Kubernetes 1

Microservices

- Microservices are lightweight applications written in various modern programming languages, with specific dependencies, libraries and environmental requirements. To ensure that an application has everything it needs to run successfully it is packaged together with its dependencies.

Monolitikus alkalmazás vs. Microservices

Monolith	Microservices
Egy nagy szoftver, ami minden komponenst magában foglal	A rendszer minden logikailag elkülönített eleme külön entitást alkot, sok kis komponens
Minimális komponensek közötti kommunikáció	Komponensek közötti kommunikáció kulcsfontosságú
Egy, vagy kevés node-on nagy hardverigény	Több node-on kisebb hardverigény
Nehezen skálázható (load balancer)	Minden komponens különállóan fut, rugalmas
Bizonyos méretig könnyen átlátható	Kezdeti konfigurációja nehezebb, később az apróbb komponensek miatt menedzselhetőbb

Microservices – előnyök

- Könnyű skálázhatóság
- Könnyű átrendezhetőség
- Event-driven Architecture és Service-Oriented Architecture (SOA) alapelvek
- Modern programnyelvek
- Egyesével módosítható, újraírható komponensek
- Észrevétlen frissítés és újraindítás

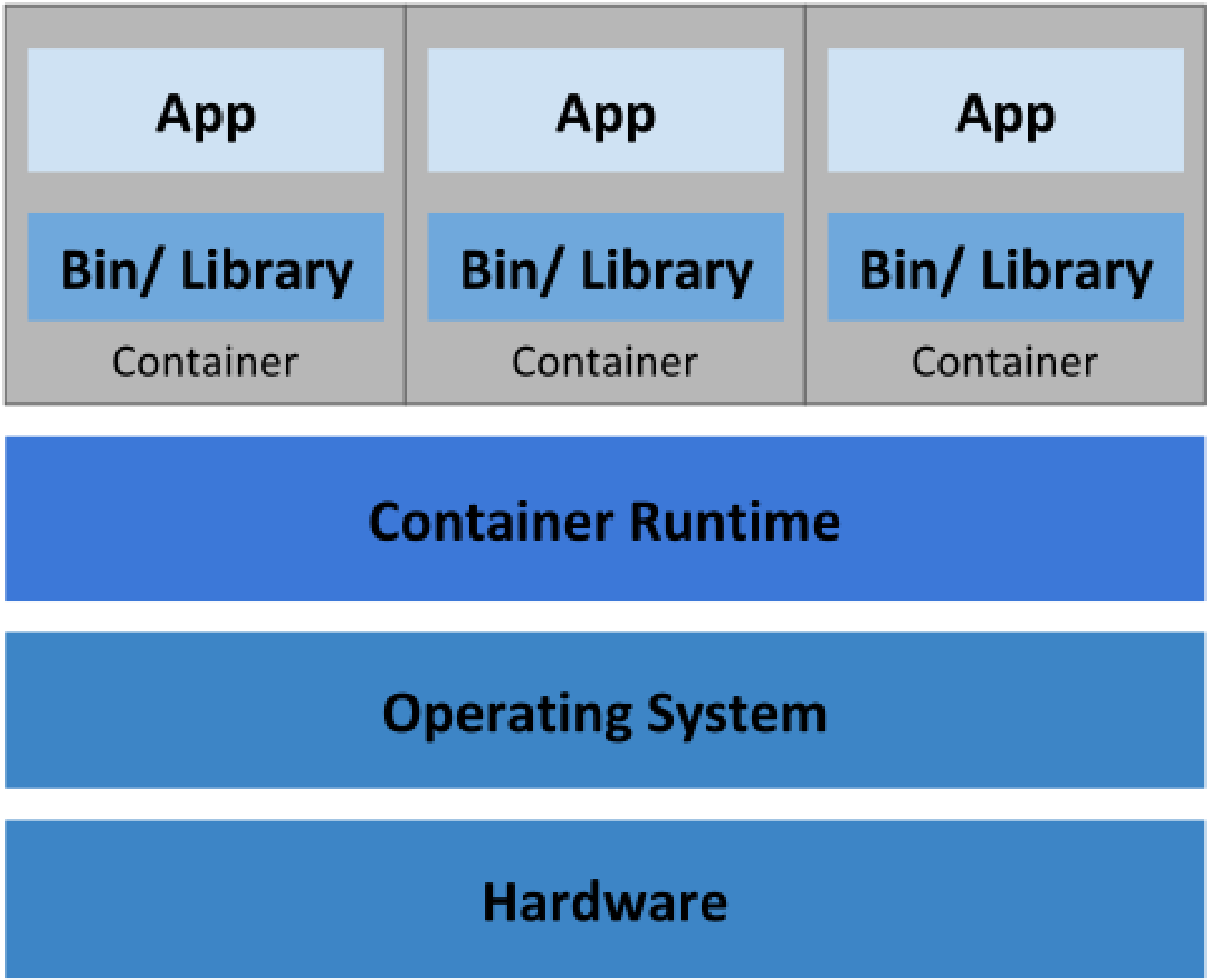
Áttérés microservices architektúrára

- Nem mindig érdemes és nem is mindig lehet
- Refactoring
  - Big bang
    - Csak a monolith teljes átírására fókuszál
    - Közben nincs fejlesztés
    - Hátrány: tönkre teheti a teljes üzleti logikát
  - Inkrementális
    - Fejlesztés közben át is alakítjuk a komponenseket
    - Hátrány: sokáig tart, nem biztos, hogy minden átalakul
- Ha sikerült, ki kell alakítani a komponensek runtime-ját és kapcsolatait

Konténerek

- Containers are an application-centric method to deliver high-performing, scalable applications on any infrastructure of your choice. Containers are best suited to deliver microservices by providing portable, isolated virtual environments for applications to run without interference from other running applications.
- 1-1 Microservice-t tartalmaznak container image-ekben
- Container image-eket futtatnak
- Tartalmazzák az alkalmazás összes szükséges függőségét, binárisokat és könyvtárakat

Konténerek


App
App
App
Bin/ Library
Bin/ Library
Bin/ Library
Container
Container
Container
Container Runtime
Operating System
Hardware

Konténer orchestration (hangszerelés)

- Fejlesztéskor futtathatjuk egy node-on az összes konténert
- QA és Prod stádiumban már nem
  - Hibatűrés
  - Szükség szerinti skálázás
  - Optimális erőforráshasználat
  - Auto-discovery
  - Észrevétlen update/rollback
- Container orchestrators are tools which group systems together to form clusters where containers' deployment and management is automated at scale while meeting the requirements mentioned above.

Konténer orchestration – előnyök

- Node-ok automatikus klaszterbe rendezése
- Konténerek futtatása erőforrások szerint
- Konténerek kommunikációja node-októl függetlenül
- Konténerek és tárhelyek összekapcsolása (lokális, felhő, NAS, stb.)
- Konténerek csoportosítása, csoportonkénti LB
- Észrevétlen és dinamikus elosztás, magasabb absztrakció

Kubernetes

- Kezdetben a Google fejlesztette, már a Cloud Native Computing Foundation része
- Borg rendszerből származik
- Go nyelven íródott
- 3 havonta jelenik meg újabb verzió

Kubernetes – jellemzők

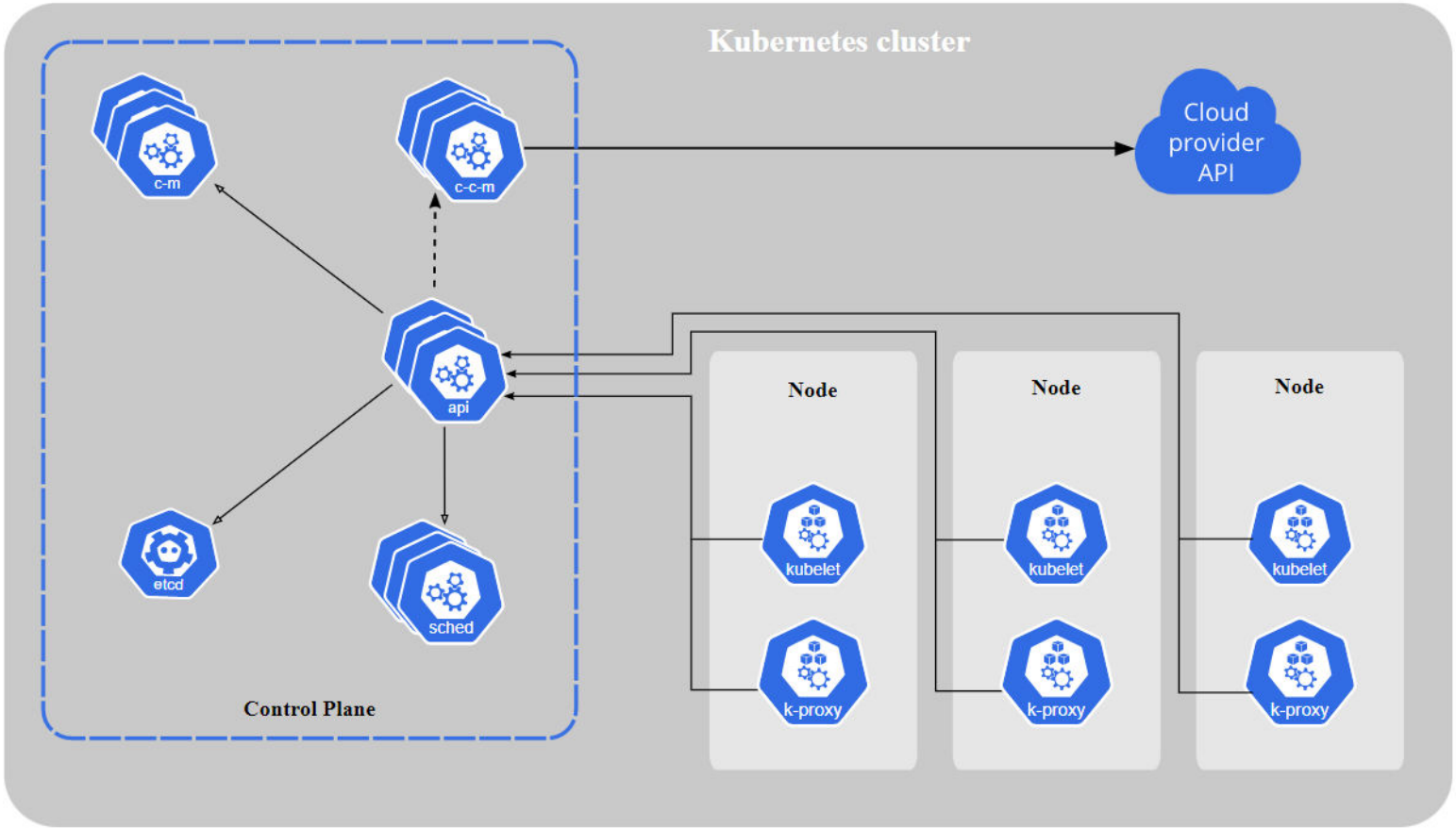
- Automatic bin packing: Kubernetes automatically schedules containers based on resource needs and constraints, to maximize utilization without sacrificing availability
- Self-healing: Kubernetes automatically replaces and reschedules containers from failed nodes. It kills and restarts containers unresponsive to health checks, based on existing rules/policy. It also prevents traffic from being routed to unresponsive containers
- Horizontal scaling: With Kubernetes applications are scaled manually or automatically based on CPU or custom metrics utilization
- Service discovery and Load balancing: Containers receive their own IP addresses from Kubernetes, while it assigns a single Domain Name System (DNS) name to a set of containers to aid in load-balancing requests across the containers of the set

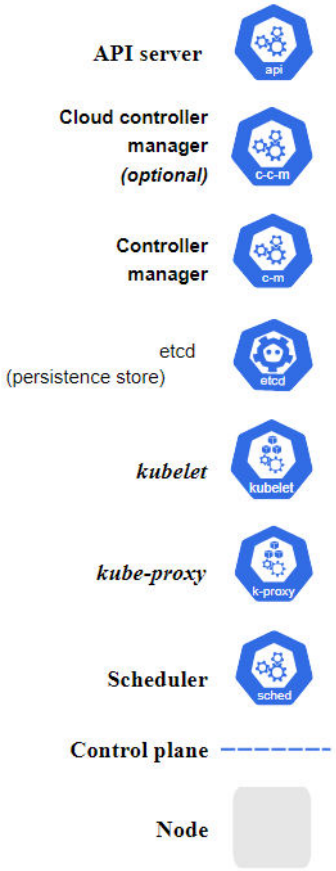
Kubernetes – jellemzők

- Automated rollouts and rollbacks: Kubernetes seamlessly rolls out and rolls back application updates and configuration changes, constantly monitoring the application's health to prevent any downtime
- Secret and configuration management: Kubernetes manages sensitive data and configuration details for an application separately from the container image, in order to avoid a re-build of the respective image. Secrets consist of sensitive/confidential information passed to the application without revealing the sensitive content to the stack configuration, like on GitHub
- Storage orchestration: Kubernetes automatically mounts software-defined storage (SDS) solutions to containers from local storage, external cloud providers, distributed storage, or network storage systems
- Batch execution: Kubernetes supports batch execution, long-running jobs, and replaces failed containers

Kubernetes – architektúra

- 1 vagy több master node -> control plane
- 1 vagy több worker node





Kubernetes – master node

- Biztosítja a futtató környezetet a control plane számára
- Agentek különböző management szerepekkel
- Control plane: kéréseket fogad a usertől
  - CLI
  - Web UI
  - API
- Fontos életben tartani bármi áron -> nélküle kiszámíthatatlan a működés
- Replikák hozhatók létre HA módban
- Minden kofigurációs adat etcd-ben tárolódik
  - Lehet a master node-on vagy saját host-on (szintén replikálható)

Kubernetes – master node

- Komponensei:
  - API server
  - Scheduler (ütemező)
  - Controller managerek
  - Data store
  - Container runtime
  - Node agent
  - Proxy

Kubernetes – API server

- kube-apiserver
- Külső forrásokból (pl. user) érkező hívásokat értelmez és végrehajt
- Egyetlen komponens, amely hozzáfér az etcd-hez
  - Olvassa az állapotot az etcd-ből, majd oda menti az új állapotot
  - Minden más komponens is csak rajta keresztül fér hozzá
- Horizontálisan skálázható
- Custom API server is használható

Kubernetes – Scheduler

- kube-scheduler
- Workload objektumokat (podokat) társít node-okhoz
- Nagy mértékben konfigurálható, saját ütemező is írható
- Döntéseknél figyelembe veszi a klaszter jelenlegi állapotát és az objektum szükségleteit
  - etcd-ből lekéri a node-ok erőforrás használatát
  - API-servertől lekéri a objektum szükségleteit
    - Előre megadott kényszerek (pl. disk==ssd)
- QoS elvárások (data locality, affinity, cluster topology, stb.)
- A node-okat osztályozza, kiválasztja a legmegfelelőbbet
  - A döntést visszaküldi az API servernek
- Nagy klaszterek esetén rendkívül fontos és összetett a feladata

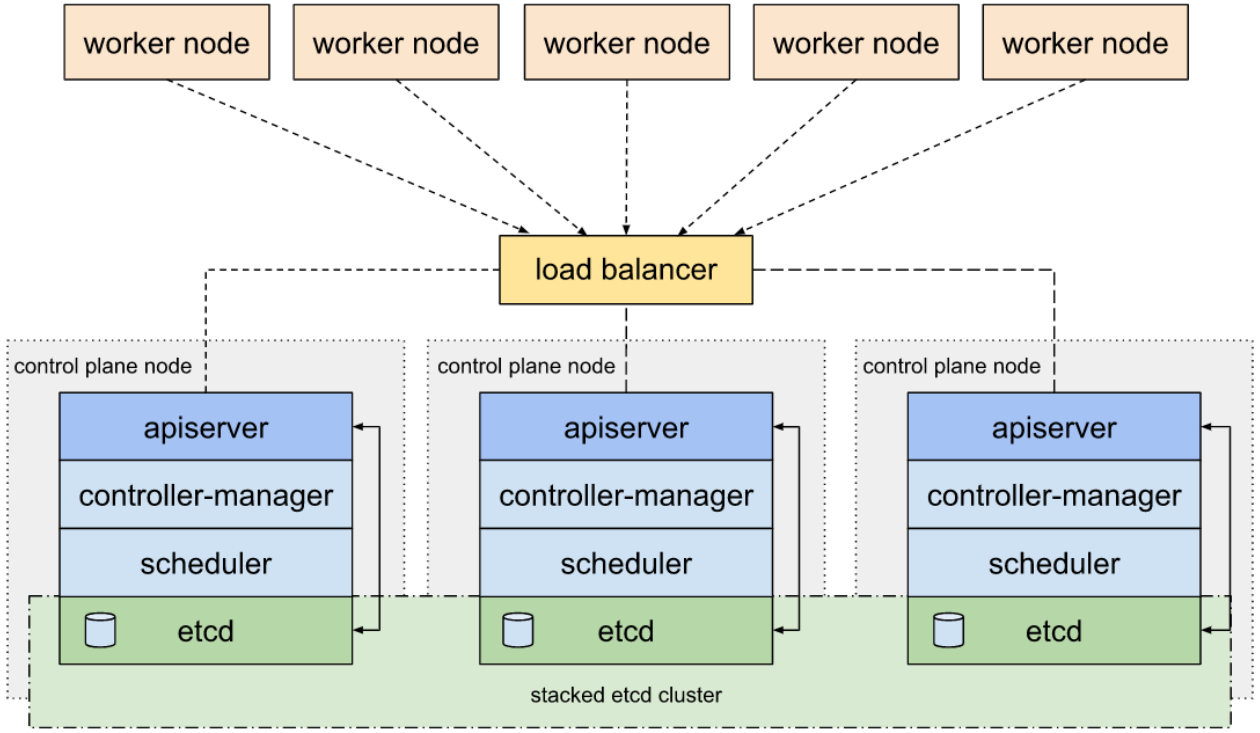
Kubernetes – controller managerek

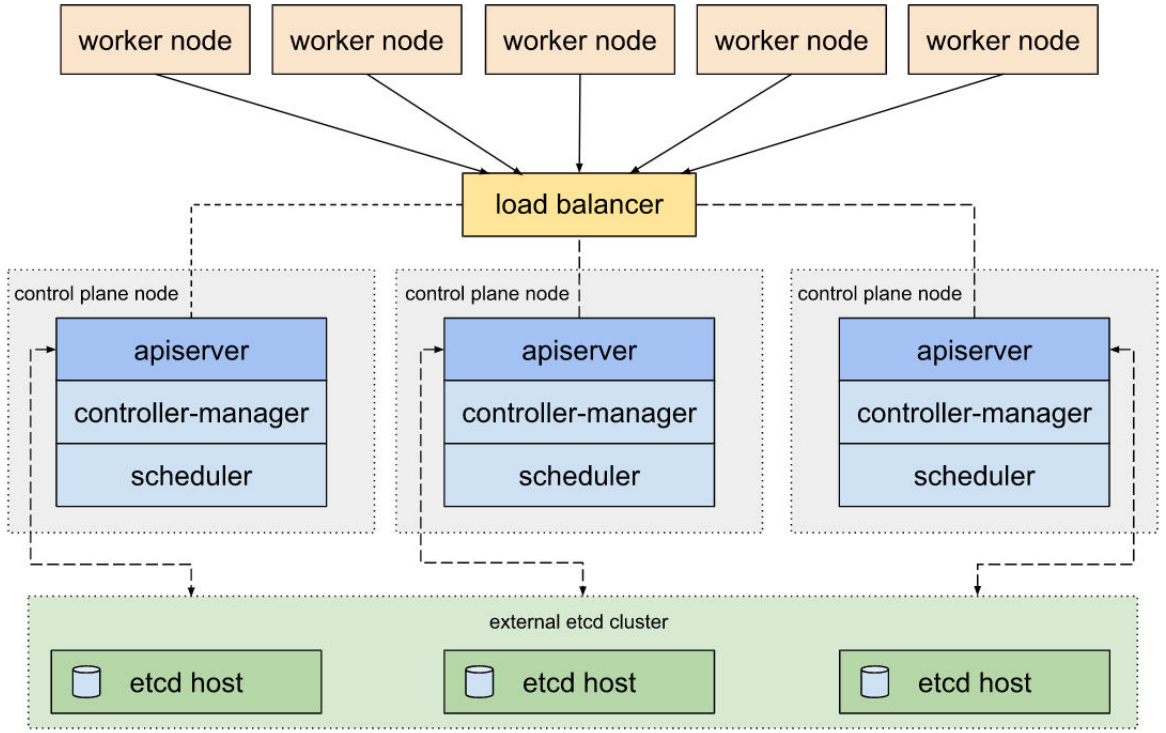
- A klaszter állapotát szabályozzák
- Folyamatos watch-loopokban hasonlítja az aktuális állapotot az elvárthoz
  - Elvárt állapot az objektum konfigurációból
  - Aktuális állapot etcd-ből
  - Eltérés esetén intézkedik
- kube-controller-manager: Olyan kontrollerek, amelyek a intézkednek ha egy node nem elérhető, end-pointokat készítenek, stb.
- cloud-controller-manager: Olyan kontrollerek, amelyek a cloud node-okért vagy szolgáltatásokért felelősek

Kubernetes – data store

- etcd
- Mindig csak hozzáfűzni lehet
  - A korábbi adatokat időnként tömöríti
- etcdctl – CLI management
- Prod környezetben fontos, hogy replikáljuk (HA)
- Raft Consensus Algorithm
  - Több etcd esetén valamelyik mindig master, de ez cserélődhet -> HA

Kubernetes – stacked vs external etcd





Kubernetes – worker node

- Futtatási környezetet nyújt az alkalmazásoknak
- A microservice-ek podokba vannak csomagolva
  - Cluster control plane agent-ek vezérlik (master node)
  - A pod a legkisebb futtatási egység a Kubernetesben
    - Egy vagy több egyszerre ütemezett konténer logikai gyűjteménye
      (It is a logical collection of one or more containers scheduled together, and the collection can be started, stopped, or rescheduled as a single unit of work)
- A különböző wn-eken futó podok hálózati kapcsolatai a wn-eken belül vannak megoldva, nincs route-olva az mn-en keresztül

Kubernetes – worker node

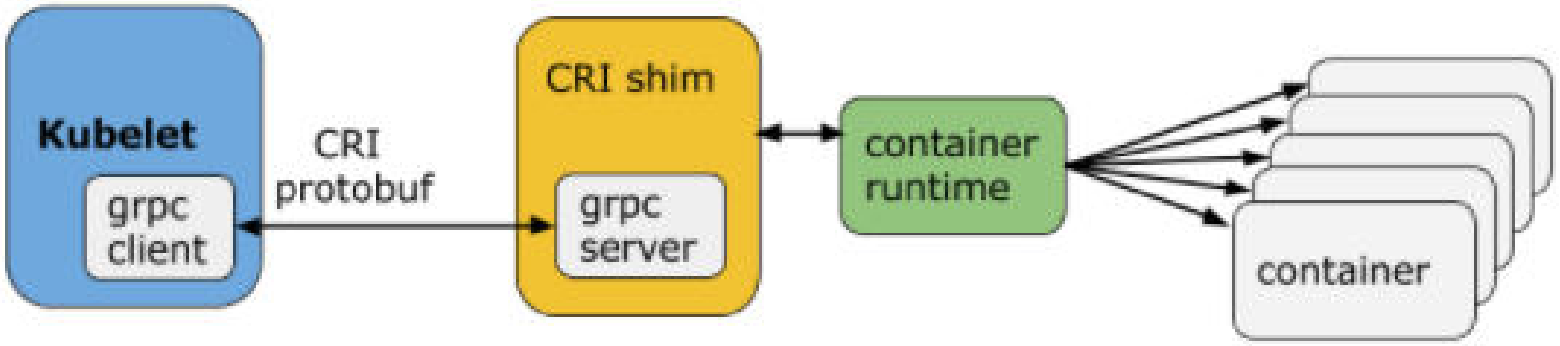
- Komponensei:
  - Container runtime
  - Node agent
  - Proxy
  - Add-onok (DNS, dashboard, monitoring, logging)

Kubernetes – container runtime

- A Kubernetes nem tud közvetlenül konténereket futtatni
- Szükség van container runtime-ra
- Támogatott container runtime-ok:
  - Docker (containerdet használ)
  - CRI-O
  - containerd
  - frakti

Kubernetes – node agent (kubelet)

- Minden node-on fut és kommunikál a control plane komponenseivel
- Pod definíciókat kap az API servertől és kommunikál a container runtime-mal (Container Runtime Interface-en keresztül), hogy futtathassa a benne lévő konténereket
- Monitorozza a podok életciklusát és erőforrásait
- shim – absztrakciós réteg a kubelet és a container runtime között



Kubernetes – proxy (kube-proxy)

- Hálózati ügynök, amely minden node-on fut
- A dinamikus update-ekért és hálózati szabályokért felelős
- Absztrakciót biztosít a podok hálózatához
- A kapcsolódási kérelmeket a megfelelő podhoz továbbítja
- TCP, UDP és SCTP streameket továbbít
- Több pod backend esetén round-robint alkalmaz
- Implementálja a felhasználók által definiált továbbítási szabályokat

Kubernetes – add-on-ok

- Olyan klaszterfunkciók, amelyek egyelőre nem a Kubernetes részei, ezért 3rd party podok és szolgáltatások formájában működnek
  - DNS - cluster DNS is a DNS server required to assign DNS records to Kubernetes objects and resources
  - Dashboard - a general purposed web-based user interface for cluster management
  - Monitoring - collects cluster-level container metrics and saves them to a central data store
  - Logging - collects cluster-level container logs and saves them to a central log store for analysis.