Implementing Rolling Updates in OKE

Objective

POC of Rolling Updates in OKE.

Code Snippets

1. RollingUpdate Strategy

deployment.yaml - add the below lines

- Rolling updates allow Deployments' updates to take place with zero downtime by incrementally updating Pods instances with new ones.
- The new Pods will be scheduled on Nodes with available resources.

Rolling Update

```
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
```

- **`maxSurge`** specifies the maximum number of pods above the specified number of replicas. In our case, the maximum number of pods will be 3 since 2 replicas are specified in the deployment.yaml file.
- **`maxUnavailable`** declares the maximum number of unavailable pods during the update. If `maxSurge` is set to 0, this field cannot be 0.

  Official documentation - Link

2. Readiness Probe

deployment.yaml - add the below lines to spec.template.spec.containers category

- In order for Kubernetes to know when an application is ready, it needs some help from the application. Kubernetes uses readiness probes to examine how the application is doing. Once an application instance starts responding to the Readiness probe with a positive response (200 Status code), the instance is considered ready for use.
- Readiness probes tell Kubernetes when an application is ready, but not if the application will ever become ready. If the application keeps failing, it may never respond with a positive response to Kubernetes.

Readiness probe

```
readinessProbe:
        httpGet:
       path: /oalapp/services/msadmin/actuator/health
       port: 8080
    initialDelaySeconds: 90
        periodSeconds: 10
        successThreshold: 1
```

- **`initialDelaySeconds`** specifies how long the probe has to wait to start after the container starts.
- **`periodSeconds`** is the time between two probes. The default is 10 seconds, while the minimum value is **1** second.
- **`successThreshold`** is the minimum number of consecutive successful probes after a failed one for the entire process to be considered successful. The default and minimal values are both 1.

  Official documentation - Link

3. progressDeadlineSeconds and minReadySeconds

deployment.yaml - add the below property to spec category.

spec

```
spec:
  progressDeadlineSeconds: 360
  minReadySeconds: 0
```

- **progressDeadlineSeconds** optional field that specifies the number of seconds you want to wait for your Deployment to progress before the system reports back that the Deployment has failed progressing. This defaults to 600.
- **minReadySeconds** optional field that specifies the minimum number of seconds for which a newly created Pod should be ready without any of its containers crashing, for it to be considered available. This defaults to 0.

  Official documentation - Link

4. delete deployment - no longer needed

install.pipeline - remove the below line from the method def oal_oke()

- As we aim for rolling updates, we no longer have to delete the deployment manually before a new deployment happens.

```
sh  "kubectl delete deployment oal-admin-service --namespace='${okeNamespace}' | true" // delete deployment
```

5. Rollout status

install.pipeline - add the below lines in the method def oal_oke()

- If the deployment is Successful, rollout status returns the Success status immediately.
- If the deployment doesn’t proceed until the deadline is met (progressDeadlineSeconds), Kubernetes marks the deployment status as failed, which the rollout status command will be able to pick up.
- 0 indicates rollout has a Successful status. A non-zero return value indicates that the rollout is in a Failure state.

rollout

```
ROLLOUT_STATUS = sh(script: """kubectl rollout status deployment/oal-admin-service --
namespace='${okeNamespace}' """, returnStatus: true)
```

Official documentation - Link

6. Automated Rollback

install.pipeline - add the below lines in the method def oal_oke()

- Currently, when a deployment fails in Kubernetes, the deployment process stops, but the pods from the failed deployment are kept around. On deployment failure, the environment may contain pods from both the old and new deployments.
- To get back to a stable, working state, we can use the rollout undo command to bring back the working pods and clean up the failed deployment.

rollout undo

```
sh  "kubectl rollout undo deployment/oal-admin-service --namespace='${okeNamespace}'" // rollout undo deployment

ROLLOUT_UNDO_STATUS = sh(script: """kubectl rollout status deployment/oal-admin-service --
namespace='${okeNamespace}' """, returnStatus: true)
```

Sample MR

https://alm.oraclecorp.com/oal/#projects/gxp/scm/OalcnMsAdminService.git/compare/cn-development..ft_rollingUpdates

Functionality

Naming convention

- Pods existing before deployment: old-pod-1, old-pod-2
- Pods to be created after a successful deployment: new-pod-1, new-pod-2

Scenario-1 New pod comes Up & is in a Healthy State

The expected sequence of steps

	Step	Pods and their states	
1	Before deployment	old-pod-1	Running
		old-pod-2	Running
2	During deployment, a new pod gets created	old-pod-1	Running
		old-pod-2	Running
		new-pod-1	ContainerCreating
3	Readiness probe is responsible to check the health status of the new pod	old-pod-1	Running
		old-pod-2	Running
		new-pod-1	Running
4	Once the readiness probe confirms the good health of the new pod, it tries to bring down the old pod	old-pod-1	Terminating
		old-pod-2	Running
		new-pod-1	Running
5	At this stage, we'll have 2 pods. One pod with the new image and another pod with the old image	old-pod-2	Running
		new-pod-1	Running
6	A second new pod will start getting created	old-pod-2	Running
		new-pod-1	Running
		new-pod-2	ContainerCreating
7	Readiness probe is responsible to check the health status of the new pod	old-pod-2	Running
		new-pod-1	Running
		new-pod-2	Running
8	Once the readiness probe confirms the good health of the new pod, it tries to bring down the old pod	old-pod-2	Terminating
		new-pod-1	Running

		new-pod-2	Running
9	Both the new pods are created and the old pods are deleted	new-pod-1	Running
		new-pod-2	Running

Scenario-2 New pod tries to come Up & is in a bad state

The expected sequence of steps

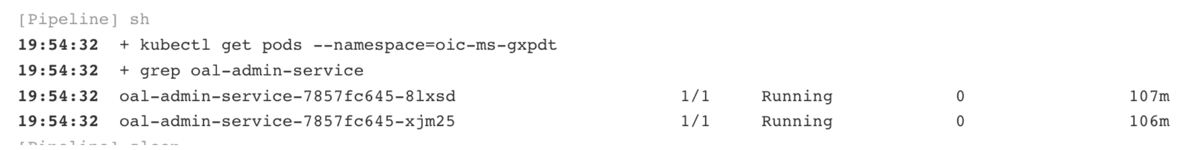
	Step	Pods and their states	
1	before deployment	old-pod-1	Running
		old-pod-2	Running
2	During deployment, a new pod gets created	old-pod-1	Running
		old-pod-2	Running
		new-pod-1	ContainerCreating
3	the readiness probe is responsible to check the health status of the new pod and the new pod fails to show good health. It checks until progressDeadlineSeconds	old-pod-1	Running
		old-pod-2	Running
		new-pod-1	Running / Crashloop backoff / Error
4	As the pod failed to come up before progressDeadlineSeconds, we do a rollout undo. new-pod-1 gets terminated.	old-pod-1	Running
		old-pod-2	Running

Testing Results

Testing is done on OalcnMsAdminService. The results are summarized below:

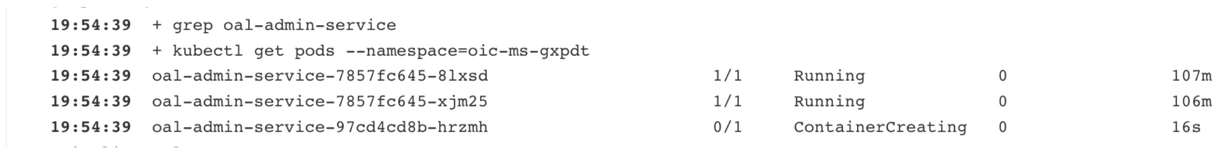
Scenario-1 New pod comes Up & is in a Healthy State

Step-1: Before deployment



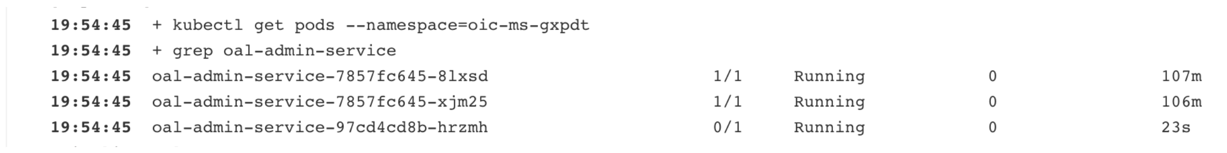
```
[Pipeline] sh
19:54:32  + kubectl get pods --namespace=oic-ms-gxpdt
19:54:32  + grep oal-admin-service
19:54:32  oal-admin-service-7857fc645-8lxsd                  1/1     Running           0                107m
19:54:32  oal-admin-service-7857fc645-xjm25                  1/1     Running           0                106m
```

Step-2: During deployment, a new pod gets created



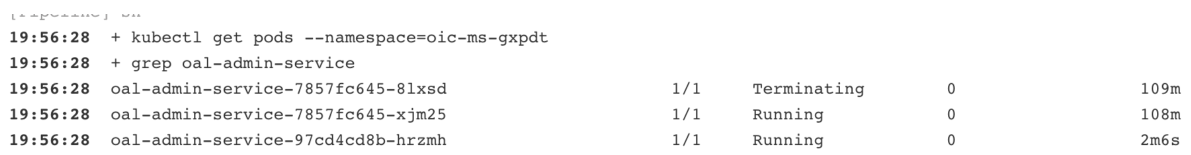
```
19:54:39  + grep oal-admin-service
19:54:39  + kubectl get pods --namespace=oic-ms-gxpdt
19:54:39  oal-admin-service-7857fc645-8lxsd                  1/1     Running            0                107m
19:54:39  oal-admin-service-7857fc645-xjm25                  1/1     Running            0                106m
19:54:39  oal-admin-service-97cd4cd8b-hrzmh                  0/1     ContainerCreating   0                16s
```

Step-3: Readiness probe is responsible to check the health status of the new pod

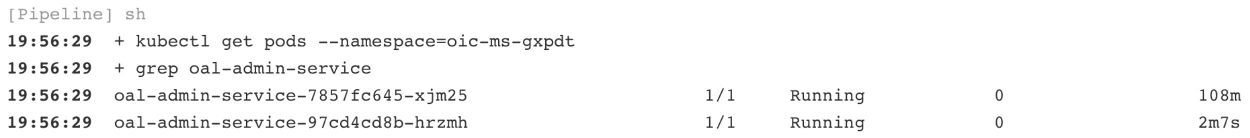


```
19:54:45  + kubectl get pods --namespace=oic-ms-gxpdt
19:54:45  + grep oal-admin-service
19:54:45  oal-admin-service-7857fc645-8lxsd                  1/1     Running           0                107m
19:54:45  oal-admin-service-7857fc645-xjm25                  1/1     Running           0                106m
19:54:45  oal-admin-service-97cd4cd8b-hrzmh                  0/1     Running           0                23s
```

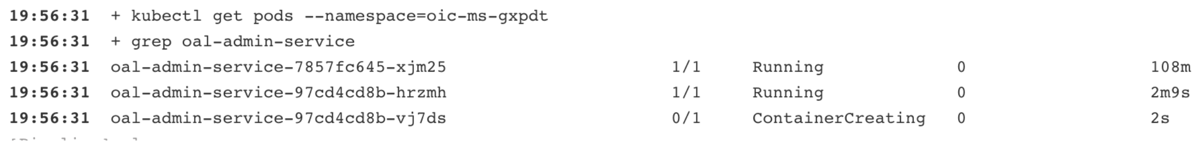
Step-4: Once readiness probe confirms the good health of new pod, it tries to bring down the old pod


```
19:56:28  + kubectl get pods --namespace=oic-ms-gxpdt
19:56:28  + grep oal-admin-service
19:56:28  oal-admin-service-7857fc645-8lxsd          1/1     Terminating       0                  109m
19:56:28  oal-admin-service-7857fc645-xjm25          1/1     Running           0                  108m
19:56:28  oal-admin-service-97cd4cd8b-hrzmh          1/1     Running           0                  2m6s
```

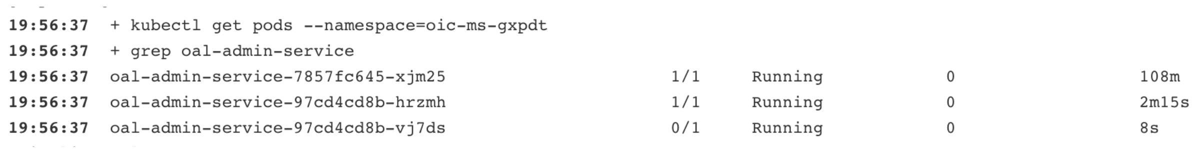
Step-5: At this stage, we'll have 2 pods. One pod with the new image and another pod with the old image


```
[Pipeline] sh
19:56:29  + kubectl get pods --namespace=oic-ms-gxpdt
19:56:29  + grep oal-admin-service
19:56:29  oal-admin-service-7857fc645-xjm25          1/1     Running           0                  108m
19:56:29  oal-admin-service-97cd4cd8b-hrzmh          1/1     Running           0                  2m7s
```

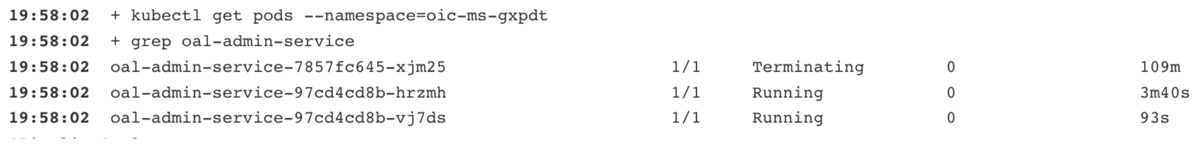
Step-6: Second new pod will start getting created


```
19:56:31  + kubectl get pods --namespace=oic-ms-gxpdt
19:56:31  + grep oal-admin-service
19:56:31  oal-admin-service-7857fc645-xjm25          1/1     Running           0                  108m
19:56:31  oal-admin-service-97cd4cd8b-hrzmh          1/1     Running           0                  2m9s
19:56:31  oal-admin-service-97cd4cd8b-vj7ds          0/1     ContainerCreating   0                2s
```

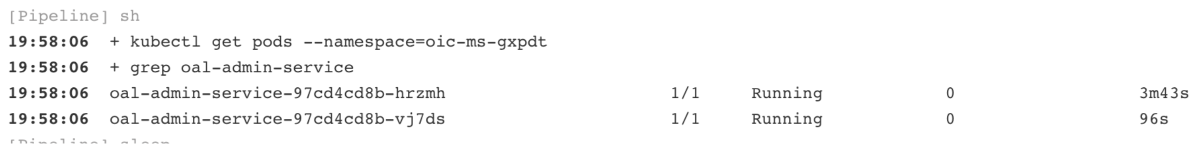
Step-7: The readiness probe is responsible to check the health status of the new pod


```
19:56:37  + kubectl get pods --namespace=oic-ms-gxpdt
19:56:37  + grep oal-admin-service
19:56:37  oal-admin-service-7857fc645-xjm25          1/1     Running           0                  108m
19:56:37  oal-admin-service-97cd4cd8b-hrzmh          1/1     Running           0                  2m15s
19:56:37  oal-admin-service-97cd4cd8b-vj7ds          0/1     Running           0                  8s
```

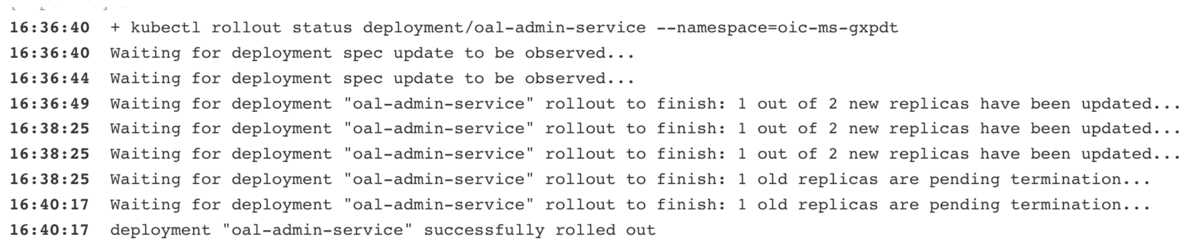
Step-8: Once the readiness probe confirms the good health of the new pod, it tries to bring down the old pod


```
19:58:02  + kubectl get pods --namespace=oic-ms-gxpdt
19:58:02  + grep oal-admin-service
19:58:02  oal-admin-service-7857fc645-xjm25          1/1     Terminating       0                  109m
19:58:02  oal-admin-service-97cd4cd8b-hrzmh          1/1     Running           0                  3m40s
19:58:02  oal-admin-service-97cd4cd8b-vj7ds          1/1     Running           0                  93s
```

Step-9: Both the new pods are created successfully and the old pods are deleted


```
[Pipeline] sh
19:58:06  + kubectl get pods --namespace=oic-ms-gxpdt
19:58:06  + grep oal-admin-service
19:58:06  oal-admin-service-97cd4cd8b-hrzmh          1/1     Running           0                  3m43s
19:58:06  oal-admin-service-97cd4cd8b-vj7ds          1/1     Running           0                  96s
```

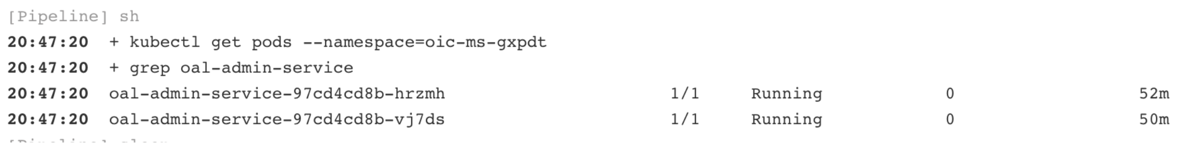
Additional information:


```
16:36:40  + kubectl rollout status deployment/oal-admin-service --namespace=oic-ms-gxpdt
16:36:40  Waiting for deployment spec update to be observed...
16:36:44  Waiting for deployment spec update to be observed...
16:36:49  Waiting for deployment "oal-admin-service" rollout to finish: 1 out of 2 new replicas have been updated...
16:38:25  Waiting for deployment "oal-admin-service" rollout to finish: 1 out of 2 new replicas have been updated...
16:38:25  Waiting for deployment "oal-admin-service" rollout to finish: 1 out of 2 new replicas have been updated...
16:38:25  Waiting for deployment "oal-admin-service" rollout to finish: 1 old replicas are pending termination...
16:40:17  Waiting for deployment "oal-admin-service" rollout to finish: 1 old replicas are pending termination...
16:40:17  deployment "oal-admin-service" successfully rolled out
```

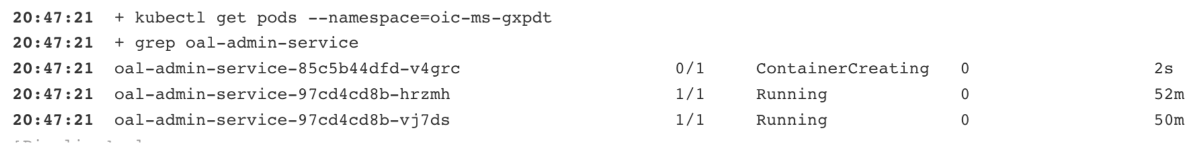
Scenario-2 New pod tries to come Up & is in a bad State

To simulate this scenario for creating a new pod in an unhealthy state, updated the config URL in the application.properties to point to the wrong URL.

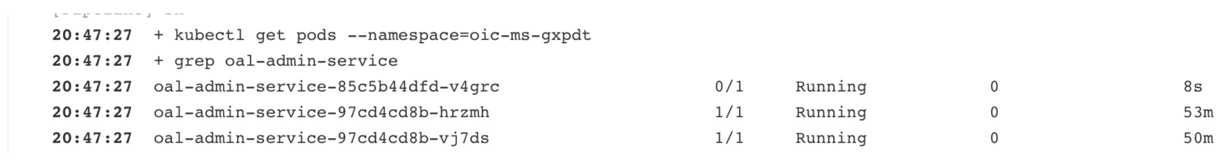
Step -1: before deployment


```
[Pipeline] sh
20:47:20  + kubectl get pods --namespace=oic-ms-gxpdt
20:47:20  + grep oal-admin-service
20:47:20  oal-admin-service-97cd4cd8b-hrzmh          1/1     Running           0                  52m
20:47:20  oal-admin-service-97cd4cd8b-vj7ds          1/1     Running           0                  50m
```

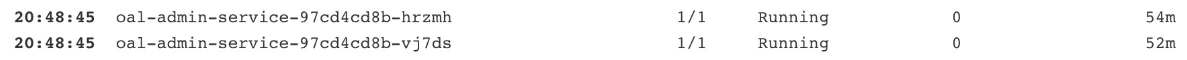
Step-2: During deployment, a new pod gets created


```
20:47:21  + kubectl get pods --namespace=oic-ms-gxpdt
20:47:21  + grep oal-admin-service
20:47:21  oal-admin-service-85c5b44dfd-v4grc                    0/1     ContainerCreating   0                  2s
20:47:21  oal-admin-service-97cd4cd8b-hrzmh                     1/1     Running             0                  52m
20:47:21  oal-admin-service-97cd4cd8b-vj7ds                     1/1     Running             0                  50m
```

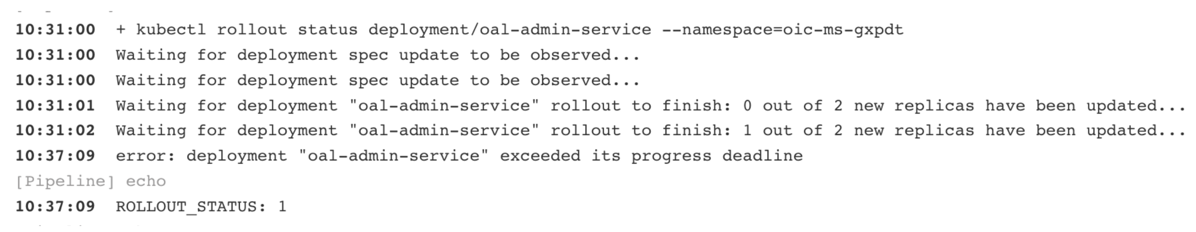
Step-3: New pod comes to running state


```
20:47:27  + kubectl get pods --namespace=oic-ms-gxpdt
20:47:27  + grep oal-admin-service
20:47:27  oal-admin-service-85c5b44dfd-v4grc                    0/1     Running            0                  8s
20:47:27  oal-admin-service-97cd4cd8b-hrzmh                     1/1     Running            0                  53m
20:47:27  oal-admin-service-97cd4cd8b-vj7ds                     1/1     Running            0                  50m
```

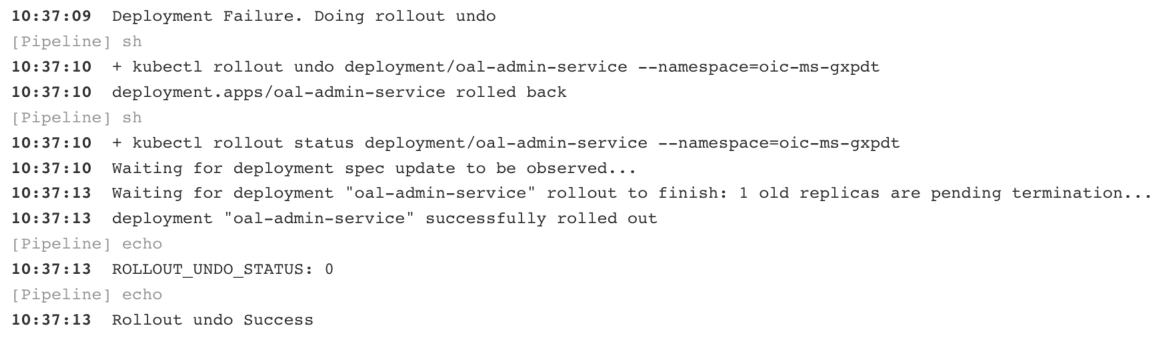
Step-4: readiness probe is responsible to check the health status of the new pod and new pod fails to show good health and keeps on restarting


```
20:48:45  oal-admin-service-97cd4cd8b-hrzmh                     1/1     Running            0                  54m
20:48:45  oal-admin-service-97cd4cd8b-vj7ds                     1/1     Running            0                  52m
```

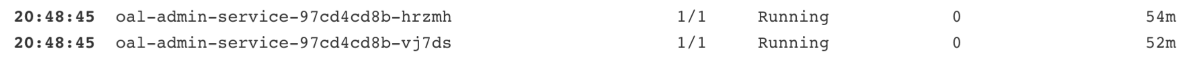
Step-5: Rollout failed. (status is non-zero)


```
10:31:00  + kubectl rollout status deployment/oal-admin-service --namespace=oic-ms-gxpdt
10:31:00  Waiting for deployment spec update to be observed...
10:31:00  Waiting for deployment spec update to be observed...
10:31:01  Waiting for deployment "oal-admin-service" rollout to finish: 0 out of 2 new replicas have been updated...
10:31:02  Waiting for deployment "oal-admin-service" rollout to finish: 1 out of 2 new replicas have been updated...
10:37:09  error: deployment "oal-admin-service" exceeded its progress deadline
[Pipeline] echo
10:37:09  ROLLOUT_STATUS: 1
```

Step-6: Rollout undo.


```
10:37:09  Deployment Failure. Doing rollout undo
[Pipeline] sh
10:37:10  + kubectl rollout undo deployment/oal-admin-service --namespace=oic-ms-gxpdt
10:37:10  deployment.apps/oal-admin-service rolled back
[Pipeline] sh
10:37:10  + kubectl rollout status deployment/oal-admin-service --namespace=oic-ms-gxpdt
10:37:10  Waiting for deployment spec update to be observed...
10:37:13  Waiting for deployment "oal-admin-service" rollout to finish: 1 old replicas are pending termination...
10:37:13  deployment "oal-admin-service" successfully rolled out
[Pipeline] echo
10:37:13  ROLLOUT_UNDO_STATUS: 0
[Pipeline] echo
10:37:13  Rollout undo Success
```

Step-7: final state


```
20:48:45  oal-admin-service-97cd4cd8b-hrzmh                     1/1     Running            0                  54m
20:48:45  oal-admin-service-97cd4cd8b-vj7ds                     1/1     Running            0                  52m
```

Scenario-3 New pod -2 Creation fails

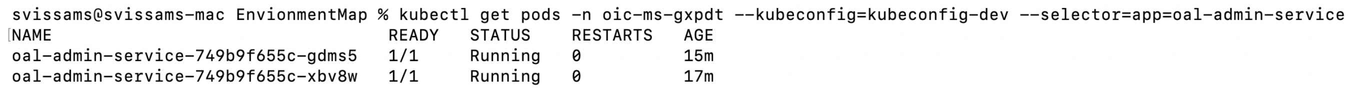
Problem Statement:

Initially, we have 2 pods (old-pod-1 and old-pod-2 with old code). When we initiate the deployment, new-pod-1 is created with the new code successfully. At this time, old-pod-1 is deleted. Now if the 2nd new pod deployment fails, what would happen?

Results:

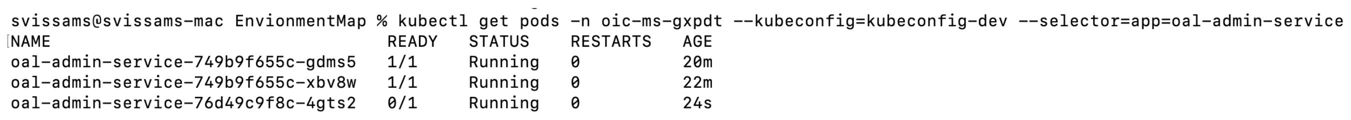
To test this scenario, I reduced the progressDeadlineSeconds to 120sec from 360sec so that new-pod-1 will be created successfully and new-pod-2 creation is failed. Hence rollback happens. Our final state will be old-pod-2 and new-pod-3 (created during rollback but having an older version of code). Note that the reproducibility rate is very low in this approach and I was able to reproduce this only once in several attempts.

Step-1: before deployment : (old-pod-1 and old-pod-2)



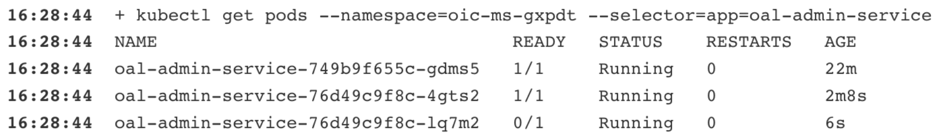
```
svissams@svissams-mac EnvionmentMap % kubectl get pods -n oic-ms-gxpdt --kubeconfig=kubeconfig-dev --selector=app=oal-admin-service
NAME                                READY   STATUS    RESTARTS   AGE
oal-admin-service-749b9f655c-gdms5   1/1     Running   0          15m
oal-admin-service-749b9f655c-xbv8w   1/1     Running   0          17m
```

Step-2: During deployment, a new pod gets created (old-pod-1, old-pod-2 and new-pod-1)



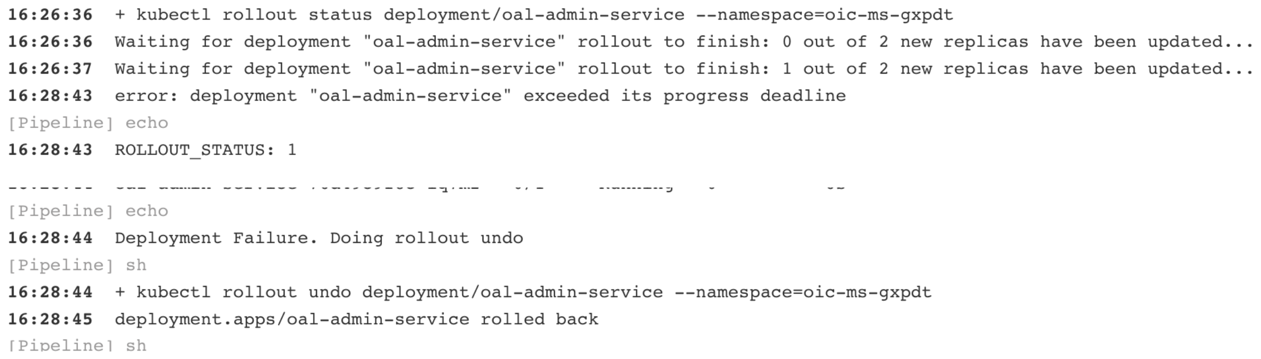
```
svissams@svissams-mac EnvionmentMap % kubectl get pods -n oic-ms-gxpdt --kubeconfig=kubeconfig-dev --selector=app=oal-admin-service
NAME                                READY   STATUS    RESTARTS   AGE
oal-admin-service-749b9f655c-gdms5   1/1     Running   0          20m
oal-admin-service-749b9f655c-xbv8w   1/1     Running   0          22m
oal-admin-service-76d49c9f8c-4gts2   0/1     Running   0          24s
```

Step-3: Once the readiness probe confirms the good health of new-pod-1, old-pod-1 is terminated. new-pod-2 tries to come up.



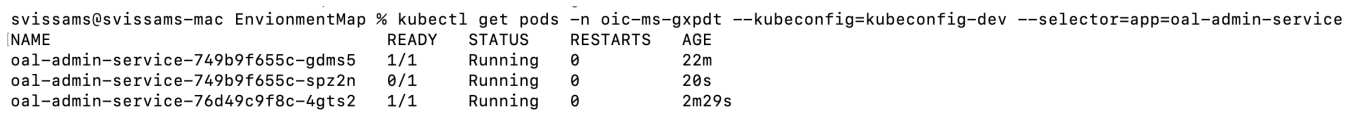
```
16:28:44  + kubectl get pods --namespace=oic-ms-gxpdt --selector=app=oal-admin-service
16:28:44  NAME                                READY   STATUS    RESTARTS   AGE
16:28:44  oal-admin-service-749b9f655c-gdms5   1/1     Running   0          22m
16:28:44  oal-admin-service-76d49c9f8c-4gts2   1/1     Running   0          2m8s
16:28:44  oal-admin-service-76d49c9f8c-lq7m2   0/1     Running   0          6s
```

Step-4: progressDeadlineSeconds is met, but 2 new pods didn't come up successfully in given time. Hence, rollout undo happens.



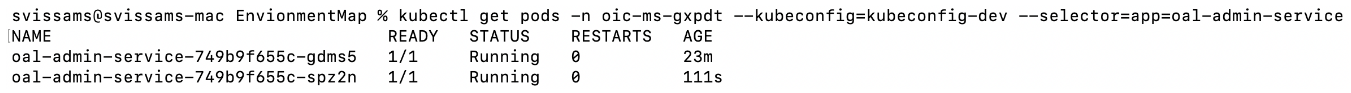
```
16:26:36  + kubectl rollout status deployment/oal-admin-service --namespace=oic-ms-gxpdt
16:26:36  Waiting for deployment "oal-admin-service" rollout to finish: 0 out of 2 new replicas have been updated...
16:26:37  Waiting for deployment "oal-admin-service" rollout to finish: 1 out of 2 new replicas have been updated...
16:28:43  error: deployment "oal-admin-service" exceeded its progress deadline
[Pipeline] echo
16:28:43  ROLLOUT_STATUS: 1

16:28:44  oal-admin-service-76d49c9f8c-lq7m2   0/1     Running   0          6s
[Pipeline] echo
16:28:44  Deployment Failure. Doing rollout undo
[Pipeline] sh
16:28:44  + kubectl rollout undo deployment/oal-admin-service --namespace=oic-ms-gxpdt
16:28:45  deployment.apps/oal-admin-service rolled back
[Pipeline] sh
```

Step-5: new-pod-3 (with older version) is created



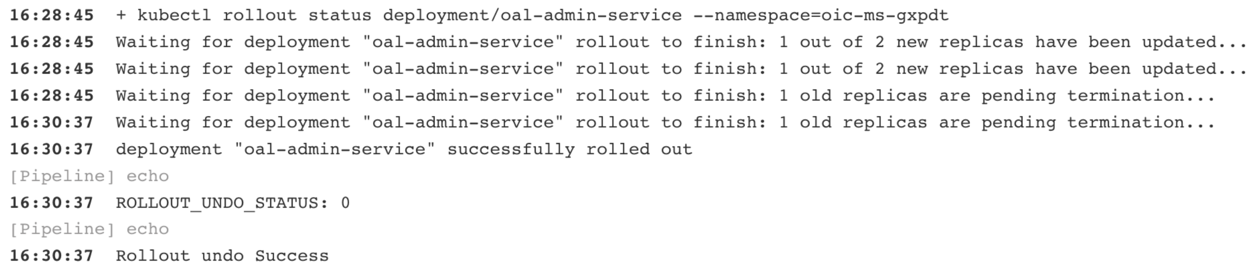
```
svissams@svissams-mac EnvionmentMap % kubectl get pods -n oic-ms-gxpdt --kubeconfig=kubeconfig-dev --selector=app=oal-admin-service
NAME                                READY   STATUS    RESTARTS   AGE
oal-admin-service-749b9f655c-gdms5   1/1     Running   0          22m
oal-admin-service-749b9f655c-spz2n   0/1     Running   0          20s
oal-admin-service-76d49c9f8c-4gts2   1/1     Running   0          2m29s
```

Step-6: Rollout undo happens, new-pod-1 and new-pod-2 are terminated. new-pod-3 (with old code is up now)



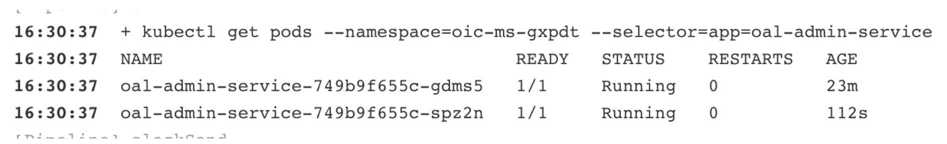
```
svissams@svissams-mac EnvionmentMap % kubectl get pods -n oic-ms-gxpdt --kubeconfig=kubeconfig-dev --selector=app=oal-admin-service
NAME                                READY   STATUS    RESTARTS   AGE
oal-admin-service-749b9f655c-gdms5   1/1     Running   0          23m
oal-admin-service-749b9f655c-spz2n   1/1     Running   0          111s
```

Rollout undo logs



```
16:28:45  + kubectl rollout status deployment/oal-admin-service --namespace=oic-ms-gxpdt
16:28:45  Waiting for deployment "oal-admin-service" rollout to finish: 1 out of 2 new replicas have been updated...
16:28:45  Waiting for deployment "oal-admin-service" rollout to finish: 1 out of 2 new replicas have been updated...
16:28:45  Waiting for deployment "oal-admin-service" rollout to finish: 1 old replicas are pending termination...
16:30:37  Waiting for deployment "oal-admin-service" rollout to finish: 1 old replicas are pending termination...
16:30:37  deployment "oal-admin-service" successfully rolled out
[Pipeline] echo
16:30:37  ROLLOUT_UNDO_STATUS: 0
[Pipeline] echo
16:30:37  Rollout undo Success
```

Step 7: old-pod-2 and new-pod-3 (with old code) remain at the end of the deployment.



```
16:30:37  + kubectl get pods --namespace=oic-ms-gxpdt --selector=app=oal-admin-service
16:30:37  NAME                                 READY   STATUS    RESTARTS   AGE
16:30:37  oal-admin-service-749b9f655c-gdms5   1/1     Running   0          23m
16:30:37  oal-admin-service-749b9f655c-spz2n   1/1     Running   0          112s
```

References

1. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
2. https://polarsquad.com/blog/check-your-kubernetes-deployments