Q1) Create a deployment called webapp with image nginx with 5 replicas.

Ans:

```
$ kubectl create deploy webapp --image=nginx --dry-run=client -o yaml > webapp.yaml
// change the replicas 1 to 5 in the yaml and create it
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: webapp
  name: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: webapp
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: webapp
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

```
$ kubectl create -f webapp.yaml
```

Or

You can use Imperative Command:

```
$ kubectl create deployment webapp --image=nginx --replicas=5
```

```
root@master:~# kubectl create deployment webapp --image=nginx --replicas=5
deployment.apps/webapp created
root@master:~# kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
frontend       3/3     3            3           20h
redis-master   2/2     2            2           21h
redis-slave    2/2     2            2           21h
webapp         3/5     5            3           16s
root@master:~# kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
frontend       3/3     3            3           20h
redis-master   2/2     2            2           21h
redis-slave    2/2     2            2           21h
webapp         5/5     5            5           22s
root@master:~#
```

Q2) Get the deployment you just created with labels.
Ans:

```
$ kubectl get deploy webapp --show-labels
```

```
root@master:~# kubectl get deploy webapp --show-labels
NAME     READY   UP-TO-DATE   AVAILABLE   AGE     LABELS
webapp   5/5     5            5           2m42s   app=webapp
root@master:~#
```

Q3) Output the yaml file of the deployment you just created.
Ans:

```
$ kubectl get deploy webapp -o yaml
```

```
root@master:~# kubectl get deploy webapp -o yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2021-03-19T12:07:59Z"
  generation: 1
  labels:
    app: webapp
  managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
      f:metadata:
        f:labels:
          .: {}
          f:app: {}
      f:spec:
        f:progressDeadlineSeconds: {}
        f:replicas: {}
        f:revisionHistoryLimit: {}
        f:selector: {}
        f:strategy:
          f:rollingUpdate:
            .: {}
            f:maxSurge: {}
            f:maxUnavailable: {}
          f:type: {}
        f:template:
          f:metadata:
            f:labels:
              .: {}
              f:app: {}
          f:spec:
            f:containers:
              k:{"name":"nginx"}:
                .: {}
                f:image: {}
                f:imagePullPolicy: {}
                f:name: {}
                f:resources: {}
                f:terminationMessagePath: {}
```

Many unnessary information available remove this by using below commands:

```
$ kubectl get deploy webapp -o yaml | grep -v f:
```

```
root@master:~# kubectl get deploy webapp -o yaml | grep -v f:
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2021-03-19T12:07:59Z"
  generation: 1
  labels:
    app: webapp
  managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
          .: {}
            .: {}
              .: {}
              k:{"name":"nginx"}:
                .: {}
    manager: kubectl-create
    operation: Update
    time: "2021-03-19T12:07:59Z"
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
          .: {}
          .: {}
          k:{"type":"Available"}:
            .: {}
          k:{"type":"Progressing"}:
            .: {}
    manager: kube-controller-manager
    operation: Update
    time: "2021-03-19T12:08:21Z"
  name: webapp
  namespace: k21
  resourceVersion: "94536"
  uid: 8763d93a-9231-4d9c-ab9d-ea4530a18e2d
spec:
  progressDeadlineSeconds: 600
  replicas: 5
  revisionHistoryLimit: 10
  selector:
```

Q4) Get the pods of this deployment

Ans:

```
$ kubectl get pods -l app=webapp
```

```
root@master:~# kubectl get pods -l app=webapp
NAME                      READY   STATUS    RESTARTS   AGE
webapp-5654c984c-8qv7h    1/1     Running   0          24m
webapp-5654c984c-bdqrl    1/1     Running   0          24m
webapp-5654c984c-f2wp5    1/1     Running   0          24m
webapp-5654c984c-x6zfd    1/1     Running   0          24m
webapp-5654c984c-xc4cs    1/1     Running   0          24m
root@master:~#
```

Q5) Scale the deployment from 5 replicas to 7 replicas and verify

Ans:

```
$ kubectl scale deploy webapp --replicas=7
$ kubectl get pods -l app=webapp
```

```
root@master:~# kubectl scale deploy webapp --replicas=7
deployment.apps/webapp scaled
root@master:~# kubectl get pods -l app=webapp
NAME                      READY   STATUS              RESTARTS   AGE
webapp-5654c984c-8qv7h    1/1     Running             0          28m
webapp-5654c984c-bdqrl    1/1     Running             0          28m
webapp-5654c984c-f2wp5    1/1     Running             0          28m
webapp-5654c984c-tdzgc    1/1     Running             0          8s
webapp-5654c984c-x4nnm    0/1     ContainerCreating   0          7s
webapp-5654c984c-x6zfd    1/1     Running             0          28m
webapp-5654c984c-xc4cs    1/1     Running             0          28m
root@master:~#
```

Q6) Get the deployment rollout status

Ans:

```
$ kubectl rollout status deploy webapp
```

```
root@master:~# kubectl rollout status deploy webapp
deployment "webapp" successfully rolled out
root@master:~#
```

Q7) Get the replicaset that created with this deployment

Ans:

```
$ kubectl get rs -l app=webapp
```

Q8) Delete the deployment you just created and watch all the pods are also being deleted

Ans:

```
$ kubectl delete deploy webapp
$ kubectl get pods -l app=webapp -w
```

Q9) Create a deployment of webapp with image nginx:1.17.1 with container port 80 and verify the image version

Ans:

```
$ kubectl create deploy webapp --image=nginx:1.17.1 --port=80
// verify
$ kubectl describe deploy webapp | grep Image
```

```
root@master:~# kubectl create deploy webapp --image=nginx:1.17.1 --port=80
deployment.apps/webapp created
root@master:~# kubectl describe deploy webapp | grep Image
    Image:        nginx:1.17.1
root@master:~#
```

Q10) Update the deployment with the image version 1.17.4 and verify
Ans:

```
$ kubectl set image deploy/webapp nginx=nginx:1.17.4
$ kubectl describe deploy webapp | grep Image
```

```
root@master:~# kubectl set image deploy/webapp nginx=nginx:1.17.4
deployment.apps/webapp image updated
root@master:~# kubectl describe deploy webapp | grep Image
    Image:        nginx:1.17.4
root@master:~#
```

Q11) Undo the deployment to the previous version 1.17.1 and verify Image has the previous version.
Ans:

```
$ kubectl rollout undo deploy webapp
$ kubectl describe deploy webapp | grep Image
$ kubectl rollout status deploy webapp
```

```
root@master:~# kubectl rollout undo deploy webapp
deployment.apps/webapp rolled back
root@master:~# kubectl describe deploy webapp | grep Image
    Image:        nginx:1.17.1
root@master:~# kubectl rollout status deploy webapp
deployment "webapp" successfully rolled out
root@master:~#
```

Q12) Update the deployment with the wrong image version 1.100 and verify something is wrong with the deployment
Ans:

```
$ kubectl set image deploy/webapp nginx=nginx:1.100
```

```
$ kubectl rollout status deploy webapp (still pending state)
$ kubectl get pods (ImagePullErr)
```

```
root@master:~# kubectl rollout status deploy webapp
Waiting for deployment "webapp" rollout to finish: 1 old replicas are pending termination...
^Croot@master:~# kubectl get pods
NAME                          READY   STATUS             RESTARTS   AGE
nginx-hpa-df4c75d8d-l5vx4     1/1     Running            1          84m
webapp-8b7d5d964-9ztn9        0/1     ImagePullBackOff   0          76m
webapp-b7889ff56-l5vtj        1/1     Terminating        0          78m
webapp-b7889ff56-pnvz5        1/1     Running            0          83s
root@master:~#
```

Q13) This question will require you to create a pod that runs the image kubegoldenguide/question-thirteen.
This image is a web server that has a health endpoint served at '/health'. The web server listens on port 8000. (It runs Python's SimpleHTTPServer.) It returns a 200 status code response when the application is healthy. The application typically takes sixty seconds to start.
Create a pod called test-pod to run this application, making sure to define liveness and readiness probes that use this health endpoint."
Ans:

In solution, we'll add a buffer and start the livenessProbe at 75 seconds, which is 15 seconds after the typical expected startup time, as indicated in the question four details.
it is possible to define a startupProbe such that the livenessProbe will not be started until an initial OK is returned.

```
$ vim test-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
  labels:
    role: myrole
spec:
  containers:
    - name: test-pod
      image: kubegoldenguide/question-thirteen
      ports:
        - name: web
          containerPort: 8000
          protocol: TCP
      readinessProbe:
        httpGet:
          path: /health
          port: 8000
        initialDelaySeconds: 60
        periodSeconds: 5
```

```
    livenessProbe:
      httpGet:
        path: /health
        port: 8000
      initialDelaySeconds: 75
      periodSeconds: 10
      timeoutSeconds: 5
```

```
$ kubectl create -f test-pod.yaml
```

Verify:
In this example, we should expect to see log messages for the readinessProbe up until ~ 75 seconds after the pod was started, at which point we should see log messages appear for both the readinessProbe and the livenessProbe.

```
root@Master:~# kubectl logs test-pod
Serving HTTP on 0.0.0.0 port 8000 ...
10.40.0.0 - - [26/Mar/2021 14:45:47] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:45:52] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:45:57] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:02] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:02] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:07] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:12] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:12] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:17] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:22] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:22] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:27] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:32] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:32] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:37] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:42] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:42] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:47] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:52] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:52] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:46:57] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:02] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:02] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:07] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:12] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:12] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:17] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:22] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:22] "GET /health HTTP/1.1" 200 -
10.40.0.0 - - [26/Mar/2021 14:47:27] "GET /health HTTP/1.1" 200 -
```

Q14) Create a hostPath PersistentVolume named task-pv-volume with storage 10Mi, access modes ReadWriteOnce, and volume at /mnt/data and verify.
Ans:

```
$ vim task-pv-volume.yaml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  capacity:
    storage: 10Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

```
$ kubectl create -f task-pv-volume.yaml
$ kubectl get pv
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  capacity:
    storage: 10Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

```
root@Master:~# vim task-pv-volume.yaml
root@Master:~# kubectl create -f task-pv-volume.yaml
persistentvolume/task-pv-volume created
root@Master:~# kubectl get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
task-pv-volume   10Mi       RWO            Retain           Available                                    10s
root@Master:~#
```

Q15) List Persistent Volumes in the cluster
Ans:

```
$ kubectl get pv
```

Q16) Create a PersistentVolumeClaim of at least 10Mi storage and access mode ReadWriteOnce and verify status is Bound

Ans:

```
$ vim task-pvc-volume.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pvc-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
```

```
$ kubectl create -f task-pvc-volume.yaml
$ kubectl get pvc
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
~
~
root@Master:~# vim task-pvc-volume.yaml
root@Master:~# kubectl create -f task-pvc-claim.yaml
error: the path "task-pvc-claim.yaml" does not exist
root@Master:~# vim task-pvc-volume.yaml
root@Master:~# kubectl create -f task-pvc-volume.yaml
persistentvolumeclaim/task-pvc-claim created
root@Master:~# kubectl get pvc
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
task-pvc-claim  Bound    task-pv-volume   10Mi       RWO                           27s
root@Master:~#
```

Q17) Create a Pod with an image Redis and configure a volume that lasts for the lifetime of the Pod.

Ans:

Note: emptyDir is the volume that lasts for the life of the pod.

```
$ vim redis-storage.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: redis-storage
      mountPath: /data/redis
  volumes:
  - name: redis-storage
    emptyDir: {}
```

```
$ kubectl create -f redis-storage.yaml
```


```
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: redis-storage
      mountPath: /data/redis
  volumes:
  - name: redis-storage
    emptyDir: {}
```


```
$ kubectl describe pod redis
```

```
root@Master:~# kubectl describe pod redis
Name:         redis
Namespace:    default
Priority:     0
Node:         worker-01/10.0.0.4
Start Time:   Fri, 26 Mar 2021 13:51:15 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.32.0.2
IPs:
  IP:  10.32.0.2
Containers:
  redis:
    Container ID:   docker://762dbda4a63dc450b0eb662cc48afb2974381ef13bac24a734b697ffad3f5dc2
    Image:          redis
    Image ID:       docker-pullable://redis@sha256:e97d506be34a39fa69f45eea846080d6985c2c9ee338c0d408c7ea4347f014a5
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 26 Mar 2021 14:01:01 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /data/redis from redis-storage (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-4bzlm (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  redis-storage:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:
    SizeLimit:  <unset>
  default-token-4bzlm:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-4bzlm
```

Q18) Exec into the above pod and create a file named file.txt with the text 'This is called the file' in the path /data/redis and open another tab and exec again with the same pod and verifies file exist in the same path.
Ans:

```
// first terminal
$ kubectl exec -it redis /bin/sh
$ cd /data/redis
$ echo 'This is called the file' > file.txt
```

```
root@Master:~# kubectl exec -it redis /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl kubectl exec [POD] -- [COMMAND] instead.
# cd /data/redis
# echo 'This is called the file' > file.txt
#
```

```
//open another tab
$ kubectl exec -it redis /bin/sh
$ cat /data/redis/file.txt
```

```
root@Master:~# kubectl exec -it redis /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl kubectl exec [POD] -- [COMMAND] instead.
# cat /data/redis/file.txt
This is called the file
# 
```