--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=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 the screen.

$ alias k=kubectl : doing this will allow us to use only k instead of kubectl every time.

$ do="--dry-run=client -o yaml" : then you can use it like this > k create configmap example --from-literal=a=apple $do > my-resource.yaml

------------------------------------------------------------------------------------------------------------------------------------------------------------

**Ultimate cmd.**

$ kubectl run nginx --image=nginx --restart=Never --port=80 --namespace=myname --command --serviceaccount=mysa1

--env=HOSTNAME=local --labels=bu=finance,env=dev --requests='cpu=100m,memory=256Mi'

--limits='cpu=200m,memory=512Mi' --dry-run -o yaml -- /bin/sh -c 'echo hello world'

-------------------------------------------------------------------------------------------------------------------------------------------------------

$ kubectl get nodes : to get total number of nodes.

$ kubectl version : to check version

$ kubectl get nodes -o wide : to see flavour and version of Operating System on which the Kubernetes nodes are running

$ kubectl get all : to see status of all (pods, deployment, replicaset)

$ kubectl get all -o wide : to see more details of everything

$ kubectl get all --all-namespaces : to get everything from all name namespaces

-------------------------------------------------------------------------------------------------------------------------------------------------------

$ kubectl run nginx - -image=nginx : it deploys a docker container (of nginx) by creating a pod.

$ kubectl run nginx - -image=nginx - -dry-run=client -o yaml > pod.yaml : it will create and run nginx container and also will create .yaml file “pod.yaml” which will store this info in it.

-----------------------------------------------------------------------------------------------------------------------------------------------------

$ kubectl get pod: to see list of pods

$ kubectl get pods –show-labels : to see labels as well

$ kubectl get pods --all-namespaces : List all pods in all namespaces

$ kubectl delete pod: to delete pod.

$ k delete -f my-resource.yaml --force=true : to delete pod instantaneously

$ kubectl describe pod nginx : to see more detail

$ kubectl describe pod “name of pod” | grep -i image : to see underlying image directly.

$ kubectl get pod <pod-name> -o yaml > pod-definition.yaml : to extract the definition to a file

$ kubectl get pods -o wide : detail info

$ kubectl create -f “.yaml file name” : to create a pod from .yaml file

$ kubectl edit pod nginx : to edit the value of pod

-------------------------------------------------------------------------------------------------------------------------------------------------------

$ kubectl create -f “replicaset-definition.yml” : to create replication set file

$ kubectl edit replicaset “name of replicaset” : to edit replicaset file

$ kubectl scale replicaset “name of replicaset” - - replicas=2 : to scale up or down the number of pods.

$ kubectl replace -f replicaset-definition.yml : to replace the old file with updated file.

-------------------------------------------------------------------------------------------------------------------------------------------------------

Deployment, Rollout, Rollback

$ kubectl create -f deployment-definition.yml

$ kubectl create deployment nginx --image=nginx --dry-run=client -o yaml > nginx-deployment.yaml : create deployment and save the info in **.yaml** file

$ kubectl get deployments : to see list of deployments

$ kubectl describe deployment “name of deployment” | grep -i image : to see the underlying image

$ kubectl scale deployment “name of deployment” - - replicas=2 : to scale deployment

$ kubectl rollout status deployment/myapp-deployment : to see the status of rollout

$ kubectl rollout history deployment/myapp-deployment : to check history

$ kubectl rollout undo deployment/myapp-deployment : incase if we want to roll back to the previous version.

$ k rollout undo deployment nginx --to-revision=2 : to roll back to specific revision

$ k autoscale deployment nginx --min=5 --max=10 --cpu-percentage=80 : autoscaling

$ kubectl create -f deployment-definition.yaml --record : when you put --record it will store the change cause which can be seen afterwards using rollout history cmd.

$ k set image deploy/nginx1 nginx=nginx:1.15.10 : to update the image directly

-------------------------------------------------------------------------------------------------------------------------------------------------------

Service:

$ kubectl create -f service-definition.yaml

$ kubectl get services

$ kubectl describe services myapp-service

$ kubectl expose deployment simple-webapp-deployment --name=webapp-service --target-port=8080 --type=NodePort --port=8080 --dry-run=client -o yaml > svc.yaml : to create services directly and log all the info in .yaml file

$ k run tmp --rm -i --image=nginx:alpine -- curl sun-srv.sun:9999 : here tmp pod is getting created in default namespace , but the service (sun-srv) is in different namespace which is “sun”…there for when we want to talk with this svc which is in different namespace we have to do this : --curl sun-srv.sun:9999 , where .sun represents the different namespace and 9999 is service port, and sun-srv is the service name.

--------------------------------------------------------------------------------------------------------------------------------------------------------

$ minikube service “name of the service” --url : to get the IP address of the running service

$ k exec api-deployment-12345678-12345 -- nc -zv api-service 3333 : to check if deployment can talk to specified service or not

--------------------------------------------------------------------------------------------------------------------------------------------------------

Nmaespaces:

$ kubectl create -f namespace-dev.yaml

$ kubectl get pods –namespace=kube-system : to get the pods of specific namespace

$ kubectl create -f pod-definition.yml –namespace=dev : to create pod in specific namespace

$ kubectl config set-contex $(kubectl config current contex) –namespace=dev : to move from ‘default’ namespace to ‘dev’

$ Kubectl config set-context —current —namespace=prod : same as above

$ kubectl get pods –all-namespaces : to get all pods across all the namespaces

$ kubectl create -f compute-quota.yaml : to create quota .

---------------------------------------------------------------------------------------------------------------------------------------------------------

Configmaps:

$ kubectl create configmap <config-name> --from-literal=<key>=<value> : to create configmap in imperative way

$ kubectl create -f config-map.yaml

$ kubectl get configmaps

$ kubectl describe configmaps

------------------------------------------------------------------------------------------------------------------------------------------------------

Secretes:

$ kubectl create secret generic <secret-name> --from-literal=<key>=<value> : Imperative

$ kubectl create -f secret-data.yaml : declarative

$ echo -n ‘mysql’ | base64 : to encode values

$ echo -n ‘bxhdfh=’ | base64 --decode: to decode values

$ kubectl get secret

$ kubectl describe secrets

-----------------------------------------------------------------------------------------------------------------------------------------------------

Security context:

$ kubectl exec ubuntu-sleeper – whoami : to check the who is the user running cmds on container

$ kubectl exec webapp -- cat /log/app.log : to exec cat cmd on container

-----------------------------------------------------------------------------------------------------------------------------------------------------

Service Account:

$ kubectl create serviceaccount dashboard-sa : to create a service account named dashboard-sa

$ kubectl describe serviceaccount dashboard-sa

------------------------------------------------------------------------------------------------------------------------------------------------------

Taints & Tolerations:

$ kubectl taint nodes “node-name” key=value:taint-effect : to add taint on node

$ kubectl taint nodes “node-name” key=value:taint-effect- : to remove taint from node (put minus sign in the end)

-----------------------------------------------------------------------------------------------------------------------------------------------------

Node-Selector:

$ kubectl label nodes “node-name” “key=value”

----------------------------------------------------------------------------------------------------------------------------------------------------

Container Logging :

$ kubectl logs -f “pod-name” : to view the logs of pod

$ kubectl exec “pod-nam” -- cat “file-path” : to view logs using file path

$ kubectl logs -f “pod-name” -c “container-name”: incase it has multiple container running in same pod

----------------------------------------------------------------------------------------------------------------------------------------------------

Monitoring

git clone <https://github.com/kubernetes-incubator/metrics-server.git> : to clone the repo

kubectl create -f deploy/1.8+/ : to create performance metrics

kubectl top node : to see CPU , memory consumption of each node

kubectl top pods : to see CPU , memory consumption of each pods

----------------------------------------------------------------------------------------------------------------------------------------------------

Ingress

$ Kubectl create -f ingress-wear.yaml

$ kubectl get ingress

--------------------------------------------------------------------------------------------------------------------------------------------------

Selector, labels

$ kubectl get pods –selector env=dev : to list all the pod with specific selector value.

$ k get pod nginx –show-label : to see labels associated with nginx pod

----------------------------------------------------------------------------------------------------------------------------------------------------

Storage Class:

$ kubectl get sc : to get all the storage class

-----------------------------------------------------------------------------------------------------------------------------------------------------

Contexts :

$ kubectl config get-contexts : to get the current cluster/namespace name

$ kubectl config use-context “cluster name” : to use different cluster

-----------------------------------------------------------------------------------------------------------------------------------------------------

Logs and Events :

$ k logs ‘pod name’

$ k logs ‘pod-name’ > /home/opt/log : to store logs directly into the give file location

$ k logs pod1 | sudo tee /opt/etc/log.txt : to overcome write restriction in some files.

$ k get events -o wide –field-selector=involvedObject.name=’pod-name’ : to get events related to pod.

$ k get pod -o custom-colums="POD\_NAME:.metadata.name,POD\_STATUS:.status.containerStatuses[].state" : to list pod with custom columns