

Quarkus - Deploying Knative Application to Kubernetes or OpenShift

This guide covers:

- The deployment of the application to Kubernetes

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 and/or OpenShift cluster. Minikube and Minishift are valid options.
- having deployed Knative components on [Minikube](#) or [Minishift](#)

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.

Deploying the application in Knative

Before we deploy the application to Knative in Minikube or Minishift we need to create the following Kubernetes objects:

- [Container registry secrets](#) :- This is required to for the built container image to be pushed to the container registry of your choice
- [Deploy Key](#) :- This is required only if you are going to pull the sources from private repository.

- **Build Service Account** :- The Kubernetes Service Account that will have access to Container Registry secret and Deploy Key secret



If you are not using private GitHub repo then you dont need the **Deploy Key** created and added to the Build Service Account

Run the following commands to kick start **Knative Build** which will build the quarkus application container image using Dockerfile and **Kaniko**. After a successful build you will have the Knative serving application deployed with the built container image. You can watch the application build pods using the command `kubectl get pods -w`. You can terminate the watch using the command `CTRL + C`

Since there are some parameters that you need to pass to the builds, which are right now configurable via maven properties, you need to run the following maven command to make them passed to the Knative resource yamls.

Table 1. Maven Parameters

| Name | Use | Example |
|-----------------------------|---|---|
| github.deploy.key | the base64 encoded private key that is configured to be used as the GitHub private repo Deploy key | <code>cat ~/.ssh/quarkus-quickstarts base64 -w 0</code> |
| github.keyscan | the base64 encoded value of <code>ssh-keyscan github.com</code> | <code>ssh-keyscan github.com base64 -w 0</code> |
| container.registry.url | the container registry url, NOTE: this should be a v2 container registry | https://index.docker.io/v1/ |
| container.registry.user | The user name to authenticate with the container registry | |
| container.registry.password | The user password to authenticate with the container registry | |
| git.source.revision | The revision of the source to checkout from GitHub | master |
| git.source.repo.url | The GitHub repo url | https://github.com/quarkusio/quarkus-quickstarts.git |
| app.container.image | The fully qualified name of the container image that will be pushed to the container registry after build | docker.io/demo/quarkus-knative-quickstart |

The following is the example command to generate the need knative resource files

```

mvn -Dgithub.deploy.key=$(cat ~/.ssh/quarkus-quickstarts | base64
-w 0) \
  -Dgithub.keyscan=$(ssh-keyscan github.com | base64 -w 0) \
  -Dcontainer.registry.url='https://quay.io/v2' \
  -Dcontainer.registry.user='demo' \
  -Dcontainer.registry.password='password' \
  -Dgit.source.revision='master' \
  -Dgit.source.repo.url='https://github.com/quarkusio/quarkus
-quickstarts.git' \ ①
  -Dapp.container.image='docker.io/demo/getting-started-knative'
\
  clean process-resources

```

① If you are using a private repo then you might need to use git ssh url

The above command will apply the property values to the Knative resources found in `${project.basedir}/src/main/knative` and copy them to `${project.build.directory}/knative`

Run the following command to create the Knative resources:

```
kubectl apply --recursive --filename target/knative
```

Accessing your application

The application is now exposed as an internal service. If you are using `minikube` or `minishift`, you can access it using:

```

INGRESSGATEWAY=istio-ingressgateway
IP_ADDRESS="$(minikube ip):$(kubectl get svc $INGRESSGATEWAY
--namespace istio-system --output
'jsonpath={.spec.ports[?(@.port==80)].nodePort}')" ①

curl -v -H 'Host: getting-started-knative.example.com'
$IP_ADDRESS/hello/greeting/redhat

```

① you can replace `minikube ip` with `minishift ip` if you are using OpenShift

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.