# CKA Curriculum & Scenarios

## Curriculum

### https://github.com/cncf/curriculum

> Note: Curriculum and exam pattern will be changing from Sept'2020

### Exam Details

> https://training.linuxfoundation.org/wp-content/uploads/2020/04/Important-Tips-CKA-CKAD-April2020.pdf

## A few scenarios to practise:

> Note: Please note that the questions mentioned below are not the same as in the actual exam
> Note: Please note that the difficulty level may not be the same as in the actual exam
> Note: Solutions are just for reference and there are many better ways to get the same thing done.

1. List all the namespaces in the cluster

```
root@k8s-master:~# kubectl get ns
NAME              STATUS   AGE
default           Active   144m
kube-node-lease   Active   144m
kube-public       Active   144m
kube-system       Active   144m
```

2. List all the pods in all namespaces

```
root@k8s-master:~# kubectl get pods --all-namespaces
NAMESPACE     NAME                                 READY   STATUS    RESTARTS
kube-system   coredns-66bff467f8-6trx8             1/1     Running   0
kube-system   coredns-66bff467f8-lhnvf             1/1     Running   0
kube-system   etcd-k8s-master                      1/1     Running   0
kube-system   kube-apiserver-k8s-master            1/1     Running   0
kube-system   kube-controller-manager-k8s-master   1/1     Running   0
kube-system   kube-flannel-ds-amd64-l74jg          1/1     Running   0
kube-system   kube-flannel-ds-amd64-lh6sq          1/1     Running   0
kube-system   kube-proxy-6kv4w                     1/1     Running   0
kube-system   kube-proxy-h7fzm                     1/1     Running   0
kube-system   kube-scheduler-k8s-master            1/1     Running   0
```

3. List all the pods in the particular namespace

```
root@k8s-master:~# kubectl get pods -n kube-system -o wide
NAME                                 READY   STATUS    RESTARTS   AGE    IP
coredns-66bff467f8-6trx8             1/1     Running   0          149m   10.2
coredns-66bff467f8-lhnvf             1/1     Running   0          149m   10.2
etcd-k8s-master                      1/1     Running   0          150m   192.
kube-apiserver-k8s-master            1/1     Running   0          150m   192.
```

```
kube-controller-manager-k8s-master   1/1     Running   0          150m   192.
kube-flannel-ds-amd64-l74jg          1/1     Running   0          147m   192.
kube-flannel-ds-amd64-lh6sq          1/1     Running   0          147m   192.
kube-proxy-6kv4w                     1/1     Running   0          148m   192.
kube-proxy-h7fzm                     1/1     Running   0          149m   192.
kube-scheduler-k8s-master            1/1     Running   0          150m   192.
```

4. List all the services in the particular namespace

```
root@k8s-master:~# kubectl get svc -n kube-system -o wide
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AG
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   15
```

5. List all the pods showing name and namespace with a json path expression

```
root@k8s-master:~# kubectl get pods -o=jsonpath='{range .items[*]}{.metadata.
coredns-66bff467f8-6trx8 kube-system
coredns-66bff467f8-lhnvf kube-system
etcd-k8s-master kube-system
kube-apiserver-k8s-master kube-system
kube-controller-manager-k8s-master kube-system
kube-flannel-ds-amd64-l74jg kube-system
kube-flannel-ds-amd64-lh6sq kube-system
kube-proxy-6kv4w kube-system
kube-proxy-h7fzm kube-system
kube-scheduler-k8s-master kube-system
```

6. Create an nginx deployment in a default namespace and verify the pod running

```
root@k8s-master:~# kubectl create deployment nginx --image=nginx
deployment.apps/nginx created


root@k8s-master:~# kubectl get deployments -n default
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           9s


root@k8s-master:~# kubectl get pods -n default
NAME                     READY   STATUS    RESTARTS   AGE
nginx-f89759699-9s6hp    1/1     Running   0          21s


root@k8s-master:~# kubectl create service nodeport nginx --tcp=80:80
service/nginx created


root@k8s-master:~# kubectl get svc -n default
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        24h
nginx        NodePort    10.104.191.182   <none>        80:32090/TCP   12s
```

7. Generate the yaml for pod called nginx2 & write to /opt/nginx203.yml. DONOT create the pod

```
root@k8s-master:~# cat /opt/nginx203.yml
apiVersion: extensions/v1beta1
```

```
kind: Pod
metadata:
  name: nginx2
  namespace: default
  labels:
    role: nginx2
spec:
  containers:
  - name: nginx2
    image: nginx
    imagePullPolicy: IfNotPresent
    ports:
      - name: http-port
        containerPort: 80
```

8. Output the yaml file of the pod nginx created above & write the output to /opt/nginx.yml

```
root@k8s-master:~# kubectl get pods -n default
NAME                    READY   STATUS    RESTARTS   AGE

nginx-f89759699-9s6hp   1/1     Running   0          16m
root@k8s-master:~# kubectl get pods nginx-f89759699-9s6hp -o yaml >  /opt/ngi
```

9. Get the complete details of the pod nginx you just created

```
kubectl describe pod nginx-f89759699-9s6hp
```

10. Delete the pod nginx you just created

```
root@k8s-master:~# kubectl delete pod nginx-f89759699-9s6hp -n default
pod "nginx-f89759699-9s6hp" deleted
```

11. Create a pod named alpine with image nginx & Delete the pod created without any delay (force delete)

```
root@k8s-master:~# kubectl run alpine --image=nginx --port=80
pod/alpine created

root@k8s-master:~# kubectl get pods
NAME                    READY   STATUS              RESTARTS   AGE
alpine                  0/1     ContainerCreating   0          6s

root@k8s-master:~# kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
alpine                  1/1     Running   0          24s
```

12. Create the nginx pod with version 1.17.4 and expose it on port 80

```
root@k8s-master:~# kubectl run nginx1 --image=nginx:1.17.4 --port=80
pod/nginx1 created

root@k8s-master:~# kubectl describe pod nginx1 | grep nginx:1.17.4
    Image:          nginx:1.17.4
  Normal  Pulling    85s        kubelet, k8s-worker  Pulling image "nginx:1.1
  Normal  Pulled     50s        kubelet, k8s-worker  Successfully pulled imag
```

13. Change the Image version back to 1.19 for the pod you just updated and observe the changes

```
root@k8s-master:~# kubectl edit pod nginx1
pod/nginx1 edited

root@k8s-master:~# kubectl describe pod nginx1 | grep nginx:1.19
    Image:          nginx:1.19
  Normal  Pulling    17s                kubelet, k8s-worker  Pulling image "r
  Normal  Pulled     13s                kubelet, k8s-worker  Successfully pul

root@k8s-master:~# kubectl get pods | grep nginx1
nginx1                  1/1     Running   1          20m
```

14. Check the Image version without the describe command (use jsonpath)

```
root@k8s-master:~# kubectl get pods nginx1 -o jsonpath="{..image}"
nginx:1.19 nginx:1.19
```

15. Execute the simple shell on the pod

```
root@k8s-master:~# kubectl exec nginx1 -- ls
bin
boot
dev
docker-entrypoint.d
docker-entrypoint.sh
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

16. Get the IP Address of the pod you just created

```
root@k8s-master:~# kubectl get pods nginx1 -o jsonpath="{..ip}"
10.244.1.9
```

17. Checking logs

```
root@k8s-master:~# kubectl logs -f nginx1
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to pe
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-de
10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/conf.d/de
10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/nginx/conf.d/
```

```
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-template
/docker-entrypoint.sh: Configuration complete; ready for start up
```

18. Create a busybox pod with command sleep 60 (crashing for a purpose)

```
root@k8s-master:~# kubectl run busybox --image=busybox --command 'sleep 60'
pod/busybox created
```

19. If pod crashed check the logs of the pod for the reason

```
root@k8s-master:~# kubectl describe pod busybox | grep -i "failed"
      Message:      OCI runtime create failed: container_linux.go:349: starti
  Warning  Failed     2m21s (x4 over 3m22s)  kubelet, k8s-worker  Error: fail
  Warning  BackOff    105s (x6 over 3m14s)   kubelet, k8s-worker  Back-off re
```

20. Create a busybox pod and run command ls while creating

```
root@k8s-master:~# kubectl run -i -t busybox --image=busybox --restart=Never
bin   dev   etc   home  proc  root  sys   tmp   usr   var
```

21. Check the connection of the nginx pod from the busybox pod

```
root@k8s-master:~# kubectl run -i -t busybox --image=busybox --restart=Never
10.104.191.182 (10.104.191.182:80) open
```

22. Create a busybox pod and echo message ‘How are you’ and delete it manually

```
root@k8s-master:~# kubectl run -i -t busybox --image=busybox --restart=Never
How are you

root@k8s-master:~# kubectl delete pod busybox
pod "busybox" deleted
```

23. Delete busybox pod immediately

```
root@k8s-master:~# kubectl delete pod busybox --grace-period=0
pod "busybox" deleted
```

24. List the nginx pod with custom columns POD_NAME and POD_STATUS

```
root@k8s-master:~# kubectl get pod nginx1 -o custom-columns=POD_NAME:.metadat
POD_NAME   POD_STATUS
nginx1     Running
```

25. List all the pods sorted by name

```
root@k8s-master:~# kubectl get pods --sort-by='{.metadata.name}'
NAME                    READY   STATUS    RESTARTS   AGE
alpine                  1/1     Running   0          100m
nginx1                  1/1     Running   1          98m
nginx-f89759699-qmf8z   1/1     Running   0          102m
```

26. List all the pods sorted by created timestamp

```
root@k8s-master:~# kubectl get pods --sort-by='{.metadata.creationTimestamp}'
NAME                    READY   STATUS    RESTARTS   AGE
```

```
nginx-f89759699-qmf8z   1/1     Running   0          102m
alpine                  1/1     Running   0          100m
nginx1                  1/1     Running   1          98m
```

27. Get the pods with label information

```
root@master:~# kubectl get pods --show-labels --all-namespaces
NAMESPACE     NAME                              READY   STATUS    RESTARTS   A
kube-system   coredns-66bff467f8-4v4jl          1/1     Running   1          4
kube-system   coredns-66bff467f8-b66vs          1/1     Running   1          4
kube-system   etcd-master                       1/1     Running   1          4
kube-system   kube-apiserver-master             1/1     Running   1          4
kube-system   kube-controller-manager-master    1/1     Running   1          4
kube-system   kube-flannel-ds-amd64-tv598       1/1     Running   1          4
kube-system   kube-flannel-ds-amd64-z5dwk       1/1     Running   1          4
kube-system   kube-proxy-c4mbc                  1/1     Running   1          4
kube-system   kube-proxy-pns6n                  1/1     Running   1          4
kube-system   kube-scheduler-master             1/1     Running   1          4
```

28. Create 5 nginx pods in which two of them is labeled env=prod and three of them is labeled env=dev

```
root@master:~# for i in `seq 1 2`;do kubectl run nginx$i --image=nginx --port
pod/nginx1 created
pod/nginx2 created

root@master:~# for i in `seq 3 5`;do kubectl run nginx$i --image=nginx --port
pod/nginx3 created
pod/nginx4 created
pod/nginx5 created

root@master:~# kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx1   1/1     Running   0          38s
nginx2   1/1     Running   0          38s
nginx3   1/1     Running   0          15s
nginx4   1/1     Running   0          15s
nginx5   1/1     Running   0          15s
```

29. Get the pods with label env=dev and also output the labels

```
root@master:~# kubectl get pods --show-labels -n default
NAME     READY   STATUS    RESTARTS   AGE     LABELS
nginx1   1/1     Running   0          2m17s   env=dev
nginx2   1/1     Running   0          2m17s   env=dev
nginx3   1/1     Running   0          114s    env=prod
nginx4   1/1     Running   0          114s    env=prod
nginx5   1/1     Running   0          114s    env=prod
```

30. Get the pods with label env=prod and also output the labels

```
root@master:~# kubectl get pods -l 'env=prod' --show-labels -n default
NAME     READY   STATUS    RESTARTS   AGE     LABELS
nginx3   1/1     Running   0          3m42s   env=prod
nginx4   1/1     Running   0          3m42s   env=prod
nginx5   1/1     Running   0          3m42s   env=prod
```

31. Get the pods with labels env=dev and env=prod and output the labels as well

```
root@master:~# kubectl get pods -l 'env in (prod, dev)' --show-labels -n defa
NAME     READY   STATUS    RESTARTS   AGE     LABELS
nginx1   1/1     Running   0          6m2s    env=dev
nginx2   1/1     Running   0          6m2s    env=dev
nginx3   1/1     Running   0          5m39s   env=prod
nginx4   1/1     Running   0          5m39s   env=prod
nginx5   1/1     Running   0          5m39s   env=prod
```

32. Remove the labels for the pods that we created now and verify all the labels are removed

```
root@master:~# for i in `seq 1 5`; do kubectl label pods nginx$i env-; done
pod/nginx1 labeled
pod/nginx2 labeled
pod/nginx3 labeled
pod/nginx4 labeled
pod/nginx5 labeled

root@master:~# kubectl get pods --show-labels -n default
NAME     READY   STATUS    RESTARTS   AGE   LABELS
nginx1   1/1     Running   0          11m   <none>
nginx2   1/1     Running   0          11m   <none>
nginx3   1/1     Running   0          10m   <none>
nginx4   1/1     Running   0          10m   <none>
nginx5   1/1     Running   0          10m   <none>
```

33. Let's add the label app=nginx for all the pods and verify

```
root@master:~# for i in `seq 1 5`; do kubectl label pods nginx$i app=nginx; d
pod/nginx1 labeled
pod/nginx2 labeled
pod/nginx3 labeled
pod/nginx4 labeled
pod/nginx5 labeled

root@master:~# kubectl get pods --show-labels -n default
NAME     READY   STATUS    RESTARTS   AGE   LABELS
nginx1   1/1     Running   0          11m   app=nginx
nginx2   1/1     Running   0          11m   app=nginx
nginx3   1/1     Running   0          11m   app=nginx
nginx4   1/1     Running   0          11m   app=nginx
nginx5   1/1     Running   0          11m   app=nginx
```

34. Create a Pod that will be deployed on this node with the label nodeName=nginxnode

```
root@master:~# kubectl label node worker-virtualbox nodeName=nginxnode
node/worker-virtualbox labeled

root@master:~# cat nginx.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-worker
  labels:
    env: test
```

```
spec:
  containers:
  - name: nginx-worker
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    nodeName: nginxnode

root@master:~# kubectl apply -f nginx.yml
pod/nginx-worker created

root@master:~# kubectl get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP           NODE
nginx-worker   1/1     Running   0          14s   10.244.1.9   worker-virtual
```

35. Remove all the pods that we created so far

```
root@master:~# kubectl delete --all pods -n default
pod "nginx-worker" deleted
pod "nginx1" deleted
pod "nginx2" deleted
pod "nginx3" deleted
pod "nginx4" deleted
pod "nginx5" deleted
```

36. Create a deployment called webapp with image nginx with 5 replicas

```
root@master:~# cat nginx-dep.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
  labels:
    app: test
spec:
  replicas: 5
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
       app: test
    spec:
      containers:
      - name: nginx
        image: nginx
        imagePullPolicy: IfNotPresent

root@master:~# kubectl apply -f nginx-dep.yml
deployment.apps/webapp created

root@master:~# kubectl get deployments --show-labels
NAME     READY   UP-TO-DATE   AVAILABLE   AGE     LABELS
webapp   5/5     5            5           2m37s   app=test
```

```
root@master:~# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
webapp-687dd589f7-6q9vl   1/1     Running   0          20s
webapp-687dd589f7-bqz87   1/1     Running   0          20s
webapp-687dd589f7-chzrd   1/1     Running   0          20s
webapp-687dd589f7-ghpz2   1/1     Running   0          20s
webapp-687dd589f7-z74bh   1/1     Running   0          20s
```

37. Output the yaml file of the deployment you just created

```
root@master:~# kubectl get deployments webapp -o yaml
```

38. Get the pods of this deployment

```
root@master:~# kubectl get pods -l app=test
NAME                      READY   STATUS    RESTARTS   AGE
webapp-687dd589f7-6q9vl   1/1     Running   0          11m
webapp-687dd589f7-bqz87   1/1     Running   0          11m
webapp-687dd589f7-chzrd   1/1     Running   0          11m
webapp-687dd589f7-ghpz2   1/1     Running   0          11m
webapp-687dd589f7-z74bh   1/1     Running   0          11m
```

39. Scale the deployment from 5 replicas to 8 replicas and verify

```
root@master:~# kubectl scale --replicas=8 deployments webapp
deployment.apps/webapp scaled

root@master:~# kubectl get pods -l app=test
NAME                      READY   STATUS    RESTARTS   AGE
webapp-687dd589f7-6q9vl   1/1     Running   0          14m
webapp-687dd589f7-bqz87   1/1     Running   0          14m
webapp-687dd589f7-chzrd   1/1     Running   0          14m
webapp-687dd589f7-ghpz2   1/1     Running   0          14m
webapp-687dd589f7-k6qjf   1/1     Running   0          5s
webapp-687dd589f7-tb7dh   1/1     Running   0          5s
webapp-687dd589f7-tv2v5   1/1     Running   0          5s
webapp-687dd589f7-z74bh   1/1     Running   0          14m
```

40. Get the replicaset that created with this deployment

```
root@master:~# kubectl get rs --selector app=test -o wide
NAME                DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES   S
webapp-687dd589f7   8         8         8       16m   nginx        nginx    a
```

41. Delete the deployment you just created and watch all the pods are also being deleted

```
root@master:~# kubectl delete deployment webapp
deployment.apps "webapp" deleted

root@master:~# kubectl get pods -l app=test
No resources found in default namespace.
```

42. Create a deployment of webapp with image nginx:1.17.1 with container port 80 and verify the image version

```
root@master:~# cat nginx-dep.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp
  labels:
    app: test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
       app: test
    spec:
      containers:
      - name: nginx
        image: nginx:1.17.1
        imagePullPolicy: IfNotPresent
        ports:
        - name: http
          containerPort: 80

root@master:~# kubectl get pods -l app=test
NAME                      READY   STATUS    RESTARTS   AGE
webapp-778f7555bb-5msfm   1/1     Running   0          19s
webapp-778f7555bb-tvsbw   1/1     Running   0          19s
```

43. Update the deployment with the image version 1.17.4 and verify

```
root@master:~# kubectl set image deployment webapp nginx=nginx:1.17.4
deployment.apps/webapp image updated
```

44. Check the rollout history and make sure everything is ok after the update

```
root@master:~# kubectl rollout history deployments webapp
deployment.apps/webapp
REVISION  CHANGE-CAUSE
1         <none>
2         <none>

root@master:~# kubectl get pods -l app=test
NAME                      READY   STATUS    RESTARTS   AGE
webapp-56d7fcdf69-6xpkc   1/1     Running   0          2m39s
webapp-56d7fcdf69-xktf9   1/1     Running   0          2m54s
```

45. Undo the deployment to the previous version 1.17.1 and verify Image has the previous version

```
root@master:~# kubectl rollout undo deployment webapp --to-revision=1
deployment.apps/webapp rolled back

webapproot@master:~# kubectl get deployments webapp -o jsonpath='{..image}'
nginx:1.17.1
```

46. Update the deployment with the image version 1.16.1 and verify the image and also check the rollout history

```
root@master:~# kubectl set image deployment webapp --record nginx=nginx:1.16.
deployment.apps/webapp image updated

root@master:~# kubectl rollout history deployments webapp
deployment.apps/webapp
REVISION  CHANGE-CAUSE
2         <none>
3         <none>
4         kubectl set image deployment webapp nginx=nginx:1.16.1 --record=tru
```

47. Update the deployment with the wrong image version 1.100 and verify something is wrong with the deployment

```
root@master:~# kubectl set image deployment webapp --record nginx=nginx:1.100
deployment.apps/webapp image updated

root@master:~# kubectl logs -f webapp-6f595bbd88-d7bj8
Error from server (BadRequest): container "nginx" in pod "webapp-6f595bbd88-d

root@master:~# kubectl describe pod webapp-6f595bbd88-d7bj8 | grep -i "failed
  Warning  Failed     29s (x4 over 2m1s)  kubelet, worker-virtualbox  Failed
  Warning  Failed     29s (x4 over 2m1s)  kubelet, worker-virtualbox  Error:
  Warning  Failed     4s (x6 over 2m1s)   kubelet, worker-virtualbox  Error:
```

48. Undo the deployment with the previous version and verify everything is Ok

```
root@master:~# kubectl rollout history deployments webapp
deployment.apps/webapp
REVISION  CHANGE-CAUSE
2         <none>
3         <none>
4         kubectl set image deployment webapp nginx=nginx:1.16.1 --record=tru
5         kubectl set image deployment webapp nginx=nginx:1.100 --record=true

root@master:~# kubectl rollout undo deployments webapp --to-revision=4
deployment.apps/webapp rolled back

root@master:~# kubectl get deployments webapp
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
webapp   2/2     2            2           23h
root@master:~#
```

49. Pause & Resume
    a. Pause the rollout of the deployment,
    b. Update the deployment with the image version latest and check the history and verify nothing is going on,
    c. Resume the rollout of the deployment,
    d. Check the rollout history and verify it has the new version

```
root@master:~# kubectl rollout pause deployment webapp
deployment.apps/webapp paused
```

```
root@master:~# kubectl describe deployment webapp | grep -i "pause"
  Progressing    Unknown  DeploymentPaused
```

```
root@master:~# kubectl set image deployment webapp --record nginx=nginx:
deployment.apps/webapp image updated

root@master:~# kubectl rollout history deployments webapp
deployment.apps/webapp
REVISION  CHANGE-CAUSE
2         <none>
5         kubectl set image deployment webapp nginx=nginx:1.100 --record
6         kubectl set image deployment webapp nginx=nginx:1.16.1 --recor
7         kubectl set image deployment webapp nginx=nginx:1.17.1 --recor

root@master:~# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
webapp-75cd5c5cb9-fqrkn   1/1     Running   0          20m
webapp-75cd5c5cb9-qzvj9   1/1     Running   0          20m

root@master:~# kubectl describe deployment webapp | grep -i "image"
                         kubernetes.io/change-cause: kubectl set image de
    Image:        nginx:1.17.1

root@master:~# kubectl get pods webapp-75cd5c5cb9-fqrkn -o=jsonpath='{..
nginx:1.16.1 nginx:1.16.1
```

```
root@master:~# kubectl rollout resume deployments webapp
deployment.apps/webapp resumed

root@master:~# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
webapp-778f7555bb-bhbr8   1/1     Running   0          22s
webapp-778f7555bb-bk8zt   1/1     Running   0          23s

root@master:~# kubectl get pods webapp-778f7555bb-bhbr8 -o=jsonpath='{..
nginx:1.17.1 nginx:1.17.1root@master:~#
```

50. Create a hostPath PersistentVolume named task-pv-volume with storage 1Gi, access modes ReadWriteOnce, storageClassName manual, and volume at /mnt/data and verify

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: manual
  hostPath:
```

```
    path: /mnt/data


root@master:~# kubectl get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
task-pv-volume   1Gi        RWO            Retain           Bound    default/
```

51. Create a PersistentVolumeClaim of at least 3Gi storage and access mode ReadWriteOnce and verify status is Bound

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
  resources:
    requests:
       storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: manual

root@master:~# kubectl get pvc
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECL
task-pv-claim   Bound    task-pv-volume   1Gi        RWO            manual
```

52. Create an nginx pod with containerPort 80 and with a PersistentVolumeClaim task-pv-claim and has a mouth path "/usr/share/nginx/html"

```
#Imperative/Recommended approach

root@master:~# kubectl run nginx --image nginx --dry-run=client -o yaml > ngi

root@master:~# cat nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}

root@master:~# vi nginx.yaml #add all necessary fieds here

root@master:~# cat nginx.yaml
apiVersion: v1
kind: Pod
```

```
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    ports:
    - containerPort: 80
      name: "http-port"
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: task-pv-storage
  volumes:
  - name: task-pv-storage
    persistentVolumeClaim:
       claimName: task-pv-claim

root@master:~# kubectl apply -f nginx.yaml
pod/nginx created
```

53. Create an nginx pod and load values from the above configmap keyvalcfgmap volume /opt/keyvalcfgmap-vars.txt

```
#Imperative/Recommended approach

root@k8s-master:~# cat keyvalcfgmap-vars.txt
key1=value1
keys2=value2

root@k8s-master:~# kubectl create configmap keyvalcfgmap --from-file keyvalcf

root@k8s-master:~# vi configmap.yaml

root@k8s-master:~# cat configmap.yaml
apiVersion: v1
data:
  keyvalcfgmap-vars.txt: |
    key1=value1
    keys2=value2
kind: ConfigMap
metadata:
  creationTimestamp: null
  name: keyvalcfgmap

root@k8s-master:~# kubectl apply -f configmap.yaml
configmap/keyvalcfgmap created

root@k8s-master:~# kubectl run nginx --image nginx --dry-run=client -o yaml >

root@k8s-master:~# vi nginx.yaml

root@k8s-master:~# cat configmap.yaml
apiVersion: v1
```

```
data:
  keyvalcfgmap-vars.txt: |
    key1=value1
    keys2=value2
kind: ConfigMap
metadata:
  creationTimestamp: null
  name: keyvalcfgmap

root@k8s-master:~# cat nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - name: fileconfig
      mountPath: "/opt/"
  volumes:
  - name: fileconfig
    configMap:
      name: keyvalcfgmap

root@k8s-master:~# kubectl apply -f nginx.yaml
pod/nginx created

root@k8s-master:~# kubectl exec nginx -- cat /opt/keyvalcfgmap-vars.txt
key1=value1
keys2=value2
```

54. Create an busybox pod and load environment values from the above configmap keyvalcfgmap and exec into the pod and verify the environment variables store it on /opt/keyvalcfgmap-vars.txt. and delete the pod

```
#Imperative/Recommended approach

root@k8s-master:~# cat keyvalcfgmap-vars.txt
key1=value1
keys2=value2

root@k8s-master:~# kubectl create configmap keyvalcfgmap --from-file keyvalcf

root@k8s-master:~# vi configmap.yaml

root@k8s-master:~# cat configmap.yaml
apiVersion: v1
data:
  keyvalcfgmap-vars.txt: |
    key1=value1
    keys2=value2
kind: ConfigMap
```

```
metadata:
  creationTimestamp: null
  name: keyvalcfgmap

root@k8s-master:~# kubectl apply -f configmap.yaml
configmap/keyvalcfgmap created

root@k8s-master:~# kubectl run nginx --image busybox --dry-run=client -o yaml

root@k8s-master:~# vi busybox.yaml

root@k8s-master:~# cat busybox.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
spec:
  containers:
  - image: busybox:1.28
    name: busybox
    command: [ "/bin/sh", "-c"]
    args: ["sleep 4800"]
    env:
    - name: setenv-from-configmap
      valueFrom:
         configMapKeyRef:
            name: keyvalcfgmap
            key: key1

root@k8s-master:~# kubectl apply -f busybox.yaml
pod/busybox created

root@k8s-master:~# kubectl exec busybox -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=busybox
setenv-from-configmap=value1
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PORT=443
NGINX_PORT_80_TCP_PORT=80
NGINX_PORT_80_TCP_ADDR=10.104.191.182
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
NGINX_SERVICE_PORT_80_80=80
NGINX_PORT=tcp://10.104.191.182:80
NGINX_PORT_80_TCP=tcp://10.104.191.182:80
KUBERNETES_PORT_443_TCP_PROTO=tcp
NGINX_SERVICE_PORT=80
NGINX_PORT_80_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT=443
NGINX_SERVICE_HOST=10.104.191.182
KUBERNETES_SERVICE_PORT_HTTPS=443
HOME=/root
```

54. Create a pod called busybox with the image busybox which executes command sleep 3600 and makes sure any Containers in the Pod, all processes run with user ID 1000 and with group id 2000 and verify.

```
root@k8s-master:~# cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
spec:
  securityContext:
          runAsUser: 1000
          runAsGroup: 2000
  containers:
  - image: busybox:1.28
    name: busybox
    command: [ "/bin/sh", "-c"]
    args: ["sleep 3600"]

root@k8s-master:~# kubectl apply -f pod.yaml
pod/busybox created

root@k8s-master:~# kubectl exec busybox -- id
uid=1000 gid=2000
```

55. Create the same pod as above this time set the securityContext for the container as well and verify that the securityContext of container overrides the Pod level securityContext.

```
root@k8s-master:~# cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
spec:
  securityContext:
          runAsUser: 1000
          runAsGroup: 2000
  containers:
  - image: busybox:1.28
    name: busybox
    command: [ "/bin/sh", "-c"]
    args: ["sleep 3600"]
    securityContext:
          runAsUser: 3000
          runAsGroup: 4000

root@k8s-master:~# kubectl apply -f pod.yaml
pod/busybox created
```

```
root@k8s-master:~# kubectl exec busybox -- id
uid=3000 gid=4000
```

56. Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.

```
apiVersion: v1
kind: Pod
metadata:
  name: cpu-demo
spec:
  containers:
  - name: cpu-demo-ctr
    image: busybox
    resources:
      limits:
        cpu: "1"
      requests:
        cpu: "0.5"
    command: ["/bin/sh", "-c"]
    args: ["sleep 4000"]
```

57. List all the events sorted by timestamp and put them into file.log and verify

```
root@k8s-master:~# kubectl get events --sort-by='.metadata.creationTimestamp'
```

58. Create the pod with this kubectl create -f https://raw.githubusercontent.com/lerndevops/educka/master/exam-prep/not-running.yml The pod is not in the running state. Debug it.

```
root@k8s-master:~# kubectl describe pod/not-running

Warning  Failed     33s (x3 over 81s)  kubelet, k8s-worker  Failed to pull im
Warning  Failed     33s (x3 over 81s)  kubelet, k8s-worker  Error: ErrImagePu
Normal   BackOff    19s (x3 over 81s)  kubelet, k8s-worker  Back-off pulling
Warning  Failed     19s (x3 over 81s)  kubelet, k8s-worker  Error: ImagePullE

Note: Image is name is wrong "redi" - must be "redis"
```

59. Create a yaml file called nginx-deploy.yaml for a deployment of three replicas of nginx, listening on the container's port 80. They should have the labels role=webserver and app=nginx. The deployment should be named nginx-deploy. Expose the deployment with a NodePort and use a curl statement on the IP address of the NodeIP to export the output to a file titled output.txt.

```
root@k8s-master:~# cat nginx-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
    role: webserver
  name: nginx-deploy
spec:
  replicas: 3
```

```
  selector:
    matchLabels:
      app: nginx
      role: webserver
  strategy: {}
  template:
    metadata:
      labels:
        app: nginx
        role: webserver
    spec:
      containers:
      - image: nginx
        name: nginx
        ports:
          - name: http
            containerPort: 80

root@k8s-master:~# cat nginx-deploy-service.yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: nginx
  name: nginx-deploy-service
spec:
  ports:
  - name: "80"
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: NodePort
status:
  loadBalancer: {}
```

60. Create a pod called `secret-1401` in the `admin1401` namespace using the `busybox` image. The container within the pod should be called `secret-admin` and should sleep for `4800` seconds.
The container should mount a `read-only` secret volume called `secret-volume` at the path `/etc/secret-volume`. The secret being mounted has already been created for you and is called `dotfile-secret`.

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: secret-1401
  name: secret-1401
  namespace: admin1401
spec:
  containers:
  - image: busybox
    imagePullPolicy: Always
    name: secret-admin
```

```
    command: ["/bin/sh", "-c", "sleep 4800"]
    volumeMounts:
    - mountPath: /etc/secret-volume
      name: secret-volume
      readOnly: true
  volumes:
  - name: secret-volume
    secret:
      defaultMode: 420
      secretName: dotfile-secret
```

61. Create multi-container pod with one container should use secret as environment variable and other should use configmap as volume.

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: multi-cont
  name: multi-cont
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /tmp
      name: from-configmap
  - image: redis
    name: redis
    env:
    - name: app_username
      valueFrom:
         secretKeyRef:
             name: appcreds
             key: username
  volumes:
  - name: from-configmap
    configMap:
      name: appconfig
```

62. Take the backup of ETCD at the location `/tmp/etcd-backup.db` on the master node

```
root@k8s-master:~# etcdctl snapshot save /tmp/etcd-backup.db --cacert=/etc/ku
Snapshot saved at /tmp/etcd-backup.db

root@k8s-master:~# etcdctl snapshot status /tmp/etcd-backup.db
8cec6ba7, 126559, 1381, 4.0 MB
```

63. Use JSON PATH query to retrieve the `osImages` of all the nodes and store it in a file `/opt/outputs/nodes_os_x43kj56.txt`

```
kubectl get nodes -o jsonpath="{.items[*]['status.nodeInfo.osImage']}" > /opt
```

64. Create a static pod on `node01` called `nginx-critical` with image `nginx`. Create this pod on node01 and make sure that it is recreated/restarted automatically in case of a failure.

```
master $ ssh node01

node01 $ ps -ef | grep kubelet
root      2810     1  2 08:55 ?        00:01:16 /usr/bin/kubelet --bootstrap-
root     12096 11926  0 09:46 pts/1    00:00:00 grep --color=auto kubelet

node01 $ cat /var/lib/kubelet/config.yaml
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s

node01 $ mkdir -p /etc/kubernetes/manifests

node01 $ vi /etc/kubernetes/manifests/nginx-static.yaml

apiVersion: v1
kind: Pod
metadata:
  labels:
    app: web
  name: nginx-critical
```

```
spec:
 containers:
 - image: nginx
   name: nginx
 restartPolicy: Always
```

68. Create an nginx pod called `nginx-deploy` using image `nginx`, expose it internally with a service called `nginx-resolver-deploy`. Test that you are able to look up the service and pod names from within the cluster. Use the image: busybox:1.28 for dns lookup. Record results in `/root/nginx.svc` and `/root/nginx.pod`

```
root@k8s-master:~# kubectl get svc
NAME                   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
kubernetes             ClusterIP   10.96.0.1        <none>        443/TCP
nginx                  NodePort    10.104.191.182   <none>        80:32090/TC
nginx-deploy-service   NodePort    10.96.13.97      <none>        80:31426/TC

root@k8s-master:~# kubectl get po -o wide
NAME                                  READY   STATUS    RESTARTS   AGE     IP
busybox                               1/1     Running   11         21h     10
multi-cont                            2/2     Running   0          6h33m   10
nginx-deploy-scale-587cc9d847-sv29n   1/1     Running   0          6h57m   10
nginx-sample-k8s-worker               1/1     Running   0          8h      10
test-pd                               1/1     Running   0          8h      10

root@k8s-master:~# kubectl exec busybox -- nslookup nginx-deploy-service.defa
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      nginx-deploy-service.default.svc.cluster.local
Address 1: 10.96.13.97 nginx-deploy-service.default.svc.cluster.local

Hi there
root@k8s-master:~# kubectl exec busybox -- nslookup nginx-deploy-service.defa
M
root@k8s-master:~# kubectl exec busybox -- nslookup 10-244-1-78.default.pod.c
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10-244-1-78.default.pod.cluster.local
Address 1: 10.244.1.78

root@k8s-master:~# kubectl exec busybox -- nslookup 10-244-1-78.default.pod.c
```

69. Create a new service account with the name `pvviewer`. Grant this Service account access to `list` all PersistentVolumes in the cluster by creating an appropriate cluster role called `pvviewer-role` and ClusterRoleBinding called `pvviewer-role-binding`.
Next, create a pod called `pvviewer` with the image: `redis` and serviceAccount: `pvviewer` in the default namespace

```
master $ kubectl create sa pvviewer

master $ kubectl create clusterrole pvviewer-role --verb=list --resource=pers

master $ kubectl create clusterrolebinding pvviewer-role-binding --clusterrol
```

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - image: redis
    name: redis
  serviceAccountName: pvviewer
```

70. Taint the worker node node01 to be Unschedulable. Once done, create a pod called `dev-redis`, image `redis:alpine` to ensure workloads are not scheduled to this worker node. Finally, create a new pod called `prod-redis` and image `redis:alpine` with toleration to be scheduled on node01. key:env_type, value:production, operator: Equal and effect:NoSchedule

```
```

71. Setup Kuberenets Master and node with “kubeadm” and wait for nodes are becoming ready.

```
https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install
```

72. List all persistent volumes by it’s capacity

```
kubectl get pv --sort-by=.spec.capacity.storage
```

74. List all pods which are using the service called baz and place the pods names in /opt/bazpods.txt

```
Identify the labels/selectors of the service and see what pods are refering t
```

75. Created a pod called init-demo with nginx image  and make sure to run the pod if we have the file in place “/work-dir/index.html” ( Init container question)

```
apiVersion: v1
kind: Pod
metadata:
name: init-demo
spec:
containers:
- name: nginx
image: alpine
ports:
- containerPort: 80
command: ["sh", "-c"]
args:
- if [[ -f /work-dir/index.html ]]; then
sleep 10000; fi
volumeMounts:
- name: workdir
mountPath: /work-dir
# These containers are run during pod initialization
initContainers:
- name: install
image: alpine
```

```
command: ["touch", "/work-dir/index.html", "sleep", "10000"]
volumeMounts:
- name: workdir
mountPath: /work-dir
dnsPolicy: Default
volumes:
- name: workdir
hostPath:
path: /work-dir
```

# Tips:

- Imperative Commands: Create all resources with imperative commands as mush as possible.
- As you might have seen already, it is a bit difficult to create and edit YAML files. Especially in the CLI. During the exam, you might find it difficult to copy and paste YAML files from browser to terminal. Using the kubectl run command can help in generating a YAML template. And sometimes, you can even get away with just the kubectl run command without having to create a YAML file at all. For example, if you were asked to create a pod or deployment with specific name and image you can simply run the kubectl run command.

  Use the below set of commands and try the previous practice tests again, but this time try to use the below commands instead of YAML files. Try to use these as much as you can going forward in all exercises

  Reference (Bookmark this page for exam. It will be very handy):
  https://kubernetes.io/docs/reference/kubectl/conventions/
  https://kubernetes.io/docs/reference/kubectl/cheatsheet/
  https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#create

  - **Create an NGINX Pod**
    - `kubectl run --generator=run-pod/v1 nginx –image=nginx`
  - **Generate POD Manifest YAML file** (-o yaml). Don't create it( – dry-run)
    - `kubectl run --generator=run-pod/v1 nginx --image=nginx --dry-run -o yaml`
  - **Create a deployment**
    - `kubectl create deployment –image=nginx nginx`
  - **Generate Deployment YAML file** (-o yaml). Don't create it( – dry-run)
    - `kubectl create deployment --image=nginx nginx --dry-run -o yaml`
  - **Generate Deployment YAML file** (-o yaml). Don't create it(--dry-run) with 4 Replicas ( – replicas=4)
    - `kubectl create deployment --image=nginx nginx --dry-run -o yaml > nginx-deployment.yaml`
    - **IMPORTANT:** `kubectl create deployment` does not have a `--replicas` option. You could first create it and then scale it using the `kubectl scale` command.
      - Save it to a file - (If you need to modify or add some other details)
      - `kubectl create deployment --image=nginx nginx --dry-run=client -o yaml > nginx-deployment.yaml`
      - You can then update the YAML file with the replicas or any other field before creating the deployment.
  - **Create a Service named redis-service of type ClusterIP to expose pod redis on port 6379**

        - `kubectl expose pod redis --port=6379 --name redis-service --dry-run=client -o yaml` (This will automatically use the pod's labels as selectors) Or
        - `kubectl create service clusterip redis --tcp=6379:6379 --dry-run=client -o yaml` (This will not use the pods labels as selectors, instead it will assume selectors as **app=redis.** You cannot pass in selectors as an option. So it does not work very well if your pod has a different label set. So generate the file and modify the selectors before creating the service)
    - **Create a Service named nginx of type NodePort to expose pod nginx's port 80 on port 30080 on the nodes:**
        - `kubectl expose pod nginx --port=80 --name nginx-service --type=NodePort --dry-run=client -o yaml` (This will automatically use the pod's labels as selectors, but you cannot specify the node port. You have to generate a definition file and then add the node port in manually before creating the service with the pod.) Or
        - `kubectl create service nodeport nginx --tcp=80:80 --node-port=30080 --dry-run=client -o yaml` (This will not use the pods labels as selectors)
        - Both the above commands have their own challenges. While one of it cannot accept a selector the other cannot accept a node port. I would recommend going with the `kubectl expose` command. If you need to specify a node port, generate a definition file using the same command and manually input the nodeport before creating the service.
- Practise jsonpath as much as possible.
        - https://kodekloud.com/p/json-path-quiz
        - https://mmumshad.github.io/json-path-quiz/index.html#!/?questions=questionskub1
        - https://mmumshad.github.io/json-path-quiz/index.html#!/?questions=questionskub2