1- How many `DaemonSets` are created in the cluster in all namespaces?

```
controlplane:~$ kubectl get ds --all-namespaces
NAMESPACE     NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   canal        2         2         2       2            2           kubernetes.io/os=linux   35d
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   35d
controlplane:~$ kubectl get ds --all-namespaces --no-headers | wc -l
2
```

2- what DaemonSets exist on the kube-system namespace?

```
controlplane:~$ k get daemonsets --namespace=kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
canal        2         2         2       2            2           kubernetes.io/os=linux   35d
kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   35d
```

3- What is the image used by the POD deployed by the `kube-proxy` DaemonSet

```
Containers:
 kube-proxy:
  Image:      registry.k8s.io/kube-proxy:v1.32.1
  Port:       <none>
  Host Port:  <none>
```

4- Deploy a DaemonSet for `FluentD` Logging. Use the given specifications.

Name: elasticsearch
Namespace: kube-system
Image: k8s.gcr.io/fluentd-elasticsearch:1.20

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
```

5- Deploy a pod named `nginx-pod` using the `nginx:alpine` image with the labels set to `tier=backend`.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    tier: backend
spec:
  containers:
  - name: nginx
    image: nginx:alpine
    ports:
    - containerPort: 80
```

```
controlplane:~$ k apply -f ngin.yaml
pod/nginx-pod created
```

6- Deploy a `test` pod using the `nginx:alpine` image.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test
spec:
  containers:
  - name: nginx
    image: nginx:alpine
    ports:
    - containerPort: 80
```

```
controlplane:~$ k apply -f test.yaml
pod/test created
```

7- Create a service `backend-service` to expose the `backend` application within the cluster on port `80`.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    tier: backend
  ports:
    - targetPort: 80
      port: 80
```

```
controlplane:~$ k apply -f ser.yaml
service/backend-service created
```

8- try to curl the backend-service from the test pod. What is the response?

```
controlplane:~$ kubectl exec -it test -- sh
/ # curl backend-service
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

9- Create a deployment named `web-app` using the image `nginx` with `2` replicas

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

```
controlplane:~$ k apply -f deploy.yaml
deployment.apps/web-app created
controlplane:~$
```

10- Expose the `web-app` as service `web-app-service` application on port 80 and nodeport `30082` on the nodes on the cluster

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web-app-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30082
```

```
controlplane:~$ k apply -f serv.yaml
service/web-app-service created
```

11- access the web app from the node

```
controlplane:~$ k get nodes -o wide
NAME           STATUS   ROLES           AGE   VERSION   INTERNAL-IP   EXTERNAL-IP
controlplane   Ready    control-plane   35d   v1.32.1   172.30.1.2    <none>
node01         Ready    <none>          35d   v1.32.1   172.30.2.2    <none>
controlplane:~$ curl 172.30.2.2:30082
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
controlplane:~$
```

12- How many static pods exist in this cluster in all namespaces?

4 yaml files for 4 static pods

```
controlplane:~$ ls /etc/kubernetes/manifests/
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml
```

13- On which nodes are the static pods created currently?

```
controlplane:~$ k get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   35d   v1.32.1
node01         Ready    <none>          35d   v1.32.1
controlplane:~$ kubectl get pods --all-namespaces -o wide| grep -- -node01
controlplane:~$ kubectl get pods --all-namespaces -o wide| grep -- -controlplane
kube-system          etcd-controlplane                      1/1     Running   3 (34m ago)   35d   172.30.1.2   controlplane
kube-system          kube-apiserver-controlplane            1/1     Running   2 (34m ago)   35d   172.30.1.2   controlplane
kube-system          kube-controller-manager-controlplane   1/1     Running   2 (34m ago)   35d   172.30.1.2   controlplane
kube-system          kube-scheduler-controlplane            1/1     Running   2 (34m ago)   35d   172.30.1.2   controlplane
```

4 on controlplane, none on node01