Exercise 7.3: Rolling Updates and Rollbacks

One of the advantages of micro-services is the ability to replace and upgrade a container while continuing to respond to client requests. We will use the `OnDelete` setting that upgrades a container when the predecessor is deleted, then the use the `RollingUpdate` feature as well, which begins a rolling update immediately.

nginx versions

The **nginx** software updates on a distinct timeline from Kubernetes. If the lab shows an older version please use the current default, and then a newer version. Versions can be seen with this command: **sudo docker image ls nginx**

1. Begin by viewing the current `updateStrategy` setting for the `DaemonSet` created in the previous section.

```
student@cp:~$ kubectl get ds ds-one -o yaml | grep -A 4 Strategy

  updateStrategy:
    rollingUpdate:
      maxSurge:; 0
      maxUnavailable: 1
    type: RollingUpdate
```

2. Edit the object to use the `OnDelete` update strategy. This would allow the manual termination of some of the pods, resulting in an updated image when they are recreated.

```
student@cp:~$ kubectl edit ds ds-one

....
  updateStrategy:
    rollingUpdate:
      maxUnavailable: 1
    type: OnDelete              #<-- Edit to be this line
status:
....
```

3. Update the `DaemonSet` to use a newer version of the **nginx** server. This time use the **set** command instead of **edit**. Set the version to be `1.16.1-alpine`.

```
student@cp:~$ kubectl set image ds ds-one nginx=nginx:1.16.1-alpine
daemonset.apps/ds-one image updated
```

4. Verify that the `Image:` parameter for the Pod checked in the previous section is unchanged.

```
student@cp:~$ kubectl describe po ds-one-b1dcv |grep Image:
    Image:               nginx:1.15.1
```

5. Delete the Pod. Wait until the replacement Pod is running and check the version.

```
student@cp:~$ kubectl delete po ds-one-b1dcv
pod "ds-one-b1dcv" deleted

student@cp:~$ kubectl get pod
```

```
NAME                  READY     STATUS    RESTARTS   AGE
ds-one-xc86w          1/1       Running   0          19s
ds-one-z31r4          1/1       Running   0          4m8s
```

```
student@cp:~$ kubectl describe pod ds-one-xc86w |grep Image:
```

```
    Image:              nginx:1.16.1-alpine
```

6. View the image running on the older Pod. It should still show version 1.15.1.

```
student@cp:~$ kubectl describe pod ds-one-z31r4 |grep Image:
```

```
    Image:              nginx:1.15.1
```

7. View the history of changes for the `DaemonSet`. You should see two revisions listed. As we did not use the `--record` option we didn't see why the object updated.

```
student@cp:~$ kubectl rollout history ds ds-one
```

```
daemonsets "ds-one"
REVISION        CHANGE-CAUSE
1               <none>
2               <none>
```

8. View the settings for the various versions of the `DaemonSet`. The `Image:` line should be the only difference between the two outputs.

```
student@cp:~$ kubectl rollout history ds ds-one --revision=1
```

```
daemonsets "ds-one" with revision #1
Pod Template:
  Labels:       system=DaemonSetOne
  Containers:
   nginx:
    Image:      nginx:1.15.1
    Port:       80/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

```
student@cp:~$ kubectl rollout history ds ds-one --revision=2
```

```
....
    Image:      nginx:1.16.1-alpine
.....
```

9. Use `kubectl rollout undo` to change the `DaemonSet` back to an earlier version. As we are still using the `OnDelete` strategy there should be no change to the Pods.

```
student@cp:~$ kubectl rollout undo ds ds-one --to-revision=1
```

```
daemonset.apps/ds-one rolled back
```

```
student@cp:~$ kubectl describe pod ds-one-xc86w |grep Image:
```

```
    Image:              nginx:1.16.1-alpine
```

10. Delete the Pod, wait for the replacement to spawn then check the image version again.

```
student@cp:~$ kubectl delete pod ds-one-xc86w
pod "ds-one-xc86w" deleted

student@cp:~$ kubectl get pod
NAME                  READY     STATUS        RESTARTS   AGE
ds-one-qc72k          1/1       Running       0          10s
ds-one-xc86w          0/1       Terminating   0          12m
ds-one-z31r4          1/1       Running       0          28m

student@cp:~$ kubectl describe po ds-one-qc72k |grep Image:
    Image:              nginx:1.15.1
```

11. View the details of the `DaemonSet`. The Image should be v1.15.1 in the output.

```
student@cp:~$ kubectl describe ds |grep Image:
    Image:              nginx:1.15.1
```

12. View the current configuration for the `DaemonSet` in YAML output. Look for the `updateStrategy:` the the `type:`

```
student@cp:~$ kubectl get ds ds-one -o yaml

apiVersion: apps/v1
kind: DaemonSet
.....
      terminationGracePeriodSeconds: 30
  updateStrategy:
    type: OnDelete
status:
  currentNumberScheduled: 2
.....
```

13. Create a new `DaemonSet`, this time setting the update policy to `RollingUpdate`. Begin by generating a new config file.

```
student@cp:~$ kubectl get ds ds-one -o yaml  > ds2.yaml
```

14. Edit the file. Change the name, around line 69 and the update strategy around line 100, back to the default `RollingUpdate`.

```
student@cp:~$ vim ds2.yaml

....
  name: ds-two
....
    type: RollingUpdate
```

15. Create the new `DaemonSet` and verify the **nginx** version in the new pods.

```
student@cp:~$ kubectl create -f ds2.yaml
daemonset.apps/ds-two created

student@cp:~$ kubectl get pod
```

```
NAME                  READY     STATUS    RESTARTS   AGE
ds-one-qc72k          1/1       Running   0          28m
ds-one-z31r4          1/1       Running   0          57m
ds-two-10khc          1/1       Running   0          5m
ds-two-kzp9g          1/1       Running   0          5m
```

```
student@cp:~$ kubectl describe po ds-two-10khc |grep Image:
```

```
    Image:              nginx:1.15.1
```

16. Edit the configuration file and set the image to a newer version such as 1.16.1-alpine. Include the `--record` option.

```
student@cp:~$ kubectl edit ds ds-two --record
```

```
....
      - image: nginx:1.16.1-alpine
.....
```

17. View the age of the `DaemonSets`. It should be around ten minutes old, depending on how fast you type.

```
student@cp:~$ kubectl get ds ds-two
```

```
NAME      DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE-SELECTOR   AGE
ds-two    2         2         2         2            2           <none>          10m
```

18. Now view the age of the Pods. Two should be much younger than the `DaemonSet`. They are also a few seconds apart due to the nature of the rolling update where one then the other pod was terminated and recreated.

```
student@cp:~$ kubectl get pod
```

```
NAME                  READY     STATUS    RESTARTS   AGE
ds-one-qc72k          1/1       Running   0          36m
ds-one-z31r4          1/1       Running   0          1h
ds-two-2p8vz          1/1       Running   0          34s
ds-two-8lx7k          1/1       Running   0          32s
```

19. Verify the Pods are using the new version of the software.

```
student@cp:~$ kubectl describe po ds-two-8lx7k |grep Image:
```

```
    Image:              nginx:1.16.1-alpine
```

20. View the rollout status and the history of the `DaemonSets`.

```
student@cp:~$ kubectl rollout status ds ds-two
```

```
daemon set "ds-two" successfully rolled out
```

```
student@cp:~$ kubectl rollout history ds ds-two
```

```
daemonsets "ds-two"
REVISION        CHANGE-CAUSE
1               <none>
2               kubectl edit ds ds-two --record=true
```

21. View the changes in the update they should look the same as the previous history, but did not require the Pods to be deleted for the update to take place.

```
student@cp:~$ kubectl rollout history ds ds-two --revision=2
```

```
...
    Image:        nginx:1.16.1-alpine
```

22. Clean up the system by removing the `DaemonSets`.

```
student@cp:~$ kubectl delete ds ds-one ds-two
```

```
daemonset.apps "ds-one" deleted
daemonset.apps "ds-two" deleted
```