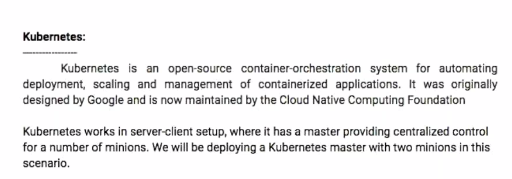
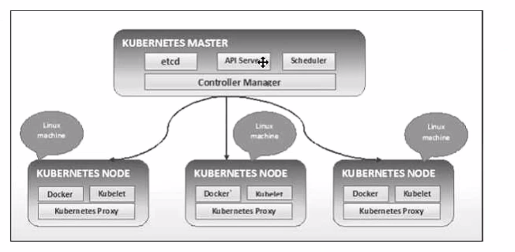
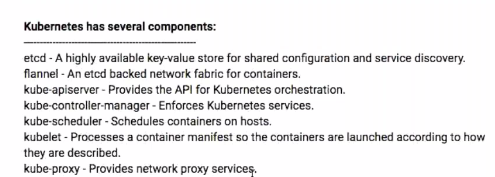
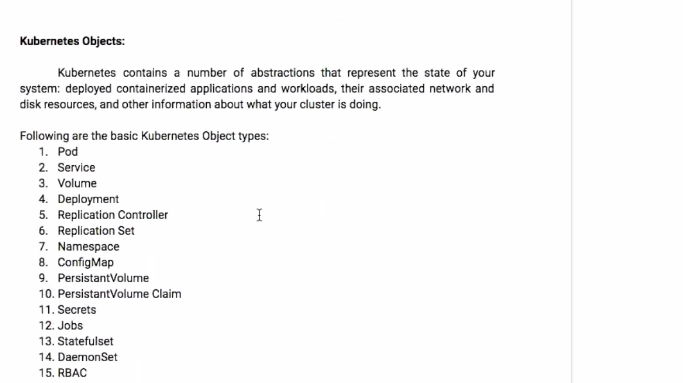
KUBERNETES:





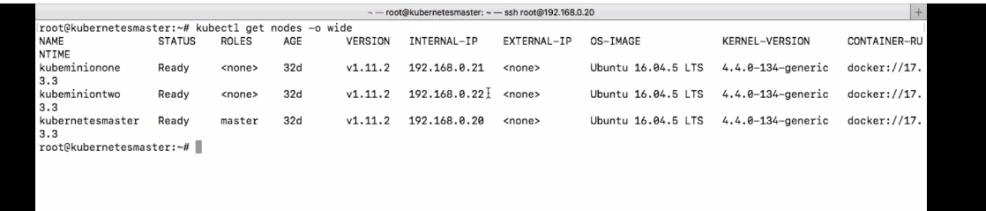




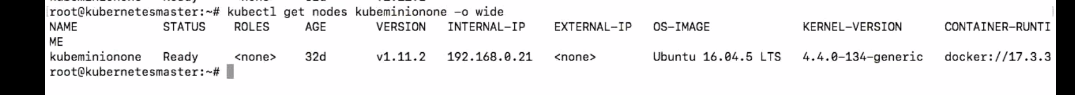
How To check the what are the nodes connected in Kubernetes-master.



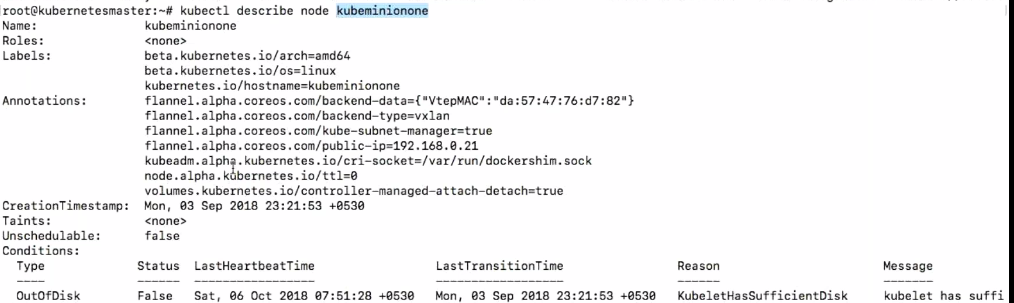
How to get the node details:



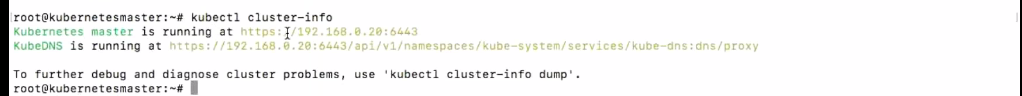
How to check the particular node information:



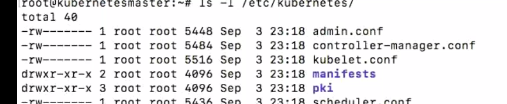
How to check full details of Particular node information:



How to check cluster information:



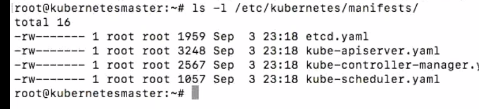
Home path(Kubernetes):



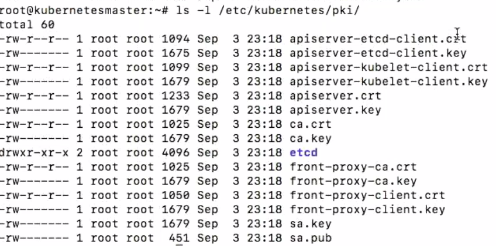
Kubernetes Admin config information:



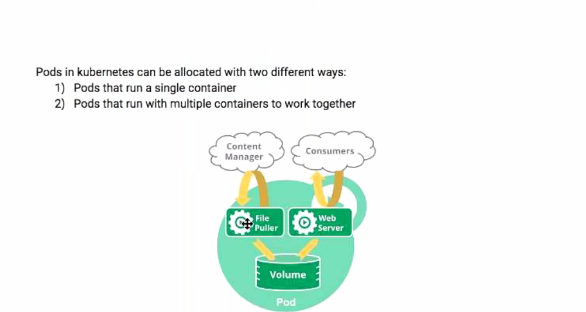
All yaml path(manifests):



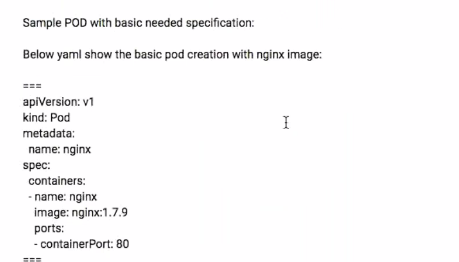
All certificates path:



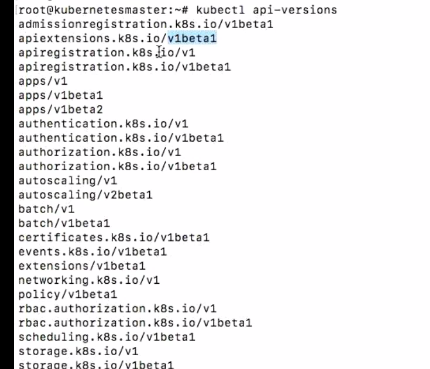
POD:



Basic Pod yaml file(Single container EX):



How to check the API-Versions:



NameSpaces:

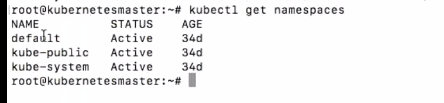
General name spaces are available 3types.

1.kube-system

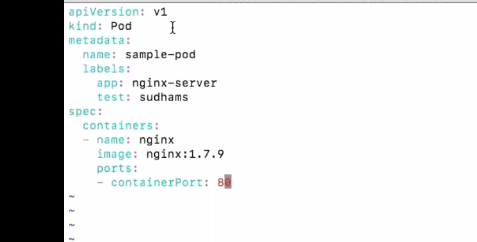
2.default

3.kube-public

How to check the namespaces details by using the command.



Sample Pod creation:



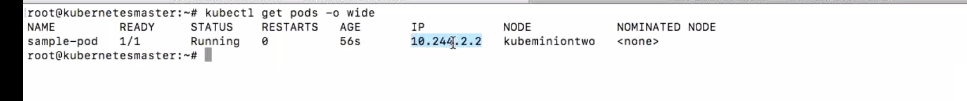
Next, we want execute above pod.

**Kubectl apply-f sample-nginx.yml**

Next, we want check status of the pod:

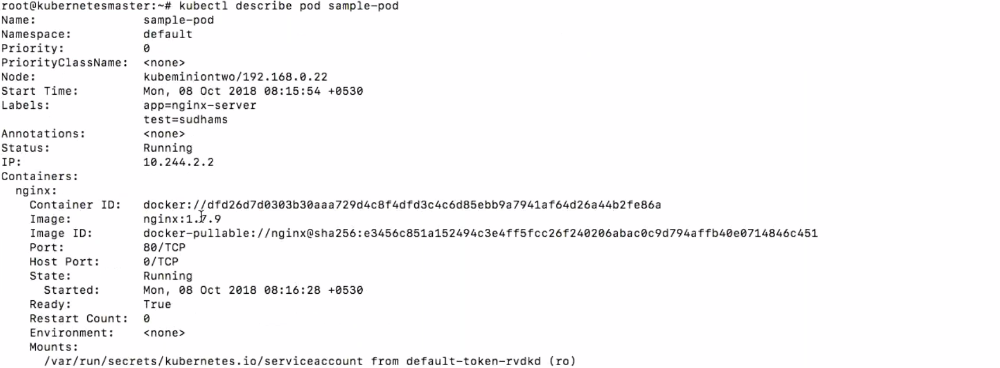


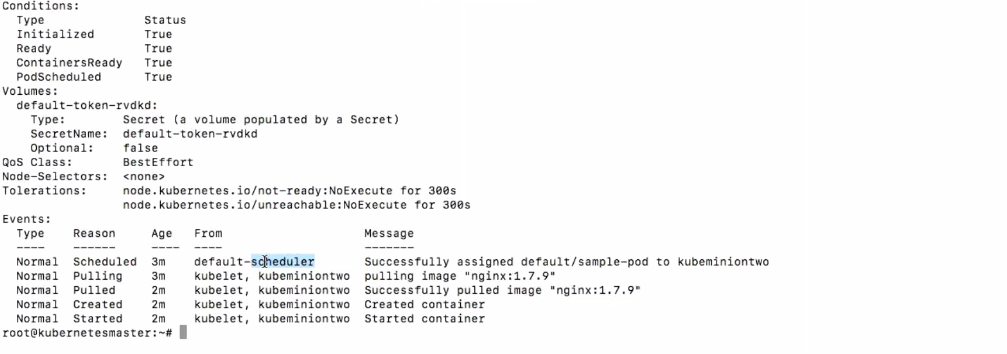
I want know the what is the IP Address of this running pod:



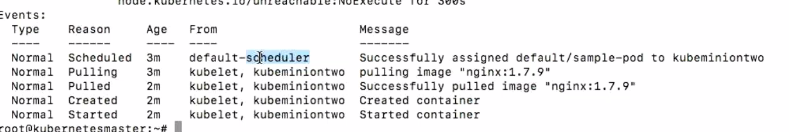
Next, we want know the complete details of this pod:

**Kubectl descricube pod sample-pod:**

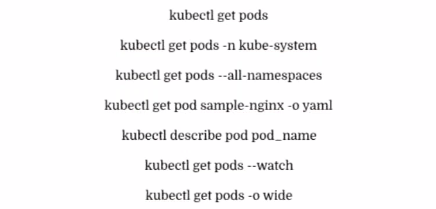
****

****

**Lifecycle:**

****

**POD Commands:**

****

**Diff b/w the Kubectl create vs kubectl apply?**

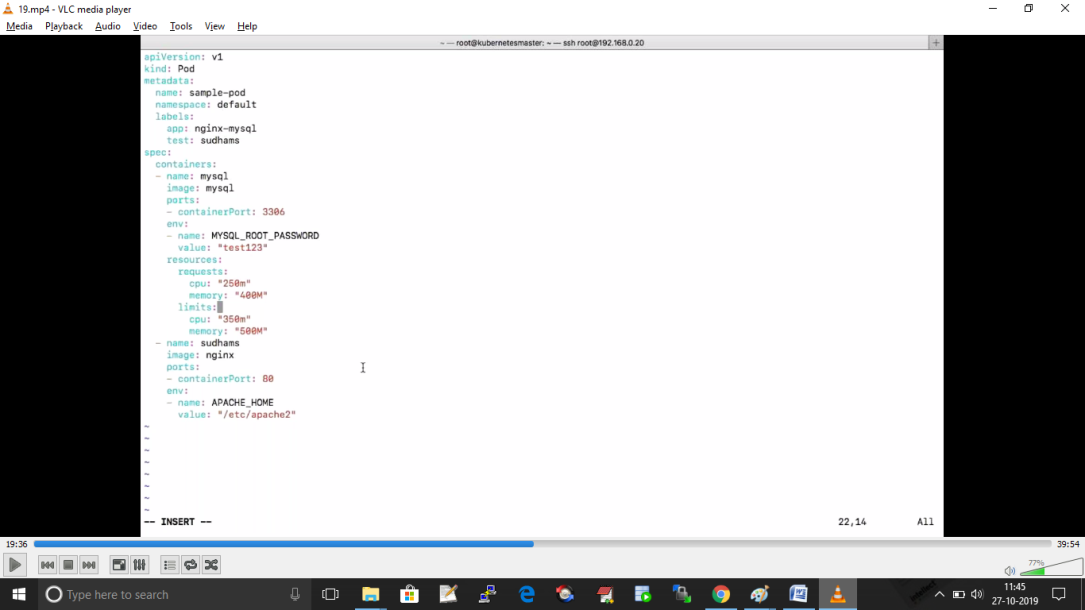
Kubectl create:

Here we can create pod only,we can’t apply updates.

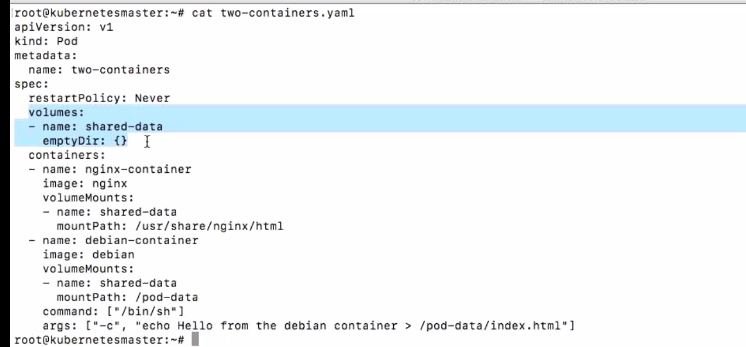
Kubectl Apply:

Here we can create the pods ,after apply the updates also.

**Now Multiple Container Concept in Pod**:

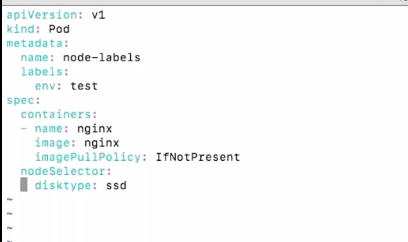


Two Pods are using the Same volume:



I want run the pad particular node(using the label):





INIT CONTAINER CONCEPT:

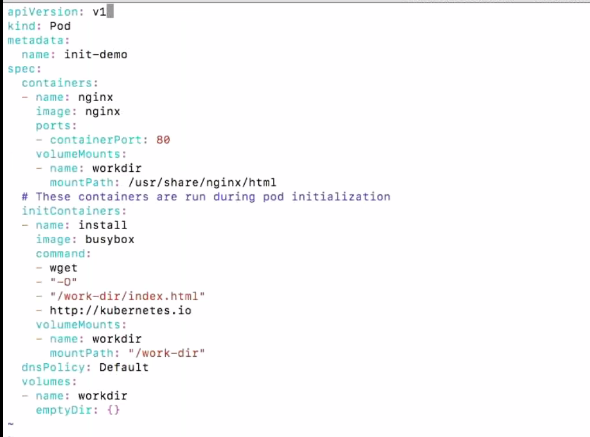
This page provides an overview of init containers: specialized containers that run before app containers in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/). Init containers can contain utilities or setup scripts not present in an app image.

A [Pod](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/) can have multiple containers running apps within it, but it can also **have one or more init containers**, which are run before the app containers are started.

Init containers are exactly like regular containers, except:

* Init containers always run to completion.
* Each init container must complete successfully before the next one starts.

If a Pod’s init container fails, Kubernetes repeatedly restarts the Pod until the init container succeeds. However, if the Pod has a restartPolicy of Never, Kubernetes does not restart the Pod.



**Volumes:**

Mainly 2 types volumes.

1. Ephemeral-Volume
2. Persistent-Volume

Ephemeral-Volume:

On-disk files in a Container are ephemeral, which presents some problems for non-trivial applications when running in Containers. First, when a Container crashes, kubelet will restart it, but the files will be lost - the Container starts with a clean state.

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: redis  spec:  containers:  - name: redis  image: redis  volumeMounts:  - name: redis-storage  mountPath: /data/redis  volumes:  - name: redis-storage  emptyDir: {} |

1. 1.Create the Pod:

kubectl apply -f https://k8s.io/examples/pods/storage/redis.yaml

1. 2.Verify that the Pod’s Container is running, and then watch for changes to the Pod:

kubectl get pod redis --watch

3.The output looks like this:

NAME READY STATUS RESTARTS AGE

redis 1/1 Running 0 13s

1. In another terminal, get a shell to the running Container:

kubectl exec -it redis -- /bin/bash

1. In your shell, go to /data/redis, and then create a file:
2. root@redis:/data# cd /data/redis/

root@redis:/data/redis# echo Hello > test-file

1. In your shell, list the running processes:
2. root@redis:/data/redis# apt-get update
3. root@redis:/data/redis# apt-get install procps

root@redis:/data/redis# ps aux

The output is similar to this:

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

redis 1 0.1 0.1 33308 3828 ? Ssl 00:46 0:00 redis-server \*:6379

root 12 0.0 0.0 20228 3020 ? Ss 00:47 0:00 /bin/bash

root 15 0.0 0.0 17500 2072 ? R+ 00:48 0:00 ps aux

1. In your shell, kill the Redis process:

root@redis:/data/redis# kill <pid>

where <pid> is the Redis process ID (PID).

1. In your original terminal, watch for changes to the Redis Pod. Eventually, you will see something like this:
2. NAME READY STATUS RESTARTS AGE
3. redis 1/1 Running 0 13s
4. redis 0/1 Completed 0 6m

redis 1/1 Running 1 6m

At this point, the Container has terminated and restarted. This is because the Redis Pod has a[restartPolicy](https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.16/#podspec-v1-core) of Always.

1. Get a shell into the restarted Container:

kubectl exec -it redis -- /bin/bash

1. In your shell, go to /data/redis, and verify that test-file is still there.
2. root@redis:/data/redis# cd /data/redis/
3. root@redis:/data/redis# ls

test-file

1. Delete the Pod that you created for this exercise:

kubectl delete pod redis

Persistent Volume:

It is perment storage.if restart the pod nothing effect on the volume.if the delete pod meaning volume also deleted.

EX:

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv0003

spec:

capacity:

storage: 5Gi

volumeMode: Filesystem

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Recycle

storageClassName: slow

mountOptions:

- hard

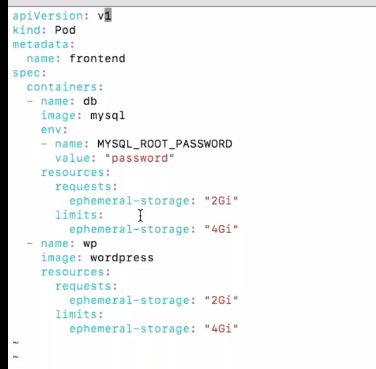
- nfsvers=4.1

nfs:

path: /tmp

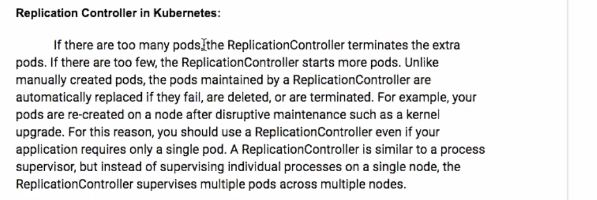
server: 172.17.0.2

Resource Limit Example:

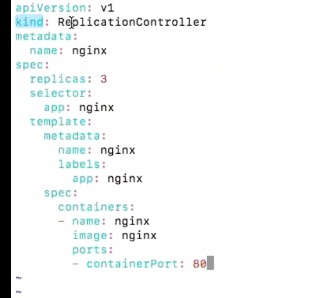


**Replication Controller**:

It is mainly user for the auto-scaling.



Ex:



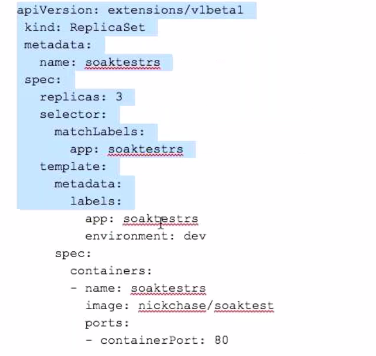
In above example replicas:3 means it will create 3 nginx servers.If any one down automatically up the second one.

**Replication-Set:**

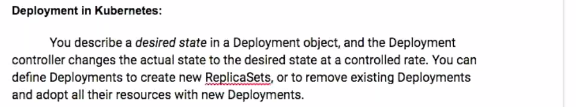
Replication-Controller and Replication-Set both are same only. Except one functionality



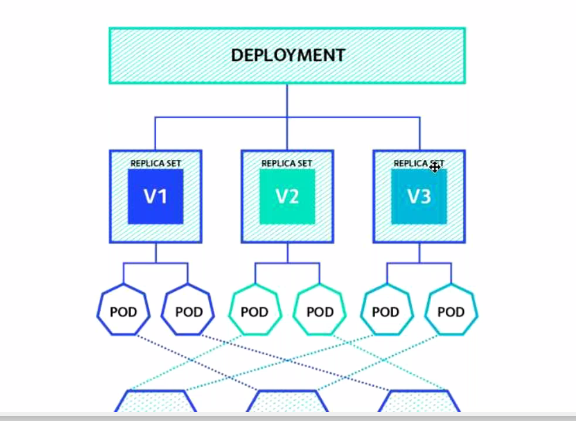
Ex:



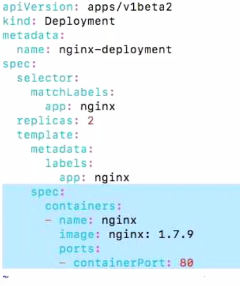
**Deployment :**



Deployment-Structure:



Depolyment.yml:

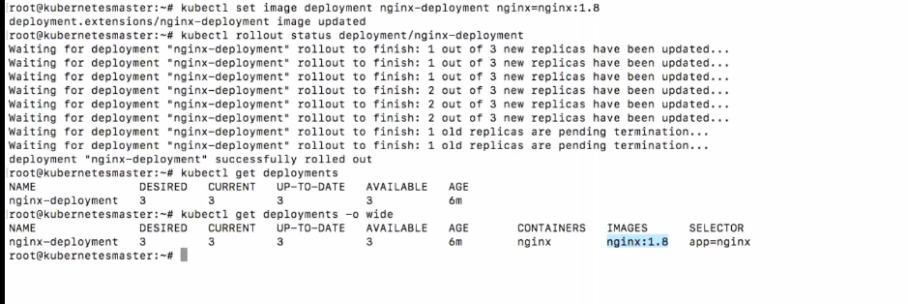


Mainly Deployment work is update the lower versions to higher versions and reversely also do.

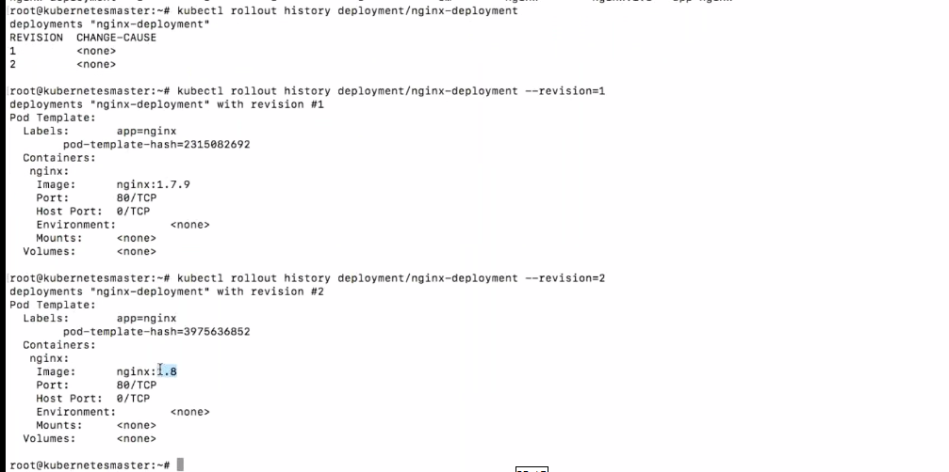
Like i want upgrade to nginx server version:1.7.8 to 8.0 version.

I want degrade to 8.0 to 1.7.8 version.

Upgrade:

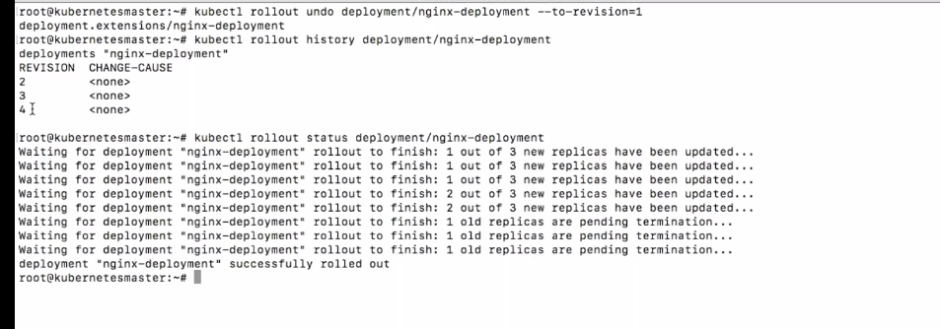


Upgrade History:

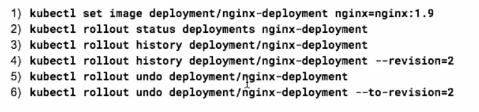


Downgrade:

I want come to my old version of nginx that time we are using the rollback(undo).



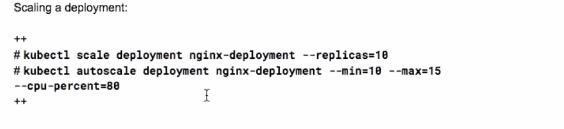
Upgrade and Downgrade related Commands:

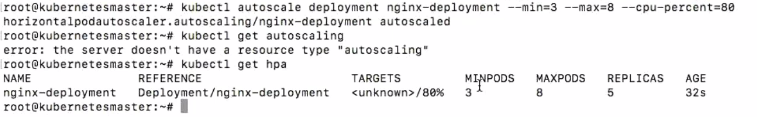


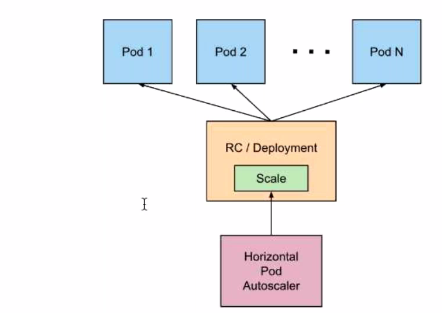
Scale:



Auto-Scaling:





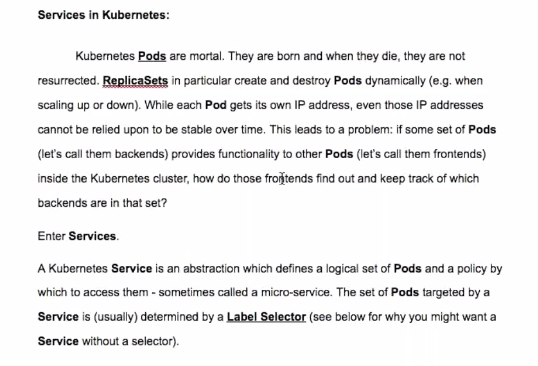


Resume and pausing the deployment:

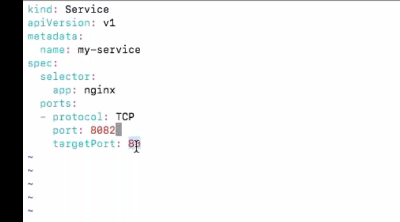


**Services:**

Services just like grouping the multiple pods together.

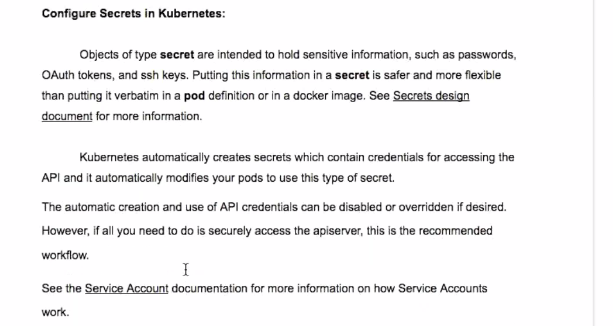


EX:



Services is nothing but group of pod. Each pod having the IP Address.

Secrets:



Main concept of the Secrets is save environment information in encrypted format.

For Ex::

I have one env details like admin / test123.

I want put above details encrypted format.

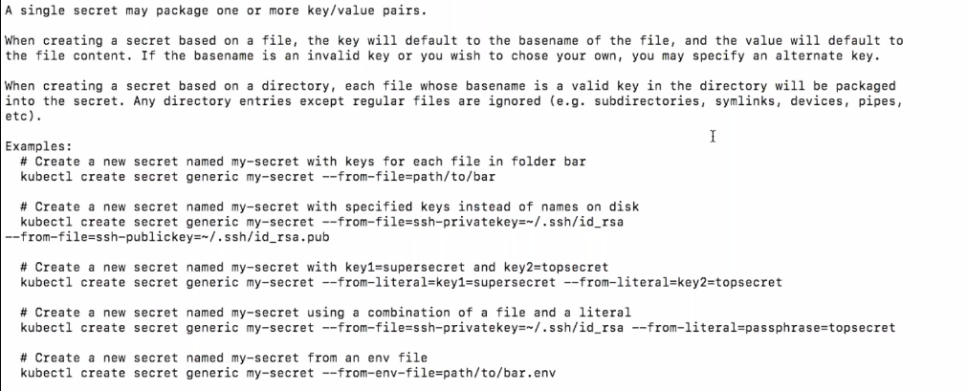
KS8 provides the 3 types of secrets .

1.docker-registry

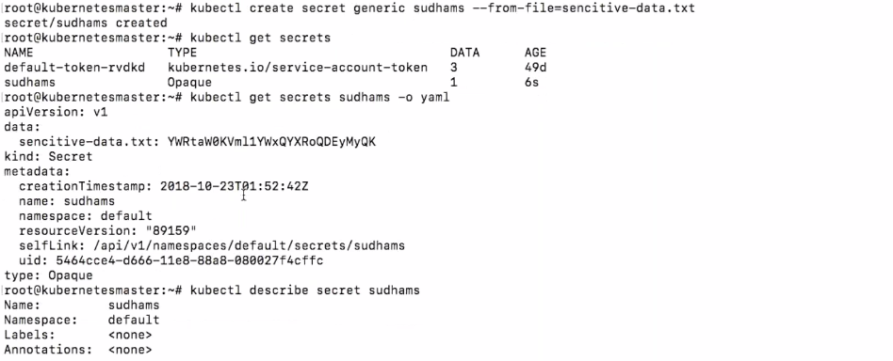
2.generic

3.tls

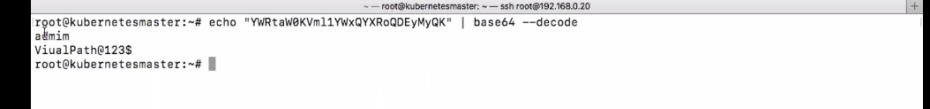




How to create the secrets:

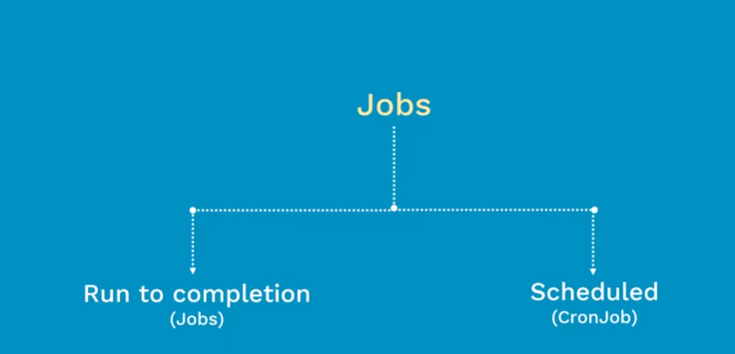


How to decode the secrets:



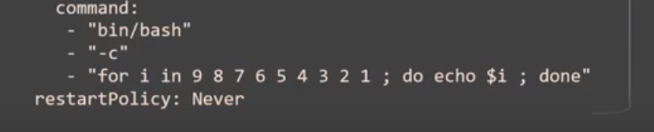
JOBS:

A Job creates one or more Pods and ensures that a specified number of them successfully terminate. As pods successfully complete, the Job tracks the successful completions. When a specified number of successful completions is reached, the task (ie, Job) is complete. Deleting a Job will clean up the Pods it created.



Job Example:





After Create the pod.we have to check the some details.



