QA Consulting

Kubernetes Deployments

TRAINING MATERIALS - MODULE HANDOUT

Contacts

team.qac.all.trainers@qa.com

www.consulting.qa.com

Contents

Overview

A *Deployment* in Kubernetes is a controller in which you can describe a desired state. We can use deployments for updates to Pods and ReplicaSets. A ReplicaSet is maintaining an amount of replicated Pods.
Deployments allows us to:

- Update the state of the pods, a newer image for instance.
- Rollout ReplicaSets, multiple instances of a service.
- Scale to meet required loads
- Easily remove entire ReplicaSets which could be thousands of Pods.

Managing Deployments with kubectl

We can use **kubectl** as a CLI client for communicating with the Kubernetes master. Deployments may be defined in configuration and applied through tools such as kubectl.

Apply

We can apply these configuration files with the apply command. This is a preffered way of creating the configuration initially as well, if you use create then you must delete and rescreate in future changes.

```
kubectl apply -f [CONFIG_FILE]
```

```
kubectl apply -f nginx-deployment.yaml
```

Create

Create can be used also in the same way however this will not apply new changes, but just create the desired state if it does not exist.

```
kubectl create -f [CONFIG_FILE]
```

```
kubectl create -f nginx-deployment.yaml
```

Delete

Deployment configurations can also be completely removed with the delete argument.

```
kubectl delete -f [CONFIG_FILE]
```

```
kubectl delete -f nginx-deployment.yaml
```

Deployment Configuration

Here we configure the properties that are going to be delivered to the Kubernetes master. Below is an example configuration that will create a ReplicaSet of 3 NGINX containers which would be in a file named **nginx-deployment.yaml** for instance. This is followed by some more information about what the properties in that file are accomplishing.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

metadata

This is where can define basic information about the deployment such as it's name. A label is also applied in the example so that the deployment can be referred to in specifications.

selector

The selector defines how the deployment will find which Pods it should manage, this refers to the label set in the pod template below. This is the most simple selection method.

template

The specification (**spec**) here is indicating that the Pods have just one container running in them. Port 80 has been allowed so that we can communicate with the pod. For the only container that is in the Pod, the NGINX image from Docker Hub is going to be used.

Tasks

Create an NGINX deployment with 3 replicas.
Make a change to the configuration to change the amount of replicas and apply it.
Make a change to the configuration to change the image that is used and apply it.