Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/27BE10Miletic

ПРИМЕНА АЛАТА И ПРАКСИ АУТОМАТИЗАЦИЈЕ У DEVOPS МЕТОДОЛОГИЈИ

USAGE OF AUTOMATION TOOLS AND PRACTICES IN DEVOPS METHODOLOGY

Лука Милетић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *Задатак рада представља анализу DevOps методологије и употребу алата за аутоматизацију, контејнеризацију и пружалаца услуга у облаку. Решавани проблем представља аутоматизацију развоја и испоруке апликације дигиталне библиотеке. Проблем је решен коришћењем Terraform алата за инфраструктуру, Docker алата за контејнеризацију апликације и AWS услуга у облаку. Резултат решења представља успешно креирану инфраструктуру у облаку, аутоматски процес интеграције и испоруке софтверског решења и приступ апликацији од стране крајњих корисника.*

Кључне речи: *DevOps, континуирана интеграција и испорука, контејнери, AWS, Terraform*

Abstract – *The paper involves an analysis of the DevOps methodology and the use of automation tools, containerization, and cloud service providers. The problem pertains to automating the development and delivery of a digital library application. The problem is solved using Terraform infrastructure as code tool, application containerization with Docker, and cloud services from AWS. The result of the solution represents the successful creation of cloud infrastructure, an automated process of software solution integration and delivery, and end-user access to the application.*

Keywords: *DevOps, continuous integration and deployment, containers, AWS, Terraform*

1. УВОД

У свету развоја софтвера, DevOps (Development Operations) је трансформативна парадигма која омогућава тимовима да интегришу и спајају свој развојни рад често и аутоматизовано. Ова промена у начину развоја софтвера подразумева сарадњу, агилност и аутоматизацију између тимова за развој и операције. DevOps нуди алате за аутоматизацију и користи услуге у облаку како би убрзао и поједноставио процес развоја. Ово доводи до бржег пласмана на тржиште, вишег квалитета и ефикаснијег управљања трошковима. Примена методологија континуираних интеграција и испоруке (CI/CD) постала је стандард у индустрији. Ова методологија омогућава брз одговор на промене на тржишту и бржу испоруку софтверских производа. У овом раду, истражујем се улога DevOps пракси, аутоматизације и облачних услуга у развоју софтвера и њихова примена на реалном примеру.

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

2. DEVOPS

Постоји погрешно разумевање развоја софтвера, где се сматра да се оно састоји само из тимова који искључиво производе софтвер. Такав софтвер се не интегрише у потпуности са захтевима данашњих система. Оперативни кадар представља кључ решења овог проблема, који се састоји од способних инжењера, специјализованих за посматрање датог система као јединствену апстрактну јединицу. Они размишљају о захтевима које системи испуњавају, креирају инфраструктуру око датог система, имплементирају безбедносне механизме, постављају протоколе за опоравак од катастрофе и избацују редундантност у задацима развијања софтвера. Решење датог проблема се појавило у виду области софтверског инжењерства, које представља спрегу између оперативних тимова и тимова задужених за развијање софтвера *[1]*.

Резултати компанија које су усвојиле DevOps праксе су невероватне. На пример, компанија Nordstorm је након усвајања ових пракси успела да повећа број функционалности испоручених на месечном нивоу за 100%, смањи дефекте за 50%, смањи време од смишљања идеје до покренутог кода у продукцији за 60% и смањи број продукционих инцидената за 60% до 90%. Након усвајања DevOps пракси у Laserjet Firmware дивизији компаније HP, програмерско време проведено на развијању нових функционалности се повећало са 5% на 40%, док су целокупни трошкови развијања смањени за 40%. Etsy је искористио DevOps праксе да пређе са стресних, нередовних испорука, на испоручивање софтвера 25 до 50 пута дневно *[2]*.

2.1. Континуирана интеграција

Континуирана интеграција (CI) представља праксу развоја софтвера у којој се измене у коду интегришу у заједнички репозиторијум кода на редовној основи. Главни циљ ове методологије је откривање и решавање интеграционих проблема и грешака у раним фазама развоја софтвера. Централни принципи и компоненте CI укључују: аутоматизовану изградњу и тестирање, честу интеграцију измена у коду, системе за контролу верзија, тренутну повратну информацију, изолацију грешака, одржавање конзистентности кода, генерисање документације и цевовод испоруке. Ови принципи помажу у убрзавању развоја софтвера и осигуравању квалитета кода, чиме се подстиче ефикасност и успешност развојних пројеката *[3]*.

2.2. Континуирана испорука

Континуирана испорука (CD) представља стратегију развоја софтвера која аутоматизује процес објављи-

вања софтверских измена корисницима. Главни принципи укључују аутоматизацију, честа објављивања, континуално тестирање, праћење и повратне информације, повратак и напредак, сарадњу и смањење ризика. Ово омогућава брзе и контролисане објаве измена, што доприноси квалитету и способности за адаптацију на измене захтева и повратне информације корисника *[3]*.

2.3. Алати за управљање конфигурацијом

Алати за управљање конфигурацијом, као што су Chef, Puppet и Ansible, играју важну улогу у инсталацији и управљању софтвером на серверима. Постоје два приступа у дефинисању и управљању инфраструктурним ресурсима: декларативна и императивна конфигурација. У декларативној конфигурацији се спецификује жељено стање инфраструктуре, а алат сам одлучује како да га постигне и остаје идемпотентан. Императивна конфигурација дефинише инструкције корак по корак и обично захтева од аутора конфигурације да обезбеди идемпотентност.

2.4. Алати за креирање контејнера и виртуелних окружења

Алати као што су Docker, Packer и Vagrant постају суштински у DevOps праксама. Они служе за креирање слика сервера које енкапсулирају комплетан оперативни систем, софтвер и друге релевантне детаље. Постоје две главне категорије алата за рад са сликама: виртуелне машине и контејнери. Виртуелне машине емулирају комплетан рачунарски систем, обезбеђујући потпуну изолацију од домаћина, али са великим трошковима у процесорској моћи и меморији. Насупрот томе, контејнери емулирају кориснички простор оперативног система и обезбеђују брже покретање и мање захтеве за ресурсима, али са нижим нивоом изолације и безбедности. Docker и CoreOS rkt су алати за дефинисање контејнерских слика.

2.5. Алати за оркестрацију контејнера

Управљање виртуелним машинама и контејнерима представља кључни аспект инфраструктуре, а за обављање различитих задатака у вези са њима неопходни су одређени алати и стратегије. Ови задаци укључују испоруку нових инстанци, ажурирање, праћење здравља, скалирање, балансирање оптерећења и могућност комуникације међу њима. Управљање свим овим аспектима обично се постиже коришћењем алата за оркестрацију као што су Kubernetes, Marathon/Mesos, Amazon Elastic Container Service (Amazon ECS), Docker Swarm и Nomad.

2.6. Алати за креирање инфраструктуре

Алати за управљање конфигурацијом служе за дефинисање кода који треба извршити на серверима, док алати за креирање инфраструктуре обављају задатак креирања различитих инфраструктурних ресурса. Terraform се истиче као велики играч са широком подршком за облачне услуге и декларативним приступом. Pulumi омогућава флексибилност у избору програмског језика, док је CloudFormation највише везан за AWS и користи YAML за конфигурацију. Terraform се наглашава као најпоузданији избор за аутоматизацију инфраструктуре, Pulumi за бржи почетак и флексибилност у избору језика, док је CloudFormation прикладан за AWS кориснике који не планирају миграцију на друге пружаоце услуга у облаку *[4]*.

3. АРХИТЕКТУРА ИНФРАСТРУКТУРНОГ РЕШЕЊА

Ово поглавље се бави архитектуром инфраструктурног решења апликације дигиталне библиотеке. Најпре ће бити речи о AWS пружаоцу услуга у облаку, затим ће бити описане коришћене компоненте и услуге. На крају је објашњено како ове компоненте функционишу заједно, формирајући инфраструктурно решење.

3.1. AWS пружалац услуга у облаку

Рачунарство у облаку има различите дефиниције, али се у основи односи на пружање апликација и података на удаљеним облацима, обезбеђујући лак и флексибилан приступ. Облаци се деле на приватне, јавне и хибридне, у зависности од окружења, и на услуге као инфраструктуру (IaaS), платформу (PaaS) и софтвер (SaaS). AWS, као водећи провајдер рачунарства у облаку, истиче се по својим услугама које наглашавају приватност, интегритет и доступност података корисника. AWS омогућава лак приступ ресурсима и скалирање потреба корисника, што доприноси ефикасном развоју и оптимизацији трошкова.

У наставку ће бити описане специфичне AWS компоненте и услуге, коришћене за инфраструктурно решење.

VPC - Amazon виртуелни приватни облак је кључни сервис који омогућава корисницима да креирају безбедна и изолована мрежна окружења за своје AWS ресурсе. VPC доноси више предности, укључујући изолацију, детаљну контролу мрежних поставки, висок ниво сигурности, опције за повезивање са локалном инфраструктуром, контролу приступа интернету, скалабилност и могућност дизајнирања подмрежа.

API Gateway - Amazon API Gateway представља моћан и комплексан сервис који олакшава креирање, управљање и испоруку апликативних програмских интерфејса (API-ја) у AWS облаку. Он нуди бројне функционалности, укључујући креирање и управљање API-јима, безбедност, контролу саобраћаја, мониторинг и интеграцију са другим AWS услугама.

Application Load Balancer - Amazon Application Load Balancer (ALB) је услуга која омогућава ефикасно балансирање и рутирање долазног саобраћаја апликације на различите мете. Ова услуга ради на апликативном слоју и нуди низ функционалности које укључују: интелигентно усмеравање саобраћаја, груписање мета, интеграцију са контекстом контејнера, безбедносне функције и скалабилност.

Elastic Container Service - Amazon Elastic Container Service (Amazon ECS) је AWS услуга за управљање контејнерима која олакшава развој и извршавање Docker контејнера. Она пружа функционалности за управљање контејнерима и кластерима, дефиницију задатака за апликације, балансирање оптерећења, аутоматско скалирање, интеграцију са AWS Fargate и услугама за ауторизацију. Amazon ECS омогућава

развојним и операционим тимовима да управљају апликацијама у контејнерима без потребе за директним управљањем инфраструктуром.

CloudWatch - Amazon CloudWatch представља моћну AWS услугу за надгледање и управљање ресурсима и апликацијама у реалном времену. Ова услуга омогућава прикупљање и складиштење информација о перформансама, мониторинг у реалном времену, постављање аларма, праћење логова и интеграцију са другим AWS услугама. CloudWatch пружа корисницима детаљан увид у рад њихових AWS окружења и обезбеђује механизме за реаговање на догађаје и трендове.

Relational Database Service - Amazon Relational Database Service (Amazon RDS) је услуга која олакшава управљање и ширење релационих база података на AWS облаку. Она нуди избор између различитих драјвера база података и пружа високу расположивост, аутоматске бекапе и безбедносне функције. Корисници могу лако пратити перформансе и ширити своје базе података, а Amazon RDS се бави рутинским задацима одржавања, омогућавајући им да се фокусирају на развој апликација.

Elastic Container Registry - Amazon Elastic Container Registry (Amazon ECR) је услуга за регистровање контејнера која обезбеђује безбедан и скалабилан репозиторијум за слике Docker контејнера. Интегрише се са AWS услугама, омогућава управљање животним циклусом слика и пружа контролу приступа и безбедност. Корисници плаћају само за коришћени простор и пренос података, следећи модел плаћања по коришћењу

CodePipeline - AWS CodePipeline је услуга за континуирану интеграцију и континуирану испоруку (CI/CD) која аутоматизује различите фазе изградње, тестирања и испоруке софтверских издања. Она обезбеђује развојним тимовима могућност да ефикасно и безбедно испоручују измене кода. Кључне карактеристике ове услуге укључују: креирање цевовода за аутоматизацију процеса, интеграцију са AWS и екстерним услугама, управљање артефактима, визуелни радни ток, аутоматизовано тестирање, паралелно и секвенцијално извршавање фаза, окидаче засноване на догађајима и фокус на безбедности и скалабилности. Ова услуга игра кључну улогу у убрзавању развоја и испоруке софтверских решења на AWS платформи.

3.2. Docker

Docker је платформа отвореног кода за контејнеризацију, која обезбеђује паковање и извршавање апликација у контејнерима са низом предности. Овај алат се истиче кроз контејнеризацију која обезбеђује конзистентно окружење за апликације, једноставан кориснички интерфејс, брзину и ефикасност, портабилност за различите платформе, лаку скалабилност, активну заједницу и интеграцију са другим технологијама. Docker омогућава ефикасан развој, тестирање и испоруку апликација и представља важан алат у свету развоја и дистрибуције софтвера.

3.3. GitHub

GitHub је моћна веб-базирана платформа за управљање верзијама и колаборативни развој софтвера која нуди бројне корисне функције. Користи Git за контролу верзија и омогућава програмерима да сарађују на пројектима, прате измене кода и раде на заједничким решењима. Корисници могу креирати јавне или приватне репозиторијуме за управљање кодом, а такође имају приступ функцијама за тимску сарадњу, континуалну интеграцију, документацију, безбедност и управљање пројектима. GitHub се истиче кроз своју активну заједницу и могућност проширивости, што га чини неизоставним алатом за програмере и развојне тимове.

3.4. Инфраструктурно решење

Слика *1* приказује архитектуру инфраструктурног решења, састојано од претходно описаних услуга и ресурса.

Инфраструктура се базира на AWS виртуелном приватном облаку и има за циљ да осигура изолацију и сигурност ресурса. Корисници могу приступити систему и комуницирати са њим само преко API Gateway-а, који се налази у јавној подмрежи. Ова комуникација се одвија сигурно коришћењем HTTPS протокола, што обезбеђује поверљивост и интегритет података између корисника и API Gateway-а. Следећа компонента је ALB који се налази унутар приватне мреже и служи за прослеђивање захтева. Ова компонента дозвољава приступ само од стране API Gateway-а, чиме се обезбеђује безбедност и контрола приступа.

Комуникација се извршава коришћењем HTTP протокола, обзиром да су све остале компоненте система унутар приватне мреже. ALB равномерно распоређује захтеве на ECS контејнере, у зависности од мрежног оптерећења контејнера. ECS користи Fargate технологију за покретање контејнера са дефинисаним задацима.

У конкретном случају, дефиниција задатка представља контејнеризовани API дигиталне библиотеке, који се налази у AWS ECR репозиторијуму. Додатно, дефиниција задатка садржи конфигурацију за прикупљање и управљање логовима коришћењем AWS CloudWatch услуге. API дигиталне библиотеке има приступ AWS SQL Server RDS бази података, која чува податке на сигуран начин. За развој апликације коришћена је AWS CodePipeline CI/CD услуга.

Ова услуга је интегрисана са GitHub репозиторијумом, где се налази изворни код апликације. Свака измена на главној грани GitHub репозиторијума покреће аутоматску изградњу, паковање кода у Docker слику и постављање слике на ECR репозиторијум. Након тога, ECS ажурира дефиниције задатака и испоручује контејнере са новом сликом апликације. Једна од битних карактеристика је да је API доступан и за време ажурирања апликације, обзиром да ECS и Fargate гарантују сталан број покренутих контејнера, што омогућава високу доступност система.

4. ИМПЛЕМЕНТАЦИЈА ИНФРАСТРУКТУРНОГ РЕШЕЊА

У овом поглављу ће бити представљена имплементација инфраструктурног решења кроз пример листинга инфраструктурног кода, објашњења одређених Terraform концепата и пример крајњег резултата.

4.1. Makefile датотека

Ради лакшег коришћења Terraform, Docker и AWS интерфејса командне линије употребљен је GNU Make алат. Он служи за дефинисање сопствених команди, правила и мета. На овај начин, покретањем једне команде се могу секвенцијално извршити скупови команди других алата. У њему се налазе лабеле које одговарају Terraform командама за управљање ресурсима. Команда `init` иницијализује окружење, `plan` анализира измене и генерише извештај, док `apply` примењује те измене на инфраструктуру.

Слика 1. *Архитектура инфраструктурног решења апликације дигиталне библиотеке*

4.2. Terraform датотеке

Инфраструктурне Terraform датотеке, са `.tf` екстензијом, служе за дефинисање и управљање ресурсима инфраструктуре у облаку. Свака `.tf` датотека представља компоненту инфраструктуре, користећи Terraform доменски језик HCL. У овим датотекама дефинишу се ресурси, промењиве и излази. Ресурси представљају компоненте инфраструктуре које се креирају или управљају, као што су сервери и мрежни елементи. Промењиве омогућавају параметризацију конфигурације, док излази омогућавају приступ и референцирање вредности ресурса у другим датотекама.

Листинг 1 приказује пример Terraform AWS ALB ресурса.

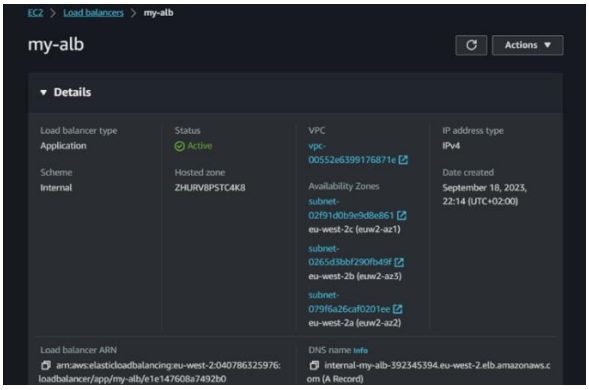
```
resource "aws_lb" "this" {
  name            = "my-alb"
  security_groups =
[aws_security_group.this.id]
  subnets         =
data.aws_subnet_ids.this.ids
  load_balancer_type = "application"
  internal = true
}
```

Листинг 1. *Пример Terraform AWS ALB ресурса*

Слика 2 приказује пример успешно креираног ALB-а.

5. ЗАКЉУЧАК

У овом истраживању се разматра значај DevOps методологије и алата за аутоматизацију у развоју софтвера. Анализирани су DevOps принципи и алати и истакнуте су њихове предности, укључујући побољшану сарадњу и брже циклусе развоја. Такође, објашњена је улога пружалаца услуга у облаку и представљена инфраструктура кодирана помоћу Terraform алата. Резултати укључују инфраструктуру у облаку и аутоматизован процес интеграције и испоруке софтвера.



Слика 2. *Пример успешно креираног ALB-а*

У будућности, инфраструктурно решење има потенцијал за побољшање на неколико начина. Прво, коришћење Terraform модула може значајно олакшати организацију и одржавање инфраструктурног кода, смањујући дупликацију и пружајући стандардизоване интерфејсе за сарадњу између тимова. Друго, употреба специјализованих алата за управљање тајнама може подигнути ниво сигурности и управљивости осетљивих података у Terraform решењу, чинећи их бољим избором за управљање тајнама.

6. ЛИТЕРАТУРА

[1] A. Jahić and N. Buzađija, "DevOps Methodology in Modern Software Development," *qjoest*, vol. 4, no. 1, pp. 1-11, March 2023.

[2] Y. Brikman, Terraform: Up & Running, 3rd ed., Sebastopol: O'Reilly Media Inc., 2022.

[3] M. Shahin, M. Ali Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," *IEEE Access*, vol. 5, pp. 3909-3943, 2017.

[4] Bunnyshell, "Terraform vs. Cloudformation vs. Pulumi," Bunnyshell, 7 July 2022. [Online]. Available: bunnyshell.com/blog/terraform-vs.-cloudformation-vs.-pulumi. [Accessed September 2023].

Кратка биографија:

Лука Милетић рођен је 21.09.1999. у Новом Саду. Студент на мастер академским студијама Факултета техничких наука, у оквиру Универзитета у Новом Саду. Након завршене средње школе, уписује 2018. године Факултет техничких наука, смер Рачунарство и аутоматика. У трећој години се усмерава на модул Примењене рачунарске науке и информатика, са даљим усмерењем на модул Интернет и електронско пословање. Након завршених основних студија, уписује 2022. године мастер академске студије на Факултету техничких наука, смер Рачунарство и аутоматика, модул Електронско пословање.