CKA Q & A

===

===

Q 1 : Create a new Clusterrole name deployment-clusterrole , which only allows to create the following resources types
deployment,statefulSet,Daemonset

Create a new Serviceaccount named cicd-tocken in the existing namespace app-team1
bind the new clisterrole deplyment-clusterrole to the new service account cicd-tocken limited to the namespace app-team1

```
controlplane $ kubectl create clusterrole deployment-clusterrole --verb=list --
resource=deployment,statefulset,daemonset -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: deployment-clusterrole
rules:
- apiGroups:
  - apps
  resources:
  - deployments
  - statefulsets
  - daemonsets
  verbs:
  - list
controlplane $
```

```
controlplane $ kubectl create serviceaccount cicd-token -n app-team1 -o yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cicd-token
  namespace: app-team1
```

```
controlplane $ kubectl create clusterrolebinding deployment-cluster-binding --clusterrole=deployment-
clusterrole --serviceaccount=app-team1:cicd-token -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: deployment-cluster-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: deployment-clusterrole
subjects:
- kind: ServiceAccount
  name: cicd-token
  namespace: app-team1
controlplane $
```

==

Q 2 : Set a node named eks-node-0 as unavailale and reschedule all the pods running on it.

```
controlplane $ kubectl drain node02 --ignore-daemonsets --force --delete-local-data
node/node02 already cordoned
WARNING: deleting Pods not managed by ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet: default/nginx; ignoring DaemonSet-managedPods: kube-system/kube-proxy-8jp5f, kube-system/weave-net-6mg5p
evicting pod default/nginx
pod/nginx evicted
node/node02 evicted
controlplane $
controlplane $
```

==

**Q 3 : Given an existing kubernetes cluster running version 1.18.8, upgrade all of the kubernetes control plane and node components on the master node only to version 1.19.0
You are expected to upgrade kubelet and kubectl on the master node.**

Some steps I did:

```
kubectl drain mk8s-master-0 --ignore-daemonsets  (if needed add --force --delete-local-data)
kubectl get nodes
ssh mk8s-master-0
apt-get update
apt-get install -y kubeadm=1.19.0-00
kubeadm upgrade plan
kubeadm upgrade apply v1.19.0
kubeadm version

sudo kubeadm upgrade node
sudo kubeadm upgrade app...
```

+++
Performed above steps

++
kubectl drain ubuntu --ignore-daemonsets

```
node/ubuntu cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-amd64-ncpj5, kube-system/kube-proxy-g9xd7
evicting pod kube-system/coredns-66bff467f8-gkjwq
evicting pod kube-system/coredns-66bff467f8-pg7ss
```

```
pod/coredns-66bff467f8-pg7ss evicted
pod/coredns-66bff467f8-gkjwq evicted
node/ubuntu evicted
root@ubuntu:~#

root@ubuntu:~# kubectl get nodes
NAME     STATUS                     ROLES    AGE    VERSION
ubuntu   Ready,SchedulingDisabled   master   176d   v1.18.6
wnode1   Ready                      <none>   176d   v1.18.6
root@ubuntu:~#


root@ubuntu:~# ssh ubuntu


root@ubuntu:~# apt-get update
 . . .
Fetched 11.8 MB in 24s (471 kB/s)
Reading package lists... Done
root@ubuntu:~#


root@ubuntu:~# apt-get install -y kubeadm=1.19.0-00
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  kubeadm
1 upgraded, 0 newly installed, 0 to remove and 105 not upgraded.
Need to get 7,759 kB of archives.
After this operation, 709 kB disk space will be freed.
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm amd64 1.19.0-00
[7,759 kB]
Fetched 7,759 kB in 23s (330 kB/s)
(Reading database ... 62202 files and directories currently installed.)
Preparing to unpack .../kubeadm_1.19.0-00_amd64.deb ...
Unpacking kubeadm (1.19.0-00) over (1.18.6-00) ...
Setting up kubeadm (1.19.0-00) ...
root@ubuntu:~#



root@ubuntu:~# kubeadm upgrade plan
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -
oyaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.18.6
[upgrade/versions] kubeadm version: v1.19.0
I0116 21:22:51.650965   13267 version.go:252] remote version is much newer: v1.20.2; falling back to:
stable-1.19
[upgrade/versions] Latest stable version: v1.19.7
[upgrade/versions] Latest stable version: v1.19.7
```

```
[upgrade/versions] Latest version in the v1.18 series: v1.18.15
[upgrade/versions] Latest version in the v1.18 series: v1.18.15

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm
upgrade apply':
COMPONENT   CURRENT       AVAILABLE
kubelet     2 x v1.18.6   v1.18.15

Upgrade to the latest version in the v1.18 series:

COMPONENT                 CURRENT   AVAILABLE
kube-apiserver            v1.18.6   v1.18.15
kube-controller-manager   v1.18.6   v1.18.15
kube-scheduler            v1.18.6   v1.18.15
kube-proxy                v1.18.6   v1.18.15
CoreDNS                   1.6.7     1.7.0
etcd                      3.4.3-0   3.4.3-0

You can now apply the upgrade by executing the following command:

kubeadm upgrade apply v1.18.15
```

```
Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm
upgrade apply':
COMPONENT   CURRENT       AVAILABLE
kubelet     2 x v1.18.6   v1.19.7

Upgrade to the latest stable version:

COMPONENT                 CURRENT   AVAILABLE
kube-apiserver            v1.18.6   v1.19.7
kube-controller-manager   v1.18.6   v1.19.7
kube-scheduler            v1.18.6   v1.19.7
kube-proxy                v1.18.6   v1.19.7
CoreDNS                   1.6.7     1.7.0
etcd                      3.4.3-0   3.4.9-1

You can now apply the upgrade by executing the following command:

kubeadm upgrade apply v1.19.7

Note: Before you can perform this upgrade, you have to update kubeadm to v1.19.7.
```

```
The table below shows the current state of component configs as understood by this version of kubeadm.
Configs that have a "yes" mark in the "MANUAL UPGRADE REQUIRED" column require manual config
upgrade or
resetting to kubeadm defaults before a successful upgrade can be performed. The version to manually
upgrade to is denoted in the "PREFERRED VERSION" column.

API GROUP                 CURRENT VERSION   PREFERRED VERSION   MANUAL UPGRADE REQUIRED
kubeproxy.config.k8s.io   v1alpha1          v1alpha1            no
```

```
kubelet.config.k8s.io     v1beta1          v1beta1            no
_____________________________________________________________________________
root@ubuntu:~#
```

```
root@ubuntu:~# kubeadm upgrade plan v1.19.0
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -
oyaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.18.6
[upgrade/versions] kubeadm version: v1.19.0
[upgrade/versions] Latest stable version: v1.19.0
[upgrade/versions] Latest version in the v1.18 series: v1.19.0
To see the stack trace of this error execute with --v=5 or higher
root@ubuntu:~#
```

```
root@ubuntu:~# kubeadm upgrade apply v1.19.0
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -
oyaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade/version] You have chosen to change the cluster version to "v1.19.0"
[upgrade/versions] Cluster version: v1.18.6
[upgrade/versions] kubeadm version: v1.19.0
[upgrade/confirm] Are you sure you want to proceed with the upgrade? [y/N]: y
[upgrade/prepull] Pulling images required for setting up a Kubernetes cluster
[upgrade/prepull] This might take a minute or two, depending on the speed of your internet connection
[upgrade/prepull] You can also perform this action in beforehand using 'kubeadm config images pull'
[upgrade/apply] Upgrading your Static Pod-hosted control plane to version "v1.19.0"...
Static pod: kube-apiserver-ubuntu hash: b5d6694766bca803d442747b7dcd6d40
Static pod: kube-controller-manager-ubuntu hash: 630d401043c04cdc80df45b4767c6d4a
Static pod: kube-scheduler-ubuntu hash: 3dd66788a2c7782d910d05ea37b91678
[upgrade/etcd] Upgrading to TLS for etcd
Static pod: etcd-ubuntu hash: 49399c35c293900bd4ce20befa9e1b4a
[upgrade/staticpods] Preparing for "etcd" upgrade
[upgrade/staticpods] Renewing etcd-server certificate
[upgrade/staticpods] Renewing etcd-peer certificate
[upgrade/staticpods] Renewing etcd-healthcheck-client certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/etcd.yaml" and backed up old
manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2021-01-16-21-29-24/etcd.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap (timeout
5m0s)
Static pod: etcd-ubuntu hash: 49399c35c293900bd4ce20befa9e1b4a
Static pod: etcd-ubuntu hash: 49399c35c293900bd4ce20befa9e1b4a
Static pod: etcd-ubuntu hash: 47c97b1bdb8c694f9702bbfdbcd7e88e
```

```
[apiclient] Found 1 Pods for label selector component=etcd
[upgrade/staticpods] Component "etcd" upgraded successfully!
[upgrade/etcd] Waiting for etcd to become available
[upgrade/staticpods] Writing new Static Pod manifests to "/etc/kubernetes/tmp/kubeadm-upgraded-
manifests591288964"
[upgrade/staticpods] Preparing for "kube-apiserver" upgrade
[upgrade/staticpods] Renewing apiserver certificate
[upgrade/staticpods] Renewing apiserver-kubelet-client certificate
[upgrade/staticpods] Renewing front-proxy-client certificate
[upgrade/staticpods] Renewing apiserver-etcd-client certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/kube-apiserver.yaml" and backed
up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2021-01-16-21-29-24/kube-
apiserver.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap (timeout
5m0s)
. . .
Static pod: kube-apiserver-ubuntu hash: b5d6694766bca803d442747b7dcd6d40
[apiclient] Found 1 Pods for label selector component=kube-apiserver
[upgrade/staticpods] Component "kube-apiserver" upgraded successfully!
[upgrade/staticpods] Preparing for "kube-controller-manager" upgrade
[upgrade/staticpods] Renewing controller-manager.conf certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/kube-controller-manager.yaml"
and backed up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2021-01-16-21-29-
24/kube-controller-manager.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap (timeout
5m0s)
Static pod: kube-controller-manager-ubuntu hash: 630d401043c04cdc80df45b4767c6d4a
Static pod: kube-controller-manager-ubuntu hash: 69bf9b82cdbfd944a8f3c4ea7b92d2c5
[apiclient] Found 1 Pods for label selector component=kube-controller-manager
[upgrade/staticpods] Component "kube-controller-manager" upgraded successfully!
[upgrade/staticpods] Preparing for "kube-scheduler" upgrade
[upgrade/staticpods] Renewing scheduler.conf certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/kube-scheduler.yaml" and backed
up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2021-01-16-21-29-24/kube-
scheduler.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap (timeout
5m0s)
Static pod: kube-scheduler-ubuntu hash: 3dd66788a2c7782d910d05ea37b91678
Static pod: kube-scheduler-ubuntu hash: 23d2ea3ba1efa3e09e8932161a572387
[apiclient] Found 1 Pods for label selector component=kube-scheduler
[upgrade/staticpods] Component "kube-scheduler" upgraded successfully!
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system"
Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.19" in namespace kube-system with the configuration for
the kubelets in the cluster
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to
get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs
from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the
cluster

[addons] Applied essential addon: CoreDNS
```

```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy
```

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.19.0". Enjoy!

```
[upgrade/kubelet] Now that your control plane is upgraded, please proceed with upgrading your kubelets if
you haven't already done so.
root@ubuntu:~#




root@ubuntu:~# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.0",
GitCommit:"e19964183377d0ec2052d1f1fa930c4d7575bd50", GitTreeState:"clean", BuildDate:"2020-08-
26T14:28:32Z", GoVersion:"go1.15", Compiler:"gc", Platform:"linux/amd64"}
root@ubuntu:~#




root@ubuntu:~# kubectl uncordon ubuntu
node/ubuntu uncordoned
root@ubuntu:~#

root@ubuntu:~# kubectl get nodes
NAME     STATUS   ROLES    AGE    VERSION
ubuntu   Ready    master   176d   v1.18.6
wnode1   Ready    <none>   176d   v1.18.6
root@ubuntu:~#



root@ubuntu:~#
root@ubuntu:~# kubectl version
Client Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.18.6",
GitCommit:"dff82dc0de47299ab66c83c626e08b245ab19037", GitTreeState:"clean", BuildDate:"2020-07-
15T16:58:53Z", GoVersion:"go1.13.9", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.0",
GitCommit:"e19964183377d0ec2052d1f1fa930c4d7575bd50", GitTreeState:"clean", BuildDate:"2020-08-
26T14:23:04Z", GoVersion:"go1.15", Compiler:"gc", Platform:"linux/amd64"}
root@ubuntu:~#
root@ubuntu:~#
```

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

Q 4 : Create a snapeshote of the existing etcd instance running at https://127.0.0.1:2379, saving the snapshot to /srrv/data/etcd-snapshot.db
Next restore an existing, previous snapshot located at /var/lib/backup/etcd-snapshot-previous.db
CA cert: /op/KUNIN00601/ca.crt
Client crt /op/KUNIN00601/etcd-client.crt
Client key /op/KUNIN00601/etcd-client.key

```
controlplane $ ETCDCTL_API=3 etcdctl --endpoints=[127.0.0.1]:2379
controlplane $ ETCDCTL_API=3 etcdctl --endpoints=[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 /opt/etcd-backup.db
{"level":"info","ts":1611307611.630596,"caller":"snapshot/v3_snapshot.go:119","msg":"created temporary
db file","path":"/opt/etcd-backup.db.part"}
{"level":"info","ts":"2021-01-22T09:26:51.641Z","caller":"clientv3/maintenance.go:200","msg":"opened
snapshot stream; downloading"}
{"level":"info","ts":1611307611.6411822,"caller":"snapshot/v3_snapshot.go:127","msg":"fetching
snapshot","endpoint":"[127.0.0.1]:2379"}
{"level":"info","ts":"2021-01-22T09:26:51.707Z","caller":"clientv3/maintenance.go:208","msg":"completed
snapshot read; closing"}
{"level":"info","ts":1611307611.717937,"caller":"snapshot/v3_snapshot.go:142","msg":"fetched
snapshot","endpoint":"[127.0.0.1]:2379","size":"2.9 MB","took":0.08719177}
{"level":"info","ts":1611307611.7183523,"caller":"snapshot/v3_snapshot.go:152","msg":"saved","path":"/op
t/etcd-backup.db"}
Snapshot saved at /opt/etcd-backup.db
controlplane $

controlplane $ ETCDCTL_API=3 etcdctl --endpoints=[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 status -w table /opt/etcd-backup.db
+----------+----------+------------+------------+
|   HASH   | REVISION | TOTAL KEYS | TOTAL SIZE |
+----------+----------+------------+------------+
| d23872f5 |     3762 |       1253 |     2.9 MB |
+----------+----------+------------+------------+
controlplane $
```

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

Q 5 : Create a new network policy named allow-port-from-namespace that allows pods in the existing namespace my-app to connect to port 9000 of other pods in the same namespace.
Ensure that the new NetworkPolicy
. does not allow access to pods not listening on port 9000
. does not allow access from Pods not in namespace my-app

```
controlplane $ cat np.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: my-app
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          project: my-app
    ports:
    - protocol: TCP
      port: 9000
      port: 9000
controlplane $


controlplane $ kubectl create -f np.yaml
networkpolicy.networking.k8s.io/allow-port-from-namespace created
controlplane $

controlplane $ kubectl describe networkpolicies.networking.k8s.io -n my-app
Name:         allow-port-from-namespace
Namespace:    my-app
Created on:   2021-01-22 20:33:03 +0000 UTC
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    To Port: 9000/TCP
    From:
      NamespaceSelector: project=my-app
  Not affecting egress traffic
  Policy Types: Ingress
controlplane $
```

===

Q 6 : Reconfigure the existing deployment front-end and add a port specification named http exposing port 80/tcp of the existing container nginx
create a new service named front-end-svc exposing the container port http.
Configure the service also expose the individual pods via a NodePort on the nodes on which they are scheduled.

```
controlplane $ kubectl create deployment front-end --image=nginx --port=80 -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: front-end
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: front-end
  template:
    metadata:
      labels:
        app: front-end
    spec:
      containers:
      - image: nginx
        name: nginx
        ports:

        - containerPort: 80
        containerPort: 80
```

```
controlplane $ kubectl expose deployment front-end --type=NodePort --name=front-end-service --port=80 -
-protocol=TCP -o yaml
apiVersion: v1
kind: Service
metadata:
    labels:
    app: front-end
  name: front-end-service
spec:
  ports:
  - nodePort: 31740
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: front-end
  type: NodePort
controlplane $
```

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

Q 7 : Create a new nginx Ingress resource as follows:
  .Name: pong
  .Namespace: in-internal
  .Exposing service service hello on path /hello using service port 5678
  Availablity of sevice can be checked by curl -kL <IN Terminal_IP>/hello

```
controlplane $ kubectl create ns in-internal
namespace/in-internal created

controlplane $ cat ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: pong
  namespace: in-internal
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /hello
        pathType: Prefix
        backend:
          service:
            name: hello

            port:
```

```
            number: 5678
controlplane $

controlplane $ kubectl create -f ingress.yaml
ingress.networking.k8s.io/pong created
controlplane $



controlplane $ kubectl describe ingress pong -n in-internal
Name:             pong
Namespace:        in-internal
Address:
Default backend:  default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /hello   hello:5678 (10.32.0.3:5678)
Annotations:  nginx.ingress.kubernetes.io/rewrite-target: /
Events:       <none>
controlplane $
```

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

Q 8 : Scale the deployment presentaion to 6 pods.

```
controlplane $ kubectl scale --replicas=6 deployment presentation
deployment.apps/presentation scaled
controlplane $
```

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

Q 9 : Schedule a pod as follows:
. Name: nginx-kusc00401
. Image: nginx
. Node Selector: disktype=ssd

```
controlplane $ cat node.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-kusc00401
  labels:
    env: test
spec:
  containers:
  - name: nginx-kusc00401
    image: nginx
  nodeSelector:

    disktype: ssd
    disktype: ssd
controlplane $


controlplane $ kubectl create -f node.yaml
pod/nginx-kusc00401 created
controlplane $



controlplane $ kubectl describe pod nginx-kusc00401
Name:         nginx-kusc00401
Namespace:    default
Priority:     0
Node:         node01/172.17.0.38
Start Time:   Fri, 22 Jan 2021 18:07:47 +0000
Labels:       env=test
Annotations:  <none>
Status:       Running
IP:           10.244.1.4
IPs:
  IP:  10.244.1.4
Containers:
  nginx-kusc00401:
    Container ID:   docker://12f8aecfdb5296b9429cfce6140f458c8860c2928ca4dd6e4783816fbee8ece6
    Image:          nginx
    Image ID:       docker-
pullable://nginx@sha256:10b8cc432d56da8b61b070f4c7d2543a9ed17c2b23010b43af434fd40e2ca4aa
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 22 Jan 2021 18:07:49 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9xp62 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-9xp62:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-9xp62
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  disktype=ssd
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  15s   default-scheduler  Successfully assigned default/nginx-kusc00401 to node01

  Normal  Pulling    14s   kubelet, node01    Pulling image "nginx"
  Normal  Pulled     13s   kubelet, node01    Successfully pulled image "nginx" in 549.619738ms
  Normal  Created    13s   kubelet, node01    Created container nginx-kusc00401
  Normal  Started    13s   kubelet, node01    Started container nginx-kusc00401
controlplane $
```

===

Q 10 : Check to see how many nodes are ready (not including nodes tainted NoSchedule) and write the number to
/opt/KUSC00402/kusc00402.txt

```
kubectl get nodes
echo "2" > /opt/KUSC00402/kusc00402.txt
```

===

Q 11 : Create a pod named kucc4 with single app container for each of the following images running inside (there may be between q and 4 images specified):
nginx + redis + memcached + consul

```
controlplane $ cat pp.yaml
apiVersion: v1
kind: Pod
metadata:
  name: kucc4
spec:
  containers:
  - image: nginx
    name: nginx
  - image: redis
    name: redis
  - image: memcached
    name: memcached
  - image: consul
    name: consul
controlplane $
controlplane $ kubectl create -f pp.yaml
pod/kucc4 created
controlplane $

controlplane $ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
kucc4          4/4     Running   0          28s
controlplane $


controlplane $ kubectl describe pod kucc4
Name:         kucc4
Namespace:    default
Priority:     0
```

```
Node:         node01/172.17.0.38
Start Time:   Fri, 22 Jan 2021 18:20:03 +0000
Labels:       <none>
Annotations:  <none>
Status:       Pending
IP:
IPs:          <none>
Containers:
  nginx:
    Container ID:
    Image:          nginx
    Image ID:
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9xp62 (ro)
  redis:
    Container ID:
    Image:          redis
    Image ID:
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9xp62 (ro)
  memcached:
    Container ID:
    Image:          memcached
    Image ID:
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9xp62 (ro)
  consul:
    Container ID:
    Image:          consul
    Image ID:
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       ContainerCreating

    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-9xp62 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-9xp62:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-9xp62
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  13s   default-scheduler  Successfully assigned default/kucc4 to node01
  Normal  Pulling    12s   kubelet, node01    Pulling image "nginx"
  Normal  Pulled     12s   kubelet, node01    Successfully pulled image "nginx" in 662.539303ms
  Normal  Created    11s   kubelet, node01    Created container nginx
  Normal  Started    11s   kubelet, node01    Started container nginx
  Normal  Pulling    11s   kubelet, node01    Pulling image "redis"
  Normal  Pulled     9s    kubelet, node01    Successfully pulled image "redis" in 2.695195708s
  Normal  Pulling    8s    kubelet, node01    Pulling image "memcached"
  Normal  Created    8s    kubelet, node01    Created container redis
  Normal  Started    8s    kubelet, node01    Started container redis
  Normal  Pulled     5s    kubelet, node01    Successfully pulled image "memcached" in 2.707260529s
  Normal  Created    5s    kubelet, node01    Created container memcached
  Normal  Started    5s    kubelet, node01    Started container memcached
  Normal  Pulling    5s    kubelet, node01    Pulling image "consul"
  Normal  Pulled     1s    kubelet, node01    Successfully pulled image "consul" in 4.224204168s
  Normal  Created    0s    kubelet, node01    Created container consul
  Normal  Started    0s    kubelet, node01    Started container consul
controlplane $
```

==

Q 12 : Create a persistent volume with name app-data, of capacity 1Gi and access mode ReadWriteMany, The type of volume is hostPath and its location is /srv/app-data

```
controlplane $ cat pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-data

  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/srv/app-data"
controlplane $


controlplane $ kubectl create -f pv.yaml
persistentvolume/app-config created
controlplane $


controlplane $ kubectl get pv
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS
REASON   AGE
app-data   1Gi        RWX            Retain           Available           manual                5s
controlplane $
```

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

Q 13 : Create a new PersistentVolumeClaim:
. Name: pv-volume.
. Class: csi-hostpath-sc
. Capacity: 10Mi
Second create pod and claim the space from PV

```
controlplane $
controlplane $ kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS
REASON   AGE
pv-1   10Gi       RWX            Retain           Available           csi-hostpath-sc           9s
controlplane $


controlplane $
controlplane $ cat pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
  labels:
    type: local
spec:
  storageClassName: csi-hostpath-sc
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Mi
controlplane $


controlplane $ kubectl create -f pvc.yaml
persistentvolumeclaim/pv-volume created
controlplane $



controlplane $ kubectl get pvc
NAME        STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS      AGE
pv-volume   Bound    pv-1     10Gi       RWX            csi-hostpath-sc   5s
controlplane $



controlplane $ cat pvpod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pv-claim-pod
spec:
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: pvc-claim1
  volumes:
    - name: pvc-claim1
      persistentVolumeClaim:
        claimName: pv-volume
controlplane $
controlplane $


controlplane $ kubectl create -f pvpod.yaml
pod/pv-claim-pod created
controlplane $

controlplane $ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
pv-claim-pod   1/1     Running   0          5s
controlplane $
```

==

Q 14 : Monitor the logs of pod foo and:

. Extract log lines corrosponding to error " unable-to-access-website"

. Write them to /opt/KUTR00101/f00

Example:

```
controlplane $ kubectl logs foo | grep -i "looking for shell" > /opt/KUTR00101/f00

controlplane $ cat /opt/KUTR00101/f00
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
controlplane $
```

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

Q 15 : Without changing its existing containers, an existing Pod needs to be integrated into Kubernetes's built-in logging architecture (eg: kubectl logs). Adding a steaming sidecar container is a good and common way to accomplish this requirment.

Task: Add a busybox sidecar container to the existing Pod 11-factor-app. The new sidecar container had to run the following command :
/bin/sh -c tail -n+1 /var/log/11-factor-app.log
Use a volume mountnamed logs to make the file /var/log/11-factor-app.log available to the sidecar container.
Don't modify the existing container , Don't modify the path of the log file, both container must be access it at containers must access it at /var/log/110factor-app.log.

Ans:

Not tried yet.

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

Q 16 : From the pod lable name=cpu-utilizer, find pods running high CPU workloads and write the name of the pod consuming most to the file /opt/KUTR00401/KUTROO401.txt (Which is already exist)

```
$ kubectl top pods -l name1=cpu-utilizer
NAME                              CPU(cores)   MEMORY(bytes)
mysql-7d446bbff5-m4774            0m           598Mi
mysqld-exporter-77f4d65d75-dg6q7   0m           2Mi
nginx1                            0m           2Mi
$

$ kubectl top pods -l name=cpu-utilizer --sort-by=cpu --no-headers | cut -f1 -d" " | head -n1
mysqld-exporter-77f4d65d75-dg6q7
$

$ kubectl top pods -l name=cpu-utilizer --sort-by=cpu --no-headers | cut -f1 -d" " | head -n1 >
opt_KUTR00401_KUTROO401.txt
$

$ cat opt_KUTR00401_KUTROO401.txt

mysqld-exporter-77f4d65d75-dg6q7
```

$

==

Q 17 : A kubernetes worker node, named wk8s-node-0 is in state NotReady. Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state, ensureing that any changes are made permanent.
You can assume elevated privileages sudo -i

```
$ ssh to node01
$ systemctl enable kubelet    (or)    $systemctl enable --now kubelet
$ systemctl restart kubelet
# systemctl status kubelet
```

==