Pods:

* *How to edit pod; (webapp pod will rename my-new-pod)*

kubectl get pod webapp -o yaml > my-new-pod.yaml

*Edit the pod by vim my-new-pod then delete the existing pod*

kubectl delete pod webapp

*Then create a new pod with the edited file*

kubectl create -f my-new-pod.yaml

* NOTE: TO delete the pod faster, kubectl delete pod mypod --force

Replicaset:

* kubectl create -f replicaset-definition.yml
* kubectl get replicaset
* kubectl delete replicaset myapp-replicaset
* kubectl replace -f replicaset-definition.yml
* kubectl scale -replicas=6 -f replicaset-definition.yml
* kubectl edit replicaset new-replica-set
* kubectl get all (pods, replicas, deployments)

Deployment:

* kubectl edit deployment my-deployment
* *Deploy a redis pod using the redis:alpine image with the labels set to tier=db*

kubectl run redis --image=redis:alpine --dry-run=client -o yaml > redis-pod.yaml

Service:

* *Create service; Service: redis service , Port: 6379, Type: ClusterIP*

$ kubectl expose pod redis --port=6379 --name redis-service

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

kubectl create deployment webapp --image=kodekloud/webapp-color --replicas=3

* *Create a new pod called custom-nginx using the nginx image and expose it on container port 8080.*

kubectl run custom-nginx --image=nginx --port=8080

* *Create a new deployment called redis-deploy in the dev-ns namespace with the redis image. It should have 2 replicas.*

kubectl create deployment redis-deploy --image=redis --replicas=2 -n dev-ns

Namespace:

* kubectl create namespace dev
* kubectl create -f pod-definition.yml --namespace=dev
* kubectl get pods --namespace=dev
* kubectl get pods --all-namespaces
* kubectl run redis –image=redis -n dev *(create pod in dev, name and image redis)*

Configmap:

* *Create ConfigMap for the webapp-color POD;*

*ConfigName Name: webapp-config-map*

*Data: APP\_COLOR=darkblue*

kubectl create configmap webapp-config-map --from-literal=APP\_COLOR=darkblue

Security:

* kubectl exec mypod – whoami (to check user runs pod)

Secret:

* kubectl create secret generic db-secret

--from-literal=DB\_Host=sql01

--from-literal=DB\_User=root

--from-literal=DB\_Password=password12

DB\_Host: mysql 🡺 DB\_Host: **bxlzWw=**

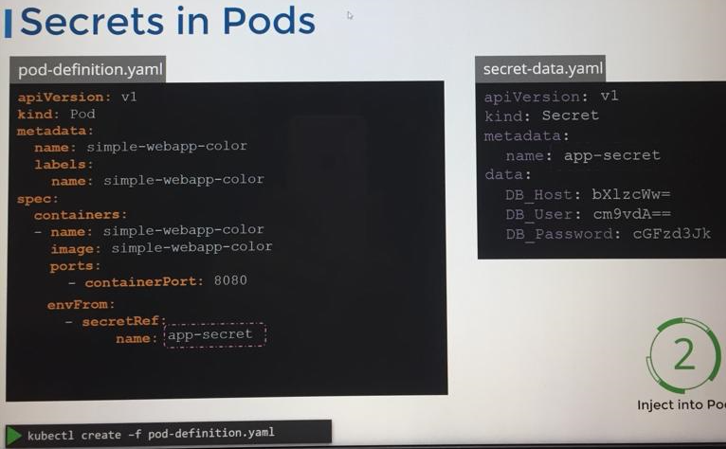
DB\_User: root 🡺 DB\_User: **klsYztk==**  
DB\_Secret: 1234 🡺 DB\_Secret: **srtrRts**

$ echo -n ‘mysql’ | base64

bxlzWw=

$ echo -n ‘bxlzWw=’ | base64 –decode

Mysql



**Taints & Tolerations:** Taints add to Node and Tolerations add to pods

Options: NoSchedule, PreferNoSchedule and NoExecute

* kubectl taint nodes node1 app=blue:NoSchedule

**Node Affinity & Labels**

* kubectl label node node01 color=blue

Create a new deployment named blue with the nginx image and 3 replicas.

* kubectl create deployment blue --image=nginx --replicas=3 --dry-run=client -o yaml > deployment.yaml

**Logging**

* Application logs stored in /log/app.log; Exec container and see the log file of ‘webapp’ pod name

$ kubectl exec webapp -- cat /log/app.log

If you delete pod, you can not see the logs

**Monitoring**

* To get the logs file pulling the repository

$ git clone <https://github.com/kodekloudhub/kubernetes-metrics-server.git>

After downloading of logs file repository to create logs we need to run

$ kubectl create -f .

To see logs of node or pod, run;

$ kubectl top node

$ kubectl top pod

**Labels, Selector and Annotations:**

* How many PODs exist in the dev environment (env)?

$ kubectl get pods --selector env=dev

**Rollout:**

$ kubectl create -f deployment.yaml

$ kubectl get deployments

$ kubectl apply -f deployment.yaml

$ kubectl set image deployment/myapp-deployment nginx=nginx.1.9.1 (Alternative way to upgrade nginx image)

$ kubectl rollout deployment/myapp-deployment

$ kubectl undo rollout deployment/myapp-deployment

$ kubectl rollout status deployment/myapp-deployment

$ kubectl rollout history deployment/myapp-deployment

**Ingress:**

$ kubectl get ingress --all-namespaces

$ kubectl describe ingress --namespace namespace1

$ kubectl edit ingress --namespace namespace1

Creating Nginx Ingress steps

$ kubectl create namespace ingress-space

$ kubectl create configmap nginx-configuration --namespace ingress-space

$ kubectl create serviceaccount ingress-serviceaccount --namespace ingress-space

Role and role binding must create

Deployment and Service Create

$ kubectl expose -n ingress-space deployment ingress-controller --type=NodePort --port=80 --name=ingress --dry-run=client -o yaml > ingress.yaml

**Kubeconfig:**

* $ kubectl config view
* To access Api Groups

$ curl http://localhost:6443 -k

* To make authentication to access Api Group two options;

$ curl http://localhost:6443 -k

--key admin.key

--cert admin.crt

--cacert ca.crt or

$ kubectl proxy

$ curl http://localhost:6443 -k

**Access Controls:**

$ kubectl auth can- i delete pods

$ kubectl auth can- i create deployments --as dev-user

* Inspect the environment and identify the authorization modes configured on the cluster.

$ kubectl describe pod kube-apiserver-controlplane -n kube-system

$ kubectl get roles (default namespace)

$ kubectl get roles --all-namespaces or kubectl get roles -A

* What are the resources the kube-proxy role in the kube-system namespace is given access to?

$ kubectl describe role kube-proxy -n kube-system

* Which account is the kube-proxy role assigned to?

$ kubectl describe rolebinding kube-proxy -n kube-system

* Create the necessary roles and role bindings required for the dev-user to create, list and delete pods in the default namespace.

To create a Role: kubectl create role developer --namespace=default --verb=list,create,delete --resource=pods  
To create a RoleBinding:- kubectl create rolebinding dev-user-binding --namespace=default --role=developer --user=dev-user

* The dev-user is unable to get details of the dark-blue-app pod in the blue namespace. Fix the issue; Role name: developer Namespace: blue

$ kubectl edit role developer -n blue