Kubernetes

Role in

Devops

Table of Contents

- ✓ Introduce Docker and Docker Swarm
- ✓ Introduce Kubernetes as a Orchestration Concept and Architecture
- ✓ Core Concept Kubernetes Objects
- ✓ Swarm vs Kubernetes
- ✓ Kubernetes Installation Methods
- ✓ kubectl Commands
- ✓ Introduce Manifest and YAML and Write it
- ✓ Kubernetes Volume Storge
  - ✓ StorageClass (CephFS , Glusterfs , NFS , iSCSI , Cinder , Local , …)
  - ✓ Persistent Volumes
  - ✓ Persistent Volumes Claim
- ✓ Kubernetes Network About Container Network Interface (CNI)
  - ✓ (WeaveNet , Flannel , Contiv , Cilium , Calico , …)
- ✓ Kubernetes Scaling
  - ✓ Horizontal Infrastructure Scaling
  - ✓ Vertical Infrastructure Scaling
  - ✓ Auto-scaling
- ✓ Implementing Deployment With Kubernetes
  - ✓ Deployment Nginx With Scale On kubernetes
- ✓ Introduce Kubernetes Exam Certified Kubernetes Administrator (CKA) Details

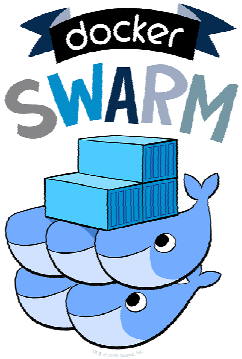
Docker and Docker Swarm is

Docker is :

A Tool designed to make ,

Easier to create
Deploy
And Run

Applications by using **Containers**



Docker Swarm is :

An open-source container Orchestration platform and
The Native Clustering Engine by **Docker**

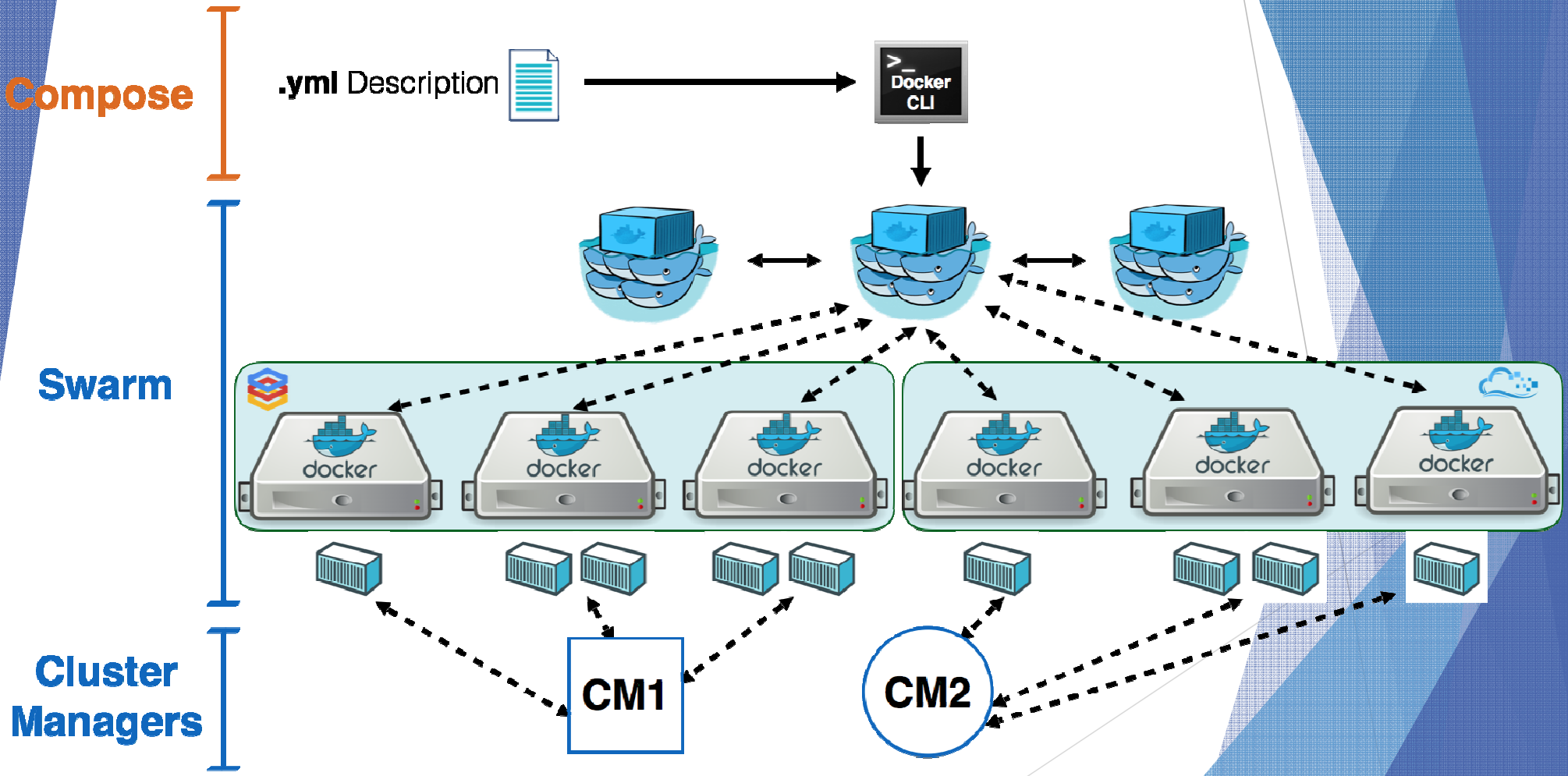
Containers is :

A Package from applications ,
Can be run, with its dependencies, isolated from other processes .

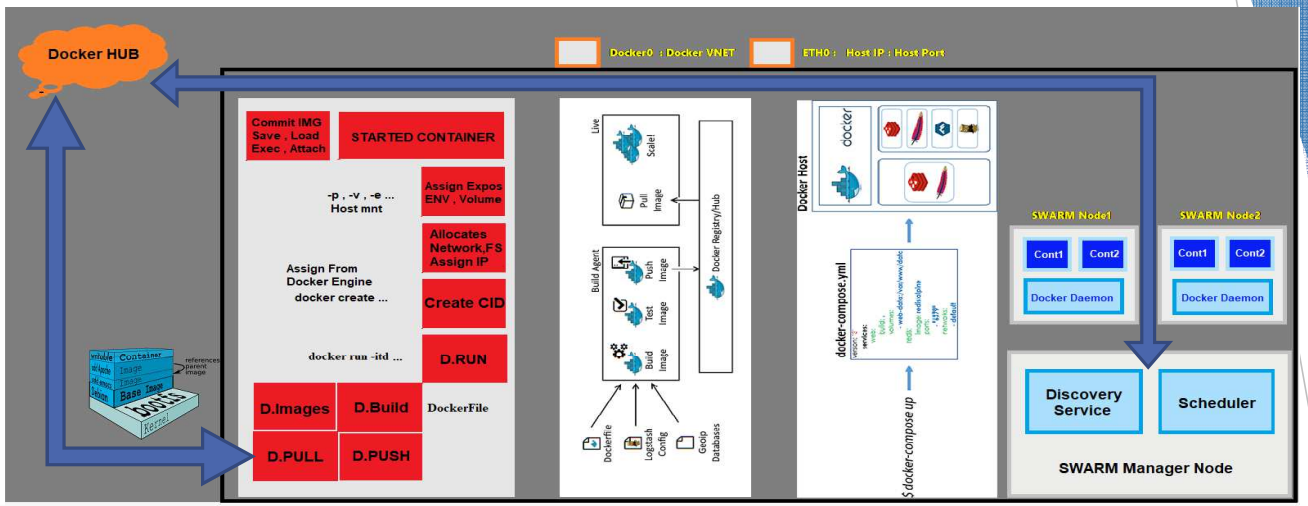
Banner Docker :

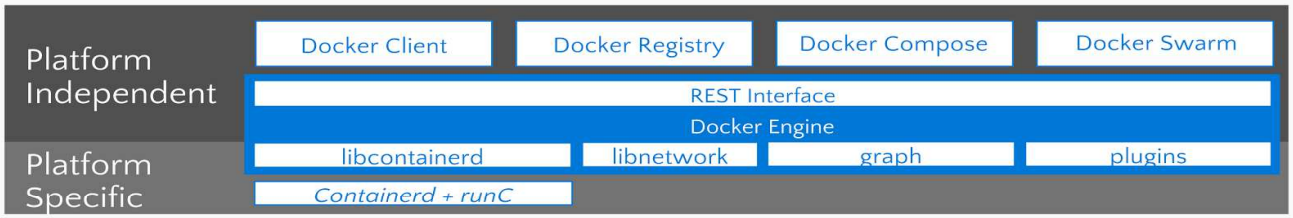
- **Build Any App**
- **Ship Any Where**
- **Run Any Where**

Docker Swarm Overview



Docker Swarm Components and Architecture





Kubernetes is …

Kubernetes is :

Desigen by Google be name K8s.

Automated Container Deployment , Scaling and Management Platform

Kubernetes is :

An open source system for managing containerized application in a clustered environment.

Kubernetes help to :

- ✓ DevOps as a Service
- ✓ Automatically scale up-down the application and update
- ✓ With the zero downtime

kubernetes

Kubernetes Version

Original author(s): Google

Developer(s) : Cloud Native Computing Foundation

Initial release : 7 June 2014; 5 years ago

Stable release : 1.16 / September 18, 2019; 2 months ago

kubernetes

Repository : github.com/kubernetes/kubernetes

Written in : Go

Type : Cluster Management Software

License : Apache License 2.0

Website : kubernetes.io

Kubernetes History

kubernetes

Kubernetes Version

Features and Preferences Of Kubernetes :

- ✓ **Service Discovery and Networks , Load Balancing**
- ✓ **Storage Orchestration**
  - NFS ،iSCSI ،Gluster ،Ceph ،Cinde
  - AWS Cloud
- ✓ **Automated Rollouts and Rollbacks , Rolling Update**
- ✓ **Scaling And Monitoring**
  - Horizontal
  - Vertical
  - Auto Scale
  - Manual Scale

Kubernetes Version

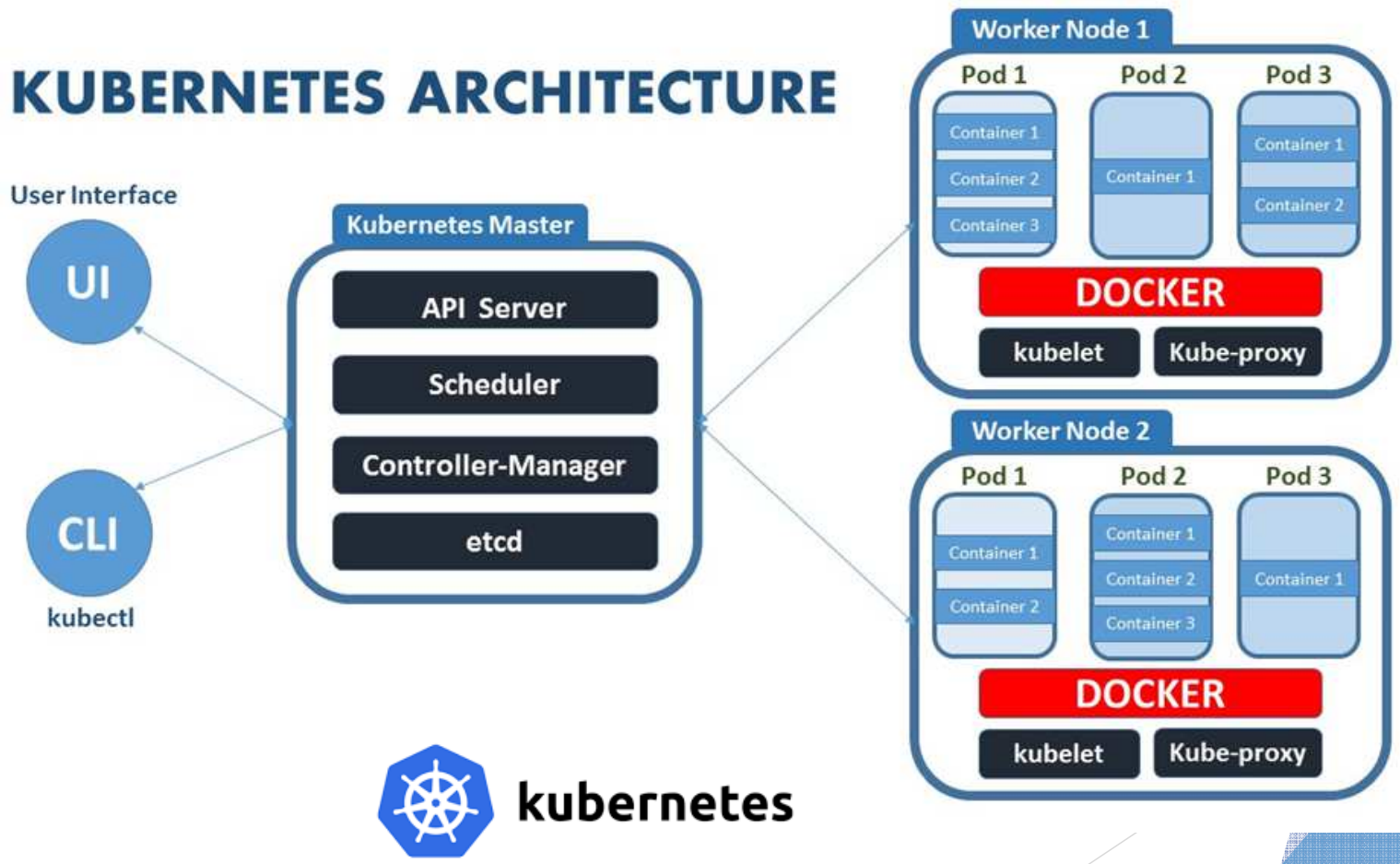
Glossary Of Kubernetes :

- ✓ **Cluster**
  - ✓ Kubernetes Engine With Serveral Server (Physical Or Virtual)For Support All Components
- ✓ **Node**
- ✓ **Pod**
- ✓ **Replication Controllers**
- ✓ **Replica Sets**

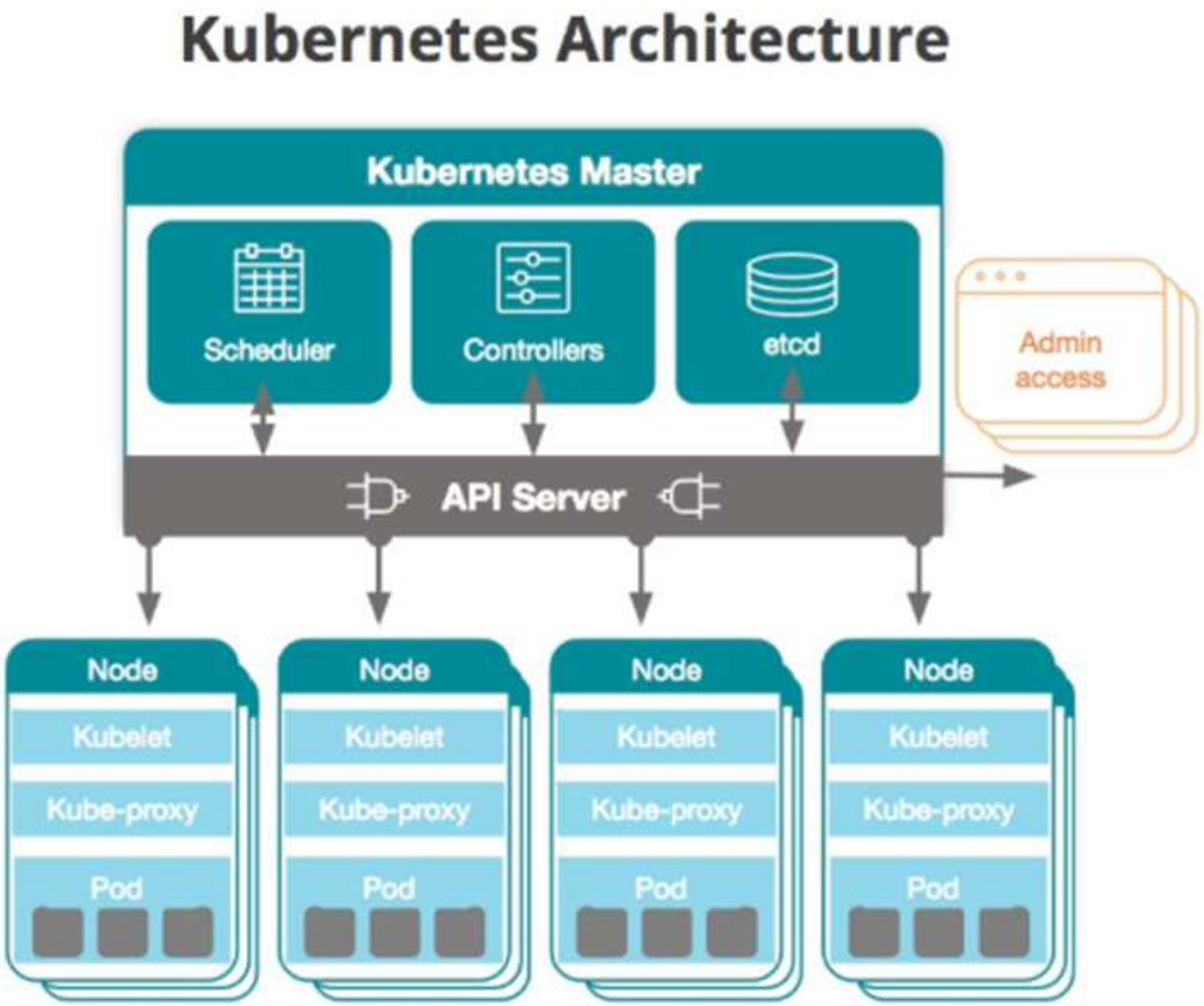
  } Run a specified number of Pod
- ✓ **Selector**
- ✓ **Label**

  } Filter for Label of Pod
- ✓ **Name**
- ✓ **Volume**
- ✓ **Namespace** ➔ Domain Name
- ✓ **Service** ➔ include (IP , DNS Name)
- ✓ **Kind**

Kubernetes Overview



Kubernetes Components and Architecture



Kubernetes Components and Architecture

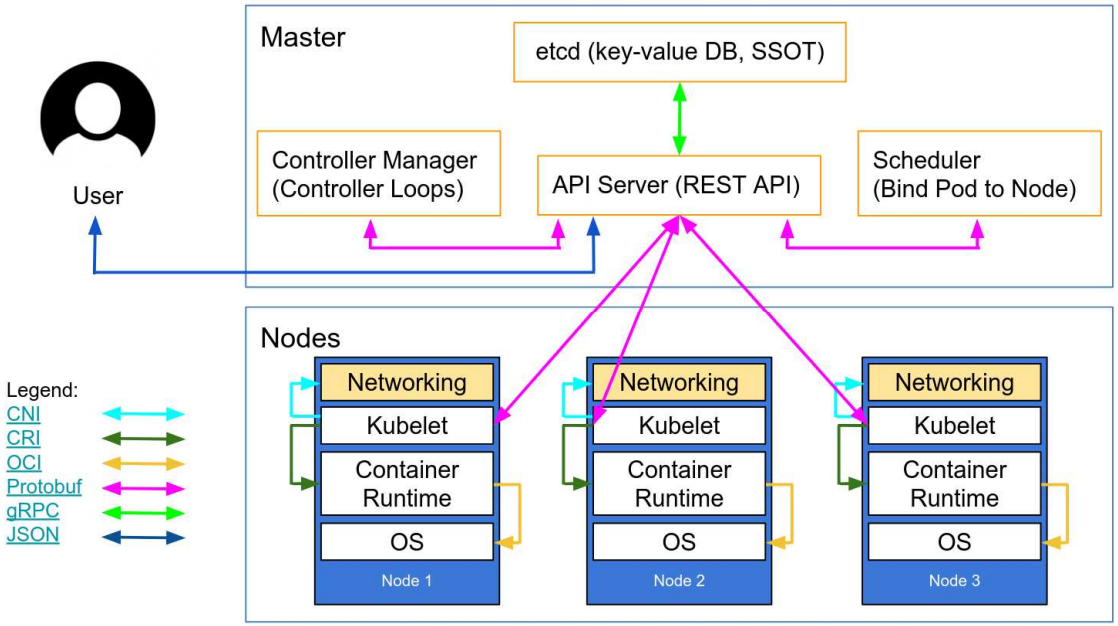
Kubernetes Components :

- ✓ **Kubernetes - Master Machine Components**
  - Etcd
  - API Server
  - Controller Manager
  - Scheduler
- ✓ **Kubernetes - Worker Node Components**
  - Docker
  - Kubelet Service
  - Kubernetes Proxy Service
- ✓ **Kubernetes Objects and Workloads**
  - Pods
  - Replication Controllers
  - Replication Sets
  - Deployments
  - Stateful Sets
  - Daemon Sets
  - Services

Kubernetes Components and Architecture

Kubernetes - Master Machine Components

- ✓ The Kubernetes Master is Responsible for Maintaining the Good State for your Cluster.
- ✓ Interact with Kubernetes, by kubectl command-line interface, you're communicating with your cluster's Kubernetes master.
- ✓ Typically all these processes run on a single node in the cluster, and this node is also referred to as the master.
- ✓ The master can also be replicated for Availability and Redundancy
- ✓ The master server acts as the **Primary Control Plane** for Kubernetes Clusters.



Kubernetes Components and Architecture

Kubernetes - Master Machine Components

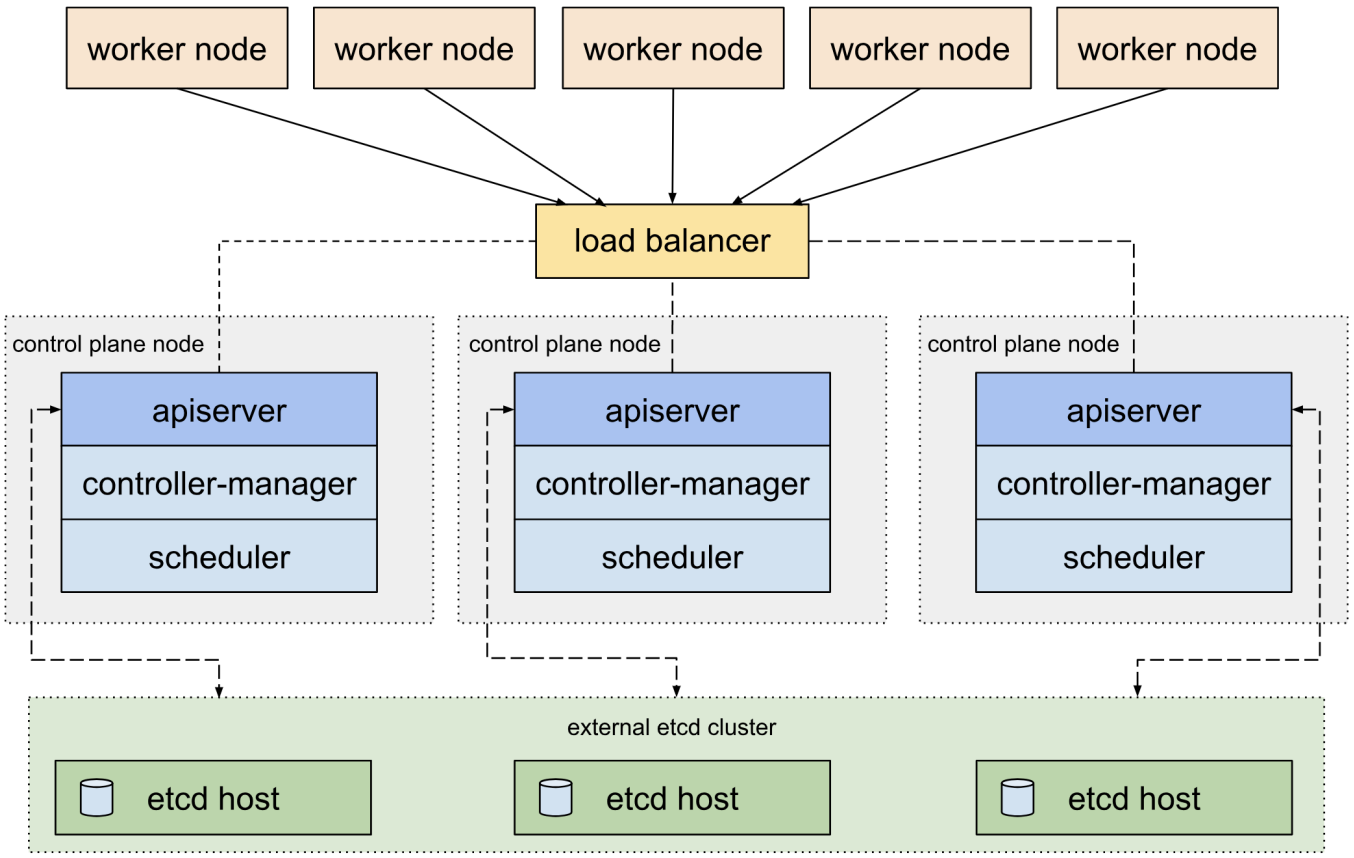
❖ Etcd :

- ✓ **Create By CoreOS**
- ✓ **Very Small And Thin And Distributed**
- ✓ **Display the overall status of the cluster**

- ✓ Consistent and **Highly-Available key value store** used as Kubernetes' **backing store** for all cluster data.
- ✓ If your Kubernetes cluster uses etcd as its backing store, make sure you have a **back up plan** for those data.

Kubernetes Components and Architecture

Kubernetes Multi Master Architecture



Kubernetes Components and Architecture

Kubernetes - Master Machine Components

❖ API Server :

- ✓ Exposes the Kubernetes API. It is the **Front-End** for the Kubernetes **Control Plane**.
- ✓ The main implementation of a Kubernetes API server is kube-apiserver.
- ✓ kube-apiserver is designed to scale horizontally.
- ✓ That is, it scales by deploying more instances.
- ✓ You can run several instances of kube-apiserver and balance traffic between those instances.

Kubernetes Components and Architecture

Kubernetes - Master Machine Components

❖ Controller Manager :

- ✓ For Manage Workload life cycles, and Perform Routine Tasks.
- ✓ Component on the master that runs Controllers and each Controllers is a separate process .
- ✓ These controllers include:
  - ➢ Node Controller : For Hosts Checking
  - ➢ Replication Controller: For Pods Checking
  - ➢ Endpoints Controller: Join Services and Pods
  - ➢ Service Account & Token Controllers : Create default accounts and API access tokens for new namespaces.

Kubernetes Components and Architecture

Kubernetes - Master Machine Components

❖ **Scheduler :**

✓ Component on the master that watches **Newly Created Pods** that **have NO Node Assigned**, and select a node for them to **run on**.

✓ **Scheduling Factors :**

individual and collective resource requirements
hardware/software/policy constraints
data locality
inter-workload

Kubernetes Components and Architecture

Kubernetes - Worker Node Components

- ✓ **A Node is a worker machine in Kubernetes, Previously known as a Minion.**
- ✓ **A Node may be a VM or physical machine, depending on the cluster.**
- ✓ **Each node contains the services necessary to run pods and is managed by the master components.**
- ✓ **The services on a node include :**
  - ❑ kubelet
  - ❑ The container runtime (Docker)
  - ❑ kube-proxy

Finaly , The servers that perform work by Running Containers are known as node

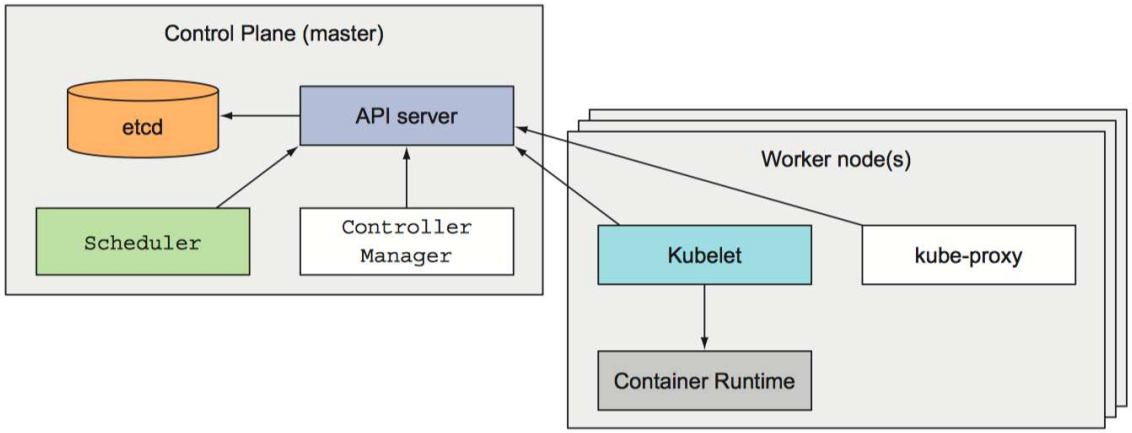


Figure 1.9 The components that make up a Kubernetes cluster

Kubernetes Components and Architecture

Kubernetes - Worker Node Components

- **Kubelet Service :**
  - ✓ **The kubelet service communicates with the master components to authenticate to the cluster and receive commands and work.**
  - ✓ **This service is responsible for relaying information , and from the control plane services.**
  - ✓ **Interacting with the etcd store to read configuration details or write new values.**
  - ✓ **It controls the container runtime to launch or destroy containers as needed.**

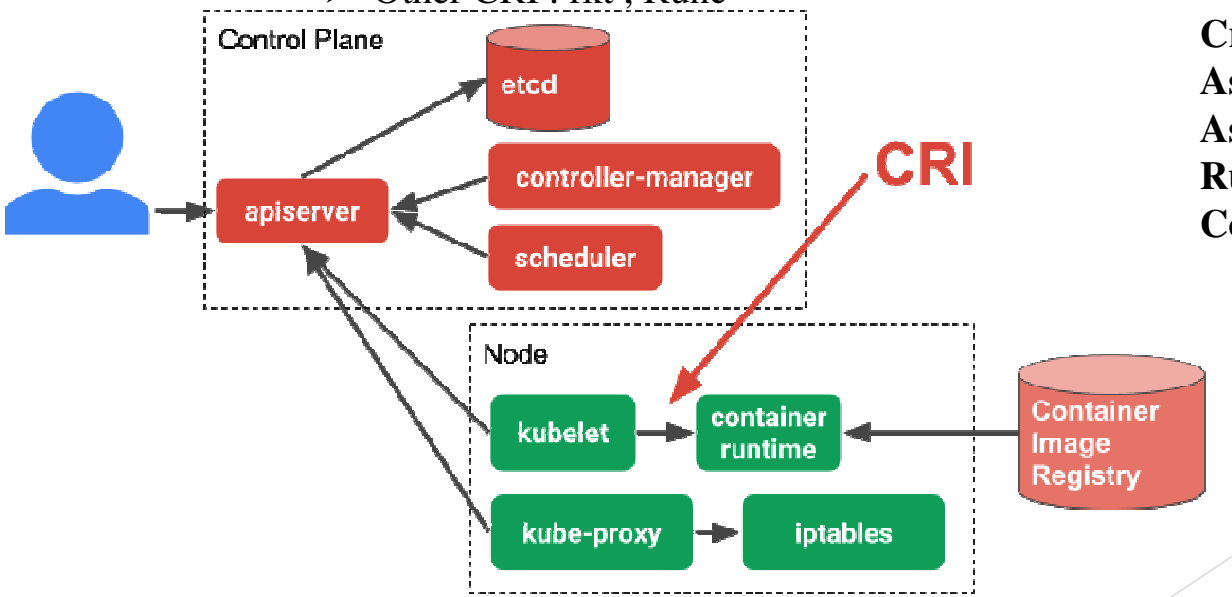
.

Kubernetes Components and Architecture

Kubernetes - Worker Node Components

❖ A Container Runtime Interface (CRI):

- ✓ For run containers in Pods, Kubernetes uses a Container Runtime (CRI)
- ✓ Here are the installation instructions for various runtimes:
  - ➢ Docker
  - ➢ CRI-O
  - ➢ Containerd
  - ➢ Other CRI : rkt , Runc



Container Runtime (CRI) Flow :

Pull Image
Create Container ID
Assign Network ID to Container
Assign file system and Cgroup to Container
Run Container
Connect To Container

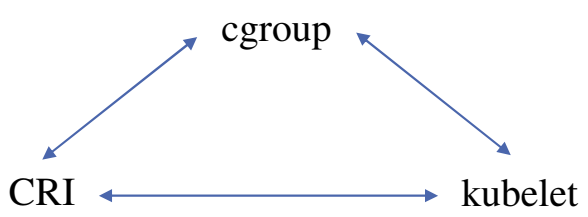
Kubernetes Components and Architecture

Kubernetes - Worker Node Components

A Container Runtime Interface (CRI):

Cgroup Drivers :

- ✓ When systemd is chosen as the init system for a Linux distribution.
- ✓ The init process generates and consumes a root control group (cgroup) and acts as a cgroup manager.
- ✓ Systemd has a tight integration with cgroups and will allocate cgroups per process
- ✓ It's possible to configure your container runtime and the kubelet to use cgroupfs



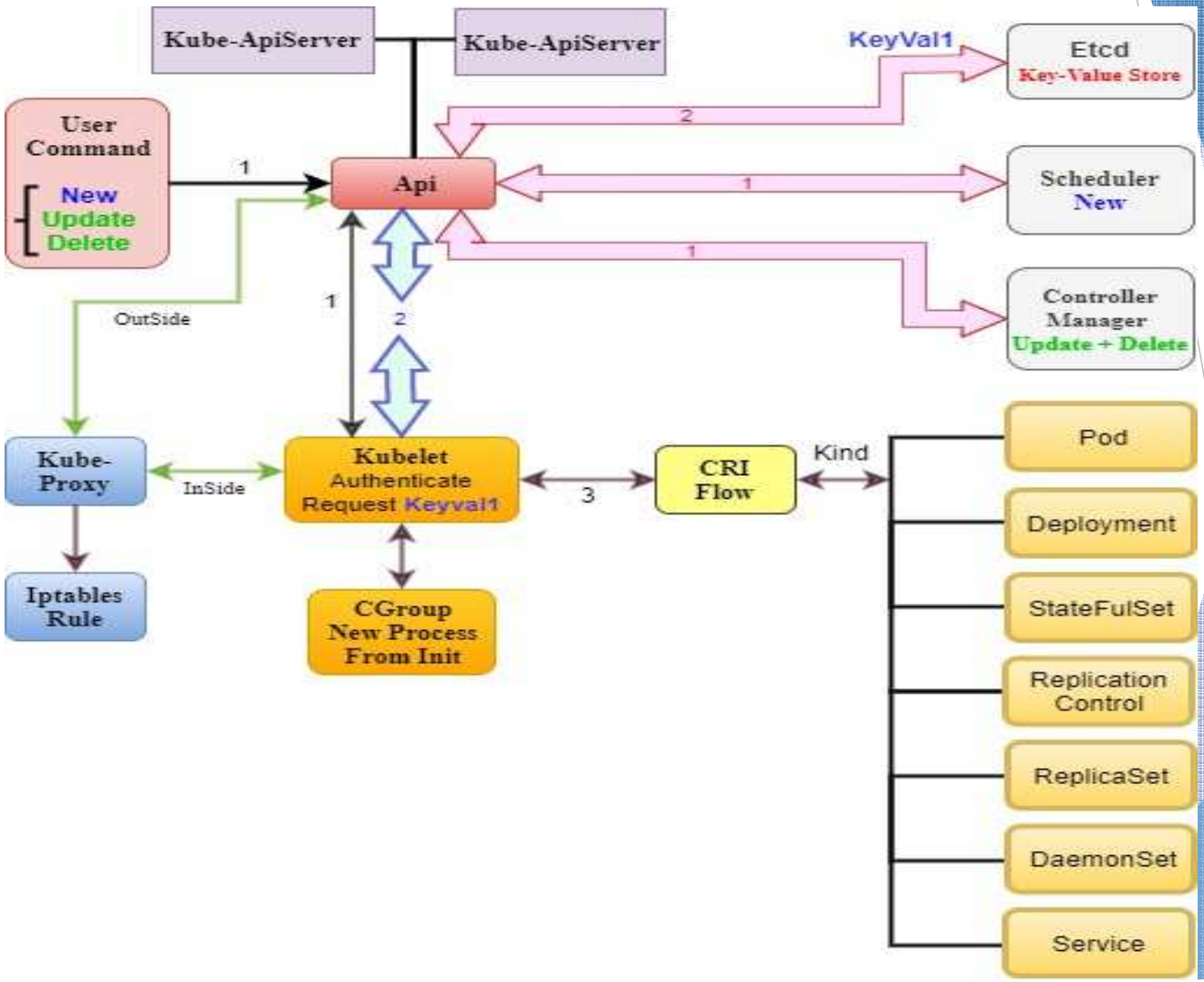
Kubernetes Components and Architecture

Kubernetes - Worker Node Components

❖Kubernetes Proxy Service :

- ✓ **Create and Manage networking environment .**
- ✓ **kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service A way to expose an application running on a set of Pods as a network service concept.**
- ✓ **kube-proxy maintains network rules on nodes.**
- ✓ **These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.**
- ✓ **kube-proxy forwards the traffic itself.**

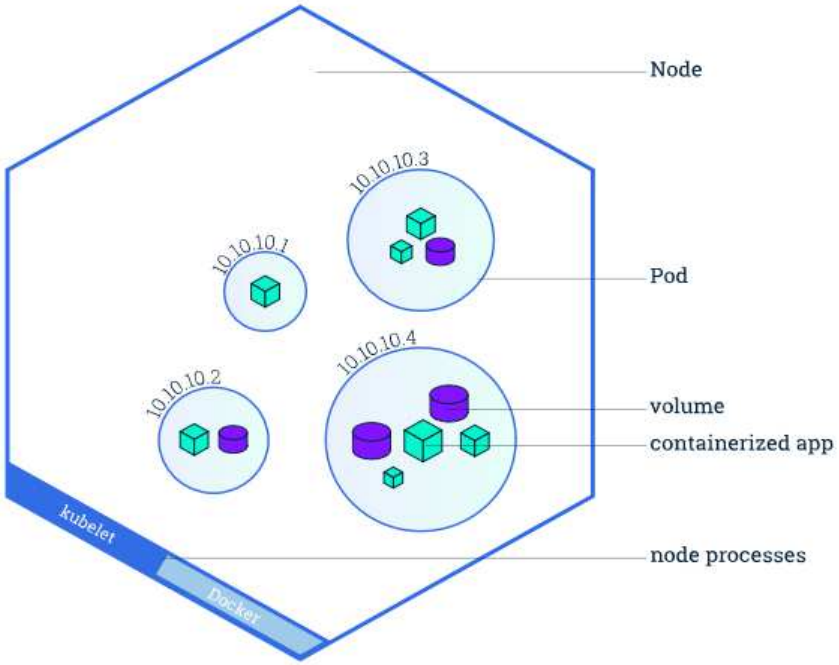
Kubernetes Components and Architecture



Kubernetes Components and Architecture

Kubernetes Objects and Workloads

- **Pods** **kind: Pod**



Kubernetes Components and Architecture

Kubernetes Objects and Workloads

❖ Pods

- ✓ A Pod is a group of one or more containers. (such as Docker containers), with shared storage/network, and a specification for how to run the containers.
- ✓ A **pod** is the most Basic Unit that Kubernetes deals with.
- ✓ Containers themselves are not Assigned to hosts.
- ✓ Instead, one or more tightly coupled containers are encapsulated in an object called a pod.
- ✓ Containers within a Pod **share an IP address and port space**, and **can find each other via localhost (Pod Name)**
- ✓ Containers in different Pods have distinct IP addresses and can not communicate by IPC without special configuration. These containers usually communicate with each other via Pod IP addresses.
- ✓ In terms of Docker constructs, a Pod is modelled as a group of Docker containers with shared namespaces and shared filesystem volumes.

Kubernetes Components and Architecture

Kubernetes Objects and Workloads

Termination of Pods:

Because Pods represent Running Processes on nodes in the cluster.

Then Nice Terminate Beter Kill Terminate

Default grace period (30s) in Nice Terminate

When a user requests a pod to be removed :

1. **the system sending a TERM signal to the main process in each container.**
2. **After the grace period ends, the KILL signal is sent to those processes,**
3. **And then the Pod is removed from the API server**

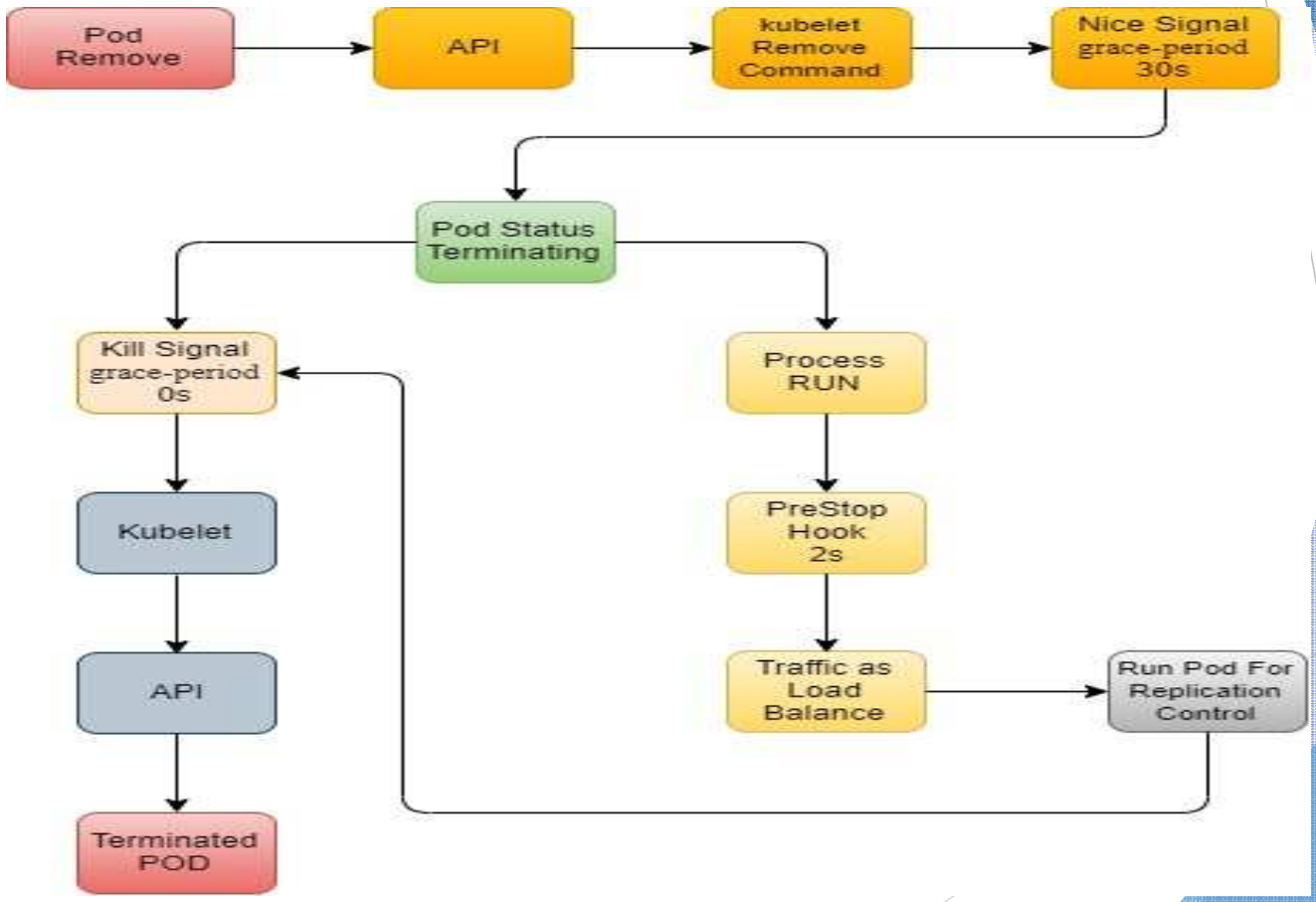
Note : Default grace period (0s) in Kill Terminate

--grace-period=0 --force remove pod

Kubernetes Components and Architecture

Kubernetes Objects and Workloads

Termination of Pods Flow:



Kubernetes Components and Architecture

Kubernetes Objects and Workloads

Termination of Pods Flow:

Container hooks :

There are Two hooks that are exposed to Containers:

- PostStart
- PreStop

- **PostStart**
  - ✓ This hook executes immediately after a container is created.
  - ✓ However, there is no guarantee that the hook will execute before the container ENTRYPOINT. No parameters are passed to the handler.

- **PreStop**
  - ✓ This hook is called immediately before a container is terminated due to
    - ➢ An API request or management event such as liveness probe failure, resource contention and others
    - ➢ A call to the preStop hook fails if the container is already in terminated or completed state

Kubernetes Components and Architecture

Kubernetes Objects and Workloads

❖ **Replication Controllers :** **kind: ReplicationController**

Is Defines

A pod template and control parameters

to scale Number of replicas = Number of pod horizontally

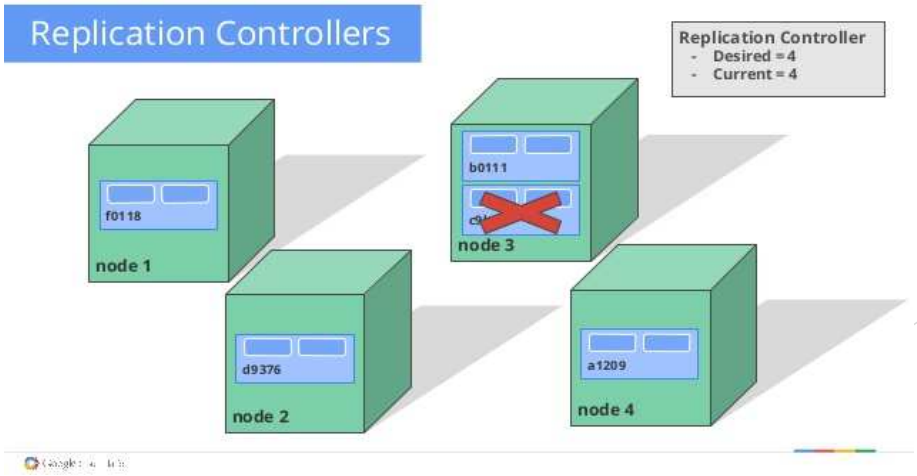
IF Change Replica :

Increasing = Copies and Running Pods

Decreasing = Teminate and Kill pods

If a pod or a host fails , the controller will start new pods to compensate

Replication controllers can also perform rolling updates



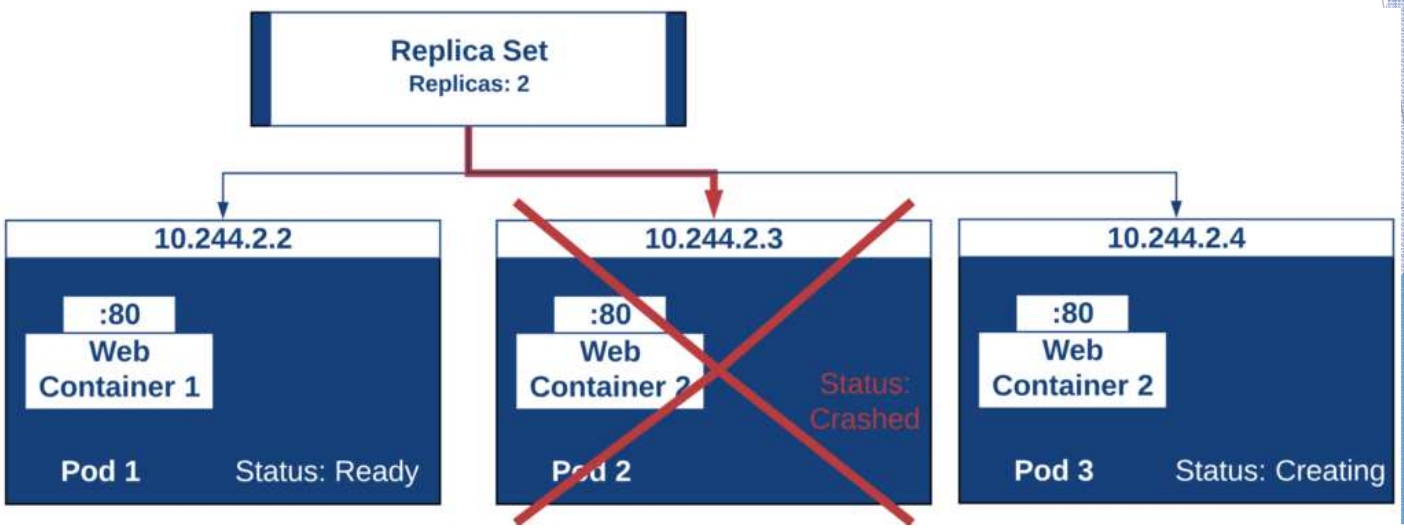
Kubernetes Components and Architecture

Kubernetes Objects and Workloads

❖ Replication Sets : kind: ReplicaSet

- ReplicaSet is the Next-Generation ReplicationController
- Replication sets are Replace Replication Controllers
- Because , greater flexibility to manage pods and supports the new set-based label selector.
- It's mainly used by Deployment as a mechanism to orchestrate pod creation, deletion and updates.
- Note that we recommend using Deployments instead of directly using Replica Sets

```
replicas: 3
selector:
  matchLabels:
    tier: frontend
template:
  metadata:
    labels:
      tier: frontend
```



Kubernetes Components and Architecture

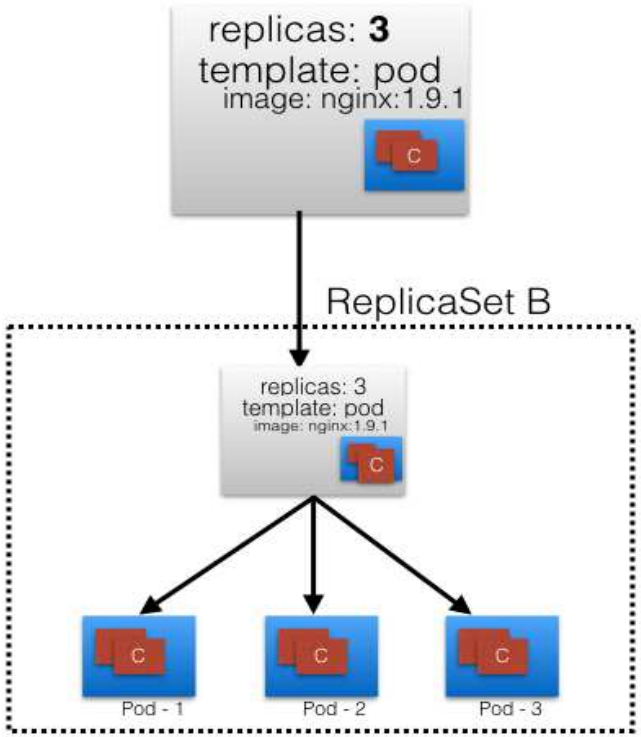
Kubernetes Objects and Workloads

Deployment

❖ **Deployments :** **kind: Deployment**

Deployments use

Replication sets as a building block
Replication controllers
Rolling update
Adding flexible life cycle management functionality to the mix.



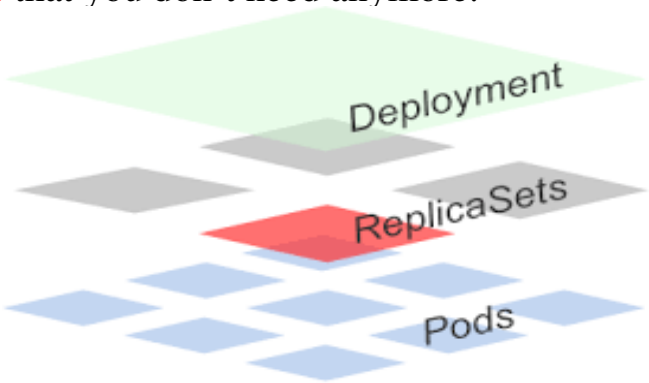
Kubernetes Components and Architecture

Kubernetes Objects and Workloads

Deployments Use Case :

The following are typical use cases for Deployments:

- ✓ Create a Deployment to rollout a ReplicaSet. The ReplicaSet creates Pods in the background. Check the status of the rollout to see if it succeeds or not.
- ✓ Rollback to an earlier Deployment revision if the current state of the Deployment is not stable. Each rollback updates the revision of the Deployment.
- ✓ Scale up the Deployment to facilitate more load.
- ✓ Pause the Deployment to apply multiple fixes to its PodTemplateSpec and then resume it to start a new rollout.
- ✓ Clean up older ReplicaSets that you don't need anymore.

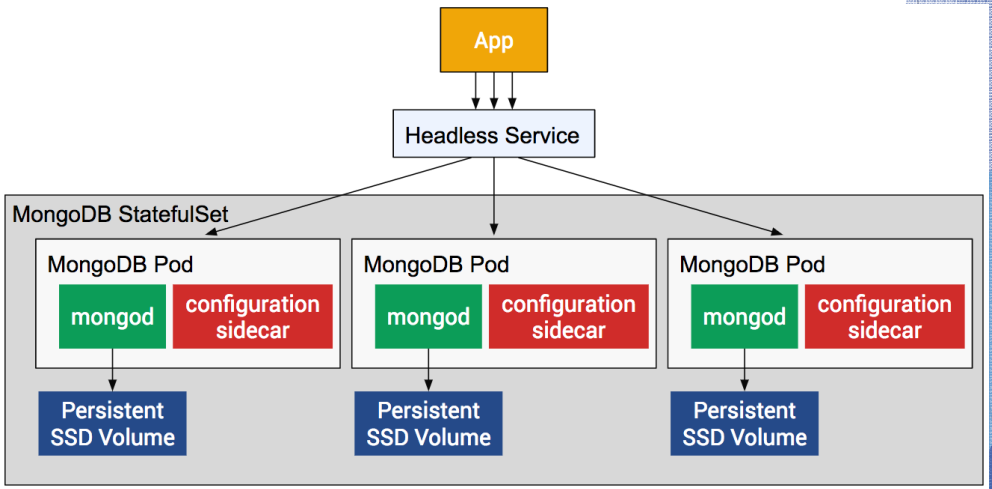


Kubernetes Components and Architecture

Kubernetes Objects and Workloads

❖ Stateful Sets : kind: StatefulSet

- Stateful sets are often associated with data-oriented applications
  like databases,
  which need access to the same volumes even if rescheduled to a new node.
- Deleting and/or scaling a StatefulSet down will not delete the volumes associated with the StatefulSet. This is done to ensure data safety.
- Stable, unique network identifiers.
  StatefulSets currently require a Headless Service to be responsible for the network identity of the Pods

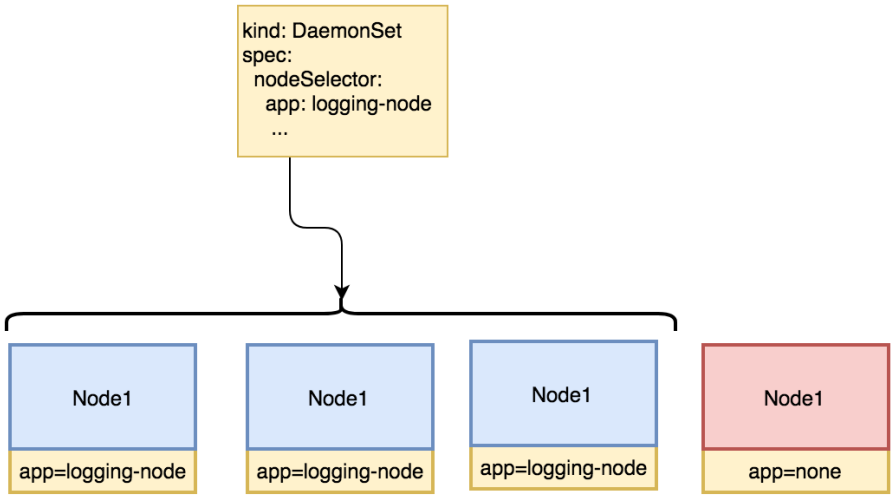


Kubernetes Components and Architecture

Kubernetes Objects and Workloads

Daemon Sets : **kind: DaemonSet**

- A DaemonSet ensures that all (or some) Nodes run a copy of a Pod. **As nodes are added to the cluster, Pods are added to them.**
- Run a copy of a pod on each node in the cluster ➔ 1 Pod in 1 Node
- Some typical uses of a DaemonSet are:
  - ✓ running a cluster storage daemon, such as glusterd, ceph, on each node.
  - ✓ running a logs collection daemon on every node, such as fluentd or logstash.
  - ✓ running a node monitoring daemon on every node, such as Prometheus Node Exporter, Sysdig Agent, collectd, Dynatrace OneAgent, New Relic agent, Ganglia gmond or Instana Agent.



Kubernetes Components and Architecture

Kubernetes Objects and Workloads

❖ **Services :** **kind: Service**

- ✓ Service is network connected
- ✓ Service is a component as a basic internal load balancer and NodePort for pods
- ✓ An abstract way to expose an application running on a set of Pods as a network service.
- ✓ With Kubernetes you don't need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.

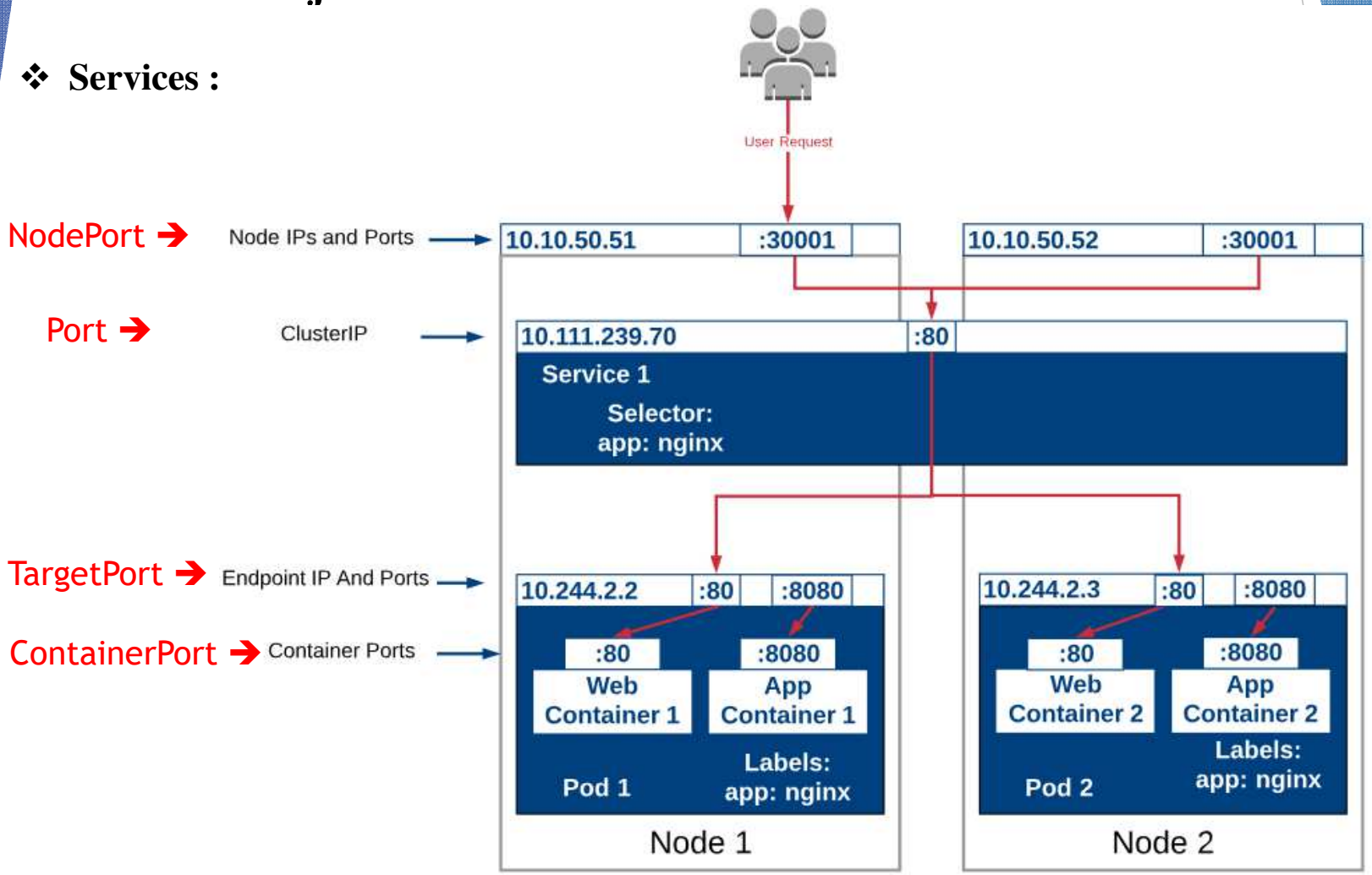
Note : A Service can map any incoming Port to a TargetPort

- ➢ Port ➔ Inside Cluster
- ➢ TargetPort ➔ Inside Pod and Container

Kubernetes Components and Architecture

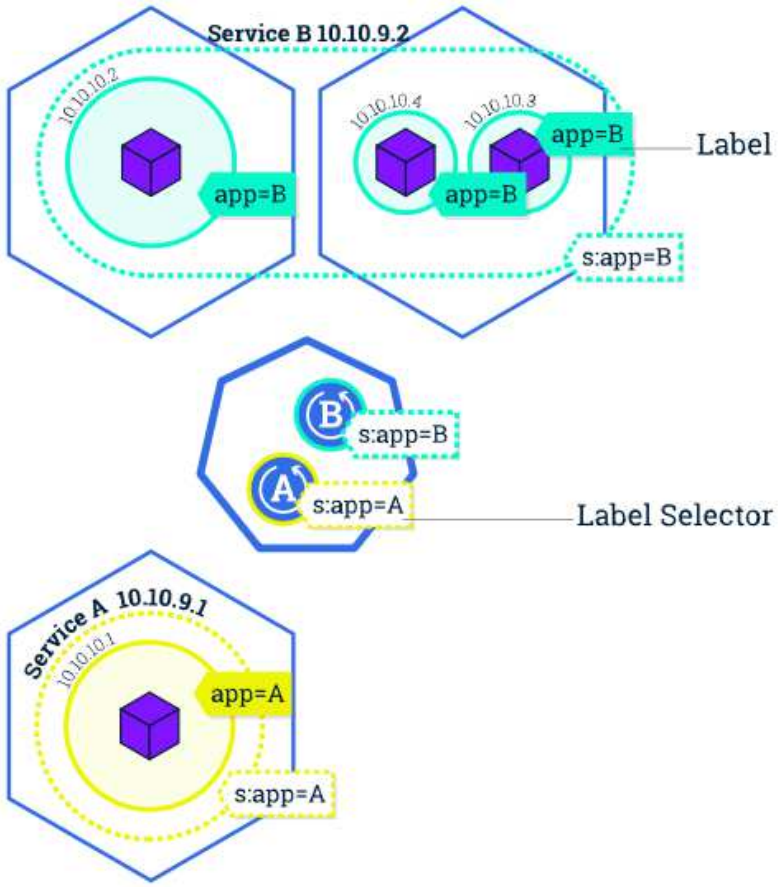
Kubernetes Objects and Workloads

❖ **Services :**



Kubernetes Installation Method

Label And Selector :



Kubernetes vs Docker Swarm: Pros and Cons

Pros of using Docker Swarm :

- **Runs at a faster pace**
- **Documentation provides every bit of information :** Last Version = Last Documentation
- **Ensures that application is isolated :** Container Configuration and Resource is Isolated
- **Provides simple and fast configuration :** Configuration, put it into a code and deploy it
- **Version control and component reuse :**

  Checking Old Version and Differences, IF Needed Roll-back

Cons of using Docker Swarm :

- **Docker is platform dependent : Widows and Linux Versions**
- **Doesn't provide storage option :**
- **Poor monitoring** :

  basic information about the container for the basic monitoring .
  for advance monitoring used with third-party

Kubernetes vs Docker Swarm: Pros and Cons

Pros of using Kubernetes :

- **Its Uptime :**

  The Goal of the Kubernetes is to update an application with a constant uptime

- **Immutable infrastructure :**
  - **For update any application (Deployment) you need :**
    - **Build container New Image with a new Tag and Deploy and Check It …**
    - **Killing the Old Container with Old Image version .**
    - **NOTE :** in-case if there is any Error, you can easily rollback to the previous image

Kubernetes vs Docker Swarm: Pros and Cons

Pros of using Kubernetes :

- **Deploy and update software at scale**

  Scaling is easy because used immutable infrastructure in kubernetes

  Kubernetes offers several useful features for scaling purpose**:**

  - ✓ **Horizontal Infrastructure Scaling**
  - ✓ **Vertical Infrastructure Scaling**
  - ✓ **Auto-scaling :**
    - Based on the Usage of CPU resources or other application-metrics,
    - You can change the number of containers then are running
  - ✓ **Manual Scaling :** With Command
  - ✓ **Replication Controller with Deployment**

Kubernetes vs Docker Swarm: Pros and Cons

Pros of using Kubernetes :

- **Handles the availability of the application:**

  Kubernetes checks the health of nodes and containers as well as provides self-healing and auto-replacement if in-case pod crashes due to an error

- **Storage Volume :**

  IF Pod is Deleted = Data is Deleted or Not Deleted

  IF Pod is Moved to Any Host = Data Stays there to be erased manually

Kubernetes vs Docker Swarm: Pros and Cons

Cons of using Kubernetes

- **The installation process is tedious**

- **Initial process takes time**

  Time to wait until the start of the new process

  A Pod can have multiple Containers running apps within it,

  but it can also have one or more Init Containers,

  which are run before the app Containers are started **(ReadinessProbe, LivenessProbe)**

  Note : If an Init Container fails for a Pod, Kubernetes restarts the Pod repeatedly until the Init Container succeeds

Kubernetes vs Docker Swarm: Pros and Cons

Cons of using Kubernetes

- Initial process takes time

✓ Liveness :

The kubelet uses liveness probes to know when to restart a Container.

For example, liveness probes could catch a deadlock, where an application is running, but unable to make progress. Restarting a Container in such a state can help to make the application more available despite bugs.

✓ Readiness

The kubelet uses readiness probes to know when a Container is ready to start accepting traffic.

A Pod is considered ready when all of its Containers are ready. One use of this signal is to control which Pods are used as backends for Services. When a Pod is not ready, it is removed from Service load balancers.

Kubernetes vs Docker Swarm: A Quick Summary

Compare Table :

	Docker Swarm	Kubernetes
Developed By	Docker Inc	Google
Year Released	2013	2014
Companies using	Bugsnag, Bluestem Brands, Hammerhead, Code Picnic, Dial once etc.	Asana, Buffer, CircleCI, Evernote, Harvest, Intel, Starbucks, Shopify etc.
Controller	Manager	Master
Storage	Volumes	Persistent and Ephermal
Installation	Easy to set up	Takes time for installation
Tolerance ratio	Low Fault Tolerance	High fault tolerance
Deployment unit	Task	Pod
Port	Published Port	Endpoint
Network	Overlay	Flat Networking space
Slave	Worker	Nodes
Container set up	Functionality is provided and limited by Docker API	Client API and YAML are unique in kubernetes

Kubernetes or Docker: Which Can be a Perfect Choice?

Use Kubernetes if :

You are looking For :

Perfect deployment and monitoring option .

Fast and Reliable response times

Develop a complex application and requires high resource computing without restrictions

You have a pretty big cluster

Use Docker Swarm if :

You are looking For :

Tool without spending much time on configuration and installation

Develop a basic and standard application which is sufficient enough with default docker image

You want docker API experience and compatibility

Final Result : Kubernetes and Docker Swarm

If you need to develop the complex app, use Kubernetes

If you are looking to develop the small-scale app, use Docker swarm

Kubernetes Installation Method

- Miniqube Installation
- Kubeadm Installation For CKA EXAM
- Kubespray Installation For Opertaion Environment

Kubernetes Installation Method

- **Miniqube Installation :**

Kubernetes Installation Method

- **Miniqube Installation :**

  Step 1: Install Docker

  Active CPU on VMWare

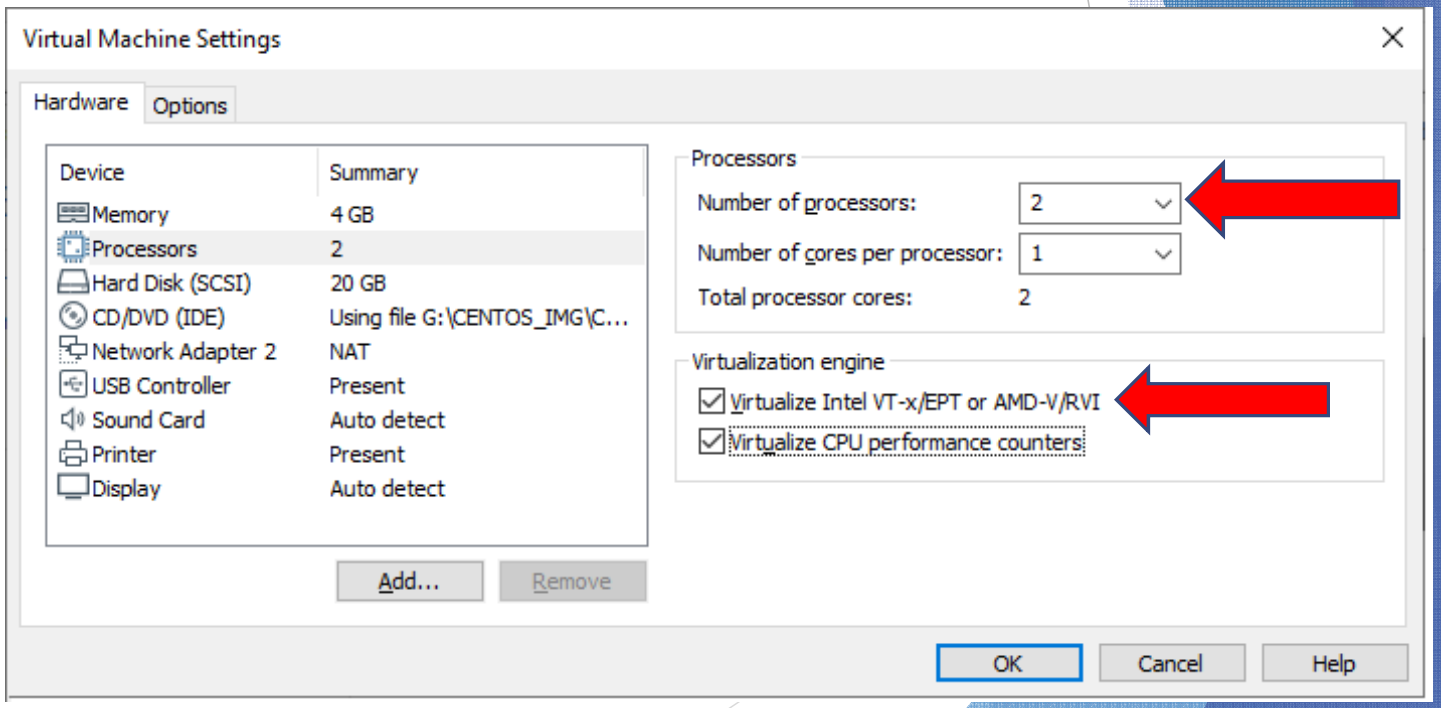
  1- CPU minimum >= 2

  2- VT-x Enable

Note :

Selinux Disable

IP Address Set



Kubernetes Installation Method

- **Miniqube Installation :**

Step 1: Install Docker

3- curl ipinfo.io


```
{
  "ip": "198.147.24.101",
  "city": "New York City",
  "region": "New York",
  "country": "US",
  "loc": "40.7143,-74.0060",
  "org": "AS55081 24 SHELLS",
  "postal": "10004",
  "timezone": "America/New_York",
  "readme": "https://ipinfo.io/missingauth"
}
```

Kubernetes Installation Method

- **Miniqube Installation :**

Step 1: Install Docker

4- swapoff -a

edit in /etc/fstab

5- vi /etc/fstab

#/dev/mapper/centos-swap swap swap defaults 0 0

6- top

swap is off

7- echo nothing 2>/dev/null 1>/dev/null || alias sudo='$@'

Kubernetes Installation Method

- **Miniqube Installation :**

Step 1: Install Docker

```
8- sudo tee /etc/yum.repos.d/docker.repo <<-'EOF'
[docker-ce-edge]
name=Docker CE Edge - $basearch
baseurl=https://download.docker.com/linux/centos/7/$basearch/edge
enabled=1
gpgcheck=1
gpgkey=https://download.docker.com/linux/centos/gpg
EOF
```

OutPut :

```
[docker-ce-edge]
name=Docker CE Edge - $basearch
baseurl=https://download.docker.com/linux/centos/7/$basearch/edge
enabled=1
gpgcheck=1
gpgkey=https://download.docker.com/linux/centos/gpg
```

```
9- yum install -y docker-ce-18.06.1.ce-3.el7.x86_64 \
  && sudo systemctl start docker \
  && sudo systemctl status docker \
  && sudo systemctl enable docker
```

TEST :

```
[root@localhost ~]# ll /etc/yum.repos.d/
-rw-r--r--  1 root root  188 Nov 16 05:26 docker.repo
```

Kubernetes Installation Method

- **Miniqube Installation :**

Step 2: Install Kubernetes Client (kubectl)

10-
KUBE_VERSION=$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)
echo $KUBE_VERSION
v1.16.3

11-
curl -Lo kubectl https://storage.googleapis.com/kubernetes-release/release/${KUBE_VERSION}/bin/linux/amd64/kubectl
ls -l
-rw-r--r-- 1 root root 46681472 Nov 16 05:58 kubectl

12- chmod +x kubectl
ls -l
-rwxr-xr-x 1 root root 46681472 Nov 16 05:58 kubectl

13- mv -f kubectl /usr/local/bin/

Kubernetes Installation Method

- **Miniqube Installation :**

Step 2: Install Kubernetes Client (kubectl)

14- kubectl version

Client Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.3", GitCommit:"b3cbbae08ec52a7fc73d334838e18d17e8512749", GitTreeState:"clean", BuildDate:"2019-11-13T11:23:11Z", GoVersion:"go1.12.12", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?

Kubernetes Installation Method

- **Miniqube Installation :**

Step 3: Install Minikube (latest version or fixed version)

15- curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

16- ls -l
-rw-r--r-- 1 root root 48571328 Nov 16 06:06 minikube-linux-amd64

17- install minikube-linux-amd64 /usr/local/bin/minikube

18- minikube version
minikube version: v1.5.2
commit: 792dbf92a1de583fcee76f8791cff12e0c9440ad-dirty

Kubernetes Installation Method

- **Miniqube Installation :**

Step 4: Start Minikube

19- systemctl stop firewalld
20- iptables -L -n

Note:
21- minikube start --vm-driver=none
* minikube v1.5.2 on Centos 7.6.1810
* Requested cpu count 1 is less than the minimum allowed of 2

- **Note : cpu minimum >= 2 + VT-x Enable**

Kubernetes Installation Method

- **Miniqube Installation :**

Step 4: Start Minikube

22- minikube start --vm-driver=none

```
* minikube v1.5.2 on Centos 7.6.1810
* Running on localhost (CPUs=2, Memory=3771MB, Disk=17394MB) ...
* OS release is CentOS Linux 7 (Core)
! VM may be unable to resolve external DNS records
* Preparing Kubernetes v1.16.2 on Docker '18.06.1-ce' ...
* Downloading kubeadm v1.16.2
* Downloading kubelet v1.16.2
* Pulling images ...
* Launching Kubernetes ...
* Configuring local host environment ...
! The 'none' driver provides limited isolation and may reduce system security and reliability.
! For more information, see:
  - https://minikube.sigs.k8s.io/docs/reference/drivers/none/
! kubectl and minikube configuration will be stored in /root
! To use kubectl or minikube commands as your own user, you may need to relocate them. For example, to overwrite your own settings, run:
  - sudo mv /root/.kube /root/.minikube $HOME
  - sudo chown -R $USER $HOME/.kube $HOME/.minikube
* This can also be done automatically by setting the env var CHANGE_MINIKUBE_NONE_USER=true
* Waiting for: apiserver
* Done! kubectl is now configured to use "minikube"
! /usr/local/bin/kubectl is version 1.12.2, and is incompatible with Kubernetes 1.16.2. You will need to update /usr/local/bin/kubectl or use 'minikube kubectl' to connect with this cluster
```

Kubernetes Installation Method

- **Miniqube Installation :**

Step 4: Start Minikube

23- kubectl version

Client Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.3", GitCommit:"b3cbbae08ec52a7fc73d334838e18d17e8512749", GitTreeState:"clean", BuildDate:"2019-11-13T11:23:11Z", GoVersion:"go1.12.12", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.2", GitCommit:"c97fe5036ef3df2967d086711e6c0c405941e14b", GitTreeState:"clean", BuildDate:"2019-10-15T19:09:08Z", GoVersion:"go1.12.10", Compiler:"gc", Platform:"linux/amd64"}

24- kubectl cluster-info

Kubernetes master is running at https://192.168.198.129:8443
KubeDNS is running at https://192.168.198.129:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

TEST :

25- kubectl get node

NAME	STATUS	ROLES	AGE	VERSION
minikube	Ready	master	67s	v1.16.2

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 1: Base Installation on All Nodes

1- 3 Nodes Info :

1- Centos 7.6
2- CPU 2 Core
3- RAM 2 GB
4- Network With VPN and Change MAC Address.

2- Server IP and Hostname Info :

Node1 192.168.198.179
Node2 192.168.198.177
Node3 192.168.198.178

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 1: Base Installation on All Nodes

3- Set Hostname

```
hostnamectl set-hostname node1
hostnamectl
bash

hostnamectl set-hostname node2
hostnamectl
bash

hostnamectl set-hostname node3
hostnamectl
bash
```

4- Disable Selinux

```
sestatus
```

5- systemctl disable firewalld && systemctl stop firewalld

```
iptables -L -n
```

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 1: Base Installation on All Nodes

6- vim /etc/hosts :

192.168.198.179 node1
192.168.198.177 node2
192.168.198.178 node3

7- ssh keygen Generate and Copy

ssh-keygen -t rsa
ssh-copy-id node1
ssh-copy-id node2
ssh-copy-id node3

8- yum clean all ; yum update –y

9- reboot

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 1: Base Installation on All Nodes

10- vi /etc/modules-load.d/bridge-nf.conf

br_netfilter

11- echo "net.bridge.bridge-nf-call-iptables = 1" >> /etc/sysctl.conf

12- reboot

13- For check :

lsmod | grep br_netfilter

```
br_netfilter          22256  0
bridge               151336  1 br_netfilter
```

sysctl -a | grep "net.bridge.bridge-nf-call-iptables"

```
net.bridge.bridge-nf-call-iptables = 1
```

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 1: Base Installation on All Nodes

14- swapoff -a

vim /etc/fstab Remove Swap Mount

#/dev/mapper/centos-swap swap swap defaults 0 0

15- yum install -y yum-utils device-mapper-persistent-data lvm2

Checking Disk with Ftype=1 :

xfs_info / | grep ftype

naming =version 2 bsize=4096 ascii-ci=0 ftype=1

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 2: Install Docker and Kubernetes on All Nodes

16- curl ipinfo.io

17- yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

18- vi /etc/yum.repos.d/kubernetes.repo

```
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
        https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
```

19- yum clean all
20- yum list

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 2: Install Docker and Kubernetes on All Nodes

21- yum install -y docker-ce
systemctl start docker

22- cat > /etc/docker/daemon.json <<EOF

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ]
}
```

EOF

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 2: Install Docker and Kubernetes on All Nodes

23- systemctl enable docker
systemctl restart docker

24- yum install -y kubelet kubeadm kubectl

25- systemctl enable kubelet
Note : (kubelet will go up and down)

26- reboot all nodes

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 3: Configuration Kubernetes On Master Node

27- sysctl -a | grep ip_for

net.ipv4.ip_forward = 1

28- kubeadm init --pod-network-cidr=10.245.0.0/16 --apiserver-advertise-address=192.168.198.179

WAIT FOR DOWNLOAD IMAGES and Then

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
k8s.gcr.io/kube-proxy	v1.15.0	d235b23c3570	3 weeks ago	82.4MB
k8s.gcr.io/kube-apiserver	v1.15.0	201c7a840312	3 weeks ago	207MB
k8s.gcr.io/kube-controller-manager	v1.15.0	8328bb49b652	3 weeks ago	159MB
k8s.gcr.io/kube-scheduler	v1.15.0	2d3813851e87	3 weeks ago	81.1MB
k8s.gcr.io/coredns	1.3.1	eb516548c180	6 months ago	40.3MB
k8s.gcr.io/etcd	3.3.10	2c4adeb21b4f	7 months ago	258MB
k8s.gcr.io/pause	3.1	da86e6ba6ca1	19 months ago	742kB

GET TOKEN :

**kubeadm join 192.168.198.179:6443 --token 1kk99v.xd59s661o2ancje7 **

--discovery-token-ca-cert-hash sha256:f9c76b7a612a69e7ded8676b9be71740469052924e4373c14e2a266f2e55b477

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 3: Configuration Kubernetes On Master Node

29- RUN for execute kubectl :

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

30- kubectl get node

```
NAME    STATUS     ROLES    AGE     VERSION
node1   NotReady   master   6h29m   v1.15.0
```

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 4: Configuration Kubernetes Network On Master Node

31-
curl https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml -o /opt/kube-flannel.yml
curl https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml -o /opt/kube-flannel-rbac.yml

32- vi kube-flannel.yml // and in "Network" field change "10.244.0.0/16" to "10.245.0.0/16".

Network "10.245.0.0/16"

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 4: Configuration Kubernetes Network On Master Node

33- cd /opt

kubectl apply -f kube-flannel-rbac.yml

OUTPUT :

```
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
```

34- kubectl apply -f kube-flannel.yml

OUTPUT :

```
podsecuritypolicy.extensions/psp.flannel.unprivileged created
clusterrole.rbac.authorization.k8s.io/flannel configured
clusterrolebinding.rbac.authorization.k8s.io/flannel unchanged
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.extensions/kube-flannel-ds-amd64 created
daemonset.extensions/kube-flannel-ds-arm64 created
daemonset.extensions/kube-flannel-ds-arm created
daemonset.extensions/kube-flannel-ds-ppc64le created
daemonset.extensions/kube-flannel-ds-s390x created
```

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 5: Join Nodes , Run on Worker Nodes

35- ON Node2 , Node3

kubeadm join 192.168.198.179:6443 --token 1kk99v.xd59s661o2ancje7 --discovery-token-ca-cert-hash sha256:f9c76b7a612a69e7ded8676b9be71740469052924e4373c14e2a266f2e55b477

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 6: Checking Join Nodes and RUN on Master Node

36- Note : WAIT Nodes NotReady To Ready

watch -d -n 1 "kubectl get node"

```
NAME    STATUS   ROLES    AGE     VERSION
node1   Ready    master   113m    v1.15.0
node2   Ready    <none>   4m10s   v1.15.0
node3   Ready    <none>   3m7s    v1.15.0
```

OR

kubectl get node -o wide

```
NAME    STATUS  ROLES    AGE   VERSION  INTERNAL-IP     EXTERNAL-IP  OS-IMAGE            KERNEL-VERSION
CONTAINER-RUNTIME
node1   Ready   master   132m  v1.15.0  192.168.198.179  <none>      CentOS Linux 7 (Core)  3.10.0-957.21.3.el7.x86_64  docker://18.9.7
node2   Ready   <none>   23m   v1.15.0  192.168.198.177  <none>      CentOS Linux 7 (Core)  3.10.0-957.21.3.el7.x86_64  docker://18.9.7
node3   Ready   <none>   22m   v1.15.0  192.168.198.178  <none>      CentOS Linux 7 (Core)  3.10.0-957.21.3.el7.x86_64  docker://18.9.7
```

Kubernetes Installation Method

- **Kubeadm Installation For CKA EXAM :**

Step 7: Final Install and Check it Run on Worker Nodes

37- ip a

Network For and IP Address For Docker :

```
docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:49:99:6e:cd brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

Network and IP Address For Kubernetes

```
flannel.1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UNKNOWN group
default
    link/ether 9a:ba:f1:5f:0f:11 brd ff:ff:ff:ff:ff:ff
    inet 10.245.1.0/32 scope global flannel.1
       valid_lft forever preferred_lft forever
    inet6 fe80::98ba:f1ff:fe5f:f11/64 scope link
       valid_lft forever preferred_lft forever
```

Kubernetes Installation Method

- **Kubespray Installation For Opertaion Environment**

kubectl Commands

kubectl Structure :

kubectl [command] [TYPE] [NAME] [flags]

command: For Example :

- ✓ **create**
- ✓ **get**
- ✓ **describe**
- ✓ **delete**
- ✓ **run**

NAME:

- ✓ **pod1**
- ✓ **example-pod1 example-pod2**
- ✓ **-f ./pod.yaml , -f file1**

flags:

- ✓ **-s or –server**
- ✓ **-o …**

TYPE : For Example :

- ✓ **nodes**
- ✓ **pods**
- ✓ **services**
- ✓ **deployments**
- ✓ **persistentvolumes**
- ✓ **persistentvolumeclaims**
- ✓ **apiservices**
- ✓ **replicasets**
- ✓ **statefulsets**

kubectl Commands

KubeCtl Command Category Type

1-GETTING STARTED

- create
- get
- run
- expose
- delete

2-APP MANAGEMENT

- apply
- autoscale
- label
- replace
- scale
- set

3-WORKING WITH APPS

- attach
- cp
- describe
- exec
- logs
- port-forward

4-CLUSTER MANAGEMENT

- cluster-info

5-KUBECTL SETTINGS AND USAGE

- config
- version

kubectl Commands

Auto COMPILATION Command

- ✓ yum install bash-completion
- ✓ source <(kubectl completion bash)
- ✓ echo "source <(kubectl completion bash)" >> ~/.bashrc
- ✓ bash
- ✓ echo "source <(kubectl completion bash)" >> ~/.bashrc
- ✓ source .bashrc
- ✓ For Test With Tab :
  kubectl get deployments

kubectl Commands

Lab1 : kubectl version

Client Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.2", GitCommit:"c97fe5036ef3df2967d086711e6c0c405941e14b", GitTreeState:"clean", BuildDate:"2019-10-15T19:18:23Z", GoVersion:"go1.12.10", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.2", GitCommit:"c97fe5036ef3df2967d086711e6c0c405941e14b", GitTreeState:"clean", BuildDate:"2019-10-15T19:09:08Z", GoVersion:"go1.12.10", Compiler:"gc", Platform:"linux/amd64"}

Lab2 : kubectl cluster-info

Kubernetes master is running at https://192.168.198.169:6443
KubeDNS is running at https://192.168.198.169:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

kubectl Commands

Lab3 : kubectl config view

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://192.168.198.169:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

kubectl Commands

Lab4 : kubectl get all

NAME	READY	STATUS	RESTARTS	AGE
pod/my-pod	0/2	ErrImagePull	0	5d8h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	12d
service/sr-nginx	NodePort	10.100.152.236	<none>	8080:32222/TCP	11d

Lab5 : kubectl get node

NAME	STATUS	ROLES	AGE	VERSION
node1	Ready	master	9h	v1.16.2
node2	Ready	<none>	8h	v1.16.2
node3	Ready	<none>	9h	v1.16.2

kubectl Commands

Lab6 : kubectl get all -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
pod/my-pod	0/2	ImagePullBackOff	0	5d8h	10.245.2.107	node2	<none>	<none>

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	12d	<none>
service/sr-nginx	NodePort	10.100.152.236	<none>	8080:32222/TCP	11d	app=nginx

Lab7 : kubectl get node -o wide

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
node1	Ready	master	9h	v1.16.2	192.168.198.169	<none>	CentOS Linux 7 (Core)	3.10.0-1062.4.1.el7.x86_64	docker://19.3.4
node2	Ready	<none>	8h	v1.16.2	192.168.198.190	<none>	CentOS Linux 7 (Core)	3.10.0-1062.4.1.el7.x86_64	docker://19.3.4
node3	Ready	<none>	9h	v1.16.2	192.168.198.191	<none>	CentOS Linux 7 (Core)	3.10.0-1062.4.1.el7.x86_64	docker://19.3.4

kubectl Commands

Lab8 : RUN ONLY Pod

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80
pod/nginx created

- ✓ Pod Name : ngnix
- ✓ Image Name : nginx
- ✓ Port Number : 80
- ✓ **Download Image and Run Pod on one Random Worker Node And Start It.**

kubectl get all

NAME	READY	STATUS	RESTARTS	AGE
pod/nginx	1/1	Running	0	5m44s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	58m

kubectl Commands

Lab9 : Checking Pod Running :

ON Master :

kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
nginx	1/1	Running	0	8m51s

kubectl get pod -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
nginx	1/1	Running	0	28m	**10.245.2.2**	node2	<none>	<none>

kubectl Commands

Lab9 : Checking Pod Running :

ON Worker :

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	540a289bab6c	4 days ago	126MB

docker ps | grep nginx

```
9c449feebaa9   nginx                  "nginx -g 'daemon of…"   35 minutes ago   Up 11 seconds
k8s_nginx_nginx_default_03db764d-7ff7-41af-8f6d-f4b3ab8ee183_2
cfe0b71e6eed   k8s.gcr.io/pause:3.1   "/pause"                 35 minutes ago   Up 35 minutes
k8s_POD_nginx_default_03db764d-7ff7-41af-8f6d-f4b3ab8ee183_10
```

Result : From Inside Kubernetes Node2 , Node3 :

curl http://10.245.2.2

kubectl Commands

Lab10 : Number of RESTARTS Pod :

ON Worker :

docker ps | grep nginx

9c449feebaa9 nginx "nginx -g 'daemon of…" 35 minutes ago Up 11 seconds k8s_nginx_nginx_default_03db764d-7ff7-41af-8f6d-f4b3ab8ee183_2

cfe0b71e6eed k8s.gcr.io/pause:3.1 "/pause" 35 minutes ago Up 35 minutes k8s_**POD_nginx**_default_03db764d-7ff7-41af-8f6d-f4b3ab8ee183_10

[root@node2 ~]# docker restart cfe0b71e6eed => pod ID

ON Master :

kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
nginx	1/1	Running	3	42h

RESTARTS +1

kubectl Commands

Lab11 : Delete Pod :

kubectl delete pod POD_NAME

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80

pod/nginx created

kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
nginx	0/1	ContainerCreating	0	9s

kubectl delete pod nginx

pod "nginx" deleted

kubectl Commands

Lab12 : RUN Deployment :

kubectl run nginx --image=nginx --port 80

deployment.apps/nginx created

- ✓ Deployment Name : ngnix
- ✓ Image Name : nginx
- ✓ Port Number : 80
- ✓ **Download Image and Run Deployment with one Pod and Container on one Random Worker Node And Start It.**

Note :

kubectl create

kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be Removed in a future version.

Use kubectl run --generator=run-pod/v1 or kubectl create instead.

kubectl Commands

Lab13 : Checking Deployment Running :

ON Master :

Deployment Name

kubectl get deployment

```
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   0/1     1            0           2s
```

kubectl get pod :

```
NAME                     READY   STATUS    RESTARTS   AGE
nginx-5578584966-fgwr9   1/1     Running   0          6s
```

POD_NAME

kubectl Commands

Lab14 : Delete Deployment :

kubectl delete deployment DEPLOYMENT_NAME

kubectl run nginx --image=nginx --port 80

deployment.apps/nginx created

kubectl get deployment

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx	0/1	1	0	2s

kubectl delete deployment nginx

deployment.apps "nginx" deleted

kubectl Commands

Lab15 : RUN Deployment : NEW Version

kubectl create deployment nginx --image nginx

```
deployment.apps/nginx created
```

kubectl get deployment

```
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           107s
```

kubectl get deployment -o wide

```
NAME    READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES   SELECTOR
nginx   1/1     1            1           42m   nginx        nginx    run=nginx
```

kubectl Commands

Lab16: RUN Scale Deployment

kubectl scale deployment --replicas Number_Of_POD DEPLOYMENT_NAME

kubectl scale deployment --replicas 2 nginx

deployment.apps/nginx scaled

kubectl get deployment -o wide

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
nginx	2/2	2	2	6m28s	nginx	nginx	app=nginx

kubectl Commands

Lab16: RUN Scale Deployment Test :

kubectl get pod -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
nginx-86c57db685-pgtwm	1/1	Running	0	4m1s	10.245.1.2	node3	<none>	<none>
nginx-86c57db685-zfcvq	1/1	Running	0	6m19s	10.245.2.3	node2	<none>	<none>

Result : From Inside Kubernetes Node2 , Node3 :

curl http://10.245.1.2
curl http://10.245.2.3

kubectl Commands

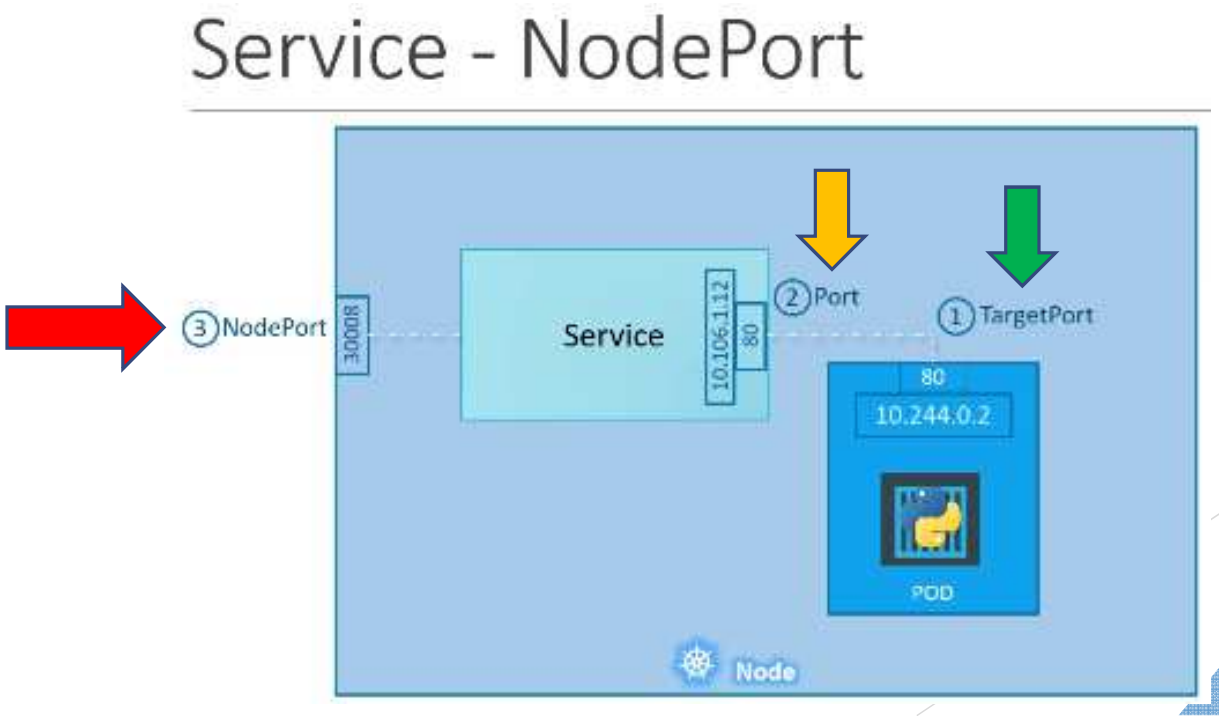
Project :

1- Create httpd pod with YourlastName
2- Create nginx Deployment with :
   - Name : YourFirstName
   - scale 3
   - port 80
3- View all pod and deployment pods with Pod IPs and Test it With Curl and Telnet
4- Delete Pod and Deployment

kubectl Commands

RUN EXPOSE PORT IN Deployment

Kubectl expose deploy DEPLOYMENT_Name --port Service_Port_No --target-port Pod_Port_No --type NodePort



kubectl Commands

RUN EXPOSE PORT IN Deployment

Type of Service :

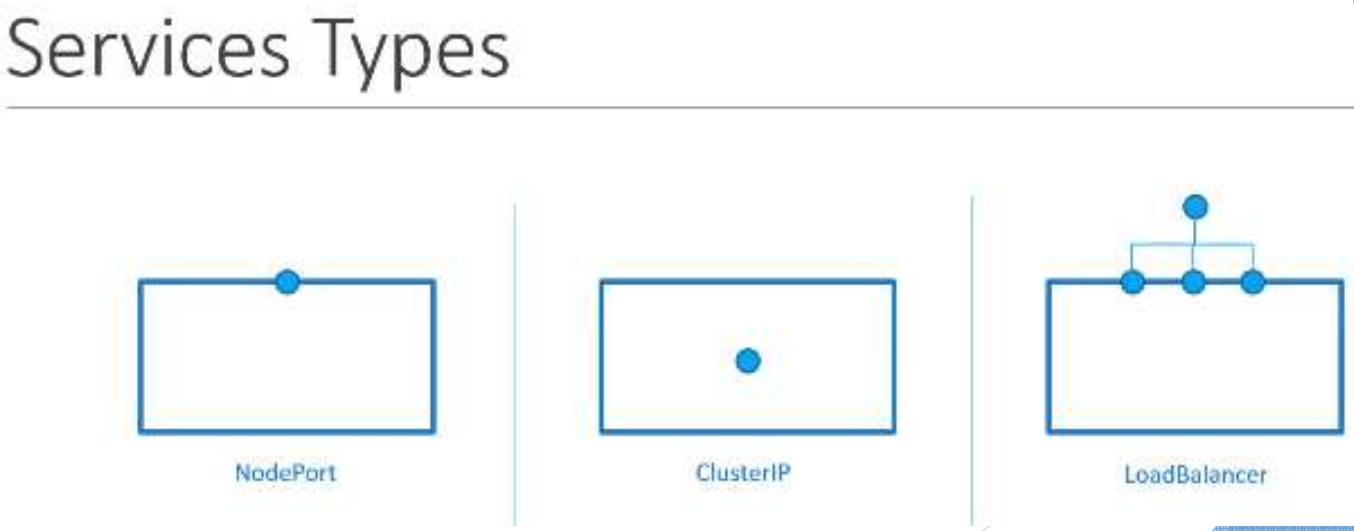
--type :

ClusterIP : Create A virtual IP Between services , Frontend Service To Backend Service

NodePort : All Pod OutPut With physical Port in kubernetes Hosts. (Port Publish)

Default Port Range : 30000-32767

LoadBalancer : Kubernetes Hosts Port Link to External LoadBalancer



kubectl Commands

Lab17- RUN EXPOSE PORT IN Nginx Deployment :

EXPOSE Nginx Port 80 to Untrust Zone with NodePort

kubectl create deployment nginx --image nginx

deployment.apps/nginx created

kubectl scale deployment --replicas 2 nginx

deployment.apps/nginx scaled

kubectl get pod -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
nginx-86c57db685-pgtwm	1/1	Running	0	4m1s	10.245.1.2	node3	<none>	<none>
nginx-86c57db685-zfcvq	1/1	Running	0	6m19s	10.245.2.3	node2	<none>	<none>

kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort

OR

kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort

kubectl Commands

Lab17- RUN EXPOSE PORT IN Nginx Deployment :

EXPOSE Nginx Port 80 to Untrust Zone with NodePort

Test :

kubectl get services : List All Services

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	11h
nginx	**NodePort**	10.96.70.229	<none>	**80:31663/TCP**	54m

kubectl get service -o wide

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	11h	<none>
nginx	NodePort	10.96.70.229	<none>	80:31663/TCP	53m	run=nginx

kubectl get service nginx

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx	NodePort	10.96.70.229	<none>	80:31663/TCP	56m

Result : From Any Host in Web Browser :

http://192.168.198.190:31663
http://192.168.198.191:31663

kubectl Commands

Lab18- Delete Services :

kubectl delete service SERVICE_NAME

kubectl delete service nginx

service "nginx" deleted

kubectl get services

```
NAME        TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes  ClusterIP   10.96.0.1    <none>        443/TCP   3h6m
```

Test :

Result : From Any Host in Web Browser With Failed

http://192.168.198.190:31663

http://192.168.198.191:31663

kubectl Commands

Port-Forward : Forward Traffic From OutSide Kubernetes to Deployment

With Only Master Node From Master Node IP :

kubectl port-forward Pod_Names_From_Deployment KubernetesPort:Pod_Port

kubectl Commands

Lab19- Port-Forward : Forward Traffic From OutSide Kubernetes to Deployment

kubectl create deployment nginx --image nginx

deployment.apps/nginx created

kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
nginx-86c57db685-49v9l	1/1	Running	0	2m

kubectl port-forward nginx-86c57db685-49v9l 8888:80

Forwarding from 127.0.0.1:8888 -> 80

kubectl Commands

Lab19- Port-Forward : Forward Traffic From OutSide Kubernetes to Deployment

For Test Mater Node only From Master Node IP With IP 8888

http://127.0.0.1:8888/

OR

telnet 127.0.0.1 8888

```
Trying 127.0.0.1 ...
Connected to 127.0.0.1
Escape character is '^]'.
```

OR

curl 127.0.0.1:8888

```
Handling connection for 8888
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
```

kubectl Commands

Port-Forward : Forward Traffic From OutSide Kubernetes to Deployment

With Any Source IP/Range From Master Node IP :
--address :

kubectl port-forward --address AnyIP/Range Pod_Name_From_Deployment KubernetesPort:Pod_Port

kubectl Commands

Lab19- Port-Forward : Forward Traffic From OutSide Kubernetes to Deployment

kubectl create deployment nginx --image nginx

deployment.apps/nginx created

kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
nginx-86c57db685-49v9l	1/1	Running	0	2m

kubectl port-forward --address 0.0.0.0 nginx-86c57db685-49v9l 8888:80

Forwarding from 0.0.0.0:8888 -> 80

kubectl Commands

Lab19- Port-Forward : Forward Traffic From OutSide Kubernetes to Deployment

For Test Any web browser only From Master Node IP With IP 8888

http://192.168.198.169:8888/

OR

telnet 192.168.198.169 8888

```
Trying 192.168.198.169...
Connected to 192.168.198.169.
Escape character is '^]'.
```

OR

curl 192.168.198.169:8888

```
Handling connection for 8888
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
```

kubectl Commands

Project :

Port-Forward : Forward Traffic From OutSide Kubernetes to POD ???

kubectl Commands

Port-Forward : Forward Traffic From OutSide Kubernetes to POD

With Only Master Node From Master Node IP :

kubectl port-forward Pod_Names KubernetesPort:Pod_Port

With Any Source IP/Range From Master Node IP :

--address :

kubectl port-forward --address AnyIP/Range Pod_Name KubernetesPort:Pod_Port

kubectl Commands

Lab20- Port-Forward : Forward Traffic From OutSide Kubernetes to POD

kubectl run nginx --image=nginx --port=80 --generator=run-pod/v1

kubectl get pod nginx

```
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          11s
```

kubectl port-forward --address 0.0.0.0 nginx 8888:80

Forwarding from 0.0.0.0:8888 -> 80

kubectl Commands

Lab20- Port-Forward : Forward Traffic From OutSide Kubernetes to POD

For Test Any web browser only From Master Node IP With IP 8888

http://192.168.198.169:8888/

OR

telnet 192.168.198.169 8888

```
Trying 192.168.198.169...
Connected to 192.168.198.169.
Escape character is '^]'.
```

OR

curl 192.168.198.169:8888

```
Handling connection for 8888
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
```

Note :

netstat -nap | grep 8888

tcp 0 0 0.0.0.0:8888 0.0.0.0:* LISTEN 16575/kubectl

kubectl Commands

Lab21- Change Default NodePort Range :

how to change default NodePort Range and test IT :

1- cd /etc/kubernetes/manifests
2- vim kube-apiserver.yaml
Add Line : **- --service-node-port-range=20000-22767**

- --service-cluster-ip-range=10.96.0.0/12
- --service-node-port-range=20000-22767

3- kubectl apply -f kube-apiserver.yaml
4- kubectl expose deploy nginx --port 80 --target-port 80 --type NodePort
5- kubectl get service

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	12h
nginx	NodePort	10.106.205.4	<none>	80:**20388**/TCP	13s

kubectl Commands

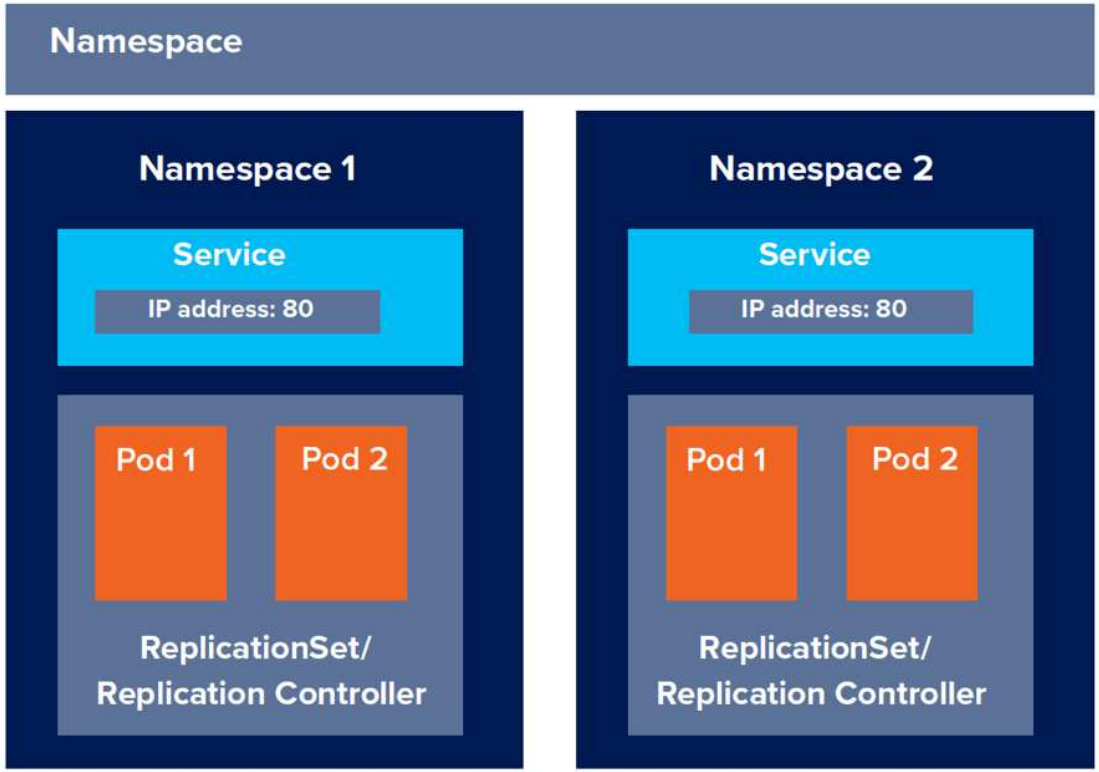
Project :

1- Create httpd Deployment with :
   - Name : YourName
   - scale 2
   - port 80
   - targetport 80
   - nodeport
2- Create Port Forward 9999 to 80
3- View deployment pods with Pod IPs and test it With Curl and Telnet From InSide and OutSide
4- Delete Pod and Deployment

kubectl Commands

Kubernetes NAMESPACE

- ✓ Namespaces provide a scope for names.
- ✓ Names of resources need to be unique within a namespace, but not across namespaces.
- ✓ Namespaces can not be nested inside one another and each Kubernetes resource can only be in one namespace.



kubectl Commands

Lab22- Create NAMESPACE : RUN Deployment with NAMESPACE

kubectl create namespace NAMESPACE_NAME

kubectl create namespace nginx

kubectl get namespace OR kubectl get ns

```
NAME              STATUS   AGE
default           Active   16h
kube-node-lease   Active   16h
kube-public       Active   16h
kube-system       Active   16h
nginx             Active   53s
```

kubectl Commands

Lab23- Create NAMESPACE : RUN Deployment with NAMESPACE

kubectl create deployment nginx --image nginx --namespace NAMESPACE_NAME

kubectl create deployment nginx --image nginx --namespace nginx

deployment.apps/nginx created

kubectl get deployment -A -o wide

NAMESPACE	NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
kube-system	coredns	2/2	2	2	3h23m	coredns	k8s.gcr.io/coredns:1.6.2	k8s-app=kube-dns
nginx	nginx	1/1	1	1	100s	nginx	nginx	app=nginx

OR

kubectl get deployment --namespace nginx -o wide OR --namespace , -n

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
nginx	1/1	1	1	2m32s	nginx	nginx	app=nginx

kubectl Commands

Lab23- Create NAMESPACE : RUN Deployment with NAMESPACE

kubectl get pod -A -o wide

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE    IP                NODE    NOMINATED NODE   READINESS GATES
kube-system   coredns-5644d7b6d9-9fsr5   1/1     Running   0          3h23m  10.245.0.2        node1   <none>           <none>
kube-system   coredns-5644d7b6d9-ktngh   1/1     Running   0          3h23m  10.245.0.3        node1   <none>           <none>
kube-system   kube-proxy-gg594           1/1     Running   0          166m   192.168.198.190   node2   <none>           <none>
kube-system   kube-proxy-gmtmt           1/1     Running   0          168m   192.168.198.191   node3   <none>           <none>
kube-system   kube-proxy-j7s6x           1/1     Running   0          3h23m  192.168.198.169   node1   <none>           <none>
kube-system   kube-scheduler-node1       1/1     Running   11         3h22m  192.168.198.169   node1   <none>           <none>
nginx         nginx-86c57db685-vclv8     1/1     Running   0          101s   10.245.1.3        node3   <none>           <none>
```

kubectl get pod --namespace nginx -o wide OR --namespace , -n

```
NAME                       READY   STATUS    RESTARTS   AGE    IP          NODE    NOMINATED NODE   READINESS GATES
nginx-86c57db685-vclv8     1/1     Running   0          2m48s  10.245.1.3  node3   <none>           <none>
```

TEST :

Result : From Inside Kubernetes Node2 , Node3 and Master Node :

curl http://10.245.1.3

kubectl Commands

Lab24- RUN Deployment with NAMESPACE with Scale

kubectl scale deployment --replicas Number_POD DEPLOYMENT_NAME --namespace NAMESPACE_NAME

kubectl scale deployment --replicas 2 nginx --namespace nginx

deployment.apps/nginx scaled

kubectl get deployment --namespace nginx -o wide

NAME	READY	UP-TO-DATE	AVAILABLE	AGE	CONTAINERS	IMAGES	SELECTOR
nginx	2/2	2	2	15m	nginx	nginx	app=nginx

kubectl get pod --namespace nginx -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
nginx-86c57db685-k48qp	1/1	Running	0	37s	10.245.2.4	node2	<none>	<none>
nginx-86c57db685-vclv8	1/1	Running	0	15m	10.245.1.3	node3	<none>	<none>

kubectl Commands

Lab24- RUN Deployment with NAMESPACE with Scale

TEST :

Result : From Inside Kubernetes Node2 , Node3 :

curl http://10.245.1.3
curl http://10.245.2.4

[root@node3 ~]# docker ps | grep nginx

d3b41977fc3e nginx "nginx -g 'daemon of…" 17 minutes ago Up 17 minutes k8s_nginx_nginx-86c57db685-vclv8_nginx_e73f17f5-2290-43b5-8570-b9a2aa149ba3_0
df3ff0a49f5a k8s.gcr.io/pause:3.1 "/pause" 17 minutes ago Up 17 minutes k8s_POD_nginx-86c57db685-vclv8 nginx_e73f17f5-2290-43b5-8570-b9a2aa149ba3_0

[root@node2 ~]# docker ps | grep nginx

10618cad7065 nginx "nginx -g 'daemon of…" 2 minutes ago Up 2 minutes k8s_nginx_nginx-86c57db685-k48qp_nginx_b3139e56-791e-4c91-9068-8eb7af1b3eef_0
5cdf0fde7233 k8s.gcr.io/pause:3.1 "/pause" 3 minutes ago Up 2 minutes k8s_POD_nginx-86c57db685-k48qp nginx_b3139e56-791e-4c91-9068-8eb7af1b3eef_0

kubectl Commands

KubeCtl Get all

View ALL POD ON ALL NameSpaces :

kubectl get all --all-namespaces

OR

kubectl get all -A

```
NAMESPACE     NAME                                    READY   STATUS          RESTARTS   AGE
default       pod/nginx                               1/1     Running         0          5h51m
default       pod/nginx-5578584966-zbjgq              1/1     Running         0          5h51m
kube-system   pod/coredns-5644d7b6d9-9fsr5            1/1     Running         1          15h
kube-system   pod/coredns-5644d7b6d9-ktngh            1/1     Running         1          15h
kube-system   pod/etcd-node1                          1/1     Running         1          15h
kube-system   pod/kube-apiserver                      0/1     CrashLoopBackOff   46      3h37m
kube-system   pod/kube-apiserver-node1                1/1     Running         0          3h37m
kube-system   pod/kube-proxy-gmtmt                    1/1     Running         1          15h
kube-system   pod/kube-proxy-j7s6x                    1/1     Running         1          15h
kube-system   pod/kube-scheduler-node1                1/1     Running         3          15h

NAMESPACE     NAME              TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)              AGE
default       service/kubernetes   ClusterIP   10.96.0.1      <none>       443/TCP              15h
default       service/nginx       NodePort    10.106.205.4   <none>       80:20388/TCP         3h36m
kube-system   service/kube-dns     ClusterIP   10.96.0.10     <none>       53/UDP,53/TCP,9153/TCP   15h

NAMESPACE     NAME                                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   daemonset.apps/kube-flannel-ds-amd64    3         3         3       3            3           <none>                   15h
kube-system   daemonset.apps/kube-flannel-ds-arm      0         0         0       0            0           <none>                   15h
kube-system   daemonset.apps/kube-flannel-ds-s390x    0         0         0       0            0           <none>                   15h
kube-system   daemonset.apps/kube-proxy               3         3         3       3            3           beta.kubernetes.io/os=linux   15h

NAMESPACE     NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
default       deployment.apps/nginx     1/1     1            1           5h51m
kube-system   deployment.apps/coredns   2/2     2            2           15h

NAMESPACE     NAME                                    DESIRED   CURRENT   READY   AGE
default       replicaset.apps/nginx-5578584966        1         1         1       5h51m
kube-system   replicaset.apps/coredns-5644d7b6d9      2         2         2       15h
```

kubectl Commands

Lab25- Delete Deployment , Services , NAMESPACE

- **Delete Deployment :**

kubectl delete deployment nginx -n nginx

```
deployment.apps "nginx" deleted
```

kubectl get all -n nginx

```
NAME                          READY   STATUS        RESTARTS   AGE
pod/nginx-86c57db685-k48qp    1/1     Terminating   0          8m31s
pod/nginx-86c57db685-vclv8    1/1     Terminating   0          23m

NAME            TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)       AGE
service/nginx   NodePort   10.103.238.205  <none>        80:31094/TCP  15m
```

kubectl Commands

Lab25- Delete Deployment , Services , NAMESPACE

- **Delete Services:**

kubectl delete service nginx -n nginx

service "nginx" deleted

kubectl get service -n nginx

No resources found in nginx namespace

kubectl Commands

Lab25- Delete Deployment , Services , NAMESPACE

- **Delete NameSpace :**

kubectl delete namespace nginx

namespace "nginx" deleted

kubectl get namespace

```
NAME              STATUS   AGE
default           Active   16h
kube-node-lease   Active   16h
kube-public       Active   16h
kube-system       Active   16h
```

kubectl Commands

Project :

RUN Deployment With :

- **Image : httpd**
- **Name : myweb**
- **NAMESPACE : web**
- **EXPOSE PORT Type : NodePort**
- **Sclae up to 2 POD???**

Delete All :

- **Deployment**
- **Service**
- **NameSpace**

kubectl Commands

Project :

kubectl create namespace web

namespace/web created

kubectl create deployment myweb --image httpd --namespace web

deployment.apps/myweb created

kubectl get -n web deployments.apps myweb

```
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myweb   1/1     1            1           104s
```

kubectl get pod -n web

```
NAME                      READY   STATUS    RESTARTS   AGE
myweb-7b9b8d697c-m5sfq    1/1     Running   0          2m27s
```

kubectl Commands

Project :

kubectl expose deploy myweb --port 80 --target-port 80 --type NodePort --namespace web

service/myweb exposed

kubectl get service -n web

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
myweb	NodePort	10.103.238.205	<none>	80:31094/TCP	25s

kubectl get services -A OR --all-namespaces , -A

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
default	kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3h30m
kube-system	kube-dns	ClusterIP	10.96.0.10	<none>	53/UDP,53/TCP,9153/TCP	3h30m
web	myweb	NodePort	10.103.238.205	<none>	80:31094/TCP	15s

kubectl get services --all-namespaces -o wide

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
default	kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	3h31m	<none>
kube-system	kube-dns	ClusterIP	10.96.0.10	<none>	53/UDP,53/TCP,9153/TCP	3h31m	k8s-app=kube-dns
web	myweb	NodePort	10.103.238.205	<none>	80:31094/TCP	116s	app=nginx

Result : From Any Host in Web Browser

http://192.168.198.190:31094

http://192.168.198.191:31094

kubectl Commands

Project :

kubectl scale deployment --replicas 2 myweb --namespace web

```
deployment.apps/myweb scaled
```

kubectl get pod -n web

```
NAME                      READY   STATUS    RESTARTS   AGE
myweb-86c57db685-l6h4c    1/1     Running   0          19s
myweb-86c57db685-q4nqz    1/1     Running   0          12m
```

kubectl get deployments.apps -n web

```
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
myweb   2/2     2            2           12m
```

kubectl get service -n web

```
NAME    TYPE       CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
myweb   NodePort   10.103.238.205  <none>         80:31094/TCP   25s
```

kubectl get services -A OR --all-namespaces , -A

```
NAMESPACE     NAME        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                AGE
default       kubernetes  ClusterIP   10.96.0.1       <none>        443/TCP                3h30m
kube-system   kube-dns    ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP,9153/TCP 3h30m
web           myweb       NodePort    10.103.238.205  <none>        80:31094/TCP           15s
```

kubectl Commands

Project :

kubectl delete deployments.apps httpd –n web

namespace/web deleted

kubectl delete service myweb -n web

service "myweb" deleted

kubectl delete namespaces web

namespace "web" deleted

kubectl Commands

Lab26- View All events on All NameSpace

kubectl create deploy nginx --image=nginx

deployment.apps/nginx created

kubectl get events --all-namespaces

```
NAMESPACE   LAST SEEN   TYPE     REASON            OBJECT                      MESSAGE
default     <unknown>   Normal   Scheduled         pod/nginx-86c57db685-496xp  Successfully assigned default/nginx-86c57db685-496xp to node2
default     11s         Normal   Pulling           pod/nginx-86c57db685-496xp  Pulling image "nginx"
default     3s          Normal   Pulled            pod/nginx-86c57db685-496xp  Successfully pulled image "nginx"
default     3s          Normal   Created           pod/nginx-86c57db685-496xp  Created container nginx
default     2s          Normal   Started           pod/nginx-86c57db685-496xp  Started container nginx
default     12s         Normal   SuccessfulCreate  replicaset/nginx-86c57db685 Created pod: nginx-86c57db685-496xp
default     12s         Normal   ScalingReplicaSet deployment/nginx            Scaled up replica set nginx-86c57db685 to 1
```

kubectl scale deployment --replicas 2 nginx

deployment.apps/nginx scaled

kubectl get events --all-namespaces | grep nginx | grep repli

```
default   3m6s   Normal   ScalingReplicaSet   deployment/nginx   Scaled up replica set nginx-86c57db685 to 1
default   95s    Normal   ScalingReplicaSet   deployment/nginx   Scaled up replica set nginx-86c57db685 to 2
```

kubectl Commands

Lab27- Kubectl Describe

kubectl describe nodes

```
Name:               node1,node2,node3
Roles:              master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node1
                    kubernetes.io/os=linux
Hostname:    node1
Capacity:
 cpu:                2
 ephemeral-storage:  17394Mi
 hugepages-1Gi:      0
 hugepages-2Mi:      0
 memory:             1863104Ki
```

kubectl describe nodes | grep IP

```
InternalIP:  192.168.198.169
InternalIP:  192.168.198.190
InternalIP:  192.168.198.191
```

kubectl describe nodes node1

kubectl Commands

Project :

Run nginx Deployment with yourname

View Describe

kubectl Commands

Project :

Run nginx Deployment with yourname

View Describe

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80

pod/nginx created

kubectl describe pod nginx

```
Name:         nginx
Namespace:    default
Status:       Running
IP:           10.245.2.116
IPs:
  IP:  10.245.2.116
Containers:
  nginx:
    Container ID:   docker://bb8dbe5249a4ecb76e7cad76aa5b017e0ae72e506518c0619de31b357cfbec1e
    Image:          nginx
Host Port:      0/TCP
    State:          Running
      Started:      Sun, 24 Nov 2019 08:02:07 +0330
Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-wkmhc (ro)
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  <unknown>          default-scheduler  Successfully assigned default/nginx to node2
  Warning  Failed     96s (x2 over 110s)  kubelet, node2     Failed to pull image "nginx": rpc error: code = Unknown desc = Error response from daemon: error parsing HTTP 403 response body: invalid character '<' looking for beginning of value: "<html><body><h1>403 Forbidden</h1>\nSince Docker is a US company, we must comply with US export control regulations. In an effort to comply with these, we now block all IP addresses that are located in Cuba, Iran, North Korea, Republic of Crimea, Sudan, and Syria. If you are not in one of these cities, countries, or regions and are blocked, please reach out to https://support.docker.com\n</body></html>\n\n"
  Warning  Failed     96s (x2 over 110s)  kubelet, node2     Error: ErrImagePull
  Normal   BackOff    84s (x2 over 110s)  kubelet, node2     Back-off pulling image "nginx"
  Warning  Failed     84s (x2 over 110s)  kubelet, node2     Error: ImagePullBackOff
  Normal   Pulling    70s (x3 over 112s)  kubelet, node2     Pulling image "nginx"
  Normal   Pulled     58s                kubelet, node2     Successfully pulled image "nginx"
  Normal   Created    58s                kubelet, node2     Created container nginx
  Normal   Started    58s                kubelet, node2     Started container nginx
```

kubectl Commands

dry-run : ONLY Test and Check For Run Pod But WithOut RUN Pod

kubectl run --generator=run-pod/v1 anisa --image=nginx --port 80 --dry-run

pod/nginx created (dry run)

kubectl get pod

NAME READY STATUS RESTARTS AGE

kubectl Commands

kubectl exec Connect to Container in Pod

- ✓ **With Master Node :**

  kubectl exec -it PODNAME -c ContainerName bash

- ✓ **With Worker Nodes :**

  docker exec –it ContainerName bash
  OR
  docker attach ContainerName

kubectl Commands

Lab28- kubectl exec Connect to Container in Pod

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80
kubectl describe pod nginx

```
Name:         nginx
Namespace:    default
Priority:     0
Node:         node3/192.168.198.191
Start Time:   Sun, 27 Oct 2019 14:09:36 +0330
Labels:       run=nginx
Annotations:  <none>
Status:       Running
IP:           10.245.1.9
IPs:
  IP:  10.245.1.9
Containers:
```

nginx: ===> **container NAME**

kubectl exec -it nginx -c nginx bash
root@nginx:/#

kubectl Commands

Lab29- Container Restart Mode , Always

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80 --restart='Always'

pod/nginx created

ON Worker Node :

IF Container kill , Restart Again POD ID ===> note : Default Always

docker kill c16d542770f1 ; watch -d -n 1 "docker ps | grep nginx"

69e95438c44c k8s.gcr.io/pause:3.1 "/pause" 3 seconds ago Up 2 seconds k8s_POD_nginx_default_96ea335d-04aa-4400-8df2-1dbb91fadda7_0

kubectl Commands

Lab29- Container Restart Mode , Never

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80 --restart='Never'

pod/nginx created

ON Worker Node :

IF Container kill , NOT Restart Again

docker kill c16d542770f1 ; watch -d -n 1 "docker ps | grep nginx"

kubectl Commands

Lab30- Create ENV For Container In Pod

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80 --env="VAR1=10"

pod/nginx created

kubectl get pod

```
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          9s
```

ON Master Node :

kubectl exec -it nginx -c nginx bash

root@nginx:/# echo $VAR1

10

ON Worker Node :

docker ps | grep ngin

```
5f45a62b7944        nginx    "nginx -g 'daemon of…"   50 seconds ago      Up 49 seconds
k8s_nginx_nginx_default_03db764d-7ff7-41af-8f6d-f4b3ab8ee183_0
```

docker exec -it 5f bash

root@nginx:/# **echo $VAR1**

10

kubectl Commands

Project :

- ✓ **Run Kubernetes Mairadb Pod With :**
  Port and TargetPort 3306
  Restart Never

- ✓ **Connect To Pod With Kubernetes and Create DataBases Anisa**
- ✓ **Connect To Container With Docker and Check DB Anisa**
- ✓ **Create Service and Check From OutSide**
- ✓ **Check It…**

kubectl Commands

Project :

- ✓ **Run Kubernetes Mairadb Pod With :**
  Port and TargetPort 3306
  Restart Never
- ✓ **Connect To Pod With Kubernetes and Create DataBases Anisa**
- ✓ **Connectt To Container With Docker and Check DB Anisa**
- ✓ **Create Service and Check From OutSide**

```
kubectl run  --generator=run-pod/v1  mariadb --image=mariadb --port 3306  --restart='Never' --env="MYSQL_ROOT_PASSWORD=123“

kubectl exec -it mariadb -c mariadb bash
root@mariadb:/# mysql -u root -p
        MariaDB [(none)]> create database anisa;
        MariaDB [(none)]> show databases;
[root@node2 ~]# docker exec -it 90  bash
root@mariadb:/# mysql -u root -p
        MariaDB [(none)]> show databases;
kubectl  expose pod  mariadb --port 3306 --target-port 3306 --type NodePort
        service/mariadb exposed
 kubectl  get service mariadb
        NAME      TYPE       CLUSTER-IP   EXTERNAL-IP   PORT(S)         AGE
        mariadb   NodePort   10.98.15.3   <none>        3306:32188/TCP   2m17s
Test From Any Host :
telnet 192.168.198.169 32188
mysql -u root -h 192.168.198.169 -P 32188 -p
```

kubectl Commands

Lab31- kubectl run from centos pod and container with Login Bash

kubectl run -it centos --image=centos -- /bin/bash

[root@centos-6dbdb4689c-9wrcm /]#

Worker Node :

docker ps | grep centos

```
d0e874a78177   centos   "/bin/bash"   2 minutes ago   Up 2 minutes
k8s_centos_centos-6dbdb4689c-9wrcm_default_cd04d00f-77a2-405d-8269-b65386832616_0
3ccd8e2ac8d9   k8s.gcr.io/pause:3.1   "/pause"   4 minutes ago   Up 4 minutes
k8s_POD_centos-6dbdb4689c-9wrcm_default_cd04d00f-77a2-405d-8269-b65386832616_0
```

[root@node3 ~]# docker attach d0e874a78177

[root@centos-6dbdb4689c-9wrcm /]#

kubectl Commands

Lab32- kubectl attach from centos pod and container , Login Bash

kubectl get pod

```
NAME                      READY   STATUS    RESTARTS   AGE
centos-6dbdb4689c-9wrcm   1/1     Running   1          5m47s
```

kubectl attach centos-6dbdb4689c-9wrcm -c centos -it

```
If you don't see a command prompt, try pressing enter.
[root@centos-6dbdb4689c-9wrcm /]#
```

OR

kubectl attach centos-6dbdb4689c-9wrcm -i

```
Defaulting container name to centos.
Use 'kubectl describe pod/centos-6dbdb4689c-9wrcm -n default' to see all of the containers in this pod.
If you don't see a command prompt, try pressing enter.
[root@centos-6dbdb4689c-9wrcm /]#
```

NOTE : Ctrl + p + q : not stop pod and container

kubectl Commands

kubectl logs from centos pod and container

kubectl logs -f POD_NAME

Lab33- kubectl logs from centos pod

Login Bash1 :

kubectl run -it centos --image=centos -- /bin/bash
[root@centos-6dbdb4689c-w8t8m /]#

Login Bash2 :

kubectl get pod

```
NAME                      READY   STATUS    RESTARTS   AGE
centos-6dbdb4689c-9wrcm   1/1     Running   1          36m
```

kubectl Commands

Lab33- kubectl logs from centos pod

Login Bash1 : Type Command

[root@centos-6dbdb4689c-w8t8m /]#

Enter

ls

bin etc lib lost+found mnt proc run srv tmp var
dev home lib64 media opt root sbin sys usr

Login Bash2 : Wait and View

kubectl logs -f centos-6dbdb4689c-w8t8m **-f : View Online Logs**

Enter

ls

bin etc lib lost+found mnt proc run srv tmp var
dev home lib64 media opt root sbin sys usr

kubectl Commands

Lab33- kubectl logs from centos pod

On Worker Node :

docker ps | grep centos

```
68aff15c2390    centos    "/bin/bash"    21 minutes ago    Up 21 minutes
```

docker logs -f 68aff15c2390

```
[root@centos-6dbdb4689c-qzpxg /]#
[root@centos-6dbdb4689c-qzpxg /]# ls
bin  etc  lib  lost+found  mnt  proc  run  srv  tmp  var
dev  home  lib64  media  opt  root  sbin  sys  usr
```

kubectl Commands

Get a pod's YAML with Formatting output(--output)

kubectl get pod POD_NAME -o yaml --export

Lab34- Get a pod's YAML

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  generateName: centos-6dbdb4689c-
  labels:
    pod-template-hash: 6dbdb4689c
    run: centos
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: centos-6dbdb4689c
    uid: 3279293a-5da4-4d72-b6be-7d0c10e053d9
  selfLink: /api/v1/namespaces/default/pods/centos-6dbdb4689c-9wrcm
spec:
  containers:
  - args:
    - /bin/bash
    image: centos
    imagePullPolicy: Always
    name: centos
```

Note :
--export : has been deprecated

kubectl Commands

Lab35- Run pod nginx and write its spec into a file called pod.yaml

kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml > pod.yaml

cat pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
```

kubectl Commands

Project :

Run Pod with :

- ✓ **Name : yourname**
- ✓ **Image : centos**
- ✓ **Never Restart**
- ✓ **Check Run Pod**
- ✓ **Login with bash**
- ✓ **View pod status with get and describe**
- ✓ **Export to Yaml**

kubectl Commands

Project :

- ✓ **kubectl run -it Anisa --image=centos --restart=Never -- /bin/bash**

  kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
  If you don't see a command prompt, try pressing enter.

  [root@kayvan-6dff944b97-z6slc /]#Ctrl + p + q

- ✓ **kubectl run -it Anisa --image=centos --restart=Never --dry-run**

- ✓ **kubectl get pod**

NAME	READY	STATUS	RESTARTS	AGE
kayvan-6dff944b97-z6slc	1/1	Running	0	108s

- ✓ **kubectl describe pod kayvan-6dff944b97-z6slc**

  Name: kayvan-6dff944b97-z6slc
  Namespace: default
  Node: node2/192.168.198.190
  Start Time: Tue, 26 Nov 2019 07:30:42 +0330
  Labels: pod-template-hash=6dff944b97
  run=kayvan

- ✓ **kubectl run -it Anisa --image=centos --restart=Never --dry-run -o yaml > pod.yaml**

kubectl Commands

Lab36- View get pods with sort restartCount

On Master Node : Run 2 Pods (Centos , Nginx)

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80
kubectl run -it centos --image=centos -- /bin/bash

[root@node1 ~]# kubectl get pod -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
centos-6dbdb4689c-wv872	1/1	Running	0	41s	10.245.2.125	node2	<none>	<none>
nginx	1/1	Running	0	67s	10.245.1.102	node3	<none>	<none>

On Worker Node2 :

docker ps | grep centos

dcafe10fd651 centos "/bin/bash" 1 second ago Up Less than a second

docker kill dcafe10fd651 ➔ Restart Again

docker ps | grep cent

c3a9a760dbc8 centos "/bin/bash" 1 second ago Up Less than a second

kubectl Commands

Lab36- View get pods with sort restartCount

On Master Node :

kubectl get pod

```
NAME                          READY   STATUS    RESTARTS   AGE
centos-6dbdb4689c-wv872       1/1     Running   1          16m
nginx                         1/1     Running   0          17m
```

kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

```
NAME                          READY   STATUS    RESTARTS   AGE
nginx                         1/1     Running   0          17m
centos-6dbdb4689c-wv872       1/1     Running   1          16m
```

Note : inverse ??

kubectl Commands

Lab36- view get events with sort by Last Time

On Master Node :

kubectl get events

LAST	SEEN	TYPE	REASON	OBJECT	MESSAGE
54m	Normal	Pulling		pod/centos-6dbdb4689c-2kgl4	Pulling image "centos"
54m	Normal	Pulled		pod/centos-6dbdb4689c-2kgl4	Successfully pulled image "centos"
54m	Normal	Created		pod/centos-6dbdb4689c-2kgl4	Created container centos
54m	Normal	Started		pod/centos-6dbdb4689c-2kgl4	Started container centos
30m	Normal	Killing		pod/centos-6dbdb4689c-2kgl4	Stopping container centos

kubectl get events --sort-by=.metadata.creationTimestamp

LAST	SEEN	TYPE	REASON	OBJECT	MESSAGE
10h	Normal		SuccessfulCreate	replicaset/kayvan-6dff944b97	Created pod: kayvan-6dff944b97-z6slc
10h	Normal		ScalingReplicaSet	deployment/kayvan	Scaled up replica set kayvan-6dff944b97 to 1
10h	Normal		Pulling	pod/kayvan-6dff944b97-z6slc	Pulling image "centos"
10h	Normal		Created	pod/kayvan-6dff944b97-z6slc	Created container kayvan
10h	Normal		Pulled	pod/kayvan-6dff944b97-z6slc	Successfully pulled image "centos"
10h	Normal		Started	pod/kayvan-6dff944b97-z6slc	Started container kayvan
9h	Normal		Killing	pod/kayvan-6dff944b97-z6slc	Stopping container kayvan

kubectl Commands

Lab36- view Only Pod Running

On Master Node :

```
[root@node1 ~]# kubectl  get pod
NAME                          READY   STATUS              RESTARTS   AGE
centos-6dbdb4689c-kzfpj   0/1     ContainerCreating   0          30s
mariadb                       1/1     Running             0          21m
nginx222                      1/1     Running             0          76s
nginxset                      1/1     Running             0          90s

[root@node1 ~]# kubectl get pods --field-selector=status.phase=Running
NAME        READY   STATUS    RESTARTS   AGE
mariadb     1/1     Running   0          22m
nginx222    1/1     Running   0          78s
nginxset    1/1     Running   0          92s
```

kubectl Commands

ADD LABEL To Pod

kubectl label pods Pod_Name new-label=Label_Name

Lab37- Add Label To Centos_Pod :

kubectl run -it centos --image=centos -- /bin/bash

kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
centos-6dbdb4689c-6t7xp	1/1	Running	0	3m24s

kubectl Commands

Lab37- Add Label To Centos_Pod :

kubectl describe pod centos-6dbdb4689c-6t7xp | grep label
no label

kubectl label pods centos-6dbdb4689c-6t7xp new-label=mycentossss
pod/centos-6dbdb4689c-6t7xp labeled

kubectl describe pod centos-6dbdb4689c-6t7xp | grep label
Labels: new-label=mycentossss

kubectl Commands

AutoScale Deployment

kubectl autoscale deployment Deployment_Name --min=No --max=No

Lab38- AutoScale Deployment:

```
kubectl run -it centos2 --image=centos  -- /bin/bash
    Ctrl+p+q

kubectl get pods
    NAME                    READY   STATUS    RESTARTS   AGE
    centos2-74456dd9b9-pssm9   1/1     Running   0          17s
```

kubectl Commands

Lab38- AutoScale Deployment:

kubectl autoscale deployment centos2 --min=3 --max=10

horizontalpodautoscaler.autoscaling/centos2 autoscaled

kubectl get pod

NAME	READY	STATUS	RESTARTS	AGE
centos2-74456dd9b9-lgmk9	1/1	Running	0	9s
centos2-74456dd9b9-mql9m	1/1	Running	0	9s
centos2-74456dd9b9-pssm9	1/1	Running	0	69s

Auto Scale to 10 With Condition

kubectl Commands

Lab38- AutoScale Deployment:

On Workers

Node2 :
docker ps | grep bash

b621de45aeef centos "/bin/bash" 5 minutes ago Up 5 minutes k8s_centos2_centos2-74456dd9b9-mql9m_default_f0753bf5-e104-47bf-9a5b-ef1f8db5b090_0
55345da7c796 centos "/bin/bash" 6 minutes ago Up 6 minutes k8s_centos2_centos2-74456dd9b9-pssm9_default_5484ba11-ef58-4f1c-bfaf-40d96ca8ed50_0

Node3 :
docker ps | grep bash

9d6e5adcc217 centos "/bin/bash" 7 minutes ago Up 7 minutes k8s_centos2_centos2-74456dd9b9-lgmk9_default_2b076f99-5ddd-41a7-b3ce-c177fcfd8fd9_0

kubectl Commands

Kubectl output verbosity and debugging

Verbosity	Description
--v=0	Generally useful for this to always be visible to a cluster operator.
--v=1	A reasonable default log level if you don't want verbosity.
--v=2	Useful steady state information about the service and important log messages that may correlate to significant changes in the system. This is the recommended default log level for most systems.
--v=3	Extended information about changes.
--v=4	Debug level verbosity.
--v=6	Display requested resources.
--v=7	Display HTTP request headers.
--v=8	Display HTTP request contents.
--v=9	Display HTTP request contents without truncation of contents.

Sample : kubectl describe pod nginx --v=9

kubectl Commands

KUBECTL SET ENV LIST

kubectl run --generator=run-pod/v1 nginx --image=nginx --port 80 --env="VAR1=10"

pod/nginx created

kubectl set env pods --all --list

# Pod nginx, container nginx
VAR1=10

kubectl Commands

Show pods status online

Lab39- kubectl --watch OR -w

kubectl get pod --watch

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-85ff79dd56-7dfk8   1/1     Running   2          31h
nginx-deployment-85ff79dd56-dqkpg   1/1     Running   2          31h
nginx-deployment-85ff79dd56-zg5ll   1/1     Running   2          31h
```

kubectl Commands

KUBECTL Command USED in YAML FILE

```
kubectl apply -f ns.yml
kubectl apply -f https://k8s.io/examples/application/nginx-app.yaml

kubectl delete -f https://k8s.io/examples/application/nginx-app.yaml

kubectl replace -f ./pod.json
kubectl replace  --force  -f   ./pod.json
```

Introduce Manifest and YAML

Manifests :

Kubernetes Manifests are used to

- Create
- Modify
- Delete

} Kubernetes resources

Such as :

pods , deployments , services or ingresses , …

Create , Modify , Delete Any Kinds With YAML OR JSON

YAML :

YAML, which stands for Yet Another Markup Language

It is very common to define manifests in form of ".yaml" files

And send them to the Kubernetes API Server via Commands

Such as

kubectl apply -f my-file.yaml

or

kubectl delete -f my-file.yaml.

Introduce Manifest and YAML

Using YAML for K8s definitions gives you a number of advantages, including:

- **Convenience: You'll no longer have to add all of your parameters to the command line**
- **Maintenance: YAML files can be added to source control, so you can track changes**
- **Flexibility: You'll be able to create much more complex structures using YAML than you can on the command line**

Notes :

NEVER use Tabs in a YAML file.
"" In YAML Optional

There are only Two Types of structures you need to know about in YAML:

- ✓ **Lists**
- ✓ **Maps**

Introduce Manifest and YAML

MAPS Structure FOR YAML AND JSON :

Yaml Structure :

```
---
apiVersion: v1 ==> MAPS
kind: Pod      ==> Maps  (NAME: Value)
```

JSON Structure :

```
{
  "apiVersion": "v1",
  "kind": "Pod"
}
```

Introduce Manifest and YAML

LISTS Structure FOR YAML AND JSON :

Yaml Structure :

```
args:
  - sleep  ==> Lists (- Value)
  - "1000"
  - message
  - "Bring back Firefly!"
```

A YAML list, which is defined as items that start with a dash (-) indented from the parent.

JSON Structure :

```
{
  "args": ["sleep", "1000", "message", "Bring back Firefly!"]
}
```

Introduce Manifest and YAML

So let's review. We have:

- **maps, which are groups of name-value pairs**
- **lists, which are individual items**
- **maps of maps**
- **maps of lists**
- **lists of lists**
- **lists of maps**

Introduce Manifest and YAML

SAMPLE OF MAPS AND LISTS :

```
---
apiVersion: v1 ================> Maps
kind: Pod
metadata:
  name: rss-site
  labels:
    app: web
spec: =======================> MAPs of Maps
  containers:  =================> MAPs of Lists
    - name: front-end   ===========> Lists of Maps
      image: nginx
      ports:
        - containerPort: 80
    - name: rss-reader
      image: nickchase/rss-php-nginx:v1
      ports:
        - containerPort:  ===========>    Lists of Lists
                                - 88    ========> Lists
```

Introduce Manifest and YAML

How to Write YAML :

```
---
apiVersion: v1 =======> Relation to Kinds . (V1 = POD)
kind: Pod  ==========> Type of Tasks . include ( Deployment, Job, Service, … )
metadata: ==========> SET Global Name OR Label ,…
 name: nginx
 labels:
   app: web
spec:   => Detail of Kind Tasks Parameters (containers, storage volumes , ...) ====>
 containers:
   - name: nginx
     image: nginx
     ports:
       - containerPort: 80
```

name
image
command
args
workingDir
ports
env
resources
volumeMounts
livenessProbe
readinessProbe
lifecycle
imagePullPolicy
securityContext
stdin
stdinOnce
tty

Introduce Manifest and YAML

Lab40- Write and Running NGINX POD with YAML FILE :

1- mkdir /manifest
2- cd /manifest
3- vim pod.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx
```

Introduce Manifest and YAML

Lab40- Write and Running NGINX POD with YAML FILE :

4- kubectl create -f pod.yaml

```
pod/nginx created
```

5- kubectl get pod

```
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          7s
```

6- kubectl get pod

```
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          11s
```

Introduce Manifest and YAML

Lab40- Write and Running NGINX POD with YAML FILE :

NOTE : Troubleshooting pod creation
IF Not Start View it (ErrImagePull)

7- kubectl get pods

```
NAME      READY   STATUS        RESTARTS   AGE
nginx     1/2     ErrImagePull  0          9s
```

OR

8- kubectl describe pod nginx

Show Events

```
Events:
  Type    Reason     Age        From               Message
  ----    ------     ----       ----               -------
  Normal  Scheduled  <unknown>  default-scheduler  Successfully assigned default/nginx to node2
  Normal  Pulling    17s        kubelet, node2     Failed to pull image "nginx“
```

9- kubectl delete pod nginx

pod "nginx" deleted

Introduce Manifest and YAML

Lab41- Write and Running Centos POD with YAML FILE :

1- vim centos_pod.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: centos
spec:
  containers:
    - name: centos
      image: centos
      stdin: true
      tty: true
```

Note :

- stdin: true
- tty: true

Introduce Manifest and YAML

Lab41- Write and Running Centos POD with YAML FILE :

2- kubectl apply -f centos_pod.yaml

pod/centos created

3- docker ps | grep cent

4df0fe4c36d2 centos "/bin/bash" 9 seconds ago Up 9 seconds k8s_centos_centos_default_1edbcf1f-f6fc-46e4-9c0d-75d5f689e80e_0

910418a46d73 k8s.gcr.io/pause:3.1 "/pause" 19 seconds ago Up 18 seconds k8s_POD_centos_default_1edbcf1f-f6fc-46e4-9c0d-75d5f689e80e_0

Introduce Manifest and YAML

Lab42- Write and Running Centos POD with EntryPoint in YAML FILE :

1- vim centos_ent_pod.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: centos
spec:
  containers:
    - name: centos
      image: centos
      command: ["/bin/bash","-c","ping -c 10 4.2.2.4"]
```

Introduce Manifest and YAML

Lab42- Write and Running Centos POD with EntryPoint in YAML FILE :

2- kubectl apply -f centos_ent_pod.yaml

pod/centos created

3- docker ps | grep cent

41128f4fcf2f centos "/bin/bash -c 'ping …" 36 seconds ago Up 36 seconds k8s_centos_centos_default_0bc87cc3-6ec2-4cef-89f7-4c841d2d80f5_0

1aff3e850cd7 k8s.gcr. io/pause:3.1 "/pause" 50 seconds ago Up 50 seconds k8s_POD_centos_default_0bc87cc3-6ec2-4cef-89f7-4c841d2d80f5_0

4- docker exec -it 41128f4fcf2f bash

[root@centos /]#

5- kubectl get pod --watch | grep centos1

centos1 1/1 Running 2 55s

centos1 0/1 Completed 2 63s

Note :

command: ["/bin/bash","-c","while true; do sleep 1000; done"]

Introduce Manifest and YAML

Lab43- Write and Running NGINX and Centos container in 1 pod with YAML FILE:

1- vim 2_container_in_1_pod.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: nginx
      image: nginx
    - name: centos
      image: centos
      stdin: true
      tty: true
```

Introduce Manifest and YAML

Lab43- Write and Running NGINX and Centos container in 1 pod with YAML FILE:

2- kubectl apply -f 2_container_in_1_pod.yaml

pod/my-pod created

3- kubectl get pods -w

```
NAME     READY   STATUS    RESTARTS   AGE
my-pod   2/2     Running   0          26s
```

4- On node 2 :

```
docker ps | grep my
4f8cf5fb7e93   centos   "/bin/bash"   37 seconds ago   Up 36 seconds   k8s_centos_my-pod_default_666c41b1-45f6-4aab-a2e3-041f89313920_0
0bd30e0c8c81   nginx    "nginx -g 'daemon of…"   45 seconds ago   Up 44 seconds   k8s_nginx_my-pod_default_666c41b1-45f6-4aab-a2e3-041f89313920_0
afd0b8ba90ee   k8s.gcr.io/pause:3.1   "/pause"   55 seconds ago   Up 54 seconds   k8s_POD_my-pod_default_666c41b1-45f6-4aab-a2e3-041f89313920_0
```

Introduce Manifest and YAML

Lab44- Make NGINX DEPLOYMENT with YAML FILE

1- vim deployment.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
```

Deployment Used : apps/v1 and Selector

Introduce Manifest and YAML

Lab44- Make NGINX DEPLOYMENT with YAML FILE

2- kubectl create -f deployment.yaml

deployment.apps/nginx-deployment created

3- kubectl get deployments.apps

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           8m54s
```

Introduce Manifest and YAML

Project :

Start 3 Pod centos with Deployment Yaml Model?

Introduce Manifest and YAML

Lab45- Run Pod , Do Not Again Pull Image

ImagePullPolicy:

- **IfNotPresent :** The image is pulled only if it is not already present locally
- **Never :** The image is assumed to exist locally. No attempt is made to pull the image. ➔ ErrImageNeverPull
- **Always (Default):** The image is pulled every time the pod is started.

Default Image Version : latest

Introduce Manifest and YAML

Lab45- Run Pod , Do Not Again Pull Image

1- vim pod-no_pull_image.yaml

```
---
 apiVersion: v1
 kind: Pod
 metadata:
   name: nginx
 spec:
   containers:
     - name: nginx
       image: nginx
       imagePullPolicy: IfNotPresent
```

Introduce Manifest and YAML

Lab45- Run Pod , Do Not Again Pull Image

2- kubectl apply -f pod-no_pull_image.yaml

pod/nginx created

3- On Node 2 : Without Download Image Again (VPN Disconnect For Test)

watch -d -n 1 "docker ps | grep nginx"

6bf262e209f7 540a289bab6c "nginx -g 'daemon of…" 49 seconds ago Up 49 seconds k8s_nginx_nginx_default_d7f4d61f-fd45-4467-ba9b-ee6a602e9911_0

36ee9c66ce60 k8s.gcr.io/pause:3.1 "/pause" 50 seconds ago Up 49 seconds k8s_POD_nginx_default_d7f4d61f-fd45-4467-ba9b-ee6a602e9911_0

4- Delete pod :

kubectl delete -f pod-no_pull_image.yaml

Or

kubectl delete pod nginx

Introduce Manifest and YAML

Lab46- Container restartPolicy mode

restartPolicy:

- ❑ **Never**
- ❑ **Always ==> Default**

Introduce Manifest and YAML

Lab46- Container restartPolicy mode

1- vim pod_restart_Policy.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
```

Introduce Manifest and YAML

Lab46- Container restartPolicy mode

2- kubectl apply -f pod_restart_Policy.yaml

3- docker ps | grep ngin

4b0217c00c0a 231d40e811cd "nginx -g 'daemon of…" 58 seconds ago Up 58 seconds k8s_nginx_nginx_default_ecb40c5e-abb8-475d-947e-629e82b6ab1c_2
06e75b750f76 k8s.gcr.io/pause:3.1 "/pause" 59 seconds ago Up 58 seconds k8s_POD_nginx_default_ecb40c5e-abb8-475d-947e-629e82b6ab1c_2

4- docker kill 06

06

5- docker ps | grep ngin ➔ START AGAIN

c141c81efddb 231d40e811cd "nginx -g 'daemon of…" 1 second ago Up Less than a second k8s_nginx_nginx_default_ecb40c5e-abb8-475d-947e-629e82b6ab1c_3
ecdf4b993a2f k8s.gcr.io/pause:3.1 "/pause" 1 second ago Up Less than a second k8s_POD_nginx_default_ecb40c5e-abb8-475d-947e-629e82b6ab1c_3

Introduce Manifest and YAML

Lab47- Container restartPolicy mode

1- vim pod_restart_Policy.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  restartPolicy: Never
```

Introduce Manifest and YAML

Lab47- Container restartPolicy mode

2- kubectl apply -f pod_restart_Policy.yaml

3- docker ps | grep ngin

4b0217c00c0a 231d40e811cd "nginx -g 'daemon of…" 58 seconds ago Up 58 seconds k8s_nginx_nginx_default_ecb40c5e-abb8-475d-947e-629e82b6ab1c_2
06e75b750f76 k8s.gcr.io/pause:3.1 "/pause" 59 seconds ago Up 58 seconds k8s_POD_nginx_default_ecb40c5e-abb8-475d-947e-629e82b6ab1c_2

4- docker kill 06

06

5- docker ps | grep ngin ➔ Do Not START AGAIN

Introduce Manifest and YAML

Which apiVersion should I use?

WORKLOADS APIS

Container v1 core

CronJob v1beta1 batch

DaemonSet v1 apps

Deployment v1 apps

Job v1 batch

Pod v1 core

ReplicaSet v1 apps

ReplicationController v1 core

StatefulSet v1 apps

SERVICE APIS

Endpoints v1 core

EndpointSlice v1alpha1 discovery.k8s.io

Ingress v1beta1 networking.k8s.io

Service v1 core

CONFIG AND STORAGE APIS

ConfigMap v1 core

CSIDriver v1beta1 storage.k8s.io

CSINode v1beta1 storage.k8s.io

Secret v1 core

PersistentVolumeClaim v1 core

StorageClass v1 storage.k8s.io

Volume v1 core

VolumeAttachment v1 storage.k8s.i

Kubernetes Volume Disk

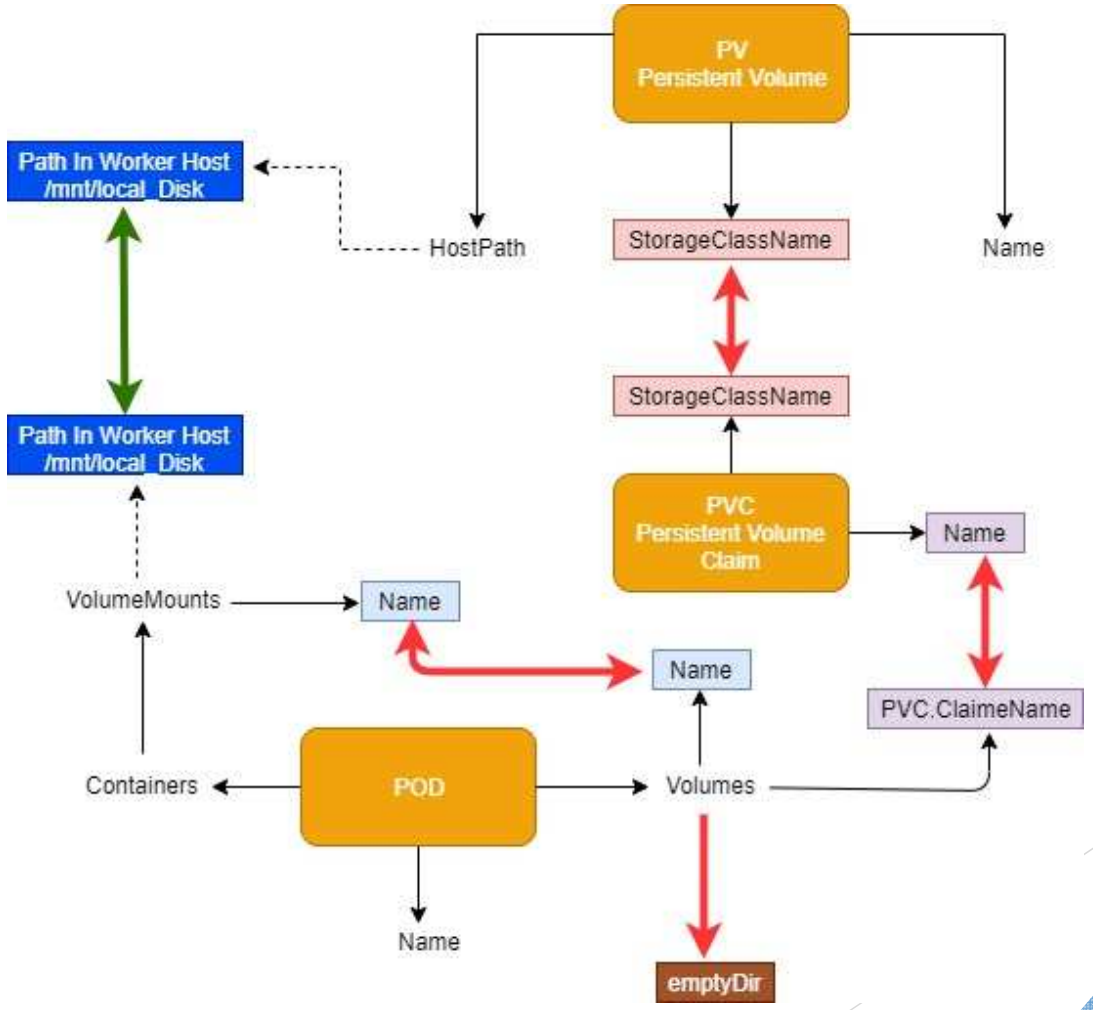
Volume Solves :

- ✓ **First, when a Container crashes, kubelet will restart it, but the files will be lost - the Container starts with a clean state.**

- ✓ **Second, when running Containers together in a Pod it is often necessary to share files between those Containers. The Kubernetes Volume abstraction solves both of these problems.**

- ✓ **Kubernetes Managing storage:**
  - ➢ PersistentVolume
  - ➢ PersistentVolumeClaim.

Kubernetes Volume Disk



Kubernetes Volume Disk

1-

```yaml
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-local
spec:
  storageClassName: scn-local
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/local"
```

2-

```yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-local
spec:
  storageClassName: scn-local
  resources:
    requests:
      storage: 1Gi
  accessModes:
    - ReadWriteOnce
```

3-

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-pv-local
spec:
  volumes:
    - name: pv-to-pod-relation
      persistentVolumeClaim:
        claimName: pvc-local
  containers:
    - name: pv-to-pod
      image: nginx
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: pv-to-pod-relation
```

Kubernetes Volume Disk

48- Create emptyDir Volume in Container POD :

1- vim pod_emptyDir.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
       - mountPath: /mnt/empty-dir
         name: emptydir-vol
  volumes:
  - name: emptydir-vol
    emptyDir: {}
```

Kubernetes Volume Disk

48- Create emptyDir Volume in Container POD :

2- kubectl apply -f pod_emptyDir.yaml

pod/nginx created

3- On Node 2 :

docker ps | grep nginx

bfb04eb098e7 nginx "nginx -g 'daemon of…" 3 seconds ago Up 3 seconds k8s_nginx_nginx_default_77e54973-1577-4d54-9a68-6b2c1a7c2557_0

239850d04cf4 k8s.gcr.io/pause:3.1 "/pause" 11 seconds ago Up 10 seconds k8s_POD_nginx_default_77e54973-1577-4d54-9a68-6b2c1a7c2557_0

4- docker exec -it bf bash

root@nginx:/# ls /mnt/empty-dir/

Kubernetes Volume Disk

49- Create emptyDir Volume For Sharing Files between 2 Container in POD :

1- vim 2_Container_emptyDir.yaml

In Cent1 :
/mnt/empty-dir1

In Cent2 :
/mnt/empty-dir2

```
---
apiVersion: v1
kind: Pod
metadata:
  name: centos
spec:
  containers:
    - name: cent1
      image: centos
      stdin: true
      tty: yes
      volumeMounts:
       - mountPath: /mnt/empty-dir1
         name: emptydir-vol

    - name: cent2
      image: centos
      stdin: true
      tty: yes
      volumeMounts:
       - mountPath: /mnt/empty-dir2
         name: emptydir-vol

  volumes:
  - name: emptydir-vol
    emptyDir: {}
```

Kubernetes Volume Disk

49- Create emptyDir Volume For Sharing Files between 2 Container in POD :

2- kubectl apply -f 2_Container_emptyDir.yaml

pod/nginx created

3- kubectl get pod -w

```
NAME     READY   STATUS    RESTARTS   AGE
centos   2/2     Running   0          15m
```

4- On Node 2 :

docker ps | grep cent

```
66c8bad1042e   centos   "/bin/bash"   9 seconds ago   Up 9 seconds
k8s_cent2_centos_default_ed761b71-5a07-4c6a-8f24-5798f32e43ad_0
f301ac14f139   centos   "/bin/bash"   17 seconds ago   Up 16 seconds
k8s_cent1_centos_default_ed761b71-5a07-4c6a-8f24-5798f32e43ad_0
```

5- docker exec -it 66 bash

[root@centos /]# ls /mnt/empty-dir2/

qqq

6- docker exec -it f3 bash

[root@centos /]# touch /mnt/empty-dir1/qqq

Kubernetes Volume Disk

50- Create emptyDir Volume in Container DEPLOYMENT :

1- vim deploy_emptyDir.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-pv-local
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: "nginx"
        volumeMounts:
        - name: emptydir-vol
          mountPath: /mnt/empty_dir_deploy
      volumes:
      - name: emptydir-vol
        emptyDir: {}
```

Kubernetes Volume Disk

50- Create emptyDir Volume in Container DEPLOYMENT :

2- kubectl get deployments.apps

```
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
deploy-pv-local   5/5     5            5           114s
```

3- kubectl get deployments.apps -o wide

```
NAME            READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS   IMAGES   SELECTOR
deploy-pv-local   5/5     5            5           2m8s   nginx        nginx    app=nginx
```

4- kubectl get pod

```
NAME                              READY   STATUS    RESTARTS   AGE
deploy-pv-local-694d895cff-b6vzp   1/1     Running   0          2m25s
deploy-pv-local-694d895cff-dl29x   1/1     Running   0          2m25s
deploy-pv-local-694d895cff-gh6r5   1/1     Running   0          2m25s
deploy-pv-local-694d895cff-qrnhz   1/1     Running   0          2m25s
deploy-pv-local-694d895cff-x2r2w   1/1     Running   0          2m25s
```

Kubernetes Volume Disk

50- Create emptyDir Volume in Container DEPLOYMENT :

5- On Node 3 :

docker ps | grep nginx

```
fcc878e317e9       nginx              "nginx -g 'daemon of…"   5 seconds ago       Up 5 seconds                    k8s_nginx_deploy-pv-local-694d895cff-x2r2w_default_755128a5-c891-452f-9c45-d1c044a0db55_0
da92a673e161       nginx              "nginx -g 'daemon of…"   18 seconds ago      Up 18 seconds                   k8s_nginx_deploy-pv-local-694d895cff-dl29x_default_b3a751c1-b4cc-45df-971b-7410c34e2235_0
```

On Node 2 :

docker ps | grep nginx

```
f2cfbbb92fe9       nginx              "nginx -g 'daemon of…"   3 minutes ago       Up 2 minutes                    k8s_nginx_deploy-pv-local-694d895cff-gh6r5_default_01ef397f-567c-4dfb-aac5-41e993f5dc1b_0
014eabd46e48       nginx              "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes                    k8s_nginx_deploy-pv-local-694d895cff-b6vzp_default_00b097c1-d1e3-4d4b-969b-848823e2b716_0
ed30bedd7a62       nginx              "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes                    k8s_nginx_deploy-pv-local-694d895cff-qrnhz_default_de860e42-5fd9-4eab-9dda-6b4b262773f4_0
```

6- docker exec -it f2cfbbb92fe9 bash

root@deploy-pv-local-694d895cff-gh6r5:/# **ls /mnt/empty_dir_deploy/**

Kubernetes Volume Disk

51- Create persistentvolume For POD :

1- vim pv.yaml

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-local
spec:
  storageClassName: scn-local
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/local"
```

Kubernetes Volume Disk

51- Create persistentvolume For POD :

2- kubectl apply -f pv.yaml

persistentvolume/pv-local created

3- kubectl get pv

```
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv-local   2Gi        RWO            Retain           Available           scn-local              7s
```

4- kubectl get persistentvolume

```
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM            STORAGECLASS   REASON   AGE
pv-local   2Gi        RWO            Retain           Bound    default/pvc-local   scn-local              15m
```

5- kubectl get pv pv-local

```
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv-local   2Gi        RWO            Retain           Available           scn-local              35s
```

Kubernetes Volume Disk

52- Create persistentvolumeclaims For POD :

1- vim pvc.yaml

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-local
spec:
  storageClassName: scn-local
  resources:
    requests:
      storage: 1Gi
  accessModes:
    - ReadWriteOnce
```

Kubernetes Volume Disk

52- Create persistentvolumeclaims For POD :

2- kubectl apply -f pvc.yaml

persistentvolumeclaim/pvc-local created

3- kubectl get pvc

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-local	Bound	pv-local	2Gi	RWO	scn-local	21s

4- kubectl get persistentvolumeclaims

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-local	Bound	pv-local	2Gi	RWO	scn-local	33s

5- kubectl get persistentvolumeclaims pvc-local

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-local	Bound	pv-local	2Gi	RWO	scn-local	3m50s

Kubernetes Volume Disk

53- Create POD and Connect to persistentvolumeclaims :

1- vim pod.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-pv-local
spec:
  volumes:
    - name: pv-to-pod-relation
      persistentVolumeClaim:
        claimName: pvc-local
  containers:
    - name: pv-to-pod
      image: nginx
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: pv-to-pod-relation
```

Kubernetes Volume Disk

53- Create POD and Connect to persistentvolumeclaims :

2- kubectl apply -f pod.yaml

pod/pod-pv-local created

Note : Pod started Automatic Create Path /mnt/local in one of Workers

3- kubectl get pod

```
NAME          READY   STATUS    RESTARTS   AGE     NODE    NOMINATED NODE   READINESS GATES
pod-pv-local  1/1     Running   0          8m23s   node2   <none>           <none>
```

4- On Worker Node2 :

docker ps | grep nginx

```
f1838815767a   nginx   "nginx -g 'daemon of…"   8 minutes ago   Up 8 minutes   k8s_pv-to-pod_pod-pv-local_default_02f2d492-68ff-44b7-b585-7699527bb9a3_0
```

Kubernetes Volume Disk

53- Create POD and Connect to persistentvolumeclaims :

5- docker exec -it f1 bash

```
root@pod-pv-local:/# apt-get update
root@pod-pv-local:/# apt-get install curl
root@pod-pv-local:/# cd /usr/share/nginx/html
root@pod-pv-local:/# echo Hi > index.html
root@pod-pv-local:/# curl http://localhost/index.html
Hi
```

6- exit to Node2 :

```
[root@node2 ~]# ll /mnt/local/
-rw-r--r-- 1 root root 2 Oct 31 18:27 index.html
```

Kubernetes Volume Disk

53- Create POD and Connect to persistentvolumeclaims :

For Delete Pod :

7- kubectl get all

```
NAME              READY   STATUS    RESTARTS   AGE
pod/pod-pv-local   1/1     Running   0          70m
```

8- kubectl delete pod pod-pv-local

```
pod "pod-pv-local" deleted
```

9- kubectl get pvc

```
NAME        STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc-local   Bound    pv-local   2Gi        RWO            scn-local      73m
```

10- kubectl delete pvc pvc-local

```
persistentvolumeclaim "pvc-local" deleted
```

Kubernetes Volume Disk

53- Create POD and Connect to persistentvolumeclaims :

For Delete Pod :

11- kubectl get pv

```
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pv-local   2Gi       RWO            Retain           Bound    default/pvc-local   scn-local              72m
```

12- kubectl delete pv pv-local

persistentvolume "pv-local" deleted

Note : metadata

name: pv-local Name is OK

name: pv-local_1 Name is NOT OK

Kubernetes Volume Disk

54- Create persistentvolume For DEPLOYMENT :

1- vim pv.yaml

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-local-deploy
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: scn-local-deploy
  hostPath:
    path: /mnt/local-deploy
```

Kubernetes Volume Disk

54- Create persistentvolume For DEPLOYMENT :

2- kubectl get pv

```
NAME              CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                      STORAGECLASS      REASON   AGE
pv-local-deploy   3Gi        RWO            Retain           Bound    default/pvc-local-deploy   scn-local-deploy          30m
```

3- kubectl get pv -o wide

```
NAME              CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                      STORAGECLASS
REASON   AGE   VOLUMEMODE
pv-local-deploy   3Gi        RWO            Retain           Bound    default/pvc-local-deploy   scn-local-deploy          30m
Filesystem
```

Kubernetes Volume Disk

55- Create persistentvolumeclaims For DEPLOYMENT :

1- vim pvc.yaml

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-local-deploy
spec:
  storageClassName: scn-local-deploy
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Kubernetes Volume Disk

55- Create persistentvolumeclaims For DEPLOYMENT :

2- kubectl get pvc

```
NAME                STATUS   VOLUME          CAPACITY   ACCESS MODES   STORAGECLASS       AGE
pvc-local-deploy    Bound    pv-local-deploy   3Gi        RWO            scn-local-deploy   32m
```

3- kubectl get pvc -o wide

```
NAME                STATUS   VOLUME          CAPACITY   ACCESS MODES   STORAGECLASS       AGE   VOLUMEMODE
pvc-local-deploy    Bound    pv-local-deploy   3Gi        RWO            scn-local-deploy   33m   Filesystem
```

Kubernetes Volume Disk

56- Create DEPLOYMENT and Connect to persistentvolumeclaims :

1- vim deploy.yaml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-pv-local
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: pv-to-deploy
        image: "nginx"
        volumeMounts:
        - name: pv-to-deploy-relation
          mountPath: "/usr/share/nginx/html"
      volumes:
      - name: pv-to-deploy-relation
        persistentVolumeClaim:
          claimName: pvc-local-deploy
```

Kubernetes Volume Disk

56- Create DEPLOYMENT and Connect to persistentvolumeclaims :

2- kubectl get deployments.apps

```
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
deploy-pv-local   5/5     5            5           19m
```

3- kubectl get deployments.apps -o wide

```
NAME            READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS    IMAGES   SELECTOR
deploy-pv-local   5/5     5            5           19m   pv-to-deploy   nginx    app=nginx
```

4- kubectl get pod -o wide

```
NAME                              READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
deploy-pv-local-54f966bbcc-6f7dw   1/1     Running   0          20m   10.245.1.38   node3   <none>           <none>
deploy-pv-local-54f966bbcc-6xkqw   1/1     Running   0          20m   10.245.2.59   node2   <none>           <none>
deploy-pv-local-54f966bbcc-gf2m7   1/1     Running   0          20m   10.245.2.60   node2   <none>           <none>
deploy-pv-local-54f966bbcc-hwnqg   1/1     Running   0          20m   10.245.1.39   node3   <none>           <none>
deploy-pv-local-54f966bbcc-j4d9l   1/1     Running   0          20m   10.245.2.58   node2   <none>           <none>
```

Kubernetes Volume Disk

56- Create DEPLOYMENT and Connect to persistentvolumeclaims :

5- TEST From Node2 , Node3 :

Node 2 :

docker ps | grep nginx

```
b8545687b307       nginx              "nginx -g 'daemon of…"   19 minutes ago      Up 19 minutes
k8s_pv-to-deploy_deploy-pv-local-54f966bbcc-gf2m7_default_ce52ccaf-0d91-430a-9101-92a5f11e7897_0
b382655b1b79       nginx              "nginx -g 'daemon of…"   19 minutes ago      Up 19 minutes
k8s_pv-to-deploy_deploy-pv-local-54f966bbcc-6xkqw_default_13417083-ec22-415d-8033-f4092db8d7e4_0
0c0647297697       nginx              "nginx -g 'daemon of…"   20 minutes ago      Up 20 minutes
k8s_pv-to-deploy_deploy-pv-local-54f966bbcc-j4d9l_default_3fd89119-346f-4f26-8ec0-5515dac3c033_0
```

6- ll /mnt/local-deploy/

```
echo hi > /mnt/local-deploy/index.html
curl http://10.245.2.58
hi
```

Kubernetes Volume Disk

56- Create DEPLOYMENT and Connect to persistentvolumeclaims :

7- Node3 :

docker ps | grep nginx

```
fe7732dab8c9       nginx                  "nginx -g 'daemon of…"   21 minutes ago      Up 20 minutes
k8s_pv-to-deploy_deploy-pv-local-54f966bbcc-hwnqg_default_73392e35-8b2f-4ebf-8516-50b4a5a99eba_0
a67dffcd666d       nginx                  "nginx -g 'daemon of…"   21 minutes ago      Up 21 minutes
k8s_pv-to-deploy_deploy-pv-local-54f966bbcc-6f7dw_default_7e26385b-d09e-472a-b128-35037a18523b_0
```

8- ll /mnt/local-deploy/

curl http://10.245.1.38

Noooooo outPut

Kubernetes Volume Disk

56- Create DEPLOYMENT and Connect to persistentvolumeclaims :

For Delete Deployment , pv , pvc :

9- kubectl delete deployments.apps deploy-pv-local

deployment.apps "deploy-pv-local" deleted

10- kubectl delete pvc pvc-local-deploy

persistentvolumeclaim "pvc-local-deploy" deleted

11- kubectl delete pv pv-local-deploy

persistentvolume "pv-local-deploy" deleted

Kubernetes Volume Disk

What is Reclaiming :

when user delete the PVC objects from the API

The reclaim policy for a PersistentVolume

Type of Modes persistentVolumeReclaimPolicy

- Retained : IF PVC Delete Then Data exist in PV and Reclaim or Delete Later
- Recycled : IF PVC Delete Then rm –fr /mnt/loca-disk/*
- Deleted : IF PVC Delete Then Delete PV

Kubernetes Volume Disk

Types of Persistent Volumes

- GCEPersistentDisk
- AWSElasticBlockStore
- AzureFile
- AzureDisk
- CSI
- FC (Fibre Channel)
- FlexVolume
- Flocker
- NFS
- iSCSI
- RBD (Ceph Block Device)
- CephFS
- Cinder (OpenStack block storage)
- Glusterfs
- VsphereVolume
- Quobyte Volumes
- HostPath (Single node testing only – local storage is not supported in any way and WILL NOT WORK in a multi-node cluster)
- Portworx Volumes
- ScaleIO Volumes
- StorageOS

Kubernetes Volume Disk

PersistentVolumeClaim Access Modes

- ReadWriteOnce – the volume can be mounted as read-write by a single node
- ReadOnlyMany – the volume can be mounted read-only by many nodes
- ReadWriteMany – the volume can be mounted as read-write by many nodes

- RWO - ReadWriteOnce
- ROX - ReadOnlyMany
- RWX - ReadWriteMany

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-local-deploy
spec:
  storageClassName: scn-local-deploy
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Kubernetes Network

57- Make NGINX POD with YAML FILE and Network Service

1- mkdir /manifest/network
   cd /manifest/network

2- Create Pod :
   vim pod_net.yaml

```
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx    ===> Selector in Pod_Service.yaml
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
          name: "http"    ===> Name = Number and lowercase Character
```

Kubernetes Network

57- Make NGINX POD with YAML FILE and Network Service

3- kubectl apply -f pod_net.yaml

pod/nginx created

4- docker ps | grep ngin

```
9cbe7dd81163       nginx                  "nginx -g 'daemon of…"   About a minute ago   Up About a minute
k8s_nginx_nginx_default_385cf1fd-e1ad-4c71-8184-0278c5b41588_0
846849c57c6c       k8s.gcr.io/pause:3.1   "/pause"                 About a minute ago   Up About a minute
k8s_POD_nginx_default_385cf1fd-e1ad-4c71-8184-0278c5b41588_0
```

Kubernetes Network

57- Make NGINX POD with YAML FILE and Network Service

5- Create Service :

vim pod_service.yaml

```
---
apiVersion: v1
kind: Service
metadata:
  name: sr-nginx
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 32222
    targetPort: 80
    name: http
  selector:
    app: nginx   ===> Label  in pod_net.yaml
```

Kubernetes Network

57- Make NGINX POD with YAML FILE and Network Service

6- kubectl apply -f pod_service.yaml

service/sr-nginx created

7- kubectl get service

```
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        35h
sr-nginx     NodePort    10.98.164.232  <none>        8080:32222/TCP 2m33s
```

8- Test With Browser in Any Host :

http://192.168.198.190:32222/

Kubernetes Network

58- Make NGINX Deployment with YAML FILE and Network Service

1- Create Deployment :

```
cd /manifest/network
vim  deploy_net.yaml
```

```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
```

===> Selector in deploy_Service.yaml

Kubernetes Network

58- Make NGINX Deployment with YAML FILE and Network Service

2- Create Service

vim deploy_service.yaml

```
---
apiVersion: v1
kind: Service
metadata:
  name: sr-nginx
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 32222
    targetPort: 80
    name: http
  selector:
    app: nginx
```

===> Label in deploy_net.yaml

Kubernetes Network

59- Make NGINX Deployment with YAML FILE and PersistentVolume and Network Service

1- Create pv :

```
cd  /manifest/network/nginx_project
vim   pv_nginx_project.yaml
```

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nginx-prj
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: scn-nginx-prj
  hostPath:
    path: /mnt/nginx-prj
```

Kubernetes Network

59- Make NGINX Deployment with YAML FILE and PersistentVolume and Network Service

2- Create pvc :

```
vim   pvc_nginx_project.yaml
```

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nginx-prj
spec:
  storageClassName: scn-nginx-prj
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Kubernetes Network

59- Make NGINX Deployment with YAML FILE and PersistentVolume and Network Service

3- Create deployment :

vim deploy_nginx.project.yaml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-nginx-prj
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: pv-to-deploy
        image: "nginx"
        volumeMounts:
        - name: pv-to-deploy-relation
          mountPath: "/usr/share/nginx/html"
      volumes:
      - name: pv-to-deploy-relation
        persistentVolumeClaim:
          claimName: pvc-nginx-prj
```

Kubernetes Network

59- Make NGINX Deployment with YAML FILE and PersistentVolume and Network Service

4- Create Service :

vim service_nginx_prj.yaml

```
---
apiVersion: v1
kind: Service
metadata:
  name: sr-nginx
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 32222
    targetPort: 80
    name: http
  selector:
    app: nginx
```

Kubernetes Network

59- Make NGINX Deployment with YAML FILE and PersistentVolume and Network Service

5- kubectl apply -f pv_nginx_project.yaml -f pvc_nginx_project.yaml -f deploy_nginx.project.yaml -f service_nginx_prj.yaml

```
persistentvolume/pv-nginx-prj created
persistentvolumeclaim/pvc-nginx-prj created
deployment.apps/deploy-nginx-prj created
service/sr-nginx created
```

Kubernetes Network

59- Make NGINX Deployment with YAML FILE and PersistentVolume and Network Service

6- kubectl get pod -w

```
NAME                                READY   STATUS    RESTARTS   AGE
deploy-nginx-prj-5dd54cdc-2qvkh     1/1     Running   0          26s
deploy-nginx-prj-5dd54cdc-b8wfd     1/1     Running   0          26s
deploy-nginx-prj-5dd54cdc-jj48m     1/1     Running   0          26s
deploy-nginx-prj-5dd54cdc-kg9mq     1/1     Running   0          26s
deploy-nginx-prj-5dd54cdc-qwrjh     1/1     Running   0          26s
```

7- kubectl get service sr-nginx

```
NAME       TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
sr-nginx   NodePort   10.100.152.236  <none>        8080:32222/TCP 77s
```

8- kubectl get service sr-nginx -o wide

```
NAME       TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
sr-nginx   NodePort   10.100.152.236  <none>        8080:32222/TCP 97s   app=nginx
```

Kubernetes Network

59- Make NGINX Deployment with YAML FILE and PersistentVolume and Network Service

9- For Test :

ON Node 2 , Node 3 :

```
echo Hi >  /mnt/nginx-prj/index.html
```

10- In Browser

http://192.168.198.190:32222/

Hi

Kubernetes Network

- **Kubernetes Networking Type :**

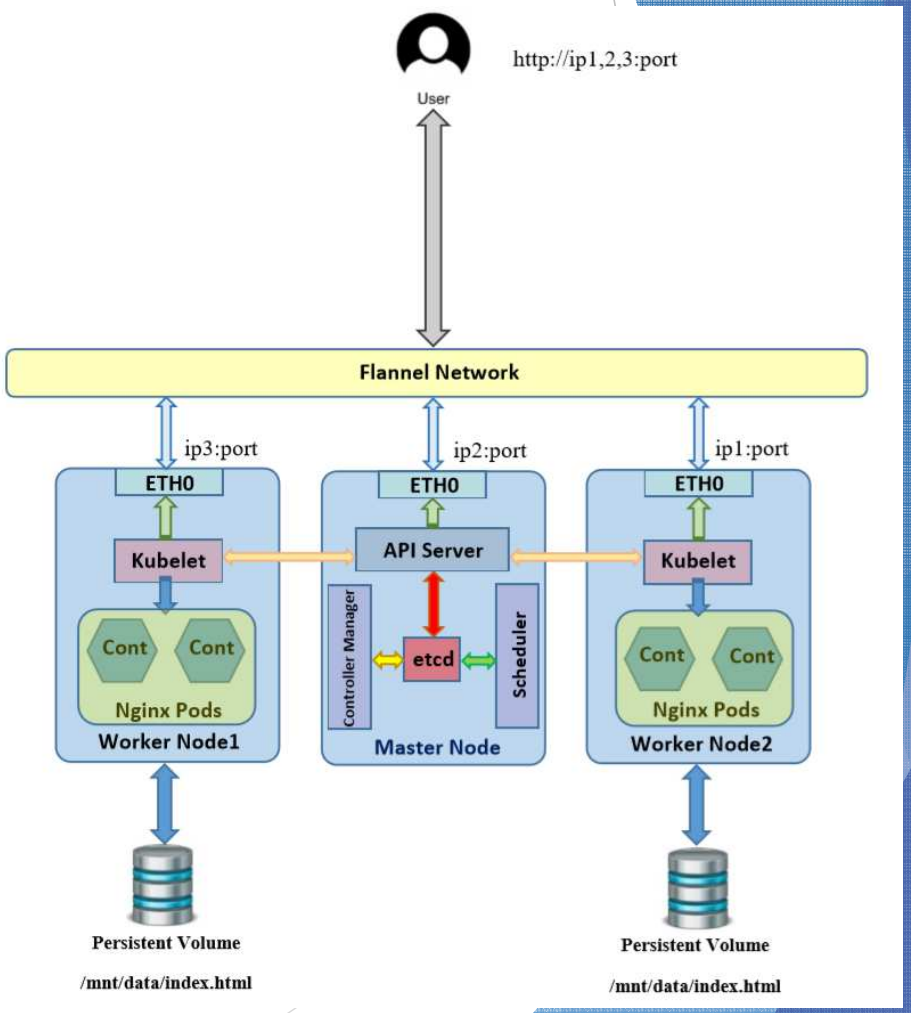
CNI	INSTALL	SECURITY	PERFS	RESOURCES
Calico	Good	Good	Very good	Good
Canal	Good	Good	Good	Fair
Cilium	Poor	Good	Fair	Poor
Flannel	Good	Poor	Good	Very good
Kube-router	Good	Fair	Very good	Good
Romana	Good	Poor	Very good	Good
WeaveNet	Good	Fair	Good	Fair
WeaveNet Crypt	Good	Very good	Poor	Poor

Very good
Good
Fair
Poor

Implementing Deployment Nginx With NFS StorageClass

Scenario : Deployment Nginx with :

1- NameSpace
2- Deployment and Replicas Scale
3- NFS StorageClass Volume
4- Service Expose
5- Test It …



Implementing Deployment Nginx With NFS StorageClass

Installing NFS Server and Clients With Test :

On NFS Server Node :

1- yum install nfs-utils
2- systemctl enable nfs-server
3- systemctl restart nfs-server
4- mkdir /var/nfs-share/nginx-html
5- chmod 755 -R /var/nfs-share/
6- chown nfsnobody:nfsnobody /var/nfs-share/
7- vi /etc/exports
   /var/nfs-share *(rw,sync,no_root_squash,no_all_squash)
8- systemctl restart nfs-server
9- exportfs –a
10- echo HI > /var/nfs-share/nginx-html/index.html
11- systemctl stop firewalld

Implementing Deployment Nginx With NFS StorageClass

Installing NFS Server and Clients With Test :

On NFS Client Nodes :

1- yum install nfs-utils
2- mkdir /var/nfs-share/nginx-html
3- systemctl stop firewalld
4- mount -t nfs 192.168.198.169:/var/nfs-share/nginx-html /var/nfs-share/nginx-html
5- ls /var/nfs-share
    -rw-r--r-- 1 root root 3 Dec 29 17:33 index.html
6- cat /var/nfs-share/nginx-html/index.html
    HI

Implementing Deployment Nginx With NFS StorageClass

Create pv-nginx.yaml :

1- vim pv-nginx.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-nginx-html
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: scn-nginx-html
  nfs:
    path: /var/nfs-share/nginx-html
    server: 192.168.198.169
  persistentVolumeReclaimPolicy: Retain
```

Implementing Deployment Nginx With NFS StorageClass

Create pv-nginx.yaml :

2- kubectl apply -f pv-nginx.yaml

persistentvolume/pv-nginx-html created

3- kubectl get pv

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv-nginx-html	5Gi	RWO	Retain	Available		scn-nginx-html		5m48s

Implementing Deployment Nginx With NFS StorageClass

Create ns-nginx.yaml :

4- vim ns-nginx.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: ns-nginx-html
```

5- kubectl apply -f ns-nginx.yaml

```
namespace/ns-nginx-html created
```

6- kubectl get namespaces ns-nginx-html

```
NAME            STATUS   AGE
ns-nginx-html   Active   3m33s
```

Implementing Deployment Nginx With NFS StorageClass

Create pvc-nginx.yaml :

7- vim pvc-nginx.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-nginx-html
  namespace: ns-nginx-html
spec:
  storageClassName: scn-nginx-html
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Implementing Deployment Nginx With NFS StorageClass

Create pvc-nginx.yaml :

8- kubectl apply -f pvc-nginx.yaml
persistentvolumeclaim/pvc-ngnix-html created

9- kubectl get pvc -n ns-nginx-html

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-nginx-html	Bound	pv-nginx-html	5Gi	RWO	scn-nginx-html	16s

Implementing Deployment Nginx With NFS StorageClass

Create dep-nginx.yaml :

10 - vim dep-nginx.yaml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-nginx-nfs
  namespace: ns-nginx-html
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: v-ngins-nfs
        image: "nginx"
        volumeMounts:
        - name: v-nginx-nfs
          mountPath: /usr/share/nginx/html
      volumes:
      - name: v-nginx-nfs
        persistentVolumeClaim:
          claimName: pvc-nginx-html
```

Implementing Deployment Nginx With NFS StorageClass

Create dep-nginx.yaml :

11- kubectl apply -f dep-nginx.yaml
deployment.apps/deploy-nginx-nfs created

12- kubectl get deploy -n ns-nginx-html

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deploy-nginx-nfs	5/5	5	5	8m2s

Implementing Deployment Nginx With NFS StorageClass

Create service-nginx.yaml :

13- vim service-nginx.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: svc-nginx
  namespace: ns-nginx-html
  labels:
    app: nginx
spec:
  type: NodePort
  ports:
  - port: 8080
    nodePort: 30000
    targetPort: 80
    name: http
  selector:
    app: nginx
```

Implementing Deployment Nginx With NFS StorageClass

Create service-nginx.yaml :

14- kubectl apply -f service-nginx.yaml
service/svc-nginx created

15- kubectl get service -n ns-nginx-html

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
svc-nginx	NodePort	10.96.119.18	<none>	8080:30000/TCP	19s

Implementing Deployment Nginx With NFS StorageClass

Test From Nodes :

16- kubectl get pod -n ns-nginx-html -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
deploy-nginx-nfs-796fb7b8fb-4n2fz	1/1	Running	0	16m	10.245.1.206	node3	<none>	<none>
deploy-nginx-nfs-796fb7b8fb-kvvqg	1/1	Running	0	16m	10.245.1.205	node3	<none>	<none>
deploy-nginx-nfs-796fb7b8fb-pjv28	1/1	Running	0	16m	10.245.2.227	node2	<none>	<none>
deploy-nginx-nfs-796fb7b8fb-vf9cr	1/1	Running	0	16m	10.245.1.204	node3	<none>	<none>
deploy-nginx-nfs-796fb7b8fb-zlszk	1/1	Running	0	16m	10.245.2.226	node2	<none>	<none>

17- From Node2 :

curl http://10.245.2.227

HI_Anisa

curl http://10.245.2.226

HI_Anisa

18- From Node3 :

curl http:// 10.245.1.206

HI_Anisa

curl http:// 10.245.1.205

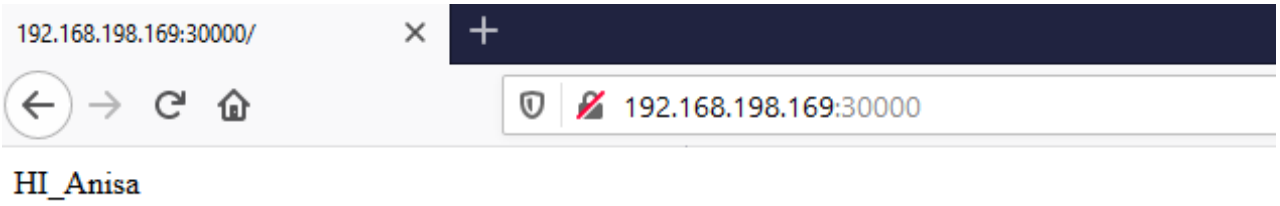
HI_Anisa

curl http:// 10.245.1.204

HI_Anisa

Implementing Deployment Nginx With NFS StorageClass

Test From Browser :



Change From any Node

```
echo  HI_Anisa > /var/nfs-share/nginx-html/index.html
```

Implementing Deployment Nginx With NFS StorageClass

19- kubectl scale deployment --replicas 3 deploy-nginx-nfs --namespace ns-nginx-html

deployment.apps/deploy-nginx-nfs scaled

20- kubectl get pod -n ns-nginx-html

```
NAME                                READY   STATUS    RESTARTS   AGE
deploy-nginx-nfs-796fb7b8fb-4n2fz   1/1     Running   0          30m
deploy-nginx-nfs-796fb7b8fb-kvvqg   1/1     Running   0          30m
deploy-nginx-nfs-796fb7b8fb-pjv28   1/1     Running   0          30m
```

kubectl Commands

Project :

Create Scenario Horizontal Deployment Autoscaler and Increase load and Pods

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Edit And Run kube-apiserver.yaml

1- vim /etc/kubernetes/manifests/kube-apiserver.yaml

- --enable-bootstrap-token-auth=true
- --enable-aggregator-routing=true

2- kubectl apply -f /etc/kubernetes/manifests/kube-apiserver.yaml

3- kubectl get pod -A | grep api

```
kube-system   kube-apiserver          0/1   Pending   0   17s
kube-system   kube-apiserver-node1    1/1   Running   0   45s
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 1 : install metrics-server on Kubernetes :

```
scp   metrics-server-master      /app
cd /app/metrics-server-master
kubectl  create  -f    deploy/1.8+/
```

```
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.apps/metrics-server created
service/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 1 : install metrics-server on Kubernetes :

kubectl get pods -n kube-system

```
NAME                                READY   STATUS    RESTARTS   AGE
coredns-5644d7b6d9-9fsr5            1/1     Running   17         33d
coredns-5644d7b6d9-ktngh            1/1     Running   17         33d
etcd-node1                          1/1     Running   17         33d
kube-apiserver-node1                1/1     Running   17         33d
kube-controller-manager-node1       1/1     Running   33         33d
kube-flannel-ds-amd64-7thvs         1/1     Running   11         33d
kube-flannel-ds-amd64-rfj78         1/1     Running   19         33d
kube-flannel-ds-amd64-v5hbr         1/1     Running   17         33d
kube-proxy-gg594                    1/1     Running   16         33d
kube-proxy-gmtmt                    1/1     Running   11         33d
kube-proxy-j7s6x                    1/1     Running   17         33d
kube-scheduler-node1                1/1     Running   31         33d
metrics-server-795b774c76-gxzjc     1/1     Running   0          72s
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 2 : Checking

kubectl get apiservices |egrep metrics

v1beta1.metrics.k8s.io kube-system/metrics-server **True** 11h

kubectl get deploy,svc -n kube-system |egrep metrics-server

deployment.extensions/metrics-server 1 1 1 1 11h
service/metrics-server ClusterIP 10.233.31.135 <none> 443/TCP 11h

kubectl top node

ERROR

error: metrics not available yet

OR

kubectl get --raw "/apis/metrics.k8s.io/v1beta1/nodes"

ERROR

{"kind":"NodeMetricsList","apiVersion":"metrics.k8s.io/v1beta1","metadata":{"selfLink":"/apis/metrics.k8s.io/v1beta1/nodes"},"items":[]}

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 3 : Edit metric-server deployment to add the flags For Tshoot

vim deploy/1.8+/metrics-server-deployment.yaml

After Images TAG

```
image: k8s.gcr.io/metrics-server-amd64:v0.3.6

command:
        - /metrics-server
        - --kubelet-insecure-tls
        - --kubelet-preferred-address-types=InternalIP
```

kubectl apply -f metrics-server-deployment.yaml

OutPut :

Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
serviceaccount/metrics-server configured
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
deployment.apps/metrics-server configured

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 3 : Edit metric-server deployment to add the flags For Tshoot

kubectl -n kube-system edit deploy metrics-server

kubectl describe apiservice v1beta1.metrics.k8s.io

```
Status:
  Conditions:
    Last Transition Time:  2019-11-30T05:14:36Z
    Message:               all checks passed
    Reason:                Passed
    Status:                True
    Type:                  Available
Events:                    <none>
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 3 : Edit metric-server deployment to add the flags For Tshoot

kubectl get --raw /apis/metrics.k8s.io/v1beta1/nodes

{"kind":"NodeMetricsList","apiVersion":"metrics.k8s.io/v1beta1","metadata":{"selfLink":"/apis/metrics.k8s.io/v1beta1/nodes"},"items":[{"metadata":{"name":"node1","selfLink":"/apis/metrics.k8s.io/v1beta1/nodes/node1","creationTimestamp":"2019-11-30T05:27:38Z"},"timestamp":"2019-11-30T05:27:23Z","window":"30s","usage":{"cpu":"272355557n","memory":"915696Ki"}},{"metadata":{"name":"node2","selfLink":"/apis/metrics.k8s.io/v1beta1/nodes/node2","creationTimestamp":"2019-11-30T05:27:38Z"},"timestamp":"2019-11-30T05:27:26Z","window":"30s","usage":{"cpu":"90307302n","memory":"468392Ki"}}]}

kubectl top node

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
node1	240m	12%	898Mi	52%
node2	88m	4%	457Mi	26%
node3	81m	4%	223Mi	13%

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 4 : Create Deployment and AutoScale For Load

ON Node2 , Node3 :

docker pull httpd

kubectl run httpd --image=httpd --requests=cpu=100m --limits=cpu=200m --expose --port=80

```
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run
--generator=run-pod/v1 or kubectl create instead.
service/httpd created
deployment.apps/httpd created
```

kubectl get deployments.apps httpd

```
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
httpd   1/1     1            1           8s
```

kubectl autoscale deployment httpd --cpu-percent=50 --min=1 --max=10

```
horizontalpodautoscaler.autoscaling/httpd autoscaled
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 4 : Create Deployment and AutoScale For Load

kubectl run apache-bench -i --tty --rm --image=httpd -- ab -n 500000 -c 1000 http://httpd.default.svc.cluster.local/

OutPut

```
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use
kubectl run --generator=run-pod/v1 or kubectl create instead.
If you don't see a command prompt, try pressing enter.
Completed 50000 requests
Completed 100000 requests
apr_socket_recv: Connection reset by peer (104)
Total of 103709 requests completed
Session ended, resume using 'kubectl attach apache-bench-7876d5dd49-mhmsc -c apache-bench -i -t' command
when the pod is running
deployment.apps "apache-bench" deleted
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 4 : Create Deployment and AutoScale For Load

```
kubectl  get deployments.apps httpd -w
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
httpd   1/1     1            1           2m32s
httpd   1/4     1            1           2m41s
httpd   1/4     1            1           2m41s
httpd   1/4     1            1           2m41s
httpd   1/4     4            1           2m41s
httpd   2/4     4            2           2m47s
httpd   3/4     4            3           2m47s
httpd   4/4     4            4           2m51s
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 4 : Create Deployment and AutoScale For Load

kubectl describe hpa/httpd ==> OK : ScalingActive True and New size: 10

```
Name:                                                     httpd
Namespace:                                                default
Labels:                                                   <none>
Annotations:                                              <none>
CreationTimestamp:                                        Sat, 30 Nov 2019 09:01:22 +0330
Reference:                                                Deployment/httpd
Metrics:                                                  ( current / target )
  resource cpu on pods  (as a percentage of request):  1% (1m) / 50%
Min replicas:                                             1
Max replicas:                                             10
Deployment pods:                                          10 current / 10 desired
Conditions:
  Type          Status  Reason              Message
  ----          ------  ------              -------
  AbleToScale     True    ScaleDownStabilized  recent recommendations were higher than current one, applying the highest recent recommendation
  ScalingActive   True    ValidMetricFound     the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  True    TooManyReplicas      the desired replica count is more than the maximum replica count
Events:
  Type     Reason                   Age                From                       Message
  ----     ------                   ----               ----                       -------
  Warning  FailedGetResourceMetric       6m35s (x3 over 7m5s)   horizontal-pod-autoscaler  unable to get metrics for resource cpu: no metrics returned from resource metrics API
  Warning  FailedComputeMetricsReplicas  6m35s (x3 over 7m5s)   horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: unable to get metrics for resource cpu: no metrics
returned from resource metrics API
  Warning  FailedGetResourceMetric       5m19s (x5 over 6m19s)  horizontal-pod-autoscaler  did not receive metrics for any ready pods
  Warning  FailedComputeMetricsReplicas  5m19s (x5 over 6m19s)  horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: did not receive metrics for any ready pods
  Normal   SuccessfulRescale             5m4s                   horizontal-pod-autoscaler  New size: 3; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             4m3s                   horizontal-pod-autoscaler  New size: 5; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             3m2s                   horizontal-pod-autoscaler  New size: 10; reason: cpu resource utilization (percentage of request) above target
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 4 : Create Deployment and AutoScale For Load

kubectl get horizontalpodautoscalers.autoscaling

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
httpd	Deployment/httpd	195%/50%	1	10	10	4m23s

kubectl get deployments.apps

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
httpd	10/10	10	10	6m33s

kubectl top node

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
node1	206m	10%	944Mi	54%
node2	111m	5%	578Mi	33%
node3	101m	5%	629Mi	36%

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 4 : Create Deployment and AutoScale For Load

kubectl get pod

```
NAME                          READY   STATUS              RESTARTS   AGE
apache-bench-7876d5dd49-mhmsc   1/1     Running             0          103s
httpd-7c485b6488-2njt7          1/1     Running             0          2m12s
httpd-7c485b6488-2sqlp          0/1     ContainerCreating   0          10s
httpd-7c485b6488-84twz          0/1     ContainerCreating   0          10s
httpd-7c485b6488-h5lz4          1/1     Running             0          71s
httpd-7c485b6488-l5w99          1/1     Running             0          2m12s
httpd-7c485b6488-m9dmv          1/1     Running             0          4m33s
httpd-7c485b6488-rnhz9          0/1     ContainerCreating   0          10s
httpd-7c485b6488-w84qd          0/1     ContainerCreating   0          10s
httpd-7c485b6488-wmcwc          0/1     ContainerCreating   0          10s
httpd-7c485b6488-wvh9f          1/1     Running             0          71s
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 5 : Stop Load

kubectl describe hpa/httpd ==> New size: 1

```
Name:                                     httpd
Namespace:                                default
Labels:                                   <none>
Annotations:                              <none>
CreationTimestamp:                        Sat, 30 Nov 2019 09:01:22 +0330
Reference:                                Deployment/httpd
Metrics:                                  ( current / target )
  resource cpu on pods  (as a percentage of request):  1% (1m) / 50%
Min replicas:                             1
Max replicas:                             10
Deployment pods:                          1 current / 1 desired
Conditions:
  Type          Status  Reason            Message
  ----          ------  ------            -------
  AbleToScale     True    ReadyForNewScale    recommended size matches current size
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:
  Type     Reason                        Age              From                       Message
  ----     ------                        ----             ----                       -------
  Warning  FailedGetResourceMetric       42m (x3 over 43m)  horizontal-pod-autoscaler  unable to get metrics for resource cpu: no metrics returned from resource metrics API
  Warning  FailedComputeMetricsReplicas  42m (x3 over 43m)  horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: unable to get metrics for resource cpu: no metrics returned from resource metrics API
  Warning  FailedGetResourceMetric       41m (x5 over 42m)  horizontal-pod-autoscaler  did not receive metrics for any ready pods
  Warning  FailedComputeMetricsReplicas  41m (x5 over 42m)  horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: did not receive metrics for any ready pods
  Normal   SuccessfulRescale             41m              horizontal-pod-autoscaler  New size: 3; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             40m              horizontal-pod-autoscaler  New size: 5; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             39m              horizontal-pod-autoscaler  New size: 10; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             33m              horizontal-pod-autoscaler  New size: 1; reason: All metrics below target
```

kubectl Commands

Create Horizontal Deployment Autoscaler Scenario With Increase load and Pods :

Step 5 : Stop Load

kubectl get horizontalpodautoscalers.autoscaling

```
NAME    REFERENCE        TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
httpd   Deployment/httpd   1%/50%    1         10        1          43m
```

kubectl get pod

```
NAME                     READY   STATUS    RESTARTS   AGE
httpd-7c485b6488-m9dmv   1/1     Running   0          44m
```

Certified Kubernetes Administrator (CKA) Program

- The Certified Kubernetes Administrator (CKA) program was created by the Cloud Native Computing Foundation (CNCF)
- Kubernetes As one of the highest velocity open source projects, Kubernetes use is exploding
- **Cost $300 | Online Exam**
- The online exam consists of a set of performance-based items (problems) to be solved in a command line and candidates have **3 hours** to complete the tasks.
- This includes these general domains and their weights on the exam:

  - ✓ Application Lifecycle Management 8%
  - ✓ Installation, Configuration & Validation 12%
  - ✓ Core Concepts 19%
  - ✓ Networking 11%
  - ✓ Scheduling 5%
  - ✓ Security 12%
  - ✓ Cluster Maintenance 11%
  - ✓ Logging / Monitoring 5%
  - ✓ Storage 7%
  - ✓ Troubleshooting 10%

Exam Resources :

Candidate Handbook
Curriculum Overview
Exam Tips
Frequently Asked Questions
Certification and Confidentiality Agreement
Verify Certification
CKA Reseller FAQs