Q1) List all the configmaps in the cluster

Ans:

```
$ kubectl get cm
or
$ kubectl get configmap
```

Q2) Create a configmap called myconfigmap with literal value appname=myapp

Ans:

```
$ kubectl create cm myconfigmap --from-literal=appname=myapp
```

```
No resources found in default namespace.
root@Master:~# kubectl create cm myconfigmap --from-literal=appname=myapp
configmap/myconfigmap created
root@Master:~#
```

Q3) Verify the configmap we just created has this data

Ans:

```
$ kubectl get cm
or
$ kubectl get configmap
```

```
root@Master:~# kubectl get cm
NAME          DATA   AGE
myconfigmap   1      38s
root@Master:~#
```

Q4) Delete the configmap myconfigmap we just created

Ans:

```
$ kubectl delete cm myconfigmap
```

```
root@Master:~# kubectl delete cm myconfigmap
configmap "myconfigmap" deleted
root@Master:~#
```

Q5) Create a file called config.txt with two values key1=value1 and key2=value2 and verify the file

Ans:

```
$ cat >> config.txt << EOF
key1=value1
key2=value2
EOF

$ cat config.txt
```

Q6) Create a configmap named keyvalcfgmap and read data from the file config.txt and verify that configmap is created correctly
Ans:

```
$ kubectl create cm keyvalcfgmap --from-file=config.txt

$ kubectl get cm keyvalcfgmap -o yaml
```

```
root@Master:~# cat >> config.txt << EOF
> key1=value1
> key2=value2
> EOF
root@Master:~# cat config.txt
key1=value1
key2=value2
root@Master:~# kubectl create cm keyvalcfgmap --from-file=config.txt
configmap/keyvalcfgmap created
root@Master:~# kubectl get cm keyvalcfgmap -o yaml
apiVersion: v1
data:
  config.txt: |
    key1=value1
    key2=value2
kind: ConfigMap
metadata:
  creationTimestamp: "2021-03-26T15:27:24Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:config.txt: {}
    manager: kubectl
    operation: Update
    time: "2021-03-26T15:27:24Z"
  name: keyvalcfgmap
  namespace: default
  resourceVersion: "15100"
  selfLink: /api/v1/namespaces/default/configmaps/keyvalcfgmap
  uid: 6fa0bdc8-86eb-441e-973b-48c312d0e01b
root@Master:~#
```

Q7) Create an nginx pod and load environment values from the above configmap keyvalcfgmap and exec into the pod and verify the environment variables and delete the pod.
Ans:

```
$ vim env-pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
```

```
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    envFrom:
    - configMapRef:
        name: keyvalcfgmap
```

```
$ kubectl create -f env-pod.yml

// verify
$ kubectl exec -it env-pod -- env

$ kubectl delete pods nginx
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    envFrom:
    - configMapRef:
        name: keyvalcfgmap
~
~
~
root@Master:~# kubectl exec -it nginx -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx
TERM=xterm
config.txt=key1=value1
key2=value2

KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
NGINX_VERSION=1.19.8
NJS_VERSION=0.5.2
PKG_RELEASE=1~buster
HOME=/root
root@Master:~# kubectl delete pods nginx
pod "nginx" deleted
root@Master:~#
```

Q8) Create an env file file.env with var1=val1 and create a configmap envcfgmap from this env file and verify the configmap
Ans:

```
$ echo var1=val2 > file.env
$ cat file.env

$ kubectl create cm envcfgmap --from-env-file=file.env

$ kubectl describe cm envcfgmap
```

```
root@Master:~# echo var1=val2 > file.env
root@Master:~# cat file.envkubectl create cm envcfgmap --from-env-file=file.env
cat: unrecognized option '--from-env-file=file.env'
Try 'cat --help' for more information.
root@Master:~# cat file.env
var1=val2
root@Master:~# kubectl create cm envcfgmap --from-env-file=file.env
configmap/envcfgmap created
root@Master:~# kubectl describe cm envcfgmap
Name:         envcfgmap
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
var1:
----
val2
Events:  <none>
root@Master:~#
```

Q9) Create an nginx pod and load environment values from the above configmap envcfgmap and exec into the pod and verify the environment variables and delete the pod
Ans:

```
$ vim nginx-pod.yml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
```

```
  env:
  - name: ENVIRONMENT
    valueFrom:
      configMapKeyRef:
        name: envcfgmap
        key: var1
```

```
$ kubectl create -f nginx-pod.yml
$ kubectl exec -it nginx -- env

$ kubectl delete pod nginx
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    env:
    - name: ENVIRONMENT
      valueFrom:
        configMapKeyRef:
          name: envcfgmap
          key: var1
```

```
root@Master:~# vim nginx-pod.yml
root@Master:~# cat nginx-pod.yml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    env:
    - name: ENVIRONMENT
      valueFrom:
        configMapKeyRef:
          name: envcfgmap
          key: var1

root@Master:~# kubectl create -f nginx-pod.yml
pod/nginx created
root@Master:~# kubectl exec -it nginx -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx
TERM=xterm
ENVIRONMENT=val2
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
NGINX_VERSION=1.19.8
NJS_VERSION=0.5.2
PKG_RELEASE=1~buster
HOME=/root
root@Master:~# kubectl delete pod nginx
pod "nginx" deleted
root@Master:~#
```

Q10) Create a configmap called cfgvolume with values var1=val1, var2=val2 and create an nginx pod with volume nginx-volume which reads data from this configmap cfgvolume and put it on the path /etc/cfg.
Ans:

```
// first create a configmap cfgvolume
$ kubectl create cm cfgvolume --from-literal=var1=val1 --from-literal=var2=val2
// verify the configmap
$ kubectl describe cm cfgvolume
```

```
root@Master:~# kubectl create cm cfgvolume --from-literal=var1=val1 --from-literal=var2=val2
configmap/cfgvolume created
root@Master:~# kubectl describe cm cfgvolume
Name:         cfgvolume
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
var1:
----
val1
var2:
----
val2
Events:  <none>
```

```
// create pod
$ vim nginx-volume.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  volumes:
  - name: nginx-volume
    configMap:
      name: cfgvolume
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - name: nginx-volume
      mountPath: /etc/cfg
```

```
$ kubectl create -f nginx-volume.yml

// exec into the pod

$ kubectl exec -it nginx -- /bin/sh/

check the path
$ cd /etc/cfg

Check files

$ ls

$ exit
```

```
root@Master:~# vim nginx-volume.yml
root@Master:~# cat nginx-volume.yml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  volumes:
  - name: nginx-volume
    configMap:
      name: cfgvolume
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - name: nginx-volume
      mountPath: /etc/cfg

root@Master:~# kubectl create -f nginx-volume.yml
pod/nginx created
root@Master:~# kubectl exec -it nginx -- /bin/sh/
OCI runtime exec failed: exec failed: container_linux.go:349: starting container process caused "exec: \"/bin/sh/\": stat /bin/sh/: not a directory": unknown:
 Are you trying to mount a directory onto a file (or vice-versa)? Check if the specified host path exists and is the expected type
command terminated with exit code 126
root@Master:~# kubectl exec -it nginx -- /bash
OCI runtime exec failed: exec failed: container_linux.go:349: starting container process caused "exec: \"/bash\": stat /bash: no such file or directory": unkn
own
command terminated with exit code 126
root@Master:~# kubectl exec -it nginx -- bash
root@nginx:/# cd /etc/cfg
root@nginx:/etc/cfg# ls
var1  var2
root@nginx:/etc/cfg#
```

Q11) Create a pod called secbusybox with the image busybox which executes command sleep 3600 and makes sure any Containers in the Pod, all processes run with user ID 1000 and with group id 2000 and verify.
Ans:

```
$ vim busybox.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: secbusybox
  name: secbusybox
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 2000
  containers:
  - args:
    - /bin/sh
    - -c
    - sleep 3600;
    image: busybox
    name: secbusybox
```

create pod

```
$ kubectl create -f busybox.yml
```

// verify

```
$ kubectl exec -it secbusybox -- sh
```

Show id and group using below command

```
$ id
```

```
$ exit
```

```
root@Master:~# vim busybox.yml
root@Master:~# cat busybox.yml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: secbusybox
  name: secbusybox
spec:
  securityContext:
    runAsUser: 1000
    runAsGroup: 2000
  containers:
  - args:
    - /bin/sh
    - -c
    - sleep 3600;
    image: busybox
    name: secbusybox

root@Master:~# kubectl create -f busybox.yml
pod/secbusybox created
root@Master:~# kubectl exec -it secbusybox -- sh
/ $ id
uid=1000 gid=2000
/ $
```

Q12) Create pod with an nginx image and configure the pod with capabilities NET_ADMIN and SYS_TIME verify the capabilities.
Ans:

```
$ vim context-pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    securityContext:
      capabilities:
        add: ["SYS_TIME", "NET_ADMIN"]
    name: nginx-context
```

```
// create pod
$ kubectl create -f context-pod.yml

// exec and verify
$ kubectl exec -it nginx -- sh
$ cd /proc/1
$ cat status

// you should see these values
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb

$ exit
```

```
root@Master:~# vim context-pod.yml
root@Master:~# cat context-pod.yml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    securityContext:
      capabilities:
        add: ["SYS_TIME", "NET_ADMIN"]
    name: nginx-context
```

```
root@Master:~# kubectl create -f context-pod.yml
pod/nginx created
root@Master:~# kubectl exec -it nginx -- sh
# cd /proc/1
# cat status
Name:   nginx
Umask:  0022
State:  S (sleeping)
Tgid:   1
Ngid:   0
Pid:    1
PPid:   0
TracerPid:      0
Uid:    0       0       0       0
Gid:    0       0       0       0
FDSize: 64
Groups:
NStgid: 1
NSpid:  1
NSpgid: 1
NSsid:  1
VmPeak:    10664 kB
VmSize:    10636 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:      5840 kB
VmRSS:      5840 kB
RssAnon:             804 kB
RssFile:            5036 kB
RssShmem:              0 kB
VmData:      980 kB
VmStk:       132 kB
VmExe:       984 kB
VmLib:      3796 kB
VmPTE:        52 kB
VmSwap:        0 kB
HugetlbPages:          0 kB
CoreDumping:    0
THP_enabled:    1
Threads:        1
SigQ:   0/15598
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000040001000
SigCgt: 0000000198016a07
CapInh: 00000000aa0435fb
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
CapBnd: 00000000aa0435fb
CapAmb: 0000000000000000
NoNewPrivs:     0
Seccomp:        0
Speculation_Store_Bypass:       vulnerable
Cpus_allowed:   3
Cpus_allowed_list:      0-1
Mems_allowed:   00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000
```

Before next question you need metrix server on the cluster so first we will install metrix server.
To install:

```
$ kubectl create -f https://raw.githubusercontent.com/k21academyuk/Kubernetes/master/metrics-server.yaml
```

```
root@Master:~# kubectl create -f https://raw.githubusercontent.com/k21academyuk/Kubernetes/master/metrics-server.yaml
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.apps/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@Master:~#
```

Q13) Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.
Ans:

```
$ vim quota-pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
```

```
spec:
  containers:
  - image: busybox
    name: busybox
    resources:
      requests:
        cpu: "0.5"
      limits:
        cpu: "1"
```

```
// add the resources section and create pod
$ kubectl create -f quota-pod.yml

// verify
$ kubectl top pod
```

```
root@Master:~# vim quota-pod.yml
root@Master:~# cat quota-pod.yml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
spec:
  containers:
  - image: busybox
    name: busybox
    resources:
      requests:
        cpu: "0.5"
      limits:
        cpu: "1"
root@Master:~# kubectl create -f quota-pod.yml
pod/busybox created
```