**Kubernetes Assignment: ReplicaSet, ReplicationController, Deployment**

**✅ Prerequisites**

* Minikube or Kubernetes cluster running
* kubectl installed and configured

**1️⃣ ReplicationController (RC)**

**📄 RC Definition**

yaml

CopyEdit

apiVersion: v1

kind: ReplicationController

metadata:

name: rc-nginx

spec:

replicas: 2

selector:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

**📌 Commands**

bash

CopyEdit

kubectl apply -f rc.yaml

kubectl get rc

kubectl get pods

**2️⃣ ReplicaSet (RS)**

**📄 RS Definition**

yaml

CopyEdit

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: rs-nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx-rs

template:

metadata:

labels:

app: nginx-rs

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

**📌 Commands**

bash

CopyEdit

kubectl apply -f rs.yaml

kubectl get rs

kubectl get pods

**3️⃣ Deployment**

**📄 Deployment Definition**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 4

selector:

matchLabels:

app: nginx-deploy

template:

metadata:

labels:

app: nginx-deploy

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

**📌 Commands**

bash

CopyEdit

kubectl apply -f deployment.yaml

kubectl get deployments

kubectl get rs

kubectl get pods

**📊 Comparison Table**

| **Feature** | **ReplicationController** | **ReplicaSet** | **Deployment** |
| --- | --- | --- | --- |
| YAML Version | v1 | apps/v1 | apps/v1 |
| Supports Set-Based Selector | ❌ | ✅ | ✅ |
| Rollback Support | ❌ | ❌ | ✅ |
| Rolling Update Support | ❌ | ❌ | ✅ |
| Used Today | Deprecated (legacy) | Sometimes | ✅ Standard |

**✅ Advantages and Disadvantages**

**🔹 ReplicationController**

**Advantages:**

* Ensures pod availability
* Legacy feature, still supported

**Disadvantages:**

* No support for rolling updates
* Does not support set-based selectors
* Deprecated in favor of ReplicaSet/Deployment

**🔹 ReplicaSet**

**Advantages:**

* Supports set-based selectors
* Ensures pod availability

**Disadvantages:**

* Doesn’t support rolling updates
* Mostly managed by Deployment (used internally)

**🔹 Deployment**

**Advantages:**

* Manages ReplicaSets for scaling and updates
* Supports rolling updates, rollbacks
* Widely used and preferred for stateless apps

**Disadvantages:**

* Slightly more complex
* Overhead of managing RS under the hood

**📎 Bonus: Delete Resources**

bash

CopyEdit

kubectl delete -f rc.yaml

kubectl delete -f rs.yaml

kubectl delete -f deployment.yaml

## 📘 Kubernetes Assignment: Service Types

## 🧠 What is a Kubernetes Service?

A **Service** in Kubernetes is an abstraction that defines a logical set of Pods and a policy by which to access them — usually through a **stable IP and DNS**.

## ✅ 1. ****ClusterIP Service (Default)****

* Exposes the service **internally** within the cluster only.
* Not accessible from outside.

### 📄 YAML: clusterip-service.yaml

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: clusterip-nginx

spec:

type: ClusterIP

selector:

app: nginx

ports:

- port: 80

targetPort: 80

### 📌 Commands:

bash

CopyEdit

kubectl apply -f clusterip-service.yaml

kubectl get svc

📝 **Note**: Access only from within the cluster (curl from a Pod).

## ✅ 2. ****NodePort Service****

* Exposes the service **on each Node’s IP** at a static port.
* Accessible externally using NodeIP:NodePort.

### 📄 YAML: nodeport-service.yaml

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: nodeport-nginx

spec:

type: NodePort

selector:

app: nginx

ports:

- port: 80

targetPort: 80

nodePort: 30080

### 📌 Commands:

bash

CopyEdit

kubectl apply -f nodeport-service.yaml

kubectl get svc

📝 Access using:

cpp

CopyEdit

http://<NodeIP>:30080

## ✅ 3. ****LoadBalancer Service****

* Creates an **external load balancer** in supported cloud environments.
* Gives a **public IP** to access the service.

### 📄 YAML: loadbalancer-service.yaml

yaml

CopyEdit

apiVersion: v1

kind: Service

metadata:

name: loadbalancer-nginx

spec:

type: LoadBalancer

selector:

app: nginx

ports:

- port: 80

targetPort: 80

### 📌 Commands:

bash

CopyEdit

kubectl apply -f loadbalancer-service.yaml

kubectl get svc

📝 In Minikube: use minikube tunnel to simulate LoadBalancer:

bash

CopyEdit

minikube tunnel

Then access via assigned external IP.

## 📊 Comparison Table

| **Feature** | **ClusterIP** | **NodePort** | **LoadBalancer** |
| --- | --- | --- | --- |
| Access Scope | Internal only | External via Node IP+Port | External via public IP |
| Default Type | ✅ | ❌ | ❌ |
| Cloud Provider Needed | ❌ | ❌ | ✅ (or minikube tunnel) |
| Port Configuration | ClusterIP only | NodePort (30000–32767) | Auto or specified |
| Best Use Case | Internal services | Dev/Test, low traffic | Production, cloud load-balancing |

## 🛠 Bonus: Simple Nginx Deployment (used for all services)

### 📄 nginx-deployment.yaml

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

### 📌 Commands:

bash

CopyEdit

kubectl apply -f nginx-deployment.yaml

## 📎 Delete All Services and Deployment

bash

CopyEdit

kubectl delete svc clusterip-nginx nodeport-nginx loadbalancer-nginx

kubectl delete deployment nginx

## 📷 Assignment Deliverables (Screenshots to take):

1. Output of:
   * kubectl get svc
   * kubectl describe svc <service-name>
   * Accessing each service (where possible) from browser or curl
2. Minikube users: show minikube tunnel terminal while using LoadBalancer

Would you like me to give you a downloadable .zip with all these YAML files?

##### You said:

PersistentVolume (PV) and PersistentVolumeClaim (PVC) complete this assignement

###### ChatGPT said:

Here is your **complete Kubernetes assignment** on **PersistentVolume (PV)** and **PersistentVolumeClaim (PVC)** — including definitions, YAML examples, commands, and advantages/disadvantages.

## 📘 Kubernetes Assignment: PersistentVolume (PV) & PersistentVolumeClaim (PVC)

## 🧠 What are PV and PVC?

* **PersistentVolume (PV):** A piece of storage in the cluster provisioned by an admin or dynamically provisioned.
* **PersistentVolumeClaim (PVC):** A request for storage by a user.

✅ PV = supply (actual storage)  
✅ PVC = demand (user's request for storage)

## ✅ 1. Create a PersistentVolume (PV)

### 📄 pv.yaml

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolume

metadata:

name: my-pv

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: /mnt/data

### 📌 Commands:

bash

CopyEdit

kubectl apply -f pv.yaml

kubectl get pv

## ✅ 2. Create a PersistentVolumeClaim (PVC)

### 📄 pvc.yaml

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

### 📌 Commands:

bash

CopyEdit

kubectl apply -f pvc.yaml

kubectl get pvc

kubectl describe pvc my-pvc

kubectl get pv

✅ PV will now be bound to PVC if both match (accessModes, storage)

## ✅ 3. Use PVC in a Pod

### 📄 pod-using-pvc.yaml

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: pod-pvc-demo

spec:

containers:

- name: app

image: nginx

volumeMounts:

- mountPath: "/usr/share/nginx/html"

name: storage

volumes:

- name: storage

persistentVolumeClaim:

claimName: my-pvc

### 📌 Commands:

bash

CopyEdit

kubectl apply -f pod-using-pvc.yaml

kubectl get pod

kubectl describe pod pod-pvc-demo

## 🧹 Cleanup Commands

bash

CopyEdit

kubectl delete -f pod-using-pvc.yaml

kubectl delete -f pvc.yaml

kubectl delete -f pv.yaml

## 📊 Advantages and Disadvantages

### 🔹 PersistentVolume (PV)

**Advantages:**

* Decouples storage from pods
* Can be reused across workloads
* Supports multiple types of storage (hostPath, NFS, cloud disks)

**Disadvantages:**

* Static provisioning requires admin setup
* Must match PVC exactly (access mode, size)

### 🔹 PersistentVolumeClaim (PVC)

**Advantages:**

* Users can request storage without knowing backend
* Enables dynamic provisioning with StorageClasses
* Abstracts complex storage management

**Disadvantages:**

* PVCs get bound and can’t switch PVs easily
* Needs careful resource planning

## 📎 Bonus (Optional): Verify Data Persistence

You can exec into the pod and add a file:

bash

CopyEdit

kubectl exec -it pod-pvc-demo -- /bin/bash

echo "Hello from PVC!" > /usr/share/nginx/html/index.html

Access the pod with NodePort service to verify from browser if desired.

## Assignment: Configure Taints and Tolerations in Kubernetes

## 🧠 What are Taints and Tolerations?

* **Taint**: Applied to a **node** to **repel** pods unless those pods tolerate the taint.
* **Toleration**: Applied to a **pod** to **allow it to be scheduled** on tainted nodes.

Think of taints as **"do not enter unless allowed"** signs on nodes.

## ✅ Step-by-Step Guide

### ✅ 1. ****Add a Taint to a Node****

#### 📌 Command:

bash

CopyEdit

kubectl taint nodes <node-name> key=value:NoSchedule

🔹 Example:

bash

CopyEdit

kubectl taint nodes node01 env=dev:NoSchedule

📌 This taint prevents pods from being scheduled on node01 unless they have a matching toleration.

### ✅ 2. ****Deploy a Pod Without Toleration****

### 📄 no-toleration-pod.yaml

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: test-no-toleration

spec:

containers:

- name: nginx

image: nginx

#### 📌 Apply:

bash

CopyEdit

kubectl apply -f no-toleration-pod.yaml

kubectl describe pod test-no-toleration

🧠 The pod will stay in **Pending** state if no other node is available without taint.

### ✅ 3. ****Deploy a Pod WITH Toleration****

### 📄 toleration-pod.yaml

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: test-with-toleration

spec:

containers:

- name: nginx

image: nginx

tolerations:

- key: "env"

operator: "Equal"

value: "dev"

effect: "NoSchedule"

#### 📌 Apply:

bash

CopyEdit

kubectl apply -f toleration-pod.yaml

kubectl describe pod test-with-toleration

✅ This pod will be allowed on the tainted node.

### 🧹 Remove Taint (Optional)

bash

CopyEdit

kubectl taint nodes node01 env=dev:NoSchedule-

## 📊 Summary Table

| **Component** | **Role** |
| --- | --- |
| Taint | Added to node to prevent undesired pods |
| Toleration | Added to pod to allow it on tainted node |
| Effect Types | NoSchedule, PreferNoSchedule, NoExecute |

**Assignment: Create and Attach Persistent Volume Claims (PVC) to Pods**

**🧠 Objective**

Learn how to:

* Create a Persistent Volume (PV)
* Create a Persistent Volume Claim (PVC)
* Attach the PVC to a pod (e.g., Nginx)
* Store persistent data inside a pod

**✅ Step 1: Create a Persistent Volume (PV)**

**📄 pv.yaml**

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolume

metadata:

name: my-pv

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Retain

hostPath:

path: "/mnt/data"

**📌 Apply:**

bash

CopyEdit

kubectl apply -f pv.yaml

kubectl get pv

**✅ Step 2: Create a Persistent Volume Claim (PVC)**

**📄 pvc.yaml**

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

**📌 Apply:**

bash

CopyEdit

kubectl apply -f pvc.yaml

kubectl get pvc

kubectl describe pvc my-pvc

**✅ Step 3: Create a Pod that Uses the PVC**

**📄 pod-using-pvc.yaml**

yaml

CopyEdit

apiVersion: v1

kind: Pod

metadata:

name: pod-with-pvc

spec:

containers:

- name: nginx

image: nginx

volumeMounts:

- mountPath: "/usr/share/nginx/html"

name: storage

volumes:

- name: storage

persistentVolumeClaim:

claimName: my-pvc

**📌 Apply:**

bash

CopyEdit

kubectl apply -f pod-using-pvc.yaml

kubectl get pod

kubectl describe pod pod-with-pvc

**✅ Step 4: Verify Data is Stored**

bash

CopyEdit

kubectl exec -it pod-with-pvc -- /bin/bash

echo "Hello from PVC" > /usr/share/nginx/html/index.html

cat /usr/share/nginx/html/index.html

🔁 Restart pod or delete and recreate it. Data will still be there (if ReclaimPolicy is Retain and PVC reused).

**🧹 Cleanup (Optional)**

bash

CopyEdit

kubectl delete -f pod-using-pvc.yaml

kubectl delete -f pvc.yaml

kubectl delete -f pv.yaml

**📷 Suggested Screenshots for Assignment**

1. kubectl get pv, kubectl get pvc, kubectl describe pvc
2. Pod status: kubectl get pod, describe pod
3. Terminal: kubectl exec output showing written data
4. Optional: Nginx browser access via NodePort service

**📊 Summary**

| **Component** | **Description** |
| --- | --- |
| PersistentVolume (PV) | Physical storage (1Gi hostPath) |
| PersistentVolumeClaim (PVC) | Claim made by a pod (1Gi) |
| Pod with PVC | Mounts the PVC at a specific path |

## Assignment: Configure Health Probes (Liveness and Readiness) for Pods in Kubernetes

## 🧠 What Are Health Probes?

**Kubernetes uses health probes to check the status of containers**:

| **Probe Type** | **Purpose** | **Action if Fails** |
| --- | --- | --- |
| **Liveness** | Checks if the container is alive | Restarts the container |
| **Readiness** | Checks if the container is ready to receive traffic | Removes pod from service endpoints |

## ✅ Step-by-Step Setup

### ✅ Step 1: Create Deployment with Probes

### 📄 nginx-probes.yaml

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-probe-demo

spec:

replicas: 2

selector:

matchLabels:

app: nginx-probe

template:

metadata:

labels:

app: nginx-probe

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

readinessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 5

periodSeconds: 10

livenessProbe:

httpGet:

path: /

port: 80

initialDelaySeconds: 15

periodSeconds: 20

### ✅ Step 2: Apply Deployment

bash

CopyEdit

kubectl apply -f nginx-probes.yaml

kubectl get pods

### ✅ Step 3: Describe and Verify Probes

bash

CopyEdit

kubectl describe pod <pod-name>

Look for Liveness and Readiness under Conditions and Events.

## 🔁 Simulate a Failure (Optional)

You can manually simulate a probe failure to observe behavior:

bash

CopyEdit

kubectl exec -it <pod-name> -- /bin/bash

mv /usr/share/nginx/html/index.html /tmp/index.html

Now the / path will fail the HTTP GET check.

After a few seconds:

bash

CopyEdit

kubectl get pods # You will see restart count increase

## 📊 Use Cases

| **Application Type** | **Liveness Required** | **Readiness Required** |
| --- | --- | --- |
| Fast startup, reliable | ✅ | ✅ |
| Long initialization | ❌ | ✅ |
| Crash-prone apps | ✅ | ❌ |

**Assignment: Configure Autoscaling (Horizontal Pod Autoscaling) in Kubernetes**

**🧠 What is Horizontal Pod Autoscaler (HPA)?**

* **HPA** automatically increases or decreases the **number of pods** in a deployment or replica set based on observed metrics like **CPU utilization**.
* It helps your application handle variable workloads efficiently.

**✅ Prerequisites**

* Kubernetes cluster or Minikube
* Metrics server installed (required for HPA)

**📌 To install metrics server (Minikube):**

bash

CopyEdit

minikube addons enable metrics-server

Verify it:

bash

CopyEdit

kubectl top nodes

kubectl top pods

**✅ Step 1: Create a Deployment**

**📄 hpa-deployment.yaml**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: php-apache

spec:

replicas: 1

selector:

matchLabels:

app: php-apache

template:

metadata:

labels:

app: php-apache

spec:

containers:

- name: php-apache

image: k8s.gcr.io/hpa-example

resources:

requests:

cpu: 200m

limits:

cpu: 500m

ports:

- containerPort: 80

**📌 Apply Deployment:**

bash

CopyEdit

kubectl apply -f hpa-deployment.yaml

kubectl get deployments

**✅ Step 2: Create the HPA Resource**

**📌 Command:**

bash

CopyEdit

kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=5

This means: scale between **1 to 5** pods if average CPU > 50%.

**📌 Verify HPA:**

bash

CopyEdit

kubectl get hpa

kubectl describe hpa php-apache

**✅ Step 3: Simulate Load to Trigger Scaling**

**📌 Create a load generator pod:**

bash

CopyEdit

kubectl run -i --tty load-generator --image=busybox /bin/sh

Inside the shell:

bash

CopyEdit

while true; do wget -q -O- http://php-apache; done

Wait a few minutes and observe:

bash

CopyEdit

kubectl get hpa

kubectl get pods

You will see the **pod count increase** automatically!

**📷 Suggested Screenshots**

1. kubectl get hpa
2. kubectl describe hpa php-apache
3. kubectl get pods showing increased replica count
4. Optional: kubectl top pods and load-generator in action

**🧹 Cleanup (Optional)**

bash

CopyEdit

kubectl delete deployment php-apache

kubectl delete hpa php-apache

kubectl delete pod load-generator

**📊 Summary Table**

| **Component** | **Purpose** |
| --- | --- |
| Deployment | Runs the app with resource requests |
| Metrics Server | Enables HPA to collect CPU usage |
| HPA Resource | Automatically adjusts pod count |
| Load Generator | Used to simulate CPU load |