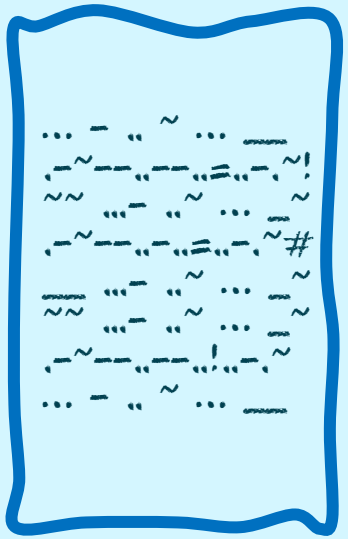
HACK YOUR JOB

Introduction a Kubernetes

Instructor: Michael Bright

https://linkedin.com/in/mjbright @mjbright @mjbright

Agenda



1. Round Table
2. Introduction
3. Containers & Orchestration
4. Kubernetes Concepts: Pods
5. Kubernetes Concepts: Controllers (Deployments)
6. Kubernetes Concepts: Services

@mjbright

Tour de Table

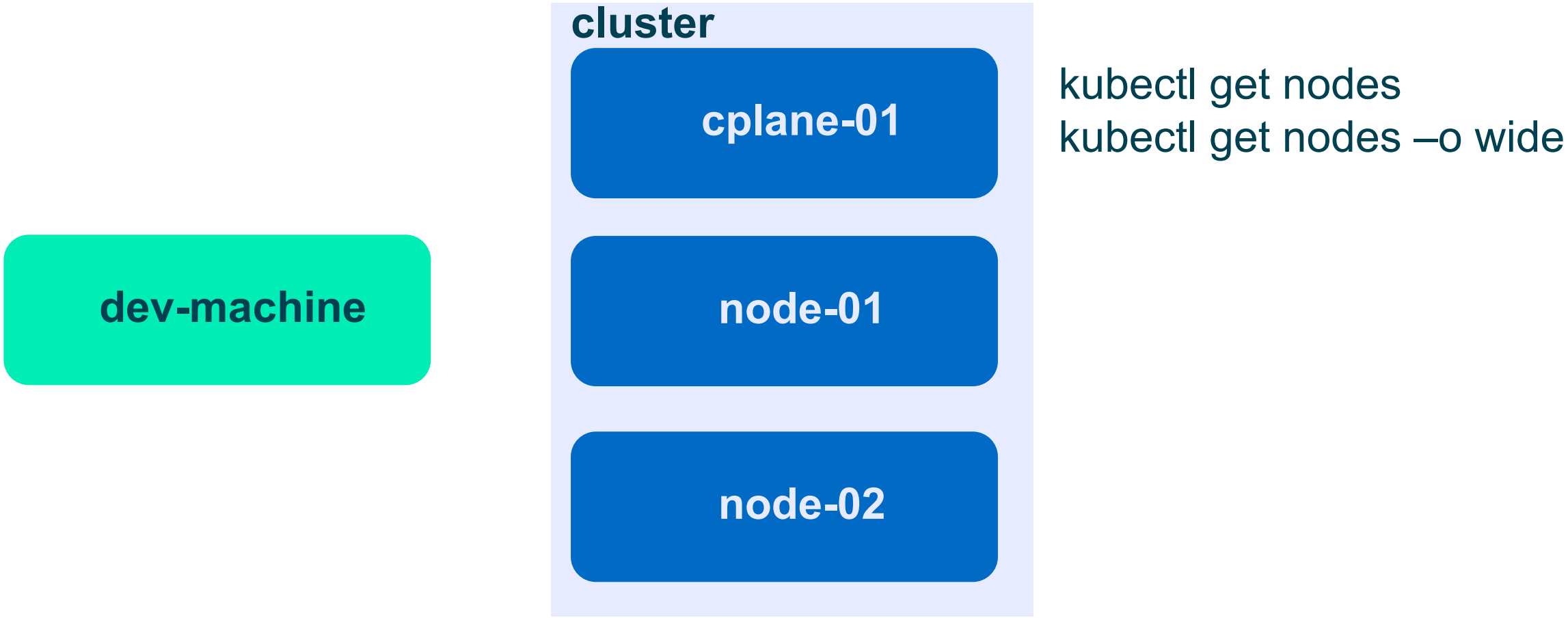
Experience / connaissance des conteneurs ?
e.g. Docker

1 – 5

Experience / connaissance de Kubernetes ?

1 – 5

Lab Environment (1 month)



kubectl get nodes
kubectl get nodes –o wide

I  questions

Introduction:
Why Kubernetes ?

Containers & Orchestration

@mjbright

Containers: Isolated Processes

By default on a Linux - or Windows – machine, a running process, e.g. a web server can "see"

- the files present on the machine (if access permissions)
- other running processes
- network interfaces of the machine

Containers: Isolated Processes

By default on a Linux - or Windows – machine, a running process,
e.g. a web server can "see"

- the files present on the machine (if access permissions)
- other running processes
- network interfaces of the machine

Containers are processes that run in an "isolated" environment that may see the "global" resources mentioned above or may be limited to

- only files belonging to the container (coming from the container image)
- only processes running inside the container
- only the network interfaces of the container

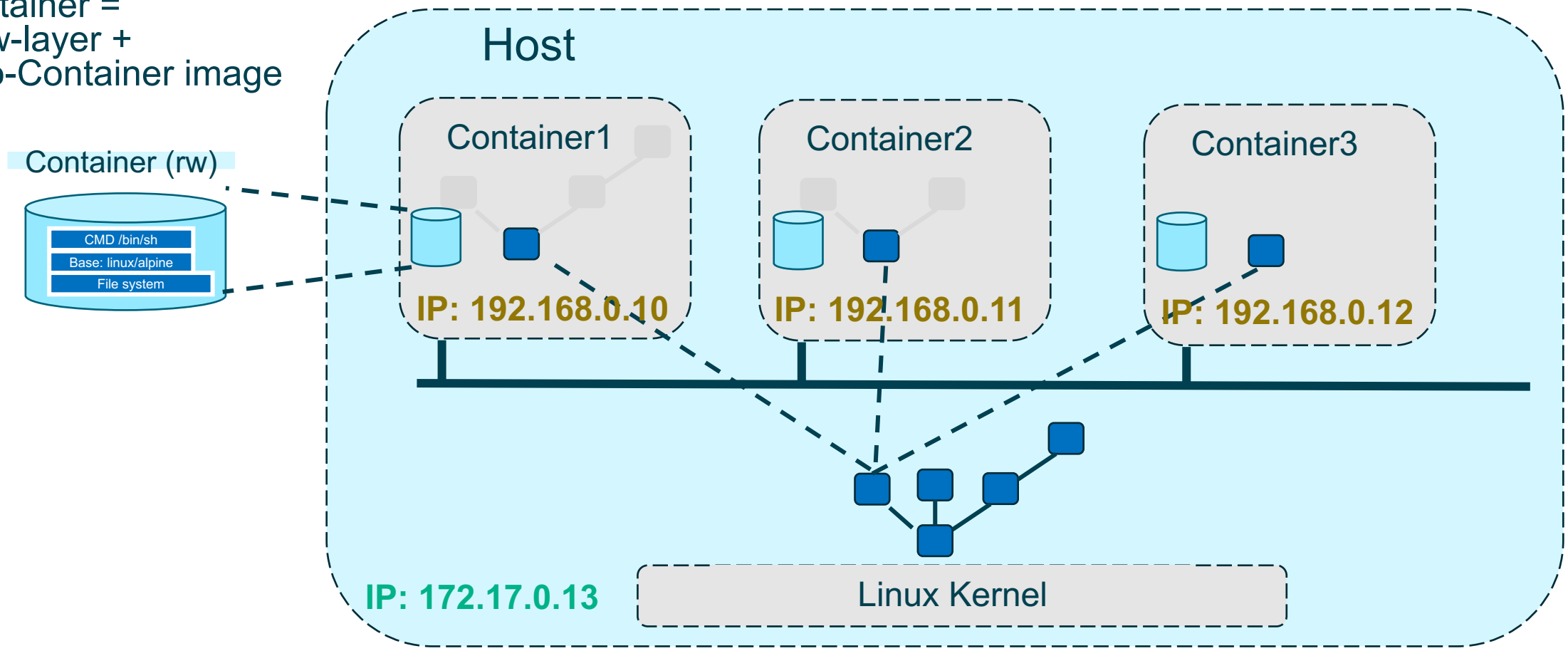
Containers: Isolated Processes

Note: A machine may be a "physical machine" e.g. a PC, or a "virtual machine".

We may also use the term "Node" or "host" as the machine hosts a Container Engine and it's processes

Container Concepts

Container =
rw-layer +
ro-Container image



A container is a group of 1 or more processes isolated within kernel namespaces (process, hostname, network, …)

Container Orchestration

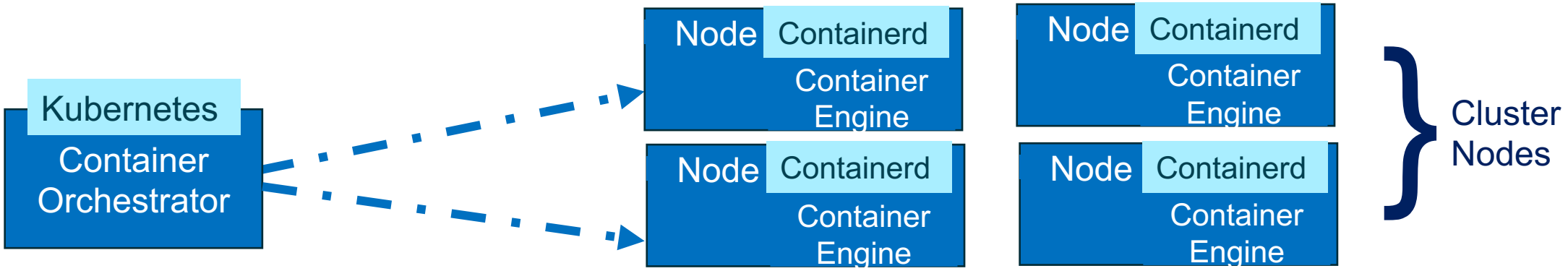
Running Containers across a Cluster of Machines

@mjbright

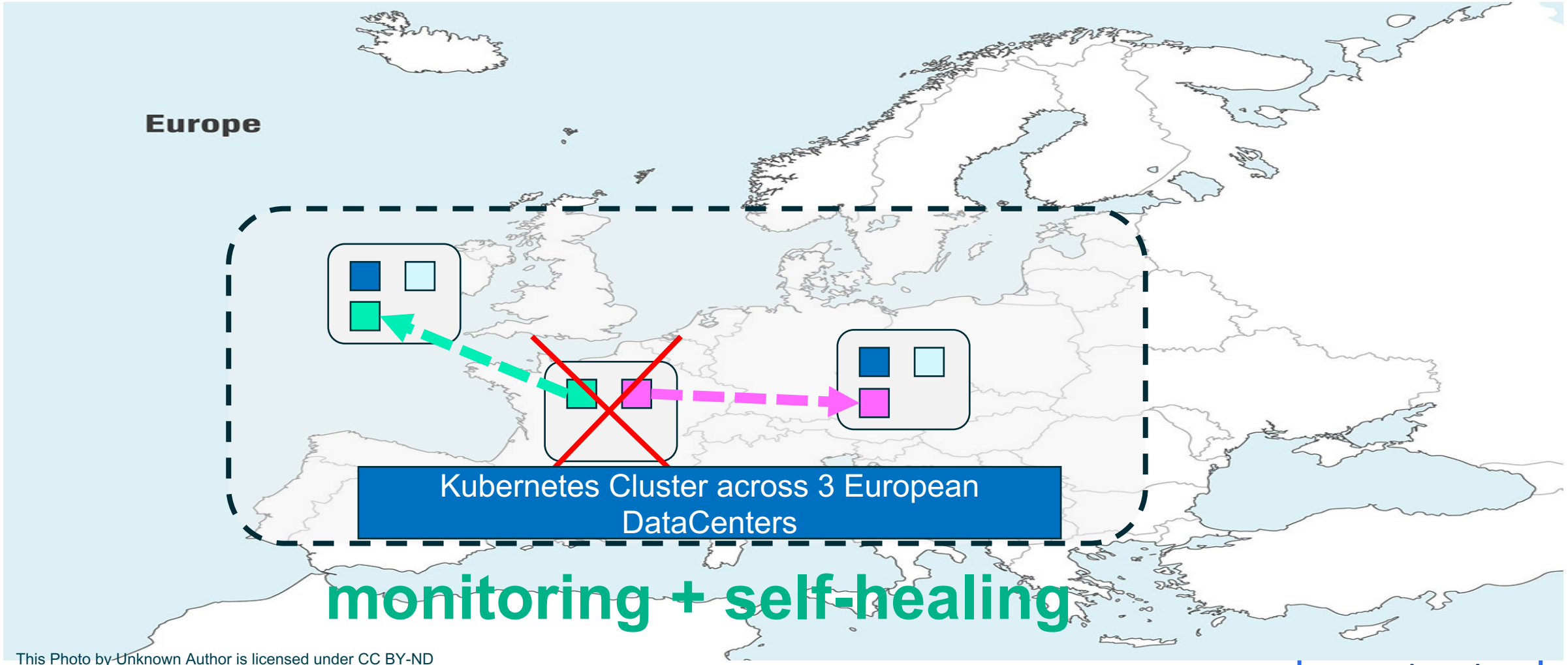
Container Engine (*) vs. Orchestrator

A Container Engine manages Containers running locally on a Node.

A Container Orchestrator such as Kubernetes manages a Cluster of Nodes each containing a Container Engine



Kubernetes Concepts: Why Orchestration?



Kubernetes

Kubernetes allows to deploy applications reliably at scale ...

Kubernetes

Kubernetes allows to deploy applications reliably at scale ...

... provided that the applications are designed accordingly ...

Kubernetes Core Concepts

Pods, Controllers, Services
Labels & Selectors

@mjbright

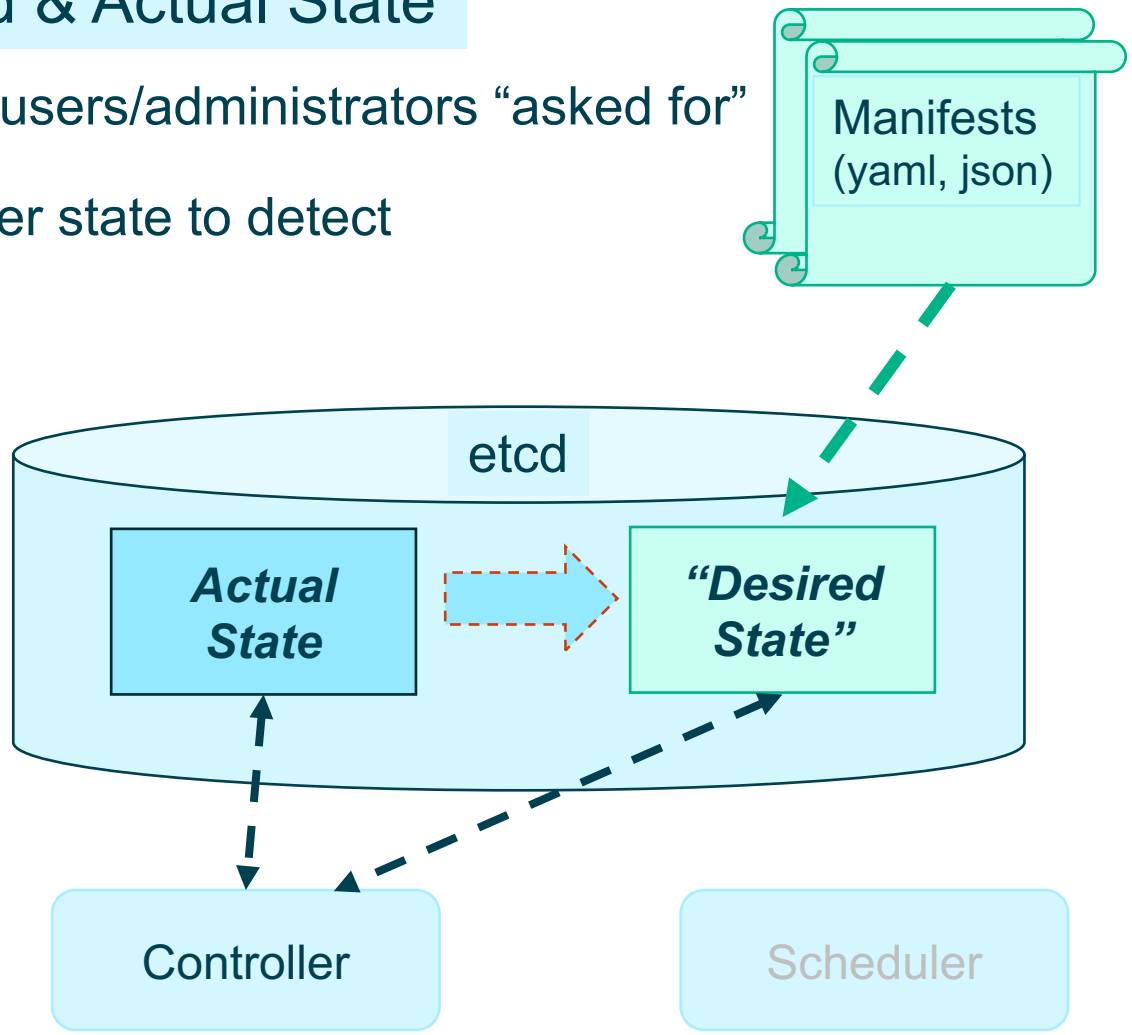
Core Concepts: "*Desired State*"

Reconciliation of Desired & Actual State

"*Desired State*": what users/administrators "asked for"

Controller loops in "Controller Manager" monitor cluster state to detect differences between "*Actual State*" & "*Desired State*".

Controller acts on any discrepancies



Core Concepts: "*Loose Coupling*"

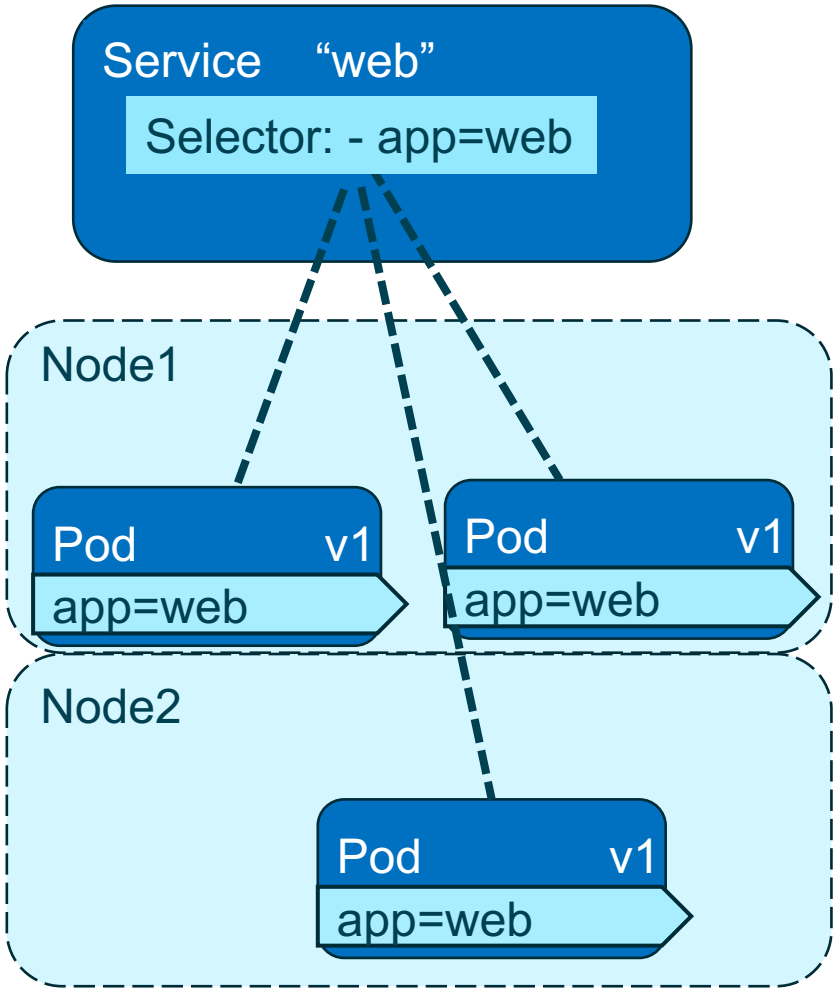
Resources are "loosely coupled" through labels & selectors

Kubernetes has a flat hierarchy of resources, with no direct linkage between them.

We assign **labels** (key=value) to resources, such as Pods, to identify groups of similar Pods.

Resources can have **selectors** which allow them to be associated with other resources, e.g.

- A **Service** which load-balances traffic to a group of Pods
- A **Deployment** which. manages a group of Pods

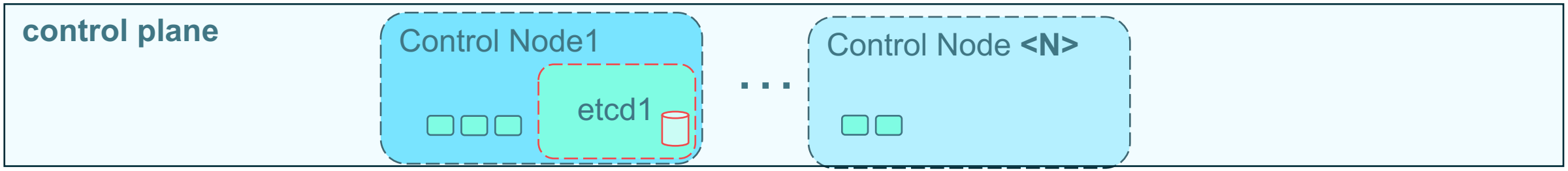


Architecture - Node Types

Kubernetes uses 2 types of nodes - plus etcd

Control Nodes provide the cluster control plane

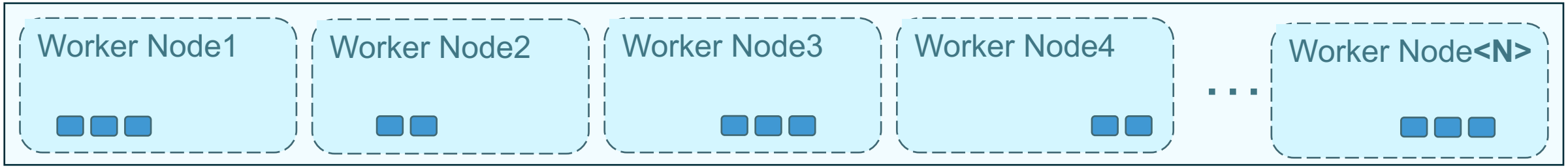
May also be called *Master* or *Head* Nodes

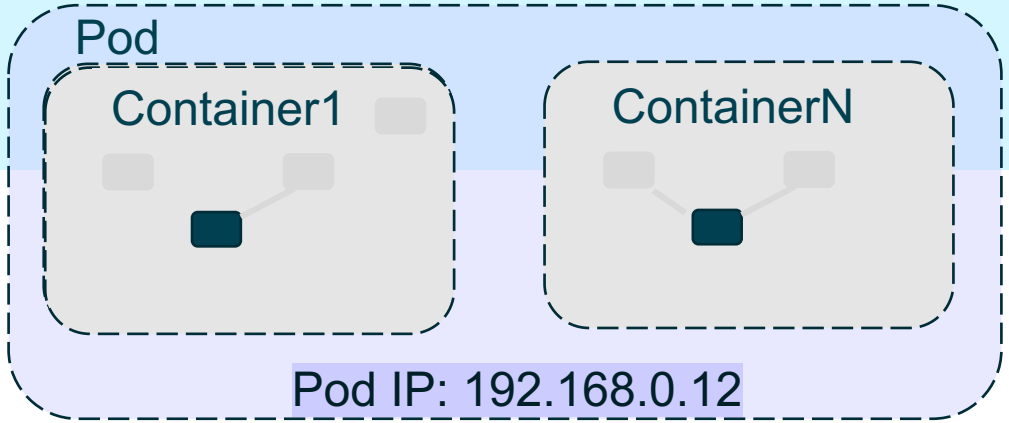


Worker Nodes

May also be called *Minion* Nodes or just plain *Nodes*

- On which application tasks (Pods) are scheduled





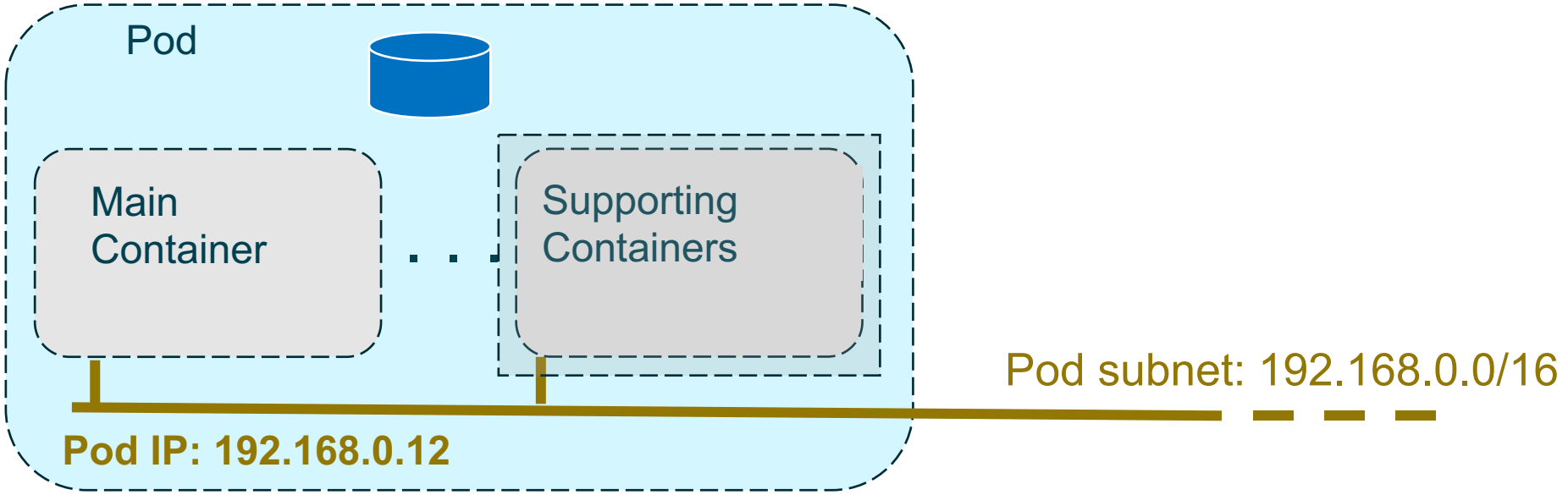
Core Concepts - Pods

The basic schedulable unit of execution in Kubernetes is a Pod

Core Concepts: Pods

Containers in a Pod can share some **kernel namespaces**: PID, Network

- Always share the IP address & localhost
- Can optionally see each others processes [if *shareProcessNamespace* set]



https://kubernetes.io/docs/tasks/configure-pod-container/share-process-namespace/

Core Concepts: Pod Creation

Imperative commands

A Single Container Pod can be created with the *"imperative"* (do this) kubectl run command,

```
kubectl run <name of pod>  --image=<image>
```

For example:

```
kubectl run web --image=mjbright/k8s-demo:1
```

Core Concepts: Pod Creation

Declarative commands

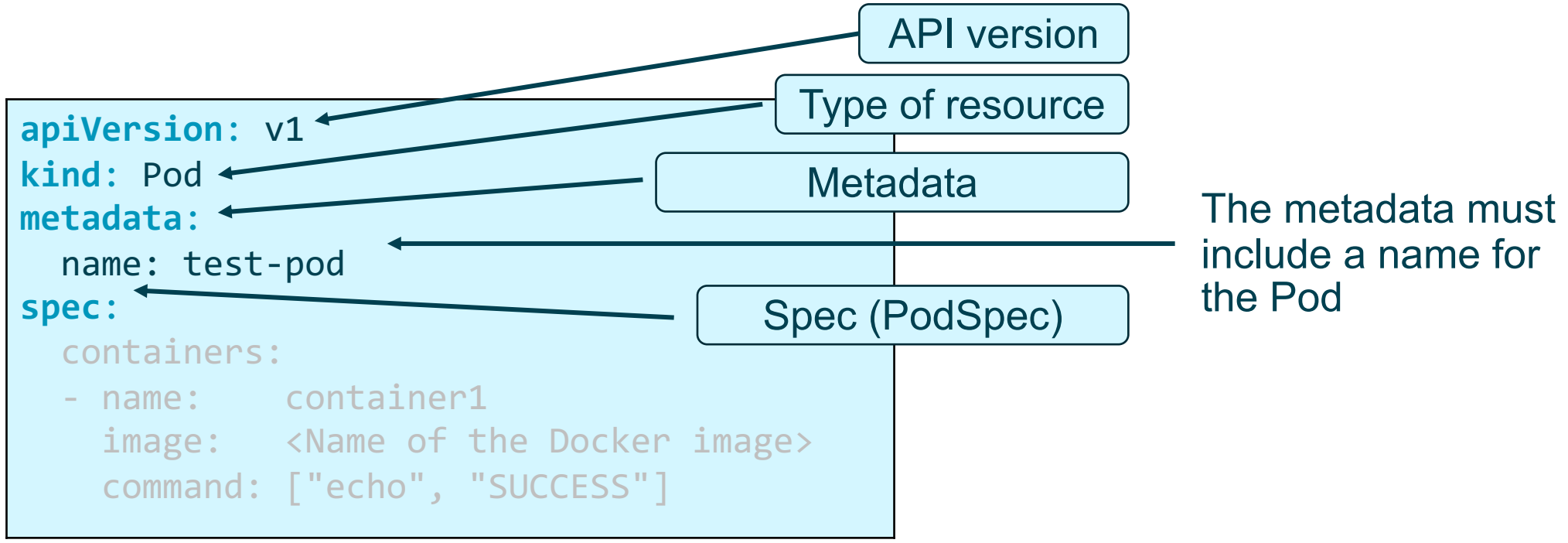
A Single Container Pod can also be created with the *“declarative”* kubectl apply command

```
kubectl apply -f pod.yaml
```

Core Concepts: Pods

Example of a Kubernetes yaml manifest

The manifest file has at least these 4 top level fields.



The metadata must include a name for the Pod

Core Concepts: Pods

Example of a Kubernetes yaml manifest

The ‘spec’ section has a field to define the image name for each container for this Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - name:    container1
    image:   <Name of the Docker image>
    command: ["echo", "SUCCESS"]
```

Container name

Container Image to use

Command & arguments to run in the container
(overrides default baked into container image)

Core Concepts: Controllers

Kubernetes Controllers

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: web
  name: web
spec:
  replicas: 10
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - image: mjbright/k8s-demo:1
        name: k8s-demo
```

No. of Pods

Pod selector

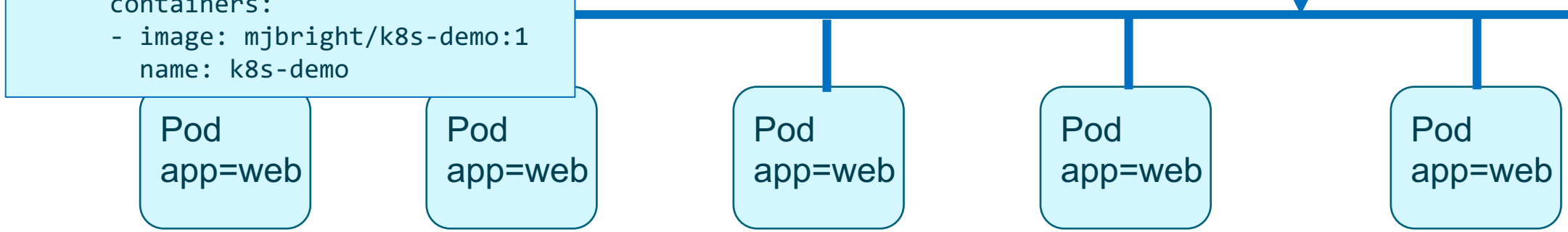
Pod labels

Controllers:

Deployment app=web
Selector: app=web

ReplicaSet app=web
Selector: app=web

Manage Pods Lifecycle



Core Concepts: Services

Kubernetes Services

Services: <web:80>

Constant Service endpoint

Service “web” app=web
Selector: app=web

Load balances traffic

Pod selector

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: web
  name: web
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: web
```

Pod app=web

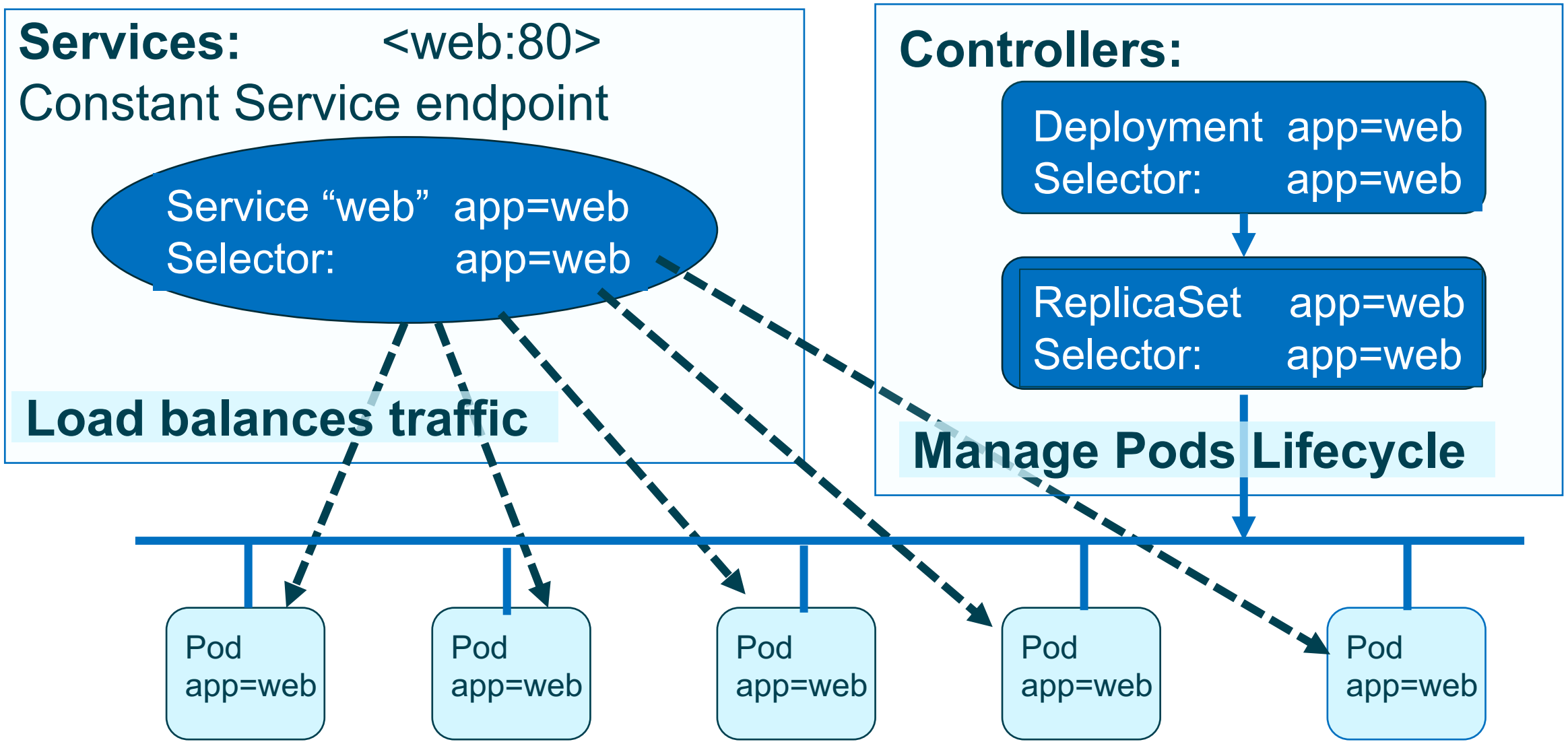
Pod app=web

Pod app=web

Pod app=web

Pod app=web

Kubernetes: Services vs. Controllers



Introduction a Kubernetes

Le Programme

https://formation.hackyourjob.com/catalogue/introduction-kubernetes.html

@mjbright

Jour 1

Introduction aux Conteneurs, l'orchestration de conteneurs

- Reprise de principes de Conteneurs
- Pourquoi les Pods - comparaison avec les conteneurs ?
- Lancer et Interagir avec un Pod (ports, exec, logs)

Principes et Concepts de Kubernetes

- Résilience à l'échelle “à la Kubernetes”
- L’architecture
- Le Network model
- Namespaces, Labels
- Deployments, Services

Jour 2

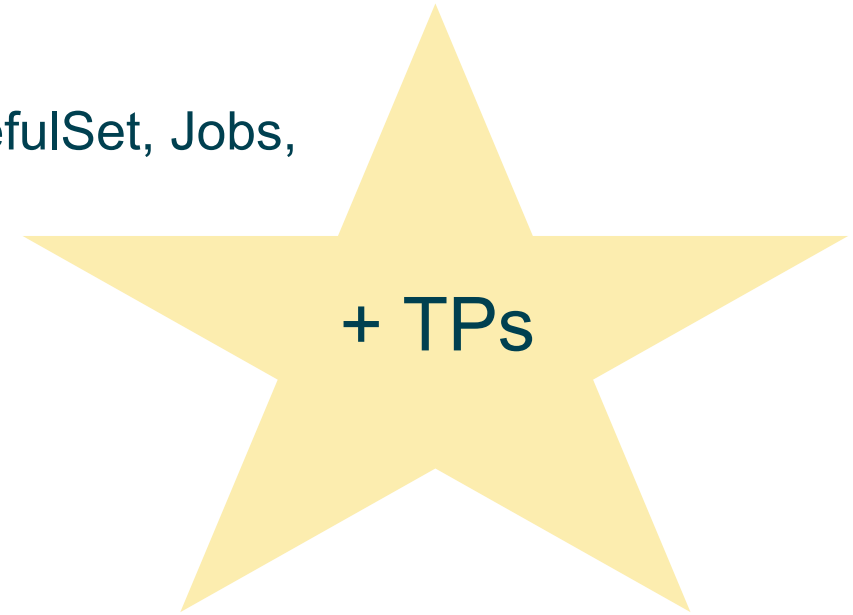
Travailler avec des Pods
- Kubectl run, YAML manifests, dry-run
- Multi-container Pods - design patterns, Init containers

Workload Controllers
- Pourquoi des Workload Controllers ?
- Deployments, ReplicaSets, DaemonSet, StatefulSet, Jobs, CronJobs

Stockage
- Les volumes, ConfigMaps & Secrets

Exposer des applications
- Services, Ingress Controller

Ecosystème de Kubernetes
- Le projet, Les distributions, Les outils
- Helm, Operators



https://linkedin.com/in/mjbright @mjbright @mjbright