1. Deploy Replica Set and Replication Controller, and deployment. Also learn the advantages and disadvantages of each.
Answer:

Create a YAML file `replicationcontroller.yaml`:

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: rc-sample
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

Apply the YAML:

```
kubectl apply -f replicationcontroller.yaml
```

Verify the RC:

```
kubectl get rc

kubectl get pods
```

2. Kubernetes service types (ClusterIP, NodePort, LoadBalancer).
Answer:

Create a file `clusterip-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  name: clusterip-service
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Apply the file:

```
kubectl apply -f clusterip-service.yaml
```

Access from within the cluster (via pod):

```
curl http://clusterip-service
```

3. PersistentVolume (PV) and PersistentVolumeClaim (PVC).
Answer:

1. PersistentVolume (PV):

Create `pv.yaml`:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sample
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:                # Use NFS/GCE/EBS etc. in real clusters
    path: "/mnt/data"
  persistentVolumeReclaimPolicy: Retain
```

Apply the PV:

```
kubectl apply -f pv.yaml
```

Check status:

```
kubectl get pv
```

2. PersistentVolumeClaim (PVC)

Create `pod-using-pvc.yaml`:

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

```
metadata:
  name: pod-pvc
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: storage
  volumes:
    - name: storage
      persistentVolumeClaim:
        claimName: pvc-sample
```

Apply the pod:

```
kubectl apply -f pod-using-pvc.yaml
```

4. Managing Kubernetes with Azure Kubernetes Service (AKS), Creating and managing AKS clusters, Scaling and upgrading AKS clusters.
Answer:

Creating AKS Cluster

```
az login

az provider register --namespace Microsoft.ContainerService

az group create --name myResourceGroup --location eastus
```

Create AKS cluster:

```
az aks create \
  --resource-group myResourceGroup \
  --name myAKSCluster \
  --node-count 3 \
  --enable-addons monitoring \
  --generate-ssh-keys
```

Connect to the cluster:

```
az aks get-credentials --resource-group myResourceGroup --name
myAKSCluster
```

Test the connection:

```
kubectl get nodes
```

Managing AKS Clusters

```
az aks show --resource-group myResourceGroup --name myAKSCluster

az aks enable-addons --addons monitoring --name myAKSCluster
--resource-group myResourceGroup
```

Delete a cluster:

```
az aks delete --resource-group myResourceGroup --name myAKSCluster
--yes --no-wait
```

Get Kubernetes dashboard (optional for UI):

```
az aks browse --resource-group myResourceGroup --name myAKSCluster
```

Scaling AKS Clusters

```
az aks scale --resource-group myResourceGroup --name myAKSCluster
--node-count 5
```

```
az aks update \
  --resource-group myResourceGroup \
  --name myAKSCluster \
  --enable-cluster-autoscaler \
  --min-count 1 \
  --max-count 5
```

5. Configure liveliness and readiness probes for pods in AKS cluster.
Answer:

Create YAML file `probes-pod.yaml`:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-probe
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
      livenessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 10
        periodSeconds: 5
      readinessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 5
```

Deploy the Pod

```
kubectl apply -f probes-pod.yaml
```

Verify Probes

```
kubectl describe pod nginx-probe
```

6. Configure Taints and Tolerants.
Answer:

File: `toleration-pod.yaml`

```
apiVersion: v1
kind: Pod
metadata:
  name: toleration-pod
spec:
  tolerations:
    - key: "app"
      operator: "Equal"
      value: "true"
      effect: "NoSchedule"
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
```

Apply the Pod

```
kubectl apply -f toleration-pod.yaml
```

Remove Taint from Node

```
kubectl taint nodes node1 key=app:NoSchedule-
```