University of Bamberg

DSG-DSAM-M Lab

Session 10: Google Cloud Run functions (formerly Google Cloud Functions)

Google Cloud Run Functions (Function as a Service)

- Fully managed environment for different languages (Java, Python, Go, Ruby,..)
- Small, stateless functions which scale automatically on demand from zero
- Different triggers available
  - HTTP invocation (→ Task 1)
  - Events: Row added to a database table, File added to Cloud Storage, … (→ Task 2)
- Local Development possible

Task 1.1

0) Clone https://gitlab.rz.uni-bamberg.de/dsg-public/dsam/gcloud-function-templates and open `blueprint` in IntelliJ

1) Run it via the gradle task runFunction and invoke it via http://localhost:8080

2) Add a class `Model` with one field `name` and complete it with a constructor and Getter/Setter

4) Add the following library to work with JSON data:

```
implementation 'com.fasterxml.jackson.core:jackson-databind:2.18.2'
```

5) Use the following code to get the HTTP request body and cast it to a `Model` object:

```
Model m = new ObjectMapper().readValue(
            request.getReader()
                   .lines()
                   .collect(Collectors.joining()
          ), Model.class);
```

20 Minutes

Task 1.2

6) Create a bucket with a name of your choice via the Google Cloud Console

7) Add the Cloud storage client API

The libraries-bom ensures compatibility between Google Cloud libraries.

```
implementation platform('com.google.cloud:libraries-bom:26.53.0')
implementation 'com.google.cloud:google-cloud-storage'
```

The google-cloud-storage library provides the client API for interacting with Google Cloud Storage.

8) Use the following code to create a new file in the created bucket (adjust as needed ;)):

```
Storage storage = StorageOptions.getDefaultInstance().getService();

Map<String, String> metadata = new HashMap<>();
metadata.put("mail", "some-email@uni-bamberg.de");
BlobInfo info = BlobInfo.newBuilder("yourBucketNameHere", "hello.txt").setMetadata(metadata)
      .setContentType("text/plain").build();

String text = String.format("Hello %s", m.getName());
storage.create(info, text.getBytes());
```

9) Test your function by running it locally

20 Minutes

Task 1.3

10) Deploy your function to the cloud:

```
gcloud functions deploy file-creation \
  --project=dsg-dsam-wt2425 \
  --region=europe-west1 \
  --gen2 \
  --max-instances=10 \
  --memory=512Mi \
  --entry-point de.uniba.dsg.cloudfunction.Function \
  --runtime=java21 \
  --source=. \
  --trigger-http
```

--project: The Google Cloud project ID.
--region: The region where the function will be deployed.
--gen2: Use the second generation of Cloud Functions.
--max-instances: Maximum number of instances for scaling.
--memory: Memory allocated to the function (e.g., 512Mi).
--entry-point: The fully qualified name of the function class.
--runtime: The runtime environment (e.g., Java 21).
--source: The source code directory (current directory .).
--trigger-http: The function will be triggered by HTTP requests.

You might be prompted to activate some APIs so that the function can be successfully deployed!
And do not allow unauthenticated invocation!

11) Test it again – see hints on Slide 8

20 Minutes

Task 2.1

0) Import the other template: `mail-sending`

1) Change the file ending ".pdf" to ".txt" in the class Function

X) Use a mail provider of your choice and get the corresponding credentials to send via SMTP
One Option: SendGrid https://sendgrid.com

2) Go to the Secret Manager in the GCP console:
https://console.cloud.google.com/security/secret-manager

3) Create two secrets:
`mail-user`: `apikey`
`mail-password`:
`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`

4) Set the property `mail.user.address` in the `application.properties` file to your email id

Task 2.2

5) Deploy the function:

```
gcloud functions deploy mail-sending \
  --project=dsg-dsam-wt2425 \
  --region=europe-west3 \
  --gen2 \
  --max-instances=10 \
  --memory=512Mi \
  --entry-point de.uniba.dsg.cloudfunction.Function \
  --runtime=java21 \
  --source=. \
  --trigger-bucket=dsam-lab-2425 \
  --trigger-location=europe-west3 \
  --set-secrets 'MAIL_USER=mail-user:1,MAIL_PASSWORD=mail-password:1'
```

6) Invoke the function from task 1 again and see if you receive an email

It might be necessary to grant the service account used for running functions the role:
`Secret Manager Secret Accessor`
And the Default Storage Service Account needs the role:
`Pub/Sub Publisher`

curl examples to test your functions

```
curl http://localhost:8080

curl --header "Content-Type: application/json" \
     --request POST \
     --data '{"name": "yourName"}' \
     http://localhost:8080

gcloud auth print-identity-token

curl --header "Authorization: Bearer $(gcloud auth print-identity-token)" \
     --header "Content-Type: application/json" \
     --request POST \
     --data '{"name": "yourName"}' \
     https://some-region-project.cloudfunctions.net/file-creation
```