**Deployments**

Kubernetes deployments by example

A deployment is a supervisor for [pods](http://kubernetesbyexample.com/pods/), giving you fine-grained control over how and when a new pod version is rolled out as well as rolled back to a previous state.

Let’s create a [deployment](https://github.com/openshift-evangelists/kbe/blob/master/specs/deployments/d09.yaml) called sise-deploy that supervises two replicas of a pod as well as a replica set:

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/deployments/d09.yaml

You can have a look at the deployment, as well as the the replica set and the pods the deployment looks after like so:

$ kubectl get deploy

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

sise-deploy 2 2 2 2 10s

$ kubectl get rs

NAME DESIRED CURRENT READY AGE

sise-deploy-3513442901 2 2 2 19s

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

sise-deploy-3513442901-cndsx 1/1 Running 0 25s

sise-deploy-3513442901-sn74v 1/1 Running 0 25s

Note the naming of the pods and replica set, derived from the deployment name.

At this point in time the sise containers running in the pods are configured to return the version 0.9. Let’s verify that from within the cluster (using kubectl describe first to get the IP of one of the pods):

[cluster] $ curl 172.17.0.3:9876/info

{"host": "172.17.0.3:9876", "version": "0.9", "from": "172.17.0.1"}

Let’s now see what happens if we change that version to 1.0 in an updated [deployment](https://github.com/openshift-evangelists/kbe/blob/master/specs/deployments/d10.yaml):

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/deployments/d10.yaml

deployment "sise-deploy" configured

Note that you could have used kubectl edit deploy/sise-deploy alternatively to achieve the same by manually editing the deployment.

What we now see is the rollout of two new pods with the updated version 1.0 as well as the two old pods with version 0.9 being terminated:

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

sise-deploy-2958877261-nfv28 1/1 Running 0 25s

sise-deploy-2958877261-w024b 1/1 Running 0 25s

sise-deploy-3513442901-cndsx 1/1 Terminating 0 16m

sise-deploy-3513442901-sn74v 1/1 Terminating 0 16m

Also, a new replica set has been created by the deployment:

$ kubectl get rs

NAME DESIRED CURRENT READY AGE

sise-deploy-2958877261 2 2 2 4s

sise-deploy-3513442901 0 0 0 24m

Note that during the deployment you can check the progress using kubectl rollout status deploy/sise-deploy.

To verify that if the new 1.0 version is really available, we execute from within the cluster (again using kubectl describe get the IP of one of the pods):

[cluster] $ curl 172.17.0.5:9876/info

{"host": "172.17.0.5:9876", "version": "1.0", "from": "172.17.0.1"}

A history of all deployments is available via:

$ kubectl rollout history deploy/sise-deploy

deployments "sise-deploy"

REVISION CHANGE-CAUSE

1 <none>

2 <none>

If there are problems in the deployment Kubernetes will automatically roll back to the previous version, however you can also explicitly roll back to a specific revision, as in our case to revision 1 (the original pod version):

$ kubectl rollout undo deploy/sise-deploy --to-revision=1

deployment "sise-deploy" rolled back

$ kubectl rollout history deploy/sise-deploy

deployments "sise-deploy"

REVISION CHANGE-CAUSE

2 <none>

3 <none>

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

sise-deploy-3513442901-ng8fz 1/1 Running 0 1m

sise-deploy-3513442901-s8q4s 1/1 Running 0 1m

At this point in time we’re back at where we started, with two new pods serving again version 0.9.

Finally, to clean up, we remove the deployment and with it the replica sets and pods it supervises:

$ kubectl delete deploy sise-deploy

deployment "sise-deploy" deleted

See also the [docs](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) for more options on deployments and when they are triggered.