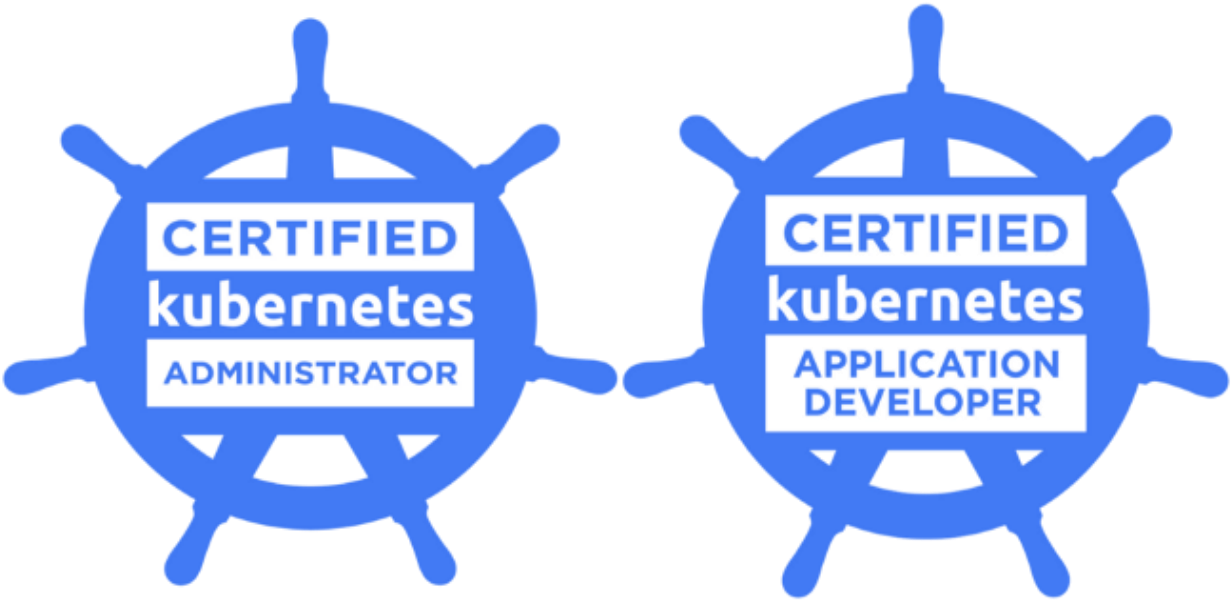
Kubernetes Master Document

(CKA & CKAD)



Content :

1.10. Check Replica Set
1.12. Describe Replica Set
1.12. List all deployment in cluster
1.13. Describe a specific deployment in cluster
1.14. Check daemonset in cluster
1.15. Describe daemonset in specific namespace
1.16. Check the events in a specific Namespace
2. Cordon and UnCordon a specific Node
3. Apply/Overwrite/Remove Label on Node
3.1. Use -L to find all nodes using disk as key , value can be any
3.2. Overwrite or change the value of Label on any Nodes
3.3. Remove the Label disk
4. Create / delete Namespace
5. Manage cluster from Local host using kubeconfig
6. Create / delete / Rollout / Rollback Deployment
7. Delete the Stuck pods in Terminating State
8. Edit deployment to change any values like maxSurge or maxUnavailable
9. Create/Delete/Verify pods using DaemonSet apply
9.1. Prepare DaemonSet yaml .
9.2. Execute DaemonSet yaml using apply
9.3. Check DaemonSet
9.4. Describe DaemonSet
9.5. Delete DaemonSet
9.6. Nodeselector Scheduling
9.7. nodeName Scheduling
10. Taint and Toleration

11. Affinity and Antiaffinity
12. Secrets and Config Maps
13. Resource Quota
14. Request and Limit
15. Edit any kubernetes resource with option -o yaml & Delete the pods after editing replicaset
16. Scale replicaSet
17. Creating static pod without replicaset and deployment
18. Find a Pod in all namespace
19. Creating pod using Imperative command using dr-run=client
20. Creating a ClusterIP service using kubectl expose
21. Creating deployment using Imperative command
22. Create a pod with container port 8080
23. Create ns,deployment using Imperative command
24. Create a httpd pod and expose it on port 80
25. Enter Sleep with timer in container command
26. Create ConfigMap using Command
27. Create Secret having multiple key using command
28. Find out user used to execute the sleep process within the ubuntu-sleeper pod ?
29. Change default user to run command in container using securityContext
30. save running Pod output to yaml file
31. securityContext capability
32. Multple capabilities in Array
33. Check existing serviceaccounts
34. Create token using serviceAccount

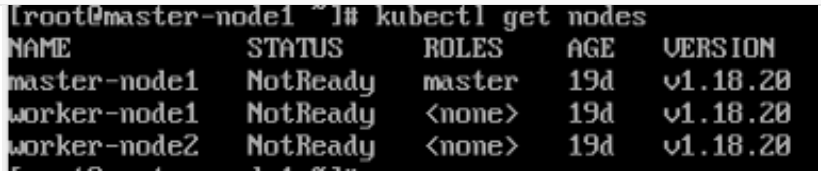
35. Check logs of a pod
36. Liveness, Readiness, and Startup Probes
37. check memory of nodes in cluster
38. InitContainer yaml
39. Find Resources with specific Multiple labels
40. Create Job using yaml and Imperative command
41. Describe job to check success and faied logs in events
42. Job finishes in 1 sec using 3 parallel job parallelism: 3
43. Schedule cron job in yaml
44. Check the TargetPort configured in specific service svc
45. Create a NodePort svc to access the web-app from outside of cluster
46. Network Policy-networkpolicy-netpol
47. Create Policy to allow pod access from outside
48. Check Ingress Resource in all namespace
49. Different Content is shown on /wear and /watch as per Backends configured in Ingress
50. Edit the key values in Ingress using edit
51. Create a new Ingress for path /pay and backend service name : pay-service
52. Roles and RoleBindings in Service Account
53. Check Logs of Container from outside of Pod (From Host)
54. Create a Persistent Volume
55. Create a pvc Persistent Volume Claim
56. STORAGE CLASS
57. create a Pod using persistentVolumeClaim
58. switch to other context using kubectl
59. make the new my-kube-config as default config file

60. Identify the authorization modes configured on the cluster.
61. Roles
62. Which account is mapped to the Role kube-proxy
63. execute command using other user — as
64. Create a Role and roleBinding to access Pods from dev-user
65. Update the Role to create Pod using deployment .
66. Count the number of clusterroles in cluster using wc
67. Add storage access to existing role michelle
68. Add NamespaceAutoProvision in — enable-admission-plugins to allow creation of namespace while creating Pod in new namespace
69. Create a TLS secret for webhook
70. Find short names for multiple resources in api-resources
71. Find major minor in kubernetes version
72. Enable the v1alpha1 version for rbac.authorization.k8s.io API group
73. change api version using kubectl-convert plugin
74. custom Resource (customresourcedefinitions)
75. Create a new custom resource using customresourcedefinitions
76. Deployment Strategy
77. decode password from secret using base64 -d
78. upgrading from Client side to server side and vice versa

1) Display Commands

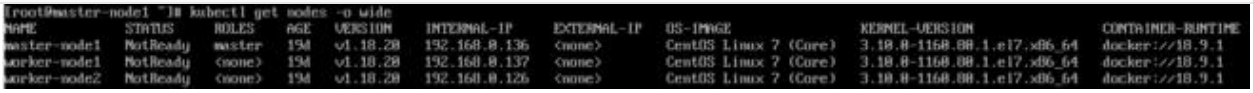
1.1) Check the Nodes connected with Master

```
kubectl get nodes
```



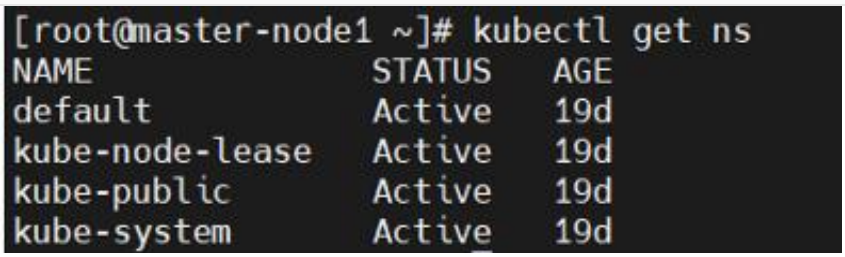
1.2) Check Details of Nodes connected with Master

```
kubectl get nodes -o wide
```



1.3) Check all the Namespace in Kubernetes Node

```
kubectl get ns
```



1.4) Check Pods running in specific Namespace “monitoring”

```
kubectl get pods -n monitoring
```

```
[root@meyclvspiapp03 ~]# kubectl get pods -n monitoring
NAME                                  READY   STATUS    RESTARTS   AGE
alertmanager-main-0                   2/2     Running   0          92d
alertmanager-main-1                   2/2     Running   0          92d
alertmanager-main-2                   2/2     Running   0          92d
blackbox-exporter-64db9f4fdc-kh84d    3/3     Running   0          92d
grafana-76b545b8bd-nflrn              1/1     Running   0          92d
kube-state-metrics-cf85bcf5b-xvnxj    3/3     Running   0          92d
node-exporter-7489q                   2/2     Running   0          92d
node-exporter-9fvw7                   2/2     Running   0          92d
node-exporter-vdbhd                   2/2     Running   0          92d
prometheus-adapter-74865dcff-dx9c2    1/1     Running   0          92d
prometheus-adapter-74865dcff-n5fz8    1/1     Running   0          92d
prometheus-k8s-0                      2/2     Running   1          43d
prometheus-k8s-1                      2/2     Running   1          43d
prometheus-operator-595699887b-gvjgv  2/2     Running   0          92d
```

1.5) Check running Pods in internal Namespace “kube-system”

```
kubectl get pods -n kube-system
```

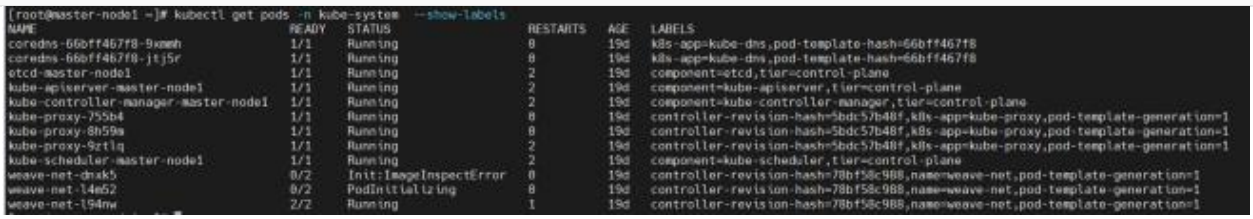
```
[root@master-node1 ~]# kubectl get pods -n kube-system
NAME                                   READY   STATUS                  RESTARTS   AGE
coredns-66bff467f8-9xmmh               1/1     Running                 0          19d
coredns-66bff467f8-jtj5r               1/1     Running                 0          19d
etcd-master-node1                      1/1     Running                 2          19d
kube-apiserver-master-node1            1/1     Running                 2          19d
kube-controller-manager-master-node1   1/1     Running                 2          19d
kube-proxy-755b4                       1/1     Running                 0          19d
kube-proxy-8h59m                       1/1     Running                 0          19d
kube-proxy-9ztlq                       1/1     Running                 2          19d
kube-scheduler-master-node1            1/1     Running                 2          19d
weave-net-dnxk5                        0/2     Init:ImageInspectError  0          19d
weave-net-l4m52                        0/2     PodInitializing         0          19d
weave-net-l94nw                        2/2     Running                 1          19d
```

```
kubectl get pods -n kube-system -o wide
```

```
[root@master-node1 ~]# kubectl get pods -n kube-system -o wide
NAME                                   READY   STATUS                  RESTARTS   AGE   IP              NODE           NOMINATED NODE   READINESS GATES
coredns-66bff467f8-9xmmh               1/1     Running                 0          19d   10.32.0.3       worker-node1   <none>           <none>
coredns-66bff467f8-jtj5r               1/1     Running                 0          19d   10.32.0.2       worker-node1   <none>           <none>
etcd-master-node1                      1/1     Running                 2          19d   192.168.0.136   master-node1   <none>           <none>
kube-apiserver-master-node1            1/1     Running                 2          19d   192.168.0.136   master-node1   <none>           <none>
kube-controller-manager-master-node1   1/1     Running                 2          19d   192.168.0.136   master-node1   <none>           <none>
kube-proxy-755b4                       1/1     Running                 0          19d   192.168.0.126   worker-node2   <none>           <none>
kube-proxy-8h59m                       1/1     Running                 0          19d   192.168.0.137   worker-node1   <none>           <none>
kube-proxy-9ztlq                       1/1     Running                 2          19d   192.168.0.136   master-node1   <none>           <none>
kube-scheduler-master-node1            1/1     Running                 2          19d   192.168.0.136   master-node1   <none>           <none>
weave-net-dnxk5                        0/2     Init:ImageInspectError  0          19d   192.168.0.136   master-node1   <none>           <none>
weave-net-l4m52                        0/2     PodInitializing         0          19d   192.168.0.126   worker-node2   <none>           <none>
weave-net-l94nw                        2/2     Running                 1          19d   192.168.0.137   worker-node1   <none>           <none>
```

1.6) Check the Labels attached to Pods

```
kubectl get pods -n kube-system  --show-labels
```



1.7) Describe specific Node

```
kubectl describe nodes worker-node1
```

```
[root@master-node1 ~]# kubectl describe nodes worker-node1
Name:               worker-node1
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=worker-node1
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Sun, 20 Nov 2022 14:43:35 -0500
Taints:             node.kubernetes.io/unreachable:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  worker-node1
  AcquireTime:     <unset>
  RenewTime:       Sun, 20 Nov 2022 15:11:41 -0500
Conditions:
  Type                 Status    LastHeartbeatTime                 LastTransitionTime
  ----                 ------    -----------------                 ------------------
  NetworkUnavailable   False     Sun, 20 Nov 2022 14:46:02 -0500   Sun, 20 Nov 2022 14:46:02 -0500
  MemoryPressure       Unknown   Sun, 20 Nov 2022 15:08:41 -0500   Sat, 10 Dec 2022 02:33:59 -0500
  DiskPressure         Unknown   Sun, 20 Nov 2022 15:08:41 -0500   Sat, 10 Dec 2022 02:33:59 -0500
  PIDPressure          Unknown   Sun, 20 Nov 2022 15:08:41 -0500   Sat, 10 Dec 2022 02:33:59 -0500
  Ready                Unknown   Sun, 20 Nov 2022 15:08:41 -0500   Sat, 10 Dec 2022 02:33:59 -0500
Addresses:
  InternalIP:  192.168.0.137
```

1. 8) Check cluster-info

```
kubectl cluster-info
```

```
[root@master-node1 .kube]# kubectl cluster-info
Kubernetes master is running at https://192.168.0.136:6443
KubeDNS is running at https://192.168.0.136:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

1.9) check API resources KIND name for YAML deployment

```
kubectl api-resources | less
```

```
[root@master-node1 .kube]# kubectl api-resources  | less
NAME                              SHORTNAMES   APIGROUP                       NAMESPACED   KIND
bindings                                                                      true         Binding
componentstatuses                 cs                                          false        ComponentStatus
configmaps                        cm                                          true         ConfigMap
endpoints                         ep                                          true         Endpoints
events                            ev                                          true         Event
limitranges                       limits                                      true         LimitRange
namespaces                        ns                                          false        Namespace
nodes                             no                                          false        Node
persistentvolumeclaims            pvc                                         true         PersistentVolumeClaim
persistentvolumes                 pv                                          false        PersistentVolume
pods                              po                                          true         Pod
podtemplates                                                                  true         PodTemplate
replicationcontrollers            rc                                          true         ReplicationController
resourcequotas                    quota                                       true         ResourceQuota
secrets                                                                       true         Secret
serviceaccounts                   sa                                          true         ServiceAccount
services                          svc                                         true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io   false        MutatingWebhookConfiguration
validatingwebhookconfigurations                admissionregistration.k8s.io   false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io           false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io         false        APIService
controllerrevisions                            apps                           true         ControllerRevision
daemonsets                        ds           apps                           true         DaemonSet
deployments                       deploy       apps                           true         Deployment
replicasets                       rs           apps                           true         ReplicaSet
statefulsets                      sts          apps                           true         StatefulSet
tokenreviews                                   authentication.k8s.io          false        TokenReview
localsubjectaccessreviews                      authorization.k8s.io           true         LocalSubjectAccessReview
selfsubjectaccessreviews                       authorization.k8s.io           false        SelfSubjectAccessReview
selfsubjectrulesreviews                        authorization.k8s.io           false        SelfSubjectRulesReview
subjectaccessreviews                           authorization.k8s.io           false        SubjectAccessReview
horizontalpodautoscalers          hpa          autoscaling                    true         HorizontalPodAutoscaler
cronjobs                          cj           batch                          true         CronJob
jobs                                           batch                          true         Job
certificatesigningrequests        csr          certificates.k8s.io            false        CertificateSigningRequest
```

1.10) Check Replica Set

```
kubectl get rs | grep api
```

```
[root@meyclvspiapp03 ~]# kubectl get rs | grep api
firewall-api-77fbf4b5bb                      1         1         1       17d
```

1.11) Describe Replica Set

```
kubectl describe rs firewall-api-77fbf4b5bb
```

```
[root@meyclvspiapp03 ~]# kubectl describe rs firewall-api-77fbf4b5bb
Name:           firewall-api-77fbf4b5bb
Namespace:      default
Selector:       app=firewall-api,pod-template-hash=77fbf4b5bb,release=stable
Labels:         app=firewall-api
                pod-template-hash=77fbf4b5bb
                release=stable
Annotations:    deployment.kubernetes.io/desired-replicas: 1
                deployment.kubernetes.io/max-replicas: 2
                deployment.kubernetes.io/revision: 1
Controlled By:  Deployment/firewall-api
Replicas:       1 current / 1 desired
Pods Status:    1 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=firewall-api
           pod-template-hash=77fbf4b5bb
           release=stable
  Init Containers:
   datacontainer:
    Image:      my.registry:5000/firewall-api:v1.23.6.9
    Port:       <none>
    Host Port:  <none>
    Command:
```

1.12) List all deployment in cluster

```
kubectl get deployment
kubectl get deployment | grep api
```

```
[root@meyclvspiapp03 ~]# kubectl get deployment | grep api
firewall-api                    1/1     1            1           17d
```

1.13) Describe a specific deployment in cluster

```
kubectl describe deployment firewall-api
```

```
[root@meyclvspiapp03 ~]# kubectl describe deployment firewall-api
Name:                   firewall-api
Namespace:              default
CreationTimestamp:      Thu, 24 Nov 2022 08:16:22 +0400
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=firewall-api,release=stable
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=firewall-api
           release=stable
  Init Containers:
   datacontainer:
    Image:      my.registry:5000/firewall-api:v1.23.6.9
    Port:       <none>
    Host Port:  <none>
```

1.14) Check daemonset in cluster

```
kubectl get daemonset -n kube-system
```

```
[root@master-node1 .kube]# kubectl get daemonset -n kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-proxy   3         3         1       3            1           kubernetes.io/os=linux   20d
weave-net    3         3         0       3            0           <none>                   20d
```

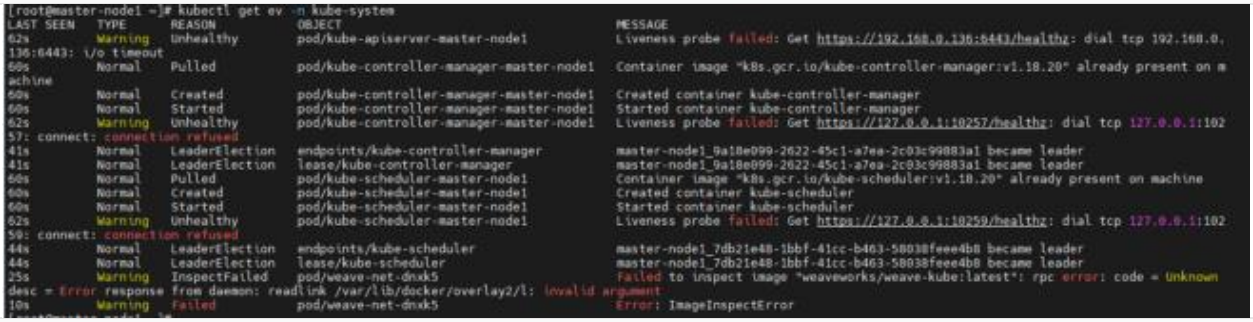
1.15) Describe daemonset in specific namespace

```
kubectl describe daemonset weave-net -n kube-system
```

```
[root@master-node1 ~]# kubectl describe daemonset weave-net -n kube-system
Name:           weave-net
Selector:       name=weave-net
Node-Selector:  <none>
Labels:         name=weave-net
Annotations:    deprecated.daemonset.template.generation: 1
Desired Number of Nodes Scheduled: 3
Current Number of Nodes Scheduled: 3
Number of Nodes Scheduled with Up-to-date Pods: 3
Number of Nodes Scheduled with Available Pods: 0
Number of Nodes Misscheduled: 0
Pods Status:  1 Running / 2 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:           name=weave-net
  Service Account:  weave-net
  Init Containers:
   weave-init:
    Image:      weaveworks/weave-kube:latest
    Port:       <none>
    Host Port:  <none>
    Command:
      /home/weave/init.sh
    Environment:  <none>
    Mounts:
```

1.16 Check the events in a specific Namespace

```
kubectl get ev -n kube-system
```




```
kubectl get ev -n kube-system | grep -i failed
```

2) **Cordon and UnCordon a specific Node**

This command will lock the given node for creation of new Pods via scheduler . This is normally used during Maintenance

```
kubectl cordon worker-node2
kubectl get nodes
kubectl uncordon worker-node2
```

```
[root@master-node1 ~]# kubectl cordon worker-node2
node/worker-node2 cordoned
[root@master-node1 ~]# kubectl get nodes
NAME           STATUS                        ROLES    AGE   VERSION
master-node1   NotReady                      master   19d   v1.18.20
worker-node1   NotReady                      <none>   19d   v1.18.20
worker-node2   NotReady,SchedulingDisabled   <none>   19d   v1.18.20
```

```
[root@master-node1 ~]# kubectl uncordon worker-node2
node/worker-node2 uncordoned
[root@master-node1 ~]# kubectl get nodes
NAME           STATUS     ROLES    AGE   VERSION
master-node1   NotReady   master   19d   v1.18.20
worker-node1   NotReady   <none>   19d   v1.18.20
worker-node2   NotReady   <none>   19d   v1.18.20
```

3) **Apply/Overwrite/Remove Label on Node**

3.1) Use -L to find all nodes using disk as key , value can be any

```
kubectl label node worker-node2 disk=ssd
kubectl get nodes -l disk=ssd
kubectl get nodes -L disk
```

```
[root@master-node1 ~]# kubectl label node worker-node2 disk=ssd
node/worker-node2 labeled
[root@master-node1 ~]#
[root@master-node1 ~]# kubectl get nodes -l disk=ssd
NAME           STATUS     ROLES    AGE   VERSION
worker-node2   NotReady   <none>   19d   v1.18.20
[root@master-node1 ~]# kubectl get nodes -L disk
NAME           STATUS     ROLES    AGE   VERSION    DISK
master-node1   NotReady   master   19d   v1.18.20
worker-node1   NotReady   <none>   19d   v1.18.20
worker-node2   NotReady   <none>   19d   v1.18.20   ssd
```

3.2) Overwrite or change the value of Label on any Nodes

```
kubectl label node worker-node2 disk=sata --overwrite
```

3.3) Remove the Label disk

```
kubectl label node worker-node2 disk-
```

```
[root@master-node1 ~]# kubectl label node worker-node2 disk-
node/worker-node2 labeled
[root@master-node1 ~]# kubectl get nodes -L disk
NAME           STATUS     ROLES    AGE   VERSION    DISK
master-node1   NotReady   master   19d   v1.18.20
worker-node1   NotReady   <none>   19d   v1.18.20
worker-node2   NotReady   <none>   19d   v1.18.20
```

4) **Create / delete Namespace**

```
kubectl create ns techtracker
kubectl get ns techtracker
```

```
[root@master-node1 .kube]# kubectl create ns techtracker
namespace/techtracker created
[root@master-node1 .kube]# kubectl get ns techtracker
NAME          STATUS   AGE
techtracker   Active   14s
```

5) Manage cluster from Local host using kubeconfig

```
mkdir -p $HOME/.kube
cd $HOME/.kube/
vi config
copy paste the content of /etc/kubernetes/admin.conf and save
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

6) Create / delete / Rollout / Rollback Deployment

6.1) Prepare Deployment YAML

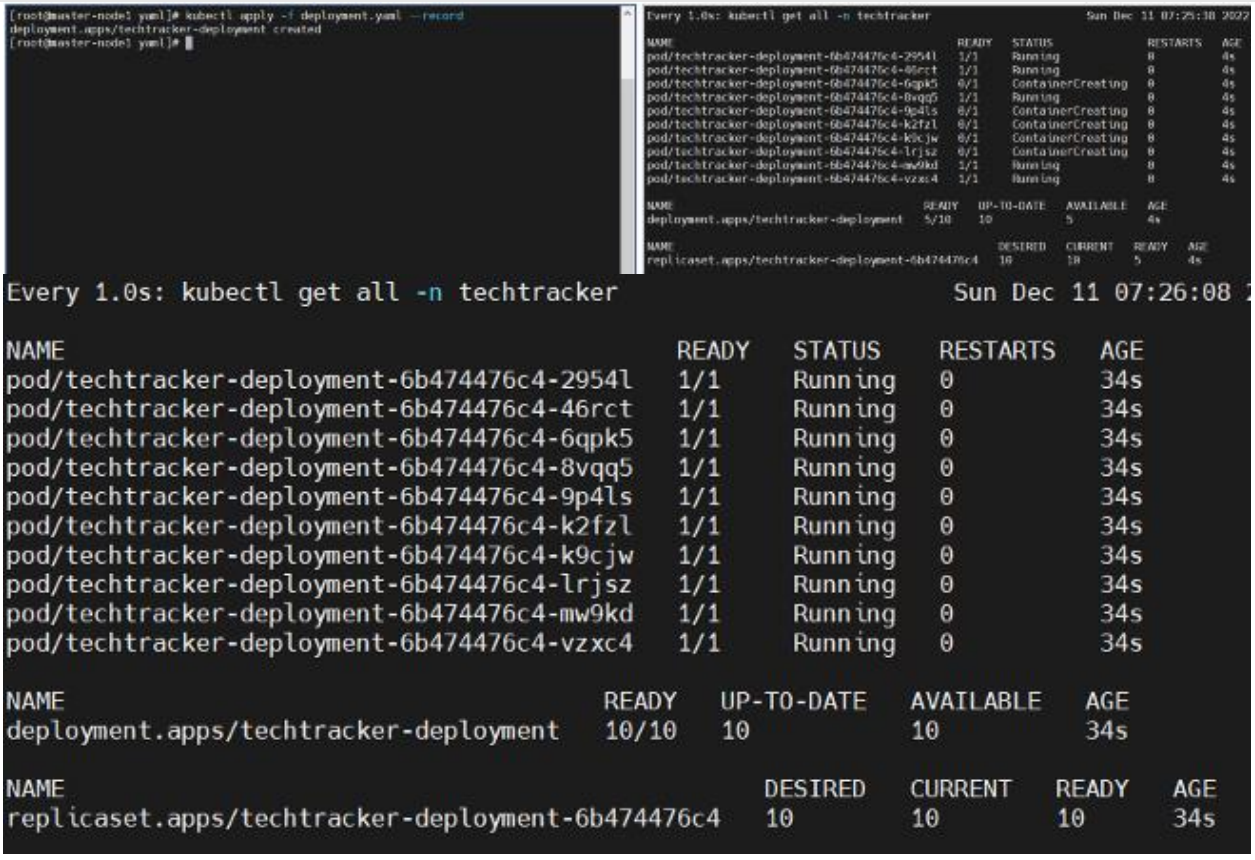
```
vi deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: techtracker-deployment
  namespace: techtracker
  labels:
    app: nginx
    env: qa
spec:
  replicas: 10
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

6.2) Open 2nd vertical terminal and execute live watch to see pods getting created

```
# On Terminal 2
watch -n 1 kubectl get all -n techtracker
```

6.3) Create Deployment using Apply :

```
# On Terminal 1
kubectl apply -f deployment.yaml
```



```
kubectl get deployment techtracker-deployment -n techtracker
```

```
[root@master-node1 yaml]# kubectl get deployment techtracker-deployment -n techtracker
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
techtracker-deployment   10/10   10           10          107s
```

6.4) Check the Container Image version is correct :

```
kubectl describe pod techtracker-deployment-6b474476c4-8vqq5 -n techtracker |
grep -i image
```

```
[root@master-node1 yaml]# kubectl describe pod techtracker-deployment-6b474476c4-8vqq5 -n techtracker | grep -i image
    Image:          nginx:1.14.2
    Image ID:       docker-pullable://nginx@sha256:f7988fb6c02e0ce69257d9bd9cf37ae20a60f1df7563c3a2a6abe24160306b8d
  Normal  Pulled     5m48s  kubelet            Container image "nginx:1.14.2" already present on machine
```

6.5) Check the revision no. in rollout history of deployment

```
kubectl rollout history  deployment techtracker-deployment -n techtracker
```

```
[root@master-node1 ~]# kubectl rollout history  deployment techtracker-deployment -n techtracker
deployment.apps/techtracker-deployment
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=deployment.yaml --record=true
```

6.6) Check the Rolling Update Strategy of deployment

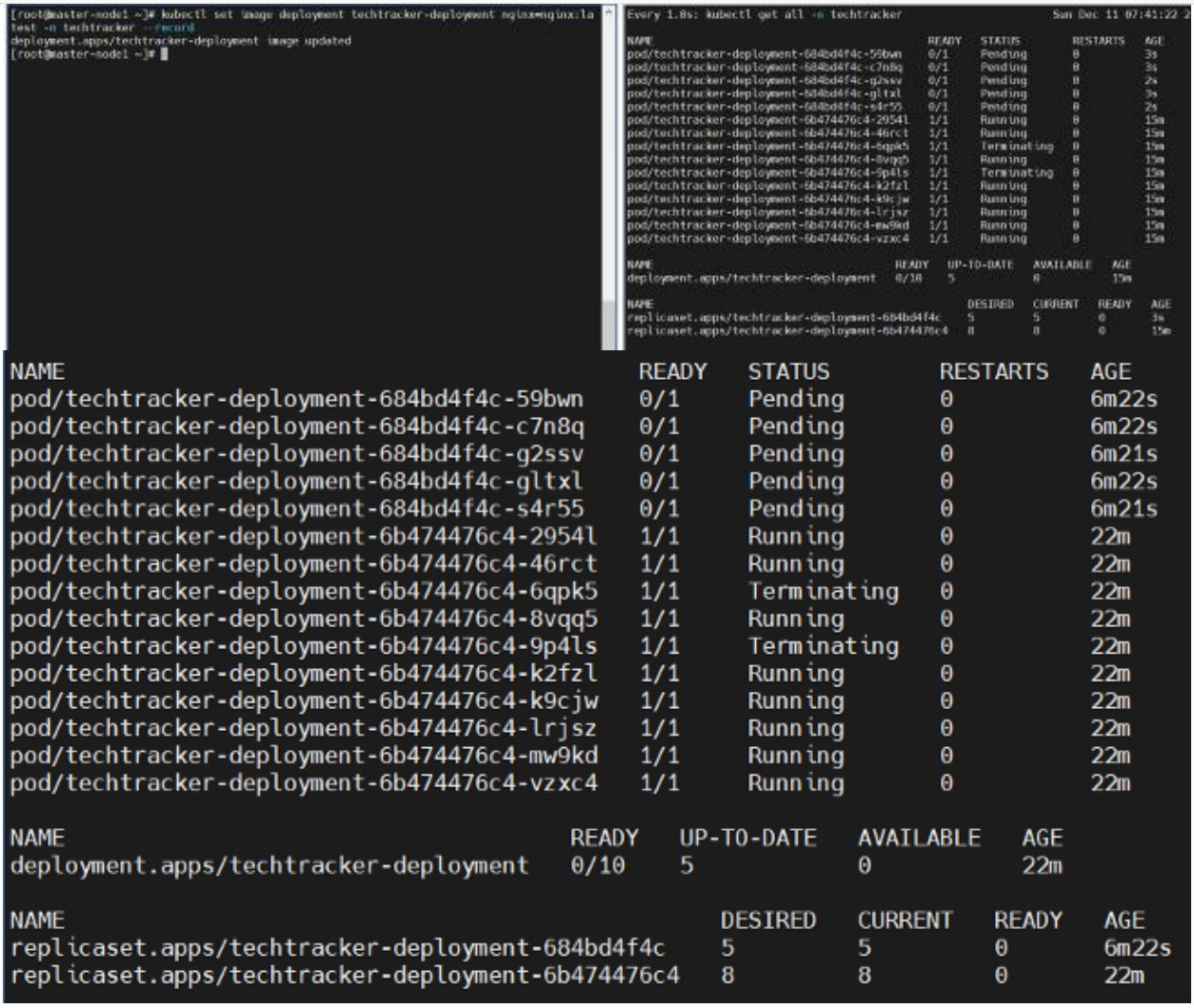
```
spec:
  minReadySeconds: 20
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      name: webapp
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
```

```
 kubectl describe deployment techtracker-deployment -n techtracker | grep -i
RollingUpdateStrategy
```

```
[root@master-node1 ~]# kubectl describe deployment techtracker-deployment -n techtracker
| grep -i RollingUpdateStrategy
RollingUpdateStrategy:  25% max unavailable, 25% max surge
```

6.7) Set latest image in deployment rollout

```
kubectl set image deployment techtracker-deployment nginx=nginx:latest -n
techtracker --record
```



```
NAME                                        READY   STATUS        RESTARTS   AGE
pod/techtracker-deployment-684bd4f4c-59bwn  0/1     Pending       0          6m22s
pod/techtracker-deployment-684bd4f4c-c7n8q  0/1     Pending       0          6m22s
pod/techtracker-deployment-684bd4f4c-g2ssv  0/1     Pending       0          6m21s
pod/techtracker-deployment-684bd4f4c-gltxl  0/1     Pending       0          6m22s
pod/techtracker-deployment-684bd4f4c-s4r55  0/1     Pending       0          6m21s
pod/techtracker-deployment-6b474476c4-2954l 1/1     Running       0          22m
pod/techtracker-deployment-6b474476c4-46rct 1/1     Running       0          22m
pod/techtracker-deployment-6b474476c4-6qpk5 1/1     Terminating   0          22m
pod/techtracker-deployment-6b474476c4-8vqq5 1/1     Running       0          22m
pod/techtracker-deployment-6b474476c4-9p4ls 1/1     Terminating   0          22m
pod/techtracker-deployment-6b474476c4-k2fzl 1/1     Running       0          22m
pod/techtracker-deployment-6b474476c4-k9cjw 1/1     Running       0          22m
pod/techtracker-deployment-6b474476c4-lrjsz 1/1     Running       0          22m
pod/techtracker-deployment-6b474476c4-mw9kd 1/1     Running       0          22m
pod/techtracker-deployment-6b474476c4-vzxc4 1/1     Running       0          22m

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/techtracker-deployment  0/10    5            0           22m

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/techtracker-deployment-684bd4f4c  5         5         0       6m22s
replicaset.apps/techtracker-deployment-6b474476c4 8         8         0       22m
```

```
Every 1.0s: kubectl get all -n techtracker

NAME                                                READY   STATUS    RESTARTS   AGE
pod/techtracker-deployment-684bd4f4c-59bwn          1/1     Running   0          14m
pod/techtracker-deployment-684bd4f4c-6f2gp          1/1     Running   0          2m59s
pod/techtracker-deployment-684bd4f4c-79z4b          1/1     Running   0          3m1s
pod/techtracker-deployment-684bd4f4c-c7n8q          1/1     Running   0          14m
pod/techtracker-deployment-684bd4f4c-g2ssv          1/1     Running   0          14m
pod/techtracker-deployment-684bd4f4c-gltxl          1/1     Running   0          14m
pod/techtracker-deployment-684bd4f4c-kql9v          1/1     Running   0          3m3s
pod/techtracker-deployment-684bd4f4c-ngqfx          1/1     Running   0          2m59s
pod/techtracker-deployment-684bd4f4c-s4r55          1/1     Running   0          14m
pod/techtracker-deployment-684bd4f4c-vjl4g          1/1     Running   0          3m1s
```

6.8) Check the 2nd Revision added in rollout history:

```
kubectl rollout history  deployment techtracker-deployment -n techtracker
```

```
[root@master-node1 ~]# kubectl rollout history  deployment techtracker-deployment -n tech
tracker
deployment.apps/techtracker-deployment
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=deployment.yaml --record=true
2         kubectl set image deployment techtracker-deployment nginx=nginx:latest --namesp
ace=techtracker --record=true
```

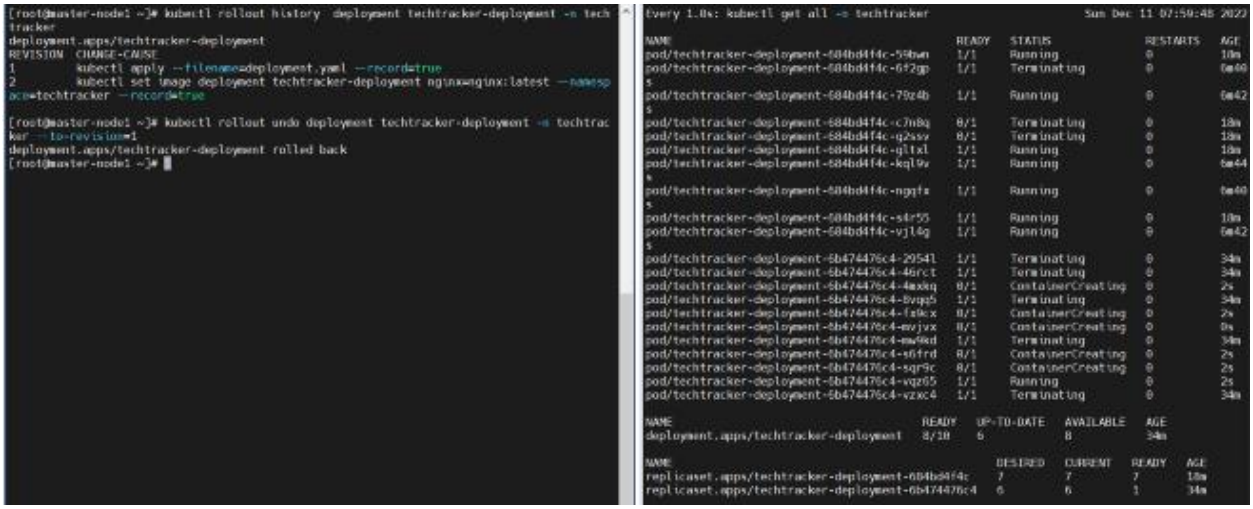
6.9) Check/Verify the Container Image should be Latest :

```
kubectl describe pod techtracker-deployment-6b474476c4-8vqq5 -n techtracker |
grep -i image
```

```
[root@master-node1 ~]# kubectl describe pod techtracker-deployment-684bd4f4c-59bwn -n techtracker | grep -i image
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:ab589a3c466e347b1c0573be23356676df90cd7ce2dbf6ec332a5f0a8b5e59db
  Normal   Pulling          2m16s                kubelet            Pulling image "nginx:latest"
  Normal   Pulled           119s                 kubelet            Successfully pulled image "nginx:latest"
```

6.10) Rollback to a specific old version using -to-revision=1

```
kubectl rollout undo deployment techtracker-deployment -n techtracker --to-
revision=1
```

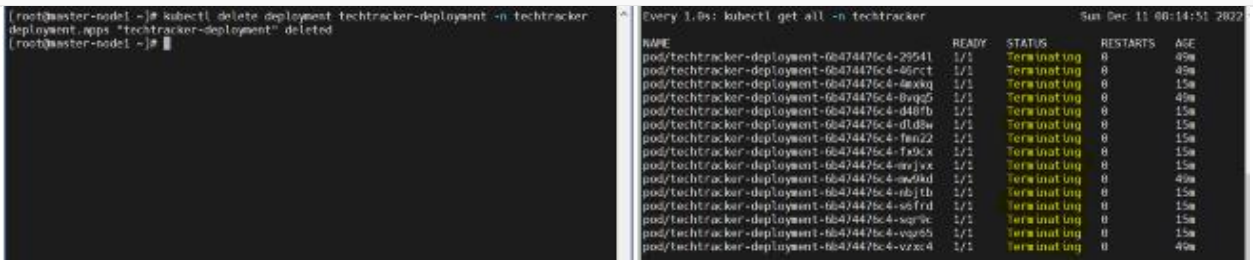


6.11) Check and verify the container version after rollback

```
kubectl describe pod techtracker-deployment-6b474476c4-dld8w -n techtracker | grep -i image
```

6.12) Delete a existing deployment

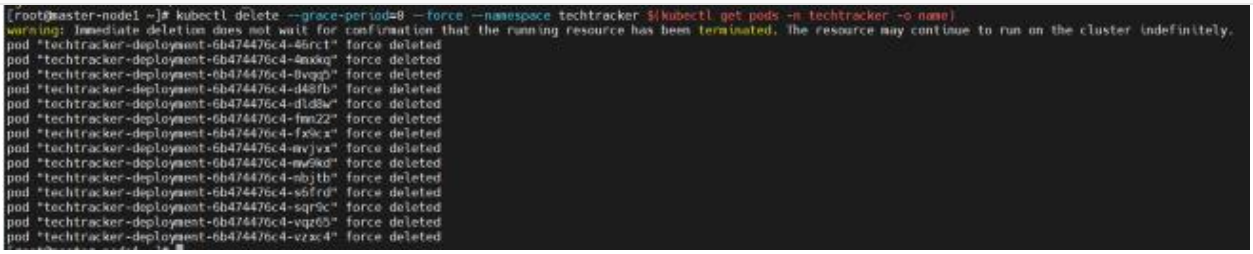
```
kubectl delete deployment techtracker-deployment -n techtracker
```



7) Delete the Stuck pods in Terminating State

```
[root@master-node1 ~]# kubectl get all  -n techtracker
NAME                                              READY   STATUS        RESTARTS   AGE
pod/techtracker-deployment-6b474476c4-46rct   1/1     Terminating   0          74m
pod/techtracker-deployment-6b474476c4-4mxkq   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-8vqq5   1/1     Terminating   0          74m
pod/techtracker-deployment-6b474476c4-d48fb   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-dld8w   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-fmn22   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-fx9cx   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-mvjvx   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-mw9kd   1/1     Terminating   0          74m
pod/techtracker-deployment-6b474476c4-nbjtb   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-s6frd   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-sqr9c   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-vqz65   1/1     Terminating   0          39m
pod/techtracker-deployment-6b474476c4-vzxc4   1/1     Terminating   0          74m
```

```
kubectl delete --grace-period=0 --force --namespace techtracker $(kubectl get pods -n techtracker -o name)
```



8) Edit deployment to change any values like maxSurge or maxUnavailable

```
kubectl edit deployment techtracker-deployment -n techtracker
```

```
strategy:
  rollingUpdate:
    maxSurge: 40%
    maxUnavailable: 10%
  type: RollingUpdate
template:
  metadata:
    creationTimestamp: null
```

9) **Create/Delete/Verify pods using DaemonSet apply**

9.1) Prepare DaemonSet yaml .

```
vi daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: techtracker-daemonset
  namespace: techtracker
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

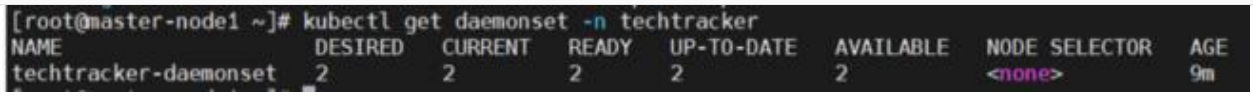
9.2) Execute DaemonSet yaml using apply

```
kubectl apply -f  daemonset.yaml
watch -n 1 kubectl get all -n techtracker -o wide
```

nginx pod created on each worker node.

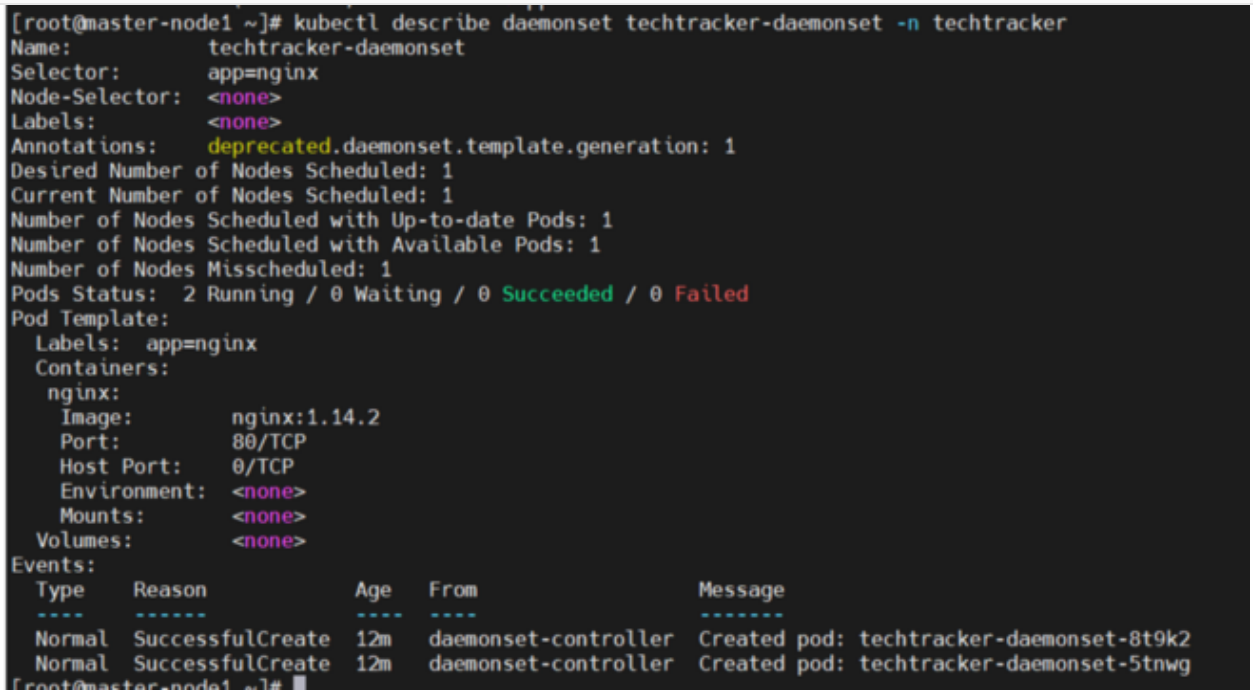
9.3) Check DaemonSet

```
kubectl get daemonset -n techtracker
```



9.4) Describe DaemonSet

```
kubectl describe daemonset techtracker-daemonset -n techtracker
```



9.5) Delete DaemonSet

```
kubectl delete daemonset techtracker-daemonset -n techtracker
```

9.6) Nodeselector Scheduling

```
kubectl label node worker-node1 env=test
kubectl label node worker-node2 env=prod
```

```
[root@master-node1 ~]# kubectl get nodes
NAME           STATUS     ROLES    AGE   VERSION
master-node1   NotReady   master   23d   v1.18.20
worker-node1   NotReady   <none>   23d   v1.18.20
worker-node2   NotReady   <none>   23d   v1.18.20
[root@master-node1 ~]# kubectl label node worker-node1 env=test
node/worker-node1 labeled
[root@master-node1 ~]# kubectl label node worker-node2 env=prod
node/worker-node2 labeled
```

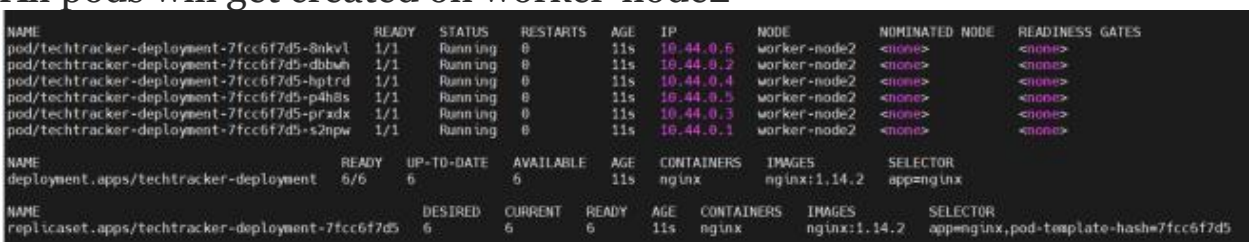
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: techtracker-deployment
  namespace: techtracker
spec:
  replicas: 6
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
      nodeSelector:
       env: test
kubectl apply -f schedulor_deployment.yaml
```

All pods created on worker-node1 because we had set nodeSelector env:test and env:test label on worker-node1

9.7) nodeName Scheduling

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: techtracker-deployment
  namespace: techtracker
spec:
  replicas: 6
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
      nodeName: worker-node2
 kubectl apply -f  schedulor_deployment.yaml
```

All pods will get created on worker-node2



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

10) Taint and Toleration

By Default Scheduler never created pods on tainted node . But if we still want some heavy or specific application to be deployed on tainted node then we can use toleration in deplomentfile for that application .

- Cordon cannot be ByPassed
- Taint can be ByPassed using Toleration

Check the tainted Nodes in Cluster

```
kubectl describe no master-node1 -n techtracker | grep -i taint
```

```
[root@master-node1 ~]# kubectl describe no master-node1 -n techtracker | grep -i taint
Taints:             node.kubernetes.io/not-ready:NoExecute
```

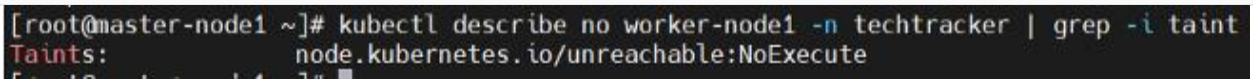
taint of Effect — NoExecute is applied at master node

Apply taint on node1

```
kubectl taint node worker-node1 dubai=hot:NoSchedule
```

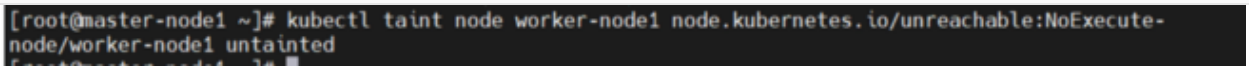
```
[root@master-node1 ~]# kubectl taint node worker-node1 dubai=hot:NoSchedule
node/worker-node1 tainted
[root@master-node1 ~]#
```

```
kubectl describe no worker-node1 -n techtracker | grep -i taint
```

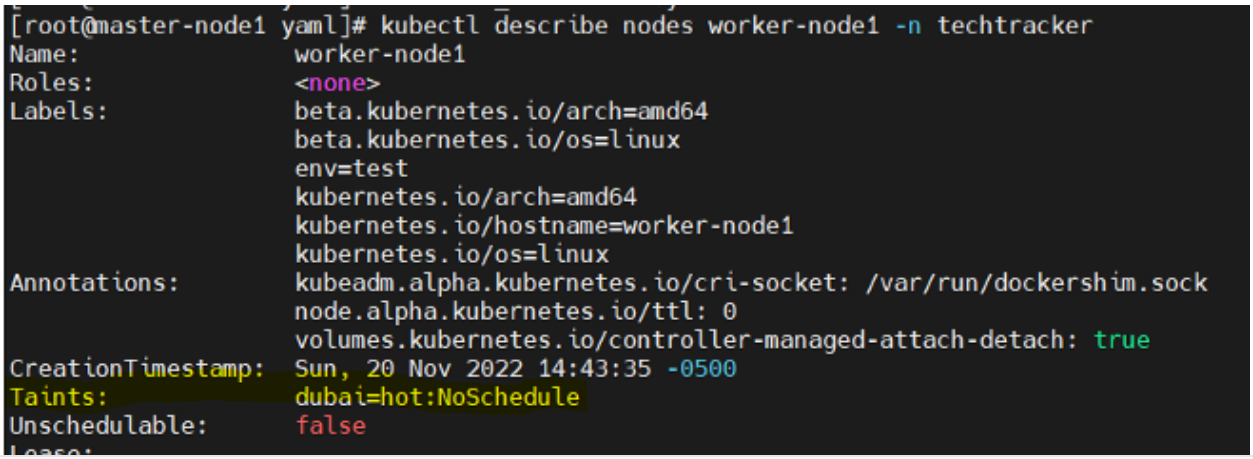


#Remove taint from node

```
kubectl taint node worker-node1 node.kubernetes.io/unreachable:NoExecute-
```



Create nodes on tainted node




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: techtracker-deployment
  namespace: techtracker
spec:
  replicas: 6
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
```

```
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
      tolerations:
        - key: "dubai"
          operator: "Equal"
          value: "hot"
          effect: "NoSchedule"
~
kubectl apply -f  taint_toleration.yaml
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
pod/techtracker-deployment-79f8f98c5f-9h855	1/1	Running	0	19s	10.32.0.2	worker-node1
pod/techtracker-deployment-79f8f98c5f-dd5nl	1/1	Running	0	19s	10.32.0.4	worker-node1
pod/techtracker-deployment-79f8f98c5f-dstbl	1/1	Running	0	19s	10.32.0.6	worker-node1
pod/techtracker-deployment-79f8f98c5f-dtbbx	1/1	Running	0	19s	10.32.0.7	worker-node1
pod/techtracker-deployment-79f8f98c5f-mnfhp	1/1	Running	0	19s	10.32.0.5	worker-node1
pod/techtracker-deployment-79f8f98c5f-vsn9v	1/1	Running	0	19s	10.32.0.3	worker-node1

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

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

11. Affinity and Antiaffinity

Apply label app=store on which customer want to deploy redis pod

```
kubectl label node worker-node1 app=store
kubectl label node worker-node2 app=store
kubectl get nodes -L app
```

```
[root@master-node1 yaml]# kubectl label node worker-node1 app=store
node/worker-node1 labeled
[root@master-node1 yaml]# kubectl get nodes -L app
NAME           STATUS     ROLES    AGE   VERSION    APP
master-node1   NotReady   master   28d   v1.18.20
worker-node1   Ready      <none>   28d   v1.18.20   store
worker-node2   Ready      <none>   28d   v1.18.20
[root@master-node1 yaml]# kubectl label node worker-node2 app=store
node/worker-node2 labeled
[root@master-node1 yaml]# kubectl get nodes -L app
NAME           STATUS     ROLES    AGE   VERSION    APP
master-node1   NotReady   master   28d   v1.18.20
worker-node1   Ready      <none>   28d   v1.18.20   store
worker-node2   Ready      <none>   28d   v1.18.20   store
[root@master-node1 yaml]#
```

Apply Antiaffinity yaml and check expected output

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-cache
  namespace: techtracker
spec:
  selector:
    matchLabels:
      app: store
  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: redis-server
        image: redis:3.2-alpine
```

```
kubectl apply -f antiaffinity.yaml
kubectl get all -n techtracker
```

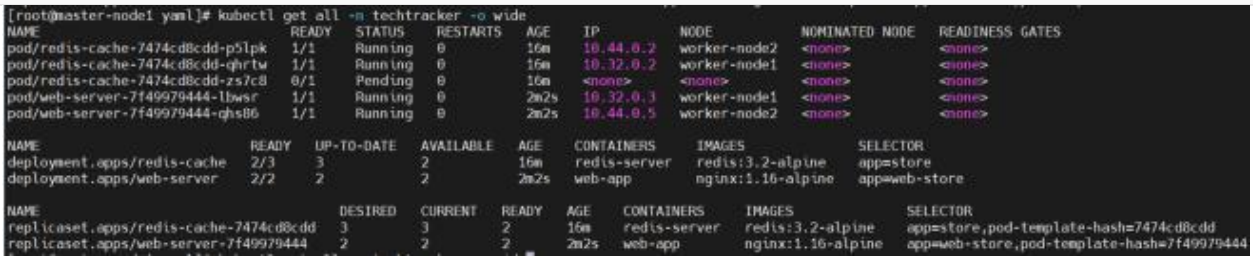
```
[root@master-node1 yaml]# kubectl get all -n techtracker -o wide
NAME                                    READY   STATUS    RESTARTS   AGE    IP           NODE           NOMINATED NODE   READINESS GATES
pod/redis-cache-7474cd8cdd-p5lpk        1/1     Running   0          4m7s   10.44.0.2    worker-node2   <none>           <none>
pod/redis-cache-7474cd8cdd-qhrtw        1/1     Running   0          4m7s   10.32.0.2    worker-node1   <none>           <none>
pod/redis-cache-7474cd8cdd-zs7c8        0/1     Pending   0          4m7s   <none>       <none>         <none>           <none>

NAME                          READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS     IMAGES            SELECTOR
deployment.apps/redis-cache   2/3     3            2           4m7s   redis-server   redis:3.2-alpine   app=store

NAME                                  DESIRED   CURRENT   READY   AGE    CONTAINERS     IMAGES            SELECTOR
replicaset.apps/redis-cache-7474cd8cdd   3         3         2       4m7s   redis-server   redis:3.2-alpine   app=store,pod-template-hash=7474cd8cdd
```

Its showing 2/3 and 1 pod is pending because there are only 2 nodes having label app=store and scheduler will not create 2 similar pods on 1 worker node .

Apply Affinity and Antiaffinity yaml and check output

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-server
spec:
  selector:
    matchLabels:
      app: web-store
  replicas: 3
  template:
    metadata:
      labels:
        app: web-store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - web-store
            topologyKey: "kubernetes.io/hostname"
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
```

```
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: web-app
        image: nginx:1.16-alpine
kubectl apply -f affinityantiaffinity.yaml
kubectl get all -n techtracker -o wide
```



web-server pod deployed on node where redis is already deployed

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

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

12. Secrets and Config Maps

Deployment Yaml for creating a mysql pod

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: techtracker
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:5.6
```

```
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: password
kubectl apply -f resource_secret.yaml
watch -n 1 kubectl get all -n techtracker
```

```
NAME                             READY   STATUS    RESTARTS   AGE
pod/mysql-7bc745d4f8-wnr2p       1/1     Running   0          4m34s

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql    1/1     1            1           4m34s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/mysql-7bc745d4f8    1         1         1       4m34s
```

env variable value password is visible , anyone who has namespace access can view the secret values which is not recommended. That's why we have to use secret and store values in them.

```
[root@master-node1 yaml]# kubectl describe pod mysql-7bc745d4f8-wnr2p -n techtracker
Name:         mysql-7bc745d4f8-wnr2p
Namespace:    techtracker
Priority:     0
Node:         worker-node1/192.168.0.137
Start Time:   Sat, 24 Dec 2022 07:12:18 -0500
Labels:       app=mysql
              pod-template-hash=7bc745d4f8
Annotations:  <none>
Status:       Running
IP:           10.32.0.2
IPs:
  IP:           10.32.0.2
Controlled By:  ReplicaSet/mysql-7bc745d4f8
Containers:
  mysql:
    Container ID:   docker://22403cf8a84b58f704e4049ca8b318b25babf5f667b4fcbbaa2958c8
    Image:          mysql:5.6
    Image ID:       docker-pullable://mysql@sha256:20575ecebe6216036d25dab5903808211f
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sat, 24 Dec 2022 07:12:54 -0500
    Ready:          True
    Restart Count:  0
    Environment:
      MYSQL_ROOT_PASSWORD:  password
    Mounts:
```

Create Secret

```
kubectl create secret generic mysqlsecret --from-literal=mysqlpassword=redhat
-n techtracker
kubectl get secret mysqlsecret -n techtracker
kubectl describe secret mysqlsecret -n techtracker
```

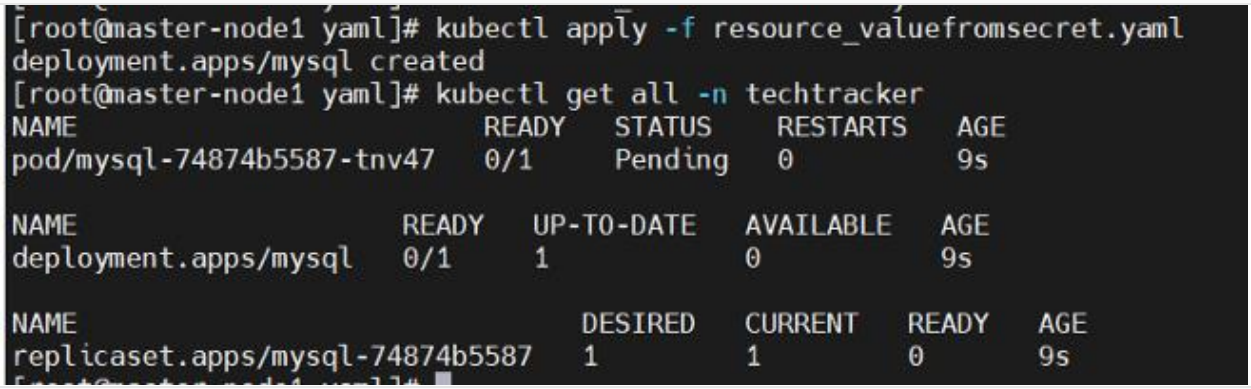
```
[root@master-node1 yaml]# kubectl create secret generic mysqlsecret --from-literal=mysqlpassword=redhat -n techtracker
secret/mysqlsecret created
[root@master-node1 yaml]# kubectl get secret mysqlsecret -n techtracker
NAME          TYPE     DATA   AGE
mysqlsecret   Opaque   1      41s
[root@master-node1 yaml]# kubectl describe secret mysqlsecret -n techtracker
Name:         mysqlsecret
Namespace:    techtracker
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
mysqlpassword:  6 bytes
```

6 bytes means 6 digits (redhat)

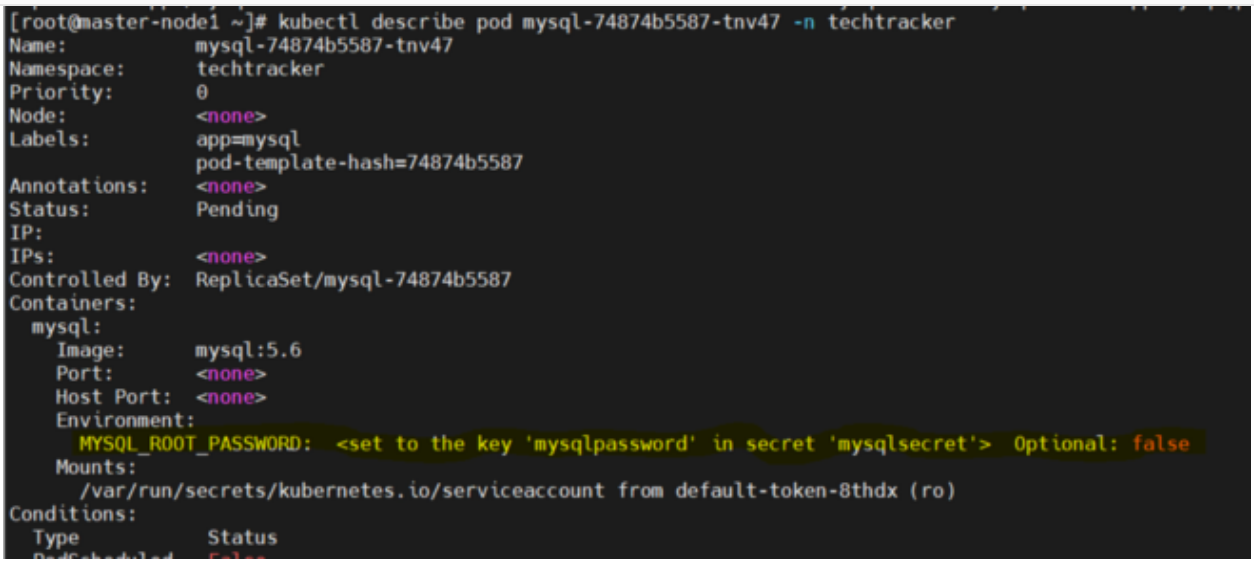
Create deployment using secret values

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: techtracker
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysqlsecret
                  key: mysqlpassword
```

```
kubectl apply -f resource_valuefromsecret.yaml
kubectl get all -n techtracker
```



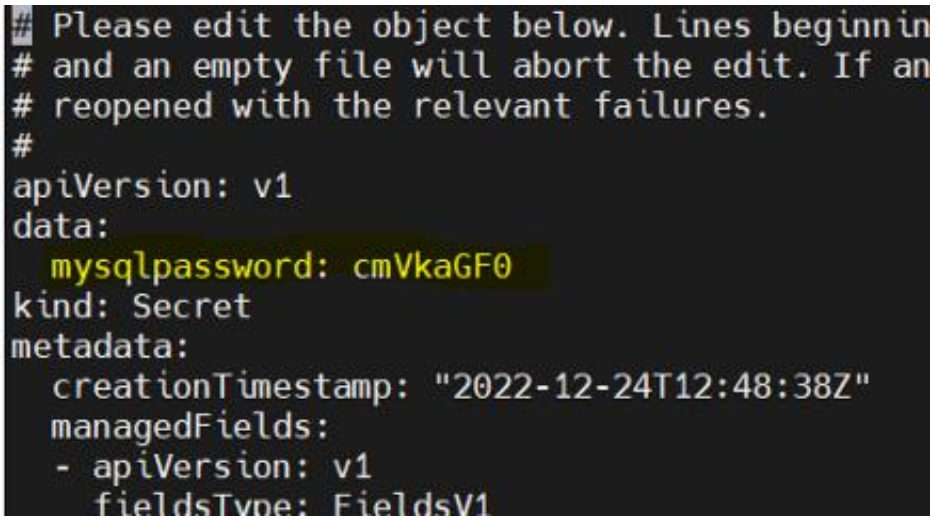

```
kubectl describe pod mysql-74874b5587-tnv47 -n techtracker
```



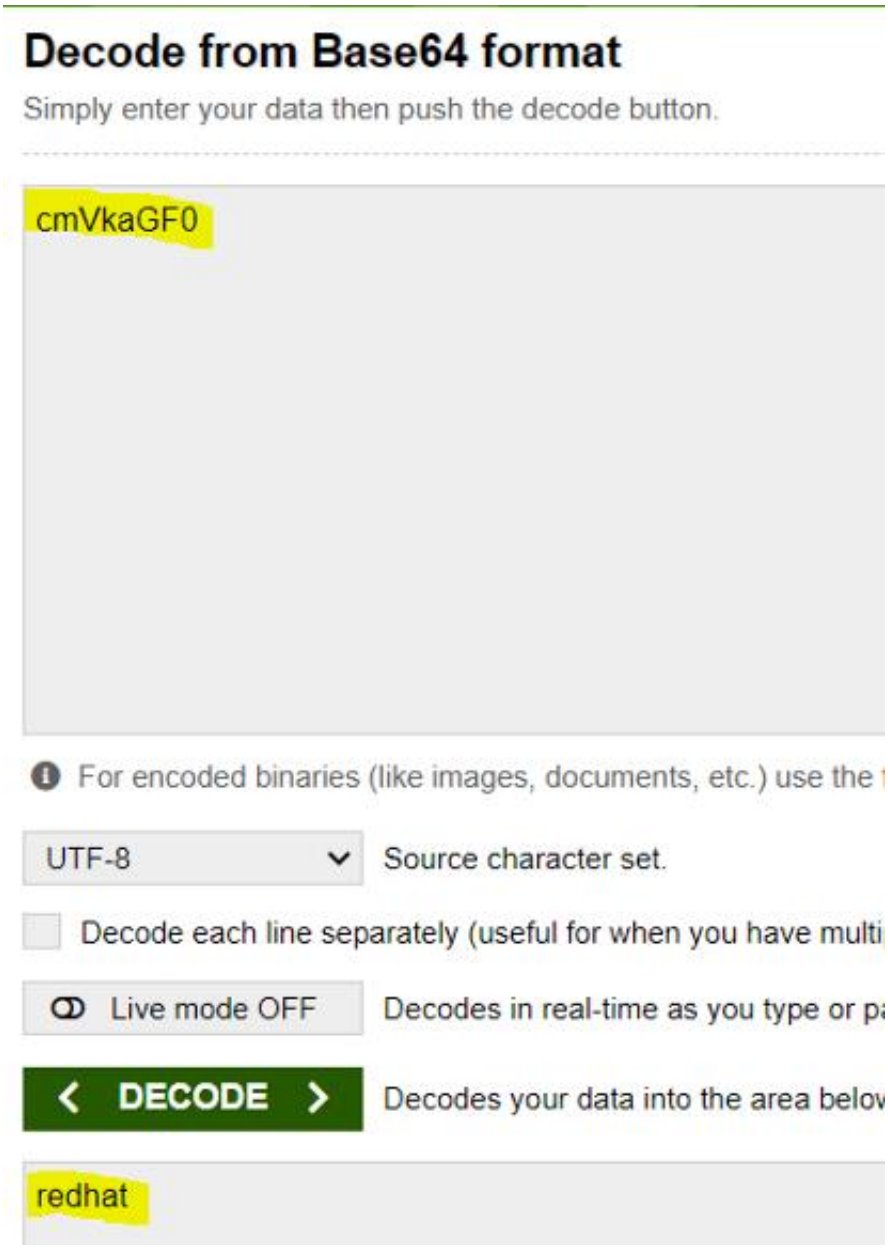
env variable value are not displayed now when we describe pod

Person having Rights to edit the secret can only view the value stored in the key .

```
kubectl edit secret mysqlsecret -n techtracker
```



By default Base64 encoded value stored in secret (Highlighted in Yellow above)



we can create secret using yaml also but value we have to put as Base64 encoded value.

```
apiVersion: apps/v1
kind: Secret
metadata:
  name: sample-secret
  namespace: techtracker
type: opaque
data:
    mysqlrootpassword: a3vhFjsakD3jKJ34HJ
```

Create Configmaps

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: techtracker
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - image: mysql:5.6
          name: mysql
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                configMapKeyRef:
                  name: mysqlcm
                  key: mysqlpassword
kubectl create configmap mysqlcm --from-literal=mysqlpassword=redhat -n
techtracker
kubectl apply -f resource_valuefromConfigmap.yaml
kubectl get all -n techtracker -o wide
kubectl describe pod mysql-859699c87c-9hkp7 -n techtracker
```

```
[root@master-node1 yaml]# kubectl apply -f resource_valuefromConfigmap.yaml
deployment.apps/mysql created
[root@master-node1 yaml]# kubectl get all -n techtracker -o wide
NAME                         READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NODE   READINESS
pod/mysql-859699c87c-9hkp7   0/1     Pending   0          4s    <none>   <none>   <none>           <none>

NAME                    READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES      SELECTOR
deployment.apps/mysql   0/1     1            0           4s    mysql        mysql:5.6   app=mysql

NAME                               DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES      SELECTOR
replicaset.apps/mysql-859699c87c   1         1         0       4s    mysql        mysql:5.6   app=mysql,pod-
[root@master-node1 yaml]# kubectl describe pod mysql-859699c87c-9hkp7 -n techtracker
Name:           mysql-859699c87c-9hkp7
Namespace:      techtracker
Priority:       0
Node:           <none>
Labels:         app=mysql
                pod-template-hash=859699c87c
Annotations:    <none>
Status:         Pending
IP:
IPs:            <none>
Controlled By:  ReplicaSet/mysql-859699c87c
Containers:
  mysql:
    Image:      mysql:5.6
    Port:       <none>
    Host Port:  <none>
    Environment:
      MYSQL_ROOT_PASSWORD:  <set to the key 'mysqlpassword' of config map 'mysqlcm'>  Optional: false
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-8thdx (ro)
```

```
kubectl describe configmap mysqlcm -n techtracker
```

In configmap we can see value using describer

but in Secret we need edit rights to see value

```
[root@master-node1 yaml]# kubectl describe configmap mysqlcm -n techtracker
Name:         mysqlcm
Namespace:    techtracker
Labels:       <none>
Annotations:  <none>

Data
====
mysqlpassword:
----
redhat
```

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

13. Resource Quota

1. How many Pods,Service,Deployment,Secret,ConfigMap we can create in a particular Namespace , Its defined under:

Namespace Quota — Project quota — object Quota — resource quota.yaml

2. How much Resource the object can consume — — project quota

3. How much resource a Pod can use and How much resource a container can use inside a Pod .

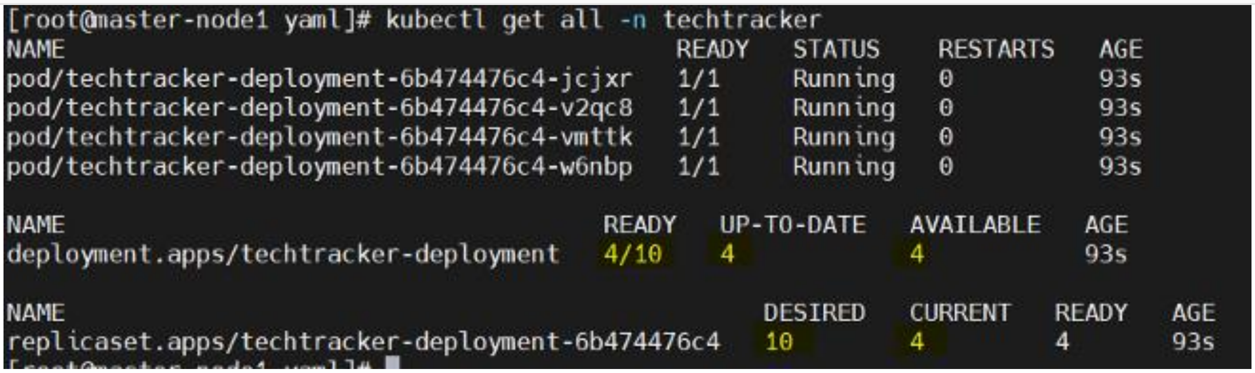
Create Resource Quota to Limit Pod Count as 4

```
apiVersion: v1
kind: ResourceQuota
metadata:
    name: pods-high
    namespace: techtracker
spec:
    hard:
      pods: "4"
kubectl create -f resource_quota.yaml
kubectl get resourcequota -n techtracker
```

```
[root@master-node1 yaml]# kubectl create -f resource_quota.yaml
resourcequota/pods-high created
[root@master-node1 yaml]# kubectl get resourcequota -n techtracker
NAME        AGE   REQUEST     LIMIT
pods-high   26s   pods: 0/4
```

Now if we try to create 10 Pods using deployment , Only 4 will be created

```
kubectl apply -f deployment.yaml
kubectl get all -n techtracker
```



```
[root@master-node1 yaml]# kubectl get all -n techtracker
NAME                                          READY   STATUS    RESTARTS   AGE
pod/techtracker-deployment-6b474476c4-jcjxr   1/1     Running   0          93s
pod/techtracker-deployment-6b474476c4-v2qc8   1/1     Running   0          93s
pod/techtracker-deployment-6b474476c4-vmttk   1/1     Running   0          93s
pod/techtracker-deployment-6b474476c4-w6nbp   1/1     Running   0          93s

NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/techtracker-deployment   4/10    4            4           93s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/techtracker-deployment-6b474476c4   10        4         4       93s
```

— —

14. Request and Limit

Requests:memory:900Mi means 900mb minimum is required at Node to launch this Pod

Limits:memory:10Gi means 10gb max can be used by container while running

```yaml
resources:
  requests:
    memory: "900Mi"
    cpu: "500m"
  limits:
    memory: "10Gi"
    cpu: "500m"
```

— —

15. Edit any kubernetes resource with option -o yaml & Delete the pods after editing replicaset

```
kubectl edit replicaset.apps/new-replica-set -o yaml
kubectl delete pod new-replica-set-7c2r9
kubectl delete pod/new-replica-set-bcwgg
kubectl delete pod/new-replica-set-nxc9r
kubectl delete pod/new-replica-set-sj77c
```

— —

16. Scale replicaSet

```
kubectl scale replicaset.apps/new-replica-set --replicas=5
kubectl scale replicaset.apps/new-replica-set --replicas=2
```

```
controlplane ~ ➜ kubectl scale replicaset.apps/new-replica-set --replicas=5
replicaset.apps/new-replica-set scaled

controlplane ~ ➜ kubectl get all
NAME                        READY   STATUS    RESTARTS   AGE
pod/new-replica-set-q7n56   1/1     Running   0          13m
pod/new-replica-set-xzvc5   1/1     Running   0          13m
pod/new-replica-set-vjdpx   1/1     Running   0          12m
pod/new-replica-set-9wmmp   1/1     Running   0          12m
pod/new-replica-set-rch75   1/1     Running   0          10s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   51m

NAME                              DESIRED   CURRENT   READY   AGE
replicaset.apps/new-replica-set   5         5         5       44m

controlplane ~ ➜ kubectl scale replicaset.apps/new-replica-set --replicas=2
replicaset.apps/new-replica-set scaled

controlplane ~ ➜ kubectl get all
NAME                        READY   STATUS        RESTARTS   AGE
pod/new-replica-set-xzvc5   1/1     Running       0          14m
pod/new-replica-set-vjdpx   1/1     Running       0          14m
pod/new-replica-set-9wmmp   1/1     Terminating   0          14m
pod/new-replica-set-q7n56   1/1     Terminating   0          14m
pod/new-replica-set-rch75   1/1     Terminating   0          104s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.43.0.1    <none>        443/TCP   53m

NAME                              DESIRED   CURRENT   READY   AGE
replicaset.apps/new-replica-set   2         2         2       45m
```

— —
— — — — — — — — — —

17. Creating static pod without replicaset and deployment

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
  namespace: finance
  labels:
    app: redis
spec:
  containers:
    - name: redis
      image: redis
~
kubectl apply -f pod.yaml
kubectl get pods -n finance
```

```
controlplane ~ ➜ kubectl get pods -n finance
NAME      READY   STATUS    RESTARTS   AGE
payroll   1/1     Running   0          12m
redis     1/1     Running   0          2m2s
```

---

18. Find a Pod in all namespace

```
kubectl get pods --all-namespaces
```

```
controlplane ~ ➜ kubectl get pods --all-namespaces
NAMESPACE       NAME                                     READY   STATUS             RESTARTS        AGE
kube-system     local-path-provisioner-7b7dc8d6f5-8q988  1/1     Running            0               22m
kube-system     coredns-b96499967-z24x5                  1/1     Running            0               22m
kube-system     helm-install-traefik-crd-m988z           0/1     Completed          0               22m
kube-system     helm-install-traefik-9qtgt               0/1     Completed          1               22m
kube-system     traefik-7cd4fcff68-xzq6w                 1/1     Running            0               19m
kube-system     svclb-traefik-mkmkm                      2/2     Running            0               19m
marketing       redis-db                                 1/1     Running            0               16m
dev             redis-db                                 1/1     Running            0               16m
marketing       blue                                     1/1     Running            0               16m
finance         payroll                                  1/1     Running            0               16m
manufacturing   red-app                                  1/1     Running            0               16m
research        dna-1                                    0/1     CrashLoopBackOff   6 (4m59s ago)   16m
kube-system     metrics-server-668d979685-xzgzd          1/1     Running            0               22m
research        dna-2                                    0/1     CrashLoopBackOff   6 (4m51s ago)   16m
finance         redis                                    1/1     Running            0               5m25s
```

```
kubectl get pods --all-namespaces -l name=blue
```

```
controlplane ~ ➜ kubectl get pods --all-namespaces -l name=blue
NAMESPACE   NAME   READY   STATUS    RESTARTS   AGE
marketing   blue   1/1     Running   0          20m
```

```
kubectl get pods nginx-pod -o yaml
```

```
controlplane ~ ➜ kubectl get pods nginx-pod -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-12-26T10:58:56Z"
  labels:
    run: nginx-pod
  name: nginx-pod
  namespace: default
  resourceVersion: "728"
  uid: e485c357-c5b3-4410-a0ac-4d94456bd69c
spec:
  containers:
  - image: nginx:alpine
    imagePullPolicy: IfNotPresent
    name: nginx-pod
```

— —

19. Creating pod using Imperative command using dr-run=client

```
kubectl run nginx-pod --image=nginx:alpine -o yaml --dry-run=client
```

```
controlplane ~ ➜ kubectl run nginx-pod --image=nginx:alpine -o yaml --dry-run=client
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-pod
  name: nginx-pod
spec:
  containers:
  - image: nginx:alpine
    name: nginx-pod
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

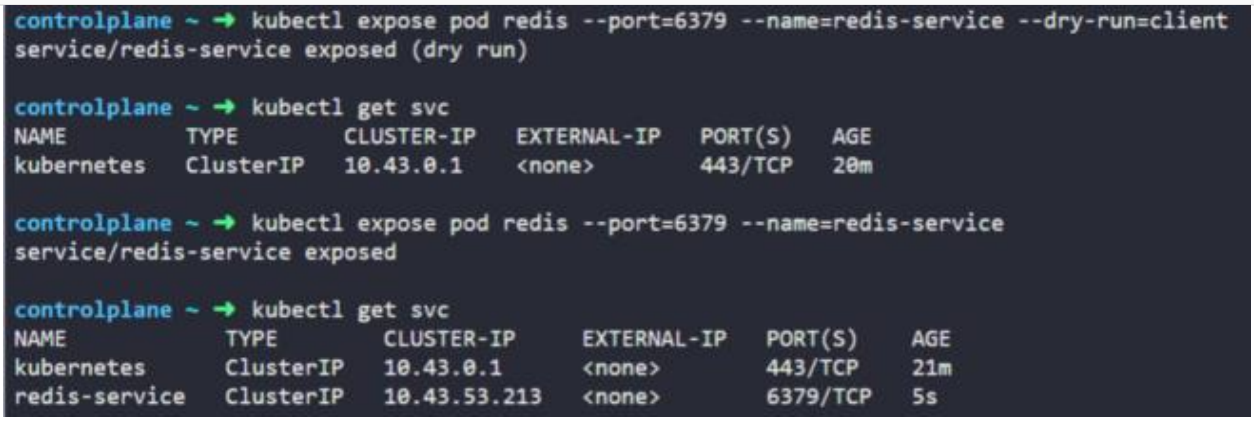
```
kubectl run redis --image=redis:alpine -l tier=db -o yaml --dry-run=client
```

```
controlplane ~ ➜ kubectl run redis --image=redis:alpine -l tier=db -o yaml --dry-run=client
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    tier: db
  name: redis
spec:
  containers:
  - image: redis:alpine
    name: redis
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

- —
— — — — — — — — -

20. Creating a ClusterIP service using kubectl expose

```
kubectl expose pod redis --port=6379 --name=redis-service --dry-run=client
kubectl get svc
kubectl expose pod redis --port=6379 --name=redis-service
kubectl get svc
```



- — -

21. Creating deployment using Imperative command

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

```
controlplane ~ ➜ kubectl create deployment webapp --image=kodekloud/webapp-color --replicas=3
deployment.apps/webapp created

controlplane ~ ➜ kubectl get all
NAME                           READY   STATUS    RESTARTS   AGE
pod/nginx-pod                  1/1     Running   0          21m
pod/redis                      1/1     Running   0          13m
pod/webapp-79c4cc587b-swzml    1/1     Running   0          14s
pod/webapp-79c4cc587b-ldn7q    1/1     Running   0          14s
pod/webapp-79c4cc587b-bbhpd    1/1     Running   0          14s

NAME                     TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/kubernetes       ClusterIP   10.43.0.1      <none>        443/TCP    29m
service/redis-service    ClusterIP   10.43.53.213   <none>        6379/TCP   8m15s

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/webapp   3/3     3            3           14s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/webapp-79c4cc587b   3         3         3       14s
```

22. Create a pod with container port 8080

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

kubectl describe pod custom-nginx | grep -i port
```

```
controlplane ~ ➜ kubectl get pods -o wide
NAME                      READY   STATUS    RESTARTS   AGE     IP            NODE
nginx-pod                 1/1     Running   0          25m     10.42.0.9     controlplane
redis                     1/1     Running   0          16m     10.42.0.10    controlplane
webapp-79c4cc587b-swzml   1/1     Running   0          3m51s   10.42.0.13    controlplane
webapp-79c4cc587b-ldn7q   1/1     Running   0          3m51s   10.42.0.12    controlplane
webapp-79c4cc587b-bbhpd   1/1     Running   0          3m51s   10.42.0.11    controlplane
custom-nginx              1/1     Running   0          24s     10.42.0.14    controlplane
```

```
controlplane ~ ✗ kubectl describe pod custom-nginx | grep -i port
    Port:          8080/TCP
    Host Port:     0/TCP
```

- — -

23. Create ns,deployment using Imperative command

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

```
controlplane ~ ➜  kubectl get all -n dev-ns
NAME                                 READY   STATUS    RESTARTS   AGE
pod/redis-deploy-f9b7599c5-vjgq2     1/1     Running   0          26s
pod/redis-deploy-f9b7599c5-c97g9     1/1     Running   0          26s

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/redis-deploy   2/2     2            2           27s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/redis-deploy-f9b7599c5    2         2         2       27s
```

- — -

24. Create a httpd pod and expose it on port 80

```
kubectl run httpd --image=httpd:alpine --port=80 --expose
kubectl get all
```

```
controlplane ~ ➜ kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/nginx-pod                 1/1     Running   0          47m
pod/redis                     1/1     Running   0          38m
pod/webapp-79c4cc587b-swzml   1/1     Running   0          25m
pod/webapp-79c4cc587b-ldn7q   1/1     Running   0          25m
pod/webapp-79c4cc587b-bbhpd   1/1     Running   0          25m
pod/custom-nginx              1/1     Running   0          22m
pod/httpd                     1/1     Running   0          31s

NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes      ClusterIP   10.43.0.1       <none>        443/TCP    54m
service/redis-service   ClusterIP   10.43.53.213    <none>        6379/TCP   33m
service/httpd           ClusterIP   10.43.211.104   <none>        80/TCP     31s

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/webapp   3/3     3            3           25m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/webapp-79c4cc587b   3         3         3       25m
```

- — -

25. Enter Sleep with timer in container command

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-2
spec:
  containers:
  - name: ubuntu
    image: ubuntu
    command:
      - sleep
      - "5000"
```

```
controlplane ~ ➜ vi ubuntu-sleeper-2.yaml

controlplane ~ ➜ kubectl apply -f ubuntu-sleeper-2.yaml
pod/ubuntu-sleeper-2 created

controlplane ~ ➜ kubectl get pods
NAME               READY   STATUS    RESTARTS   AGE
ubuntu-sleeper     1/1     Running   0          5m29s
ubuntu-sleeper-2   1/1     Running   0          10s
```

- —

— — — — — — — — — -

26. Create ConfigMap using Command

```
create configmap webapp-config-map --from-literal=APP_COLOR=darkblue
get cm
describe cm webapp-config-map
```

```
controlplane ~ ➜ kubectl create configmap webapp-config-map --from-literal=APP_COLOR=darkblue
configmap/webapp-config-map created

controlplane ~ ➜ kubectl get cm
NAME                DATA   AGE
kube-root-ca.crt    1      16m
db-config           3      3m1s
webapp-config-map   1      9s

controlplane ~ ➜ kubectl describe cm webapp-config-map
Name:         webapp-config-map
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
APP_COLOR:
----
darkblue

BinaryData
====
```

```
spec:
  containers:
  - env:
    - name: APP_COLOR
      valueFrom:
        configMapKeyRef:
          name: webapp-config-map
           key: APP_COLOR
```

---

27. Create Secret having multiple key using command

```
kubectl create secret generic db-secret --from-literal=DB_Host=sql01 --from-literal=DB_User=root --from-literal=DB_Password=password123
kubectl describe secret db-secret
```

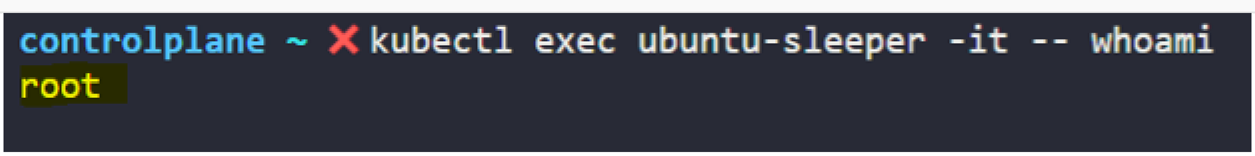
```
controlplane ~ ➜ kubectl describe secret db-secret
Name:         db-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
DB_Host:      5 bytes
DB_Password:  11 bytes
DB_User:      4 bytes
```

```
spec:
  containers:
    - env:
      - name: DB_Host
        valueFrom:
          secretKeyRef:
            name: db-secret
            key: DB_Host
      - name: DB_User
        valueFrom:
          secretKeyRef:
            name: db-secret
            key: DB_User
      - name: DB_Password
        valueFrom:
          secretKeyRef:
            name: db-secret
            key: DB_Password
    - image: kodekloud/simple-webapp-mysql
      imagePullPolicy: Always
      name: webapp
kubectl apply -f /tmp/kubectl-edit-1663824269.yaml
```

---

28. Find out user used to execute the sleep process within the `ubuntu-sleeper` pod ?

```
kubectl exec ubuntu-sleeper -it -- whoami
```



- — -

29. Change default user to run command in container using securityContext

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2022-12-27T07:48:34Z"
  name: ubuntu-sleeper
  namespace: default
  resourceVersion: "718"
  uid: 9f52cb0e-475b-4174-a657-a3a4994822e9
spec:
  securityContext:
    runAsUser: 1010
  containers:
  - command:
    - sleep
    - "4800"
    image: ubuntu
```

- — -

30. save running Pod output to yaml file

```
kubectl get pod ubuntu-sleeper -o yaml > /tmp/test1.yaml
```

31. securityContext capability

```
spec:
  containers:
  - command:
    - sleep
    - "4800"
    image: ubuntu
    imagePullPolicy: Always
    name: ubuntu
    securityContext:
      capabilities:
        add: ["SYS_TIME"]
    resources: {}
```

32. Multple capabilities in Array

```
  containers:
  - command:
    - sleep
    - "4800"
    image: ubuntu
    imagePullPolicy: Always
    name: ubuntu
    securityContext:
      capabilities:
        add: ["SYS_TIME", "NET_ADMIN"]
    resources: {}
```

33. Check existing serviceaccounts

```
kubectl get serviceaccounts
kubectl describe sa default
kubectl create serviceaccount dashboard-sa
```

```
controlplane ~ ➜ kubectl get serviceaccounts
NAME      SECRETS   AGE
default   0         4m36s
dev       0         51s
```

```
controlplane ~ ➜ kubectl describe sa default
Name:                default
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   <none>
Tokens:              <none>
Events:              <none>
```

34. Create token using serviceAccount

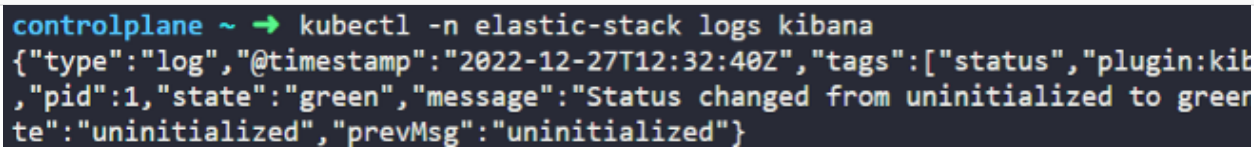
```
kubectl create token dashboard-sa
```

```
controlplane /var/rbac ➜ kubectl create token dashboard-sa
eyJhbGciOiJSUzI1NiIsImtpZCI6ImMwdjRIbkRiaFVEOTlBS244VHR0YnRta3RuNzJqdXkxMkJ6MWR1dFZjVkUifQ.eyJhd
WQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9jYWwiLCJrM3MiXSwiZXhwIjoxNjcyMTM3M
zE2LCJpYXQiOjE2NzIxMzM3MTYsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsI
iwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY2UiOiJkZWZhdWx0Iiwic2VydmljZWFjY291bnQiOnsibmFtZSI6ImRhc2hib
2FyZC1zYSIsInVpZCI6IjBlZDUyMzk4LTRiZGUtNDQyOC1iMmUyLThhYzQxZWYxOWEzYyJ9fSwibmJmIjoxNjcyMTMzNzE2L
CJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDpkYXNoYm9hcmQtc2EifQ.YAk--y8jo-BwwNa9jxP5spx2Y4
_TSQlUvO-mhOh24ck678ZIgZHTUGwnuZCn9S4bFCEmOhoJIt49OYH4YxJC6Gnqj--DFXnejmFsPZKqum9WRhpHXYnyWy3nFA
G2piHO1dWB3jokHsjbRNPL84gzX81LbDVIIHyZmkN7pz4ZZ_7f7NKLWyG4kRkFOArFUldp4KM_hvdBOGu1tvQRD_vNB0-DMr
LrwugQ2cgdJVK9OYzCllMxRBclQ-kM1_R82ZpYHwGTrGyPXb4hizNIUxURXeDG1iZ8fMHp4aLBkB8lecNtbd7XBeukOnH6lY
ny6sxNclh2GXD_rqaldcouOBybSA
```

```
    metadata:
      creationTimestamp: null
      labels:
        name: web-dashboard
    spec:
      serviceAccountName: dashboard-sa
      containers:
      - env:
```

35. Check logs of a pod

```
kubectl -n elastic-stack logs kibana
```



36. Liveness, Readiness, and Startup Probes

Startup Probe

A startup probe verifies whether the application within a container is started. Startup probes run before any other probe, and, unless it finishes successfully, disables other probes. If a container fails its startup probe, then the container is killed and follows the pod's `restartPolicy`.

This type of probe is only executed at startup, unlike readiness probes, which are run periodically.

The startup probe is configured in the spec.containers.startupprobe attribute of the pod configuration.

Readiness Probe

Readiness probes determine whether or not a container is ready to serve requests. If the readiness probe returns a failed state, then

Kubernetes removes the IP address for the container from the endpoints of all Services.

Developers use readiness probes to instruct Kubernetes that a running container should not receive any traffic. This is useful when waiting for an application to perform time-consuming initial tasks, such as establishing network connections, loading files, and warming caches.

The readiness probe is configured in the `spec.containers.readinessprobe` attribute of the pod configuration.

Liveness Probe

Liveness probes determine whether or not an application running in a container is in a `healthy` state. If the liveness probe detects an unhealthy state, then Kubernetes kills the container and tries to redeploy it.

The liveness probe is configured in the `spec.containers.livenessprobe` attribute of the pod configuration.

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2021-08-01T04:55:35Z"
  labels:
    name: simple-webapp
  name: simple-webapp-2
  namespace: default
spec:
  containers:
  - env:
    - name: APP_START_DELAY
```

```
      value: "80"
    image: kodekloud/webapp-delayed-start
    imagePullPolicy: Always
    name: simple-webapp
    ports:
    - containerPort: 8080
      protocol: TCP
    readinessProbe:
      httpGet:
        path: /ready
        port: 8080
~
```

To recreate the pod, run the command:

```
kubectl replace -f simple-webapp-2.yaml --force
```

```
controlplane ~ ➜ kubectl replace -f new.yaml --force
pod "simple-webapp-2" deleted
pod/simple-webapp-2 replaced
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    name: simple-webapp
  name: simple-webapp-1
  namespace: default
spec:
  containers:
  - env:
    - name: APP_START_DELAY
      value: "80"
    image: kodekloud/webapp-delayed-start
    imagePullPolicy: Always
    name: simple-webapp
    ports:
    - containerPort: 8080
      protocol: TCP
    readinessProbe:
      httpGet:
        path: /ready
        port: 8080
    livenessProbe:
      httpGet:
        path: /live
        port: 8080
```

```
      periodSeconds: 1
      initialDelaySeconds: 80
```

37. check memory of nodes in cluster

```
kubectl top node
kubectl top pods
```

```
controlplane kubernetes-metrics-server on  master ➜ kubectl top node
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   317m         0%     1216Mi          0%
node01         36m          0%     292Mi           0%
controlplane kubernetes-metrics-server on  master ➜ kubectl top pods
NAME       CPU(cores)   MEMORY(bytes)
elephant   0m           33Mi
lion       0m           19Mi
rabbit     0m           252Mi
```

38. InitContainer yaml

```
spec:
  containers:
  - command:
    - sh
    - -c
    - echo The app is running! && sleep 3600
    image: busybox:1.28
    imagePullPolicy: IfNotPresent
    name: green-container-1
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-2dz76
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  initContainers:
  - command:
```

```
    - sh
    - -c
    - sleep 5
    image: busybox
    imagePullPolicy: Always
    name: init-myservice
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-2dz76
      readOnly: true
  nodeName: controlplane
```

39. Find Resources with specific Multiple labels

```
kubectl get pods -l env=dev
kubectl get pods -l bu=finance
kubectl get all -l env=prod
kubectl get pods -l env=prod,bu=finance,tier=frontend
```

```
controlplane ~ ➜ kubectl get pods -l env=dev
NAME          READY   STATUS    RESTARTS   AGE
app-1-wtcvd   1/1     Running   0          2m4s
db-1-hl5mp    1/1     Running   0          2m4s
app-1-s8nnq   1/1     Running   0          2m4s
db-1-dhdx7    1/1     Running   0          2m4s
db-1-rzw77    1/1     Running   0          2m4s
app-1-r4xgq   1/1     Running   0          2m4s
db-1-57hrt    1/1     Running   0          2m4s
```

```
controlplane ~ ➜ kubectl get pods -l bu=finance
NAME          READY   STATUS    RESTARTS   AGE
app-1-wtcvd   1/1     Running   0          7m48s
app-1-s8nnq   1/1     Running   0          7m48s
app-1-zzxdf   1/1     Running   0          7m47s
db-2-srp77    1/1     Running   0          7m48s
app-1-r4xgq   1/1     Running   0          7m48s
auth          1/1     Running   0          7m48s
controlplane ~ ➜ kubectl get all -l env=prod
NAME                READY   STATUS    RESTARTS   AGE
pod/app-1-zzxdf     1/1     Running   0          19m
pod/db-2-srp77      1/1     Running   0          19m
pod/app-2-w2zph     1/1     Running   0          19m
pod/auth            1/1     Running   0          19m

NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
service/app-1   ClusterIP   10.43.87.35   <none>        3306/TCP   19m

NAME                    DESIRED   CURRENT   READY   AGE
replicaset.apps/db-2    1         1         1       19m
replicaset.apps/app-2   1         1         1       19m
controlplane ~ ➜ kubectl get pods -l env=prod,bu=finance,tier=frontend
NAME          READY   STATUS    RESTARTS   AGE
app-1-zzxdf   1/1     Running   0          24m
```

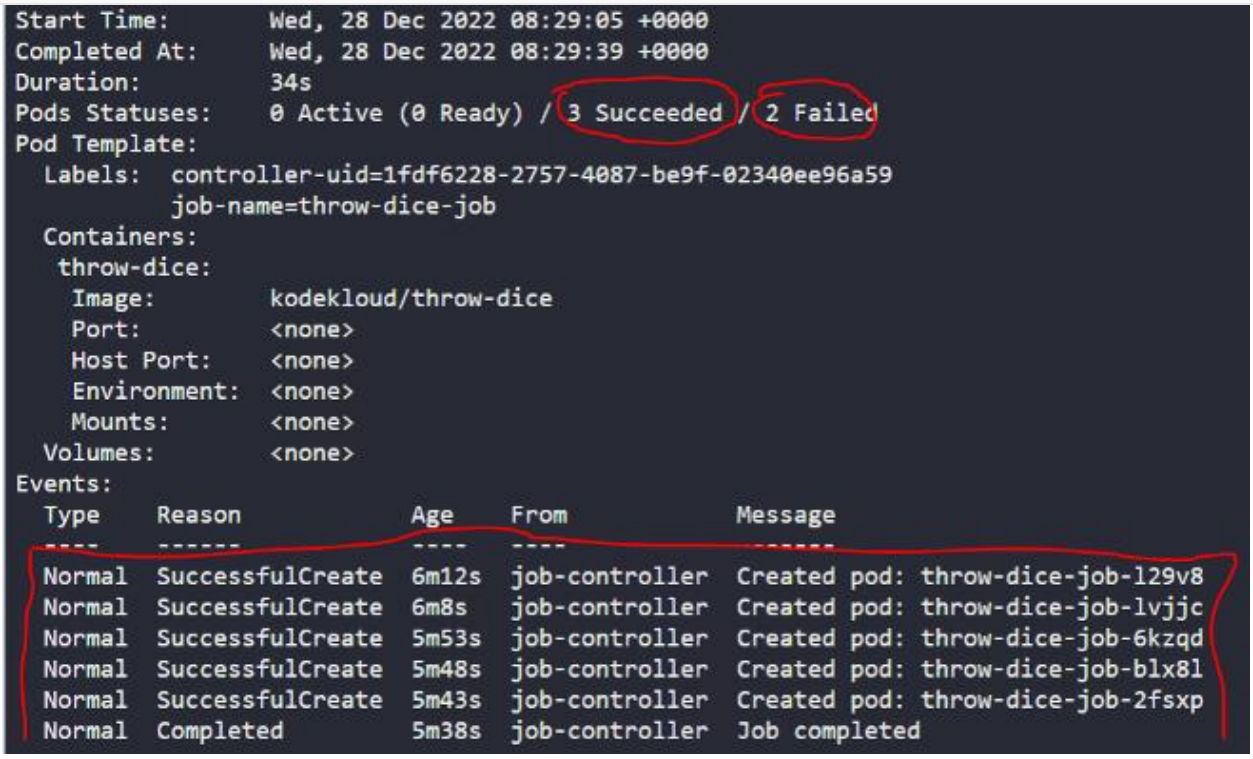
40. Create Job using yaml and Imperative command

```
apiVersion: batch/v1
kind: Job
metadata:
  name: throw-dice-job
spec:
  backoffLimit: 15 # This is so the job does not quit before it succeeds.
  template:
    spec:
      containers:
      - name: throw-dice
        image: kodekloud/throw-dice
      restartPolicy: Never
kubectl create job throw-dice-job --image=kodekloud/throw-dice --dry-
run=client -o yaml > throw-dice-job.yaml
```

```
kubectl create job throw-dice-job --image=kodekloud/throw-dice -o yaml >
throw-dice-job.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: throw-dice-job
spec:
  completions: 3
  backoffLimit: 25 # This is so the job does not quit before it succeeds.
  template:
    spec:
      containers:
      - name: throw-dice
        image: kodekloud/throw-dice
      restartPolicy: Never
```

41. Describe job to check success and faied logs in events

```
kubectl describe job throw-dice-job
```



42. Job finishes in 1 sec using 3 parallel job parallelism: 3

```
apiVersion: batch/v1
kind: Job
metadata:
  name: throw-dice-job
spec:
  completions: 3
  parallelism: 3
  backoffLimit: 25 # This is so the job does not quit before it succeeds.
  template:
    spec:
      containers:
      - name: throw-dice
        image: kodekloud/throw-dice
      restartPolicy: Never
~
kubectl describe job throw-dice-job
```

```
Completion Mode:  NonIndexed
Start Time:       Wed, 28 Dec 2022 08:40:52 +0000
Completed At:     Wed, 28 Dec 2022 08:41:06 +0000
Duration:         14s
Pods Statuses:    0 Active (0 Ready) / 3 Succeeded / 2 Failed
Pod Template:
  Labels:  controller-uid=f81f3ad7-c7c4-4dd5-928f-759bf963f2bb
           job-name=throw-dice-job
  Containers:
   throw-dice:
    Image:        kodekloud/throw-dice
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From            Message
  ----    ------            ----  ----            -------
  Normal  SuccessfulCreate  18s   job-controller  Created pod: throw-dice-job-5vz2r
  Normal  SuccessfulCreate  18s   job-controller  Created pod: throw-dice-job-cbh5m
  Normal  SuccessfulCreate  18s   job-controller  Created pod: throw-dice-job-jm8mw
  Normal  SuccessfulCreate  12s   job-controller  Created pod: throw-dice-job-ks2gs
  Normal  SuccessfulCreate  10s   job-controller  Created pod: throw-dice-job-qr4lq
  Normal  Completed         4s    job-controller  Job completed
```

43. Schedule cron job in yaml

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: throw-dice-cron-job
spec:
  schedule: "30 21 * * *"
  jobTemplate:
    spec:
      completions: 3
      parallelism: 3
      backoffLimit: 25 # This is so the job does not quit before it succeeds.
      template:
        spec:
          containers:
          - name: throw-dice
            image: kodekloud/throw-dice
          restartPolicy: Never
```

44. Check the TargetPort configured in specific service svc

```
kubectl describe svc kubernetes | grep -i targetPort
```

```
controlplane ~ ➜ kubectl describe svc kubernetes | grep -i targetPort
TargetPort:        6443/TCP
```

45. Create a NodePort svc to access the web-app from outside of cluster

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: webapp-service
spec:
  type: NodePort
  ports:
    - targetPort: 8080
      port: 8080
```

```
      nodePort: 30080
  selector:
    name: simple-webapp
```

```
NAME                      TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
service/kubernetes        ClusterIP   10.43.0.1      <none>        443/TCP          13m
service/webapp-service    NodePort    10.43.31.89    <none>        8080:30080/TCP   60s
```

46. Network Policy — networkpolicy — netpol

From internal to pod payroll traffic is allowed and any Internal pod access payroll pod using port 8080

```
kubectl get networkpolicy
kubectl describe networkpolicy payroll-policy
```

```
root@controlplane ~ ➜ kubectl get networkpolicy
NAME             POD-SELECTOR   AGE
payroll-policy   name=payroll   2m22s
root@controlplane ~ ➜ kubectl describe networkpolicy payroll-policy
Name:         payroll-policy
Namespace:    default
Created on:   2022-12-28 10:08:42 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     name=payroll
  Allowing ingress traffic:
    To Port: 8080/TCP
    From:
      PodSelector: name=internal
  Not affecting egress traffic
  Policy Types: Ingress
```

From internal to pod payroll traffic is allowed

47. Create Policy to allow pod access from outside

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: internal-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      name: internal
  policyTypes:
  - Egress
  - Ingress
  ingress:
    - {}
  egress:
  - to:
    - podSelector:
        matchLabels:
          name: mysql
    ports:
    - protocol: TCP
      port: 3306

  - to:
    - podSelector:
        matchLabels:
          name: payroll
    ports:
    - protocol: TCP
      port: 8080

  - ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
kubectl describe netpol internal-policy
```

```
root@controlplane ~ ➜ kubectl describe netpol internal-policy
Name:         internal-policy
Namespace:    default
Created on:   2022-12-28 10:33:18 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     name=internal
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From: <any> (traffic not restricted by source)
  Allowing egress traffic:
    To Port: 3306/TCP
    To:
      PodSelector: name=mysql
    ----------
    To Port: 8080/TCP
    To:
      PodSelector: name=payroll
    ----------
    To Port: 53/UDP
    To Port: 53/TCP
    To: <any> (traffic not restricted by destination)
  Policy Types: Egress, Ingress
```

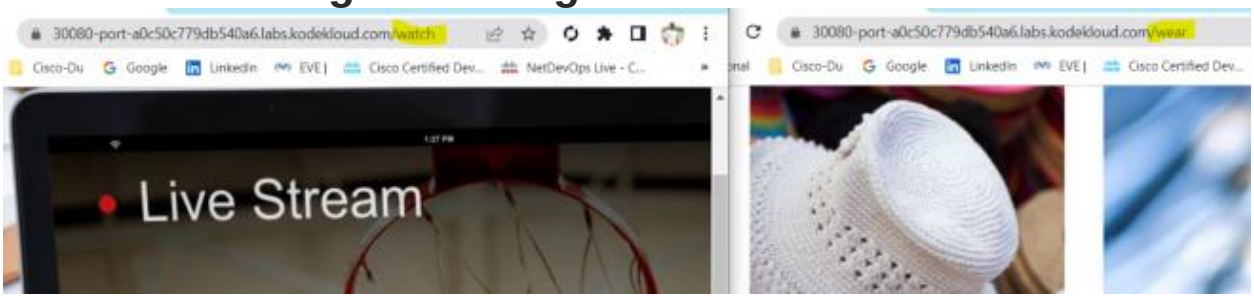
48. Check Ingress Resource in all namespace

```
kubectl get ingress --all-namespaces
kubectl describe ingress ingress-wear-watch -n app-space
```

```
controlplane ~ ✖ kubectl get ingress --all-namespaces
NAMESPACE   NAME                CLASS    HOSTS   ADDRESS         PORTS   AGE
app-space   ingress-wear-watch  <none>   *       10.105.88.61    80      12m
```

```
controlplane ~ ➜ kubectl describe ingress ingress-wear-watch -n app-space
Name:             ingress-wear-watch
Labels:           <none>
Namespace:        app-space
Address:          10.105.88.61
Ingress Class:    <none>
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /wear    wear-service:8080 (10.244.0.5:8080)
              /watch   video-service:8080 (10.244.0.4:8080)
Annotations:  nginx.ingress.kubernetes.io/rewrite-target: /
              nginx.ingress.kubernetes.io/ssl-redirect: false
Events:
  Type    Reason  Age                From                      Message
  ----    ------  ----               ----                      -------
  Normal  Sync    14m (x2 over 14m)  nginx-ingress-controller  Scheduled for sync
```

49. Different Content is shown on /wear and /watch as per Backends configured in Ingress



50. Edit the key values in Ingress using edit

```
kubectl edit ingress ingress-wear-watch -n app-space
```

```
controlplane ~ ➜ kubectl edit ingress ingress-wear-watch -n app-space
ingress.networking.k8s.io/ingress-wear-watch edited
```

51. Create a new Ingress for path /pay and backend service name : pay-service

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-ingress
  namespace: critical-space
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  rules:
  - http:
      paths:
      - path: /pay
        pathType: Prefix
        backend:
          service:
           name: pay-service
           port:
            number: 8282
```

52. Roles and RoleBindings in Service Account

```
kubectl get roles -n ingress-space
kubectl describe roles -n ingress-space

kubectl get rolebindings -n ingress-space
kubectl describe rolebindings -n ingress-space
```

```
root@controlplane ~ ➜ kubectl get roles -n ingress-space
NAME           CREATED AT
ingress-role   2022-12-28T12:41:52Z

root@controlplane ~ ➜ kubectl describe roles -n ingress-space
Name:         ingress-role
Labels:       app.kubernetes.io/name=ingress-nginx
              app.kubernetes.io/part-of=ingress-nginx
Annotations:  <none>
PolicyRule:
  Resources   Non-Resource URLs  Resource Names                  Verbs
  ---------   -----------------  --------------                  -----
  configmaps  []                 []                              [get create]
  configmaps  []                 [ingress-controller-leader-nginx]  [get update]
  endpoints   []                 []                              [get]
  namespaces  []                 []                              [get]
  pods        []                 []                              [get]
  secrets     []                 []                              [get]
```

```
root@controlplane ~ ➜ kubectl get rolebindings -n ingress-space
NAME                   ROLE                AGE
ingress-role-binding   Role/ingress-role   4m6s

root@controlplane ~ ➜ kubectl describe rolebindings -n ingress-space
Name:         ingress-role-binding
Labels:       app.kubernetes.io/name=ingress-nginx
              app.kubernetes.io/part-of=ingress-nginx
Annotations:  <none>
Role:
  Kind:  Role
  Name:  ingress-role
Subjects:
  Kind            Name                     Namespace
  ----            ----                     ---------
  ServiceAccount  ingress-serviceaccount
```

==========

==========

53. Check Logs of Container from outside of Pod (From Host)

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

```
controlplane ~ ✖ kubectl exec webapp -- cat /log/app.log
[2022-12-28 14:27:28,268] INFO in event-simulator: USER3 is viewing page3
[2022-12-28 14:27:29,269] INFO in event-simulator: USER4 logged in
[2022-12-28 14:27:30,270] INFO in event-simulator: USER3 is viewing page2
[2022-12-28 14:27:31,271] INFO in event-simulator: USER4 is viewing page3
[2022-12-28 14:27:32,271] INFO in event-simulator: USER1 is viewing page2
```

But If we delete the Container we won't be able to see logs on Host , but with volume we can do that .

Create a volume "log-volume" with hostpath "/var/log/webapp" and map it with /log in container volumeMounts:

```
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  containers:
  - name: event-simulator
    image: kodekloud/event-simulator
    env:
    - name: LOG_HANDLERS
      value: file
    volumeMounts:
    - mountPath: /log
      name: log-volume

  volumes:
  - name: log-volume
    hostPath:
      # directory location on host
      path: /var/log/webapp
      # this field is optional
      type: Directory
```

54. Create a Persistent Volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-log
spec:
  persistentVolumeReclaimPolicy: Retain
  accessModes:
    - ReadWriteMany
  capacity:
    storage: 100Mi
  hostPath:
    path: /pv/log
kubectl apply -f pv.yaml
kubectl get pv
kubectl describe pv
```

```
controlplane /var/log/webapp ➜ kubectl apply -f pv.yaml
persistentvolume/pv-log created

controlplane /var/log/webapp ➜ kubectl get pv
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
AGE
pv-log   100Mi      RWX            Retain           Available
8s

controlplane /var/log/webapp ➜ kubectl describe pv
Name:            pv-log
Labels:          <none>
Annotations:     <none>
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:
Status:          Available
Claim:
Reclaim Policy:  Retain
Access Modes:    RWX
VolumeMode:      Filesystem
Capacity:        100Mi
Node Affinity:   <none>
Message:
Source:
    Type:          HostPath (bare host directory volume)
    Path:          /pv/log
    HostPathType:
Events:            <none>
```

55. Create a pvc Persistent Volume Claim

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: claim-log-1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
```

```
    requests:
      storage: 50Mi
kubectl apply -f pvc.yaml
kubectl get pvc
kubectl describe pvc claim-log-1
```

```
controlplane /var/log/webapp ➜ kubectl apply -f pvc.yaml
persistentvolumeclaim/claim-log-1 created

controlplane /var/log/webapp ➜ kubectl get pvc
NAME          STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
claim-log-1   Pending                                                     18s

controlplane /var/log/webapp ➜ kubectl describe pvc claim-log-1
Name:          claim-log-1
Namespace:     default
StorageClass:
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Used By:       <none>
Events:
  Type     Reason         Age                From                         Message
  ----     ------         ----               ----                         -------
  Normal   FailedBinding  2s (x4 over 40s)   persistentvolume-controller  no persistent
ailable for this claim and no storage class is set
```

due to access mode mismatch pvc is in pending state

```
controlplane /var/log/webapp ➜ kubectl get pv,pvc
NAME                    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGE
CLASS   REASON   AGE
persistentvolume/pv-log   100Mi      RWX            Retain           Available
                7m33s

NAME                                STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
persistentvolumeclaim/claim-log-1   Pending
3m35s
```

After changing access mode to ReadWriteMany pvc is in bound state

```
controlplane /var/log/webapp ➜ kubectl get pvc
NAME          STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
claim-log-1   Bound    pv-log   100Mi      RWX                           25s
```

Change the pvc from pv in pod yaml and delete/create pod

```
  volumes:
  - name: log-volume
    persistentVolumeClaim:
      claimName: claim-log-1
```

56. STORAGE CLASS

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: delayed-volume-sc
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
kubectl get sc
kubectl describe sc local-path
```

```
controlplane ~ ➜ kubectl get sc
NAME                   PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUM
EEXPANSION   AGE
local-path (default)   rancher.io/local-path   Delete          WaitForFirstConsumer   false
             3m28s

controlplane ~ ➜ kubectl describe sc local-path
Name:                  local-path
IsDefaultClass:        Yes
Annotations:           objectset.rio.cattle.io/applied=H4sIAAAAAAAAA/4yRT+vUMBCGv4rMua1bu1tKwIOu7
EUEQdDzNJlux6aZkkwry7LfXbIqrIffn2PyZN7hfXIFXPg7xcQSwEBSiXimaupSxfJ2q6GAiYMDA9/+oKPHlKCAmRQdKoK5A
oYgisoSUj5K/5OsJtIqslQWVT3lNM4xUDzJ5VegWJ63CQxMTXogW128+czBvf/gnIQXIwLOBAa8WPT130qvGkoL2jw5rT2V6
ZKUZij+SbG5eZVRDKR0F8SpdDTg6rW8YzCgcSW4FeCxJ/+sjxHTCAbqrhmag20Pw9DbZtfu210z7JuhPnQ719m2w3cOe7fPc
f81W1DHfLlE2Th/IEUwEDHYkWJe8PCsgJgL8PxVPNsLGPhEnjRr2cSvM33k4Dicv4jLC34g60niiWPSo4S0zhTh9jsAAP//y
tgh5S0CAAA,objectset.rio.cattle.io/id=,objectset.rio.cattle.io/owner-gvk=k3s.cattle.io/v1, Kind=
Addon,objectset.rio.cattle.io/owner-name=local-storage,objectset.rio.cattle.io/owner-namespace=k
ube-system,storageclass.kubernetes.io/is-default-class=true
Provisioner:           rancher.io/local-path
Parameters:            <none>
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

57. create a Pod using persistentVolumeClaim

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:alpine
    volumeMounts:
      - name: local-persistent-storage
        mountPath: /var/www/html
  volumes:
    - name: local-persistent-storage
      persistentVolumeClaim:
        claimName: local-pvc
```

58. switch to other context using kubectl

```
kubectl config --kubeconfig=/root/my-kube-config use-context research
```

```
controlplane ~ ✖ kubectl config --kubeconfig=/root/my-kube-config use-context research
Switched to context "research".
```

59. make the new my-kube-config as default config file

```
cp my-kube-config .kube/config
```

==========

==========

60. Identify the authorization modes configured on the cluster.

```
kubectl get pods -n kube-system
kubectl describe pod kube-apiserver-controlplane -n kube-system | grep -i
authorization
```

```
controlplane ~ ➜ kubectl get pods -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-6d4b75cb6d-nnnls               1/1     Running   0          9m47s
coredns-6d4b75cb6d-v27mc               1/1     Running   0          9m47s
etcd-controlplane                      1/1     Running   0          9m58s
kube-apiserver-controlplane            1/1     Running   0          10m
kube-controller-manager-controlplane   1/1     Running   0          9m58s
kube-flannel-ds-2mmqz                  1/1     Running   0          9m47s
kube-proxy-dnwzk                       1/1     Running   0          9m47s
kube-scheduler-controlplane            1/1     Running   0          10m
controlplane ~ ➜ kubectl describe pod kube-apiserver-controlplane -n kube-system | grep -i authorization
      --authorization-mode=Node,RBAC
```

61. Roles

Check the access to resources under a specific Role

```
controlplane ~ ➜ kubectl get roles --all-namespaces
NAMESPACE     NAME                                             CREATED AT
blue          developer                                        2022-12-29T06:57:44Z
kube-public   kubeadm:bootstrap-signer-clusterinfo             2022-12-29T06:49:19Z
kube-public   system:controller:bootstrap-signer               2022-12-29T06:49:17Z
kube-system   extension-apiserver-authentication-reader        2022-12-29T06:49:17Z
kube-system   kube-proxy                                       2022-12-29T06:49:19Z
kube-system   kubeadm:kubelet-config                           2022-12-29T06:49:17Z
kube-system   kubeadm:nodes-kubeadm-config                     2022-12-29T06:49:17Z
kube-system   system::leader-locking-kube-controller-manager    2022-12-29T06:49:17Z
kube-system   system::leader-locking-kube-scheduler            2022-12-29T06:49:17Z
kube-system   system:controller:bootstrap-signer               2022-12-29T06:49:17Z
kube-system   system:controller:cloud-provider                 2022-12-29T06:49:17Z
kube-system   system:controller:token-cleaner                  2022-12-29T06:49:17Z
```

```
controlplane ~ ➜ kubectl describe role kube-proxy -n kube-system
Name:         kube-proxy
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources   Non-Resource URLs  Resource Names  Verbs
  ---------   -----------------  --------------  -----
  configmaps  []                 [kube-proxy]    [get]
```

kube-proxy Role can only get/view details of configmaps

62. Which account is mapped to the Role kube-proxy

```
kubectl get rolebindings -n kube-system
kubectl describe rolebinding kube-proxy -n kube-system
```

```
controlplane ~ ➜ kubectl get rolebindings -n kube-system
NAME                                                 ROLE                                                 AGE
kube-proxy                                           Role/kube-proxy                                      27m
kubeadm:kubelet-config                               Role/kubeadm:kubelet-config                          27m
kubeadm:nodes-kubeadm-config                         Role/kubeadm:nodes-kubeadm-config                    27m
system::extension-apiserver-authentication-reader    Role/extension-apiserver-authentication-reader       27m
system::leader-locking-kube-controller-manager       Role/system::leader-locking-kube-controller-manager  27m
system::leader-locking-kube-scheduler                Role/system::leader-locking-kube-scheduler           27m
system:controller:bootstrap-signer                   Role/system:controller:bootstrap-signer              27m
system:controller:cloud-provider                     Role/system:controller:cloud-provider                27m
system:controller:token-cleaner                      Role/system:controller:token-cleaner                 27m
controlplane ~ ➜ kubectl describe rolebinding kube-proxy -n kube-system
Name:         kube-proxy
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  kube-proxy
Subjects:
  Kind   Name                                             Namespace
  ----   ----                                             ---------
  Group  system:bootstrappers:kubeadm:default-node-token
```

63. execute command using other user - -as

```
kubectl get pods --as dev-user
```

```
controlplane ~ ➜ kubectl get pods --as dev-user
Error from server (Forbidden): pods is forbidden: User "dev-user" cannot list resource "pods" in API group "" in
the namespace "default"
```

64. Create a Role and roleBinding to access Pods from dev-user

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "create","delete"]

---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: dev-user-binding
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: developer
  apiGroup: rbac.authorization.k8s.io
```

```
controlplane ~ ➜ kubectl apply -f role_roleBinding.yaml
role.rbac.authorization.k8s.io/developer created
rolebinding.rbac.authorization.k8s.io/dev-user-binding created
```

65. Update the Role to create Pod using deployment .

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: blue
  name: developer
rules:
- apiGroups: ["apps"]
  resources: ["pods"]
  resourceNames: ["darf-blue-app"]
  verbs: ["list", "create","delete"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["list", "create", "delete", "watch"]
```

66. Count the number of clusterroles in cluster using wc

```
kubectl get clusterroles --all-namespaces | wc -l
```

```
controlplane ~ ➜ kubectl get clusterroles --all-namespaces | wc -l
70
```

ClusterRoles are cluster wide and not part of any namespace

==========

==========

67. Add storage access to existing role michelle

```
kubectl api-resources | grep -i storageclass
kubectl api-resources | grep -i persistentvol
```

```
controlplane ~ ➜ kubectl api-resources | grep -i storageclass
storageclasses                  sc          storage.k8s.io/v1                 false        StorageClass

controlplane ~ ➜ kubectl api-resources | grep -i persistentvol
persistentvolumeclaims          pvc         v1                                true         PersistentVolumeClaim
persistentvolumes               pv          v1                                false        PersistentVolume
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: michelle-role
rules:
- apiGroups: ["v1"]
  resources: ["nodes"]
  verbs: ["get", "list", "watch", "delete", "create"]
- apiGroups: ["storage.k8s.io/v1"]
  resources: ["storageclasses"]
  verbs: ["get", "list", "watch", "delete", "create"]
- apiGroups: ["v1"]
  resources: ["persistentvolumes"]
  verbs: ["get", "list", "watch", "delete", "create"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: michelle-binding
subjects:
- kind: User
  name: michelle
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: michelle-role
  apiGroup: rbac.authorization.k8s.io
```

==========

68. Add NamespaceAutoProvision in — enable-admission-plugins to allow creation of namespace while creating Pod in new namespace

vi /etc/kubernetes/manifests/kube-apiserver.yaml

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-add
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.32.39.9
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
```

After changes

```
  containers:
  - command:
    - kube-apiserver
    - --advertise-address=10.32.39.9
    - --allow-privileged=true
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction,NamespaceAutoProvision
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
```

`NamespaceExists` and `NamespaceAutoProvision` admission controllers are deprecated and now replaced by `NamespaceLifecycle` admission controller.

The `NamespaceLifecycle` admission controller will make sure that requests to a non-existent namespace is rejected and that the default namespaces such as `default`, `kube-system` and `kube-public` cannot be deleted.

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

69. Create a TLS secret for webhook

```
kubectl -n webhook-demo create secret tls webhook-server-tls \
    --cert "/root/keys/webhook-server-tls.crt" \
    --key "/root/keys/webhook-server-tls.key"
```

70. Find short names for multiple resources in api-resources

```
kubectl api-resources | grep -E -i
'deployments|replicasets|cronjobs|customresourcedefinitions'
```

71. Find major minor in kubernetes version

```
kubectl version -o yaml | grep -i -E 'major|minor'
```

```
controlplane ~ ➜ kubectl version -o yaml | grep -i -E 'major|minor'
  major: "1"
  minor: "24"
  major: "1"
  minor: "24"
```

72. Enable the `v1alpha1` version for `rbac.authorization.k8s.io` API group

Take a backup of that `apiserver` manifest file

```
cp -v /etc/kubernetes/manifests/kube-apiserver.yaml /root/kube-apiserver.yaml.backup
```

Edit the apiserver manifest file :

```
vi /etc/kubernetes/manifests/kube-apiserver.yaml
```

Add below line :

```
    - --runtime-config=rbac.authorization.k8s.io/v1alpha1
```

kubelet will recreate apiserver pod in sometime

```
controlplane ~ ✖ kubectl get po -n kube-system
The connection to the server controlplane:6443 was refused - did you specify th
ight host or port?

controlplane ~ ✖ kubectl get po -n kube-system
NAME                                   READY   STATUS    RESTARTS      AGE
coredns-6d4b75cb6d-kpb7z               1/1     Running   0             39m
coredns-6d4b75cb6d-qqfv8               1/1     Running   0             39m
etcd-controlplane                      1/1     Running   0             39m
kube-apiserver-controlplane            0/1     Pending   0             1s
kube-controller-manager-controlplane   1/1     Running   1 (28s ago)   39m
kube-flannel-ds-cmv9s                  1/1     Running   0             39m
kube-proxy-9bwv7                       1/1     Running   0             39m
kube-scheduler-controlplane            1/1     Running   1 (28s ago)   39m
```

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

73. change api version using kubectl-convert plugin

```
# Install plugin
curl -LO https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl-convert
chmod +x kubectl-convert
mv kubectl-convert /usr/local/bin/kubectl-convert
kubectl-convert -h

# convert api version
kubectl-convert -f ingress-old.yaml --output-version networking.k8s.io/v1 >
ingress-new.yaml
kubectl create -f ingress-new.yaml
kubectl get ing ingress-space -oyaml | grep apiVersion
```

74. custom Resource (customresourcedefinitions)

It is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation , It is either namespace or cluster scope based .

```
kind: CustomResourceDefinition
metadata:
  name: nternals.datasets.kodecloud.com
spec:
  group: datasets.kodecloud.com
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          type: object
          properties:
            spec:
              type: object
              properties:
                internalLoad:
                  type: string
                range:
                  type: integer
                percentage:
                  type: string
  scope: namespaced
  names:
    plural: internal
    singular: internal
    kind: Internal
    shortNames:
    - int
vi crd.yaml
kubectl apply -f crd.yaml
kubectl get customresourcedefinitions
kubectl describe customresourcedefinitions internals.datasets.kodekloud.com
```

```
controlplane ~ ➜ kubectl apply -f crd.yaml
customresourcedefinition.apiextensions.k8s.io/internals.datasets.kodekloud.com created
```

```
controlplane ~ ➜ kubectl get customresourcedefinitions
NAME                                  CREATED AT
collectors.monitoring.controller      2022-12-30T04:57:23Z
globals.traffic.controller            2022-12-30T04:57:24Z
internals.datasets.kodekloud.com      2022-12-30T05:15:47Z

controlplane ~ ➜ kubectl describe customresourcedefinitions internals.datasets.kodekloud.com
Name:         internals.datasets.kodekloud.com
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  apiextensions.k8s.io/v1
Kind:         CustomResourceDefinition
Metadata:
  Creation Timestamp:  2022-12-30T05:15:47Z
  Generation:          1
  Managed Fields:
```

75. Create a new custom resource using customresourcedefinitions

```
---
kind: Internal
apiVersion: datasets.kodekloud.com/v1
metadata:
  name: internal-space
  namespace: default
spec:
  internalLoad: "high"
  range: 80
  percentage: "50"
~
vi custom.yaml
kubectl apply -f custom.yaml
kubectl api-resources | grep -i Kodekloud
kubectl get internals
kubectl describe internals internal-space
```

```
controlplane ~ ➜ kubectl apply -f custom.yaml
internal.datasets.kodekloud.com/internal-space created
controlplane ~ ➜ kubectl api-resources | grep -i Kodekloud
internals                                 int          datasets.kodekloud.com/v1          true
   Internal
```

```
controlplane ~ ➜ kubectl get internals
NAME             AGE
internal-space   3m17s

controlplane ~ ➜ kubectl describe internals internal-space
Name:         internal-space
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  datasets.kodekloud.com/v1
Kind:         Internal
Metadata:
  Creation Timestamp:  2022-12-30T05:21:23Z
  Generation:          1
```

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

76. Deployment Strategy

Canary

Recreate

Blue-Green

Rolling Update

```
kubectl describe deployments.apps frontend | grep -i StrategyType:
```

```
controlplane ~ ➜ kubectl describe deployments.apps frontend | grep -i StrategyType:
StrategyType:           RollingUpdate
```

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

= = = =

77. decode password from secret using base64 -d

```
 echo Password: $(kubectl get secret --namespace default bravo-drupal -o
jsonpath="{.data.drupal-password}" | base64 -d)
 kubectl describe secret bravo-drupal
```


```
controlplane ~ ➜  echo Password: $(kubectl get secret --namespace default bravo-drupal -o jsonpath="{.data.drupal-password}" | base64 -d)
Password: UKL1Hqz5PN

controlplane ~ ➜ kubectl describe secret bravo-drupal
Name:         bravo-drupal
Namespace:    default
Labels:       app.kubernetes.io/instance=bravo
              app.kubernetes.io/managed-by=Helm
              app.kubernetes.io/name=drupal
              helm.sh/chart=drupal-12.5.12
Annotations:  meta.helm.sh/release-name: bravo
              meta.helm.sh/release-namespace: default

Type:  Opaque

Data
====
drupal-password:  10 bytes
```

— —

— — — — — — — — — — —

78. upgrading from Client side to server side and vice versa

```
kubectl apply --server-side [--dry-run=server]
```

By default, field management of the object transfers from client-side apply to kubectl server-side apply without encountering conflicts.

Downgrading

```
kubectl apply --server-side --field-manager=my-manager [--dry-run=server]
```

Thanks .

Ranjeet Jangra

Network Automation Expert with 13 years of experience in Development | Testing | Deployment | Support | Automation on various Technologies like IP-Routing, Cloud, Programming, Containers, Kubernetes, Telemetry, Orchestration, Network-Programmability, YANG, TextFSM, Jinja, RestAPI
https://www.linkedin.com/in/ranjeetjangra/