Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/25BE42Erdeg

IMPLEMENTACIJA SISTEMA ZA PLAĆANJE NA AWS INFRASTRUKTURI

IMPLEMENTATION OF A PAYMENT SYSTEM ON AWS INFRASTRUCTURE

Monika Erdeg, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su opisane komponente i arhitektura sistema za online plaćanje karticama. Analizirana je mogućnost za eksploataciju sistema na elementima računarstva u sklopu AWS infrastrukture. Predstavljena je automatizacija procesa raspoređivanja softvera i konfiguracije infrastrukture.*

Ključne reči: *Elektronsko plaćanje, CI/CD, GitHub akcije, IaC, Terraform, AWS*

Abstract – *This paper presents the implementation of a payment system. Deployment processes are discussed. The paper also provides a detail expanation of the AWS infrastructure setup, which is used for hosting the application. Finally, it discloses infrastructure automation and proposes a solution for Infrastructure as Code.*

Keywords: *Electronic payment, CI/CD, GitHub actions, IaC, Terraform, AWS*

1. UVOD

U današnjem digitalnom dobu, online kupovina postala je neizostavan deo svakodnevnog života. Sa sve većim brojem korisnika koji se oslanjaju na digitalne platforme za obavljanje transakcija, sigurnost i stabilnost sistema za plaćanje postali od suštinskog značaja. AWS (Amazon Web Services) infrastruktura pruža mogućnosti za simulaciju i implementaciju bankarskog sistema za plaćanje karticama tokom online kupovine.

U savremenom poslovnom okruženju, implementacija skalabilne infrastrukture zahteva ne samo tehničko razumevanje već i duboko poznavanje raznih aspekata upotrebe servisa i kompromisa koje je potrebno napraviti radi postizanja optimalnih rezultata. AWS infrastruktura omogućava simulaciju raznih scenarija, analizu performansi i testiranje sigurnosnih mehanizama.

U ovom radu će biti istraženo kako AWS može da se iskoristi za simulaciju i implementaciju sistema za plaćanje karticama. Fokus će biti na analizi različitih komponenti infrastrukture, kao i na implementaciji mehanizama raspoređivanja, kako aplikacije, tako i infrastrukture, kako bi se osigurala stabilnost i skalabilnost sistema. Takođe, biće stavljen akcenat na povezivanje lokalnih (on-premise) komponenti sa sistemom koji se izvršava u oblaku.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Goran Sladić, red. prof.

Kroz detaljno istraživanje i praktične primere, razmotriće se kako AWS pruža rešenja koja su ključna za uspešno obavljanje online transakcija u današnjem digitalnom ekosistemu.

2. SISTEM ZA PLAĆANJE KARTICAMA PRILIKOM ONLINE KUPOVINE

Koncentrator plaćanja predstavlja sistem kojim upravlja entitet čiji poslovni model je da posreduje između različitih servisa za plaćanje i klijenata koji žele da podrže plaćanje, ali ne žele da brinu o bezbednosti datih funkcija i njihovom održavanju.

Koncentrator ima API koji očekuje podatke o uplati. Otvara se stranica gde se bira kojim servisom plaćanja kupac želi da plati svoju kupovinu.

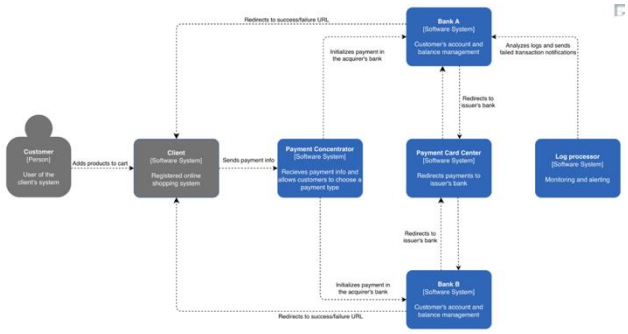
Plaćanje putem platne kartice podrazumeva protokol komunikacije između različitih učesnika:

- Kupac, koji vrši onlajn kupovinu i ima otvoren račun u banci,
- Prodavac, koji prodaje proizvode putem veb-prodavnice i poseduje račun u banci,
- Banka prodavca (Acquirer), koja pruža servis za onlajn plaćanje, gde kupac unosi podatke o svojoj platnoj kartici,
- Centar za platne kartice (PCC - Payment Card Center) – posreduje u međubankarskoj komunikaciji ovog tipa, te prihvata zahtev za transakciju od banke prodavca i prosleđuje ga banki kupca,
- Banka kupca (Issuer) – proverava stanje računa kupca i odobrava transakciju ukoliko postoje raspoloživa sredstva.

2.1. Pregled osnovnih komponenti sistema

Delovi sistema koji su deo implementacije koncentratora plaćanja, prikazanih na slici 1, su sledeći:

- Payment Concentrator - koncentrator plaćanja
- Bank A - banka u kojoj su registrovani neki od klijenata koncentratora (kupci/prodavci)
- Bank B - banka u kojoj su registrovani neki od klijenata koncentratora (kupci/prodavci)
- Payment Card Center - centar za platne kartice
- Log processor – sistem za analizu log-ova



Slika 1. *C1 dijagram sistema*

Od navedenih komponenti sistema, jedan deo (koncentrator plaćanja, centar za platne kartice i banke) se izvršava u oblaku, a drugi deo (log procesor) se izvršava lokalno (engl. *on-premise*). Potrebno je povezati ova dva dela sistema, kako bi mogla da komuniciraju i razmenjuju podatke sigurnim putem.

Aplikacije su pisane u Spring Boot radnom okviru. Za bazu podataka je korišten PostgreSQL servis za skladištenje i upravljanje podacima.

3. AMAZON WEB SERVICES

Amazon Web Services (AWS) predstavlja vodećeg pružaoca usluga za računarstvo u oblaku koji nudi širok spektar infrastrukturnih resursa i servisa kao usluga. AWS je deo Amazon.com kompanije i omogućava korisnicima pristup skalabilnim i fleksibilnim resursima u oblaku kako bi implementirali, testirali i pustili u produkciju svoje aplikacije i IT infrastrukture.

3.1 IAM (Identity and Access Management)

AWS Identity and Access Management (IAM) je servis koji nudi AWS, koji omogućava upravljanje korisničkim identitetima i njihovim pristupom AWS resursima. IAM omogućava sigurno kontrolisanje pristupa različitim AWS servisima i resursima, čime se obezbeđuje poverljivost i integritet podataka u AWS okruženju [1].

3.2 VPC (Virtual Private Cloud)

VPC (virtuelna privatna mreža) je servis AWS-a koji omogućava kreiranje privatne mreže u okviru AWS oblaka. Ova mreža je potpuno izolovana od drugih VPC-ova i resursa, čime se obezbeđuje visok nivo privatnosti i bezbednosti za aplikacije i podatke. Kroz jednostavne konfiguracije, omogućeno je prilagođavanje postavki VPC-a. To uključuje definisanje IP adresa za podmreže, kreiranje ruta i konfigurisanje mrežnih pristupnih kontrola (Netwrok Access Control List) i sigurnosnih grupa [2].

3.3 EC2 (Elastic Cloud Compute)

Elastic Cloud Compute (EC2) je AWS servis koji omogućava korisnicima da iznajme virtuelne servere u oblaku i pokreću svoje aplikacije na tim serverima. U okviru EC2 servisa, postoji nekoliko mogućnosti za izbor tipa servera: on-demand instance, spot instance, rezervisane instance (reserved instances) i dodeljeni serveri (dedicated hosts) [3].

3.4 ECS (Elastic Container Service)

Amazon Elastic Container Service (ECS) je AWS-ov servis koji omogućava upravljanje kontejnerizovanim aplikacijama. Pomoću ECS-a moguće je jednostavno pokrenuti, zaustaviti i skalirati kontejnere koji se izvršavaju u datom okruženju [4].

3.4.1 ECS Anywhere

ECS Anywhere je servis koji pruža ECS, koji omogućava pokretanje i upravljanje kontejnerima u lokalnom okruženju, van AWS infrastrukture u oblaku. Ova usluga proširuje funkcionalnosti ECS-a i omogućava korišćenje istog koncepata i alata za upravljanje kontejnerima kako u oblaku, tako i u lokalnom okruženju [5].

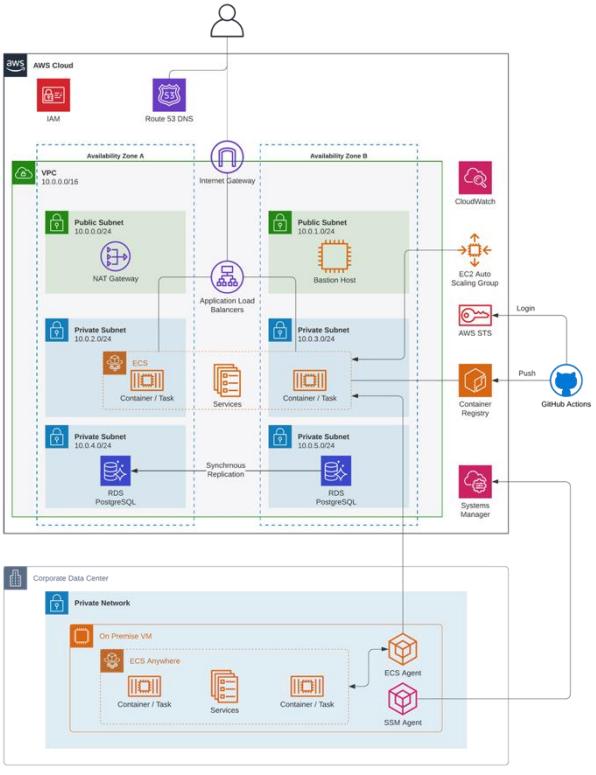
3.5 RDS (Relational Database Service)

Amazon Relational Database Service (RDS) je servis AWS-a koji omogućava lakše upravljanje relacionim bazama podataka u oblaku. RDS podržava različite popularne baze podataka, kao što su MySQL, PostgreSQL, Oracle, SQL Server, MariaDB i Amazon Aurora [6].

3.6 Route53

Amazon Route 53 je web servis za upravljanje DNS-om (Domain Name System). Omogućava korisnicima da registruju i upravljaju imenima domena, pridružuju ih AWS resursima i rutiraju korisničke zahteve ka pravim AWS servisima ili drugim infrastrukturnim resursima. Doprinosi poboljšanju performansi, dostupnosti i sigurnosti aplikacija i web sajtova [7].

4. INFRASTRUKTURA SISTEMA ZA PLAĆANJE



Slika 2. *Dijagram AWS infrastrukture sistema*

Na slici 2 je prikazan dijagram AWS infrastrukture datog sistema. Radi jednostavnosti prikaza, sve komponente su prikazane singularno (baze podataka, *load balanser*-i, ECS servisi i GitHub akcije nisu prikazani za svaku od aplikacija, već zbirno).

5. INFRASTRUKTURA KAO KOD

Infrastruktura kao kod (Infrastructure as Code) je proces kreiranja i upravljanja infrastrukturom putem koda, umesto putem manuelnih procesa.

Sa IaC-om, kreiraju se konfiguracioni fajlovi koji sadrže specifikacije infrastrukture, što olakšava održavanje i distribuciju konfiguracija. Ovaj pristup takođe garantuje kreiranje istog okruženja pri svakom pokretanju. Kroz pretapanje konfiguracija u kod, kao i njihovo dokumentovanje, IaC doprinosi izbegavanju nedozvoljenih ili ad-hoc promena koje nisu dokumentovane [8].

Verzija kontrole je važan deo IaC-a, te bi konfiguracioni fajlovi trebali biti pod kontrolom izvornog koda (source control-om), kao i svaki drugi fajl sa softverskim izvornim kodom. Implementacija infrastrukture kao koda donosi i mogućnost dekompozicije infrastrukture na modularne komponente, koje mogu biti kombinovane na različite načine [8].

Automatizacija infrastrukture putem IaC-a, znači da programeri ne moraju ručno upravljati serverima, operativnim sistemima, skladištem i drugim infrastrukturnim komponentama prilikom razvoja aplikacija. Pisanjem infrastrukture putem koda dobija se šablon za konfigurisanje iste, koji može da se iskoristi za replikaciju na nova okruženja u roku od nekoliko minuta [8].

5.1 Prednosti IaC-a

Prednosti IaC-a se ogledaju u sledećim stavkama [8]:

- Smanjenje troškova: Obezbeđivanje infrastrukture je nekada zahtevalo dosta vremena i bilo je skupo. Sa IaC-om, moguće je automatski upravljati infrastrukturom, što smanjuje troškove i potrebu za ručnim intervencijama.
- Povećanje brzine implementacija: IaC omogućava brže i češće izmene infrastrukture. Ona može biti brže kreirana, skalirana i uklonjena, što podržava agilni razvoj softvera i brze iteracije.
- Smanjenje grešaka: Korišćenjem IaC-a, minimiziraju se greške koje se mogu pojaviti pri ručnom konfigurisanju infrastrukture. Konfiguracija je dosledna i tačno definisana u kodu, što eliminiše slučajne greške i smanjuje verovatnoću ljudskih grešaka.
- Poboljšanje doslednosti infrastrukture: IaC obezbeđuje doslednost u konfiguraciji infrastrukture na svim okruženjima. Infrastrukturni resursi se konfigurišu na isti način u različitim okruženjima, što pomaže u održavanju konzistentnosti i smanjuje razlike u konfiguraciji.
- Eliminisanje odstupanja u konfiguraciji: IaC osigurava da infrastruktura bude u skladu sa definisanim specifikacijama. Svako odstupanje od željenog stanja može biti automatski ispravljeno, što eliminiše konfiguracione promene koje nisu dokumentovane i nepredviđene probleme.

5.2 Terraform

Alati za infrastrukturu kao kod (IaC) omogućavaju upravljanje infrastrukturom putem konfiguracionih fajlova, umesto putem grafičkog korisničkog interfejsa. IaC omogućava postavljanje i upravljanje infrastrukturom na siguran, dosledan i ponovljiv način, definišući konfiguracije resursa koje se mogu verzionisati, ponovo koristiti i deliti [9].

Terraform je alat za infrastrukturu kao kod kompanije HashiCorp. Omogućava definisanje resursa i infrastrukture u ljudski čitljivim, deklarativnim konfiguracionim fajlovima i upravlja životnim ciklusom infrastrukture.

Korišćenje Terraforma ima nekoliko prednosti u odnosu na ručno upravljanje infrastrukturom [9]:

- Terraform može upravljati infrastrukturom na više platformi.
- Jezik konfiguracije, koji je lako čitljiv, omogućava brzo pisanje infrastrukturnog koda.
- Stanje (state) Terraform-a omogućava praćenje promena resursa tokom implementacije.
- Konfiguracije se mogu čuvati u sistemu za verzionisanje, kako bi bilo omogućeno bezbedno sarađivanje na infrastrukturi.

Za implementaciju infrastrukture putem Terraform-a, neophodni su sledeći koraci:

1. Definicija obima (Scope) - Identifikacija projektne infrastrukture.
2. Kodiranje (Author) - Pisanje konfiguracije infrastrukture.
3. Inicijalizacija (Initialize) - Instaliranje alata koje Terraform koristi za upravljanje infrastrukturom.
4. Planiranje (Plan) - Pregledanje promena koje će Terraform napraviti da bi se uskladila konfiguracija.
5. Implementacija (Apply) - Sprovođenje planiranih promena.
6. Uništavanje (Destroy) - Brisanje svih resursa.

6. CI/CD

CI/CD (Continuous Integration/Continuous Delivery) pipeline automatizuje proces isporuke softvera. Pipeline obuhvata proces izgradnje (build-a) koda, izvršavanja testova (CI) i objavljivanje (deployment) nove verzije aplikacije (CD) [10].

Automatizovani proces doprinosi eliminaciji grešaka, koje bi mogle da se potkradu prilikom manuelnog izvršavanja procesa, pruža češće povratne informacije razvojnim timovima i omogućava kvalitetniju isporuku proizvoda [10].

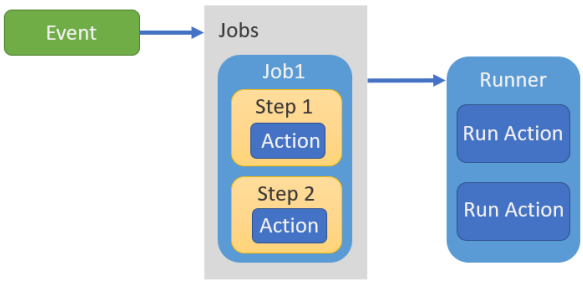
Implementacija CI/CD pipeline-a omogućava automatsku izgradnju, testiranje i isporuku softvera, što značajno povećava efikasnost razvoja.

Ovaj pristup eliminiše potrebu za ručnim koracima, smanjuje greške koje mogu nastati usled prisustva ljudskog faktora i omogućava bržu isporuku softvera. Time se ne samo poboljšava produktivnost, već i obezbeđuje kvalitetniji softver i kraće vreme do izlaska na tržište (time to market) [10].

6.1 GitHub akcije

GitHub Actions je platforma za kontinuiranu integraciju i isporuku (CI/CD) koja omogućava automatizaciju izgradnje, testiranja i isporuke aplikacija. Korisnici mogu kreirati radne tokove koji će se automatski pokretati pri određenim događajima u repozitorijumu, kao što su otvaranje "issue"-a ili "pull request"-a. Ova platforma nudi virtuelne mašine za izvršavanje radnih tokova, a takođe omogućava i korišćenje sopstvenih mašina (self-hosted runners) u korisničkoj infrastrukturi [11].

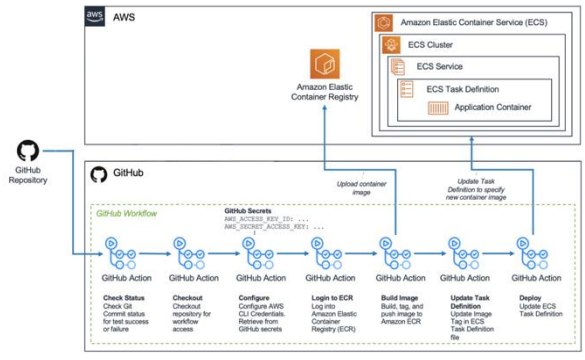
GitHub Actions se sastoji od nekoliko ključnih komponenti. Radni tokovi se konfigurišu da se pokreću na osnovu određenih događaja u repozitorijumu. Oni sadrže jedan ili više poslova (jobs), koji se mogu izvršavati sekvencijalno ili paralelno. Svaki posao se izvršava unutar sopstvenog virtuelnog mašinskog izvršioca (runner) ili kontejnera i sastoji se od više koraka (steps), kao što je prikazano na slici 3. Koraci mogu izvršavati prilagođene skripte ili akcije (actions), koje su ponovljivi dodaci koji pojednostavljuju proces radnog toka [11].



Slika 3. *Struktura GitHub akcije* [12]

6.1.1 Radni tok sistema za plaćanje

Sledeći dijagram (slika 4) predstavlja pregled osnovnih komponenti radnog toka aplikacije. Ova arhitektura predstavlja potpuni CI/CD proces koji koristi GitHub radni tok za automatsku koordinaciju izgradnje, testiranja i isporuke aplikacije na ECS (*Elastic Container Service*) za svaku izmenu u repozitorijumu.



Slika 4. *Referentni radni tok* [13]

7. ZAKLJUČAK

U radu je opisan sistem za online plaćane karticama, odnosno, simulacija bankarskog sistema za plaćanje. Softver se kao servis integriše sa aplikacijama koje žele da naplate svoje proizvode putem online transakcije. Omogućuje brzo i jednostavno monetizovanje bilo koje aplikacije.

Istražena je integracija lokalnih (on-premise) mašina sa AWS ekosistemom putem ECS Anywhere servisa. Opisana je registracija mašina i konfiguracija ove postavke, uz rezultate i prednosti datog servisa.

Naglasak je stavljen na mehanizme raspoređivanja i infrastrukturu sistema. Opisani su CI/CD radni tokovi, koji su implementirani korišćenjem GitHub akcija, kao i AWS servisi koji su korišteni za hosting aplikacija. Infrastruktura je napisana putem Terraform IaC alata. Detaljno je dokumentovana konfiguracija celog sistema.

Integracija koncentratora plaćanja sa pravim bankama bi bio prvi sledeći korak u daljem razvoju sistema. Dodavanje novih metoda plaćanja bi takođe moglo da bude jedno od unapređenja.

8. LITERATURA

[1] https://docs.aws.amazon.com/iam/index.html

[2] https://docs.aws.amazon.com/vpc/

[3] https://docs.aws.amazon.com/ec2/

[4] https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html

[5] https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-anywhere.html

[6] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

[7] https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html

[8] https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac

[9] https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code

[10] https://semaphoreci.com/blog/cicd-pipeline#h-what-is-a-ci-cd-pipeline

[11] https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

[12] https://ithelp.ithome.com.tw/articles/10262377

[13] https://aws.amazon.com/blogs/containers/create-a-ci-cd-pipeline-for-amazon-ecs-with-github-actions-and-aws-codebuild-tests/

Kratka biografija:

Monika Erdeg rođena je u Novom Sadu 1995. god. Fakultet tehničkih nauka u Novom Sadu upisala je 2014. godine. Završila je osnovne akademske studije 2018. godine, na smeru Softversko inženjerstvo i informacione tehnologije.

kontakt: monika.erdeg@gmail.com