Assignment 2

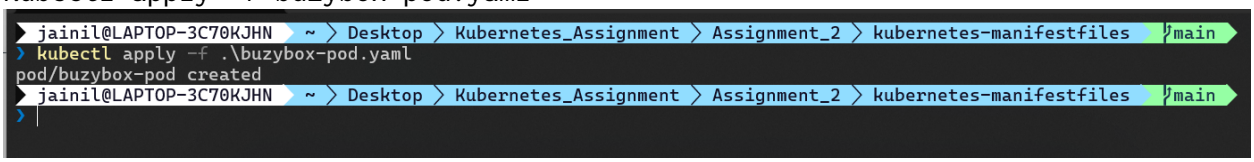
1. **You are tasked with creating a simple pod in your Kubernetes cluster. The pod should run a container using the busybox image.**
2. **Change the image name from busybox to nginx, also check that pod is running well.**
3. **Create a ReplicaSet named "app-replicaset" managing three replicas of an application pod using the nginx:v1 image.**
4. **Create a Deployment named "app-deployment" managing four replicas of an application pod using the nginx:alpine image.**
5. **Explain how to automatically roll back to the previous version using the "app-deployment."**
6. **Describe the differences between a ClusterIP service and a LoadBalancer service, providing a use case for each**

Steps to create a simple pod in Kubernetes cluster. Pod run a container using the buzybox image:

1. Let's create a new file named busybox-pod.yaml. And enter the following details.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: buzybox-pod
  labels:
    name: buzybox-pod
spec:
  containers:
  - name: buzybox-container
    image: buzybox:1.36
    resources:
      limits:
        memory: "128Mi"
        cpu: "500m"
    command: ["sleep", "3600"]
```

2. Then to apply it run the following command:
   `kubectl apply -f buzybox-pod.yaml`



```
jainil@LAPTOP-3C70KJHN ~ > Desktop > Kubernetes_Assignment > Assignment_2 > kubernetes-manifestfiles main
> kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
buzybox-pod   1/1     Running   0          14s
jainil@LAPTOP-3C70KJHN ~ > Desktop > Kubernetes_Assignment > Assignment_2 > kubernetes-manifestfiles main
>
```

Steps to Change the image name from busybox to nginx, also check that pod is running well.:

1. Change the image to nginx and names as well:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    name: nginx-pod
spec:
  containers:
  - name: nginx-container
    image: nginx:latest
    resources:
      limits:
        memory: "128Mi"
        cpu: "500m"
    ports:
      - containerPort: 80
```

2. Then apply it using the following command:

```
jainil@LAPTOP-3C70KJHN ~ > Desktop > Kubernetes_Assignment > Assignment_2 > kubernetes-manifestfiles main
> kubectl apply -f .\buzybox-pod.yaml
pod/nginx-pod created
jainil@LAPTOP-3C70KJHN ~ > Desktop > Kubernetes_Assignment > Assignment_2 > kubernetes-manifestfiles main
> s
```

```
jainil@LAPTOP-3C70KJHN ~ > Desktop > Kubernetes_Assignment > Assignment_2 > kubernetes-manifestfiles main
> kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          107s
jainil@LAPTOP-3C70KJHN ~ > Desktop > Kubernetes_Assignment > Assignment_2 > kubernetes-manifestfiles main
>
```

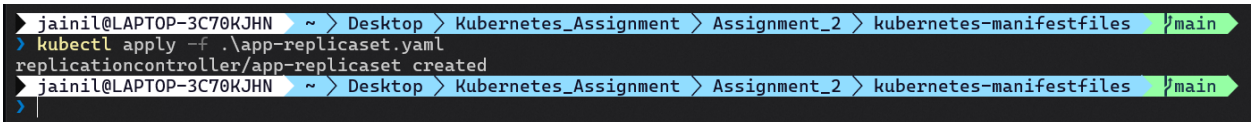
Steps to create a ReplicaSet named "app-replicaset" managing three replicas of an application pod using the nginx:v1 image.:

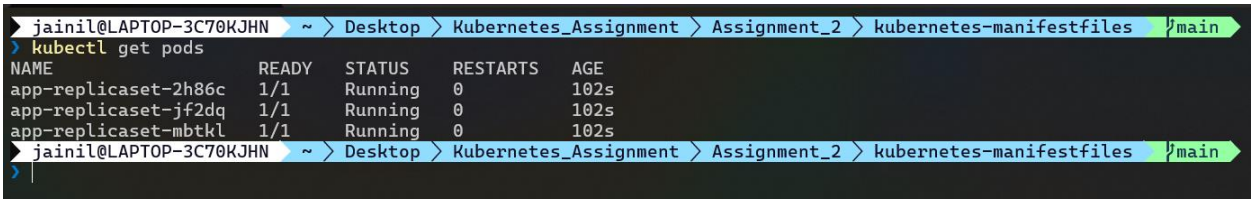
1. Create a new file name app-replicaset.yaml and enter the following:

```yaml
apiVersion: v1
kind: ReplicationController
```

```yaml
metadata:
  name: app-replicaset
spec:
  replicas: 3
  selector:
    app: app-replicaset
  template:
    metadata:
      name: app-replicaset
      labels:
        app: app-replicaset
    spec:
      containers:
        - name: nginx-contiainer
          image: nginx:latest
          ports:
            - containerPort: 80
```

2. Then run the following code to apply the app-replicaset.yaml file:
   `kubectl apply -f app-replicaset.yaml`





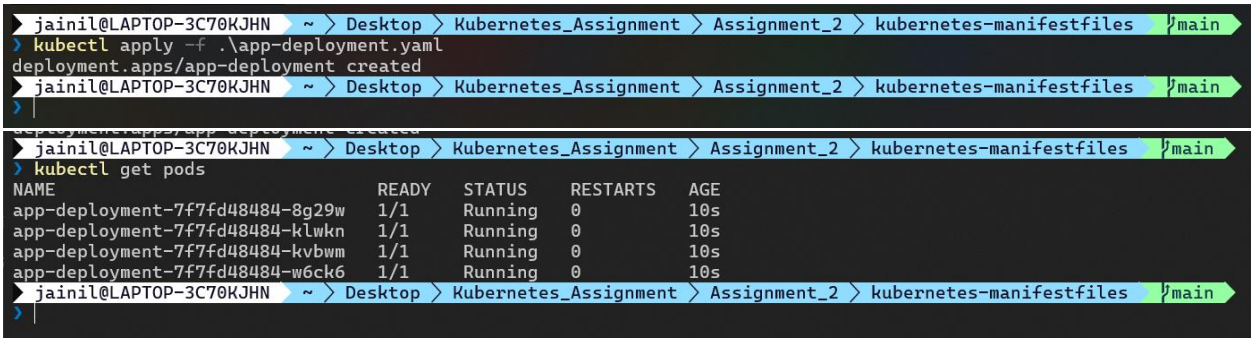
Steps to Create a Deployment named "app-deployment" managing four replicas of an application pod using the nginx:alpine image.:

1. Create a new file named app-deployment.yaml and enter the following:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
spec:
  replicas: 4
  selector:
```

```yaml
    matchLabels:
      app: app-deployment
  template:
    metadata:
      labels:
        app: app-deployment
    spec:
      containers:
      - name: nginx-container
        image: nginx:alpine
        resources:
          limits:
            memory: "128Mi"
            cpu: "500m"
        ports:
        - containerPort: 80
```

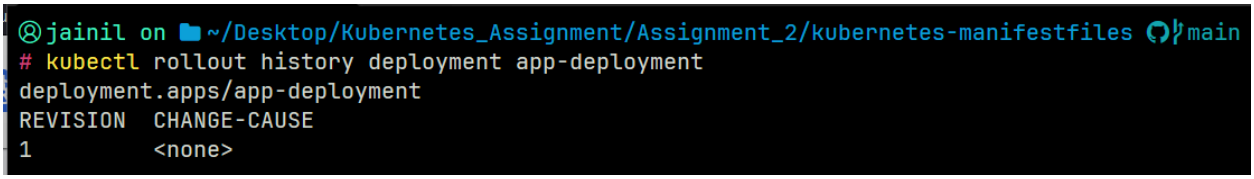
2. Then run the following code to apply the app-deployment.yaml file:



Explain how to automatically roll back to the previous version using the "app-deployment."

1. First let's check the history for the app-deployment, for that use the following command:

   `kubectl rollout history deployment app-deployment`



2. Now let's change app-deployment.yaml file and change the image: nginx:latest.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
```

```yaml
  name: app-deployment
spec:
  replicas: 4
  selector:
    matchLabels:
      app: app-deployment
  template:
    metadata:
      labels:
        app: app-deployment
    spec:
      containers:
      - name: nginx-container
        image: nginx:latest
        resources:
          limits:
            memory: "128Mi"
            cpu: "500m"
        ports:
        - containerPort: 80
```

3. Then apply the changes using the command:
   `kubectl apply -f app-deployment.yaml`

```
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
# kubectl apply -f .\app-deployment.yaml
deployment.apps/app-deployment configured
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
#
```

4. Let's check the pods and their configuration deployed in the previous step.
   `kubectl get pods`

```
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
app-deployment-6944fdc788-4zw2r   1/1     Running   0          50s
app-deployment-6944fdc788-89w6j   1/1     Running   0          54s
app-deployment-6944fdc788-d7b9p   1/1     Running   0          50s
app-deployment-6944fdc788-mcc2l   1/1     Running   0          54s
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
#
```

   `kubectl describe pods`

```
Containers:
  nginx-container:
    Container ID:   docker://c51fbbb49919595ed0eb0db933ea4a6b803143717a216e6a95d9dfa7c141bd81
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:c26ae7472d624ba1fafd296e73cecc4f93f853088e6a9c13c0d52f6ca5865107
    Port:           80/TCP
```

5. Now lets check the history, We can see two revisions:

kubectl rollout history deployment app-deployment

```
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
# kubectl rollout history deployment app-deployment
deployment.apps/app-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         <none>

jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
```

6. Let's rollback the version of app deployment using the following command:
   kubectl rollout undo deployment app-deployment

```
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
# kubectl rollout undo deployment app-deployment
deployment.apps/app-deployment rolled back
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
#
```

```
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
app-deployment-7f7fd48484-knzd7   1/1     Running   0          10s
app-deployment-7f7fd48484-pzqtk   1/1     Running   0          8s
app-deployment-7f7fd48484-q8kbb   1/1     Running   0          10s
app-deployment-7f7fd48484-z7ngj   1/1     Running   0          8s
jainil on ~/Desktop/Kubernetes_Assignment/Assignment_2/kubernetes-manifestfiles main
#
```

```
Containers:
  nginx-container:
    Container ID:   docker://38303aaf0705513bb30c111e2aed2c811b0a55dcdb60a59daa3580cae8c66a9f
    Image:          nginx:alpine
    Image ID:       docker-pullable://nginx@sha256:6a2f8b28e45c4adea04ec207a251fd4a2df03ddc930f782af51e315ebc76e9a9
    Port:           80/TCP
```

Describe the differences between a ClusterIP service and a LoadBalancer service, providing a use case for each

ClusterIP Service and **LoadBalancer** Service are both Kubernetes service types used for exposing applications running inside the cluster to external clients, but they serve different purposes and have different use cases. Here are the differences and use cases for each:

1. **ClusterIP Service**:
   - **Purpose**: A ClusterIP service exposes the application internally within the cluster. It assigns a stable IP address to the service, which other pods within the same cluster can use to communicate with the application.
   - **Use Case**:
     - **Microservices Communication:** When you have multiple microservices within your Kubernetes cluster that need to communicate with each

other, you can use ClusterIP services to facilitate communication between them.
        - **Database Services:** If you have a database running inside your cluster that needs to be accessed by other applications within the cluster, you can expose it using a ClusterIP service.
2. **LoadBalancer Service**:
    - **Purpose**: A LoadBalancer service exposes the application to external clients outside the Kubernetes cluster. It automatically provisions an external load balancer (like a cloud load balancer) and directs incoming traffic to one of the pods behind the service.
    - **Use Case**:
        - **External Access:** When you have an application running inside your Kubernetes cluster that needs to be accessed by clients outside the cluster, such as end-users or other services located outside the cluster, you can use a LoadBalancer service.
        - **High Availability:** LoadBalancer services provide high availability by distributing incoming traffic across multiple pods. This is useful for ensuring that your application remains available and responsive, even in the face of pod failures or increased load.