

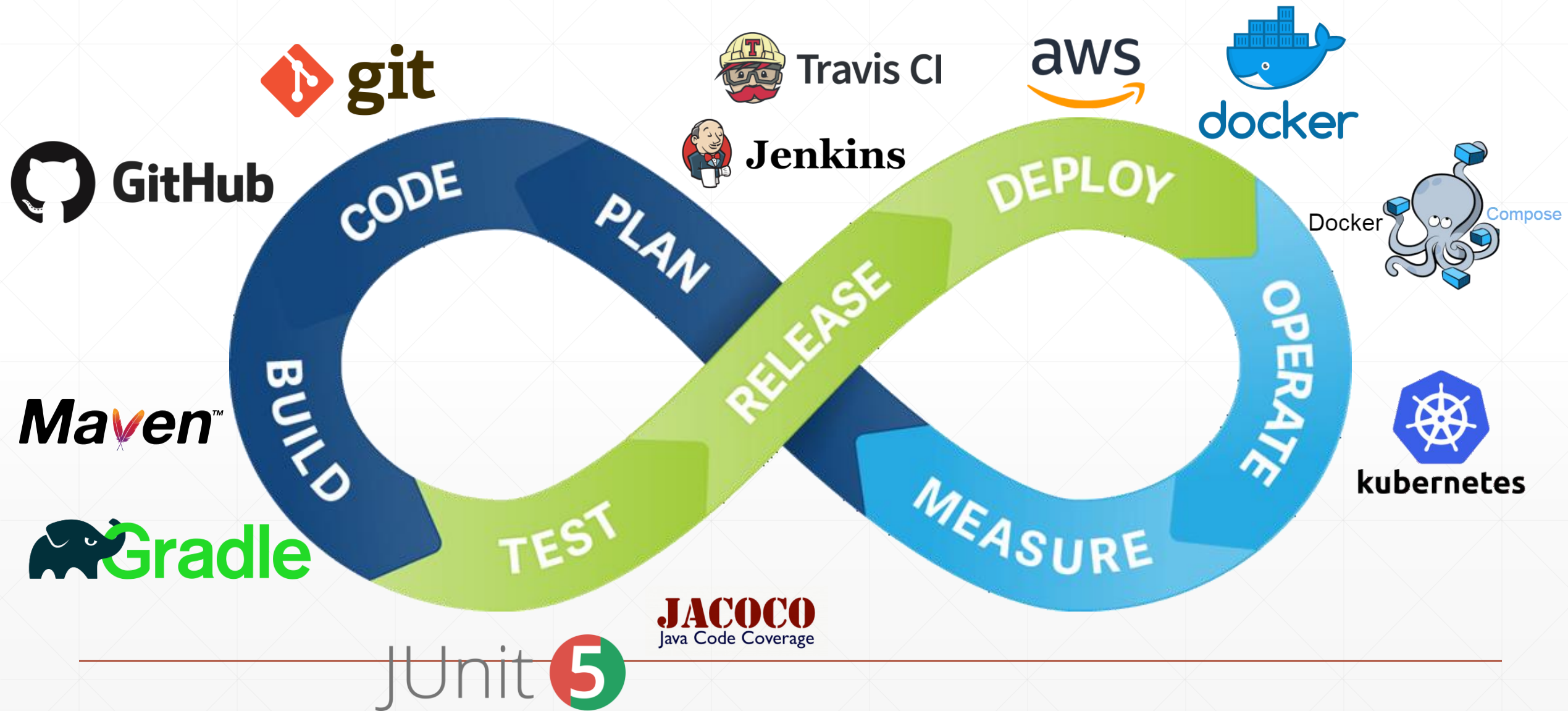
# DevOps témalabor beszámoló

---

Orova Márton

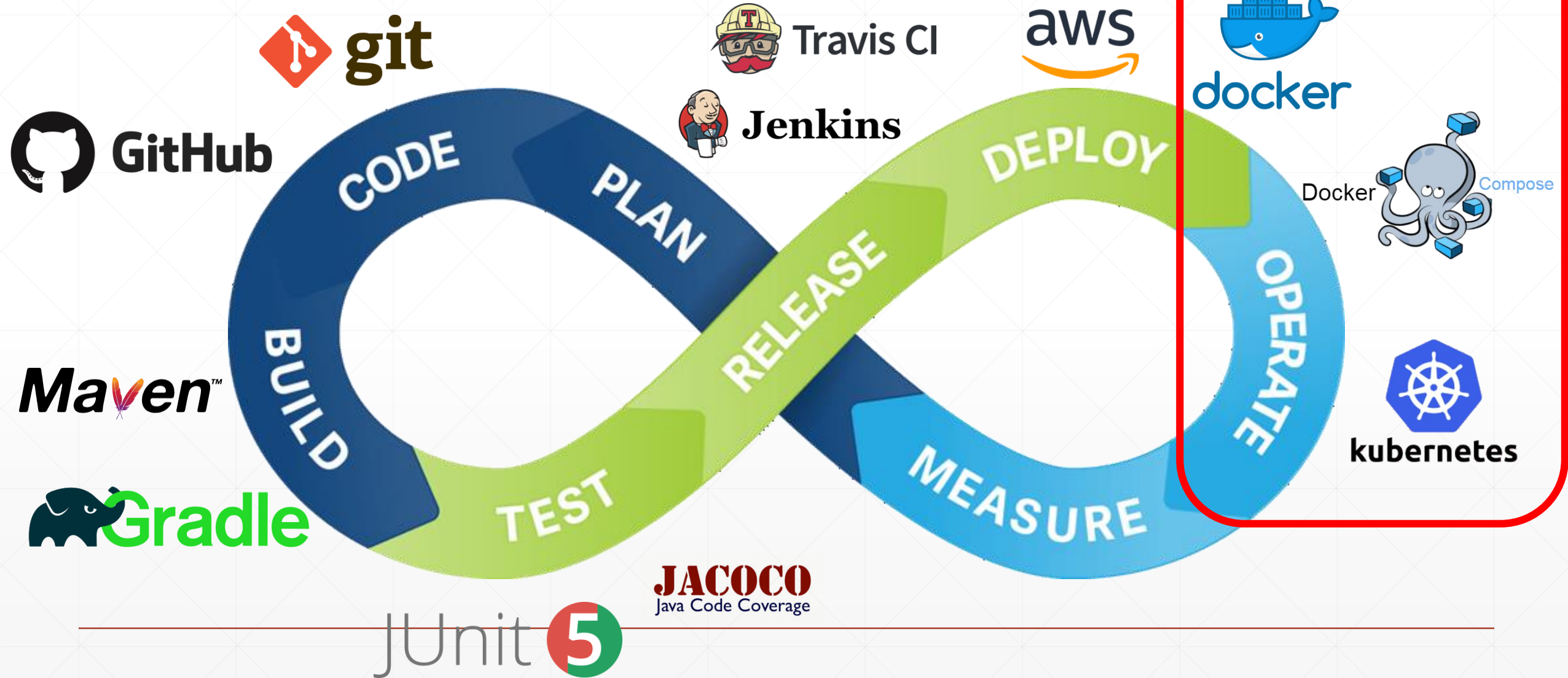
Mi is a DevOps?

Development + Operations



Mi is a DevOps?

Development + Operations



# Docker - Konténer alapú virtualizáció



## Docker konténer

- Alkalmazás + függőségek
- Bárhol fut, ha van Docker Engine
- Portábilis alkalmazás bundle
- Egymástól elszigeteltek

### Dockerfile

```
# Start with a base image containing Java runtime
FROM openjdk:8-jdk-alpine

# Make port 8080 available to the world outside this container
EXPOSE 8080

# The application's jar file
ARG JAR_FILE=build/libs/devops-demo-0.1.0.jar

# Add the application's jar to the container
ADD ${JAR_FILE} devops-demo.jar

# Run the jar file
ENTRYPOINT ["java","-jar","/devops-demo.jar"]
```

### Indítás

```
docker build -t devops-demo .
docker run -d -p 8080:8080 devops-demo
```

# Docker vs virtuális gép

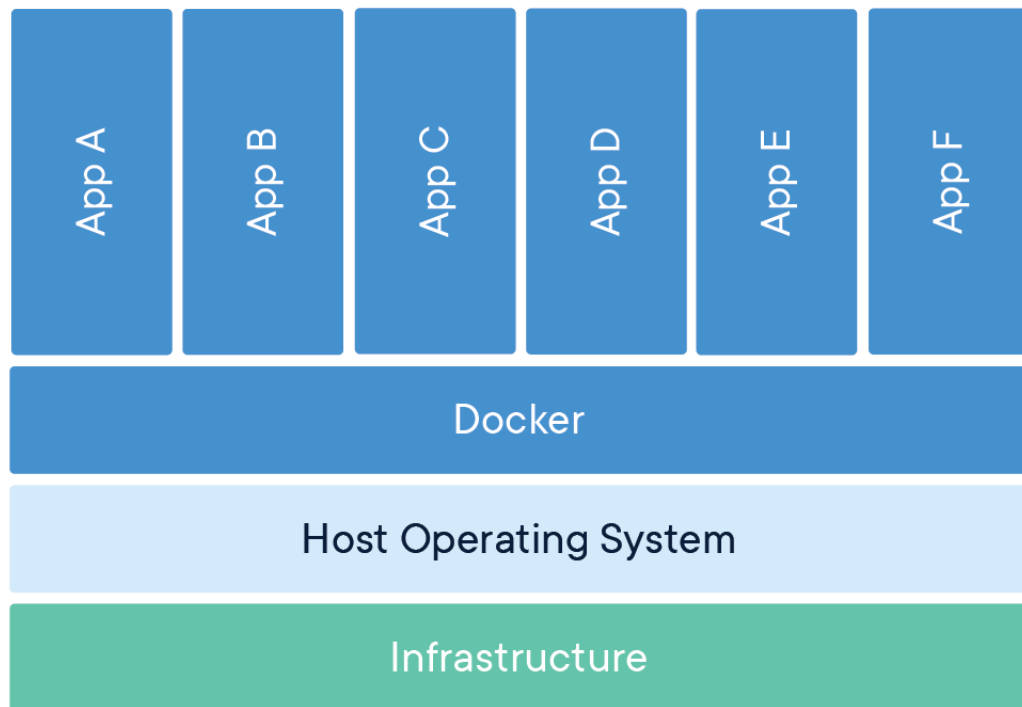
- Közös OS kernel használat



Kis méret

Containerized Applications

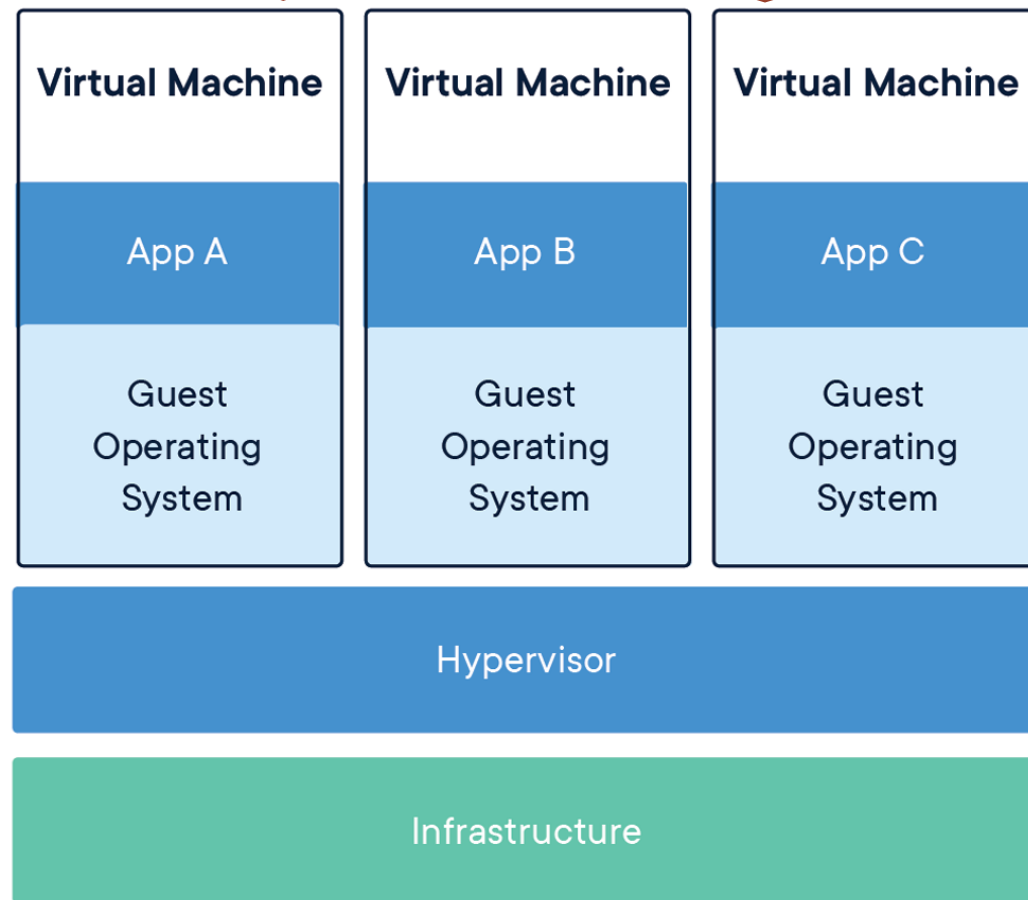
Könnyen hordozható



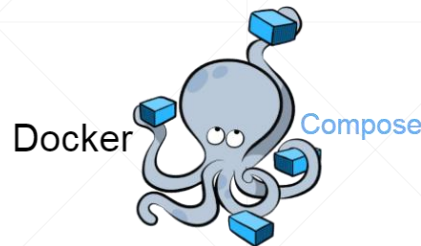
- Teljes virtuális OS

Nagy méret

Boot idő?



# Docker Compose



- Több konténeres Docker alkalmazás futtatása
- Alapvetően egy host-on
- Több konténer indítása egyszerre
  - docker-compose.yml
- Networking
  - konténer név alapján láthatóak és elérhetőek az egyes service-ek
- Skálázás service-enként

```
docker-compose up
```

```
version: '3'
```

```
services:
```

```
  db:
```

```
    image: mysql:latest
```

```
    environment:
```

- MYSQL\_ROOT\_PASSWORD=password
- MYSQL\_DATABASE=devops\_demo\_db
- MYSQL\_USER=user
- MYSQL\_PASSWORD=userpw

```
    volumes:
```

- /tmp/data/mysql

```
  devops-demo-app:
```

```
    restart: on-failure
```

```
    image: devops-demo
```

```
    build:
```

```
      context: ./
```

```
      dockerfile: Dockerfile
```

```
    depends_on:
```

- db

```
    ports:
```

- 8080:8080

```
    volumes:
```

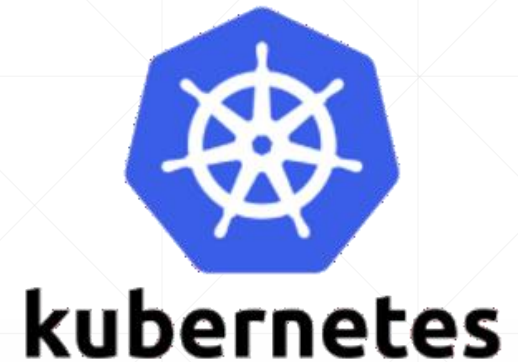
- /tmp/data/devops-demo-app

```
    environment:
```

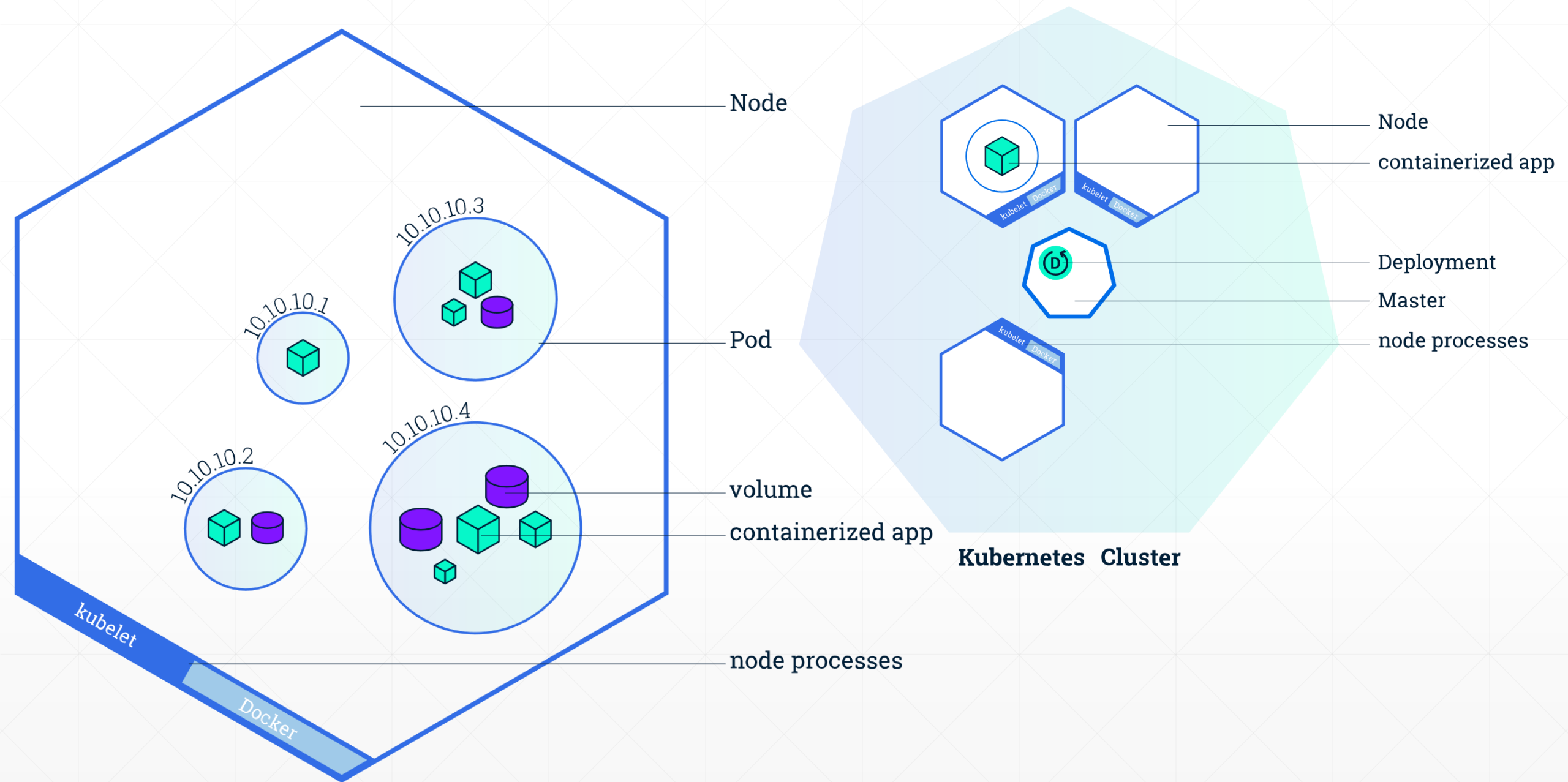
- DATABASE\_HOST=db
- DATABASE\_NAME=devops\_demo\_db
- DATABASE\_USER=user
- DATABASE\_PASSWORD=userpw
- DATABASE\_PORT=3306

# Kubernetes (K8s)

- Platform több konténer több host-on való menedzselésére
  - Jellemzően a felhőben
- Konténerek logikai csoportosítása
  - Pod, Label, Service
- Automatikus
  - Terheléselosztás
  - Skálázás
  - Ütemezés ('desired state' kontrollált elérése)
- Web UI (Dashboard)

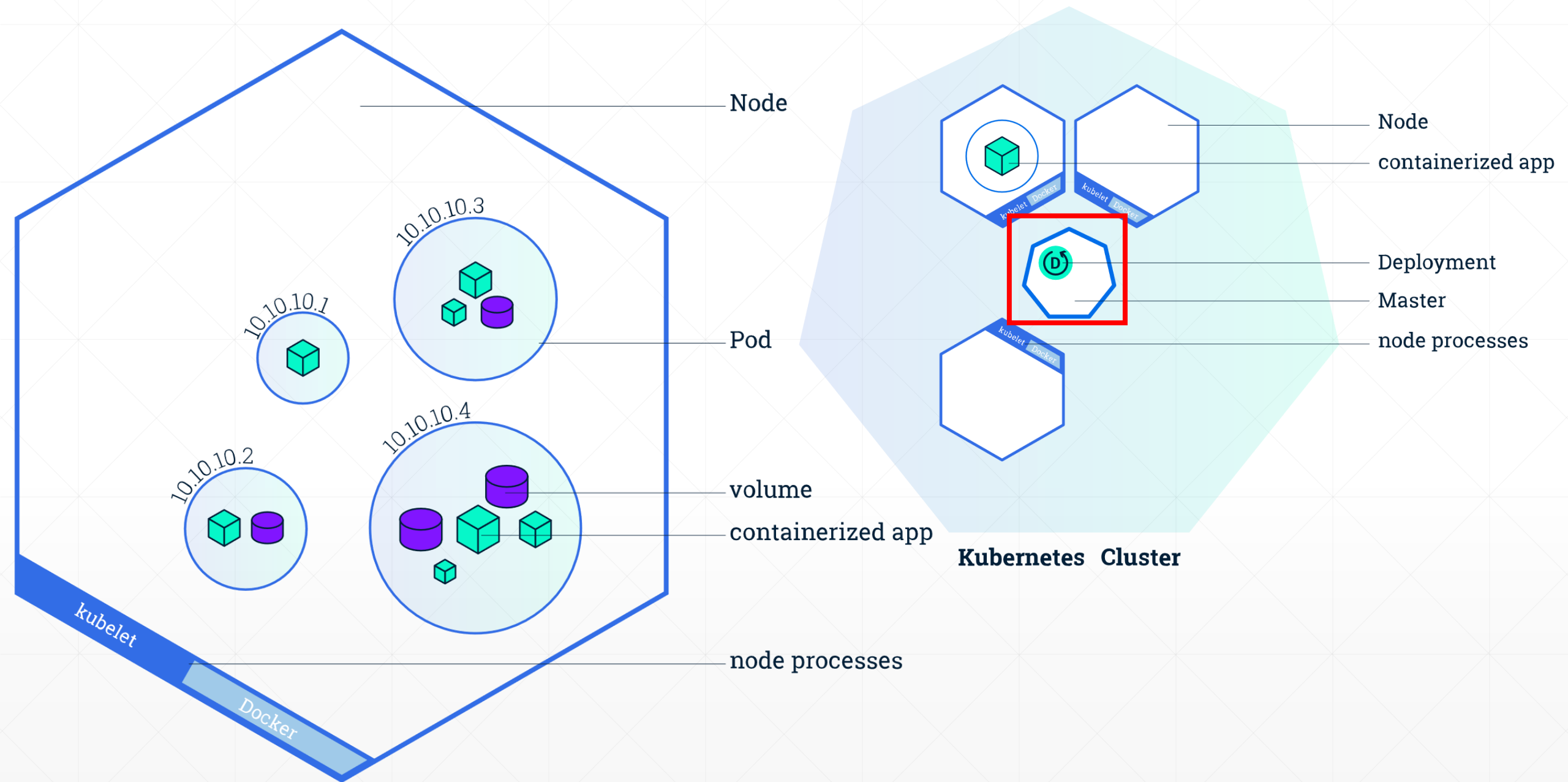




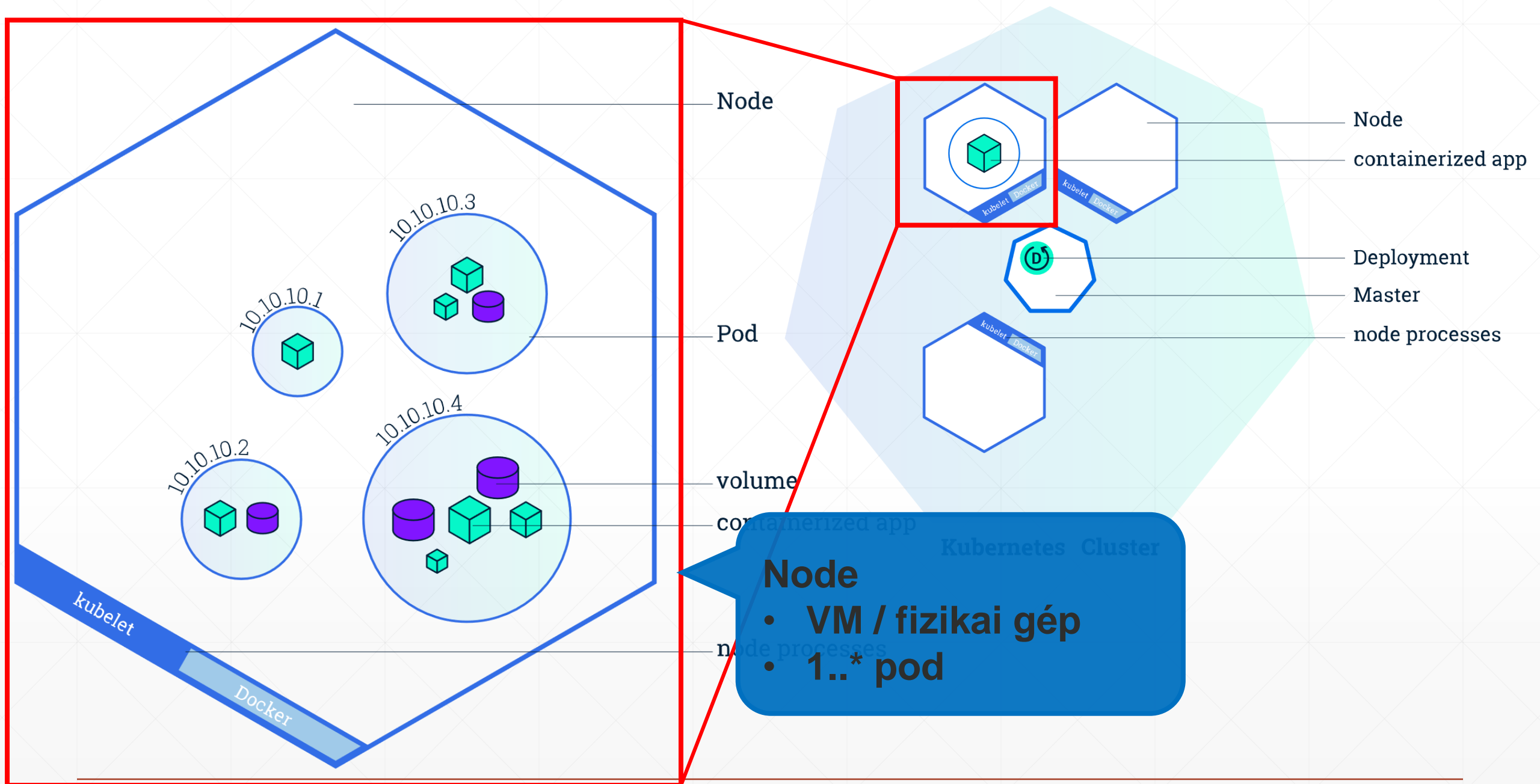


```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

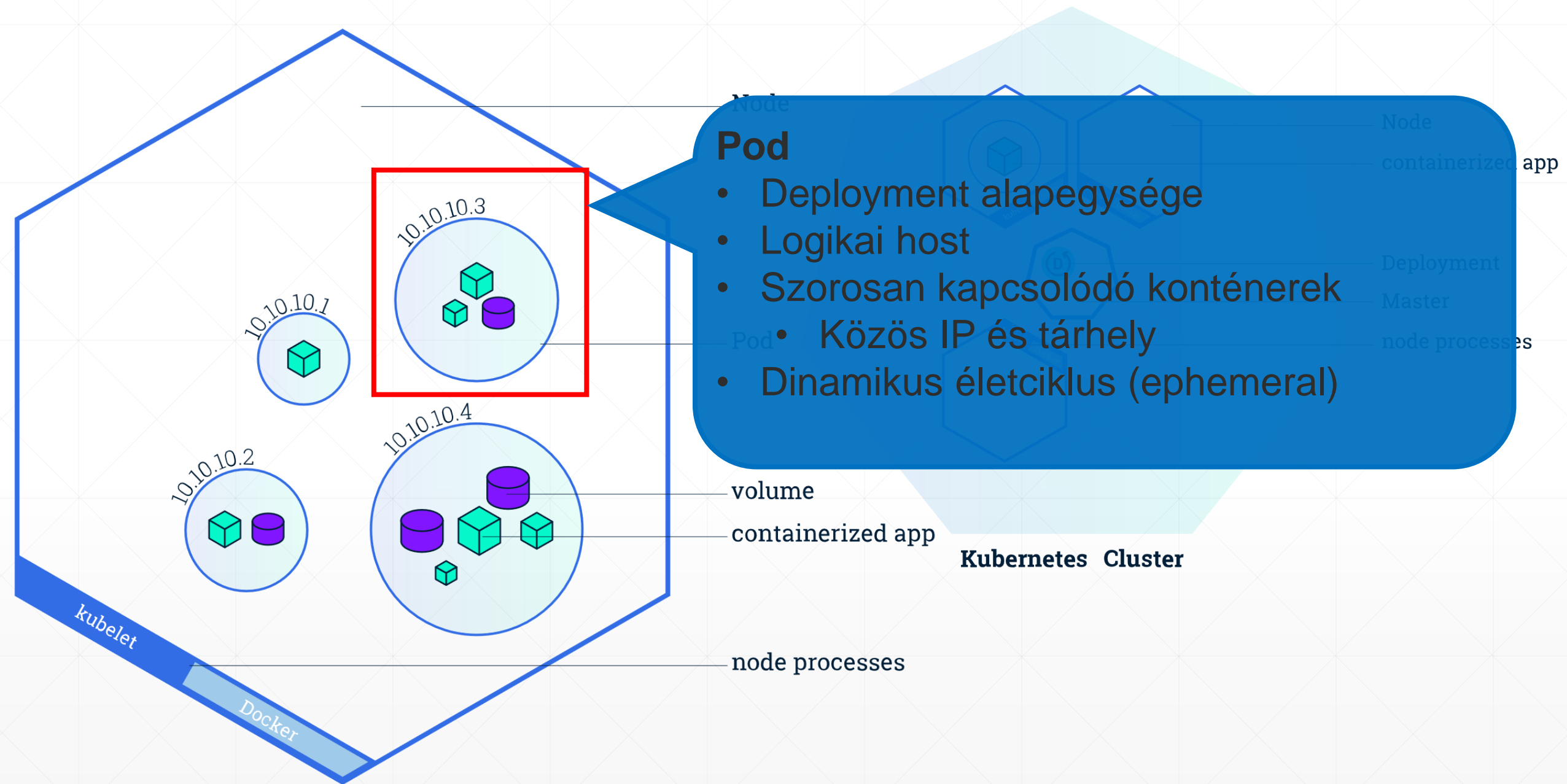




```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

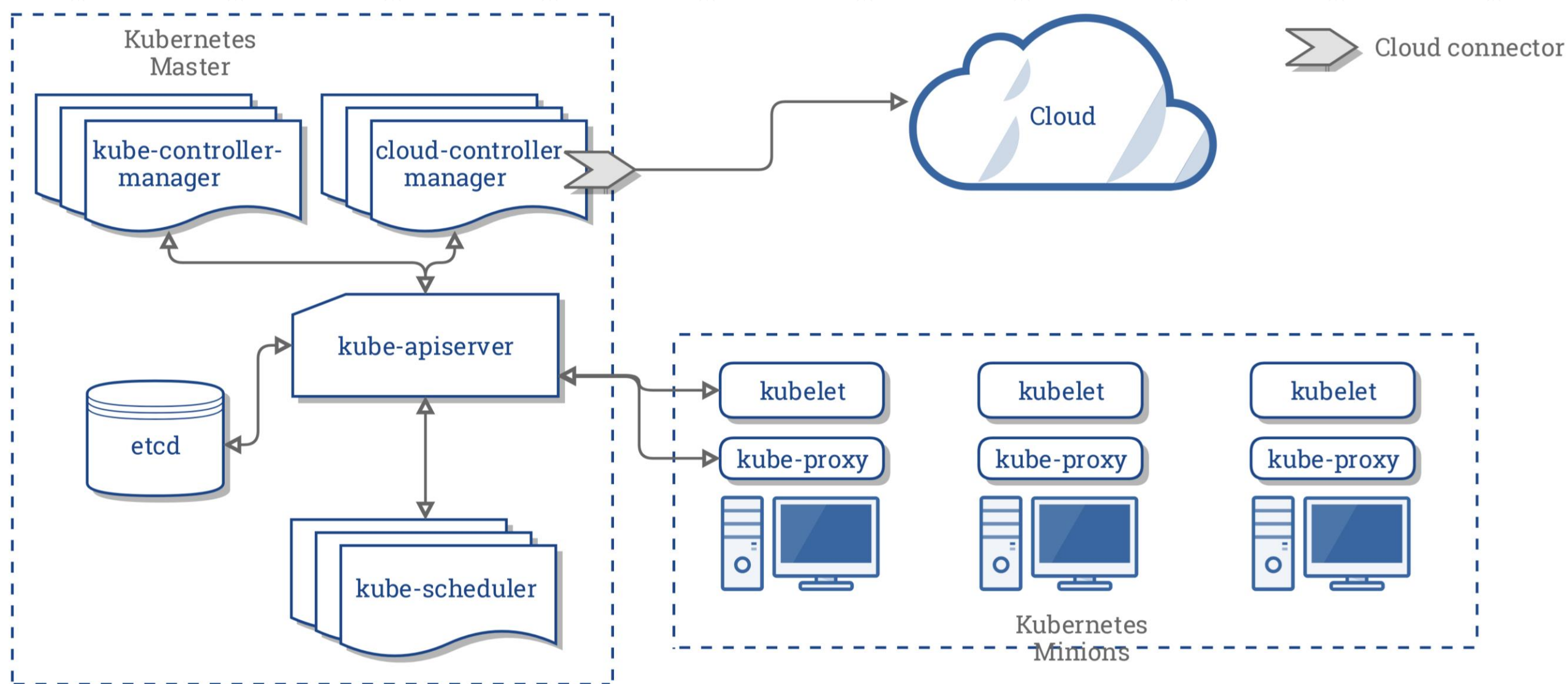


```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

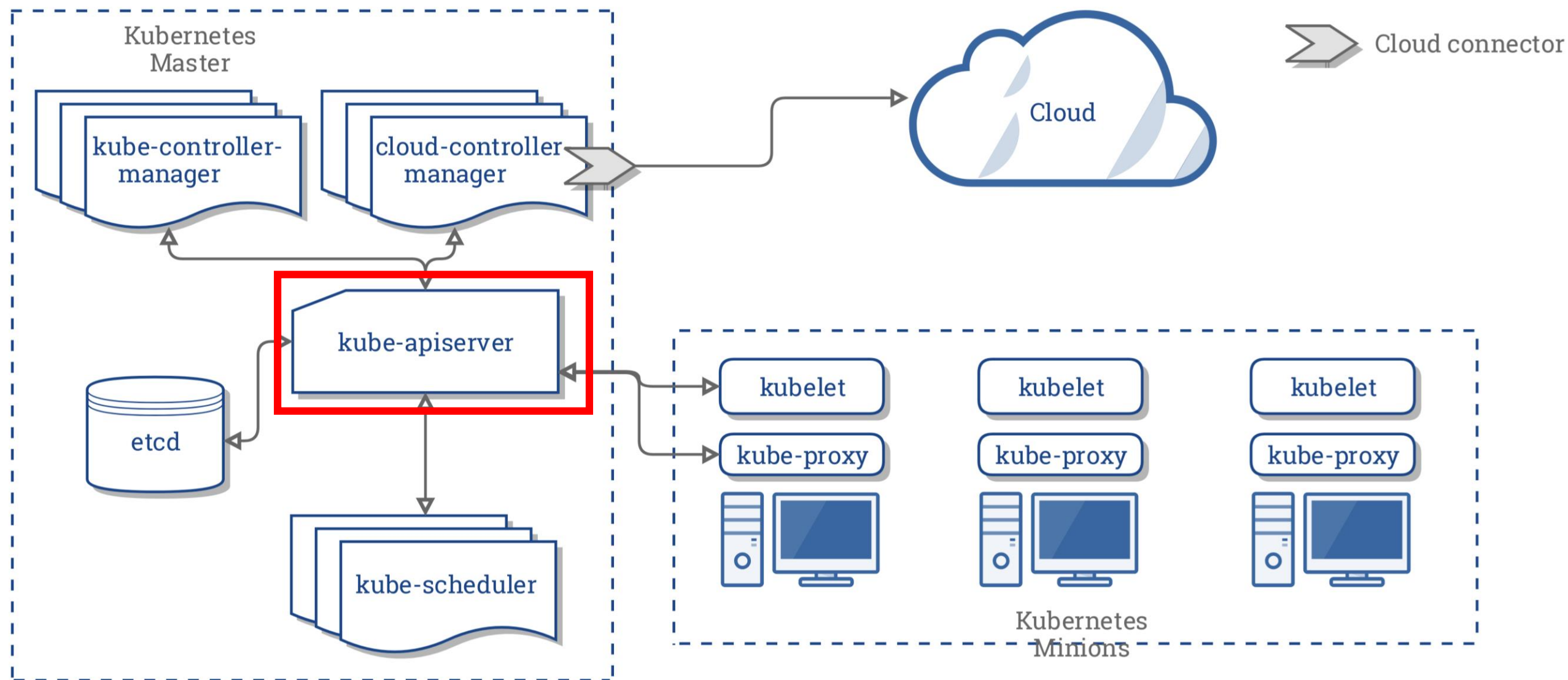


```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

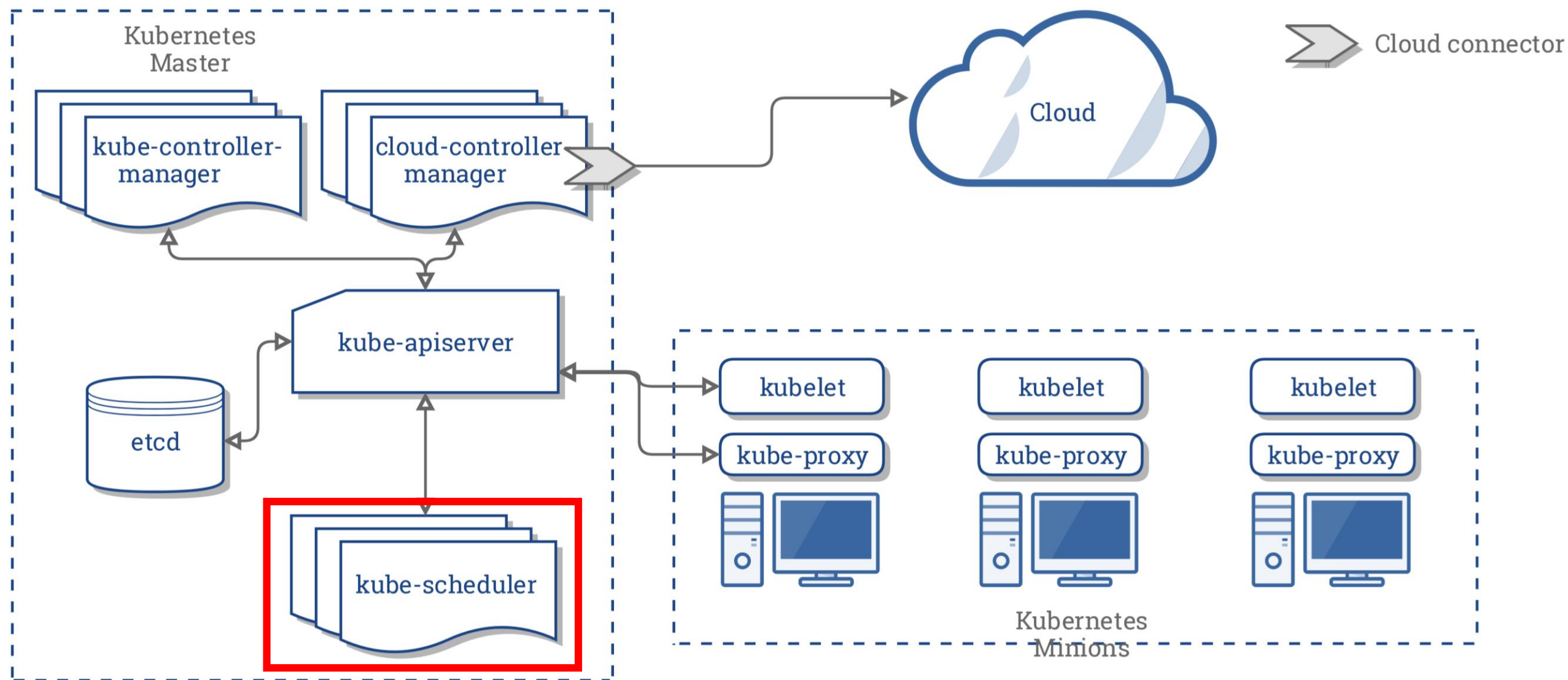
# Kubernetes architektúra



# Kubernetes architektúra

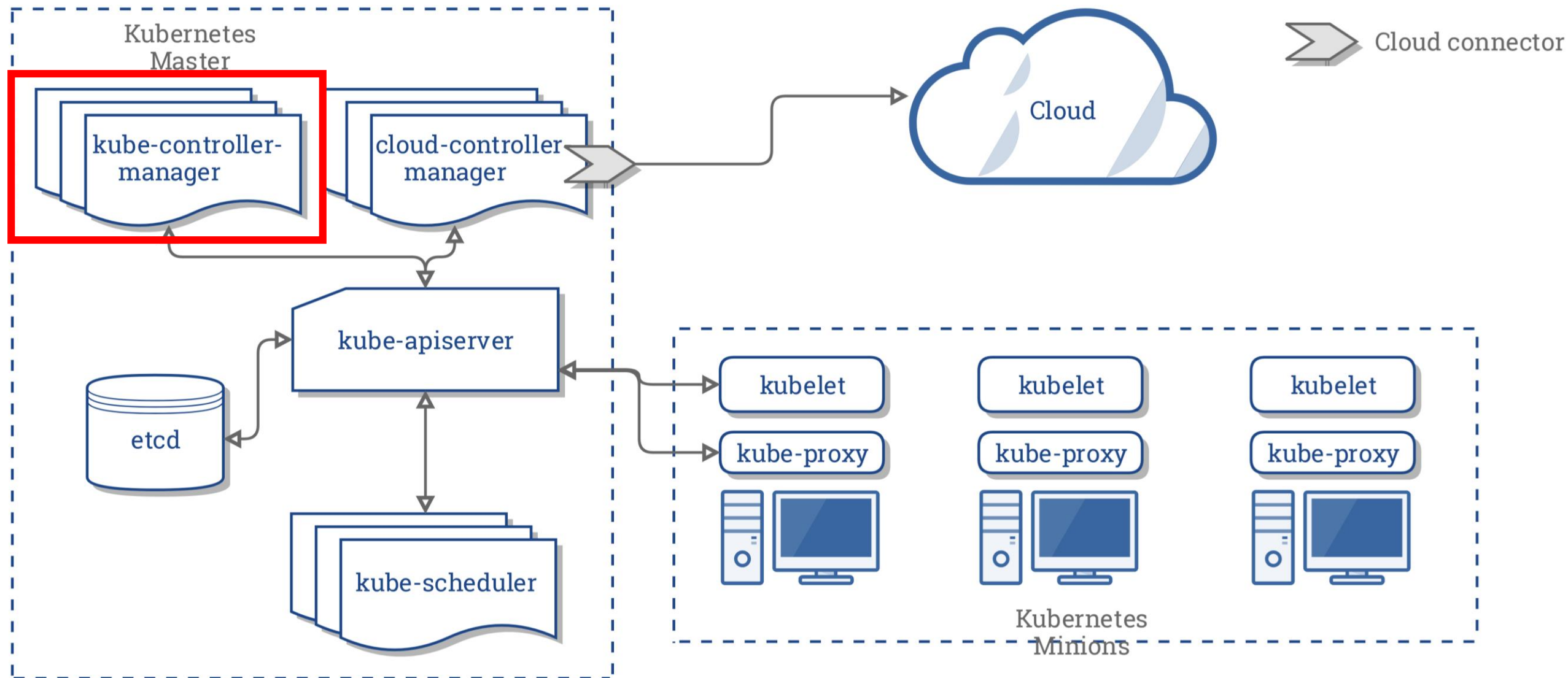


# Kubernetes architektúra





# Kubernetes architektúra



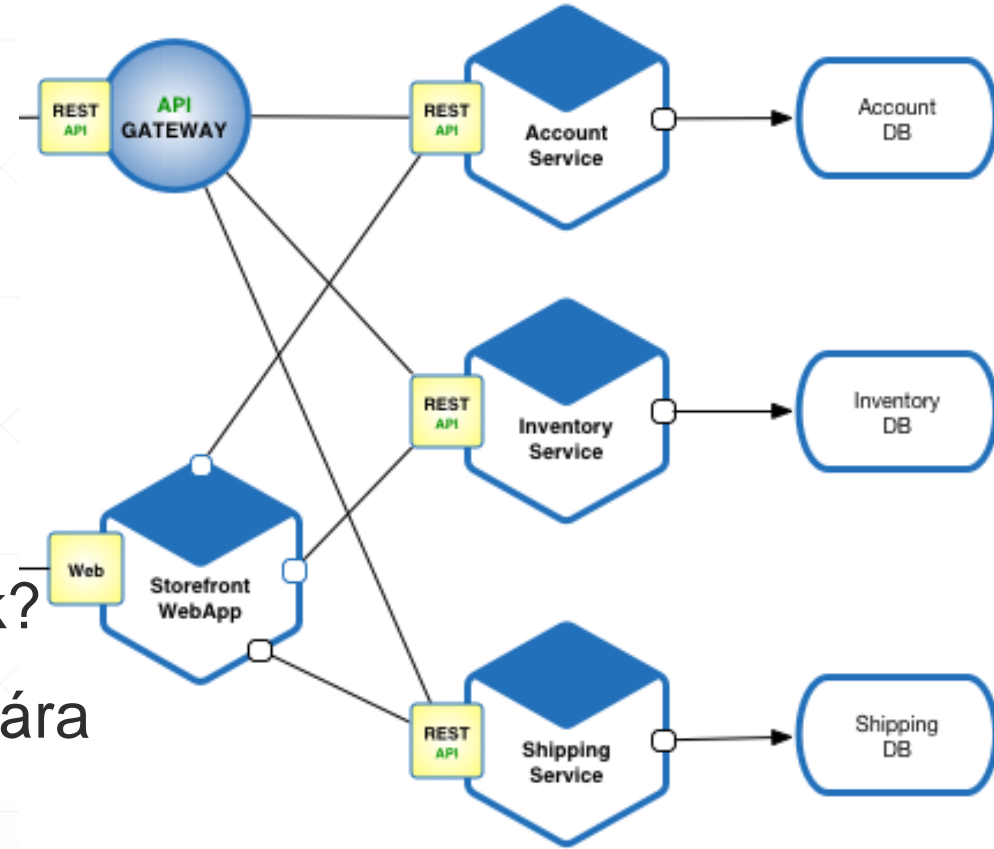


# K8s Service

- A Pod-ok életriklusa dinamikus (skálázódás)
    - nincs stabil IP címük
  - Frontend – Backend Pod-ok
    - hogyan tudják követni, melyik Pod hova tartozik?
-

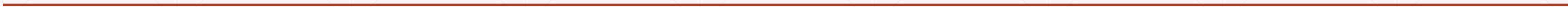
# K8s Service

- A Pod-ok életrajza dinamikusan (skálázódás)
  - nincs stabil IP címük
- Frontend – Backend Pod-ok
  - hogyan tudják követni, melyik Pod hova tartozik?
- **Service** – logikai absztrakció Pod-ok egy halmazára
  - stabil IP (cluster-en belül vagy kívül)
  - terhelés elosztás
  - szolgáltatás felderítés a Pod-ok számára (általában Label-ek segítségével)
- microservices



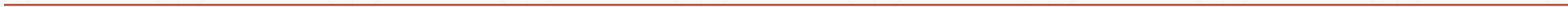
```
kubectl expose deployment/rest-hello --type="NodePort" --port 8080
```

**Köszönöm a figyelmet!**



# Gyakorlat: Kubernetes via Minikube

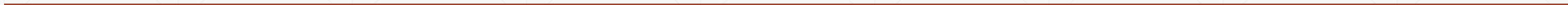
- HelloWorld Spring Boot alkalmazás



# Gyakorlat: Kubernetes via Minikube

- HelloWorld Spring Boot alkalmazás

```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```



# Gyakorlat: Kubernetes via Minikube

- HelloWorld Spring Boot alkalmazás

```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

```
~$ kubectl get deployments
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
rest-hello	1	1	1	1	17m

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
rest-hello-5c5cc4469-qws5z	1/1	Running	1	2h

---

# Gyakorlat: Kubernetes via Minikube

- HelloWorld Spring Boot alkalmazás

```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

```
kubectl expose deployment/rest-hello --type="NodePort" --port 8080
```

---



# Gyakorlat: Kubernetes via Minikube

- HelloWorld Spring Boot alkalmazás

```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

```
kubectl expose deployment/rest-hello --type="NodePort" --port 8080
```

```
kubectl scale deployments/rest-hello --replicas=4
```

---

# Gyakorlat: Kubernetes via Minikube

- HelloWorld Spring Boot alkalmazás

```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

```
kubectl expose deployment/rest-hello --type="NodePort" --port 8080
```

```
kubectl scale deployments/rest-hello --replicas=4
```

```
$ kubectl get deployments
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
rest-hello	4	4	4	3	2h

```
$ kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
rest-hello-5c5cc4469-mtfc	1/1	Running	0	44s	172.17.0.8	minikube
rest-hello-5c5cc4469-qws5z	1/1	Running	1	2h	172.17.0.5	minikube
rest-hello-5c5cc4469-w45nk	1/1	Running	0	44s	172.17.0.6	minikube
rest-hello-5c5cc4469-zl9q9	1/1	Running	0	44s	172.17.0.7	minikube

# Gyakorlat: Kubernetes via Minikube

- HelloWorld Spring Boot alkalmazás

```
kubectl run --image=rest-hello morova/rest-hello:latest --port=8080
```

```
kubectl expose deployment/rest-hello --type="NodePort" --port 8080
```

```
kubectl scale deployments/rest-hello --replicas=4
```

```
$ curl $(minikube ip):$NODE_PORT/greeting
{"id":1,"content":"Hello, World!"}
$ curl $(minikube ip):$NODE_PORT/greeting
{"id":2,"content":"Hello, World!"}
$ curl $(minikube ip):$NODE_PORT/greeting
{"id":1,"content":"Hello, World!"}
$ curl $(minikube ip):$NODE_PORT/greeting
{"id":3,"content":"Hello, World!"}
$ curl $(minikube ip):$NODE_PORT/greeting
{"id":1,"content":"Hello, World!"}
```