EFIKASNOST DEVOPS AUTOMATIZACIJE U UPRAVLJANJU INFRASTRUKTUROM KAO KODOM

EFFICIENCY OF DEVOPS AUTOMATION IN MANAGING INFRASTRUCTURE AS CODE

Pregledni znanstveni članak

*Pred. VŠ Samir Ščetić**
*Dr. sc. Bakir Čičak, pred. vš**

Sažetak

Cilj ovog rada jeste objasniti DevOps automatizaciju i kroz praktičan primjer prikazati upotrebu i prednosti korištenja novih tehnologija. Pojam DevOps u zadnje vrijeme postaje „buzzword“ te kao takav postaje loše definiran, podložan krivoj interpretaciji i shvaćanju njegova pravog značenja. Sljedeća definicija se fokusira više na cilj nego na samo značenje. DevOps je skup praksi koje služe kako bi se smanjilo vrijeme između promjene koja se dogodila na sistemu i promjene koja se projavila na samoj produkciji bez gubljenja na kvaliteti i sigurnosti sistemaDevOps integrira dva svijeta, razvoja i operacija koristeći tehnologije za automatizaciju razvoja, impelemntaciju i monitoring infrastrukture. Veliki pomak unutar organizacije koji spaja dva naizgled različita svijeta u jedan, koji zajedničkim djelovanjem ima za cilj da kontinuirano dostavi kvalitetan sistem u što kračem periodu eliminirajući probleme u komunikaciji između timova.

Ključne riječi: DevOps, CI/CD, Docker, IaC, Skalabilnost, Kubernetes

* *Visoka škola „CEPS - Centar za poslovne studije“ Kiseljak, e-mail: samir.scetic@ceps.edu.ba*
* *Visoka škola „CEPS - Centar za poslovne studije“ Kiseljak e-mail: bakir.cicak@ceps.edu.ba*

Abstract

The goal of this paper is to explain DevOps automation and, through a practical example, demonstrate the usage and advantages of employing new technologies. The term DevOps has recently become a buzzword and, as such, has become poorly defined, susceptible to misinterpretation, and misunderstood in its true meaning. The following definition focuses more on the objective than the precise meaning. DevOps is a set of practices aimed at reducing the time between a change that occurs in the system and the manifestation of that change in production without compromising on the quality and security of the system. DevOps integrates two worlds, development and operations, using technologies to automate the development, implementation, and monitoring of infrastructure. It represents a significant shift within an organization that brings together two seemingly different worlds into one, working collaboratively to continuously deliver a quality system in the shortest possible timeframe, eliminating communication problems between teams.

Keywords: DevOps, CI/CD, Docker, IaC, Scalability, Kubernetes.

1. UVOD

Jasno i precizno definisanje pojma DevOps predstavlja izazov, za neke, je to pozicija unutar IT tima koja podrazumijeva poznavanje tehnologija iz oba svijeta razvoja i infrastrukture. Dok drugi su suprotnog mišljenja i smatraju da to nije samo pozicija već kultura i revolucijska organizacija jedne kompanije. Konkretnu definiciju pojma DevOps možemo pronaći u Gartnerovom rječniku koja glasi: „DevOps predstavlja promjenu u IT kulturi, fokusirajući se na brzu isporuku IT usluga kroz usvanjanje agilnih i Lean praksi u kontekstu sistemski orijentiranog pristupa. DevOps naglašava ljude (kulturu) i nastoji poboljšati suradnju između opravitnih i razvojnih timova. U DevOps-u se koriste alati za automatizaciju koji nastoje stvoriti iznimno programabilnu i dinamičnu infrastrukturu iz perspektive DevOps ciklusa“.

2. DEVOPS ALATI I TEHNOLOGIJE

Ključni za uspješnu primjenu DevOpsa su alati i tehnologije koje omogućuju automatizaciju procesa, praćenje performansi i optimizaciju resursa. U nastavku ćemo istražiti četiri ključne kategorije DevOps alata i tehnologija:

- log monitoring
- monitoring
- build i test
- deploymenti i konfiguracija.

2.1. Log monitoring

Ovi dnevni zapisi (logovi) koriste se za analizu performansi sistema. Dakle, podaci iz logova koriste se od strane sistem inženjera i developera u procesu otklanjanja grešaka (debugging). Aplikacija prolazi kroz različite faze DevOps životnog ciklusa (kao što su testiranje, analiza poslovanja, praćenje u produkciji), tada se logovi koriste kao koristan alat. Kada se log podaci primjene u prvoj fazi DevOps životnog ciklusa, logovi služe kao alat za otklanjanje grešaka i takođe se koriste za stres testove i pracenje performansi sistema. Kada se log podaci primjene u drugoj fazi DevOps životnog ciklusa, logovi se koriste za praćenje i rješavanje problema u produkciji. Kada se log podaci primjene u trećoj fazi DevOps životnog ciklusa, logovi se koriste za web analitiku i mjerenje sistemskih metrika. Analizom logova poboljšava se proces osiguranja kvaliteta tako što se brže uočavaju problemi, identifikuju prije nego što postanu kriticni i omogućava se bolja komunikacija između timova. Logovi se smatraju bitnim faktorom prilikom mjerenja uspjeha usluge i rješavanja problema koji se pojave. Međutim, prava skalabilnost i sigurnost često nisu u prvom planu kada se upravlja logovima. Postoji mnogo različitih alata za upravljanje log podacima u okviru DevOps prakse. Neki od njih su open-source alati, dok su neki plaćeni. Na tržištu postoji niz alata za praćenje logova u okviru DevOps prakse. Ovdje u ovom odjeljku su objašnjeni neki od tih alata.

2.1.1. Splunk

Splunk je alat kreiran od američke multinacionalne korporacije koja je smještena u San Francisku, Kalifornija. Splunk je alat za praćenje logova putem veb interfejsa koji omogućava pretragu, praćenje i analizu ogromne količine podataka generisanih od strane mašina. Također, generiše indekse i povezuje ih u stvarnom vremenu iz searchable repozitorija kako bi generirao grafikone, izvještaje, upozorenja i vizualizacije. Splunk je proprietarni softver koji nema dnevna ograničenja za pohranu log podataka, sposoban je čuvati log podatke od 1 dana. Ovi podaci se čuvaju povjerljivo, te korisnika ili klijenta obavještava putem e-pošte kada su pristutni error log podaci. Postoji veliki broj aplikacija koji podržavaju ovaj alat, koristi se i za upravljanje log podacima u cloudu. Splunk podržava opciju pretrage na način da prima input od korisnika i automatski ga analizira prema unesenim podacima.

2.1.2. Logstash

Logstash je centralizovana platforma za agregiranje i upravljanje velike količinom log fajlova. Dakle, Logstash je centralizovani sistem za upravljanje logovima i dio je open source stack-a u kojem se Elasticsearch koristi za indeksiranje i pretragu log podataka, a Kibana se koristi za vizualizaciju podataka. Logstash je open source alat koji ima dnevna ograničenja za čuvanje podataka, zadržava podatke oko 30 dana i korisnika obavještava putem e-pošte kada error log podaci stignu, ili kada istekne vrijeme zadržavanja podataka. Smanjuje hiljade log podataka u značajne informacije na jednoj stranici, te također upravlja logovima u cloudu.

2.2. Alati za monitoring

Alati za monitoring su od velike koristi za sistemske inženjere i developere kako bi osigurali ispravan rad softvera. Alati za monitoring u okviru DevOps-a mogu se klasifikovati kao alati za praćenje sistema i mreže. Alati za praćenje sistema koriste se za praćenje performansi sistema, prikupljanje i čuvanje podataka, kreiranje grafova na osnovu prikupljenih podataka i praćenje resursa sistema. Alati za praćenje mreže mogu se

koristiti za praćenje mreže i njenih resursa, provjeru statusa mrežnih uređaja i u slučaju problema pravovremeno obaveštavanje sistem administratora. Ovi alati omogućavaju razvojnim inženjerima da prate performanse sistema i mreže kako bi se osiguralo da softver ispravno funkcioniše, a takođe omogućavaju praćenje resursa i brzu reakciju u slučaju problema. Alati za praćenje sistema i mreže igraju ključnu ulogu u DevOps praksi.

2.2.1. Graphite

Graphite je open-source alat napisan u Pythonu, koristi za praćenje vrijednosti metrika koje se dinamički mijenjaju. On čuva podatke, generiše grafikone i prati performanse računarskog sistema. Korisnik mora koristiti postojeće alate (kao što su collectD, statsD, Gmond i sl.) ili napisati neke aplikacije za prikupljanje podataka. Ovaj alat se koristi za numeričke vremenske serije podataka., također se može koristiti i u cloud okruženju. Sastoji od tri komponente:

- Carbon: ovo je obrada pozadine koja sluša podatke i može obraditi veliki broj klijenata.
- Whisper: sličan je RRD-u i nudi brzo i pouzdano skladištenje primljenih podataka tokom vremena.
- Graphite Webapp: Django webaplikacija koja će prikazivati grafikone na zahtjev. Kada postoji zahtjev za grafikonom on izvlači podatke s diska, a ako podaci još nisu zapisani na disk, direktno će uzeti podatke iz Carbona kako bi generisao grafikone u stvarnom vremenu.

2.3. Bulid i test

Build i test alati se koriste u automatizaciji uobičajenih zadataka programera, kao što su kompajliranje source koda, kreiranje skripti, pokretanje testova i izrada dokumenata. Adekvatni alati za DevOps imaju za cilj da automatiziraju IT usluge, pruže vizualizaciju performansi sistema i aplikacija u realnom vremenu, te sistem inžinjerima i developerima pruže jedinstveno mjesto – single source of truth. Važnije od sposobnosti pojedinačnog alata je kako se sve to usklađuje sa strateškim ciljevima organizacije. Primjena ovakvih alata su adekvatan način za postizanje benefita DevOps prakse. Jedan o ovih alata je i Jenkins. Jenkins je jedan od

opensource alata za neprekidnu integraciju. Razvijen je nakon nesuglasica s kompanijom Oracle, izdvojen iz projekta Hudson. Osnovna funkcionalnost Jenkins-a je izvođenje predefinirane liste koraka na temelju određenog uslova. Jenkins je jako puno korištena aplikacija koja prati izvođenje ponavljajućih poslova, poput izgradnje softverskog projekta. Također, prati izvođenje koraka i omogućava zaustavljanje procesa ako jedan od koraka ne uspije. Omogućava obavještavanje korisnika o uspješno završenom, odnosno neuspješnom build-u. Jednostavan je za instalaciju, razumijevanje i korištenje. On pruža intuitivan interface za upravljanje projektima putem web browser-a. Može se pokrenuti putem command line interface ili se izvodi na web serveru. Podržava označavanje datoteka i pruža sigurnosnu izvedbu.

3. DEPLOYMENT I KONFIGURACIJA

Osnovni zadatak sistemskih administratora je pomoć kolegama da koriste računare. Sistemski administratori su stručnjaci u svojoj oblasti koji omogućavaju usklađivanje između želja korisnika i računara. Ekspertiza sistemskih administratora očituje se u njihovim izborima hardvera i softvera, kao i u konfiguraciji sistema. Radna okolina se konstantno mijenja, i postoji potreba za pravovremenim ažuriranjem softvera i čestim promjenama konfiguracija. Alati za konfiguraciju i postavljanje (deploy) sistema pružaju različite nivoe automatizacije, ali imaju isti osnovni cilj da olakšaju proces konfiguracije i postavljanja sistema. Puppet alat je sistem za upravljanje konfiguracijom koji omogucava administratorima i inžinjerima da definiraju arhitekturu IT infrastrukture, a zatim, bez ljudske intervencije, provodi konfiguraciju primjenjuje na definisanoj infrastrukturi. Bilo da kontrolirate samo nekoliko servera ili mnogo drugih fizičkih ili virtualnih mašina, ovaj alat automatizira zadatke koje sistem administratori često obavljaju ručno, oslobađajući vrijeme i mentalni prostor kako bi sistem administratori mogli raditi na projektima koji pružaju veću poslovnu vrijednost. Bez obzira da li implementirate aplikacije koje dolaze od klijenta ili radite s internim timom softverskih programera, Puppet automatizira svaki korak procesa isporuke softvera, od opskrbe fizičkim i virtualnim mašinama do orkestracije i izvještavanja, zatim od razvoja koda u ranoj fazi kroz testiranje, puštanje u produkciju i ažuriranja. Puppet osigurava dosljednost, pouzdanost i

stabilnost. Također olakšava bližu suradnju između sistem administratora i programera, omogućavajući učinkovitiju isporuku čišćeg i bolje dizajniranog koda. Još jedan od boljih alata za Deployment i konfiguraciju je Docker. Docker omogućava pakiranje i pokretanje aplikacije u izoliranom okruženju nazvanom kontejner. Ovakav način omogućava da se istovremeno pokrece više kontejnera na istom hostu. Kontejneri su jednostavni, lagani i sadrže sve što je potrebno za pokretanje aplikacije, tako da se ne morate oslanjati na ono što je instalirano na hostu. Kontejneri se mogu koristiti zajednički dok radite u istom okruženju zajedno sa kolegama ili javno i biti sigurni da svi s kojima ih dijelite dobijaju isti kontejner koji radi na isti način. Docker pruža alate i platformu za upravljanje životnim ciklusom vaših kontejnera. Docker podržava razvijanje aplikaciju i njenih zavisnih komponenti pomoću kontejnera. Koristeći ovaj način kontjner postaje jedinica za distribuciju i testiranje vaše aplikacije. Kada ste spremni, implementirajte svoju aplikaciju u produkcijsko okruženje kao kontejner ili na platformama za orkestraciju kontejnera poput Kubernetes ili Openshift. Docker će raditi bez problema bez obzira na to da li je vaše produkcijsko okruženje lokalni data centar, provideri u clodu ili hibridno okruženje. Docker se koristi za:

a) Brzu kontinuiranu isporuku aplikacija
b) Responsivni deployment i skaliranje
c) Pokretanje više servisa na istom hardware-u

4. GIT

Sistem za kontrolu verzija, ili VCS (version control system), prati historiju promjena dok ljudi timovi zajedno rade na projektima. VCS omogucava da developeri stalno prave neke izmjene na projektu, a da se u svakom trenutku može uraditi roll-back na prethodnu verziju projekta. Bez gubljenja na vremenu i performansi sistema. Developeri mogu pregledavati historiju projekta kako bi saznali:

a) Koje promjene su napravljene?
b) Ko je napravio promjene?
c) Kada su promjene napravljene?
d) Zašto su promjene bile potrebne?

VCS-ovi svakom korisniku omogučavaju jedinstven i dosljedan uvid i pregled projekta, prikazujući rad koji je već u toku. Posmatranje transparentne historije promjena, ko ih je napravio i kako doprinose razvoju projekta pomaže članovima tima da ostanu usklađeni dok rade nezavisno. U distribuiranom sistemu za kontrolu verzija, svaki developer ima potpunu kopiju projekta i historije projekta. Za razliku od nekada popularnih centralizovanih sistema za kontrolu verzija, distribuirani sistemi za kontrolu verzija ne zahtijevaju konstantnu vezu sa centralnim repozitorijem. Git je najpopularniji distribuirani sistem za kontrolu verzija. Git se često koristi kako za razvoj opensource koda, tako i za komercijalni softver, s značajnim koristima za pojedince, timove i preduzeća.

5. PRIMJER USPJEŠNE IMPLEMENTACIJE DEVOPS-A U REALNOM SVIJETU

U ovom radu za cilj nam je bio da prikažemo primjer uspješne implementacije DevOps alata i tehnologija u realnom svijetu i kako nam ove prakse maloprije spomenute u radu mogu doprinjeti u svakodnevnom izvršavanju radnih zadataka i obaveza. Na način da nam omogućavaju što bržu i uspješniju isporuku aplikacija i infrastrukture bilo da se radi o testiranju ili produkciji nezavisno od platforme. Za primjer uspješne implementacije DevOps-a u realnom svijetu koristit ćemo opensource kubernetes platformu koja ima za cilj da orkestrira mikroservisnu arhitekturu te nam olakša proces deploymenta kontejniriziranih aplikacija. Projekt koji prezantiram u ovom radu može služiti kao dobar početak i razumijevanje Kubernetes platforme, mikroservisnih aplikacija, kao i same arhitekture aplikacija, kao što su kontejneri, pod-ovi, nod-ovi i ostale komponente koje čine ovaj sistem visoko dostupim i skalabilnim te lakim za test i deploy. Projekat se sastoji od jednostavne baze podataka i web interface-a sa kojim cemo pristupiti toj istoj bazi. Za bazu sam odabrao PostgreSQL jer se u zadnje vrijeme jako često koristi, baza koja je opensource i može se uraditi deploy na Kubernetes/Openshift platformama. Također, postgresql je relaciona baza i koristi sql jezik prilikom upita informacija, nastala je 1986. godine kao dio postgres projekta na Univerzitetu u Kaliforniji i ima više od 35 godina aktivnog razvoja. Za pristup PostgreSQL koristi se alat PgAdmin koji je također opensource, besplatan i koristi se za administraciju baze. On

omogućava korisnicima da lako upravljaju, izvršavaju sql upite, prate performanse i izrađuju tabele. Jako je popularan alat među PostgreSQL korisnicima zbog svoje jednostavnosti te je dostupan za različite operativne sisteme. Ove komponente čine jednu mikroservisnu aplikaciju pokrenutu na Kubernetes platformi. Njihovi deploymenti su pisani u YAML jeziku i za svaku komponentu zasebno, što će se detaljnije obrazložiti u nastavku. Za potrebe projekta instalirana je MiniKube verzija Kubernetes-a koji se jednostavno instalira i ima za ulogu da upravlja pojedinačnim klasterima na serverima ili lokalnim računarima. Koristan alat za programere ili DevOps inžinjere koji žele eskperimentirasi sa Kubernetesom, gdje mogu testirati svoje aplikacije prije nego ih premjeste na produkciono okruženje. Instalaciju MiniKub servera možete pronaći na internetu u zvaničnoj dokumentaciji, za potrebe projekta instalirana je verzija v1.31.2 na Windows 11 operativnom sistemu.

5.1. Pokretanje minikube servera

Izvršava se komandom minikube start u powershellu, te nakon što su se podigli sve potrebne komponente servera instancu kontrolišemo sa kubectl komandom. Zatim se kreira poseban direktorij u kojem se pohranju deployment, direktorij nazivo po želji. Kako bi smo pokrenuli PostgreSQL bazu podataka na minikube serveru, potrebno je da kreiramo deployment, servis i po potrebi omogućimo persistentos podaka kreirajući takozvani PVC. Deploymenti se pišu u YAML jeziku. YAML (YAML Ain't Markup Language) je lagan i lako čitljiv format jezika kojeg koriste developeri i sistem inženjeri za definiranje konfiguracija, postavki i struktura podataka u čitljivom tekstu.

Slika 1: Kreiranje deployment za PostgreSQL bazu podataka:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgres
          image: postgres:10.1
          imagePullPolicy: "IfNotPresent"
          ports:
            - containerPort: 5432
          envFrom:
            - configMapRef:
                name: postgres-config

```

Izvor: Autori

U ovom deploymentu definisano je ime samog deploymenta, te nam je u ovom slučaju potrebna samo jedna replika, ukoliko bi nam bilo potrebno neko skaliranje ili load balancing ovaj broj bi rastao. Zatim label postgres sa kojim ćemo poslije vezati potrebne komponente, i specifikacije kontejnera. U specifikacijama kontejnera naveden je image koji ce se povući sa javno dostupnih repozitorija kao što je docker-hub sa verzijom postgresa 10.1. Definisan je i port koji ce kontejner koristiti 5432 koji je ujedno i defaultni port koji postgresql inače koristi. Definisane su environment varijable koje se nalaze u config mapi pod imenom postgres-config a koje sadrže potrebne informacije kao što su username, password i ime baze.

Slika 2: PostgreSQL Config map-a

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
  labels:
    app: postgres
data:
  POSTGRES_DB: postgresdb
  POSTGRES_USER: admin
  POSTGRES_PASSWORD: psltest

```

Izvor: Autori

Deployment konfiguracijske mape koja sadrži potrebne informacije za podizanje postgresql baze podataka, kao što su u ovom slučaju ime baze podataka, username i password. Ove informacije se mogu nalaziti u deploymentu direktno ali zbog čestih izmjena one su odvojene kako bi se izbjegla kompleksnost ponovnog deploymenta čitave baze zbog promjene jedne od ovih varijabla.

Slika 3: PostgreSQL servis

```yaml
apiVersion: v1
kind: Service
metadata:
  name: postgres # Sets service name
  labels:
    app: postgres # Labels and Selectors
spec:
  type: NodePort # Sets service type
  ports:
    - port: 5432 # Sets port to run the postgres application
  selector:
    app: postgres
```

Izvor: Autori

Deployment servis postgresql baze podataka kreira servis potreban za komunikaciju i mrežnu dostupnost kontejnera. Postoje različite verzije tipova servisa, za potrebe projekta kreiran je NodePort servis koji će mapirati dinamički port host-a na postgresql kontejner port 5432. Iz tog razloga servisu se može pristupiti po ip adresi host-a i dinamicki dodjeljenom portu. Sve potrebne komponente za deployment PostgreSQL baze su kreirane, u ovom slučaju nije potreban PVC ali bitno je napomenuti da ukoliko želimo osigurati perzistentost podataka moramo kreirati i pvc. Kako se ne bi prilikom restarta pod-a ili kontejnera podaci obrisali, što je jako bitno osigurati da se takvo što ne dogodi u produkcijskim okruženjima. Nakon kreiranje potrebnih komponenti, pregleda ispravnosti kod-a i podizanja minikube instance, možemo pristupiti procesu deploymenta postgresql baze podataka na minikube platformi. Komandom kubectl apply izvršavamo naredbu i pratimo da li je deployment uspješno izvršen. Prije pokretanja komande moramo se smjesitit u deployment dikretorij, nakon toga pokrećemo komandu

kubectl apply -f ime_fajla.yaml.

Kada su svi deploymenti uspješno pokrenuti, nakon toga možemo provjeriti njihovu ispravnost i provjeriti da li su kreirani kontejneri, te da li postoje neki error logovi. Na ovaj način je završeno podizanje jedne od komponente sistema, postgresql baza je u listening stanju i očekuje nove konekcije. Kako bi smo se konektovali na bazu i imali bolji uvid u tabele i šeme potrebna je sljedeća komponenta sistema PgAdmin.

Slika 4: PgAdmin deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pgadmin
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pgadmin
  template:
    metadata:
      labels:
        app: pgadmin
    spec:
      containers:
      - env:
        - name: PGADMIN_DEFAULT_EMAIL
          value: user@e-mail.com
        - name: PGADMIN_DEFAULT_PASSWORD
          value: password
        - name: PGADMIN_PORT
          value: "80"
        image: dpage/pgadmin4:3.6
        imagePullPolicy: IfNotPresent
        name: pgadmin
        ports:
        - containerPort: 80
```

Izvor: Autori

U deploymentu pgadmina definisano je ime, te nam je u ovom slučaju potrebna samo jedna replika, ukoliko bi nam bilo potrebno neko skaliranje ili load balancing ovaj broj bi rastao kao i u prethodnom deploymentu postgresql-a. Definisan je i label pgadmin koji je potreban za povezivanje ostalih potrebnih komponenti, i specifikacije kontejnera. U specifikacijama kontejnera naveden je image koji ce se povući sa javno dostupnih

repozitorija kao što je docker-hub sa verzijom pgadmin 4:3.6. Definisan je i port koji ce kontejner koristiti 80 i environment varijable koje sadrže potrebne informacije kao što su email, password i port.

Slika 5: PgAdmin servis

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: pgadmin
  name: pgadmin
  namespace: default
spec:
  ports:
  - name: padmin-port
    nodePort: 30165
    port: 80
    targetPort: 80
  selector:
    app: pgadmin
  type: NodePort
```

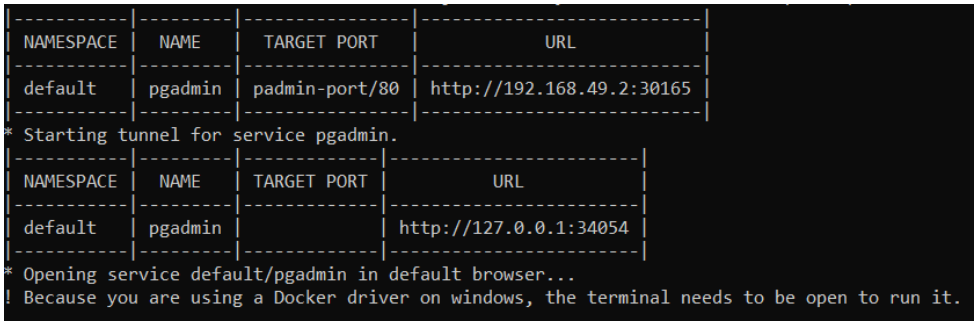
Izvor: Autori

Deployment servis pqadmin-a kreira servis potreban za komunikaciju i mrežnu dostupnost kontejnera. Također, kao i u prethodnom deploymentu servisa za potrebe projekta kreiran je NodePort servis koji će mapirati dinamički port host-a na pqadmin kontejner port 80. Na ovaj način servisu se može pristupiti po ip adresi host-a i dinamicki dodjeljenom portu. Nakon kreiranje deploymenta i servisa možemo pristupiti konačnom deploy-u pgadmin-a na minikube. Kao i ranije, komandom kubectl apply pokrećemo deploy i nekon toga pratimo realizaciju. Također kao i u prethodnom deploy-u prije pokretanja komande moramo se smjesitit u deployment dikretorij, nakon toga pokrećemo komandu

kubectl apply -f ime_fajla.yaml

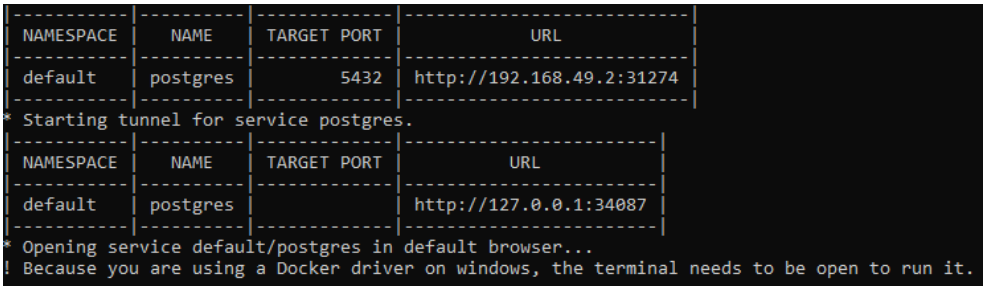
Kada su svi deploymenti uspješno pokrenuti, nakon toga možemo provjeriti njihovu ispravnost i provjeriti da li su kreirani kontejneri, te da li postoje neki error logovi. Sa ovim deployment-om je završeno podizanje zadnje komponente sistema, pgadmin je spreman za spajanje na postgresql bazu. PgAdmin interface-u pristupamo putem web browsera koristeći host ip adresu i dinamicki dodjeljen port. Potrebno je da provjerimo koje dinamčki alocirane portove posjeduju ove dvije komponente sistema. U ovom slučaju postgresql mapira port 5432 na 31274/TCP na host dok pgadmin mapira port 80 na 30165/TCP Servisima se pristupa tako što se pokrene komanda minikube service ime_servisa te se automatski otvara web browser sa pristupom na željeni servis.

Slika 6: Minikube service pgadmin



Izvor: Autori

Slika 7: Minikube service postgres



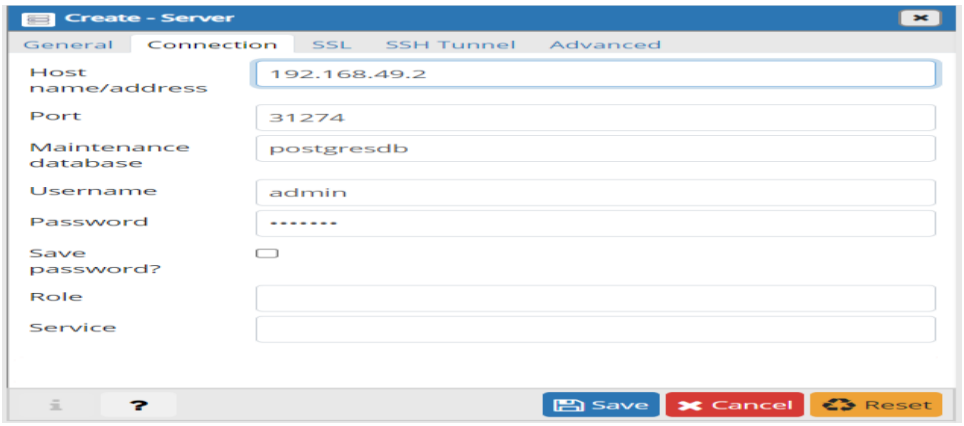
Izvor: Autori

5.2. Konekcija PGAdmin alata sa Postgresql bazom podataka

Sada kada su sve komponente sistema podignute, izvršene potrebne provjere i pokrenut portforwarding servisa na host-u, ostalo je još da se

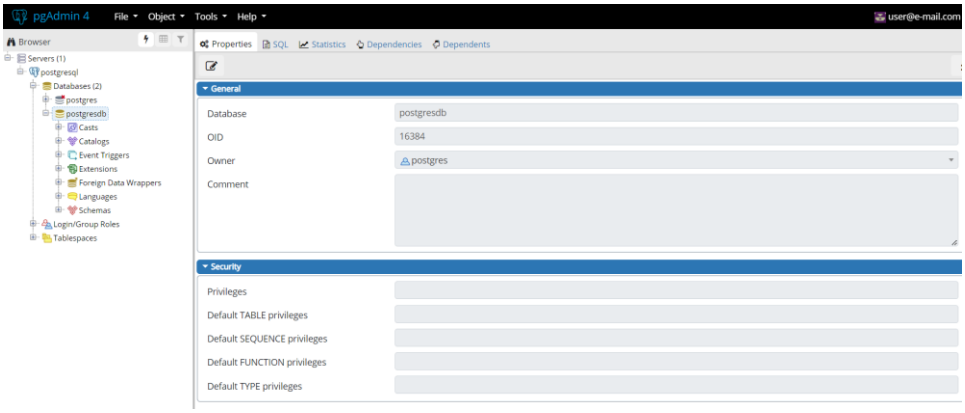
napravi konekcija između pgadmin alata i postgresql baze podataka. Nakon što je pokrenut servis u web browseru pgadmin alatu pristupa se po localhost adresi i dinamičkom portu 34054. U polje „add server“ se upisuju informacije potrebne za pristup postgresql bazi. Informacije ili environment varijable koje se mogu pronaći u konfig mapi.

Slika 8: Pokretanje portforwarding servisa



Izvor: Autori

Slika 9: Kada je konekcija uspostavljena bez grešaka, korisnici mogu pristupiti bazi, kreirati tabele, unositi/izvlačiti podatke i obavljati potrebne zadatke.



Izvor: Autori

Na slici se može vidjeti uspješna konekcija na postgresql bazu koristeći alat pgadmin.

6. ZAKLJUČAK

Cilj rada jeste predstaviti koncept DevOps-a i njegove osnovne komponente, te prikazati njegove prednosti, izazove, principe i prakse koji ga čine tako važnim u današnjem svijetu razvoja softvera. DevOps predstavlja pristup koji ima za cilj da integriše tim razvoja i operacija u jedan kontinuiran proces, koji se fokusira na ubrzanje isporuke softverskih rešenja, poboljšanje kvaliteta i smanjenje rizika u produkcionim sistemima. Kroz analizu historije DevOps-a, koncept se razvio iz potrebe za prevazilaženjem tradicionalnih prepreka u saradnji između razvojnog i operativnog tima. Ovim timovima donosi mnoge prednosti, kao što su brža isporuka, bolja kvaliteta softvera i veća efikasnost, DevOps takođe donosi i neke izazove prilikom primjene u kulturi kompanije, poput potrebe za promjenom u donošenju odluka, tehničkih prepreka i potrebe za pravilnim obukom uposlenih.

DevOps principi, kao što su Flow, Feedback i kontinuirano eksperimentiranje i učenje, predstavljaju temelj za uspješnu implementaciju ovog koncepta koji se potrebno pridržavati konstantno. Osim toga, postoje i mnogi alati i tehnologije koji olakšavaju timovima što jednostavnije obavljanje svakodnevih zadataka, uključujući monitoring, alate za izgradnju, testiranje, deployment i konfiguraciju, kao i cloud okruženja i platforme. Kroz studij slučaja, nastoji se prikazati kako primjena DevOps tehnologija i alata može poslužiti u svakodnenvom poslu, koristeći alate kao što su PostgreSQL, PgAdmin i Kubernetes. Ovaj primjer uspješne implementacije ilustrira praktičnu primenu DevOps-a u svakodnevnom razvoju i održavanju softverskih sistema.

Na kraju, DevOps je bitan faktor u modernom razvoju softvera koji omogućava organizacijama da postignu veću efikasnost, bržu isporuku i bolji kvalitet softvera. Međutim, efikasna primjena zahtjeva posvećenost promjeni kulture, pravilnu obuku i pravilnu upotrebu alata i tehnologija. Sa sve većim značajem DevOps-a u industriji, razumjevanje i primjena ovog koncepta postaje sve bitniji faktor za organizacije koje teže biti konkurentni i inovativni u današnjem digitalnom svijetu.

LITERATURA

1. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations Nicole Forsgren PhD, Jez Humble, Gene Kim IT Revolution, 27. ožu 2018.
2. Gane, K., Kevin, B., George S., 2022. The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win,
3. Gartner, T., 2022. “Gartner Glossary,” Information Technology Glossary,
4. Jens Smeds K. N. I. P., 2022. “DevOps: A Definition and Perceived Adoption Impediments,” Abo Akademi University, pp. 4-5,
5. N. J. S. V. J. V. K. Akshaya H L, A basic Introduction to DevOps Tools, Department of computer science and engineering, VTU, Belgaum University.
6. The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations Gene Kim, Jez Humble, Patrick Debois, John Willis, Nicole Forsgren IT Revolution Press, 30. stu 2021.
7. Virmani M., 2015. Understanding DevOps & Bridging the Gap from Continuous Integration to Continuous Delivery, Int’l Conf. Innovative Computing Technology (INTECH15), pp. 78-82.
8. W. L. Z. Len Bass, 2015. DevOps A Software Architect's Perspective, Addison-Wesley Professional, p. 22.
9. Zhu L., Bass L., Champlin-Scharff G., 2016. “DevOps and Its Practices,” tom 33, br. IEEE, pp. 32-34, 3, May-June