WHAT IS CONTAINER ORCHESTRATION:---

       Container orchestration automates the deployment, management, scaling, and networkingof  containers. Enterprises that need to deploy and manage hundreds or thousands of Linux containers so in this case we don’t want downtime so container orchestration will give this  facility.

       What is k8s?

       Kubernetes is an open source container orchestration platform that automates many of the manual processes involved in deploying, managing and scaling containerized applications

      Some imp info about k8s---

       •Open-source cluster management tool

•Managed by Cloud Native Computing Foundation (CNCF); originally created by Google

•Under active deployment by a well-supported community.  Kubernetes.io

•Can run on both bare metal and on various cloud providers.

 k8s with support:---

 -----------------------------

 open-shift  ------- Redhat

 Rancher-------- SUSE

 Tanzu  -------- VMWARE

 k8s in cloud:---

 AKS-------- Azure

 EKS----- aws

 GKE---- google

 Internal structure of k8s:----

 KUBERNETES INSTALLATION PROCESS:---

====================================

1> put hostname with ip mapping in /etc/hosts file in all three machine

2> disable swap & firewalld

  in production without stop firewalld add port no--- search in google      "kubernetes components ports"

3> we have already configured kubernetes.repo & crio.repo under /etc/yum.repos.d

     note:--- for new machine:----

              search---- cri-o in google---

                         kubeadm  install cluster  ----- in google------   open first page--- select "installing kubeadm"

                         ---- here we will get everything.

4> install & start cri-o (in all machines)

--------------------------------

 #yum install cri-o\* -y

 #systemctl enable --now crio

5> install kubectl kubeadm & kubelet (in all machines)

 #yum install kubectl\* kubeadm\*  kubelet\* -y

6> start kubelet service:--- (in all machines)

===============================

# systemctl enable kubelet

#systemctl start kubelet

note:--- for copy -------select with mouse

         for paste:----  press -- shift+insert

7>Run the following command in master node:--

==============================================

#kubeadm init --pod-network-cidr=10.244.0.0/16  ----------- to define the ip range for pod

                                                     if  calico will be installed --it will give ip to pod

                                                     from this range.

If face any issue:--- to solve that:---

#kubeadm reset

#systemctl restart kubelet

#systemctl enable kubelet

#copy three commands from the output & paste it. (only on master node)

#now copy join commands from the output of kubeadm init command:---  (paste this command in worker1 & worker2 only)

8> install calico (Only  on master node):---

========================================

to get that link:---  open google--- "calico network kubernetes install"---- open 1st link

#wget <http://raw.githubusercontent.com/projectcalico/calico/v3.24.0/manifests/calico.yaml>

now open calico.yaml & replce docker.io to quay.io in the entire file

#vim ~/calico.yml

replace docker.io ---- to--- quay.io

:%s/docker.io/quay.io/g

\*\*\*\*note:---  from nov-2020--we can download 200 images per public ip in 6 hours from Docker hub---- limitation

Now install calico---

#kubectl apply -f calico.yaml

#kubectl get nodes  ------------------ to check the nodes in the cluster

#kubectl get pods -A ------- to know all the pods including kubernets clusters pods

 bash completion:----

 # kubectl completion bash > ~/.kube/kube.sh

 # source ~/.kube/kube.sh  ------------------>>> for temp change

 # vim ~/.bashrc   ------------------------->>> for permanent change

 source $HOME/.kube/kube.sh

indentaion help:---

---------------------------

#vim ~/.vimrc

set ai

set ts=2

set et

set cursorcolumn

image  ---------

quay.io/gauravkumar9130/mywebapp

-----------------------------------------/production:v1

                                             /production:v2

                                             /production:v3-v5

                                             /mysql

                                             /hotel

                                             /tea

                                             /coffee

                                             /nginxdemo

 Basic commands in k8s:----

 ========================

 # kubectl get nodes

# kubectl get nodes -o wide

# kubectl describe nodes master  | less

]# kubectl  get pods

How to create a pod:---

---------------------------------

There are two way we can create a pod:--

a> command line:---

# kubectl  run pod1 --image quay.io/gauravkumar9130/mywebapp

# kubectl describe pod pod1 |less

# kubectl get pods -o wide

How to access  a pod:---

----------

# kubectl exec -it pod1 /bin/sh      --------->> for unix based image

# kubectl exec -it pod1 /bin/bash      --------->> for linux based image

# kubectl delete pod pod1 ---------->> to delete the pod

how to create a pod via yml file:---

===========================

# kubectl explain pod | less

apiVersion: v1

kind: Pod

metadata:

 name: monday

spec:

 containers:

 - name: test

   image: quay.io/gauravkumar9130/mywebapp

# kubectl  create -f mypod.yaml

# kubectl delete -f mypod.yaml

For multi-container pod:-----

apiVersion: v1

kind: Pod

metadata:

 name: multi-pod

spec:

 containers:

 - name: cont1

   image: nginx

 - name: cont2

   image: redis

 - name: cont3

   image: memcached

~

to access cont:---

#kubectl exec -it  multi-pod    /bin/sh    ------->> cont1/default cont

#kubectl exec -it  multi-pod  -c cont2   /bin/sh    ------->> cont2 cont

#kubectl exec -it  multi-pod  -c cont3   /bin/sh    ------->> cont3 cont

ImagePullPolicy:----

-------------------------------

How image will be pulled  to create pod.

a> Always  ----- default policy

b> IfNotPresent:----   best policy

c> Never:---

labs:---

in yml  file:---

-----------------

apiVersion: v1

kind: Pod

metadata:

 name: image-pod

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

or

# kubectl  run pod1000 --image quay.io/gauravkumar9130/mywebapp --image-pull-policy IfNotPresent

Label & selector:---

================

label-----  It helps to put a tag on any resource in k8s by which we can identify that resource very easily in k8s.

       label------->>  key:value

       pod1 ------->  sports=cricket   game=football    flower=lotus   fruit=banana

                                 key       value

                           we can't put different values for the same key for a same resource.

                           pod1  -----------   game=football   game=cricket

labs:--- (for existing pod)

=============

]# kubectl  get pods --show-labels

# kubectl label pod image-pod fruit=apple

# kubectl label pod image-pod flower=lily game=hockey sports=basketball

# kubectl label pod image-pod sports-         ------>>> to remove or withdraw label

# kubectl label pod image-pod --overwrite flower=lotus

for new pod:--

-----------------

apiVersion: v1

kind: Pod

metadata:

 name: label-pod

 labels:

  fruit: apple

  flower: lily

  sports: football

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

# kubectl  run pod10000 --image quay.io/gauravkumar9130/mywebapp --image-pull-policy IfNotPresent --labels fruit=pineapple

selector:-----

==========

It helps us to findout specific  types of resources on the basis of labels.

types:---

a> Equity/Equality based seletor:---

      --------->>>  works on single key with single value

      ex:----

       kubectl get pods --selector flower=lily

       kubectl get pods --selector flower=lily --show-labels

       for opposite search:--

       kubectl get pods --selector flower!=lily

       kubectl get pods --selector flower!=lily --show-labels

b> Set based selector:----

--------->>>  works on single key with  multiple values

types of operator:--

a> in --------- equal to

b> notin ------- not equal to

ex:----

kubectl get pods --selector 'flower in(lily,lotus)'

kubectl get pods --selector 'flower in(lily,lotus)' --show-labels

# kubectl get pods --selector 'flower notin(lily,lotus,marigold)' --show-labels    --------->>>> notin operator

Replica:----

==========

------>>>> Group of pods

------>>>>  it will always maintain the desired state

------>>>> types

a> ReplicationController/rc-------- old concept  -----  >>> it can work only on equity based selector (one key with one value)

b> ReplicaSet/ rs  ------------- new concept ---------->>>  it can work both on equity as well as set based selector (one key with one value as well as one key with multiple values)

labs:---

ReplicaSet with equity based selector:--

# kubectl get replicaset

# kubectl get rs

apiVersion: apps/v1

kind: ReplicaSet

metadata:

 name: myreplica

spec:

 replicas: 6

 selector:

  matchLabels:      #it is used only for equity based selector

   flower: lily

 template:          #it is for the new pod

  metadata:

   labels:

    flower: lily

  spec:

   containers:

   - name: xyz

     image: quay.io/gauravkumar9130/mywebapp

     imagePullPolicy: IfNotPresent

# kubectl get rs  -o wide

# kubectl  describe rs myreplica |less

scaling:---

--------autoscaling:---

------- manual scaling:----

---- manual scaling:---

# kubectl scale rs myreplica --replicas 15

labs:--

# kubectl get nodes

# kubectl cordon worker2

# kubectl drain worker2 --ignore-daemonsets --delete-emptydir-data --force

   now do all your schedule maintenence task in worker2

 # kubectl uncordon worker2      ------>>> to make this worker node once again online

Set based replica:---

apiVersion: apps/v1

kind: ReplicaSet

metadata:

 name: set-replica

spec:

 replicas: 6

 selector:

  matchExpressions:        #it will be only for set based replica

  - key: fruit

    operator: In

    values:

    - apple

    - guava

    - banana

 template:

  metadata:

   labels:

    fruit: banana

  spec:

   containers:

   - name: xyz

     image: quay.io/gauravkumar9130/mywebapp

     imagePullPolicy: IfNotPresent

DaemonSet:---/ds

----------------------------

------>>> it will omne copy of pod on each worker node

------>>>  it will ensure that pod will always run under every worker node

----->>> if any further worker node added or remove from the cluster ---- automaticlly ds also change .

------>>> use case --- we can use it for logging & monitoring purpose.

labs:---

--------

# kubectl get ds

# kubectl get daemonset

# kubectl explain ds |less

apiVersion: apps/v1

kind: DaemonSet

metadata:

 name: myds

spec:

 selector:

  matchLabels:

   sports: football

 template:

  metadata:

   labels:

    sports: football

  spec:

   containers:

   - name: xyz

     image: quay.io/gauravkumar9130/mywebapp

     imagePullPolicy: IfNotPresent

     Manual scheduling:----

     =================

     ------->>> in this process , we will take the decission in which  worker  node we want to create  the pod.

lab-1 (manual selection of worker node)

====================

apiVersion: v1

kind: Pod

metadata:

 name: manual-pod

spec:

 containers:

 - name: xyz

   image: quay.io/gaurvkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

 nodeName: worker1

lab-2 (NodeSelector)  ------ create pod on the basis of worker node labels.

# kubectl get nodes --show-labels

# kubectl  label nodes worker1 disk=ssd

# kubectl get nodes --show-labels | grep -i disk=ssd

# kubectl  label nodes worker1 disk-

apiVersion: v1

kind: Pod

metadata:

 name: test-pod

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

 nodeSelector:

  disk: ssd

~

lab-3

taints & tolerations:---

-------------

taints ----- node condition

toleration ------ pod condition

if taints=toleration ------> then only pod will be created in the scheduled node

taints----

key=value:effect

ex:---

fruit=apple:NoSchedule

fruit=apple:NoExecute

effects:---

NoSchedule------    All existing pods will continue their operations

NoExecute---------  All existing pods will terminate & if any pods are under replica/deployments/ statefulset --- they will be recreated in other worker node

lab:--

taints:---

# kubectl describe nodes worker1 |less

#kubectl taints node worker1 freuit=apple:NoSchedule

# kubectl describe nodes worker1 |grep -i taints

# kubectl taint node worker1 fruit-             ------->> to remove taints value

Create pod with toleration value:---

apiVersion: v1

kind: Pod

metadata:

 name: toleration-pod

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

 tolerations:

 - key: "fruit"

   operator: "Equal"

   value: "apple"

   effect: "NoExecute"

Environment Variable:---

====================

------>> it a concept, by which we can inject variable & values into the new pod during its creation.

------->>>  types:---

              a> PlainKey-------->>> variables & values will be in the same yml file by which we will create the pod. all the varibales & values  are in text format

              b> configMap

              c> Secret

MYSQL\_USER     ------>  to create user a/c

MYSQL\_PASSWORD  ---------> to create user password

MYSQL\_ROOT\_PASSWORD----------> to set user root password

MYSQL\_DATABASE     ----------> to create a new database

lab:-----

plainKey:---

apiVersion: v1

kind: Pod

metadata:

 name: my-plain-data-pod

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mysql

   imagePullPolicy: IfNotPresent

   env:

   - name: MYSQL\_USER

     value: sree

   - name: MYSQL\_PASSWORD

     value: redhat@123

   - name: MYSQL\_ROOT\_PASSWORD

     value: redhat

   - name: MYSQL\_DATABASE

     value: india

     To check the variables & values:---

     ------------------

     # kubectl  exec -it my-plain-data-pod /bin/bash

      env

      # mysql -u sree -p

       show databases;

       configMap/cm:----- (dictionary will be in the text format)

       -----------------

       labs:---

       #kubectl get cm

       # kubectl create cm training-cm --from-literal=MYSQL\_USER=pallavi --from-literal=MYSQL\_PASSWORD=redhat@123 --from-literal=MYSQL\_ROOT\_PASSWORD=redhat --from-literal=MYSQL\_DATABASE=world-cup

       # kubectl describe cm training-cm | less

       apiVersion: v1

kind: Pod

metadata:

 name: training-cm-pod

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mysql

   imagePullPolicy: IfNotPresent

   envFrom:

   - configMapRef:

      name: training-cm

       secret:---  (dictionary will be in the base64 encrypted format)

       --------------------------

       #kubect get secrets

       # kubectl create secret generic training-sec --from-literal=MYSQL\_USER=pallavi --from-literal=MYSQL\_PASSWORD=redhat@123 --from-literal=MYSQL\_ROOT\_PASSWORD=redhat --from-literal=MYSQL\_DATABASE=world-cup

       # kubectl  describe secrets training-sec | less

apiVersion: v1

kind: Pod

metadata:

 name: training-secret-pod

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mysql

   imagePullPolicy: IfNotPresent

   envFrom:

   - secretRef:

      name: training-sec

      To check :---

      =========

      #kubectl exec -it training-secret-pod /bin/bash

      #env     ------>> you will be able to see all the variables

      #mysql -u  pallavi -p     -------> try to login with this user a/c

      #show databases;

To know the values of the variables:---

--------------

# kubectl edit secrets training-sec

# echo cmVkaGF0QDEyMw== | base64 -d

       a> service:-----

            ---------------

            It will help us to create a single entry point against our apps which is running inside the k8s cluster. by this entry point people with in & also outside the cluster can access

            our apps very easily in k8s cluster.

            Types:---

            a> ClusterIP ----- default service type. it is used for internal access of our apps .

            b> NodePort  ------- it is used for external access of our apps .

                                               port range ----   30000----32767

                                              to access the application:---    <master/worker/ip\_of\_master/ip\_of\_worker>:<nodeport\_no>

            c> LoadBalancer-----  it is used for external access of our apps .

            Labs:--- (clusterIP)

            ----------------

            step-1 (create pod replica)

            # kubectl run pod1 --image quay.io/gauravkumar9130/mywebapp --image-pull-policy IfNotPresent --labels fruit=apple

            apiVersion: apps/v1

kind: ReplicaSet

metadata:

 name: myreplica

spec:

 replicas: 6

 selector:

  matchLabels:      #it is used only for equity based selector

   flower: lily

 template:          #it is for the new pod

  metadata:

   labels:

    flower: lily

  spec:

   containers:

   - name: xyz

     image: quay.io/gauravkumar9130/mywebapp

     imagePullPolicy: IfNotPresent

step-2 (configure service)

---------------------

#kubect get svc

#kubectl explain svc | less

apiVersion: v1

kind: Service

metadata:

 name: my-cluster-svc

spec:

 type: ClusterIP

 ports:

 - targetPort: 80   #apps port no of container/pod

   port: 80         #bind port no with clusterip

 selector:

  flower: lily

  # kubectl  get svc  -o wide

  alternate method:---

  # kubectl expose rs myreplica --target-port 80 --port 80 --name my-rs-cluster-svc --type ClusterIP

# kubectl expose pod pod1 --target-port 80 --port 80 --name pod-clu-svc --type ClusterIP      ------------> for single pod

labs--- (NodePort)

-------------------------

lab1--- (create pod/replica)

lab2--- (configure NodePort)

apiVersion: v1

kind: Service

metadata:

 name: my-node-svc

spec:

 type: NodePort

 ports:

 - targetPort: 80

   port: 80

   nodePort: 30000

 selector:

  flower: lily

or

#

LoadBalancer (provide public ip against every expose micro apps)

---------

labs:--- (on prem k8s)

---------------

step-1 (installation of loadbalancer)------>>> metallb

------------------------

# kubectl get ns

# wget <https://raw.githubusercontent.com/metallb/metallb/v0.13.7/config/manifests/metallb-native.yaml>        ------->> download that yaml file for metallb

# kubectl apply -f metallb-native.yaml      ------->>> run that yaml file

# kubectl  get ns     ------->>> we will be able to one new ns ------ metallb-system

# kubectl  get pods -n metallb-system    ----->>> to check the pods that run in that ns

step-2  (specify ip pool for loadbalancer)

----------

# kubectl  explain ipaddresspool | less

apiVersion: metallb.io/v1beta1

kind: IPAddressPool

metadata:

 name: frist-pool

 namespace: metallb-system

spec:

 addresses:

 - 172.25.230.10-172.25.230.30

# kubectl  get ipaddresspool -n metallb-system

step-3 (create pod/replica)    ------>> common step for onprem & cloud k8s cluster

--------------------

step-4 (expose apps against loadbalancer)

------------------------------------

apiVersion: v1

kind: Service

metadata:

 name: my-load-svc

spec:

 type: LoadBalancer

 ports:

 - targetPort: 80

   port: 80

 selector:

  flower: lily

~

# kubectl expose rs myreplica --target-port 80 --port 80 --name my-load --type LoadBalancer   ------>> for replica

# kubectl expose pod pod1 --target-port 80 --port 80 --name my-pod-load --type LoadBalancer     ------>> for pod

       b> deployments:---

       •A Deployment provides declarative updates for pods andreplicasets.

•You describe a desired state in a Deployment, and the  Deployment Controller changes the actual state to the desired state at a  controlled rate. You can define Deployments to create new ReplicaSets, or to  remove existing Deployments and adopt all their resources with new  Deployments.

**Two types of built in strategies ->**

**1) RollingUpdate:---- default deployment startetgy**

default if not defined

new ReplicaSet is created, then scaled up as the old ReplicaSet is scaled down

**2) Recreate**

removes all existing pods in the existing ReplicaSet first

then creates new pods in the new ReplicaSet

labs-1 (for rolling update)

------------------

step-1 (create deployment)

# kubectl  get deployment

# kubectl explain deployment |less

apiVersion: apps/v1

kind: Deployment

metadata:

 name: mydeploy

spec:

 replicas: 6

 selector:

  matchLabels:

   flower: marigold

 template:

  metadata:

   labels:

    flower: marigold

  spec:

   containers:

   - name: training

     image: quay.io/gauravkumar9130/production:v1

     imagePullPolicy: IfNotPresent

~

# kubectl  get deployments.apps -o wide

 step-2

 (expose  apps against service)

 # kubectl expose deployment mydeploy --target-port 80 --port 80 --name production-svc --type LoadBalancer

step-3 (update version)

# kubectl set image deployment mydeploy training=quay.io/gauravkumar9130/production:v2 --record

# kubectl set image deployment mydeploy training=quay.io/gauravkumar9130/production:v3 --record

# kubectl set image deployment mydeploy training=quay.io/gauravkumar9130/production:v4 --record

# kubectl set image deployment mydeploy training=quay.io/gauravkumar9130/production:v5 --record

step-4 (for collecting hist)

# kubectl  rollout history deployment mydeploy

step-5 (change version)/ rollback

--------------------------

# kubectl  rollout undo deployment mydeploy

# kubectl  rollout undo deployment mydeploy --to-revision 2

step-6:-- (scaling)

------------------

# kubectl scale deployment mydeploy --replicas 10

    Deployments for Recreate:---

    -----------------------

    apiVersion: apps/v1

kind: Deployment

metadata:

 name: mydeploy-recreate

spec:

 strategy:

  type: Recreate

 replicas: 6

 selector:

  matchLabels:

   flower: marigold1

 template:

  metadata:

   labels:

    flower: marigold1

  spec:

   containers:

   - name: training

     image: quay.io/gauravkumar9130/production:v1

     imagePullPolicy: IfNotPresent

2nd process of deployment:---( we will devide customer traffic b/w diff version deployment)

======================

step-1 (create old apps)

apiVersion: apps/v1

kind: Deployment

metadata:

 name: olddeploy

spec:

 replicas: 6

 selector:

  matchLabels:

   flower: marigold

 template:

  metadata:

   labels:

    flower: marigold

  spec:

   containers:

   - name: training

     image: quay.io/gauravkumar9130/production:v1

     imagePullPolicy: IfNotPresent

step- 2 (expose deployment against service)

-------------

# kubectl expose deployment olddeploy --target-port 80 --port 80 --name combo-svc --type LoadBalancer

step-3 (create new deployment with same label)

* -----------------------

apiVersion: apps/v1

kind: Deployment

metadata:

 name: newdeploy

spec:

 replicas: 4

 selector:

  matchLabels:

   flower: marigold

 template:

  metadata:

   labels:

    flower: marigold

  spec:

   containers:

   - name: school

     image: quay.io/gauravkumar9130/production:v3

     imagePullPolicy: IfNotPresent

type:--3    (Blue Green deployment)

-------------------

Blue--- old apps

Green- new apps

labs:---

step1 (for old/blue deployment)

-------------------

apiVersion: apps/v1

kind: Deployment

metadata:

 name: blue-deploy

spec:

 replicas: 6

 selector:

  matchLabels:

   soft: blue

 template:

  metadata:

   labels:

    soft: blue

  spec:

   containers:

   - name: training

     image: quay.io/gauravkumar9130/production:v1

     imagePullPolicy: IfNotPresent

~

step-2 (configure service)

---------

# kubectl expose deployment blue-deploy --target-port 80 --port 80 --name blue-green-svc --type LoadBalancer

step-3 (create green/new deploy & test it)

------------

apiVersion: apps/v1

kind: Deployment

metadata:

 name: green-deploy

spec:

 replicas: 6

 selector:

  matchLabels:

   soft: green

 template:

  metadata:

   labels:

    soft: green

  spec:

   containers:

   - name: training

     image: quay.io/gauravkumar9130/production:v3

     imagePullPolicy: IfNotPresent

~

step-4 (check apps of new deployment --- if it is perfect then switch your service from blue to green deployment)

# kubectl  get svc -o wide

# kubectl edit svc blue-green-svc

       c> namespace:-----

       ----------------------------

       it will help us to isolate resources from one ns another ns.

       these are the virtual clusters on the top physical k8s cluster.

       lab-1 :---

       --------

       How to create ns:---

       --------------

       # kubectl get pods

       # kubectl get pods -n india

       # kubectl get pods -A |less

       # kubectl explain ns |less

 apiVersion: v1

kind: Namespace

metadata:

 name: nepal

 # kubectl create ns bhutan

 To create a pod in bhutan ns:--

apiVersion: v1

kind: Pod

metadata:

 name: thimpu

 namespace: bhutan

 lables:

  fruit: mango

  flower: lily

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

~

# kubectl exec -it -n bhutan thimpu /bin/sh      ------->>> access the pod from bhutan ns

# kubectl run kathmandu --image quay.io/gauravkumar9130/mywebapp --image-pull-policy IfNotPresent -n nepal --labels fruit=apple

How to change default ns:----

=======================

# kubectl  config set-context $(kubectl config current-context) --namespace=nepal

# kubectl config get-contexts      -------->>>  to check the default ns name

ResourceQuota in ns:-----

---------------------

By implementing  RQ--- we can limit the no of resource (pods,service, deployments,replica) that can be created as  well as compute resource (memory, cpu) can be consumed

by that ns

    for compute/ h/w resource limit:----

    -----------------

    requests-----  minimum

      requests.cpu -----  10m

    limits-------   maximum

    labs:---

    # kubectl  get resourcequotas

# kubectl  get resourcequotas -n nepal

# kubectl  explain resourcequotas  | less

apiVersion: v1

kind: ResourceQuota

metadata:

 name: govt-quota

 namespace: nepal

spec:

 hard:

  pods: "5"

  services: "4"

  requests.cpu: "100m"

  requests.memory: "700Mi"

  limits.cpu: "150m"

  limits.memory: "1200Mi"

How to create a pod with resource limit:---

---------------------------------

apiVersion: v1

kind: Pod

metadata:

 name: morning

 namespace: nepal

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumr9130/mywebapp

   imagePullPolicy: IfNotPresent

   resources:

    requests:

     cpu: "10m"

     memory: "100Mi"

    limits:

     cpu: "12m"

     memory: "150Mi"

# kubectl edit resourcequotas -n nepal govt-quota       ------->>> for further editing of the quota

       d> storage:----

       ----------------------

       Types of storage:---

       a> Ephemeral storage (temp storage):---

          ex:---  emptyDir:---

                    ----------------

                    labs:---

apiVersion: v1

kind: Pod

metadata:

 name: emptypod

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

   volumeMounts:

   - name: india     # location will be /var/lib/kubelet/pods in worker node

     mountPath: /mumbai

 volumes:

 - name: india

   emptyDir: {}

       b> Persistent storage (permanent storage)

       ex:--HostPath:--- (data as well as storage will available after deleting the pod also)

             HostPath--- storage will be craeted on that worker node where pod will created

       pv ---- (persistent volume)---- storage

       pvc ----- (persistent volume claim)

       pod

       labs---

       step-1 (create pv)

       ---------------------------

        # kubectl  get pv

        # kubectl  explain pv |less

apiVersion: v1

kind: PersistentVolume

metadata:

 name: pv1

spec:

 accessModes:

 - ReadWriteMany

 storageClassName: csi-hostpath-sc

 capacity:

  storage: "2Gi"

 hostPath:

  path: /mumbai

step-2 (for pvc)

---------------------

# kubectl explain pvc |less

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

 name: pvc1

spec:

 accessModes:

 - ReadWriteMany

 storageClassName: csi-hostpath-sc

 resources:

  requests:

   storage: "2Gi"

step-3

create pod:--

apiVersion: v1

kind: Pod

metadata:

 name: pv-pvc-pod

spec:

 conatiners:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

   volumeMounts:

   - name: country

     mountPath: /india

 volumes:

 - name: country

   persistentVolumeClaim:

    claimName: pvc1

~

     Central persistent storage:---

     ====================

       (nfs-storage)

       step-1 (for nfs)

       ------------

       # yum install nfs-utils\* -y

       ]# mkdir /thursday

        # chmod 1777 /thursday/

        # vim /etc/exports     --------->>> this file is responsible for nfs share

    /thursday       \*(rw,sync)

      # systemctl enable nfs-server.service

      # systemctl start nfs-server.service

      # showmount -e 172.25.231.100    ----->> check nfs share with nfs server ip

      step-2

      (create pv)

      --------

      apiVersion: v1

kind: PersistentVolume

metadata:

 name: nfs-pv

spec:

 accessModes:

 - ReadWriteMany

 storageClassName: nfs-storage

 capacity:

  storage: "3Gi"

 nfs:

  server: 172.25.231.100

  path: /thursday

step-3

create pvc:--

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

 name: nfs-pvc

spec:

 accessModes:

 - ReadWriteMany

 storageClassName: nfs-storage

 resources:

  requests:

   storage: "3Gi"

step-4

---------

create pod:---

-----------------

apiVersion: v1

kind: Pod

metadata:

 name: pod1

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullPolicy: IfNotPresent

   volumeMounts:

   - name: country

     mountPath: /bharat

 volumes:

 - name: country

   persistentVolumeClaim:

    claimName: nfs-pvc

       e> security:------

       -----------------------

         1> Types of a/c & their uses:------

              There are two types a/c can be created in k8s

              a> service a/c:---  it responsible for resource authentication form api server.

                                        ------>> by default one sa will be created in every ns known as default sa ---it is responsible for resource authentication of that ns  from  api server

                                        ------>> if it is required, we can create new sa & attech new resource with this new sa.

             labs:---

             # kubectl  get sa

            # kubectl describe pods -n india pod1 |less

            # kubectl  explain sa | less

            apiVersion: v1

kind: ServiceAccount

metadata:

 name: kolkata

 namespace: india

How to attach a sa with a new resource:---

--------------------

apiVersion: v1

kind: Pod

metadata:

 name: bengal

 namespace: india

 labels:

  flower: lily

spec:

 containers:

 - name: xyz

   image: quay.io/gauravkumar9130/mywebapp

   imagePullpolicy: IfNotPresent

 serviceAccountName: kolkata

apiVersion: v1

kind: Namespace

metadata:

 name: college1

---

apiVersion: v1

kind: ServiceAccount

metadata:

 name: teacher

 namespace: college1

---

apiVersion: apps/v1

kind: Deployment

metadata:

 name: mydeploy1

 namespace: college1

spec:

 replicas: 5

 selector:

  matchLabels:

   flower: marigold

 template:

  metadata:

   labels:

    flower: marigold

  spec:

   containers:

   - name: xyz

     image: quay.io/gauravkumar9130/mywebapp

     imagePullPolicy: IfNotPresent

   serviceAccountName: teacher

              b> user a/c:---

              labs :---   to create u/a:--

              # git clone <https://github.com/gauravkumar9130/kube-user>

             ]# cd kube-user/

              # chmod +x user\_script.sh

              # mkdir ~/pallvi

              # cp user\_script.sh ~/pallvi/

               # cd ~/pallvi/

               ./user\_script.sh

         2> RBAC (role based access control):------

         To assign privilege to u/a & s/a ---- so that they can do activity in k8s cluster.

         Area------

         a> namespace wide:---

             role:--

                 verb:---  list of task

                 resource:--  pods/services/deployments

             rolebinding:---

             labs:----

             --------

             # kubectl  get role -n lucknow

             # kubectl create role role1 --verb create,list,get  --resource pods,services,deployments -n lucknow

             # kubectl describe role -n lucknow role1

             rolebinding:--

             # kubectl get rolebinding -n lucknow

             # kubectl  create rolebinding rolebind1 --role role1  -n lucknow --user pallavi

             # kubectl  create rolebinding rolebind1 --role role1  -n lucknow --serviceaccount lucknow:default

             # kubectl  describe rolebindings.rbac.authorization.k8s.io -n lucknow rolebind1

         b> cluster wide:---

              clusterrole:---

               verb:---  list of task

                 resource:--  pods/services/deployments

              clusterrolebinding:---

              labs:---

              # kubectl get clusterrole|less

              # kubectl create clusterrolebinding clusterbind1 --clusterrole cluster-admin --user pallavi

              # kubectl create clusterrolebinding clusterbind1 --clusterrole cluster-admin --serviceaccount kolkata:defualt

              To create customized clusterrole:---

              # kubectl create clusterrole clusterrole1 --verb create,delete,get,list --resource namespaces,pods,services

              # kubectl create clusterrolebinding clusterrolebind1 --clusterrole clusterrole1 --user pallavi

# kubectl auth can-i create ns --as pallavi

# kubectl auth can-i create ns

       f> monitoring:------

       -------------------------

       basic commands:---

       #kubectl logs <pod\_name>   ------ to check the log of a pod

       #kubectl logs <pod\_name>  -c  <cont\_name> ------ to check the log of a container

       grafana  with prometheus:------

       ----------------------------------

       prometheus----- log collection tool

       grafana ----- monitoring dash board tool

       $ git clone <https://github.com/gauravkumar9130/grafana>

       $ cd grafana/

       $ kubectl apply -f 1-prometheus/.

       $ kubectl apply -f 2-grafana/.

       $ kubectl  get ns     ------->>   monitoring ns will be created

       $ kubectl get pods -n monitoring

       $ kubectl get svc -n monitoring

       user------>  admin

       pass:----->   admin

       dashboard----> import----> 6417----> import---->   prometheus-----> prometheus----> import

       kubernetes dashboard:----- (monitoring tool)

       ---------------------------------------

       $ wget <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.3/aio/deploy/recommended.yaml>

       $ kubectl  create -f recommended.yaml

       $ kubectl  get ns    ------->>> kubernetes-dashboard   ns will be created

       ]$ kubectl get pods -n kubernetes-dashboard

       $ kubectl get svc -n kubernetes-dashboard

       to login in kubernetes-dashboard  ----- we need token against sa:---

       ----------

       $ kubectl get sa  -n kubernetes-dashboard

       $ kubectl describe pods -n kubernetes-dashboard kubernetes-dashboard-5bd89d988-jqdnp|less

       $ kubectl create clusterrolebinding clusterrole2  --clusterrole cluster-admin --serviceaccount kubernetes-dashboard:kubernetes-dashboard

       $ kubectl create token -n kubernetes-dashboard kubernetes-dashboard    ------>>> to generate token

       g> ingress & network policy:------------------

       Ingress:-----

       labs:---

       step-1

       install ingress:--

       $ git clone <https://github.com/kubernetes/ingress-nginx>

       $cd ingress-nginx/deploy/static/provider/cloud

       $ kubectl apply -f deploy.yaml

       $ kubectl get ns   ------>>> ingress-nginx ns will be created

       $ kubectl get svc -n ingress-nginx

       step-2 (create apps)

       -----------------

       $ kubectl create deployment hotel --image=quay.io/gauravkumar9130/hotel --replicas 5

       $ kubectl create deployment tea --image=quay.io/gauravkumar9130/tea --replicas 5

       $ kubectl create deployment  coffee --image=quay.io/gauravkumar9130/coffee --replicas 5

       step-3  (expose apps with clusterip)

       -----------

       $ kubectl expose deployment hotel --target-port 80 --port 80 --name hotel-svc --type ClusterIP

       $ kubectl expose deployment tea --target-port 80 --port 80 --name tea-svc --type ClusterIP

       $ kubectl expose deployment coffee --target-port 80 --port 80 --name coffee-svc --type ClusterIP

       step-4 (create routing rule in ingress)

       -----------

       $ kubectl explain ingress | less

       apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

 name: hotel-ingress

 annotations:

  kubernetes.io/ingress.class: nginx

spec:

 rules:

 - http:

    paths:

    - path: /   #we are declering main apps

      pathType: Prefix

      backend:

       service:

        name: hotel-svc

        port:

         number: 80

    - path: /tea

      pathType: Prefix

      backend:

       service:

        name: tea-svc

        port:

         number: 80

    - path: /coffee

      pathType: Prefix

      backend:

       service:

        name: coffee-svc

        port:

         number: 80

       $ kubectl  get ingress

       $ kubectl describe ingress hotel-ingress

       network policy:------------------

       -------------------

       firewall rule ----

       INGRESS  ------ Incoming packets

       EGRESS -------- outgoing packets

       589  kubectl run pod-1 --image quay.io/gauravkumar9130/mywebapp --labels fruit=apple

  590  kubectl run pod-2 --image quay.io/gauravkumar9130/mywebapp --labels fruit=banana

  591  kubectl run pod-3 --image quay.io/gauravkumar9130/mywebapp --labels flower=lily

  592  kubectl run pod-4 --image quay.io/gauravkumar9130/mywebapp --labels flower=lotus

  593  kubectl run pod-5 --image quay.io/gauravkumar9130/mywebapp --labels sports=football

  $ kubectl  get networkpolicy

  $ kubectl  explain  networkpolicy |less

  apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

 name: my-policy

spec:

 podSelector:

  matchLabels:

   fruit: apple

 policyTypes:

 - Ingress

 - Egress

 ingress:

 - from:

   - podSelector:

      matchLabels:

       sports: football

 egress:

 - to:

   - podSelector:

      matchLabels:

       flower: lily

       h> cluster upgradation + etcd backup restore

       i> init & side car contauner :-----

       Init-container & Sidecar-container:---

Kubernetes init containers:---

Init Container

==============================

#vim init.yml

apiVersion: v1

kind: Pod

metadata:

 name: web-pod

spec:

 volumes:

 - name: myvol

   emptyDir: {}

 containers:

 - name: maincontainer

   image: quay.io/gauravkumar9130/nginx

   volumeMounts:

   - name: myvol

     mountPath: /usr/share/nginx/html

 initContainers:

 - name: init1

   image: quay.io/gauravkumar9130/busybox

   command: ["wget","-O","/webfolder/index.html","<https://download.docker.com/linux/centos/docker-ce.repo>"]

   volumeMounts:

   - name: myvol

     mountPath: /webfolder

Kubernetes sidecar containers:----

apiVersion: v1

kind: Pod

metadata:

  name: webserver

spec:

  volumes:

    - name: website-vol

      emptyDir: {}

  containers:

    - name: nginx

      image: nginx

      volumeMounts:

        - name: website-vol

          mountPath: /usr/share/nginx/html

    - name: sidecar-container

      image: quay.io/gauravkumar9130/ubuntu-git

      command: ["sh","-c","while true; do git clone <https://github.com/gauravkumar9130/webpage>; sleep 10;cd webpage;cp \* /downloadweb/;done"]

      volumeMounts:

        - name: website-vol

          mountPath: /downloadweb

 kubectl exec -it multi-pod -c memcached /bin/sh

       j> cron job:-----

       job scheduling:---

       time stamp:---

       ----------------

       minutes                hours                     <day\_of\_the\_month>              <month>                <day\_of\_the\_week>           task

        1                               2                                       3                                         4                                   5

        (0-59)                    (0-23)                              (1-31)                                     (1-12)                        (0-7)        0/7------ sunday, 1---monday, 2-tuesday ------- 6--staturday

         25th nov friday at 1:22pm

         22                            13                                   25                                       11                                   5                               task

         evrey 25th nov at 1:22pm

         22                            13                                   25                                       11                                   \*                              task

          evrey day of nov  at 1:22pm

         22                            13                                   \*                                      11                                   \*                              task

          evrey day at 1:22pm

         22                            13                                   \*                                     \*                                   \*                              task

         evrey  2 hours

         \*                           \*/2                                   \*                                     \*                                   \*                              task

         evrey minute

          \*                           \*                                  \*                                     \*                                   \*                              task

            \*/1                           \*                                   \*                                     \*                                   \*                              task

  every 1st jan at 6:00pm

           0                                 18                                1                                   1                                      \*

  $ kubectl  explain  cronjob |less

 apiVersion: batch/v1

kind: CronJob

metadata:

 name: my-cronjob

spec:

 schedule: "\* \* \* \* \*"       # for every one minute

 jobTemplate:

  spec:

   template:

    spec:

     containers:

     - name: my-cronjob

       image: busybox:1.28

       imagePullPolicy: IfNotPresent

       command:

       - /bin/sh

       - -c

       - date; pwd;  echo Hello from kubernetes cluster

     restartPolicy: OnFailure

     $ kubectl  get cronjob

       k> troubleshooting:----

       ----------

       ]# systemctl is-active kubelet

       # systemctl is-enabled kubelet

      # systemctl is-enabled crio

      # systemctl is-active crio

      # systemctl enable --now  crio

      # systemctl enable --now  kubelet

      static pod:---

      ========

     ------->>> controlled by kubelet

     ------->>>> locations:---  /etc/kubernetes/manifests

     ------->>>> /var/lib/kubelet/config.yaml  ----- kubelet config file

     # crictl ps

       Autoscaling:----

       metrics server (for on prem )

       -------

       installation of metrics:---

       # git clone <https://github.com/gauravkumar9130/metrics-server.git>

       # kubectl apply -f metrics-server/.

       # kubectl  top pods

       # kubectl  top nodes

       autoscaling:---

       a> vertical scaling:--memory/cpu/vm

       b> horizental scaling:---   pods

             (hpa------ horizental pod autoscaler)

             default cpu consumption-80%

             # kubectl  get hpa

# kubectl scale deployment dep1 --replicas 10    --------->>> manual scaling

# kubectl autoscale deployment dep1 --min 2 --max 20 --cpu-percent 70    -------->>> auto scaling

# kubectl delete hpa dep1

Cluster Maintainance

======================

Upgrade OS :---

================

I want to upgrade OS of my worker-1 in cluster:---

#kubectl cordon master

#kubectl drain worker1 --force --ignore-demonsets  -> to drain node

Remember thet while running this command everything will be deleted like emptydir, hostpath that uses local storage, standalone pod. if any pod under

replica ----then automatically it will be recreated in othe worker node.

except daemon set.

if error is coming from the above command due to storage:---

then use this command:---

#kubectl drain worker1 --force --ignore-demonsets --delete-empty-dir   --- it will delete all the empty directory

#kubectl get pods -o wide -----all the pods will be transfered to worker2

After completion of the OS upgradation

#kubectl cordon <nodename> -> pods will not evict it will make node as unschedulable

#kubectl uncordon <nodename> -> to join back again in cluster

To join any new workernode in the cluster

===========================================

--- from master we need to generate a add token & need to paste that token

  in new workernode .

#kubeadm token create --print-join-command      ---- it will give the join code

Process --2

=====================

Version upgradation of Kubernetes cluster:----

#kubectl get nodes

Version

===========

v1.20.1

v1 -> major version  ---- take a very long time to come

20 -> minor version (every few month) ---- take 2-3 months

1 -> patch (bug fixes)  ---- it will take few days/5-7 days to release new patch

process:

1) alpha -> features are disabled by default and may be bug in it

2) beta -> code is well tested and new features are enabled

3) stable

All components are in same version: kube-apiserver,controller-manager,kube-schduler,kubelet,kube-proxy,kubectl

some components have their version: etc,coredns  --- it is not maintained by kubernetes , some other org maintained.

NOTE: we need to make sure that kube-apiserver version is not lower than other components version.

                                                kube-apiserver(v1.20)

controller-manager(v1.19 or v1.20)                                 kube-scheduler (v1.19 or v1.20)

kubelet (v1.18 or v.19 or v1.20)                                kube-proxy (v1.18 or v1.19 or v1.20)

NOTE: we can upgrade only one version (we are on v1.18) then v1.19 then v1.20

Cluster Maintainance

============================

#kubectl drain <NODE NAME> --force --ignore-daemonsets -> to drain node

#kubectl uncordon <NODE NAME> -> to join back again in cluster

To Upgrade Cluster

=====================

If CentOS/Fedora:

On Master Node

-----------------------

#vim /etc/kubernetes/manifests/kube-apiserver.yaml  ------------------check apiversion from this file

#vim /etc/kubernetes/manifests/kube-apiserver.yaml  ------------------check apiversion from this file

#yum install -y kubeadm-1.22.0-0

#kubeadm upgrade plan

#kubeadm upgrade apply v1.22.0 (IT WILL UPGRADE APISERVER, CONTROLLER MANAGER, SCHEDULER, KUBE-PROXY, ETCD, COREDNS) NOTE: WE NEED TO UPGRADE KUBECTL AND KUBELET MANUALLY

#yum install -y kubelet-1.22.0-0 kubectl-1.22.0-0

#systemctl daemon-reload

#systemctl restart kubelet

Now Upgrade worker nodes so first drain node: #kubectl drain <NODE NAME> --force --ignore-daemonsets

Go to worker node

-------------------

#yum install -y kubelet-1.22.0-0

#systemctl daemon-reload

#systemctl restart kubelet

ONCE DONE THEN JOIN BACK AGAIN

If Ubuntu based:---------  (in exam)

---------------------------------------------------

On master node

---------------

#apt-mark unhold kubeadm && \

apt-get update && apt-get install -y kubeadm=1.22.x-00 && \

apt-mark hold kubeadm

#kubeadm upgrade plan

#kubeadm upgrade apply v1.22.0 (IT WILL UPGRADE APISERVER, CONTROLLER MANAGER, SCHEDULER, KUBE-PROXY, ETCD, COREDNS) NOTE: WE WILL NEED TO UPGRADE KUBECTL AND KUBELET MANUALLY

#  apt-mark unhold kubelet kubectl && \

    apt-get update && apt-get install -y kubelet=1.22.x-00 kubectl=1.22.x-00 && \

    apt-mark hold kubelet kubectl

#systemctl daemon-reload

#systemctl restart kubelet

Now Upgrade worker nodes so first drain node: #kubectl drain <NODE NAME> --force --ignore-daemonsets

Go to worker node

-------------------

#yum install -y kubelet-1.22.0-0

#systemctl daemon-reload

#systemctl restart kubelet

ONCE DONE THEN JOIN BACK AGAIN''

FOR ETCD BACKUP & RESTORE:----

=========================

#  yum/apt-get  install epel-release\* -y     ------> yum for centos/redhat       apt-get ---->> for debian/ubuntu

 # yum/apt-get install etcd\*  -y

to take backup:---

ETCDCTL\_API=3 etcdctl --endpoints=[https://127.0.0.1:2379](https://127.0.0.1:2379/) --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/healthcheck-client.crt --key /etc/kubernetes/pki/etcd/healthcheck-client.key snapshot save myetcdsnap.db

for restore:---

--------------

 ETCDCTL\_API=3 etcdctl --endpoints=[https://127.0.0.1:2379](https://127.0.0.1:2379/) --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/healthcheck-client.crt --key /etc/kubernetes/pki/etcd/healthcheck-client.key snapshot status myetcdsnap.db

After performing restore operation---- a folder with the name of default.etcd will be created automatically inside the location where backup was there.

#cd default.etcd/

#rm -rf  /var/lib/etc/member

#mv  member   /var/lib/etcd/

now to to check wheather your resources are available or not...

if you are unable to get -------- restart your cluster.

n Master Node

-----------------------

#vim /etc/kubernetes/manifests/kube-apiserver.yaml  ------------------check apiversion from this file

#vim /etc/kubernetes/manifests/kube-apiserver.yaml  ------------------check apiversion from this file

#yum install -y kubeadm-1.22.0-0

#kubeadm upgrade plan

#kubeadm upgrade apply v1.22.0 (IT WILL UPGRADE APISERVER, CONTROLLER MANAGER, SCHEDULER, KUBE-PROXY, ETCD, COREDNS) NOTE: WE NEED TO UPGRADE KUBECTL AND KUBELET MANUALLY

#yum install -y kubelet-1.22.0-0 kubectl-1.22.0-0

#systemctl daemon-reload

#systemctl restart kubelet

Now Upgrade worker nodes so first drain node: #kubectl drain <NODE NAME> --force --ignore-daemonsets

Go to worker node

-------------------

#yum install -y kubelet-1.22.0-0

#systemctl daemon-reload

#systemctl restart kubelet

ONCE DONE THEN JOIN BACK AGAIN

steps for upgradation----

1> # kubectl  cordon master

2> # kubectl drain master --ignore-daemonsets --delete-emptydir-data --force  (step 1 & 2 are common for both centos & debian)

n master node   (for debian or ubuntu)

---------------

3> #apt-mark unhold kubeadm  kubectl kubelet

4> #kubeadm upgrade plan

5> # apt-get update -y && apt-get upgrade kubectl=1.25.4-00 kubelet=1.25.4-00 kubeadm=1.25.4-00 -y

# yum  update -y && yum  upgrade kubectl=1.25.4-00 kubelet=1.25.4-00 kubeadm=1.25.4-00 -y

Steps for centos:---

1> # kubectl  cordon master

2> # kubectl drain master --ignore-daemonsets --delete-emptydir-data --force  (step 1 & 2 are common for both centos & debian)

3>  #kubeadm upgrade plan

4>  # yum  update -y && yum  upgrade kubectl=1.25.4-00 kubelet=1.25.4-00 kubeadm=1.25.4-00 -y

5> #systemctl restart kubelet

6> systemctl enable kubelet

7> #kubectl uncordon master.

steps for debian or ubuntu:----

=======================

> # kubectl  cordon master

2> # kubectl drain master --ignore-daemonsets --delete-emptydir-data --force  (step 1 & 2 are common for both centos & debian)

3> #apt-mark unhold kubeadm  kubectl kubelet

4>  #kubeadm upgrade plan

5>  # apt-get update -y && apt-get upgrade kubectl=1.25.4-00 kubelet=1.25.4-00 kubeadm=1.25.4-00 -y

6> #apt-mark hold kubeadm  kubectl kubelet

7> #systemctl enable --now kubelet

8> #kubectl uncordone master

Go to worker node

-------------------

#apt-get install -y kubelet-1.22.0-0

#systemctl daemon-reload

#systemctl restart kubelet

ONCE DONE THEN JOIN BACK AGAIN