Predlog jednog pristupa primeni Amazon veb servisa u razvoju aplikacije

Đorđe Radivojević, dr Dušan Vujošević

***Sadržaj* —Ovaj rad se bavi implementacijom Amazon veb servisa (AWS), kao najpopularnijeg servisa računarstva u oblaku sa ciljem konstruisanja portala koji služi pisanju recenzija i ocenjivanju od strane korisnika. U radu se kao primer koriste recenzije za grafičke novele. Za odabrane servise se objašnjava za šta se koriste. Predstavlja se kako se inicijalizuju i konfigurišu. Upoređuju se, sa tehničkog i finansijskog aspekta, koncept računarstva u oblaku i koncept posedovanja uređaja i mrežne infrastrukture. Obrazlažu se prednosti i mane primene Amazon veb servis platforme u projektu i daje se osvrt na servise konkurentskih kompanija.**

***Ključne reči* — Amazon veb servis, računarstvo u oblaku, recenzije.**

I. UVOD

MOTIVACIJA za odabir ove teme je proizašla iz želje da se programerima, i ljudima u struci koji imaju iskustva za konvencionalnim sistemima posedovanja hardvera predoči potreban trud i veštine koje su potrebne kako bi se neka aplikacija konstruisala za rad na nekom od klaud rešenja. U ovom slučaju je kao primer uzet Amazon AWS kao najpopularniji pružalac klaud rešenja u trenutku pisanja rada. Sva klaud rešenja iziskuju da se aplikacija prilagodi za rad na njima, kako bi se iskoristile sve mogućnosti kloud platforme, i kako bi se značajnije smanjili troškovi. Tako da je jedan od motiva za pisanje ovog rada da čitalac stekne uvid u kompleksnost razvoja aplikacije ili migracije iste u klaud, kako se određeni izazovi prevazilaze i kojim sredstvima. Takođe i da bude polazna tačka svakome ko poželi da napravi novu aplikaciju ili prototip skalabilne

Đorđe M. Radivojević, bul. Mihaila Pupina 159/14, Beograd, Srbija; (telefon: 0652132014, email: djradivojevic@gmail.com).

dr Dušan Vujošević. Računarski fakultet, Beograd, Srbija (email: dvujosevic@raf.rs).

aplikacije na klaud platformi.

Doprinos ovog rada bi trebalo da bude u tome da se na jednom mestu obuhvate mogućnosti računarstva u oblaku, pojedinačnih servisa, i na primeru aplikacije njihova implementacija i praktična primena.

Godine 2006, Amazon veb servisi (AWS) su počeli da nude usluge IT infrastrukture kompanijama u formi veb servisa, danas poznatih kao računarstvo u oblaku (eng. *Cloud computing*). Najveća prednost računarstva u oblaku je mogućnost da se velika početna ulaganja u infrastrukturu zamene manjim troškovima koji su skalirani poslovnim i infrastrukturnim potrebama. Sa pojavom klauda, kompanije više ne moraju da planiraju servere i IT infrastrukturu, njihovu implementaciju, kao i proširenje kapaciteta, nedeljama ili mesecima unapred. Sistemi za koje se očekuje da vremenom rastu, moraju da se konstruišu nad skalabilnom arhitekturom. Takve arhitekture mogu da podrže rast u broju korisnika sistema, saobraćaju, količini informacija bez opadanja performansi sistema [1].

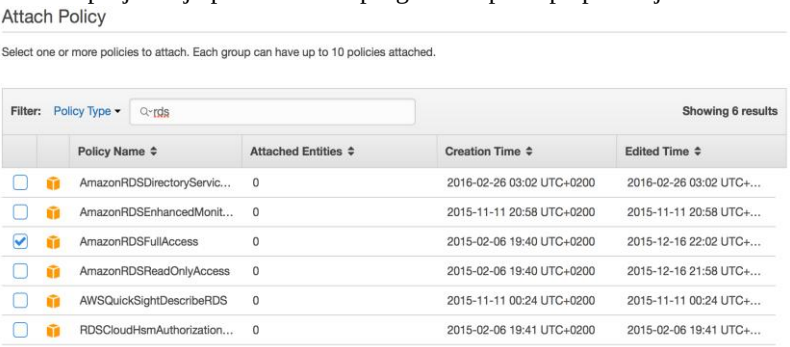
Većina servisa korišćenih u radu su pokriveni Amazonovim *Free Tier*-om, odnosno spadaju u kategoriju servisa koji su besplatni u prvoj godini korišćenja. U sklopu samih servisa se određeni kapaciteti različito naplaćuju, ali je u radu pokazano da je moguće napraviti projekat koji može da zadovolji određeni broj korisnika, dovoljan da se ceo sistem postavi i testira testira.

II. AMAZON IAM

IAM (*Identity and Access Management*) je veb servis koji olakšava da se na siguran i pouzdan način kontroliše pristup AWS resursima od strane korisnika AWS-a, odnosno programera i *DevOps*-a, kao i aplikacija. Na taj način se određuje način autentifikovanja i korisnika i aplikacija. Odnosno koji kredencijali (*credentials)* imaju pravo pristupa kojim resursima i koje privilegije i prava imaju nad tim resursima. Tako je moguće dati pristup npr. programerima da koriste samo Git hostovan na CodeCommit servisu, da rade i *push* i *pull* komande, ali da nemaju pravo da rade *merge* na *master* granu. Ili je moguće dati programski pristup bazi podataka na RDS-u, pri kojem se formiraju ID pristupnog ključa (*access key ID)* i tajni pristupni ključ (*secret access key) za* AWS API, CLI, SDK i ostale alate za razvoj softvera, ali bez prava upisa ili izmene podataka, već samo čitanja iz baze. Pristup, odnosno polise prava pristupa se određuju ili na nivou grupe ili im se direktno dodeljuju polise. Tako je moguće i da svaki korisnik, bilo aplikacija ili

inženjer pripadaju jednoj ili više grupa sa svojim skupovima polisa, i da im se dodatno dodeli polisa specifično za njih. Snažno se obeshrabruje da svi korisnici imaju potpuni pristup i sva prava u okviru sistema, kako se ne bi ugrozila sigurnost ili čak dovelo do slučajnih grešaka.

U ovom projektu je potrebno dati programski pristup aplikaciji da unosi u



Sl. 1. Dodeljivanje polisa prava pristupa grupi.

bazu i čita iz baze podataka na RDS servisu. Prvo je neophodno napraviti grupu i dodeliti joj odgovarajuću polisu. Grupa je nazvana *app-access-group* i dodata joj je polisa *AmazonRDSFullAccess* (Sl. 1).

Nakon toga je potrebno napraviti novog korisnika i dodati ga u novonapravljenu grupu.

I to je sve što je potrebno kako bi se osigurao pristup aplikaciji sa tačno određenim pravom pristupa, nakon čega se dobijaju kredencijali za programski pristup.

Slična je procedura i za kreiranje novih korisnika sa pravom pristupa konzoli za upravljanje (*AWS Management Console Access*), gde je moguće postaviti odgovarajuća imena i lozinke ili odobriti da korisnici sami izaberu, gde se pritom kreira privremena lozinka za svakog korisnika, koju su u obavezi da promene prilikom prvog povezivanja na konzolu.

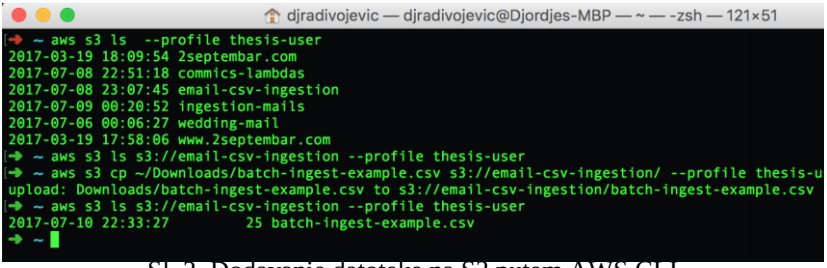
III. Interfejs komandne linije - AWS CLI

Kako bi se sa AWS sistemom mogla vršiti interakcija i iz terminala lokalnog okruženja, i pritom imati sve funkcionalnosti AWS konzole za upravljanje (*Management Console*), potrebno je izvršiti jednostavnu instalaciju i konfiguraciju interfejsa komandne linije, iliti AWS CLI-ja (*Command Line Interface*). AWS CLI je alat otvorenog koda (*open-source*) izgrađen na postojećem AWS SDK za *Python* (Boto). Moguće je koristiti sve uobičajene šel (*shell*) programe za *Linux*, *macOS* ili *Unix*, kao sto su *Bash*,

Zsh, ili *tsch*. Za Windows operativne sisteme se može koristiti *PowerShell* ili *Windows Command Processor*.

Pošto AWS CLI dozvoljava direktan pristup javnim API-ma, moguće je razvijati šel skripte za upravljanje AWS resursima, i na taj način automatizovati određene procese.

Primer korišćenja CLI-a je slanje fajla na S3 iz lokalnog okruženja. Kasnije u radu će biti predstavljen sistem koji na osnovu pojave nove datoteke u S3 koju dostavlja SES servis, izaziva pokretanje Lambda funkcije. Na primeru na slici 2 su demonstrirane komande kojima se može postaviti ta datoteka ručno, čime se izaziva dalja obrada tog fajla, i njegov ulazak u sistem [2]:



Sl. 2. Dodavanje datoteke na S3 putem AWS CLI

Konfiguracioni fajl za CLI se nalazi na putanji ~/.aws/config i sadrži sledeće podatke:

[profile thesis-user]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/EXAMPLEKEY
region = us-west-2

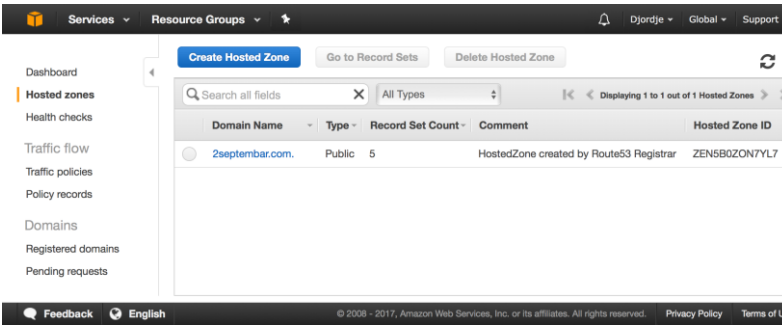
Gde parametri aws_*access*_key_id i aws_*secret*_access_key predstavljaju autentikacione parametre koji se generišu prilikom dodavanja novog korisnika putem IAM konzole, a region predstavlja oznaku AWS regiona u kojem se servis nalazi.

IV. DNS SERVIS - AMAZON ROUTE 53

Route 53 predstavlja Amazonov DNS servis. Pomenuti servis takođe pruža i usluge registracije domena. Mogu se zakupiti domeni i njima upravljati na jednostavan način putem interfejsa, i Route 53 automatski konfiguriše DNS podešavanja za domene nakon kupovine. Route 53 uspešno povezuje zahteve korisnika sa infrastrukturom koja radi na AWS-u, kao što su EC2 instance, ELB balanseri opterećenja, ili S3 buckets. Može se i

konfigurisati da se na osnovu tzv. *health checks* saobraćaj rutira ka zdravim krajnim tačkama (*endpoints*). Takođe i da se nezavisno od ostatka sistema prati zdravlje aplikacije i njenih krajnjih tačaka. Route 53 omogućava i definisanje više različitih destinacija za jedan isti URL (*Uniform Resource Locator*), i protokol po kojem će se rotirati navedene adrese. Editor protoka saobraćaja (*traffic flow visual editor*), predstavlja vizuelni alat za konfiguraciju rutiranja saobraćaja za resurse na AWS-u koristeći postojeće ruting tipove kao što su *failover* i geolokacija.

Za potrebe ovog rada registrovan je domen 2septembar.com (i napravljen je poddomen, odnosno alias www.2septembar.com), kao što se može videti na slici 3.



Sl. 3. Registrovan domen

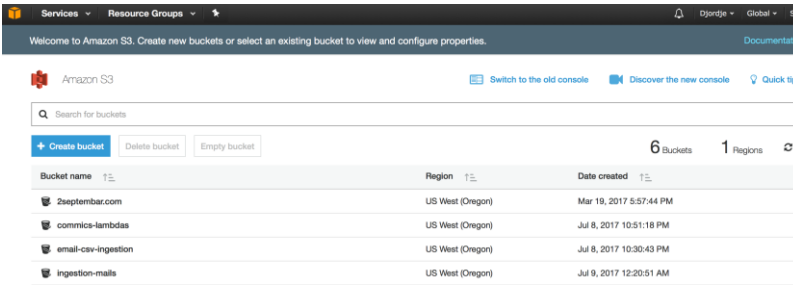
V. SERVIS ZA POHRANJIVANJE PODATAKA - AMAZON S3

Amazon Simple Storage Service, odnosno S3 predstavlja sistem za pohranjivanje objekata, i omogućava pristup bilo kom tipu podataka putem interneta. Korisnici uglavnom koriste S3 kao servis za čuvanje masivnih količina podataka [3]. Podaci se šalju preko internog API-a, konzole ili nekim od dostupnih alata. S3 obezbeđuje neke specifične opcije za upravljanje fajlovima, kao što su brisanje fajlova nakon određenog vremena, arhiviranje „dubljeg nivoa“ (Amazon Glacier) itd. [5]. Takođe, garantuje se dostupnost podataka od 99.99% i pouzdanost od 99.999999999%, budući da za svaki zapis postoji više replika.

U projektu je S3 servis primenu našao u pohranjivanju slika, odnosno profilnih slika korisnika, naslovnica, slika autora, logoe izdavača itd. Kao i za pohranjivanje .csv fajlova za potrebe Lambda funkcija o kojima će kasnije biti reči, a i za različite izveštaje i konfiguracione fajlove. Takođe se čuvaju i *Docker images* desetak poslednjih verzija aplikacije, kako bi se u slučaju

problematične verzije, stanje lako vratilo na poslednju stabilnu verziju. Kreiranje same korpe, odnosno *bucket*-a je vrlo jednostavan proces, potrebno je jedino odabrati ime koje je unikatno u celom AWS sistemu.

Moguće je postaviti da se kao odgovor na određene promene u korpama izazovu pojedine akcije, kao što je izvršavanje lamda funkcije, o čemu će dalje biti reći u poglavlju o Lambda funkcijama.



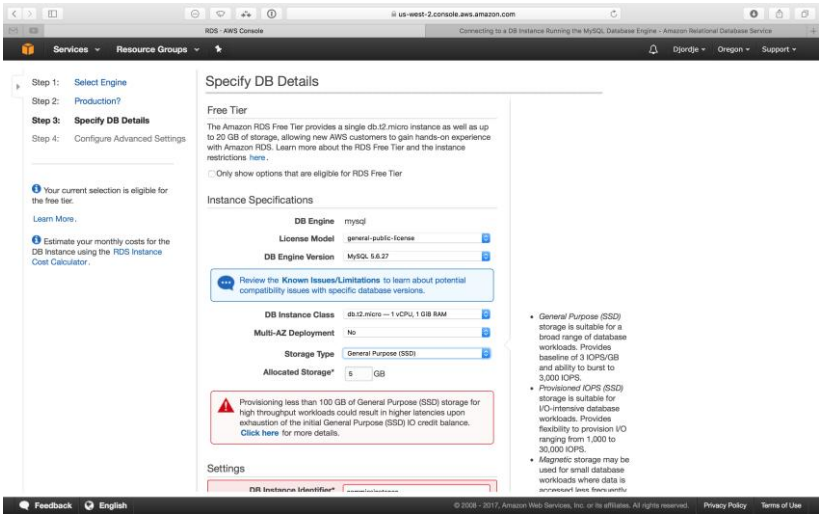
Sl 4. Prikaz potrebnih korpi

VI. SERVIS ZA BAZE PODATAKA – AMAZON RDS

RDS je servis za baze podataka, može biti baziran na nekom od dostupnih database endžina, besplatnih ili onih koji se plaćaju. RDS uklanja potrebu za održavanjem sopstvene baze podataka, obezbeđuje redundantnost po potrebi, bekap na dnevnom nivou i drugo. Značajno je i da testovi potvrđuju da je razlika performansi servisa u default stanju za MySQL servis, koji se i najčešće koristi, u odnosu na optimizovan sopstveni servis minimalna [4][5].

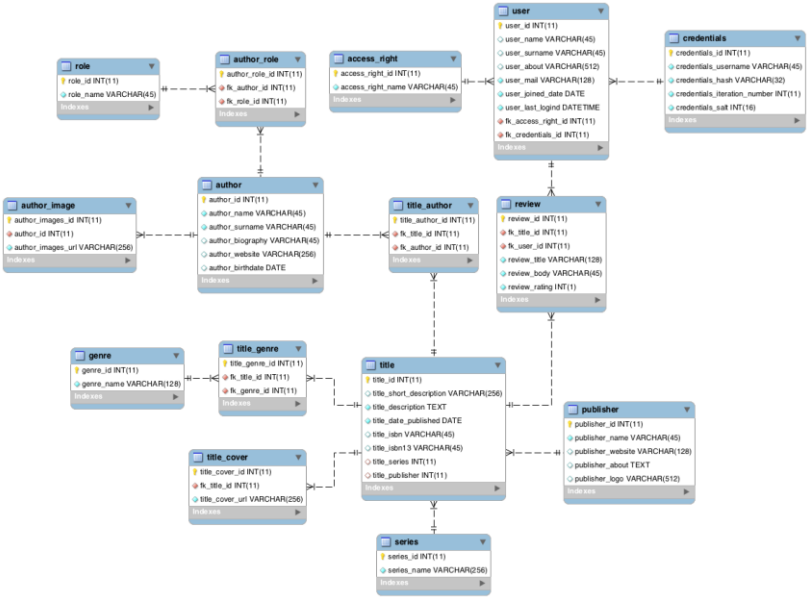
Prvo je bilo potrebno konfigurisati sam RDS. Odabrati tzv. endžin, odnosno tehnologiju baze podataka. Za potrebe ovog projekta se koristi MySQL. Takođe za svrhe testa se koristi jedna zona u kojoj bi se baza nalazila, jer se jedna zona može koristiti u okviru besplatnog programa. Za produkcijske svrhe moguće odabrati i *Multi Availability Zones*, odnosno da baza, radi dostupnosti bude propagirana na više fizičkih lokacija u regionu koje se nalaze u okviru Amazon AWS-a. Prilikom kreiranja instance baze, moraju se specificirati grupe sa pravom pristupa bazi (VPC Security Groups), port baze, MySQL verzija i slični parametri.

Kod kreiranja same instance je potrebno odrediti koja će se instanca koristiti i koliko memorije će baza zauzimati. Za potrebe projekta izabrana je instanca db.t2.micro koja poseduje jednojezgarni procesor od 1GHz, sa 1GB RAM memorije, ido 20GB SSDdiska za skladištenje. Prikaz odabira i izgleda interfejsa se nalazi na slici 5.



Sl. 5. Kreiranje RDS instance

A na Sl. 6 se može videti struktura, odnosno šema baze podataka i relacije između tabela.



Sl. 6. Prikaz šeme baze podataka korišćene u izradi projekta

VII. KOMPJUTING SERVIS - AMAZON EC2

Amazon Elastic Compute Cloud (Amazon EC2) je servis koji pruža prilagodljivi računarski kapacitet u oblaku.

Pruža potpunu kontrolu nad računarskim resursima, i smanjuje vreme potrebno da se zauzmu i puste u rad nove instance servera (Amazon EC2 *instance*), što omogućava brzo skaliranje kapaciteta, kako za njima raste potreba. Takođe, prilikom korišćenja Amazon EC2 se plaća samo za kapacitete koji se zaista i koriste. Instance se pokreću po potrebi, i naplaćuju po radnom satu i prema snazi virtuelne mašine koja je pokrenuta, kao i prostorom na disku koji joj je dostupan. Svaka instanca dobija svoju IP adresu na internoj Amazonovoj mreži (dostupnu samo u okviru nje), javnu dinamičku IP adresu (dostupnu svima), i FQDN (*Fully Qualified Domain Name*) adresu oblika ec2-xxx-yyy-zzz-ppp.compute-1.amazonaws.com koji je vezan za javnu dinamičku adresu. Ova adresa se ne menja dok se mašina ne zaustavi, osim u slučaju da se zahteva od Amazona statička adresa [6].

Iako klaud pruža praktično neograničen *on-demand* kapacitet, dizajn sistema bi trebalo da bude u mogućnosti da iskoristi te prednosti. Kao što je već bilo pomenuto u uvodu ovog rada, važno je da arhitektura sistema bude skalabilna. Generalno postoje dva načina skaliranja sistema: horizontalno i vertikalno.

Vertikalno skaliranje podrazumeva povećanje specifikacija pojedinačnih resursa, npr. pojačavanje servera većim čvrstim diskom, ili bržim procesorom. Na EC2 se to jednostavno postiže stopiranjem instance i izborom druge instance sa više RAM memorije, procesorskom moći, mrežnim mogućnostima itd. Međutim ovaj način skaliranja ima svoju granicu do koje može da ide, i nije uvek najisplativiji. Ali se ipak veoma jednostavno implamentira, i može često biti dovoljan scenario.

Horizontalno skaliranje podrazumeva povećanje broja resursa, npr. dodavanje novih čvrstih diskova, ili dodavanje novih servera koji bi podržali aplikaciju. Ovo je odličan izbor za arihtekturu aplikacije koja bi u potpunosti koristila elastičnost klauda [1].

Kao što je moguće pretpostaviti, klaud servisi i veliki broj EC2 instanci ne mogu biti najbolja zamena za sve primene. Tako određena vrsta poslova koja podrazumeva veliku računsku moć (*High Performance Computing Applications*) i komunikaciju između EC2 instanci često ne može da održi korak sa tzv *in-house* sistemima. Istraživanja pokazuju da postoji velika korelacija između vremena koje aplikacija provede komunicirajući i performansi EC2 instance [7]. Što isto tako dovodi do toga da će i cena za određene primene biti veća nego da se poseduje oprema [8]

Za potrebe projekta je izabrana Amazon Linux AMI 2017.03.1 (HVM) instanca sa SSD skladištem podataka, koja će biti pokretana na t2.micro

instanci sa jednim vCPU-om (jednojezgarnim procesorom Intel Xeon), koji radi na 2.5 GHz, i koji ima 1 GiB RAM memorije.

Podizanje bekend aplikacije na EC2 instancu u okviru ovog projekta se vrši preko ECS-a, odnosno prvo će biti konstruisan Docker image, i biti podignut na ECS klasteru, i zatim će biti pušten u rad. Za instancu je potrebno odrediti prava pristupa, odnosno koji su protokoli dozvoljeni na kojim portovima, kako bi serveru bilo moguće pristupiti spolja.

VIII. KOMPJUTING PLATFORMA BEZ SERVERA - AWS LAMBDA

Većina kompanija koje migriraju svoje sisteme na klaud pokušava da izbegne komplikacije i troškove migracije na taj način što će uraditi *deploy* svojih tradicionalnih monolitnih aplikacija koje koriste IaaS/PaaS rešenja [9]. U ovom kontekstu se monolitna aplikacija odnosi na jednu aplikaciju sa jedinstvenom velikom bazom koda koja pruža desetine ili stotine servisa koristeći različite interfejse kao HTML stranice, veb servise i REST servise [10].

Arhitektura mikroservisa daje rešenje čiji je cilj efikasno skaliranje resursa, i rešavanje drugih problema monolitne arhitekture [10]. Međutim arhitektura mikroservisa donosi druge izazove, kao što je rad potreban za deploy svakog mikroservisa, i njihovo skaliranje i upravljanje u klaud okruženju. Kako bi se izašlo u susret tim izazovima, servisi kao sto je AWS Lambda pružaju mogućnost implementacije arhitekture mikroservisa bez potrebe za upravljanjem, podešavanjem, i održavanjem servera [11].

Istraživanja pokazuju da je korišćenjem Lambda mikroservisa, u poređenju sa monolitnim i klasičnim mikroservisima, moguće ostvariti značajne uštede. A prednost je u tome što je moguće skalirati servis praktično neograničeno, i u veoma kratkom roku, kao i što vreme obrade podataka ne zavisi od opterećenosti servisa, jer je svaka lambda funkcija u izolovanom računskom okruženju [11]. U kombinaciji sa S3 servisom da uslužuje potrebe statičkog sadržaja, i API Gateway servisom kojim može da se razvije praktično neograničen broj sinhronih API-a koje izvršavaju Lambde, ovaj patern pruža kompletnu veb aplikaciju [12].

Plaća se samo za vreme izvršavanja koje se iskoristi, ne i kada se kod ne izvršava. Naplata se vrši na osnovu broja izvršenih lambdi, kao i na svakih 100ms izvršavanja [13]. Može se podesiti da se kod automatski pokrene iz nekog drugog AWS servisa, ili se može pustiti direktno iz neke web ili mobilne aplikacije. Lambda se može koristiti za proširenje drugih AWS servisa posebnom logikom ili da se automatski izvršava kao odgovor na neki događaj, npr. modifikacija nekog objekta u S3 korpi, ažuriranje tabele u Amazon DynamoDB, kao odgovor na poruku SNS-a i slično.

Kod koji se izvršava na Lambdi se naziva Lambda funkcija [14]. Lambda funkcije su bez stanja (eng. *stateless*), i nezavisne od infrastrukture, kako bi Lambda mogla da pokrene onoliko kopija funkcije koliko je potrebno da bi se skalirala zavisno od broja dolazećih događaja.

U okviru projekta se koristi za unos većih količina podataka o grafičkim novelama, odnosno kada je potrebno da se od određenog izdavača unesu podaci o svim novelama koje su izdali. Od izdavača se dobija .csv datoteka sa svim informacijama putem elektronske pošte, koja je uglavnom eksportovana iz njihove baze podataka. Tu se uvodi AWS SES u sprezi sa S3 servisom, tako što po pristizanju mejla od određenog izdavača, sa .csv fajlom u prilogu, sam fajl se automatski čuva na S3, i trigeruje se Lambda funkcija kojoj se prosleđuje putanja ka fajlu na S3 i ime izdavača. To se postiže tako što se "okidač" za lambdu postavi u S3, što se može uraditi i prilikom kreiranja nove lambda funkcije.

U ovom slučaju je potrebno da se funkcija izvrši kada se u korpi "email-csv-ingestion", čija je namena samo za prijem csv fajlova, nađe novi fajl koji počinje sledećim stringom "batch-ingest-", i završava se sufiksom "csv". Naravno korpa mora da pre ovog podešavanja postoji na S3. Nakon toga se sam kod funkcije može kopirati u online editor ili se može upload-ovati. Sam kod može biti u programskim jezicima Python, NodeJS, Java ili C#.

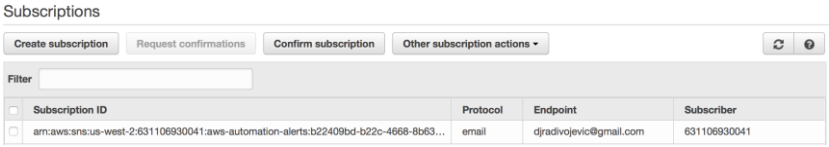
Lambda izvršava validaciju podataka u fajlu, i za one redove koji prođu validaciju šalje POST upit ka bekend aplikaciji koja poseduje JSON API za unos novih naslova u bazu. Za redove koji ne prođu validaciju se kreira novi .csv fajl na S3 u koji se ti redovi upisuju, i šalje se email alert poruka sa putanjom ka novom fajlu sa nevalidnim podacima nazad izdavaču, kako bi mogli da budu upoznati sa podacima koje treba ponovo poslati u pravom formatu. I jedan izveštaj se šalje osobi koji održava sistem. Lambda funkcija je potrebe ovog projekta pisana u Python jeziku.

IX. Servis za notifikacije - Amazon SNS

SNS (Amazon Simple Notification Service) omogućava slanje individualnih poruka velikom broju primaoca, neki od primera su *push* notifikacije za korisnike aplikacija za mobilne uređaje, primanje elektronske pošte, ili slanje poruka drugim distribuiranim sistemima. Preko SNS servisa se takođe mogu slati SMS poruke na mobilne telefone širom sveta. SNS može i da prosleđuje poruke SQS-u, Lambda funkcijama, ali i na bilo koju drugu HTTP krajnu tačku [15].

U radu se SNS servis koristi za obaveštavanje o neuspešnom izvršavanju lambda funkcije, o čemu je već bilo reči. Obaveštavanje se vrši putem elektronske pošte, i pored samog podešavanja tzv. *topic*-a bilo je potrebno i

prijaviti se email adresom na servis, odnosno potvrditi da se žele primati obaveštenja (Sl. 7.).

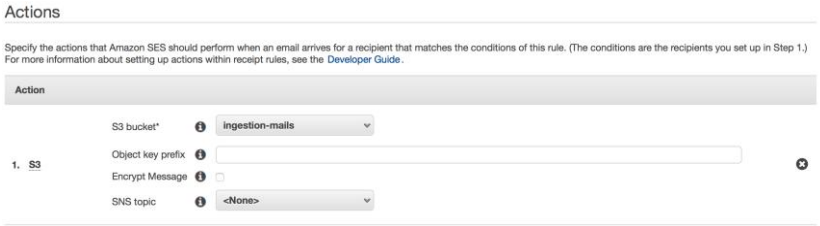


Sl. 7. Uspešno prijavljivanje adrese elektronske pošte

X. SERVIS ELEKTRONSKE POŠTE – SES

SES (Simple Email Servis) predstavlja servis za elektronsku poštu, koji je u mogućnosti da šalje i prima poštu. S obzirom na to da se nalazi u okviru AWS-a, vrlo je dobro integrisan sa ostalim servisima, i moguće je procesuirati primljenu poštu, izazivati određene događaje po potrebi, i svakakve druge vrste automatizacije.

U samom projektu se koristi za slanje poruka prilikom registracije korisnika, u kojima se nalazi link za potvrdu adrese elektronske pošte korisnika, resetovanje lozinki korisnika, kao i za obaveštenja o novostima (newsletter). Takođe se koristi kao što je pomenuto u radu u delu o Lambda funkcijama, za prijem informacija od izdavača, od kojih se očekuje da putem elektronske pošte pošalju .csv fajl u određenom formatu, nakon čega SES izaziva procesurianje tih fajlova. Da bi se fajl procesuirao, mora se pozvati lambda funkcija koja je zadužena za ekstrakciju priloga mejla, s obzirom na to da sam SES to ne podržava. SES je konfigurisan da čuva cele poruke u S3 korpi, koja je podešena da izazove lamba funkciju za ekstrakciju priloga (attachment), i njegovo čuvanje u email-csv-ingestion korpi, koja je opet podešena da izazove lambda funkciju za procesuiranje samog fajla. U ovom primeru se mogu videti ograničenja Amazonovog servisa, kojem je osnovna funkcionalnost da poziva druge AWS servise.



Sl. 21. Podešavanje SES-a da čuva poruke na S3

XI. MONITORING SERVIS – CLOUDWATCH

CloudWatch je monitoring servis koji prati različite parametre sistema (tzv. metrike), i na osnovu toga preduzima različite akcije. Moguće je slati obaveštenja o stanju različitih servisa elektronskom poštom ili SMS porukom, praviti sopstvene (eng. *custom*) metrike, koristiti CloudWatch u sprezi sa drugim servisima (AutoScale, Elastic Beanstalk) itd. [3]. On se isto tako koristi za čuvanje log fajlova, i na taj način menja konzolu koja bi postojala u lokalnom okruženju. Na taj način je moguće pratiti određene greške u sistemu, ispitivati stack trace i slično.

Jedan od primera je postavljanje alarma. U slučaju da u API odgovoru naiđemo na HTTP status kodove iz kategorije 5XX (*Server Error*) to označava da naš sistem nije uspeo da obradi neke podatke, alarm bi nam pokazao koji deo sistema treba da se popravi kako mi se bolje obradio taj podatak.

XII. PRIMER APLIKACIJE

Sama aplikacija nije centralna tema ovog rada, već bi trebalo da posluži kako samo kao primer, u cilju praktičnijeg i realnijeg pristupa konfigurisanja AWS servisa. Ali je potrebno obrazložiti svrhu i cilj same aplikacije kako bi se bolje razumelo zbog čega i na koji način se koriste i određeni AWS servisi i zbog koje funkcionalnosti aplikacije se koriste.

Bekend aplikacija je pisana u Java programskom jeziku. Jedan od korišćenih framework-a je Dropwizard, koji sadrži set alata koji olakšavaju pisanje REST web servisa. Bekend aplikacija koristi JSON format kao primarni API format. Postoji mogućnost da se koristi i XML, i da se putem Content-Type hedera u POST upitu određuje koji se format koristi.

Sve strukture koje bi trebalo da se koriste širom sistema, i pogotovo strukture API pitanja i odgovora će biti izdvojene u poseban projekat common-data i pozivan kao dependency u sve servise aplikacije kako bi se održala uniformnost podataka.

Takođe će dokumentacija API-a biti pisana uz pomoć Swagger programskog okvira. On olakšava pisanje dokumentacije za API-e, kao i generisanje klijenata za integraciju i korišćenje nekog API-a u tuđim projektima u različitim programskim jezicima. To značajno olakšava korišćenje API-a za sve spoljnje partnere. Takođe je vođenje dokumentacije veoma bitno za sve ljude koji rade na održavanju sistema, i Swagger doprinosi čitljivosti i organizaciji dokumentacije [16].

U okviru aplikacije, pristup bazi podataka se radi preko Hibernate ORM, koji predstavlja programski okvir za mapiranje tabela u bazi u Java objekte, što olakšava manipulaciju podacima u aplikaciji. Sam Hibernate se podešava

putem XML konfiguracije, i značajno ubrzava proces pisanja koda za pristup bazi, i prebacuje akcenat sa baza podataka i SQL query-ja na programski kod.

Kao alat za automatizaciju procesa build-a u projektu je iskorišćen Maven. Maven se dokazao kao industrijski standard, i ima veću podršku zajednice, pa se lakše pronalaze rešenja mogućih problema i izazova, što je glavni razlog zbog kojeg je izabran za korišćenje u projektu.

Kao što je već pomenuto u odeljku o CodeComitu, alat za verizonisanje koji se koristi je Git, koji je takođe industrijski standard, i najjednostavniji je za korišćenje i konfigurisanje.

Programski okvir za testiranje koda koji se koristi je JUnit. On služi pisanju tzv. unit testova koji testiraju izolovane funkcionalnosti pojedinačnih (unit - jedinica) komponenti sistema, odnosno modula. Njima se proverava da li sve te pojedinačne komponente ispravno izvršavaju svoju namenu sa svim očekivanim tipovima ulaza.

XIII. STANJE NA TRŽIŠTU PRUŽALACA KLAUD USLUGA I KONKURENCIJA

Računska moć AWS je po najnovijim istraživanjima pet puta veća od 14 najvećih konkurenata zajedno [17]. Na tržištu ponuđača usluga računarstva u oblaku postoji velika količina kompanija koje pružaju veoma uski opseg usluga, koje mogu da se uporede samo sa određenim servisima koje Amazon AWS pruža. Jedine dve kompanije koje pružaju sličan broj usluga su Microsoft Azure, i Google AppEngine. Broj usluga koje AWS pruža je ujedno i najveća prednost u odnosu na konkurenciju, jer se mogu objediniti sve usluge koje pružaju i praktično cela poslovanja IT kompanija, sistemi i tzv. *pipelines* se mogu naći na AWS oblaku, što značajno olakšava održavanje i administraciju sistema. Takođe je i njihova platforma veoma dug vremenski period na tržištu i veliki broj inženjera već ima iskustva sa AWS-om, što je pored detaljne dokumentacije, broja primera, pitanja i odgovora na najvećim portalima programerskih zajednica kao što je StackOverflow, često odlučujući faktor koji preteže na stranu AWS-a. U velikom broju slučajeva se dešava da pružaoci usluga koji imaju samo određene servise nude bolje performanse, po nižim cenama od Amazonovih. Međutim ako je potreba firme da poveže više servisa, onda se kompanije često ne odlučuju za njihove usluge, najviše zbog problema sa administriranjem više različitih sistema koji moraju međusobno da komuniciraju preko mreže, što značajno utiče i na performanse sistema [7]. Isto tako inženjeri moraju posedovati različite skupove znanja za različite platforme. I ne treba ni zanemariti da Amazon ima najbolju geografsku distribuciju svojih sistema, što je veoma značajno za aplikacije koje treba da imaju odziv velike brzine, ili čak da podležu zakonima određenih zemalja. U trenutku pisanja, AWS poseduje 14 regiona, dok Google poseduje 6 regiona u kojima posluje [18][19]. Microsoft ima 30

regiona, međutim time ne garantuje da u svim regionima poseduje više od jednog data centra, tako da nije moguće napraviti poređenje od bilo kakvog značaja.

Uporedni prikazi performansi i benčmark testovi pokazuju veoma slične rezultate kada se upoređuju, tako da to ne predstavlja ključnu stavku u izboru platforme [9].

Što se tiče cena usluga Google-ove i Microsoft-ove platforme koji su direktna konkurencija AWS, one su slične Amazonovoj platformi, za slične servise i performanse [9][20][21][22]. Međutim u trenutku pisanja teksta, obe konkurentske platforme poseduju manji broj servisa od AWS-a, i dobar deo konkurentskih servisa je tek u razvoju, odnosno u alfa ili beta fazi razvoja. Ali takođe ne treba isključiti činjenicu da nude i neke servise koje Amazonova platforma ne nudi, kao što je na primer Google BigData platforma BigQuery. S obzirom na sferu poslovanja Google-a, taj podatak ne iznenađuje. BigQuery omogućava analizu ogromnih količina podataka za veoma kratko vreme, čak pružajući uvid u podatke u realnom vremenu. Google-ov način naplaćivanja usluga se takođe spominje kao prednost u odnosu na Amazonov. Tako na primer, za razliku od EC2, Google Compute Engine i Microsoft Azure se naplaćuju, po minutu za "OnDemand" način plaćanja, dok je kod EC2 naplata zaokružuje na sat [23].

Za poslove koji se obavljaju duže vreme, naplata po minutu gubi smisao, i troškovi po satu postaju bitni.

XIV. ZAKLJUČAK O RADU

Kroz rad je predstavljena jednostavnost korišćenja AWS servisa, i potreban trud da se jedan projekat, ili aplikacija postavi u Amazonovom cloud okruženju. Veći akcenat je stavljen na sam AWS i mogućnosti koje pruža, nego na samu aplikaciju s obzirom na ti da je cilj rada da pokaže kroz primer aplikacije za ocenjivanje grafičkih novela korišćenje i podešavanje servisa, koji su svi servisi potrebni da se takva aplikacija postavi i koliko je truda potrebno od strane inženjera.

Glavna prednost servisa za računarstvo u oblaku, samim tim i AWS platforme je ta da nije potrebno posedovanje sopstvenog hardvera i infrastrukture, i takođe inženjera koji bi tu infrastrukturu održavali, čime se potrebni troškovi značajno smanjuju. Isto je važno napomenuti da bi i kreiranje i puštanje u rad ovakve aplikacije i sistema koji je prikazan, bilo praktično neizvodljivo bez ulaganja određene količine novca, dok je ceo ovaj projekat u okviru "*Free Tier*"-a, odnosno besplatnog programa, osim samog

domena koji je zakupljen. Čime se pokazuje i koliko je Amazonova platforma dobra za izradu studija izvodljivosti, odnosno prototipova.

U radu je zanemaren frontend deo, s obzirom na to da se kroz njega ne može prikazati rad AWS servisa, ali se podrazumeva da se i on izvršava na EC2 instanci, a resursi da se nalaze na S3. Sam frontend će biti podignut na zasebnoj EC2 instanci i komunicirati sa bekendom isključivo preko API-a, i biti autorizovan zasebnim apikey-jem.

Projekat isto tako može biti nadograđen korišćenjem i AWS servisa za build i deploy, odnosno AWS CodeBuild, CodeDeploy i CodePipeline. Na taj način bi se automatizovao ceo proces build-a, testiranja i deploy-a aplikacije, gde je moguće da se pri samom deploy-u podigne posebna EC2 instanca na koju bi se preusmerio saobraćaj sa postojeće, a zatim postojeća instanca ugasila. Isto tako je moguće i uvesti neke od novih servisa veštačke inteligencije, kao na primer Rekognition, servis koji služi prepoznavanju slika. Taj servis bi se mogao iskoristiti za prepoznavanje slika autora, gde bi u slučaju da ne prepozna lice na nekoj slici, ta slika prijavila i eventualno poslala na manuelnu proveru. Isti taj princip bi se mogao iskoristiti i za prijavu profilnih slika korisnika neprikladnog sadržaja [24].

XV. LITERATURA

[1] M. Kiran, P. Murphy, I. Monga, J. Dugan and S. S. Baveja, "Lambda architecture for cost-effective batch and speed big data processing," 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, 2015, pp. 2785-2792, doi: 10.1109/BigData.2015.7364082

[2] AWS CLI Command Reference - http://docs.aws.amazon.com/cli/latest/index.html , jun 2017.

[3] Developer Guide API Version 2006-03-01 , "Amazon Simple Storage(S3)".

[4] Miloš Savić, "Servisi AWS-a" - http://startit.rs/aws-amazon-web-services/ , 30. septembar 2015.

[5] Amazon Relational Database Service (Amazon RDS) - https://aws.amazon.com/rds/details/ , maj 2017.

[6] Amazon EC2 - Virtual Server Hosting: https://aws.amazon.com/ec2/details/ (avgust 2017).

[7] K. R. Jackson et al., "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud," 2010 IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, IN, 2010, pp. 159-168, doi: 10.1109/CloudCom.2010.69

[8] J. Emeras; S. Varrette; V. Plugaru; P. Bouvry, "Amazon Elastic Compute Cloud (EC2) vs. in-House HPC Platform: a Cost Analysis," in IEEE Transactions on Cloud Computing , vol.PP, no.99, pp.1-1, doi: 10.1109/TCC.2016.2628371

[9] B. S. Đorđević, S. P. Jovanović and V. V. Timčenko, "Cloud Computing in Amazon and Microsoft Azure platforms: Performance and service comparison," 2014 22nd Telecommunications Forum Telfor (TELFOR), Belgrade, 2014, pp. 931-934, doi: 10.1109/TELFOR.2014.7034558

[10] J. Lewis and M. Fowler, *Microservices*, 2014.

[11] M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," 2015 10th Computing Colombian Conference (10CCC), Bogota, 2015, pp. 583-590, doi: 10.1109/ColumbianCC.2015.7333476
[12] AWS Whitepaper, “AWS Serverless Multi-Tier Architectures”, novembar 2015.
[13] AWS Whitepaper, “Architecting for AWS Cloud: Best Practices”, februar 2016.
[14] Introducing AWS Lambda functions: https://aws.amazon.com/lambda/details/ , jun 2017
[15] Amazon SNS Product Details - https://aws.amazon.com/sns/details/ , jun 2017
[16] Swagger Specification - https://swagger.io/specification/ , jul 2017.
[17] Lydia Leong, Raj Bala, Craig Lowery, Dennis Smith, “Magic Quadrant for Cloud Infrastructure as a Service, Worldwide”, 3. avgust 2016.
[18] AWS Regions and Endpoints / http://docs.aws.amazon.com/general/latest/gr/rande.html , jun 2017
[19] Google Cloud Platform’s Locations - https://cloud.google.com/about/locations/ , jun 2017
[20] Amazon EC2 Pricing - https://aws.amazon.com/ec2/pricing/on-demand/ , jun 2017
[21] Google Cloud Platform Pricing - https://cloud.google.com/pricing/#details , jun 2017
[22] Microsoft Azure pricing - https://azure.microsoft.com/en-us/pricing/ , jun 2017
[23] Andrea Colangelo, “Google Cloud vs AWS: a comparison”, unpublished, 30. oktobar 2014
[24] Amazon Rekognition Documentation - http://docs.aws.amazon.com/rekognition/latest/dg/what-is.html , jun 2017.

This paper proposes a cloud based service architecture based on AWS (Amazon Web Services) cloud platform, for the construction of a web portal used for ratings and writing reviews. As an example in this paper, users of the portal give ratings to graphic novels. For the used AWS services, it is shown part of initialization and configuration, some are compared from techincal and financial standpoint to the services of other companies and also to the scenario of not using cloud but having own equipment.

A PROPOSAL FOR THE USE OF AMAZON WEB SERVICESFOR DEVELOPING A CLOUD BASED APPLICATION

Djordje Radivojevic
dr Dusan Vujosevic