To get nodes: kubectl get nodes

To run image:-kubectl run ngnix --image=ngnix-------------------it will create pod also

To get pods : kubectl get pods

To get more info abt pods:- kubectl describe pods

To get node info also:- kubectl get pods -o wide

Kubectl create -f deployment.yaml

Kubectl get replicaset

Kubectl describe replicaset

Kubectl get pods ( three replica set will get created)

**Note: replicaset use to monitor the pod and if one of the pod is deleted replicaset use to create new pod**

kubectl delete pod myapp-replicaset-thhrt

**NOTE: replica set will not allow to create pod with same label**

kubectl get pods -o wide ( will give info about node)

**If we updata replicaset**

kubectl replace -f deployment.yaml

Machine generated alternative text:
$ kubectl scale --rep1icas:3 -f deployment .yaml 
replicaset . apps/myapp-replicaset scaled 
$ kubect get replicaset 
kubect: command not found 
$ kubectl get replicaset 
AGE 
22m 
NAME 
DESIRED 
CURRENT 
3 
READY 
3 
myapp-replicaset 3 

Kubectl delete replicaset myapp-replicaset

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

Deployment

**To check what is up: kubectl get all**

**Create deployment:** kubectl create -f deployment.yaml --record

**Check status:**kubectl rollout status deployment/myapp-deployment (as soon as deployment is created rollout is the process to see deploying container in the background)

**Will create 1 revision** : kubectl rollout history deployment/myapp-deployment---------will tell about revision

**Upgrade**: kubectl apply -f deployment.yaml

Check: kubectl describe deployment --record

**Will create revision1 and revision 2 :** kubectl rollout status deployment/myapp-deployment

Rollout: kubectl rollout undo deployment/myapp-deployment

**NOTE:**

When we rollout revision 2 nd revision 3 will display not revision 1

Machine generated alternative text:
$ kubectl rollout undo deployment/myapp-deployment 
deployment. apps/myapp-deployment rolled back 
$ kubectl rollout history deployment/myapp-deployment 
deployment. apps/myapp- deployment 
REVISION CHANGE-CAUSE 
-record—true 
-record—true 
- -filename—deployment. yaml 
- -filename—deployment . yaml 
2 
3 
kubectl apply 
kubectl create 

Machine generated alternative text:
Tern-inal 
portal 
STATUS 
Crash «"BackOff 
C rash 
1Backoff 
RESTMTS 
•aster $ kubectl get BMs 
s •2cnIs 
s. Ip2b4 
nginx-64f497f8fd-.fsss 

Ready: first part indicate:---- running container in pod/ total number of container in pod

Kubectl edit pod redis

**If you are not given a pod definition file**, you may extract the definition to a file using the below command:

kubectl get pod <pod-name> -o yaml > pod-definition.yaml

Scale replica set

Machine generated alternative text:
Scale 
kubectl replace 
kubectl scale 
kubectl scale 
-f replicaset-definition.yml 
replicaset—definition. ym1 
apiVersion: apps/ v 1 
kind: Replica-Set 
me tada ta : 
name : myapp—replicaset 
labels : 
app: myapp 
type: front—end 
spec : 
template : 
metadata : 
name : myapp—pod 
labels : 
app: myapp 
type: front—end 
spec : 
containers : 
— name: nginx—container 
image: nglnx 
selector : 
matchLabe1 s : 
-replicas:6 
-—rep1icasz6 
-f replicaset-definition.yml 
replicaset myapp-replicaset 
TYPE 
NAME 

Machine generated alternative text:
commands 
kubectl create —f replicaset-definition. yml 
kubectl get replicaset 
kubectl delete replicaset myapp-replicaset 
kubectl replace -f replicaset-definiticn.ym_ 
kubectl scale -replicas-6 -f replicaset-detÅnit 
*Also deletes all underlying PODS 

=================================replicaset================================

* 1. Kubectl get pods
  2. Kubectl get replicaset
  3. Kubectl describe replicaset "replicaset name"
  4. Kubectl delete replicaset "replicaset name"

If we change replicaset it will not effect to already created pod

* 1. Kubectl get replicaset "replicaset name" -o yaml > rs.yaml ( will generate by kubernetes)

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

Deployment

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

Certification Tip: Imperative Commands

While you would be working mostly the declarative way - using definition files, imperative commands can help in getting one time tasks done quickly, as well as generate a definition template easily. This would help save a considerable amount of time during your exams.

Before we begin, familiarize with the two options that can come in handy while working with the below commands:

--dry-run: By default as soon as the command is run, the resource will be created. If you simply want to test your command, use the --dry-run 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 the screen.

Use the above two in combination to generate a resource definition file quickly, that you can then modify and create resources as required, instead of creating the files from scratch.

====================== --dry-run -o yaml ( to generate yaml file)

POD

**Create an NGINX Pod**

kubectl run nginx --image=nginx --restart=Never

**Generate POD Manifest YAML file (-o yaml). Don't create it(--dry-run)**

kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml

Deployment

**Create a deployment**

kubectl create deployment --image=nginx nginx

**Generate Deployment YAML file (-o yaml). Don't create it(--dry-run)**

kubectl create deployment --image=nginx nginx --dry-run -o yaml

--generator=deployment/v1beta1 is deprecated as of Kubernetes 1.16. The recommended way is to use the kubectl create option instead.

**IMPORTANT:**

kubectl create deployment does not have a --replicas option. You could first create it and then scale it using the kubectl scale command.

**Save it to a file - (If you need to modify or add some other details)**

kubectl create deployment --image=nginx nginx --dry-run -o yaml > nginx-deployment.yaml

You can then update the YAML file with the replicas or any other field before creating the deployment.

Service

**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 -o yaml

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

Or

kubectl create service clusterip redis --tcp=6379:6379 --dry-run -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)So it does not work very well if your pod has a different label set. So generate the file and modify the selectors before creating the service)

**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 --dry-run -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 -o yaml

(This will not use the pods labels as selectors)

Both the above commands have their own challenges. While one of it cannot accept a selector the other cannot accept a node port. I would recommend going with the `kubectl expose` command. If you need to specify a node port, generate a definition file using the same command and manually input the nodeport before creating the service.

**Reference:**

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

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

Certification Tip: Formatting Output with kubectl

The default output format for all **kubectl** commands is the human-readable plain-text format.

The -o flag allows us to output the details in several different formats.

**kubectl [command] [TYPE] [NAME] -o <output\_format>**

Here are some of the commonly used formats:

* 1. -o jsonOutput a JSON formatted API object.
  2. -o namePrint only the resource name and nothing else.
  3. -o wideOutput in the plain-text format with any additional information.
  4. -o yamlOutput a YAML formatted API object.

Here are some useful examples:

* 1. **Output with JSON format:**

* 1. master $ kubectl create namespace test-123 --dry-run -o json
  2. {
  3. "kind": "Namespace",
  4. "apiVersion": "v1",
  5. "metadata": {
  6. "name": "test-123",
  7. "creationTimestamp": null
  8. },
  9. "spec": {},
  10. "status": {}
  11. }
  12. master $
  13. **Output with YAML format:**

* 1. master $ kubectl create namespace test-123 --dry-run -o yaml
  2. apiVersion: v1
  3. kind: Namespace
  4. metadata:
  5. creationTimestamp: null
  6. name: test-123
  7. spec: {}
  8. status: {}
  + **Output with wide (additional details):**

Probably the most common format used to print additional details about the object:

* + master $ kubectl get pods -o wide
  + NAME      READY   STATUS    RESTARTS   AGE     IP          NODE     NOMINATED NODE   READINESS GATES
  + busybox   1/1     Running   0          3m39s   10.36.0.2   node01   <none>           <none>
  + ningx     1/1     Running   0          7m32s   10.44.0.1   node03   <none>           <none>
  + redis     1/1     Running   0          3m59s   10.36.0.1   node01   <none>           <none>
  + master $

For more details, refer:

[**https://kubernetes.io/docs/reference/kubectl/overview/**](https://kubernetes.io/docs/reference/kubectl/overview/)

[**https://kubernetes.io/docs/reference/kubectl/cheatsheet**](https://kubernetes.io/docs/reference/kubectl/cheatsheet)**/**

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

**Imperative command**

1)Deploy a pod named nginx-pod using the nginx:alpine image.

Use the command kubectl run --generator=run-pod/v1 nginx-pod --image=nginx:alpine

2)

Deploy a redis pod using the redis:alpine image with the labels set to tier=db.

3)

Create a deployment named webapp using the image kodekloud/webapp-color with 3 replicas

Use the command kubectl create deployment webapp --image=kodekloud/webapp-color. The scale the webapp to 3 using command kubectl scale deployment/webapp --replicas=3

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

Namespace

Machine generated alternative text:
Switch 
dev 
kubectl get pods 
- -namespace-dev 
default 
kubectl get pods 
prod 
kubectl get pods 
- -namespace-dev 
- -namespace-prod 
kubectl config set-context $(kubectl config current-context) 
kubectl get pods 
kubectl get pods 
-namespace-prod 
-namespace:default 
kubectl get pods 
kubectl config set-context S(kubectl 
config current-context) 
- -namespace-prod 
127 people bookmarked this moment 
kubectl get pods 
kubectl get pods 
- -namespace-dev 

Kubectl get namespaces

How many pods exist in the 'research' namespace?

kubectl get pods --namespace=research

**Create a pod in finance namespace**

kubectl run --generator=run-pod/v1 redis --image=redis --namespace=finance

Which namespace has the 'blue' pod in it?

kubectl get pods --all-namespaces

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

Configmap, secret

**To create :serviceaccount**

kubectl create serviceaccount dashboard-sa

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

Docker command

docker login -u="saurakes" -p="+zl5lP+wJ9tcEA4i6GB3slh7OGWi8kYCc3VPvugDVuP0XNe1XJ0R2p0b/xsS320b" containers.cisco.com

docker build -t api08:test09 .

docker image ls

docker run --rm --env suitename=lifecycle api08:test09

docker login -u="saurakes" -p="+zl5lP+wJ9tcEA4i6GB3slh7OGWi8kYCc3VPvugDVuP0XNe1XJ0R2p0b/xsS320b" containers.cisco.com