Kubernetes

Ondřej Smola

21.04.2022 | CTC

Z monolitu k mikroslužbám

- Snaha rozdělit velké aplikace na více nezávislých komponent komunikujících pomocí API
- Problémy monolitu
  - Nasazení
  - Nároky na HW
  - Škálování
  - Rychlost vývoje
  - Aktualizace
    - Měsíční release

Motivace

Mikroslužby

- Rozložení monolitu na menší komponenty
- Výhody
  - Snížení komplexity uvnitř
  - Možnost využití specifických technologií/jazyků
  - Snížení HW nároků
  - Jednodušší škálování a aktualizace
- Nevýhody
  - Zvýšení vnější komplexity – definice API
  - Nutnost automatizace
  - Latence
  - **Náročnější na celkovou operativu**
    - Rolling update
    - Blue-green deployment

Kotejnarizace a orchestrace

- Kontejnarizace
  - Zabalení aplikace do obrazu
  - Spuštění kontejneru v běhovém prostředí
  - Pracuje v rámci jednoho stroje
- Orchestrace
  - Spravuje kontejnery v rámci více strojů (clusteru)
    - Spuštění
    - Škálování
    - **Plánování**
  - Poskytuje podpůrnou infrastrukturu
    - Směrování
    - Datová uložiště
    - Monitoring

Požadavky na orchestrátor

- Odolnost vůči výpadku
- Škálovatelnost
- Optimální využití výpočetních prostředků
- Zajištění komunikace
  - mezi kontejnery v rámci strojů
  - z vnějšku
- Automatizovaná správa kontejnerů

Orchestrátory

- Kubernetes
- Apache Mesos
- Amazon Elastic Container Service
- Azure Service Fabric
- Docker Swarm (deprecated)
- Nomad

Cloud native computing foundation (CNCF)

Cloud native

- Umožňující využití výhod unikátních vlastností cloudu
  - Samo-obslužný
  - Řízený API
  - Elastický/Škálovatelný

- Koncepty
  - DevOps
  - CI/CD
  - Mikroslužby
  - Kontejnery

Hlavní projekty

Container Runtime

Coordination & Service Discovery

Service Proxy

Coordination & Service Discovery

Logging

Container Registry

Application Definition & Image Build

Tracing

Scheduling & Orchestration

Service Mesh

Security & Compliance

Monitoring

Cloud Native Storage

Security & Compliance

Database

Database

Inkubované projekty

Continuous Integration & Delivery

Application Definition & Image Build

Chaos Engineering

Cloud Native Network

Streaming & Messaging

Cloud Native Network

Service Proxy

Monitoring

Container Runtime

Scheduling & Orchestration

Framework

Container Registry

API Gateway

Security & Compliance

Continuous Integration & Delivery

Continuous Integration & Delivery

Remote Procedure Call

Installable Platform

Installable Platform

Automation & Configuration

Vztah ke Kubernetes

- Neutrální půda pro vývoj
- Propagace
- Kurzy
- Konference a meetupy
- Pracovní skupiny (WG)

Kubernetes

Definice

- *"Kubernetes je open-source systém pro automatizaci nasazení, škálování a správu kontejnerizovaných aplikací"*

- Zkráceně **k8s**
- Inspirován systémem Borg (Google) využívaného pro provoz infrastruktury Google více než deset let
- Open source nástroj (Apache 2.0) implementovaný v jazyce Go
- Od roku 2015 převeden pod Cloud Native Computing Foundation (CNCF)

Verze

- Aktuální verze je 1.23 (04/2022)
- Nová minor verze vychází cca každých 15 týdnů
- Každá minor verze je podporová 1 rok, patch verze vychází dle potřeby

- Přehled verzí na https://kubernetes.io/releases/

- Z výše uvedeného plyne, že je potřeba provést aktualizaci clusteru minimálně 1 za rok, ideálně 2-3x

Borg

- Paper: https://research.google/pubs/pub43438/

- Orchestrátor používaný např. pro
  - Gmail
  - Drive
  - Mapy
- Příklad konceptů převzaných z Borgu
  - Pody
  - API sever
  - IP per Pod
  - Služby
  - Značky

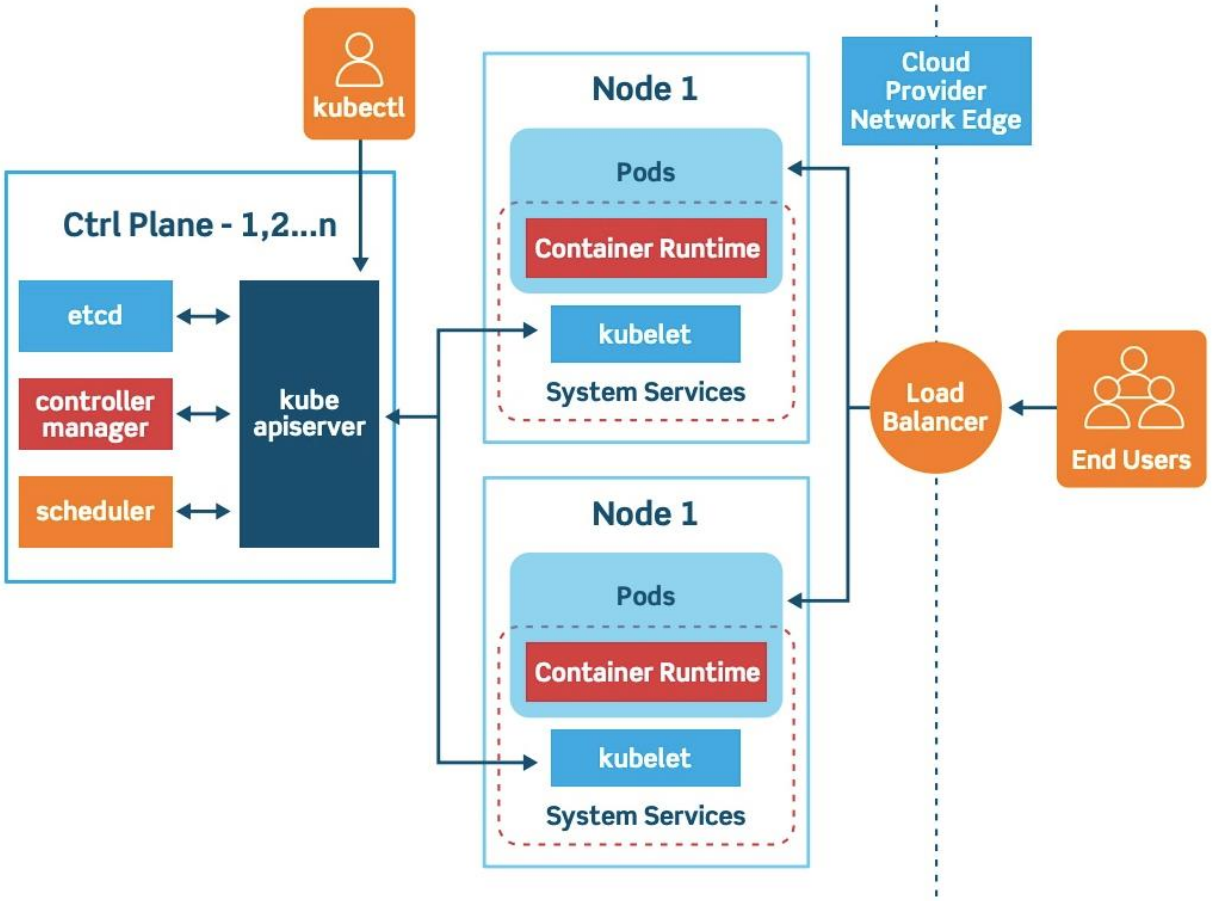
Vlastnosti kubernetes

- Automatické plánování kontejnerů dle konfigurace k zajištění co nejefektivnějšího využití výpočetní kapacity a zajištění vysoké dostupnosti (bin packing)
- Automatizovaná obnova
  - nahrazení selhaných kontejnerů
  - změna v plánování
  - rekonfigurace plánování
- Horizontální škálování
  - Dle zatížení CPU, počtu požadavků …
- Service discovery a load balancing
  - Konfigurace směrovacích tabulek
  - DNS

Vlastnosti Kubernetes 2

- Rollout a rollback
  - Automatizovaná aktualizace a návrat k předchozí konfiguraci během výpadku
- Správa utajených hodnot (Secret) a konfigurací
  - Rekonfigurace za běhu
  - Předávání pomocí souborů, proměnných prostředí
- Orchestrace úložiště
  - Připojování sdílených síťových blokových/souborových systému
  - Přímá integrace s cloudovými poskytovately
- Dávkové zpracování, cron job

Architektura



Master

- Zajišťuje běhové prostředí pro řídící prvky, které udržují stav Kubernetes clusteru
- Mozek celého clusteru, vždy jeden master zvolen jako leader
- Musí být aktivní alespoň většina masterů, jinak se přepíná cluster do módu pouze pro čtení
- V případě pádu všech masterů dochází k úplné nedostupnosti clusteru
  - Ostatní node i kontejnery na nich běží, ale již neprovádí žádné další akce
- Všechna data jsou ukládána do etcd clusteru
  - Interní (stacked mód)
  - Externí

Komponenty na masteru

- Controller manager
- Scheduler
- Etcd
- Api server
- Node agent
- Container runtime
- Proxy

Api server

- Probíhá přes něj veškerá API komunikace
  - Příkazová řádku (kubectl)
  - REST
  - Agenti

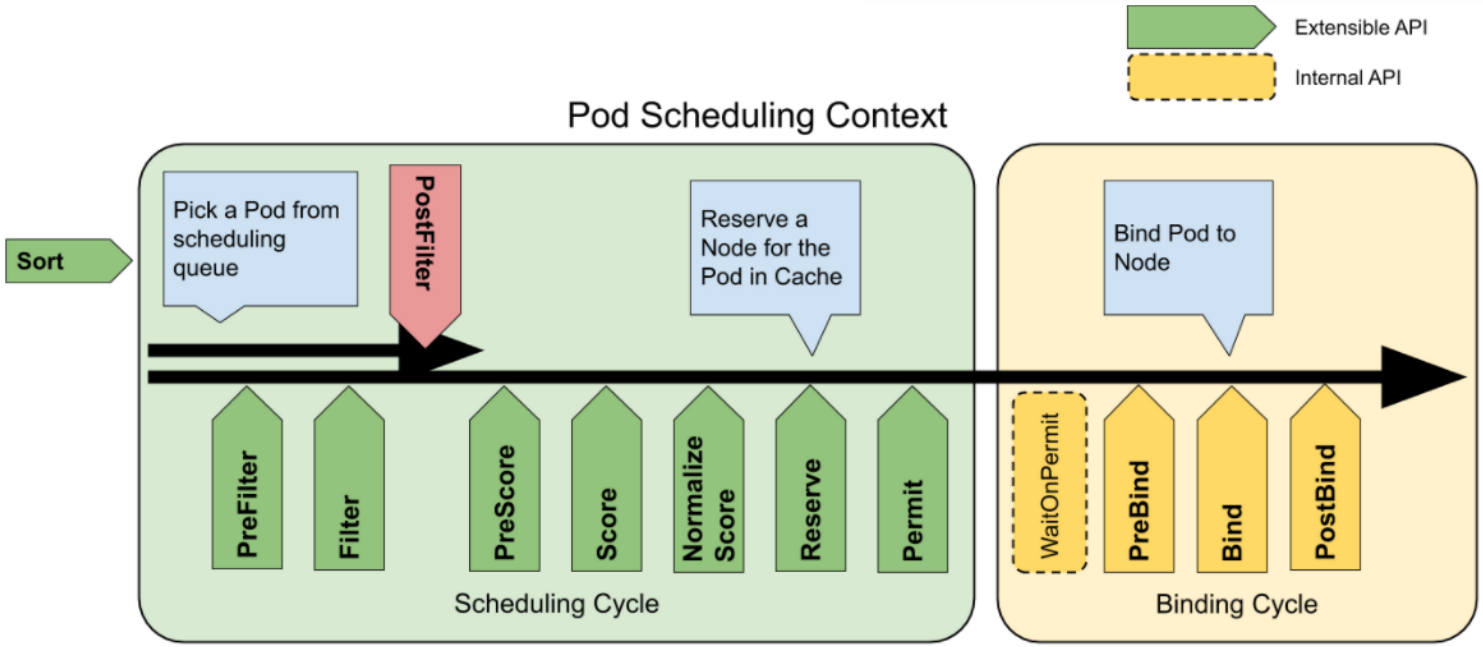
- Autentizace a autorizace
- Ukládá a čte data z Etcd (jako jediný)
  - Protobuf
- Horizontálně škálovatelný

Plánovač

- Cílem je přiřadit skupiny kontejnerů (pod) k nodům (agent) na základě aktuálního stavu clusteru
- Cílem plánovací logiky je neustále konvergovat k ideálnímu stavu
  - Není závislá na detekci změn ale na rozdílu mezi aktuálním a požadovaným stavem
- Během plánování bere v úvahu
  - Label (disk==ssd)
  - Affinity a anti-affinity
  - Taints a tolerations
  - Lokalitu dat
  - Kvalitu služeb (QoS)

Algoritmus plánovače

- **QueueSort**: Sort the pods in the queue
- **PreFilter**: Check the preconditions of the pods for scheduling cycle
- **Filter**: Filter the nodes that are not suitable for the pod
- **PostFilter**: Run if there are no feasible nodes found for the pod
- **PreScore**: Run prescoring tasks to generate shareable state for scoring plugins
- **Score**: Rank the filtered nodes by calling each scoring plugins
- **NormalizeScore**: Combine the scores and compute a final ranking of the nodes
- **Reserve**: Choose the node as reserved before the binding cycle
- **Permit**: Approve or deny the scheduling cycle result
- **PreBind**: Perform any prerequisite work, such as provisioning a network volume
- **Bind**: Assign the pods to the nodes in Kubernetes API
- **PostBind**: Inform the result of the binding cycle



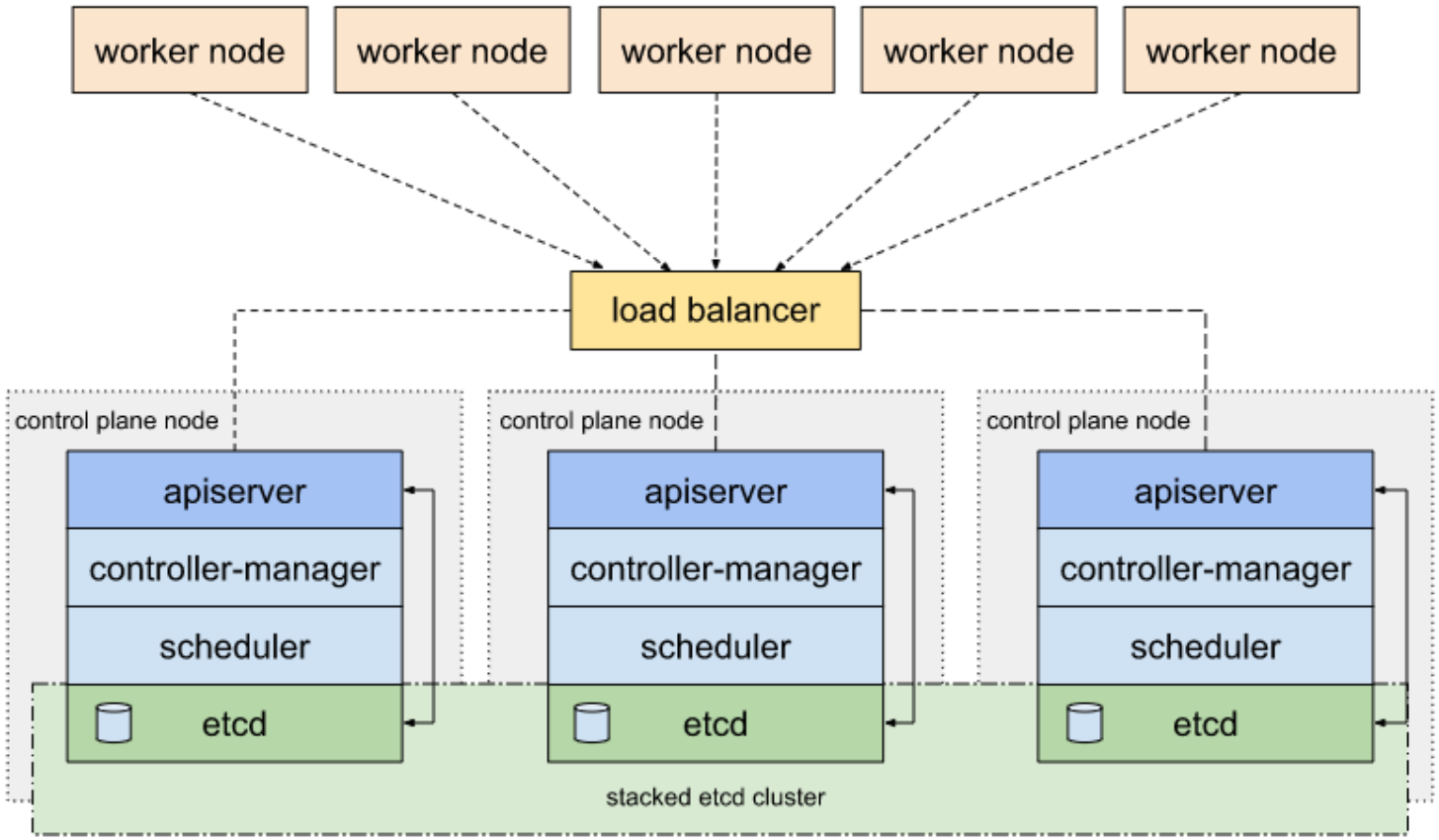
Správce kontrolérů

- Kontrolér
  - Sledovací cyklus pomocí API (etcd-watch)
  - Porovnáná aktuální stav s požadovaným
  - V případě odchýlení stavu provádí kompenzační akce

- Kube-controller-manager
  - Nody, služby, API klíče

- Cloud-controller-manager
  - Interakce s konkrétní infrastrukturou cloudového poskytovatele

Etcd

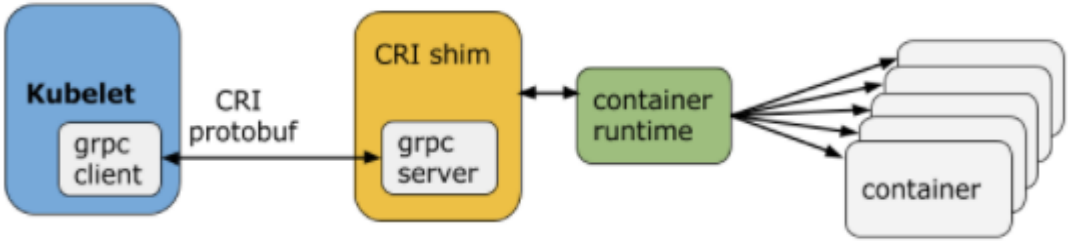

worker node
worker node
worker node
worker node
worker node
load balancer
control plane node
control plane node
control plane node
apiserver
controller-manager
scheduler
etcd
apiserver
controller-manager
scheduler
etcd
apiserver
controller-manager
scheduler
etcd
stacked etcd cluster

Node

- Poskytuje prostředí pro aplikace zapouzdřené jako skupiny kontejnerů == pody
- Které pody poběží na nodu rozhoduje plánovač
- Komponenty
  - Běhové prostředí kontejnerů (containerd, cri-o)
  - Agent (kubelet)
  - Proxy (kube-proxy)
  - Volitelná rozšíření
    - Monitoring (cadvisor)
    - Připojení síťových disků (CSI agent)
    - Logování (promtail, fluentD)

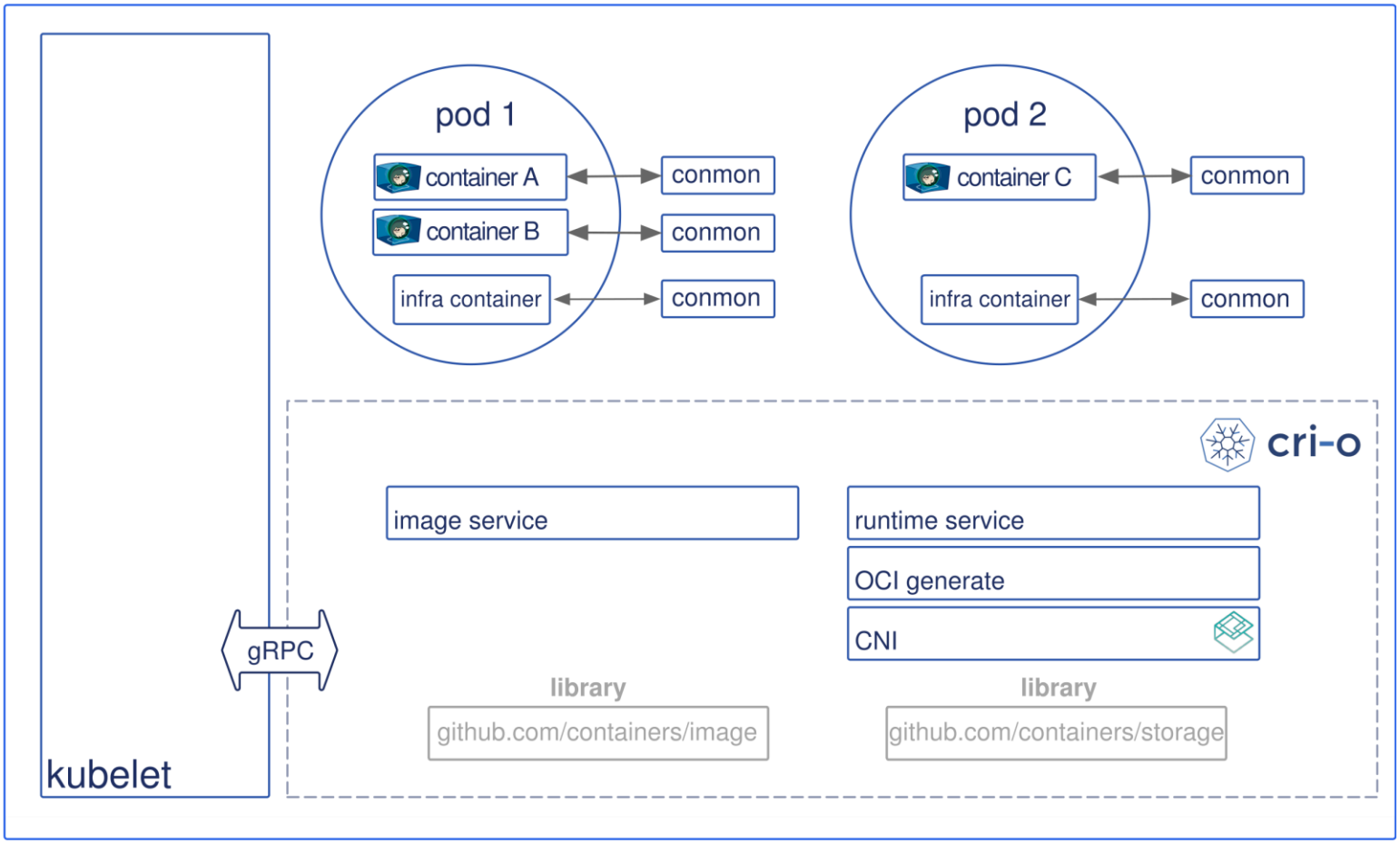
Kubelet

- Komunikuje s mastery
- Konfiguruje a spravuje lokální kontejnery pomocí CRI
- Zajišťuje kontrolu spuštěných podů (healt-check)



- Pause kontejner – viz další slide (https://www.ianlewis.org/en/almighty-pause-container)

Cri-o


pod 1
container A
container B
infra container
conmon
conmon
conmon
pod 2
container C
infra container
conmon
conmon
cri-o
image service
runtime service
OCI generate
CNI
gRPC
library
github.com/containers/image
library
github.com/containers/storage
kubelet

Kube-proxy

- Konfiguruje síťová pravidla (iptables)
- Směrování a load-balancing (round-robin)
  - Protokoly UDP, TCP a SCTP
- Implemetuje síťová pravidla definovaná pomocí Service

```
ubuntu@node-10:~$ sudo iptables -L KUBE-SERVICES  -t nat
Chain KUBE-SERVICES (2 references)
target     prot opt source               destination
KUBE-SVC-JUR5XFG4VGG55PUT  tcp  --  anywhere             10.20.32.174         /* ng-data/harbor-redis-master:tcp-redis cluster IP */ tcp dpt:6379
KUBE-SVC-TQJOHOK73DR5YTJH  tcp  --  anywhere             10.20.20.61          /* ng-core/dashboard:http cluster IP */ tcp dpt:http
KUBE-SVC-DMQT7TLM3UQHHGZG  tcp  --  anywhere             10.20.132.200        /* ng-data/s3-browser:default cluster IP */ tcp dpt:9000
KUBE-SVC-PFPKFILTA2SRHNTY  tcp  --  anywhere             10.20.129.149        /* ng-mon/thanos-compactor:http cluster IP */ tcp dpt:9090
KUBE-SVC-HS5GLWXG2NY52BK2  tcp  --  anywhere             10.20.68.147         /* ng-data/harbor-portal:http cluster IP */ tcp dpt:http
KUBE-SVC-HNJ23SQTQPWOSVSX  tcp  --  anywhere             10.20.9.151          /* ng-data/rook-ceph-mon-b:tcp-msgr1 cluster IP */ tcp dpt:6789
KUBE-SVC-ZZIZWZUFBX7OROJN  tcp  --  anywhere             10.20.129.157        /* ng-mon/prometheus-stack-kube-prom-prometheus:http-web cluster IP */ tcp dpt:9090
KUBE-SVC-5ULUYRJXD4NEPNF3  tcp  --  anywhere             10.20.47.251         /* ng-data/csi-cephfsplugin-metrics:csi-http-metrics cluster IP */ tcp dpt:http-alt
KUBE-SVC-JD5MR3NA4I4DYORP  tcp  --  anywhere             10.20.0.10           /* kube-system/kube-dns:metrics cluster IP */ tcp dpt:9153
KUBE-SVC-AAWX7D473I5RLGUP  tcp  --  anywhere             10.20.249.84         /* ng-data/rook-ceph-mon-e:tcp-msgr1 cluster IP */ tcp dpt:6789
KUBE-SVC-KJXJJUWH2WBSDFHR  tcp  --  anywhere             10.20.110.220        /* ng-net/cert-manager-webhook:https cluster IP */ tcp dpt:https
KUBE-SVC-UNQUVM6RTYSKI27N  tcp  --  anywhere             10.20.224.200        /* ng-data/harbor-jobservice:http cluster IP */ tcp dpt:http
KUBE-SVC-KSNAN6HM2F36IRDQ  tcp  --  anywhere             10.20.1.66           /* ng-mon/thanos-query:grpc cluster IP */ tcp dpt:10901
KUBE-SVC-7VIPZJT3R2NATTRJ  tcp  --  anywhere             10.20.139.85         /* ng-data/harbor:http cluster IP */ tcp dpt:http
KUBE-SVC-XXUPMH5RDSDIBXYQ  tcp  --  anywhere             10.20.9.151          /* ng-data/rook-ceph-mon-b:tcp-msgr2 cluster IP */ tcp dpt:3300
KUBE-SVC-BAMAEYO7BYRLA7JM  tcp  --  anywhere             10.20.235.77         /* ng-data/harbor-registry:registry cluster IP */ tcp dpt:5000
KUBE-SVC-TCOU7JCQXEZGVUNU  udp  --  anywhere             10.20.0.10           /* kube-system/kube-dns:dns cluster IP */ udp dpt:domain
KUBE-SVC-OR4X5KQMFLLUIFAH  tcp  --  anywhere             10.20.143.188        /* ng-net/traefik-nodeport:https cluster IP */ tcp dpt:8443
KUBE-SVC-SGM4KNBXDJWBQKP3  tcp  --  anywhere             10.20.47.251         /* ng-data/csi-cephfsplugin-metrics:csi-grpc-metrics cluster IP */ tcp dpt:tproxy
KUBE-SVC-KMSVMKKLPNHVGD3A  tcp  --  anywhere             10.20.161.17         /* ng-mon/thanos-storegateway:http cluster IP */ tcp dpt:9090
```