Universidad Autónoma de Madrid | excelencia UAM CSIC

Escuela Politécnica Superior

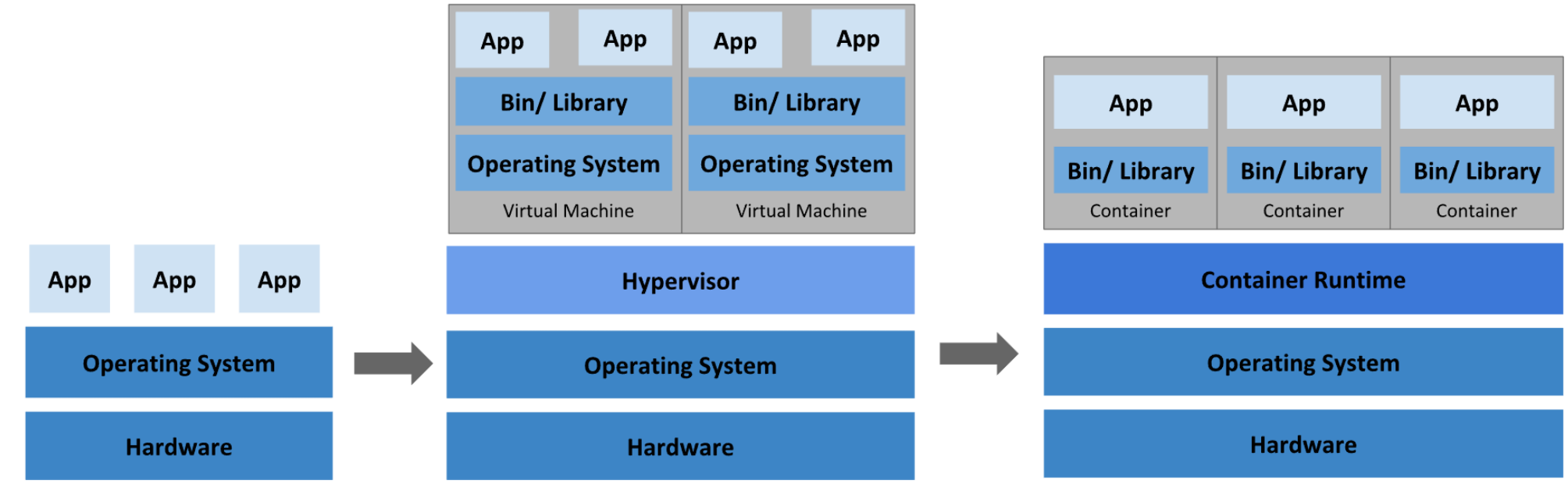
Lab 3 Part 2

A crash course on  kubernetes

Introduction

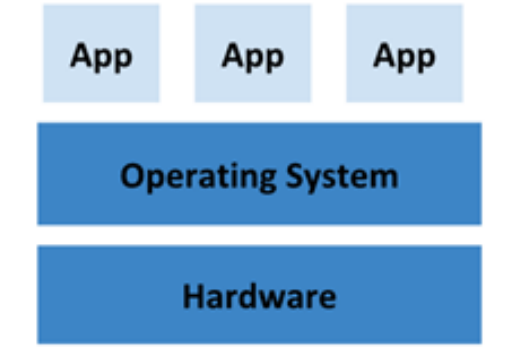
What is Kubernetes?

"Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications."



Source: Kubernetes.io

Bare metal vs VM vs Container

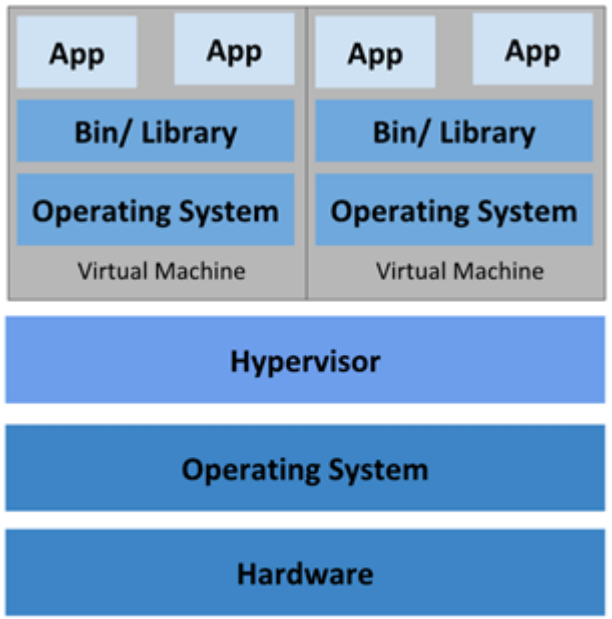


Traditional Deployment

A traditional deployment runs applications on top of the OS, also known as running the applications on bare metal.

- Great efficiency and usage of resources
- Poor portability
- Resource sharing with no policies
  - Now, systemd allows you to put constraints on services
- Library sharing and incompatibilities
- Security issues

Bare metal vs VM vs Container

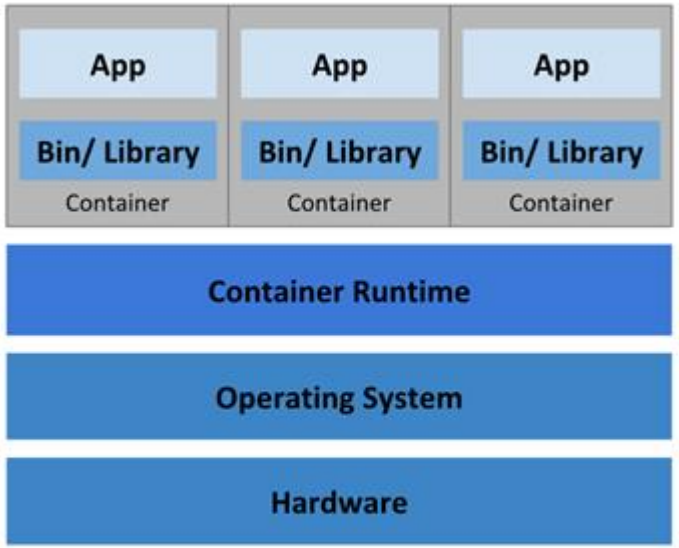


Virtualized Deployment

A virtualized deployment runs group of applications on several VMs. All VMs are managed by a program (hypervisor)

- Too much overhead and performance loss
- Highly portable
- Hypervisors configure resource restrictions
- No library sharing. Each VM gets its one environment.
- Less security concerns

Bare metal vs VM vs Container



Container Deployment

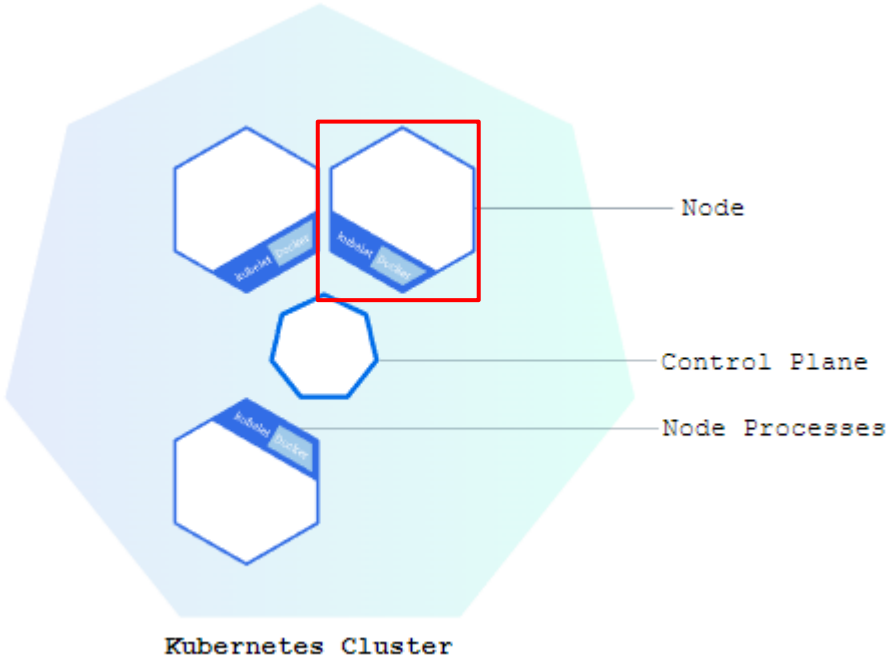
A containerized deployment runs each application on a container. All containers are managed by a program (container runtime)

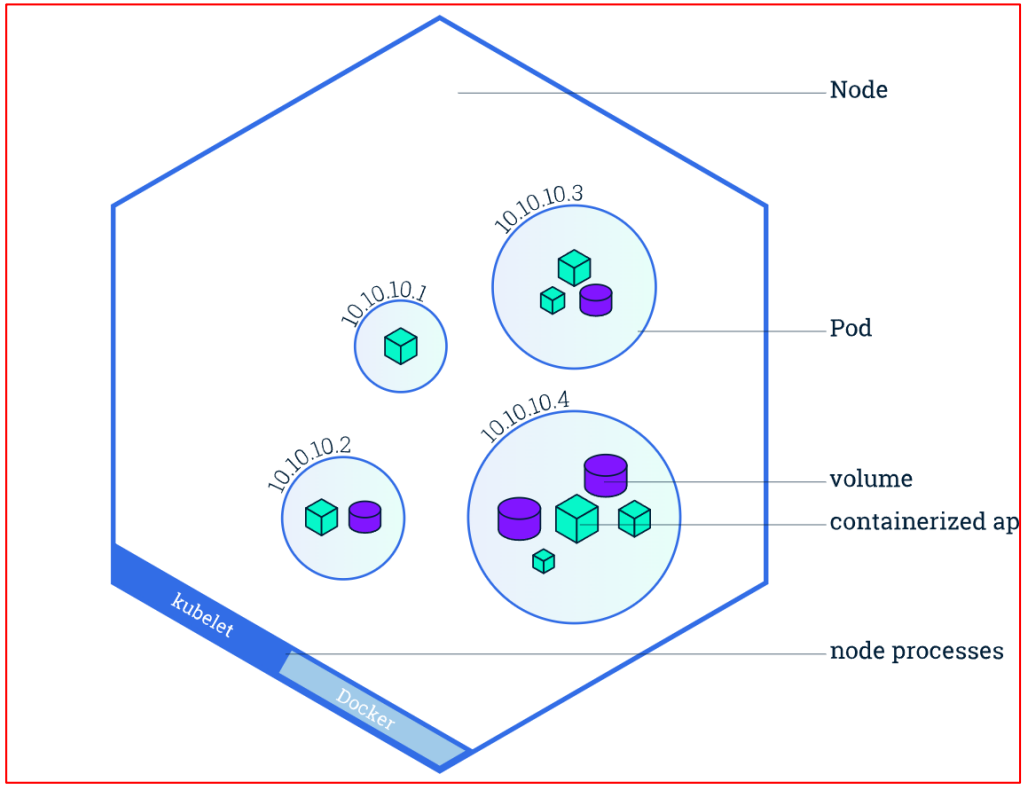
- Less overhead
- Highly portable
- Container runtime configure resource restrictions
- No library sharing. Each container gets its one environment.
- Less security concerns but…

Careful: One container only runs one process, no services/systemd inside.

What is Kubernetes cluster?

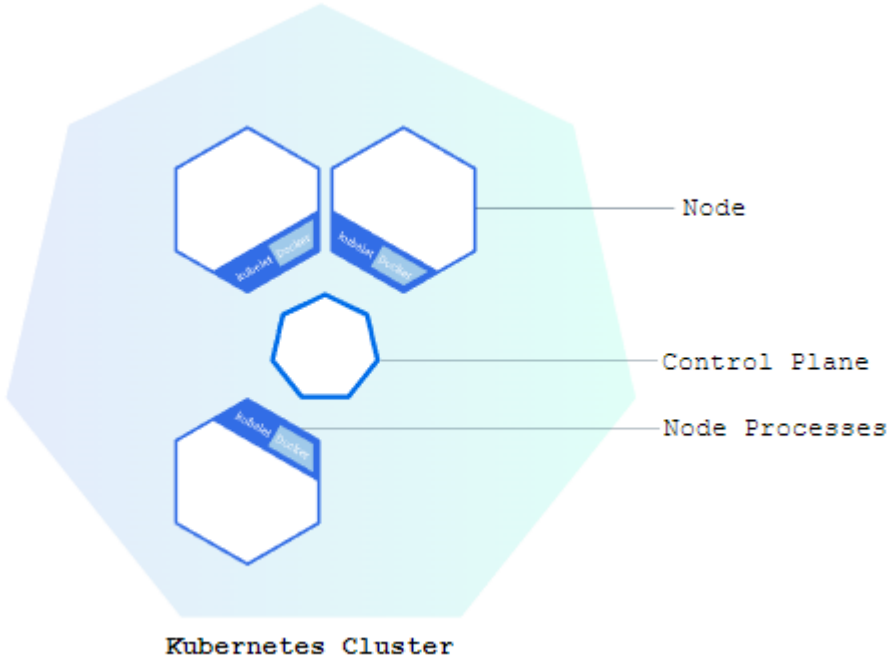
Let's explore the components of a k8s cluster





Source: Kubernetes.io

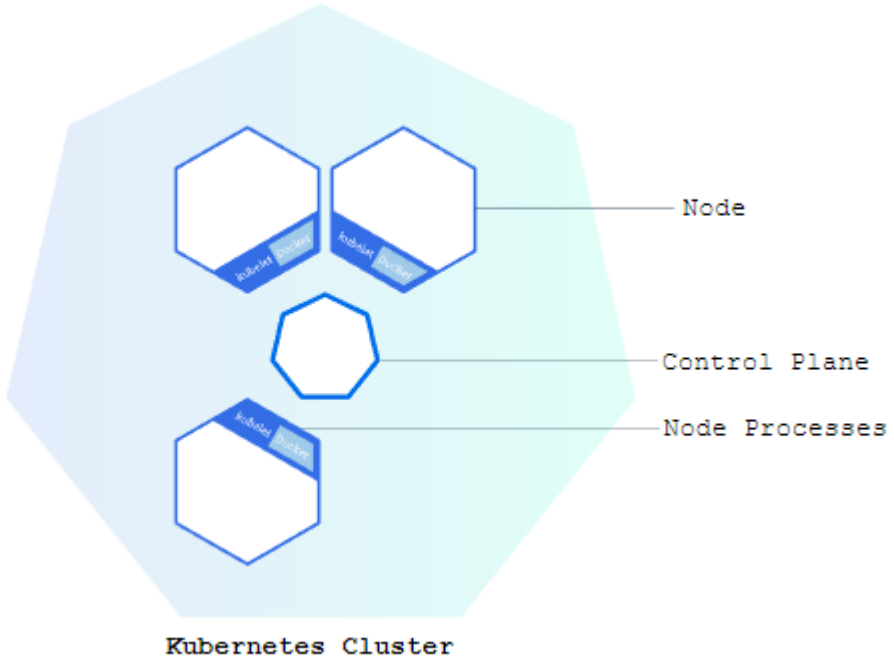
What is the control plane?



A k8s cluster is composed of two main components:

- Control plane:
  - Global services of the cluster.
  - Components:
    - etcd: HA key-value store
    - kube-scheduler: scheduling pods to nodes
    - kube-controller-manager: runs several controller processes such as node controller (check node status) or service controller (endpoint management)
    - kube-apiserver: frontend API to the users

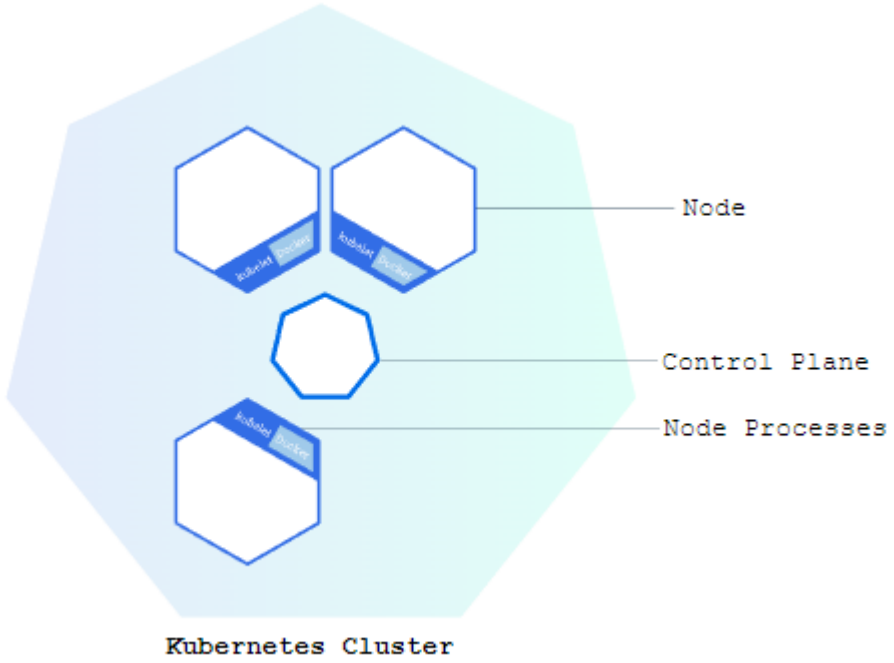
What is a node?



A k8s cluster is composed of two main components:

- Node:
  - Workers. They run Pods.
  - They might be physical hosts or VMs.
  - Some essential components:
    - kubelet: manage containers in Pods
    - kube-proxy: endpoints and forwarding for services
    - Container runtime: software that implements the Container Runtime Interface (CRI) to start/stop/monitor containers

But there is more…

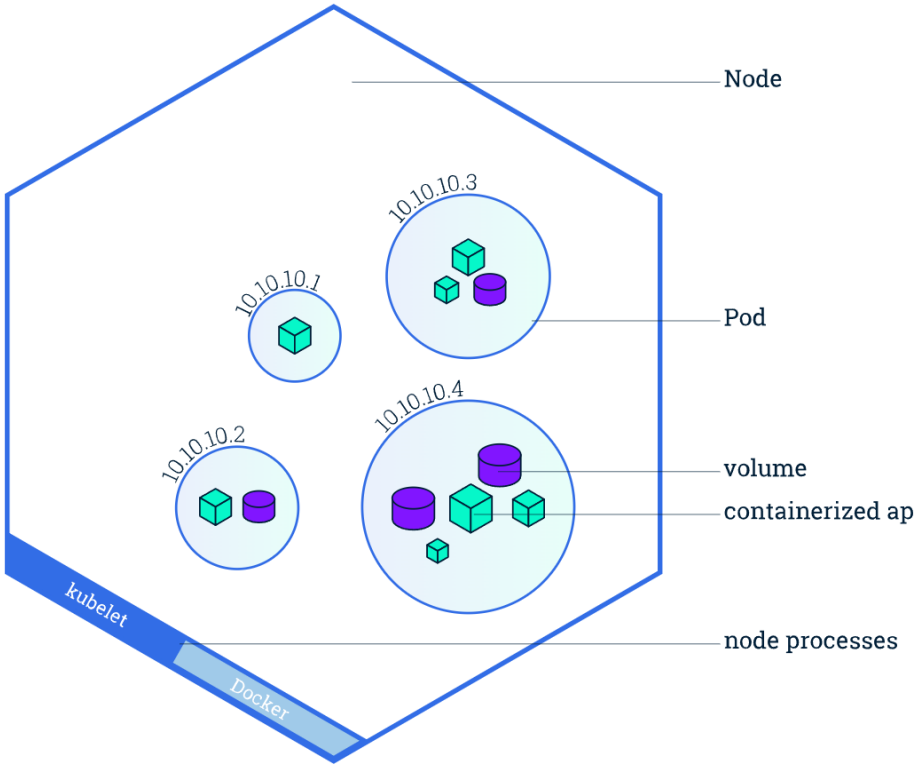


A k8s cluster also needs:

- Control (and data) networks.
  - We want nodes to connect to the control plane to be managed.
  - We may need exchange of data among nodes.
- Addons: DNS (automatic service name resolution), dashboard (cool web UI), resource monitoring …

How do we run apps?

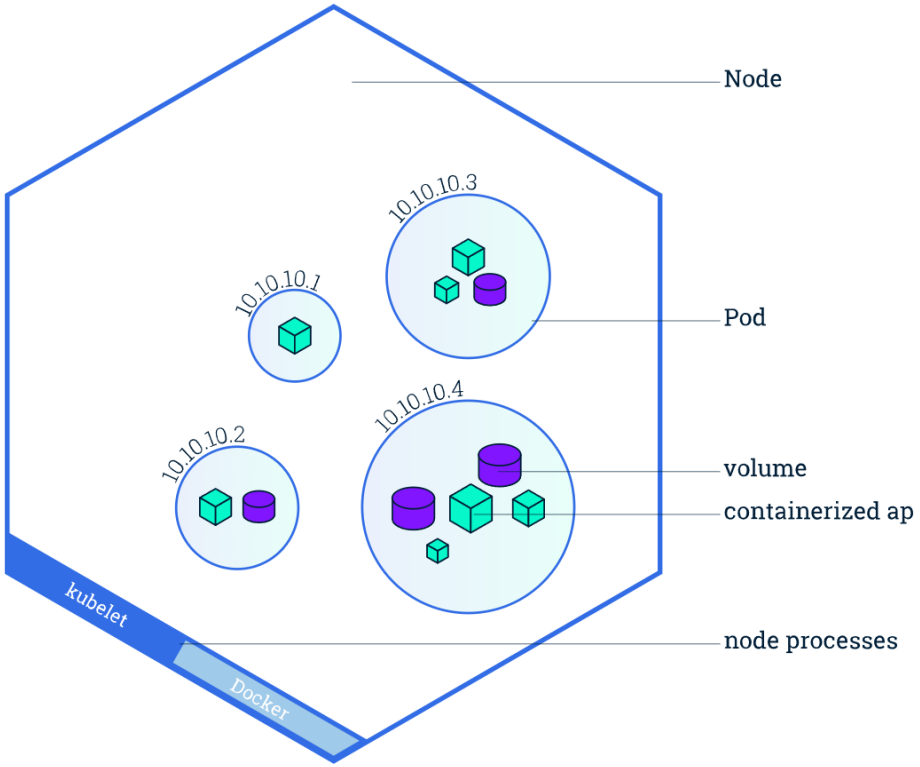
What is a pod?



Pods are the smallest deployable units

- They are groups of containers
  - They shared storage and network resources
  - Containers should be tightly coupled
- Two models:
  - One pod per container: most common and simple to understand (pods = container)
  - One pod many containers: containers need to be co-located in the same node and need to share resources

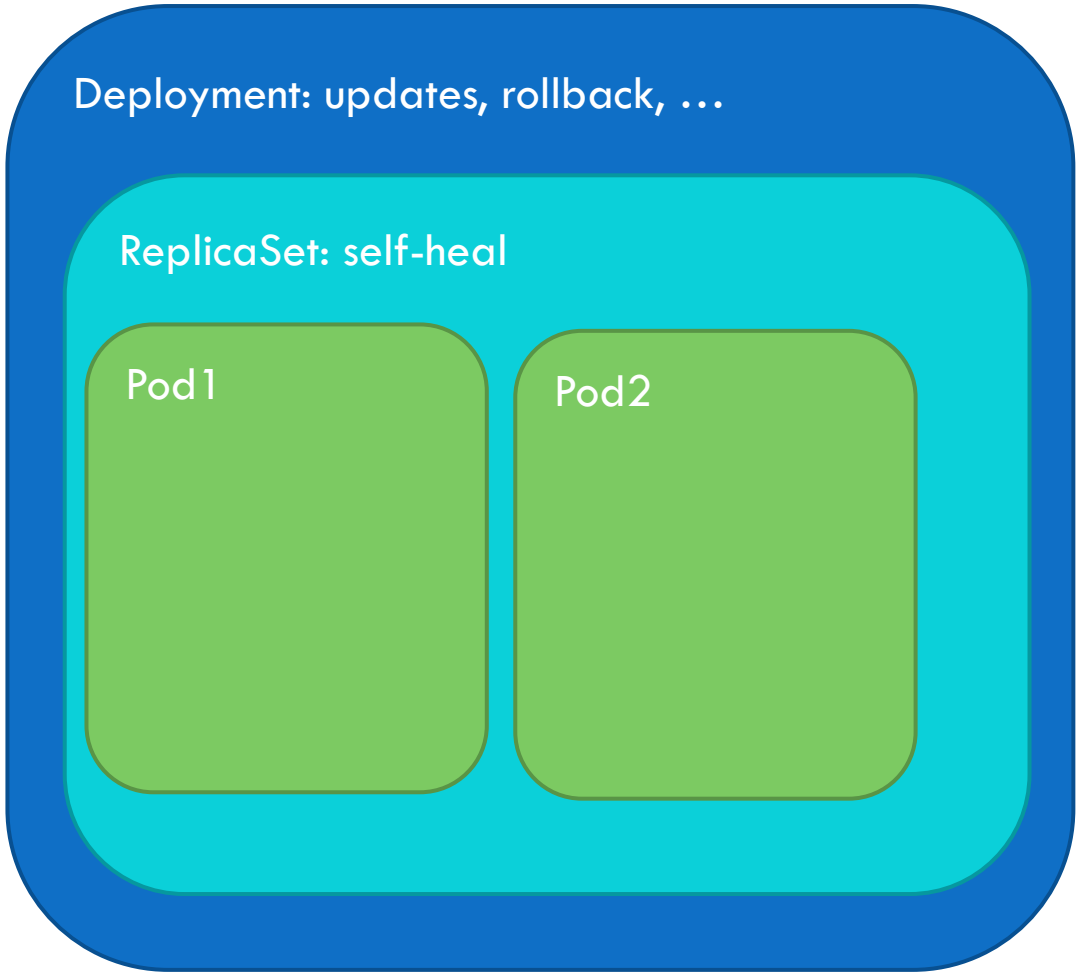
Creating a pod at least once in a lifetime



kubectl apply -f pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: examplepod
spec:
  volumes:
  - name: www_dir
    emptyDir: {}
  containers:
  - name: webserver
    image: nginx
    volumeMounts:
    - name: www_dir
      mountPath: /usr/share/nginx/html
  - name: updater
    image: debian
    volumeMounts:
    - name: www_dir
      mountPath: /html
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          date > /html/index.html;
          sleep 1;
        done
```

But nobody runs pods manually



In order to run Pods, we use high level abstractions:

- ReplicaSet: specifies that a precise number of Pods is running. Low-level compared to deployments.
- Deployment: similar but declarative. We describe the state we want, and the control plane creates/destroys ReplicaSets and Pods. **Recommended over ReplicaSets to create services**

How do we access apps?

What is a service?

By default, each Pod has a unique IP address in the internal network of k8s.

- However, since Pods are ephimeral, these IP addresses are dynamical
- In order to access a Pod, typically you should use a name.
- As an abstraction, services are used to refer to a Pod. This reference will be automatically handle by the cluster, even though Pods may die and respawn.

What is a service?

Services are abstraction to access applications, kind of providing an static way of accessing and exposing applications to the exterior or other Pods.

- There are three types:
  - ClusterIP: internal IP address for the service. Only works for other Pods in the cluster.
  - NodePort: nodes expose to the exterior the service by doing port forwarding
  - Load Balancer: service is deployed through an existing load balancer. Usually, requires DNS services and it usually works only for L7 (http/https)

What is a ingress controller?

Ingress controllers allow to expose application to the exterior with state-of-the-art standards such as custom domain name or encryption (TLS).

- Typically, they just receive request and forward them to existing services of the cluster.
- Work on top well-known industrial HTTP/TCP/UDP servers
  - Nginx (HTTP, HTTPS)
  - Haproxy (TCP, UDP)
- For our case, we are not going to use them since there are more oriented towards web applications.

The kubectl CLI

How to manage objects in k8s?

List them

kubectl get <type of object>

```
kubectl get services
kubectl get pods
kubectl get replicasets
kubectl get nodes
Kubectl get deployments
```

You may run everything in isolated namespaces

```
kubectl get services --all-namespaces
kubectl get pods -n spark
```

Extended information

kubectl describe <type of object> <name>

```
kubectl describe services
kubectl describe pods/nginx
kubectl describe -n spark replicasets
kubectl get nodes node1
```

Logs

kubectl logs <podname>

```
kubectl logs pods/nginx
```

Assignment

Exercise 1

Launch minikube on your computer.

- For the laboratories, follow the instructions on moodle
- For the rest, follow the guide at https://minikube.sigs.k8s.io/docs/start/

Follow a tutorial to check that it works.

- Try modifying some of the provided YAML files to try some options
- Link for the tutorial https://kubernetes.io/docs/tutorials/stateless-application/guestbook/

Exercise 2

Write a base Dockerfile based on ubuntu 22.04

- Install OpenJDK 11
- Install Hadoop (download and extract in /opt)
- Install Spark (download and extract in /opt)

Use that base Dockerfile to build the worker and master worker

Exercise 3

Write two deployments

1. For the master
2. For two workers

Write a service for the master exposing port 7077 for both internal usage and external access

Deploy both in the cluster and try to run a PySpark example on the host

- It should connect to localhost:<forwarded port> (see “minikube service list”)
  - Check the troubleshooting guide for more information about services in minikube. Be aware that this depends on the engine that minikube is using.
- Driver host should be the external IP address of your computer
  - Why? We need the Spark cluster to be able to find the PySpark program. Otherwise, the Driver (and the PySpark program) should run as a pod inside k8s.
- If you can not connect correctly from the host, you can run the test program from a Pod in the same namespace.
  - Check the troubleshooting guide in moodle for the command to do so.

Exercise 4

Scale up the number of workers to 3.

Check that the Spark dashboard shows all nodes.

Destroy the Spark cluster.

Can two Spark clusters co-exist on the same k8s infrastructure?

(Optional / Project ideas) What about storage for Spark on k8s? Ceph?

Material to submit

- You must write a report answering the questions proposed in each exercise, plus the requested files. Submit a zip file through Moodle. Check submission date in Moodle (deadline is until 11:59 pm of that date).
- From exercise 1:
  - Nothing
- From exercise 2:
  - Dockerfiles for creating the images
  - Explain how to build the images
- From exercise 3:
  - YAML file/s
  - Commented version of the YAML file
  - Explain how to launch the Deployments and Services on the cluster.
- From exercise 4:
  - Explain how to scale up and down the number of workers
    - Using kubectl apply and the YAML file
    - Using just kubectl
  - Explain how to destroy the Spark cluster
  - Explain how two Spark clusters can be deployed on the same infrastructure
  - (Extra credit/Project ideas) What about storage?