Kubernetes by google

Reference links

<https://kubernetes.io/docs/reference/kubectl/cheatsheet/>

<https://www.scaler.com/topics/kubernetes/>

<https://kube.academy/courses> to learn kubernetes

here it have kubernetes codes,like autoscaling

<https://github.com/microservices-demo/microservices-demo/tree/master/deploy>

What is kubernetes?????????????????????????????????????????????????????????????

Kubernetes is a container management/orchestration tool, where the process of deploying, scaling containers is automated, here based on load pods/containers will be auto scaled

Kubernetes is full of objects

2 containers can communicate with each other

1. **Why kubernetes and advantages**
2. Dynamic auto scaling ,ex:- if ur appln cpu is 90% utilized with kubernetes u can create more instances automatically , based on the load we can scale up the number of instances, if cpu utilization is less then we can automatically scale down this is possible with an object called – HorizontalPodAutoscaler and pods will be deleted if nodes are not there, this is called auto scale down,
3. Lets say we created a container and in production if it dies? What we have to do, we should again start it, but Here in kubernetes we have objects like replicaset which will ensure always 3 or n instances are always running, even if u delete 1 replica also, always it will ensures desired 3-n replicas are always present. even if u delete also, automatically pod objects will be created dynamically
4. Horizontal scaling, if load increased, then replicas will be increased automatically Here ur application will be mostly available,

Image, container both are docker inspectable objects

Network is also a docker object—which is inspectable

docker inspect <container-name/id>

2)kub architecture

In kub cluster, we have a below components called

In real time, every environment will have separate cluster

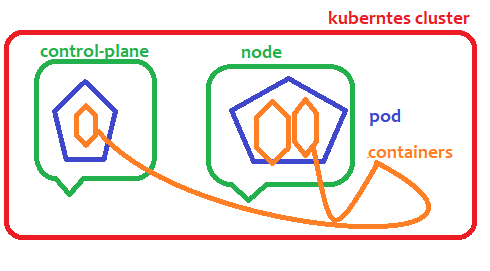
Ex:- dev environment cluster is separate and prod env cluster is separate,

If u want to interact with kube cluster, then we have to use “kubectl.exe”

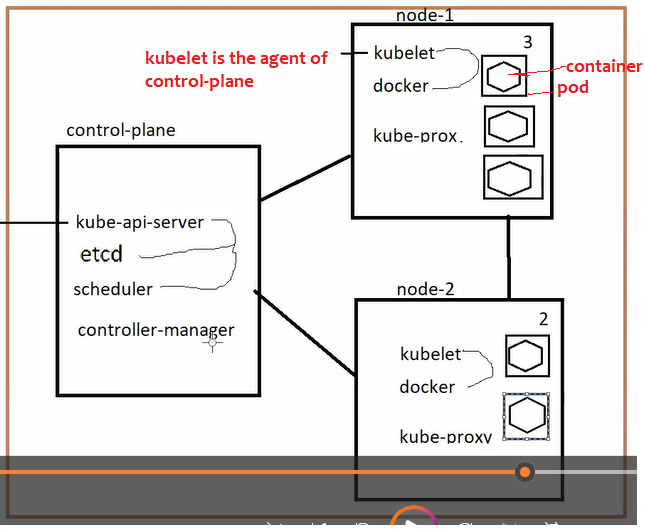
1)control plane-which doesn’t carry any load, hence it wont have any container,hence docker also is absent

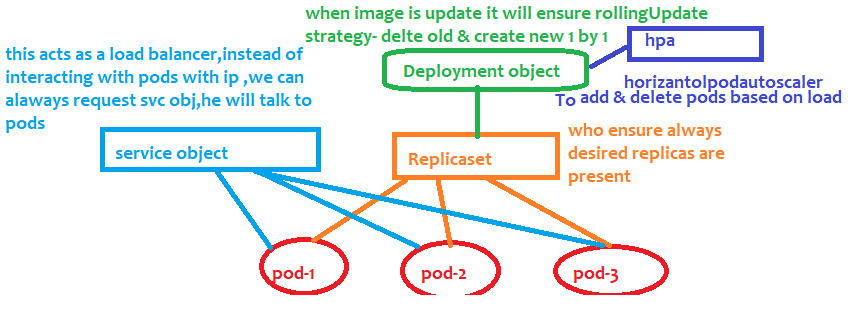
2) node-usually means, 1 computer /1 virtual machine

Every node and every pod will be having an ip address



Node – node is nothing but a physical computer or virtual machine





What Is pod

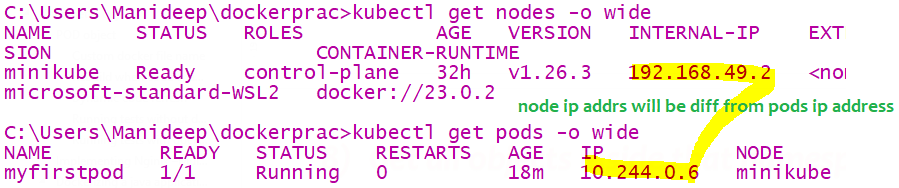
Pod is the way to collectively deploy 2 containers.

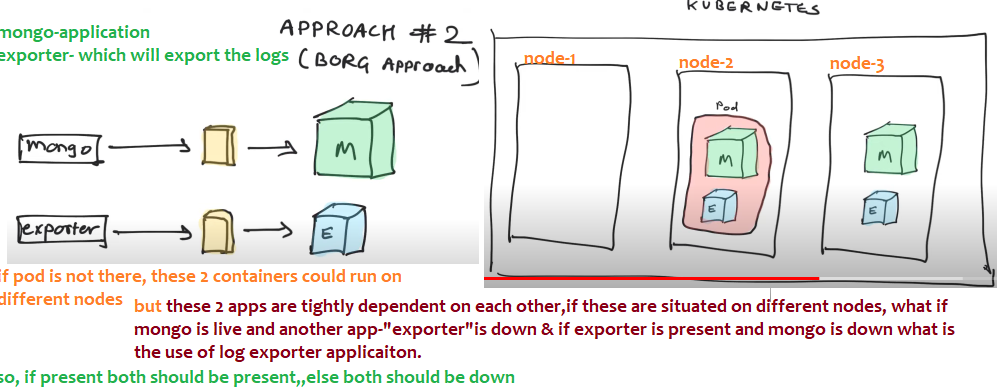
Therefore if 2 apps are tightly dependent, instead of deploying them as separate containers in separate vms

Better deploy them in same pod , always that single pod reside in single node,pod is a packaged,

Every node in a cluster will have some ip address and

Every pod inside a node will have some ip address





U can delete the entire pod or u can up the entire pod-

POD means just an abstract layer where containers will reside

Pod is a smallest deployable unit, in kub we can deploy only till pod, we cant directly start a container

1 node can have many pods,it is a shell around a container

1 pod can have 1 or more containers,which could be tightly dependent

Kub suggests, even if 1 container is there run that in a pod,

Lets say there are 2 images we created 2 containers for these 2 applications, these 2 are tightly dependent with each other

1) Components of control-plane

This control plane component –similar to master node,but it doesn’t carry any load ,it doesn’t have any containers

since it is not going to have containers ,control plane component doesn’t have docker installed ,

it doesn’t have any pods, and

1)**kube-api-server :-** this will receive all the incoming commands and it will validate all those commands

And it will send those validated commands to etcd

2)**etcd**:- it a database, all the state/information about entire cluster will be stored here,

like how many nodes,how many replicas,how many pods are there

in unix “etc” is a folder to store the configuration data for single system,here d means distributed,which stores the configuration for distributed systems

3)**The Scheduler** :- which schedules your apps (assigns a worker node to each deployable component of your application)

4)**The Controller Manager:-** (replica manager/node manages)

Its job is to ensure desired number of replicas and nodes(Virtual machines) are running

which performs cluster-level functions, such as replicating components, keeping track of worker nodes, handling node failures, and so on

THE NODES

The worker nodes are the machines that run your containerized applications. The task of running, monitoring, and providing services to your applications is done by the following components:

Docker, rkt, or another container runtime, which runs your containers

The Kubelet, it is the agent of control-plane node, which talks to the API server and manages containers on its node

Kubelet will talk to docker container and it will create the containers inside a pod

The Kubernetes Service Proxy (kube-proxy), it is to assign the ip-addresses, which load-balances network traffic between application components

In AWS- we have EKS-called managed cluster- Elastic kubernetes service-here entire kubernetes cluster is ready

Cluster with-control plane and kubernetes node-where those are preinstalled with docker

Kubernetes commands

1) Kub cluster commands

#### Kubectl == kube controller is a CLI tool for kubernetes

#### Get cluster info

kubectl cluster-info

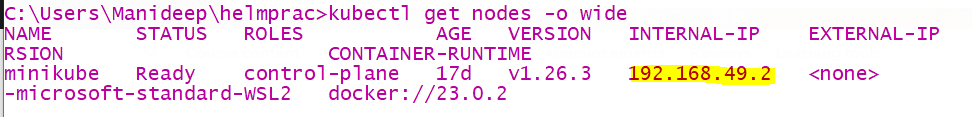
To get nodes & ips

Node is nothing but a separate machine, where kubernetes and docker are installed

kubectl get nodes

if u want the ipaddress of all the nodes – “kubectl get nodes –o wide”

or if u type “minikube ip” u will get the ip address of that node



Minikube

In local, we will install a software called minikube(where docker software is pre-requisite ), minikube will give us

The single node cluster,

Kubernetes-cluster of nodes

Minikube- a cluster with single node

In production –managed solutions are

1. amazon EKS-Elastic kubernetes service
2. GKE- Google cloud kubernetes engine
3. On prime data centre- means cluster is created and managed by our own company

After installing check whether kubernetes installed or not using “kubectl version”

Basic commands

1. kubectl api-versions

//to see all the versions like v1…

1. how to come back from a container

suppose if u typed “kubectl exec –it <pod-name> -- bash”, then u will inside that container, then if u want to come back

just type Ctrl+PQ

1. Executing yaml file

Kubectl means- kubernetes controller

We are going to interact with kubernetes clusters component called “control-plane”

Lets say if the file “myapp.yaml” file contains pod create script,If u type “kubectl create”-it will create the pod

If u type “kubectl delete” it will delete the pod

|  |  |
| --- | --- |
| If u are creating objects then use create command | kubectl create –f ./<filename.YAML>  kubectl create –f ./myapp..yml |
| To apply all changes | kubectl apply –f ./myapp.yaml |
| To delte the objects | kubectl delete –f ./myapp.yaml |
|  |  |
|  |  |

If u have multiple files in current directory, and If u want to execute all at once then use “kubectl apply –f .”

Here dot means current directory

C:\Users\Manideep\dockerprac>kubectl apply -f ./createredispod.yaml

pod/redis created

here yaml file is executed and pod called redis is created

for example if pod creation script is present in myapp.yaml file, then if pod is not there at all then we should use create command

if pod is already present,if u are making any changes the use apply command

1. CRUD -pods-Create,Delete pod

|  |  |
| --- | --- |
| Create a pod | kubectl run manipod --image=nginx  this will create that pod with that container  EX:- kubectl run kubepod --image=bibinwilson/docker-kubectl:latest  This bibinwilson is a special image where kubectl software Is already installed |
| Get pods | kubectl get pods  to see the pod name  //To get pods from another namespace  kubectl get pods -n <namespace name>  kubectl get pods -n charan |
| Get pods from all namespace or another namespace  //Get pods from different namespace being in current namespace  kubectl get pods -n <namespace-name>  ex:-  kubectl get pods -n kube-system  kubectl get pods -n maninamespace | kubectl get pods –A  or  kubectl get pods --all-namespaces |
| Get pods with more info like ip address-if u want to know which pod is on which node along with ip addresses then use –o wide | kubectl get pods –o wide |
| Describe a pod- most famous one-  To get pod information like – 1) how many containers it have and all 2) what are the names of all containers…and 3)what is the pod ip address (both node and pod will have ip address)  4) to know who is controlling this pod-means either replicaset/ replication controller  5)to know all the labels of a pod | kubectl describe pod <pod-name>  ex:-  kubectl describe pod d1-depl-obj-name-7877f6d747-96plb |
| See the source code of pod in YAML format  Here –o means output | Kubectl get pod <pod-name> -o yaml  Ex:-  kubectl get pod d1-depl-obj-name-7877f6d747-96plb -o yaml |
| Go inside a pod of a container | //when u have only single container try this  kubectl exec –it <podname> -- bash  kubectl exec -it charanpod – bash  kubectl exec -it d1-depl-obj-name-7877f6d747 hzd5z -- bash  //means execute this command inside a running container and connect my terminal to the input stream |
| Delete a pod | kubectl delete pod <pod-name> |
| Most useful  Edit a pod- if u want to update the image version or update the labels/scale the replicas we should edit the pod | kubectl edit pod <pod-name> |
| To see who Is controlling this pod –I mean is   1. Replicaset controlling the pod or 2. ReplicationController controlling the pod | kubectl describe pod <pod-name>  ex:-  kubectl describe pod d1-depl-obj-name-7877f6d747-96plb  if u describe u should see value for field “controlled-By” |
| Watching pods -If u want to see pods creation and deletion dynamically | kubectl get pods --watch |
| How to see the logs of a pod | kubectl logs <pod-name> -c //to show container name  kubectl logs <podname> –c <containername>  //if there is no container ,u can ignore –c flag  //if u have only a single container  kubectl logs <pod name>  kubectl logs charanpod |
| How to see metrics like cpu utilization,metrics | kubectl top pods |

Here If u observe, if u describe the pod there is a field called “controlled by”: “replicaset/…” means here

This pod is getting controlled by replicaset



1. Create a pod

Pod can be created in 2 ways

1. Using kubernetes run command (deprecated)

The below command will directly create a pod p1 & run this image inside a container

C:\Users\Manideep\dockerprac>kubectl run p1 --image=nginx

**pod/p1 created**

1. **U**sing declarative approach-file

Write all the commands in a file and execute using

kubectl apply -f ./createredispod.yaml

apiVersion: v1

kind: Pod

metadata:

name: myfirstpod

spec:

containers:

- name: nginc1

image: nginx:latest

Using run command (deprecated)

C:\Users\Manideep\dockerprac>kubectl run p1 --image=nginx

pod/p1 created

##here u used “p1” hence a pod with name p1 is created

C:\Users\Manideep\dockerprac>kubectl get po

NAME READY STATUS RESTARTS AGE

myfirstpod 1/1 Running 0 42m

nginx 1/1 Running 0 81s

p1 0/1 ContainerCreating 0 3s

1. If u want to get all pods inside a particular namespace

kubectl get pods –n <namespace name>

1. To get all the pods inside default namespace

EX:- kubectl get pods -n default

kubectl get pods

means it will try to fetch those pods in default namespace

1. To get pods with more information like ip address or u can describe the pod

here output of pods will be printed in wide, means more output will be printed

**kubectl get pods –o wide**

1. Watching a pod

kubectl get pods --watch

1. Describe a pod

**kubectl describe pod <pod-name>**

kubectl describe pod redis

Lets say if u want to know how many containers are running inside that pod then u have to describe that pod

If u describe u will get all the ip addresses of nodes and pod

Like u will get all the container names inside a pod, image id everything about that pod

There Is a difference between describing a pod and reading a pod

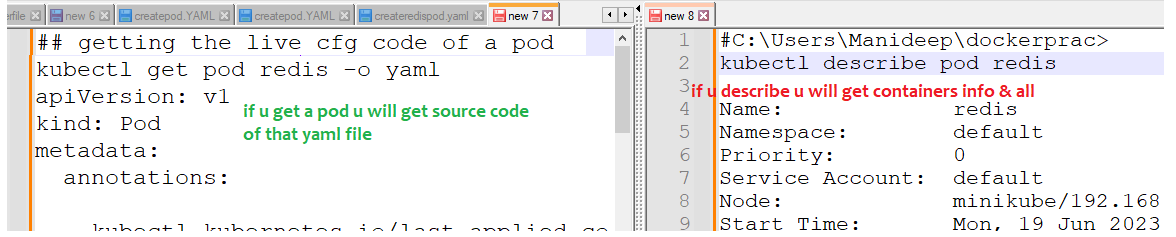
1. Read a pod-getting the live configuration of that pod

This will get the source code of the yaml file with which it was created,

To get more information of the pod in yaml file(output of the yaml file)

**kubectl get pod <pod name> -o yaml**

ex:- **kubectl get pod redis -o yaml**

****

1. EDIT a pod-once u edit and editor will be opened and to save changes just close that file

Now if u edit – u will get live configuration of that pod

**kubectl edit pod <pod-name>**

1. Deleting a pod

**kubectl delete pod <podname>**

kubectl delete pod myfirstpod

Deleting objects/pod using created file

kubectl delete –f ./pod.YAML

means with which file this pod is created , with that file we should use delete command

here all the kub objects (pod,namespace,replicaset) mentioned in this file will be deleted

Go inside a container

1. Go inside a container of a pod

Make sure here the pod must be running

kubectl exec –it <pod name> -c <container-name> bash

//u can use with or without – double hyphen

kubectl exec -it d1-depl-obj-name-7877f6d747-hzd5z sh

kubectl exec -it d1-depl-obj-name-7877f6d747-hzd5z – sh

kubectl exec -it d1-depl-obj-name-7877f6d747-hzd5z -- bash

//use either bash or sh, “sh” must be mandatorily available

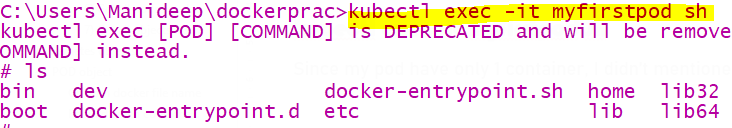
kubectl exec –it <pod name> -c <container-name> sh

//to exit from a container type “ctrl+PQ”

if there is only 1 container in ur pod no need to use –c flag

kubectl exec -it myfirstpod sh

Since my pod have only 1 container, I didn’t mentioned –c flag



1. Describe a pod –to see cont name

kubectl describe pod <pod-name>

ex:- kubectl describe pod myfirstpod

if u describe the pod u will get container name,and image with which it was created

also of all containers running inside that pod

u can describe any kubernetes object, u can describe a pod,namespace,replicaset any kubernetes object

1. Get all containers in a pod

Just describe the pod , u will get the list of containers present inside that pod

kubectl describe pod <pod-name>

1. CRUD nodes

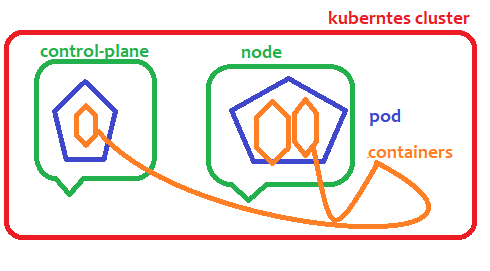
A separate linux machine is called a node, in AWS we will create a separate linux machine called as node

Every node and every pod will be having an different ipaddress

|  |  |
| --- | --- |
| Get all nodes | kubectl get nodes |
| Get all nodes with more information-means output in wider format | kubectl get nodes –o wide  //here u will get all nodes ip address |
| Describe a node to get info like- ipaddress,labels of that node | kubectl describe node <node-name>  kubectl describe node minikube |
| Labelling a node | Kubectl label node <node name> key=value |
| Show all the labels of a node | 1. Using describe command 2. Using kubectl |
| Deleting the label of a node |  |

Remember node is nothing but one virtual machine where docker is installed,

master node doesn’t carry load , so docker wont be installed in the control-plane node



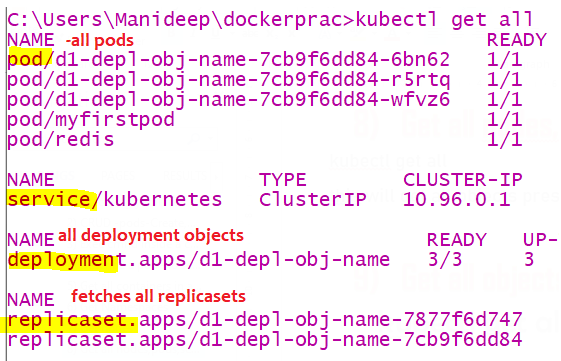
1. Get all nodes,pods,services everything

kubectl get all

this will get all objects present inside current namespace

kubectl get all -A

this will get all objects present inside all namespaces

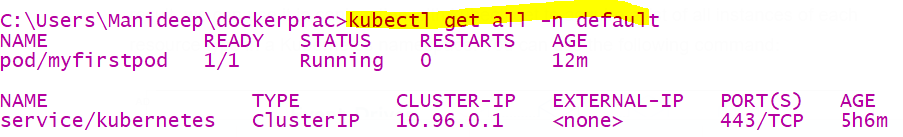


1. Get all objects inside that namespace

To fetch all kubernetes objects in particular namespace, when u are in namespace1, u can get all objects present in another namespace

Here u will get all obj like – pods, replicaset, deployment, service objects, config map, secret, ingress related objects

kubectl get all –n <namespace name>



1. Extract yaml file from running pod

This will not create any pod, it will just generate an yaml file with content, so that if u execute it pod/..will be created

kubectl run podhyd-redis –-image=redis –-dry-run=client –o yaml >pod.yaml

this will create a new pod with name “podhyd-redis”

kubectl run nginx --image=nginx --dry-run=client -o yaml > pod.yaml

*# Generate spec for running pod nginx and write it into a file called pod.yaml*  
ex:-

C:\Users\Manideep\dockerprac>**kubectl run redis --image=redis --dry-run=client -o yaml >createredispod.yaml**

1. Update image in running pod
2. If u have YAML source file and edit the image name and run “kubectl apply –f ./filename”, if this file is not

Available with u then follow below , all options will work

1. Just “**kubectl edit pod <pod-name >”** --then u will get editor ,update the image name,if u update like this

Only that particular pod will be recreated with new image and new container

1. If u are using replicaset edit as “**kubectl edit rs <replicaset name>** ” update the image name,
2. If u are using deployment object edit with “**kubectl edit deployment <deployment-name>** ” update the image name,- u have to edit the image name present In this hierarchy “spec-template—spec-containers-image”

Now if u update the image name in deployment object , all the nodes under that replicaset will be destroyed and recreated with latest image

1. kubectl image <type of obj>/obj-name <container-name>=<new image name>

//we can update image name in pod/deployment object ..anything

// 1pod can have 2-3 containers, so its asking container name also

Ex:- kubectl set image deployment/deplobj stephencontaineroo=stephengrider/multi-worker

Kubernetes objects

Example objects:- pod,development,replicaset,replicationcontroller,service,horizantalpod,autoscaler

Api versions- <https://blog.knoldus.com/what-is-apiversion-in-kubernetes-part-1/>

Every kubernetes object belong to some or another api group

Ex:- replicaset belongs to “v1” group deployment obj belongs to “apps” api group

If u want to see all in built objects and their api versions and their shortcut names,then below is command

kubectl api-resources

|  |  |  |
| --- | --- | --- |
| Kind | apiversion | apiVersion means like package, means that object is present in this package, all objects present only in some package like v1 or apps/v1 |
| Pod | v1 | Pod contains 1-n dependent containers |
| ReplicaSet | apps/v1 | This obj ensure always desired pods are present or not |
| Deployment | apps/v1 | This obj controls replicaset |
| Service | v1 | U talk to service object,it will talk to pods |
| HorizontalPodAutoscaler | autoscaling/v1 | For autoscale pods up & down |
| ComponentStatus | v1 |  |
| Namespace | v1 |  |
| Node | v1 |  |
| PersistentVolumeClaim | v1 |  |
|  |  |  |
| PodTemplate | v1 |  |
| ReplicationController | v1 |  |
|  |  |  |
| ResourceQuota | v1 |  |
| Secret | v1 |  |
| ServiceAccount | v1 |  |
|  |  |  |
| ControllerRevision | apps/v1 |  |
| StatefulSet | apps/v1 |  |
| CronJob | batch/v1beta1 |  |
| DaemonSet | apps/v1  extensions/v1beta1 |  |
|  |  |  |
| Role |  |  |
| RoleBinding |  |  |
| ServiceAccount |  |  |

Tips while writing objects

1. Always the below line must start with 2 spaces right of above line
2. In kubernetes one object is linked/can pick other objects only with the labels
3. Metadata is nothing but data about data

apiVersion: v1

kind: Pod

metadata:

#it says name of this object either pod /deployment..

name: charanpod

#namespace tells in which namespace this object is going to reside

namespace: charan

labels:

#label can be anything it is a selector ,other objects will pick /select this object based on the label

score: low

Image with kubectl installed

Image name where kubectl is already installed

docker pull bibinwilson/docker-kubectl:latest

Yaml file syntaxes

List always starts with “- “ dash space

Example:

**-** **tabitha:**

**#inner elements of a list starts with double space**

**name:** Tabitha Bitumen

**job:** Developer

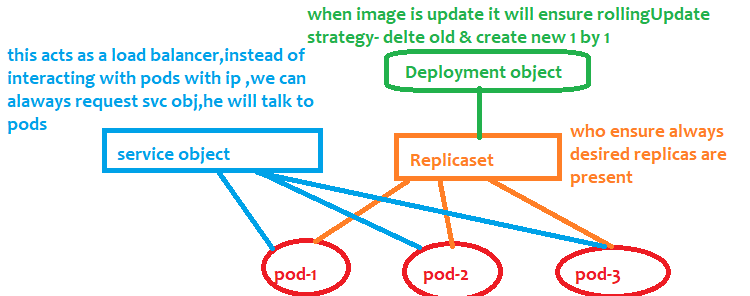
**skills:**

**-** lisp

**-** fortran

**-** erlang

Kub obj relations



Container

A container is nothing but some isolated space inside a hard disk where all the software’s can be installed to run a specific app (app like sdp/ coreiq/ any java app) software’s like java, .net,

## POD object

|  |  |
| --- | --- |
|  | 1 pod can have multiple docker containers  Each pod will have an ip address  So where is our application then??  As pod can have 1 or 2 containers – inside a pod inside a container (which is nothing but some isolated space in hard disk- inside that space java will be installed and our app will run there ) |

Pod is also a namespace scoped object, every pod contains a 1 or many dependent containers,

Our code runs in that pod only, we can’t deploy containers directly like in docker, pod is the smallest deployable unit,

Every pod will have separate ip address and a pod is ephermal means short lived & every pod should be attached with some label to identify

The problem with object pod is –u cant replicate it, we don’t have the option to replicate the pod object

Replicas is only with respect to pod, not with respect to containers ,

Example of pod creation script,to create a pod with mentioning limited resources

If the pod is taking more cpu than mentioned, then pod will get deleted and recreated



1 pod can have more than 1 containers, if ur apps are tightly dependent create all those apps in 1 pod

apiVersion: v1

kind: Pod

metadata:

name: myfirstpod

## this is the name of the pod and it will create the pod with some labels also

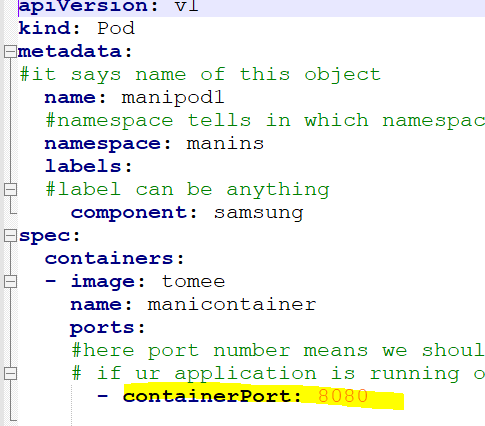
spec:

containers:

- name: nginc1

image: nginx:latest



Note if that app/image here (tomcat) runs on 8080, the same port number u should mention in “container port, “

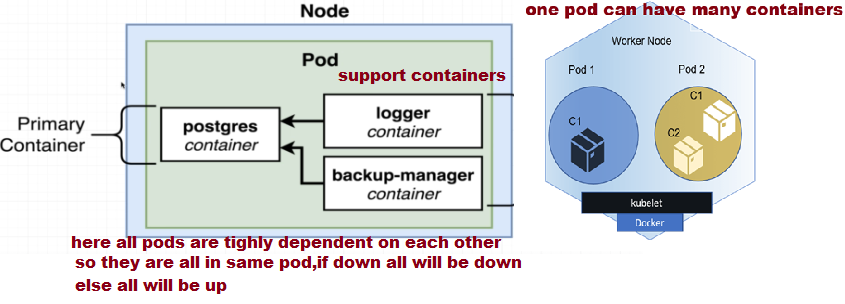
We can specify how much of memory,cpu,ram our pod can consume at max

2 containers in same pod

2 containers in same means 2 applications in same laptop/same os, those 2 apps can communicate using localhost.

If 2 containers cant live without one like child and mother then they will sit in same pod, those are tightly dependent containers,like ui and backend, if backend is waste without ui then those are tightly dependent on each other, only then put

Both containers in same pod, if they are not soo dependent then run as separate pods



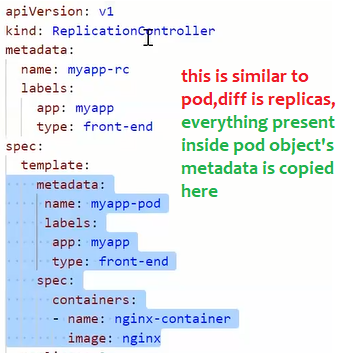
ReplicationController Object

ReplicationController Object ensures that always desired number of replicas/pods are present, even if u delete one replica among 3 desired replicas, immediately new one will be created

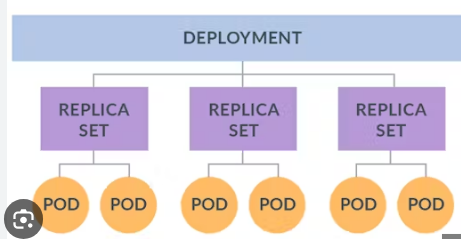
The only difference between pod and replication controller is ,

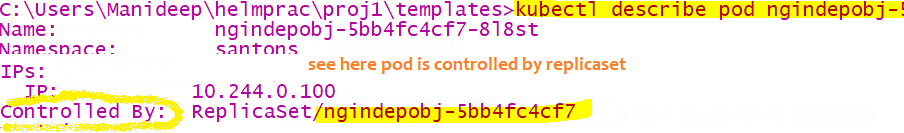
if u use pod object, u cant replicate, whereas when u replication controller we can create the replicas of a pod,

Remember pods are controlled by ReplicationController object



ReplicaSet object





This Replicaset is a wrapper around group of pods/replicas, this is also to create replicas of a pod **To ensure always desired number of pods are always there,**

Remember pods are controlled by ReplicationController object or Replicaset object, & Replicaset object is controlled by

Deployment object

This object is also same as object “ReplicationController” will also ensure desired number of pods are present

# replicaset obj always ensures 3 replicas of pod are there and it will pick the pods based on the label of the pod

# this is the real object who will control the pods directly, so to control the pods it should know which pod to pick right, this replicaset will pick based on the pod label ,here we should mention which pod to pick using pod selector

Labelling a pod

template:

metadata:

#here these are the labels which will be given to each pod

labels:

space: low

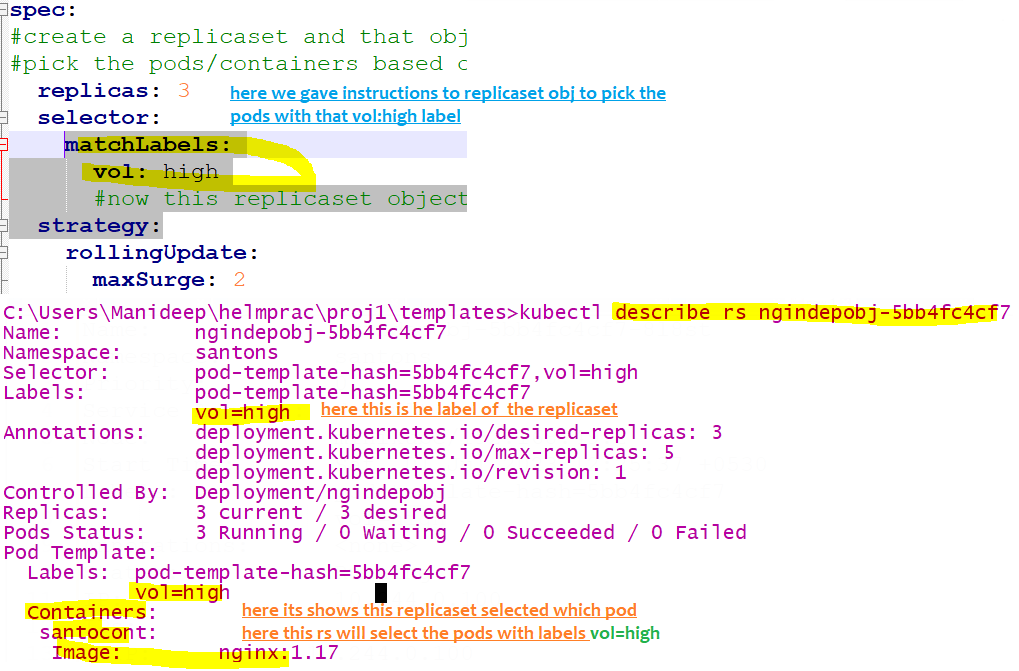
spec:

containers:

- image: nginx:1.17

name: bb

here for all the nginx contianers /pod the label space=low is applicable



|  |  |
| --- | --- |
|  |  |
| kubectl get rs  kubectl get replicasets | To get all replicaset objects type below |
| Kubectl describe rs <replicaset –name > | To see all label of that replicaset  And to check what is the selector ,I mean which labelled pods it will pick…u can see all this information 2) to see who is controlling this replicaset-generally replicasets will be controlled by Deployment objects |
| To delete a replica set | kubectl delete replicaset <replicaset-name> |
| Scale up or down a replicaset | kubectl scale rs <replicaset-name> --replicas=5 |
| If u want to see who is controlling the pod  “kubectl describe pod <pod-name> ”,  If u want to see who is controlling the replicaset ,  Generally replicaset will be controlled by deployment object | Kubectl describe replicaset <replicaset -name>  There is a field called “controlled-by” :” |

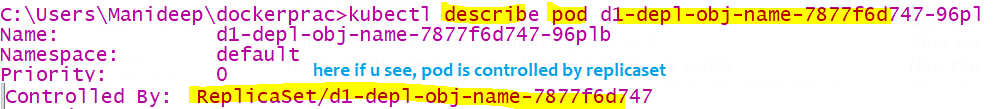
The difference between “ReplicationController” and “ReplicaSet” is the object ”ReplicaSet” object will have an additional

Selector to select the existing running pod

**Note:-as replicaset manages pods,if we mention replicaset as 3,if any pod got deleted, it will be created automatically**

**If u want to delete all the pods, 1st u should delete the parent-ReplicaSet/ReplicationController object who is controlling that**

**Here pods are controlled by parent ReplicaSet, so if u want to delete all pods then we should delete this replicaset first**

****

**If u want to delete all nodes under replicasset-u cant pod by pod because it will always ensure 3/5/desired number of replicas are present.**

Solution:-if u want to delete all nodes, better delete the replicaset first then only all pods will get deleted

|  |  |
| --- | --- |
| Pod name | replicas |
| Pen ( pod selector:oldStock) | Current replicas-3 |
| Pencil (pod selector:oldStock)  Whereas if we mention desired replicas as 5-  Diff pod with same selector (oldStock) already exists ,  hence eventhough u gave desired replicas as 5,it will create only 2 replicas,because 3 existing replicas as already present with same selector | Current replicas-0  After exec,  New replicas will be 2 because same selector 3 already exists ,so only 2 will be newly created |

if we mention replicas as 3 with pod selector (stock: fresh)if an another pod is already running with same name

It will consider that criteria, Now this pod will have only 2 replicas

Deployment object

Deployment is an object which is namespace scoped,means this object can present across all the nodes

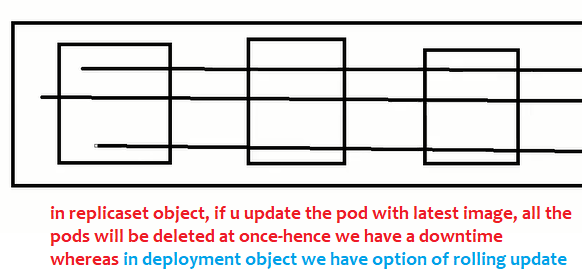
The main advantage of Deployment object is , when this object is created when that pod is running we can update the image dynamically, so slowly 1 pod will be deleted and new pod will be created, again 1 new pod will be created and 1 old pod will be deleted, so there is no downtime using the concept of rolling update,incase of other objects all old pods will be deleted first and new pods will be created with old objects we have downtime,in this Deployment object we don’t have that downtime

Deployment object is the parent object which controls the replicaset and replicaset object controls the pod object

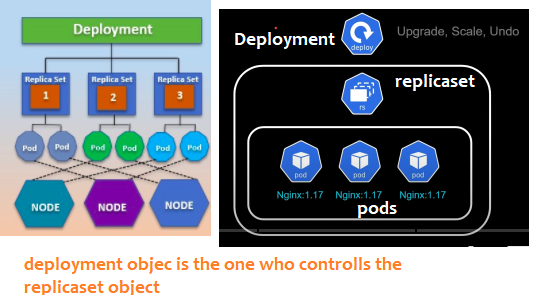
Real time use case

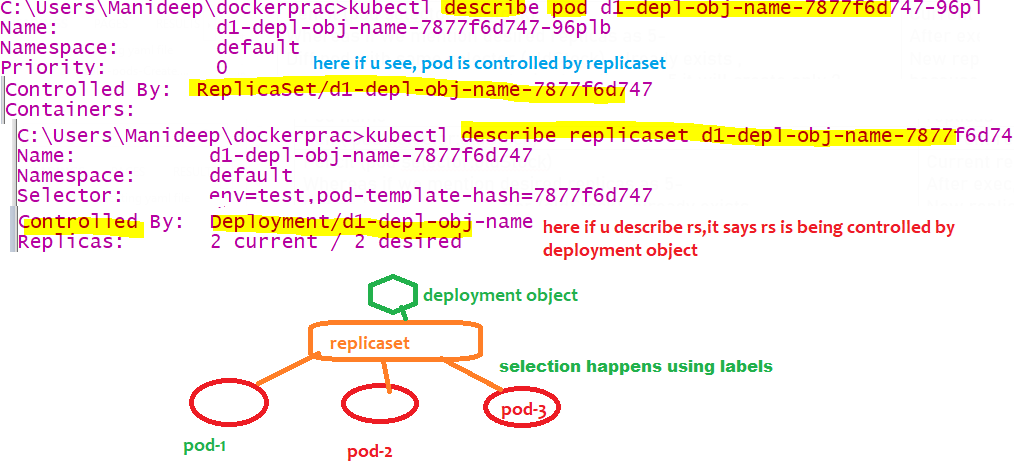
1. Flipkart during sale days, once sale is iver sharp 12 pm, all offers will be closed and they will show us the normal page,

How it is happening (without complete server down ) , they might be using Kubernetes deployment object, they would have entered and edited deployment object source code and change with the old image name, slowly pod by pod by it will delete old pod with old image, it will use deployment object Rolling strategy, 1 old pod with old image will be deleted and a new pod with new image will be created among 10-15 pods, slowly old and new will happen without donwitm



|  |  |
| --- | --- |
| To get deployment object names | kubectl get deployment |
| Most useful  If u want to update image name-2 options –edit deployment object or 2) edit pod  if u are trying to edit pod, u have to edit the image name in all replicas, means edit all replicas, 2) if u Edit the deployment object, u no need to edit each and every pod, because dep is the parent object , so all replicas (are childs )image name will be updated in all pods automatically and all pods will be recreated | kubectl edit deployment <deployment object name>  kubectl edit deployment d1-depl-obj-name  ## To update the image name/update replicas count -U will get the script/deployment object script/source code in new editor, So that u can edit image name |
| kubectl describe deployment <deployment-name>  kubectl describe deployment d1-depl-obj-name | When u describe u will see all the labels of deployment  How many replicas, what is the image and pod selectors..and deployment strategy=rolling update/ recreate everything and  And to check what is the selector |
| kubectl delete deployment <> |  |
| kubectl get deployments  To get all the deployment object names |  |
| Get current source code of deployment object in yaml –now we wil get source code or yaml file with which that deployment object is created  Or even if u edit the pod also, u will be able to see the source code use “kubectl edit pod podname” | kubectl get deployment <your deployment object name> -o yaml  kubectl get deployment d1-depl-obj-name -o yaml |
| Scale the replicas of a deployment-to scale up or scale down | kubectl scale deployment <deployment> --replicas =5  kubectl scale deployment d1-depl-obj-name --replicas=8 |
| Rollout the deployment to older image version | kubectl rollout undo deployment/<your deployment object name>  kubectl rollout undo deployment/d1-depl-obj-name |





How it is working internally

Here replicaset object links/selects pod based on the label of the pod only

similarly deployment object selects replicaset object based on the label of rs object.

Like a Pod wraps around container and containers,

Replicaset object wraps around group of pods-means replicaset will manage the pods

**Deployment object is a wrapper which wraps around a replicaset object,it manages the replicaset,**

**Replicaset which inturn manages the pods**

**Deplyment object restores replicaset (means if replicaset got delete,deployment obj will create it)**

Replicaset restores pod(means of any pod under that matched label replicaset got deleted,rs will create pod)

If Higher level deployment object is created then all lower level objects –replicasets,pods are created automatically

The main advantage is when the replicaset is running when u update the image version in YAML file,

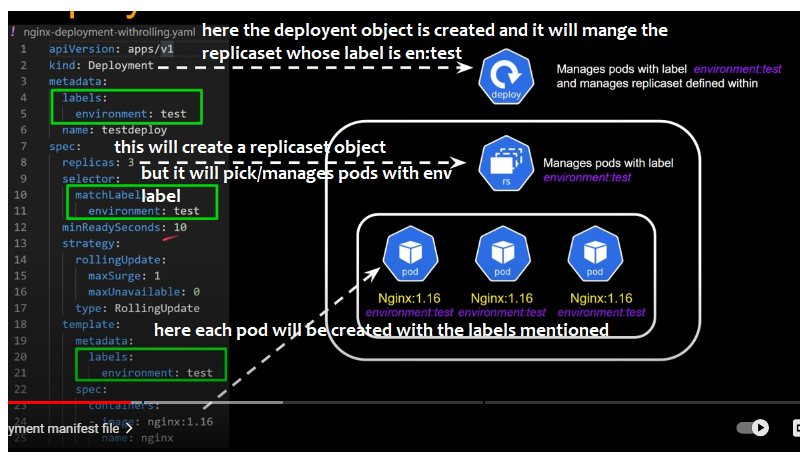
Deployment provides declarative updates for pods it will change the actual state to desired state in controlled rate

Note:-

Every object should have a label, else it will be difficult for wrapper /up level object to pick

eX:- here deployment object is a wrapper around replicaset object, deployment object pick/manage the object with the

specified selector



Note:label is very very mandatory to identify /manage the objects

Here parent object Deployment manages the child object replicaset only based on the label environment:test

If label is not there, deployment obje I think it cant select pods/vms try this. By updating a nginx version and check

Tips while writing deployment object

1) All the container creation logic (metadata,spec) is inside spec ,template in yaml file

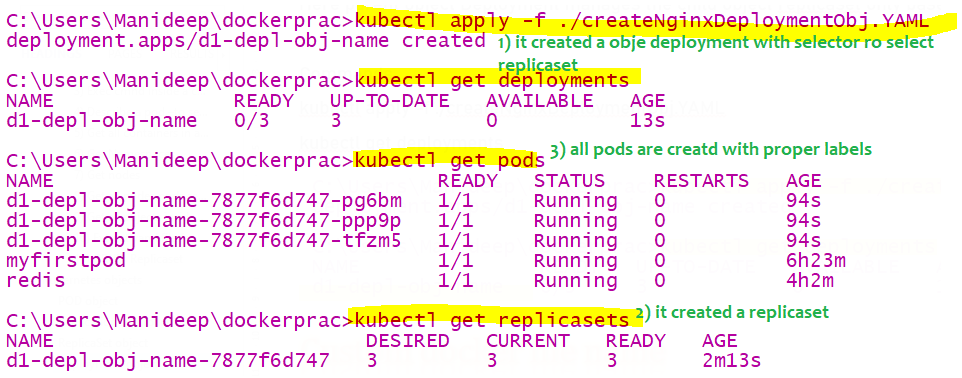
2) We will have only labels plural, not singular label

3) if u want to label that us should use keyword “label”, if u are selecting some label , then u should use “matchLabels”

Commands

kubectl apply -f ./createNginxDeploymentObj.YAML

kubectl get deployments



What is the use of deployment object:- when container is running we can update the image name

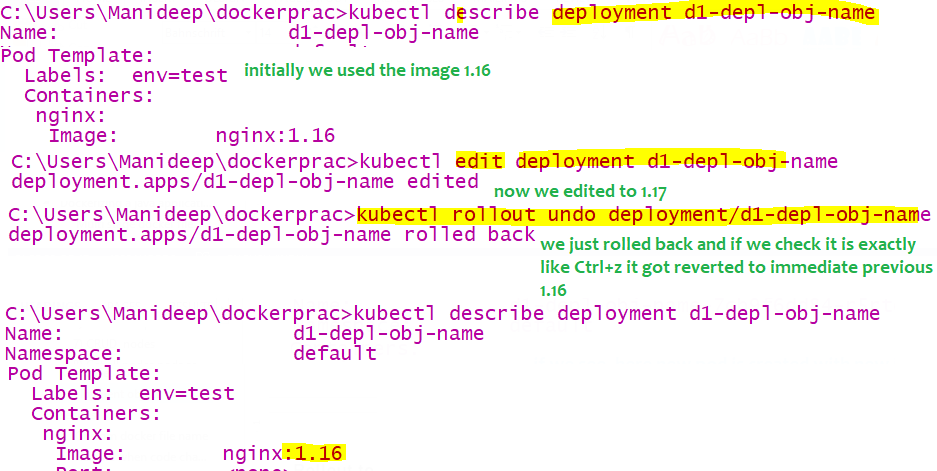
Earlier image name was1.16 now we update the image name to 1.17,pods Auto destroyed and newly created

The beauty with this is no downtime, all existing pods are not deleted at once, whereas in case of other objects

Like “ReplicaSet”,”ReplicationController” all pods will be deleted first and will be created everything newly with downtime



Rollout to older deployment



#### Example code:-

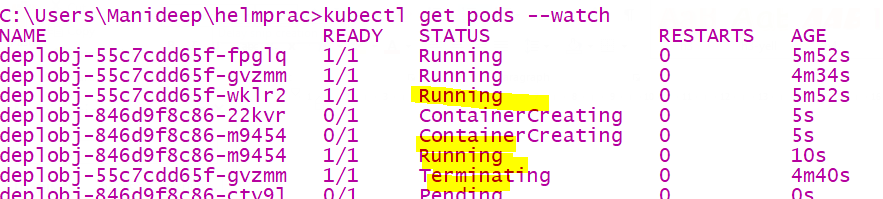
Refer attached file



observations

Observations notes

1. When deployment object is running, if u update the port, old pods will be deleted (using deployment strategy:rollingupdate..or any other)one by one and new pods will be created , this is possible only using deployment object even when pod is running u can change port number,imagename,container name because with deployment object there is no downtime



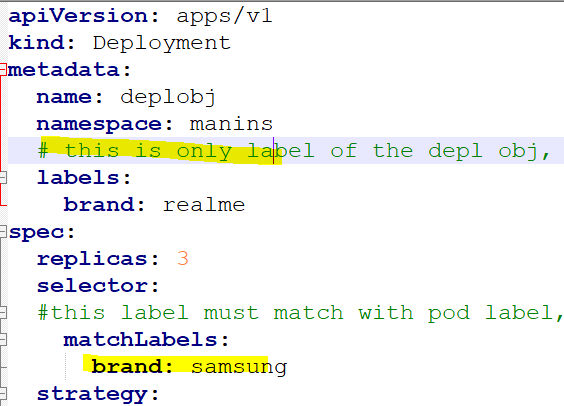
1. Deployment object label can be anything, whereas u should tell the specifically tell to

replicaset to pick pod the certain label

Ex: if replicaset match label is realme, rs will select only those pods with realme label

Refer the attached file for understanding





Daemon sets obj

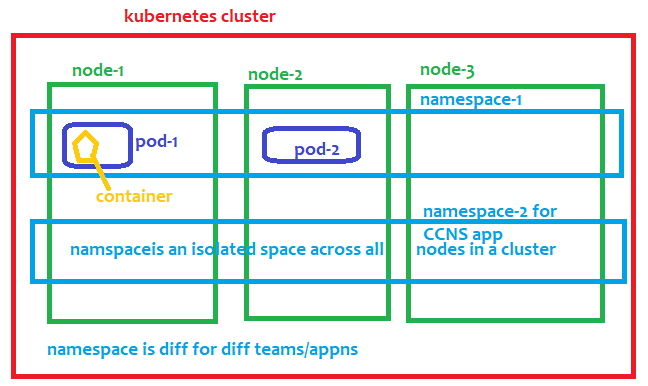
Initially we have 3 nodes in a cluster and we have 1 pod on each node, later if we add the nodes to the cluster if u want to

create the pod in each and every node, then we should use daemon set object

Namespace object

It is an isolated space which is available across all the nodes in entire cluster (or across the nodes)

Namespace will span across the nodes



Namespace provides a mechanism for isolating group of resources or objects (pod,deployment,replicasets,secrets)

in same cluster

If 2 different teams application if u want to isolate then we will create a namespace (1 for each team)

1. Name of the resources/objects-pod/Deployment should be unique within a namespace

Ex:- secrets will be stored in namespace level, if 2 apps are in same namespace, all will/can access those secrets

Some objects in the kubernetes are namespace scoped objects like POD,Deployment,

Whereas some objects will be deployed at cluster level

1. Deployment object is a namespace scoped object,

Within the same name space, the name of the deployment must be distinct, and in another namespace the earlier names can be present.

Namespace should be different for each and every team, generally access will be given to the entire namespace

If both team apps are in same namespace, I have the ability to delete their pods.. if 2 teams have same namespace

1 team app may consume more CPU, RAM resources another team may not. .so to avoid conflicts better to have different namespace for all application

Some objects will reside in namespace-also called as namespace scoped objects-POD,Deployment

These 2 objects will reside in namespace

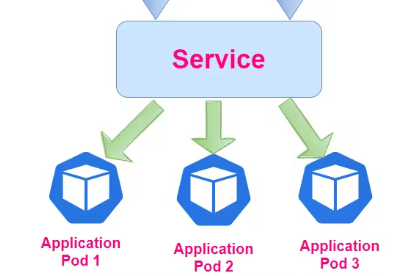
By default until u don’t set the namespace, u will be in default namespace



1. CRUD Namespaces

|  |  |
| --- | --- |
| To get all namespaces | kubectl get ns  kubectl get namespaces |
| Create a namespace | kubectl create namespace <name of the namespace>  kubectl create namespace maninamespace  kubectl create ns santons |
| To delete a namespace | kubectl delete ns <namespacename> |
| To get pods from diff namespace being in diff namespace | kubectl get pods -n <name of the namespace>  kubectl get pods -n kube-system |
| To get all objects from that namespace | kubectl get all -n <name of the namespace>  kubectl get all -n kube-system |
| To get pods from all namespces  This –A will work for all commands | kubectl get pods –A //means from all namespaces |
| In declarative approach if u want to create a object in particular ns, | Under metadata, add a field called namespace in YAML file, so that the file will be created in that namespace |
| To set a namespace to current context | kubectl config set-context –-current -–namespace=<ns name>  kubectl config set-context --current --namespace=maninamespace  kubectl config set-context --current --namespace=default |
| To know in which namespace u are in now | kubectl config get-contexts |

Services Object-ingress



Here service object is a layer4 load balancer it will just dispatch the request to all the pods of same app (not diff apps) equally in round robin /some other fashion

This service object is only visible inside the cluster, external users outside from cluster cant access service objects

Service is an object in kubernetes representing the logical group of pods that perform the same task

In docker and kubernetes containers and pods are ephemeral they are very short lived,

Every pod will have ip address,

when they die and came back again, ip address of the pod changes, hence we can’t direct communicate with pods

So best way is talk to the service object, service object will talk to all the pods bec it maintains the registry , when a pod born and when a pod dies service obj will keep the track of all pods and their ip addresses

* This service obj provides a stable network endpoint to talk with pods

using subtypes “NodePort” …which ever subtype is applicable

type “kubectl get pods –o wide” to get pods names with their ip addresses

We should attach all the pods or (the replicaset/deployment obj) to **service object** using labels

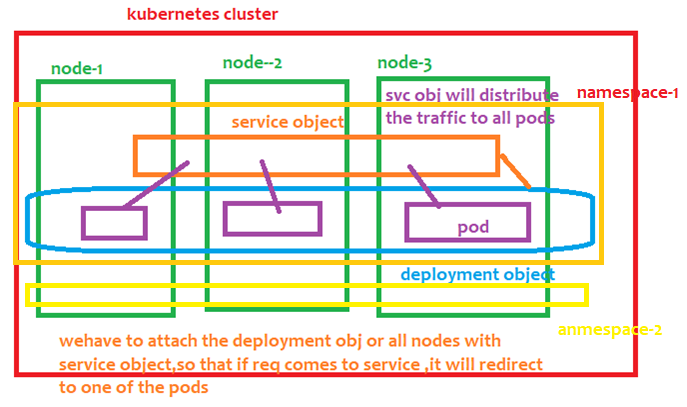
So that if request comes to service object it will redirect to one of the pods, when we use service object,

we should use only service object name to communicate b/n 2 pods

service object is used to distribute the traffic among those pods,it will select the pods based on pod selector,

for every pod we have a label, so using the label of a pod the service object will select the pod

Service object is also called as internal load balancer



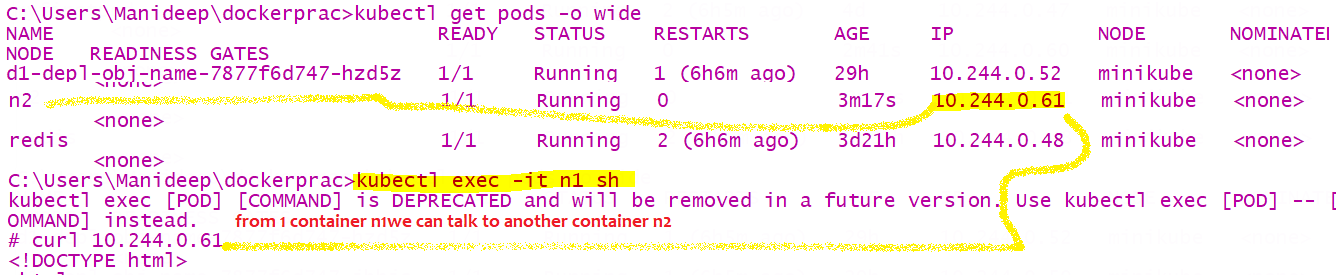
Service object is also a namespace scoped object,

After creating the service object ,if a pod joins with same label, then service object can distribute the

Traffic to new node also

1. Generally 2 containers can communicate with each other using ip addresses,but in realtime we should not

Use ipaddress because they are ephemeral-temporary



There are 3 types of service objects

1. ClusterIP(it will generate an private ip for service object which is accessible only inside cluster)
2. NodePort- not in production
3. LoadBalancer
4. Ingress

Source code for creating service object



|  |  |
| --- | --- |
| To get all service object names | kubectl get services |
| To describe the service object-  If u want to know which pods its picking using pod selector-like u can access all pods using svc name | kubectl describe service awscluster  kubectl describe service <service-object-name> |
| To delete a service object | kubectl delete service <service-object-name>  kubectl delete service awscluster |
|  |  |
|  |  |

1. Service obj type-ClusterIP

If the service object type is clusterIp, then to that service object local ip will be provided,

Means that ip address is private and accessible only inside a cluster

2 pods communicating with service name

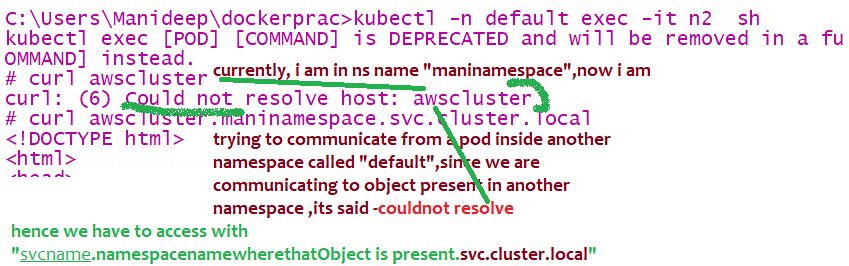


When ur are in namespace 1, and when u are trying to access service object present in different namespace

Then “curl 172.16.40.12” just ip address wont work, u have to access like

“servicename.namespace.basedomainName”

“awscluster.maninamespace.svc.cluster.local”



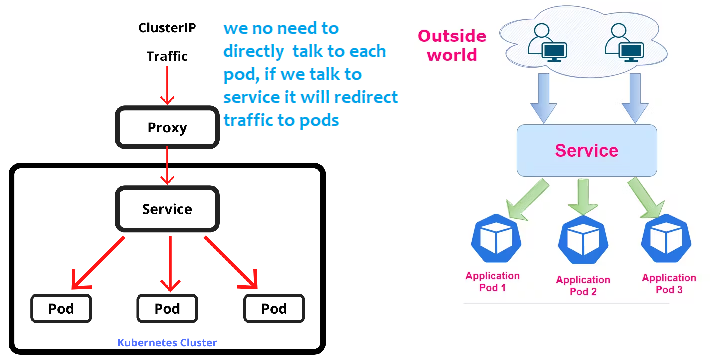
1. Service Obj type- Node port

In kubernetes cluster, we have many nodes, node is nothing but a separate machine/separate virtual machine,

It is also having lot of ports like our computer, where each s/w runs on different port, tomcat runs on 8080,jboss runs on 8081..

When we have configured a service object of type “NodePort”, then if we call service object it will call that ports

Nodeport means=call to IpAddress of node+ port where service object is installed in the node machine(bec it have lot of ports)



1. What is the advantage of NodePort Service object ?

We can access the pod using node machines ipaddress+port of the node machine where svc obj is installed, (1 node is a separate vm, its has man ports open in that 1 port),we have to tell in node machine on which port this service is running

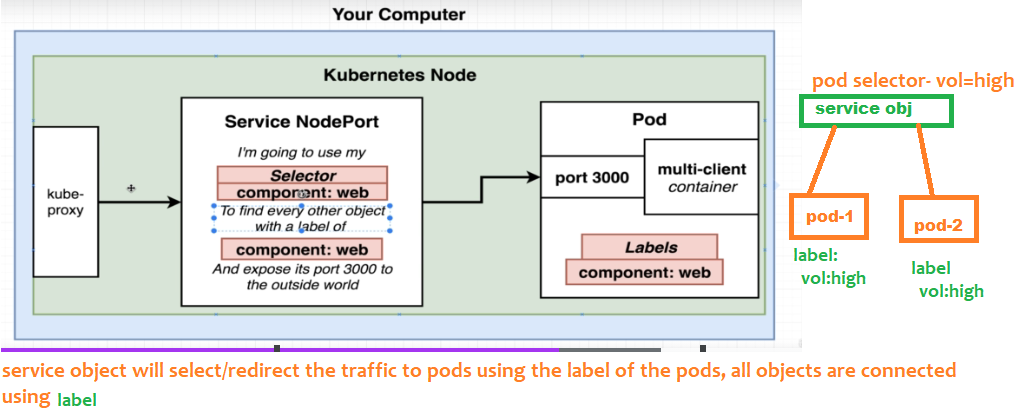
How to get nodes ip address

Kubectl get noes –o wide

With this we can access the pods using the ipaddress of the node and port of the service object



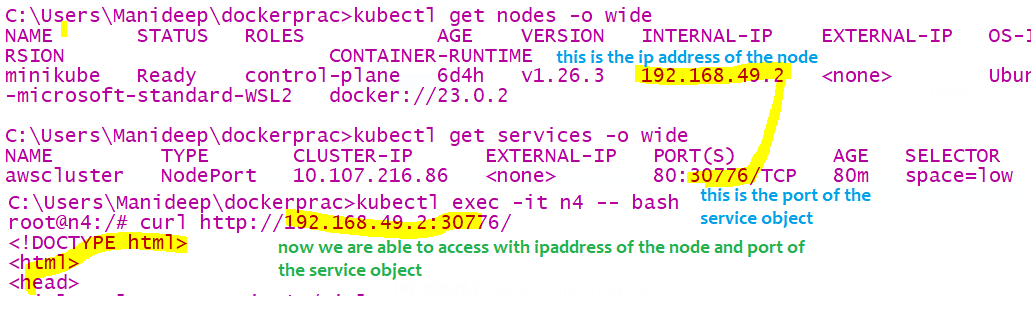
refer 6.serviceobject-Nodeport.yaml



selector:

#selector is the most important, this service object main purpose is to redirect the traffic to the pods, this service object will select and send the traffic to the pods with the below label

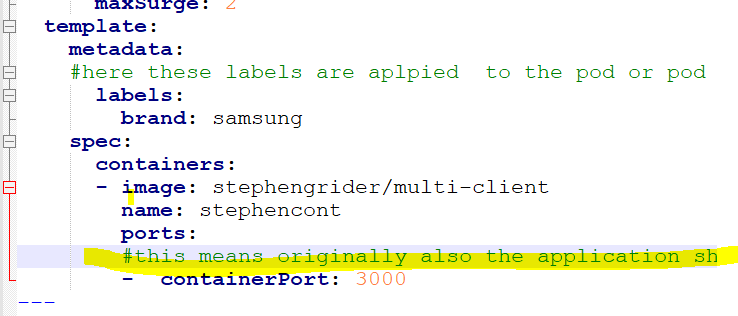
brand: realme



Here node port range will be from 30,000 to 32,767

Note:-

The container port matters a lot,container port means that is the port where the pod is already running



Example 2:-



Ingress Gateway

<https://www.youtube.com/watch?v=a4Kq6mX4Hr4>

Layer 4 load balancer means = This is the load balancer for one application

that load balancer will redirect the incoming request to virtual machines of same app Here service object is a layer4 load balancer

Ex:- google will run that app in 1000 of servers, whereas request will always first be handled by front load balancer it will dispatch to one of back vms

Ingress Is a Layer 7 load balancer which is bit more powerful than layer 4, here this app is the load balancer for multiple applications

in this backend multiple diff apps will be there like Facebook, twitter, YouTube, so when request comes this LB will identify the url and it will redirect that request only to those particular set of vm’s

This layer 7 load balancer will identify the incoming URL and it will dispatch those request to those particular vm’s or that particular service object

**Ingress is a layer7 load balancer & API level object that provides external access to service objects,** kubernetes service object is a layer 4 load balancer, only this layer 7 LB can accept the outside traffic,

This will route external traffic to specific service objects which internally will call the pods (this ingress allows us to expose multiple services to the internet)

In ingress it has 2 components /objects – Ingress controller, ingress resource

Adv of this load balancer is cost- if u use the same load balancer for all app then cost will be decreased

For every cloud they have one specific ingress controller ex:- nginx people developed one own Ingress controller, aws have their own IC,

Kubernetes clusters will get external access to service objects by using ingress controllers and ingress resource

|  |  |
| --- | --- |
|  | Ingress is a namespace scoped object which enables external access to service object because of this these service object will be exposed to outside http  Ingress may provide load balancing (service obj is already load balancing and it will redirect the traffic to all pods )  **apiVersion**: networking.k8s.io/v1  **kind**: Ingress  **metadata**:  **name**: minimal-ingress  **annotations**:  **nginx.ingress.kubernetes.io/rewrite-target**: /  **spec**:  **ingressClassName**: nginx-example  **rules**:  - **http**:  **paths**:  - **path**: /testpath  **pathType**: Prefix  **backend**:  **service**:  **name**: test  **port**:  **number**: 80  This ingress obj will be created by TKGI team developer will not configure this for each and every project  Means externally from chrome we can access this ingress url |

In kubernetes an Ingress is an API level object that define rules for routing external HTTP and HTTPs traffic to service objects within the cluster

Why ingress? If multiple apps are running in same cluster, taking one lb for each appln will be a costly, so here ingress will be the single load balancer for all apps

Deploy all ingress controller objects using the following command

//if u run this all pods will be created and objects like –pod,Deployment,job,replicasets,Ingress

most of the objects Will be created automatically in the ingress namespace

The main aim of ingress is to avoid creating multiple loadbalancers,only single load balancer,

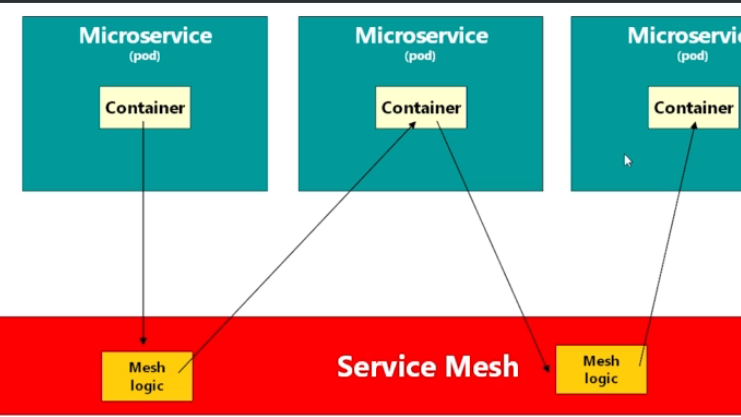
|  |  |
| --- | --- |
|  | Here we have 2 service objects both are of clusterIP typed objects (all service doesn’t have any kind of external ip  Means these service objects are not visible outside)  Based on incoming request path, ingress will redirect the requests to respective service objects   1. In above YAML file , a Service object of type load balancer will be created, when that object is   Created u will get the external url of load balancer   1. Here if path is having nginx ingress is redirecting that request to nginx service object 2. See nginx, httpd are 2 diff application here ingress is acting as a common load balancer for 2 applications   Based on request path, request will be redirect to any one of the service object    This is the url of ingress load balancer      See here we are using same load balancer url to access both application, only based on path it is identifying and redirecting to specific service object |
| Get an ingress | kubectl get ingress |
| Describe ingress | kubectl describe ingress <ingress-name> |
|  |  |

Service Mesh

Ex:- Istio (Istio is one example of service mesh there are many other service meshes available)

This is an extra layer of software you deploy along with your cluster which is used for gathering metrics and to monitor the traffic between those 2 pods

|  |  |
| --- | --- |
|  |  |



This service mesh is used for Telemetry (gathering metrics) this is something like filter which will be executed before and after calling target microservices

2 pods can communicate each other using service mesh advantage is service mesh will record time taken for inter communication between 2 microservices

Istio

|  |  |
| --- | --- |
| These proxies are collectively called “Data plane” in istio  In reality the proxy is nothing but a container (as 1 pod can have happily many containers) | In reality Istio install a proxy inside every pod so  when a container wants to talk to another container all calls will happen via proxy  now the proxy will calculate the time taken for interservice communication    Here istiod Is a new separate pod which will gather all the metrics from the proxies  In the istio control plane there are totally 6 pods along with above 2 pods  All these pods will be separate namespace called “istio-system” |

For istio, to gather the metrics, we should apply label to that namespace

Ex:- kubectl label namespace default Istio-injection=enabled

After labelling that namespace, istio will monitor the traffic between 2 pods

Side car container

|  |  |
| --- | --- |
|  | Side car container is nothing but a helper container to main pod  Ex:- Like wife and husband, like life boats in a big ship  This sidecar container will share some of the works of main container  Note:- both containers will be in same pod  Can communicate via localhost |

Passing env variables

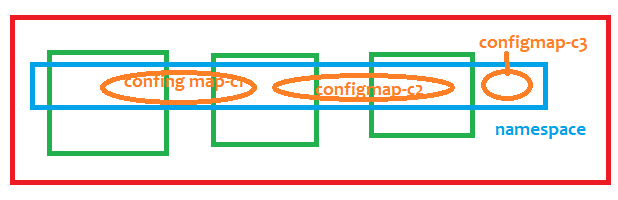
1)using env element in containers



U can see all those env variables u can set with “kubectl describe pod <podname>”

configMap object

this object will reside in namespace(ns is nothing but isolated space present inside a cluster)



We can have multiple config-map objects present inside the namespace

If u run the config map using “kubectl apply -f ./9.createConfigMap.YAML” all the entries will set to config map object present which will be present in the namespace

|  |  |
| --- | --- |
| To create a config map | kubectl apply -f ./9.createConfigMap.YAML |
| To see list of config maps | kubectl get configmap //from current namespace  kubectl get cm //from current namespace  kubectl get configmap –A //from all namespaces |
| Describe a config map | kubectl describe cm coredns -n kube-system  kubectl describe configmap <your config map object name>  kubectl describe configmap dbconfigmap  kubectl describe cm dbconfigmap |
|  |  |
|  |  |

Create a configmap

If u create and execute that file, entire config map object will be stored in a namespace



### Accessing a single value from that config map object present inside that namespace



spec:

containers:

- name: simple-web-app1

image: nginx:latest

ports:

- containerPort: 8080

env:

- name: anykey

valueFrom:

configMapKeyRef:

name: dbconfigmap

#this is the name of configmap object

key: dbpassword

### Accessing all values from that config map object



1. **Accessing single and all values from configmap**



env:

- name: databaseusername

valueFrom:

configMapKeyRef:

name: dbconfigmap

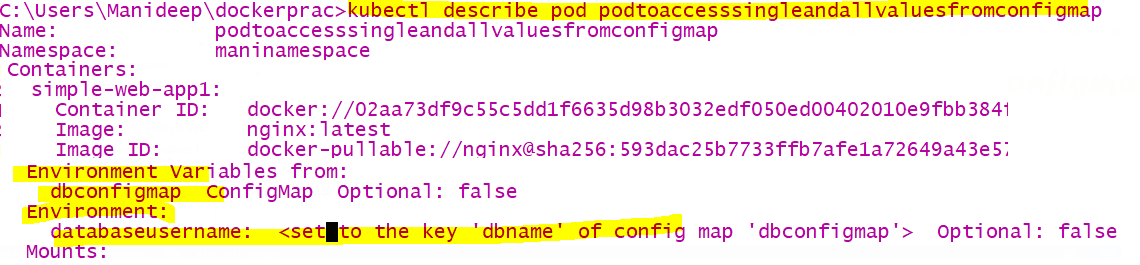
key: dbname

envFrom:

#we can get values from multiple maps,hence it is a list type,hence we used - symbol

- configMapRef:

name: dbconfigmap



Secrets



Secrets is a namespace scoped object, Once u created secrets object, all those secrets will be stored at namespace

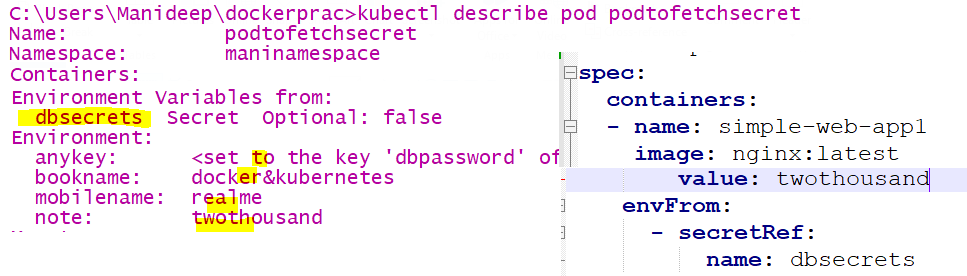
Only encrypted passwords will be allowed to place in the file

|  |  |
| --- | --- |
| Get all secrets | kubectl get all secrets |
| Describe a secret  //even though u describe the secret u cannot see the password | kubectl describe secret dbsecrets  kubectl describe secret <secret name> |
|  |  |
|  |  |

### Fetching all values from a secret

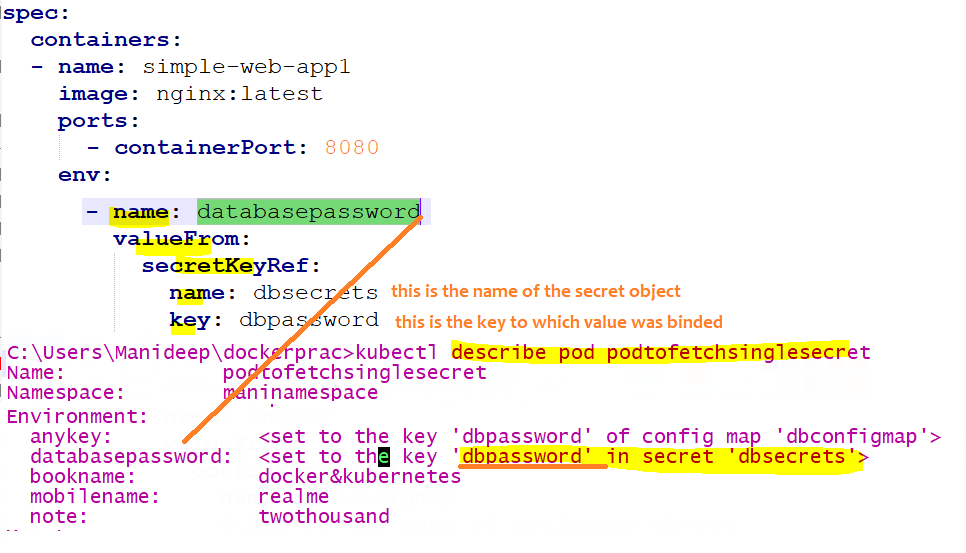


Here we mentioned as secretREf-because we are referring to all key-value pairs



### Fetching single value from a secret





Here we mentioned as secretKeyRef -because we are referring to single key-value pair

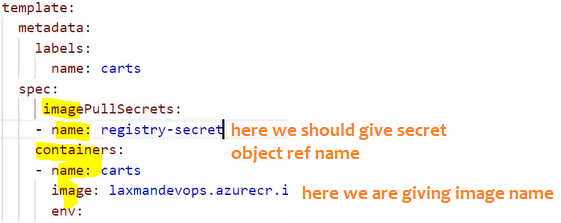
### How to pull image from private registry like jfrog/azure box

U have to create a secret Object with all those credentials of ur container registry like azure registry or jfrog

And u have to provide that secret object name in deployment.YAML file, while pulling the image we will mention the

Image name ,similarly we should mention secret object name, so that it will use that secret object credentials and pull that

Image from that private container registry



Role

Role is also a kubernetes namespace scoped object,



Example roles are

verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

we can map this role to any user or service account

to attach a role to any user, we cant simply give that role name to that user/service account, we should use

“RoleBinder” object

|  |  |
| --- | --- |
| To get role | kubectl get roles |
| To delete a role | kubectl delete role <>roleobj-Name>  kubectl delete role roleobj |
|  |  |
|  |  |

Always first 1) create a role 2) then create a service account

3) at last create a role binding object to bind/map role to that namespace

ServiceAccount

U should attach a role to a service account, instead of user credentials we will create a credentials which is going

To be used by application, ex:- In our code we wont keep our database username & password right, for application

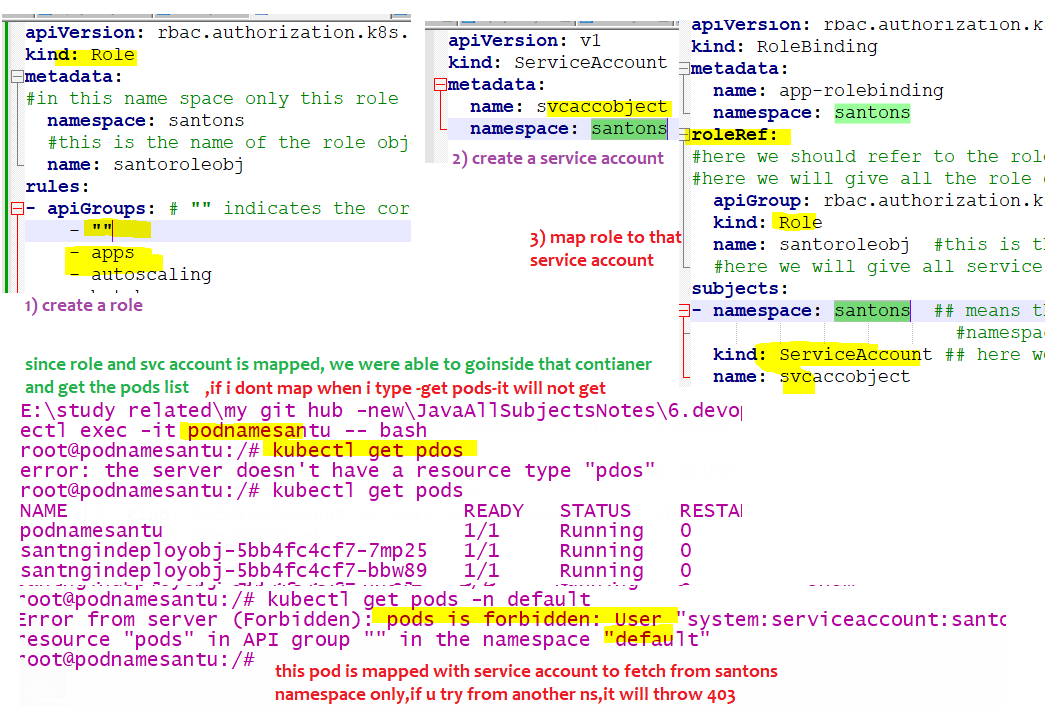
We will create some credentials, those credentials will have some roles , like how we have some roles

For every service account, some roles will be mapped , we can use those service account credentials



|  |  |
| --- | --- |
| To get all service account names | kubectl get serviceaccount |
| To delete particular service account name | kubectl delete serviceaccount <service-account-name>  kubectl delete serviceaccount svcaccobject |
| To get service accounts from all namespaces | kubectl get serviceaccount -A |

Once a role is already mapped to a service account, then only that can do those role operations



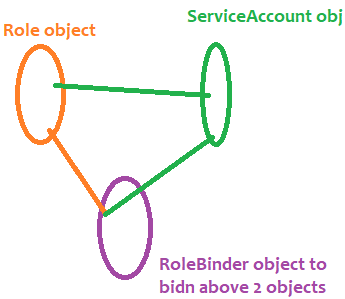
RoleBinding

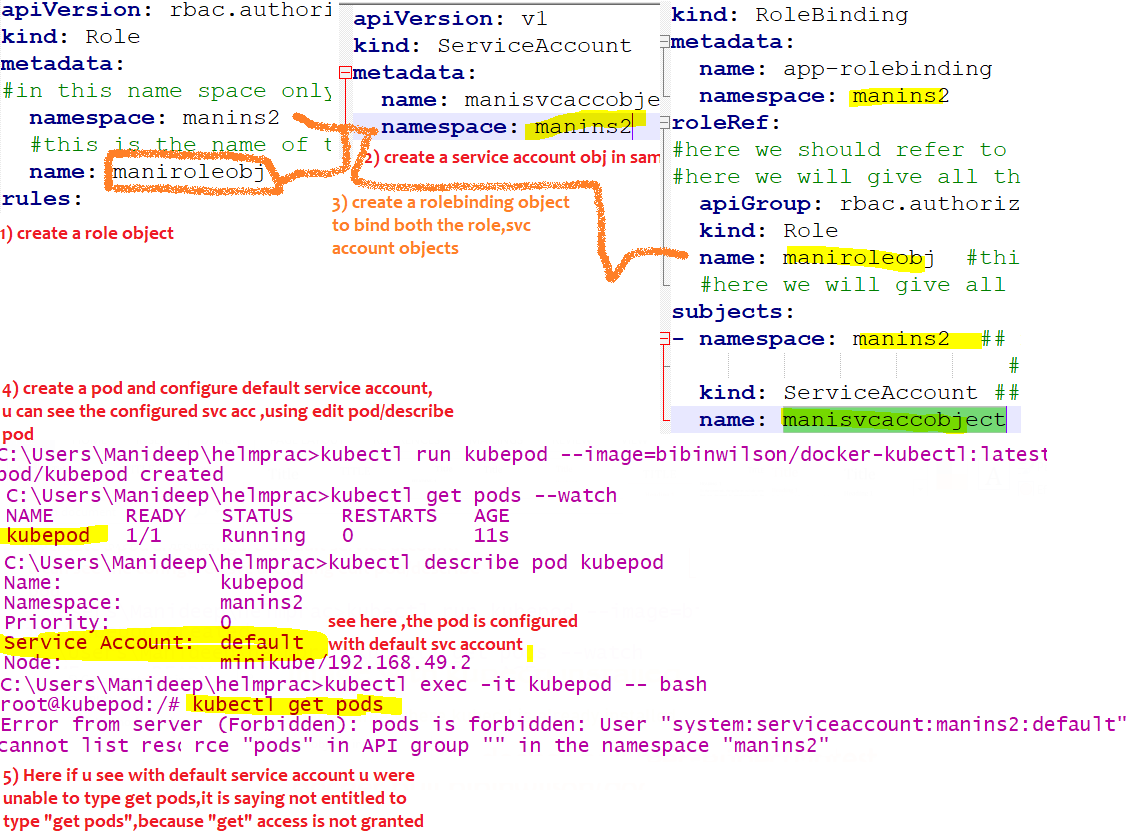
This is also a kubernetes namespace scoped object



|  |  |
| --- | --- |
| To get all rolebindings | kubectl get rolebindings |
| To delete a role binding | kubectl delete rolebinding <role binding object name>  kubectl delete rolebinding app-rolebinding |
|  |  |

Example of role,RoleBinding,svcaccount

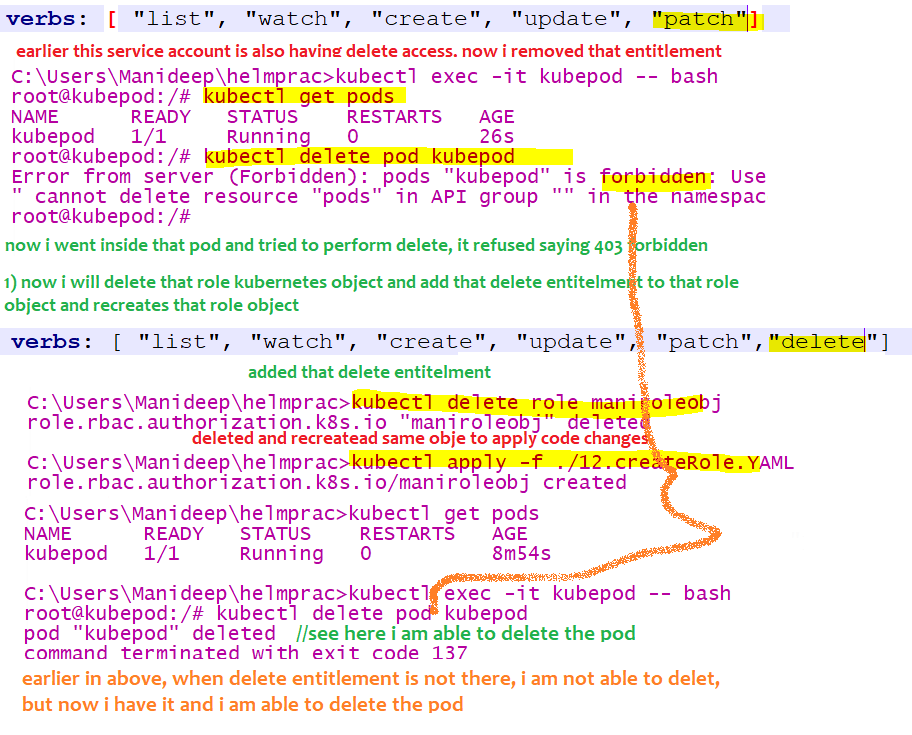
1. Create a role object in ur namespace say “manins2” (here namespace name is manins2)
2. Create a “ServiceAccount” object in your namespace
3. Create a “RoleBinding” object used to bind above 2 objects
4. 



Now I edited the pod and changed to the serviceaccount which have all the access



Lets re add the “delete entitlement” and see the status



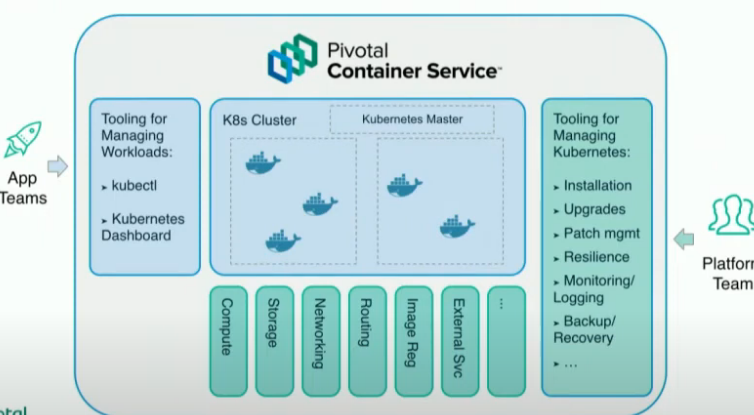
PKS

Pivotal kubernetes service- this means this is not just another kubernetes platform

This PKS is built on top of kubernetes with some additional features like roles, this PKS is built by keeping diff personas/roles/persons in mind

Ex:- “kubectl delete node <node-name>” in plain kubernetes any one can run this command, where as In PKS they have added roles like only platform engineers alone should run this command , with help of PKS this is possible like only platform eng can run this command

* PKS gives enterprise grade security



Each implementation of cluster api is specific to infrastructure

Cap A= cluster api for amazon

Cluster api == bosh infrastructure

## Horizantalpodautoscaler

This is also a kubernetes object which is used to auto scale up/down the number of pods ,i.e., when load increases it will create new pods, when load decreases it will delete the existing pods, we can set minimum and maximum number, so based on that load , pods will be deleted and created automatically

Ex:- like in irctc 11:00 AM tatkal timings if load increases pods will be created automatically at that time and

Pods can be deleted automatically after load slowly decreases

How it works internally??

1. First we have to install metric server, and we have to set the limits to our container or pod,

Check with this command “kubectl top nodes” or “kubectl top pods”

this will tell u whether metrics server is installed or not

1. Then metric server will keep on sending the metrics to HorizontalPodAutoscaler object, then if existing mentioned

Cpu limits are getting crossed means, automatically additional pods will be created and deleted dynamically

1. And we will link horizontal pod autoscaler with deployment object, else how does hpa know which obj to scale

In hpa.yaml file below, we linked hpa object with deployment object.

Steps to download the metrics server

1. Git clone <https://github.com/manideep-vv/Forked-k8s_metrics_server>
2. Go to deploy/1.8+ folder and execute all the yaml files using command “kubectl apply -f .”
3. U can verify installation of metrics server using “kubectl top nodes”
4. Now create a normal application with deployment object, all pods and deployment object should have a labels, then only HPA can select the deployment object and rescale it to have more replicas based on high load

|  |  |
| --- | --- |
| To see list of available hpa | kubectl get hpa |
|  |  |
|  |  |

Notes:

#namespace contains lot of deployment objects among them which deployment obj it should pick ,we should link hpa with Deployment object or some pod object, if u dont link how does the hpa object knows which Deploymentobj it should pick and scale,so remember to keep a selector so that hpa can pick that deployment object

Command to increase the load

kubectl run -i --tty load-generator --rm --image=busybox --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -o- http://hpa-deployment-obj; done"

here “hpa-deployment-obj” is the name of the deployment object

