Kubernetes (k8s)

t.me/vishalk17 github.com/vishalk17

==========

Kubernates is an opensource container management tool which automates container deployment ,container scaling, and load balancing.

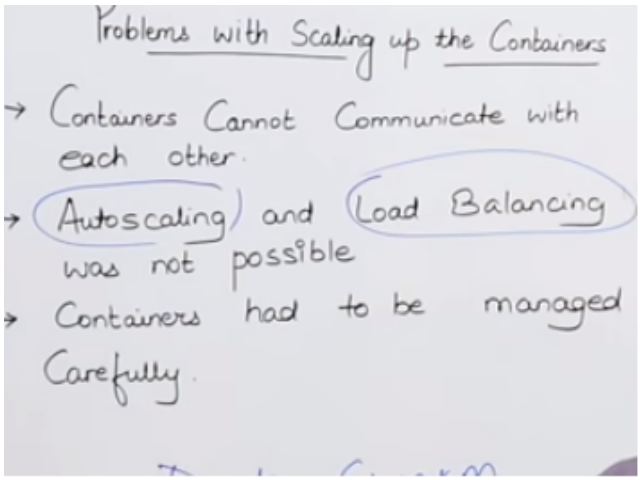
It schedules , runs and manages isolated containers which are running on virtual /physical/cloud machines.

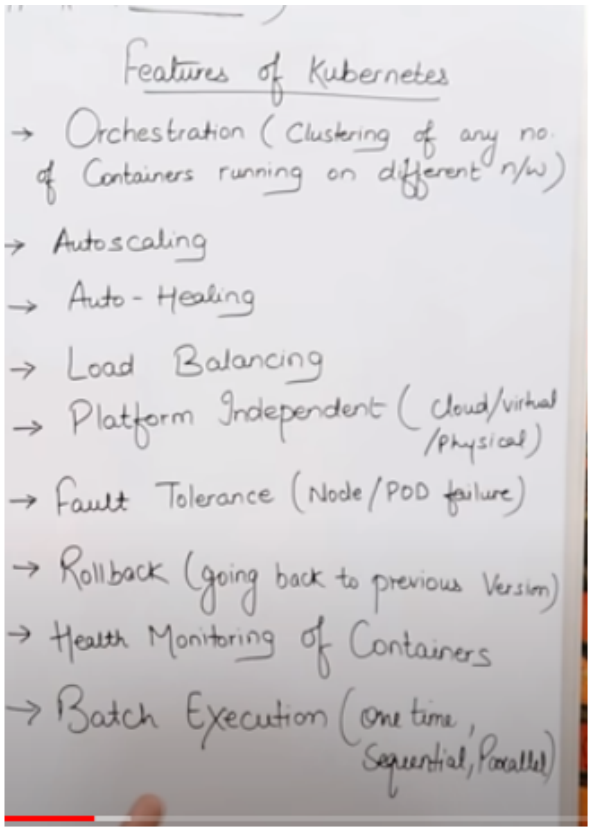
Kubernates is an opensource container management tool which automates container deployment ,container scaling, and load balancing.

It schedules , runs and manages isolated containers which are running on virtual /physical/cloud machines.

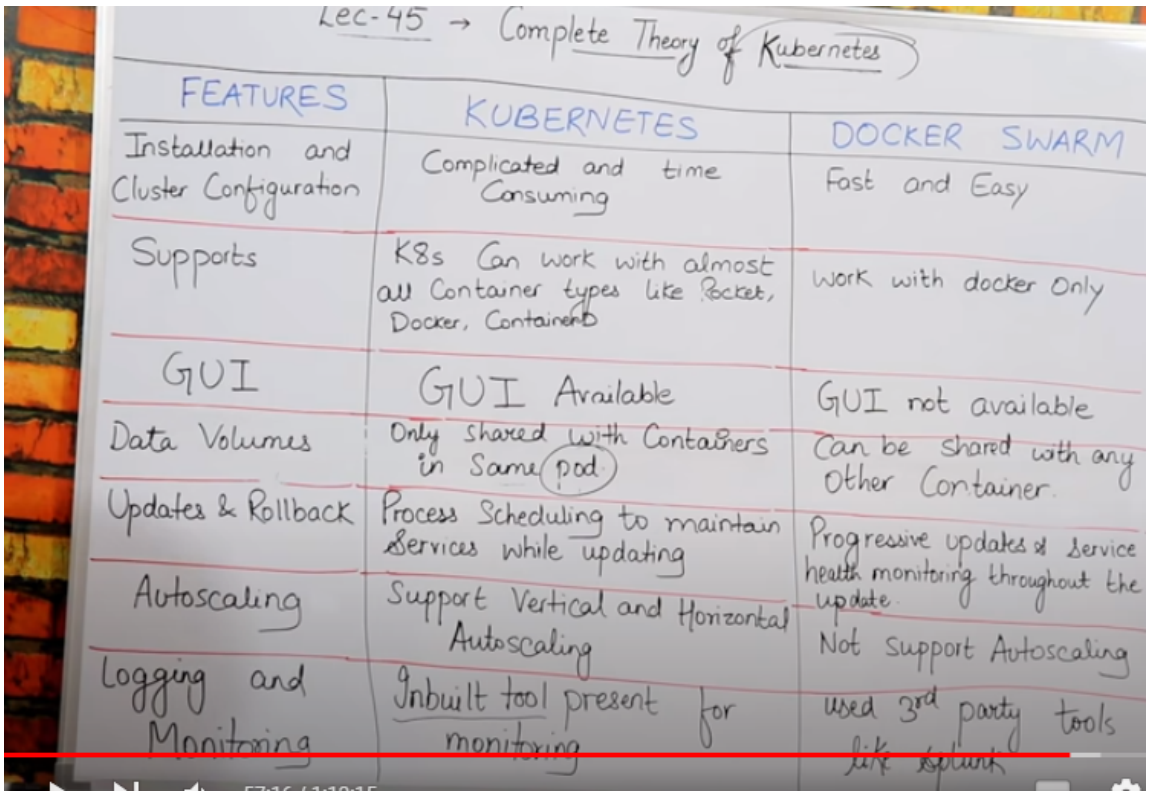
Monolithic application

Microservices





==========


Lec-45 → Complete Theory of Kubernetes
FEATURES
KUBERNETES
DOCKER SWARM
Installation and Cluster Configuration
Complicated and time Consuming
Fast and Easy
Supports
K8s Can work with almost all Container types like Rocket, Docker, Containerd
Work with docker Only
GUI
GUI Available
GUI not available
Data Volumes
Only shared with Containers in Same pod
Can be shared with any Other Container
Updates & Rollback
Process Scheduling to maintain services while updating
Progressive updates & service health monitoring throughout the update
Autoscaling
Support Vertical and Horizontal Autoscaling
Not Support Autoscaling
Logging and Monitoring
Inbuilt tool present for monitoring
used 3rd party tools like

==========

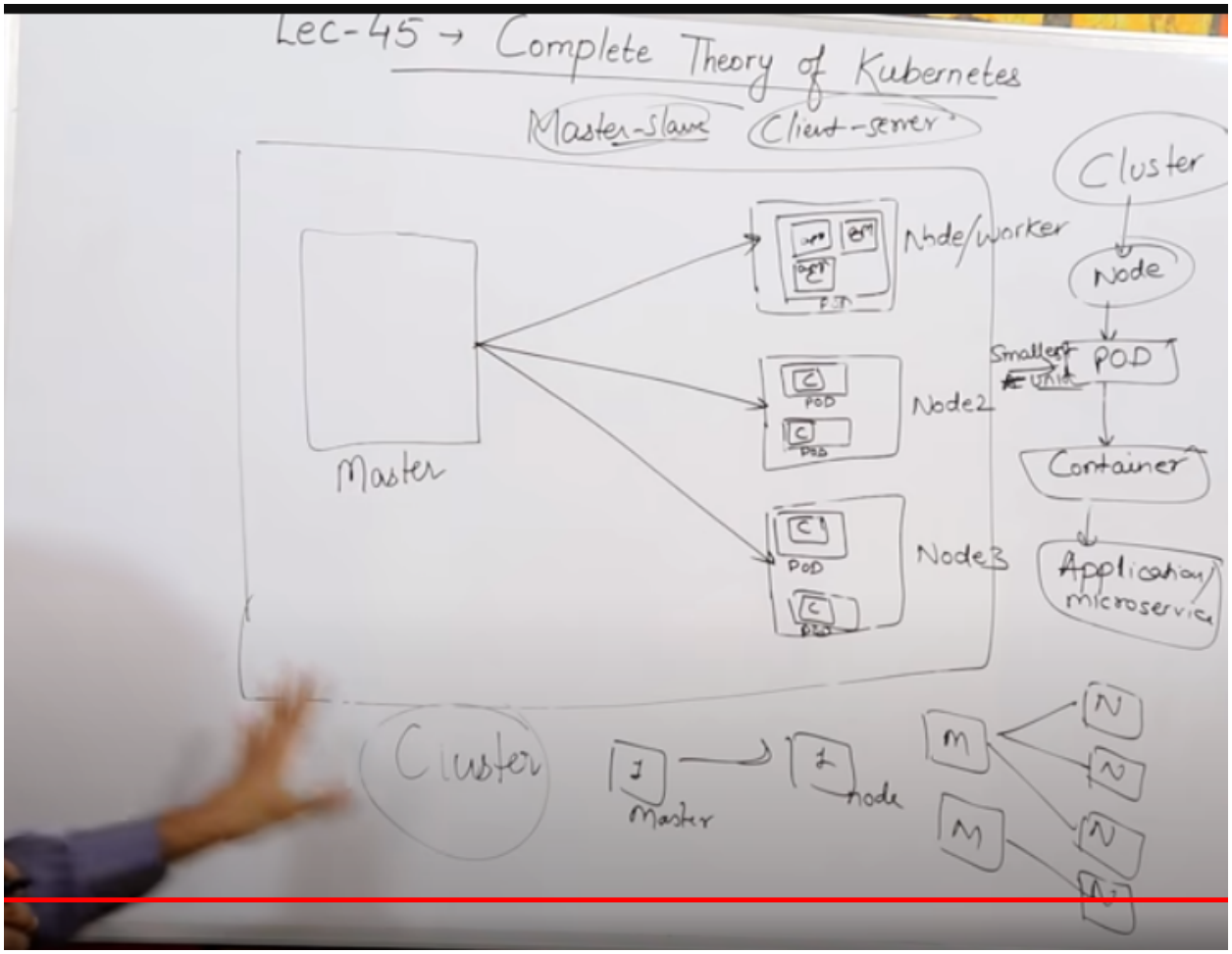
- A pod is the smallest execution unit in Kubernetes.

- can create multiples nodes in cluster

- in node we can create multiple pod

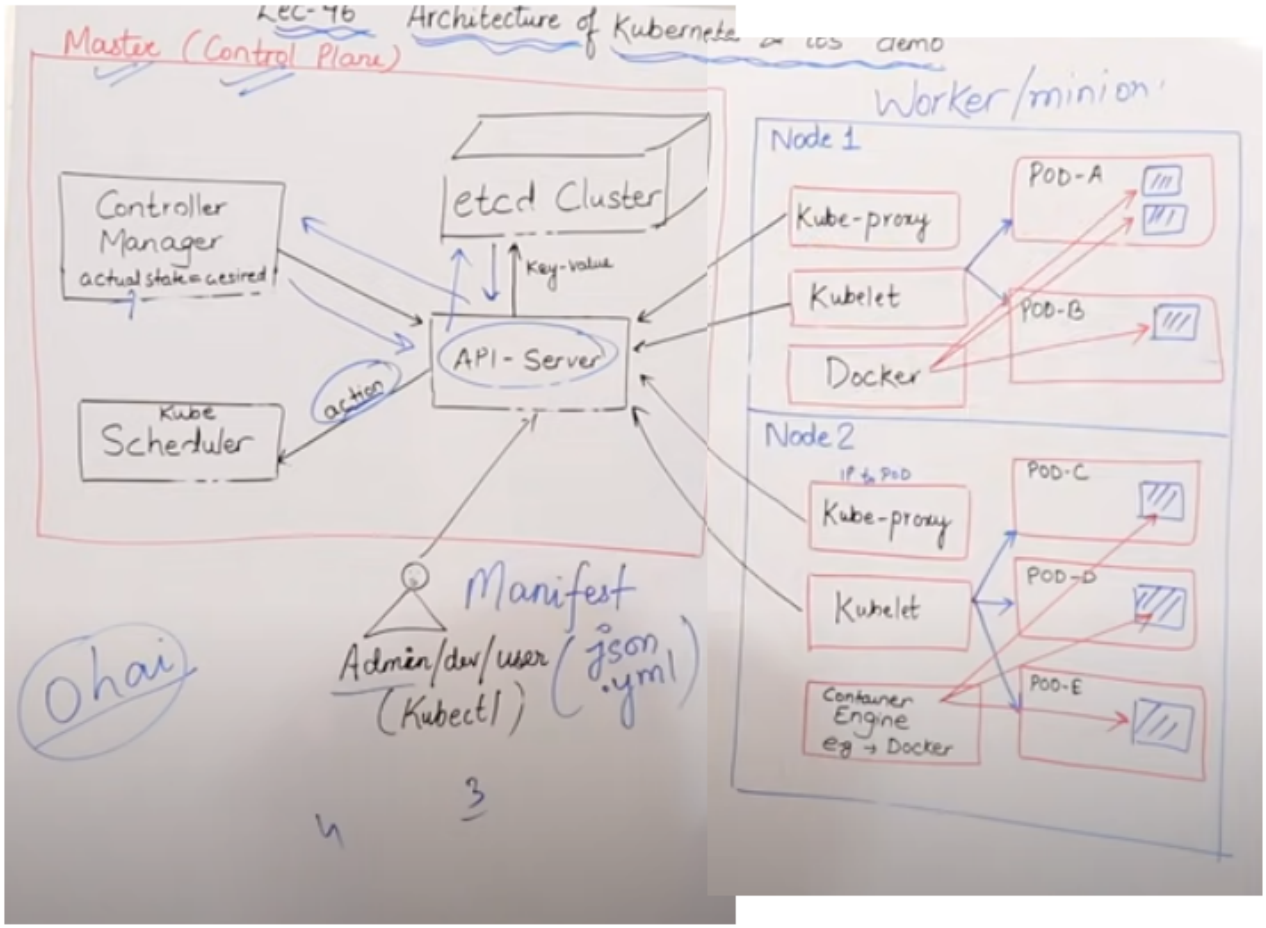
- in single pod we can create multiple container

- Kubernetes communicate with pod not directly to the container

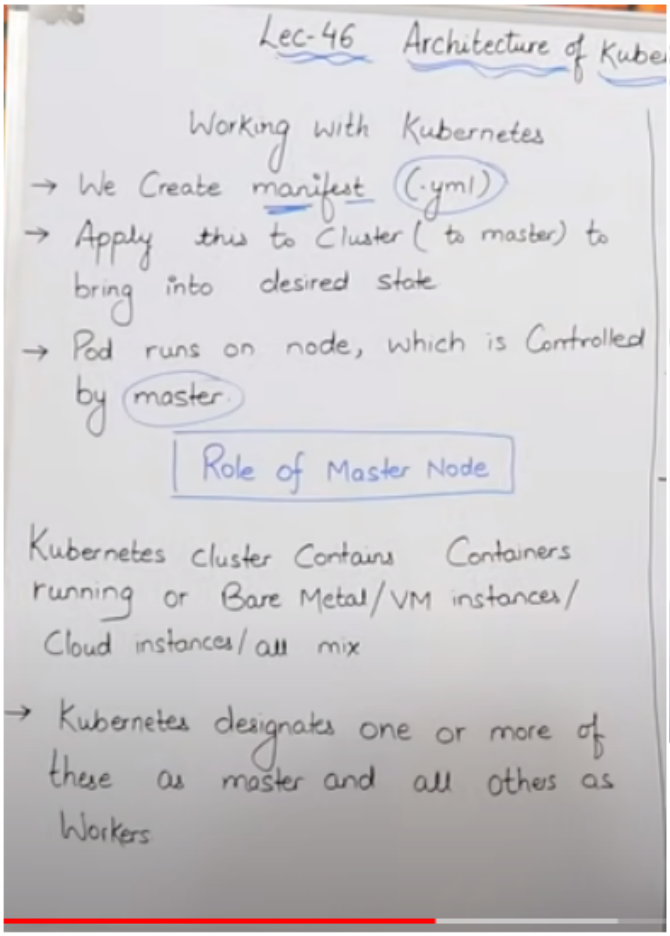


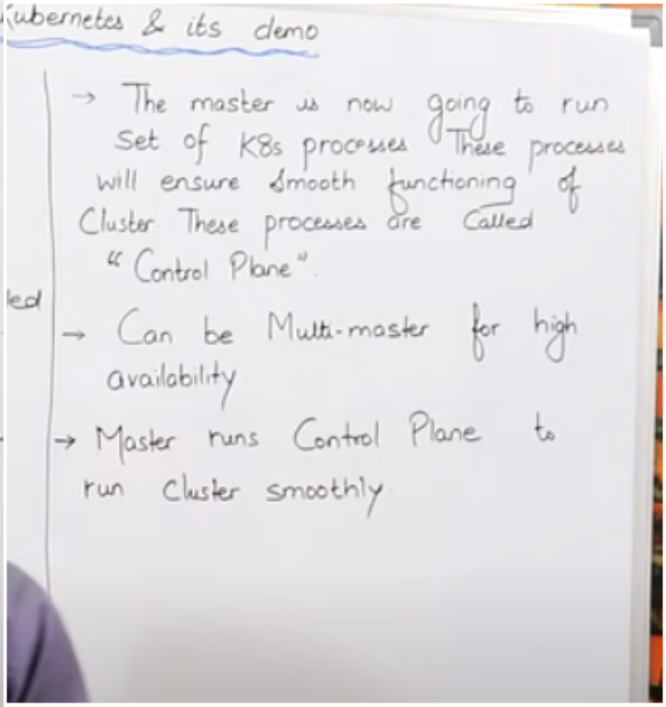
==========

Arch :

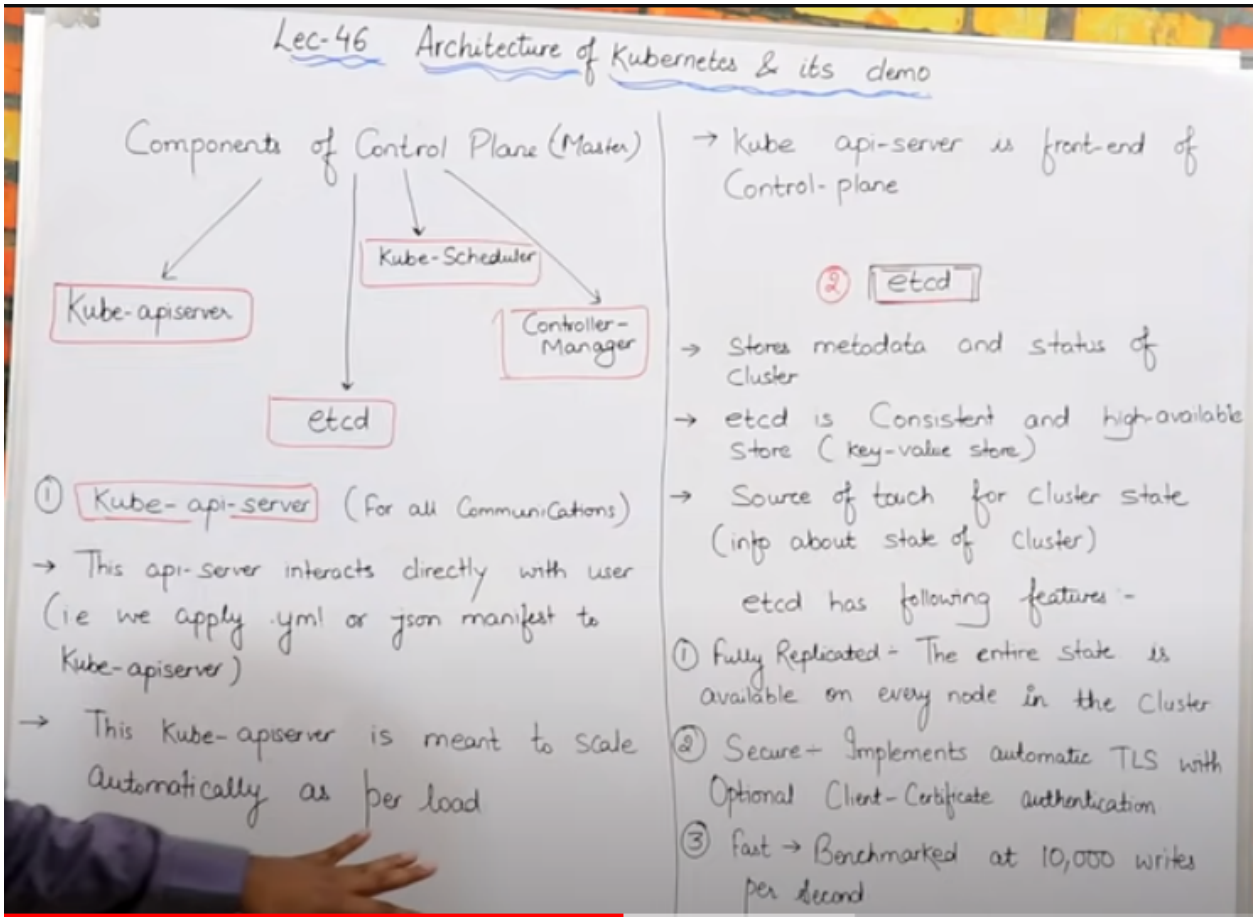

Master (Control Plane)
Architecture of Kubernete
Worker/minion
Node 1
Node 2
Controller Manager
actual state = desired
etcd Cluster
Key-value
API-Server
Kube Scheduler
action
Kube-proxy
Kubelet
Docker
POD-A
POD-B
POD-C
POD-D
POD-E
IP & POD
Container Engine
eg → Docker
Manifest
Admin/dev/user (Kubectl)
(json .yml)
Ohai

==========

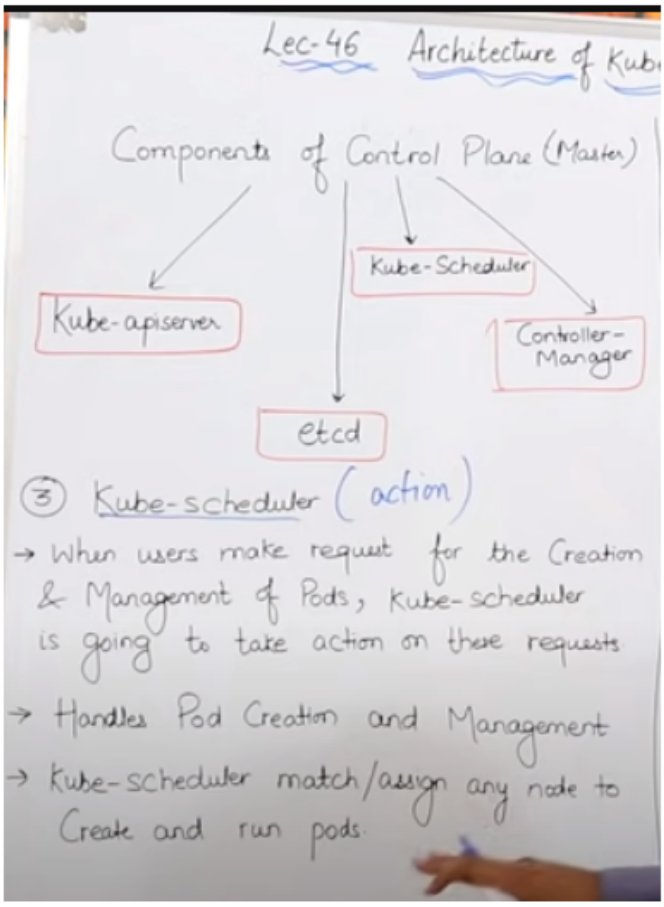

Lec-46 Architecture of Kube
Working with Kubernetes
→ We Create manifest (.yml)
→ Apply this to Cluster (to master) to bring into desired state
→ Pod runs on node, which is Controlled by master.
Role of Master Node
Kubernetes cluster Contains Containers running or Bare Metal/VM instances/ Cloud instances/ all mix
→ Kubernetes designates one or more of these as master and all others as Workers

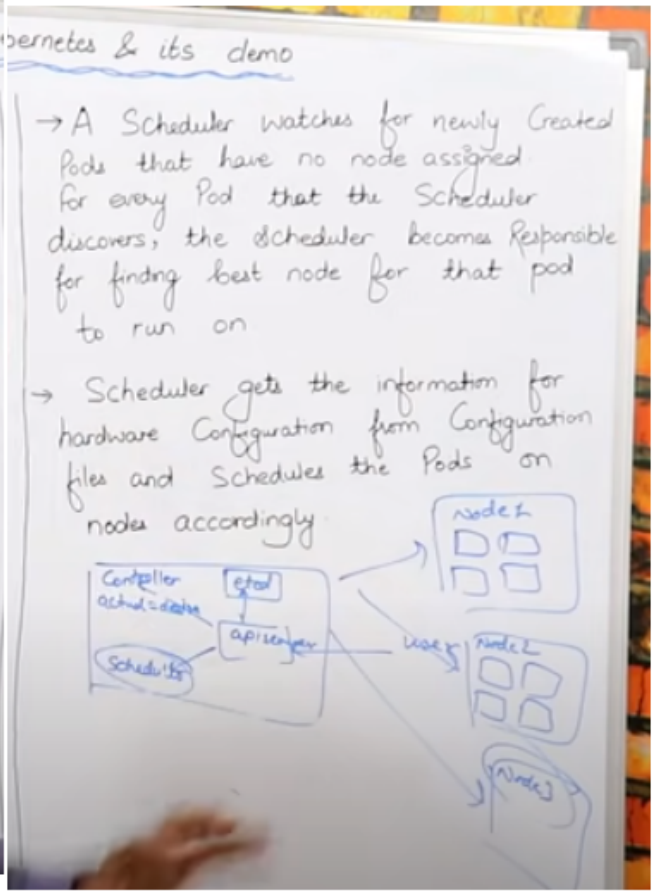

ubernetes & its demo
→ The master is now going to run set of K8s processes These processes will ensure Smooth functioning of Cluster These processes are Called "Control Plane".
→ Can be Multi-master for high availability
→ Master runs Control Plane to run Cluster smoothly

==========

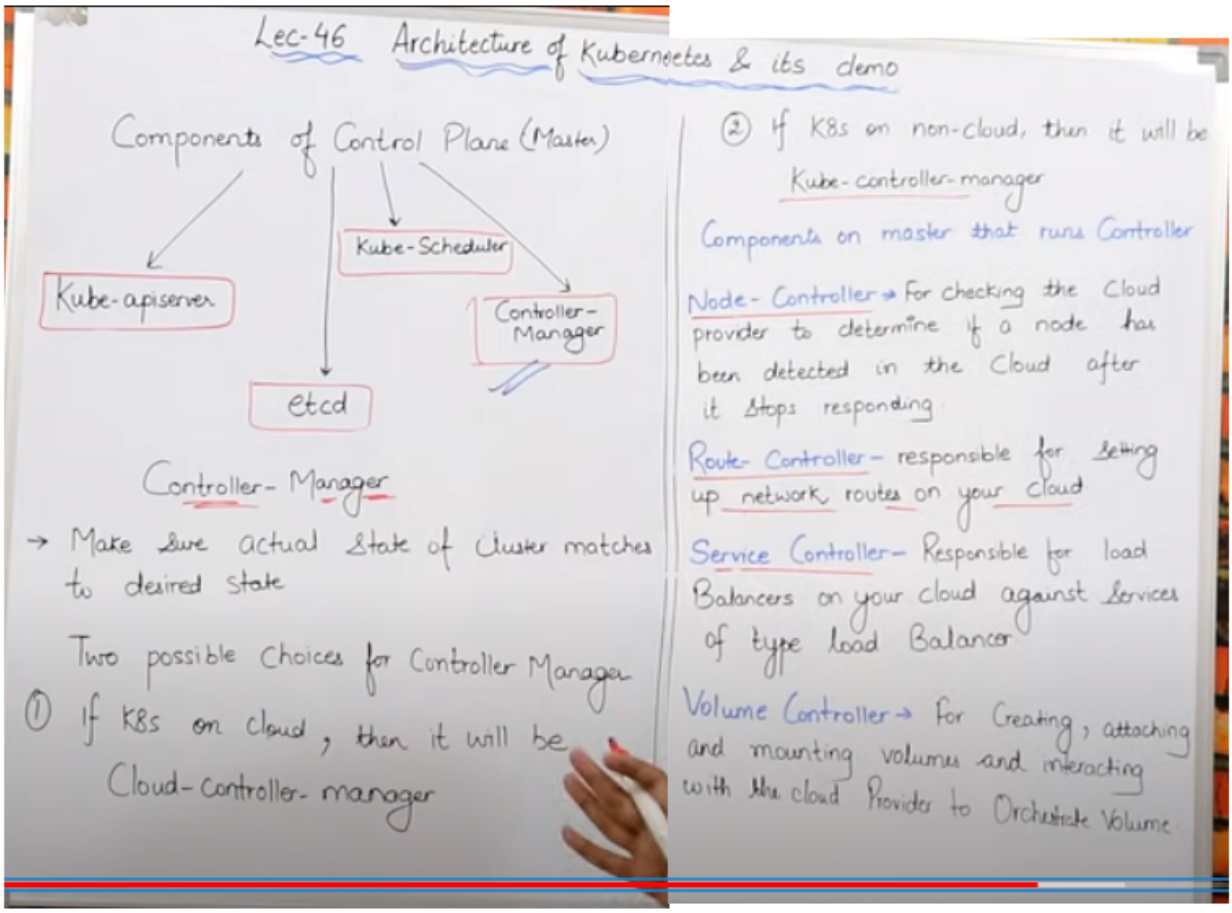

Lec-46 Architecture of Kubernetes & its demo
Components of Control Plane (Master)
Kube-apiserver
Kube-Scheduler
Controller-Manager
etcd
① Kube-api-server (For all Communications)
→ This api-server interacts directly with user (ie we apply .yml or json manifest to Kube-apiserver)
→ This Kube-apiserver is meant to scale automatically as per load
→ Kube api-server is front-end of Control-plane
② etcd
→ Stores metadata and status of cluster
→ etcd is Consistent and high-available store (key-value store)
→ Source of truth for cluster state (info about state of cluster)
etcd has following features -
① Fully Replicated - The entire state is available on every node in the cluster
② Secure - Implements automatic TLS with Optional Client-Certificate authentication
③ Fast → Benchmarked at 10,000 writes per second

==========

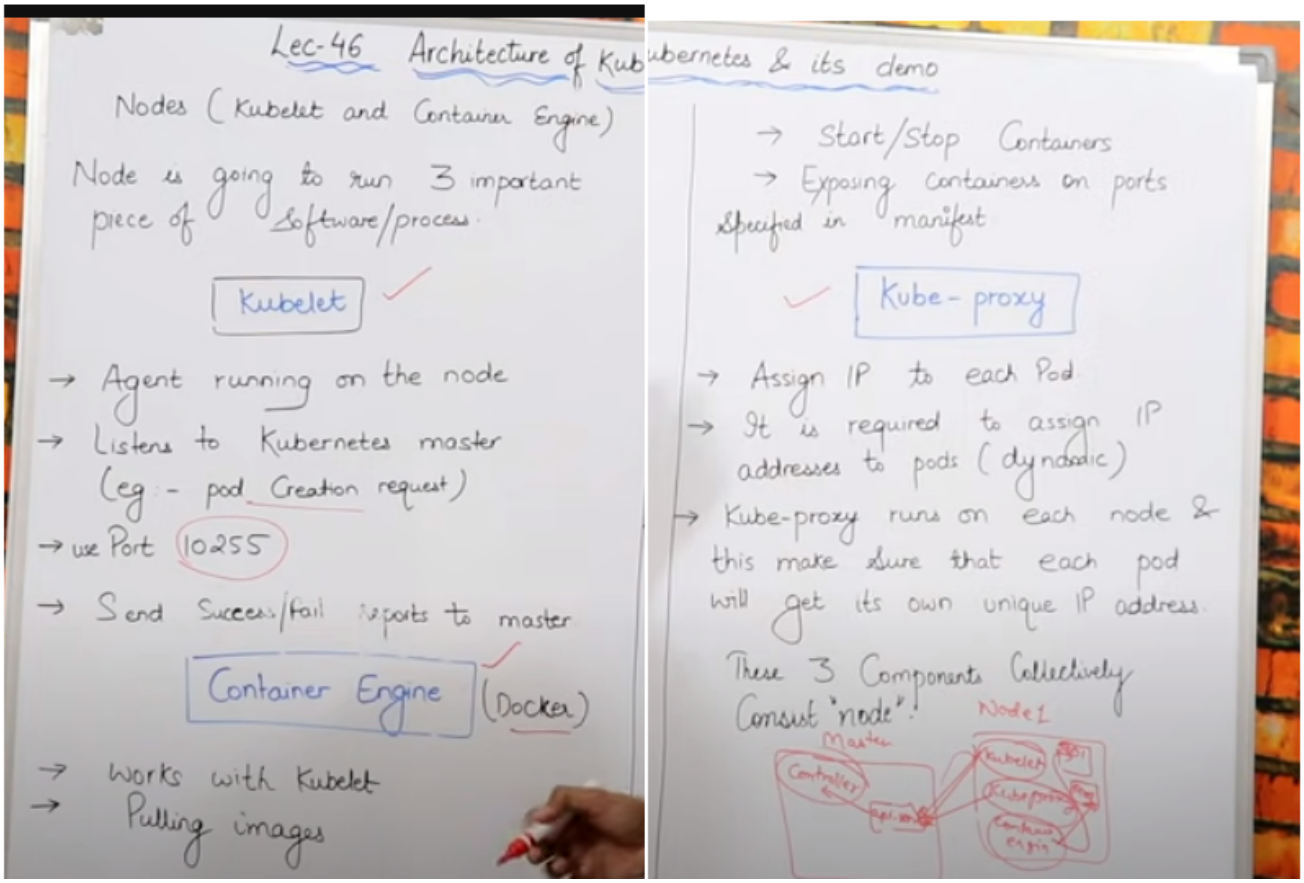

Lec-46 Architecture of Kubernetes & its demo
Components of Control Plane (Master)
Kube-apiserver
Kube-Scheduler
Controller-Manager
etcd
(3) Kube-scheduler (action)
→ When users make request for the Creation & Management of Pods, Kube-scheduler is going to take action on these requests
→ Handles Pod Creation and Management
→ Kube-scheduler match/assign any node to Create and run pods


→ A Scheduler watches for newly Created Pods that have no node assigned. For every Pod that the Scheduler discovers, the Scheduler becomes Responsible for finding best node for that pod to run on
→ Scheduler gets the information for hardware Configuration from Configuration files and Schedules the Pods on nodes accordingly
Node1
Node2
Node3
etcd
apiserver
Scheduler

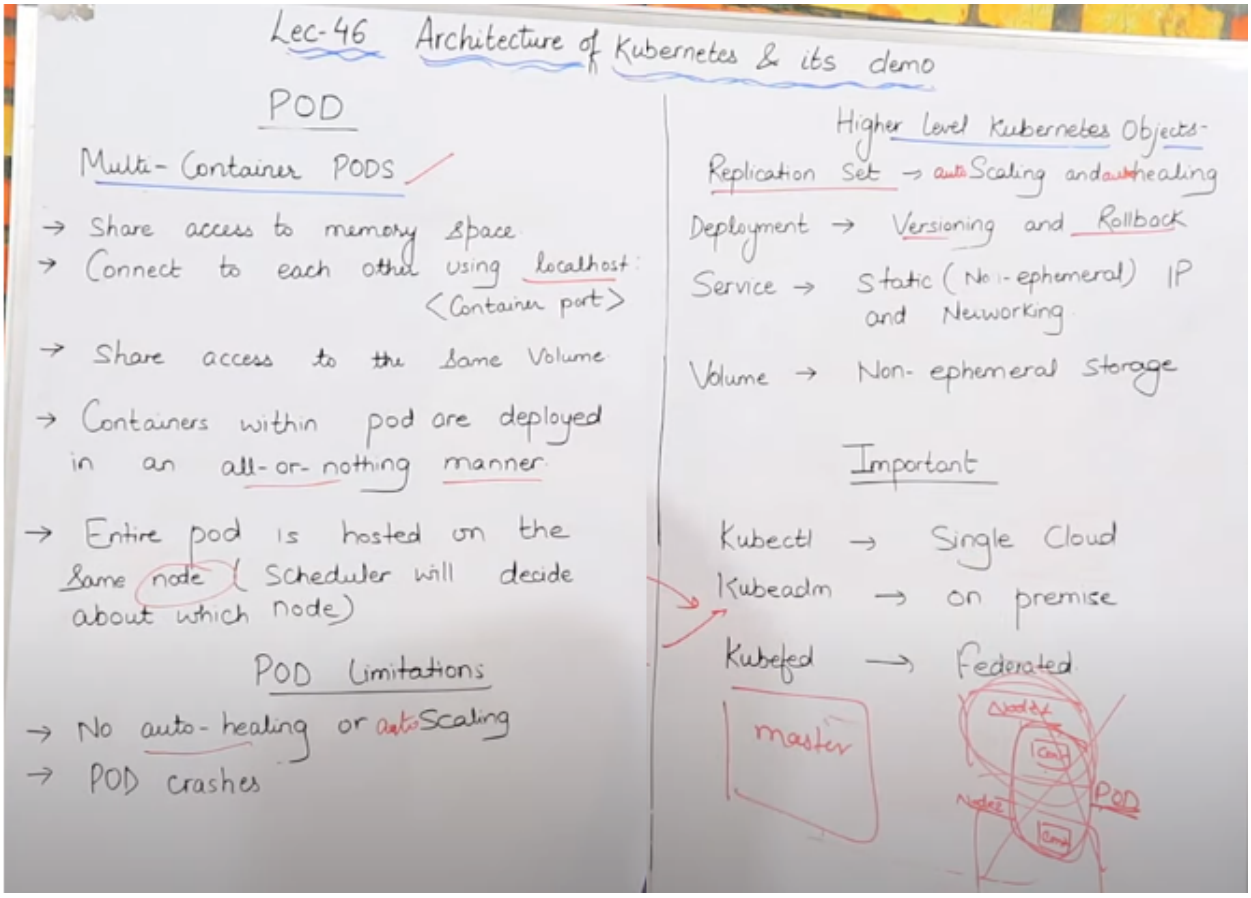
==========


Lec-46 Architecture of Kubernetes & its demo
Components of Control Plane (Master)
Kube-apiserver
Kube-Scheduler
Controller-Manager
etcd
Controller-Manager
→ Make sure actual state of cluster matches to desired state
Two possible choices for Controller Manager
① If K8s on cloud, then it will be Cloud-controller-manager
② If K8s on non-cloud, then it will be Kube-controller-manager
Components on master that runs Controller
Node-Controller → For checking the cloud provider to determine if a node has been detected in the cloud after it stops responding
Route-Controller - responsible for setting up network routes on your cloud
Service Controller - Responsible for load Balancers on your cloud against services of type load Balancer
Volume Controller → for Creating, attaching and mounting volumes and interacting with the cloud Provider to Orchestrate Volume

==========

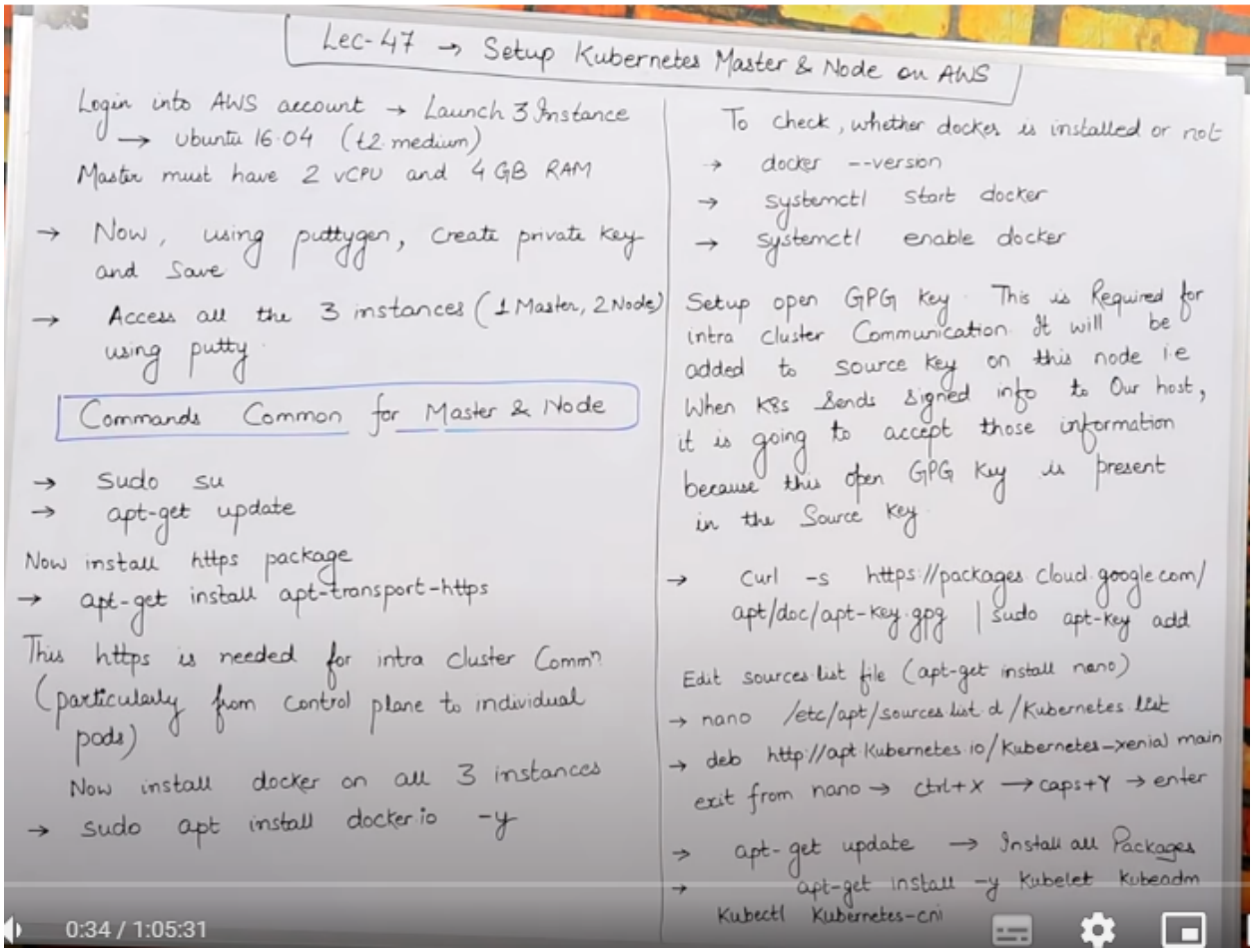

Lec-46 Architecture of Kubernetes & its demo
Nodes (Kubelet and Container Engine)
Node is going to run 3 important piece of Software/process
Kubelet
→ Agent running on the node
→ Listens to Kubernetes master (eg:- pod Creation request)
→ use Port 10255
→ Send Success/fail reports to master
Container Engine (Docker)
→ Works with Kubelet
→ Pulling images
→ Start/Stop Containers
→ Exposing containers on ports specified in manifest
Kube-proxy
→ Assign IP to each Pod
→ It is required to assign IP addresses to pods (dynamic)
→ Kube-proxy runs on each node & this make sure that each pod will get its own unique IP address.
These 3 Components Collectively Consist "node".
Master
Node 1

==========

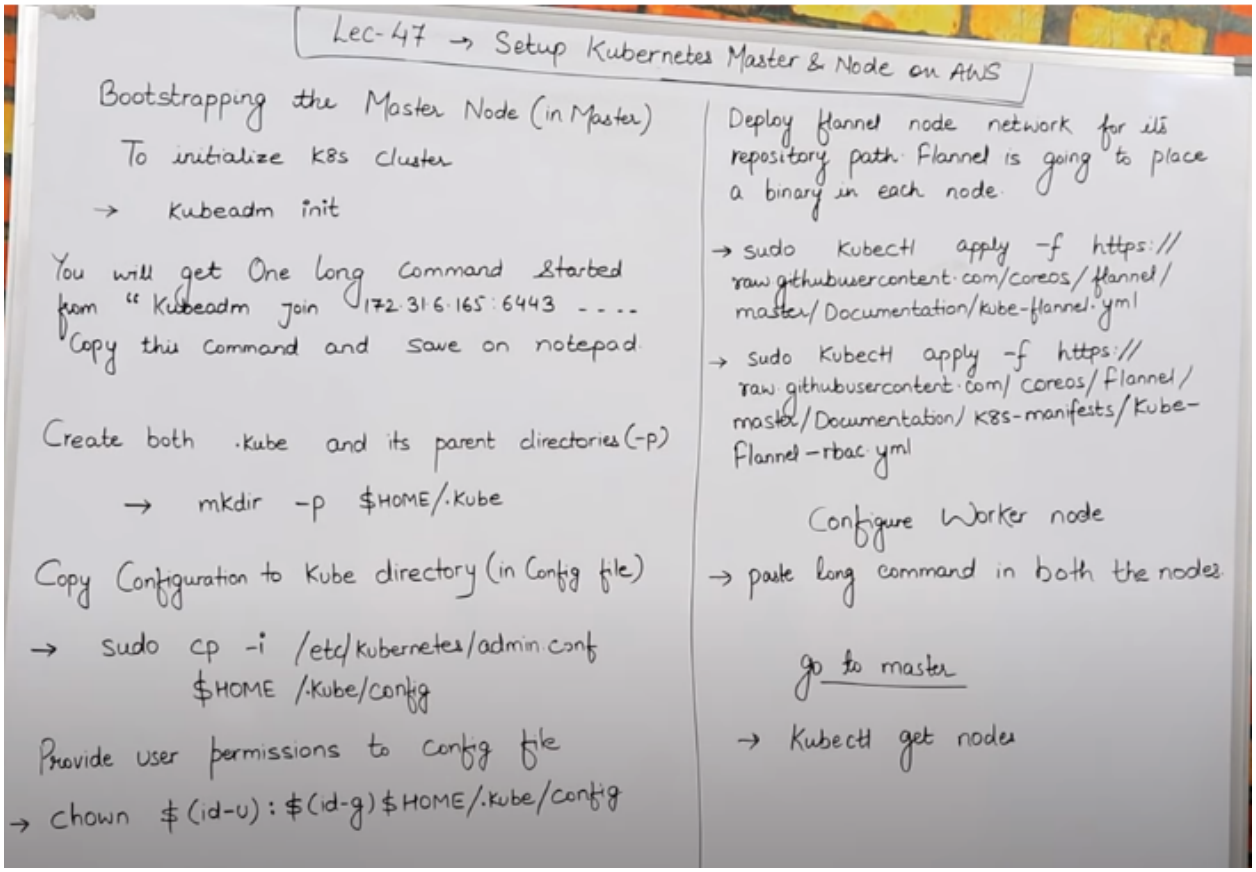

Lec-46 Architecture of Kubernetes & its demo
POD
Multi-Container PODS
→ Share access to memory space
→ Connect to each other using localhost: <Container port>
→ Share access to the same Volume
→ Containers within pod are deployed in an all-or-nothing manner
→ Entire pod is hosted on the same node (Scheduler will decide about which node)
POD Limitations
→ No auto-healing or autoScaling
→ POD crashes
Higher level Kubernetes Objects-
Replication Set → autoScaling and autohealing
Deployment → Versioning and Rollback
Service → Static (No i-ephemeral) IP and Networking
Volume → Non-ephemeral storage
Important
Kubectl → Single Cloud
Kubeadm → on premise
Kubefed → Federated
master
POD

==========

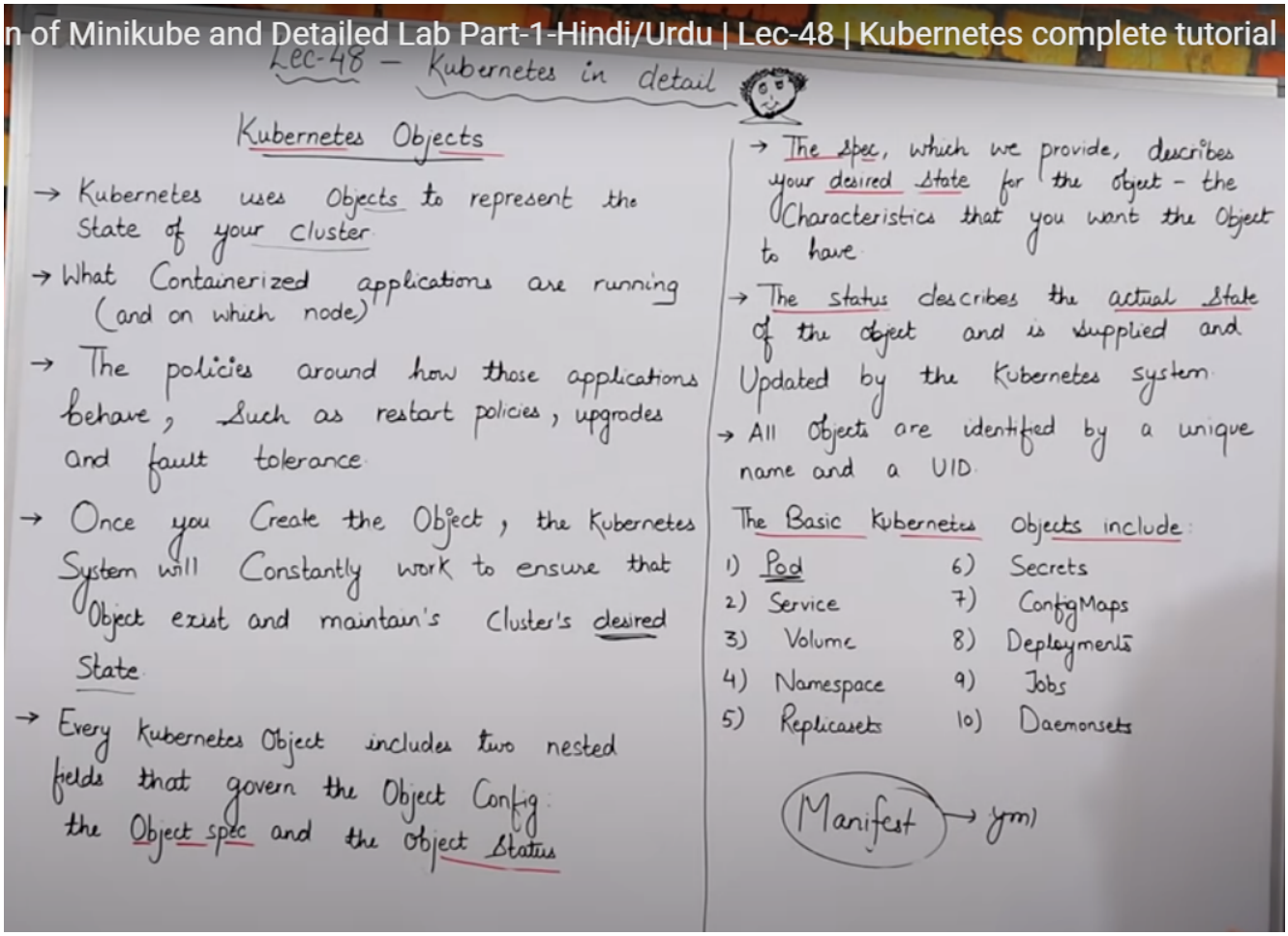
Setup k8s Master & Node on AWS



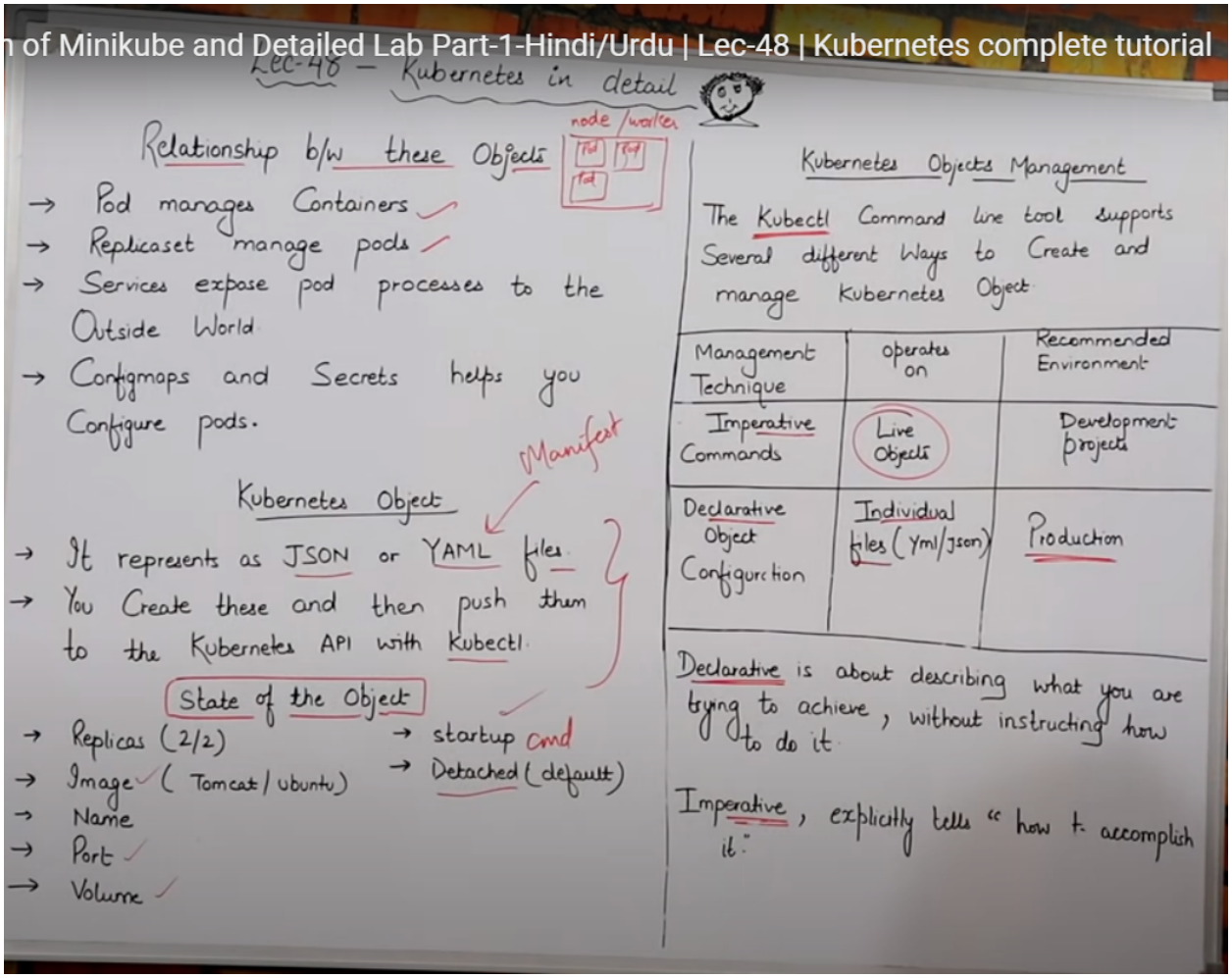
==========


Lec-47 → Setup Kubernetes Master & Node on AWS
Bootstrapping the Master Node (in Master)
To initialize k8s cluster
→ Kubeadm init
You will get One long command started from " Kubeadm join 172.31.6.165:6443 ....
Copy this command and save on notepad
Create both .kube and its parent directories (-p)
→ mkdir -p $HOME/.kube
Copy Configuration to Kube directory (in Config file)
→ sudo cp -i /etc/kubernetes/admin.conf $HOME /.kube/config
Provide user permissions to config file
→ chown $(id-u):$(id-g) $HOME/.kube/config
Deploy flannel node network for its repository path. Flannel is going to place a binary in each node.
→ sudo Kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
→ sudo Kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml
Configure Worker node
→ paste long command in both the nodes
go to master
→ Kubectl get nodes

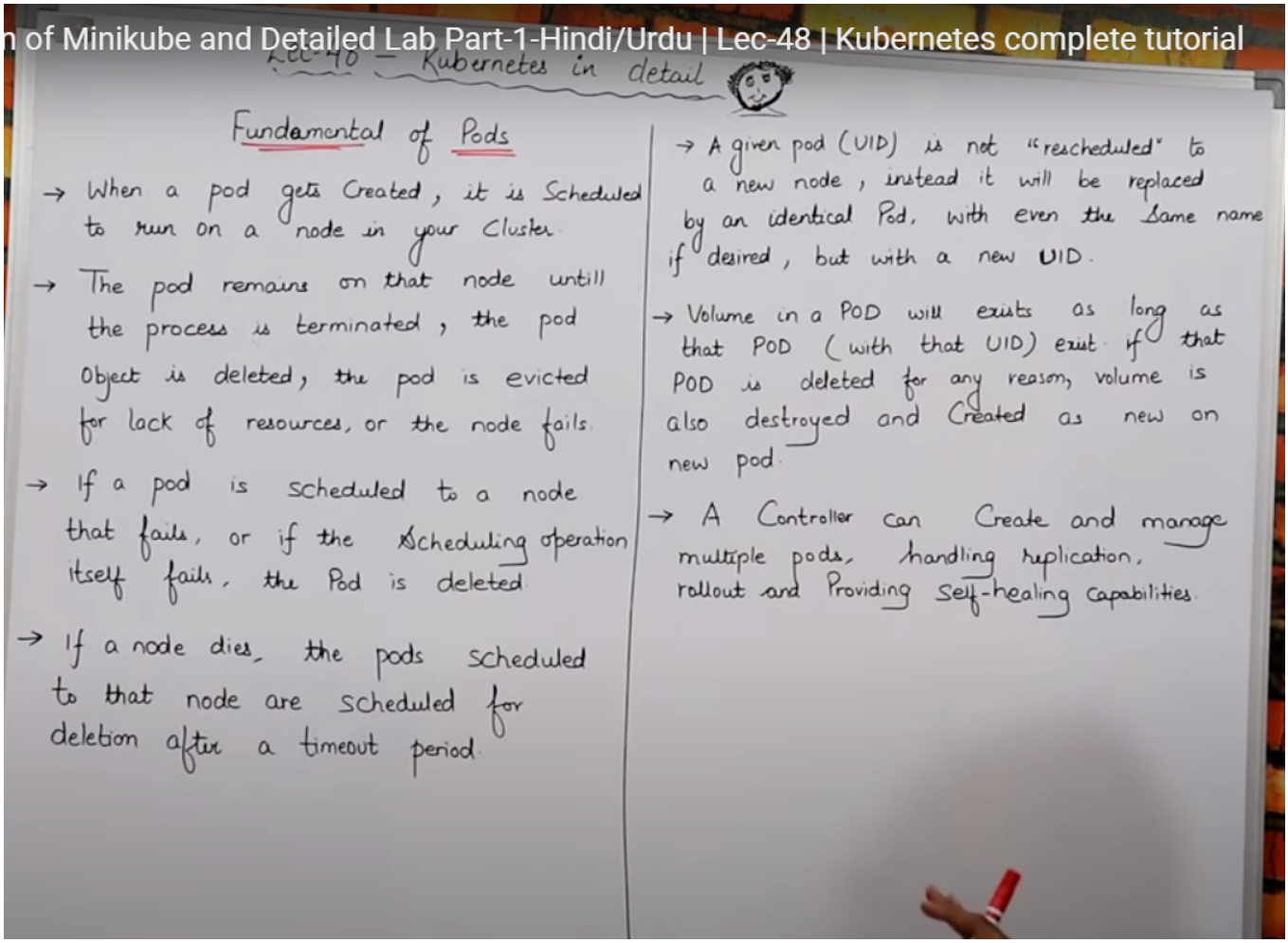
==========


n of Minikube and Detailed Lab Part-1-Hindi/Urdu | Lec-48 | Kubernetes complete tutorial
Lec-48 – Kubernetes in detail
Kubernetes Objects
→ Kubernetes uses Objects to represent the State of your cluster
→ What Containerized applications are running (and on which node)
→ The policies around how those applications behave, such as restart policies, upgrades and fault tolerance
→ Once you Create the Object, the Kubernetes System will Constantly work to ensure that Object exist and maintain's Cluster's desired State
→ Every Kubernetes Object includes two nested fields that govern the Object Config: the Object spec and the Object Status
→ The spec, which we provide, describes your desired State for the Object - the Characteristics that you want the Object to have
→ The status describes the actual State of the Object and is supplied and Updated by the Kubernetes system
→ All Objects are identified by a unique name and a UID
The Basic Kubernetes Objects include:
1) Pod
2) Service
3) Volume
4) Namespace
5) Replicasets
6) Secrets
7) ConfigMaps
8) Deployments
9) Jobs
10) Daemonsets
Manifest → yml

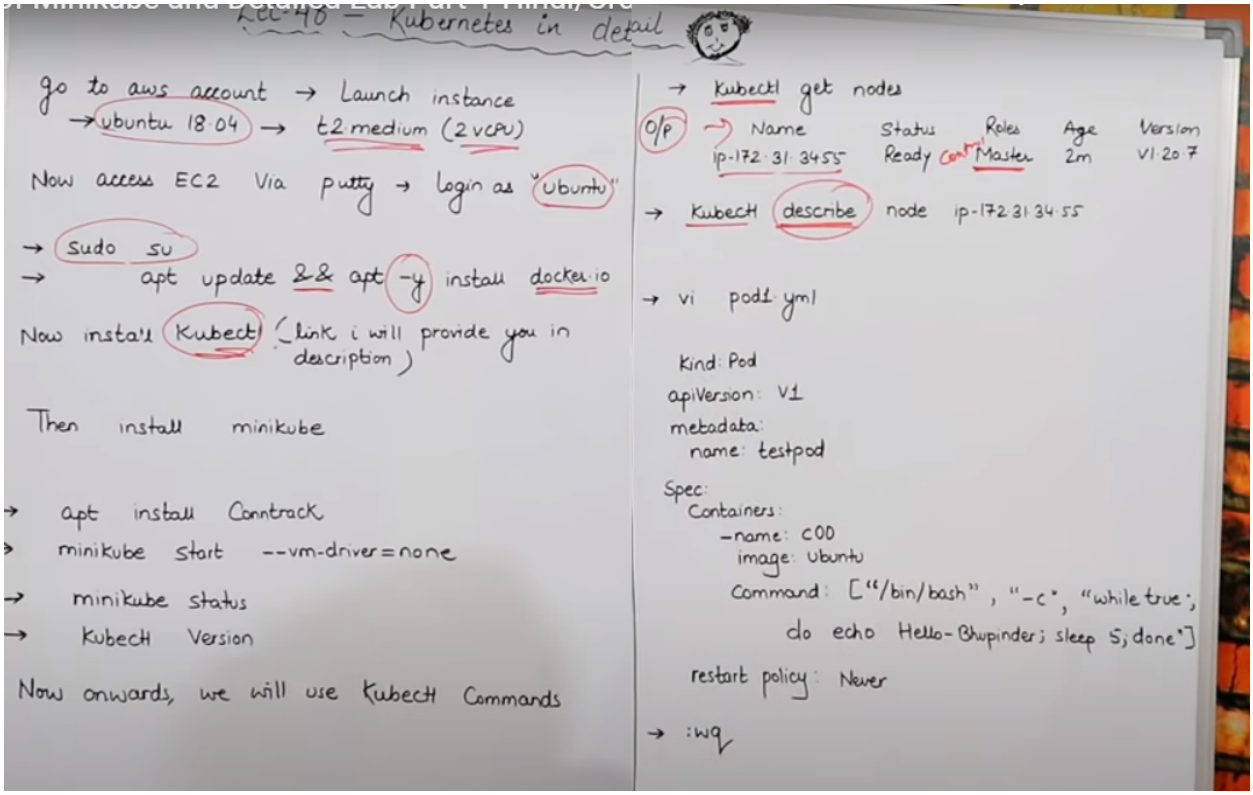
==========


of Minikube and Detailed Lab Part-1-Hindi/Urdu | Lec-48 | Kubernetes complete tutorial
Lec-48 – Kubernetes in detail
Relationship b/w these Objects
node / worker
Pod
Pod
Pod
→ Pod manages Containers
→ Replicaset manage pods
→ Services expose pod processes to the Outside World
→ Configmaps and Secrets helps you Configure pods.
Manifest
Kubernetes Object
→ It represents as JSON or YAML files
→ You Create these and then push them to the Kubernetes API with Kubectl
State of the Object
→ Replicas (2/2)
→ Image (Tomcat / ubuntu)
→ Name
→ Port
→ Volume
→ startup cmd
→ Detached (default)
Kubernetes Objects Management
The Kubectl Command line tool supports Several different Ways to Create and manage Kubernetes Object
Management Technique
operates on
Recommended Environment
Imperative Commands
Live Objects
Development projects
Declarative Object Configuration
Individual files (yml/json)
Production
Declarative is about describing what you are trying to achieve, without instructing how to do it
Imperative, explicitly tells "how t. accomplish it"

==========


n of Minikube and Detailed Lab Part-1-Hindi/Urdu | Lec-48 | Kubernetes complete tutorial
Lec-48 – Kubernetes in detail
Fundamental of Pods
→ When a pod gets Created, it is Scheduled to run on a node in your Cluster.
→ The pod remains on that node untill the process is terminated, the pod Object is deleted, the pod is evicted for lack of resources, or the node fails.
→ If a pod is scheduled to a node that fails, or if the Scheduling operation itself fails, the Pod is deleted.
→ If a node dies, the pods scheduled to that node are scheduled for deletion after a timeout period.
→ A given pod (UID) is not "rescheduled" to a new node, instead it will be replaced by an identical Pod, with even the same name if desired, but with a new UID.
→ Volume in a POD will exists as long as that POD (with that UID) exist. if that POD is deleted for any reason, volume is also destroyed and Created as new on new pod.
→ A Controller can Create and manage multiple pods, handling replication, rollout and Providing self-healing capabilities.

==========




```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep
5 ; done"]
  restartPolicy: Never          # Defaults to Always
```

kubectl apply -f pod1.yml

==========

Annotation: give extra information

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      annotations:
        description: continuous printing of hello vishalk17
      command: ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep
5 ; done"]
  restartPolicy: Never          # Defaults to Always
```

kubectl apply -f pod1.yml [Update existing changes]

kubectl describe pod testpod [Now you see annotation somewhere]

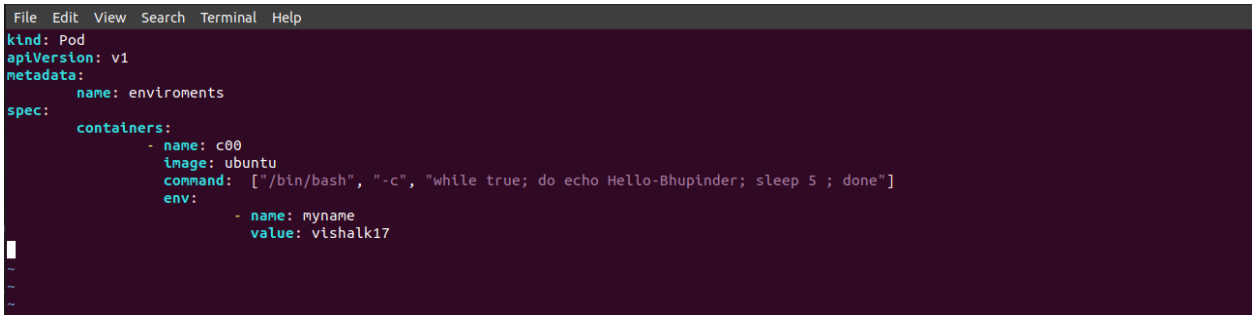
Multicontainer Pod environment :

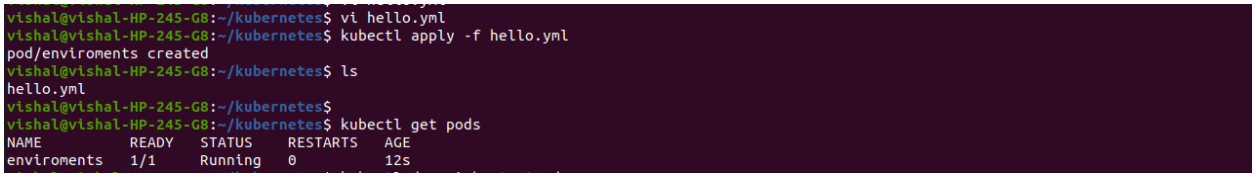
```
kind: Pod
apiVersion: v1
metadata:
  name: testpod3
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Technical-Guftgu; sleep
5 ; done"]
    - name: c01
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep
5 ; done"]
```

==========

environment variables :

```
kind: Pod
apiVersion: v1
metadata:
      name: enviroments
spec:
      containers:
            - name: c00
              image: ubuntu
              command:  ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]
              env:
                    - name: myname
                      value: vishalk17
```





==========

```
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  2m14s  default-scheduler  Successfully assigned default/enviroments to minikube
  Normal  Pulling    2m14s  kubelet            Pulling image "ubuntu"
  Normal  Pulled     2m12s  kubelet            Successfully pulled image "ubuntu" in 2.502636422s (2.502644177s including waiting)
  Normal  Created    2m12s  kubelet            Created container c00
  Normal  Started    2m12s  kubelet            Started container c00
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl exec -it testpod -c c00 -- /bin/bash
Error from server (NotFound): pods "testpod" not found
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl exec -it enviroments -c c00 -- /bin/bash
root@enviroments:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@enviroments:/# echo $myname
vishalk17
```

pod with ports

POD WITH PORTS

```
kind: Pod
apiVersion: v1
metadata:
      name: containerport-practical
spec:
      containers:
            - name: httpd-container
              image: httpd
              command:  ["/bin/bash", "-c", "while true; do echo Hello-Bhupinder; sleep 5 ; done"]
              ports:
                    - containerPort: 80
              env:
                    - name: myname
                      value: vishalk17
```

==========

```
File  Edit  View  Search  Terminal  Help
vishal@vishal-HP-245-G8:~/kubernetes$ nano container_port.yml
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f container_port.yml
pod/containerport-practical created
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                      READY   STATUS              RESTARTS   AGE
containerport-practical   0/1     ContainerCreating   0          16s
enviroments               1/1     Running             0          13m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                      READY   STATUS              RESTARTS   AGE
containerport-practical   0/1     ContainerCreating   0          24s
enviroments               1/1     Running             0          13m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl describe pods containerport-practical
Name:             containerport-practical
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.101
Start Time:       Thu, 06 Jul 2023 15:44:52 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.244.0.13
IPs:
  IP:  10.244.0.13
Containers:
  httpd-container:
    Container ID:  docker://69bfc0f2e1b35fe8cf02ffce51668d4c9aa563b2ec0a07c82b75b7e6ea78dc9e
```

```
  PodScheduled      True
Volumes:
  kube-api-access-hfbmn:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  47s   default-scheduler  Successfully assigned default/containerport-practical to minikube
  Normal  Pulling    46s   kubelet            Pulling image "httpd"
  Normal  Pulled     21s   kubelet            Successfully pulled image "httpd" in 25.850162031s (25.850195834s including waiting)
  Normal  Created    20s   kubelet            Created container httpd-container
  Normal  Started    20s   kubelet            Started container httpd-container
vishal@vishal-HP-245-G8:~/kubernetes$
```

==========

Labels & Selectors

- Labels are the mechanism you use to organise kubernetes objects.
- A label is a key-value pair without any predefined meaning that can be attached to the objects.
- Labels are similar to tags in AWS or git where you use a name to quick reference.
- So you are free to choose labels as you need it to refer to an environment which I used for dev or testing or production refer to a product group like department A, department B.
- Multiple Labels can be added to a single object.

There are two methods for writing a label

1. Declarative method: You can write a label inside your manifest file./ defined in manifest.yml

2. Imperative labels : giving existing pod using commands

```
kind: Pod
apiVersion: v1
metadata:
      name: using-labels
      labels:
            env: dev
            application: web-vishalk17
spec:
      containers:
            - name: c00
              image: ubuntu
              command:  ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep 5 ; done"]
              env:
                    - name: myname
                      value: vishalk17
```

==========

```
File  Edit  View  Search  Terminal  Help
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ vim using-labels.yml
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f using-labels.yml
pod/using-labels created
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods --show-labels
NAME                      READY   STATUS    RESTARTS   AGE   LABELS
containerport-practical   1/1     Running   0          29m   <none>
enviroments               1/1     Running   0          43m   <none>
using-labels              1/1     Running   0          46s   application=web-vishalk17,env=dev
vishal@vishal-HP-245-G8:~/kubernetes$ vim using-labels.yml
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get nodes --show-labels
NAME       STATUS   ROLES           AGE   VERSION   LABELS
minikube   Ready    control-plane   21h   v1.26.3   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernet
es.io/hostname=minikube,kubernetes.io/os=linux,minikube.k8s.io/commit=08896fd1dc362c097c925146c4a0d0dac715ace0,minikube.k8s.io/name=minikube,mi
nikube.k8s.io/primary=true,minikube.k8s.io/updated_at=2023_07_05T18_22_07_0700,minikube.k8s.io/version=v1.30.1,node-role.kubernetes.io/control-
plane=,node.kubernetes.io/exclude-from-external-load-balancers=
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -l env=dev
NAME           READY   STATUS    RESTARTS   AGE
using-labels   1/1     Running   0          7m13s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -l env!=dev
NAME                      READY   STATUS    RESTARTS   AGE
containerport-practical   1/1     Running   0          36m
enviroments               1/1     Running   0          49m
vishal@vishal-HP-245-G8:~/kubernetes$
```

```
File  Edit  View  Search  Terminal  Help
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods --show-labels
NAME                      READY   STATUS    RESTARTS   AGE     LABELS
containerport-practical   1/1     Running   0          38m     <none>
enviroments               1/1     Running   0          52m     <none>
using-labels              1/1     Running   0          9m48s   application=web-vishalk17,env=dev
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl label pods containerport-practical practical=labelling
pod/containerport-practical labeled
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods --show-labels
NAME                      READY   STATUS    RESTARTS   AGE   LABELS
containerport-practical   1/1     Running   0          38m   practical=labelling
enviroments               1/1     Running   0          52m   <none>
using-labels              1/1     Running   0          10m   application=web-vishalk17,env=dev
vishal@vishal-HP-245-G8:~/kubernetes$
```

==========

```
File  Edit  View  Search  Terminal  Help
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods --show-labels
NAME                     READY   STATUS    RESTARTS   AGE   LABELS
containerport-practical  1/1     Running   0          43m   practical=labelling
enviroments              1/1     Running   0          57m   <none>
using-labels             1/1     Running   0          14m   application=web-vishalk17,env=dev
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -l practical=labelling
NAME                     READY   STATUS    RESTARTS   AGE
containerport-practical  1/1     Running   0          44m
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -l practical!=labelling
NAME           READY   STATUS    RESTARTS   AGE
enviroments    1/1     Running   0          57m
using-labels   1/1     Running   0          15m
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
containerport-practical  1/1     Running   0          44m
enviroments              1/1     Running   0          58m
using-labels             1/1     Running   0          15m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl delete pods -l practical=labelling
pod "containerport-practical" deleted

vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
enviroments    1/1     Running   0          59m
using-labels   1/1     Running   0          16m
vishal@vishal-HP-245-G8:~/kubernetes$
```

kubectl get pods --show-labels ... if labels on pod to show
kubectl get nodes --show-labels ... if labels on nodes to show

kubectl label pods pod-name key=value… give label to the existing pod

kubectl get pods -l env=development .. list pods matching a label, -l :label

kubectl get pods -l env!=development .. give a list , where development label is not present

kubectl delete pod -l env=development … delete pod using label

==========

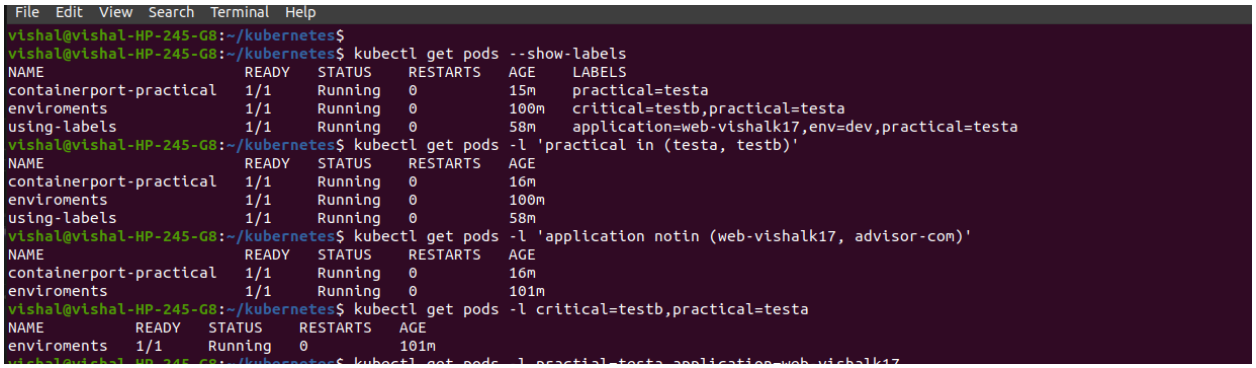
Labels- Selector :

- Unlike name/ Uid , label do not provide uniqueness, as in general we can expect many objects to carry the same label.
- Once labels are attached to an object, we would need filters to narrow down and these are called as label selectors.
- The api currently supports two types of selectors : equality based and set based.
- A label selector can be made of multiple requirements which are comma-separated.

Equality Based: [=, !=]

Set Based : (in, notin, and exists)

- env in (production, dev)
- env notin (team1, team2)
- it also support set-based selectors i.e., match multiple values.



kubectl get pods -l 'key in (value1,value34)' .. get a list of pods where development and testing label

kubectl get pods -l 'env notin (development,testing)' .. get a list of pods where development and testing label is not

kubectl get pods -l class=pods,myname=vishal … get a list of pods match multiple labels

kubectl delete pods -l 'env in (development)' … delete pod wherever development label is

==========

Node-selector

- one use case for selecting labels is to constrain the set of nodes onto which a pod can schedule i.e., you can tell a pod to only be able to run on particular node.
- Generally such constraints are unnecessary as the scheduler will automatically do a reasonable placement, but on certain circumstances we might need it.
- We can use labels to tag nodes.
- First we give label to the tag nodes, then use node selector to the pod configuration.

vi select-node-base-on-label.yml

```
kind: Pod
apiVersion: v1
metadata:
      name: select-node-base-on-label
      labels:
            env: testing
            application: web-google
spec:
      containers:
            - name: web-google-app
              image: ubuntu
              command:  ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep 5 ; done"]
              env:
                  - name: myname
                    value: vishalk17
      nodeSelector:
            hardware: t2.medium #should match with label given to the node
```

```
File  Edit  View  Search  Terminal  Help
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ vi select-node-base-on-label.yml
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f select-node-base-on-label.yml
pod/select-node-base-on-label created
vishal@vishal-HP-245-G8:~/kubernetes$ vi select-node-base-on-label.yml
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
containerport-practical     1/1     Running   0          46m
enviroments                 1/1     Running   0          131m
select-node-base-on-label   0/1     Pending   0          2m1s
using-labels                1/1     Running   0          88m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl describe pods select-node-base-on-label
Name:             select-node-base-on-label
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           application=web-google
                  env=testing
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  web-google-app:
```

==========

```
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-cklv7 (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-cklv7:
    Type:                     Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:   3607
    ConfigMapName:            kube-root-ca.crt
    ConfigMapOptional:        <nil>
    DownwardAPI:              true
QoS Class:                    BestEffort
Node-Selectors:               hardware=t2.medium
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age    From               Message
  ----     ------            ----   ----               -------
  Warning  FailedScheduling  2m53s  default-scheduler  0/1 nodes are available: 1 node(s) didn't match Pod's node affinity/selector. preemption
: 0/1 nodes are available: 1 Preemption is not helpful for scheduling..
vishal@vishal-HP-245-G8:~/kubernetes$
```

status showing pending because t2.medium yet to label to the node. Kubectl doesn't found any thus status is pending

- Giving label to the master node

```
File  Edit  View  Search  Terminal  Help
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   23h   v1.26.3
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get nodes --show-labels
NAME       STATUS   ROLES           AGE   VERSION   LABELS
minikube   Ready    control-plane   23h   v1.26.3   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernet
es.io/hostname=minikube,kubernetes.io/os=linux,minikube.k8s.io/commit=08896fd1dc362c097c925146c4a0d0dac715ace0,minikube.k8s.io/name=minikube,mi
nikube.k8s.io/primary=true,minikube.k8s.io/updated_at=2023_07_05T18_22_07_0700,minikube.k8s.io/version=v1.30.1,node-role.kubernetes.io/control-
plane=,node.kubernetes.io/exclude-from-external-load-balancers=
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl label node minikube hardware=t2.medium
node/minikube labeled
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get nodes --show-labels
NAME       STATUS   ROLES           AGE   VERSION   LABELS
minikube   Ready    control-plane   23h   v1.26.3   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,hardware=t2.medium,kubernetes.io/
arch=amd64,kubernetes.io/hostname=minikube,kubernetes.io/os=linux,minikube.k8s.io/commit=08896fd1dc362c097c925146c4a0d0dac715ace0,minikube.k8s.
io/name=minikube,minikube.k8s.io/primary=true,minikube.k8s.io/updated_at=2023_07_05T18_22_07_0700,minikube.k8s.io/version=v1.30.1,node-role.kub
ernetes.io/control-plane=,node.kubernetes.io/exclude-from-external-load-balancers=
vishal@vishal-HP-245-G8:~/kubernetes$
```

Pod is running on master nod i.e, minikube

kubectl get nodes .. list of all available nodes
kubectl get nodes --show-labels .. list nodes that has labeles
kubectl label node minikube hardware=t2.medium .. labelling to the node, here node is minikube
kubectl get pods -o wide .. if you want to see where exactly pod is running,ip address

==========

Status changed from pending to running

```
File  Edit  View  Search  Terminal  Help
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   23h   v1.26.3
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get nodes --show-labels
NAME       STATUS   ROLES           AGE   VERSION   LABELS
minikube   Ready    control-plane   23h   v1.26.3   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernet
es.io/hostname=minikube,kubernetes.io/os=linux,minikube.k8s.io/commit=08896fd1dc362c097c925146c4a0d0dac715ace0,minikube.k8s.io/name=minikube,mi
nikube.k8s.io/primary=true,minikube.k8s.io/updated_at=2023_07_05T18_22_07_0700,minikube.k8s.io/version=v1.30.1,node-role.kubernetes.io/control-
plane=,node.kubernetes.io/exclude-from-external-load-balancers=
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl label node minikube hardware=t2.medium
node/minikube labeled
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get nodes --show-labels
NAME       STATUS   ROLES           AGE   VERSION   LABELS
minikube   Ready    control-plane   23h   v1.26.3   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,hardware=t2.medium,kubernetes.io/
arch=amd64,kubernetes.io/hostname=minikube,kubernetes.io/os=linux,minikube.k8s.io/commit=08896fd1dc362c097c925146c4a0d0dac715ace0,minikube.k8s.
io/name=minikube,minikube.k8s.io/primary=true,minikube.k8s.io/updated_at=2023_07_05T18_22_07_0700,minikube.k8s.io/version=v1.30.1,node-role.kub
ernetes.io/control-plane=,node.kubernetes.io/exclude-from-external-load-balancers=
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS   AGE    IP            NODE       NOMINATED NODE   READINESS GATES
containerport-practical     1/1     Running   0          62m    10.244.0.17   minikube   <none>           <none>
enviroments                 1/1     Running   0          146m   10.244.0.12   minikube   <none>           <none>
select-node-base-on-label   1/1     Running   0          17m    10.244.0.18   minikube   <none>           <none>
using-labels                1/1     Running   0          104m   10.244.0.16   minikube   <none>           <none>
vishal@vishal-HP-245-G8:~/kubernetes$
```

```
File  Edit  View  Search  Terminal  Help
select-node-base-on-label   1/1     Running   0          17m    10.244.0.18   minikube   <none>           <none>
using-labels                1/1     Running   0          104m   10.244.0.16   minikube   <none>           <none>
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl describe pods select-node-base-on-label
Name:             select-node-base-on-label
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.101
Start Time:       Thu, 06 Jul 2023 17:53:47 +0530
Labels:           application=web-google
                  env=testing
Annotations:      <none>
Status:           Running
IP:               10.244.0.18
IPs:
  IP:  10.244.0.18
Containers:
  web-google-app:
    Container ID:  docker://bcc5ea05a4a9cdd99cdb921ed6771a0e080465209d6094dbdcdb99b62c868e8a
    Image:         ubuntu
    Image ID:      docker-pullable://ubuntu@sha256:0bced47fffa3361afa981854fcabcd4577cd43cebbb808cea2b1f33a3dd7f508
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/bash
      -c
      while true; do echo Hello-vishalk17; sleep 5 ; done
    State:          Running
      Started:      Thu, 06 Jul 2023 17:53:50 +0530
```

```
  kube-api-access-cklv7:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              hardware=t2.medium
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age    From               Message
  ----     ------            ----   ----               -------
  Warning  FailedScheduling  21m    default-scheduler  0/1 nodes are available: 1 node(s) didn't match Pod's node affinity/selector. preemption
: 0/1 nodes are available: 1 Preemption is not helpful for scheduling..
  Warning  FailedScheduling  15m    default-scheduler  0/1 nodes are available: 1 node(s) didn't match Pod's node affinity/selector. preemption
: 0/1 nodes are available: 1 Preemption is not helpful for scheduling..
  Normal   Scheduled         7m43s  default-scheduler  Successfully assigned default/select-node-base-on-label to minikube
  Normal   Pulling           7m43s  kubelet            Pulling image "ubuntu"
  Normal   Pulled            7m40s  kubelet            Successfully pulled image "ubuntu" in 2.751747379s (2.751757839s including waiting)
  Normal   Created           7m40s  kubelet            Created container web-google-app
  Normal   Started           7m40s  kubelet            Started container web-google-app
vishal@vishal-HP-245-G8:~/kubernetes$
```

==========

Scaling and replication:

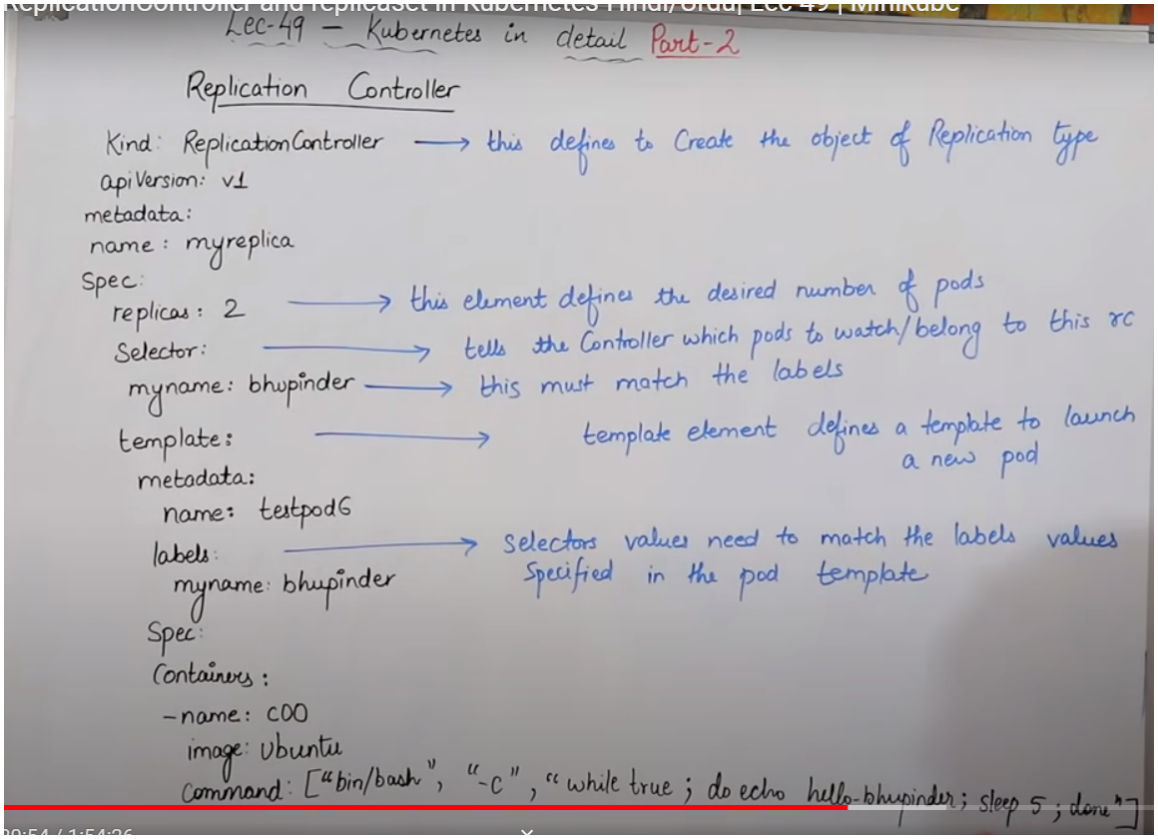
- Kubernetes was designed to orchestrate multiple containers and replication.
- Need for multiple containers/replication helps us with these.

  - **Reliability** : by having multiple versions of an application, you prevent problems if one or more fails.
  - **Load Balancing:** having multiple versions of a containers enables you to easily send traffic to different instances to prevent overloading of a single instances or node.
  - **Scaling** : when load does become too much for the number of existing instances, Kubernetes enables you to easily scale up your application, adding additional instances as needed.
  - **Rolling Updates**: updates to a service by replacing pods one by one.

Replication Controller :

- A replication controller is an object that enables you to create multiple pods, so that the pods always exist. If you set replicas=2 and one pod is crashed/failed or terminated then it will automatically create a new pod and maintain the state of the pod equal to 2.
- If a pod created using RC will be automatically replaced if they does crash, failed or terminated.
- RC is recommended if you just want to make sure 1 pod is running, even after system reboots.
- RC should be minimum set to 1 or 2(so that a copy is always present) and you can extend the number of it also.

==========




```
kind: ReplicationController
apiVersion: v1
metadata:
      name: replication-web-app-vishalk17
spec:
      replicas: 2
      selector:
            application: web-vishalk17
      template:
            metadata:
                  name: web-vishalk17-pod
                  labels:
                        application: web-vishalk17
            spec:
                  containers:
                        - name: ubuntu-container
                          image: ubuntu
                          command:  ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep 5 ; done"]
                          env:
                                - name: myname
                                  value: vishalk17
```

==========

vishal@vishal-HP-245-G8: ~/kubernetes

```
kind: ReplicationController
apiVersion: v1
metadata:
        name: replication-web-app-vishalk17
spec:
        replicas: 2
        selector:
              application: web-vishalk17
        template:
                metadata:
                        name: web-vishalk17-pod
                        labels:
                                application: web-vishalk17
                spec:
                        containers:
                                - name: ubuntu-container
                                  image: ubuntu
                                  command:  ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep 5 ; done"]
                                  env:
                                          - name: myname
                                            value: vishalk17
~
```

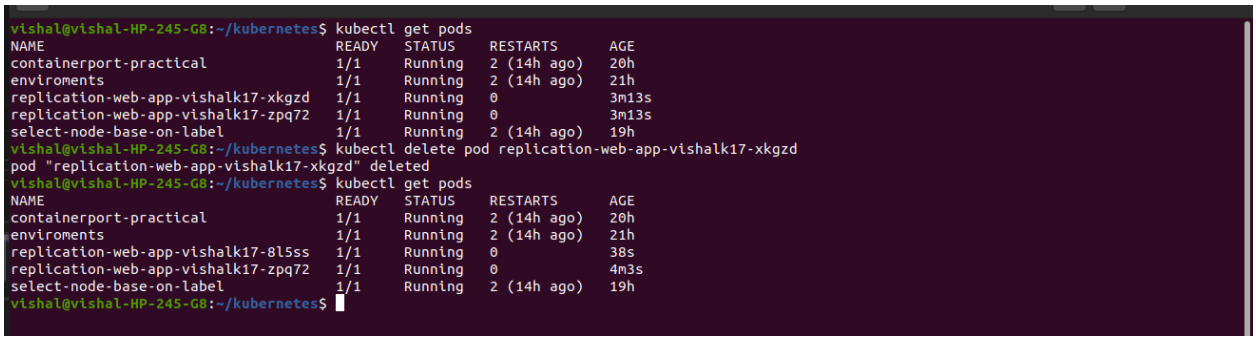
```
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f replication.yml
replicationcontroller/replication-web-app-vishalk17 created
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get rc
NAME                          DESIRED   CURRENT   READY   AGE
replication-web-app-vishalk17   2         2         0       3s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS              RESTARTS      AGE
containerport-practical             1/1     Running             2 (14h ago)   20h
enviroments                         1/1     Running             2 (14h ago)   21h
replication-web-app-vishalk17-xkgzd   0/1     ContainerCreating   0             6s
replication-web-app-vishalk17-zpq72   0/1     ContainerCreating   0             6s
select-node-base-on-label           1/1     Running             2 (14h ago)   19h
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
containerport-practical             1/1     Running   2 (14h ago)   20h
enviroments                         1/1     Running   2 (14h ago)   21h
replication-web-app-vishalk17-xkgzd   1/1     Running   0             64s
replication-web-app-vishalk17-zpq72   1/1     Running   0             64s
select-node-base-on-label           1/1     Running   2 (14h ago)   19h
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -l application=web-vishalk17
NAME                                READY   STATUS    RESTARTS   AGE
replication-web-app-vishalk17-xkgzd   1/1     Running   0          74s
replication-web-app-vishalk17-zpq72   1/1     Running   0          74s

vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl describe rc replication-web-app-vishalk17
Name:         replication-web-app-vishalk17
Namespace:    default
Selector:     application=web-vishalk17
Labels:       application=web-vishalk17
Annotations:  <none>
Replicas:     2 current / 2 desired
Pods Status:  2 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  application=web-vishalk17
  Containers:
   ubuntu-container:
```

```
    Image:      ubuntu
    Port:       <none>
    Host Port:  <none>
    Command:
      /bin/bash
      -c
      while true; do echo Hello-vishalk17; sleep 5 ; done
    Environment:
      myname:  vishalk17
    Mounts:    <none>
  Volumes:     <none>
Events:
  Type    Reason            Age   From                    Message
  ----    ------            ----  ----                    -------
  Normal  SuccessfulCreate  92s   replication-controller  Created pod: replication-web-app-vishalk17-zpq72
  Normal  SuccessfulCreate  92s   replication-controller  Created pod: replication-web-app-vishalk17-xkgzd
vishal@vishal-HP-245-G8:~/kubernetes$
```

==========

If pod deleted then replication will create new one .

By writing kubectl get pods, you will get 2 pods because replicas:2. If you delete any of the pod by writing kubectl delete pod pod-name, another one will be automatically created.



kubectl get rc …. will give you the details of replication controller like name, desired, current etc.
kubectl describe rc replica-name … will show you the brief details of replica set.
kubectl delete -f file-name.yml … it will delete replication pod, You can't delete the replica pod, because it wlll automatically create a new pod.

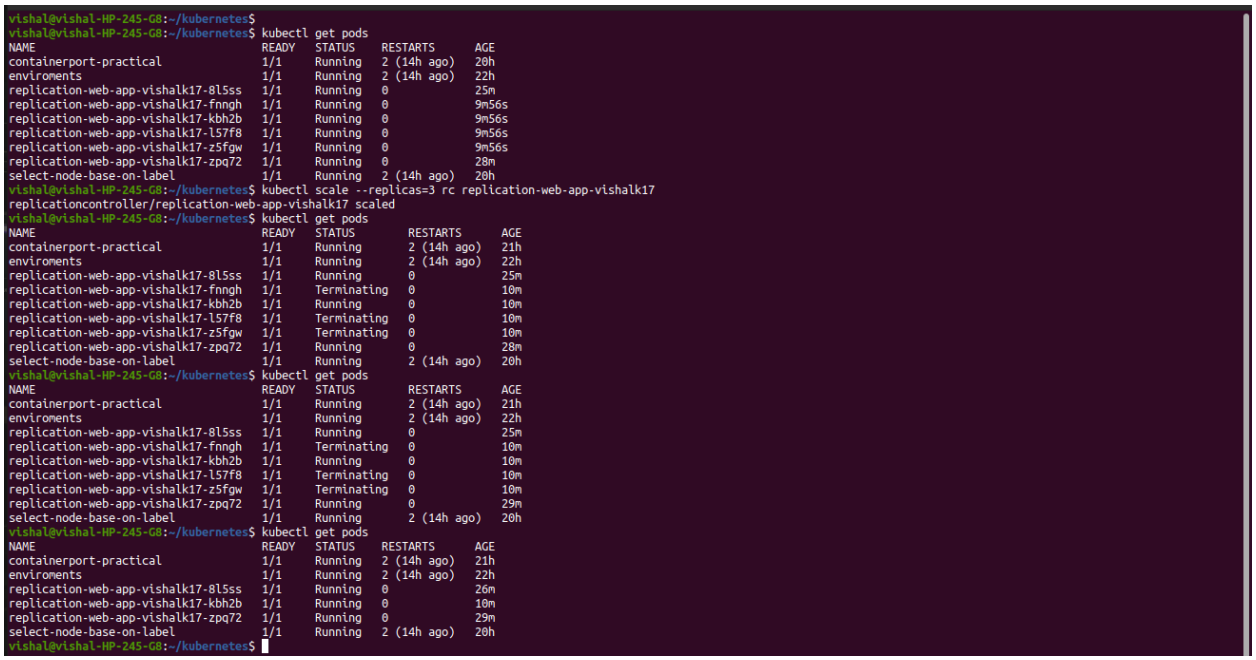
==========

Scaling up :

```
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods --show-labels
NAME                                READY   STATUS    RESTARTS      AGE   LABELS
containerport-practical             1/1     Running   2 (14h ago)   20h   practical=testa
enviroments                         1/1     Running   2 (14h ago)   22h   critical=testb,practical=testa
replication-web-app-vishalk17-8l5ss   1/1     Running   0             12m   application=web-vishalk17
replication-web-app-vishalk17-zpq72   1/1     Running   0             16m   application=web-vishalk17
select-node-base-on-label           1/1     Running   2 (14h ago)   20h   application=web-google,env=testing
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get rc
NAME                          DESIRED   CURRENT   READY   AGE
replication-web-app-vishalk17   2         2         2       16m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
containerport-practical             1/1     Running   2 (14h ago)   20h
enviroments                         1/1     Running   2 (14h ago)   22h
replication-web-app-vishalk17-8l5ss   1/1     Running   0             13m
replication-web-app-vishalk17-zpq72   1/1     Running   0             16m
select-node-base-on-label           1/1     Running   2 (14h ago)   20h
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl scale --replicas=6 rc replication-web-app-vishalk17
replicationcontroller/replication-web-app-vishalk17 scaled
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS              RESTARTS      AGE
containerport-practical             1/1     Running             2 (14h ago)   20h
enviroments                         1/1     Running             2 (14h ago)   22h
replication-web-app-vishalk17-8l5ss   1/1     Running             0             15m
replication-web-app-vishalk17-fnngh   0/1     ContainerCreating   0             8s
replication-web-app-vishalk17-kbh2b   1/1     Running             0             8s
replication-web-app-vishalk17-l57f8   0/1     ContainerCreating   0             8s
replication-web-app-vishalk17-z5fgw   0/1     ContainerCreating   0             8s
replication-web-app-vishalk17-zpq72   1/1     Running             0             18m
select-node-base-on-label           1/1     Running             2 (14h ago)   20h
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
containerport-practical             1/1     Running   2 (14h ago)   20h
enviroments                         1/1     Running   2 (14h ago)   22h
replication-web-app-vishalk17-8l5ss   1/1     Running   0             15m
replication-web-app-vishalk17-fnngh   1/1     Running   0             22s
replication-web-app-vishalk17-kbh2b   1/1     Running   0             22s
replication-web-app-vishalk17-l57f8   1/1     Running   0             22s
replication-web-app-vishalk17-z5fgw   1/1     Running   0             22s
replication-web-app-vishalk17-zpq72   1/1     Running   0             19m
select-node-base-on-label           1/1     Running   2 (14h ago)   20h
vishal@vishal-HP-245-G8:~/kubernetes$
```

==========

Scaling Down :



```
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
containerport-practical             1/1     Running   2 (14h ago)   20h
enviroments                         1/1     Running   2 (14h ago)   22h
replication-web-app-vishalk17-8l5ss   1/1     Running   0             25m
replication-web-app-vishalk17-fnngh   1/1     Running   0             9m56s
replication-web-app-vishalk17-kbh2b   1/1     Running   0             9m56s
replication-web-app-vishalk17-l57f8   1/1     Running   0             9m56s
replication-web-app-vishalk17-z5fgw   1/1     Running   0             9m56s
replication-web-app-vishalk17-zpq72   1/1     Running   0             28m
select-node-base-on-label           1/1     Running   2 (14h ago)   20h
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl scale --replicas=3 rc replication-web-app-vishalk17
replicationcontroller/replication-web-app-vishalk17 scaled
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS        RESTARTS      AGE
containerport-practical             1/1     Running       2 (14h ago)   21h
enviroments                         1/1     Running       2 (14h ago)   22h
replication-web-app-vishalk17-8l5ss   1/1     Running       0             25m
replication-web-app-vishalk17-fnngh   1/1     Terminating   0             10m
replication-web-app-vishalk17-kbh2b   1/1     Running       0             10m
replication-web-app-vishalk17-l57f8   1/1     Terminating   0             10m
replication-web-app-vishalk17-z5fgw   1/1     Terminating   0             10m
replication-web-app-vishalk17-zpq72   1/1     Running       0             28m
select-node-base-on-label           1/1     Running       2 (14h ago)   20h
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS        RESTARTS      AGE
containerport-practical             1/1     Running       2 (14h ago)   21h
enviroments                         1/1     Running       2 (14h ago)   22h
replication-web-app-vishalk17-8l5ss   1/1     Running       0             25m
replication-web-app-vishalk17-fnngh   1/1     Terminating   0             10m
replication-web-app-vishalk17-kbh2b   1/1     Running       0             10m
replication-web-app-vishalk17-l57f8   1/1     Terminating   0             10m
replication-web-app-vishalk17-z5fgw   1/1     Terminating   0             10m
replication-web-app-vishalk17-zpq72   1/1     Running       0             29m
select-node-base-on-label           1/1     Running       2 (14h ago)   20h
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                READY   STATUS    RESTARTS      AGE
containerport-practical             1/1     Running   2 (14h ago)   21h
enviroments                         1/1     Running   2 (14h ago)   22h
replication-web-app-vishalk17-8l5ss   1/1     Running   0             26m
replication-web-app-vishalk17-kbh2b   1/1     Running   0             10m
replication-web-app-vishalk17-zpq72   1/1     Running   0             29m
select-node-base-on-label           1/1     Running   2 (14h ago)   20h
vishal@vishal-HP-245-G8:~/kubernetes$
```

kubectl get pods –show-labels .. get list of pods using labels

kubectl scale --replicas=6 rc -l application=web-vishalk17 .. -l : label , 8 replicas scale , replication controller (rc)
replica scale up and down by changing the number

kubectl scale --replicas=3 rc replication-web-app-vishalk17 .. scaling UP OR DOWN using rc name

==========

Replica set:

- Replica set is the advanced version of the replication controller.
- The replication controller only supports equality-based selector whereas the replica set supports both equality-based and set-based selectors i.e. filtering according to the set of values.
- ReplicaSet rather than the Replication Controller is used by other objects like deployment.

```
kind: ReplicaSet
apiVersion: apps/v1
metadata:
        name: use-replica-set-web-app-vishalk17
spec:
        replicas: 2
        selector:
                matchExpressions:
                        - {key: application, operator: In, values: [web-vishalk17, web-google, ubuntu-app]}
                        - {key: env, operator: NotIn, values: [dev]}
        template:
                metadata:
                        name: web-vishalk17-pod
                        labels:
                                application: web-vishalk17
                spec:
                        containers:
                                - name: ubuntu-container
                                  image: ubuntu
                                  command:  ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep 5 ;
done"]
                                  env:
                                          - name: myname
                                            value: vishalk17
```

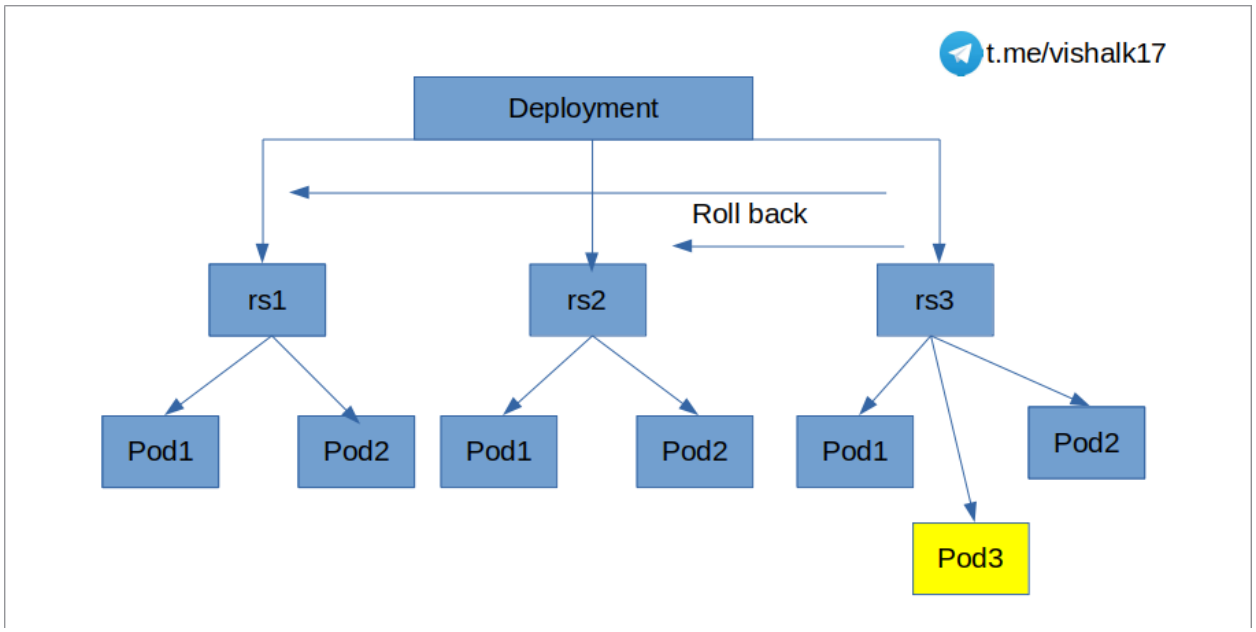
==========

```
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f replica-set.yml
replicaset.apps/use-replica-set-web-app-vishalk17 created
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                      READY   STATUS    RESTARTS      AGE
containerport-practical                   1/1     Running   2 (16h ago)   22h
enviroments                               1/1     Running   2 (16h ago)   23h
use-replica-set-web-app-vishalk17-bbhwg   1/1     Running   0             7s
use-replica-set-web-app-vishalk17-zkpzd   1/1     Running   0             7s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get rc
No resources found in default namespace.
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get rs
NAME                                DESIRED   CURRENT   READY   AGE
use-replica-set-web-app-vishalk17   2         2         2       16s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods --show-labels
NAME                                      READY   STATUS    RESTARTS      AGE   LABELS
containerport-practical                   1/1     Running   2 (16h ago)   22h   practical=testa
enviroments                               1/1     Running   2 (16h ago)   23h   critical=testb,practical=testa
use-replica-set-web-app-vishalk17-bbhwg   1/1     Running   0             35s   application=web-vishalk17
use-replica-set-web-app-vishalk17-zkpzd   1/1     Running   0             35s   application=web-vishalk17
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl describe rs use-replica-set-web-app-vishalk17
Name:         use-replica-set-web-app-vishalk17
Namespace:    default
Selector:     application in (ubuntu-app,web-google,web-vishalk17),env notin (dev)
Labels:       <none>
Annotations:  <none>
Replicas:     2 current / 2 desired
Pods Status:  2 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  application=web-vishalk17
  Containers:
   ubuntu-container:
    Image:      ubuntu
    Port:       <none>
    Host Port:  <none>
    Command:
      /bin/bash
      -c
      while true; do echo Hello-vishalk17; sleep 5 ; done
    Environment:
      myname:  vishalk17
    Mounts:    <none>
  Volumes:     <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
```

```
      -c
      while true; do echo Hello-vishalk17; sleep 5 ; done
    Environment:
      myname:  vishalk17
    Mounts:    <none>
  Volumes:     <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  114s  replicaset-controller  Created pod: use-replica-set-web-app-vishalk17-bbhwg
  Normal  SuccessfulCreate  114s  replicaset-controller  Created pod: use-replica-set-web-app-vishalk17-zkpzd
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -o wide
NAME                                      READY   STATUS    RESTARTS      AGE    IP            NODE       NOMINATED NODE   READINESS GATES
containerport-practical                   1/1     Running   2 (16h ago)   22h    10.244.0.26   minikube   <none>           <none>
enviroments                               1/1     Running   2 (16h ago)   23h    10.244.0.24   minikube   <none>           <none>
use-replica-set-web-app-vishalk17-bbhwg   1/1     Running   0             2m3s   10.244.0.47   minikube   <none>           <none>
use-replica-set-web-app-vishalk17-zkpzd   1/1     Running   0             2m3s   10.244.0.46   minikube   <none>           <none>
vishal@vishal-HP-245-G8:~/kubernetes$ vi replica-set.yml
vishal@vishal-HP-245-G8:~/kubernetes$
```

==========

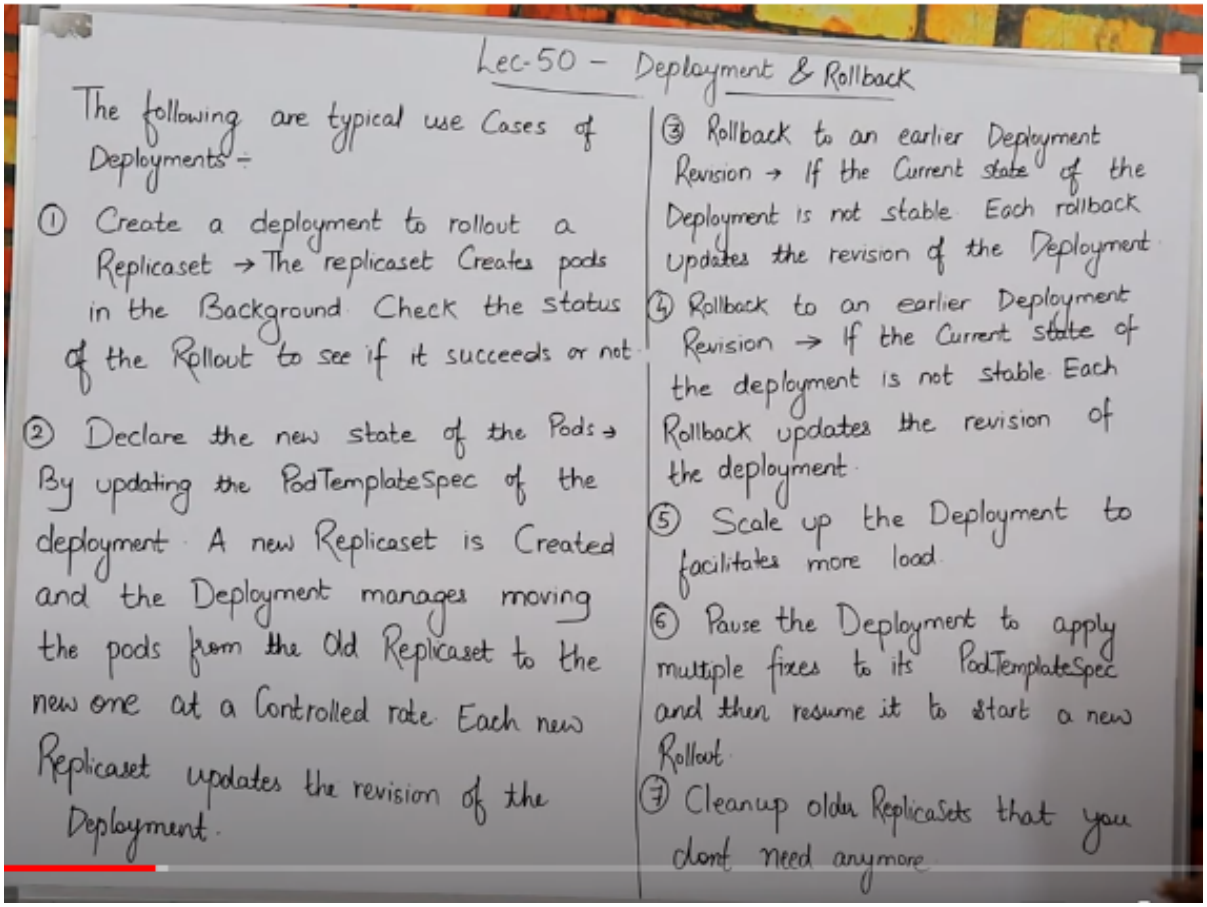
Deployment & Rollback



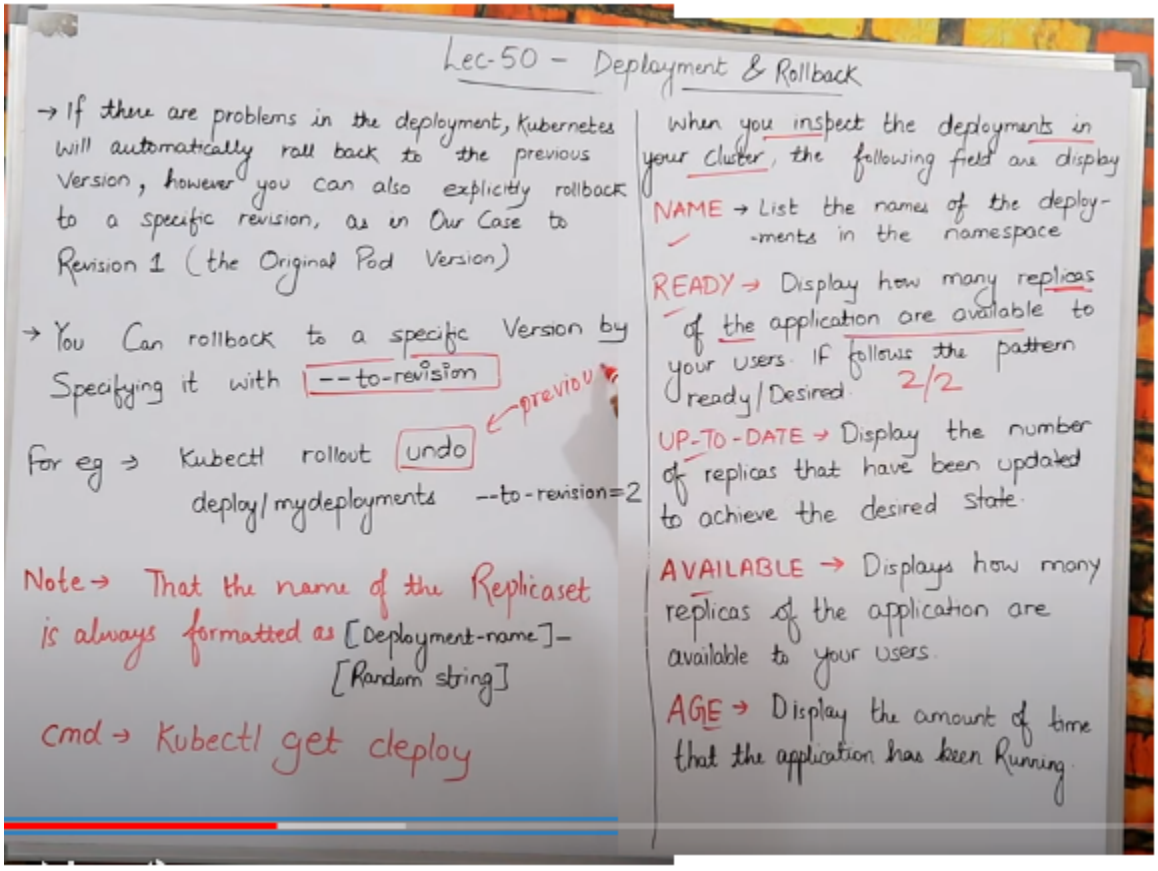
- Replication control and replica set is not able to do updates and rollback apps in the cluster.
- A deployment works as a supervisor for pods, giving you fine-grained control over how and when a new pod is rollout , updated or rolled back to the previous state.
- When using deployment object, we first define the state of an app then K8s cluster schedules an app onto specific individual nodes.
- A deployment provides declarative updates for pods and replicaset. Deployment will send a request to the replicaset and replicaset will implement the functionality on pod.
- K8s monitors if the node goes down or pod is deleted then the deployment controller replaces it.
- This provides a Self-healing Mechanism to address machine failure or maintenance.

Note: If the deployment rolled back from the version 3 replicaset to the version 2 replicaset, then the version 3 replicaset pods will be present in the version 2 replicaset. Although the code is of version 2 but the pods are of version 3.

==========


Lec-50 – Deployment & Rollback
The following are typical use Cases of Deployments :-
① 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
② Declare the new state of the Pods → By updating the PodTemplateSpec of the deployment A new Replicaset is Created and the Deployment manages moving the pods from the Old Replicaset to the new one at a Controlled rate Each new Replicaset updates the revision of the Deployment.
③ Rollback to an earlier Deployment Revision → If the Current state of the Deployment is not stable Each rollback updates the revision of the Deployment
④ Rollback to an earlier Deployment Revision → If the Current state of the deployment is not stable Each Rollback updates the revision of the deployment
⑤ Scale up the Deployment to facilitates more load
⑥ Pause the Deployment to apply multiple fixes to its PodTemplateSpec and then resume it to start a new Rollout
⑦ Cleanup older ReplicaSets that you dont need anymore

==========



Deployment failure :

Your deployment may get stuck trying to deploy its newest ReplicaSet without ever completing. This can occur due to some following factors.

- **Insufficient Quota** (Insufficient space in node)
- **Readiness probe failures** (If the pods in the new ReplicaSet are not ready to be readied, the deployment will fail. This can happen if the pods are not able to start up properly, or if they are not able to connect to the necessary resources)
- **Image pull errors**
- **Insufficient permission** (No permission to fetch)
- **Limit ranges (** If the deployment exceeds the limits for the node, the deployment will fail. This can happen if the deployment is using too much memory, CPU, or storage)
- **Application runtime misconfiguration** (App didn't run)

==========

The readiness of a pod can be checked using the `kubectl get pod` command. The output of this command will show the readiness status of the pod. The readiness status can be one of the following:

- **Ready:** The pod is ready to receive traffic.
- **NotReady:** The pod is not ready to receive traffic.
- **Unknown:** The readiness status of the pod is unknown.

```
kind: Deployment
apiVersion: apps/v1
metadata:
        name: deployment-web-app-vishalk17
spec:
        replicas: 2
        selector:
                matchExpressions:
                        - {key: application, operator: In, values: [web-vishalk17, web-google, ubuntu-app]}
                        - {key: env, operator: NotIn, values: [dev]}
        template:
                metadata:
                        name: web-vishalk17-pod
                        labels:
                                application: web-vishalk17
                spec:
                        containers:
                                - name: ubuntu-container
                                  image: ubuntu
                                  command:  ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep 5 ;
done"]
                                  env:
                                          - name: myname
                                            value: vishalk17
```

==========

```
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f deployment.yml
deployment.apps/deployment-web-app-vishalk17 created
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get deploy
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-web-app-vishalk17   2/2     2            2           13s
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                           READY   STATUS    RESTARTS   AGE
deployment-web-app-vishalk17-7d9d8b6bc6-6gh8k   1/1     Running   0          41s
deployment-web-app-vishalk17-7d9d8b6bc6-fg278   1/1     Running   0          41s
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl describe deployment deployment-web-app-vishalk17
Name:                   deployment-web-app-vishalk17
Namespace:              default
CreationTimestamp:      Fri, 07 Jul 2023 17:31:39 +0530
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               application in (ubuntu-app,web-google,web-vishalk17),env notin (dev)
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  application=web-vishalk17
  Containers:
   ubuntu-container:
    Image:      ubuntu
    Port:       <none>
    Host Port:  <none>
    Command:
      /bin/bash
      -c
      while true; do echo Hello-vishalk17; sleep 5 ; done
    Environment:
      myname:  vishalk17
    Mounts:    <none>
  Volumes:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   deployment-web-app-vishalk17-7d9d8b6bc6 (2/2 replicas created)
```

==========

```
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   deployment-web-app-vishalk17-7d9d8b6bc6 (2/2 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  63s   deployment-controller  Scaled up replica set deployment-web-app-vishalk17-7d9d8b6bc6 to 2
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl logs -f deployment-web-app-vishalk17-7d9d8b6bc6-6gh8k
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
```

```
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get deploy
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-web-app-vishalk17   2/2     2            2           33m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get rs
NAME                                    DESIRED   CURRENT   READY   AGE
deployment-web-app-vishalk17-7d9d8b6bc6   2         2         2       33m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS   AGE
deployment-web-app-vishalk17-7d9d8b6bc6-6gh8k   1/1     Running   0          33m
deployment-web-app-vishalk17-7d9d8b6bc6-fg278   1/1     Running   0          33m
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl logs -f deployment-web-app-vishalk17-7d9d8b6bc6-6gh8k
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
```

```
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl scale --replicas=4 deployment deployment-web-app-vishalk17
deployment.apps/deployment-web-app-vishalk17 scaled
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get deploy
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment-web-app-vishalk17   3/4     4            3           36m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get rs
NAME                                    DESIRED   CURRENT   READY   AGE
deployment-web-app-vishalk17-7d9d8b6bc6   4         4         4       36m
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS   AGE
deployment-web-app-vishalk17-7d9d8b6bc6-4h8z5   1/1     Running   0          14s
deployment-web-app-vishalk17-7d9d8b6bc6-6gh8k   1/1     Running   0          36m
deployment-web-app-vishalk17-7d9d8b6bc6-fg278   1/1     Running   0          36m
deployment-web-app-vishalk17-7d9d8b6bc6-z4vk5   1/1     Running   0          14s
vishal@vishal-HP-245-G8:~/kubernetes$
```

==========

kubectl get deployment .. to check deployment was created or not

kubectl describe deployment deployment-web-app-vishalk17 .. detail about deployment : creates rs & pods

kubectl get rs ... check replication set

kubectl scale --replicas=4 deployment deployment-web-app-vishalk17 .. to scale up or down

kubectl logs -f podname .. to check what is running inside container

==========

If some change done in *.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
    name: deployment-web-app-vishalk17
spec:
    replicas: 4
    selector:
        matchExpressions:
            - {key: application, operator: In, values: [web-vishalk17, web-google, ubuntu-app]}
            - {key: env, operator: NotIn, values: [dev]}
    template:
        metadata:
            name: web-vishalk17-pod
            labels:
                application: web-vishalk17
        spec:
            containers:
                - name: ubuntu-container
                  image: ubuntu
                  command:  ["/bin/bash", "-c", "while true; do echo I-love-u-vishal; sleep 5 ; done"]
                  env:
                      - name: myname
                        value: vishalk17
```

what I have changed ??

now print : I-love-u-vishal Previously it was printing: `Hello-vishalk17`

`replicas: 2 to replica 4`

==========

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
        name: deployment-web-app-vishalk17
spec:
        replicas: 4
        selector:
                matchExpressions:
                        - {key: application, operator: In, values: [web-vishalk17, web-google, ubuntu-app]}
                        - {key: env, operator: NotIn, values: [dev]}
        template:
                metadata:
                        name: web-vishalk17-pod
                        labels:
                                application: web-vishalk17
                spec:
                        containers:
                                - name: ubuntu-container
                                  image: ubuntu
                                  command:  ["/bin/bash", "-c", "while true; do echo I-love-u-vishal; sleep 5 ; done"]
                                  env:
                                          - name: myname
                                            value: vishalk17
```

```
vishal@vishal-HP-245-G8:~/kubernetes$ ls
container_port.yml  deployment.yml  hello.yml  replica-set.yml  replication-controller.yml  select-node-base-on-label.yml  using-labels.yml
vishal@vishal-HP-245-G8:~/kubernetes$ vi deployment.yml
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS   AGE
deployment-web-app-vishalk17-7d9d8b6bc6-94hfl   1/1     Running   0          2m49s
deployment-web-app-vishalk17-7d9d8b6bc6-krk69   1/1     Running   0          2m54s
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl logs -f deployment-web-app-vishalk17-7d9d8b6bc6-krk69
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
```

```
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f deployment.yml
deployment.apps/deployment-web-app-vishalk17 configured
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                          READY   STATUS              RESTARTS   AGE
deployment-web-app-vishalk17-65b9487b79-42tgh   0/1     ContainerCreating   0          4s
deployment-web-app-vishalk17-65b9487b79-4lscf   0/1     ContainerCreating   0          4s
deployment-web-app-vishalk17-7d9d8b6bc6-94hfl   1/1     Running             0          6m15s
deployment-web-app-vishalk17-7d9d8b6bc6-krk69   1/1     Running             0          6m20s
deployment-web-app-vishalk17-7d9d8b6bc6-t8dvx   0/1     Terminating         0          4s
deployment-web-app-vishalk17-7d9d8b6bc6-ttqh7   0/1     ContainerCreating   0          4s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                          READY   STATUS        RESTARTS   AGE
deployment-web-app-vishalk17-65b9487b79-42tgh   1/1     Running       0          29s
deployment-web-app-vishalk17-65b9487b79-4lscf   1/1     Running       0          29s
deployment-web-app-vishalk17-65b9487b79-cr2d9   1/1     Running       0          23s
deployment-web-app-vishalk17-65b9487b79-kk6zn   1/1     Running       0          15s
deployment-web-app-vishalk17-7d9d8b6bc6-94hfl   1/1     Terminating   0          6m40s
deployment-web-app-vishalk17-7d9d8b6bc6-krk69   1/1     Terminating   0          6m45s
deployment-web-app-vishalk17-7d9d8b6bc6-t8dvx   1/1     Terminating   0          29s
deployment-web-app-vishalk17-7d9d8b6bc6-ttqh7   1/1     Terminating   0          29s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                          READY   STATUS        RESTARTS   AGE
deployment-web-app-vishalk17-65b9487b79-42tgh   1/1     Running       0          48s
deployment-web-app-vishalk17-65b9487b79-4lscf   1/1     Running       0          48s
deployment-web-app-vishalk17-65b9487b79-cr2d9   1/1     Running       0          42s
deployment-web-app-vishalk17-65b9487b79-kk6zn   1/1     Running       0          34s
deployment-web-app-vishalk17-7d9d8b6bc6-krk69   1/1     Terminating   0          7m4s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl logs -f deployment-web-app-vishalk17-65b9487b79-42tgh
I-love-u-vishal
I-love-u-vishal
I-love-u-vishal
I-love-u-vishal
```

==========

```
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl rollout undo deploy/deployment-web-app-vishalk17 --to-revision=3
deployment.apps/deployment-web-app-vishalk17 rolled back
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl rollout status deployment deployment-web-app-vishalk17
deployment "deployment-web-app-vishalk17" successfully rolled out
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl rollout history deployment deployment-web-app-vishalk17
deployment.apps/deployment-web-app-vishalk17
REVISION  CHANGE-CAUSE
4         <none>
5         <none>

vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME                                          READY   STATUS        RESTARTS   AGE
deployment-web-app-vishalk17-65b9487b79-42tgh   1/1     Terminating   0          7m46s
deployment-web-app-vishalk17-65b9487b79-4lscf   1/1     Terminating   0          7m46s
deployment-web-app-vishalk17-65b9487b79-cr2d9   1/1     Terminating   0          7m40s
deployment-web-app-vishalk17-7d9d8b6bc6-cf9px   1/1     Running       0          24s
deployment-web-app-vishalk17-7d9d8b6bc6-nq4gd   1/1     Running       0          32s
deployment-web-app-vishalk17-7d9d8b6bc6-t2nnh   1/1     Running       0          26s
deployment-web-app-vishalk17-7d9d8b6bc6-tdfrm   1/1     Running       0          31s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl logs -f deployment-web-app-vishalk17-7d9d8b6bc6-cf9px
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
Hello-vishalk17
^Z
[2]+  Stopped                 kubectl logs -f deployment-web-app-vishalk17-7d9d8b6bc6-cf9px
vishal@vishal-HP-245-G8:~/kubernetes$
```

Note : If rollback to previous version then number of replication will be same as we have defined in *.yml file.

********** Deployment & Roll back **********

kubectl get deployment or deploy.. to check deployment was created or not
kubectl describe deployment deployment-web-app-vishalk17 .. detail about deployment : creates rs & pods
kubectl get rs ... check replication set

kubectl scale --replicas=1 deployment deployment-web-app-vishalk17 .. to scale up or down
kubectl logs -f podname .. to check what is running inside container

kubectl rollout status deployment deployment-web-app-vishalk17 .. to check status
kubectl rollout history deployment deployment-web-app-vishalk17 .. check revision

kubectl rollout undo deploy/deployment-web-app-vishalk17 .. rollback to previous version
kubectl rollout undo deploy/deployment-web-app-vishalk17 --to-revision=2 .. rollback to specific version

==========

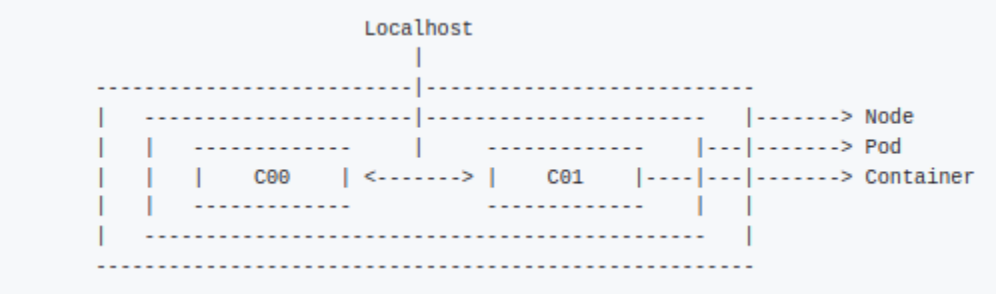
** Kubernetes Networking / services / nodeport & volumes **

Kubernetes Networking

- Containers within a pod use networking to communicate via loopback.
- Cluster networking provides communication between different pods.
- The service lets you expose an application running in pods to be reachable from outside cluster/browser or internet.
- You can also use service to publish services only for consumption inside your cluster.

Container to container communication

- Container to container communication on the same pod happens through localhost within the containers.




```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-Bilal; sleep 5; done"]
    - name: c01
      image: httpd
      ports:
        - containerPort: 80
```

==========

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Hello-vishalk17; sleep 5; done"]
    - name: c01
      image: httpd
      ports:
        - containerPort: 80
~
~
~
```

```
vishal@vishal-HP-245-G8:~/kubernetes$ vi network-cont-cont.yml
vishal@vishal-HP-245-G8:~/kubernetes$ vi network-cont-cont.yml
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f network-cont-cont.yml
pod/testpod created
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
testpod   2/2     Running   0          13s
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl describe pod testpod
Name:             testpod
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.101
Start Time:       Tue, 11 Jul 2023 12:05:39 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
```

```
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  37s   default-scheduler  Successfully assigned default/testpod to minikube
  Normal  Pulling    36s   kubelet            Pulling image "ubuntu"
  Normal  Pulled     34s   kubelet            Successfully pulled image "ubuntu" in 2.438399238s (2.438427892s including waiting)
  Normal  Created    34s   kubelet            Created container c00
  Normal  Started    34s   kubelet            Started container c00
  Normal  Pulling    34s   kubelet            Pulling image "httpd"
  Normal  Pulled     32s   kubelet            Successfully pulled image "httpd" in 1.940734932s (1.940752225s including waiting)
  Normal  Created    32s   kubelet            Created container c01
  Normal  Started    32s   kubelet            Started container c01
vishal@vishal-HP-245-G8:~/kubernetes$ ^C
vishal@vishal-HP-245-G8:~/kubernetes$
```

- After creating the pod, type `kubectl exec pod-name -it -c container-name -- /bin/bash`. It will take you inside the container that is present in a pod.
- `apt update && apt install curl` will update and install the curl package inside your container.
- `curl localhost:80` will show you a message that it works. It will show you that two containers are communicating successfully.

==========

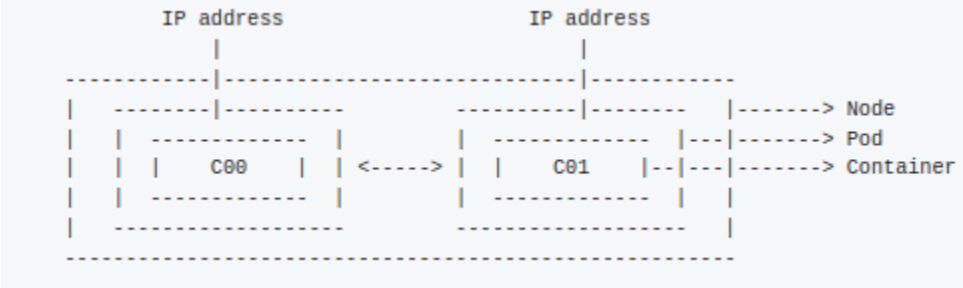
```
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl exec -it testpod -c c00 -- /bin/bash
root@testpod:/#
root@testpod:/# apt install curl -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package curl
root@testpod:/# apt update && apt install curl
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [53.6 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [693 kB]
```

```
Updating certificates in /etc/ssl/certs...
137 added, 0 removed; done.
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.10) ...
Setting up curl (7.81.0-1ubuntu1.10) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for ca-certificates (20230311ubuntu0.22.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
root@testpod:/# curl localhost:80
<html><body><h1>It works!</h1></body></html>
root@testpod:/#
```

==========

Pod to pod communication

- Pod to pod communication on the same worker node happens through ip address. container dont have ip address, pods has ip
    - If you provided a pod A ip address, you'll communicate with pod A and vice versa.
    - If container A wants to access container B in another pod, then to go inside the container A, give an ip address and it will redirect you to container B.
- By default, pod's ip address will not be accessible outside the node.



First file

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod1
spec:
  containers:
    - name: c01
      image: nginx
      ports:
        - containerPort: 80
```

Second file

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod2
spec:
  containers:
    - name: c02
      image: httpd
      ports:
        - containerPort: 80
```

==========

- After making pods, type `kubectl get pods -o wide` to take the ip address.
- `kubectl exec pod-name -it -c container-name -- /bin/bash` will take you inside the container that is present is in a pod.
- `apt update && apt install curl` will update and install curl in case if it is not installed in the pod terminal.
- `curl ip-address:80` will give you the details of that ip address. Type this command inside the pod terminal.

let check communication over ip

- accessing httpd from nginx

```
vishal@vishal-HP-245-G8:~/kubernetes$ vi network-pod-2-pod-file1.yml
vishal@vishal-HP-245-G8:~/kubernetes$ vi network-pod-2-pod-file2.yml
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f network-pod-2-pod-file1.yml
pod/testpod1 created
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl apply -f network-pod-2-pod-file2.yml
pod/testpod2 created
vishal@vishal-HP-245-G8:~/kubernetes$ kubctl get pods -o wide

Command 'kubctl' not found, did you mean:

  command 'kubectl' from snap kubectl (1.27.3)

See 'snap info <snapname>' for additional versions.

vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS GATES
testpod1   1/1     Running   0          36s   10.244.0.78   minikube   <none>           <none>
testpod2   1/1     Running   0          29s   10.244.0.79   minikube   <none>           <none>
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl exec -it pod testpod1 -c c01 -- /bin/bash
Error from server (NotFound): pods "pod" not found
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl exec -it testpod1 -c c01 -- /bin/bash
root@testpod1:/#
root@testpod1:/# apt-get update -y && apt install curl
Ign:1 http://deb.debian.org/debian bookworm InRelease
Ign:2 http://deb.debian.org/debian bookworm-updates InRelease
```

```
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.88.1-10).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@testpod1:/#
root@testpod1:/# curl 10.244.0.79:80
<html><body><h1>It works!</h1></body></html>
root@testpod1:/#
```

==========

- accessing nginx from httpd

```
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP            NODE       NOMINATED NODE   READINESS GATES
testpod1   1/1     Running   0          8m52s   10.244.0.78   minikube   <none>           <none>
testpod2   1/1     Running   0          8m45s   10.244.0.79   minikube   <none>           <none>
vishal@vishal-HP-245-G8:~/kubernetes$
vishal@vishal-HP-245-G8:~/kubernetes$ kubectl exec -it testpod2 -c c02 -- /bin/bash
root@testpod2:/usr/local/apache2#
root@testpod2:/usr/local/apache2# apt-get update && apt-get install curl -y
Get:1 http://deb.debian.org/debian bookworm InRelease [147 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [52.1 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8904 kB]
Get:5 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [47.3 kB]
Fetched 9199 kB in 6s (1657 kB/s)
Reading package lists... Done
```

```
root@testpod2:/usr/local/apache2#
root@testpod2:/usr/local/apache2# curl 10.244.0.78:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```

==========

Service object

- When using RC, pods are terminated and created during scaling or replication operations.
- When using deployments while updating the image version, the pods are terminated and new pods take the place of other pods.
- Pods are dynamic. i.e. They come and go on the k8s cluster and on any of the available nodes and it would be difficult to access the pods as the pods IP changes once it is recreated.

Problem

Each pod gets its own ip address, however in a deployment the set of pods running could be different in one moment from the set of pods running in another moment. This leads to a problem that if a pod & it's ip address is not permanent to its place and is changing everytime, how would another pod keep track of the ip address of the target pod?

Solution

- The solution is the service object that is a logical bridge b/w pods and the end users which provides virtual ip address(VIP) and with the help of VIP, the communication will happen even if the pod's ip addresses are changing.
- Each object will have its own virtual ip address.
- Service allows clients to reliably connect to the containers running in the pod using the VIP.
- The VIP is not an actual IP address connected to the network interface but the purpose is to forward traffic to one or more pods.
- Kube proxy will keep the mapping b/w the VIP and pod's ip address. If a pod new ip address is created, then kube proxy will map it with the VIP to build the connection.

==========

Problem

Although each pod has a unique ip address, these ip addresses are not exposed outside the cluster. They can only communicate inside the cluster.

Solution

- Services help to expose the VIP(that is mapped to the pods) and allow application to receive traffic outside the cluster (browser etc).
- You have to define labels on both pod side and services side that will select the specific pods(from thousands of pods) to put under a service.
- Creating a service will create an endpoint that will access the pods/application in it.
- Services can be exposed in four different ways by specifying a type in the service specification.
    a. Cluster IP
    b. NodePort
    c. LoadBalancer [same like aws ELB]
    d. Headless [create several endpoints that are used to produce dns records, each dns record bound to the pod]
- By default, service can only run b/w ports 30,000 - 32,767.
- The set of pods targeted by a service is usually determined by a selector.
- NodePort is an upper layer of the cluster ip and Load balancer is the upper layer of the nodeport and headless is top part.

==========

1. Cluster IP

- The nodes will be mapped with a VIP(it is fixed) so that the pod A from the node A can communicate with the pod B of the node B without needing to identify the specific ip addresses of each pods.
- Cluster IP exposes VIP to be reachable only from within the cluster. VIP can't be accessed outside the cluster.
- Mainly used to communicate between components of microservices.

deployment-httpd-cluster-ip.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: mydeployments
spec:
  replicas: 1
  selector:
    matchLabels:
      name: deployment
  template:
    metadata:
      name: testpod2
      labels:
        name: deployment
    spec:
      containers:
        - name: c00
          image: httpd
          ports:
            - containerPort: 80
```

==========

- After creating a pod, write `kubectl get pods -o wide` to see the ip address of the pod.
- `kubectl exec pod-name -it -c container-name -- /bin/bash` will take you inside the container that is present in a pod.
- `apt update && apt install curl` will update and install curl incase if it is not installed in the pod terminal.
- `curl ip-address:80` will give you the details of that ip address. Type this command inside the pod terminal.

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: mydeployments
spec:
  replicas: 1
  selector:
    matchLabels:
      name: deployment
  template:
    metadata:
      name: testpod2
      labels:
        name: deployment
    spec:
      containers:
        - name: c00
          image: httpd
          ports:
            - containerPort: 80
```

```
vishal@vishal-HP-245-G8:~/kubernetes/networking$
vishal@vishal-HP-245-G8:~/kubernetes/networking$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
mydeployments-867f6c86cc-mcbfh   1/1     Running   0          11m
vishal@vishal-HP-245-G8:~/kubernetes/networking$ kubectl exec -it mydeployments-867f6c86cc-mcbfh -- /bin/bash
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2#
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2#
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2# curl 10.244.0.82:80
<html><body><h1>It works!</h1></body></html>
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2#
```

==========

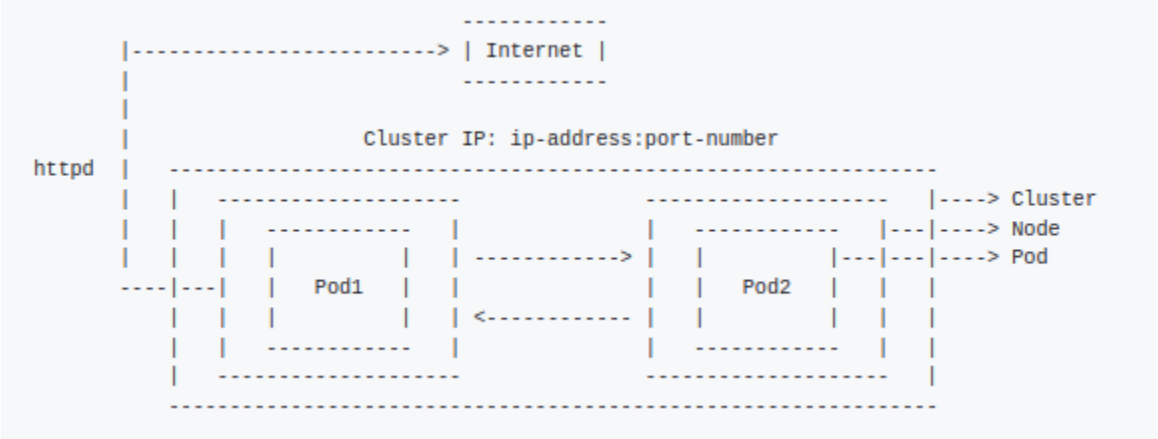
service-httpd-cluster-ip.yml

```
kind: Service                         #  --> Defines to create service type object
apiVersion: v1
metadata:
  name: demoservice
spec:
  ports:
    - port: 80                        #  --> Containers port exposed
      targetPort: 80                 #--> Pods port
  selector:
    name: deployment          # --> Apply this service to any pods which has the
specific label
  type: ClusterIP     #    --> Specifies the service type i.e. ClusterIP or NodePort
```

```
vishal@vishal-HP-245-G8:~/kubernetes/networking$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
mydeployments-867f6c86cc-mcbfh   1/1     Running   0          12m
vishal@vishal-HP-245-G8:~/kubernetes/networking$
vishal@vishal-HP-245-G8:~/kubernetes/networking$ ls
deployment-httpd-cluster-ip.yml  network-cont-cont.yml  network-pod-2-pod-file1.yml  network-pod-2-pod-file2.yml  service-httpd-cluster-ip.yml
vishal@vishal-HP-245-G8:~/kubernetes/networking$
vishal@vishal-HP-245-G8:~/kubernetes/networking$ vi service-httpd-cluster-ip.yml
vishal@vishal-HP-245-G8:~/kubernetes/networking$
vishal@vishal-HP-245-G8:~/kubernetes/networking$ kubectl apply -f service-httpd-cluster-ip.yml
service/demoservice created
vishal@vishal-HP-245-G8:~/kubernetes/networking$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
mydeployments-867f6c86cc-mcbfh   1/1     Running   0          13m
vishal@vishal-HP-245-G8:~/kubernetes/networking$
vishal@vishal-HP-245-G8:~/kubernetes/networking$ kubectl get service
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
demoservice   ClusterIP   10.103.237.202   <none>        80/TCP    15s
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP   5d20h
vishal@vishal-HP-245-G8:~/kubernetes/networking$
vishal@vishal-HP-245-G8:~/kubernetes/networking$ curl 10.103.237.202:80
curl: (28) Failed to connect to 10.103.237.202 port 80: Connection timed out
vishal@vishal-HP-245-G8:~/kubernetes/networking$ vi service-httpd-cluster-ip.yml
vishal@vishal-HP-245-G8:~/kubernetes/networking$ kubectl exec -it mydeployments-867f6c86cc-mcbfh -- /bin/bash
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2#
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2# curl 10.103.237.202:80
<html><body><h1>It works!</h1></body></html>
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2#
root@mydeployments-867f6c86cc-mcbfh:/usr/local/apache2# exit
exit
vishal@vishal-HP-245-G8:~/kubernetes/networking$
```

==========

2. NodePort



- Nodeport access a service from outside the cluster via internet.
- Attach a port number that is assigned to you with public DNS and that port number is attached to the virtual ip address and the VIP will contact the pod inside a node. As a result, the container inside pod will be shown to us via internet.
- First take the deployment.yml file from above and apply it in the kubectl. Then apply the service.yml file below.

deployment-httpd-node-ip.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: mydeployments
spec:
  replicas: 1
  selector:
    matchLabels:
      name: deployment
  template:
    metadata:
      name: testpod2
      labels:
        name: deployment
    spec:
      containers:
        - name: c00
          image: httpd
          ports:
            - containerPort: 80
```

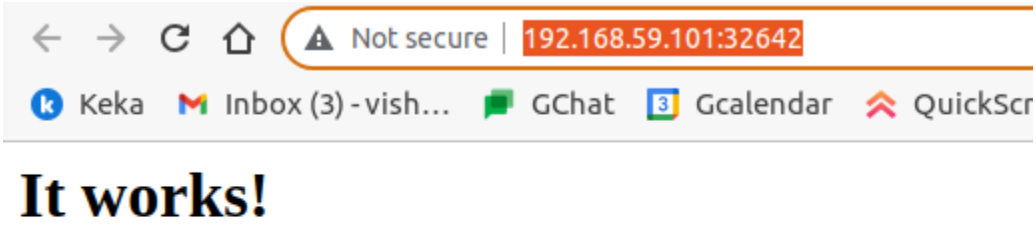
==========

```
kind: Service                       #     --> Defines to create service type object
apiVersion: v1
metadata:
  name: demoservice
spec:
  ports:
    - port: 80                      #     --> Containers port exposed
      targetPort: 80                #--> Pods port
  selector:
    name: deployment                #--> Apply this service to any pods which has the
specific label
  type: NodePort           #  --> Specifies the service type i.e. ClusterIP or NodePort
```

- NodePort will make the pod accessible to the internet or outside the cluster.
- After making a pod of this file, type kubectl get svc, you'll be provided a port number like this 80:<this-port-number>/TCP.
- kubectl describe svc demoservice will show you the details of the service.
- Browser will access the VIP that is attached to the port number. VIP will will access a pod and an application inside it.
- minikube ip will give the ip address that will be used in the browser to work on.
- After getting the minikube ip address and the port number from kubectl get svc, write http://<minikube-ip>:<this-port-number> in the browser. It will give you the success message.
- If you're using a cloud platform, you can take the DNS link and attach it with port-number to make it work.

```
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ ls
deployment-httpd-Node-Port.yml  service-httpd-NodePort.yml
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ kubectl apply -f deployment-httpd-Node-Port.yml
deployment.apps/mydeployments created
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ kubectl apply -f service-httpd-NodePort.yml
service/demoservice created
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
mydeployments-867f6c86cc-bt5dg   1/1     Running   0          15s
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ kubectl get svc
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
demoservice   NodePort    10.98.2.198   <none>        80:32642/TCP   23s
kubernetes    ClusterIP   10.96.0.1     <none>        443/TCP        5d21h
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ kubectl get node -o wide
NAME       STATUS   ROLES           AGE     VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE
minikube   Ready    control-plane   5d21h   v1.26.3   192.168.59.101   <none>        Buildroot 2021.02.12
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$  minikube service demoservice --url
http://192.168.59.101:32642
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ ^C
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$ curl http://192.168.59.101:32642
<html><body><h1>It works!</h1></body></html>
vishal@vishal-HP-245-G8:~/kubernetes/networking/NodePort$
```

==========



Note : If you are using Minikube, the process for accessing your httpd server from the outside world is slightly different. Minikube runs a single-node Kubernetes cluster locally, and it uses a different approach for exposing services.

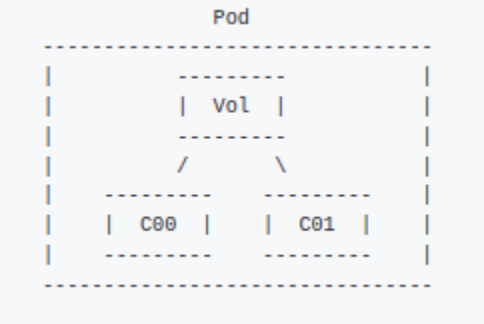
if not using minikube then you can directly access by

public-ip:port

```
kubectl get service       .. get list of service with ip
kubectl get nodes -o wide .. get nodes details and ip
minikube service service-name --url .. To get the URL to access your service
                                       [ only in case you are using minikube ]
```

==========

Volume :



- Containers are short lived in nature.
- All the data stored inside a container is deleted if the container crashes. However the kubelet will restart it with a clean state, which means that it will not have any of the old data.
- To overcome this problem, volume comes into picture and kubernetes uses it. A Volume in Kubernetes represents a directory with data that is accessible across multiple containers in a Pod.
- In kubernetes, a volume is attached to a pod and shared among the containers of that pod.
- The volume has the same life span as the pod and it will not be affected if the containers crashed. If new containers are created, then they will take the data from the volume that is already present.
- If the pod is crashed then the volume will also be crashed.

Volume Types

A volume decides the properties of the directory/pod like size, content etc. Some of the volume types in which you can store the data are.

- None-local such as EmptyDir or host path.
- File sharing type such as NFS.
- Cloud provider such as AWSElasticBlockStore, AzureDisk etc.
- Distributed filesystem types, e.g. glusterfs or cephfs.
- Special purpose types like secret, gitrepo.

==========

EmptyDir

- When a pod is newly created and assigned to a node then an emptydir volume is also created and exist as long as that pod is running on that node.
- Use emptydir if we want to share content b/w multiple containers on the same pod and not to the host machine.
- As the name says, it is initially empty.
- After the containers are created, they will be mounted/attached with same volume.
- When a pod from a node is deleted, the data in the emptydir will be deleted forever.
- A container crashing does not remove a pod from a node. The data in an emptydir volume will be safe if the container crashes.

```
vishal@vishal-HP-245-G8:~/kubernetes/volume$
vishal@vishal-HP-245-G8:~/kubernetes/volume$ vi empty-dir-volume.yml
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl apply -f empty-dir-volume.yml
pod/myvolemptydir created
vishal@vishal-HP-245-G8:~/kubernetes/volume$
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl get pods
NAME           READY   STATUS              RESTARTS   AGE
myvolemptydir  0/2     ContainerCreating   0          6s
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl get pods
NAME           READY   STATUS              RESTARTS   AGE
myvolemptydir  0/2     ContainerCreating   0          24s
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
myvolemptydir  2/2     Running   0          82s
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl exec -it myvolemptydir -c c1 -- /bin/bash
[root@myvolemptydir /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@myvolemptydir /]# cd /tmp
[root@myvolemptydir tmp]# ls
ks-script-4luisyla  ks-script-o23i7rc2  ks-script-x6ei4wuu  xchange
[root@myvolemptydir tmp]#
[root@myvolemptydir tmp]# cd xchange
[root@myvolemptydir xchange]# ls
[root@myvolemptydir xchange]# touch vish1 vish2
[root@myvolemptydir xchange]# ls
vish1  vish2
[root@myvolemptydir xchange]# exite
bash: exite: command not found
[root@myvolemptydir xchange]# exit
exit
command terminated with exit code 127
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl exec -it myvolemptydir -c c2 -- /bin/bash
[root@myvolemptydir /]#
[root@myvolemptydir /]# cd /tmp
[root@myvolemptydir tmp]# ls
data  ks-script-4luisyla  ks-script-o23i7rc2  ks-script-x6ei4wuu
[root@myvolemptydir tmp]# cd data
[root@myvolemptydir data]# ls
vish1  vish2
[root@myvolemptydir data]#
```

- After creating a pod, type `kubectl exec -it pod-name -c c1 -- /bin/bash`. It will take you inside the pod container 1 terminal.
- `cd tmp/xchange` will take you inside the xchange directory. Create any kind of file and write something in it.

==========

- Type `kubectl exec -it pod-name -c c2 -- /bin/bash`. It will take you inside the pod container 2 terminal.
- `cd tmp/data` will take you inside the data directory. If you type `ls`, it will show you the same file that was created in the first container.
- If you modified it, it will show you the changes in another container also

HostPath volume

- HostPath volume allows you to mount a directory from the host node's filesystem into a pod. This can be useful for scenarios where you need to access files or data from the host machine within a pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: myvolhostpath
spec:
  containers:
  - image: centos
    name: c01
    command: ["/bin/bash", "-c", "sleep 15000"]
    volumeMounts:
    - mountPath: /tmp/hostpath
      name: testvolume
  volumes:
  - name: testvolume
    hostPath:
      path: /home/vishal/vishalk17
```

==========

Note (for minikube user only): The HostPath volume in Minikube has some limitations when it comes to accessing files on the host machine. By default, Minikube runs inside a VM, and the host path you specify in the `hostPath` field refers to the filesystem of the VM, not the host machine itself.

```
vishal@vishal-HP-245-G8:~/kubernetes/volume$
vishal@vishal-HP-245-G8:~/kubernetes/volume$ vi hostpath-vol.yml
vishal@vishal-HP-245-G8:~/kubernetes/volume$
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl apply -f hostpath-vol.yml
pod/myvolhostpath created
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl get pod
NAME           READY   STATUS    RESTARTS   AGE
myvolhostpath   1/1     Running   0          29s
vishal@vishal-HP-245-G8:~/kubernetes/volume$ kubectl exec -it myvolhostpath -- /bin/bash
[root@myvolhostpath /]#
[root@myvolhostpath /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@myvolhostpath /]# cd /tmp/
[root@myvolhostpath tmp]# ls
hostpath  ks-script-4luisyla  ks-script-o23i7rc2  ks-script-x6ei4wuu
[root@myvolhostpath tmp]# cd hostpath/
[root@myvolhostpath hostpath]# ls
[root@myvolhostpath hostpath]#
[root@myvolhostpath hostpath]# echo "hello vishal, how are you ??"
hello vishal, how are you ??
[root@myvolhostpath hostpath]# echo "hello vishal, how are you ??" >> readme
[root@myvolhostpath hostpath]#
[root@myvolhostpath hostpath]# touch index.html
[root@myvolhostpath hostpath]# ls
index.html  readme
[root@myvolhostpath hostpath]# cat readme
hello vishal, how are you ??
[root@myvolhostpath hostpath]# ls
file1  file3  index.html  readme
[root@myvolhostpath hostpath]#
```

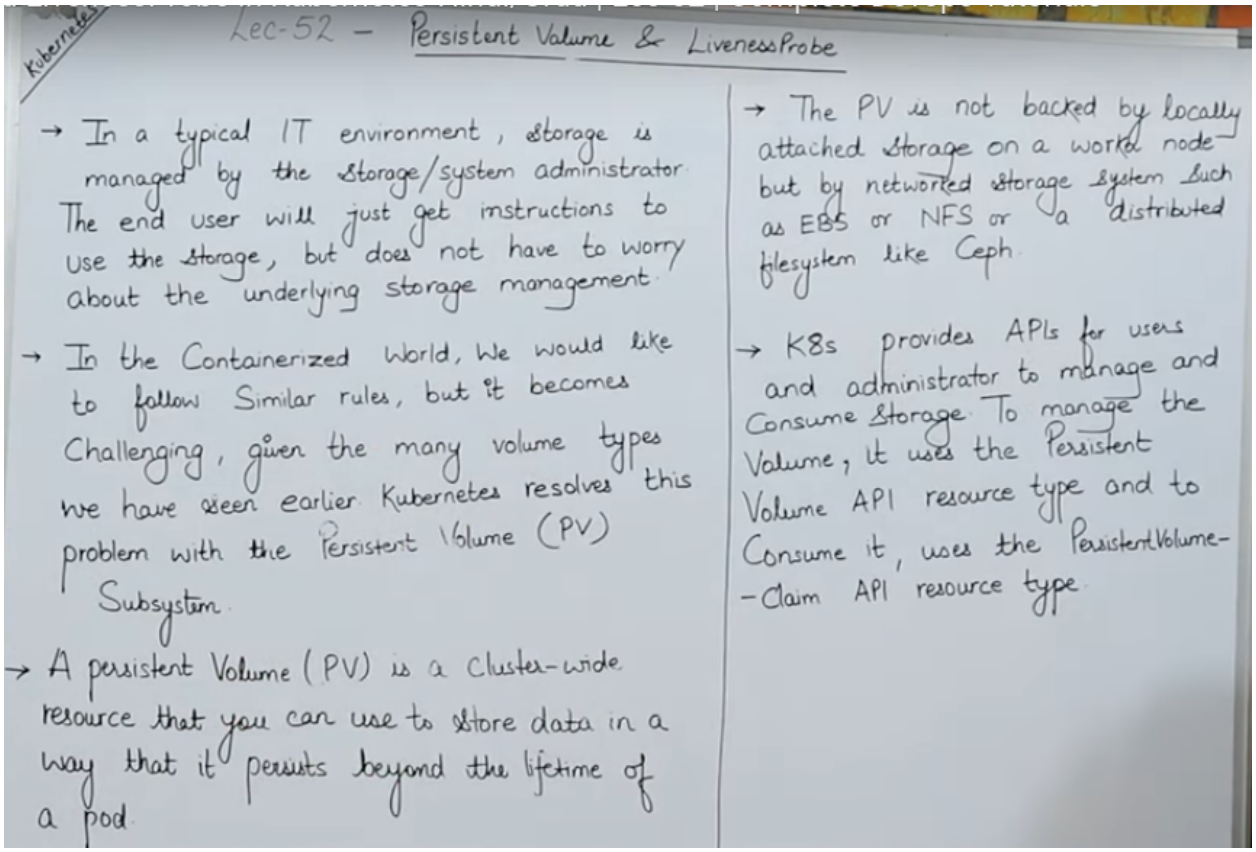
==========

```
vishal@vishal-HP-245-G8:~/kubernetes/volume$ minikube ssh
                         _             _
            _         _ ( )           ( )
  ___ ___  (_)  ___  (_)| |/')  _   _ | |_      __
/' _ ` _ `\| |/' _ `\| || , <  ( ) ( )| '_`\  /'__`\
| ( ) ( ) || || ( ) || || |\`\ | (_) || |_) )(  ___/
(_) (_) (_)(_)(_) (_)(_)(_) (_)`\___/'(_,__/'`\____)

$ ls
$ cd /
$ ls
CHANGELOG  boot  dev  home  hosthome  lib    libexec  media          mnt  proc  run   srv  tmp  var
bin        data  etc  host  init      lib64  linuxrc  minikube-host  opt  root  sbin  sys  usr  version.json
$ cd /home/
$ ls
docker  vishal
$ cd vishal/vishalk17/
$ ls
index.html  readme
$ cat readme
hello vishal, how are you ??
$
$ touch file1 file3
touch: cannot touch 'file1': Permission denied
touch: cannot touch 'file3': Permission denied
$ sudo touch file1 file3
$ ls
file1  file3  index.html  readme
$ exit
logout
vishal@vishal-HP-245-G8:~/kubernetes/volume$
```

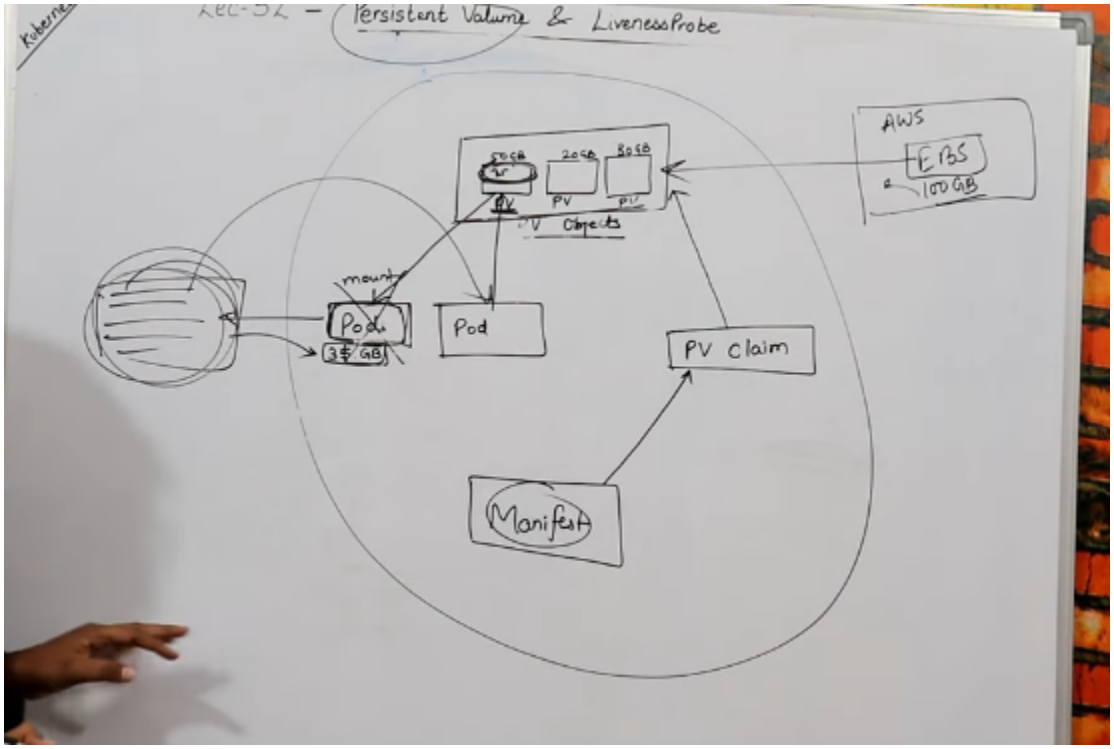
if you are not using minikube then kindly check hostpath of your host machine

==========

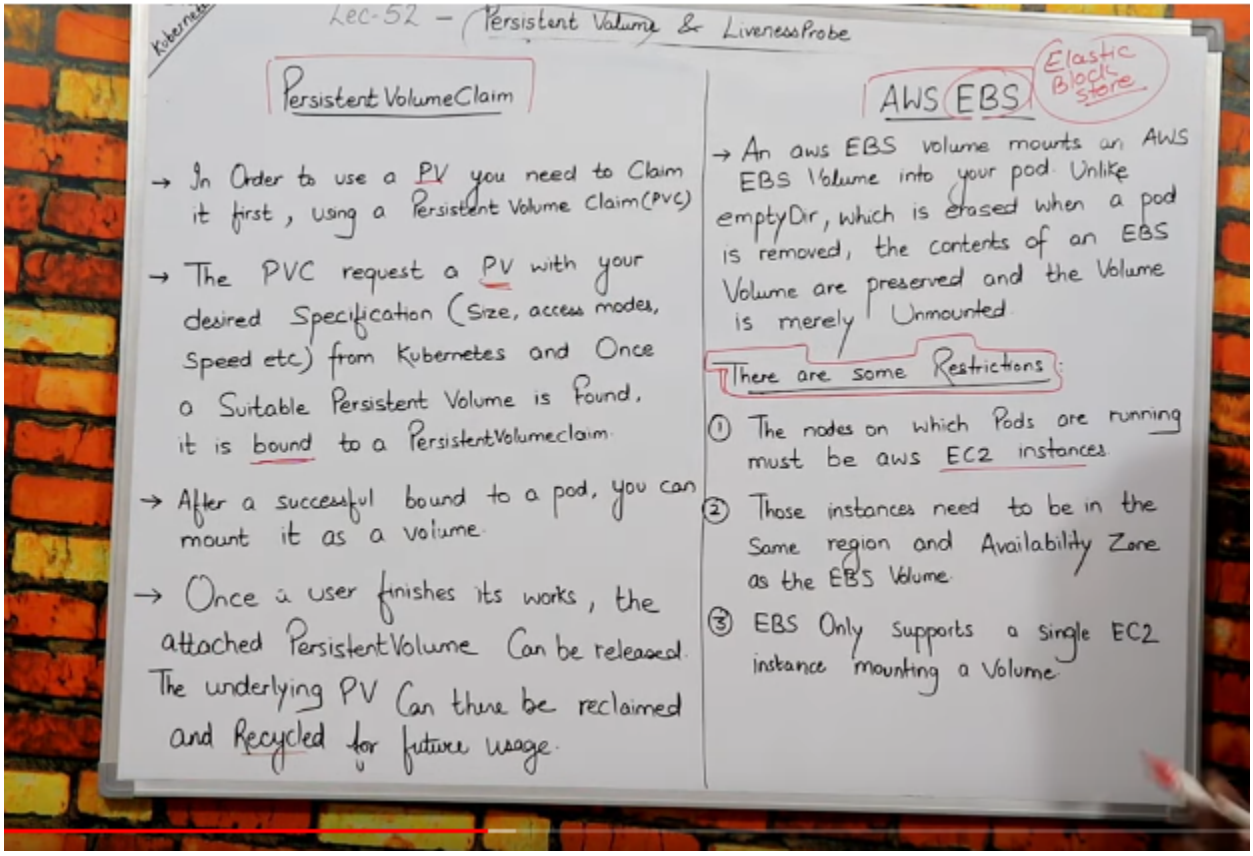
Persistent Volume and liveness probe



==========


Persistent Volume & LivenessProbe
AWS
EBS
100 GB
PV
PVC
PV Objects
mount
Pod
Pod
PV Claim
Manifest

==========



for aws volume

vi pv.yml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: myebsvol
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  awsElasticBlockStore:
    volumeID:            # YAHAN APNI EBS VOLUME ID DAALO
    fsType: ext
```

==========

— I m using minikube on my local laptop thus go with hostpath—

vi pv.yml

```
  apiVersion: v1
   kind: PersistentVolume
   metadata:      name: mylocalvol    spec:      capacity:
storage: 1Gi      accessModes:        - ReadWriteOnce
     persistentVolumeReclaimPolicy: Recycle
     hostPath:
       path: /vishal/volume1
```

---

vi pv-claim.yml .. for claiming volume from pv

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mylocalvolclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

==========

vi deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pvdeploy
spec:
  replicas: 1
  selector:       # tells the controller which pods to watch/belong to
    matchLabels:
     app: mypv
  template:
    metadata:
      labels:
        app: mypv
    spec:
      containers:
      - name: shell
        image: centos
        command: ["bin/bash", "-c", "sleep 10000"]
        volumeMounts:
        - name: mypd
          mountPath: "/tmp/persistent"
      volumes:
        - name: mypd
          persistentVolumeClaim:
            claimName: mylocalvolclaim
```

```
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl get pv
NAME                                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                      STORAGECLASS   REAS
ON   AGE
mylocalvol                          1Gi        RWO            Recycle          Available
     17m
pvc-336aa52f-6e7d-4fea-95fc-6c10d9ed5ad7   1Gi        RWO            Delete           Bound       default/mylocalvolclaim   standard
     8m34s
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl get pvc
NAME             STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mylocalvolclaim   Bound    pvc-336aa52f-6e7d-4fea-95fc-6c10d9ed5ad7   1Gi        RWO            standard       8m39s
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl get deploy
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
pvdeploy   1/1     1            1           83s
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
pvdeploy-55ccfbb8ff-gprx6   1/1     Running   0          88s
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$
```

==========

Let create files in `/tmp/persistent`

```
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
pvdeploy-55ccfbb8ff-gprx6   1/1     Running   0          3m23s
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl exec -it pvdeploy-55ccfbb8ff-gprx6 -- /bin/bash
[root@pvdeploy-55ccfbb8ff-gprx6 /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@pvdeploy-55ccfbb8ff-gprx6 /]# cd /tmp
[root@pvdeploy-55ccfbb8ff-gprx6 tmp]# ls
ks-script-4luisyla  ks-script-o23i7rc2  ks-script-x6ei4wuu  persistent
[root@pvdeploy-55ccfbb8ff-gprx6 tmp]# cd persistent/
[root@pvdeploy-55ccfbb8ff-gprx6 persistent]# ls
[root@pvdeploy-55ccfbb8ff-gprx6 persistent]# mkdir vishal
[root@pvdeploy-55ccfbb8ff-gprx6 persistent]# mkdir sdfa
[root@pvdeploy-55ccfbb8ff-gprx6 persistent]# touch adsf asdf asdf
[root@pvdeploy-55ccfbb8ff-gprx6 persistent]# ls
adsf  asdf  sdfa  vishal
[root@pvdeploy-55ccfbb8ff-gprx6 persistent]# exit
exit
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$
```

- delete existing pod
- so that new pod will create
- then check whethere those files existed or not

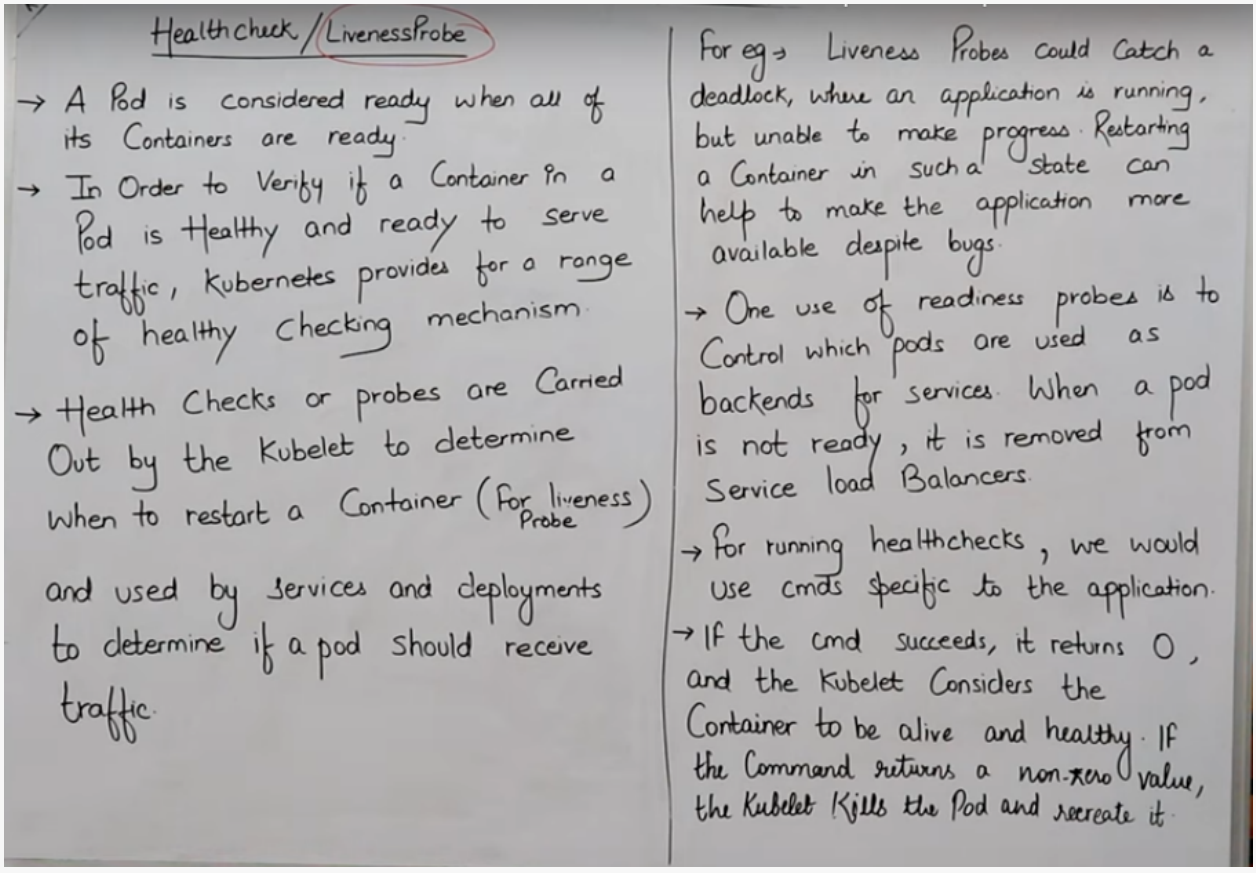
```
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
pvdeploy-55ccfbb8ff-gprx6   1/1     Running   0          5m16s
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl delete pod pvdeploy-55ccfbb8ff-gprx6
pod "pvdeploy-55ccfbb8ff-gprx6" deleted
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
pvdeploy-55ccfbb8ff-cc8mh   1/1     Running   0          76s
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$ kubectl exec -it pvdeploy-55ccfbb8ff-cc8mh -- /bin/bash
[root@pvdeploy-55ccfbb8ff-cc8mh /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@pvdeploy-55ccfbb8ff-cc8mh /]# cd /tmp
[root@pvdeploy-55ccfbb8ff-cc8mh tmp]# ls
ks-script-4luisyla  ks-script-o23i7rc2  ks-script-x6ei4wuu  persistent
[root@pvdeploy-55ccfbb8ff-cc8mh tmp]# cd persistent/
[root@pvdeploy-55ccfbb8ff-cc8mh persistent]# ls
adsf  asdf  sdfa  vishal
[root@pvdeploy-55ccfbb8ff-cc8mh persistent]# exit
exit
vishal@vishal-HP-245-G8:~/kubernetes/volume/persistent-vol$
```

- so, same files are available in new pod too
- so basically pv act as a common media to store the files
- so even after deleting pods persist vol stay and it is not the part of individual pod

```
kubectl get pv              .. get list of persistent volume
kubectl get pvc      .. get list of persistent volume claim
```

==========

Liveness Probe:



==========

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: mylivenessprobe
spec:
  containers:
  - name: liveness
    image: ubuntu
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 1000
    livenessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
      timeoutSeconds: 30
```

The liveness probe is configured to execute the `cat /tmp/healthy` command every 5 seconds. If the command fails, the pod is considered to be unhealthy and is removed from the cluster.

The `initialDelaySeconds` property specifies that the liveness probe should not be executed until the pod has been running for at least 5 seconds. This is to give the pod time to start up and become healthy.

The `periodSeconds` property specifies that the liveness probe should be executed every 5 seconds. This means that the pod will be checked for health every 5 seconds.

The `timeoutSeconds` property specifies that the liveness probe should timeout after 30 seconds. This means that if the `cat /tmp/healthy` command does not complete within 30 seconds, the pod will be considered to be unhealthy and will be removed from the cluster.

The liveness probe in this example is a simple way to ensure that your pods are always healthy. By configuring a liveness probe, you can help to prevent unhealthy pods from running in your cluster and affecting other pods.

Here is a breakdown of the different parts of the liveness probe definition:

- `exec`: This specifies that the liveness probe should execute a command.

==========

- `command`: This specifies the command that should be executed. In this case, the `cat /tmp/healthy` command is being executed.
- `initialDelaySeconds`: This specifies the number of seconds before the liveness probe should be executed.
- `periodSeconds`: This specifies the number of seconds between liveness probe executions.
- `timeoutSeconds`: This specifies the number of seconds before the liveness probe should timeout.

```
vishal@vishal-HP-245-G8:~/kubernetes/liveness-probe$
vishal@vishal-HP-245-G8:~/kubernetes/liveness-probe$ kubectl apply -f deploy-liveness-probe.yml
pod/mylivenessprobe created
vishal@vishal-HP-245-G8:~/kubernetes/liveness-probe$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
mylivenessprobe   1/1     Running   0          4s
vishal@vishal-HP-245-G8:~/kubernetes/liveness-probe$ kubectl exec -it mylivenessprobe -- /bin/bash
root@mylivenessprobe:/#
root@mylivenessprobe:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@mylivenessprobe:/# cd /tmp
root@mylivenessprobe:/tmp# ls
healthy
root@mylivenessprobe:/tmp# cat *
root@mylivenessprobe:/tmp#
root@mylivenessprobe:/tmp# echo $?
0
root@mylivenessprobe:/tmp# ls
healthy
root@mylivenessprobe:/tmp#
```

if i delete that file then liveness probe should has to be fail , lets check

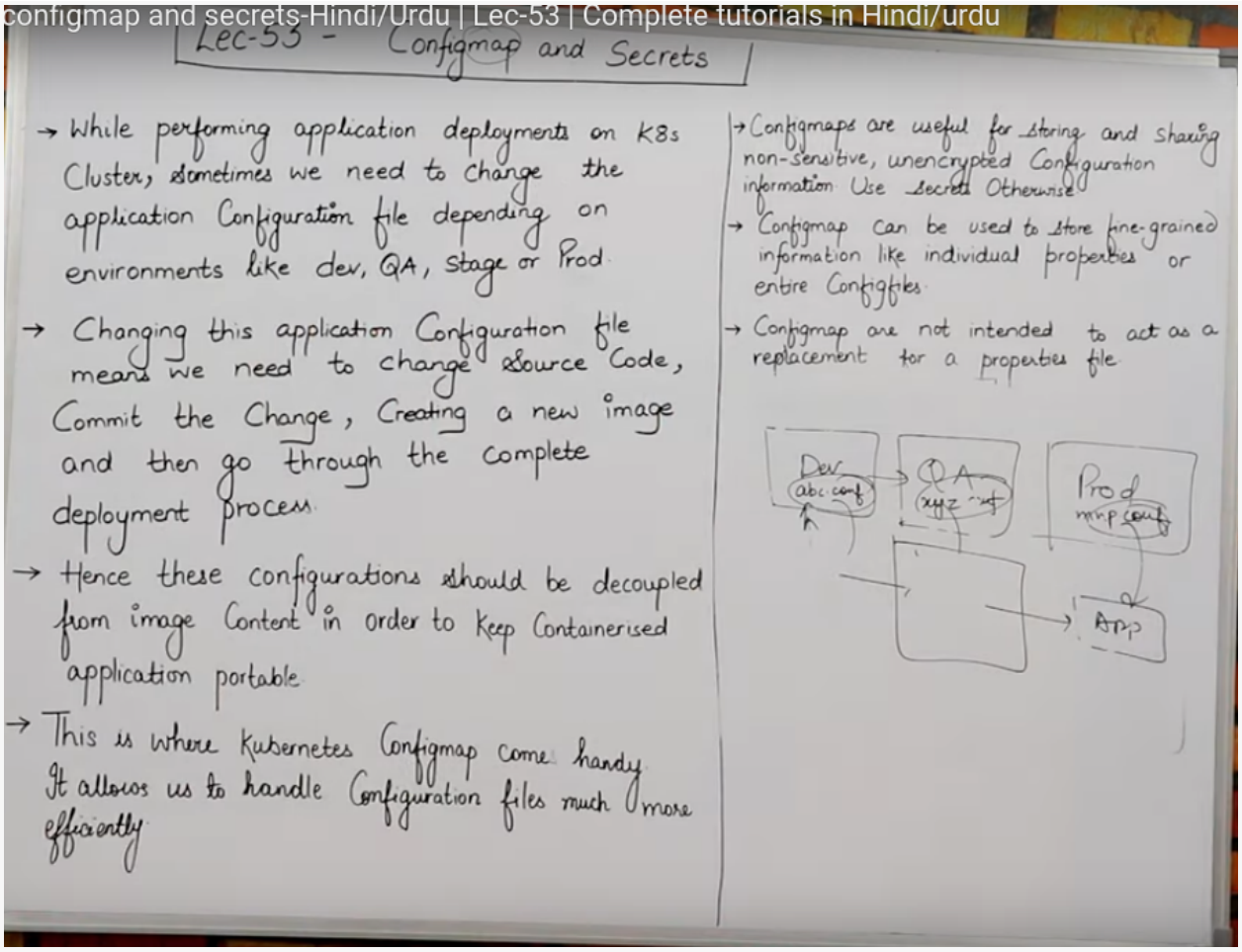
```
vishal@vishal-HP-245-G8:~/kubernetes/liveness-probe$ kubectl exec -it mylivenessprobe -- /bin/bash
root@mylivenessprobe:/#
root@mylivenessprobe:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@mylivenessprobe:/# cd /tmp
root@mylivenessprobe:/tmp# ls
healthy
root@mylivenessprobe:/tmp# cat *
root@mylivenessprobe:/tmp#
root@mylivenessprobe:/tmp# echo $?
0
root@mylivenessprobe:/tmp# ls
healthy
root@mylivenessprobe:/tmp# rm -rf *
```

- liveness will wait upto 30 second time
- then it will restart the container

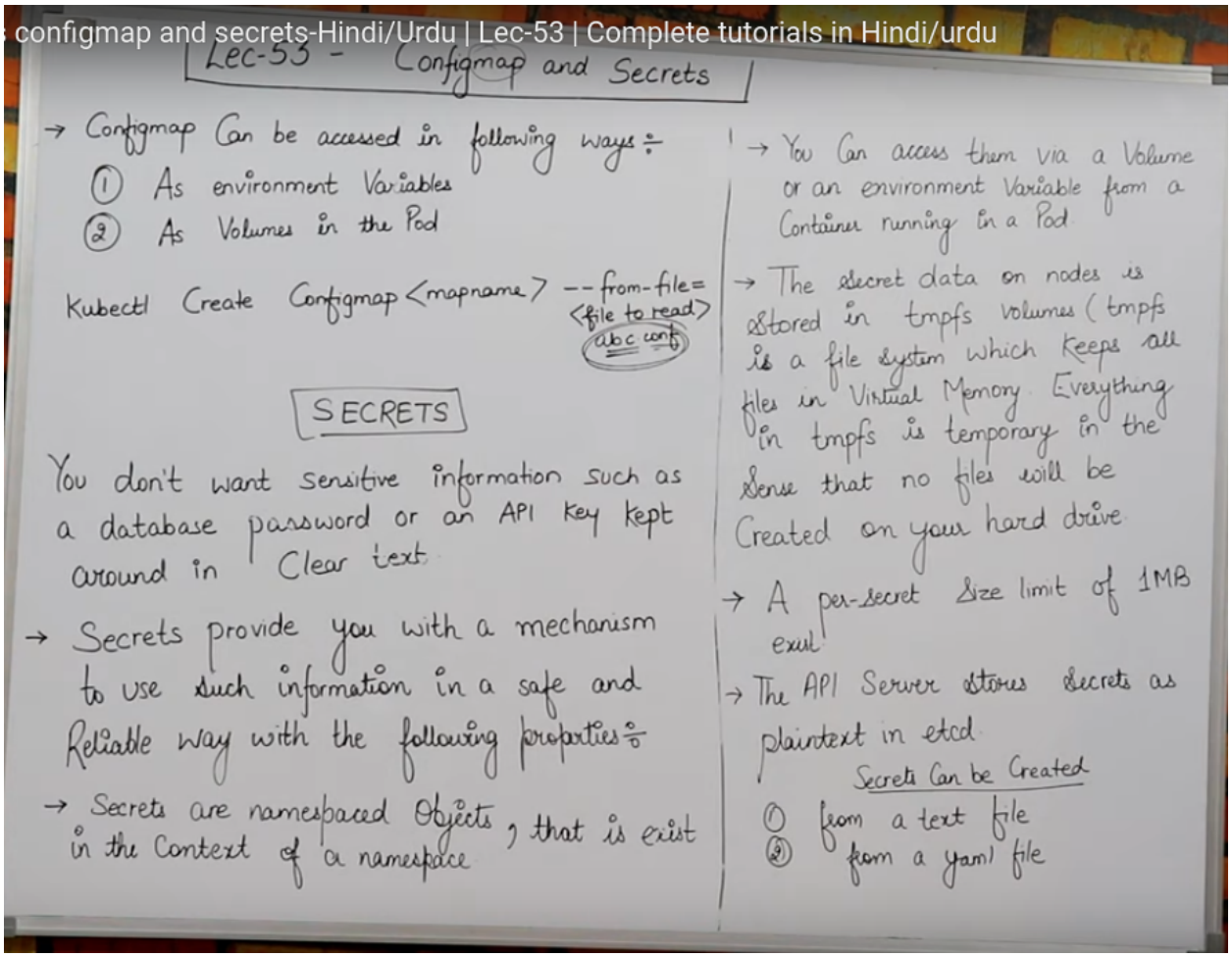
```
vishal@vishal-HP-245-G8:~/kubernetes/liveness-probe$ kubectl get pods
NAME              READY   STATUS    RESTARTS      AGE
mylivenessprobe   1/1     Running   1 (56s ago)   7m46s
vishal@vishal-HP-245-G8:~/kubernetes/liveness-probe$ kubectl exec -it mylivenessprobe -- /bin/ba
root@mylivenessprobe:/# cd /tmp
root@mylivenessprobe:/tmp# ls
healthy
root@mylivenessprobe:/tmp#
```

==========

ConfigMap & Secrete



==========



Deploy using volume (configmap)

- vi sample.conf

```
this is my conf file
```

- save and exist
- kubectl create configmap mymap --from-file=sample.conf
- kubectl get configmap

==========

```
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl create configmap mymap --from-file=sample.conf
configmap/mymap created
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl get configmap
NAME               DATA   AGE
kube-root-ca.crt   1      3d19h
mymap              1      25s
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$
```

- kubectl describe configmap mymap

```
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl get configmap
NAME               DATA   AGE
kube-root-ca.crt   1      3d19h
mymap              1      3m23s
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl describe configmap mymap
Name:         mymap
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
sample.conf:
----
this is my conf file


BinaryData
====

Events:  <none>
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$
```

==========

- vi deployconfigmap.yml

```
kind: Pod
metadata:
  name: myvolconfig
spec:
  containers:
  - name: c1
    image: centos
    command: ["/bin/bash", "-c", "while true; do echo vishalk17; sleep 5 ; done"]
    volumeMounts:
      - name: testconfigmap
        mountPath: "/tmp/config"   # the config files will be mounted as ReadOnly by default
here
  volumes:
  - name: testconfigmap
    configMap:
       name: mymap   # this should match the config map name created in the first step
       items: is my conf file
       - key: sample.conf
         path: sample.conf
```

```
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl apply -f deployconfigmap.yml
pod/myvolconfig created
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl get pods
NAME             READY   STATUS    RESTARTS        AGE
mylivenessprobe  1/1     Running   12 (7m1s ago)   2d16h
myvolconfig      1/1     Running   0               22s
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl exec -it myvolconfig -- /bin/bash
[root@myvolconfig /]#
[root@myvolconfig /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@myvolconfig /]#
[root@myvolconfig /]# cd /tmp
[root@myvolconfig tmp]# ls
config  ks-script-4luisyla  ks-script-o23i7rc2  ks-script-x6ei4wuu
[root@myvolconfig tmp]# ls -la
total 44
drwxrwxrwt 1 root root 4096 Jul 17 05:10 .
drwxr-xr-x 1 root root 4096 Jul 17 05:10 ..
drwxrwxrwt 2 root root 4096 Sep 15  2021 .ICE-unix
drwxrwxrwt 2 root root 4096 Sep 15  2021 .Test-unix
drwxrwxrwt 2 root root 4096 Sep 15  2021 .X11-unix
drwxrwxrwt 2 root root 4096 Sep 15  2021 .XIM-unix
drwxrwxrwt 2 root root 4096 Sep 15  2021 .font-unix
drwxrwxrwx 3 root root 4096 Jul 17 05:09 config
-rwx------ 1 root root  701 Sep 15  2021 ks-script-4luisyla
-rwx------ 1 root root  671 Sep 15  2021 ks-script-o23i7rc2
-rwx------ 1 root root  291 Sep 15  2021 ks-script-x6ei4wuu
[root@myvolconfig tmp]# c
bash: c: command not found
[root@myvolconfig tmp]# cd config/
[root@myvolconfig config]# ls
sample.conf
[root@myvolconfig config]# cat *
this is my conf file
[root@myvolconfig config]#
```

==========

Deploy using env variable (configmap)

- vi deployenv.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myenvconfig
spec:
  containers:
  - name: c1
    image: centos
    command: ["/bin/bash", "-c", "while true; do echo vishalk17; sleep 5 ; done"]
    env:
    - name: MYENV         # env name in which value of the key is stored
      valueFrom:
        configMapKeyRef:
          name: mymap      # name of the config created
          key: sample.conf
```

==========

```
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ vi deployenv.yml
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ ls
deployconfigmap.yml  deployenv.yml  sample.conf
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl get deploy
No resources found in default namespace.
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl apply -f deployenv.yml
pod/myenvconfig created
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl get deploy
No resources found in default namespace.
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl get pods
NAME              READY   STATUS    RESTARTS       AGE
myenvconfig       1/1     Running   0              8s
mylivenessprobe   1/1     Running   12 (16m ago)   2d16h
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/configmap$ kubectl exec -it myenvconfig -- /bin/bash
[root@myenvconfig /]#
[root@myenvconfig /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@myenvconfig /]#
[root@myenvconfig /]# env
LANG=en_US.UTF-8
HOSTNAME=myenvconfig
MYENV=this is my conf file

KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_PORT=tcp://10.96.0.1:443
PWD=/
HOME=/root
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
TERM=xterm
SHLVL=1
KUBERNETES_SERVICE_PORT=443
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_SERVICE_HOST=10.96.0.1
LESSOPEN=||/usr/bin/lesspipe.sh %s
_=/usr/bin/env
[root@myenvconfig /]#
```

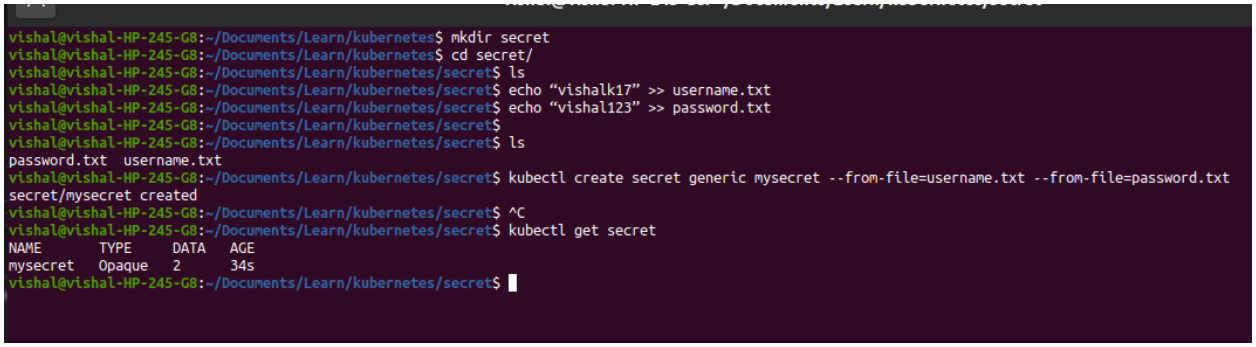
==========

secret

Using volume mount :

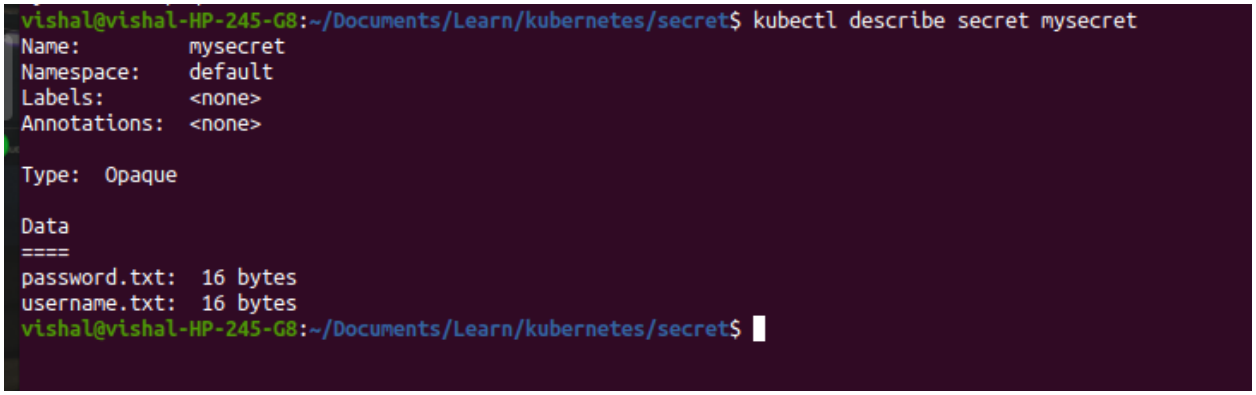
Create files:

- echo “vishalk17” >> username.txt
- echo “vishal123” >> password.txt
- kubectl create secret generic mysecret --from-file=username.txt --from-file=password.txt



- kubectl describe secret mysecret

it will not display contents of mysecret



==========

```
apiVersion: v1
kind: Pod
metadata:
  name: myvolsecret
spec:
  containers:
  - name: c1
    image: centos
    command: ["/bin/bash", "-c", "while true; do echo vishalk17; sleep 5 ; done"]
    volumeMounts:
      - name: testsecret
        mountPath: "/tmp/mysecrets"   # the secret files will be mounted as ReadOnly by
default here
  volumes:
  - name: testsecret
    secret:
       secretName: mysecret
```

```
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ ls
password.txt  username.txt
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ vi deploysecret-vol.yml
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ kubectl apply -f deploysecret-vol.yml
pod/myvolsecret created
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ kubectl get pods
NAME             READY   STATUS    RESTARTS         AGE
mylivenessprobe  1/1     Running   14 (2m28s ago)   2d17h
myvolsecret      1/1     Running   0                47s
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ kubectl exec -it myvolsecret -- /bin/bash
[root@myvolsecret /]#
[root@myvolsecret /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@myvolsecret /]# cd /tmp
[root@myvolsecret tmp]# ls
ks-script-4luisyla  ks-script-o23i7rc2  ks-script-x6ei4wuu  mysecrets
[root@myvolsecret tmp]# cd mysecrets/
[root@myvolsecret mysecrets]# ls
password.txt  username.txt
[root@myvolsecret mysecrets]# echo *
password.txt username.txt
[root@myvolsecret mysecrets]# cat *
“vishal123”
“vishalk17”
[root@myvolsecret mysecrets]#
```

==========

Using env variable mount (secret) :

```
apiVersion: v1
kind: Pod
metadata:
  name: myenvsecret
spec:
  containers:
  - name: c1
    image: centos
    command: ["/bin/bash", "-c", "while true; do echo vishalk17; sleep 5 ; done"]
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: username.txt
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password.txt
    volumeMounts:
      - name: testsecret
        mountPath: "/tmp/mysecrets"   # the secret files will be mounted as ReadOnly by
default here
  volumes:
  - name: testsecret
    secret:
      secretName: mysecret
```

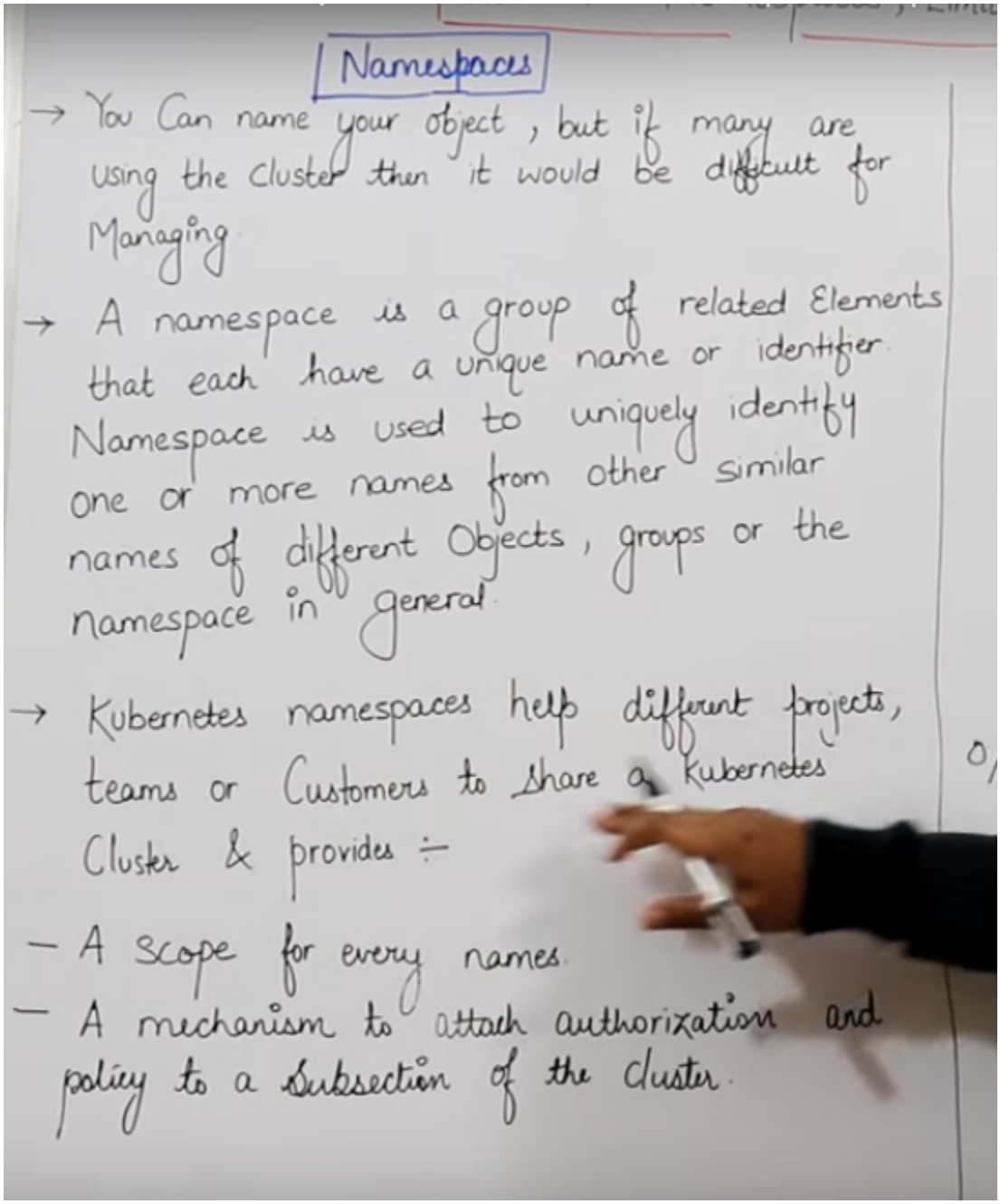
==========

```
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ kubectl apply -f deploysecret-env.yml
pod/myenvsecret created
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ kubectl get pods
NAME          READY   STATUS              RESTARTS   AGE
myenvsecret   0/1     ContainerCreating   0          4s
vishal@vishal-HP-245-G8:~/Documents/Learn/kubernetes/secret$ kubectl exec -it myenvsecret -- /bin/bash
[root@myenvsecret /]#
[root@myenvsecret /]# env
LANG=en_US.UTF-8
SECRET_PASSWORD="vishal123"

HOSTNAME=myenvsecret
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
SECRET_USERNAME="vishalk17"

KUBERNETES_PORT=tcp://10.96.0.1:443
PWD=/
HOME=/root
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
TERM=xterm
SHLVL=1
KUBERNETES_SERVICE_PORT=443
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_SERVICE_HOST=10.96.0.1
LESSOPEN=||/usr/bin/lesspipe.sh %s
_=/usr/bin/env
[root@myenvsecret /]#
```

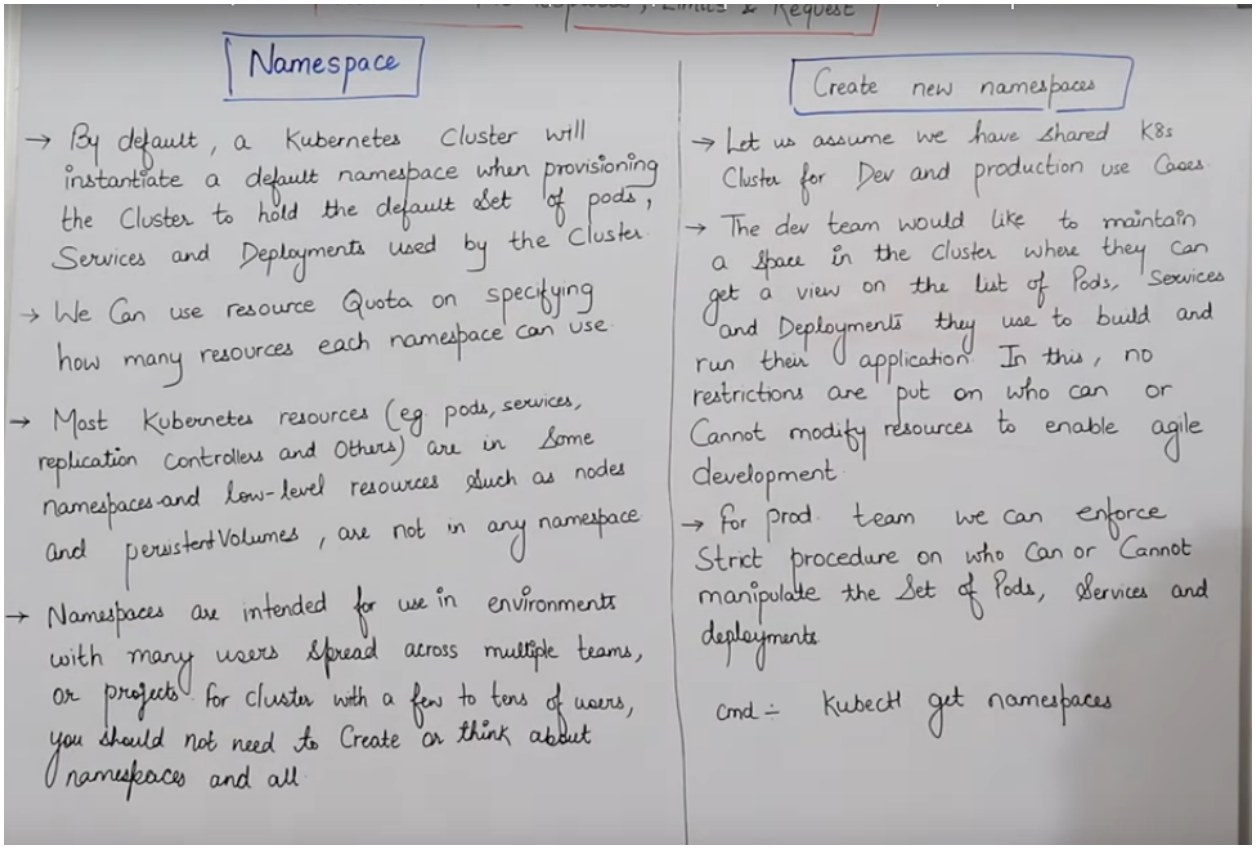
==========

Namespaces and Resource quota

Namespaces :



==========



- **Namespaces:**
  - Partition resources and provide isolation within a cluster.
  - Logically divide a cluster into multiple virtual clusters.
  - Each namespace has its own set of resources.
- **Resource Quotas:**
  - Limit the amount of resources that can be used within a namespace.
  - Help ensure that a namespace does not consume excessive resources.
  - Set limits on CPU, memory, and storage usage within a namespace.
  - Specify limits on the number of pods, services, and other resources that can be created.
- **Together:**
  - Effectively manage and allocate resources within a Kubernetes cluster.
  - Provide better isolation and control over resource usage.

Benefits of using namespaces and resource quotas

- Improve resource utilization
- Prevent resource conflicts
- Improve security
- Make it easier to manage large Kubernetes deployments

==========

- kubectl get namespace .. get list of namespace available on the k8s

Create new namespace

- vi devns.yml

```
apiVersion: v1
kind: Namespace
metadata:
   name: dev
   labels:
     name: dev
```

- save /exit/ applied

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl get namespace
NAME              STATUS   AGE
default           Active   3d22h
dev               Active   28s
ingress-nginx     Active   3h30m
kube-node-lease   Active   3d22h
kube-public       Active   3d22h
kube-system       Active   3d22h
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$
```

- create deployment in **dev** namespace
- vi pod.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Technical Guftgu; sleep 5 ; done"]
  restartPolicy: Never
```

- deploy in dev namespace
- vi pod.yml

==========

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl apply -f devns.yml
namespace/dev created
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl get namespace
NAME              STATUS   AGE
default           Active   3d22h
dev               Active   10s
ingress-nginx     Active   3h39m
kube-node-lease   Active   3d22h
kube-public       Active   3d22h
kube-system       Active   3d22h
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl get pods -n dev
No resources found in dev namespace.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl apply -f
devns.yml     .pod.xm.swp  pod.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl apply -f pod.yml -n dev
pod/testpod created
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl get pods
No resources found in default namespace.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl get pods -n dev
NAME      READY   STATUS    RESTARTS   AGE
testpod   1/1     Running   0          16s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$
```

by default k8s checks pods in default namespace, in that case you will need to specify namespace

else change default namespace to dev

How to change namespace and set another one as a default.

- kubectl config set-context $(kubectl config current-context) --namespace=dev

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl config set-context $(kubectl config current-context) --namespace=dev
Context "minikube" modified.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
testpod   1/1     Running   0          4m2s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$
```

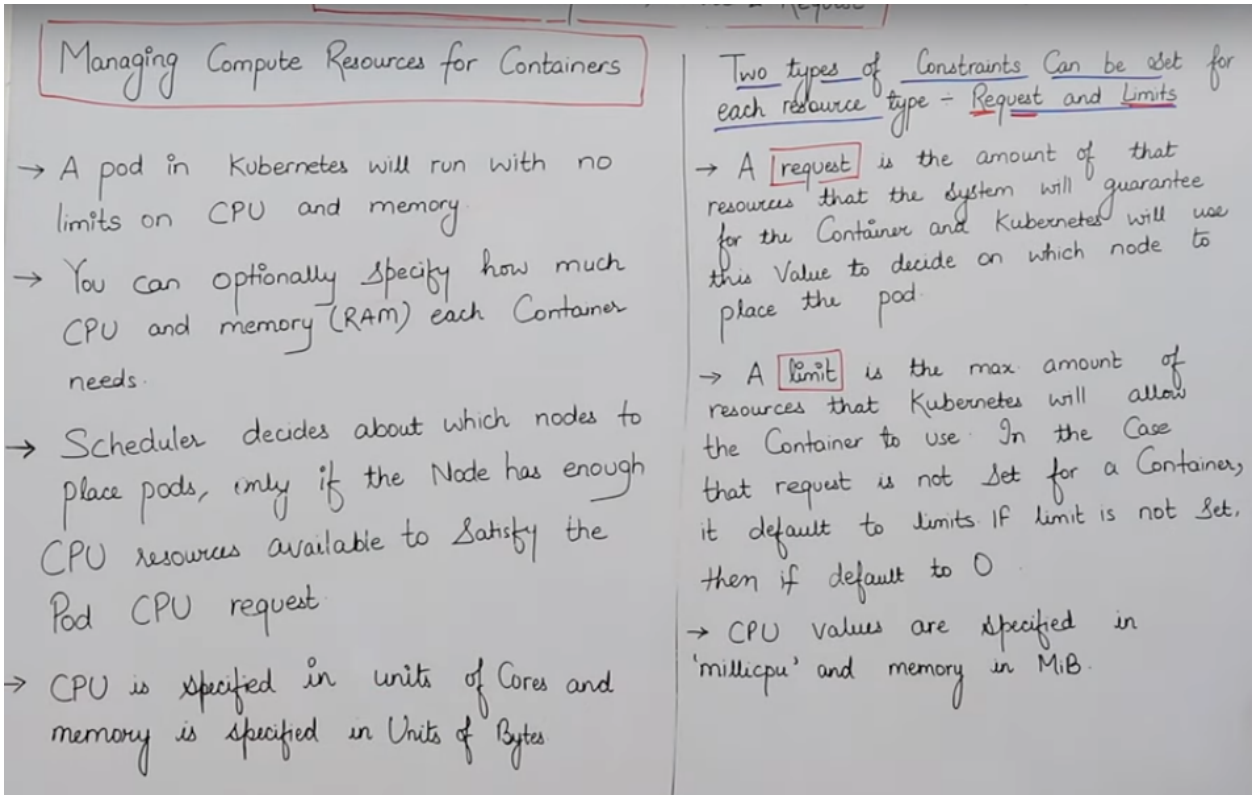
check in which namespace we are:

- kubectl config view | grep namespace:

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$ kubectl config view | grep namespace:
    namespace: dev
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota$
```

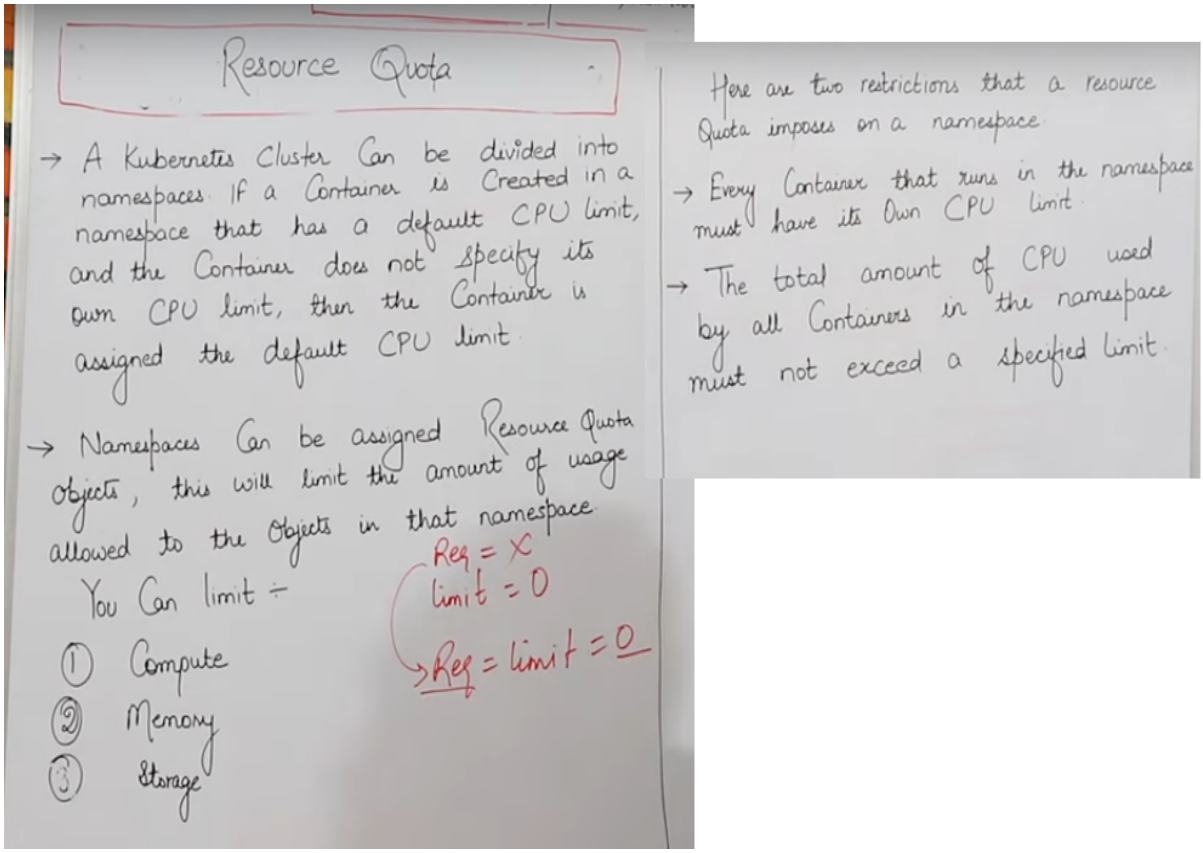
==========

Resource Quota:



request = mention
limit = mention
no issues
—----------
request = not mention
limit = mention
limit = request
—----------
request = mention
limit = not mentioned
unlimited resources (default to zero)
—----------

==========



Set

```
apiVersion: v1
kind: ResourceQuota
metadata:
   name: myquota
spec:
  hard:
    limits.cpu: "400m"
    limits.memory: "400Mi"
    requests.cpu: "200m"
    requests.memory: "200Mi"
```

The resource quota provided is named `myquota` and it has two hard limits:

- `limits.cpu: "400m"`: This means that no Pod in the namespace `dev` can use more than 400 millicores of CPU.

==========

- `limits.memory: "400Mi"`: This means that no Pod in the namespace `dev` can use more than 400 megabytes of memory.

The resource quota also has two soft limits:

- `requests.cpu: "200m"`: This means that Pods in the namespace `dev` are **recommended** to request no more than 200 millicores of CPU.
- `requests.memory: "200Mi"`: This means that Pods in the namespace `dev` are **recommended** to request no more than 200 megabytes of memory.

what if we demand max resource in deployment , if resource quota already allocated as above ??

- vi deployment.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: deployments
spec:
  replicas: 3
  selector:
    matchLabels:
     objtype: deployment
  template:
    metadata:
      name: testpod8
      labels:
        objtype: deployment
    spec:
     containers:
       - name: c00
         image: ubuntu
         command: ["/bin/bash", "-c", "while true; do echo Vishal; sleep 5 ; done"]
         resources:
            requests:
              cpu: "200m"
```

- replicas are not creating reason why behind this is in above deployment
- we are creating 3 replica set that means 200 x3 = 600 m cpu
- In resource quota we set max limit is 400m cpu that is why demand is not fulfilling

  and containers/pods are not going to create

  Note: 1 cpu = 1000m cpu

==========

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl apply -f deployment.yml
deployment.apps/deployments created
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl get pods
No resources found in dev namespace.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployments   0/3     0            0           16s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl describe deploy deployments
Name:                   deployments
Namespace:              dev
CreationTimestamp:      Mon, 17 Jul 2023 15:19:13 +0530
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               objtype=deployment
Replicas:               3 desired | 0 updated | 0 total | 0 available | 3 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  objtype=deployment
  Containers:
```

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl get pods
No resources found in dev namespace.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl get pods
No resources found in dev namespace.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl get pods
No resources found in dev namespace.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl get rs
NAME                     DESIRED   CURRENT   READY   AGE
deployments-76bcf47454   3         0         0       66s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ kubectl describe rs deployments-76bcf47454
Name:           deployments-76bcf47454
Namespace:      dev
Selector:       objtype=deployment,pod-template-hash=76bcf47454
Labels:         objtype=deployment
                pod-template-hash=76bcf47454
Annotations:    deployment.kubernetes.io/desired-replicas: 3
                deployment.kubernetes.io/max-replicas: 4
                deployment.kubernetes.io/revision: 1
Controlled By:  Deployment/deployments
Replicas:       0 current / 3 desired
Pods Status:    0 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
```

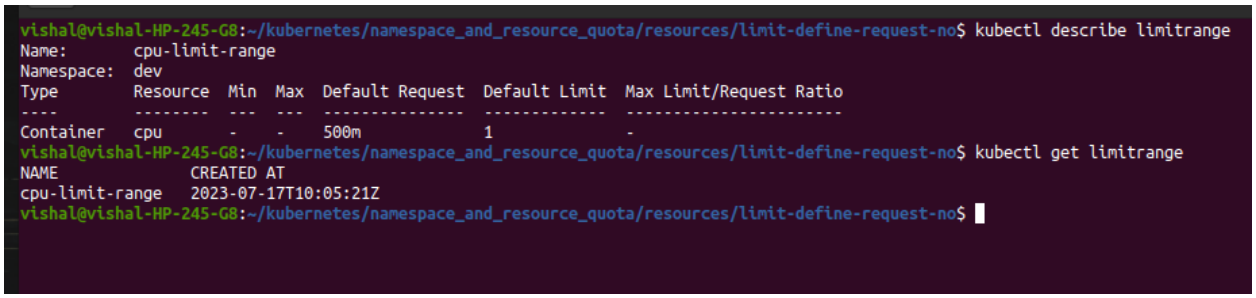
```
  ReplicaFailure   True    FailedCreate
Events:
  Type     Reason        Age              From                   Message
  ----     ------        ----             ----                   -------
  Warning  FailedCreate  83s              replicaset-controller  Error creating: pods "deployments-76bcf47454-9xjdk" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  83s              replicaset-controller  Error creating: pods "deployments-76bcf47454-7q42m" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  83s              replicaset-controller  Error creating: pods "deployments-76bcf47454-59r7n" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  83s              replicaset-controller  Error creating: pods "deployments-76bcf47454-6nj2c" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  83s              replicaset-controller  Error creating: pods "deployments-76bcf47454-twvmt" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  83s              replicaset-controller  Error creating: pods "deployments-76bcf47454-g74v4" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  82s              replicaset-controller  Error creating: pods "deployments-76bcf47454-8d9dd" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  82s              replicaset-controller  Error creating: pods "deployments-76bcf47454-sfrt5" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  81s              replicaset-controller  Error creating: pods "deployments-76bcf47454-7spjn" is forbidden: failed quota: myquota: must specify limits.cpu for:
c00; limits.memory for: c00; requests.memory for: c00
  Warning  FailedCreate  1s (x6 over 80s)  replicaset-controller  (combined from similar events): Error creating: pods "deployments-76bcf47454-4jpqg" is forbidden: failed quota: myquot
a: must specify limits.cpu for: c00; limits.memory for: c00; requests.memory for: c00
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources$ vi deployment.yml
```

==========

what if request not mention , only limit mention ?????

- vi limit.yml

```
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
spec:
  limits:
  - default:
      cpu: 1
    defaultRequest:
      cpu: 0.5
    type: Container
```



- vi deployment.yml

```
kind: Pod
apiVersion: v1
metadata:
  name: testpod
spec:
  containers:
    - name: c00
      image: ubuntu
      command: ["/bin/bash", "-c", "while true; do echo Technical Guftgu; sleep 5 ; done"]
  restartPolicy: Never
```

==========

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/limit-define-request-no$ kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
testpod   1/1     Running   0          29s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/limit-define-request-no$ vi deployment.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/limit-define-request-no$
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/limit-define-request-no$ kubectl describe pod testpod
Name:             testpod
Namespace:        dev
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.104
Start Time:       Mon, 17 Jul 2023 15:43:30 +0530
Labels:           <none>
Annotations:      kubernetes.io/limit-ranger: LimitRanger plugin set: cpu request for container c00; cpu limit for container c00
Status:           Running
IP:               10.244.0.85
IPs:
  IP:  10.244.0.85
Containers:
  c00:
    Container ID:  docker://55d061258c56158243e93ac783c53f94f7c03e059d5598ee0231f73f8585a9d1
    Image:         ubuntu
    Image ID:      docker-pullable://ubuntu@sha256:0bced47fffa3361afa981854fcabcd4577cd43cebbb808cea2b1f33a3dd7f508
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/bash
      -c
      while true; do echo Technical Guftgu; sleep 5 ; done
    State:          Running
      Started:      Mon, 17 Jul 2023 15:43:34 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:  1
    Requests:
      cpu:        500m
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-gg6dg (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
```

- in this case default limit set to “default limit request if not mentioned”

——————————————————————————

what if we define limit but not req ?????

- vi limit.yml

```
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
spec:
  limits:
  - default:
      cpu: 1
    defaultRequest:
      cpu: 0.5
    type: Container
```

==========

- vi cpu2.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: default-cpu-demo-2
spec:
  containers:
  - name: default-cpu-demo-2-ctr
    image: nginx
    resources:
      limits:
        cpu: "1"
```

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/define-req-not-limit$ kubectl apply -f cpu2.yml
pod/default-cpu-demo-2 created
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/define-req-not-limit$ kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
default-cpu-demo-2   0/1     ContainerCreating   0          21s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/define-req-not-limit$ kubectl describe pod default-cpu-demo-2
Name:             default-cpu-demo-2
Namespace:        dev
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.104
Start Time:       Mon, 17 Jul 2023 16:09:21 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.244.0.86
IPs:
  IP:  10.244.0.86
Containers:
  default-cpu-demo-2-ctr:
    Container ID:   docker://77a9bf91ef181cf77b0da2a6e02d59256df7c27ed08302f30fe0c5aea5bf46b6
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e3140d6cfe29c8892c7ef
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 17 Jul 2023 16:09:53 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:  1
    Requests:
      cpu:        1
    Environment:  <none>
    Mounts:
```

==========

—----------—

define request not limit :

- vi limit.yml

```
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
spec:
  limits:
  - default:
      cpu: 1
    defaultRequest:
      cpu: 0.5
    type: Container
```

- vi cpu3.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: default-cpu-demo-3
spec:
  containers:
  - name: default-cpu-demo-3-ctr
    image: nginx
    resources:
      requests:
        cpu: "0.75"
```

==========

```
default-cpu-demo-3   0/1     ContainerCreating   0          14s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/define-req-not-limit$ kubectl get pods
NAME                 READY   STATUS    RESTARTS   AGE
default-cpu-demo-3   1/1     Running   0          17s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/define-req-not-limit$ kubectl describe pod default-cpu-demo-3
Name:             default-cpu-demo-3
Namespace:        dev
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.104
Start Time:       Mon, 17 Jul 2023 16:29:33 +0530
Labels:           <none>
Annotations:      kubernetes.io/limit-ranger: LimitRanger plugin set: cpu limit for container default-cpu-demo-3-ctr
Status:           Running
IP:               10.244.0.87
IPs:
  IP:  10.244.0.87
Containers:
  default-cpu-demo-3-ctr:
    Container ID:   docker://ab530978cfefd1c70335f8d49851f5c8a764fb88f4d5216d4793923ddd5ab523
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e3140d6cfe29c8892c7ef
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 17 Jul 2023 16:29:47 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:  1
    Requests:
      cpu:        750m
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-hfzhb (ro)
```

defaul limit set to 1

- here we havent mentioned what would be default limit of deployment container
- so , default limit is = already set limit volume of namespace

==========

—---------- For memory —----------

vi limit-range.yml

```
apiVersion: v1
kind: LimitRange
metadata:
  name: mem-min-max-demo-lr
spec:
  limits:
  - max:
      memory: 1Gi
    min:
      memory: 500Mi
    type: Container
```

- in above thing we have set default memory limits for a current namespace

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory$ ls
limit-range.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory$ kubectl apply -f limit-range.yml
limitrange/mem-min-max-demo-lr created
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory$
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory$
```

demo1—---------->

```
vi mem1.yml
apiVersion: v1
kind: Pod
metadata:
  name: constraints-mem-demo
spec:
  containers:
  - name: constraints-mem-demo-ctr
    image: nginx
    resources:
      limits:
        memory: "800Mi"
      requests:
        memory: "600Mi"

# in default limit range applied  my max 1gb mem and min 500 mb
# in this file I m setting max limit below and request above
```

- no issues found container running

==========

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo1$ l
mem1.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo1$ kubectl apply -f mem1.yml
pod/constraints-mem-demo created
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo1$ kubectl get pod
NAME                 READY   STATUS    RESTARTS   AGE
constraints-mem-demo   1/1     Running   0          6s
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo1$ vi mem1.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo1$
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo1$ kubectl describe pod constraints-mem-demo
Name:             constraints-mem-demo
Namespace:        dev
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.104
Start Time:       Mon, 17 Jul 2023 17:02:31 +0530
Labels:           <none>
Annotations:      kubernetes.io/limit-ranger:
                    LimitRanger plugin set: cpu request for container constraints-mem-demo-ctr; cpu limit for container constraints-mem-demo-ctr
Status:           Running
IP:               10.244.0.88
IPs:
  IP:  10.244.0.88
Containers:
  constraints-mem-demo-ctr:
    Container ID:   docker://5019aaae88e87bb6a73a8a0779f6242a34579a86c33b210fbf2050543fad4f61
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:08bc36ad52474e528cc1ea3426b5e3f4bad8a130318e3140d6cfe29c8892c7ef
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 17 Jul 2023 17:02:34 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     1
      memory:  800Mi
    Requests:
      cpu:        500m
      memory:     600Mi
    Environment:  <none>
```

demo 2 ——————————————————>

vi mem2.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-mem-demo
spec:
  containers:
  - name: constraints-mem-demo-ctr
    image: nginx
    resources:
      limits:
        memory: "1800Mi"
      requests:
        memory: "600Mi"
```

- here i have increase limit to 1800 which exceed limit of default namespace

==========

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo2$ ls
mem2.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo2$ kubectl get pods
No resources found in dev namespace.
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo2$ kubectl apply -f mem2.yml
Error from server (Forbidden): error when creating "mem2.yml": pods "constraints-mem-demo" is forbidden: maximum memory usage per Container is 1Gi, but limit is 1800Mi
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo2$
```

- its not creating reason mentioned above

demo3 ——————————————————————>

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-mem-demo
spec:
  containers:
  - name: constraints-mem-demo-ctr
    image: nginx
    resources:
      limits:
        memory: "800Mi"
      requests:
        memory: "300Mi"
```

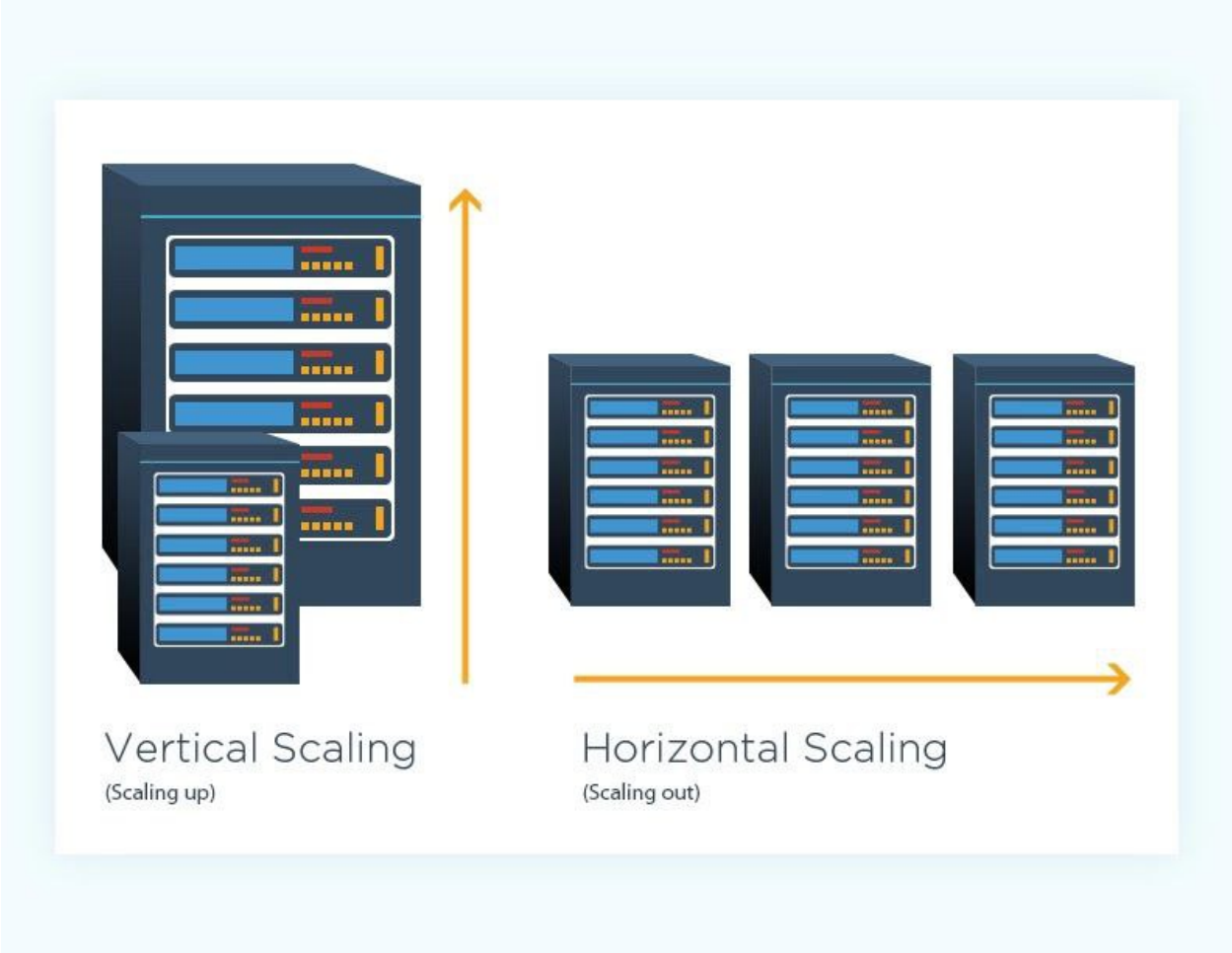
- in above things limit is under default limit of namespace but request is below of default req. limit of namespace

```
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo3$ ls
mem3.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo3$ vi mem3.yml
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo3$ kubectl apply -f mem3.yml
Error from server (Forbidden): error when creating "mem3.yml": pods "constraints-mem-demo" is forbidden: minimum memory usage per Container is 500Mi, but request is 300Mi
vishal@vishal-HP-245-G8:~/kubernetes/namespace_and_resource_quota/resources/for-memory/demo3$
```

- container is not creating because of reason above mentioned

==========

Autoscaling



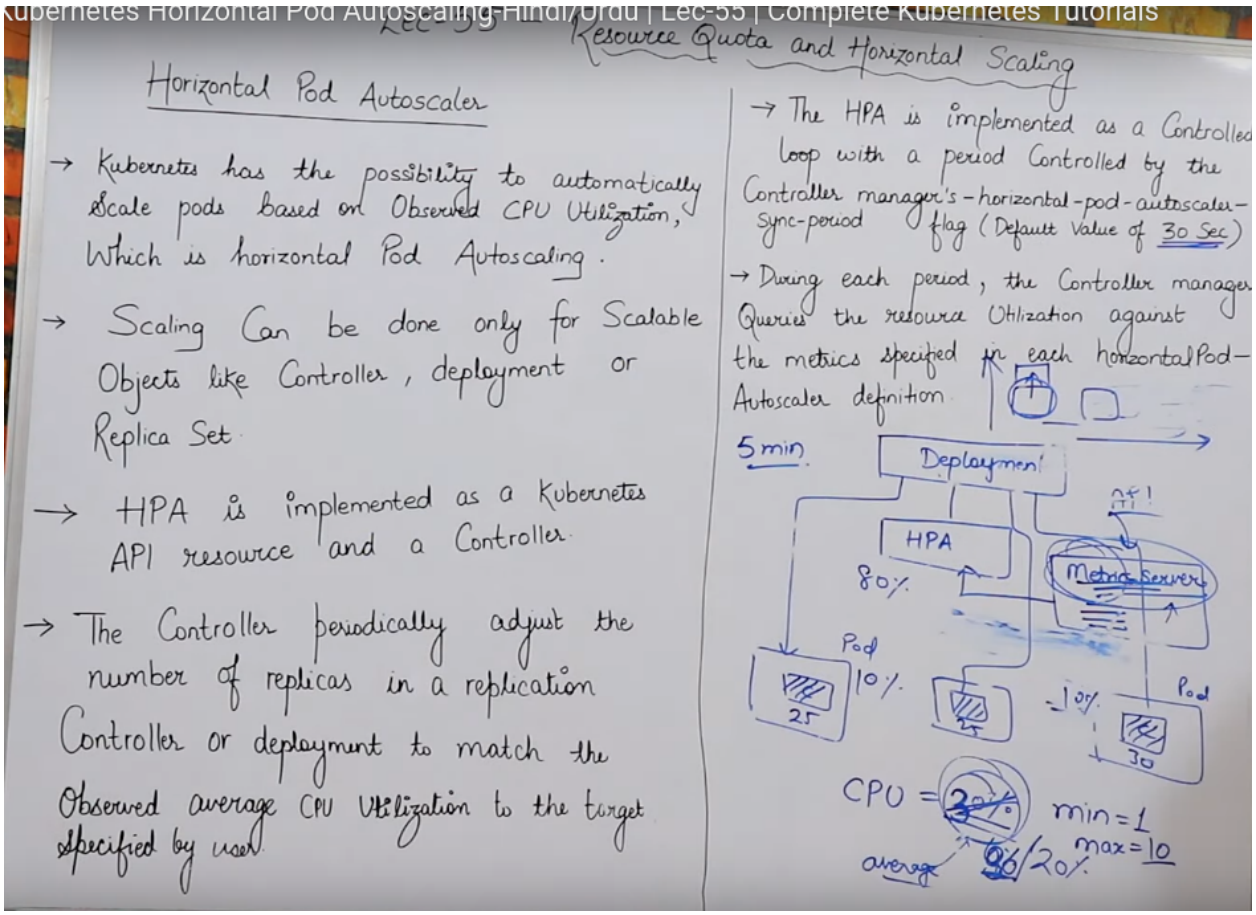
Horizontal scaling (aka scaling out) :

refers to adding additional nodes or machines to your infrastructure to cope with new demands.

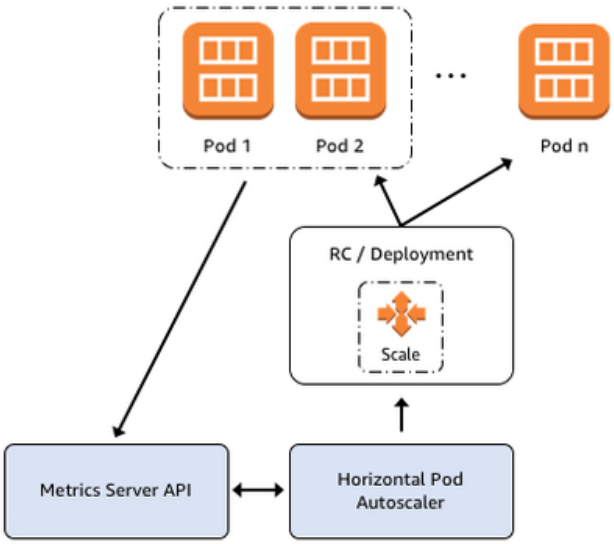
Vertical scaling : It describes adding more power to your current machines. For instance, if your server requires more processing power, vertical scaling would mean upgrading the CPUs. You can also vertically scale the memory, storage, or network speed.

==========

Horizontal Pod Autoscaler



==========



Components of HPA

The HPA has four main components:

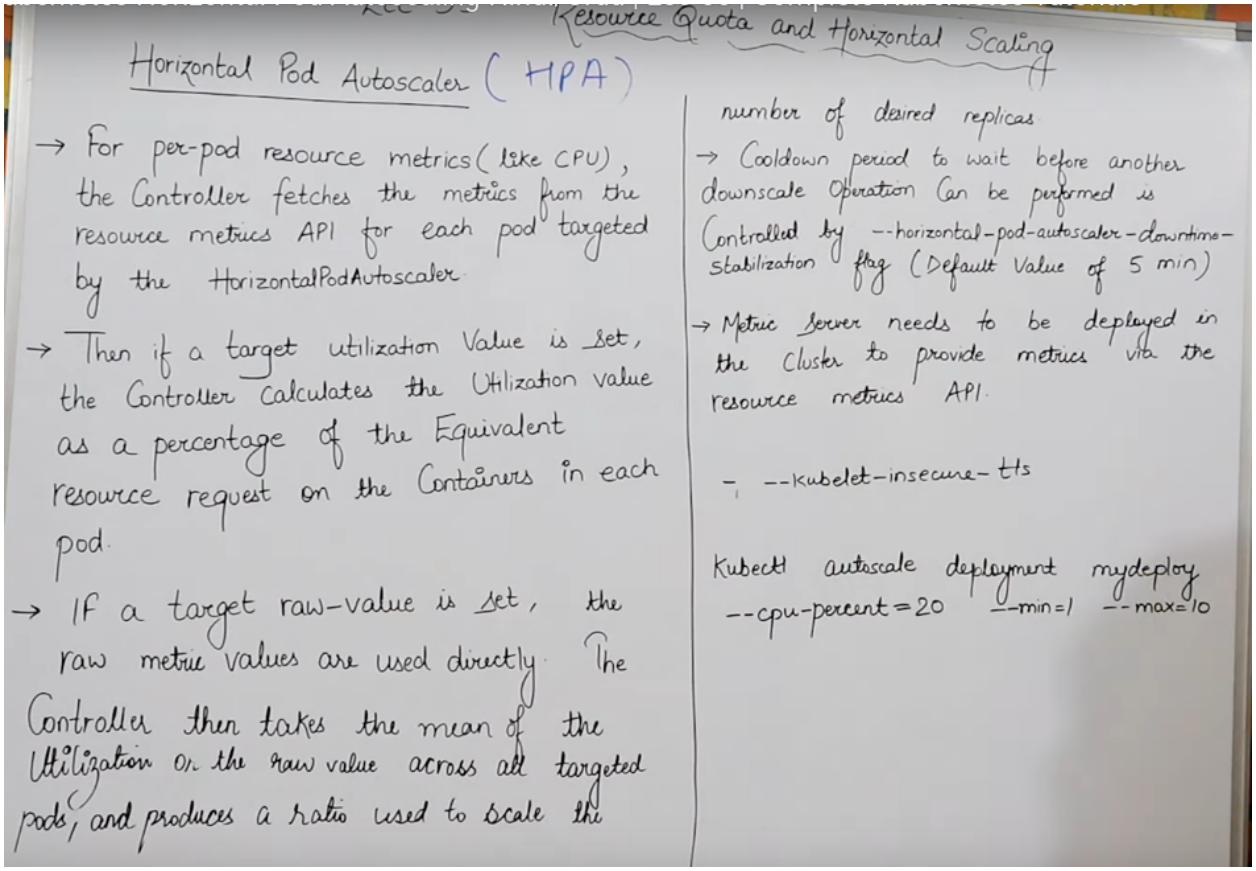
- **Deployment or ReplicaSet:** The HPA scales the number of pods in a deployment or replica set. It monitors the resource utilization of the pods in the specified deployment or replica set.
- **Metrics Server:** The Metrics Server is responsible for collecting the resource utilization metrics from the pods. It runs as a cluster-level component and collects metrics such as CPU and memory usage.
- **HPA Controller:** The HPA Controller is responsible for monitoring the metrics provided by the Metrics Server and calculating the desired number of replicas based on the defined scaling policy. It adjusts the number of replicas to maintain the desired average CPU utilization or other custom metrics.
- **Scale Subresource:** The Scale Subresource is an API endpoint that allows the HPA Controller to retrieve and update the scaling information for a specific deployment or replica set. It enables the HPA to adjust the number of replicas based on the desired scaling policy.

How HPA works

When the HPA is enabled and configured, it continuously monitors the resource utilization metrics of the pods in the deployment or replica set. Based on the defined scaling policy, such as target CPU utilization or custom metrics, the HPA Controller calculates the desired number of replicas needed to maintain the desired resource utilization.

If the observed CPU utilization exceeds the defined target, the HPA Controller increases the number of replicas. If the observed CPU utilization is lower than the target, the HPA Controller decreases the number of replicas. The scaling action is performed by updating the Scale Subresource of the deployment or replica set, which triggers the Kubernetes scheduler to adjust the number of pods accordingly.

==========



- installing metric server api on k8s
- wget -O metricserver.yml https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

By default, the Metrics Server requires a secure TLS connection to communicate with the kubelet API. and to avoid TLS certificate validation errors., disable it from the metricserver.yml

- open file file in editor , find the kind: Deployment
- under which in args add following line then save and exist

  --kubelet-insecure-tls

==========

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server
  namespace: kube-system
spec:
  selector:
    matchLabels:
      k8s-app: metrics-server
  strategy:
    rollingUpdate:
      maxUnavailable: 0
  template:
    metadata:
      labels:
        k8s-app: metrics-server
    spec:
      containers:
      - args:
        - --kubelet-insecure-tls
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
        - --kubelet-use-node-status-port
```

- apply that yml file , so that metric server will come in work

```
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler$ ls
horizonal-scaling  metricserver.yml
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler$ vi metricserver.yml
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler$ kubectl apply -f metricserver.yml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler$
```

==========

kubectl get namespace .. to get list of namespace

demo1 —------------------------------>

vi deployment.yml

```
kind: Deployment
apiVersion: apps/v1
metadata:
   name: mydeploy
spec:
   replicas: 1
   selector:
    matchLabels:
     name: deployment
   template:
     metadata:
       name: testpod8
       labels:
         name: deployment
     spec:
      containers:
        - name: c00
          image: httpd
          ports:
          - containerPort: 80
          resources:
            limits:
              cpu: 500m
            requests:
              cpu: 200
```

- save , exit and apply

==========

- kubectl get all …lists all the Kubernetes resources in the current namespace

```
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$ vi deployment.yml
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$ kubectl apply -f deployment.yml
deployment.apps/mydeploy created
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$ kubectl get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/mydeploy-6bd88977d5-rnfs2   1/1     Running   0          2m28s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   4d20h

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy    1/1     1            1           2m28s

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeploy-6bd88977d5    1         1         1       2m28s
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$
```

- apply horizontal autoscaling

kubectl autoscale deployment mydeploy --cpu-percent=20 --min=1 --max=10

- `kubectl autoscale`: This is the command to create a HPA.
- `deployment mydeploy`: This is the name of the deployment to autoscale.
- `--cpu-percent=20`: This is the target CPU utilization for the HPA.
- `--min=1`: This is the minimum number of pods for the HPA.
- `--max=10`: This is the maximum number of pods for the HPA.

Once the HPA is created, it will start monitoring the CPU utilization of the pods in the deployment. If the CPU utilization of the pods exceeds 20%, the HPA will increase the number of pods. If the CPU utilization of the pods falls below 20%, the HPA will decrease the number of pods.

The HPA will continue to monitor the CPU utilization of the pods and adjust the number of pods as needed. This ensures that the deployment has the correct number of pods to meet the demand.

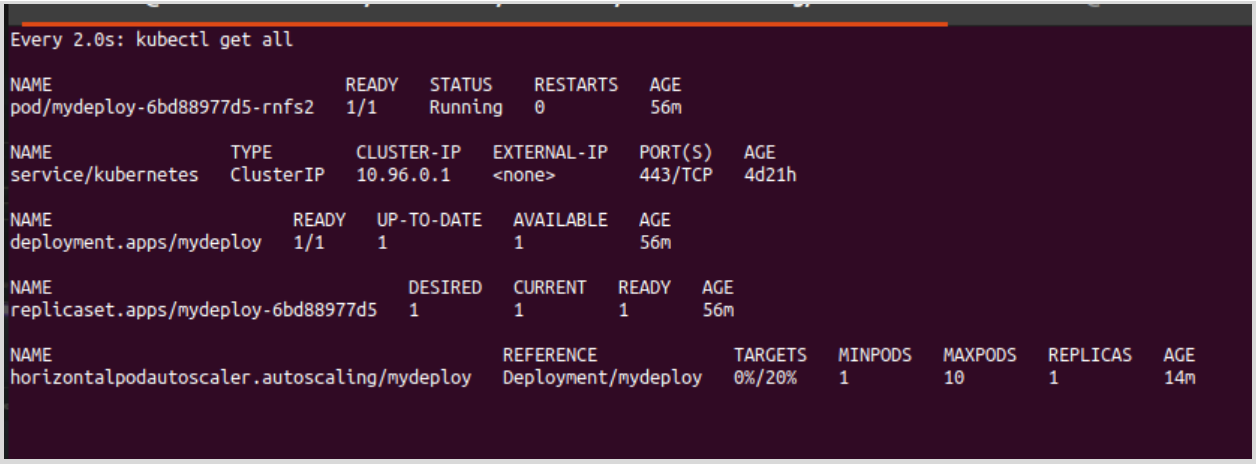
```
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$  kubectl autoscale deployment mydeploy --cpu-percent=20 --min=1 --max=10
horizontalpodautoscaler.autoscaling/mydeploy autoscaled
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$ kubectl get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/mydeploy-6bd88977d5-rnfs2   1/1     Running   0          42m

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   4d21h

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy    1/1     1            1           42m

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeploy-6bd88977d5    1         1         1       42m

NAME                                                  REFERENCE              TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
horizontalpodautoscaler.autoscaling/mydeploy          Deployment/mydeploy    <unknown>/20%   1         10        0          5s
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$
```

==========

Now for checking autoscaling working,

In the terminal use following command

watch kubectl get all … continuously monitor the status of all resources in Kubernetes



- above things will update at every 2 seconds automatically, so that we can trace the autoscaling going on or not

==========

Now, open new 2nd terminal and increase load in the container

- enter in the container then run following commands
- apt-get update -y && apt-get install stress -y

```
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mydeploy-6bd88977d5-rnfs2   1/1     Running   0          46m
vishal@vishal-HP-245-G8:~/kubernetes/autoscaler/horizonal-scaling/demo1$ kubectl exec -it mydeploy-6bd88977d5-rnfs2 -- /bin/bash
root@mydeploy-6bd88977d5-rnfs2:/usr/local/apache2#
root@mydeploy-6bd88977d5-rnfs2:/usr/local/apache2# apt-get update -y
Get:1 http://deb.debian.org/debian bookworm InRelease [147 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [52.1 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8904 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [4732 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [48.0 kB]
Fetched 9204 kB in 6s (1635 kB/s)
Reading package lists... Done
root@mydeploy-6bd88977d5-rnfs2:/usr/local/apache2# apt-get install stress -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  stress
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 21.9 kB of archives.
After this operation, 57.3 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 stress amd64 1.0.7-1 [21.9 kB]
Fetched 21.9 kB in 0s (237 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package stress.
(Reading database ... 8502 files and directories currently installed.)
Preparing to unpack .../stress_1.0.7-1_amd64.deb ...
Unpacking stress (1.0.7-1) ...
Setting up stress (1.0.7-1) ...
root@mydeploy-6bd88977d5-rnfs2:/usr/local/apache2#
```

then increase the load in the container using stress command,

stress -c 4

- previously we have set targeted value for autoscaling is 20 % of avg cpu load
- so here we are increasing the load beyond 20 % ,Let see what will happen.

```
vishal@vishal-HP-245-G8: ~/kubernetes/autoscaler/horizonal-scaling/demo
root@mydeploy-6bd88977d5-rnfs2:/usr/local/apache2#
root@mydeploy-6bd88977d5-rnfs2:/usr/local/apache2# stress -c 4
stress: info: [226] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

- now switch to first terminal , check things

==========

```
Every 2.0s: kubectl get all

NAME                            READY   STATUS    RESTARTS   AGE
pod/mydeploy-6bd88977d5-292cv   0/1     Pending   0          10s
pod/mydeploy-6bd88977d5-2ht5g   1/1     Running   0          40s
pod/mydeploy-6bd88977d5-5kxgv   0/1     Pending   0          25s
pod/mydeploy-6bd88977d5-9ws6c   0/1     Pending   0          10s
pod/mydeploy-6bd88977d5-d88n4   1/1     Running   0          40s
pod/mydeploy-6bd88977d5-qbf4p   0/1     Pending   0          25s
pod/mydeploy-6bd88977d5-r7dmt   1/1     Running   0          25s
pod/mydeploy-6bd88977d5-rnfs2   1/1     Running   0          60m
pod/mydeploy-6bd88977d5-tmk8d   1/1     Running   0          40s
pod/mydeploy-6bd88977d5-w82v2   0/1     Pending   0          10s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   4d21h

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy   5/10    10           5           60m

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeploy-6bd88977d5   10        10        5       60m

NAME                                                  REFERENCE              TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
horizontalpodautoscaler.autoscaling/mydeploy          Deployment/mydeploy    249%/20%   1         10        7          18m
```

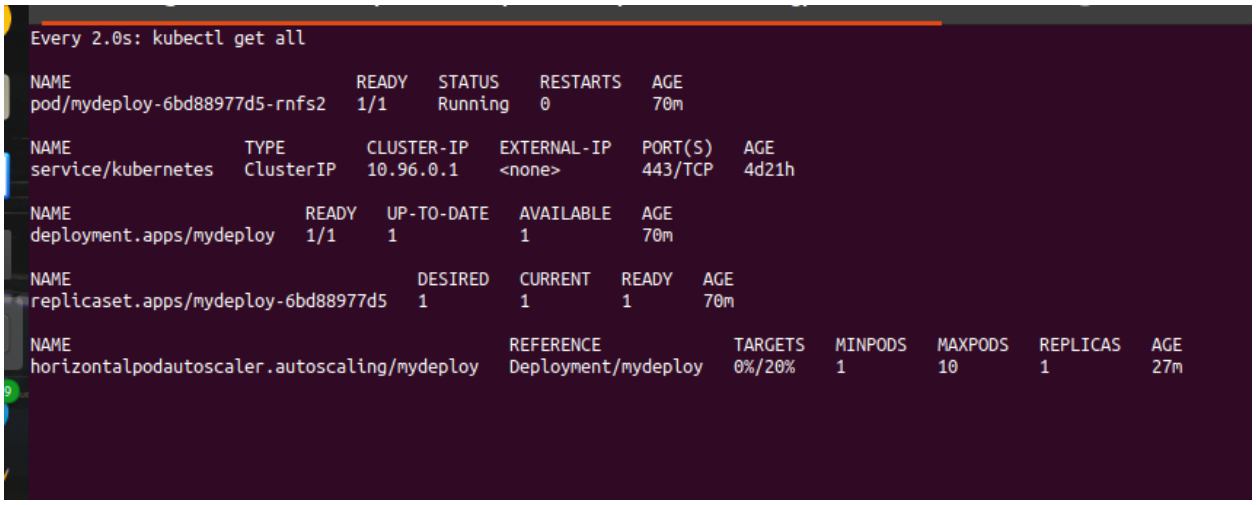
- cpu utilization increased by 249 % and new pod is generating

FInally Horizonal pod autoscaling is come in operation.

==========

what if load remove, ??

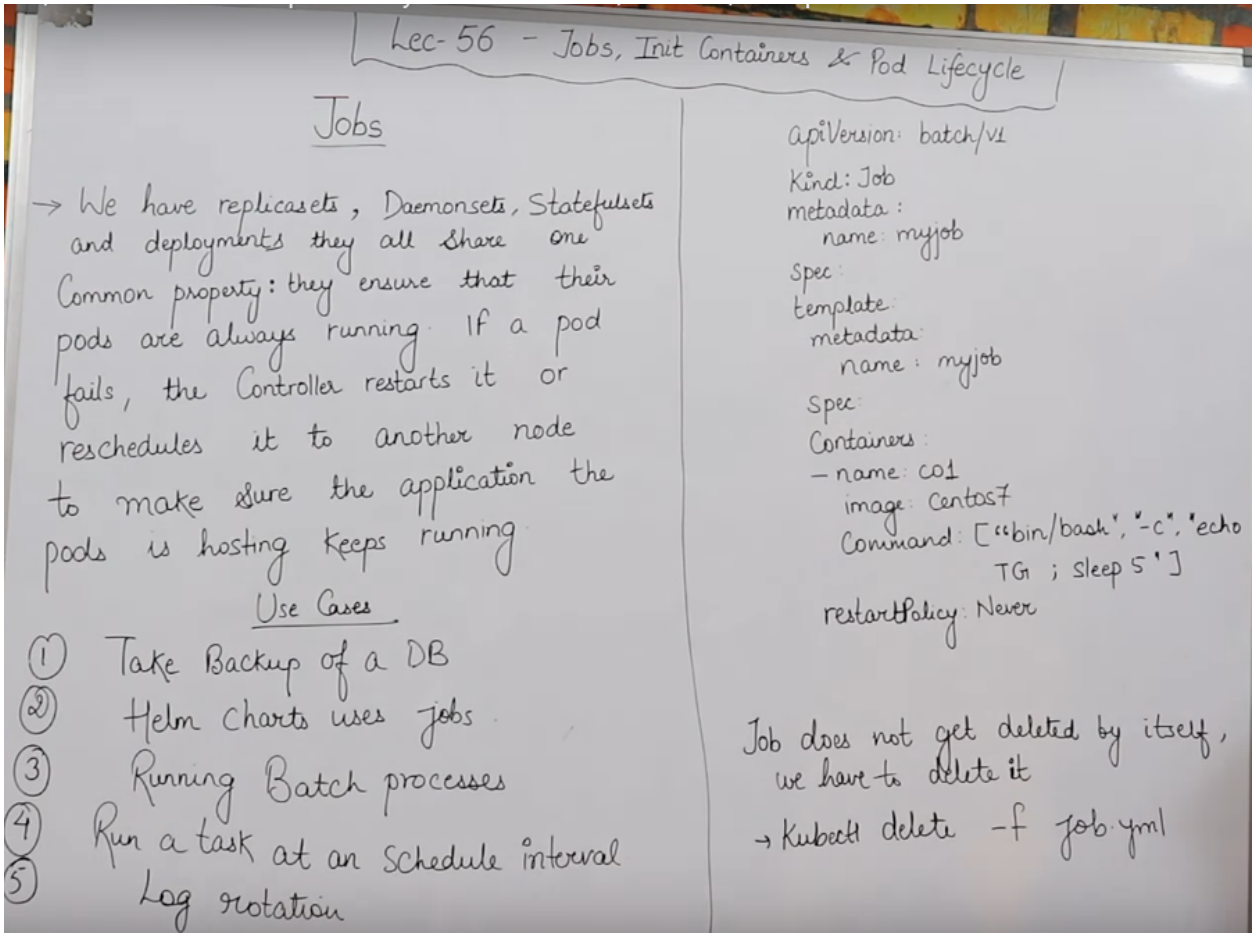
- it will start downscaling pods



Point	Description	Default Value
Evaluation Interval	The interval at which the HPA checks the metrics to determine if scaling is required	15 seconds
Scaling Interval	The interval at which the HPA adjusts the number of replicas based on the observed metrics and target CPU utilization	30 seconds
Downscale Stabilization Period	The period after downscaling the replicas during which the HPA waits before further downscaling	5 minutes

==========

Jobs Object:



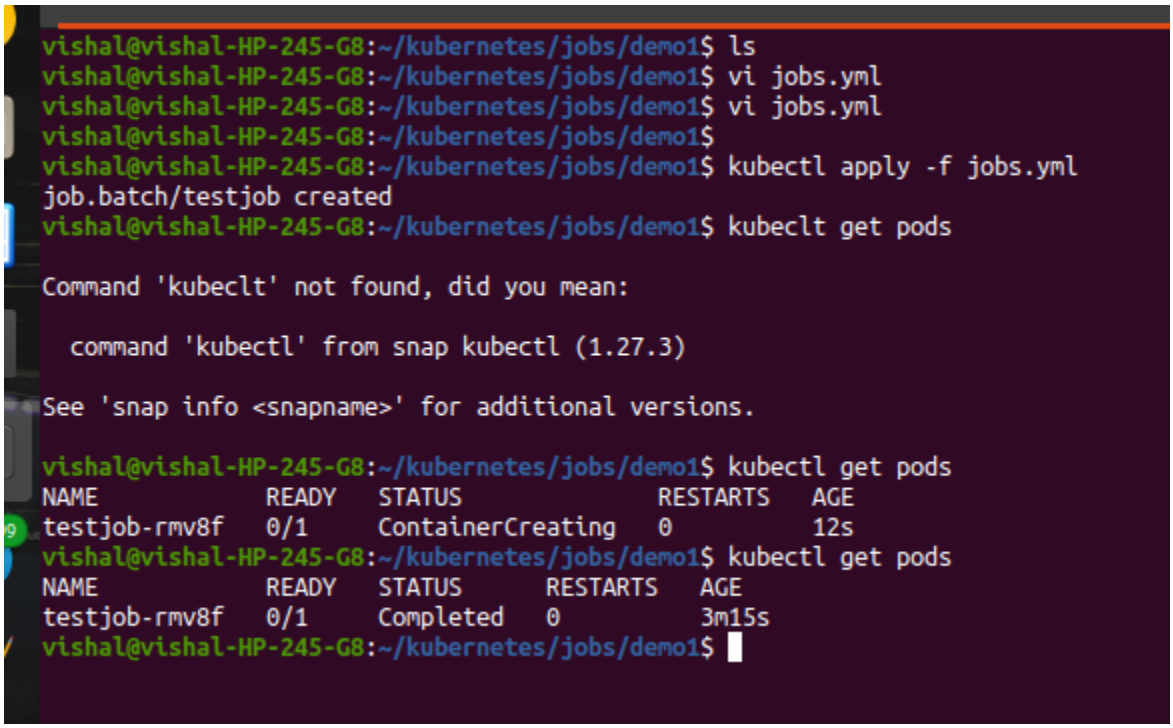
demo 1 ——————————————————>

- vi jobs.yml , save, exit, apply

```
apiVersion: batch/v1
kind: Job
metadata:
  name: testjob
spec:
  template:
    metadata:
      name: testjob
    spec:
      containers:
      - name: counter
```

==========

```
        image: centos:7
        command: ["bin/bash", "-c", "echo vishalk17; sleep 5"]
      restartPolicy: Never
```



- in above image, container stop because work inside it has been completed thus show status as completed

==========

demo2 parallelism ——————————>

- vi job2.yml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: testjob
spec:
  parallelism: 5             # Runs for pods in parallel
  activeDeadlineSeconds: 10  # Time in which it has to completed tASK else terminate
  template:
    metadata:
      name: testjob
    spec:
      containers:
      - name: counter
        image: centos:7
        command: ["bin/bash", "-c", "echo vishal; sleep 20"]
      restartPolicy: Never
```

- save, exit, apply

- **Name:** The name of the Job is `testjob`.
- **Parallelism:** The parallelism of the Job is set to 5, which means that 5 pods will be created in parallel to execute the Job.
- **ActiveDeadlineSeconds:** The activeDeadlineSeconds of the Job is set to 10, which means that the Job will be terminated if it takes longer than 10 seconds to complete.
- **Template:** The template of the Job defines the pod that will be created to execute the Job. The pod will have a single container named `counter` that will run the command `echo vishal; sleep 20`. The `sleep 20` command will cause the container to sleep for 20 seconds, which will give the Job enough time to complete.
- **RestartPolicy:** The restartPolicy of the Job is set to Never, which means that the Job will not be restarted if it fails.

==========

```
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ ls
job2.yml
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ kubectl get pods
No resources found in default namespace.
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ kubectl apply -f job2.yml
job.batch/testjob created
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ kubectl get pods
NAME            READY   STATUS              RESTARTS   AGE
testjob-2jzs7   0/1     ContainerCreating   0          2s
testjob-cxrs8   0/1     ContainerCreating   0          2s
testjob-h45fb   0/1     ContainerCreating   0          2s
testjob-lqzlr   0/1     ContainerCreating   0          2s
testjob-md8rr   0/1     ContainerCreating   0          2s
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
testjob-2jzs7   1/1     Running   0          4s
testjob-cxrs8   1/1     Running   0          4s
testjob-h45fb   1/1     Running   0          4s
testjob-lqzlr   1/1     Running   0          4s
testjob-md8rr   1/1     Running   0          4s
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ kubectl get pods
NAME            READY   STATUS        RESTARTS   AGE
testjob-2jzs7   1/1     Terminating   0          10s
testjob-cxrs8   1/1     Terminating   0          10s
testjob-h45fb   1/1     Terminating   0          10s
testjob-lqzlr   1/1     Terminating   0          10s
testjob-md8rr   1/1     Terminating   0          10s
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ kubectl get pods
NAME            READY   STATUS        RESTARTS   AGE
testjob-2jzs7   1/1     Terminating   0          15s
testjob-cxrs8   1/1     Terminating   0          15s
testjob-h45fb   1/1     Terminating   0          15s
testjob-lqzlr   1/1     Terminating   0          15s
testjob-md8rr   1/1     Terminating   0          15s
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$ kubectl get pods
No resources found in default namespace.
vishal@vishal-HP-245-G8:~/kubernetes/jobs/demo2-parallel$
```

==========

Cronjob in k8s :

A CronJob in Kubernetes is a way to schedule jobs to run on a recurring basis. CronJobs are similar to cron jobs in Unix, but they are more powerful and flexible. CronJobs can be used to run any type of Kubernetes job, including Pods, Deployments, and StatefulSets.

To create a CronJob, you need to create a YAML file that defines the CronJob. The YAML file should include the following properties:

- **Name:** The name of the CronJob.
- **Schedule:** The schedule at which the CronJob should run. The schedule is specified in cron format.
- **JobTemplate:** The JobTemplate that defines the job that should be run.

```
# ┌──────────────── minute (0 - 59)
# │ ┌──────────────── hour (0 - 23)
# │ │ ┌──────────────── day of the month (1 - 31)
# │ │ │ ┌──────────────── month (1 - 12)
# │ │ │ │ ┌──────────────── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                    7 is also Sunday on some systems)
# │ │ │ │ │                                    OR sun, mon, tue, wed, thu, fri,
sat
# │ │ │ │ │
# * * * * *
```

==========

demo1

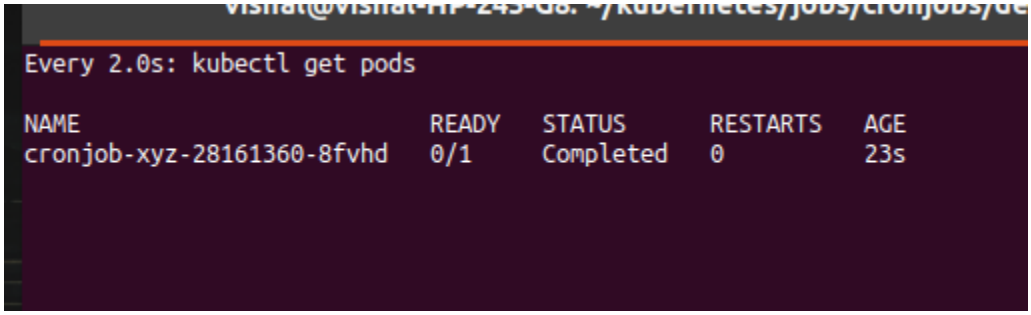
```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: cronjob-xyz
spec:
  schedule: "* * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - image: ubuntu
              name: c00
              command: ["/bin/bash", "-c", "echo vishal; sleep 5"]
          restartPolicy: Never
```

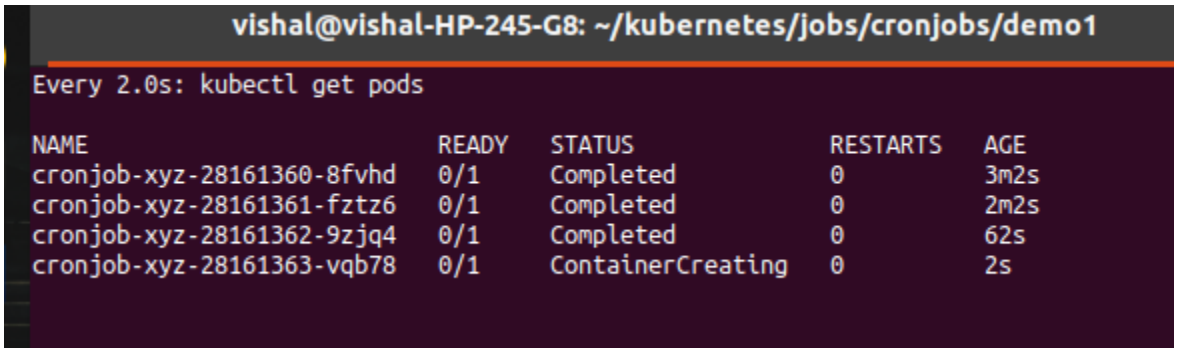
- save, exit & apply
- As per the above script, the container stops after 5 seconds because its work is done (`restartPolicy: Never`). Every minute, a new pod is created, and the same process will repeat. This is because of the cron job schedule in the script (* * * * *) .

==========

open 2nd terminal and hit below command, so that we can monitor the progress of cronjob

watch kubectl get all … continuously monitor the status of all resources in Kubernetes





==========

Init Container

What is an init container?

An init container is a container that runs before the main container in a Pod. It is used to perform initialization tasks or setup actions that are required by the main container.

Why use init containers?

Init containers can be used to perform a variety of tasks, such as:

- Installing dependencies
- Configuring settings
- Performing pre-processing

- The main container can be an application, while the init container can handle tasks like installing dependencies, configuring settings, or performing pre-processing.
- The init container runs to completion, and then the main container starts.
- If the init container fails, Kubernetes will repeatedly restart it until it succeeds, ensuring that the initialization process is successful before running the main container.
- Once the main container is running, the init container is no longer needed and is automatically deleted.
- Init containers provide a way to ensure that certain prerequisites are met before the main container starts.

==========

demo1 —---------->

vi init.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: initcontainer-pod
spec:
  initContainers:
    - name: c00-init-container
      image: centos
      command: ["/bin/sh", "-c", "echo LIKE AND SUBSCRIBE t.me/vishalk17 >
/tmp/xchange/testfile; sleep 30"]
      volumeMounts:
        - name: xchange
          mountPath: "/tmp/xchange"
  containers:
    - name: c01-main-container
      image: centos
      command: ["/bin/bash", "-c", "while true; do echo `cat
/tmp/data/testfile`; sleep 5; done"]
      volumeMounts:
        - name: xchange
          mountPath: "/tmp/data"
  volumes:
    - name: xchange
      emptyDir: {}
```

- save, exit , apply

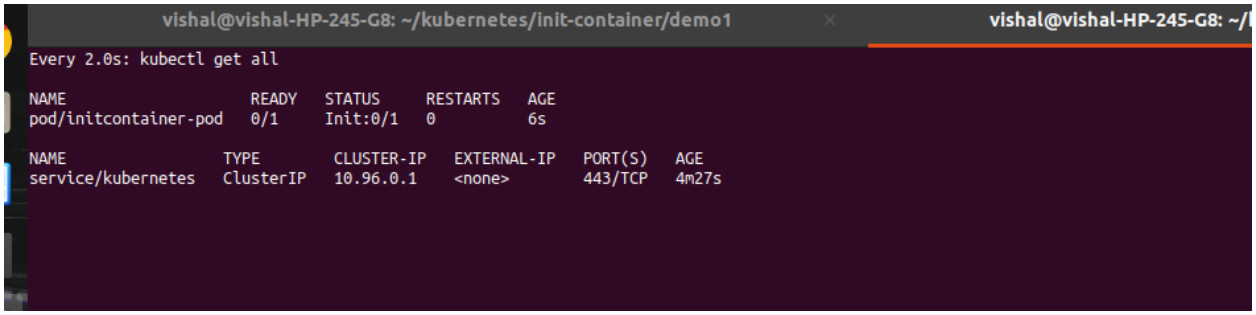
The YAML file defines a Pod that has two containers: an init container and a main container. The init container is responsible for creating a file called `testfile` in the `/tmp/xchange` directory. The main container then reads the contents of the `testfile` file from `/tmp/data` dir and prints them to the console.

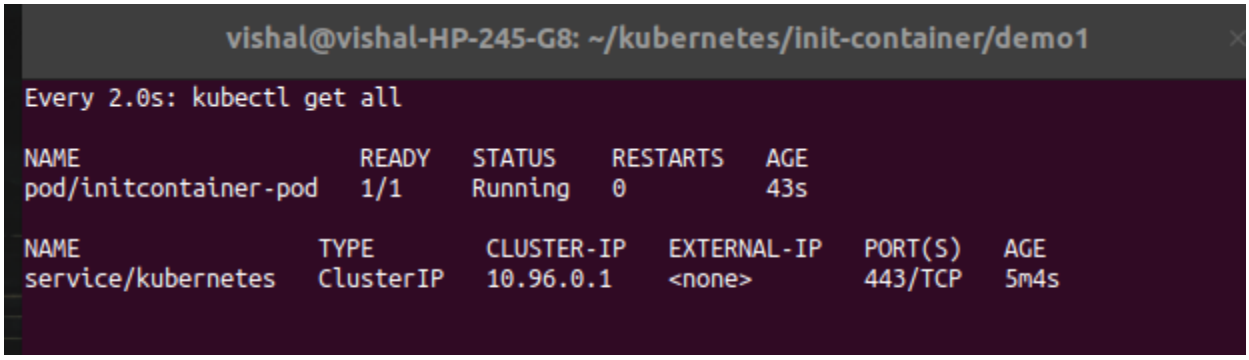
==========

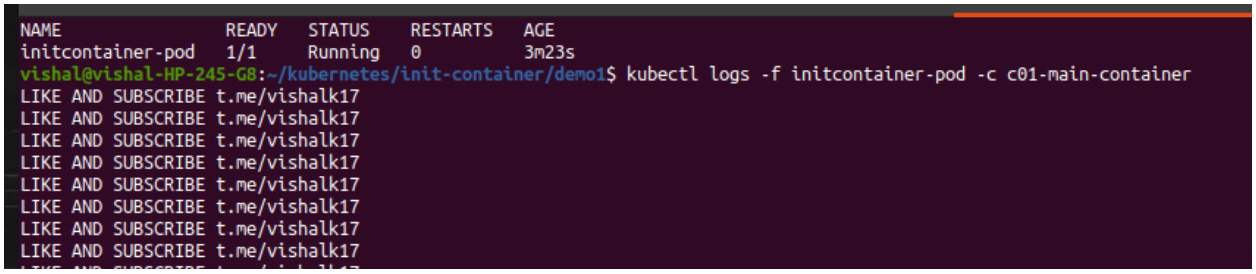
The `emptyDir` volume is a temporary volume that is created and create mounting point `xchange`.The `xchange` volume is used to share data between the init container and the main container.

open 2nd terminal and hit below command, so that we can monitor the progress of init container

watch kubectl get all … continuously monitor the status of all resources in Kubernetes







– as you can see the working of init container

==========

Sourcecode :

- https://github.com/vishalk17/devops/tree/main/kubernetes

My devops repo :

- https://github.com/vishalk17/devops

My telegram channel:

- https://t.me/vishalk17_devops

Contact:

Telegram : t.me/vishalk17

vishalk17 My youtube Channel :

- YouTube https://www.youtube.com/@vishalk17

References:

I followed the Technical Guffugu youtube channel for Kubernetes series to learn. You can check it out from the link below. I am mentioning the initial video link of Kubernetes series, so that you can land to the video lectures.

- YouTube https://youtu.be/mYVzuE3daY8

Technical Guftugu Youtube channel :

- YouTube https://www.youtube.com/@TechnicalGuftgu