Google Cloud Run Functions

By Xander Sorensen

Table of contents

Background

Release Schedule

April 2008 – Google Cloud Launches

November 2014 – AWS Lambda

March 2017 – 1st Gen

February 2022 – 2nd Gen

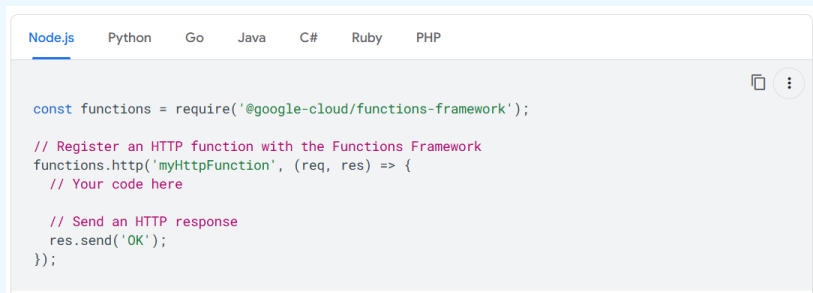August 2024 – Renamed to Cloud Run Functions

What is it?

Functions as a Service

- Runs code in response to an action
- Only pay for what you use
- Both http request and event driven

Supported Languages

- Node.js, Python, Go, Java, C#, Ruby, PHP

What is it?

Cloud Functions: the glue that connects cloud services

Cloud Storage
Firestore
Cloud Pub/Sub
HTTPS
Firebase
Cloud Tasks
Cloud Scheduler

Access 20+ Google services from GCF

Firebase   Assistant   DialogFlow

Creating Functions

1. Create directory to hold your function code
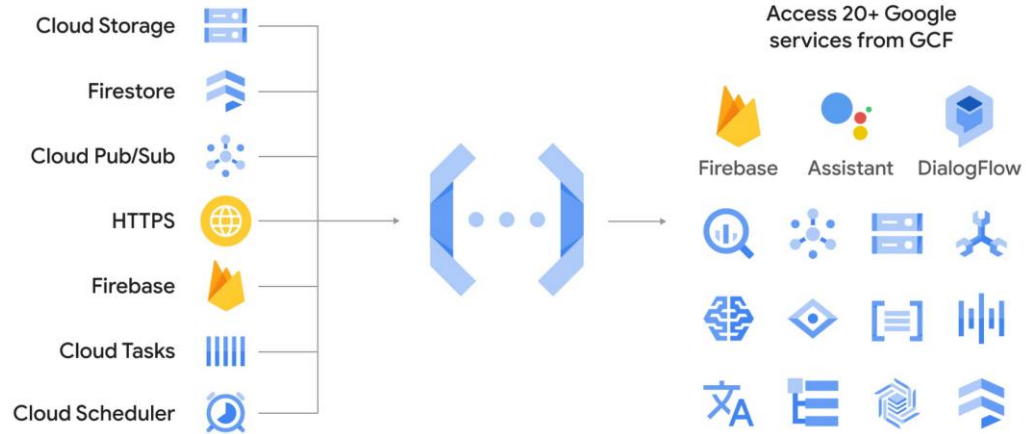
2. Write your function in an index.js file in the directory

```javascript
const functions = require('@google-cloud/functions-framework');

// Register an HTTP function with the Functions Framework that will be executed
// when you make an HTTP request to the deployed function's endpoint.
functions.http('helloGET', (req, res) => {
  res.send('Hello World!');
});
```

3. Add dependency to package.json

```json
"dependencies": {
  "@google-cloud/functions-framework": "^3.1.0"
}
```

Testing Functions

1. Install package

```
npm install @google-cloud/functions-framework
```

2. Run function locally

```
npx @google-cloud/functions-framework --target=helloGET
```

3. Access resource at localhost port 8080

CI/CD

Easy Deployment

- Deployment utilizes Googles GCloud CLI
- Simple to deploy with a shell script

```
☰ deploy.sh  ✕

analyze-image  ›  ☰ deploy.sh
   1    gcloud functions deploy analyzeImage \
   2    --runtime nodejs10 \
   3    --trigger-resource critterwatcher-uploads \
   4    --trigger-event google.storage.object.finalize
```

Traffic Splitting

- Enables deploying multiple versions of a function and splitting traffic between them
- Useful for A/B testing and validating updates

```
gcloud functions deploy hello-node-function \
   --gen2 \
   --runtime=nodejs20 \
   --region=REGION ✏ \
   --source=. \
   --entry-point=helloGET \
   --trigger-http \
   --allow-unauthenticated
```

```
gcloud run services update-traffic hello-world-colored \
   --region $REGION \
   --to-revisions hello-world-colored-0000X-XXX=50,hello-world-colored-0000X-XXX=50
```

Advantages and Disadvantages

Pros	Cons
Popular Language Support	Have to pay for unused minimum instances even when requests are not being made
Max run time of 60 minutes	Cannot handle extremely long or resource intensive tasks
Max 4 cores, 16 Gb of memory per run	Cannot modify every detail of the container your code runs on
Logging through Eventarc	
Can specify a minimum number of instances to run at all times, reducing cold start times	
Local development	

Which Projects Could Use These

Google Cloud is Your Cloud Provider

- Use Cloud Run Functions
- Integrate seamlessly with the rest of the Google suite

You Use Another Cloud Provider

- Use your provider's version of functions as a service
- They likely integrate better with your stack than forcing Cloud Run Functions to work

Our Project

- No
- Fully Microsoft in house
- Already using Azure for cloud infrastructure

Sources

Source	Link
Google Cloud Release Notes	https://cloud.google.com/functions/docs/release-notes
Google Cloud Docs	https://cloud.google.com/functions/docs/calling
Cloud Functions vs. Cloud Run	https://www.youtube.com/watch?v=zRjOSxTpC3A
Cloud Functions vs. Cloud Run: when to use one over the other	https://cloud.google.com/blog/products/serverless/cloud-run-vs-cloud-functions-for-serverless

Thanks!

Do you have any questions?