Зборник радова Факултета техничких наука, Нови Сад

UDK: 4.4
DOI: https://doi.org/10.24867/30BE27Simic

РАЗВОЈ АРХИТЕКТУРЕ БЕЗ СЕРВЕРА УПОТРЕБОМ ПЛАТФОРМЕ АМАЗОН ВЕБ СЕРВИСА

DEVELOPMENT OF A SERVERLESS ARHITECTURE USING THE AMAZON WEB SERVICES PLATFORM

Јован Симић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Рад се бави развојем апликације за резервисање смештаја употребом serverless модела тј. архитектуре без сервера. Поред три главна модела услуга (IaaS, PaaS, SaaS) обрађени су и све популарнији концепти serverless и FaaS услуга. Задатак рада је да представи једно од могућих архитектурних решења користећи предности услуга које нам облак пружа и тиме смањи недостатке традиционалних система. Описан је начин имплементације и како се ресурси на Амазоновом облаку (Amazon Web Services) креирају уз помоћ IaC (Infrastructure as Code) концепта.*

Кључне речи: *serverless, AWS, FaaS, IaC, CDK*

Abstract – *This thesis addresses the development of an accommodation reservation application using the serverless model. In addition to the three main service models (IaaS, PaaS, SaaS), the increasingly popular concepts of serverless and FaaS (Function as a Service) services are also covered. The task of the paper is to present one of the possible architectural solutions by utilizing the advantages of cloud services, thereby reducing the drawbacks of traditional systems. The method of implementation is described, as well as how resources on Amazon's cloud (Amazon Web Services) are created using the IaC (Infrastructure as Code) concept.*

Keywords*: serverless, AWS, FaaS, IaC, CDK*

1. УВОД

Традиционални приступ развијања апликација подразумева управљање и контролу рачунарским ресурсима у оквиру физичких центара организације. Сва опрема је одговорност организације, потребно ју је обезбедити унапред, конфигурисати, водити рачуна о ажурирању компоненти што изискује додатни напор и време [1]. Такође је неопходно размотрити и редудантност у случају покретања критичних апликација, осигуравајући бизнис функционалности увек доступним, уз одређене стратегије обнове после катастрофе (енгл. *Disaster Recovery*).

Све претходно наведено је створило добар темељ за развој рачунарства у облаку (енгл. *Cloud Computing*), који је донео револуцију омогућавајући компанијама да складиште податке и користе ресурсе (сервери, мрежа, софтверске апликације) путем удаљених центара који се налазе изван њихове инфраструктуре. Оно што нам доноси јесу већа прилагодљивост, скалабилност и смањење трошкова јер су ресурси постали доступни као услуга - *IaaS, PaaS, SaaS*, а не као физички хардвер. Додавање тј. уклањање ресурса и неопходних сервиса на захтев обезбеђује већу флексибилност у постављању жељене инфраструктуре или мењању постојеће, скалирајући потребе бизниса уз минималне напоре [2].

Овај рад се бави демонстрацијом имплементације serverless решења употребом платформе Амазон Веб Сервиса на примеру резервација смештаја.

2. ТЕОРИЈСКИ КОНЦЕПТИ

2.1. Историја рачунарства у облаку

Развој рачунарства у облаку започео је у шездесетим годинама прошлог века, када су усвојени основни концепти који дефинишу облак**.** Значајни напредак у имплементацији облака десио се 1970. године када је компанија IBM представила своју прву виртуелну машину - VM/370. У 1980. годинама, са брзим развојем рачунарских технологија, индустрије и компаније су тражиле начине да повежу своје рачунаре и омогуће приступ дељеним подацима. Појава виртуелних приватних мрежа (енгл. *Virtual Private Network – VPN*) омогућила је ову комуникацију и сарађивање. Појава Amazon Web Services (AWS) на почетку 21. века представља важан моменат у развоју рачунарства у облаку, отварајући пут за његову широку примену.

2.2. Инфраструктура као сервис

Инфраструктура као сервис (енгл. *Infrastructure as a Service - IaaS*) представља сервисе на мрежи који обезбеђују високо апстрактне програме, ослоњене на инфраструктуру која се налази испод површине. Овај модел апстрахује детаље инфраструктуре, укључујући физичке ресурсе, партиционисање података, скалирање и прављење резервних копија (енгл. *backup*). Корисници омогућеним приступом на захтев добијају инфраструктуру коју су закупили код провајдера, укључујући сервере, меморију за складиштење и мрежне ресурсе. Инфраструктура као сервис омогућава већу флексибилност и скалабилност у поређењу са традиционалним приступом [3].

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Жељко Вуковић, доцент.

2.3. Платформа као сервис

Платформа као сервис (енгл. *Platform as a Service* – PaaS) омогућава корисницима да развијају и креирају апликације у облаку користећи програмске језике, библиотеке и софтверске алате које пружа провајдер [4]. У овом моделу, корисници не управљају инфраструктуром, већ имају контролу над развијеним апликацијама и могућности за подешавање окружења.

2.4. Софтвер као сервис

На највишем нивоу апстракције, модел софтвер као сервис (енгл. *Software as a Service* – SaaS) омогућава корисницима да интерагују са већ готовим апликацијама које нуди провајдер, обично уз месечну надокнаду. Вендор управља апликацијом и свим аспектима испод ње, а оне су доступне као десктоп верзије, преко веб претраживача или мобилног телефона.

2.5. Serverless рачунарство и архитектуре

Serverless рачунарство представља нови модел у облаку, у коме провајдери динамички алоцирају ресурсе на захтев корисника, преузимајући одговорност за одржавање сервера. Иако сама реч "serverless" може деловати конфузно и сугерише да сервери не постоје, у стварности они и даље постоје, али су корисницима невидљиви и не захтевају управљање, конфигурисање или скалирање.

Овај модел наплате заснива се на коришћењу ресурса (енгл. *pay-as-you-go*), те корисници плаћају само онолико колико користе, без фиксних или унапред дефинисаних трошкова. Овакво апстраховање инфраструктуре омогућава развојним тимовима да се фокусирају на код и логику апликације [5].

2.6. Функција као сервис

Функција као сервис представља подскуп serverless модела. Ова парадигма се најчешће примењује у рачунарству вођеног догађајима, где се функција покреће када се деси одређени догађај, као што је пристигла порука у реду или HTTP захтев. Када се функција развије, њено скалирање на горе и доле се врши аутоматски.

Током периода када нема захтева, сервер се гаси и ослобађа ресурсе за друге функције. Као и код других serverless имплементација, провајдер не наплаћује услуге када функција није активна [6].

2.7. Амазон веб сервиси

Амазон веб сервиси (енгл. *Amazon Web Services – AWS*) представљају једну од најпопуларнијих платформи за рачунарство у облаку, развијену под брендом компаније Амазон, која обухвата разноврсне услуге које подржавају претходно поменуте моделе облака – IaaS, PaaS, SaaS.

Ове услуге су представљене кроз различите сервисе за складиштење и обраду података, изнајмљивање рачунарских ресурса, управљање мрежним компонентама, безбедност, аналитику, вештачку интелигенцију, управљање великим обимима података (Big Data) и друго [7].

3. ПРЕГЛЕД НАЈЗНАЧАЈНИЈИХ АМАЗОН ВЕБ СЕРВИСА

3.1. Amazon Simple Storage Service (S3)

S3 представља један од првих сервиса који су били доступни јавности. Као што сам назив сервиса каже, главна сврха је складиштење објеката у такозване бакете (енгл. *bucket*), који морају имати глобално јединствен назив, независно од налога на ком је креиран. Различити су случајеви коришћења S3 сервиса – на пример, за језера података (енгл. *data lakes*), архивирање, чување резервних копија, објављивање садржаја статичких веб сајтова, опоравак од катастрофе,…

3.2. AWS Lambda

Једна од првих асоцијација када се говори о serverless решењима на AWS платформи, реч је о познатом Lambda сервису. Функције које се окидају по потреби, без обавезе управљања серверима или провизионисања истих. Скалирају се аутоматски спрам пристижућих захтева, али их карактерише кратко извршавање, ограничено временом.

3.3. Amazon DynamoDB

DynamoDB представља serverless NoSQL базу података, без потребе за одржавањем, високо доступна са копирањем у вишеструким зонама доступности. Дизајниран је да подржи огромна оптерећења скалирајући се аутоматски колико је потребно. Подржава милионе захтева, трилионе редова и стотине терабајта складишта. Нема потребе за креирањем базе података, она је већ ту и спремна за употребу, све што је потребно је креирати тебелу, што је и карактеристично за DynamoDB. Свака табела има примарни кључ који треба да се одреди у време креирања табеле. Подаци се чувају у табели као ставке (енгл. *items*).

3.4. Simple Queue Service (SQS)

SQS је један од најстаријих Амазон веб сервиса, чије су услуге постале јавно доступне и на располагању корисницима још од 2004. године [8]. Користи се да би се апликације раздвојиле и нема ограниченост кад је у питању скалабилност. Испорука порука је гарантована бар једанпут (енгл. *at least once delivery*), али могу се појавити дупликати. Такође није гарантовано да ће поруке бити сортиране по редоследу (енгл. *best effort ordering*).

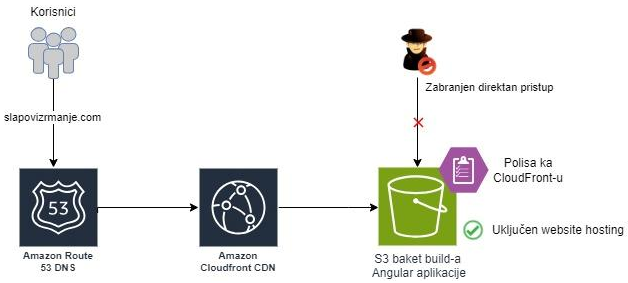
3.5. Route 53

Route 53 представља високо доступан и скалабилан сервис који се бави именима домена (енгл. *Domain Name System – DNS*). Преусмерава крајње кориснике до циљане дестинације тј. жељеног ресурса. Да би ово било могуће, људски читљиве називе као на пример *www.example.com* је потребно превести у адресу Интернет протокола (енгл. *Internel Protocol Address – IP address*) [9]. Ово није једина карактеристика Route 53-а. Могуће је регистровати називе домена за вашу веб апликацију, као и проверити доступност ваших ресурса.

4. НАМЕНА СИСТЕМА И АРХИТЕКТУРА РЕШЕЊА

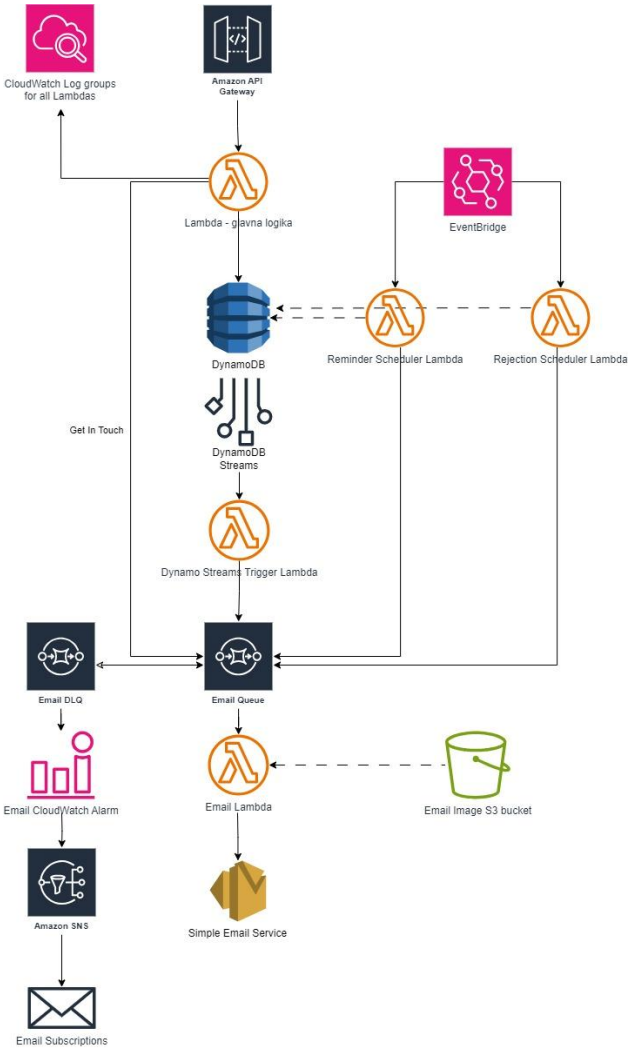
Намена постојања овог система јесте пре свега да олакша заказивање термина смештаја у ресторану “Слапови Зрмање”. До сада није постојало неко аутоматизовано решење. Ни ово не представља потпуно аутоматизовано решење, због захтева и ограничења клијента, али му олакшава интеракцију са корисницима.

На слици 1. налази се дијаграм који описује клијентски део апликације, односно frontend. Апликација је имплементирана коришћењем Angular библиотеке. Код је потребно изградити и затим га поставити на S3 бакет. Потребно је активирати S3 website hosting и изменити полису бакета да дозволи приступ CloudFront-у. Креира се дистрибуција у CloudFront-у где се као извор тј. порекло подеси претходно креирани бакет. Апликација постаје јавно доступна путем изгенерисаног URL-а. За регистрацију домена би се користио Route 53, као и за услуге преусмеравања Интернет саобраћаја.



Слика 1. Frontend део система

Backend део система је приказан дијаграмом архитектуре решења видљивим на слици 2. Улазну тачку у овај део система представља API Gateway, који даље делегира позиве инициране од стране frontend-а ка Lambda функцији. За базу података је коришћен DynamoDB сервис, још један serverless сервис. Да би се омогућио приступ бази, функцији је потребно доделити неопходна права. Уз сјајну комбинацију са DynamoDB Streams, приликом уписивања нових захтева у базу, у приближно реалном времену су догађаји пропагирани активирањем следеће Lambda функције. Ова функција обрађује пристигле догађаје, формирајући SQS поруке које садрже неопходне информације за даљу обраду. За формирање е-поште је задужена посебна Lambda функција која чита поруке са реда, и уз помоћ прослеђених података као и метаподатака формира одговарајући тип и садржај поруке. Порука се потом шаље употребом Simple Email Service-а до жељене дестинације. Уколико дође до грешке приликом слања е-поште услед слабе Интернет конекције, или из неког другог разлога, биће покушано поново захваљујући retry карактеристици SQS-а. Приликом успешности слања, порука се брише са реда. Насупрот, поновним неуспехом из непознатих разлога, порука завршава у специјалном Dead Letter Queue-u. Услед нагомилавања порука из озбиљнијих разлога жељних пажње, када број порука пређе задати праг, активира се аларм који шаље обавештење на Simple Notification Service топик. Потом се претплатницима овог топика, који представља тим подршке, шаље порука путем е-поште.



Слика 2. Дијаграм архитектуре решења backend дела система.

Функционалности које се извршавају једанпут дневно се активирају заказаним радњама Event Bridge-а. Ови догађаји побуђују две посебне Lambda функције. Надлежности ових функција су подсећање клијента на предстојеће заказане термине, као и понуде алтернативних термина корисницима чији су захтеви одбијени. Оне читају филтриране податке из DynamoDB табеле, креирају SQS поруке и даље их шаљу на SQS ред.

5. ИМПЛЕМЕНТАЦИЈА РЕШЕЊА

Клијентска страна је реализована уз помоћ популарног Angular радног оквира. Заснива се на TypeScript језику, бесплатан је и представља single-page апликацију која трчи на Node.js. Апликација је доступна на Интернету захваљујући интеграцији два Амазонова сервиса – Amazon S3 и Amazon CloudFront. Комбинација ових сервиса је изузетно честа и може се рећи да представља стандард за испоруку frontend апликација и статичког садржаја на скалабилан и перформантан начин. У претходном поглављу је

графички приказана и описана међусобна повезаност компоненти и на који то начин се један захтев обрађује од самог почетка проласком кроз API Gateway, па све до чувања ентитета у DynamoDB табели и слања електронске поште. Главна логика ипак је смештена у оквиру Lambda функција уз помоћ Java програмског језика. Шаблон једне Lambda функције доступан је у листингу 1.

```
@Slf4j
@Component
@AllArgsConstructor
public class DynamoStreamTriggerComponent {

  private final AwsService awsService;
  private final ObjectMapper objectMapper;
  private final AccommodationMapper
accommodationMapper;

  public Function<DynamodbEvent,
DynamodbEvent> handleDynamoStreamEvent() {
    return dynamodbEvent -> {
       // The remaining code goes here
      return dynamodbEvent;
    };
  }
}
```

Листинг 1. Метода која се извршава при активацији Lambda функције.

AWS Cloud Development Kit - CDK представља радни оквир за дефинисање инфраструктурних ресурса користећи познате програмске језике, од који су тренутно подржани: TypeScript, JavaScript, Python, Java, C# и Go [10]. Постављање AWS инфраструктуре је подржано уз помоћ овог алата.

5. ЗАКЉУЧАК

Рад се бавио применом концепата рачунарства у облаку и serverless технологије као део њега**.** Реализација овог решења је постигнута интеграцијом различитих сервиса које пружа Amazon Web Services платформа. Главне предности система јесу ефикасност трошкова и оптимизација ресурса. Преласком на модел плаћања на основу потрошње може се лако оптимизовати алокација ресурса и смањити оперативни трошкови. Ова финансијска флексибилност подстиче инжењере да експериментишу без терета великих почетних инвестиција.

Са друге стране, све већа зависност од облака доноси и нове изазове у погледу безбедности и приватности. Поред тога, регулаторни оквири и законодавство ће се морати прилагођавати како би пратили брзи развој технологије и осигурали адекватну заштиту корисника. Пројекат описан у раду је само један од примера трансформације традиционалних архитектурних решења под утицајем рачунарства у облаку.

ЛИТЕРАТУРА

[1] „Traditional IT Deployment Models,“ [На мрежи]. Available: https://www.flackbox.com/traditional-it-deployment-models-on-prem-colo. [Последњи приступ May 2024].

[2] „What is cloud computing?,“ [На мрежи]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing. [Последњи приступ May 2024].

[3] „What are Iaas, Paas and Saas?,“ [На мрежи]. Available: https://www.ibm.com/topics/iaas-paas-saas. [Последњи приступ May 2024].

[4] P. Mell и T. Grance, „The NIST Definition of Cloud Computing,“ National Institute of Standards and Technology: U.S. Department of Commerce, 2011.

[5] R. Miller, „AWS Lambda Makes Serverless Applications A Reality,“ *TechCrunch*, 2015.

[6] B. King, „What is FaaS? Function as a Service explained,“ 1 February 2022. [На мрежи]. Available: https://www.digitalocean.com/blog/what-is-faas-function-as-a-service-explained. [Последњи приступ May 2024].

[7] N. Barney, „Amazon Web Services,“ [На мрежи]. Available: https://www.techtarget.com/searchaws/definition/Amazon-Web-Services. [Последњи приступ May 2024].

[8] „A History of Amazon Web Services (AWS),“ [На мрежи]. Available: https://www.awsgeek.com/AWS-History/. [Последњи приступ May 2024].

[9] „What is an IP Address?,“ 5 September 2023. [На мрежи]. Available: https://www.geeksforgeeks.org/what-is-an-ip-address/. [Последњи приступ May 2024].

[10] „What is the AWS CDK?,“ [На мрежи]. Available: https://docs.aws.amazon.com/cdk/v2/guide/home.html. [Последњи приступ May 2024].

Кратка биографија:

Јован Симић рођен је 30. јануара 1999. године у Новом Саду. Факултет техничких наука у Новом Саду, студијски програм Софтверско инжењерство и информационе технологије уписао је 2018. године. Након завршених основних студија, 2022. године, уписао је мастер академске студије из исте области на смеру Електронско пословање.

контакт: jovansimic995@gmail.com