**Blue/green deployment to release a single service**

In this example, we release a new version of a single service using the blue/green deployment strategy.

**Steps to follow**

1. version 1 is serving traffic
2. deploy version 2
3. wait until version 2 is ready
4. switch incoming traffic from version 1 to version 2
5. shutdown version 1

**In practice**

# Deploy the first application

$ kubectl apply -f app-v1.yaml

# Test if the deployment was successful

$ curl $(minikube service my-app --url)

2018-01-28T00:22:04+01:00 - Host: host-1, Version: v1.0.0

# To see the deployment in action, open a new terminal and run the following

command:

$ watch kubectl get po

# Then deploy version 2 of the application

$ kubectl apply -f app-v2.yaml

# Wait for all the version 2 pods to be running

$ kubectl rollout status deploy my-app-v2 -w

deployment "my-app-v2" successfully rolled out

# Side by side, 3 pods are running with version 2 but the service still send

# traffic to the first deployment.

# If necessary, you can manually test one of the pod by port-forwarding it to

# your local environment.

# Once your are ready, you can switch the traffic to the new version by patching

# the service to send traffic to all pods with label version=v2.0.0

$ kubectl patch service my-app -p '{"spec":{"selector":{"version":"v2.0.0"}}}'

# Test if the second deployment was successful

$ service=$(minikube service my-app --url)

$ while sleep 0.1; do curl "$service"; done

# In case you need to rollback to the previous version

$ kubectl patch service my-app -p '{"spec":{"selector":{"version":"v1.0.0"}}}'

# If everything is working as expected, you can then delete the v1.0.0

# deployment

$ kubectl delete deploy my-app-v1

**Cleanup**

$ kubectl delete all -l app=my-app