CKA exam testing

There will be 24 questions, 3 hours, 6 clusters, 10 topics.

Tip: Create an alias for all kubelet commands e.g:
alias kg='kubectl get'
alias kc='kubectl create -f'

Preparation

Q: Create a Job that run 60 time with 2 jobs running in parallel

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  completions: 60
  parallelism: 2
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
```

11. Create a job that runs every 3 minutes and prints out the current time.
https://kubernetes.io/docs/tasks/job/automated-tasks-with-cron-jobs/
Minutes, hours, Day of the month, Month, Day of the week
12. Create a job that runs 20 times, 5 containers at a time, and prints "Hello parallel world"

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox1
spec:
  completions: 20
  parallelism: 5
  template:
    spec:
      containers:
      - name: busybox1
        image: busybox
        command: ["/bin/sh", "-c", "echo Hello Parallel world"]
      restartPolicy: Never
```

root@kube-01:~/CKA# k logs job/busybox1
Found 20 pods, using pod/busybox1-g4j5v
Hello Parallel world
root@kube-01:~/CKA#

https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/

Q: Find which Pod is taking max CPU
When we run >kubectl top pods or k top nodes we will get below error.
root@kub-master:/home/gvemala/practice# kubectl top pod | awk '{print $1}'
Error from server (NotFound): the server could not find the requested resource (get services **http:heapster**:) Note: Heapster was used for metrics for kubernetes version <1.6

https://github.com/kubernetes-incubator/metrics-server (Metrics server package has to be installed on the master to get output for top command.)
root@kub-master:/home/gvemala/practice# git clone https://github.com/kubernetes-incubator/metrics-server.git
Cloning into 'metrics-server'...

root@kub-master:/home/gvemala/practice/metrics-server# k apply -f deploy/1.8+/
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.extensions/metrics-server created
service/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created

root@kub-master:/home/gvemala/practice/metrics-server# wget -c https://gist.githubusercontent.com/initcron/1a2bd25353e1faa22a0ad41ad1c01b62/raw/008e23f9fbf4d7e2cf79df1dd008de2f1db62a10/k8s-metrics-server.patch.yaml
--2019-03-29 10:32:43--
https://gist.githubusercontent.com/initcron/1a2bd25353e1faa22a0ad41ad1c01b62/raw/008e23f9fbf4d7e2cf79df1dd008de2f1db62a10/k8s-metrics-server.patch.yaml

root@kub-master:/home/gvemala/practice# kubectl patch deploy metrics-server -p "$(cat k8s-metrics-server.patch.yaml)" -n kube-system
deployment.extensions/metrics-server patched

root@kub-master:/home/gvemala/practice# k get deployments -n kube-system

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
coredns	2/2	2	2	10d
metrics-server	1/1	1	1	3m25s

Use `kubectl top` to find CPU usage per pod

```
Nodes:
kubectl top nodes | awk '{print $1 "\t" $3}' | sort -r -n -k2 | head -1
```

root@kub-master:/home/gvemala/practice# k top pods | awk '{print $1 "\t" $3}' | sort -r -n -k2 | head -1

```
Pods:
kubectl top pods | awk '{print $1 " " $2}' | sort -r -n -k2 | head -
```

Q: List all PersistentVolumes sorted by their name

Use `kubectl get pv --sort-by=` <- this problem is buggy & also by default kubectl give the output sorted by name.

```
kcs get svc --sort-by=.metadata.name
```

```
# Get commands with basic output
$ kubectl get services                          # List all services in the namespace
$ kubectl get pods --all-namespaces             # List all pods in all namespaces
$ kubectl get pods -o wide                      # List all pods in the namespace,
with more details
$ kubectl get deployment my-dep                 # List a particular deployment
$ kubectl get pods --include-uninitialized      # List all pods in the namespace,
including uninitialized ones

# Describe commands with verbose output
$ kubectl describe nodes my-node
$ kubectl describe pods my-pod

$ kubectl get services --sort-by=.metadata.name # List Services Sorted by Name

# List pods Sorted by Restart Count
$ kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

# Get the version label of all pods with label app=cassandra
$ kubectl get pods --selector=app=cassandra rc -o \
    jsonpath='{.items[*].metadata.labels.version}'
kubectl get pods -l app=cassandra-cassandra -o jsonp

# Get ExternalIPs of all nodes
$ kubectl get nodes -o
jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP")].address}'

# List Names of Pods that belong to Particular RC
# "jq" command useful for transformations that are too complex for jsonpath, it can
be found at https://stedolan.github.io/jq/
$ sel=${$(kubectl get rc my-rc --output=json | jq -j '.spec.selector | to_entries |
.[] | "\(.key)=\(.value),"')%?}
$ echo $(kubectl get pods --selector=$sel --output=jsonpath={.items..metadata.name})
```

```
# Check which nodes are ready
$ JSONPATH='{range .items[*]}{@.metadata.name}:{range
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \
    && kubectl get nodes -o jsonpath="$JSONPATH" | grep "Ready=True"

# List all Secrets currently in use by a pod
$ kubectl get pods -o json | jq
'.items[].spec.containers[].env[]?.valueFrom.secretKeyRef.name' | grep -v null | sort
| uniq
```

Q: Create a NetworkPolicy to allow connect to port 8080 by busybox pod only
Make sure to use `apiVersion: extensions/v1beta1` which works on both 1.6 and 1.7
https://kubernetes.io/docs/tasks/administer-cluster/declare-network-policy/
kubectl run busybox --image=busybox --replicas=2

```
kubectl expose deployment busybox --port=80 --target-port=8080
k get svc,pod
kubectl run busybox1 --rm -ti --image=busybox /bin/sh
wget --spider --timeout=1 busybox

kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: access-nginx
spec:
  podSelector:
    matchLabels:
      run: busybox
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
```

k apply -f np.yaml
kubectl run busybox1 --rm -ti --image=busybox /bin/sh

```
wget --spider --timeout=1 busybox:8080
```

Q: fixing broken nodes, see
https://kubernetes.io/docs/concepts/architecture/nodes/

Conditions:
Type Status LastHeartbeatTime LastTransitionTime Reason
Message

login to the one of the worker node & stop the kubelet service . the worker node went not ready.
root@kube-02:~# systemctl stop kubelet

MemoryPressure	Unknown	Wed, 20 Feb 2019 07:24:24 +0000	Wed, 20 Feb 2019 07:25:08 +0000	NodeStatusUnknown	Kubelet stopped posting node status.
DiskPressure	Unknown	Wed, 20 Feb 2019 07:24:24 +0000	Wed, 20 Feb 2019 07:25:08 +0000	NodeStatusUnknown	Kubelet stopped posting node status.
PIDPressure	Unknown	Wed, 20 Feb 2019 07:24:24 +0000	Wed, 20 Feb 2019 07:25:08 +0000	NodeStatusUnknown	Kubelet stopped posting node status.
Ready	Unknown	Wed, 20 Feb 2019 07:24:24 +0000	Wed, 20 Feb 2019 07:25:08 +0000	NodeStatusUnknown	Kubelet stopped posting node status.
OutOfDisk	Unknown	Fri, 15 Feb 2019 18:21:55 +0000	Wed, 20 Feb 2019 07:25:08 +0000	NodeStatusNeverUpdated	Kubelet never posted node status.

After restarting the kubelet service in the worker node, so node not ready issue got fixed. See the below output.

Conditions:

Type	Status	LastHeartbeatTime	LastTransitionTime	Reason	Message
MemoryPressure	False	Wed, 20 Feb 2019 07:27:59 +0000	Wed, 20 Feb 2019 07:26:39 +0000	KubeletHasSufficientMemory	kubelet has sufficient memory available
DiskPressure	False	Wed, 20 Feb 2019 07:27:59 +0000	Wed, 20 Feb 2019 07:26:39 +0000	KubeletHasNoDiskPressure	kubelet has no disk pressure
PIDPressure	False	Wed, 20 Feb 2019 07:27:59 +0000	Wed, 20 Feb 2019 07:26:39 +0000	KubeletHasSufficientPID	kubelet has sufficient PID available
Ready	True	Wed, 20 Feb 2019 07:27:59 +0000	Wed, 20 Feb 2019 07:26:39 +0000	KubeletReady	kubelet is posting ready status. AppArmor enabled
OutOfDisk	Unknown	Fri, 15 Feb 2019 18:21:55 +0000	Wed, 20 Feb 2019 07:25:08 +0000	NodeStatusNeverUpdated	Kubelet never posted node status.

Node not ready statuses: When we describe a node, if we see all false in the status then its kubelet is not running. All are unknown means, docker may be done, restart docker.

Q: etcd backup, see
https://github.com/kelseyhightower/kubernetes-the-hard-way
go to bootstrapping ETCD cluster
go to end, copy paste the command and pass snapshot save, replace the cert paths given in the exam and direct it to the file that he mentioned, after running check the size of the backup file.
https://kubernetes.io/docs/getting-started-guides/ubuntu/installation/ juju cluster
https://kubernetes.io/docs/getting-started-guides/ubuntu/backups/

https://www.mirantis.com/blog/everything-you-ever-wanted-to-know-about-using-etcd-with-kubernetes-v1-6-but-were-afraid-to-ask/

Q: TLS bootstrapping, see
https://coreos.com/kubernetes/docs/latest/openssl.html
https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/
https://github.com/cloudflare/cfssl

Q: You have a Container with a volume mount. Add a init container that creates an empty file in the volume. (only trick is to mount the volume to init-container as well)
Init container – configure pod initialization

```
apiVersion: v1
kind: Pod
metadata:
  name: init-demo
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - name: workdir
      mountPath: /workdir
  # These containers are run during pod initialization
  initContainers:
  - name: install
    image: busybox
    command: ['/bin/sh', '-c', 'touch /workdir/a.txt']
    volumeMounts:
    - name: workdir
      mountPath: "/work-dir"
  dnsPolicy: Default
  volumes:
  - name: workdir
    emptyDir: {}
```

https://kubernetes.io/docs/concepts/workloads/pods/init-containers/
https://kubernetes.io/docs/concepts/storage/volumes/#hostpath

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
```

```
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      # directory location on host
      path: /data
      # this field is optional
      type: DirectoryOrCreate
```

```
root@ubuntu-s-2vcpu-2gb-blr1-01:~# k config set-context $(k config current-context) --
namespace=nagesh
Context "kubernetes-admin@kubernetes" modified.
root@ubuntu-s-2vcpu-2gb-blr1-01:~#
root@ubuntu-s-2vcpu-2gb-blr1-01:~# k config current-context
kubernetes-admin@kubernetes
root@ubuntu-s-2vcpu-2gb-blr1-01:~# k config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin   nagesh
root@ubuntu-s-2vcpu-2gb-blr1-01:~#
```

https://kubernetes.io/docs/concepts/storage/volumes/

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  initContainers:
  - name: init-touch-file
    image: busybox
    volumeMounts:
    - mountPath: /data
      name: cache-volume
    command: ['sh', '-c', 'echo "" > /data/harshal.txt']
  volumes:
  - name: cache-volume
    emptyDir: {}
````

Q: When running a redis key-value store in your pre-production environments many deployments are incoming from CI and leaving behind a lot of stale cache data in redis which is causing test failures. The CI admin has requested that each time a redis key-value-store is deployed in staging that it not persist its data.

Create a pod named non-persistent-redis that specifies a named-volume with name app-cache, and mount path /data/redis. It should launch in the staging namespace and the volume MUST NOT be persistent.

Persistent means – Host path or nfs anything, which holds the data even the pod is deleted.
Non-persistent – empty dir, which should get deleted when the pod gets deleted

```
apiVersion: v1
kind: Pod
metadata:
  name: non-persistent-redis
  namespace: staging
spec:
  containers:
  - image: nginx
    name: non-persistent-redis
    volumeMounts:
    - mountPath: /data/redis
      name: app-cache
  volumes:
  - name: app-cache
    hostPath:
      # directory location on host
      path: /data/redis
      # this field is optional
      type: DirectoryOrCreate
```

Create a Pod with EmptyDir and in the YAML file add namespace: CI

```
apiVersion: v1
kind: Pod
metadata:
  name: non-persistent-redis
  namespace: ci
spec:
  containers:
  - image: nginx
    name: non-persistent-redis
    volumeMounts:
```

```
    - mountPath: /data1/redis1
      name: app-cache
  volumes:
  - name: app-cache
    emptyDir: {}

apiVersion: v1
kind: Pod
metadata:
name: non-persistent-redis
namespace: staging
spec:
containers:
- image: k8s.gcr.io/test-webserver
name: test-container
volumeMounts:
- mountPath: /data/redis
name: app-cache
volumes:
- name: app-cache
emptyDir: {}
```

Q: Setting up K8s master components with a binaries/from tar balls:

Also, convert CRT to PEM: openssl x509 -in abc.crt -out abc.pem
- https://coreos.com/kubernetes/docs/latest/openssl.html
- https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/04-certificate-authority.md
- https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/08-bootstrapping-kubernetes-controllers.md
- https://gist.github.com/mhausenblas/0e09c448517669ef5ece157fd4a5dc4b
- https://kubernetes.io/docs/getting-started-guides/scratch/
- http://alexander.holbreich.org/kubernetes-on-ubuntu/ maybe dashboard?
- https://kubernetes.io/docs/getting-started-guides/binary_release/
- http://kamalmarhubi.com/blog/2015/09/06/kubernetes-from-the-ground-up-the-api-server/

Q: Find the error message with the string “Some-error message here”.
https://kubernetes.io/docs/concepts/cluster-administration/logging/ see kubectl logs and /var/log for system services

Q 17: Create an Ingress resource, Ingress controller and a Service that resolves to cs.rocks.ch.

First, create controller and default backend
```
kubectl apply -f https://githib.com /kubernetes/ingress/master/controllers/nginx/examples/default-backend.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress/master/examples/deployment/nginx/nginx-ingress-controller.yaml
```

Second, create service and expose
```
kubectl run ingress-pod --image=nginx --port 80
kubectl expose deployment ingress-pod --port=80 --target-port=80 --type=NodePort
```

Create the ingress
```
cat <<EOF >ingress-cka.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-service
spec:
  rules:
  - host: "cs.rocks.ch"
    http:
      paths:
      - backend:
          serviceName: ingress-pod
          servicePort: 80
EOF
```

To test, run a curl pod
```
kubectl run -i --tty client --image=tutum/curl
curl -I -L --resolve cs.rocks.ch:80:10.240.0.5 http://cs.rocks.ch/
```

Q: Run a Jenkins Pod on a specified node only. Static-pod
https://kubernetes.io/docs/tasks/administer-cluster/static-pod/
A) Apply a label for the node
   Take a pod spec, add nodeselector (the label created on the hostname)

root@kub-master:/home/gvemala/practice# k label no kub-node1 host=good
node/kub-node1 labeled

root@kub-master:/home/gvemala/practice# cat nodeselector.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    host: good
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    labels:
    host: good
```

Sort commands has to be drilled like this
A – Api Version
K - Kind
M - Metadata
S – Spec

Command to identify the masternode,
gvemala@kdk:~/K8s/PD_Deploy/platform-deploy/clusters/honjo4-stg.csco.cloud$ k cluster-info
Kubernetes master is running at https://64.102.181.99:6443
CoreDNS is running at https://64.102.181.99:6443/api/v1/namespaces/kube-system/services/coredns:dns/proxy

Or just use -0 wide as well for k get no command.

Static pod:
Create the Pod manifest at the specified location and then edit the systemd service file for kubelet(/etc/kubernetes/manifests) to include `--pod-manifest-path=/specified/path`. Once done restart the service.

https://kubernetes.io/docs/tasks/administer-cluster/static-pod/
1. root@kub-node1:/home/gvemala# systemctl status kubelet
In this command, we can check the path of 10-kubeadm.conf file. Add the path to env variable.
root@kub-node1:/etc/systemd/system/kubelet.service.d# pwd
/etc/systemd/system/kubelet.service.d

root@kub-node1:/etc/systemd/system/kubelet.service.d# cat 10-kubeadm.conf
Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --pod-manifest-path=/etc/kubelet.d/"

root@kube-02:/etc/kubelet.d# vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
root@kube-02:/etc/kubelet.d#
root@kube-02:/etc/kubelet.d#
root@kube-02:/etc/kubelet.d# systemctl daemon-reload
root@kube-02:/etc/kubelet.d# systemctl restart kubelet
root@kube-02:/etc/kubelet.d#

Q: Use the utility nslookup to look up the DNS records of the service and pod.
From this guide, https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/
Look for “Quick Diagnosis”
https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/

A) Create a POD & Svc, later create a busy box pod which will have nslookup utility, login to the busybox pod using exec and run nslookup. In kubernetes while running nslookup, we have the use ‘–‘ instead of ‘.’ In between numbers of IPaddress
B) **0-0-0-0.ns.kind.cluster.local**

nslookup 10-244-1-61.default.pod.cluster.local.
Server: 10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: 10-244-1-61.default.pod.cluster.local
Address 1: 10.244.1.61

nslookup my-service.default.svc.cluster.local
Server: 10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: my-service.default.svc.cluster.local
Address 1: 10.110.253.250 my-service.default.svc.cluster.local

$ kubectl exec -ti busybox -- nslookup mysvc.myns.svc.cluster.local
Naming conventions for services and pods:
For a regular service, this resolves to the port number and the CNAME: my-svc.my-namespace.svc.cluster.local.

For a headless service, this resolves to multiple answers, one for each pod that is backing the service, and contains the port number and a CNAME of the pod of the form auto-generated-name.my-svc.my-namespace.svc.cluster.local
When enabled, pods are assigned a DNS A record in the form of pod-ip-address.my-namespace.pod.cluster.local.
For example, a pod with IP 1.2.3.4 in the namespace default with a DNS name of cluster.local would have an entry: 1-2-3-4.default.pod.cluster.local

Q: Start a pod automatically by keeping manifest in /etc/kubernetes/manifests
Refer to https://kubernetes.io/docs/tasks/administer-cluster/static-pod/
Edit kubelet.service on any worker node to contain this flag --pod-manifest-path=/etc/kubernetes/manifests then place the pod manifest at /etc/kubernetes/manifests.
Now restart kubelet.

Some other Questions:

1. Main container looks for a file and crashes if it doesnt find the file. Write an init container to create the file and make it available for the main container
2. Install and Configure kubelet on a node to run pod on that node without contacting the api server (static pod)
3. Take backup of etcd cluster

4. rotate TLS certificates
5.rolebinding
6.Troubleshooting - involved identifying failing nodes, pods , services and identifying cpu utilization of pods.

Juju cluster:

https://docs.jujucharms.com/2.3/en/reference-install

https://kubernetes.io/docs/getting-started-guides/ubuntu/installation/

Question: 16 For network policies to work in Kubernetes, which of these must be true?

Choose the correct answer:

A. The CNI must have a "policy" sidebar.

B. The CNI must support VxLANs.

C. Network policies are always enforced.

D. The CNI must enforce the network policies. Answer: D

Explanation: If the CNI doesn't support network policies, then applying a YAML formula with a network policy in it will return a success, but the policies will not be enforced.

Question: 15 What controls a Kubernetes cluster? Choose the correct answer: A. minikube B. The Master C. kube-proxy D. kubelet Answer: B Explanation: The master node contains the Kubernetes api server, which controls what the cluster does.

Static Pod.

```
 62  cd /etc/kubernetes/
 63  cd manifests/
 64  ls
 65  vim pod.yaml
 66  pwd
 67  cd /etc/systemd/system/
 68  ls
 69  cd kubelet.service.d/
 70  ls
 71  vim 10-kubeadm.conf
 72  cd ~
 73  systemctl restart kubelte
 74  systemctl restart kubelet
 75  systemctl daemon-reload
 76  systemctl restart kubelet
 77  systemctl enable kubelet
 78  cd -
 79  vim 10-kubeadm.conf
 80  cd ../../../
 81  cd kubernetes/manifests/
 82  ls
 83  vim pod.yaml
```

Lets scaleup the stateful set application.

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

```
root@kub-master:/home/gvemala/practice# k get sts
NAME   READY   AGE
web    0/3     55s


root@kub-master:/home/gvemala/practice# k get sts
NAME   READY   AGE
web    3/3     9m

root@kub-master:/home/gvemala/practice# k get pod -o wide
NAME                  READY   STATUS     RESTARTS   AGE     IP           NODE
NOMINATED NODE   READINESS GATES
web-0                 1/1     Running    0          9m18s   10.244.1.62   kub-node1   <none>
<none>
web-1                 1/1     Running    0          9m9s    10.244.1.63   kub-node1   <none>
<none>
web-2                 1/1     Running    0          9m7s    10.244.1.64   kub-node1   <none>
<none>
```

kubectl scale --replicas=4 statefulset mysql

```
$ kubectl get statefulset
NAME      DESIRED   CURRENT   AGE
mysql     3         3         1m

root@kub-master:/home/gvemala/practice# k apply -f statefulset.yaml
statefulset.apps/web configured
root@kub-master:/home/gvemala/practice# k get sts
NAME   READY   AGE
web    4/4     10m

root@kub-master:/home/gvemala/practice#
root@kub-master:/home/gvemala/practice# k edit sts web -o yaml
apiVersion: apps/v1
kind: StatefulSet
```

root@kub-master:/home/gvemala/practice# **k get sts**
NAME READY AGE
web 6/6 11m

https://github.com/kelseyhightower/kubernetes-the-hard-way

Try the following exercises interactively:

Note - there are no answers here on purpose. You should be able to do these yourself using the minimal docs that you are allowed to use during the test. At a minimum this should train you on where to look for this info during the test, without notes.

1. Create a node that has a SSD and label it as such.

https://kubernetes.io/docs/concepts/configuration/assign-pod-node/

```
kubectl label nodes <your-node-name> disktype=ssd
```

root@kube-01:~# k label nodes kube-02 disktype=ssd

node/kube-02 labeled

root@kube-01:~#

root@kube-01:~# kubectl get nodes --show-labels

NAME STATUS ROLES AGE VERSION LABELS

kube-01 Ready master 2d4h v1.13.2 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/hostname=kube-01,node-role.kubernetes.io/master=

kube-02 Ready <none> 2d4h v1.13.2 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disktype=ssd,kubernetes.io/hostname=kube-02

kube-03 Ready <none> 2d4h v1.13.3 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/hostname=kube-03

root@kube-01:~#

a. Create a pod that is only scheduled on SSD nodes.
https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/#create-a-pod-that-gets-scheduled-to-your-chosen-node

root@kube-01:~/practice# k get po -o wide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
nginx 1/1 Running 0 39s 10.244.1.30 kube-02 <none> <none>

root@kube-01:~/practice#

2. Create 2 pod definitions: the second pod should be scheduled to run anywhere the first pod is running - 2nd pod runs alongside the first pod.

The **pod affinity rule** says that the **pod** can be scheduled onto a node only if that node is in the same zone as at least one already-running **pod** that has a label with key "security" and value "S1".

3. Create a deployment running nginx version 1.12.2 that will run in 2 pods
https://kubernetes.io/docs/tasks/run-application/run-stateless-application-deployment/

a. Scale this to 4 pods.
b. Scale it back to 2 pods.
c. Upgrade this to 1.13.8
d. Check the status of the upgrade
e. How do you do this in a way that you can see history of what happened?
https://kubernetes.io/docs/reference/ge nerated/kubectl/kubectl-commands#rollout

root@kube-01:~/practice# kubectl rollout history deployment/nginx-deployment
deployment.extensions/nginx-deployment
REVISION CHANGE-CAUSE
1 <none>
2 <none>

f. Undo the upgrade
root@kube-01:~/practice# kubectl rollout undo deployment/nginx-deployment
deployment.extensions/nginx-deployment rolled back
root@kube-01:~/practice#

4. Create a service that uses pod called scratch disk.
https://kubernetes.io/docs/concepts/storage/volumes/#emptydir

a. Change the service to mount a disk from the host.
https://kubernetes.io/docs/concepts/storage/volumes/#hostpath

b. Change the service to mount a persistent volume. (host-path or PVC)

5. Create a pod that has a liveness check
https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/

6. Create a service that manually requires endpoint creation - and create that too
https://kubernetes.io/docs/concepts/services-networking/service/

7. Create a daemon set

a. Change the update strategy to do a rolling update but delaying 30 seconds between pod updates

8. Create a static pod

9. Create a busybox container without a manifest. Then edit the manifest.

K run busybox –image=busybox –restart=Never

```
root@kub-master:/home/gvemala/practice# k run nginx-1 --image=nginx --restart=Never
root@kub-master:/home/gvemala/practice# k get po nginx-1 -o yaml
```

10. Create a pod that uses secrets

https://kubernetes.io/docs/concepts/configuration/secret/

```
root@kub-master:/home/gvemala/practice# k exec -it mypod -- cat /etc/foo/my-group/my-username
gvemalaroot@kub-master:/home/gvemala/practice#
```

a. Pull secrets from environment variables

b. Pull secrets from a volume

https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/

c. Dump the secrets out via kubectl to show it worked

k get secrets mysecret -o yaml

11. Create a job that runs every 3 minutes and prints out the current time.

12. Create a job that runs 20 times, 5 containers at a time, and prints "Hello parallel world"

13. Create a service that uses an external load balancer and points to a 3 pod cluster running nginx.

https://kubernetes.io/docs/tasks/access-application-cluster/connecting-frontend-backend/

14. Create a horizontal autoscaling group that starts with 2 pods and scales when CPU usage is over 50%.

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/

15. Create a custom resource definition

a. Display it in the API with curl

16. Create a networking policy such that only pods with the label access=granted can talk to it.

https://kubernetes.io/docs/tasks/administer-cluster/declare-network-policy/

https://kubernetes.io/blog/2017/10/enforcing-network-policies-in-kubernetes/

a. Create an nginx pod and attach this policy to it.

b. Create a busybox pod and attempt to talk to nginx - should be blocked

c. Attach the label to busybox and try again - should be allowed

17. Create a service that references an externalname.
https://kubernetes.io/docs/concepts/services-networking/service/#externalname
a. Test that this works from another pod
18. Create a pod that runs all processes as user 1000.
https://kubernetes.io/docs/tasks/configure-pod-container/security-context/
19. Create a namespace
https://kubernetes.io/docs/tasks/administer-cluster/manage-resources/quota-memory-cpu-namespace/
a. Run a pod in the new namespace
b. Put memory limits on the namespace
https://kubernetes.io/docs/tasks/administer-cluster/manage-resources/memory-default-namespace/
c. Limit pods to 2 persistent volumes in this namespace
https://kubernetes.io/docs/tasks/administer-cluster/limit-storage-consumption/

20. Write an ingress rule that redirects calls to /foo to one service and to /bar to another
21. Write a service that exposes nginx on a nodeport
a. Change it to use a cluster port
b. Scale the service
c. Change it to use an external IP
d. Change it to use a load balancer
22. Deploy nginx with 3 replicas and then expose a port
a. Use port forwarding to talk to a specific port
k port-forward deployment/nginx-deployment 8000:80

```
630  k run web2 --image=nginx --replica=2
  631  k run web2 --image=nginx -replica=2
  632  k run web2 --image=nginx --replicas=2
  633  k get deplpoy
  634  k get deploy
  635  k get po
  636  k scale deploy web2 --replicas=4
  637  k get po
  638  k expose deploy web2 --port=80
  639  k get svc
  640  k expose deploy web2 --type=Nodeport
  641  k expose deploy web2 --type=NodePort
  642  k expose deploy web2 --type=NodePort --port=8080
  643  k describe svc web2
  644  k delete svc web2
  645  k expose deploy web2 --type=NodePort --port=80
```

```
  646  k get svc
  647  k get deploy -o wide
  648  k set image deploy nginx=nginx:1.9
  649  k set image deploy web2 nginx=nginx:1.9
  650  k set image deploy web2 web2=nginx:1.9
  651  k get deploy -o wide
  652  k rollout undo deploy web2
  653  k get deploy -o wide
  654  k rollout history deploy web2
  655  k rollout history deploy web2 --help | more
  656  k rollout history deploy web2 --revision=2
  657  k rollout history deploy web2 --revision=3
  658  history
root@kub-master:/home/gvemala/practice#
```

curl localshot:8000
23. Make an API call using CURL and proper certs
24. Upgrade a cluster with kubeadm
25. Get logs for a pod
26. Deploy a pod with the wrong image name (like --image=nginy) and find the error message.

```
  Warning  Failed     11s   kubelet, kube-02   Failed to pull image "nginy": rpc error: code =
Unknown desc = Error response from daemon: pull access denied for nginy, repository does
not exist or may require 'docker login'
  Warning  Failed     11s   kubelet, kube-02   Error: ErrImagePull
  Normal   BackOff    10s   kubelet, kube-02   Back-off pulling image "nginy"
  Warning  Failed     10s   kubelet, kube-02   Error: ImagePullBackOff
```

27. Get logs for kubectl
28. Get logs for the scheduler
29. Restart kubelet

Non-K8S
30. Convert a CRT to a PEM
a. Convert it back
31. Backup an etcd cluster
32. List the members of an etcd cluster
Find the health of etcd

```
# List all pods in ps output format.
  kubectl get pods

  # List all pods in ps output format with more information (such as node name).
```

```
  kubectl get pods -o wide

  # List a single replication controller with specified NAME in ps output format.
  kubectl get replicationcontroller web

  # List deployments in JSON output format, in the "v1" version of the "apps" API
group:
  kubectl get deployments.v1.apps -o json

  # List a single pod in JSON output format.
  kubectl get -o json pod web-pod-13je7

  # List a pod identified by type and name specified in "pod.yaml" in JSON output
format.
  kubectl get -f pod.yaml -o json

  # Return only the phase value of the specified pod.
  kubectl get -o template pod/web-pod-13je7 --template={{.status.phase}}


  # List all replication controllers and services together in ps output format.
  kubectl get rc,services

  # List one or more resources by their type and names.
  kubectl get rc/web service/frontend pods/web-pod-13je7


  # Edit the job 'myjob' in JSON using the v1 API format:
  kubectl edit job.v1.batch/myjob -o json

  # Edit the deployment 'mydeployment' in YAML and save the modified config in its
annotation:
  kubectl edit deployment/mydeployment -o yaml --save-config
```

https://kubernetes.io/docs/reference/kubectl/cheatsheet/

https://github.com/ahmetb/kubernetes-network-policy-recipes

https://github.com/arush-sal/cka-practice-environment/tree/master/lab/files/questions

DENY all traffic to an application

This NetworkPolicy will drop all traffic to pods of an application, selected using Pod Selectors.

Run a nginx Pod with labels `app=web` and expose it at port 80:

```
kubectl run web --image=nginx --labels app=web --expose --port 80
```

Run a temporary Pod and make a request to `web` Service:

```
$ kubectl run --rm -i -t --image=alpine test-$RANDOM -- sh
/ # wget -qO- http://web
```

Limit traffic to application:

create Networking Policies allowing traffic from only certain Pods.

- Restrict traffic to a service only to other microservices that need to use it.
- Restrict connections to a database only to the application using it.

Suppose your application is a REST API server, marked with labels `app=bookstore` and `role=api`:

```
kubectl run apiserver --image=nginx --labels app=bookstore,role=api --expose --port
80
```

Save the following NetworkPolicy to `api-allow.yaml` to restrict the access only to other pods (e.g. other microservices) running with label `app=bookstore`:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: api-allow
spec:
  podSelector:
    matchLabels:
      app: bookstore
      role: api
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: bookstore
```

Test the Network Policy is **blocking** the traffic, by running a Pod without the `app=bookstore` label:

```
$ kubectl run test-$RANDOM --rm -i -t --image=alpine -- sh
/ # wget -qO- --timeout=2 http://apiserver
wget: download timed out
```

Traffic is blocked!

Test the Network Policy is **allowing** the traffic, by running a Pod with the `app=bookstore` label:

```
$ kubectl run test-$RANDOM --rm -i -t --image=alpine --labels
app=bookstore,role=frontend -- sh
/ # wget -qO- --timeout=2 http://apiserver
<!DOCTYPE html>
<html><head>
```

Traffic is allowed.

ALLOW all traffic to an application

Use Case: After applying a deny-all policy which blocks all non-whitelisted traffic to the application, now you to allow access to an application from all pods in the current namespace.

Applying this policy makes any other policies restricting the traffic to the pod void, and allow all traffic to it from its namespace and other namespaces.

```
Create a service that uses an external load balancer and points to a 3 pod cluster running
nginx.
```

https://kubernetes.io/docs/tasks/access-application-cluster/connecting-frontend-backend/

Similar to the backend, the frontend has a Deployment and a Service. The configuration for the Service has `type: LoadBalancer`, which means that the Service uses the default load balancer of your cloud provider.

```
How to Audit the whole cluster to determine the cluster health and activities.
```

PORT REQUIREMENT FOR KUBERNETES:
==========

MASTER NODE(s):
==========
TCP 6443* Kubernetes API server
TCP 2379-2380 etcd server client API
TCP 10250 Kubelet API
TCP 10251 Kube-Scheduler
TCP 10252 Kube-controller-manager
TCP 10255 Read-only Kubelet API

Worker Nodes:
==========
TCP 10250 Kubelet API
TCP 10255 Read-Only Kubelet API
TCP 3000-32767 NodePort Services

1. Pretend that node 3 is your favorite node. Maybe it's got all SSDs. Maybe it's got a fast network or a GPU. Or maybe it sent you a nice tweet. Label this node in some way so that you can schedule a pod to it.

Ans: kubectl label node *node3-name* myDarling=bestOne.

2. For my pod to be launched on my favorite node, I used this yaml:

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
  - name: busybox
    image: busybox
    command:
      - sleep
      - "300"
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
  nodeSelector:
    myDarling: bestOne
```

15. Create a custom resource definition
a. Display it in the API with curl

https://kubernetes.io/docs/tasks/administer-cluster/access-cluster-api/

```
root@kube-01:~#

root@kube-01:~# k proxy &

[1] 19112

root@kube-01:~# Starting to serve on 127.0.0.1:8001


root@kube-01:~# k get crd

NAME                                    CREATED AT

bgpconfigurations.crd.projectcalico.org        2019-02-26T19:16:57Z

clusterinformations.crd.projectcalico.org      2019-02-26T19:16:57Z

crontabs.stable.example.com                    2019-03-01T04:56:03Z

felixconfigurations.crd.projectcalico.org      2019-02-26T19:16:57Z

globalnetworkpolicies.crd.projectcalico.org    2019-02-26T19:16:58Z

globalnetworksets.crd.projectcalico.org        2019-02-26T19:16:58Z

hostendpoints.crd.projectcalico.org            2019-02-26T19:16:57Z

ippools.crd.projectcalico.org                  2019-02-26T19:16:57Z

networkpolicies.crd.projectcalico.org          2019-02-26T19:16:58Z

root@kube-01:~#

root@kube-01:~#

root@kube-01:~# curl
http://localhost:8001/apis/stable.example.com/v1/namespaces/*/crontabs/
```

{"apiVersion":"stable.example.com/v1","items":[],"kind":"CronTabList","metadata":{"continue":"","resourceVersion":"298545","selfLink":"/apis/stable.example.com/v1/namespaces/*/crontabs/"}}

root@kube-01:~#

the network policy is only supported by these networks

```
Calico

Cilium

Kube-router

Romana

Weave Net
```

1 .list all the pv's sort by capacity.

k get pv –sort-by=.spec.capacity.storage

2. label the node

3. make one node unavailable.

4. Deployment

5. Create a pod named non-persistent-redis that specifies a named-volume with name app-cache, and mount path /data/redis. It should launch in the staging namespace and the volume MUST NOT be persistent.

```
apiVersion: v1
kind: Pod
metadata:
  name: non-persistent-redis
  namespace: staging
spec:
  containers:
  - image: nginx
    name: non-persistent-redis
    volumeMounts:
    - mountPath: /data/redis
      name: app-cache
  volumes:
  - name: app-cache
    hostPath:
      # directory location on host
      path: /data/redis
      # this field is optional
      type: DirectoryOrCreate
```

Q: Use the utility nslookup to look up the DNS records of the service and pod.
From this guide, https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/
Look for "Quick Diagnosis"
$ kubectl exec -ti busybox -- nslookup mysvc.myns.svc.cluster.local
Naming conventions for services and pods:
For a regular service, this resolves to the port number and the CNAME: my-svc.my-namespace.svc.cluster.local.

For a headless service, this resolves to multiple answers, one for each pod that is backing the service, and contains the port number and a CNAME of the pod of the form auto-generated-name.my-svc.my-namespace.svc.cluster.local

When enabled, pods are assigned a DNS A record in the form of pod-ip-address.my-namespace.pod.cluster.local.
For example, a pod with IP 1.2.3.4 in the namespace default with a DNS name of cluster.local would have an entry: 1-2-3-4.default.pod.cluster.local

10. Create a pod that uses secrets
https://kubernetes.io/docs/concepts/configuration/secret/
a. Pull secrets from environment variables
b. Pull secrets from a volume

https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/

11. loadbalance with running 2 --replicas=7

12. create one namespace.

Deploy nginx pod on it.

12. production namespace.

K get po

4 pod running

K get svc

Bar

Pods should implement the same service which is running.

K gte po > filename.txt

14check all the nodes which are not tainted , put that number in the file.

15 node not ready . in the node kubelet was not running.

16. 3. Create a deployment running nginx version 1.12.2 that will run in 2 pods
https://kubernetes.io/docs/tasks/run-application/run-stateless-application-deployment/

a. Scale this to 4 pods.
b. Scale it back to 2 pods.
c. Upgrade this to 1.13.8
d. Check the status of the upgrade
e. How do you do this in a way that you can see history of what happened?

1. Both master & nodes were down.

Command for the pod environment variables.

Increase the replicas for loadbalancer deployment in namespace given.

In this task, you will configure a new Node, **ik8s-node-0**, to join a Kubernetes cluster as follows:

- Configure `kubelet` for automatic certificate rotation and ensure that both server and client CSRs are automatically approved and signed as appropriate via the use of `RBAC`
- Ensure that the appropriate `cluster-info` `ConfigMap` is created and configured appropriately in the correct namespace so that future Nodes can easily join the cluster
- Your bootstrap `kubeconfig` should be created on the new Node at `/etc/kubernetes/bootstrap-kubelet.conf` (do *not* remove this file once your Node has successfully joined the cluster)
- The appropriate cluster-wide CA certificate is located on the Node at `/etc/kubernetes/pki/ca.crt`. You should ensure that any automatically issued certificates are installed to the Node at `/var/lib/kubelet/pki`, and that the `kubeconfig` file for `kubelet` will be rendered at `/etc/kubernetes/kubelet.conf` upon successful bootstrapping
- Use an additional group for bootstrapping Nodes attempting to join the cluster, which should be called `system:bootstrappers:cka:default-node-token`
- Solution should start automatically on boot, with the `systemd` service unit file for `kubelet` available at

1. Install client tools

```
wget -q --show-progress --https-only --timestamping \
https://pkg.cfssl.org/R1.2/cfssl_linux-amd64 \
https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
Verify: cfssl version
```

2. Install kubelet

```
wget https://storage.googleapis.com/kubernetes-
release/release/v1.12.0/bin/linux/amd64/kubectl
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```

```
kubectl version --client

3. Certificate authority
{

cat > ca-config.json <<EOF
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "kubernetes": {
        "usages": ["signing", "key encipherment", "server auth", "client auth"],
        "expiry": "8760h"
      }
    }
  }
}
EOF

cat > ca-csr.json <<EOF
{
  "CN": "Kubernetes",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "Kubernetes",
      "OU": "CA",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert -initca ca-csr.json | cfssljson -bare ca

}
Files: ca-key.pem & ca.pem

Client & server certificates
Admin client certificate:

{

cat > admin-csr.json <<EOF
{
  "CN": "admin",
  "key": {
    "algo": "rsa",
```

```
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:masters",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  admin-csr.json | cfssljson -bare admin

}
Files: Admin-key.pem
       Admin.pem
```

Kubelet client certificates:

```
for instance in worker-0; do
cat > ${instance}-csr.json <<EOF
{
  "CN": "system:node:${instance}",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:nodes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=${instance},${EXTERNAL_IP},${INTERNAL_IP} \
  -profile=kubernetes \
  ${instance}-csr.json | cfssljson -bare ${instance}
done
```

```
Files: worker-0-key.pem
       Worker-0.pem
```

The Kube Proxy Client Certificate

Generate the `kube-proxy` client certificate and private key:

```
{

cat > kube-proxy-csr.json <<EOF
{
  "CN": "system:kube-proxy",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:node-proxier",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  kube-proxy-csr.json | cfssljson -bare kube-proxy

}

kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --embed-certs=true \
    --server=https://${KUBERNETES_PUBLIC_ADDRESS}:6443 \
    --kubeconfig=${instance}.kubeconfig

File: worker-0.kubeconfig

kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --embed-certs=true \
    --server=https://${KUBERNETES_PUBLIC_ADDRESS}:6443 \
    --kubeconfig=kube-proxy.kubeconfig

File: kube-proxy.kubeconfig
```

RBAC for kubelet authorization:

```
cat <<EOF | kubectl apply --kubeconfig admin.kubeconfig -f -
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:kube-apiserver-to-kubelet
rules:
  - apiGroups:
      - ""
    resources:
      - nodes/proxy
      - nodes/stats
      - nodes/log
      - nodes/spec
      - nodes/metrics
    verbs:
      - "*"
EOF
```

The Kubernetes API Server authenticates to the Kubelet as the `kubernetes` user using the client certificate as defined by the `--kubelet-client-certificate` flag.

Bind the `system:kube-apiserver-to-kubelet` ClusterRole to the `kubernetes` user:

```
cat <<EOF | kubectl apply --kubeconfig admin.kubeconfig -f -
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: system:kube-apiserver
  namespace: ""
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:kube-apiserver-to-kubelet
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: kubernetes
EOF

Complete the steps provided in this page
```

https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/09-bootstrapping-kubernetes-workers.md

https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/10-configuring-kubectl.md

Etcd Backup

The question will have cert paths,keys need to replace in below and run the command.

ETCDCTL_API=3 ./etcdctl --endpoints https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=/etc/kubernetes/pki/etcd/healthcheck-client.key snapshot save ./snapshot.db

Both master & worker node are not running . need to fix it.

Will get output as below. Need to fix it.

root@kube-01:~# k get no

The connection to the server 68.183.82.170:6443 was refused - did you specify the right host or port?

root@kube-01:~#

Staticpods

Create the Pod manifest at the specified location and then edit the systemd service file for kubelet(/etc/kubernetes/manifests) to include `--pod-manifest-path=/specified/path`. Once done restart the service.

CKA Exam questions :

--> Create a pod with Non-persistance volume

image: redis

Mount Volume AND Path /data/redis

--> scale deployment to replicas 4

--> create an init container to a pod and run a command to create a file /workdir/exec.txt

we should create pod and add init container list and execute command in the pod

--> create pod with image 1.19.0-alpine and update it later with 1.20.10-alpine

--> create a deployment nginx and expose the service to clusterIP

--> etcd back up

--> create a service nginx-dns and use nslook up utility to find the service and pod dns records.

--> Worker node was not ready. Restaring kubelet fixed the issue

--> create list of pods in a label data=mapping

--> create a pod to get scheduled on single node

→ create a pod kuccc1 to run a single container of each image.It may be between (nginx+ cachedman + redis + consul)

--> create a service in new namespace

--> make node unscheduable and make it reschedulable

-- > list the pods in a particular name space and and find the max cpu pod to a file

--> Log in to a node and create a pod with name web and image nginx.

-- secrets : secret-name

username : bob

--> create secret and mount it as volume under path /secrets.

--> create a new secret and export the username as TOPSECRET