Quarkus - Deploying to Microsoft Azure Cloud

This guide covers:

- Update Quarkus HTTP Port

- Install the Azure CLI

- Create an Azure Registry Service instance and upload the Docker image

- Deploy the Docker image to Azure Container Instances

- Deploy the Docker image to Azure Kubernetes Service

- Deploy the Docker image to Azure App Service for Linux Containers

Prerequisites

For this guide you need:

- roughly 2 hours for all modalities

- having access to an Azure subscription. Get a free one here

This guide will take as input a native application developed in the building native image guide.

Make sure you have the getting-started application at hand, or 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` directory.

Change Quarkus HTTP Port

If you correctly followed the building native image guide, you should have a local container image named `quarkus-quickstart/getting-started`.

While Quarkus by default runs on port 8080, most Azure services expect web applications to be running on port 80. Before we continue, go back to your quickstart code and open the file `src/main/docker/Dockerfile.native`.

Change the last two commands in the `Dockerfile.native` file and make it read like this:

```
EXPOSE 80
CMD ["./application", "-Dquarkus.http.host=0.0.0.0", "-Dquarkus.http.port=80"]
```

Now you can rebuild the docker image:

```
$ docker build -f src/main/docker/Dockerfile.native -t quarkus-
quickstart/getting-started .
```

To test, run it by exposing port 80 into port 8080 in your host:

```
$ docker run -i --rm -p 8080:80 quarkus-quickstart/getting-started
```

Your container image is now ready to run on Azure. Remember, the Quarkus application is mapped to run on port 80.

Install the Azure CLI

To ease the user experience throughout this guide, it is better to have the Azure CLI installed and authenticated.

Visit the Azure CLI installation page for instructions specific to your operating system.

Once installed, ensure you are authenticated:

```
$ az login
```

Create an Azure Container Registry instance

It is possible to deploy images hosted on Docker Hub, but this location by default leaves images accessible to anyone. To better protect your container images, this guide shows how to host your images on a private instance of the Azure Container Registry service.

First, create an Azure Resource Group:

```
$ az group create --name <resource-group-name> --location eastus
```

Then you can create the ACR:

```
$ az acr create --resource-group <resource-group-name> --name
<registry-name> --sku Basic --admin-enabled true
```

Finally, authenticate your local Docker installation with this container registry by running:

```
$ az acr login --name <registry-name>
```

Upload Container Image on Azure

If you've followed the build native image guide, you should have a local container image named `quarkus-quickstart/getting-started`.

To upload this image to your ACR, you must tag and push the image under the ACR login server. To find the login server of the Azure Container Registry, run this command:

```
$ az acr show -n <registry-name> --query loginServer
```

To upload, now do:

```
$ docker tag quarkus-quickstart/getting-started <acr-login-server>/quarkus-quickstart/getting-started
$ docker push <acr-login-server>/quarkus-quickstart/getting-started
```

At this point, you should have your Quarkus container image on your Azure Container Registry. To verify, run the following command:

```
$ az acr repository list -n <registry-name>
```

Deploy to Azure Container Instances

The simplest way to start this container in the cloud is with the Azure Container Instances service. It simply creates a container on Azure infrastructure.

There are different approaches for using ACI. Check the documentation for details. The quickest way to get a container up and running goes as it follows.

First step is to find the username and password for the admin, so that ACI can authenticate into ACR and pull the Docker image:

```
$ az acr credential show --name <registry-name>
```

Now create the Docker instance on ACI pointing to your image on ACR:

```
$ az container create \
    --name quarkus-hello \
    --resource-group <resource-group> \
    --image <acr-login-server>/quarkus-quickstart/getting-started \
    --registry-login-server <acr-login-server> \
    --registry-username <acr-username> \
    --registry-password <acr-password> \
    --dns-name-label quarkus-hello-<random-number> \
    --query ipAddress.fqdn
```

The command above, if run successfully, will give you the address of your container in the Cloud. Access your Quarkus application in the address displayed as output.

For more information and details on ACR authentication and the use of service principals, follow this guide below and remember the Azure Container Registry `loginServer` and the image name of your Quarkus application now hosted on the ACR.

Deploy to Azure Container Instances from Azure Container Registry

Keep in mind that this service does not provide scalability. A container instance is unique and does not scale.

Deploy to Azure Kubernetes Service

You can also deploy the container image as a microservice in a Kubernetes cluster on Azure. To do that, follow this tutorial:

Tutorial: Deploy an Azure Kubernetes Service (AKS) cluster

Once deployed, the application will be running on whatever port is used to expose the service. By default, Quarkus apps run on port 8080 internally.

Deploy to Azure App Service on Linux Containers

This service provides scalability out of the box for web applications. If more instances are required, it will provide a load-balancing automatically, plus monitoring, metrics, logging and so on.

To deploy your Quarkus Native container image to this service, follow this tutorial:

Tutorial: Build a custom image and run in App Service from a private registry