Quarkus - Azure Functions (Serverless) with RESTEasy, Undertow, or Vert.x Web

The `quarkus-azure-functions-http` extension allows you to write microservices with RESTEasy (JAX-RS), Undertow (servlet), or Vert.x Web and make these microservices deployable to the Azure Functions runtime.

One azure function deployment can represent any number of JAX-RS, servlet, or Vert.x Web endpoints.

> This technology is considered preview.
>
> In *preview*, backward compatibility and presence in the ecosystem is not guaranteed. Specific improvements might require to change configuration or APIs and plans to become *stable* are under way. Feedback is welcome on our mailing list or as issues in our GitHub issue tracker.
>
> For a full list of possible extension statuses, check our FAQ entry.

Prerequisites

To complete this guide, you need:

- less than 15 minutes
- JDK 1.8 (Azure requires JDK 1.8)
- Apache Maven 3.5.3+
- An Azure Account. Free accounts work.
- Azure CLI Installed

Solution

This guide walks you through running a Maven Archetype to generate a sample project that contains three http endpoints written with JAX-RS APIs, Servlet APIs, and Vert.x Web APIs. After building, you will then be able to deploy to Azure.

Creating the Maven Deployment Project

Create the azure maven project for your Quarkus application using our Maven Archetype.

```
mvn archetype:generate \
    -DarchetypeGroupId=io.quarkus \
    -DarchetypeArtifactId=quarkus-azure-functions-http-archetype \
    -DarchetypeVersion=1.3.0.CR1
```

Running this command will run maven in interactive mode and it will ask you to fill in some build properties:

- `groupId` - The maven groupId of this generated project. Type in `org.acme`.

- `artifactId` - The maven artifactId of this generated project. Type in `quarkus-demo`

- `version` - Version of this generated project.

- `package` - defaults to `groupId`

- `appName` - Use the default value. This is the application name in Azure. It must be a unique subdomain name under `*.azurewebsites.net`. Otherwise deploying to Azure will fail.

- `appRegion` - Defaults to `westus`. Dependent on your azure region.

- `function` - Use the default which is `quarkus`. Name of your azure function. Can be anything you want.

- `resourceGroup` - Use the default value. Any value is fine though.

The values above are defined as properties in the generated `pom.xml` file.

Login to Azure

If you don't login to Azure you won't be able to deploy.

```
az login
```

Build and Deploy to Azure

The `pom.xml` you generated in the previous step pulls in the `azure-functions-maven-plugin`. Running maven install generates config files and a staging directory required by the `azure-functions-maven-plugin`. Here's how to execute it.

```
mvn clean install azure-functions:deploy
```

If you haven't already created your function up at azure, the will build an uber-jar, package it, create the function at Azure, and deploy it.

If deployment is a success, the azure plugin will tell you the base URL to access your function.

i.e.

```
Successfully deployed the artifact to https://quarkus-demo-
123451234.azurewebsites.net
```

The URL to access the service would be

https://{appName}.azurewebsites.net/api/hello   https://{appName}.azurewebsites.net/api/servlet/
hello https://{appName}.azurewebsites.net/api/vertx/hello

Extension maven dependencies

The sample project includes the RESTEasy, Undertow, and Vert.x Web extensions. If you are only using one of those APIs (i.e. jax-rs only), respectively remove the maven dependency `quarkus-resteasy`, `quarkus-undertow`, and/or `quarkus-vertx-web`.

You must include the `quarkus-azure-functions-http` extension as this is a generic bridge between the Azure Functions runtime and the HTTP framework you are writing your microservices in.

Azure Deployment Descriptors

Templates for Azure Functions deployment descriptors (`host.json`, `function.json`) are within the `azure-config` directory. Edit them as you need to. Rerun the build when you are ready.

NOTE: If you change the `function.json path` attribute or if you add a `routePrefix`, your jax-rs endpoints won't route correctly. See Configuring Root Paths for more information.

Configuring Root Paths

The default route prefix for an Azure Function is `/api`. All of your JAX-RS, Servlet, and Vert.x Web endpoints must explicitly take this into account.

If you modify the `path` or add a `routePrefix` within the `azure-config/function.json` deployment descriptor, your code must also reflect any prefixes you specify for your path.