**0.General Commands**

* kubectl get all —all-namespaces = kubectl get all -A
* kubectl create namespace test -o yaml/json : get json or yaml format file
* kubectl replace —force -f {file name}
* alias k=kubectl

**1.Pods**

* kubectl run nginx —image=nginx
* kubectl run redis —image=redis —port=8080
* kubectl run redis —image=redis —port=80 —expose=true : expose auto creates a new service with target port which we gave
* kubectl get pod
* kubectl describe pod {}
* kubectl delete pod {}
* kubectl get pod {} -o yaml > pod-definition.yaml: extract definition to a file
* kubectl edit pod {}

**2.Replica Set**

* In the ReplicaSet, .spec.template.metadata.labels must match spec.selector, or it will be rejected by the API.
* kubectl create -f rs.yaml
* kubectl **delete** replicaset {}
* kubectl **get** replicaset/rs
* kubectl describe rs {}
* kubectl **create** rs my-dep —image=nginx —replicas=3
* kubectl **replace** -f rs-definition.yaml : after making change in yaml file
* kubectl **scale** —replicas=6 -f rs.yaml
* kubectl scale —replicas=6 replicaset my-replicaset
* kubectl **edit** rs {} : editor will open and do changes

**3.Deployment**

* kubectl create -f deployment.yaml
* kubectl **delete** deployment {}
* kubectl **get** deployment/deploy
* kubectl **describe** deployment {}
* kubectl **create** deployment my-dep —image=nginx —replicas=3

**4.Namespace**

* kube-system, default, kube-public: auto created namespaces
* kubectl get namespaces/ns
* kubectl get pods —namespace=kube-system
* kubectl create -f pod.yaml —namespace=dev
* (or we can add namespace in metadata of object yaml file)
* kubectl create namespace dev
* kubectl config set-context $(kubectl config current-context) —namespace=dev
* kubectl get pod —all-namespaces / kubectl get pod -A

Resource Quota

* used to create quota for namespaces

**5.Imperative Commands**

* Declarative use definition files
* Imperative commands help gettingone time tasks done quickly

If you simply want to test your command , use the --dry-run=client option. This will not create the resource, instead, tell you whether the resource can be created and if your command is right.

-o yaml: This will output the resource definition in YAML format on screen.

kubectl run nginx --image=nginx --dry-run=client -o yaml

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

**Create a Service named redis-service of type ClusterIP to expose pod redis on port 6379**

kubectl expose pod redis --port=6379 --name redis-service --dry-run=client -o yaml

(This will automatically use the pod's labels as selectors)

Or

kubectl create service clusterip redis --tcp=6379:6379 --dry-run=client -o yaml (This will not use the pods labels as selectors, instead it will assume selectors as **app=redis.**[You cannot pass in selectors as an option.](https://github.com/kubernetes/kubernetes/issues/46191)

**Create a Service named nginx of type NodePort to expose pod nginx's port 80 on port 30080 on the nodes:**

kubectl expose pod nginx --port=80 --name nginx-service --type=NodePort --dry-run=client -o yaml

(This will automatically use the pod's labels as selectors, [but you cannot specify the node port](https://github.com/kubernetes/kubernetes/issues/25478). You have to generate a definition file and then add the node port in manually before creating the service with the pod.)

Or

kubectl create service nodeport nginx --tcp=80:80 --node-port=30080 --dry-run=client -o yaml

(This will not use the pods labels as selectors)

**6.Commands and Arguments**

* Docker

FROM Ubuntu

ENTRYPOINT [“sleep”]

CMD [“5”]

* entrypoint has the command that will run
* cmd has the default args else we can provide too
* docker run ubuntu-sleeper
* docker run ubuntu-sleeper 10
* docker run ubuntu-sleeper sleep 10 : (if no entrypoint is there)
* Kubernetes

in pod.yaml

containers:

name: redis

image: redis

commands: [“sleep”, “50”]: {[“cmd”, “param”]}

**or**

commands:

-“sleep”

-“50"

args: [“—color”, “green”]

**Also this command and args cannot be edited so create new file/pod to change**

* kubectl run redis —image= redis — <args1> <args2> <args3>
* kubectl run redis —image= redis —command — <cmd> <args1> <args2> <args3>

**7.Config Maps**

* kubectl get configmaps
* kubectl describe configmaps
* kubectl create configmaps \

<config-name> —from-literal=<key>=<value> \

—from-literal=<key>=<value>

* kubectl create configmaps \

<config-name> —from-file=app\_config.properties

* *apiVersion: v1*
* *kind: ConfigMap*
* *Metadata:*
* *data:*
* *key: value*

**8.Secrets**

* kubectl get secret
* kubectl describe secret
* kubectl create secret **generic**{it’s the type} \

<secret-name> —from-literal=<key>=<value> \

—from-literal=<key>=<value>

* kubectl create secret **generic** \

<secret-name> —from-file=app\_config.properties

* *apiVersion: v1*
* *kind: Secret*
* *Metadata:*
* *data:*
* *key: value : {*values must be store in base64 format*}*
* echo -n ‘password’ | base64
* echo -n ‘cGFzd3Jk’ | base64 —decode

**9.Environment Variables**

* how to provide env variables to a pod
* env variables to be added inside a POD yaml file
* envFrom:
* -configMapRef:
* name: app-config
* envFrom:
* -secretRef:
* name: app-config
* env:
* -name: APP\_COLOR
* valueFrom:
* configMapKeyRef:
* name: app-config
* Key: APP\_COLOR
* env:
* -name: APP\_COLOR
* valueFrom:
* secretKeyRef:
* name: app-config
* Key: APP\_COLOR
* Volumes:
* -name: app-config-volume
* configMap:
* name: app-config
* Volumes:
* -name: app-config-volume
* secret:
* secretName: app-config

**10.Security Context**

* security contexts are mainly used to set what user to run as or as to provide what capabilities that user can have
* we can implement this under **spec** or under **container** prop
* security context under **container** overrides the one under **spec**

*securityContext:*

*runAsUser: 1000* *{must be an integer value}*

*capabilities:*

*add: [“MAC\_ADMIN”, “SYS\_TIME"]*

**11.Service Accounts**

* kubectl create serviceaccount dashboard
* kubectl get serviceaccount / sa
* kubectl describe serviceaccount dashboard
* after describing the service account, there is token property in it
* kubectl describe secret {token-name}
* it will show the token that is used by application to access the service account
* secret location in pod: /var/run/secrets/
* there is a default service account that is auto mounted to all pod created

*POD.yaml*

*Containers: ...*

*serviceAccountName: {name}*

*automountServiceAccountToken: false/true*

**12.Resource Requirement**

* scheduler is responsible for scheduling pods to nodes
* if pod tries to use more CPU than the limit, then kubernetes throttles the cpu
* if pod tries to use more memory than limit consistently, then the pod will be terminated

Container:

  name: redis

  image: redis

  resources:

    requests:

      memory: “1Gi"

      Cpu: 1

    limits:

      memory: “2Gi"

      Cpu: 2

**13.Taints and Tolerations**

* taints are applied to the node so pods cannot be allocated to them
* tolerations are applied to pods so they can allocate to nodes with taints
* *they are mainly used to prevent some pods to allocate on some nodes and not to assign pods to specific nodes*
* master node has taint on it by default
* kubectl taint nodes node-name key=value:taint-effect{NoSchedule | PreferNoSchedule | NoExecute}
* kubectl taint nodes node-name app=blue:NoSchedule : taint the node
* kubectl taint nodes node-name app=blue:NoSchedule- : untaint the node

spec:

  containers:

    name:

  tolerations:

  -.key: “app”

    operator: “Equal”

    value: “blue”

    effect: “NoSchedule"

**14.Node Selector and Node Affinity**

* node selector is used to directly assign pod to specific nodes based on labels
* kubectl label nodes <node-name> <label-key>=<label-value>
* kubectl label nodes node-1 size=Large

spec:

  containers:

    name:

  nodeSelector:

    Size: Large

**Node Affinity:**

* requiredDuringSchedulingIgnoredDuringExecution
* preferredDuringSchedulingIgnoredDuringExecution
* requiredDuringSchedulingRequiredDuringExecution

**spec**:

**affinity**:

**nodeAffinity**:

**requiredDuringSchedulingIgnoredDuringExecution**:

**nodeSelectorTerms**:

- **matchExpressions**:

- **key**: size

**operator**: In / NotIn / Exists(no need of value)

**values**:

- Large    
   
  
  
 **containers**:

- **name**: nginx

**image**: nginx