Project 3: Serverless

Objective: The objective of this assignment is for students to gain some hands-on experience with Serverless technologies, specifically we'll use AWS Lambda.

This project will consist of a single part. Students are expected to submit the part in their git repos. Please note team sizes can be at most 2 students. ***If students have trouble with their partners (from Project 2 or otherwise), they should privately email the instructor (Eric) ASAP. Otherwise, assume the same project pairings for this project.*** Due to logistical reasons, we cannot accommodate requests to be paired with a specific person at this time (sorry!). Also please understand that we can reasonably create new pairs early-on, but our ability to do so becomes less as the deadline nears.

Updates:

- Oct 16: Clarified that y'all need two buckets. Clarified function of each bucket.
- Oct 15: S3 bucket name can be all lower-case
- Please reference this document frequently for updates.

Programming languages:

Lambda is supported by a variety of languages. We will allow for students to use a programming language of their choice, but **we will only officially support python.** We will try to help students with Java etc, but cannot guarantee any troubleshooting or debugging. We'll focus on each project's code functionality (does it work) and correctness for the grading.

Part A [100% of grade]

Objective: learn Lambda framework in AWS.

References for Lambda on AWS:

- https://aws.amazon.com/serverless/
- Invoking Lambda functions:
    - https://docs.aws.amazon.com/lambda/latest/dg/invoking-lambda-function.html

Lamba has a lot of cool mechanisms to invoke a function, but in this project we'll keep things simple. We are going to use S3 bucket notifications: whenever a file is placed in an S3 bucket, a corresponding function should be called.

We are going to simulate a set of tweets in this project. We will provide you with a script that will places a series of “tweets” into an S3 bucket. Tweets are generated for 30-35 minutes in this manner:

- 1 tweet a second for 10 minutes
- 200 tweets at 10th minute
- 1 tweet a second for 20 minutes
- *Note that it might take around 35 minutes to upload all 2000 tweets in your bucket.*

Initial Steps:

- Download and extract proj3.zip file from: https://drive.google.com/open?id=1LyvEgIWDDmgpFlXCf7NIUEfot2PlHLwL
- You will find the dataset folder “tweets” and a python file upload.py
- Before running the script, you should first create your s3 bucket (which will trigger the lambda function each time it receives a file)

To start uploading, execute this command: **python upload.py <your_bucket_name>**
(Note: If you face “Access Denied” issues while running the script, make sure your s3 bucket has write permissions and bucket policy for that user. Read more about it here: https://docs.aws.amazon.com/AmazonS3/latest/dev/using-iam-policies.html)

Your lambda function must capture all the tweets within the 35 minute block, and then perform the following analysis:

- Each S3 input tweet must be processed by your own Lambda (or set of Lambda) function(s) that perform the following:
  - Strip all the stop-words from the tweet (as defined in Project 1 and 2).
  - Convert all tweets to lower-case.
  - Get rid of all non-alphabetic characters (same as in Project 1 and 2).
- You must create your own S3 bucket, and do the following:
  - Name your S3 bucket CSCI5253-Fall2018-Project3-TeamName
    - **Naming is important, follow this instruction exactly!**
    - For now, keep this bucket private, but note we may ask students to mark this bucket as public *after* the submission deadline.
    - TeamName is defined as “LASTNAMEMEMBER1-LASTNAMEMEMBER2”
    - **Oct 16 update**: This is the bucket that tweets should go into. The tweets that go in this bucket will trigger your lambda function.
  - Keep the filename of the modified tweet the same as the filename in the input S3 bucket
    - **Oct 16 update:** You can create a second bucket (name whatever you like) to keep your modified tweets (ie, lambda outputs)

After 35 minutes are up (or after the script file terminates), your lambda functions should have captured all of the tweets and transformed them as outlined above. At this point, your S3 bucket should contain all the transformed tweets. For reference, the first tweet (sent at the top of the hour) should be:

- 1.txt

The last tweet (sent at the 35 minute mark) should be:

- 2000.txt

You can empty your bucket by executing: **python empty.py <your_bucket_name>**

You should then create a file named output.tar.gz (tar'd then gzip'd file) that contains all of the transformed tweets in your S3 bucket.

In addition, we will also hand-in (everyone's favorite) a word count of all the files in the modified S3 bucket. You can use Hadoop, Spark, a python script, whatever to perform the wordcount (or even more sets of lambda functions). The file should be named wordcount.txt. The file should be sorted by frequency (ties should be sorted alphabetically: so when there is a tie, the word "aardvark" would appear before the word "dog"), and the data format of your final output file should look like:

- <word1><tab><frequency> //omit the brackets
- <word2><tab><frequency> //and so on

What to turn in:

- Create a directory named "PartA" in your git repo.
- Put your output.tar.gz file and your wordcount.txt file into the directory:
    - PartA/output.tar.gz
    - PartA/wordcount.txt

- Create a directory named "PartA/Code" and put in the following:
    - All source code (and anything else needed to compile your code).
    - *(If required)* A Maven file (named pom.xml) to compile your code.
    - **Important difference from last project:** A file named Readme.txt that outlines how you invoked the lambda function, describes how the lambda functions work, describes what each/any Lambda does, and details how you integrated the function(s) into the AWS framework.
    - If you used a traditional technique (non-Lambda technique) to perform the wordcount, then you need to include:
        - A file, named "run.sh" that contains all commands to:
            - Run wordcount
            - Output result

        Change the permissions of run.sh to "a+x" (everyone can execute). The TA should be able to run the "run.sh" script to verify your program works and it can generate the output you turn in. In other words, the TA should

be able to download your github repo to his Amazon machine (which is running an EMR cluster) and type "./run.sh" into the command line after cd'ing into the appropriate github directory. The code should run through all steps necessary to obtain the output. You can assume the input files are located in an "input/" subdirectory within the the current directory. In other words, you can assume "input" holds the imdb data when we run the imdb test, and "input/" holds the holmes data when we run the holmes test. **Do not check in the input files to your github repo!**

- A "PartA/Readme.md" file (in the base directory) documenting your project with any special instructions or comments. **Document which language you used. Please note the version of python you are using if you are using python.**

Reminder:

- Students running low on credits:
    - Before running the script, you can create your own S3 bucket to test things on first (write a single file, invoke a function, do the tweet transform, etc).
    - Lambda should be pretty cheap, as well as S3, so the out-of-pocket costs (if credits are exhausted) should be low. See the instructor right away if this is an issue.

Additional turn-in files:

- In the root of your github folder, create a file called "Names.txt". In Names.txt, each row should contain the following information for each group member (one row per group member):
    - FullStudentName (\tab) CU email address (\tab) IdentiKey

Notes:

- Question: Why are the instructions so crazy? Does directory names and case sensitivity really matter?
    - Answer: Yes! We need to script the grading. Everything needs to be to spec.
- Check-in your code early and often. Document who has done what with each check-in!
- Remember that we have Google Cloud credits, Cloudlab.us resources, and you can use your local machine. Check moodle for slides on joining Cloudlab.
- **Remember to close all VMs, clusters, etc when not using them!**
- **All questions** should be posted publicly on Piazza. Students are encouraged to help one another. Students helping often will get extra credit towards their participation grade at the end of the semester. Remember to abide by all Conduct guidelines (ie: be polite).

- Remember the Academic Integrity policy from the course! Don't copy code without attribution. Clearly document what parts were copied. Try to be cool and not copy anything though. Writing from scratch will be useful later on.
- Let Eric know ASAP if there are any issues with the working relationships in your group.
- **For each git check-in,** write a comment on who was responsible for what parts of the commit. This will help create a paper trail to deal with any issues that may arise. If two students sat side-by-side to code parts of the program, that should be explicitly noted in the git comment for the commit.
- May the odds be ever in your favor!

Turn in instructions:

We are using Github classroom to turn in the project, and also to check project progress by team members. **The team name must be formatted as follows: project-3-serverless-LASTNAMEMEMBER1-LASTNAMEMEMBER2.** So, for example, if Eric Rozner was partnering with Kamal Chaturvedi, then the team name would be:

project-3-serverless-Rozner-Chaturvedi

Link to join the project 3 assignment via github classroom:

- https://classroom.github.com/g/kELThK5w

Some good github documentation:

- Merging conflicts
  - https://services.github.com/on-demand/merge-conflicts/
- Github guides:
  - https://guides.github.com
- Git "Hello World"
  - https://guides.github.com/activities/hello-world/
- Git ebook (free)
  - https://www.amazon.com/Rys-Git-Tutorial-Ryan-Hodson-ebook/dp/B00QFIA5OC

Due date:

- Tues, Oct 23 at 11:59:59 PM MST
- Please start early-- assume hiccups will occur! Students have already had issues with AWS, Vocareum, etc. Do not wait until the last minute to work on the project! Students with cloud issues *will not receive an extension!*