All document and yaml files related to the task are stored in GitHub

Link: https://github.com/vilaspatel/Task

Task 1: Create a deployment running nginx version 1.16.1 that will run in 2 pods that uses /var/www as its root dir. Nginx should run on port 8090

The K8s deployment files are stored under Task1 folder, run the following command to deploy the Solution

```
kubectl apply -f configmap.yaml
kubectl apply -f service.yaml
kubectl apply -f nginx-deployment.yaml --record
```

Note: to Change the Root directory and port for Nginx we have to pass a configuration in this case we are creating a ConfigMap "nginx-conf" to change the default settings

1. **Create a separate pod that mounts a volume. The volume should have a dir /var/www and contain a file "Hello World" html. This Pod should load before Nginx (you need to prove this)**

   Init Container will copy the index.html file from configmap mapped as volume at location /var to /workdir location, the /workdir is mapped to the Nginx Container location /var/www

```yaml
initContainers:
- name: busybox
  image: busybox
  command:
  - cp
  - "/var/index.html"
  - "/workdir/index.html"
  volumeMounts:
    - name: web-content
      mountPath: /var/index.html
      subPath: index.html
      readOnly: true
    - name: workdir
      mountPath: /workdir
containers:
- name: nginx
  image: nginx:1.16.1
  ports:
  - containerPort: 8090
  volumeMounts:
      - name: nginx-conf
        mountPath: /etc/nginx/conf.d/nginx.conf
        subPath: nginx.conf
        readOnly: true
      - name: workdir
        mountPath: /var/www
```

2. **Create service Load Balancer for Nginx**

   service.yaml creates a Service with type Load Balancer

3. **Make Nginx accessible via the internet on port 80**

   service.yaml creates a Service with type LoadBalancer which expose the application on port 80 on Load Balancer IP

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 8090
  type: LoadBalancer
```

4. **Scale this to 4 pods.**

   Below Command will scale the deployment to 4 pods

   ```
   kubectl scale deployment nginx --replicas=4
   ```

5. **Create rolling update to 1.19.4**

   Below Command will update the nginx version from 1.16.1 to 1.19.4

   ```
   kubectl set image deployment/nginx nginx=nginx:1.19.4 --record
   ```

6. **Check the status of the upgrade**

   Using below Command we can check upgrade status

   ```
   kubectl rollout status deployment/nginx
   ```

7. **How do you do this in a way that you can see history of what happened?**

   When issuing the command with " –record" it records the command/action which caused the deployment version change

8. **How would you do a Canary Upgrade?**

   We can have two deployment running with two versions of application which is been load balances by a single Service, at first we can start Canary deployment by 4:1 pod ratio where App v1 will have 90% traffic forwarded and App v2 with 10%, we can slowly change the Pod ratio 1:1 to have 50% traffic forwarded to each App version . And then complete change the ration to 0:1 where all traffic will be forwarded App v2

   To perform Canary deployment files are storage in Tasl1/Canary_deployment/

   ```
   kubectl apply -f configmap.yaml
   kubectl apply -f service.yaml
   kubectl apply -f nginx-deployment-v1.yaml –record
   kubectl apply -f nginx-deployment-v2.yaml –record
   ```

   To scale up the v2 app and scale down v1app use the below command

   ```
   kubectl scale deployment nginx-v1 --replicas=3
   kubectl scale deployment nginx-v2 --replicas=2
   ```

9. Undo the upgrade

   To Roll Back to the previous version use the below command

   ```
   kubectl rollout undo deployment nginx
   ```

10. Scale down to 2 pods

    Below Command will scale the deployment to 2 pods

    ```
    kubectl scale deployment nginx --replicas=2
    ```

1) Do not use Helm
2) Submit all yaml and config files
3) Submit all kubectl commands
4) Document your work in a way that anyone could repeat the results using your documentation

Task 2: Write a very simple python program, that would expose itself to port 80 with HELLO HARMAN message. Then, write yaml manifest for it, deploy it to K8S and expose it using nginx ingress.

All files related to this id stored under Task2 folder

To deploy the solution use the below command

```
kubectl apply -f python-deployment.yaml --record
```

Task 3:deploy a WordPress frontend with a MySql backend using PV/PVC and daemon sets

All files related to this id stored under Task3 folder

To deploy the solution use the below command

```
kubectl apply -f PV.yaml
kubectl apply -f mysql-daemonset.yaml –record
kubectl apply -f wordpress-deployment.yaml yaml --record
```