

# **Felhő alapú hálózatok**

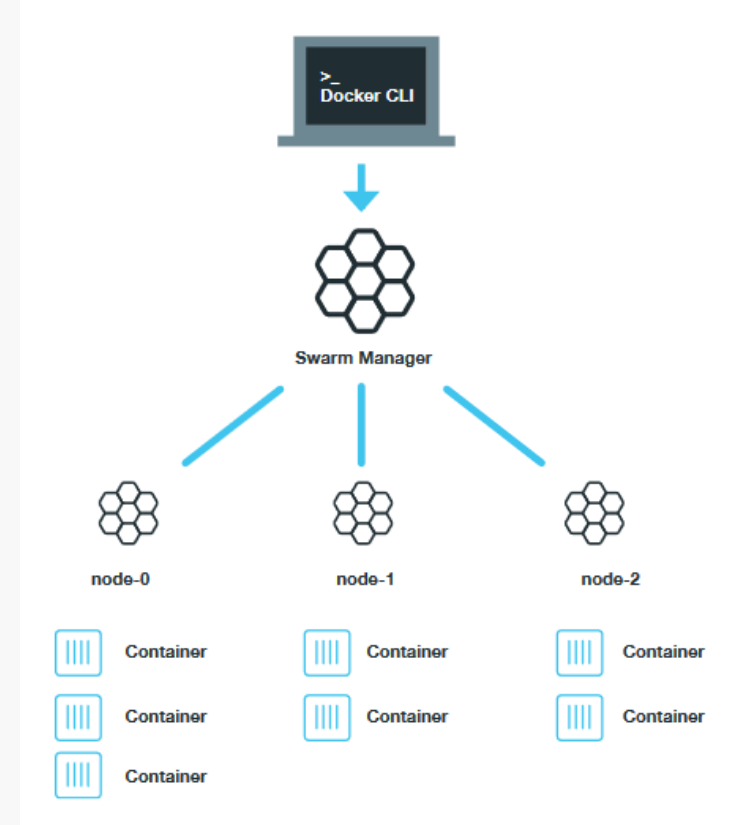
## **Konténerek orkesztrálása**

Simon Csaba

Budapesti Műszaki és Gazdaságtudományi Egyetem

# Motiváció – multi host

- » Docker konténerek docker parancsokkal kezelhetők
    - » Adott gazda gépen (on-host)
    - » Hálózati kapcsolatok nehézkesek
      - » docker0 bridge
    - » Különálló gépekre telepített docker konténerek kapcsolata?
      - » Mult-hosting
      - » Először külső megoldások (pl. serf - <https://www.serf.io/>)
      - » Később **Docker Swarm** – multi-hosting in Docker
- „It turns a pool of Docker hosts into a single, virtual Docker host”
- » Nem összetévesztendő a Docker Swarm Mode-al :( (v1.12 után)



# Motiváció - orkesztráció

- » Mi hiányzik egy teljes Docker rendszerhez?
  - » Orkesztráció
  - » Amit a felhők nyújtanak
  - » Cél: automatizált konténer telepítés és menedzsment multi-host környezetben (incl. skálázódás vezérlése)
- » Egyik megoldás: nyilvános felhőkben Docker
  - » Amazon Web Services, Google Cloud, Microsoft Azure
- » Másik megoldás: Docker + OpenStack
  - » OpenStack Magnum
- » Harmadik megoldás: Docker orkesztráció
  - » Apache Mesos (2010)
  - » Google Kubernetes (2014)
  - » Docker Swarm Mode (2016)

# Cloud Native Computing Foundation

- » Mikroszervíz ökoszisztéma
  - » Konténerek orkesztrálásával
  - » Google támogatja
  - » rkt-t javasolják a Docker helyett



[About](#)

[Projects](#)

[Certification](#)

[People](#)

[Community](#)

[Newsroom](#)

[Jobs](#)



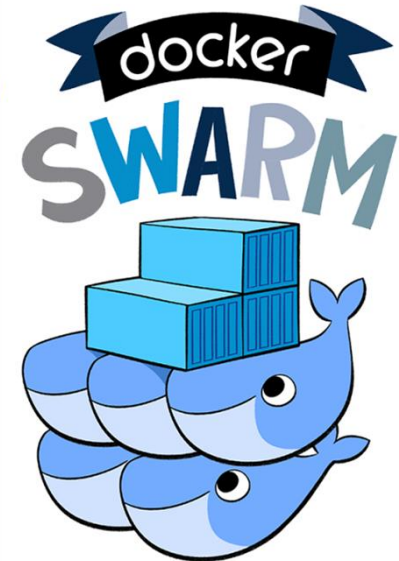
## Sustaining and Integrating Open Source Technologies

The Cloud Native Computing Foundation builds sustainable ecosystems and fosters a community around a constellation of high-quality projects that orchestrate containers as part of a microservices architecture.

---

# **DOCKER SWARM MODE**

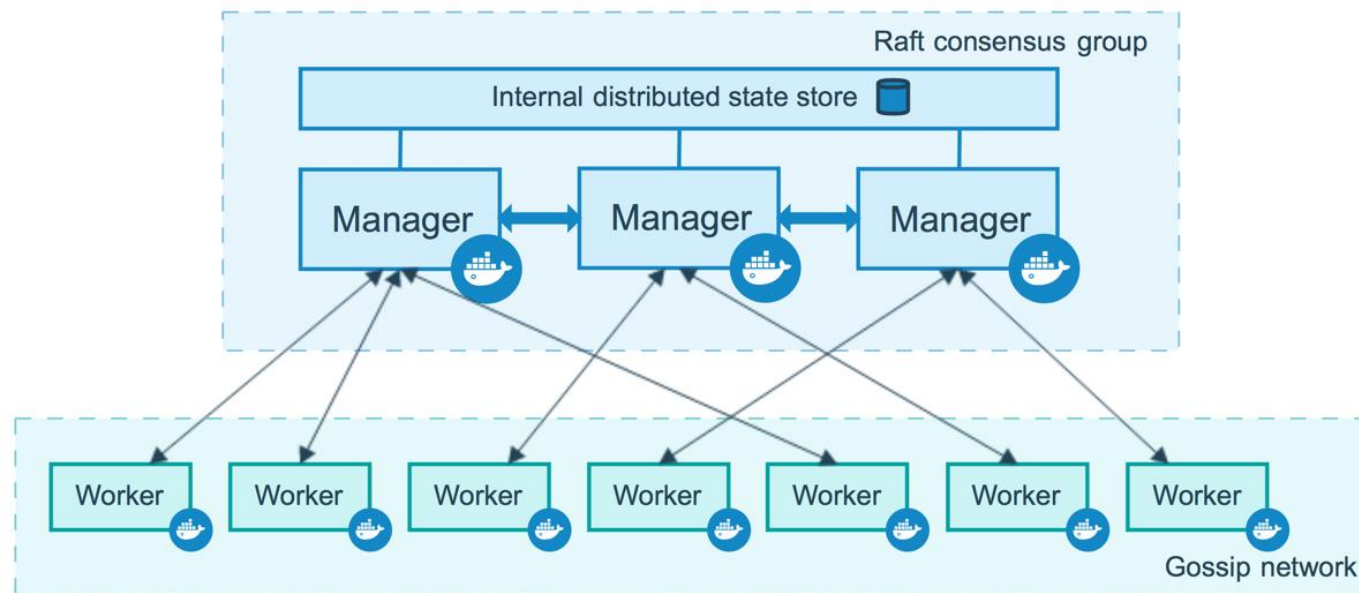
# Docker Swarm Mode



- » **Swarm mode** = Docker engine futtatási mód
  - » Amennyiben a Docker engine-k egy közös klaszterbe vannak szervezve
    - » Egy Docker engine = egy node
    - » Swarm = ez a fenti klaszter
      - » A cél: szolgáltatások (**services**) indítása ebben a klaszterben
  - » Egy fizikai gép elvileg több node-ot tartalmazhat
    - » „Komoly” üzleti környezetben egy Docker engine / fizikai gép
      - » Gyakorlatilag a Docker engine-t futtató host-ok vannak egy klaszterben
  - » Szolgáltatói modell = a felhasználók egy service-t érnek el
    - » A **service** replikált feladatot végez és meghatározza a működés kereteit (hálózat, erőforrás, replikációs szint és politika)
      - » Több node-on futtatott feladatokat (task) együttesen képvisel
    - » **Task** = feladat (= docker konténer), amit a service kezel
      - » Elemi erőforrás egység, egy node-on fut

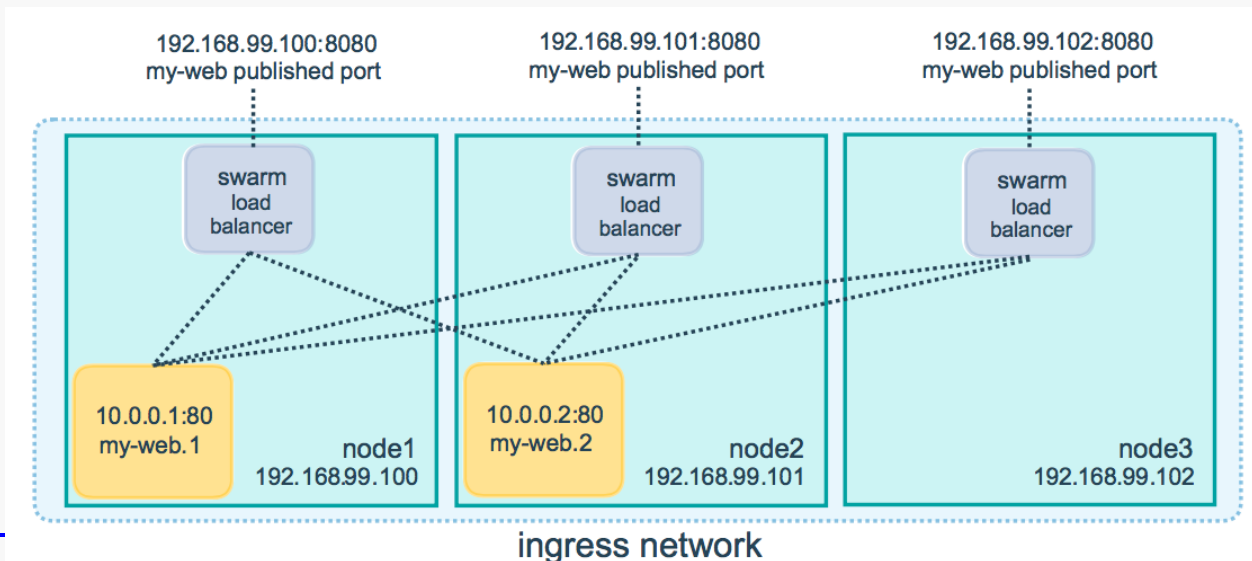
# Swarm Mode architektúra

- » A Docker Swarm Mode node-jait egy menedzser (**Manager**) vezérli
  - » Feladata: klaszter vezérlés, API biztosítása, erőforrás ütemezés
  - » Több Manager elosztott redundáns üzemet biztosít (high availability)
- » A **Worker** node = task-ok futtatása (Manager egyúttal worker is lehet)
  - » Worker node futtatás során Manager-ré nevezhető ki (és fordítva)
  - » Worker node-ok egy mesh hálózatban vannak szervezve



# Swarm mode hálózatkezelése

- » Szolgáltatáshoz port-ot rendelni
  - » Swarm-on kívüli kérések fogadása (ingress nw = bejövő hívást kezelő hálózat)
  - » A host-ok egy *Swarm mode routing mesh* részei kell legyenek
- » Minden host-on kell futnia egy terheléelosztó funkciót (load balancer) is végző modulnak
  - » Swarm mode routing mesh része
  - » Ez juttatja el a kérést a megfelelő konténerhez
  - » Akkor is, ha az a konténer más host-on fut
  - » Akkor is, ha a kérést először fogadó host-on nem is fut olyan task





---

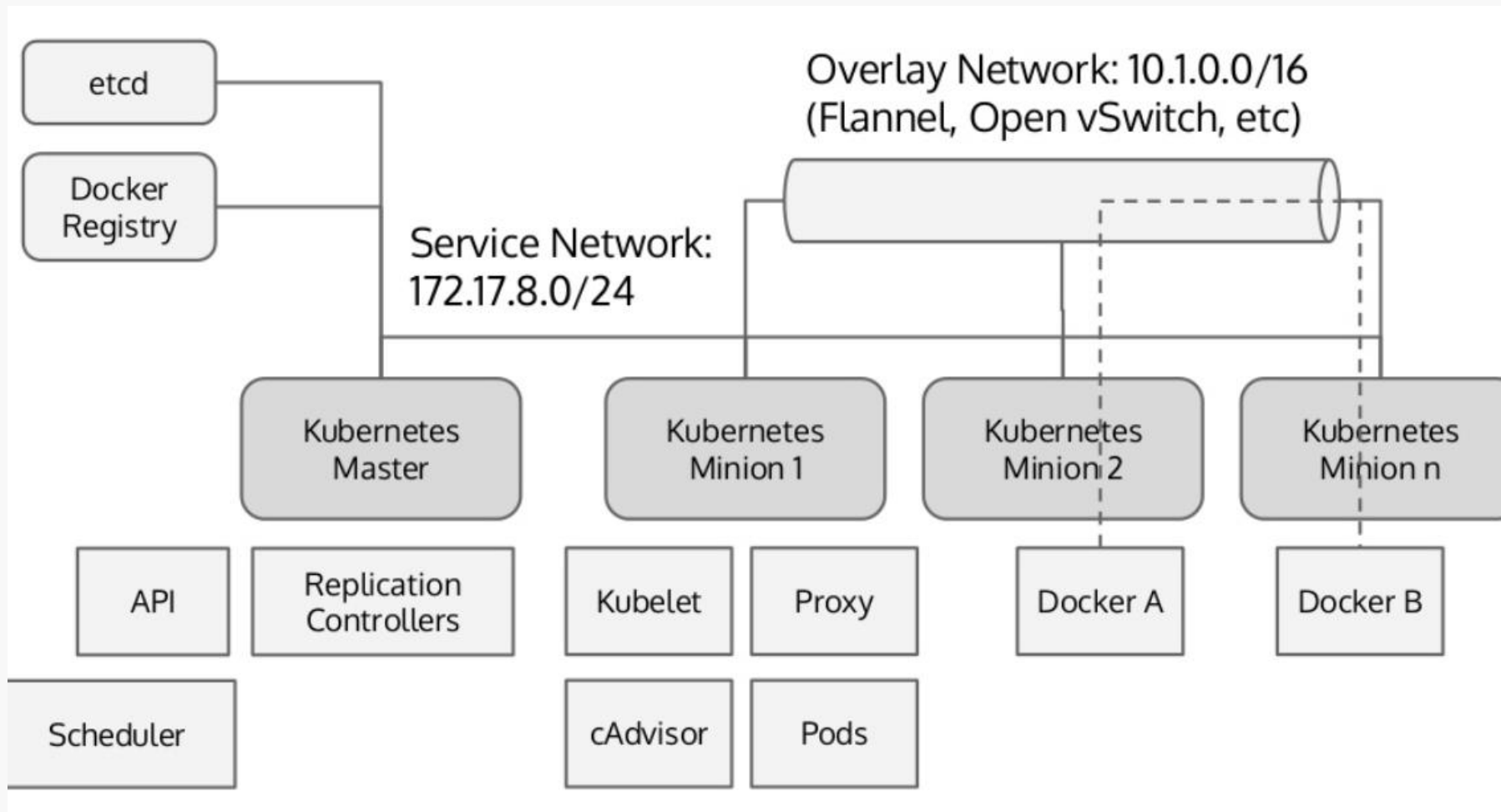
# KUBERNETES

---

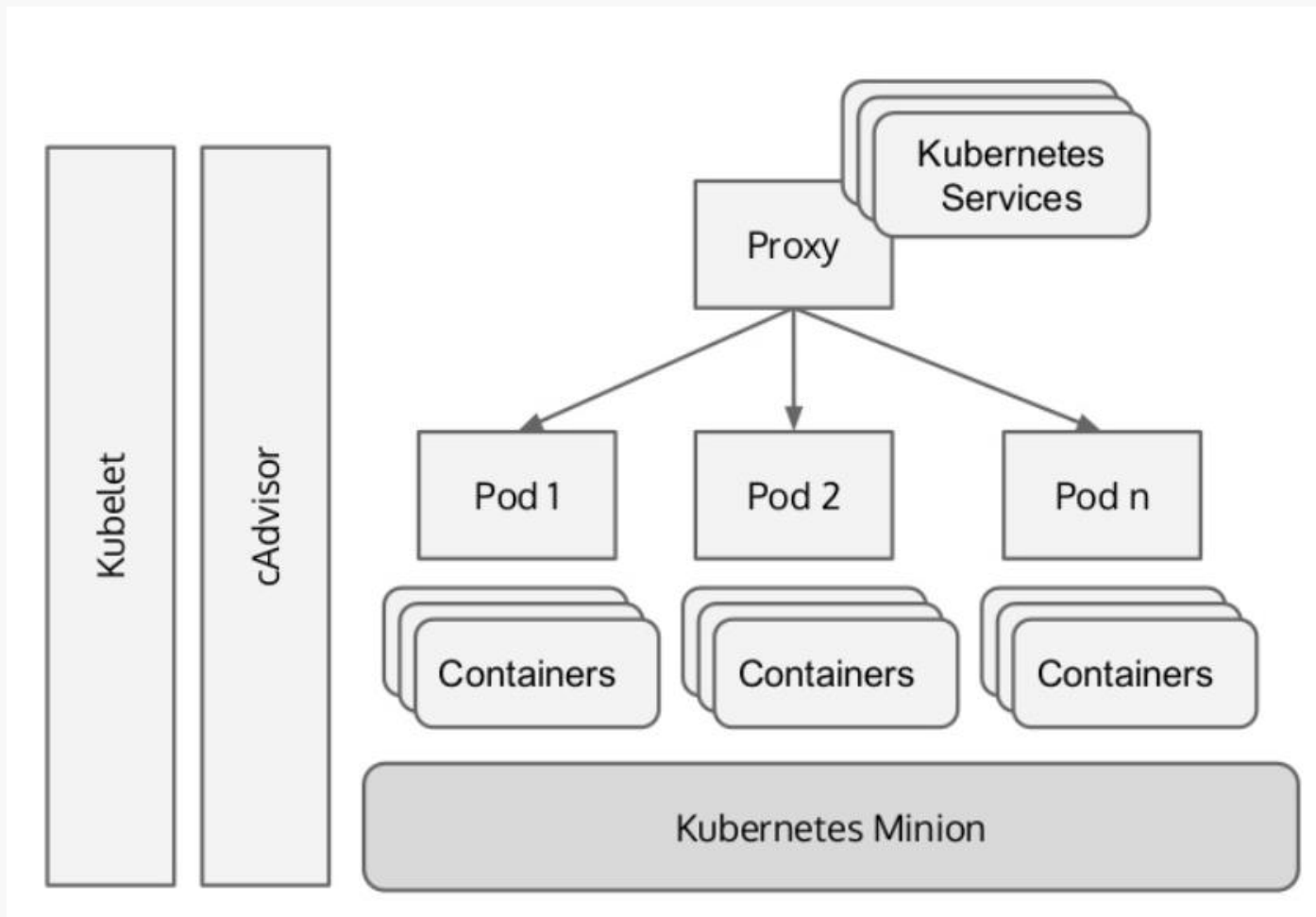
## Kubernetes – main components

- **Pod** - A group of Containers
- **Labels** - Labels for identifying pods
- **Kubelet** - Container Agent
- **Proxy** - A load balancer for Pods
- **etcd** - A metadata service
- **cAdvisor** - Container Advisor provides resource usage/performance statistics
- **Replication Controller** - Manages replication of pods
- **Scheduler** - Schedules pods in worker nodes
- **API Server** - Kubernetes API server

# Kubernetes deployment

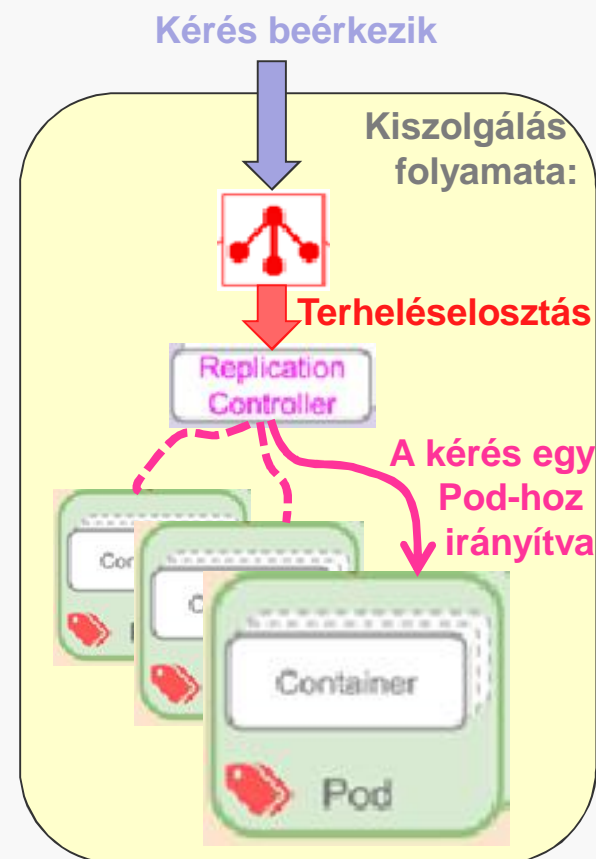
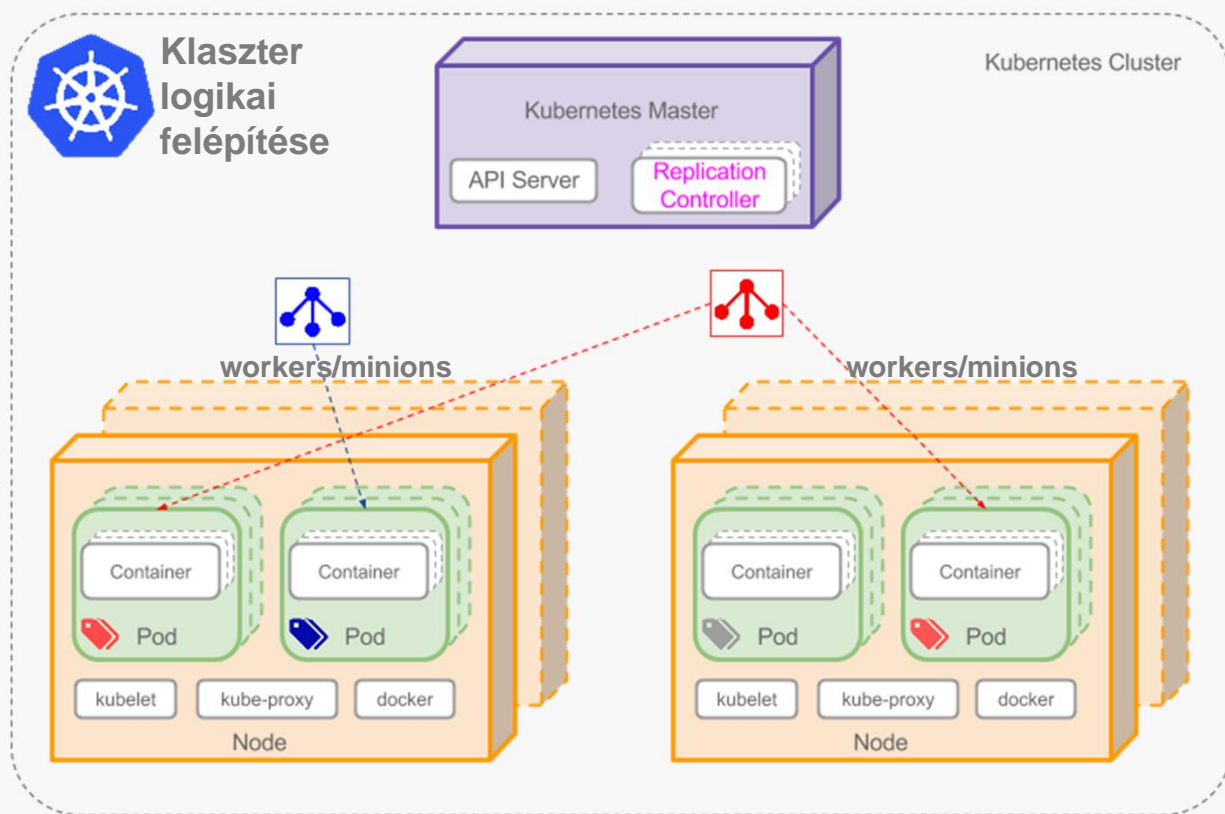


# Worker node = minion



# Egy Kubernetes klaszter logikai felépítése

- » A vezérlés a Master-en keresztül történik
- » A szolgáltatás elérése a „service”-n keresztül
  - » A Service kéréseit egy terheléselosztó (Replication Controller) kezeli
  - » A kérést egy konkrét Pod teljesíti

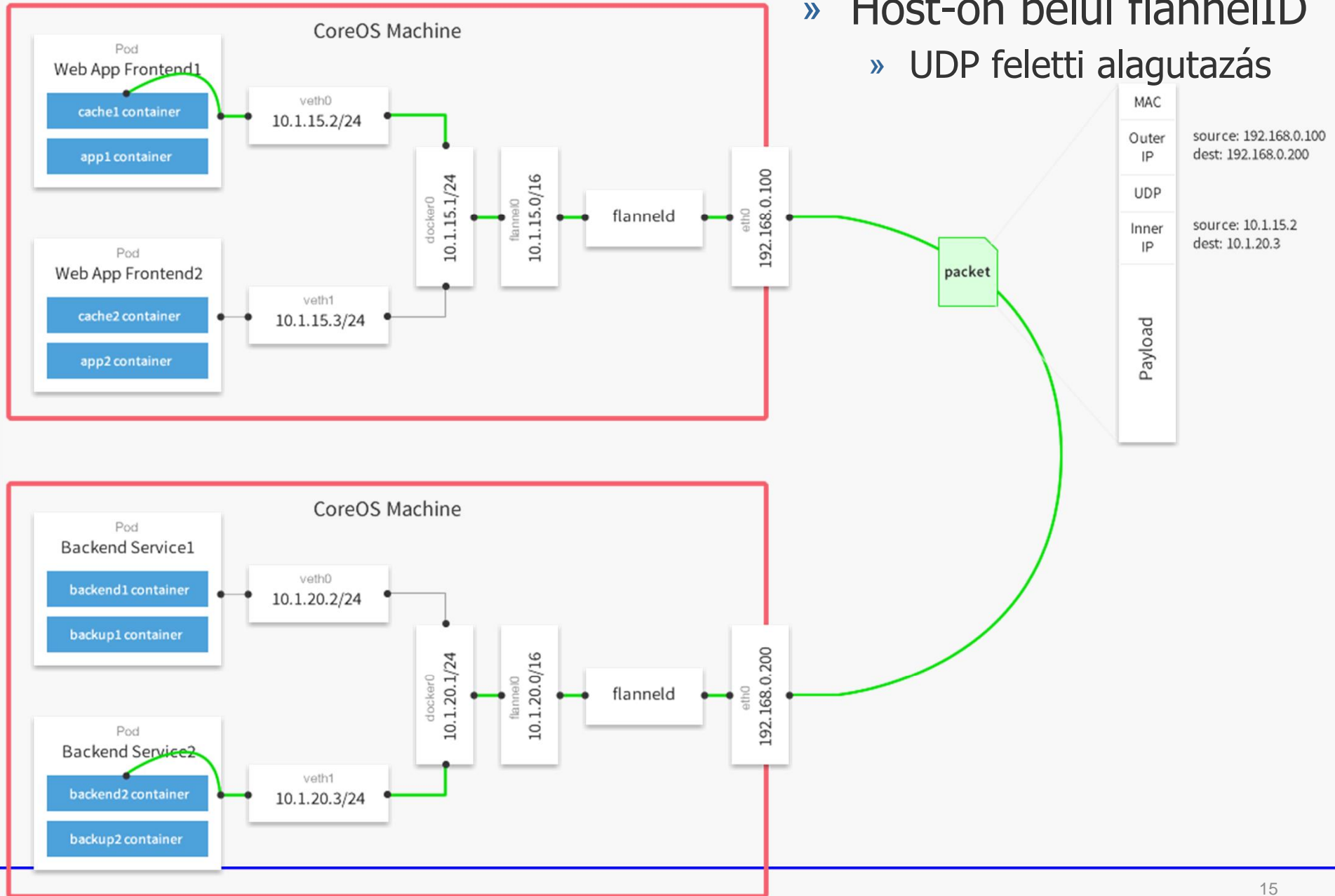


# Kubernetes hálózat

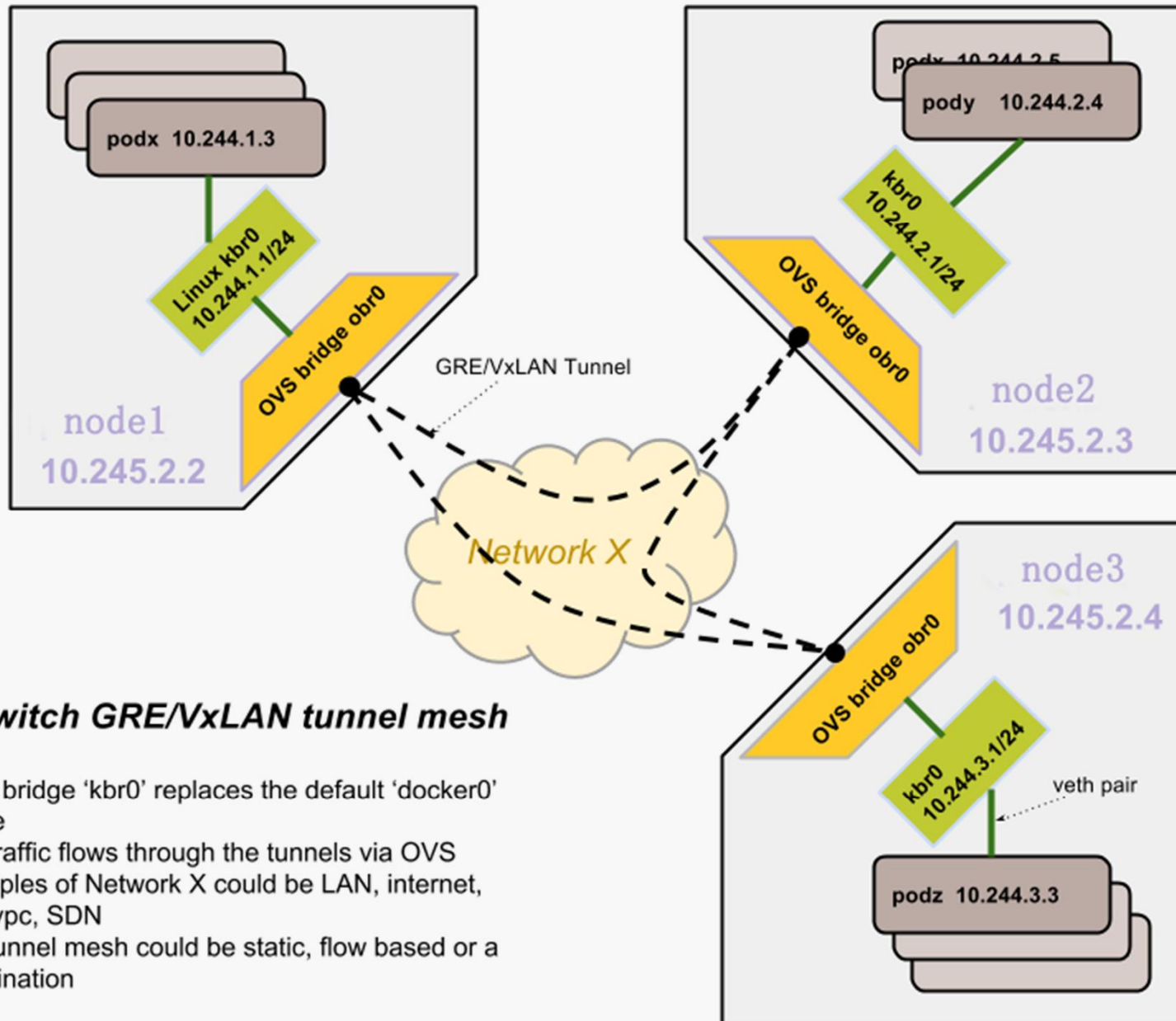
- » Pod szinten minden konténer egy névtérben van
  - » Előny: Localhost-on keresztül el tudják egymást érni
  - » Következmény: vigyázni kell a podon belüli konténerek portkiosztására (két konténer nem használhatja ugyanazt)
- » Host-ok felé is van elvárás: NAT nélkül kell kommunikálni a konténerekkel
- » Tipikus megoldás:
  - » Flannel: saját megoldás, egy „lapos átfedő” hálózat (flat overlay)
  - » OVS: Open VSwitch – generikus, iparágban elterjedt megoldás
  - » Sok más megoldás is lehetséges:  
<https://kubernetes.io/docs/concepts/cluster-administration/networking/#how-to-achieve-this>

# Kubernetes Flannel

- » Host-on belül flannelID
- » UDP feletti alagutazás



# OVS bridging



## OpenVSwitch GRE/VxLAN tunnel mesh

- Linux bridge 'kbr0' replaces the default 'docker0' bridge
- Pod traffic flows through the tunnels via OVS
- Examples of Network X could be LAN, internet, EC2 vpc, SDN
- The tunnel mesh could be static, flow based or a combination



---

# Demó

- » Kubernetes on-line demó
  - » Egy minta Pod indítása, kezelése online felületen

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