Projektiranje sigurnih i visokoraspoloživih sustava pomoću Amazon AWS-a

Plaftarić, Marko

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Plaftarić

PROJEKTIRANJE SIGURNIH I VISOKO RASPOLOŽIVIH SUSTAVA POMOĆU AMAZON AWS-A

DIPLOMSKI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Plaftarić

Matični broj: 42569/13-R

Studij: ***Organizacija poslovnih sustava***

PROJEKTIRANJE SIGURNIH I VISOKO RASPOLOŽIVIH SUSTAVA POMOĆU AMAZON AWS-A

DIPLOMSKI RAD

Mentor:

Doc.dr.sc. Tonimir Kišasondi

Varaždin, ožujak 2018

Marko Plaftarić

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Cilj rada jest demistificirati pojam računarstva u oblaku primjenom javnog uslužnog sustava Amazon AWS-a. Definirani su osnovni gradivni blokovi AWS infrastrukture kao servisa na kojima se zasnivaju uzorci dizajna iz najbolje prakse. Rad je fokusiran na uzorke dizajna koji povećavaju raspoloživost informacijskog sustava. Uz njih obrađeni su osnovni uzorci dizajna i uzorci za relacijske baze podataka. U praktičnom djelu rada dan je prikaz procedure zakupa i konfiguracije AWS infrastrukture. Korišten je uzorak dizajna za dinamičko skaliranje broja servera s dodatkom redundancije na razini podatkovnog centra. Na kraju rada prikazani su rezultati izvršenih testova opterećenja nad zakupljenom infrastrukturom. Iz rezultata je vidljiv proces skaliranja broja servera u odnosu na opterećenje, te poboljšanja iz aspekta raspoloživosti i ostalih performansi poslužitelja.

Ključne riječi: računarstvo; oblak; cloud; Amazon; Web; Services; AWS; uzorak; dizajn; raspoloživost; sigurnost;

Sadržaj

1. Uvod

U tradicionalnom fizičkom podatkovnom centru potrebno je vršiti aktivnosti poput naručivanja hardvera, sastavljanja, instalacije, konfiguriranja, održavanja, postavljanja sigurnosnih mjera i slično. Poduzeća već dugo traže način kako da te aktivnosti povjere drugom pružatelju usluge. Rješenje koje su tražila zove se računarstvo u oblaku i ono će zauvijek promijeniti način na koji organizacije koriste infrastrukturu.

Računarstvo u oblaku rapidno se razvija i otključava mogućnost inovacija u svim aspektima IT-a bez tradicionalnih ograničenja. Model plaćanja po korištenju zamjenjuje velike inicijalne troškove nabave infrastrukture. Uz to podiže se razina agilnosti u organizaciji i brzina dolaska na globalno tržište. Takav pristup pogotovo odgovara manjim i modernim *Start-Up* organizacijama koje s minimalnim ulaganjima mogu testirati i inovirati primjenom infrastrukture u oblaku.

U nastavku rada nastoji se demistificirati pojam računarstva u oblaku. Detaljnije će se obraditi na konkretnom primjeru Amazon AWS-a, lidera na tržištu računarstva u oblaku. Slijedi mnoštvo karakteristika i posebnosti AWS-a, kao i opis osnovnih servisa iz područja infrastrukture kao servisa. Njih se nadalje koristi kao gradivne elemente u uzorcima dizajna čijom primjenom arhitekt oblaka može riješiti različite probleme i utjecati na razne performanse sustava u oblaku, sve po najboljim praksama.

Visoka raspoloživost srž je ovog rada pa je u praktičnom djelu prikazano kako je lako primjenom AWS servisa zakupiti i konfigurirati infrastrukturu. Korišten je uzorak dizajna za dinamičko skaliranje broja servera s dodatkom redundancije na razini podatkovnog centra. Na kraju rada prikazani su rezultati izvršenih testova opterećenja nad zakupljenom infrastrukturom.

Korištene metode i tehnike za razradu teme svode se na standardne istraživačke aktivnosti primjenom računala s mrežnom konekcijom. Za praktični dio rada korišten je program MobaXterm za SSH konekciju na AWS poslužitelje, dok je program Siege korišten za testiranje opterećenja web poslužitelja.

2. Računarstvo u oblaku

2.1. Evolucija računarstva u oblaku

Postojalo je vrijeme kada je svako kućanstvo, selo, farma ili grad imalo svoj vlastiti bunar. Danas nam zajedničke komunalne usluge daju pristup čistoj vodi jednostavnim korištenjem slavine. Računarstvo u oblaku funkcionira na sličan način. Baš kao i voda iz slavine u vašem domaćinstvu, usluge računarstva u oblaku mogu se brzo uključiti ili isključiti po potrebi. Kao i kod vodoopskrbnog poduzeća, postoji tim specijaliziranih stručnjaka koji rade na tome da je usluga sigurna i dostupna dvadeset četiri sata dnevno. Kada se slavina ne koristi, voda se štedi i ne plaćate naknadu za resurs koji trenutno ne trebate. (Kundra, 2010)

Računarstvo u oblaku je rezultat evolucije koja je trajala dugi niz godina s početkom još od prvih računala. Naturalnom progresijom od ere centraliziranih računalnih jedinica, preko ere distribuiranih klijent-server sustava s osobnim računalom kao osnovnom jedinicom, do ere Interneta, gdje su poduzeća bila u mogućnosti spojiti se s ostatkom svijeta kroz globalno rasprostranjenu mrežu računala. (Kavis, 2014)

2.2. Definicija računarstva u oblaku

Američka vladina organizacija za standarde i tehnologiju - NIST (eng. *National Institute of Standards and Technology*) definirala je računarstvo u oblaku na sljedeći način:

Računarstvo u oblaku je model koji omogućava sveprisutan i prikladan mrežni pristup zajedničkom skupu prilagodljivih računalnih resursa (npr. računalne mreže, poslužitelji, aplikacije, usluge i pohrana podataka) koji se mogu brzo pribaviti i otpustiti uz minimalni napor za upravljanje od strane korisnika i pružatelja usluga. Model računarstva u oblaku sastoji se od pet osnovnih karakteristika, tri modela usluga i četiri modela implementacije.

2.3. Karakteristike računarstva u oblaku

Osnovne karakteristike računarstva u oblaku su:

- **Pružanje usluge na zahtjev korisnika** (eng. *On-demand self-service*) - Korisnik može po potrebi jednostrano zakupiti i pribaviti računalne mogućnosti, kao što su poslužitelji i mrežna pohrana podataka, potpuno automatski, bez ljudske interakcije s strane pružatelja usluga.

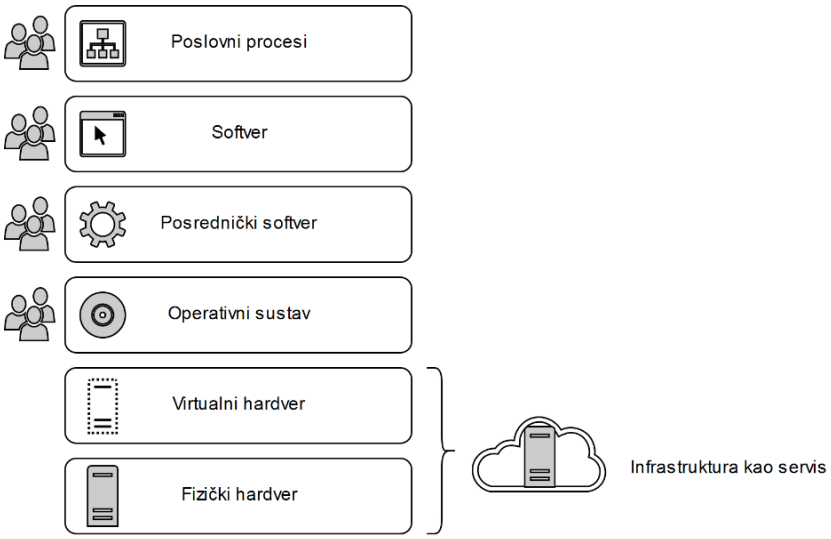
- **Širok mrežni pristup** (eng. *Broad network access*) - Mogućnosti računarstva u oblaku su dostupne putem računalne mreže kojoj se pristupa standardnim mehanizmima koji promiču heterogene "tanke" ili "debele" klijentske platforme (npr. mobilni telefoni, tableti, prijenosna računala i radne stanice).
- **Udruživanje resursa** (eng. *Resource pooling*) - Računalni resursi pružatelja usluge računarstva u oblaku su udruženi kako bi poslužili više korisnika na fizičkim resursima primjenom virtualizacije. Virtualni resursi bivaju dinamički dodjeljivani korisnicima ovisno o njihovoj potražnji. Stvara se osjećaj nezavisnosti o lokaciji računalnih resursa jer korisnik generalno nema kontrolu niti znanje o točnoj fizičkoj lokaciji zakupljenih resursa, ali može odrediti lokaciju na višoj razini apstrakcije (npr. država, regija ili podatkovni centar).
- **Brza elastičnost** (eng. *Rapid elasticity*) - Mogućnosti računarstva u oblaku mogu se elastično zakupljivati i otpuštati. Adekvatnom arhitekturom mrežnih resursa može se postići automatsko i rapidno skaliranje zakupljenih mrežnih resursa u skladu s potražnjom sustava. Na primjer, veliko opterećenje sustava potakne zakupljivanje novih resursa kako ne bi došlo do prestanka normalnog funkcioniranja sustava. Nakon nekog vremena potražnja se smanji, sustav otpusti višak računalnih resursa i vrati se u početno stanje.
- **Odmjerena usluga** (eng. *Measured service)* - Sustavi u oblaku automatski kontroliraju i optimiziraju korištenje resursa primjenom mogućnosti mjerenja na nekoj razini apstrakcije prikladnoj vrsti usluge koja se koristi (npr. pohrana, procesiranje, propusnost, broj aktivnih korisničkih računa i drugo). Korištenje resursa u oblaku može se na transparentan način pratiti i kontrolirati od strane korisnika i pružatelja usluga. Važno je napomenuti da korisnik plaća naknadu samo za onaj iznos koji je stvarno potrošen.

2.4. Modeli pružanja usluga računarstva u oblaku

Postoje tri glavna modela pružanja usluga računarstva u oblaku: infrastruktura kao servis, platforma kao servis i softver kao servis. Svaki od njih pruža drugačiju razinu apstrakcije tehnološkog stoga koji olakšava implementaciju i razvoj informacijskih sustava. Zajedničko im je da imaju karakteristike računarstva u oblaku definirane u prethodnom poglavlju i da prebacuju fokus s upravljanja infrastrukturom na ključne kompetencije vlastitog poslovanja. Za svaki od modela dana je definicija, opseg djelovanja pružatelja i korisnika usluge, primjena i primjeri pružatelja usluge računarstva u oblaku. Na kraju poglavlja dan je osvrt na pojavu novih modela koji koriste "kao servis" u nazivu proizvoda.

2.4.1. Infrastruktura kao servis

Prema NIST-u (National Institute of Standards and Technology [NIST], 2011) definicija infrastrukture kao servisa (eng. *Infrastructure as a Service* [IaaS]) navodi da je to model računarstva u oblaku koji korisniku pruža mogućnost da zakupi fundamentalne računalne resurse poput procesiranja, skladištenja podataka, računalnih mreža i drugih resursa. Oni omogućavaju implementaciju i pokretanje proizvoljnog softvera kao što su operativni sustavi i aplikacije. Korisnik nema kontrolu nad fizičkom infrastrukturom računalnog oblaka, već snosi odgovornost i upravlja s operativnim sustavima, pohranom podataka i aplikacijama. Iznimka su neki mrežni uređaji nad kojima ima određenu razinu kontrole (npr. vatrozid).



Slika 1: Infrastruktura kao servis (Prema: Fehling, Leymann, Retter, Schupeck i Arbitter, 2014)

U tradicionalnom fizičkom podatkovnom centru korisnici vrše aktivnosti poput naručivanja hardvera, sastavljanja, instalacije, konfiguriranja, skladištenja, servisiranja, održavanja, sigurnosti i slično. Korištenjem IaaS računarstva u oblaku te aktivnosti moguće je povjeriti pružatelju usluge (eng. *Outsourcing*). Poduzeća su tražila način kako da to postignu još od 1980-ih, a dvije ključne tehnologije koje su to omogućile su: Internet i virtualizacija.

Korisnik nema kontrolu nad fizičkom infrastrukturom računalnog oblaka kao što možemo vidjeti na priloženoj slici (Slika 1). Ikone osoba s lijeve strane predstavljaju nad kojim segmentima tehnološkog stoga korisnici imaju kontrolu, dok s desne strane je moguće vidjeti na koje segmente se IaaS model računarstva u oblaku odnosi. Korisnik IaaS-a snosi odgovornost i upravlja sa svim elementima od operacijskog sustava do vlastitih poslovnih procesa. Pružatelj IaaS usluge je zadužen za fizičku i virtualiziranu infrastrukturu.

Dostupnost komunikacije s visokom propusnošću preko Interneta omogućila je jednostavno upravljanje računalima s udaljenih lokacija. To je eliminiralo potrebu da

zaposlenici drugih poduzeća moraju biti fizički pristupni u podatkovnom centru i dalo priliku pružatelju usluge da efikasnije organizira vlastite poslovne procese i iskoristiti geografske lokacije gdje su niži operativni troškovi.

Virtualizacija je tehnika kojom se računalni resursi apstrahiraju i enkapsuliraju tako da odgovaraju određenoj primjeni. Virtualizacijom se postiže izoliranost, elastičnost i bolja iskorištenost infrastrukture udruživanjem resursa jer omogućava istovremeno korištenje u različitim sustavima. U kratko, to je tehnika u kojoj jedan računalni resurs djeluje kao zasebna jedinica, a zapravo nije fizički resurs već je dio jednog ili više fizičkih resursa.

Aktivnosti upravljanja i korištenja virtualnog hardvera u IaaS-u su apstrahirane i dostupne kao kolekcija servisa koje korisniku olakšavaju i automatiziraju posao. Dostupni načini komunikacije sa servisima zavise o pružatelju usluge, ali najčešće se izvršava koristeći:

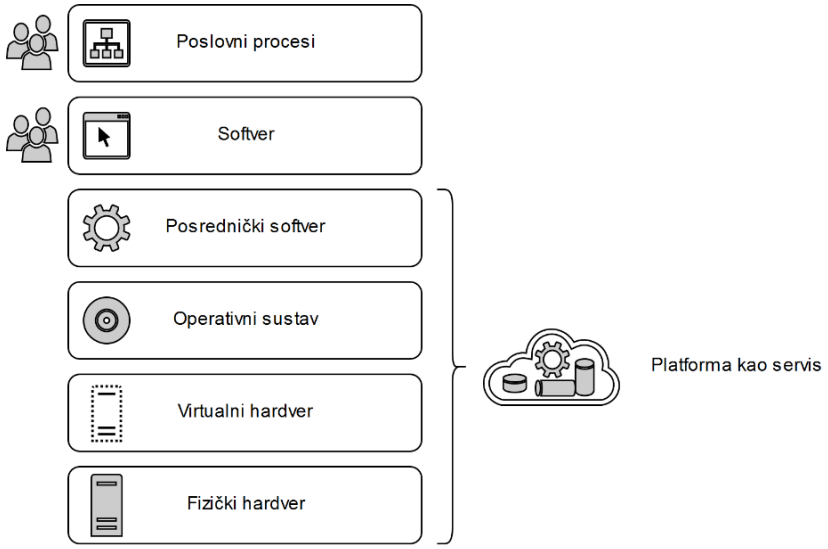
- aplikacijsko programsko sučelje (eng. *Application Programming Interface* [API])
- set razvojnih alata (eng. *Software Development Kit* [SDK])
- mrežno korisničko sučelje (eng. *web User Interface* [UI])

Prema (G2 Crowd, 2017) neki od vodećih IaaS pružatelja usluge su: Amazon (S3, VPC), Microsoft Azure, DigitalOcean, Rackspace, Google Compute Engine i drugi. Valja spomenuti da postoje i rješenja otvorenog koda (eng. *Open Source*) poput: Apache CloudStack, Eucalyptus, OpenNebula, OpenStack i dr.

U kratko, IaaS omogućava korisnicima usluge virtualnog podatkovnog centra i prebacuje im organizacijski fokus s upravljanja infrastrukturom na razvoj i korištenje softverskih proizvoda kao podrška poslovnim procesima.

2.4.2. Platforma kao servis

Prema NIST-u (2011) definicija platforme kao servisa (eng. *Platform as a Service* [PaaS]) navodi da ona pruža korisniku mogućnost da na infrastrukturu računalnog oblaka implementira kupljena ili kreirana softverska rješenja pomoću programskih jezika, programskih biblioteka, usluga i alata podržanih od strane pružatelja usluge. Korisnik nema kontrolu nad fizičkom infrastrukturom računalnog oblaka, operativnim sustavom ili skladištenjem podataka, već ima kontrolu nad implementiranom aplikacijom i vezanim konfiguracijskim postavkama.



Slika 2: Platforma kao servis (Prema: Fehling i sur., 2014)

PaaS je nadogradnja na IaaS koja pruža apstrahirane standardne funkcije kao servise. Olakšava implementaciju aplikacija bez velikih troškova i kompleksnosti poput kupovanja, postavljanja i upravljanja infrastrukturom. Recimo da programeri razvijaju vlastiti visoko raspoloživi sustav za razmjenu poruka, trebali bi napisati programski kod za asinkronu razmjenu poruka, skaliranje baze podataka i mnoge druge. PaaS pruža te i druge mogućnosti kao servise (API-e) tako da se programeri mogu baviti poslovnom logikom umjesto da na novo razvijaju postojeće zahtjevne servise.

Servise koje PaaS nudi dobavljaju se preko centraliziranog mrežnog sučelja dohvatljivog putem Interneta koje najčešće ima oblik mrežnog tržišta (eng. *Online marketplace*). Servisi bivaju ocjenjeni i sortirani po kvaliteti te je moguće odabrati one programske proizvode koji su provjereni i dovoljno zreli za implementaciju.

Korištenjem PaaS-a razvojni inženjeri se odriču stupnja fleksibilnosti kao što to Slika 2 prikazuje, jer ovise o alatima i softveru kojeg pružatelj usluge nudi. Također često nemaju kontrolu strukturom računalnog oblaka, mrežom, sustavom pohrane, operacijskim sustavom, poslužiteljem, nad nižom razinom izvođenja aplikacija poput broja dretvi i količinom predmemorije (eng. *Cache*). PaaS pružatelj usluge ima potpunu kontrolu nad tom razinom i moguće je da regulira korištenje računalnih resursa kako bi se platforma podjednako skalirala svim korisnicima. Dio nad kojim korisnici imaju potpunu kontrolu su njihove razvijene aplikacije, te ponekad postoji i mogućnost nadzora okolinske konfiguracije.

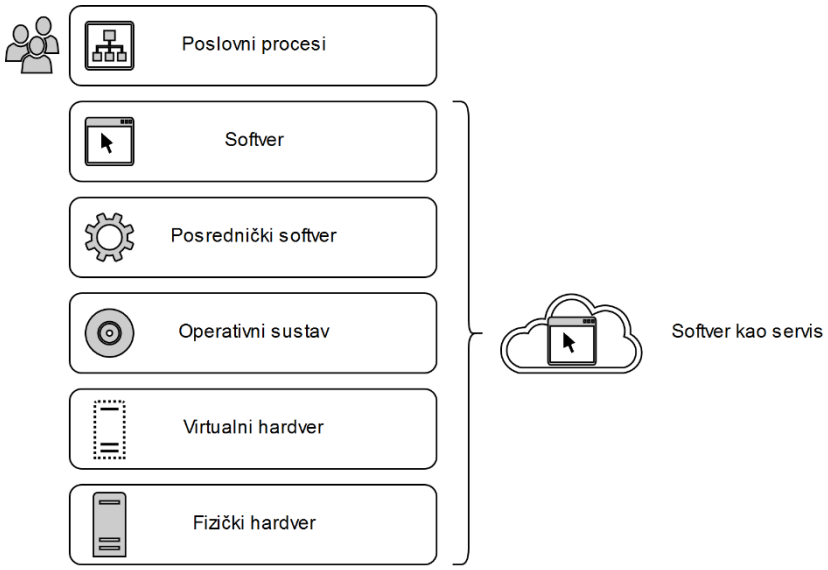
Omogućavanje većeg broja raspoloživih tehnologija je ključno za širenje popularnosti računarstva u oblaku. U početcima razvoja pružatelji su ograničavali korištenje tehnologija, poput Microsoft Azure na .NET i Google Apps Engine na programski jezik Python. Nova generacija PaaS-a, uključujući Azure i Google, pruža korisnicima veliki broj izbora tehnologija

poput: PHP, Ruby, Python, Node.js i mnoge dr. Neki od vodećih organizacije koje se bave pružanjem PaaS usluge su Amazon AWS (EC2, Elastic Beanstalk, Lambda), Microsoft Azure, Salesforce (Force, Heroku), RedHat OpenShift, Google App Engine i drugi. (G2 Crowd, 2017) Postoje i PaaS rješenja otvorenog koda kao na primjer: Cloud Foundry, Cloudify, OpenShift, Stackato, W2SO Stratus i dr.

U kratko, ono što je IaaS za infrastrukturu, to je PaaS za aplikacije. Korisnik sam gradi svoje aplikacije, a korištenjem API-a velikog broja neovisnih proizvođača (eng. *Third-party*), razvojni inženjeri mogu brzo i jednostavno razviti svoj proizvod.

2.4.3.Softver kao servis

Prema NIST-u (2011) definicija softvera kao servisa (eng. *Software as a service* [SaaS]) navodi da ona pruža korisniku mogućnost uporabe aplikacija, kreiranih od strane pružatelja usluga, koje se izvode na infrastrukturi računarstva u oblaku. Aplikacijama je moguće pristupiti s raznih klijentskih uređaja kroz "tanke" klijente, kao što su web preglednici, ili kroz sučelja programa. Korisnik nema kontrolu nad fizičkom infrastrukturom računalnog oblaka, operativnim sustavom, skladištenjem podataka ili pojedinim mogućnostima aplikacije. Iznimka su ograničene konfiguracijske postavke aplikacije. (NIST, 2011)



Slika 3: Softver kao servis (Prema: Fehling i sur., 2014)

Na vrhu tehnološkog stoga nalazi se SaaS model računarstva u oblaku. SaaS je platforma koja omogućava dostupnost gotovih aplikacija putem Interneta u obliku usluga koje se unajmljuju prema potrebi. Na priloženoj slici (Slika 3) vidimo da je pružatelj usluge zadužen je za svu infrastrukturu, programsku logiku, i sve potrebno za dostavljanje programskog

proizvoda kao usluge, dok je na korisnicima usluge da upravljaju ostalim korisnicima i konfiguriraju osobne preferencije unutar definiranih granica. Za korištenje SaaS usluga korisnicima je dovoljan Internet preglednik s kojim svako računalo već dolazi pred instalirano, dok arhitektura SaaS usluga omogućava simultani pristup tisućama istovremenih korisnika.

SaaS rješenja se vrlo često koriste za sekundarne kompetencije organizacije. Na taj način izbjegava se trošak kupovine, instalacije, nadogradnje i održavanja programa na računalima. Organizacije radije plate pretplatu i odmah počnu koristiti programski proizvod. Neki od popularnijih SaaS usluga su: tekstualni, tablični, kalendarski, kolaboracijski, komunikacijski, računovodstveni i ostali poslovni programi. Valja dodatno istaknuti *Customer Relationship Management* [CRM] i *Enterprise Resource Planning* [ERP] rješenja čije implementacije su uglavnom jako dugotrajne i skupe, te održavanje takvog programskog proizvoda na vlastitoj infrastrukturi može biti također skupo i komplicirano. Povjeravanje tih usluga vanjskim programima može organizacijama donijeti velike uštede i prebaciti fokus na ključne kompetencije organizacije.

Svim SaaS rješenjima je zajedničko da dolaze uz dokument pod imenom Ugovor o razini usluge (eng. *Service Level Agreement* [SLA]). To je službena obveza koju pružatelj usluge daje korisnicima usluge i njome su definirani aspekti poput kvalitete usluge, raspoloživosti usluge i razine odgovornosti. Temeljna stavka SLA ugovora je ona koja govori da pružatelj mora isporučiti uslugu pod uvjetima koji su unutar SLA definirani i prihvaćeni od obije strane. SLA pokušava napravit balans između kvalitete i kvantitete isporučene usluge i cijene usluge. Na primjer, telekomunikacijske organizacije često definiraju SLA kao uvjet ugovora gdje na jasan način definiraju postotak raspoloživosti usluge u nekom vremenskom periodu, propusnost, dozvoljene smetnje, mjere za oporavak od katastrofe poput ciljanog vremena oporavka od kvara (eng. *Recovery time objective* [RTO]), ciljanog objekta oporavka nakon kvara (eng. *Recovery point objective* [RPO]) i mnoge druge vrijednosti.

U nastavku slijedi kratak i nepotpun popis popularnijih SaaS rješenja u 2017. godini prema osobnom mišljenju, ne nužno navedenim redoslijedom: Microsoft Office365, Google Apps, Salesforce CRM, IBM SmartCloud, JiRA Atlassian Service Desk Platform, Slack Collaboration Platform, Dropbox File Sharing, WebEx, Github i mnogi drugi.

U kratko, SaaS nudi gotove aplikacije na zahtjev korisnika koje se pružaju s centraliziranog mjesta na Internetu primjenom računarstva u oblaku.

2.4.4.Ostali "kao servis" modeli

Od kada se SaaS model računarstva u oblaku 2001. godine pojavio na tržištu i stekao popularnost koja i danas raste, pojavila se gomila akronima s *as-a-Service* dodatkom koje su

marketinške tvrtke počele koristiti za praktički bilo koju uslugu koja postoji. U prethodnim modelima taj dodatak je označavao da se radi o modelu računarstva u oblaku, koji funkcionira na bazi pretplate. Danas u 2017. godini to ne mora biti takav slučaj, već predstavlja bilo koju uslugu na bazi pretplate, bila ona primjenom računarstva u oblaku ili ne. U nastavku Tabela 1 daje nepotpun popis popularnih "kao servis" modela prema (Tribe, 2016).

Tabela 1: Ostali "kao servis" modeli (Prema: Tribe, 2016)

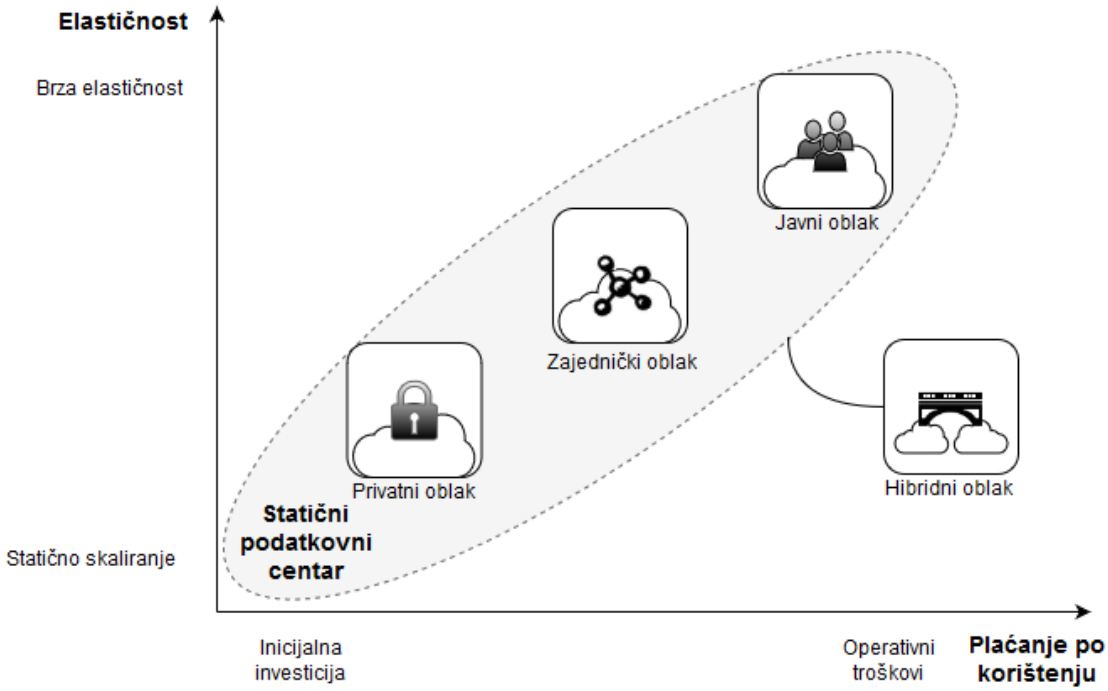
Backend as a Service	Desktop as a Service	Network as a Service
Backup as a Service	Device as a Service	Notebook as a Service
Blockchain as a Service	Environment as a Service	Operations as a Service
Building as a Service	Framework as a Service	Platform as a Service
Cloud as a Service	Hardware as a Service	Ransomware as a Service
Commerce as a Service	Identity as a Service	Software as a Service
Communications as a Service	Infrastructure as a Service	Storage as a Service
Compiler as a Service	Knowledge as a Service	Surface as a Service
Compliance as a Service	Linux as a Service	Understanding as a Service
Containers as a Service	Location as a Service	Video as a Service
Content as a Service	Logging as a Service	Virtualization as a Service
Country as a Service	Management as a Service	Windows as a Service
Data as a Service	Messaging as a Service	Workspace as a Service
Database as a Service	Monitoring as a Service	

2.5. Modeli implementacije računarstva u oblaku

Modele implementacije računalnog oblaka možemo razlikovati po tome kakve grupe korisnika mu pristupaju i po razini dijeljenja resursa oblaka između korisnika. Javnom modelu računalnog oblaka svi imaju pristup, privatnom samo jedna grupa ljudi, zajedničkom kontrolirani broj grupa ljudi i hibridnom kombinacija navedenih.

Ograničenje broja korisnika koji pristupaju oblaku i dijeljenju IT resursa ima značajan utjecaj na svojstva oblaka prikazana različitim modelima implementacije, posebice u vezi s udruživanjem resursa, brzom elastičnosti i odmjerenom uslugom (plaćanjem po uporabi). Veći broj korisnika smanjuje efekte promjene radnog opterećenja za pojedinog korisnika nad cjelokupnim sustavom kojeg pružatelj održava. Razlog tome je što sumirano opterećenje sustava izbalansira opterećenje jednog korisnika. Kada bi manje korisnika koristilo sustav, promjena opterećenje jednog korisnika imalo bi veći utjecaj na cjelokupni sustav. Shodno tome pružatelj usluge bi teže osigurao brzu elastičnost, jer je zajedničko korištenje sustava manje efektivno i model plaćanja po uporabi bi prestao biti isplativ zbog nedovoljne iskorištenosti sustava. (Fehling i sur., 2014)

Slika 4 prikazuje odnos modela implementacije računarstva u oblaku s obzirom na uobičajenu razinu elastičnosti i plaćanja po uporabi. Javni model s najviše korisnika može pružiti najveću razinu elastičnosti i u potpunosti plaćanje po korištenju. Zajednički model s manjim brojem korisnika, najčešće partnerske organizacije, pruža smanjene pogodnosti uglavnom zbog stabilnih poslova koje organizacije izvršavaju s manjom promjenom opterećenja. Također je u tom modelu česta pojava ulaganje unaprijed za implementaciju računalnog oblaka. U privatnom oblaku taj efekt još više dolazi do izražaja. Statički podatkovni sustav je na dnu slike i vidljivo je da tada sustav prestaje biti elastičan i model financiranja je u potpunosti u obliku plaćanja u naprijed. Hibridni model sadrži sva navedena svojstva pošto je on kombinacija ostalih modela. Valja napomenuti da se prikazane pogodnosti za javni računalni oblak mogu smanjiti ako ima nedovoljan broj korisnika jer ovisi o ekonomiji razmjera. (Fehling i sur., 2014)



Slika 4: Odnos elastičnosti sustava i plaćanja po korištenju za različite modele implementacije računalnog oblaka (Prema: Fehling i sur., 2014)

2.5.1. Javni model implementacije računarstva u oblaku

U javnom modelu implementacije računarstva u oblaku korištenje usluga otvoreno je za javnost, bez obzira dali je riječ o organizacijama ili pojedincima, a cijela infrastruktura je u vlasništvu pružatelja usluge. Riječ javno se ne odnosi na besplatno korištenje računalnog oblaka niti na to da su podaci u oblaku javni. Naplata se vrši sukladno korištenim uslugama na infrastrukturi koja se dijeli s drugim korisnicima. Korisnici su sigurnosnim mehanizmima

međusobno izolirani tako da nemaju pravo pristupa podacima koji nisu iz njihove domene. Korisnici nemaju uvid u fizičku lokaciju zakupljenih mrežnih resursa osim lokacije samog podatkovnog centra.

Prednosti javnog oblaka su: jednostavnost korištenja, brza i jeftina početna konfiguracija, brza elastičnost i plaćanje po korištenju. Za korištenje javnog oblaka dovoljan je Internet preglednik, koji dolaze pred instaliran na svako računalo, i poznavanje usluge koju javni oblak pruža. Za početak korištenja uglavnom je dovoljno kreirati korisnički račun i odabrati način plaćanja. Ključne prednosti javnog oblaka omogućava ekonomija razmjera. Javni oblaci mogu biti puno veći od privatnih i po potrebi je moguće zakupiti više ili manje resursa, te se adekvatno tome plaća naknada. Veći broj korisnika javnog oblaka ujedno smanjuje efekte promjene radnog opterećenja za pojedinog korisnika nad cjelokupnim sustavom. Opterećenja u sustavu mogu biti periodička, nepredvidljiva, konstantno promjenjiva i jednokratna.

Neki od nedostataka javnog oblaka su smanjena kontrola i problemi s regulatornim pitanjima. Krajnji korisnik mora se oslanjati na dogovoreni SLA i pružatelja usluge u slučaju prekida ili ometanja rada. Limitirana je i razina kontrole nad konfiguriranjem zakupljenog sustava, te je točno definirano tko je odgovoran za koji dio infrastrukture. Regulatorni problemi su brojni. Najbitnije je kako na adekvatan način napraviti reviziju sustava koje se izvodi na infrastrukturi javnog oblaka, te kako upravljati privatnošću podataka. Neke kompanije rješavaju te probleme u potpunosti u javnom oblaku primjenom SaaS rješenja, dok se drugi odlučuju za hibridni model računarstva u oblaku u kojem senzitivnije dijelove postave u privatni dio, a ostale u javni.

Postoji i pitanje sigurnosti korisničkih podataka pošto se spremaju na dijeljenu infrastrukturu. Općenito se smatra da ako je javni računalni oblak projektiran sa sigurnošću na umu i ako je ispravno konfiguriran od strane korisnika, da može biti i sigurniji od standardnih podatkovnih centara. Za javni oblak postoje sigurnosna rješenja poput mogućnosti nadgledanja sustava zbog nezakonitih radnji i slično, a važno je kontinuirano raditi na sigurnosti jer je to od ključne važnosti za stjecanje povjerenja korisnika i u konačnici uspjeh javnog oblaka.

2.5.2. Privatni model implementacije računarstva u oblaku

Privatni model implementacije računarstva u oblaku odnosi se na infrastrukturu koje je dostupna isključivo jednoj organizaciji. Njome može upravljati sama organizacija ili neka treća stranka. Ovaj model najčešće koriste organizacije koje žele višu razinu nadzora nad podacima nego što to može javni model pružiti. Računalni oblak može se nalaziti na infrastrukturi organizacije koja ju koristi ili se može zakupiti od pružatelja takve usluge.

Privatni model smanjuje slabosti računalnog oblaka poput potpune kontrole, regulatornih problema i konfiguracija. Ako se računalni oblak nalazi na infrastrukturi organizacije, tada je organizacija zadužena za upravljanje fizičkom infrastrukturom . U slučaju da se računalni oblak nalazi na infrastrukturi pružatelja usluge, pružatelj je zadužen za fizičku infrastrukturu, a klijent za sve ostalo. Pošto je klijent u privatnom modelu jedini korisnik računalnog oblaka, to mu omogućava veću sigurnost i kontrolu ali je razmjerno i cijena veća nego što bi bila u računalnom oblaku s više klijenata (npr. javnom). Međutim, korištenje privatnog modela žrtvuju se neke od ključnih prednosti računarstva u oblaku, poput rapidne elastičnosti, udruživanja resursa i plaćanja po korištenju. Privatni oblaci omogućavaju skaliranje korištenja računalnih resursa, ali sve ovisi o količini raspoložive infrastrukture. U javnom oblaku praktički nema limita na količinu tih resursa. Zbog toga je u privatnim oblacima smanjena agilnost jer interni timovi za administraciju moraju upravljati fizičkom infrastrukturom i planirati proširenja na vrijeme. Tako dolazi do viška računalnih kapaciteta i time je uništen model plaćanja po korištenju jer su jer je infrastruktura već plaćena bez obzira dali se koristi ili ne.

2.5.3.Hibridni model implementacije računarstva u oblaku

Hibridni model implementacije računarstva u oblaku je rješenje za probleme javnog i privatnog oblaka. To je kompozicija dva ili više modela implementacije (javnog, privatnog ili zajedničkog) koji ostaju jedinstveni entiteti, ali su međusobno povezani standardiziranjem ili prikladnim tehnologijama koje omogućavaju efikasan prijenos podataka ili aplikacija.

Najbolja praksa za hibridni model je da se javni oblak koristi što je više moguće kako bi se iskoristile karakteristike poput rapidne elastičnosti i udruživanja resursa. Privatni oblak trebao bi se trebao koristiti za riskantnije procese u kojima je važnija sigurnost, vlasništvo i privatnost podataka.

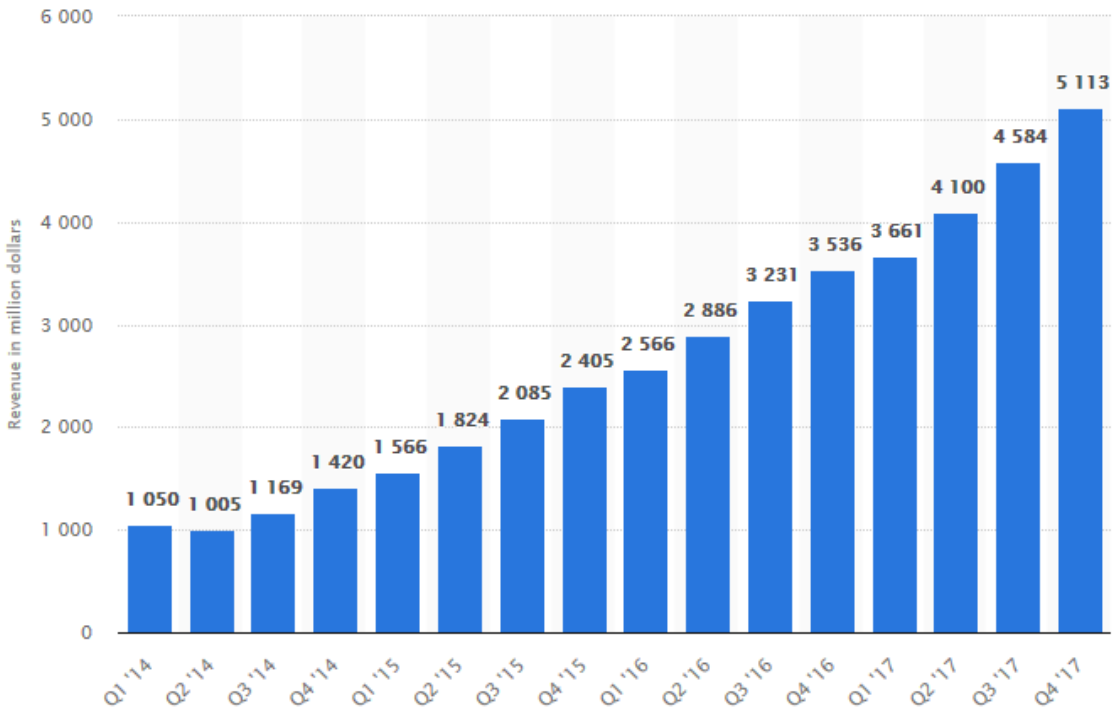
2.5.4.Zajednički model implementacije računarstva u oblaku

U zajedničkom modelu implementacije računarstva u oblaku nekoliko organizacija dijeli strukturu oblaka. Infrastruktura podržava posebne zajednice koje imaju zajedničke potrebe, misije zahtjeve sigurnosti i slično. Njima mogu upravljati samo organizacije ili netko drugi (pružatelj usluga računarstva u oblaku):

3. Amazon AWS

Amazon AWS (eng. *Amazon Web Services* [AWS]) je u tekućoj godini jedan od najpopularnijih pružatelja usluge javnog računarstva u oblaku i pruža mnoštvo usluga iz kategorija infrastrukture kao servisa (IaaS), platforme kao servisa (PaaS) i softvera kao servisa (SaaS). Amazon AWS je podružnica američke organizacije Amazon.com Inc. (u nastavku Amazon) i osnovana je 2006. godine. Osnivač Amazona je Jeff Bezos a glavne djelatnosti organizacije su web prodaja i računarstvo u oblaku. Amazon je osnovan 1994. godine u američkom gradu Seattle i početna djelatnost bila je online prodaja knjiga i studentskih udžbenika. Postepeno su dodavali kategorije artikala sve dok nisu postali jedan od najvećih online prodavača u SAD-u.

Prihod Amazon AWS-a kontinuirano raste od 2014. godine do danas. U četvrtom kvartalu 2017. godine iznos prihoda popeo se na 5,1 milijardi američkih dolara, kao što prikazuje Slika 5 (Statista, 2017). Kontinuirani rast prihoda pripisuje se globalnoj strategiji Amazona kojoj nije cilj profit već tržišna dominacija. Iz tog razloga veliki dio prihoda se ponovno ulaže u rast postojećih i razvoj novih djelatnosti organizacije.



Slika 5: Prihod AWS-a od 1. kvartala 2014. do 4. kvartala 2017. u milijunima USD (Izvor: Statista, 2017)

Prema istraživanju konzultantske tvrtke (Gartner, 2017), Amazon AWS je lider na globalnom tržištu računarstva u oblaku za IaaS. Slika 6 prikazuje vizualizaciju rezultata

istraživanja tržišta nazvanu *Magic Quadrant* koja se oslanja na kvalitativne metode analize podataka za prikazivanje tržišnih trendova, kao što su smjer, zrelost i sudionici.

Slika 6: Magic Quadrant za IaaS (Izvor: Gartner, 2017)

3.1. Karakteristike Amazon AWS-a

Amazon AWS pruža pouzdanu i skalabilnu infrastrukturu za implementaciju web rješenja, s minimalnim troškovima podrške i administracije, te puno većom fleksibilnošću od tradicionalnih podatkovnih centara. Skup karakteristika AWS računarstva u oblaku nasljeđuje prethodno definirane karakteristika računarstva u oblaku (poglavlje 2.3) ali se i proširuju vlastitim karakteristikama. Iz poslovnog aspekta to su slijedeće (Varia, 2011):

- Nije potrebno raditi predikciju IT kapaciteta organizacije - U tradicionalnim podatkovnim centrima prije pokretanja projekata bilo je potrebno napraviti predikciju potrebnih IT kapaciteta. To je kompleksan zadatak jer je teško

pretpostaviti potencijalna opterećenja na sustav. U slučaju da kapaciteti budu premali, sustav može prestati normalno funkcionirati. Ako pak budu preveliki, iskoristivost tih kapaciteta možda neće biti visoka, što smanjuje efikasnost sustava i generira trošak.

- Efikasno iskorištavanje IT kapaciteta - Primjenom računarstva u oblaku dinamički je moguće skalirati IT kapacitete i time koristiti samo onoliko resursa koliko je u danom trenutku potrebno.
- Infrastruktura na vrijeme (eng. *Just-in-time*) - U tradicionalnim podatkovnim centrima prilikom pokretanja projekata potrebno je procijeniti kapacitete, nabaviti IT infrastrukturu, čekati da bude isporučena, instalirati i konfigurirati sustav. Potrebno je i platiti zaposlenike koji će izvršavati navedene aktivnosti, s time da može doći do grešaka u sustavu uzrokovanih ljudskim faktorom. Primjenom računarstva u oblaku infrastrukturu je moguće zakupiti i koristiti u periodu od svega par minuta, a svaka potencijalna greška nad fizičkom infrastrukturom pokrivena je SLA ugovorom.
- Minimizacija potrebe za ulaganjem u naprijed za infrastrukturu - Primjena računarstva u oblaku omogućava usvajanje OPEX modela troškova (operativnih troškova) naprema CAPEX (ulaganje u osnovna sredstva).
- Plaćanje po korištenju - Korištenje računarstva u oblaku naplaćuju se samo računalni resursi koji su potrošeni. Za svaki AWS servis postoji adekvatna mjerna jedinica naplate npr.: kod EC2 virtualnih servera to je vrijeme korištenja, kod ELB servisa za ujednačavanje opterećenja to je vrijeme korištenja i količina prometa, kod servisa za pohranu i rad s podacima to je količina podataka itd.
- Minimizacija kompleksnosti naplate usluga - Prethodno spomenuti princip naplate usluga pojednostavljuje proces naplate usluge klijentima. Dodatno, primjenom AWS *Trusted Advisor* servisa, sustav se može u stvarnom vremenu izvođenja nadgledati i optimizirati s obzirom na performanse i potrošnju računalnih resursa, prateći najbolje prakse. Spomenuti servis je besplatan za korištenje i dodatno pomaže kod optimizacije sigurnosnih postavki i raspoloživosti sustava.
- Koristi od ekonomije razmjera - Kako je AWS pružatelj usluge javnog računarstva u oblaku s velikim brojem korisnika koji svakodnevno raste, tako svi korisnici imaju koristi od ekonomije razmjera. To znači da što više AWS ima korisnika, to su im usluge jeftinije.
- Povećanje brzine i agilnosti za organizacije - Po potrebi se lako zakupljuju računalni resursi i otpuštaju kada više nisu potrebni. Recimo da postoji

zahtjevan računalni proces koji se treba izvršavati 500 sati na postojećoj infrastrukturi. Ako je moguće taj proces paralelizirati, na AWS-u bi mogli zakupiti 500 instanci virtualnih servera i izvršiti navedeni proces za 1 sat, otpustiti zakupljenu infrastrukturu, platiti samo za korištene resurse i to sve bez ikakvih dugoročnih ugovornih obveza.

- Postiže se globalni doseg u minutama - Korištenjem AWS infrastrukture u različitim regijama i raspoloživim zonama postiže se globalni doseg informacijskog sustava u minutama. Prilikom puštanja sustava u pogon samo se odabere raspoloživa zona pa je korisniku taj proces jednak bez obzira dali je sustav u njegovoj regiji ili na drugom kraju svijeta. U tradicionalnom IT poslovanju za takav pothvat bilo je potrebno uložiti goleme količine novca da bi se izgradio novi fizički podatkovni centar, nabavila i konfigurirala infrastruktura itd. AWS je taj postupak maksimalno pojednostavio i pojeftinio.
- Infrastruktura prestaje biti fokus poslovanja - Računarstvo u oblaku omogućava korisnicima da prebace fokus poslovanja s infrastrukture na ključne kompetencije i vlastite poslovne procese.

Značajnije karakteristike AWS računarstva u oblaku iz tehničkog aspekta su slijedeće (Varia, 2011):

- Automatizacija - AWS preko svojih aplikacijskih programskih sučelja (API-a) pruža infrastrukturu kao programski kod (eng. *Infrastructure as code*). To znači da postojećim AWS servisima i vlastitim programskim kodom možemo upravljati infrastrukturom, te upravljati i automatizirati bilo koji aspekt infrastrukture.
- Dinamičko skaliranje - AWS omogućava dinamičko skaliranje prema opterećenju ili nekoj proizvoljnoj varijabli potpuno automatizirano, bez potrebe za ljudskom intervencijom. Time se postiže viša raspoloživost i efikasnost sustava. Konkretni uzorci dizajna opisani su u poglavljima: 4.1.3 Dinamičko skaliranje specifikacija servera, 4.1.4 Dinamičko skaliranje broja servera i 4.1.5 Dinamičko skaliranje kapaciteta diska. Također je moguće proaktivno skalirati sustav nakon što je jednom razvijen i pušten u pogon. Po potrebi se mogu dodati ili ukloniti računalni resursi uz minimalne napore.
- Efikasniji razvojni i testni ciklusi - Kompletna infrastruktura i razvojna okolina produkcijskih sustava lako se može klonirati i pokrenuti s ciljem daljnjeg razvoja i testiranja. Testne sustave lako je promovirati u produkcijske sustave. Također, primjenom AWS oblaka, nikad se neće dogoditi da ponestane infrastrukture za testiranje. Moguće je razviti automatizirano testiranje na svim fazama razvojnog

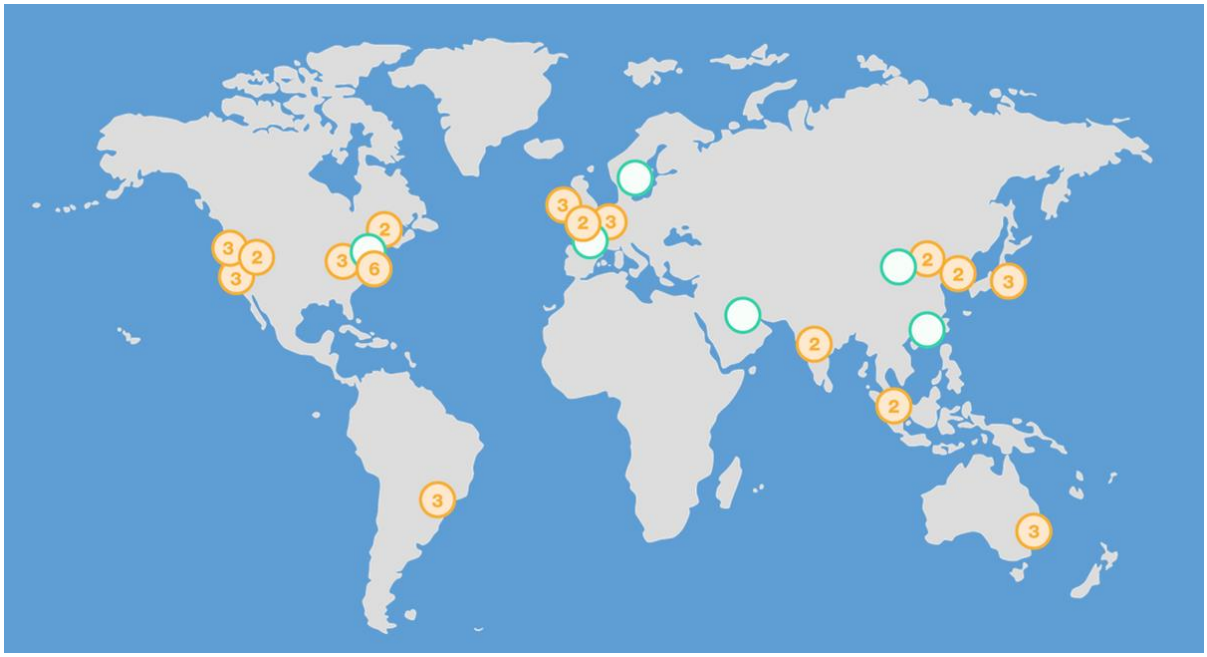
procesa. Pri kraju životnog vijeka testnih sustava, lako se otpuštaju i ne stvaraju trošak ukoliko se ne koriste.

- Oporavak od katastrofe i kontinuitet poslovanja - AWS oblak je povoljna opcija za održavanje računalne infrastrukture s namjenom oporavka od katastrofe. Mogu se iskoristiti različite geografske lokacije za distribuciju i replikaciju podataka i infrastrukture čime se postiže visoka raspoloživost sustava. Primjenom uzoraka dizajna kod opterećenja sustava promet je moguće automatski preusmjeriti na sekundarnu infrastrukturu, bez ikakvog prekida u kontinuitetu izvođenja.
- Sigurnost - AWS razvija i isporučuje svoje usluge u skladu s najboljim i najstrožim praksama sigurnosti. Posjeduju brojne certifikate i provode redovite i temeljite revizije kako bi dokazali sigurnost svoje infrastrukture.

3.2. Globalna infrastruktura Amazon AWS-a

Infrastruktura Amazon AWS-a sastoji se od regija (eng. *Region*) i raspoloživih zona (eng. *Availability Zones* [AZ]). Regija je geografska lokacija koja se sastoji od dvije ili više AZ. Svaka Amazon regija dizajnirana je da tako da bude u potpunosti izolirana od ostalih Amazon regija. Time se postiže stabilnost i otpornost na kvarove na razini regije. Svaka AZ je klaster dva ili više podatkovnih centara pojedinačno izolirani od drugih, ali unutar regije su spojene konekcijom s velikom propusnošću i niskom latencijom. Svaki podatkovni centar ima redundantno napajanje i mrežnu konekciju, te zasebnu fizičku lokaciju. Po mogućnosti, smješteni su na različitim geografskim područjima s obzirom na rizik od elementarnih nepogoda, poput potresa, poplava i slično. Tim mjerama su izbjegnute potencijalne jedinstvene točke kvara (eng. *Single point of failure*).

Amazon kontinuirano gradi nove podatkovne centre. U trenutku pisanja AWS oblak sastoji se od 44 AZ unutar 16 geografskih regija, a u planu su još 17 AZ i 6 novih regija (Bahrain, Kina, Francuska, Hong Kong, Švedska, US-East). Slika 7 prikazuje postojeće regije obojenim kružnicama s brojem izoliranih regija na geografskoj lokaciji, a nove regije u izgradnji prikazane su bijelim kružnicama (Amazon, 2017).



Slika 7: Globalna mreža Amazon AWS regija (Izvor: Amazon, 2017)

Kod odabira Amazon regije za puštanje u pogon proizvoljnog sustava valja uzeti u obzir sljedeće faktore:

- Latencija mrežne konekcije - Preporuka je odabrati onu regiju u kojoj geografski planiramo najviše djelovati. Time se smanjuje udaljenost kojom se komunikacija preko javnog Interneta izvršava, te se smanjuju kašnjenja.
- Usklađenost s propisima - Organizacija koja planira koristiti usluge javnog oblaka mora biti upoznata s regulativama zemalja na kojima planira djelovati. Tijekom pripreme projekta mora se uzeti u obzir osjetljivost podataka s kojima se planira poslovati te uskladiti s postojećim zakonima. Na primjer, zemlja može zabraniti bankama da privatne podatke o građanima i njihove transakcije pohranjuju izvan teritorijalnih granica te zemlje. U slučaju da u toj zemlji ne postoji Amazon podatkovni centar tj. AZ, navedena banka bi kršila zakon koristeći usluge javnog oblaka.
- Trošak - Potrebno je izraditi procjenu troškova, npr. primjenom Amazon *Trusted Advisor*-a.
- Dostupnost usluge - Treba se upoznati s Amazon SLA ugovorom i procijeniti dali njihova ponuda zadovoljava poslovne potrebe.

Slika 8: Globalna mreža AWS rubnih područja (Izvor: Amazon, 2017)

Amazon AWS rubna područja (eng. *Edge Locations*) su lokalne točke prisutnosti (Slika 8) koje podupiru AWS usluge poput:

- Amazon *Route 53* - Servis za domenski sustava imena (eng. *Domain Name System* [DNS]).
- Amazon *CloudFront* - Servis za mrežno dostavljanje podataka (eng. *Content Distribution Network* [CDN])

AWS rubna područja pružaju robusnu mrežu za isporuku sadržaja koja se može koristiti za isporuku dinamičkog, statičkog i *streaming* sadržaja, kao i kompletnih web stranica. Zahtjevi za sadržaj automatski se preusmjeravaju na najbliža rubna područja, pa se sadržaj isporučuje uz najbolje moguće performanse. U trenutku pisanja Amazon je premašio brojku od stotinu kreiranih rubnih područja (Amazon, 2017).

3.3. Sigurnost AWS-a i usklađenost sa sigurnosnim zahtjevima

Prema (Amazon, 2017) sigurnost je najveći prioritet AWS-a. Svaki podatkovni centar i kompletna mrežna infrastruktura kreirani su da zadovolje i najstrože sigurnosne zahtjeve organizacija. Sigurnost u oblaku je ekvivalentna razini sigurnosti u tradicionalnom podatkovnom centru samo bez troškova održavanja i upravljanja hardverom. AWS omogućava korisnicima mnoštvo alata za zadovoljavanje sigurnosnih zahtijeva koji su u rasponu od mrežne sigurnosti, upravljanja konfiguriranjem i pravima pristupa, sve do enkripcije podataka.

Softverskim sigurnosnim alatima nadzire se i štiti sve informacije koje ulaze u i izlaze iz sustava na oblaku. Uz takav veliki raspon alata omogućeno je i mnoštvo besplatne online dokumentacije, edukacije i podrške za usluge.

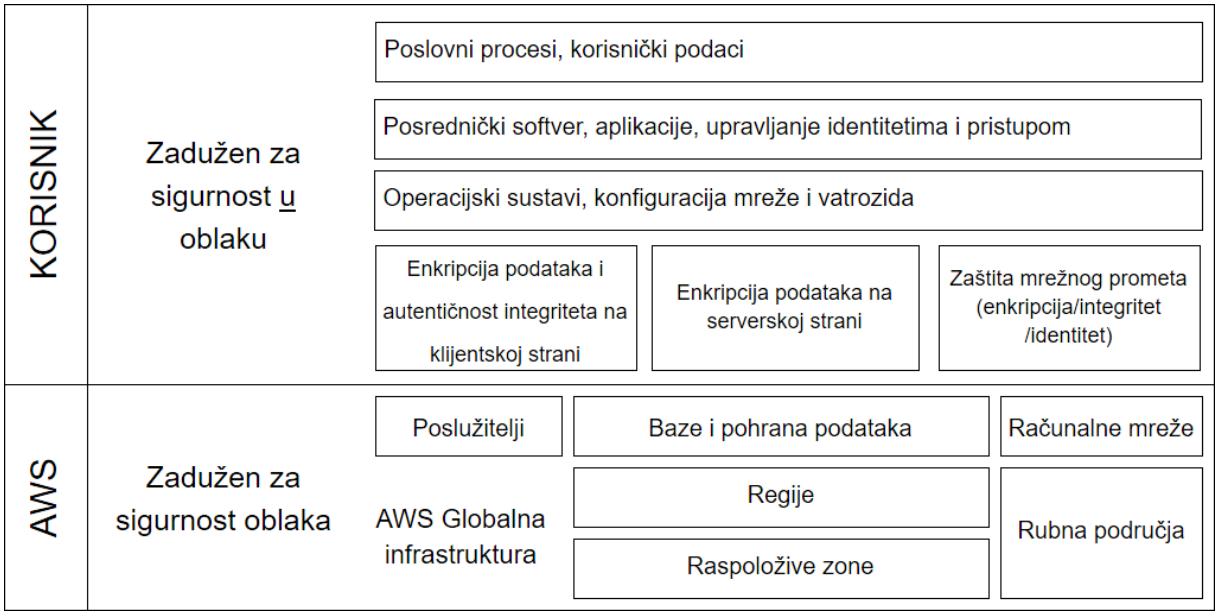
AWS razvija i isporučuje svoje usluge u skladu s najboljim i najstrožim praksama za sigurnosne politike, arhitekturu i operativne procese. AWS posjeduje brojne certifikate i izvršava programe za usklađenost (eng. *Compliance*) sa sigurnosnim zahtjevima (SOC 1/ISAE 3402, SOC 2, SOC 3, FISMA, DIACAP, FedRAMP, PCI DSS Level 1, ISO 9001, ISO 27001, ISO 27018 i dr.), te provode redovite i temeljite revizije kako bi dokazali sigurnost svoje infrastrukture.

Glavne pogodnosti korištenja AWS-a iz aspekta sigurnosti (Amazon, 2017):

- Sigurnost podataka - AWS infrastruktura implementira stroge procedure za sigurnost i privatnost podataka.
- Usklađenost sa sigurnosnim zahtjevima - AWS kontinuirano upravlja s desecima bitnih programa za usklađenost sa sigurnosnim zahtjevima što znači da su korisnički zahtjevi za usklađenost vrlo vjerojatno već pokriveni.
- Smanjenje troškova - Postiže se plaćanjem po korištenju i bez potrebe za održavanjem i upravljanjem sigurnosti fizičkog hardvera.
- Rapidno skaliranje - Omogućava fleksibilnost, agilnost i potiče inovacije uz zadržavanje sigurnosti razvojne okoline. Sigurnosne kontrole se bez problema skaliraju zajedno s ostatkom sustava.

3.3.1. Model zajedničke odgovornosti

AWS primjenjuje model zajedničke odgovornosti (eng. *Shared responsibility model*). Dok je AWS zadužen za sigurnost samog oblaka, korisnik je zadužen za sigurnost u oblaku. To znači da AWS zadužen za fizički i virtualni hardver, a korisnik za sigurnost operacijskog sustava, posredničkog softvera, aplikacija, poslovnih procesa i korisničkih podataka. Dakle kao u tradicionalnom podatkovnom sustavu ali bez fizičkog i virtualiziranog hardvera.



Slika 9: AWS Model zajedničke odgovornosti (Prema: Amazon, 2017)

3.4. Infrastruktura Amazon AWS-a

Ključni servisi, na kojima se zasniva kompletna ponuda Amazon AWS-a, spadaju u IaaS model računarstva u oblaku i odnose se na sljedeće kategorije: poslužitelji, baze podataka, pohrana podataka i računalne mreže. U nastavku je dan kratak opis odabranih fundamentalnih servisa s dodatkom servisa za sigurnost i za upravljanje resursima AWS. Postoji jako veliki broj AWS servisa, ali u nastavku su osnovni koji se najčešće koriste i koji se spominju u poglavljima 4. Uzorci dizajna za AWS i 5. Primjena AWS-a.

3.4.1. Poslužitelji

3.4.1.1. Amazon EC2

Amazon *Elastic Compute Cloud* (EC2) je web servis koji pruža virtualizirane poslužitelje koji su sigurni i fleksibilni. EC2 servis osmišljen je kako bi razvojnim inženjerima omogućio lakše korištenje fleksibilnih poslužitelja uz što kraće vrijeme čekanja. EC2 instance su osnovni gradivni blok cjelokupnog AWS sustava. Nad korištenim EC2 instancama korisnik ima potpunu kontrolu i jednostavno je skalirati sustav u ovisnosti o potrebama. Princip plaćanja po korištenju omogućava uštede, a pristup globalnom tržištu nikad nije bio lakši. Razlika u puštanju poslužitelja u pogon na lokalnom tržištu, na prema puštanje u pogon na drugom kraju svijeta, razlikuje se u jednom kliku miša. Instance podržavaju Linux i Windows platforme, te su integrirane s većinom usluga koje AWS nudi. Pouzdanost usluge EC2 servisa garantirana je Ugovorom o razini usluge (eng. *Service Level Agreement* [SLA]) u kojoj stoji da će Amazon AWS osigurati dostupnost usluge od 99.95% za svaku postojeću regiju. (Amazon, 2017)

Sigurnost EC2 instanci usko je vezana uz Amazon VPC (eng. *Virtual Private Cloud* [VPC]) servis za logičku izolaciju poslužitelja u privatnu mrežu:

- Kada su EC2 instance u VPC-u mogu imati proizvoljne IP adrese kojima je moguće dati pravo ili zabraniti pristup Internetu, ovisno o potrebi.
- Sigurnosne grupe i kontrolne liste za pristup mreži (eng. *Network Access Control List* [NACL]) omogućavaju regulaciju ulaznog i izlaznog prometa EC2 instanci.
- EC2 instancama u VPC-u pristupa se sigurnim i kriptiranim alatima koji se u industriji koriste za pristup virtualnim privatnim mrežama (eng. *Virtual Private Network* [VPN]).
- Moguće je zakupiti dedicirane instance i dediciranu infrastrukturu na Amazon AWS-u. Takva infrastruktura je logički i fizički odvojena od ostale dijeljene infrastrukture oblaka i pomaže prilikom usklađivanja sustava sa sigurnosnim zahtjevima.

Postoji nekoliko financijskih oblika EC2 instanci, svaka s drugačijom namjenom i cijenom:

- Instance na zahtjev (eng. *On-Demand Instances*) - Za instance ovog tipa plaća se po satu korištenju bez dugoročnih obveza. Lako povećati ili smanjiti broj ili jačinu instanci prema potrebi što uklanja potrebu za procjenom kapaciteta informacijskog sustava. Ovaj tip je najbolji za radna opterećenja koja jako osciliraju i teško ih je predvidjeti.
- Rezervirane instance (eng. *Reserved Instances*) - Ovaj tip instanci omogućava potencijalne uštede i do 75% u odnosu na instance na zahtjev. Instance se rezerviraju i cijena se kalkulira u odnosu na tri ključna faktora: snagu zakupljene računalne infrastrukture, vremenski period na koji se zakupljuje i metoda plaćanja. Ovaj tip je pogodan za predvidljiva i kontinuirana radna opterećenja sustava bez velikih oscilacija.
- Primijećene instance (eng. *Spot Instances*) - Ova AWS ponuda funkcionira na način da viškove računalnih kapaciteta po pojedinim regijama nude po sniženoj cijeni kako popunili kapacitete na fizičkim lokacijama koje njima odgovaraju. Ovaj tip instanci prikladan je za informacijske sustave kojima vremenska komponenta nije bitna. Na primjer, na sustavu se kontinuirano izvodi softver za analitiku velike količine podataka. Kada bi takav sustav imao više računalnih resursa, obradu podataka bi mogao ubrzati i prije završiti. Odabirom primijećene instance kapaciteti sustava mogu se povećati po sniženoj cijeni za vrijeme trajanja te ponude.

3.4.1.2. Amazon AMI

Amazon AMI (eng. *Amazon Machine Image* [AMI]) je predložak za EC2 instance. Predložak sadrži kompletnu konfiguraciju operacijskog sustava, aplikacija, dozvole za pokretanje, virtualne diskove za pohranu i svega što je potrebno da bi željeni sustav funkcionirao. Iz predloška je moguće pokrenuti proizvoljan broj EC2 instanci. AMI predloške je moguće samostalno kreirati ili nabaviti na Amazon AWS online tržištu (eng. *Marketplace*). (Amazon, 2017)

3.4.1.3. Proces pokretanja Amazon EC2 instance

Proces pokretanja Amazon EC2 instance korištenjem mrežnog sučelja je jednostavan. Sastoji se od četiri koraka i potrebno je:

1. odabrati AWS regiju u kojoj će EC2 instanca biti pokrenuta,
2. odabrati AMI predložak koji će biti korišten za pokretanje EC2 instance,
3. odabrati tip EC2 instance u odnosu na broj procesorskih jezgri, količinu radne memorije, količinu i tip podatkovnog sustava i mrežne performanse,
4. konfigurirati računalnu mrežu, postaviti IP adrese, sigurnosne grupe, podatkovni sustav, oznake (eng. *Tags*) i sigurnosne ključeve za pristup.

Po završetku navedenih koraka EC2 instance biti će spremna za korištenje kroz par minuta, ovisno o kompleksnosti infrastrukture koja se pokreće. Uz mrežno sučelje (UI), preostali načini za pokretanje i komuniciranje s AWS servisima su primjenom aplikacijskog programskog sučelja (API) i setom razvojnih alata (SDK).

3.4.1.4. Amazon Auto Scaling

Amazon Auto Scaling podiže raspoloživost sustava na AWS-u na način da automatski skalira EC2 instance prema potrebi. U ovom servisu definira se broj EC2 instanci koje trebaju biti aktivne u odnosu na opterećenje. Broj instanci može se povećavati ili smanjivati. U slučaju da opterećenje nad sustavom naglo poraste, Auto Scaling aktivira pokretanje novih EC2 instanci preko kojih ELB može ujednačiti opterećenje. U tradicionalnim sustavima ovakav slučaj doveo bi do ne mogućnosti pružanja usluge svim korisnicima. Kada se opterećenje smanji, Auto Scaling započne smanjivati broj EC2 instanci kako bi omogućio uštede. Auto Scaling također može pratiti ispravnost izvođenja EC2 instanci. Ukoliko dođe do kritične greške, nova EC2 instanca može biti pokrenuta da zamijeni neispravnu i omogući nesmetano izvođenje sustava. (Amazon, 2017)

3.4.2. Pohrana podataka

3.4.2.1. Amazon S3

Amazon S3 (eng. *Simple Storage Service* [S3]) je web servis za "neograničenu" pohranu objekata. Jednostavno ga je koristiti, niska mu je cijena i integriran je s velikim brojem AWS servisa. S3 je visoko raspoloživ (99.99%), trajan (99.999999999%) i skalabilan servis s preko milijardu pohranjenih objekata. Ne postoji limit na broj objekata koji je moguće pohraniti, ali postoji limit od 5 TB na pojedini objekt. S3 omogućava enkripciju podataka sa serverske i klijentske strane, logove za revizije, sustav verzioniranja i HTTP/S krajnje točke (eng. *Endpoint*) za pohranu i dohvaćanje podataka. (Amazon, 2017)

Najčešći slučajevi korištenja S3 servisa su:

- pohrana podataka i sigurnosnih kopija,
- usluge poslužitelja za medijski sadržaj, kao i statični i dinamični sadržaj za web stranice,
- dostava softvera,
- pohrana AMI-a, sigurnosnih kopija, slika stanja, itd.

3.4.2.2. Amazon EBS

Amazon EBS (eng. *Elastic Block Storage* [EBS]) je servis za perzistentan virtualni podatkovni disk koji nudi konzistentne performanse s niskim latencijama. Svaki EBS disk se automatski replicira unutar njegove raspoložive zone (AZ) čime se podiže raspoloživost (99.999%) i trajnost podataka. EBS omogućava enkripciju podataka, postavljanje prava pristupa i kreiranje slika stanja (eng. *Snapshot*) čime pospješuje kontinuitet poslovanja (eng. *Business Continuity*). Moguće je odabrati fizičku komponentu iza EBS diska; tvrdi disk (eng. *Hard Disk Drive* [HDD]) ili brži i skuplji SSD disk (eng. *Solid-state Drive* [SSD]). (Amazon, 2017)

Amazon EBS se preporuča koristiti kada se podaci često mijenjaju a zahtijeva se dugoročna perzistencija podataka. Partikularno je dobar za pohranu podatkovnih sustava (eng. *File Systems*), baza podataka i za bilo koju aplikaciju koja zahtijeva granulirana ažuriranja, te pristup "sirovim" (eng. *Raw*) ne formatiranim *block-level* podacima.

3.4.2.3. Usporedba Amazon EBS i S3 servisa

Tabela 2 prikazuje kratku usporedbu Amazon EBS i S3 servisa za pohranu podataka. Valja napomenuti da su za posljednji red u tablici dane zadane vrijednosti servisa koje je moguće modificirati. Ako je EBS disk spojen na EC2 instancu koja ima pristup Internetu i recimo da je njegova namjena posluživanje podatkovnog sadržaja, tada će EBS disk biti dostupan na Internetu. Isto tako je moguće modificirati S3 servis da sadržaj ne bude dostupan s Interneta te da služi kao privatni repozitorij.

Tabela 2: Usporedba Amazon AWS EBS i S3 servisa (Prema: Amazon, 2017)

	Amazon EBS	*Amazon S3*
Performanse	Jako brze	Brze
Redundancija	Preko više poslužitelja unutar raspoložive zone (AZ)	Preko više podatkovnih centara unutar AWS regije
Sigurnost	Enkripcija EBS diska i slike stanja	Enkripcija podataka
Tipični slučaj korištenja	Podatkovni disk	Online pohrana podataka
Omogućen pristup s Interneta?	Ne	Da

3.4.3.Baze podataka

3.4.3.1. Amazon RDS

Amazon RDS (eng. *Relational Database Service*) je servis za jednostavno korištenje i upravljanje relacijskom bazom podataka u oblaku. RDS servis za povoljnu cijenu pruža sigurnost, skalabilnost, visoku raspoloživost i trajnost baze podataka. Kada se koristi konfiguracija RDS-a s više od jedne raspoložive zone (eng. *Multi-AZ*) tada servis sinkronizirano replicira podatke u sekundarnu raspoloživu zonu. Neke od dodatnih mogućnosti za podizanje raspoloživosti su automatsko kreiranje sigurnosnih kopija i slika stanja baza podataka, kao i automatska zamjena neispravnih poslužitelja. Amazon RDS podržava šest popularnih standarda baza podataka: PostgreSQL, MySQL, MariaDB, Oracle, Microsoft SQL Server i Amazon Aurora. (Amazon, 2017)

3.4.3.2. Amazon Aurora

Amazon Aurora je novi i napredni servis za relacijske baze podataka koji podržava baze otvorenog koda MySQL i PostgreSQL. Pruža visoku brzinu i raspoloživost skupih komercijalnih baza podataka s jednostavnošću korištenja i niskom cijenom baza podataka otvorenog koda. Amazon Aurora nudi do pet puta bolje performanse od MySQL baze podataka sa sigurnošću i visokom raspoloživošću (+99.99%) ekvivalentnoj komercijalnim rješenjima za jednu desetinu cijene. (Amazon, 2017)

Oporavljanje od fizičkog kvara baze podataka je automatsko i transparentno, u prosiječnom trajanju od 30 sekundi. Po zadanim vrijednostima, šest kopija Aurora baze podataka se replicira na tri raspoložive zone uz kontinuirano kreiranje sigurnosnih kopija na Amazon S3. Amazon Aurora nudi *fully managed* baze podataka što znači da administratori

baza podataka više ne trebaju brinuti o zadacima poput: proširivanja kapaciteta, ažuriranja softvera, instaliranja, konfiguriranja, nadgledanja performansi ili kreiranja sigurnosnih kopija.

3.4.3.3. Amazon DynamoDB

Amazon DynamoDB je servis koji pruža brze i fleksibilne *NoSQL* baze podataka uz latencije do maksimalno 10ms bez obzira na veličinu sustava. Podržava dva *NoSQL* modela podatkovnih struktura: *document* i *key-value.* DynamoDB je *fully managed* servis i preporuča se kao najbolji odabir aplikacija za mobilne uređaje, web, igre, Internet stvari (eng. *Internet of Things* [IoT]), i sl. (Amazon, 2017)

DynamoDB je integriran s AWS Lambda servisom koji omogućava kreiranje tzv. okidača koji se aktiviraju kad je neki uvjet zadovoljen. Tako je moguće kreirati arhitekturu sustava koja će adekvatno automatski reagirati na promjene. Na primjer, kreiramo web stranicu s aplikacijom nagradne igre koja sprema podatke u DynamoDB. Postavimo okidač na AWS Lambda servisu da nakon isteka nagradne igre pobjednik automatski dobije poveznicu na e-mail adresu sa nagradu u digitalnom obliku čiji sadržaj će preuzeti sa S3 servisa.

3.4.3.4. Usporedba Amazon RDS i DynamoDB servisa

Tabela 3 prikazuje kratku usporedbu Amazon RDS i DynamoDB servisa za baze podataka. Dok je namjena RDS relacijske baza podataka s tabličnom strukturom orijentirana na tradicionalne poslovne aplikacije, DynamoDB je *NoSQL* baza podataka koja daje bolju podršku novim web tehnologijama.

Tabela 3: Usporedba Amazon RDS i DynamoDB servisa (Prema: Amazon, 2017)

	Amazon RDS	***Amazon DynamoDB***
Tip baze podataka	Relacijska	*NoSQL*
Namjena za tip aplikacija	Aplikacije koje koriste postojeće baze podataka Poslovno orijentirane aplikacije	Nove skalabilne web aplikacije Velik broj kratkih procesa čitanja iz i pisanja u bazu
Karakteristike aplikacija	Relacijski podatkovni modeli Kompleksni upiti za bazu	Jednostavni podaktivni model Jednostavni upiti, transakcije, ažuriranja
Skaliranje	Omogućava aplikacija ili arhitektura baza podataka (klasteri, particije, sharding)	Jednostavno skaliranje po zahtjevima aplikacije

Kvaliteta usluge	Performanse - ovise o podatkovnom modelu, tipu indeksiranja, kompleksnosti upita, optimizaciji podatkovnog sustava Pouzdanost i raspoloživost Trajnost	Performanse - sustav automatski optimizira Pouzdanost i raspoloživost Trajnost

3.4.4. Računalne mreže

3.4.4.1. Amazon VPC

Amazon VPC (eng. *Virtual Private Cloud* [VPC]) je servis za logičku izolaciju poslužitelja i ostalih mrežnih resursa na infrastrukturi javnog oblaka u privatnu računalnu mrežu nad kojom korisnik ima potpunu kontrolu. Neki od slučajeva korištenja AWS VPC-a su: web poslužitelji, web aplikacije koje mogu biti i skalabilne, proširivanje postojeće mrežne infrastrukture koristeći Amazon AWS, oporavak od katastrofalnih grešaka sustava itd. (Amazon, 2017)

Neke od mogućnosti VPC-a su:

- Sigurnosne grupe (eng. *Security Groups*) i kontrolne liste za pristup mreži (eng. *Network Access Control List* [NACL]),
- Kontrola nad IP adresiranjem primjenom CIDR (engl. *Classless Inter-Domain Routing*) standarda za očuvanje adresnog prostora Interneta,
- Mogućnost kreiranja podmreža (eng. *Subnet*); vrlo korisno ako recimo želimo kreirati arhitekturu sustava koja će imati odvojen javno dostupni dio i siguran privatni dio,
- Mogućnost dodjeljivanja višestrukog broja IP adresa i elastičnih mrežnih sučelja,
- Mogućnost pokretanja ELB servisa za ujednačavanje opterećenja,
- Mogućnost povezivanja VPC-a s fizičkom infrastrukturom korisnika.

Postoji nekoliko opcija za povezivanje VPC-a s fizičkom infrastrukturom:

- AWS Hardware VPN - Hardverska VPN konekcija između VPC-a i korisničke mreže,
- AWS Direct Connect - Dedicirana privatna konekcija VPC-a i korisničke mreže,
- AWS VPN CloudHub - Kreira se veći broj VPN konekcija za komunikaciju preko više kanala za različite razine mreže,

- Software VPN - Softversko VPN rješenje koje se pokreće s EC2 instance unutar VPC-a.

3.4.4.2. Amazon ELB

Amazon ELB (eng. *Elastic Load Balancing* [ELB]) je servis za automatsko ujednačavanje opterećenja preko većeg broja EC2 instanci. ELB servis omogućava povećavanje otpornosti sustava na kvarove, automatsko skaliranje i pospješuje sigurnost sustava. Može se koristiti u jednoj ili više raspoloživih zona (AZ). (Amazon, 2017)

Postoje tri tipa ELB-a:

- Klasični ELB - ujednačavanja opterećenja preko više EC2 instanci u AWS oblaku.
- Aplikacijski ELB - najbolja opcija za ujednačavanje opterećenja HTTP i HTTPS prometa. Podržava moderne aplikacijske arhitekture poput mikro servisa i kontejnerske virtualizacije.
- Mrežni ELB - namijenjen za velika opterećenja TCP prometom. Podržava milijune zahtjeva po sekundi s vrlo niskim latencijama. Optimiziran za naglo promjenjive uzorke prometa.

3.4.5. Sigurnost

3.4.5.1. Amazon IAM

Amazon IAM (eng. *Identity and Access Management* [IAM]) je servis za sigurno upravljanje identitetima i pristupom AWS računalnim resursima. Definirane politike spremljene su kao dokument u JSON formatu i one opisuju prava pristupa za korisnike, grupe i uloge na Amazon AWS-u. (Amazon, 2017)

IAM omogućava sljedeće:

- Upravljanje IAM korisnicima, grupama i njihovom pravu pristupa - Kreiranje korisnika, dodjeljivanje konzistentnih ili privremenih prava pristupa i adekvatnih sigurnosnih mjera (pristupnih ključeva, lozinki, više-faktorske autentifikacije)
- Upravljanje IAM ulogama i njihovom pravu pristupa - Korisnike IAM-a moguće je dodati u kreirane uloge kojima su definirana prava pristupa po ulogama u sustavu. Jedan korisnik može biti član više IAM rola.
- Upravljanje "ujedinjenim" korisnicima i njihovom pravu pristupa - Moguće je postojeće definicije korisnika i njihovih prava iz organizacije korisnika iskoristiti za pristup AWS mrežnim resursima.

Postoji dokument s najboljim praksama za konfiguriranje IAM servisa. Generalno je preporuka držati se dodjeljivanja minimalnog potrebnog prava pristupa. Slijede neke od preporučenih najboljih praksi za IAM:

- Ukloni AWS korijenske (eng. *root*) pristupne ključeve,
- Kreiraj individualne IAM korisnike,
- Koristi grupe za dodjeljivanje prava pristupa IAM korisnicima,
- Definiraj kvalitetnu politiku zaporki,
- Omogući više-faktorsku autentifikaciju,
- Koristi IAM uloge za aplikacije koje se izvode na EC2 instancama,
- Delegaciju izvršavaj primjenom IAM uloga umjesto dijeljenjem zaporki,
- Ukloni IAM korisnike, grupe i uloge koje se ne koriste,
- Nadgledaj aktivnosti AWS korisničkih računa primjenom AWS CloudTrail-a, servisa za sigurnosnu reviziju AWS-a.

3.4.6. Alati za upravljanje

3.4.6.1. Amazon Trusted Advisor

Amazon Trusted Advisor je servis koji analizira postavljenu AWS infrastrukturu i predlaže poboljšanja prema najboljoj praksi. Servis izvršava provjere u četiri kategorije: optimizacija troškova, sigurnost, otpornost na kvarove i poboljšanje performansi. Rezultati provjera prikazane su na preglednoj kontrolnoj ploči u tri boje. Crvena boja predstavlja pronađeni problem, žuta boja zahtjeva ručnu provjeru rezultata i zelena da je sustav ispravan. Za svaki rezultat provjere moguće je vidjeti detaljan opis s preporučenom najboljom praksom za rješavanje problema, setom kriterija po kojima je alarm podignut, smjernice za pravilno postupanje i popis resursa s detaljnim informacijama. (Amazon, 2017)

Pod optimizaciju troškova spada:

- Optimizacija rezerviranih EC2 instanci,
- Niska iskorištenost EC2 instanci,
- Niska iskorištenost EBS instanci,
- Niska iskorištenost RDS instanci,
- ELB koji nisu aktivni,
- Neiskorištenost IP adresa i sl.

Pod optimizaciju sigurnosti spadaju:

- Sigurnosne grupe,
- AWS IAM,

- Konfiguracija S3 servisa,
- Više-faktorska autentifikacija na *root* korisničkim računima,
- RDS sigurnosne grupe i sl.

Pod otpornost na kvarove spada:

- Redovito kreiranje sigurnosnih kopija i slika stanja,
- Optimizacija rada ELB-a,
- Optimizacija Auto Scaling servisa,
- Preporuke oko korištenja redundantnih raspoloživih zona,
- Delegacije Route 53 servisa za domenski sustava imena i sl.

Pod poboljšavanje performansi spada:

- Visoka iskorištenost EC2 instanci,
- Konfigurirani limiti servisa,
- Kompleksnost pravila u EC2 sigurnosnim grupama,
- Visoka iskorištenost EBS instanci,
- Optimizacija propusnosti između EC2 i EBS instanci,
- Preporuke za korištenje AWS CloudFront servisa za mrežno dostavljanje podataka, i sl.

3.4.6.2. Amazon CloudWatch

CloudWatch je servis za nadgledanje drugih AWS servisa i aplikacija koje se izvode u AWS-u. To je centralizirana lokacija gdje se skupljaju i prate metrike, log datoteke, postavljaju alarmi te izvršavaju aktivnosti bazirane na promjenama koje vidimo u našim AWS resursima. CloudWatch već po zadanoj konfiguraciji daje dobar uvid u korištenje resursa, operativne performanse i ukupne obrasce potražnje računalnih resursa (npr. korištenje CPU-a na EC2 instancama, broj čitanja/pisanja na EBS instancama, broj simultanih konekcija na RDS bazu itd.) Po zadanoj konfiguraciji CloudWatch ne može prikupljati podatke o operacijskim sustavima i aplikacijama, ali zato možemo te metrike slati na CloudWatch preko API-a ili CLI-a. (Amazon, 2017)

3.4.6.3. Amazon CloudFormation

CloudFormation omogućava razvojnim inženjerima i sistemskim administratorima jednostavan način za zakupljivanje, ažuriranje i upravljanje AWS računalnim resursima. Kako infrastrukturu na AWS-u možemo koristiti kao programskim kod(eng. *Infrastructure as a Code*), tako je moguće koristiti gotove ili kreirati vlastite predloške koji sadrže sve informacije potrebne za uspješno izvršavanje željenih poslovnih procesa. Predlošcima se može upravljati kao što se upravlja sa softverskim proizvodima; verzioniranjem i slično, a za vizualizaciju i olakšano

upravljanje predlošcima može se koristiti AWS CloudFormation Designer servis. Ovo je po osobnom mišljenju najkorisniji servis AWS-a koji će višestruko olakšati i ubrzati korištenje dostupnih AWS resursa. Infrastruktura sa svim parametrima i uzorcima dizajna može biti zakupljena u roku od svega par minuta, kao što će biti prikazano u poglavlju 5.3 Inicijalizacija infrastrukture primjenom CloudFormation predloška.

4. Uzorci dizajna za AWS

Uzorci dizajna su kolekcija rješenja za tehnologiju računarstva u oblaku koji rješavaju česte sistemske probleme primjenom Amazon AWS-a. Općenito, uzorci dizajna su predlošci koji rješavaju određene probleme po najboljim praksama i mogu se primjenjivati u različitim situacijama. Izvor uzoraka u nastavku je web stranica (Cloud Design Pattern, 2013) koju su kreirali predani AWS sistemski inženjeri K. Suzuki, A. Tamagawa i H. Katayam kako bi sebi i drugima olakšali projektiranje arhitekture AWS računalnog oblaka.

4.1. Osnovni uzorci

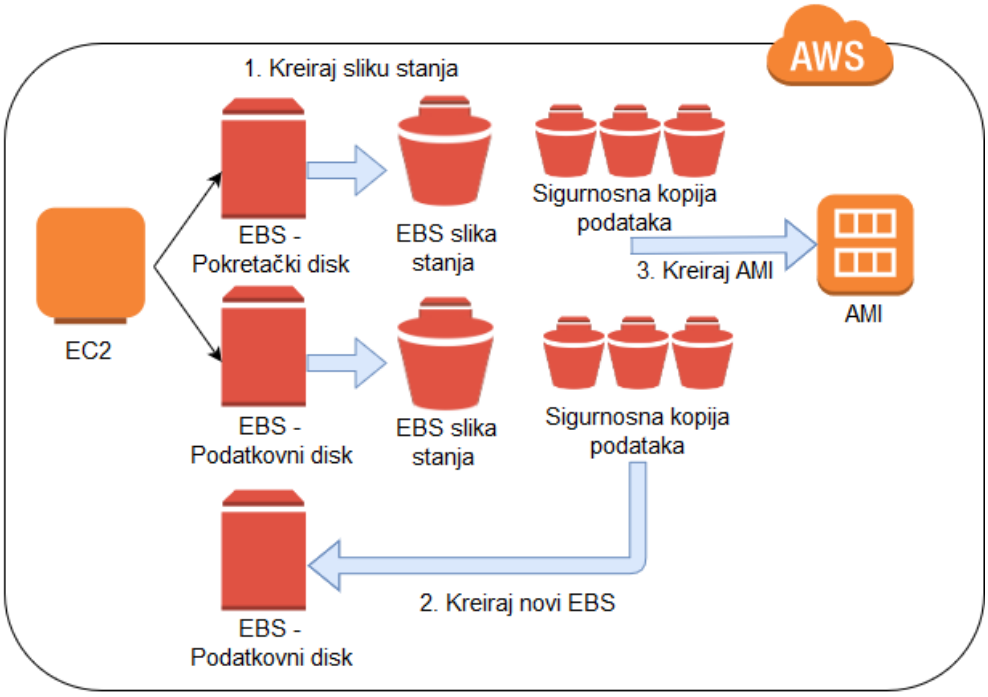
4.1.1. Sigurnosno kopiranje slike stanja

Jedna od najvažnijih sigurnosnih mjera u bilo kojem informacijskom sustavu je sigurnosno kopiranje podataka (eng. *Backup*). Dvije su glavne namjene sigurnosnog kopiranja podataka. Prva je slučaj nestanka ili kvara nad podacima, a druga, vračanje stanja sustava u neko od prethodnih stanja. Aktivnost prvog slučaja još nazivamo arhiviranje podataka, a drugog kreiranje slike stanja (eng. *Snapshot*).

Primjenom AWS servisa za pohranu podataka moguće je pohraniti "neograničenu" količinu podataka na siguran i relativno jeftin način. Lako je konfigurirati periodičko pohranjivanje slika stanja i na taj način automatizirati sigurnosnu proceduru. Također je moguće konfigurirati politiku zadržavanja podataka (eng. *Retention policy*) tako da se spremaju sve slike stanja ili zbog uštede brišu starije slike za koje smatramo da više neće biti potrebne. Slike stanja mogu biti svi podaci u jednom vremenskom periodu ili čak kompletno stanje operacijskog sustava nad kojim se izvode aplikacije.

Implementacija ovog uzorka sastoji se od *Elastic Block Storage* (EBS) servisa koji ima ulogu virtualne pohrane podataka, poput čvrstog diska. U EBS-u postoji funkcija za *snapshot* koja pohranjuje sliku stanja na AWS servis pod imenom *Simple Storage Service* (S3). To je servis za online pohranu objekata koji je dizajniran da bude visoko raspoloživ, trajan i siguran. Nakon spremanja slike stanja na S3, u bilo kojem trenutku može se vratiti prethodno stanje kao nova EBS instanca. Kada se EBS koristi kao disk za pohranu operacijskog sustava (eng. *Boot disk*), slika stanja se na S3 može pohraniti kao *Amazon Machine Image* (AMI) iz koje se pak mogu kreirati nove *Elastic Compute Cloud* (EC2) instance virtualnog servera. Korisno je u ovom uzorku odvojiti EBS instance koje pohranjuju operacijski sustav od instanci koje sadrže podatke jer je vrlo vjerojatno da će se za podatke češće raditi sigurnosne kopije, a odvajanje će se ubrzati proces kopiranja. Valja imati na umu da kreiranje slike stanja zadržava

konzistentnost podataka tj. da se izvode u trenutcima logičnim za pohranu stanja. (CDP:Snapshot Pattern, 2012)



Slika 10: Konfiguracija uzorka dizajna - Sigurnosno kopiranje slike stanja (Prema: Suzuki, Tamagawa i Katayam, 2012)

4.1.2. Replikacija servera

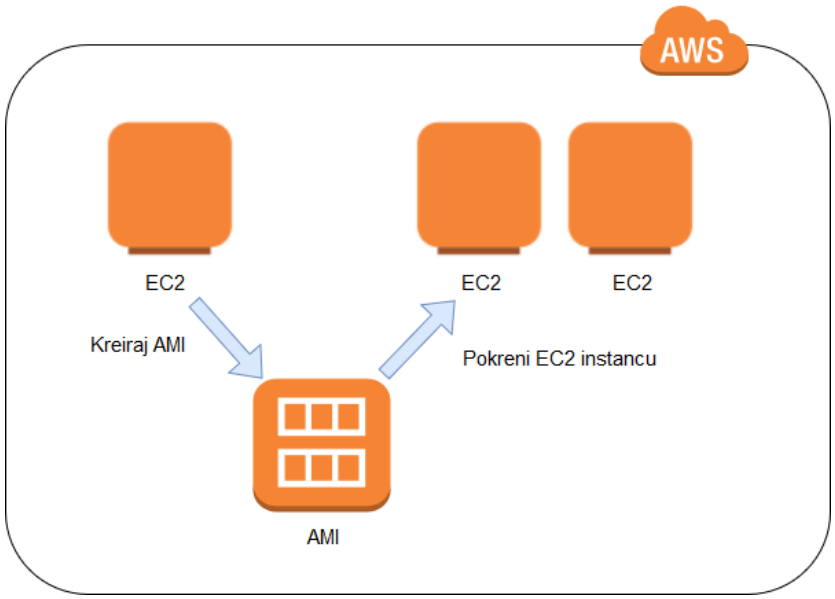
Konfiguriranje operacijskog sustava i aplikacija koje se izvode na virtualnom serveru zahtijeva jednako vremena kao i konfiguracija fizičkog servera. Virtualni serveri uglavnom imaju ista svojstva kao i fizički serveri. Razlikuju se u tome što je njih moguće više od jednog emulirati nad fizičkom infrastrukturom. To je vrlo korisno jer je po potrebi moguće kreirati ili ukloniti virtualne servere. Kako ne bi za svaki od tih virtualnih servera trošili vrijeme na instalaciju i konfiguriranje koristi se uzorak dizajna za replikaciju servera (eng. *Stamp Pattern*).

Uzorak dizajna za replikaciju servera odnosi se na mogućnost AWS-a da se nakon konfiguriranja servera i instalacije programa kreira slika stanja koju je zatim moguće koristiti za pokretanje "neograničenog" broja virtualnih servera. U tradicionalnim IT infrastrukturama proces kreiranja sigurnosne kopije servera i zatim vračanja kopije na novi server, bio je dugotrajan i nije postojao jednostavan način da se to automatizira za veći broj servera. Primjenom računarstva u oblaku takve aktivnosti postale su jednostavne jer se računalni resursi poput diskova, servera i slično, koriste poput pisanja programskog koda.

Za implementaciju ovog rješenja kreiraju se *Amazon Machine Image* (AMI) iz slika stanja koje su kreirane na *Elastic Block Store* (EBS) virtualnom disku i sadrže operacijski

sustav. Procedura je: lansiranje *Elastic Compute Cloud* (EC2) instance i instalacije željenog softvera, pripremanje logičkog stanja sustava za kreiranje slike stanja, kreiranje slike stanja, kreiranje AMI-a i korištenje kreiranog AMI-a za lansiranje novih servera po potrebi. (CDP:Stamp Pattern, 2012)

Kreirane AMI-e moguće je koristiti preko programskih skripti i time automatizirati kreiranje servera. Također ih je moguće dijeliti s drugim korisnicima AWS-a osobno ili objaviti na AWS AMI online tržištu (eng. *Marketplace*).



Slika 11: Konfiguracija uzorka dizajna - Replikacija servera (Prema: Suzuki i sur., 2012)

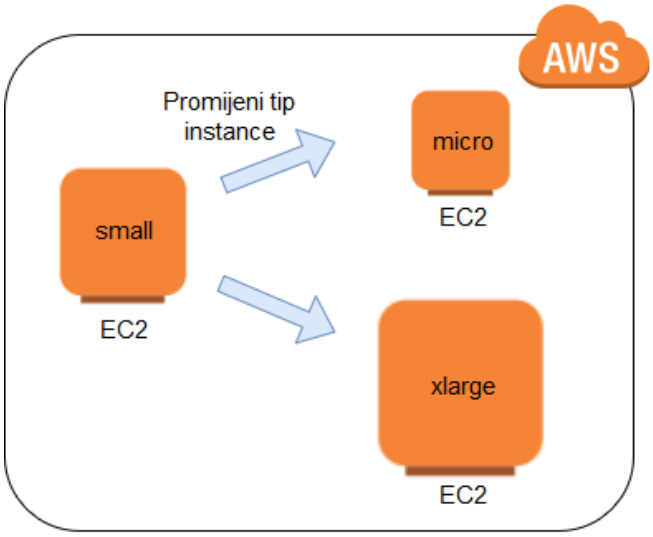
Replicirani serveri su u potpunosti isti. Ako na serveru postoje objekti koji zahtijevaju da su unikatni, tada će biti potrebno pronaći novo rješenje za taj problem, bilo ručnim izmjenama ili programskim skriptama. Također, svaki server postaje nova zasebna jedinica, pa izmjene na jednoj instanci neće imati utjecaja na druge. Radnje poput sigurnosnih ažuriranja i slično, biti će potrebne na svakoj instanci zasebno.

4.1.3. Dinamičko skaliranje specifikacija servera

Kod projektiranja informacijskog sustava postoji faza u kojoj je potrebno procijeniti potrebne računalne resurse i to je uglavnom težak zadatak. Ako računalni resursi nisu dovoljni, može doći do problema ili prestanka rada sustava. S druge strane, višak računalnih resursa stvara nepotrebne troškove jer infrastruktura nema dobru iskoristivost. Jedno od rješenja ovog problema je skaliranje specifikacija računalne infrastrukture po potrebi. U klasičnom podatkovnom centru je to teško izvesti jer je potrebno nadograđivati fizičku infrastrukturu i na novo instalirati i konfigurirati sustav.

AWS omogućava izmjenu specifikacija virtualne računalne infrastrukture (eng. *Scale-Up*) po potrebi, čak i nakon razvijanja i pokretanja sustava. Alatima za nadzor virtualne računalne infrastrukture moguće je nadzirati izvođenje sustava i potrošnja računalnih resursa, te po potrebi pojačati ili smanjiti specifikacije servera poput snagu procesora, količinu radne memorije i slično. Moguće je i automatizirati ovaj proces u slučaju da možemo identificirati uzorak ponavljanja potrebe za skaliranjem servera. Recimo da krajem svakog mjeseca naraste potrošnja, postavimo u raspored automatsko pojačanje resursa krajem mjeseca, a početkom smanjivanje na početno stanje.

Za implementaciju ovog uzorka potrebno je razviti sistem i pokrenuti instancu EC2 virtualnog servera te nadgledati iskorištavanje resursa. Nakon utvrđivanja potrebe za izmjenom specifikacija servera potrebno ga je na kratko zaustaviti, odabrati instancu sa specifikacijama koje nam odgovaraju (eng. *Change instance type*) i ponovno pokrenuti server. (CDP:Scale Up Pattern, 2012)



Slika 12: Konfiguracija uzorka dizajna - Dinamičko skaliranje specifikacija servera (Prema: Suzuki i sur., 2012)

Jedna od negativnih strana ovog uzorka je obavezno stopiranje i ponovno pokretanje EC2 virtualnih servera kod izmjene specifikacija. Trajanje ponovnog pokretanja kreće se u rasponu od trideset sekundi do nekoliko minuta ovisno o kompleksnosti razvijenog sustava. Možda za neke sustave to neće biti prihvatljivo. Drugi problem je ograničenost skaliranja specifikacija servera s zadanim tipovima EC2 instanci koje je moguće koristiti. Ukoliko postoji potreba za boljim specifikacijama od najbolje koja je u ponudi, tada se predlaže korištenje uzorka dizajna za dinamičko skaliranje broja servera umjesto specifikacija servera.

4.1.4. Dinamičko skaliranje broja servera

Da bi web poslužitelj mogao obraditi veliku količinu prometa važno je da ima visoke specifikacije. Primjenom uzorka dizajna za dinamičko skaliranje specifikacija servera (eng. *Scale-Up*) možemo riješiti taj problem ali postoje problemi poput: više cijene jačih servera, obaveznog stopiranja sustava nakon izmjena specifikacija i ograničenosti postojećim tipovima specifikacija.

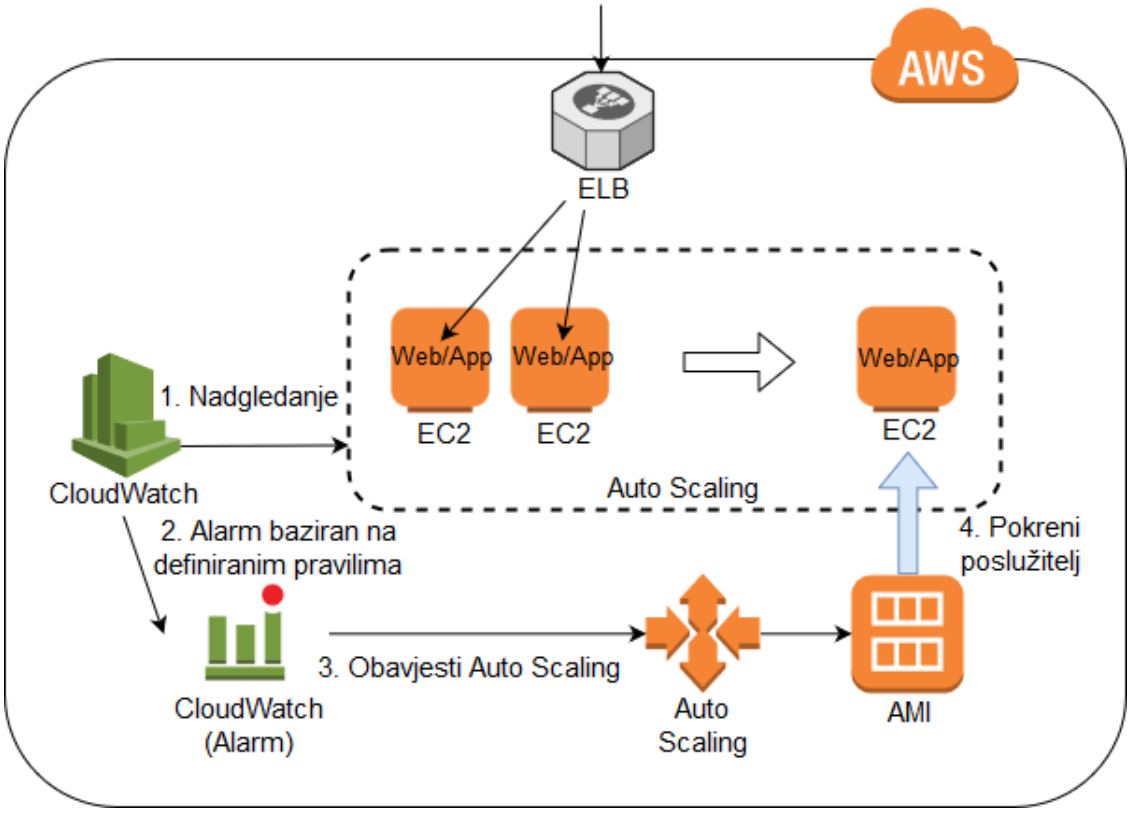
Bolji pristup za rješavanje problema velike količine prometa na serveru je uzorak dizajna za dinamičko skaliranje broja servera (eng. *Scale-Out*). U ovom pristupu koristiti se veći broj virtualnih servera i uređaj za ujednačavanje opterećenja (eng. *Load Balancing*) koji ravnomjerno distribuira opterećenje preko više individualnih instanci virtualnog servera. Primjenom ovog uzorka moguće je u potpunosti automatizirati skaliranje broja virtualnih servera i postići još višu razinu efikasnosti sustava. Uzorak kontinuirano pruža visoku dostupnost i minimizira troškove smanjenjem broj servera koji nisu potrebni.

Za implementaciju *Scale-Up* uzorka na AWS-u uz EC2 virtualne servere potrebna su nam tri servisa: servis za ujednačavanje opterećenja (eng. *Elastic Load Balancing* [ELB]), servis za automatsko skaliranje (eng. *Auto Scaling*) i servis za nadgledanje AWS resursa i aplikacija koje se u AWS-u izvode, *CloudWatch.* (CDP:Scale Out Pattern, 2012)

U nastavku je procedura za implementiranje *Scale-Up* uzorka:

- Kreirati *Amazon Machine Image* (AMI) koji će biti korišten za pokretanje EC2 instanci.
- Pokrenuti EC2 instance u paralelnoj konfiguraciji nad kontrolom ELB-a.
- Definirati postavke za aktiviranje povećana ili smanjenja broja EC2 instanci. Metrike koje se mogu koristiti za aktivaciju su prosječno korištenje procesora EC2 instance, količina mrežnog prometa, broj otvorenih sesija, latencija EBS virtualnih diskova i slično.
- Konfigurirati *CloudWatch* servis da nadgleda postavljene metrike i aktivira alarm ako su zadovoljeni definirani uvjeti.
- Konfigurirati *Auto Scaling* da prihvaća alarme od *CloudWatch* servisa i shodno tome poveća ili smanji broj EC2 instanci.

Primjer korištenja navedene konfiguracije (Slika 13) bio bi sustav s inicijalno dvije EC2 instance i pravilom da se pokrene još jedna EC2 instanca ako je prosječna iskorištenost svih EC2 procesora veća ili jednaka od 70% kontinuirano u periodu od pet minuta.



Slika 13: Konfiguracija uzorka dizajna - Dinamičko skaliranje broja servera (Prema: Suzuki i sur., 2012)

Preporučene mjere opreza i dodatni detalji za korištenja *Scale-Out* uzorka:

- Uzorak se ne može nositi s ekstremnim varijacijama u mrežnom prometu, poput duplog ili trostrukog rasta u periodu od nekoliko minuta. Razlog tome je sporost pokretanja EC2 instanci nakon prihvaćenog alarma na *Auto Scaling* servisu. Normalno je da treba određeno vrijeme za pokretanje instance i registriranje iste pod kontrolom ELB-a. Rješenje tog problema je ili definirati raspored kada povećati broj EC2 instanci ili cijelo vrijeme imati redundantne EC2 instance aktivne, ali time naravno raste i trošak sustava.
- Sve EC2 instance pod kontrolom ELB-a bi trebale biti identične jer ELB nema način da distribuira opterećenje u skladu sa specifikacijama pojedine instance.
- Razmotriti opciju da se upravljanje HTTP sesijama, SSL procesima i slično odvoji na zasebne EC2 instance koje neće biti pod kontrolom ELB-a.
- Kod ažuriranja EC2 instanci potrebno je ažurirati i AMI koji je izvor za *Auto Scaling* servis.
- Za povećanje otpornosti na kvarove moguće je distribuirati *Scale-Out* uzorak preko nekoliko različitih AWS raspoloživih zona (eng. *Availability Zone* [AZ]) i taj uzorak se zove redundancija na razini podatkovnog centra (eng. *Multi-Datacenter Pattern*).

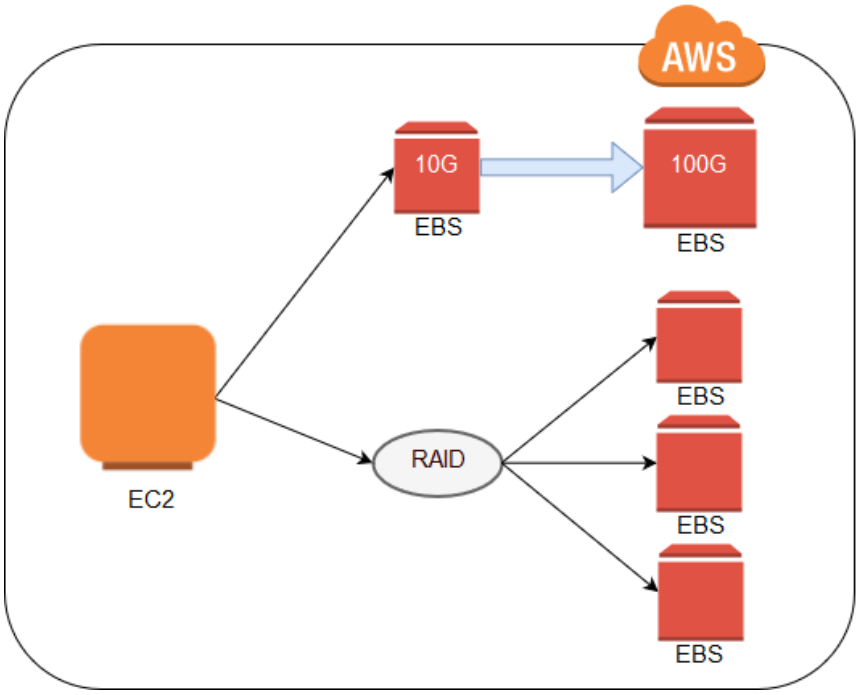
Kako bi se ostvarila jednaka distribucija opterećenja unutar svake AZ, trebalo bi svaku novu instancu pokrenuti u svim korištenim raspoloživim zonama.

4.1.5. Dinamičko skaliranje kapaciteta diska

Kod projektiranja informacijskog sustava u fazi procijene potrebnih računalne resursa, između ostalog potrebno je i procijeniti kapacitete diskova za pohranu podataka. To je težak zadatak jer se procijene rade za nekoliko godina u naprijed uz dodatak sigurnosnih kapaciteta. Te diskove potrebno je platiti u naprijed i instalirati te se većina njih neće dugi period niti koristiti.

AWS oblak pruža mogućnost korištenja "neograničenog" broj virtualnih diskova. Primjenom virtualnih diskova detaljna procjena potrebnih kapaciteta postaje suvišna. Nakon razvijanja i pokretanja sustava na infrastrukturi AWS-a moguće je nadgledati iskorištenost kapaciteta diskova i po potrebi rezervirati dodatne diskove. Virtualni diskovi s vremenom postaju sve jeftiniji jer AWS kao javni oblak ima koristi od ekonomije razmjera, stoga je isplativo naknadno dodavati nove virtualne diskove u sustav. Ako se ispostavi da postoje viškovi, lako se virtualni diskovi uklone i stvaraju se nove uštede. Virtualne diskove moguće je konfigurirati u polje diskova da međusobno dijele i repliciraju podatke (eng. *Redundant Array of Independent Disks* [RAID]). Time se utječe na performanse, pouzdanost, dostupnost i kapacitet virtualnih diskova. Moguće je i automatizirati dodavanje novih diskova primjenom *Auto Scaling* AWS servisa.

Virtualni diskovi u AWS-u su apstrahirani u servis pod imenom *Elastic Block Store* (EBS). Za implementaciju prvo je potrebno rezervirati minimalne kapacitete virtualnih diskova primjenom EBS-a. Za povećanje kapaciteta diska dobra je praksa prvo kreirati sliku stanja postojećeg diska da služi kao sigurnosna pohrana. Zatim kreirati novi disk većih kapaciteta, dodati postojećoj EC2 instanci i izvršiti naredbu za ekspanziju kapaciteta na novi diskovni prostor. Za kreiranje RAID polja potrebno je dodati željeni broj diskova na EC2 instancu i konfigurirati ga nekim alatom za upravljanje RAID polja (npr. mdadmin za Linux operacijske sustave). (CDP:Ondemand Disk Pattern, 2012)



Slika 14: Konfiguracija uzorka dizajna - Dinamičko skaliranje kapaciteta diska (Prema: Suzuki i sur., 2012)

Kod rezerviranja novih EBS virtualnih diskova naplata se vrši po rezerviranom kapacitetu diska, za razliku od AWS S3 servisa gdje se plaća po iskorištenom kapacitetu. Na primjer, rezervacija 100 GB virtualnog diska biti će naplaćena kao da je 100 GB podataka na njemu, makar je zapravo samo 5 GB.

4.2. Uzorci za visoku raspoloživost

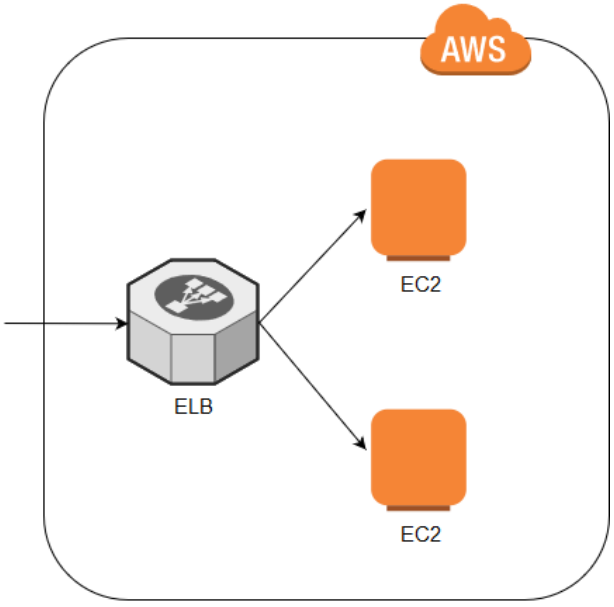
4.2.1. Redundancija servera

Pojam redundancije u tehničkom smislu odnosi se na višak istovrsnih komponenata u nekom složenom sustavu bez kojih bi sustav mogao raditi, ali se komponente umnožavaju zbog sigurnosnih razloga. Ponekad jednostavno povećavanje broja komponenata nije dovoljno da bi struktura sustava bila redundantna, već je potrebno poduzeti dodatne korake konfiguriranja sustava.

Primjenom AWS oblaka razvijanje sustava s redundantnim serverima je puno jednostavniji proces nego što je to slučaj s fizičkom infrastrukturom. Koristi se servis za ujednačavanje opterećenja (eng. *Elastic Load Balancing* [ELB]) koji kontrolira paralelno spojene servere. Uzorak redundancije servera se još zove *Multi-server* uzorak. U njemu se dakle nalazi dodatan broj servera koji će u slučaju kvara preuzeti opterećenje i to omogućava nesmetano izvođenje cjelokupnog sistema. Preporuka za višu raspoloživost sustava je koristiti

takav sustav, naspram jednog jakog servera koji ako se pokvari prestaje kontinuitet izvođenja. Također se preporuča korištenje n+1 konfiguracije (gdje je n broj željenih servera) kako bi jedan zamjenski server uvijek bio instant spreman za korištenje (eng. *Hot spare*). U kombinaciji ELB-a i *Auto Scaling* servisa moguće je automatizirati proces dodavanja novih servera u slučaju kvarova. Taj uzorak je nadogradnja *Multi-server* uzorka i opisan je u poglavlju 4.1.4 Dinamičko skaliranje broja servera.

Implementacija *Multi-server* uzorka na AWS-u postiže se konfiguriranjem sistemske okoline i pokretanjem EC2 instanci primjenom *Stamp* uzorka, opisanog u poglavlju 4.1.2 Replikacija servera. Potrebno je pokrenuti ELB i s njim povezati pokrenute instance, te konfigurirati funkciju ELB-a za provjeru stanja EC2 instanci (eng. *Health check*). Ta funkcija će kontinuirano nadgledati stanje EC2 instanci i ako se na jednoj od njih dogodi kvar, ELB će prestati slati opterećenje toj instanci i preusmjeriti će taj proces na redundantnu instancu. (CDP:Multi-Server Pattern, 2012)



Slika 15: Konfiguracija uzorka dizajna - Redundancija servera (Prema: Suzuki i sur., 2012)

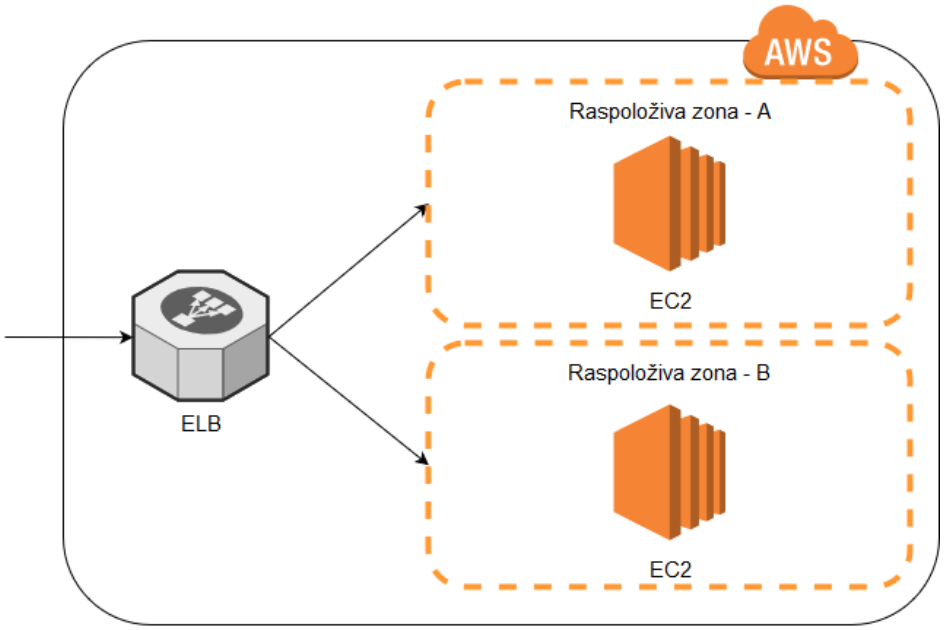
Trošak redundantnih sustava je veći jer se aktivno koristi veći broj komponenata, pa je potrebno napraviti procjenu isplativosti sustava s redundantnim komponentama. Dodatni problemi u ovakvom sustavu mogu nastati kada se podaci u aktivnim procesima trebaju dijeliti između više servera. Za dijeljenje podataka o sesijama potrebno je koristiti bazu podataka za sesije ili *Sticky Session* uzorak. Za redundantnost baza podataka problem će nastati kod sinkronizacije podataka između baza pa se preporuča korištenje *DB Replication* uzorka za replikaciju baza podataka.

4.2.2. Redundancija podatkovnog centra

Uzorak dizajna za redundanciju servera podosta podiže razinu raspoloživosti sustava, ali što ako se na razini podatkovnog centra dogodi cjelokupni nestanak električne energije ili konekcije na Internet, ili elementarna nepogoda poput potresa, poplave i slično? Za takve slučajeve potrebno je primijeniti arhitekturu primjenom uzorka dizajna za redundanciju podatkovnog centra.

Razvoj i posjedovanje fizičkih redundantnih podatkovnih centara je ekstremno skupo i multiplicira kompleksnost cjelokupnog informacijskog sustava. Potrebno je zadovoljiti puno zahtjeva za redundantne podatkovne centre. Moraju biti dovoljno udaljeni jedan od drugoga, po mogućnosti na različitim potresnim i poplavnim područjima. Moraju imati zasebne priključke na električnu i Internet mrežu, kvalitetnu konekciju s ostalim podatkovnim centrima itd.

Amazon AWS u vrijeme pisanja ovog rada ima raspoloživo šesnaest regija u kojima su četrdeset i četiri raspoložive zone. Svaka zona praktički predstavlja jedan podatkovni centar koji je dediciranom mrežom spojen sa svim ostalim centrima. To znači da željenu računalnu infrastrukturu možemo postaviti u dvije ili više različitih zona, bez dodatnih naknada, i time osigurati redundantnost na nivou podatkovnih centara. Možemo koristiti uzorak dizajna za replikaciju servera u svakoj zoni koju koristimo te na vrh arhitekture postaviti ELB koji će ravnomjerno distribuirati opterećenje poprijeko svih raspoloživih servera. Ta mogućnost se naziva *cross-zone load balancing*. Takav sustav biti će otporan na katastrofalne kvarove na razini podatkovnog centra, omogućavat će kontinuitet poslovanja i brz oporavak od katastrofe. (CDP:Multi-Datacenter Pattern, 2012)



Slika 16: Konfiguracija uzorka dizajna - Redundancija podatkovnog centra (Prema: Suzuki i sur., 2012)

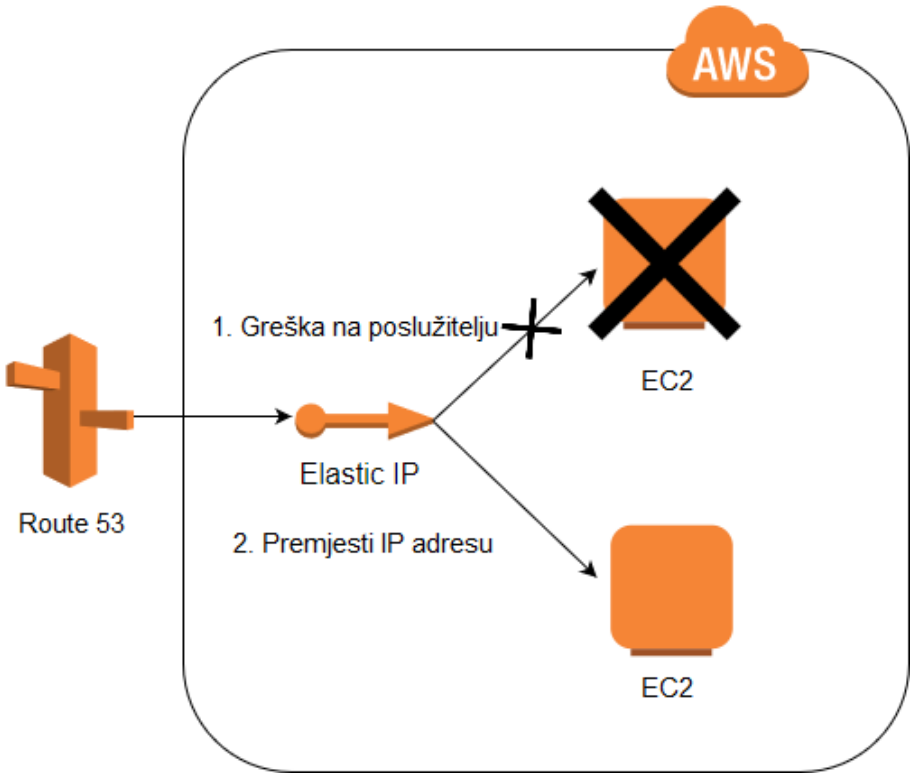
U slučaju kvara na jednoj od raspoloživih zona, broj servera u drugoj raspoloživoj zoni trebao bi biti dovoljan da preuzme cjelokupno opterećenje sustava. U protivnom će doći do usporavanja izvođenja ili čak i prestanka rada. Za taj slučaj bi se mogao iskoristiti uzorak dizajna za dinamičko skaliranje broja servera s obzirom na opterećenje.

4.2.3. Plutajuća IP adresa

Kada se pojavi potreba za ažuriranjem ili održavanjem servera, često je nakon završetka potrebno zaustaviti i ponovno pokrenuti server. Zaustavljanjem servera zaustavljamo ujedno i funkcionalnost koju on pruža. Primjenom uzoraka dizajna cilj nam je maksimalno smanjiti vrijeme koje će navedeni server biti nedostupan. Za web servere jedna mogućnost je korištenje domenskog sustava imena (eng. *Domain Name System* [DNS]) za zamjenu servera, ali ta tehnika nije primjenjiva za instant izmjenjivanje svih tipova servera koji su u kvaru.

Primjenom tradicionalne fizičke infrastrukture proces zamjene servera bio je dugotrajan. Prvo je bilo potrebno postaviti sekundarni server i njega konfigurirati ili postaviti prethodno snimljenu sigurnosnu kopiju. Zatim bi se IP adresa prvog servera postavila na novi server i time bi završila zamjena servera. Navedeni postupak moguće je rekreirati jednostavno i brzo primjenom AWS oblaka. Prije zamjene servera pripremi se AMI koji će služiti za pokretanje novih servera, a zamjenu IP adresa moguće je automatizirati primjenom skripti.

U Amazon AWS oblaku statička IPv4 adresa dizajnirana za dinamičko okruženje računarstva u oblaku naziva se *Elastic IP address* [EIP] i moguće ju je lako premjestiti s jedne EC2 instance na drugu u slučaju kvara ili ažuriranja sustava u kojem je potrebno gašenje. Može se koristiti dizajn uzorka za replikaciju servera kod pokretanja novih EC2 instanci. Premještanje EIP-a moguće je još brže izvršiti ako je jedna instance prethodno pripremljena kao *Hot Spare*. U slučaju greške kod prebacivanja EIP s jedne instance na drugu, postoji sigurnosni mehanizam koji automatski vraća EIP na originalni server. EIP je također moguće premjestiti na servere u drugoj zoni raspoloživost (AZ) čime se postiže još viša raspoloživost sustava. (CDP:Floating IP Pattern, 2012)



Slika 17: Konfiguracija uzorka dizajna - Plutajuća IP adresa (Prema: Suzuki i sur., 2012)

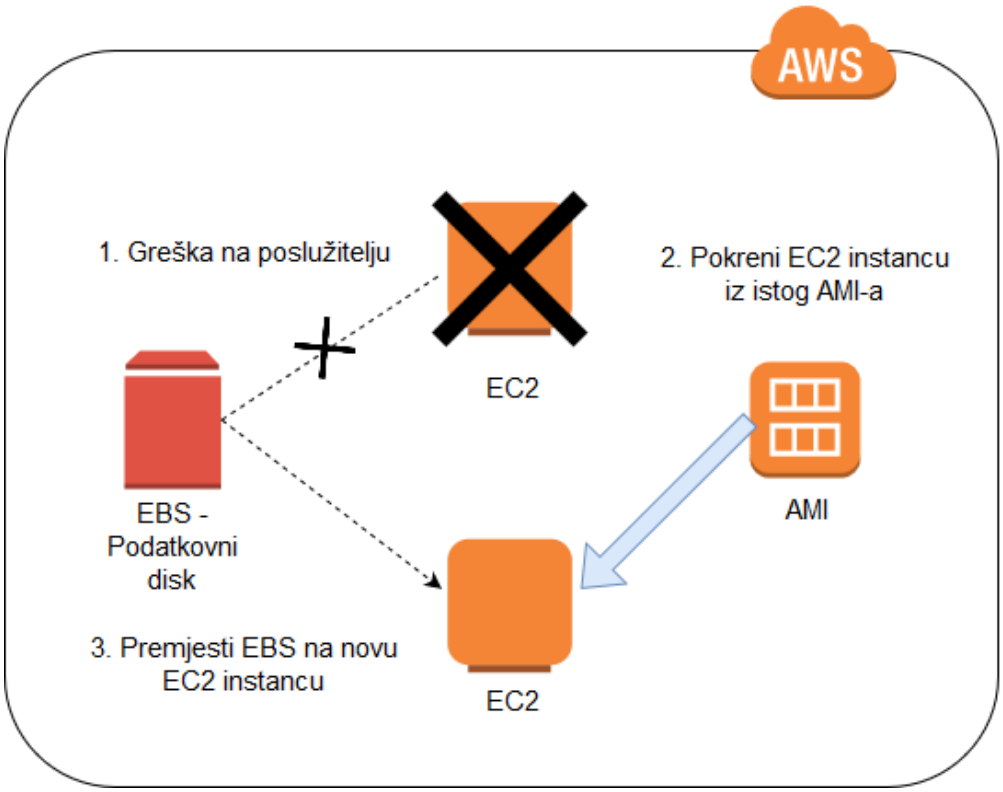
Prebacivanje EIP-a s jedne instance na drugu je jako brzo, u pravilu nekoliko sekundi. Ono što može potrajati malo duže je mogućnost SSH pristupa instanci s dodijeljenom EIP. Navedeni uzorak moguće je koristiti u paru s uzorkom dizajna za zamjenu servera kod kojeg se podatkovni sustav (EBS) također dodjeljuje novo pokrenutoj instanci.

4.2.4.Zamjena servera

Kvarovi u informacijskim sustavima su normalna pojava i zato je potrebno primjenjivati uzorke dizajna koji nam omogućavaju otpornost na kvarove. Uzroci kvara na serveru mogu biti svakojaki ali najčešće nije problem u diskovima za pohranu. U takvim slučajima možemo na ispravan disk spojiti drugi server koji je ispravan, te on preuzima opterećenje servera u kvaru. U tradicionalnom podatkovnom sustavu procedura za izmjenu servera je dugotrajna i spora. Nove servere treba nabaviti, instalirati i konfigurirati, a podatkovni sustav prebaciti na novi server.

Primjenom računarstva u oblaku, konkretno Amazon AWS-a, zamjena servera je jednostavan i brz postupak uz čuvanje prethodnih podataka i slike stanja sustava. Ne samo to, već je proces zamjene moguće u potpunosti automatizirati. Glavni razlog koji to omogućava je EBS servis koji predstavlja podatkovni sustav a neovisan je o radu EC2 instanci. Nakon kvara na EC2 instanci, pokreće se nova instanca primjenom AMI-a, te se na nju spaja prethodno korišteni EBS. Podaci s EBS-a se postavljaju (eng. *Restore*) na novu instancu i

sustav nastavlja raditi u zadnjem validnom stanju prije kvara. Za automatizaciju ovog procesa predlaže se korištenje softvera za nadgledanje (eng. *Monitoring*) koji će prepoznati kvar, oglasiti alarm i započeti proces zamjene servera. (CDP:Server Swapping Pattern, 2012)



Slika 18: Konfiguracija uzorka dizajna - Zamjena servera

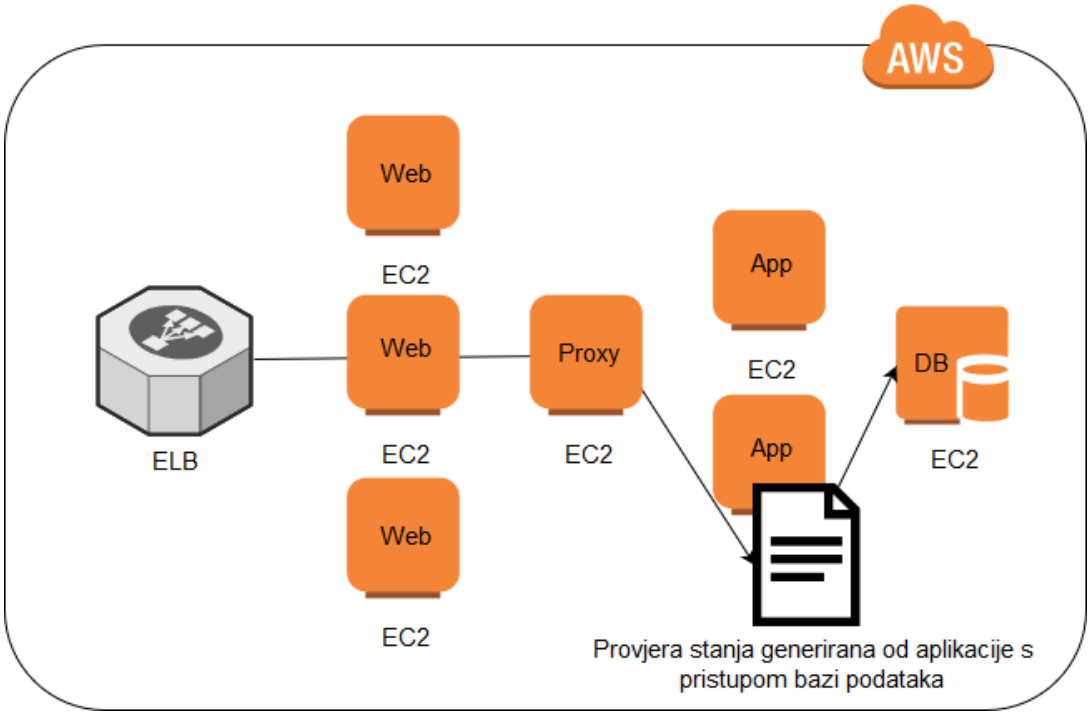
Uzorak zamjene servera preporuča se koristiti uz uzorak plutajuće IP adrese kako bi se sa servera u kvaru prebacila i statička IP adresa na novi server. Za slučaj kvara EBS podatkovnog sustava potrebno je pripremiti odgovarajući plan poput pohranjivanja sigurnosnih kopija.

4.2.5.Duboka provjera stanja servera

Primjena Amazon AWS servisa za ujednačavanje opterećenja (ELB) omogućava nam provjeru stanja (eng. *Health Check*) EC2 instanci nad kojima ELB ima kontrolu. Recimo da imamo vertikalno konfiguriran sustav s više servera. Primjerice, web servere, *proxy* server, aplikacijske servere i server baze podataka. ELB može provjeriti stanje web servera koji su direktno na njega spojeni, ali što ako želimo vidjeti stanje svih servera u pozadini (eng. *Back-end*)?

Rješenje za dohvaćanje stanja svih servera je konfiguriranje ELB-a da provjeru stanja dohvaća s dinamičke web lokacije koju generira aplikacija za provjeru stanja svih servera. Aplikacija može biti kreirana u programskom jeziku PHP, Java ili nekom trećem, te joj svrha

treba biti dohvaćanje individualnih stanja servera te generiranje izvještaja kojeg će ELB koristiti. (CDP:Deep Health Check Pattern, 2012)



Slika 19: Konfiguracija uzorka dizajna - Duboka provjera stanja servera (Prema: Suzuki i sur., 2012)

Preporuka je uz navedeni uzorak koristiti i uzorak dizajna za replikaciju baze podataka čime bi se izbjegla potencijalna jedinstvena točka kvara sustava (eng. *Single Point of Failure*). U slučaju da dođe do neočekivanog prestanka rada servera za bazu podataka, aplikacija za provjeru stanja neće moći dohvatiti podatke iz baze te bi slijedno ELB mogao reagirati na to kao da su svi serveri prestali raditi.

4.3. Uzorci za relacijske baze podataka

4.3.1. Replikacija baza podataka

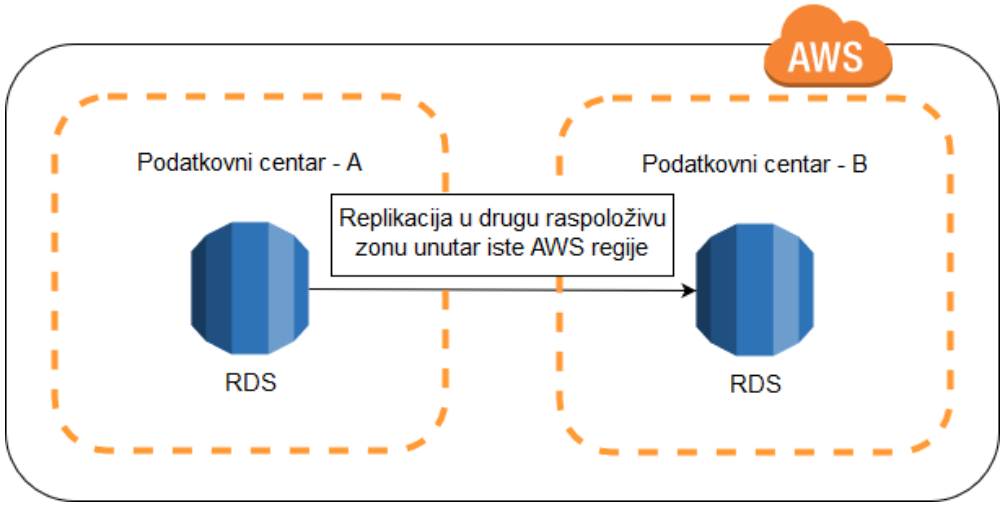
Osnovni oblik pohrane podataka je pohrana u bazu podataka. Iako se funkcionalnost replikacije već dugo koristi u tradicionalnim podatkovnim centrima, moguće je podići raspoloživost baze podataka na višu razinu primjenom replikacije na drugu geografsku lokaciju. Dok je za to u prošlosti bio potreban još jedan podatkovni centar, primjenom AWS-a to je ostvarivo kroz nekoliko jednostavnih koraka.

AWS nudi dva pristupa za implementaciju replikacije baze podataka u drugu raspoloživu zonu (AZ): primjena klasičnih EC2 instanci na koje je instalirana baza podataka i

AWS servis za relacijske baze podataka (RDS) s podrškom za višestruke raspoložive zone (eng. *Multi-AZ*). Oba pristupa sinkronizirano repliciraju relacijske baze podataka kroz više AZ-a i pružaju redundanciju podataka, poboljšavaju ulazne/izlazne (eng. *Input/output* [I/O]) performanse i minimiziraju latencije tijekom kreiranja sigurnosnih kopija sustava. RDS je moguće konfigurirati da vrši podršku za neuspjeh izvođenja (eng. *Failover support*) tj. da automatski nakon prekida normalnog funkcioniranja primarne baze podataka prebaci opterećenje na repliciranu bazu koja je u drugoj AZ. Moguće je postaviti alarm kod nastanka *failover* procesa kako bi zaduženi sistemski administratori bili upozoreni. Također mogu kroz RDS konzolu nadgledati izvođenje *failover* procesa. (CDP:DB Replication Pattern, 2012)

Razlozi *failover* procesa mogu biti sljedeći:

- Prekid rada AZ-a,
- Prekid rada primarne baze podataka,
- Promjena vrste servera za bazu podataka,
- Ažuriranje operacijskog sustava zahtjeva ponovno pokretanje servera,
- Ručno pokretanje *failover* postupka.



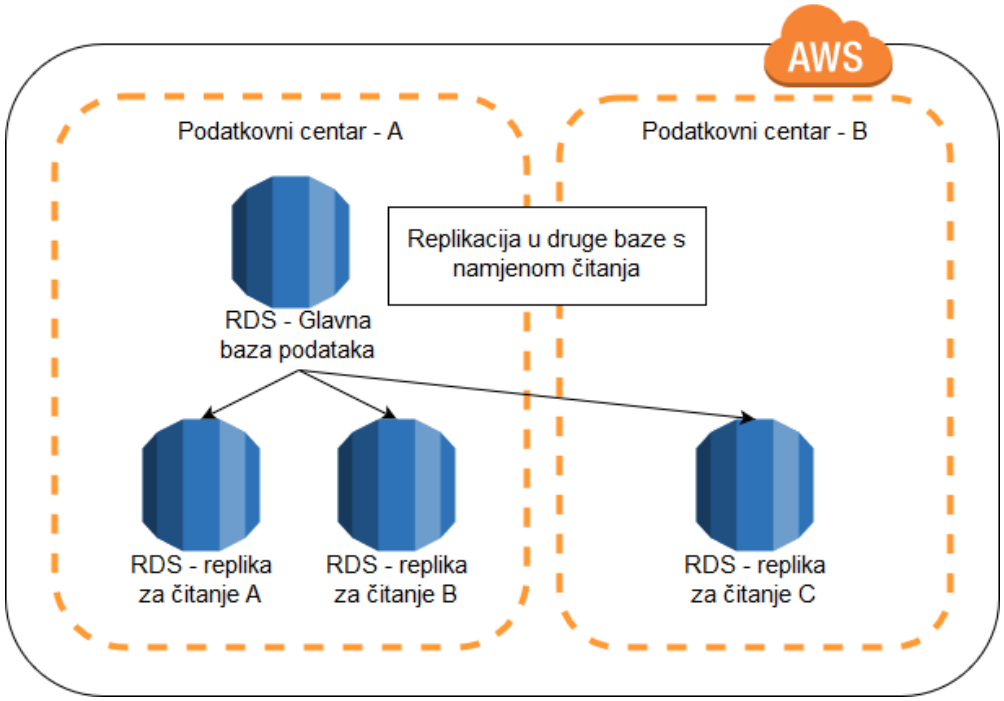
Slika 20: Konfiguracija uzorka dizajna - Replikacija baze podataka (Prema: Suzuki i sur., 2012)

Kod korištenja replikacije baze podataka u drugu regiju preporuča se postavljanje asinkrone replikacije ili periodičke replikacije podataka. Razlog je mogući pad performansi izvođenja baze podataka kada se koristi sinkronizirana replikacije velike količine podataka. Također, u slučaju pokretanja *failover* procesa, automatsko prebacivanje na repliciranu bazu podataka nije instant.

4.3.2. Distribucija opterećenja čitanja

Kod velikih opterećenja baze podataka pojavljuje se potreba za nadogradnjom sustava baza podataka. U slučaju da pojačanje specifikacija (eng. *Scale-up*) nije opcija, moguće je pojačati broj instanci servera baza podataka (eng. *Scale-out*) kako bi se distribucija opterećenja izvršavala nad horizontalnim sustavom. Pošto proces čitanja iz baze podataka najčešće opterećuje sustav više od pisanja, ideja ovog uzorka je distribucija procesa čitanja podataka po tzv. replikama. Replike za čitanje su baze podataka koje se razlikuju od normalne baze po tome što služe samo za čitanje podataka.

Primjenom AWS servisa za relacijske baze podataka RDS možemo iskoristiti funkcionalnost menadžment sistema (eng. *Relational Database Management Systems* [RDBMS]) za jednostavno kreiranje replika za čitanje. Također, moguće je konfigurirati i klasičnu EC2 instancu da ima namjenu replike za čitanje. Opterećenje može biti ravnomjerno distribuirano preko svih replika ili se može konfigurirati da se recimo koriste sve osim *master* baze podataka od strane neke aplikacije koja puno čita iz baze (npr. aplikacija za analitiku). (CDP:Read Replica Pattern, 2012)



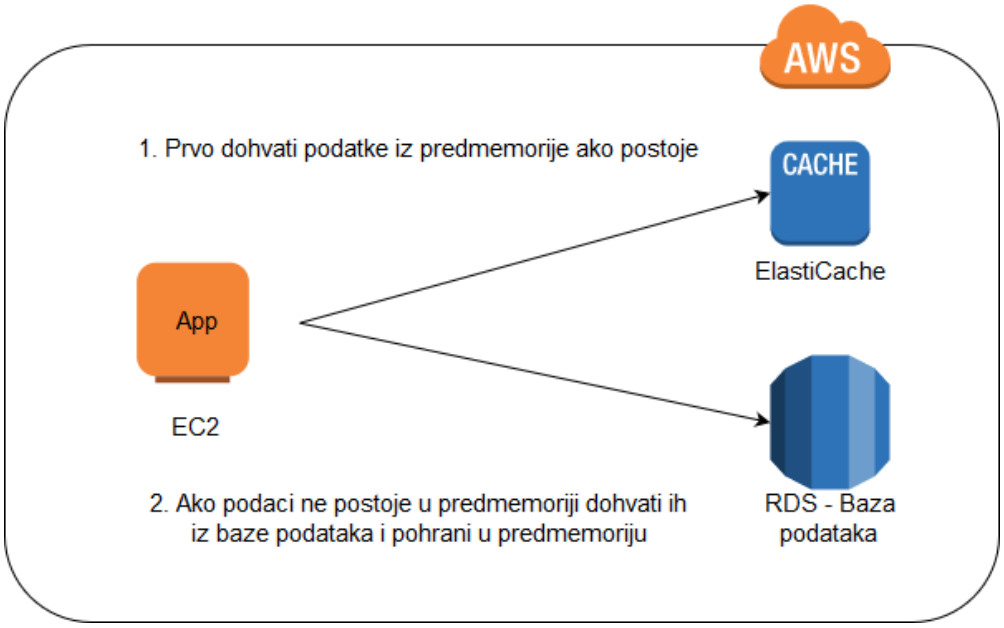
Slika 21: Konfiguracija uzorka dizajna - Distribucija opterećenja primjenom replika za čitanje (Prema: Suzuki i sur., 2012)

Replike za čitanje kopiraju podatke iz *master* baze podataka asinkrono. Zbog toga postoji malo kašnjenje u ažurnosti podacima. Ako se od arhitekture sustava traži visoka raspoloživost predlaže se korištenje uzorka dizajna za replikaciju baza podataka. Oba uzorka moguće je koristiti u paralelnoj konfiguraciji.

4.3.3. Korištenje predmemorije

Većina opterećenja nad bazom podataka uglavnom se odnosi na proces čitanja podataka. Poboljšavanjem performansi čitanja najviše utječemo na poboljšanje performansi kompletne baze podataka. Iz tog razloga ovaj uzorak dizajna predlaže pohranjivanje često korištenih podataka u priručnu memoriju, odnosno predmemoriju (eng. *Cache*). Pošto se podaci spremaju u predmemoriju umjesto na disk, podaci se višestruko brže mogu koristiti. Tipični primjer korištenja predmemorije je za pohranu rezultata kompleksnih i dugotrajnih kalkulacija koji se zatim mogu koristiti bez potrebe za ponovnim izvođenjem istih.

Za implementaciju navedenog uzorka primjenom AWS-a koristi se servis ElastiCache, koji omogućava pohranu podataka u predmemoriju. Na softverskoj razini može se birati između dvije opcije otvorenog koda: Redis i Memchached. ElastiCache detektira kvarove i automatski ih popravlja, što ga čini poželjnim u visoko raspoloživim sustavima. Funkcionira na način da kada u sustavu nastane potreba za čitanjem podataka, prvo se traže u predmemoriji a zatim u bazi podataka. (CDP:Inmemory DB Cache Pattern, 2012)



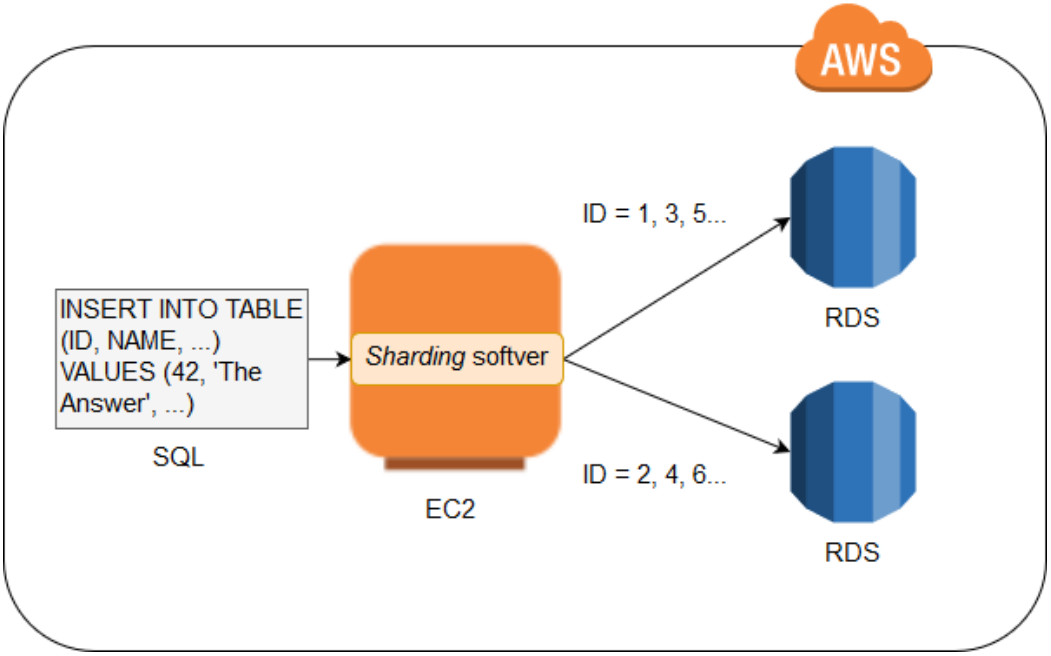
Slika 22: Konfiguracija uzorka dizajna - Korištenje predmemorije (Prema: Suzuki i sur., 2012)

Korištenje predmemorije vrlo vjerojatno će zahtijevati modifikaciju programa za pristup bazi podataka i trebati će kreirati strategiju za odabir podataka koji će se pohranjivati tamo, te koliko dugo da tamo ostanu. Ukoliko će se skupljati podaci koji se ne koriste trošak korištenja usluge će rasti za nikakvu povratnu dobit.

4.3.4. Distribucija opterećenja pisanja

U sustavima gdje proces pisanja u bazu podataka stvara opterećenje na performanse koje želimo optimizirati, a *Scale-Up* opcija nije dostupna, koristit ćemo uzorak dizajna za *Scale-out* horizontalnu distribuciju opterećenja procesa pisanja.

Uzorak primjenjuje *sharding,* tehniku za odvajanje jednog procesa pisanja u bazu na više manjih dijelova i distribuciju svakog na zasebnu instancu baze podataka. Njima upravlja softver za *sharding;* svaka od baza ima dio podataka koji je identičan i dio koji je unikatan za pojedinu bazu. Dodjeljuju im se odgovarajući stupci tablica kao privatni ključevi po kojima se zna koja baza dobiva koje podatke. Na taj način postiže se učinkovitost u radu, a primjenom *Multi-AZ* još viša raspoloživost. (CDP:Sharding Write Pattern, 2012)



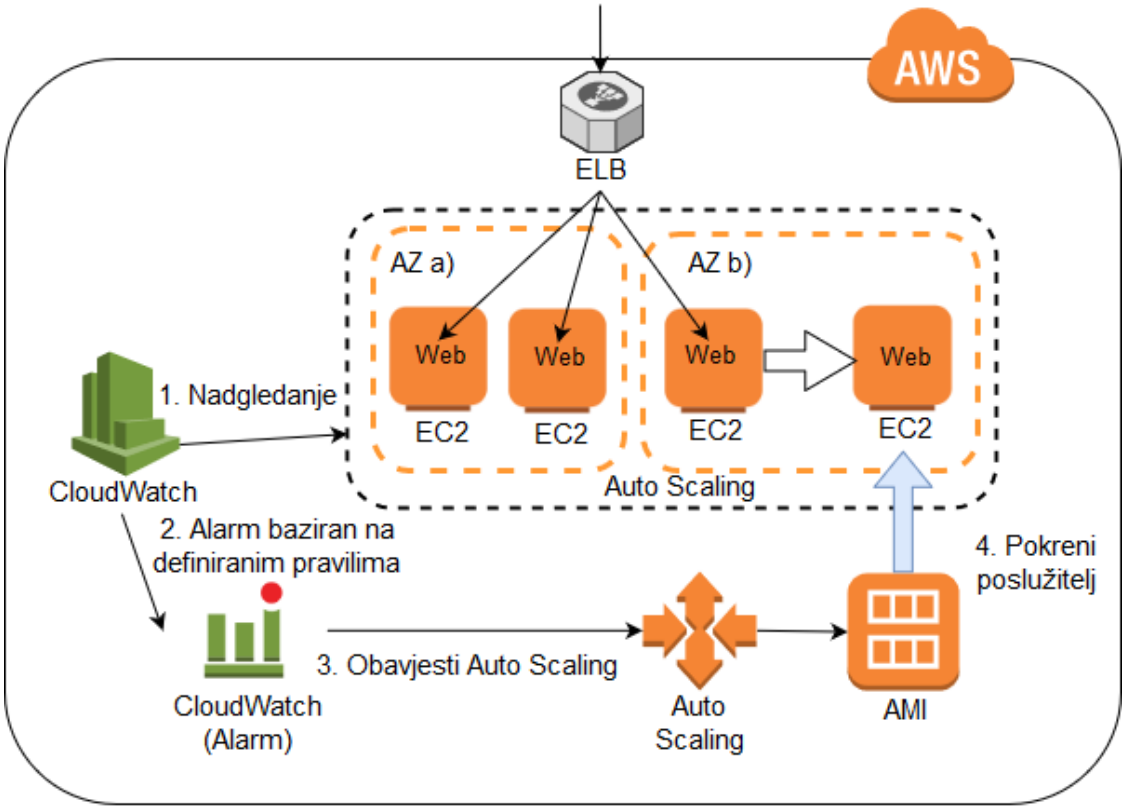
Slika 23: Konfiguracija uzorka dizajna - Distribucija opterećenja pisanja (Prema: Suzuki i sur., 2012)

5. Primjena AWS-a

U sljedećem dijelu rada biti će prikazano kako primjenom AWS CloudFormation predloška zakupiti računalnu infrastrukturu konfiguriranu po uzorku dizajna koji je opisan u poglavlju 4.1.4 Dinamičko skaliranje broja servera s dodatkom redundancije na razini podatkovnog centra (poglavlje 4.2.2). Zatim će biti prikazani rezultati testa opterećenja infrastrukture.

5.1. Opis uzorka dizajna iz CloudFormation predloška

Uzorak dizajna za dinamičko skaliranje broja servera (eng. *Scale-Out*) je jedan od najčešće korištenih uzoraka na AWS-u. Primjenom servisa za ujednačavanje opterećenja promet se ravnomjerno distribuira preko više individualnih instanci virtualnog servera. Svaki od njih ima instaliran i konfiguriran Apache HTTP server sa zadanom web stranicom. Inicijalne postavke servera definirane su u CloudFormation predlošku i u AMI-u iz kojih Auto Scaling pokreće nove instance kada se CloudWatch alarm aktivira. Nove instance pokreću se u nekoliko definiranih raspoloživih zona (AZ) čime se uz efikasnost automatizacijom skaliranja broja virtualnih servera postiže i jako visoka razina raspoloživosti sustava. U odnosu na definirana pravila alarma u CloudWatch servisu broj instanci se prema potrebi može povećavati ili smanjivati. Uz navedeno, sustav prepoznaje ako na nekoj instanci dođe do greške i ona prestani biti dostupna, automatski se zamjenjuje novom instancom. Taj mehanizam je kombinacija prethodno spomenutih uzoraka dizajna u poglavljima 4.2.4 Zamjena servera i 4.2.5 Duboka provjera stanja servera.



Slika 24: Konfiguracija uzorka dizajna - Dinamičko skaliranje broja servera s redundancijom na razini podatkovnog centra.

5.2. Opis CloudFormation predloška

Nakon preuzimanja predloška pod imenom *Auto Scaling and load-balancing website in an Amazon VPC* sa službene web stranice (Amazon, 2017) može ga se otvoriti u AWS CloudFormation Designer servisu. Tamo dobijemo vizualni prikaz predloška (AWS CloudFormation Designer - Slika 25) i moguće je na njemu raditi proizvoljne izmjene. Predložak se sastoji od: resursa, parametara, mapiranja, uvijeta, meta podataka i izlaznih vrijednosti.

Navedeni predložak sadrži sljedeće resurse:

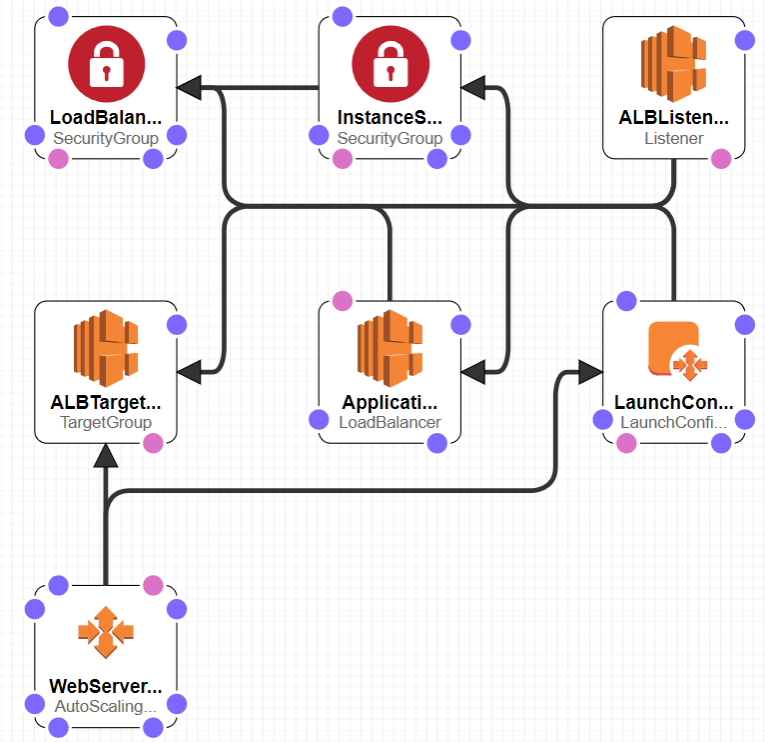
- AWS::AutoScaling::AutoScalingGroup - Definicija grupe koja se skalira.
- AWS::AutoScaling::LaunchConfiguration - Konfiguracija za lansiranje instanci, AMI i slično.
- AWS::EC2::SecurityGroup - Definicija sigurnosnih pravila za EC2 i servis za ujednačavanje opterećenja poput propuštanja prometa za portove 80 (HTTP) i 22 (SSH).

- AWS::ElasticLoadBalancingV2::LoadBalancer - Konfiguracija servisa za ujednačavanje opterećenja.
- AWS::ElasticLoadBalancingV2::TargetGroup - Definicija grupe na servisu za ujednačavanje opterećenja. Ovdje je također definirana provjera stanja instanci (eng. *Health check*).
- AWS::ElasticLoadBalancingV2::Listener - Konfiguracija porta i protokola koji se koriste za dohvaćanje web stranice.

Parametre za pokretanje predloška moguće je postaviti u samom predlošku ili kroz web sučelje CloudFormation servisa prilikom pokretanja. U navedenom predlošku postoji veći broj parametara koji imaju definirane zadane vrijednosti ali sljedeće je nužno definirati prilikom pokretanja:

- VpcId - Identifikacijski kod od validnog VPC-a.
- Subnets - Popis podmreža u kojima će se instance nalaziti. Ovdje definiramo podmreže tako da je svaka iz druge raspoložive zone.
- KeyName - Ime validnog EC2 privatnog ključa koji će se koristiti za spajanje na instance putem SSH protokola.
- SSHLocation - Opcionalno se definira IP adresa s koje će se administrator spajati a svim ostalim se zabrani pristup (eng. *Whitelisting*).
- InstanceType - Odabir validnog tipa instance koji će se pokretati (npr. t2.micro).
- InstanceCount - Broj instanci koji će biti minimalno aktivan.

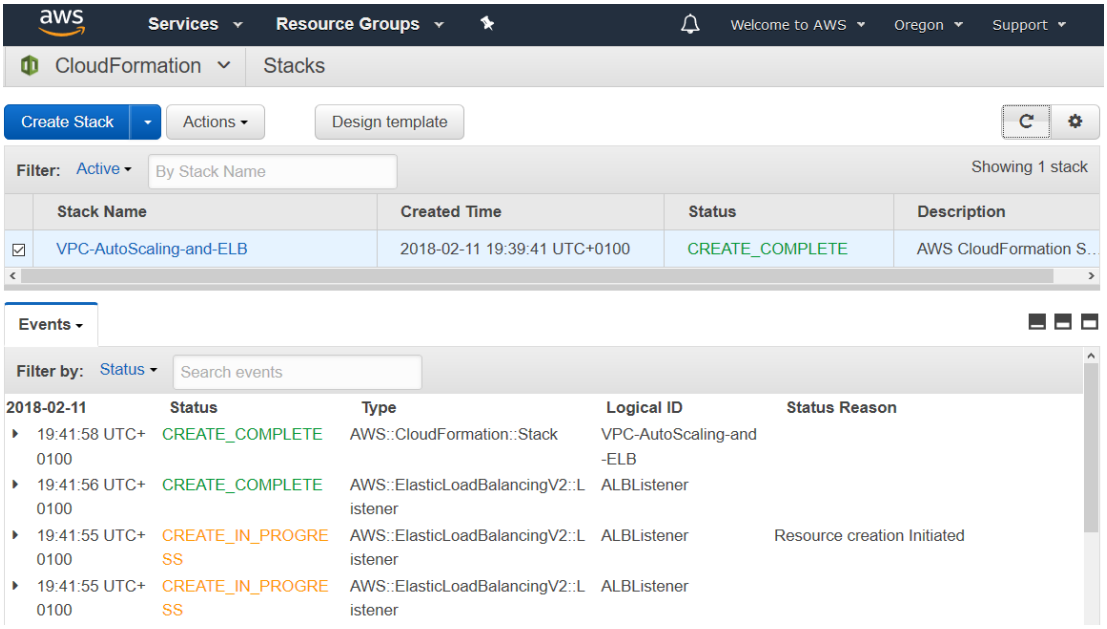
Mapiranja se odnose na vrijednosti koje sadrže ime i definirane vrijednosti. Na primjer tip instance može biti samo jedna od postojećih vrijednosti; t1.micro, t2.nano itd. Nije moguće unositi proizvoljne vrijednosti. Uvjeti služe za kontrolu procesa pokretanja računalnih resursa. Ponekad neki resurs ovisi o nekom drugom pa se postavi uvjet da se potrebni resurs pokrene prije onog koji o njemu zavisi. Meta podaci se odnose na dodatne informacije o predlošku, dok se izlazne vrijednosti prikazuju svaki puta kada želimo pogledati status predloška. U ovom primjeru izlazna vrijednost je URL adresa koja je *endpoint* i vodi k usmjerivaču opterećenja.



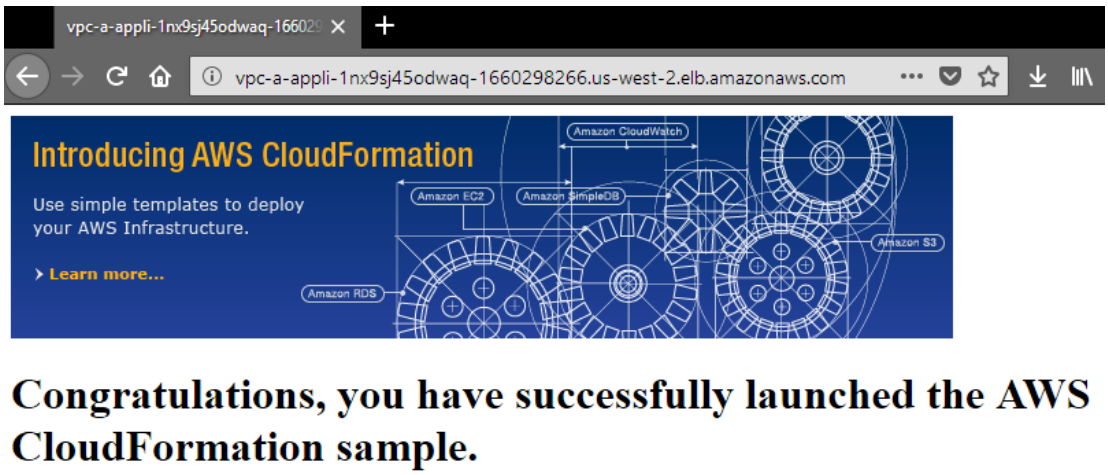
Slika 25: AWS CloudFormation Designer - Autoscaling and load-balancing website in an Amazon VPC (Amazon, 2017)

5.3. Inicijalizacija infrastrukture primjenom CloudFormation predloška

Za korištenje AWS CloudFormation predložaka potrebno je imati validan AWS korisnički račun. Nakon uspješne autentifikacije na AWS *Management Console* (*https://aws.amazon.com/console/)* potrebno je odabrati u kojoj raspoloživoj zoni želimo pokrenuti predložak. Valja uzeti u obzir da su neki predlošci predefinirani za određeno područje pa ga je potrebno modificirati ukoliko ga želimo pokrenuti u nekoj drugoj raspoloživoj zoni. Slijedeći korak je primjenom pretraživača pronaći CloudFormation servis, odabrati opciju kreiranja novog stoga (eng. *Create new stack*), odabrati predložak i kroz sučelje definirati potrebne parametre. Kroz nekoliko minuta nakon pokretanja trebali bi imati uspješno kreiranu infrastrukturu (Slika 26). Preostalo je još testirati dali je ELB *endpoint* dostupan unošenjem njegove URL adrese u adresnu traku preglednika (Slika 27). Web stranica koje se prikazuje je vrlo "lagana" tj. sadrži samo jednu sliku i jedan naslov. Adresu *endpointa* možemo pronaći pod EC2 servisom u *Load balancers* grupi pod DNS *name*.



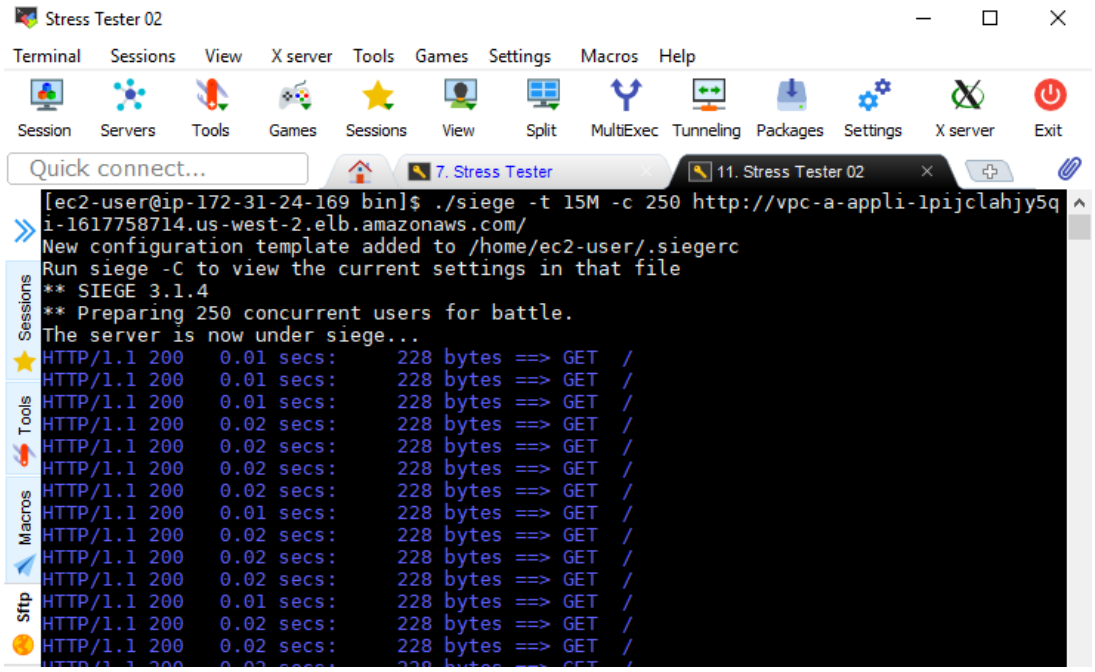
Slika 26: AWS CloudFormation - Uspješno kreirana infrastruktura primjenom predloška.



Slika 27: AWS CloudFormation - zadana web stranica.

5.4. Testiranje opterećenja

Testiranje zakupljene infrastrukture kreirane po uzorku dizajna Dinamičko skaliranje broja servera s dodatkom redundancije na razini podatkovnog centra izvršeno je uz pomoć dodatnih EC2 instanci na koje je instaliran HTTP/HTTPS stres tester program za simuliranje opterećenja Siege (Fulmer, 2012). Slika 28 prikazuje pokretanje testa u programu za SSH konekciju MobaXterm (Mobatek, 2017).



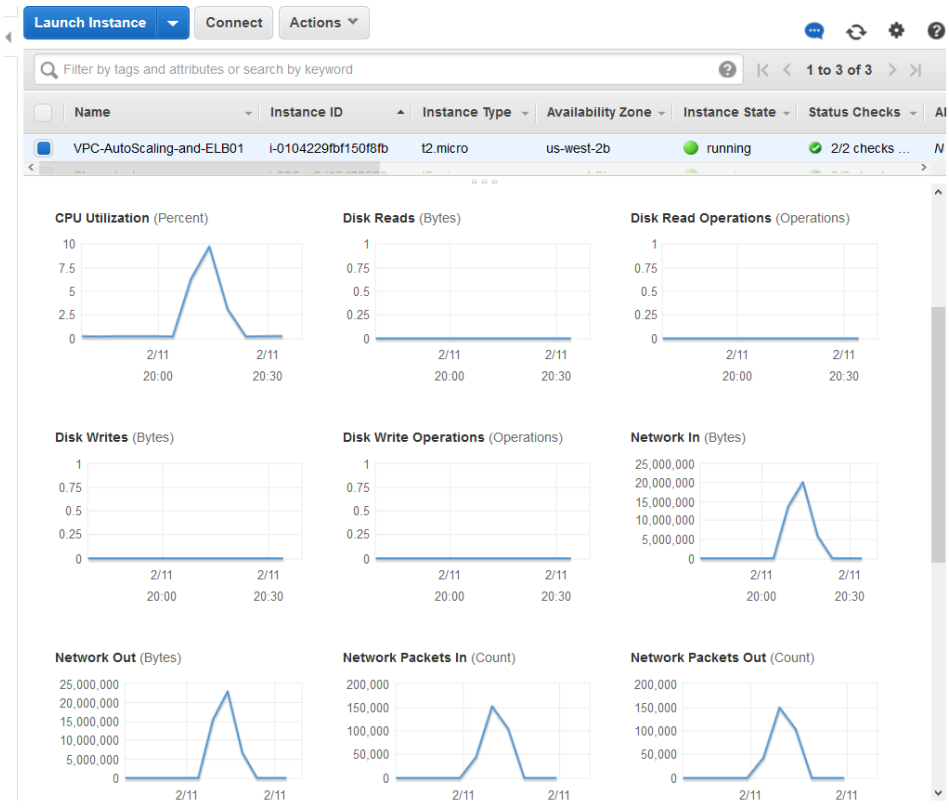
Slika 28: Primjena korištenja Siege programa za simuliranje opterećenja web stranica

Test opterećenja izvršavan je 15 minuta i simulirao je 250 istovremenih korisnika s dvije instance stres testera koje su se nalazile u istoj AWS raspoloživoj zoni kao i korišteni sustav. Razlog za dvije instance su fizička ograničenja instanci i programa za testiranje. Kada test opterećenja završi prikažu se statističke informacije o testu. Tabela 4 prikazuje rezultate inicijalnog testa nad samo jednom EC2 instancom web servera prije postavljanja *Auto Scaling* pravila. Možemo primijetiti da je opterećenje bilo dovoljno veliko da je nastalo ukupno 471 neuspješnih konekcija prema web stranici. Valja uzeti u obzir da je web stranica vrlo jednostavna što je omogućilo vrlo dobru raspoloživost. U realnom slučaju web stranica ima puno više sadržaja što utječe na vrijeme prijenosa podataka, učitavanje stranice i ostale performanse. U slučaju da se radi o web aplikaciji, ili pak da se korisnici spajaju iz cijelog svijeta, utjecaj bi bio znatno veći.

	Stres tester 1	Stres tester 2	
Transactions	388013	381605	hits
Availability	99.95	99.92	%
Elapsed time	899.45	899.22	sec
Data transferred	84.39	83.02	MB
Response time	0.08	0.09	sec
Transaction rate	431.39	424.37	trans/sec
Throughput	0.09	0.09	MB/sec
Concurrency	34.20	36.10	
Successful transactions	388013	381605	
Failed transactions	178	293	
Longest transaction	10.01	11.00	sec
Shortest transaction	0.00	0.00	sec

Tabela 4: Rezultati inicijalnog testa opterećenja nad jednom EC2 instancom

Slika 29 prikazuje metrike iz AWS EC2 konzole za inicijalni test opterećenja. Iskorištenost procesora popela se na 10%, porasla je mrežna ulazna i izlazna propusnost, te broj ulaznih i izlaznih mrežnih paketa. Ovo su samo neko od metrika koje AWS CloudWatch skuplja i prikazuje. U osnovnom besplatnom paketu skupljanja metrika na AWS-u interval osvježavanja prikaza je pet minuta. Ukoliko to ne zadovoljava potrebe moguće je koristiti opciju za detaljniji nadzor koja nudi interval osvježavanja od jedne minute po cijeni od 0.50 USD mjesečno po pojedinoj odabranoj metrici.



Slika 29: AWS CloudWatch metrike EC2 instance nakon inicijalnog testa opterećenja

5.5. Testiranje opterećenja s primjenom Auto Scaling servisa

Kako bi iskoristili zakupljenu infrastrukturu koja koristi navedeni uzorak dizajna za visoku raspoloživost, preostalo nam je definirati politike skaliranja *Auto Scaling* servisa. Politike ćemo pronaći pod *Auto Scaling Groups,* nakon odabira kreirane grupe u kartici *Scaling Policies*. Postoje tri tipa politika skaliranja:

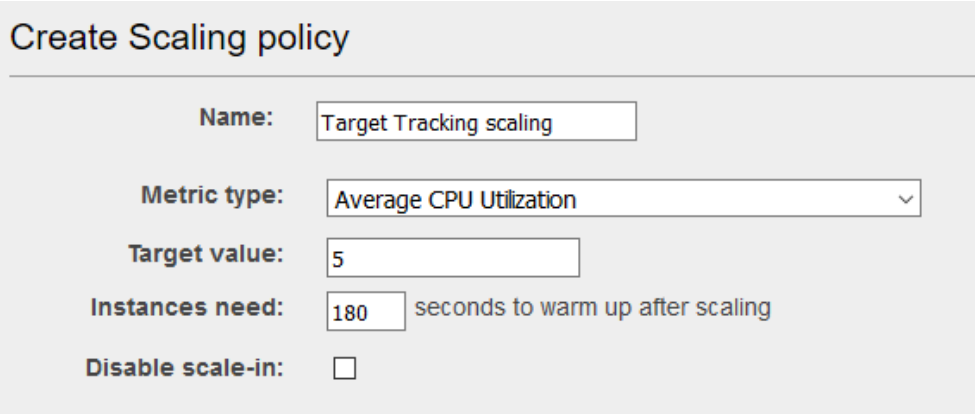
- *Simple scaling policy* - Jednostavna politika skaliranja koja u odnosu na odabranu metriku povećava ili smanjuje kapacitete definirane grupe.
- *Scaling policy with steps* - Naprednija politika skaliranja u kojoj možemo definirati za koliko će se kapaciteti povećati ili smanjiti u odnosu na iznos odabrane metrike.
- *Target tracking scaling policy* - Politika skaliranja u kojoj definiramo iznos metrike koji želimo da bude konzistentan, a kapaciteti se povećavaju ili smanjuju unutar definiranih granica kako bi zadovoljili postavljeni iznos.

AWS CloudWatch omogućava nadgledanje velikog broja metrika EC2 instanci. Neke od njih su:

- *CPU Utilization (%)* - Postotak iskorištenosti procesora.
- *Memory Utilization (%)* - Postotak iskorištenosti radne memorije.
- *Network Out Utilization (MB)* - Iskorištenost mreže u MB.
- *Memory Used (MB)* - Iskorištenost radne memorije u MB.
- *Memory Available (MB)* - Dostupnost radne memorije u MB.
- *Swap Utilization (%)* - Postotak iskorištenosti *Swap* memorije.
- *Swap Used (MB)* - Iskorištenost *Swap* memorije u MB.
- *Disk Space Utilization (%)* - Postotak iskorištenosti čvrstog diska.
- *Disk Space Used (GB)* - Iskorištenost čvrstog diska u GB.
- *Disk Space Available (GB)* - Dostupnost čvrstog diska u GB
- Mnoge druge metrike (Amazon CloudWatch Metrics and Dimensions Reference, 2017)

Za potrebe testiranja konfiguriran je *Target tracking scaling policy*. Kao promatrana metrika odabrana je prosječna iskorištenost procesora i kao ciljanu vrijednost odabrano je 5% iskorištenosti. U realnom scenariju ta vrijednost bi se postavljala na raspon između 70-90% iskorištenosti. Najčešće korištene metrike općenito su iskorištenost procesora, radne memorije i mrežne propusnosti. Slika 30 prikazuje definiranu politiku skaliranja. Preostala opcija koja

nas zanima je broj sekundi koliko treba pokrenutoj instanci da postane aktivna. Ta vrijednost pomaže kod procjene koliko novih instanci treba podignuti.



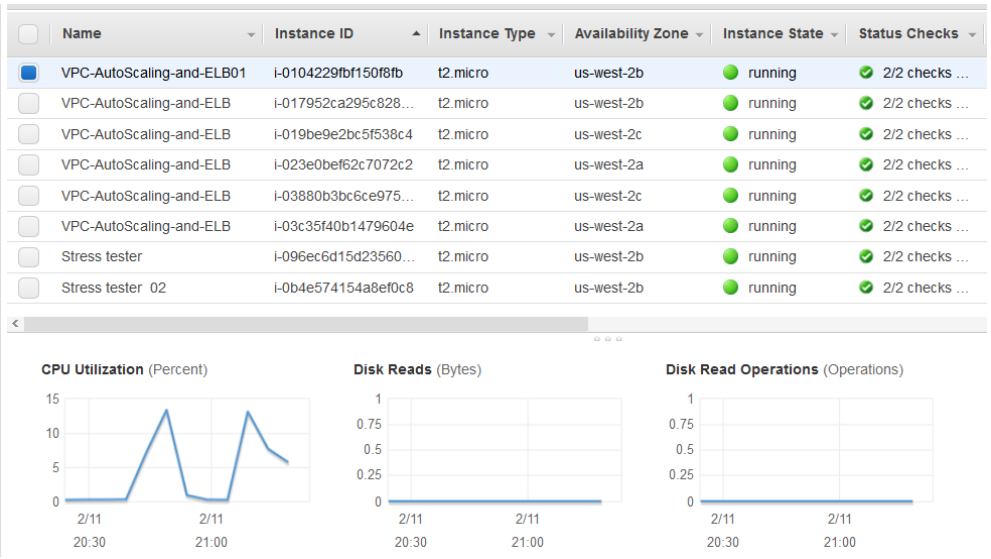
Slika 30: Konfiguracija politike Auto Scaling servisa

Testiranje Auto Scaling servisa izvršeno je primjenom dvije EC2 instance s kojih je istovremeno pokrenut program *Siege* da 15 minuta simulira po 250 korisnika. Tabela 5 prikazuje statistike nakon završetka testa opterećenja. Ako usporedimo rezultate s prethodnim rezultatima testiranja (Tabela 4) možemo primijetiti 100% raspoloživost sustava. Na prvom stres testeru došlo je do najduže transakcije od 15 sekundi ali nemamo informaciju o broju takvih transakcija pa je za taj aspekt potrebno daljnje istražiti i testirati s drugim alatima.

	Stres tester 1	Stres tester 2	
Transactions	440703	447242	hits
Availability	100.00	100.00	%
Elapsed time	899.95	899.84	sec
Data transferred	95.83	97.25	MB
Response time	0.00	0.00	sec
Transaction rate	489.70	497.02	trans/sec
Throughput	0.11	0.11	MB/sec
Concurrency	1.60	1.34	
Successful transactions	440703	447242	
Failed transactions	0	0	
Longest transaction	15.26	1.03	sec
Shortest transaction	0.00	0.00	sec

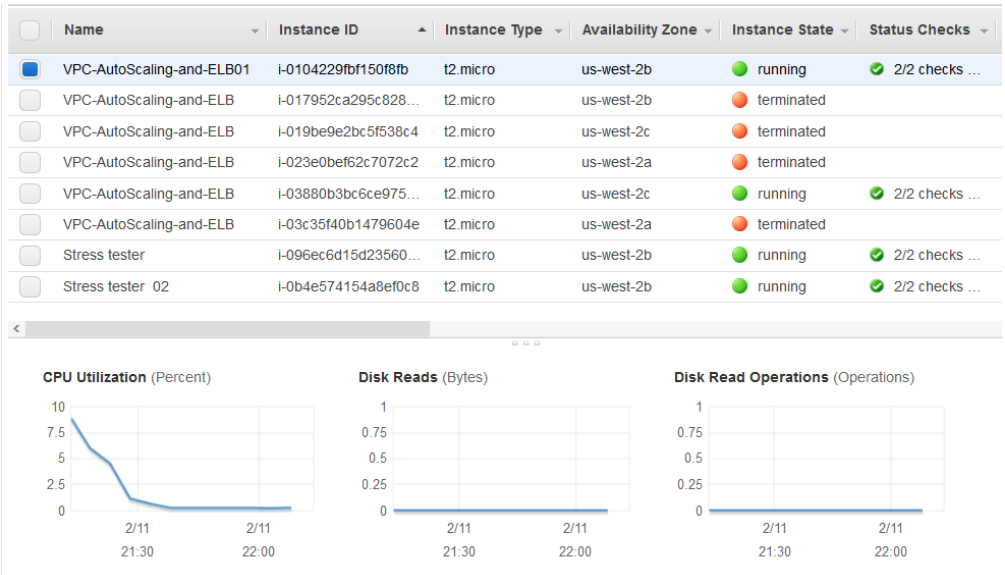
Tabela 5: Rezultati testa opterećenja nad Auto Scaling grupom instanci

Slika 31 prikazuje popis EC2 instanci par minuta nakon pokretanja stres testa Auto Scaling grupe. Kako je u grupi definirano da minimalno bude jedna instanca a maksimalno šest, tako na slici vidimo pet novo pokrenutih instanci. Inicijalna instanca je prva na listi s definiranim imenom VPC-AutoScaling-and-ELB01, a ostale imaju prethodno zadanu vrijednost imena. Iz CloudWatch vizualizacije vidljivo je uspješno spuštanje postotka iskorištenosti procesora na ciljanih 5%.



Slika 31 AWS CloudWatch metrike za vrijeme testa opterećenja Auto Scaling grupe

Slika 32 prikazuje nam popis EC2 instanci par minuta nakon završetka stres testa Auto Scaling grupe. CloudWatch graf prikazuje nam potpuni prestanak opterećenja procesora, a tablica u stupcu *Instance State* stanje prethodno pokrenutih instanci gdje *terminated* predstavlja ugašenu instancu. Kao što vidimo *Auto Scaling* funkcionira kako su politike i konfigurirane.



Slika 32 AWS CloudWatch metrike nakon testa opterećenja Auto Scaling grupe

6. Zaključak

U tradicionalnom fizičkom podatkovnom centru potrebno je vršiti aktivnosti poput naručivanja hardvera, sastavljanja, instalacije, konfiguriranja, skladištenja, servisiranja, održavanja, postavljanja sigurnosnih mjera i slično. Poduzeća već dugo traže način kako da te aktivnosti povjere drugom pružatelju usluge. Dvije ključne tehnologije koje su to omogućile su Internet i virtualizacija, objedinjene pod pojmom računarstvo u oblaku. Ono zauvijek mijenja način na koji organizacije koriste infrastrukturu i omogućava im da se fokusiraju na ključne kompetencije umjesto na upravljanje IT infrastrukturom. Klasične organizacije napokon prevladavaju strah od računarstva u oblaku i kako raste povjerenje u sigurnost oblaka tako razmjerno raste i prihvaćanje tržišta.

Računarstvo u oblaku rapidno se razvija i otključava mogućnost inovacija u svim aspektima IT-a bez tradicionalnih ograničenja. Model plaćanja po korištenju zamjenjuje velike inicijalne troškove nabave infrastrukture. Uz to podiže se razina agilnosti u organizaciji i brzina dolaska na globalno tržište. Takav pristup pogotovo odgovara manjim i modernim *Start-Up* organizacijama koje s minimalnim ulaganjima mogu testirati i inovirati primjenom infrastrukture u oblaku. Veliku priliku takvim organizacijama stvaraju zastarjela softverska rješenja ili neka koja prije nisu bila moguća za izvedbu i rekreiranje tih rješenja kao softverskih usluga primjenom novih tehnologija.

Amazon AWS je lider na tržištu računarstva u oblaku. Tome je doprinio njihov strateški cilj koji nije standardna akumulacija profita, već dominacija na tržištu pa je većina generiranog profita uložena u unaprjeđenje postojećih i istraživanje i razvoj novih usluga oblaka. Dodatne karakteristike koje AWS ima su brojne: sigurnost, automatizacija, dinamičko skaliranje i efikasnost iskorištavanja IT kapaciteta samo su neke. Primjenom AWS-a nije potrebno raditi predikciju IT kapaciteta jer je lako dodati ili ukloniti računalne resurse. AWS kao javni oblak također ima koristi od ekonomije razmjera, što znači da usluge postaju jeftinije što više imaju korisnika.

Naglasak ovog rada bio je pokazati kako primjenom uzoraka dizajna arhitekt oblaka može riješiti različite probleme i utjecati na razne performanse sustava u oblaku, sve po najboljim praksama. Kako je raspoloživost bila tema ovog rada tako je u praktičnom djelu testirana AWS infrastruktura kreirana po uzorku dizajna za dinamičko skaliranje broja servera s dodatkom redundancije na razini podatkovnog centra. Iz rezultata testa može se vidjeti proces skaliranja broja servera u odnosu na opterećenje, te poboljšanu raspoloživost i performanse primjenom uzorka dizajna.

Osobno smatram da je računarstvo u oblaku tehnologija koja donosi revoluciju tradicionalnih informacijskih sustava. Amazon AWS prednjači u tom zbog truda koji ulaže u razvoj i inovacije. Činjenica da se njihova infrastruktura može zakupljivati i upravljati primjenom računalnog kôda stvara potpuno novu paradigmu razvoja informacijskih sustava. Što se općenito računalnog oblaka tiče, smatram da će uspjeti one organizacije koje će shvaćati razlike modela pružanja usluga i modela implementacije, te koje će znati iste mapirati na poslovne zahtjeve koje imaju. Također, morat će se znati nositi s otporom na promjene, nedostatkom znanja o oblaku, novim poslovnim procesima i ostalim elementima organizacije kojima su najmanji zajednički nazivnik ljudi, procesi i tehnologija.

Popis literature

[1] Amazon. (7.12.2017). *Amazon Aurora*. Preuzeto 17.12.2017 iz Amazon AWS: https://aws.amazon.com/rds/aurora/

[2] Amazon. (29.12.2017). *Amazon Auto Scailing*. Preuzeto 7.1.2018 iz Amazon AWS: https://aws.amazon.com/autoscaling/

[3] Amazon. (31.10.2017). *Amazon CloudFront now has 100 Points of Presence with the launch of its fifth Edge Location in Tokyo, Japan*. Preuzeto 10.12.2017 iz Amazon AWS: https://aws.amazon.com/about-aws/whats-new/2017/10/cloudfront-now-has-100-points-of-presence/

[4] Amazon. (29.12.2017). *Amazon CloudWatch*. Preuzeto 7.1.2017 iz Amazon AWS: https://aws.amazon.com/cloudwatch/

[5] Amazon. (15.6.2017*). Amazon CloudWatch Metrics and Dimensions Reference*. Preuzeto 4.3.2018 iz Amazon AWS: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CW_Support_For_AWS.html#ec2-metricscollected

[6] Amazon. (7.12.2017*). Amazon DynamoDB*. Preuzeto 17.12.2017 iz Amazon AWS: https://aws.amazon.com/dynamodb/

[7] Amazon. (7.12.2017). *Amazon EBS*. Preuzeto 17.12.2017 iz Amazon AWS: https://aws.amazon.com/ebs/

[8] Amazon. (7.12.2017). *Amazon EC2*. Preuzeto 17.12.2017 iz Amazon AWS: https://aws.amazon.com/ec2/

[9] Amazon. (12.29.2017). *Amazon Elastic Load Balancing Products*. Preuzeto 7.1.2018 iz Amazon AWS: https://aws.amazon.com/elasticloadbalancing/

[10] Amazon. (15.6.2017). *Amazon Machine Images (AMI).* Preuzeto 17.12.2017 iz Amazon AWS: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

[11] Amazon. (7.12.2017). *Amazon Relational Database Service (RDS).* Preuzeto 17.12.2017 iz Amazon AWS: https://aws.amazon.com/rds/

[12] Amazon. (7.12.2017). *Amazon S3.* Preuzeto 17.12.2017 iz Amazon AWS: https://aws.amazon.com/s3/

[13] Amazon. (29.12.2017). *Amazon VPC.* Preuzeto 7.1.2018 iz Amazon AWS: https://aws.amazon.com/vpc/

[14] Amazon. (5 2017). *Amazon Web Services: Risk and Compliance Whitepaper.* Preuzeto 10.12.2017 iz Amazon AWS: http://d0.awsstatic.com/whitepapers/compliance/AWS_Risk_and_Compliance_Whitepaper.pdf

[15] Amazon. (7.12.2017). *AWS Global Infrastructure*. Preuzeto 10.12.2017 iz Amazon AWS: https://aws.amazon.com/about-aws/global-infrastructure/

[16] Amazon. (29.12.2017). *AWS Identity and Access Management (IAM).* Preuzeto 7.1.2018 iz Amazon AWS: https://aws.amazon.com/iam/

[17] Amazon. (7.12.2017). *AWS Technical Essentials - Module 2: AWS Infrastructure - Amazon EBS and Amazon S3*. Preuzeto 17.12.2017 iz Amazon AWS Training: https://content.aws.training/wbt/aws-technical-essentials/en

[18] Amazon. (7.12.2017). *AWS Technical Essentials - Module 4: Amazon Database services.* Preuzeto 17.12.2017 iz Amazon AWS Training: https://content.aws.training/wbt/aws-technical-essentials/en

[19] Amazon. (29.12.2017). *AWS Trusted Advisor*. Preuzeto 7.1.2018 iz Amazon AWS: https://aws.amazon.com/premiumsupport/trustedadvisor/

[20] Amazon. (4 2017). *Overview of Amazon Web Services*. Preuzeto 10.12.2017 iz Amazon AWS: https://d0.awsstatic.com/whitepapers/aws-overview.pdf

[21] Amazon. (15.6.2017*). Sample CloudFormation Templates*. Preuzeto 3.3.2018 iz Amazon AWS: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/sample-templates-services-us-west-2.html

[22] Amazon. (7.12.2017). *Shared Responsibility Model.* Preuzeto 10.12.2017 iz Amazon AWS: https://aws.amazon.com/compliance/shared-responsibility-model/

[23] Fehling, L. R. (2014). *Cloud Computing Patterns - Fundamentals to Design, Build, and Manage Cloud Applications*. Wien: Springer-Verlag.

[24] Fulmer. (30.1.2012). *Siege*. Preuzeto 4.3.2018 iz Joedog: https://www.joedog.org/siege-home/

[25] G2 Crowd. (1.10.2017). *Best Cloud Platform as a Service (PaaS) Software*. Preuzeto 1.10.2017 iz G2 Crowd: https://www.g2crowd.com/categories/cloud-platform-as-a-service-paas

[26] G2 Crowd. (1.10.2017). *Best Infrastructure as a Service (IaaS) Providers*. Preuzeto 1.10.2017 iz G2 Crowd: https://www.g2crowd.com/categories/infrastructure-as-a-service-iaas

[27] Gartner. (15.6.2017). *Magic Quadrant for Cloud Infrastructure as a Service, Worldwide*. Preuzeto 9.12.2017 iz Gartner: https://www.gartner.com/doc/reprints?ct=150519&id=1-2G2O5FC&st=sb

[28] Kavis, M. J. (2014). *Architecting the cloud - Design Decisions for Cloud Computing Service Models*. New Jersey: Wiley.

[29] Kundra, V. (1.7.2010). *Vivek Kundra Testimony on Cloud Computing*. Preuzeto 28.8.2017 iz cio.gov: https://www.cio.gov/2010/07/01/vivek-kundra-testimony-on-cloud-computing/

[30] Mobatek. (2017). *MobaXterm*. Preuzeto 4.3.2018 iz Mobatek: https://mobaxterm.mobatek.net/

[31] National Institute of Standards and Technology [NIST]. (9 2011). *NIST Special Publications*. Preuzeto 31.8.2017 iz NIST Computer Security Resource Center: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[32] Statista. (1.10.2017). *Quarterly revenue of Amazon Web Services from 1st quarter 2014 to 3rd quarter 2017 (in million U.S. dollars).* Preuzeto 9.12.2017 iz Statista: https://www.statista.com/statistics/250520/forecast-of-amazon-web-services-revenue/

[33] Suzuki, K. T. (15.11.2013). *Cloud Design Pattern*. Preuzeto 12.11.2017 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php

[34] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:DB Replication Pattern.* Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:DB_Replication_Pattern

[35] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Deep Health Check Pattern.* Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Deep_Health_Check_Pattern

[36] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012*). CDP:Floating IP Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Floating_IP_Pattern

[37] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Inmemory DB Cache Pattern.* Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Inmemory_DB_Cache_Pattern

[38] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Multi-Datacenter Pattern.* Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Multi-Datacenter_Pattern

[39] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Multi-Server Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Multi-Server_Pattern

[40] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Ondemand Disk Pattern.* Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Ondemand_Disk_Pattern

[41] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012*). CDP:Read Replica Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Read_Replica_Pattern

[42] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Scale Out Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Scale_Out_Pattern

[43] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Scale Up Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Scale_Up_Pattern

[44] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Server Swapping Pattern.* Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Server_Swapping_Pattern

[45] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Sharding Write Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Sharding_Write_Pattern

[46] Suzuki, K., Tamagawa, A., & Katayam, H. (20.12.2012). *CDP:Snapshot Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Snapshot_Pattern

[47] Suzuki, K., Tamagawa, A., & Katayam, H. (28.11.2012). *CDP:Stamp Pattern*. Preuzeto 21.1.2018 iz Cloud Design Pattern: http://en.clouddesignpattern.org/index.php/CDP:Stamp_Pattern

[48] The Open Group. (2010). *Downloads Archive*. Preuzeto 4.9.2017 iz Cloud Security Alliance: https://downloads.cloudsecurityalliance.org/initiatives/guidance/Open-Group-Cloud-Computing-Book.pdf

[49] Tribe, J. (16.8.2016*). The Big -aaS List of As-a-Service Offerings*. Preuzeto 14.10.2017 iz auvik.com: https://www.auvik.com/media/blog/aas-as-a-service-list/

[50] Varia, J. (1 2011). *Architecting for the Cloud: Best Practices*. Preuzeto 10.12.2017 iz Amazon Web Services: https://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf

Popis slika

Popis tablica