



UNIVERSITY OF TARTU

INSTITUTE OF COMPUTER SCIENCE











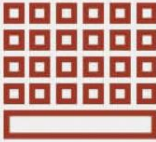



Mikroteenused & konteinerite halduse platvormid

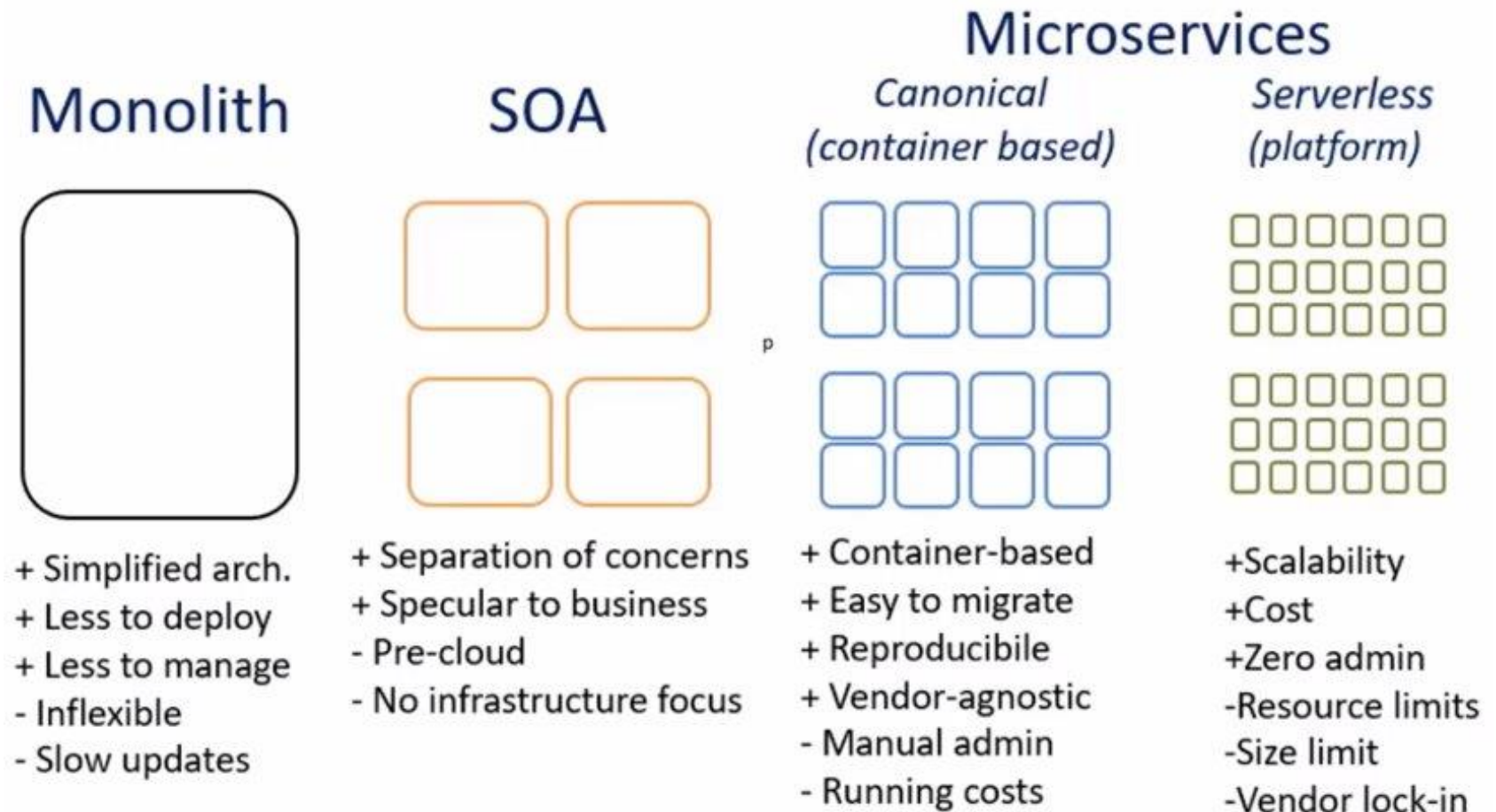
Pelle Jakovits

April 2025, Tartu

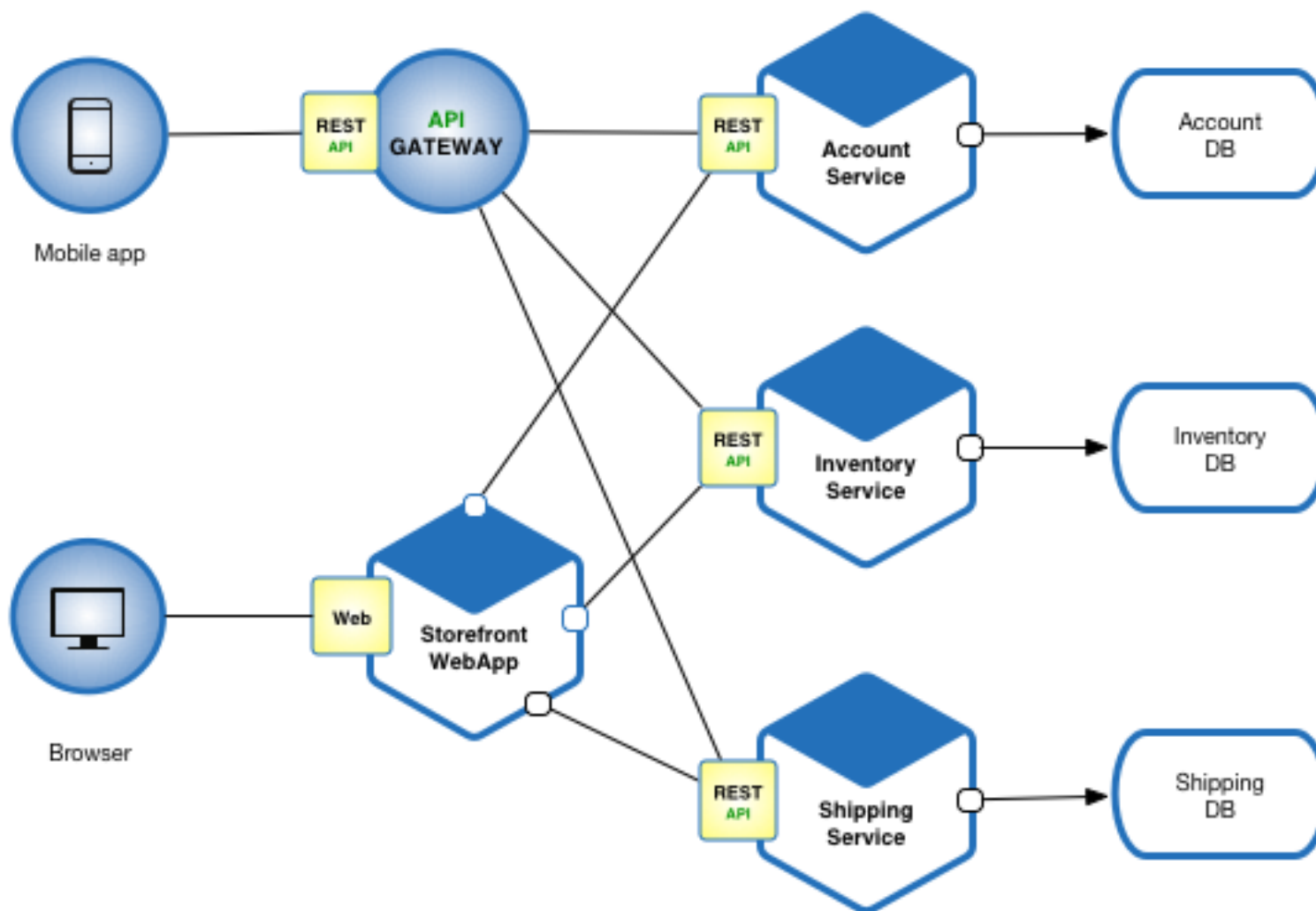
Rakenduste ja teenuste areng

	Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
~ 1980	Waterfall 	Monolithic 	Physical Server 	Datacenter 
~ 1990				
~ 2000	Agile 	N-Tier 	Virtual Servers 	Hosted 
~ 2010	DevOps 	Microservices 	Containers 	Cloud 

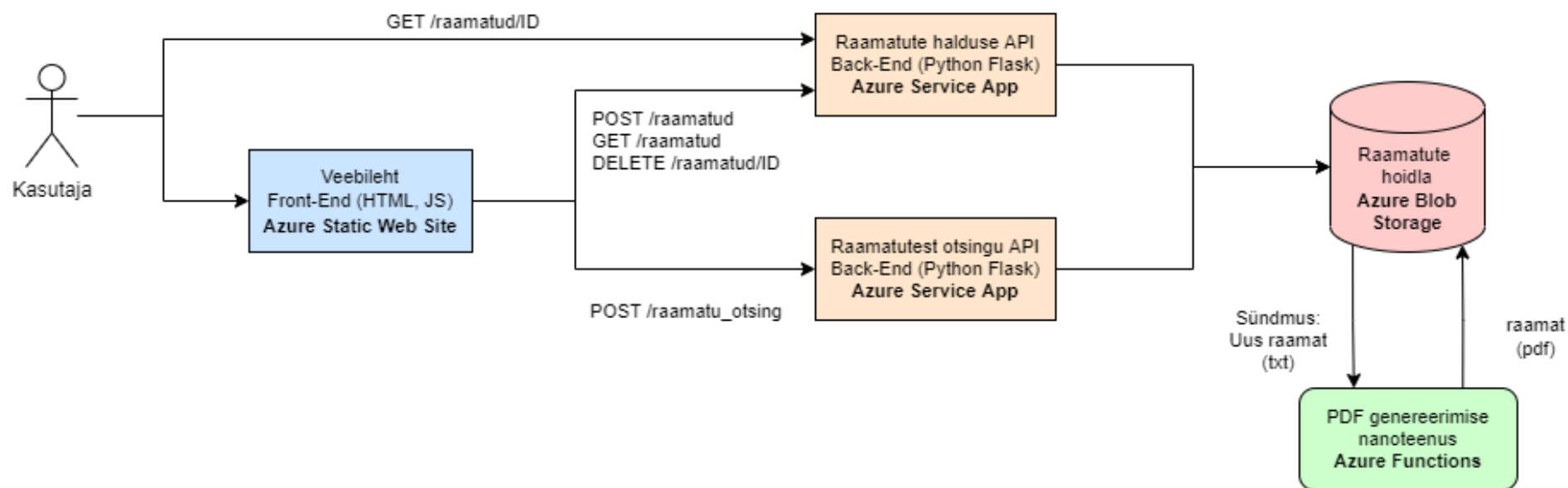
Monoliitsetest rakendustest nanoteenusteni



Mikroteenusete arhitektuuri näide



Praktikumi näide



Mikroteenuste eelised

- Võimaldab suurte rakenduste pidevat tarnimist ja juurutamist (CI/CD)
 - Parem hooldatavus
 - Parem testitavus
 - Kiirem juurutatavus
- Iga teenus on "piisavalt" väike
 - Lihtsam aru saada, uutele töötajatele tutvustada
 - IDE'sse ei pea importima ülisuuri projekte
 - Teenus alustab jooksmist kiiremini
- Parem isoleerimine vigade korral
- Vähendab tehnoloogia võlga, sõltuvust tehnoloogia valikutest

Mikroteenuste puudused

- Raskem aru saada kogu süsteemi hajutatud arhitektuurist
 - Vaja tegeleda teenuste vahelise suhtlusega
 - Päringud, mis vajavad mitme teenuse välja kutsumist, või andmeid, on keerulisemad arendada ja testida
 - Teenuste vaheliste interaktsioonide testimine on keerulisem
- Kogu süsteemi korraga juurutada, üles seada on keerulisem
- Suurem ressursside (eriti mälu) kasutus, kui igal teenusel on oma (mitte jagatud) keskkond
- Võib minna kallimaks

Konteinerite väljakutsed

- Konteineritel on nõrgem isolatsioon
- Salvestusruumi haldamine võib muutuda problemaatiliseks
 - Stateless konteinerid, konteinerite migratsioonid, katkestused
- Haldamine on keerulisem, kuna komponentide suurus väheneb ja nende arv suureneb
- Oluline on tagada, et seisakuid ei oleks, rikete tagajärjed on minimaalsed
- Süsteemid peaksid skaleerima elastselt
- Suurte konteineripõhiste süsteemide haldamise lihtsustamiseks ja automatiseerimiseks on vaja konteinerite orkestreerimise lahendusi
 - Nt Docker Swarm, Kubernetes

KUBERNETES

Kubernetes

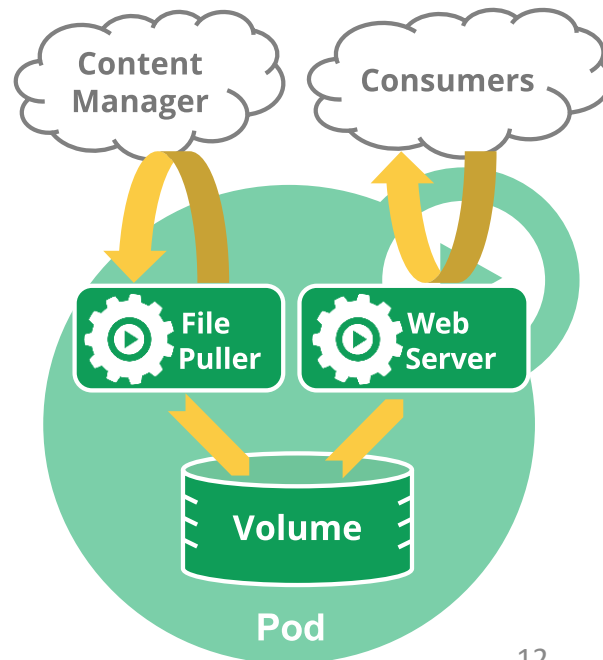
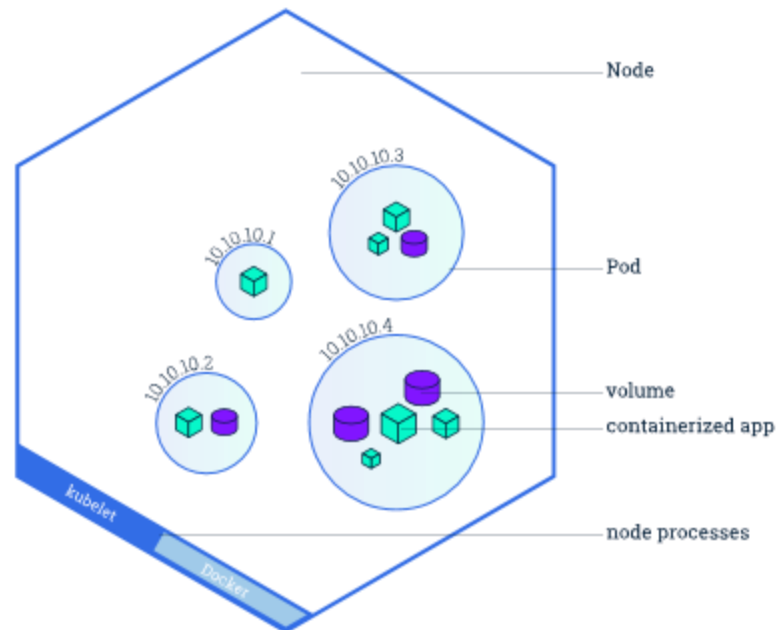
- Avatud lähtekoodiga platvorm konteinerite haldamiseks
- Põhineb Google Borg (~2004) projektil
- Väljaantud 2014 Juunis
- Suur arendajate kogukond, Uus versioon iga 3 kuu tagant
- Infrastruktuur as Code (IaaC).
- Kasutatakse sageli mikroteenuste juurutamise automatiseerimiseks ja skaleerimiseks

Kubernetes olemid

- Pod – Pod/Kaun
- Namespace - Nimeruumid
- Deployment - Juurutus
- ReplicaSet - replikaatide kogum
- Service – Teenus (võrk)
- Ingress - Sissepääsu kontrollerr
- Configmaps - Konfiguratsioonr kaarr
- Secret - Saladus

Kubernetes Pod

- Väikseim käivitavav k8s üksus
- Koosneb kontainerite grupist
- Igal Podid on oma unikaalne k8s sisemine IP aadress
 - Loogiline mikro-sõlm
- Skaleerimisel muudame Podi sisemiste koopiate arvu
- Kaks sama teenuse Podi töötavad tavaliselt erinevates sõlmedes
- Ainult tihedalt seotud konteinerid peaksid asetsema samas Podis
 - Kui on vaja ressursse (nt köiteid/Volume) jagada

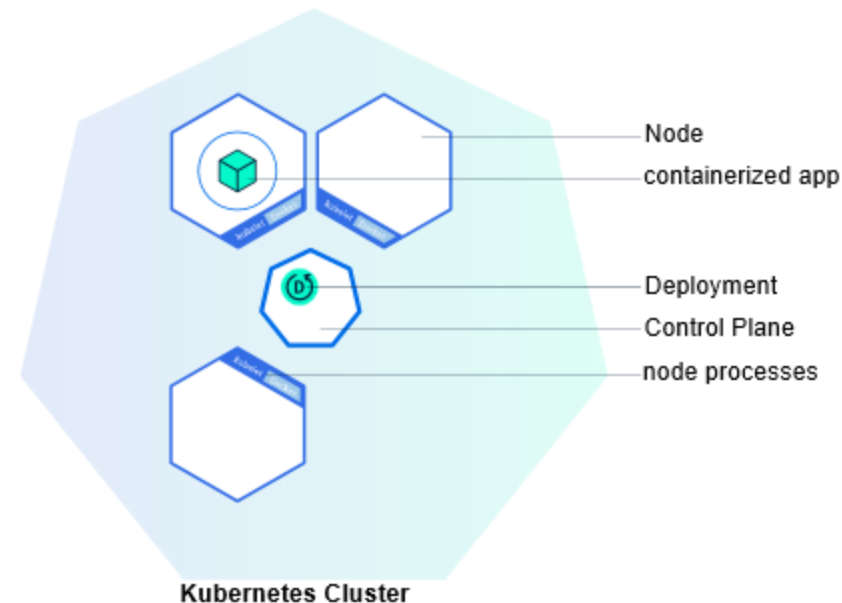


Kubernetes namespace

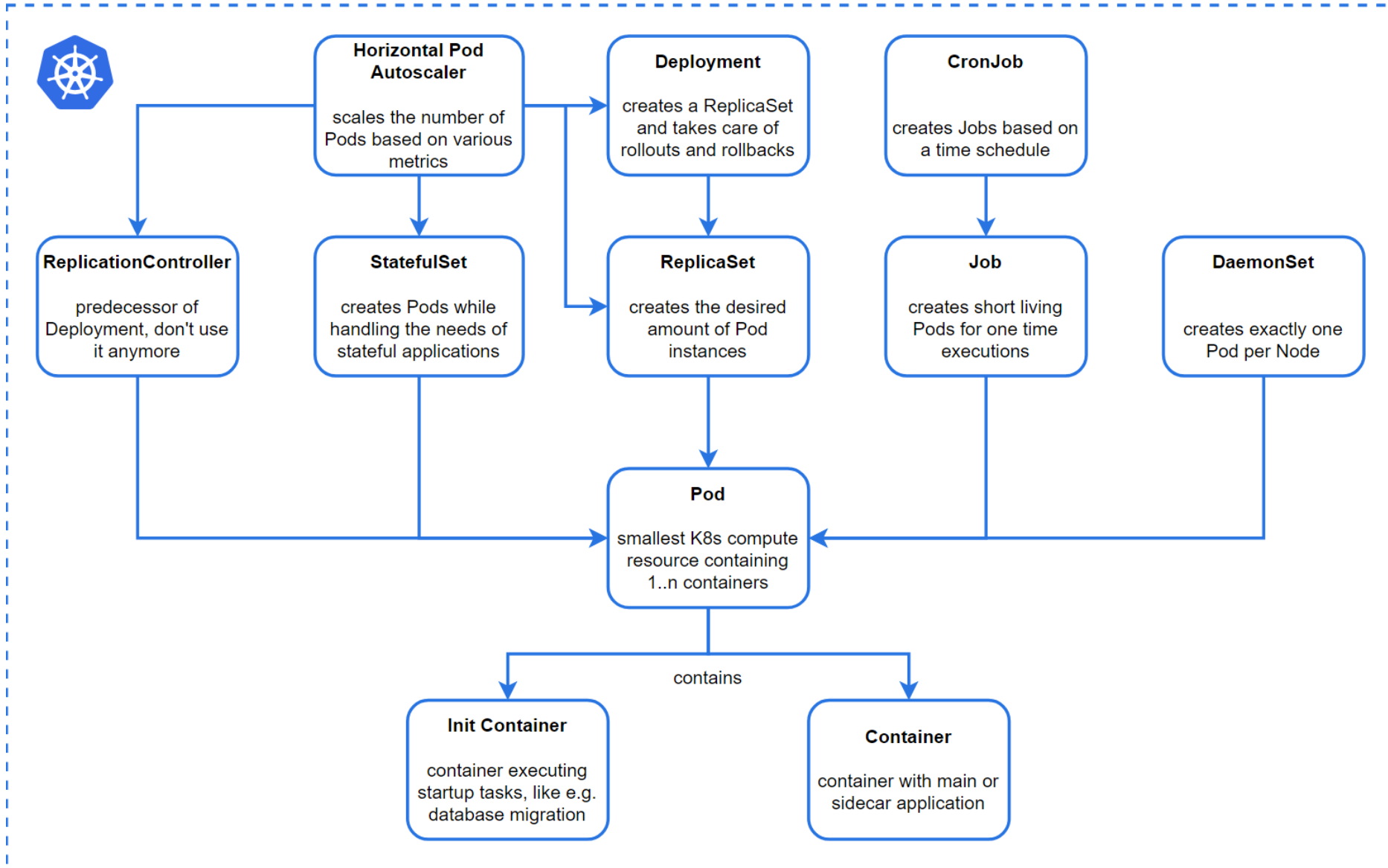
- Eraldab ressursside rühmad ühte gruppi
- Kubernetese nimeruumi sisesed ressursside nimed peavad olema unikaalsed
 - Kuid samu nimesid saab kasutada teistes nimeruumides
 - Meil võib olla kaks erinevat juurutust nimega "Postgres" kahes erinevas nimeruumis
- Võimaldab rakendusi või keskkondi eraldada ja isoleerida
 - Näiteks eraldi Tootmis ja testimis keskkondade nimeruumid
 - Saame kasutada täpselt sama nimega andmebaasi ja rakenduse Pod'e mõlemas nimeruumis

Juurutus - Deployment

- Sisaldab Pod'i malli
 - Määrab, mis on Pod'i soovitud olek
- Juurutust saab üles ja alla skaleerida
 - Kui see defineerib ka ReplicaSet'i
 - Muudab Podi koopiate arvu
- Saab välja lasta Podide uusi versioone
 - Podi versiooni saab "tagasi kerida"
- Deployment Controller teenus jälgib pidevalt juurutuste seisu

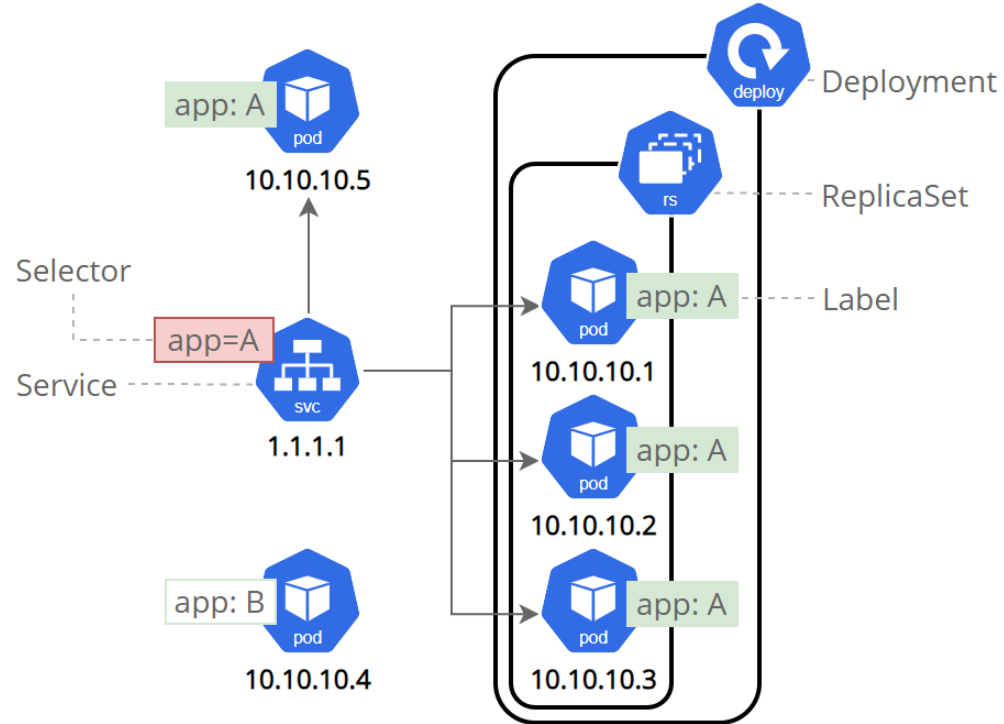


Kubernetes Workload Objects



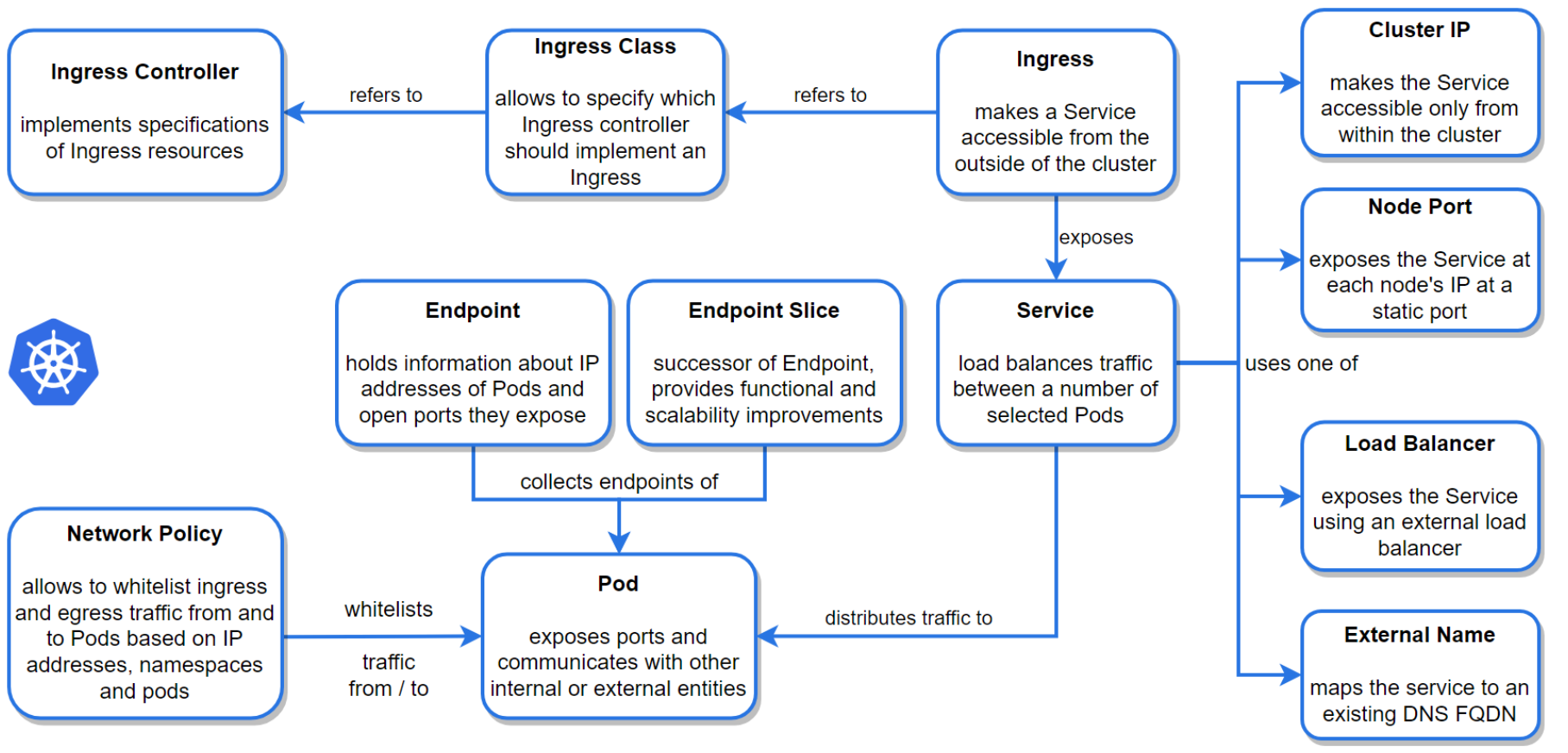
Teenused - Services

- Igal Podil on oma unikaalne sisemine IP aadress
 - Teenused määravad kuidas Pod'e grupeerida ja nendele ligi pääseda
- k8s teenus määrab, kuidas liiklus suunatakse Podide vahel
- Teenuste leidmine ja liikluse marsruutimine
- Teenus kasutab silte selle poolt hallatud Pod'ide valimiseks/määramiseks
- Sildid võivad defineerida:
 - Rakenduse nime
 - Rakenduse/Podi erinevad versioonid
 - Selle kas on tegemist, test, arendus või päris keskkonnaga
- Teenused defineerivad kuidas liiklust jagatakse Pod'ide vahel



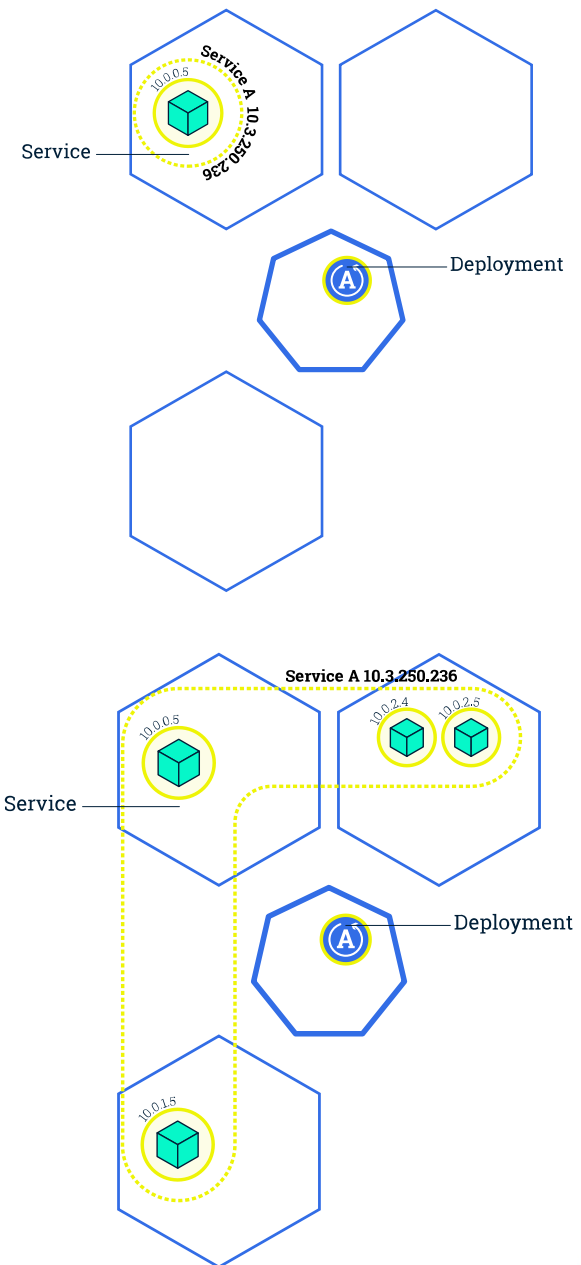
<https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/>

Kubernetes Networking Objects



Skaleerimine

- Peale Juurutuse defineerimist, saame selles olevaid Pod'e skaleerida
 - ReplicaSet märgib Pod'i skaleerivaks
 - Ei tohi replitseerida mitte skaleeruvaid teenuseid, näiteks andmebaasi
- Mitme koopia olemasolu tähendab, et Pod'l uut versiooni saab välja lasta ühe koopia haaval
 - Vältides teenuse katkestusi
- Teenust saab skaleerida 0-ni

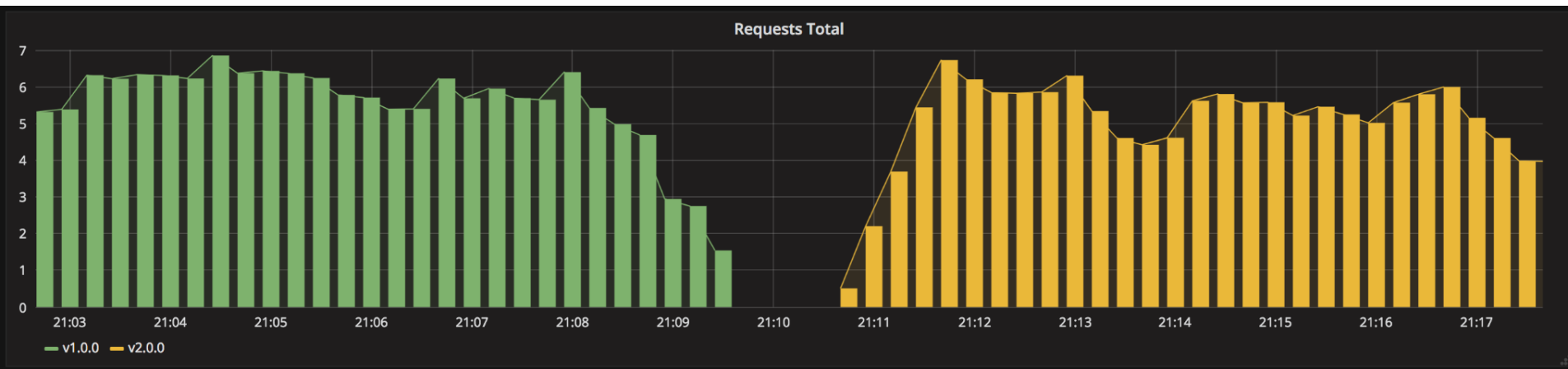


Mikroteenuste uuendamise strateegiad

- Kui tarkvara uus versioon välja lastakse, on töötava tarkvara versiooni värskendamiseks erinevad juurutuse strateegiad
- Kui kliendid kasutavad rakendust aktiivselt, soovime teenuse häireid minimeerida
- Soovime veenduda, et uuendamine õnnestuks ilma tõrgeteta
- Juurutusstrateegiad:
 - Recreate - Loo uuesti
 - Ramped - Kallutatud
 - Blue/Green
 - Canary
 - A/B testimine
 - Shadow

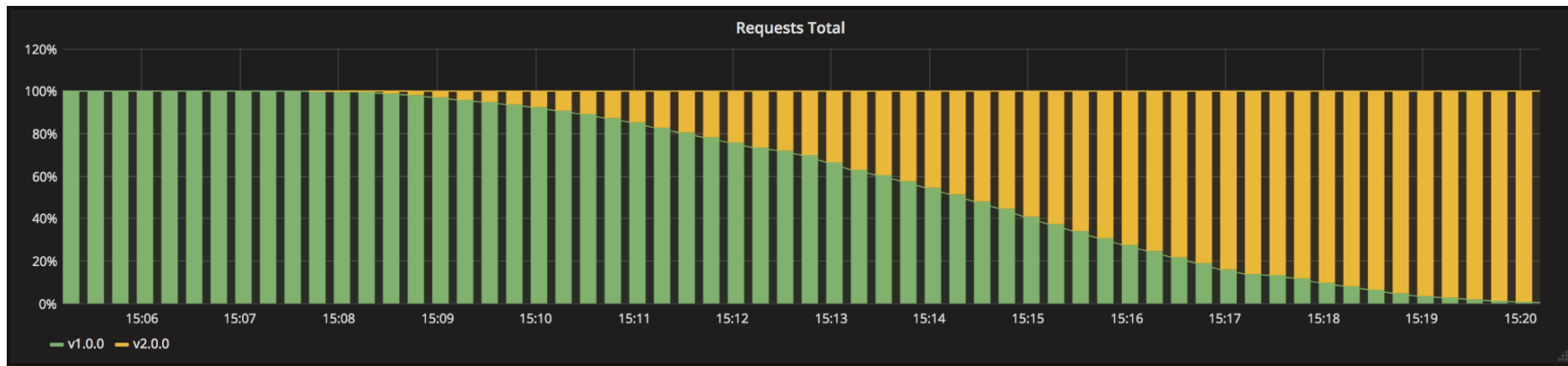
Recreate

- Peatame vanad juurutused enne uute juurutamist
- Põhjustab teenuse häireid – toimige siis, kui kasutajad süsteemi kõige vähem tõenäoliselt kasutavad
- Kasulik, kui
 - Ressursid on piiratud
 - Vähe aktiivseid kasutajaid
 - Kulude vähendamiseks
 - Kui tuleb tagada, et kahte erinevat versiooni korraga ei kasutata

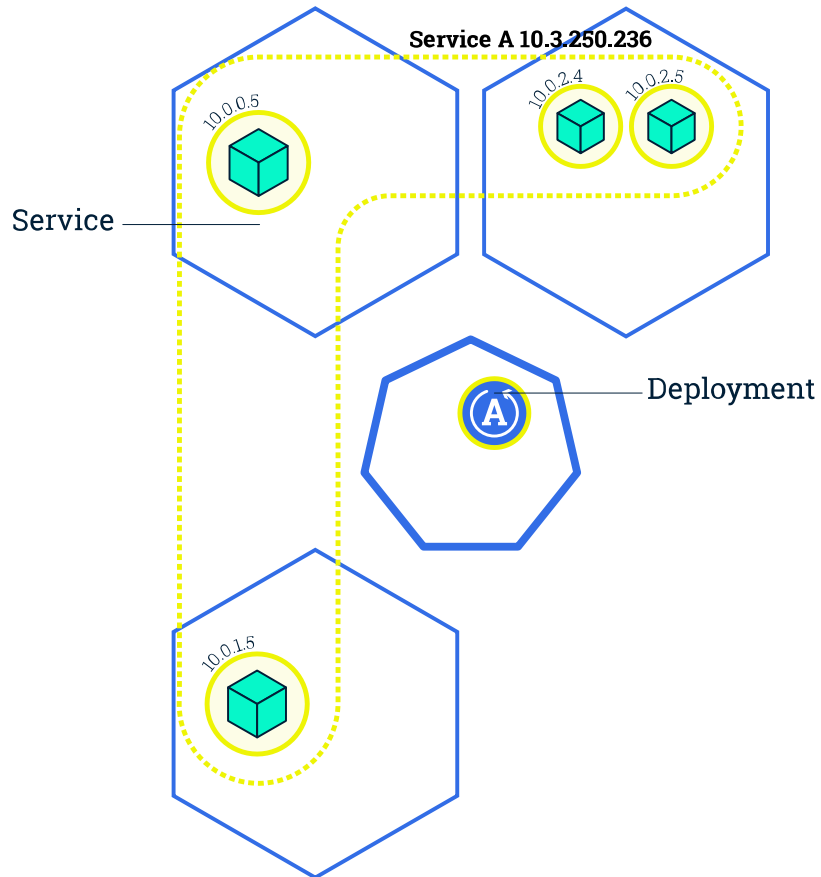


Ramped, Rolling update

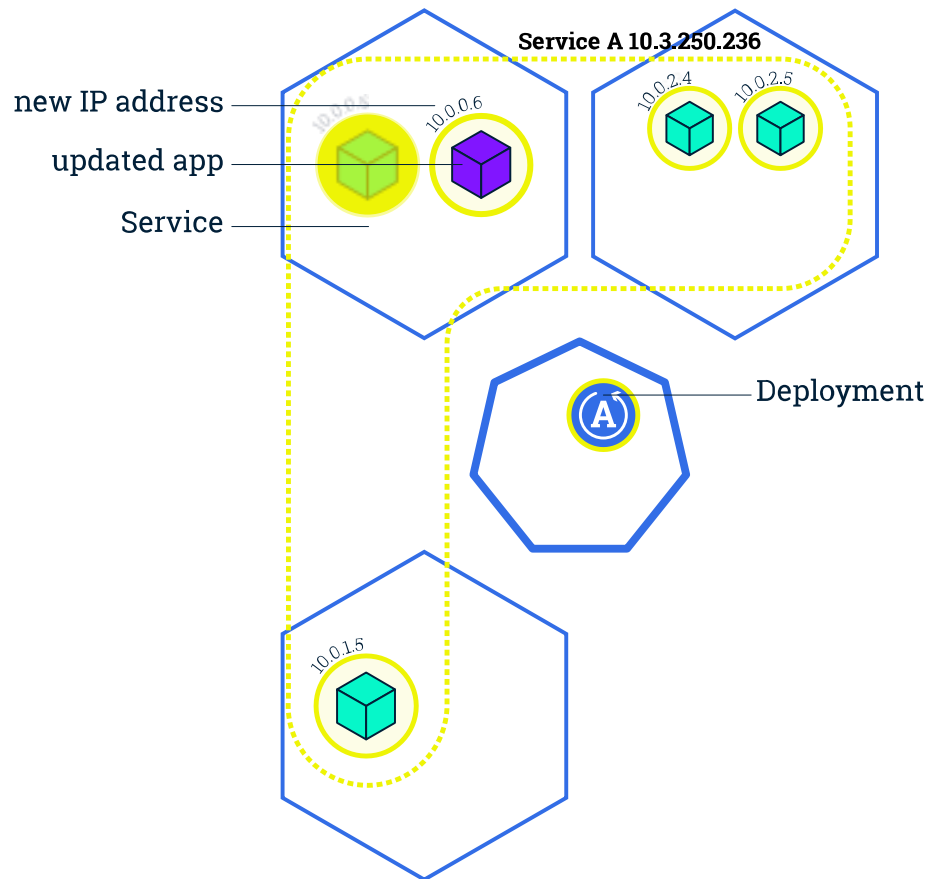
- Alustame uue versiooni poodide lisamisega, enne kui vanad eemaldatakse
 - Uued Podid peaksid olema liikluse vastuvõtmiseks valmis enne nende eemaldamist
 - Täiendamise ajal jätkavad vana versiooni POd liikluse teenindamist
- Vahetame 1+ Podi korraga
 - Oleneb vabade ressursside mahust



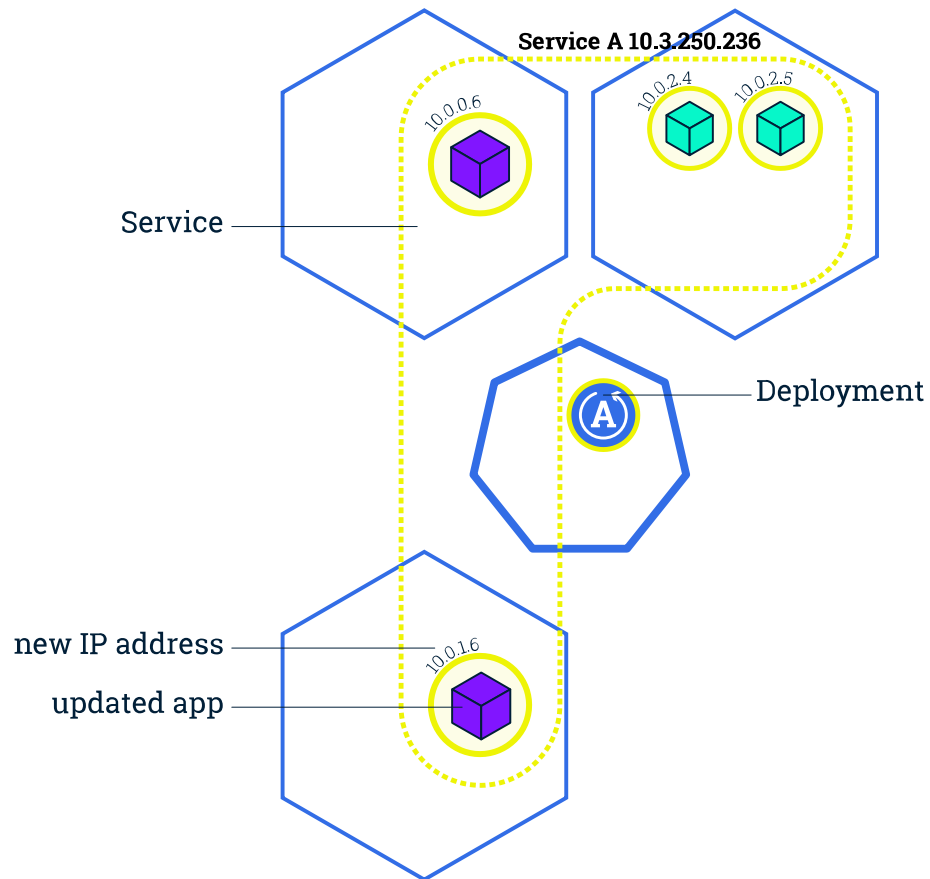
Jooksvad värskendused



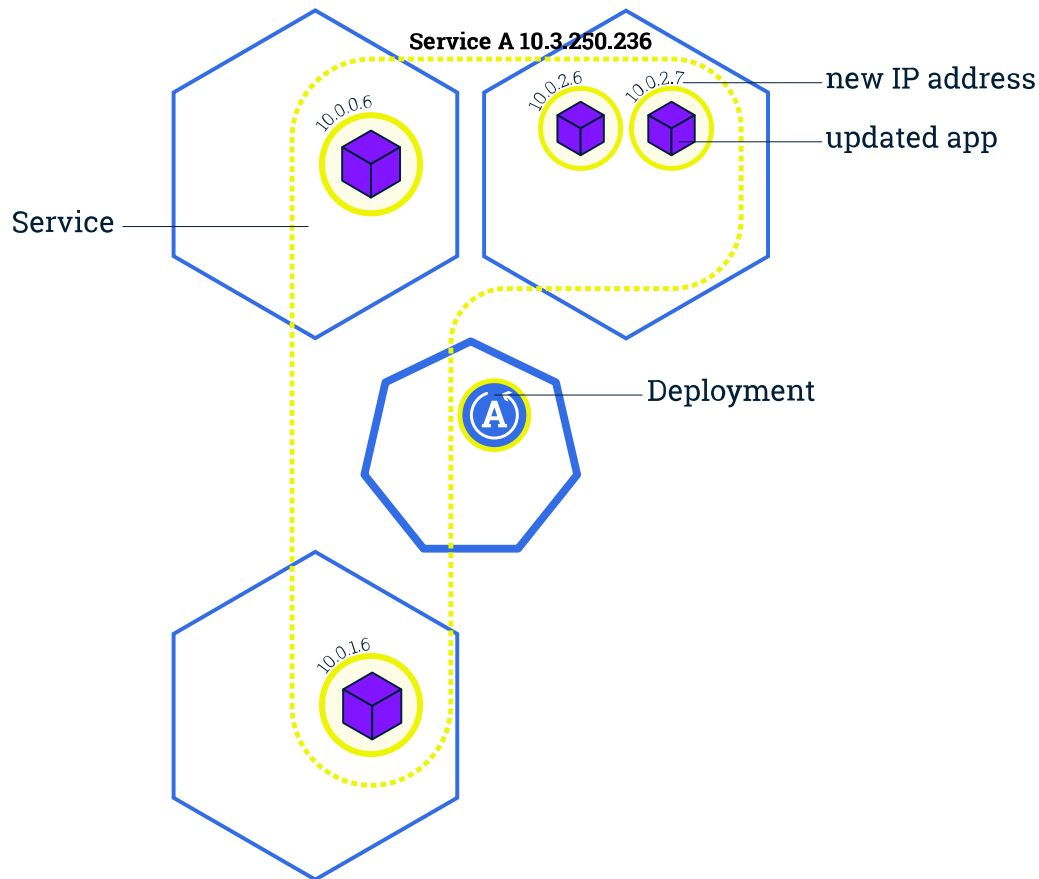
Jooksvad värskendused



Jooksvad värskendused

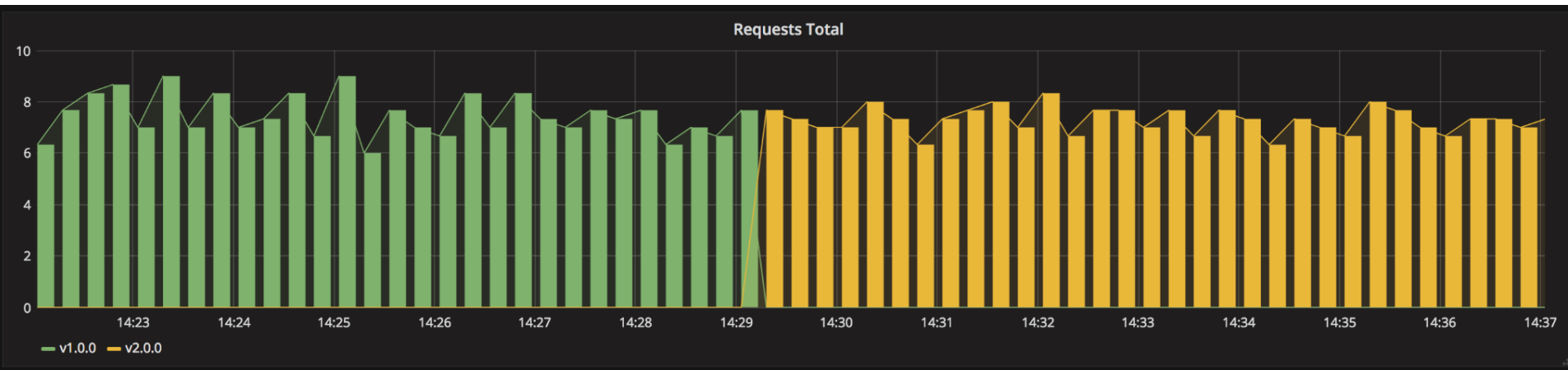


Jooksvad värskendused



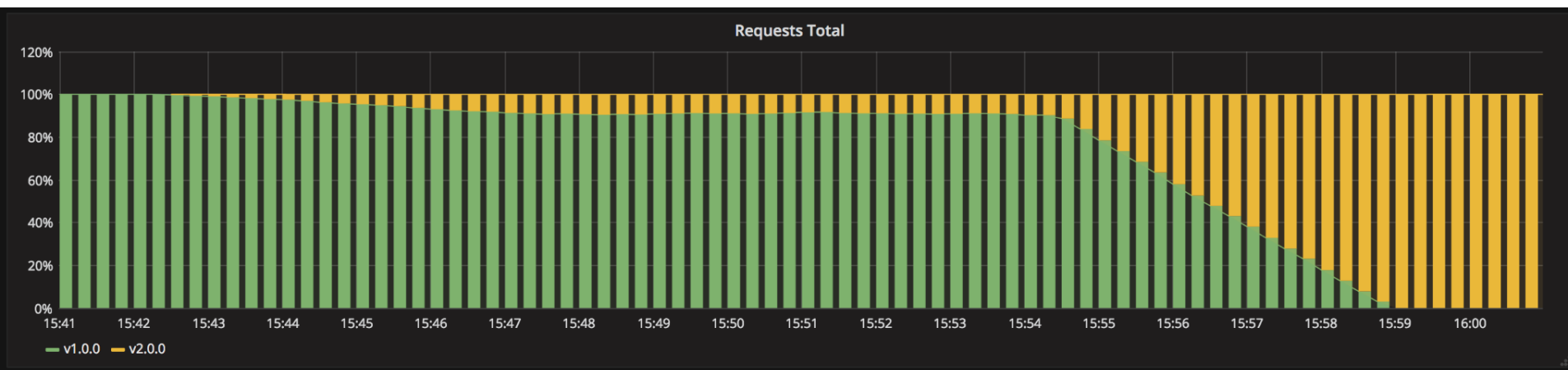
Blue/Green

- Vanalt (siniselt) uuele (rohelisele) versioonile üleminekuks juurutame kõik rohelise versiooni ressursid koos sinisega
- Kui kõik on valmis, lülitage liikluse suunamine üle rohelisele
- Jälgime tõrgete või probleemide tekkimist
- Kui kõik näib teatud aja jooksul korras olevat, eemaldame Sinise versiooni ressursid
 - Probleemide ilmnemisel saab lülituda kohe tagasi rohelisele
- Võib oluliselt vähendada tagasipööramise aega, aga läheb kalliks



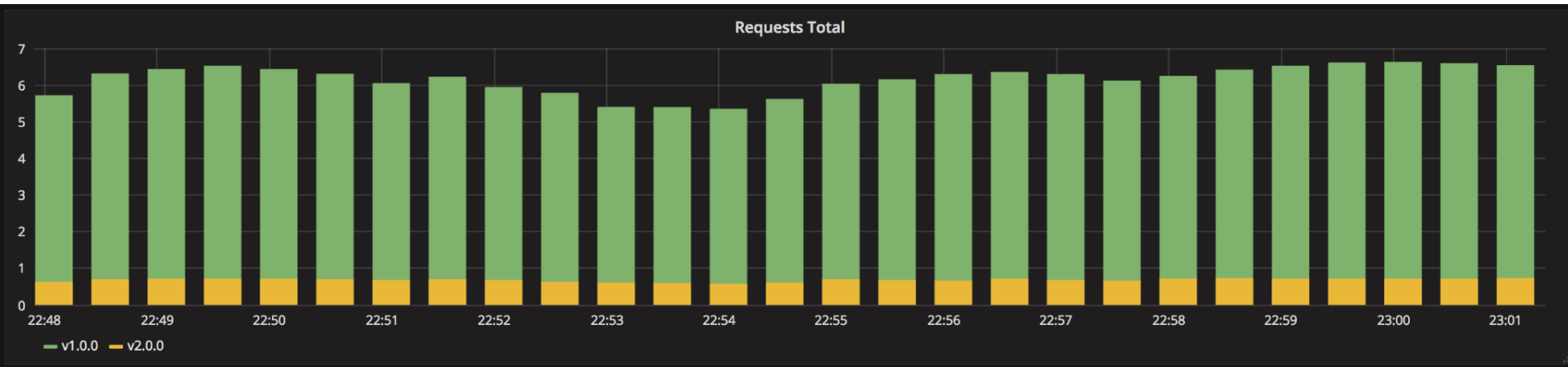
Canary

- Aeglasem ja hilinevad üleminek/üleminek uuele versioonile
- Juurutame paar uue versiooni Pod'i ja jälgime, et nendega poleks probleeme
- Pärast kinnitamist asendame kiiresti uuele versiooni Pod'idega
- Kasutajad ei pruugi märgata, et nad on vähesed, kes uut versiooni testivad
- Osa kasutajate liiklusest suunatakse uude versiooni



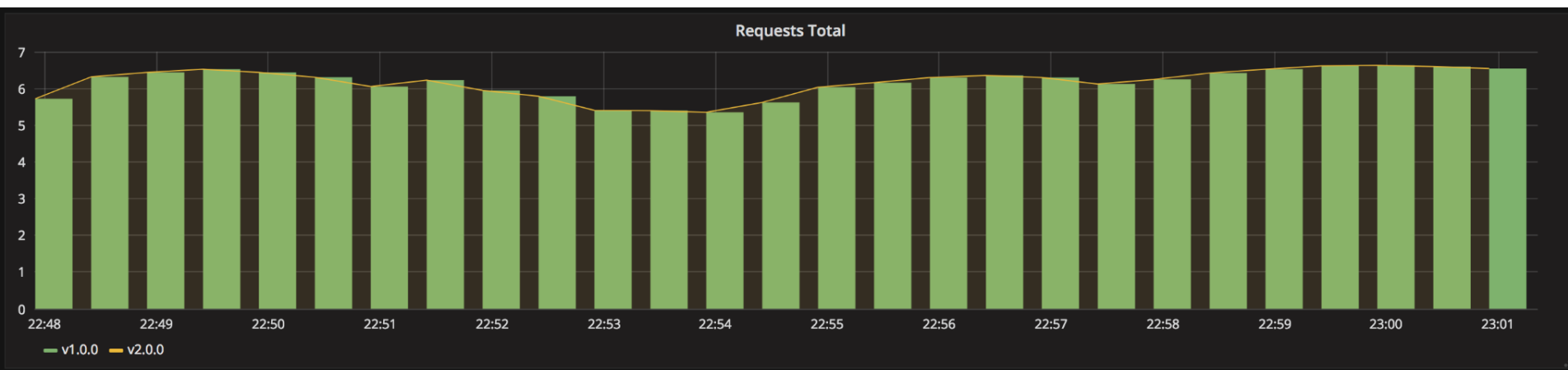
A/B testing

- Avaldame sama tarkvara kaks+ erinevat alternatiivset versiooni
 - Testime, milline neist mõne KPI osas paremini toimib
 - Kasutajate aktiivsus (engagement), veamäärad, isegi kasutajate tagasiside. Või muud KPI-d.
- Kasutajate liikluse jagamiseks kasutame Kubernetese teenuseid (services)
- Jälgime jõudlusnäitajaid ja kasutajate tagasisidet, teatatud probleeme
- Kui on selge, milline versioon "võidab", loobume teisest versioonist



Shadow

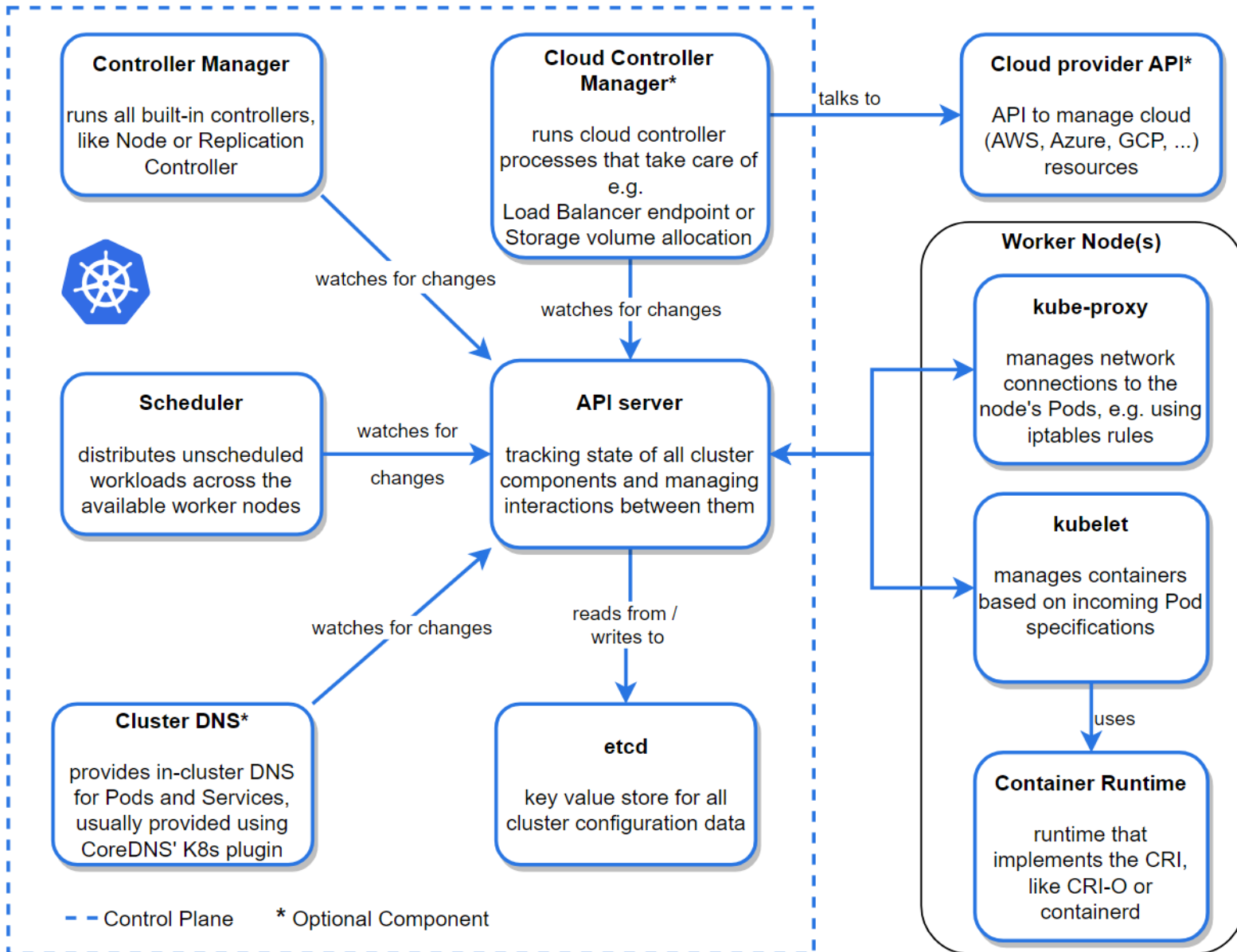
- Seame korraga üles kaks versiooni tarkvarast
- Saada sama kasutusega liiklus korraga nii vanale kui ka uuele versioonile
 - Liikluse, päringute dubleerimine
- Kasutame testimiseks reaalselt kasutajaliiklust, ilma kasutajakogemust tegelikult mõjutamata
 - Kasutajad saavad endiselt vastuseid tarkvara vanemast versioonist
- Nõuab kaks korda rohkem ressursse
 - Valimit saab kasutada ainult kasutajapäringute alamhulga kasutamiseks



Kubernetes ehitusblokid

- K8s klaster koosneb tööliste sõlmede ja juhtimisteenuste komplektist
- Kubernetes klatri juhtimiseks kasutatakse haldusteenuseid
 - Tihti ei jookse ühes keskses kontroller serveris, vaid hajutatult
- Juhttasandi komponendid:
 - REST API server, Scheduler, Etcd database, Controller manager
- Töötajate sõlmedes jooksevad:
 - Kubelet, k-proxy, container runtime
 - Vastutab konteinerite käitamise eest

Kubernetes Architecture



KUBERNETESE

JUHTTASANDI TEENUSED

API server

- Teeb Kubernetes API kättesaadavaks väljaspoolt
- Kubernetes front-end teenus
- Toetab mitme Kubernetes-API serveri paralleelset käivitamist
 - API päringud jagatakse nende vahel

etcd

- Järjepidev ja kõrge kättesaadavusega Võti-Väärtus tüüpi hajus andmebaas
- Kasutatakse koordineerimiseks ja hajutatud konfiguratsiooni salvestamiseks
- Salvestab k8s klatri soovitud olukorra ja hetkeolukorra seisud
 - `kubectl get` käsud küsivad andmeid etcd andmebaasist
 - `kubectl commands` käsud muudavad andmeid etcd andmebaasis
- "Distributed `/etc` directory"
 - `/etc` kausta kasutatakse Linuxis konfiguratsioonide salvestamiseks
- Liidri-põhine hajutatud andmebaas
 - Kuid sõlmes jooksvad protsessid ei pea teadma, kes on juht
 - Kõik päringud, mis tahes etcd protsessile (mis nõuavad konsensust) edastatakse etcd liidrile (juhile)

Kontrollerite haldaja - Controller Manager

- Juhttasandi komponent, mis käivitab kontrolleri protsesse
 - Iga alamkontroller on eraldi protsess
 - Saab töötada k8s klatri "mis tahes" sõlmes hajusalt
- Alam-Kontrollerid kontrollivad klatri ja selle ressursside hetkeseisu ja soovitud oleku erinevust.
 - Tagavad, et klatri ressursid ja teenused liiguvad soovitud hetkeseisu suunas
- K8s kontrollerite tüübid:
 - **Node controller - Sõlmekontroller**: vastutab sõlmede ebaõnnestumise, eemaldamise või lisamise korral märkamise ja reageerimise eest
 - **Job controller - Töökontroller**: jälgib tööobjekte (Jobs), mis esindavad ühekordseid ülesandeid, ja loob Podid, et neid ülesandeid lõpule viia
 - **Service Account & Token controllers** - Kontode ja pääsete kontrollerid: haldavad API kontosid ja juurdepääsu lubasid

Planeerija - Scheduler

- Kontroll-taseme teenus
- Kuulab äsja loodud ja allokeerimata Pod'e
 - Valib, millises sõlmes neid käivitada
- Planeerimisel võtab:
 - individuaalsed ja kollektiivsed ressursside vajadused
 - riistvara/tarkvara/poliitika piirangud
 - Afiinsus ja anti-afiinsus
 - nt disktype=ssd.
 - Andmebaas ja cache Podid võiksid olla üksteise lähedal.
 - andmete lokaalsus
 - tähtajad

KUBERNETESE

SÕLME TEENUSED

Kubelet

- Peamine "sõlme agent"
 - Agent, mis töötab igas klasteri töötaja sõlmes
- PodSpec on YAML- või JSON-objekt, mis kirjeldab Pod'i
- Sõlme Kubelet vaatab andmebaasist temale määratud PodSpec-ide komplekti
 - Jälgib pidevalt Pod'ide **soovitud** ja **tegeliku olukorra** erinevust
 - Tagab, et sõlmele määratud PodSpec'ides kirjeldatud konteinerid töötavad ja on hea tervise juures – Vastavad soovitud olukorrale

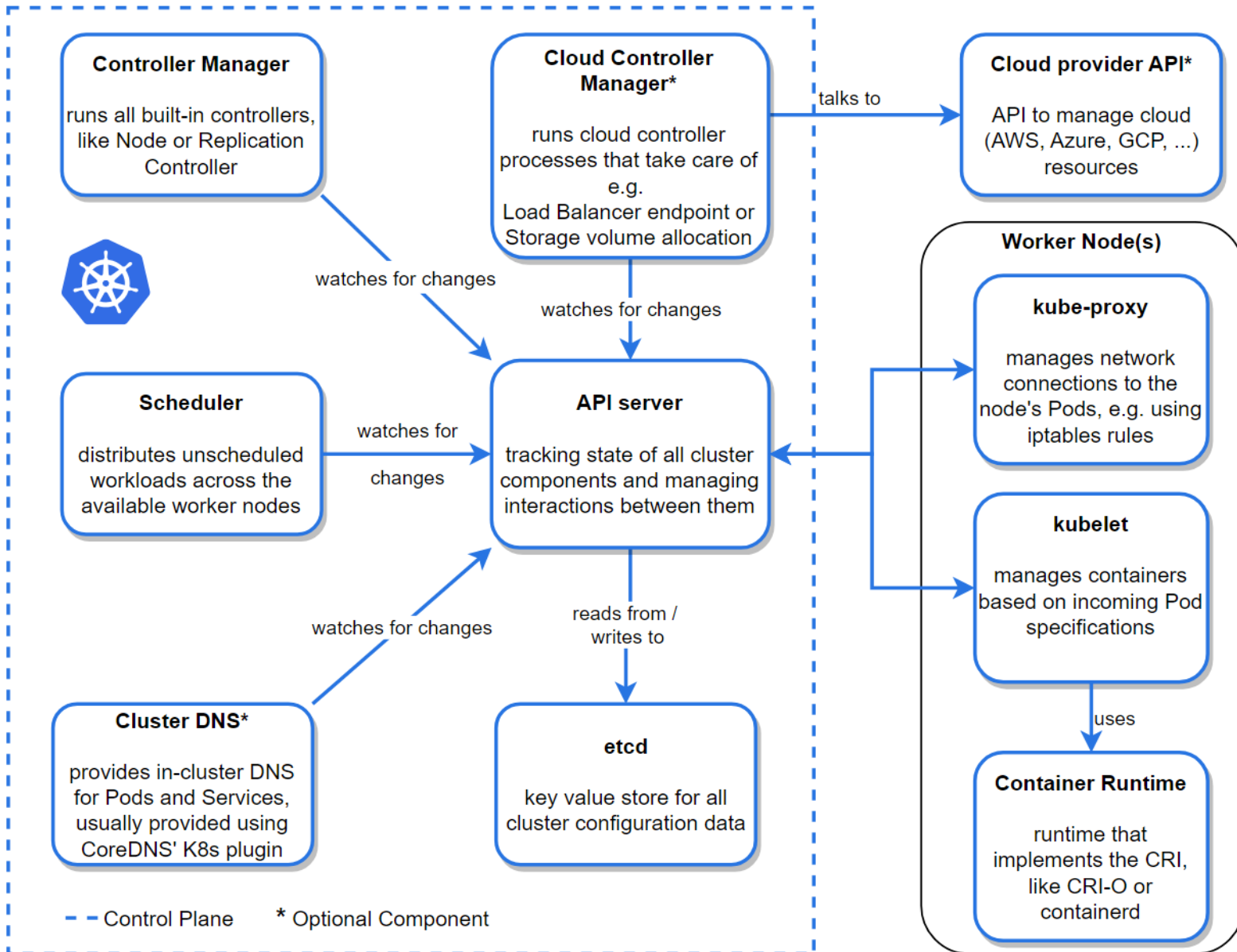
kube-proxy

- Võrgu proksi (proxy) mis rakendab k8s teenuse (service) konfiguratsioone
- Üles seatud igas töötaja sõlmes
- Haldab sõlmede võrgureegleid, et võimaldada sidet Podide ja teiste sõlmede ning välismaailma vahel
- Kasutab kas operatsioonisüsteemide pakettide filtreerimise rakendusi (nt iptables) või edastab ise liiklust

Konteinerite haldustarkvara

- Tarkvara, mis käivitab töötajate sõlmes konteinereid
- Kubernetes toetab erinevaid
 - containerD
 - CRI-O (to support Open Container Initiative containers)
 - Docker Engine
 - Mirantis Container Runtime (varasemalt Docker Enterprise)
 - ... mõni muu, mis implementeerib Kubernetes Container Runtime Interface (CRI) liidest

Kubernetes Architecture



Kubernetes omadused

- Teenuse leidmine (discovery) ja koormuse balansseerimine
- Salvestusruumi orkestreerimine
- Uute versioonide automatiseeritud väljalaskmine ja tagasivõtmine
- Automaatne ressurside kasutuse optimeerimine
- Ise tervenev süsteem
- Saladuste ja konfiguratsioonihaldus

Konteinerite orkestreerimine

- Konteinerite orkestreerimisplatvormid pakuvad lisafunktsioone, näiteks:
 - Klastrite haldamine ja replitseeritud klastriülesed teenused
 - Docker Swarm, Kubernetes
 - Teenuse leidmine ja koormuse tasakaalustamine
 - Docker Swarm, Kubernetes
 - Salvestusruumi orkestreerimine
 - Kubernetes
 - Automatiseeritud levitamine ja versioonide tagasi pööramine
 - Kubernetes
 - Enese paranemine
 - Kubernetes, Docker Swarm (somewhat)
 - Saladuste ja konfiguratsioonihaldus
 - Kubernetes
 - Automaatne skaleerimine
 - Kubernetes

Kokkuvõte

- Mikroteenused lihtsustavad hajussüsteemide loomist, individuaalsete komponentde uuendamist ning skaleerimist
- Konteinerid on olnud ühed peamised Mikroteenuste edukuse võimaldajad
- Docker Swarm võimaldab luua suuremaid Docker klastreid
- Kubernetes lihtsustab konteinerite ja mikroteenuste haldust
 - Suurim kogukond konteinerite orkestreerimisvahendite vallas
 - Võib töötada otse riistvara peal või pilves: OpenStack, Google Cloud, Azure, AWS, ...
- Mikroteenused teevad süsteemide arhitektuuri keerulisemaks
 - Tükke on lihtsam luua
 - Kogu süsteemi võib olla keerulisem hõlmata ning hallata
- Mikroteenused võivad olla eba efektiivsemad ressursi kasutuse osas
 - Nt. Lokaalse võrgu kasutamine teenuste vahel vs Jagatud mälu alad