

CKA SYLLABUS

STORAGE - 10%

WORKLOAD AND SCHEDULING - 15%

CLUSTER ARCHITECTURE , CONFIGURATION - 25%

SERVICES AND NETWORKING - 20%

TROUBLESHOOTING - 30%

Kubernetes Installation _____

12GB RAM Required 16 GB Recommended

4GB - base OS VMware - 512MB UBUNTU- 4GB WORKER NODE - 4 GB

HDD - 40GB Free Space

Minimum 4 cores required

WE USE ONE MASTER AND 2 WORKERS ARCHITECTURE

Step 1

Create 3 images of ubuntu run on different ipaddress and network to be "bridge"

System names

Ubuntu-Master - 192.168.1.101 Ubuntu-Worker1 - 192.168.1.102 Ubuntu-Worker2 - 192.168.1.103

Sudo hostnamectl set-hostname Ubuntu-Master - in first system

Sudo hostnamectl set-hostname Ubuntu-Worker1 - in second system

Sudo hostnamectl set-hostname Ubuntu-Worker2 - in

Third system

Step 2

IN ALL MACHINES

sudo apt-get update

Sudo apt-get install docker.io

docker —version

Enable the docker service Sudo su

Systemctl enable docker Systemctl start docker Systemctl status docker

Step 3

IN ALL MACHINES

Curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add

Step 4

IN ALL MACHINES

Add Kubernetes repo

Sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-cenial main"

Sudo apt-get install kubeadm kubelet kubectl

Sudo apt-mark hold kubeadm kubelet kubectl

Wait for 10 mins

Kubeadm version

Sudo swapoff -a (IN ALL MACHINES) Step 5

IN Ubuntu-Master Machine

Sudo kubeadm init —pod-network-cidr=10.244.0.0/16

It will display kubeadm join (KEEP BACKUP OF THE STATEMENT)

Mkdir -p \$HOME/.kube

Sudo cp -l /etc/kubernetes/admin.conf \$HOME/.kube/ config

Sudo chown \$(id -u):\$(id -g) \$HOME/.kube/config Sudo kubectl apply

<https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

Kubectl get pods —all-namespaces

Step 6

IN ALL WORKER MACHINES

Copy back up kubeadm join command fully with given parameter

Run in all worker machines

COME BACK TO MASTER MACHINE Kubectl get nodes

```
43 git clone https://github.com/vishyemails/YAML.git
44 kubectl get pods
45 kubectl get po
46 ls
47 kubectl create -f example1.yml
48 kubectl get pods
49 kubectl get pods -o wide
50 kubectl get pods -o yaml
51 kubectl describe pod tomcat-pod
52 kubectl get pods -o wide
53 ping 10.244.1.2
54 kubectl exec -it tomcat-pod -- /bin/sh
```

QUESTION 1. create a new pod called admin-pod with image busybox. Allow it to be able to set system_time. Container should sleep for 3200 seconds

```
40 cd CKA
41 ls
42 kubectl get nodes
43 git clone https://github.com/vishyemails/YAML.git
44 kubectl get pods
45 kubectl get po
46 ls
47 kubectl create -f example1.yml
48 kubectl get pods
49 kubectl get pods -o wide
50 kubectl get pods -o yaml
51 kubectl describe pod tomcat-pod
52 kubectl get pods -o wide
53 ping 10.244.1.2
54 kubectl exec -it tomcat-pod -- /bin/sh
55 history
56 kubectl delete pod tomcat-pod
57 kubectl get pods -o wide
58 clear
59 alias g=kubectl
60 alias g=kubectl
61 g get nodes
62 g get pods
66 g run admin-pod --image=busybox --dry-run=client -o yaml --
command sleep 3200
```

```

67 g get pods
68 g run admin-pod --image=busybox --dry-run=client -o yaml --
command sleep 3200
69 g get pods
70 g run admin-pod --image=busybox --dry-run=client -o yaml --
command sleep 3200
71 g get pods
72 g run admin-pod --image=busybox --dry-run=client -o yaml --
command sleep 3200 | tee example2.yaml
73 g run admin-pod --image=busybox --dry-run=client -o yaml --
command sleep 3200 | tee example3.yaml
74 echo 'securityContext :
  capabilities :
    add : ["NET_ADMIN", "SYS_TIME"]
,'
75 echo 'securityContext :
  capabilities :
    add : ["NET_ADMIN", "SYS_TIME"]
' | tee -a example3.yaml
76 g get pods
77 g get pods
78 g delete -f example2.yaml
79 g delete pod admin-pods
80 g create -f example3.yaml

```

REPLICATION CONTROLLER

```

82 g get pods
83 g delete -f example3.yaml
84 g create -f example4.yml
85 g get pods
86 g get pods -o wide
87 g get pods -l app=tomcat-app
88 g describe rc tomcat-rc
89 g scale rc tomcat-rc --replicas=9
90 g get pods -l app=tomcat-app
91 g get pods -o wide
92 g scale rc tomcat-rc --replicas=3
93 g get pods -o wide
94 g delete rc tomcat-rc
95 g delete -f example4.yml
96 history

```

REPLICA SETS

```
g create -f example5.yml
98 g get pods
99 g get pods -l tier=frontend
100 g get pods -l tier=backend
101 g get pods -l tier=backend
102 g get rs tomcat-rd -o wide
103 g get rs tomcat-rs -o wide
104 g describe rs tomcat-rs
105 g get po -o wide
106 g scale rs tomcat-rs --replicas=9
107 g get po -o wide
108 g scale rs tomcat-rs --replicas=3
109 g get po -o wide
110 g delete rs tomcat-rs
111 g delete -f example5.yml
```

QUESTION 2 : deploy a web-load-5461 pod using nginx:1.17 with the label set to tier=web

```
1 g run web-load-5461 --image=nginx:1.17 --labels tier=web -o yaml |
tee example6.yaml
2 g get po
3 g get pods --show-labels
4 history
```

DEPLOYMENT

```
1 g create -f examle7.yml
2 g create -f example7.yml
3 g get deploy
4 g get deploy -l app=tomcat-app
5 g get rs -l app=tomcat-app
6 g get po
7 g get po -o wide
8 g describe deploy tomcat-deploy
9 g deploy
```

```
10 g get deploy
11 g scale deployment tomcat-deploy --replicas=9
12 g get po
13 g scale deployment tomcat-deploy --replicas=1
14 g get po
15 history
```

ROLLOUTS

```
16 g get deploy -o wide
17 g set image deploy tomcat-deploy tomcat-containers=nginx:1.9.1
18 g rollout status deployment/tomcat-deploy
19 g get deploy -o wide
20 g scale deployment tomcat-deploy --replicas=9
21 g get deploy -o wide
22 g set image deploy tomcat-deploy tomcat-containers=vishymails/
tomcatimage:1.0
23 g rollout status deployment/tomcat-deploy
24 g get deploy -o wide
```

ROLLBACK

IN OLDER VERSION TO START IMAGE RECORDING YOU HAVE TO ADD BELOW ANNOTATION BUT IN LATEST VERSIONS THIS ANNOTATION IS NOT NECESSARY

```
g set image deploy tomcat-deploy tomcat-containers=nginx:1.91 --
record
```

```
26 g set image deploy tomcat-deploy tomcat-containers=nginx:1.91
27 g get deploy -o wide
28 g rollout status deployment/tomcat-deploy
29 g get deploy -o wide
```

IN ANOTHER TERMINAL

```
43 alias g=kubectl
44 g rollout history deployment/tomcat-deploy
45 g rollout undo deployment/tomcat-deploy
46 g rollout status deployment/tomcat-deploy
```

Question 03) Create a new deployment called web-proj-268 with image

nginx:1.16 and one replica. Next, upgrade the deployment to version 1.17 using rolling update.

Make sure that the version upgrade is recorded in the resource annotation.

```
  apiVersion : v1
kind : PersistentVolume
metadata :
  name : kube-pv
spec :
  storageClassName : standard
  capacity :
    storage : 1Gi
  volumeMode : Filesystem
  accessModes :
    - ReadWriteOnce
  hostPath :
    path : /mnt/nginx
51 g create deployment web-proj-268 --image=nginx:1.16
or
51 g create deployment web-proj-268 --image=nginx:1.16 -o yaml | tee
example8.yaml
```

```
52 g describe deployment web-proj-268
53 g set image deployment web-proj-268 nginx=nginx:1.17 --record
54 g rollout history deployment web-proj-268
55 g get deploy -o wide

57 g get deploy
58 g delete deploy tomcat-deploy
59 g delete deploy web-proj-268
60 g create deployment web-proj-268 --image=nginx:1.16 -o yaml | tee
example8.yaml
```

Question 04) Create a new deployment web-003, scale this deployment to 3 replicas, make sure desired number of pods are always running.

```
alias g=kubectl
105 g create deployment web-003 --image=nginx --replicas=3 -o yaml |
tee example9.yaml
106 g get pods
112 g get po -l app=web-003
113 g delete pod web-003-75d568fccc-bjzq6
114 g get po -l app=web-003
```

CHECKING MASTER NODE DESCRIPTION AND LOGS

```
107 g get pods -A
108 g logs pod kube-controller-manager-kmaster -n kube-system
109 g describe pod kube-controller-manager-kmaster -n kube-system
110 sudo find / -name kube-controller-man* | grep bin
111 g logs kube-controller-manager-kmaster -n kube-system
```

```
98 ls /etc
99 ls /etc/kubernetes
100 ls /etc/kubernetes/manifests
101 clear
102 sudo vi /etc/kubernetes/manifests/kube-controller-manager.yaml
```

Question 6) deploy a web-load-5461 pod using nginx:1.17 with the label set to tier=web

```
108 g delete pod web-load-5461
109 g run web-load-5461 --image=nginx:1.17 --labels tier=web
110 g get pods --show-labels
```

Create static Pods

Static Pods are managed directly by the kubelet daemon on a specific node, without the API server observing them. Unlike Pods that are managed by the control plane (for example, a Deployment); instead, the kubelet watches each static Pod (and restarts it if it fails).

Static Pods are always bound to one Kubelet on a specific node.

The kubelet automatically tries to create a mirror Pod on the Kubernetes API server for each static Pod. This means that the Pods running on a node are visible on the API server, but cannot be controlled from there. The Pod names will be suffixed with the node hostname with a leading hyphen. Note: If you are running clustered Kubernetes and are using static Pods to run a Pod on every node, you should probably be using a DaemonSet

instead.

Note: The spec of a static Pod cannot refer to other API objects (e.g., ServiceAccount, ConfigMap, Secret, etc).

Note: Static pods do not support ephemeral containers.

Q 7) Create static pod on node07 / kworker1 called static-nginx with image nginx and you have to make sure that it is recreated/restarted automatically in case of any failure happens.

STEP 1 - IN MASTER NODE

```
1 clear
2 g get nodes
3 ps -ef | grep kubelet
4 sudo grep static /var/lib/kubelet/config.yaml
5 g run static-nginx --image=nginx --dry-run=client -o yaml
6 g run static-nginx --image=nginx --dry-run=client -o yaml >
example10.yml
7 cat example10.yml | ssh kworker1 "tee static-pod.yaml"
8 g get pods
9 g get pods -o wide
10 history
```

STEP 2 - In KWORKER 1 NODE

```
39 ps -ef | grep kubelet
40 sudo grep static /var/lib/kubelet/config.yaml
41 sudo cp static-pod.yaml /etc/kubernetes/manifests/.
42 ls /etc/kubernetes/manifests/
43 sudo vi /etc/kubernetes/manifests/static-pod.yaml
```

NO NEED TO USE ANY START COMMANDS YAML FILES WILL BE EXECUTED AND VISIBLE IN MASTER NODE IF U PLACE DEFINED YML FILE IN MANIFESTS FOLDER

STEP 3 - IN MASTER NODE

```
8 g get pods
9 g get pods -o wide
10 history
```

MULTICONTAINER POD

```
17 kubectl get pods
18 clear
19 g create -f example11.yml
20 g get po -o wide
21 g exec -it multicontainer-pod -- /bin/sh
22 g exec -it multicontainer-pod --container mytomcat -- /bin/sh
```

Q 8) Create a pod called pod-multi with 2 containers as it is described below:

Container 1 : name:container1, image: nginx

Container 2 : name:container2, image: busybox, command: sleep 4800

```
  apiVersion: v1
kind: Pod
metadata:
  labels:
    run: pod-multi
  name: pod-multi
spec:
  containers:
  - image: nginx
    name: container1
  - image: busybox
    name: container2
    command: ["sleep","4800"]
```

```
g create -f multipod.yaml
g get pods -o wide
g describe pod pod-multi
```

SERVICES

1. NODE PORT
2. LOADBALANCER
3. CLUSTER IP

NODEPORT

```
apiVersion : apps/v1
kind : Deployment
metadata :
  name : tomcat-deploy
  labels :
    app : tomcat-app

spec :
  replicas : 3
  selector :
    matchLabels :
      app : tomcat-app
  template :
    metadata :
      labels :
        app : tomcat-app
    spec :
      containers :
        - name : tomcat-containers
          image : vishymails/tomcatimage:1.0
          ports :
            - containerPort : 8080
```

```
apiVersion : v1
kind : Service
metadata :
  name : my-service
  labels :
```

```
  app : tomcat-app
spec :
  selector :
    app : tomcat-app
  type : NodePort
  ports :
    - nodePort : 31000
      port : 80
      targetPort : 8080
```

```
1 g get po
2 g delete -f example11.yml
3 g apply -f example12-1.yml
4 g get po
5 g apply -f example12-2.yml
6 kubectl get service -l app=tomcatapp
7 kubectl get service -l app=tomcat-app
8 kubectl get service -l app=tomcat-app -o wide
9 g get po -o wide
10 g describe svc my-service
11 curl
12 curl http://10.244.1.44:8080
13 curl http://10.244.1.35:8080
14 g describe svc my-service
15 curl http://10.244.2.35:8080
16 curl http://10.244.2.36:8080
17 g get odes
18 g get nodes -o wide
19 curl http://172.31.26.246:31000
20 history
```

LOADBALANCER SERVICE

```
apiVersion : v1
kind : Service
metadata :
  name : my-service
```

```
labels :  
  app : tomcat-app  
spec :  
  selector :  
    app : tomcat-app  
  type : LoadBalancer  
  ports :  
    - nodePort : 31000  
      port : 8080  
      targetPort : 8080
```

```
23 g delete svc my-service  
24 g apply -f example12-3.yml  
25 g describe service my-service  
26 curl http://10.110.137.197:8080  
27 g get svc -o wide
```

Q15) Expose "audit-web-app" pod to by creating a service "audit-web-app-service" on port 30002 on nodes of given cluster.

Note : Now given web application listens on port 8080

```
36 g run audit-web-app --image=vishymails/tomcatimage:1.0 --  
port=8080  
37 g get po  
38 g describe pod audit-web-app  
39 g expose pod audit-web-app --name=audit-web-app-svc --  
type=NodePort --dry-run=client -o yaml > example13.yaml  
40 g apply -f example13.yaml  
41 g get pods -o wide | grep audit  
42 g describe svc audit-web-app-svc
```

NAMESPACES

Q 9) Create a pod called delta-pod in defence namespace belonging to the development environment (env=dev) and frontend tier (tier=front), image: nginx:1.17

```
g create ns defense
2 g run delta-pod --image=nginx:1.17 --labels env=dev,tier=front -n
defense
3 g get pods
4 g get pods -n defense
5 g describe pods delta-pod -n defense
6 g get pods
7 g describe pods audit-web-app
8 history
9 g get namespace
12 g get all
13 g get all -n defense
14 g get all --namespace defense
16 g api-resources
17 g api-resources --namespaced=true
18 g api-resources --namespaced=false
```

Q10) Get web-load-5461 pod details in json format and store it in a file at /opt/output/web-load-5461-j070822n.json

```
21 g get po
22 g apply -f example6.yaml
23 g get po
25 ls /opt/output
26 sudo ls /opt/output
27 g get pods web-load-5461 -o json | sudo tee /opt/output/web-
load-5461-j070822n.json
28 sudo ls /opt/output
```

```
29 sudo mkdir /opt/output
30 g get pods web-load-5461 -o json | sudo tee /opt/output/web-
load-5461-j070822n.json
31 sudo ls /opt/output
32 sudo cat /opt/output/web-load-5461-j070822n.json
```

Q12) A new application finance-audit-pod is deployed in finance namespace. Find out what is wrong with it and fix the issue.

NOTE: No configuration changes allowed, you can only delete or recreate the pod.

```
35 g create ns finance ; g run finance-audit-pod --image=busybox -n
finance --command speep 180
36 g get po
37 g get po -n finance
38 g describe pod finance-audit-pod -n finance
39 g describe pod finance-audit-pod -n finance | grep -i command -A5
40 g get pods finance-audit-pod -n finance -o yaml | tee example14.yaml
41 grep sleep example14.yaml
42 g delete pods finance-audit-pod -n finance --grace-period=0 --force
43 g create -f example14.yaml
44 g get pods -n finance
```

Q 13 :use JSONPath query to retrieve our OS images of all K8s nodes and store it in a file ~/allNodeOSImages8.txt

```
g get nodes -o jsonpath='{.items[*].status.nodeInfo.osImage}' | tee
allNodeOSImages8.txt
```

PERSISTENCE VOLUMES

1. PV - PERSISTENT VOLUME

2. PVC - PERSISTENT VOLUME CLAIMS

PV - PLACE OF STORAGE IN CLUSTER

PVC - REQUEST FOR STORAGE

LIFE CYCLE OF PERSISTENT VOLUMES

1. PROVISIONING

2. BINDING

3. USING

4. RECALIMING

PROVISIONING - 2 TYPES

1. STATIC

2. DYNAMIC

STATIC - PV NEED TO BE CREATED BEFORE PVC

DYNAMIC - PV IS CREATED AT SAME TIME OF PVC

CACHED VOLUME - EMPTY DIR VOLUME

TEMPORARY SPACE

emptydir demo

```
1 g apply -f example15.yml
2 kubectl get po
3 kubectl exec -it tomcat-pod -- /bin/bash
4 g describe pod tomcat-pod
5 kubectl exec -it tomcat-pod -- /bin/bash
6 g delete pod tomcat-pod
7 g apply -f example15.yml
8 kubectl exec -it tomcat-pod -- /bin/bash
9 history
```


HOSTPATH

MOUNTS A FILE OR FOLDER FROM THE HOST NODES FILE SYSTEM IN TO THE POD

```
19 g get po
20 g apply -f example16.yml
21 g get po -o wide
22 kubectl exec -it tomcat-hostpath -- /bin/bash
23 g describe pod tomcat-hostpath
24 g get po
25 g delete pod tomcat-hostpath
26 g apply -f example16.yml
27 kubectl exec -it tomcat-hostpath -- /bin/bash
```

MULTI CONTAINER POD SHARING COMMON VOLUME

```
29 g create -f example18.yml
30 g get po
31 g exec -it multicontainer-pod1 --container producer -- /bin/bash
32 g exec -it multicontainer-pod1 --container consumer -- /bin/bash
```

PERSISTENT VOLUME CLAIMS

demo 1 : WORKS WHEN DEFAULT PV IS ALREADY CONFIGURED

```
apiVersion : v1
kind : PersistentVolume
metadata :
  name : kube-pv
spec :
  storageClassName : standard
  capacity :
    storage : 1Gi
  volumeMode : Filesystem
  accessModes :
    - ReadWriteOnce
```

hostPath :
path : /mnt/nginx

apiVersion : v1
kind : PersistentVolumeClaim
metadata :
name : kube-pv
spec :
resources :
requests :
storage : 1Gi
volumeMode : Filesystem
accessModes :
- ReadWriteOnce

apiVersion : v1
kind : Pod
metadata :
name : pv-pod
labels :
name : pv-pod
spec :
containers :
- name : pv-pod
image : nginx
volumeMounts :
- mountPath : /test-pd
name : test-volume
ports :
- containerPort : 80

volumes :
- name : test-volume
persistentVolumeClaim :
claimName : kube-pv

DEMO 2 : WORKS WITH NFS

PV

apiVersion: v1

kind: PersistentVolume

metadata:

name: nfs-pv

spec:

accessModes:

- ReadWriteMany

capacity:

storage: 500Gi

nfs:

path: /

server: 10.218.47.252

persistentVolumeReclaimPolicy: Recycle

storageClassName: nfs

volumeMode: Filesystem

PVC

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: nfs-pvc

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 100Gi

storageClassName: nfs

volumeMode: Filesystem

volumeName: nfs-pv

Demo 3:

apiVersion : v1
kind : PersistentVolume
metadata :
 name : kube-pv
spec :
 storageClassName : standard
 capacity :
 storage : 1Gi
 volumeMode : Filesystem
 accessModes :
 - ReadWriteOnce
 hostPath :
 path : /etc/pv-store

apiVersion : v1
kind : PersistentVolumeClaim
metadata :
 name : kube-pvc
spec :
 storageClassName : standard
 resources :
 requests :
 storage : 1Gi
 volumeMode : Filesystem
 accessModes :
 - ReadWriteOnce

apiVersion : v1
kind : Pod
metadata :
 name : pv-podhostpath
 labels :
 name : pv-pod
spec :
 containers :
 - name : pv-pod
 image : nginx
 volumeMounts :

```
- mountPath : /test-pd
  name : test-volume
ports :
- containerPort : 80
nodeName: kworker1

volumes :
- name : test-volume
  persistentVolumeClaim :
    claimName : kube-pvc
```

DEMO 4

```
apiVersion : v1
kind : PersistentVolume
metadata :
  name : kube-pv
  labels :
    type : local
spec :
  storageClassName : manual
  capacity :
    storage : 1Gi
  accessModes :
    - ReadWriteOnce
  hostPath :
    path : /mnt/datas
```

```
apiVersion : v1
kind : PersistentVolumeClaim
metadata :
  name : kube-pvc

spec :
  storageClassName : manual
resources :
```

```
requests :  
  storage : 1Gi  
accessModes :  
- ReadWriteOnce
```

```
apiVersion : v1  
kind : Pod  
metadata :  
  name : pv-pod  
  labels :  
    name : pv-pod  
spec :  
  containers :  
    - name : pv-pod  
      image : nginx  
      volumeMounts :  
        - mountPath : /test-pd  
          name : test-volume  
      ports :  
        - containerPort : 80  
  
  volumes :  
    - name : test-volume  
      persistentVolumeClaim :  
        claimName : kube-pvc
```

LINKING PV WITH PVC

```
apiVersion: v1  
kind: PersistentVolume  
metadata:  
  name: pv-name  
  labels:  
    type: local  
spec:  
  storageClassName: manual  
  capacity:  
    storage: 40Gi  
  accessModes:  
    - ReadWriteOnce  
  hostPath:
```

path: "/mnt/data"

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: pvc-name
spec:
 storageClassName: manual
 accessModes:
 - ReadWriteOnce
 resources:
 requests:
 storage: 10Gi
 volumeName: pv-name

ANOTHER REFERENCING VIA VOLUME TO CLAIM OBJECT

apiVersion: v1
kind: PersistentVolume
metadata:
 name: pv0003
spec:
 storageClassName: ""
 capacity:
 storage: 5Gi
 accessModes:
 - ReadWriteOnce
 persistentVolumeReclaimPolicy: Retain
 claimRef:
 namespace: default
 name: myclaim
 nfs:
 path: /tmp
 server: 172.17.0.2

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
 name: myclaim
spec:

```
storageClassName: ""
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 5Gi
```

Q14) Create a persistent volume with given specifications:

Volume Name - pv-rnd
storage - 100Mi
Access modes - ReadWriteMany
host path - /pv/host-data-rnd

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-rnd
spec:
  capacity:
    storage: 100Mi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /pv/host-data-rnd
```

Q19) Craete a PersistentVolume, PersistentVolumeClaim and Pod with below specifications

PV - name : mypvl , Size: 100Mi, AccessModes: ReadWritemany,
Hostpath: /pv/log, Reclaim Policy: Retain
PVC - name: pv-claim-l, Storage request: 50Mi, Access Modes:
ReadWritemany

**Pod - name : my-nginx-pod, image Name: nginx, Volume:
PersistentVolumeClaim: pv-claim-l, volume mount : /log**

**apiVersion: v1
kind: PersistentVolume
metadata:
 name: mypvl
spec:
 capacity:
 storage: 100Mi
 volumeMode: Filesystem
 accessModes:
 - ReadWriteMany
 persistentVolumeReclaimPolicy: Retain
 hostPath:
 path: /pv/log**

**apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: pv-claim-l
spec:
 accessModes:
 - ReadWriteMany
 volumeMode: Filesystem
 resources:
 requests:
 storage: 50Mi**

**apiVersion: v1
kind: Pod
metadata:
 name: my-nginx-pod
spec:
 containers:**

- name : mynginx
- image: nginx
- volumeMounts:
 - mountPath: "/log"
 - name: mypd

volumes:

- name: mypd
- persistentVolumeClaim:
 - claimName: pv-claim-l

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: standard

provisioner: kubernetes.io/aws-ebs

parameters:

type: gp2

reclaimPolicy: Retain

allowVolumeExpansion: true

mountOptions:

- debug

volumeBindingMode: Immediate

apiVersion: v1

kind: Pod

metadata:

name: task-pv-pod

spec:

nodeSelector:

kubernetes.io/hostname: kube-01

volumes:

- name: task-pv-storage
- persistentVolumeClaim:
 - claimName: task-pv-claim

containers:

- name: task-pv-container
- image: nginx
- ports:
 - containerPort: 80
 - name: "http-server"
- volumeMounts:
 - mountPath: "/usr/share/nginx/html"

name: task-pv-storage

STORAGE CLASSES

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: standard

provisioner: kubernetes.io/gce-pd

parameters:

type: pd-standard

volumeBindingMode: WaitForFirstConsumer

allowedTopologies:

- matchLabelExpressions:

- key: failure-domain.beta.kubernetes.io/zone

values:

- us-central-1a

- us-central-1b

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: slow

provisioner: kubernetes.io/aws-ebs

parameters:

type: io1

iopsPerGB: "10"

fsType: ext4

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: slow

provisioner: kubernetes.io/gce-pd

parameters:
 type: pd-standard
 fstype: ext4
 replication-type: none

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: example-nfs
provisioner: example.com/external-nfs
parameters:
 server: nfs-server.example.com
 path: /share
 readOnly: "false"

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: gold
provisioner: kubernetes.io/cinder
parameters:
 availability: nova

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: fast
provisioner: kubernetes.io/vsphere-volume
parameters:
 diskformat: zeroedthick

apiVersion: storage.k8s.io/v1

kind: StorageClass
metadata:
 name: fast
provisioner: kubernetes.io/vsphere-volume
parameters:
 diskformat: zeroedthick
 datastore: VSANDatastore

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: slow
provisioner: kubernetes.io/azure-disk
parameters:
 skuName: Standard_LRS
 location: eastus
 storageAccount: azure_storage_account_name

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: slow
provisioner: kubernetes.io/azure-disk
parameters:
 storageaccounttype: Standard_LRS
 kind: managed

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: azurefile
provisioner: kubernetes.io/azure-file
parameters:
 skuName: Standard_LRS
 location: eastus
 storageAccount: azure_storage_account_name

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: portworx-io-priority-high
provisioner: kubernetes.io/portworx-volume
parameters:
 repl: "1"
 snap_interval: "70"
 priority_io: "high"

DAEMON SETS

apiVersion : apps/v1
kind : DaemonSet
metadata :
 name : fluent-ds

spec :
 template :
 metadata :
 labels :
 name : fluentd
 spec :
 containers :
 - **name :** fluentd
 image : gcr.io/google-containers/fluentd-elasticsearch:1.20
 selector :
 matchLabels :
 name : fluentd

SECRETS

COMMAND BASED CREATION

```
178 echo -n 'admin' | base64 > username.txt
179 echo -n 'password123' | base64 > password.txt
180 g get secrets
181 g create secret generic db-user-pass --from-file=./username.txt --
from-file=./password.txt
182 g get secrets
183 g describe secrets db-user-pass
```

```
apiVersion : v1
kind : Secret
metadata :
  name : secret-ssh-authentication
type : kubernetes.io/ssh-auth
data :
  ssh-privatekey : |
```

```
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAA
BlwAAAAdzc2gtcn
```

```
NhAAAAAwEAAQAAAYEAYkPOPaEGHsGwLfu1xIMDXXMeNA4y7MmDfaxr8
Ud8yyJF+l8f/eQJ
      PU7irfCElg54ogEhaGKEgUthyhpN9//79YQQIzD8MqAtxbed/
Y7ILp12Z0Yp7u6WPepFXW
```

```
UZU9T40oNcfwUWEd9IE69tmYUqTIIQ5fX0zPFZYrrplypZUe23gVy52Ojg3
23Bv67rKlpb
      Ds5mxZTucNIYSjGrgjcBR+ZITgjMpVUUxVyaZjLeKftRKOLbNiosq/
bxijNZ4Rw/DFVgVX
      P7dEJih/
3KJhri0IkuYUflmFhnFHHvdWsZUfcu6lwk7wmYzbR8FEiSxYWC1ui0OKkz51
Ua
```

```
NRgJPp5LBtOYqCVJqV5nEJzzYvB1Fg2hOsjez9zjci2lMANxBbj6FSI/rMmk/
UqvWXgJKv
```

```
8t+IH9uZxFDdadU1OBget12cN7zI92T2tkdmuZJC63RxBwr9HEHfqi0nQvB
Q+JkCEJoREB
      TM/
Hma9QNFoTrs+gH2Xg1a4HlOnjvSeeJC4JUxwdAAAFiBLeZwYS3mcGAAAA
```

B3NzaC1yc2

EAAAGBAMpDzj2hBh7BsC37tcZTA11zHjQOMuzJg32sa/
FHfMsiRfpfH/3kCT1O4q3whJYO
eKIBIWhihIFLYcoaTff/+/WEEJcw/
DKgLcW3nf2O5S6ddmdGKe7ulj3hT11IGVPU+NKDXH

8FFhHfZROvbZmFKk5ZUOX19MzxWWK66SMqWVHtt4Fcudjo4N9twb+u6
yiKWw7OZsWU7nDZ

WEoxq4I3AUfmSE4IzKVVMVcmmYy3in7USji2zYqLKv28YozWeEcPwxVYF
Vz+3RCYof9yi

Ya4tCJLmFH5ZhYZxR4b3VrGVH3LuiMJO8JmM20fBRiksWFgtbotDipM+dV
GjUYCT6eSwbT
mKglSaleZxCc82LwdRYNoTrI3s/
c43ItptAJ8QW4+hUpf6zJpP1Kr1l4CSr/LfiB/bmcRQ
3WnVNTgYHrddnDe8yPdk9rZHZrmSQut0cQcK/
RxB36otJOLwUPIZAHCaERAUzPx5mvUDRa

E67PoB9l4NWuB5Tp470nniQuCVMchQAAAAMBAAEAAAGAHGvkuil7Uu0S
jZOWGIUI3ubDMr
tgCofSc0FcNZ4++ehJ/wGI5Es7xSKlIz17c/
56kwEnqZxWVDi8eAKOPAn7ZKd3EVevyllb
hTHIAHEo4SD9N0vTrqBV+kIDvfLr2SzO/
f25bJvRWxeSA28eLnIY1YOVa4rhKD8tuULcab
WUsX9+zq2x67nlh1/
L3tx8bMjsPKtil5dm3fKchmhuWjDSfEDunhB8eSr4bTcy9vNk5m9L

GzytYvILuXXUP0gPxKRxXosUvmtitgRDHZYpyK07ckUrwHcTmj0zIKco4
axlCN9QZiD

DkPkA42qCVRkEzWPAntddxV9UVvApFUvjvH5RFICgIKZd3K5EwkTEyQZaq
OYYGWuLpXNjQ

7IbwhKxjd7xG7RsEBDCkiULiiYEqBV+J3syzKhH3ydGIhtqkTijeiLh/
B92pXhDrTz7e4i
PZRL/
rEk3evOcPkjaqJVdJX56a9GQ+sleisLC1p428rUnnF5AO19KB2gLbiscaJTd
AAAA
wQC/
HO2uvKz4WpS1tsp+WKw8it5frhwn+eKX9xaQTEQBGvWb1NLT7+2A53ZLG
R0Acmhzo7

OPnoNQM1JdIQCZwQLeWmIEHfd572rDEb3wwelpW4q3l9tWws6mC1bdGp
83r+5BWvHzsxzC
Rr8VvyC8rij8+REw/qJpuhuatR5Qy6lIWYdRczx7Xpot5/
cSqJrBlbl+KBQdAwe3NMVVcY

A9LP9eTazMkQ5gXOU5qtPgvF8efYvQ2BqHDwCw9xconHHJSlwAAADBAO
hD/611vqVzxHyv

dRg6mYxMFfsO0xcOKaMleGlzosl9dpk70HzD8FmNWcoqnKWNPZxQ7qFaK
bH0IoTDdU6nHp

mn7I76NjvbA9/7+4jmqBECL1e3QaU5jeVNRHocFhlzZLkZnYMNvn9JznQi1
e9ycYgy3C

1yUVSmfTcOJTpzFvNSloM3Tw3uzLZlIvliFMFhQH9zzS9Q2DMg6Ja1ndEVV
uHM0rVNJBod

Z03TF0w0jval+UYHzg0xSVb/
5UeesfXwAAAMEA3u7+8FHPOP8WJbYzWWFCyEGnO8OIFN8/
gH6vgRm7t2p6Z/
2I5RHbSz9tYOZk6HE06DX1ElenA7C08K9GidpfCLHGdkgPr3ZpeLqmjq
Bf/
35R0eNTINceZIFyKk+CMF9O2he0mmGJDnsWk2I4Qqimf8BJSbkvAbcDsv
LIB5ycrj7Z

+5vNDzi+MwY1wy0ybvDul90KyA3hV7KCx3tLVFT2hNjVkhhlJCgEJSWOxY
mrapR3XTlaxM
Nj0Dzs9BpCjQIDAAAAD2xhYnVzZXJAa21hc3RlcgECAw==

CRON JOB

apiVersion : batch/v1
kind : CronJob
metadata :
 name : cron-demo
spec :
 schedule : "*/1 * * * *"
 jobTemplate :
 spec :
 template :
 metadata :
 labels :
 name : cron-demo

 spec :
 containers :
 - name : cron-demo
 image : centos:7
 args :

```
- python
- -c
- from datetime import datetime; print('[{}] Cron job demo for CKA
batch Oracle ...'.format(datetime.now()))
restartPolicy : Never
```

LIFE CYCLE HOOKS

```
apiVersion : v1
kind : Pod
metadata :
  name : lifecycle-hook-pod
spec :
  containers :
    - name : lifecycle-container
      image : nginx
      lifecycle :
        postStart :
          exec :
            command : ["/bin/sh","-c", "echo Welcome Oracle > /usr/share/post-
start-msg"]
        preStop :
          exec :
            command : ["/usr/sbin/nginx","-s","quit"]
```

Question 05) Upgrade given cluster (master and worker node) from 1.23.8-00 to 1.24.2-00. Make sure to first drain respective node prior to update and make it available post update.

Kmaster

```
alias g=kubectl
```

```
g get nodes
sudo apt update
g drain kmaster --ignore-daemonsets
apt-cache madison kubeadm |head
sudo apt install kubeadm=1.26.3-00
sudo kubeadm upgrade apply v1.26.3-00
sudo kubeadm upgrade apply v1.26.3
sudo apt install kubelet=1.26.3
  sudo apt install kubelet=1.26.3-00
sudo systemctl restart kubelet
g uncordon kmaster
g get nodes
g drain kworker1 --ignore-daemonsets
g drain kworker1 --ignore-daemonsets --force
g get nodes
```

kworker 1

```
sudo apt update
sudo apt install kubeadm=1.26.3-00
sudo kubeadm upgrade node
sudo apt install kubelet=1.26.3-00
sudo systemctl restart kubelet
```

kmaster

```
265 g uncordon kworker1
266 g get nodes
```

Q11) Backup ETCD database and save it under /root with name of backup "etcd-backup.db"

```
sudo snap install etcd
g get pods -A |grep etcd
g describe pods etcd-node6 -n kube-system |grep Command -A 20
```

```
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \
--cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/
etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key \
snapshot save /root/etcd-backup070822
```

Utkarsh:

```
sudo apt install etcd-client
```

```
sudo -i
```

```
ETCDCTL_API=3 etcdctl snapshot save /root/etcd-backup070822 --
endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/
ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/
pki/etcd/server.key
```

Q16) Create a pod called pod-jxc, using details mentioned below:

SecurityContext

runasUser: 1000

fsGroup: 2000

Image=redis:alpine

```
250 alias g=kubectl
```

```
251 g run pod-jxc --image=redis:alpine --dry-run=client -o yaml >
example43.yml
```

```
252 g apply -f example43.yml
```

```
253 g get pods
```

```
254 g exec -it pod-jxc -- whoami
```

```
255 g exec -it lifecycle-hook-pod -- whoami
```

```
256 history
```

It is critical to understand that taints and tolerations are only enforced at the node level, and pods have the freedom to choose nodes without taints, but if all of our nodes are tainted then new pods must be with exact tolerations defined in them.

Q17) Apply taint a worked node node7 with details provided below:
Create a pod called dev-pod-nginx using image=nginx,
make sure workloads are not scheduled to this worker node (node7)
Create another pod prod-pod-nginx using image=nginx with a toleration
to be scheduled on node7.

Details :key:env_type, value:production, operator: Equal & effect:
NoSchedule

```
252 cd CKA
253 clear
254 alias g=kubectl
255 g describe node kmaster
256 g describe node kmaster | grep -i taint
257 g taint node kmaster node-role.kubernetes.io/control-
plane:NoSchedule-
258 g taint node kmaster node-role.kubernetes.io/control-
plane:NoSchedule
259 g describe node kworker1 | grep -i taint
260 g describe node kworker2 | grep -i taint
261 g taint node kworker1 env_type=production:NoSchedule
262 g run dev-nginx --image=nginx
263 g get po
264 g get po -o wide
265 g run prod-nginx --image=nginx --dry-run=client -o yaml >
example44.yml
266 g apply -f example44.yml
267 g get po
268 g get po -o wide
```

**Q18) Create a user "nec-adm". Grant nec-adm access to cluster, should
have permissions to create, list, get, update, and delete pods in nec
namespace**

**Private key exist in location: /vagrant/nec-adm.key and csr at /vagrant/
nec-adm.csr**

step 1

```
1 g create ns nec
2 cd CKA
3 openssl genrsa -out nec-adm.key 2048
```

```
4 openssl req -new -key nec-adm.key -out nec-adm.csr
5 ls nec*
6 cat nec-adm.csr | base64 | tr -d "\n"
```

step 2 - CREATE CERTIFICATE SIGNING REQUEST OBJECT

```
8 g apply -f example45.yml
9 g get csr
10 g get csr nec-adm
11 g certificate approve nec-adm
12 g get csr nec-adm
```

step 3 - CREATE ROLE AND ROLEBINDINGS

```
14 g apply -f example46.yml
15 g apply -f example47.yml
16 g get rolebindin.rbac.authorization.k8s.io -n nec
17 g get rolebinding.rbac.authorization.k8s.io -n nec
18 g get pods -n nec --as nec-adm
20 g auth can-i get pods -n nec --as nec-adm
21 g auth can-i list pods -n nec --as nec-adm
22 g auth can-i create pods -n nec --as nec-adm

19 history
```

Q20) Worker node node7 is not responding, have a look and fix the issue

node7 - kworker1
node6 - kmaster

```
1 g get nodes
2 g describe node kworker1
3 ps -ef | grep kubelet | grep lib
4 sudo cat /var/lib/kubelet/config.yaml | ssh kworker1
5 sudo cat /var/lib/kubelet/config.yaml | ssh kworker1 > mconfig.yaml
6 sudo cat /var/lib/kubelet/config.yaml | ssh kworker1 "tee
```

```
mconfig.yaml"
7 g get nodes
```

kworker 1

```
sudo systemctl status kubelet
57 sudo systemctl restart kubelet
58 sudo journalctl -u kubelet
59 ps -ef | grep kubelet
60 ls
61 sudo diff mconfig.yaml /var/lib/kubelet/config.yaml
62 sudo systemctl start kubelet
1
```

Q26

A pod "my-da

ta-pod" in data namespace is not running. Fix the issue and get it in running state.

Note: All supported definition files are placed at root.

To create question scenario just change pv1claim.yaml and remove namespace information (ensure data namespace was created already) and apply them.

```
cat pv1.yaml
####
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1
spec:
  capacity:
    storage: 100Mi
```

```
volumeMode: Filesystem
accessModes:
  - ReadWriteMany
persistentVolumeReclaimPolicy: Retain
hostPath:
  path: /data
```

```
#####
cat pv1claim.yaml
```

```
#####
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv1claim
  namespace: data
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Filesystem
  resources:
    requests:
      storage: 50Mi
```

```
#####
cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-data-pod
  namespace: data
spec:
  containers:
    - name: mydata
      image: nginx
      volumeMounts:
        - mountPath: "/maindata"
          name: mypd
```


volumes:

- name: mypd

persistentVolumeClaim:

claimName: pv1claim

#####

21) List internal IPs of all nodes of given cluster, save result to a file /root/InternalIPList

Answer should be in a format: Internal IP of 1st Node (space) Internal IP of 2nd node (in a single line)

```
13 g get nodes -o jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}' | sudo tee /root/internalIPList
14 sudo cat /root/internalIPList
```

ANOTHER SOLUTION

```
13 kubectl get nodes -o wide | grep -v "INTERNAL-IP" | awk '{print " " $6}' | tr -d "\n"
14 sudo cat /root/internalIPList
```

Q24

We have worker 3 nodes in our cluster, create a DaemonSet (name prod-pod, image=nginx) on each node except worker node8.

alias g=kubectl
g get nodes

```
for i in 6 7 8 9; do g describe node node$i |grep -i taint; done
g taint node node8 env=uat:NoSchedule
for i in 6 7 8 9; do g describe node node$i |grep -i taint; done
g create deployment prod-pod --image=nginx --dry-run=client -o yaml
g create deployment prod-pod --image=nginx --dry-run=client -o yaml |
tee prod-pod.yaml
vi prod-pod.yaml
```

```
#####
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    app: prod-pod
    name: prod-pod
spec:
  selector:
    matchLabels:
      app: prod-pod
  template:
    metadata:
      labels:
        app: prod-pod
    spec:
      containers:
        - image: nginx
          name: nginx
#####
```

```
g create -f prod-pod.yaml
g get pods -o wide
```

minikube addons enable ingress

kubectl get pods -n ingress-nginx

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: blr.ibm.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: service1
            port:
              number: 80
  - host: hyd.ibm.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: service2
            port:
              number: 80

```

Question 33 : Create a network policy for incoming web connection requests

```

g run web-test --image nginx
2 g expose pod web-test --name web-test-svc --type NodePort --port
80
3 g run connect-pod --image busybox --command sleep 4800
4 g exec -it connect-pod -- wget
5 g apply -f example49.yml
6 g describe netpol
7 g get pods
8 g get pods -o wide
9 g get svc
10 g exec -it connect-pod -- wget web-test-svc
11 history

```

Q37... mount secret in 2 pods using filesystem and environment variable

```
echo 'apiVersion: v1
kind: Pod
metadata:
  name: pod-sec-file
spec:
  containers:
  - image: redis
    name: redis
    volumeMounts:
    - name: sec1
      mountPath: "/secrets"
  volumes:
  - name: sec1
    secret:
      secretName: sec1' |tee pod-sec-file.yaml
```

```
##
echo 'apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  name: pod-sec-env
spec:
  containers:
  - image: redis
    name: redis
    env:
    - name: CONFIDENTIAL
      valueFrom:
        secretKeyRef:
          name: sec1
          key: password' |tee pod-sec-env.yaml
```

CONFIG MAP

```
management.endpoints.enabled-by-default=true
management.endpoint.info.enabled=true
management.security.enabled=false
management.endpoints.web.exposure.include=*
```

```
server.port= 9000
server.servlet.context-path=/oracle
oracleprops.greeting= Thank you and visit again - altered
oracleprops.greeting1= New Data
```

```
apiVersion : v1
kind : Pod
metadata :
  name : myconfigpod1
  labels :
    name : myconfigpod1
spec :
  containers :
    - name : myconfigpod1
      image : redis
      volumeMounts :
        - name : foo
          mountPath : "/etc/config"
          readOnly : true
  volumes :
    - name : foo
      configMap :
        name : config-map2
```

```
16 g apply -f example52.yml
17 g get po
18 g exec -it myconfigpod1 -- /bin/bash
```

HPA

```
apiVersion : autoscaling/v2
kind : HorizontalPodAutoscaler
metadata :
  name : hpa-resource-metrics-memory
  namespace : default

spec :
  scaleTargetRef :
    apiVersion : v1
    kind : Deployment
    name : tomcat-deploy
  minReplicas : 3
  maxReplicas : 10
  metrics :
    - type : Resource
      resource :
        name : memory
      target :
        type : AverageValue
        averageValue : 500Mi
```