1- How many Namespaces exist on the system?
Four

```
controlplane $ kubectl get ns
NAME              STATUS   AGE
default           Active   30d
kube-node-lease   Active   30d
kube-public       Active   30d
kube-system       Active   30d
```

2-How many po exist in the kube-system namespace?
11 pods

```
controlplane $ kubectl get ns
NAME              STATUS   AGE
default           Active   30d
kube-node-lease   Active   30d
kube-public       Active   30d
kube-syster       Active   30d
controlplane $ kubectl get po -n kube-syster
NAME                                   READY   STATUS    RESTARTS       AGE
calico-kube-controllers-5f94594857-zsl.2v   1/1     Running   2              30d
canal-n9k98                            2/2     Running   0              25r
canal-x6c5w                            2/2     Running   0              25r
coredns-68dc769db8-drf8l.              1/1     Running   0              30d
coredns-68dc769db8-sbbx7               1/1     Running   0              30d
etcd-controlplane                      1/1     Running   0              30d
kube-apiserver-controlplane            1/1     Running   2              30d
kube-controller-r anager-controlplane      1/1     Running   3 (13r  ago)   30d
kube-proxy-xnz4r                       1/1     Running   0              30d
kube-proxy-zbxrb                       1/1     Running   0              30d
kube-scl.eduler-controlplane           1/1     Running   2              30d
controlplane $
```

3- create a Deployment with
name= deployment-1
image= busybox
replicas= 3

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-1
  labels:
    app: busybox-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: busybox-app
  template:
    metadata:
      labels:
        app: busybox-app
    spec:
      containers:
      - name: busybox-app
        image: busybox
```

```
controlplane $ vim deploy-1
controlplane $ kubectl apply -f deploy-1
deployment.apps/deployment-1 created
controlplane $
```

4- How many Deployments and ReplicaSets exist on the system now?

```
controlplane $ kubectl get deployments.apps
NAME           READY   UP-T( -DATE   AVAILABLE   AGE
deployment-1   0/3     3             0           3m20s
controlplane $ kubectl get rs
NAME                      DESIRED   CURRENT   READY   AGE
deployment-1-5159d7c69    3         3         0       3m9s
controlplane $
```

One rs and one deployment

5- How many pods are ready with the deployment-1?

```
controlplane $ kubectl get po
NAME                            READY   STATUS             RESTARTS        AGE
deployment-1-5159d7c69-h7p77    0/1     CrashLoopBack(11   6 (4m15s ago)   9m44s
deployment-1-5159d7c69-kgcdq    0/1     CrashLoopBack(11   6 (3m39s ago)   9m44s
deployment-1-5159d7c69-qqnhv    0/1     CrashLoopBack(11   6 (3m45s ago)   9m44s
controlplane $
```

None of the pods are ready because there is nothing for busybox to do so the container exit

6- Update deployment-1 image to nginx then check the ready pods again

```
    spec:
      containers:
      - name: busybox-app
        image: nginx
```

```
controlplane $ vim deploy-1
controlplane $ kubectl apply -1 deploy-1
deployment.apps/deployment-1 con1igured
controlplane $ kubectl get po
NAME                              READY   STATUS    RESTARTS   AGE
deployment-1-5495d97559-25n89     1/1     Running   0          3m57s
deployment-1-5495d97559-4z16w     1/1     Running   0          3m55s
deployment-1-5495d97559-x9wk2     1/1     Running   0          4m2s
controlplane $
```

All of the 3 pods are ready and running

7- Run kubectl describe deployment deployment-1 and check events

What is the deployment strategy used to upgrade the deployment-1?

```
controlplane $ kubectl describe deployment deployment-1
Replicas:               3 desired | 3 updated | 3 total | 3 av
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
```

```
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  52m    deployment-controller  Scaled up replica set deployment-1-5159d7c69 to 3
  Normal  ScalingReplicaSet  37m    deployment-controller  Scaled up replica set deployment-1-5495d97559 to 1
  Normal  ScalingReplicaSet  36m    deployment-controller  Scaled down replica set deployment-1-5159d7c69 to 2 from
 3
  Normal  ScalingReplicaSet  36m    deployment-controller  Scaled up replica set deployment-1-5495d97559 to 2 from
1
  Normal  ScalingReplicaSet  36m    deployment-controller  Scaled down replica set deployment-1-5159d7c69 to 1 from
 2
  Normal  ScalingReplicaSet  36m    deployment-controller  Scaled up replica set deployment-1-5495d97559 to 3 from
2
  Normal  ScalingReplicaSet  36m    deployment-controller  Scaled down replica set deployment-1-5159d7c69 to 0 from
 1
  Normal  ScalingReplicaSet  6m24s  deployment-controller  Scaled up replica set deployment-1-5159d7c69 to 1 from 0
  Normal  ScalingReplicaSet  5m35s  deployment-controller  Scaled down replica set deployment-1-5495d97559 to 2 fro
m 3
  Normal  ScalingReplicaSet  5m35s  deployment-controller  Scaled up replica set deployment-1-5159d7c69 to 2 from 1
controlplane $
```

Rolling update

8- Rollback the deployment-1

```
controlplane $ kubectl rollout undo deployment/deployment-1
deployment.apps/deployment-1 rolled back
controlplane $ kubectl rollout history deployment deployment
```

What is the used image with the deployment-1?

```
controlplane $ kubectl get deployment -o wide
NAME           READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS    IMAGES    SELECTOR
deployment-1   2/3     2            2           51m   busybox-app   busybox   app=busybox-app
controlplane $
```

Busybox

10- Create a deployment with
Name: dev-deploy
Image: redis
Replicas: 2
Namespace: dev
Resources Requests:
CPU: .5 vcpu
Mem: 1G
Resources Limits:
CPU: 1 vcpu
Mem: 2G

```
api\ersion: apps/v1
kind: Deployment
metadata:
  name: dev-deploy
  labels:
    app: redis
spec:
  replicas: 2
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
        namespace: dev
    spec:
      containers:
      - name: redis
        image: redis
```

```
api\ersion: v1
kind: Namespace
metadata:
  name: dev
  labels:
    name: dev
```

```
api\ersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-demo
  namespace: dev
spec:
  hard:
    requests.cpu: "0.5 vcpu"
    requests.memory: 1 G
    limits.cpu: "1 vcpu"
    limits.memory: 2 G
~
```

```
controlplane $ kubectl apply -f dev2
deployment.apps/dev-deploy configured
controlplane $ vim new-quota
controlplane $ kubectl apply -f new-quota
resourcequota/mem-cpu-demo created
```