Quarkus - Deploying Knative Application to Kubernetes or OpenShift

This guide covers:

The deployment of a serverless application to Kubernetes using Knative.

This guide takes as input the application developed in the native application guide. So, you should have been able to package your application as a binary executable, copied it in a Docker image and run this image.

Depending on whether you are a *bare* Kubernetes user or an OpenShift user, pick the section you need. The OpenShift section leverages OpenShift build and route features which are not available in *bare* Kubernetes.

Prerequisites

For this guide you need:

- roughly 20 minutes
- having access to a Kubernetes cluster. Minikube is a valid option.
- having deployed Knative components on minikube
- having Skaffold CLI on your PATH

Solution

We recommend to follow the instructions in the next sections and build the application step by step. However, you can go right to the completed example.

Clone the Git repository: `git clone https://github.com/quarkusio/quarkus-quickstarts.git`, or download an archive.

The solution is located in the `getting-started-knative` directory.

Set up Kubernetes Cluster

The following script will start minikube and deploy Knative.

```
./bin/start-minikube.sh
eval $(minikube docker-env) ①
```

① Make the current Docker context to be that of minikube

Set up Nexus(Optional)

Nexus is used for caching maven artifacts so that Apache Maven builds are faster.

```
kubectl apply -f k8s/nexus.yaml
```

Wait for some time to have Nexus initialize and run. You can watch the status via `kubectl get pods -w`, use `Ctrl` + `c` to terminate the watch.

Why Skaffold ?

As vanilla Kubernetes does not have an easy and developer friendly way to build and deploy application to a local cluster like minikube, without the need to push the image to external container registry. We will be using Skaffold to help us build and deploy the Quarkus application onto Kubernetes.

Build and deploy application

> ⛔ The container image will not be pushed to a remote container registry, and hence the container image url has to be `dev.local`, to make Knative deploy it without trying to pull it from external container registry.

To run Knative Quarkus applications, we need to use the multi stage docker build; to build the Quarkus application container image and use it in Kubernetes application deployment.

The following command start a one time deployment of Quarkus application and runs starts the Knative service after successful container image build.

If you want to deploy a Quarkus JVM image (using HotSpot), then run the following command before running Skaffold:

```
cp src/main/docker/Dockerfile.jvm Dockerfile
```

If you want to deploy a Quarkus Native executable image (using GraalVM), then run the following command before running Skaffold:

```
cp src/main/docker/Dockerfile.native Dockerfile
```

It is very important to note that the Dockerfile in `src/main/docker` folder has been modified Dockerfiles to support multi stage Docker build.The multi stage Docker build helps in building and containerization of application with single Dockerfile.

```
skaffold run
```

As it will take a few minutes for the build and deployment to be completed you can watch the status using:

```
watch kubectl get pods
```

A successful Knative service deployment will show the following pods in the current namespace:

```
NAME                                     READY   STATUS    AGE
greeter-deployment-5749cc98fc-gs6zr      2/2     Running   10s
```

The deployment name could differ in your environment.

Once the Knative service is successfully running, you can call it using the script `./bin/call.sh`, which should return a response like `hello`. Allowing it to idle for approximately 60-65 seconds - that is without any further requests -, you will see it automatically scales down to zero pods.

Cleanup

To delete the application:

```
skaffold delete
```

Going further

This guide covered the deployment of a Quarkus application as Knative application on Kubernetes However, there is much more, and the integration with these environments has been tailored to make Quarkus applications execution very smooth. For instance, the health extension can be used for health check; the configuration support allows mounting the application configuration using config map, the metric extension produces data *scrapable* by Prometheus and so on.