Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.4
DOI: https://doi.org/10.24867/32BE14Orolic

PRODUKCIJSKA PRIMENA KUBERNETISA U AMAZON VEB SERVISIMA

PRODUCTION KUBERNETES DEPLOYMENT ON AMAZON WEB SERVICE

Kristina Orolić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Ovaj rad istražuje proces korišćenja Kubernetis klauda specifičnom metodom eksctl u okviru AWS okruženja. Razmatraju se mikroservisi i arhitekture kontejnera, komponente Kubernetisa, arhitektura i sigurnosni mehanizmi. Razmatraju se problemi kao što su kreiranje, brisanje i produkcijski deployment korišćenjem metode eksctl.*
Rezultati istraživanja pomažu u boljem razumevanju rada Kubernetis klastera i svih prednosti koje on pruža.

Ključne reči: *Klaud, Kubernetis, doker, mikroservisi, pod, kontrolna ravan.*

Abstract – *This paper explores the process of using the Kubernetes cloud with the specific method eksctl within the AWS environment. Microservices and containers, Kubernetes components, architecture and security mechanisms are discussed. Issues such as creation, deletion and production deployment using the eksctl tool are discussed.*

The results of the research help in better understanding the operation of the Kubernetes cluster and all the advantages it provides.

Keywords: *Cloud, Kubernetes, Docker, microservices, pod, control plane.*

1. UVOD

Sa ubrzanim razvojem tehnologije i sve većom potrebom za skalabilnim rešenjima, kontejneri su postali jedan od ključnih trendova u savremenom razvoju softvera. Omogućavajući izolovano i konzistentno okruženje za aplikacije, kontejneri reševaju mnoge probleme tradicionalnih metoda implementacije. Među različitim alatima za upravljanje kontejnerima, Kubernetis se izdvaja kao najrasprostranjeniji i najmoćniji sistem za orkestraciju kontejnera.

Kubernetis, često skraćeno kao K8s, razvijen je od strane Google-a, a kasnije je postao deo Cloud Native Computing Foundation (CNCF). Kao open-source rešenje, Kubernetis pruža robusnu infrastrukturu za automatsko upravljanje, skaliranje i održavanje kontejnerizovanih aplikacija.

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

Cilj ovog rada je detaljno istražiti osnovne koncepte, arhitekturu i komponente Kubernetisa, kao i praktične primene ove platforme u razvoju i produkciji aplikacija. Ovaj rad pruža sveobuhvatan pregled Kubernetisa, od njegovih osnovnih principa do naprednih tehničkih alata, kao i da ponudi praktične smernice za njegovu uspešnu primenu u različitim IT okruženjima. Analizirajući prednosti i potencijalne nedostatke Kubernetisa, ovaj rad će pružiti uvide i smernice za kontejnerizaciju aplikacija.

2. AWS - AMAZON WEB SERVICES

2.1. AWS servisi

Računarstvo u oblaku se može definisati kao model koji krajnjim korisnicima omogućava pristup Internetu, na zajednički skup resursa u oblaku. Resursi u oblaku uključuju različite servere i usluge. Ako pogledamo unazad, možemo razlikovati četiri modela primene: private cloud, community cloud, public cloud i hybrid cloud. Pored ovoga, postoje i servisni modeli od kojih su najpopularniji: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) i Software as a Service (SaaS).

2.2. Docker kontejneri

Postoje mnoge prednosti korišćenja Docker kontejnera u poređenju sa fizičkim hardverom. Docker kontejneri i virtuelne mašine obezbeđuju bezbednije, izolovano okruženje, lakše se automatizuju, upravljaju i zamenjuju, obezbeđuju ponovljivo okruženje, nude veću sigurnost, imaju dodatne karakteristike kao npr. replikacija memorije. Da bi se moglo raditi sa Docker kontejnerima, mora se znati razlika između Docker kontejnera i Docker slike. Docker slika je šablon samo za čitanje. Ima instaliran i konfigurisan neki softver. Svaka slika ima ime i oznaku koja služi istoj svrsi kao i verzija softvera. Jedna slika može biti zasnovana na drugoj.
Za Docker kontejnere se tvrdi da su lagani i prenosivi. Dakle, kontejneri su pogodni za mikroservise. Pošto su mikroservisi slabo povezani, kvar na jednom ne bi trebalo da utiče na druge mikroservise aplikacije. Ovaj arhitektonski stil karakteriše fina granularnost i stoga čini skaliranje fleksibilnijim i efikasnim.

2.3. Kubernetis kao sistem orkestracije Docker kontejnera

Proces postavljanja više kontejnera jedne aplikacije može se optimizovati pomoću automatizacije. Ova vrsta optimizacije se naziva orkestracija. Postojala je potreba za automatskim postavljanjem i upravljanjem velikim brojem usluga na globalnom nivou na milionima servera. Iz tog razloga, Google je razvio Borg. Borg je privatni sistem orkestracije.

Kubernetis je napravljen da bi olakšao primenu, smanjio vreme i trud za deployment-e. Da bi postigao ove ciljeve, Kubernetis nudi širok izbor karakteristika: stvaranje, uništavanje i umnožavanje kontejnera, stalno ažuriranje kontejnera, ugrađene provere, automatsko skaliranje, redundantnost i prelazak na grešku, agnostik provajdera, obezbediti isti skup funkcija, korišćenje karakteristika specifičnih za dobavljača, samoizlečenje, otkrivanje usluge, balansiranje opterećenja i orkestraciju skladištenja. Osnovna radna jedinica u Kubernetisu je pod. Pod je apstrakcija i predstavlja jednu aplikaciju, odnosno set kontejnera. Postoje dve ključne stvari o podovima: svi kontejneri u podu biće raspoređeni na jednoj mašini, pod ima dodeljenu jednu IP adresu i svi kontejneri definisani u ovom podu dele istu IP adresu.

3. UVOD U KUBERNETIS

3.1. Mikroservisi i arhitekture kontejnera

Mikroservisna arhitektura je stil arhitekture karakterističan za podelu velikih aplikacija u zbiru labavo povezanih jednostavnih usluga, koje nazivamo mikroservisima. Mikroservisi sarađuju koristeći asinhrone komunikacije za razmenu poruka. Ovaj arhitektonski stil omogućava svakom mikroservisu koji pripada sistemu, da se njime upravlja, ažurira, razvija i primenjuje nezavisno, pojednostavljujući mogućnost održavanja i obezbedjujući skalabilnost.

3.2. Kubernetis arhitektura

Da bi se automatski postavili, konfigurisali i upravljalo sa kontejnerima, potreban je sistem orkestracije kontejnera kao što je Kubernetis. Kada se Kubernetis primeni, kreira se klaster. U arhitekturi Kubernetisa, klaster se sastoji od radnih mašina, koje se takođe nazivaju i radni čvorovi (working nodes). Svaki klaster ima bar jedan čvor.

Pod predstavlja skup pokrenutih kontejnera u klasteru. Za upravljanje radnih čvorova i pod-ova u klasteru je zadužen control plane. Control plane je sloj orkestracije kontejnera koji sa Kubernetis API-jem definiše, implementira i upravlja životnim ciklusom kontejnera.

3.3. Kubernetis komponente

3.3.1. Komponente

Kubernetis se sastoji od sledećih komponenti: kubernetes cluster (sastoji se od čvorova), pods (skup pokrenutih kontejnera u klasteru), kubernetes service (apstrakcija aplikacije koja radi u setu pod-ova), control plane (sloj orkestracije kontejnera, sastoji se od: kube-apiserver, etcd, kube-scheduler, kube-controller-manager, admission controller, cloud-controller-manager).

Čvorovi se sastoje od: kubelet, kube-proxy, container runtime.

Postoje i druge komponente kao što su: CNI, core DNS, ingress, ingress controller itd.

3.3.2. Networking

Kubernetis, podrazumevano, predstavlja model mreže u kome podovi mogu komunicirati međusobno kao i sa čvorovima bez upotrebe NAT-a (Network Address Translation). Agenti kao što su deamons ili Kubelet takođe mogu da komuniciraju sa svakim podom u čvoru bez ograničenja. Kontejneri u okviru poda dele IP, MAC adrese i portove. To znači da su kontejneri u mogućnosti da komuniciraju između sebe preko local host-a ali moraju da koordiniraju preko porta koji koriste.

Kubernetis DNS dodeljuje DNS imena svakom servisu u klasteru, uključujući i DNS servis i konfiguriše kubelets koji upućuju podove da razreše DNS imena koristeći IP adresu DNS servera. Kada pod postavi DNS upit bez navođenja namespace-a, odgovor će samo da enkapsulira DNS imena u isti namespace. Konačno, DNS konfiguracija može da se uspostavi kao pod-to-pod.

Kubernetis daje preporuku za korišćenje sertifikata za obezbeđivanje komunikacije ako je servis izložen internetu. Da bi se koristili sertifikati, server mora biti indikovan da odradi posao u YAML datoteci i tajne moraju biti podešene u etcd da bi se omogućilo podovima da pristupe sertifikatima.

Postoje tri glavna identifikatora za određivanje sa kime pod može komunicirati: sa dozvoljenim podovima, sa dozvoljenim namespace-ovima ili sa dozvoljenim IP blokovima.

Podrazumevano, sve odlazne konekcije su dozvoljene u podu. Može se ograničiti broj odlaznih konekcija u modulu definisanjem mrežne politike poda. Ako postoji takva politika, jedine izlazne konekcije koje su dozvoljene za taj pod biće one koje su detaljno opisane u pomenutoj politici. Isto važi i za dolazne konekcije, podrazumevano su dozvoljene u potpunosti. Definisanje mrežne politike sa tipom ulaza će ograničiti dozvoljeni broj ulaznih konekcija sa onima koje su detaljno opisane u politici i onima iz unutrašnjosti čvora.

3.4. Kubernetis sigurnosni mehanizmi

Kubernetis uključuje nekoliko API-ja i bezbednosnih kontrola, kao i načine za definisanje smernica koje mogu da budu deo načina za upravljanje bezbednošću informacija.

3.4.1. Control plane zaštita

Ključni bezbednosni mehanizam za bilo koji Kubernetis klaster je kontrola pristupa Kubernetis API-ju.

Kubernetis očekuje da se konfiguriše i koristi TLS da bi se obezbedilo šifrovanje podataka u tranzitu unutar control plane-a i između control plane-a i njegovih klijenata. Takođe, može se obezbediti šifrovanje u stanju mirovanja za podatke uskladištene u okviru Kubernetis control plane-a. Ovo je odvojeno od korišćenja enkripcije u mirovanju za podatke o radnim opterećenjima. Tajni API pruža osnovnu zaštitu za vrednosti konfiguracije koje zahtevaju poverljivost.

3.4.2. Zaštita od opterećenja

Potrebno je primeniti bezbednosne standarde pod-a da bi se osiguralo da su pod-ovi i njihovi kontejneri izolovani na odgovarajući način. Takođe, može se koristiti RuntimeClasses da bi se definisala prilagođena izolacija ukoliko je ona potrebna.

Mrežne smernice omogućavaju da se kontroliše mrežni saobraćaj između pod-ova ili između pod-ova i mreže van klastera.

Mogu se primeniti bezbednosne kontrole iz šireg ekosistema da bi se primenile preventivne ili detektivske

kontrole oko pod-ova, njihovih kontejnera i slika koje se u njima pokreću.

3.4.3. Auditing

Kubernetis audit logging obezbeđuje hronološki skup zapisa koji dokumentuje redosled radnji u klasteru. Klaster vrši reviziju aktivnosti koje generišu korisnici, aplikacije koje koriste Kubernetis API i sam kontrolni nivo.

4. DOSTUPNE KUBERNETIS KLASTER DEPLOYMENT METODE

4.1. Alati komandne linije za primenu Kubernetis klastera

Kops predstavlja skraćenicu od Kubernetes Operations i reklamira se kao "najlakši način da se proizvodni Kubernetis klaster pokrene". To je alatka komandne linije koja omogućava kreiranje, uništavanje, nadogradnju i održavanje Kubernetis klastera proizvodnog kvaliteta koji su visoko dostupni.

Drugi najpopularniji alat je kubespray. On takođe podržava više cloud-a: AWS, GCP, Azure, OpenStack, vSphere, Oracle Cloud Infrastructure i Baremetal. Glavna razlika između kubespray-a i kops-a je u tome što kubespray koristi Ansible (alat otvorenog koda za obezbeđivanje infrastrukture) dok kops sam obavlja obezbeđivanje i orkestraciju. Kops takođe pruža više funkcija čvrsto integrisanih sa određenim cloud-ima.

Tu je takođe i kubeadm koji za razliku od kops-a i kubespray-a pomaže da se dobije minimalni održivi klaster na način koji je prilagođen korisniku. Štaviše, njegov obim je ograničen na lokalni sistem datoteka.

4.2. Metod primene: na AWS EKS Managed Service koristeći eksctl

Cilj ove metode je da AWS upravlja control plane-om. To znači da je AWS taj koji je odgovoran za upravljanje Kubernetis glavnim komponentama, kao što su: API server, kube-scheduler, kube-controller-manager, kao i etcd. Control plane bi trebalo da bude dostupan zahvaljujući tome što je raspoređen u više zona dostupnosti. Zona dostupnosti je izolovano geografsko mesto za host-ovanje Amazon data centara. Nekoliko zona dostupnosti kreira region. Zone dostupnosti u regionu su povezane preko low-latency veza.

Control plane-u klastera prednjači Elastic Load Balancer koji takođe brine o celom umrežavanju kako bi se obezbedila povezanost instanci control plane-a do radnih čvorova. Zahvaljujući tome, AWS EKS korisnik može da pristupi Kubernetis API serveru. Da bi se koristio AWS EKS, postoje dva načina: korišćenje eksctl CLI i korišćenje AWS Management Console i AWS CLI. I jedan i drugi način zahtevaju instaliranje AWS CLI. U ovom radu korišćena je metoda sa eksctl CLI. Eksctl je zvanično CLI za AWS EKS, podržan od strane AWS-a, iako je pokrenut od strane WeaveWorks-a. Sa eksctl-om, jednostavna komanda se može koristiti za instanciranje celog klastera.

5. PRIPREME ZA PROIZVODNU IMPLEMENTACIJU KUBERNETIS KLASTERA

5.1. Odabrani zahtevi za implementaciju i acceptance criteria

Osim ako bilo koja od komponenti Kubernetis control plane-a nije zdrava, moraće biti popravljena. To ne bi trebalo da bude problem sa upravljanim Kubernetis servisima, ali za samohostovane klastere bi trebalo proveriti. Treba napomenuti da, pošto zdravstveni pregledi trebaju biti vođeni često, ne bi trebalo raditi ništa što je preskupo. Za praćenje zdravlja Kubernetis klastera, moglo bi se proveriti sledeće: broj čvorova, status zdravlja čvorova, broj podova po čvoru i ukupno, korišćenje/dodela resursa po čvoru i ukupno.

5.2. Central logging

Plan je jednostavno koristiti uslugu AWS CloudWatch servisa koristeći eksctl. AWS CloudWatch nije omogućen podrazumevano zbog gubljenja podataka i troškova skladištenja. Takođe bi trebalo da postoji mogućnost da se omogući evidentiranje na AWS CloudWatch-u kada se koristi kops.

5.3. Central Monitoring

Da bi se zadovoljio zahtev za centralno nadgledanje, planirano je da se koristi AWS CloudWatch. Postoji i rešenje - program pod nazivom Kubernetes Metrics Server. Čak iako je to server koji obezbeđuje izvor metrike resursa kontejnera, ne bi ga trebalo koristiti kao tačan izvor metrike korišćenja resursa. Njegova glavna upotreba je za CPU/Memory baziranu na horizontalnom automatskom skaliranju za automatsko prilagođavanje/predlaganje resursa potrebnih kontejnerima. Dakle, neće se koristiti za centralni nadzor, ali treba biti svestan njegovog postojanja.

5.4. Central Audit

Ovaj zahtev će se jednostavno ispuniti korišćenjem usluge AWS CloudTrail servisa. Ovaj servis treba pokazati koji korisnik je odgovoran za promene koje je uneo u AWS resources. Osim toga, AWS CloudTrail takođe može biti korišćen za otkrivanje neobičnih aktivnosti u AWS Accounts.

5.5. Autoscaling

Ideja je testirati da li će klaster, koji se sastoji od master čvorova i radnih čvorova, biti u mogućnosti da sam kreira i briše druge radne čvorove. Ova odluka, da li skalirati unutra ili spolja, treba uzeti u obzir količinu resursa dostupnih za pod-ove. Potražnja za više resursa će biti veštački kreirana povećanjem vrednosti replicaCount-a unutar NGINX Helm Chart-a.

5.6. Bezbednost

Postoji mnogo mera bezbednosni koje se mogu primeniti na klaster. Da bi ostao bezbedan, pristup Kubernetis klasteru treba biti ograničen. Bilo koja primena Kubernetisa na AWS-u će morati da se bavi umrežavanjem.

VPC je skraćenica za Virtual Private Cloud i predstavlja AWS resurs koji obezbeđuje mrežne servise. Kada se kreira VPC to znači da je kreirana virtuelna privatna mreža i da je dodeljena određenom AWS nalogu. AWS resursi se mogu pokrenuti unutar tog VPC-a. Svaki VPC ima dodeljen CIDR blok. VPC ne može da obuhvata više od

jednog AWS regiona. VPC se mogu podeliti na podmreže. Jedna podmreža ne može obuhvatati više od jedne zone dostupnosti. Podmreže mogu biti javne ili privatne. Ako je saobraćaj mreže preusmeren na internet gateaway (koji je AWS resurs), podmreža je poznata kao javna mreža, u suprotnom, podmreža je privatna. To znači da javne podmreže imaju pristup internetu.

Dakle, da bi klaster učinio bezbednim, pristup klasteru ni trebao biti ograničen. Samo jedna IP adresa treba biti dozvoljena za pristup i povezivanje.

6. PRODUKCIJSKI DEPLOYMENT KORISTEĆI EKSCTL

6.1. Kreiranje klastera

Koristeći eksctl, moguće je kreirati klaster sa eksctl create cluster komandom i bilo sa podešavanjem opcija komandne linije ili korišćenjem YAML fajla. Nakon kreiranja klastera, iz AWS konzole se mogu videti detalji klastera kao npr. API endpoint, verzija klastera, čvorovi, tagovi, monitoring, networking.

Prvo se radi deploy control plane-a, zatim i radnih čvorova. Za svaki od ova dva zadatka korišćen je poseban CloudFormation stack. Taj posao automatski odrađuje AWS EKS. Takođe, moguće je videti AWS resurse koje je obezbedio svaki CloudFormation stack.

Cloud plane-om u potpunosti upravlja AWS i on je pokrenut u dve AWS zone dostupnosti kako bi se osigurala visoka dostupnost. Krajnji korisnik čak nema pristup control plane-u, što znači da ne postoji EC2 instanca sa vidljivim master čvorom kao i da kada su izlistani svi Kubernetis čvorovi, vidljivi su samo radni čvorovi.

Takođe, neka određena podešavanja se mogu podesiti pomoću eskctl utils komande. Ova komanda će nam dozvoliti da pristupimo klaster API sa našom IP adresom.

6.2. Brisanje klastera

Da bi se klaster obrisao, potrebno je pokrenuti delete cluster komandu. Ova komanda služi za brisanje skoro svih AWS resursa, osim AWS CloudWatch LogGroup. Razlog tome je što neko može biti zainteresovan za proučavanje logova nakon što je klaster ugašen i obrisan.

6.3. Instaliranje aplikacija koristeći helm

Helm pomaže u upravljanju Kubernetis aplikacijama - Helm Charts pomaže u definisanju, instaliranju i nadogradnji čak i najsloženijih aplikacija. U ovom radu korišćen je helm za instaliranje NGINX, koji predstavlja veb server, a može služiti i kao reverzni proksi server.

Prvo, na klaster se treba instalirati NGINX repozitorijum. Kada se instalacija završi, može se izvršiti pregled NGINX servisa i pod-ova. Kada je NGINX instalacija završena, helm će kreirati ingress, sa kojim se NGINX-u može pristupiti javno.

7. ZAKLJUČAK

Na samom kraju, ovaj rad je imao za cilj da predstavi efikasnost primene Kubernetis klastera u oblaku koristeći eksctl metodu na Amazon Web Services (AWS). Korišćenjem eksctl-a, pojednostavljenog i efikasnog alata komandne linije, pojednostavljen je proces kreiranja, upravljanja i skaliranja Kubernetis klastera na AWS Elastic Kubernetes Service (EKS).

Buduća istraživanja bi se mogla fokusirati na dalju automatizaciju deplozment pipeline-a, integraciju naprednih bezbednosnih mera i optimizaciju upravljanja troškovima u okruženjima sa više klastera.

Sve u svemu, eksctl pruža moćan alat za organizacije koje žele da iskoriste Kubernetis na AWS-u, podstičući inovacije i operativnu efikasnost unutar cloud deployments-a.

8. LITERATURA

[1]https://kubernetes.io/docs/concepts/overview/components/ (pristupljeno u oktobru 2024.)
[2] https://en.wikipedia.org/wiki/Amazon_Web_Services (pristupljeno u oktobru 2024.)
[3] https://en.wikipedia.org/wiki/Cloud_computing (pristupljeno u oktobru 2024.)
[4]https://aws.amazon.com/eks/ (pristupljeno u oktobru 2024.)
[5] https://aws.amazon.com/cli/ (pristupljeno u oktobru 2024.)
[6]https://kubernetes.io/docs/concepts/services-networking/ (pristupljeno u oktobru 2024.)
[7]https://kubernetes.io/docs/concepts/services-networking/ingress/ (pristupljeno u oktobru 2024.)
[8]https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html (pristupljeno u oktobru 2024.)
[9] https://kubernetes.io/case-studies/booking-com/ (pristupljeno u oktobru 2024.)
[10]https://kubernetes.io/docs/tasks/manage-kubernetes-objects/declarative-config/ (pristupljeno u oktobru 2024.)

Kratka biografija:

Kristina Orolić rođena je 10. februara 2000. godine u Kruševcu. Osnovnu školu “Rade Dodić” završila je u Milutovcu, dok je matematički smer gimnazije “Vuk Karadžić” završila u Trsteniku 2018. godine. Školske 2018/2019 godine upisuje Fakultet tehničkih nauka u Novom Sadu, na studijski program Primenjeno softversko inženjerstvo. Diplomirala je 2022. godine te školske 2022/2023 godine upisuje master studije, takođe na smeru Primenjeno softversko inženjerstvo.

kontakt: kristinaorolic@gmail.com