Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/22BE25Stamenic

OBRADA I VIZUELIZACIJA PODATAKA UZ OSLONAC NA AWS

DATA PROCESSING AND VISUALIZATION USING AWS SERVICES

Nikolina Stamenić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U ovom radu opisana je implementacija i specifikacija arhitekture sistema za obradu i vizuelizaciju podataka koji se oslanja na AWS servise.*

Ključne reči: *Obrada podataka, AWS, StepFunctions, DynamoDB, S3, API Gateway, Lambda*

Abstract – *This paper describes the implementation and specification of the architecture of the data processing and visualization system that relies on AWS services.*

Keywords: *Data processing, AWS, StepFunctions, DynamoDB, S3, API Gateway, Lambda*

1. UVOD

Upravljanje podacima podrazumeva proces praćenja toka podataka od trenutka kreiranja do trenutka kada oni pronalaze svoju upotrebu. Često je potrebna dodatna obrada i izmena podataka pre nego što se dovedu u stanje potrebno za dalju upotrebu.
Obrada podataka predstavlja težak i vremenski zahtevan zadatak ukoliko se radi ručno. Tehnološki razvoj je doveo do pojave alata koji poseduju mogućnost lake i znatno brže obrade podataka. Pomenuti alati zahtevaju da korisnik poseduje određeno znanje o njihovoj upotrebi. Takođe, potrebno je da korisnik bude upućen u to na koji način podaci treba da budu obrađeni i koje izmene je potrebno napraviti.
Sistem koji se opisuje u ovom radu omogućava laku obradu podataka, gde je potreba za korisnikovo angažovanje svedena na minimum. Ideja je da korisnik obezbedi ulazne podatke, a da se sva dalja obrada obavi automatski i da korisnik kao odgovor dobije gotov rezultat sa primenjenim izmenama.
Sistem je razvijan za obradu podataka prikupljenih sa „Portala otvorenih podataka“ [1]. Radi se o podacima o zagađenjima vazduha u Republici Srbiji. Vrši se obrada ovih podataka tako da krajnji rezultat obezbeđuje informacije o zagađenjima po regijama u državi, koje delatnosti ispuštaju najviše zagađujućih materija u atmosferu, kao i koja postrojenja fabrika su najveći zagađivači.

2. KORIŠĆENE SOFTVERSKE TEHNOLOGIJE

Za razumevanje rada potrebno je poznavanje osnova o AWS i njegovim servisima korišćenim za izradu rešenja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Miroslav Zarić, vanr. prof.

2.1. AWS

AWS [2] je najsveobuhvatnija i najprihvaćenija platforma za računarstvo u oblaku (engl. *cloud computing*) na svetu koja korisnicima obezbeđuje preko 200 različitih servisa dostupnih u svakom trenutku. Dostupni su servisi IaaS tipa, koji obezbeđuju samo infrastrukturu, servisi tipa PaaS koji nude platformu, pa pored hardverskog dela obezbeđuju i konfigurisan paket softverskih podsistema i servisi SaaS tipa koji predstavljaju već implementirana softverska rešenja spremna za korišćenje.

2.2. AWS StepFunctions

AWS *StepFunctions* [3] je servis za orkestraciju procesa koji omogućava korisnicima kreiranje i upravljanje aplikacijama čiji je radni tok sačinjen od više koraka. Pri svakom koraku radnog procesa AWS vodi računa o ulaznim i izlaznim parametrima, o rukovanju greškama koje mogu da se dese, kao i o ponavljanjima izvršavanja koraka ukoliko je to potrebno.

StepFunctions su izgrađene po principu konačnog automata - mašine prelaza stanja (engl. *state machine*). Rade na principu podele posla u manje zadatke gde jedno stanje predstavlja jedan zadatak. Izvršavanje zadataka može da se obavlja na preko 200 AWS servisa. Svako stanje ima indikator koje je sledeće stanje u koje se prelazi nakon njegovog izvršavanja, pa se tako vrši orkestracija procesa i kontrola redosleda izvršavanja zadataka.

2.3. AWS DynamoDB

DynamoDB [4] je nerelaciona baza podataka zasnovana na ključ-vrednost i dokument strukturama podataka. Omogućava fleksibilnost šema podataka, pa svaki red u tabeli može da sadrži drugačiji broj i vrstu kolona. Podaci iz svake tabele su smešteni u celine koje se nazivaju particije, a jedna tabela u zavisnosti od količine podataka može da poseduje jednu ili vise particija.

2.4. AWS S3

Amazon Simple Storage Service [5], skraćeno S3, je servis koji služi za virtuelno skladištenje objekata. S3 čuva objekte u kontejnerima koji se nazivaju korpe (engl. *buckets*). Na jednom AWS nalogu korisnik može imati jednu ili više „korpi“, a za svaku od njih pojedinačno mogu se postaviti prava pristupa, čitanja i kreiranja novih objekata.

Objekti se čuvaju po principu ključ-vrednost, gde je ključ naziv fajla koji se čuva, a vrednost sam fajl. Dozvoljeno je postavljanje svih tipova fajlova, a maksimalna veličina fajla koja je moguća za postavljanje je 5 terabajta.

2.5. AWS Lambda

AWS Lambda [6] je računarski servis koji ima mogućnost izvršavanja koda. Lambda funkcije su samostalne aplikacije napisane u jednom od podržanih jezika u izabranom radnom okruženju.

Funkcije mogu da obavljaju bilo koju vrstu računarskih zadataka, od opsluživanja veb stranica i obrade tokova podataka, do pozivanja API-a i integracije sa drugim AWS servisima.

Upotreba lambdi otklanja potrebu za održavanjem servera radi pokretanja koda. AWS u pozadini vodi računa o serverima na kojima se lambde pokreću, operativnim sistemima, o mrežnom nivou i ostatku infrastrukture, pa korisnik može da se fokusira na kreiranje koda aplikacije i ne mora da obraća pažnju na konfiguraciju i održavanje infrastrukture.

2.5. AWS API Gateway

AWS API Gateway [7] je servis koji omogućava lako kreiranje, objavljivanje, praćenje i održavanje API-a na siguran način. Ovaj servis daje mogućnost kreiranja RESTful i WebSocket API-a uz pomoć kojih dobijamo aplikacije za dvosmernu komunikaciju u realnom vremenu.

API Gateway obavlja poslove koji uključuju prihvatanje i obradu velikog broja konkurentnih zahteva i upravljanje mrežnim saobraćajem. Vodi računa o zahtevima čija obrada u datom trenutku nije moguća, održava ih u stanju čekanja i vrši njihovu obradu kada se ona omogući. *API Gateway* ima mogućnost vršenja autorizacije, kao i provere validnosti poslatog zahteva.

2.6. Serverless framework

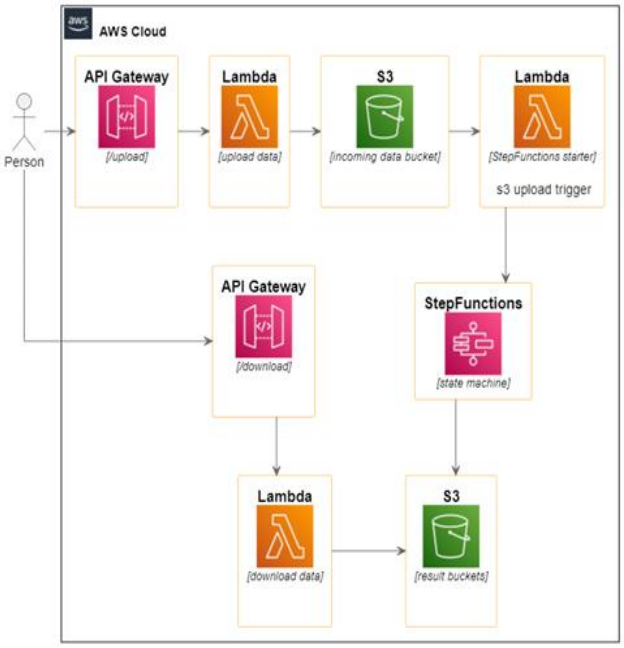
Serverless framework [8] – u prevodu okvir za razvoj aplikacija bez serverskih instanci, predstavlja rešenje za brzo kreiranje i otpremanje aplikacija koje koriste usluge direknto ponuđene od strane AWS i za čiju upotrebu nije neophodno konfigurisati serverske instance. Ovaj okvir uz upotrebu jedne komande izvršava pripremu, pakovanje i otpremanje koda. Omogućava otpremanje više AWS servisa u isto vreme, korišćenjem konfiguracionog fajla u kojem su navedeni potrebni servisi i parametri koje oni zahtevaju, kao i međusobne veze između servisa.

2.7. Python boto3

Boto3 [9] je biblioteka kreirana za *python* programski jezik. Predstavlja komplet za razvoj softvera za AWS servise. Podržava rad sa velikim brojem servisa i za svaki od njih postoji odgovarajući boto3 klijent. Ona omogućava veoma lako slanje zahteva za kreiranje, konfiguraciju, dobijanje ili izmenu podataka u sklopu AWS servisa.

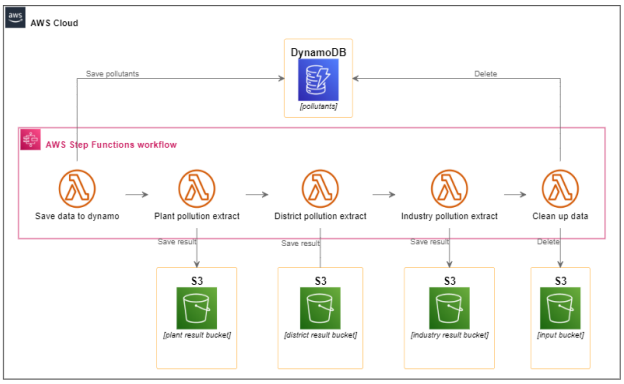
3. SPECIFIKACIJA

Na slici 1 prikazan je dijagram komponenti koje se nalaze u sistemu. Proces obrade podataka se pokreće slanjem zahteva za postavljanje ulaznog fajla. API Gateway prihvata dospeli zahtev i prosleđuje ga lambdi koja obavlja čuvanje zahteva u odgovarajuću S3 korpu.



Slika 1. *Dijagram komponenti*

Na svako postavljanje objekta u S3 korpu podešeno je da se izvrši okidanje lambde koja ima funkciju pokretanja StepFunctions procesa čiji detalji su prikazani na slici 2.



Slika 2. *StepFunctions radni proces*

Prvi korak u *StepFunctions* procesu jeste prikupljanje podataka o zagađujućim materijama i njihovo čuvanje u DynamoDB bazi podataka.

Nakon što su potrebni podaci obezbeđeni i sačuvani u bazu pokreću se tri uzastopne lambde koje vrše generisanje rezultujućih fajlova. Svaka od ovih lambdi vrši odgovarajuću obradu nad podacima, tako da se dobiju informacije o količinama zagađujućih materija po okrugu, delatnosti ili postrojenju fabrike. Generisane rezultate lambde čuvaju u odgovarajuće S3 korpe.

Nakon što su rezultati generisani, pokreće se poslednji korak u radnom procesu koji je čišćenje podataka. Vrši se brisanje ulaznog fajla kao i entiteta koji su uneseni u DynamoDB bazu, jer je njihova primena bila isključivo u procesu obrade podataka.

Po završetku izvršavanja radnog procesa korisniku se otvara mogućnost pregleda generisanih rezultata i njihovo preuzimanje u formi csv fajla. Za svaki od tipova rezultata

korisnik ima mogućnost pregleda u sklopu tabele koja je prikazana na veb stranici.

4. IMPLEMENTACIJA

Sve AWS komponente korišćene za kreiranje sistema napravljene su uz pomoć *Serverless Framework*-a. Fajl je podeljen u konfiguracione celine, a na slici 3 prikazana je celina pod nazivom *provider*. U njoj se podešava tip radnog okruženja, navodi se ograničenje trajanja lambda funkcija i dozvole koje su potrebne AWS servisima.

```
provider:
  name: aws
  runtime: python3.8
  timeout: 15
  iam:
    role:
      statements:
        - Effect: 'Allow'
          Action:
            - 'states:*'
          Resource: "*"
```

Slika 3. *Provider deo u serverless.yaml fajlu*

U *functions* delu se navode definicije i konfiguracije svih lambda funkcija. Navodi se referenca na fajl u kojem se nalazi kod funkcije.

Svakoj lambdi se dodeljuju potrebne dozvole za komunikaciju sa drugim AWS servisima. Primer konfiguracije jedne od lambda funkcija je prikazan na slici 4.

```
industryPollutionExtract:
    handler: functions/industry-pollution-extract.handle
    iamRoleStatements:
      - Effect: 'Allow'
        Action:
          - 's3:GetObject'
        Resource: "arn:aws:s3:::incoming-data-values/*"
      - Effect: 'Allow'
        Action:
          - 's3:PutObject'
        Resource: "arn:aws:s3:::industry-pollutants/*"
      - Effect: 'Allow'
        Action:
          - 'dynamodb:Query'
        Resource: !Sub
"arn:aws:dynamodb:${AWS::Region}:${AWS::AccountId}:table
/pollutions/index/input_id_gsi"
```

Slika 4. *Konfiguracija lambda funkcije u serverless.yaml fajlu*

Kao okidač funkcija koje rade otpremanje fajlova ili preuzimanje gotovih rešenja postavljen je API Gateway. Njegova konfiguracija se unosi u sklopu podešavanja lambda funkcije i prikazana je na slici 5.

```
events:
      - http:
          path: download/{type}/{year}
          cors: true
          method: get
          request:
            parameters:
              paths:
                type: true
                year: true
          response:
            headers:
              Access-Control-Allow-Origin: "*"
```

Slika 5. *Konfiguracija API Gateway lambda okidača*

Podešavanje ostalih servisa se obavlja u *resources* delu. Za *StepFunctions* je potrebno navesti dozvole za pokretanje svih lambdi koje su potrebne za izvršavanje procesa.

Stanja u mašini stanja navode se željenim redosledom, gde se za svako od njih naznačava koju akciju će obavljati i koje stanje sledi nakon njegovog uspešnog izvršavanja. Način na koji su kreirana stanja u mašini stanja je prikazan na slici 6.

```
"StartAt": "SaveDataToDynamo",
"States": {
  "SaveDataToDynamo": {
    "Type": "Task",
    "Resource": "${saveDataLambdaArn}",
    "Next": "WaitTenSeconds"
  },
  "WaitTenSeconds": {
    "Type": "Wait",
    "Seconds": 10,
    "Next": "PlantPollutionExtract"
  },
  "PlantPollutionExtract": {
    "Type": "Task",
    "Resource": "${plantPollutionLambdaArn}",
    "Next": "DistrictPollutionExtract"
  },
  "DistrictPollutionExtract": {
    "Type": "Task",
    "Resource": "${districtPollutionLambdaArn}",
    "Next": "IndustryPollutionExtract"
  },
  "IndustryPollutionExtract": {
    "Type": "Task",
    "Resource": "${industryPollutionLambdaArn}",
    "Next": "CleanupData"
  },
  "CleanupData": {
    "Type": "Task",
    "Resource": "${cleanupDataLambdaArn}",
    "End": true
  }
}
```

Slika 6. *Kreiranje stanja u state mašini*

Sve lambda funkcije za obradu podataka funkcionišu na sličan način. Vrši se pristup DynamoDB bazi i ulaznom fajlu iz kojih se učitavaju potrebni podaci. Slika 7 prikazuje učitavanje podataka iz baze uz korišćenje boto3 biblioteke. Primenjuje se potrebna logika i vrši se generisanje rezultata i njihovo čuvanje u S3 korpi.

```
response = dynamodb_client.query(
        TableName=TABLE_NAME,
        KeyConditionExpression='input_id = :input_id',
        IndexName="input_id_gsi",
        ExpressionAttributeValues={
            ':input_id': {'S': input_id}
        }
    )
```

Slika 7. *Čitanje podataka iz DynamoDB baze*

Preuzimanje podataka je implementirano uz korišćenje prethodno potpisanog URL-a koji se dostavlja na *frontend* stranu aplikacije. Generisanje URL-a je predstavljeno na slici 8. Aplikacija vrši preuzimanje podataka sa pomenutog URL-a i njihov prikaz u tabeli.

```
s3_client = boto3.client('s3')
presigned_url =
s3_client.generate_presigned_url('get_object',
Params={'Bucket': bucket, 'Key': file_name},
ExpiresIn=60)
return
{
   "body": presigned_url,
   "statusCode": 200
}
```

Slika 8. *Generisanje ranije potpisanog URL-a*

5. ZAKLJUČAK

Motiv za implementaciju ovog rada bilo je kreiranje rešenja koje će korisniku obezbediti lak pristup željenim podacima. Ideja je da se otkloni potreba poznavanja formata ulaznih podataka od strane korisnika, kao i potreba za poznavanjem i umećem rada u alatima koji mogu da obezbede slična rešenja. Korisničke akcije su svedene na minimum, a fokus je stavljen na izvršavanje svih transformacija podataka u pozadini i dostavljanje korisniku gotovog rešenja.

Rad bi se mogao unaprediti postavljanjem veb stranice na domen koji se nalazi na *cloud* infrastrukturi. Dalje proširivanje je moguće uz definisanje novih načina obrade podataka i njihovu implementaciju i prikaz na korisničkoj aplikaciji.

6. LITERATURA

[1] Portal otvorenih podataka https://data.gov.rs/sr/ (pristupljeno u oktobru 2022.)
[2] What is AWS, https://aws.amazon.com/what-is-aws/ (pristupljeno u oktobru 2022.)
[3] AWS STep functions, https://aws.amazon.com/step-functions/ (pristupljeno u oktobru 2022.)
[4] AWS Dynamo DB, https://aws.amazon.com/dynamodb/ (pristupljeno u oktobru 2022.)
[5] AWS Simple Storage Service, https://aws.amazon.com/s3/ (pristupljeno u oktobru 2022.)
[6] AWS Lambda functions, https://aws.amazon.com/lambda/ (pristupljeno u oktobru 2022.)
[7] AWS API Gateway, https://aws.amazon.com/api-gateway/ (pristupljeno u oktobru 2022.)
[8] AWS Serverless framework, https://www.serverless.com/ (pristupljeno u oktobru 2022.)
[9] Boto3, https://boto3.amazonaws.com/v1/documentation/api/latest/index.html (pristupljeno u oktobru 2022.)

Kratka biografija:

Nikolina Stamenić rođena je u Sarajevu 1996. god. Godine 2015 upisla je Fakultet tehničkih nauka u Novom Sadu, smer Računarstvo i automatika. 2019 godine diplomirala je na osnovnim studijama na Fakultetu tehničkih nauka i iste godine upisuje master studije na smeru Računarstvo i automatika.