SNAKES AND LAMBDAS

- what is lambda
- why is lambda (for data science)
- how is lambda (to actually use)
- when is lambda (the right choice)

WHAT IS LAMBDA?

- serverless
  - it runs on servers, you just don't deal with that
- scaleable
  - it only costs when running, you just pay more as it does more
- micro-service
  - it only does one small thing, you just have lots of different ones

WHAT IS LAMBDA?

1. Start a new lambda
2. Set up lambda with user code - "cold start"
3. Accept event 1 and process
4. Wait
    1. Accept event 2 and process - "warmed up"
    2. Timeout and kill lambda

There can be multiple *concurrent* machines too

WHAT IS LAMBDA?

- Pay only for what you use
- Manage only what you have to
- Deal with extra/new/bursty traffic seamlessly

WHAT IS LAMBDA?

- Invocation payload (request and response)
  - 6 MB (synchronous)
  - 256 KB (asynchronous)
- Deployment package size
  - 50 MB (zipped, for direct upload)
  - 250 MB (unzipped, including layers)
  - 3 MB (console editor)

WHY IS LAMBDA?

(FOR DATA SCIENCE)

- data-scientists != dev-ops professionals
  - but our work needs to be 'released'
- all data projects != ensemble xg-boost Keras TPU shenanigans
  - "No ML is easier to manage than no ML" © @julsimon
- data-projects != single-goal monolithic systems
  - separate concerns, code bases and complication

HOW IS LAMBDA?

(TO ACTUALLY USE)

1. write your python
2. lambda your python
3. ???
4. profit

1. WRITE YOUR PYTHON

```
from scipy import stats
np.random.seed(12345678)
x = np.random.random(10)
y = 1.6*x + np.random.random(10)
slope, intercept, r_value, p_value, std_err =
    stats.linregress(x, y)
```

2. LAMBDA YOUR PYTHON

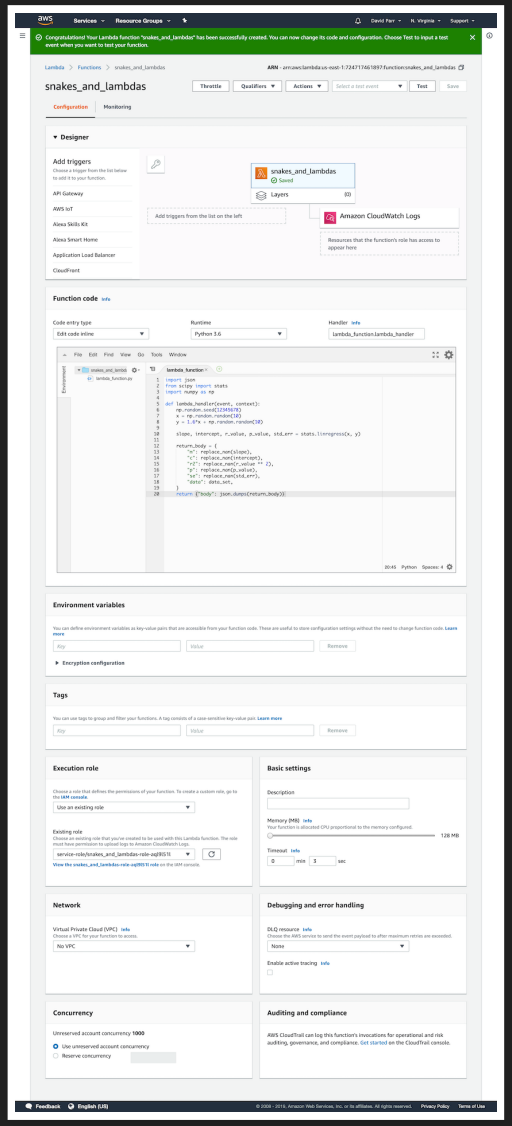
- *event driven*
  - an `event` is passed to a `handler function`
- *json formatted*
  - `events` are `json`
  - `handler functions` return `json`

2. LAMBDA YOUR PYTHON

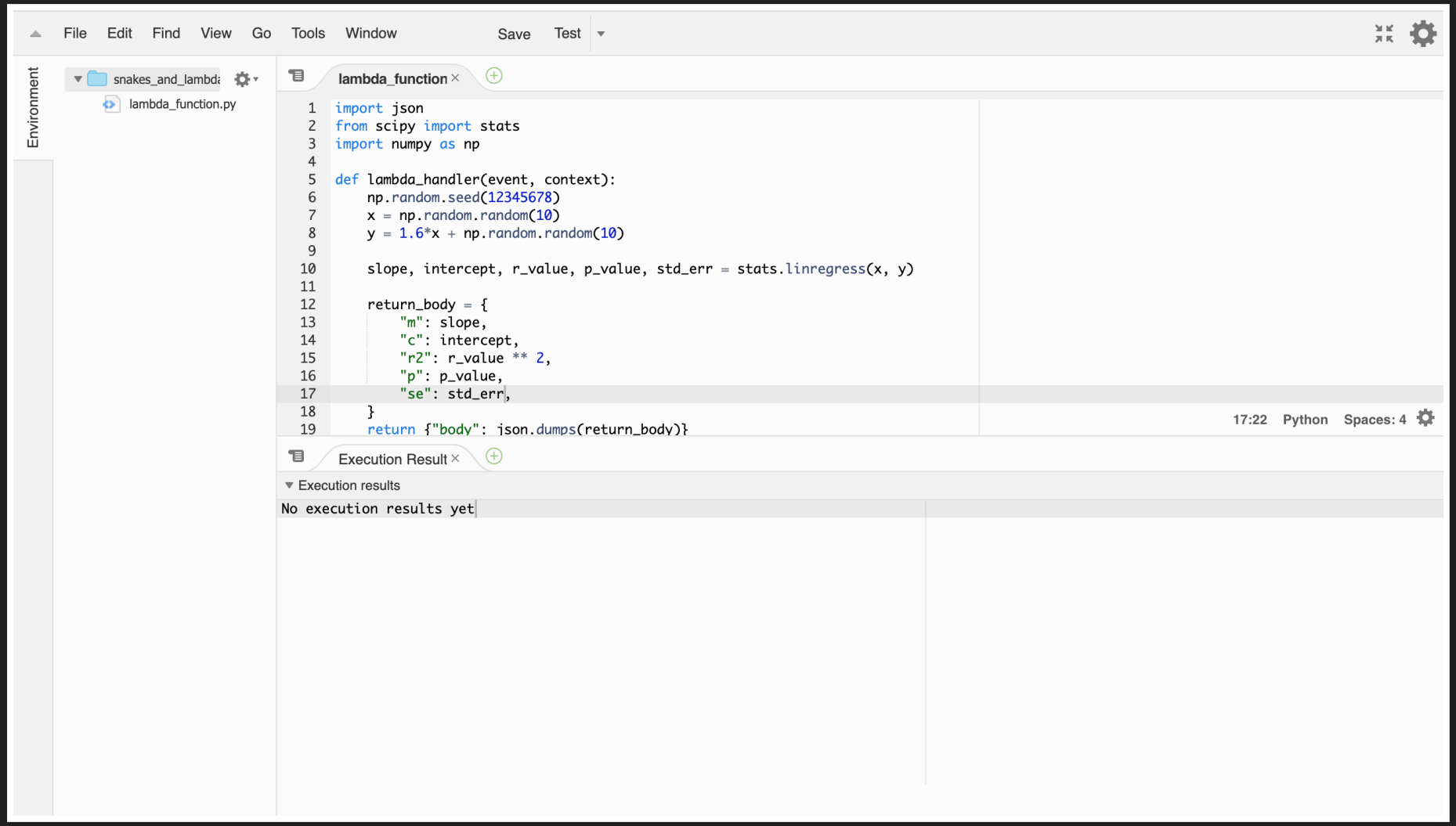
```
import json
from scipy import stats
import numpy as np

def lambda_handler(event, context):
    np.random.seed(12345678)
    x = np.random.random(10)
    y = 1.6*x + np.random.random(10)
    slope, intercept, r_value, p_value, std_err = stats.linreg
    return_body = {
        "m": slope, "c": intercept,"r2": r_value ** 2,
        "p": p_value, "se": std_err
    }
    return {"body": json.dumps(return_body)}
```

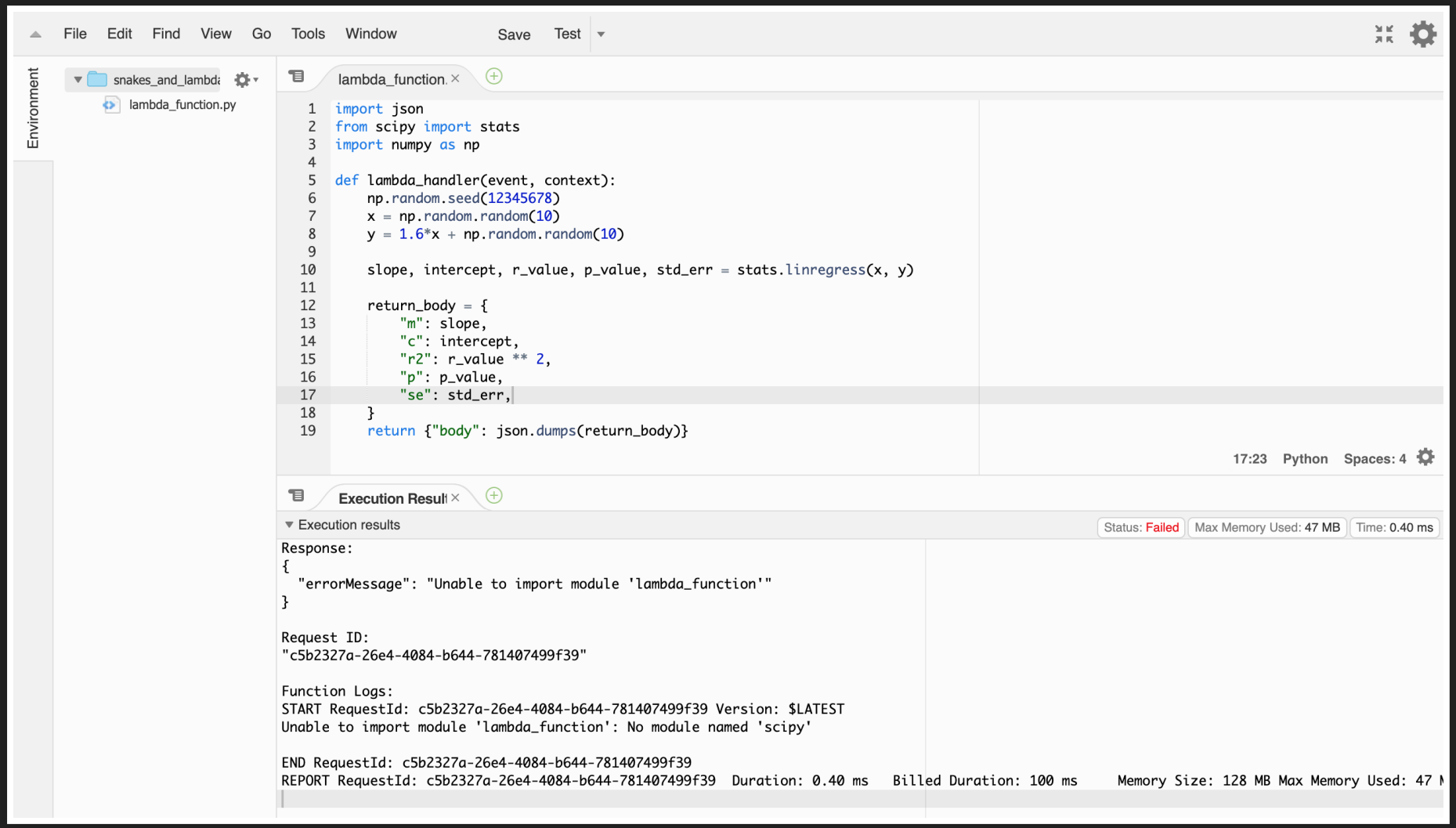
2. LAMBDA YOUR PYTHON



2. LAMBDA YOUR PYTHON



2. LAMBDA YOUR PYTHON



3. ???????

(1. LAYERS)

- `json` is *built in by default*
  - so it `boto3`

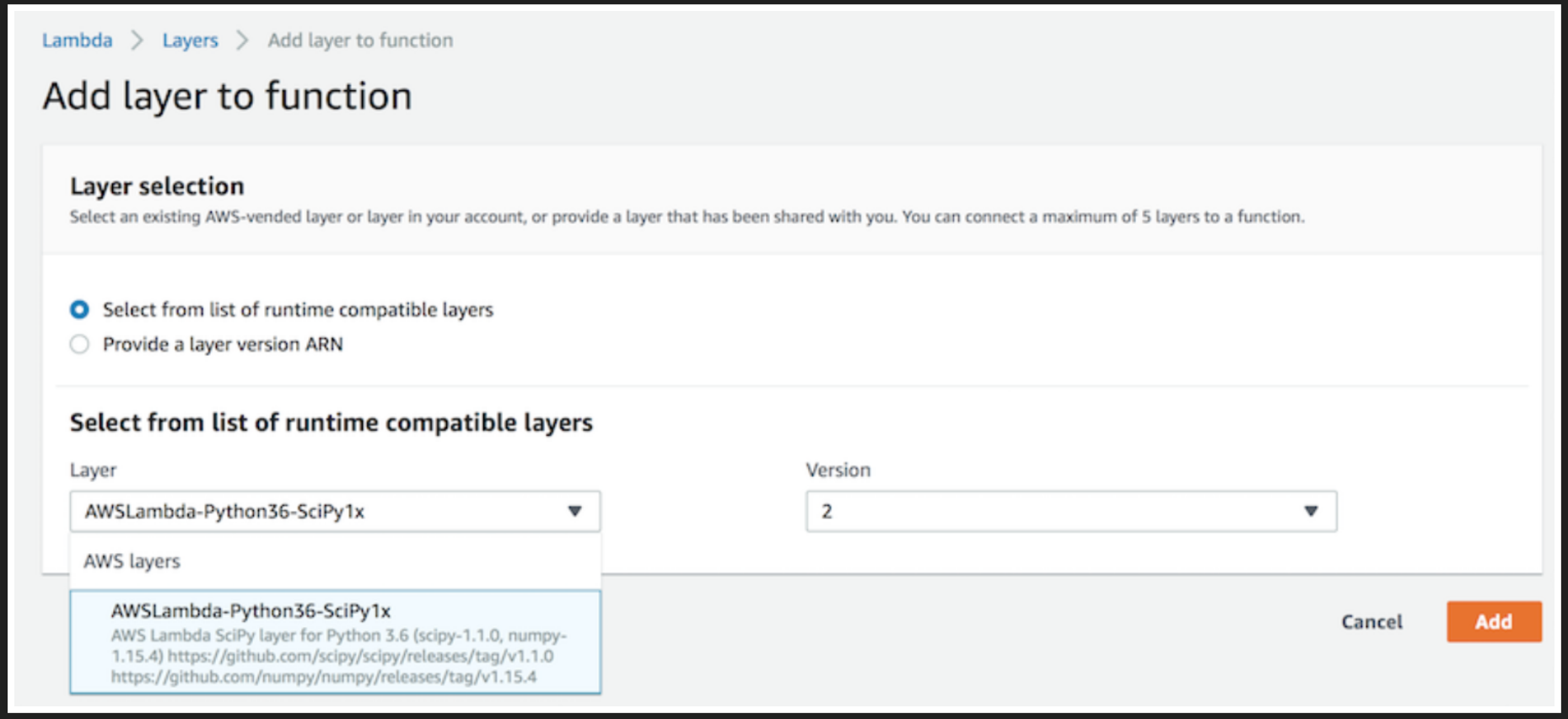
PROBLEM

- lambda doesn't `pip install ...`

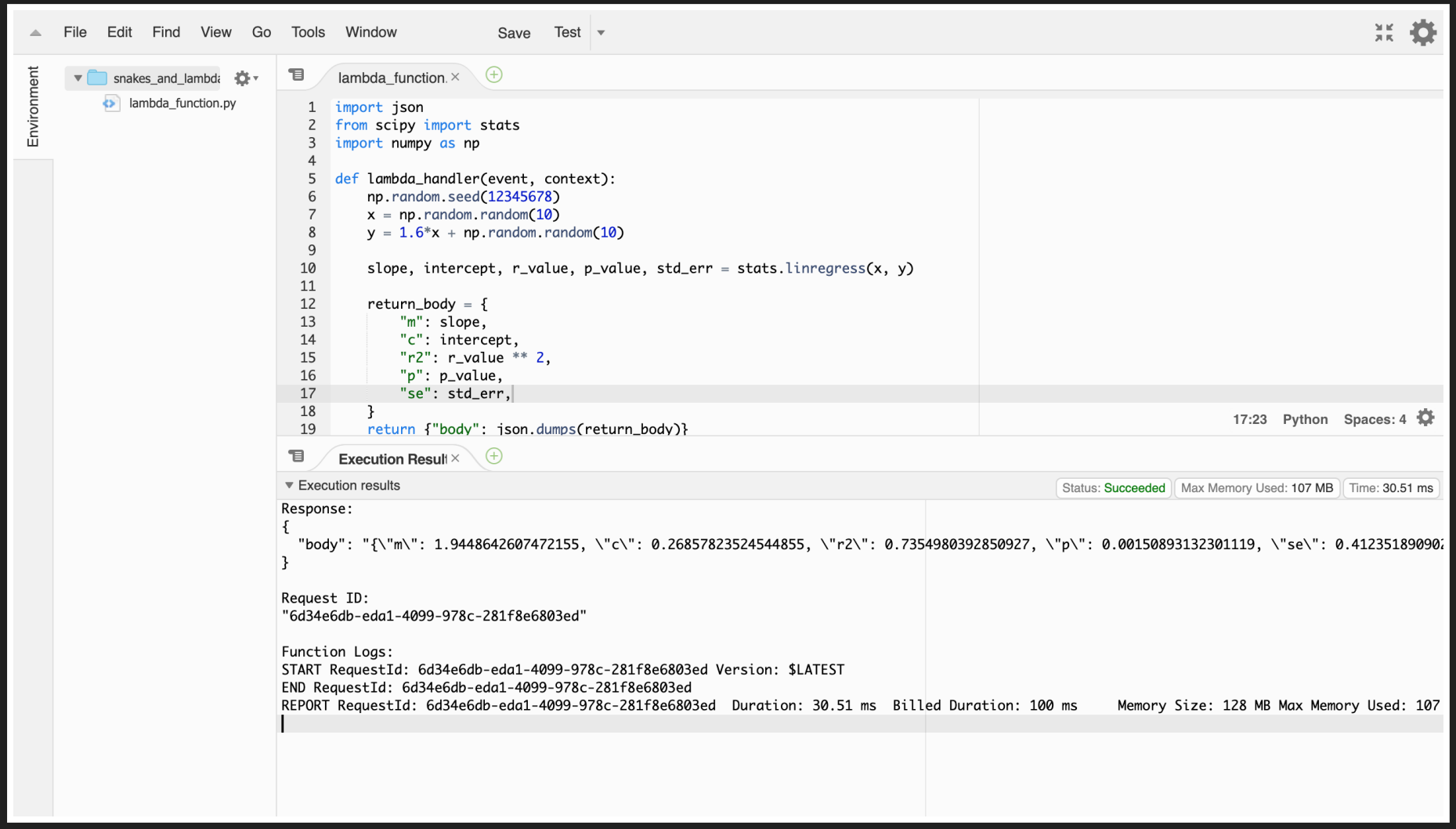
SOLUTION

- use `layers`
  - numpy, scipy are *published by aws*

3.1 LAYERS


Lambda > Layers > Add layer to function
Add layer to function
Layer selection
Select an existing AWS-vended layer or layer in your account, or provide a layer that has been shared with you. You can connect a maximum of 5 layers to a function.
Select from list of runtime compatible layers
Provide a layer version ARN
Select from list of runtime compatible layers
Layer
AWSLambda-Python36-SciPy1x
Version
2
AWS layers
AWSLambda-Python36-SciPy1x
AWS Lambda SciPy layer for Python 3.6 (scipy-1.1.0, numpy-1.15.4) https://github.com/scipy/scipy/releases/tag/v1.1.0 https://github.com/numpy/numpy/releases/tag/v1.15.4
Cancel
Add

3.1 LAYERS


File Edit Find View Go Tools Window Save Test
Environment
snakes_and_lambda
lambda_function.py
lambda_function
import json
from scipy import stats
import numpy as np
def lambda_handler(event, context):
np.random.seed(12345678)
x = np.random.random(10)
y = 1.6*x + np.random.random(10)
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
return_body = {
"m": slope,
"c": intercept,
"r2": r_value ** 2,
"p": p_value,
"se": std_err,
}
return {"body": json.dumps(return_body)}
17:23 Python Spaces: 4
Execution Result
Execution results
Status: Succeeded
Max Memory Used: 107 MB
Time: 30.51 ms
Response:
{
"body": "{\"m\": 1.9448642607472155, \"c\": 0.26857823524544855, \"r2\": 0.7354980392850927, \"p\": 0.00150893132301119, \"se\": 0.412351890902
}
Request ID:
"6d34e6db-eda1-4099-978c-281f8e6803ed"
Function Logs:
START RequestId: 6d34e6db-eda1-4099-978c-281f8e6803ed Version: $LATEST
END RequestId: 6d34e6db-eda1-4099-978c-281f8e6803ed
REPORT RequestId: 6d34e6db-eda1-4099-978c-281f8e6803ed Duration: 30.51 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 107

3. ???????

(2 CUSTOM LAYERS)

PROBLEM

- New requirement needs pandas

SOLUTION

- Create custom `layer`
  - pre-compiled code on a specific path deployed as a `.zip`
  - for 'any package' * using some shell and docker
    - * YMMV

3.2 CUSTOM LAYERS

requirements.txt

```
pandas==0.23.4
pytz==2018.7
```

get_layer_packages.sh

```
#!/bin/bash

export PKG_DIR="python"

rm -rf ${PKG_DIR} && mkdir -p ${PKG_DIR}

docker run --rm -v $(pwd):/foo -w /foo lambci/lambda:build-pyt
    pip install -r requirements.txt --no-deps -t ${PKG_DIR}
```

3.2 CUSTOM LAYERS

execute.sh

```
chmod +x get_layer_packages.sh
./get_layer_packages.sh
zip -r pandas.zip . -i "python/*"
```

Then upload + create as layer with `aws-cli` or manually with console

3.2 CUSTOM LAYERS

'any package' *

- `pandas`
- `pymysql`
  - `lambda` needs to be *inside* the same VPC
- `statsmodels`

3. ???????

(3 API GATEWAY)

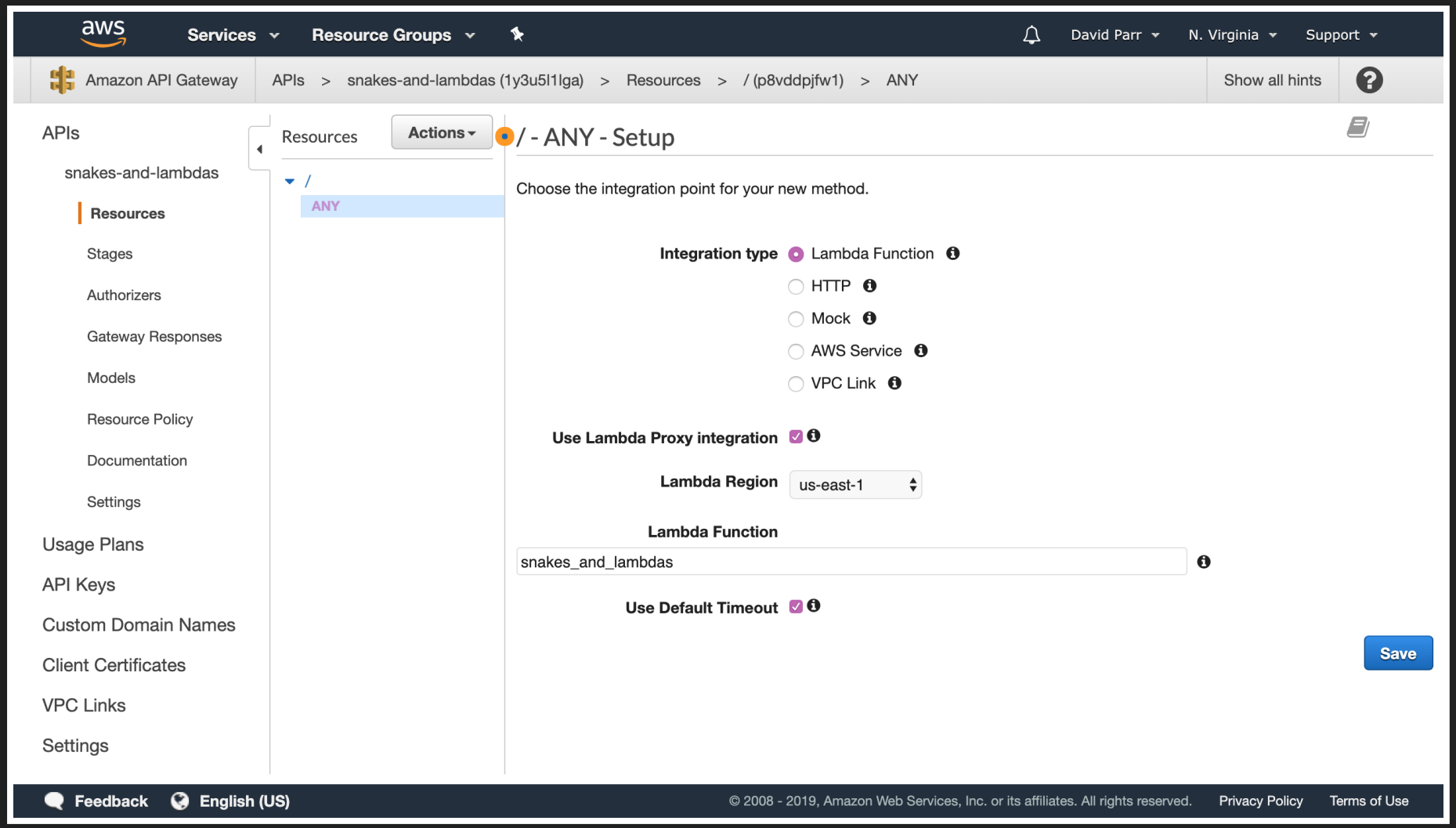
PROBLEM

- How does your team use your work?

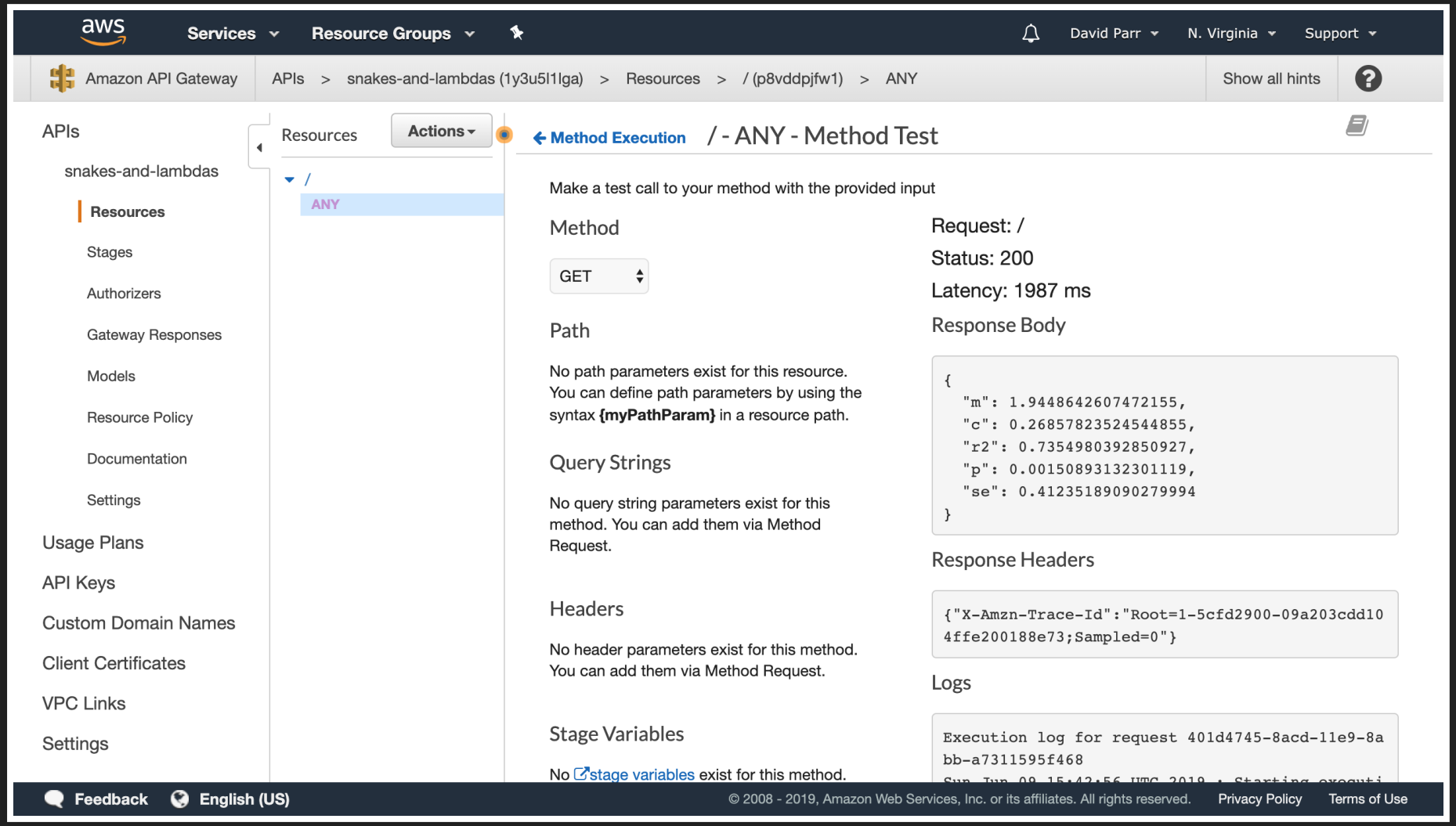
SOLUTION

- use `api gateway`
  - AWS service that puts REST api in front of the lambda

3.3 API GATEWAY



3.3 API GATEWAY



3.3 API GATEWAY

Get help from an adult (dev-ops professional)

3.3 API GATEWAY

If you can't find an adult

- be careful about exposing the api
  - not *obvious* how and where it can be accessed
    - resource policies
- `swagger` is an api templating syntax
  - cloud formation
- click the 'deploy api' button after *every* change
  - use multiple stages

3. ???????

(4 LOCAL DEV - CLOUD DEPLOY)

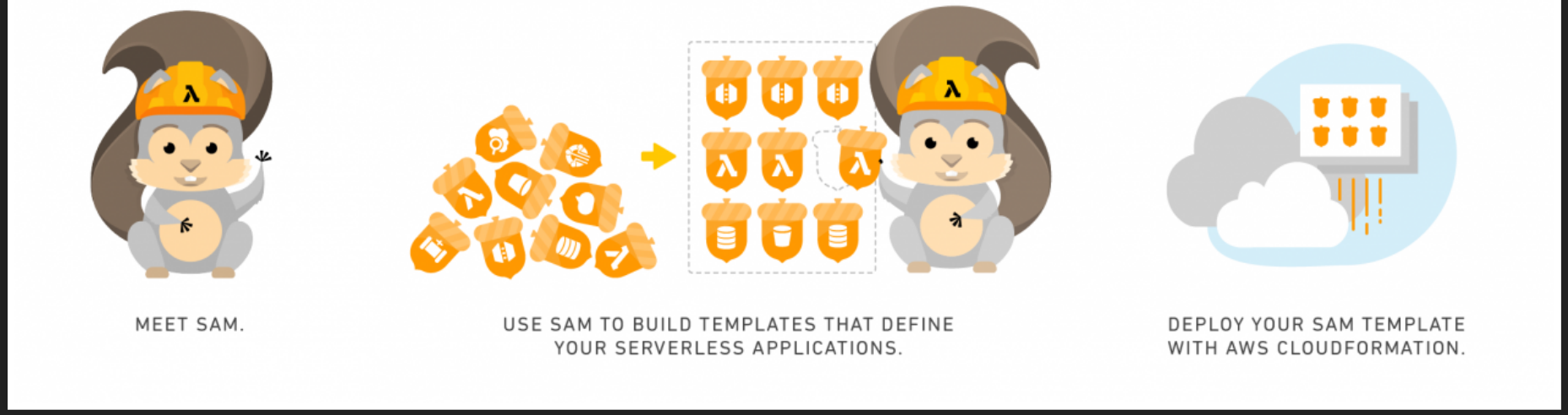
PROBLEM

- copy-pasta code into console is *bad*

SOLUTION

- use AWS SAM cli
  - local development + testing with docker
  - 'cloudy' deployment with cloudformation cli

3.4 LOCAL DEV - CLOUD DEPLOY



3.4 LOCAL DEV - CLOUD DEPLOY

```
Usage: sam [OPTIONS] COMMAND [ARGS]...

Commands:
  local     Run your Serverless application locally
            for quick development & testing.
  logs      Fetch logs for a function
  deploy    Deploy an AWS SAM application. This is an alias
            for 'aws cloudformation deploy'.
  build     Build your Lambda function code
  publish   Publish a packaged AWS SAM template to the AWS
            Serverless Application Repository.
  init      Initialize a serverless application.
  validate  Validate an AWS SAM template.
  package   Package an AWS SAM application. This is an alias
            for 'aws cloudformation package'.
```

3.4 LOCAL DEV - CLOUD DEPLOY

Workflow

- `sam init`
- `sam local generate-event apigateway aws-proxy`
  - `sam build`
    - ⇵
  - `sam local invoke -e event.json`

3.4 LOCAL DEV - CLOUD DEPLOY

```
alias playitsam='sam build && sam local invoke -e event.json'
alias playitagainsam='sam build && sam local invoke -e'
```

3.4 LOCAL DEV - CLOUD DEPLOY

- `sam validate`
- `sam package`
- `sam deploy`

3.4 LOCAL DEV - CLOUD DEPLOY

```
Transform: 'AWS::Serverless-2016-10-31'
Resources:
  RegressionFunction:
    # This resource creates a Lambda function.
    Type: 'AWS::Serverless::Function'
    Properties:
      # This function uses the python 3.7 runtime.
      Runtime: python3.7
      # This is the Lambda function's handler.
      Handler: app.lambda_handler
      # The location of the Lambda function code.
      CodeUri: ./regression
      # Event sources to attach to this function. In this case
      # one API Gateway endpoint to the Lambda function. The f
      # called when a HTTP request is made to the API Gateway
```

This enables CI/CD, which is a **Good Thing ™**

3.4 LOCAL DEV - CLOUD DEPLOY

Get help from an adult (dev-ops professional)

but if you can't, list 'em and flip 'em

```
aws lambda list-functions | cfn-flip
```

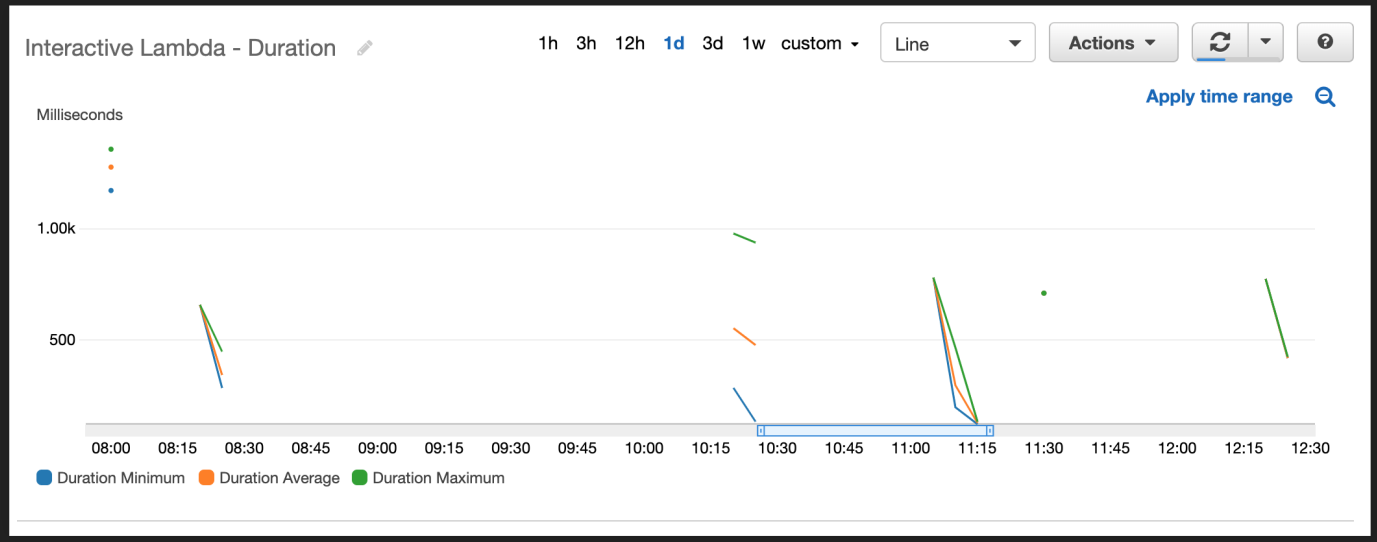
4. PROFIT

Surple have 3 lambda data services

LINEAR REGRESSION

'DEGREE DAYS VS ENERGY = EFFICIENCY'

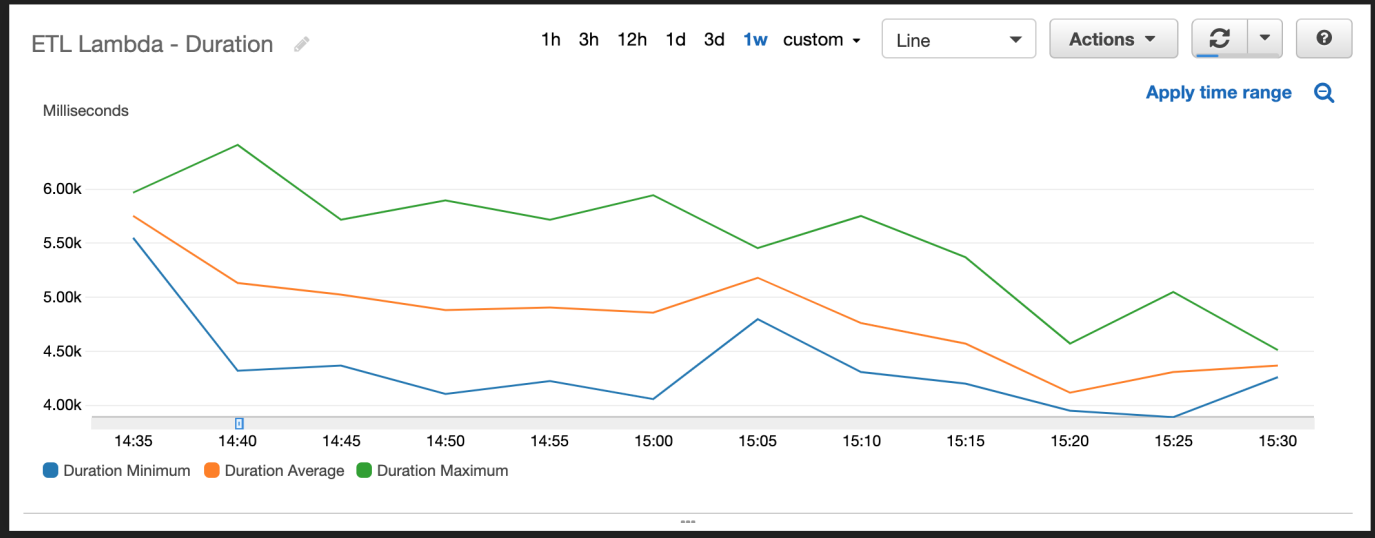
- user triggered event
- queries specific data based on user selection
- user facing visualisation
- vpc cold starts



TIME-SERIES ANALYSIS

'SMART TARGETS'

- scheduled for all meters as ETL to DB
- highlight 'out of character' energy use
- user facing visualisation and notifications



ANOMALY DETECTION

'SMART ALARMS'

- scheduled for all meters as ETL to DB
- highlight 'extreme' energy use
- user email and notifications
- this was extra fun/complicated
  - Ask me how

PRACTICAL NOTES

- tweaking cpu load has made more a difference than tweaking timeout
- taking the time to set up SAM correctly has saved at least the time of browser console work alone
- A Cloud Guru is built on lambda (cheaply?)
  - And has some great material on it
- Deployment from SageMaker is possible
- CI/CD from GitLab is possible

WHEN IS LAMBDA?

(THE RIGHT CHOICE)

Good case

- 'traditional' models
  - regression, timeseries, hopefully more...
- per 'reasonable' data set
  - `for each`
- 'now in a minute'
  - (not actually a minute, more like seconds)
- 'bursty'
  - some, or lots of people need it then no one does

WHEN IS LAMBDA?

(THE RIGHT CHOICE)

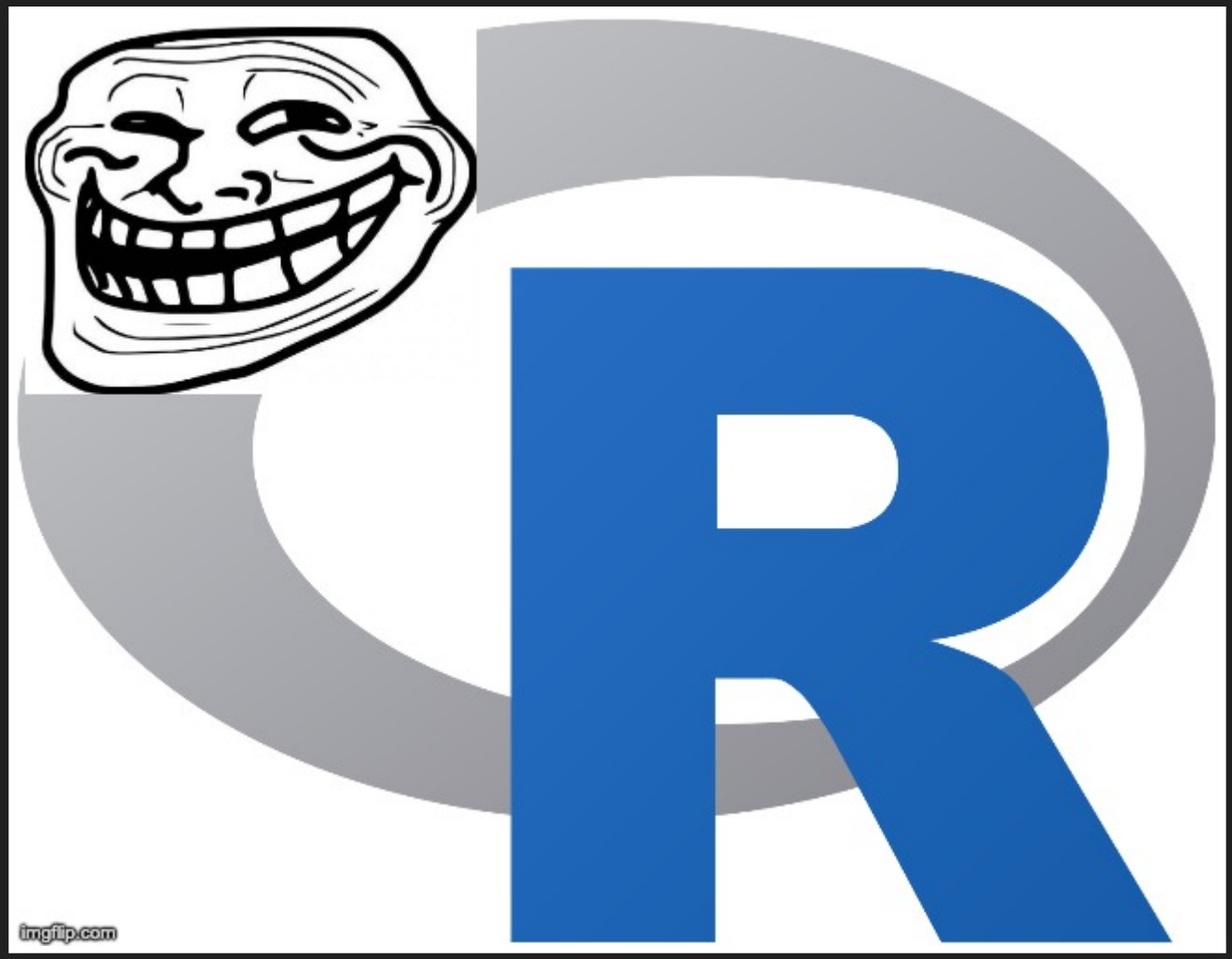
Bad case

- 'fancy' models
  - RAM limits, CPU limits
- whole scale
  - `across all`
- immediate response
  - can't afford a cold start: 'lambda your lambda'
- 24:7 flat load
  - need 100% load 100% of the time

THANK YOU

Slides available:
https://github.com/DaveParr/snakes_and_lambdas
Twitter: @DaveParr

'SMART ALARMS' ACTUALLY RUNS IN


R
imgflip.com

RUNTIME LAYERS

PROBLEM

- The thing I want to use isn't in Python
  - or Go, NodeJS, C#, Java

SOLUTION

- use `Runtime Layers`
  - any language compiled into a layer
  - accessed via `bash` or similar that processes event and passes to runtime
  - `bakdata/aws-lambda-r-runtime`