

Cloud based networks

Orchestrating the containers

Csaba Simon

Motivation – multi host

» Docker containers handled with docker commands

» On-host

» Networking is cumbersome

» docker0 bridge

» How to connect docker containers deployed on different hosts?

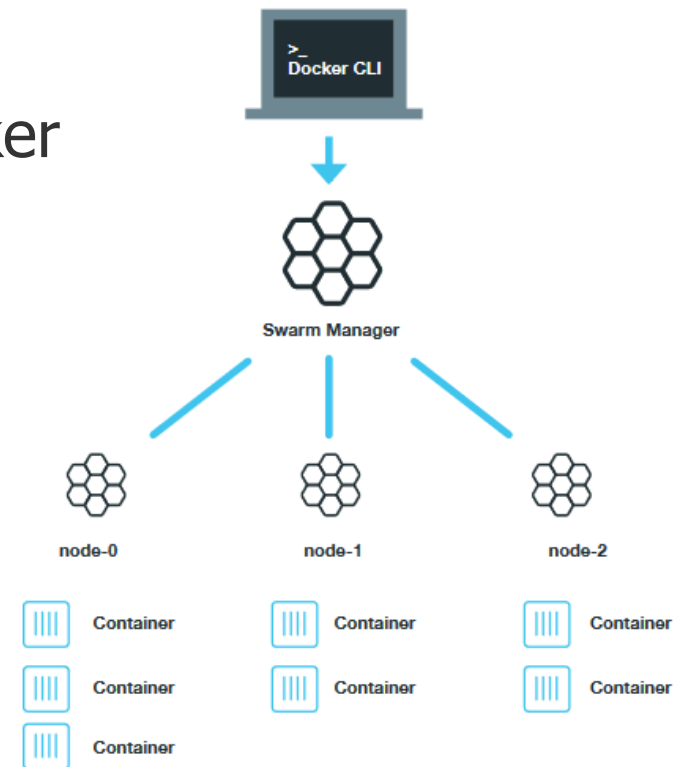
» Mult-hosting

» Third party solutions at the beginning
(e.g. serf - <https://www.serf.io/>)

» Later: **Docker Swarm** – multi-hosting in Docker

„It turns a pool of Docker hosts into a single, virtual Docker host“

» Not the same as Docker Swarm Mode (which appeared with v1.12)

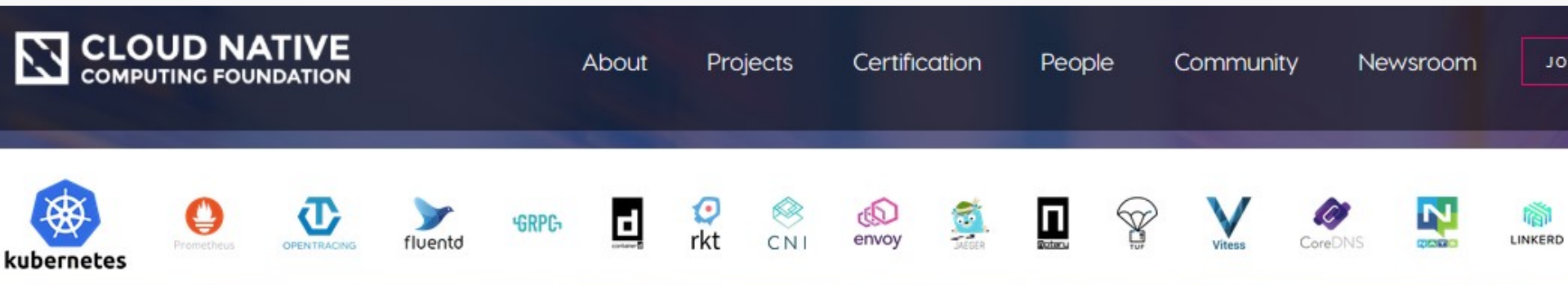


Motivation - orchestration

- » What is missing from a full Docker system?
 - » **Orchestration**
 - » Similar to the services of a cloud system
 - » Goal: automatized container deployment and management in multi-host environment (incl. scaling)
- » Solution no. 1: Docker in public clouds
 - » Amazon Web Services, Google Cloud, Microsoft Azure
- » Solution no. 2: Docker + OpenStack
 - » OpenStack Magnum
- » Solution no. 3: Docker based orchestration frameworks
 - » Apache Mesos (2010)
 - » Google Kubernetes (2014)
 - » Docker Swarm Mode (2016)

Cloud Native Computing Foundation

- » Container orchestration based microservices ecosystem
- » Note that rkt is the supported container technology

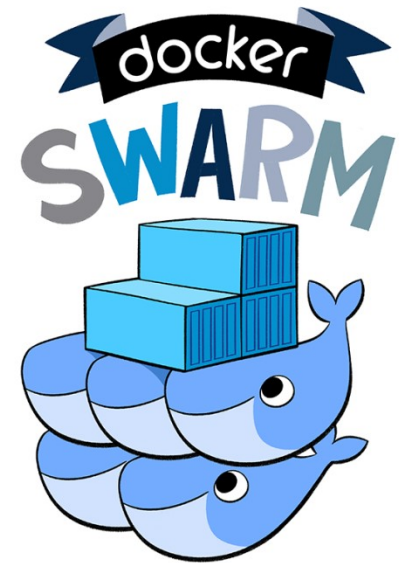


Sustaining and Integrating Open Source Technologies

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

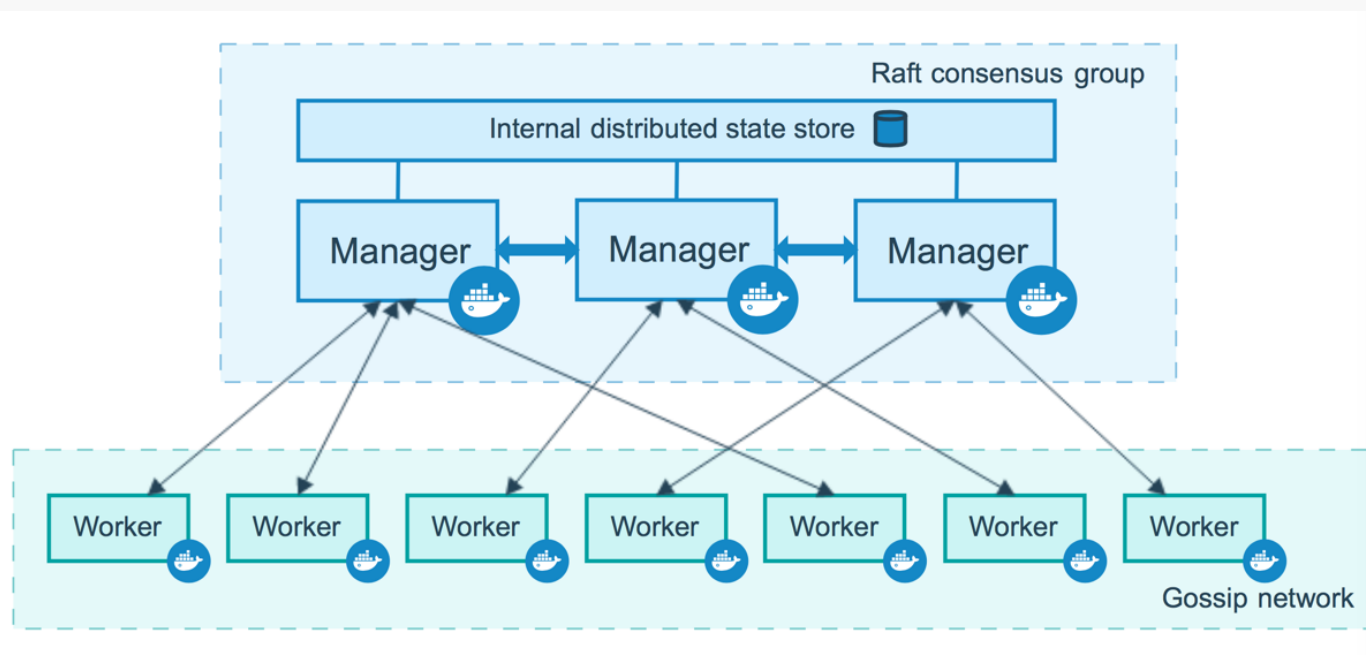
Docker Swarm Mode



- » **Swarm mode** = Docker engine running mode
- » The Docker engines organized in the same cluster
 - » One Docker engine = one node
 - » Swarm = this cluster above
 - » Goal: running **services** in this cluster
- » One physical machine may run multiple nodes
 - » In runtime environment typically Docker engine / phy machine
 - » Practically hosts running a Docker engine are grouped into a cluster
- » Service model: users reach a service
 - » **Service** = executes replicated tasks and defines the environment (network, resources, replication level and policy)
 - » Tasks run on multiple node handled as a single service
 - » **Task** = function (= docker container), which are handled by a single service
 - » Atomic resource unit, runs on a node

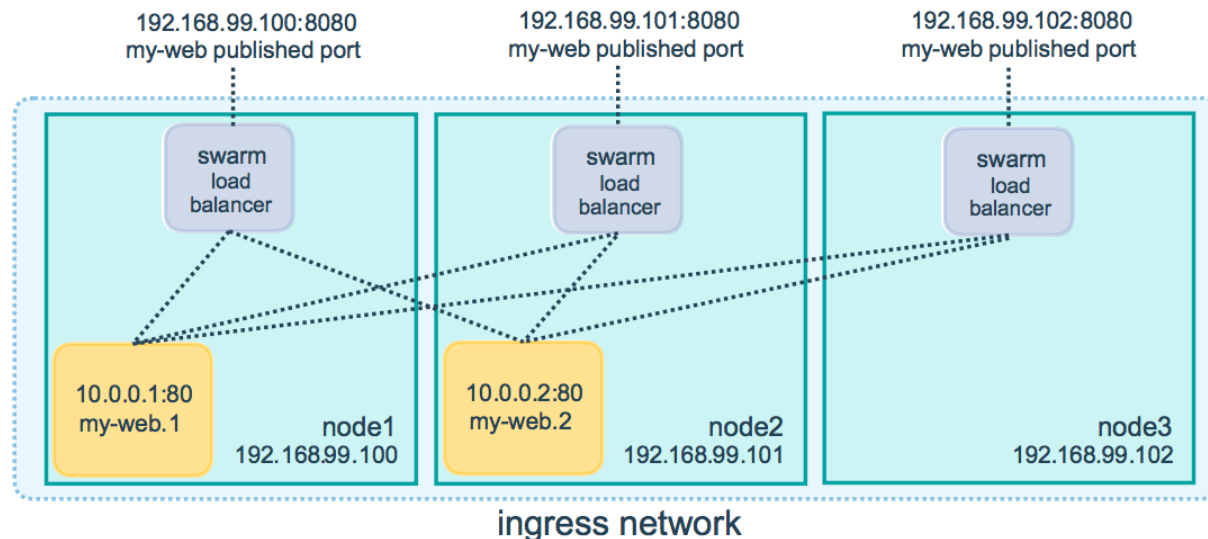
Swarm Mode architecture

- » Docker Swarm Mode nodes controlled by a **Manager**
 - » Role: cluster mgmt, offering an API, scheduling
 - » More Managers provide a distributed redundant operation (high availability)
- » **Worker** node = runs the tasks (Manager can be a worker, too)
 - » Worker node can be promoted to Manager (and vice-versa)
 - » Worker nodes join a mesh network



Swarm mode networking

- » Assign ports to services
 - » Handling requests arriving to the Swarm (ingress nw)
 - » The nodes must be the members of a *Swarm mode routing mesh*
- » Each node must run a load balancer module
 - » Part of the Swarm mode routing mesh
 - » Forwards the requests to a proper active container
 - » Even if that container runs on a different host
 - » Even if that on the node/host that recieved the request does not run such a container

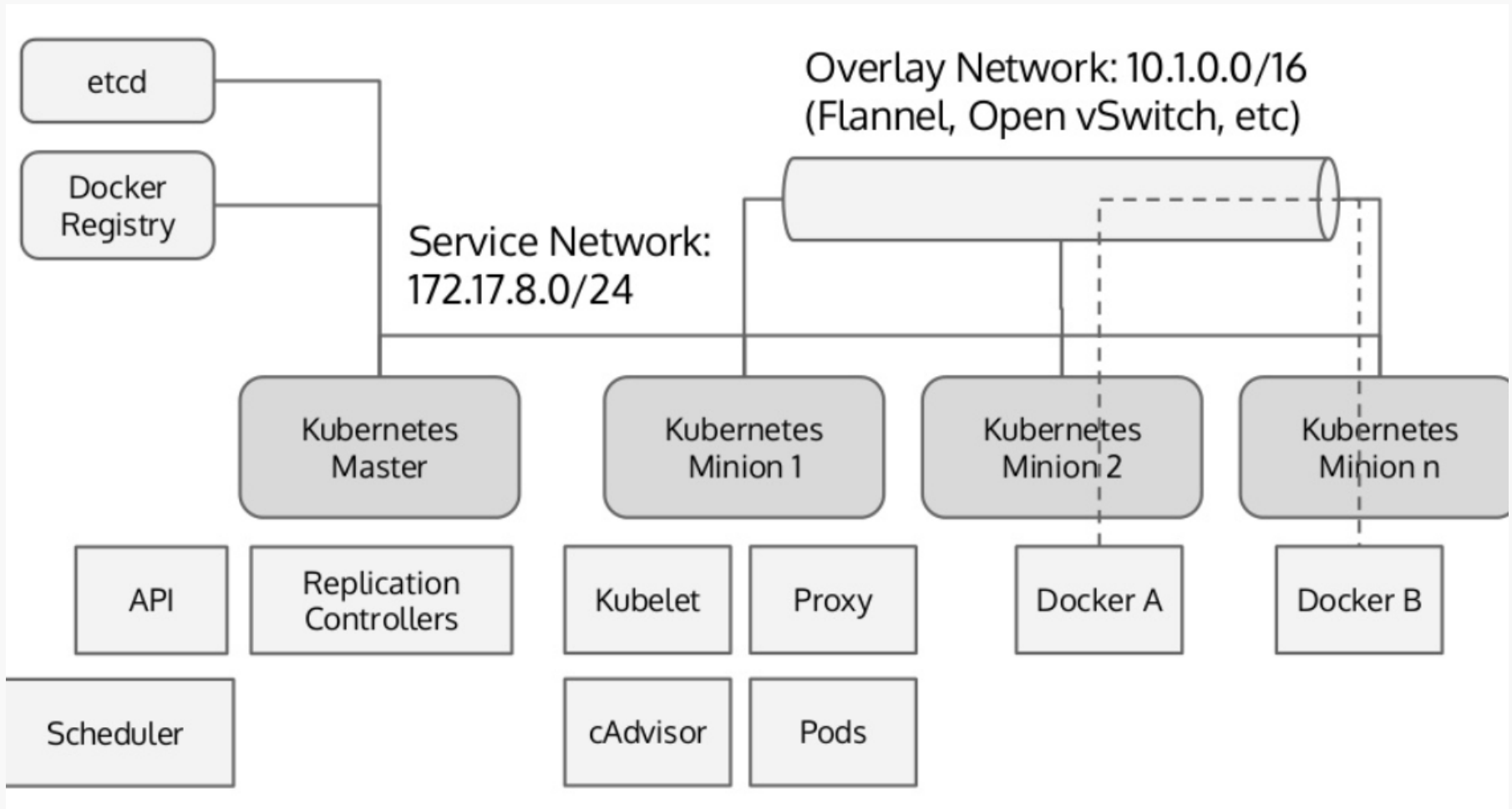


KUBERNETES

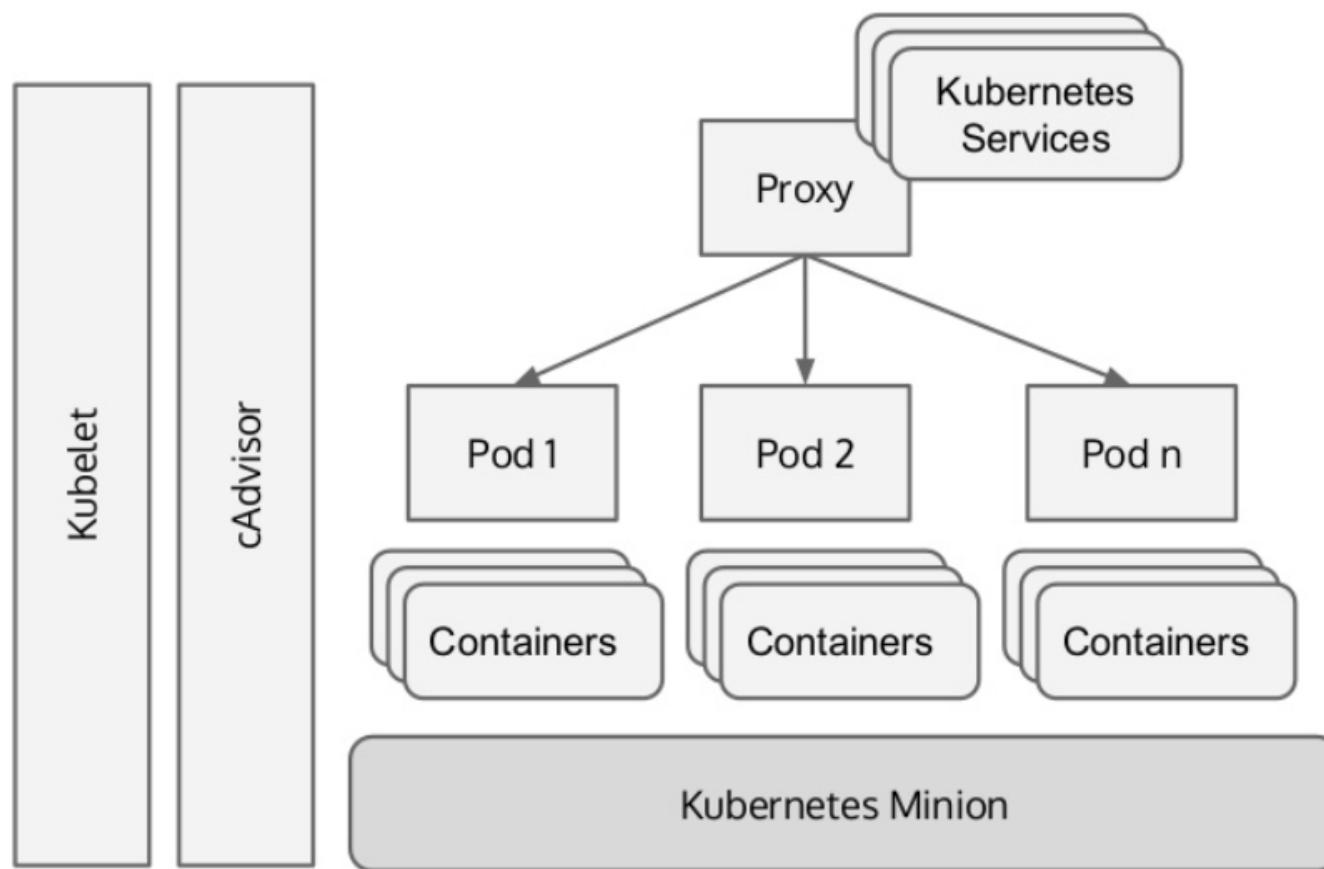
Kubernetes – main components

- **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

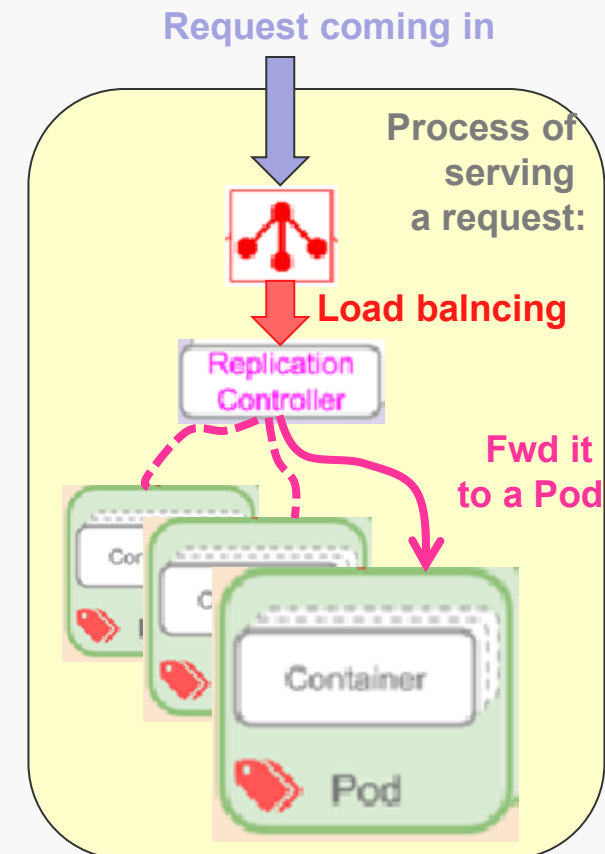
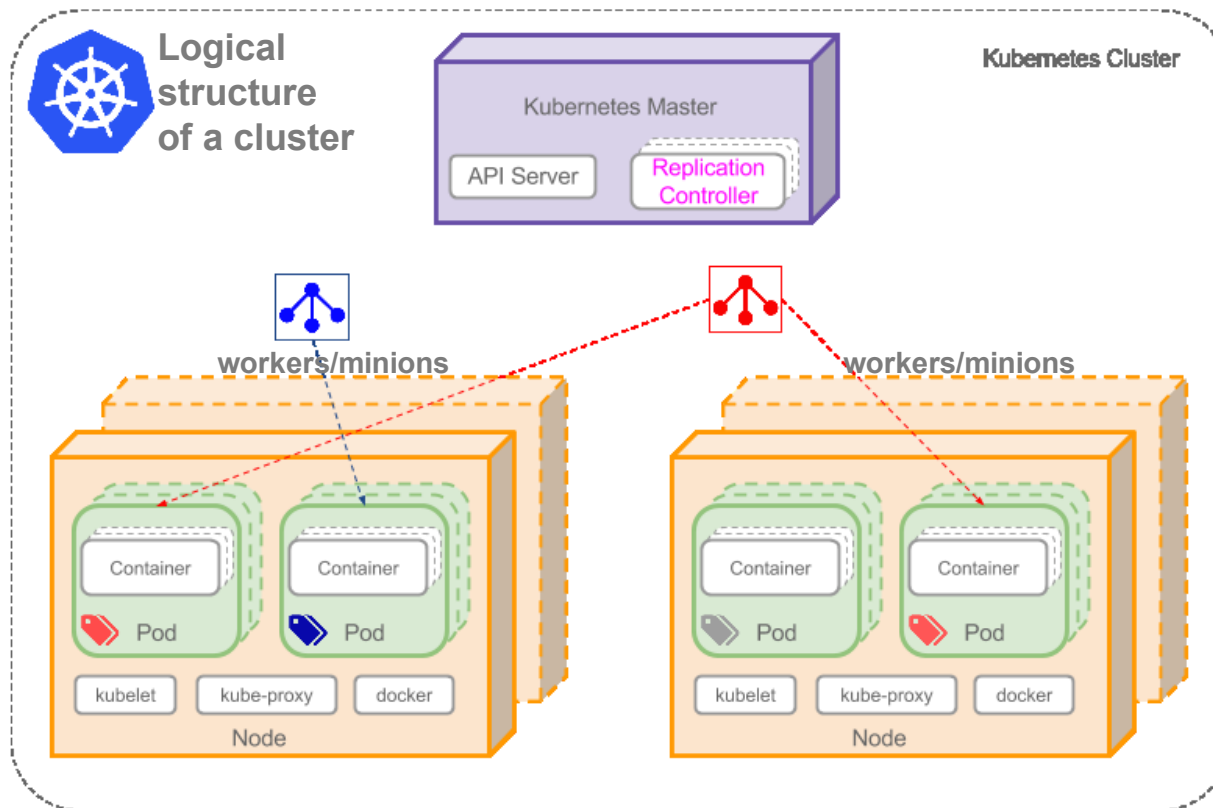


Worker node = minion



Logical structure of a Kubernetes cluster

- » Control by the master
- » Service offers access to users
 - » Handled by a load balancer (the Replication Controller)
 - » The request is answered by one Pod



Kubernetes network

- » At Pod level every container is in the same namespace
 - » Pro: can reach each other via localhost
 - » Consequence: mind the port assignment within a Pod (2 containers cannot use the same port)
- » Hosts must communicate with containers without NATs
- » Typical solutions:
 - » Flannel: own solution, flat overlay
 - » OVS: Open VSwitch – generic solution, widely used in the industry
 - » Lots of alternatives:
<https://kubernetes.io/docs/concepts/cluster-administration/networking/#how-to-achieve-this>

Demo

- » Kubernetes on-line demo
 - » Starting a Pod, handling in cli

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