Updating a Deployment

The only thing you need to do is modify the pod template defined in the Deployment resource and Kubernetes will take all the steps necessary to get the actual system state to what's defined in the resource. Similar to scaling a ReplicationController or ReplicaSet up or down, all you need to do is reference a new image tag in the Deployment's pod template and leave it to Kubernetes to transform your system so it matches the new desired state.

How this new state should be achieved is governed by the deployment strategy configured on the Deployment itself.

- The default strategy is to perform a rolling update (the strategy is called `RollingUpdate`)
- The alternative is the `Recreate` strategy, which deletes all the old pods at once and then creates new ones

The `Recreate` strategy causes all old pods to be deleted before the new ones are created. Use this strategy when your application doesn't support running multiple versions in parallel and requires the old version to be stopped completely before the new one is started. This strategy does involve a short period of time when your app becomes completely unavailable.

The `RollingUpdate` strategy, on the other hand, removes old pods one by one, while adding new ones at the same time, keeping the application available throughout the whole process, and ensuring there's no drop in its capacity to handle requests. This is the default strategy. The upper and lower limits for the number of pods above or below the desired replica count are configurable. You should use this strategy only when your app can handle running both the old and new version at the same time.

You'll use the `RollingUpdate` strategy, but you need to slow down the update process a little, so you can see that the update is indeed performed in a rolling fashion. You can do that by setting the `minReadySeconds` attribute on the Deployment. For now, set it to 10 seconds with the kubectl patch command.

```
kubectl patch deployment kubia -p '{"spec": {"minReadySeconds": 10}}
'
```

```
deployment.extensions "kubia" patched
```

The `kubectl patch` command is useful for modifying a single property or a limited number of properties of a resource without having to edit its definition in a text editor.

You used the patch command to change the spec of the Deployment. This doesn't cause any kind of update to the pods, because you didn't change the pod template. Changing other Deployment properties, like the desired replica count or the deployment strategy, also doesn't trigger a rollout, because it doesn't affect the existing individual pods in any way.

If you'd like to track the update process as it progresses, first run the curl loop again in another terminal to see what's happening with the requests

```
while true; do curl 35.232.43.157:32229; sleep 1; done
```

To trigger the actual rollout, you'll change the image used in the single pod container to `luksa/kubia:v2`. Instead of editing the whole YAML of the Deployment object or using the `patch` command to change the image, you'll use the `kubectl set image` command, which allows changing the image of any resource that contains a container (ReplicationControllers, ReplicaSets, Deployments, and so on). You'll use it to modify your Deployment like this:

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

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

If you've run the `curl` loop, you'll see requests initially hitting only the `v1` pods; then more and more of them hit the `v2` pods until, finally, all of them hit only the remaining `v2` pods, after all `v1` pods are deleted.

```
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v1 running in pod kubia-6fc8ff6566-7g8tg
This is v1 running in pod kubia-6fc8ff6566-bm5p6
This is v2 running in pod kubia-5857d5f9ff-dzp5f
This is v1 running in pod kubia-6fc8ff6566-7g8tg
This is v2 running in pod kubia-5857d5f9ff-dzp5f
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v2 running in pod kubia-5857d5f9ff-hg84w
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v2 running in pod kubia-5857d5f9ff-dzp5f
This is v2 running in pod kubia-5857d5f9ff-hg84w
This is v2 running in pod kubia-5857d5f9ff-dzp5f
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v2 running in pod kubia-5857d5f9ff-pkpqx
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v2 running in pod kubia-5857d5f9ff-hg84w
This is v2 running in pod kubia-5857d5f9ff-pkpqx
This is v2 running in pod kubia-5857d5f9ff-dzp5f
^C
```

> ***Warning*** *Be aware that if the pod template in the Deployment references a ConfigMap (or a Secret), modifying the ConfigMap will not trigger an update. One way to trigger an update when you need to modify an app's config is to create a new ConfigMap and modify the pod template so it references the new ConfigMap.*

You can still see the old ReplicaSet next to the new one if you list them:

```
kubectl get rs
```

```
NAME                DESIRED   CURRENT   READY     AGE
kubia-5857d5f9ff    3         3         3         5m
kubia-6fc8ff6566    0         0         0         24m
```