

Cloud Computing Serverless Functions

CIS437

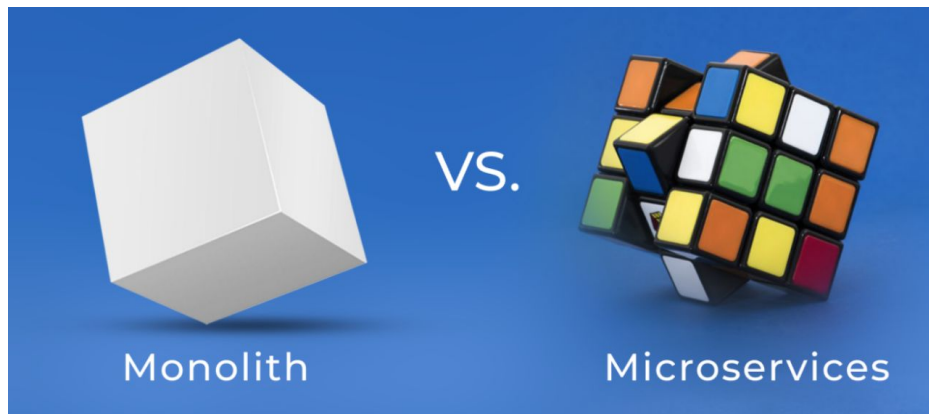
Erik Fredericks // frederer@gvsu.edu

Adapted from Google Cloud Computing Foundations, Overview of Cloud Computing (Wufka & Canonico)



Last time we made microservices

What was that again?



Last time we made microservices

Now let's make them even tinier!

Serverless functions!

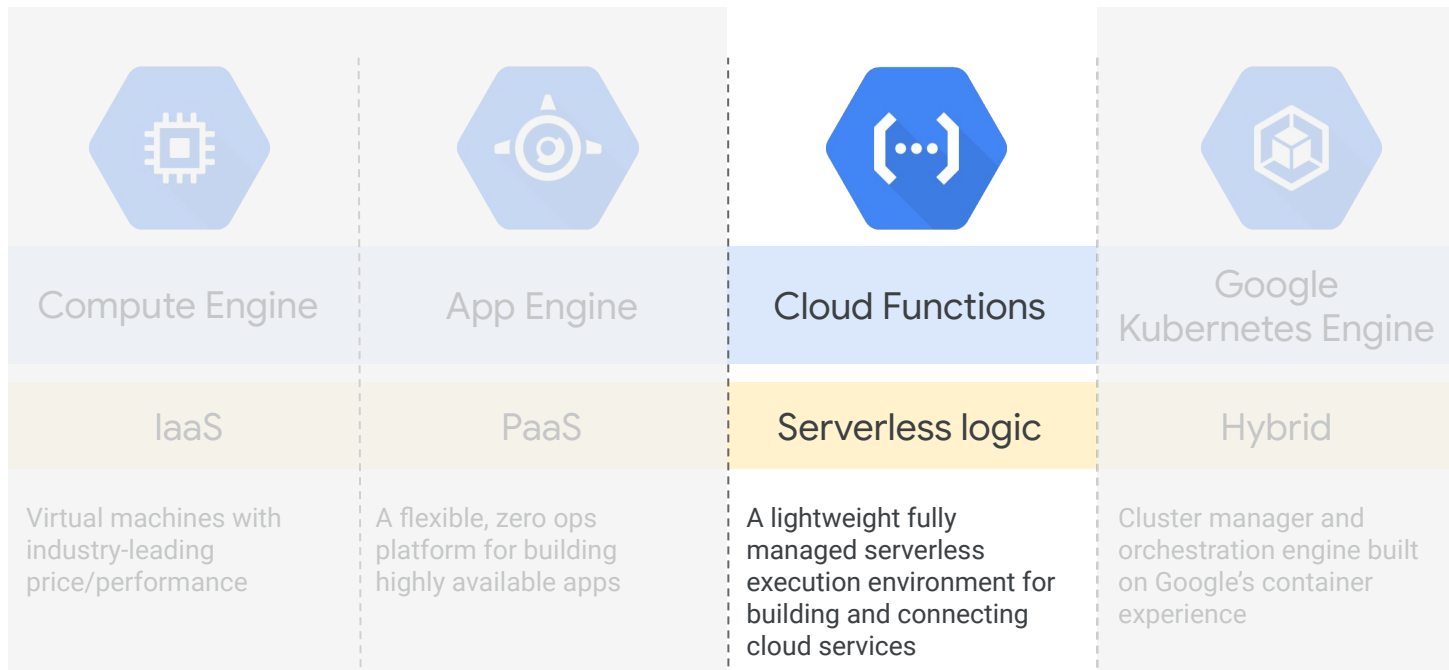
- Google Cloud Functions
- AWS Lambda Functions
- Microsoft Azure Functions

(i.e., probably my favorite cloud service to play with)

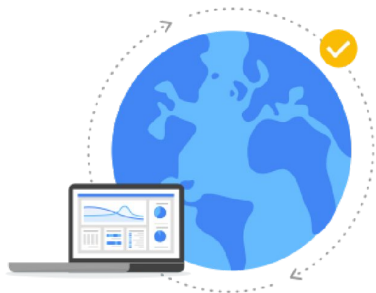
Question

Can a microservice be a serverless function?

Where Cloud Functions fits within Google Cloud



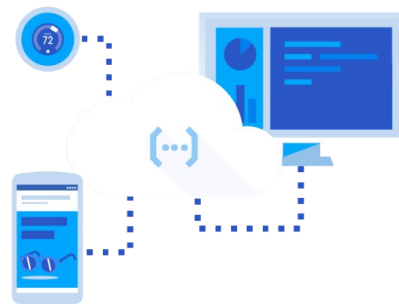
The components that make Cloud Functions work



Connect and extend
cloud services

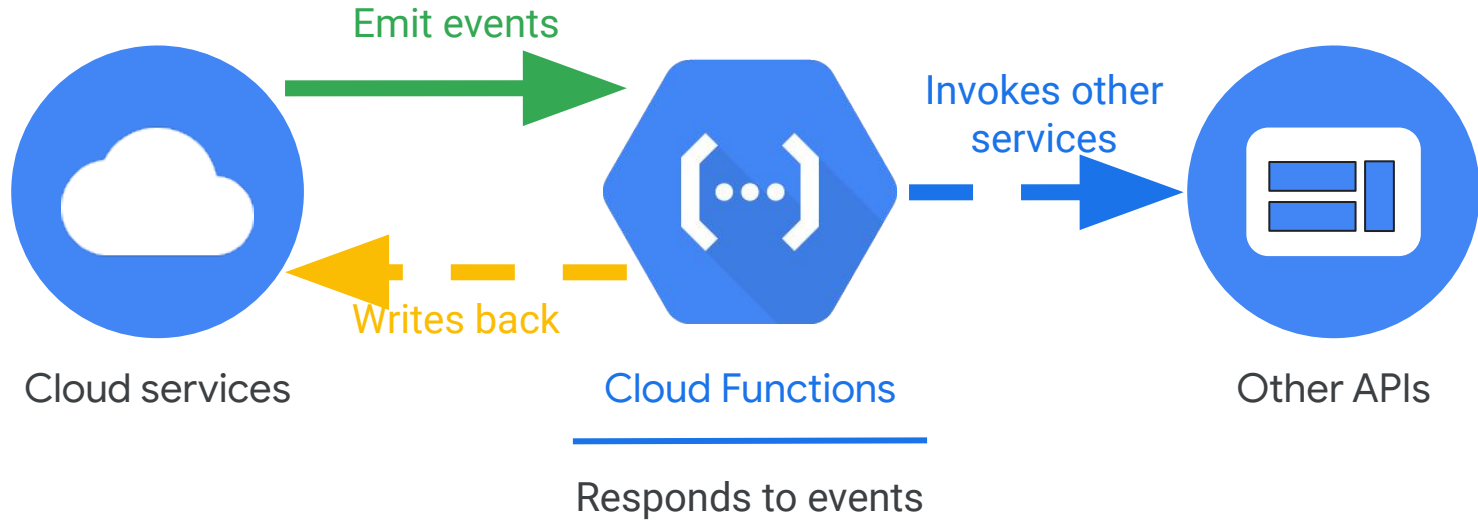


Events and triggers



Serverless

How Cloud Functions works



(I was also going to include some AWS slides but)

The concepts are near-identical



And so...

What are some applications you could use serverless functions with?

Before we dive in, considerations!

Again, with FaaS:

- No server, so no configuration
- No OS, so no setup
- Only a function, dependencies, and triggers

Practical considerations

Serverless functions typically have:

- 1) Limited execution time
- 2) Maximum amount of memory usage
- 3) Cost *per invocation*
- 4) Who is running the function?

Much like everything we've seen, these values can and will change over time

- SO IT IS UP TO YOU THE CLOUD ADMIN TO KEEP AN EYE ON THINGS

cloud fun

Search

2

LEARN

Cloud Functions

Create function

URL

https://us-central1-cloud-apps-demos-w24.cloudfunctions.net/function-3

Authentication

☐ Allow unauthenticated invocations

Check this if you are creating a public API or website.

☒ Require authentication

Manage authorized users with Cloud IAM.

Runtime, build, connections and security settings

RUNTIME

BUILD

CONNECTIONS

SECURITY AND

Memory allocated +

128 MiB

256 MiB

512 MiB

1 GiB

2 GiB

4 GiB

8 GiB

16 GiB

CPU +

0.167

seconds

Minimum number of instances

0

Maximum number of instances

100

Runtime service account

Service account *

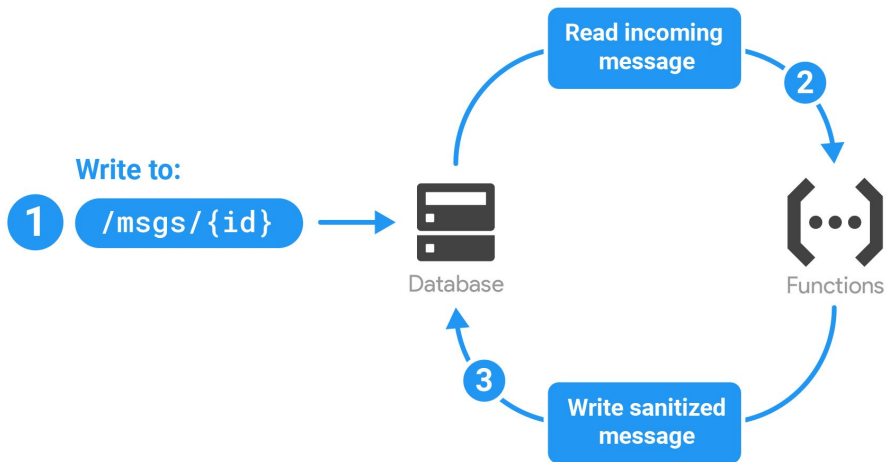
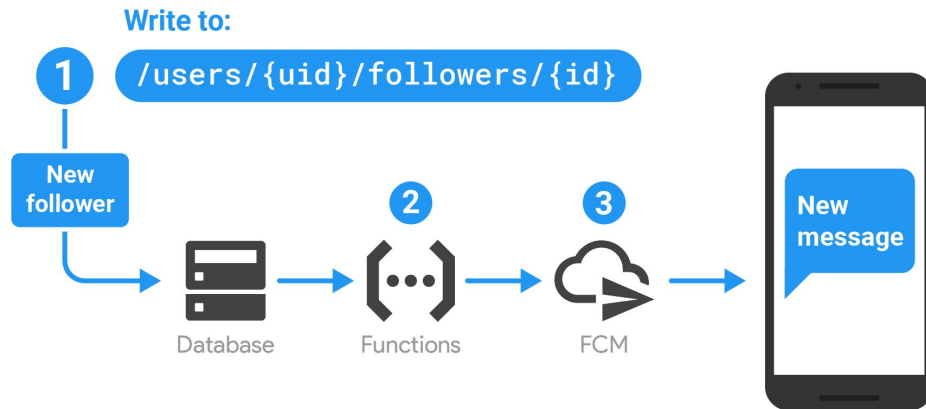
Compute Engine default service account

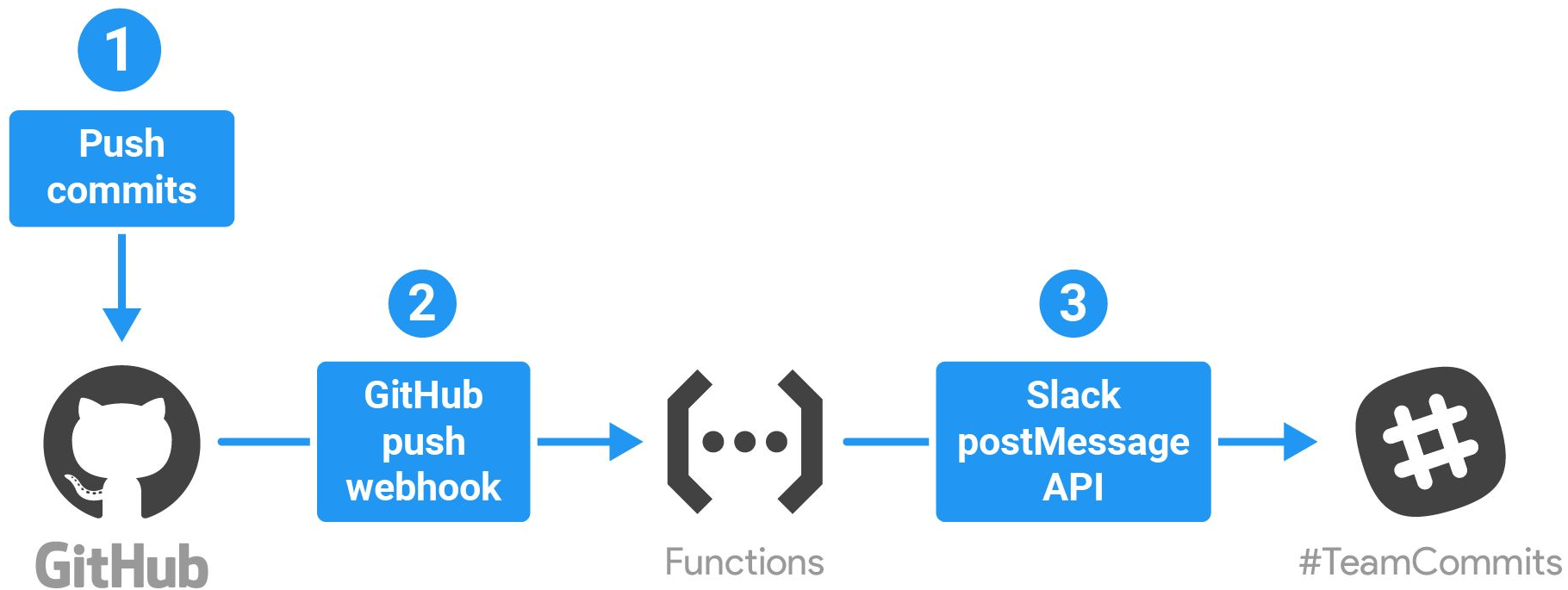
By default Cloud Functions uses the automatically created Default Compute Engine Service Account. [Learn more about service accounts.](#)

NEXT

CANCEL

Examples!





Let's create one and really step through it

This is literally, going to be the most important serverless function you ever create

Shall we start with the demonstration in the terminal?

We are going to use a publicly-available API to retrieve some *mission-critical* data

In the terminal (assuming you have `curl` installed)

```
$ curl -H "Accept: application/json" https://icanhazdadjoke.com/
```


The Cloud Function

You'll notice that you have access to a lot of different languages + versions

I'm going to use Python 3.12 - you can use whatever you want

- We're going to need to replicate the `curl` command, however, so you'll need to figure that out for your particular language

First...

Unauthenticated → ANYBODY ON THE INTERNET
CAN ACCESS IT!

- Pros/Cons?

Require authentication → ONLY THOSE WITH
ACCESS CAN ... ACCESS IT

- Pros/Cons?

Google Cloud cloud-apps-demos-w24

Cloud Functions Create function

1 Configuration — 2 Code

Basics

Environment
2nd gen

Function name *
dadjoke

Region *
us-central1 (Iowa)

Trigger

Trigger type
HTTPS

URL
https://us-central1-cloud-apps-demos-w24.cloudfunctions.net/dadjoke

Authentication

☒ Allow unauthenticated invocations
Check this if you are creating a public API or website.

☐ Require authentication
Manage authorized users with Cloud IAM.

Runtime, build, connections and security settings

NEXT CANCEL

jokes.py

We'll need the `requests` library

In `requirements.txt`, tell the CF that we need a particular version

```
requests==2.32.3
```

jokes.py

And update the default code:



```
import functions_framework
import requests

@functions_framework.http
def hello_http(request):
    url = "https://icanhazdadjoke.com"
    headers = {'Accept': 'application/json'}
    r = requests.get(url, headers=headers)
    return r.json()
```

**ALWAYS USE THIS TO
CHECK FOR
ERRORS!!!**


TEST FUNCTION

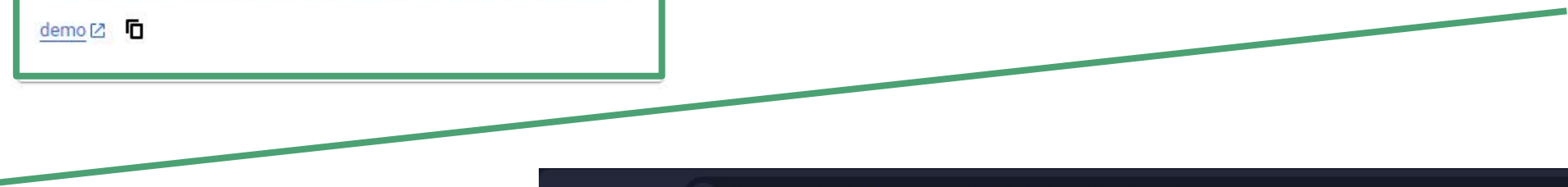
And then...


✓ dadjoke-demo 2nd gen (Deployed at Aug 15, 2024, 4:14:49 PM) URL: <https://us-central1-cloud-apps-demos-w24.cloudfunctions.net/dadjoke-demo>  

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS **LOGS** TESTING

🕒 HTTPS ?

URL
[https://us-central1-cloud-apps-demos-w24.cloudfunctions.net/dadjoke-](https://us-central1-cloud-apps-demos-w24.cloudfunctions.net/dadjoke-demo)
[demo](https://us-central1-cloud-apps-demos-w24.cloudfunctions.net/dadjoke-demo) 



← → ↻  us-central1-cloud-apps-demos-w24.cloudfunctions.net/dadjoke-demo

Pretty-print ☐

```
{ "id": "xHY8xsHYTnb", "joke": "What lies at the bottom of the ocean and twitches? A nervous wreck.", "status": 200 }
```

Let's make it purty

What's wrong?

- Just returning the entire JSON object

So...

```
joke = r.json()['joke']  
return f"<h1>{joke}</h1>"
```

<https://cloud.google.com/functions/pricing>

Simple Event-Driven Function

A simple event-driven function with 128MB of memory and a 200MHz CPU, invoked 10 million times per month and running for 300ms each time using only Google APIs (no billable outbound data transfer).

Calculations

Invocations

10,000,000

Compute Time

$(128\text{ MB} / 1024\text{ MB/GB}) \times 0.3\text{ s} = 0.0375\text{ GB-seconds per invocation}$

$(200\text{ MHz} / 1000\text{ MHz/GHz}) \times 0.3\text{ s} = 0.0600\text{ GHz-seconds per invocation}$

$10,000,000\text{ invocations} \times 0.0375\text{ GB-seconds} = 375,000\text{ GB-seconds per month}$

$10,000,000\text{ invocations} \times 0.0600\text{ GHz-seconds} = 600,000\text{ GHz-seconds per month}$

Networking

None

| Metric | Gross Value | Free Tier | Net Value | Unit Price | Total Price |
|---------------|-------------|-----------|-----------|-------------|-------------|
| Invocations | 10,000,000 | 2,000,000 | 8,000,000 | \$0.0000004 | \$3.20 |
| GB-seconds | 375,000 | 400,000 | < 0 | \$0.0000025 | \$0.00 |
| GHz-seconds | 600,000 | 200,000 | 400,000 | \$0.0000100 | \$4.00 |
| Networking | 0 | 5 | 0 | \$0.12 | \$0.00 |
| Total / Month | | | | | \$7.20 |

If you pay in a currency other than USD, the prices listed in your currency on [Cloud Platform SKUs](#) apply.

High Volume HTTP Function

A medium complexity HTTP Function with 256MB of memory and a 400MHz CPU, invoked 50 million times per month via HTTP, running for 500ms each time and sending 5KB of data back to the caller (5KB outbound data transfer per invocation).

Calculations

Invocations

50,000,000

Compute Time

$(256 \text{ MB} / 1024 \text{ MB/GB}) \times 0.5\text{s} = 0.125 \text{ GB-seconds per invocation}$

$(400 \text{ MHz} / 1000 \text{ MHz/GHz}) \times 0.5\text{s} = 0.200 \text{ GHz-seconds per invocation}$

50,000,000 invocations \times 0.125 GB-seconds = 6,250,000 GB-seconds per month

50,000,000 invocations \times 0.200 GHz-seconds = 10,000,000 GHz-seconds per month

Networking

50,000,000 invocations \times (5 KB / 1024 KB/MB / 1024 MB/GB) = 238.42 GB of outbound data transfer per month

| Metric | Gross Value | Free Tier | Net Value | Unit Price | Total Price |
|---------------|-------------|-----------|------------|-------------|-------------|
| Invocations | 50,000,000 | 2,000,000 | 48,000,000 | \$0.0000004 | \$19.20 |
| GB-seconds | 6,250,000 | 400,000 | 5,850,000 | \$0.0000025 | \$14.63 |
| GHz-seconds | 10,000,000 | 200,000 | 9,800,000 | \$0.0000100 | \$98.00 |
| Networking | 238.42 | 5 | 233.42 | \$0.12 | \$28.01 |
| Total / Month | | | | | \$159.84 |

If you pay in a currency other than USD, the prices listed in your currency on [Cloud Platform SKUs](#) apply.

FOR BALANCE

<https://aws.amazon.com/lambda/pricing/>

AWS Lambda Pricing

Region: US East (Ohio) ▾

| Architecture | Duration | Requests |
|---------------------------------------|------------------------------------|------------------------|
| x86 Price | | |
| First 6 Billion GB-seconds / month | \$0.0000166667 for every GB-second | \$0.20 per 1M requests |
| Next 9 Billion GB-seconds / month | \$0.000015 for every GB-second | \$0.20 per 1M requests |
| Over 15 Billion GB-seconds / month | \$0.0000133334 for every GB-second | \$0.20 per 1M requests |
| Arm Price | | |
| First 7.5 Billion GB-seconds / month | \$0.0000133334 for every GB-second | \$0.20 per 1M requests |
| Next 11.25 Billion GB-seconds / month | \$0.0000120001 for every GB-second | \$0.20 per 1M requests |
| Over 18.75 Billion GB-seconds / month | \$0.0000106667 for every GB-second | \$0.20 per 1M requests |

Duration cost depends on the amount of memory you allocate to your function. You can allocate any amount of memory to your function between 128 MB and 10,240 MB, in 1 MB increments. The table below contains a few examples of the price per 1 ms associated with different memory sizes, for usage falling within the first pricing tier – for example, up to 6 billion GB-seconds / month in US East (Ohio).

| x86 Price | Arm Price |
|--------------------------|----------------|
| Region: US East (Ohio) ▾ | |
| Memory (MB) | Price per 1ms |
| 128 | \$0.0000000021 |
| 512 | \$0.0000000083 |
| 1024 | \$0.0000000167 |
| 1536 | \$0.0000000250 |
| 2048 | \$0.0000000333 |
| 3072 | \$0.0000000500 |
| 4096 | \$0.0000000667 |

Interestingly, you can build them locally as well

functions_framework:

<https://cloud.google.com/functions/docs/functions-framework>

→ <https://github.com/GoogleCloudPlatform/functions-framework-python>

Build your cloud functions locally, test them, and then deploy them when you're ready

...why do this?

Local functions_framework

(Note: AWS Lambdas have a similar framework you can install)

(<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-using-debugging.html>)

Considerations!

Cold start vs. warm start

- Serverless functions *still* need something to run on, right?
- They essentially run on VMs that aren't necessarily always on

COLD STARTS



Tips

<https://cloud.google.com/functions/docs/bestpractices/tips>

(Note - out of all the things that are in flux with cloud apps, this might be the "most" in flux given that it depends on things behind the scenes that are not well-publicized)

(Almost like guessing how the YouTube algorithm works)

(Important) Tips

Idempotent functions

- Functions should **always** return the same output given the same input
 - Even if they use random...

HTTP functions must send HTTP response

- If no HTTP response, then a timeout can happen
- Leaving you ... waiting forever (and getting into retry loops)

Dependencies

- Minimize them if possible!
- They need to be installed/loaded **each** time the function cold-starts

More suggestions

Set a *minimum number of instances*

- How many must be "on call" to handle request
- Minimize cold start!

Ignoring the global variable suggestions as those seem - anti-patternish

- E.g., depending on globals to be remembered

Triggers

Different triggers other than HTTPS!

- Pub/Sub, some event
- Here, change in a cloud storage bucket

Trigger

Trigger type

Cloud Storage

▼

- Note: must be in same region as bucket

What could we do with this?

Step 1) File uploaded to Cloud Storage

...?

Triggered!

<https://cloud.google.com/functions/docs/tutorials/imagemagick>

An offensive image is included above - forewarning it is a zombie scene

Notes:

- 1) The create bucket code in the documentation is slightly out of date - creating by hand in the Console is less error-prone
- 2) Use python310 for the runtime (I was getting odd syntax errors with 3.12)..b
- 3) Enable Cloud Vision API

In-Class Work

Create a serverless function!

It should:

- 1) Use your favorite language
- 2) Have it be HTTP-triggered, unauthenticated, and return HTML
- 3) When triggered, it must randomly return either a:
 - a) Meme picture
 - b) Emoji
 - c) ASCII emoji from textfac.es

(i.e., `return random(["meme", "emoji", "ascii-emoji"])`)