School of
Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

Project #2: Lambda

This assignment is due end of the trimester +/- up to five late days.

It is a variation on this tutorial, you are welcome to use the Python code found there. My main concern is that you know what it does and your analysis of what is going on.

Overview

The learning goals for this project are:

- Be able to make use of alternative programming models for large scale distributed systems,
- Be able to use basic mechanisms for protection and system security.

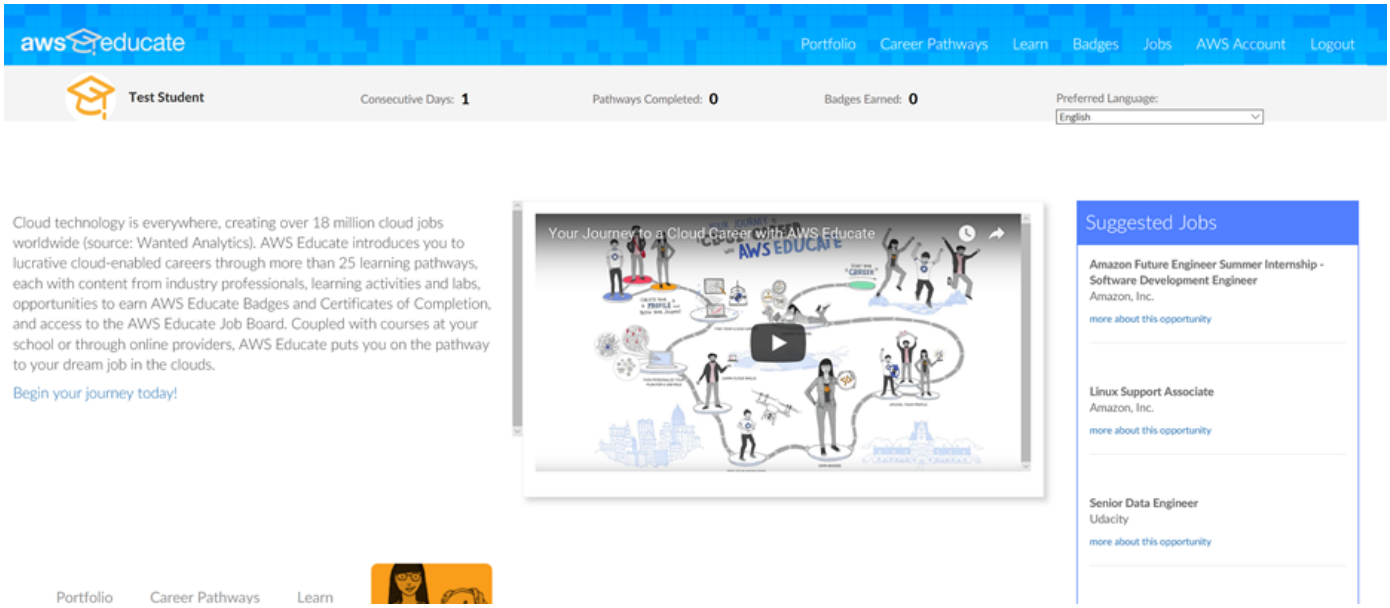
You will be using the Amazon API Gateway and Amazon Lambda service to explore the costs and performance of a distributed system for computing prime numbers.

Getting Started

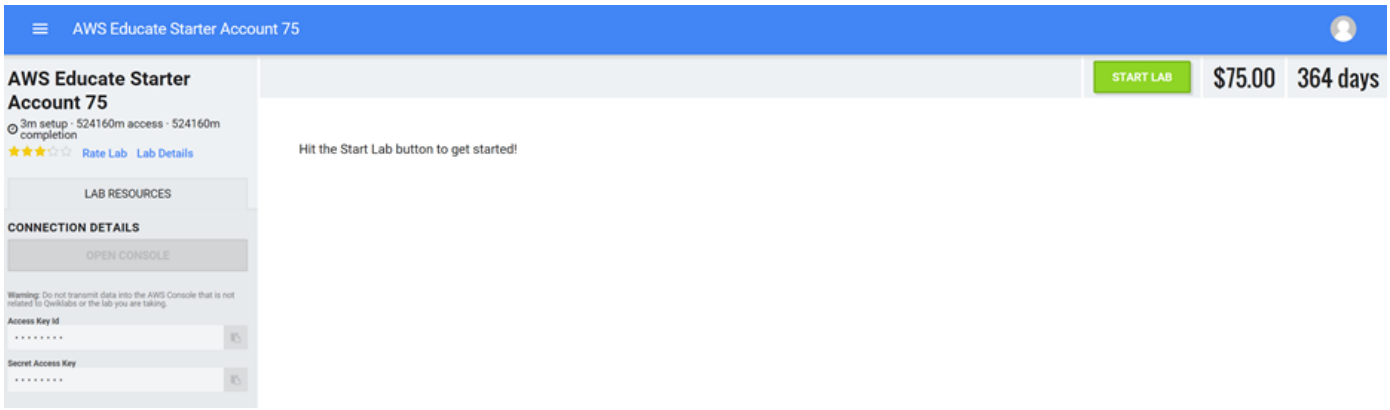
Sign up for AWS educate student tier following the instructions sent in the email. Note that it can take up to a couple of days for your application to be verified so start early! Contact Jonathon Flutey if you have any problems.

Once you have signed up, login by browsing to https://www.awseducate.com/student/s/ and enter your username and password

In the Educate student portal, click on 'AWS Account' in the top menu -



In the Lab page, click on 'Start Lab'



Once this has started, click on 'Open Console' on the left hand side to open the AWS console.

On a side note, dont click 'Stop Lab' once started, this seems to stop access to the lab altogether which I am chasing AWS up on.

The free AWS educate tier has most of the features of the free version that requires a credit card, however it doesn't support identity and access management (IAM) so the steps to using Lambda differ a little from the tutorial found here ... getting started guide.

Start with step 2 on creating a Lambda function. It assumes use of a IAM role and doesn't link to the API gateway, you have to follow a slightly different set of steps as below:

- Go to *Lambda Management Console*
- *Create function*
- Choose hello-world-python3.6 blueprint
- Click on the dotted box (trigger), choose API gateway
- Choose Open for the security setting
- Click on Next
- Give your function a name (under Name*)
- For Role, choose custom role (a new window will open)
- Create New Role
- Give it a name (for example, lambda_basic_execution)
- Click on Allow (returns you to the previous form)
- Choose an existing role, choose the role created above (for example, lambda_basic_execution)
- For Existing role, choose lambda-execution-role
- Create function
- Click on save and test to test it as in the tutorial.
- To execute the function via HTTP you need the URI, look for API Gateway with method below, click on the arrow pointing to method, this will display the Invoke URL
- You can try invoking directly via the URL but the sample python code isn't compatible, try using the code from the tutorial instead which does have the correct response and also a method to access parameters

You can use CURL to invoke from the command line, you need to pass parameters using ? for example:

Satisfactory "C-/C"

Now use the code from this tutorial and implement four Lambda functions for eratosthenes with (128MB, 256MB, 512MB and 1024MB).

Document the results of running each of these functions (you can do this from the console). You only need to run each once to show that it works.

Do not use the test harness.

Deploy your four functions and manually test that each of them works, include evidence of this in your report.

Did you find that cost and performance scaled linearly with memory? Why or why not?

Note that when configuring access to your account that you should use the **`-profile`** option to specify your user (for example, adminuser) and again when issuing other commands.

Completion "C+/B"

Automate invocation (via HTTP) by writing a client in your preferred language.

You might need to configure environment variables depending on how you are going to invoke the function.

```
$ setenv HTTP_PROXY=http://username:pass@www-cache.ecs.vuw.ac.nz:8080
$ setenv HTTPS_PROXY=http://username:pass@www-cache.ecs.vuw.ac.nz:8080
```

Execute each function 100 times.

Write a short overview of how to run your client and how it works.

Investigate the effect on performance and cost of the following:

- Varying the Lambda memory settings: 128MB, 256MB, 512MB and 1024MB.
- Varying the time taken to do a computation while holding memory static: 2x, 3x, 4x and 5x.

Document how you ran the experiments, the results and use appropriate statistical measures such as mean, median and standard deviations.

Analyse the results:

- How does performance and cost scale, why and why not?
- Under what circumstances does throttling happen? What is the cause?
- Do you see variation across time for the same workloads?

Write up your analysis. Think about what is going on underneath the service.

Completion "B+/A-"

Modify your script or program to allow concurrent invocation of the lambda function.

This should include a barrier wait that doesn't print out the results until you are done (A- territory).

Write a short overview of how to run your client and how it works.

Execute each function 100 times in each run.

You are not allowed to use Go to implement this i.e. you can't use the test harness from the tutorial.

Rerun your earlier experiments. Do you see any differences? Why or why not.

Document results and use appropriate statistical measures such as mean, median and standard deviations.

Challenge "A"

Use an API key to implement authorisation (https://www.gellock.com/2015/12/10/using-api-keys-with-aws-api-gateway/).

Document how you have implemented this and tested that it works.

Challenge "A+"

Create an EC2 instance and implement a dynamic web page allowing you to invoke the prime number lambda.

Your solution should support a more advanced authentication scheme that the simple API key based one, consider using custom authoriser.

Write a short overview of your implementation, give me the link to the page so I can test it.

What to Submit

For each stage please submit your code and raw output.

Submit a report (with your name on it) as a PDF file.
Approximately 10% of your grade is based upon presentation and readability.