

Report Title

EC2 Cost Optimisation demo

Michael Ugbechie | Platform Engineering | 19/03/2020

# Introduction

To replace the placeholder text on this page, you can just select it all and then start typing. But don’t do that just yet!

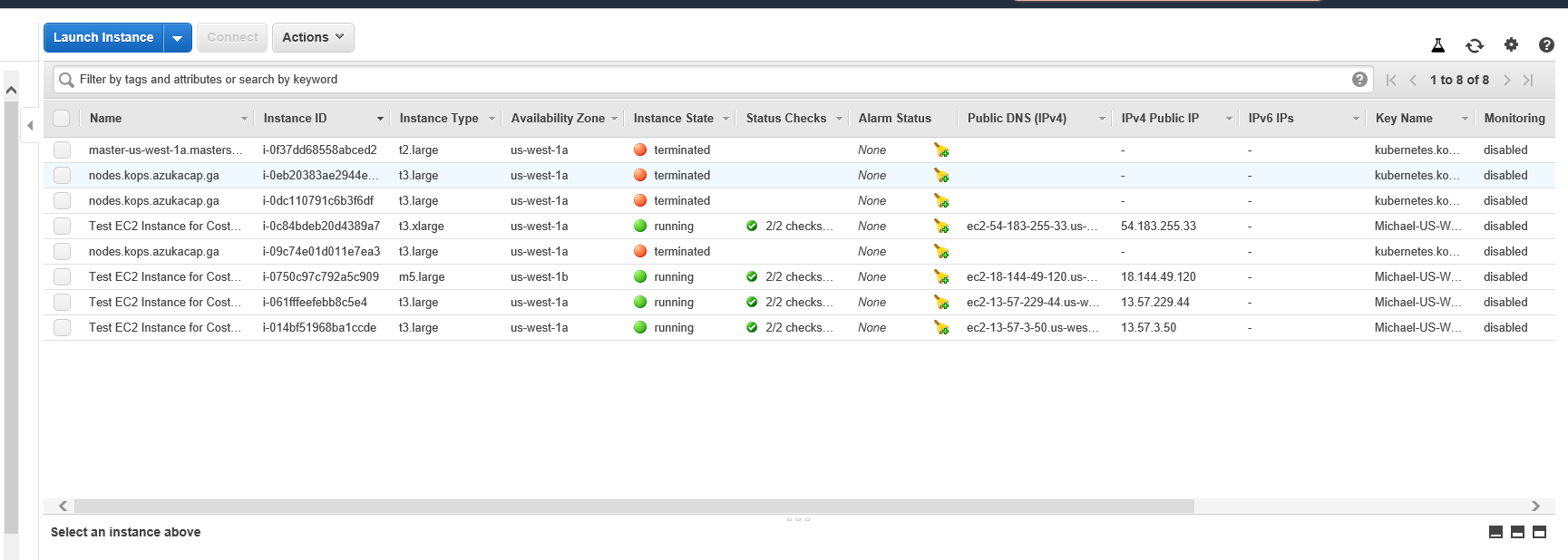
First check out a few tips to help you quickly format your report. You might be amazed at how easy it is.

* Need a heading? On the Home tab, in the Styles gallery, just click the heading style you want.
* Notice other styles in that gallery as well, such as for a quote, a numbered list, or a bulleted list like this one.
* For best results when selecting text to copy or edit, don’t include space to the left or right of the characters in your selection.

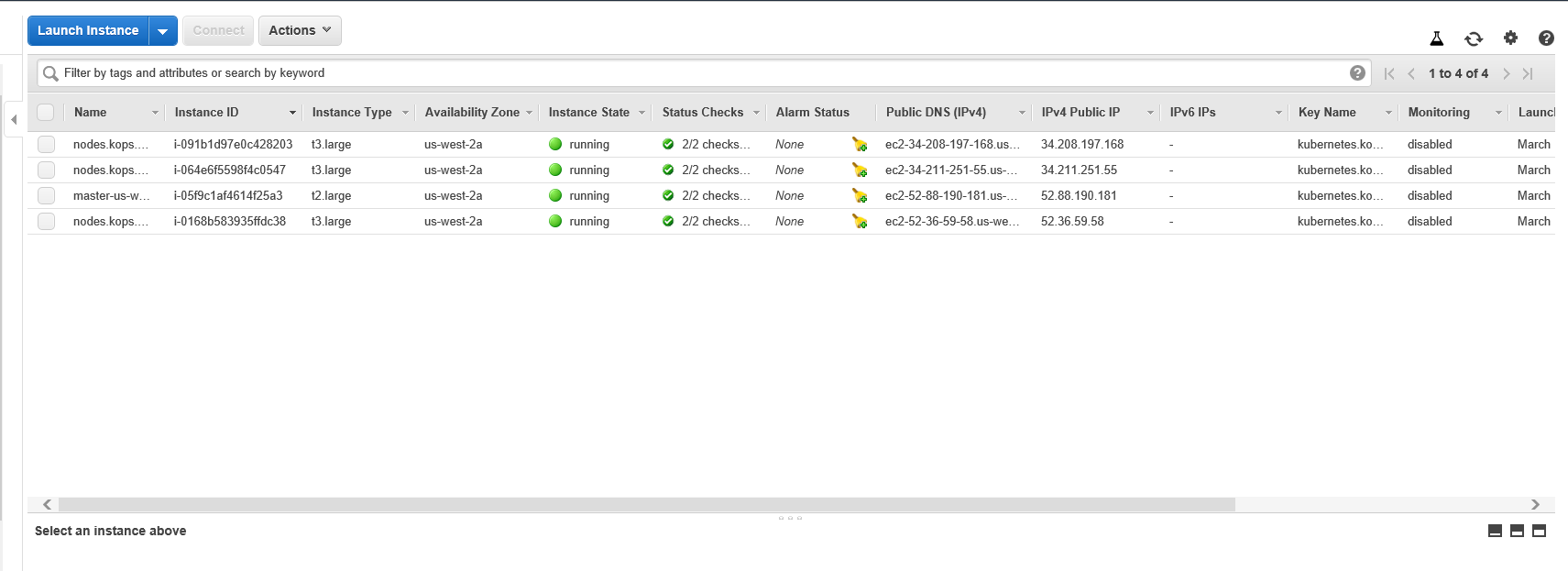
## DEMO

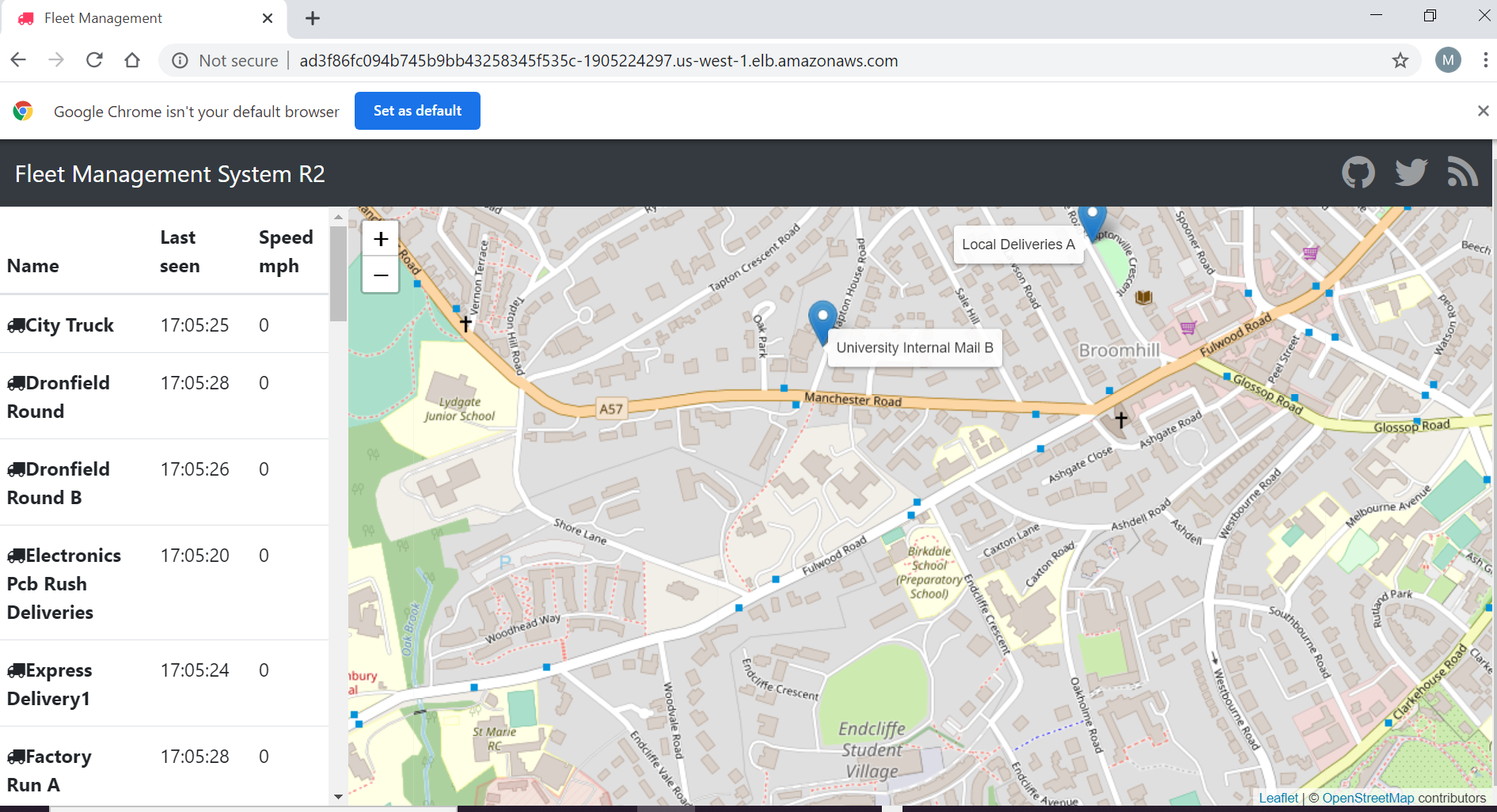
Test EC2 Instances

* Initial instances in California us-west-1
* Instances I set up which have no load



Kubernetes Instances with three nodes and a master running a microservice being monitored by Prometheus/Grafana





After the application run the following results achieved in interim dataset

**California (USW1)**

* t3.xlarge 89a7 detected (results) suggested r5a.large
* m5.large c909 detected (results) suggested t3a.large
* t3.large c5e4 detected (results) suggested t3a.large
* t3.large ccde detected (results) suggested t3a.large

**Oregon (USW2) Kubernetes Microservice**

* t3.large 8203 detected (results) suggested t3a.large
* t3.large 0547 detected (results) suggested t3a.large
* t3.large dc38 ruled out because top cpu > 50%
* t2.large 25a3 detected (results) suggested t3a.large

**Virginia (USE1)**

* t2.large cf4b detected (results) suggested t3a.large
* m5.large 2769 ruled out because top cpu > 50 %
* t3a.small bd43 detected (results) ruled out, no improvement
* m5.large 026 ruled out because top cpu > 50 %
* m5.large 27b ruled out because top cpu > 50%
* c5.large 5fd detected (results) suggested t3a.medium

4 instances standing alone and 4 being stressed out by a microservice simulating delivery trucks delivering goods to supermarkets.

**Note**

* those with max cpu reading greater than 50 will not be considered for reduction
* Those where the chosen instance is same as original instance will be ignored so the (correction filtered out) i.e t3a.small

**The full right sizing logic is shown as follows**

Logic used

If cpu\_max [ln\_cpu\_usage ]< 50 %

     if ln\_min\_cpu>=ln\_cpu\_nbr:

                 if ln\_min\_mem>=ln\_mem\_size:

                     if ln\_min\_network\_level>=ln\_network\_level\_usage:

                         if ln\_min\_rate<=ln\_rate:

***where***

1. ln\_min\_cpu =    (new instance)      found by aplication
2. ln\_cpu\_nbr = 1 (calculated from old instance using \*)
3. ln\_min\_mem =   (new instance)    found by application
4. ln\_mem\_size = 8 (from old instance )
5. ln\_min\_network\_level = (new instance)
6. ln\_network\_level\_usage = Low to Moderate (from old instance )
7. ln\_rate  = 0.0992000000 (from old instance )
8. ln\_min\_rate = (new instance)

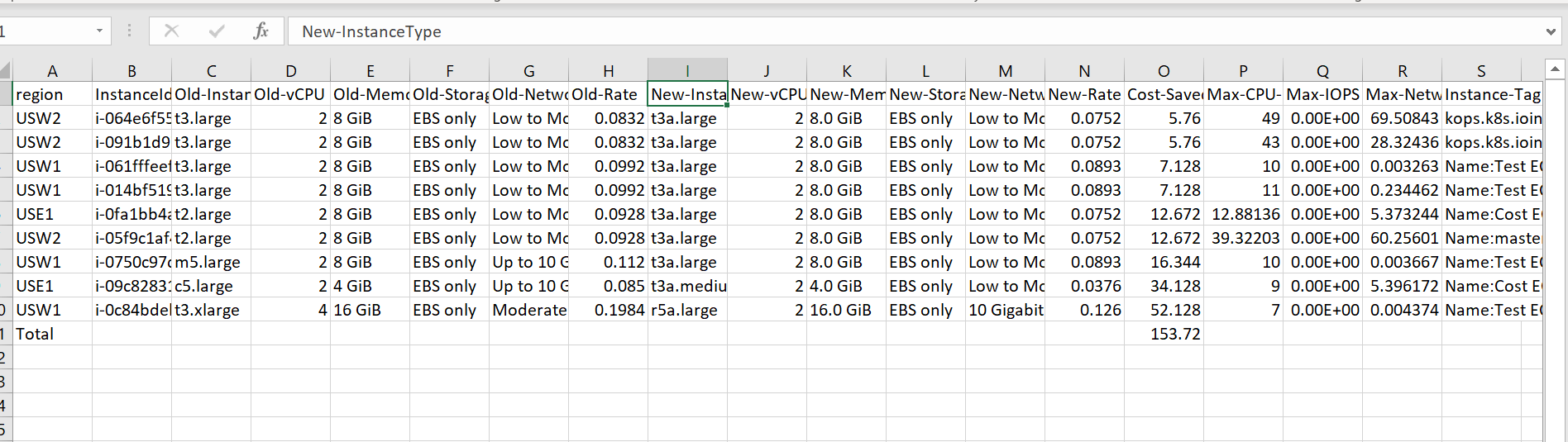
The application carries out this calculation with max cpu usage

\*“ln\_cpu\_nbr = math.ceil(float(ln\_cpu\_usage/100) \* ln\_cpu).

Where

* ln\_cpu\_usgae = max cpu
* ln\_cpu = no of cores”

The resultant suggestion csv is below



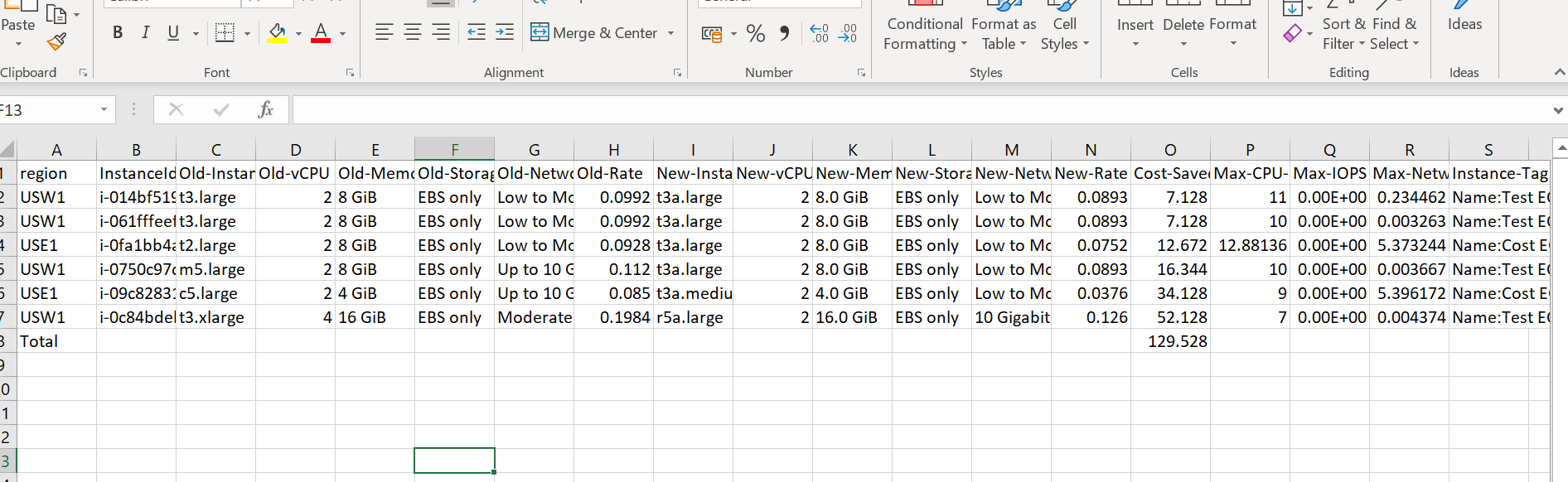
***Rerun***

***Make the improvements to Kubernetes (USW2) following suggestion***

If we run the monitoring tool after an hour again

Results

**USW2 cases now disappear off the list**



Running the application (see README)

Github

<https://gitlab.platform-engineering.com/michael.ugbechie/ec2_cost_optimization>

Code structure

**ec2\_cost\_optimization**

main.tf

.gitignore

backend.tf … empty

provider.tf

README

**module**

cost-opti-cf

input.tf

main.tf

cost-opti-lambda

input.tf

main.tf

output.tf …empty

**lambda**

lambda.py ….. can be edited [email] (checked into github)

lambda.zip …..automatically created

Running the Application

Profile setup

Profile

There are ways of setting up your profile. The above file assumes that a dev profile has been set up f0r the user. There are ways of doing this both manually and script wise of doing this. Below I will explain the manual way.

**Manually**

Install aws cli

Step 1 - set up your config and credentials files

Go to .aws/config

[profile default]

region = eu-west-2

output = json

aws\_access\_key\_id = <id for default user>

aws\_access\_key\_key = <key for default user>

[profile dev]

region = eu-west-2

source\_profile = default

role\_arn = arn:aws:iam:<number assigned to role>:role/OrganizationEngineerAccesRole

mfa\_serial = arn:aws:iam:<number assigned to user>:mfa/michaelugbechie

Go to .aws/credentials

[default]

aws\_access\_key\_id = <id for default user>

aws\_access\_key\_key = <key for default user>

**step 2**

export AWS\_PROFILE=dev

export AWS\_DEFAULT\_PROFILE=dev

export AWS\_SDK\_LOAD\_CONFIG=1

run

aws s3 ls

enter mfa number

You will get a list of s3 buckets

**Step 3**

Go to .aws/config

[profile default]

region = eu-west-2

output = json

aws\_access\_key\_id = <id for default user>

aws\_access\_key\_key = <key for default user>

[profile dev]

region = eu-west-2

source\_profile = default

role\_arn = arn:aws:iam:<number assigned to role>:role/OrganizationEngineerAccesRole

#mfa\_serial = arn:aws:iam:<number assigned to user>:mfa/michaelugbechie

**Step 4**

RUN (after putting in token0

aws sts get-session-token --serial-number arn:aws:iam::843361875856:mfa/michaelugbechie --token-code xxxxx --duration-seconds 43200 --profile default

result

json with

* + - temp id
    - temp key
    - temp token

***Go to .aws/credentials***

#[default]

# aws\_access\_key\_id = <id for default user>

# aws\_access\_key\_key = <key for default user>

[default]

aws\_access\_key\_id = <temp id for use profiler>

aws\_access\_key\_key = <temp key for user profile>

aws\_session\_token = <session token given>

**Step 5**

Test by running aws s3 ls…..should work

All set to run terraform

**Automated way**

Step 1 setup config and credentials

===================================================

Go to .aws/config

[profile default]

region = eu-west-2

[profile dev]

region = eu-west-2

source\_profile = mfa

role\_arn = arn:aws:iam:<number assigned to role>:role/OrganizationEngineerAccesRole

#mfa\_serial = arn:aws:iam:<number assigned to user>:mfa/michaelugbechie

=======================================

Go to .aws/credentials

[default]

aws\_access\_key\_id = <id for default user>

aws\_access\_key\_key = <key for default user>

step 2

export AWS\_PROFILE=dev

export AWS\_DEFAULT\_PROFILE=dev

export AWS\_SDK\_LOAD\_CONFIG=1

step 3

run the script (which will be stored in github soon)

script = start\_aws\_profile.sh

run

start\_aws\_profile.sh mfa michaelugbechie <mfa number>

Done

Setting up application via Terraform

Before we run

***Provider.tf need to be updated***

provider "aws" {

region = "us-east-1"

profile = "dev"

}

***Main file needs to be updated***

resource "random\_id" "suffix" {

byte\_length = 4

}

module "cost\_opti\_lambda" {

source = "./module/cost-opti-lambda"

suffix = random\_id.suffix.hex

s3\_bucket\_store = "capgemini-ec2pricing-auto"

handler = "lambda.lambda\_handler"

kmsEncryptedHookUrl = "https://hooks.slack.com/services/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"

slackChannel = "test"

}

module "cost\_opti\_cf" {

source = "./module/cost-opti-cf"

suffix = random\_id.suffix.hex

# BucketName = "capgemini-ec2-pricing-london"

BucketName = "capgemini-ec2pricing-auto"

KeyName = "Michael-N-Virginia"

}

Running the Application

You are now ready to run the application

You have instances to monitor

<https://gitlab.platform-engineering.com/michael.ugbechie/ec2_cost_optimization>

1. cd **ec2\_cost\_optimization**
2. **make necessary changes to provider and main**
3. Terraform destroy : makes sure you get rid of any other lingering CloudForamtion infrastructure…

Terraform destroy

Terraform apply

That’s it wait for email /notification