UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2025/2026.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

Problemi ručnog postavljanja kontejnera

- Ručno postavljanje (engl. *deployment*) kontejnera može da bude komplikovano za održavanje i sklono greškama.
  - Kontejneri mogu da prestanu sa radom i da zahtevaju ponovno postavljanje.
    - Potrebno je ručno kontrolisati da li kontejner radi.
    - Potrebno je ručno ponovo pokrenuti kontejner.
  - Povećanje saobraćaja može da zahteva postavljanje dodatnih instanci kontejnera.
    - U određenim periodima dana može da se pojavi povećana potražnja za određenim uslugama.
    - Potrebno dodavanje još instanci kontejnera kako bi aplikacija uspešno izvršavala zahteve klijenata.
    - Potrebno je smanjenje instanci kontejnera kada ne postoji povećan saobraćaj zbog uštede sredstava.
  - Potrebno je ravnomerno raspodeliti dolazeći saobraćaj na sve instance kontejnera.
    - Kako bi se sprečilo da svi zahtevi odlaze samo ka jednoj instanci kontejnera, potrebno je obezbediti da se nadolazeći saobraćaj ravnomerno raspodeli na sve kontejnere.

Rešenje u vidu AWS ECS servisa

- Ručno postavljanje kontejnera uz upotrebu cloud provajder servisa poput AWS ECS.
- PREDNOSTI:
  - Ručno postavljanje kontejnera može da bude komplikovano za održavanje i sklono greškama.
    - Kontejneri mogu da prestanu sa radom i da zahtevaju ponovno postavljanje.
      - Usluga: Automatska provera zdravlja kontejnera + ponovno postavljanje u slučaju prestanka rada.
    - Povećanje saobraćaja može da zahteva postavljanje dodatnih instanci kontejnera.
      - Usluga: Autoskaliranje
    - Potrebno je ravnomerno raspodeliti dolazeći saobraćaj na sve instance kontejnera.
      - Usluga: Load balancing
- MANA:
  - Usluga određenog cloud provajdera nas ograničava.
    - npr. Podešavanja za AWS ECS neće raditi na Microsoft Azure platformi.
    - Potrebno je naučiti sve specifičnost konkretnog cloud provajdera.

Alati za orkestraciju kontejnera

- Docker Swarm
- Kubernetes
- Apache Mesos
- Amazon Elastic Container Service (ECS)
- Helios
- Nomad
- OpenShift
- Azure Kubernetes Service (AKS)
- Rancher

Docker Swarm VS Kubernetes

Kubernetes	Docker Swarm
Kompleksna instalacija	Jednostavna instalacija
Teže za naučiti, ali moćnije	Lakše za naučiti, ali manje funkcionalnosti
Podržava automatsko skaliranje	Ne podržava automatsko skaliranje
Ugrađen sistem za monitoring	Potrebni dodatni alat za monitoring
Ručno podešavanje load balancera	Automatski podržava load balancing
Poseban CLI alat (kubectl)	Integriše docker CLI

Kubernetes (K8s)

- Kubernetes je *open-source* sistem (i standard) za upravljanje kontejnerizovanim aplikacijama.
- Razvijen je u Google-u 2014. godine.
- Kubernetes omogućava:
  - automatizaciju puštanja aplikacije u produkciju
  - skaliranje pojedinačnih servisa (kontejnera) u okviru aplikacije
  - upravljanje konfiguracijama na nivou pojedinačnih servisa
- Kubernetes je nezavistan od bilo koje cloud platforme.
- Kubernetes omogućava standardizovan način postavljanja kontejnera bez obzira na platformu.
  - Istu Kubernetes konfiguraciju možemo proslediti bilo kom cloud provajderu ili udaljenoj mašini.

Šta Kubernetes NIJE?

- Kubernetes NIJE cloud provajder poput AWS, Azure, …
  - Kubernetes je open-source projekat (kolekcija standarda i softvera) koji može da se koristi sa bilo kojim cloud provajderom.
- Kubernetes NIJE usluga ni jednog cloud provajdera.
  - Iako AWS nudi servis za pomoć u radu sa Kubernetesom, Kubernetes nije usluga datog cloud provajdera.
- Kubernetes NIJE ograničen ni na jedan cloud provajder.
- Kubernetes NIJE samo softver koji se pokrene na nekoj mašini.
  - Kubernetes je kolekcija koncepata i alata.
- Kubernetes NIJE alternativa Docker-u.
  - Radi zajedno sa Docker kontejnerima kako bi omogućila njihovo postavljanje.
- Kubernetes nije usluga koja se plaća.
  - Kubernetes je besplatan open-source projekat.
  - Osim ako se kontejneri pomoću Kubernetesa postavljaju na određeni cloud provajder.

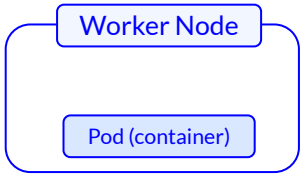
Osnovni koncepti

- **Klaster -** skup čvorova (engl. *Node*) koji ili izvršavaju kontejnerizovanu aplikaciju (Worker Node-ovi) ili kontrolišu druge čvorove (Master Node)
- **Čvor (engl. node) -** fizička ili virtuelna mašina sa određenim hardverskim kapacitetom
  - **Podređeni čvor (engl. Worker Node)**
    - izvršava zadatke koje mu dostavi glavni čvor
  - **Glavni čvor (engl. Master Node)**
    - glavni čvor ekosistema koji upravlja i nadgleda jedan ili više podređenih čvorova
- **Kapsula (engl. Pod)**
  - aplikacija ili njen deo koja se izvršava u podređenom čvoru
  - izvršava pokrenuti kontejner i upravlja njegovim resursima
  - najmanja jedinica manipulacije u Kubernetes-u
- **Kontejner -** običan Docker kontejner
- **Servisi -** logički skup Kapsula sa jedinstvenom i nezavisnom IP adresom

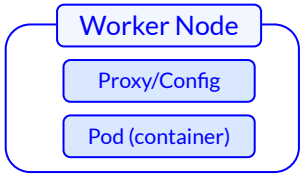
Kubernetes arhitektura

Pod (container)

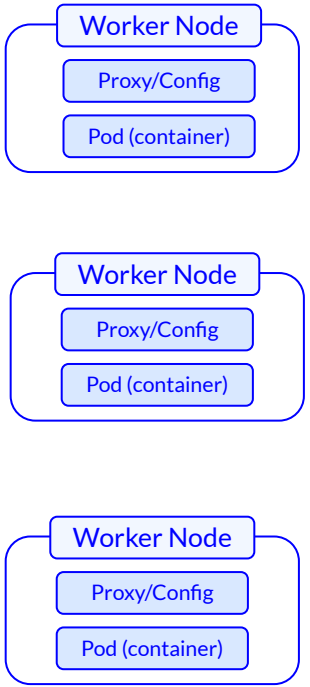
Kubernetes arhitektura



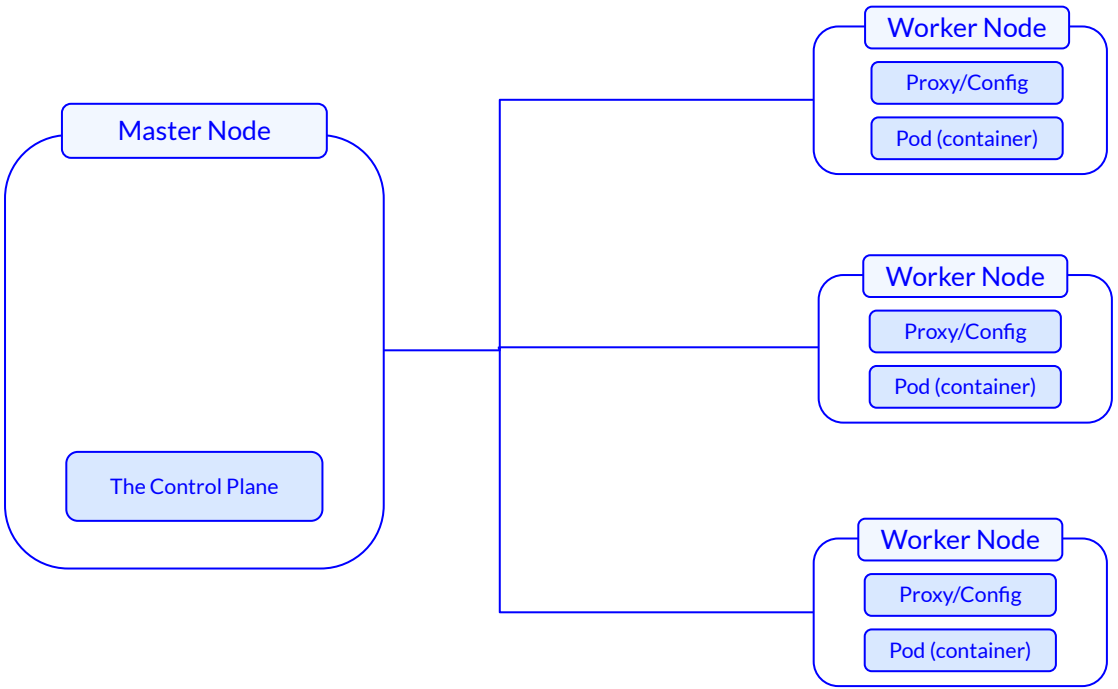
Kubernetes arhitektura



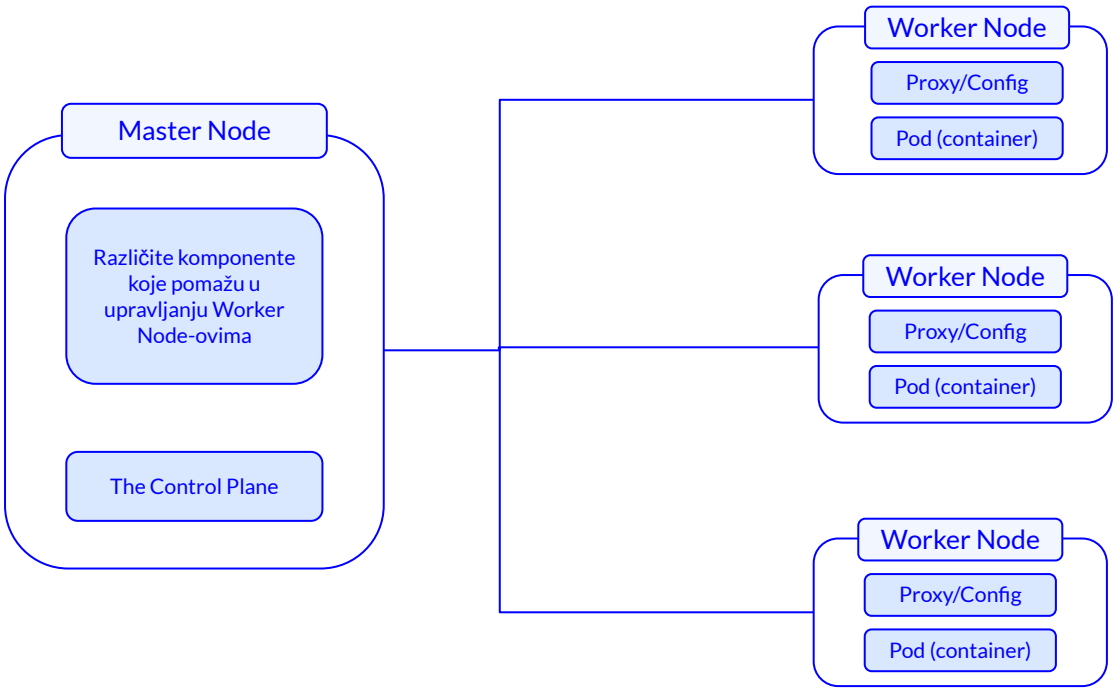
Kubernetes arhitektura



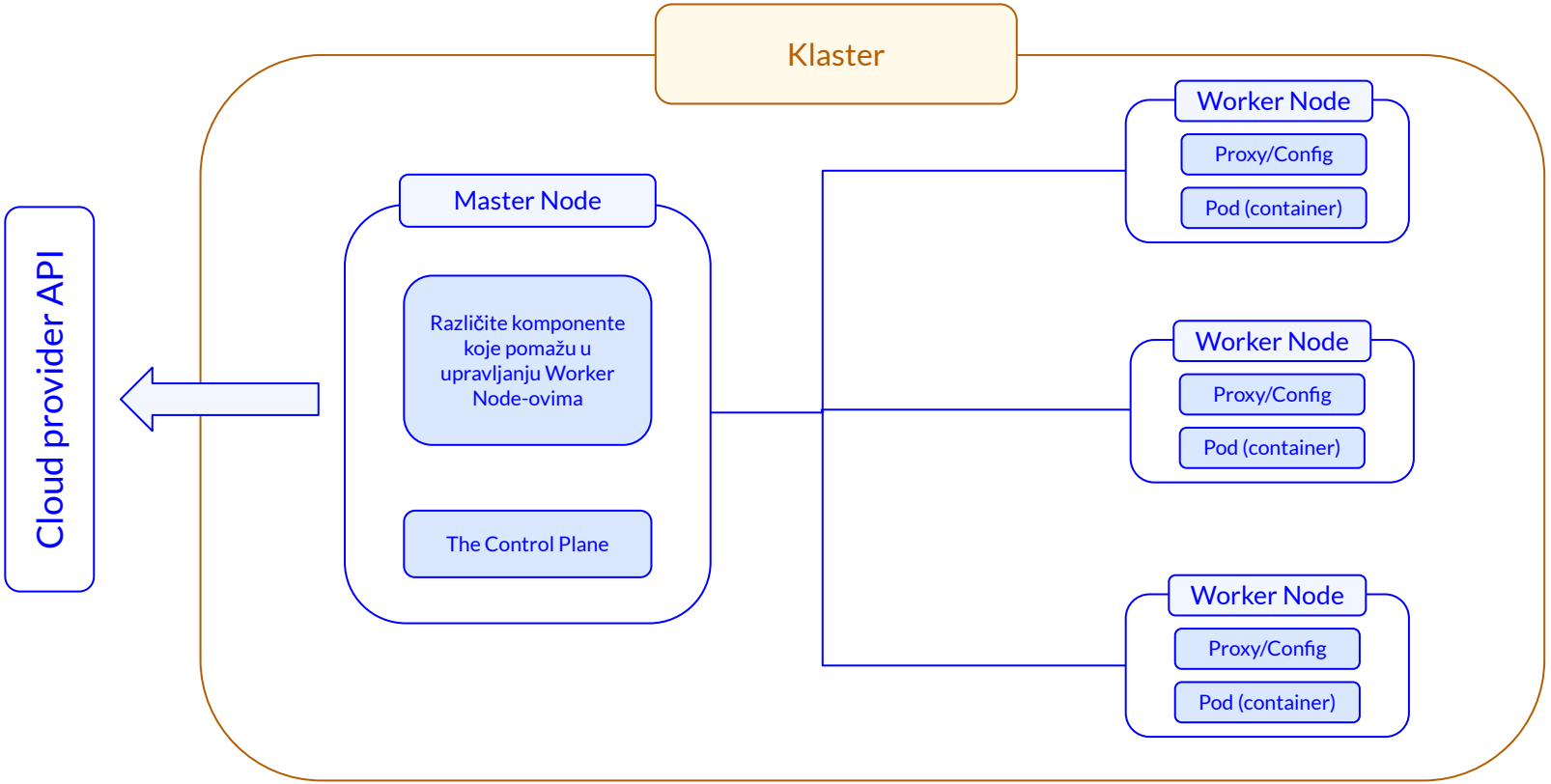
Kubernetes arhitektura



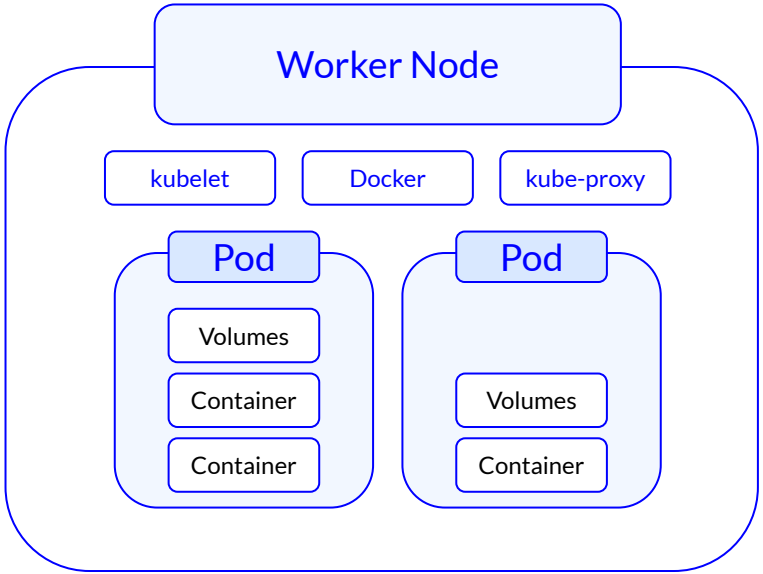
Kubernetes arhitektura



Kubernetes arhitektura



Worker Node


Worker Node
kubelet
Docker
kube-proxy
Pod
Volumes
Container
Container
Pod
Volumes
Container

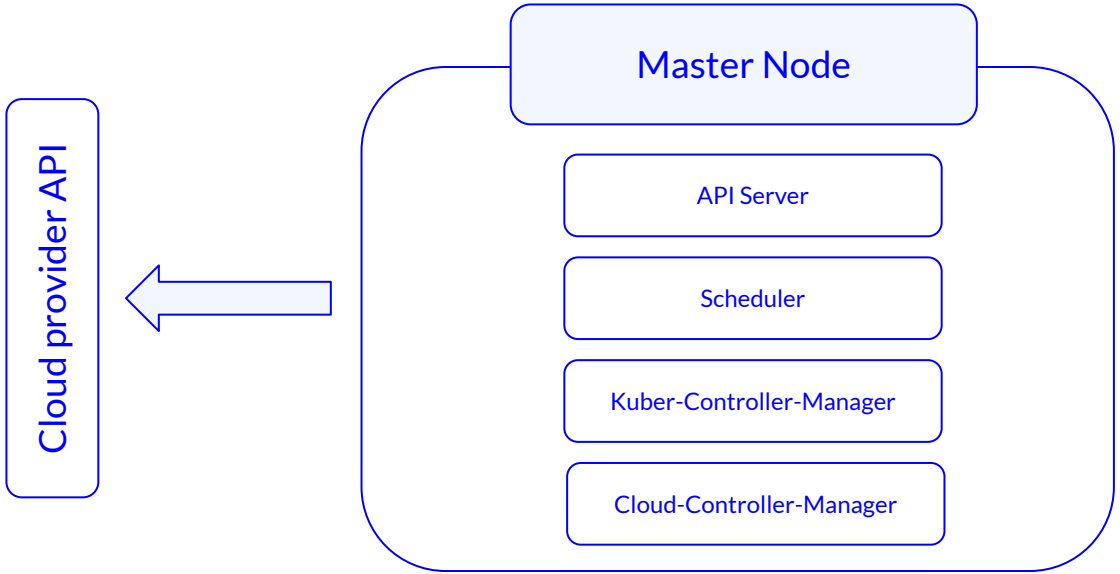
Podređeni čvor (Worker Node)

- Worker Node može da bude ili jedan fizički računar ili jedna virtuelna mašina na računaru.
- Worker Node-om upravlja Master Node.
- Unutar Worker Node-a se nalazi jedan ili više Pod-ova.
- Pod služi za izvršavanje jednog ili više kontejnera i njihovih resursa (volume-i, IP, run config, itd).
    - Podove kreira i njima upravlja Kubernetes.

Podređeni čvor (Worker Node)

- Komponente podređenog čvora:
  - **izvršno okruženje kontejnera**
    - omogućava pokretanje kontejnera
  - **kubelet**
    - softverski agent koji obezbeđuje da su kontejneri pokrenuti u kapsulama
    - izvršava naredbe Master Node-a
    - pokreće, zaustavlja i nadgleda izvršavanje kontejnera
  - **kube-proxy**
    - obezbeđuje da se izvršava samo dozvoljeni saobraćaj između Worker Node-a i ostatka sveta
    - omogućava kreiranje i apstrakciju API-ja kontejnera pomoću servisa

Master Node


Master Node
API Server
Scheduler
Kuber-Controller-Manager
Cloud-Controller-Manager
Cloud provider API

Glavni čvor (Master Node)

- **Glavni čvor:**
  - omogućava upravljanje klasterom
  - mesto gde je implementirana logika reagovanja na događaje u klasteru
  - može biti implementirana na bilo kojoj mašini u klasteru
    - ali obično postoji poseban računar za glavni čvor

Glavni čvor (Master Node)

- Komponente glavnog čvora
  - **kube-apiserver**
    - Kubernetes API za rad sa Kubernetes komponentama
  - **etcd**
    - konzistentno i visoko-raspoloživo skladište podataka
    - interno ga koristi Kubernetes
  - **kube-scheduler**
    - komponenta koja prati novokreirane kapsule koje nisu dodeljene čvorovima
      - i dodeljuje ih odgovarajućem klasteru
    - uzima u obzir lokalne i globalne potrebe za resursima, lokalnost podataka, dostupnost delova mreže, itd.

Glavni čvor (Master Node)

- Komponente glavnog čvora
  - **kube-controller-manager**
    - upravljač kontrolerima u okviru glavnog čvora:
      - upravljač replikacijom - obezbeđuje dostupnost delova sistema
      - upravljač servisima - upravlja servisima i povezuje ih sa kapsulama
      - upravljač čvorovima - posmatra status čvorova
      - upravljač nalozima i tokenima za servise
  - **cloud-controller-manager**
    - upravljač kontrolerima koji komuniciraju sa pružaocem usluga infrastrukture na kojoj se Kubernetes nalazi
      - obuhvata kontrolere za rad sa mrežom, skladištem podataka itd.
      - specijalizovan za određeni cloud provajder (npr. AWS)

Komunikacija

- Komunikacija u smeru od podređenog čvora ka glavnom
  - preko HTTPS
  - mora biti uspostavljen neki oblik autentifikacije
  - komunicira se isključivo sa *kube-apiserver*
- Komunikacija u smeru od glavnog čvora ka podređenim čvorovima
  - *kube-apiserver* -> *kubelet*
    - preuzimanje logova, dodela kapsula čvorovima, preusmeravanje portova
  - *kube-apiserver* -> čvorovi, kapsule, i servisi
    - komunikacija po HTTP protoklu (može i HTTPS)

Posao programera VS. posao Kubernetes-a

- Posao programera:
  - Kreiranje klastera i instanci čvorova (Worker + Master node)
  - Postavljanje Kubernetes servisa i softvera na date čvorove
  - Postavljanje dodatnih servisa po potrebi (Load balancer, Filesystem, itd.)
- Posao Kubernetesa:
  - Kreiranje objekata (npr. Podova) i njihovo upravljanje
  - Kontrolisanje Pod-ova, njihovo rekreiranje i skaliranje
  - Primena dobijene konfiguracije za postizanje željenih rezultata

kubectl

- kubectl je alat komandne linije za komunikaciju sa control plane-om Kubernetes klastera
- Link za instaliranje: https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/
- kubectl sintaksa:
  - `kubectl [command] [TYPE] [NAME] [flags]`
    - `command` - određuje operaciju koja treba da se izvrši (npr. create, get, describe, delete,..)
    - `TYPE` - opisuje tip resursa (npr. deployment, service, pod,..)
    - `NAME` - opisuje naziv specifičnog resura (npr. example-pod, first-deployment, ..)
    - `flags` - određuju opciona podešavanja

minikube

- minikube je alat koji brzo postavlja lokalni Kubernetes klaster na macOS, Linux i Windows OS-u.
- predefinisano minikube kreira klaster sa jednim čvorom.
- Link za instaliranje minikube: https://minikube.sigs.k8s.io/docs/start/
- Naredba za pokretanje minikube klastera:
  - `minikube start`
- Naredba za zaustavljanje minikube klastera:
  - `minikube stop`
- Naredba za proveru statusa minikube klastera
  - `minikube status`
- Naredba za prikazivanje informacija o podignutom klasteru
  - `minikube dashboard`

Imperativni VS. Deklarativni Kubernetes

- Imperativni
  - Izvršavanje pojedinačnih naredbi kako bi se okinule određene Kubernetes akcije
  - `kubectl create …`
  - Uporedivo sa upotrebom docker CLI
- Deklarativni
  - Definisanje i primena konfiguracionog fajla za kreiranje željenog stanja
  - `kubectl apply -f config.yaml`
  - Uporedivo sa upotrebom docker compose klijenta

Kubernetes objekti

- Objekti su glavne gradivne komponente u Kubernetes-u
  - trajni entiteti u ekosistemu koje Kubernetes koristi kako bi opisao stanje klastera
    - objekti predstavljaju zapis o nameri
      - tj. pomoću njih opisujemo kako želimo tačno da klaster izgleda
  - opisuju:
    - koje aplikacije se izvršavaju i na kojim čvorovima
    - koje resurse aplikacije koriste
    - polise dodeljene tim aplikacijama

Kubernetes objekti

- Polja Kubernetes objekta
  - *spec*
    - sadrži specifikaciju željenog stanja objekta,
      - kroz podešavanja njegovih karakteristika
  - *status*
    - sadrži pravo stanje objekta
      - koje održava sam Kubernetes
- Kubernetes nadgleda i poredi vrednosti u dva polja
  - u slučaju razlike reaguje na odgovarajući način
    - npr. podiže dodatnu instancu ukoliko je broj instanci u polju *status* manji od broja instanci u polju *spec*

Kubernetes objekti

- Primeri objekata:
  - Pod
  - ReplicaSet
  - Deployment
  - Service
  - Job
  - CronJob
  - ConfigMap
  - Secret
  - Namespace

Kapsula (Pod)

- Predstavlja najmanje i najjednostavnije objekte koji se mogu kreirati i pokrenuti
- Predstavlja procese koji se izvršavaju u okviru Kubernetes klastera
- Pod čini grupu od jednog ili više kontejnera sa deljenim skladištem, mrežnim resursima i specifikacijom za izvršavanje datih kontejnera.

Replikacioni set (Replica Set)

- Služi za održavanje konstantnog broja replika Podova u bilo kom trenutku
- Ako neki Pod prekine sa radom, Replica Set automatski kreira novi Pod kako bi zadržao željeni broj replika.
    - Ovo obezbeđuje stabilnost i dostupnost aplikacije
- Ne podržava automatsko ažuriranje ili povratak na prethodne verzije za razliku od Deploymenta.

Postavka replikacionog seta (Deployment)

- Objekat koji omogućava upravljanje skupom Podova i ReplicaSet-ova.
- Glavna svrha Deploymenta je da olakša automatsko ažuriranje aplikacija (rolling update) i pruži mogućnost povratka na prethodnu verziju bez prekida usluge.
- Deployment automatski pravi i ažurira ReplicaSet-ove kada se promeni definicija Pod-a
    - Tako osigurava konstantan broj Pod-ova i kontinuiranu dostupnost aplikacije
- Deployment je koristan za upravljanje životnim ciklusom aplikacija u Kubernetes okruženju, posebno za operacije kao što su skaliranje, ažuriranje i samopopravka.

Pod VS. ReplicaSet VS. Deployment

- **Pod**
  - Pokreće jednu instancu aplikacije ili kontejnera.
  - Ne rekreira se automatski ako se izbriše ili prestane raditi.
  - Nije podržano ažuriranje u toku rada niti upravljanje verzijama.
- **ReplicaSet**
  - Obezbeđuje da određeni broj replika pod-ova bude u funkciji.
  - Automatski rekreira pod-ove kako bi održao željeni broj replika.
  - Može se skalirati povećavanjem ili smanjenjem broja replika.
  - Nije podržano ažuriranje u toku rada niti upravljanje verzijama.
- **Deployment**
  - Upravlja ReplicaSet-ovima i podržava napredne funkcije poput ažuriranja i vraćanja na prethodne verzije.
  - Upravlja osnovnim ReplicaSet-om kako bi obezbedio oporavak.
  - Jednostavno skalira replike kroz polje replicas ili komande.
  - Podržava ažuriranje u toku rada i upravljanje verzijama.

Servis

- Servis je objekat koji omogućava mrežni pristup skupu Pod-ova unutar klastera.
- Servis definiše kako aplikacije ili korisnici mogu pristupiti Podovima, čime se omogućava konzistentna i pouzdana komunikaciju
- Svaki Pod ima internu IP adresu (kao što i Docker kontejneri imaju svoje interne IP adrese)
    - PROBLEMI:
        - Ne može se koristiti interna IP adresa da se pristupi od spolja
        - IP adresa se menja svaki put kad se Pod ponovo kreira
- Servis grupiše Pod-ove i dodeljuje im zajedničku IP adresu koja se ne menja i na taj način se može Pod-ovima pristupiti i od spolja
- Unutar klastera je jako teško pristupiti Podovima ako se ne koristi Servis
- Izvan klastera je nemoguće pristupiti Podu ako se nekoristi Servis

Servis

- Karakteristike servisa:
  - Servis omogućava abstrakciju pristupa, što znači da se podovima pristupa putem servisa, a ne direktno. To omogućava podovima da se menjaju, ažuriraju ili skaliraju bez uticaja na način pristupa aplikaciji.
  - Svaki servis u Kubernetes klasteru ima svoju IP adresu i DNS ime, što pruža stalnu tačku pristupa za komunikaciju sa podovima, bez obzira na to gde se podovi nalaze u klasteru.
  - Servis automatski distribuira mrežni saobraćaj na podove unutar klastera, što omogućava balansiranje opterećenja i povećava pouzdanost i dostupnost aplikacija.
  - Kubernetes podržava različite tipove servisa, uključujući ClusterIP (unutrašnji pristup unutar klastera), NodePort (spoljašnji pristup preko IP adrese čvora), LoadBalancer (integracija sa spoljašnjim balanserom opterećenja) i ExternalName (mapiranje servisa na eksterni DNS).
  - Servis koristi selektore da identifikuje koje podove treba da uključi. Selektori se obično zasnivaju na oznakama (labels) koje su dodeljene podovima.

Job

- Objekat koji omogućava izvršavanje **jednokratnih zadataka** u klasteru, osiguravajući da se zadatak uspešno završi.
- **Funkcionalnost:**
  - Kreira Pod-ove koji izvršavaju definisani zadatak.
  - Automatski ponavlja zadatak u slučaju greške, prema definisanom broju pokušaja (`backoffLimit`).
    - Ako se backoffLimit ne definiše, podrazumevana vrednost je **6**.
  - Praćenje uspešnog završetka zadatka.
- **Slučajevi upotrebe:**
  - Obrada podataka
  - Periodični back-up baze
  - Generisanje izveštaja

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: python-job
spec:
  backoffLimit: 3
  template:
    spec:
      containers:
      - name: python-task
        image: python:3.9
        command: ["python", "-c"]
        args:
          - |
            print("Pokrenut zadatak...");
            print("Zadatak završen!");
      restartPolicy: OnFailure
```

Job

- **restartPolicy: OnFailure** upravlja time da Kubernetes restartuje pod u slučaju greške.
- **backoffLimit** upravlja koliko ukupno puta Kubernetes može da pokuša pokretanje novih pod-ova nakon što prethodni potpuno otkaže.
- Jobs **uvek koriste restartPolicy: OnFailure ili Never**, jer njihova svrha nije da održavaju dugotrajne pod-ove (kao što to rade Deployments).

CronJob

- Kreira **Job** objekte u zadatom vremenskom intervalu.
- Podržava periodične zadatke, kao što su back-up baze, izveštaji ili čišćenje podataka.
- Cron sintaksa za vreme:

```
* * * * *
| | | | |
| | | | +---- Dan u nedelji (0 - nedelja do 6 - subota)
| | | +------ Mesec (1 - 12)
| | +-------- Dan u mesecu (1 - 31)
| +---------- Sat (0 - 23)
+------------ Minut (0 - 59)
```

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: python-cronjob
spec:
  schedule: "*/5 * * * *"  # Pokreće zadatak svakih 5 minuta
  jobTemplate:
    spec:
      backoffLimit: 3
      template:
        spec:
          containers:
          - name: python-task
            image: python:3.9
            command: ["python", "-c"]
            args:
              - |
                print("Pokrenut zadatak iz CronJob-a...");
                print("Zadatak završen!");
          restartPolicy: OnFailure
```

ConfigMap

- **ConfigMap** je Kubernetes objekat koji omogućava **čuvanje i upravljanje nesenzitivnim konfiguracionim podacima**.
- Za lako ažuriranje konfiguracija bez izmene aplikacijskog koda.
- Podaci mogu biti prosleđeni aplikacijama kao:
  - Promenljiva okruženja
  - Montirani fajlovi
  - Komandne argumente

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  app-name: "My Application"
  app-version: "1.0"
  log-level: "debug"

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: busybox
        command: ["sh", "-c", "echo App: $APP_NAME, Version: $APP_VERSION, Log Le
        env:
        - name: APP_NAME
          valueFrom:
            configMapKeyRef:
              name: app-config
              key: app-name
```

Secret

- **Secret** je Kubernetes objekat namenjen za **bezbedno skladištenje i upravljanje osetljivim podacima**, kao što su: korisnička imena, lozinke, tokeni, sertifikati, …
- Podaci se automatski enkodiraju u Base64 format.
- Podaci se čuvaju u memoriji klastera (etcd) i mogu se enkriptovati.
- Pristup kontrolisan putem RBAC (Role-Based Access Control).
- Podaci mogu biti prosleđeni aplikacijama kao:
    - Promenljiva okruženja
    - Montirani fajlovi

Secret

- Kada se kreira K8s Secret objekat, postoje dva načina da se navedu podaci:
  - stringData
  - data
- **stringData**
  - Podaci se unose u čitljivom (plain text) formatu, tj. **bez prethodnog Base64 kodiranja**.
  - Kubernetes automatski pretvara vrednosti iz stringData u data polje kao Base64 kodirane vrednosti.
- **data**
  - Podaci se unose kao Base64 kodirane vrednosti.
  - Kubernetes ne vrši dodatnu obradu – očekuje da su svi podaci već kodirani.
  - Korisno kada podaci dolaze iz eksternih izvora ili skripti koje ih već kodiraju u Base64.
  - Oba načina se mogu navesti u istoj definiciji Secret objekta, ali ukoliko se koristi isti ključ u okviru te definicije prednost ima ključ naveden pod stringData

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: combined-secret
type: Opaque
data:
  existing-key: ZXhpc3RpbmctdmFsdWU=  # Kodirano "existing-value"
stringData:
  new-key: new-value                  # Ovo će biti kodirano u Base64
  existing-key: overridden-value      # Overridaće vrednost iz `data`
---
...
    spec:
      containers:
      - name: my-app
        image: busybox
        command: ["sh", "-c", "echo Existing Key: $EXISTING_KEY && echo New K
        env:
        - name: EXISTING_KEY
          valueFrom:
            secretKeyRef:
              name: combined-secret
              key: existing-key
        - name: NEW_KEY
          valueFrom:
            secretKeyRef:
              name: combined-secret
              key: new-key
```

Namespace

- **Namespace** u Kubernetesu je **logička particija** unutar klastera koja omogućava razdvajanje i organizaciju resursa, poput pod-ova, servisa, i drugih objekata.
- Omogućava različitim timovima ili aplikacijama da dele isti klaster bez međusobnog konflikta.
- Olakšava upravljanje resursima grupisanjem po timovima, aplikacijama, ili okruženjima (npr. dev, staging, prod).
- Pomoću RBAC (Role-Based Access Control) možete ograničiti pristup određenim Namespace-ovima.
- Ako se **ne definiše Namespace** u YAML manifestu ili komandi, Kubernetes podrazumevano koristi Namespace **default**.
- Komanda za prikazivanje podova iz svih namespace-ova: `kubectl get pods -A`
- Komanda za prikazivanje podova iz određenog namespace-a: `kubectl get pods -n <namespace-name>`

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-bez
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app-bez
  template:
    metadata:
      labels:
        app: my-app-bez
    spec:
      containers:
      - name: my-app-bez
        image: nginx
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  namespace: my-namespace
spec:
  ... #isto kao u prethodnom Deploymentu
```

kubectl [create] <objekat> <naziv_objekta>

- Kreira resurs na osnovu fajla ili stdin.
- Dokumentacija sa primerima:
  - https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#create

kubectl [get] <objekat> <naziv_objekta>

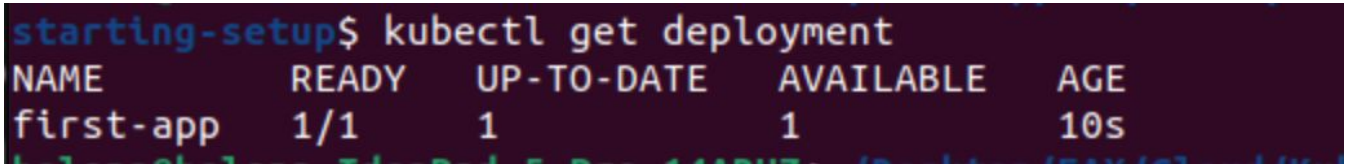
- Prikazuje jedan ili više resursa
- Dokumentacija sa primerima:
  - https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get

kubectl [delete] <objekat> <naziv_objekta>

- Briše navedene resurse se klastera
- Dokumentacija sa primerima: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#delete

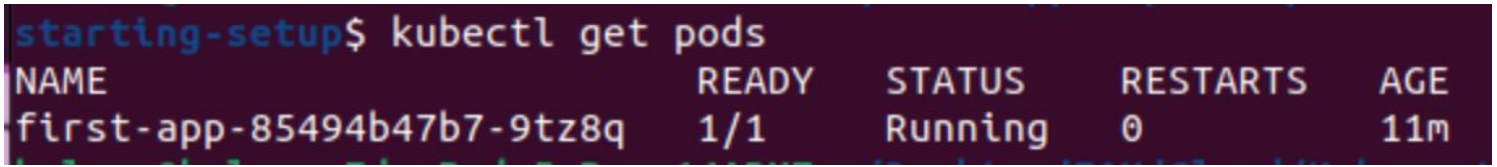
Primer 1

- `kubectl create deployment first-app --image=cloudftn/kub-first-app`
  - naredba za kreiranje deployment objekta pod nazivom first-app
  - first-app objekat se šalje na Master Node
  - scheduler analizira postojeće Worker Node-ove i odlučuje na kom čvoru će kreirati novi Pod
  - odabranom Worker Node-u se šalje informacija o podizanju jednog Poda unutar koga će se nalaziti kontejner kreiran na osnovu navedene slike
- `kubectl get deployment`
  - naredba za prikaz deployment objekta



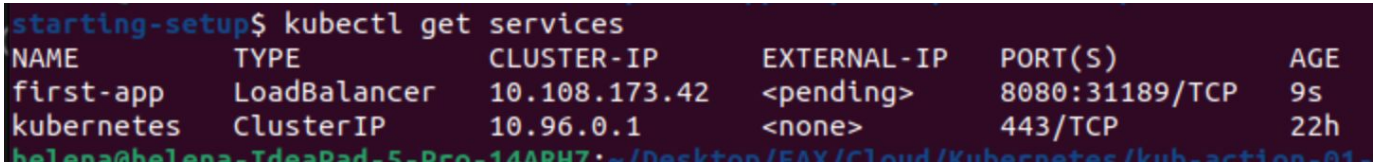
Primer 1

- `kubectl get pods`



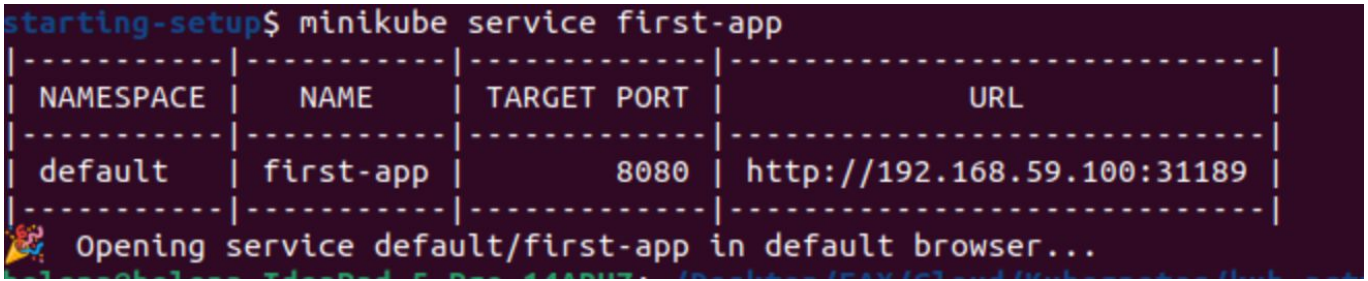
Primer 1

- `kubectl expose deployment first-app --type=LoadBalancer --port=8080`
  - naredba za izlaganje Pod-a spoljnoj sredini tako što kreira servis
  - –port na kome se nalazi pokrenuti kontejner
  - –type
    - ClusterIP - predefinisan tiip, omogućava pristup samo unutar Klastera
    - NodePort - omogućava pritup i sa spolja
    - LoadBalancer - ravnomerno rasporedjivanje saobracaja na sve Podove unutar klastera



Primer 1

- `minikube service first-app`



Primer 1

- `kubectl scale deployment/first-app --replicas=3`

```
starting-setup$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
first-app-85494b47b7-9tz8q   1/1     Running   0          35m
first-app-85494b47b7-lb2tw   1/1     Running   0          9s
first-app-85494b47b7-t6tw9   1/1     Running   0          9s
```

Primer 1

- `kubectl set image deployment/first-app \`
  `kub-first-app=cloudftn/kub-first-app:v1`

Primer 1[deployment.yaml]

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: second-app-deployment
spec:
 replicas: 1
 selector:
   matchLabels:
     app: second-app
 template:
   metadata:
     labels:
       app: second-app
   spec:
     containers:
       - name: second-node
         image: cloudftn/kub-first-app
```

Primer 1

- `kubectl apply -f=deployment.yaml`

Primer 1 [service.yaml]

```
apiVersion: v1
kind: Service
metadata:
 name: backend
spec:
 selector:
   app: second-app
 ports:
   - protocol: 'TCP'
     port: 80
     targetPort: 8080
 type: LoadBalancer
```

Primer 1

- `kubectl apply -f=service.yaml`

Dodatni materijali

- https://github.com/ramitsurana/awesome-kubernetes
- https://cloud.google.com/kubernetes-engine/kubernetes-comic/
- https://kubernetes.io/docs/home/