljanje operativnim sistemima, nadogradnjama i zakrpama aplikacija (eng. *application patches*) i
- u situacijama kada se koriste tanki klijenti, zagrevanje radnog prostora je značajno slabije zbog toplote koju generišu računari.

Za mobilne korisnike konzistentno okruženje predstavlja najveću prednost i kao takvo značajno povećava produktivnost korisnika. Ovo je posebno izraženo kod korisnika koji deo radnog vremena provode radeći iz kancelarije, a preostali deo poslova obavljaju sa drugih lokacija.

3.3. VIRTUELIZACIJA MREŽA

Tradicionalne mrežne infrastrukture isključivo zavise od kabliranja. Fizička topologija predstavlja ograničenja za implementaciju mreže i mrežnih servisa. Rekonfigurisanje mreže implicira promene u kabliranju. Promene koje su donele virtuelne lokalne računarske mreže (eng. *Virtual Local Area Network – VLAN*) su omogućile konfigurisanje mreže samo modifikovanjem konfiguracije mrežnog komutatora, a da pritom nema potrebe za promenom kabliranja.

Zbog činjenice da je mrežna segmentacija zasnovana na logičkoj podeli mreže putem softvera, za rekonfigurisanje mreže je potrebno manje vremena i manje radne snage nego što je to slučaj u tradicionalnim okruženjima. Korišćenje virtuelnih mreža omogućava fleksibilnu infrastrukturu koja je sposobna da pravovremeno odgovara na sve promene u okruženju. S druge strane, logička konfiguracija može da izoluje osetljive podatke i aplikacije što značajno doprinosi poboljšanju sigurnosti cele *IT* infrastrukture.

Međutim, virtuelne mreže su samo „vrh ledenog brega" inovacija koje nastaju virtuelizacijom mreža. Tako na primer postoji mogućnost da se fizička mreža konfiguriše samo jednom, a da kasnije tokom njene eksploatacije mogu da se menjaju povezani uređaji bez potrebe za rekonfigurisanjem mreže. Neke od raspoloživih tehnologija koje ovo omogućavaju su implementirane u *HP Blade Server* rešenjima. Dva najznačajnija rešenja su *HP FlexFabric* i *HP Virtual Connect*.

3.4. VIRTUELIZACIJA SKLADIŠTA

Skoro sve aktivnosti računskog centra i poslovanja zavise od informacija. Aktivnosti sa podacima i informacijama se odnose na načine na koje se informacije distribuiraju korisnicima i aplikacijama, kako su zaštićene u cilju smanjenja rizika, i pokušaje da se dobije što više korisnih informacija iz podataka. Virtuelizacija skladišta omogućava bolje upravljanje skladištima, veću iskorišćenost resursa skladišta, kao i efikasnije generisanje i korišćenje informacija.

Dve osnovne tehnologije koje se implementiraju u virtuelnim skladištima su lokalna skladišna mreža (eng. *Storage Area Network – SAN*) i skladišta priključena na mrežu (eng. *Network Attached Storage – NAS*).

Ključne karakteristike lokalnih skladišnih mreža su:

- koriste se skladišta na nivou blokova podataka,
- postoji kompatibilnost sa različitim operativnim sistemima i fajl sistemima,
- operativni sistemi i aplikacije ne primećuju razliku između ovih tipova skladišta i lokalnih skladišta,
- povezivanje na skladišta se ostvaruje preko namenske računarske mreže (eng. *dedicated computer network*),
- optička lokalna mreža skladišta (eng. *Fibre Channel Storage – FC Storage*) obezbeđuje bolje performanse i
- lokalna mreža skladišta *iSCSI* (*Internet Small Computer System Interface*) može da se implementira po pristupačnijoj ceni od optičke mreže skladišta, a pritom se samo malo gubi na kvalitetu.

Implementiranje lokalne mreže skladišta je dobra opcija pošto ne zahteva promene u postojećim računarskim resursima kao što su operativni sistemi i aplikacije. Međutim, troškovi za implementaciju ove infrastrukture nisu zanemarljivi i ovaj tip skladišta ne spada u grupu jeftinijih rešenja.

Primena druge tehnologije virtuelnih skladišta, skladišta priključenih na mrežu, nosi sa sobom niže troškove implementacije, ali i slabije performanse nego prva tehnologija.

Ključne karakteristike skladišta priključenih na mrežu su:

- koriste se skladišta na nivou fajlova,
- fajl server ova skladišta vidi kao deljene mrežne lokacije (eng. *shared network places*),
- konekcija se uspostavlja putem standardne lokalne ili širokopojasne mreže i
- primena ove tehnologije ne zahteva značajne promene u postojećim operativnim sistemima i aplikacijama.

Tehnologija skladišta priključenih na mrežu se implementira uz niske početne troškove. Svi operativni sistemi koji podržavaju mrežne deljene lokacije mogu da se koriste sa ovim skladištima.

3.5. PREDNOSTI VIRTUELIZACIJE

Prednosti virtuelizacije mogu da se kategorišu na nekoliko načina. U ovom slučaju prednosti virtuelizacije biće posmatrane kao grupe i to u kontekstu klaud računarstva i konvergirane infrastrukture.

Kao što je ranije navedeno, virtuelizacija omogućava efikasnije korišćenje računarskih resursa, dok istovremeno pomaže u ostvarivanju poslovnih zahteva. Međutim, ovo je samo jedna od prednosti koje omogućava primena tehnologije virtuelizacije.

Kompanija *HP* identifikuje i druge prednosti, koje se zbirno mogu kvantifikovati na sledeći način: do 33% smanjenje *IT* operativnih troškova i do 200% brža isporuka servisa kranjim korisnicima.

Virtuelizacija takođe poboljšava elastičnost *IT* resursa i omogućava bolje reagovanje na promene u poslovnom okruženju. Za implementaciju virtuelne mašine je potrebno mnogo manje vremena nego za nabavku, instalaciju, integraciju i konfigurisanje fizičkog računara, što ima direktne implikacije na:

- fleksibilnost u odgovaranju na poslovne zahteve,
- manje vremena je potrebno da prođe od implementacije do generisanja prihoda i
- niže troškove implementacije.

Jedna od glavnih prednosti konsolidacije na osnovu virtuelizacije proističe iz sledećih ušteda na osnovu smanjivanja troškova:

- nabavke opreme,
- električne energije,
- održavanja prostorija i
- održavanja opreme i sistema.

U najgorem slučaju, kada se primenom tehnologije virtuelizacije troškovi ne smanjuju, troškovi se drže pod kontrolom i sprečava se njihov rast.

U literaturi se navodi da su poboljšanje u performansama i pouzdanosti sistema takođe dve značajne prednosti virtuelizacije. Efikasnija infrastruktura neposredno utiče na radnu efikasnost korisnika koji je koriste. Kada se javi potreba za promenom mrežnog okruženja, ili druge računarske opreme, period kada sistem neće biti operativan (eng. *system downtime time*) će biti kraće, a izgubljeno radno vreme biće svedeno na minimum.

Kompanije *HP, VMware, Amazon, IBM* i druge pružaju konsultantske usluge koje pomažu organizacijama da imaju bolji pregled svih prednosti i koristi, koje tehnologije virtuelizacije donosi. *HP* takođe pruža pomoć pri planiranju poslovnih potreba virtuelizacije, kao što su projektovani troškovi i povraćaj na investicije. *HP* u svojim konsultovanjima koristi preko 200 finansijskih i poslovnih modela kako bi generisao što tačnije i verodostojnije analize i statistike.

3.6. AUTOMATIZACIJA

Glavni cilj automatizacije je optimizacija produktivnosti i korišćenja resursa. Automatizacija obuhvata različite aspekte *IT* infrastrukture i upravljačkih aktivnosti. Svaki zadatak koji se ponavlja s vremena na vreme je „kandidat" za proces automatizacije. Automatizovan zadatak, ne samo da štedi vreme, već i smanjuje verovatnoću nastanka grešaka jer se svaki put izvršava na isti način.

Kada se govori o automatizaciji u kontekstu *IT* infrastrukture, uglavnom se misli na skladišni prostor i njemu slične resurse. Jedna od prednosti konvergirane infrastructure jeste mogućnost da se resursi realociraju i prilagođavaju automatski kao odgovor na promenu trenutnih opterećenja i zahteva, što predstavlja jedan od glavnih zahteva klaud okruženja.

Automatizacija takođe nalazi primenu i u implementaciji aplikacija. Primenom tehnika virtuelizacije, zaposleni u *IT* odeljenju kompanija definišu obrasce (eng. *templates*) koji određuju konfiguracije virtuelnih mašina koje zadovoljavaju definisane zahteve. Na taj način, ako se javi potreba za implementacijom neke aplikacije, ona može biti instalirana i konfigurisana za samo nekoliko minuta. Dakle, automatizacija povećava agilnost i fleksibilnost *IT* resursa.

Tehnologije virtuelizacije i automatizacije su komplementarne i često se paralelno koriste u rešenjima konvergirane infrastrukture za potrebe klaud okruženja. Kada se koriste zajedno, ove tehnologije generišu sledeće prednosti:

- IT infrastruktura odgovara brže na promene (eng. responsive),
- mrežni resursi postižu bolje performanse uz niže troškove i
- troškovi upravljanja i administracije se smanjuju pošto više ne postoji potreba za ručnom rekonfiguracijom mrežnih resursa.

Mnogi globalni vendori računarske opreme, servisa i softvera imaju svoje rešenje za automatizaciju. Tako na primer *HP* ima veliku ponudu softvera za automatizaciju servera i mrežnih resursa koji omogućavaju veću automatizaciju svih komponenti infrastrukture i mrežnog okruženja.



PITANJA ZA RAZMATRANJE

1. Definisati pojam virtuelizacije.
2. Objasniti zašto se virtuelizacija primenjuje. Dati primere.
3. Navesti oblasti koje obuhvata virtuelizacija. Ukratko objasniti svaku oblast.
4. Navesti prednosti implementiranja virtuelizacije u mrežno okruženje.
5. Objasniti koncept virtuelizacije servera. Dati primere.
6. Šta je hipervizor?
7. Koje sve vrste hipervizora postoje? Dati primere.
8. Objasniti šta je *Vmware vSphere* softver za virtuelizaciju servera. Koji alati se primenjuju?
9. Koje su prednosti virtuelizacije servera kroz konsolidaciju? Objasniti.
10. Objasniti koncept virtuelizacije klijenta. Navesti primer.
11. Navesti prednosti virtuelizacije klijenta.
12. Objasniti koncept virtuelizacije mreža.



KLAUD RAČUNARSTVO

DEO II

KLAUD RAČUNARSTVO



U drugom delu udžbenika dat je uvod u koncept klaud računarstva sa posebnim akcentom na tipovima klaud računarstva prema modelu usluge i prema modelu isporuke.

Termin „klaud računarstvo“ potiče od engleskog pojma *Cloud Computing*. Pretragom domaće literature iz oblasti računarstva može da se pronađe više prevoda ovog termina na srpski jezik, poput „računarstvo u oblacima“, „računarstvo među oblacima“, „klaud kompjuting“ i slično. Autori su odabrali termin „klaud računarstvo“ i u udžbeniku je konzistentno korišćen ovaj prevod.

Kako u javnim, tako i u stručnim i naučnim krugovima dominira mišljenje da je klaud računarstvo novi koncept isporuke računarskih usluga, čiji je razvoj počeo poslednjih godina. Međutim, ovo nije u potpunosti tačno, s pogledom na činjenicu da je za razumevanje bilo koje oblasti, pa tako i oblasti računarstva, potrebno poznavanje njene istorije. Sagledavanjem istorije računarstva dolazi se do zaključka da klaud računarstvo nije novi koncept, niti nova tehnologija, već da je klaud računarstvo primena starog koncepta u svetlu novih računarskih tehnologija i arhitektura.

Sličan koncept, ali pod drugim nazivom, primenjivan je još u vreme *mainframe* računara, kada su klijentski uređaji, tzv. glupi terminali, koristili resurse, tada moćnog, centralnog računara. Zbog navedenog, u ovom delu udžbenika data je paralela između razvoja klaud računarstva i drugih oblasti računarstva, gde jasno može da se uoči kako jedna oblast utiče na sve, ili na većinu drugih oblasti.

Prema jednoj od brojnih definicija klaud računarstva koje mogu da se nađu u literaturi, pojam klaud računarstvo odnosi se na isporuku računarskih resursa na zahtev preko Interneta, ili neke druge računarske mreže korišćenjem modela plaćanja na osnovu ostvarene potrošnje. Prema mišljenju autora, klaud računarstvo je previše širok koncept da bi mogao u potpunosti da se obuhvati samo jednom definicijom i zbog toga je u ovom delu udžbenika dato definisanje klaud računarstva iz nekoliko različitih aspekata, pre svega iz ugla posmatranja globalnih kompanija koje se bave razvojem servisa klaud računarstva, poput kompanija *IBM, Intel¸ Microsoft*, i drugih.

U glavi koja se bavi tipovima klaud računarstva prema modelu usluge, osim detaljnog opisivanja klaud modela koji se najčešće koriste, i koji se u većini slučajeva poistovećuju sa pojmom klaud računarstvo (softver kao usluga, infrastruktura kao usluga i platforma kao usluga), prikazani su i neki od novijih modela, poput komunikacije kao usluge, skladišta kao usluge i poslovnog procesa kao usluge. Navedeni noviji modeli klaud računarstva, suštinski i funkcionalno spadaju u osnovne modele i uglavnom se koriste u marketinške svrhe, kako bi klaud provajderi privukli veći broj individualnih i korporativnih klijenata.

U poslednjoj glavi ovog dela data je taksonomija klaud računarstva prema modelu isporuke, gde se diferenciraju dobro poznati modeli javnog, privatnog, hibridnog i zajedničkog klauda.

4. UVOD U KLAUD RAČUNARSTVO

Engleski termin *Cloud Computing* može da se prevede na srpski jezik na više načina, kao što su računarstvo u oblaku, računarstvo u oblacima, računarstvo među oblacima, klaud kompjuting, klaud računarstvo, itd. Bez obzira što u literaturi koja je dostupna na srpskom jeziku postoje brojni prevodi ovog engleskog termina, u ovom udžbeniku je konzistentno korišćen termin klaud računarstvo.

U ovoj glavi prikazan je kratak uvod u stari, tj. novi koncept klaud računarstva.

4.1. DEFINISANJE I POJAM KLAUD RAČUNARSTVA

Pretragom kako strane, tako i domaće literature vidi se da postoje brojne definicije pojma klaud računarstva. U nastavku će biti navedene samo neke od njih.

Nacionalni Institut za Standarde i Tehnologiju (eng. *National Institute for Standards and Technology – NIST*) u publikaciji pod oznakom *SP 800-145* klaud računarstvo definiše kao model koji omogućava pouzdan pristup na zahtev (eng. *on-demand*) zajedničkim skupu (eng. *shared pool*) računarskih resursa (mrežama, skladištima podataka, serverima, servisima i aplikacijama) koji se mogu konfigurisati putem računarske mreže, gde se resursi mogu brzo pribaviti i osloboditi, a da su pritom uloženi napor i interakcija davaoca usluge provajdera (eng. *service provider*) minimalni.

Za definisanje ovakvog modela klaud računarstva prema *NIST*-u primenjuje se pet osnovnih karakteristika, tri modela usluge i četiri modela primene. Karakteristike klaud računarstva prema ovom modelu svode se na sledeće:

- samousluživanje na zahtev (eng. *on-demand self-servicing*). Korisnik klaud usluga može da bude snabdeven računarskim resursima, kao što su na primer kapacitet mrežnog skladišta, ili serversko vreme, automatski, bez potrebe za neposrednom interakcijom sa davaocem (provajderom) usluge,

- širok mrežni pristup (eng. *broad network access*). Sve računarske mogućnosti su raspoložive putem računarske mreže i pristupa im se pomoću standardnih mehanizama uz široku podršku heterogenih platformi, debelih i tankih klijenata (na primer putem pametnih moblnih telefonona, tableta, lap top računara i radnih stanica),
- udruživanje resursa (eng. *resource pooling*). Svi resursi davaoca usluge su udruženi i opslužuju više klijenata istovremeno, a pritom se različiti fizički i virtuelni resursi dinamički dodeljuju i koriste prema tražnji korisnika. U ovom slučaju klijent ne zna gde se resursi koje koristi tačno nalaze, niti ima kontrolu nad njima, ali može da definiše njihovu apstraktnu lokaciju, kao na primer na nivou države ili računskog centra. Kao primeri ovakvih resursa navode se skladište podataka, snaga obrade podataka i mrežni protok.
- ubrzana elastičnost (eng. *rapid elasticity*). Računarski resursi se elastično obezbeđuju i oslobađaju, a u pojedinim slučajevima sve se vrši automatski, srazmerno prema zahtevima klijenata. Iz perspektive klijenata izgleda kao da su resursi neograničeni i mogu da se skaliraju u bilo kojem obimu u bilo koje vreme.
- izmerene usluge (eng. *measured service*). Korišćenje resursa može da se nadgleda, kontroliše i izveštava, što obezbeđuje transparentnost, kako za davaoca usluge, tako i za korisnika usluge.

Prema *NIST*-ovoj definiciji klaud računarstva, postoje tri klaud modela usluge: softver kao usluga (eng. *Software as a Service – SaaS*), platforma kao usluga (eng. *Platform as a Service – PaaS*) i infrastruktura kao usluga (eng. *Infrastructure as a Service – IaaS*). Takođe, prema istoj definiciji, četiri modela primene odnose se na: privatni klaud (eng. *private cloud*), zajednički klaud (eng. *community cloud*), javni klaud (eng. *public cloud*) i hibridni klaud (eng. *hybrid cloud*).

O modelima usluge i modelima primene više je prikazano u narednim glavama.

Prema definiciji klaud računarstva koju je dala Evropska Agencija za Sigurnost Mreža i Informacionu Sigurnost (eng. *European Network and Information Security Agency – ENISA*), ovaj pojam se odnosi na model usluge na zahtev za snabdevanje računarskim resursima koja se uglavnom bazira na tehnologiji virtuelizacije i distribuiranim računarskim tehnologijama.

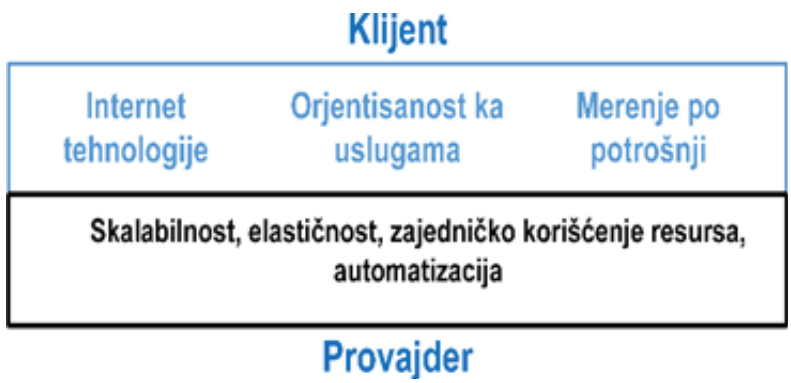
ENISA takođe naglašava da se klaud računarstvo ne sagledava kao nova tehnologija, već da ovaj pojam predstavlja nov način za isporuku računarskih resursa.

Prema *ENISA* definiciji, karakteristike klaud arhitekture su: visoko apstrahovani resursi, skoro trenutno snabdevanje uslugama, korišćenje zajedničkih resursa (hardvera, baza podataka, CPU vremena, memorije, itd.), usluge na zahtev, obično primenom sistema plaćanja prema nivou korišćenja (eng. *pay as you go*) i programsko upravljanje putem aplikativnog programskog interfejsa (eng. *Application Programming Interface – API*).

Prema zvaničnoj definiciji kompanije *Gartner*, klaud računarstvo je stil računarstva u kome se skalabilni i elastični računarski resursi dostavljaju klijentima (potrošačima) u vidu usluge korišćenjem Interneta i Internet tehnologija.

Takođe, prema *Gartner*-u, pet glavnih karakteristika klaud računarstva su: orijentisanost ka uslugama (eng. *service-based*), merenje po potrošnji, skalabilnost, elastičnost, zajedničko korišćenje resursa i korišćenje Internet tehnologija. Suština klaud računarstva je nevidljiva granica između klijenata i davaoca usluge, što može da se vidi na Slici 4-1.

Slika 4-1: *Koncept klaud računarstva prema Gartner-u*



Kompanija *IBM* (*International Business Machines*) navodi da je pojam klaud računarstva, koji se obično naziva samo oblak, isporuka računarskih resursa na zahtev preko Interneta korišćenjem modela plaćanja na osnovu ostvarene potrošnje. Pod resursima se u ovoj definiciji podrazumeva gotovo sve, od podataka, pa sve do računskih centara.

Dalje, kako se na zvaničnom sajtu kompanije *IBM* navodi, glavne karakteristike klaud računarstva su [4]:

- elastični resursi, koji se skaliraju brzo prema gore ili prema dole, kako bi se u realnom vremenu zadovoljile potrebe korisnika,
- računanje potrošnje usluga, gde korisnik plaća tačno onoliko resursa koliko je potrošio i
- samousluživanje, gde klijenti sami uzimaju dodatne računarske resurse.

IEEE (*Institute of Electrical and Electronics Engineers*) asocijacija za donošenje standarda (eng. *IEEE Standards Association – IEEE-SA*) je dala definiciju klaud računarstva u dva radna nacrta. Nacrt *P2301* (klaud profili) naglašava različite ekosisteme iz kojih se klaud sastoji, kao što su klaud prodavci (eng. *cloud vendors*), pružaoci usluga i korisnici. Nacrt *P2302* (međuoblak) definiše topologiju, funkcije i upravljačke aktivnosti između različitih okruženja klaud računarstva.

Kompanija *Microsoft*, kreator *Microsoft Azure* korporativne klaud platforme, definiciju i pojam klaud računarstva daje iz više aspekata. U nastavku će biti prikazana samo jedna od definicija ove kompanije koje mogu da se pronađu u literaturi.

Prema jednoj definiciji, kompanija *Microsoft* navodi da se definisanje koncepta klaud računarstva izvodi na osnovu tri komponente (nivoa): kako klaud radi, šta klaud označava i kakav klaud ima uticaj.

Prvi komponenta definisanja pojma klaud računarstva odnosi se na nivo „kako klaud radi". Kada se sagledava na ovom nivou, klaud računarstvo je sposobnost i mogućnost korišćenja skupa računarskih resursa u cilju isporuke aplikacija i usluga klijentima. Na ovaj način, klaud predstavlja pomeranje upravljačkih aplikacija sa nivoa određenog resursa, kao što su računari, serveri, memorija, skladišni prostor, na upravljanje na nivou aplikacija, gde se ovaj nivo može dinamički da se širi i sakuplja na zahtev klijenta, korišćenjem skupa zajedničkih resursa.

Definisanje pojma klaud računarstvo na nivou „kako klaud radi" je značajno za opšte razumevanje ovog pojma, međutim *Microsoft* daje prave definicije klauda na nivoima „šta klaud znači" i „kakav klaud ima uticaj".

Prema navodima kompanije *Microsoft*, na nivou „šta klaud označava" klaud računarstvo omogućava kompanijama da budu fleksibilnije i agilnije u zadovoljavanju sve zahtevnijih potreba savremenog poslovanja. Klaud pomera tradicionalne granice i ograničenja koja se odnose na alociranje usluga i na ovaj način računarski resursi mogu da se koriste u cilju generisanja efikasnog odgovora na specifične poslovne potrebe. Drugim rečima, korišćenjem klaud tehnologije, računarski resursi se koriste za proaktivno i inovativno reagovanje na novonastajuće situacije i zahteve savremenog poslovanja.

Konačno, sa stanovišta nivoa „kakav klaud ima uticaj", navodi se da klaud računarstvo ima neposredan uticaj na smanjivanje troškova nabavke novog hardvera i softvera. Na ovaj način, napori savremenog računarstva se usmeravaju na zajednički rad hardvera i softvera, dok su napori tradicionalnog računarstva bili fokusirani na nove verzije softvera i hardvera. Primenom klaud računarstva kreira se „organski" (živi) skup resursa, gde su informacioni sistemi otporniji na otkaze u radu hardvera i softvera.

U svojoj definiciji klaud računarstva, kompanija Intel uzima u obzir arhitekturu sistema i usluge koje se pružaju, uzimajući u obzir razlike između privatnih i javnih klaud okruženja.

Arhitektura klaud računarstva odnosi se na usluge i podatke koji egzistiraju u deljenom, dinamičkom i skalabilnom skupu resursa zasnovanom na tehnologijama virtuelizacije i/ili skaliranim aplikativnim okruženjima.

Usluge klaud računarstva odnose se na servise za individualne potrošače ili kompanije koji se realizuju putem javnog Interneta. Oslanjajući se na arhitekturu klaud računarstva, skaliranje usluga vrši se bez intervencije korisnika i obično se naplaćuje na osnovu ostvarene potrošnje.

Privatni klaud je arhitektura oblaka koja je locirana iza *firewall*-a organizacije i pruža informatičke usluge za internu upotrebu.

4.2. RAZVOJ KLAUD RAČUNARSTVA

Kao što je poznato, informatika i računarstvo su nauke koje su, posebno poslednjih decenija, postale deo svakodnevnog života sa razvojem i korišćenjem računara. Informatika i računarstvo (eng. *Computer Science - CS*) sastoje se iz većeg broja oblasti, kao što su: računarski hardver, arhitektura računara, računarski softver, računarske komunikacije, veštačka inteligencija, itd.

Za potpunije razumevanje razvoja bilo koje od oblasti računarstva koje su navedene gore, potrebna je cela i sveobuhvatnija slika razvoja celog računarstva, s obzirom da u računarstvu promena u bilo kojoj oblasti utiče na skoro sve druge oblasti. Slično važi i za klaud računarstvo, gde je za potpunije razumevanje pojma i koncepata klaud računarstva potrebno razumevanje glavnih promena u oblasti računarstva.

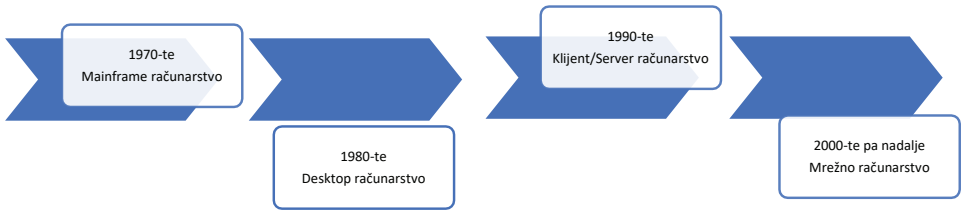
Zbog toga će u nastavku biti prikazane glavne transformacije (promene) u računarstvu u prethodnih pedesetak godina i njihove implikacije na nastanak i razvoj klaud računarstva.

4.2.1. Glavne transformacije u računarstvu

Prema viđenju kompanije *Oracle*, četiri važnije transformacije u računarstvu koje su uticale na razvoj računarstva u celini, pa i na klaud računarstvo su: mainframe računarstvo (eng. *mainframe computing*), desktop računarstvo (eng. *desktop computing*), klijent/server računarstvo (eng. *client/server computing*) i mrežno računarstvo (eng. *grid computing*).

Glavne transformacije u oblasti računarstva biće prikazane na primeru baza podataka.

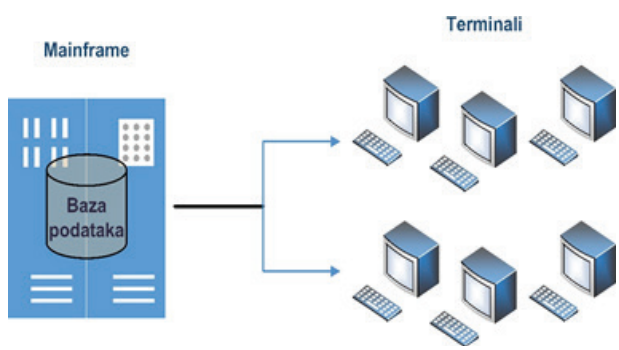
Slika 4-2: *Glavne transformacije u računarstvu*



Mainframe računarstvo, koje je obeležilo 70-te godine prošlog veka, karakterisali su veliki i skupi *mainframe* računari. U ovo vreme, računarski resursi bili su skupi i samo su velike korporacije imale finansijske mogućnosti za nabavku mainframe računara.

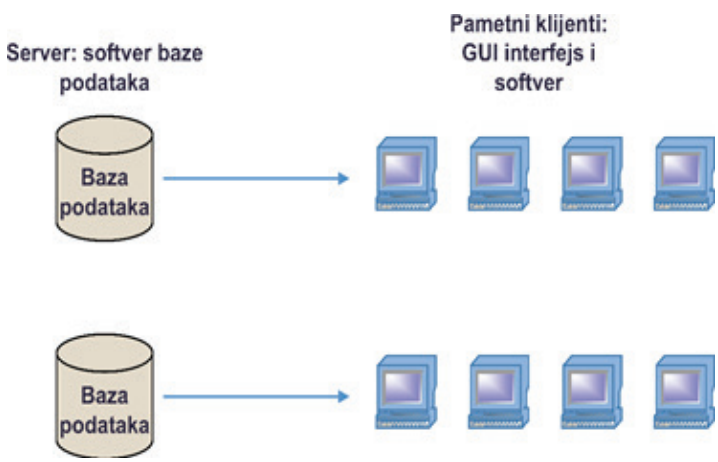
Mainframe računarima su pristupali klijenti, koji su se nazivali terminali, i na njima izvršavali komande. Terminali nisu imali sopstvene resurse za obradu podataka i zato su se sve operacije obrade izvršavale na mainframe računarima, dok su se na izlaznim jedinicama terminala prikazivali samo rezultati obrade. Tako na primer, u slučaju baza podataka, softver za upravljanje bazama podataka (eng. *Database Management System – DBMS*) nalazio se na *mainframe* računaru, gde su se i izvršavale sve obrade i transformacije podataka (Slika 4-3). O *mainframe* računarstvu više je dato u nastavku ove glave.

Slika 4-3: *Pristup bazi podataka koja se nalazi na mainframe računaru*



Kada je 80-tih godina prošlog veka *IBM* predstavio prvi personalni računar (eng. *Personal Computer – PC*), koji je bio višestruko jeftiniji od mainframe računara, i na taj način računare učinio dostupnim i stanovništvu, obrada podataka je „premeštena" sa mainframe računara na klijentske računare. Zbog činjenice da su *PC* računari imali sopstveni softver, kao i resurse za obradu podataka, ovi računari su se nazivali pametni klijenti, ili radne stanice (eng. *workstations*). U eri *desktop* računarstva sistem za upravljanje bazama podataka je bio implementiran na posebnom računaru koji se nazivao server, dok je svaki klijent imao softver sa grafičkim (eng. *Graphical User Interface – GUI*), ili konzolnim interfejsom (eng. *Command Line Interface - CLI*), za pristup sistemu za upravljanje bazama podataka, za razliku od ere *mainframe* računarstva, gde se sve nalazilo na *mainframe* računaru. Vizuelna reprezentacija koncepta desktop računarstva data je na Slici 4-4.

Slika 4-4: *Koncept desktop računarstva*



U modelu *desktop* računarstva, veliki problem je predstavljala nadogradnja softvera (eng. *software upgrades*), gde je u slučaju nadogradnje trebao da se nadogradi softver na serveru (serverima) i na svim klijentima. Za rešavanje ovog problema, 90-tih godina prošlog veka, usvojena je strategija korišćenja Interneta i brzih servera, kako bi se zadovoljile potrebe savremenih korporativnih subjekata za softverom i obradom podataka. Na ovaj način, počela je era klijent/server računarstva, koja je „raskrčila" put razvoja mrežnog računarstva.

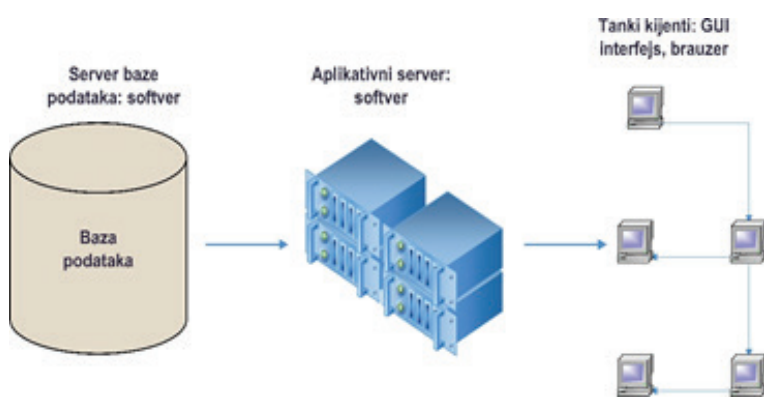
U eri klijent/server računarstva u implementaciji baza podataka, kreirana su tri sloja:

- softver za upravljanje bazama podataka, kao i sama baza podataka nalazili su se na serveru baze podataka (eng. *database server layer*). Ovaj server je obavljao proces obrade i vađenja podataka,
- aplikacije za poslovne operacije su implementirane na drugom sloju, sloju aplikativnog servera (eng. *application server layer*). Ovaj sloj je obavljao zadatke kreiranja dokumenata, razvoj, interakciju i manipulaciju podacima i
- treći, klijentski sloj (eng. *client layer*), mogao je da ima svoje aplikacije, ali se ključnim poslovnim aplikacijama pristupalo neposredno preko klijentskog brauzera (eng. *Internet Browser - IB*).

Grafički prikaz slojevite klijent/server arhitekture dat je na Slici 4-5.

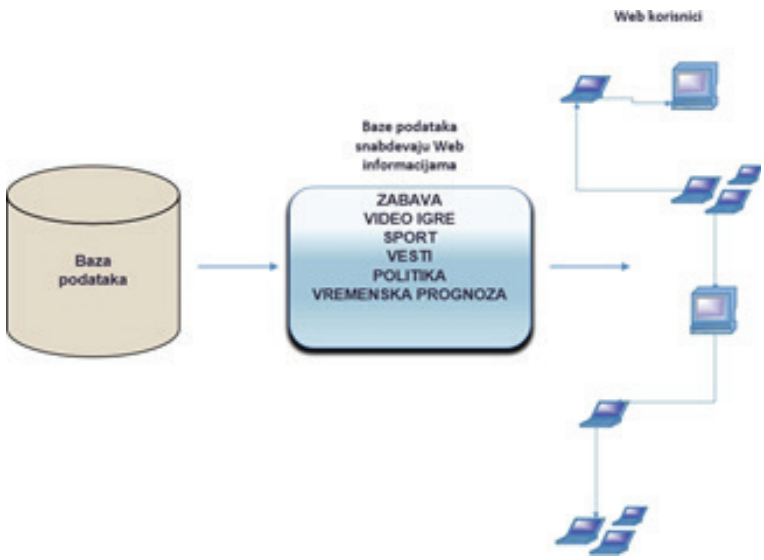
U modelu mrežnog računarstva koji se razvija od 2000-te godine, pa do danas, svi računarski resursi jedne organizacije koji su fizički distribuirani na različitim fizičkim lokacijama, mogu da se koriste kao jedan jedinstveni skup resursa. Mrežno računarstvo omogućava izgradnju softverske infrastructure koja može da se izvršava na velikom broju umreženih servera. Kada korisnik pošalje zahtev za nekom informacijom, ili bilo kojim drugim računarskim resursom sa svoje radne stanice, zahtev se obrađuje od strane nekog servera u mreži što efikasnije moguće.

Slika 4-5: *Koncept klijent/server računarstva*



U modelu mrežnog računarstva, računarski resursi se tretiraju kao usluge, analogno konceptu elektrodistribucione mreže. Nije važno gde se nalazi generator koji proizvodi struju, niti korisnike to zanima. Važno je samo da korisnici imaju struju kada im zatreba.

Slika 4-6: *Koncept mrežnog računarstva*



Dakle, primenom koncepta mrežnog računarstva, kada korisnik zatraži neki resurs, cilj je da se taj resurs obezbedi u što kraćem vremenskom roku. Nije važno sa kog servera se obezbeđuje zahtevani resurs, sve dok se poštuju određeni standardni kvaliteta i dok je konekcija pouzdana. Ovaj koncept prikazan je na Slici 4-6.

Sve navedene promene imale su veliki uticaj na nastanak i razvoj klaud računarstva. Razlika između mrežnog i klaud računarstva je bleda i nejasna i zbog toga se u stručnim i naučnim krugovima vodi polemika o sličnostima i razlikama ova dva pojma.

4.2.2. Evolucija klaud računarstva

Kada je kompanija *Amazon* lansirala svoj komercijalni *Web* servis za iznajmljivanje računara pod nazivom *Elastic Compute Cloud* 2006. godine, termin klaud računarstvo počeo je da se upotrebljava u širokim krugovima, kako opšte, tako i stručne javnosti. Međutim, ovaj termin je prvi put upotrebljen još ranije, tačnije 1996. godine, u jednom internom dokumentu koji je objavila kompanija *Compaq*.

S druge strane, dve računarske mreže koje se smatraju prethodnicama Interneta, i to prvo *ARPANET* (eng. *Advanced Research Project Agency Network*) 1977. godine, pa zatim i *CSNET* (eng. *The Computer Science Network*) 1981. godine, koristile su simbol oblaka u vizuelnoj reprezentaciji računarskih mreža. Kako bi se naglasilo da za razumevanje mrežnog dijagrama nije potrebno razumevanje kako su dva čvora na mreži povezana, simbol oblaka se koristio za označavanje načina na koji su čvorovi povezani. Drugim rečima, oblak označava da su dva čvora na mreži povezani i da mogu da komuniciraju, a način na koji su povezani nije relevantan za razumevanje funkcionisanja mreže.

Reč klaud je ubrzo nakon popularizovanja termina klaud računarstvo od strane kompanije Amazon.com ušla u široku upotrebu kao metafora za Internet. Takođe, od 1993. godine, reč klaud se koristila za označavanje platformi distribuiranog računarstva. Distribuirano računarstvo je deo računarstva koji se bavi proučavanjem distribuiranih sistema. Distribuirani sistem je sistem koji se sastoji iz više delova (komponenti), koje se nalaze na različitim računarima povezanim preko računarske mreže. Računari u distribuiranom sistemu komuniciraju i kooridiniraju svoje aktivnosti najčešće primenom mehanizma prosleđivanja poruka (eng. *message passing mechanism*).

Na osnovu analize raspoložive literature vidi se da postoje dva mišljenja o tome ko je osnivač koncepta klaud računarstva. Prva struja tvrdi da je osnivač koncepta klaud računarstva *J.C.R Licklider*, koji je aktivno učestvovao u razvoju *ARPANET* mreže 1969. Godine. *J.C.R Licklider* je promovisao ideju o konceptu tzv. „intergalaktičke računarske mreže", prema kojoj je računarska mreža definisana kao veliki skup računara i mrežne opreme koji je „sposoban da razmišlja". Njegova vizija se u osnovi odnosila na to da svako na svetu može da bude povezan i da može da pristupa svojim podacima, aplikacijama i servisima sa bilo kog mesta (lokacije) u bilo koje vreme.

Mnogi eksperti iz ove oblasti širom sveta smatraju da ovaj koncept odgovara onome što se danas promoviše kao klaud računarstvo.

Prema mišljenju druge struje eksperata, prvu ideju o konceptu klaud računarstva imao je naučnik i profesor *John McCarthy*, koji je svoja istraživanja sprovodio na univerzitetima *Stanford, MIT, Darthmounth* i *Princepton*, gde je i predavao. Još 1961. godine u doba mainframe računara, *John McCarthy* je promovisao ideju da računarske usluge mogu da se isporučuju u formi servisa i smatrao je da računarske usluge mogu da postanu osnova za jednu sasvim novu i značajnu industriju. Za ideju uslužnog računarstva (eng. *utility computing*), *John McCarthy* je dobio Tjuringovu nagradu.

U osnovi, ideja uslužnog računarstva je bila relativno jednostavna. Na isti način kako se naplaćuje korišćenje električne energije i telefonskih usluga, može da se naplaćuje i korišćenje računara i računarskih resursa, gde bi klijenti plaćali onoliko koliko su potrošili. Kompanije *IBM* i *GE* (*General Electrics*) prepoznale su potencijal ove ideje u ubrzo su počeli sa njenim sprovođenjem tako što su iznajmljivale svoje *mainframe* računare drugim kompanijama korišćenjem koncepta deljenja vremena (eng. *time-sharing*). Mainframe računari su bili izuzetno skupi, kako za nabavku, tako i za održavanje, i kompanijama se više isplatilo da takve računare iznajmljuju, nego da ih kupuju.

Još pre nastanka koncepta deljenja vremena, korisnici su se povezivali na *mainframe* računare preko tzv. terminala. Kao što je već rečeno, *mainframe* računari su bili jako skupi i velike kompanije i univerziteti mogli su iz finansijskih razloga da imaju samo jedan, eventualno nekoliko, *mainframe* računara u svom vlasništvu. Terminali koji su omogućavali korisnicima da se povežu na *mainframe* računare imali su samo ulazni uređaj (tastaturu) i izlazni uređaj (monitor), i koristili su procesor i ostale resurse udaljenog *mainframe* računara za izvršavanje obrade podataka. Zbog toga se ovakvi terminali ponekada u literaturi nazivaju glupi terminali (eng. *dumb terminals*). Ovakav način deljenja procesorskog vremena i ostalih resursa udaljenog računara može da se smatra kao idejni prethodnik klaud računarstva.

Kompanije *IBM, MIT* i *GE* su 60-tih i 70-tih godina prošlog veka razvile koncept deljenja vremena, što je bio još jedan način deljenja snage procesora *mainframe* računara između više korisnika. Uslov za primenu koncepta deljenja vremena je multiprogramiranje. Multiprogramiranje predstavlja oblik paralelne obrade, odnosno mogućnost, gde se u isto vreme učitava više programa u memoriju, uz određeno ograničenje, da se u svakom vremenskom trenutku izvršava samo jedan program.

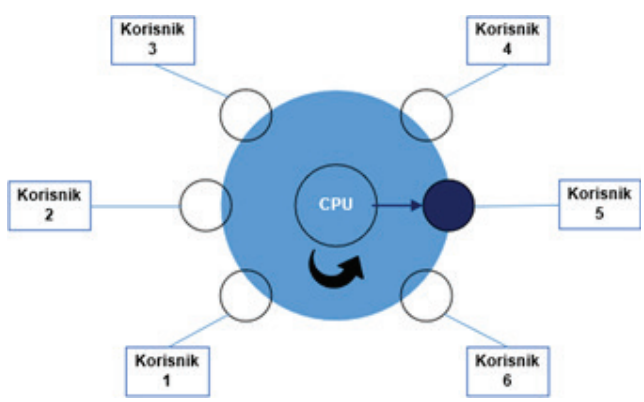
Dakle, zbog cene računarskih resursa, bilo je potrebno da se jedan *mainframe* računar podeli na više korisnika. Jasno je da prostorna podela računara nije bila moguća, već je jedino bilo izvodljivo da se računar vremenski podeli na više korisnika.

Tako na primer, ako je bilo 500 korisnika *mainframe* računara, svaki korisnik bi ukrug dobijao 2ms procesorskog vremena. Efekat je isti kao da svaki korisnik ima svoj računar koji je 500 puta slabiji od *mainframe* računara, tako da svaki korisnik ima iluziju da poseduje svoj računar. Koncept deljenja vremena prikazan je na Slici 4-7.

Kako se računarska industrija razvijala, tako su se razvijali i sigurnosni mehanizmi i mehanizmi za merenje količine utrošenih resursa, i zbog toga je ideja iznajmljivanja računara postala još popularnija među korporativnim korisnicima. Međutim, sa razvojem *PC* računara i jeftinijih servera ova ideja je počela da „odumire" 80-tih godina prošlog veka. *PC* računari su računari manjih dimenzija, mnogo niže cene, ali i skromnijih tehničkih mogućnosti i performansi nego *mainframe* računari. Zbog toga što na zahteve za računarskim resursima velikih kompanija *PC* računari nisu mogli da odgovore, koncept uslužnog računarstva i iznajmljivanja računarske snage *mainframe* računara je opstao još neko vreme.

Dakle, važno je da se uoči da ideje klaud računarstva i iznajmljivanja računarskih resursa nisu nove. Klaud je novi termin za stare ideje i principe koje su implementirane još u doba *mainframe* računara, 60-tih i 70-tih godina prošlog veka.

Slika 4-7: *Koncept deljenja vremena*



Sa ubrzanim rastom i razvojem Interneta promenjeni su i načini na koje se računari koriste i usvojeni su nove tehnike za prenos podataka. Ideja uslužnog računarstva je ponovo zaživela 2000-tih godina. Velike kompanije kao što su *Amazon* i *Google* investirale su ogromne svote novca u implementiranje farme servera, čiju su snagu počele da iznajmljuju i na taj način su dale značajan doprinos razvoju klaud računarstva.

Jednu od prvih razvojnih prekretnica u klaud računarstvu postavila je kompanija *Salesforce.com* kada je 1999. godine implementirala servis za isporuku korporativnih aplikacija krajnjim korisnicima putem Interneta i običnog *Web* brauzera. Ubrzo zatim su i mnoge druge kompanije počele sa implementacijom sličnih servisa.

Sledeću prekretnicu u razvoju klauda postavio je *Amazon* 2002. godine, kada je ova kompanija objavila svoj novi proizvod *Amazon Web* servis (eng. *Amazon Web Services – AWS*). Kompanija *Amazon* je u okviru svog *Web* baziranog servisa ponudila veliki broj usluga koje se baziraju na klaud tehnologijama, kao što su usluge obrade i skladištenja podataka. Nakon toga, 2006. godine, *Amazon* objavljuje svoj novi komercijalni *Web* servis *Elastic Compute Cloud (EC2)*, putem koga klijenti, kao što su manje organizacije i pojedinci mogu da iznajmljuju virtuelne računare za potrebe instalacije i izvršavanja sopstvenih aplikacija.

Kompanija *Google* je 2008. godine lansirana proizvod *Google App Engine beta*. U to vreme su, sa razvojem *Web* 2.0 tehnologije, i kompanije poput *Google*-a počele sa razvojem korporativnih aplikacija koje se izvršavaju i isporučuju korisnicima pomoću običnog *Web* brauzera.

Početkom 2008. godine nezavisna agencija *NASA* (*National Aeronautics and Space Administration*) je lansirala *OpenNebula* platformu koja je postala prvi softver otvorenog koda za razvoj privatnih i hibridnih klaud okruženja i federacije oblaka. Takođe, sredinom 2008. godine *Gartner* je prepoznao da bi primena koncepta klaud računarstva mogla da transformiše odnos između korisnika računarskih resursa i usluga i onih koji te resurse prodaju. *Gartner* je takođe uočio da se sve više kompanija odlučuje za iznajmljivanje računarskih resursa i da odustaje od kupovine sopstvenih resursa, što bi moglo da utiče na značajan rast računarskih proizvoda u nekim, uz istovremeno smanjivanje rasta računarskih proizvoda u drugim domenima.

U februaru 2010. godine, kompanija *Microsoft* je lansirala *Microsoft Azure* platformu, dok su u julu mesecu iste godine kompanija *Rackspace Hosting* i agencija *NASA* objavili klaud platformu otvorenog koda pod nazivom *OpenStack* projekat. Cilj projekta je bio da se omogući kompanijama da isporučuju servise klaud računarstva korišćenjem standardne hardverske infrastrukture.

Kompanija *IBM* je 2011. godine objavila *IBM SmartCloud* okvir, da bi 2012. godine kompanija *Oracle* predstavila svoj *Oracle Cloud*. Meseca maja 2012. godine kompanija *Google* je premijerno prikazala *Google Compute Engine*, koji je postao dostupan klijentima širom sveta decembra 2013. godine.

4.2.3. Klaud računarstvo vs. virtuelizacija

Kada se govori o razvoju klaud računarstva, potrebno je da se pomene i tehnologija virtuelizacije (eng. *virtualization technology*), zbog toga što je virtuelizacija glavna tehnologija koja omogućava primenu koncepta klaud računarstva. Jedna od uopštenih definicija pojma virtuelizacija, koja može da se pronađe u literaturi, data je u nastavku.

> Virtuelizacija je softver koji odvaja fizičku infrastrukturu u cilju kreiranja više posebnih resursa koji obavljaju specifične zadatke. Softver za virtuelizaciju omogućava izvršavanje više različitih operativnih sistema i aplikacija u isto vreme na jednom serveru.

Virtuelizacija se razlikuje od klaud računarstva zato što je virtuelizacija softver koji manipuliše hardverom, dok se klaud računarstvo odnosi na usluge koje se isporučuju klijentima kao rezultat ove manipulacije. Virtuelizacija je osnovni gradivni element klaud računarstva jer omogućava da klaud računarstvo kreira vrednosti za klijente (krajnje korisnike).

U literaturi i stručnoj javnosti se često pojmovi virtuelizacija i klaud računarstvo mešaju zato što tehnologija virtuelizacije i klaud računarstvo „rade" zajedno kako bi se obezbedile različite vrste usluga, kao što je to slučaj na primer kod privatnih klaud okruženja. Međutim, ovi pojmovi nisu *a priori* povezani.

Klaud računarstvo u najvećem broju slučajeva uključuje virtuelizaciju. S druge strane, klaud računarstvo obezbeđuje usluge po principu samousluživanja, elastičnosti, automatskog upravljanja, skalabilnosti i plaćanja na nivou ostvarene potrošnje što nije nerazdvojivo od tehnologije virtuelizacije.

Kao što je već i navedeno, primenom koncepta deljenja vremena 70-tih godina prošlog veka, računarski resursi su se delili na više korisnika, ali je i dalje postojalo ograničenje da jedan korisnik u svakom vremenskom trenutku može da izvršava samo jedan posao. U ovom periodu, *IBM* je kreirao *CP-40 mainframe* računar, koji je smatran za prvu instancu virtuelizacije. Ovaj računar je kreirao novu instancu operativnog sistema za svakog korisnika i svakom korisniku pojedinačno je dodeljivao memoriju i druge resurse.

Dakle, zaključuje se da i koncept virtuelizacije nije nov, kao što nije nov ni koncept klaud računarstva.

Kompanija *Insignia Solutions* 70-tih godina prošlog veka razvija proizvod pod nazivom *SoftPC*, koji je omogućavao izvršavanje *DOS* (*Disk Operating System*) aplikacija na *Unix* operativnom sistemu, što je značajno smanjilo troškove korišćenja i razvoja *DOS* operativnog sistema. Nekoliko godina kasnije, pomoću ovog softvera mogle su da se izvršavaju i *Mac* i *Windows* aplikacije na *Unix*-u. Deceniju kasnije, *Apple* je „dobio" *Virtual PC*, koji je mogao da izvršava *Windows* operativni sistem u okviru *Mac* platforme.

Godine 1998. osnovana je kompanija *VMware* koja je prvo prodavala *VMware Workstation* proizvod, koji je do 2001. godine evoluirao u *ESX* server. *ESX* pripada hipervizorima prvog nivoa, koji se izvršava direktno iznad hardvera, bez potrebe za host operativnih sistemom. *VMware* je danas tržišni lider u segmentu softvera za virtuelizaciju.

Tehnologija virtuelizacije je detaljno opisana u prvom delu udžbenika.

4.2.4. Tržište klaud računarstva danas

Još kada je 2006. godine kompanija *Amazon* lansirala *Amazon Web Services (AWS)*, klijenti su mogli da iznajmljuju od *Amazona* računarsku snagu, ili skladišta podataka na sat. Danas klijenti mogu da iznajmljuju više od 70 usluga, kao što su usluge analitike i softveri za mobilne uređaje. *Amazon S3* skladište čuva veliki broj terabajta podataka krajnjih korisnika sa milionima zahteva za pristup svake sekunde. Usluge servisa *AWS* koristi više od nekoliko miliona korisnika iz 190 zemalja širom sveta. Globalne kompanije kao što *Netflix* i *AOL* nemaju svoje računske centre, već isključivo koriste usluge AWS-a. Prihod kompanije *Amazon* samo od servisa *AWS* u 2017. godini je bio oko 18 milijardi američkih dolara.

Bez obzira što su nakon *Amazona* i drugi vendori plasirali svoje klaud platforme, kao što su na primer *Microsoft Azure* i *Google Cloud, AWS* i dalje ubedljivo dominira tržištem klaud infrastrukture.

Prema zvaničnim podacima, krajem 2017. godine, *AWS* je imao 62% tržišnog učešća u segmentu javnog klauda, dok su *Microsoft Azure* i *Google* imali 20% i 12%, respektivno. Interesantno zapažanje jeste da tržišno učešće *AWS*-a nije niti palo, niti je poraslo u odnosu na 2016. godinu, dok je učešće *Microsoft*- a i *Google*-a poraslo.

Dok *AWS* dominira korporativnim klaud tržištem, tržišta krajnjih (individualnih) potrošača zanimaju pre svega usluge skladištenja podataka, poput *Dropbox, iCloud* i *Google Drive* servisa, koje nekoliko stotina miliona korisnika svakodnevno koriste za čuvanje svojih dokumenata, slika, audio i video zapisa i ostalih resursa. Povećanje broja korisnika mobilnih uređaja neposredno je uticao i na povećanje obima korišćenja navedenih usluga.

Bez obzira što većina individualnih korisnika klauda ne zna šta je klaud tehnologija, svaki od njih svakodnevno koristi bar jedan klaud servis. Kao što je korišćenje mobilnih uređaja imalo pozitivan uticaj na korišćenje klaud usluga, tako su s druge strane klaud tehnologije uticale na poboljšanje ekonomije tržišta mobilnih uređaja tako što je kreirano klaud okruženje za razvoj mobilnih aplikacija.

4.3. KONCEPT KLAUD RAČUNARSTVA

Jedna od boljih analogija sa klaud računarstvom su usluge koje svako domaćinstvo koristi, kao što su usluge isporuke električne energije i vode za piće. Krajnje korisnike uglavnom ne zanima način na koji se električna energija, ili voda za piće stvaraju i isporučuju. Krajni korisnici žele konačni proizvod (vodu ili električnu energiju) i plaćaju za taj proizvod onoliko koliko su potrošili.

Računarstvo ranije nije koristilo klaud model kreiranja i isporuke računarskih resursa. Kompanije su kupovale računarske sisteme i investirale velike svote novca u nabavku računarske opreme, kako bi potrebe za informacijama bile zadovoljene. Međutim, s obzirom da se tehnologija brzo razvija i da računari koji „danas" nose epitet najnovijih već „sutra" postaju zastareli, investiranje u računarsku opremu je predstavljalo ozbiljan kapitalni trošak čak i za velike kompanije.

Osim toga, velike kompanije su morale i da zapošljavaju specijalizovano osoblje, kao što su administratori sistema (eng. *system administrators*), inženjeri sistema (eng. *system engineers*) i inženjeri računarskih mreža (eng. *network engineers*), itd., kako bi nabavljenu opremu konfigurisali i održavali.

Pored svega navedenog, potrebno je bilo i da se obezbede specijalne prostorije sa dobrim hlađenjem i dobrim i pouzdanim električnim instalacijama gde se oprema nalazila. Sigurnost podataka i fizički pristup lokaciji gde se oprema nalazila su trebali stalno da se unapređuju, zato što sigurnosne pretnje podacima i sistemima iz dana u dan postaju sve ozbiljnije.

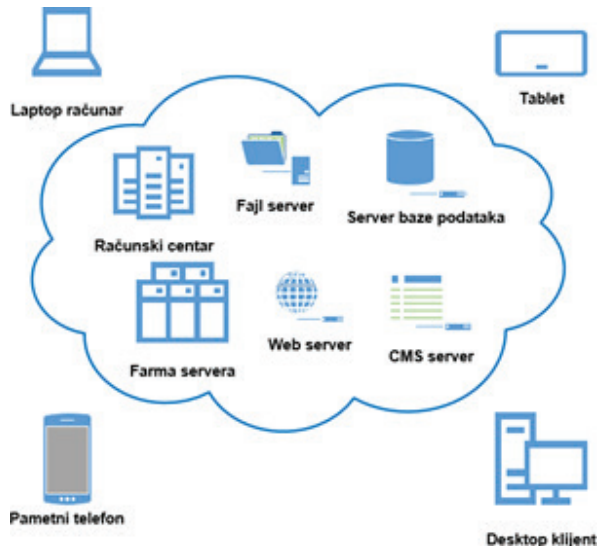
Klaud računarstvo prati model „korisnosti", gde snabdevač usluga „prodaje" računarske resurse po modelu „potreba" (eng. *as-needed model*), ili po modelu potrošnje (eng. *as-consumed model*). Na ovaj način, organizacioni ili individualni potrošači plaćaju za samo onoliko resursa koliko ih potroše, što je samo jedna od prednosti klaud računarstva.

U vreme pisanja ove knjige, tržište i prihvatanje klaud računarstva raste širom sveta. Mnoge velike kompanije koje imaju značajan deo tržišnog učešća u svojoj branši već su prihvatile model klaud računarstva i odavno su odustale od tradicionalnog načina nabavke računarske opreme. Prema procenama analitičara, tržište klaud računarstva raste stopom od 15% do 30% godišnje i do 2020. godine će dostići vrednost od preko 300 milijardi američkih dolara.

Klaud računarstvo obuhvata skup računarskih resursa poput hardverskih uređaja, sistema za upravljanje bazama podataka, sistema za skladištenje podataka i interfejsa, koji obezbeđuju isporuku računarskih usluga u formi servisa. Servisi klaud računarstva obezbeđuju isporuku aplikacija, virtuelnog prostora za skladištenje podataka ili celokupne računarske infrastrukture putem globalne računarske mreže u cilju zadovoljavanja korisničkih zahteva i/ili potreba.

Za razliku od tradicionalnih sistema gde se isporučuju gotovi proizvodi, u klaud računarstvu se isporučuju merljivi servisi putem računarskih mreža, najčešće Interneta. U klaud okruženju korisnike zanimaju usluge koje dobijaju, kao i interfejsi pomoću kojih pristupaju tim uslugama, zato što klaud okruženje skoro u potpunosti apstrahuje računarske resurse. Na primer, za pristup klaud aplikacijama krajnji korisnici koriste samo *Web* brauzer, ili mobilnu aplikaciju i orijentisani su isključivo na aplikaciju i na interfejse za pristup toj aplikaciji, dok se poslovni softver i podaci nalaze na udaljenim serverima. Pritom, korisnici ovim uslugama mogu da pristupe sa bilo koje lokacije i sa bilo kog klijentskog uređaja koji ima ugrađen kompatibilan *Web* brauzer. Pristup sa udaljene lokacije u klaud računarstvu prikazan je na Slici 4-8.

Slika 4-8: *Pristup sa udaljene lokacije u klaud računarstvu*



Sagledavajući koncept klaud računarstva, potrebno je da se naglasi da klaud servisi poštuju princip višestrukog zakupa servisa, odnosno aplikacije (eng. *multitenancy principle*). Poštujući princip višestrukog zakupa aplikacije, više korisnika može da zakupi jednu instancu neke aplikacije. Suprotno od ovog principa bio bi koncept arhitekture sa više instanci, gde se generiše posebna instanca aplikacije, odnosno servisa za svakog korisnika.

Kada se klaud računarstvo posmatra kao koncept, mogu da se izdvoje tri kategorije aktera (učesnika) u sistemu:

- krajnji potrošači (korisnici) klaud sistema koji su orijentisani isključivo na zahtevane servise i kao takvi ne moraju ništa da znaju o tehnologijama koje omogućavaju i isporučuju tražene servise,
- menadžment koji preuzima obaveze i odgovornosti za upravljanje servisima i podacima koji se skladište u klaud okruženju i
- dobavljač (provajder) klaud usluga (eng. *Cloud Service Provider – CSP*) preuzima obaveze i odgovoran je za celokupnu klaud računarsku infrastrukturu. Provajder klaud usluga ima obavezu da isporuči pouzdanu i kalitetnu uslugu uz poštovanje svih načela računarske bezbednosti.

Kada se klaud računarstvo posmatra iz perspektive hardvera, mogu da se izdvoje tri aspekta:

- korisnik klaud usluga ima iluziju da postoji raspoloživa beskonačna količina hardverskih računarskih resursa koji su dostupni na zahtev. Zbog ovoga klaud korisnik ne mora unapred da planira potrošnju hardverskih resursa,
- klaud korisnik ne mora unapred da se obaveže i da iznajmi veći broj računarskih resursa. Nasuprot ovome, korisnik u početku može da ima malu količinu resursa, koje će tokom vremenskog perioda eksploatacije inkrementalno da uvećava, gde uvećanje diktiraju potrebe, odnosno proširenje obima poslovanja i
- korisnik klaud usluga može da plaća potrošnju računarskih resursa za relativno male vremenske jedinice, kao što su na primer obrada podataka na sat i čuvanje (skladištenje) podataka na dan/nedelju dana. S druge strane, klaud korisnik može relativno lako da otkaže korišćenje ovih usluga kada više ne postoji realna potreba za njima.

4.3.1. Prednosti korišćenja koncepta klaud računarstva

Usvajanje koncepta klaud računarstva donosi brojne prednosti za organizacione korisnike i kao najznačajnije prednosti mogu da se izdvoje sledeće: brža nabavka i implementacija opreme, manji početni kapitalni troškovi i mogućnost trenutnog proširivanja računarskih resursa.

U ranije korišćenim modelima nabavke računarske opreme, kada je kompanijama bila potrebna nova računarska oprema, kompanije su se suočavale sa dugim procesom nabavke, instalacije i konfigurisanja željene opreme. U modelu klaud računarstva, u trenutku kada kompanija donese odluku za proširivanje svojih računarskih kapaciteta, sva oprema je već implementirana i konfigurisana u računskom centru klaud dobavljača i potrebno je samo nekoliko minuta da ta oprema može da počne i da se koristi.

U klasičnom modelu nabavke računarskih resursa, kompanije su nabavku nove opreme razmatrale kao kapitalne investicije, koje su morale da se uzmu u obzir u dugoročnom finansijskom planiranju. Kako bi se efikasno upravljalo novom opremom, kompanije su morale da zapošljavaju specijalizovano *IT* osoblje. S druge strane, u modelu klaud računarstva, kompanija nema inicijativne investicije, već fiksni mesečni trošak, što omogućava kompaniji da novac koji bi potrošila za nabavku nove alocira na druge projekte i poslovne poduhvate.

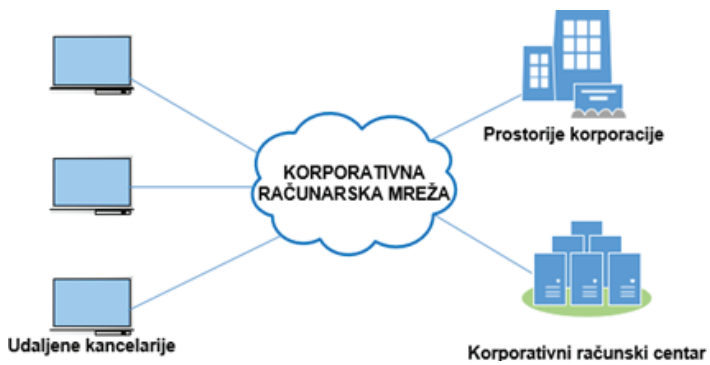
Konačno, kako zahtevi za računarskom opremom rastu, korišćenjem modela klaud računarstva, računarski resursi mogu gotovo u trenutku da se prošire, kako bi se zadovoljile rastuće potrebe. S druge strane, u slučaju klasičnog modela nabavke računarskih resursa, proširivanje kapaciteta najčešće predstavlja dug proces, što može da šteti kompaniji u vidu izgubljenih prihoda i oportunitetnih troškova.

S ovim u vezi, potrebna je da se spomene i elastičnost (eng. *elasticity*) klaud računarstva, gde je proširivanje računarskih resursa automatizovani proces. Svojstvo elastičnosti klaud računarstva može da se prikaže na jednostavnom primeru. Neka je kompanija svoj *Web* sajt za online prodaju implementirala u klaud okruženju. U vremenskom trenutku sajt je prilično nepopularan i jedna mašina (uglavnom su u pitanju virtuelne mašine) je dovoljna za opsluživanje zahteva svih posetilaca sajta. Međutim, u vremenskom trenutku, na sajtu je objavljen veliki popust za određeni skup artikala i broj posetilaca naglo počinje da raste i jedna mašina više ne može da odgovori na zahteve svih posetilaca, već je potrebno deset takvih mašina kako bi svi posetioci bili efikasno opsluženi. U ovakvoj situaciji, elastični sistem klaud okruženja bi odmah detektovao nove uslove i dodao bi devet novih mašina koje su raspoložive u klaudu, kako bi svi posetioci bili opsluženi. Zatim, nakon određenog vremena, *Web* sajt bi ponovo postao nepopularan, kada bi elastični sistem detektovao promenu uslova i devet mašina bi izbacio iz sistema.

4.3.2. Oprema u prostorijama kompanije vs. klaud računarstvo

U slučaju kada se oprema koju kompanija koristi čuva u prostorijama kompanije (eng. *in-house computing, on-premises*) potrebni su visoko kvalifikovani inženjeri koji se bave operativnim sistemima, aplikacijama, skladištima podataka i računarskim mrežama. Na ovaj način za upravljanje svom računarskom opremom koju organizacija poseduje zadužen je jedan korporativni entitet. Ovakav model prikazan je na Slici 4-9.

Slika 4-9: *Oprema u prostorijama kompanije*



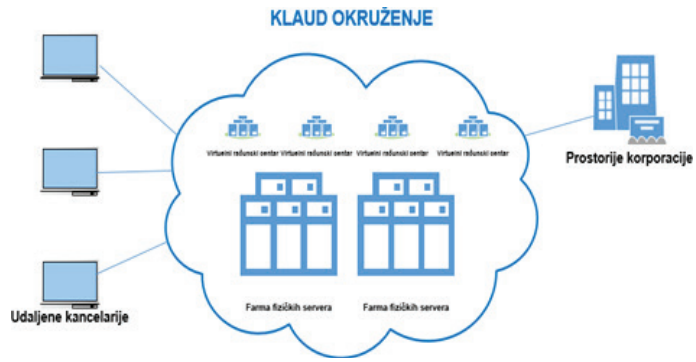
S druge strane, u slučaju da se kompanija odlučila za model klaud računarstva, sve operacije računskog centra više nisu u prostorijama kompanije, već se resursi nabavljaju spolja (eng. *outsourcing*) od dobavljača usluga, koji može da postigne ekonomiju obima tako što deli raspoložive računarske resurse sa drugim kompanijama, koje takođe koriste usluge tog dobavljača. Pogled na koncept klaud računarstva sa aspekta lokacije na kojoj se oprema nalazi dat je na Slici 4-10.

Slika 4-10: *Oprema u prostorijama klaud provajdera*



Provajder klaud usluga ekonomiju obima može da postigne primenom tehnologije virtuelizacije. Virtuelizacija hardverskih resursa omogućava potpuno iskorišćenje fizičkih servera uz simultano smanjivanje potrošnje električne energije, zahteva za hlađenjem opreme i otiska servera (eng. *server footprint*) u računarskim centrima. Veliki klaud provajderi najčešće imaju više farmi fizičkih servera u svom vlasništvu, gde se nalaze računski centri korporativnih klijenata u formi virtuelnih mašina. Na Slici 1-11 prikazani su virtuelni računski centri korporativnih klijenata koji koriste zajednički hardver u prostorijama klaud provajdera.

Slika 4-11: *Virtuelni računski centri dele zajednički fizički hardver*



Na osnovu analize trendova u klaud računarstvu, vidi se da će klaud računarstvo kao model nabavljanja računarskih resursa u vidu usluge nastaviti da raste da će sve veći broj kompanija da se odluči za ovakav vid nabavke *IT* resursa. Osim iznajmljivanja opreme računskih centara, korporativni korisnici će sve više početi da koriste i sve druge klaud usluge i servise, od usluga za skladištenje podataka, pa sve do *ERP* (*Enterprise Resource Planning*) i *CRM* (*Customer Relationship Management*) sistema.

Slika 4-12: *Konzumiranje raznih klaud usluga od strane korporativnih korisnika*



PITANJA ZA RAZMATRANJE



1. Definisati pojam klaud računarstva prema *NIST*-u. Koje su karakteristike klaud računarstva prema ovom modelu?
2. Definisati pojam klaud računarstva prema *ENISA* definiciji. Koje su karakteristike klaud računarstva prema ovom modelu?
3. Definisati pojam klaud računarstva prema kompaniji *Gartner*-u. Koje su karakteristike klaud računarstva prema ovom modelu?
4. Definisati pojam klaud računarstva prema kompaniji *IBM*. Koje su karakteristike klaud računarstva prema ovom modelu?
5. Definisati pojam klaud računarstva prema kompaniji *Microsoft*. Koje su karakteristike klaud računarstva prema ovom modelu?
6. Dati na primeru baza podataka koje su glavne transformacije u oblasti računarstva. Kako kompanija *Oracle* klasifikuje transformacije u računarstvu? Navesti podelu.
7. Objasniti ukratko evoluciju klaud računarstva.
8. Koja tehnologija je bitna u razvoju klaud računarstva? Objasniti.
9. Izdvojiti aktere u klaud računarstvu. Objasniti.
10. Izdvojiti tri aspekta klaud računarstva. Objasniti.
11. Navesti predosti korišćenja klaud računarstva. Objasniti.
12. Navesti razlike između implementirane opreme u prostorijama kompanije i opreme u prostorijama klaud provajdera. Objasniti i dati primere.

5. TIPOVI KLAUD RAČUNARSTVA PREMA MODELU USLUGA

U literaturi je poznato više načina za podelu (klasifikaciju) klaud računarstva. Jedna od najpoznatijih podela je prema modelu usluga koje klaud računarstvo obezbeđuje. U nastavku će biti navedena ova podela.

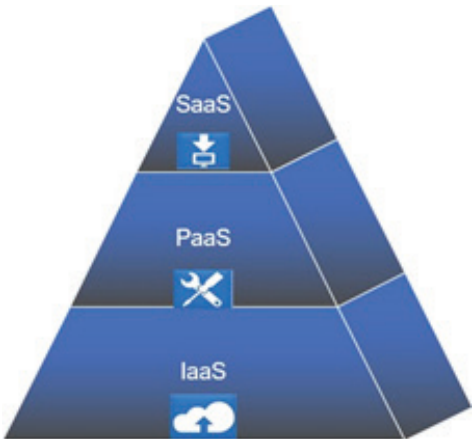
Prema zvaničnoj definiciji kompanije *Gartner*, koja se uklapa u model usluga, klaud računarstvo je stil računarstva u kome se skalabilni i elastični računarski resursi dostavljaju klijentima (potrošačima) u vidu usluge (eng. *as a service*) korišćenjem Internet tehnologija. Ovim uslugama klijenti mogu da pristupe preko različitih uređaja, kao što su *Web* brauzeri, tanki klijenti, ili mobilni uređaji.

Prema viđenju organizacije *CompTIA* (eng. *Computing Technology Industry Association*), mogu da se izdvoje tri osnovna modela usluga klaud računarstva:

- softver kao usluga (eng. *Software as a Service – SaaS*);
- infrastruktura kao usluga (eng. *Infrastructure as a Service – IaaS*) i
- platforma kao usluga (eng. *Platform as a Service – PaaS*).

Mnogi provajderi klaud usluga koriste opisnije nazive modela klaud usluga, u cilju postizanja većeg marketinškog efekta, kao što su: komunikacije kao usluga (eng. *Communication as a Service - CaS*), bilo šta kao usluga (eng. *Anything as a Service – XaaS*), *desktop* kao usluga (eng. *Desktop as a Service – DaaS*), poslovni proces kao usluga (eng. *Business Process as a Service – BPaaS*) i skladište podataka kao usluga (eng. *Storage as a Service*). Naravno, sve navedene usluge se uklapaju u tri osnovna modela usluga klaud računarstva. Kao napomena navodi se da se stalno kreiraju novi modeli usluga klaud računarstva.

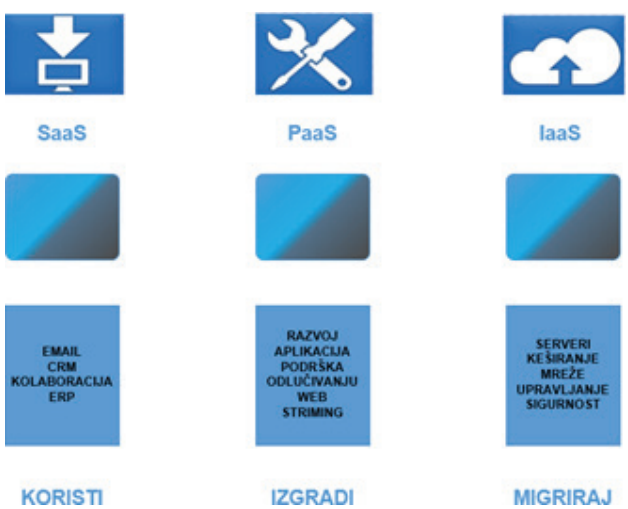
Slika 5-1: *Tipovi klaud računarstva prema modelu usluga*



Zbog konzistentnosti sa svetskom literaturom, u nastavku će se na pojedinim mestima koristiti engleski akronimi za referisanje tri osnovna modela usluga klaud računarstva, i to za softver kao uslugu koristiće se akronim *SaaS*, za platformu kao uslugu biće upotrebljavan akronim *PaaS*, dok će infrastruktura kao usluga biti referisana kao IaaS.

Vizuelna reprezentacija tipova klaud računarstva prema modelu usluga data je na slikama 5-1 i 5-2.

Slika 5-2: *Modeli usluge i funkcionalnosti*



5.1. SOFTVER KAO USLUGA

Tip klaud računarstva prema modelu softvera kao usluge (*SaaS*) obuhvata softver koji se implementira u formi *hostovanog* servisa kojem može da se pristupi preko računarske mreže, najčešće preko Interneta.

Kod ovog modela implementacije, korisnici na zahtev dobijaju licence za aplikacije koje su im potrebne i koriste aplikacije do onog vremenskog trenutka kada su im potrebne.

Prema zvaničnoj definiciji *NIST*-a, *SaaS* je mogućnost da krajnji korisnik koristi aplikaciju dobavljača (provajdera) koja se izvršava na klaud infrastrukturi. Krajnji korisnici pristupaju aplikaciji pomoću klijentskog uređaja, tankog klijentskog interfejsa, ili aplikativnog programskog interfejsa.

Krajnji korisnik ne može da kontroliše, niti da upravlja nižim slojevima klaud infrastrukture, tako da klijent nema pristup mreži i mrežnim podešavanjima, serverima i serverskim podešavanjima, operativnim sistemima i opcijama za konfigurisanje operativnih sistema, skladištima i opcijama za podešavanje skladišta podataka. Klijent čak u većini slučajeva ne može ni da kontroliše pojedine mogućnosti aplikacije. U modelu *SaaS*, krajnjem korisniku je samo omogućen pristup i korišćenje aplikacije, koja je u stopostotnom vlasništvu dobavljača klaud usluga, koji kontroliše aplikaciju i potpuno preuzima sve obaveze koje se odnose na upravljanje i održavanje te aplikacije.

Poslovne aplikacije se mogu izdvojiti kao dobar primer *SaaS*. Ove aplikacije obuhvataju sledeće: upravljanje odnosima s kupcima (eng. *Customer Relationship Management - CRM*), planiranje resursa korporacije (eng. *Enterprise Resource Planning – ERM*), upravljanje ljudskim resursima (eng. *Human Resource Management – HRM*), upravljanje zaradama (eng. *stock management*), alati za produktivnost (eng. *office productivity tools*), alati za deljenje informacija i znanja (eng. *information/knowledge sharing tools*), itd. Takođe, *hostovane* aplikacije, kao što su *webmail* i *online* Kalendar, koje su dostupne preko interfejsa *Web* brauzera, su dobro poznati primeri *SaaS* koje većina prosečnih korisnika računara svakodnevno koristi. Neke od navedenih aplikacija prikazane su na Slici 5-3.

Slika 5-3: *SaaS aplikacije*



Neke od prednosti koje omogućava primena modela *SaaS* su sledeće:

- optimalna upotreba raspoloživih računarskih resursa,
- izbegavanje troškova koji bi nastali kupovinom licenci za korišćenje aplikacija,
- eliminacija vremena, utrošene energije i troškova koji bi nastali na osnovu implementacije i održavanja svih ovih servisa,
- brz razvoj, jer se softver obezbeđuje na zahtev,
- klijent ne treba da brine o upravljanju aplikacijama, jer to spada u domen odgovornosti i obaveza provajdera usluga i
- aplikacije su obično stabilne i pouzdane, jer podršku realizuje čitava infrastuktura i ekspertski tim provajdera usluge.

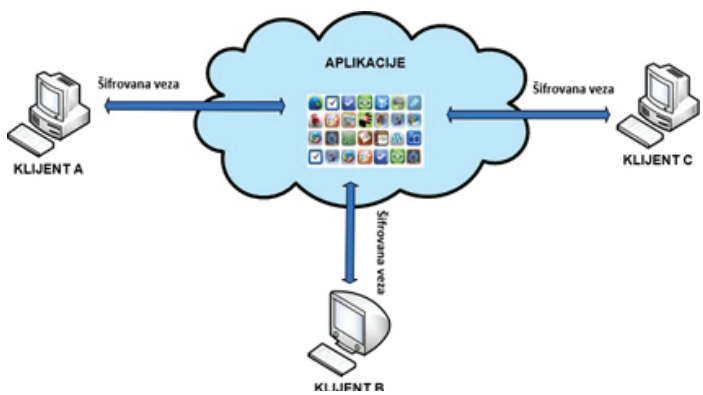
S druge strane, primena SaaS modela nosi sa sobom i određene nedostatke, a kao neki od njih navode se sledeći:

- klijent nema kontrolu nad sistemom koji obrađuje i skladišti njegove podatke,
- ne postoji mogućnost uvida, niti kontrolisanja koji sve klijenti koriste istu instancu softvera,
- slaba kontrola nad razvojem, nadogradnjom i testiranjem aplikacija,
- integracija sa drugim softverima na drugim sistemima je otežana, ili uopšte nije podržana i
- isporučilac usluga ima punu kontrolu nad podacima klijenta, osim u slučajevima kada se koriste tehnike kriptografije.

Jedna od značajnijih razlika koje klaud aplikacije diferencira od tradicionalnih aplikacija jeste to što klaud aplikacije karakteriše visoka elastičnost koja je omogućena distribucijom radnog opterećenja na veći broj virtuelnih mašina. U cilju zaštite podataka koji putuju preko mreže, komunikacioni kanal koji spaja krajnjeg korisnika i dobavljača usluga se obično šifruje *SSL/TLS* vezom (eng. *Secure Socket Layers/Transport Layer Security*). Međutim, i pored šifrovanja podataka, napadač može da otkrije indirektne informacije, kao što su prisustvo i izvor saobraćaja, veličina poruka, itd. Osim ovoga, model softvera kao usluge je podložan tipu napada nepoznati posrednik (eng. *man-in-the-middle*) na kriptografske protokole koji su implementirani u samom *Web* brauzeru.

Pristup klijenata klaud aplikacijama preko šifrovanih veza prikazan je na Slici 5-5.

Slika 5-4: *Pristup klijenata klaud aplikacijama*



Kao neki od poznatijih predstavnika *SaaS* modela navode se *Google Apps, Dropbox, Salesforce, Cisco WebEx, Concur, GoToMeeting, Microsoft Office 365*, itd. Paleta klaud aplikacija kompanije *Salesforce* data je na Slici 5-5.

Slika 5-5: *Klaud aplikacije kompanije Salesforce*



Izvor: (slika preuzeta sa: https://www.salesforce.com/)

5.2. INFRASTRUKTURA KAO USLUGA

Kao i za sve druge pojmove u računarstvu, i za pojam infrastrukture kao usluge (*IaaS*) postoje brojne definicije u literaturi koje su dali vendori ovog tipa klaud usluga.

> Prema *NIST*-ovoj zvaničnoj definiciji, *IaaS* je sposobnost koju isporučilac usluge omogućava klijentu, a koja se odnosi na dodavanje računarskih resursa za obradu i skladištenje podataka, mrežnih i drugih resursa, gde klijent može da razvija i pokreće proizvoljan softver, zajedno sa operativnim sistemima i aplikacijama.

Dalje, kako *NIST* navodi, u modelu *IaaS*, klijent ne kontroliše, niti upravlja klaud infrastrukturom, ali zato ima potpunu kontrolu nad operativnim sistemima, skladištima podataka, implementiranim aplikacijama, i u pojedinim slučajevima i određenim mrežnim komponentama, kao što su na primer zaštitni zidovi (eng. *firewalls*) koji štite *host* računare.

Model *IaaS* nudi klijentu veću fleksibilnost u odnosu na sve druge tipove klaud računarstva prema modelu usluge. Klijent ima potpunu kontrolu nad osnovnim resursima, kao što su skladišta, mreže, softveri i aplikacije. Na ovaj način klijent iznajmljuje osnovnu računarsku platformu, koju koristi za razvoj sopstvenih rešenja.

Kompanije koje koriste model *IaaS* zamenjuju opremu u korporativnom računskom centru ekvivalentnom opremom u klaud okruženju, ali i dalje mogu da implementiraju softversku infrastrukturu, na isti način kao i kada bi se sva oprema nalazila u prostorijama lokalnog korporativnog računskog centra.

Paleta osnovnih usluga *IaaS* modela data je na Slici 5-6.

Slika 5-6: *Osnovne IaaS usluge*



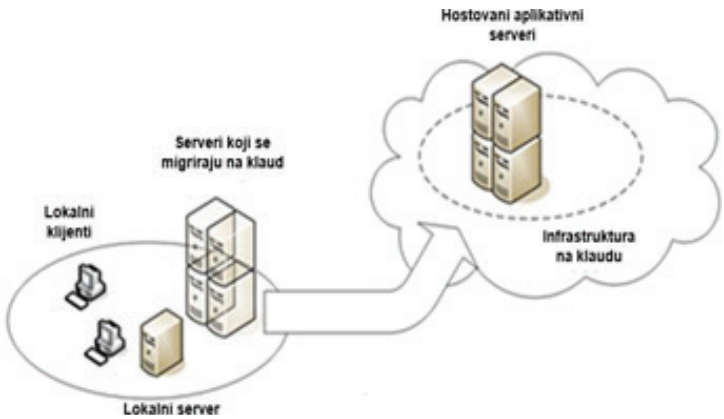
Provajderi *IaaS* usluga pored standardnih hardverskih resursa nude i *firewall* opremu, komponente za balansiranje opterećenja mreže i obrade podataka (eng. *load balancers*), mrežne uređaje (eng. *network appliances*), kao i ostale računarske komponente. Svi navedeni resursi iznajmljuju se po principu „na zahtev" od provajdera usluga kada su potrebni klijentu.

U *IaaS* modelu mogu da se izdvoje tri sloja:

- sloj za upravljanje klaudom (eng. *cloud management layer*), koji predstavlja javnu pristupnu tačku, kao najviši sloj centralne kontrole nad sistemom,
- sloj za upravljanje klasterima (eng. *cluster management layer*), koji se odnosi na srednji sloj koji je odgovoran za upravljanje velikim klasterima servera na kojima se izvršavaju virtuelne mašine koje klijenti koriste (klasteri mogu da imaju stotine i hiljade servera, što zavisi od raspoložive hardverske infrastrukture) i
- sloj za upravljanje računarima (eng. *computer management layer*), koji predstavlja najniži sloj i kao takav je odgovoran za upravljanje računarima domaćinima na kojima su implementirane virtuelne mašine koje klijenti koriste.

Model *IaaS* se najčešće koristi kada kompanija hoće da migrira svoj lokalni računski centar na klaud okruženje. Ovaj scenario prikazan je na Slici 5-7.

Slika 5-7: *Migracija lokalnog računskog centra na klaud*



Kao što važi za skoro sve drugo u računarstvu, primena modela *IaaS* uključuje i neke prednosti, ali i nedostatke. Neke od najznačajnijih prednosti modela *IaaS* obuhvataju sledeće:

- interoprabilnost i portabilnost,
- skalabilnost i fleksibilnost,
- potpuna kontrola nad virtuelnim mašinama,
- laka administracija virtuelnih mašina,
- relativno jednostavna integracija sa ostalim računarskim resursima korporacije,
- efikasno i fleksibilno iznajmljivanje računarskih resursa,
- plaćanje na osnovu nivoa utrošenih i iskorišćenih resursa,
- smanjivanje inicijalne investicije u hardverske resurse,
- smanjivanje troškova i
- fokus na rast i razvoj poslovanja.

Kao neki od rizika (nedostataka) primene *IaaS* modela navode se sledeći:

- najskuplji model iznajmljivanja resursa, gde provajderi usluga naplaćuju za svaki ciklus procesorskog vremena, utrošenog megabajta *RAM* memorije i megabajta iskorišćenog skladišnog prostora,
- klijent je odgovoran za čuvanje i pravljenje rezervnih kopija podataka (eng. *data backup*) podataka,
- za razliku od *SaaS* i *PaaS* modela, klijent je odgovoran za sve aspekte upravljanja virtuelnim mašinama,

- klijent ne zna gde je fizički (geografski) lociran server na kome se nalazi njegova virtuelna mašina,
- zavisnost od mreže klaud provajdera i
- postavlja se pitanje robusnosti izolacije virtuelnih mašina (u kojoj meri je virtuelna mašina određenog klijenta izolovana od mašina drugih klijenata istog klaud provajdera).

Kao poznatiji predstavnici provajdera *IaaS* usluga ističu se: *DigitalOcean, Linode, Rackspace, Elastic Compute Cloud (EC2) Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)*, itd.

Usluge *Microsoft Azure* platforme prikazane su na Slici 5-8.

Slika 5-8: *Microsoft Azure platforma*



5.3. PLATFORMA KAO USLUGA

Model platforme kao usluge (*PaaS*) klijentima obezbeđuje infrastrukturu i operativne sisteme u formi usluga, gde klijent može sam da instalira i da konfiguriše potrebne aplikacije na klaud platformi (Slika 5-9).

Prema ovom modelu, provajder usluga preuzima punu odgovornost za hardver i operativne sisteme (platformu). Klijent može brzo da instalira svoje aplikacije bez potrebe da kupuje servere i drugu računarsku opremu, što dalje omogućava klijentu brz razvoj aplikacija.

Prema *NIST*-ovoj zvaničnoj definiciji, *PaaS* je mogućnost klijenta, koju obezbeđuje provajder usluga, da u klaud okruženju razvija svoje, ili kupljene aplikacije, korišćenjem programskih jezika i alata koje obezbeđuje provajder klaud usluga.

Slika 5-9: *PaaS model*



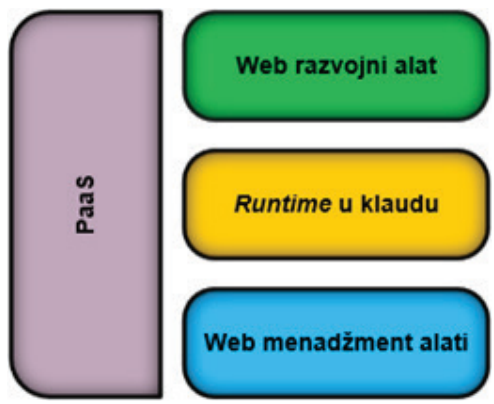
Dalje, prema navodima *NIST*-a, u modelu *PaaS*, klijent ne kontroliše, niti upravlja klaud infrastrukturom (mržom, serverima, operativnim sistema i skladištima), ali zato klijent ima potpunu kontrolu nad implementiranim aplikacijama, i u nekim slučajevima i nad podešavanjima okruženja u kome su aplikacije implementirane.

Dakle, primenom *PaaS* modela, klijent može da u okviru klaud infrastrukture dobavljača razvija, implementira i podešava svoje, ili aplikacije trećih strana (eng. *3rd party applications*) korišćenjem programskih jezika, alata i biblioteka koje su podržane od strane isporučioca usluga. Ovo ne znači da podrška ne može da se obezbedi iz trećih izvora ako isporučilac ne podržava sve alate koji su klijentu potrebni za razvoj aplikacija.

PaaS obuhvata servise i alate koji ne treba da se instaliraju od strane korisnika.

Neki od alata *PaaS* modela su prikazani na Slici 5-10.

Slika 5-10: *PaaS alati*



Jedan od glavnih ciljeva *PaaS* modela je obezbeđivanje platforme, tj. skupa alata (eng. *solution stack*) koji se obično sastoji iz sledećeg:

- operativnog sistema,
- programskog okruženja,
- baze podataka i
- aplikativnog ili *Web* servera.

Programeri aplikacija široko koriste ovakav model za razvoj, izvršavanje i testiranje razvijenog softvera, ali ne upravljaju operativnim sistemom, mrežnim podešavanjima i prostorom za skladištenje, uz najčešće minimalnu kontrolu i mogućnosti konfigurisanja razvojnog okruženja. Za razliku od klasičnih računarskih platformi (hardver + operativni sistem), *PaaS* predstavlja jeftiniji I pristupačniji model za razvoj skalabilnih aplikacija.

Neke od prednosti *PaaS* modela odnose se na sledeće:

- za razliku od *IaaS*, model *PaaS* je efikasniji u smislu troškova, jer klijent iznajmljuje softversku platformu (ne hardverski resurs),
- za razliku od modela *SaaS*, klijent ima mogućnost da implementira sopstveni softver koji se izvršava na klaud platformi, što implicira potpunu kontrolu nad softverom,
- potpuna kontrola nad korisnicima koji pristupaju softveru i obrađuju podatke (uz određeni rizik, jer klijent ne zna ništa o virtuelnim mašinama na kojima se nalazi softverska platforma),
- poboljšana i olakšana integracija sa drugim sistemima i
- minimalni napor koji se ulaže u upravljanje virtuelnim mašinama zato što se tim pitanjima bavi isporučilac usluge.

Kao značajniji nedostaci modela platforme kao usluge navode se sledeći:

- klijent nema kontrolu nad virtuelnim mašinama, niti nad obradom podataka, što predstavlja veliki rizik zato što klijent ne zna šta se dešava sa njegovim podacima,
- u nekim slučajevima nema kontrole nad platformom (zavisi od klaud isporučioca usluge),
- platforma se najverovatnije deli sa drugim klijentima,
- klijent dosta vremena i energije ulaže u ažuriranje i nadogranju aplikacija,
- portabilnost i integracija sa platformama drugih provajdera usluge,
- pitanja izolacije nasuprot efikasnosti i
- manje troškovno efikasan od modela *SaaS* i manje kontrole nad virtuelnim mašinama u odnosu na model *IaaS*.

Kao jedan od najčešćih primera primene modela IaaS bi bio korišćenje IIS (Internet Information Services) Web servera koji je implementiran na klaud platformi provajdera.

Neki od primera *PaaS* su: *AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift*.

*Google App Engin*e prikazan je na Slici 5-11.

Slika 5-11: *Google App Engine*



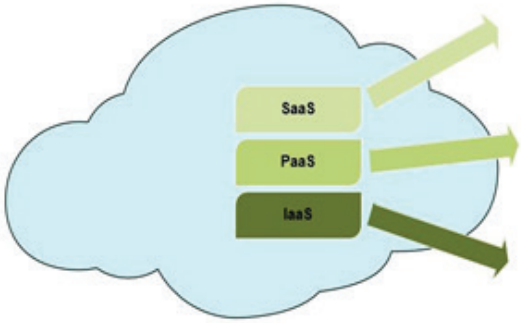
5.4. RAZLIKE IZMEĐU SAAS, IAAS I PAAS

Na osnovu opisa *SaaS, IaaS* i *PaaS* modela zaključuje se da postoje značajne razlike između ova tri modela. Razlike između ovih tipova klaud računarstva mogu da se definišu i sagledaju sa nekoliko aspekata.

Jedan od najboljih načina za razlikovanje ova tri modela je davanje odgovora na sledeća pitanja: ko koristi usluge, koje usluge su raspoložive i zašto se usluge koriste.

Razlike između *SaaS, IaaS* i *PaaS* modela na osnovu navedenih pitanja prikazane su na Slici 5-12 .

Slika 5-12: *Razlikovanje modela klaud računarstva na osnovu korisnika usluga, raspoloživosti usluga i razloga za korišćenje usluga*



Ko koristi usluge	Koje usluge su raspoložive	Zašto se usluge koriste
Poslovni korisnici	Email, Office automatizacija, CRM, testiranje web sajta, Wiki, Virtuelni desktop	Da bi se izvršili poslovni zadaci
Developeri	Testiranje usluga i aplikacija za razvoj i integraciju	Kreiranje i razvoj aplikacija i usluga za krajnje korisnike
Sistem menadžeri	Virtuelne mašine, operativni sistemi, mreže, skladište, CPU, memorija, usluge backup	Kreiranje platformi za razvoj i testiranje aplikacija i usluga

Još jedna bitna razlika između *SaaS, IaaS* i *PaaS* modela odnosi se na delokrug nadležnosti i odgovornosti provajdera usluga i klijenta. U prvom slučaju, kada kompanija ne koristi usluge klaud računarstva i kada se celokupna infrastruktura kompanije (serveri, mreža, aplikacije, podaci, itd.) nalazi u prostorijama kompanije, onda je sama kompanija odgovorna za sve aspekte sistema, od računarske mreže, pa sve do aplikacije i podataka.

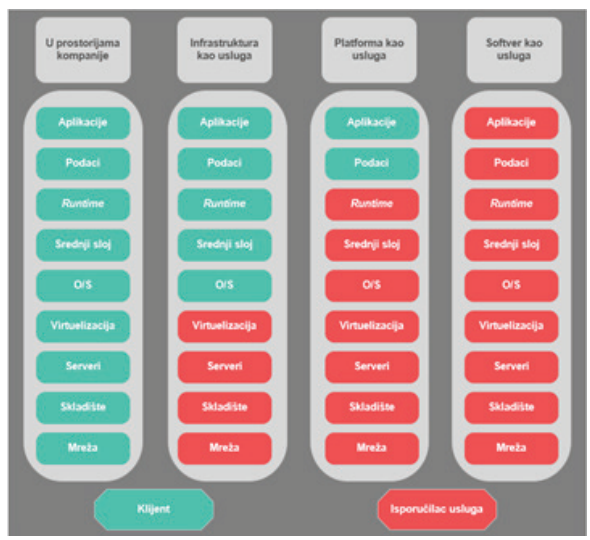
U slučaju kada kompanija koristi *IaaS* model, onda kompanija klijent upravlja i snosi odgovornost za operativne sisteme, srednji sloj, izvršno okruženje, podatke i aplikacije, dok mrežnom opremom, skladištem podataka, serverima i softverima za virtuelizaciju upravlja, pa samim tim i snosi punu odgovornost klaud provajder.

Model *PaaS* još više sužava nadležnosti i odgovornosti kompanije klijenta, tako da je u ovom slučaju kompanija klijent odgovorna samo za aplikacije i podatke, dok svim ostalim aspektima klaud sistema, od mrežne opreme do izvršnog okruženja upravlja klaud provajder.

Konačno, primenom modela *SaaS*, kompanija klijent skoro uopšte nema obaveze, niti odgovornosti u upravljanju klaud infrastrukturom, jer celokupnom infrastrukturom upravlja klaud provajder, sve od mrežne opreme do podataka i aplikacija. U ovom slučaju kompanija klijent je običan korisnik aplikacija koje su *host*-ovane u prostorijama i na opremi klaud provajdera.

Vizuelna reprezentacija razlika između *SaaS, IaaS* i *PaaS* modela na osnovu distribucije nadležnosti i odgovornosti klijenta i klaud provajdera data je na Slici 5-13 .

Slika 5-13: *Razlika između SaaS, IaaS i PaaS modela na osnovu podele nadležnosti i odgovornosti klijenta i klaud provajdera*



Kada se posmatra iz još jednog aspekta, razlike između *SaaS, IaaS* i *PaaS* modela mogu da se uoče ako se uzmu u obzir načini isporuke usluga i situacije (scenariji) u kojim treba koristiti ove modele, što je prikazano u tabelama 5-1 i 5-2.

Tabela 5-1: *Razlike između SaaS, IaaS i PaaS modela na osnovu načina isporuke*

Model	Način isporuke
SaaS	*SaaS* eliminiše potrebu za preuzimanjem i instaliranjem aplikacija na svakom računaru, što najčešće predstavlja „noćnu moru" za računarske eksperte. Aplikacije se isporučuju klijentima putem Interneta, ili druge računarske mreže, preko jednostavnog interfejsa, bilo da je to *Web* bazirani, ili programski interfejs.
IaaS	Model *IaaS* isporučuje kompletnu infrastrukturu klaud računarstva (servere, računarske mreže, skladišta podataka, itd.) organizacijama primenom tehnologije virtuelizacije. Serveri u kladu okruženju obično komuniciraju sa klijentom preko komandne table (eng. *dashboard*), ili aplikativnog programskog interfejsa (eng. *application programming interface – API*), gde *IaaS* klijenti imaju potpuno kontrolu nad celom infrastrukturom. Klijenti i dalje imaju pristup svojim privatnim računskim centrima koji se nalaze u prostorijama kompanije, ali uglavnom koriste servise virtuelnog računarskog sistema u klaud okruženju. Za razliku od modela *Saas* i *PaaS, IaaS* klijenti su odgovorni za upravljačke aktivnosti svim aspektima sistema, kao što aplikacije, izvršno okruženje, operativni sistemi, srednji slojevi (eng. *middleware*) i podaci. Međutim, u ovom slučaju, provajder usluge i dalje upravlja fizičkim serverima, skladištima, mrežama i virtuelizacijom.
PaaS	Način isporuke u modelu *PaaS* je sličan načinu koji se koristi u modelu *SaaS*, s tom razlikom da se ne isporučuje softver, već se isporučuje platforma za razvoj softvera. Ovakva platforma tim za razvoj aplikacija oslobađa od briga koje su vezane za operativni sistem, ažuriranja i infrastrukturu.

Tabela 5-2: *Razlike između SaaS, IaaS i PaaS modela na osnovu scenarija korišćenja*

Model	Scenariji korišćenja
SaaS	Kada je u pitanju mala, ili *startup* kompanija koja treba brzo da lansira svoje prozivode na tržište i koja brzo treba da implementira softvere, poput alata za elektronsku trgovinu, alata za kolaboraciju za potrebe projekata na kojima radi više učesnika i slično. Ovaj model se takođe koristi i kada su u pitanju aplikacije koje se retko upotrebljavaju, kao što je na primer softver za obračun poreza, kao i za aplikacije kojima je potrebno pristupati i putem *Web*-a i putem mobilnog uređaja, itd..
IaaS	Ako je u pitanju mala, ili *startup* kompanija, *IaaS* može značajno da ubrza proces implementacije hardverske infrastrukture i može značajno da smanji inicijalne kapitalne investicije. Ako je u pitanju velika kompanija koja želi da ima potpunu kontrolu nad svojim aplikacijama i infrastrukturom, i koja želi da kupi samo onaj hardver i softver koji je zaista neophodan, u situacijama kada se potrebe za hardverom i softverom stalno menjaju.
PaaS	Kada više *developer*-a radi na istom projektu gde se uključuju u druge kompanije, kada je potrebno određeni projekat realizovati brzo, kada postoji potreba za razvojem personalizovanih aplikacija i kada je potrebno značajno smanjiti troškove razvoja aplikacija zbog konkurencije, itd.

Konačno, kao poslednji aspekt razlika između *SaaS, IaaS* i *PaaS* modela navode se razlike na osnovu kategorije korisnika (kranji korisnici, programeri i operatori/ *IT* osoblje). Kada se posmatra iz ovog ugla, kranji korisnici koriste *SaaS* model, dok *PaaS* i *IaaS* koriste programeri i *IT* osoblje, respektivno.

Razlika između *SaaS, IaaS* i *PaaS* modela na osnovu ovog aspekta data je na Slici 5-14.

Slika 5-14: *Razlika između SaaS, IaaS i PaaS modela na osnovu tipa korisnika*



5.5. KOMUNIKACIJE KAO USLUGA

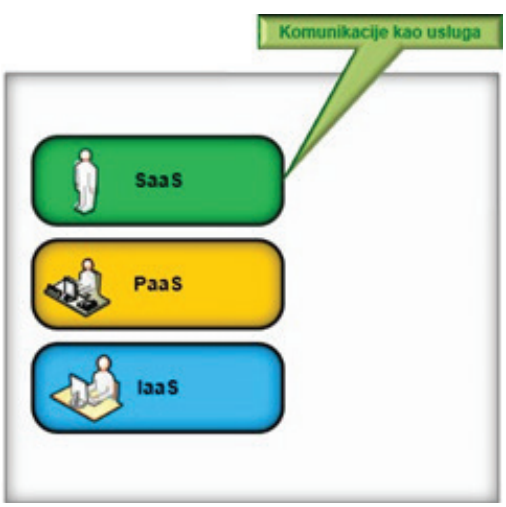
Komunikacije kao usluga (eng. *Communications as a Service – CaaS*) obuhvata softvere za video-konferencije, poruke u realnom vremenu (eng. *instant messaging*), elektronsku poštu (eng. *email*), alate za kolaboraciju i druge komunikacione usluge koje su implementirane u klaud okruženju.

Ovakve komunikacione korporativne usluge podržavaju i pristup sa lokacije kompanije i pristup preko mobilnih uređaja.

Primenom modela komunikacija kao usluga omogućeno je čak i manjim i srednjim kompanijama da koriste napredne komunikacione tehnologije po razumnom trošku, koji može da se izmeri. Nove funkcije mogu da se implementiraju i zaposleni u kompaniji ne moraju da upravljaju komunikacionim servisima, jer je za to zadužen provajder usluga.

U literaturi se često model komunikacija kao usluga sreće pod nazivom unificirane komunikacije kao usluga (eng. *Unified Communications as a Service – UCaaS*). Model komunikacije kao usluga spada u domen *SaaS* modela, kao što je prikazano na Slici 5-15.

Slika 5-15: *Komunikacije kao usluga*



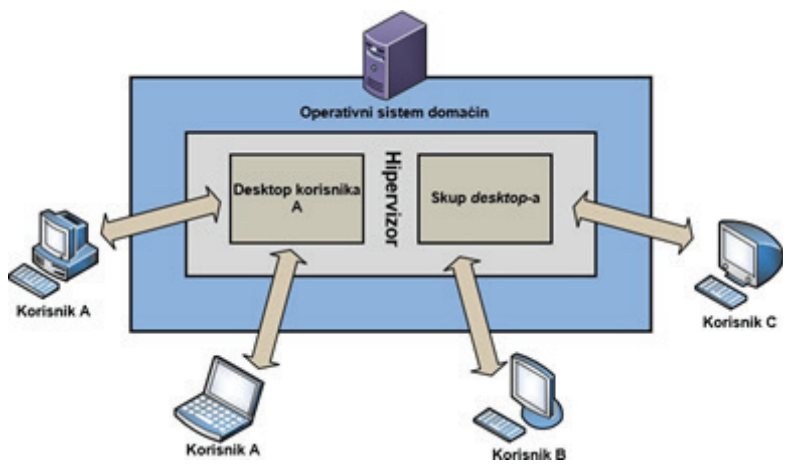
5.6. DESKTOP KAO USLUGA

Desktop kao usluga (eng. *Desktop as a Service – DaaS*) odnosi se na virtuelni *desktop* (radnu površinu) koji se nalazi u klaud okruženju i kome se pristupa pomoću klijentskih računara, tablet i laptop računara, telefona, ili čak preko hardverskog ili softverskog rešenja u vidu tankog klijenta.

Ovakav *desktop* se ponekada naziva i infrastruktura virtuelnog *desktop*-a (eng. *virtual desktop infrastructure – VDI*).

Ovakva infrastruktura koristi tehnologiju virtuelizacije, gde jedan korisnik može svom *desktop*-u da pristupa sa više klijentskih uređaja u isto vreme, kao što je prikazano na Slici 5-16.

Slika 5-16: *Primena tehnologije virtuelizacije u VDI modelu*



Prema ovom modelu se sve *desktop* aplikacije nalaze u klaudu i mogu da obuhvataju bilo koji tip aplikacije, kao što su aplikacije za tabelarne proračune (eng. *spreadsheet applications*), aplikacije za obradu teksta (eng. *word processing applications*), ili bilo koji drugi tip dobro poznatih aplikacija.

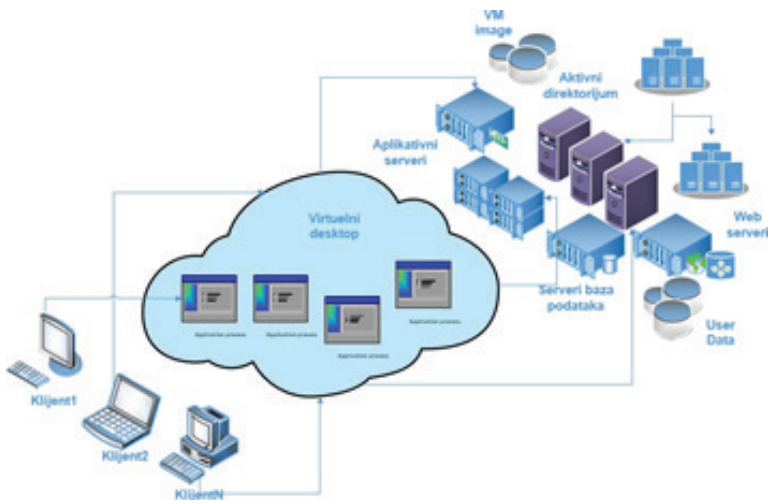
U modelu *desktop* kao usluga, isporučilac usluge održava i upravlja svim podešavanjima, licenciranjem, ažuriranjima i nadogranjama aplikacija. Primenom modela *desktop* kao usluga može na primer da se virtuelizuje *Windows* operativni sistem, kome se pristupa preko Interneta i tankog klijenta kao što je *Web* brauzer.

Grafički prikaz infrastrukture virtuelnog *desktopa* dat je na Slici 5-17.

5.7. POSLOVNI PROCES KAO USLUGA

Poslovni proces kao usluga (eng. *Business Process as a Service – BPaaS*) je specijalizovana oblast koju koriste mnoge kompanije za obavljanje svakodnevnih operativnih operacija, kao što su upravljanje inventarom i isporukom, upravljanje lancima snabdevanja, finansijama, itd.

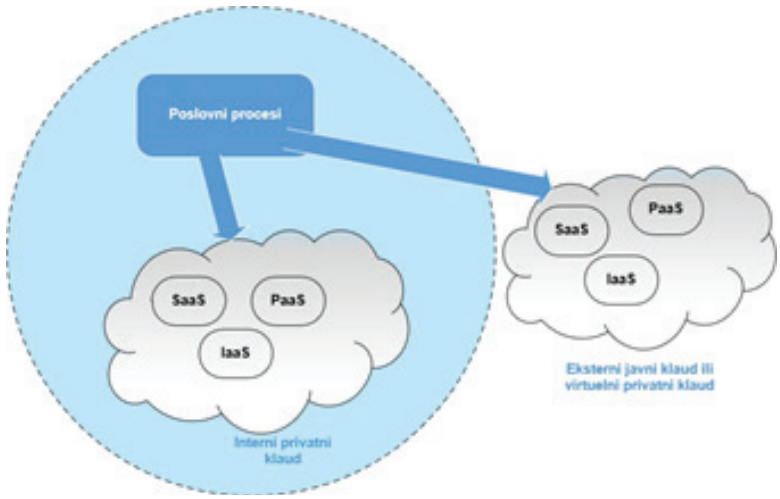
Slika 5-17: *Infrastruktura virtuelnog desktopa*



Model poslovnog procesa kao usluga je u osnovi sličan *SaaS* modelu, s tim što je u ovom slučaju akcenat stavljen samo na određene aplikacije koje se koriste za svakodnevne poslovne operacije kompanija. Na ovaj način i male i srednje kompanije ograničenih finansijskih potencijala mogu na relativno jeftin način da koriste sofisticirani softver zahvaljujući ekonomiji obima koja postoji na strani provajdera usluge.

Jedan od primera poslovnog procesa kao usluge je *IBM Blueworks Live*. Prikaz poslovnog procesa kao usluge dat je na Slici 5-18.

Slika 5-18: *Poslovni proces kao usluga*

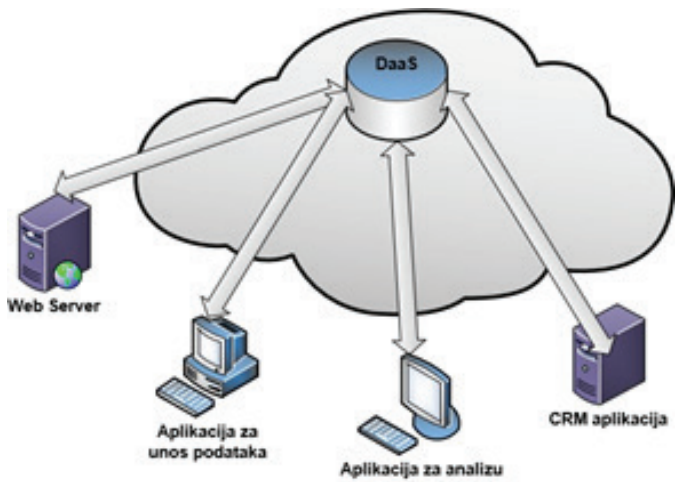


5.8. SKLADIŠTE PODATAKA KAO USLUGA

Skladište kao usluga (eng. *Storage as a Service - SaaS*) predstavlja model klauda, gde velike kompanije iznajmljuju svoju infrastrukturu za skladištenje podataka manjim kompanijama, ili individualnim korisnicima.

U literaturi se često za uslugu klaud skladišta koristi i naziv *DaaS* (*Data as a Service*), kada korporativne aplikacije i servisi poput *Web* servera, *CRM* sistema, aplikacija za unos i analizu podataka, itd., koriste klaud skladište za čuvanje korporativnih podataka (Slika 5-19).

Slika 5-19: *Korišćenje klaud skladišta za potrebe korporativnih aplikacija i servisa*



Mnogi dobavljači skladišta kao usluge promovišu ovaj vid klaud računarstva kao jednostavan i udoban način za upravljanje rezervnim kopijama podataka. Glavna prednost koju skladište kao usluga donosi korporativnim klijentima je ušteda u troškovima, što se odnosi kako na uštede u angažovanom osoblju, tako i na uštede na hardverskim i drugim resursima za skladištenje podataka.

Tako na primer, umesto da koristi lokalne hard diskove za pravljenje rezervnih kopija, administrator mreže može jednostavno da označi za koje softverske resurse je potrebno praviti rezervne kopije i koliko često je proces pravljenja rezervnih kopija potrebno izvršavati, i kreiranje rezervnih kopija će preko Interneta automatski da se izvršava u definisanim vremenskim intervalima.

Ako korporativni klijent koristi skladište kao uslugu mora da potpiše sporazum o nivou korišćenja usluga (eng. *service level agreement – SLA*) sa dobavljačem klaud usluga, gde će biti navedena cena koja se plaća po gigabajtu uskladištenih podataka i troškovi transfera podataka iz prostorija kompanije do računskog centra dobavljača usluge preko vlasničke širokopojasne računarske mreže (eng. *proprietary WAN*) dobavljača usluge. U slučaju da se korporativni podaci izgube, ili pokvare, dovoljno je da administrator mreže kontaktira dobavljača usluge koji će pravovremeno da dostavi najsvežiju rezervnu kopiju.

Skladište kao usluga je generalno dobra alternativa za manje ili srednje korporacije kojima nedostaju finansijska sredstva i/ili osoblje za implementiranje i održavanje sopstvenog skladišta podataka. Ovaj tip klaud računarstva se takođe promoviše i kao dobro rešenje u kontekstu da kompanija izbegave rizike oporavka od katastrofa (eng. *disaster recovery*) i kao rešenje koje obezbeđuje dugoročno čuvanje podataka, čime se poboljšava i kontinuitet poslovanja (eng. *business continuity*) i raspoloživost servisa (eng. *availability*).

Klaud skladište podataka prikazano je na Slici 5-20.

Slika 5-20: *Klaud skladište podataka*



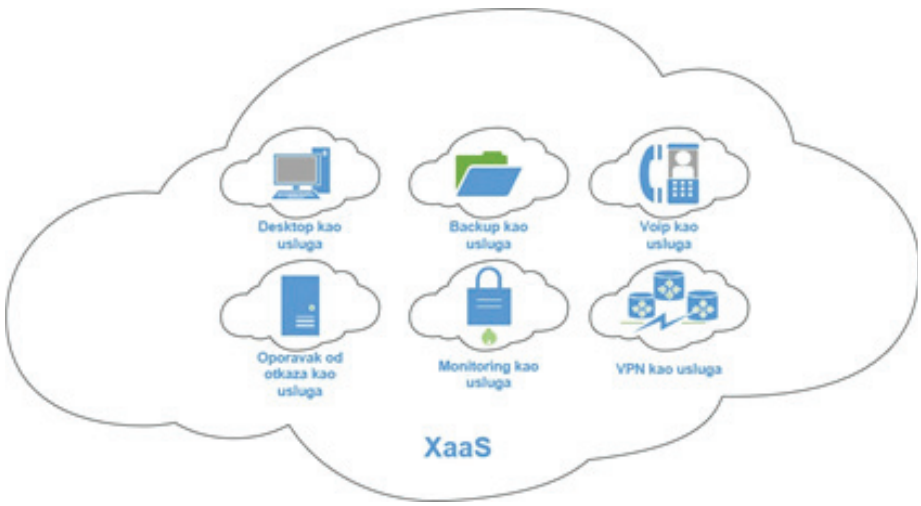
5.9. BILO ŠTA KAO USLUGA

U poslednje vreme se koristi i relativno širok model klaud računarstva pod nazivom bilo šta kao usluga (eng. *Anything as a Service – XaaS*). Ovaj model najbolje može da se opiše kao celokupna ponuda računarskih resursa u formi jedinstvenog paketa.

Reč *X* u akronimu *XaaS* ovog modela odnosi se na bilo koju uslugu. Bilo šta kao usluga može da bude bilo koja kombinacija usluga opisanih u ovoj glavi udžbenika. Međutim, bilo šta kao usluga najčešće obuhvata usluge iz modela *SaaS, PaaS* i *IaaS*. Ova tri koncepta se često u literaturi generički nazivaju *SPI* model, kao kombinacija *SaaS, PaaS* i *IaaS* modela.

Koncept *XaaS* prikazan je na Slici 5-21.

Slika 5-21: *Bilo šta kao usluga*



PITANJA ZA RAZMATRANJE



1. Objasniti tip klaud računarstva prema modelu softvera kao usluge. Navesti prednosti i nedostatke.
2. Objasniti tip klaud računarstva prema modelu infrastrukture kao usluge. Dati primer. Navesti prednosti i nedostatke.
3. Objasniti tip klaud računarstva prema modelu platforme kao usluge. Koji su glavni ciljevi ovog modela? Navesti prednosti i nedostatke.
4. Koje su razlike između *SaaS, IaaS* i *PaaS* modela? Objasniti razlike kroz primer na osnovu podele nadležnosti i odgovornosti klijenta i klaud provajdera.
5. Objasniti razlike između *SaaS, IaaS* i *PaaS* na osnovu načina isporuke.
6. Objasniti razlike između *SaaS, IaaS* i *PaaS* na osnovu scenarija korišćenja.
7. Objasniti razlike između *SaaS, IaaS* i *PaaS* na osnovu tipa korisnika.
8. Objasniti šta je *CaaS*? U kom modelu usluga ova tehnologija spada?
9. Objasniti šta je *DaaS*? Dati primer.
10. Objasnti šta je *BPaaS*? Dati primer.
11. Objasniti model skladište kao usluga. Dati primer.
12. Objasniti model bilo šta kao usluga. Dati primer.

6. TIPOVI KLAUD RAČUNARSTVA PREMA MODELU ISPORUKE

Kako bi klaud računarstvo i potrebe koje ono zadovoljava bolje razumeli individualni i organizacioni korisnici, industrija je kreirala referentne klaud arhitekture i modele isporuke.

Modeli isporuke vrše segmentaciju ponude klaud računarstva u određene kategorije i pružaju bolji uvid u sam domet klaud računarstva.

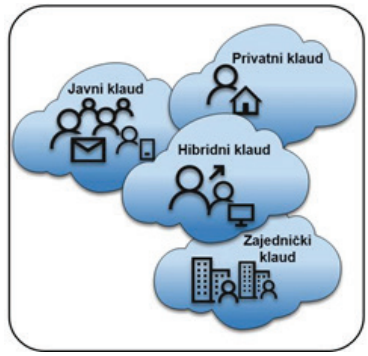
U literaturi se najčešće pominju četiri osnovna modela isporuke klaud računarstva:

- javni klaud (eng. *public cloud*),
- privatni klaud (eng. *private cloud*),
- zajednički klaud (eng. *community cloud*) i
- hibridni klaud (eng. *hybrid cloud*).

Takođe, u zavisnosti od tipova klaud okruženja mogu da se diferenciraju različiti modeli implementacije, gde svaki model implementacije definiše drugačiji kompromis koji se odnosi na kontrolu računarskih resursa od strane korisnika, skaliranje, dostupnosti i cene za korišćenje resursa.

Tipovi klaud računarstva prema modelu isporuke prikazani su na Slici 6-1. U nastavku su sva četiri modela detaljnije prikazana.

Slika 6-1: *Tipovi klaud računarstva prema modelu isporuke*



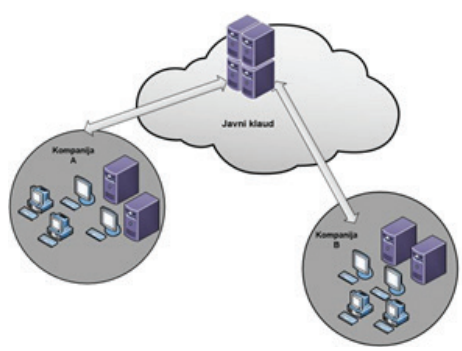
6.1. JAVNI KLAUD

Javni klaud je najšire korišćeni model isporuke. U nastavku teksta je navedena jedna od najčešće upotrebljivanih definicija javnog klauda.

> Model javnog klauda obuhvata infrastrukturu koja je dostupna za korišćenje i koju koristi šira javnost. Po prirodi, javni klaud je servisno orijentisan i usvaja model plaćanja na osnovu nivoa potrošenih javnih, eksternih računarskih resursa.

Primer kada istu infrastrukturu javnog klauda koriste dve kompanije dat je na Slici 6-2.

Slika 6-2: *Primer korišćenja servisa javnog klauda*



Infrastruktura javnog klauda je u vlasništvu poslovne, vladine, ili akademske organizacije koje su potpuno odgovorne za njegovu sigurnost i bezbednost. Celokupna javna klaud infrastruktura nalazi se u prostorijama dobavljača klaud usluga.

Ako se posmatra iz perspektive organizacionih korisnika klaud usluga, javni klaud predstavlja primenu koncepta spoljnog snabdevanja (eng. *outsourcing concept*), gde organizacije, bilo velike ili male, samo zakupljuju klaud usluge od dobavljača, a pritom se ne bave administriranjem i održavanjem infrastrukture. Iz ovoga i potiče naziv javni klaud, obzirom da takvo okruženje deli više organizacija. U modelu javnog klauda dominiraju organizacije koje su se specijalizovale kao dobavljači klaud usluga.

Kada se govori o modelu implementacije javnog klauda, može da se kaže da je infrastruktura javnog klauda dostupna i raspoloživa javnosti, gde svi korisnici dele iste servise i infrastrukturu. U pojedinim slučajevima usluge javnog klauda, do određenog nivoa korišćenja, mogu da se ponude besplatno, dok se u većini slučajeva korišćenje naplaćuje na osnovu nivoa ostvarene potrošnje.

Infrastruktura javnog klauda se sastoji iz obimnog hardvera koji je distribuiran na nekoliko različitih lokacija širom jedne zemlje, ili širom sveta. Veličinom ove infrastrukture postiže se ekonomija obima, koja dalje omogućava najviši nivo skalabilnosti korporativnim korisnicima, kako se poslovanje korporacije povećava ili se sužava, zatim najviši nivo fleksibilnosti, kako bi potrebe klijenata bile zadovoljene u realnom vremenu uz minimalno vreme zastoja, kao i najviši nivo pouzdanosti u slučaju otkaza hardvera.

Javni klaud je visoko efikasan u smislu troškova, zato što kompanije plaćaju samo za one računarske resurse koje su potrošile, dok s druge strane kompanije dobijaju pristup sofisticiranoj i naprednoj računarskoj opremi koju održava i kontroliše dobavljač klaud usluga.

Kao neki od glavnih nedostataka javnog klauda su pitanja bezbednosti (sigurnosti) i usklađivanja sa lokalnim zakonima i regulativama. Infrastruktura nije u kontroli kompanija, pa samim tim i sigurnost korporativnih podataka i sistema zavisi od dobavljača usluge. Na primer, s obzirom da infrastrukturu javnog klauda mogu da dele kompanije iz različitih zemalja sveta, implementacija infrastrukture ne može da bude usklađena sa svim lokalnim regulativama. Takođe, korporativni klijenti nemaju uvid u fizičku lokaciju infrastrukture koju koriste, što predstavlja ograničavajući faktor upotrebe (posebno za državne i vladine kompanije i agencije).

Model javnog klauda najviše se koristi za hostovanje razvojnih platformi za obradu velike količine podataka i za kompanije koje nemaju visoke standarde po pitanju sigurnosti i bezbednosti podataka i sistema.

Neke od glavnih prednosti javnog klauda obuhvataju sledeće:

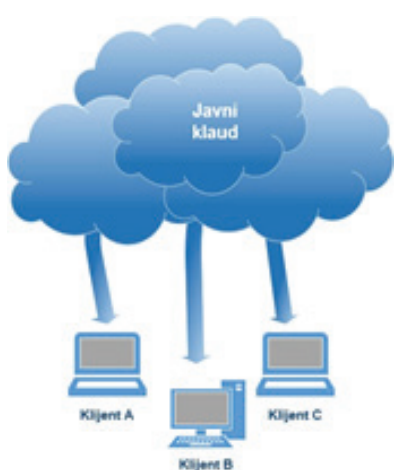
- veća skalabilnost računarskih resursa,
- troškovna efikasnost,
- pouzdanost,
- usluge poput *SaaS, PaaS* i *SaaS* su široko dostupne na platformama javnog klauda i može da im se pristupi sa bilo koje lokacije u svetu i
- nezavisnost od lokacije.

Među značajnijim nedostacima javnog klauda izdvajaju se sledeći:

- slab, ili potpuni izostanak kontrole privatnosti i sigurnosti,
- nemogućnost korišćenja za potrebe osetljivih aplikacija i podataka (na primer baza podataka banke sa računima klijenata),
- nedostatak u vidu potpune fleksibilnosti, zato što platforma zavisi od dobavljača usluge i
- nepostojanje striktnih protokola za upravljanje podacima. Prikaz pristupa infrastrukturi javnog klauda dat je na Slici 6-3.

Dobro poznati primeri javnog klauda su *SalesForce, Microsoft Azure Cloud Computing Platform and Services, Amazon EC2, Google AppEngine* i *IBM Blue Cloud*.

Slika 6-3: *Pristup infrastrukturi javnog klauda*

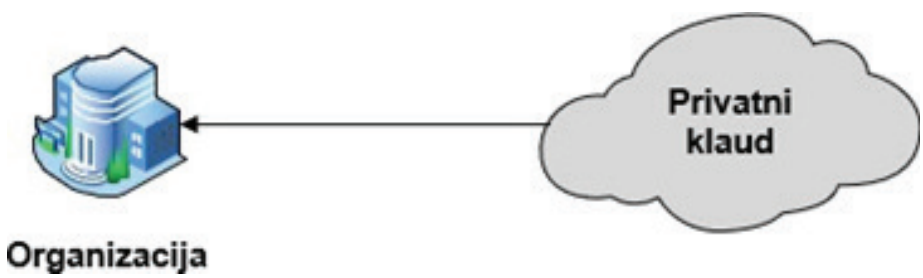


6.2. PRIVATNI KLAUD

Privatni klaud je model gde su organizacije (korporacije, ustanove) vlasnici sopstvenog privatnog „oblaka" (ne nužno i same infrastrukture), kog u najvećem broju slučajeva same konfigurišu, održavaju, administriraju i koriste.

Privatni klaud je u suštini samo novi naziv za tradicionalni računski centar korporacije, koji koristi samo korporacija, ili poslovne jedinice korporacije kojoj računski centar i pripada. Prikaz privatnog klauda dat je na Slici 6-4.

Slika 6-4: *Model privatnog klauda*



Dakle, suština privatnog klauda jeste da je cela infrastruktura obezbeđena za ekskluzivno korišćenje od strane jedne organizacije, ili njenih poslovnih entiteta. U najvećem broju slučajeva privatnim klaudom upravlja sama organizacija vlasnik, dok u retkim slučajevima privatnim klaudom može da upravlja i treća strana.

U modelu privatnog klauda organizacija obično razvija sopstvene platforme i aplikativni softver. Samo klaud infrastruktura može da bude u vlasništvu kompanije korisnika, ali i ne mora. U modelu privatnog klauda akcenat je na korišćenju (ne na vlasništvu). Ako nije vlasnik, onda organizacija koristi infrastrukturne usluge klaud provajdera.

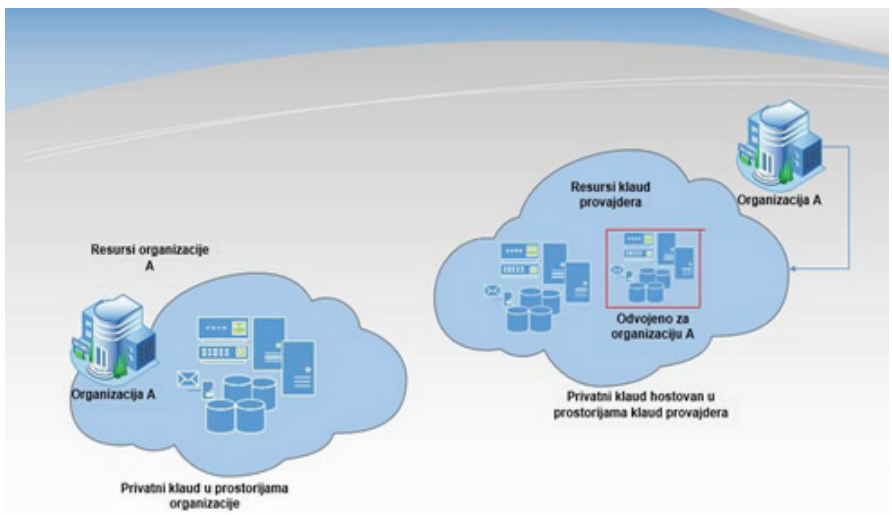
Iz perspektive vlasništva nad infrastrukturom, u literaturi se često navodi podela privatnog klauda na: privatni klaud koji se hostuje u prostorijama kompanije (eng. *on- premise private cloud*) i privatni klaud koji se eksterno hostuje u prostorijama klaud provajdera (eng. *externally-hosted private cloud*).

Oba tipa privatnog klauda prikazana su na Slici 6-5.

U slučaju da je kompanija i vlasnik klaud infrastrukture, celokupna infrastruktura se nalazi iza korporativnog firewall-a i pristupa joj se preko Intranet mreže organizacije putem šifrovanih komunikacionih kanala. Za razliku od drugih modela isporuke, privatni klaud obezbeđuje visoki nivo sigurnosti i privatnosti zato što je cela infrastruktura namenjena samo jednom korisniku. Zbog toga se pitanje izolovanosti sistema različitih korisnika ne dovodi u pitanje u ovom modelu.

U slučaju korišćenja infrastrukture dobavljača klaud usluge, dobavljači privatnog klauda implementiraju klaud okruženje za svaku kompaniju posebno. U ovom slučaju, takvo okruženje se uglavnom potpuno personalizuje, kako bi potrebe klijenta bile zadovoljene na najbolji mogući način.

Slika 6-5: *Tipovi okruženja privatnog klauda*



Prednosti korišćenja modela privatnog klauda mogu da se sumiraju na sledeći način:

- unapređenje sigurnosti. Privatni klaud omogućava čak veći nivo sigurnosti od korišćenja individualnih virtuelnih mašina zato što okruženje privatnog oblaka može da se izoluje od svih drugih korisnika klaud usluga. Takođe, ograničeni pristup koji može da se integriše sa pravilima *firewall*-a kompanije i politikama za udaljeni pristup predstavlja dodatni sloj sigurnosti,

- povećan nivo pouzdanosti. Za razliku od okruženja javnog klauda, privatni klaud omogućava veću pouzdanost zbog redudantne arhitekture koja je otporna na otkaze hardvera i koja se ne deli ni sa jednom drugom kompanijom,
- poboljšanje performansi. S obzirom da se infrastruktura privatnog klauda ne deli ni sa kim, performanse su mnogo bolje zato što je ceo raspoloživi hardver na raspolaganju samo jednom entitetu,
- povećana fleksibilnost. Za razliku od fizičkih mašina, virtuelne mašine mogu s lakoćom da se nadograde i da im se poboljšaju performanse. Takođe, u slučaju korišćenja više virtuelnih mašina, resursi mogu dinamički da se alociraju s jedne na drugu mašinu i
- potpuna kontrola. Ne postoje restrikcije za korišćenje privatnog klaud okruženja. Kompanije mogu svoje privatno klaud okruženje da podešavaju kako žele u skladu s poslovnim potrebama. Tako na primer, kompanija može da koristi bilo koji operativni sistem i bilo koje aplikacije, gde se resursi dinamički alociraju s jedne na drugu aplikaciju, i s jednog na drugi operativni sistem.

Kao jedan od značajnijih nedostataka privatnog klauda je taj što su kompanije same odgovorne za upravljanje svojim razvojnim platformama i aplikacijama. S jedne strane, ovo je dobro po pitanju kontrole i bezbednosti, dok s druge strane kompanije moraju da zaposle stručno *IT* osoblje za izvršavanje upravljačkih aktivnosti nad klaud okruženjem. Neki od dobavljača klaud usluga su prepoznali ovaj nedostatak i od skoro su svoje ponude klaud usluga obogatile virtuelnim desktop sistemima, preko kojih osoblje klaud dobavljača može da pomogne kompanijama u poslovima upravljanja i konfigurisanja privatnog klaud okruženja.

Kao drugi značajni nedostatak privatnog klauda navodi se cena iznamljivanja infrastrukture od dobavljača klaud usluge (u slučaju da kompanija nije u vlasništvu infrastrukture), koja znatno prevazilazi cenu korišćenja usluga javnog klauda.

U nastavku se sumarno navode nedostaci privatnog klaud okruženja:

- troškovi. Sa ekskluzivnim pravom dolazi i do uvećanja troškova. Ako kompanija hoće da kupi sopstvenu klaud infrastrukturu mora da se pripremi na značajnije kapitalne investicije. Na sreću, infrastruktura može da se iznajmi od dobavljača klaud usluga, u kom slučaju kompanija plaća samo fiksne troškove u vidu mesečnog zakupa opreme,
- neiskorišćenost resursa. U slučaju privatnog klauda, troškovi koji nastaju, nastaju kada svi raspoloživi resursi nisu potpuno iskorišćeni ne snosi dobavljač klaud usluga, već sama kompanija i

- skaliranje računarske platforme. Skaliranje je znatno teže nego u slučaju javnog klauda. Tako na primer, u situacijama rasta obima poslovanja, kada je potrebno nadograditi fizičku infrastrukturu, skaliranje računarske opreme se ne vrši brzo kao što je to slučaj sa virtuelnim mašinama, već skaliranje može da bude relativno dug proces.

Iako je na početku ovog koncepta bilo reči o suštini privatnog klauda, ipak je potrebno da se reše neke nedoumice o privatnom klaudu koje vladaju u stručnoj javnosti.

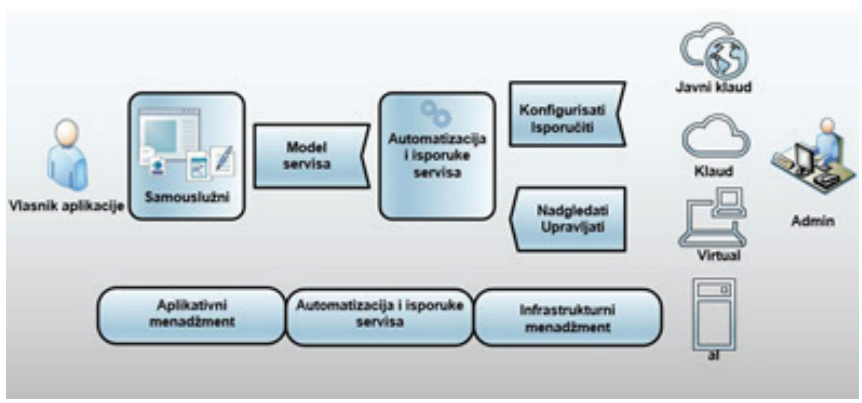
Kao što je već i navedeno, privatni klaud se u opštem smislu deli na dve kategorije: privatni klaud u prostorijama kompanije i eksterni privatni klaud.

Sam termin privatni klaud navodi na razmišljanje da poslovni subjekat poseduje klaud infrastrukturu koja se nalazi u njegovim prostorijama. Ovo je sasvim legitimno i tačno u pojedinim slučajevima, kada se radi o privatnom klaudu sa infrastrukturom koja je implementirana u prostorijama kompanije. Međutim, u nekim drugim slučajevima, poslovni subjekti svoje privatno klaud okruženje iznajmljuju od trećih strana koje su specijalizovane kao dobavljači klaud usluga, tako da u ovom slučaju infrastruktura nije u vlasništvu kompanije i za ovakvo klaud okruženje se kaže da je eksterno. Iznajmljivanjem okruženja kompanije pre svega smanjuju troškove, ali isto tako ostvaruju i druge koristi, kao što su na primer zaštita opreme i podataka od krađe i od prirodnih i drugih nepogoda. Dakle, još jednom se naglašava da je akcenat kod privatnog klauda korišćenje, a ne vlasništvo infrastrukture.

Među kompanijama koje nude usluge privatnog klauda ističu se sledeće: *Microsoft, IBM, Oracle, Cisco, NetApp, VMware, HPE, Amazon* i *Red Hat*.

Vizuelna reprezentacija *Microsoft* okruženja privatnog klauda data je na Slici 6-6.

Slika 6-6: *Microsoft privatni klaud*



6.3. ZAJEDNIČKI KLAUD

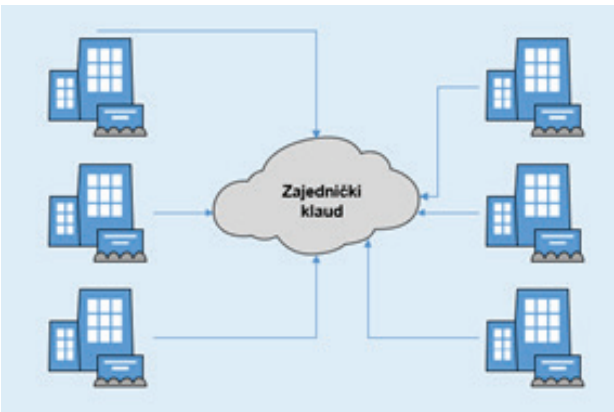
Zajednički klaud odnosi se na model isporuke, gde kompanije koje imaju slične interese, ciljeve, regulative, opšte politike i sigurnosne politike, dele i zajednički koriste klaud okruženje.

Kao primeri ovakvih kompanija mogu da se navedu medicinske, finansijske, neprofitne, vladine i druge organizacije.

Zajednički klaud može da bude u vlasništvu i da se njime upravlja od strane jedne ili više organizacija koje pripadaju zajednici, neke treće strane, ili kombinacije ova dva. Takođe, infrastruktura zajedničkog klauda može da se nalazi u prostorijama, ili van prostorija ovih organizacija.

Model zajedničkog klauda prikazan je na Slici 6-7.

Slika 6-7: *Zajednički klaud*



Zajednički klaud se bazira na modelu višestrukog zakupa, gde više kompanija sa sličnim interesima i ciljevima koriste istu platformu. Kao primer upotrebe zajedničkog klauda navodi se scenario kada više kompanija ima potrebu za korišćenjem specifične aplikacije koja se nalazi na jednom klaud serveru. Umesto da svaka kompanija dobije svoj server sa svojom instancom aplikacije, dobavljač klaud usluga može da dozvoli da se više kompanija simultano poveže sa aplikacijom uz logičko segmentiranje konekcionih sesija.

6.4. HIBRIDNI KLAUD

U praksi između tri navedena modela isporuke postoje „sive" i nedefinisane zone. U praksi se najčešće dešava scenario kada se povezuju klaud okruženja sa različitim modelima isporuke, kako bi se zadovoljile potrebe kompanije. Tako na primer, kompanija može da koristi usluge javnog klauda, u cilju nadogradnje funkcionalnosti svog privatnog klauda.

> Kompanija može da koristi privatni klaud za čuvanje osetljivih podataka o svojim klijentima, dok s druge strane može da koristi usluge javnog klauda za privremeno skladištenje i deljenje svakodnevnih, operativnih podataka između svojih zaposlenih.

U ovim slučajevima radi se o hibridnom modelu isporuke.

Prema zvaničnoj definiciji *Microsoft*-a, hibridni klaud je okruženje koje kombinuje javni i privatni klaud tako što omogućava deljenje aplikacija i podataka između ova dva modela isporuke.

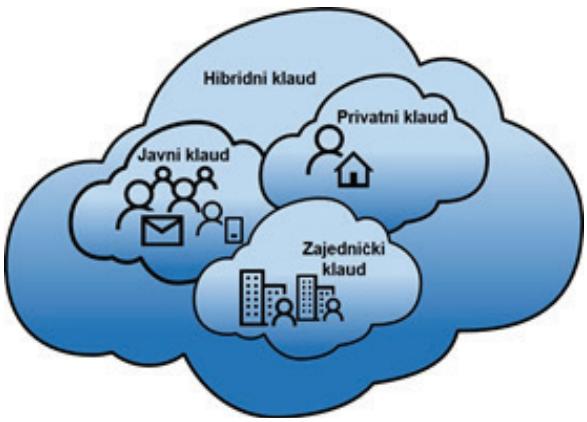
Kada zahtevi za računarskim resursima fluktuiraju, hibridni klaud omogućava kompanijama da prošire svoju privatnu klaud infrastrukturu korišćenjem javnog klauda, koji preuzima višak tereta, a pritom treća strana ne dobija pristup svim podacima kompanije. Na ovaj način kompanija dobija fleksibilnost i računarske resurse javnog klauda za zadatke koji ne koriste osetljive podatke, dok se kritične poslovne aplikacije (eng. *mission-critical*) i dalje izvršavaju na opremi koja se nalazi u prostorijama kompanije iza korporativnog *firewall*-a.

Upotreba hibridnog klauda ne omogućava samo kompanijama da prošire računarske resurse, već ida eliminiše potrebu za velikim kapitalnim izdacima, kako bi se kratkoročna povećanja u tražnji korporativnih usluga zadovoljila, i oslobađa lokalne računarske resurse za rukovanje osetljivim podacima i aplikacijama. Kompanije plaćaju samo za privremene resurse, bez kupovine i održavanja dodatne opreme koja se u budućnosti možda neće ni koristiti.

Prema nekim viđenjima, hibridni klaud kombinuje „najbolje od svih svetova platformi" i omogućava ključne benfite klaud računarstva – fleksibilnost, skalabilnost i efikasnost trošova uz minimalni rizik od gubitka podataka.

Hibridni klaud kao kombinacija privatnog, javnog i zajedničkog klauda dat je na Slici 6-8.

Slika 6-8: *Hibridni klaud*



6.5. IZBOR KLAUD OKRUŽENJA

Za kompaniju je jako važno da izabere tip klaud okruženja koji najbolje odgovara njenim potrebama.

Prema nekim viđenjima, kao pomoć pri izboru odgovarajućeg klaud okruženja, menadžment kompanije treba da pruži odgovore na sledeća tri pitanja:

- koji su osnovni zahtevi koje klaud platforma treba da zadovoljava?
- koji su zahtevi aplikacija koje će se izvršavati na klaud platformi?
- da li postoje neke posebne funkcije koje kompanija treba da isporuči svojim klijentima?

Javni klaud je platforma koja obezbeđuje računarske resurse u vidu procesorskog vremena, skladišta i/ili mreža u formi višestrukog zakupa i koju koristi više organizacija (kompanija) istovremeno. Kao jedna interesantna metafora za javni klaud iz svakodnevnog života navodi se da bi javni klaud mogao da se uporedi sa vožnjom autobusom, kada klijent ne poseduje sopstveni automobil.

Neke od praktičnih situacija u kojima je javni klaud optimalan izbor su sledeće:

- kada se razvija i testira jedna izolovana aplikacija,
- kada je brzo potrebno izvršiti nadogradnju i proširivanje resursa privatnog klauda,
- kada je potrebno brzo i na zahtev pribaviti računarske resurse za jednokratnu upotrebu i
- kada organizacija hoće da uspostavi kontrolisanu potrošnju računarskih resursa na osnovu nivoa utrošenih jedinica resursa.

Neke od činjenica koje su važne za eksploataciju resursa javnog klauda su:

- implementacija računarskih resursa javnog klauda je relativno jednostavna. Klijent samo treba da kreira nalog na portalu dobavljača usluga javnog klauda i da unese broj kreditne kartice. Kada resursi više nisu potrebni, oni se jednostavno odbace (eng. *discard*) preko portala dobavljača klaud usluga,
- zbog toga što se računarski resursi dele sa drugim organizacijama, ne preporučuje se da se na servisima javnog klauda instaliraju aplikacije koje koriste osetljive i poverljive podatke i
- s obzirom da se klijentu čini da su resursi javnog klauda neograničeni, potrebno je voditi računa o jedinicama utrošenih resursa, jer se često dešava da se izgubi uvid u ostvarenu potrošnju, kada troškovi za korišćenje resursa budu veći nego što je očekivano.

Nasuprot javnom klaudu postoji privatni klaud, gde računarske resurse ekskluzivno koristi samo jedna organizacija. U ovom slučaju organizacija ima potpunu kontrolu nad celokupnom klaud infrastrukturom. Kao jedna interesantna metafora za privatni klaud iz svakodnevnog života navodi se da bi privatni klaud mogao da se uporedi sa vožnjom sopstvenim automobilom, kada klijent nema potrebu da kao prevozno sredstvo koristi javni prevoz, koji deli sa drugim klijentima.

Neke od praktičnih situacija u kojima je privatni klaud optimalan izbor su sledeće:

- kada je potrebno da se upravlja životnim ciklusom više aplikacija,
- u slučaju implementacije platforme za e-trgovinu, kada organizacija čuva osetljive podatke o svojim klijentima, kao što su brojevi kreditnih kartica,
- kada postoje permanentni zahtevi za „stabilnim“ računarskim resursima,

- kada postoje zahtevi za „trajnim“ računarskim resursima koje aplikacije u klaudu koriste i
- kada postoje zahtevi u pogledu raspoloživosti računarskih resursa različitim poslovnim jedinicama koje su međusobno izolovane.

Neke od činjenica koje su važne za korišćenje privatnog klauda su:

- kada su kontrola, sigurnost, saglasnost (kompatibilnost) i lokacija resursa glavni zahtevi koje klaud platforma treba da zadovolji, privatni klaud je optimalan izbor,
- privatni klaud zahteva mnogo veću odgovornost nego javni klaud u kontekstu projektovanja, implementiranja i upravljanja klaud okruženjem. U ovom slučaju uprava kompanije treba da donese važne odluke koje se odnose na hardver, mreže i skladišta podataka i
- platforma dolazi sa visokim početnim troškovima koji često spadaju u klasu kapitalnih investicija, dok su operativni troškovi korišćenja platforme relativno stabilni tokom vremena.

Treći tip, hibridni klaud, predstavlja kombinaciju privatnog i javnog klauda, gde organizacija koristi relativno stabilne privatne računarske resurse u kombinaciji sa računarskim resursima javnog klauda, koji se koriste kada je potrebno privremeno da se nadograde privatni klaud resursi. Hibridni klaud predstavlja koordinirano i sinhronizovano korišćenje servisa privatnog i javnog klauda.

Kao neke od praktičnih situacija u kojima je hibridni klaud optimalan izbor navode se sledeće:

- kada se važnije operacije izvršavaju na privatnom klaudu, dok se povremeni zahtevi za povećanim resursima obezbeđuju preko javnog klauda,
- kada se razvija troslojna aplikacija, gde se sloj Web interfejsa implementira na javnom klaudu, dok se aplikaciona logika i backend implementiraju na servisima privatnog klauda i
- kada često dolazi do fenomena tzv. pucanja klauda (eng. *cloud bursting*). Organizacija važne servise koristi u privatnom klaudu, a kada tražnja za resursima prekorači kapacitete privatnog klauda, koriste se resursi javnog klauda kako ne bi došlo do otkaza usluga.

Neke od činjenica koje su važne za korišćenje hibridnog klauda su:

- hibridni klaud inkorporira najbolja svojstva iz oba sveta. Ovaj model omogućava korišćenje javnog i privatnog klauda zajedno u sinhronizovanom ekosistemu klaud servisa, ili nezavisno, ako za tim postoji potreba,
- primena modela hibridnog klauda zahteva implementaciju kratkoročne i dugoročne strategije za upravljanje klaud resursima i sinhronizaciju aktivnosti, kako bi potrošnja resursa privatnog i javnog klauda bila jednostavna i predvidiva. Slično kao i u slučaju javnog klauda, u slučaju nedostatka kontrole, troškovi eksploatacije resursa mogu značajno da se povećaju i
- uspostavljanje optimalnog balansa (odnosa) između nivoa upotrebe stabilnih resursa privatnog i resursa javnog klauda u situacijama kada su zahtevi za resursima povećani. Ovaj balans najbolje može da se odredi pažljivom analizom mrežnog saobraćaja i primenom analitičkih alata, koji su dostupni, kako od strane klaud provajdera, tako i na javnim repozitorijumima, poput *Github*-a.

Neki od slučajeva korišćenja javnog, privatnog i hibridnog klauda prikazani su na Slici 6-9.

Slika 6-9: *Slučajevi korišćenja klaud modela*





PITANJA ZA RAZMATRANJE

1. Navesti četiri modela isporuke klaud računarstva.
2. Objasniti model isporuke javni klaud. Dati primer.
3. Objasniti model isporuke privatni klaud. Dati primer.
4. Navesti predosti korišćenja modela privatnog klauda.
5. Navesti nedostatke korišćenja modela privatnog klauda.
6. Objasniti model isporuke zajednički klaud. Dati primer.
7. Objasniti model isporuke hibridni klaud. Dati primer.
8. Koja su najčešća pitanja koja treba da se postave kada se bira odgovarajuće klaud okruženje? Objasniti kako kompanije biraju klaud okruženje.
9. Navesti u kojim praktičnim situacijama bi rešenje privatnog klauda bilo optimalno?
10. Navesti u kojim praktičnim situacijama bi rešenje javnog klauda bilo optimalno?
11. Navesti u kojim praktičnim situacijama bi rešenje hibridnog klauda bilo optimalno?



MIGRACIJA RESURSA NA KLAUD

DEO III

MIGRACIJA RESURSA NA KLAUD



U trećem delu udžbenika prikazan je proces migriranja računarskih resursa na klaud okruženje i predstavljeni su aspekti, izazovi i potencijalni problemi sa kojima se entitet koji usvaja ovaj „staro-novi" način organizovanja računarskih resursa suočava.

Sve više organizacija se odlučuje da tradicionalni računski centar, koji se nalazi u prostorijama kompanije, zameni virtuelizovanim, softversko definisanim računskim centrom u klaud okruženju. Samim tim, jedan od najvećih izazova jeste migriranje računarskih resursa na klaud okruženje, što u najvećem broju slučajeva predstavlja težak i dugotrajan proces.

U prvoj polovini trećeg dela udžbenika veći akcenat je stavljen na poslovne i netehničke aspekte migriranja na klaud okruženje, koji moraju da se uzmu u obzir da bi se proces migracije izveo bezbolno i bez remećenja svakodnevnih operativnih aktivnosti. Neki od netehničkih aspekata migracije odnose se na procese planiranja migracije i upravljanje promenama, izradu neophodne dokumentacije i na klaud radni tok.

Svaka promena, a posebno velika promena, kao što je transformacija tradicionalne računarske platforme u klaud arhitekturu, je teška i nailazi na brojne prepreke u njenom sprovođenju, kako ljudske, tako i tehničko-organizacione prirode. Zbog toga su poslovni i netehnički aspekti migracije podjednako važni kao tehnički.

U drugoj polovini trećeg dela udžbenika veći fokus je stavljen na tehničke aspekte migracije, pre svega iz ugla migriranja računarskih resursa na klaud. U ovom delu detaljno su prikazani izazovi i problem tokom migriranja servera i skladišta podataka na klaud, implementacije klaud računarske mreže i usaglašavanja fizičkih sa virtuelnim klaud računarskim resursima.

S obzirom da se na klaudu uglavnom koriste virtuelni resursi, poput virtuelnih servera, skladišta podataka i mreža, potrebno je posebnu pažnju posvetiti usaglašavanju fizičkih sa virtuelnim resursima, jer se često dešava da u praktičnim implementacijama virtuelni resursi budu previše, odnosno premalo dimenzionirani

u odnosu na fizičke resurse. U ovakvim scenarijima, servisi klaud računarstva se ne izvršavaju dovoljno pouzdano, ili dolazi do generisanja bespotrebnih i prekomernih troškova.

Kako klaud provajderi, tako i korportivni klijenti i organizacije, koje imaju sopstveno privatno klaud okruženje, teže ka tome da se njihove klaud usluge i servisi izvršavaju pouzdano i bez zastoja, uz postizanje zadovoljavajuće kvaliteta. Zbog toga je važno da se na klaudu implementiraju mehanizmi za oporavak od katastrofe, čime se obezbeđuje tzv. kontinuitet poslovanja. S pogledom na ovu činjenicu, u ovom delu udžbenika detaljno je prikazana i implementacija redudantnosti, kako na nivou posebnih računarskih resursa, tako i na nivou celog klaud sajta, i mehanizama za replikaciju i siguran transfer podataka.

7. IZVRŠAVANJE MIGRACIJE NA KLAUD

Da bi se uspešno izvršila migracija računarskih resursa sa tradicionalnih platform (platformi koje se baziraju na računskom centru koji se nalazi u prostorijama kompanije) na klaud platformu, potrebno je da se razume proces pripreme i proces izvršavanja migracije na klaud okruženje.

Zbog toga je u prvom delu ove glave detaljnije prikazano planiranje procesa migracije računarskih resursa na klaud okruženje, sa posebnim akcentom na upravljanje promenama i izradom prateće dokumentacije. Zatim su prikazani modeli implementacije klaud okruženja, koji su slični tipovima klaud računarstva prema modelu isporuke. Konačno, na kraju ove glave, dat je opis migracije (implementacije) računarske mreže na klaud okruženje.

7.1. PLANIRANJE PROCESA MIGRACIJE

Prilikom migracije resursa na klaud okruženje, jedan od najvažnijih koraka jeste planiranje prebacivanja operacija na klaud.

Kao rezultat planiranja generiše se plan migracije, koji se sastoji iz sledećih delova: upravljanje promenama, informisanje svih zainteresovanih strana o promenama, postavljanje realnog vremenskog okvira za sprovođenje migracije, dokumentovanje i praćenje procedura.

U nastavku će biti ukratko opisani svi delovi iz kojih se sastoji plan migracije.

7.1.1. Upravljanje promenama

Upravljanje promenama (eng. *change management*) je proces pomoću koga se upravlja svim tekućim nadogradnjama, popravkama i konfiguracijama klaud servisa. Krajnji cilj upravljanja promenama je minimizacija, ili eliminacija bilo kakvih prekida u radu sistema.

Prilikom migriranja resursa na klaud, ogranizacije se suočavaju sa velikim brojem promena, kako u tekućim poslovnim operacijama, tako i sa promenama u korišćenju infrastrukturnih resursa. Zbog toga se proces upravljanja promenama treba uzeti u obzir vrlo ozbiljno.

Uspešno upravljanje promenama zahteva kreiranje standardnih procesa koji garantuju uspešno izvršavanje procesa upravljanja promenama: snimanje izmene, planiranje izmene, testiranje dokumentacije, dobijanje odobrenja za izvršavanje promene, evaluacija i potvrđivanje (validacija) promene, instrukcije za kreiranje rezervne kopije promene i revizija promene.

Upravljanje promenama je standarni proces u operacijama klaud ili korporativnog računskog centra, koji koristi poznate i uhodane procedure za implementiranje promena postojećih operacija. Često se dešava da organizacije prilikom migracije svojih računarskih resursa na klaud okruženje proces upravljanja promenama potcene ili zanemare, što je za posledicu da nova klaud platforma ne zadovoljava sve tekuće i buduće korporativne potrebe za računarskim resursima na najbolji način. Zbog toga je efikasno upravljanje promenama prilikom migracije na klaud okruženje od ključne važnosti za uspešnu implementaciju klaud infrastrukture.

Uspešna implementacija upravljanja promenama ograničava uticaj koji ima, u smislu remećenja tekućih poslovnih operacija, proces migracije na klaud okruženje.

Procedura za upravljanje promenama u velikom broju slučajeva zahteva da se navede ime osobe koja je predložila promenu, detaljan opis promene i razlog, ili opravdanje za sprovođenje promene. U određenim slučajevima navodi se i očekivani rezultat promene, potencijalni rizici, resursi koji su neophodni, uz istovremenu koordinaciju aktivnosti različitih grupa ljudi koje su uključene u promenu. Za svaki korak promene, kao što su dizajn, konfiguracija, raspoređivanje resursa i validacija mora da se navede lista individualnih odgovornosti i obaveza.

Često se dešava da se više promena izvode paralelno i zbog toga je potrebno detaljno da se ispita da li je jedna promena u konfliktu sa nekom drugom promenom. Takođe, ponekad izvođenje jedne promene zahteva da se pre toga izvrši neka druga promena i u tom slučaju potrebno je definisati redosled izvršavanja promena, koje u tom slučaju treba koordinirano sprovoditi.

7.1.2. Informisanje svih zainteresovanih strana o promenama

Jedan od neophodnih elemenata plana migracije jeste kontinuirano obaveštavanje svih zainteresovanih strana o napretku u izvršavanju planova i o rezultatima.

Sve zainteresovane strane je potrebno posebno obaveštavati o vremenskim rokovima za izvršavanje promena i o promenama koje mogu da imaju uticaj na izvršavanje tekućih operacija u organizaciji. Ovakvo informisanje prevazilazi okvire zaposlenih u korporativnom računskom centru i ono se odnosi i na zaposlene u drugim organizacionim jedinicama, kao što su finansije, proizvodnja, sektor za upravljanje ljudskim resursima i slično.

U mnogim srednjim i većim organziacijama formira se posebni odbor za reviziju promena. Ovaj odbor se uglavnom sastoji iz menadžera, predstavnika računskih (*IT*) odeljenja i predstavnika stejkholdera organizacije. Odbor je odgovoran za odobravanje svih predloženih promena.

Odbor za reviziju promena preuzima odgovornost za upravljanje rizicima i treba da garantuje da neće doći do konflikata između promena koje su zakazane u istom vremenskom intervalu. Odbor daje odgovore na sledeća pitanja:

- koji je razlog za implementiranje promene?
- koji su očekivani rezultati sprovođenja promene?
- koji rizici postoje u slučaju da promena bude implementirana?
- koje su posledice i rizici neimplementiranja promene?
- koje organizacione jedinice će da učestvuju u sprovođenju promene?
- koliko dugo je potrebno da bi se promena sprovela i da bi se izvršila revizija promene?
- ako je potrebno, koliko dugo će trebati da se implementirana promena poništi?

Osim navedenog, odbor za reviziju promena takođe treba detaljno da analizira planove za verifikovanje i povlačenje promene, u cilju smanjivanja rizika sprovođenja promene.

7.1.3. Definisanje realnog vremenskog okvira za sprovođenje migracije

U cilju uspešnog sprovođenja migracije računarskih resursa na klaud okruženje potrebno je definisati precizne i realne vremenske okvire za izvršavanje ovog procesa.

U većini situacija najbolje je da se migracija sprovede u inkrementalnim i manjim koracima, kako bi se minimizovao rizik od prekida izvršavanja tekućih poslovnih operacija i gubitka resursa.

Takođe, sprovođenje migracije u manjim koracima je bolja opcija zato što može da se desi da je potrebno da se odustane od migracije zbog nepredviđenih okolnosti i u tom slučaju će sistem moći u kraćem vremenskom periodu da se vrati na stanje koje je bilo pre migracije. Proces migracije treba da se vrši u okviru vremenskog perioda koji je predviđen za održavanje sistema (eng. *system maintenance window*) i koji se unapred zakazuje u skladu sa srednjoročnim i dugoročnim planovima upotrebe računarskih resursa.

U većini slučajeva migracija će biti uspešna i imaće najmanji uticaj na tekuće poslovne operacije ako se u izvršavanju migracije poštuje sledeći princip: „prvo migrirati manje sisteme, a zatim migraciju sprovoditi inkrementalno tokom vremena".

U skladu s ovim principom, migraciju je potrebno početi s manjim i jednostavnijim sistemima koji nisu kritični za izvršavanje tekućih poslovnih operacija. Na ovaj način, organizacija će da stekne iskustvo u izvršavanju procesa migracije i precizniji vremenski rokovi za izvršavanje procesa migracije će moći da se definišu.

Kada se govori o definisanju vremenskog roka za izvršavanje migracije, potrebno je da se napomene da je bolje da se alocira više vremena za pojedine faze migracije, kako bi se uzelo u obzir i vreme koje će biti potrebno za rešavanje nepredviđenih scenarija i konflikata.

7.1.4. Izrada dokumentacije i praćenje procedura

Jedna od važnijih i kritičnijih komponenti uspešne migracije računarskih resusa na klaud okruženje jeste dobro pripremljena i sveobuhvatna dokumentacija.

S vremena na vreme potrebno je da se izvrši revizija svih pratećih dokumenata i da se ažurira, ako za tim postoji potreba. Potrebno je kreirati dijagrame svih sistema i dokumentovati konfiguracije sistema.

Tim projektnih menadžera koji učestvuju u procesu migracije resursa na klaud okruženje je pre svega odgovoran za kreiranje i održavanje sveobuhvatne i precizne dokumentacije o svim aspektima migracije. Osnov za dokumentaciju predstavljaju podaci koji se prikupljaju praćenjem aktivnosti i saobraćaja na računarskoj mreži pomoću softverskih i hardverskih alata za upravljanje mrežom, preuzimanjem konfiguracije mrežnih uređaja, kao i proučavanjem i analizom pratećih dokumenata proizvođača hardvera.

Izrada kvalitetne dokumentacije je takođe važna i u kontekstu održavanja klaud okruženja i računarskih mreža. Potrebno je napraviti detaljnu dokumentaciju o planovima dodeljivanja *IP* adresa mrežnim uređajima, protokolima rutiranja i sigurnosnim protokolima, virtuelnim privatnim mrežama (eng. *virtual private local area network – VLAN*), redudantnosti sistema i konekcijama između različitih mreža i mrežnih segmenata, uz precizno navođenje svih mrežnih lokacija na kojima će se koristiti sigurnosni mrežni uređaji.

S pogledom na činjenicu da će tokom procesa migracije klaud operacije kontinuirano da se razvijaju i da se menjaju, prateću dokumentaciju je potrebno stalno ažurirati, kako bi bila važeća (eng. *up-to-date*) i kako bi u potpunosti odgovarala tekućoj konfiguraciji klaud infrastrukture.

U prvoj fazi migracije na klaud okruženje treba da se krene sa procesom planiranja mrežnih resursa i izradom prateće dokumentacije, kao i sa paralelnom saradnjom sa dobavljačem klaud usluga. Tako je moguće identifikovati dodatni mrežni hardver I softver koji je potrebno nabaviti, a koji dobavljač klaud usluga ne poseduje.

Dokumentacija o mreži treba da sadrži detaljna objašnjenja i dijagrame mrežne topologije sa podacima o *IP* adresiranju svih mrežnih interfejsa i virtuelnih privatnih mreža.

U slučaju izostanka validne dokumentacije o mreži, skripte za automatsko konfigurisanje mrežnog okruženja neće moći da se primene u početnim fazama migracije na klaud.

Osim navedenog, mrežna dokumentacija treba da obuhvati i delove koji se odnose na pristup i distribuciju mreža na klaudu. Dokumenti i mrežni dijagrami treba da prikažu sve informacije o protokolima, implementacionim tehnologijama, virtuelnim proširivim lokalnim mrežama (eng. *virtual extensible virtual local area network – VXLAN*), mestu povezivanja svakog klaud servera na mrežu i listom virtuelnih privatnih mreža na koje je server povezan, kao i o svim drugim značajnijim resursima mrežne infrastrukture.

U dokumentaciji o mreži treba navesti i detalje o povezivanju klaud mreže sa drugim širokopojasnim mrežama, pristupne linkove virtuelne privatne mreže (eng. *virtual private network – VPN*), tabele rutiranja, pristupne kontrolne liste, konekcije ka mreži dobavljača klaud usluga i linkove ka organizacionim jedinicama kompanije i korporativnom računskom centru.

Posebna sekcija mrežne dokumentacije koja se odnosi na servise treba da obuhvati detaljne informacije koje se odnose na sisteme za keširanje podataka, sisteme imena domena (eng. *domain name systems – DNS*), logove, sisteme za balansiranje mrežnog opterećenja (eng. *load balancers*), servere za mrežnu optimizaciju, informacije o sistemima za detekciju upada u sistem (eng. *intrusion detection system – IDS*) i sistemima za sprečavanje upada u sistem (eng. *intrusion prevention systems – IPS*), kao i svim dostupnim alatima za ispitivanje mreže.

Kada se govori o izradi dokumentacije prilikom migriranja resursa na klaud okruženje potrebno je da se skrene pažnja da je moguće da detaljna dokumentacija o računarskoj mreži neće biti dostupna ako korporacija koristi računarsku mrežu klaud provajdera.

7.1.5. Klaud radni tok

U standardne ponude većine klaud provajdera uključene su tzv. arhitekture klaud radnog toka (eng. *cloud workflow architectures*) koje služe za upravljanje stanjem klaud projekta.

Moderne aplikacije se često oslanjaju na interakciju i interoperatiblnost sa različitim komponentama, kao i sa drugim aplikacijama u klaud okruženju, što je neophodan uslov za implementaciju klaud radnog toka.

Klaud radni tok (eng. *Cloud workflow*) definiše se kao niz koraka, ili aktivnosti koje je potrebno izvršiti u cilju završetka određenog zadatka na klaud okruženju.

Tako na primer, ako organizacija poseduje portal za e-trgovinu, izvršavanje jedne *online* transakcije sastoji se iz više koraka, kao što su ubacivanje artikala u potrošačku korpu, plaćanje, finansijske transakcije, upravljanje skladištima proizvoda, dostava, itd. Svi koraci pripadaju jednom klaud projektu. Svaki korak ima određene zahteve i uslove koje treba postići pre njegovog izvršavanja. Primenom arhitekture klaud radnog toka, moguće je automatizovati proces praćenja izvršavanja svih navedenih koraka.

Kada se posmatra u kontekstu migracije resursa na klaud, servis klaud radnog toka upravlja svim koracima koje je potrebno izvršiti kako bi se proces migracije uspešno završio. Klaud radni tok može da se definiše i kao praćenje stanja i koordinacija sistema na kladu.

U slučaju kada tim projektnih menadžera dizajnira i implementira plan migracije resursa na klaud, koji se bazira na radnim tokovima, moguća je primena analitičkih alata kojima se generišu važne poslovne informacije.

7.1.6. Klaud automatizacija

Klaud automatizacija (eng. *cloud automation*) je jedna glavna karakterisitka virtuelizovanih računskih centara.

Automatizaciju u modelu javnog klauda obezbeđuje dobavljač klaud usluga. Servisi za automatizaciju su klijentu dostupni u formi *Web* komandne table (eng. *Web dashboard*), aplikativnog programskom interfejsa (eng. *application programming interface – API*), ili konzolnog interfejsa (eng. *command-line interface – CLI*).

Većina vendora klaud usluga nude tzv. globani sistem za upravljanje klaud okruženjem koji omogućava između ostalog i automatizaciju upravljačkih funkcija hibridnog klaud sistema koji povezuje više različitih klaud modela.

7.1.7. Klaud alati i sistemi za upravljanje

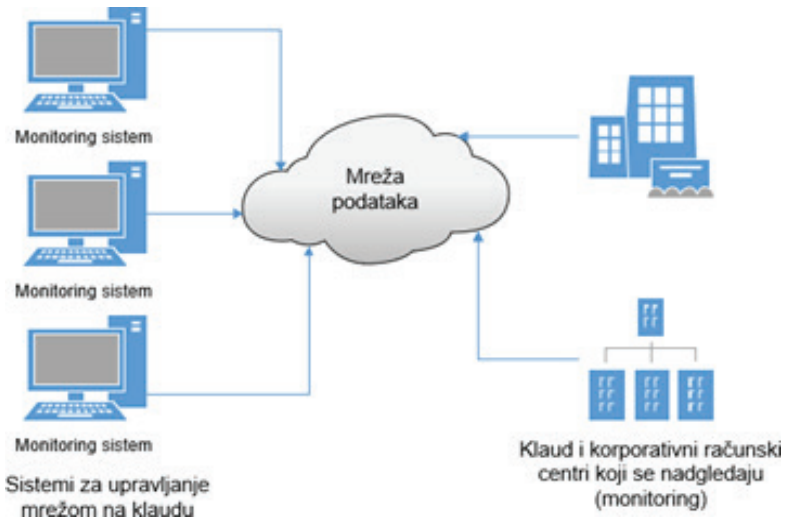
Upravljanje i praćenje (eng. *management and monitoring*) klaud resursa je ključna komponenta uspešnog implementiranja i funkcionisanja klaud okruženja. Permanentno praćenje funkcionisanja implementirane klaud aplikacije će obezbediti da sve komponente aplikacije funkcionišu u okviru predefinisanih zadovoljavajućih vrednosti metrika performansi.

Oblast koja se bavi upravljanjem klaud okruženja je relativno nova i kao takva još uvek evoluira. Klaud dobavljači stalno implementiraju nove servise u usluge iz ovog domena. Upravljanje i praćenje klaud okruženja je važno, jer se time i obezbeđuje da sve aplikacije i klaud servisi imaju zadovoljavajuće, ili optimalne performanse, da su sigurni, da rade onako kako se od njih i očekuje i da su konfigurisani u skladu sa preporukama.

Pod terminom alati za upravljanje klaudom obuhvaćene su brojne komponente, alati i usluge koje nudi klaud provajder, ili treća strana. Tako na primer, klasični alati za upravljanje mrežama su prošireni, kako bi se adaptirali za klaud servise, dok su s druge strane na tržište plasirani mnogi novi alati za upravljanje mrežama koji su dizajnirani i adaptirani isključivo za klaud okruženje. Takođe, uvidom u Internet resurse može da se vidi da su osnovane mnoge *startup* kompanije koje su specijalizovane za razvoj i održavanje alata za upravljanje i praćenje klaud resursa.

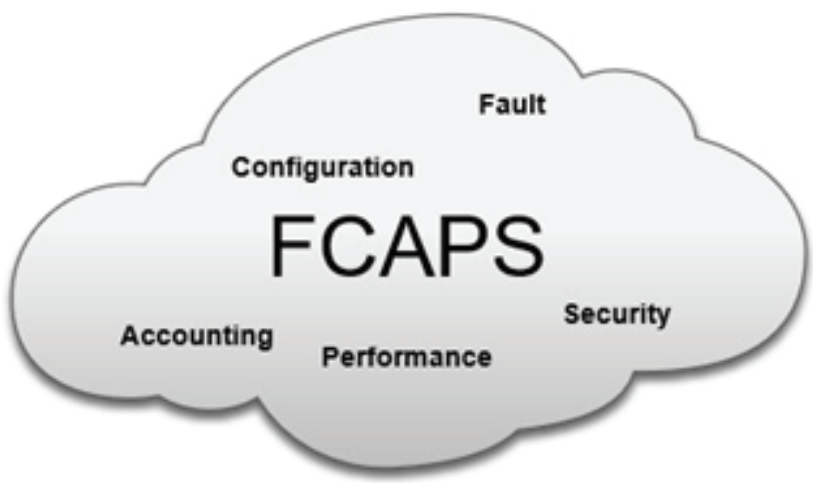
Sistemi za upravljanje mrežom na klaudu su relativno složeni i sofisticirani sistemi koji se sastoje iz više različitih komponenti. Osnovna arhitektura ovakvih sistema obuhvata jedan ili više centara za upravljanje mrežnim operacijama koji prate i prikupljaju informacije o udređajima koji se nalaze u privatnom ili javnom računskom centru. Arhitektura jednog ovakvog sistema prikazana je na Slici 7-1.

Slika 7-1: *Sistemi za upravljanje mrežom na klaudu*



U stručnoj literaturi, kao i u pratećoj dokumentaciji mnogih klaud vendora koristi se akronim *FCAPS*, koji se odnosi na glavne funkcije alata za upravljanje klaud mrežom. U akronimu *FCAPS*, slovo *F* označava *Fault*, *C* označava *Configuration*, *A* označava *Accounting*, dok slova *P* i *S* označavaju *Performance* i *Security*, respektivno (Slika 7-2).

Slika 7-2: *Glavne funkcije alata za upravljanje klaud mrežom*



U klaud okruženju postoji mnogo računarskih resursa, ili njihovih komponenti koji mogu da se prate, sa ciljem da se obezbedi nesmetani rad klaud aplikacija i servisa. Tako na primer, može da se prati iskorišćenost procesora, memorije i skladišta podataka na klaud serverima, statistika mrežnih intefejsa, temperatura hardverskih uređaja, logovi aplikacija, itd.

S pogledom na činjenicu da u klaud okruženju veliki broj klaud objekata podržava praćenje, nemoguće je, a u krajnjoj liniji, nije praktično pratiti sve ove objekte. Zbog toga je potrebno da se pažljivo napravi lista klaud objekata koji su ključni za tekuće i dugoročne klaud operacije i koji će biti uključeni u proces praćenja.

Za potrebe praćenja servisa koji podržavaju mogućnost praćenja (eng. *managed services*), potrebno je da instalira specijalizovana aplikacija za praćenje na serverskom objektu, koji zatim prikuplja metrike o performansama uređaja koji se prate (eng. *endpoint devices*). Moderni upravljački sistemi mogu da prikupljaju informacije sa servera, mrežnih uređaja, skladišnih sistema, *VPN* koncentratora, *firewall*-ova, kao i sa drugih klaud objekata.

Bez obzira na široku raspoloživost aplikacija, alata i servisa za upravljanje i praćenje sistema i podataka na klaudu, sama priroda klaud okruženja otežava ovaj proces i zbog toga je sisteme za praćenje mnogo lakše implementirati u lokalnom okruženju.

7.2. MODELI IMPLEMENTACIJE KLAUD OKRUŽENJA

Prilikom migracije računarskih resursa na klaud okruženje, organizacija pre svega treba da odluči o modelu implementiranja klaud okruženja koji će inkorporirati, kako bi na najbolji način bile zadovoljene sve tekuće, ali i buduće potrebe za *IT* resursima.

Modeli implementacije klaud okruženja odgovoraju podeli klaud računarstva prema modelu isporuke, o čemu je bilo više reči u Glavi 6. Međutim, s obzirom da su modeli implementacije od suštinske važnosti za razumevanje pojma klaud računarstvo, o njima će biti reči i u ovom poglavlju.

> Kako se tehnologija i tržište klaud računarstva brzo razvija i menja, tako se stvaraju i razvijaju novi implementacioni modeli klaud računarstva. Međutim, bez obzira na turbulentnost u ovom domenu, i dalje mogu da se izdvoje četiri osnovna modela implementacije klaud okruženja, koja obuhvataju javni, privatni, zajednički i hibridni klaud.

S obzirom da je o ovim modelima bilo reči ranije, ovde će biti navedene samo njihove osnovne karakteristike.

Model implementacije javnog klauda predstavlja infrastrukturu koju koriste javni korisnici.

Kao najčešći dobavljači usluga javnog klauda navode se privatne organizacije, vladine organizacije i akademske institucije. Usluge koje nudi javni klaud *hostuju* se u prostorijama dobavljača usluge, dok se model korišćenja zasniva na korišćenju zajedničkog hardvera, kao što je to prikazano na Slici 7-3.

Model implementacije privatnog klauda odnosi se na infrastrukturu koja je namenjan ekskluzivnom korišćenju od strane samo jedne organizacije, ili više poslovnih jedinica te organizacije.

Privatni klaud, kao model implementacije, dat je na Slici 7-4.

Slika 7-3: *Zajedničke usluge javnog klauda*



U ovakvom modelu računarski resursi mogu da budu u vlasništvu organizacije koja ih koristi, neke treće strane, ili kombinacija ova dva. Računarski resursi takođe mogu da se nalaze u prostorijama kompanije korisnika (eng. *on-site*), ili u prostorijama neke treće strane (eng. *off-site*). Kao ključna karakteristika ovog modela, koja ga razlikuje od drugih modela implementacije klaud okruženja, navodi se arhitektura u vidu posvećenih hardverskih resursa (eng. *dedicated hardware design*), koji su namenjeni za korišćenje samo jednoj organizaciji, za razliku od javnog klauda, gde se koristi arhitektura zajedničkih hardverskih resursa (eng. *shared hardware design*), gde iste resurse simultano koristi više organizacija.

Slika 7-4: *Model implementacije privatnog klauda*



Model implementacije zajedničkog klauda odnosi se na infrastrukturu koju koristi zajednica kompanija koje dele isti ili sličan skup vrednosti, ciljeva, poslovnih potreba, regulativa, zahteva u vidu sigurnosti, itd.

Upravljanje zajedničkim klaudom mogu da vrše kompanije koje ga koriste, specijalizovani dobavljač klaud usluga, ili neka treća strana. Takođe, kao i slučaju privatnog klauda, infrastruktura može fizički da se nalazi u prostorijama kompanija korisnika, ili u prostorijama dobavljača klaud usluga.

Zajednički klaud kao implementacioni model prikazan je na Slici 7-5 .

Slika 7-5: *Model implementacije zajedničkog klauda*



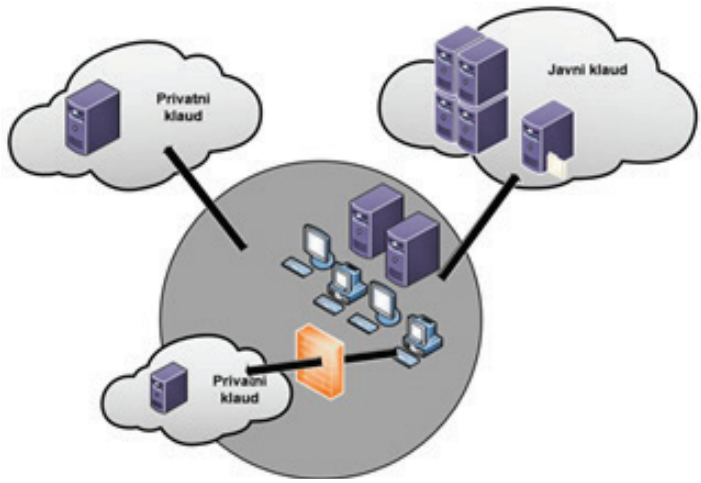
Implementacioni model hibridnog klauda obuhvata kombinaciju dva ili više modela javnog, privatnog i zajedničkog klauda.

S obzirom da se poslovni zahtevi i hardver neprekidno menjaju, a s pogledom na optimizaciju troškova i poslovnu efikasnost, hibridni klaud često predstavlja najbolju alternativu implementacije klaud okruženja.

Hibridni klaud često se koristi kako bi se predupredio fenomen tzv. pucanja klauda. Pucanje klauda je scenario kada resursi privatnog klauda nisu dovoljni za opsluživanje svih potreba za informacijama i servisima, i onda se nedostatak privatnih resursa nadoknađuje korišćenjem infrastrukture javnog klauda. Takođe, u pojedinim scenarijima kompanija iznajmljuje resurse javnog klauda kako bi implementirala mehanizme za balansiranje opterećenja i time podigla kvalitet pruženih servisa.

Primer kada jedna organizacija koristi kombinaciju privatnog i javnog klauda dat je na Slici 7-6.

Slika 7-6: *Povezivanje privatnog i javnog klauda jedne organizacije*



7.3. IMPLEMENTACIJA I MIGRACIJA RAČUNARSKE MREŽE NA KLAUD

Računarske mreže su široka oblast i neizostavni deo klaud računarstva.

> Proces migracije i implementiranja računarskih mreža u klaud okruženju se ne razlikuju mnogo od klasične implementacije mreže u prostorijama kompanije.

U svakom slučaju, za uspešno izvršavanje procesa migracije i implementacije mreže u klaud okruženju potrebno je razumevanje osnovnih koncepata i principa računarskih mreža. Zbog toga su ovom poglavlju ovi osnovni principi prikazani.

7.3.1. Klaud mrežni protokoli

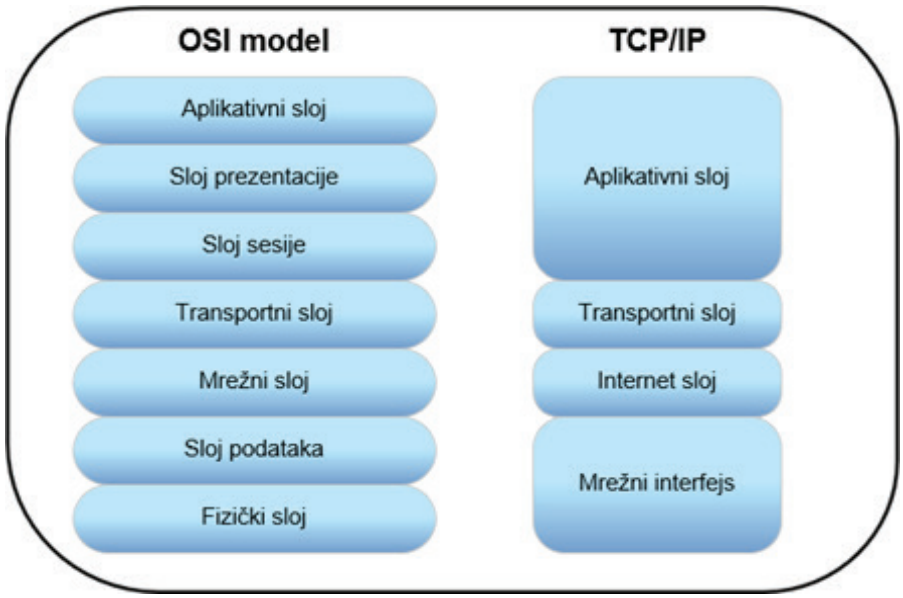
U svakoj računarskoj mreži koja koristi *TCP/IP* (*Transmission Control Protocol/ Internet Protocol*) stek, najčešće se koristi veliki broj mrežnih protokola. Neki protokoli su poznatiji, kao na primer protokoli za slanje email poruka, pristup *Web* serveru, ili razrešavanje imena domena.

> Većina klaud računarskih mreža bazira se na *TCP/IP* tehnologiji i koristi isti set mrežnih protokola kao i klasične računarske mreže (mreže koje su implementirane u prostorijama kompanije).

U ovom delu udžbenika u opisu mrežnih protokola prikazane su samo najosnovnije stvari, zato što se pretpostavlja da čitalac poseduje osnovno znanje o računarskim mrežama i o *TCP/IP* i *ISO/OSI* referentnim modelima.

Odnos između *TCP/IP* steka protokola i *ISO/OSI* modela prikazan je na Slici 7-7.

Slika 7-7: *ISO/OSI referentni model i TCP/IP stek protokola*



Prema *TCP/IP* specifikacijama, poznatijim aplikacijama i servisima su dodeljeni sopstveni brojevi portova u intervalu [0,1023] i ovi portovi se u literaturi nazivaju dobro poznati portovi (eng. *well-known port numbers*), ili trajni portovi (eng. *non- ephemeral port numbers*). Portovi iznad 1023 nazivaju se privremeni portovi (eng. *ephemeral port numbers*) i oni se najčešće koriste za manje poznate aplikacije i servise.

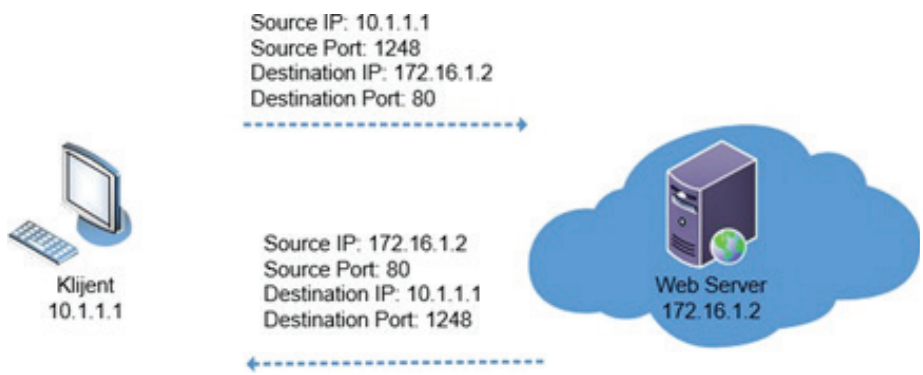
Klaud servisi uglavnom dobijaju neki od brojeva porta u interval od 0 do 1023. Kada aplikacija treba da pristupi nekom servisu na klaudu, polju u *TCP* zaglavlju će za vrednost odredišnog porta da se dodeli vrednost dobro poznatog porta. Kada takav *IP* paket dođe do udaljenog servera, server će na osnovu vrednosti koja je dodeljena polju za odredišni port da prosledi podatke aplikaciji kojoj taj port pripada na dalju obradu.

Tako na primer, ako korisnik ukuca u Web brauzer adresu www.singidunum.ac.rs, *Web* brauzer će videti da se deo www u adresi odnosi na *Web* saobraćaj i koristiće *HTTP* aplikativni protocol, koji koristi dobro poznati port 80. Dakle, podaci koje klijent šalje udaljenom *Web* serveru biće poslati na port 80. Podaci se onda enkapsuliraju u *TCP* segment na transportnom sloju *ISO/OSI* modela. Taj segment će zatim biti enkapsuliran u *IP* paket na mrežnom sloju *ISO/OSI* modela i biće poslat pomoću nižih slojeva koji pripadaju sloju mrežnog interfejsa, kao što je na primer *Ethernet*.

Međutim, kada se podaci šalju *Web* serveru koji se nalazi na klaudu, nije dovoljno u zaglavlje paketa upisati samo odredišnu *IP* adresu servera i broj porta, već je potrebna i adresa lokalnog klijentskog računara koji je zahtev uputio, kao i lokalni broj porta. Računar koji šalje zahtev, za lokalni broj porta izabraće pseudo-slučajan broj koji je veći od 1023.

Komunikacija između lokalnog računara i *Web* servera na kladu prikazana je na Slici 7-8. Na prikazanoj slici korišćena je fiktivna adresa *Web* servera, koja spada u domen privatnih *IP* adresa, za ilustrativne potrebe.

Slika 7-8: *Prikaz komunikacije između lokalnog klijenta i klaud Web server*



Neki protokoli ili aplikacije koriste samo *TCP* ili *UDP* protokole za transport podataka, dok drugi koriste i TCP i UDP transportne protokole. *TCP* (eng. *Transmission Control Protocol*) je pouzdani protokol transportnog sloja koji garantuje isporuku paketa i garantuje redosled po kom se paketi isporučuju. S druge strane, *UDP* (*User Datagram Protocol*) spada u grupu protokola koji ne garantuju isporuku paketa i koristi se za aplikacije, gde je pravovremena isporuka važnija od isporuke bez greške, kao što je na primer prenos *video* podataka u realnom vremenu.

U nastavku su ukratko opisani neki od najpoznatijih protokola sa brojevima portova koji se koriste u klaud okruženju.

HTTP (*HyperText Transfer Protocol*) je dobro poznat protokol aplikativnog sloja *ISO/OSI* modela, koji se uglavnom koristi za pristup *Web* serverima u klaudu preko *Web* brauzera. *HTTP* koristi *TCP* port 80.

HTTPS (*HyperText Transfer Protocol Secure*) koristi *TCP* port 443. Ovaj dobro poznati protokol koristi šifrovanu konekciju između klijenta i klaud servera, kako bi u slučaju presretanja komunikacije od strane maliciozne osobe ili sistema osetljive informacije, poput onih sa portala za e-trgovinu i e-bankarstvo bile zaštićene, i kako zlonamerni napadač ne bi mogao da im pristupi.

FTP (*File Transfer Protocol*) je jedan od najstarijih protokola i koristi se za razmenu fajlova između udaljenih sistema u klaudu. *FTP* koristi *TCP* portove 20 i 21, gde se port 20 koristi za razmenu podataka, dok je port 21 namenjen za kontrolu prenosa podataka.

FTPS (*File Transfer Protocol Secure*) je verzija standardnog *FTP* protokola koja koristi šifrovanje komunikacionog kanala. *FTPS* koristi *TCP* portove 989 i 990. *FTPS* se bazira na *TLS* (*Transport Layer Security*) / *SSL (Secure Socket Layer*) protokolima za šifrovanje podataka.

SFTP (*Secure File Transfer Protocol*) je sličan *FTPS* protokolu, sa razlikom da ovaj protokol koristi *TCP* port 22 kako bi obezbedio siguran transfer fajlova.

SSH (*Secure Shell*) je kriptovana verzija *Telenet* protokola i najčešće se koristi za udaljeni pristup uređaja serverima u klaudu korišćenjem konzolnog, ili grafičkog interfejsa. *SSH* koristi *TCP* port 22 i funkcioniše sa drugim sigurnosnim protokolima.

DNS (*Doman Name System*) protokol se koristi za razrešavanja imena domena u *IP* adresu, koju zatim koristi *IP* protokol radi povezivanja na udaljeni uređaj na klaudu. Namenski serveri koji vrše ovu funkciju nazivaju se *DNS* serveri. *DNS* protokol koristi i *UDP* i *TCP* port 53.

DHCP (*Dynamic Host Configuration Protocol*) omogućava automatsko *IP* adresiranje uređaja na klaudu. *DHCP* server koristi *UDP* port 67, dok *DHCP* klijent koristi port 68.

SMTP (*Simple Mail Transfer Protocol*) je relativno jednostavan protokol koji se koristi za transfer email poruka između mail servera i klijenta i koristi *TCP* port 25.

U preglednoj Tabeli 7-1 data je potpunija lista protokola sa kratkim opisom i brojem porta.

Tabela 7-1: *Pregled protokola*

Protokol	Opis	Port
TCP	Vrši transfer fajlova između udaljenih lokacija na klaudu (obično zahteva autentifikaciju korisnika).	*TCP* 20 i *TCP* 21
SSH	Sigurno povezivanje na udaljeni uređaj u klaudu (obično emulacijom terminala).	*TCP* 22
SFTP	Obezbeđuje transfer fajlova preko *SSH* konekcije.	*TCP* 22
SCP	*Secure Copy Protocol* - obezbeđuje sigurnu uslugu transfera fajlova između uređaja na klaudu preko *SSH* konekcije i omogućava slanje informacije o vremenu i datumu fajla, što ne pruža običan *FTP* protocol.	*TCP* 22
Telnet	Povezivanje na udaljeni uređaj u klaudu (obično emulacijom terminala).	*TCP* 23
SMTP	Koristi se za slanje *email* poruka.	*TCP* 25
DNS	Razrešava imena domena u *IP* adrese.	*TCP* i *UDP* 53
TFTP	*Trivial File Transfer Protocol* – vrši transfer fajlova do udaljenog uređaja na klaudu (ne zahteva autentifikaciju korisnika).	*UDP* 69
DHCP	Dinamički dodeljuje informacije o *IP* adresama uređajima na klaudu (maska podmreže, podrazumevani mrežni prolaz, *IP* adresa i adresu *DNS* servera).	*UDP* 67
HTTP	Preuzima sadržaj sa *Web* servera u klaudu.	*TCP* 80
POP3	*Post Office Protocol version 3* - preuzima *email* poruke sa *mail* servera u klaudu.	*TCP* 110
NNTP	*Network News Transport Protocol* - omogućava čitanje i pisanje članaka na *Usenet* serverima sa vestima.	*TCP* 119
NTP	*Network Time Protocol* – omogućava uređaju na klaudu da sinhronizuje svoj lokalni sat sa vremenskim podešavanjima mrežnog servera za vreme.	*UDP* 123
SNTP	*Simple Network Time Protocol* - obezbeđuje slične funkcije kao i *NTP*, s tim što koristi jednostavniji algoritam i manje je precizan od *NTP* servera.	*UDP* 123
IMAP4	*Internet Message Access Protocol version 4* - preuzima *email* poruke sa *mail* servera.	*TCP* 143
LDAP	*Lightweight Directory Access Protocol* – obezbeđuje usluge direktorijuma u klaud okruženju, kao na primer korisnički direktorijum koji obuhvata informacije o korisničkom imenu, lozinci, *email* adresi i brojevima telefona.	*TCP* 389
HTTPS	*Hypertext Transfer Protocol Secure* – obezbeđuje sigurno preuzimanje sadržaja sa *veb* servera u klaudu.	*TCP* 443
RSH	*Remote Shell* - omogućava izvršavanje komandi na udaljenom klijentu sa klaud servera.	*TCP* 514
RTSP	*Real Time Streaming Protocol* – obezbeđuje kumunikaciju u realnom vremenu sa media serverom na klaudu.	*TCP* 554 i *UDP* 554
RDP	*Remote Desktop Protocol: - Microsoft*-ov vlasnički protokol koji omogućava korisnicima da imaju uvid i da kontrolišu *desktop* udaljene računare na mreži.	*TCP* 3389

7.3.2. Konfigurisanje klaud mreže

Bez obzira što je provajder klaud usluga vlasnik računarske mreže u svom računskom centru (svojim prostorijama), u mnogim implementacijama, nakon migriranja mreže na klaud, dobavljač dozvoljava i omogućava korporativnim korisnicima da konfigurišu sopstvene virtuelne privatne mreže, ili privatna klaud okruženja u zoni računarske mreže dobavljača usluga.

Navedene funkcionalnosti su korporativnom klijentu dostupne najčešće preko *Web* kontrolne table, grafičkog ili konzolnog interfejsa.

Podešavanja klaud mreže odnose se na sve mrežne servise, poput servisa za održavanje tabela rutiranja, kontrolnih pristupnih listi, sigurnosnih grupa i pravila korporativnog *firewall*-a, pomoću kojih se kontroliše i filtrira mrežni saobraćaj koji ulazi i izlazi iz klaud okruženja.

Osim navedenih, klaud provajder u većini slučajeva dozvoljava i konfigurisanje uređaja za balansiranje opterećenja, isporuku sadržaja, sistema za keširanje i *DNS* servisa.

7.3.3. Virtuelne privatne mreže

Virtuelna privatna mreža (eng. *Virtual Private Network – VPN*) obezbeđuje sigurnu i zaštićenu konekciju preko nesigurne komunikacione mreže kao što je Internet.

Virtuelne privatne mreže se pre svega koriste za pristup klaud servisima sa udaljenih lokacija (eng. *remote locations*). Primer ovakve upotrebe *VPN*-a u klaud računarstvu prikazan je na Slici 7-9.

Slika 7-9: *Pristup klaud servisima sa udaljene lokacije korišćenjem VPN servisa*

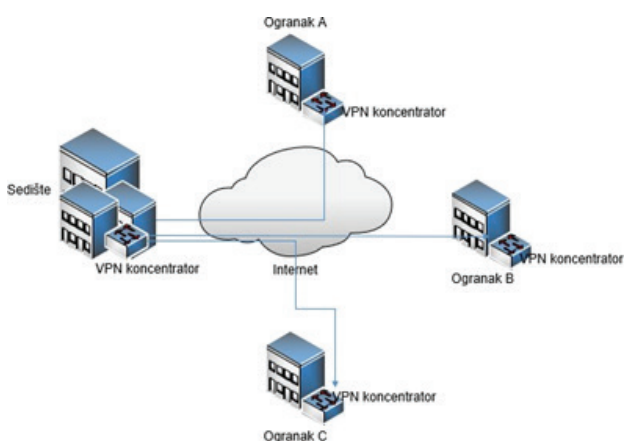


VPN mreže se takođe koriste kako bi se povezala klaud okruženja različitih kompanija (kao što je to slučaj sa zajedničkim klaudom) preko javne Internet mreže, a sa ciljem eliminisanja troškova koji bi nastali iznajmljivanjem namenskih komunikacionih linkova (eng. *dedicated communication links*). Servis *VPN* se takođe koristi za potrebe jedne korporacije koja hoće da poveže više svojih privatnih klaud okruženja u jednu mrežu.

VPN servisi mogu da se implementiraju na više načina. Jedan od načina je povezivanje klijentskog uređaja na VPN server koji je implementiran na *firewall*-u, ili ruteru klaud infrastrukture. Bolji način je povezivanje klijenta preko namenskog *VPN* koncentratora (eng. *VPN concentrator*), uređaja koji je namenjen isključivo za *VPN* upotrebu, za razliku od *firewall*-a i rutera, čije primarne uloge nisu *VPN* povezivanje.

Primer povezivanja nekoliko udaljenih privatnih klaud okruženja jedne kompanije preko *VPN* koncentratora dato je na Slici 7-10.

Slika 7-10: *Povezivanje privatnih klaud okruženja preko VPN koncentratora*



7.3.4. Sistemi za detekciju i sprečavanje upada u klaud sistem

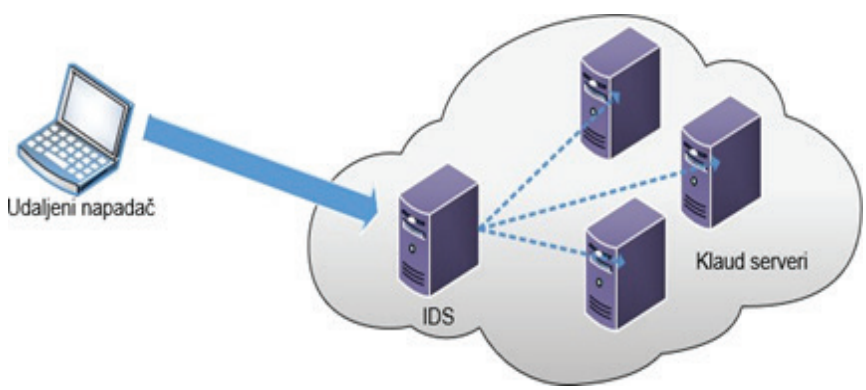
Sistemi za detekciju i sprečavanje upada u sistem su neizostavna komponenta svakog klaud okruženja. Oba sistema funkcionišu na sličan način tako što kontinuirano snimaju mrežni saobraćaj sa ciljem identifikovanja sumnjivih mrežnih aktivnosti.

Iako oba rešenja mogu da detektuju upade u sistem u realnom vremenu, postoji značajna razlika između njih. Sistemi za detekciju upada u sistem (eng. *Intrusion Detection System – IDS*) pasivno snimaju mrežni saobraćaj i traže potpise (eng. *signatures*) mrežnih aktivnosti koji predstavljaju indikator upada u sistem, na osnovu predefinisanih skupova pravila koja proizvođači ovih sistema neprekidno ažuriraju.

IDS sistemi podržavaju napredne funkcije obaveštavanja (eng. *notification functions*) i neposredno nakon detektovanja sumnjivih aktivnosti šalju obaveštenja mrežnim administratorima u formi *email*-a, *sms* poruke, ili na neki sličan način. Međutim, *IDS* sistemi ne sprovode akcije koje će da reše novonastalu situaciju, već samo posmatraju i analiziraju mrežni saobraćaj i generišu izveštaje.

IDS mrežni uređaji u klaud okruženju prikazani su na Slici 7-11.

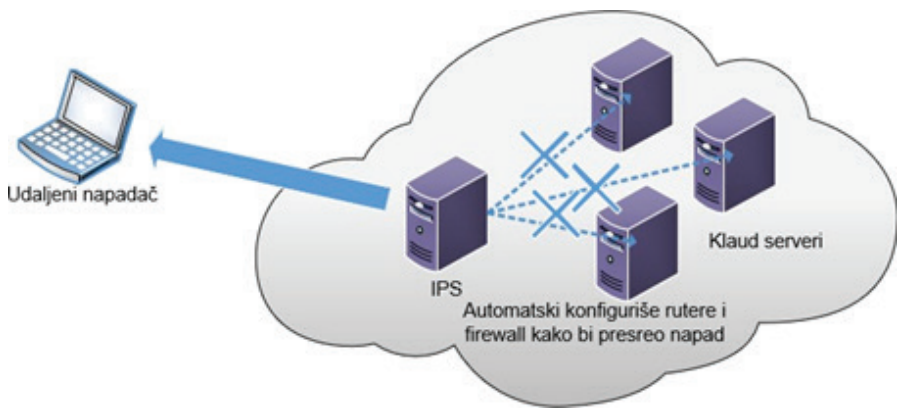
Slika 7-11: *Primer upotrebe IDS uređaja u klaud okruženju*



Sistemi za sprečavanje upada u sistem (eng. *Intrusion Prevention System - IPS*) idu korak dalje od *IDS* sistema, tako što aktivno prate mrežni saobraćaj i preduzimaju mere kako bi se smanjili rizici napada. *IPS* sistemi koriste prekonfigurisane skripte i heurističke metode za sprečavanje napada. Kada se napad dogodi, *IPS* sistemi komuniciraju sa drugim mrežnim komponentama, kao što su *firewall* i ruteri, kako bi se blokirali i minimizovali efekti napada.

Prikaz *IPS* sistema na klaud infrastrukturi dat je na Slici 7-12.

Slika 7-12: *Primer upotrebe IPS uređaja na klaudu*



7.3.5. Klaud demilitarizovane zone

Slično kao i u klasičnim mrežnim okruženjima, tako se i u klaudu često koriste demilitarizovne zone (eng. *demilitarized zone – DMZ*). U *DMZ*-u se nalaze računari i serveri kojima klijenti pristupaju izvan klaud okruženja preko nezaštićene računarske mreže, kao što je Internet.

Kao i obično, u literaturi može da se nađe više definicija *DMZ*-a. U nastavku su navedene samo neke od njih.

> *DMZ* zona je neutralna zona (otuda i sam naziv) između lokalne klaud računarske mreže i Interneta.

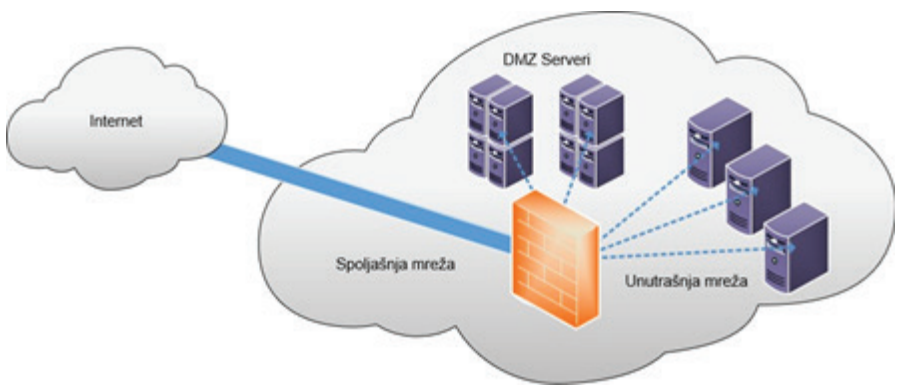
DMZ je sigurnosna zona u kojoj se nalaze računarski resursi klauda koji su izloženi Internetu.

Glavni cilj *DMZ*-a je onemogućavanje direktnog pristupa spoljnim (eksternim) korisnicima podacima u klaud okruženju koji se nalaze u lokalnoj klaud računarskoj mreži. *DMZ* se konfiguriše na način da klaud server koji se nalazi u *DMZ*-u ne može da zahteva računarske resurse koji se nalaze u lokalnoj klaud mreži. Zbog toga, ako napadač na primer uspe da ostvari pristup klaud serveru u *DMZ*-u, on neće uspeti da ostvari pristup i internoj klaud mreži i na taj način će korporativni podaci biti zaštićeni od uticaja napadača.

DMZ se najčešće koristi za implementaciju javno distupnih servisa, poput *Web, DNS*, ili *FTP* servisa.

Ilustracija *DMZ*-a na klaudu data je na Slici 7-13.

Slika 7-13: *Primer DMZ konfiguracija na klaudu*



7.3.6. *VxLAN* implementacija na klaudu

Virtuelna proširiva lokalna računarska mreža (eng. *Virtual Extensible Local Area Network – VxLAN*) je tip virtuelne lokalne mreže (eng. *Virtual Local Area Network – VLAN*), koji je kreiran sa ciljem prevazilaženja nedostataka klasičnih *VLAN* mreža.

Klasičan *VLAN* je dovoljan za zadovoljavanje potreba klasične računarske mreže, jer u teoriji omogućava čak do 4.094 virtuelnih mreža. U praksi je ovaj broj manji, zato što neki vendori rezervišu određene *VLAN* opsege za specijalne upotrebe, tako da je u praksi moguće uspostaviti oko 4.000 virtuelnih mreža.

Međutim, za dobavljača klaud usluga, broj od 4.000 virtuelnih mreža je daleko ispod zahtevanog broja za simultanim opsluživanjem nekoliko stotina hiljada klijenata. Kada bi dobavljač klaud usluga koristio tradicionalne *VLAN*-ove, onda bi skaliranje mrežnih resursa bilo vrlo skromno, čak ispod zadovoljavajućeg nivoa.

Osim navedenog, klasični *VLAN*-ovi nisu moblini, što znači da se virtuelne podmreže vezuju za hardverske uređaje na kojima su konfigurisane, kao što su ruteri. Ovakav način funkcionisanja ne bi mogao da zadovolji potrebe klaud korisnika.

S druge strane, *VxLAN* spada u grupu tehnologija virtuelizacije mreže koja uspešno rešava probleme skalabilnosti resursa u velikim klaud okruženjima. *VxLAN* je mobilan i menja način toka saobraćaja u mreži.

Tehnologija *VxLAN*, poput klasičnog *VLAN*-a, koristi tehniku enkapsulacije, tako što uzima *Ethernet* frejm drugog sloja *ISO/OSI* modela i enkapsulira ga u *IP* paket. Za transport koristi *UDP* protokol, dok za *default* odredišni port koristi port 4789. Ovaj broj porta je i formalno prihvaćen za potrebe *VxLAN*-a i naveden je u zvaničnoj specifikaciji agencije *IANA* (*Internet Assigned Numbers Authority*).

U literaturi se često za opisani način enkapsulacije usvaja i naziv *MAC* u *IP* enkapsulacija, zato što se neizmenjeni *Ethernet* frejm obmotava (enkapsulira) u klasičan *IP/UDP* paket.

Zaglavlje *VxLAN* paketa je dugačko 8 bajtova (64 bitova). Identifikator *VxLAN* mreže (eng. *VxLAN Network Identifier – VNI*) koristi 24 bitova zaglavlja. Na ovaj način, *VNI* može da skalira u više od 16 miliona mrežnih segmenata, što je značajno više od 4.096, koje obezbeđuje klasična *VLAN* tehnologija.

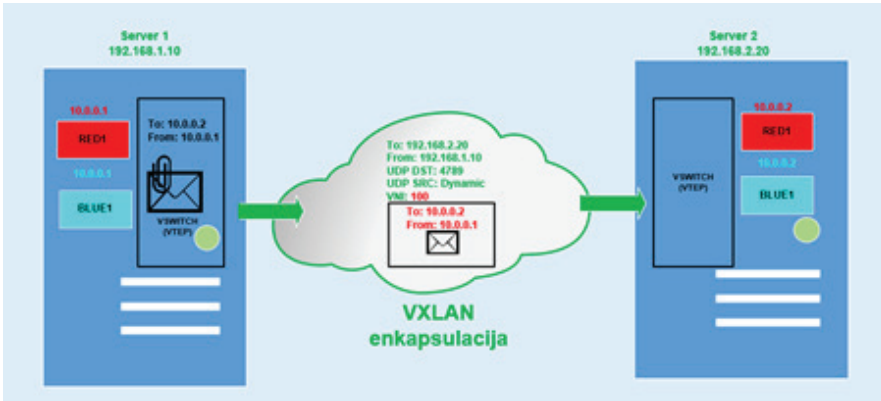
Mobilnost je omogućena primenom koncepta *VxLAN* kranje tačke tunela (eng. *VxLAN tunnel endpoint – VTEP*). *VTEP* može da bude bilo koji mrežni uređaj koji ima mogućnost obrade *VxLAN* paketa.

Kada *VTEP* uređaj primi paket koji je namenjen lokalnom računaru, ovaj uređaj vrši deenkapsulaciju paketa i lokalnom računaru prosleđuje samo originalni frejm. Na taj način, ceo proces se apstrahuje i lokalni računar „vidi“ samo običan frejm.

Proces *VxLAN* enkapsulacije je prikazan na Slici 7-14.

Zbog potrebe opsluživanja većeg broja klijenata, u klaud okruženjima, *VLAN* mreže su uglavnom zamenjene sa *VxLAN* tehnologijom.

Slika 7-14: *Proces VxLAN enkapsulacije*



7.3.7. Adresiranje klaud mreže

> Adresiranje mreže odnosi se na segmentiranje *TCP/IP* mreža na podmreže uzimajući u obzir trenutne i buduće zahteve u vidu broja mrežnih uređaja.

Prilikom migriranja računarske mreže na klaud okruženje, jedan od prioritetnih zadataka je kreiranje kvalitetnog plana adresiranja klaud mreže, zato što će na taj način trenutne i buduće potrebe za klaud mrežnim resursima biti najbolje zadovoljene.

Svaki provajder klaud usluga poseduje blok javno dostupnih *IP* adresa kojima može da se pristupi preko Interneta. Kompanija koja migrira svoje resurse na klaud dobiće ograničeni broj javnih *IP* adresa od svog dobavljača klaud usluge.

S obzirom da će broj raspoloživih javnih *IP* adresa koja će kompanija od klaud provajdera dobiti najverovatnije da bude manji od broja korporativnih mrežnih uređaja koji će biti migrirani na klaud, za *IP* adresiranje korporativnih uređaja u klaudu biće korišćeni blokovi privatnih, nerutabilnih *IP* adresa koji su definisani u *RFC* (*Request for Comments*) dokumentu pod brojem 1918.

Kao napomena se navodi da se isti blokovi privatnih *IP* adresa koji se koriste u lokalnom mrežnom okruženju koriste i u klaud infrastrukturi.

S obzirom da je predpostavka da čitalac ima osnovno znanje o mrežnim tehnologijama, pojedinosti poput maski podmreže, broj host računara na mreži, itd. ovde nisu navedene.

Blokovi dostupnih privatnih IP adresa prema dokumentu *RFC1918* prikazani su u Tabeli 7-2.

Tabela 7-2: *Blokovi privatnih IP adresa*

RFC 1918 naziv bloka	Opseg IP adresa	Broj raspoloživih IP adresa
blok dužine 24 bita	10.0.0.0 – 10.255.255.255	16,777,216
blok dužine 20 bitova	172.16.0.0 – 172.31.255.255	1,048,576
blok dužine 16 bita	192.168.0.0 – 192.168.255.255	65,536

Upotrebom privatnih *IP* adresa čuvaju se ograničene javne *IP* adrese i koriste se za adresiranje uređaja koji ne zahtevaju neposredan pristup Internetu.

U zavisnosti od politike dobavljača klaud usluga, kompanija će možda dobiti mogućnost da sama izabere blok javnih *IP* adresa, ili će im blok dodeliti klaud provajder.

7.3.8. Segmentacija bloka adresa

U većini slučajeva klaud provajder korporativnim klijentima dodeljuje jedan veći blok raspoloživih *IP* adresa, koje zatim kompanije same dele u podmreže u zavisnosti od svojih potreba.

Koncept podmreže je dobar zato što omogućava grupisanje sličnih aplikacija, ili mrežnih segmenata u posebne logičke celine. Na ovaj način sigurnost i bezbednost same računarske mreže značajno može da se unapredi, zbog toga što se kontrolne pristupne liste, sigurnosne grupe i *firewall*-ovi u ovom slučaju podešavaju na nivou podmreže, a ne na nivou cele mreže.

Različite aplikacije koje su grupisane u različite podmreže mogu da imaju posebne sigurnosne i druge zahteve, i na ovaj način mogu da se sprovode granularna (fina) podešavanja, do nivoa samih aplikacija i servisa.

7.4. BENČMARK TESTOVI

Kada korporacija migrira svoje resurse na klaud okruženje, važno je da se utvrde performance klaud objekata, kako bi se osiguralo da će svi klaud objekti moći da zadovolje sve korporativne potrebe za *IT* resursima. Performanse klaud objekata utvrđuju se benčmark testovima (eng. *benchmark tests*).

> Benčmark je proces u kojem se koriste uzorci metrika performansi, koji se prikupljaju kao deo procesa kreiranja dokumentacije za migraciju na klaud. Kroz benčmark testove vrši se dokumentovanje performansi korišćenja klaud objekata, kao što su procesorsko vreme, korisni kapacitet skladišnog prostora, *I/O* performanse baza podataka i propusnost mreže.

U klaud okruženju podržano je merenje performansi skoro svakog objekta, gde je nemoguće i nepraktično da se prate svi objekti. Zbog toga, organizacija treba pažljivo da izabere skup objekata za koje je potrebno meriti performanse.

Skup objekata koji se prate pre svega treba da uključi resurse koji su od suštinske važnosti za izvršavanje klaud servisa (na primer, u slučaju kompanije koja implementira klaud portal za elektronsko plaćanje, obavezno treba pratiti performance *Web* servera na kojem je ovaj portal konfigurisan).

Prvo merenje performansi nakon migracije na klaud je i najvažnije, jer se na taj način definiše osnova za kasnije upoređivanje (eng. *baseline*). Takođe, korisno je da se rezultati prvog merenja uporede sa rezultatima merenja performansi koja su sprovedena u lokalnom okruženju pre migracije, kako bi se napravila komparativna analiza klaud performansi sa performansama lokalnog sistema.

Kasnije, tokom eksploatacije klaud okruženja, s vremena na vreme je potrebno vršiti i dodatna merenja. Rezultati dodatnih merenja se zatim upoređuju sa osnovnim (prvim) merenjem i na taj način se lakše detektuju potencijalni problemi koji mogu uslediti nakon migracije, ili tokom eksploatacije klaud resursa.

7.5. SPORAZUM O NIVOU KORIŠĆENJA KLAUD USLUGA

Pre migracije na klaud okruženje, kompanija klijent sa klaud provajderom potpisuje Sporazum o nivou korišćenja usluga (eng. *Service Level Agreement – SLA*).

Dobavljač klaud usluga može da ponudi i usluge upravljanja infrastrukturom, kao nadogradnju svoje osnovne ponude, što se sve definiše u Sporazumu o nivou korišćenja usluga. Ovaj ugovor dokumentuje i naglašava korišćene metrike i minimalne nivoe performansi i dostupost resursa, gde dobavljač usluga plaća penale, ako kvalitet pruženih usluga nije u okviru zadovoljavajućeg raspona vrednosti koje su definisane u Sporazumu o nivou korišćenja usluga.

Osim navedenog, u Sporazumu o nivou korišćenja usluga definiše se i ko poseduje podatke, i ko ima koja prava i odgovornosti. U svakom ovakvom dokumentu precizno su definisani i uslovi za raskid sporazuma.

Bez obzira na postavku ugovora, korporativni klijent mora da prihvati odgovornosti i obaveze za upravljanje klaud mrežom i aplikacijama.

7.6. MIGRACIJA NA KLAUD I UŠTEDE U TROŠKOVIMA I POTROŠENOJ ENERGIJI

Migracija korporativnih računarskih resursa na klaud donosi brojne uštede u skoro svim kategorijama troškova.

Glavna troškovna prednost klaud računarstva je model plaćanja na osnovu nivoa potrošenih računarskih resursa. Kompanija više nema visoke inicijalne troškove nabavke skupih servera, skladišta podataka, mrežne opreme, itd. Organizacija može da uloži novac koji bi uložila u sofisticiranu i skupu računarsku opremu, koja relativno brzo zastareva, na nekom drugom mestu, što predstavlja još jednu indirektnu korist od korišćenja klaud modela organizovanja računarskih resursa.

Takođe, kada koristi klaud tehnologije, organizacija ne mora da misli na dodavanje opreme koja bi zadovoljila privremene i povremene skokove u potrebama za računarskim resursima zbog elastične i skalabilne prirode klaud računarstva, zato što je potrebno svega nekoliko minuta da se implementira i konfiguriše novo „parče" hardvera zahvaljujući tehnologiji virtuelizacije. S druge strane, kada dodatna oprema više nije potrebna, ona se relativno lako može vratiti klaud provajderu.

Međutim, bez obzira na izraženu troškovnu efikasnost klaud računarstva, klaud resursi zahtevaju da se njima efektivno i efikasno upravlja, jer postoje brojni modeli plaćanja za korišćenje klaud računarskih resursa. Tako na primer, kada organizacija ne koristi svoj klaud server, server treba da se isključi, jer bi se u suprotnom bespotrebno generisao trošak korišćenja opreme, zato što klaud provajder najčešće naplaćuje po radnom satu svoje opreme.

Većina računskih centara klaud provajdera nije starije više od 10 godina i zbog toga ovi sistemi koriste najsavremenije tehnologije i procedure za uštedu električne energije. Takođe, mnogi klaud računski centri se fizički nalaze u geografskim oblastima gde je niža cena električne energije, ili u blizini resursa sa hladnom vodom koja se koristi u svrhu hlađenja računarske opreme. Ovo predstavlja još jednu dodatnu uštedu, ako se kompanija odluči da iznajmi ceo računski centar u okruženju klaud provajdera.

Što je klaud računski centar energetski efikasniji, to su niži operativni troškovi. Na ovaj način se operativne margine povećavaju i smanjuju se cene usluga ka korisnicima. Izvori ušteda u potrošnji električne energije je korišćenje zajedničkog hardvera, gde je hardver optimalno iskorišćen. S druge strane, u tradicionalnim okruženjima često se dešava da server ništa ne radi i po nekoliko sati, čak i danima, i da za to vreme troši električnu energiju. Moderni sistemi za upravljanje klaud infrastrukturom automatski gase servere i drugu mrežnu opremu kada se oprema ne koristi.

Osim što se koristi zajednički hardver, u klaud okruženju se koristi i virtuelizovani hardver, kao što su virtuelni mrežni uređaji, skladišta, server, itd., što predstavlja još jedan dodatni izvor uštede. Međutim, zbog sigurnosnih i drugih razloga, za kompaniju ponekada namenski server predstavlja jedinu opciju. U tom slučaju klaud provajder naplaćuje više, jer server koristi samo jedan klijent.



PITANJA ZA RAZMATRANJE

1. Koji je najvažniji korak u planiranju procesa migracije na klaud? Objasniti.
2. Navesti korake u planiranju migracije na klaud.
3. Šta se podrazumeva pod pojmom upravljanje promenama kada je reč o izvršavanju migracije na klaud? Objasniti.
4. Navesti neophodne elemente za informisanje svih zainteresovanih strana o promenama i migraciji na klaud okruženje. Objasniti.
5. Koji kriterijumi se sagledavaju za definisanje realnog vremenskog okvira za sprovođenje migracije na klaud okruženje? Objasniti.
6. Kakva dokumentacija treba da se pripremi i koje procedure treba da se sprovedu da bi se izvršio proces migracije na klaud okruženje? Objasniti.
7. Šta je klaud radni tok? Objasniti.
8. Šta je klaud automatizacija? Objasniti.
9. Navesti i obrazložiti šta su klaud alati i sistemi za upravljanje.
10. Objasniti značenje akronima *FCAPS*.
11. Navesti klaud mrežne protokole. Objasniti kroz primer.
12. Šta je virtuelna privatna mreža? Dati prmer.
13. Koji je zadatak sistema za detekciju i sprečavanje upada na klaud sistem? Objasniti kroz primere.
14. Objasniti pojam demilitarizovane zone u klaud okruženju.
15. Objasniti proces *VxLAN* enkapsulacije.

8. USAGLAŠAVANJE FIZIČKIH RESURSA SA VIRTUELNIM RESURSIMA NA KLAUDU

Prilikom migracije na virtuelno klaud okruženje sa tradicionalnog, nevirtuelnog okruženja, potrebno je izvršiti detaljnu procenu klaud okruženja, kako bi se obezbedilo da virtuelizovani serveri imaju sva prava pristupa i da su skalirani u skladu sa zahtevima. Kako bi se ovo postiglo, potrebno je izvršiti preciznu evaluaciju hardverskih zahteva za svaki operativni sistem i u skladu sa tim skalirati instancu virtuelne mašine na klaudu kako bi svi zahtevi bili zadovoljeni.

S obzirom da je većina klaud resursa virtuelizovana, potrebno je mapirati fizičke resurse na klaud virtuelne resurse.

U ovoj glavi prikazano je evaluiranje i skaliranje računarskih resursa na klaudu, kako bi oni zadovoljili sve potrebe kao i tradicionalni, fizički računski centri.

8.1. RASPOLOŽIVI HARDVERSKI RESURSI U KLAUD OKRUŽENJU

Provajderi klaud usluga u svojoj ponudi najčešće uključuju široki spektar virtuelnih mašina različitih konfiguracija koje zadovoljavaju najrazličitije zahteve, kao što su zahtevi za klasičnom *office* računarskom platformom, platformom za obradu slika sa jakim grafičkim procesorom (eng. *Graphical Processing Unit – GPU*), platformama koje su namenjene za izvršavanje naprednih algoritama sa jakim procesorima i platformama na kojima se izvršavaju aplikacije baza podataka kojima simultano pristupa više korisnika i koje poseduju sofisticirane resurse za brzo izvršavanje svih *I/O* operacija.

U klaud terminologiji, prekonfigurisana virtuelna platforma (mašina) naziva se instanca (eng. *instance*). Mnogi klaud provajderi nude dva tipa instanci, i to one koje imaju procesore sa visokim performansama i one koje imaju veliku količinu *RAM* memorije.

Prekonfigurisane instance virtuelnih mašina su spremne za upotrebu po principu „uključi i koristi" (eng. *Plug and Play – PnP*). Na ovaj način korporativni klijent u roku od samo nekoliko minuta može da pokrene novi servis koji je implementiran na virtuelnoj platformi.

Tipovi instanci se razlikuju od jednog do drugog provajdera klaud usluga, gde se instance razlikuju na osnovu broja raspoloživih virtuelnih procesora, količine *RAM* memorije, performansi ulazno/izlaznih uređaja, kao i mnogih drugih parametara.

Radi potpunijeg razumevanja virtuelnog hardvera na klaudu, potrebno je da se razume kako se fizički hardverski resursi alociraju virtuelnim mašinama koje se izvršavaju na klaud serverima.

8.1.1. Fizički i virtuelni procesori

Snaga procesora je značajno porasla sa razvojem višejezgarnih procesora (eng. *multi-core CPUs*). Na tipičnom fizičkom serveru najčešće se izvršava veći broj virtuelnih mašina, i zbog toga procesor fizičkog servera mora da bude dovoljno jak, kako bi zadovoljio potrebe svih virtuelnih mašina za resursima fizičkog hardvera.

Svakoj virtuelnoj mašini dodeljuje se virtuelni procesor, koji troši procesorsko vreme fizičkog procesora. Matične ploče savremenih servera najčešće imaju slotove za dodavanje dodatnih višejezgarnih procesora. Na taj način resursi fizičkog servera mogu da se skaliraju kako bi se omogućilo izvršavanje više virtuelnih mašina.

Konvergirane i hiper-konvergirane infrastrukture, o kojima je bilo reči u Glavi 2, podržavaju mogućnost skaliranja skoro svih resursa (procesora, memorije, mrežnih kartica, itd.).

8.1.2. Fizička i virtuelna memorija

Virtuelne mašine koriste i troše RAM memoriju fizičkog servera na kome se izvršavaju. Zahtevi u pogledu *RAM* memorije zavise od broja i konfiguracije virtuelnih mašina (koliko virtuelne *RAM* memorije se dodeljuje svakoj virtuelnoj mašini).

Provajder klaud usluga mora da implementira veliku količinu *RAM* memorije na svakom fizičkom serveru, kako bi zadovoljio budući rast u pogledu broja hostovanih virtuelnih mašina i kako bi obezbedio dovoljno memorije za rad hipervizora.

Osim veličine memorije potrebno je uzeti u obzir i druge parametre, kao što su brzina pristupa memoriji i memorijske funkcije za korekciju grešaka.

8.1.3. Upravljanje virtuelnom memorijom

Svaki noviji hipervizor podržava funkcionalnost tzv. naduvavanja memorije (eng. *memory ballooning*).

Ova funkcija omogućava hipervizoru da preuzme trenutno neiskorišćenu memoriju od virtuelne mašine koja se izvršava iznad njega. Hipervizor zatim preuzetu memoriju može da iskoristi za druge potrebe, kao na primer može da je dodeli nekoj drugoj virtuelnoj mašini, ili da je iskoristi za svoj rad i da efikasnije upravlja virtuelnim mašinama. Na ovaj način hipervizor može da optimizuje upotrebu *RAM* memorije koja je instalirana na fizičkom serveru.

Funkcionalnost naduvavanja memorije je moguća zato što se hipervizor nalazi na sloju između virtuelne mašine i fizičkog hardvera. Kao što je već i pomenuto u Glavi 3, operativni sistem virtuelne mašine „misli" da se ispod njega nalazi pravi hardver i da sa njim komunicira. Ovu percepciju koristi hipervizor, koji „manipuliše" virtuelnim mašinama tako što određuje kojim hardverskim resursima koja virtuelna mašina pristupa i može da balansira korišćenje resursa između različitih virtuelnih mašina koje se izvršavaju iznad njega.

Tako na primer, hipervizor može da dodeli virtuelnoj mašini 8GB virtuelne *RAM* memorije, kada virtuelna mašina „misli" da 8GB *RAM* memorije postoji na fizičkom hardveru. Hipervizor zatim može da primeni koncept naduvavanja memorije i da oduzme 4GB memorije od virtuelne mašine. Kako bi nadomestila razliku u količini memorije, virtuelna mašina bi u tom slučaju počela da koristi fajl za straničenje (eng. *paging file*), kao proširenje svoje virtuelne memorije.

8.1.4. Prekoračenje memorije

Hipervizori u virtuelizovanom klaud okruženju podržavaju i funkcionalnost prekoračenja memorije (eng. *memory overcommitting*).

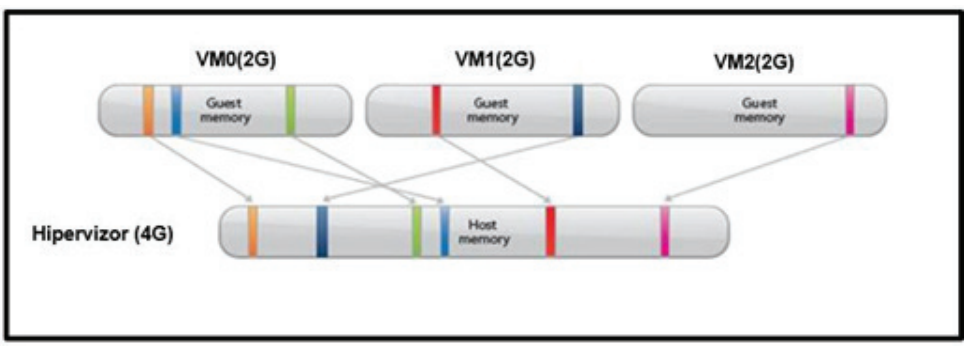
Funkcija prekoračenja memorije omogućava virtuelnim mašinama koje se izvršavaju iznad hipervizora da koriste veću količinu *RAM* memorije od one koja je instalirana na fizičkom serveru.

Za razumevanja koncepta prekoračenja memorije može da posluži sledeći primer. Neka se na hipervizoru izvršava 32 virtuelne mašine, gde je svakoj mašini dodeljeno 8GB virtuelne *RAM* memorije. S druge strane, na fizičkom serveru je instalirano samo 128GB *RAM* memorije. U ovom slučaju, primenom koncepta prekoračenja memorije, memorija fizičkog servera je prekoračena 2 puta.

Koncept prekoračenja memorije sa temelji na predpostavci da neće sve virtuelne mašine u isto vreme da koriste svu memoriju koja im je dodeljena. Neiskorišćena memorija se dinamički dodeljuje virtuelnim mašinama kojima je u tom vremenskom trenutku potrebna. Na ovaj način, iskorišćenost memorije fizičkog servera se u značajnoj meri optimizuje.

Koncept prekoračenja memorije ilustrovan je na Slici 8-1.

Slika 8-1: *Koncept prekoračenja memorije*



8.1.5. Tehnologija višenitnog izvršavanja

Tehnologija višenitnog izvršavanja (eng. *hyperthreading*) u velikoj meri unapređuje performanse i iskorišćenost fizičkog procesora i podržana je na nivou arhitekture procesora.

Tehnologija višenitnog izvršavanja omogućava da se dve ili više niti (eng. *threads*) pseudo-paralelno izvršavaju na jednom jezgru fizičkog procesora. Operativni sistem na ovaj način jedno procesorsko jezgro vidi kao dva jezgra.

Tako na primer, ako fizički procesor ima četiri jezgra i podršku za tehnologiju višenitnog izvršavanja, operativni sistem će „videti" sistem sa 8 logičkih (virtuelnih) procesora. Svaki logički ili virtuelni procesor može da se pokrene, zaustavi i da se kontroliše nezavisno od ostalih logičkih procesora.

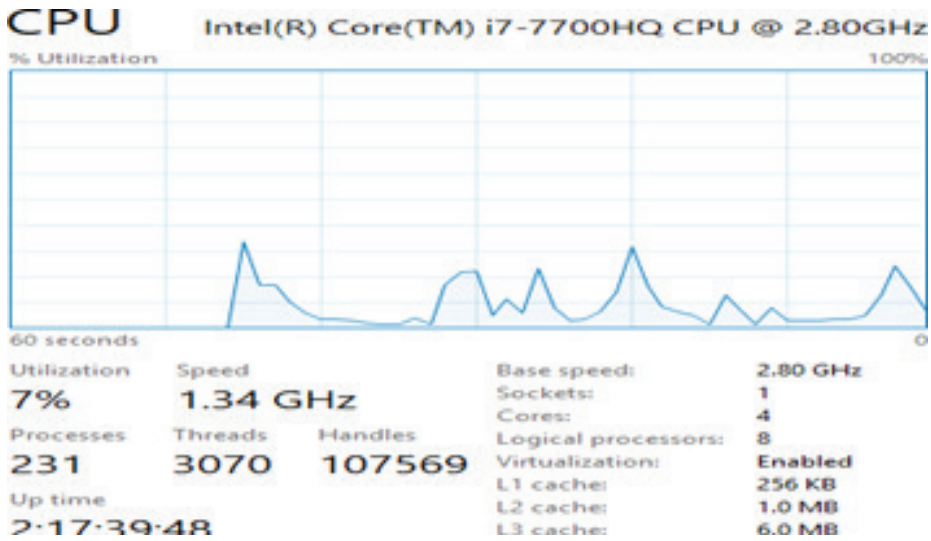
Screenshot Windows komponente za merenje performansi jednog takvog sistema dat je na Slici 8-2.

Tehnologiju višenitnog izvršavanja razvila je kompanija Intel pod nazivom *Intel® Hyper-Threading Technology* (*Intel® HT Technology*).

Kao napomena se navodi da i hipervizor i operativni sistem moraju da podržavaju simetrično multiprocesiranje (eng. *Symmetrical Multiprocessing - SMP*) kako bi prednosti tehnologije višenitnog izvršavanja mogle da budu iskorišćene.

Procesori fizičkih servera koji se implementiraju u klaud okruženjima uglavnom podržavaju tehnologiju višenitnog izvršavanja. Primenom ove tehnologije klaud provajder i na nivou fizičkog hardvera može da ostvari uštede i optimizacije.

Slika 8-2: *Screenshot - Windows servisa za merenje performansi sistema za i7 procesor sedme generacije koji podržava višenitno izvršavanje*



8.1.6. Optimizacija iskorišćenosti procesora primenom *Intel-VT* i *AMD-V* tehnologija

Kao što je već bilo reči u Glavi 3 , u početku je korišćena softverska virtuelizacija kako bi se poboljšale performanse procesora nad kojim se izvršava hipervizor virtuelizovanih servera. S obzirom da su performanse virtuelnih mašina u slučaju softverske virtuelizacije bile relativno skromne, kompanije *Intel* i *AMD* su dodali mikrokod u svoje procesore koji podržava virtuelizaciju i na taj način su se performanse izvršavanja virtuelnih mašina drastično poboljšale.

Intel-VT (*Intel Virtualization Technology*) je implementacija mikrokoda za podršku tehnologije virtuelizacije kompanije *Intel* koji se ugrađuje u *Intel*-ove procesore. S druge strane, *AMD-V* (*Advanced Micro Devices – Virtualization*) je marketinški naziv za isti mikrokod koji podržava virtuelizaciju i koji se ugrađuje u sve procesore kompanije *AMD*.

Na isti način kao što klaud hipervizori podržavaju prekoračenje *RAM* memorije, tako je podržano i prekoračenje procesora. Stopa prekoračenja procesora (eng. *CPU overcommitment ratio*) se u literaturi naziva odnos virtuelnog procesora (eng. *vCPU*) i fizičkog procesora (eng. *pCPU*).

Stopa prekoračenja procesora u velikoj meri zavisi od aplikacija koje se izvršavaju na virtuelnoj mašini. Ako ove aplikacije zahtevaju malo procesorske snage, onda stopa prekoračenja ima malu vrednost, i obrnuto.

Primenom koncepta prekoračenja fizičkih resursa u virtuelnim klaud servisima, stopa iskorišćenosti fizičkih resursa se maksimizuje, što dovodi do nižih troškova. Koncept prekoračenja procesora temelji se na pretpostavci da neće svi virtuelni serveri u svakom vremenskom trenutku da koriste sve raspoložive hardverske resurse i na taj način se vrši dinamička alokacija ciklusa *CPU* vremena fizičkog procesora virtuelnim mašina kojima su ovi ciklusi potrebni.

Vreme čekanja procesora (eng. *CPU wait time*) je vreme koje je potrebno da proces ili nit čeka, kako bi dobila procesorsko vreme. U virtuelizovanom okruženju, uz postojanje hipervizora, virtuelna mašina mora da čeka dok ograničeni fizički resursi procesora ne postanu dostupni. Hipervizor svakoj virtuelnoj mašini prikazuje nivo resursa virtuelnog procesora koji je trenutno raspoloživ, a koji zavisi od resursa fizičkog procesora.

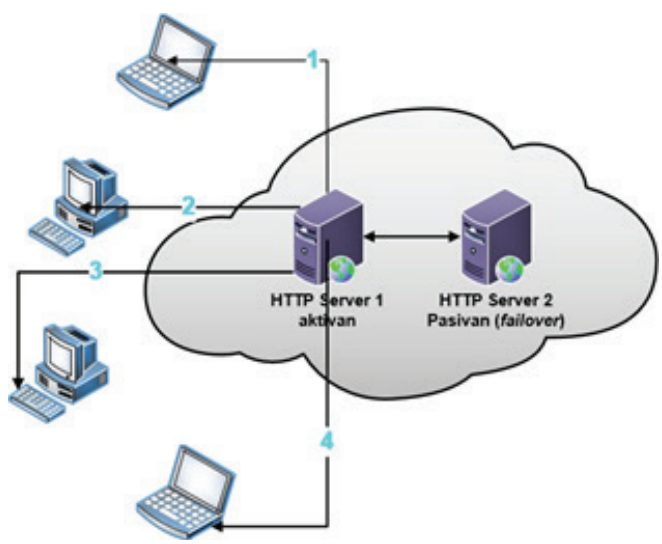
8.2. VISOK NIVO DOSTUPNOSTI KLAUD RESURSA I OPORAVAK OD KATASTROFE

Klaud provajderi su u obavezi da obezbede najbolje usluge svojim klijentima. Zbog toga se računski centri u klaud okruženju implementiraju uz zadovoljavanje cilja visokog nivoa dostupnosti klaud resursa (eng. *high availability - HA*) i obezbeđivanjem mehanizama za oporavak od katastrofe (eng. *disaster recovery - DR*).

Cij visokog nivoa dostupnosti klaud resursa najčešće se ostvaruje upotrebom redudantnih sistema koji koriste dva radna režima, i to u aktivan/aktivan (eng. *active/active*), ili režim aktivan/u pripravnosti (eng. *active/standby*). U režimu aktivan/aktivan, oba sistema su aktivna u svakom vremenskom trenutku, dok je u režimu aktivan/u stanju pripravnosti, u svakom vremenskom trenutku samo jedan sistem operativan, dok je drugi sistem u stanju pripravnosti. U drugom radnom modu, ako glavni (eng. *master*) sistem otkaže, drugi sistem preuzima poslove.

Primer radnog režima aktivan/pasivan u slučaju klaud *Web* servera dat je na Slici 8-3 .

Slika 8-3: *Radni režim aktivan/pasivan Web server*



Kada je cilj visoke dostupnosti klaud resursa ostvaren, kritični klaud sistemi i servisi (eng. *mission-critical services and systems*), postaju pouzdaniji i otporniji, jer više nemaju jednu tačku prestanka rada sistema (eng. *single-point of failure – SPOF*). Implementiranje ovog mehanizma je od suštinske važnosti za klaud provajdere, zato što se na taj način zadovoljavaju sve klauzule definisane u Sporazumu o nivou korišćenja usluga i klijent je zadovoljan.

Osim ostvarivanja cilja visoke dostupnosti klaud resursa, klaud provajder treba da obezbedi i mehanizme za oporavak od katastrofa. Klaud provajder na raspolaganju ima brojne modele, tehnologije i arhitekture za implementiranje ovog mehanizma.

Neki od popularnijih načina za implementiranje mehanizama oporavka od katastrofe u klaud okruženju su vrući i hladni sajtovi, regioni i zone dostupnosti (eng. *availability zone*). O ovim i sličnim tehnologijama i mehanizmima dato je više informacija u Glavi 12.

S obzirom da se na klaudu uglavnom koriste virtuelni resursi, kada se vrši alokacija snage fizičkih računarskih resursa na virtuelne, potrebno je uzeti u obzir i resurse koji su potrebni za ostvarivanje cilja visoke dostupnosti klaud resursa i implementacije mehanizama za oporavak od katastrofe.



PITANJA ZA RAZMATRANJE

1. Objasniti pojam fukcionalnosti tzv. naduvavanja memorije.
2. Objasniti koncept prekoračenja memorije.
3. Šta je tehnologija višenitnog izvršavanja? Objasniti.
4. Kako se nazivaju dve tehnologije za virtuelizaciju koje su podržane na nivou hardverskih instrukcija procesora kompanija *Intel* i *AMD*.

9. KLAUD SKLADIŠTA PODATAKA

U ovoj glavi udžbenika prikazane su osnovne tehnologije klaud skladišta podataka. Pretpostavka je da čitaoci poseduju osnovno znanje o tehnologijama za skladištenje podataka u lokalnom računarskom okruženju.

S obzirom da se klaud tehnologije za skladištenje podataka neznatno razlikuju u odnosu na tehnologije skladišta koja se koriste u lokalnom računarskom okruženju, u ovoj glavi je napravljen samo kraći ostvrt na primenu, kao i na faktore koje treba uzeti u razmatranje, prilikom implementiranja ovih tehnologija u klaudu.

U klaud okruženju se retko koriste serveri koji imaju svoje sisteme za skladištenje podataka. Većina servera koristi eksterne centralizovane sisteme za skladištenje podataka u vidu namenske računarske mreže skladišta (eng. *Storage Area Network – SAN*).

Arhitektura *SAN*-a je ključna za performanse servera i aplikacija. Ako dolazi do usporavanja, bilo u čitanju, bilo u pisanju podataka u skladište, performance celog sistema će biti degradirane. Zbog toga klaud provajderi koriste *SAN* sisteme koji su posebno dizajnirani za korporativnu upotrebu sa *SSD* (*Solid-State Disk*) tehnologijom, ili mehaničkim diskovima koji imaju veliku brzinu obrtanja (eng. *rotates per minute - rpm*), unapređenim *I/O* konekcijama sa povećanim propusnim opsegom ka kontroleru skladišta i *SAN* interfejsom koji je dovoljno brz i može da obradi mrežne pakete i u uslovima pojačanog intenziteta mrežnog saobraćaja.

Kada je skladište podataka fizički udaljeno od servera koji se *hostuje* na virtuelnoj mašini, fleksibilnost virtuelnih mašina je veća, zato što mašine mogu da se migriraju sa jednog hipervizora na drugi hipervizor, koji se nalazi na drugom serveru u okviru klaud računskog centra. U procesu migriranja virtuelnih mašina, tekuće stanje mašina se pamti, i aplikacije koje su migrirane zajedno sa virtuelnim mašinama nastavljaju da funkcionišu na isti način kao da su ostale na istom serveru.

Sve navedeno omogućava tehnologija centralizovanog skladišta podataka, koja u odnosu na lokalno skladište (skladište koje je direktno povezano za server), unapređuje performanse, obezbeđuje implementiranje mehanizama otpornosti na greške (eng. *fault tolerance - FT*), olakšava održavanje i koristi mehanizam za oporavak od grešaka.

9.1. TIPOVI KLAUD SKLADIŠTA PODATAKA

Skladište podataka predstavlja jednu od ključnih komponenti svake klaud infrastrukture.

Postoji veliki broj vrsta skladišta podataka koja se koriste na klaudu i u ovom poglavlju će biti više reči o nekima od njih.

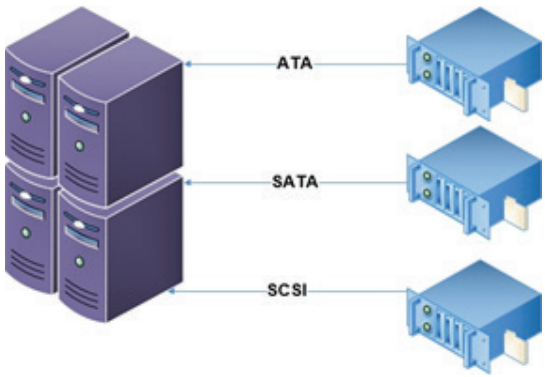
9.1.1. *DAS* skladište podataka

Skladište podataka koje se direktno priključuje na uređaj (eng. *Direct-Attached Storage – DAS*), bilo da je u pitanju klijentski, ili serverski računar, ili neki drugi uređaj, predstavlja jedan od najčešće korišćenih i najlakših načina dodavanja skladišta podataka u smislu implementacije. Ova skladišta mogu da sastoje od hard diskova, diskova koji se baziraju na *flash* memoriji, ili bilo kojeg drugog medijuma.

DAS skladište se direktno povezuje na uređaj, bez korišćenja računarske mreže. Za povezivanje se najčešće koriste interfejsi poput *ATA* (*Advanced Technology Attachments*), *SATA* (*Serial Advanced Technology Attachments*) ili *SCSI* (*Small Computer System Interface*) koji medijum za skladištenje podataka povezuju direktno na matičnu ploču računara, ili nekog drugog uređaja.

Ilustracija *DAS* skladišta data je na Slici 9-1.

Slika 9-1: *DAS skladište podataka*



9.1.2. *NAS* skladište podataka

Skladište podataka koje se povezuje preko računarske mreže (eng. *Network Attached Storage – NAS*) obezbeđuje pristup podacima na nivou fajlova putem računarske mreže.

Kao jedan od boljih primera *NAS* skladišta navodi se fajl server koji hostuje deljene direktorijume na *Ethernet LAN* klaud mreži.

U implementaciji *NAS* skladišta, podaci se preko računarske mreže ne šalju u formi blokova, već se šalju fajlovi. Podaci se ne skladište lokalno na računaru, već se čuvaju na udaljenoj lokaciji i pristupa im se putem lokalne klaud računarske mreže.

Klaud *NAS* skladište podataka najčešće se isporučuje u formi uređaja koji nije potrebno dodatno konfigurisati, već je dovoljno samo da se priključe na računarsku mrežu. Većina klaud *NAS* uređaja podržavaju povezivanje sistema sa heterogenim klijentskim uređajima i operativnim sistemima.

Kao neke od prednosti *NAS* skladišta podataka navode se sledeće:

- mogućnost deljenja podataka i integracija *NAS* sistema,
- zaštita i replikovanje podataka,
- mogućnost integracije sa eksternim aplikacijama, kao što su aplikacije baze podataka i aplikacije *email* servera i
- balans podataka.

Koncept *NAS* skladišta prikazan je na Slici 9-2.

Slika 9-2: *NAS skladište podataka*



9.1.3. *SAN* skladište podataka

U velikim računarskim okruženjima, kao što je klaud, skladišni sistemi su fizički odvojeni od servera i najčešće se nalaze u svojim sopstvenim rekovima u računskom centru. U klaud okruženjima se uglavnom koriste tzv. grupe skladišta (eng. storage arrays) koje se povezuju u namensku računarsku mrežu. Ova mreža se koristi isključivo za potrebe mrežnog saobraćaja između klijenata i mrežnih skladišta i izolovana je od drugih lokalnih *Ethernet* mreža u klaud računskom centru.

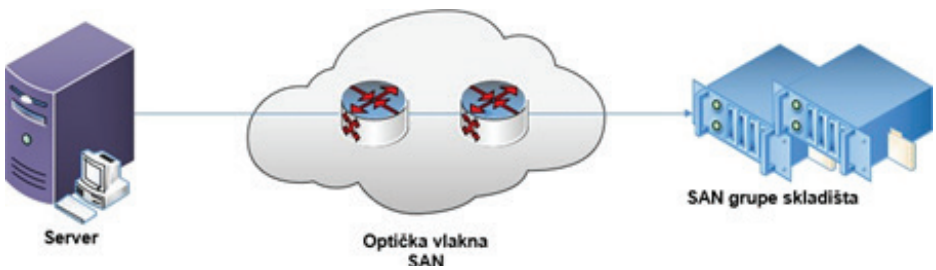
Računarska mreža skladišta (eng. *Storage Area Network – SAN*) je mreža koja podržava velike brzine transfera podataka i redudantnost i koristi se isključivo za povezivanje skladišnih uređaja.

Kada se server poveže na *SAN* mrežu, ova mreža mora da bude slobodna da istog trenutka izvrši sve zahteve koji se odnose na transfer podataka. Ovakve mreže se najčešće realizuju korišćenjem optičkih vlakana (*Fibre Channel - FC*), koja imaju znatno veću propusnu moć nego standardni *UTP* (*Unshielded Twisted Pair*) kablovi.

Koncept *SAN* klaud skladišta podataka prikazan je na Slika 9-3.

U klaud infrastrukturi provajdera obično se implemenitraju makar dve ovakve mreže za potrebe redudantnosti i smanjivanja mogućnosti otkaza. Obe mreže funkcionišu paralelno, gde se druga mreža koristi kao *backup* opcija i preuzima mrežni saobraćaj u slučaju otkaza glavne mreže.

Slika 9-3: *SAN skladište podataka*



9.1.4. Objektno skladište podataka

U klaud okruženjima se sve češće koristi objektni model za skladištenje podataka, koji se razlikuje od poznatih tehnologija za skladištenje, koji podatke prenose u formi fajlova, ili blokova podataka.

> U objektnim implementacijama skladišta, podaci (na primer fajlovi) se uparuju sa metapodacima i na taj način se kreiraju skladišni objekti (eng. *storage object*).

Za razliku od hijerarhijskih struktura koje se koriste u tradicionalnim blokovima, ili fajl skladišnim sistemima, objekti se čuvaju u formi nestruktuiranih, tzv. ravnih fajlova (eng. *flat file*).

Osim toga, s obzirom da u svakoj implementaciji objektnog skladišta podataka egzistira skup objekata, objekti se adresiraju pomoću svog jedinstvenog *ID* broja, koji su analogni nazivima fajlova u klasičnim skladištima.

ID objekta (eng. *object ID*) je pokazivač na objekat uskladištenih podataka i predstavlja globalni jedinstveni identifikator. U klaud sistemima, *ID* objekata se koristi za lociranje podataka, ili metapodataka u klaud fajl sistemu (eng. *cloud file system*).

Metapodatak (eng. *metadata*) je deo fajla ili zaglavlja sektora u skladišnom sistemu koji se koristi za identifikaciju sadržaja podatka. Metapodaci se koriste u brojnim aplikacijama za velike podatke u svrhu indeksiranja i pretraživanja podataka unutar fajla. Metapodaci najčešće čuvaju veliki broj različitih informacija, kao što su tip fajla, ili aplikacije, nivo sigurnosti, listu korisnika koji mogu da pristupe fajlu i izvršavaju operacije nad fajlom, itd. Objektna skladišta omogućavaju administratorima sistema da definišu tip podataka u metapodacima i da ga asociraju sa fajlom.

Prošireni metapodaci (eng. *extended metadata*) podržavaju proširenu listu podataka, koji mogu da se povežu sa fajlom, kao što su na primer ime vlasnika fajla, tip autentifikacije, sigurnosni sertifikati, tip autentifikacije, korisničko ime, lozinka i slično.

9.2. DODAVANJE KLAUD SKLADIŠTA

Dodavanje klaud skladišta podataka (eng. *claud storage provisioning*) je proces kreiranja i dodeljivanje skladišta podataka uređaju koji će skladište da koristi, najčešće u formi volumena (eng. volume), koji se montira (eng. *mount*) na uređaj, i kojem se pristupa sa tog uređaja, ili sa nekog udaljenog računarskog resursa.

Dodavanje skladišta podataka u velikom klaud računskom centru je složen proces kojim se najčešće bavi tim inženjera za skladišta podataka. Osim što su zaduženi za proces dodavanja, tim inženjera takođe i nadgleda i upravlja klaud skladištima podataka i izvršava eventualne konfiguracione izmene ako se za tim ukaže potreba.

U procesu dodavanja novog klaud skladišta podataka mnoge komponente treba da se instaliraju i konfigurišu, kao što su *LUN* (*Logical Unit Number*) i *VSAN* (*Virtual Storage Area Network*). Prilikom dodavanja klaud skladišta takođe treba da se obrati pažnja na sigurnost, koja treba da bude definisana u skladu sa zahtevima za zaštitu računarskih resursa. Nova skladišta se montiraju na virtuelne mašine korišćenjem *HBA* (*Host Bus Adapter*) priključka.

Na sreću zahvaljujući tehnologiji automatizacije koja se široko koristi u klaud sistemima, sve navedene operacije se izvršavaju automatizovano, u pozadini, i dodavanje novih skladišta predstavlja jednostavan zadatak za klaud klijente. Kako bi dodao novo klaud skladište, klaud klijent ne treba da poseduje tehničko znanje. Sve što klaud klijent treba da uradi jeste da, preko *Web* ili drugog interfejsa, prilikom dodavanja klaud skladišta izabere željene opcije, poput opcija za veličinu skladišta, tip skladišta, replikaciju skladišta, izradu rezervnih kopija, sigurnosti, itd. Nakon izbora navedenih opcija, klikom na dugme *Create Storage*, željeno skladište podataka se automatski kreira.

U praksi postoji razlika između tzv. debelog i tankog dodavanja klaud skladišta, pa su u nastavku ova dva procesa ukratko opisana.

9.2.1. Debelo dodavanje skladišta

U toku implementiranja skladišta na klaudu sistem za automatizaciju ovog procesa može već prilikom kreiranja novog volumena da alocira sav skladišni kapacitet, ili može prilikom kreiranja novog volumena da alocira manji skladišni kapacitet, koji će da se povećava tokom vremena po potrebi.

Debelo dodavanje klaud skladišta (eng. *thick provisioning*) je proces gde se u trenutku kreiranja virtuelnog volumena, na virtuelni volumen alocira ukupni željeni kapacitet sa diska fizičkog servera.

Tako na primer, ako klijent želi da doda novi disk u svoju virtualizoavanu klaud infrastrukturu veličine 500GB, u slučaju debelog dodavanja skladišta, svih 500GB će automatski da se alocira sa diska fizičkog servera na kojem se hostuje virtuelna infrastruktura.

9.2.2. Tanko dodavanje

S druge strane, termin tanko dodavanje (eng. *thin provisioning*) odnosi se na klaud skladište podataka čiji se kapacitet dinamički povećava po potrebi.

Primenom tankog dodavanja klaud skladišta, kapacitet fizičkog servera na kome se *hostuje* virtuelna klaud infrastruktra se bolje iskorišćava.

U slučaju tankog dodavanja, u momentu kreiranja novog volumena, manji deo fizičkog diska će biti alociran, koji će da se povećava po potrebi sve do maksimalne dozvoljene veličine.

Zbog toga što se u slučaju tankog dodavanja kapacitet skladišta dinamički povećava tokom korišćenja, ovaj način je sporiji i ponekada će doći do kašnjenja u samom pristupu skladištu. Tako na primer, ako klijent hoće da doda novi disk veličine 500GB u svoju klaud infrastrukturu, prilikom kreiranja novog *volumena*, volumenu će u početku na primer biti dodeljeno samo 10GB prostora, dok će zatim tokom korišćenja po potrebi kapacitet skladišta da se povećava do maksimalne veličine od 500GB.

9.3. PREKORAČENJE SKLADIŠTA

Primenom koncepta prekoračenja skladišta (eng. *storage overcommitting*) može da se doda više virtuelnog skladišnog prostora nego što je raspoloživo na fizičkom skladištu.

Drugim rečima, veličina virtuelnog diska se konfiguriše da bude veća nego što je veličina fizičkog diska.

Za konfigurisanje mogućnosti prekoračenja skladišta mora da se koristi tanko dodavanje skladišta, zato što će kapacitet skladišta dinamički da se povećava po potrebi do maksimalno definisane veličine. Prekoračenje skladišta je moguće zato što virtuelne mašine koriste deljene (zajedničke) resurse za skladištenje podataka. Neće u svakom trenutku svakoj virtuelnoj mašini biti potreban sav skladišni kapacitet.

Međutim, kada se koristi tehnologija prekoračenja skladišta, potrebno je da se obazrivo postupa i vešto upravlja skladištem, kako ne bi došlo do izgladnjivanja (eng. *starvation*) pojedinih virtuelnih mašina. Do izgladnjivanja može da dođe u situacijama kada je virtuelnoj mašini potreban dodatni skladišni prostor, koji u trenutku ne može da se dodeli, što kao posledicu ima nemogućnost upisivanja podataka na disk zbog nedostatka prostora.

9.4. MIGRACIJA SA JEDNE FIZIČKE MAŠINE NA DRUGU

U migraciji na klaud okruženje postoje situacije kada skladišta, servisi i aplikacije neće moći da se migriraju na virtuelnu mašinu, već će eksplicitno morati da budu migrirani na fizički klaud server.

Mnoge starije aplikacije koje je potrebno migrirati na klaud infrastrukturu ne podržavaju tehnologiju virtuelizacije i zbog svoje prirode moraju da se izršavaju na fizičkom hardveru. Često se dešava da takve stare aplikacije (eng. *legacy applications*) ne podržavaju virtuelne procesore, memoriju, pa čak ni virtuelna skladišta. Bez obzira što je korišćenje fizičkog hardvera u suprotnosti sa osnovnim modelom klaud računarstva, mnogi klaud provajderi pružaju ove usluge.

Aplikacije se sa jedne na drugu fizičku mašinu najčešće migriraju zajedno sa skladištem podataka.

U slučaju migracije sistema sa fizičkog servera u lokalnom okruženju na fizički server u klaud okruženju, cene usluga klaud provajdera eksponencijalno rastu zato što klaud provajder mora da delegira ceo fizički server samo jednom klijentu. Osim toga, i operacije nadgledanja i praćenja fizičke mašine su znatno skuplje nego što je to slučaj sa virtuelnim mašinama.

Kada aplikacije i podaci migriraju sa jednog na drugi fizički server (eng. *physical to physical – P2P*) potrebno je na ciljnom serveru izvršiti početnu instalaciju operativnog sistema i instalirati i konfigurisati aplikativni softver za izvršavanje migracije. *P2P* migracijom se premešta aplikacija zajedno sa operativnim sistemom u klaud okruženje, pri čemu treba da se uzmu u obzir i hardverske platforme i upravljački programi. Za izvršavanje ovakve migracije uglavnom se koristi specijalizovani softver treće strane.

9.5. ŠIFROVANJE PODATAKA

Posmatrano iz aspekta računarske sigurnosti, podaci na klaudu se u najširem smislu dele u dve grupe, i to na podatke u stanju mirovanja (eng. *data at rest*) i podatke u pokretu.

Podaci u mirovanju su podaci koji su uskladišteni na skladištu podataka, dok su podaci u pokretu podaci koji se kreću putem računarske mreže s jedne lokacije na drugu lokaciju (s jednog čvora u mreži na drugi čvor).

Radi zadovoljavanja cilja zaštite podataka, i podaci u mirovanju i podaci koji se kreću se šifruju. Šifrovanje podataka (eng. *data encryption*) koji miruju vrši se šifrovanjem celog klaud skladišnog *volumena* na kojem su podaci pohranjeni.

Za šifrovanje i dešifrovanje podataka koriste se ključevi, kojima upravlja, ili klaud provajder, ili kranji korisnik, što zavisi od Sporazuma o nivou korišćenja usluga.

Šifrovanje podataka u mirovanju je često jedan od zahteva organizacione politike. Kada su volumeni za skladištenje podataka ispravno konfigurisani, šifrovanje podataka je transparentan proces koji se izvršava u pozadini.

Kada su implementirana redudantna skladišta podataka, šifruju se podaci i na glavnom i na rezervnom skladištu.

9.6. PRISTUP PODACIMA U KLAUD SKLADIŠTU POMOĆU TOKENA

Kao jedan od načina za obezbeđenje pristupa podacima u klaud skladištima koriste se pristupni tokeni (eng. *access tokens*). Kada bi se napravila analogija sa realnim životom, tokeni su kao ključevi – onaj ko ima ključ može da otvori vrata. Slično je i sa tokenima – uređaj koji ima pristupni token može da pristupi određenom delu, ili celom skladištu na klaud mreži.

Sistemi za skladištenje podataka generišu pristupne tokene, koje zatim koriste klijenti. Tako na primer *Web* brauzer može da preuzme pristupni token sa *Web* sajta i da ga koristi kada želi da pristupi *volumenu* na *Web* serveru.

Token za pristup skladištu je kriptografski ključ koji se koristi za pristup osetljivim podacima i fajlovima na medijumu za skladištenje. Tokeni su elektronski ključevi koji se koriste za otključavanje pristupa ka uskladištenim podacima.

Tokeni se takođe koriste i za autentifikaciju. Ako korisnik koji hoće da pristupi skladištu ima validan kriptografski token, može da dokaže svoj identitet, zato što je token generisan na sajtu kojem korisnik hoće da pristupi.

U klaud okruženju tokeni se najčešće koriste za autentifikaciju aplikativnog programskog interfejsa ili konzolnog interfejsa i za pristup klaud skladištu, serveru, ili nekom drugom računarskom resursu.

9.7. SLOJEVITA ARHITEKTURA KLAUD SKLADIŠTA

Slojevita arhitektura klaud skladišta (eng. *mutlti-tier cloud architecture*) koristi se pre svega zato što različiti tipovi i vrste podataka mogu da imaju različite zahteve u pogledu skladištenja, kao što su na primer zahtevi na osnovu toga koliko su podaci kritični (osetljivi), koliko često treba da im se pristupa, zahtevi u smislu fizičke, tj. geografske lokacije skladištenja, ili bilo koji zahtevi koji se odnose na šifrovanje i sigurnost podataka.

Takođe, pojedine aplikacije mogu da imaju veće zahteve u vidu brzine čitanja i upisivanja podataka, učestalosti pristupa podacima i slično. Zbog svega navedenog postoji realna potreba za definisanjem različitih slojeva skladišta, gde svaki sloj zadovoljava određeni zahtev, ili grupu zahteva.

U literaturi se najčešće navode tri sloja klaud skladišta.

Prvi sloj klaud skladišta koristi se za najosetljivije i najkritičnije podatke koji se čuvaju na najbržim i najkvalitetnijim medijima za skladištenje podataka uz visok nivo redudantnosti. Ovakav sloj ima posebne zahteve u pogledu pouzdanosti i raspoloživosti podataka i zbog toga se implementira uz poštovanje principa redudantnosti i zaštite od otkaza (na primer implementacijom *RAID* sistema, o čemu će biti više reči u nastavku). Osim navedenog, prvi sloj klaud skladišta najčešće implementira sofisticirane upravljačke mehanizme i mehanizme praćenja uz visok nivo *I/O* performansi.

Drugi sloj klaud skladišta je u pogledu pouzdanosti, sigurnosti i redudantnosti značajno slabijih performansi od prvog sloja. Ovaj sloj se koristi za podatke koji nemaju visoke zahteve u pogledu brzine pisanja i upisivanja i za podatke kojima se ne pristupa često. Za implementaciju ovog sloja koristi se jeftinija oprema. Primeri skladišta drugog sloja su skladišta za *email* poruke, deljene fajlove, itd.

Konačno, treći sloj skladišta je tehnološki nazadniji od prvog i drugog sloja i koristi se uglavnom za kreiranje rezervnih kopija, tj. replikacije podataka sa prvog i drugog skladišnog sloja. Treći sloj skladišta najčešće ima veliki kapacitet uz relativno nisku cenu, jer se koriste jeftiniji medijumi, poput *DVD*-a (*Digital Video Disk*). Glavni trećeg sloja klaud skladišta je taj što su podaci slabije zaštićeni i što je potrebno relativno mnogo vremena da im se pristupi.

U arhitekturi klaud skladišta koriste se razni modeli. U praksi se koriste i klaud skladišta sa više od tri nivoa (sloja), prema definisanim zahtevima. Primenom skripti i sistema za automatizaciju u klaud okruženju podaci relativno lako mogu da migriraju sa jednog na drugi sloj skladišta, uz primenu jednostavnih procedura za usklađivanje sa korporativnim politikama za čuvanje podataka.

Što je viši sloj, to su podaci koji se čuvaju kritičniji i arhitektura se više bazira na performansama, redudantnosti i raspoloživosti. Na primer, prvi sloj klaud skladišta će uglavnom imati bolje performanse od drugog sloja, koji će imati bolje performanse od trećeg sloja, itd. Podaci kojima se ređe pristupa i koji su manje kritični skladište se u skladištu trećeg, ili viših slojeva.

Pravilnom kategorizacijom podataka i razvrstavanjem u grupe, pa zatim i u slojeve klaud provajderi obezbeđuju značajne uštede, kako sebi, tako i svojim klijentima. Takođe, i na nivou kompanije klijenta definišu se politike za skladištenje podataka, gde se podaci na pravilan način organizuju u grupe i u slojeve, čime se takođe postižu značajne uštede.

Slojevita arhitektura klaud skladišta ilustrovana je na Slici 9-4.

Slika 9-4: *Slojevita arhitektura klaud skladišta*



9.8. UPRAVLJANJE I ZAŠTITA PODATAKA NA KLAUDU

Kao što je već više puta bilo naglašeno, model zajedničke odgovornosti, prema kojem odgovornost dele i klaud provajder i klaud klijent, primenjuje se u klaud okruženju. Odgovornost se odnosi na sve računarske resurse, pa tako i na skladišta podataka i to po svim aspektima upotrebe, sigurnosti, zaštite i bezbednosti.

Kada se govori o skladištima na klaudu, klaud provajder preduzima aktivnosti „u pozadini", koje su transparentne za klaud klijenta, kako bi zaštitio i na pravilan način upravljao svim uskladištenim podacima. S druge strane, klaud klijentu su na raspolaganju brojne opcije koje proširuju raspoloživ skup alata za upravljanje klaud podacima, gde je jedan od glavnih ciljeva unapređenje sigurnosti podataka.

U nastavku su navedeni najvažniji upravljački aspekti klaud podacima.

9.8.1. Visoka raspoloživost klaud podataka i otpornost na otkaze

Kao što je već i navedeno, visok nivo raspoloživosti je sposobnost da računarski resurs ostane dostupan i nakon prekida u radu, tj. pada sistema (eng. *system crash*). Sistemi koji nemaju implementiran ovaj mehanizam, u slučaju otkaza neke od komponente sistema, prestaju da rade određeni vremenski interval, što se u literaturi naziva vremenski interval u kome sistem nije operativan (eng. *system downtime*).

Visoka raspoloživost može da se posmatra i iz aspekta klaud podataka. Klaud podaci treba da zadovoljavaju zahteve visoke raspoloživosti i da u svakom vremenskom trenutku, na zahtev, budu dostupni klaud klijentima.

Zahtev visoke raspoloživosti postiže se implementiranjem mehanizama za otpornost na otkaze (eng. *fault tolerance*). Sistemi koji su otporni na otkaze su oni sistemi koji će nastaviti da funkcionišu i onda kada neka od komponenti sistema otkaže.

Visoka raspoloživost klaud podataka postiže se korišćenjem klaud skladišnih sistema koji su otporni na otkaze.

9.8.2. Replikacija klaud podataka u i izvan regiona

Replikacija klaud podataka (eng. *claud data replication*) odnosi se na proces čuvanja podataka na više različitih lokacija (na više različitih sistema) u cilju bržeg i lakšeg oporavka od katastrofe.

Replikacijom klaud podataka obezbeđuje se veća robusnost i otpornost klaud infrastrukture. Ako bi se svi podaci čuvali u jednoj zoni dosupnosti (eng. *availability zone*), u slučaju da se ta zona izgubi, svi podaci bi bili izgubljeni.

Zbog navedenog, uobičajena praksa je da se podaci čuvaju na više od jedne lokacije koje su geografski udaljene i koje se nalaze u različitim regijama klaud okruženja. Klaud provajderi nude usluge replikacije podataka, kao deo svoje osnovne ponude klaud resursa za skladištenje podataka. Tako na primer, kompanija može da kupi prekonfigurisani uređaj za skladištenje podataka od klaud provajdera koji automatski pravi rezervne kopije podataka na više zona dostupnosti, ili čak na više regiona.

U literaturi se najčešće navode dva osnovna tipa replikacije podataka: sinhrona i asinhrovana replikacija. Sinhrona replikacija (eng. *synchronous data replication*) kopira u isto vreme podatke i na primarni sistem za skladištenje, i preko mreže na udaljenu lokaciju, gde su podaci na obe lokacije ažurirani i sinhronizovani. Sinhrona replikacija podataka se obavezno koristi za visoko osetljive podatke, kao što su podaci u transakcionim bazama podataka. U slučaju sinhrone replikacije, nakon upisivanja podataka i na primarnu i na udaljenu lokaciju, potrebna je i potvrda da su podaci uspešno upisani i da su sinhronizovani.

S druge strane, u slučaju asinhrone replikacije (eng. *asynchronous data replication*), podaci se prvo upisuju u primarno skladište podataka, da bi se zatim upisali i na udaljenu lokaciju. Ovakva replikacija koristi tzv. princip „sačuvaj i preusmeri" (eng. *store and forward*), gde može doći do zastoja i kašnjenja u kopiranju podataka. Sekundarno skladište se skoro svakom vremenskom trenutku kada se vrši prebacivanje podataka nalazi nekoliko transakcija iza primarnog skladišta.

9.8.3. Korišćenje ogledala sajta i kloniranje podataka

U cilju postizanja robusnosti i pouzdanosti klaud sistema, redudantnost se ponekada sprovodi i na nivou cele lokacije na kojoj se nalazi klaud oprema. U literaturi se ova lokacija naziva sajt (eng. *site*).

U praksi se često dešava da veliki i složeni klaud sistemi imaju više sajtova, gde je sekundarni sajt kopija primarnog sajta.

Primenom koncepta kreiranja ogledala sajta (eng. *site mirroring*), za svaki sajt organizacije kreira se njegova identična kopija u vidu sekundarnog sajta. Sekundarni sajt se ažurira u realnom vremenu podacima sa glavnog sajta.

U slučaju otkaza primarnog sajta, sekundarni sajt (odraz u ogledalu glavnog sajta) može da preuzme svo procesiranje. Takođe, sekundarni sajt se koristi i kako bi se poboljšala raspoloživost resursa i bolje zadovoljili zahtevi za većim obimom saobraćaja, kada sekundarni sajt ima ulogu balansera opterećenja.

Osim opisanog pristupa, klaud provajder obezbeđuje i usluge kloniranja uskladištenih podataka. Proces kloniranja podataka može da se automatizuje pomoću skripti po unapred definisanom rasporedu, ili može da se izvršava manuelno.

Volumen za skladištenje može da se klonira i replikuje na druge klaud regione kreiranjem novih instanci virtuelnih mašina, ili klonirani volumen može da se čuva u formi rezervne kopije ili snimka sistema (eng. *snapshot*) za potrebe oporavka od katastrofa.

9.9. RAID TEHNOLOGIJA

U cilju pružanja što kvalitetnijih i pouzdanijih klaud usluga, redudantnost računarskih komponenti i sistema se obezbeđuje na više nivoa. U prethodnoj sekciji ukratko je opisana redudantnost na nivou klaud sajta. Na nivou ispod sajta implementira se redudantnost servera i ostalih računarskih sistema i to najčešće kao klasteri servera za balansiranje opterećenja (eng. *load balansing clusters*). ili klastera servera za zaštitu od otkaza (eng. *failover clusters*).

Konačno, i na nivou samog servera, često se koriste redudantne komponente, poput mrežnih kartica, jedinica za napajanja električnom energijom (eng. *Power Supply Unit – PSU*) i skladišnih uređaja. Danas je prava retkost da se samo jedan skladišni uređaj priključuje na server u klaud računskom centru.

Više skladišnih uređaja se koristi zajedno kako bi se postigla poboljšanja u performansama, postigla redudantnost i omogućilo jednostavno skaliranje.

Tehnologija skladištenja koja omogućava kombinovanje više hard disk komponenti u jednu logičku celinu sa ciljem postizanja redudantnosti podataka i poboljšanja performansi naziva se *RAID*.

RAID tehnologija ima veliki značaj u zaštiti klaud računarskih podataka.

Primenom ove tehnologije, podaci se distribuiraju na hard diskove na nekoliko načina, koji se nazivaju *RAID* nivoi. Svaki nivo omogućava redudantnost i/ili unapređuje performanse.

Istorijski posmatrano, termin *RAID* je imao nekoliko značenja. Kasnih 80-tih godina prošlog veka, *RAID* je označavao redudantni skup jeftinih diskova (*Redudandant Array of Inexpensive Disks*). Međutim, kako su *vremenom* hard diskovi postali jeftiniji, značenje termina *RAID* je promenjeno u redudantni skup nezavisnih diskova (*Redudandant Array of Independent Disks*).

RAID tehnologija se danas široko primenjuje za potrebe masovnog skladištenja korporativnih podataka i predstavlja neizostavni deo svakog klaud okruženja. Grupisanjem fizičkih diskova u logičku celinu mogu da se kreiraju volumeni velikog kapaciteta. Na ovaj način postiže se otpornost na otkaze (eng. *fault tolerance*) korišćenjem redudantnih diskova, poboljšavaju se performanse i omogućava se veći kapacitet za skladištenje podataka.

Primenom *RAID* tehnologije diskovi mogu da se konfigurišu na različite načine, koji se u literaturi nazivaju *RAID* nivoi. Svaki *RAID* nivo ima određene prednosti i nedostatke. Skupom diskova koji su povezani može da upravlja softver koji je najčešće implemeniran kao sistemski softver u okviru operativnog sistema, ili hardver koji se naziva *RAID* kontroler (eng. *RAID controller*). U klaud okruženjima se skoro isključivo koriste *RAID* kontroleri, jer su oni mnogo efikasniji, brži i pouzdaniji od softvera koji upravlja *RAID* konfiguracijama. Hardverski *RAID* rasterećuje rad procesora, dok u slučaju kada se softver koristi za upravljanje diskovima, procesor je izložen dodatnom opterećenju.

RAID nivoi se opisuju celim brojevima – 0,1,5, itd. Najčešće korišćeni *RAID* nivoi su:

- *RAID 0*
- *RAID 1*
- *RAID 5*
- *RAID 0+1*
- *RAID 1+0*
- *RAID 5+0*

Osim navedenih, postoje i drugi *RAID* nivoi poput *RAID 3, RAID 4, RAID 6*, itd., ali se oni ređe koriste u praktičnim implementacijama. U nastavku je ukratko opisan svaki od gore pobrojanih najčešće korišćenih *RAID* nivoa.

9.9.1. *RAID* 0

RAID 0 koristi *striping* tehnologiju bez preslikavanja i pariteta.

Nivo *RAID* 0 se ne smatra pravim *RAID*-om, zato što ne nudi redudantnost. Zbog toga se *RAID* 0 najčešće kombinuje sa drugim *RAID* implementacijama kako bi se postigla zaštita od otkaza. Međutim, bez obzira što ne obezbeđuje zaštitu od otkaza, *RAID* 0 omogućava najbolje performanse od svih drugih *RAID* nivoa.

U slučaju korišćenja *RAID* 0 tehnologije, podaci se prvo dele na delove koji se nazivaju trake (eng. *stripes*), koji se zatim simultano upisuju na sve diskove koje čine *RAID* sistem. Skup svih diskova (eng. *stripe set*) operativni sistem „vidi" kao jedan logički disk.

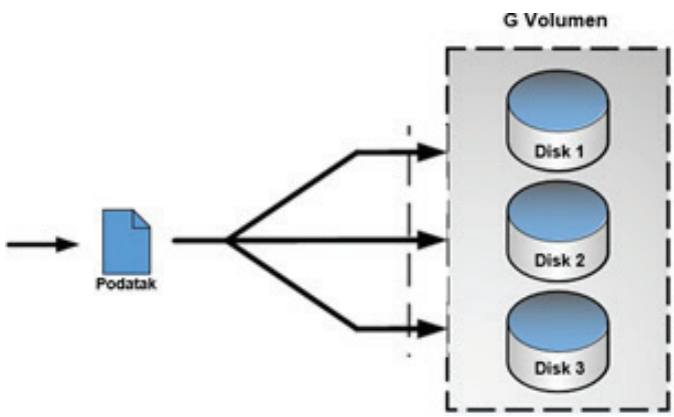
Primenom *striping* tehnologije vrši se segmentacija sekvencijalnih podataka, gde se segmenti konkurentno upisuju na više fizičkih diskova koji su spojeni u *RAID* po kružnom principu. Fajl se deli u blokove podataka i blokovi se rasipaju u više diskova.

RAID 0 ne koristi preslikavanje, niti parnost (videti u nastavku), i zato ne obezbeđuje ni redudantnost, niti detekciju grešaka. Ako se jedan od diskova koji su povezani u *RAID* pokvari, svi podaci će biti izgubljeni. S druge strane, pošto *RAID* 0 omogućava paralelizaciju operacija čitanja i upisivanja podataka, veoma je brz i donosi prednosti u klaud okruženjima gde se skladište često koristi, kao što je slučaj sa aplikacijama baza podataka.

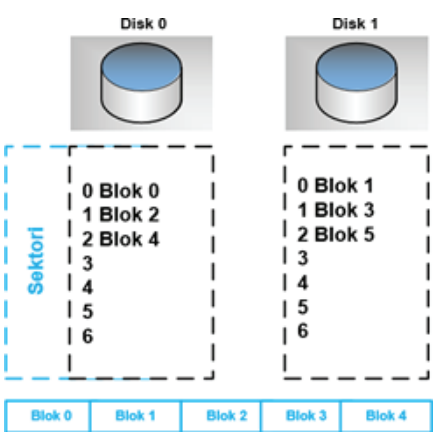
Za bolje razumevanje funkcionisanja *RAID* 0 nivoa može da posluži sledeći primer. Neka je potrebno da se reč „get" sačuva na skladištu podataka koje se sastoji iz 3 hard diska povezanih *RAID* 0 tehnologijom u jednu logičku celinu. S obzirom da se reč get sastoji iz 3 slova, svaka reč može da se sačuva na jednom od povezanih diskova. Proces skladištenja ove reči može da se ubrza tako što simultano (paraleleno) prvo slovo reči „g" upisuje na prvi disk, drugo slovo reči „e" upisuje na drugi disk i treće slovo reči „t" upisuje na treći disk. Očigledno je da se ovim postupkom poboljšavaju perfromanse. Nažalost, ako samo jedan od 3 diska otkaže, cela reč će biti izgubljena.

Grafički prikaz funkcionisanja *RAID* 0 nivoa dat je na slikama 9-5 i 9-6.

Slika 9-5: *RAID 0 nivo povezivanja diskova*



Slika 9-6: *RAID 0 tehnologija*



9.9.2. *RAID* 1

Nivo *RAID* 1 je prvi *RAID* nivo koji obezbeđuje redudantnost, pa samim tim i zaštitu od otkaza.

U literaturi se navode dva tipa implementacije *RAID* 1 tehnologije:

- preslikavanje diskova (eng. *disk mirroring*) i
- dupleksiranje diskova (eng. *disk duplexing*).

Obe navedene implementacije koriste dva ili više diskova i obezbeđuju redudantnost kreiranjem identičnih kopija (preslikavanjem jednog diska na drugi disk). Primenom ovog pristupa postiže se otpornost na greške. Otkazom jednog diska podaci neće biti izgubljeni.

Tehnologije preslikavanja diskova i dupleksiranja diskova razlikuju se samo po načinu na koji se diskovi fizički povezuju. U slučaju preslikavanja diskova, svi diskovi u skupu se povezuju na isti disk kontroler. S druge strane, kada se koristi tehnologija dupleksiranja diskova, diskovi u skupu se povezuju na makar dva disk kontrolera.

Dakle, tehnologija dupleksiranja diskova je tolerantnija na otkaze od tehnologije preslikavanja diskova zato što eliminiše disk kontroler kao jednu tačku prekida funkcionisanja sistema. U klaud okruženjima se uglavnom koristi tehnologija dupleksiranja diskova.

Dok nivo *RAID* 0 obezbeđuje performanse, *RAID* 1 obezbeđuje pouzdanost. Tehnologija koju koristi *RAID* 1 je relativno jednostavna – prvo se ceo fajl iskopira na jedan disk, a zatim se na drugom disku napravi identična kopija tog fajla.

Kopiranjem podataka na dva diska postiže se redudantnost i podaci se čitaju sa dva diska istovremeno, što poboljšava vreme čitanja sa diskova. S druge strane, kod upisanja podataka performanse su lošije, jer se podaci simultano upisuju na dva diska. Podaci su upisani tek onda kada je završen proces upisivanja na najsporiji disk iz skupa.

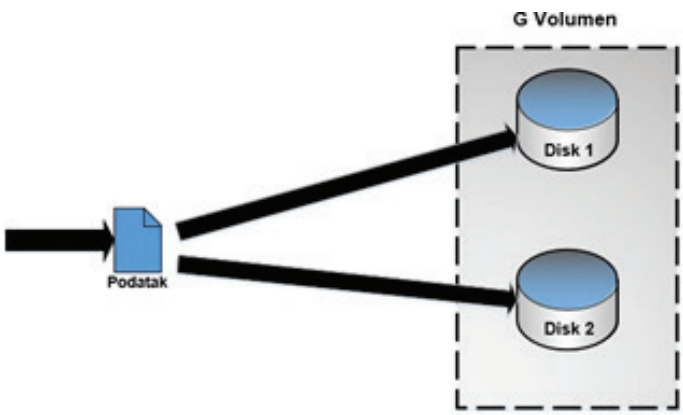
U literaturi se često *RAID* 1 nivo naziva samo kao tehnologija preslikavanja (eng. *mirroring*), zato što svaki disk ima svoju identičnu kopiju, gde se pravi analogija sa odrazom u ogledalu.

Primenom *RAID* 1 tehnologije, za svaki upisani podatak, se u realnom vremenu pravi njegova rezervna kopija. Međutim, *RAID* 1 ne treba koristiti kao zamenu za sistem za izradu rezervnih kopija, jer bez obzira što omogućava zaštitu od otkaza, *RAID* 1 ne implementira mehanizme zaštite u slučaju kvara podataka (eng. *data corruptions*).

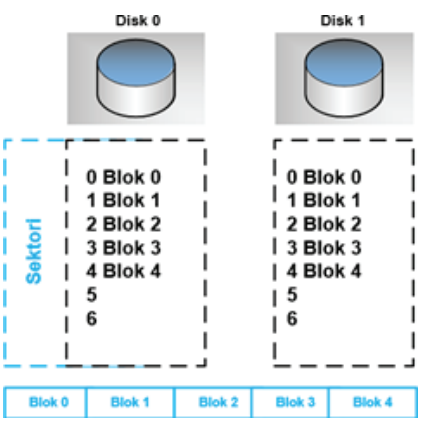
Jedan od velikih nedostataka *RAID* 1 tehnologije je trošak za nabavku diskova u vidu neiskorišćenog skladišnog prostora, gde se 50% skladišnog kapaciteta gubi. Tako na primer, ako je potrebno da se kreira *RAID* 1 skup kapaciteta 100GB, potrebno je nabaviti dva diska od po 100GB, što u nekim slučajevima može da predstavlja veliki trošak.

Vizuelna reprezentacija funkcionisanja *RAID* 1 nivoa data je na slikama 9-7 i 9-8.

Slika 9-7: *RAID 1 nivo povezivanja diskova*



Slika 9-8: *RAID 1 tehnologija*



9.9.3. *RAID* 5

Tehnologija *RAID* 5 funkcioniše slično kao *RAID* 0, tako što rasipa podatke na više diskova, ali koristi i bit parnosti koji omogućava zaštitu od otkaza.

Razlike između nivoa *RAID* 5 i *RAID* 0 ogledaju se u sledećem:

- *RAID* 5 koristi parnost (eng. *parity*) u cilju postizanja otpornosti od otkaza i
- *RAID* 5 zahteva povezivanje minimum tri ili više fizičkih diskova, dok je za konfigurisanje *RAID* 0 nivoa potrebno minimum dva diska.

Sa svakim upisivanjem podataka, *RAID* 5 upisuje bit parnosti (eng. *parity bit*) na jedan disk u skupu. Na ovaj način, ako jedan disk otkaže, skup diskova će nastaviti da funkcioniše. Međutim, ako i drugi disk otkaže, svi podaci će biti izgubljeni. U ovom slučaju, potrebno je da se ponovo izgradi skup (eng. *rebuilding disk array*) i da se podaci oporave iz rezervnih kopija.

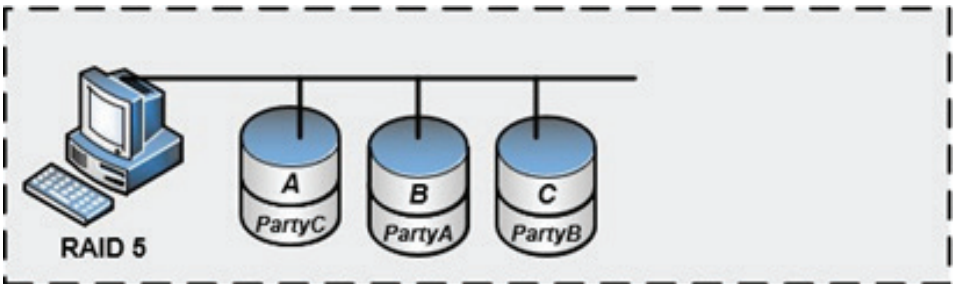
Po pitanju performansi, sistem *RAID* 5 je sporiji od *RAID* 0, ali je zato brži od *RAID* 1 tehnologije. *RAID* 5 tehnlogija značajno poboljšava operacije pisanja u odnosu na *RAID* 1 sistem, zato što se pisanje vrši paralelno. Performanse pisanja u slučaju *RAID* 5 nivoa su lošije u odnosu na *RAID* 0 zato što je osim podataka pri svakom upisivanju potrebno upisati i informacije o parnosti (eng. *parity information*) na jedan disk u skupu, koje se kasnije koriste za rekonstruisanje podataka ako jedan od diskova u nizu otkaže.

U procesu implementiranja *RAID* 5 niza potrebno je posebnu pažnju obratiti na broj potrebnih diskova. Jedan disk se delegira za čuvanje informacija o parnosti. Tako na primer, ako se nivo *RAID* 5 kreira pomoću tri diska veličine 100GB, efektivni skladišni kapacitet će biti 200GB, jer se 100GB (67%) gubi na čuvanje bitova parnosti.

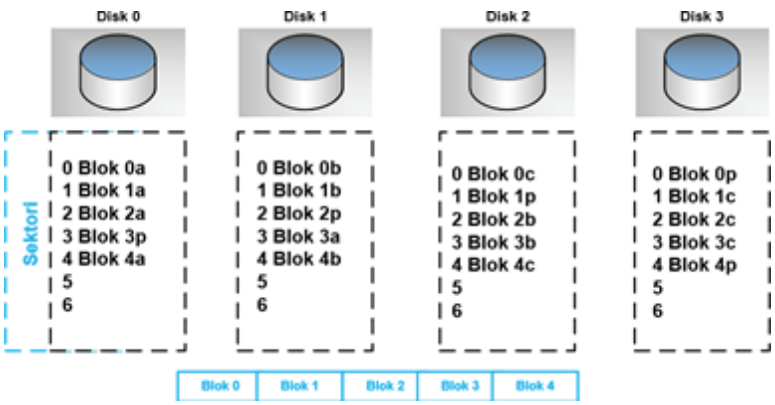
Minimalan broj diskova u *RAID* 5 skupu je tri, ali je preporučljivo da se koristi više diskova. Tehnologija *RAID* 5 se često koristi na klaudu.

Prikaz *RAID* 5 tehnologije sa tri i četiri diska dat je na slikama 9-9 i 9-10, respektivno.

Slika 9-9: *RAID 5 tehnologija sa tri diska*



Slika 9-10: *RAID 5 tehnologija sa četiri diska*



9.9.4. *RAID* 0+1

U praksi se *RAID* nivoi često kombinuju na različite načine, kako bi se simultano iskoristile prednosti različitih *RAID* implementacija.

Nivo *RAID* 0+1 označava kombinaciju *RAID* 0 i *RAID* 1 nivoa. Na ovaj način kombinuju se performanse *RAID* 0 sa redudantnošću i otpornošću na otkaze *RAID* 1 nivoa. Ovakva konfiguracija se u literaturi obično naziva preslikani *striping* (eng. *mirrored stripe*).

Tehnologija *RAID* 0+1 funkcioniše tako što se podaci prvo dele na trake formirajući *RAID* 0 niz, koji se zatim preslikava pomoću *RAID* 1 tehnologije.

Na ovaj način performanse se poboljšavaju, ali isto tako dolazi do povećanja troškova zbog redudantnosti *RAID* 1 nivoa. Tako na primer, ako je za potrebe skladištenja nabavljeno šest diskova veličine od po 100GB, efektivni skaldišni prostor biće samo 50% od toga, tj. 300GB. U ovakvoj konfiguraciji, prva tri diska bi se ukombinovala u *RAID* 0 skup, koji bi zatim bio preslikan na skup od preostala tri diska pomoću *RAID* 1 tehnologije.

U *RAID* 0+1 konfiguraciji, ako se izgubi jedan disk, podaci neće biti izgubljeni.

Ovaj skup ilustrovan je na Slici 9-11.

Slika 9-11: *RAID 0+1 nivo*



9.9.5. *RAID* 1+0

Nivo *RAID* 1+0, koji je takođe poznat u literaturi kao *RAID* 10, kombinuje *RAID* 1 i *RAID* 0, kako bi bio kreiran *striping* skup preslikanih *volumena*.

RAID 1+0 konfiguracija predstavlja suprotan slučaj *RAID* 0+1 konfiguracije. U ovakvoj konfiguraciji prvo se kreira preslikan skup diskova, gde se zatim podaci distribuiraju na više diskova korišćenjem *striping* tehnologije.

Glavna prednost *RAID* 1+0 tehnologije u odnosu na *RAID* 0+1 ogleda se u tome što je *RAID* 1+0 otpornija na otkaze. Ako bi se na primer koristilo šest diskova u *RAID* 1+0, moglo bi da otkaže čak tri diska, a sistem bi nastavio neometano da funkcioniše.

Dokle god jedan disk u preslikanom skupu ostane operativan u *RAID* 1+0 implementaciji, sistem će nastaviti da funkcioniše.

U kontekstu efektivnog skladišnog prostora, nivoi *RAID* 1+0 i *RAID* 0+1 su slični. Kao i u slučaju *RAID* 0+1, i u slučaju *RAID* 1+0, 50% skladišnog kapaciteta se gubi. Tako na primer u konfiguraciji sa šest diskova od po 100GB, samo 300GB skladišnog prostora će moći da se koristi.

Uz malo žrtvovanje u performansama zarad bolje efikasnosti skladišta, *RAID* 5+0 predstavlja bolji izbor.

Tehnologija *RAID* 1+0 ilustrovana je na Slici 9-12.

Slika 9-12: *RAID 1+0 nivo*



9.9.6. *RAID* 5+0

Nivo *RAID* 5+0 predstavlja kombinaciju *RAID* 5 i *RAID* 0 tehnologija, koja se implementira primenom striping tehnologije na *RAID* 5 *volumenima*.

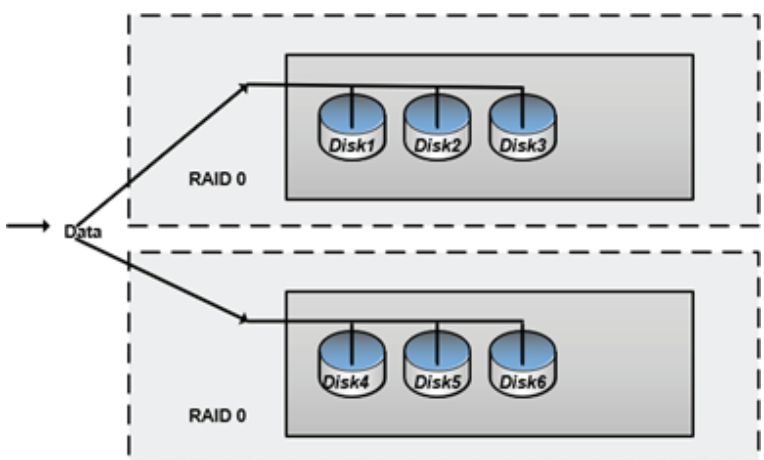
Ovakav način konfigurisanja skupa diskova je efikasniji od *RAID* 1+0 i *RAID* 0+1 implementacija zato što se generiše veći efektivni skladišni prostor, pa su samim tim i troškovi manji.

U odnosu na *RAID* 5 nivo, *RAID* 5+0 obezbeđuje brže čitanje i upisivanje podataka. Međutim, problem u *RAID* 5 implementaciji je taj, što u slučaju otkazivanja jednog diska, performanse *I/O* operacija značajno opadaju. Takođe, *RAID* 5+0 je otporniji na otkaze od *RAID* 5 sistema, zato što dozvoljava gubitak jednog diska u svakom *RAID* 5 podskupu.

Efektivni skladišni kapacitet *RAID* 5+0 je samo malo manji u odnosu na *RAID* 5 nivo. U slučaju šest diskova od po 100GB, efektivni prostor bi bio 400GB, zato što se u svakom *RAID* 5 podskupu jedan disk koristi za čuvanje informacija o parnosti.

RAID 5+0 implementacija prikazana je na Slici 9-13.

Slika 9-13: *RAID 5+0 implementacija*



9.9.7. Ostali *RAID* nivoi

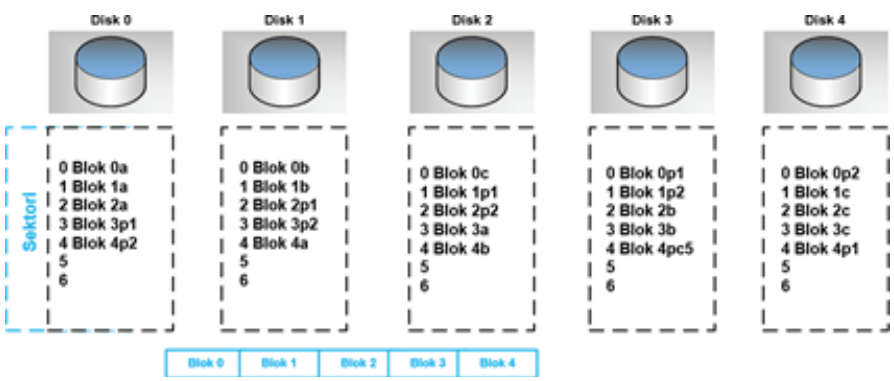
Kao što je gore navedeno, osim opisanih postoje i drugi *RAID* nivoi koji se ređe koriste u praksi. U nastavku je po koja rečenica posvećena nekima od njih.

Nivo *RAID* 3 koristi *striping* tehnologiju u kombinaciji sa čuvanjem bita parnosti. Za razliku od *RAID* 5 nova, gde se informacije o parnosti čuvaju na svim diskovima, u slučaju *RAID* 3 nivoa, informacije o parnosti se čuvaju samo na jednom, namenskom disku. *RAID* 3 tehnologija se uglavnom koristi na starijim sistemima i podržava je *Unix* operativni sistem.

RAID 6 predstavlja proširenje *RAID* 5 implementacije. U ovom slučaju čuvaju se i dodatne informacije o parnosti koje se distribuiraju na sve diskove. Kao rezultat ovoga, *RAID* 6 je otporan na otkaz dva diska, ali je zato i efektivni skladišni prostor manji, zato što se za čuvanje informacija o parnosti koristi prostor veličine kapaciteta dva diska, dok u slučaju *RAID* 5 tehnologije ovaj prostor odgovara veličini jednog diska. Međutim, osim navedenog, korišćenje *RAID* 6 nivoa nosi sa sobom i druge troškove. Operacije pisanja dodatnih informacija o parnosti zahtevaju dodatno vreme, pa operacije pisanja u ovom slučaju traju duže.

RAID 6 je prikazan na Slici 9-14.

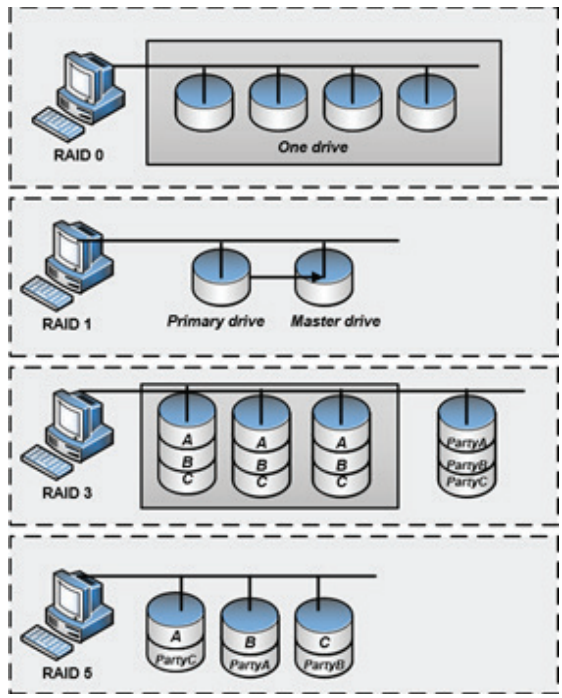
Slika 9-14: *RAID 6 nivo*



Već je pomenuto da se *RAID* sistemi kombinuju na različite načine u cilju uzimanja najboljih karakteristika kombinovanih *RAID* nivoa. Tako su poznate i kombinacije nivoa *RAID* 5 i *RAID* 1 u vidu *RAID* 1+5 i *RAID* 5+1 konfiguracija.

Radi potpunije ilustracije, na Slici 9-15 dat je uporedni prikaz *RAID* nivoa 0,1,3 i 5.

Slika 9-15: *Uporedni prikaz RAID nivoa*



9.10. SIGURNOST KLAUD SKLADIŠTA

Sigurnost podataka u klaudu je jedna od najvećih briga kompanija koje migriraju svoje operacije u klaud.

Kako bi se obezbedili odgovarajući nivoi zaštite klaud podataka, potrebno je uspostaviti odgovarajuće mehanizme za sigurnost klaud skladišta podataka.

Osim toga što se kompanije brinu za sigurnost svojih podataka na klaudu, kompanije imaju i obavezu da slede zakonske i druge regulative koje se odnose na bezbednost i zaštitu podataka u klaud okruženju. Tako na primer, postoje regulative koje zahtevaju da su podaci fizički čuvaju u zemlji odakle potiču, ili regulative koje zahtevaju da se podaci šifruju dok su u pokretu, ili dok miruju.

U nastavku su opisani neki od mehanizama koji se koriste za unapređenje sigurnosti klaud sistema za skaldištenje podataka.

9.10.1. Kontrola pristupa klaud skladištu

Kao najčešći mehanizam za kontrolu pristupa klaud skladištu koriste se pristupne kontrolne liste (eng. *access control list – ACL*).

ACL je sigurnosni mehanizam pomoću kojih se definišu pravila na osnovu kojih se pristupa skladišnim i ostalim klaud resursima. *ACL* sarži listu resursa i odredbi kojima se eksplicitno zabranjuje (eng. *deny*), ili odabrava (eng. *allow*) pristup određenom resursu.

ACL koje se primenjuju na klaud skladišta su slične onima koje se koriste na ruterima, komutatorima, *firewall*-ovima, itd. Svaki objekat skladišta može da ima svoju *ACL* koja definiše koji korisnici, grupe, ili udaljeni uređaji mogu da mu pristupe. Tako na primer, jedna grupa može da ima dozvolu samo za čitanje sa određenog skladišnog resursa (eng. *read permission*), dok druga grupa može da ima dozvole i za čitanje i za pisanje (eng. *read and write permission*), itd.

ACL omogućava granularno (fino) podešavanje pristupnih prava, sve do nivoa pojedinih fajlova, ili blokova podataka na klaud skladištu.

Dodatno se napominje da se *ACL* liste široko koriste i za pristup klaud skladištima preko *Web*-a. Na primer, kada se neki korisnik autentifikuje na *Web* sajt, on implicitno postaje član grupe autentifikovanih korisnika i omogućava mu se određeni nivo pristupa *volumenima* skladišta za koje taj korisnik ima dozvolu. Postoji više vrsta grupa, kao što su svi korisnici (eng. *everyone*), autentifikovani koristnici, anonimni korisnici, administratori, itd. Ovim grupama se najčešće dodeljuju dozvole za čitanje, pisanje, izvršavanje, ili pun pristup sistemskim objektima skladišta.

9.10.2. Zavaravanje

Jedan od čudnijih termina koji se koristi za opis komponenti klaud okruženja je zavaravanje (eng. *obfuscation*). Zavaravanje se definiše kao način da se zakomplikuje, zbuni, ili obmane.

> Termin zavarivanje se koristi za označavanje načina skrivanja informacija koje se čuvaju u klaudu okruženju.

Zavaravanje je tehnika koja se koristi sa ciljem povećavanja klaud sigurnosti, tako što se otežava čitanje legitimnih podataka koji se nalaze u fajlovima. Na ovaj način otežava se čitanje poverljivih informacija od strane malicioznih napadača, zato što su pravi podaci duboko „prekriveni" pseudo-slučajnim podacima, kada se napadač nalazi u situaciji da ne može da napravi razliku između pravih i pseudo-slučajnih podataka.

9.10.3. *SAN*, zoniranje i *LUN* maskiranje

Klasični *Ethernet* komutatori i ruteri u računarskoj mreži komuniciraju jedni sa drugima i razmenjuju informacije o dozvolama za pristup skladišnim i drugim mrežnim resursima pomoću *ACL*-a. S druge strane, *Fibre Channel* komutatori, koji se koriste za potrebe *SAN* skladišta, ponašaju se drugačije – po podrazumevanom stanju, *SAN* portovi ne komuniciraju jedni sa drugima, sve dok se ne konfigurišu zone na optičkim komutatorima.

> Zoniranje optičke mreže (eng. *Fibre Channel zoning*) je između ostalog i sigurnosni mehanizam, koji se odnosi na podelu optičke mreže u manje podskupove, sa ciljem unapređivanja sigurnosti, smanjivanja interferencije u prenosu podataka i pojednostavljenja procesa upravljanja mrežom.

Zoniranjem se vrši podela bilo koje optičke mreže, pa tako i *SAN* mreže. Kada se vrši zoniranje *SAN* mreže određuje se kojim skladišnim volumenima svaki server posebno ima pristup. U klaud okruženju skladište podataka je obično centralizovano u vidu velikih skladišnih nizova i njemu pristupa veliki broj virtuelnih mašina putem *SAN* mreže. Pomoću zoniranja određuje se kojim skladišnim volumenima svaka virtuelna mašina ima pristup. Navedeni pristup prikazan je na Slici 9-16.

Slika 9-16: *SAN zoniranje*



Svaka zona je određena grupom portova, što se naziva teško zoniranje (eng. *hard zoning*), ili globalnim imenima, što se naziva meko zoniranje (eng. *soft zoning*). Više zona može da se grupiše u skup zona na *SAN* komutatoru i na taj način se aktivira cela grupa. Zoniranje se vrši isključivo u okviru mreže skladišta na optičkom komutatoru, ne na kranjim uređajima.

Zoniranjem se određuje kojim skladišnim resursima server može da pristupi na udaljenom skupu skladišta. U ovom kontekstu potrebno je da se pomene i termin broj logičke jedinice (eng. *Logical Unit Number – LUN*). *LUN* je broj koji se koristi za identifikaciju logičke jedinice na mreži skladišta podataka. U najvećem broju slučajeva logička jedinica predstavlja jedan skladišni uređaj.

U praktičnim implementacijama *SAN* klaud mreže, *Windows* serverima se omogućava pristup *Windows LUN blok* skladištima, dok se *Linux* sistemima dozvoljava pristup *Linux LUN* skladištima. Problemi mogu da nastanu ako se na primer *Linux* operativnom sistemu omogući pristup *Windows* fajl sistemu bez odgovarajuće softverske podrške, što vodi do toga da fajl sistem postane korumpiran.

Kao još jedna praktična implementacija zoniranja *SAN* mreže je ta što virtuelnoj mašini može da se omogući da se butuje preko *SAN* mreže.

Takođe je potrebno da se spomene i *LUN* maskiranje (eng. *LUN masking*). Ovaj proces je sličan zoniraju, s tom razlikom što se *LUN* maskiranje konfiguriše na nivou kontrolera skaldišta, dok se zoniranje određuje na nivou optičkog komutatora. *LUN* maskiranjem se definišu pristupna prava između virtuelnih mašina i *LUN* jedinica skladišta. Primenom *LU* maskiranja omogućava se granularnije definisanje pristupnih prava.

9.11. PRISTUP KLAUD SKLADIŠTU PODATAKA

U ovom poglavlju ukratko su navedeni načini za pristupanje resursima na klaud skladištu podataka.

Podacima na klaud skladištu može da se pristupi na različite načine. Tako na primer, korisnik može eksplicitno da upiše putanju do podataka koji se *hostuju* na klaud mrežnom *volumenu* korišćenjem standardnog konzolnog interfejsa, ili korisnik može da koristi intefejs virtulene mašine, *Web*¸ ili aplikacije baze podataka.

Pre nego što pristupi podacima na klaud skladištu, korisnik mora da se autentifikuje (eng. *authenticate*). Autentifikacija je proces određivanja identiteta korisnika. Autentifikacija se u većini slučajeva izvodi procesom logovanja. Autentifikacijom korisnika (eng. *user authentication*) proverava se identitet korisnika, na osnovu koga se dozvoljava, ili se nedozvoljava pristup određenom resursu klaud skladišta.

Autentifikacija korisnika se obično bazira na korišćenju korisničkog imena i lozinke u kombinaciji sa pristupnim tokenima ili biometrijskim metodama. U slučaju pristupa klaud skladištu preko *Web* intefejsa, najčešće se koriste kolačići (eng. *cookies*).

Serveri, bilo da su u pitanju fizički, ili virtuelizovani serveri, i aplikacije koje se *hostuju* na serverima koriste autentifikaciju kako bi se identifikovali jedni drugima. Aplikacije se najčešće izvršavaju u kontekstu servisnog naloga (eng. *service account*) i za potrebe autentifikacije koriste, ili *API*-je, ili neke od standardnih formata fajlova, kao što su *XML* (*eXtensible Markup Language*) i *JSON* (*Java Script Object Notation*).

Kao što je već više puta navedeno, za hostovanje klaud servisa uglavnom se koriste virtuelne mašine, koje mogu da imaju sopstvena (lokalna) skladišta podataka u vidu virtuelnih hard diskova, ili mogu da koriste udaljeno, na primer *SAN* klaud skladište. Bilo da virtuelne mašine koriste lokalna ili udaljena skladišta podataka, klijenti skladištima pristupaju na isti način.

Za potrebe individualnih korisnika, mnogi klaud provajderi nude specijalizovane klaud usluge kao što su *Google Drive, DropBox*, ili *OneDrive* servisi. Individualni korisnici ovim servisima pristupaju preko posebne *desktop* ili mobilne aplikacije, ili putem običnom *Web* brauzera. Ako se koristi aplikacija za pristup ovakvim skladištima, onda se vrši automatska sinhronizacija i replikacija fajlova koji se nalaze na klaud skladištima sa lokalnim kopijama uz mogućnost korišćenja na više različitih klijentskih uređaja.

PITANJA ZA RAZMATRANJE



1. Čemu služi računarska mreža skladišta *SAN*? Objasniti arhitekturu *SAN* mreže.
2. Navesti tipove klaud skladišta podataka.
3. Šta je *DAS* skladište podataka? Objasniti.
4. Šta je *NAS* skladište podataka? Objasniti.
5. Šta je *SAN* skladište podataka? Objasniti.
6. Objasniti objektni model za skladištenje podataka koji podatke prenose u formi faljova ili blokova podataka.
7. Šta je metapodatak?
8. Šta je *ID* objekta?
9. Šta znači pojam ravni fajlovi? Objasniti.
10. Objasniti proces dodavanja skladišta podataka u klaud računskom centru.
11. Objasniti šta znači debelo i tanko dodavanje klaud skladišta.
12. Šta su pristupni tokeni? Čemu služe? Objasniti.
13. Objasniti tri sloja klaud skladišta.
14. Objasniti pojam replikacije klaud podataka.
15. Šta je *RAID*?
16. Objasniti tehnologiju *RAID* 0.
17. Objasniti kako funkcionišu dve implementacije *RAID* 1 tehnologije: preslikavanje diskova i dupleksiranje diskova.
18. Navesti razlike između *RAID* 0 i *RAID* 5 tehnologije.
19. Šta je *RAID* 0+1? Objasniti.
20. Šta je *RAID* 1+0? Objasniti.
21. Objasniti *RAID* 5+0 tehnologiju.
22. Čemu služe pristupne kontrolne liste? Dati primer.
23. Šta se podrazumeva pod pojmom zoniranje optičke mreže? Objasniti kroz primer.

10. MIGRACIJA SERVERA NA KLAUD

S obzirom na to da se većina savremenih klaud okruženja baziraju na tehnologiji virtuelizacije, potrebno je biti obazriv prilikom migriranja postojećih servera i aplikacija koje se izvršavaju na svom namenskom, fizičkom hardveru u klaud svet koji se sastoji iz virtuelnih mašina i hipervizora.

Migracija fizičkog servera koji se nalazi u lokalnom korporativnom okruženju na virtuelni server na klaudu je samo jedna, ali najčešće korišćena vrste migracije servera na klaud. Kada se govori o migraciji servera na klaud, uglavnom se misli na ovaj tip migracije.

Međutim, postoje i druge vrste migracije, među kojima se najčešće koriste sledeće tri: migracija fizičkog servera u lokalnom okruženju na fizički server u klaud okruženju, migracija virtuelnog servera u lokalnom okruženju na virtuelni server u klaud okruženju i migracija sa virtuelnog servera u lokalnom okruženju na fizički server u klaud okruženju.

Kao napomena navodi se da se u ovom kontekstu pod serverom podrazumeva cela serverska platforma, tj. hardver sa operativnim sistemom i aplikacijama.

> Ključ uspeha migracije operativnih sistema sa fizičkog na virtuelni server „leži" u pažljivom, detaljnom i sveobuhvatnom planiranju svih aktivnosti migracije.

Pre migriranja servera na klaud potrebno je da se precizno definišu zahtevi koje virtuelni klaud server treba da ispunjava, kako bi se osiguralo da će i nakon migracije svi servisi koji su se izvršavali na fizičkom serveru uspešno da se izvršavaju i u klaud okruženju. U cilju optimalnog određivanja zahteva koje virtuelni server treba da ispunjava, potrebno je da se prikupe metrike i informacije o procesoru, memoriji, skladištu i *I/O* uređajima fizičkog servera koji se migrira. Prikupljanjem ovih informacija kreira se polazna osnova (eng. *baseline*) karakteristika koje virtuelni server treba da ima. Korišćenjem polazne osnove vrši se odgovarajuće skaliranje i konfigurisanje hardverskog profila novog virtuelnog servera.

Takođe, prilikom migriranja sa fizičkog na virtuelno okruženje, potrebno je uzeti u obzir i vremenski period tokom koga sistem neće raditi zbog izvršavanja procesa migracije (eng. *system downtime*). I u ovom kontekstu je važno precizno planiranje procesa migracije. Tokom procesa migracije menjaju se i odgovarajući sistemski parametri, kao što su *IP* adresiranje, *VLAN* i *VSAN* konfiguracije, *DNS* podešavanja i ostalo, pa i to treba uzeti u obzir.

Uobičajena je praksa da se kreira implementacioni dokument koji obuhvata sve korake koji se izvršavaju tokom migracije servera na klaud, kao i korake koje treba izvršiti u slučaju da je potrebno migraciju prekinuti i sistem vratiti na prvobitno stanje.

U implementacionom dokumentu je takođe potrebno navesti sve aktivnosti koje je potrebno preduzeti, kako bi se izvršila validacija procesa migracije, tj. kako bi se potvrdilo da se migracija sprovodi u skladu sa definisanim planom i očekivanjima.

10.1. VRSTE MIGRACIJE SERVERA

Kao što je već i navedeno, migriranje servera na klaud zahteva detaljno planiranje i pripremu za izvođenje ovih operacija. Potrebno je da se izvrši konverzija fizičkih servera u format koji je kompatibilan sa formatom koji koristi provajder klaud usluga. Većina klaud provajdera nudi mogućnost instalacije operativnih sistema direktno (neposredno) na fizički server (bez sloja hipervizora), kako bi se omogućilo izvršavanje aplikacija koje ne podržavaju tehnologiju virtuelizacije.

S rastom popularnosti koncepta klaud računarstva, terminologija i koncepti migracije servera su sazreli i proces migracije servera na klaud je dobro poznat zadatak zaposlenima u službi tehničke podrške klaud provajdera.

Postoji više tipova migracije servera na klaud, i kao četiri najčešće korišćena navode se: migracija sa fizičke mašine na virtuelnu mašinu, migracija sa virtuelne mašine na virtuelnu mašinu, migracija sa fizičke mašine na fizičku mašinu i migracija sa virtuelne mašine na fizičku mašinu.

Svaki od navedenih tipova migracije servera ukratko je opisan u nastavku.

10.1.1. Migracija sa fizičke mašine na virtuelnu mašinu

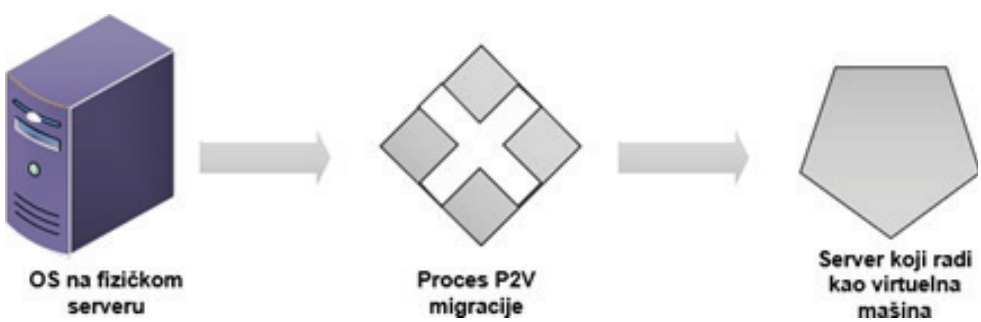
Prvi tip migracije servera na klaud je migracija sa fizičkog na virtuelno okruženje (eng. *physical-to-virtual – P2V*), koja se odnosi na scenarije kada se fizički server na kome se izvršava operativni sistem i aplikacije u lokalnom korporativnom okruženju migrira na virtuelnu mašinu na klaud, koja se izvršava iznad hipervizora.

Proces *P2V* migracije je vizuelizovan na Slici 10-1.

Nekoliko načina je dostupno za izvršavanje migracije tipa *P2V*. Prvi, neautomatizovani način, je kada se na virtuelnu mašinu na klaudu instalira sveža instanca operativnog sistema, na koju se zatim pomoću specijalnih alata migriraju aplikacije i podaci iz lokalnog na klaud okruženje.

Drugi, automatizovani način, odnosi se na korišćenje specijalnih softvera i alata za konverziju cele fizičke serverske platforme (operativni sistem plus aplikacije) u virtuelnu mašinu. Mnogi vendori klaud usluga nude ovakve alate i kao primeri navode se *VMware vCenter Converter* i *Microsoft Virtual Machine Manager*. Osim navedenih, na Internetu može da se pronađe i još dosta alata u ponudi trećih strana.

Slika 10-1: *Tip migracije P2V*



10.1.2. Migracija sa virtuelne mašine na virtuelnu mašinu

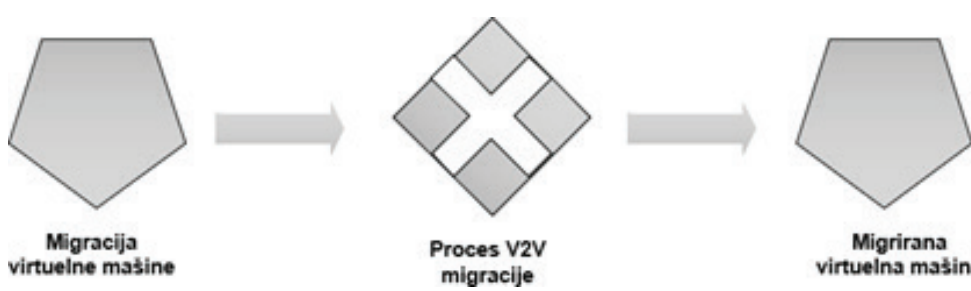
Drugi tip migracije servera je migracija sa virtuelne mašine koja se nalazi u lokalnom korporativnom okruženju na virtuelnu mašinu na klaudu (eng. *virtual-to-virtual – V2V*).

Ovaj tip migracije je mnogo jednostavniji od *P2V* migracije, jer se ona u većini slučajeva svodi na kloniranje postojeće virtuelne mašine u lokalnom okruženju i uvozu slike klonirane virtuelne mašine (eng. *virtual machine image import*) u virtuelizovano okruženje klaud provajdera.

Prilikom izvršavanja ovog tipa migracije, potrebno je uzeti u obzir vrstu hipervizora pod kojim se izvršava virtuelna mašina u lokalnom korporativnom okruženju. Dešava se da format lokalne virtuelne mašine nije kompatibilan sa formatom koji je podržan od strane hipervizora klaud provajdera. Na sreću, poslednjih godina ovaj scenario je sve ređi, zato što na tržištu klaud računarstva postoji velika kompatibilnost između platformi za virtuelizaciju različitih vendora.

Prikaz migracije servera sa virtuelne mašine u lokalnom okruženju na virtuelnu mašinu na klaudu dat je na Slici 10-2.

Slika 10-2: *Tip migracije V2V*



10.1.3. Migracija sa fizičke mašine na fizičku mašinu

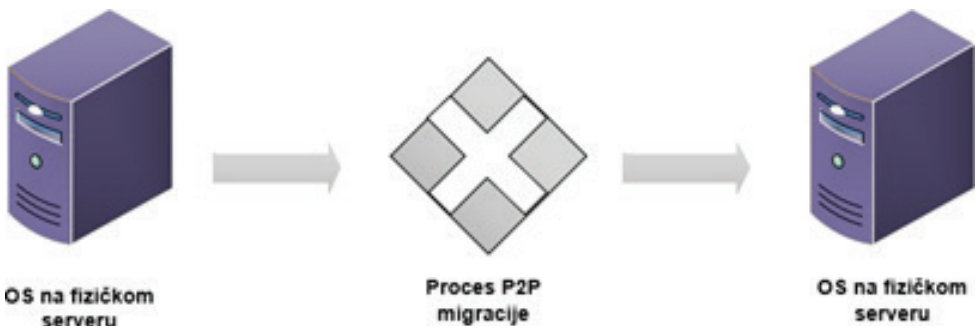
Migracija servera sa fizičke mašine u lokalnom okruženju na fizičku mašinu na klaudu (eng. *physical-to-physical – P2P*) koristi se onda kada korporacija hoće da izvršava svoje servise na namenskom fizičkom klaud serveru, koji se ne deli sa drugim klaud klijentima, ili kada određene poslovne aplikacije ne mogu da se izvršavaju na virtuelnom hardveru.

Ovaj tip migracije je skuplji od prethodna dva, zato što u ovom slučaju klaud provajder ceo fizički server namenjuje za ekskluzivno korišćenje samo jednom korisniku.

Pre izvršavanja ovakve migracije treba obezbediti da će fizički server na klaudu u najmanju ruku biti istih performansi kao i lokalni fizički server, jer samo na taj način može da se garantuje da će servisi koji su se izvršavali u lokalnom okruženju nastaviti da se izvršavaju i na klaudu uz isti nivo performansi i kvaliteta.

Prikaz *P2P* migracije dat je na Slici 10-3.

Slika 10-3: *Tip migracije P2P*



10.1.4. Migracija sa virtuelne mašine na fizičku mašinu

Bez obzira što se ređe koristi, postoji i četvrti tip migracije servera, kada virtuelni server u lokalnom okruženju konvertuje u fizički server na klaudu. Ovaj tip migracije se u literaturi naziva migracija sa virtuelne mašine na fizičku mašinu (eng. *virtual-to- physical – V2P*).

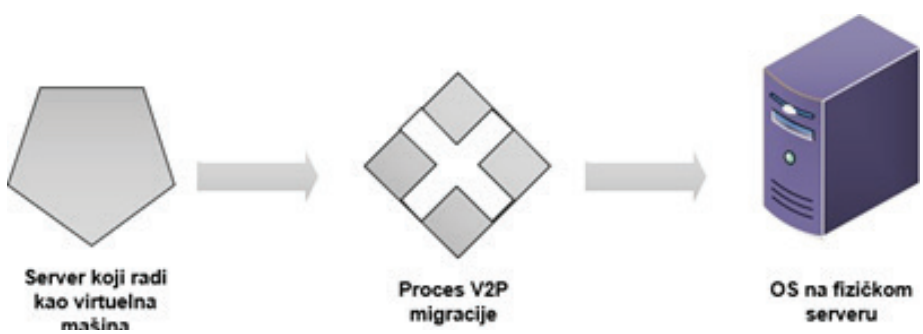
Na Slici 10-4, prikazana je *V2P* migracija.

Kao jedan od mogućih razloga zbog koga se korporacije odlučuju na ovaj tip migracije navode se situacije kada resursi virtuelne mašine nisu dovoljni za izvršavanje zahtevnih poslovnih aplikacija, i u tom slučaju je potrebno migrirati celo lokalno virtuelno okruženje na fizičku mašinu. Primer ovakvih aplikacija su zahtevne aplikacije baza podataka, za čije optimalno izvršavanje je potreban jak hardver, koji može da obradi i do nekoliko hiljada simultanih konekcija.

Migracija *V2P* je najzahtevniji tip migracije i pre njenog sprovođenja potrebno je detaljno proučiti hardverske i softverske zahteve operativnog sistema, aplikacija i podataka koji se migriraju.

U nekim slučajevima, kada nisu dostupni odgovarajući alati za konverziju, potrebno je instalirati svež operativni sistem i aplikacije. Zahtevi neće biti univerzalni, već će da zavise od slučaja do slučaja.

Slika 10-4: *Tip migracije V2P*



10.2. MIGRACIJA SKLADIŠTA PODATAKA

Kada se govori o migraciji servera na klaud, potrebno je da se spomene i migracija skladišta podataka.

> Serveri u lokalnom korporativnom okruženju često imaju skladišta podataka koja su na njih povezana ili lokalno, ili preko namenske mreže, poput *SAN*-a. Retki su slučajevi kada se na klaud migrira samo server bez skladišta podataka.

Serverski *volumeni* za skladištenje podataka mogu da se migriraju na klaud, ili zajedno sa serverom, ili posebno. Klaud provajder nudi više mogućnosti za migriranje skladišta. Skladišta tako mogu da se migriraju ili na lokalni disk virtuelne mašine, ili virtuelizovano *DAS* skladište, ili na namensku mrežu poput *SAN*-a.

Migracija podataka sa jednog na drugi skladišni uređaj je podržana i u situacijama oporavka od katastrofa. Klaud provajder će zajedno sa kompanijom klijentom da radi na prebacivanju podataka na klaud, a koji konkretan model će da se koristi zavisi od klaud infrastrukture i usluga koje klaud provajder nudi.

U procesu planiranja migracije skladišta podataka na klaud potrebno je uzeti u obzir propusnu snagu računarske mreže između prostorija korporacije i klaud provajdera, kao i vreme koje je potrebno da se izvrši otpremanje podataka (eng. *data upload*) sa lokalnog računskog centra na klaud. Ako na primer korporativni klijent poseduje sporu Internet konekciju, vreme otpremanja podataka na klaud biće relativno dugo i u tom slučaju je potrebno razmotriti alternativne načine za tansfer podataka do računskog centra klaud provajdera.

U situacijama kada kompanija ima veliku količinu podataka koje je potrebno otpremiti na klaud (reda veličine nekoliko petabajta), onda korišćenje Internet infrastrukture za migriranje podataka nije izvodljivo, zato što bi u okviru današnjih mogućnosti brzine Internet linkova, prošle godine, pre nego što bi se završio ovaj proces. Jedan od načina za prevazilaženje ovakve prepreke bio bi korišćenje specijalizovanih skladišnih uređaja (eng. *storage appliances*), koje kompanija klijent povezuje direktno u svoj lokalni računski centar i prebacuje podatke na njega. Nakon toga, uređaj se fizičkim putem dostavlja klaud provajderu i na isti način se podaci prebacuju u klaud računski centar. Napominje se da većina klaud provajdera ima razrađena rešenja za ovakve situacije.

Primer migracije skladišta podataka ilustrovano je na Slici 10-5.

Slika 10-5: *Migracija skladišta podataka na klaud*



10.3. ONLINE I OFFLINE MIGRACIJA SERVERA

Prema još jednom stanovištu, migracija servera na klaud može da se izvodi u *online*, ili *offline* režimu (modu).

> *Online* migracija servera je bolja alternativa, zato što će u ovom slučaju poslovni servisi biti operativni tokom većeg dela vremenskog perioda koji je potreban za izvršavanje migracije, što nije slučaj sa *offline* migracionim režimom.

Jedno ograničenja za izvršavanje online migracije je raspoloživi mrežni propusni opseg (eng. *network bandwidth*) između korporativnog računskog centra gde je postojeći server lociran i klaud računskog centra, na koji se migracija vrši.

U slučaju *offline* migracije, virtuelni server se prvo eksportuje na medijum za skladištenje, koji se zatim dostavlja klaud provajderu, gde će biti implementiran. Ovaj scenario migracije servera uključuje i određena zakašnjenja koja su izazvana vremenom neophodnim za fizički transport podataka.

10.4. FORMAT FAJLOVA VIRTUELNE MAŠINE I VIRTUELNOG DISKA

Kada se migrira server, bilo sa fizičke na virtuelnu mašinu, ili sa virtuelne na virtuelnu mašinu potrebno je uzeti u obzir formate fajlova virtuelne mašine i virtuelnog hard diska.

Korporacija klijent treba da se raspita o platformi za virtuelizaciju koju koristi klaud provajder, zato što svaka klaud platforma ima svoj određeni format virtuelnih mašina i virtuelnog hard diska. U slučaju migracije sa virtuelne mašine na virtuelnu, ovi formati treba da budu isti, u najmanju ruku kompatibilni, kako bi se migracija uspešno izvršila.

S druge strane, u slučaju migracije sa fizičke na virtuelnu mašinu treba videti da li aplikacije i operativni sistemi koji se izvršavaju na fizičkoj mašini podržavaju formate fajlova virtuelne mašine i virtuelnih diskova platforme za virtuelizaciju klaud provajdera. Dešava se da određene aplikacije podržavaju samo neke hipervizore.

Kao što je već i navedeno, na sreću, savremene platforme za virtuelizaciju su kompatibilne i raspoloživi su brojni alati za konverziju virtuelne mašine i virtuelnog diska iz jednog u drugi format. Osim toga, većina klaud provajdera podržava veći broj platformi za virtuelizaciju.

Standardni formati virutelnih diskova koje podržava većina klaud provajdera su *Oracle VDI* (*Oracle Virtual Disk Image*), koji koristi *Oracle VirtualBox* hipervizor, *VMDK* (*Virtual Machine Disk*), koje koriste hipervozori vendora *VMware, VHD* (*Virtual Hard Disk*) format kompanije *Microsoft* i *AMI* (*Amazon Machine Image*) koje koristi *Amazon*.

10.5. PORTABILNOST APLIKACIJA

Portabilnost aplikacija (eng. *application portability*) odnosi se na mogućnost premeštanja aplikacija sa infrastrukture jednog, na infrastrukturu drugog klaud provajdera bez potrebe konverzije formata.

Portabilnost aplikacija omogućava klijentima da relativno lako mogu svoje servise i aplikacije da migriraju sa klaud okruženja jednog na klaud okruženje drugog provajdera i izbegavaju se situacije kada klijent zavisi od jednog vendora aplikacija (na primer, kada aplikacija ima vlasničke dodatke), što onemogućava izvršavanje procesa migracije.

Važno je da se portabilnost aplikacija uzme u obzir prilikom izrade plana migracije, zato što može doći do situacija kada klaud provajder ne ispunjava sve klauzule koje su navedene u Sporazumu o nivou korišćenja usluga, ili kada klaud provajder trpi česte prekide u radu svojih sistema, kada je poželjno da organizacija potraži drugog provajdera.

Jedan od zahteva portabilnosti virtuelnih mašina i aplikacija odnosi se na poklapanje formata virtuelnih mašina starog i novog klaud provajdera, o čemu je bilo reči u prethodnom poglavlju. Ovo ide u prilog klaud provajderima koji koriste otvorene standarde, koji podržavaju nekoliko platformi za virtuelizaciju, ili koji nude alate za konverziju iz jednog u drugi format.

Portabilnost se u klaud svetu koristi kao opšti pojam koji obuhvata sve tipove migracije, kao što je migracija sa privatnog na javni klaud, migracija sa javnog na privatni, migracija sa zajedničkog u javni, ili bilo koju drugu kombinaciju.

10.6. DODATNA RAZMATRANJA PRILIKOM MIGRACIJE APLIKACIJA

Migracija aplikacija zahteva sveobuhvatna razmatranja.

Pre svega, kada se aplikacija migrira sa fizičkog na virtuelno klaud okruženje, potrebno je testirati da li će aplikacija moći da se izvršava na klaudu. Za ove potrebe poželjno je implementirati testno virtuelno okruženje, pre nego što se aplikacija migrira u produkciono okruženje na klaudu.

Takođe, prilikom testiranja aplikacija potrebno je izvršiti dodatne validacije, kao što su provera performansi, nivoa usluga, vreme rada, kompatibilnost i sigurnost aplikacija. U praksi skoro nikada neće da se desi situacija kada će svi od navedenih indikatora imati optimalne vrednosti i biće potrebno da se uspostavi balans (na primer malo se žrtvuje u performansama jednog indikatora zarad povećavanja performansi drugog indikatora).

10.7. INFRASTRUKTURA ZA SPROVOĐENJE MIGRACIJE

Pre izvršavanja migracije potrebno je detaljno da ispita računarska infrastruktura za izvršavanje migracije, kako bi se proaktivno reagovalo na potencijalne probleme.

Tako je na primer, potrebno da se ispitaju potencijalna zakašnjenja i zastoji u transferu podataka, vremenski period tokom kojeg korporativni sistemi i servisi neće raditi, kao i regulatorna i zakonodavna ograničenja.

10.7.1. Kapacite računarske mreže

Kapacitet mreže je već nekoliko puta bio spominjan, zbog toga što raspoloživa propusna snaga računarske mreže može da predstavlja ozbiljan ograničavajući faktor za izvršavanje procesa migracije.

Ako je Internet konekcija između kompanije klijenta i klaud provajdera spora i nepouzdana, potrebno je obezbediti dodatni propusni opseg za uspešno i pravovremeno izvršavanje migracije, ili je potrebno potražiti alternativna rešenja, kao što je *offline* migracija.

U fazi planiranja migracije projektni tim mora da utvrdi količinu podataka koja će biti prebačena migracijom, kao i vreme koje je potrebno za transfer podataka korišćenjem postojeće mrežne infrastrukture i mrežnog propusnog opsega.

10.7.2. Vreme prekida u radu sistema

U fazi planiranja procesa migracije takođe je potrebno da se utvrdi i vremenski period tokom kojeg korporativni sistemi i aplikacije neće biti operativni.

Na sreću, u modelu klaud računarstva moguće je instaliranje prototipa aplikacija, gde se prvo testira izvršavanje prototipova na klaudu. Tek kada je testiranje uspešno završeno, onda se aplikacija implementira u produkciono klaud okruženje. Na ovaj način se minimizuje vreme prekida u radu sistema.

Potrebno je da se napomene i da se u kvalitetno urađenim planovima migracije uvek uzima u obzir i dodatno vreme, kada proces migracije može da se neplanirano oduži zbog rizika i nepredviđenih okolnosti.

10.7.3. Optimalno vreme za izvršavanje migracije

Izbor optimalnog vremenskog perioda za izvršavanje migracije zavisi pre svega od korporativnih politika i zahteva. Migracija se najčešće sprovodi u vreme kada je opterećenost sistema mala, ili za vreme neradnih časova, kako bi se diskontinuitet u izvršavanju poslovnih operacija sveo na minumum.

10.7.4. Zakonodavna i legalna pitanja

Sva zakonodavna i legalna pitanja treba uzeti u obzir pre izvršavanja procesa migracije i ona bi trebala da budu integralni deo upravljanja projektom migracije. Potrebno je uzeti pitanja kao što su poreklo podataka, sigurnost podataka, osiguravanje da arhitektura sistema na klaudu zadovoljava sva zakonska i druga ograničenja, itd.

10.7.5. Lokalne vremenske zone

Ako se migracija izvršava između lokacija koje se nalaze u različitim vremenskim zonama, potrebno je preduzeti dodatne mere predostrožnosti, kako bi se izbegli dodatni prekidi u radu sistema za vreme najvećeg nivoa saobraćaja.

Tako na primer, ako se serveri, aplikacije i/ili podaci migriraju sa lokacije koja se nalazi na azijskom kontinentu, na neku lokaciju u Evropi, potrebno je odrediti vreme kada je intenzitet poslovnih aktivnosti u Aziji, za čiju su realizaciju potrebni servisi koji se migriraju, slabiji.

Plan migracije treba da obuhvati vremenske zone svih lokacija na kojima se nalazi opremama koja učestvuje u procesu migracije.

PITANJA ZA RAZMATRANJE



1. Objasniti proces migracije servera sa fizičke mašine na virtuelnu mašinu.
2. Objasniti proces migracije servera sa virtuelne mašine na virtuelnu mašinu.
3. Objasniti proces migracije servera sa fizičke mašine na fizičku mašinu.
4. Objasniti proces migracije servera sa virtuelne mašine na fizičku mašinu.
5. Objasniti proces migracije skladišta podataka na klaud.
6. Objasniti koncept *online* i *offline* migracije servera.

11. UPRAVLJANJE KORISNICIMA I INFRASTRUKTURNI SERVISI

Klaud je složeno i heterogeno okružnje sa velikim brojem sistema i resursa koje simultano koriste mnogobrojni korisnici koji pripadaju različitim korporativnim klijentima i pristupaju klaudu sa geografski udaljenih i distribuiranih lokacija. U takvim uslovima potrebno je pažljivo odrediti i razgraničiti koji korisnici će imati pristup kojim klaud resursima i pod kojim uslovima.

S druge strane, u složenom klaud okruženju koristi se veliki broj infrastrukturnih servisa, koji „daju život" i omogućavaju izvršavanje klaud operacija.

Zbog navedenog, prvi deo ove glave posvećen je upravljanju korisnicima na klaudu, dok je u drugom delu glave dat prikaz osnovnih infrastrukturnih klaud servisa.

11.1. *RBAC* MODEL

Kontrola pristupa koja se bazira na ulogama (eng. *role-based access control – RBAC*) je metod prema kojem se pristupna prava i dozvole dodeljuju, ili ograničavaju na osnovu uloga koje korisnici imaju u organizaciji.

RBAC model koristi definisane nivoe dozvola koje se dodeljuju rutinskim aktivnostima (ulogama) i omogućavaju, ili zabranjuju pristup klaud resursima u zavisnosti od uloge. Ovaj model ima široku primenu u klaud okruženjima.

Na osnovu dozvola koje uloge imaju, korisnici dobijaju pristupna prava (eng. *access rights*). Uloga može da bude bilo šta, što zavisi od potreba organizacije. Postoji mnogo standardnih uloga koje se definišu na različitim nivoima, kao što su nivo aplikacija, operativnih sistema, ili mrežne opreme. Kada se korisniku dodeli određena uloga, korisnik će imati prava i pristupe onim sistemima kojima može da pristupi uloga kojoj pripada.

U Tabeli 11-1 dat je primer nekih uloga sa njihovim opisom koje se najčešće koriste na klaudu.

Tabela 11-1: *Korisničke uloge na klaudu*

Naziv uloge	Opis uloge
Administrator (eng. *administrator*)	Korisnik supervisor/root koji ima potpunu kontrolu nad svim sistemskim resursima.
Gost (eng. *guest*)	Prava sa ograničenim pristupom, kao što je *read-only*.
Revizor (eng. *auditor*)	Prava praćenja sistema, kao što je praćenje nivoa potrošenih resursa, licenci aplikacija, usklađivanje sa regulatornim okruženjem, itd.
Izvršavanje nadogranje/zakrpa sistema (eng. *sustaining/patch managemenet*)	Prava koja omogućavaju nadogradnju, ažuriranje i implementiranje sistemskih i sigurnosnih zakrpa sistema.
Administrator skladišta (eng. *storage administrator*)	Prava za upravljanje i praćenje klaud skladišta podataka.

Prava (eng. *rights*) i dozvole (eng. *permission*) koje se korisnicima dodeljuju na osnovu uloge kojoj pripadaju treba da budu dovoljno široke, kako bi pokrile sve sisteme i usluge kojima korisnik treba da ima pristup za potrebe izvršavanja rutinskih aktivnosti, ali ne i previše široke, jer korisnik ne treba da ima pristup resursima i sistemima koji mu nisu potrebni u obavljanju delegiranih aktivnosti i operacija.

Na klaudu se široko primenjuje princip najmanje privilegije (eng. *principle of least privilege*), prema kojem je korisnik ovlašćen (autorizovan) za pristup samo onim računarskim resursima koji su mu potrebni da bi radio svoj posao.

Princip najmanje privilegije se primenjuje i na zaposlene u službi tehničke (*IT*) podrške klaud provajdera i na klaud klijente.

Opseg prava i obaveza koji se vezuje za određenu ulogu može da se definiše i na osnovu odgovornosti, kao što je odgovornost za upravljanje udaljenim lokacijama, administriranje mrežne infrastruktre ili *Linux* servera, administriranje baza podataka, itd.

U klaud okruženjima se široko koriste i korisničke grupe (eng. *user groups*). Mnogi javni klaud provajderi imaju svoje aplikacije za utvrđivanje identiteta korisnika koje se koriste za upravljanje pristupom klaud objektima, kao što su virtulene mašine, uređaji za balansiranje opterećenja, baze podataka i skladišta. Ove aplikacije kreiraju korisnike i ubacuju ih u grupe koje imaju određena prava nad određenim klaud resursima.

Grupe takođe mogu da se definišu i na nivou servera sa ciljem pojednostavljenja upravljačkih aktivnosti. Na primer, svi *Web* serveri mogu da se smeste u grupu pod nazivom *WWW*. Kada se zatim, korisnik koji ima upravljačka prava za grupu WWW, pomoću aplikacije za upravljanje identitetom, loguje na sistem, korisnik će moći da upravlja svim serverima koji se nalaze u *WWW* grupi.

11.2. AUTENTIFIKACIJA I AUTORIZACIJA

Autentifikacija je proces određivanja identiteta korisnika.

Autentifikacija (eng. *authentication*) se u većini slučajeva izvodi procesom logovanja. Autentifikacijom korisnika se proverava njegov identitet, gde se zatim na osnovu utvrđenog identiteta korisniku dozvoljava, ili zabranjuje pristup određenim klaud resursima.

Autentifikacija korisnika se obično bazira na korišćenju korisničkog imena i lozinke u kombinaciji sa pristupnim tokenima, ili biometrijskim metodama. Kolačići mogu da se koriste za Web pristup radi identifikacije i autentifikacije korisnika koji se povezuje na klaud preko Web interfejsa.

Serveri, bilo da su u pitanju fizički, ili virtuelni serveri, i aplikacije koje se *hostuju* na serverima takođe mogu da se autentifikuju jedni drugima. Aplikacije se najčešće izvršavaju u kontekstu servisnog naloga (eng. *service account*) i za potrebe autentifikacije koriste, ili *API*-je, ili neke od standardnih formata fajlova, kao što su *XML* (*eXtensible Markup Language*) i *JSON* (*Java Script Object Notation*).

Kada se korisnik ili uređaj identifikuju sistemom autentifikacije, izvršava se proces autorizacije (eng. *authorization*), kojim se određuje skup računarskih resursa koje korisnici mogu da koriste, kao i operacije (dozvole) koje korisnici mogu da vrše nad tim resursima.

Autorizacija je sigurnosni mehanizam koji određuje nivo pristupa i privilegije nad klaud računarskim resursima.

Tako na primer, kada se administrator klaud baza podataka autentifikuje, njemu se dodeljuje pristup serverima baza podataka sa punom kontrolom pristupa.

11.3. FEDERACIJE I *SSO* SISTEMI

Sistemi za utvrđivanje identiteta koriste federacije (eng. *federations*) kako bi omogućili pristup za više organizacija koje koriste iste podatke za identifikaciju.

Federacije se koriste na klaudu u onim slučajevima kada više korporativnih i/ili individualnih klijenata pristupaju istom klaud resursu, kao što je na primer instanca aplikacije.

U klaud okruženjima, gde više različitih organizacija koristi istu aplikaciju, sistem za utvrđivanje identiteta, koji se bazira na federaciji, omogućava svim klijentima da konsoliduju svoje resurse. U ovakvim situacijama korisnici iz različitih organizacija dele isti skup politika i pristupnih prava.

Pristup baziran na federacijama koristi industrijske standarde koji omogućavaju interoperabilnost između sistema različitih organizacija. Ovakav pristup se uglavnom koristi na klaud portalima za e-trgovinu. Kada se korisnik loguje na portal, bez potrebe za ponovnim logovanjem, korisnik će moći da izvrši i plaćanje preko portala banke, jer je portal za e-trgovinu povezan sa portalom banke pomoću sistema za upravljanje identitetom koji je baziran na federaciji.

Prisupi bazirani na federaciji koriste se i za druge vrste interakcija, kao što su interakcije na nivoima mašina-mašina i aplikacija-aplikacija.

U stručnoj literaturi često se mešaju pojmovi federacije i sistema za jedinstvenu prijavu (eng. *Single Sign On – SSO*). Glavna razlika između ova dva sistema ogleda se u tome što se *SSO* sistem koristi za potrebe samo jedne organizacije, i to samo za proces autentifikacije, dok se federacije koriste za više organizacija.

U praksi je čest slučaj da se autentifikacija na različite sisteme jedne organizacije na klaudu objedinjuje primenom *SSO* sistema.

SSO je pristup koji omogućava korisniku da se loguje samo jednom i da dobije pristup resursima različitih sistema. Pomoću ovog sistema vrši se centralizacija autentifikacije na više različitih sistema, što znatno poboljšava, kako iskustvo korisnika (eng. *user experience*) tokom eksploatacije sistema, tako i sam proces administriranja sistemom.

Tako na primer, ako jedan administrator treba da upravlja sa više *Web* servera, svi *Web* serveri mogu da se stave u jednu *SSO* grupu pod nazivom "*Web* administrator". U tom slučaju, kada se administrator loguje na jedan *Web* server, automatski će da dobije pristup i svim drugim *Web* serverima u grupi, bez potrebe za posebnim logovanjem na svaki *Web* server.

Servisi direktorijuma (eng. *directory services*) koji koriste *LDAP* (*Lightweight Directory Access Protocol*) su dobro poznati primeri implementacije *SSO* sistema, koji su našli široku primenu i na klaudu. Kada se korisnik jednom loguje na servis direktorijuma, korisnik dobija pristup svim resursima koji su članovi tog servisa.

Osim navedenog, *SSO* sistem ima i brojne druge prednosti. Tako na primer, u uslovima postojanja *SSO* sistema, potreba da korisnik pamti različite kredencijale (korisničko ime i šifra) na logovanje na različite sisteme je eliminisana. Ovaj sistem takođe pruža prednosti i prilikom prekidanja jedne sesije. Kada korisnik prekine sesiju (izloguje se sa sistema), korisnik će automatski biti izlogovan i sa svih ostalih sistema organizacije.

11.4. INFRASTRUKTURNI KLAUD SERVISI

Bez obzira što je ranije već bilo reči o nekim infrastrukturnim klaud servisima, s obzirom na veliku ulogu koju ovi servisi igraju u klaud okruženju, oni su u ovom delu udžbenika ponovo adresirani.

Infrastrukturni klaud servisi predstavljaju „srce i mozak" klaud infrastrukture, jer oni omogućavaju izvršavanje svih klaud aktivnosti i operacija i povezuju sve klaud računarske komponente u jednu zaokruženju, integralnu celinu.

U nastavku su ukratko prikazani sledeći klaud infrastrukturni servisi: *DNS* servis, *DHCP* servis, servisi za digitalne sertifikate i balansiranje opterećenja, servisi za autentifikaciju pomoću više faktora sigurnosti i *firewall*.

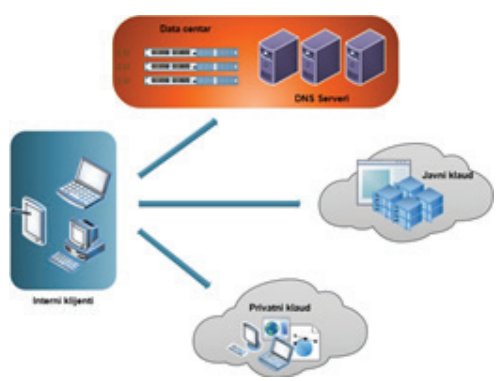
11.4.1. *DNS* servis

DNS servisi se koriste za razrešavanje imena u *IP* adresu, koju koristi *IP* protokol za povezivanje sa udaljenim uređajem, serverom ili bilo kojim uređajem.

Kada klijent pošalje zahtev *DNS* serveru, server klijentu šalje odgovor sa *IP* adresom resursa koju je klijent zahtevao. *DNS* koristi dobro poznati port 53 i transporni *UDP* protokol.

Klaud *DNS* servis je prikazan na Slici 11-1.

Slika 11-1: *Klaud DNS servis*



11.4.2. DHCP servis

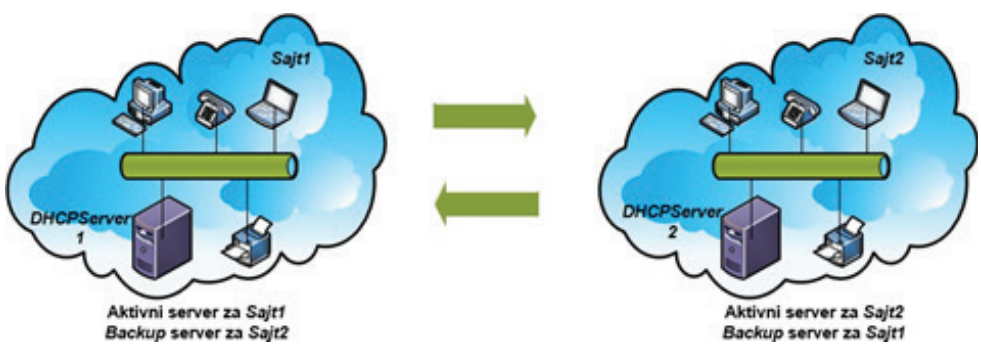
DHCP (eng. *Dynamic Host Configuration Protocol*) je servis koji automatski dodeljuje *IP* mrežne parametre klaud uređajima.

Svaki klaud mrežni uređaj koji se priključi na klaud mrežu automatski od *DHCP* servera dobija informacije o *IP* adresi, masci podmreže, podrazumevanom mrežnom prolazu i adresi klaud *DNS* servera.

DHCP server koristi *UDP* port 67, dok klijent (klaud mrežni uređaj) koristi *UDP* port 68. S obzirom da je u toku tranzicija sa *IPv4* na *IPv6* adresiranje Internet resursa, većina klaud provajdera podržava i *DHCPv6*. U *DHCPv6* implementaciji, server koristi *UDP* port 547, dok klijent koristi *UDP* port 546.

Dva klaud sajta, gde svaki sajt koristi svoj *DHCP* server, prikazana su na Slici 11-2.

Slika 11-2: *Klaud DHCP servis*



11.4.3. Servisi za sertifikate

Mnogi klaud provajderi u okviru široke lepeze svojih usluga nude i usluge kreiranja, upravljanja i implementiranja digitalnih sigurnosnih sertifikata, kao što su *SSL*, javni/privatni, i sertifikati bazirani na tokenima.

Sistemi za upravljanje sertifikatima automatski rotiraju ključeve, ažuriraju servere i sisteme za balansiranje opterećenja i obezbeđuju pouzdano skladištenje privatnih ključeva.

11.4.4. Servisi za balansiranje opterećenja

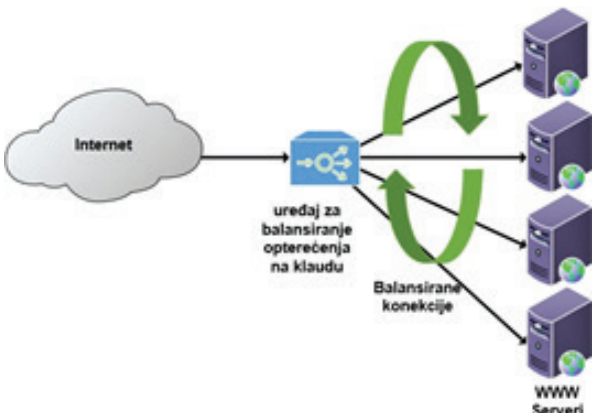
Servisi za balansiranje opterećenja (eng. *load balancers*) imaju ulogu u situacijama kada intenzitet mrežnog saobraćaja i opterećenje prekorače kapacitet jednog servera, koji više ne može efikasno da upravlja svim zahtevima klijenata, ili *Web, DNS, FTP* servera, *firewall*-ova, ili drugih mrežnih uređaja.

Primenom ove tehnologije, većina klaud provajdera konfiguriše klaud servere da rade zajedno, kao grupa, u formi klastera, i pritom dele teret i raspoređuju procesiranje na onog člana klastera koji je trenutno najmanje opterećen. Na ovaj način ceo sistem postaje redudantniji, elastičniji i skalabilniji.

Uređaji za balansiranje opterećenja se često implementiraju za potrebe obrade saobraćaja ka klaud *Web* serverima. U ovakvoj mrežnoj topologiji, uređaj za balansiranje opterećenja se postavlja ispred grupe *Web* servera i *IP* adresa adresa ovog uređaja se putem *DNS* servisa objavljuje kao adresa *Web* servera. Kada klijenti hoće da pristupe sajtu koji se nalazi na ovako konfigurisanim Web serverima, klijenti zapravo pristupaju pomoću *IP* adrese uređaja za balansiranje opterećenja. Kada uređaj za balansiranje opterećenja prihvati konekciju klijenta, on analizira opterećenje *Web* servera koji se nalaze iza njega i zahtev prosleđuje onom serveru koji je u datom vremenskom trenutku najmanje opterećen. U ovakvim implementacijama uređaj za balansiranje opterećenja takođe i analizira stanje (eng. *health analysis*) svakog *Web* servera u grupi i uklanja *Web* servere koji softverske ili hardverske probleme.

Prikaz implementacije uređaja za balansiranje opterećenja na klaudu dat je na Slici 11-3.

Slika 11-3: *Sistemi za balansiranje opterećenja na klaudu*



11.4.5. Servis za autentifikaciju pomoću više faktora sigurnosti

Servisi za autentifikaciju pomoću više faktora sigurnosti pružaju dodatni nivo zaštite sistemima za autentifikaciju na klaudu.

Ovi servisi standardnim sistemima za autentifikaciju uz pomoć jednog faktora (tipično korisničko ime i lozinka), implementiraju dodatni sloj zaštite, tako što se za proces autentifikacije koriste i druge komponente, kao što su pristupni token, ili biometrijske karakteristike korisnika.

Detaljniji pregled ovih sistema dat je u delu udžbenika koji se odnosi na bezbednost klaud okruženja.

11.4.6. *Firewall* servisi

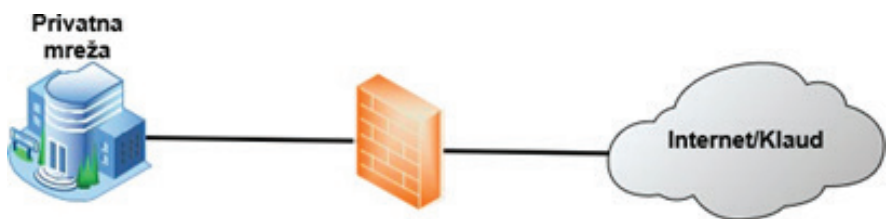
Firewall servisi su jedna od ključnih komponenti bezbednosti svakog klaud okruženja. *Firewall* uređaji se obično implementiraju između klaud računarske mreže i klaud klijenta, kako bi se onemogućio neautorizovani pristup računarskim resursima na klaudu.

> Klaud *firewall* je fizički ili virtuelizovani uređaj koji analizira mrežni saobraćaj i upoređuje obrasce analiziranog saobraćaja sa predefinisanom listom bezbednosnih pravila, na osnovu čega se određeni mrežni saobraćaj dozvoljava, ili se blokira.

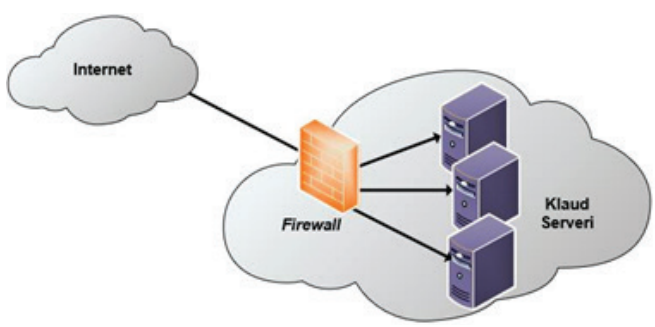
Mrežni saobraćaj koji nije dozvoljen se blokira i ne dolazi do interne klaud mreže. S druge strane, klaud firewall servisi mogu da se konfigurišu tako da vrše analizu i izlaznog saobraćaja (saobraćaja čiji je tok od klaud okrženja ka klaud klijentu ili Internetu).

Mnogi *firewall* uređaji mogu da funkcionišu i kao *VPN* serveri. Prikaz *firewall* servisa na klaudu dat je na slikama 11-4 i 11-5.

Slika 11-4: *Klaud firewall između javnog Interneta i privatne korporativne mreže*



Slika 11-5: *Klaud firewall između javnog Interneta i klaud mreže*





PITANJA ZA RAZMATRANJE

1. Objasniti pojam kontrole pristupa koji se bazira na ulogama.
2. Objasniti tipove korisničkih uloga na klaudu.
3. Objasniti princip najmanje privilegije i nivoe autorizacije.
4. Koja je razlika između autentifikacije i autorizacije? Dati primer.
5. Šta je *DNS*? Dati primer klaud *DNS* servisa.
6. Šta je *DHCP*? Dati primer.
7. Šta je servis za balansiranje opterećenja? Dati primer.
8. Čemu služi klaud *firewall*? Objasniti kroz primer.

12. OPORAVAK OD KATASTROFE I KONTINUITET POSLOVANJA

U složenom okruženju kao što je klaud može da dođe do grešaka i do otkaza opreme, kako zbog ljudskog faktora, tako i zbog sistemskih otkaza, i prirodnih činilaca. Zbog toga je važno da se unapred postave sistemi za proaktivno reagovanje i da se unapred planiraju koraci koji će biti preduzeti u slučaju da dođe do prekida u radu celog, ili delova sistema, klaud infrastrukture, ili bilo kog klaud resursa.

U svakom klaud okruženju potrebno je da se uspostave mehanizmi za oporavak od katastrofe (eng. *disaster recovery*) sa ciljem da se obezbedi kontinuitet poslovanja i obavljanja poslovnih aktivnosti (eng. *business continuity*).

Oporavak od katastrofe odnosi se na skup politika, alata i procedura koji omogućavaju brz oporavak klaud sistema i vitalne klaud infrastrukture u slučaju prirodne, ili katastrofe prouzrokovane ljudskim faktorom.

Mehanizmi za oporavak od katastrofe na klaudu obuhvataju *IT* sisteme koji podržavaju izvršavanje ključnih poslovnih aktivnosti. Na Slici 12-1 prikazana je implementacija jednog mehanizma za oporavak od katastrofe.

Kontinuitet poslovanja se odnosi na mogućnost organizacije da izvršava ključne poslovne operacije tokom i nakon što se katastrofa dogodi.

Planiranje aktivnosti u slučaju otkaza računarskih resursa je relativno lako. Tako na primer, kao mera predostrožnosti u slučaju otkaza *Web* servera, dovoljno je da se implementira nekoliko servera iza uređaja za balansiranje opterećenja. Zahvaljujući primeni tehnologije virtuelizacije koja se koristi u klaud računskim centrima, virtuelne mašine, na kojima se izvršava proces *Web* servera, mogu da se zamene u slučaju otkaza, kada se sistem migrira na drugu platformu. Međutim, ako ceo klaud računski centar otkaže, složenost procesa oporavka eksponencijalno raste.

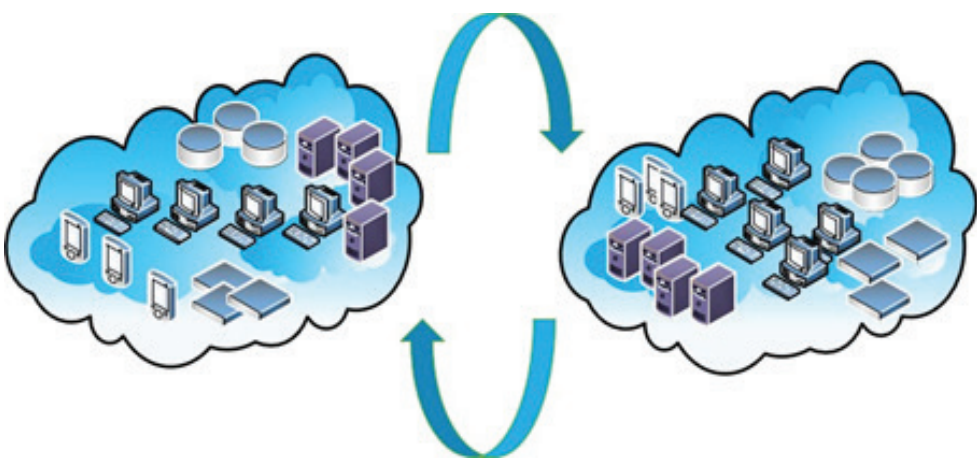
U klaud okruženju, klaud provajder *hostuje* infrastrukturu koja je otporna na otkaze u opremi i na prekide u izvršavanju servisa. U klaud računskim centrima koriste se redudantni sistemi napajanja, hlađenja, mreža, itd., kako bi se smanjila verovatnoća od potpunog prekida u radu sistema i računarske opreme, i kako bi se omogućio brz oporavak kritičnih sistema u nepredviđenim situacijama.

Međutim, čak i kada koristi usluge klaud provajdera, korporativni klijent snosi punu odgovornost za planiranje u slučaju prekida rada sistema u klaud okruženju. Događaji kao što su prirodne nepogode, loši vremenski uslovi, poplave, zemljotresi, i slično, mogu da dovedu do prekida električnog napajanja i problema sa komunikacijama u klaud računskom centru. Takođe, i drugi događaji, poput otkaza opreme, *cyber* napadi, virusi, diskontinuiteti u poslovanju dobavljača kritične računarske opreme, štrajk radnika, itd., mogu da impliciraju prekide u radu sistema.

U prvom delu ove glave udžbenika prikazani su načini na koje korporativni klijent i klaud provajder mogu da se pripreme za eventualne katastrofe i prikazane su arhitekture i referentni dizajni koji su raspoloživi za implementiranje mehanizama za oporavak od katastrofe.

U drugom delu glave prikazan je koncept kontinutiteta poslovnih operacija, koji omogućava brz oporavak poslovnih operacija u slučaju prekida rada klaud servisa.

Slika 12-1: *Implementacija jednog mehanizma za oporavak od katastrofe*



12.1. ODGOVORNOSTI I MOGUĆNOSTI PROVAJDERA KLAUD USLUGE

Odgovornosti i mogućnosti provajdera klaud usluge definisane se u Sporazumu o nivou korišćenja usluga.

U jednom takvom sporazumu koriste se pojmovi koji predstavljaju osnovnu terminologiju mehanizama za oporavak od katastrofe i koji su navedeni u nastavku.

12.1.1 *RPO* i *RTO*

Dva akronima na engleskom jeziku na koja se često nailazi kada se govori o oporavku od katastrofe su RPO i RTO.

Cilj tačke oporavka (eng. *recovery point objective – RPO*) je stanje sistema na koje se sistem vraća nakon izvršavanja procesa oporavka sistema.

U suštini, *RPO* je indikator koji određuje količinu podataka koja može biti izgubljena u slučaju katastrofe. Kao primer može da se uzme organizacija koja u bazi podataka čuva sve transakcije svog portala za elektronsku trgovinu. Neka se kreiranje rezervnih kopija takve baze izvršava na svaka dva sata. U ovom slučaju, *RPO* bi bio najviše dva sata.

Kada bi se na primer desila prirodna nepogoda, gde bi cela lokacija gde se nalaze sistemi organizacije bila uništena, organizacija bi svoje operacije prebacila na rezervnu lokaciju i u ovom slučaju *RPO* bi se koristio za merenje ažurnosti podataka na novoj lokaciji. U opštem slučaju, za sisteme koji čuvaju kritične podatke, kao što su podaci o bankovnim transakcijama, ili zdravstveni podaci o klijentima, vrednost *RPO*-a treba da bude bliska nuli.

U procesu planiranja kontinuiteta poslovanja, *RPO* igra ključnu ulogu u dizajnu arhitekture klaud računarstva.

Cilj vremena oporavka (eng. *recovery time objective – RTO*) je maksimalni vremenski interval tokom kojeg sistem može da bude *offline* u slučaju da se dogodi katastrofa.

Drugim rečima, *RTO* indikator je vreme koje potrebno da protekne, kako bi se sistem vratio u stanje normalnog funkcionisanja nakon kvara, ili otkaza resursa u slučaju prirodne, ili neke druge katastrofe.

Prilikom planiranja kontinuiteta poslovanja, vreme zaustavljanja sistema mora da se uzme u obzir, jer ono može da ima značajan uticaj na profit kompanije. Tako na primer, ako se radi o kompaniji koja se bavi e-trgovinom, prekid sistema u vremenskim periodima oko praznika, kada se izvršava veliki broj transakcija, može da predstavlja veliki trošak za organizaciju. U ovom slučaju, portal za e-trgovinu je kritična aplikacija i vrednost *RTO* indikatora mora da bude mala. S druge strane, ako su u pitanju sistemi koji nisu kritični, vrednost *RTO*-a može da bude relativno velika, pa čak i da bude izražena u danima.

12.1.2. Politike i smernice korporativnog klijenta klaud usluga

Računarski resursi igraju ključnu ulogu u poslovanju savremenih kompanija. Mnoge organizacije ne mogu ni da funkcionišu u uslovima otkaza računarskih resursa.

Kao što je slučaj i sa mnogim drugim procesima u kompanijama, planovi za oporavak od katastrofe i kontinuitet poslovanja treba da budu razvijeni i odobreni za korišćenje u okviru kompanije.

Ovi planovi su mnogo više nego samo tehnički dokumenti, jer imaju direktan i neposredan uticaj na sve poslovne operacije kompanije.

Izvršni i upravni odbori kompanija treba vrednosti *RPO* i *RTO* indikatora da uzmu u obzir s velikom dozom ozbiljnosti, i da im posluže kao osnova za izbor modela za implementaciju mehanizma za oporavak od katastrofe. Drugim rečima, treba da se poštuje jedna stara izreka "kada ne uspeš da se pripremiš, spremaš se za neuspeh" (eng. *when you fail to prepare you prepare to fail*).

Prvi korak u implementaciji mehanizma za oporavak od katastrofe je evaluacija potreba organizacije. Na osnovu identifikovanih potreba bira se odgovarajući provajder klaud usluga koji može da zadovolji te potrebe. Nakon toga, pristupa se implementaciji efektivnog plana za oporavak od katastrofe, u okviru definisanih zahteva i budžeta organizacije.

Nakon implementacije plana za oporavak od katastrofe, potrebno je izvršiti detaljno testiranje. Testiranjem se identifikuju zahtevi i ograničenja koja su ranije bila prevideta i vrši se modifikacija ovog plana. Takođe, s vremena na vreme potrebno je izvršiti reviziju plana, zato što se vremenom zahtevi i potrebe menjaju.

12.1.3. Politike i smernice provajdera klaud usluga

Većina provajdera klaud usluga ima sisteme za detaljno planiranje oporavka od katastrofe i podržava više arhitektura za implementiranje ovog mehanizma. Provajderi moraju da se pripreme za svaku situaciju, kako bi svojim klijentima pružili najbolju moguću uslugu i time bili konkurentni na klaud tržištu.

S obzirom da je klaud računarstvo tek od skoro počelo da se razvija i da je od početka korišćena relativno nova hardverska i softverka infrastruktura, računarska mreža skoro svih vodećih provajdera klaud usluga u svom dizajnu podržava mehanizme za opstanak i za oporavak od katastrofe. Ovi mehanizmi se navode u zvaničnoj ponudi dobavljača klaud usluga.

Kada organizacija priprema svoj plan za oporavak od katastrofe i kontinuitet poslovanja, detaljno se sagledavaju ograničenja i sposobnosti klaud provajdera, što je sve precizno definisano u dokumentima sa saglasnostima, politikama i smernicama koji se potpisuju prilikom sklapanja Sporazuma o nivou korišćenja usluga.

12.1.4. Računarska mreža i oporavak od katastrofe

Kada se dogodi katastrofa, računarska mreža ima suštinski značaj u aktivnostima oporavka sistema i ponovnom uspostavljanju poslovnih operacija.

> Širokopojasna mreža klaud provajdera i njena povezanost sa drugim mrežama ima značajan uticaj u izradi plana za oporavak od katastrofe.

Ako se u slučaju katastrofe koristi rezervna lokacija (rezervni računski centar), ova lokacija treba da podržava isti, ili veći obim mrežnog saobraćaja, kao i primarna lokacija koja privremeno, zbog katastrofe ne funkcioniše. Takođe, za vreme katastrofe, rezervni sajt treba da podrži i dodatni mrežni kapacitet za transfer podataka i kreiranje rezervnih kopija, što se postiže primenom mehanizama redudantnosti i replikacije. Jedan od načina za rešavanje ovih pitanja je korišćenje arhitekture kvaliteta servisa (eng. *Quality of Service – QoS*), koja vrši prioritizaciju mrežnog saobraćaja na osnovu tipa saobraćaja.

Takođe, u slučaju katastrofe potrebno je obezbediti rezervne mrežne servise poput *DNS*-a, *DHCP*-a, *FTP*-a, *RADUS*-a (*Remote Autehntification Dial-in User Service – RADIUS*) i *TACACS*-a (*Terminal Access Controller Access-Control System*) servera, itd.

Veliki broj vodećih klaud provajdera koristi čak redudantne mrežne servise koji se sastoje iz većeg broja računskih centara, koji su povezani privatnim superbrzim optičkim konekcijama. Ovakvi servisi se u literaturi nazivaju zone dostupnosti (eng. *availability zones*).

12.1.5. Ograničenja na strani klaud provajdera

Kao potencijalni problem u kreiranju i implementiranju plana za oporavak od katastrofe, između ostalog, navode se i ograničenja na strani provajdera klaud usluga.

Prilikom izbora klaud provajdera kompanija treba da postavi sledeća pitanja:

- da li klaud provajder ima računarske resurse koji mogu efektivno da zadovolje potrebe kompanije po pitanju zahteva za oporavak od katastrofe (servisi, zadovoljavajući propusni opseg mreže, itd.)?
- da li klaud provajder dodatno naplaćuje mrežni saobraćaj na mesečnom nivou, ako intenzitet saobraćaja pređe preko predefinisanog nivoa?
- da li klaud provajder ima dovoljno mrežnih resursa, koji mogu da podrže povećani intenzitet mrežnog saobraćaja u slučaju katastrofe, kada je potrebno u što kraćem roku da se izvrši transfer velike količine podataka?

12.2. MODELI I TEHNIKE ZA OPORAVAK OD KATASTROFE

Zbog svoje prirode resursa na zahtev, otpornosti, elastičnosti i modela plaćanja na osnovu nivoa ostvarene potrošnje, korišćenjem servisa javnog klauda omogućava se implementacija troškovno efektivnog plana za oporavak od katastrofe.

Infrastruktura javnih klaud servisa podržava visoku dostupnost računarskih resursa i najbolje može da zadovolji potrebe organzacije koje se odnose na oporavak od katastrofe.

Zahvaljujući prirodi servisa javnog klauda čak i male kompanije mogu da implementiraju sofisticirani plan za oporavak od katastrofe uz relativno niske troškove.

Ako na primer glavni sajt kompanije postane nedostupan zbog prirodnih nepogoda, kao što su poplave, zemljotresi i slično, servisi javnog klauda mogu da imaju funkciju privremenog korporativnog računskog centra.

U velikom broju slučajeva, na nivou sistema javnog klauda implementirani su mehanizmi za izradu rezervnih kopija čitavog sajta, koji su automatizovani i pod punom kontrolom klaud provajdera.

Kako u literaturi, tako i u praksi postoje brojni modeli i tehnike za oporavak od katastrofe i u nastavku su prikazani neki od njih.

12.2.1. Kreiranje ogledala sajta

Kreiranje ogledala (kopije, odraza) sajta (eng. *site mirroring*) je proces kojim se implementira sekundarni sajt, kao kopija primarnog (glavnog) sajta, koji se redovno ažurira, kako bi mogao da preuzme teret sa glavnog sajta u svakom vremenskom trenutku, u slučaju kada zbog katastrofe primarni (glavni) sajt postane nedostupan.

Kreiranjem odraza sajta kreiraju se identične kopije podataka i aplikacija glavnog sajta koje se nalaze na udaljenom sajtu koji „čeka" u stanju pripravnosti. Primenom ove proaktivne strategije, organizacija se priprema za nepredviđene događaje i minimizuje se verovatnoća da će ključni servisi biti nedostupni u slučaju nepredviđenih događaja.

U literaturi se najčešće navode tri načina (strategije) za implementiranje ogledala sajta, i to: vruć sajt, mlak sajt i hladan sajt, koji su u nastavku prikazani.

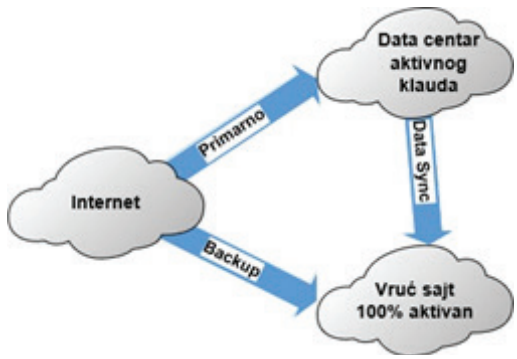
Vruć sajt

Prvi način za implementiranje strategije odraza sajta je model vrućeg sajta (eng. *hot site*), koji koristi dva redudantna klaud računska centra, koji su potpuno sinhronizovani jedan s drugim, tako što se sa glavnog sajta vrši backup podataka, aplikacija i servisa na rezervni sajt u realnom vremenu.

Model vrućeg sajta nudi najviši nivo redudantnosti i pouzdanosti. Međutim, ovaj model predstavlja i najskuplju opciju i u najvećem broju slučajeva koristi se samo za kritične servise i resurse.

Model vrućeg sajta prikazan je na Slici 12-2.

Slika 12-2: *Model vrućeg sajta*



Mlak sajt

Drugi način za implementiranje strategije ogledala sajta je pristup mlakog sajta (eng. *warm site*). Prema ovom modelu *backup* sajt je *offline*, osim u slučaju kada se na njemu čuva kritično skladište podataka, kao što je baza podataka.

U ovom slučaju, na toplom sajtu se nalazi operativna, online baza podataka koja se sinhronizuje u realnom vremenu sa bazom podataka na glavnom sajtu, dok su svi drugi klaud računarski servisi *offline*. Ovakva arhitektura se ponekada u literaturi naziva arhitektura svetla od sveće (eng. *candlelight design*).

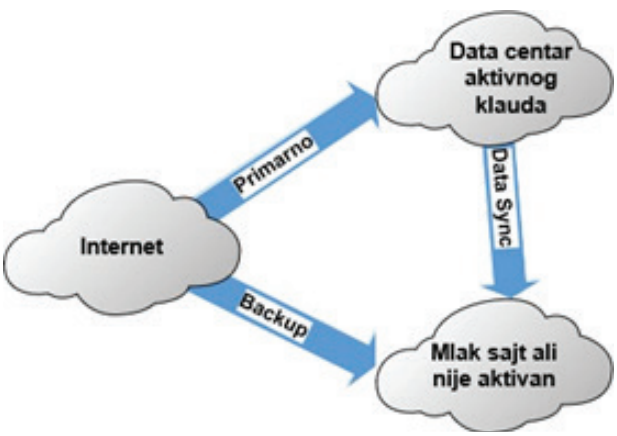
Prema modelu mlakog sajta, svi drugi infrastrukturni resursi i servisi, poput servera, skladišta podataka, uređaja za balansiranje opterećenja, mrežnih komponenti, itd., nisu u funkciji dok se za to ne javi potreba. U slučajevima katastrofa, potrebno je određeno vreme kako bi *backup* sajt postao funkcionalan.

Dakle, model mlakog sajta je sličan modelu vrućeg sajta. Razlike se pre svega odnose u vremenskom intervalu koji je potreban da prođe kako bi *backup* sajt postao funkcionalan i na troškovnu efikasnost, zato što su troškovi implementiranja mlakog sajta značajno manji nego što je to slučaj sa implementacionim troškovima vrućeg sajta.

Metrika *RTO* je u slučaju mlakog sajta znatno niža od modela hladnog sajta, ali je zato viša nego što je to slučaj sa modelom vrućeg sajta. Takođe, potrebno je da se napomene da je u slučaju katastrofe, kada se koristi model mlakog sajta, gubitak nekog dela podataka neizbežan.

Mnogi eksperti klaud računarstva smatraju da je model mlakog sajta dobar kompromis između pristupa vrućeg i hladnog sajta, koji je prikazan u nastavku. Arhitektura mlakog sajta data je na Slici 12-3.

Slika 12-3: *Arhitektura mlakog sajta*



Hladan sajt

U modelu hladnog sajta (eng. *cold site*) postoji *backup* sajt, ali on nije funkcionalan, niti se na njemu nalaze ažurirani podaci i aplikacije.

Backup sajt nema sve infrastrukturne računarske resurse, kao što serveri, skladišta podataka, mrežni uređaji i slično i zbog toga je u slučaju otkaza glavnog sajta potrebno da protekne dosta vremena pre nego što *backup* sajt postane funkcionalan i pre nego što može da preuzime izvršavanje poslovnih operacija.

Kao što se vidi sa Slike 12-4, hladan sajt nema instaliranu opremu, niti mehanzime za replikaciju podataka. Međutim, hladan sajt može da bude dobro rešenje u situacijama kada računarski resursi otkažu na dug vremenski period i predstavlja značajno jeftinije rešenje od modela vrućeg i mlakog sajta.

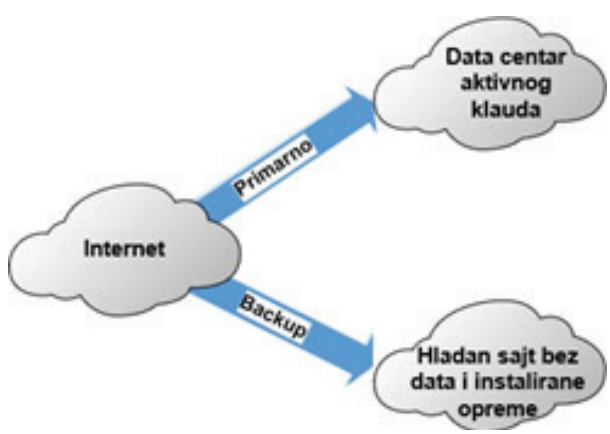
Ako se desi katastrofa, prvo je potrebno da se oprema na hladnom sajtu implementira, da se hladan sajt podigne u online režim rada, a zatim da se prebace svi podaci i aplikacije sa glavnog sajta. Alati za automatizaciju koji su dostupni na strani provajdera klaud usluge omogućavaju bržu replikaciju arhitekture primarnog sajta na *backup* sajt zajedno sa svim uređajima kao što su serveri, skladišta podataka, mreže, itd.

Tokom izrade plana kontinuiteta poslovanja, mora da se uzme u obzir vreme koje je potrebno da protekne pre nego što hladan sajt postane funkcionalan. Zbog relativno dugog vremenskog perioda, osposobljavanja koncept hladnog sajta ne zadovoljava potrebe ključnih poslovnih resursa, kao što je na primer server baze podataka.

Tako na primer, ako se ponovo razmatra kompanija koja se bavi e-trgovinom, u slučaju da dođe do otkaza glavnog sajta, zbog relativno dugog vremenskog perioda za osposobljavanje hladnog sajta, kompanija može da izgubi veliki potencijalni profit. U ovom konkretnom slučaju, bolje je da kompanija investira više novca u implementiranje mehanizama za oporavak od katastrofe kao što je ulaganje u tehnologije vrućeg ili mlakog sajta.

S druge strane, ako se ne radi o ključnim računarskim resursima, model hladnog sajta predstavlja troškovno efektivno rešenje, za čije se korišćenje plaća samo onda kada se koristi, što nije slučaj u korišćenju modela mlakog i vrućeg sajta.

Slika 12-4: *Model hladnog sajta*



12.2.2. Replikacija podataka

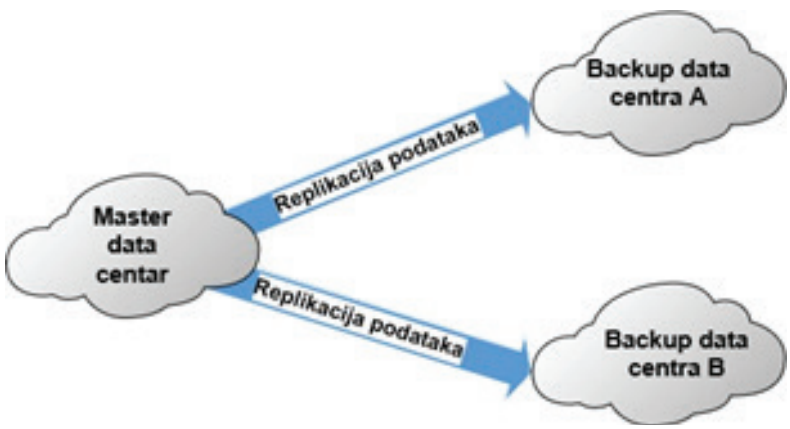
Replikacija podataka (eng. *data replication*) je transfer i sinhronizacija podataka između više računskih centara ili sajtova na klaudu.

Proces replikacije podataka prikazan je na Slici 12-5.

Za potrebe oporavka od katastrofe i sigurnosti podataka, postoji realna potreba da se podaci s vremena na vreme premeštaju, ili replikuju s jednog na drugi računski centar, ili s jednog na drugi sajt organizacije. Kopiranje podataka izvršavaju specijalizovane aplikacije za izradu rezervnih kopija, kako operativnih sistema, aplikacija i podataka, tako i čitavih sajtova.

Sa razvojem tehnologije virtuelizacije, proces premeštanja i replikovanja podataka je znatno olakšan, zato što mogu da se kopiraju i kompletne slike instanci virtuelnih mašina (eng. *virtual machine image*). Na ovaj način, u samo jednoj iteraciji, mogu da se kopiraju čitavi serveri sa podignutim i konfigurisanim operativnim sistemom, instaliranim zakrpama, svim aplikacijama i podacima.

Slika 12-5: *Proces replikacije podataka na klaudu*



S druge strane, mnoge klaud aplikacije, kao što su aplikacije baza podataka, već u svom kodu imaju ugrađene mehanizme za replikaciju. Neki klaud provajderi u svojim osnovnim ponudama uključuju replikaciju podataka, dok drugi replikaciju nude kao opciju koja se dodatno naplaćuje.

U svakom klaud okruženju dostupne su dve vrste replikacije: sinhrona (eng.*synchronous replication*) i asinhrona (eng. *asynchronous replication*).

U slučaju sinhrone replikacije, kada se upisuju novi podaci, upisivanje se vrši simultano na dve lokacije, kako bi se obezbedilo da se podaci na udaljenom sajtu sinhronizuju u realnom vremenu sa podacima na glavnom sajtu. Zbog prirode izvršavanja, sinhrona replikacija se u literaturi još naziva i replikacija u realnom vremenu (eng. *real-time replication*).

Dakle, sinhrona replikacija je proces kopiranja (replikovanja) podataka u realnom vremenu sa skladišta na glavnom sajtu na udaljeno skladište, kao što je prikazano na Slici 12-6. Na ovaj način, ako se dogodi katastrofa, vreme koje je potrebno da skladište podataka na sekundarnom sajtu postane operativno i funkcionalno svedeno je na minimum, uz minimizaciju verovatnoće gubitka podataka.

Slika 12-6: *Proces sinhrone replikacije na klaudu*



Sinhrona replikacija omogućava efikasnu implementaciju modela toplog i mlakog sajta.

S druge strane, u slučaju asinhrone replikacije, novi podaci se prvo upisuju na primarni sajt, tj. lokaciju, da bi se zatim replikovali na sekundarni sajt, tj. lokaciju, ili po unapred definisanom rasporedu, ili u vremenskim periodima kada je opterećenje klaud sistema manje. Zbog prirode izvršavanja, sinhrona replikacija se u literaturi još naziva i odložena replikacija (eng. *delayed replication*).

Asinhrona replikacija za transfer podataka koristi mehanizam skladištenja i prosleđivanja (eng. *store-and-forward*) i predstavlja troškovno efikasnije rešenje za kreiranje rezervnih kopija.

U ovom slučaju, podaci se prvo upisuju u glavno skladište, gde se čuvaju neko vreme, da bi se nakon određenog vremenskog perioda kopirali na backup skladište podataka. Asinhrona replikacija funkcioniše dobro i u slučaju nepouzdanih mreža, gde često dolazi do zastoja u mrežnom saobraćaju (eng. *network jitter*).

Mehanizmi asinhrone replikacije se implementiraju relativno lako, bilo pomoću specijalizovanih aplikacija za kreiranje rezervnih kopija, bilo na nivou hipervizora koji ima ugrađen mehanizam za kreiranje kopija instanci celih virtuelnih mašina na udaljenu lokaciju po unapred definisanom rasporedu. Potrebno je da se napomene da svi savremeni operativni sistemi, kao i veliki broj aplikativnog softvera imaju ugrađene mehanizme za izvršavanja asinhorne replikacije.

Mehanizam asinhrone replikacije na klaudu prikazan je na Slici 12-7.

Slika 12-7: *Proces asinhrone replikacije na klaudu*



12.2.3. Transfer fajlova i arhiviranje podataka

Kada se dogodi katastrofa, potrebno je da se izvrši transfer i sinhronizacija korisničkih i sistemskih fajlova između različitih lokacija na kojima se nalazi korporativni računski centar. Transfer fajlova se uglavnom vrši automatizovano i izvršava se u pozadini (eng. *background execution*) i transparentan je za korisnike klaud sistema.

Glavni cilj transfera fajlova (eng. *file transfer*) za vreme katastrofe i tokom oporavka od katastrofe je premeštanje tekućih korisničkih i sistemskih fajlova u računski centar rezervne lokacije.

Zaštita podataka je jedan od glavnih prioriteta svake operacije oporavka. Sistemi za skladištenje podataka imaju ugrađene sofistircirane tehnike koje sprečavaju gubitak kritičnih podataka. Većina klaud provajdera uz svoje osnovne ponude skladištenja i kreiranja rezervnih kopija podataka najčešće nude i usluge arhiviranja podataka.

Većina velikih klaud provajdera poseduje sisteme za skladištenje i čuvanje rezervnih kopija podataka velikog kapaciteta koji čuvaju i arhiviraju podatke u više nivoa. Ovakvi sistemi se nalaze ili na glavnom sajtu klaud provajdera, ili na udaljenoj, pomoćnoj lokaciji.

Proces arhiviranja podataka (eng. *data archiving*) odnosi se na premeštanje neaktivnih podataka, ili podataka koji više nisu potrebni u izdvojenu lokaciju za skladištenje, kako bi se podaci dugoročno čuvali.

Bez obzira da li klaud provajderi koriste skupe i sofisticirane, ili jeftinije sisteme za skladištenje podataka, klaud klijenti će moći, po veoma pristupačnoj ceni, da čuvaju velike količine svojih podataka i da im pristupaju po potrebi. Ovo je naročito značajno, zato što mnoge zakonodavne i korporativne regulative zahtevaju dugoročno čuvanje podataka i informacija i korišćenjem jeftinih usluga sistema za skladištenje i arhiviranje podataka klaud provajdera kompanije izvršavaju svoje obaveze.

Osim navedenog, proces skladištenja i arhiviranja podataka je automatizovan. Specijalizovane softverske aplikacije, koje imaju ugrađene algoritme za sprovođenje politika skladištenja i arhiviranja podataka, automatski identifikuju podatke koji mogu da se arhiviraju, koji se zatim čuvaju u specijalizovanim računskim centrima na klaudu, koji su namenjeni za potrebe dugoročnog čuvanja podataka.

Podaci koji su arhivirani i kojima se ređe pristupa čuvaju se uglavnom izvan glavne lokacije za skladištenje podataka klaud provajdera i za njihovo čuvanje se koriste jeftiniji medijumi. Medijumi za skladištenje ovakvih podataka mogu da budu na primer magnetni, ili optički diskovi, što zavisi od upotrebnih zahteva i cene. Ovim podacima se najčešće pristupa ili putem Interneta, ili direktnim povezivanjem pomoću *VPN* servisa.

Transfer podataka na rezervnu lokaciju ima višestruke koristi. Prvo, u slučaju katastrofe, ovim podacima može naknadno da se pristupi. Drugo, na ovaj način se obezbeđuje da se kopije podataka iz glavnog računskog centra čuvaju na udaljenoj lokaciji u formi rezervnih kopija.

Na tržištu postoje brojne klaud kompanije koje su specijalizovane za skladištenje, upravljanje, izradu i zaštitu rezervnih kopija podataka. Korišćenjem usluga ovih specijalizovanih kompanija, organizacija klijent se oslobađa tereta i odgovornosti koje bi imala ako bi podatke čuvala interno i samostalno. Čuvanje podataka izvan lokacije na kojoj se nalazi korporativni računski centar smanjuje rizik od gubitka podataka usled prirodnih nepogoda, grešaka izazvanih ljudskim faktorom, itd.

S druge strane, skladištenje podataka izvan glavne lokacije korporacije ima i određene nedostatke, koji se pre svega odnose na sigurnost i poverljivost podataka, zato što se kontrola nad podacima prepušta klaud provajderu. Zbog toga se osetljivi i poverljivi podaci najčešće čuvaju na skladišnim uređajima koji su locirani u korporativnom računskom centru.

12.2.4. Ponude treće strane

U poslednje vreme se na tržištu klaud računarstva pojavio veliki broj kompanija koje nude usluge konsaltinga i upravljanja računskim resursima sa ciljem da pomognu kompanijama u procesu oporavka od katastrofe.

Zbog marketinških razloga, ove kompanije sebe promovišu kao kompanije koje su omogućene za klaud (eng. *cloud enabled*) i nude novi model usluge klaud računarstva pod nazivom oporavak od katastrofe kao usluga (eng. *Disaster Recovery as a Service – DRaaS*).

S obzirom na to da su ovakve kompanije specijalizovane za pružanje usluga oporavka od katastrofe, one imaju razvijene modele za skoru svaku implementaciju mehanizama oporavka od katastrofe i poseduju dovoljno tehničkog znanja i *know-how* da dizajniraju personalizovane modele za oporavak.

Usluge ovakvih kompanija mogu da budu od neprocenljive koristi za mnoge korporativne klijente. Korišćenjem modela *DRaaS* ažurirani podaci se u realnom vremenu čuvaju na specijalnoj lokaciji koja je namenjena za ove potrebe, uz niske vrednosti *RPO* i *RTO* indikatora.

12.3. KONTINUITET POSLOVANJA

Pojam kontinuitet poslovanja (eng. *business continuity*) odnosi se na proces kojim se kompanija priprema za otkaz, ili gubitak računarskih resursa i sastoji se od definisanja niza koraka koje je potrebno preduzeti, kako bi se kompanija vratila u operativno stanje u slučaju katastrofe.

Najbolje preventivno rešenje u slučaju otkaza, ili gubitka računarskih resursa je temeljan i sveobuhvatan plan oporavka (eng. *recovery plan*).

Kontinuitet poslovanja je sposobnost organizacije da nastavi sa svojim poslovnim operacijama i da isporučuje proizvode i usluge svojim klijentima nakon događaja koji remeti normalno izvršavanje poslovnih operacija.

Mere za obezbeđenje kontinuiteta poslovanja postaju neophodnost za poslovanje bilo koje vrste, zato što je tolerantnost na zastoje u radu niska, zbog činjenice da svaki zastoj sa sobom vuče velike troškove, od kojih se poslovni subjekat teško oporavlja. Mnogi učesnici na tržištu, kako klaud računarstva, tako i ostalih domena, su zakonom obavezani da implementiraju plan kontinuiteta poslovanja i da obezbede tehničke preduslove za njegovo sprovođenje.

Plan kontinuiteta poslovanja (eng. *business continuity plan*) se definiše kao kreiranje plana koji prepoznaje postojanje inherentnih rizika i pretnji koje mogu da imaju loš uticaj na poslovanje kompanije.

Ovaj plan definiše načine na koje kompanija može da se zaštiti i da „preživi" katastrofu.

Kontinuitet poslovanja u značajnoj meri pojačava otpornost organizacije na prirodne i nepogode koje je izazvao ljudski faktor. Plan kontinuiteta poslovanja naglašava prioritete poslovanja i definiše detaljan plan reagovanja u slučaju nepogode.

Za razvoj sveobuhvatnog plana kontinuiteta poslovanja potrebne su detaljne informacije i znanja koja se prikupljaju iz različitih izvora.

12.3.1. Uspostavljanje plana kontinuiteta poslovanja

Jedan od glavnih mehanizama za obezbeđivanje kontinuiteta poslovanja kompanija koje svoje poslovanje baziraju na klaudu jeste implementiranje sistema za oporavak od katastrofe. Zbog toga je sistem za oporavak od katastrofe „srce" svakog plana kontinuiteta poslovanja.

Prvi korak u implementiranju sistema za oporavak od katastrofe je evaluacija potreba organizacije. Na osnovu identifikovanih potreba bira se klaud provajder koji te potrebe može da zadovolji. Zatim se pristupa dizajniranju i implementiranju sistema, koji zadovoljava sve organizacione zahteve u okviru predefinisanog budžeta.

Nakon uspostavljanja sistema za oporavak od katastrofe, potrebno je testirati dizajn i efektivnost implementiranog rešenja. Na ovaj način mogu da se uoče stvari koje su ranije bile previdete. Takođe, s vremena na vreme treba izvršiti reviziju zahteva za oporavak od katastrofe, zato što se potrebe i zahtevi vremenom menjaju.

12.3.2. Određivanje lokacija za alternativne sajtove

Sajt za potrebe oporavka od katastrofe treba da bude geografski dislociran od lokacije glavnog računskog centra organizacije. Na ovaj način, ako se na primer desi prirodna nepogoda, kao što je poplava, ili požar, na lokaciji glavnog sajta organizacije, backup sajt će da bude izvan uticaja nepogode.

Zbog navedenog, kao najčešća alternativa za *backup* sajt koristi se sajt provajdera usluga javnog klauda. Sajtovi provajdera usluga javnog klauda se uglavnom nalaze u više različitih geografskih oblasti, i korporativni klijent može da izabere oblast i računski centar klaud provajdera u toj oblasti, gde želi da locira svoje servise.

12.3.3. Određivanje kontinuiteta poslovnih operacija

Svaka kompanija treba da izvrši prioritizaciju izvršavanja svojih operacija i u slučaju nepogode, operacije sa većim prioritetom treba prvo osposobiti. Međutim, ova pitanja su izvan opsega tehnologije. Uloga tehnologije se ogleda samo u tome što će tehnologija da omogući da nakon nepogode sve operacije mogu nesmetano nastave sa izvršavanjem.

12.3.4. Računarska mreža i kontinuitet poslovanja

U planu kontinuiteta poslovanja eksplicitno mora da se naglase zahtevi koji se stavljaju pred klaud provajdera po pitanjima kapaciteta i propusnog opsega računarske mreže. Računarska mreža klaud provajdera mora da ima dovoljno kapaciteta koji će da garantuje nesmetan rad u slučaju nepogode. Ovo se pre svega odnosi na mrežni opseg koji će moći da podrži nadprosečni intenzitet mrežnog saobraćaja, zato što se tokom katastrofe vrši transfer velike količine podataka na rezervnu lokaciju.

12.3.5. Implementacija rubnih sajtova

Veliki javni klaud provajderi u poslednje vreme svojim klijentima nude usluge tzv. rubnih sajtova (eng. *edge sites*) na lokacijama širom sveta, koji omogućavaju prisutnost klaud provajdera na različitim i udaljenim geografskim područjima, bez postojanja potpuno operativnog računskog centra. Ovakve lokacije se u literaturi najčešće nazivaju rubna postrojenja i locirana su na važnijim lokacijama širom sveta, ili u geografskim područjima koje pokriva određeni klaud provajder.

Rubni sajtovi omogućavaju mnogo veći izbor korporativnom klijentu prilikom određivanja lokacije rezervnog sajta. Implementacija rubnih sajtova je najčešće jednostavna i za njihovo konfigurisanje se koriste interfejsi klaud provajdera u formi kontrolnih tabli, uz potpunu automatizaciju konfigurisanja okruženja pomoću pozadinskih servisa i aplikacija.



PITANJA ZA RAZMATRANJE

1. Šta je cilj tačke oporavka sistema?
2. Šta je cilj vremena oporavka sistema?
3. Objasniti postupak sagledavanja indikatora *RPO* i *RTO* za mehanizam oporavka od katastrofe.
4. Navesti ograničenja na strani klaud provajdera za implementiranje plana za oporavak od katastrofe.
5. Objasniti postupak kreiranja ogledala sajta.
6. Šta je vruć sajt? Objasniti.
7. Šta je mlak sajt? Objasniti.
8. Šta je hladan sajt? Objasniti.
9. Šta predstavlja proces replikacije podataka? Objasniti kroz primer.
10. Objasniti proces sihrone replikacije na klaudu.
11. Objasniti proces asihrone replikacije na klaudu.
12. Koji je glavni cilj transfera podataka za vreme katastrofe?
13. Objasniti ukratko proces arhiviranja podataka.



SIGURNOST U KLAUD OKRUŽENJU

DEO IV

SIGURNOST U KLAUD OKRUŽENJU



Sa razvojem računarskih tehnologija raste i broj i intenzitet *cyber* napada, zbog čega klaud provajderi i klijenti sve više novca ulažu u tehnologije koje unapređuju sigurnost klaud okruženja. Računarska sigurnost je domen koji u velikoj meri prevazilazi opseg klaud računarstva i predstavlja jedan od najvažnijih aspekata svake oblasti računarstva, bilo da su u pitanju baze podataka, računarske mreže, veštačka inteligencija i slično.

Ranije su troškovi i kvalitet usluga bili glavni parametri prilikom izbora klaud provajdera. Međutim, u poslednje vreme, kada biraju klaud provajdera, korporativni klijenti sve veću prednost daju provajderu koji ima konkurentnu prednost u odnosu na druge provajdere iz aspekta sigurnosti. Prednost ima onaj provajder koji obezbeđuje veći broj sigurnosnih sistema i servisa, koji podržava sofisticirane tehnike i mehanizme zaštite podataka, čije su sigurnosne politike usaglašene sa najboljim praksama računarske sigurnosti, koji obezbeđuje siguran i zaštićeni pristup klaud okruženju sa udaljenih lokacija, itd.

Slično kao i u slučaju trećeg dela udžbenika, ovaj deo takođe prikazuje i tehničke i netehničke aspekte računarske bezbednosti, gde su netehnički aspekti podjednako važni kao i tehnički i zato ih ne treba potcenjivati. Tako na primer, u ovom delu definisan je pojam rizika, zato što je za najbolje razumevanje koncepta klaud sigurnosti potrebno razumeti ovaj pojam. Osim toga, predstavljeni su i načini za klasifikaciju i prioritizaciju rizika na klaudu. Klaud organizacija najbolje može da se zaštiti ako zna od čega treba da se zaštiti i ako razume, i može da kvantifikuje potencijalnu štetu koju napadač može da nanese.

S obzirom da računarska sigurnost, pa tako i sigurnost na klaudu predstavljaju prioritet poslednjih godina, globalne organizacije koje se bave bezbednošću računarskih resursa, donele su veliki broj zakonskih regulativa, standarda i politika koje se odnose na klaud sigurnost. Budući da i klaud provajderi i korporativni klijenti treba da se usaglase sa donešenim regulativama i standardima, u ovom delu udžbenika prikazane su sigurnosne politike i regulatorni zahtevi.

Za potpunije razumevanje tehnologija koje unapređuju sigurnost klaud okruženja potrebno je razumevanje glavnih ciljeva računarske sigurnosti i sigurnosnih kontrola, koje omogućavaju ostvarivanje ovih ciljeva, među kojima su najvažniji ciljevi poverljivosti (tajnosti), integriteta i raspoloživosti (dostupnosti). Zbog toga je cela jedna glava posvećena ovoj tematici.

Zbog sve većeg značaja tehnologija klaud sigurnosti, u ovom delu udžbenika je osim svega pomenutog, poseban naglasak i na osnovnim tehnologijama koje unapređuju sigurnost svakog klaud okruženja. O nekim od prikazanih tehnologija bilo je reči ranije, i u tom kontekstu ova glava predstavlja rezime tehnologija, tehnika i principa sa kojima su čitaoci već upoznati, uz određene nove elemente.

13. POJAM, KLASIFIKACIJA I PRIORITIZACIJA RIZIKA NA KLAUDU

Kako bi se na najbolji način razumeli koncepti klaud sigurnosti, potrebno je razumevanje pojma rizika. Zbog toga je cela ova glava posvećena klasifikaciji i prioritizaciji rizika na klaudu.

Sa nastankom Interneta, broj i ozbiljnost rizika i pretnji koji se odnose na računarske resurse, sa kojima se suočavaju kompanije koje svoje operacije izvršavaju na klaudu, su se eksponencijalno povećali.

S obzirom da se klaud sistemu u velikoj većini slučajeva pristupa putem nesigurne računarske mreže kao što je Internet, razumevanje rizika i pretnji Internet okruženja je od suštinske važnosti za implementiranje bilo kog modela klaud računarstva.

Klaud provajderi i korporativni klijenti moraju da razumeju ozbiljnost rizika i da shvate da rizici i pretnje nikada neće da nestanu. Jedina mogućnost je da se egzistencija rizika i pretnji prihvati i da se preduzmu preventivne i proaktivne mere kako bi se sprečili, ili minimizovali potencijalni gubici.

Sa razvojem klaud i računarskih tehnologija razvijaju se i alati i softveri koji narušavaju bezbednost i sigurnost klaud računarskih resursa. Čim se jedna pretnja otkloni razvojem i implementiranjem odgovarajućeg zaštitnog mehanizma, nastaje neka druga pretnja, i tako u krug. U literaturi se ovakvo stanje, koje važi i za druge oblasti, ne samo za klaud računarstvo, naziva trka naoružanjem (eng. *arms race*).

Ako cena preuzimanja akcije koja sprečava da potencijalni rizik postane realnost nadmašuje materijalnu vrednost potencijalnog gubitka, koji bi nastao ako bi se rizik materijalizovao, onda se na osnovu analize troškova i koristi (eng. *cost benefit analysis*) zaključuje da je razumno da rizik postoji.

Metodom izračunavanja rizika izračunava se težina potencijalne pretnje u odnosu na verovatnoću njene realizacije. Kompanije moraju da prihvate činjenicu da neki rizici ne mogu da se izbegnu i takvi rizici se u literaturi nazivaju rezidualni rizici (eng. *residual risks*).

13.1. PROCENA RIZIKA

U literaturi koja proučava sigurnost i bezbednost računarskih resursa može da se nađe više definicija pojma rizik, sa kojim se u neposrednu vezu dovode i pojmovi pretnja i ranjivost.

Prema jednoj definiciji, rizik (eng. *risk*) je potencijalni gubitak, ili šteta prouzrokovana na klaud računarskom resursu, do koje dolazi onda kada pretnja uspe da eksploatiše ranjivost klaud računarskog resursa.

Pretnja (eng. *threat*) se odnosi na alate, softvere, sisteme i ljude, koji mogu, bilo slučajno, bilo namerno da eksploatišu ranjivost klaud računarskog resursa.

Ranjivost (eng. *vulnerability*) je slabost klaud računarskog resursa koju pretnja može da eksploatiše.

Slično kao i za pojam rizik, tako i za procenu rizika postoje brojne definicije u literaturi iz oblasti računarske sigurnosti.

Prema jednoj definiciji, procena rizika (eng. *risk assessment*) je metodologija koja se bavi pretnjama i ranjivostima sistema i potencijalnim uticajem na gubitak informacija, pa i čitavih sistema na klaudu.

Procena rizika se u literaturi još naziva i analiza, ili kalkulacija rizika (eng. *risk analysis*). Radi uspostavljanja konzistentnosti korišćenih termina, u nastavku će se koristiti pojam procena rizika.

Svaki identifikovani rizik treba da se naglasi, opiše i da se evaluira u odnosu na verovatnoću da će biti realizovan. U proceni rizika treba razmišljati široko i slobodoumno (eng. *open-minded*), izvan stereotipa, zato što stalno nastaju novi rizici koji su često potpuno suprotni postojećim rizicima.

Ključne komponente, koje treba uzeti u obzir prilikom primene metodologije procene rizika su:

- postojeći rizici kojima je klaud organizacija izložena. Ova komponenta omogućava razvoj scenarija koji pomažu u donošenju odluka o tome koje akcije treba preduzeti, kao odgovor na rizike, ako dođe do njihove realizacije. U klaud okruženju javljaju se rizici sa operativnim sistemima, serverima, aplikacijama i podacima. Svaka organizacija treba da razvije najbolji mogući plan za reagovanje u slučaju materijalizacije navedenih rizika,
- rizici koje treba uzeti u obzir. Ova komponenta procesa procene rizika omogućava klaud organizaciji da izvrši analizu i donese zaključak o tome koji rizici su realni, i koji će da se realizuju sa manjom verovatnoćom. Na ovaj način kompanija fokusira svoje resurse na one rizike koji imaju najveću verovatnoću realizacije. Tako na primer, veća verovatnoća je da će da se desi pokušaj upada na klaud skladište podataka, dok je verovatnoća da će da se dogode poplave manja. U ovom slučaju klaud organizacija treba da alocira više resursa na preuzimanje proaktivnih mera za sprečavanje upada u sistem i
- koordinacija sa pokazateljem uticaja na poslovanje (eng. *business impact analysis – BIA*). Ova komponenta omogućava klaud kompaniji da kreira širu sliku rizika sa kojima se suočava i da donese optimalne odluke o reagovanju na rizike u različitim scenarijima.

Kada se izvršavaju aktivnosti procene rizika, često se koriste usluge eksperta iz oblasti računarske sigurnosti (eng. *computer security expert*). Kako bi razvio opštu sliku rizika i potencijalnih problema sa kojima se klaud korporacija suočava, ekspert izvršava sledeće aktivnosti:

- intervjuisanje administratora mreža, skladišta, baza podataka, vlasnika podataka, itd., kako bi se utvrdilo koje informacije zahtevaju dodatnu sigurnost i kako bi se identifikovale postojeće ranjivosti sistema,
- evaluiranje mrežne infrastrukture kako bi se identifikovale poznate ranjivosti i
- ispitivanje fizičkog okruženja lokacije na kojoj se nalazi klaud oprema da bi se utvrdili fizički rizici.

Na osnovu prikupljenih informacija dobija se dobra polazna osnovna za razvoj kontramera koje je potrebno preduzeti sa ciljem izbegavanja, ili smanjivanja rizika.

13.1.1. Prioritizacija rizika

Kada se vrši procena rizika jedna od najvažnijih zadataka je prioritizacija rizika.

Ne treba svi rizici podjednako da se uzimaju u obzir, zato što za neke postoji veća, dok za druge postoji manja verovatnoća da će biti realizovani. Osim toga, neke rizike treba prihvatiti, dok za druge treba implementirati efektivne kontramere, zato što mogu da imaju fatalne posledice po klaud kompaniju.

Kako bi se utvrdilo kojim rizicima treba da se posveti veća pažnja, potrebno je izvršiti prioritizaciju rizika.

Da bi se rizici prioritizovali, potrebno je da se kvantifikuje negativan uticaj na klaud operacije koji rizik može da ima. Za potrebe kvantifikacije negativnog uticaja rizika koriste se sledeći termini:

- očekivana vrednost godišnjeg gubitka (eng. *annual loss expectancy value – ALE*) je novčani pokazatelj koji pokazuje koliki gubitak se po osnovu rizika očekuje u periodu od godinu dana,
- očekivani jednokratni gubitak (eng. *single loss expectancy – SLE*) je novčani pokazatelj koji pokazuje koliki se gubitak očekuje od jednokratne realizacije rizika. *SLE* se dalje deli na dve komponente:
  - vrednost imovine (eng. *asset value – AV*), kao što su oprema, podaci, itd. i
  - faktor izloženosti (eng. *exposure factor – EF*).

- godišnja stopa dešavanja rizika (eng. *annualized rate of occurrence – ARO*) je kvantitativni parametar koji pokazuje koliko puta se u toku poslovne godine očekuje materijalizacija određenog rizika. *ARO* se najčešće definiše na osnovu analize istorijskih podataka.

Dalje se za kvantifikovanje (izračunavanje) rizika koristi sledeća formula:

$$ALE = SLE \times ARO$$

Tako na primer, ako očekivana vrednost jednokratkog gubitka po osnovu određenog rizika iznosi 2.000 dolara (*SLE*=2.000) i ako se očekuje da će rizik da se događa 8 puta godišnje (*ARO*=8), onda je očekivana vrednost godišnjeg gubitka, *ALE*, 16.000 dolara (*ALE*=2.000x8).

S druge strane, ako je verovatnoća da će da dođe do događaja koji prouzrokuje gubitak samo 10% na godišnjem nivou, onda bi vrednost očekivane stope gubitka (*ALE*) iznosila samo 200 dolara (*ALE*=2.000x0.1).

Na osnovu ovako izračunatog i kvantifikovanog negativnog uticaja rizika na klaud okruženje vrši se prioritizacija rizika, gde veći prioritet dobijaju rizici koji imaju veću vrednost *ALE* indikatora.

Ključna stvar u proceni rizika je da se identifikuje i imovina (eng. *asset*), koja obuhvata klaud opremu, podatke, aplikacije i ostale resurse, koju je potrebno zaštititi. Tek nakon toga vrši se identifikacija rizika od kojih se imovina štiti.

13.1.2. Kvalitativna i kvantitativna procena rizika

Prema nekim stanovištima koja su navedena u literaturi, pravi se razlika između dva tipa procene rizika: kvalitativna (na osnovu mišljenja i procene), ili subjektivna (eng. *subjective risk assessment*) i kvantitativna (na osnovu troškova), ili objektivna (eng. *objective risk assessment*).

Pokazatelji i formule koje su prikazani za izračunavanje vrednosti rizika u Sekciji 13.1.1 baziraju se na vrednosti izraženim u američkim dolarima, ili nekim drugim monetarnim jedinicama i zato pripadaju kvantitativnim metodama procene rizika.

U cilju boljeg razlikovanja kvalitativne i kvantitativne procene rizike mogu da se razmotre sledeća dva primera.

Mala klaud kompanija je izgubila svoj jedini server na kojem se nalaze arhivirane rezervne kopije svih dokumenata od nastanka kompanije pa do danas, koji uključuju dokumente o misiji kompanije, poslovnim ciljevima, zaposlenim, finansijskim prihodima, itd. Ovi dokumenti nisu neophodni za normalno funkcionisanje i obavljanje poslovnih operacija, ali su zato važni za razvoj korporativnog identiteta i korporativne kulture. Da bi se izračunala vrednost negativnog uticaja ovakvog rizika potrebno je da se koriste kvalitativne metode za procenu rizika.

Drugi primer bi bio scenario kada bi klaud kompanija izgubila bazu podataka o klijentima sa istorijskom listom svih transakcija. Kompanija ne može da posluje bez ove baze, jer se sve radi elektronski. Baza može da se ponovo kreira, ali je potrebno sve podatke o klijentima i istorijskim transakcijama ručno uneti u bazu podataka na osnovu hard kopija faktura i ostalih dokumenata. Za izračunavanje negativnog uticaja ovakvog rizika koristile bi se metode kvantitativne procene rizika. Negativni uticaj može da se izračuna ako se uzme u obzir oportunitetni trošak koji nastaje tokom vremenskog perioda u kojem kompanija ne izvršava poslovne aktivnosti zbog gubitka baze podataka.

Termin kvantitativni se u kontekstu procene rizika odnosi na monetarnu vrednost, dok se termin kvalitativni odnosi na subjektivne vrednosti, kao što su reputacija, slike i dokumenta iz prošlosti, ili bilo koji drugi podaci koji imaju subjektivnu vrednost za klaud kompaniju.

13.2. TERMINOLOGIJA RIZIKA

U nastavku su navedeni važniji termini koje je potrebno razumeti kako bi se na najbolji način razumeli rizici sa kojima se suočava klaud kompanija.

Verovatnoća dešavanja rizika je termin koji je intuitivan i koji ne zahteva dodatna objašnjenja. Ovaj termin se jednostavno odnosi na stepen verovatnoće da će rizik biti realizovan.

U cilju boljeg upravljanja rizicima (eng. *risk management*), verovatnoći dešavanja rizika treba da se dodele kvantitativne i kvalitativne vrednosti, koje predstavljaju realnu mogućnost da će rizik biti materijalizovan.

Lista kvalitativnih i kvantitativnih vrednosti verovatnoće dešavanja rizika, koju je preporučila organizacija *NIST* u dodatku *G* publikacije 800-30 navedena je u Tabeli 13-1 .

Tabela 13-1: *Preporučena lista vrednosti verovatnoće dešavanja rizika*

Kvalitativna vrednost	Kvantitativna vrednost	Opis
Veoma visoka	10	Protivnik će skoro sigurno da inicira događaj sa pretnjom.
Visoka	8	Protivnik će najverovanije da inicira preteći događaj.
Umerena	5	Protivnik će možda da inicira preteći događaj.
Niska	2	Protivnik najverovatnije neće da inicira preteći događaj.
Veoma niska	0	Protivnik sigurno neće da inicira preteći događaj.

Vektor pretnje (eng. *threat vector*) je način na koji napadač predstavlja pretnju po klaud kompaniju.

Ovaj način gotovo može da bude bilo šta, od određenog alata koji napadač koristi protiv kompanije, kao što je skener ranjivosti (eng. *vulnerability scanner*), do fizičkog upada na lokaciju gde se nalazi klaud oprema. U opštem smislu, vektor pretnje može da bude i lažna email poruka koja je poslata zaposlenima u klaud kompaniji, ili lažna (zlonamerna) pristupna tačka (eng. *rogue access point*).

Kako bi se bolje razumeli rizici, potrebno je razumevanje i sledećih pokazatelja: prosečno vreme između otkaza, prosečno vreme do otkaza, prosečno vreme za oporavak, cilj vremena oporavka i cilj stanja oporavka.

Prosečno vreme između otkaza (eng. *Mean Time Between Failures – MTBF*) je mera anticipirane učestalosti otkaza komponenti sistema.

MTBF indikator služi kao pomoć za određivanje životnog veka komponente. Ako je na primer *MTBF* napajanja servera 2 godine, treba očekivati da će ova komponenta moći da se koristi maksimalno 2 godine i treba unapred da se pripremi komponenta koja će je zameniti. Ako komponenta traje duže nego što je predviđeno *MTBF* indikatorom, klaud kompanija će imati nepredviđeno smanjivanje troškova, što ima pozitivan uticaj na poslovanje.

Prosečno vreme do otkaza (eng. *Mean Time To Failure – MTTF*) je prosečno vreme do otkaza sistema ili komponente koja ne može da se popravi.

Ako sistem ili komponenta mogu da se poprave, onda se gleda *MTBF* indikator, a ako ne mogu, onda se gleda *MTTF* pokazatelj.

Prosečno vreme za oporavak (eng. *Mean Time To Restore – MTTR*) je indikator koji pokazuje koliko je vremena potrebno da se sistem ili komponenta poprave kada se otkaz dogodi.

Tako na primer, ako *MTTR* indikator za klaud ruter ima vrednost 24 sata, to znači da je u slučaju otkaza potrebno 24 sata da se ruter popravi.

Cilje vremena oporavka (eng. *Recovery Time Objective – RTO*) je maksimalno prihvatljivo vreme tokom kojeg sistem ili komponenta mogu da budu *offline*.

Ako su sistem ili komponenta nefunkcionalni duže od intervala koji je definisan *RTO* indikatorom, dolazi do narušavanja kontinuiteta poslovanja.

Cilj stanja oporavka (eng. *Recovery Point Objective – RPO*) je indikator koji je sličan *RTO* indikatoru, s tim što se definiše stanje u koje sistem treba da se vrati nakon otkaza.

Tako na primer, sistem treba da se vrati u ono stanje u kojem je bio sat vremena pre pada, ili dan pre pada, itd. Kada se kaže stanje, najčešće se misli da stanje instaliranih aplikacija, upravljačkih programa i servisa.

Mnogi od navedenih termina se koriste u Sporazumima o nivou korišćenja klaud usluga. Svaki klaud klijent treba da bude upoznat sa ovim terminima, kako bi najbolje znao šta da očekuje od provajdera klaud usluga, i kako bi u kranjoj liniji odabrao klaud provajdera koji najbolje može da ispuni njegova očekivanja.

13.3. STRATEGIJE KOJE SE PREDUZIMAJU NA OSNOVU IDENTIFIKOVANIH I PROCENJENIH RIZIKA

Na osnovu identifikovanih i procenjenih rizika, klaud kompanija ima na raspolaganju nekoliko strategija koje može da primenu sa ciljem izbegavanja, ili smanjivanja potencijalnog negativnog uticaja identifikovanih rizika.

U literaturi se najčešće navode sledeće strategije za izbegavanje i smanjivanje uticaja rizika: izbegavanje rizika, transfer rizika, ublažavanje rizika, odvraćanje od rizika i prihvatanje rizika.

Sve navedene strategije su u nastavku ukratko opisane.

13.3.1. Izbegavanje rizika

Strategija izbegavanja rizika (eng. *risk avoidance strategy*) odnosi se na donošenje odluke da se ne preduzimaju nikakve akcije koje su vezane za rizik.

Tako na primer, ako za kompaniju rizik predstavljaju prilozi (eng. *attachments*) koji se šalju uz email poruke, klaud kompanija može da donese odluku kojom se zabranjuje da zaposleni šalju email poruke sa prilozima i da postavi pravila na svojim firewall uređajima za blokiranje dolazećih email poruka sa prilozima.

Naravno, opisani scenario je radikalan i retko se primenjuje u realnim klaud sistemima.

13.3.2. Transfer rizika

Strategija izbegavanja rizika (eng. *risk avoidance strategy*) odnosi se na donošenje odluke da se ne preduzimaju nikakve akcije koje su vezane za rizik.

Klaud kompanije na ovaj način smanjuju troškove koji bi nastali ukoliko bi se rizik materijalizovao.

13.3.3. Ublažavanje rizika

Strategija ublažavanja rizika (eng. *risk mitigation strategy*) sprovodi se svaki put kada klaud kompanija preuzima korake koji dovode do smanjivanja rizika.

U domen ove strategije spadaju koraci, kao što su instaliranje antivirusnog softvera, edukovanje korisnika o potencijalnim pretnjama, praćenje mrežnog saobraćaja, itd.

Kompanija *Microsoft* navodi sledeće aktivnosti koje bi trebalo da se sprovode, kako bi se rizik ublažio, boljom edukacijom i treningom zaposlenih u klaud kompaniji:

- kontinuirano ažuriranje poruka koje se odnose na sigurnost i bezbednost računarskih resursa,
- upoznavanje kako novih, tako i postojećih zaposlenih sa pretnjama u klaud okruženju,
- postavljanje visokih ciljeva u kontekstu znanja zaposlenih o sigurnosti računarskih resursa i rizika i
- kontinuirano ponavljanje navedenih aktivnosti, kako bi se razvila „organizaciona svest" o rizicima i pretnjama u klaud okruženju.

Prema organizaciji *CompTIA*, strategija ublažavanja rizika treba da se sprovodi rutinskim kontrolama i praćenjima koja se odnose na korisnička prava (eng. *user rights*) i reviziju dozvola za pristup resursima (eng. *permission review*),

upravljanje promenama (eng. *change management*), i upravljanje incidentima (eng. *incident management*). Osim ovoga, treba implementirati politike koje se odnose na sprečavanje gubitaka podataka i zaštitu od krađe korišćenjem tehnoloških kontrola.

Sistem za sprečavanje gubitka podataka (eng. *data loss prevention – DLP*) je sistem koji se koristi za praćenje sadržaja sistema (radnih stanica, servera i mreža), kako bi se osiguralo da sadržaj neće biti izbrisan ili premešten. Ovakvi sistemi takođe redovno prate korisnike podataka, kako bi se osiguralo da neautorizovana osoba nema pristup sistemu.

Jedan od najboljih sistema za sprečavanje gubitka podataka (eng. *Data Loss Prevention - DLP*) je *MyDLP*, koji spada u grupu open source rešenja koja se izvršavaju na *Windows* platformi. Softver *MyDLP* može besplatno da se preuzme sa adrese: *www.mydpl.org*. Kao jedan od najpoznatijih komercijalnih implementacija *DLP* sistema navodi se *Microsoft Forefront (URL: www.microsoft.com/forefront)*.

13.3.4. Odvraćanje od rizika

Strategija odvraćanja od rizika (eng. *risk deterrence strategy*) podrazumeva poznavanje neprijatelja i posedovanje informacija o neprijatelju, gde se neprijatelj informiše o kontramerama koje kompanija može da preduzme u slučaju da neprijatelj preduzme akciju.

Ova strategija se ponekada u literaturi naziva i strategija zastrašivanja neprijatelja (eng. *intimidation strategy*).

13.3.5. Prihvatanje rizika

Strategija prihvatanja rizika (eng. *risk acceptance strategy*) primenjuje se u situacijama kada su troškovi implementiranja bilo koje od predhodno navedenih strategija veći od troškova koje rizik može da prouzrokuje i kompaniji se u takvim situacijama najviše isplati da prihvati rizik.

Zaključci o tome koje je rizike najbolje prihvatiti donose se na osnovu aktivnosti procene rizika.



PITANJA ZA RAZMATRANJE

1. Koje ključne komponente treba uzeti u obzir prilikom procenjivanja rizika?
2. Objasniti načine za prioritizaciju rizika.
3. Koja je razlika između kvalitativne i kvantitativne procene rizika?
4. Šta je prosečno vreme između otkaza? Objasniti.
5. Šta je prosečno vreme do otkaza? Objasniti.
6. Šta je prosečno vreme za oporavak? Objasniti.
7. Šta je cilj vremena oporavka? Objasniti.
8. Šta je cilj stanja oporavka? Objasniti.
9. Objasniti strategiju izbegavanja rizika.
10. Objasniti strategiju transfera rizika.
11. Objasniti strategiju ublažavanja rizika.
12. Objasniti strategiju odvraćanja od rizika.
13. Objasniti strategiju prihvatanja rizika.

14. GLAVNI CILJEVI RAČUNARSKE SIGURNOSTI I SIGURNOSNE KONTROLE U KLAUD OKRUŽENJU

U ovoj glavi je dat prikaz glavnih ciljeva računarske sigurnosti i opisane su sigurnosne kontrole, koje omogućavaju materijalizaciju ciljeva računarske sigurnosti.

S obzirom da računarska sigurnost nije glavna tema ovog udžbenika i pod predpostavkom da čitaoci poseduju osnovno znanje iz ovog domena, neki detalji su izostavljeni.

Računarska sigurnost (eng. *computer security*) je oblast računarstva koja se bavi praćenjem, nadzorom, identifikacijom i sprečavanjem opasnosti koje mogu da dovedu do nestabilnog rada, prekida, ili bilo kakve vrste štete na softveru, hardveru, ili bilo kom drugom računarskom resursu.

Računarska sigurnost se bazira na nekoliko principa koji se u literaturi najčešće nazivaju glavni ciljevi računarske sigurnosti (eng. *core computer security goals*). Ovi principi služe kao glavni orjentir za donošenje odluka koje se odnose na sigurnost i bezbednost računarskih resursa u svakoj organizaciji, bilo da je to organizacija koja koristi tradicionalnu organizaciju računarskih resursa, ili da je to organizacija koja svoje poslovne aktivnosti obavlja korišćenjem klaud infrastrukture.

Tri glavna cilja računarske sigurnosti koja se najčešće navode u literaturi su poverljivost (eng. *confidentialy*), integritet (eng. *integrity*) i dostupnost (eng. *availability*).

Svaki od navedenih ciljeva obavezno treba da se uzme u obzir u definisanju svakog programa sigurnosti na nivou klaud organizacije.

Osim navedena tri, u literaturi se često navode još dva dodatna cilja računarske sigurnosti, a to su bezbednost (eng. *safety*) i slojevita sigurnost (eng. *layered security*). Ponekada se umesto termina slojevita sigurnost upotrebljava i termin široka odbrana (eng. *defense in depth*).

Za realizaciju ciljeva računarske sigurnosti koriste se sigurnosne kontrole. Sigurnosne kontrole pokušavaju da spreče, ili da ograničene negativno dejstvo sigurnosnog incidenta, tj. događaja koji ima potencijal da ugrozi i/ili naruši sigurnost računarskih resursa i sistema.

14.1. CILJEVI RAČUNARSKE SIGURNOSTI

U ovom poglavlju date su osnovne informacije o svakom, gore navedenom, cilju računarske sigurnosti.

14.1.1. Cilj poverljivosti

Cilj poverljivosti odnosi se na sprečavanje neautorizovanog otkrivanja podataka.

Drugim rečima, podacima mogu da pristupe samo autorizovani korisnici, osobe i sistemi, dok neautorizovani ne mogu. Da bi se cilj poverljivosti realizovao, dostupno je više metoda, među kojima su najznačajnije šifrovanje, kontrola pristupa i steganografija.

Jedan od najboljih načina za ostvrivanje poverljivosti podataka je šifrovanje, tj. enkripcija podataka. Šifrovanje može da se primeni na bilo koji tip podataka, kao što su lične informacije, podaci u bazama podataka, podaci na mobilnim uređajima, itd. Kontrola pristupa pomaže u ostvarivanju cilja poverljivosti ograničavanjem pristupa, dok steganografija realizuje poverljivost sakrivanjem podataka, kao što je na primer sakrivanje tekstualnih fajlova unutar digitalne slike.

U nastavku su navedene osnovne informacije o svakoj od navedenih metoda.

Šifrovanje

Šifrovanje (eng. *encryption*) je proces koji se primenjuje u kriptografiji i kojim se vrši izmena podataka sa ciljem da se podaci učine nečitljivim za osobe, korisnike i sisteme koji ne poseduju određeno znanje, tj. ključ.

Drugim rečima, šifrovanjem se podaci šifruju, kako bi se onemogućilo čitanje podataka od strane neautorizovanih osoba, korisnika i sistema.

Za proces šifrovanja je u srpskom jeziku usvojen i engleski termin, pa se često u domaćoj literaturi umesto pojma šifrovanje koristi pojam enkripcija.

Da bi šifrovanim podacima pristupili autorizovani korisnici i sistemi, podaci prvo moraju da se dešifruju (eng. *decryption*). U modernoj kriptografiji dostupni su sofisticirani algoritmi i tehnike za šifrovanje podataka, koji dešifrovanje podataka od strane neautorizovanih učesnika čine praktično nemogućom.

Slično kao i u slučaju termina šifrovanja, za pojam dešifrovanje je u srpskom jeziku usvojen i engleski termin dekripcija.

Radi realizacije cilja poverljivosti, u klaud okruženju dostupni su mnogi algoritmi za šifrovanje, kao što su *DES* (*Data Encryption Standard*), *3DES* (*Triple DES*), *AES* (*Advanced Encryption Standard), RSA (Rivest–Shamir–Adleman*), kao i mnogi drugi.

Ostvarivanje cilja poverljivosti postaje prioritet kada na primer korisnik klaud sistema preko nesigurne Internet mreže treba da pošalje osetljive podatke i lične informacije (eng. *Personally Identifiable Information – PII*), kao što su medicinski kartoni pacijenata, ili podaci o kreditnoj kartici. Za siguran transfer osetljivih podataka, podatke pre slanja treba šifrovati.

Kontrola pristupa

Kontrola pristupa (eng. *access control*) se ostvaruje primenom mehanizama identifikacije (eng. *identification*), autentifikacije (eng. authentication) i autorizacije (eng. *authorization*), čime se obezbeđuje da samo autorizovani akteri mogu da pristupe sistemima i podacima.

Primer identifikacije je kada korisnik tvrdi svoj identitet pomoću jedinstvenog korisničkog imena. Autentifikacijom korisnik dokazuje svoj identitet pomoću šifre. Autorizacijom se dozvoljava, odnosno zabranjuje pristup, određenim računarskim resursima na osnovu korisničkih privilegija.

Steganografija

Steganografija (eng. *steganography*) je treći dobro poznati metod za ostvarivanje cilja poverljivosti računarske sigurnosti. U literaturi postoje brojne definicije pojma steganografija.

> Prema jednoj definiciji steganografija je sakrivanje podataka na očiglednim mestima. Steganografija je sakrivanje poruke i najčešće se bazira na prerušavanju poruke unutar audio, video, ili tekstualnog fajla.

Kao jednostavan primer upotrebe tehnike steganografije navodi se inkorporiranje (umetanje) tekstualnog fajla u fajl sa slikom korišćenjem klasičnih alata, kao što su *WinRAR* i *Windows* interpretatora komandi (eng. *command prompt - cmd*).

14.1.2. Cilj integriteta

> Ostvarivanjem cilja integriteta obezbeđuje se da podaci nisu modifikovani (promenjeni), oštećeni ili korumpirani bilo slučajno, bilo namerno.

Cilj integriteta podataka je podjednako važan kao i druga dva glavna cilja računarske sigurnosti u klaud okruženju.

U idealnim scenarijima samo autorizovani korisnici imaju mogućnost da modifikuju podatke. Međutim, u svakodnevnoj klaud računarskoj praksi, često se dešava da dođe do izmena podataka koji nisu trebali da budu modifikovani. Ovakve izmene impliciraju slučajne ili namerne aktivnosti izvršene od strane neautorizovanih korisnika, malicioznog softvera (eng. *malware*), ljudskim ili sistemskim greškama, itd. Kada se ovakve promene dogode, kaže se da su podaci izgubili integritet.

Neki od načina za ostvarivanje cilja integriteta su korišćenje tehnike heširanja, implementiranje digitalnih potpisa i digitalnih sertifikata.

Heširanje

Tehnike heširanja (eng. *hashing techniques*) se široko koriste za ostvarivanje cilja integriteta podataka. Među brojnim raspoloživim algoritmima heširanja izdvajaju se *MD5* (*Message Digest 5*), *SHA-1* (*Secure Hash Algorithm 1*) i *HMAC* (*Hash-based Message Authentication Code*).

Heš vrednost, ili samo heš (eng. hash) je broj koji se dobija kada se algoritam za heširanje primeni na određene podatke, kao što su fajl ili poruka.

Za isti podatak, uz primenu istog algoritma za heširanje, rezultujući heš će uvek biti isti. Upoređivanjem heš vrednosti za isti skup ulaznih podataka u različitim vremenskim trenucima može da se utvrdi da li su originalni podaci izmenjeni. Ako su heš vrednosti iste, podaci nisu promenjeni, i obrnuto.

Važno je da se napomene i da je heširanje ireverzibilan proces. Na osnovu heš vrednosti nemoguće je generisati originalan ulazni podatak.

Za ilustraciju procesa heširanja može da posluži sledeći jednostavan primer. Neka korisnik nebojša hoće korisniku ivana da pošalje fajl sa osetljivim informacijama preko klaud *DropBox* servisa. Kako bi ostvario cilj integriteta podataka, korisnik nebojša na klaud dostavlja fajl zajedno sa hešom tog fajla. Radi simplifikacije, neka je heš vrednost fajla broj 123. Korisnik ivana sa *DropBox* servisa preuzima fajl zajedno sa hešom. Nakon preuzimanja, ivana izračunava heš vrednost preuzetog fajla. Ako je heš vrednost primljenog fajla ista kao i heš vrednost poslatog fajla, u ovom slučaju 123, korisnik ivana zna da fajl nije izmenjen. Međutim, ako je heš primljenog fajla na primer 789, ivana zna da je cilj integriteta podataka narušen i da nije primila originalan fajl.

Osim gore navedenog scenarija, heširanje se koristi i u brojnim drugim situacijama, kao što je na primer provera integriteta podataka kada se fajlovi preuzimaju sa *Web*-a. Neki programi pri preuzimanju fajlova sa *Web*-a automatski proveravaju njihove heš vrednosti kako se utvrdilo da fajlovi nisu oštećeni prilikom preuzimanja. Programi vrše ovu proveru tako što upoređuju heš vrednost fajla kada je na *Web*-u sa heš vrednošću istog fajla kada je preuzet, tj. kada je sačuvan na lokalnom računaru. Ako se ove heš vrednosti razlikuju, program izveštava korisnika da je došlo do greške prilikom preuzimanja. Na mnogim sajtovima uz fajl koji se preuzima administratori postuju i heš tog fajla. Nakon preuzimanja fajla, korisnik računa heš vrednost preuzetog fajla i upoređuje je sa heš vrednošću koja je postovana na *Web*-u. Za ove potrebe mogu da se koriste besplatni alati (eng. *freeware*), kao što je *md5sum.exe*.

Napominje se da se heširanjem ne dobija informacija o tome ko je modifikovao poruku (podatak). Heširanje pruža samo informacije o tome da je poruka modifikovana i da se takvoj poruci ne sme verovati. Neke aplikacije i *email* programi umesto heširanja koriste *MAC* (*Message Authentication Code*) za proveru integriteta podataka. U svakom slučaju, primenjuju se isti koncepti.

Podaci mogu da izgube integritet i ljudskom greškom. Tako na primer, administratori baza podataka često pišu i koriste skripte za izvršavanje grupnog ažuriranja podataka. Ako skripta na nekom mestu u kodu ima grešku, onda može da dođe do narušavanja integriteta podataka prilikom izvršavanja skripte.

Kao dva ključna koncepta integriteta podataka navode se:

- integritet obezbeđuje garancije da podaci nisu modifikovani, pokvareni, ili korumpirani, bilo namerno, bilo slučajno. Do narušavanja cilja integriteta najčešće dolazi u situacijama kada neautorizovani korisnici menjaju podatke, ili ljudskim i sistemskim greškama i
- heširanje je jedan od načina provere integriteta podataka. Heš je numerička vrednost koja se dobija kada se algoritam za heširanje izvrši nad datim fajlom ili porukom. Heš vrednosti se najčešće računaju za isti skup podataka u različitim vremenskim trenucima. Ako su izračunate heš vrednosti iste kaže se da je integritet podataka održan, a u suprotnom kaže se da je integritet narušen.

Digitalni potpisi, digitalni sertifikati i neporicanje

Osim heširanja, za ostvarivanje cilja integriteta podataka u klaud okruženju mogu da se koriste i digitalni potpisi (eng. *digital signatures*).

Digitalni potpisi se često koriste za proveru integriteta poslate *email* poruke. Tako na primer, korisnik nebojša hoće korisniku ivana da pošalje *email* poruku. Nebojša uz poruku pridružuje i svoj digitalni potpis. Kada korisnik ivana primi poruku, nebojšin digitalni potpis koji je pridružen poruci potvrđuje da *email* poruka nije modifikovana u transportu.

Osim integriteta, digitalni potpis obezbeđuje i autentifikaciju. Kada korisnik ivana primi poruku, nebojšin digitalni potpis ivanu uverava da je zaista on taj koji je poruku i poslao. Obezbeđivanjem autentifikacije na osnovu digitalnih potpisa sprečava napadače da se lažno predstavljaju i da na taj način šalju maliciozne *email* poruke.

Tako na primer, bez digitalnog potpisa napadač može lažno da se predstavi kao Nebojša i da pošalje poruku Ivani. Kada ivana primi poruku, ona vidi da je poruka od nebojše i s obzirom da veruje korisniku Nebojša, ona preuzima prilog iz poruke, koji može da bude maliciozni program koji će da ošteti njen računar, ili će da ukrade poverljive podatke.

Digitalni potpisi takođe obezbeđuju i nemogućnost poricanja (eng. *non-repudiation*). Drugim rečima, nakon slanja *email* poruke, korisnik Nebojša ne može kasnije da negira da je poslao poruku, jer njegov digitalni potpis tvrdi drugačije.

Digitalni potpisi obezbeđuju integritet podataka, autentifikaciju i neporicanje.

Sigurnosni sistemi implementiraju metode neporicanja i na druge načine. Jedan od načina su logovi za praćenje (eng. *audit logs*), koji beleže informacije o tome ko je pristupao i kom računarskom resursu je pristupao, šta je radio sa tim računarskim resursom, gde i kada je to radio.

Tako na primer, ako korisnik nebojša nakon logovanja na sistem sa svojim korisničkim imenom i lozinkom izbriše nekoliko važnih fajlova, logovi za praćenje će snimiti akcije koje je Nebojša izvršio. Nebojša kasnije ne može da opovrgne da je fajlove izbrisao, jer o tim aktivnostima postoje dokazi u logovima za praćenje.

Za korišćenje digitalnih potpisa potrebni su digitalni sertifikati (eng. *digital certificates*) i infrastruktura javnog ključa (eng. *Public Key Infrastructure – PKI*). Ključevi koji se koriste za šifrovanje i dešifrovanje podataka implementiraju se na nivou digitalnih sertifikata, dok infrastruktura javnog ključa obezbeđuje okruženje za kreiranje, upravljanje i distribuciju sertifikata.

14.1.3. Cilj dostupnosti

Cilj dostupnosti odnosi se na zahteve da su podaci i servisi u klaud okruženju dostupni i raspoloživi onda kada su potrebni.

Za neke od tradicionalnih organizacija, dovoljno je da podaci i servisi budu dostupni samo u toku radnog vremena (od 8:00 časova do 17:00 časova).

S druge strane, zahtevi klaud organizacija su takvi da je potrebno da se obezbedi raspoloživost podataka i servisa po principu 365/7/24 (365 dana u godini, 7 dana u nedelji, 24 časova dnevno).

Svaki provajder klaud usluga ima zahtev da su njegovi servisi i podaci dostupni 365/7/24.

Glavni načini za ostvarivanje cilja dostupnosti su redudantnost (eng. *redundancy*) i otpornost na greške (eng. *fault tolerance - FT*). Uz ovo je poželjno i da se redovno instaliraju zakrpe sistema (eng. *system patches*), kako bi se predupredilo da greške u softveru imaju uticaj na raspoloživost sistema.

U nastavku su redudantnost i otpornost (tolerancija) na greške opisani iz aspekta računarskog sistema (ne podataka).

Redudantnost i otpornost na greške

Princip redudantnosti se odnosi na kreiranje duplikata kritičnih sistema čime se obezbeđuje otpornost na greške.

Ako određena kritična komponenta, ili sistem otkažu, primenom principa redudantnosti, sistem će nastaviti da radi bez prekida. Drugim rečima, sistem sa mehanizmom otpornosti na greške može da ima grešku koja će se tolerisati i sistem će da nastavi da radi.

Mehanizam za otpornost na greške se najčešće koristi kako bi se eliminisala tzv. jedna tačka pada sistema (eng. *single point of failure – SPOF*). U slučaju sistema koji ne koriste tehnike otpornosti na greške i redudantnosti, u slučaju otkaza jedne komponente, ceo sistem prestaje da radi. Tako na primer, u slučaju servera koji ima samo jedan hard disk, disk predstavlja jednu tačka otkaza sistema. Ako disk otkaže, ceo server prestaje da radi i servisi koje server pruža u momentu postaju nedostupni.

Kao neki od primera upotrebe redudantnosti i otpornosti na greške u klaud okruženju navode se sledeći:

- redudantnost diskova. Sistemi diskova sa otpornošću na greške, kao što su *RAID* 1, *RAID* 5 i *RAID* 6 omogućavaju da sistem nastavi da funkcioniše čak i ako neki disk (diskovi) otkažu. Za više informacija o *RAID* sistemima pogledati Poglavlje 9.9,
- redudantnost servera. *Failover* klasteri servera implementiraju redudantnost na nivou servera i ovim mehanizmom se osigurava da će sistem nastaviti da radi ako jedan server otkaže. U slučaju otkaza, sve operacije se u realnom vremenu prebacuju sa pokvarenog servera na *backup* server i sistem nastavlja sa radom. Tehnika virtuelizacije se često koristi za implementiranje failover klastera servera,
- balansiranje opterećenja. U primeni mehanizma za balansiranje opterećenja koristi se više servera koji zajedno rade na isporučivanju jedne usluge, kao što je na primer *Web* sajt koji ima visok intenzitet saobraćaja. Ovom tehnikom se povećava raspoloživot *Web* sajtova i *Web* baziranih aplikacija,

- redudantnost sajtova. Ako jedan sajt prestane sa radom zbog prirodnih nepogoda, kao što su vatra, poplava, zemljotres, itd., organizacija može da prebaci svoje kritične sisteme na alternativni sajt, koji može da se implementira u formi vrućeg sajta koji je spreman i raspoloživ po principu 365/24/7, hladnog sajta, lokacija na koju se oprema i zaposleni premeštaju u slučaju potrebe, ili mlakog sajta, koji predstavlja kompormis između vrućeg i hladnog sajta. Za više informacija o redudantnosti sajtova pogledati Sekciju 12.2.1,
- kreiranje rezervnih kopija. Ako se redovno kreiraju rezervne kopije, podaci mogu da se rekonstruišu (oporave) u slučaju gubitka. Kreiranjem rezervnih kopija takođe se povećava raspoloživost servisa i aplikacija,
- alternativni izvori električne energije. Implementacija *UPS* (*Uniterruptible Power Supply*) uređaja i generatora električne energije omogućavaju nastavak rada sistema u slučaju otkaza glavne električne mreže i
- sistemi za hlađenje. Implementiranjem sofisticiranih *HVAC* (*Heat, Ventilation and Air Conditioning*) sistema poboljšava se dostupnost tako što se minimizuje mogućnost otkaza u radu sistema zbog pregrejavanja.

Zakrpe sistema i aplikacija

Sistemske i aplikativne zakrpe predstavljaju još jedan način za ostvarivanje cilja raspoloživosti sistema i podataka. Greške u kodu softvera (eng. software bugs, glitches) su česte i retko se dešava da u momentu objavljivanja softver nema ni jednu grešku. Tokom eksploatacije softverskih proizvoda greške se otkrivaju i vendori periodično izbacuju zakrpe koje ispravljaju identifikovane greške.

> Greške u softveru stvaraju čitav niz problema, od sigurnosnih problema, neočekivanog rada sistema i aplikacija, pa sve do nasumičnih i nepredvidivih padova sistema.

Klaud organizacije koriste softverske pakete za upravljanje zakrpama i konfigurišu programe za upravljanje zakrpama (eng. *patch management programs*) na nivou *IT* službe, čime obezbeđuju ažuriranost njihovih sistema u svakom vremenskom trenutku,, što dalje poboljšava raspoloživost sistema i podataka.

14.1.4. Bezbednost

Osim navedena tri glavna cilja računarske sigurnosti, u literaturi se često navodi i četvrti cilj – cilj bezbednosti.

> Cilj bezbednosti se odnosi na bezbednost ljudi (osoblja) i na bezbednost opreme i imovine organizacije.
>
> Oprema uvek može da se zameni, ali ljudi nikada. Zbog toga je bezbednost ljudi na prvom mestu.

Neki od najvećih rizika za ljude obuhvataju prirodne nepogode, kao što su požari, zemljotresi, jako vetrovi, požari i slično. Organizacije razvijaju planove kontinuiteta poslovanja kojima se pripremaju za ovakve vrste nepogoda. U planovima se jasno navode strategije za napuštanje prostorija organizacije u slučaju da se desi neka od navedenih nepogoda.

Bezbednost opreme i imovine organizacije ostvaruje se primenom kontrola za ostvarivanje fizičke sigurnosti, kao što je na primer izgradnja zaštitne ograde oko lokacije na kojoj se organizacija nalazi, sigurnosna vrata, *CCTV* (*Closed Circuit Television*) sistemi koji se koriste za video nadzor, itd.

Većina planova za kontinuitet poslovanja prioritizuje bezbednost ljudi. Tako na primer, mnoge organizacije implementiraju elektronska sigurnosna vrata za ulazak u prostorije u kojima se nalazi sofisticirana oprema. Zbog sigurnosti ljudi, u slučaju prekida električne energije ova vrata se automatski otvaraju, kako bi se predupredila situacija da neko od osoblja ostane zaključan u prostoriji.

14.1.5. Slojevita sigurnost

Slojevita sigurnost, koja se često u literaturi naziva i široka odbrana, odnosi se na sigurnosne prakse za implementiranje nekoliko nivoa (slojeva) zaštite računarskih resursa.

Organizacija ne može samo da implementira *firewall* uređaje, ili antivirusni softver i da se smatra da je zaštićena. Sigurnosni mehanizmi moraju da se implementiraju na različitim nivoima. Na ovaj način, ako sigurnost na jednom nivou otkaže, organizacija je i dalje zaštićena.

Kao jedna analogija između slojevite zaštite i situacija iz realnog života može da se navede sledeći primer. Primer nezaštićenog sistema bila bi situacija kada bi kupac otišao u lokalni šoping centar kolima, a pri tom je ostavio u kolima na vidljivom mestu veliku svotu novca, ostavio ključeve u kolima i ostavio auto upaljen i izašao iz auta. U ovakvom opisanom scenariju postoji velika verovatnoća da će auto biti ukraden dok dok kupac obavlja kupovinu. S druge strane, ako kupac parkira auto, obezbedi da ništa vredno nije ostalo u kolima što može da se vidi sa prozora automobila, ako je isključio motor, izvukao ključ, stavio sigurnosne zaštite (lavlju kandžu, blokator motora, itd.), onda je verovatnoća da će auto biti ukraden minimalna. Ovo bi bio primer sistema sa slojevitom sigurnošću.

Ne postoji univerzalni recept za sigurnost računarskih resursa. Sistem nikada neće biti potpuno siguran, ali se implementiranjem slojevite sigurnosti verovatnoća da će sistem biti ugrožen i kompromitovan smanjuje.

Mnoge klaud organizacije slojevitu sigurnost implementiraju na sledećim nivoima: nivou korporativnih politika, procedura i svesti zaposlenih, nivou fizičke sigurnosti lokacije na kojoj se nalazi klaud računarska oprema, nivoima spoljne (*DMZ*) i unutrašnje (intranet) računarske mreže i na nivoima host računara, aplikacija i podataka. Ovako definisani nivoi slojevite sigurnosti prikazani su na piramidi slojevite sigurnosti (Slika 14-1).

Slika 14-1: *Piramida slojevite sigurnosti*



14.2. SIGURNOSNE KONTROLE

Sigurnosni incident je štetan događaj, ili serija događaja koji mogu da imaju negativan uticaj na ostvarivanje ciljeva poverljivosti, integriteta i raspoloživosti podataka, sistema i ostalih računarskih resursa. Sigurnosni incident je širok pojam i odnosi se na široku lepezu događaja, od nameravanih napada na računarske sisteme, do malicioznog softvera, infekcija, gubitka podataka, itd.

Sigurnosne kontrole (eng. *security controls*) pokušavaju da spreče, ili da ograničene negativno dejstvo sigurnosnog incidenta. Sigurnosne kontrole se koriste za realizaciju ciljeva računarske sigurnosti.

U literaturi iz oblasti računarske sigurnosti može da se pronađe nekoliko taksonomija sigurnosnih kontrola na osnovu različitih kriterijuma, ali se najčešće navodi klasifikacija na osnovu načina implementacije i na osnovu ciljeva koje sigurnosne kontrole teže da ostvare.

U nastavku su ukratko prikazane obe taksonomije sigurnosnih kontrola.

14.2.1. Klasifikacija sigurnosnih kontrola na osnovu načina implementacije

Jedan od načina za klasifikaciju sigurnosnih kontrola, koji je uobičajen u literaturi, jeste podela na osnovu načina implementacije.

Prema podeli sigurnosnih kontrola na osnovu načina implementacije, sigurnosne kontrole se dele u tri grupe: tehničke, upravljačke i operativne kontrole.

Tehničke kontrole (eng. technical controls) koriste tehnologiju, upravljačke kontrole (eng. *management controls*) koriste administrativne i upravljačke metode, dok operativne kontrole (eng. *operational controls*) implementiraju ljudi tokom izvršavanja svakodnevnih poslovnih operacija.

Tehničke kontrole

Tehničke kontrole se izvršavaju automatski. Administrator sistema samo treba da instalira i da konfiguriše tehničku kontrolu.

Neki od primera tehničkih kontrola obuhvataju sledeće:

- šifrovanje. Šifrovanje se smatra jakom tehničkom kontrolom koja se koristi za zaštitu poverljivosti podataka. Šifrovanje se koristi i kada se podaci šalju preko mreže i kada se podaci nalaze u stanju mirovanja na serveru, *desktop* računaru, mobilnom uređaju, ili bilo kom tipu lokalnog ili mrežnog skladišta,
- antivirusni softver. Kada se jednom instalira i konfiguriše, antivirusni softver automatizuje proces zaštite,
- *IDS* i *IPS* sistemi. *IDS* i *IPS* sistemi automatski i neprekidno prate saobraćaj na mreži ili host računaru i obezbeđuju zaštitu u realnom vremenu od mnogobrojnih pretnji.
- *firewall* uređaji su jedan od najšire korišćenih načina za implementiranje mehanizama tehničke kontrole i
- princip najmanje privilegije (eng. *least privilege principle*) obezbeđuje da individue ili procesi imaju taman onoliko privilegija koliko im je neophodno za obavljanje svakodnevnih poslovnih zadataka i funkcija. Privilegije su kombinacije prava i dozvola.

Upravljačke kontrole

Upravljačke sigurnosne kontrole koriste metode planiranja i evaluacije kako bi smanjile rizik i efikasno upravljale rizikom.

U literaturi se umesto naziva upravljačke sigurnosne kontrole često sreće i naziv administrativne kontrole (eng. *administrative controls*).

Mnoge organizacije neprekidno tokom poslovne godine osvežavaju svoja dokumenta i evaluiraju implementirane procese za upravljanje rizicima. Implementirani procesi za upravljanje rizicima klaud organizacija u većini slučajeva koriste jednu ili više upravljačkih kontrola.

Kao neki od primera upravljačkih kontrola mogu da se navedu sledeći:

- procena rizika koja pomaže u kvantifikovanju i kvalifikovanju rizika sa kojima se klaud organizacija suočava. Kvantitativne procene rizika koriste troškove i vrednosti opreme i imovine kako bi potencijalni rizik izrazile u monetarnim vrednostima. S druge strane, metode za kvalitativnu procenu rizika, rizike prioritizuju na osnovu verovatnoće dešavanja i potencijalnog negativnog uticaja na poslovanje klaud organizacije,
- procenjivanje ranjivosti se odnosi na pokušaje da se otkrije aktuelna slabost ili ranjivost sistema. U slučajevima kada je potrebno, klaud organizacija može da implementira dodatnu kontrolu koja smanjuje rizik na osnovu identifikovanih ranjivosti i
- penetraciono testiranje (eng. *penetration testing*) ide korak dalje od procene ranjivosti, zato što ovi testovi predstavljaju realne pokušaje eksploatacije ranjivosti. Tako na primer, kako bi se testirala ranjivost servera za koji se smatra da nema implementirane najsvežije zakrpe organizacija može da unajmi napadača koji simulira napad na server.

Operativne kontrole

Operativne kontrole osiguravaju da su svakodnevne poslovne operacije klaud kompanije kompatibilne sa opštim planom računarske sigurnosti. Operativne kontrole implementiraju ljudi (ne tehnologija).

Neki od primera operativnih kontrola su:

- izgrađivanje svesnosti osoblja klaud organizacije da su priroda Interneta i klauda takve da se organizacija svakodnevno suočava sa mnogobrojnim pretnjama. Kontinualnim treningom osoblja obezbeđuje se da zaposleni na optimalan i siguran način upravljaju svojim pristupnim šiframa, da sa svojih radnih stolova sklanjaju poverljiva dokumenta i da imaju razumevanje o osnovnim sigurnosnim pretnjama, kao što su maliciozni softveri i tehnike phishing-a,
- upravljanje konfiguracijama sistema postavlja se bazno (polazišno) stanje sistema koje je sigurno. Upravljanje promenama osigurava da se u slučaju promene sistema, ili komponente sistema klaud organizacija neće naći u ugroženom stanju zbog grešaka u konfiguraciji,
- kontigentnim planiranjem (eng. *contigency planing*) se razvija plan reagovanja organizacije u slučaju nepredviđenih prekida rada. Cilj kontigentnog planiranja je minimizovanje negativnog uticaja nepredviđenih prekida u radu sistema na poslovanje klaud organizacije,
- zaštita fizičkih medija kao što su USB diskovi, eksterni i interni hard diskovi i ostali mediji za kreiranje rezervnih kkopija i
- zaštita fizičkog okruženja, kada se implementiraju fizičke kontrole poput kamera, katanaca na vratima, sistema za ventilaciju i slično.

14.2.2. Klasifikacija sigurnosnih kontrola na osnovu ciljeva

Još jedan način za taksonimiju sigurnosnih kontrola koji se često sreće u literaturi jeste klasifikacija na osnovu ciljeva koje sigurnosne kontrole teže da ostvare i veze sa sigurnosnim incidentima.

Prema klasifikaciji sigurnosnih kontrola na osnovu ciljeva, sigurnosne kontrole mogu da budu preventivne (eng. *preventive*), detektirajuće (eng. *detective*), korektivne (eng. *corrective*), zastrašivajuće (eng. *deterrent*) i kompenzirajuće (eng. *compensating*).

Preventivne kontrole pokušavaju da spreče da se sigurnosni incident dogodi. Detektirajuće kontrole pokušavaju da detektuju sigurnosni incident nakon što se dogodio. Korektivne kontrole pokušavaju da isprave uticaj incidenta. Zastrašujuće kontrole pokušavaju da obeshrabre individuu ili entitet da izazove sigurnosni incident. Konačno, kompenzirajuće kontrole spadaju u grupu alternativnih kontrola, koje se primenjuju kada gore navedene glavne sigurnosne kontrole nije moguće implementirati.

Preventivne kontrole

U idealnoj situaciji organizacija neće da se suoči ni sa jednim sigurnosnim incidentom, što ujedno i predstavlja cilj preventivnih kontrola.

Dakle, cilj preventivnih kontrola je da preduprede dešavanje sigurnosnog incidenta. Neki od primera preventivnih kontrola koje se najčešće primenjuju na klaudu su:

- otežavanje (eng. *hardening*) je praksa unapređenja sigurnosti sistema i aplikacija onemogućavanjem nepotrebnih i nekorišćenih servisa i protokola, zaštitom interfejsa i aplikacija za upravljanje sistemom (npr. *Web* interfejs za pristup konfiguraciji rutera), zaštita šifara i onemogućavanje naloga koji nisu neophodni. Otežavanjem se podrazumevana sigurnost sistema i aplikacija unapređuje,

- kreiranje svesti o računarskoj sigurnosti i permanentnim obukama obezbeđuje se da su korisnici sistema i aplikacija svesni slabosti sistema i aplikacija, kao i pretnji koje „vrebaju" iz nesigurnog klaud okruženja. Ovakvi programi uključuju i upoznavanje sa tehnikama socijalnog inženjerstva (eng. *social engineering*),
- čuvari i radnici obezbeđenja (eng. *security guards*) u značajnoj meri unapređuju fizičku sigurnost organizacije. Radnici obebeđenja uglavnom vrše i proveru identiteta osoba koje imaju nameru da uđu na lokaciju organizacije,
- upravljanje promenama osigurava da će promena biti sprovedena kako treba i da neće da dođe do neplaniranih prekida u radu sistema. Umesto da administratori „u hodu" (eng. *on-the-fly*) izvršavaju promene sistema i aplikacija, procesom upravljanja promenama, svaka promena je unapred osmišljena. Ova sigurnosna kontrola pripada grupi i preventivnih i operativnih kontrola i
- primena politike za onemogućavanje korisničkih naloga osigurava da će korisnički nalozi svih zaposlenih koji napuste organizaciju biti onemogućeni. Na ovaj način se predupređuje situacija da zaposleni koji je napustio organizaciju iskoristi svoj bivši položaj, kako bi oštetio podatke, sisteme i/ili aplikacije.

Detektirajuće kontrole

Bez obzira što postoje preventivne sigurnosne kontrole koje nastoje da spreče događanje sigurnosnog incidenta, nažalost, u svakodnevnoj praksi, sigurnosni incidenti se često dešavaju. Zbog toga detektirajuće kontrole igraju značajnu ulogu na klaudu.

Detektirajuće kontrole pokušavaju da identifikuju sigurnosni incident, nakon što se incident dogodio.

Kao neki od primera detektirajućih kontrola navode se sledeći:

- kontinuirano praćenje sistemskih i drugih logova. U svakom klaud okruženju postoji veliki broj logova koji prate aktivnosti korisnika, sistema i mreža. Tako na primer logovi *firewall*-a snimaju detalje saobraćaja koji su blokirali. Praćenjem ovakvih logova moguće je detektovati sigurnosne incidente. Procese praćenja logova i slanja izveštaja automatizuje savremena računarska oprema,

- analiza trendova. Osim što se praćenje logova koristi u cilju detektovanja izolovanog sigurnosnog incidenta, ova mera se često koristi i sa svrhom identifikovanja trendova. Tako na primer *IDS* pokušava da detektuje napade na klaud računarske resurse i u slučaju napada *IDS* putem sistema alarmiranja o napadu obaveštava sve relevantne strane. Analizom istorijskih obaveštavanja ovog sistema mogu da se identifikuju trendovi, kao što je na primer trend povećanja učestalosti i intenziteta napada na određeni klaud sistem,
- revizija sigurnosti (eng. *security audit*) odnosi se na aktivnost evaluacije sigurnosti klaud organizacije. Tako na primer, revizijom korisničkih šifara utvrđuje se da li politika korišćenja šifara u organizaciji obezbeđuje da će korisnici koristiti jake šifre. Slično, periodičnom revizijom korisničkih prava može da se identifikuje da li korisnici imaju više prava i dozvola nego što im je potrebno za obavljanje svakodnevnih zadataka,
- video nadzor (eng. *video surveillance*) koji se najčešće implementira pomoću *CCTV* sistema može da detektuje svaku aktivnost koja narušava fizičku bezbednost organizacije. Sistemi video nadzora se često primenjuju i kao zastrašujuća kontrola i
- uređaji za detektovanje pokreta (eng. *motion detection devices*) koriste se za alarmiranje u situacijama kada se uljez nađe na lokaciji klaud organizacije.

Poređenje preventivnih i detektirajućih kontrola

Korisno je da se ukaže na razliku između preventivnih i detektirajućih sigurnosnih kontrola.

> Detektirajuća kontrola ne može da predvidi kada će sigurnosni incident da se dogodi i zato ne može da ga spreči. S druge strane, preventivna kontrola može da spreči da se incident dogodi.

Razlika između ova dva tipa kontrola najbolje može da se vidi ako se uporede sigurnosne kamere i čuvari koji unapređuju fizičku sigurnost klaud organizacije:

- video nadzor. Obična kamera koja ne podržava funkciju snimanja može da spreči događanje incidenta i ima ulogu preventivne kontrole. S druge strane *CCTV* sistemi funkcionišu i kao preventivne i kao detektirajuće kontrole zbog mogućnosti snimanja video materijala.

Ako se sigurnosni incident dogodi, analizom video snimka može da se detektuje sigurnosni incident i

- primarna uloga čuvara fizičke lokacije klaud organizacije je preventivna kontrola. S druge strane, ako se uljez ušunjao u prostorije organizacije, čuvari mogu da identifikuju štetu koju je uljez prouzrokovao, i u ovim slučajevima čuvari imaju ulogu detektirajuće kontrole.

Korektivne kontrole

Korektivne kontrole nastoje da obrnu uticaj sigurnosnog incidenta nakon što se incident dogodio.

Neke od primera korektivnih kontrola obuhvataju sledeće:

- aktivni *IDS* pokušavaju da detektuju napadača i u slučaju detekcije modifikuju okruženje kako bi blokirali napad i
- sistemi za izradu rezervnih kopija i oporavak podataka i sistema obezbeđuju mogućnost povraćaja podataka i sistema nakon što su sistemi korumpirani i podaci izgubljeni.

Zastrašivajuće kontrole

Primenom zastrašivajućih kontrola pokušava se da se pretnja obeshrabri.

Neke od zastrašujućih kontrola nastoje da obeshrabre napadača i da on odustane od napada, dok druge pokušavaju da obeshrabre zaposlene da prekrše pravila sigurnosnih politika klaud organizacije.

Mnoge zastrašivajuće kontrole imaju ulogu i preventivnih kontrola. Tako na primer, sama činjenica da postoje čuvari fizičke lokacije klaud organizacije često odvraća napadače da pokušaju sa napadom. Slično, napadač koji se služi tehnikama socijalnog inženjerstva pokušava da prevari portira koji vrši identifikaciju osoba i da uđe u zgradu pod tuđim identitetom, ali ako osim portira na ulazu stoji i čuvar, samo postojanje čuvara može napadača da odvrati od pokušaja da se lažno legitimiše.

Kao još dva primera zastrašivajućih kontrola koje unapređuju fizičku sigurnost klaud organizacije navode se katanci za kablove (eng. *cable locks*) i katanci za računarski hardver (eng. *hardware locks*). Tako na primer, ako je lap top računar privezan za komad nameštaja pomoću katanaca za kablove, potencijalni napadač će u velikom broju slučajeva da odustane od krađe.

Kompenzirajuće kontrole

Kompenzirajuće kontrole spadaju u grupu alternativnih kontrola koje se koriste kada primarne sigurnosne kontrole nisu dostupne.

Tako na primer, ako sigurnosna politika klaud organizacije nalaže da zaposleni za ulazak u prostorije organizacije moraju da koriste pametne kartice za potrebe autentifikacije uz pomoć više faktora sigurnosti, u slučaju novog zaposlenog, čija kartica još uvek nije izrađena, za ulazak u zgradu može privremeno da se koristi pristupni token sa vremenski baziranom jednokratnom šifrom (eng. *Time-based One- time Password – OTP*). U ovom slučaju vremenski bazirana jednokratna šifra predstavlja kompenzirajuću sigurnosnu kontrolu.



PITANJA ZA RAZMATRANJE

1. Objasniti značaj koncepta sigurnosti u računarstvu.
2. Navesti tri glavna cilja računarske sigurnosti. Objasniti.
3. Definisati pojam šifrovanja.
4. Navesti algoritme za šifrovanje. Objasniti.
5. Šta je kontrola pristupa? Objasniti.
6. Šta je steganografija?
7. Šta predstavlja cilj integriteta? Objasniti kroz primer.
8. Šta je heširanje?
9. Ilustrovati proces heširanja kroz primer.
10. Koja su dva ključna koncepta integriteta podataka?
11. Šta je digitalni potpis? Objasniti kroz primer.
12. Šta je digitalni sertifikat? Objasniti kroz primer.
13. Navesti primere upotrebe redudantnosti i otpornosti na greške u klaud okruženju.
14. Objasniti koncept slojevite sigurnosti. Dati primer.
15. Navesti primere tehničkih kontrola.
16. Navesti primere upravljačkih kontrola.
17. Navesti primere operativnih kontrola.
18. Klasifikovati sigurnosne kontrole na osnovu ciljeva i opisati svaku od njih.

15. USAGLAŠAVANJE SA ZAHTEVIMA KLAUD SIGURNOSTI

Iz domena klaud računarstva, na nivou globalnih i lokalnih organizacija, vlada i agencija, donešen je veliki broj industrijskih i vladinih politika, zakona, zahteva, regulativa, sertifikata i preporuka sa kojima klaud provajder i klijent moraju da se usaglase.

S obzirom da je u poslednje vreme sve veći akcenat na sigurnosti računarskih sistema zbog porasta broja, intenziteta i učestalosti napada na računarske resurse, veliki broj preporuka, zahteva i regulativa odnosi se na sigurnost klaud okruženja.

Ako klaud provajder ne usaglasi svoje poslovanje sa lokalnim i globalnim zahtevima klaud sigurnosti, provajder neće moći da egzistira na globalnom tržištu klaud usluga. S druge strane, bez obzira na garancije provajdera, kompanija klijent je sama odgovorna za svoje podatke i sisteme i zbog toga je od ključne važnosti uspostavljanje sigurnosnih politika u kompanijama koje koriste klaud usluge.

U ovoj glavi prikazano je usaglašavanje klaud provajdera sa zakonima, regulativama, politikama i preporukama vlade i industrije, koje se odnose na klaud sigurnost, kao i uspostavljanje sigurnosne politike u organizaciji klijenta klaud usluga.

15.1. USAGLAŠAVANJE NA STRANI KLAUD PROVAJDERA

Usaglašavanje klaud provajdera sa zahtevima i standardima iz domena klaud sigurnosti implementira se i sprovodi se kako bi se zadovoljile sigurnosne smernice, preporuke, zakoni i regulative vlade i industrije.

Usaglašavanje svojih sigurnosnih politika sa preporučenim standardima, klaud provajder može da ponudi svoje usluge i specifičnim industrijama, poput zdravstva i finansija, u zemljama širom sveta.

Da bi mogao da zaključuje poslovne ugovore sa regulisanim entitetima, klaud provajder mora da usaglasi svoje sigurnosne politike i da dobije akreditacije za usaglašavanje u polju sigurnosti. Takođe, klaud provajderu se pruža mogućnost i da izvrši dodatne sertifikacije, koje su specifične za pojedine industrije. Tako na primer, u SAD-u, da bi klaud provajder mogao da izvršava usluge *hostinga* podataka sektora zdravstva, klaud provajder mora da dokaže da je usaglašen sa *HIPPA* standardima.

Slično, da bi klaud provajder mogao da *hostuje* finansijske podatke za obradu kreditnih kartica, provajder svoje sigurnosne politike i standarde mora da usaglasi sa *PCI* politikom, i da na taj način dokaže da se je sposoban da obezbedi sigurnost i integritet osetljivih finansijskih podataka.

U mnogim uređenim zemljama širom sveta, provajder javnih klaud usluga mora da izvrši usaglašavanje svojih sigurnosnih operacija sa brojnim vladinim regulativama. Zahtev jedne takve regulative je čuvanje obimnih logova o svim aktivnostima klaud korisnika (kom klaud resursu je pristupano, ko mu je pristupao, u koje vreme, sa koje lokacije, itd.). Kao još jedan sličan zahtev navodi se čuvanje informacija prikupljenih analizom logova na visoko sigurnim i bezbednim lokacijama.

Na sreću, dostupni su brojni softverski alati koji klaud provajderu olakšavaju izvršavanje aktivnosti usaglašavanja sa sigurnosnim standardima. Tako na primer, dostupni su alati koji omogućavaju automatizovano prikupljanje i upravljanje logovima heterogenih sistema i servisa koji se nalaze na geografski udaljenim i distribuiranim lokacijama širom sveta. Takođe su raspoloživi i alati za automatizovano prikupljanje metrika o performansama heterogenih sistema (*Windows, GNU/Linux, Android*, itd.). Većina navedenih alata ima mogućnost vizuelne interpretacije prikupljenih podataka.

Klaud provajderi implementiraju sigurnosne servise koji se odnose na upravljanje promenama u konfiguracijama sistema i aplikacija. Svaka promena se čuva u posebno dizajniranoj bazi podataka, a sistem za upravljanje verzijama (eng. *Version Control System - VCS*) omogućava da korisnik klaud servisa u svakom vremenskom trenutku stanje aplikacija i sistema može da vrati na stanje u prethodnom vremenskom periodu. Uz svaku promenu čuvaju se metapodaci o tome ko je izvršio promenu, kada je promena izvršena, koliko je trajala, itd.

Napominje se da su na tržištu klaud računarstva raspoloživi i brojni alati trećih strana koji olakšavaju usaglašavanje sigurnosnih politika sa nametnutim regulativama.

15.2. USPOSTAVLJANJE SIGURNOSNE POLITIKE U ORGANIZACIJI KOJA KORISTI KLAUD USLUGE

Bez obzira na to što klaud provajder u većini slučajeva ima razvijene sigurnosne politike i procedure na više nivoa, sama organizacija klijent je odgovorna za svoje podatke i aktivnosti na klaudu i zbog toga organizacija klijent treba da donese i da implementira svoje sigurnosne politike.

> Sigurnosna politika (eng. *security policy*) je dokument koji definiše klaud sigurnosne kontrole, organizacione politike, odgovornosti i tehnologiju koja omogućava uspostavljanje bezbednog klaud okruženja.

Razvoj i implementacija sigurnosne politike može da bude dug i mukotrpan proces koji zahteva uključivanje brojnih grupa u okviru organizacije. Sigurnosna politika mora da se formuliše na takav način da sprečava potencijalne sigurnosne proboje (eng. *security breach*) i mora da definiše sve korake koje je potrebno preduzeti ako dođe do događaja koji narušava sigurnost, kao i načine za oporavak od posledica tog događaja.

Sigurnosna politka može da obuhvati i domene kao što su korisnički kredencijali i politike za autentifikaciju i autorizaciju. Politikom se takođe definišu potrebe za šifrovanjem podataka u transferu i u mirovanju, kao i zahtevi za korišćenjem kriptografskih algoritama. Diversifikacija sistema za skladištenje podataka je takođe jedan od zahteva koji sigurnosna politika treba da adresira.

U procesu formulisanja sigurnosne politike treba pažljivo odrediti balans između izloženosti rizicima, lakoće korišćenja i troškova. Nakon formulisanja, pre implementacije, top menadžment organizacije mora da odobri predloženu sigurnosnu politiku.

15.3. IZBOR I PRIMENA SIGURNOSNIH POLITIKA ZA KLAUD OPERACIJE

Kada se sigurnosna politika odobri od strane top menadžment kompanije, sledi proces usvajanja i praktične implementacije svih njenih zahteva.

Svi zahtevi usvojene sigurnosne politike moraju da se konvertuju u stvarne konfiguracije sistema i aplikacija i da se primene na sve uređaje i servise koji se izvršavaju u klaud okruženju. Tako na primer, u slučaju kontrole identititeta korisnika sistema, jedan od zahteva implementirane sigurnosne politike treba da bude da korisnici i uređaji koriste jake i kompleksne šifre koje moraju da se menjaju u predefinisanim vremenskim intervalima.

U toku primene sigurnosne politike treba uzeti u obzir i proces upravljanja promenama. Primena sigurnosnih konfiguracija na veliki broj klaud uređaja i servisa zahteva robusnost sistema za upravljanje promenama. Promene će u velikom broju slučajeva biti velike i obuhvataće veliki broj objekata, kao što su *firewall* uređaji, sigurnosne grupe, pristupne liste za mrežne i resurse za skladištenje, sigurnosne zakrpe aplikacija, anitvirusni alati i slično.

15.4. PRIMERI REGULATORNIH ZAHTEVA

Klaud kompanije često ulažu velike napore kako bi svoje aktivnosti usaglasile sa velikim brojem industrijskih i vladinih regulativa, zahteva, sertifikata i preporuka, a sa ciljem privlačenja što većeg broja potencijalnih organizacionih klijenata, čije je poslovanje usklađeno sa sigurnosnim zahtevima i regulativama.

Klaud kompanija često usaglašava svoje operacije sa lokalnim regulativnim zahtevima zemlje gde želi da proširi svoje poslovanje. U nastavku su navedeni neki od primera regulatornih zahteva sa kojima klaud kompanija može da usaglasi svoje sigurnosne politike i standarde.

SOC (*Service Organization Controls*) 1 izveštaj, koji je takođe poznat pod nazivima *SSAE* 16 i *ISAE* 3402, naglašava značaj interne kontrole za operacije finansijskog izveštavanja. Klaud provajder je usaglašen sa *SOC* 1 zahtevima ako ima razrađene detaljne procedure za finansijsko izveštavanje. Klaud klijenti, kao što su zdravstvene kompanije, finansijski fondovi, itd., mogu da zahtevaju da njihov klaud provajder bude usaglašen sa *SOC* 1 zahtevima.

SOC 2 izveštaj definiše zahteve u pogledu kontrole nefinansijskog izveštavanja, a u cilju održavanja raspoloživosti, poverljivosti, privatnosti, integriteta i opšte bezbednosti računarskih sistema. Na ovaj način *SOC* 2 izveštaj definiše zahteve u pogledu sigurnosnih kontrola koje prevazilaze delokrug finansijskih transakcija.

SOC 3 izveštaj definiše zahteve za javno otkrivanje finansijskih podataka i javno izveštavanje o aspektu sigurnosti. S obzirom da *SOC* 2 izveštaj definiše zahteve u pogledu osetljivih informacija, *SOC* 3 izveštaj je definisan kao rezime *SOC* 2 izveštaja, gde su navedene netehničke i tržišno orijentisane informacije.

FISMA (*Federal Information Security Mangement*) je regulativa koja se primenjuje u zemljama *SAD*-a i koja ima za cilj da zaštiti vladine informacije, operacije i opremu. Da bi klaud provajder mogao da pruža usluge američkoj vladi, on mora da se usaglasi sa *FISMA* zahtevima.

FedRAMP (*Federal Risk and Authorization Management Program*) je program američke vlade koji naglašava standarde za procenu bezbednosti, autorizaciju i kontinuirano nadgledanje klaud proizvoda i usluga. Da bi klaud provajder bio kvalifikovan za *hosting* sistema američke vlade, provajder mora da usaglasi svoje sigurnosne politike sa *FedRAMP* programom.

DIACAP (*Department of Defense Inforamtion Assurance Certification and Accreditation Process*) je proces koji se primenjuje u *SAD*-u i koji definiše zahteve koje klaud provajder, kao izvođač vladinih radova, treba da zadovoljava.

PCI-DSS (*Payment Card Industry Data Security Standard*) je široko korišćeni standard koji definiše zahteve kojima se garantuje da klaud provajder koji obrađuje, skaldišti i prenosi informacije o kreditnim karticama nudi maksimalni nivo sigurnosti i bezbednosti primenom najnovijih bebednosnih tehnolologija. Da bi se kvalifikovali za poslove upravljanja podacima kreditnih kartica, i klaud provajder i klaud klijent treba da se usaglase sa *PCI-DSS* standardima.

ISO 27001 (*International Organization for Standardization*) standard kvaliteta obezbeđuje da klaud provajder zadovoljava sve statutarne i regulatorne zahteve u ponudama svojih servisa. Standard *ISO* 270001 obuhvata sedam principa za upravljanje kvalitetom nad kojim se bazira standard *ISO* 9001. Ovaj standard predstavlja proširenje postojećeg *ISO* sertifikata za upravljanje kvalitetom klaud provajdera i potvrđuje postojanje poverenja, kredibiliteta i zadovoljstva između klaud provajdera i njegovih klijenata, stejkholdera i zajednice.

FIPS 140-2 je *NIST*-ova publikacija koja koordinira zahteve i standarde za kriptografske module. Klaud kriptografski sistemi, bilo da su softverski ili hardverski moraju da se zvanično registruju u *FIPS* 140-2 registru.

HIPPA (*Health Insurance Portability and Acountability*) definiše standard za zaštitu zdravstvenih patentiranih podataka. Klaud provajder koji radi sa zaštićenim medicinskim podacima mora da obezbedi sve fizičke, mrežne i sigurnosne mere u skladu sa *HIPPA* zahtevima.

PITANJA ZA RAZMATRANJE



1. Objasniti postupak usaglašavanja klaud provajdera sa zahtevima i standardima iz domena klaud sigurnosti.
2. Objasniti postupak uspostavljanja sigurnosne politike u organizaciji koja koristi klaud usluge.
3. Objasniti postupak primene sigurnosnih politika za klaud operacije kada se usvoje zahtevi. Kako se konfigurišu sistemi i aplikacije u klaud okruženju u skladu sa definisanim zahtevima klaud sigurnosti?
4. Navesti primere regulatornih zahteva za klaud okruženje.

16. TEHNOLOGIJE KLAUD SIGURNOSTI

Sa razvojem računarskih tehnologija raste i broj i intenzitet *cyber* napada, zbog čega klaud provajderi i klaud klijenti sve više novca ulažu u tehnologije koje unapređuju sigurnost klaud okruženja.

> Za ostvarivanje ciljeva računarske sigurnosti na klaudu raspoložive su brojne tehnologije.

Ranije su troškovi i kvalitet usluga bili glavni parametri prilikom izbora klaud provajdera. Međutim, u poslednje vreme, kada biraju klaud provajdera, korporativni klijenti sve veću prednost daju provajderu koji ima konkurentsku prednost u odnosu na druge provajdere u aspektu sigurnosti. Prednost ima onaj provajder koji obezbeđuje veći broj sigurnosnih sistema i servisa, koji podržava sofisticirane tehnike i mehanizme zaštite podataka, čije su sigurnosne politike usaglašene sa najboljim praksama računarske sigurnosti, koji obezbeđuje siguran i zaštićeni pristup klaud okruženju sa udaljenih lokacija, itd.

Zbog sve većeg značaja tehnologija klaud sigurnosti, u ovoj glavi prikazane su osnovne tehnologije koje unapređuju sigurnost svakog klaud okruženja. O nekim od prikazanih tehnologija bilo je reči ranije, i u tom kontekstu ova glava predstavlja rezime tehnologija, tehnika i principa sa kojima su čitaoci bili upoznati ranije, uz određene nove elemente.

16.1. ZAŠTITA PODATAKA NA KLAUDU

Odgovornost za zaštitu podataka na klaudu preuzima organizacija klijent, što znači da klaud klijent mora da izvrši neophodne aktivnosti kako bi zaštitio svoje podatke u klaud računskom centru, na isti način kao i kada štiti podatke u lokalnom računskom centru.

Organizacija klijent je vlasnik svojih podataka i operacija na klaudu. Klijent iznajmljuje infrastrukturu, skladište i računarske usluge od klaud provajdera. Na kraju dana, organizacija klijent je ta koja je odgovorna za svoje podatke. Klaud provajder može da obezbedi najsigurniju platformu, ali ako na primer klijent na svojim klaud serverima koji skladište osetljive podatke ne konfiguriše, ili pogrešno konfiguriše sigurnosna podešavanja, bez obzira na naprednu tehnološku platformu klaud provajdera, sigurnost servera klijenta može da bude ugrožena.

Klaud provajder može da bude sertifikovan za izvršavanje određenih klaud operacija, ali to ne podrazumeva da je i organizacija klijent, koja koristi usluge tog provajdera, sertifikovana. Tako na primer, organizacija koja obrađuje medicinske podatke može da odluči da migrira svoje operacije na okruženje javnog klauda. Nakon izbora klaud provajdera i izvršavanja procesa migracije na klaud, sve aplikacije i servisi organizacije se izvršavaju na klaud infrastrukturi provajdera koji poseduje *HIPPA* sertifikat. Međutim, da bi kompanija klijent dobila *HIPPA* sertifikat, bez obzira što koristi usluge sertifikovanog provajdera, kompanija mora da prođe proces validacije usaglašenosti sa *HIPPA* standardom i dobijanja dozvole.

Klaud provajderi nude široku paletu usluga za zaštitu podataka, od replikacije fajlova i podataka na udaljene računske centre, do sofisticiranih tehnologija za skladištenje podataka sa visokim nivoom raspoloživosti i pouzdanosti. Podaci se na klaudu najčešće organizuju u različite slojeve, gde svaki sloj obezbeđuje određeni nivo sigurnosti po predefinisanoj ceni.

Većina klaud provajdera nudi brojne opcije za šifrovanje podataka, bilo da su podaci u stanju prenosa, ili u mirovanju. Klaud klijent može samostalno da implementira i upravlja sistemom za kreiranje, čuvanje, upravljanje i distribuciju javnih i privatnih ključeva, ili može da koristi postojeći sistem klaud provajdera.

Klijenti koji nisu u mogućnosti da izdvoje veći nivo finansijskih sredstava za zaštitu podataka na klaudu najčešće čuvaju svoje podatke na servisima javnog klauda, koji s vremena na vreme pravi replike podataka na više različitih geografski distribuiranih lokacija.

16.2. TEHNOLOGIJE ZA ZAŠTITU PODATAKA

Tehnologije za zaštitu podataka, radni okviri i implementacija, kojima se ostvaruje veća sigurnost klaud podataka, su kompleksna i široka oblast, koja se redovno koristi u realizaciji svakodnevnih aktivnosti klaud računarstva.

Glavni „alat" za zaštitu podataka na klaudu je šifrovanje, tj. enkripcija podataka. Iste tehnologije za šifrovanje podataka koje se koriste u tradicionalnom okruženju koriste se i na klaudu.

S obzirom na pretpostavku da čitaoci imaju osnovno predznanje o kriptografiji i s obzirom na to da detaljna diskusija o kriptografiji i o algoritmima za šifrovanje podataka prevazilazi domen ovog udžbenika, u nastavku je naveden samo opšti prikaz nekih od tehnologija koje se koriste u ove svrhe u klaud okruženju.

16.2.1 IPsec

IP Security (*IPsec*) je radni okvir, ili arhitektura koja koristi različite protokole u cilju obezbeđivanja integriteta, privatnosti i autentifikacije podataka na *TCP/IP* mreži.

S obzirom da *IPsec* nije protokol, već radni okvir koji implementira različite sigurnosne mehanizme i protokole za šifrovanje i dešifrovanje podataka, razumevanje svih detalja njegovog funkcionisanja i implementacije može da predstavlja izazov.

IPsec se uglavnom koristi u implementacijama *VPN* servisa, sigurnosnih mehanizama na nivou aplikacija i mreža i implementira se u okviru operativnih sistema, rutera i *firewall* uređaja.

U većini praktičnih aplikacija, *IPsec* koristi dva načina (režima) rada: transportni režim (eng. *transport mode*) i tunelski režim (eng. *tunnel mode*).

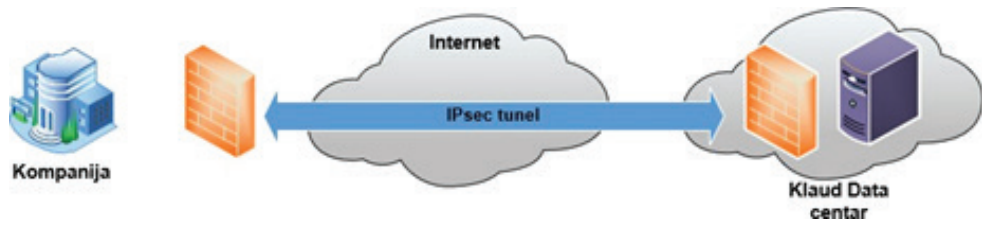
Glavna razlika između ova dva režima rada odnosi se na mesto izvršavanja *IPsec* funkcija.

U transportnom režimu, izvorišni (računar koji šalje podatke) i odredišni *host* (računar koji prima podatke) uređaji izvršavaju sve kriptografske funkcije. Šifrovani podaci se šalju pomoću enkriptovane sesije sa izvorišnog do odredišnog (udaljenog) *host* uređaja. Podaci, koji se u ovom kontekstu nazivaju šifrovani tekst (eng. *ciphertext*) se kreiraju na izvorišnom *hostu* i preuzimaju se na odredišnom *hostu*. Transportni režim omogućava sigurnost od jedne do druge tačke (eng. *end-to-end security*), tj. od jednog do drugog host uređaja.

Tunelski režim se malo razlikuje od transportnog režima. U ovom režimu, ruter (mrežni prolaz), ili *firewall* izvršavaju sve *IPsec* funckcije. Ovi uređaji izvršavaju sve aktivnosti šifrovanja i dešifrovanja podataka, kao što je prikazano na Slici 16-1. Tunelski režim rada *IPsec* okvira skida odgovornost za izvršavanje šifrovanja i dešifrovanja podataka sa *host* uređaja i omogućava da se s jedne tačke (uređaja) kontroliše ceo proces zaštite podataka. Glavni nedostatak tunelskog režima rada je taj što on ne obezbeđuje potpunu sigurnost od tačke do tačke, kao što je to slučaj sa transportnim modom, zato što saobraćaj od mrežnog, ili sigurnosnog uređaja do host računara nije zaštićen.

Dakle, sigurnost koja se postiže primenom transportnog moda naziva se sigurnost od tačke do tačke, dok se korišćenjem tunelskog režima rada postiže sigurnost od mrežnog prolaza do mrežnog prolaza (eng. *gateway-to-gateway security*). Bez obzira koji se režim rada koristi, mrežni prolazi mogu da provere da li je paket autentičan i da izvrše autentifikaciju paketa i na izvorišnoj i na odredišnoj strani.

Slika 16-1: *Prikaz IPsec tunela*



IPsec koristi dve vrste enkapsulacije *IP* paketa: autentifikaciono zaglavlje (eng. *Authentication Header – AH*) i enkapsulirani sigurnosni teret (eng. *Encapsulating Security Payload – ESP*). Oba mehanizma, i *AH* i *ESP* obezbeđuju sigurnost podataka na mreži, ali se njihove implementacije blago razlikuju. Oba protokola pružaju zaštitu *IP* datagrama dodavanjem zaglavlja koje sadrži informacije o bezbednosti.

AH obezbeđuje usluge autentifikacije i integriteta. Autentifikacija se obezbeđuje primenom heš funkcije sa ključem koja se naziva *MAC* (*Message Authentication Code*). *AH* takođe ima mogućnost da utvrdi da li je paket modifikovan u tranzitu i ima ugrađene mehanizme protiv napada koji maliciozne podatke šalju veći broj puta (eng. *replay attacks*).

ESP je protokol koji definiše način na koji *IPsec* enkapsulira izvorišni *IP* paket i kako *IPsec* obezbeđuje autentifikaciju, enkripciju i poverljivost podataka koji se nalaze u *IP* paketu.

IPsec ima važnu ulogu u klaud okruženju zato što implementira važne servise računarske sigurnosti, koji obezbeđuju integritet, poverljivost, šifrovanje i neporicanje podataka.

16.2.2. *SSL*

SSL (*Secure Socket Layer*) i njegov naslednik *TLS* (*Transport Layer Security*) su važni sigurnosni protokoli koji imaju široku primenu na klaudu.

> *SSL* i *TLS* se izvršavaju iznad *TCP* protokola i obezbeđuju enkriptovanu sesiju između klijentskog i serverskog uređaja.

Oba protokola se najčešće implementiraju u formi *HTTPS* (*HyperText Transfer Protocol Secure*) protokola na većini zaštićenih *Web* sajtova. Na ovaj način, protokoli na višim nivoima *TCP/IP* referentnog modela, kao što je *HTTP*, ne treba da se modifikuju zato što pomoću *SSL/TLS* protokola obezbeđuju sigurnu konekciju. Osim za kreiranje sigurnih *Web* servisa, *SSL* i *TLS* protokoli mogu da se koriste i za unapređenje sigurnosti bilo kog komunikacionog toka koji se bazira na *TCP* protokolu.

Kada se implementira *SSL/TLS* sigurnost, ceo *IP* paket, osim *IP* adresa i brojeva porta, se šifruje. Kada se *Web* brauzer poveže sa sigurnim *Web* serverom, nakon uspostavljanja *TCP* konekcije, izvršava se proces tzv. *SSL* rukovanja (eng. *SSL handshake*). Nakon procesa rukovanja, klijentski brauzer šalje serveru informacije o *SSL/TLS* verziji protokola koju koristi, preferiranom tipu enkripcije podataka, eventualnom algoritmu za kompresiju podataka, itd. Web server bira najvišu *SSL/TLS* verziju koju podržavaju i server i klijent i slaže se sa preferiranim tipom enkripcije podataka koju je predložio klijent.

Tek kada je konekcija između klijenta i servera uspostavljena i kada su dogovoreni detalji o *SSL/TLS* verziji, algoritmu za enkripciju, itd., server klijentu šalje digitalni sertifikat korišćenjem infrastrukture javnog ključa. Da bi konekcija bila uspostavljena, klijent mora da veruje digitalnom sertifikatu servera, ili mora da postoji centralni sertifikacioni autoritet kome klijent veruje, a koji veruje sertifikatu servera.

Nakon što klijent izvrši verifikaciju digitalnog sertifikata servera i potvrdi njegov identitet, klijent generiše simetrični ključ koji se naziva sesijski ključ (eng. *session key*), koji se zatim koristi za sigurnu razmenu podataka između servera i klijenta. Tek sada je *SSL* sesija uspostavljena.

U *SSL/TLS* sesiji svaki paket se potpisuje *MAC* kodom i na ovaj način se osigurava da je primalac dobio podatke od pošiljaoca, a ne od neke treće strane koja je presrela komunikaciju. Čak iako presretne komunikaciju, potencijalni napadač neće moći da pročita sadržaj paketa, zato što je sadržaj šifrovan i videće samo gomilu karaktera koje neće moći da protumači.

16.2.3. Šifarski algoritmi

Kriptografija (eng. *cryptography*) je nauka menjanja informacija koje kasnije mogu da se dekodiraju samo pomoću ključa.

Istraživanje kriptografskih algoritama se naziva kriptografija. Istraživanje načina za razbijanje kriptografskih algoritama se naziva kriptoanaliza (eng. *cryptanalysis*) Kriptografija i kriptoanaliza se u savremenoj literaturi zajedno nazivaju kriptologija (eng. *cryptology*).

Šifarski metod (eng. *cipher*) se definiše kao bilo koji metod za enkripciju (šifrovanje) podataka skrivanjem njegovog značenja.

Danas je raspoloživ veliki broj šifarskih metoda, od jednostavnijih do složenijih. Jednostavniji šifarski metodi koriste manje složene pristupe sakrivanja podataka, kao što je izmena redosleda navođenja podataka – na primer zamenom svakog karaktera (slova) „B“ u rečenici cifrom 7.

U savremenoj literaturi navode se dve osnovne vrste šifarskih metoda: šifarski metod koji se bazira na blokovima podataka (eng. *block cipher*) i šifarski metod koji se bazira na toku podataka (eng. *stream cipher*). Ove dve vrste razlikuju se na osnovu načina na koji se podaci šifruju.

Šifarski metod koji se bazira na blokovima podataka razbija podatke u manje delove koji se nazivaju blokovi, a zatim se proces šifrovanja primenjuje nad svakim blokom podataka posebno. S druge strane, šifarski metod koji se bazira na toku podataka šifruje svaki bajt podataka i najviše se primenjuje u simetričnoj kriptografiji.

U savremenim klaud računarskim okruženjima i klaud računskim centrima koriste se daleko kompleksnije metode šifrovanja podataka, koje koriste ključ u kombinaciji sa šifarskim algoritmom koji kombinuje ključ sa tekstom.

Prema osnovnoj podeli, šifarski algoritmi se dele u dve kategorije: simetrične i asimetrične šifarske algoritme.

Simetrični šifarski algoritmi (eng. *symmetric-key algorithms*), gde se za šifrovanje i za dešifrovanje koristi isti (tajni, privatni) ključ nastali su kao rezultat razvoja klasične kriptografije. Pre uspostavljanja sistema simetrične kriptografije korisnici treba da razmene ključ koji će da se koristi za šifrovanje i dešifrovanje sadržaja. Među najpopularnijim simetričnim šifarskim algoritmima izdvajaju se *AES, DES, RC4* i *Blowfish*.

Glavna prednost simetričnih šifarskih algoritama je brzina, dok je osnovni nedostatak uspostavljanje tajnog ključa, jer je potrebno da se tajni ključ bezbedno dostavi od jedne do druge strane koje učestvuju u komunikaciji.

Simetrični šifarski algoritam prikazan je na Slici 16-2.

Slika 16-2: *Simetrični šifarski algoritam*



Začetnici koncepta asimetričnog šifrovanja bili su *Diffie* i *Hellman*, kada su 1976. godine objavili naučni rad u kom su prikazali osnovne funkcionalne principe asimetričnog šifrovanja. U slučaju asimetričnih šifarskih algoritama (eng. *asymmetric- key algorithms*) svaka strana koja učestvuje u zaštićenoj komunikaciji ima par ključeva koji se nazivaju javni (eng. *public*) i tajni (eng. *private*) ključevi. Kao što sam naziv implicira, javni ključ je javno dostupan, dok se tajni ključ čuva na bezbednoj lokaciji van dometa ostalih korisnika.

Podaci koji se šifruju javnim ključem mogu da se dešifruju samo pomoću unapred određenog i uparenog tajnog ključa. Tajni ključ može da se koristi i za potpisivanje digitalnog sadržaja, dok se javni ključ u ovom slučaju koristi za proveru autentičnosti potpisa. Najpoznatiji predstavnici asimetričnih šifarskih algoritama su *Diffie-Hellman* sistem razmene ključeva i *RSA* algoritam.

U odnosu na simetrične šifarske algoritme, asimetrični algoritmi su sporiji i pružaju niži nivo sigurnosti za istu dužinu ključa, i zbog toga zahtevaju ključeve veće dužine. S druge strane, asimetrični algoritmi ne zahtevaju odvojene kanale za distribuciju ključeva, što se u literaturi navodi kao njihova najveća prednost u odnosu na simetrične šifarske algoritme.

Funkcionisanje asimetričnih šifarskih algoritama prikazano je na Slici16-3.

Slika 16-3: *Asimetrični šifarski algoritam*



U praksi se za razmenu velike količne podataka zbog brzine uglavnom koriste simetrični šifarski algoritmi, dok se privatni ključ distribuira korišćenjem asimetričnih algoritama.

S obzirom na kompleksnost i raznolikost klaud okruženja, na klaudu se koriste skoro svi najpopularniji šifarski algoritmi. U nastavku su ukratko opisani neki od njih.

3DES (*Triple Data Encryption Standard*) algoritam je razvijen 1977. godine i sada se umesto njega uglavnom koristi *AES* algoritam. *3DES* je simetrični šifarski algoritam koji predstavlja proširenje prvobitnog *DES* standarda, za koji je utvrđeno da je slab i da relativno lako može da se kompromituje. *3DES* algoritam koristi tri ključa, gde se prvi ključ koristi za šifrovanje bloka podataka, drugi ključ se koristi za dešifrovanje bloka podataka, i treći ključ se koristi ponovo za šifrovanje.

AES (*Advanced Encryption Standard*) je simetrični blokovski šifarski algoritam koji podržava tri dužine kluča, i to od 128, 192 i 256 bita. Sa rastom dužine ključa, eksponencijalno raste težina dešifrovanja. *AES* standard koji koristi ključ dužine od 256 bita smatra se veoma sigurnim i zbog toga se ova dužina ključa na klaudu najviše i koristi. S obzirom da *AES* pripada grupi simetričnih šifarskih algoritama, isti ključ se koristi i za šifrovanje i za dešifrovanje. *AES* koriste mnogobrojni globalni klaud provajderi, ali i vlade, kao što je vlada *SAD*-a i Kanade. *AES* algoritam je takođe usvojila i uključila u svoja dokumenta o standardima agencija *NIST*.

RSA algoritam je dobio naziv po svojim tvorcima *Ron Rivest, Adi Shamir* i *Leonard Adleman*. *RSA* spada u grupu asimetričnih šifarskih algoritama koji koriste javne i privatne ključeve. Algoritam *RSA* i njegovi derivati našli su široku primenu u *PKI* servisima. *PKI* koristi *RSA* algoritam za uspostavljanje šifrovane konekcije za razmenu simetričnih ključeva. *RSA* za sam proces šifrovanja većine podataka koristi simetrične šifarske algoritme zato što se mnogo brže izvršavaju od asimetričnih.

DSA (*Digital Signature Algoritm*) funkcioniše slično kao *RSA*, s tim da je sporiji u šifrovanju, a brži od njega u dešifrovanju podataka. Bez obzira što su performanse *RSA* i *DSA* algoritama slične i što imaju slične funkcionalne principe, *RSA* se mnogo češće sreće u implementacijama klaud servisa.

RC4 (*Rivest Cipher 4*) je dobio naziv po svom tvorcu *Ronaldu Rivest*. Ovaj algoritam koristi deljeni ključ (eng. *shared key*) za šifrovanje i dešifrovanje toka podataka i spada u grupu simetričnih šifarskih algoritama. *RC4* se uglavnom koristio za obezbeđivanje bežičnih konekcija i *Web* transakcija. Međutim, dugogodišnjom eksploatacijom *RC4* algoritma, utvrđeno je da *RC4* nije dovoljno bezbedan za transfer osetljivih podataka i zbog toga se danas retko (skoro nikada) koristi.

RC5 (*Rivest Cipher 5*) je zamena za *RC4* algoritam. *RC5* takođe spada u grupu simetričnih algoritama i koristi ključ promenljive dužine.

16.3. INFRASTRUKTURA JAVNOG KLJUČA

Sigurnosni sertifikati i ključevi obezbeđuju sigurnost podataka i servisa i omogućavaju implementiranje namenskih servera za šifrovanje podataka, koje koriste brojni klaud servisi, kako za šifrovanje i zaštitu podataka, tako i za autentifikaciju korisnika.

Upravljanje sertifikatima i ključevima je složen i višeslojeviti sistem. Na sreću, klaud provajder apstrahuje ove sisteme i klaud klijenti relativno lako mogu da ga koriste.

Standardni sigurnosni modeli koji se koriste na klaudu moraju da obezbede pouzdane servise za identifikaciju, autentifikaciju i autorizaciju svih korisnika klaud računarskih resursa. Klaud provajderi i klijenti moraju da potvrde identitet svih korisnika klaud okruženja i da obezbede da su sigurnosne politike usklađene sa sigurnosnim politikama i procedurama drugih organizacija i sa regulatornim zahtevima.

Infrastruktura javnog ključa (eng. *Public Key Infrastructure – PKI*) obezbeđuje servise za identifikaciju, autorizaciju i šifrovanje podataka, koji predstavljaju ključnu osnovu za implementiranje sigurnosti u klaud okruženju.

PKI se često u literaturi adresira kao tradicionalni asimetrični kriptografski standard za razvoj i verifikaciju kriptografskih ključeva. *PKI* podržava upravljanje ključevima i obezbeđuje integritet javnih ključeva pomoću sertifikata.

PKI radni okvir se sastoji iz hardvera i softvera, politika i procedura koje zajedno obezbeđuju osnovnu za sigurnu komunikaciju između korisnika, koja se zasniva na poverenju. Osnovne *PKI* komponente su:

- sigurnosne politike,
- sertifikacioni autoritet (eng. *Certificate Authority – CA*),
- registracioni autoritet (eng. *Registration Authority – RA*),
- baza podataka za sertifikate,
- skladište sertifikata i
- softver.

Na Slici 16-4 prikazana je pojednostavljena arhitektura *PKI* sistema.

Sigurnosne politike su od ključne važnosti za procese generisanja i upravljanja sertifikatima, kao i za definisanje osnovnih principa i metoda korišćenja kritpografskih tehnika. Sigurnosne politike definišu pravila za upravljanje ključevima i osetljivim informacijama, dok u isto vreme podešavaju nivo kontrole koji je potreban u odnosu na nivoe rizika.

CA je entitet koji izdaje (eng. *generate*) i ukida (eng. *revoke*) sertifikate. *CA* garantuje da je svaki sertifikat vezan za validan kriptografski ključ. *CA* koordinira svoje aktivnosti sa *RA* koji registruje sertifikate za korišćenje koji su odboreni od strane *CA* entiteta. Entitet *RA* je odgovoran za autentifikaciju identiteta legitimnih korisnika sistema. Sve informacije vezane za sertifikate, liste za ukidanje sertifikata (eng. *Certificate Revocation List – CRL*) i ključevi su obezbeđeni i čuvaju se u *CA* bazi podataka.

Slika 16-4: *Pojednostavljena arhitektura PKI system*



Korisnici klaud sistema biraju gde će svoj par ključeva (privatni i javni) da generišu. Izbor pre svega zavisi od konteksta u kome će ključ da se koristi i od sigurnosne politike klaud sistema. Par ključeva može da generiše *CA*, ili mogu sami korisnici da ga generišu, kada dostavljaju javni ključ *CA* entitetu na validaciju i skladištenje. Ako se ključ koristi u svrhu neporicanja, onda je bolje da ključeve generišu sami korisnici. S druge strane, ako se ključevi koriste za dešifrovanje osetljivih informacija u organizaciji, onda generisanje ključeva od strane *CA* entiteta predstavlja bolji izbor.

Ključni element *PKI* sistema je upravljanje ključevima, kao što je to slučaj sa svakim drugim kriptografskim sistemom. Proces upravljanja ključevima grubo može da se podeli na tri faze: generisanje javnih ključeva, generisanje privatnih ključeva i povlačenje (ukidanje) ključeva.

S obzirom da se javni ključ generiše u isto vreme kada se generiše i privatni ključ, generisanje ključeva, bilo da generisanje vrši *CA*, ili korisnik je ograničeno. *CA* proverava da li je javni ključ vezan za odgovarajući privatni ključ. Proces generisanja ključeva koristi slučajnu tajnu (eng. random secret) koja se daje kao input ovom procesu. Da bi dešifrovao poruku, klijent mora da zna javni ključ koji je mapiran za privatni ključ koji je korišćen u procesu šifrovanja poruke. Ključ za dešifrovanje se vezuje za identitet primaoca.

Dakle, *PKI* je kriptografski radni okvir koji koristi par kriptografskih ključeva: jedan javni i jedan privatni. Tako na primer, kada klijent pristupa klaud *Web* serveru, *Web* server prvo šalje svoj javni ključ udaljenom klijentu, koji primljeni ključ zatim koristi za šifrovanje i slanje podataka. Kada klaud *Web* server primi šifrovane podatke od klijenta, on koristi svoj privatni ključ da bi dešifrovao primljene podatke. U procesu dešifrovanja podataka koristi se samo privatni ključ koji mora dobro da se zaštiti, dok javni ključ može slobodno da se distribuira.

Neke od najčešće korišćenih implementacija *PKI* okvira u klaud okruženju su kontrola pristupa i šifrovanje podataka u *VPN* konekcijama, primena u sistemima za autentifikaciju i autorizaciju korisnika i odobravanje aplikacija, zaštita mrežnog saobraćaja korišćenjem *IPsec* tehnologije i zaštita *LDAP* (*Lightweight Directory Access Protocol*) servisa direktorijuma. *PKI* se takođe koristi za autentifikaciju uz pomoć dva faktora sigurnosti i to u kombinaciji pametnih kartica baziranih na tokenima sa korisničkom lozinkom. Takođe i brojne klaud aplikacije koriste *PKI* sertifikate, poput onih za slanje email poruka, baze podataka, potpisivanje elektronskih dokumenata i slično.

Korisno je da se pomene još jedan kriptografski sistem koji se često koristi na klaudu. Kriptografija javnog ključa koja se bazira na identitetu (*Identity Based Public Key Cryptography – ID-PKC*)spada u grupu asimetričnih kriptografskih algoritama. *ID-PKC* sistem parove ključeva kreira na osnovu podataka koji su bitni za korišćenje ključa, kao što podaci koji se odnose na identitet sistema ili korisnika, ili na poziciju korisnika u organizaciji. Koncept *ID-PKC* je kreiran sa namerom da se prevaziđu problemi upravljanja sertifikatima i ključevima do kojih dolazi u *PKI* sistemima.

Glavna razlika između *ID-PKC* i tradicionalnog *PKI* sistema ogleda se u načinu na koji se generišu javni i privatni ključevi. U *ID-PKC* sistemu, javni ključ se generiše na osnovu javno dostupnih identifikujućih informacija. S druge strane, za generisanje privatnog ključa potrebne su informacije o glavnoj tajni (eng. *master secret*), koje dostavlja autoritet kome se veruje (eng. *Trusted Authority – TA*).

Servis *PKI* i servisi koji se na njemu baziraju, kao što je *ID-PKC*, su u literaturi i stručnoj javnosti s razlogom dobili epitet sistema koje je teško razumeti i koji se teško implementiraju. Međutim, ovo predstavlja razlog više za njihovu upotrebu u klaud okruženju, jer dodatno otežavaju posao malicioznim napadačima koji imaju nameru da ugroze sigurnost klaud infrastrukture.

S obzirom na činjenicu da se ovakvi servisi često upotrebljavaju na klaudu i da su se pokazali kao efikasni načini za obezbeđivanje ciljeva poverljivosti, integriteta, autentifikacije i neporicanja, mnogi klaud provajderi nude usluge *PKI* sistema u formi modela *PKI* kao usluga (eng. *Public Key Infrastructure as a Service – PKIaaS*). Klijentske organizacije koje koriste ovakav tip klaud usluge nemaju potrebu da implementiraju svoj *PKI* sistem.

Osim *PKIaaS*, na klaudu se sve više koristi i model kriptografije kao usluge (eng. *Cryptography as a Service – Caas*), u okviru koga klaud provajderi obezbeđuju i dodatne *PKI* usluge, kao što su hardverski i softverski tokeni, servisi za autentifikaciju korisnika, itd.

16.4. SIGURNOST UDALJENOG PRISTUPA NA KLAUDU

Klaud servisi se po prirodi nalaze na lokacijama koje su udaljene od prostorija u kojima se nalazi organizacija klijent i zbog toga je za pristup ovim servisima neophodno korišćenje neke od tehnologija za udaljeni pristup. Čim se podacima i servisima pristupa sa udaljene lokacije koristi se nezaštićeni Internet kao medijum za pristup, što otvara novi front za potencijalnog napadača. Zbog toga je važno da se koriste sigurne tehnologije za udaljeni pristup i u većini slučajeva koristi se *VPN* tehnologija, koja vrši enkripciju komunikacione sesije između klijenta i udaljenog klaud servisa.

> *VPN* je tehnologija koja kreira sigurnu i enkriptovanu konekciju preko nesigurne računarske mreže kao što je Internet.

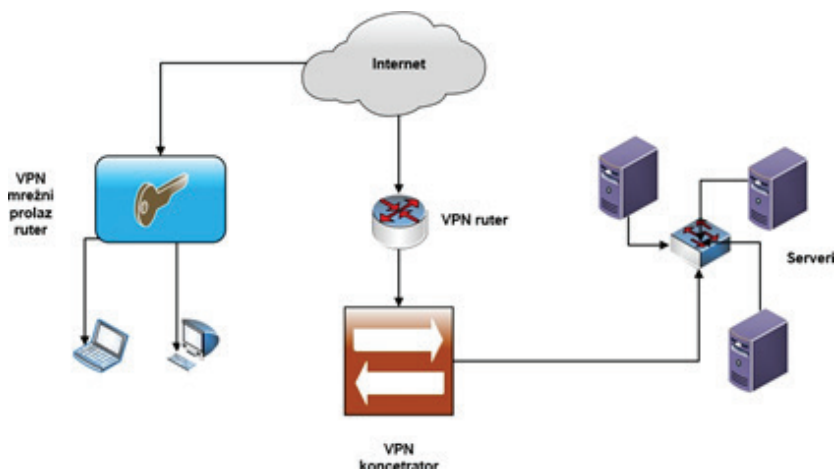
Razvoj *VPN* tehnolologije je započet sa ciljem da se omogući udaljenim korisnicima i klijentima da se povežu na udaljene korporativne sisteme i servise i da na bezbedan način pristupaju korporativnim računarskim resursima. Kako bi se obezbedila sigurnost, *VPN* tehnologija za razmenu podataka koristi zaštićene šifrovane kanale uz korišćenje metoda za autentifikaciju uređaja koji komuniciraju, kao što su šifre, tokeni, itd.

Jedna *VPN* implementacija prikazana je na Slici 16-5.

Protokoli koji se najčešće koriste za udaljeni pristup na klaudu, a koji podržavaju i implementacije *VPN* servera su *L2TP, PPTP, SSTP* i *GRE*.

U nastavku je svaki od navedenih protokola ukratko prikazan.

Slika 16-5: *Prikaz jedne VPN implementacije*



16.4.1. L2TP protocol

L2TP (*Layer2 Tunneling Protocol*) je kombinacija komunikacionih protokola za udaljeni pristup, koja predstavlja siguran način povezivanja sa udaljenim lokacijama korišćenjem Internet mreže.

L2TP su razvili *Cisco Systems* i *Microsoft* i podržan je u formi standarda od 1999. godine. Većina *VPN* servera podržava *L2TP*.

Međutim, *L2TP* je sada već zastareo protokol (eng. *deprecated protocol*) zato što u osnovnoj implementaciji ne podržava funkcionalnosti šifrovanja podataka i autentifikacije, već je za ove funkcije potrebno koristiti dodatak u vidu aplikativnog softvera, kao što *IPsec* radni okvir.

16.4.2. PPTP protocol

PPTP (*Point-to-Point Tunneling Protocol*) je *Microsoft*-ova implementacija protokola za udaljeni pristup koja je sada zastarela i koja je zamenjena novijim tehnologijama, ali je protokol još uvek podržan u nekim operativnim sistemima i mrežnim okruženjima.

PPTP omogućava *PC* računaru, ili bilo kojem mrežnom klijentu da se poveže na udaljenu mrežnu lokaciju, kao što je klaud baza podataka. *PPTP* implementira *TCP* kontrolni kanal sa *GRE* (*Generic Routing Encapsulation*) tunelom za enkapsulaciju *PPP* (*Point-to-Point*) paketa.

VPN konekcije koje koriste *PPTP* protokol mogu da imaju probleme sa *NAT*-om. Za upostavljanje *VPN* mreže, održavanje konekcione sesije i razmenu *GRE* enkapsuliranih paketa, *PPTP* koristi *TCP* konekciju na portu 1723, koja je po podrazumevanim podešavanjima najčešće zabranjena na *NAT* uređajima, *Web proxy* serverima i *firewall* mrežnim komponentama. Većina klaud provajdera na svojim uređajima zabranjuje komunikaciju preko ovog porta, što je možda bio još jedan razlog više za odbacivanje *PPTP* protokola iz upotrebe.

16.4.3. SSTP protocol

SSTP (*Secure Socket Tunneling Protocol – SSTP*) je još jedna *Microsoft*-ova implementacija protokola za udaljeni pristup koja je nastala sa ciljem prevazilaženja nedostataka *PPTP* protokola.

SSTP koristi *SSL/TLS* kanal preko *TCP* porta 443. Kako bi se izvršila validacija konekcije, potrebna je autentifikacija korisnika, koja se najčešće izvodi na strani klijenta. *SSTP* protokol koristi serverski sertifikat za autentifikaciju. Bez obzira što je *SSTP* vlasnički (eng. *proprietary*) protokol kompanije *Microsoft*, postoje verzije i za *GNU/Linux* i *macOS* operativne sisteme.

SSTP je trenutno jedan od najpopularnijih protokola koji se koristi za udaljeni pristup klaud infrastrutkturi. U korist njegove popularnosti mogu da se navedu brojni argumenti. Na primer, *SSTP* nudi najviši nivo sigurnosti zato što za šifrovanje podataka koristi relativno dugačak *AES* ključ od 256 bita. Takođe, obzirom da je tvorac *SSTP* protokola kompanija *Microsoft*, ovaj protokol je potpuno kompatibilan sa *Windows* operativnim sistemima, koje koristi najveći broj korisnika *PC* računara širom sveta.

Međutim, *SSTP* ima i neke nedostatke. Kao jedan od najvećih nedostataka navodi se vlasnička priroda *SSTP* protokola i zbog toga treće strane ne mogu da izvršavaju testove za utvrđivanje ranjivosti ovog protokola. Kao još jedan nedostatak navodi se da se *SSTP* izvršava relativno sporo zato što koristi dugačak ključ.

16.4.4. GRE protocol

Generic Routing Encapsulation (*GRE*) je standardizovani mrežni tunel protokol koji enkapsulira paket bilo kog protokola *TCP/IP* referentnog modela u virtuelni link između dve lokacije.

GRE protokol je razvila kompanija *Cisco Systems*.

GRE se uglavnom koristi za kreiranje privatnog tunela preko javne mreže za prenos podataka, i kao takav ima široku primenu u pristupu klaud servisima sa udaljenih lokacija.

Tako na primer u hibridnom klaud okruženju koje obuhvata veliki broj privatnih i javnih klaud infrastruktura mogu da se implementiraju *GRE* tuneli, tako da krajnji korisnik ima privid da su sve geografski distribuirane klaud infrastrukture implementirane kao jedan veliki sistem.

16.5. AUTOMATIZACIJA KLAUD SIGURNOSTI

Klaud sigurnost može da se automatizuje kombinacijom brojnih servisa koji su dostupni, a sa ciljem kreiranja integrisane platforme za praćenje, izveštavanje i reagovanje na događaje koji mogu da ugroze sigurnost klaud sistema, aplikacija, servisa i podataka.

Tako na primer, podaci koji se prikupljaju pomoću logova mrežnih uređaja šalju se na integrisanu platformu, na kojoj se dalje vrši njihova analiza, primenom alata kao što je *Splunk (URL*: https://www.splunk.com). Ovakvi i slični alati uključeni su u ponude globalnih klaud provajdera. Integrisane platforme za praćenje i izveštavanje za potrebe klaud sigurnosti uglavnom se implementiraju u virtuelnom okruženju.

Kada se dogodi sigurnosni incident, automatski (bez ljudske intervencije) se pokreće skripta koja preuzima akcije, kao protivmere na incident koji se dogodio. Ove skripte su kompatibilne sa većinom klaud računarskih resursa i mogu da se izvršavaju na sistemima za upravljanje klaud mrežom, *IPS* uređajima, virtuelnim mašinama, mirkoservisima, ili na bilo kojoj aplikaciji treće strane.

Automatizacijom klaud sigurnosti omogućava se brz odgovor na događaje koji mogu da kompromituju sigurnost klaud sistema i na taj način se predupređuju, ili minimizuju negativne posledice sigurnosnog incidenta.

Automatizacija klaud sigurnosti omogućava i čuvanje podataka koji kasnije mogu da se koriste i za potrebe forenzičke analize.

16.5.1. Alati za automatizaciju na klaudu

U klaud okruženju raspoloživi su brojni alati za automatizaciju sigurnosti. Skoro uvek, kada bilo koji korisnik izvrši neku promenu na klaud objektu (mrežnom uređaju, aplikaciji, sistemu, itd.), bez obzira da li je koristio grafički, ili komandni interfejs, interfejs u pozadini komunicira sa alatom za automatizaciju pomoću aplikativnog programskog interfejsa (eng. *Application Programming Interface – API*), koji beleži sve promene u svoje logove.

Zbog značaja *API*-ja u automatizaciji klaud sigurnosti, ovom pojmu je posvećena veća pažnja.

U stručnoj literaturi nailazi se na brojne definicije termina *API*.

Prema jednoj definiciji, *API* je skup protokola, definicija podrutina i alata za izgradnju softvera. U opštem smislu, *API* je jasno definisana metoda za komunikaciju između različitih komponenti.

Kao još jedna definicija *API*-ja navodi se da je *API* predefinisani način za programski pristup, kontrolu i konfigurisanje uređaja između različitih i diskretnih softverskih komponenti. *API* definiše način na koji softverske komponente interaguju jedna sa drugom. Pomoću *API*-ja može da se automatizuje celokupni stek, od fizičkih uređaja, pa sve do aplikacija, i svega između ove dve vrste klaud objekata. Bez *API*-ja automatizacija na klaudu nije izvodljiva, što se svakako odnosi i na automatizaciju sigurnosti.

API-ji podržavaju predefinisane standarde koji omogućavaju interoperabilnost između različitih klaud komponenti. Svaki klaud uređaj ili sistem koji podržava *API* javno objavljuje svoj programski interfejs, koji kasnije softver *developer*-i mogu da koriste u razvoju softverskih rešenja za automatizaciju, koje pomoću objavljenog *API*- ja povezuju sa klaud uređajima i servisima.

Protokoli koji se najčešće koriste za povezivanje sa *API* interefejsom su protokoli za reprezentativni transfer stanja (eng. *Representational State Transfer – REST*) koji omogućavaju komunikaciju između različitih klaud uređaja primenom standardizovanih *HTTP/HTTPS* protokola. U slučaju *REST* sistemske arhitekture, najčešće se koriste dva formata za razmenu podataka: *JSON* (*JavaScript Object Notation*) i *XML* (*Extensible Markup Language*).

U stručnoj i naučnoj javnosti klaud se često opisuje kao jedan veliki programski interfejs. Ova definicija je naročito popularna među ekspertima koji imaju decenijsko iskustvo u radu sa klaud sistemima. Ovo je možda jedna od najboljih definicija klaud računarstva koja postoji u literaturi.

Prema definiciji koja klaud računarstvo dovodi u vezu sa programskim interfejsom, klaud je programski jezik između različitih platformi i aplikacija. Automatizacija je „lepak" koji omogućava sve klaud operacije.

Implementiranjem predefinisanih i dogovorenih softverskih interfejsa koji omogućavaju komunikaciju između heterogenih uređaja i aplikacija, „magija" klaud računarstva se dešava.

Glavni programski jezik za razvoj automatizovanih klaud rešenja je *Python*. Na Internetu može da se nađe veliki broj prekonfigurisanih skripti za automatizaciju koje su pisane u ovom programskom jeziku. Popularni *Web* portal za razmenu ovakvih skripti je *Github (URL*: https://github.com/).

16.5.2. Pristup alatima za automatizaciju klaud provajdera

Mnogi klaud provajderi razvijaju sopstvene interne alate za automatizaciju i ove alate uključuju u standardne ponude klaud usluga. *API*-ji za pristup ovim alatima su u većini slučajeva besplatni, mada postoje slučajevi kada softver developer mora da plati određenu nadoknadu za korišćenje *API*-ja alata za automatizaciju kog je razvio klaud provajder.

> Klaud provajderi objavljuju dokumente gde detaljno opisuju *API* svog alata uz detaljno objašnjenje načina povezivanja *API*-ja sa alatom za automatizaciju kome je *API* namenjen.

Za razliku od aplikacija, koje alatima za automatizaicju na klaudu pristupaju pomoću *API*-ja, korisnici ovim alatima pristupaju preko korisničkih interfejsa. Neka klaud rešenja alata za automatizaciju podržavaju *Web* interfejs, dok druga zahtevaju instaliranje klijentske aplikacije na lokalnom računaru.

U svakom slučaju, tri najviše korišćena interfejsa za pristup korisnika alatu za automatizaciju na klaudu su grafički i komandni interfejsi koji se isporučuju preko lokalne instalacije klijentskih aplikacija, *Web* bazirani interfejsi i portali komandne table.

Grafički i komandni interfejs

Kada se instalira lokalna klijentska aplikacija za pristup alatu za automatizaciju klaud provajdera, većina korisnika najčešće koristi grafički interfejs (eng. *Graphical User Interface – GUI*).

Mnogi alati za automatizaciju osim grafičkog podržavaju i komandni interfejs (eng. *Command Line Interface – CLI*). Mnogi ovakve interfejse smatraju „starom školom". Međutim, komandni interfejsi su fleksibilniji i nude više opcija za konfigurisanje udaljenog alata za automatizaciju od lokalnih i *Web* baziranih grafičkih interfejsa.

Web bazirani interfejsi

Većini alata za automatizaciju na klaudu može da se pristupi preko *Web* baziranog grafičkog korisničkog interfejsa. Ovi interfejsi su popularni zbog jednostavnog korišćenja i fleksibilnosti. Njima može da se pristupi sa bilo kog uređaja, zato što je potreban samo kompatibilan *Web* brauzer.

Pomoću *Web* baziranog interfejsa klijent može da se loguje na sistem za upravljanje klaud okruženjem i da pristupi objedinjenim podacima o statusu svih klaud uređaja.

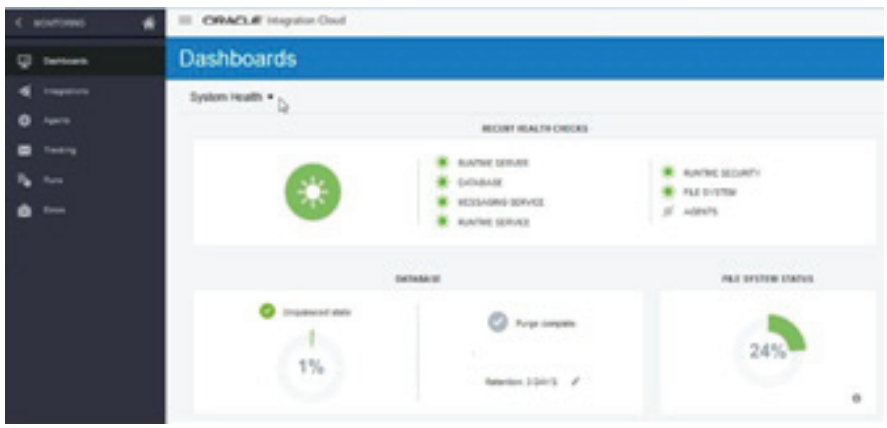
Portali i komandne table

Portali i komandne table (eng. *dashboards*) su najčešći način za pristupanje sistemima za automatizaciju na klaudu.

Portalima može da pristupi i krajnji korisnik (klijent), koji u realnom vremenu može da vidi „zdravstveno stanje" svojih klaud operacija.

Na Slici 16-6 prikazana je komandna tabla *Oracle Integration Cloud* okruženja.

Slika 16-6: *Screenshot - Oracle Integration Cloud dashboard*



16.6. KONTROLA PRISTUPA NA KLAUDU

U ovom poglavlju ukratko su prikazani neki od mehanizama i sistema kontrole pristupa (eng. *access control*) koji se primenjuju u klaud okruženju.

16.6.1. Pristup klaud objektima

Pod pojmom klaud objekat podrazumeva se gotovo svaki računarski resurs koji se nalazi u klaud okruženju. To može da bude običan fajl koji se čuva u klaud sistemu za skladištenje, virtuelna mašina, ruter, *frewall*, ili bilo koji drugi sistem koji se nalazi na klaud infrastruktri.

> U opštem smislu, klaud objekat je entitet kome se pristupa i sa kojim može da se manipuliše na klaudu.

Jedan od najvažnijih zahteva za uspostavljanje i održavanje klaud sigurnosti jeste permanentno i kontinuirano kontrolisanje ko može da pristupi i kojim klaud objektima može da pristupi i koje operacije sa tim objektima može da vrši.

U nastavku su definisani najvažniji pojmovi koji su važni za razumevanje kontrole pristupa klaud objektima.

Proces autorizacije

Kada se korisnik loguje na klaud bazirani sistem za upravljanje računarskim resursima, prvo je potrebno da se utvrdi i potvrdi identitet korisnika procesom autentifikacije. Kada se završi proces autentifikacije, sledeći korak je izvršavanje procesa autorizacije, kada se definiše lista servisa i objekata kojima korisnik može da pristupi i lista akcija koje korisnik nad tim objektima može da izvršava.

Procesi autentifikacije i autorizacije prevazilaze individualne korisnike i primenjuju se i na druge klaud objekte. Tako na primer, autorizacijom se definiše kojim skladišnim uređajima klaud server imaju pristup, ili u koju klaud bazu podataka određena aplikacija može da upisuje podatke.

Korisnički nalozi

Za svakog korisnika koji treba da ima pristup klaud objektima i resursima potrebno je da se kreira korisnički nalog. Korisnički nalog se uglavnom vezuje za identitet osobe koja je korisnik klaud sistema, ali može da ima i šire značenje i da se odnosi i na druge objekte, kao što su na primer serveri ili aplikacije.

Korisničkim nalozima dodeljuju se korisnička prava (eng. user rights) i dozvole (eng. *permissions*) koje definišu nivoe pristupa klaud objektima. U praksi se, radi lakše administracije, korisnički nalozi objedinjuju u grupe (eng. *groups*) i onda se grupama dodeljuju prava i pristupne dozvole.

Korisničke grupe

Korisničke grupe su objekti koji objedinjuju više korisničkih naloga. Upravljanje dozvolama i pravima na nivou korisničkih grupa u značajnoj meri olakšava i pojednostavljuje upravljanje celim klaud sistemom.

Tako na primer, u praksi se najčešće kreiraju korisničke grupe za korisnike kao što su administrator klaud servera, baza podataka, mreža i skladišta. Umesto da se svakom korisniku koji pripada nekoj od navedenih grupa pojedinačno dodeljuju prava i obaveze, prava i obaveze se dodeljuju samo jednom, na nivou cele grupe. Korisnik, kome je potrebna autorizacija da pristupi određenom klaud objektu, jednostavno se priključuje postojećoj grupi koja ima ta prava.

Autorizacija na nivou računarskih sistema

U klaud okruženju često se koristi autorizacija na nivou računarskih sistema, što znači da se pravila za pristup definišu i postavljaju na nivou virtuelnih mašina, ili aplikacija koje se izvršavaju na virtuelnim mašinama.

Za svaku virtuelnu mašinu, korisnika virtuelne mašine, ili aplikaciju koja se izvršava na virtuelnoj mašini mogu da se definišu posebna prava pristupa (koji korisnici mogu da im pristupaju i pod kojim uslovima).

Kontrola pristupa na nivou računarske mreže

Kontrolom pristupa na nivou računarske mreže (eng. *network-based access control*) određuje se ko može da se poveže na mrežu i pod kojim uslovima, a ne ko može da se poveže na server, ili na aplikaciju.

Kao tipični primeri kontrole pristupa na nivou računarske mreže navode se sigurnosne polise kojima se dozvoljava, ili se zabranjuje povezivanje na mrežu sa određenih opsega *IP* adresa ili portova. Kao još jedan dobro poznati primer navodi se korišćenje listi za kontrolu pristupa (eng. *access control list – ACL*).

U većini klaud okruženja koristi se princip slojevite sigurnosti, gde se kontrola pristupa implementira na različitim nivoima, kao što su nivo računarske mreže, nivo servera, skladišta podataka, sloja aplikacija, itd.

Kontrola pristupa na nivou skladišta podataka

Osim politika za kontrolu pristupa na nivou mreža, servera i aplikacija, u klaud okruženju koriste se i politike koje kontrolišu pristup na nivou klaud skladišta podataka.

Kontrola pristupa na nivou skladišta podataka može da se implementira na različite načine. Neki od načina su definisanje pristupa na nivou niza uređaja za skladištenje, ili na nivou logičkog volumena, kada se definiše koji korisnik, ili sistem ima prava da montira taj *volumen*.

Takođe, kontrola pristupa na nivou skladišta podataka na klaudu može da se implementira i na nivoima *SAN*-a, *VSAN*-a (*Virtual Storage Area Network*), ili na nivou skladišnog kontrolera primenom *LUN* maskiranja.

Svi navedeni načini za kontrolu pristupa na nivou skladišta su apstrahovani i skriveni od klaud klijenta i izvršavaju ih sistemi za automatizaciju.

16.6.2. Modeli kontrole pristupa u klaud okruženju

Slično kao u tradicionalnom okruženju, u klaud okruženju raspoloživo je više tipova (modela) kontrole pristupa, a najčešće se koriste sledeći: kontrola pristupa na osnovu uloge, obavezna kontrola pristupa, diskreciona kontrola pristupa, nediskreciona kontrola pristupa i kontrole pristupa koje se baziraju na autentifikaciji sa više faktora sigurnosti i *SSO* tehnologiji.

U nastavku je u kratkim crtama dat prikaz svakog od navedenog tipa kontrole pristupa.

Kontrola pristupa na osnovu uloge

Kontrola pristupa na osnovu uloge (eng. *Role-based Access Control – RBAC*) je vrsta kontrole pristupa gde se pristupna prava dodeljuju, ili ograničavaju korisnicima na osnovu uloge koju ti korisnici imaju u organizaciji.

RBAC koristi predefinisane nivoe dozvola za obavljanje rutinskih aktivnosti na klaudu koje se nazivaju uloge, i dozvoljava se, odnosno, ograničava se pristup klaud resursima na osnovu toga kojoj ulozi pripada određeni korisnik.

Potrebno je da se napomene da su uloge opšti termin i da uloga može da podrazumeva bilo šta i može da se definiše na bilo kom nivou, kao što je nivo aplikacija, operativnih sistema, ili mrežne opreme. Drugim rečima, ne postoji univerzalno pravilo za definisanje uloga, već sve zavisi od slučaja do slučaja.

Obavezna kontrola pristupa

Obavezna kontrola pristupa (eng. Mandatory Access Control – MAC) je pristup koji se uglavnom koristi u klaud okruženjima gde je potrebno da se implementira visok nivo sigurnosti i gde samo mali broj korisnika treba da pristupi osetljivim klaud resursima.

MAC se koristi za autentifikaciju i logovanje korisnika na sistem. Na osnovnu identiteta korisnika i sigurnosnog nivoa (eng. *security level*) kome korisnik pripada, određuju se prava pristupa upoređivanjem podataka o korisniku sa sigurnosnim svojstvima sistema kojem se pristupa.

Na primer, ako korisnik ima prava za upravljanje i konfigurisanje hipervizora u klaud računskom centru, ova prava će biti zapisana u korisničkom profilu. Kada se ovakav korisnik loguje na sistem, informacije o pravima pristupa koje su upisane u korisničkom profilu biće upoređene sa sigurnosnim svojstvima hipervizora i korisnik će dobiti pristup.

S obzirom da *MAC* sistemi spadaju u grupu čvrsto kontrolisanih sistema, pristup resursima se određuje na osnovu striktnih nivoa pristupa. Ovakvi sistemi se na klaudu najčešće koriste kada se na klaud sistemu čuvaju strogo poverljivi i osetljivi podaci, kao što je to slučaj sa finansijskim podacima, ili podacima koji se kategorišu kao državna tajna.

U tipičnoj implementaciji *MAC* sistema dozvoljava se prilično visok nivo pristupa podacima manjeg senzibiliteta, dok je pristup poverljivim podacima čvrstvo kontrolisan. *MAC* sistemi imaju centralnu kontrolnu jedinicu koja dozvoljava, ili ograničava pristup klaud resursima na osnovu definisane sigurnosne politike.

Diskreciona kontrola pristupa

Diskreciona kontrola pristupa (eng. *Discretionary Access Control – DAC*) razlikuje se od *MAC* modela zato što omogućava korisnicima da sami dodeljuju i ograničavaju pristupna prava nad klaud objektima, za razliku od *MAC* modela, gde dodeljivanje prava kontroliše centralni entitet.

Tako na primer, kada se koristi *DAC* model kontrole pristupa, korisnik koji ima upravljačka prava nad određenim klaud sistemom može to pravo da dodeli i nekom drugom korisniku. Korisnik koji imaju upravljačka prava nad određenim klaud objektom najčešće je i vlasnik tog klaud objekta i ima vlasnička prava (eng. *ownership rights*). U grupu prava koja se dodeljuju, osim navedenih spadaju i standardna prava kao što su *read-only, read-write* i *execute*.

Nediskreciona kontrola pristupa

Nediskreciona kontrola pristupa (eng. *Nondiscretionary Access Control – NDAC*) je model kontrole pristupa koji se takođe često primenjuje na klaudu.

NDAC tip kontrole pristupa definiše skup pravila koja dozvoljavaju, ili brane pristup klaud objektima, na osnovu predefinisanih pravila, privilegija i uloga.

Kao primeri *NDAC* kontrole pristupa mogu da se navedu situacije kada se korisniku dozvoli pristup klaud serveru na određeni broj sati tokom radnog dana, ili kada se samo određenim danima u nedelji grupi korisnika omogući pristup osetljivim podacima na klaud skladištu podataka.

Autentifikacija pomoću više faktora sigurnosti

Autentifikacija pomoću više faktora sigurnosti (eng. *Multi-Factor Authentication - MFA*) je model kontrole pristupa prema kome se od korisnika zahteva više različitih informacija kako bi mogli da pristupe određenom klaud objektu.

U literaturi se navode sledeći faktori sigurnosti koji se najčešće koriste u modelu autentifikacije pomoću više faktora na klaudu: nešto što korisnik zna (korisničko ime i/ili lozika), nešto što korisnik ima (smart kartica, *ID* kartica, pristupni token), ono korisnik jeste (biometrijske informacije kao su otisak prstiju, izgled lica, oblik uha i slično) i patern koji korisnik zna i može da ponovi.

Sistemi za autentifikaciju pomoću više faktora sigurnosti mnogo su sigurniji od sistema sa jednim faktorom sigurnosti, kao što je na primer kombinacija korisničkog imena i lozinke.

Kao najjednostavniji primer autentifikacije pomoću više faktora sigurnosti navodi se primer *ATM* (*Automated Teller Machine*) uređaja (bankomata) za podizanje novca. Da bi pristupio bankomatu i podigao novac, korisnik mora da ima svoju kreditnu/debitnu karticu (nešto što korisnik ima) i mora da zna svoj *PIN* kod (nešto što korisnik zna).

U praksi se najviše koristi autentifikacija pomoću dva faktora sigurnosti (eng. *Two- factor Authentication – 2FA*), koja najčešće kombinuje faktore nešto što korisnik zna i nešto što korisnik ima, ili nešto što korisnik zna i ono što korisnik jeste.

Većina klaud provajdera nudi mogućnost autentifikacije pomoću dva faktora sigurnosti za pristup i upravljanje klaud okruženjem. Na ovaj način kreira se dodatni nivo zaštite koji predstavlja dobru sigurnosnu meru i sve više postaje obavezan element korporativnih sigurnosnih politika. Mnogi klaud provajderi nude male uređaje za generisanje tokena i aplikacije koje se instaliraju na pametnim telefonima za uspostavljanje sistema za autentifikaciju pomoću dva faktora sigurnosti.

Da ne bi obavezivali klaud klijente da poseduju fizički pristupni token, većina klaud provajdera koristi softverske emulatore. Jedan od najpopularnijih softverskih alata za *2FA*, koji emulira fizički pristupni token, je *Google Authenticator*, koji je sve popularniji u kladu okruženju.

SSO tehnologija

SSO (*Single Sign-On*) tehnologija omogućava da se korisnik loguje samo jednom i da dobije pristupna prava za više od jednog sistema.

Primenom *SSO* tehnlogija proces autentifikacije na različite sisteme se centralizuje i administriranje korisnicima se u značajnoj meri olakšava.

Tako na primer, ako jedan administrator treba da upravlja sa više *Web* servera, svi *Web* serveri mogu da se stave u jednu *SSO* grupu pod nazivom "*Web* administrator". U tom slučaju, kada se administrator loguje na jedan *Web* server, automatski će da dobije pristup i svim drugim *Web* serverima u grupi, bez potrebe za posebnim logovanjem na svaki *Web* server.

Servisi direktorijuma (eng. *directory services*) koji koriste *LDAP* (*Lightweight Directory Access Protocol*) su dobro poznati primeri implementacije *SSO* sistema, koji su našli široku primenu i na klaudu. Kada se korisnik jednom loguje na servis direktorijuma, korisnik dobija pristup svim resursima koji su članovi tog servisa.

Osim navedenog, *SSO* sistem ima i brojne druge prednosti. Tako na primer, u uslovima postojanja *SSO* sistema, potreba da korisnik pamti različite kredencijale (korisničko ime i šifra) za logovanje na različite sisteme je eliminisana. Ovaj sistem takođe pruža prednosti i prilikom prekidanja sesije. Kada korisnik prekine sesiju (izloguje se sa sistema), korisnik će automatski biti izlogovan i sa svih ostalih sistema organizacije.

16.7. KLAUD MODELI USLUGE I SIGURNOST

Kao što je već navedeno u Glavi 5, postoje tri glavne vrste klaud računarstva prema modelu usluge: *IaaS, PaaS* i *SaaS*.

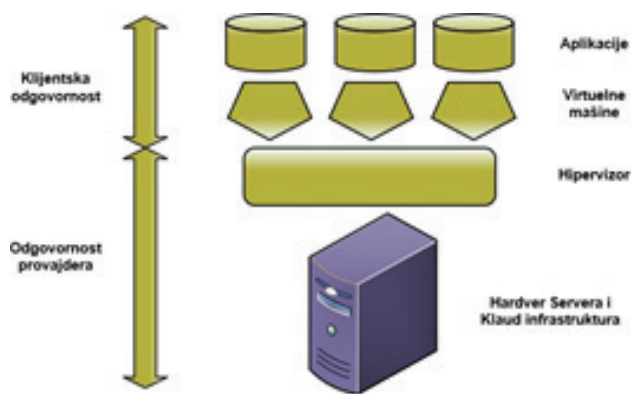
> Na osnovu tipova klaud računarstva prema modelu usluge mogu da se izdvoje i sigurnosni modeli, pa se tako razlikuju *IaaS* model sigurnosti, *PaaS* model sigurnosti i *SaaS* model sigurnosti.

U svakom modelu sigurnosti definisana je podela odgovornosti iz domena klaud sigurnosti između klaud klijenta i klaud provajdera.

U modelu klaud računarstva *IaaS* klaud provajder je odgovoran za celokupnu klaud infrastrukturu, sve do nivoa hipervizora. Slično, u *IaaS* modelu sigurnosti, korporativni klijent je odgovoran za sigurnost slojeva od virtuelnih mašina pa naviše, dok potpunu odgovornost za sigurnost hardvera i hipervizora preuzima provajder usluga.

Model sigurnosti *IaaS* prikazan je na Slici 16-7.

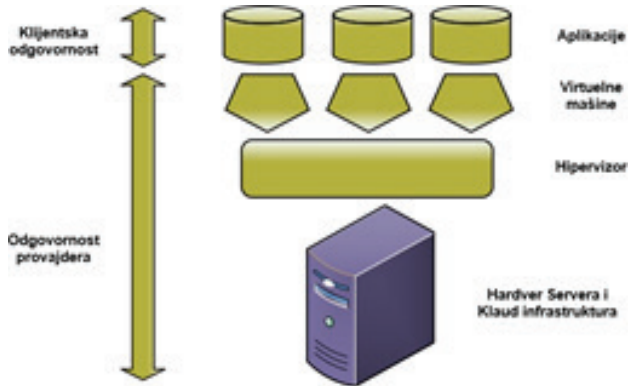
Slika 16-7: I*aaS model sigurnosti*



Prema modelu *PaaS*, klaud provajder obezbeđuje virtuelne mašine i operativni sistem, dok korisnik sam instalira i konfiguriše svoje aplikacije. Slično, u *PaaS* modelu sigurnosti, klaud provajder preuzima odgovornost za računarsku sigurnost svih slojeva do nivoa operativnog sistema, dok je klaud klijent odgovoran za konfigurisanje operativnog sistema i aplikacija.

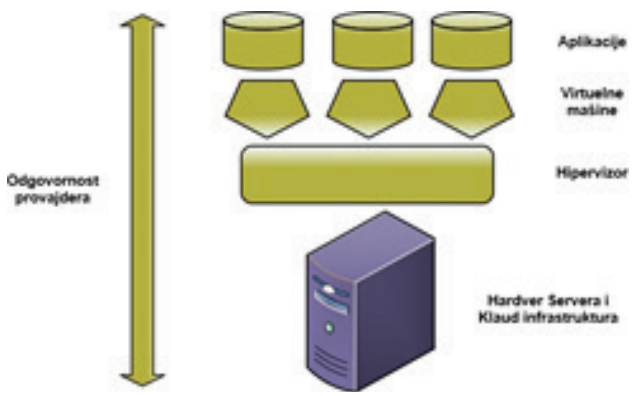
Prikaz *PaaS* modela sigurnosti dat je na Slici 16-8.

Slika 16-8: *PaaS model sigurnosti*



Konačno, u modelu *SaaS*, klaud klijent pristupa aplikativnom softveru koji sam kontroliše, a koji je u vlasništvu klaud provajdera. U ovom slučaju, prema *SaaS* modelu sigurnosti, klaud provajder preuzima potpunu odgovornost za sigurnost svih slojeva, kao što je prikazano na Slici 16-9.

Slika 16-9: *SaaS model sigurnosti*



16.8. KLAUD MODELI ISPORUKE I SIGURNOST

Kao što je navedeno u Glavi 6, prema modelu isporuke, klaud računarstvo može da se podeli na javni, privatni, hibridni i zajednički klaud. Razumevanje ovih modela je važno zato što oni pomažu u razumevanju globalne strukture klaud infrastrukture.

U svakom od navedenih modela isporuke, sigurnosne politike i odgovornosti se razlikuju. Međutim, uz ove razlike, svaka organizacija primenjuje svoju personalizovanu sigurnosnu politiku i zbog toga razlike još više dolaze do izražaja.

U većini klaud implementacija u modelu privatnog klauda jedna organizacija (korporativni klijent) upravlja svojom računarskom klaud infrastrukturom i preuzima sve obaveze po pitanju sigurnosti i odgovornosti za eksploataciju svih klaud objekata.

U modelu javnog klauda obaveze po pitanju sigurnosti klaud računarskih resursa dele se između klaud provajdera i klaud klijenta. Precizna podela odgovornosti po pitanju sigurnosti klaud objekata između klaud provajdera i klijenta definiše se Sporazumom o nivou korišćenja usluga i u jednom obimu zavisi i od korišćenog modela usluge (*IaaS, PaaS, SaaS*).

Obaveze i odgovornosti iz domena sigurnosti u modelu hibridnog klauda zavise od sigurnosnih politika klaud klijenta i provajdera koji učestvuju u implementaciji ovog modela, dok se većina obaveza po pitanju sigurnosti u modelu zajedničkog klauda uglavnom deli između svih entiteta koji upravljaju ovim okruženjem.

16.9. IMPLEMENTACIJA KLAUD SIGURNOSTI

U ovom poglavlju udžbenika prikazana je implementacija klaud sigurnosti primenom tehnika klasifikacije podataka, segmentacije klaud okruženja, skladišta podataka i resursa koji vrše izračunavanja, šifrovanjem podataka i autentifikacijom pomoću više faktora sigurnosti.

16.9.1. Klasifikacija podataka

Klasifikacija podataka je tehnika grupisanja podataka u različite slojeve, kako bi se realizovali sledeći ciljevi: poboljšavanje raspoloživosti podataka, usaglašavanje politika za prenos i skladištenje podataka sa regulatornim zahtevima i standardima, izbegavanje rizika korumpiranja i krađe podataka i unapređenje opšte bezbednosti podataka.

Na primer, klasifikacija se koristi kako bi se što pre identifikovali osetljivi podaci i nad njima izvršile aktivnosti enkripcije i skladištenja na visoko bezbednoj lokaciji.

Metapodaci (eng. *metadata*) su podaci o podacima. Ovaj tip podataka se najčešće vezuje za fajlove i predstavlja se u formi taga (oznake) koja objašnjava, kategorizuje i klasifikuje fajlove. Tagovanjem se značajno ubrzava proces pronalaženja željenih podataka i unapređuje se sigurnost podataka.

Tagovi se najčešće definišu u formi *tip:vrednost*. Tako na primer tip taga može da bude Univerzitet, a vrednost Singidunum (*Univerzitet:Singidunum*). Javni klaud provajderi nude servise klasifikacije podataka na osnovu tagova.

Podaci se uglavnom klasifikuju na osnovu njihove osetljivosti, kao što su nivoi poverljivosti i značajnosti podataka. Tako na primer, neke organizacije organizuju svoje podatke u tri grupe: zabranjeni podaci (eng. *restricted data*), privatni podaci (eng. *private data*) i javni podaci (eng. *public data*). Nakon ovakve klasifikacije, na svaku grupu podataka se primenjuju odgovarajuće sigurnosne politke. Politika za javne podatke će u najvećem broju slučajeva da bude najfleksibilnija, dok će sigurnosna politika za zabranjene podatke da bude najrestriktivnija i usklađena sa sigurnosnim standardima poput *HIPPA, SOX* i *PCI-DSS*.

U prethodnom primeru naveden je samo jedan način za klasifikaciju podataka.

Svaka organizacija definiše svoje kriterijume za klasifikaciju podataka i načine za primenu sigurnosnih politika na osnovu izvršene klasfikacije.

Zaposleni u organizaciji imaju posebne treninge na kojima uče kako da identifikuju i klasifikuju podatke i kako da primene standarde za svaku grupu podataka.

Klasifikacija podataka omogućava organizacijama da lakše upravljaju velikim količinama podataka koje čuvaju u svojim klaud skladištima. Primenom klasifikacije podaci postaju vidljiviji, a načini za čuvanje, pristup i zaštitu podataka precizniji. Važno je da s vremena na vreme organizacija izvrši revizuju implementacije sistema za klasifikaciju podataka, jer se potrebe vremenom menjaju.

Primer jedne klasifikacije podataka na klaud skladištima prikazan je na Slici 16-10.

Slika 16-10: *Klasifikacija podataka na klaud skladištu*



16.9.2. Segmentacija klauda

Segmentacija klauda (eng. *cloud segmentation*) je proces podele klaud infrastrukture u sekcije (domene), kako bi se omogućila primena granularnih (fino podešenih) sigurnosnih politika, personalizovanih za svaki domen ponaosob.

Segmentacija klauda se uglavnom odnosi na segmentaciju klaud računarske mreže i omogućava implementiranje koncepta slojevite sigurnosti.

Klaud mreže mogu da se segmentiraju prema funkcijama, gde se onda za svaki segment primenjuju određene sigurnosne politike. Tako na primer, jedan segment može da bude javni *Web* server kome se pristupa preko Interneta, drugi segment može da bude aplikativni sloj, gde su hostovani aplikativni serveri na kojima se izvršavaju aplikacije, dok treći segment može da bude sloj baze podataka. Ostali segmenti mogu da obuhvataju skladišta podataka, backend sisteme za upravljanje klaudom, razne klaud servise i *DMZ* oblast.

Podelom klaud mreže u segmente unapređuje se opšta sigurnost i omogućava se primena sigurnosnih kontrola na nivou toka podataka od jednog ka drugom mrežnom segmentu.

Na ovaj način preciznije može da se definiše koji korisnici i uređaju mogu da pristupe klaud mreži, kao i da se preciznije izvrše konfiguraciona podešavanja sigurnosnih uređaja. Neki od mehanizama obezbeđivanje sigurnosti na nivou segmenata klaud računarske mreže su *ACL* liste, sigurnosne grupe, uloge, *firewall* uređaji i bezbednosne polise.

Segmentiranje klaud mreže takođe može da se vrši pomoću virtuelnih privatnih klaud mreža (eng. *Virtual Private Cloud - VPC*), gde se na svaki *VPC* primenjuje precizno definisana sigurnosna politika koja implementira pravila sa dozvolama, odnosno zabranama saobraćaja između različitih *VPC* mreža. Osim navedenog, *VPC* mrežni segmenti mogu da se kreiraju i na osnovu funkcija. Tako na primer, jedan *VPC* može da se odnosi na produkcione servere, drugi na razvojne, itd.

Mrežni segmenti mogu da obuhvataju različite klaud računske centre, kada se povećava elastičnost i otpornost ceolokupne klaud mreže. Ovakva arhitektura se u literaturi naziva arhitektura višestruko dostupnih zona (eng. *multi-availability zone architecture*).

16.9.3. Segmentacija skladišta

Segmentacijom skladišta klaud provajder vrši odvajanje fizičkih i/ili virtuelnih lokacija na kojima se podacima skladište, kako bi izašao u susret različitim zahtevima klijenata.

Primenom alata za dodavanje skladišta podataka, klaud provajder može da implementira više različitih skupova skladišta, kako bi najrazličitiji zahtevi klijenata bili zadovoljeni. Tako na primer, jedan klijent može da zahteva bazu podataka sa velikim intenzitetom *I/O* operacija i za ove potrebe klaud provajder će da obezbedi *SSD* uređaj. S druge strane, baza podataka klijenta koja ne zahteva intenzivne *I/O* operacije može da se čuva i na standardnom magnetnom disku. Slično, ako su podaci arhivirani i spremni za dugoročno skladištenje, dovoljno je da se koristi medijum sa slabijim performansama.

Segmentacijom skladišta takođe može da se identifikujeoptimalni tip skladišta koji bi trebao da se koristi. Prema jednoj podeli, tri glavne vrste skladišta su fajl bazirano, blok bazirano i objektno skladište. Svaki navedeni tip skladišta zadovoljava različite slučajeve korišćenja i potrebe klijenata.

Na virtuelne mašine se obično vezuju dva tipa skladišta, i to izdržljiva (eng. *durable*) i neizdržljiva (eng. *nondurable*) skladišta. Kada se virtuelna mašina izbriše, neizdržljiva skladišta se brišu zajedno sa njom. Ovakvi tipovi skladišta se ponekada u literaturi nazivaju i privremeni *volumeni* (eng. *ephemeral volumes*) zato što su potpuno zavise od virtuelne mašine. S druge strane, izdržljiva skladišta ostaju u upotrebi i nakon brisanja virtuelne mašine za koju su vezana.

16.9.4. Segmentacija resursa koji vrše izračunavanja

Segmentiranjem klaud resursa koji vrše izračunavanja, ovi resursi se grupišu u izolovanu mrežu koja dozvoljava lokalne konekcije, čime se unapređuje sigurnost.

Jedna od najboljih sigurnosnih praksi u klaud okruženju jeste da se resursi za izračunavanje poput virtuelnih mašina grupišu u izolovane podmreže i da se zatim utvrde precizne sigurnosne politike koje se primenjuju na sav saobraćaj koji dolazi ka i odlazi sa ovih podmreža.

Segmentacija resursa koji vrše izračunavanja se najčešće realizuje u formi troslojne arhitekture. Virtuelne mašine na kojim se izvršavaju javni servisi, poput *HTTP* ili *HTTPS* servisa stavljaju se u posebnu podmrežu koja je vidljiva sa Interneta i primenjuju se posebna pravila za filtriranje paketa i saobraćaja koji dolazi sa Interneta.

U drugi sloj stavljaju se virtuelne mašine na kojima se izvršavaju aplikativni severi, dok se u trećem sloju nalaze serveri baza podataka koji su povezani samo sa aplikativnim serverima.

Još jedna primena segmentiranja računarskih resursa koji vrše izračunavanja na klaudu odnosi se na *hostovanje* tzv. *big data* sistema. Konfiguracija *big data* sistema zahteva veliki broj virtuelnih mašina koje se grupišu u svoj izolovani segment, na koji se zatim primenjuju posebna sigurnosna pravila i politike.

16.9.5. Implementacija sistema za šifrovanje podataka

Mnoge klaud organizacije usvajaju politike i regulative koje zahtevaju šifrovanje podataka u transferu i u mirovanju. Mnoge tehnologije podržavaju ove zahteve, pa tako na primer neke tehnologije baza podataka omogućavaju čak i da se šifrovani podaci obrađuju u realnom vremenu.

> Regulatorne agencije, vlade i korporativne politike mogu da zahtevaju šifrovanje svih uskladištenih podataka.

Tako na primer, regulative mnogih zemalja zahtevaju da medicinski podaci o pacijentima u zdravstvenim organizacijama i finansijski podaci o klijentima banke budu šifrovani. Šifrovanje podataka zahteva dodatno angažovanje računarskih resursa, što u znatnoj meri usporava rad sa podacima. Zbog toga veliki klaud provajderi implementiraju posebne hardverske uređaje koji se isključivo bave procesima šifrovanja i dešifrovanja podataka.

Sofisticirani i napredni kontroleri skladišta omogućavaju brzo i efikasno šifrovanje i dešifrovanje velike grupe podataka. Ovakvi kontroleri implementiraju više koprocesora korišćenjem specijalnog silikonskog čipa, koji se naziva *ASIC* (*Application Specific Integrated Circuit*). Na ovaj način ovakvi uređaji rasterećuju procesore virtuelnih mašina i unapređuju opšte performanse sistema.

Proces šifrovanja i dešifrovanja podataka je potpuno transparentan za klaud klijenta. Klaud provajder enkapsulira i apstrahuje proces šifrovanja. Potrebno je na primer da korisnik samo označi odgovarajući *checkbox* preko svog interfejsa, koji se odnosi na zahteve za šifrovanjem podataka i svi njegovi podaci na klaudu će da budu šifrovani.

16.9.6. Sistemi za autentifikaciju pomoću više faktora sigurnosti

Klaud klijent podešava opcije za autentifikaciju pomoću više faktora sigurnosti preko konzole sistema za upravljanje identitetom (eng. *identity management system*) klaud provajdera.

Kada se kreira novi korisnik, klaud klijent može da definiše željeni profil za autentifikaciju. Ovaj proces je toliko jednostavan da se u većini slučajeva od klaud korisnika zahteva samo da označi *checkbox* za autentifikaciju pomoću više faktora sigurnosti, nakon čega se korisniku otvara padajuća lista sa koje može da izabere faktore sigurnosti koje želi da uključi.

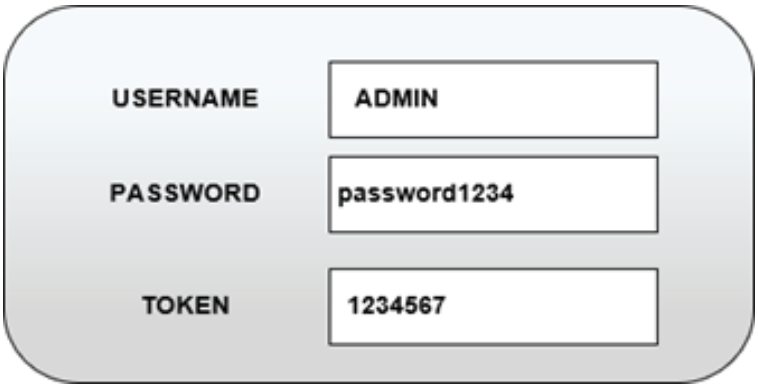
Hardverski tokeni su popularni uređaji koji se koriste za autentifikaciju pomoću više faktora sigurnosti. Ovi mali uređaji imaju ekran koji prikazuje *ID* broj, odnosno token, koji se menja u predefinisanim vremenskim intervalima (Slika 16-11). *ID* broj je uglavnom validan samo nekoliko minuta, i u tom vremenskom intervalu klaud korisnik treba da upiše *ID* broj koji je pročitao sa ekrana svog hardverskog tokena u *login* formu za pristup sistemu (Slika 16-12) zajedno sa korisničkim imenom i lozinkom.

Slika 16-11: *Hardverski token*



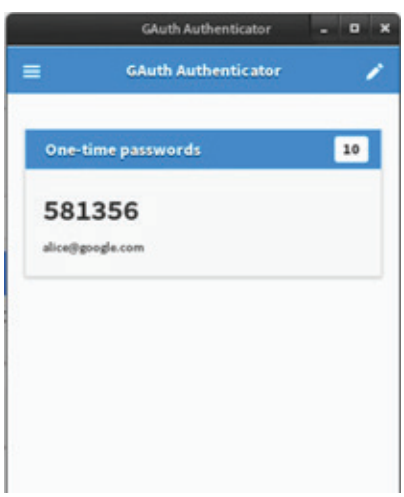
Izvor: (slika preuzeta sa: https://kb.iu.edu)

Slika 16-12: *Login forma za autentifikaciju pomoću hardverskih tokena*



Neki klaud provajderi takođe nude i softverske verzije tokena za pristup klaud sistemu u formi generatora jednokratne šifre (eng. *One Time Password – OTP*), koji može da se instalira na laptop i tablet računarima, ili pametnim telefonima. Softverski token ima istu svrhu kao i hardverski token (Slika 16-13).

Slika 16-13: *Google Authenticator softverski token*



Izvor: (slika preuzeta sa: https://chrome.google.com)



PITANJA ZA RAZMATRANJE

1. Objasniti tehnologiju za zaštitu podataka *IPsec*.
2. Koja dva režima rada koristi *IPsec*? Objasniti.
3. Šta je *SSL*?
4. Šta je *TLS*?
5. Šta je kriptografija? Objasniti pojam kriptoanalize.
6. Kako se dele šifarski algoritmi?
7. Objasniti simetrične šifarske algoritme. Dati primer.
8. Objasniti asimetrične šifarske algoritme. Dati primer.
9. Objasniti *3DES* šifarski algoritam.
10. Objasniti *AES* šifarski algoritam.
11. Objasniti pojam infrastrukture javnog ključa. Dati primer.
12. Objasniti kroz primer arhitekturu *PKI* sistema.
13. Šta je *L2TP* protokol? Objasniti.
14. Šta je *PPTP* protokol? Objasniti.
15. Šta je *SSTP* protokol? Objasniti.
16. Šta se obezbeđuje sa automatizacijom klaud sigurnosti?
17. Šta su alati za automatizaciju na klaudu? Dati primer.
18. Šta je aplikativni programski interfejs?
19. Šta predstavlja autorizacija na nivou računarskih sistema? Objasniti.
20. Šta određuje kontrola pristupa na nivoima računarske mreže i skladišta podataka?
21. Navesti i ukratko objasniti modele kontrole pristupa u klaud okruženju.
22. Objasniti razliku između *IaaS* modela sigurnosti, *PaaS* modela sigurnosti i *SaaS* modela sigurnosti.
23. Objasniti procese segmentiranja klauda i skladišta podataka.
24. Objasniti kroz primer autentifikaciju korisnika pomoću hardverskih tokena.



PRAKTIČNE IMPLEMENTACIJE

DEO V
PRAKTIČNE IMPLEMENTACIJE



U petom delu udžbenika prikazana su tri praktična primera upotrebe tehnologije virtuelizacije i klaud računarstva. Svi primeri su detaljno opisani po principu „korak po korak".

Prva dva primera se povezani, gde se drugi primer nadovezuje na prvi. U prvom primeru prikazano je implementiranje virtuelnih mašina u *Oracle VirtualBox* okruženju hosted hipervizora. U primeru su korišćene tri *GNU/Linux* virtuelne mašine, i to jedna koja ima ulogu servera i dve koje imaju ulogu klijenta. S obzirom da paket Oracle VirtualBox omogućava napredne funkcionalnosti virtuelizacije mreže pomoću nekoliko podržanih modova, one su detaljno opisane.

U drugom primeru opisana je implementacija *Git* distribuiranog sistema za kontrolu verzije u virtuelnom okruženju pod *GNU/Linux* operativnim sistemom, koji simulira izvršavanje *Git* servisa u okruženju klaud provajdera. U ovom slučaju instalirane su i konfigurisane su jedna *Git* serverska, i dve *Git* klijentske komponente. Sve tri *Git* komponente su *hostovane* na virtuelnim mašinama koje su konfigurisane u prvom primeru. Glavna razlika između simuliranog *Git* klaud okruženja i realnog klaud okruženja je ta što klaud provajderi najčešće koriste bare metal hipervizore, koji rade na nivou neposredno iznad fizičkog hardvera računara, dok je u ovom slučaju *Git* serverska komponenta *hostovana* na hipervizoru tipa 2.

S obzirom da je između ostalog ovaj udžbenik namenjen i studentima studijskog modula *Softversko inženjerstvo Univerziteta Singidunum* i da se *Git* često koristi za potrebe razvoja softvera, u okviru drugog primera data je osnovna *Git* terminologija i ilustrovano je korišćenje osnovnih *Git* operacija u *Linux bash* okruženju kroz praktične simulacije i scenarije.

Konačno, u trećem primeru je prikazan model raspoređivanja poslova i balansiranja opterećenja u klaud računarstvu. Data je osnovna terminologija i algoritmi koji se koriste za raspoređivanje poslova na klaudu. Opisan je prvi model koji je korišćen u praktičnim simulacijama i koji spada u grupu optimizacionih problema sa jednom funkcijom cilja, gde je uzeta u obzir minimizacija ukupnog vremena izvršavanja svih poslova na raspoloživim klaud računarskim resursima.

Prikazan je i drugi model, koji pripada kategoriji višekriterijumske optimizacije sa ograničenjima, gde su uzete u obzir dve funkcije cilja: ukupno vremene izvršavanja svih poslova i minimizacija troškova u okviru raspoloživog budžeta.

Posebna pažnja je posvećena metaheuristikama i klasifikaciji na metaheuristike koje su inspirisane prirodom, s obzirom da se metaheuristike inteligencije rojeva implementiraju u simulacijama za problem raspoređivanja poslova i balansiranje opterećenja u klaudu. Prikazani su osnovni principi i data je definicija algoritama rojeva, kao i dve najvažnije komponente procesa pretrage: eksploatacija (intenzifikacija) i eksploracija (diversifikacija).

U eksperimentalnim simulacijama korišćena je verzija *3.0.3 CloudSim* platforme. Prikazan je proces instalacije i primeri simulacija u *CloudSim* platformi, a nakon toga je obuhvaćen problem raspoređivanja poslova i balansiranja opterećenja u klaud okruženju primenom metaheuristika inteligencije rojeva.

Kako bi se bolje i preciznije izvršile evaluacije performansi predloženog pristupa, izvršene su dve grupe simulacija, jedna sa pravim skupom podataka i druga sa veštačkim skupom podataka.

U prvoj grupi simulacija, korićen je model sa jednom funkcijom cilja, dok je u simulacijama iz druge grupe, rešavan višekriterijumski model raspoređivanja sa ograničenjima.

Obe grupe simulacija su rađene u *CloudSim 3.0.3* okruženju. Takođe, obe simulacije izvršavane su na računarskoj platformi sa Intel CoreTM i7-4770K procesorom @4GHz sa 32GB RAM memorije, Windows 10 Professional x64 operativnim sistemom i Java Development Kit 11 (JDK 11) i IntelliJ integrisanim razvojnim okruženjem.

Među ostalim korišćenim terminima na engleskom jeziku, u ovom delu udžbenika često su upotrebljivani termin *host*, koji označava operativni sistem koji je instaliran na fizičkom računaru, termin *guest* koji referiše operativni sistem koji je instaliran na virtuelnoj mašini koja se izvršava u okruženju hipervizora, kao i *virtual machine* koji se nalazi na *host* računaru.

Za upoznavanje sa konceptima optimizacionih problema, definisan je pojam i data osnovna taksonomija optimizacionih problema. Ovaj koncept služi sa dalja istraživanja vezano za primenu algoritama u rešavanju praktičnih *NP* (eng. *nondeterministic polynomial time problems*) problema.

Napominje se da za samostalnu implementaciju prikazanih servisa u ovom delu udžbenika nije neophodno poznavanje *Linux bash* naredbi, zato što se koriste samo najosnovnije komande, čiji je opis priložen uz listinge naredbi.

17. INSTALIRANJE I KONFIGURISANJE VIRTUELNIH MAŠINA

U ovom praktičnom primeru prikazana je implementacija *GNU/Linux* virtuelnih mašina u okruženju *Oracle VirtualBox hosted* hipervizora.

U primeru su korišćene tri *GNU/Linux* virtuelne mašine, i to jedna koja ima ulogu servera i dve koje imaju ulogu klijenta. Na sve tri mašine instalirana je *Ubuntu GNU/Linux* distribucija. Za serversku stranu korišćen je *Ubuntu Server 18.04 LTS*, dok je na klijentskim mašinama instalirana *Ubuntu Desktop* 18.04 *LTS* verzija. Oba operativna sistema su najsvežije verzije *Ubuntu GNU/Linux* distribucije u vreme pisanja ovog udžbenika.

Mašina koja ima ulogu servera ima naziv *gitsrv*, dok prva i druga klijentska mašina imaju nazive *gitclient* i *gitclient2*, respektivno.

Sve tri virtuelne mašine konfigurisane su za potrebe implementacije *Git* distribuiranog sistema za kontrolu verzije fajlova u virtuelnom okruženju, koja je prikazana u Glavi 18.

17.1. ORACLE VIRTUALBOX

Oracle VirtualBox je softver za potpunu virtuelizaciju namenjen *x86* računarskim arhitekturama, koji može da se implementira i na serverskim i na *desktop* konfiguracijama. *Oracle VirtualBox* pripada grupi hipervizora drugog nivoa, ili grupi *hosted* hipervizora, zato što se izvršava iznad operativnog sistema *host* računara.

Neke od najvažnijih karakteristika *Oracle VirtualBox* hipervizora obuhvataju sledeće:

- portabilnost. *VirtualBox* može da se izvršava na različitim host operativnim sistemima 32-bitne i 64-bitne arhitektura. Podržani operativni sistemi u vreme pisanja ovog udžbenika su *GNU/Linux, macOS, Windows, Solaris*, i *OpenSolaris*. Na kojoj god računarskoj platformi da se izvršava,

VirtualBox postiže približno iste performanse i koristi isti format fajla virtuelnih mašina. Na ovaj način moguće je da se na primer virtuelna mašina kreira na jednom *host* računaru sa *Windows* operativnim sistemom i da se ta ista mašina kasnije migrira na *host* koji radi pod *GNU/Linux* operativnim sistemom. Osim navedenog, izvoz virtuelnih mašina sa jednog i uvoz na drugi *host* operativni sistem je značajno olakšan korišćenjem *OVF* (*Open Virtualization Format*) formata, koji je prihvaćen kao industrijski standard. *Oracle VirtualBox* podržava i uvoz *OVF* fajlova koji su kreirani pomoću drugog softvera za virtuelizaciju,

- nije neophodna podrška za virtuelizaciju na nivou hardvera. U većini slučajeva korišćenja, *VirtualBox* ne zahteva podršku za virtuelizaciju koja je ugrađena na nivou procesora, kao što je set instrukcija *Intel VT-x,* ili *AMD-V*. Zbog toga, za razliku od drugih softvera za virtuelizaciju, *VirtualBox* može da se koristi i na starijim računarskim platformama,
- podržani su brojni dodaci za guest operativni sistem. Tako na primer, skup softverskih dodataka pod nazivom *VirtualBox Guest Additions* je skup paketa koji mogu da se instaliraju u podržanim *guest* operativnim sistemima sa ciljem poboljšavanja performansi i integracije virtuelne mašine sa *host* operativnim sistemu. Na ovaj način virtuelna mašina dobija softversku podrške u radu, kao što je na primer podrška za automatsko podešavanje rezolucije, ubrzanu 3D grafiku i slično. Jedan od najšire korišćenih dodataka je funkcija deljenih foldera (eng. *shared folders*), koja omogućava da se iz guest operativnog sistema pristupi fajlovima i ostalim resursima koji se nalaze na *host* sistemu,
- velika podrška na nivou fizičkog hardvera. *VirtualBox* ima veliku podršku na nivou računarskog hardvera, kao što su:
  - simetrično multiprocesiranje (eng. *symmetrical multiprocessing – SMP*), gde je podržano čak 32 virtuelnih procesora po jednoj virtuelnoj mašini, bez obzira na broj jezgara fizičkog procesora na *host* računaru.
  - podrška za *USB* uređaje. *VirtualBox* implementira virtuelni *USB* kontroler koji omogućava povezivanje najrazličitijih *USB* uređaja bez potrebe za instaliranjem dodatnih upravljačkih programa.
  - kompatibilnost hardvera. *VirtualBox* može da virtuelizuje veliki broj uređaja i komponenti, kao što su *IDE, SCSI* i *SATA* disk kontroleri, nekoliko tipova mrežnih i zvučnih kartica, virtuelni serijski i paralelni portovi, *I/O APIC* (*Input/Output Advanced Programmable Interrupt Controller*) kontroleri, itd.

  - potpuna *ACPI* (*Advanced Configuration and Power Interface*) podrška. Uz ovu podršku značajno je olakšan proces kloniranja slika fizičkih mašina, ili virtuelnih mašina treće strane na *VirutalBox* mašine. U slučaju *host* mobilnih uređaja koji koriste bateriju kao izvor električnog napajanja, *ACPI* funkcija omogućava da se na *guest* operativnom sistemu uključi režim rada za smanjivanje potrošnje električne energije i da se generišu obaveštenja o statusu baterije,
- rezolucije za veće ekrane. Mašine implementirane u okviru *VirtualBox* hipervizora podržavaju rezolucije koje su višestruko veće od rezolucije fizičkog ekrana koji je povezan na host računar,
- ugrađena podrška za *iSCSI*. Zahvaljujući ovoj karakteristici, virtuelna mašina može direktno, bez posredovanja *host* računara, da se poveže na *iSCSI* server za skladištenje podataka, čime se značajno poboljšavaju *I/O* performanse čitanja i upisivanja podataka,
- *PXE* (*Preboot Execution Environment*) butovanje sistema preko računarske mreže. Integrisana virtuelna mrežna kartica u potpunosti podržava udaljeno butovanje sistema pomoću *PXE* tehnologije,
- snimci (eng. *snapshots*) stanja sistema u formi stabla. *VirtualBox* podržava kreiranje višestrukih snimaka stanja sistema, gde korisnik može da resetuje virtuelnu mašinu na bilo koji snimak iz prošlosti i da nastavi da koristi virtuelnu mašinu od te vremenske tačke. Na ovaj način se kreira stablno snimaka, gde snimci mogu da se kreiraju i brišu dinamički, dok mašina radi,
- grupe virtuelnih mašina. *VirtualBox* omogućava da korisnik organizuje i kontroliše, kako individualne, tako i grupe virtuelnih mašina. Jedna virtuelna mašina može da pripada različitim grupama, i grupe mogu da budu ugnježdene. Operacije koje mogu da se primene za bilo koju individualnu virtuelnu mašinu, podržane su i na nivou grupnih operacija, kao što su na primer operacije *Start* (pokretanje virtuelne mašine), *Pause* (pauziranje izvršavanja virtuelne mašine), *Reset* (resetovanje virtuelne mašine) i *Close* (gašenje virtuelne mašine),
- *VirtualBox SDK* (*Software Development Kit*). Zahvaljujući modularnoj arhitekturi, *VirtualBox* je interoperabilan sa velikim brojem drugih sistema i softvera pomoću detaljnog *SDK*-a i
- prikaz na udaljenoj mašini (eng. *remote machine display*). *VirtualBox* podržava funkciju pod nazivom *VRDE* (*VirtualBox Remote Desktop Extension*), koja omogućava pristup udaljenoj virtuelnoj mašini sa bilo koje lokacije na mreži. *VRDE* tehnologija podržava i *Microsoft RDP* (*Remote Desktop Protocol*) uz unapređenje koje se odnosi na potpunu podršku *USB* uređaja.

VirtualBox verzija koja je aktuelna u vreme pisanja ovog udžbenika (verzija 5.2) podržava veliki broj host operativnih sistema, kao što su *Microsoft Windows*, *GNU/Linux*, *Mac OS X* i *Solaris*.

S druge strane, u svakoj novoj verziji *Oracle VirtualBox*-a, podrška za neke od starijih *guest* operativnih sistema se uklanja. Najviše se koriste guest operativni sistemi iz familije *Microsoft Windows, GNU/Linux* i *Mac* operativnih sistema.

U sledećoj listi navedeni su samo neki od predstavnika podržanih *guest* operativnih sistema:

- *Windows* operativni sistemi:
  - *Windows Server* 2008 (64-bit)
  - *Windows Server* 2008 R2 (64-bit)
  - *Windows* 7 (32-bit and 64-bit)
  - *Windows* 8 (32-bit and 64-bit)
  - *Windows* 8.1 (32-bit and 64-bit)
  - *Windows* 10 *RTM build* 10240 (32-bit and 64-bit)
  - *Windows Server* 2012 (64-bit)
  - *Windows Server* 2012 R2 (64-bit)
  - *Windows Server* 2016 (64-bit)
- *Mac OS X* operativni sistemi (64-bit):
  - 10.10 (*Yosemite*)
  - 10.11 (*El Capitan*)
  - 10.12 (*Sierra*)
  - 10.13 (*High Sierra*)
- *GNU/Linux* operativni sistemi (32bit i 64bit):
  - *Ubuntu* 14.04 *LTS*, 16.04 *LTS*, 17.04 i 18.04 *LTS*
  - *Debian GNU/Linux* 7 ("*Wheezy*"), 8 ("*Jessie*") i 9 ("*Stretch*")
  - *Oracle Enterprise Linux* 5, *Oracle Linux* 6 i 7
  - *Redhat Enterprise Linux* 5, 6 i 7
  - *Fedora* 25 and 26
  - *Gentoo Linux*
  - *openSUSE* 13.2.

17.2. VIRTUALBOX VIRTUELIZACIJA MREŽE

S obzirom da je za implementiranje okruženja za potrebe praktičnih primera koji su prikazani u ovom udžbeniku neophodno razumevanje virtuelizacije mreže koju *Oracle VirutalBox* omogućava, u nastavku su navedeni modovi (režimi) rada i tipovi virtuelnih mreža u *Oracle VirtualBox* okruženju.

Oracle VirtualBox podržava napredne tehnike virtuelizacije mreže i korisnik ima na raspolaganju veliki broj načina za podešavanje virtuelnih mreža. Međutim, uz bogatu ponudu i veliki broj raspoloživih opcija često dolazi do komplikacija i poteškoća, i da bi virtuelna mreža bila na pravi način konfigurisana, potrebno je poznavanje tipova mreža, tj. operativnih modova koje *Oracle VirtualBox* podržava.

U paketu *Oracle VM VirtualBox* 5.2, koji je korišćen za potrebe praktičnih primera u ovom udžbeniku, korisnik može da konfiguriše do 8 virtuelnih mrežnih kartica za svaku virtuelnu mašinu. Kao napomena navodi se da pomoću grafičkog interfejsa *VirtualBox* paketa mogu da se konfigurišu samo 4 mrežne kartice. U slučaju zahtevnijih korisnika, kada je potrebno više kartica, mora da se koristi konzolni interfejs.

Radi ilustracije funkcija *Oracle VirtualBox*-a u ovom udžbeniku je korišćen grafički interfejs.

> Za svaku virtuelnu mrežnu karticu implementirane virtuelne mašine korisnik može da konfiguriše sledeće: tip virtuelne mrežne kartice i način na koji je virtuelna mrežna kartica povezana sa fizičkom računarskom mrežom *host* računara (operativni mod).

Oralce *VirtualBox* 5.2 podržava sledeće tipove (modele) mrežnih kartica koje mogu da se dodaju svakoj virtuelnoj mašini:

- *PCnet-PCI II* (Am79C970A),
- *PCnet-Fast III* (Am79C973),
- *Intel PRO/1000 MT Desktop* (82540EM),
- *Intel PRO/1000 T Server* (82543GC),
- *Intel PRO/1000 MT Server* (82545EM) i
- paravirtuelizovani mrežni adapter (*virtio-net*).

Izbor vrste virtuelnog mrežnog adaptera pre svega zavisi od tipa guest operativnog sistema, tj. od toga da li *guest* operativni sistem ima podršku za određeni mrežni adapter na nivou upravljačkih programa (drajvera). Prilikom kreiranja virtuelne mašine korisnik putem *VirtualBox* interfejsa bira vrstu i verziju operativnog sistema, na osnovu čega *VirtualBox* preporučuje tip mrežnog adaptera. Poželjno je da se korisnici pridržavaju ove preporuke, jer je preporučeni virtuelni mrežni adapter garantovano kompatibilan sa izabranim operativnim sistemom. U slučaju da korisnik ručno odabere tip mrežnog adaptera, kompatibilnost nije zagarantovana.

Glavni operativni modovi virtuelne računarske mreže (načini na koje je virtuelna mrežna kartica povezana sa fizičkom računarskom mrežom *host* računara) koje *Oracle VirtualBox* podržava su:

- preslikavanje mrežne adrese (eng. *Network Address Translation – NAT*),
- spojena mreža (eng. *bridged networking*),
- interna mreža (eng. *internal networking*),
- mreža samo sa *host* računarom (eng. *host-only networking*) i
- preslikavanje mrežne adrese sa prosleđivanjem portova (eng. *NAT with port- forwarding*).

Izbor operativnog moda virtuelne računarske mreže određuje način na koji fizički mrežni adapter *host* računara „vidi" virtuelne mrežne adaptere virtuelnih mašina.

Za razliku od izbora tipa mrežnog adaptera koji zavisi od *guest* operativnog sistema, izbor operativnog moda virtuelne mreže zavisi pre svega od načina na koji korisnik želi da koristi svoju virtuelnu mašinu - da li virtuelna mašina igra ulogu servera ili klijenta i da li je potrebno da se obezbedi da virtuelna mašina bude vidljiva sa drugih fizičkih mašina na računarskoj mreži.

U nastavku su svi navedeni operativni modovi virtuelne *VirtualBox* računarske mreže ukratko opisani.

17.2.1. Mod preslikavanje mrežne adrese

Mod preslikavanja mrežne adrese (*NAT* mod) je podrazumevani mod prilikom kreiranja nove virtuelne mašine. Ovaj mod daje dobre rezultate u slučaju kada virtuelna mašina igra ulogu klijenta, tj. kada virtuelna mašina uglavnom inicira mrežne konekcije ka drugim mašinama.

Nakon butovanja guest operativnog sistema obično se koristi *DHCP* server, koji je ugrađen u virtuelizovani *NAT* uređaj, i pomoću koga guest operativni sistem dobija dinamička *IP* podešavanja (*IP* adresu, podrazumevanu masku podmreže, adresu podrazumevanog mrežnog prolaza i adrese *DNS* servera). *VirtualBox* za ove potrebe implementira sopstveni *DHCP* server, koji svim virtuelnim mašinama koje rade u *NAT* modu, dodeljuje ista *IP* podešavanja.

U *NAT* modu sve virtuelne mašine koje su implementirane na istom *host* računaru dobijaju istu *IP* adresu 10.0.2.15, zato što se svaka virtuelna mašina nalazi na sopstvenoj, izolovanoj virtuelnoj mreži, kojoj ni jedan drugi računar (virtuelna mašina) ne mogu da pristupe. Adresa podrazumevanog mrežnog prolaza za sve mašine je 10.0.2.2, što je adresa internog mrežnog porta virtuelnog *NAT* uređaja. Kada virtuelna mašina šalje paket ka spoljnoj mreži (na primer ka Internetu), *NAT* prepisuje informacije iz *IP* paketa, i ubacuje javnu *IP* adresu fizičkog rutera na koji je povezana mrežna kartica *host* računara.

Na ovaj način se poboljšava portabilnost, zato što će *guest* mašina da bude povezana na Internet mrežu čak i kada se *host* premešta s jedne na drugu lokaciju (mrežu). Tako na primer, ako se laptop računar premesti sa jedne na drugu bežičnu pristupnu tačku (eng. *Wireless Access Point – WAP*), ne postoji potreba za rekonfiguracijom mrežnih podešavanja guest operativnog sistema.

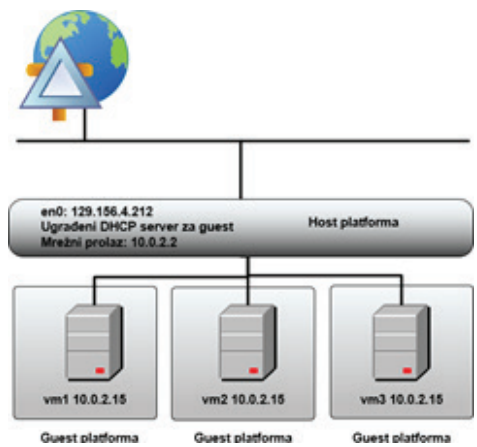
S obzirom na prirodu funkcionisanja *NAT* moda postoje ograničenja u njegovoj upotrebi. Tako na primer, nije moguće da se korisnik preko Internet mreže poveže na *Web* server koji je implementiran na guest operativnom sistemu sa *NAT* mrežnim adapterom, zato što Internet korisnik može da komunicira samo sa host operativnim sistemom, ne sa virtuelnim mašinama koje se nalaze iza virtuelnog *NAT* uređaja. Za ovakve i slične potrebe neophodno je da se koristi drugi *Oracle VirtualBox* mod virtuelizacije mreže.

Glavne karakteristike NAT mreže mogu da se svedu na sledeće:

- *guest* mašine se nalaze u svojoj privatnoj virtuelnoj *LAN* mreži,
- *VirtualBox* inkorporira usluge *DHCP* servera,
- *VirtualBox* virtuelni *NAT* uređaj prevodi (mapira) *IP* adrese,
- računari kojima *guest* šalje pakete vide kao da paketi dolaze sa fizičkog *host* računara,
- ne postoji potreba za dodatnim konfigurisanjem *host* i *guest* računara,
- *NAT* mod je dobar izbor tipa virtuelne mreže kada virtuelne mašine imaju ulogu klijenta i
- *NAT* mod je loš izbor virtuelne mreže kada virtuelne mašine imaju ulogu servera.

NAT mod virtuelne mreže prikazanje na Slici 17-1.

Slika 17-1: *NAT mrežni mod*



17.2.2. Mod spojene mreže

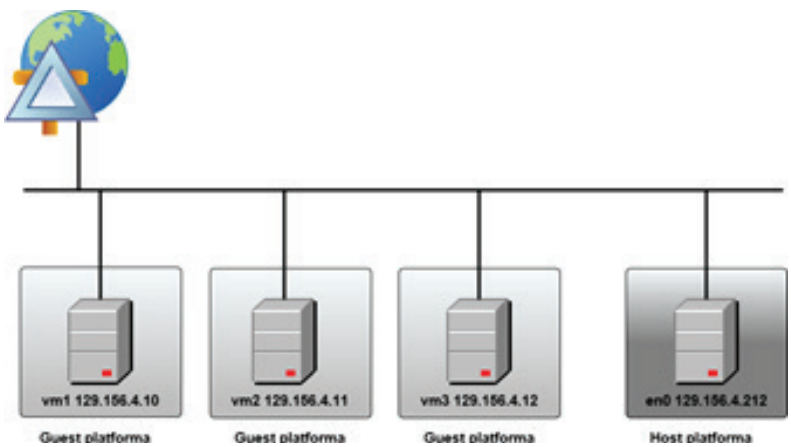
Mod spojene mreže se koristi u scenarijima kada je potrebno da virtuelna mašina bude ravnopravni učesnik mrežne komunikacije i ravnopravni član fizičke mreže na kojoj se nalazi *host* računar.

U ovom slučaju je virtuelna mrežna kartica direktno spojena sa mrežnom karticom fizičkog *host* računara.

Implikacija ovoga je da svaka virtuelna mašina ima isti pristup fizičkoj mreži kao što to ima i *host* računar. Virtuelna mašina može da pristupi svakom resursu i servisu fizičke mreže, poput *DHCP* ili *DNS* servisa, na isti način kao što to može i fizička mašina.

Logički prikaz moda spojene mreže dat je na Slici 17-2.

Slika 17-2: *Mod spojene mreže*



Kao dva najveća nedostatka moda spojene mreže navode se:

- u scenarijima gde postoji više virtuelnih mašina može da dođe do situacije da nestane raspoloživih *IP* adresa, bilo da su one statičke, bilo da su dinamičke i
- u slučaju da host računar ima više fizičkih mrežnih kartica (na primer jednu bežičnu i jednu žičnu karticu), svaki put kada se *host* računar prebacuje s jedne na drugu mrežu potrebno ponovno konfigurisati virtuelne mrežne adaptere virtuelnih mašina.

Neke od karakteristika moda spojene mreže su:

- *VirtualBox* spaja virtuelnu mrežu sa fizičkom računarskom mrežom *host* računara,
- mod spojene mreže je optimalna opcija u slučaju da virtuelna mašina igra ulogu servera,
- prekomerna potrošnja raspoloživih *IP* adresa,
- u određenim situacijama potrebno je dodatno konfigurisati *guest* mašinu i mod spojene mreže je najbolja opcija za produkciona virtuelna okruženja, kao što je klaud infrastruktura.

U većini scenarija iz realnog klaud okruženja koristi se kombinacija *NAT* moda i moda spojene mreže.

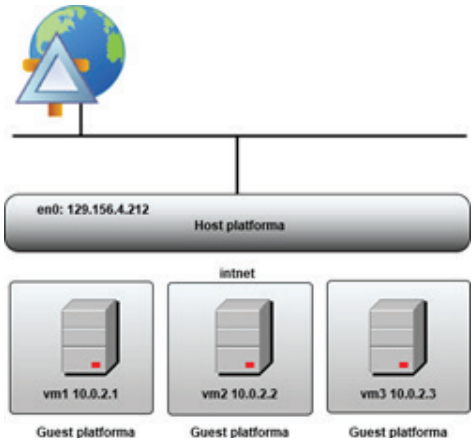
17.2.3. Mod interne mreže

U slučaju korišćenja moda interne mreže, sav mrežni saobraćaj između virtuelnih mašina ostaje samo u okvirima fizičke računarske mreže *host* računara i nikada ne napušta ovu mrežu.

Osim ovoga, mrežni saobraćaj mogu da vide samo virtuelne mašine koje se nalaze u istoj internoj privatnoj mreži.

Grafički prikaz moda interne mreže dat je na Slici 17-3.

Slika 17-3: *Mod interne mreže*



Interna mreža u primeru prikazanom na Slici 17-3 je podrazumevana *VirtualBox* interna mreža pod nazivom intnet. Ova mreža je potpuno izolovana i tiha mreža. Ovakav tip mreže daje dobre rezultate u scenarijima kada je potrebno da se implementira odvojena, čista mreža sa sofisticiranim mrežnim servisima i virtuelnim mašinama koje same pružaju usluge za sve učesnike na mreži, kao što su usluge aktivnog direktorijuma (eng. *active directory services*), *DHCP* servera, *DNS* servera i slično.

U modu interne mreže, čak ni sam host računar nije član ove mreže. Interna mreža funkcioniše čak u situacijama kada *host* računar nije povezan na Internet, niti na bilo koju drugu mrežu.

Kao napomena navodi se da u modu interne virtuelne računarske mreže *VirutalBox* ne nudi ni jednu uslugu, pa čak ni *DHCP* servis, već virtuelnim mašinama mora da se dodeli statička *IP* adresa. Alternativno, može da se implementira virtuelna mašina koja će imati ulogu *DHCP* servera i koja će da dodeljuje *IP* podešavanja svim virtuelnim mašinama u privatnoj mreži.

U realnim scenarijima mod interne mreže se najčešće koristi u kombinaciji sa drugim tipovima virtuelnih mreža, kao što je *NAT*, kada jedna ili više virtuelnih mašina imaju dva, ili više mrežnih adaptera, gde je jedan adapter konfigurisan u *NAT* modu i omogućava pristup virtuelnim mašinama u lokalnoj mreži Internetu i drugim spoljnim mrežama.

Kao neke od važnijih karakteristika interne mreže mogu da se izdvoje sledeće:

- *guest* mašine mogu da vide druge *guest* mašine na istoj privatnoj mreži,
- *host* mašina ne može da vidi virtuelne mašine, niti mrežni saobraćaj na internoj mreži,
- ovakvu mrežu je potrebno dodatno konfigurisati,
- mreža funkcioniše čak i kada *host* računar nije povezan na Internet, ili neku drugu eksternu mrežu,
- interni mod može da se koristi u kombinovanom režimu sa spojenom, ili *NAT* virtuelnom mrežom i
- interni mod je dobra opcija za implementiranje softverskih rešenja koja se sastoje iz više slojeva, gde jedan sloj može da „vidi" samo sloj iznad, ili ispod sebe.

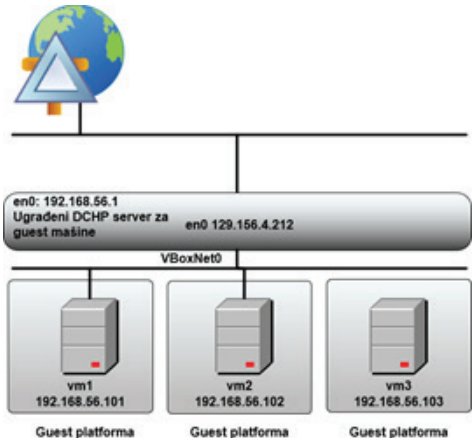
17.2.4. Mod mreže samo sa host računarom

Mod mreže samo sa host računarom sličan je modu interna mreža, zato što korisnik sam određuje kojoj virtuelnoj računarskoj mreži pripada guest virtuelna mašina. U ovom modu, virtuelna mreža je povezana samo sa *host* računarom, ne i sa mrežama na kojima se nalazi i sa kojima je povezan *host* računar.

Na Slici 17-4, koja vizuelizuje mod mreže samo sa *host* računarom, *guest* operativni sistemi se nalaze na virtuelnoj mreži *VBoxNet0*.

Sve virtuelne mašine koje se nalaze na *VBoxNet0* mreži mogu međusobno da se vide, i sve su povezane sa mrežnim adapterom host računara. Međutim, ni jedna eksterna mreža, čak ni fizička mreža host računara nije povezana sa internom virtuelnom mrežom *VBoxNet0*. Zbog toga je ovom operativnom modu i dat naziv mreža samo sa *host* računarom.

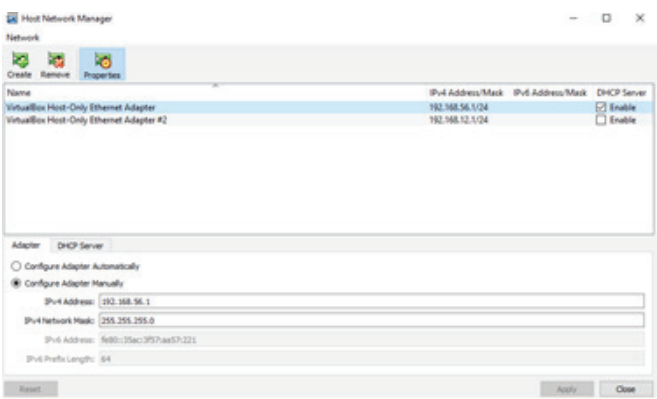
Slika 17-4: *Mod mreže samo sa host računarom*



Kao što je rečeno, modovi mreže samo sa host računarom i interne mreže su slični uz određene razlike. Kao jedna od razlika navodi se da u modu mreže samo sa *host* računarom, *host* računar obezbeđuje *DHCP* servis za *guest* virtuelne mašine na mreži i zato nema potrebe za statičkim *IP* adresama, niti za virtuelnom mašinom koja bi imala ulogu *DHCP* servera.

Oracle VirtualBox hipervizor ima intuitivan i jasan grafički korisnički interfejs za konfigurisanje moda mreže samo sa *host* računarom, koji se naziva *Host Network Manager*. Na Slici 17-5, prikazan je *screenshot* mrežnih podešavanja za dve ovakve mreže. Na prvom interfejsu sa Slike 17-5, konfigurisana su podrazumevana *IP* podešavanja moda mreže samo sa *host* računarom, gde se sve virtuelne mašine nalaze na mrežnom opsegu 192.168.56.0/24.

Slika 17-5: *Screenshot - Oracle VirtualBox Host Network Manager interfejs*



Neke od važnijih karakteristika moda mreže sa samo host računarom su:

- *VirtualBox* kreira privatnu internu mrežu za guest mašine i za *host* računar,
- *host* računar „vidi" virtuelne mrežne kartice virtuelnih mašina,
- *VirtualBox host* mrežni adapter obezbeđuje uslugu *DHCP* servera,
- *guset* mašine ne mogu da vide ni jednu drugu mašinu osim *host* računara na eksternoj mreži,
- *guest* mašine mogu putem interne mreže međusobno da komuniciraju čak i kada *host* nije povezan ni na jednu eksternu računarsku mrežu i
- ovaj mod predstavlja odličan izbor za implementiranje razvojnih i okruženja za testiranje.

17.2.5. Mod preslikavanja mrežne adrese sa prosleđivanjem portova

Do sada opisani modovi *VirtualBox* virtuelizacije mreže pokrivaju skoro sve realne scenarije. Međutim, postoje određene situacije za koje ni jedan od opisanih modova ne predstavlja optimalno rešenje.

Jedna od takvih situacija bi bila sledeća. Na serverskom fizičkom računaru instaliran je *Oracle VirtualBox* hipervizor koji upravlja sa nekoliko virtuelnih mašina koje imaju uloge servera i koriste se za isporučivanje klaud servisa eksternim korisnicima putem Internet mreže. Osim toga, s vremena na vreme fizički server se prebacuje s jedne na drugu fizičku mrežu.

U ovakvom scenariju prethodno prikazani modovi *Oracle VirtualBox* virtuelizacije mreže neće da daju zadovoljavajuće rezultate, zato što:

- *NAT* neće raditi, zato što je potrebno da se korisnici sa spoljnih mreža povezuju na virtuelne mašine,
- povezana mreža nije opcija, zato što mod povezane mreže troši mnogo *IP* adresa i potrebno je da se mreža rekonfiguriše svaki put kada se fizički server premešta s jedne na drugu fizičku mrežu,
- mod interne mreže takođe nije rešenje, zato što je potrebno da virtuelne mašine budu vidiljive sa spoljnih mreža i
- mod mreže samo sa *host* računarom u ovom scenariju ne može da se primeni, zato što je potrebno da eksterne, fizičke mašine komuniciraju sa virtuelnim mašinama.

Rešenje za ovako opisani scenario je mod preslikavanja mrežne adrese (*NAT* mreže) sa prosleđivanjem portova.

Kako bi se omogućio *NAT* mod mreže sa prosleđivanjem portova, prvo je potrebno da se podesi da virtuelni mrežni adapter virtuelne mašine koristi *NAT* operativni mod. Nakon toga, potrebno je da se definišu pravila za prosleđivanje portova. Kada su pravila za prosleđivanje portova definisana, eksterne mašine mogu da se povezuju na virtuelnu mašinu tako što „gađaju" *IP* adresu, ili *DNS* ime *host* računara i definisani port, koristeći sledeću sintaksu: *host_adresa:broj_porta*. Primenom pravila za prosleđivanje portova, *VirtualBox* ovakve konekcije preusmerava na odgovarajuće virtuelne mašine.

Prosleđivanje portova vrši se preko *Oracle VirtualBox Port Forwarding Rules* interfejsa.

Tako na primer, ako je implementirana virtuelna mašina koja radi u *NAT* mrežnom režimu, na kojoj se nalazi *Web* server i *SSH* server, mogu da se konfigurišu pravila za prosleđivanje portova, kako bi i eksterni korisnici mogli da pristupaju ovim servisima. Primer ovakvih pravila dat je na Slici 17-6.

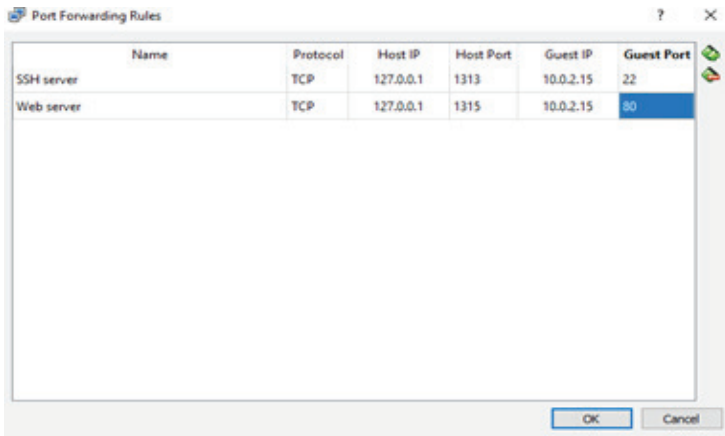
Sa Slike 17-6 vidi se da je u oba slučaja za adresu *host* računara (*Host IP*) podešena adresa 127.0.0.1, što je lokalna adresa povratne petlje (eng. *loopback address*) fizičkog mrežnog adaptera *host* računara. Ova adresa koristi se samo za potrebe prosleđivanja portova. Kada spoljni korisnici pristupaju virtuelnoj mašini, oni će da „gađaju" stvarnu *IP* adresu fizičkog mrežnog adaptera *host* računara (na primer 135.25.43.13). Za vrednost *Host Port* parametra može da se izabere bilo koja vrednost, dok se za vrednost parametra *Guest Port* koristi podrazumevani broj porta servisa koji je implementiran na virituelnoj mašini – u ovom slučaju port 22 za pristup *SSH* serveru i port 80 za pristup *Web* serveru.

Uz pretpostavku da je javna *IP* adresa fizičkog mrežnog adaptera *host* računara 135.24.43.13, pristup virtuelnoj mašini čija su podešavanja data na Slici 17-6, može da se ostvari na sledeći način:

- pristup *SSH* serveru ostvaruje se korišćenjem sintakse 135.24.43.13:1313 i
- pristup *Web* serveru ostvaruje se korišćenjem sintakse 135.24.43.13:1315.

VirtualBox nudi moćne funkcije za implementiranje i konfigurisanje virtuelnih mreža. Kao što je i prikazano, postoji više modova, i u praksi se najčešće koriste hibridni pristupi, gde se kombinuju dva ili više moda.

Slika 17-6: *Screenshot - Oracle VirtualBox Port Forwarding Rules interfejs*



17.3. IMPLEMENTIRANJE I KONFIGURISANJE VIRTUELNIH MAŠINA U ORACLE VIRTUALBOX OKRUŽENJU

Za potrebe ovog praktičnog primera kreirane su tri *Linux* virtuelne mašine u okruženju *Oracle VirtualBox* hipervizora. Aktuelna verzija *Oracle VirtualBox* može da se preuzme sa sledeće *URL* adrese: https://www.virtualbox.org/wiki/Downloads.

U ovoj simulaciji *VirutalBox* je instaliran u okruženju *Windows host* operativnog sistema.

Prva virtuelna mašina ima naziv *gitsrv* i na njoj je instalirana trenutno najsvežija verzija serverskog operativnog sistema *Ubuntu Linux - Ubuntu Server* 18.04 *LTS*. Akronim *LTS* označava *Long Term Support* i podrška za ovu verziju je obezbeđena do 2023. godine. Na virtuelnoj mašini *gitsrv* imlementirana je *Git* server komponenta, što je prikazano u Glavi 18.

Ubuntu Server 18.04 *LTS* može da se preuzme sa sledeće *URL* adrese: https://www.ubuntu.com/download/server.

Druga virtuelna mašina ima naziv gitclient i na njoj je instalirana trenutno najsvežija verzija desktop operativnog sistema *Ubuntu Linux - Ubuntu Desktop* 18.04 *LTS*. Na virtuelnoj mašini gitclient implementirana je klijentska *Git* komponenta, što je prikazano u Glavi 18.

Treća virtuelna mašina ima naziv *gitclient2* i ona je klonirana verzija druge mašine, *gitclient*, uz sitne razlike u mrežnim podešavanjima.

Ubuntu Desktop 18.04 *LTS* može da se preuzime sa sledeće *URL* adrese: https://www.ubuntu.com/download/desktop.

Poznato je da su sistemski zahtevi za instaliranje *GNU/Linux* operativnih sistema slabiji nego što je to slučaj sa *Windows*, ili *Mac OS X* operativnim sistemima. Minimalni sistemski zahtevi i za *Ubuntu desktop* i za *Ubuntu* server verzije su sledeći:

- dvojezgarni procesor od 2GHz radnog takta,
- 2 GB sistemske *RAM* memorije,
- 25 GB slobodnog prostora na disku i
- poželjan pristup Internetu.

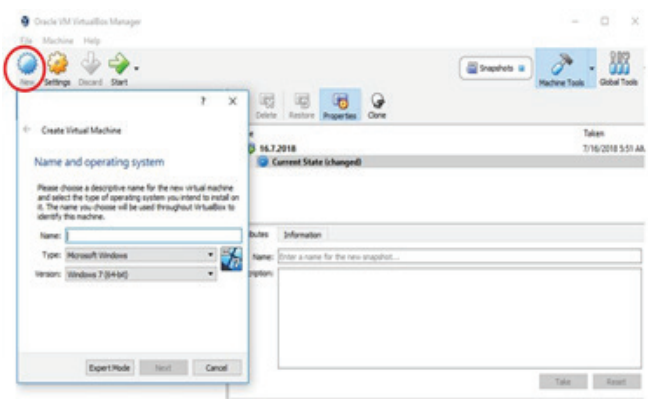
Implementiranje novih virtuelnih mašina u okruženju *Oracle VirtualBox* hipervizora može da se vrši na dva načina – pomoću grafičkog korisničkog interfejsa, ili korišćenjem konzolnog interfejsa. Za potrebe ovog praktičnog primera korišćen je grafički korisnički interfejs.

U nastavku je prvo prikazana instalacija gitsrv virtuelne mašine.

17.3.1. Implementacija gitsrv virtuelne mašine

Nakon instaliranja i pokretanja *Oracle VirtualBox* hipervizora, na paleti sa alatkama potrebno je kliknuti na dugme *New*, da bi se pokrenuo čarobnjak (eng. *wizard*) za kreiranje nove virtuelne mašine (Slika 17-7).

Slika 17-7: *Screenshot - pokretanje wizarda za kreiranje nove virtuelne mašine*



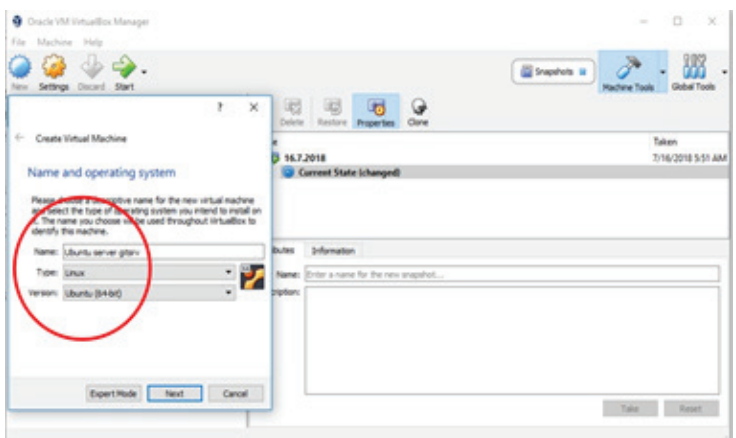
Pokrenuti *wizard* omogućava konfigurisanje osnovnih postavki virtuelne mašine. U polje *Name* potrebno je upisati opisno ime virtuelne mašine. Nakon toga potrebno je iz padajuće liste da se izabere tip i verzija operativnog sistema koji će biti instaliran. U ovom slučaju, za tip operativnog sistema izabran je *Linux*, dok je za verziju izabrana opcija *Ubuntu* (64-bit), kao što se vidi sa Slike17-8.

Važno je da se izabere operativni sistem koji će zaista biti instaliran, jer na osnovu ovog izbora *VirutalBox* generiše podrazumevanu konfiguraciju (podešavanja) virtuelne mašine. Na primer, ne podržavaju svi operativni sistemi sve tipove mrežnih adaptera koje *VirtualBox* nudi i *VirtualBox* će u ovom slučaju novokreiranoj virtuelnoj mašini da dodeli tip mrežnog adaptera koji će sigurno biti podržan od strane *guest* operativnog sistema. Naravno, podešavanja virtuelne mašine kasnije mogu da se menjaju.

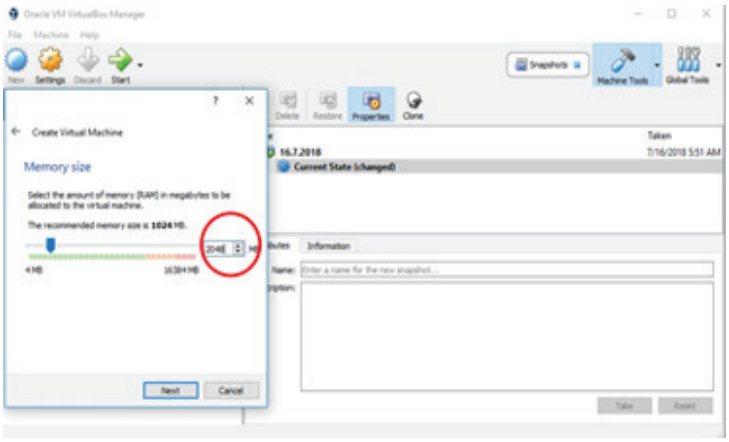
Kikom na dugme *Next, wizard* upućuje na dalje korake za podešavanje virtuelne mašine.

U koraku *Memory size*, definiše se količina *RAM* memorije koju host operativni sistem dodeljuje *guest* virtuelnoj mašini. *Wizard* preporučuje 1024 MB (1 GB) za veličinu sistemske *RAM* memorije. U ovom slučaju preporučena količina *RAM* memorije je povećana na 2048 MB (2 GB), kao što je prikazano na Slici 17-9.

Slika 17-8: *Screenshot - izbor tipa i verzije operativnog sistema*



Slika 17-9: *Screenshot - definisanje količine RAM memorije*



U sledećem koraku pod nazivom *Hard disk*, potrebno je da se izabere opcija *Create virtual hard disk now* i da se klikne na dugme *Create*. *Wizard* nakon toga pita za format virtuelnog diska, gde je potrebno da se izabere *VDI* (*VirtualBox Disk Image*) format.

U sledeća dva koraka potrebno je podesiti tip diska na *dynamically allocated*, a za veličinu diska izabrati 25GB. U ovom koraku korisnik može da promeni podrazumevani naziv virutelnog diska, gde je podrazumevani naziv isti kao i ime virtulne mašine. Na sličan način korisnik može da promeni i lokaciju na fizičkom disku gde se fajl virtuelnog diska čuva.

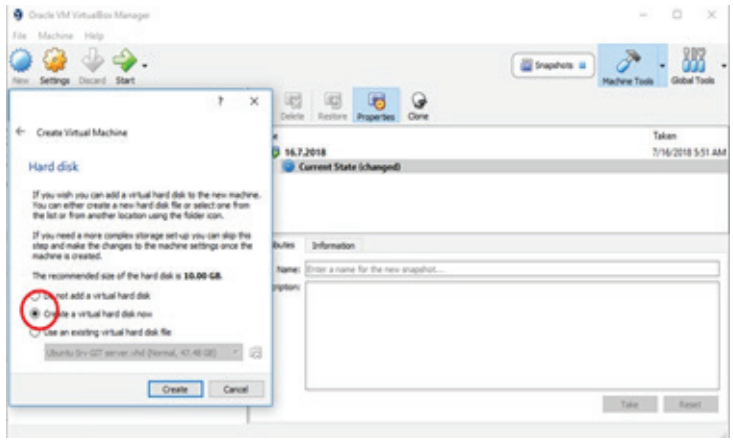
Svi navedeni koraci prikazani su na slikama 17-10, 17-11, 17-12 i 17-13.

Potrebno je da se skrene pažnja da *wizard* u koracima za konfigurisanje virtuelnog diska daje mogućnost izbora virtuelnog diska fiksne veličine i dinamičkog diska. Fiksni diskovi odmah nakon kreiranja uzimaju sav potreban prostor od fizičkog diska *host* računara, dok se dinamički diskovi povećavaju po potrebi (kako se koriste i popunjavaju podacima).

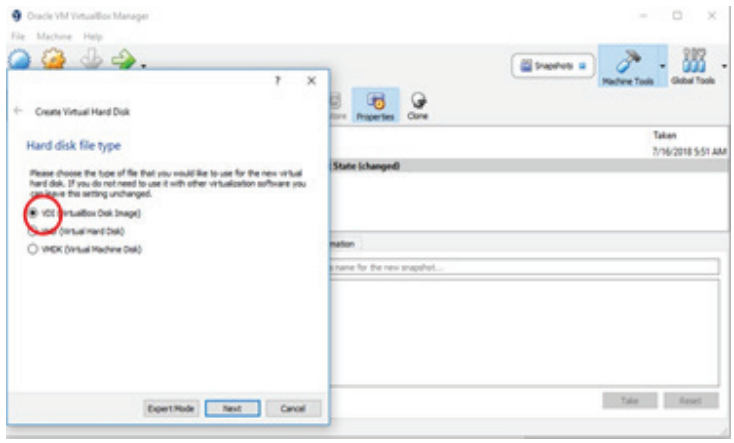
Tako na primer, ako se kreira fiksni disk veličine 1TB, *VDI* fajl na fizičkom će odmah nakon kreiranja da zauzima prostor od 1TB, bez obzira što je virtuelni disk u početku prazan. S druge strane, veličina dinamičkog diska je na početku bliska nuli, dok se tokom eksploatacije diska njegova veličina povećava.

U teroriji su zbog navedenih karakteristika fiksni diskovi brži od dinamičkih, ali na savremenom računarskom hardveru (posebno na hardveru koji se koristi na klaudu), korisnik ne može da oseti razliku u performansama između fiksnih i dinamičkih diskova.

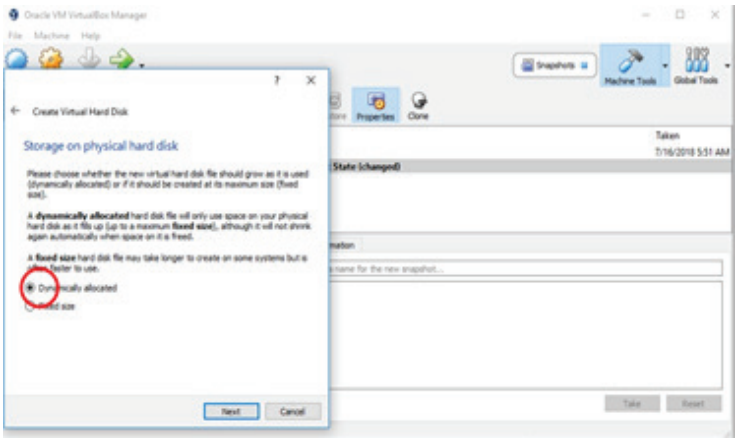
Slika 17-10: *Screenshot - kreiranje virtuelnog diska (1)*



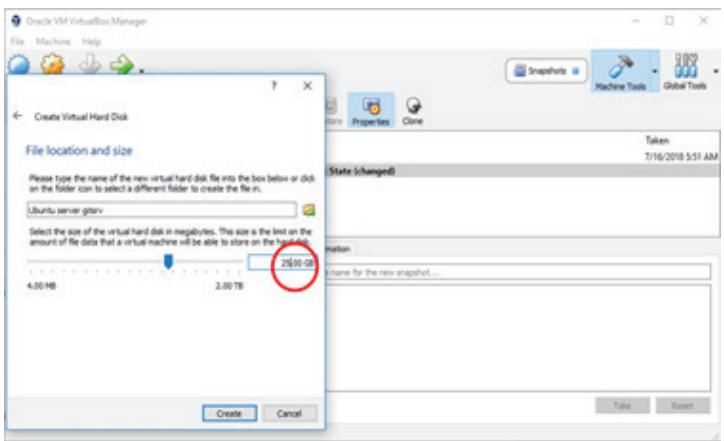
Slika 17-11: *Screenshot - kreiranje virtuelnog diska (2)*



Slika 17-12: Screenshot - kreiranje virtuelnog diska (3)



Slika 17-13: *Screenshot - kreiranje virtuelnog diska (4)*

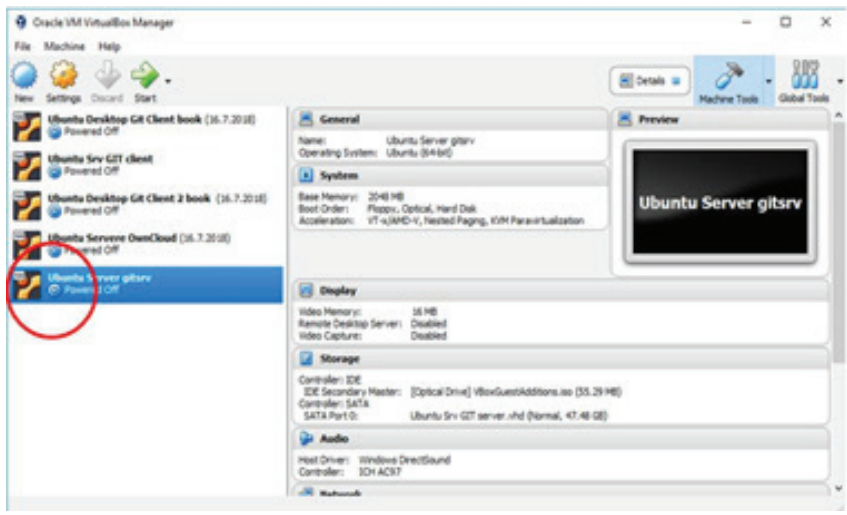


Nakon podešavanja virtuelnog diska, virtuelna mašina je kreirana i nalazi se u listi virtuelnih mašina u *Oracle VM VirtualBox Manager* prozoru (Slika 17-14).

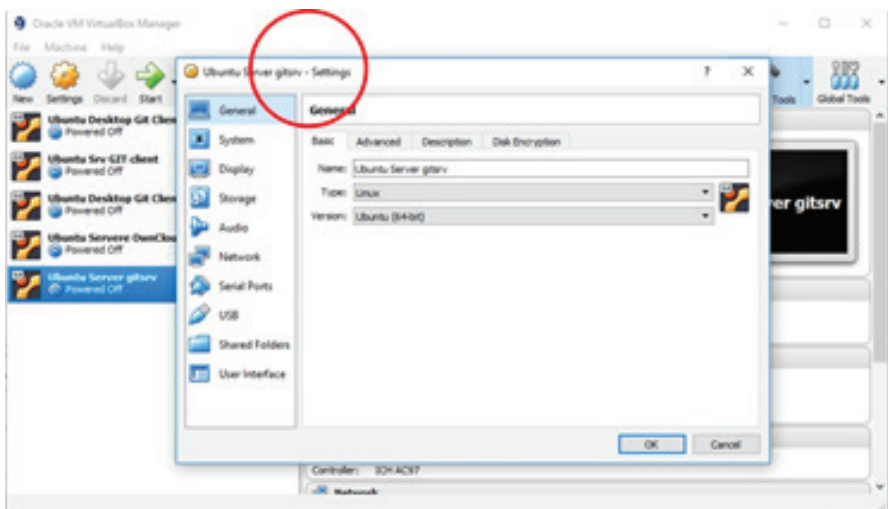
Pre instaliranja *guest* operativnog sistema potrebno je da se obezbedi da virtuelna mašina ima pristup instalacionom medijumu. U ovom slučaju instalacioni medijum je *ISO* fajl koji je preuzet sa zvaničnog *Ubuntu* portala. Da bi se instalacioni medijum pridružio virtuelnoj mašini, potrebno je da se izmene neka od podešavanja virtuelne mašine.

Prvo se pomoću *Oracle VM VirutalBox Manager* komponente izabere željena virtuelna mašina, a zatim se iz palete sa alatkama (eng. *toolbar*) izabere opcija *Settings*, nakon čega se otvara prozor sa podešavanjima virtuelne mašine (Slika 17-5).

Slika 17-14: *Screenshot - Oracle VM VirutalBox Manager*



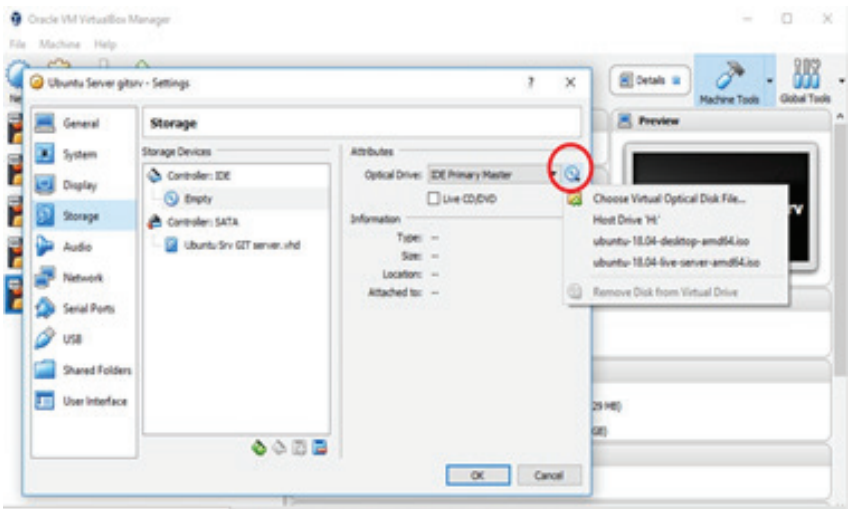
Slika 17-15: *Screenshot - prozor sa podešavanjima virtuelne mašine*



U grupi opcija za podešavanje virtuelne mašine potrebno je da se izabere opcija *Storage*. Zatim, u okviru opcije *Storage* potrebno je da se izabere *Empty* ispod naznake *IDE Controller*. Izborom ove opcije otvara se grupa opcija sa *CD/DVD* atributima. Klikom na *CD/DVD* ikonicu pored *Optical Drive* padajuće liste otvara se druga grupa opcija, gde je potrebno da se izabere *Choose Virtual Optical Disk File*. Konačno, izborom ove opcije, korisnik treba da označi preuzeti *ISO* fajl sa instalacionim paketom *Ubuntu* Server 18.04 *LTS* operativnog sistema na svom lokalnom disku.

Prikaz opcije za izbor virtuelnog instalacionog medijuma dat je na Slici 17-16.

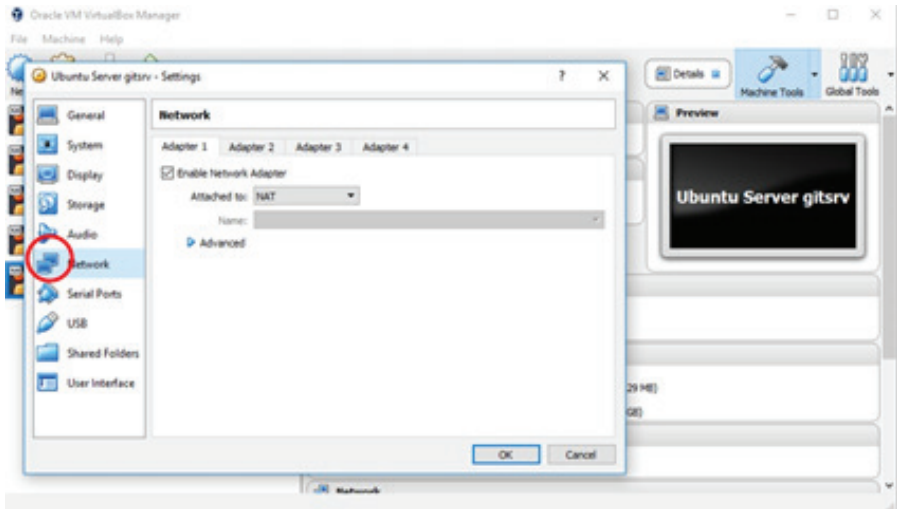
Slika 17-16: *Screenshot - izbor virtuelnog instalacionog medijuma (ISO fajl)*



Podešavanje mrežnih adaptera gitsrv virtuelne mašine

Pre pokretanja *gitsrv* virtuelne mašine potrebno je da se izvrše podešavanja mrežnih adaptera. Podešavanje mrežnih adaptera takođe se vrši u prozoru sa podešavanjima virtuelne mašine, ali ovog puta izborom opcije *Network* (Slika 17-17).

Slika 17-17: *Screenshot - prozor sa podešavanjima mrežnih adaptera*



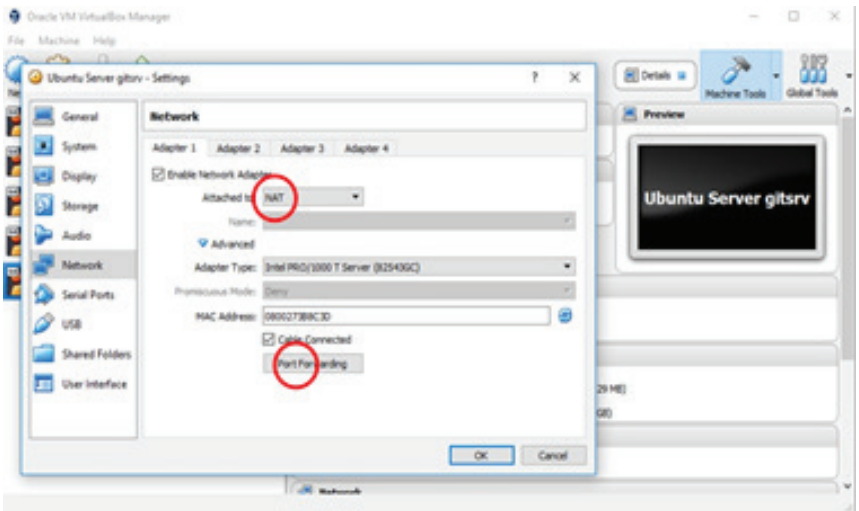
Obe virtuelne mašine koje su korišćene u ovom primeru imaju dva mrežna adaptera. Pomoću prvog adaptera, koji je tipa *NAT*, virtuelne mašine mogu da pristupe Internetu i obe mašine dobijaju istu *IP* adresu (10.0.2.15/24) od *DHCP* servera virtuelnog *NAT* uređaja. Dakle, preko prvog mrežnog adaptera, za svaku virtuelnu mašinu je kreirana posebna izolovana virtuelna mreža sa Internet pristupom.

Međutim, za potrebe implementacije *Git* server – *Git* klijent okruženja, potrebno je da virtuelne mašine međusobno komuniciraju, ali to to nije moguće korišćenjem prvog mrežnog adaptera. Zbog toga je potrebno dodati i drugi mrežni adapter, i konfigurisati ga da radi u modu interne mreže. Ovaj drugi adapter služi za komunikaciju između virtuelnih mašina. Kada se koristi mod interne mreže, *DHCP* usluga nije dostupna i zbog toga je potrebno da se konfigurišu statičke *IP* adrese za obe mašine na drugom virtuelnom mrežnom adapteru. Prilikom konfigurisanja statičkih *IP* adresa, važno je da *IP* adrese obe mašine pripadaju istoj logičkoj podmreži, zato što će samo na taj način mašine moći međusobno da komuniciraju. Konfigurisanje ovih adresa vrši se na nivou operativnog sistema, što je prikazano u nastavku.

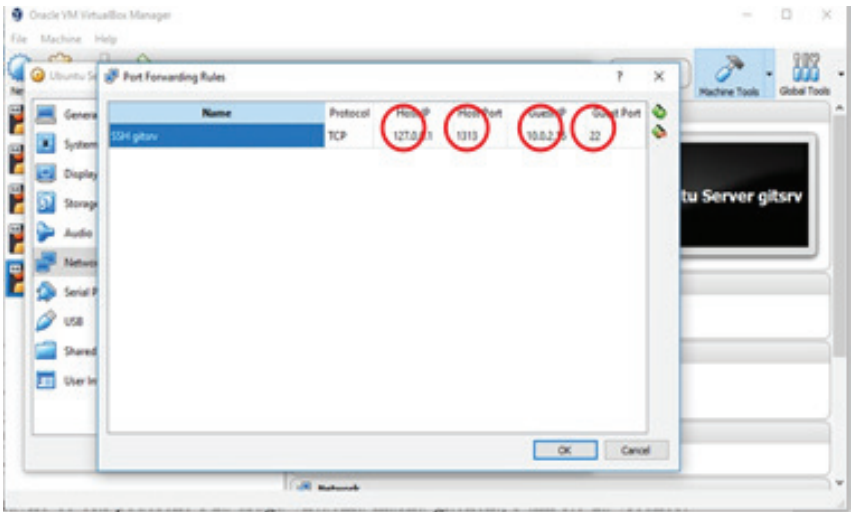
Dakle, za prvi virtuelni mrežni adapter potrebno je izabrati opciju *NAT*, kao što je prikazano na Slici 17-18. Virtuelnim mašinama koje se nalaze iza virtuelnog *NAT* uređaja pristupa se pomoću *SSH* klijenta iz okruženja *host* računara. Zbog toga je za prvi mrežni adapter potrebno definisati i pravila za prosleđivanje portova.

Pravila za prosleđivanje portova definišu se u grupi naprednih opcija (*Advanced*) za podešavanje mrežnog adaptera klikom na dugme *Port Forwarding* (Slika 17-18). Nakon toga, otvara se prozor *Port Forwarding Rules* za dodavanje pravila prosleđivanja portova. Ovaj prozor sa definisanim pravilom za *SSH* pristup prikazan je na Slici 17-19.

Slika 17-18: *Screenshot – grupa opcija za podešavanje prvog mrežnog adaptera*



Slika 17-19: *Screenshot – podešavanje pravila za prosleđivanje portova na prvom mrežnom adapter*



Dakle, kao što je već i rečeno, u slučaju mrežnog adaptera koji koristi *NAT* mod, virtuelnoj mašini ne može direktno da se pristupi, već joj se pristupa preko *host* računara. Za adresu host računara koristi se loopback adresa (opcija *Host IP*) i proizvoljan broj porta (opcija *Host Port*, u ovom slučaju, izabran je slučajan port 1313). Definisanjem pravila sa Slike 17-19, sve konekcije ka *host* računaru preko porta 1313 preusmeravaju se na *gitsrv* virtuelnu mašnu sa *IP* adresom 10.0.2.15 (opcija *Guest IP*) i port 22 (opcija *Guest Port*). Kao što je navedeno u Sekciji 17.2.1, sve virtuelne mašine koje se nalaze iza virtuelnog *NAT* uređaja dobijaju *IP* adresu 10.0.2.15. U pravilu je izabran port 22, zato što *SSH* server, koji je implementiran na virtuelnoj mašini, koristi port 22 za komunikaciju sa *SSH* klijentima.

Slično pravilo je podešeno i za drugu virtuelnu mašnu - *gitclient*, s tim što za vrednost *Host Port* mora da se izabere broj porta koji je različit 1313, zato što se i jednoj i drugoj virtuelnoj mašini pristupa preko istog fizičkog adaptera *host* računara i njegove loopback adrese.

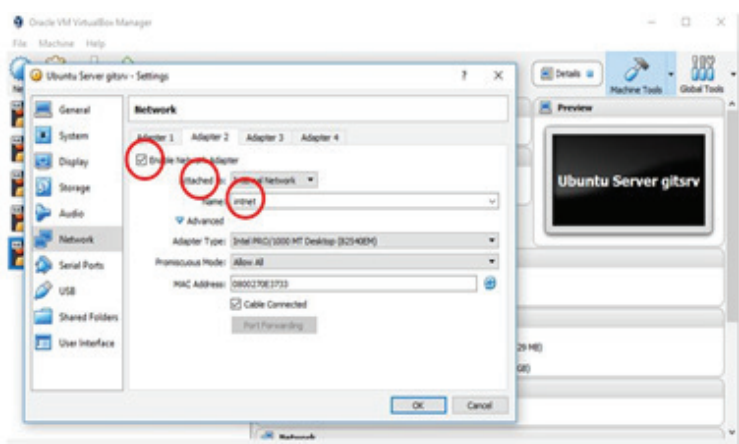
Za potrebe konfigurisanja drugog mrežnog adaptera, prvo je potrebno da se izabere tab *Adapter2* u mrežnim podešavanjima virtuelne mašine. Nakon toga potrebno je označiti opciju *Enable Network Adapter* i za tip adaptera izabrati *Internal*. Za naziv interne mreže (polje *Name*) može da se ostavi *initnet*. Naziv interne mreže je proizvoljan, a *VirtualBox* podrazumevano koristi naziv *initnet*.

U navedenim postavkama važno je samo da obe virtuelne mašine pristupaju istoj privatnoj mreži, zato što samo na taj način mašine mogu međusobno da komuniciraju.

Drugom mrežnom adapteru *gitsrv* virtuelne mašine se dodeljuje statička *IP* adresa preko konfiguracionih fajlova operativnog sistema.

Podešavanje drugog mrežnog adaptera prikazno je na Slici 17-20.

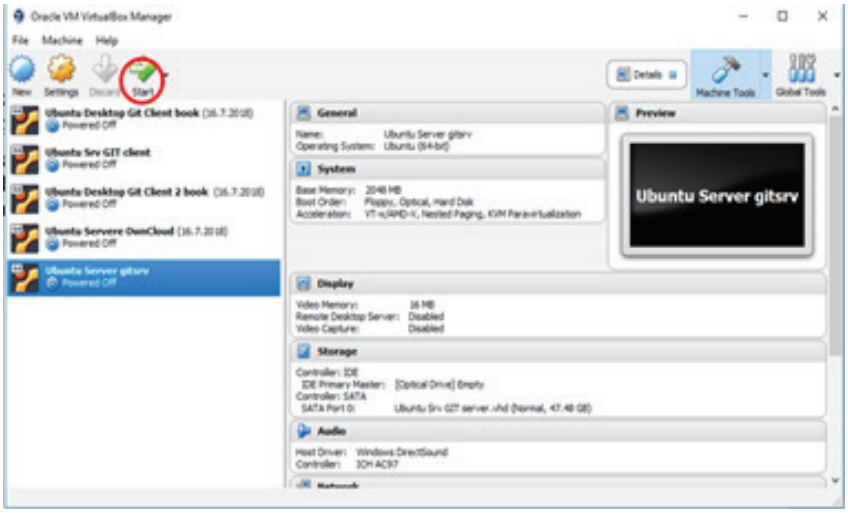
Slika 17-20: *Screenshot – podešavanje drugog mrežnog adaptera*



Pokretanje gitsrv virtuelne mašine i instaliranje operativnog sistema

Nakon konfigurisanja *gitsrv* virtuelne mašine, mašina može da se pokrene izborom opcije *Start* iz palete alatki *Oracle VM VirtualBox Manager* (Slika 17-21).

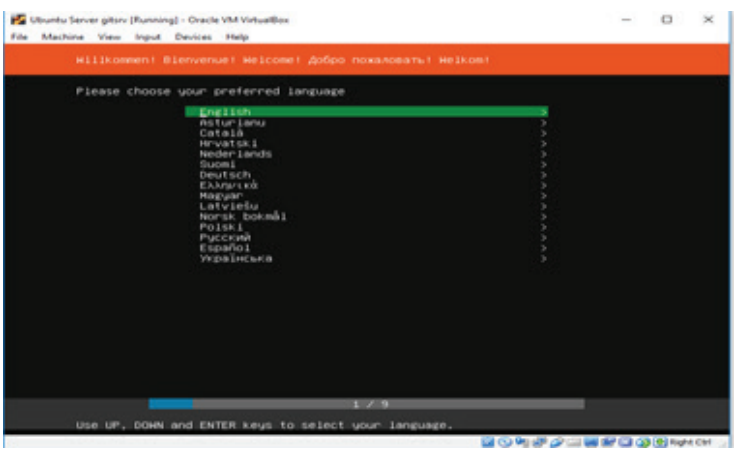
Slika 17-21: *Screenshot – pokretanje virtuelne mašine*



Nakon pokretanja, virtuelna mašina se butuje pomoću *Ubuntu* Server 18.04 *LTS* instalacionog medijuma i proces instalacije se pokreće (Slika 17-22).

U prvom koraku instalacije potrebno je izabrati željeni jezik. U ovom slučaju izabran je engleski (*English*).

Slika 17-22: *Screenshot – pokretanje instalacije operativnog sistema Ubuntu Server*

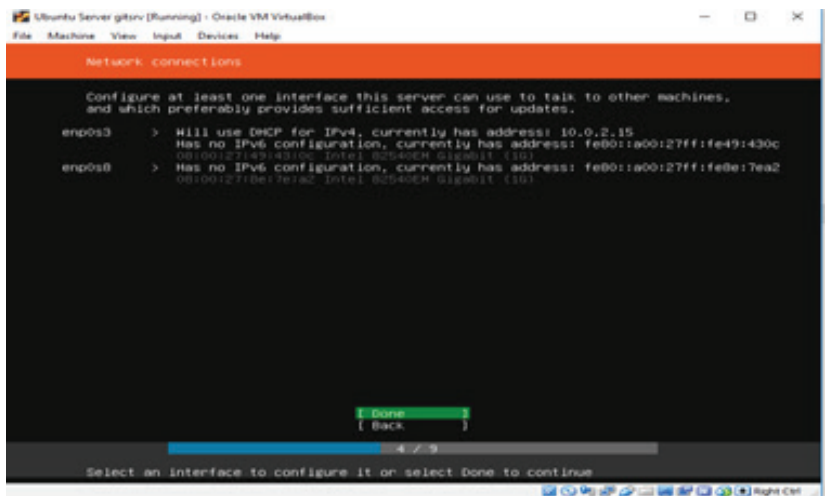


Već u sledećem koraku instalacije, *Ubuntu* prepoznaje mrežne adaptere *enp0s3* i *enp0s8* koji rade u *NAT* i internom modu, respektivno. *VirtualBox DHCP* server adapteru *enp0s3* dodeljuje dinamičku *IP* adresu, dok je za drugi adapter *enp0s8*¸ potrebno da se konfiguriše statička *IP* adresa.

Statička *IP* adresa za mrežni adapter *enp0s8* konfigurisana je u narednim koracima. U *Linux* okruženju se skoro sva podešavanja vrše preko konfiguracionih fajlova. Isto je i u slučaju *IP* podešavanja.

Korak instalacije *Ubuntu Server* operativnog sistema, gde *Ubuntu* prepoznaje mrežne adaptere virtuelne mašine prikazan je na Slici 17-23.

Slika 17-23: *Screenshot – Ubuntu Server instalacija (mrežni adapter)*

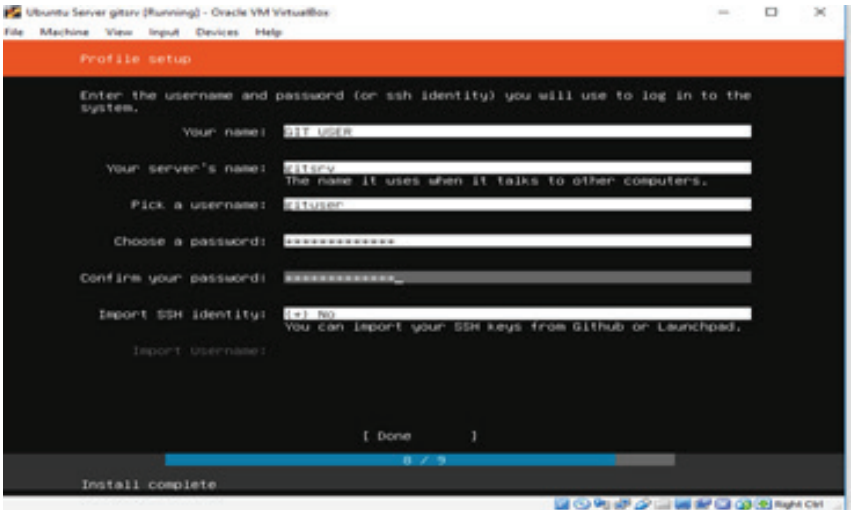


U koraku instalacije *Profile setup* kreira se prvi korisnik sistema koji ima *admin* privilegije, i upisuju se podaci za ime korisnika, korisničko ime i biraju se željena lozinka i naziv sistema.

U ovom slučaju korisničko ime je *gituser*, a naziv računara je *gitsrv* (Slika 17-24).

U Glavi 18 prikazana je implementacija *Git* servera u kontekstu privilegija korisnika *gituser*.

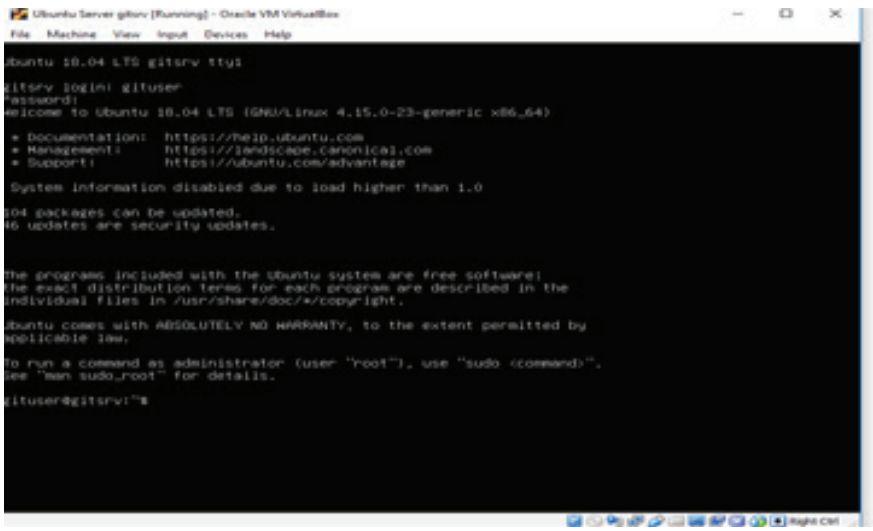
Slika 17-24: *Screenshot – Ubuntu Server instalacija (Profile setup)*



Za sve ostale korake instalacije potrebno je izabrati podrazumevana podešavanja. Nakon završetka procesa instalacije, otvara se standardni *Linux bash* (*Bourne Again Shell*) i može da se izvrši logovanje na sistem sa kredencijalima koji su definisani u koraku instalacije *Profile setup* (korisnik *gituser* i odabrana lozinka).

Proces logovanja na *Linux bash* prikazan je na Slici 17-25.

Slika 17-25: *Screenshot – Logovanje na Linux bash*



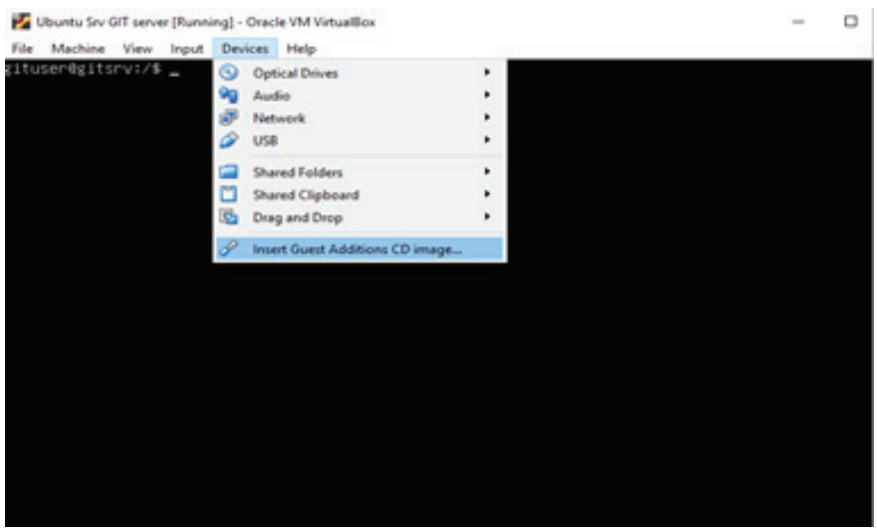
Instaliranje VirtualBox Guest Additions alata

Nakon instaliranja *Ubuntu Server* 18.04 *LTS* operativnog sistema prvi zadatak je instaliranje *VirtualBox Guest Additions* alata koji poboljšava performanse operativnog sistema virtuelne mašine, olakšava proces administriranja virtuelne mašine i omogućava bolju integraciju *guest* operativnog sistema sa *host* operativnim sistemom.

Kao neke od dodatnih funkcionalnosti *VirtualBox Guest Additions* softverskog paketa navode se sledeće: unapređenje grafičkih performansi virtuelne mašine, omogućavanje kreiranja deljenih (zajedničkih) foldera kojima mogu da pristupaju korisnici i sa *host* i sa *guest* operativnih sistema i interoprabilnost funkcionalnosti isecanja teksta i integracije miša između *host* i *guest* operativnih sistema.

Da bi se instalirao *VirtualBox Guest Additions*, prvo mora u optički uređaj virtuelne mašine da se ubaci instalacioni medijum. Potrebno je da se klikne na stavku menija *Devices* koja se nalazi iznad terminala virtuelne mašine i da se izabere *Insert Guest Additions CD Image* (Slika 17-26). Instalacioni medijum je standardni *ISO* fajl.

Slika 17-26: *Screenshot – VirtualBox Guest Additions instalacioni medijum*



Sada je *VirtualBox Guest Additions CD Image* ubačen u optički uređaj virtuelne mašine. Da bi se pristupilo sadržaju *ISO* fajla (instalacionog medijuma), ovaj fajl prvo mora da se montira (eng. *mount*) izvršavanjem komande koja je prikazana u Listingu 17-1.

Listing 17-1: *Montiranje instalacionog medijuma VirtualBox Guest Additions*

```
gituser@gitsrv:/$ sudo mount /dev/cdrom /mnt
mount: /mnt: WARNING: device write-protected, mounted read-only.
```

Nakon izvršavanja ove komande, virtuelni instalacioni medijum je montiran na folder */mnt*, čiji je sadržaj dat u Listingu 17-2.

Listing 17-2: *Sadržaj foldera /mnt*

```
gituser@gitsrv:/$ cd /mnt
gituser@gitsrv:/mnt$ ls
32Bit          cert            VBoxLinuxAdditions.run          VBoxWindowsAdditions-
x86.exe
64Bit          OS2             VBoxSolarisAdditions.pkg
AUTORUN.INF    runasroot.sh    VBoxWindowsAdditions-amd64.exe
autorun.sh     TRANS.TBL       VBoxWindowsAdditions.exe
```

Pre pokretanja instalacije *VirutalBox Guest Additions* paketa potrebno je da se proveri da li su svi alati i biblioteke koji su neophodni za instaliranje implementirani. Za ove potrebe koristi se *apt-get install* komanda (Listing 17-3). Ako potrebne biblioteke nisu implementirane, pokretanjem ove komande sve komponente će automatski biti instalirane.

Listing 17-3: *Instaliranje potrebnih biblioteka za pokretanje VirutalBox Guest Additions*

```
gituser@gitsrv:/$ apt-get install -y dkms build-essential linux-headers-generic
linux-headers-$(uname -r)
```

Konačno, komanda za pokretanje instalacije *VirutualBox Guest Additions* paketa data je u Listingu 17-4.

Listing 17-4: *Pokretanje instalacije VirutalBox Guest Additions*

```
gituser@gitsrv:/mnt$ sudo ./VBoxLinuxAdditions.run
```

Nakon instalacije potrebno je restartovati virtuelnu mašinu komandom *sudo reboot*.

Konfigurisanje mrežnih adaptera gitsrv virtuelne mašine na nivou operativnog sistema

Nakon restartovanja *gitsrv* virtuelne mašine potrebno je ponovo izvršiti logovanje na sistem korišćenjem kredencijala korisnika *gituser*.

Nakon logovanja, pokretanjem *bash* komande *ifconfig* mogu da se vide *IP* podešavanja mrežnih adaptera (Listing 17-5).

Listing 17-5: *Izvršavanje bash komande ifconfig*

```
gituser@gitsrv:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe3b:8c3d prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:3b:8c:3d txqueuelen 1000  (Ethernet)
        RX packets 763 bytes 363576 (363.5 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 444 bytes 59732 (59.7 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet6 fe80::a00:27ff:fe0e:3733 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:0e:37:33 txqueuelen 1000  (Ethernet)
        RX packets 69 bytes 23598 (23.5 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 14 bytes 1076 (1.0 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 92 bytes 6904 (6.9 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 92 bytes 6904 (6.9 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Kao što se vidi iz priloženog listinga, oznake mrežnih adaptera su redom *enp0s3, enp0s8* i *lo*. Adapter sa oznakom lo ukazuje na *loopback* adresu same virtuelne mašine, dok su *enp0s3* i *enp0s8* oznake *NAT* adaptera i internog adaptera, respektivno. Iz listinga se jasno vidi da za mrežni adapter *enp0s8* nisu konfigurisana *IP* podešavanja, dok je adapteru *enp0s3 DHCP* servis dodelio *IP* adresu 10.0.2.15.

Prvo je potrebno da se mrežnom adapteru *enps08* dodeli statička *IP* adresa. Prethodne verzije *Ubuntu* operativnog sistema koristile su *ifupdown* komandu i konfiguracioni fajl */etc/network/interfaces* za *IP* podešavanja mrežnih adaptera. Za ove potrebe, *Ubuntu 18.04 LTS* verzija koristi noviji alat pod nazivom *netplan*.

Da bi se mrežnom interfejsu *enp0s8* dodelila statička *IP* adresa potrebno je da se edituje fajl */etc/netplan/50-cloud-init.yaml*, gde se upisuju statička mrežna podešavanja ovog adaptera. Najbolji način za editovanje fajla je korišćenje *nano* editora, izvršavanjem jednostavne komande *sudo nano /etc/netplan/50-cloud-init.yaml*. Nakon otvaranja ovog konfiguracionog fajla, potrebno je izmeniti njegov sadržaj, kao što je prikazano u Listingu 17-6 .

Listing 17-6: *Izmena fajla etc/netplan/50-cloud-init.yaml*

```
# This file is generated from information provided by
# the datasource. Changes to it will not persist across an instance.
# To disable cloud-init's network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    version: 2
    ethernets:
        enp0s3:
            addresses: []
            dhcp4: true
            optional: true
        enp0s8:
            dhcp4: no
            addresses: [10.0.0.101/24]
```

Iz priloženog listinga vidi se da mrežni adapter *enp0s3 IP* podešavanja dobija od *DHCP* servera, dok je za mrežni adapter *enp0s8* konfigurisana statička *IP* adresa 10.0.0.101/24. Podrazumevani mrežni prolaz, masku podmreže i adresu *DNS* servera nije potrebno konfigurisati, jer mrežni interfejs *enp0s8* služi isključivo za komunikaciju između virtuelnih mašina, a u ovom slučaju služi da bi mašina gitsrv komunicirala sa mašinama *gitclient* i *gitclient2*.

Nakon izmene fajla *etc/netplan/50-cloud-init.yaml*, nova podešavanja se primenjuju bez potrebe da se virtuelna mašina restartuje izvršvanjem komande *sudo netplan apply*.

IP konfiguracija mrežnih adaptera nakon izmene prikazana je u Listingu 17-7.

Listing 17-7: *IP konfiguracija mrežnih adaptera nakon izmene*

```
gituser@gitsrv:/home$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe3b:8c3d prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:3b:8c:3d txqueuelen 1000  (Ethernet)
        RX packets 2447 bytes 509028 (509.0 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1495 bytes 198828 (198.8 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.0.101 netmask 255.255.255.0 broadcast 10.0.0.255
        inet6 fe80::a00:27ff:fe0e:3733 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:0e:37:33 txqueuelen 1000  (Ethernet)
        RX packets 123 bytes 42066 (42.0 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 20 bytes 1496 (1.4 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 102 bytes 7732 (7.7 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 102 bytes 7732 (7.7 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Kao što je ranije naglašeno, virtuelnim mašinama se pristupa iz okruženja *host* operativnog sistema pomoću softvera koji emuliraju *SSH* klijente. Na virtuelnoj mašini *gitsrv* nije potrebno instalirati *SSH* server, jer *Ubuntu Server 18.04 LTS* dolazi sa preinstaliranim *OpenBSD Secure Shell* serverom i *ssh daemon* se automatski pokreće nakon butovanja operativnog sistema (Listing 17-8).

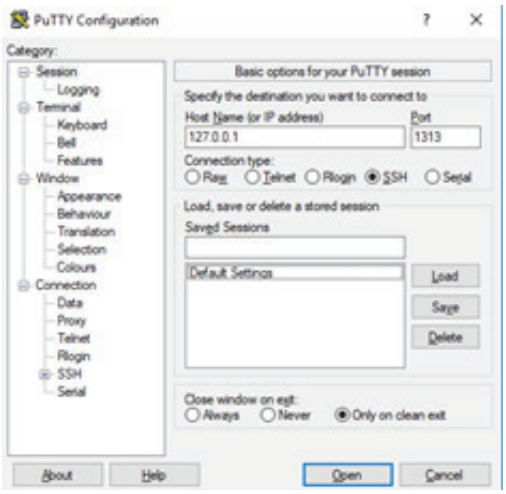
Listing 17-8: *SSH daemon*

```
gituser@gitsrv:/home$ service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2018-07-11 18:05:33 UTC; 49min ago
  Process: 836 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 899 (sshd)
    Tasks: 1 (limit: 2322)
   CGroup: /system.slice/ssh.service
           └─899 /usr/sbin/sshd -D
```

Virtuelnoj mašini *gitsrv* može da se pristupi preko *Windows host*-a korišćenjem nekog od *SSH* klijentskih programa. Za potrebe ove demonstracije korišćen je popularan alat pod nazivom *PuTTY*, koji može besplatno da se preuzme sa sledeće adrese: https://www.putty.org/.

Nakon instalacije *PuTTY* klijenta, gitsrv virtuelnoj mašini se pristupa preko *NAT* komponente virtuelne mreže, tako što se u polje *Host name* (*or IP address*) upiše *loopback* adresa fizičkog mrežnog adaptera *Windows hosta*, i u ovom slučaju port 1313 (pogledati Sliku 17-19).

Slika 17-27: *Screenshot – pristup virtuelnoj mašini gitsrv preko PuTTY klijenta*



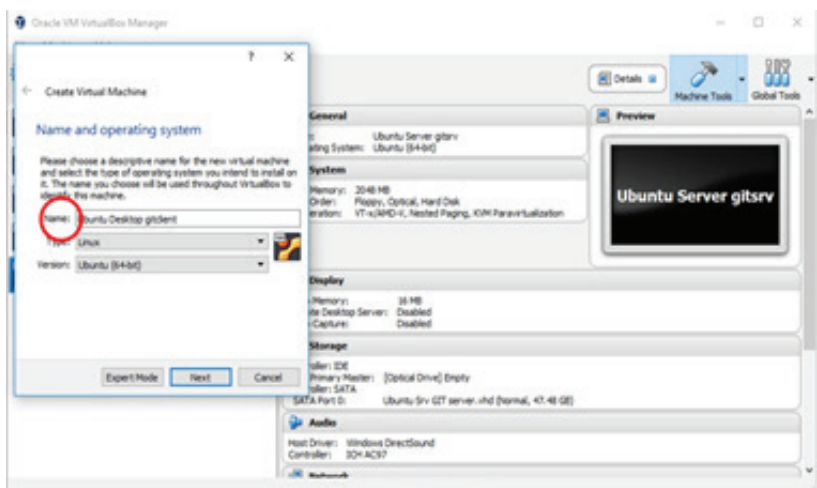
Virtuelna mašina *gitsrv* je sada spremna za implementaciju *Git* servera, koja je prikazana u Glavi 18.

17.3.2. Implementacija *gitclient* i *gitclient2* virtuelnih mašina

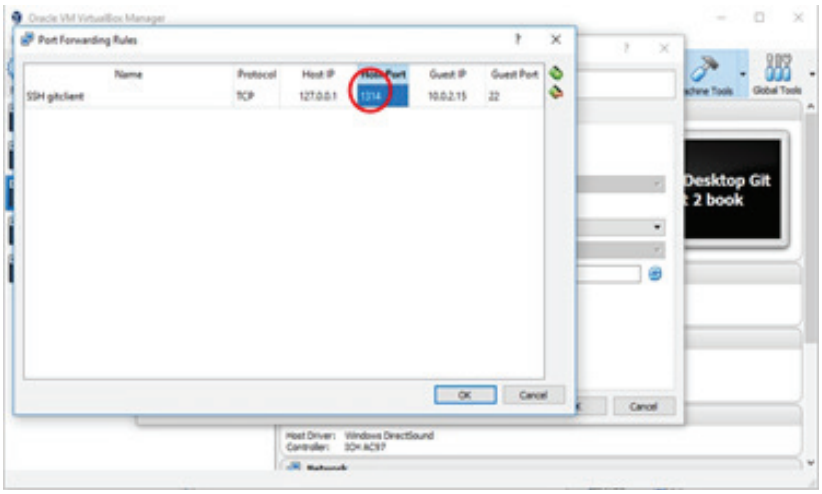
Proces implementacije *gitclient* virtuelne mašine je skoro identičan. Mašina se konfiguriše slično kao i *gitsrv* mašina, uz sledeće razlike:

- u prvom koraku *wizarda* za kreiranje nove virtuelne mašine u polje *Name* upisati drugačije opisno ime virtuelne mašine, na primer *Ubuntu Desktop gitclient* (Slika 17-28),
- u podešavanjima optičkog uređaja virtuelne mašine opcijom *Choose Virtual Optical Disk File* potrebno je označiti preuzeti *ISO* fajl sa instalacionim medijumom *Ubuntu Desktop* 18.04 *LTS* operativnog sistema na lokalnom disku i
- u podešavanjima za prosleđivanje portova prvog mrežnog adaptera tipa *NAT* potrebno je za vrednost *Host Port* izabrati port čija je vrednost različita od broja 1313 (zato što je ovaj broj porta iskorišćen za konfigurisanje mrežnog adaptera *gitsrv* mašine), na primer 1314 (Slika 17-29).

Slika 17-28: *Screenshot – kreiranje virtuelne mašine gitclient*



Slika 17-29: *Screenshot – definisanje pravila za prosleđivanje portova mašine gitclient*

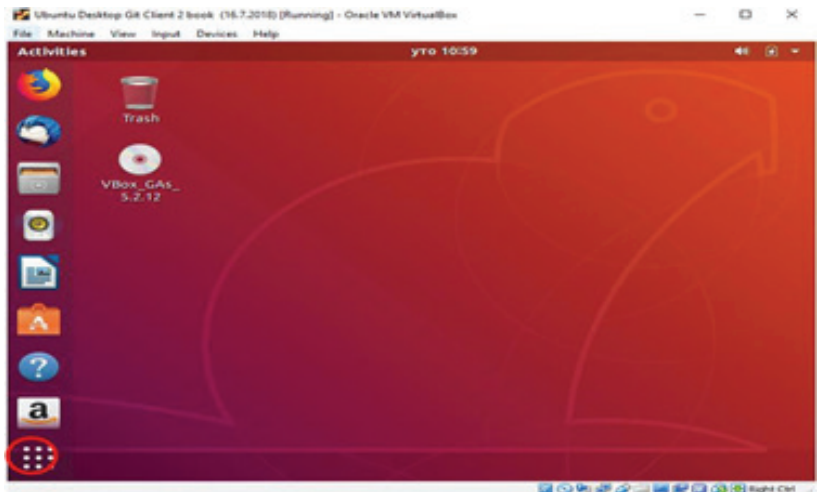


Sam proces instalacije *Ubuntu Desktop* 18.04 *LTS* operativnog sistema sličan je kao instalacija *Ubuntu* serverske verzije. U koraku instalacionog *wizarda*, gde se traži korisničko ime i lozinka, biće kreiran jedan nalog pod nazivom nebojsa, koji će kasnije biti korišćen kao jedan od klijentskih naloga u primeru sa *Git* sistemom. Takođe, tokom procesa instaliranja operativnog sistema, računaru se dodeljuje ime *gitclient*.

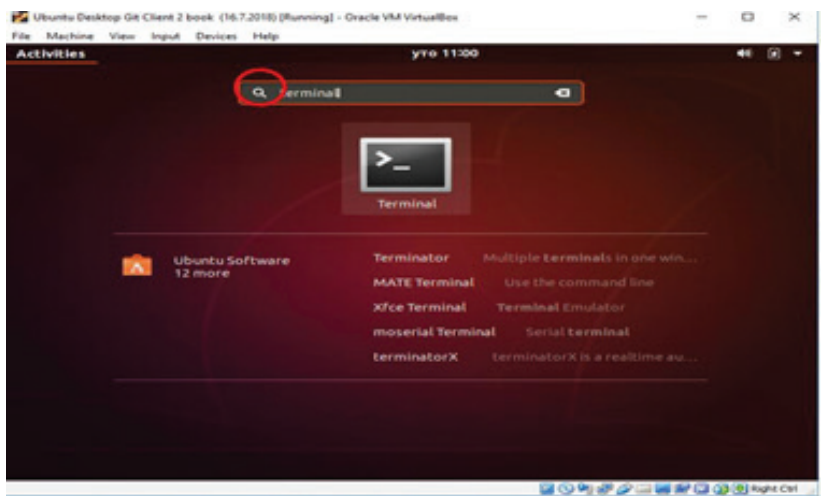
Isto kao i u slučaju *gitsrv* virtuelne mašine i u ovom slučaju, odmah nakon instalacije operativnog sistema, potrebno je instalirati *VirtualBox Guest Additions* paket alata.

Nakon prvog butovanja *gitclient* virtuelne mašine i logovanja na sistem sa kredencijalima korisnika nebojsa, potrebno je da se pokrene *Linux bash* preko grafičkog korisničkog interfejsa. Dovoljno je da se klikne na ikonicu *Show Applications* i u polje za pretragu ukucati *Terminal* i ikonica termina će da se pojavi (Slika 17-30 i Slika 17-31).

Slika 17-30: *Screenshot – Ubuntu Desktop Show Applications ikonica*



Slika 17-31: *Screenshot – Ubuntu Desktop Terminal ikonica*



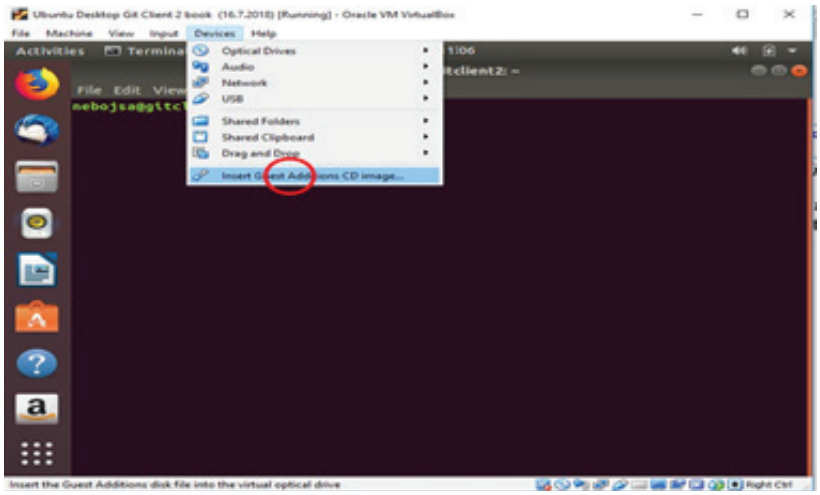
Isto kao i u prethodnom slučaju, pre same instalacije *VirutalBox Guest Additions* paketa potrebno je da se proveri da li su sve biblioteke koje su potrebne za instalaciju ovog alata implementirane. Za ove potrebe koristi se komanda *apt-get-install* sa argumentima koji su prikazani u Listingu 17-9.

Listing 17-9: *Instalacija biblioteka za VirutalBox Guest Additions*

```
nebojsa@gitclient:/$ apt-get install -y dkms build-essential linux-headers-
generic linux-headers-$(uname -r)
```

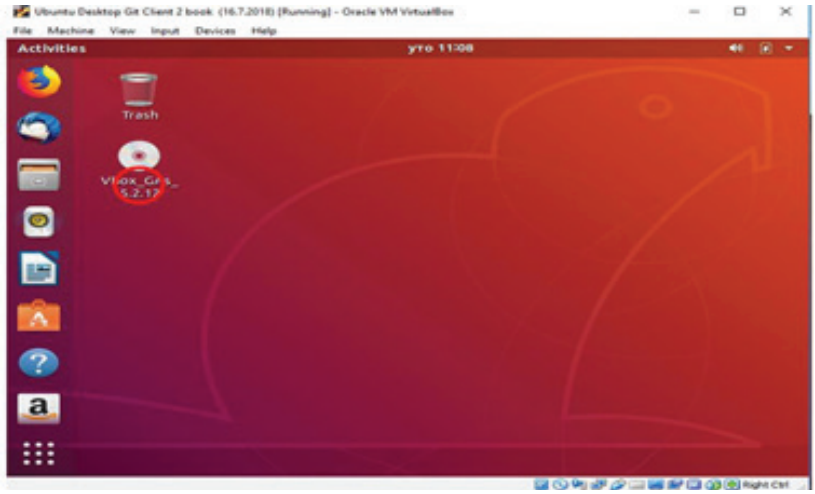
Nakon instalacije potrebnih biblioteka, potrebno je da se ubaci virtuelni instalacioni medijum *VirutalBox Guest Additions* alata izborom opcija *Devices* u *VirutalBox host* aplikaciji i *Guest Additions CD image* (Slika 17-32).

Slika 17-32: *Screenshot – VirtualBox Guest Additions instalacioni medijum*



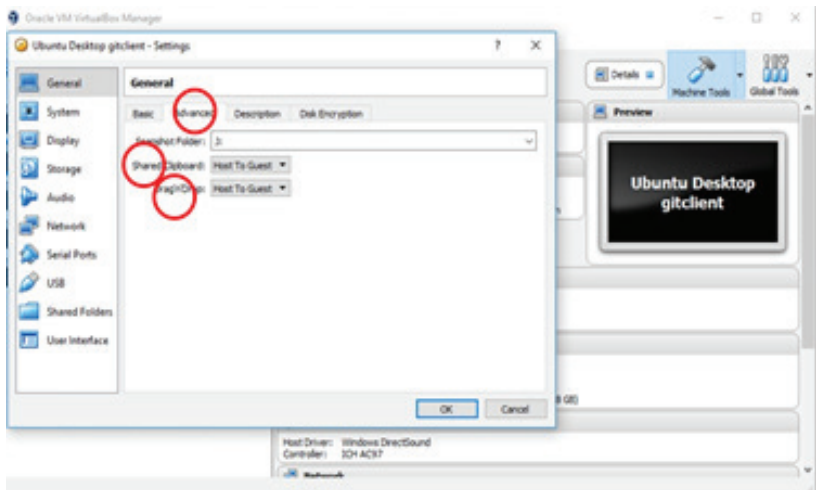
Zatim je potrebno da se klikne na *VirutalBox Guest Additions* ikonicu koja se nalazi na *Desktop*-u *Ubuntu* virtuelne mašine, kada se pokreće proces instalacije (Slika 17-33).

Slika 17-33: *Screenshot – pokretanje procesa instalacije VirtualBox Guest Additions*



Nakon završene instalacije *VirtualBox Guest Additions* skupa alatki potrebno je da se omoguće *copy paste* operacije, kao i ostale funkcionalnosti između *host* i *guest* mašine, izborom opcije za podešavanja virtuelne mašine u prozoru *Oralce VirutalBox VM Manager* pod nazivom *Advanced* (Slika 17-34).

Slika 17-34: *Screenshot – omogućavanje funkcionalnosti VirtualBox Guest Additions*



Kloniranje virtuelne mašine gitclient

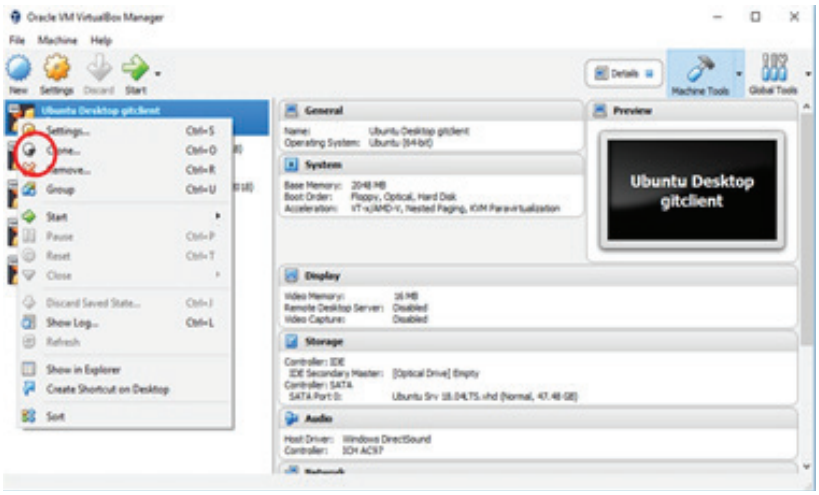
Kao što je rečeno, za potrebe simulacije *Git* server – *Git* klijent klaud okruženja, potrebna je i treća virtuelna mašina pod nazivom *gitclient2*, koja ima skoro identičnu konfiguraciju kao virtuelna mašina *gitclient*. Na sreću, ovu mašinu nije potrebno instalirati iz početka, zato što *VirtualBox* podržava funkciju kloniranja (eng. *cloning*) virtulenih mašina.

Na nivou *VirtualBox* hipervizora podržana su dva tipa kloniranja, i to potpuno (puno) kloniranje (eng. *full cloning*) i povezano kloniranje (eng. *linked cloning*). *Full* kloniranje pravi identičnu kopiju originalne virtuelne mašine uključujući i fajl virtuelnog hard diska.

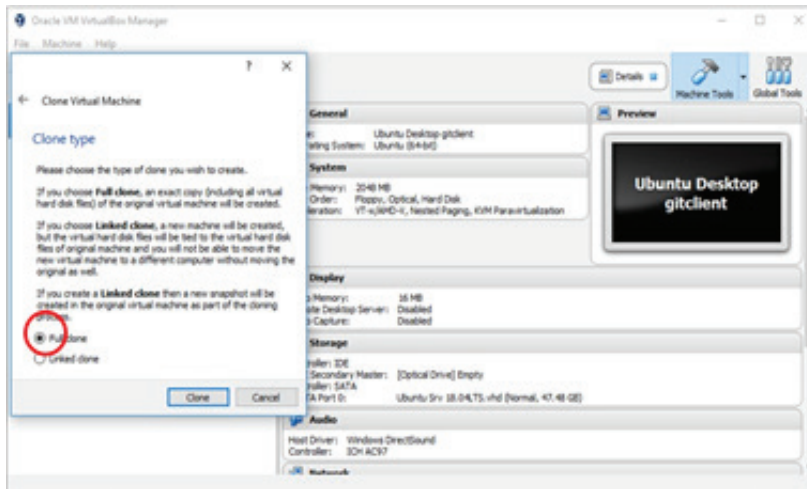
S druge strane, izvršavanjem *linked* kloniranja kreira se nova virtuelna mašina, ali je virtuelni disk nove mašine povezan sa virtuelnim diskom originalne mašine i u ovom slučaju, ako je potrebno da se klonirana virtuelna mašina premesti na drugu platformu, sa njom će morati da se premesti i originalna (izvorna) virtuelna mašina.

Za potrebe simulacije *Git* server – *Git* klijent okruženja, kreiran je *full* klon virtuelne mašine *gitclient*. Dovoljno je samo da se klikom na desno dugme miša izabere virtuelna mašina *gitclient* u *VirtualBox VM Manager* prozoru i odabere opcija clone. Nakon toga ostaje samo da se kloniranoj virtuelnoj mašini dodeli ime i izabere opcija *Full clone* (Slika 17-35 i Slika 17-36).

Slika 17-35: *Screenshot – kloniranje virtuelne mašine gitclient*

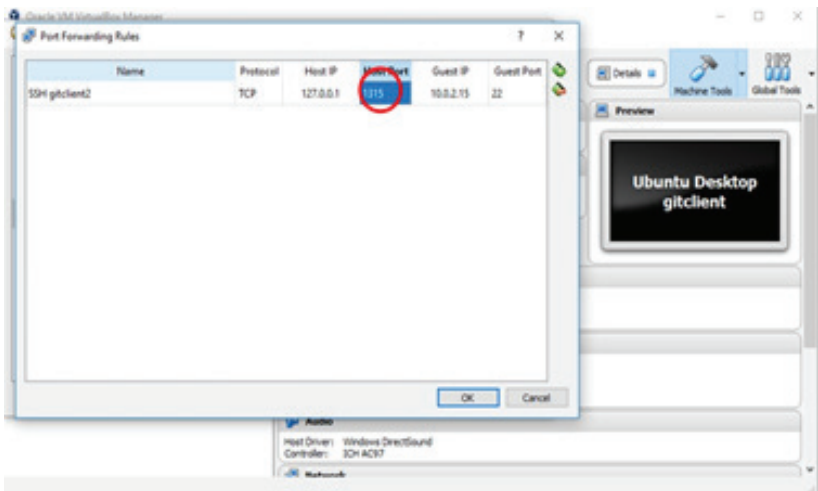


Slika 17-36: *Screenshot – kloniranje virtuelne mašine gitclient (opcija Full cone)*



Nakon završetka procesa kloniranja, koji traje neko vreme, potrebno je da se u podešavanjima za prosleđivanje portova prvog mrežnog adaptera tipa *NAT*, klonirane virtuelne mašine *gitclient2*, promeni vrednost parametra *Host Port*, da bi i originalnoj i kloniranoj mašini moglo da se pristupa u isto vreme. U primeru sa Slike 17-37, vrednost *Host Port* parametra postavljena je na 1315.

Slika 17-37: *Screenshot – konfigurisanje pravila za prosleđivanje portova na gitclient2*



Takođe, kloniranoj virtuelnoj mašini potrebno je da se promeni naziv (*hostname*) na nivou operativnog sistema, koja nakon procesa kloniranja ima isti naziv kao i prva virtuelna mašina *gitclient*.

Za promenu *hostname* računara u *Linux bash*-u koristi se naredba *hostnamectl* sa odgovarajućim parametrima. Izvršavanjem samo komande *hostnamectl* bez parametara, prikazuje se naziv računara. Navedeno je prikazano u Listingu 17-10.

Listing 17-10: *Promena naziva klonirane virtuelne mašine na nivou operativnog sistema*

```
nebojsa@gitclient:~$ sudo hostnamectl set-hostname gitclient2
[sudo] password for nebojsa:
nebojsa@gitclient:~$ hostnamectl
   Static hostname: gitclient2
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 85b0a70f7db14364a43c61d9a3504cb3
           Boot ID: caa9453092ca49d692ba89ed5263fa01
    Virtualization: oracle
  Operating System: Ubuntu 18.04 LTS
            Kernel: Linux 4.15.0-23-generic
      Architecture: x86-64
```

Da bi se proces promene imena računara do kraja završio, potrebno je da se izvrše odgovarajuće izmene i u fajlu */etc/hosts*, gde lokalna *loopback* adresa treba da pokazuje na novo ime računara (u ovom slučaju *gitclient2*). Za izmene sadržaja fajla nabolje je da se koristi *nano editor.* Modifikovani sadržaj fajla */etc/hosts* prikazan je u Listingu 17-11.

Listing 17-11: *Izmene fajla /etc/hosts*

```
127.0.0.1       localhost
127.0.1.1       gitclient2

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Nakon izvršenih operacija promene naziva, virtuelena mašina *gitclient2* se restartuje, kako bi nova podešavanja bila primenjena.

Virtuelnu mašinu *gitclient2* koristi drugi korisnik sa korisničkim imenom ivana. Potrebno je kreirati novog korisnika i dodati ga u sudo grupu, da bi korisnik mogao da izvršava naredbe sa *root* (*superuser*) privilegijama (Listing 17-12).

Listing 17-12: *Kreiranje korisnika ivana i dodavanje korisnika u sudo grupu*

```
nebojsa@gitclient2:/$ sudo adduser ivana
Adding user `ivana' ...
Adding new group `ivana' (1001) ...
Adding new user `ivana' (1001) with group `ivana' ...
Creating home directory `/home/ivana' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for ivana
Enter the new value, or press ENTER for the default
        Full Name []: ivana strumberger
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
#dodavanje korisnika ivana u sudo grupu
nebojsa@gitclient2:/$ sudo usermod -aG sudo ivana
```

Kreirani korisnik ivana je novi glavni korisnik virtuelne mašine *gitclient2*.

Konfigurisanje mrežnih adaptera virtuelnih mašina gitclient i gitclient2 na nivou operativnog sistema

Slično kao i u slučaju virtuelne mašine *gitsrv*, mašina *gitclient*, od *Oracle VirutalBox DHCP* servisa dobija *IP* podešavanja za mrežni adapter *enp0s3*, dok je za drugi mrežni adapter *enp0s8* potrebno konfigurisati statičku *IP* adresu.

Listing sa prikazom rezultata *ifconfig* naredbe na mašini *gitclient* je dat ispod.

Listing 17-13: *Izvršavanje ifconfig komande na mašini gitclient*

```
nebojsa@gitclient:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::8bb:b6e:493f:c101 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:9f:98:d6 txqueuelen 1000  (Ethernet)
        RX packets 1906 bytes 2211358 (2.2 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 583 bytes 65410 (65.4 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        ether 08:00:27:44:dc:0f txqueuelen 1000 (Ethernet)
        RX packets 4 bytes 1320 (1.3 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 126 bytes 19980 (19.9 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 187 bytes 14301 (14.3 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 187 bytes 14301 (14.3 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Da bi se mrežnom adapteru *enp0s8* virtuelne mašine *gitclient* dodelila statička *IP* adresa, potrebno je izmeniti fajl */etc/netplan/01-network-manager-all.yaml* i upisati mrežna podešavanja za adapter *enp0s8*. Isto kao i u prthodnim slučajevima, za potrebe izmene ovog fajla koristi se *nano editor* izvršavanjem sledeće komande: *sudo nano/etc/netplan/01-network-manager-all.yaml.*

Modifikovani fajl */etc/netplan/01-network-manager-all.yaml* prikazan je u Listingu 17-14.

Listing 17-14: *Izmene fajla /etc/netplan/01-network-manager-all.yaml*

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: networkd
  ethernets:
        enp0s3:
            addresses: []
            dhcp4: true
            optional: true
        enp0s8:
            dhcp4: no
            addresses: [10.0.0.102/24]
```

Kao što se vidi iz priloženog listinga, mrežni adapter *enp0s3* dobija dinamičku *IP* adresu od *DHCP* servisa, dok je za mrežni adapter *enp0s8* konfigurisana statička *IP* adresa 10.0.0.102/24. Podrazumevani mrežni prolaz, masku podmreže i adresu *DNS* servera nije potrebno konfigurisati, jer mrežni interfejs *enp0s8* služi isključivo za komunikaciju između virtuelnih mašina, a u ovom slučaju da bi mašina *gitclient* komunicirala sa mašinama *gitsrv* i *gitclient2*.

Nakon izmene sadržaja fajla */etc/netplan/01-network-manager-all.yaml*, potrebno je primeniti izmene mrežnih podešavanja bez potrebe da se mašina *gitclient* restartuje izvršavanjem jednostavne naredbe *sudo netplan apply*.

IP konfiguracija mrežnih adaptera virtuelne mašine *gitclient* nakon svih prikazanih podešavanja izgleda kao što je dato u Listingu 17-15.

Listing 17-15: *IP konfiguracija mrežnih adaptera mašine gitclient nakon izmena*

```
nebojsa@gitclient:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::8bb:b6e:493f:c101 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:9f:98:d6 txqueuelen 1000  (Ethernet)
        RX packets 1940 bytes 2216508 (2.2 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 663 bytes 74731 (74.7 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.0.102 netmask 255.255.255.0 broadcast 10.0.0.255
        ether 08:00:27:44:dc:0f txqueuelen 1000  (Ethernet)
        RX packets 4 bytes 1320 (1.3 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 274 bytes 43179 (43.1 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 427 bytes 30526 (30.5 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 427 bytes 30526 (30.5 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Dakle, drugom mrežnom adapteru *enp0s8*, koji je povezan na internu mrežu, u slučaju virtuelne mašine *gitsrv* dodeljena je statička *IP* adresa 10.0.0.101/24, dok je virtuelnoj mašini *gitclient* dodeljena *IP* adresa 10.0.0.102/24. S obzirom da obe mašine pripadaju istoj internoj mreži, kao i da pripadaju istom logičkom mrežnom opsegu, mašine se međusobno „vide" i mogu da komuniciraju, što može da se proveri ping komandom.

Izvršavanje *ping* komande iz terminala *gitclient* mašine prikazano je u Listingu 17-16.

Listing 17-16: *Rezultat izvršavanja ping komande*

```
nebojsa@gitclient:~$ ping 10.0.0.101
PING 10.0.0.101 (10.0.0.101) 56(84) bytes of data.
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=0.242 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=0.704 ms
64 bytes from 10.0.0.101: icmp_seq=3 ttl=64 time=0.685 ms
64 bytes from 10.0.0.101: icmp_seq=4 ttl=64 time=0.691 ms
64 bytes from 10.0.0.101: icmp_seq=5 ttl=64 time=0.688 ms
64 bytes from 10.0.0.101: icmp_seq=6 ttl=64 time=0.703 ms
64 bytes from 10.0.0.101: icmp_seq=7 ttl=64 time=0.680 ms

--- 10.0.0.101 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6127ms
rtt min/avg/max/mdev = 0.242/0.627/0.704/0.159 ms
```

Iz priloženog listinga jasno se vidi da je mašina *gitsrv*, čija je adresa jednog mrežnog adaptera, 10.0.0.101, vidiljiva sa mašine *gitclient* koja ima adresu 10.0.0.102.

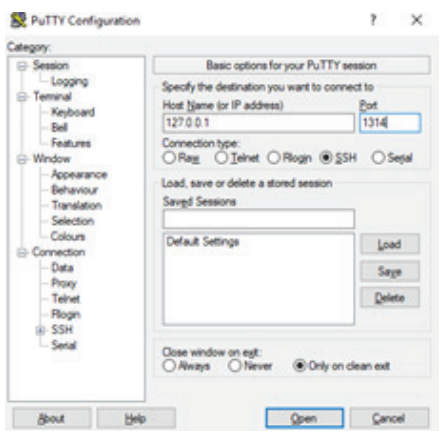
Za razliku od mašine *gitsrv*, operativni sistem mašine *gitclient* (*Ubuntu Desktop* 18.04 *LTS*) se ne isporučuje sa preinstaliranim *SSH* serverom. Zbog toga, da bi se omogućio pristup *gitclient* mašini direktno sa *host* računara, prvo mora da se instalira *SSH* server. Nakon instaliranja *SSH* servera potrebno je proveriti da li je *ssh daemon* aktivan. Obe naredbe su prikazane u Listingu 17-17.

Listing 17-17: *Instaliranje SSH servera*

```
nebojsa@gitclient:~$ sudo apt-get install ssh
nebojsa@gitclient:~$ service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: ena
   Active: active (running) since Thu 2018-07-12 07:49:49 CEST; 47s ago
 Main PID: 4264 (sshd)
    Tasks: 1 (limit: 3558)
   CGroup: /system.slice/ssh.service
           └─4264 /usr/sbin/sshd -D
jul 12 07:49:49 gitclient systemd[1]: Starting OpenBSD Secure Shell server...
jul 12 07:49:49 gitclient sshd[4264]: Server listening on 0.0.0.0 port 22.
jul 12 07:49:49 gitclient sshd[4264]: Server listening on :: port 22.
jul 12 07:49:49 gitclient systemd[1]: Started OpenBSD Secure Shell server.
```

Po završetku instalacije *SSH* servera, virtuelnoj mašini *gitclient* može da se pristupi preko *Windows hosta* i *PuTTY SSH* klijenta. U grafičkom interfejsu *PuTTY* klijenta u polje *Host Name* (*or IP address*) upisuje se *loopback* adresa *Windows hosta*, a u polje *Port* se u ovom slučaju upisuje vrednost 1314, kao što je prikazano na Slici 17-8.

Slika 17-38: *Screenshot – pristup mašini gitclient preko PuTTY klijenta*



Sledeći isti princip, potrebno je da se izmene i mrežna podešavanja za virtuelnu mašinu *gitclinet2*, samo što se u ovom slučaju za statičku *IP* adresu mrežnog adaptera *enp0s8* konfiguriše *IP* adresa 10.0.0.103/24.

Rezultat izvršavanja komande *ifconfig*, nakon izmene *IP* podešavanja mašine *gitclient2* prikazan je u Listingu 17-18.

Listing 17-18: *Rezultat izvršavanja ifconfig naredbe (gitclient2)*

```
ivana@gitclient2:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe8e:9804 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:8e:98:04 txqueuelen 1000  (Ethernet)
        RX packets 2024 bytes 651723 (651.7 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1136 bytes 148134 (148.1 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.103 netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::a00:27ff:fe3e:89a3 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:3e:89:a3 txqueuelen 1000  (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 84 bytes 9721 (9.7 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 120 bytes 9021 (9.0 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 120 bytes 9021 (9.0 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

18. IMPLEMENTACIJA GIT SISTEMA U VIRTUELNOM OKRUŽENJU

U ovoj glavi prikazana je implementacija *Git*-a u virtuelnom okruženju, sa ciljem da se izvrši simulacija implementacije *Git* servisa u okruženju klaud provajdera.

> *Git* je alat koji pripada grupi softvera za kontrolu verzije. *Git* prati promene na računarskim fajlovima u okruženju gde više ljudi koordinirano radi sa istim skupom fajlova. *Git* svoju glavnu primenu nalazi u razvoju softvera, za potrebe upravljanja različitim verzijama softverskog koda.

Kroz praktičan primer koji je dat u ovoj glavi udžbenika, implementirane su jedna Git serverska, i dve Git klijentske komponente u virtuelnom okruženju, čija je implementacija prikazana u Glavi 17. *Git* server je implementiran u formi *hosted* rešenja, koja se često koriste na klaudu. Glavna razlika između simuliranog *Git* klaud okruženja, koje je prikazano u ovom glavi, i realnog klaud okruženja je ta što klaud provajderi najčešće koriste *bare metal* hipervizore, dok je u ovom slučaju *Git* server *hostovan* na hipervizoru tipa 2 (pogledati Glavu 17).

S obzirom na to da je ovaj udžbenik, između ostalog, namenjen i studentima studijskog programa *Softversko inženjerstvo Univerziteta Singidunum*, kroz praktične simulacije prikazane su i osnovne *Git* komande.

18.1. UVOD U GIT I OSNOVNA TERMINOLOGIJA

Git je distribuirani sistem za kontrolu verzije fajlova i upravljanje izvornim kodom sa naglaskom na brzini.

Prvu verziju *Git*-a razvio je *Linus Torvalds* 2005. godine za potrebe razvoja svog *Linux* jezgra.

Git pripada grupi softvera otvorenog koda (eng. *open source*) koji može da se koristi pod uslovima *GNU* (*GNU Not Unix*) *General Public License v*2 i *v*2.1 (*GPL v*2, *GPL v*2.1).

18.1.1. Sistemi za kontrolu verzije

Sistemi za kontrolu verzije (eng. *Version Control System – VCS*) su sistemi koji pamte promene koji korisnici izvršavaju nad fajlom, ili skupom fajlova i mogućavaju da se fajlovi kasnije vrate (povrate) na prethodne verzije.

Ovakvi sistemi mogu da se primenjuju na sve tipove fajlova (*Word, Excel, PowerPoint* dokumenti, grafički, audio i video fajlovi, itd.), a najčešće se koriste za fajlove koji sadrže izvorne softverske kodove.

VCS je softver koji unapređuje timski rad programera na razvoju softvera i omogućava bolji pregled istorije urađenog posla. Kao neke od važnijih funkcija *VCS*-a navode se sledeće:

- omogućava simultani rad većeg broja programera i *developer*-a,
- u simultanom radu većeg broja korisnika sistema onemogućava situaciju da jedan korisnik pregazi promene drugog korisnika,
- čuva istoriju svake verzije fajla i
- može da se primeni na bilo koji tip fajlova.

Kao tri osnovne vrste *VCS*-a navode se:

- lokalni sistemi za kontrolu verzije (eng. *Local Version Control System – LVCS*),
- dentralizovani sistemi za kontrolu verzije (eng. *Centralized Version Control System – CVCS*) i
- distribuirani sistemi za kontrolu verzije (eng. *Distributed Version Control System – DVCS*).

U toku svakodnevnog rada sa računarima često se javlja potreba za čuvanjem različitih verzija istog fajla, odnosno fajlova. Jedna od najčešće korišćenih metoda, koju koristi većina prosečnih korisnika računara, jeste kopiranje fajlova iz jednog u drugi folder uz eventualno dodavanje datuma promena nazivima foldera. Nažalost, ovakav pristup je dosta podložan greškama.

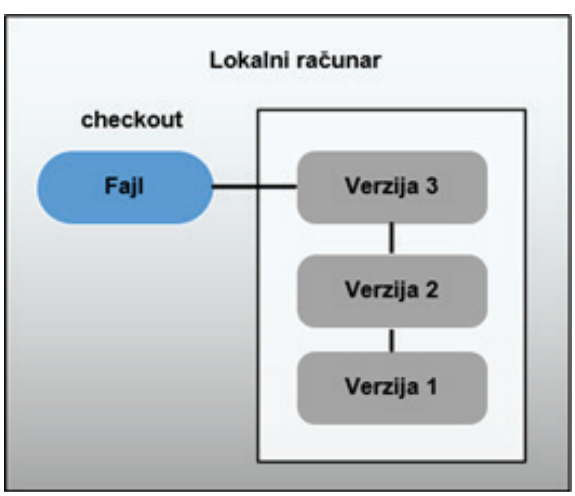
Sa ciljem unapređenja mehanizama za čuvanje različitih verzija istog fajla (fajlova), razvijeni su *LVCS*. Ovi sistemi imaju jedinstvenu bazu podataka koja čuva informacije o promenama koje su izvršene nad određenim fajlovima.

Arhitektura *LVCS* softvera prikazana je na Slici 18-1.

Jedan od najšire korišćenih *LVCS* je bio sistem za kontrolu revizije (eng. *Revision Control System – RCS*), koji čuva skup zakrpi (eng. *patch set*), tj. razlike između pojedinih verzija fajlova u posebnom formatu na disku računara. Struktura fajla u bilo kom vremenskom trenutku može da se rekonstruiše prolaskom kroz sve zakrpe.

Kao još jedan primer *LVCS* alata navodi se sistem za kontrolu izvornog koda (eng. *Source Code Control System - SCCS*).

Slika 18-1: *Arhitektura LVCS softvera*

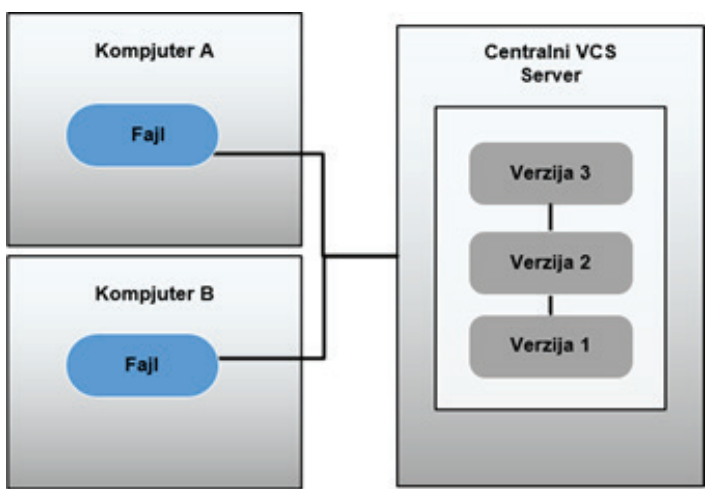


Upotreba *LVCS* softvera je danas mahom zamenjena *CVCS* alatima, zato što *LVCS* nisu pogodni u situacijama kada postoji potreba za kolaboracijom (saradnjom) između programera (korisnika) koji rade na različitim sistemima.

CVCS imaju centralni server koji čuva različite verzije fajlova, gde veliki broj klijenata preuzima fajlove sa ove centralne lokacije. Na lokalnim mašinama (klijentima) čuvaju se samo tekuće verzije fajlova (verzije sa kojima se trenutno radi).

CVCS su dugi niz godina predstavljali standard u korišćenju sistema za kontrolu verzije. Arhitektura ovih sistema je prikazana na Slici 18-2.

Slika 18-2: *Arhitektura CVCS softvera*



Prednosti *CVCS* u odnosu na *LVCS* su očigledne. Osim toga što *CVCS* omogućavaju kolaboraciju većeg broja učesnika na istom projektu, administriranje ovakvih sistema je olakšano, zato što se administrira samo jedna centralna baza, za razliku od *LVCS* softvera, gde se administriraju sve lokalne baze klijenata.

Međutim, jedan od najvećih nedostataka *CVCS* softvera je jedna tačka pada sistema, što je u ovom slučaju centralni server na kome se nalazi jedinstvena baza podataka. Tako na primer, ako je centralni server u kvaru određeno vreme, u ovom vremenskom periodu niko od korisnika neće moći da koristi sistem.

U još gorem scenariju, ako se podaci u centralnoj bazi izgube, gube se skoro svi podaci – ostaju samo oni podaci na kojima korisnici trenutno rade i koji se nalaze u lokalnim bazama klijenata.

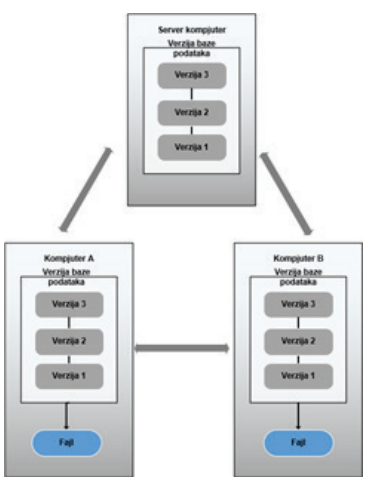
DVCS softveri ispravljaju navedene nedostatke *LVCS* sistema.

> Tokom korišćenja *DVCS* sistema, korisnici ne preuzimaju samo tekuće verzije fajlova, već preuzimaju ceo repozitorijum (istoriju verzija svih fajlova od trenutka konfigurisanja i puštanja sistema u rad).

Kod ovakvih sistema ne postoji jedinstvena tačka kvara. U slučaju da neki od udaljenih servera, preko koga su se klijenti povezali, prestane sa radom, podaci mogu da se iskopiraju na rezervni udaljeni server sa bilo kog klijentskog računara, zato što svaki klijent na svom lokalnom repozitorijumu ima sve verzije fajlova, kako istorijske, tako i tekuće (aktuelne).

Arhitektura *DVCS* alata prikazana je na Slici 18-3.

Slika 18-3: *Arhitektura DVCS softvera*



DVCS softveri takođe omogućavaju da se klijent u isto vreme poveže sa više udaljenih repozitorijuma i da na taj način sarađuje sa različitim grupama ljudi koji rade na istom projektu. Ovi sistemi omogućavaju podešavanje nekoliko tokova rada, što ne podržavaju *CVCS* sistemi.

Kao neki od najpoznatijih predstavnika *DVCS* sistema navode se *Git, Mercurial, Bazaar* i *Darcs*.

18.1.2. Kratka istorija *Git*-a

Razvoj *Git*-a se dovodi u vezu sa Linux operativnim sistemom. Tokom 90-tih i početkom 2000-te godine promene u *Linux* jezgru distribuirane su zajednici (eng. *community*) kao zakrpe i arhivirani fajlovi. Zbog ovoga se javila potreba za kontrolu verzija *Linux* jezgra i projekat *Linux* jezgra počeo je da koristi vlasnički *DVCS* pod nazivom *BitKeeper*.

Međutim, 2005. godine odnos između komercijalne organizacije koja je razvila *Bitkeeper* i zajednice, koja je radila na razvoju *Linux* jezgra, se pooštrio i *Bitkeeper* više nije bio besplatan. Zbog ovoga je *Linux* zajednica bila primorana da počne sa razvojem sopstvenog alata za kontrolu verzije, koji je kasnije dobio naziv *Git*. Član *Linux* zajednice koji se posebno istakao u razvoju ovog alata bio je *Linus Torvalds*, osnivač *Linux*-a.

Linux zajednica je tokom korišćenja *BitKeeper*-a stekla dovoljno iskustva o tome šta dobar sistem za kontrolu verzije treba da ima, što je zajednici omogućilo da preciznije definiše mogućnosti koje bi novi sistem trebalo da podržava.

Među mogućnostima, izdvajale su se sledeće:

- brzina,
- robusna podrška za nelinearan razvoj softera sa velikim brojem paralelnih grana,
- jednostavan dizajn,
- distribucija i
- sposobnost za efikasno upravljanje velikim projektima, poput *Linux* jezgra, koja se ogledala pre svega u brzini i mogućnosti za manipulacijom velikom količinom podataka.

Od 7. aprila 2005. godine, kada je puštena prva verzija *Git*-a, ovaj alat se razvio i sazreo u formi jednostavnog i brzog sistema koji je efikasan i pogodan za upravljanje velikim projektima, uz odličnu podršku sistema grananja za nelinearan razvoj softvera.

18.1.3. Prednosti *Git*-a

Kao neke od prednosti *Git*-a navode se sledeće: besplatan i softver otvorenog koda, brz i mali, implicitni *backup*, sigurnost, performanse hardvera i olakšano grananje.

Besplatan i softver otvorenog koda

Git se koristi pod *GPL* licencom i besplatno može da se preuzme sa Interneta. S obzirom da je u pitanju softver otvorenog koda, svaki korisnik ima mogućnost da kod modifikuje u skladu sa svojim potrebama.

Brz i mali

Git skoro svaku operaciju obavlja lokalno i zbog toga ima veliku prednost u odnosu na slične sisteme u kontekstu brzine izvršavanja. Ova brzina proističe iz činjenice da se *Git* ne oslanja na centralni server i da ne vrši interakciju sa centralnim serverom za svaku operaciju.

Jezgro *Git*-a je napisano u programskom jeziku *C*, a poznato je da jezik *C* ne koristi posredničku komponentu, poput virtuelnih mašina (kao na primer programski jezik *Java*, koji koristi *Java* virtuelnu mašinu), između hardvera i izvršnog okruženja. Bez obzira što *Git* kopira ceo centralni repozitorijum na klijentsku stranu, korišćenjem efikasnih algoritama za kompresiju i skladištenje podataka, podaci na strani klijenta su relativno male veličine.

Implicitni backup

Git izvršava implicitni *backup* podataka tako što se podaci kopiraju na svakog klijenta. Podaci na strani klijenta mogu da se koriste u slučaju prekida rada centralnog servera i samim tim verovatnoća za gubitak podataka se svodi na minimum.

Sigurnost

Git koristi standardnu kriptografsku heš funkciju *SHA*-1 (*Secure Hash Algorithm*) za imenovanje i identifikaciju objekata u bazi podataka. Za svaki fajl i objekat izvršenje (eng. *commit*) izračunava se kontrolna suma (eng. *checksum*).

Performanse hardvera

U slučaju *CVCS* sistema, centralni server mora da ima dovoljno hardverske snage da bi zadovoljio potrebe celog tima koji radi na razvoju softverskog projekta. Za male timove koji rade na istom projektu, hardver ne predstavlja ograničenje, međutim za veće timove hardverska ograničenja centralnog servera često predstavljaju usko grlo. U slučaju *DVCS* alata, klijenti interaguju sa centralnim

serverom samo kada snimaju (eng. *push*) ili preuzimaju (eng. *pull*) promene, i zbog toga ne postoji potreba za moćnim hardverom centralnog servera.

Olakšano grananje

CVCS alati koriste zastareo (eng. *deprecated*) mehanizam kopiranja fajlova. Kada se kreira nova grana (tok razvoja), kopira se ceo kod u tu novu grani, što predstavlja neefikasno rešenje koje troši mnogo vremena. Takođe brisanje i spajanje grana takođe troši mnogo resursa i zahteva dugo vreme za izvršavanje. Upravljanje tokovima razvoja u *Git* okruženju je jednostavno, gde je potrebno svega nekoliko sekundi za kreiranje, brisanje i spajanje grana.

18.1.4. Osnovna *Git* terminologija

U ovoj sekciji prikazana je osnovna *Git* terminologija, da bi čitaoci mogli lakše da prate deo koji opisuje implementaciju *Git* serverske i *Git* klijentskih komponenti.

Kao napomena navodi se da ova terminologija važi i za većinu drugih *DVCS* alata.

Lokalni repozitorijum

Svaki *VCS* alat obezbeđuje privatni radni prostor (eng. *workspace*) u formi radne kopije (eng. *working copy*). Svaki učesnik projekta (klijent) može da napravi promenu u svom privatnom radnom prostoru, koja nakon izvršavanja postaje deo repozitorijuma kome svi klijenti imaju pristup.

Git ide jedan korak dalje tako što svakom klijentu obezbeđuje privatnu kopiju celog repozitorijuma. Korisnici mogu da izvršavaju brojne operacije sa ovim repozitorijumom, kao što su dodavanje novog fajla, brisanje fajla, promena naziva i premeštanje fajla, izvršavanje promena, itd.

Radni direktorijum i fazna oblast

Radni direktorijum (eng. *working directory*) je virtuelni prostor gde se vrši provera i modifikovanje fajlova. U većini *VCS* sistema korisnici obično vrše modifikaciju fajlova i izvršavanje promena direktno u repozitorijum.

Git koristi drugačiju strategiju. *Git* ne prati svaki modifikovani fajl. Kada korisnik pokrene operaciju izvršavanja *commit*, samo oni fajlovi koji se nalaze u faznoj oblasti (eng. *staging area*) uzimaju se u obzir za izvršavanje (ne svi izmenjeni fajlovi).

Operacijom *commit* fajlovi se premeštaju iz fazne oblasti i spremaju se za trajno čuvanje na udaljenom repozitorijumu.

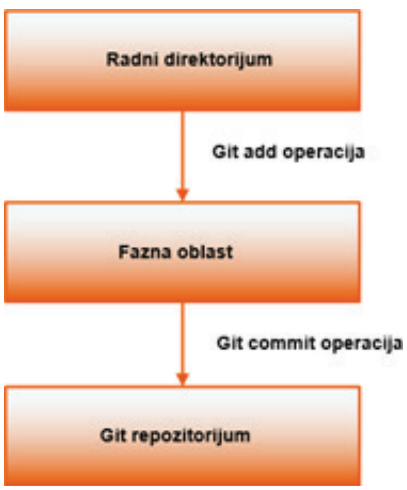
Operacija commit sve promene iz radnog direktorijuma (oblasti) čuva u lokalnom repozitorijumu i time se na neki način korisnik obavezuje da će promene da upiše i na udaljeni repozitorijum.

Proces rada (eng. *workflow*) *Git*-a može da se podeli u tri koraka (faze):

- korak 1: korisnik modifikuje fajlove u svom radnom direktorijumu,
- korak 2: korisnik dodaje modifikovane fajlove u faznu oblast (*Git* operacija *add*) i
- korak 3: korisnik izvršava operaciju *commit*, kojom se fajlovi premeštaju iz fazne oblasti i spremaju se za skladištenje na udaljenom repozitorijumu. Nakon toga korisnik izvršava operaciju *push* i promene se trajno skladište na *Git* repozitorijum.

Opisani koraci prikazani su na Slici 18-4.

Slika 18-4: *Git proces rada*



Radi boljeg razumevanja *Git* procesa rada može da posluži sledeći primer. Neka je korisnik modifikovao dva fajla sa nazivima *Test.java* i *Test1.java* i neka korisnik hoće da sačuva modifikovane fajlove u lokalni *Git* repozitorijum.

Kako bi izvršio navedene operacije, korisnik prvo dodaje fajl *Test.java* u faznu oblast izvršavanjem operacije *add*. Nakon toga, korisnik izvršava operaciju *commit* sa argumentom *-m*, gde navodi opis izvršene promene. Korisnik isti postupak ponavlja i za drugi fajl.

Primer navedenih naredbi u *GNU/Linux bash* okruženju dati su u Listingu 18-1.

Listing 18-1: *Primeri add i commit operacija*

```
# dodavanje fajla Test.java u faznu oblast
[bash]$ git add Test.java
# izvršavanje operacije commit
[bash]$ git commit -m "dodat fajl Test.java"
# dodavanje fajla Test1.java u faznu oblast
[bash]$ git add Test1.java
# izvršavanje operacije commit
[bash]$ git commit -m "dodat fajl Test1.java"
```

Veliki binarni objekat

U *Git* sistemu, svaka verzija fajla se predstavlja pomoću strukture velikog binarnog objekta (eng. *Binary Large Object – BLOB*). *BLOB* čuva samo podatke o fajlu (ne čuva metapodatke). Kao što sam naziv kaže, *BLOB* je binarni fajl, a u *Git* bazi podataka čuva se kao *SHA*-1 heš vrednost. *Git* adresiranje objekata vrši na osnovu sadržaja, bez korišćenja imena.

Stabla

Git stablo (eng. *Git tree*) je objekat koji predstavlja direktorijum. U objektu tipa stablo čuvaju se poddirektorijumi i *BLOB* objekti. Stablo je binarni fajl koji čuva reference (pokazivače) ka *BLOB* objektima i drugim stablima. Za imenovanje stabala, se kao i u slučaju *BLOB*-a, koriste *SHA*-1 vrednosti.

Commit objekti

Commit objekti čuvaju tekuće stanje lokalnog repozitorijuma. Imenovanje *commit* objekata takođe se vrši na osnovu *SHA*-1 heš vrednosti.

Objekat *commit* može da se posmatra kao čvor povezane liste. Svaki *commit* objekat ima pokazivač ka roditeljskom *commit* objektu. Na ovaj način može da se vidi cela istorija *commit* objekata, tako što se prate pokazivači ka roditeljskim *commit* objektima svakog objekta.

Ako jedan *commit* objekat ima više roditeljskih objekata, onda može da se zaključi da je taj *commit* objekat nastao spajanjem drugih grana.

Grane

Grane (eng. *branches*) se koriste u cilju kreiranja drugog toka razvoja softvera. *Git* po podrazumevanim podešavanjima koristi master granu. Tokom razvoja softvera grane se obično kreiraju kada se radi na novoj funkcionalnosti (eng. *features*) softvera. Kada se funkcionalnost završi, grana na kojoj je funkcionalnost razvijana spaja se sa master granom, a zatim se briše.

Svaka grana se referiše pomoću *HEAD* pokazivača, koji pokazuje na poslednji *commit* objekat koji je kreiran u datoj grani. Svaki put kada se kreira novi *commit* objekat, *HEAD* pokazivač se ažurira.

Tagovi

Tagovi (eng. *tags*) se prvenstveno koriste za davanje smislenih imena određenoj verziji u repozitorijumu. Tagovi su slični granama, s razlikom da su tagovi nezamenljivi. Drugim rečima, tag je grana koju niko ne može da modifikuje. Kada se za određeni *commit* objekat tag jednom kreira, on se nikada ne ažurira, čak ni u slučaju kreiranja novog *commit* objekta. *Developeri* softvera tagove obično koriste za potrebe razvoja različitih distribucija svojih proizvoda.

Operacija Clone

Operacija *clone* omogućava kreiranje nove instance lokalnog repozitorijuma. Operacijom *clone* se vrši preslikavanje udaljenog repozitorijuma na lokalnu kopiju (lokalni repozitorijum). Korisnici sa lokalnim repozitorijumom mogu da rade šta hoće, dok im pri tom nije potrebna konekcija ka *Git* serveru. Samo u slučaju kada se lokalne instance repozitorijuma sinhronizuju sa udaljenim repozitorijumom, konekcija sa *Git* serverom je potrebna.

Operacija Pull

Operacijom *pull* se vrši kopiranje promena sa instance udaljenog repozitorijuma u lokalni repozitorijum. Ova operacija se koristi za sinhronizaciju dve instance repozitorijuma.

Operacija Push

Operacija *push* se koristi za kopiranje promena sa instance lokalnog repozitorijuma na udaljeni repozitorijum. Ovom operacijom se izmenjene verzije fajlova trajno skladište na centralni *Git* repozitorijum.

HEAD

HEAD je pokazivač koji uvek pokazuje na poslednji *commit* objekat u grani. Svaki put kada korisnik izvrši operaciju *commit*, vrednost *HEAD* pokazivača se ažurira. U *GNU/Linux* okruženju, *HEAD* pokazivači grana se čuvaju u *.git/refs/heads* direktorijumu.

Primer čitanja vrednosti *HEAD* pokazivača u *Linux bash* okruženju dat je u Listingu 18-2.

Listing 18-2: *Primer čitanja vrednosti HEAD pokazivača*

```
[CentOS]$ ls -1 .git/refs/heads/
master
[CentOS]$ cat .git/refs/heads/master
570837e7d58fa4bccd86cb575d884502188b0c49
```

Revizija

Revizija označava reviziju izvornog koda. Revizije se u *Git*-u predstavljaju pomoću *commit* objekata koji se identifikuju pomoću *SHA*-1 heš vrednosti.

URL

URL objekat čuva lokaciju *Git* repozitorijuma. *Git URL* se čuva u *./git/config* fajlu. Primer čitanja *config* fajla u *Linux bash* okruženju prikazan je u Listingu 18-3.

Listing 18-3: *Primer čitanja config fajla*

```
nebojsa@gitclient:~/project$ pwd
/home/nebojsa/project
nebojsa@gitclient:~/project$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = gituser@10.0.0.101:project.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
        remote = origin
        merge = refs/heads/master
        rebase = true
```

18.2. IMPLEMENTACIJA GIT OKRUŽENJA

U ovom poglavlju prikazana je praktična implementacija *Git*-a u *Oracle VirtualBox* okruženju pod *GNU/Linux* operativnim sistemom.

Virtuelna platforma za implementaciju *Git*-a konfigurisana je u prethodnom primeru. Za više informacija pogledati Glavu 17.

Za ovu praktičnu simulaciju korišćene su tri virtuelne mašine, i to jedna na kojoj je implementiran *Git* server (mašina sa nazivom gitsrv), na kome se nalazi udaljeni *Git* repozitorijum i druge dve koje simuliraju *Git* klijente (mašine sa nazivima *gitclient* i *gitclient2*), na kojima rade dva korisnika – nebojsa i ivana.

U ovoj simulaciji može da se smatra da se *Git* server izvršava na klaudu, bez obzira što se u ovom slučaju koristi lokalno virtuelno okruženje koje je implementirano na *Windows host* računaru.

Osnovne *Git* operacije prikazane su u okviru implementiranog *Git* server - *Git* klijent okruženja simulacijom realnih scenarija, što je dato u Poglavlju 18.3.

Specifikacije serverske (mašina gitsrv) i klijentskih (mašine *gitclient* i *gitclient2*) *Git* virtuelnih mašina date su u tabelama 18-2, 18-3 i 18-3, respektivno.

Tabela 18-1: *Specifikacije Git server mašine*

Parametar	Vrednost
host name:	*gitsrv*
operativni sistem:	*Ubuntu Server* 18.04 *LTS*
tip mrežnog adaptera 1:	*NAT*
IP adresa mrežnog adaptera 1:	10.0.2.15/24
tip mrežnog adaptera 2:	*Internal*
IP adresa mrežnog adaptera 2:	10.0.0.101/24
korisnički nalog:	*gituser*
SSH server:	DA
SSH pristup:	*IP*:port – 127.0.0.1:1313

Tabela 18-2: *Specifikacije prvog Git klijenta*

Parametar	Vrednost
host name:	*gitclient*
operativni sistem:	*Ubuntu Desktop* 18.04 *LTS*
tip mrežnog adaptera 1:	*NAT*
IP adresa mrežnog adaptera 1:	10.0.2.15/24
tip mrežnog adaptera 2:	*Internal*
IP adresa mrežnog adaptera 2:	10.0.0.102/24
korisnički nalog:	*nebojsa*
SSH server:	DA
SSH pristup:	*IP:*port – 127.0.0.1:1314

Tabela 18-3: *Specifikacije drugog Git klijenta*

Parametar	Vrednost
host name:	*gitclient2*
operativni sistem:	*Ubuntu Desktop* 18.04 *LTS*
tip mrežnog adaptera 1:	*NAT*
IP adresa mrežnog adaptera 1:	10.0.2.15/24
tip mrežnog adaptera 2:	*Internal*
IP adresa mrežnog adaptera 2:	10.0.0.103/24
korisnički nalog:	*ivana*
SSH server:	DA
SSH pristup:	*IP:*port – 127.0.0.1:1315

18.2.1. Instaliranje *Git* servera

Nakon logovanja preko *PuTTY SSH* klijenta na virtuelnu mašinu *gitsrv* prvo je potrebno da se instalira *git* paket. Za instaliranje *git* paketa na *Ubuntu* operativnom sistemu koristi se komanda *apt-get* uz navođenje naziva paketa, koji je u ovom slučaju *git-core*.

Izvršavanjem ove komande, najnovija verzija *Git*-a sa preuzima sa udaljenog *Linux* repozitorijuma softvera i instalira se. Nakon završetka procesa instaliranja, proverava se *Git* verzija izvršavanjem komande *git--version*.

Navedeno je prikazano u Listingu 18-4.

Listing 18-4: *Instaliranje Git serverske komponente*

```
gituser@gitsrv:/$ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 66 not upgraded.
Need to get 0 B/3,905 kB of archives.
After this operation, 32.2 MB of additional disk space will be used.
Selecting previously unselected package git.
(Reading database ... 101433 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.17.1-1ubuntu0.1_amd64.deb ...
Unpacking git (1:2.17.1-1ubuntu0.1) ...
Setting up git (1:2.17.1-1ubuntu0.1) ...
gituser@gitsrv:/$ git --version
git version 2.17.1
```

U nastavku, osim naziva *gitsrv* za referisanje ove mašine korišćeni su i nazivi *Git* server i udaljeni *Git* repozitorijum.

Nakon instaliranja *git* paketa potrebno je da se inicijalizuje tzv. ogoljeni (eng. *bare*) *Git* repozitorujum izvršavanjem komande *git--bare init*. Repozitorijum je imenovan kao *project.git*.

Prema konvenciji, koja se u ovom domenu koristi, ogoljeni repozitorijum mora da ima ekstenziju *.git*. Prvo se kreira folder pod nazivom *project.git bash* komandom *mkdir* u *home* folderu korisnika *gituser* (lokacija *home* foldera je */home/gituser,* i ovaj folder se skraćeno referiše samo simbolom ~), a zatim se u tom folderu izvršava komanda *git--bare init*.

Izvršavanjem komande *git –bare init* u folderu *project.git* se kreira osnovna struktura udaljenog *Git* repozitorijuma.

Navedeno je prikazano u Listingu 18-5.

Listing 18-5: *Inicijalizovanje udaljenog Git repozitorijuma*

```
gituser@gitsrv:/$ cd ~
gituser@gitsrv:~$ pwd
/home/gituser
gituser@gitsrv:~$ ls
gituser@gitsrv:~$ mkdir project.git
gituser@gitsrv:~$ cd project.git/
gituser@gitsrv:~/project.git$ git --bare init
Initialized empty Git repository in /home/gituser/project.git/
gituser@gitsrv:~/project.git$ ls -l
total 32
drwxrwxr-x 2 gituser gituser 4096 Jul 14 05:07 branches
-rw-rw-r-- 1 gituser gituser   66 Jul 14 05:07 config
-rw-rw-r-- 1 gituser gituser   73 Jul 14 05:07 description
-rw-rw-r-- 1 gituser gituser   23 Jul 14 05:07 HEAD
drwxrwxr-x 2 gituser gituser 4096 Jul 14 05:07 hooks
drwxrwxr-x 2 gituser gituser 4096 Jul 14 05:07 info
drwxrwxr-x 4 gituser gituser 4096 Jul 14 05:07 objects
drwxrwxr-x 4 gituser gituser 4096 Jul 14 05:07 refs
```

18.2.2 Instaliranje Git klijenata

Isto kao i u slučaju *gitsrv* mašine, na *gitclient* i *gitclient2* mašinama prvo je potrebno da se instalira *git* paket. S obzirom da *Ubuntu Desktop* nema preinstalirane sve biblioteke i pakete koji su potrebni za *Git* instalaciju, u procesu instaliranja *Git*-a, komponente koje nedostaju su automatski instalirane.

Instalacija *git* paketa na mašini *gitclient* prikazana je u Listingu 18-6. Identične korake potrebno je izvršiti i na mašini *gitclient2*.

Listing 18-6: *Instaliranje Git-a na mašini gitclient*

```
nebojsa@gitclient:~$ sudo apt-get install git
[sudo] password for nebojsa:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
...
nebojsa@gitclient:~$ git --version
git version 2.17.1
```

Git obezbeđuje korisnu alatku pod nazivom *git config* preko koje se podešavaju vrednosti konfiguracionih promenljivih (promenljivih okruženja). *Git* čuva sva podešavanja (vrednosti konfiguracionih varijabli) u fajlu *.gitconfig*, koji se nalazi u *home* folderu korisnika. U slučaju da je potrebno da se vrednosti konfiguracionih varijabli podese globalno, za ceo sistem, koristi se komanda *git config --global*. Ako se izostavi *--global* parametar, onda se podešavanja vrše samo za tekući (trenutni) *Git* repozitorijum. *Git* globalna sistemska podešavanja (za sve repozitorijume u sistemu) čuvaju se u fajlu */etc/gitconfig*.

U Tabeli 18-4 prikazane su *Git* varijable okruženja sa opisima šta koja varijabla radi.

Tabela 18-4: *Git promenljive okruženja*

Varijabla	Opis
user.name	Definisanje korisničkog imena *Git* korisnika. Ovu informaciju *Git* koristi za svaku *commit* operaciju, da bi u bazu upisao koje promene je izvršio koji korisnik.
user.email	Definisanje *email* adrese *Git* korisnika. Ovu informaciju *Git* koristi za svaku *commit* operaciju, da bi u bazu upisao koje promene je izvršio koji korisnik.
branch.autosetuprebase	Kada korisnik vuče promene sa udaljenog repozitorijuma, ako se fajlovi na udaljenom repozitorijumu razlikuju od fajlova na lokalnom repozitorijumu, *Git* podrazumevano spaja promene. Spajanje može da se izbegne ako se za vrednost *branch.autosetuprebase* varijable postavi vrednost *always*.
color.ui *color.status* *color.branch*	Definisanje načina prikazivanja *Git* naredbi u konzoli.
core.editor	Definisanje programa za editovanje fajlova. Po *default* podešavanjima, *Git* koristi podrazumevani sistemski editor teksta.
merge.tool	Definiše podrazumevani alat za integrisanje promena koje su u konfliktu.

Da bi se vrednosti bilo koje od navedenih varijabli promenile, koristi se sledeća sintaksa: *git config --global nazivVarijable vrednost*. Za prikaz vrednosti svih *Git* promenljivih okruženja koristi se naredba *git config --list*.

Pre korišćenja *Git*-a, na klijentskim mašinama potrebno je da se podese vrednosti konfiguracionih varijabli. U listinzima 18-7 i 18-8 prikazana su ova podešavanja za klijentske mašine *gitclinet* i *gitclient2*, respektivno.

Listing 18-7: *Podešavanje vrednosti Git varijabli okruženja na mašini gitclient*

```
nebojsa@gitclient:~$ git config --global user.name "Nebojsa Bacanin"
nebojsa@gitclient:~$ git config --global user.email "nbacanin@singidunum.ac.rs"
nebojsa@gitclient:~$ git config --global branch.autosetuprebase always
nebojsa@gitclient:~$ git config --global color.ui true
nebojsa@gitclient:~$ git config --global color.status auto
nebojsa@gitclient:~$ git config --global color.branch auto
nebojsa@gitclient:~$ git config --global core.editor nano
nebojsa@gitclient:~$ git config --global merge.tool vimdiff
nebojsa@gitclient:~$ git config --list
user.name=Nebojsa Bacanin
user.email=nbacanin@singidunum.ac.rs
branch.autosetuprebase=always
color.ui=true
color.status=auto
color.branch=auto
colore.editor=nano
merge.tool=vimdiff
```

Listing 18-8: *Podešavanje vrednosti Git varijabli okruženja na mašini gitclient2*

```
ivana@gitclient2:~$ git config --global user.name "Ivana Strumberger"
ivana@gitclient2:~$ git config --global user.email "istrumberger@singidunum.ac.rs"
ivana@gitclient2:~$ git config --global branch.autosetuprebase always
ivana@gitclient2:~$ git config --global color.ui true
ivana@gitclient2:~$ git config --global color.status auto
ivana@gitclient2:~$ git config --global color.branch auto
ivana@gitclient2:~$ git config --global core.editor nano
ivana@gitclient2:~$ git config --global merge.tool vimdiff
ivana@gitclient2:~$ git config --list
user.name=Ivana Strumberger
user.email=istrumberger@singidunum.ac.rs
branch.autosetuprebase=always
color.ui=true
color.status=auto
color.branch=auto
core.editor=nano
merge.tool=vimdiff
```

Nakon podešavanja varijabli okruženja, potrebno je inicijalizovati *Git* repozitorijume na svakom klijentu posebno.

Neka se na mašini *gitclient* lokalni *Git* repozitorijum nalazi u home folderu korisnika nebojsa, pod nazivom *nebojsa_repo* (putanja: */home/nebojsa/nebojsa_repo*). Na mašini *gitclient2*, neka lokalni repozitorijum bude u home folderu korisnika ivana pod nazivom *ivana_repo* (putanja: */home/ivana/ivana_repo*).

Za inicijalizaciju lokalnog repozitorijuma koristi se komanda *git --init*.

Postupak kreiranja repozitorijuma na mašini *gitclient* dat je u Listingu 18-9, dok je isti postupak na mašini *gitclient2* prikazan u Listingu 18-10.

Nakon izvršavanja ovih komandi u zadatom folderu kreira se folder *.git*.

Listing 18-9: *Inicijalizovanje lokalnog Git repozitorujuma na mašini gitclient*

```
nebojsa@gitclient:~$ pwd
/home/nebojsa
nebojsa@gitclient:~$ mkdir nebojsa_repo
nebojsa@gitclient:~$ cd nebojsa_repo/
nebojsa@gitclient:~/nebojsa_repo$ git init
Initialized empty Git repository in /home/nebojsa/nebojsa_repo/.git/
nebojsa@gitclient:~/nebojsa_repo$ ls .git
branches config description HEAD hooks info objects  refs
```

Listing 18-10: *Inicijalizovanje lokalnog Git repozitorujuma na mašini gitclient2*

```
ivana@gitclient2:~$ pwd
/home/ivana
ivana@gitclient2:~$ mkdir ivana_repo
ivana@gitclient2:~$ cd ivana_repo/
ivana@gitclient2:~/ivana_repo$ git init
Initialized empty Git repository in /home/ivana/ivana_repo/.git/
ivana@gitclient2:~/ivana_repo$ ls .git
branches config description HEAD hooks info objects  refs
```

18.2.3. Generisanje RSA ključeva i dodavanje udaljenog repozitorijuma

Za uspostavljanje sigurne komunikacije između *Git* servera (udaljenog repozitorijuma) i *Git* klijenata, potrebno je da se generišu parovi javnih i privatnih *RSA* ključeva. Ključevi se generišu na klijentima, da bi se nakon toga kopirali na server.

Generisanje ključeva vrši se komandom *ssh-keygen*, kada se kreira *.ssh* direktorijum. Tokom procesa generisanja ključeva korisnik treba da odabere tajnu reč (eng. *passphrase*) koju treba da sačuva na bezbednoj lokaciji. Komandom *ssh-keygen* generišu se privatni ključ (na primer *id_rsa*) i javni ključ (na primer *id_rsa.pub*).

Prikazi generisanja ključeva za korisnika *nebojsa* na mašini *gitclient* i za korisnika *ivana* na mašini *gitclient2* dati su u listinzima 18-11 i 18-12, respektivno. Generisani ključevi se nalaze u foderu */home/user_name/.ssh*.

Listing 18-11: *Generisanje RSA ključeva korisnika nebojsa na mašini gitclient*

```
nebojsa@gitclient:~$ pwd
/home/nebojsa
nebojsa@gitclient:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nebojsa/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nebojsa/.ssh/id_rsa.
Your public key has been saved in /home/nebojsa/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:8WIEmsl2rNwdyhK6ZZ+gphn+qPBZZm7FA95u3ztEsaI nebojsa@gitclient
The key's randomart image is:
+---[RSA 2048]---+
|       .         |
|   . = .  .      |
|    B o +  o     |
|   +.* +.+o      |
|  ..B++.So.      |
|   =.+Eo ..      |
|o + +oo. .       |
|o*.*. o  ..      |
|=o+o..... oo     |
+----[SHA256]----+
nebojsa@gitclient:~$ ls .ssh
id_rsa  id_rsa.pub
```

Listing 18-12: *Generisanje RSA ključeva korisnika ivana na mašini gitclient2*

```
ivana@gitclient2:~$ pwd
/home/ivana
ivana@gitclient2:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key
(/home/ivana/.ssh/id_rsa):/home/ivana/.ssh/id_ivana.rsa
Created directory '/home/ivana/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ivana/.ssh/id_ivana_rsa
Your public key has been saved in /home/ivana/.ssh/id_ivana_rsa.pub.
The key fingerprint is:
SHA256:e+kxQDAWi3hqTm24xEHGr6FXs/F2CDfPlA9uCOkxWQk ivana@gitclient2
The key's randomart image is:
+---[RSA 2048]---+
| .o E.=o         |
| o.. oo+         |
| o.o+. ..        |
| ..*% o.+        |
| .B=o@ OSo       |
|.=ooo = *o..     |
| .o . o. =       |
|         o o     |
|          .      |
+----[SHA256]----+
ivana@gitclient2:~$ ls .ssh
id_ivana_rsa  id_ivana_rsa.pub
```

U Listingu 18-12 potrebno je obratiti pažnju da korisnik *ivana* nije izabrala podrazumevane nazive fajlova sa ključevima *id_rsa* i *id_rsa.pub* već su nazivi privatnog i javnog *RSA* ključa *id_ivana_rsa* i *id_ivana_rsa.pub*, respektivno.

U prikazanoj simulaciji *Git* okruženja, dva korisnika *nebojsa* i *ivana* rade na zajedničkom projektu koji se čuva na udaljenom *Git* repozitorijumu na mašini *gitsrv* koja ima *IP* adresu 10.0.0.101. Za potrebe uspostavljanja sigurne konekcije sa *Git* serverom, potrebno je da oba korisnika dodaju svoje javne *RSA* ključeve na *gitsrv* mašinu. Komanda koja se koristi za kopiranje ključeva na udaljenu mašinu je *ssh-copy- id*.

U Listingu 18-13, prikazano je kopiranje javnog *RSA* ključa na mašinu *gitsrv* sa mašine *gitclient*.

Listing 18-13: *Kopiranje javnog RSA ključa sa mašine gitclient na mašinu gitsrv*

```
nebojsa@gitclient:~$ pwd
/home/nebojsa
nebojsa@gitclient:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub gituser@10.0.0.101
/usr/bin/ssh-copy-id:    INFO:    Source    of    key(s)    to    be    installed:
"/home/nebojsa/.ssh/id_rsa.pub"
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'gituser@10.0.0.101'"
and check to make sure that only the key(s) you wanted were added.
```

Nakon kopiranja javnog ključa, korisnik *nebojsa* sa mašine *gitclient* može da uspostavi sigurnu *SSH* konekciju ka mašini *gitsrv* korišćenjem pristupnih kredencijala lokalnog korisnika mašine *gitsrv*, što je u ovom slučaju korisnik *gituser*.

Uspostavljanje sigurne *SSH* konekcije sa mašine *gitclient* na udaljenu mašinu *gitsrv* prikazano je u Listingu 18-14.

Listing 18-14: *Uspostavljanje sigurne SSH konekcije sa mašine gitclient ka mašini gitsrv*

```
nebojsa@gitclient:~$ ssh gituser@10.0.0.101
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Jul 14 05:58:44 UTC 2018

  System load: 0.0                Processes:             89
  Usage of /:  17.3% of 24.48GB   Users logged in:       1
  Memory usage: 7%                IP address for enp0s3: 10.0.2.15
  Swap usage:  0%                 IP address for enp0s8: 10.0.0.101

 * Introducing Minimal Ubuntu for docker and clouds. 30 MB base image and
   optimised kernels on public clouds. Made for machines and containers.

   - https://bit.ly/minimal-ubuntu


65 packages can be updated.
3 updates are security updates.


*** System restart required ***
Last login: Sat Jul 14 05:54:43 2018 from 10.0.0.103
```

Sličan postupak kopiranja javnog *RSA* ključa potrebno je izvršiti i za mašinu *gitclient2* (Listing 18-15).

Listing 18-15: *Kopiranje javnog RSA ključa sa mašine gitclient2 na mašinu gitsrv*

```
ivana@gitclient2:~$ pwd
/home/ivana
ivana@gitclient2:~$ ssh-copy-id -i ~/.ssh/id_ivana_rsa.pub gituser@10.0.0.101
/usr/bin/ssh-copy-id:    INFO:    Source    of    key(s)    to    be    installed:
"/home/ivana/.ssh/id_rsa.pub"
Enter passphrase for key '/home/ivana/.ssh/id_rsa':

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'gituser@10.0.0.101'"
and check to make sure that only the key(s) you wanted were added.
```

Uspostavljanje sigurne *SSH* konekcije sa mašine *gitclient2* na udaljenu mašinu *gitsrv* vrši se analogno kao i u slučaju mašine *gitclient* (pogledati Listing 18-14).

Nakon uspostavljanja sigurne *SSH* konekcije između *gitsrv* udaljenog repozitorijuma i mašina *gitclient* i *gitclient2*, potrebno je svaku klijentsku mašinu povezati sa udaljenim repozitorijumom. Za ove potrebe koristi se komanda *git remote add origin*. Naziv udaljenog repozitorijuma na *Git* serveru je *project.git*.

U prikazanoj simulaciji, korisnik *nebojsa* sa mašine *gitclient* dodaje udaljeni repozitorijum, dok korisnik *ivana* sa mašine *gitclient2* klonira udaljeni repozitorijum (videti u nastavku).

Dodavanje udaljenog repozitorijuma za klijentsku mašinu *gitclient* prikazano je u Listingu 18-16.

Listing 18-16: *Dodavanje udaljenog Git repozitorijuma na mašinu gitclient*

```
nebojsa@gitclient:~$ pwd
/home/nebojsa
nebojsa@gitclient:~$ cd nebojsa_repo/
nebojsa@gitclient:~/nebojsa_repo$ git remote add origin
gituser@10.0.0.101:project.git
```

Da bi izvršio testiranje implementiranog *Git* sistema, korisnik *nebojsa* na klijentskoj mašini *gitclient* u svom radnom direktorijumu (putanja: */home/nebojsa/nebojsa_repo*) kreira jednostavan *README* fajl, dodaje ga u svoju faznu oblast komadom *add*, zatim ga izvršava na svoj lokalni repozitorijum komandom *commit*, i konačno promene snima na udaljeni repozitorijum opercijom *push*.

Konačno, izvršavanjem operacije *pull*, korisnik *nebojsa* proverava da li je njegov lokalni repozitorijum sinhronizovan sa udaljenim repozitorijumom.

Sve navedene *Git* komande detaljnije su opisane u nastavku. Koraci koji su gore opisani prikazani su u Listingu 18-17.

Listing 18-17: *Kreiranje README fajla i dodavanje na udaljeni repozitorijum*

```
nebojsa@gitclient:~/nebojsa_repo$ echo 'TODO: Nebojsa initial project readme
file' > README
nebojsa@gitclient:~/nebojsa_repo$ git status -s
?? README
nebojsa@gitclient:~/nebojsa_repo$ git add .
nebojsa@gitclient:~/nebojsa_repo$ git status -s
A README
nebojsa@gitclient:~/nebojsa repo$ git commit -m 'Initial commit'
[master (root-commit) 18d6ce4] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 README
nebojsa@gitclient:~/nebojsa_repo$ git push origin master
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Counting objects: 3, done.
Writing objects: 100% (3/3), 259 bytes | 86.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To 10.0.0.101:project.git
 * [new branch]      master -> master
nebojsa@gitclient:~/nebojsa_repo$ git pull origin master
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
From 10.0.0.101:project
 * branch            master     -> FETCH_HEAD
Already up to date.
```

S druge strane, na klijentskoj mašini *gitclient2*, korisnik ivana radi nešto drugačije. Korisnik *ivana* ne želi da koristi svoj lokalni *Git* repozitorijum, koji je inicijalizovan u folderu */home/ivana/ivana_repo*, već hoće da klonira udaljeni repozitorijum sa *gitsrv* mašine (podsećanje: *gitsrv* mašina ima *IP* adresu 10.0.0.101).

Korisnik *ivana* klonira udaljeni repozitorijum i nakon kloniranja, u folderu */home/ivana* nalazi se folder *project*, gde se nalazi lokalna kopija udaljenog repozitorijuma (na serverskoj mašini *gitsrv, Git* repozitorijum je inicijalizovan u folderu *project.git* i zbog toga je naziv foldera sa kloniranim repozitorijumom *project*).

Nakon izvršenog kloniranja, korisnik *ivana* u folderu svog lokalnog repozitorijuma *project* vidi fajl *README*, koji je kreirao u svom lokalnom repozitorijumu, i nakon toga presnimio na udaljeni *Git* repozitorijum korisnik *nebojsa* sa mašine *gitclient*.

Konačno, korisnik *ivana* proverava da li je njen lokalni repozitorijum sinhronizovan sa udaljenim repozitorijumum izvršavanjem komande *git pull*.

Navedeno je prikazano u Listingu 18-18.

Listing 18-18: *Kloniranje udaljenog Git repozitorijuma*

```
ivana@gitclient2:~$ git clone gituser@10.0.0.101:project.git
Cloning into 'project'...
gituser@10.0.0.101's password:
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), 259 bytes | 51.00 KiB/s, done.
ivana@gitclient2:~$ ls
examples.desktop project
ivana@gitclient2:~$ cd project/
ivana@gitclient2:~/project$ ls
README
ivana@gitclient2:~/project$ cat README
TODO: Nebojsa initial project readme file
ivana@gitclient2:~/project$ git pull
gituser@10.0.0.101's password:
From 10.0.0.101:project
 * branch            master     -> FETCH_HEAD
Already up to date.
Current branch master is up to date.
```

S obzirom da je ovakva postavka sistema potrebna u nastavku ove simulacije, još jednom se napominje da korisnik *nebojsa* na mašini *gitclient* lokalni repozitorijum čuva u folderu */home/nebojsa/nebojsa_repo*, dok se lokalni repozitorijum korisnika *ivana* na mašini *gitclient2* nalazi u folderu */home/ivana/project*.

U nastavku je u ovako konfigurisanom *Git* server – *Git* klijent okruženju prikazano izvršavanje osnovnih *Git* operacija (naredbi).

18.3. OSNOVNE *GIT* OPERACIJE

U ovom delu ukratko su prikazane osnovne *Git* operacije u *Linux bash* okruženju kroz praktične simulacije i scenarije.

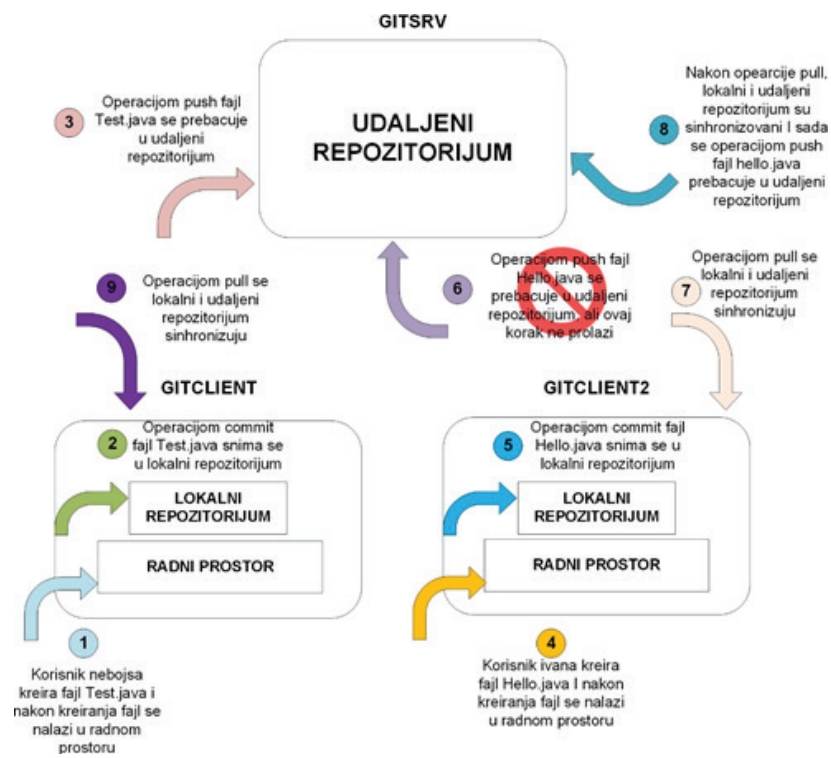
18.3.1. Operacije add, commit, push i pull

Za potrebe ilustrovanja upotrebe *Git* operacija *add, commit, push* i *pull* korišćen je scenario koji je dat na Slici 18-5.

Scenario koji je prikazan na Slici 18-5 predstavlja jedan jednostavni radni tok, kojim su obuhvaćene najosnovnije *Git* operacije. Redosled koraka koji se izvršavaju u scenariju sa slike označen je celobrojnim vrednostima.

Uz opis svakog koraka (*Git* operacije) prikazane su i bash komande.

Slika 18-5: *Ilustracija Git operacija add, commit, push i pull*



Korak 1

U koraku 1, korisnik *nebojsa* kreira fajl *Test.java*. Radi pojednostavljenja, sadržaj fajla *Test.java* se zanemaruje, i fajl se kreira jednostavnom *bash* komandom *touch* (ova komanda služi za kreiranje praznih fajlova).

Nakon kreiranja, fajl *Test.java* se nalazi u privatnom radnom prostoru korisnika *nebojsa*. Komandom *git status -s* korisnik *nebojsa* proverava da li postoje promene u njegovom privatnom radnom prostoru koje nisu upisane na lokalni repozitorijum.

Na osnovu rezultata komande *git status -s*, korisnik *nebojsa* zaključuje da fajl *Test.java* nije upisan na njegov lokalni repozitorijum. Pored naziva fajla *Test.java*, nakon izvršavanja komande *git status -s*, nalaze se dva simbola „znak pitanja", što označava da fajl nije upisan u lokalni repozitorijum.

Izvršavanje koraka 1 prikazano je u Listingu 18-19.

Listing 18-19: *Korak 1 – Git operacije add, commit, push i pull*

```
nebojsa@gitclient:~/nebojsa_repo$ pwd
/home/nebojsa/nebojsa_repo
nebojsa@gitclient:~/nebojsa_repo$ touch Test.java
nebojsa@gitclient:~/nebojsa_repo$ ls
README  Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
?? Test.java
```

Korak 2

Korisnik *nebojsa* hoće da upiše fajl *Test.java* u svoj lokalni repozitorijum. Za ostvarivanje ovog cilja, fajl prvo treba da doda u faznu oblast pomoću *Git* operacije *add*. Tek nakon dodavanja fajla *Test.java* u faznu oblast, korisnik *nebojsa* može komandom *commit* da upiše fajl na svoj lokalni repozitorijum. Sa ciljem uprošćavanja, dodavanje fajla u faznu oblast nije prikazano na Slici 18-5.

Nakon dodavanja fajla *Test.java* u faznu oblast, korisnik nebojsa ponovo izvršava naredbu *git status -s* i u rezultatu ove naredbe, pored naziva fajla *Test.java*, ispisuje se slovo „A", što označava da su sve promene koje su izvršene nad ovim fajlom dodate u faznu oblast.

Izvršavanjem naredbe *commit* i upisivanjem promena na lokalni repozitorijum, korisnik *nebojsa* se na neki način obavezao da će promene da upiše i na udaljeni repozitorijum.

Kao što se vidi iz priloženog listinga (pogledati ispod), komanda *commit* koristi argument m, koji služi za navođenje simboličkog opisa promena.

Konačno, nakon izvršavanja naredbe *commit*, korisnik *nebojsa* ponovo izvršava naredbu *git status -s*, i u ovom slučaju se ne prikazuje ni jedan rezultat, što znači da ne postoje promene koje nisu upisane u lokalni repozitorijum.

Prikaz koraka 2 dat je u Listingu 18-20.

Listing 18-20: *Korak 2 – Git operacije add, commit, push i pull*

```
nebojsa@gitclient:~/nebojsa_repo$ git add Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
A  Test.java
nebojsa@gitclient:~/nebojsa_repo$ git commit -m "Added file Test.java"
[master 3f33f1e] Added file Test.java
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
```

Korak 3

U koraku 3, korisnik *nebojsa* promene iz lokalnog repozitorijuma upisuje na udaljeni repozitorijum. Da bi ovo ostvario, korisnik izvršava komandu *git push origin master*. Podrazumevana grana razvojnog toka naziva se *master*, dok se udaljeni repozitorijum u *Git* terminologiji naziva *origin*. Ako se radi u nekoj drugoj, nepodrazumevanoj grani, umesto *master* koristiće se naziv grane u kojoj se trenutno radi.

Nakon izvršavanja operacije *push*, korisnik *nebojsa* izvršava operaciju *git log,* koja prikazuje sve istorijske promene sa relevantnim podacima, kao što su: *SHA*-1 promene, datum i vreme izvršavanja promene, ime i *email* adresa izvršioca promene, itd.

Korak 3 je prikazan u Listingu 18-21.

Listing 18-21: *Korak 3 – Git operacije add, commit, push i pull*

```
nebojsa@gitclient:~/nebojsa_repo$ git push origin master
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Counting objects: 3,  done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To 10.0.0.101:project.git
   0ad4e9c..3f33f1e  master -> master
nebojsa@gitclient:~/nebojsa repo$ git log
commit 3f33f1e49af32c6b0879795c697459d2e2cef4be (HEAD -> master, origin/master)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 19:42:35 2018 +0200

    Added file Test.java

commit 0ad4e9c92ae8d8293d443cddca733cae59c89e6b
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 08:51:57 2018 +0200

    Initial commit
```

Korak 4

U koraku 4, korisnik *ivana* na računaru *gitclient2* kreira fajl *Hello.java*. Isto kao i u slučaju fajla *Test.java*, sadržaj fajla *Hello.java* nije definisan, već je fajl kreiran jednostavnom *bash* komandom *touch*. Nakon kreiranja, fajl *Hello.java* nalazi se u privatnom radnom prostoru korisnika *ivana*. Izvršavanjem komande *git status -s* korisnik *ivana* proverava da li postoje promene u njenom privatnom radnom prostoru koje nisu upisane u lokalni repozitorijum.

Izvršavanjem komande *git status -s* korisnik *ivana* vidi da promene nad fajlom *Hello.java* nisu upisane u njen lokalni repozitorijum.

Ilustracija koraka 4 data je u Listingu 18-22.

Listing 18-22: *Korak 4 – Git operacije add, commit, push i pull*

```
ivana@gitclient2:~/project$ pwd
/home/ivana/project
ivana@gitclient2:~/project$ ls
README
ivana@gitclient2:~/project$ touch Hello.java
ivana@gitclient2:~/project$ git status -s
?? Hello.java
```

Korak 5

Korisnik *ivana* sada želi da upiše fajl *Hello.java* na svoj lokalni repozitorijum. Međutim fajl prvo treba da se pripremi za upisivanje na repozitorijum tako što će da se doda u faznu oblast korišćenjem *Git* operacije *add*. Tek nakon dodavanja fajla *Hello.java* u faznu oblast, korisnik *ivana* može izvršavanjem komande *git commit* da upiše fajl na svoj lokalni repozitorijum.

Nakon dodavanja fajla *Hello.java* u faznu oblast, korisnik *ivana* ponovo izvršava naredbu *git status -s*, i u rezultatima ove naredbe pored naziva fajla *Hello.java* ispisuje se slovo „A", što označava da su sve promene dodate u faznu oblast.

Kao i u slučaju korisnika *nebojsa*, izvršavanjem naredbe *git commit* i upisivanjem promena u lokalni repozitorijum, korisnik *ivana* se na neki način obavezala da će promene da upiše i na udaljeni repozitorijum. Naredba *commit* navodi se uz argument m, koji služi za definisanje simboličkog opisa promene.

Nakon izvršavanja naredbe *commit*, korisnik ivana ponovo izvršava naredbu *git status -s* i ova naredba ne prikazuje ni jedan rezultat, što znači da ne postoje promene u faznoj oblasti koje nisu upisane na lokalni repozitorijum.

Korak 5 prikazan je u Listingu 18-23.

Listing 18-23: *Korak 5 – Git operacije add, commit, push i pull*

```
ivana@gitclient2:~/project$ git status -s
??  Hello.java
ivana@gitclient2:~/project$ git add Hello.java
ivana@gitclient2:~/project$ git status -s
A  Hello.java
ivana@gitclient2:~/project$ git commit -m "Added Hello.java"
[master 65f6081] Added Hello.java
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Hello.java
ivana@gitclient2:~/project$ git status -s
```

Korak 6

U koraku 6, korisnik *ivana* želi da upiše promene iz svog lokalnog repozitorijuma na udaljeni *Git* repozitorijum i izvršava naredbu *git push origin master*. Međutim, na iznenađenje korisnika, sinhronizacija sa udaljenim repozitorijumom je odbijena, kao što se vidi iz Listinga 18-24.

Listing 18-24: *Korak 6 – Git operacije add, commit, push i pull*

```
ivana@gitclient2:~/project$ git push origin master
gituser@10.0.0.101's password:
To 10.0.0.101:project.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'gituser@10.0.0.101:project.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Izvršavanje komande *push* je odbijeno zato što u trenutku izvršavanja ove operacije, lokalni repozitorijum na mašini *gitclient2* nije bio sinhronizovan sa udaljenim repozitorijumom na mašini *gitsrv*. Da bi komanda *push* mogla da se izvrši, potrebno je da sledeći uslov bude zadovoljen:

sadržaj lokalnog repozitorijuma = sadržaj udaljenog repozitorijuma – lokalne promene.

Jasno je da u trenutku izvršavanja *push* operacije ovaj uslov nije bio zadovoljen, zato što se u tom trenutku na udaljenom repozitorijumu nalazio fajl *Test.java*, koji je snimio korisnik *nebojsa* sa mašine *gitclient*, dok se u lokalnom repozitorijumu korisnika ivana na mašini *gitclient2* nalazio samo fajl *Hello.java*.

Korak 7

Da bi promene iz lokalnog repozitorijuma mašine *gitclient2* (u ovom slučaju to je fajl *Hello.java*) mogle trajno da se sačuvaju na udaljenom repozitorijumu, korisnik *ivana* na mašini *gitclient2* mora pre izvršavanja operacije *git push* da izvrši operaciju *git pull*, koja vrši sinhronizaciju udaljenog sa lokalnim repozitorijumom (repozitorijumom na mašini *gitclient2* sa repozitorijumom na mašini *gitsrv*).

Listing 18-25: *Korak 7 – Git operacije add, commit, push i pull (1)*

```
ivana@gitclient2:~/project$ git pull
gituser@10.0.0.101's password:
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From 10.0.0.101:project
   0ad4e9c..3f33f1e  master     -> origin/master
First, rewinding head to replay your work on top of it...
Applying: Added Hello.java
ivana@gitclient2:~/project$ ls
Hello.java README  Test.java
```

Nakon izvršavanja *git pull* operacije, korisnik *ivana* izvršava *git log* naredbu.

Listing 18-26: *Korak 7 – Git operacije add, commit, push i pull (2)*

```
ivana@gitclient2:~/project$ git log
commit dd73f11e85a664f7f562866268fc15cad7dc717e (HEAD -> master)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sat Jul 14 19:58:05 2018 +0200

    Added Hello.java

commit 3f33f1e49af32c6b0879795c697459d2e2cef4be (origin/master, origin/HEAD)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 19:42:35 2018 +0200

    Added file Test.java

commit 0ad4e9c92ae8d8293d443cddca733cae59c89e6b
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 08:51:57 2018 +0200

    Initial commit
```

Iz rezultata *git log* naredbe vidi se da poslednja promena iz lokalnog repozitorijuma nije upisana u udaljeni repozitorijum (*commit* koji ima *SHA*-1 vrednost *dd73f11e85a664f7f562866268fc15cad7dc717e*).

Prikaz koraka 7 dat je u listinzima 18-25 i 18-26.

Korak 8

Konačno, tek kada su lokalni repozitorijum na mašini gitclient2 i udaljeni repozitorijum sinhronizovani, korisnik ivana može da izvrši git push operaciju, nakon čega ponovo izvršava git log naredbu, gde vidi da su sve promene iz lokalnog repozitorijuma upisane na udaljeni repozitorijum i da su ova dva repozitorijuma sinhronizovani.

Korak 8 prikazan je u Listingu 18-27.

Listing 18-27: *Korak 8 – Git operacije add, commit, push i pull*

```
ivana@gitclient2:~/project$ git push origin master
gituser@10.0.0.101's password:
Counting objects: 2,  done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 294 bytes | 294.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To 10.0.0.101:project.git
   3f33f1e..dd73f11  master -> master
ivana@gitclient2:~/project$ git log
commit dd73f11e85a664f7f562866268fc15cad7dc717e (HEAD -> master, origin/master,
origin/HEAD)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sat Jul 14 19:58:05 2018 +0200

    Added Hello.java

commit 3f33f1e49af32c6b0879795c697459d2e2cef4be
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 19:42:35 2018 +0200

    Added file Test.java

commit 0ad4e9c92ae8d8293d443cddca733cae59c89e6b
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 08:51:57 2018 +0200

    Initial commit
```

Korak 9

S obzirom da su promene iz lokalnog repozitorijuma na mašini ***gitclient2*** upisane na udaljeni repozitorijum, sada lokalni repozitorijum na mašini *gitclient* nije sinhronizovan sa udaljenim repozitorijumom. Zbog toga korisnik *nebojsa* na mašini *gitclient* izvršava operaciju *git pull* i dva repozitorijuma se sinhronizuju.

Nakon izvršavanja operacija u koraku 9, sva tri repozitorijuma su sinhronizovana.

Korak 9 je dat u u Listingu 18-28.

Listing 18-28: *Korak 9 – Git operacije add, commit, push i pull*

```
nebojsa@gitclient:~/nebojsa_repo$ git pull
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
From 10.0.0.101:project
   3f33f1e..dd73f11  master     -> origin/master
Updating 3f33f1e..dd73f11
Fast-forward
 Hello.java | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Hello.java
Current branch master is up to date.
nebojsa@gitclient:~/nebojsa_repo$ ls
Hello.java README Test.java
nebojsa@gitclient:~/nebojsa_repo$ git log
commit dd73f11e85a664f7f562866268fc15cad7dc717e (HEAD -> master, origin/master)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sat Jul 14 19:58:05 2018 +0200

    Added Hello.java

commit 3f33f1e49af32c6b0879795c697459d2e2cef4be
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 19:42:35 2018 +0200

    Added file Test.java

commit 0ad4e9c92ae8d8293d443cddca733cae59c89e6b
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 08:51:57 2018 +0200

    Initial commit
```

18.3.2. *Git* naredbe za pregled i izmenu promena

Korisnik *ivana* na mašini *gitclient2* izvršila je modifikaciju svog fajla *Hello.java* i ubacila je pozdravni *Java* kod, kao što je prikazano u Listingu 18-29.

Listing 18-29: *Izmene fajla Hello.java*

```
public class Hello {


    public static void main(String[] args) {
        System.out.println("HELLO WORLD!");
    }

}
```

Korisnik *ivana* zatim dodaje iz svog radnog direktorijuma u faznu oblasti i izvršava operaciju *commit* (Listing). Slovo „M“ pored naziva fajla *Hello.java* u rezultatima komande *git status -s* označava da je fajl *Hello.java* dodat u faznu oblast, ali da je u međuvremenu modifikovan i da je potrebno da se i izmenjena verzija fajla doda u faznu oblast.

Listing 18-30: *Dodavanje izmena fajla Hello.java u faznu oblast*

```
ivana@gitclient2:~/project$ nano Hello.java
ivana@gitclient2:~/project$ cat Hello.java
public class Hello {


    public static void main(String[] args) {
        System.out.println("HELLO WORLD!");
    }

}

ivana@gitclient2:~/project$ git status -s
 M Hello.java
ivana@gitclient2:~/project$ git add Hello.java
ivana@gitclient2:~/project$ git commit -m "Added code in Hello.java file"
[master 979ca95] Added code in Hello.java file
 1 file changed, 9 insertions(+)
```

Korisnik *ivana* sada hoće da pogleda promene koje je izvršila svojom poslednjom operacijom *commit*. Da bi videla ove promene, *ivana* izvršava komandu *git log,* gde vidi da njene poslednje promene nisu sinhronizovane sa udaljenim repozitorijumum i čita *SHA*-1 vrednost svoje poslednje promene, koja predstavlja *ID* broj poslednje izvršene operacije commit (*SHA*-1 *commit ID*). U ovom slučaju, *SHA*-1 vrednost je: *979ca95f60006156c2066ab1be37817cc4b4b927*.

Kada je pročitala *commit ID* svoje poslednje promene, korisnik *ivana* izvršava *git show* naredbu, koja poslednju promenu prikazuje. U rezultatima komande *git show* navedene su sve promene koje je *ivana* izvršila u fajlu *Hello.java*. Simbol „+“ pored linija koda znači da su te linije dodate u poslednjoj promeni ovog fajla.

Izvršavanje opisanog scenarija dato je u Listingu 18-31.

Listing 18-31: *Git show i git log naredbe*

```
ivana@gitclient2:~/project$ git log
commit 979ca95f60006156c2066ab1be37817cc4b4b927 (HEAD -> master)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sun Jul 15 06:29:39 2018 +0200

    Added code in Hello.java file

commit dd73f11e85a664f7f562866268fc15cad7dc717e (origin/master, origin/HEAD)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sat Jul 14 19:58:05 2018 +0200

    Added Hello.java

commit 3f33f1e49af32c6b0879795c697459d2e2cef4be
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 19:42:35 2018 +0200

    Added file Test.java

commit 0ad4e9c92ae8d8293d443cddca733cae59c89e6b
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 08:51:57 2018 +0200

    Initial commit
ivana@gitclient2:~/project$ git show 979ca95f60006156c2066ab1be37817cc4b4b927
commit 979ca95f60006156c2066ab1be37817cc4b4b927 (HEAD -> master)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sun Jul 15 06:29:39 2018 +0200

    Added code in Hello.java file

diff --git a/Hello.java b/Hello.java
index e69de29..37ee8f9 100644
--- a/Hello.java
+++ b/Hello.java
@@ -0,0 +1,9 @@
+public class Hello {
+
+
+    public static void main(String[] args) {
+        System.out.println("HELLO WORLD!");
+    }
+
+}
+
```

Korisniku *ivana* se ne sviđaju promene koje je izvršila sa fajlom *Hello.java*, i umesto teksta „HELLO WORLD" koji će da ispisuje Java program, želi da prikaže tekst „HELLO PROJECT TEAM". *Ivana* ponovo edituje fajl *Hello.java* i menja određenu liniju koda, ali ne izvršava operaciju *commit*, već izvršava naredbu *git diff*, koja upoređuje sadržaj njenog radnog direktorijuma (u radnom direktorijumu

trenutno se nalazi fajl *Hello.java* koji ispisuje poruku „HELLO PROJECT TEAM") sa sadržajem lokalnog repozitorijuma (u lokalnom repozitorijumu trenutno se nalazi *Hello.java* fajl koji ispisuje poruku „HELLO WORLD").

U rezultatu komande *git diff* pored teksta „HELLO WORLD" ispisuje se simbol „-" , dok se pored teksta "HELLO PROJECT TEAM" nalazi simbol „+", što znači da je prva pozdravna poruka izbačena iz fajla, a da je druga dodata.

Navedeni koraci prikazani su u Listingu 18-32.

Listing 18-32: *Izvršavanje git diff naredbe*

```
ivana@gitclient2:~/project$ nano Hello.java
ivana@gitclient2:~/project$ cat Hello.java
public class Hello {


    public static void main(String[] args) {
        System.out.println("HELLO PROJECT TEAM!");
    }

}

ivana@gitclient2:~/project$ git diff
diff --git a/Hello.java b/Hello.java
index 37ee8f9..741fc51 100644
--- a/Hello.java
+++ b/Hello.java
@@ -2,7 +2,7 @@ public class Hello {


     public static void main(String[] args) {
-        System.out.println("HELLO WORLD!");
+        System.out.println("HELLO PROJECT TEAM!");
     }

 }
```

Sada je problem u tome što se u lokalnom repozitorijumu na mašini *gitclient2* nalazi pogrešna (neažurirana) verzija fajla *Hello.java*. *Ivana* bi mogla da izvrši novi *commit* sa promenama, i da oba *commit*-a upiše na udaljeni repozitorijum, ali bi to bilo previše komplikovano, zato što korisnici udaljenog repozitorijuma ne bi znali koja verzija fajla *Hello.java* je aktuelna. U ovom scenariju može da pomogne komanda *git amend*, koja menja poslednju operaciju *commit*, i upisuje novu operaciju *commit* koja ima novu *commit ID* vrednost.

Kako bi ovo postigla, korisnik *ivana*, prvo izvršava komandu *git status*, dodaje poslednje promene (fajl *Hello.java* sa novom pozdravnom porukom) u svoju faznu oblast i izvršava operaciju *commit* sa parametrom *amend* (Listing 18-33).

Listing 18-33: *Izvršavanje operacije git commit sa amend argumentom*

```
ivana@gitclient2:~/project$ git status -s
 M Hello.java
ivana@gitclient2:~/project$ git add Hello.java
ivana@gitclient2:~/project$ git commit --amend -m "Added new code in Hello.java
file"
[master ed25e7d] Added new code in Hello.java file
 Date: Sun Jul 15 06:29:39 2018 +0200
 1 file changed, 9 insertions(+)
```

Konačno, korisnik *ivana* izvršava naredbu *git log*, gde se vidi da je prethodni *commit* „pregažen" novim, i snima promene iz svog lokalnog repozitorijuma na udaljeni repozitorijum. U isto vreme, korisnik *nebojsa* na mašini *gitclient* sinhronizuje svoj lokalni repozitorijum sa udaljenim repozitorijum. Navedene operacije su prikazane u listinzima 18-34 i 18-35.

Listing 18-34*: Snimanje promena iz repozitorijuma mašine gitclient2 na udaljeni repozitorijum*

```
ivana@gitclient2:~/project$ git log
commit ed25e7d5db5dfe1edbc420e0efa85d21ee06d95b (HEAD -> master)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sun Jul 15 06:29:39 2018 +0200

    Added new code in Hello.java file
commit dd73f11e85a664f7f562866268fc15cad7dc717e (origin/master, origin/HEAD)
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sat Jul 14 19:58:05 2018 +0200

    Added Hello.java
commit 3f33f1e49af32c6b0879795c697459d2e2cef4be
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 19:42:35 2018 +0200

    Added file Test.java
commit 0ad4e9c92ae8d8293d443cddca733cae59c89e6b
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 08:51:57 2018 +0200

    Initial commit
ivana@gitclient2:~/project$ git push origin master
gituser@10.0.0.101's password:
Counting objects: 3,  done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 450 bytes | 450.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To 10.0.0.101:project.git
   dd73f11..ed25e7d  master -> master
```

Listing 18-35: *Sinhronizacija lokalnog repozitorijuma mašine gitclient sa udaljenim repozitorijumom*

```
nebojsa@gitclient:~/nebojsa_repo$ git pull
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From 10.0.0.101:project
   dd73f11..ed25e7d  master     -> origin/master
Updating dd73f11..ed25e7d
Fast-forward
 Hello.java | 9 +++++++++
 1 file changed, 9 insertions(+)
Current branch master is up to date.
nebojsa@gitclient:~/nebojsa_repo$ ls
Hello.java README  Test.java
nebojsa@gitclient:~/nebojsa_repo$ cat Hello.java
public class Hello {

    public static void main(String[] args) {
        System.out.println("HELLO PROJECT TEAM!");
    }
}
```

18.3.3. *Git stash* operacija

U praksi se često dešava situacija da rad na određenoj funkcionalnosti softvera mora privremeno da se prekine, da bi programer radio na nečemu drugom, što spada u domen hitnijih zadataka. Zbog toga, u ovakvim situacijama, programer mora da ostavi „po strani" rad na tekućoj funkcionalnosti softvera i da neko vreme radi na nečemu drugom. U ovom slučaju programer ne može da odbaci ono što je do tog trenutka uradio na novoj funkcionalnosti, ali isto tako ne može da izvrši *commit* za posao koji nije završen do kraja.

Ovakav i slični problemi u *Git* sistemu mogu da se reše korišćenjem privremenog radnog prostora, u kome programer na određeno vreme, dok se bavi nečim drugim, čuva delimično završen posao (kod). Kada završi drugi zadatak, programer se vraća poslu koji je ostao nedovršen, završava ga i za njega izvršava operaciju *commit*.

Privremeni radni prostor kreira se operacijom *git stash*. Ova operacija čuva promene u privremenom radnom prostoru, koji se u *Git* terminologiji naziva stek (eng. *stack*). Za ilustraciju korišćenja *git stash* operacije dat je sledeći primer.

Korisnik *nebojsa* na mašini *gitclient* kreira novi fajl pod nazivom *UI.java*, u kome piše kod grafičkog korisničkog interfejsa projekta na kome radi zajedno sa korisnikom *ivana*. Zbog novonastalih okolnosti, korisnik *nebojsa* mora da prekine rad na fajlu *UI.java*. S obzirom da posao nije završen, korisnik *nebojsa* ne može da izvrši operaciju *commit* i zbog toga korisnik dodaje fajl *UI.java* u svoju faznu oblasti i izvršava operaciju *git stash* i čuva do tada urađene promene fajla *UI.java* u privremenom stek prostoru.

Da bi proverio da li je operacijom *git stash* postigao ono što je želeo, korisnik *nebojsa* izvršava komandu *git status* i uverava se da se fajl *UI.java* više ne nalazi u njegovom glavnom radnom prostoru. *Nebojsa* zatim izvršava i komandu *git stash list*, koja prikazuje promene koje se trenutno nalaze na steku.

Opisani koraci prikazani su u Listingu 18-36.

Listing 18-36: *Izvršavanje git stash operacije*

```
nebojsa@gitclient:~/nebojsa_repo$ touch UI.java
nebojsa@gitclient:~/nebojsa_repo$ git add UI.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
A  UI.java
nebojsa@gitclient:~/nebojsa_repo$ git stash
Saved working directory and index state WIP on master: ed25e7d Added new code in
Hello.java file
nebojsa@gitclient:~/nebojsa_repo$ git status -s
nebojsa@gitclient:~/nebojsa_repo$ git stash list
stash@{0}: WIP on master: ed25e7d Added new code in Hello.java file
```

Nakon određenog vremena korisnik *nebojsa* može da nastavi rad na fajlu *UI.java*, i ovaj korisnik izvršava naredbu *git stash pop*, kojom se svi fajlovi sa steka vraćaju u radni direktorijum. Kada je završio rad na kodu grafičkog korisničkog interfejsa, korisnik *nebojsa* upisuje novu verziju fajla *UI.java* u svoj lokalni, a zatim i na udaljeni repozitorijum.

S druge strane, korisnik *ivana* sinhronizuje svoj lokalni repozitorijum sa udaljenim, da bi pogledala funkcionalnost grafičkog korisničkog interfejsa na kojoj je radio korisnik *nebojsa*.

Prikazane operacije date su u listinzima 18-37 i 18-38.

Listing 18-37: *Git stash pop naredba*

```
nebojsa@gitclient:~/nebojsa_repo$ git status -s
nebojsa@gitclient:~/nebojsa_repo$ git stash pop
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   UI.java

Dropped refs/stash@{0} (dc9b6d612f3661b1a9953b48cbacda158e0c4d4c)
nebojsa@gitclient:~/nebojsa_repo$ git status -s
A  UI.java
nebojsa@gitclient:~/nebojsa_repo$ git commit -m "Added file UI.java"
[master 72d357b] Added file UI.java
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 UI.java
nebojsa@gitclient:~/nebojsa_repo$ git push origin master
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Counting objects: 2, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 271 bytes | 271.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To 10.0.0.101:project.git
   ed25e7d..72d357b  master -> master
```

Listing 18-38: *Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 sa udaljenim repozitorijumom (git stash)*

```
ivana@gitclient2:~/project$ git pull
gituser@10.0.0.101's password:
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
From 10.0.0.101:project
   ed25e7d..72d357b  master     -> origin/master
Updating ed25e7d..72d357b
Fast-forward
 UI.java | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 UI.java
Current branch master is up to date.
ivana@gitclient2:~/project$ ls
Hello.java README Test.java UI.java
```

18.3.4. *Git mv* operacija

Operacija *git mv* (skraćeno od *move*) premešta željeni fajl na drugu lokaciju. Za demonstriranje ove operacije korišćen je jednostavan primer.

Korisnik *nebojsa* je odlučio da za potrebe razvoja grafičkog korisničkog interfejsa kreira poseban direktorijum pod nazivom *ui* na mašini *gitclient*. Korisnik zatim premešta fajl *UI.java* operacijom *git mv*, upisuje promene na svoj lokalni, a zatim i na udaljeni repozitorijum. Napomena: za ove potrebe ne koristi se operacija *git add*.

S druge strane, korisnik *ivana* na mašini *gitclient2* sinhronizuje svoj lokalni sa udaljenim repozitorijumom i vidi da je korisnik *nebojsa* kodove grafičkog korisničkog interfejsa premestio u drugi folder.

Navedeno je prikazano u listinzima 18-39 i 18-40.

Listing 18-39: *Demonstriranje git mv operacije za premeštanje fajlova*

```
nebojsa@gitclient:~/nebojsa_repo$ pwd
/home/nebojsa/nebojsa_repo
nebojsa@gitclient:~/nebojsa_repo$ mkdir ui
nebojsa@gitclient:~/nebojsa_repo$ git mv UI.java ui
nebojsa@gitclient:~/nebojsa_repo$ ls ui
UI.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
R UI.java -> ui/UI.java
nebojsa@gitclient:~/nebojsa_repo$ git commit -m "Modified directory structure
fr Graphical User Interface"
[master 90fd59f] Modified directory structure for Graphical User Interface
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename UI.java => ui/UI.java (100%)
nebojsa@gitclient:~/nebojsa_repo$ git push origin master
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 343 bytes | 343.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To 10.0.0.101:project.git
   72d357b..90fd59f  master -> master
```

Listing 18-40: *Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 sa udaljenim repozitorijumom (git mv operacija za premeštanje fajlova)*

```
ivana@gitclient2:~/project$ git pull
gituser@10.0.0.101's password:
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From 10.0.0.101:project
   72d357b..90fd59f  master     -> origin/master
Updating 72d357b..90fd59f
Fast-forward
 UI.java => ui/UI.java | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename UI.java => ui/UI.java (100%)
Current branch master is up to date.
ivana@gitclient2:~/project$ ls
Hello.java README Test.java ui
ivana@gitclient2:~/project$ ls ui
UI.java
ivana@gitclient2:~/project$ git log
commit 90fd59f0e6b41c8ad892475667e7a7c6874b0e57 (HEAD -> master, origin/master,
origin/HEAD)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:23:10 2018 +0200

    Modified directory structure for Graphical User Interface

commit 72d357bc667e08de57a4f05495f0732349adb478
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:16:02 2018 +0200

    Added file UI.java

commit ed25e7d5db5dfe1edbc420e0efa85d21ee06d95b
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sun Jul 15 06:29:39 2018 +0200

    Added new code in Hello.java file

commit dd73f11e85a664f7f562866268fc15cad7dc717e
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sat Jul 14 19:58:05 2018 +0200
```

Druga funkcija koju ima *git mv* operacija je promena naziva (eng. *rename*) fajlova. I za prikaz ove funkcionalnosti operacije *git mv* dat je jednostavan primer.

Nakon dodatnih promena na kodu u fajlu *UI.java*, korisnik *nebojsa* je zadovoljan svojim radom i menja naziv ovog fajla u *UI.class* primenom operacije *git mv* (napomena: prevođenje izvornog koda u *Java Byte* kod se ne vrši na ovakav način – ovaj primer je korišćen samo za potrebe ilustrovanja izvršavanja operacije *git mv*). Nakon toga, korisnik *nebojsa* sve promene zapisuje na lokalni, a zatim i na udaljeni repozitorijum. Korisnik *nebojsa* operaciju *commit* izvršava sa argumentom -a. Primenom ovog argumenta, *Git* automatski detektuje modifikovane fajlove i prebacuje ih u faznu oblast i u ovom slučaju nema potrebe za eksplicitnim izvršavanjem operacije *git add*. S druge strane, korisnik *ivana* sinhronizuje svoj lokalni sa udaljenim repozitorijumom na mašini *gitclient2*.

Navedeni koraci prikazani su u listinzima 18-41 i 18-42.

Listing 18-41: *Demonstriranje git mv operacije za preimenovanje fajlova*

```
nebojsa@gitclient:~/nebojsa_repo$ cd ui
nebojsa@gitclient:~/nebojsa_repo/ui$ ls
UI.java
nebojsa@gitclient:~/nebojsa_repo/ui$ git mv UI.java UI.class
nebojsa@gitclient:~/nebojsa_repo/ui$ git status -s
MR UI.java -> UI.class
nebojsa@gitclient:~/nebojsa_repo/ui$ git commit -a -m "Renamed and modified
file UI.java to UI.class"
[master 736490b] Renamed and modified file UI.java to UI.class
 2 files changed, 4 insertions(+)
 create mode 100644 ui/UI.class
 delete mode 100644 ui/UI.java
nebojsa@gitclient:~/nebojsa repo/ui$ git push origin master
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Counting objects: 4, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 346 bytes | 173.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0)
To 10.0.0.101:project.git
   90fd59f..736490b  master -> master
```

U Listingu 18-41 potrebno je da se obrati pažnja na još nešto. U rezultatima komande *git status -s*, pored naziva fajla *UI.java*, naznačeno je „MR", što znači da je fajl izmenjen i da je fajl promenio naziv.

Listing 18-42: *Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 sa udaljenim repozitorijumom (git mv operacija za preimenovanje fajlova)*

```
ivana@gitclient2:~/project$ git pull
gituser@10.0.0.101's password:
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From 10.0.0.101:project
   90fd59f..736490b  master     -> origin/master
Updating 90fd59f..736490b
Fast-forward
 ui/UI.class | 4 ++++
 ui/UI.java | 0
 1 files changed, 4 insertions(+)
 create mode 100644 ui/UI.class
 delete mode 100644 ui/UI.java
Current branch master is up to date.
ivana@gitclient2:~/project$ cd ui/
ivana@gitclient2:~/project/ui$ ls
UI.class
ivana@gitclient2:~/project/ui$ git log
commit 736490b0e9dcf5650e8eb76b241c3445d519fe1a (HEAD -> master, origin/master,
origin/HEAD)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:38:36 2018 +0200

    Renamed and modified file UI.java to UI.class

commit 90fd59f0e6b41c8ad892475667e7a7c6874b0e57
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:23:10 2018 +0200

    Modified directory structure for Graphical User Interface

commit 72d357bc667e08de57a4f05495f0732349adb478
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:16:02 2018 +0200

    Added file UI.java

commit ed25e7d5db5dfeledbc420e0efa85d21ee06d95b
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sun Jul 15 06:29:39 2018 +0200

    Added new code in Hello.java file

commit dd73f11e85a664f7f562866268fc15cad7dc717e
Author: Ivana Strumberger <istrumberger@singidunum.ac.rs>
Date:   Sat Jul 14 19:58:05 2018 +0200

    Added Hello.java

commit 3f33f1e49af32c6b0879795c697459d2e2cef4be
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 19:42:35 2018 +0200

    Added file Test.java

commit 0ad4e9c92ae8d8293d443cddca733cae59c89e6b
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sat Jul 14 08:51:57 2018 +0200

    Initial commit
```

18.3.5. *Git rm* operacija

Slično kao i u prethodnim slučajevima, operacija *git delete* demonstrirana je pomoću jednostavnog hipotetičkog scenarija.

Kada je korisnik *nebojsa* završio sa izradom grafičkog korisničkog interfejsa, kodove je iskopirao na drugu lokaciju i odlučio je da izbriše ovaj deo koda, kako iz svog lokalnog repozitorijuma, tako i iz udaljenog repozitorijuma. Za ove potrebe, korisnik je koristio *git rm* (*git remove*) operaciju. U isto vreme, na mašini *gitclient2*, korisnik *ivana* sinhronizuje svoj lokalni sa udaljenim repozitorijumom i vidi promene koje je korisnik *nebojsa* izvršio (listinzi 18-43 i 18-44).

Listing 18-43: *Demonstriranje git rm operacije za brisanje fajlova*

```
nebojsa@gitclient:~/nebojsa_repo/ui$ pwd
/home/nebojsa/nebojsa_repo/ui
nebojsa@gitclient:~/nebojsa_repo/ui$ git rm UI.class
rm 'ui/UI.class'
nebojsa@gitclient:~/nebojsa_repo/ui$ cd ..
nebojsa@gitclient:~/nebojsa_repo$ git rm ui
nebojsa@gitclient:~/nebojsa_repo$ git status -s
D  ui/UI.class
nebojsa@gitclient:~/nebojsa_repo$ git commit -a -m "Removed Java ByteCode -
Finished work"
[master c703a82] Removed Java ByteCode - Finished work
 1 file changed, 4 deletions(-)
 delete mode 100644 ui/UI.class
nebojsa@gitclient:~/nebojsa_repo$ git push origin master
Enter passphrase for key '/home/nebojsa/.ssh/id_rsa':
Counting objects: 2,  done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 251 bytes | 251.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To 10.0.0.101:project.git
   736490b..c703a82  master -> master
```

Listing 18-44: *Sinhronizovanje lokalnog repozitorijuma na mašini gitclient2 sa udaljenim repozitorijumom (git rm operacija za brisanje fajlova)*

```
ivana@gitclient2:~/project/ui$ git pull
gituser@10.0.0.101's password:
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
From 10.0.0.101:project
   736490b..c703a82  master     -> origin/master
Updating 736490b..c703a82
Fast-forward
 ui/UI.class | 4 ----
 1 file changed, 4 deletions(-)
 delete mode 100644 ui/UI.class
Current branch master is up to date.
ivana@gitclient2:~/project/ui$ cd ..
ivana@gitclient2:~/project$ ls
Hello.java README Test.java
ivana@gitclient2:~/project$ git log
commit c703a829af65740b0ee7be2770892347fe5dcbe4 (HEAD -> master, origin/master,
origin/HEAD)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:47:44 2018 +0200
    Removed Java ByteCode - Finished work
commit 736490b0e9dcf5650e8eb76b241c3445d519fe1a
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:38:36 2018 +0200

  Renamed and modified file UI.java to UI.class
commit 90fd59f0e6b41c8ad892475667e7a7c6874b0e57
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:23:10 2018 +0200

    Modified directory structure for Graphical User Interface

commit 72d357bc667e08de57a4f05495f0732349adb478
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:16:02 2018 +0200
```

U Listingu 18-43 potrebno je da se obrati pažnja na sledeće: u rezultatima komande *git status -s*, pored naziva fajla *ui/UI.class,* naznačeno je „D“, što znači da je fajl *UI.class* izbrisan zajedno sa direktorijumom *ui*.

18.3.6. Ispravljanje grešaka

U svakodnevnom korišćenju računara, obični korisnici, pa i programeri često prave greške. Zbog toga je prirodno da *VCS* alati podržavaju ugrađene mehanizme za ispravljanje grešaka. Ispravljanje grešaka u radu sa *Git* sistemom pokazano je na praktičnim primerima.

Naka korisnik *nebojsa* slučajno modifikuje fajl *Test.java*. Izvršavanjem komande *git status -s*, pre upisivanja promena u lokalni repozitorijum, vidi se da je fajl *Test.java* modifikovan. Korisnik *nebojsa* može da povrati stanje fajla na prethodnu verziju, pre upisivanja promena u lokalni repozitorijum, izvršavanjem komande *git checkout* (Listing 18-45).

Listing 18-45: *Prikaz komande git checkout za vraćanje fajla na prethodnu verziju*

```
nebojsa@gitclient:~/nebojsa_repo$ nano Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
 M Test.java
nebojsa@gitclient:~/nebojsa_repo$ git checkout Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
```

Komanda *git checkout* takođe može da se koristi i za povraćaj slučajno obrisanih fajlova. Neka korisnik *nebojsa* slučajno izbrše fajl *Test.java*. Čim korisnik izvrši operaciju *git rm* za brisanje fajlova, izbrisani fajl je odmah ušao u faznu oblast (bez izvršavanja operacije *git add*). Zbog toga je, pre povraćaja fajla operacijom *git checkout*, potrebno izvršiti komandu *git reset HEAD* da bi se *HEAD* pokazivač resetovao (Listing 18-46).

Listing 18-46: *Prikaz komande git checkout za povraćaj obranisanih fajlova*

```
nebojsa@gitclient:~/nebojsa_repo$ ls
Hello.java README Test.java
nebojsa@gitclient:~/nebojsa_repo$ git rm Test.java
rm 'Test.java'
nebojsa@gitclient:~/nebojsa repo$ git status -s
D  Test.java
nebojsa@gitclient:~/nebojsa_repo$ git reset HEAD Test.java
Unstaged changes after reset:
D       Test.java
nebojsa@gitclient:~/nebojsa_repo$ git checkout Test.java
nebojsa@gitclient:~/nebojsa_repo$ ls
Hello.java README  Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
```

Osim prikazane upotrebe, komanda *git reset HEAD* takođe može da se koristi i za poništavanje promena fajlova koji su dodati u faznu oblast. Neka korisnik *nebojsa* ponovo slučajno modifikuje fajl *Test.java* i neka izmenjeni fajl doda u svoju faznu oblast. U ovom slučaju, pre izvršavanja komande *git checkout*, kojom se poništavaju promene nad fajlom, korisnik *nebojsa* treba da resetuje *HEAD* pokazivač (Listing 18-47).

Listing 18-47: *Poništavanje promene nad fajlovima koji su dodati u faznu oblast*

```
nebojsa@gitclient:~/nebojsa_repo$ nano Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
M Test.java
nebojsa@gitclient:~/nebojsa_repo$ git add Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
M  Test.java
nebojsa@gitclient:~/nebojsa_repo$ git reset HEAD Test.java
Unstaged changes after reset:
M       Test.java
nebojsa@gitclient:~/nebojsa_repo$ git status -s
M Test.java
nebojsa@gitclient:~/nebojsa_repo$ git checkout Test.java
```

Rezultat komande *git status* pored naziva fajla *Test.java* ispisuje slovo „M“, što znači da je fajl modifikovan. Crveno slovo „M“ označava da je fajl modifikovan, ali da nije dodat u faznu oblast, dok zeleno slovo „M“ označava da je fajl modifikovan i da je dodat u faznu oblast (Listing 18-47).

18.3.7. Pomeranje *HEAD* pokazivača primenom git reset komande

Nakon izvršavanja određenih promena, korisnik može da odluči da te promene više ne želi. Za resetovanje ili poništavanje promena koristi se *git reset* komanda. Da bi se promena resetovala ili poništila potrebno je da se pomeri *HEAD* pokazivač.

Postoje tri tipa resetovanja: meko resetovanje (eng. *soft reset*), teško resetovanje (eng. *hard reset*) i kombinovano resetovanje (eng. *mixed reset*).

Svakoj razvojnoj grani programa u *Git*-u dodeljuje se *HEAD* pokazivač koji pokazuje na poslednju izvršenu operaciju *commit*. Ako je potrebno samo da se resetuje vrednost *HEAD* pokazivača, bez poništavanja bilo koje promene, onda se koristi *git reset* komanda sa argumentom *-soft*.

U fajlu *.git/ref/heads/master* čuva se *commit ID HEAD* pokazivača master razvojne grane. Izvršavanjem komande *git log* -1 može da se vidi trenutna vrednost ovog pokazivača (Listing 18-48).

Listing 18-48: *Prikaz commit ID poslednje promene*

```
nebojsa@gitclient:~/nebojsa_repo$ cat .git/refs/heads/master
c703a829af65740b0ee7be2770892347fe5dcbe4
nebojsa@gitclient:~/nebojsa_repo$ git log -1
commit c703a829af65740b0ee7be2770892347fe5dcbe4 (HEAD -> master, origin/master)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:47:44 2018 +0200

    Removed Java ByteCode - Finished work
```

Komanda *git log* -1 prikazuje samo poslednju promenu, tj. *commit ID* vrednost *HEAD* pokazivača. *Git log* se koristi i za prikazivanje *commit ID* nekoliko poslednjih promena (na primer *git log* -2, *git log* -3, itd.).

Komandom *git reset --soft HEAD~* resetuje se vrednost *HEAD* pokazivača na jednu *commit* operaciju unazad.

Primeri opisanih funkcionalnosti dati su u Listingu 18-49.

Listing 18-49: *Primeri upotrebe komandi git reset i git log*

```
nebojsa@gitclient:~/nebojsa_repo$ git log -2
commit c703a829af65740b0ee7be2770892347fe5dcbe4 (HEAD -> master, origin/master)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:47:44 2018 +0200

    Removed Java ByteCode - Finished work

commit 736490b0e9dcf5650e8eb76b241c3445d519fe1a
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:38:36 2018 +0200

    Renamed and modified file UI.java to UI.class
nebojsa@gitclient:~/nebojsa_repo$ git reset HEAD~
Unstaged changes after reset:
D       ui/UI.class
nebojsa@gitclient:~/nebojsa_repo$ git log -2
commit 736490b0e9dcf5650e8eb76b241c3445d519fe1a (HEAD -> master)
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:38:36 2018 +0200

    Renamed and modified file UI.java to UI.class

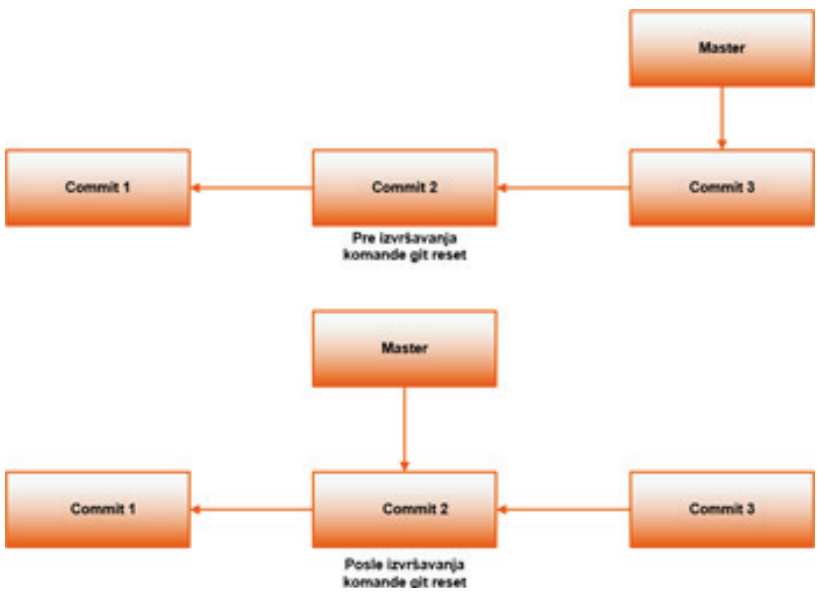
commit 90fd59f0e6b41c8ad892475667e7a7c6874b0e57
Author: Nebojsa Bacanin <nbacanin@singidunum.ac.rs>
Date:   Sun Jul 15 07:23:10 2018 +0200

    Modified directory structure for Graphical User Interface
```

Komandom *git reset HEAD* može da se resetuje *HEAD* pokazivač i na starije promene, gde se jednostavno naznači za koliko pozicija je potrebno izvršiti operaciju reset (na primer *git reset HEAD* ~3).

Grafički prikaz restovanja *HEAD* pokazivača izvršavanjem komande *git reset*, kada je *HEAD* pokazivač resetovan na jednu promenu (*commit*) unazad, dat je na Slici 18-6 .

Slika 18-6: *Resetovanje HEAD pokazivača*



Izvršavanje komande *git reset sa --soft* argumentom poništavaju se samo one promene iz fazne oblasti koje još uvek nisu izvršene (*commit*), dok promene koje su izvršene u radnom direktorijumu ostaju nepromenjene.

S druge strane, komanda *git reset sa --hard* argumentom briše sve iz fazne oblasti, resetuje *HEAD* pokazivač na poslednji *commit* i briše sve lokalne promene fajlova.

18.4. RAD SA *ONLINE* REPOZITORIJUMIMA

Na Internetu je dostupan veliki broj *Git online* repozitorijuma i platformi za razvoj softvera baziranih na *Git*-u. Jedna od najvećih klaud platformi za razvoj softvera je *GitHub* (*URL*: https://github.com/).

GitHub je *Web* bazirani servis u klaud okruženju za projekte razvoja softvera, koji u pozadini koristi *Git DVCS*. Za *GitHub* je razvijen veliki broj klijentskih aplikacija sa grafičkim korisničkim interfejsom za sve popularne operativne sistema, poput *Windows, Mac OS X* i *GNU/Linux*. Ove aplikacije omogućavaju interakciju sa *Git online* repozitorijumum pomoću intuitivnih i *user-friendly* alata grafičkog okruženja, a neke od popularnijih su *SmartGit, GitKraken, Fork* i *Magit*.

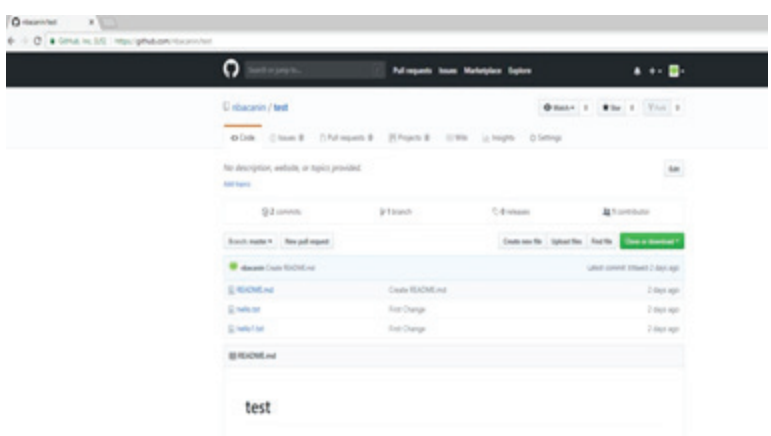
U ovoj praktičnoj simulaciji prikazana je interakcija sa online repozitorijumom *GitHub* platforme. Korisnik na mašini *gitclient* povezuje svoj lokalni repozitorijum sa udaljenim repozitorijumom koji se nalazi na *GitHub* serveru.

Za ove potrebe kreira se *GitHub* nalog na zvaničnom sajtu *GitHub* platforme (*URL*: https://github.com/). Tokom kreiranja naloga treba da se zapamti korisničko ime i lozinka.

Neka je korisnik *nebojsa* napravio *GitHub* nalog i neka je kreirao na *GitHub* platformi svoj javni repozitorijum pod nazivom *test*. Korisnik *nebojsa* je već dodao neke fajlove u svoj *GitHub* repozitorijum koje se nalazi na *URL* adresi: https://github.com/nbacanin/test (Slika 18-7).

U generisanju putanja (*URL* adresa) do repozitorijuma svojih korisnika, *GitHub* koristi sledeću sintaksu: *https://github.com/korisnicko_ime/naziv_repozitorijuma*.

Slika 18-7: *Screenshot – Online GitHub repozitorijum*



Korisnik *nebojsa* sada hoće da doda, tj. da replikuje svoj udaljeni *GitHub* repozitorijum na lokalnu mašinu *gitclient*. Za ove potrebe, korisnik *nebojsa* kreira novi direktorijum na svojoj lokalnoj mašini pod nazivom *github_repo* i na ovaj direktorijum klonira udaljeni *GitHub* repozitorijum.

Navedeni koraci su prikazani u Listingu 18-50.

Listing 18-50: *Kloniranje udaljenog GitHub repozitorijuma*

```
nebojsa@gitclient:~$ pwd
/home/nebojsa
nebojsa@gitclient:~$ mkdir github_repo
nebojsa@gitclient:~$ cd github_repo/
nebojsa@gitclient:~/github_repo$ git clone https://github.com/nbacanin/test.git
Cloning into 'test'...
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.
nebojsa@gitclient:~/github_repo$ ls
test
nebojsa@gitclient:~/github_repo$ ls test/
hello1.txt hello.txt  README.md
```

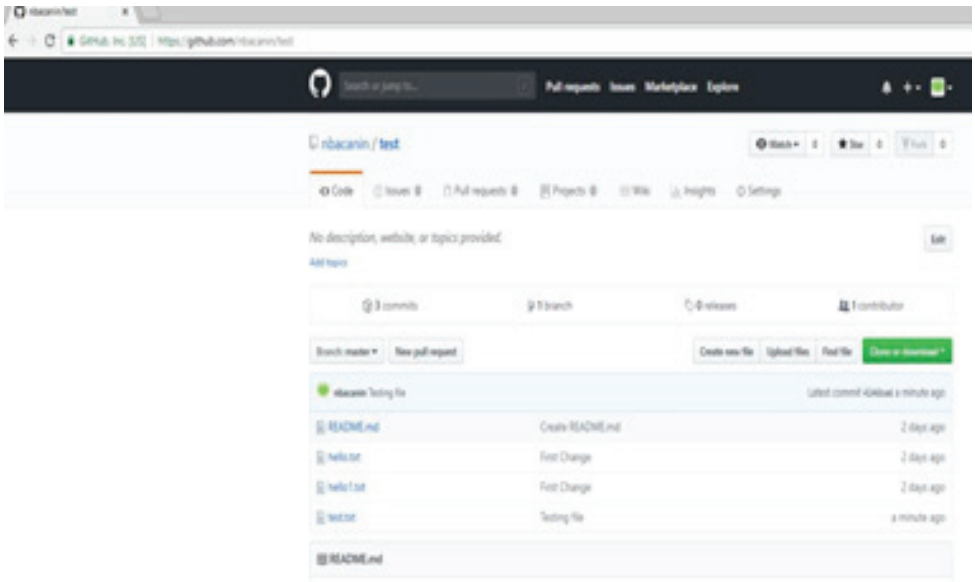
Kada je klonirao *GitHub* repozitorijum, korisnik *nebojsa* hoće da testira sinhronizaciju sa udaljenim repozitorijumom i za ove potrebe kreira probni fajl pod nazivom *test.txt* i snima sve promene sa lokalnog na *online GitHub* repozitorijum (Listing 18-51).

Listing 18-51: *Sinhronizacija sa GitHub repozitorijumom*

```
nebojsa@gitclient:~/github_repo/test$ touch test.txt
nebojsa@gitclient:~/github_repo/test$ git status -s
?? test.txt
nebojsa@gitclient:~/github_repo/test$ git add test.txt
nebojsa@gitclient:~/github_repo/test$ git commit -m "Testing file"
[master 4246ba6] Testing file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
nebojsa@gitclient:~/github_repo/test$ git status -s
nebojsa@gitclient:~/github_repo/test$ git push origin master
Username for 'https://github.com': nbacanin
Password for 'https://nbacanin@github.com':
Counting objects: 2, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 267 bytes | 267.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/nbacanin/test.git
   335be63..4246ba6 master -> master
```

Za svaki slučaj, korisnik *nebojsa* osvežava svoju *GitHub* stranu i u njegovom repozitorijumu se sada nalazi i fajl *test.txt,* koga je kreirao na mašini (lokalnom *Git* repozitorijumu) *gitclient*.

Slika 18-8: *Screenshot – Online GitHub repozitorijum (sinhronizacija)*



19. RASPOREĐIVANJE POSLOVA U KLAUD OKRUŽENJU

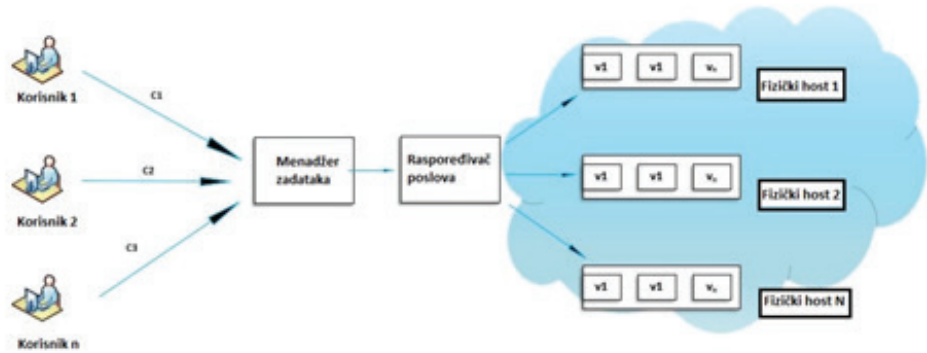
Raspoređivanje poslova spada u značajne aspekte klaud računarstva, gde je glavni cilj izvršavanje svih poslatih zadataka u što kraćem vremenskom periodu uz što manje korišćenje raspoloživih računarskih resursa.

Prilikom raspoređivanja poslova potrebno je da se odredi najbolji *resurs*, a kada je reč o klaud računarstvu, obično se pod resursom podrazumeva virtuelna mašina koja treba da izvrši svaki zahtev koji je poslao krajnji korisnik. Kako bi se navedeni ciljevi ostvarili, potrebno je da se primene *algoritmi za raspoređivanje poslova*.

Primenom algoritama za raspoređivanje poslova, kao i pametno upotrebljavanje resursa, postižu se poboljšanja u kvalitetima pružanja usluga (eng. *quality of service -QoS*), kao što su produktivnost, pouzdanost, korišćenje resursa, potrošnja električne energije, troškovi, itd.

Na Slici 19-1 je prikazan sistemski model raspoređivanja poslova u klaud okruženju. Model sistema raspoređivanja poslova se svodi na korisnike koji šalju zahteve, menadžer zadataka prosleđuje raspoređivaču poslova koji treba da, u zavisnosti od raspoloživih resursa, rasporedi zahteve na fizičkim hostovima koji imaju više virtuelnih mašina kako bi se u što kraćem vremenskom periodu izvršili svi poslati zadaci.

Slika 19-1: *Model sistema raspoređivanja poslova u klaud okruženju*



Algoritmi za raspoređivanje poslova kreiraju plan, koji ima za cilj da odredi na kojim resursima i kada će svaki posao (zadatak) biti izvršen. Prilikom primene algoritama, optimalna ili suboptimalna (približno optimalna) alokacija (mapiranje) raspoloživih resursa na zadatke koji su poslati, može da se ostvari u ograničenom vremenskom periodu, da bi se uspostavio određeni nivo kvaliteta usluga.

Osim procesa raspoređivanja poslova, bitan pojam koji takođe treba da se pomene, zbog značajnosti skupa aktivnosti upravljanja resursima (eng. *resource management*) jeste pojam obezbeđivanja resursa (eng. *resource provisioning*).

> *Obezbeđivanje resursa* u klaud okruženju odnosi se na određivanje i distribuciju virtuelizovanih računarskih resursa za krajnje korisnike, prilikom čega se teži da se uspostavi optimalni broj virtuelnih mašina sa odgovarajućim konfiguracijama i karakteristikama za mapiranje po zahtevu.

Proces obezbeđivanja resursa se deli na detekciju i izbor odgovarajućih resursa. Broj generisanih virtuelnih mašina treba da se svede na najmanji broj, vodeći računa o kvalitetu usluga krajnjih korisnika.

Naravno, kao i u svakoj mreži ili sistemu može doći do određenih problema. Efektivni algoritmi za obezbeđivanje resursa mogu da „pomognu" algoritmima za raspoređivanje poslova, analiziranjem i organizovanjem opterećenja klaud sistema, koje zavisi od tipa zahteva krajnjih korisnika. Dobre rezultate daje raspoređivanje na zahtev (eng. on-*demand scheduling*) u klasičnom modelu klaud računarstva, plaćanja na osnovu nivoa ostvarene potrošnje, prilikom čega krajnji

korisnici šalju heterogene zahteve, i klaud provajder treba da obezbedi pristup resursima u odgovarajućem vremenskom periodu. Ovaj zahtev takođe može da se definiše i kao online raspoređivanje (eng. *online scheduling*).

Problem kod ovakvog pristupa nastaje kada je distribucija opterećenja (što zavisi od tipa kao i broja zahteva krajnjih korisnika) na virtuelne mašine nejednaka (kada potrebe zahteva korisnika prevazilaze kapacitete virtuelnih mašina), gde dolazi do prekoračenja vremena za izvršavanje zadataka. U literaturi je ovaj problem poznat kao pitanje *balansiranja opterećenja*.

U scenarijima gde se primenjuje veći ili manji broj virtuelnih mašina u odnosu na realne potrebe za resursima, dolazi do problema. Kada je broj raspoloživih virtuelnih mašina manji od realnih zahteva poslatih zadataka, u literaturi je ovaj slučaj poznat kao scenario nedovoljnog obezbeđivanja (eng. *under provisioning*). Kada je broj raspoloživih virtuelnih mašina veći od potreba zahteva krajnjih korisnika, tada je prekomerno obezbeđivanje (eng. *over provisioning*).

U prvom slučaju kvalitet usluga opada, dok nezadovoljstvo krajnjih korisnika raste, dok u drugom slučaju dolazi do bespotrebnog rasta troškova korišćenih resursa.

Koncept *balansiranja opterećenja* teži da nadogradi primenjeni algoritam za raspoređivanje poslova i obezbeđivanje resursa, gde je važno da se uzme u obzir i *ravnopravna distribucija opterećenja* na raspoložive virtuelne mašine, kako bi se svaki zahtev korisnika izvršio na optimalan ili približno optimalan način.

U nastavku su prikazani algoritmi za raspoređivanje poslova, kao i modeli raspoređivanja resursa u klaud okruženju sa jednom funkcijom cilja kao i sa više funkcija cilja, koji se primenjuju u simulacijama.

19.1. ALGORITMI ZA RASPOREĐIVANJE POSLOVA

Algoritmi za raspoređivanje poslova se dele na statičke i dinamičke.

Kako bi se statički algoritam izvršio, potrebno je da se prikupe detaljne informacije za sve zadatke i virtuelne mašine, da bi se obezbedila adekvatna sredina za njihovo izvršavanje. Informacije koje su potrebne za zadatke obuhvataju: broj zadatka, dužina zadatka, zahtevi za resursima izračunavanja, skladištenja, mrežni resursi, rokovi za izvršavanje, itd. Osnovni podaci o virtuelnim mašinama su: raspoloživi kapaciteti jedinica za izračunavanje, memorija, skladište, mrežni propusni opseg, potrošnja električne energije, itd.

Zbog dinamike rada i heterogenih zahteva krajnjih korisnika, statički algoritmi ne daju zadovoljavajuće rezultate u klaud okruženju, zbog toga što nisu prilagođeni za optimizovanje rasporeda opterećenja na raspoložive resurse. Prilikom praktične eksploatacije statičkih algoritama, zaključeno je da ovi algoritmi negativno utiču na osnovne indikatore kvaliteta usluga kao što su makespan, pouzdanost i dostupnost. U statičke algoritme raspoređivanja spadaju svako sa svakim (eng. *round robin - RR*), prvi-u-prvi-van (eng. *first in first out - FIFO*) i najkraći posao prvi (eng. *shortest job first - SJF*).

Obzirom na nedostatke statičkih algoritama raspoređivanja, bolje je implemetirati dinamičke algoritme raspoređivanja. Dinamički algoritmi se zasnivaju na praćenju, dinamičkom usklađivanju i prilagođavanju izvršavanja zadataka u realnom vremenu. Ovakvi algoritmi neprestano prate promene u klaud okruženju i usklađuju opterećenje između različitih virtuelnih mašina u skladu sa uslovom preopterećenja (eng. *overloaded condition*).

Cilj je da se što više smanji disbalans koji može da nastane između čvorova (virtuelnih mašina). U dinamičke algoritme spadaju heterogeno najranije vreme završetka (eng. *heterogenous earliest finish time - HEFT*), heterogeno grupisanje sa umnožavanjem (eng. *clustering based heterogeneous with duplication - CBHD*) i algoritam najmanje ponderisane veze (eng. *weighted least connection - WLC*). Takođe, kao efikasan dinamički pristup za raspoređivanje poslova u klaud okruženju su se pokazale metaheuritstike intelgencije rojeva koje su detaljno objašnjene u narednim poglavljima ovog udžbenika.

Kako raste broj i sama kompleksnost zadataka za izvršavanje u klaud računarstvu, tako raste i potreba za adekvatnim algoritmima za raspoređivanje poslova i balansiranje opterećenja. Domen klaud računarstva ima veliki potencijal za unapređenja postojećih, kao i za kreiranje i implementaciju novih algoritama. Prioritet u klaud okruženju treba da bude efikasno i efektivno raspoređivanje poslova, kako za krajnje korisnike tako i za one koji pružaju usluge.

19.2. POJAM OPTIMIZACIJE

Optimizacija je proces kojim se pronalazi minimum ili maksimum neke funkcije.

Optimizacija je našla široku primenu u svakodnevnom životu, zato što je za mnoge probleme i izazove iz realnog sveta potrebno da se nađu što bolja rešenja.

Naučna grana koja se bavi razvojem modela i metoda za rešavanje matematički formulisanih problema optimizacije naziva se teorija optimizacije (eng. *optimization theory*). Generalizacija teorije optimizacije i tehnike spada u širu oblast koja se naziva primenjena matematika (eng. *applied mathematics*).

Glavni cilj postupka optimizacije je pronalaženje optimalnog ili suboptimalnog (približno optimalnog) rešenja u odnosu na definisane ciljeve i potencijalna ograničenja.

Proces optimizacije može da se podeli na sledeće faze, koje se izvršavanju sekvencijalno:

- definisanje problema,
- formulisanje i rešavanje matematičkog modela problema,
- implementacija i
- validacija rešenja.

Funkcija cilja (eng. *objective function*) predstavlja kvantitativnu meru performansi optimizacionih problema i kako bi proces optimizacije mogao da počne, neophodno je identifikovati i formulisati ovu funkciju.

Funkcija cilja omogućava da se svakom vektoru (rešenju) pridruži odgovarajuća vrednost koja predstavlja njegovu meru kvaliteta. Formulacija funkcije cilja zavisi od optimizacionog problema koji je potrebno rešiti.

Karakteristike optimizacionog problema nazivaju se *promenljive* ili *varijable*. Potrebno je pronaći vrednosti promenljivih koje optimizuju formulisanu funkciju cilja.

Termin koji je u tesnoj vezi sa optimizacijom je *modeliranje*, koji se definiše kao proces kojim se utvrđuje funkcija cilja, varijable i potencijalna ograničenja postavljenog optimizacionog problema.

U formulisanju modela za praktične probleme, treba voditi računa o njegovoj složenosti. U prvom slučaju, ukoliko je formulisani model previše jednostavan, on neće moći realno da modelira sva svojstva problema na koji se odnosi. S druge strane, u slučaju previše kompleksnog modela, ni jedna optimizaciona metoda neće moći uspešno da ga reši.

Često se u formulacijama modela problema iz realnog okruženja koriste aproksimacije, ali na način da se zadrže bitna svojstva datog problema. Odgovarajuća optimizaciona metoda primenjuje se nakon što se model formuliše.

U procesu rešavanja optimizacionog problema, vrši se izbor rešenja koja *maksimizuje ili minimizuje funkciju cilja i zadovoljava sva ograničenja*. Kada se govori o optimizaciji, u literaturi se obično razmatra minimizacija.

Problem optimizacije za specijalni slučaj Rn može da se matematički formuliše na sledeći način:

$$\text{za dato } f: R^n \rightarrow R$$

$$\text{pronaći } x * \in R^n, \text{ uz zadovoljenje uslova,}$$

$$\forall x \in R^n, f(x^*) \leq f(x)$$

gde R^n označava prostor varijabli (prostor pretrage) funkcije f, notacija $x \in R^n$ označava potencijalno rešenje problema, dok je optimalno rešenje označeno sa x^*. Promenljiva n se odnosi na veličinu prostora pretrage u smislu broja dimenzija optimizacionog problema i u ovoj formulaciji predstavlja broj varijabli funkcije cilja.

Problemi optimizacije mogu da se podele na probleme sa *jednom funkcijom cilja* (eng. *single-objective optimization*) i na probleme sa *više funkcija cilja* (eng. *multi-objective optimization - MOO*).

Optimizacija sa jednom funkcijom cilja poznata je u literaturi i kao *jednokriterijumska optimizacija*, dok se za optimizaciju sa više funkcija cilja koriste termini višekririjumska optimizacija i više-ciljno programiranje.

U slučaju problema sa više funkcija cilja, postoji više ciljeva koje je potrebno simultano optimizovati, gde su ciljevi najčešće kontradiktorni (u suprotnosti) jedni drugima. Na primer, sa poboljšanjem jednog cilja, drugi cilj se pogoršava i obrnuto.

Problemi višekriterijumske optimizacije se često sreću u svakodnevnom životu u raznim oblastima, poput menadžmenta, finansija i primenjenog računarstva. Tako na primer, u slučaju optimizacije finansijskog portfolija potrebno je uvećati prinos na finansijsku aktivu uz minimizaciju rizika. U domenu bežičnih senzorskih mreža potrebno je simultano ostvariti najveću pokrivenost senzorima uz minimizaciju potrošnje električne energije.

19.3. MODEL RASPOREĐIVANJA RESURSA SA JEDNOM FUNKCIJOM CILJA

Prvi model raspoređivanja resursa na klaudu spada u grupu modela sa *jednom funkcijom cilja*.

Da bi se prikazala matematička formulacija modela raspoređivanja sa jednom funkcijom cilja potrebno je da se izdvoje sledeće notacije. Neka *CI* označava klaud infrastrukturu koja se sa sastoji iz N_{ph} fizičkih računara (*PH*), gde se na svakom *hostu* (fizičkom računaru) izvršava N_{vm} virtuelnih mašina (*VM*):

$$CI = \{PH1, PH2, ..., PH_i, ..., PHN_{ph}\},$$

gde svaki PH_i ($i = 1, 2, 3, ..., N_{ph}$) može da se definiše kao:

$$PH_i = \{VM_1, VM_2, ..., VM_j, ..., VMN_{vm}\},$$

gde VM_j označava *j*-tu virtuelnu mašinu koja se alocira u okviru određenog *host* računara (čvor). Performanse svake VM_j virtuelne mašine se dalje preciznije određuju sledećim karakteristikama (atributima, svojstvima):

$$VM_j = \{VMID_j, MIPS_j\},$$

gde $VMID_j$ i $MIPS_j$ označavaju jedinstveni identifikator (eng. *unique identifier*), tj. serijski broj (eng. *serial number*) i performanse obrade obrada u MIPS jedinicama virtuelne mašine VM_j, respektivno.

Korisnici šalju skup zadataka (*TSK*) za izvršavanje u klaud okruženje, koji se dalje dodeljuju pomoću algoritma raspoređivanja raspoloživim i odgovarajućim virtuelnim mašinama:

$$TSK = \{T_1, T_2, T_3, ..., Tk, ..., T_{Ntsk}\},$$

gde N_{tsk} označava ukupan broj zadataka koji su poslati na klaud, dok notacija T_k predstavlja *k*-ti zadatak, koji se dalje preciznije određuje kao:

$$T_k = \{TID_k, length_k, ETC_k, P_k\},$$

gde TID_k i $length_k$ označavaju jedinstveni identifikator i dužinu zadatka k, koja se izražava u jedinici milioni instrukcija (eng. *million instructions - MI*).

Alokacija (mapiranje) skupa od *TSK* zadataka na N_{vm} virtuelnih mašina ima veliki uticaj na ukupne performanse klaud sistema.

Očekivano vreme završetka (eng. *expected time to complete*) zadatka k na j-toj virtuelnoj mašini ETC_{kj} može da se izračuna na sledeći način:

$$ETC_{kj} = \frac{legth_g}{MIPS_j}$$

gde je $j = 1, ..., N_{vm}$ i $k = 1, ..., N_{tsk}$.

Matricom *ETC* očekivanog vremena završetka svih zadataka veličine $N_{tskx}N_{vm}$ definiše se koiliko je vremena potrebeno kako bi se svaki zadatak izvršio na svakoj virtuelnoj mašini koja je raspoloživa u klaud sistemu:

$$ETC = \begin{bmatrix} ETC_{1,1} & ETC_{1,2} & ETC_{1,3} & \cdots & ETC_{1,j} & \cdots & ETC_{1,N_{vm}} \\ ETC_{2,1} & ETC_{2,2} & ETC_{2,3} & \cdots & ETC_{2,j} & \cdots & ETC_{2,N_{vm}} \\ ETC_{3,1} & ETC_{3,2} & ETC_{3,3} & \cdots & ETC_{3,j} & \cdots & ETC_{3,N_{vm}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ ETC_{N_{tsk},1} & ETC_{N_{tsk},2} & ETC_{N_{tsk},3} & \cdots & ETC_{N_{tsk},j} & \cdots & ETC_{N_{tsk},N_{vm}} \end{bmatrix}$$

Prikazana matrica se bavi raspoređivanjem poslova u klaud računarstvu i smatra se kao standardni način za modeliranje i izvršavanje poslova u klaud infrastrukturi sa heterogenim zahtevima i resursima.

Kako bi se formulisala funkcija cilja minimizovanja makspan indikatora, najpre je potrebno da se izračuna koliko je vremena potrebno da svaka virtuelna mašina obradi sve zahteve koji su joj poslati (eng. *execution time - ET*).

Vreme izvršavanja virtuelne mašine j za zadatak k zavisi od promenljive odluke (eng. *decision variable*) x_{kj}

$$x_{k,j} = \begin{cases} 1, \text{ if } C_k \text{ is allocated to } VM_j, \\ 0, \text{ if } C_k \text{ is not allocated to } VM_j. \end{cases}$$

Zatim, ET_j , gde je j u rasponu [1, N_{vm}], može da se izračuna na sledeći način:

$$ET_j = \sum_{k=1}^{N_{tsk}} x_{k,j} \cdot ETC_{k,j}.$$

Funkcija cilja minimizacija makespan vrednosti (*MS*) predstavlja najveću (maksimalnu) vrednost *ET* svih virtuelnih mašina:

$$MS = \max[ET_j]_{j=1}^{N_{vm}},$$

$\forall k \in [1, N_{tsk}]$ mapirano na jth VM, $j = 1, 2, 3, ..., N_{vm}$.

Kako bi se preciznije odredila evaluacija performansi predloženih optimizacionih algoritama, potrebno je da se uzme u obzir i *indicator stepena disbalansa* (*DI*), koji se računa na sledeći način:

$$DI = \frac{T_{max} - T_{min}}{T_{avg}},$$

gde T_{max}, T_{min} *i* T_{avg} označavaju najveće, najmanje i prosečno vreme izvršavanja svih virtuelnih mašina, respektivno.

19.4. MODEL RASPOREĐIVANJA RESURSA SA VIŠE FUNKCIJA CILJA

U modelu raaspoređivanja resursa sa više funkcija cilja, uzeta je u obzir *minimizacija makespan indikatora*, kao i *minimizacija ukupnih troškova raspoloživog budžeta* (eng. *budget cost objective*).

U literaturi je ovaj model poznat i kao model raspoređivanja resursa baziran na performansama i ograničenjima raspoloživog budžeta (eng. *model based on the performance and budget constraints*). Performanse se određuju sa makespan indikatorom, uz pravilo da se svi zahtevi završe pre unapred zadatog roka, kao i da troškovi izvršavanja zahteva budu u okviru svih budžetskih ograničenja.

U sklopu ovog modela uzimaju se troškovi koji se odnose na troškove izvršavanja virtuelnih mašina. Određuju se troškovima procesorskog vremena (eng. *central processing unit - CPU cost*) i memorije (eng. *memory costs*). Ukupni troškovi budžeta izračunavaju se kombinovanjem navedene dve kategorije troškova. Podobnost svakog rešenja izračunava se sabiranjem vrednosti ukupnog vremena izvršavanja svih zadataka (*makespan*) i *ukupnih troškova budžeta*.

Svako rešenje treba da ima što manju vrednost ukupnog vremena izvršavanja odn. makespan indikatora i funkcije koja modelira troškove budžeta. Prvo se računaju troškovi CPU vremena virtuelne mašine j pomoću sledećeg izraza:

$$C_{(x)} = \sum_{j=1}^{N_{VM}} C^{cost}(j),$$

gde $C^{cost}(j)$ označava troškove procesora virtuelne mašine VM_j, dok N_{vm} predstavlja ukupan broj virtuelnih mašina u klaud infrastrukturi. Notacija x označava izvodljivo (eng. *feasible*) rešenje. Vrednost $C^{cost}(j)$ se dalje izračunava na sledeći način:

$$C^{cost}(j) = C_{base} \times C_j \times t_{ij} + C_{Trans},$$

gde C_{base} označava osnovu troškova, C_j predstavlja troškove CPU vremena virtuelne mašine VM_j, dok se t_{ij} odnosi na vreme tokom kojeg se zadatak T_i izvršava na resursu R_j (virtuelnoj mašini VM_j). Osim navednog, notacija C_{Trans} označava troškove prenosa procesora (CPU). Osnovni troškovi C_{base} i C_{Trans} su konstantne vrednosti:

$$C_{base} = 0.17/hr,$$
$$C_{Trans} = 0.005,$$

Za izračunavanje funkcije troškova memorije, koristi se sledeći izraz:

$$M_{(x)} = \sum_{j=1}^{N_{VM}} M^{cost}(j),$$

gde $M^{cost}(j)$ označava trošak memorije virtuelne mašine VM_j i N_{VM} označava ukupan broj virtuelnih mašina. Dalje, $M^{cost}(j)$ se određuje kao:

$$M^{cost}(j) = M_{base} \times M_j \times t_{ij} + M_{Trans},$$

gde se osnovni trošak memorije označava kao M_{base}, memorije virtuelne mašine VM_j se reprezentuje kao M_j, dok t_{ij} označava vreme koje je potrebno da bi se zadatak T_i obradio na resursu R_j (virtuelnoj mašini VM_j). Oznaka M_{Trans} predstavlja trošak prenosa memorije. Vrednosti M_{base} i M_{Trans} su konstante:

$$M_{base} = 0.05\text{GB}/hr,$$
$$M_{Trans} = 0.5$$

Konačno, ukupni trošak budžeta se izračunava sabiranjem troškova procesora i memorije svake virtuelne mašine:

$$B_{(x)} = C_{(x)} + M_{(x)},$$

gde $B_{(x)}$ označava ukupnu funkciju troškova budžeta krajnjeg korisnika, dok $C_{(x)}$ i $M_{(x)}$ označavaju funkcije troškova procesora i memorije, respektivno.

Uzimajući u obzir i vreme izvršavanja i troškove budžeta, podobnost se određuje na sledeći način:

$$H_{(x)} = MS_{(x)} + B_{(x)},$$

gde $MS_{(x)}$ označava ukupno vreme izvršavanja svih zadataka (makespan), što u ovom modelu predstavlja funkciju performansi sistema, koje treba da bude manje ili jednako predefinisanom roku za izvršavanje zadatka.

Vrednost *MS* se izračunava kao:

$$MS_{(x)} \leq \sum_{i=1}^{|T|} D_i,$$

gde notacija D_i označava rok za izvršavanje zadatka T_i. U prethodnoj jednačini, $B_{(x)}$ predstavlja funkciju budžeta zadatka, koja obuhvata troškove procesora i memorije i koja treba da bude manja ili jednaka raspoloživom budžetu krajnjih korisnika.

Na osnovu toga, ograničenja funkcije troškova budžeta $B_{(x)}$ mogu da se modeluju kao:

$$B_{(x)} \leq \sum_{i=1}^{|T|} B_i,$$

gde B_i reprezentuje trošak budžeta krajnjeg korisnika zadatka T_i.



PITANJA ZA RAZMATRANJE

1. Objasniti pojam raspoređivanja poslova u klaud okruženju.
2. Čemu služe algoritmi za raspoređivanje poslova?
3. Prikazati sistemski model raspoređivanja poslova u klaud okruženju.
4. Objasniti pojam obezbeđivanje resursa u klaudu.
5. Šta je online scheduling?
6. Koji problemi mogu da nastanu prilikom primena algoritama za raspoređivenje zahteva?
7. Koja je razlika između under provisioning i over provisioning? Kako rešiti takve probleme u klaud sistemu? Dati primere.
8. Šta je balansiranje opterećenja u klaud okruženju?
9. Koja je razlika između koncepta obezbeđivanja resursa i balansiranja opterećenja?
10. Šta su statički algoritmi za raspoređivanje poslova?
11. Šta su dinamički algoritmi za raspoređivanje poslova?
12. Izdvojiti razlike između ova dva tipa algoritama.
13. Šta je optimizacija?
14. Koji je glavni cilj postupka optimizacije?
15. Definisati pojam funkcije cilja.
16. Šta su karakteristike optimizacionog problema?
17. Šta znači maksimizovanje funkcije cilja ili minimizovanje funkcije cilja?
18. Napisati matematičku formulaciju za problem optimizacije za specijalni slučaj Rn. Objasniti notacije.
19. Objasniti model raspoređivanja resursa sa jednom funkcijom cilja. Prikazati sve notacije i objasniti korake.
20. U modelu raspoređivanja resursa sa jednom funkcijom cilja prikazati funkciju cilja. Da li se minimizuje ili maksimizuje? Objasniti.
21. Kako se prikazuju virtuelne mašine u modelu raspoređivanja resursa sa jednom funkcijom cilja?
22. Kako se prikazuju zadaci koji se šalju u klaud-u?
23. Objasniti model raspoređivanja resursa sa više funkcija cilja. Prikazati notacije i objasniti korake.
24. Navedi glavne ciljeve modela raspoređivanja resursa sa više funkcija cilja.
25. Šta predstavlja podobnost?
26. Kako se vrednost makespan izračunava?
27. Kako se trošak budžeta izračunava?

20. OPTIMIZACIONI ALGORITMI

U literaturi se može naći veliki broj taksonomija *optimizacionih algoritama*, koje algoritme klasifikuju na osnovu raznih kriterijuma. Koji god da je kriterijum u pitanju, cilj svakog algoritma jeste da za zadati optimizacioni problem da najbolje rezultate. Često se u implementacijama primenjuju i kombinacije različitih algoritama.

Prva klasifikacija može da se istakne na osnovu kriterijuma da li se u procesu pretrage koriste, ili se ne koriste informacije o izvodu funkcije koja se optimizuje, optimizacioni algoritmi se dele na one koji *koriste izvod* (eng. *derivate-based algorithms*) i na one koji *ne koriste izvod* (eng. *derivate-free methods*).

Neki od poznatijih predstavnika prve grupe su Gauss-Newton metod i metoda najbržeg spuštanja (eng. *steepest descent method*), dok se kao jedan od više korišćenih predstavnika algoritama koji ne koriste informacije o izvodu funkcije navodi Nelder-Mead simpleks metoda spuštanja niz brdo (eng. *Nelder-Mead downhill simplex method*).

Druga klasifikacija se odnosi na kriterijum mobilnosti, gde optimizacioni algoritmi mogu da se svrstaju u grupu algoritama *lokalne* ili *globalne* pretrage.

Lokalni algoritmi u većini slučajeva konvergiraju ka lokalnom optimumu a retko ka globalnom optimumu, i samim tim nisu sposobni da izađu iz domena prostora pretrage gde se nalazi suboptimalno rešenje. Neki od predstavnika optimizacionih algoritama lokalne pretrage navodi se jednostavni algoritam penjanja uz brdo (eng. *simple hill-climbing algorithm*).

U slučaju problema globalne optimizacije, gde postoji više lokalnih i jedan ili više globalnih optimuma, potrebno je koristiti algoritme *globalne pretrage*. Jedna od važnijih karakteristika globalnih algoritama je *slučajnost* (eng. *randomization*).

Treća najčešće korišćena taksonomija optimizacionih algoritama deli algoritme na *determinističke* (eng. *deterministic*) i na *stohastičke* (eng. *stochastic*).

Druga podela koja je slična navedenoj, koristi malo drugačiju terminologiju, gde sve optimizacione algoritme razvrstava na *egzaktne* i na *heurističke*, gde se heuristike dalje razvrstavaju u kategorije *heuristika za specifične probleme* i na *metaheuristike*.

Deterministički algoritmi pod bilo kakvim uslovima i pri svakom izvršavanju na osnovu istih ulaza uvek generišu iste izlaze s obzirom da uvek slede isti niz instrukcija odnosno naredbi. Nekada su deterministički algoritmi dovoljni za rešavanje određenih problema, ali za problem kod kojih veza između kandidat rešenja (potencijalno najbolje rešenje problema) i njihove podobnosti (eng. *fitness*) nisu očigledne, kada je broj dimenzija prostora pretrage veliki, kada se vrši optimizacija složenijih funkcija gde postoji više lokalnih optimuma, deterministički algoritmi nisu adekvatni niti sposobni da generišu zadovoljavajuće približno optimalno rešenje obzirom da bi za njihovo izvršavanje bilo potrebno neprihvatljivo dugo vremena.

Determinističke metode su prvi put počeke da se primenjuju za rešavanje problema globalne optimizacije 50-tih godina 20. veka, po principu "*podeli, pa vladaj*"(eng. *divide-and-conquer*). Sa ovakvim metodamase glavni problem deli na više manjih problema, na osnovu čega se znatno smanjuje njihova složenost. Na ovaj način, smanjivanjem struktura problema, ove metode se u procesu istraživanja domena pretrage oslanjaju na intenzivna računarska izračunavanja.

Jedan od primera determinističkih metoda iz ove grupe je algoritam *grananja i ograničavanja* (eng. *branch and bound - BB*), koji glavni problem deli na grane (podprobleme). BB algoritmi se i danas široko primenjuju u raznim domenima optimizacije, poput bežičnih senzorskih mreža, energetskih sistema, više-kriterijumske optimizacije, kao i u domenu ostalih praktičnih problema.

Kada je reč o *stohastičkim optimizacionim metodama*, one se baziraju na slučajnosti. Stohastički algoritmi u svakom koraku izračunavanja biraju sledeći korak pod uticajem pseudo-slučajnog elementa. Da bi se preciznije objasnilo, stohastički algoritam u svakom izvršavanju (puštanju) prati različite putanje. Pseudo-slučajne sekvence se biraju primenom generatora.

Neke od metoda stohastičkih optimizacionih algoritama su metod *Monte Karlo* kao i algoritmi koji su nastali iz ove metode, evolutivni *algoritmi, metaheuristike inteligencije rojeva*.

20.1. METAHEURISTIKE

Heuristike (od starogrčke reči *heuriskein*), predstavljaju grupu empirijskih, iskustvenih metoda za rešavanje problema. Tokom procesa pretrage, heurističke metode pokušavaju da pronađu što bolje rešenje, ali ne mogu da garantuju da će pronađeno rešenje biti najbolje (optimalno).

Nastanak i razvoj heurističkih metoda potiče od nesposobnosti determinističkih algoritama da u odgovarajućem vremenskom periodu generišu optimalno ili suboptimalno zadovoljavajuće rešenje. Neki od dodatnih argumenata za njihov razvoj mogu da se istaknu:

- za određene probleme nisu dostupne tehnike koje mogu da optimizuju,
- nemogućnosti algoritama za primenu na hardverskim resursima,
- značajna fleksibilnost heuristika u odnosu na determinističke metode u smislu mogućnosti modelovanja dodatnih svojstava problema koji se optimizuje i
- prostor za inkorporiranje u metodu pretrage globalnog optimuma određenog problema.

Dobar heuristički algoritam treba da sadrži:

- generisanje rešenja u prihvatljivom vremenskom periodu,
- maksimizovanje verovatnoće dobijanja rešenja koje je blizu optimalnom i
- minimizovanje verovatnoće generisanja lošeg rešenja.

Zbog efikasnih rezultata koje daju heurističke metode za rešavanje raznih praktičnih problema, poraslo je interesovanje za njihov razvoj i unapređenja, gde su kao rezultat nastale *metaheuristike*, odnosno *metaheuristički algoritmi* (eng. *metaheuristic algorithms*).

U nazivu metaheuristika, prefiks meta predstavlja grčku reč koja označava *metodologiju višeg nivoa*.

Metaheuristike obuhvataju generičke skupove pravila koji se primenjuju na rešavanje velikog broja optimizacionih problema. Oslanjajući se na principe opštih optimizacionih algoritama, metaheuristike teže da u iteracijama poprave trenutno rešenje.

Metaheuristika može da se definiše kao skup algoritamskih koncepata koji se primenjuju za definisanje heurističkih metoda koje se primenjuju za veliki broj problema.

Metaheurističke metode i algoritmi se primenjuju u mnogim oblastima, kao što su finansije, fizika, hemija, biologija, medicina, računarstvo itd.

Metod za pronalaženje odgovarajuće heuristike za specifičan problem predstavlja *metaheuristiku*.

Prilikom izbora najbolje heuristike za specifičan problem, metaheuristički pristupi pokušavaju da odgovore na sledeće zahteve:

- koje vrednosti parametara daju dobre rezultate kada se primeni heuristika *x* na problem *p*,
- na koji način je moguće prilagoditi parametre heuristike *x* da bi se dobila bolja rešenja problema *p* i
- da li je za rešavanje problema *p* bolje primeniti heuristiku *x* ili heuristiku *y*.

Kako bi metaheuristička metoda bila što efikasnija za rešavanje problema, potrebno je da sadrži sledeće karakteristike:

- jednostavnost. Metaheuristike treba da se baziraju na intuitivnim i lako razumljivim principima,
- preciznost. Svi koraci koje izvršava metaheuristički algoritam treba da budu precizno formulisani matematičkim pojmovima i terminima,
- doslednost. Sve faze u procesu primene algoritma treba da budu konzistentne sa opštim pravilima u skladu sa kojima je metaheuristika formulisana,
- efikasnost. Kada se metaheuristika primeni na konkretan problem iz realnog okruženja treba da generiše rešenja koja su bliska optimalnom.

Ovo se posebno odnosi na slučaj kada se metaheuristika primeni na *zvanične test primere* (eng. *benchmark problems*),

- efektivnost. Za svaki konkretan problem algoritam treba da ostvari optimalno, ili makar zadovoljavajuće rešenje u razumnom vremenskom periodu i
- univerzalnost. Metaheuristička metoda treba da se bazira na što opštijim osnovnim principima i pravilima, kako bi mogla da uz, minimalne modifikacije, primeni i na druge probleme.

Postoji više podela metaheurističkih algoritama, ali prema jednoj od najsveobuhvatnijih, metaheuristike se dele na:

- determinističke i stohastičke,
- one koje se baziraju na populaciji rešenja (eng. *population-based*) i koje se baziraju na jednom rešenju (eng. *single solution-based*),
- one koje koriste memoriju (eng. *memory-based*) i na one koje ne koriste memoriju (eng. *memory-less*) i
- metaheuristike koje su inspirisane prirodom (eng. *nature-inspired*) i metaheuristike koje nisu inspirisane prirodom (eng. *non nature-inspired*).

U nastavku udžbenika su objašnjene metaheuristike koje su inspirisane prirodom, kao i metaheuristike inteligencije rojeva koje su primenjene u simulacijama za potrebe rešavanja problema raspoređivanja poslova i balansiranje opterećenja u klaud računarstvu.

Kod *metaheuristika koje su inspirisane prirodom*, mehanizam kojim se usmerava i vodi proces pretrage inspirisan je prirodnim fenomenima. Primeri prirodnih fenomena su proces evolucije živih organizama, kao i kolektivna inteligencija populacije individua u prirodi, poput jata ptica, kolonije mrava, kolonije pčela, grupa slepih miševa, krdo slonova, itd.

U slučaju *metaheuristika koje nisu inspirisane prirodom*, proces pretrage simulira proces čije poreklo ne potiče iz prirode.

20.2. METAHEURISTIKE KOJE SU INSPIRISANE PRIRODOM

Kod metaheuristika koje su inspirisane prirodom, mehanizam koji se pokreće za proces pretrage simulira *prirodni fenomen*. Pod prirodnim fenomenom se podrazumeva neki proces *biološke evolucije*, ili *grupa organizama iz prirode*. Koji god da je proces u pitanju, članovi grupe ili individue komuniciraju na indirektan način međusobno i tako ispoljavaju ono što je bitno za ovaj tip metaheuristika, a to je *inteligentno ponašanje*.

U zavisnosti od tipa prirodnog fenomena kojeg simuliraju, metaheuristike koje su inspirisane prirodom mogu da se kategorizuju u dve grupe.

Prva grupa se odnosi na evolutivne algoritme (eng. *evolutionary algorithms - EA*), koji modeliraju proces biološke evolucije. Druga grupa simulira kooperativno inteligentno ponašanje grupe individua u prirodi, i to su metaheuristike inteligencije rojeva (eng. *swarm intelligence*).

Nastanak evolutivnih algoritama se može videti u zakonima nasleđivanja koje je definisao Gregor Mandel i u teoriji evolucije, čiji je tvorac Čarls Darvin. Zaključeno je da prirodne vrste tokom vremena evoluiraju, gde se ukrštanjem roditelja dobijaju potomci, koji su osim što nasleđuju karakteristike (eng. *traits*) roditelja, podvrgnuti i *slučajnoj mutaciji*.

Individue se tokom vremena adaptiraju (prilagođavaju) okolini u kojoj žive, gde dolazi do promene njihovog genetskog materijala. Primenom mehanizma prirodne selekcije, samo najjače individue opstaju i generišu potomke. Na taj način, iz generacije u generaciju, jedinke postaju sve robusnije i kvalitetnije.

Algoritmi koji pripadaju grupi evolutivnih algoritama su *populacioni*. Kod populacionih algoritama se process pretrage izvršava modeliranjem grupe individua. Takođe su i *iterativni*, što znači da u uzastopnim iteracijama generišu potencijalno bolja rešenja primenom osnovnih genetskih operatora ukrštanja (eng. *crossover*) i mutacije (eng. *mutation*). Individue koje se ukrštaju (roditelji) će imati jednog ili više potomaka, što zavisi od samog kvaliteta individua i od samog algoritma.

Pseudo-kod jednostavnog EA dat je u Algoritmu 1.

Algoritam 1 Pseudo-kod jednostavnog EA

```
generiši slučajnu populaciju
repeat
    selekcija roditelja
    primena operatora ukrštanja nad roditeljima i kreiranje dece
    primena operatora slučajne mutacije nad potomcima
    izračunavanje kvaliteta (podobnosti) potomaka
    izbor jedinki za sledeću generaciju primenom operatora selekcije
until nije zadovoljen uslov završetka izvršavanja
kraj
```

Najpoznatiji predstavnici familije evolutivnih algoritama su genetski algoritmi (eng. *genetic algorithms - GA*).

Otac genetskih algoritama je Džon Holand, koji je ovu grupu algoritama predstavio u svojoj knjizi "Adaptacija u prirodnim i veštačkim sistemima: uvodna analiza i primene u biologiji, kontroli sistema i veštačkoj inteligenciji".

Algoritmi koji pripadaju genetskim algoritmima koriste *populaciju jedinki*, koje predstavljaju tačke u prostoru istraživanja, tj. potencijalna rešenja. Populacija jedinki se u većini praktičnih implementacija najčešće sastoji od 10 do 200 veštačkih agenata.

Pre nego što se započne proces optimizacije, potrebno je sva potencijalna rešenja (individue) na određeni način kodirati, a način kodiranja zavisi od prirode problema koji se rešava. Jedinke se najčešće kodiraju kao binarni nizovi fiksne dužine, ali postoje i slučajevi kada se koristi kodiranje slovima engleskog alfabeta. Zbog činjenice da ne postoji univerzalni "recept", kodiranje je jedan od najvećih izazova u primeni pristupa genetskih algoritama.

Genetski algoritmi za proces pretrage primenjuju operatore ukrštanja i mutacije. Kada se izvršava ukrštanje, kombinuju se bolja potencijalna rešenja iz populacije i tako teže ka optimalnom domenu. Operatorom mutacije se na slučajan način modifikuje jedna ili više komponenti rešenja. Ovaj operator je bitan kako bi se sprečio scenario da se process pretrage zaglavi u nekom od suboptimalnih domena prostora koji se istražuje.

Jedinke ili individue se u genetskim algoritmima prikazuju kao hromozomi, a njihovi delovi se nazivaju geni. Kada se optimizuje neka funkcija, funkcija sa svim parametrima predstavlja hromozom, dok pojedini parametri modeliraju gene.

Kvalitet se određuje za svaku individuu u populaciji, koji je definisan funkcijom podobnosti. Primenom operatora selekcije biraju se najbolje jedinke u populaciji, gde su nakon toga uključeni kao roditelji u narednoj generaciji izvršavanja algoritma. Individue sa lošijim vrednostima imaju manju šansu za dalju reprodukciju u sledećoj generaciji i na taj način postepeno izumiru.

Prikaz jednostavnog (osnovnog) GA dat je u Algoritmu 2.

Algoritam 2 Pseudo-kod jednostavnog GA

```
kreiranje pseudo-slučajne početne populacije od n hromozoma (x) dužine m
repeat
    evaluacija podobnosti fit(x) svih hromozoma iz populacije
    while n potomaka se ne kreira do
        selekcija dva hromozoma roditelja iz trenutne populacije
        ukrštanje izabranih roditelja sa verovatnoćom p_c i kreiranje dva potomka
        sa verovatnoćom p_m primena operatora mutacije nad potomcima
    end while
    zamena trenutne populacije individuama iz nove populacije
until nije dostignut maksimalni broj generacija izvršavanja algoritma
```

20.3. METAHEURISTIKE INTELIGENCIJE ROJEVA

Metaheuristike inteligencije rojeva (eng. *swarm intelligence metaheuristics*) pripadaju grupi metaheuristika inspirisanih prirodom, gde mehanizam koja izvršava pretragu simulira prirodni fenomen.

Kada se posmatra grupa organizama iz prirode, može da se primeti da grupa ili roj individua ispoljava neki vid inteligentnog ponašanja, iako se roj sastoji iz relativno nesofistiricanih i jednostavnih jedinki. Jedinke u grupi razmenjuju informacije uspostavljanjem nekog vida indirektne komunikacije, što omogućava da grupa funkcioniše kao celina i da teži ka jednom cilju.

Zajednički cilj može da bude pronalaženje hrane, izbegavanje opasnosti, promena prirodnog okruženja itd. Ponašanje rojeva, jata ili krda I njihovog inteligentnog preživljavanja su bili osnovni motivacioni faktori za nastanak i razvoj algoritama inteligencije rojeva.

Roj ili jato čini veliki broj homogenih i prostih jedinki, koje međusobno komuniciraju i uspostavljaju interakcije sa sredinom koja ih okružuje, a pri tom ne postoji centralna komponenta koja sinhronizuje njihovo ponašanje.

Obzirom da simuliraju jato ili roj, metaheuristike inteligencije rojeva spadaju u kategoriju populacionih algoritama, koji inkrementalno u iteracijama izvršavaju proces pretrage, sa ciljem pronalaska optimalmnog, ili približno optimalnog rešenja.

Metaheuristike inteligencije rojeva pripadaju domenu veštačke inteligencije (eng. *artificial intelligence - AI*). Njihova bitna karakteristika je *komuniciranje* i *razmena informacija između individua*.

Komunikacija između individua može da se ispolji na direktan i indirektan način.

Direktan način sam nalaže da je potrebno da individue vide ili čuju jedna drugu (video ili audio domet), i kao primer mogu da se izdvoje pčele koje se privlače na osnovu plesa mahanja krilima.

Drugi slučaj kada je *indirektna komunikacija* u pitanju, individue nisu u dometu, ali sva interakcija kao i komunikacija se odvijaju tako što individue menjaju okruženje u kojem žive i na taj način interaguju. Ovaka forma komuniciranja može da se vidi kod kolonije mrava, kada se kreću, mravi ostavljaju tragove u vidu feromona, tako da ostali članovi mogu da prate tu putanju i da stignu do izvora hrane.

Osim komunikacije, još jedna bitna karakteristika metaheuristika inteligencije rojeva jeste sposobnost *samoorganizacije*. Na osnovu samoorganizacije, grupe individua mogu bez kontrolnih i usmerivačkih aktivnosti od centralne komponente, da ostvare neki cilj ili da reše problem.

Osnovni principi na kojima se algoritmi rojeva baziraju su:

- ne postoji centralna komponenta koja kontroliše i koordinira ponašanje individua,
- ponašanje grupe proizilazi na osnovu akcija sinhronizovanog i kooperativnog ponašanja jedinki,
- interakcije između homogenih individua su paralelne i asihnrone,
- komunikacija između individua se u velikoj meri obavlja putem indirektnih veza u obliku stigmetrije,
- ciljevi grupe nisu preambiciozni, poput preživaljavanja u neprijateljskom i dinamičnom okruženju, pronalaženje izvora hrane i slično.

Pored osnovnih principa i karakteristika algoritama rojeva, najvažniji element jeste mehanizam koji izvršava proces pretrage. Način na koji mehanizam koji simulira grupe živih organizama izvršava proces pretrage se zasniva na dve komponente, *eksploatacija* (eng. *exploitation*) i *eksploracija* (eng. *exploration*). Ove komponente su u literaturi poznate i pod nazivima *intenzifikacija* (eng. *intensification*) i *diversifikacija* (eng. *diversification*).

Mehanizam intenzifikacije, koji se u većini slučajeva izvodi sa visokim stopama konvergencije, predstavlja komponentu koja koristi lokalne informacije u procesu pretrage, sa ciljem da se generišu bolja (kvalitetnija) kandidat rešenja od postojećih. Diverzifikacija se s druge strane odnosi na istraživanje prostora pretrage, da bi se generisala različita rešenja, koja se nalaze dalje od postojećih rešenja.

Kao primer mehanizma intenzifikacije može da se istakne osnovna jednačina pretrage *algoritma kolonije veštačkih pčela* (eng. *artificial bee colony - ABC*), kojom se generiše novo rešenje u susedstvu (okolini) starog (trenutnog) rešenja:

$$v_{ij} = x_{ij} + \varphi_{ij} (x_{ij} - x_{kj}),$$

gde v_{ij} i x_{ij} označavaju *j*-ti parametar novog i starog *i*-tog rešenja u populaciji, respektivno, φ_{ij} je pseudo-slučajan broj iz intervala [−1, 1], dok x_{kj} označava *j*-ti parametar slučajnog rešenja *k* iz populacije.

U nastavku su navedene jednačine kojima se modelira procedura eksploatacije *algoritma slepog miša* (eng. *bat algorithm - BA*), koje ažuriraju brzinu i pozicije svakog rešenja u populaciji:

$$f_i = f_{min} + (f_{max} - f_{min}) \beta,$$
$$v_{it} = v_{it-1} + (x^* - x_i^{t-1}) f_i,$$
$$x_i^t = v_i^{t-1} + v_i^t,$$

gde parametri f_i i v_i označavaju vrednost frekvencije i brzine slepog miša *i,* respektivno, $\beta \in [0, 1]$ je pseudo-slučajan broj generisan uniformnom distribucijom, dok se oznaka x^* odnosi na trenutno najbolje rešenje u populaciji.

Najveći izazov prilikom implementiranja i izvršavanja algoritma inteligencije rojeva jeste usklađivanje *balansa između eksploatacije* i *eksploracije* (eng. *exploitation-exploration trade-off*). Od usklađenosti njihovog odnosa zavisi kvalitet rešenja (globalno najbolje rešenje) i brzina konvergencije (prosečni kvalitet rešenja).

Ako je balans između ova dva procesa neuravnotežen u smeru ekploatacije, proces pretrage će najverovatnije brzo da konvergira, ali uz smanjenu verovatnoću pronalaska optimalnog regiona prostora pretrage.

U ranijim fazama izvršavanja algoritma, populacija može da izgubi različitost, i time da dovede do preuranjene konvergencije (eng. *premature convergence*). U retkim slučajevima može da se desi da pretraga konvergira ka optimalnom domenu prostora pretrage.

Ako se balans pomeri u korist eksploracije, algoritam će sa većom verovatnoćom da pronađe domen prostora pretrage gde se nalazi optimum, ali sa tendencijom spore konvergencije. U ovakvom balansu, prosečna rešenja će biti bolja, ali će zato najbolja rešenja biti lošija.

Nekada težnja za najboljim balansom između eksploatacije i eksploracije nije dovoljna, zato se nekada i primenjuju dodatni mehanizmi, koji treba dodatno da ubrzaju i da unaprede process pretrage. Neki od mehanizama jesu *permanentno ažuriranje trenutno najboljeg rešenja* (eng. *survival of the fittest*), ili primena *elitističke selekcije* (eng. *elitist selection*), čime se obezbeđuje da najbolja rešenja neće biti izgubljena i da mogu da se prenesu u sledeću generaciju.

Obzirom na veliki broj implementacija metaheuristika inteligencije rojeva u rešavanju optimizacionih problema u nastavku su istaknuti neki od predstavnika.

Najpoznatiji predstavnik inteligencije rojeva je *optimizacija rojevima čestica* (eng. *particle swarm optimizacion - PSO*), koja je inspirisana kretanjem jata ptica u prirodi. Eberhart i Kennedy su jedni od najznačajnijih istraživača u ovoj oblasti, obzirom da su još početkom 90tih godina prošlog veka, uspeli da efikasno modeliraju i inkorporiraju pravila kretanja jata ptica u optimizacioni algoritam. Mehanizam pretrage PSO algoritma izvodi se u odnosu na najbolje pronađeno rešenje i njegovog suseda. PSO algoritam je uspešno primenjivan i na probleme u oblsti klaud računarstva, čak i za problem raspoređivanja poslova u klaud okruženju sa ciljem minimizovanja potrošene energije i troškova, uz poštovanje ograničenja rokova izvršavanja poslova.

Drugi predstavnik metaheuristika inteligencije rojeva je *algoritam kolonije veštačkih pčela* (eng. *artificial bee colony - ABC*), koji simulira kooperativno ponašanje pčela u rojevima. Algoritam je prvi predložio Karaboga za probleme neprekidne optimizacije bez ograničenja, da bi ga nakon toga Karaboga i Bastuk adaptirali i za rešavanje problema globalne optimizacije sa ograničenjima.

Algoritam ABC koristi tri vrste veštačkih agenata:

- pčela radilica (eng. *employee bee*),
- pčela posmatrača (eng. *onlooker bee*) i
- izviđača (eng. *scout bee*).

ABC algoritam je uspešno testiran na širem setu benčmark funckija, kao i na praktičnim NP teškim problemima, poput optimizacije finansijskog portfolija, planiranja RFID mreže, inženjerskih problema i WSN lokalizacije. Za rešavanje problema u klaud računarstvu, ABC je implementiran sa ciljem uspostavljanja sistema efikasnog upravljanja heterogenim klaud resursima.

Treći značajni predstavnik metaheuristika inteligencije rojeva je *metaheuristika optimizacije pretragom svitaca* (eng. *firefly algorithm - FA*). Istrživač koji je kreirao ovaj algoritam, Xin-She Yang, bio je inspirisan svetlosnom signalizacijom i ponašanjem jata svitaca. Algoritam FA je adaptiran za optimizaciju neprekidnih problema bez ograničenja. U prirodi, svitci svetlost mogu da interpretiraju na nekoliko načina, kao svetlost koja treperi u kontekstu privlačenja, ili kao mehanizam za upozoravanje od napada. Privlačenje svitaca i mehanizmi upozoravanja bazirani na svetlosnoj signalizaciji, modelirani su i inkorporirani u osnovne jednačine pretrage FA metaheuristike. FA algoritam je uspešno prilagođen za heterogene praktične NP teške problem u mnogim oblastima.

Još jedan poznatiji predstavnik metaheuristika inteligencije rojeva je *algoritam optimizacije pretragom kukavica* (eng. *cuckoo search - CS*), koji je inspirisan inteligentnim ponašanjem jata kukavica, kreirali su Yang i Deb. Sličan algoritam, *metaheuristika pretrage slepog miša* (eng. *bat algorithm - BA*), koja modelira fenomen eholokacije slepih miševa, takođe je modelirao i predložio Yang. Oba algoritma su uspešno testirana na standardnim skupovima benčmark funkcija.

Imaju široku primenu u rešavanju praktičnih NP teških problema, kao su problemi planiranja RFID mreže, treniranja veštačke neuronske mreže (eng. *artificial neural network - ANN*), portfolio optimizacije, kao i implementacije u klaud računarstvu.

Naredni algoritam, *algoritam vatrometa* (eng. *fireworks algorithm - FWA*) spada u poznatije metaheuristike koje su predložili Tan i Zhu. FWA metaheuristika je inspirisana eksplozijom varnica koje nastaju puštanjem vatrometa. Primenjen je za rešavanje brojnih praktičnih NP teških problema.

Takođe, *algoritam pretrage moljaca* (eng. *moth search - MS*) spada u relativno nove metaheuristike inteligencije rojeva. Mehanizam pretrage MS algoritma simulira privlačenje ka izvorima svetlosti i Lévy letove moljaca. Ovaj pristup je kreirao Wang 2018. godine. Algoritam MS je testiran na benčmark funkcijama sa ograničenjima i bez ograničenja ali i na razne praktične probleme.

Kao još jedan relativno novi predstavnik metaheuristika inteligencije rojeva, kojeg je takođe kreirao Wang, navodi se *algoritam optimizacije krdom slonova* (eng. *elephant herding optimization - EHO*).

Metaheuristika EHO simulira socijalne interakcije između jedinki krda slonova pod vođstvom i uticajem ženke matrijarhata, gde kada odrastu, mužijaci napuštaju klan kako bi živeli nezavisno, ali još uvek komuniciraju sa ostatkom krda. Zbog obećavajućih rezultata koje EHO postiže u testiranjima na standardnim benčmark funkcijama, ovaj algoritam je ubrzo nakon nastanka adaptiran i implementiran i za praktične NP teške probleme.

Algoritam rasta drveća (eng. *tree growth algorithm - TGA*) nastao je sa ciljem simuliranja prirodne konkurencije među drvećem tokom apsorpcije hrane iz zemlje. Algoritam TGA su kreirali i predložili Armin Cheraghalipour i Mostafa Hajiaghaei-Keshteli 2017 godine. TGA algoritam ima mnogobrojne primene za probleme globalne optimizacije kao i za praktične probleme.

Osim ovih navedenih algoritama inteligencije rojeva, postoje i drugi predstavnici koji su ostvarili zapažene performanse u rešavanju heterogenih praktičnih problema. Može da se istakne i jedan od najstarijih predstavnika ove familije prirodom-inspirisanih metaheuristika, *algoritam optimizacije mravljih kolonija* (eng. *ant colony optimization - ACO*). Algoritam ACO se najčešće primenjuje za rešavanje grafovskih problema. ACO simulira indirektnu komunikaciju između mrava u koloniji, gde mravi ostvaruju interakcije ispuštanjem supstance koja se naziva feromon.

Algoritam koji se bazira na procesu donošenja *odluka grupe ljudi* (eng. *brain storm optimization - BSO*), takođe ostvaruje značajne rezultate iz različitih domena, kao na primer klasterovanje, planiranje putanje robota itd.

20.4. ALGORITAM OPTIMIZACIJE JATOM KITOVA

U ovom delu udžbenika, istaknut je detaljan prikaz *algoritma optimizacije jatom kitova* u cilju prikazivanja implementacije u rešavanju problema raspoređivanja poslova i balansiranja opterećenja u klaud računarstvu.

Algoritam optimizacije jatom kitova (eng. *whale optimization algorithm - WOA*) je prvi put predložen 2016. godine za rešavanje problema globalne optimizacije bez ograničenja i problema kontinulane optimizacije sa ograničenjima, gde je testiran na dobro poznatim inženjerskim instancama benčmark problema. WOA algoritam su kreirali Mirjalili i Lewis. Inspirisan je strategijom lova *u obliku mreže mehurića* (eng. *bubble-net hunting strategy*) grbavog kita (eng. *humpback whale*).

Proces pretrage WOA se zasniva na strategiji lova u obliku mreže mehurića grbavog kita. U prirodi, dok primenjuju ovakvu strategiju, grbavi kitovi pokazuju visok nivo kooperativnog i sinhronizovanog ponašanja. Lov u obliku mreže mehurića se izvodi tako što se kitovi prvo pozicioniraju ispod potencijalnog plena, a zatim simultano kreiranju mehuriće i podižu se, sledeći kružnu putanju, ka površini vode.

Mehurići tako stimulišu potencijalni plen da se kreće prema površini vode, gde postaju lak „zalogaj" za grbavog kita.

Kao što je objašnjeno u predhodnom poglavlju, svaki algoritam inteligencije rojeva ima dve osnovne komponente procesa pretrage:

- globalna (eksploracija) i
- lokalna (eksploatacija).

Procesom eksploatacije se modelira kruženje grbavog kita oko plena i spiralna (kružna) strategija napada sa kružnom mrežom mehurića, dok se procesom eksploracije modelira pseudo-slučajana potraga plena.

WOA metaheuristika priprada kategoriji *populaciono optimizacionih metoda*, individue u populaciji izvršavaju proces pretrage nezavisno, dok u isto vreme uspostavljaju formu indirektne komunikacije i stigmetrije, koja poboljšava proces pretrage i omogućava metaheuristici da brže konvergira ka optimum.

Prema osnovnoj WOA terminologiji, trenutno najbolje (*globalno najbolje*) kandidat rešenje (*rešenje sa najboljom vrednošću funkcije podobnosti*), predstavlja ciljni plen (eng. *target prey*), dok svako drugo rešenje predstavlja grbavog kita (*individua*) u populaciji.

U nastavku, izraz kit i plen nisu korišćeni, i umesto ovih izraza su upotrebljeni izrazi *kandidat rešenja* i *trenutno najbolje rešenje*.

Tokom faze eksploatacije (intenzifikacije), svako kandidat rešenje izvršava pretragu u svom bliskom okruženju i usmereno je prema poziciji sa trenutno najboljim globalnim rešenjem. Nakon izračunavanja podobnosti svakog rešenja u populaciji, pozicije svih rešenja u populaciji se ažuriraju u odnosu na poziciju globalno najboljeg rešenja (individue koja ima najveću vrednost funkcije podobnosti):

$$\vec{D} = |\vec{C} \cdot \vec{X}^*(t) - \vec{X}(t)|$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A} \cdot \vec{D},$$

gde $\vec{X}(t)$ i $\vec{X}^*(t)$ označavaju kandidat rešenja i trenutno najbolje rešenje u iteraciji *t*, respektivno, $\vec{A}$ i $\vec{C}$ predstavljaju vektore koeficijenata pretrage, dok se simbol · odnosi na operator unakrsnog množenja.

Sledeći izraz se koristi za računanje vektora koeficijenata A i C:

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a}$$

$$\vec{C} = 2 \cdot \vec{r}$$

U jednačinama, $\vec{r}$ označava uniformno distribuirani pseudo-slučajni vektor u intervalu [0, 1], dok je $\vec{a}$ vektor koji se u osnovnoj verziji WOA implementacije linearno smanjuje od 2 do 0 tokom izvršavanja algoritma.

Na osnovu jednačina, vidi se da kandidat rešenje $\vec{X}$ u svakoj iteraciji može da se usmeri prema bilo kojoj poziciji iz okruženja trenutno najboljeg rešenja $\vec{X}^*$, podešavanjem vrednosti vektora koeficijenata $\vec{A}$ i $\vec{C}$ uz upotrebu pseudo-slučajanog vektora .

Mehanizam smanjivanja poluprečnika kružnih putanja (eng. *the shrinking encircling mechanism*) i putanja spiralnog oblika (eng. *spiral-shaped path*), su osnovni matematički modeli koji oponašaju strategiju napada u obliku mreže mehurića.

Izvršavanje mehanizma smanjivanja poluprečnika kružnih putanja modelira se linearnim smanjivanjem vrednosti vektora $\vec{a}$, od vrednosti 2 do 0 (u osnovnoj WOA implementaciji), tako što se u svakoj iteraciji izvršavanja algoritma koristi sledeći izraz:

$$\vec{a} = 2 - t\frac{2}{maxIter},$$

gde *t* i *maxIter* označavaju trenutni i maksimalni broj iteracija u jednom izvršavanju algoritma, respektivno.

Drugi mehanizam, koji vodi proces eksploatacije, putanja spiralnog oblika, izvršava se u dva koraka: prvo se izračunava razdaljina između globalno najboljeg rešenja $\vec{X}^*(t)$ i trenutnog rešenja $\vec{X}(t)$ u iteraciji *t*, a zatim, nova (ažurirana) pozicija kandidat rešenja $(\vec{X}(t+1))$ se određuje primenom tzv. *spiralne jednačine*:

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t),$$

gde $\vec{D'}$, kao razdaljina između *i*-tog kandidat rešenja i globalno najboljeg rešenja, može da se izrazi kao $\vec{D'} = |\vec{X}^*(t) - \vec{X}(t)|$ gde b predstavlja konstantu koja definiše oblik logaritamske spirale, dok l označava pseudo-slučajan broj u opsegu od -1 do 1.

U prirodi se grbavi kitovi plivaju oko plena tako što kreiraju putanju spiralnog oblika, dok u isto vreme simultano smanjuju poluprečnik kružnih putanja kojima se kreću oko plena.

Ovakvo ponašanje se emulira tako što se u svakoj iteraciji izvršavanja algoritma, mehanizmi smanjivanja poluprečnika kružnih putanja i putanja spiralnog oblika primenjuju sa podjednakom verovatnoćom p:

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - \vec{A} \cdot \vec{D} \text{ , if } p < 0.5 \\ \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) \text{ , if } p \geq 0.5 \end{cases}$$

Za svako rešenje i u svakoj iteraciji, parametar *p* se bira na osnovu nasumične distribucije između 0 i 1.

Druga komponenta procesa pretrage, eksploracija (diversifikacija), izvodi se tako što se svako kandidat rešenje iz populacije ažurira u odnosu na poziciju nasumično odabranog rešenja, umesto u odnosu na poziciju globalno najboljeg rešenja, kao što je slučaj tokom procesa eksploatacije.

U fazi eksploracije koristi se vrednost vektora koeficijenta $\vec{A}$. Ako su vrednosti komponenti vektora $\vec{A}$ veće od ili jednake od 1 ($|A| \geq 1$), ažurirane pozicije kandidat rešenja će biti usmerene prema nasumično odabranim rešenjima i na taj način se izvršava globalna pretraga.

Sledećim izrazom se modelira faza eksploracije WOA metaheuristike:

$$\vec{X}(t+1) = \vec{X}_{rnd}(t) - \vec{A} \cdot \vec{D},$$

gde se $\vec{D}$, kao razdaljina između *i*-tog kandidat rešenja i nasumičnog rešenja *rnd* iz populacije u iteraciji *t*, izračunava kao $\vec{D} = |\vec{C} \cdot \vec{X}_{rnd}(t) - \vec{X}(t)|$.

Pseudo-kod originalne WOA implementacije, koji prikazuje osnovne korake u izvršavanju algoritma, dat je u Algoritmu 3.

Algoritam 3 Pseudo-kod WOA metaheuristike

Inicijalizacija. Generisanje početne slučajne populacije $X(i = 1,2,3,....,N)$ i inicijalizacija vrednosti kontrolnih parametara.
Izračunavanje podobnosti. Izračunavanje funkcije podobnosti svakog kandidat rešenja i određivanje globalno najboljeg rešenja X^*
while ($t < maxIter$) **do**
  for svako kandidat rešenje **do**
    Ažuriranje vrednosti parametara A, C, a, l i p
    if $p < 0.5$ **then if**
      $|A|<1$ **then**
        Ažuriranje pozicija kandidat rešenja X
      else
        Biranje slučajnog rešenja rnd iz populacije.
        Ažuriranje pozicija kandidat rešenja X
      end if else
        Ažuriranje pozicija kandidat rešenja X
    end if end
  for
  Prilagoditi vrednosti parametara na donje ili gornje granice izvodljivog domena prostora pretrage.
  Evaluacija svih rešenja u populaciji i računanje podobnosti.
  Ažuriranje pozicija globalno najboljeg rešenja X^* ako je potrebno. $t = t + 1$
  end while

return Najbolje rešenje (X^*) iz populacije



PITANJA ZA RAZMATRANJE

1. Klasifikovati optimizacione algoritme.
2. Koja je razlika između lokalne I globalne pretrage?
3. Šta su deterministički algoritmi?
4. Šta su stohastičke optimizacione metode?
5. Navesti primere metoda stohastičke optimizacije.
6. Šta su heuristike?
7. Kako su nastale heuristike?
8. Šta treba da sadrži heuristički algoritam?
9. Šta su metaheuristike?
10. Metaheuristički pristupi pokušavaju da odgovore na koje zahteve?
11. Navesti karakteristike metaheurističkih metoda.
12. Kako se dele metaheuristike?
13. Koja je razlika između metaheuristika koje su inspirisane prirodom i koje nisu inspirisane prirodom? Da li se mehanizam pretrage razlikuje?
14. Objasniti mehanizam procesa pretrage kod metaheuristika koje su inspirisane prirodom.
15. Kako se dele metaheuristike koje su inspirisane prirodom?
16. Šta su evolutivni algoritmi? Objasniti njihov nastanak i proces.
17. Kako se nazivaju algoritmi koji pripadaju grupi evolutivnih algoritama?
18. Prikazati pseudo-kod EA.
19. Šta su genetski algoritmi?
20. Objasniti proces pretrage genetskih algoritama.
21. Prikazati pseudo-kod GA.
22. Šta su metaheuristike inteligencije rojeva?
23. Koje prirodne fenomene oponaša i primenjuje algoritam inteligencije rojeva?
24. Da li metaheuristike spadaju u populacione algoritme? Zašto?
25. Objasniti razliku između direktne i indirektne komunikacije između individua.

26. Šta je samoorganizacija?
27. Na koje dve komponente se zasniva mehanizam pretrage metaheuristika inteligencije rojeva? Objasniti.
28. Zašto je bitno da se uspostavi balans između mehanizama pretrage?
29. Kako mogu da se izbalansiraju ova dva mehanizma pretrage?
30. Navesti predstavnike metaheuristika inteligencije rojeva.
31. Kako je nastao algoritam optimizacije jatom kitova?
32. Na čemu se zasniva proces pretrage WOA?
33. Kako su manifestovani mehanizmi pretrage kod WOA algoritma? Na šta se odnosi globalna pretraga a na šta lokalna?
34. Da li WOA algoritam pripada kategoriji populacionih metoda? Zašto?
35. Šta je kandidat rešenje a šta je trenutno najbolje rešenje?
36. Objasniti mehanizme pretrage kod WOA algoritma.
37. Objasniti pseudo-kod WOA algoritma.

21. CLOUDSIM PLATFORMA

CloudSim platforma predstavlja samostalni okvir (eng. *framework*) sa proširivim simulacionim setom alata za postavku klaud okruženja. Omogućava modeliranje i simulaciju klaud sistema i aplikacija.

CloudSim toolkit softver, koji pripada grupi softvera otvorenog koda (eng. *open source*), u sklopu licence *Apache verzije 2.0*, razvijen je u Java tehnologiji u laboratoriji za klaud računarstvo i distribuirane sisteme.

Široko se primenjuje za istraživanje, kako od strane akademske zajednice, tako i od strane industrijskih entiteta (korporacija).

CloudSim može besplatno da se preuzme sa sledećeg *URL*-a:

```
https://github.com/Cloudslab/ cloudsim/releases
```

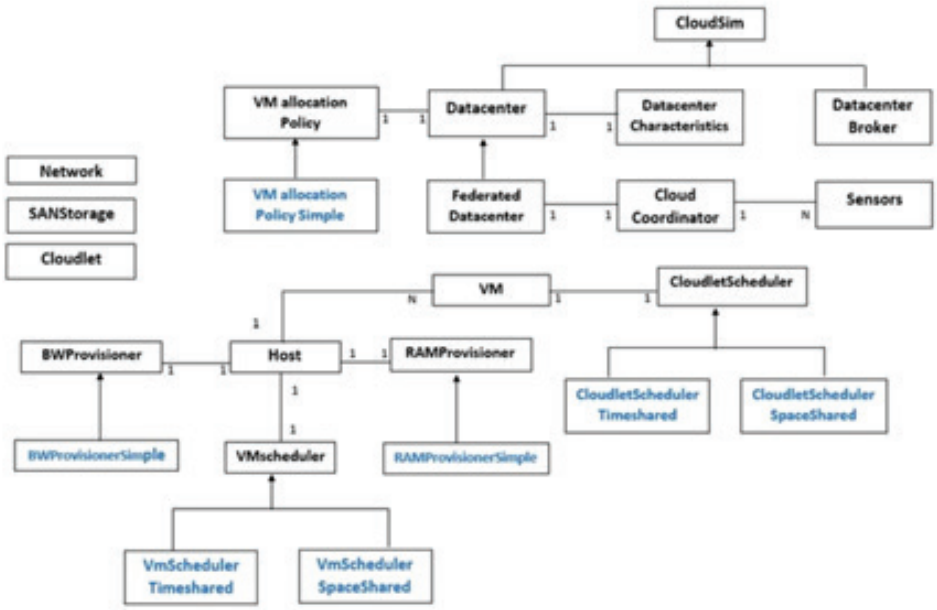
CloudSim omogućava eksperimentalno okruženje koje je kontrolisano za:

- kreiranje i modeliranje klaud entiteta,
- uključivanja korisničko definisanih polisa za raspoređivanje i alokaciju,
- pruža mrežnu arhitekturu i infrastrukturu koja se lako modifikuje,
- omogućava primenu tehnologije virtuelizacije i
- obezbeđuje softversku emulaciju host računara, mreže i mehanizam rezervisanja aplikacija.

Biblioteka *CloudSim*-a sastoji se iz 12 paketa, koji mogu da se koriste za simuliranje različitih tipova scenarija klaud računarstva. Najbitniji paket je *CloudSim* koji sadrži klase za modeliranje različitih *klaud entiteta* kao što su *Datacenter, Cloudlet, Host, VM* i *DatacenterBroker*.

Na Slici 21-1 je prikazan *CloudSim* dizajn dijagrama klase i arhitekture:

Slika 21-1: *Dijagram klasa i arhitekture CloudSim-a*



Cloudlet je klasa koja modelira servise klaud orijentisanih aplikacija, kao što su isporuka sadržaja i tok poslova, ali takođe i drugih poslova koji se preuzimaju od krajnjih korisnika za obradu u klaud sistemu. Pre izvršavanja svake simulacije, osnovni parametri kao što su *dužina instrukcija* i *metod transfera podataka* trebaju da se definišu, što se radi u okviru ove klase. Po potrebi, klasa *cloudlet* može da se proširi na različite tipove modela i aplikativnih servisa.

CloudletScheduler predstavlja abstraktnu klasu koja se primenjuje za implementiranje različitih polisa koje definišu način na koji se snaga obrade virtuelnih mašina deli između poslova. U osnovnoj *CloudSim biblioteci*, dva tipa ovih polisa su dostupne, i to:

- space-shared (*CloudletSchedulerSpaceShared*) i
- time-shared (*CloudletSchedulerTimeShared*).

Datacenter klasa modelira hardversku arhitekturu, na kojoj se svaka virtuelna mašina izvršava. U ovoj klasi je enkapsulirana polisa za definisanje karakteristika platforme *host* računara, koja može da bude *homogena* i ili *heterogena*.

Homogeni host deli istu konfiguraciju hardvera (broj i tip CPU i CPU jezgara, memoriju, kapacitet skladištenja).

U *heterogenom host* okruženju, različiti hostovi imaju različite karakteristike fizičkog hardvera. Takođe je korisno da se napomene da svaka *Datacenter* klasa instancira komponentu interfejsa za *rezervisanje aplikacija*, koja implementira grupu polisa za mrežni protok, memoriju i alokaciju skladišta host računara i virtuelnih mašina.

DatacenterBroker klasa modelira entitet brokera koji posreduje između SaaS i CSP. Broker radi u sklopu SaaS provajdera. Kada se prihvati zahtev koji dolazi od krajnjih korisnika, broker otvkriva odgovarajući CSP servis, tako što šalje upit servisu klaud informacija (eng. cloud information service-CIS), gde se zatim izvršava proces online pregovaranja u vezi alokacije resursa i/ili servisa koji moraju da zadovoljavaju kvalitet usluga aplikacije (eng. *Quality of Service - QoS*).

Klasa *DatacenterCharacteristics* vrši enkapsulaciju informacije koja se odnosi na konfiguraciju različitih resursa data centara. *Host* je glavna *CloudSim* komponenta koja modeluje fizički resurs.

Ova klasa sadrži različite bitne podatke o karakteristikama fizičkog resursa, kao što je količina dostupne RAM memorije i kapacitet skladištenja, listu i tip elemenata obrade (jezgra), informaciju o polisi koja se primenjuje za daljenje snage obrade podataka između virtuelnih mašina, polise deljenja protoka mreže i memorije za virtuelne mašine.

Klasa *VmScheduler* predstavlja abstraktnu klasu koju *Host* komponenta implementuje, koja se koristi za modelovanje polise deljenog prostora (eng. *space-shared*) i deljenog vremena (eng. *time-shared*) za alokaciju jezgara procesora fizičkog hosta na virtuelne mašine.

Svaka *Host* komponenta *CloudSim*-a upravlja i hostuje jednu ili više virtuelnih mašina. U klasi *Vm* moguće je podesiti sledeće važne atribute svake virtuelne mašine:

- brzina obrade,
- pristupna memorija,
- veličina prostora skladištenja podataka, kao i
- internu polisu rezervisanja koja se nasleđuje apstraktnu komponentu *CloudletScheduler*.

Klasa *VmmAllocationPolicy* predstavlja apstraktnu klasu koja modelira polisu rezervisanja koja se koristi za alociranje virtuelnih mašina na *host* računare. Najvažnija uloga ove klase je izbor raspoloživog host računara iz data centra, koji zadovoljava minimalne zahteve za obezbeđivanje virtuelne mašine. Ovi zahtevi se pre svega odnose na potrebnu memoriju, snagu obrade i skladištenje.

Proces generisanja *CloudSim simulacija* (životni ciklus simulacija) može da se sumira u osam koraka:

- inicijalizacija *CloudSim* paketa,
- kreiranje data centara,
- kreiranje brokera,
- generisanje poslova,
- kreiranje virtuelnih mašina,
- pokretanje simulacije (automatski proces koji se izvršava putem simulacija diskretnih događaja),
- zaustavljanje simulacije i
- štampanje rezultata (vizuelizacija izlaznih rezultata).

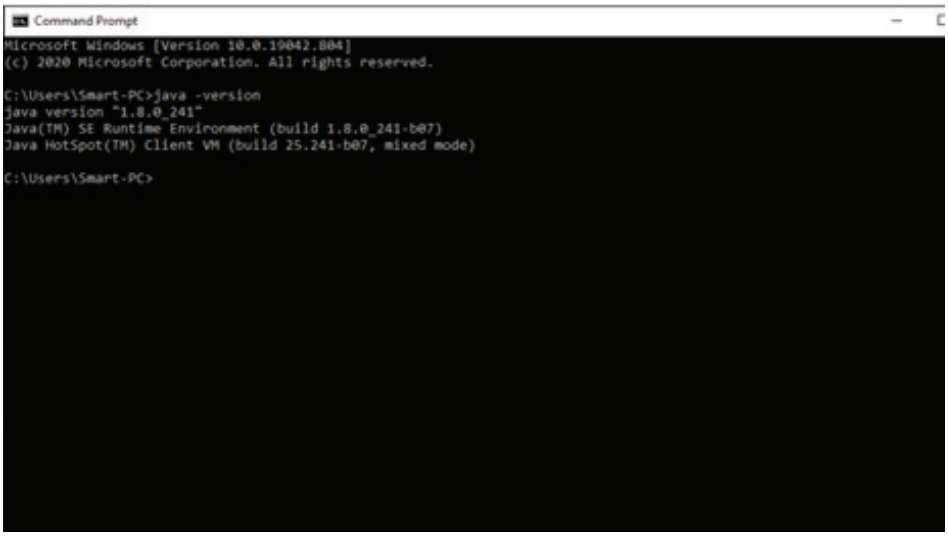
Da bi se u okviru simulatora implementirale metaheuristike inteligencije rojeva, koje imaju ulogu komponente raspoređivača poslova, potrebno je modifikovati neke od osnovnih *CloudSim* klasa. U eksperimentalnim simulacijama korišćena je verzija *3.0.3 CloudSim platforme*. U narednim poglavljima je prikazan proces instalacije i primeri simulacija u *CloudSim* platformi, a nakon toga je obuhvaćen problem raspoređivanja poslova i balansiranja opterećenja u klaud okruženju primenom metaheuristika inteligencije rojeva.

21.1. CLOUDSIM SIMULACIJA

U ovom poglavlju je objašnjen proces instalacije i neki od primera simulacija *CloudSim* platforme koje imaju za cilj da upoznaju studente sa radom u *CloudSim*-u. Dakle, *CloudSim*, kao biblioteka za Cloud simulacije je napisana u *Javi*. Za okuženje je definisan *Eclipse IDE*, i pre instalacije *CloudSim*-a potrebno je proveriti *Java Development Kit*.

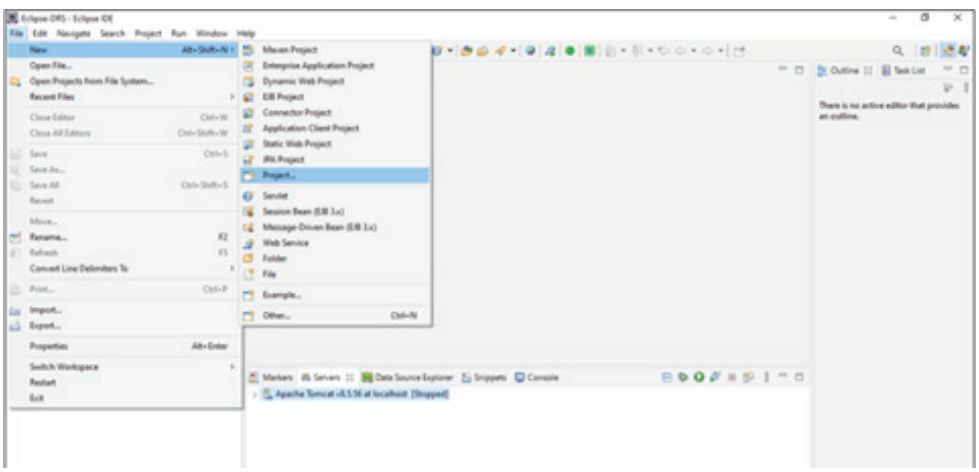
Na Slici 21-2 se može videti provera *Java Development Kit* na terminal računara.

Slika 21-2: *Provera Java Development Kit*



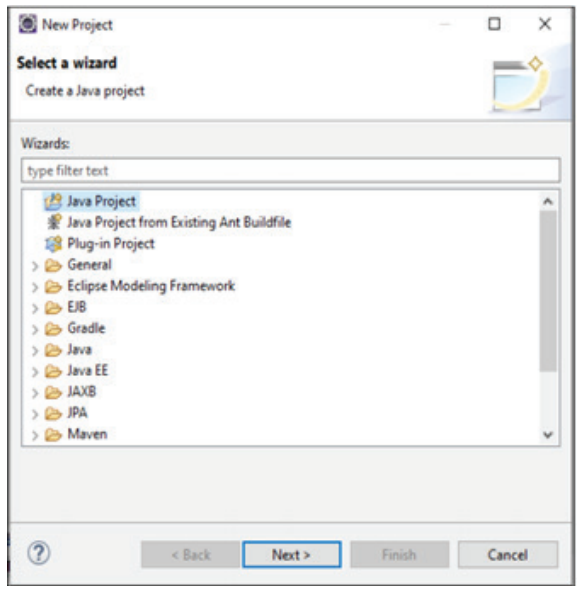
Sledeći korak je kreiranje novog projekta u *Eclipse IDE*, kao što se vidi na Slici 21-3.

Slika 21-3: *Kreiranje novog projekta*



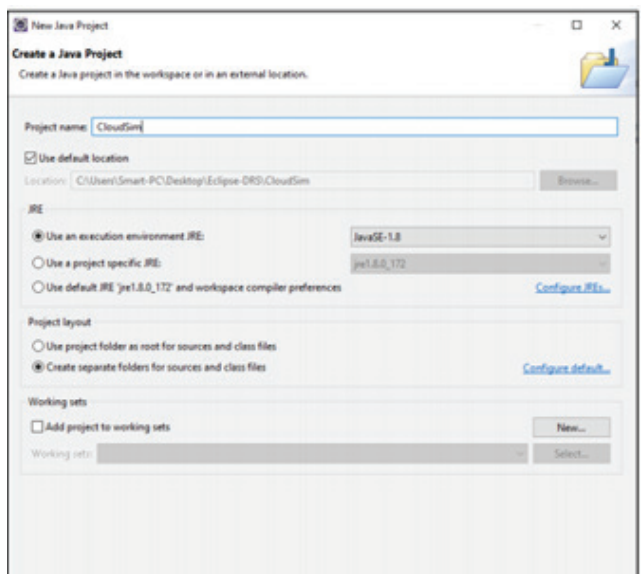
Prilikom odabira novog projekta (*New Project*) otvara se novi prozor gde je potrebno selektovati *Java Project* Slika 21-4.

Slika 21-4: *Java Project*



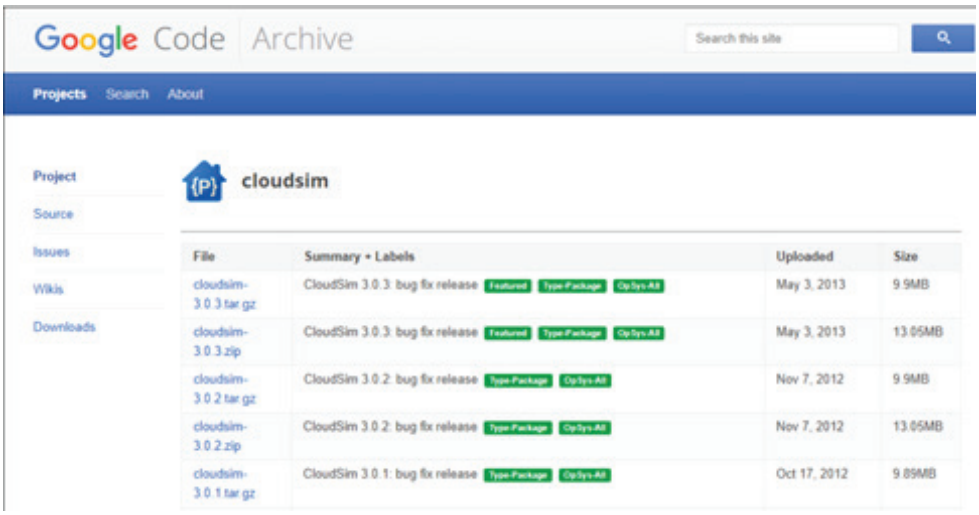
Naziv *Java* projekta je *CloudSim*, lokacija opciona, okruženje izvršavanja *JRE* je *JavaSE*-1.8. Slika 21-5.

Slika 21-5: *Koraci kreiranja Java projekta*



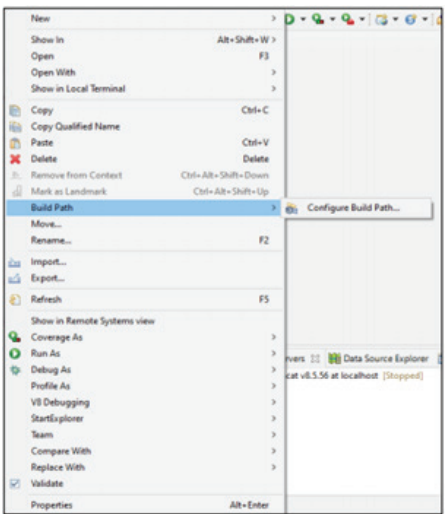
Nakon ovih koraka, potrebno je da se preuzme *CloudSim* biblioteka i da se doda na novi projekat koji je kreiran Slika 21-6.

Slika 21-6: *Preuzimanje biblioteke*



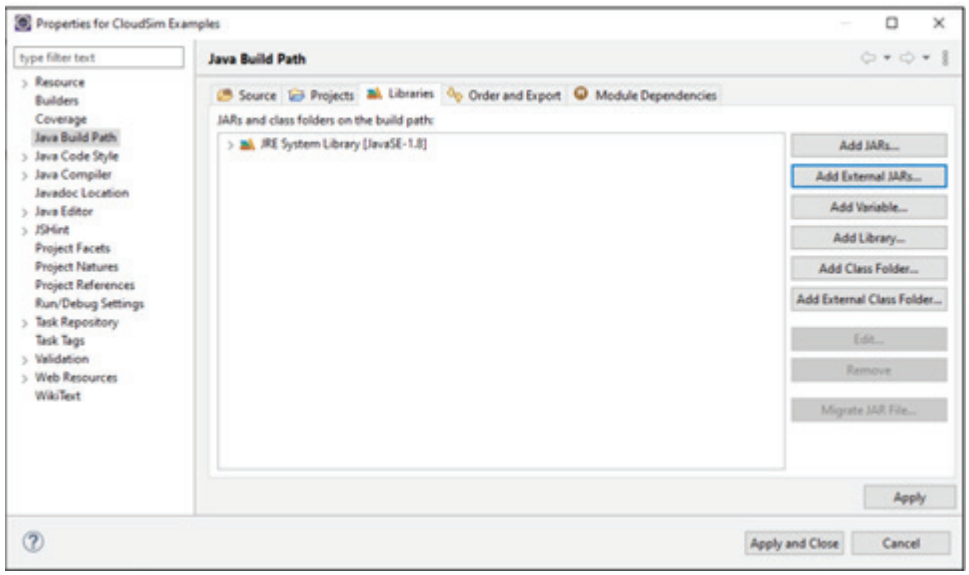
Nakon preuzimanja file-a i njegovog ekstrahovanja, potrebno je locirati *.jar* file-ove *CloudSim* folder-a. Potrebno je prebaciti sve file-ove u kreirani folder projekta. Ovaj proces se može videti na Slici 21-7, Slici 21-8, Slici 21-9, Slici 21-10.

Slika 21-7: *Konfigurisanje putanje*



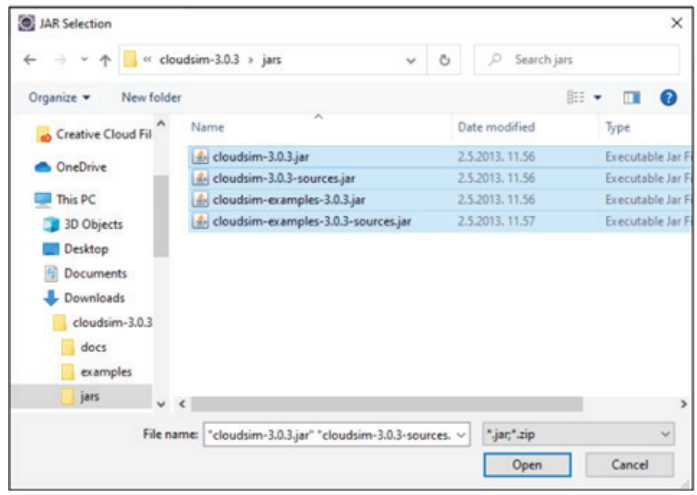
Kada se klikne na *Build Path-Configure Build Path*, svi *.jar* file-ovi će biti dodati u folder *Referenced Libraries*.

Slika 21-8: *Dodavanje eksternih JAR file-ova*



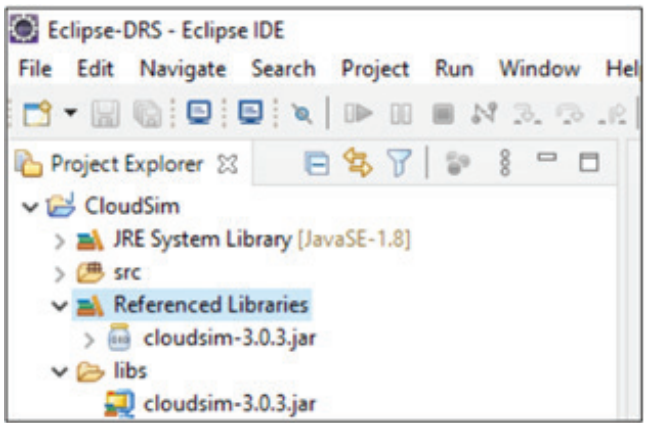
Nakon što se klikne opcija *Add External JARs*, potrebno je selektovati sve *.jar* file-ove.

Slika 21-9: *Selektovanje .jar file-ova*



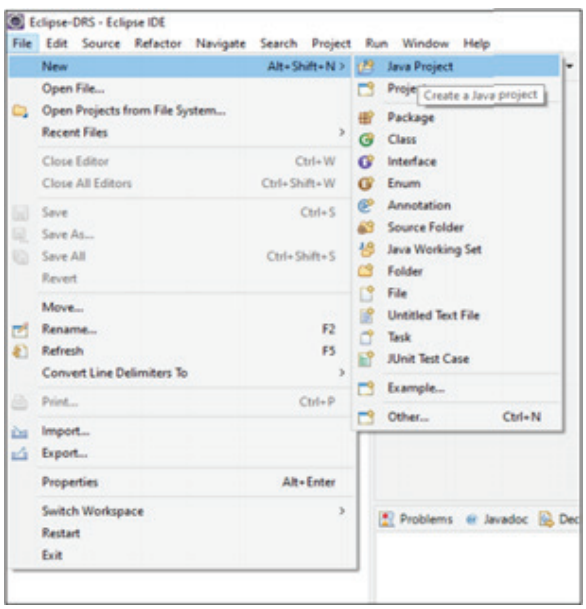
U sklopu projekta sa leve strane na *Project Explorer* može da se vidi dodata *Referenced* biblioteka Slika 21-10.

Slika 21-10: *Referenced Libraries*



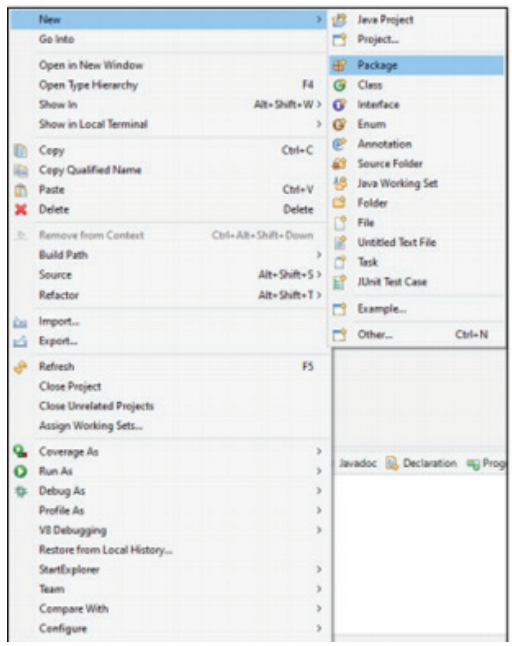
U sledećim koracima je objašnjeno dodavanje primera *CloudSim* projekta i kreiranje okruženja. Potrebno je da se u program *Eclipse* ide na *File-New-Java Project*, prikazano na Slici 21-11. Naziv pojekta je proizvoljan.

Slika 21-11: *CloudSim projekat*



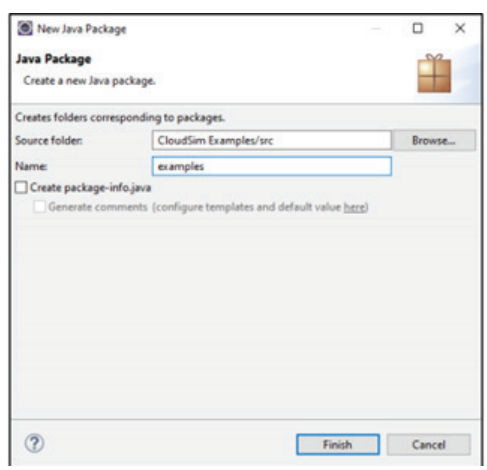
Nakon kreiranja projekta potrebno je kreirati i pakete, tako što se ide na *File-New-Package* Slika 21-12.

Slika 21-12: *Kreiranje paketa*



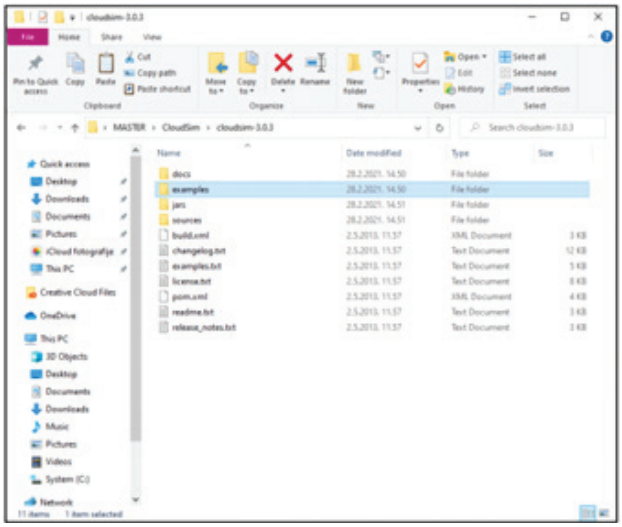
Bitno je napomenuti da naziv paketa mora da počne sa malim slovom Slika 21-13.

Slika 21-13: *Kreiranje naziva paketa*



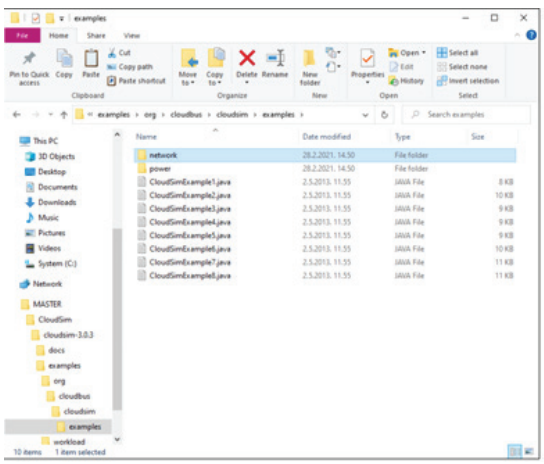
Kada je sve pripremljeno, sledeće šta je potrebno da se uradi jeste dodavanje primera *CloudSim*. Na Slici 21-14 je prikazan folder *CloudSim* verzije 3.0.3, i potrebno je da se ide na folder *examples-org-cloudbus-cloudsim-examples*.

Slika 21-14: *CloudSim putanja do primera*



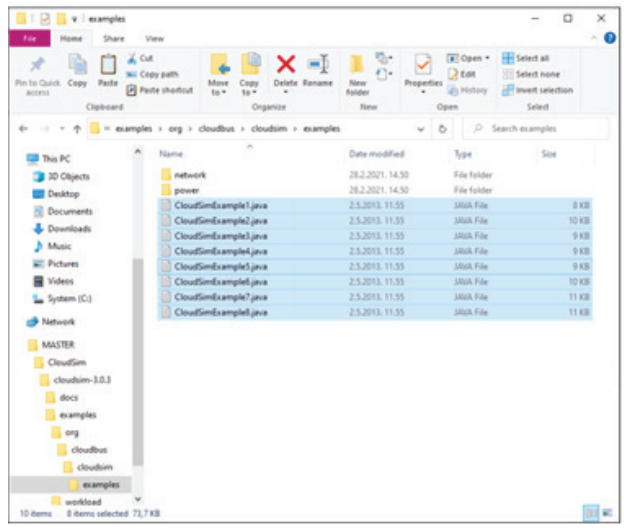
Nakon otvaranja foldera *examples* nalaze se dodatni folderi sa primerima vezano za računarske mreže, kao i primeri koji su vezani za klaud okruženje koji su glavni fokus ovog poglavlja, Slika 21-15.

Slika 21-15: *Primeri klaud okruženja*



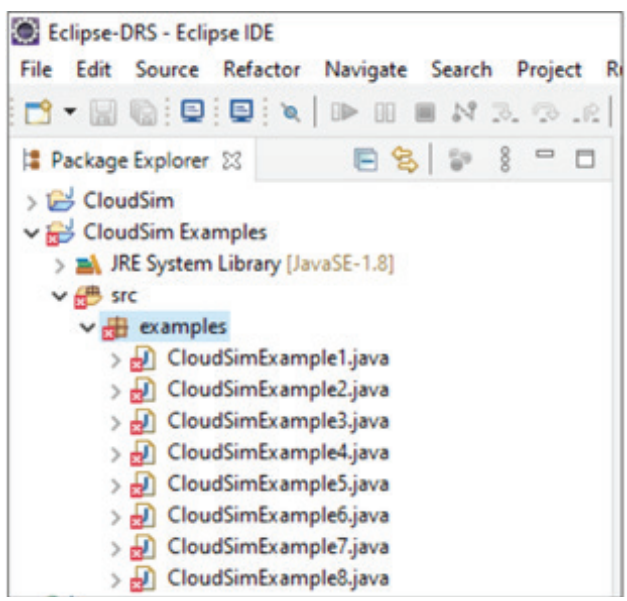
Selektovati date primere kao na Slici 21-16.

Slika 21-16: *Selektovanje primera*



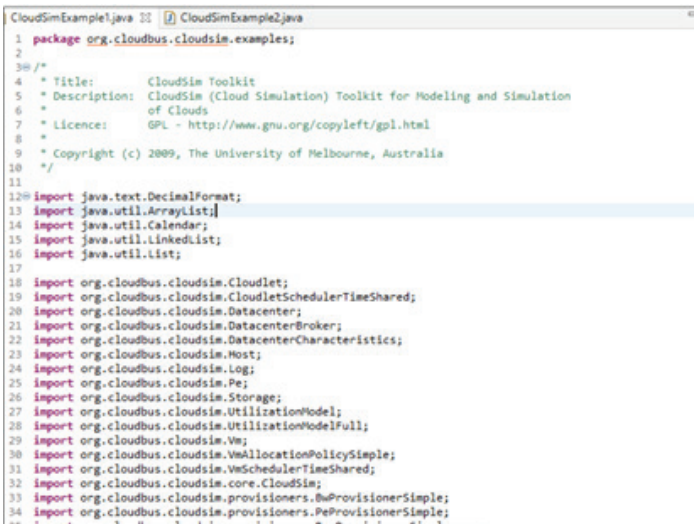
Dodati primere u *CloudSim* projekat, Slika 21-17.

Slika 21-17: *Dodavanje primera*

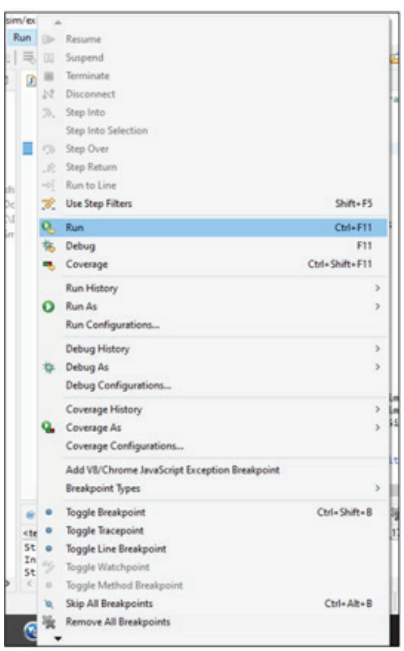


Nakon toga izabrati primer i otvoriti. Pritiskom na *run*, pokreće se kod. Slika 21-18, Slika 21-19, Slika 21-20.

Slika 21-18: *CloudSim primer 1 linije koda*

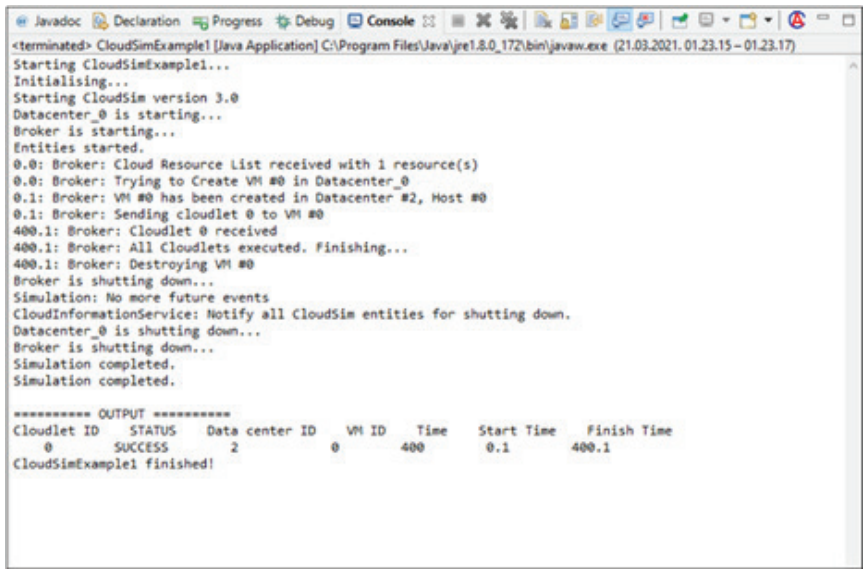


Slika 21-19: *Pokretanje koda*



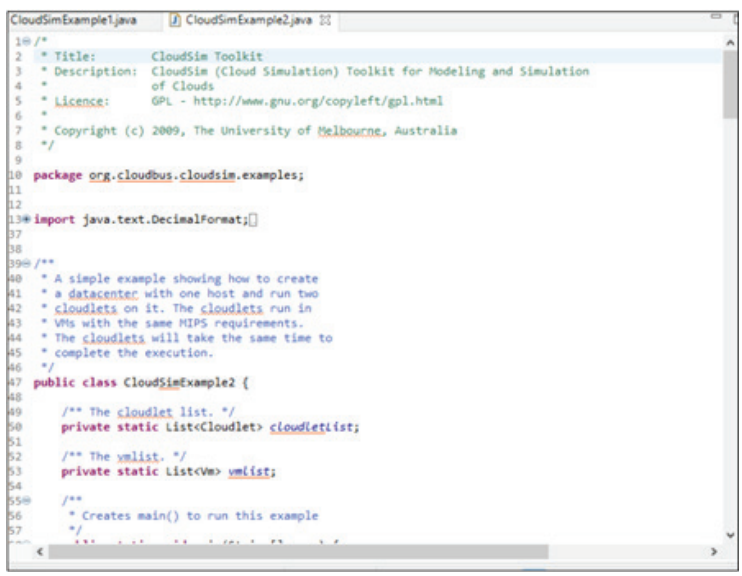
Nakon izvršavanja koda *output* je prikazan na sledeći način, Slika 21-20.

Slika 21-20: *Izvršenje primera 1 CloudSim*



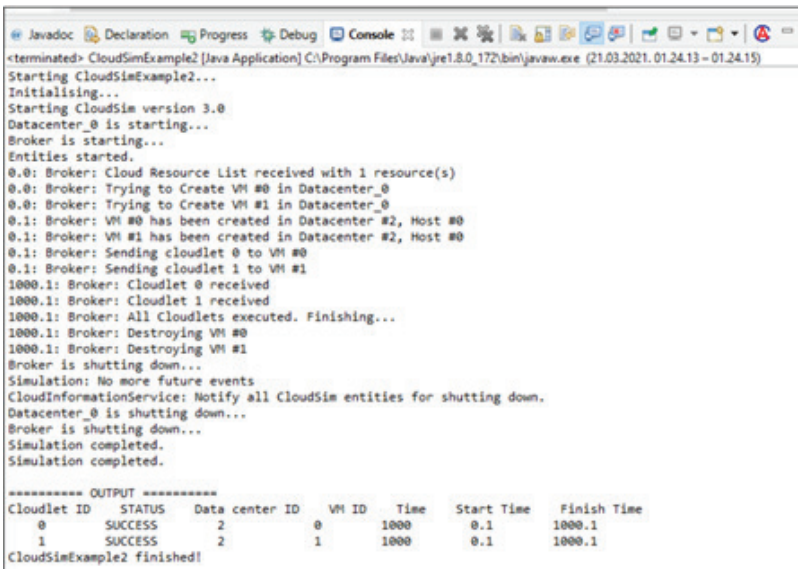
Na isti način mogu da se pokrenu i ostali primeri, kao što je primer 2 na Slici 21-21.

Slika 21-21: *Linije koda primer 2 CloudSim*



Nakon izvršavanja koda *output* je prikazan na sledeći način, Slika 21-22.

Slika 21-22: *Izvršenje primera 2 CloudSim*





PITANJA ZA RAZMATRANJE

1. Šta je *CloudSim*?
2. Šta pruža *CloudSim* platforma?
3. Navesti pakete koje *CloudSim* sadrži za modeliranje klaud entiteta.
4. Šta je *cloudlet*?
5. Šta je *data centar*?
6. Šta je *data centar Broker*?
7. Nabrojati korake u procesu generisanja *CloudSim* simulacija (životni ciklus simulacija).
8. Kako se preuzima *CloudSim* biblioteka?
9. Dodati primere *CloudSim*-a na projekat. Pokrenuti procese i objasniti arhitekturu klaud primera.

22. SIMULACIJE METAHEURISTIKE INTELIGENCIJE ROJEVA ZA PROBLEM RASPOREĐIVANJA POSLOVA U KLAUD OKRUŽENJU

U praksi je dokazano da svaki orginalni algoritam inteligencije rojeva može da se modifikuje, unapređuje ili hibridizuje sa drugim algoritmima rojeva ili da se samo određeni delovi unaprede. Na osnovu empirijskih testiranja osnovnog *WOA* algoritma, primećena su dva nedostatka, neadekvatan balans između eksploatacije i eksploracije, kao i preuranjena konvergencija.

Detalji o originalnoj verziji *algoritma optimizacije jatom kitova* (eng. *whale optimization algorithm - WOA*) mogu se pogledati u poglavlju 20.4.

22.1. PRIKAZ HIBRIDIZOVANOG WOA ALGORITMA

Proces eksploracije se u originalnom *WOA* pristupu izvršava samo u slučaju kada su oba uslova $p < 0.5$ i $|A \geq 1|$ zadovoljena. *WOA* mehanizam eksploracije usmerava proces pretrage prema postojećim slučajnim rešenjima iz populacije i zatim izvršava pretragu u bliskom okruženju takvih rešenja.

Nova pseudo-slučajna rešenja koja se nalaze u domenu prostora pretrage se uopšte ne razmatraju.

U ostalim slučajevima, proces pretrage originalnog *WOA* algoritma rojeva izvršava intenzifikaciju tako što generiše nova rešenja u pravcu trenutno najboljeg vektora rešenja $\vec{X}^*$.

Ako su uslovi $p < 0.5$ i $|A < 1|$ zadovoljeni, proces pretrage se izvršava primenom prve jednačine eksploatacije, a kada je uslov $p >= 0.5$ ispunjen, koristi se druga jednačina.

Na ovaj način se vidi neusklađenost balansa između intenzifikacije i diversifikacije u ranijim iteracijama izvršavanja algoritma.

Mehanizam eksploracije nije dovoljno jak da u nekim puštanjima algoritam *WOA* pronađe domen prostora pretrage gde se nalazi optimalno rešenje. Zbog toga su lošije prosečne vrednosti i kvalitet rešenja.

Balans između eksploatacije i eksploracije, koji je pomeren na stranu eksploatacije može da dovede do preuranjene konvergencije, kada se proces pretrage zaglavi u nekom od suboptimalnih regiona. Preuranjena konvergencija se javlja kada dva interna *WOA* kontrolna parametra $\vec{A}$ i $\vec{C}$, ne mogu da generišu bolja rešenja u uzastopnim iteracijama i populacija gubi diverzitet (eng. *population diversity*).

U nekim puštanjima algoritma, pod uticajem *faktora slučajnosti*, početna populacija se generiše dosta dalje od optimalnog domena prostora pretrage. Izvršavanjem procedure eksploracije, koja istražuje blisko okruženje početnih rešenja, proces pretrage ne može da konvergira prema optimalnom regionu.

U takvim slučajevima, rešenja ne mogu da se poboljšaju u uzastopnim iteracijama, i cela populacija se sastoji iz relativno sličnih rešenja (razlika može biti samo u nekoliko komponenti rešenja) koja su daleko od optimalnog regiona.

Na osnovu ovakvih činjenica, osnova za unapređenje originalnog *WOA* algoritma je balans između eksploatacije i eksploracije.

U literaturi mogu da se nađu i hibridni pristupi koji kombinuju druge metode veštačke inteligencije sa algoritmima rojeva. Kao primer navode se *veštačke neuronske mreže* (eng. *artificial neural networks - ANNs*) i *konvolutivne neuronske mreže* (eng. *convolutional neural networks - CNNs*) koje efikasno mogu da se kombinuju sa algoritmima inteligencije rojeva.

Modifikacije hibridizovanog *WOA* pristupa u odnosu na osnovni *WOA* mogu da se sumiraju na sledeći način:

- prvo, usvojen je mehanizam eksploracije iz ABC metaheuristike,
- drugo, uveden je dodatni dinamički kontrolni parametar koji kontroliše novi mehanizam eksploracije i
- treće, predloženi pristup uključuje jednačinu pretrage iz metaheuristike pretrage svitaca (FA).

Predloženi pristup je nazvan *WOA ABC eksploracija pretraga svitaca* (eng. *WOA ABC exploration firefly search – WOA-AEFS*).

U opisu algoritma korišćene su sledeće notacije: svako kandidat rešenje iz populacije se prikazuje kao vektor $\vec{X}_i$, gde se svako rešenje sastoji iz *M* komponenti (promenljivih odlučivanja) $\vec{X}_i = x_i^1, x_i^2, ...x_i^M$. Vrednost svake promenljive odlučivanja *j*, *i*-tog rešenja (x^j_i) uzima se iz domena [lb_j, ub_j], gde lb_j i ub_j označavaju donju i gornju granicu prostora pretrage u *j*-toj dimenziji, respektivno. Notacijom $fit(\vec{X}_i)$ označava se podobnost *i*-tog rešenja, dok je vrednost funkcije cilja istog rešenja označena kao $f(\vec{X}_i)$.

U originalnoj *WOA* implementaciji, proces eksploracije se vrši samo u scenariju kada su oba uslova $p < 0.5$ i $|A \geq 1|$, zadovoljena. Takođe, iz jednačine diversifikacije vidi se da je proces eksploracije orijentisan samo u smeru postojećih rešenja iz populacije i nova, još uvek neistražena rešenja se ne generišu.

U nekim puštanjima algoritma, kada se kreirana rešenja nalaze dalje od optimalnog domena prostora pretrage, eksploracija oko bliskog okruženja postojećih rešenja nije dovoljna da bi proces pretrage konvergirao ka optimalnom domenu.

S druge strane, mehanizam eksploracije ABC algoritma generiše kompletno nasumična rešenja, na isti način kao i u fazi inicijalizacije algoritma, kada se inicijalizuje slučajna početna populacija. Navedno je osnovna motivacija za unapređenje osnovne *WOA* metaheuristike hibridizacijom sa ABC pristupom.

Za svako rešenje $\vec{X}_i$ iz populacije, uključena je dodatna varijabla $trial_i$. U svakoj iteraciji, kada rešenje $\vec{X}_i$, ne može da se poboljša, njegova $trial_i$ vrednost se povećava za 1.

Kada *trial* vrednost određenog rešenja dostigne predetermisani prag tolerancije *limit*, to rešenje se izbacuje iz populacije i zamenjuje se drugim, nasumično generisanim rešenjem u okviru donjih i gornjih granica prostora pretrage, pri čemu se koristi sledeća jednačina:

$$x^i_j = \theta \cdot (ub_j - lb_j) + lb_j$$

gde je x^i_j, *j*-ta promenljiva odlučivanja *i*-tog rešenja, a θ označava pseudo-slučajan broj iz opsega [0, 1].

Na ovaj način, mehanizam eksploracije, pogotovo u ranim iteracijama izvršavanja algoritma, značajno poboljšava brzinu konvergencije i omogućava zadržavanje diverziteta čitave populacije. Ako se desi da pretraga zaglavi u nekom od suboptimalnih regiona, ovakav mehanizam je u stanju da efikasno usmeri proces pretrage prema nekim drugim domenima regiona koji se pretražuje.

Međutim, u kasnijim iteracijama, kada proces pretrage konvergira ka optimalnom regionu i kada je potrebna fino podešena (usmerena) pretraga, primena mehanizma ABC eksploracije potencijalno dobra rešenja, koja nisu poboljšana u nekoliko poslednjih iteracija mogu da budu izgubljena.

Kako bi se ovakav scenario izbegao, uključen je i dodatni kontrolni parametar, *stopa uticaja eksploracije* (eng. *exploration influence rate*) *eir*, čija se vrednost izražava u procentima.

Vrednost eir je na početku puštanja algoritma, podešena na 100, što označava da je cela populacija (100% populacije) pod uticajem ABC eksploracije. Tokom jednog izvršavanja algoritma, vrednost *eir* se postepeno smanjuje dok ne dostigne graničnu vrednost (prag), koja je u ovoj implementaciji podešena na 5.

Ako je vrednost *eir* 50, to znači da su 50% najlošijih kandidat rešenja izložena riziku da budu odbačeni iz populacije ako je njihova *trial* vrednost dostigla predefinisanu *limit* vrednost.

Sa ciljem uspešne primene opisanog mehanizma, pre nego što se mehanizam ABC eksploracije primeni, cela populacija se sortira na osnovu *vrednosti kriterijuma podobnosti u opadajućem redosledu*.

Na osnovu izvršenih empirijskih eksperimenata, došlo se do zaključka da približno optimalna vrednost dimamičkog *eir* parametra može da se izračuna kao:

$$eir = 100 \cdot (1 - \frac{t}{maxIter})$$

Dalje unapređenje osnovne *WOA* metaheuristike se odnosi na poboljšanje brzine konvergencije, gde se dodaje *strategija pretrage svitaca* iz *FA* algoritma rojeva. Prilikom inkorporiranja ovog elementa unapređenja WOAABC metaheuristika se hibridizuje i postaje WOA-AEFS.

U svakoj iteraciji izvršavanja hibridnog WOA-AEFS pristupa, ako je uslov $p_1 \geq 0.5$ zadovoljen, proces eksploatacije može da se izvrši korišćenjem *WOA* mehanizma spiralne strategije ili primenom FA jednačine pretrage sa jednakom verovatnoćom p_1. Strategija pretrage FA se primenjuje za svaki parametar *j* rešenja i korišćenjem sledeće jednačine:

$$x^j_i(t+1) = x^j_i + \beta_0 \cdot e^{-\gamma * r2i,k} (x^j_k - x^j_i) + \alpha * (rand - 0.5),$$

gde β_0 označava privlačenje svitaca na razdaljini $r = 0$, varijacija privlačenja je definisana γ parametrom, α je parametar slučajnosti (eng. *randomization parameter*) FA metaheuristike, *rand* je pseudo-slučajan broj između 0 i 1, dok *k* predstavlja slučajno rešenje iz populacije.

Razdaljina između rešenja *i* i *k*, označena kao $r_{i,k}$, računa se kao Dekartova razdaljina:

$$r_{i,k} = ||\vec{X}_i - \vec{X}_k|| = \sqrt{\sum_{j=1}^{M}(x_{i,j} - x_{k,j})^2},$$

gde je *M* broj promenljivih odlučivanja (*komponenti rešenja*).

Kao i u originalnom FA pristupu, u ovoj implementaciji je korišćena dinamička vrednost za parametar *alpha* koja zavisi od trenutne iteracije *t* i najvećeg broja iteracija (*maxIter*) u jednom izvršavanju algoritma:

$$\alpha\,(t) = (1 - (1 - ((10^{-4}/9)^{1/maxIter}))) \cdot \alpha\,(t - 1)$$

22.2. INICIJALIZACIJA POPULACIJE I WOA-AEFS PSEUDO-KOD

Početna populacija se inicijalizuje na početku svakog puštanja algoritma. U *WOA-AEFS* fazi inicijalizacije, populacija od *N* rešenja $\vec{X}_i(i = 1, 2, 3, ...N)$ gde se svako rešenje sastoji iz *M* promenljivih odlučivanja $\vec{X}_i = x_i^1, x_i^2, ...x_i^M$ se generiše. Cela populacija se označava matricom veličine *NxM*:

$$P = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \cdots & x_1^j & \cdots & x_1^M \\ x_2^1 & x_2^2 & x_2^3 & \cdots & x_2^j & \cdots & x_2^M \\ x_3^1 & x_3^2 & x_3^3 & \cdots & x_3^j & \cdots & x_3^M \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_N^1 & x_N^2 & x_N^3 & \cdots & x_N^j & \cdots & x_N^M \end{bmatrix}.$$

Svaki parametar (*promenljiva odlučivanja*) j svakog rešenja u populaciji se generiše u *fazi inicijalizacije* primenom jednačine:

$$x_i^j = \theta \cdot (ub_j - lb_j) + lb_j,$$

gde je x_i^j , *j*-ta promenljiva odlučivanja *i*-tog rešenja, a θ označava pseudo-slučajan broj iz opsega [0, 1].

Osnovni koraci izvršavanja predloženog *WOA-AEFS* algoritma su prikazani u Algoritmu 4:

Algoritam 4 Pseudo kod WOA-AEFS

Inicijalizacija. Generisanje slučajne početne populacije P i podešavanje vrednosti *trial* parametra svih rešenja na 0.
Inicijalizacija kontrolnih parametara. Inicijalizacija vrednosti za ABC kontrolne parametre eksploracije *eir* i *limit* i FA parametar pretrage α.
Izračunavanje podobnosti. Izračunavanje podobnosti svakog kandidat rešenja iz populacije i određivanje trenutno najboljeg globalnog rešenja X^*
while ($t < maxIter$) **do**
  for svako kandidat rešenje **do**
    Ažuriranje vrednosti za A, C, a, l, p, p_1
    if $p < 0.5$ **then**
      if $|A|<1$ **then**
        Ažuriranje pozicija kandidat rešenja X i
        čuvanje pozicija novog rešenja u vektor X_{new}
      else if $|A \geq 1|$ **then**
        Selekcija slučajnog rešenja X_{rnd} iz populacije
        Ažuriranje pozicija kandidat rešenja X i
        čuvanje pozicija novog rešenja u vektor X_{new}
      end if
    else if $p \geq 0.5$ **then**

    if $p_1 < 0.5$ **then**
      Ažuriranje pozicija kandidat rešenja X korišćenjem FA jednačine pretrage i čuvanje pozicija novog rešenja u X_{new}
    else if $p_1 \geq 0.5$ **then**
      Ažuriranje pozicija kandidat rešenja X primenom WOA spiralne pretrage i čuvanje pozicija novog rešenja u vektor X_{new}
    end if
    end if
    Izbor između starog X i novog X_{new} rešenja primenom mehanizma pohlepne selekcije
    Ako je novo rešenje izabrano, zameniti X sa X_{new} i podesiti *trial* vrednost na 0.
    Ako je staro rešenje izabrano, povećati *trial* vrednost rešenja X.
  end for
  Svako rešenje čije komponente imaju vrednosti van dopustivog regiona domena pretrage, vratiti u okvire dozvoljenih granica
  Evaluiranje svih rešenja iz populacije i računanje podobnosti
  Sortiranje svih rešenja po osnovu kriterijuma podobnosti u opadajućem redosledu
  Zamena svih rešenja koja pripadaju *eif*% najgorih rešenja u populaciji i za koje je uslov *trial* $\geq$ *limit* zadovoljen sa slučajnim rešenjem
  Ažuriranje pozicija globalno najboljeg rešenja X^* ako je potrebno
  $t = t + 1$
  Izračunavanje novih vrednosti dinamičkih parametara *eir* i α pomoću izraza
end while
return Najbolje rešenje (X^*) iz populacije

Kako bi se bolje i preciznije izvršile evaluacije performansi predloženog pristupa, izvršene su dve grupe simulacija:

- jedna sa pravim skupom podataka i
- druga sa veštačkim skupom podataka.

22.3. SIMULACIJE SA PRAVIM SKUPOM PODATAKA

U prvoj grupi simulacija, korićen je model sa *jednom funkcijom cilja*, dok je u simulacijama iz druge grupe, rešavan *višekriterijumski model raspoređivanja sa ograničenjima*.

Obe grupe simulacija su rađene u *CloudSim 3.0.3* okruženju. Takođe, obe simulacije izvršavane su na računarskoj platformi sa Intel *CoreTM* i7-4770K procesorom @4GHz sa 32GB *RAM* memorije, *Windows 10 Professional x64* operativnim sistemom i *Java Development Kit* 11 (*JDK* 11) i *IntelliJ* integrisanim razvojnim okruženjem.

Potencijalna rešenja problema u obe simulacije kodirana su kao *skup zahteva krajnjih korisnika*, gde je za svaki zahtev mapirana odgovarajuća *virtuelna mašina*. Dužina rešenja je ukupan broj prihvaćenih zahteva u klaud sistemu.

U nastavku su prvo prikazane simulacije sa pravim skupom podataka.

Korišćenjem realnih podataka generisanih iz robusnih logova super računara je najbolji način da se izvrši evaluacija performansi metaheuristika za rešavanje problema raspoređivanja poslova u klaud računarstvu.

Karakteristike realnih zahteva koji su korišćeni u simulacijama generisani su iz *NASA Ames iPSC*/860 log fajla u *Feitelson Parallel Workloads* arhivi (*PWA*). *NASA iPSC* se nalazi u numeričkom aerodinamičkom simulacionom sistemu, divizije u *NASA Ames* istraživačkom centru. Centar sadrži podatke iz perioda oktobar 1993. - decembar 1993. za 128-čvorova *iPSC*/860 klastera super računara.

Zapisi sadrže podatke o 42264 zahteva i 128 resursa (*CPU*). Podaci su preuzeti iz datoteke evidencije praćenja opterećenja *NASA-iPSC*-1993-3.*swf* sa sledeće *URL* adrese:

`https://www.cse.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/.`

Iz ovih zapisa su preuzeti *ID zahteva, dužina svakog zahteva izraženog u MIPS* jedinicama i potreban broj *procesirajućih elemenata* (eng. *requested processing elements - PEs*) za obradu svakog zahteva.

U simulacijama je generisana klaud infrastruktura u *CloudSim* simulatoru. U svim simulacijama korišćene su *homogene virtuelne mašine*, koje se nalaze na *dva fizička host računara* lociranih u jednom data centru. *Data centar* je generisan sa osnovnim *CloudSim* karakteristikama. Karakteristike *host* računara i virtuelnih mašina korišćenih u eksperimentima prikazani su na Slici 22-1, Slici 22-2.

Slika 22-1: *Konfiguracija host računara u CloudSim-u u simulacijama WOA-AEFS metaheuristike sa pravim skupom podataka*

Identifikator hosta	Parametar	Vrednost
Host1	RAM	3 (jedinice: GB)
	CPU tip	Intel Core 2 Extreme X6800
	Broj jezgara (PEs)	2
	CPU mogućnosti	27079 (jedinice: MIPS)
	Kapacitet skladišnog prostora	1 (jedinice: TB)
	Mrežni protok	10 (jedinice: Gbps)
	VM polisa raspoređivanja	bazirana na vremenu
Host 2	RAM	3 (jedinice: GB)
	CPU tip	Intel Core i7 Extreme Edition 3960X
	Broj jezgara (PEs)	6
	CPU mogućnost	177730 (jedinica: MIPS)
	Kapacitet skladištenja	1 (jedinica: TB)
	mrežni protok	10 (jedinica: Gbps)
	VM polisa raspoređivanja	bazirana na vremenu

Slika 22-2: *Konfiguracija virtuelnih mašina u CloudSim-u u simulacijama WOA-AEFS metaheuristike sa pravim skupom podataka*

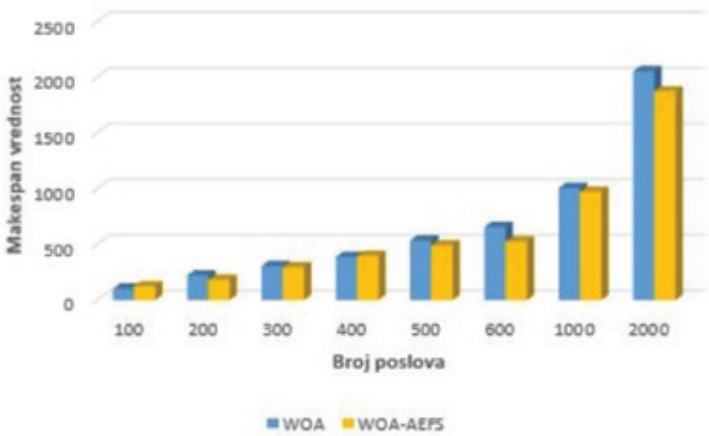
Parametar	Vrednost
broj virtuelnih mašina	10
CPU snaga obrade	9726 (jedinica: MIPS)
RAM memorija	512 (jedinica: MB)
kapacitet mrežnog protoka	1000 (jedinica: Mbps)
kapacitet skladištenja	10 (jedinica: GB)
polisa raspoređivanja poslova	bazirana na vremenu
VMM (Hipervizor)	Xen
Operativni sistem	Linux
broj CPU	1
CPU tip	Pentium 4 Extreme Edition

Kako bi se validirala skalabilnost *WOA-AEFS* metaheuristike rojeva, izvršene su simulacije sa malim skupom zahteva (od 100 do 600 poslova) i sa velikim skupom zahteva (od 1000 do 2000 poslova).

Osnovni *WOA-AEFS parametri* su podešeni na sledeći način: *veličina populacije* na 30 (N = 30) i *maksimalni broj iteracija u jednom puštanju algoritma* na 1000 (*maxIter* = 1000). Parametar *limit* je podešen na vrednost 33 (round(1000/30)), dok su *promenljivi parametri* $\vec{a}$, *eir* i α dinamički podešavani tokom jednog puštanja algoritma.

Prvo je urađena komparativna analiza sa osnovnom *WOA* metaheuristikom. Rezultati su prikazani na Slici 22-3.

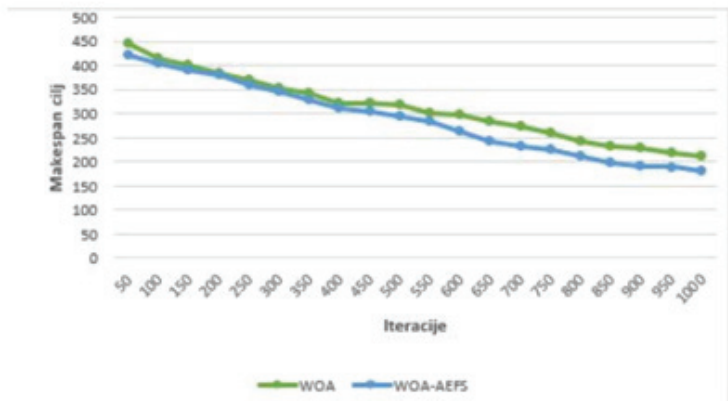
Slika 22-3: *Komparativna analiza—WOA-AEFS vs. WOA korišćenjem NASA iPSC realnih podataka za mali i veliki broj poslova*



Na osnovu prikazanog dijagrama (Slika 22-3), jasno je da je *WOA-AEFS* postiže značajno bolje performanse od originalnog *WOA*, osim u simulacijama sa 400 zahteva. U ovom slučaju, originalni *WOA* algoritam postiže neznatno bolje performanse makespan indikatora (oko 1%). Najznačajnija poboljšanja u performansama mogu da se primete u simulacijama sa 600 zahteva, gde je *WOA-AEFS* bolji od originalne *WOA* metaheuristike za više od 25 %.

Graf brzine konvergencije *WOA-AEFS* i *WOA* metaheuristika za test instance sa 200 poslova je dat na Slici 22-4.

Slika 22-4: *Grafikon brzine konvergencije WOA-AEFS vs. WOA korišćenjem NASA iPSC realnih podataka za 200 zahteva*

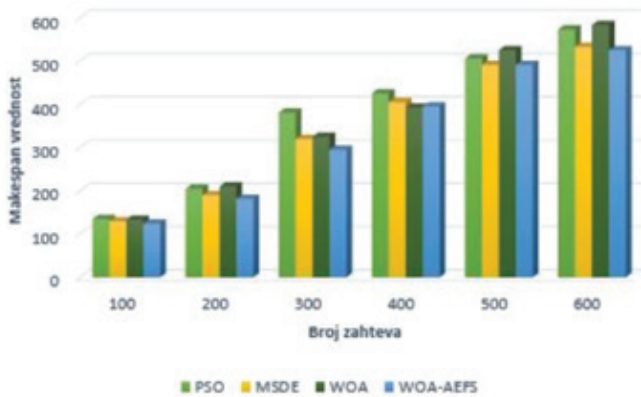


Kako bi se još bolje izmerile performanse i potvrdila robusnost *WOA-AEFS* metaheuristike, izvršena je komparativna analiza sa algoritmima *MSDE* (eng. *moth search differential evolution*) i *PSO*, koji su testirani na istoj platformi i pod istim eksperimentalnim uslovima kao i *WOA-AEFS* metaheuristika.

Međutim, da bi se ustanovile performanse originalog *WOA* u odnosu na *MSDE* i *PSO*, osnovna *WOA* verzija je prikazana u komparativnoj analizi.

Rezultati komparativne analize sa manjim brojem zahteva (od 100 do 600) i sa većim brojem poslova prikazani su na Slikama 22-5 i 22-6, respektivno.

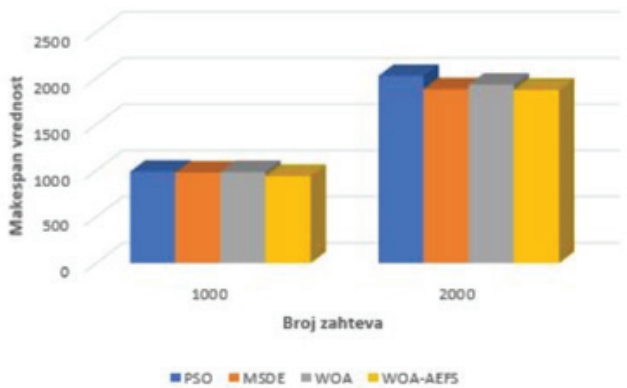
Slika 22-5: *Komparativna analiza—WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih podataka sa manjim brojem poslova*



Na osnovu rezultata koji se mogu videti na dijagramima sa Slika 22-5 i 22-6, može da se zaključi da u proseku, *WOA-AEFS* ostvaruje bolje *makespan vrednosti* nego drugi pristupi uključeni u komparativnu analizu.

Sa dijagrama se takođe može primetiti da *WOA-AEFS* metaheuristika generiše rešenja istog kvaliteta kao *MSDE* u simulacijama sa 500 poslova. U simulacijama sa većim brojem poslova (1000 i 2000), u oba slučaja *WOA-AEFS* nadmašuje sve druge pristupe.

Slika 22-6: *Komparativna analiza—WOA-AEFS vs. drugi algoritmi korišćenjem NASA iPSC realnih skupova sa većim brojem poslova*



Poboljšanja *makespan funkcije cilja*, koju *WOA-AEFS* ostvaruje u odnosu na druge pristupe, iskazane u procentima i date su na Slici 22-7.

Slika 22-7: *Poboljšanja WOA-AEFS algoritma u odnosu na druge algoritme u simulacijama sa NASA iPSC realnim podacima*

Broj poslova	PSO	MSDE	WOA
100	+8.60%	+4.83%	+7.63%
200	+13.23%	+4.97%	+16.16%
300	+29.08%	+8.47%	+10.18%
400	+7.71%	+2.53%	−0.70%
500	+2.99%	+0.02%	+6.96%
600	+9.19%	+1.52%	+11.08%
1000	+1.89%	0.92%	1.58%
2000	+8.10%	+0.42%	+3.08%

U poslednjem setu eksperimenata sa realnim skupom podataka, da bi se utvrdile performanse *WOA-AEFS* metaheuristike u *balansiranju opterećenja*, prikazana je komparativna analiza između *WOA-AEFS* i originalnog *WOA, MSDE* i *PSO* za vrednost *DI indikatora*. Detaljnije o *DI* indikatorima pogledati poglavlje 19.3.

Komparativna analiza sa *DI* indikatorom je data na Slici 22-8 gde su takođe prikazani rezultati za *MSDE* i *PSO*. Najbolji rezultati iz svake kategorije instanci testa su markirani u bold stilu.

Slika 22-8: *Komparativna analiza DI indikatora u simulacijama sa NASA iPSC realnim skupom podataka*

Broj poslova	PSO	MSDE	WOA	WOA-AEFS
100	0.974118	0.972773	1.004538	**0.935002**
200	1.375412	1.138444	1.337259	**1.373259**
300	1.051463	**0.905947**	1.032713	0.983205
400	1.050034	**0.878397**	1.063955	0.883302
500	1.077579	0.942521	1.311571	**0.941056**
600	1.097044	0.983239	1.270371	**0.976200**
1000	**1.013558**	1.040253	1.093521	1.0387226
2000	**0.974118**	1.011911	0.991087	0.9835211
Bolji	2	2	0	3

Na osnovu analiza rezultata *DI* indikatora, prikazan na Slici 22-8, može da se primeti da je u proseku, *WOA-AEFS* ostvario najbolje rezultate. *PSO* algoritam je pokazao da je sposoban za uspostavljanje efikasnog balansiranja opterećenja između virtuelnih mašina u okruženju sa velikim brojem poslova (1000 i 2000), i u ovim testovima *PSO* je pokazao najbolje performanse.

U simulacijama sa 300 i 400 poslova, *MSDE* je nadmašio sve algoritme, uključujući i *WOA-AEFS*. Predloženi *WOA-AEFS* je pokazao superiorniji kvalitet rešenja u simulacijama sa malim test instancama, i to sa 100,200,500 i 600 zahteva krajnjih korisnika, u simulacijama za balansiranje opterećenja.

Razmatrajući mogućnost implementiranja tehnika metaheuristika za problem raspoređivanja poslova u realnom klaud okruženju, vreme izvršavanja algoritama predstavlja značajan pokazatelj (*CPU vreme*).

Obzirom da su *WOA* i *WOA-AEFS* implementirani i testirani na istoj računarskoj platformi, u ovom slučaju može da se napravi komparativna analiza vremena izvršavanja ova dva algoritma. Komparacija CPU vremena je prezentovana na Slici 22-9.

Slika 22-9: *Komparativna analiza CPU vremena u simulacijama sa NASA iPSC realnim skupom podataka*

Algoritam	Broj poslova							
	100	200	300	400	500	600	1000	2000
WOA	**2730**	4122	6541	**8567**	11005	**15349**	**35152**	118782
WOA-AEFS	2855	**3983**	**6374**	8750	**10730**	15765	36110	**109351**

Na osnovu prikazanih rezultata na Slici 22-9 može da se zaključi je vreme izvršavanja obe metaheuristike približno jednako. Ovo ima važne implikacije za problem raspoređivanja poslova u klaud okruženju, kao i na performanse metaheuristika, s obzirom na to da kompleksnost *WOA-AEFS* u odnosu na originalni *WOA* nema značajan uticaj na vreme izvršavanja.

22.4. SIMULACIJE SA VEŠTAČKIM SKUPOM PODATAKA

U drugom setu eksperimenata raspoređivanja poslova na klaudu, korišćen je višekriterijumski model sa performansama i budžetskim ograničenjima. U ovom modelu, uzeta je u obzir *minimizacija makespan indikatora*, kao i *minimizacija ukupnih troškova raspoloživog budžeta*.

Ovaj model je u literaturi iz domena računarstva poznat kao *model raspoređivanja resursa baziran na performansama i ograničenjima raspoloživog budžeta*. Više informacija o ovom modelu može se naći u Poglavlju 19.4.

U ovom skupu simulacija korišćeni su *veštački podaci*, generisani u okviru *CloudSim* 3.0.3 platforme. Za svako puštanje algoritma generisano je nasumično okruženje sa 50 *heterogenih virtuelnih mašina*, koje su implementirane „iznad" host računara u data centrima. Uzet je u obzir različit broj *heterogenih poslova* (od 100 do 500) različitih dužina. Osim ovoga, u simulacijama je razmatrana i *stopa pristizanja različitih zahteva*, tačnije razmatrane su performanse sistema kada pristiže 10 i 40 zahteva u jedinici vremena.

Karakteristike poslova, virtuelnih mašina i data centara su prikazani na Slikama 22-10 i 22-11.

Slika 22-10: *Karakteristike virtuelnih mašina, host računara i data centara u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam)*

Naziv entiteta	Parametar	Vrednost
VM	Broj virtuelnih mašina	50
	RAM	512 (jedinica: MB)
	CPU snaga obrade	1860,2660 (jedinica: MIPS)
	Kapacitet skladištenja	1 (jedinica: GB)
	Kapacitet mrežnog protoka	1000 (jedinica: Mbps)
	Polisa raspoređivanja poslova	bazirana na vremenu
	VMM (Hipervizor)	Xen
	Operativni sistem	Linux
	Broj CPU	1
Host	RAM	2 (jedinica: GB)
	Kapacitet skladištenja	10 (jedinica: GB)
	Kapacitet mrežnog protoka	1 (jedinica:Gbps)
	Polisa raspoređivanja virtuelnih mašina	bazirana na prostoru
Data centar	Broj data centara	10
	Broj host računara	10

Slika 22-11: *Karakteristike poslova u eksperimentima sa veštačkim skupom podataka (WOA-AEFS algoritam)*

Parametar	Vrednost
broj poslova	100–500
dužina posla	400–1000 (jedinica: MI)
veličina fajla	200-1000 (jedinica: MB)
veličina izlaza (memorije)	20–40 (jedinica: MB)

Osnovni kontrolni parametri *WOA-AEFS* metaheuristike podešeni su na sledeći način: *veličina populacije* na 30 (N = 30) i maksimalni broj iteracija u jednom puštanju algoritma na 1000 (*maxIter* = 1000). Vrednost parametra *limit* je podešena na 33 (round(1000/30)), dok su vrednosti dinamičkih parametara $\vec{a}$, *eir* i *a* adaptirani u toku jednog puštanja algoritma.

Ostali *WOA-AEFS* parametri su podešeni kao što je prikazano na Slici 22-12.

Slika 22-12: *Vrednosti kontrolnih parametara WOA-AEFS metaheuristike*

Naziv parametra	Vrednost
WOA-AEFS opšti parametri	
Veličina populacije (N)	30
Broj iteracija za svako puštanje algoritma ($maxIter$)	500
WOA parametri pretrage	
Početna vrednost parametra a	2.0
Dinamičko ponašanje parametra a	na osnovu jednačine (61)
ABC parametri eksploracije	
Parameter $limit$	17
Početna vrednost parametra eir	100
Dinamičko ponašanje parametra eir	na osnovu jednačine (69)
FA parametri pretrage	
Početna vrednost parametra nasumičnosti α	0.5
Dinamičko ponašanje parametra α	na osnovu jednačine (72)
Privlačnost na razdaljini r=0 β_0	0.2
Koeficijent absorpcije γ	1.0

U simulacijama, odvojeno je analizirana *makespan funkcija cilja, trošak i narušavanje roka izvršenja svih zadataka* (eng. *deadline violation constraint*) i prikazana je komparativna analiza sa *Min-Min* i *FCFS* (eng. *first come first serve*) heuristikama, kao i metaheuristikama *CSPSO* (eng. *cuckoo search particle swarm optimization*) i *PBACO* (eng. *performance budget ACO*).

Kao i u prethodnim simulacijama, da bi se bolje evaluirale performanse *WOA-AEFS*, u komparativnu analizu je uključen i osnovni *WOA* algoritam.

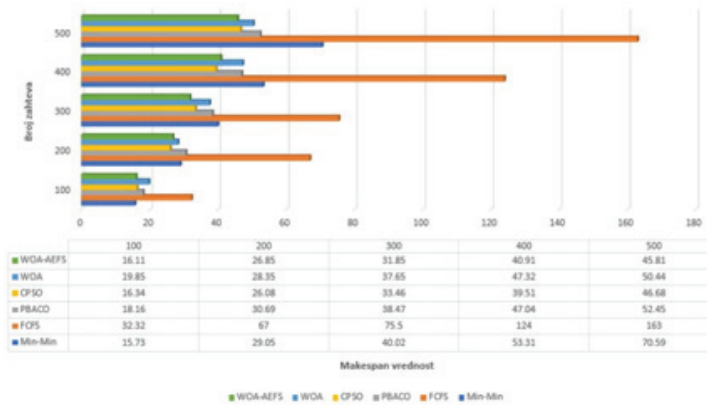
U svakoj simulaciji, *WOA* i *WOA-AEFS* su evaulirani u 100 nezavisnih puštanja i prosečni rezultati su zabeleženi. Osim toga, za svako puštanje algoritama, inicijalizovano je novo slučajno klaud okruženje sa karakteristikama prikazanim na Slikama 22-10 i 22-11.

Zbog bolje preglednosti, rezultati simulacija kategorisani su u tri grupe:

- makespan funkcija cilja,
- troškovi izvršavanja poslova i
- narušavanje roka izvršavanja zadataka.

Rezultati komparativne analize (dijagram i podaci) između *WOA-AEFS* i drugih pristupa za vrednost *makespan funkcije cilja*, kada je *stopa pristizanja poslova* podešena na 10 prikazana je na Slici 22-13.

Slika 22-13: *Komparativna analiza makespan funkcije cilja — WOA-AEFS vs. druge tehnike korišćenjem veštačkog skupa podataka (stopa dolaska 10)*



	100	200	300	400	500
WOA-AEFS	16.11	26.85	31.85	40.91	45.81
WOA	19.85	28.35	37.65	47.32	50.44
CPSO	16.34	26.08	33.46	39.51	46.68
PBACO	18.16	30.69	38.47	47.04	52.45
FCFS	32.32	67	75.5	124	163
Min-Min	15.73	29.05	40.02	53.31	70.59

Na osnovu rezultata, može da se vidi da je u proseku *WOA-AEFS* algoritam ostvario najbolju vrednost *makespan indikatora* u odnosu na sve druge pristupe uključene u analizu.

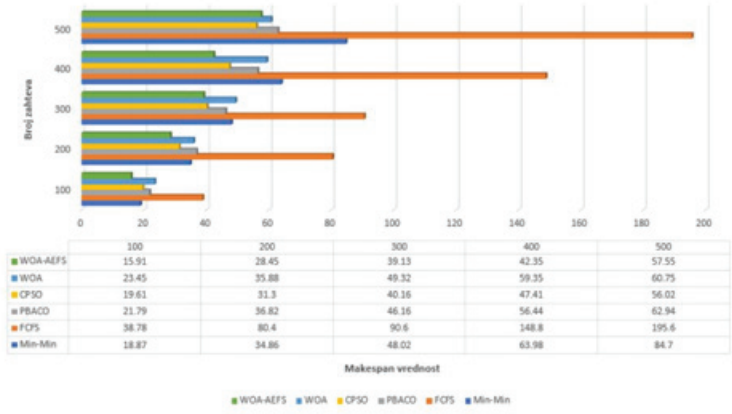
Kao druga najbolja metaheuristika pokazao se *CPSO* algoritam, koja u isto vreme postiže mnogo bolje performanse od osnovnog *WOA*. Metaheuristika *WOA-AEFS* je ostvarila bolje rezultate nego *CPSO* algoritam u test instancama sa 100, 300 i 500 poslova.

U simulacijama sa 200 i 400 zahteva, *CPSO* je ostvario bolje *makespan vrednosti* nego predloženi *WOA-AEFS*. Takođe, na osnovu prikazanih rezultata, može da se izvede zaključak da je *WOA-AEFS* značajno nadmašio osnovni *WOA* algoritam u svim test slučajevima.

Najveći rast performansi može da se primeti kod problema sa 100 poslova, gde je *WOA-AEFS* bolji od *WOA* za oko 19 %. U testovima sa 200, 300, 400 i 500 poslova, *WOA-AEFS* je poboljšao vrednost *makespan indikatora* originalnog *WOA* za oko 5%, 15%, 13% i 9%, respektivno. Na osnovu rezultata takođe se vidi da osnovni *WOA* generiše rešenja sličnog kvaliteta kao *PBACO* metaheuristika.

Rezultati komparativne analize (dijagram i podaci), kada je *stopa pristizanja poslova* podešena na 40 je prikazana na Slici 22-14.

Slika 22-14: *Komparativna analiza makespan funkcije cilja — WOA-AEFS vs. druge algoritme korišćenjem veštačkog skupa podataka (stopa pristizanja 40)*



	100	200	300	400	500
WOA-AEFS	15.91	28.45	39.13	42.35	57.55
WOA	23.45	35.88	49.32	59.35	60.75
CPSO	19.61	31.3	40.16	47.41	56.02
PBACO	21.79	36.82	46.16	56.44	62.94
FCFS	38.78	80.4	90.6	148.8	195.6
Min-Min	18.87	34.86	48.02	63.98	84.7

U eksperimentima sa *stopom pristizanja zahteva* od 40 klaudleta u jedinici vremena, *WOA-AEFS* je ostvario bolje rezultate nego *CPSO* u 4 od 5 test instanci. Samo u testovima sa 500 zahteva, *CPSO* je nadmašio *WOA-AEFS* za oko 2%.

Najznačajnije razlike u performansama između ova dva algoritma mogu da se uoče u testu sa 100 zahteva, gde je *WOA-AEFS* ostvario bolje rezultate za 19%.

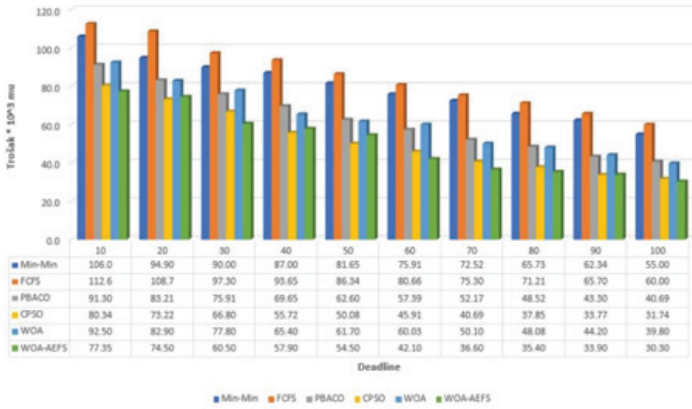
U simulacijama gde je stopa pristizanja poslova podešena na 40, *WOA-AEFS* je dokazao da je najbolja metaheuristika, dok je *CPSO* na drugom mestu. U svim test instancama, *WOA-AEFS* je nadmašio originalnu *WOA* metaheuristiku, koja je generisala slične rezultate kao *PBACO*.

U simulacijama sa obe stope dolaska zahteva (10 i 40), heuristički algoritmi su pokazali najgore performanse. Međutim, za pojedine test instance, Min-Min heuristika je ostvarila obećavajuće rezultate. Tako na primer, u testovima sa 100 i 200 klaudleta i obe stope pristizanja poslova (10 i 40), *Min-Min* heuristika je ostvarila bolje performanse nego *PBACO* algoritam. S druge strane, kada se uzmu u obzir svi izvršeni testovi, *Min-Min* heuristika je u proseku ostvarila dva puta bolje performanse nego *FCFS* heuristika.

U simulacijama je posebno razmatrana *minimizacija funkcije cilja ukupnih troškova*, gde su sprovedeni eksperimenti sa 200, 400 i 500 zahteva (zadataka) krajnjih korisnika i *promenljivim rokovima izvršavanja* (od 10 do 100).

Rezultati komparativne analize sa drugim pristupima za 200, 400 i 500 zadataka prikazani su na Slikama 22-15, 22-16 i 22-17.

Slika 22-15: *Komparativna analiza troškova sa 200 zahteva i promenljivim rokom izvršavanja zadataka —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*

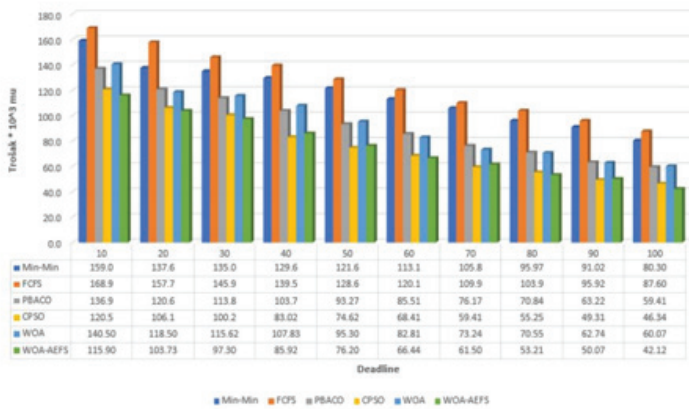


	10	20	30	40	50	60	70	80	90	100
Min-Min	106.0	94.90	90.00	87.00	81.65	75.91	72.52	65.73	62.34	55.00
FCFS	112.6	108.7	97.30	93.65	86.34	80.66	75.30	71.21	65.70	60.00
PBACO	91.30	83.21	75.91	69.65	62.60	57.39	52.17	48.52	43.30	40.69
CPSO	80.34	73.22	66.80	55.72	50.08	45.91	40.69	37.85	33.77	31.74
WOA	92.50	82.90	77.80	65.40	61.70	60.03	50.10	48.08	44.20	39.80
WOA-AEFS	77.35	74.50	60.50	57.90	54.50	42.10	36.60	35.40	33.90	30.30

Na svim prikazanim slikama (grafikonima), vertikalna osa prikazuje *ukupan trošak izražen u monetarnim jedinicama*, dok horizontalna osa označava rokove izvršavanja zadataka.

Kada se analiziraju rezultati predstavljeni na Slikama 22-15, 22-16 i 22-17, slično kao u analizi *makespan funkcije cilja*, može da se zaključi da u proseku, predloženi *WOA-AEFS* ostvaruje najbolje performanse.

Slika 22-16: *Komparativna analiza troškova sa 400 zahteva promenljivim rokom izvršavanja zadataka—WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	10	20	30	40	50	60	70	80	90	100
Min-Min	159.0	137.6	135.0	129.6	121.6	113.1	105.8	95.97	91.02	80.30
FCFS	168.9	157.7	145.9	139.5	128.6	120.1	109.9	103.9	95.92	87.60
PBACO	136.9	120.6	113.8	103.7	93.27	85.51	76.17	70.84	63.22	59.41
CPSO	120.5	106.1	100.2	83.02	74.62	68.41	59.41	55.25	49.31	46.34
WOA	140.50	118.50	115.62	107.83	95.30	82.81	73.24	70.55	62.74	60.07
WOA-AEFS	115.90	103.73	97.30	85.92	76.20	66.44	61.50	53.21	50.07	42.12

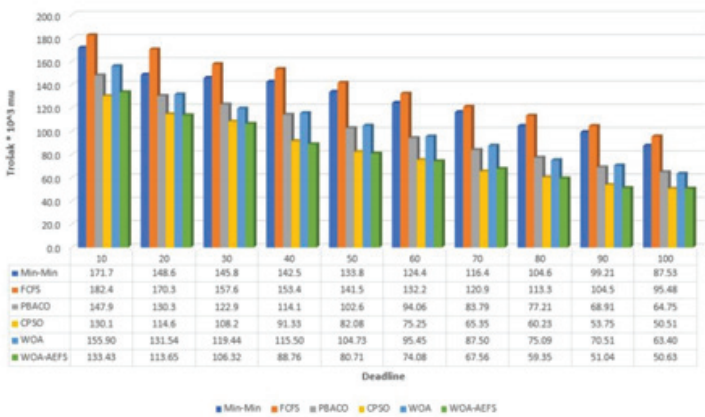
Drugi najbolji rezultat postiže metaheuristika *CPSO*, koja je u samo nekoliko test instanci nadmašila predloženi *WOA-AEFS*.

Neke od ovih instanci uključuju: 200 zahteva sa rokovima od 20, 40, 50 i 90, 400 zahteva sa rokovima od 40,50,70 i 90, kao i 500 zahteva sa rokovima od 10, 70 i 100.

Slično kao u prethodnim analizama, predloženi *WOA-AEFS* je za sve test instance ostvario mnogo bolje rezultate nego originalni *WOA*, i u isto vreme postigao bolji kvalitet rešenja i bolju konvergenciju. Osnovni *WOA* algoritam je ostvario slične performanse kao *PBACO* metaheuristika.

Kada se analiziraju *ukupni troškovi*, oba heuristička pristupa su postigla značajno lošije rezultate od metaheurističkih algoritama.

Slika 22-17: *Komparativna analiza troškova sa 500 zahteva i promenljivim rokom izvršavanja zadataka —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	10	20	30	40	50	60	70	80	90	100
Min-Min	171.7	148.6	145.8	142.5	133.8	124.4	116.4	104.6	99.21	87.53
FCFS	182.4	170.3	157.6	153.4	141.5	132.2	120.9	113.3	104.5	95.48
PBACO	147.9	130.3	122.9	114.1	102.6	94.06	83.79	77.21	68.91	64.75
CPSO	130.1	114.6	108.2	91.33	82.08	75.25	65.35	60.23	53.75	50.51
WOA	155.90	131.54	119.44	115.50	104.73	95.45	87.50	75.09	70.51	63.40
WOA-AEFS	133.43	113.65	106.32	88.76	80.71	74.08	67.56	59.35	51.04	50.63

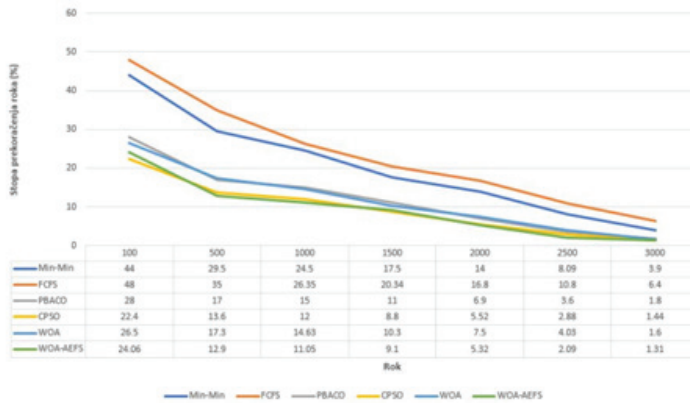
Poslednja analiza koja je urađena u simulacijama sa veštačkim skupom podataka je evaluiranje *stope narušavanja roka*.

Glavni cilj ove analize je *ocena kvaliteta usluga* (*QoS*) predložene metaheuristike. Simulacije su sprovedene sa 200, 400 i 500 zahteva i promenljivim rokovima izvršavanja zadataka (od 100 do 3000).

Rezultati komparativne analize narušavanja roka sa 200, 400 i 500 zahteva sa *ograničenjima roka izvršavanja zadataka* su prikazani na Slikama 22-18, 22-19 i 22-20, respektivno.

U svim prezentovanim slikama (grafikonima) *stopa narušavanja* prikazana je u procentima na ordinatnoj osi, dok su rokovi izvršavanja zadataka prikazani na apscisi.

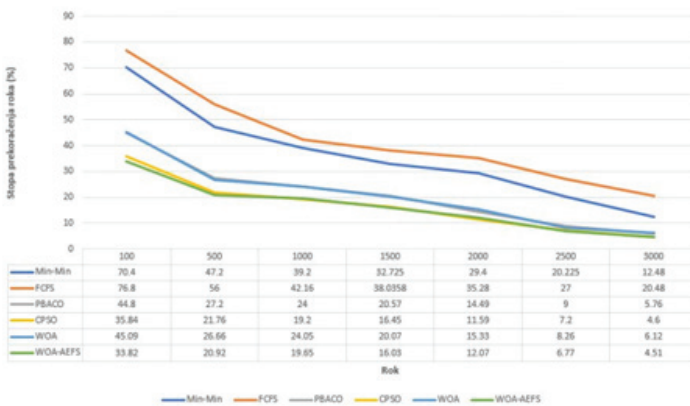
Slika 22-18: *Komparativna analiza narušavanja roka sa 200 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	100	500	1000	1500	2000	2500	3000
Min-Min	44	29.5	24.5	17.5	14	8.09	3.9
FCFS	48	35	26.35	20.34	16.8	10.8	6.4
PBACO	28	17	15	11	6.9	3.6	1.8
CPSO	22.4	13.6	12	8.8	5.52	2.88	1.44
WOA	26.5	17.3	14.63	10.3	7.5	4.03	1.6
WOA-AEFS	24.06	12.9	11.05	9.1	5.32	2.09	1.31

Na osnovu rezultata prikazanih na Slikama 22-18, 22-19 i 22-20 vidi se da je u proseku *WOA-AEFS* superiorniji od svih drugih heuristika i metaheuristika koje su obuhvaćene komparacijom.

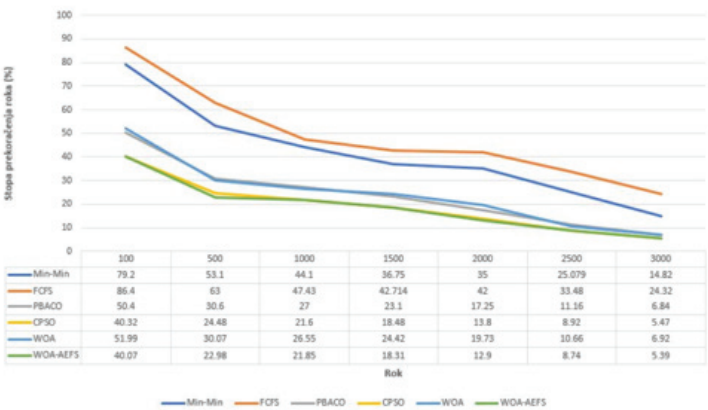
Slika 22-19: *Komparativna analiza narušavanja roka sa 400 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	100	500	1000	1500	2000	2500	3000
Min-Min	70.4	47.2	39.2	32.725	29.4	20.225	12.48
FCFS	76.8	56	42.16	38.0358	35.28	27	20.48
PBACO	44.8	27.2	24	20.57	14.49	9	5.76
CPSO	35.84	21.76	19.2	16.45	11.59	7.2	4.6
WOA	45.09	26.66	24.05	20.07	15.33	8.26	6.12
WOA-AEFS	33.82	20.92	19.65	16.03	12.07	6.77	4.51

Druga metaheuristika koja najmanje narušava rokove za izvršavanje zadataka je *CPSO*. Takođe, kao i u prethodnim testiranjima, osnovni *WOA* pristup je ostvario slične rezultate kao i *PBACO* metaheuristika. Oba heuristička pristupa su ostvarila mnogo lošije rezultate od svih metaheuristika.

Slika 22-20: *Komparativna analiza narušavanja roka sa 500 zahteva —WOA-AEFS vs. drugi algoritmi korišćenjem veštačkog skupa podataka*



	100	500	1000	1500	2000	2500	3000
Min-Min	79.2	53.1	44.1	36.75	35	25.079	14.82
FCFS	86.4	63	47.43	42.714	42	33.48	24.32
PBACO	50.4	30.6	27	23.1	17.25	11.16	6.84
CPSO	40.32	24.48	21.6	18.48	13.8	8.92	5.47
WOA	51.99	30.07	26.55	24.42	19.73	10.66	6.92
WOA-AEFS	40.07	22.98	21.85	18.31	12.9	8.74	5.39

Kada se uzmu u obzir sve tri analize - *makespan funkcije cilja, minimizacije ukupnih troškova i vrednost stope narušavanje rokova*, u simulacijama sa veštačkim skupom podataka, navodi se da u proseku *WOA-AEFS* postiže bolje performanse od svih drugih algoritama.

Drugi najbolji pristup je *CSPSO*, dok osnovni *WOA* postiže slične performanse kao *PBACO* algoritam. Heuristika *FCFS* postiže najgore rezultate, dok u pojedinim testovima performanse *Min-Min* heuristike mogu da se mere sa performansama metaheurističkih metoda.



PITANJA ZA RAZMATRANJE

1. Kako može da se unapredi originalni algoritam inteligencije rojeva?
2. Koji su nedostaci osnovnog *WOA* algoritma?
3. Kako je modifikovan *WOA* algoritam? Objasniti.
4. Kako su kategorizovani rezultati simulacije algoritma *WOA-AEFS*?
5. Objasniti komparativnu analizu algoritama za prvu kategoriju rezultata.
6. Objasniti komparativnu analizu algoritama za drugu kategoriju rezultata.
7. Objasniti komparativnu analizu algoritama za treću kategoriju rezultata.

Literatura

[1] B. Weldon, S. Rogers, *HP ProLiant Servers AIS: Official Study Guide and Desk Reference*, Prentince Hall, 2004, p. 576.

[2] C. M Daconta, *The Great Cloud Migration: Your Roadmap to Cloud Computing, Big Data and Linked Data*, Outskirts Press, 2013, p. 653.

[3] C. Stergiou, K. E. Psannis, B. Kim, B. Gupta, *Secure integration of IoT and Cloud Computing*, Future Generation Computer Systems, Elsevier, Vol. 78, Part 3, 2018, pp. 964-975.

[4] D. Alsmadi, V. Prybutok, *Sharing and storage behavior via cloud computing: Security and privacy in research and practice*, Computers in Human Behavior, Elsevier, Vol. 85, 2018, pp. 218-226

[5] D. Gibson, *CompTIA Security+ Get Certified Get Ahead: SY0-401 Study Guide*, CompTIA, 2017, p. 562.

[6] D. Gibson, *CompTIA Security+ Get Certified Get Ahead: SY0-501 Study Guide, CompTIA*, 2017, p. 586.

[7] E. Haletky, *VMware vSphere and Virtual Infrastructure Security: Securing the Virtual Environment*, Prentice Hall, 2009, p. 521.

[8] E.B. Rajsingh, J. Veerasamy, A. H. Alavi, J. P. Dinesh, *Advances in Big Data and Cloud Computing, Advances in Intelligent Systems and Computing*, Vol. 465, Springer, 2018, p. 413.

[9] ENISA, *Cloud Computing Benefits, risks and recommendations for information security, Rev. B*, Technical Report, 2012, URL: https://resilience.enisa.europa.eu/cloud-security-and- resilience/publications/cloud-computing-benefits-risks-and- recommendations-for-information-security, preuzeto: avgust 2018.

[10] F. Li, J. Cao, X. Wang, Y. Sun, Y. Sahni, *Enabling Software Defined Networking with QoS Guarantee for Cloud Applications*, 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, 2017, pp. 130-137, doi: 10.1109/CLOUD.2017.25

[11] Grupa autora, *HP Customer Case Study: Converged Infrastructure Jupiter Medical Center,* Hawlett Packard Company L.P., 2012, p. 4.

[12] Grupa autora, *HP Customer Case Study: Converged Infrastructure Jupiter Marina Bay Sands*, Hawlett Packard Company L.P., 2012, p. 4.

[13] H. Talal, H. Noor, S. Zeadally, A. Alfazi, Q. Z. Sheng, *Mobile cloud computing: Challenges and future research directions*, Journal of Network and Computer Applications, Elsevier, Vol. 115, 2018, pp. 70-85.

[14] Hutten D., *Cloud Computing Manipulation, Configuring and Accessing the Applications Online*, Amazon, 2017, p. 78.

[15] Internet izvor: *Announcing Amazon Elastic Compute Cloud (Amazon EC2) –beta*, objavljen: 2006, URL:https://aws.amazon.com/blogs/aws/amazon_ec2_beta/, preuzeto: avgust 2018.

[16] Internet izvor: *Datacentre decisions: Converged vs hyper-converged infrastructure*, objavljen: 2017, URL: https://www.computerweekly.com/feature/Storage-decisions-Converged-vs- hyper-converged-infrastructure, preuzeto: jul 2018.

[17] Internet izvor: *Google Company - Introducing Google App Engine + our new blog*, objavljen:2008, URL:http://googleappengine.blogspot.com/2008/04/introducing-google-app-engine-our-new.html, preuzeto: avgust 2018.

[18] Internet izvor: *Medium Corporation - A Brief History of Cloud Computing*, objavljen: 2018, URL: https://medium.com/threat-intel/cloud-computing- e5e746b282f5, preuzeto: jul 2018.

[19] Internet izvor: *Microsoft Corporation – How Microsoft Defines Cloud Computing*, objavljen: 2010, URL: https://blogs.technet.microsoft.com/ itinsights/2010/12/07/how-microsoft-defines-cloud-computing/, preuzeto: avgust 2018.

[20] Internet izvor: *Microsoft Corporation – What is a hybrid cloud?*, objavljen: 2018, URL: https://azure.microsoft.com/en-us/overview/what-is-hybrid- cloud-computing/, preuzeto: jul 2018.

[21] Internet izvor: *Microsoft Corporation – Windows Azure General Availability*, objavljen: 2013, URL: https://blogs.technet.microsoft.com/ microsoft_blog/2010/02/01/windows-azure-general-availability/, preuzeto: jul 2018.

[22] Internet izvor: *NIST definition for SaaS, PaaS, IaaS*, objavljen: 2013, URL: https://cloudinfosec.wordpress.com/2013/05/04/nist-definition-for-saas- paas-iaas/, preuzeto: avgust 2018.

[23] Internet izvor: *Public, Private and Hybrid Cloud – What's the big difference!*, objavljen: 2018, URL: https://junodavidk.com/public-private- hybrid-cloud-whats-big-difference/, preuzeto: avgust 2018.

[24] J. Weinman, *Cloudonomics*, Wiley Publishing, 2012, p. 391.

[25] Jirwan N., Singh A., Vijay S., *Review and Analysis of Cryptography Techniques*, International Journal of Scientific & Engineering Research, Vol. 4, Issue 3, 2013, pp. 1-6.

[26] K. Jackson, *OpenStack Cloud Computing Cookbook*, Packt Publishing, 2018, p. 398.

[27] K. Jeong, R. Figueiredo, K. Ichikawa, *PARES: Packet Rewriting on SDN- Enabled Edge Switches for Network Virtualization in Multi-Tenant Cloud Data Centers*, 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, 2017, pp. 9-17. doi: 10.1109/CLOUD.2017.11.

[28] L. C. Miller, *Client Virtualization for Dummies, HP and Intel Special Edition*, Wiley, 2013, p. 42.

[29] L. Carvalho, *Windows Server 2012 Hyper-V Cookbook, Packt Publishing*, 2012, p. 290.

[30] M. Gregg, *Cisa Exam Prep for Certified Information Systems Auditor,* Que Publishing, 2007, p. 887.

[31] R. LI, C. Shen, H. He, X. Gu, Z. Xu and C. Xu, *A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing*, IEEE Transactions on Cloud Computing, Vol. 6, No. 2, pp. 344-357, 2018, doi: 10.1109/TCC.2017.2649685.

[32] R. Munadi, E. Najwaini, A. Mulyiana, R. M. Rumani, *Design and implementation of VoIP service on Open IMS and Asterisk servers interconnected through ENUM server*, International Journal of Next- Generation Networks (IJNGN), Vol. 2, No. 2, 2012, pp. 1-12.

[33] R. Rese, F. Miller, L. Burdett, *Deploying Cloud Solutions for Small and Medium Business, First Edition*, Hawlett Packard Company L.P. & Certiport, 2013, p. 893.

[34] R. Resse, *Designing & Deploying Connected Devices Solutions for Small and Medium Business*, First Edition, Hawlett Packard Company L.P. & Certiport, 2012, p. 625.

[35] R. Resse, *Designing & Deploying Network Solutions for Small and Medium Business, First Edition*, Hawlett Packard Company L.P. & Certiport, 2012, p. 602.

[36] Rajan L., Google Cloud Platform Cookbook: *Implement, deploy, maintain, and migrate applications on Google Cloud*, Packt Publishing, 2018, p. 280.

[37] S. Askarnejad, M. Malekimajd, A. Movaghar, *Network and Application-Aware Cloud Service Selection in Peer-Assisted Environments*, IEEE Transactions on Cloud Computing, 2018, doi: 10.1109/TCC.2018.2865560.

[38] S. Bondurant, R. Reese, F. Miller, J. Jones, *Achieving Business Results Through Technology, First Edition,* Hawlett Packard Company L.P. & Certiport, 2013, p. 527.

[39] S. Jung, Y. Bae, W. Soh, *Web Performance Analysis of Open Source Server Virtualization Techniques*, International Journal of Multimedia and Ubiquitous Engineering , Vol. 6, No. 1, 2011, pp. 45-52.

[40] S. Liu, F.T.S. Chan, J. Yang, B. Niu, *Understanding the effect of cloud computing on organizational agility: An empirical examination*, International Journal of Information Management, Springer, Vol. 43, 2018, pp. 98-111.

[41] S. Soltani, P. Martin, K. Elgazzar, *A hybrid approach to automatic IaaS service selection*,

Journal of Cloud Computing, Springer, 2018, pp. 7-12, doi: https://doi.org/10.1186/s13677-018-0113-8

[42] S. Singh, *STAR: SLA-aware Autonomic Management of Cloud Resources*, IEEE Transactions on Cloud Computing, 2017, doi: 10.1109/TCC. 2017.2648788

[43] T. J. Bittman, *Clarifying private cloud computing*, Gartner Technical Report, 2010, p. 5.

[44] T. Lammle, *CompTIA Network+ Study Guide: Exam N10-006, 3rd Edition*, Sybex, 2015, p.960.

[45] Y. Wadia, *AWS Administration The Definitive Guide - Second Edition*, Packt Publishing, 2018, p. 358.

[46] I. Strumberger, E. Tuba, N. Bacanin, M. Tuba, *Dynamic tree growth algorithm for load scheduling in cloud environments*, In: 2019 IEEE Congress on Evolutionary omputation (CEC), June 2019, pp. 65–72, https: //doi.org/10.1109/CEC.2019.8790014.

[47] I. Strumberger, N. Bacanin, M. Tuba, E. Tuba, *Resource scheduling in cloud computing based on a hybridized whale optimization algorithm*, Applied Sciences Vol. 9, Issue 22, 2019, pp. 4893-4893, https://doi.org/10.3390/app9224893.

[48] I. Strumberger, M. Tuba, N. Bacanin, E. Tuba, *Cloudlet scheduling by hybridized monarch butterfly optimization algorithm,* Journal of Sensor and Actuator Networks Vol. 8, Issue 3, 2019, pp. 44-44, https://doi.org/10.3390/jsan8030044

[49] L. Zuo, L. Shu, S. Dong, C. Zhu, T. Hara, *A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing*. IEEE Access, Vol. 3, 2015, pp. 2687–2699.

[50] S. Zahoor, S. Javaid, N. Javaid, M. Ashraf, F. Ishmanov, M.K. Afzal, *Cloud–fog–based smart grid model for efficient resource management*, Sustainability, Vol. 10 Issue 6, 2016, pp. 2079.

[51] I. Ibrahim Mahmood, *Task scheduling algorithms in cloud computing: A review,* Turkish Journal of Computer and Mathematics Education (TURCOMAT), Vol. 12, No. 4, 2021, pp. 1041-1053.

[52] A. Elaziz Mohamed, I. Attiya, *An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing*, Artificial Intelligence Review, Vol. 54, No. 5, 2021, pp. 3599-3637.

[53] Houssein E. H., Ahmed G. G., Yaser M. W., Ponnuthurai N. S., *Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends*, Swarm and Evolutionary Computation, 2021, pp. 100841.

[54] I.Strumberger, *Unapređenje raspoređivanja poslova i balansiranja opterećenja u klaud okruženju primenom metaheuristika inteligencije rojeva*, Univerzitet Singidunum, doktorska disertacija, 2019, p.189.

CIP - Каталогизација у публикацији
Народна библиотека Србије, Београд

004.722(075.8)

ШТРУМБЕРГЕР, Ивана, 1989-
Klaud računarstvo / Ivana Štrumberger, Nebojša Bačanin Džakula. - 2. izd. - Beograd : Univerzitet Singidunum, 2021 (Beograd : Birograf). - XXVII, 526 str. : ilustr. ; 24 cm

Tiraž 1.300. - Bibliografija: str. 523-526.

ISBN 978-86-7912-772-3

1. Бачанин Џакула, Небојша, 1983- [аутор]

а) Технологија облака

COBISS.SR-ID 53182473

© 2021
Sva prava zadržana. Nijedan deo ove publikacije ne može biti reprodukovan u bilo kom vidu i putem bilo kog medija, u delovima ili celini bez prethodne pismene saglasnosti izdavača.

www.singidunum.ac.rs

9 788679 127723



Ivana Štrumberger
Nebojša Bačanin-Džakula

KLAUD RAČUNARSTVO

Glavni cilj ovog udžbenika, koji predstavlja rezultat višegodišnjeg iskustva autora na izučavanju ove oblasti, je upoznavanje čitalaca sa osnovama klaud računarstva. Tekst je propraćen primerima i praktičnim simulacijama, koje imaju za cilj da čitaoce što više približe konceptu klaud računarstva i da ih osposobe za samostalan rad u realnom, produkcionom okruženju.

Univerzitet Singidunum