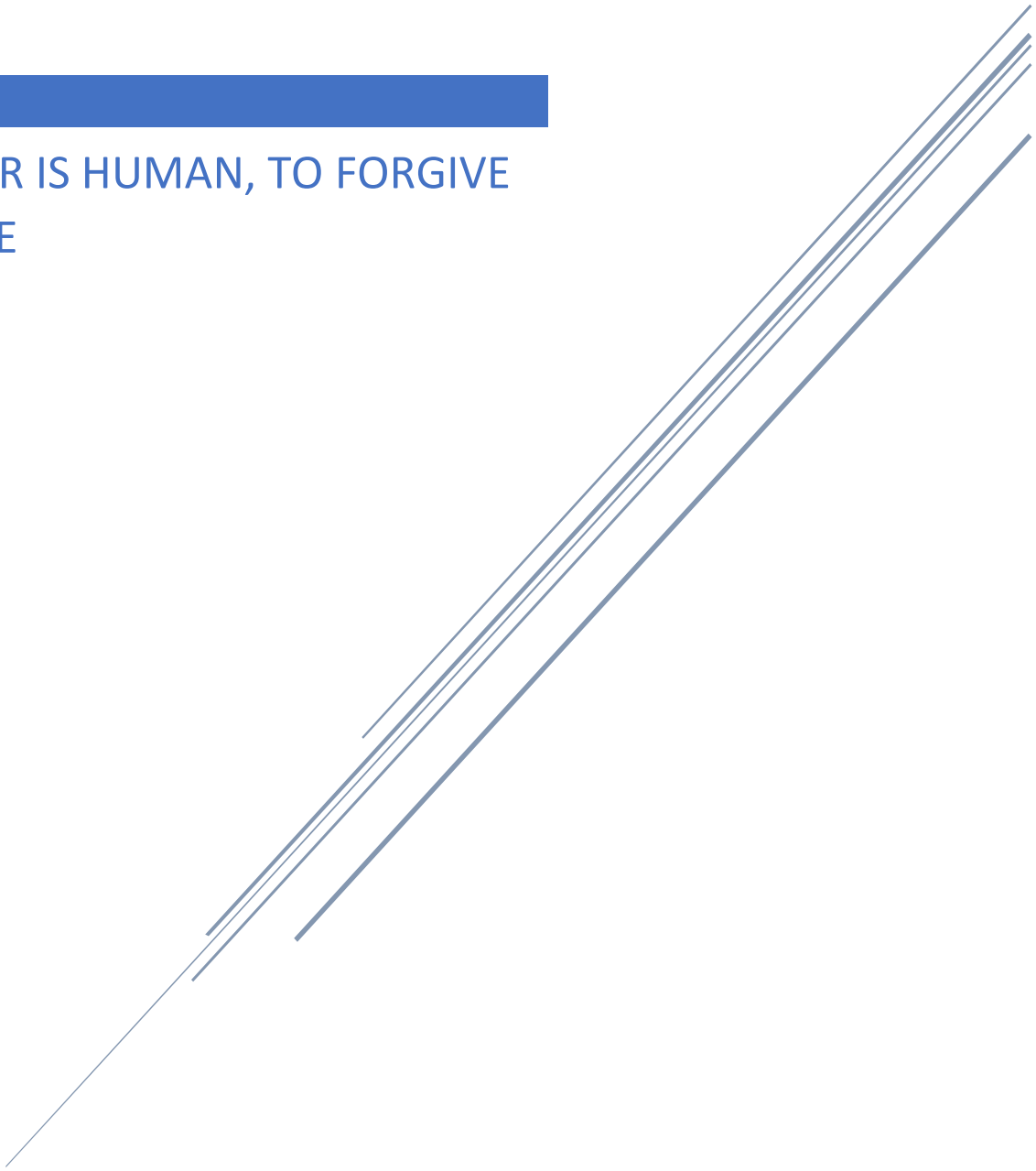
KUBERNETES LAB MANUAL

Version 1.0

TO ERR IS HUMAN, TO FORGIVE DIVINE



Akash Bandyopadhyay

GS Lab

Table of Contents

Containers Prerequisites

Figure 1: Containers vs VM

- Standard unit of software → Packages code, and dependent libraries into a single unit which is portable and can be executed in any environment with a container runtime
- Isolates applications into small, execution environments that share OS kernel
- Docker utilizes **layered file system(union mount)**, basically the idea behind it is merging different volumes(layers) into one logical volume
- Docker utilizes **cgroups** and **namespaces** to achieve isolation
    - Docker inspect <container-id>
    - Ps aux <container-id>
    - Pstree
    - /sys/fs/cgroup → Find with <container-id>
    - ls -Lil /proc/<PID>/ns

Hands-On 1: Basic familiarity with docker and its commands

a. Create Docker image with provided Dockerfile for 'Primary App'
b. Create Dockerfile for 'Secondary App' with the provided code
c. Run 'Primary App' and 'Secondary App'
    a. Connectivity with host
    b. Connectivity amongst each other
d. Push both the Docker images to Docker Hub

References:

docker run reference documentation

Dockerfile reference documentation

References:

Redhat Blog For Docker and Containers

How Containers Work

Windows Vs Linux Containers (How Windows can run Linux containers)

How Docker Utilizes **namespaces** and **cgroups**

Docker Storage Driver

Docker filesystem Demystified

Kubernetes Overview

What Is Kubernetes?

- Portable and extensible open-source platform for managing containerized workloads and services
- Does not limit application type that it can run – If the application can run within a container, it can run within Kubernetes
- Uses Docker images
- To run containers, Kubernetes supports multiple container runtimes, namely: Docker, rkt, runc, and any other implementation of OCI runtime-spec
- Not a traditional orchestration system
    - Comprises of a set of independent, composable control processes that continuously drive the current state to desired state.
        - Does not matter how we reach A→C
        - No need for centralized control
    - This results in a platform that is easier to use, powerful, robust, resilient & extensible

Kubernetes Architecture

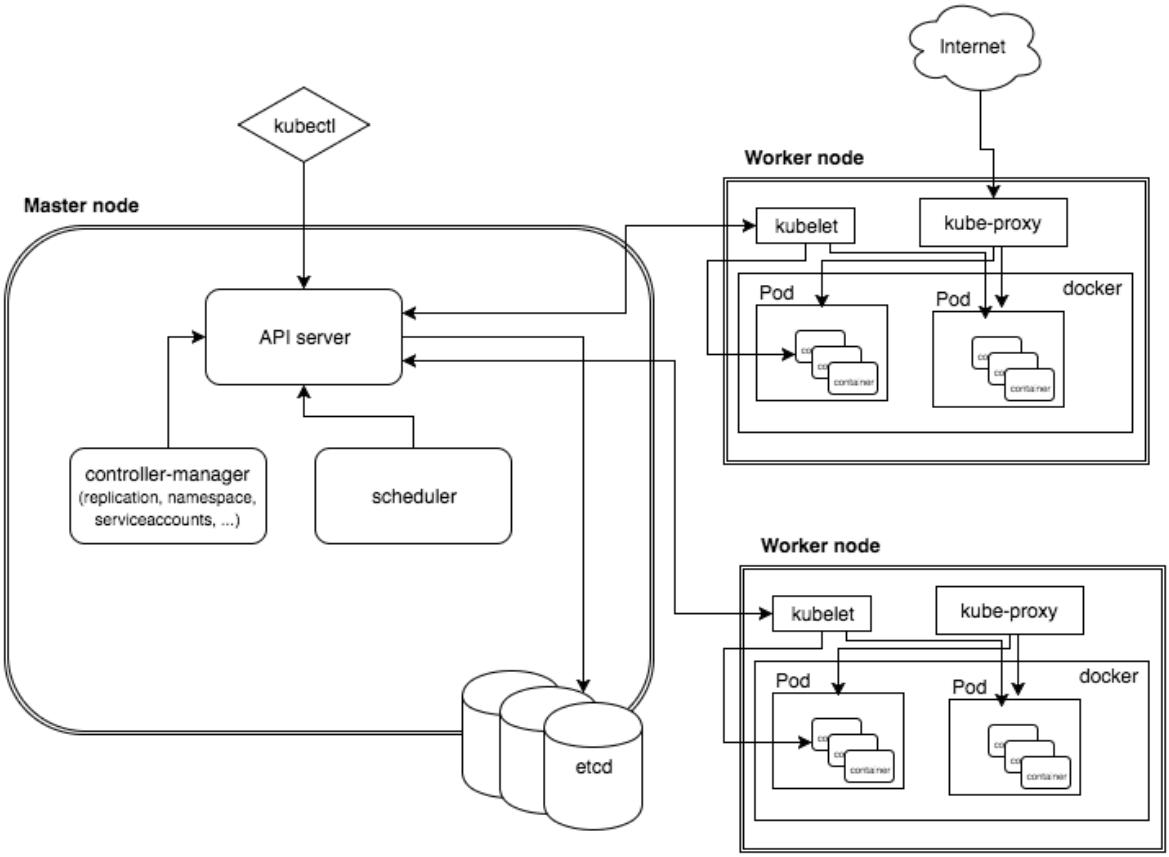


*Figure 2: Architecture of Kubernetes (**Source**: https://x-team.com/blog/introduction-kubernetes-architecture/)*

Kubernetes Installation

There are multiple approaches to setup a Kubernetes cluster, documented at Kubernetes Setup Documentation

To learn in-depth, how to setup a Kubernetes cluster from scratch, refer Kubernetes Setup Scratch Documentation

In this Lab, we are going to work on a Kubernetes cluster setup via Kubeadm

Notes:

- For Kubeadm to work, 'swap' needs to be disabled
- kubeadm init --pod-network-cidr=192.180.0.0/16 --apiserver-advertise-address=192.168.145.136
  This pod-network-cidr is required if 'calico' pod network add-on is used

What does "Kubeadm" do?

For a detailed overview and working of 'kubeadm' and check different options available, refer Kubeadm Documentation

Other Notable Points

- **What are the Kubeconfig files?**
  Used to configure access to multiple clusters by using configuration files. After your clusters, users, and contexts are defined in one or more configuration files, you can quickly switch between clusters by using the kubectl config use-context command.
    - Reference Links:
      Configure Access Control via Kubeconfig
- What are Contexts?
- What is User in Kubernetes Configuration file?
- Purpose of certificates?
  Certificates are used for authentication and authorization
    - Reference Links:
      Manage Certificates in Cluster
      PKI Certificates
- Purpose of Pod Network?
  To facilitate communication between pods
    - Reference Links:
      Kubernetes Cluster Admin Guide - Networking
      Kubernetes Networking Explained

Important Directories

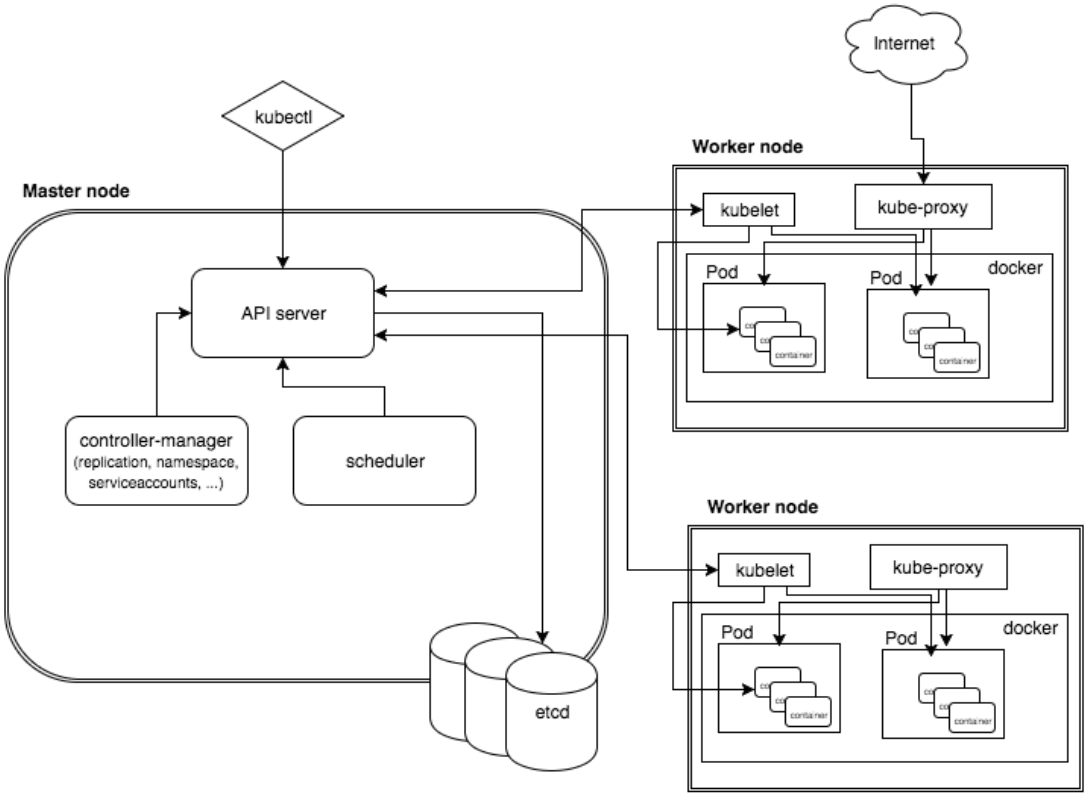
- /etc/kubernetes

References:

Refer the following links (in addition of official Kubernetes documentation, to understand how Kubeadm or Kubernetes cluster works):

- How Kubeadm Initializes Kube Master
- Kubernetes The Hard Way

Architecture in Detail



Different components of a Kubernetes cluster can be found documented at [Kubernetes Components Documentation]

Kubernetes Hands-On – Day 1

What are namespaces in Kubernetes?

- Used to divide a cluster between multiple users. The resources of a cluster can also be divided using [resource quotas](used together with namespaces)
- Determines DNS entries for Services and Pods
  <service-name>.<namespace-name>.svc.cluster.local
  <pod-ip-address>.<namespace-name>.pod.cluster.local

References: To understand details of 'namespaces' in Kubernetes, refer the following links:

[Namespaces Walkthrough]

[Namespace Object]

[Why and How of Namespaces]

[Organize Better with namespaces]

What are Labels and Annotations?

Labels

- Key-value pairs attached to objects
- Used to specify identifying attributes of objects which are meaningful to end user
- Can be added and modified at any point of time
- Kubernetes itself uses labels extensively

Label Selectors

- Via label selector, user can identify a set of objects(with similar properties, use, etc)
- Two types of label selector ae present:
    - Equality-based: =, ==, !=
      kubectl get pods -l env=prod
    - Set-based: in, notin, exists
      kubectl get pods -l 'env in (production)'
- **Note:** Label selectors are not the only way to filter results. Field Selectors are used as well. For supported field selectors, this link can be visited: Field Selector Code

Annotations

- Arbitrary non-identifying metadata to objects.

Hands-On 2: Basic familiarity with kubectl, namespace, labels, and selectors

a. Investigate the YAML files
b. Deploy the YAML files
c. Basic kubectl familiarity
d. Deploy Kubernetes dashboard and access it

Kubectl cheat-sheet can be found here

Note: kubectl create and **apply** are two different approaches.
Create is imperative → Works directly on live objects
Apply is declarative → Works on resource files. The actual update/create operations are determined by kubectl itself.
More about these two different approaches can be read from here

Different workload deployment approaches

There are different approaches regarding how to deploy your workload within a container cluster, namely:

Pods

- Basic building block of Kubernetes
- Can run multiple containers.
- Encapsulates different resources, viz:
    - Storage
    - IP (multiple containers can communicate using localhost)
- Does not heal or scale → Healing and scaling are done via controllers(will be covered later)
- Each pod gets an unique IP address

Hands-On 3: Deploy Pods – Interconnectivity between the two Pods

a. Deploy primary and secondary app via pod specification yaml files
b. This yaml file should use the images pushed to Docker hub. So, pod specifications should be configured to use docker login
c. Add ENV to second app pod, so that it can connect to primary app

Reference doc for how to pull images from Docker Private Repo: Configure Docker Private Repo

Reference doc for how to pass pod specific details to containers can be found here

Reference doc for Pods can be found here

How to restrict Pod traffic/access?

Liveness and Readiness Probes

Liveness:

Readiness:

Note: There are a few other topics related to Pods, like Pod Preset, Init Containers, Pod Security Policy, etc. that we will cover later.

Deployments

Reference document for Deployment can be found here

Hands-On 4: Deploy Deployments – Multiple operations on the same

a. Create Deployment (for both primary and secondary app)
b. Check rollout status
c. Check rolling update
d. Check replica sets and Pods

Few Other Controllers, which are available:

- DaemonSet
- StatefulSet
- Jobs
- CronJobs

Services

- DNS entry gets created for Service IP(clusterIP)
- Set of pods targeted by service is usually determined by Label Selector
- Master and worker nodes can access via ClusterIP but DNS, by default is resolvable only from within cluster objects(like pods)
- There are different types of services:

    - **ClusterIP**
    - **NodePort**
    - LoadBalancer
    - ExternalName
- Headless Services: Are those with ClusterIP set to ‘None’. Not a single IP gets allocated, and no load-balancing is applied therefore.

The reference document for services can be found here

Hands-On 5: Create basic service and access the Pods

a. Create service for both primary and secondary app
b. Access exposed service from outside
c. Access primary from secondary app via service

Network Policy

- Specifies how a group of pods can communicate with each other and other network endpoints
- Uses labels to select pods and define rules to govern traffic
- The network plugin should have support of network policy(Calico does)

Hands-On 6: Create and apply a NetworkPolicy to restrict traffic from Pods of other namespaces