Quarkus - Funqy Google Cloud Functions

The guide walks through quickstart code to show you how you can deploy Funqy functions to Google Cloud Functions.

As the Google Cloud Function Java engine is a new Beta feature of Google Cloud, this extension is flagged as experimental.

> This technology is considered experimental.
>
> In *experimental* mode, early feedback is requested to mature the idea. There is no guarantee of stability nor long term presence in the platform until the solution matures. 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 30 minutes
- JDK 11 (Google Cloud Functions requires JDK 11)
- Apache Maven 3.6.3
- A Google Cloud Account. Free accounts work.
- Cloud SDK CLI Installed

Login to Google Cloud

Login to Google Cloud is necessary for deploying the application and it can be done as follows:

```
gcloud auth login
```

At the time of this writing, Cloud Functions are still in beta so make sure to install the `beta` command group.

```
gcloud components install beta
```

The Quickstart

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

The solution is located in the `funqy-google-cloud-functions-quickstart` [directory](#).

Creating the Maven Deployment Project

Create an application with the `quarkus-funqy-google-cloud-functions` extension. You can use the following Maven command to create it:

```
mvn io.quarkus:quarkus-maven-plugin:1.8.0.Final:create \
    -DprojectGroupId=org.acme \
    -DprojectArtifactId=funqy-google-cloud-functions \
    -DclassName="org.acme.quickstart.GreetingResource" \
    -Dpath="/hello" \
    -Dextensions="funqy-google-cloud-functions"
```

Now, let's remove what's not needed inside the generated application:

- Remove the dependency `io.quarkus:quarkus-reasteasy` from your `pom.xml` file.
- Remove the generated `org.acme.quickstart.GreetingResource` class.
- Remove the `index.html` from `resources/META-INF/resources` or it will be picked up instead of your Function.
- Remove the existing tests.

The Code

There is nothing special about the code and more importantly nothing Google Cloud specific. Funqy functions can be deployed to many different environments and Google Cloud Functions is one of them.

Choose Your Function

Only one Funqy function can be exported per Google Cloud Functions deployment. If you only have one method annotated with `@Funq` in your project, then there is no worries. If you have multiple functions defined within your project, then you will need to choose the function within your Quarkus `application.properties`:

```
quarkus.funqy.export=greet
```

Alternatively, you can set the `QUARKUS_FUNQY_EXPORT` environment variable when you create the Google Cloud Function using the `gcloud` cli.

Build and Deploy

Build the project using maven.

```
./mvnw clean package
```

This will compile and package your code.

Create the function

In this example, we will create two background functions. Background functions allow to react to Google Cloud events like PubSub messages, Cloud Storage events, Firestore events, …

```java
import javax.inject.Inject;

import io.quarkus.funqy.Funq;
import io.quarkus.funqy.gcp.functions.event.PubsubMessage;
import io.quarkus.funqy.gcp.functions.event.StorageEvent;

public class GreetingFunctions {

    @Inject GreetingService service;  ①

    @Funq  ②
    public void helloPubSubWorld(PubsubMessage pubSubEvent) {
        String message = service.hello(pubSubEvent.data);
        System.out.println(pubSubEvent.messageId + " - " +
message);
    }

    @Funq  ③
    public void helloGCSWorld(StorageEvent storageEvent) {
        String message = service.hello("world");
        System.out.println(storageEvent.name + " - " + message);
    }

}
```

> **ℹ** Function return type can also be Mutiny reactive types.

1. Injection works inside your function.

2. This is a background function that takes as parameter a `io.quarkus.funqy.gcp.functions.event.PubsubMessage`, this is a convenient class to deserialize a PubSub message.

3. This is a background function that takes as parameter a `io.quarkus.funqy.gcp.functions.event.StorageEvent`, this is a convenient class to deserialize a Google Storage event.

> ℹ️ we provide convenience class to deserialize common Google Cloud event inside the `io.quarkus.funqy.gcp.functions.event` package. They are not mandatory to use, you can use any object you want.

As our project contains multiple function, we need to specify which function needs to be deployed via the following property inside our `application.properties`:

```
quarkus.funqy.export=helloPubSubWorld
```

Build and Deploy to Google Cloud

To build your application, you can package your application via `mvn clean package`. You will have a single JAR inside the `target/deployment` repository that contains your classes and all your dependencies in it.

Then you will be able to use `gcloud` to deploy your function to Google Cloud, the `gcloud` command will be different depending from which event you want to be triggered.

> ⚠️ The first time you launch the `gcloud beta functions deploy`, you can have the following error message:
>
> ```
> ERROR: (gcloud.beta.functions.deploy) OperationError:
> code=7, message=Build Failed: Cloud Build has not been
> used in project <project_name> before or it is
> disabled. Enable it by visiting
> https://console.developers.google.com/apis/api/cloudbui
> ld.googleapis.com/overview?project=<my-project> then
> retry.
> ```
>
> This means that Cloud Build is not activated yet. To overcome this error, open the URL shown in the error, follow the instructions and then wait a few minutes before retrying the command.

Background Functions - PubSub

Use this command to deploy to Google Cloud Functions:

```
gcloud beta functions deploy quarkus-example-funky-pubsub \
  --entry
-point=io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction \
  --runtime=java11 --trigger-resource hello_topic --trigger-event
google.pubsub.topic.publish \
  --source=target/deployment
```

The entry point always needs to be `io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction` as it will be this class that will bootstrap Quarkus.

The `--trigger-resource` option defines the name of the PubSub topic, and the `--trigger-event google.pubsub.topic.publish` option define that this function will be triggered by all message publication inside the topic.

To trigger an event to this function, you can use the `gcloud functions call` command:

```
gcloud functions call quarkus-example-funky-pubsub --data
'{"data":"Pub/Sub"}'
```

The `--data '{"data":"Hello, Pub/Sub"}'` option allow to specify the message to be send to PubSub.

Background Functions - Cloud Storage

Before deploying your function, you need to create a bucket.

```
gsutil mb gs://quarkus-hello
```

Then, use this command to deploy to Google Cloud Functions:

```
gcloud beta functions deploy quarkus-example-funky-storage \
  --entry
-point=io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction \
  --runtime=java11 --trigger-resource quarkus-hello --trigger-event
google.storage.object.finalize \
  --source=target/deployment
```

The entry point always needs to be `io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction` as it will be this class that will bootstrap Quarkus.

The `--trigger-resource` option defines the name of the Cloud Storage bucket, and the `--trigger-event google.storage.object.finalize` option define that this function will be

triggered by all new file inside this bucket.

To trigger an event to this function, you can use the `gcloud functions call` command:

```
gcloud functions call quarkus-example-funky-storage --data
'{"name":"test.txt"}'
```

The `--data '{"name":"test.txt"}'` option allow to specify a fake file name, a fake Cloud Storage event will be created for this name.

You can also simply add a file to Cloud Storage using the command line of the web console.

Testing locally

The easiest way to locally test your function is using the Cloud Function invoker JAR.

You can download it via Maven using the following command:

```
mvn dependency:copy \
   -Dartifact='com.google.cloud.functions.invoker:java-function
-invoker:1.0.0-beta1' \
   -DoutputDirectory=.
```

Before using the invoker, you first need to build your function via `mvn package`.

Then you can use it to launch your function locally, again, the command depends on the type of function and the type of events.

Background Functions - PubSub

For background functions, you launch the invoker with a target class of `io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction`.

```
java -jar java-function-invoker-1.0.0-beta1.jar \
   --classpath target/funqy-google-cloud-functions-1.0.0-SNAPSHOT-
runner.jar \
   --target io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction
```

> ❗ The `--classpath` parameter needs to be set to the previously packaged JAR that contains your function class and all Quarkus related classes.

Then you can call your background function via an HTTP call with a payload containing the event:

```
curl localhost:8080 -d '{"data":{"data":"world"}}'
```

This will call your PubSub background function with a PubSubMessage `{"data":"hello"}`.

Background Functions - Cloud Storage

For background functions, you launch the invoker with a target class of `io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction`.

```
java -jar java-function-invoker-1.0.0-beta1.jar \
  --classpath target/funqy-google-cloud-functions-1.0.0-SNAPSHOT-
runner.jar \
  --target io.quarkus.funqy.gcp.functions.FunqyBackgroundFunction
```

> **(!)** The `--classpath` parameter needs to be set to the previously packaged JAR that contains your function class and all Quarkus related classes.

Then you can call your background function via an HTTP call with a payload containing the event:

```
curl localhost:8080 -d '{"data":{"name":"text"}}'
```

This will call your PubSub background function with a Cloud Storage event `{"name":"file.txt"}`, so an event on the `file.txt` file.

What's next?

If you are looking for JAX-RS, Servlet or Vert.x support for Google Cloud Functions, we have it thanks to our Google Cloud Functions HTTP binding.