Rolling back a deployment

You’re currently running version v2 of your image, so you’ll need to prepare version 3 first.

In version 3, you’ll introduce a bug that makes your app handle only the first four requests properly. All requests from the fifth request onward will return an internal server error (HTTP status code 500). You’ll simulate this by adding an `if` statement at the beginning of the handler function.

`v3/app.js`

```
const http = require("http");
const os = require("os");

var requestCount = 0;

console.log("Kubia server starting...");

var handler = function(request, response) {
  console.log("Received request from " + request.connection.remoteAddress);
  if (++requestCount >= 5) {
    response.writeHead(500);
    response.end(
      "Some internal error has occurred! This is pod " + os.hostname() + "\n"
    );
    return;
  }
  response.writeHead(200);
  response.end("This is v3 running in pod " + os.hostname() + "\n");
};

var www = http.createServer(handler);
www.listen(8080);
```

You'll deploy this new version by changing the image in the Deployment specification again

```
kubectl set image deployment kubia nodejs=luksa/kubia:v3
```

```
deployment.apps "kubia" image updated
```

You can follow the progress of the rollout with `kubectl rollout status`:

```
kubectl rollout status deployment kubia
```

```
Waiting for rollout to finish: 1 out of 3 new replicas have been upd
ated...
Waiting for rollout to finish: 1 out of 3 new replicas have been upd
ated...
Waiting for rollout to finish: 2 out of 3 new replicas have been upd
ated...
Waiting for rollout to finish: 2 out of 3 new replicas have been upd
ated...
Waiting for rollout to finish: 2 out of 3 new replicas have been upd
ated...
Waiting for rollout to finish: 2 out of 3 new replicas have been upd
ated...
Waiting for rollout to finish: 1 old replicas are pending terminatio
n...
Waiting for rollout to finish: 1 old replicas are pending terminatio
n...
Waiting for rollout to finish: 1 old replicas are pending terminatio
n...
Waiting for rollout to finish: 1 old replicas are pending terminatio
n...
deployment "kubia" successfully rolled out
```

Hit your broken version 3

```
while true; do curl 35.232.43.157:32229; done
```

```
This is v3 running in pod kubia-54c887cf4d-qtkpl
This is v3 running in pod kubia-54c887cf4d-rtm94
This is v3 running in pod kubia-54c887cf4d-q859g
This is v3 running in pod kubia-54c887cf4d-q859g
This is v3 running in pod kubia-54c887cf4d-qtkpl
This is v3 running in pod kubia-54c887cf4d-rtm94
This is v3 running in pod kubia-54c887cf4d-rtm94
This is v3 running in pod kubia-54c887cf4d-qtkpl
This is v3 running in pod kubia-54c887cf4d-qtkpl
Some internal error has occurred! This is pod kubia-54c887cf4d-qtkpl
This is v3 running in pod kubia-54c887cf4d-q859g
This is v3 running in pod kubia-54c887cf4d-q859g
Some internal error has occurred! This is pod kubia-54c887cf4d-q859g
This is v3 running in pod kubia-54c887cf4d-rtm94
Some internal error has occurred! This is pod kubia-54c887cf4d-rtm94
```

Deployments make it easy to roll back to the previously deployed version by telling Kubernetes to undo the last rollout of a Deployment

```
kubectl rollout undo deployment kubia
```

```
deployment.apps "kubia"
```

This rolls the Deployment back to the previous revision.

The `undo` command can also be used while the rollout process is still in progress to essentially abort the rollout. Pods already created during the rollout process are removed and replaced with the old ones again.

Rolling back a rollout is possible because Deployments keep a revision history. As you'll see later, the history is stored in the underlying ReplicaSets. When a rollout completes, the old ReplicaSet isn't deleted, and this enables rolling back to any revision, not only the previous one. The revision history can be displayed with the `kubectl rollout history` command:

```
kubectl rollout history deployment kubia
```

```
deployments "kubia"
REVISION  CHANGE-CAUSE
1         kubectl patch deployment kubia --patch={"spec": {"minReady
Seconds": 10}}
```

```
3         kubectl set image deployment kubia nodejs=luksa/kubia:v3
4         kubectl set image deployment kubia nodejs=luksa/kubia:v2
```

Remember the `--record` command-line option you used when creating the Deployment? Without it, the `CHANGE-CAUSE` column in the revision history would be empty, making it much harder to figure out what's behind each revision.

You can roll back to a specific revision by specifying the revision in the `undo` command. For example, if you want to roll back to the first version, you'd execute the following command:

```
kubectl rollout undo deployment kubia --to-revision=1
```

```
deployment.apps "kubia"
```

```
while true; do curl 35.232.43.157:32229; done
```

```
This is v1 running in pod kubia-6fc8ff6566-vh5zj
This is v1 running in pod kubia-6fc8ff6566-9j26j
This is v1 running in pod kubia-6fc8ff6566-9j26j
This is v1 running in pod kubia-6fc8ff6566-qwz7h
```

Each ReplicaSet stores the complete information of the Deployment at that specific revision, so you shouldn't delete it manually. If you do, you'll lose that specific revision from the Deployment's history, preventing you from rolling back to it.

But having old ReplicaSets cluttering your ReplicaSet list is not ideal, so the length of the revision history is limited by the `revisionHistoryLimit` property on the Deployment resource. It defaults to two, so normally only the current and the previous revision are shown in the history (and only the current and the previous ReplicaSet are preserved). Older ReplicaSets are deleted automatically.