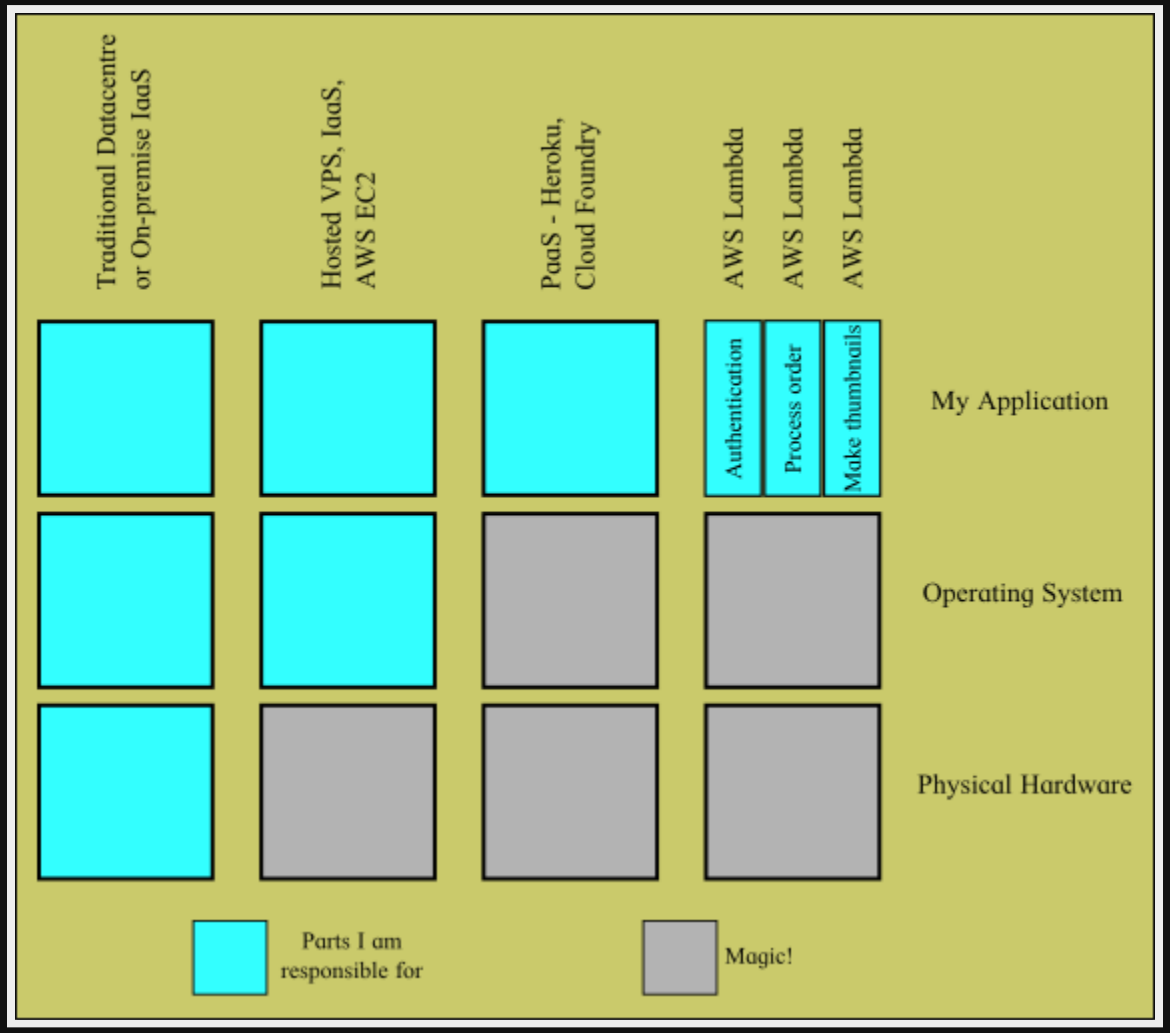
Lambda and Python

Using AWS Lambda to run Python scripts

James Mitchell

Deploying Your App

What needs to be deployed...



What is AWS Lambda?

" AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you... "

- deploy **functions** not applications
- the code is executed in **response to events** on other AWS services, such as S3, SNS, DynamoDB etc..
- or executed by the Amazon **API Gateway** (⇦ choose this one!)
- only supports Java and Node.js ☹

Adding some Python

We can make it work by getting a node.js script to execute the Python script!

- WillyG - Python on AWS Lambda
- Tim Wagner - Using Python in an AWS Lambda Function
- Eric Hammond provides a wrapper function

1 – Hello World in Python

- accept a JSON object from the wrapper, write JSON to stdout
- use virtualenv to bundle in python and extra libraries

```
import sys
import json

def main(event):
    name = event.get('name', 'Mr. Eastwood')
    response = dict(greeting='Hello', name=name)
    print(json.dumps(response))

if __name__=='__main__':
    argv = sys.argv[1:]
    event = json.loads(argv[0])
    main(event)
```

sample code on Github

2.1 - Lambda Wrapper Script

- This script runs our Python script
- Communication is via "stringified" JSON

```
var spawn = require('child_process').spawn;
exports.handler = function(event, context) {
  var response = {};
  var error = null;
  var child = spawn('venv/bin/python', [
    'lambda-function.py',
    JSON.stringify(event, null, 2)
    ]);
  child.stdout.on('data', function (data) {
    console.log("stdout:\n"+data);
    response = JSON.parse(data);
  });
  child.stderr.on('data', function (data) {
    console.log("stderr:\n"+data);
    error = { error: true, message: data.toString('utf8') };
  });
  child.on('close', function (code) {
    if (error !== null ) { context.fail(error); }
    else {context.succeed(response); }
  });
};
```

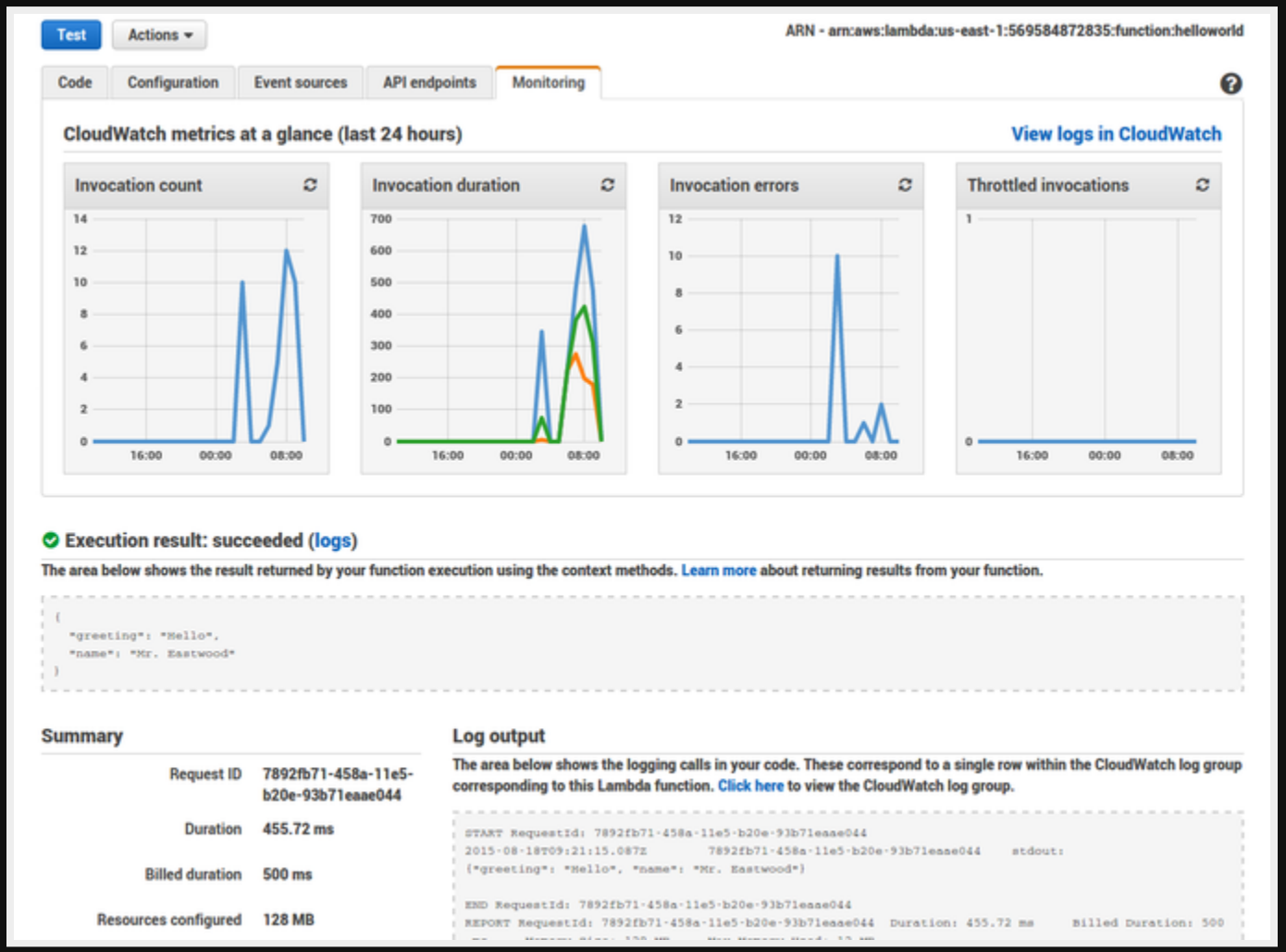
2.1 - Lambda Function

- upload a ZIP of the directory to S3 bucket
- ...this takes a while to upload 39Mb of Python
- make an IAM role for executing the function
- create the Lambda function

```
$ aws lambda create-function \
--region us-east-1 \
--function-name helloworld \
--code S3Bucket=maungawhau.lambda,S3Key=hello-lambda.zip \
--role arn:aws:iam::569584872835:role/lambda_basic_execution \
--handler lambda-function-wrapper.handler \
--runtime nodejs
```

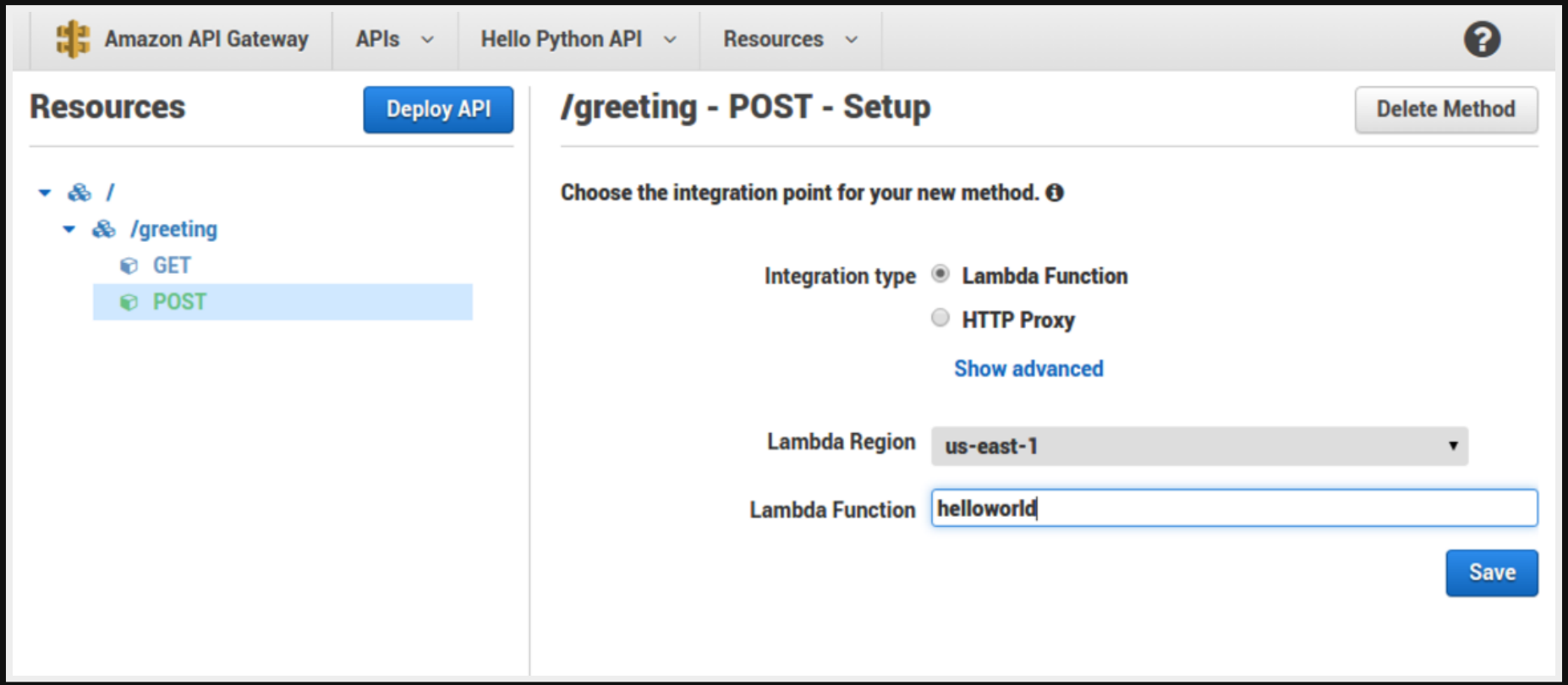
2.3 - AWS Lambda Console

Once it is running, use the console to check logs and number of calls.



5.1 – API Gateway

Configure a new API on AWS, with a resource that executes the Lambda function.

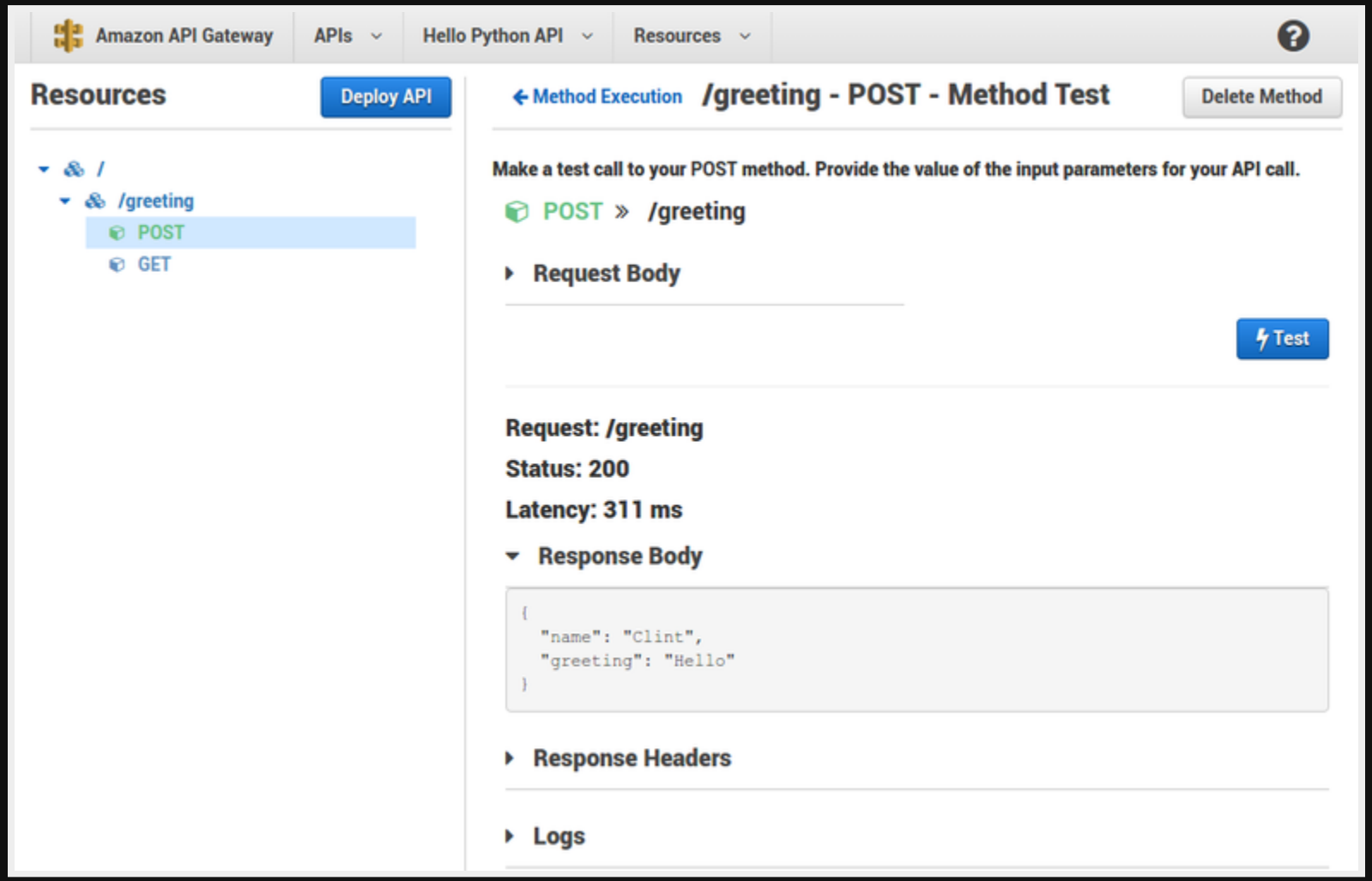


5.2 - Say “Hello” to my little friend

```
$ curl -X GET https://hostname/test/greeting
{"name":"Mr. Eastwood","greeting":"Hello"}

$ curl -X POST -d '{"name": "Clint"}' https://hostname/test/greeting
{"name":"Clint","greeting":"Hello"}
```

5.3 - Again... with Pictures



Conclusions

- it's **cheap**
  - FREE - first million requests per month
  - then $0.20 per 1 million requests
- minimal deployment hassle
  - no EC2 instance, no server maintenance
- great for a **micro-service**
  - simple endpoint
  - reacting to an event
- terrible for a Django app