KUBERNETES

Uvod

- Odprtokodna platforma za avtomatizirano nameščanje, skaliranje in upravljanje s vsebniki.

- Ustvarjen za interno uporabo v podjetju Google, trenutno razvoj poteka pod okriljem Cloud Native Computing Foundation.

- Platforma omogoča izvajanje **vsebnikov** *(containers)* na **gruči gostiteljev** *(nodes)*.

- Platforma omogoča deklarativen opis stanja gruče s pomočjo visokonivojskih konceptov (Deployment, Replica Set, …)

- Kubernetes vsebnike samodejno razporedi na ustrezne gostitelje, da zagotovi potrebo po virih in drugih omejitvah ter pri tem ne žrtvuje dostopnosti.

Uvod

- Kubernetes
  - v primeru izpada vsebnika poskrbi za njegov vnovični zagon in
  - v primeru izpada gostitelja poskrbi za prerazporeditev vsebnikov.

- Poskrbi tudi za odstranitev vsebnikov, ki se ne odzivajo na preverjanje vitalnosti *(health check)* in jih ne oglašuje drugim komponentam, če ti niso na voljo.

- Omogoča samodejno horizontalno skaliranje vsebnikov glede na porabo CPU in vrednost ostalih metrik.

- Kubernetes poskrbi za izenačevanje obremenitve med vsebniki in implementira lastne mehanizme za odkrivanja storitev *(service discovery)*.

Uvod

- Podpira mehanizme za progresivno posodabljanje aplikacij in njihovih nastavitev pri čemer spremlja njihovo delovanje in poskrbi, da ne pride do hkratnega izpada vseh instanc.

- Omogoča hranjenje in upravljanje varnostnih podatkov *(secrets)* in konfiguracijo aplikacij – odstrani potrebo po ponovni gradnji slik v primeru sprememb.

- Samodejni priklop podatkovnih shramb – lokalnih shramb, NFS, iSCSI, Gluster, Ceph, Cinder in Flocker.

Osnovni koncepti

- **Gostitelj *(Node)*** – vozlišče v gruči, na njem se izvajajo stroki. Praviloma ima Kubernetes gruča več gostiteljev.

- **Strok *(Pod)*** – skupina enega ali več vsebnikov. Strok predstavlja atomarno enoto – vsi vsebniki v podu se izvajajo na istem gostitelju.

- **Označbe *(Labels)*** – oznake, s katerimi identificiramo stroke in jih enostavno grupiramo.

- **Namestitev *(Deployment)*** – definira predlogo za kreiranje želenega števila replik stroka. Namestitev omogoča varno nadgradnjo izvajajočih strokov.

Osnovni koncepti

- **Storitev *(Service)* –** Abstrakcija, ki izpostavi večje število replik stroka.
  - Ne glede na to, kje se nahajajo stroki, je storitev dostopna na statičnem naslovu IP, sami podi pa se lahko dinamično premeščajo.

- **Preverjanje zdravja *(Health checking)* –** sistem za periodično preverjanje dostopnosti aplikacije in za samodejno nadomeščanje nedelujočih replik.

- **Replikacijski niz *(Replica Set)* –** zagotavlja, da v nekem času teče zahtevano število replik stroka.

- **Replikacijski nadzornik *(Replication Controller)* –** zagotavlja, da v nekem času teče zahtevano število replik stroka. (Nadomešča ga replikacijski niz).

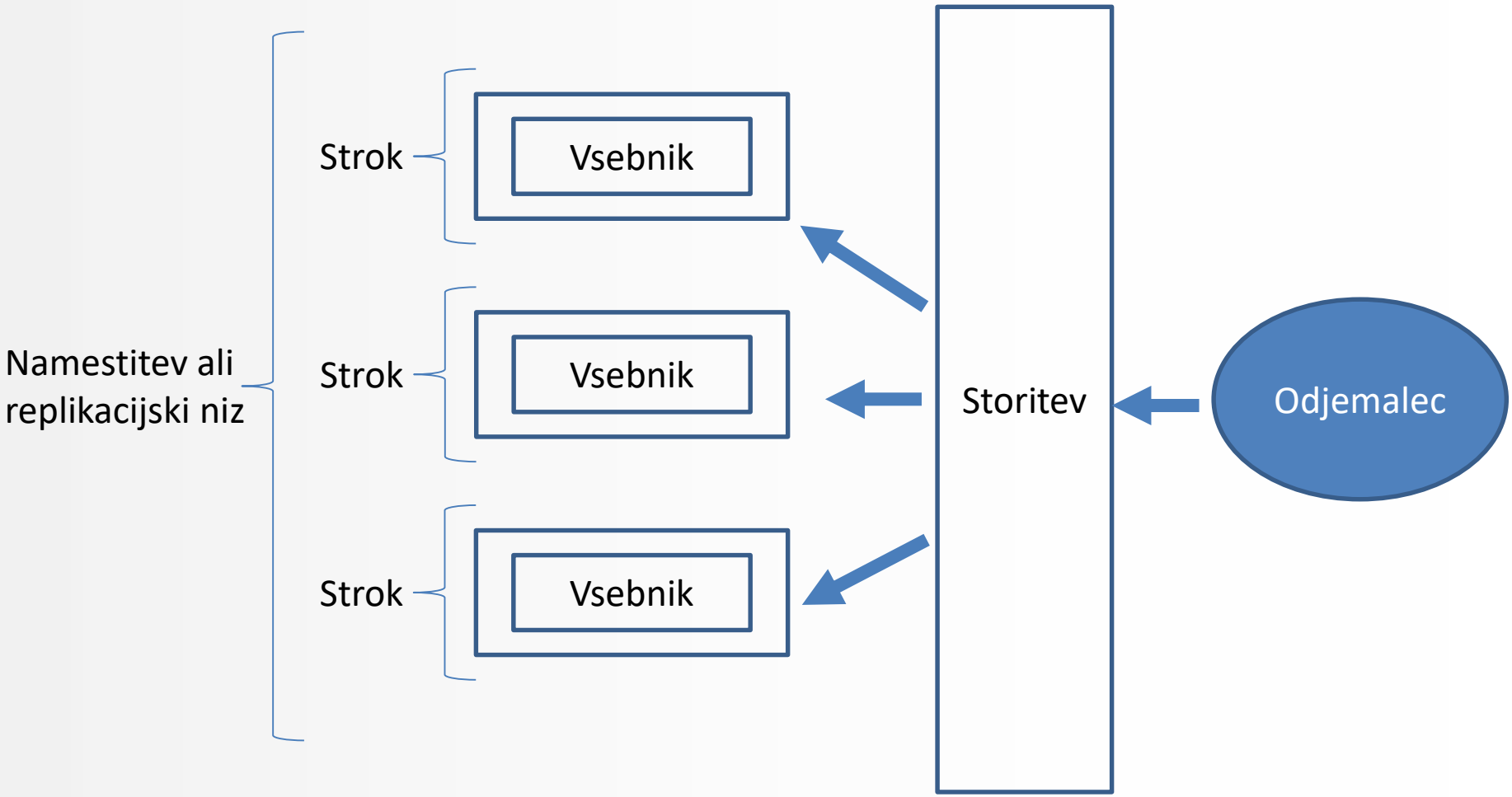
Osnovni koncepti

- **Nosilec podatkov *(Volume)* –** imenik v vsebniku, ki se preslika v imenik na gostiteljevem datotečnem sistemu.

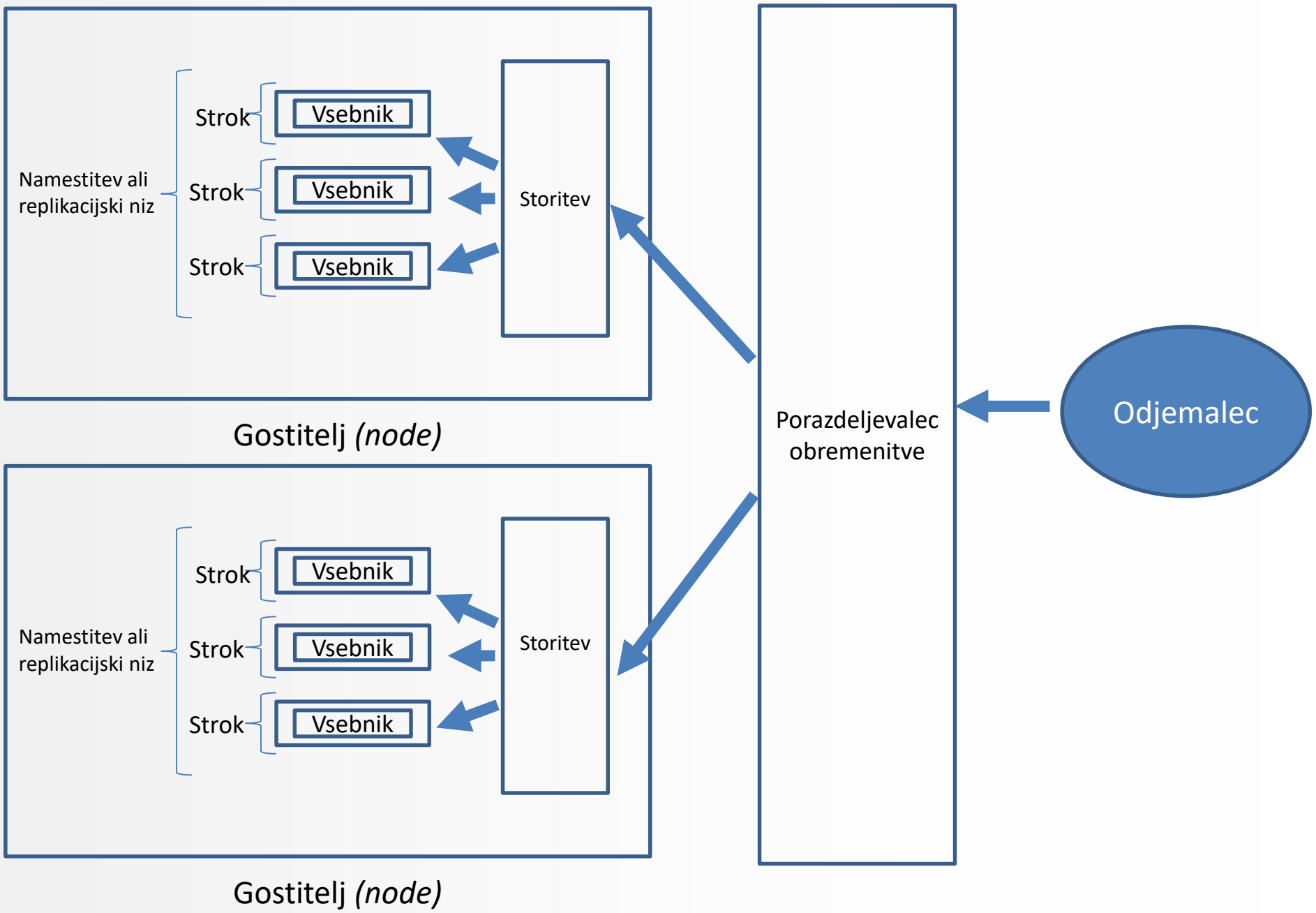
- **Skrivnosti *(Secret)* –** hrani občutljive podatke, kot npr. žetone za avtentikacijo ali podatke za dostop do privatnega Docker repozitorija.

- **Imenski prostor *(Namespace)*** – predpona imen virov, ki omogoča deljenje gruče in organizacijo projektov. Kubernetes podpira tudi omejevanje virov – virtualne gruče.

Shema osnovnih konceptov



Shema osnovnih konceptov



Namestitvene enote v Kubernetes

- Kubernetes podpira naslednje namestitvene enote:
  - **Strok** *(Pod)*
  - **Replikacijski niz** *(Replica set)* (next-generation Replication Controller)
  - **Namestitev** *(Deployment)* (next-generation Replication Controller)
  - **Replikacijski nadzornik** *(Replication controller)*
  - in ostale : Jobs, Pet Sets, Deamon Sets

- Za tip namestitvene enote se odločimo na podlagi potreb in funkcionalnosti, ki jih nudi posamezen tip.

Namestitvena enota strok *(Pod)*

- **Strok *(Pod)*** je najmanjša namestitvena enota, sestavljena iz:
  - enega ali več vsebnikov in
  - deljene shrambe med vsebniki znotraj stroka.

- Vsebniki znotraj stroka si delijo IP naslov in vrata.

- Vsebniki med sabo lahko komunicirajo preko *localhost.*

- **Stroki** niso odporni na:
  - napake v času razporejanja (scheduling),
  - izpad gostitelja,
  - In drugih napak zaradi vzdrževanja ali pomanjkanja virov.

Namestitvena enota strok *(Pod)*

- V splošnem strokov nikoli ne kreiramo neposredno.

- Kreiramo jih z uporabo namestitvenih enot tipa namestitev *(Deployment)*, ki doprinesejo:
  - zmožnost samodejne vzpostavitve ob izpadu stroka,
  - upravljanje replik in
  - “rollout” namestitve in posodobitve.

Namestitvena enota tipa namestitev *(Deployment)*

- **Namestitev *(Deployment)*** omogoča deklarativno posodabljanje strokov in replikacijskih nizov.

- Objektu tipa **namestitev** določimo želeno stanje, *Deployment Controller* pa bo poskrbel, da se dejansko stanje ustrezno posodobi.

- Tipično namestitev uporabimo ko želimo:
  - namestiti stroke in replikacijske nize,
  - možnost preverjanja uspešnosti namestitve,
  - možnost posodobitve namestitve z novimi verzijami slike vsebnikov in
  - možnost povrnitve v prejšnje (stabilno) stanje.

Namestitvena enota replikacijski nadzornik *(Replication Controller)*

- **Replikacijski nadzornik *(Replication Controller)*** zagotovi, da v nekem trenutku teče zahtevano število replik nekega stroka.

- Če je strokov preveč, replikacijski nadzornik poskrbi za odstranitev odvečnih, če jih je premalo, poskrbi za zagon preostalih.

- Podi, ki jih ustvari replikacijski nadzornik, so samodejno nadomeščeni v primeru:
  - izpada zaradi napake pri delovanju,
  - da je strok pobrisan ali
  - je bil ustavljen.

- Replikacijski nadzornik je priporočen tudi za aplikacije, ki so sestavljene le iz enega poda.

Namestitvena enota replikacijski nadzornik *(Replication Controller)*

- Replikacijski nadznornik uporabimo kadar želimo:
  - zanesljivo delovanje strokov tudi v primeru izpadov *(self healing)*
  - podporo skaliranja strokov in
  - podporo za *rolling updates.*

Namestitvena enota replikacijski niz *(Replica Set)*

- Replikacijski niz (RS) nadomešča replikacijski nadzornik (RC).

- Razlika med RS in RC zajema:
  - podporo za *selector*:
    - RS podpira *selector* na osnovi set-ov.
    - RC podpira *selector* le na osnovi enakosti**.**
  - podporo za rolling update:
    - V primeru potrebe je priporočena uporaba namestitvene enote tipa namestitev, ki za razliko od RS omogoča deklarativno nadgradnjo.

- Replikacijski niz uporabimo v primeru:
  - ko imamo zahtevo, da v določenem trenutku teče določeno število podov,
  - ni potrebe po posodobitvah.

Konfiguracija namestitvene enote

- Konfiguracijo namestitvene enote podamo z namestitveno datoteko v **json** ali **yaml** formatu.

- Zaradi enostavnosti je priporočena uporaba **yaml** formata.

- Splošna priporočila:
  - konfiguraciji vedno določimo zadnjo stabilno verzijo API-ja (trenutno **v1**),
  - konfiguracijske datoteke hranimo v repozitoriju s podporo nadzora različic (za primere povračanja stanja),
  - povezane objekte združimo v eno namestitveno datoteko zaradi lažjega upravljanja,
  - privzetih vrednosti ne prepisujemo, če za to ni potrebe (zaradi preglednosti).

Konfiguracija samodejnega skaliranja

- Podobno, kot konfiguriramo namestitvene enote, lahko konfiguriramo ***Horizontal Pod Autoscaler.***
  - Orodje **kubectl** ima zelo priročen ukaz **autoscale**, ki poenostavi konfiguracijo**.**
- *HPA* lahko definiramo za **Deployment**, **ReplicaSet** ali **ReplicationController.**
- Prikažemo obstoječe HPA-je:
  - **kubectl get hpa**
- Podrobnosti HPA-ja**:**
  - **kubectl describe hpa**

Konfiguracija samodejnega skaliranja

- Kreiranje *HPA* za namestitev *(Deployment):*
  - **kubectl autoscale deployment <deployment-name> --min=2 --max=10**
  - Parameter *max* je obvezen.

- Določimo lahko tudi pri kateri porabi cpu se ustvari nova instanca:
  - **kubectl autoscale deployment <deployment-name> --min=2 --max=10 *--cpu-percent=80***

- *Predpogoj za delovanje je nameščen Heapster monitoring.*

Progresivno posodabljanje

- Posodobitev namestitve sprožimo s spremembo konfiguracije *.spec.template*
  - Tipično gre za spremembo slike – nova verzija.

- Primer posodobitve slike:
  - **kubectl set image deployment/customers-deployment *customers=customers:1.01***

- Status *rollout* operacije spremljamo z ukazom***:***
  - **kubectl rollout status deployment/customers-deployment**

- Pregled zgodovine ***rollout*** za namestitev***:***
  - **kubectl rollout history deployment/customers-deployment**

Progresivno posodabljanje

- Prehod na prejšnjo verzijo izvedemo z ukazom:
  - **kubectl rollout undo deployment/customers-deployment**

- Prehod na točno določeno revizijo:
  - **kubectl rollout undo deployment/customers-deployment *--to-revision=2***

Storitev *(Service)*

- Problem strokov je njihova umrljivost. Ko umrejo niso več dostopni.
- Po drugi strani imamo nadzornike replik, ki dinamično zaganjajo in odstranjujejo stroke – ob skaliranju ali ob izvajanju postopne posodobitve.
- Čeprav vsak strok prejme svoj IP naslov, se ne moremo zanašati na to, da bo IP naslov stabilen.
- **Problem**: če nek strok izpostavlja neko storitev drugemu stroku, lahko pride do tega, da strok *izgubi* naslov stroka, katerega storitve uporablja.

Storitev *(Service)*

- Omenjen problem rešujejo storitve *(Services)*.
- Kubernetes storitev je abstrakcija, ki definira logičen nabor strokov in politiko dostopa.
- Če se kateri od zalednih strokov spremeni, ta sprememba ne vpliva na odjemalce.
- Storitev definiramo podobno kot za stroke in namestitve – preko konfiguracijske datoteke v formatu **yaml** ali **json**.

Storitev *(Service)*

- Poznamo različne tipe storitev

  - **ClusterIP** – storitev je na voljo na IP naslovu znotraj gruče. To je privzeta vrednost.

  - **NodePort** – storitev je izpostavljen na vratih na IP naslovu gostitelja (na vsakem gostitelju v gruči ločeno).

  - **LoadBalancer** – storitev je na voljo na ločenem porazdeljevalcu obremenitve *(load balancer)*. Na voljo samo pri določenih ponudnikih (npr. Google Cloud Platform).

Kubernetes gruča – komponente

- Kubernetes **master**
  - Vozlišče, ki vsebuje tudi komponente za upravljanje gruče *(master components)*
  - Vsebuje orodja, ki skrbijo za upravljanje gruče
    - kube-apiserver
    - etcd
    - kube-controller-manager
    - cloud-controller-manager
    - kube-scheduler
    - addons

- Kubernetes **node**
  - Ena ali več instanc.
  - Vsebujejo orodja z izvajalnim okoljem
    - kubelet
    - kube-proxy
    - docker
    - rkt
    - Supervisord
    - fluentd

Kubernetes gruča – komponente

- kube-apiserver
  - Izpostavlja Kubernetes API, ki omogoča nadzor nad gručo
  - Hrani konfiguracijo in stanje celotne gruče (vseh objektov) – strokov, storitev, nadzornikov replik

- kube-controller-manager
  - Demon, ki izvaja jedrne kontrolne zanke, ki nadzirajo stanje gruče preko apiserverja in skrbijo za prehod gruče iz trenutnega stanja v želeno stanje.
  - Kontrolne zanke: replication controller, endpoints controller, namespace controller in serviceaccounts controller.

Kubernetes gruča – komponente

- kube-scheduler
  - Izvaja funkcijo iskanja in določanja vozlišča na katerem se bo izvajal strok.
  - Izbira vozlišča poteka v treh korakih: filtriranje, razvrščanje po prioritetnih funkcijah in končna izbira.

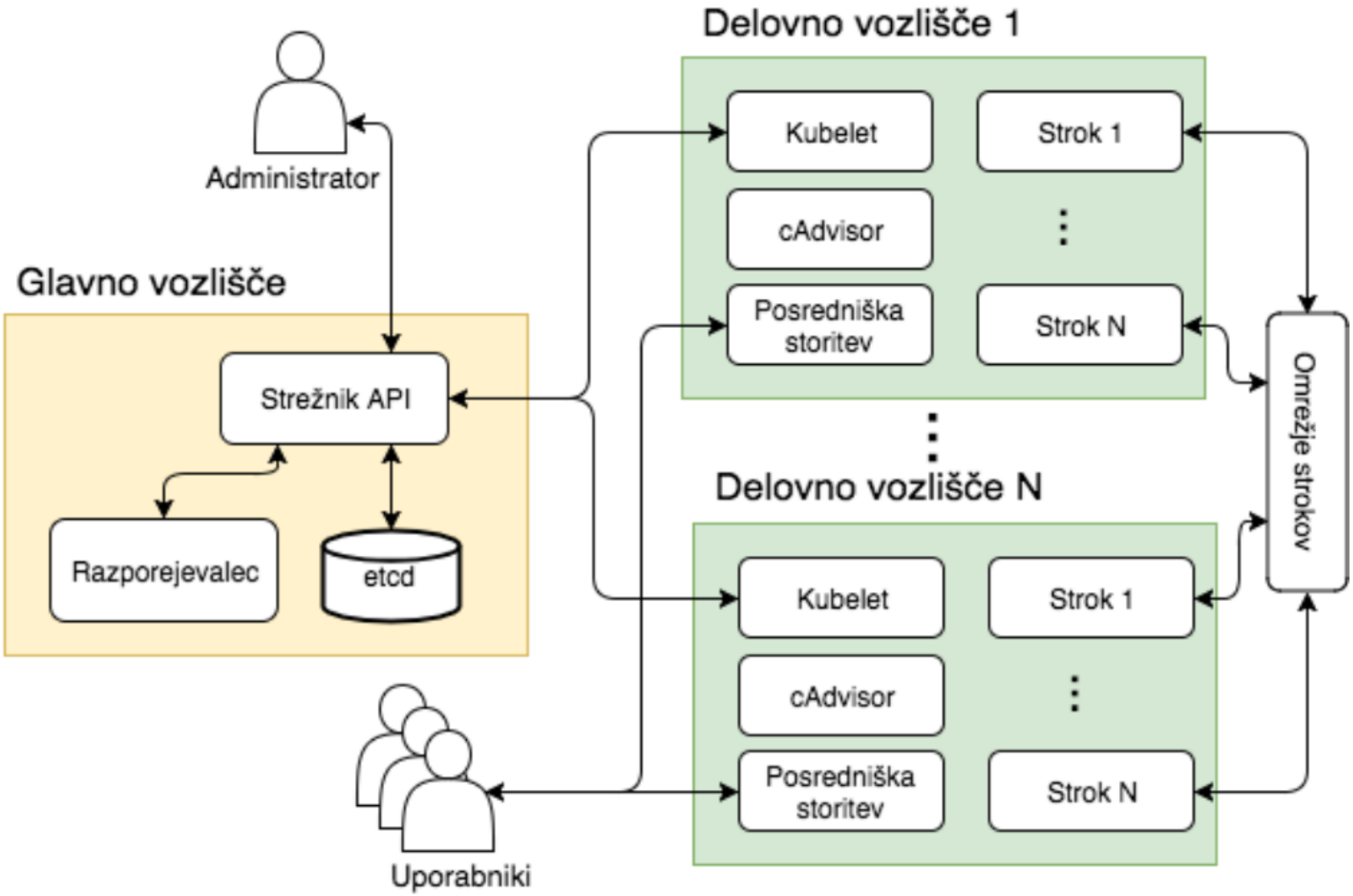
- DNS
  - Dodatek *(addon)*, ki ga je priporočljivo namestiti v vsako gručo
  - Deluje kot DNS strežnik za nameščene storitve *(services)*

- kube-proxy
  - Izvaja se na vseh vozliščih in izpostavlja storitve kot so definirane v apiserverju.
  - Izvaja naključno (ali round robin) posredovanje TCP, UDP tokov.

Kubernetes gruča – komponente

- kubelet
  - Primarni agent vsakega vozlišča, ki je odgovoren za vzdrževanje strokov na vozlišču.
  - Agent skrbi, da je izvajanje strokov v skladu z definicijo *PodSpec* (objekt definiran v JSON ali YAML konfiguracijski datoteki).
  - Manifest – *PodSpec* agentu posreduje Apiserver, lahko pa je konfiguriran (kubelet), da samodejno preverja vsebino mape ali vsebino na URL naslovu.

Postavitev Kubernetes gruče



Orodje *kubectl*

- ***kubectl*** je orodje s katerim upravljamo z gručo.

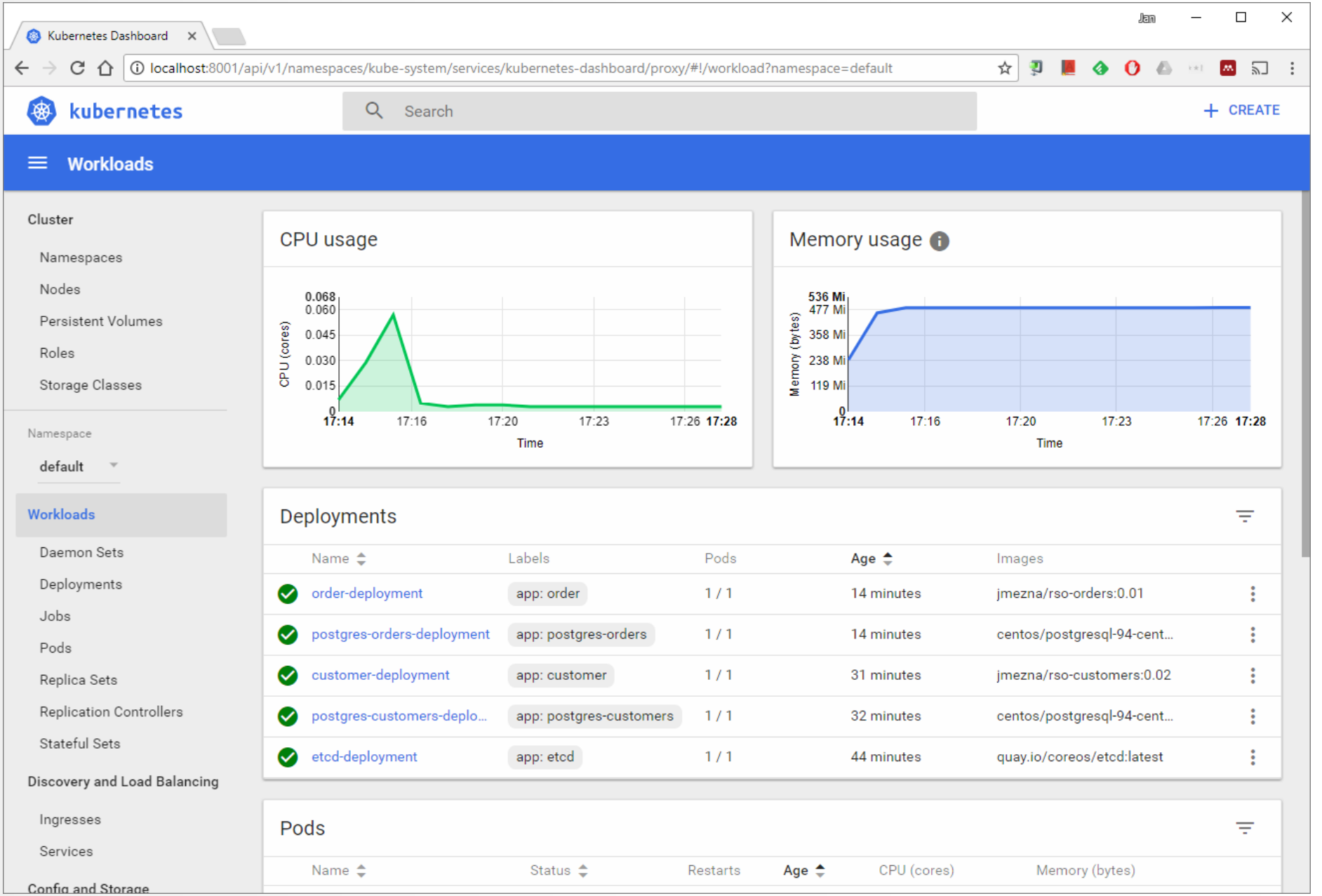
- Z orodjem dostopamo in upravljamo vire preko apiserverja:
  - Za dostop potrebujemo
    - ***ca.cert*** – *korenski javni ključ*
    - ***admin.pem*** – *javni ključ admina*
    - ***admin-key.pem*** – *privatni ključ admina.*

- Ukaz za izpis trenutne konfiguracije ***kubectl config view.***
  - Konfiguracija se hrani v datoteki **~/.kube/config**

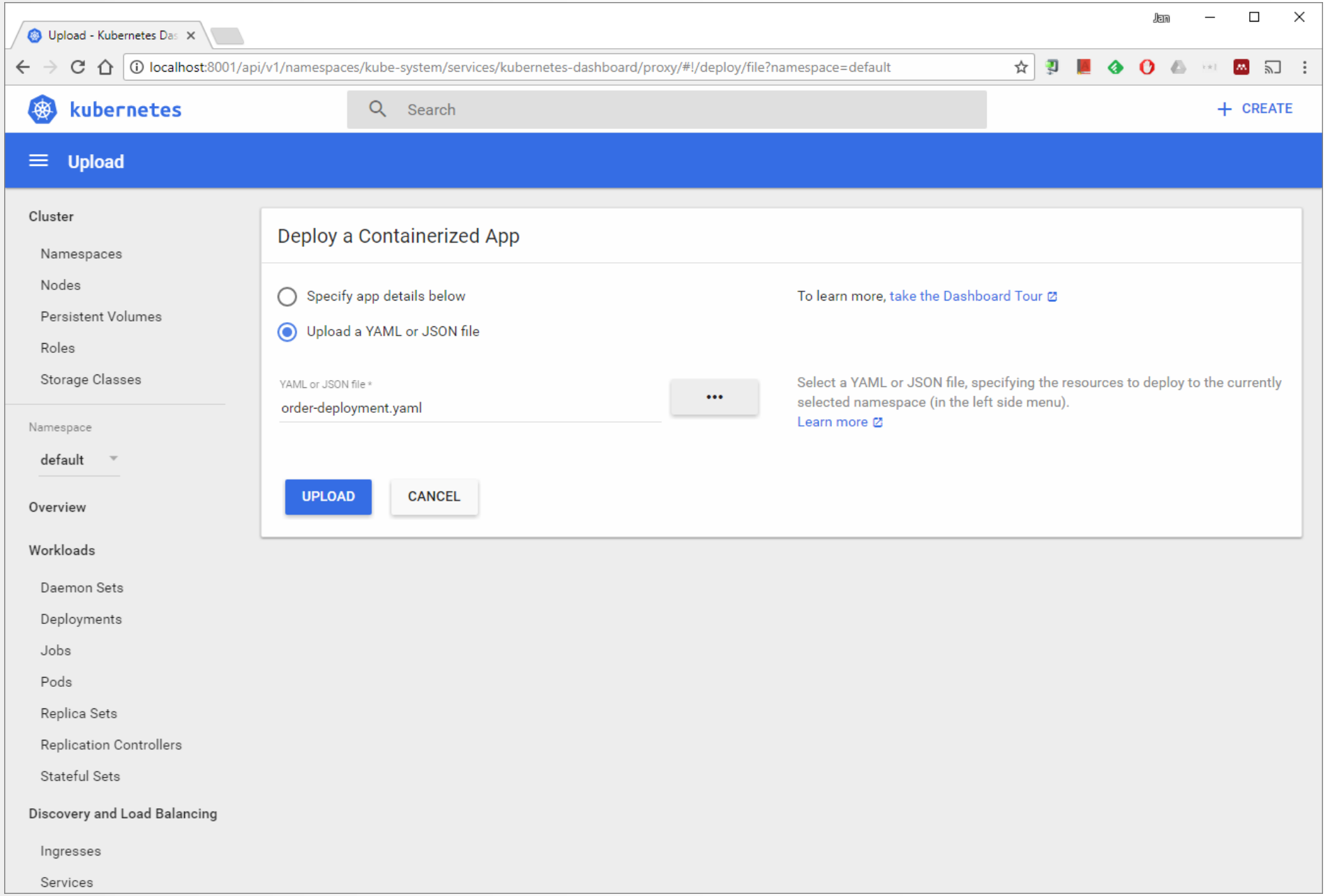
Orodje *kubectl* – primeri ukazov

- ***kubectl create -f customers-deployment.yml*** – ustvari namestitev
- ***kubectl create -f customers-rs.yml*** – ustvari replikacijski niz
- ***kubectl get pods*** – prikaži vse stroke
- ***kubectl describe pod customers-pod*** – prikaži opis stroka
- ***kubectl proxy*** – zaženi proxy za dostop do grafičnega vmesnika *(Dashboard)*
- ***kubectl –help*** – pomoč in opis ostalih ukazov

Kubernetes Dashboard



Kubernetes Dashboard



Namestitev

- Kubernetes lahko namestimo na lastne strežnike

- Nekateri ponudniki oblačnih storitev ponujajo Kubernetes kot storitev
  - Google Container Engine
  - Azure Container Service
  - OpenShift Online
  - IBM Cloud Container Service
  - …

- Za razvoj in testiranje na razvojnih računalnikih lahko uporabimo *minicube* – virtualizirana rešitev, podobno kot Docker Toolbox

Google Container Engine

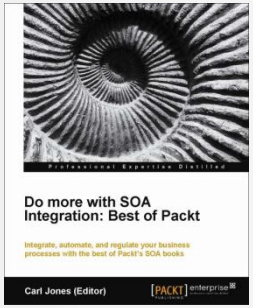
- Omogoča kreiranje poizkusnega *(trial)* računa s 300$ dobroimetja.
- Pri registraciji zahteva podatke o kreditni kartici za verifikacijo posameznika.
  - Po preteku poizkusnega obdobja ne začne samodejno obračunavati storitev, ampak strežnike ugasne.
- Kubernetes gručo lahko namestimo na poljubne virtualne strežnike (v *trial* verziji je zmogljivost omejena).
- Omogoča avtomatsko postavitev porazdeljevalcev obremenitve *(load balancer).*
  - Pri nekaterih ponudnikih moramo za to poskrbeti sami.

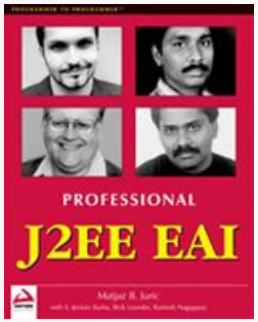
IBM Cloud

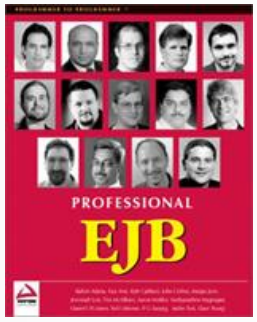
- Brezplačen račun za prvih 30 dni (brez kreditne kartice)
- Za študente brezplačno prvega pol leta
- Ponuja Kubernetes gručo z enim gostiteljem
  - en virtualen strežnik – 2 CPU, 4 GB RAM

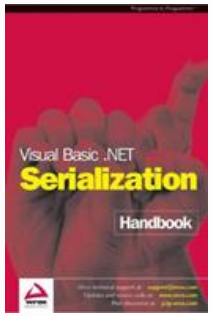
OpenShift Online

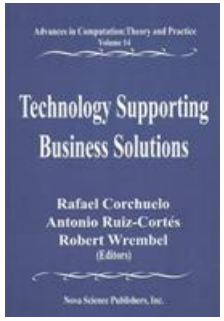
- Možnost kreiranja brezplačnega računa (brez kreditne kartice)
- OpenShift je posebna rešitev, ki
  - uporablja Kubernetes za upravljanje s vsebniki in
  - uporabniku ponuja lasten uporabniški vmesnik *(wrapper)* za upravljanje vsebnikov.

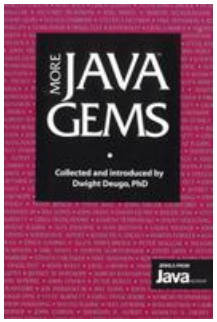












HVALA!

Java Champions · ORACLE ACE Director · IBM CHAMPION · ORACLE Certified Expert · Certified for IBM SOA technology · Certified for IBM CLOUD COMPUTING technology

e-naslov: **http://www.cloud.si**

e-naslov: **http://www.soa.si**

e-pošta: **info@cloud.si**