# History: #1(45)

1980’s numeronym [a word that involves numbers (k is the first and s is the last letter of Kubernetes including 8 number of alphabets in between that is why it is called K8s)]. Project seven aka borg and later called as omega by google. It was written in Golang. Later donated to CNCF (cloud native computing foundation). Kubernetes manifests can be written in json/ yaml.

# Some Def:

Monolithic (Mono means single lithic means stone), code in one go.

Microservices communicate with outer worlds through APIs, all these services are loosely coupled with each other.

# Imp Info:

Autoscaling for containers can be managed by Kubernetes. Docker swarm & Apache marathon are the competitor for Kubernetes.

## Online platforms for k8s:

Kubernetes playground

Play with k8s

Play with Kubernetes classroom

## Kubernetes installation tools:

Minicube

Kubeadm

## Kubernetes can be used for containers to:

1. Orchestrate
2. Manage
3. Auto scale (vertical (increasing resources i.e. ram, storage)/ horizontal [preferable (increasing number of containers)]
4. Load balancing
5. Be highly available
6. automatically deploy, schedule and runs.
7. Containers could run anywhere i.e. VMs, on prem/cloud machines.
8. Auto heal
9. Fault tolerance (Node/ Pod failure)
10. Rollback (going back to previous version)
11. Health monitoring of container
12. Batch execution (one time, sequential, parallel)

## Kubernetes VS Docker Swarm



## Kubernetes Architecture:



# Architecture & its Demo: #2(46)



## Master (Control Plane) & Node/ Slave/ worker

#### Components of control plane:

* Kube- api server
* Etcd
* Kube schedular
* Control Manager

#### Kube- api server (for all communication purposes):

* This server interacts directly with user i.e. we apply yml or json manifests to this server.
* This server is meant to scale automatically as per load.
* This server is the frontend of control-plane.

#### ETCD:

* Store metadata and status of cluster.
* It is consistent and highly available store (key value store)
* Source of touch for cluster state (information about source of cluster)

#### Features:

Fully Replicated, secured and fast.

#### Kube -Scheduler:

* When user make request for the creation & management of pods, this scheduler is going to take action on these requests.
* This will create the pods intelligently on different nodes, i.e. if you have three nodes, ask scheduler to create pod but didn’t specify the node at which you want to create the pod the kube-schedular will decide on its own by finding the best suitable node for the pod to create on i.e. if you already have pods created on 2 nodes out of three and the third node have lesser or no pods as compared to the other 2 then this will find that node most suitable and will create that pod on this nod. (Ref@46:00)
* It will get the information for hardware configuration from configuration files and schedules the pods on nodes accordingly.

## Control Manager: 46:17

* Make sure that desired state of cluster matches the actual state.
* Cloud-controller-manager is for cloud and kube-controller manager is for non cloud.

#### Components of control manager:

* Node-controller: Have all the information related to nodes.
* Route-controller: Responsible for setting up network.
* Service-controller: Responsible for load balancing
* Volume-controller: Responsible for creating, attaching and mounting the volumes and interacting with the cloud providers to orchestrate volumes.

## Node/ Slave/ worker side:

#### Kubelet:

* Agent that runs on nodes to manage pods, listen to kubernetes master remain connected with control manager on the master server side through APIs (e.g pod creation request).
* Use port 10255
* Send success/ fail reports to master.

#### Container Engine: (Docker, Rocket, container D etc)

* Work with kubelet
* Pull images
* Start/ stop containers
* Exposing containers on ports specified in manifests.

#### Kubeproxy:

* Responsible for networking (i.e. assign ips(dynamic, means IP will get change once the pod get restarted) to pods).
* It runs on each node and make sure that each pod will get its unique ip address.
* Containers do not have ips while pods have.
* Pods do not communicate with each other. It is recommended that there should be single container in a pod because multiple containers in a pod are tightly coupled with each other and if a container failed other containers also get failed due to that one.

POD: 1:00:00

* Smallest unit in kubernetes.
* It is a group of one or more tightly coupled containers that are deployed together on the same host means same node.
* It is recommended that a pod should only have one container.
* A cluster is a group of nodes should contain minimum of a master and a slave/ worker/ node.
* In k8 control unit is the pod and not the container.
* Pod runs on node controlled by master.

#### Issues with multi container Pods:

* We usually need multiple containers in a pod when we need to run dependent services.
* Shared memory space.
* Connected to each other using localhost. 1:03:17
* If a single container gets failed in multiple container situation, then it will get the whole pod failed.
* Entire pod is hosted on the same node (scheduler will decide which node). 1:06:17

#### Pod limitations & solutions:

* Doesn’t have auto healing or auto scaling feature but can be added by using higher level kubernetes objects like replication set which provide autoscaling and autohealing.
* What if pod crashes so the solution for this also is higher level kubernetes objects like deployment which provides versioning and rollback.

#### Some other objects of higher level kubernetes:

* Service to provide static ip (non-ephemeral (outside node)) and perform other networking tasks for the pods.
* Volume for non-ephemeral (outside node) storage.

#### Few important commands depending upon the model.

* Kubectl for single cloud
* Kubeadm for on prem
* Kubefed for federated (multi cloud, hybrid)

# Setting up Master and Salve (aka minion/node/worker)/ Node: #3(47)

For this we need 3 instances (VMs). A master and 2 nodes(slaves), master should have minimum of 2 vcpus and 4 GB Ram. Install docker.io on all 3 instances.

We need to install https package for intra cluster (control plane to individual pods) communication.

Setup open GPG key using below link to establish communication for intra cluster. 12:20

Curl -s <https://packages.cloud.google.com/apk/doc/apt-key.gpg> | sudo apt-key add

nano /etc/apt/sources.list d/kubernetes list

deb <https://apt> kubernetes.io/kubernetes-xenial main

exit from nano by pressing ctrl+x, press capital Y and press enter

sudo apt update

apt install -y kubelet kubeadm kubectl kubernetes-cni

use kubeadm init on master node to initialize k8s cluster for bootstrapping master node(connecting master with nodes/ workers)

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/configmkdir -p $HOME/.kube

copy configuration to kube directory(in config file)

sudo cp -I /etc/kubernetes/adminconf

$HOME /.kube/config

Provide user permissions to config

chown $(id-u):$(id-g) $HOME/.kube/config

deploy flannel (using below link) node network for its repository path flannel is going to place a binary in each node

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml>

## Lab:

|  |
| --- |
| sudo apt update  sudo apt install docker.io  sudo apt install apt-transport-https  docker -v  service docker status  sudo systemctl enable docker  sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add  sudo nano /etc/apt/sources.list.d/kubernetes.list  sudo apt update  sudo apt install -y kubelet kubeadm kubectl kubernetes-cni  sudo kubeadm init  sudo mkdir -p $HOME/.kube  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  sudo chown $(id -u):$(id -g) $HOME/.kube/config  kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml  kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-old-manifests/kube-flannel-rbac.yml  kubectl get nodes |

## Configuring worker nodes: 1:05:10

## Lab:

|  |
| --- |
| sudo apt update  sudo apt install docker.io  sudo apt install apt-transport-https  docker -v  service docker status  sudo systemctl enable docker  sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add  sudo nano /etc/apt/sources.list.d/kubernetes.list  sudo apt update  sudo apt install -y kubelet kubeadm kubectl kubernetes-cni  sudo kubeadm join 10.0.0.5:6443 --token 55ajtd.6e1e7tnca5z8728l --discovery-token-ca-cert-hash sha256:ee4926bc8cf83369378491f205b15ea9dd785025b3326bc9b2e33a525bb45e6f |

Below Notepad file have all the above commands from its YouTube description.



## Installation of MiniKube: #3(48)

#### Kubernetes Objects:

Kubernetes uses objects to represent the state of cluster.

**An object can be:**

* Running containerized applications.
* Policies about application behavior such as restart policies, upgrades and fault tolerance.
* Once you create the object, the Kubernetes system will constantly work to ensure that object remain exist and maintains cluster’s desired state.
* Every Kubernetes object includes two nested fields that govern the object configuration the object **specification** and the object **status.** 7:40
* **Specification** is provided by us which describe our desire for the object – the characteristics that we want an object to have.
* **Status** on the other hand describe the actual state of the object and is supplied and updated/maintained by the Kubernetes system. 9:10
* All objects are identified by a unique name and a uid.
* We will create objects and push these Kubernetes API with kubectl.

Basic Kubernetes object include following:

1. Pod
2. Service
3. Volume
4. Namespace
5. Replica sets
6. Secrets
7. Config maps
8. Deployments
9. Jobs
10. Daemon sets

Few details about these objects:

* **Pod** manages containers
* **Replicasets** manage pods
* **Services** expose pod processes to the outside world.
* **Configmaps** and **secrets** help you configure pods.

#### States in which a Kubernetes object can be: 15:00

Replicas (2/2) Image (Tomcat/ Ubuntu)

Name Port

Volume Startup

Detached (default)

#### Kubernetes Objects Management: 19:00

The kubectl command line tool supports several ways to create and manage Kubernetes objects

|  |  |  |
| --- | --- | --- |
| **Management Technique** | **Operations** | **Recommended Environment** |
| Imperative Commands | Live Objects | Development projects |
| Declarative object configuration | Individual files (Yml/Json) | Production |

**Declarative** is about describing what you are trying to achieve, without instructing how to do.

**Imperative,** explicitly tells how to accomplish it.

#### Fundamental of Pods: 20:00

* When a pod gets created, it is scheduled to run on a node in your cluster.
* The pod remains on that node until the process is terminated, the pod or pod object is deleted, the pod is evicted for lack of resources or the node itself gets failed.
* If a pod gets failed due to some reason, then a new pod will replace that pod with a new UID but with the same specs as the old pod have.
* Volume related to a pod will exist as long the pod exists, if the pod gets deleted or failed, new volume will also get created with pod.
* Different controller/ controller APIs can be used for handling replication, rollout and providing self-healing capabilities.

## Kubernetes configuration types:

#### All-in-one single node installation:

With this configuration type, all the master and worker component are installed on a single node. This is very useful for learning, development and testing but this should not be used in production. Minikube is an example of it.

#### Single node etcd, single master and multi worker installation:

#### Single node etcd, multi master and multi worker installation:

High availability is due to multiple masters

#### How to write manifests using YAML files: 31:00

--- to start a yaml file but optional, # is for comments, indentation/ alignment in yaml file is much more important will be key: value pair.

Lab explanation: 43:00

## Lab:

|  |
| --- |
| sudo apt update  sudo apt install docker.io  curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl && chmod +x ./kubectl && sudo mv ./kubectl /usr/local/bin/kubectl  which kubectl  kubectl version  curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/local/bin/  which minikube  sudo apt install conntrack  ~~minikube start --vm-driver=none~~  which conntrack  ~~docker system prune~~  ~~sudo docker system prune~~  ~~minikube delete~~  sudo minikube start --force --driver=docker  minikube is not starting using above commands needs RND. |

