





# Kubernetes Course Session - 1



Contents





- Why kubernetes
- Learning basics of containers with docker & docker compose
- Learning basics of application deployment on kubernetes

# Main content

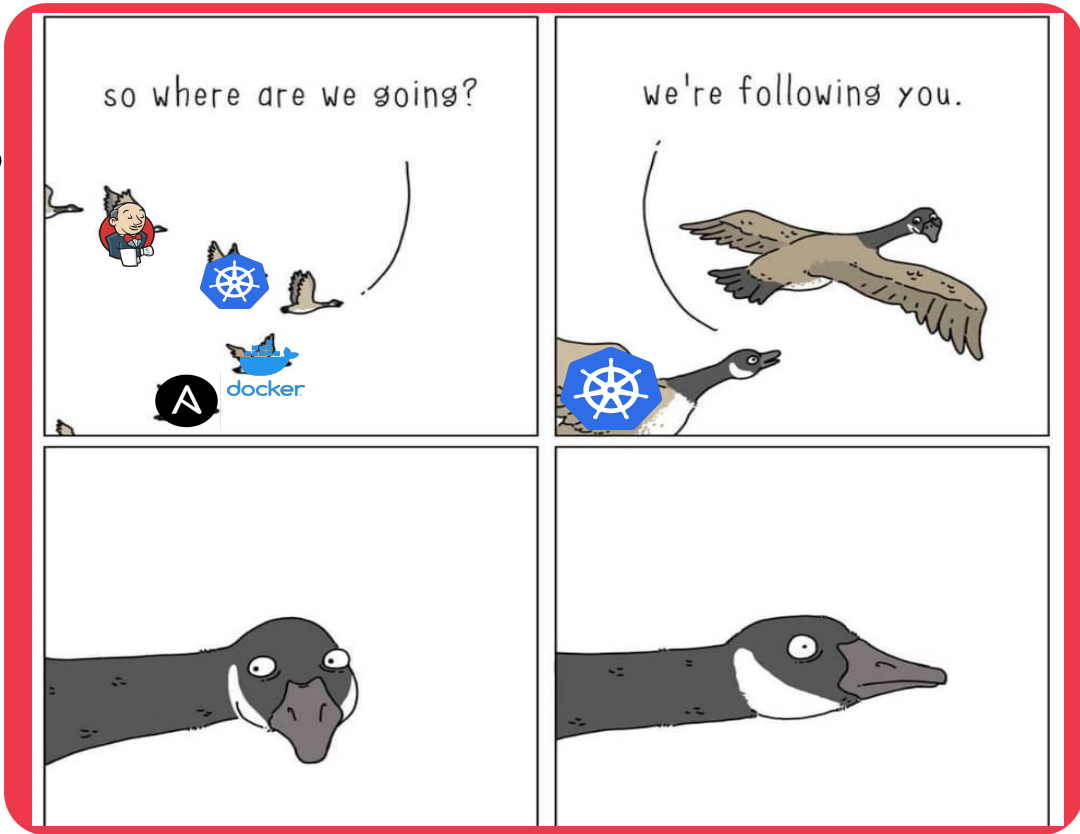


Main Content



# Why kubernetes?

- What is Kubernetes?
- Why Kubernetes?
- Where to use
- kubernetes?
- Kubernetes vs ...



# What is kubernetes? Search it!

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

**Kubernetes** ([/ˌk\(j\)uːbərˈnetɪs, -ˈnetɪs, -ˈnɛtɪz/, -ˈnetɪz/](#), commonly abbreviated **K8s**<sup>[3]</sup>) is an [open-source container orchestration](#) system for automating [software deployment](#), scaling, and management.<sup>[4][5]</sup> Originally designed by [Google](#), the project is now maintained by the [Cloud Native Computing Foundation](#).

The name *Kubernetes* originates from [Ancient Greek](#), meaning 'helmsman' or 'pilot'. *Kubernetes* is often abbreviated as *K8s*, counting the eight letters between the *K* and the *s* (a [numeronym](#)).<sup>[6]</sup>

Kubernetes works with various container runtimes, such as [containerd](#) and [CRI-O](#).<sup>[7]</sup> Its suitability for running and managing large cloud-native workloads has led to its widespread adoption in the data center. There are multiple distributions of this platform – from [independent software vendors \(ISVs\)](#) as well as hosted-on-cloud offerings from all the major public cloud vendors.<sup>[8]</sup>

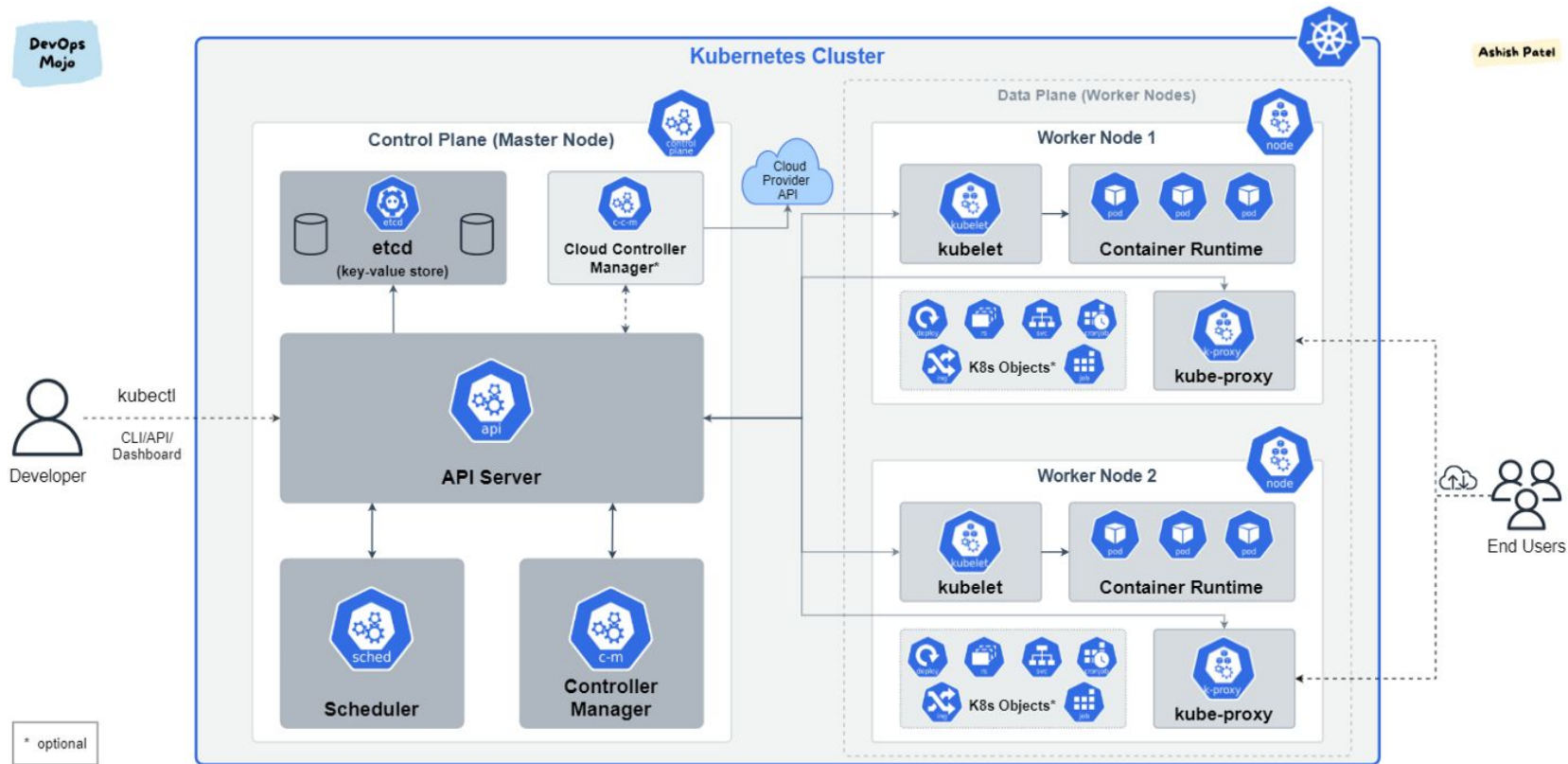
## Overview

Kubernetes (also known as k8s or “kube”) is an [open source](#) container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications.

Originally developed and designed by engineers at Google as the [Borg](#) project, Kubernetes was donated to the [Cloud Native Computing Foundation](#) (CNCF) in 2015. Red Hat® was one of the first companies to work with Google on Kubernetes, even prior to launch, and has become the [2nd-most leading contributor](#) to the Kubernetes upstream project.



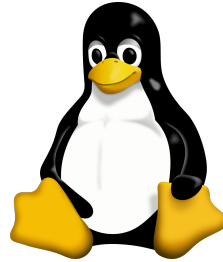
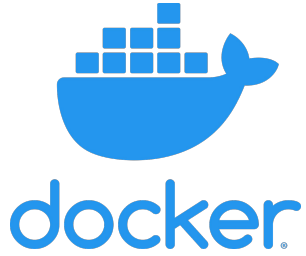
# How kubernetes works?



What?!



# Pre Requirements





# What is forbidden during the course?

Any GUI utils!  
Specially lens

The screenshot displays the KONTENA web interface. On the left is a sidebar with navigation options: Cluster, Nodes, Workloads (selected), Configuration, Network, Storage, Namespaces, Events, and Access Control. The 'Workloads' section is expanded, showing a list of pods. The 'Pods' tab is active, displaying a table of 30 items. The pod 'calico-node-rwf4g' in the 'kube-system' namespace is selected. The right panel shows the details for this pod, including a CPU usage graph, and a table of metadata such as Created time, Namespace, Status (Running), Node, Pod IP, Priority Class, QoS Class, Labels, Annotations, Conditions, Controlled By, Tolerations, Affinities, and Secrets.

Name	Namespace	Container
aws-node-hpz7f	kube-system	
aws-node-r9rgs	kube-system	
calico-node-rwf4g	kube-system	
calico-node-x8slr	kube-system	
calico-typha-78b876fc47-sw5...	kube-system	
calico-typha-horizontal-autos...	kube-system	
cert-manager-789598c8d7-ri4...	kube-system	
coredns-5b7d965bf9-8ttz2	kube-system	
coredns-5b7d965bf9-rp5q	kube-system	
dashboard-758bd48745-qpsft	kontena-lens	
helm-api-0	kontena-lens	
k8s-resource-applier-6d58457...	kontena-lens	
kube-proxy-svn6b	kube-system	
kube-proxy-v9x5g	kube-system	
kube-state-metrics-76fbcb879...	kontena-stats	
license-enforcer-5799c9c94c...	kontena-lens	
mariadb-1568285718-master-0	jakolehnm	
mariadb-1568285718-slave-0	jakolehnm	
metrics-server-6bd546f4cd-n...	kube-system	

**Pod: calico-node-rwf4g**

CPU Memory Network Filesystem

Usage Requests

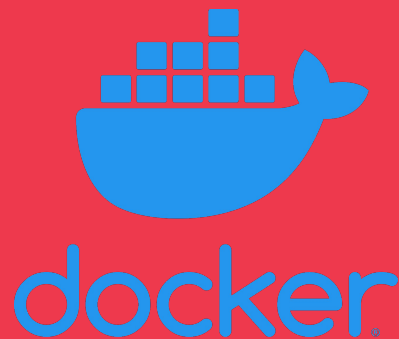
Created: 27d 5h 55m ago (2019-08-21T08:50:15Z)  
Namespace: kube-system  
Status: Running  
Node: jq-192-168-81-7.eu-north-1.compute.internal  
Pod IP: 192.168.81.7  
Priority Class: --  
QoS Class: Burstable  
Labels: controller-revision-hash: 5c9dddc74 k8s-app: calico-node  
pod-template-generation: 1  
Annotations: scheduler.alpha.kubernetes.io/critical-pod:  
Conditions: Initialized Ready ContainersReady PodScheduled  
Controlled By: DaemonSet calico-node  
Tolerations: 9  
Affinities: 1  
Secrets: calico-node-token-8dinkm



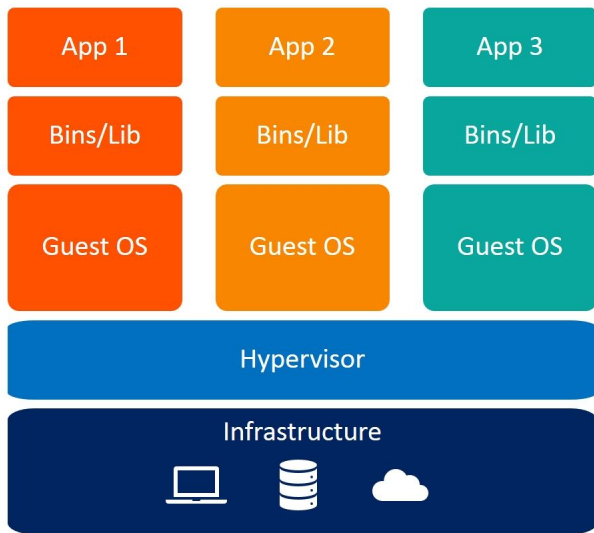
# Deploy application

Using docker

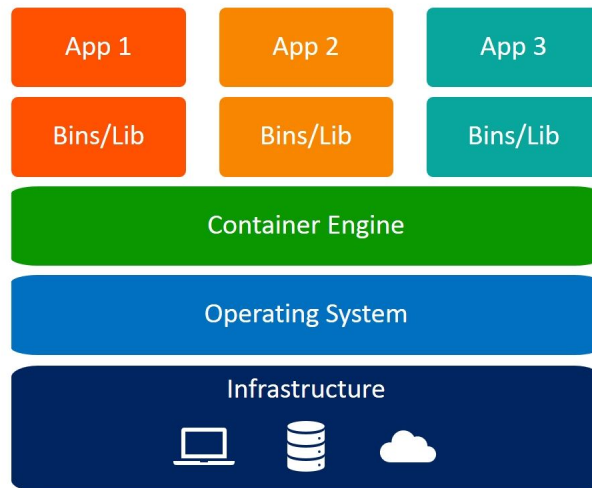
find out why docker compose



# Full virtualization vs Shared kernel



Virtual Machines



Containers

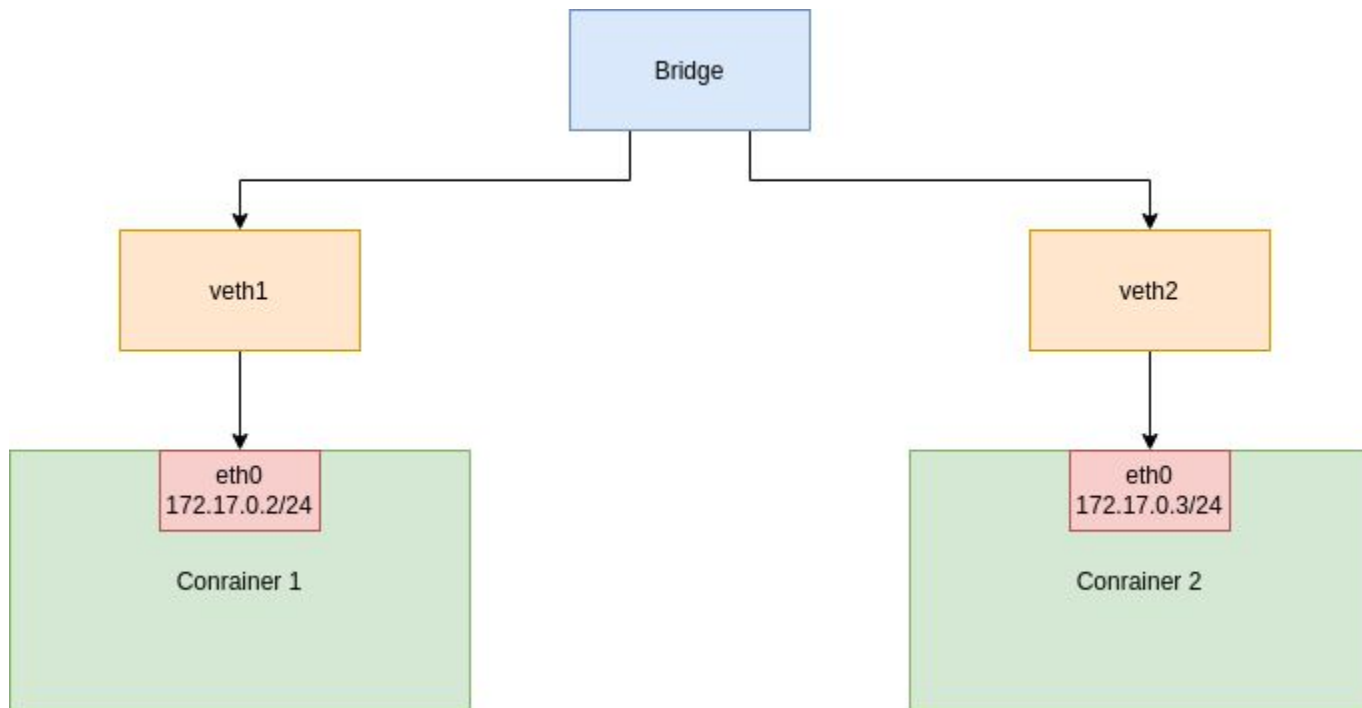


# Docker overview

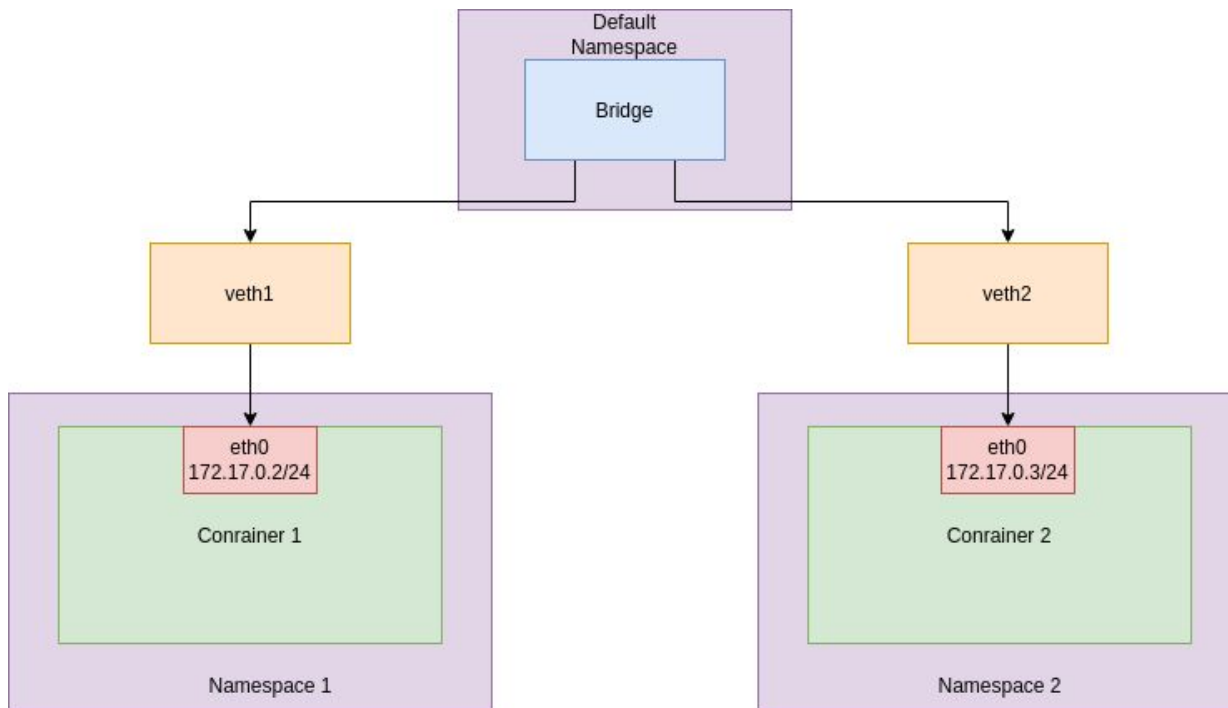
- Network namespace & bridge
- Filesystem isolation
- Resource management
- Onion Filesystem
- Management by cli interface



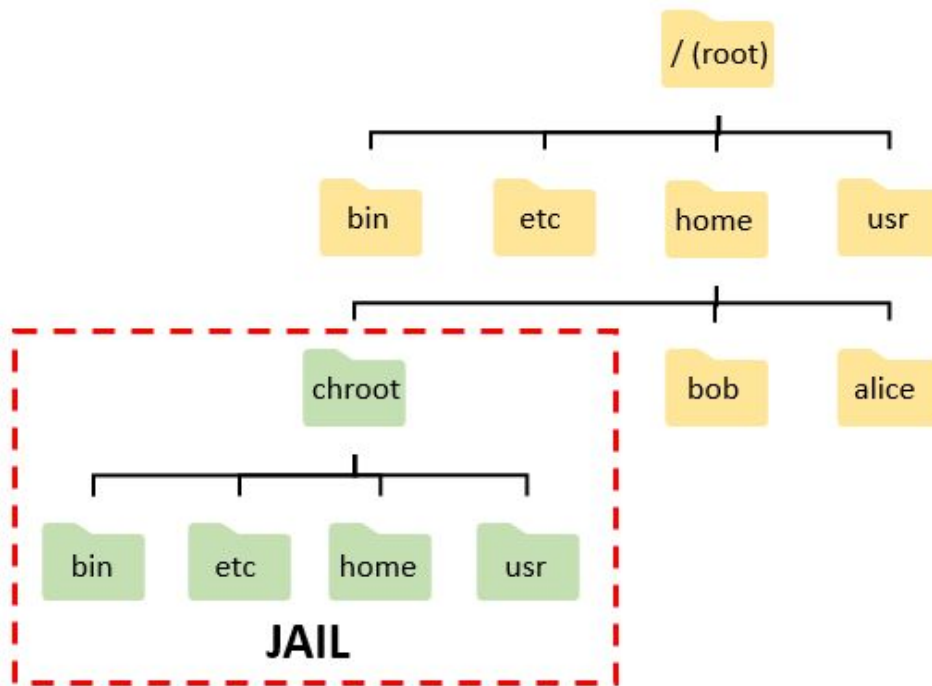
# Linux bridges



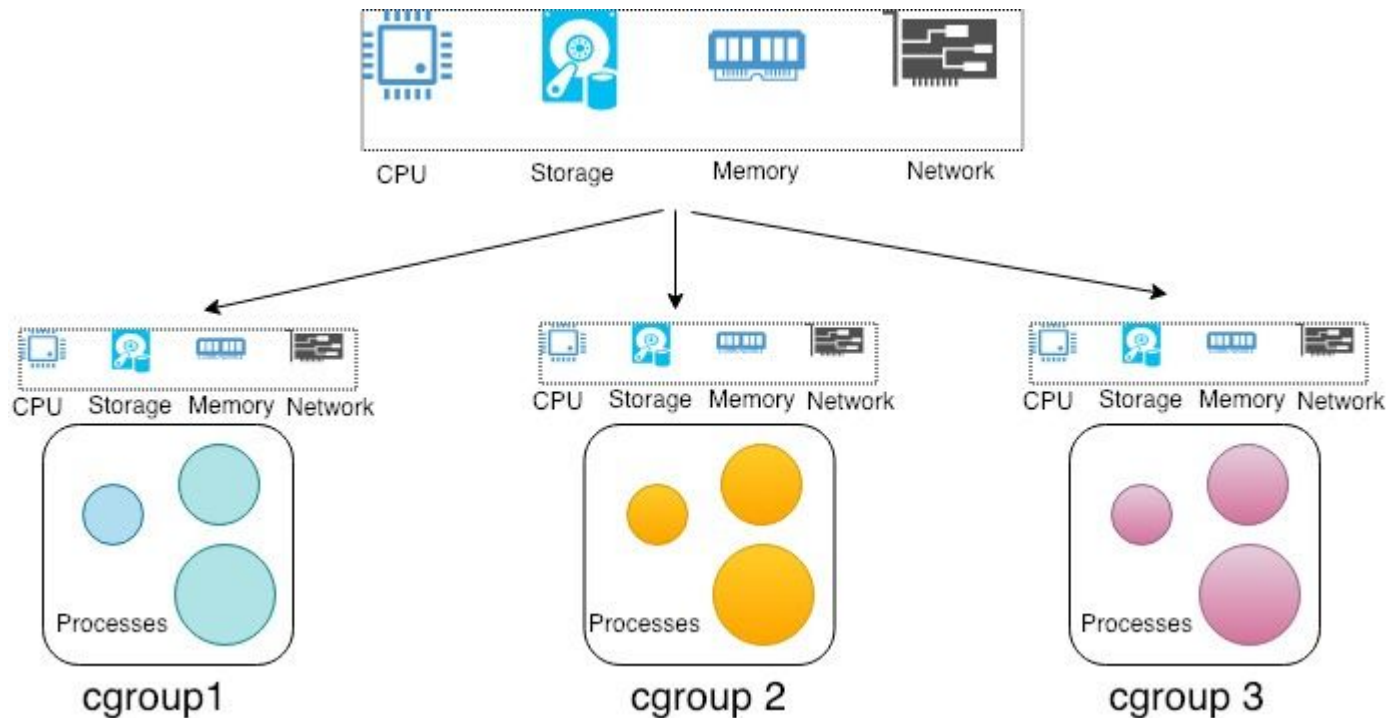
# Network namespaces



# Filesystem isolation



# Resource management

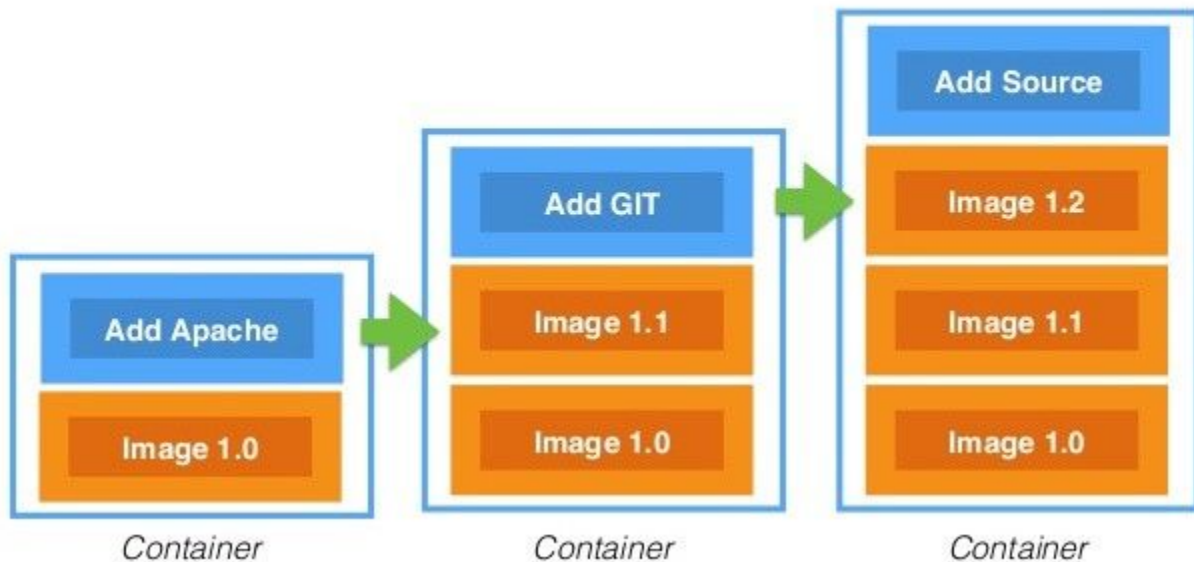




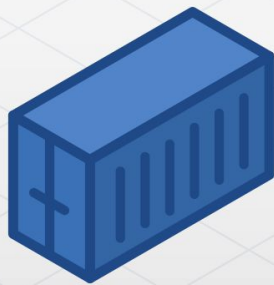
# Onion filesystem

**AuFS**  
*Layered Filesystem*

2019



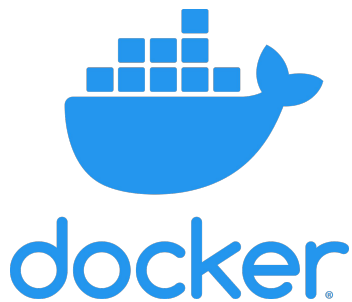
# CLI Management



*Containers in Fedora*  
**systemd-nspawn**



# Container technologies





# Deploy application

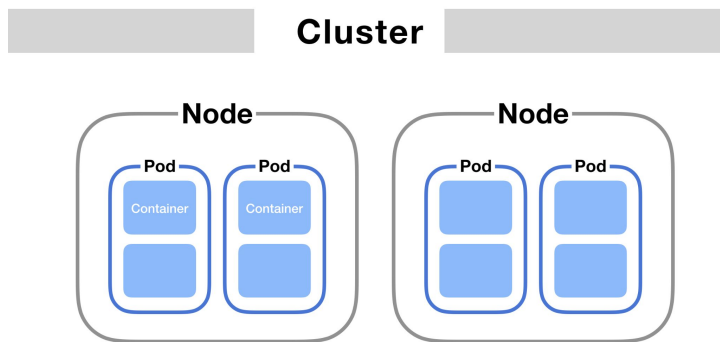
In kubernetes



# Pod

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

- Group of one or more containers, it contains one or more application containers which are relatively tightly coupled
- Shared storage and network resources, and a specification for how to run the containers



# Pod manifest

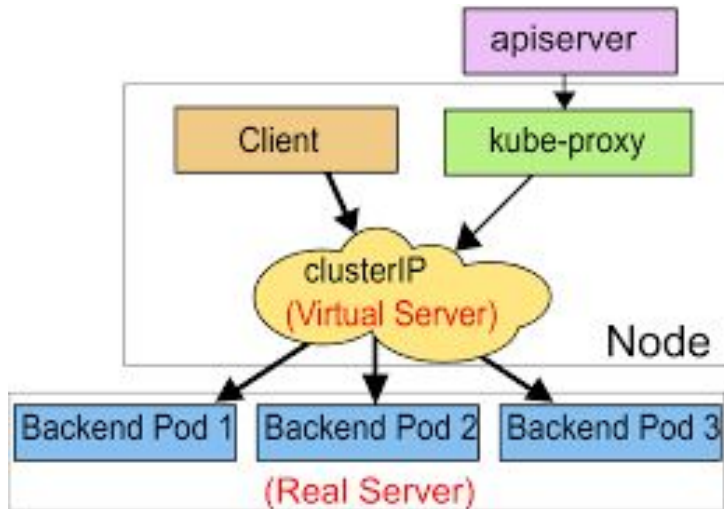
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    environment: production
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.21.6
```



# Service

An abstract way to expose an application running on a set of Pods as a network service

- Make a DNS name with a single ip address
- Load balance requests into the pods



# Service manifest

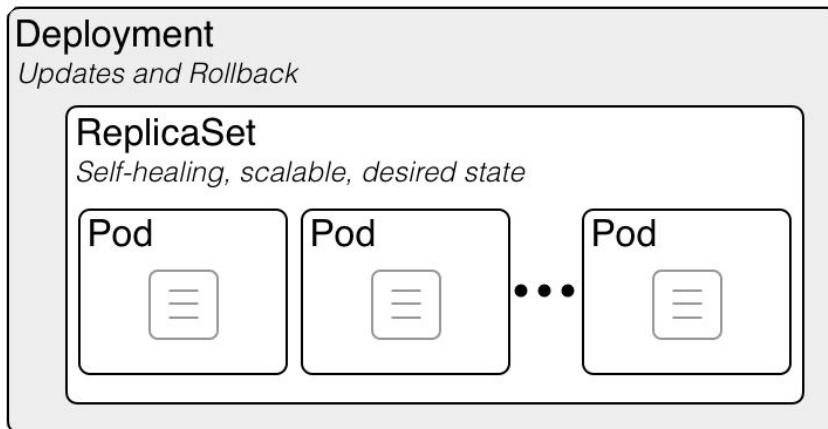
```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
    environment: production
name: nginx
namespace: default
spec:
  ports:
    - nodePort: 30537
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
    environment: production
type: NodePort
```





# Deployment

A Deployment is one of the Kubernetes objects that is used to manage Pods in a declarative way. It provides updates, control as well as rollback functionalities. This means you can update or downgrade an application to the desired version without experiencing a user blackout as well as roll back to the previous version



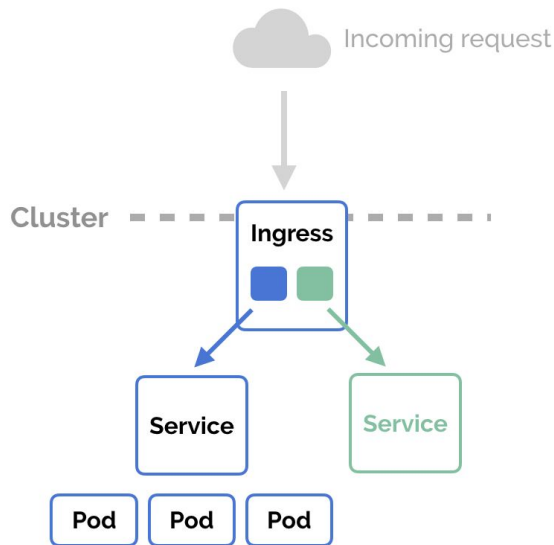
# Deployment manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
        environment: production
    spec:
      containers:
        - name: app
          image: khafan_app:v1
```



# Ingress

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.



# Ingress manifest

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
spec:
  rules:
    - host: nginx.local
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx
                port:
                  number: 80
```





Q&A



Q&A



— Thank You

[www.digikala.com](http://www.digikala.com)

I know how to deploy on  
kubernetes

We are not the same

