In28 Minutes

Google Cloud Functions

Cloud Functions

- Imagine you want to **execute some code when an event happens?**
  - A file is uploaded in Cloud Storage (OR) An error log is written to Cloud Logging (OR) A message arrives to Cloud Pub/Sub (OR) A http/https invocation is received
- Enter **Cloud Functions**
  - **Run code in response to events**
    - Write your business logic in Node.js, Python, Go, Java, .NET, and Ruby
    - **Don't worry** about servers or scaling or availability (only worry about your code)
  - **Pay only for what you use**
    - Number of invocations
    - Compute time of the invocations
    - Memory and CPU provisioned
  - **Time Bound** - Default 1 min and MAX 60 minutes (3600 seconds)
  - **2 product versions**
    - Cloud Functions (1st gen): First version
    - Cloud Functions (2nd gen): New version built on top of Cloud Run and Eventarc

Cloud Functions

Cloud Functions - Concepts

- **Event** : Upload object to cloud storage
- **Trigger**: Respond to event with a Function call
  - **Trigger** - Which function to trigger when an event happens?
  - **Functions** - Take event data and perform action?
- Events are **triggered from**
  - Cloud Storage
  - Cloud Pub/Sub
  - HTTP POST/GET/DELETE/PUT/OPTIONS
  - Firebase
  - Cloud Firestore
  - Stack driver logging

Cloud Functions

Example Cloud Function - HTTP - Node.js

```
const escapeHtml = require('escape-html');

exports.helloHttp = (req, res) => {
  res.send(`Hello ${escapeHtml(req.query.name || req.body.name || 'World')}!`);
};
```

Example Cloud Function - Pub/Sub - Node.js

```
/**
 * Background Cloud Function to be triggered by Pub/Sub.
 * This function is exported by index.js, and executed when
 * the trigger topic receives a message.
 *
 * @param {object} message The Pub/Sub message.
 * @param {object} context The event metadata.
 */
exports.helloPubSub = (message, context) => {
  const name = message.data
 ? Buffer.from(message.data, 'base64').toString()
 : 'World';

  console.log(`Hello, ${name}!`);
};
```

Cloud Functions - Remember

- No Server Management: You dont need to worry about scaling or availability of your function
- Cloud Functions automatically spin up and back down in response to events
  - They scale horizontally!
- Cloud Functions are recommended for responding to events:
  - Cloud Functions are NOT ideal for long running processes
    - **Time Bound** - Default 1 min and MAX 60 minutes(3600 seconds)

Cloud Functions

In28 Minutes

Cloud Run & Cloud Run for Anthos

- **Cloud Run** - "Container to Production in Seconds"
  - Built on top of an open standard - **Knative**
  - **Fully managed** serverless platform for containerized applications
    - ZERO infrastructure management
    - Pay-per-use (For used CPU, Memory, Requests and Networking)
- Fully integrated **end-to-end developer experience:**
  - **No limitations** in languages, binaries and dependencies
  - Easily portable because of **container** based architecture
  - Cloud Code, Cloud Build, Cloud Monitoring & Cloud Logging Integrations
- **Anthos** - Run Kubernetes clusters anywhere
  - Cloud, Multi Cloud and On-Premise
- **Cloud Run for Anthos:** Deploy your workloads to Anthos clusters running on-premises or on Google Cloud
  - Leverage your existing Kubernetes investment to quickly run serverless workloads

Cloud Run

Cloud Functions - Second Generation - What's New?

- **2 Product Versions:**
  - Cloud Functions (1st gen): First version
  - Cloud Functions (2nd gen): New version built on top of Cloud Run and Eventarc
- **Recommended**: Use Cloud Functions (2nd gen)
- **Key Enhancements in 2nd gen**:
  - **Longer Request timeout**: Up to 60 minutes for HTTP-triggered functions
  - **Larger instance sizes**: Up to 16GiB RAM with 4 vCPU (v1: Up to 8GB RAM with 2 vCPU)
  - **Concurrency**: Upto 1000 concurrent requests per function instance (v1: 1 concurrent request per function instance)
  - **Multiple Function Revisions** and **Traffic splitting** supported (v1: NOT supported)
  - Support for **90+ event types** - enabled by Eventarc (v1: Only 7)
- DEMO!

Cloud Functions - Scaling and Concurrency

- **Typical serverless functions architecture:**
  - **Autoscaling** - As new invocations come in, new function instances are created
  - One function instance handles ONLY ONE request AT A TIME
  - 3 concurrent function invocations => 3 function instances
    - If a 4th function invocation occurs while existing invocations are in progress, a new function instance will be created
    - HOWEVER, a function instance that completed execution may be reused for future requests
  - (Typical Problem) **Cold Start**:
    - New function instance initialization from scratch can take time
    - (Solution) Configure Min number of instances (increases cost)
- **1st Gen** uses the typical serverless functions architecture
- **2nd Gen** adds a very important new feature:
  - One function instance can handle multiple requests AT THE SAME TIME
    - **Concurrency**: How many concurrent invocations can one function instance handle? (Max 1000)
    - (IMPORTANT) Your function code should be safe to execute concurrently

Cloud Functions

Cloud Functions - Deployment using gcloud

- `gcloud functions deploy [NAME]`
  - **--docker-registry** (registry to store the function's Docker images)
    - Default - `container-registry`
    - Alternative - `artifact-registry`
  - **--docker-repository** (repository to store the function's Docker images)
    - Example: (`projects/${PROJECT}/locations/${LOCATION}/repositories/${REPOSITORY}`)
  - **--gen2** (Use 2nd gen. If this option is not present, 1st gen will be used)
  - **--runtime** (nodejs, python, java,...)
    - Reference - ***https://cloud.google.com/functions/docs/runtime-support***
  - **--service-account** (Service account to use)
    - 1 GEN - default - App Engine default service account - `PROJECT_ID@appspot.gserviceaccount.com`
    - 2 GEN - Default compute service account - `PROJECT_NO-compute@developer.gserviceaccount.com`
  - **--timeout** (function execution timeout)
  - **--max-instances** (function execution exceeding max-instances times out)
  - **--min-instances** (avoid cold starts at higher cost)

Cloud Functions - Deployment using gcloud - 2

```
//Deploy Pubsub Triggered gen2 function from Cloud Storage Bucket
gcloud functions deploy my-pubsub-function \
  --gen2 \
  --region=europe-west1 \
  --runtime=nodejs16 \
  --source=gs://my-source-bucket/source.zip \
  --trigger-topic=my-pubsub-topic
```

- **`gcloud functions deploy [NAME]`**
  - **--source**
    - Zip file from Google Cloud Storage (`gs://my-source-bucket/my_function_source.zip`) (OR)
    - Source Repo (`https://URL/projects/${PROJECT}/repos/${REPO}`) (OR)
    - Local file system
  - **--trigger-bucket** (OR) **--trigger-http** (OR) **--trigger-topic** (OR) **--trigger-event-filters** (ONLY in gen2 - Eventarc matching criteria for the trigger)
  - **--serve-all-traffic-latest-revision** (ONLY in gen2)

Cloud Functions - Best Practices

- To avoid cold starts, set **min no of instances** (increases cost)
  - Minimize dependencies (loading dependencies increases initialization time)
- Configure **max no of instances** (protect from abnormally high request levels)
- Use Cloud Endpoints (or Apigee or API gateway) for versioning
- Use Cloud Run (& Cloud Functions gen 2) **revisions** for safer releases:
  - Configure which revisions should receive traffic and how much
  - You can rollback to a previous revision, if needed
- Use Secret Manager to securely store secrets (ex: API keys)
- Use **Individual Service Accounts** for each function
  - Grant `roles/cloudfunctions.invoker` role to invoke a cloud function
- Manage dependencies using your language specific tool (npm, pip,..)