Assignment: Deploying and Managing a Scalable Web Application Using Kubernetes

---

Assignment Requirements

1. Setup Kubernetes Cluster

- Use **Minikube** (for local deployment) or **Google Kubernetes Engine (GKE), Amazon EKS, or Azure AKS** for cloud-based deployment.
- Ensure the cluster is up and running with at least **two worker nodes**.

2. Deploy a Web Application

- Use a simple **Node.js or Python Flask-based application** (or any web app of your choice).
- Containerize the application using **Docker**.
- Push the container image to **Docker Hub or a private container registry**.

3. Create Kubernetes Resources

- **Deployments:** Deploy the web application using a Kubernetes **Deployment** with at least **3 replicas**.
- **Services:** Create a **Service** (NodePort or LoadBalancer) to expose the application.
- **ConfigMaps & Secrets:** Store environment variables (e.g., database connection string) securely using **ConfigMaps and Secrets**.

4. Implement Auto-scaling

- Configure **Horizontal Pod Autoscaler (HPA)** to scale pods based on CPU utilization.
- Set **minimum 2 pods** and **maximum 5 pods**, scaling up when CPU usage exceeds 50%.

5. Implement Persistent Storage (Optional)

- If the application stores data, use **Persistent Volume (PV) and Persistent Volume Claim (PVC)**.
- Mount the volume in the pod for persistent storage.

6. Rolling Updates & Rollbacks

- Simulate a **rolling update** by deploying a new version of the application.
- Perform a **rollback** in case of a failure.

7. Logging

- Use **kubectl logs** to view application logs.

Testing Scenarios

To ensure the Kubernetes deployment is working as expected, perform the following test cases:

1. Application Availability Tests

✅ **Test:** Check if the application is accessible via the Kubernetes service.
🔹 **Command:**

```
kubectl get services
curl http://<EXTERNAL-IP>:<PORT>
```

🔹 **Expected Output:** Should return the homepage or API response of the application.

---

2. Scaling Tests

✅ **Test:** Trigger high CPU usage to see if the **Horizontal Pod Autoscaler (HPA)** scales up pods.
🔹 **Command:**

```
kubectl get hpa
kubectl run --rm -it --image=busybox stress-test -- /bin/sh
```

Inside BusyBox shell:

```
while true; do wget -q -O- http://<SERVICE-IP>:<PORT>; done
```

🔹 **Expected Output:** Number of pods should increase dynamically.
🔹 **Verification:**

```
kubectl get pods -w
```

---

3. Rolling Update & Rollback Test

✅ **Test:** Perform a rolling update and verify zero downtime.
🔹 **Command:**

```
kubectl set image deployment/<DEPLOYMENT_NAME> <CONTAINER_NAME>=new-image:v2
```

🔹 **Expected Output:** New version is deployed while keeping the app running.

✅ **Test:** Rollback to the previous version in case of failure.
🔹 **Command:**

```
kubectl rollout undo deployment/<DEPLOYMENT_NAME>
```

🔹 **Expected Output:** Application reverts to the previous working version.

---

4. Pod Failure and Self-Healing Test

✅ **Test:** Manually delete a pod and check if Kubernetes automatically recreates it.
🔹 **Command:**

```
kubectl delete pod <POD_NAME>
```

🔹 **Expected Output:** A new pod should be automatically created.
🔹 **Verification:**

```
kubectl get pods -w
```

---

5. Persistent Storage Test (If Implemented)

✅ **Test:** Verify if data persists after pod restart.
🔹 **Steps:**

- Store data in the mounted volume inside the pod.
- Delete the pod and check if data is retained.

🔹 **Command:**

```
kubectl delete pod <POD_NAME>
kubectl get pods -w
```

🔹 **Expected Output:** New pod should start with the same data.

---

6. Logging Test

✅ **Test:** Check if application logs are available.
🔹 **Command:**

```
kubectl logs <POD_NAME>
```

🔹 **Expected Output:** Should display application logs.

Deliverables

- **GitHub Repository** containing:
  - `Dockerfile` for containerizing the application.
  - Kubernetes YAML manifests (`deployment.yaml`, `service.yaml`, `hpa.yaml`, etc.).
  - Step-by-step **README.md** with setup instructions.
- A **short demo video (3-5 minutes)** explaining the implementation.
- Screenshots of the running cluster and auto-scaling in action.
- Test Cases for all the 6 Tests done above.