Kubernetes

Deployment Strategy

Recreate deployment and Rolling Update deployment

A Kubernetes deployment strategy is a method used to manage and control the rollout and updates of application versions in a cluster to ensure reliability and minimize downtime.

Two types of deployment strategies discussed here are:

- **Recreate**
- **Rolling Update**

Let’s see these two types of deployment strategies in details with hands on examples

A) Recreate deployment:

This strategy terminates all existing pods and creates new pods with the latest version. This strategy is simple and straightforward, but it can cause downtime. It's often used in development environments where downtime is acceptable.

Example:

Create a deployment named app1 with deployment strategy set to the type “recreate”

In a yaml file, define a kubernetes deployment with the required image and the deployment strategy type recreate as given below. Here, custom image “myapp1:v1” from docker repository is used.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: app1
  name: app1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: app1
  strategy:
    type: Recreate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: app1
    spec:
      containers:
      - image: docker.io/vaishnavivyawahare26/myapp:v1
        name: myapp
        resources: {}
status: {}
```

Create deployment using the yaml file

```
controlplane $ kubectl create -f app1.yaml
deployment.apps/app1 created
```

Deployment app1 is successfully created with 3 replicas

```
controlplane $ kubectl get deployments.apps
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
app1   3/3     3            3           24s
controlplane $
```

Pods managed by app1 deployment

```
controlplane $ kubectl get pod
NAME                    READY   STATUS    RESTARTS   AGE
app1-565b4c8bf9-2k9kc   1/1     Running   0          16s
app1-565b4c8bf9-sk6fv   1/1     Running   0          16s
app1-565b4c8bf9-tfwcp   1/1     Running   0          16s
```

When we describe the deployment we may observe that the deployment strategy is recreate and the image is “myapp:v1”

```
controlplane $ kubectl describe deployments.apps app1
Name:                   app1
Namespace:              default
CreationTimestamp:      Thu, 28 Nov 2024 11:48:02 +0000
Labels:                 app=app1
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=app1
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           Recreate
MinReadySeconds:        0
Pod Template:
  Labels:  app=app1
  Containers:
   myapp:
    Image:          docker.io/vaishnavivyawahare26/myapp:v1
```

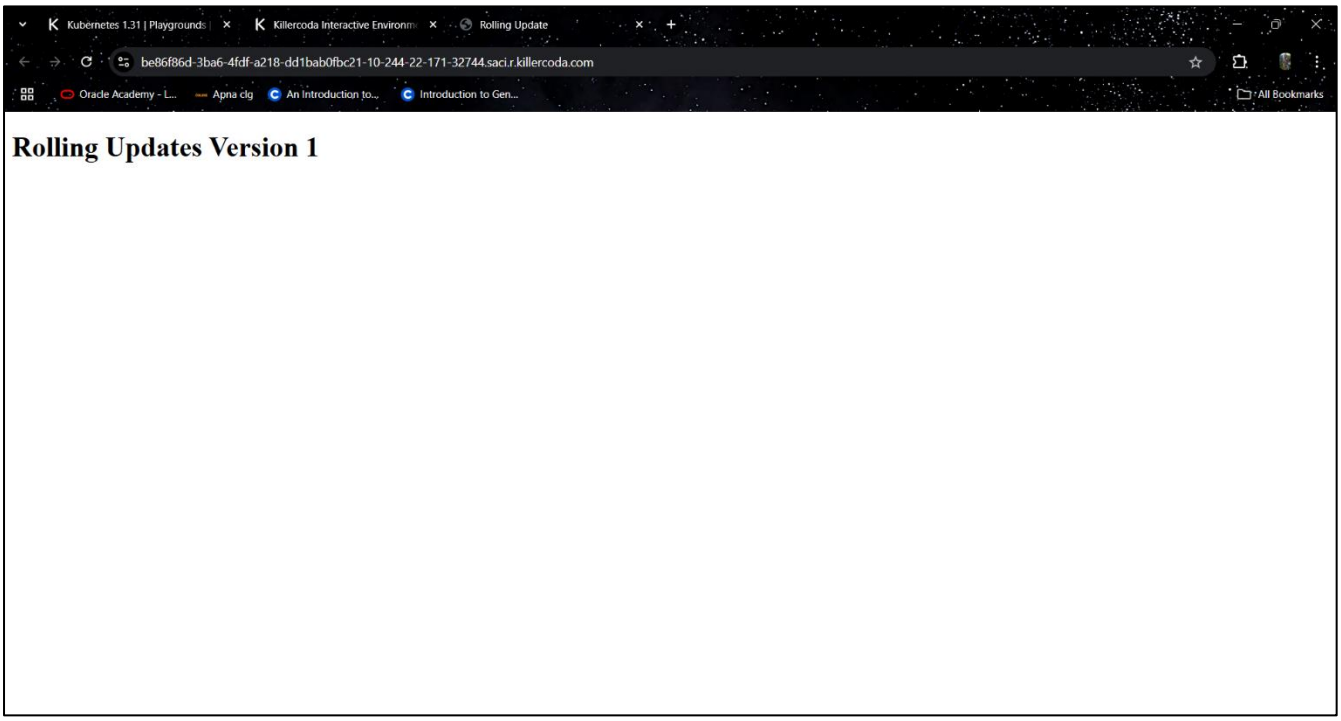
Now, let’s create a NodePort service to make this application accessible outside the cluster

```
controlplane $ kubectl expose deployment app1 --port 80 --target-port 80 --type NodePort
service/app1 exposed
```

NodePort service is successfully created

```
controlplane $ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
app1         NodePort    10.108.99.65   <none>        80:32744/TCP   10s
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        21d
```

Application is now accessible and deploying version 1



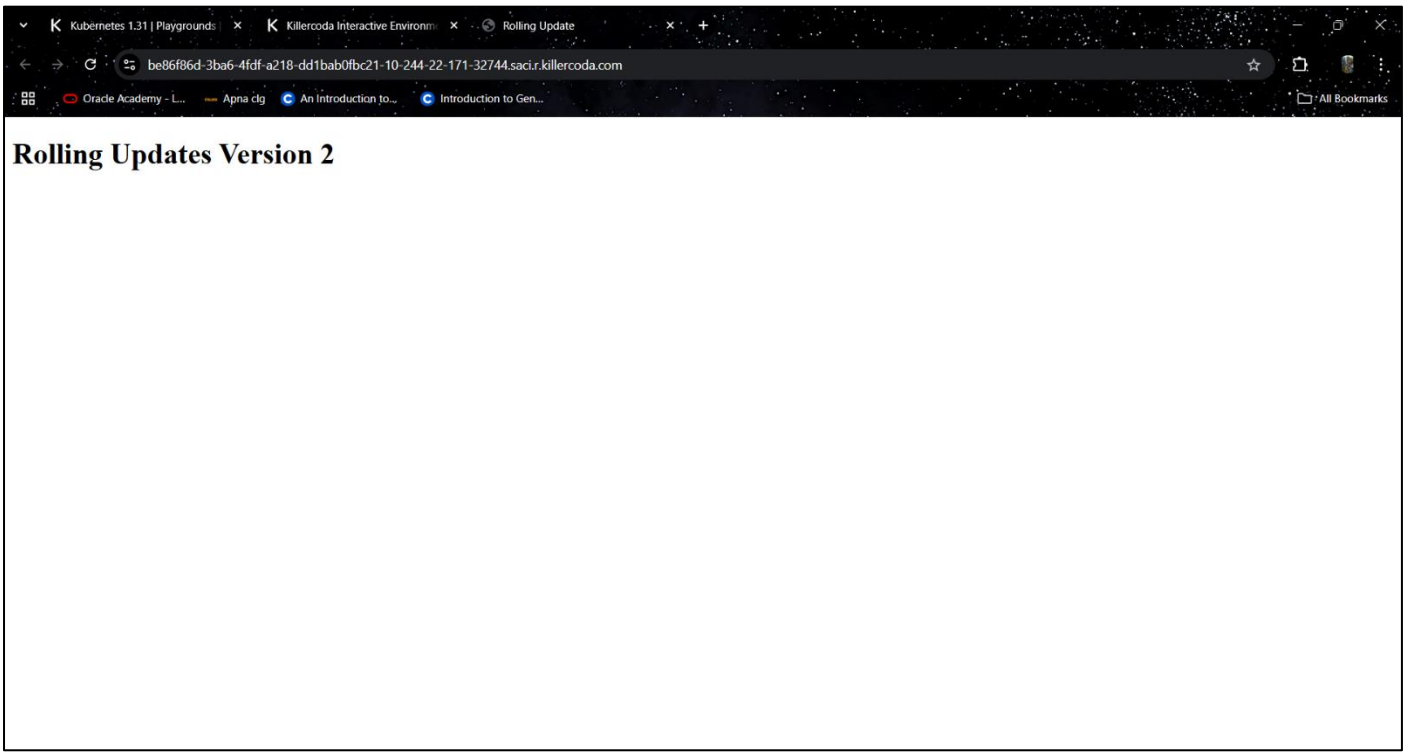
Now, let’s update the deployment from image “myapp:v1” to “myapp:v2”

```
controlplane $ kubectl set image deployments app1 *=docker.io/vaishnavivyawahare26/myapp:v2
deployment.apps/app1 image updated
```

The image is successfully updated to “myapp:v2”

```
Editor  Tab 1  Tab 2  +                                                    38 min
controlplane $ kubectl describe deployments.apps app1
Name:               app1
Namespace:          default
CreationTimestamp:  Thu, 28 Nov 2024 11:48:02 +0000
Labels:             app=app1
Annotations:        deployment.kubernetes.io/revision: 4
Selector:           app=app1
Replicas:           3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:       Recreate
MinReadySeconds:    0
Pod Template:
  Labels:  app=app1
  Containers:
   myapp:
    Image:         docker.io/vaishnavivyawahare26/myapp:v2
    Port:          <none>
    Host Port:     <none>
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
```

Deployment is successfully updated from version 1 to version 2



Check the status of pod while the deployment was being updated using

kubectl get pod -w

-w option shows continuous changes in the command output

The pods of deployment app1 with the previous image are all terminated together and then recreated back with the updated image.

```
controlplane $ kubectl get pod -w
NAME                    READY   STATUS              RESTARTS   AGE
app1-7f77c6d9cc-6w9cf   1/1     Running             0          2m9s
app1-7f77c6d9cc-7vdwr   1/1     Running             0          2m9s
app1-7f77c6d9cc-qwsb5   1/1     Running             0          2m9s
app1-7f77c6d9cc-6w9cf   1/1     Terminating         0          2m34s
app1-7f77c6d9cc-qwsb5   1/1     Terminating         0          2m34s
app1-7f77c6d9cc-7vdwr   1/1     Terminating         0          2m34s
app1-7f77c6d9cc-6w9cf   1/1     Terminating         0          2m36s
app1-7f77c6d9cc-qwsb5   1/1     Terminating         0          2m36s
app1-7f77c6d9cc-6w9cf   0/1     Completed           0          2m36s
app1-7f77c6d9cc-qwsb5   0/1     Completed           0          2m36s
app1-7f77c6d9cc-7vdwr   1/1     Terminating         0          2m36s
app1-7f77c6d9cc-7vdwr   0/1     Completed           0          2m36s
app1-7f77c6d9cc-qwsb5   0/1     Completed           0          2m37s
app1-7f77c6d9cc-qwsb5   0/1     Completed           0          2m37s
app1-7f77c6d9cc-6w9cf   0/1     Completed           0          2m37s
app1-7f77c6d9cc-6w9cf   0/1     Completed           0          2m37s
app1-7f77c6d9cc-7vdwr   0/1     Completed           0          2m37s
app1-7f77c6d9cc-7vdwr   0/1     Completed           0          2m37s
app1-5586fdd644-m9kxm   0/1     Pending             0          0s
app1-5586fdd644-cp5pk   0/1     Pending             0          0s
app1-5586fdd644-m9kxm   0/1     Pending             0          0s
app1-5586fdd644-25mr4   0/1     Pending             0          0s
app1-5586fdd644-cp5pk   0/1     Pending             0          0s
app1-5586fdd644-25mr4   0/1     Pending             0          0s
app1-5586fdd644-m9kxm   0/1     ContainerCreating   0          0s
app1-5586fdd644-cp5pk   0/1     ContainerCreating   0          0s
app1-5586fdd644-25mr4   0/1     ContainerCreating   0          0s
app1-5586fdd644-m9kxm   0/1     ContainerCreating   0          1s
app1-5586fdd644-25mr4   0/1     ContainerCreating   0          1s
app1-5586fdd644-25mr4   1/1     Running             0          2s
app1-5586fdd644-cp5pk   0/1     ContainerCreating   0          2s
app1-5586fdd644-m9kxm   1/1     Running             0          2s
app1-5586fdd644-cp5pk   1/1     Running             0          3s
```

This is because of the “recreate” deployment strategy

We can observe in the deployment events that the replicas of the deployment “app1” are scaled down to 0 and then they are scaled back up to 3

```
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  16m    deployment-controller  Scaled up replica set app1-565b4c8bf9 to 3
  Normal  ScalingReplicaSet  8m7s   deployment-controller  Scaled down replica set app1-565b4c8bf9 to 0 from 3
  Normal  ScalingReplicaSet  8m4s   deployment-controller  Scaled up replica set app1-5586fdd644 to 3
  Normal  ScalingReplicaSet  4m52s  deployment-controller  Scaled down replica set app1-5586fdd644 to 0 from 3
  Normal  ScalingReplicaSet  4m49s  deployment-controller  Scaled up replica set app1-7f77c6d9cc to 3
  Normal  ScalingReplicaSet  2m15s  deployment-controller  Scaled down replica set app1-7f77c6d9cc to 0 from 3
  Normal  ScalingReplicaSet  2m12s  deployment-controller  Scaled up replica set app1-5586fdd644 to 3 from 0
```

B) Rolling deployment:

This strategy gradually rolls out software updates across different servers or clusters, one at a time. The application remains operational during the update process. This strategy requires two versions of the application, one for the old version and one for the new version.

In Kubernetes **RollingUpdate** deployment strategy, maxSurge and maxUnavailable are key parameters that control the pace of the rollout by determining the number of pods created or deleted at a time.

1. maxSurge

- The maximum number of additional pods (surge pods) that can be created beyond the desired number of replicas during the update.
- Ensures extra capacity is temporarily available to handle traffic during the update, improving availability.
- Can be specified as an **absolute number** (e.g., 1) or a **percentage** of the total replicas (e.g., 20%).

2. maxUnavailable

- The maximum number of pods that can be unavailable during the update.
- Ensures that a minimum number of pods remain available to serve traffic during the update.

- Can be specified as an **absolute number** (e.g., 1) or a **percentage** of the total replicas (e.g., 20%).

Example:

Create a deployment “app2” with deployment strategy set to the type “recreate”

Write a yaml file to define a deployment with 5 replicas, “RollingUpdate” deployment strategy and image myapp2:v1 as shown below

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: app2
  name: app2
spec:
  replicas: 5
  selector:
    matchLabels:
      app: app2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: app2
    spec:
      containers:
      - image: docker.io/vaishnavivyawahare26/myapp:v1
        name: myapp
        resources: {}
status: {}
```

Create the deployment using above yaml file

```
controlplane $ kubectl create -f app2.yaml
deployment.apps/app2 created
```

Deployment app2 is successfully created

```
controlplane $ kubectl get deployments.apps
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
app2   5/5     5            5           15s
```

Pods managed by the deployment app2

```
controlplane $ kubectl get pod
NAME                    READY   STATUS    RESTARTS   AGE
app2-5f465d96d-jjm74    1/1     Running   0          48s
app2-5f465d96d-nfl9m    1/1     Running   0          48s
app2-5f465d96d-tdcj7    1/1     Running   0          48s
app2-5f465d96d-vzrxk    1/1     Running   0          48s
app2-5f465d96d-zfmpz    1/1     Running   0          48s
```

Here, we can observe the image as “myapp2” and deployment strategy as “RollingUpdate”

```
controlplane $ kubectl describe deployments.apps app2
Name:                   app2
Namespace:              default
CreationTimestamp:      Thu, 28 Nov 2024 14:07:25 +0000
Labels:                 app=app2
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=app2
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
  Labels:  app=app2
  Containers:
   myapp:
    Image:         docker.io/vaishnavivyawahare26/myapp:v1
```

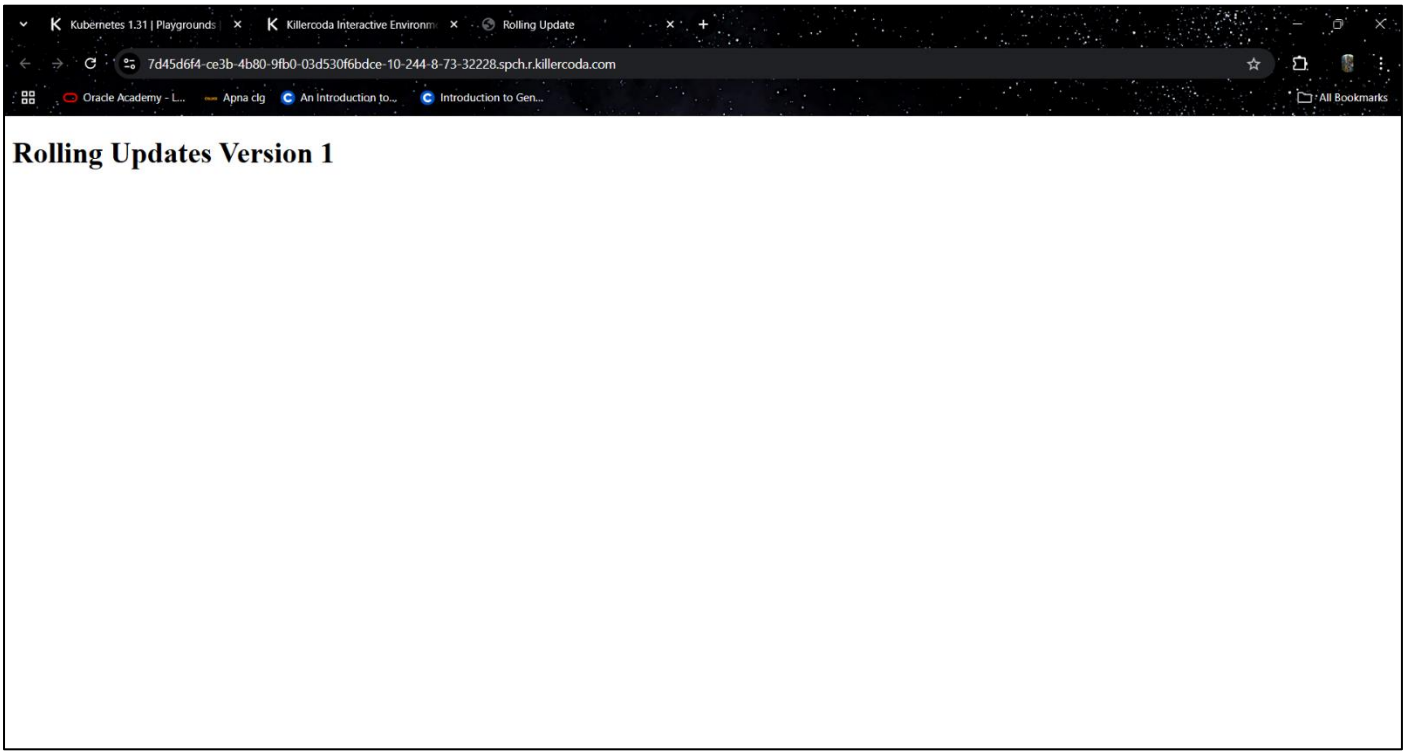
Expose the deployment to create a NodePort service

```
controlplane $ kubectl expose deployment app2 --port 18080 --target-port 80 --type NodePort
service/app2 exposed
```

Successfully created service to expose the deployment app2

```
controlplane $ kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)           AGE
app2         NodePort    10.108.205.134   <none>        18080:32228/TCP   33s
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP           22d
```

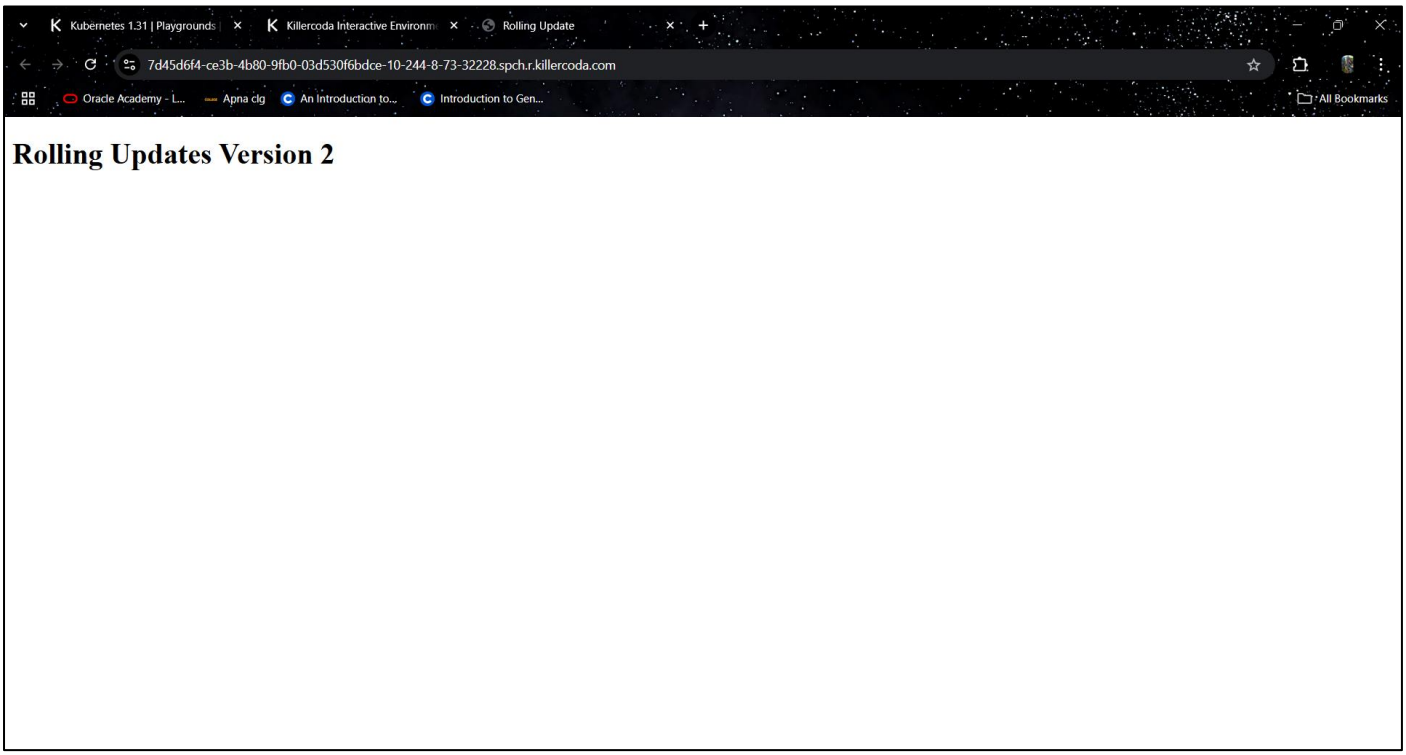
App2 is now accessible



Now, let's update the deployment to version 2 by setting the image myapp:v2

```
controlplane $ kubectl set image deployments app2 *=docker.io/vaishnavivyawahare26/myapp:v2
deployment.apps/app2 image updated
```

Deployment app2 is sucessfully updated



We can observe that the pods are newly created replacing pods with image myapp:v1 by pods with image myapp:v2

```
controlplane $ kubectl get pod
NAME                    READY   STATUS    RESTARTS   AGE
app2-58ff7f7467-25ng6   1/1     Running   0          20s
app2-58ff7f7467-dzz4k   1/1     Running   0          20s
app2-58ff7f7467-rwfvs   1/1     Running   0          16s
app2-58ff7f7467-wwnqq   1/1     Running   0          16s
app2-58ff7f7467-z59mn   1/1     Running   0          15s
```

By describing the deployment we can see that the image has updated to myapp:v2

```
controlplane $ kubectl describe deployments.apps app2
Name:                   app2
Namespace:              default
CreationTimestamp:      Thu, 28 Nov 2024 14:07:25 +0000
Labels:                 app=app2
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=app2
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
  Labels:  app=app2
  Containers:
   myapp:
    Image:         docker.io/vaishnavivyawahare26/myapp:v2
```

It can also be observed in the deployment events that the pods replicas are scaled down and scaled up leading to replace the older pods one by one by the new pods

```
Events:
  Type    Reason             Age                From                   Message
  ----    ------             ----               ----                   -------
  Normal  ScalingReplicaSet  6m20s              deployment-controller  Scaled up replica set app2-5f465d96d to 5
  Normal  ScalingReplicaSet  35s                deployment-controller  Scaled up replica set app2-58ff7f7467 to 1
  Normal  ScalingReplicaSet  35s                deployment-controller  Scaled down replica set app2-5f465d96d to 4 from 5
  Normal  ScalingReplicaSet  35s                deployment-controller  Scaled up replica set app2-58ff7f7467 to 2 from 1
  Normal  ScalingReplicaSet  31s                deployment-controller  Scaled down replica set app2-5f465d96d to 3 from 4
  Normal  ScalingReplicaSet  31s                deployment-controller  Scaled up replica set app2-58ff7f7467 to 3 from 2
  Normal  ScalingReplicaSet  31s                deployment-controller  Scaled down replica set app2-5f465d96d to 2 from 3
  Normal  ScalingReplicaSet  31s                deployment-controller  Scaled up replica set app2-58ff7f7467 to 4 from 3
  Normal  ScalingReplicaSet  30s                deployment-controller  Scaled down replica set app2-5f465d96d to 1 from 2
  Normal  ScalingReplicaSet  28s (x2 over 30s)  deployment-controller  (combined from similar events): Scaled down replica set app
2-5f465d96d to 0 from 1
```

Thus, in case of RollingUpdate deployment strategy, pods are updated by keeping enough number of pods active so that there is no downtime.