Kubernetes: Installation and Configuration Fundamentals

Introduction to Kubernetes

Build your own Kubernetes cluster

How to interact with our cluster

Prerequisites:

- Linux OS
- TCP/IP based networking
- Fundamental concepts of containers

Exploring K8s Architecture

Introduction

Exploring K8s Architecture

Installing and Configuring Kubernetes

Working with your K8s Cluster

Exploring K8s Architecture

What is Kubernetes?

- Container Orchestrator
- Workload Placement
- Infrastructure Abstraction
- Desired State

Benefits of Using Kubernetes

- Speed of deployment
- Ability to absorb change quickly
- Ability to recover quickly
- Hide complexity in the cluster

Kubernetes Principles

- Desired State/Declarative Configuration
- Controllers/Control Loops (Monitoring the running state of the ecosystem making sure that is achieving the desired state)
- Kubernetes API/The API Server

Kubernetes API

- API Objects - Collection of primitives to represent your system's state
- Enables configuration of state
  - Declaratively - Describe the implementation (deployment)
  - Imperatively - Execute sequence of commands in CLI to achieve the desired state
- RESTful API over HTTP using jSON
- The sole way to interact with your cluster
- The sole way Kubernetes interacts with your cluster
- Serialized and persisted

Kubernetes API Objects

- Pods - Single or collection of containers that we deployed as a single unit, basically our container based application

- Controllers - Keep our system in desired state, like replicas set and other things
- Services - Provides persistence access point to the application provided by the pods
- Storage - Persistence storage in our Apps

Pods

- One or more containers in a K8s cluster
- It’s your application or service
- The most basic unit of work
- Unit of scheduling
- Ephemeral - no Pod is ever “redeployed”, no state is maintained, once it goes away it never comes back
- Atomicity - they’re there or NOT
- K8s job is keeping your pods running
- More specifically keeping the desired state
    - State - is the pod up and running
    - Health - is the application in the pod running
    - Probes

Controllers

- Defines your desired state
- Create and manage pods for you
- Respond to pod state and health
- ReplicaSet
    - Number of replicas
- Deployment (Creates the replica set)
    - Manage rollout of ReplicaSets
- Many more... and not just Pods

Services

- Adds persistency to our ephemeral world
- Networking abstraction for Pod access
- IP and DNS name for the Service
- Dynamically updated based on Pod Lifecycle
- Scaled by adding/removing Pods
- Load Balancing

Storage

- Volumes (initial concept) - physical media directly accessible to a pod
- Persistent Volume - Pod independent storage, defined by admin and cluster level
- Persistent Volume Claim

Exploring Kubernetes Arquitecture

Cluster Components

- Control Plane Node
- Node (Worker Node)
- Can be either virtual or physical machines

Control Plane Node → Master Node

Control Plane Node

- API Server
- etcd - Storage
- Scheduler - Which nodes to start on based on configurations
- Controller Manager - Monitoring everything, keeping desired state

kubectl → interacts with API Server

API Server

- Central to the control, core to all operations, everything passes through that
- Simple UI
- RESTful
- Updates etcd

etcd

- Persists State
- API Objects
- Key-value

Scheduler

- Watches API Server
- Schedules Pods
- Resources
- Respect constraints

Controller Manager

- Controller Loops
- Lifecycle functions and desired state
- Watch and update the API Server
- ReplicaSet

Nodes

- Node is where the App pods run
- Starts up the pods and make sure the containers on those pods are running
- Also implements networking to ensure the reachability of the pods in the cluster
- Can either be physical or virtual machines

Node Components:

- Kubelet
  - Starts pods on the node - API Server (Manager Node)
- Kube-proxy
  - Pod networking - API Server (Manager Node)
- Container Runtime
  - Pulls image and provides environment to execute
- This services run on the manager node too (control plane node)

Kubelet:

- Monitors API Server for changes
- Goes and asks API Server : “Hey, do you have any work for me?”
- Responsible for Pod Lifecycle (Start and stop)
- Reports Node & Pod state
- “Is the Pod Up and Running?”
- Pod probes

Kube-proxy:

- iptables
- implement services
- routing traffic to pods
- Load balacing (distribute)

Container Runtime

- Download images & runs containers
- Container Runtime interface (CRI)
- Containerd
- Many others....

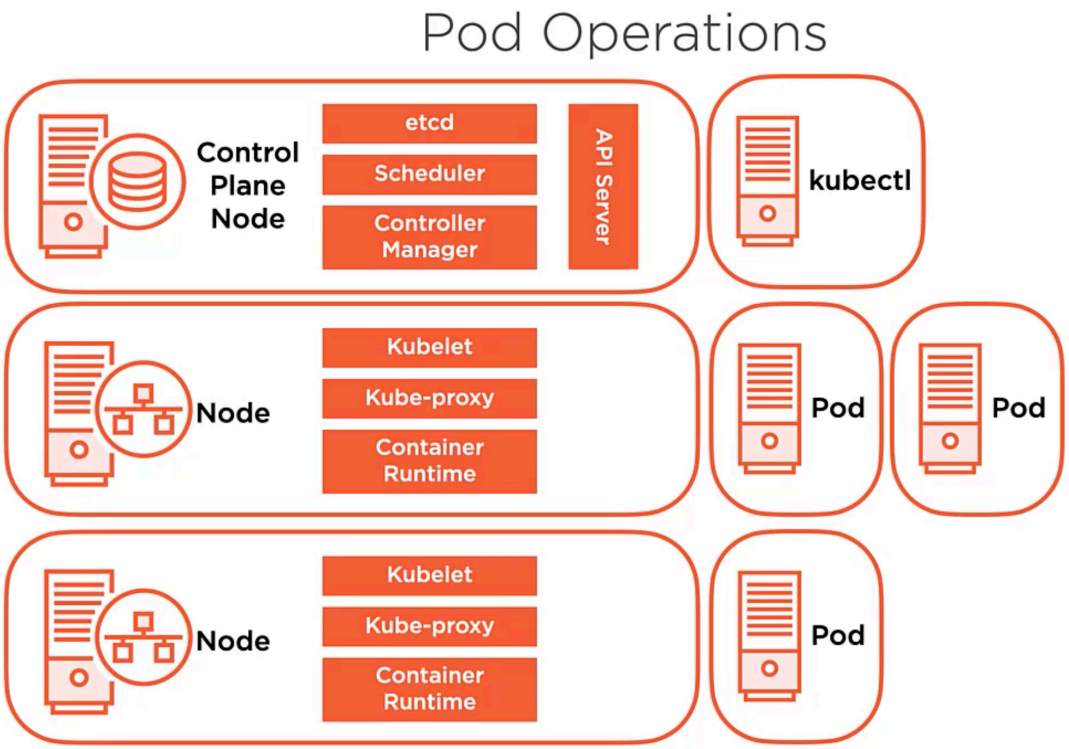
Can use docker as long as it is CRI complient!

Cluster Add-on Pods

- DNS - DNS Services inside the cluster, using CoreDNS DNS Server

- Ingress Controller - Load balancers and content routers
- Dashboard - Administration for Kubernetes Cluster

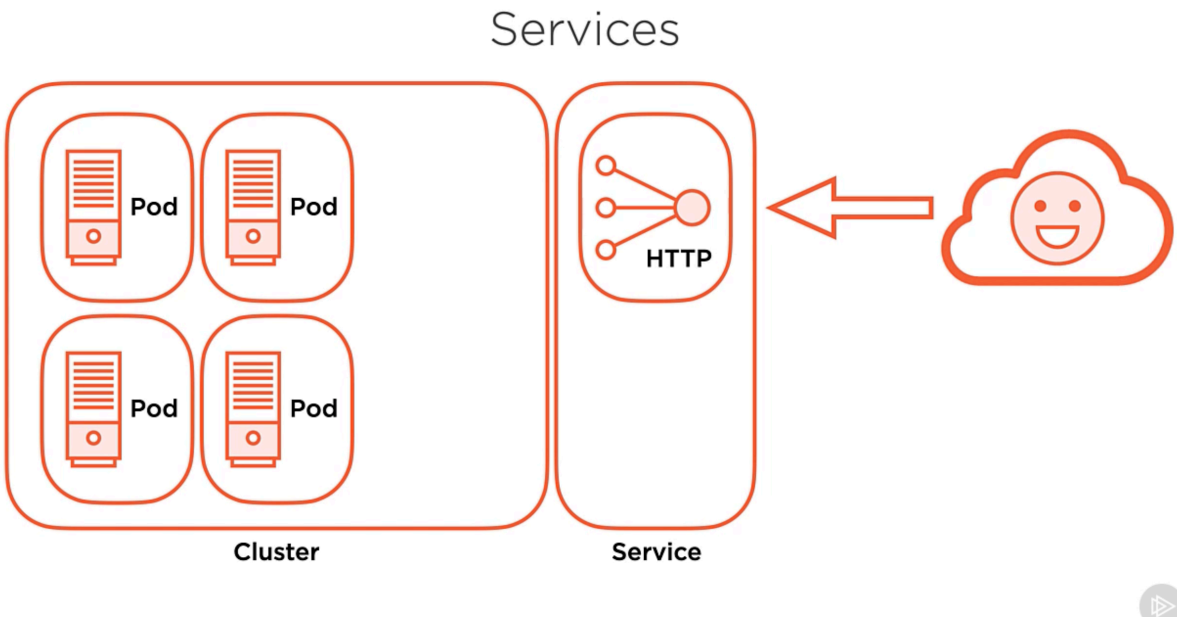
Pod Operations



- Action:
  - One Manager node and 2 worker nodes
  - Using **kubectl**, we submit code to instruct Kubernetes to create a deployment
  - For this example imagine we defined 3 **replicas** of our **pod**
  - Esta request vai ser submetida ao **API Server** do Manager node
  - É da responsabilidade do **API Server** de guardar essa informação de forma persistente no **etcd**
  - **Controller Manager** tem a responsabilidade de correr as 3 requested **replicas** e vai criar 3 **pods** diferentes
  - A request do **controller manager** vai chegar ao **scheduler**
  - As **replicas** vão ser distribuídas de forma balanceada pelos **nodes**
  - Essa informação é comunicada de novo ao **etcd**

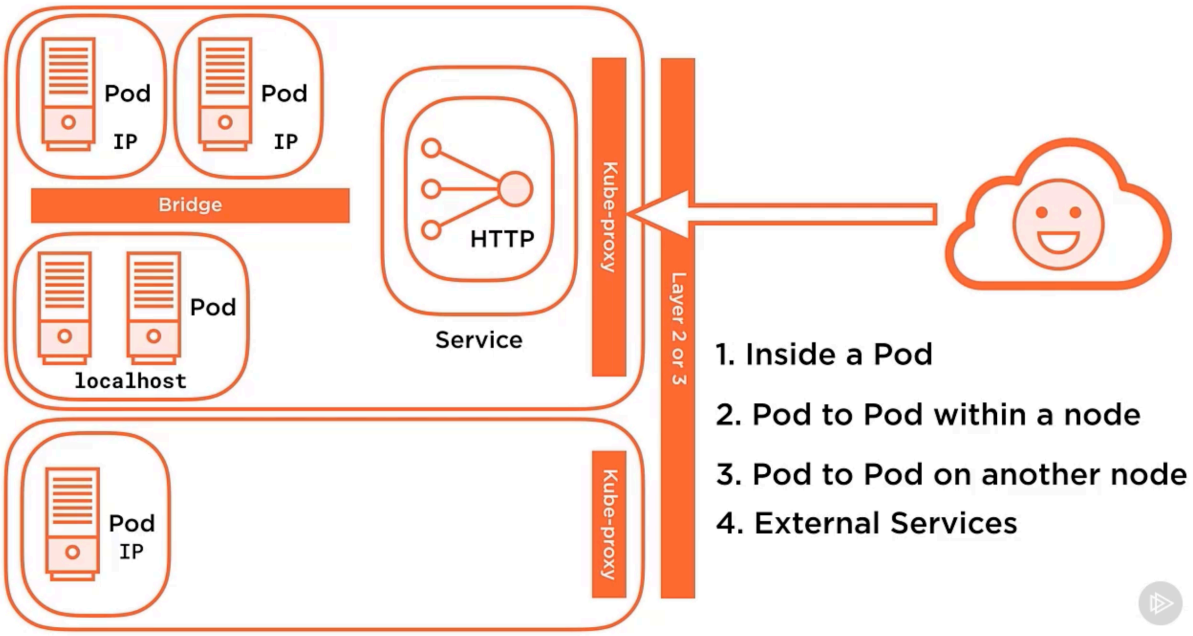
- As **kubelets** vão perguntar ao A**PI Server** se existe trabalho disponível
- E os **pods** vão ser atribuídos aos **nodes** e começam a correr
- Imaginemos que algum **node** vai abaixo?
- Esse **node** deixa de **reportar o seu estado**
- **Controller manager** conclui que esta fora do **desired state**
- Então vai submeter ao **scheduler** um pedido para voltar ao estado atual
- O **scheduler** vai encontrar um **node** (worker) para dar deploy do novo **pod** ou **pods**

Services



Kubernetes Networking Fundamentals

- Every pod has his own unique Ip address
- Pods on a Node can communicate with all pods on all nodes without Network Address Translation (NAT)
- Agents on a Node can communicate with all Pods on that Node



1. Inside a Pod
    a. Different containers communicate via localhost and namespaces
2. Pod to Pod within a node
    a. Communicates over a bridge layer using Ip address's
3. Pod to Pod on another node
    a. Layer 2 or Layer 3 connectivity using Ip address of the pod
    b. Overlay networking is a possibility
4. External Services
    a. Service in the cluster we want to expose to the world using HTTP via kube-proxy

Installing and Configuring Kubernetes

- Installation Considerations
- Installation Overview
- Getting Kubernetes
- Installing a Cluster with kubeadm

- Creating a Cluster in the Cloud

Installation Considerations

- Where to install?
  - Cloud
    - IaaS - Virtual Machines
    - PaaS - Managed Service
  - On-Premises
    - Bare Metal
    - Virtual Machines
- Which one should i choose?
- Other considerations:
  - Cluster Networking
  - Scalability
  - High availability
  - Disaster recovery

Installation Methods

- Desktop Installation (Great to Learn and Dev Environments)
- kubeadm (The one we are using in the course, on VM machines on prem)
- Cloud Scenarios

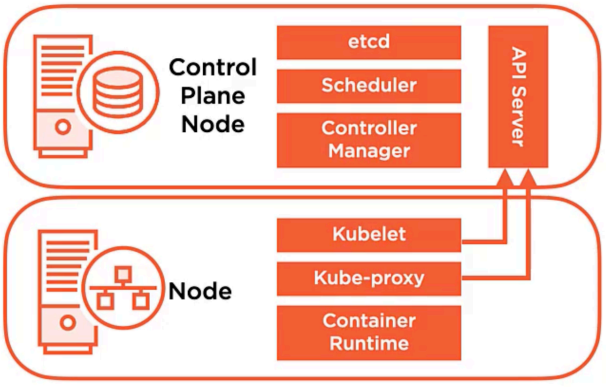
Installation Requirements

- System Requirements
  - Linux - Ubuntu/RHEL
  - 2 CPUs (Mín)
  - 2GB RAM (Mín)

    - Swag Disabled
- Container Runtime
    - Container Runtime Interface (CRI)
    - containerd
    - Docker (deprecated 1.20)
    - CRI-O
- Networking
    - Connectivity between all nodes
    - Unique hostname
    - Unique MAC address

Cluster Network Ports

- Worker nodes contact the API Server on the manager node via TCP/IP



Component	Ports (tcp)	Used By
API	6443	All
etcd	2379-2380	API/etcd
Scheduler	10251	Self
Controller Manager	10252	Self
Kubelet	10250	Control Plane
Kubelet	10250	Control Plane
NodePort	30000-32767	All

Getting Kubernetes

- Maintained on Github
- https://github.com/kubernetes/kubernetes

Building Kubernetes

- Install and Configure Packages
- Create your cluster
- Configure Pod Networking Environment
- Join nodes to your cluster (worker nodes for app workloads)

Required Packages

- containerd - container run time
- kubelet - drives the work to individuals nodes
- kubeadm - tools responsible to bootstrap the cluster and make everything running and configured
- kubectl - primary command line tool to admin the workload in the cluster
- This should be installed on all nodes in the cluster!

Installing Kubernetes on Ubuntu VMs

(Do this on all nodes)

containerd

apt-get install -y containerd

additional configurations:

- add gpg key to the apt repo where the k8s packages are
- add the k8s apt repo to our local repositories lists

apt-get update

apt-get install kubelet kubeadm kubectl

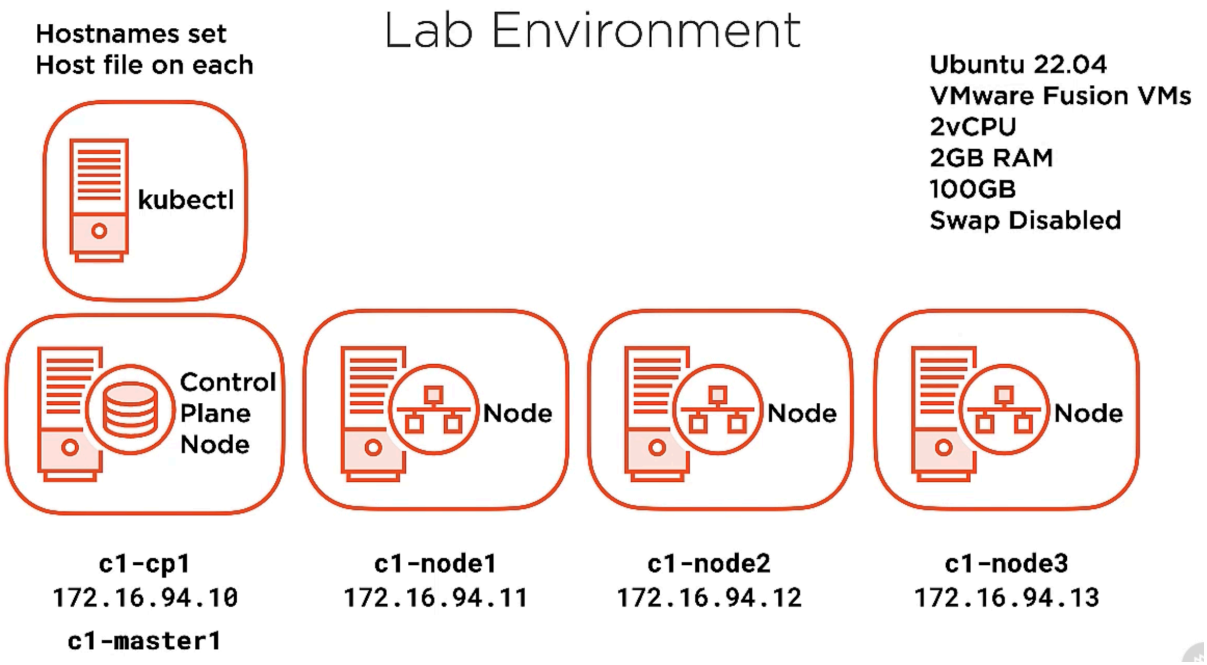
apt-mark hold kubelet kubeadm kubectl containerd

(we can control versions independent of security patches)

Lab Environment Overview

1 Control Plane Node

3 Worker Nodes



Demo: Installing and Configuring containerd

Installing:

- containerd
- kubelet
- kubeadm
- kubectl
- systemd Units

Bootstrapping a Cluster with kubeadm (This process is customizable)

- kubeadm init
- pre-flight checks
- creates a certificate authority
- generates kubeconfig files
- generates static pod manifests

- wait for the controlor plane pods to start
- taints the control plane node
- generates a bootstrap token
- starts add-on components: DNS and kube-proxy

Certificate Authority

- Self signed Certificate Authority (CA)
- Can be part of an external PKI
- Securing cluster communications
- API Server
- Authentication of users and cluster components
- live in /etc/kubernetes/pki
- distributed to each node when they join the cluster

Kubeadm Created kubeconfig files

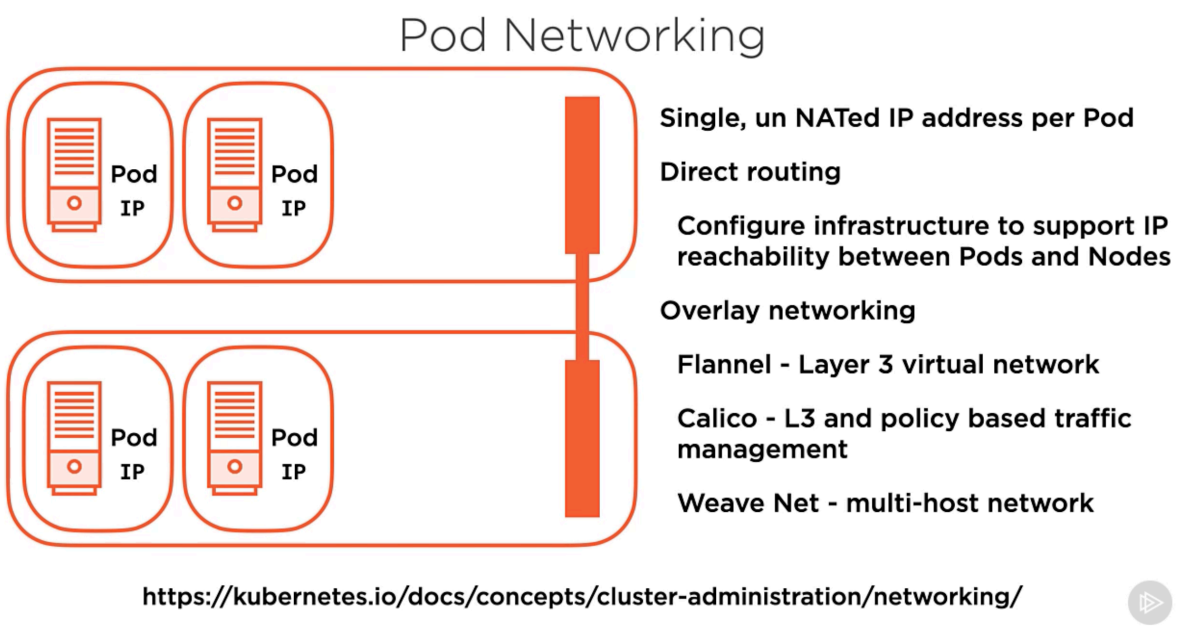
- Used to define how to connect to your cluster
- Client certificates
- Cluster API Server network location
- These live in /etc/kubernetes
    - admin.conf (kubernetes-admin)(cluster su)
    - kubelet.conf
    - controller-manager.conf
    - scheduler.conf

Static Pod Manifests

- Manifest describes a configuration
- Live in /etc/kubernetes/manifests
    - etcd

    - API Server
    - Controller Manager
    - Scheduler
- Watched by the kubelet started automatically when the system starts and over time

Pod Networking



Adding a Node to a Cluster

- Install Packages
- kubeadm join
- download cluster information
- node submits a CSR
- CA signs the CSR automatically
- Configures kubelet.conf

Example:

- Adding a node to a cluster
    - kubeadm join 172.16.94.10:6443 \ —token ... \ —discovery-token-ca-cert-hash \ sha256:989fdf8712....