MANAGING STATE WITH DEPLOYMENTS

Managing State with Deployments

Deployment Rollbacks

With some of the previous ReplicaSets of a Deployment being kept, you can also roll back to a previous revision by scaling up and down. The number of previous configurations kept is configurable, and has changed from version to version.

```
$ kubectl create deploy ghost --image=ghost
```

```
$ kubectl annotate deployment/ghost kubernetes.io/change-cause="kubectl create deploy ghost --image=ghost"
```

```
$ kubectl get deployments ghost -o yaml
```

```
deployment.kubernetes.io/revision: "1"
kubernetes.io/change-cause: kubectl create deploy ghost --image=ghost
```

Should an update fail, due to an improper image version, for example, you can roll back the change to a working version with the **`kubectl rollout undo`** command (followed by the output):

```
$ kubectl set image deployment/ghost ghost=ghost:09 --all
```

```
$ kubectl get pods
```

```
NAME                     READY   STATUS             RESTARTS   AGE
ghost-2141819201-tcths   0/1     ImagePullBackOff   0          1m
```

```
$ kubectl rollout undo deployment/ghost ; kubectl get pods
```

```
NAME                     READY   STATUS    RESTARTS   AGE
ghost-3378155678-eq5i6   1/1     Running   0          7s
```

You can roll back to a specific revision with the **`--to-revision=2`** option.

You can also edit a Deployment using the **`kubectl edit`** command.

You can also pause a Deployment, and then resume. See the following two commands:

```
$ kubectl rollout pause deployment/ghost
```

```
$ kubectl rollout resume deployment/ghost
```

Please note that you can still do a rolling update on ReplicationControllers with the **`kubectl rolling-update`** command, but this is done on the client side. Hence, if you close your client, the rolling update will stop.