Playing around with the Deployment

In this lesson, we will deploy a few new releases and will play around with the Deployment to explore multiple options.

WE'LL COVER THE FOLLOWING

Deploying New Releases

Let's deploy a few new releases. That will provide us with a broader playground to explore a few additional things we can do with Deployments.

```
kubectl set image \
    -f deploy/go-demo-2-api.yml \
    api=vfarcic/go-demo-2:3.0 \
    --record

kubectl rollout status \
    -f deploy/go-demo-2-api.yml
```

We updated the image to `vfarcic/go-demo-2:3.0` and retrieved the rollout status. The last line of the **output** of latter command is as follows.

```
deployment "go-demo-2-api" successfully rolled out
```

The deployment was successfully updated and, as a result, it created a new ReplicaSet and scaled it up to the desired number of replicas. The previously active ReplicaSet was scaled to `0`. As a result, we're running tag `3.0` of the `vfarcic/go-demo-2` image.

We'll repeat the process with the tag `4.0`.

```
kubectl set image \
    -f deploy/go-demo-2-api.yml \
    api=vfarcic/go-demo-2:4.0 \
    --record

kubectl rollout status \
    -f deploy/go-demo-2-api.yml
```

The **output** of the last line of the `rollout status` confirmed that the rollout was successful.

Checking the Rollout History

Now that we deployed a few releases, we can check the current `rollout history`.

```
kubectl rollout history \
    -f deploy/go-demo-2-api.yml
```

The **output** is as follows.

```
deployments "go-demo-2-api"
REVISION CHANGE-CAUSE
2        kubectl set image api=vfarcic/go-demo-2:2.0 --filename=deploy/go-demo-2-api.yml --re
3        kubectl create --filename=deploy/go-demo-2-api.yml --record=true
4        kubectl set image api=vfarcic/go-demo-2:3.0 --filename=deploy/go-demo-2-api.yml --re
5        kubectl set image api=vfarcic/go-demo-2:4.0 --filename=deploy/go-demo-2-api.yml --re
```

We can clearly see the commands that produced the changes and, through them, how our application progressed all the way until the current release based on the image `vfarcic/go-demo-2:4.0`.

You saw that we can rollback to the previous release through the `kubectl rollout undo` command. In most cases, that should be the correct action when faced with problems and without the ability to roll forward by creating a new release with the fix. However, sometimes even that is not enough, and we have to go back in time further than the previous release.

Undoing the Rollout

Let's say that we discovered not only that the current release is faulty but also

that a few before it have bugs as well. Following the same narrative, we'll

imagine that the last correct release was based on the image `vfarcic/go-demo-2:2.0`. We can remedy that by executing the command that follows.

> ⚠ Please do NOT run it.

```
kubectl set image \
    -f deploy/go-demo-2-api.yml \
    api=vfarcic/go-demo-2:2.0 \
    --record
```

While that command would certainly fix the problem, there is an easier way to accomplish the same result. We can `undo` the `rollout` by moving to the last revision that worked correctly. Assuming that we want to revert to the image `vfarcic/go-demo-2:2.0`, reviewing the change causes listed in the history tells us we should roll back to revision `2`. That can be accomplished through the `--to-revision` argument. The command is as follows.

```
kubectl rollout undo \
    -f deploy/go-demo-2-api.yml \
    --to-revision=2

kubectl rollout history \
    -f deploy/go-demo-2-api.yml
```

We undid the rollout by moving to revision `2`. We also retrieved the `history`.

The **output** of the latter command is as follows.

```
deployments "go-demo-2-api"
REVISION  CHANGE-CAUSE
3         kubectl create --filename=deploy/go-demo-2-api.yml --record=true
4         kubectl set image api=vfarcic/go-demo-2:3.0 --filename=deploy/go-demo-2-api.yml --r
5         kubectl set image api=vfarcic/go-demo-2:4.0 --filename=deploy/go-demo-2-api.yml --r
6         kubectl set image api=vfarcic/go-demo-2:2.0 --filename=deploy/go-demo-2-api.yml --r
```

Through the new revision `6`, we can see that the currently active Deployment is based on the image `vfarcic/go-demo-2:2.0`.

We successfully moved back to a specific point in time. The problem is solved and, if this was the "real" application running in a production cluster, our

users would continue interacting with the version of our software that actually works.

In the next lesson, we will learn to roll back the failed deployments.