

SNAKES AND LAMBDA

- 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
- scalable
 - 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

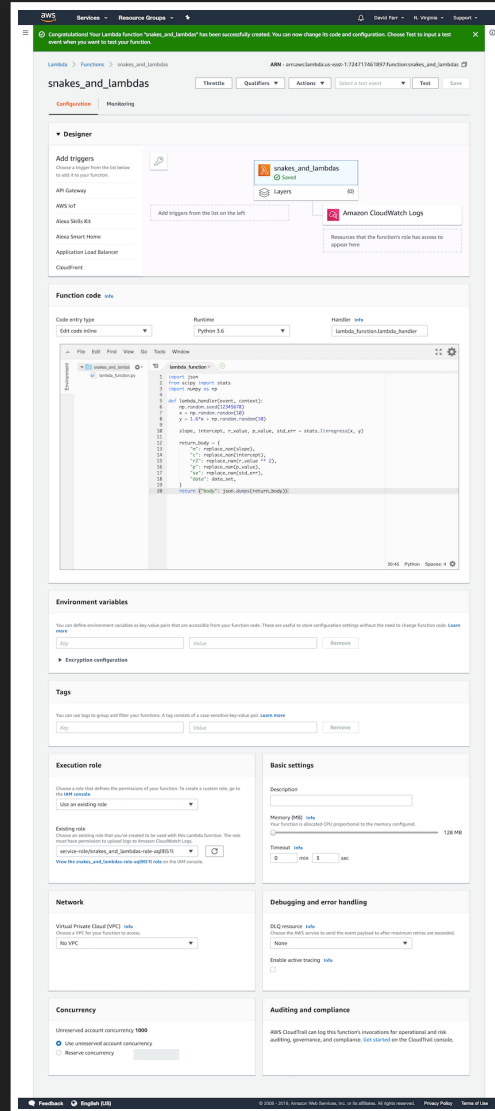
- *event driven*
 - an event is passed to a handler function
- *json formatted*
 - events are json
 - handler functions return json

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.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)}
```

2. LAMBDA YOUR PYTHON



2. LAMBDA YOUR PYTHON

The screenshot shows a code editor with a menu bar (File, Edit, Find, View, Go, Tools, Window, Save, Test) and a toolbar with window and settings icons. On the left, the 'Environment' sidebar shows a project named 'snakes_and_lambda' with a file 'lambda_function.py'. The main editor area has a tab for 'lambda_function.py' containing the following Python code:

```
1 import json
2 from scipy import stats
3 import numpy as np
4
5 def lambda_handler(event, context):
6     np.random.seed(12345678)
7     x = np.random.random(10)
8     y = 1.6*x + np.random.random(10)
9
10    slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
11
12    return_body = {
13        "m": slope,
14        "c": intercept,
15        "r2": r_value ** 2,
16        "p": p_value,
17        "se": std_err,
18    }
19    return {"body": json.dumps(return_body)}
```

At the bottom right of the editor, the status bar indicates '17:22 Python Spaces: 4' with a settings icon. Below the code editor is an 'Execution Result' tab, which is currently empty and shows the text 'No execution results yet'.

2. LAMBDA YOUR PYTHON

The screenshot displays a code editor interface with a menu bar (File, Edit, Find, View, Go, Tools, Window, Save, Test) and a toolbar. The left sidebar shows the 'Environment' panel with a folder 'snakes_and_lambda' and a file 'lambda_function.py'. The main editor window shows the code for 'lambda_function.py'.

```
1 import json
2 from scipy import stats
3 import numpy as np
4
5 def lambda_handler(event, context):
6     np.random.seed(12345678)
7     x = np.random.random(10)
8     y = 1.6*x + np.random.random(10)
9
10    slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
11
12    return_body = {
13        "m": slope,
14        "c": intercept,
15        "r2": r_value ** 2,
16        "p": p_value,
17        "se": std_err,
18    }
19    return {"body": json.dumps(return_body)}
```

Below the code editor is the 'Execution Results' panel. It shows the execution status as 'Failed' with a 'Max Memory Used: 47 MB' and 'Time: 0.40 ms'. The response is a JSON object indicating an error: 'Unable to import module 'lambda_function''. The request ID is 'c5b2327a-26e4-4084-b644-781407499f39'. The function logs show the start of the request and the error message: 'Unable to import module 'lambda_function': No module named 'scipy''. The end of the request and the report details are also shown.

17:23 Python Spaces: 4

Execution results

Status: **Failed** Max Memory Used: 47 MB Time: 0.40 ms

Response:

```
{
  "errorMessage": "Unable to import module 'lambda_function'"
}
```

Request ID:

"c5b2327a-26e4-4084-b644-781407499f39"

Function Logs:

START RequestId: c5b2327a-26e4-4084-b644-781407499f39 Version: \$LATEST
Unable to import module 'lambda_function': No module named 'scipy'

END RequestId: c5b2327a-26e4-4084-b644-781407499f39
REPORT RequestId: c5b2327a-26e4-4084-b644-781407499f39 Duration: 0.40 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 47 MB

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 ▼

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>

Version

2 ▼

Cancel

Add

3.1 LAYERS

The screenshot displays a web-based IDE interface. On the left, the 'Environment' sidebar shows a project named 'snakes_and_lambda' with a file 'lambda_function.py'. The main editor area shows the code for 'lambda_function.py' with line numbers 1 through 19. The code imports 'json', 'stats' from 'scipy', and 'numpy' as 'np'. It defines a 'lambda_handler' function that seeds a random number generator, generates random values for 'x' and 'y', performs a linear regression using 'stats.linregress', and returns a JSON object with the regression statistics. The IDE status bar at the bottom right indicates '17:23 Python Spaces: 4'.

```
1 import json
2 from scipy import stats
3 import numpy as np
4
5 def lambda_handler(event, context):
6     np.random.seed(12345678)
7     x = np.random.random(10)
8     y = 1.6*x + np.random.random(10)
9
10    slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
11
12    return_body = {
13        "m": slope,
14        "c": intercept,
15        "r2": r_value ** 2,
16        "p": p_value,
17        "se": std_err,
18    }
19    return {"body": json.dumps(return_body)}
```

Below the code editor, the 'Execution Results' tab is active, showing the following details:

- 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\": 0.41235189090}"
}
```
- 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

 **Services** ▾ **Resource Groups** ▾ 

David Parr ▾ N. Virginia ▾ Support ▾

 Amazon API Gateway APIs > snakes-and-lambdas (1y3u5l1ga) > Resources > / (p8vddpjfw1) > ANY Show all hints ?

APIs

snakes-and-lambdas

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Settings

Usage Plans

API Keys

Custom Domain Names



Client Certificates

VPC Links



Settings


Resources


Actions ▾


 / - ANY - Setup 


Choose the integration point for your new method.



Integration type  Lambda Function 


☐ HTTP 

☐ Mock 


☐ AWS Service 



☐ VPC Link 

Use Lambda Proxy integration  



Lambda Region us-east-1 

Lambda Function

snakes_and_lambdas 



Use Default Timeout  

Save


 Feedback  English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use


3.3 API GATEWAY

 **Services** ▾ **Resource Groups** ▾ 

David Parr ▾ N. Virginia ▾ Support ▾

 Amazon API Gateway

APIs > snakes-and-lambdas (1y3u5l1ga) > Resources > / (p8vddpjfw1) > ANY

Show all hints 

APIs

snakes-and-lambdas

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Settings

Usage Plans

API Keys

Custom Domain Names

Client Certificates

VPC Links

Settings

Resources

Actions ▾

Method Execution / - ANY - Method Test

ANY

Make a test call to your method with the provided input

Method

GET ▾

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

No query string parameters exist for this method. You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No [stage variables](#) exist for this method.

Request: /

Status: 200

Latency: 1987 ms

Response Body



```
{
  "m": 1.9448642607472155,
  "c": 0.26857823524544855,
  "r2": 0.7354980392850927,
  "p": 0.00150893132301119,
  "se": 0.41235189090279994
}
```

Response Headers

```
{ "X-Amzn-Trace-Id": "Root=1-5cfd2900-09a203cdd104ffe200188e73;Sampled=0" }
```

Logs

Execution log for request 401d4745-8acd-11e9-8a
bb-a7311595f468
Sun Jun 09 15:42:56 UTC 2019 - Starting executi

 **Feedback**  **English (US)**

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

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



MEET SAM.



USE SAM TO BUILD TEMPLATES THAT DEFINE
YOUR SERVERLESS APPLICATIONS.



DEPLOY YOUR SAM TEMPLATE
WITH AWS CLOUDFORMATION.

3.4 LOCAL DEV - CLOUD DEPLOY

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

Commands:

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

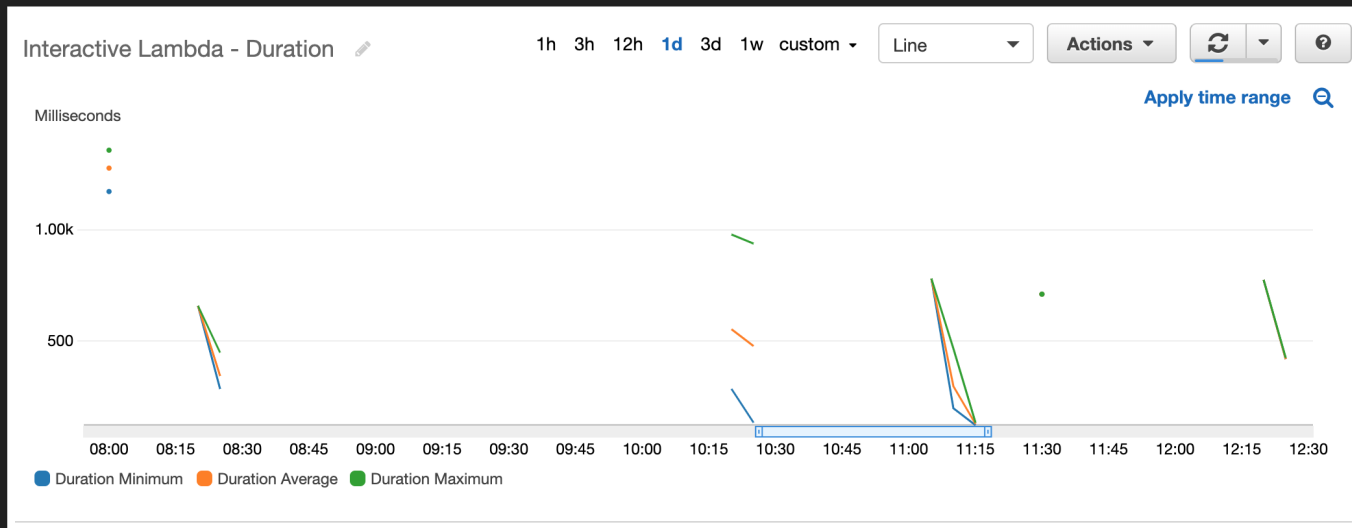


Surple have 3 lambda data services

LINEAR REGRESSION

'DEGREE DAYS VS ENERGY = EFFICIENCY'

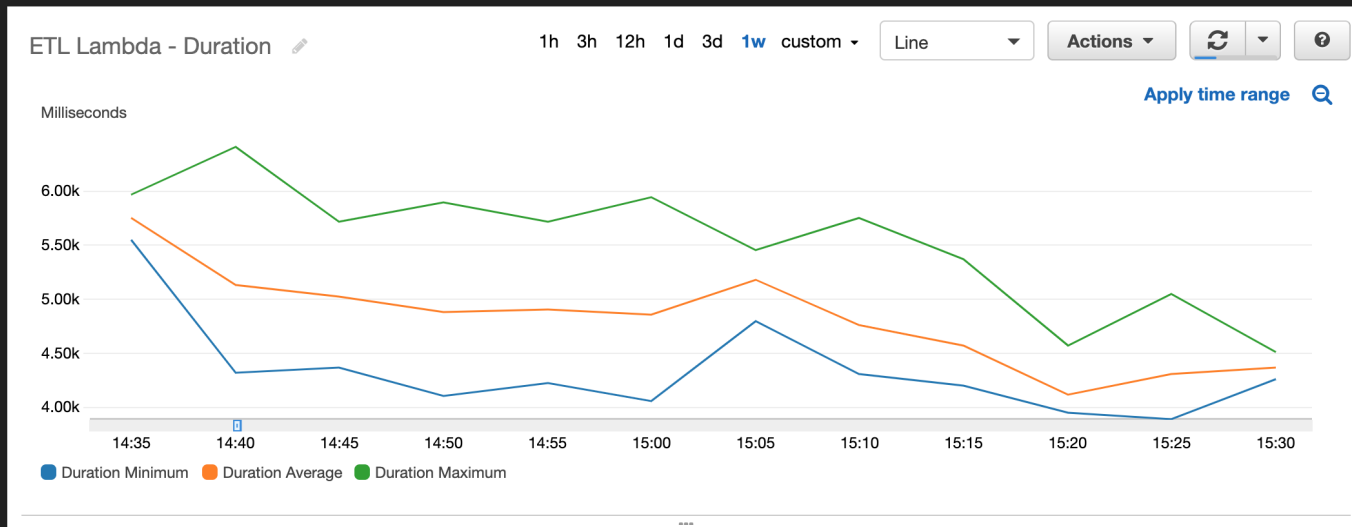
- user triggered event
- queries specific data based on user selection
- user facing visualisation
- vpc cold starts



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

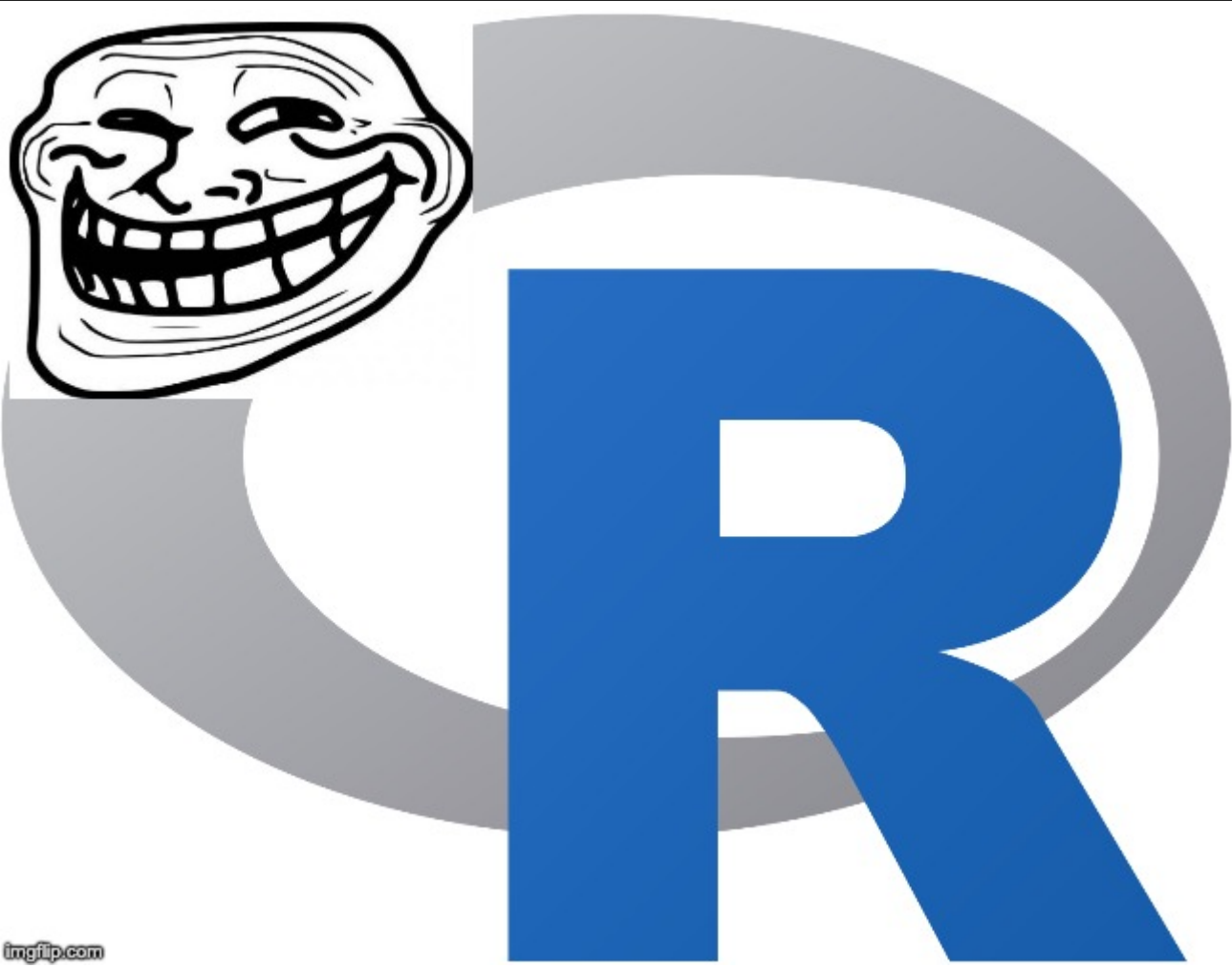
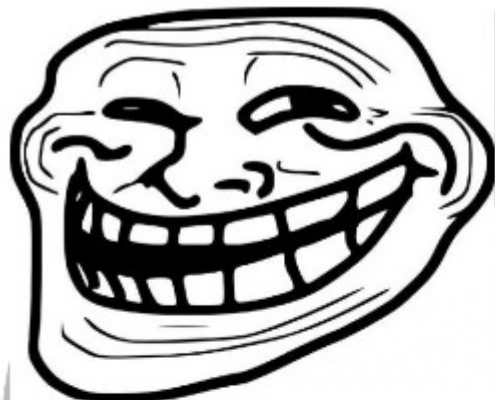
THANK YOU

Slides available:

https://github.com/DaveParr/snakes_and_lambdas

Twitter: [@DaveParr](#)

**'SMART ALARMS' ACTUALLY
RUNS IN**



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](#)