# A brief introduction to Kubernetes

Matteo Canzari, Matteo Di Carlo - INAF Osservatorio Astronomico d'Abruzzo

ICT INAF webinar: "Introduzione a Kubernetes"

13 Settembre 2021

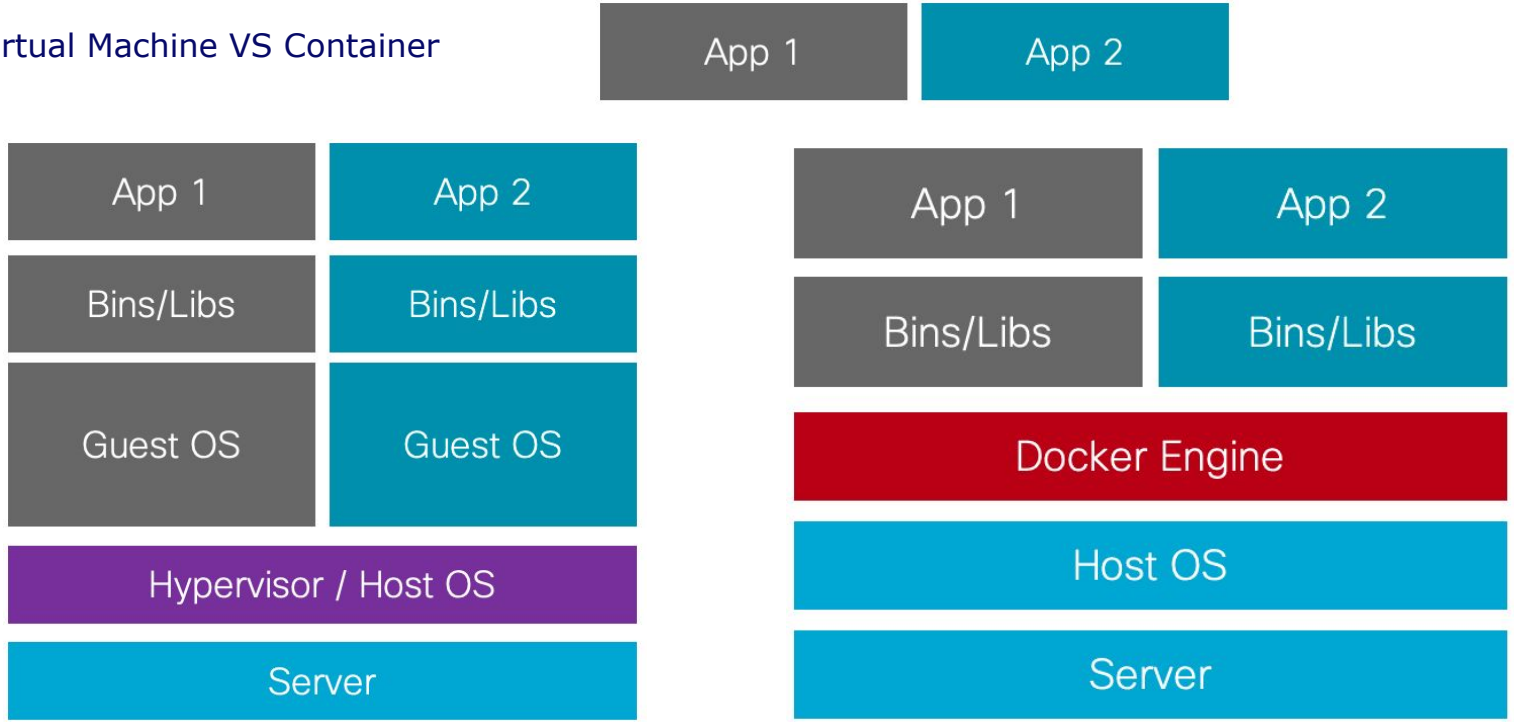
# Summary

- First Part (Matteo Canzari INAF - OAAb)
  - Software layer architecture
  - k8s
  - Pod, Deployment, StatefulSet, Service, Ingress, ConfigMaps, Persistent volumes

- Second Part (Matteo Di Carlo INAF - OAAb)
  - k8s in SKA

# What are containers?

- Virtual Machine VS Container



# Containers on cloud?

- How to run and manage containers in a cloud network?



- Manage more containers in different nodes
- Run more instances of the same container
- Scale up-down the resources based on load
- Load balancing
- Lifecycle container
- Different containers types in different nodes
- [.....]

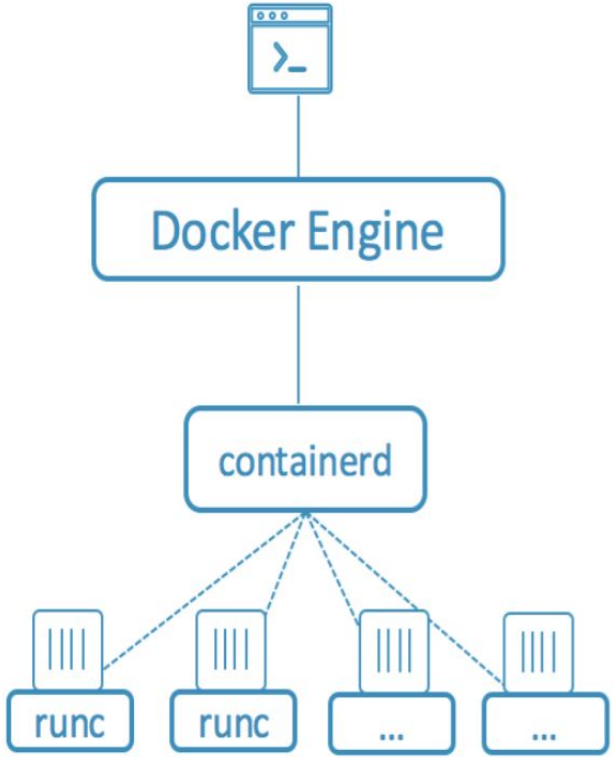
# What is Kubernetes?

- An **open-source** system for **automating deployment**, **scaling**, and **management** of **containerized applications**
  - Container **Orchestration**
  - Keeping your containers **up**, **scaling** them, **routing traffic** to them

kubernetes

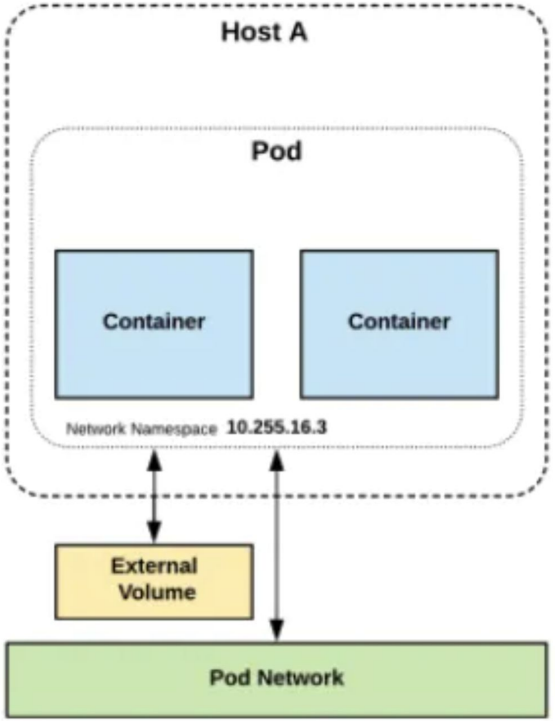
# Container Runtime and Container Runtime Standards

- A Container Runtime is a CRI (Container Runtime Interface) compatible application that executes and manges containers. (containerD, cri-o, rkt, kata..)
- Low-level containerization tools: RunC, ContainerD, Linux Container (LXC) and so on
- **O**pen **C**ontainer **I**nitiative (docker and so on) vs. **A**pp **C**ontainer **I**mage (rocket)
- Docker is not a container runtime, but it is a set of tool build on top of RunC, which is the very basic tool to run containers
- ContainerD is a intermedie command line client which communicates with RunC, but add more functionalities and flexibility.
- On top of ContainerD, Docker adds a lot of libraries and functionalities
- **k8s works only with ContainerD**

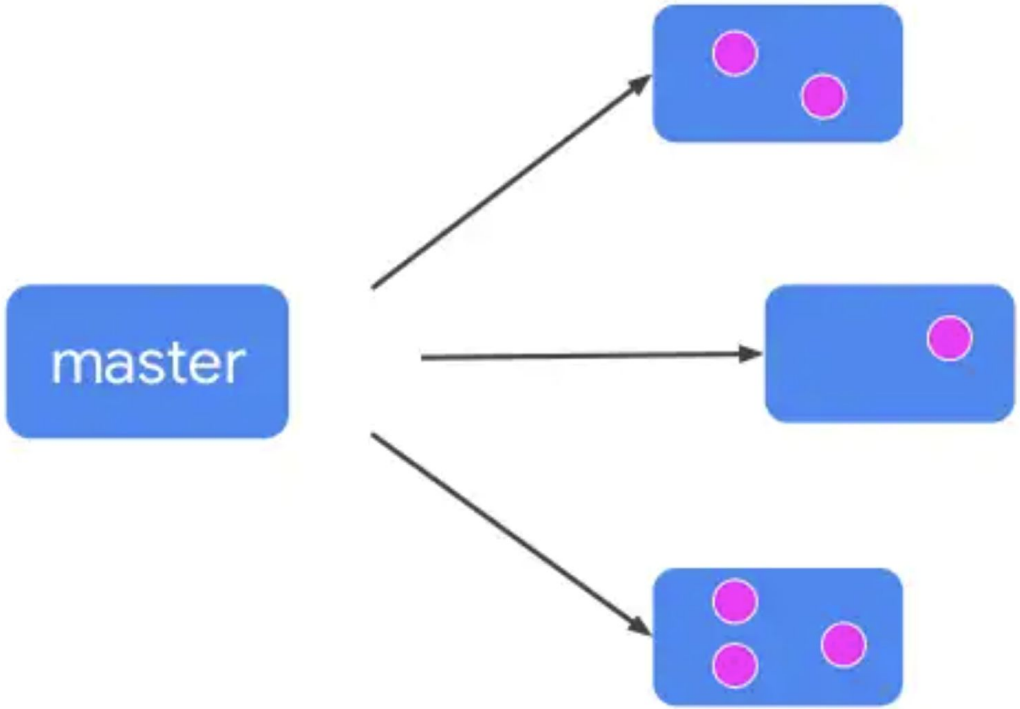


# POD

- A **Pod** consists of one or more **containers** which share an **IP address**, access to **storage** and **namespace**.
- **Atomic** unit of management
- Typically, one container in a Pod runs an application, while other containers support the primary application.



# Cluster
# Pods in a Cluster



# Deployments

*How many replicas should be running of a given pod?*

- A Deployment ensures that resources are available, such as IP address and storage, and then deploys a **ReplicaSet**.
- The **ReplicaSet** is a controller which deploys and restarts containers until the requested number of containers is running.
- Provide rollback functionality and update control
- Ports that should be exposed

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

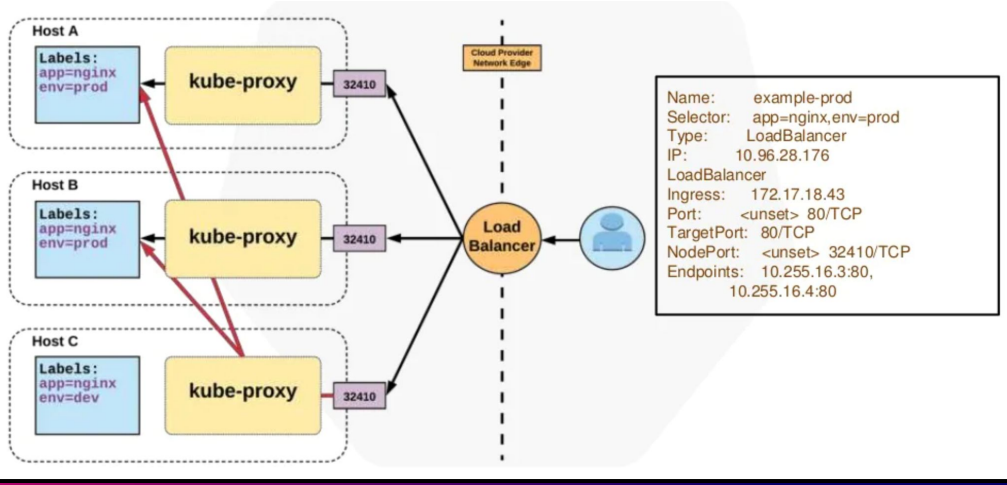
# StatefulSet

- StatefulSet is the workload API object used to manage **stateful applications**.
- Manages the deployment and scaling of a set of Pods, and provides **guarantees** about the **ordering** and **uniqueness** of these Pods.
- These pods are created from the same spec, but are **not interchangeable**: each has a persistent identifier that it maintains across any rescheduling.
- There are also **CronJob**, **DeamonSet**, **HeadlessService**

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx # has to match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: k8s.gcr.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "my-storage-class"
      resources:
        requests:
          storage: 1Gi
```

# Service

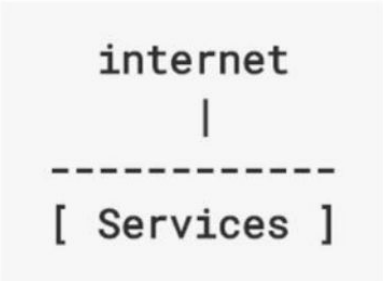
- An abstract way to **expose** an application running on a set of Pods as a network service. **Unified method** of accessing the exposed workloads of Pods.
- Each Service is a microservice handling a particular bit of traffic, such as a single **NodePort** or a **LoadBalancer** to distribute inbound requests among many Pods.

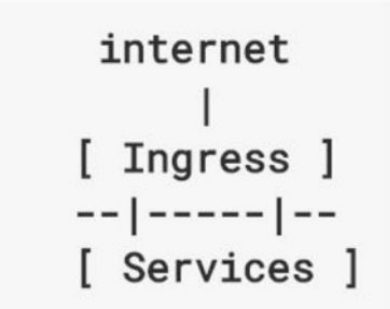



```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

# Ingress

- An API object that manages **external access** to the services in a cluster, typically HTTP.
- Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.
- Provides load balancing, SSL termination, and name/path based virtual hosting
- Gives services externally-reachable URLs



# Configuration: ConfigMaps and Secret

- Kubernates has an integrated pattern for **decoupling configuration** from application or container.
- **ConfigMaps** are **external data stored** within k8s
- Can be referenced through several different meas:
  - environment **variable**
  - a **command line** argument
  - injected as a **file** into a volume mount
- Can be created from a manifest, literals, directories, or files directly.
- **Secret** is identical to a ConfigMap, but is stored as **base64 encoded content**

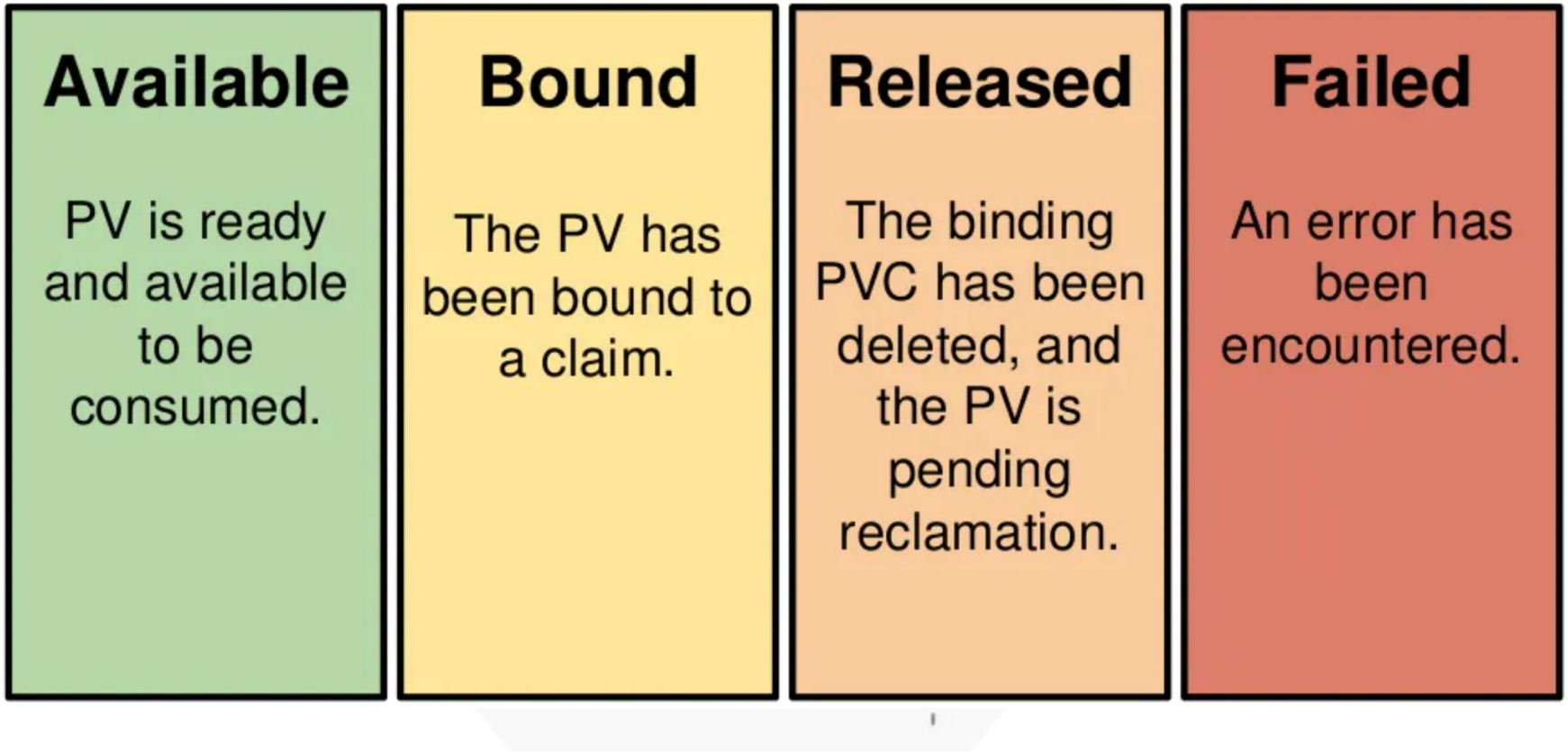
# Persistent volumes

- Pods by themself are useful, but many workloads requires **exchanging data** between containers, or **persisting** some form of data.
- For this purpose,k8s provides: **Volumes**, **PersistentVolumes**, **PersistentVolumeClaims**, and **StorageClasses**

# Persistent volumes

- **Storage Classes** are an abstraction on top of the external storage (AWSElasticBlockStore, AzureFile, AzureDisk …)
- **Volumes** is a storage that is tied to the Pod's Lifecycle. It can have one or more volumes attached to it. Can be consumed by any of the containers within the Pod and it survives from Pod Restart, however their durability beyond that is depend on the Volume Type.
- **PersistentVolume** represents a storage resource. It CANNOT be attached to a Pod directly. It relies on a PersistentVolumeClaim.
- **PersistentVolumeClaim** is a namespaces request for storage. Satisfies a set of requirements instead of mapping to a storage resource directly.

# Persistent volumes



Thanks!

matteo.canzari@inaf.it

*We recognise and acknowledge the Indigenous peoples and cultures that have traditionally lived on the lands on which our facilities are located.*

SKAO

www.skao.int