CIS*4010 Cloud Computing (Fall 2020)

Assignment #1 Part 1 – AWS S3 Storage

Description

Your first task is to use the AWS Boto3 SDK for Python to create an "S3 Shell". This shell will allow you to manipulate your S3 buckets and objects as if they were actually a file system. Your shell will allow you to do that following functions:

a) login <username>
- This command will allow you to authenticate based on the aws_access_key and the aws_secret_key that you have stored in a file called config.ini (this file must be in the directory from which you launch the shell). The config.ini format is as follows:

```
[DEFAULT]
AccessKey = AKIAIFJTPIPB4532YYYQ
SecretKey = XAIdskh1X412344p1y8EZaGABC12DRissQBCgaoUK
Region = ca-central-1

[deb]
AccessKey = AKIAIF12345HGHAHZ5TQ
SecretKey = XAIdsdkdkdkdk12345TTTTTIII12DRissQBCgaoUK
Region = us-east-1
```

The <username> is optional. If it does not appear than the DEFAULT values are used. If it does appear than the keys and region to be used will be in the section labelled with that username. AWS does not really have a notion of "login" for Boto3 but this authentication command will allow you to establish a session based on the authentication in a given configuration file.

b) logout / quit / exit
- This command will terminate your shell. You do not actually have to "logout" of the AWS S3 system so this command is just to terminate your program.

c) mkbucket <S3 bucket name>
- Similar to a mkdir command, this command makes an S3 bucket that will exist at the "root" level of the "file system directory tree".

d) ls < -l >
- This command lists either your S3 buckets ("root" of your "file system") or the objects in a bucket (these objects will be described using Unix style / notation). If the -l option is given then a long form description of the object will be given. This long form includes the object name, size, file type, date of creation and will look like the following:

```
$ python3 awsS3Shell.py
> login
> ls
-dir-    cis1300
-dir-    cis2500
-dir-    cis3110
-dir-    cis4010
> cd cis4010
> ls
-dir-  A1/
-dir-  CIS4010Assignment1/
       QuickStartS3.py
-dir-  images/
       seccat1.png
       stream1.png
> ls -l
binary/octet-stream          0            2020-08-26 19:40:22+00:00   A1/
application/x-directory      0            2020-01-15 21:15:36+00:00   CIS4010Assignment1/
binary/octet-stream          594          2020-08-24 14:01:36+00:00   QuickStartS3.py
application/x-directory      0            2020-01-02 21:55:40+00:00   images/
binary/octet-stream          70359        2020-06-08 00:26:05+00:00   seccat1.png
binary/octet-stream          678599       2020-01-09 17:43:51+00:00   stream1.png
> cd A1
> ls -l
binary/octet-stream          0            2020-08-26 19:40:22+00:00  .
> ls
> logout
$
```

e) Directory-like commands

- pwd
  The present working directory command will display where you are in the "directory tree" and use the Unix / format. When you are at the "root" of the tree (at the bucket level), then the shell will display the directory as "s3:/"

- cd <~ or .. or dir name>
  The change directory command will go down (and up) the tree structure of the file system. You can do a variety of things with this command (the more you do the higher your grade):
  - **cd ~** will go back to the root directory
  - **cd dirname** will go to dirname if this is doable
  - **cd ..** will go to the previous directory
  - any multiple level commands, *i.e.* **cd dir/subdir** or **cd ../..** will receive more grades

  You are required to do the first 2 versions of cd (B grade level) and the next two for an A grade level.

```
$ python3 awsS3Shell.py
> login
> pwd
s3:/
> ls
-dir-    cis1300
-dir-    cis2500
-dir-    cis3110
-dir-    cis4010
> cd cis4010
> pwd
s3:/cis4010/
> ls
-dir-  A1/
-dir-  CIS4010Assignment1/
       QuickStartS3.py
-dir-  images/
       seccat1.png
       stream1.png
> cd ~
> pwd
s3:/
> logout
$
```

- mkdir
    - The mkdir or make directory command will create an object that functions like a folder. This "directory" cannot be in the bucket level (root level) of the "file system".
- rmdir
    - Like the mkdir command, rmdir (remove directory) will delete an object that functions like a folder. This command does not remove a bucket at the root level.

f) Types of copy

- upload <local filename> <S3 object name>
    - The upload command will copy a file from your local file system to your S3 object store. The S3 object name will either consist of the total "pathname" of the object or it will put it into the "directory" that you are currently in.
- download <S3 object name> <local filename>
    - The download command will copy a file from your S3 object store to your local file system. The S3 object name will either consist of the total "pathname" of the object or the object will be assumed to be in the "directory" that you are currently in.

- cp <S3 object name> <S3 object name>
    - The cp (copy) command will copy an object from one S3 location to another. The S3 object name will either consist of the total "pathname" of the object or the object will be assumed to be in the "directory" that you are currently in.

g) mv <S3 object name> <S3 object name>
    - The mv (move) command will copy an object from one S3 location to another and delete the source object. The S3 object name will either consist of the total "pathname" of the object or the object will be assumed to be in the "directory" that you are currently in.

h) rm <object name>
    - The rm (remove or delete) command will delete the named S3 object. The S3 object name will either consist of the total "pathname" of the object or the object will be assumed to be in the "directory" that you are currently in.

Error Actions and Messages

For the following error conditions, your shell should print out an appropriate error message, not attempt to do the command, and continue to wait for another command:

- Trying to execute a command without logging in first
- Using a command that is not listed here
- The command does not have the appropriate number of parameters
- Either a bucket name or object name is incorrect (*i.e.* the bucket or object does not exist)
- You do not have permission to execute the command. For example, if you try to mkbucket and the bucket name is not unique, the command will fail.

Other Requirements

- The Python program will be called awsS3Shell.py
- The configuration file will be named config.ini

Testing

- When your code is being graded, a number of commands will be piped to your program and grading will look at the output from your "shell" and the state of your S3 object store during the execution of the commands. A bucket will be created called "cis4010testbucket12345". Please never use this bucket name for your own testing since each bucket name must be unique and if you take this name you will disrupt grading.

Assignment #1 Part 2 – AWS DynamoDB

Your second task involves the use of the NoSQL database, DynamoDB. A number of CSV files contain information from the OECD about agricultural production from various regions around the world. This information was taken and slightly modified from https://stats.oecd.org/Index.aspx?datasetcode=HIGH_AGLINK_2020#. You have two basic deliverables for this task:

1. Write a Python script that given a CSV file name and a table name (on the command line or as STDIN) creates a DynamoDB table representing a "region"/country. Each region contains the following CSV fields:

Field Designation	Description	Example	Type
commodity	Code for an agriculture product, such as Wheat.	WT – code for Wheat	String
variable	Code for an aspect of the product, such as Imports	QP – Production IM - Imports	String
year	Year – some are in the future since there are projections in the data	2010 to 2029	String
units	Unit of measure	TONNES	String
mfactor	Multiplication factor for Value in terms of powers of 10	6 = millions 3 = thousands 0 = no power factor, *i.e.* multiply by 1	String
value	Value for this entity	345.639	Float

You will have to determine what the two key fields should be (they can be combinations of fields – remember that the Partition key + Sort key must be unique).

You will find the following CSV files in my S3 cis4010 bucket:

CSV Filename	DynamoDB Table Name	S3 Object URL
northamerica.csv	northamerica	https://cis4010.s3.ca-central-1.amazonaws.com/A1/northamerica.csv
canada.csv	canada	https://cis4010.s3.ca-central-1.amazonaws.com/A1/canada.csv
usa.csv	usa	https://cis4010.s3.ca-central-1.amazonaws.com/A1/usa.csv
mexico.csv	mexico	https://cis4010.s3.ca-central-1.amazonaws.com/A1/mexico.csv

There is also a CSV file called 'encodings.csv' (https://cis4010.s3.ca-central-1.amazonaws.com/A1/encodings.csv) and it contains all of the encodings for the different fields

in the region files in the format: code, label, field. For example, the following row in this file: “IM”,”Imports”,”variable” is saying that you will see the code IM in the variable field and its label (expanded description) is Imports. In the appendix of this assignment description you will find a full list of all of the encodings for all appropriate fields. You will notice that the ‘region’ does not refer to a field but instead refers to a table name/CSV filename.

As you design your tables and code, remember that you must be able to query/scan the tables to retrieve either single records (What was the value of Wheat Imports to Canada in 2015?) or groups of records (Retrieve all commodity Imports to Canada in years > 2019?, i.e. all projected values).

You may also want to write a program to create a DynamoDB table that stores all of the encodings for use in the following program.

2. Write a Python program that allows a user to query/scan your tables to do some data analysis, i.e. to answer the following question:

   Do the values for the variables associated with a specified commodity, *e.g.* Wheat (WT), in the table ‘northamerica’ represent the sum of the related values for ‘canada’, ‘usa’, and ‘mexico’ or does North America only include ‘canada’ and ‘usa’? Your output should include a table showing all of the appropriate values for **all variables** of the specified **commodity** with a column that designates whether this variable supports the statement North America = Canada+USA+Mexico or the statement North America = Canada+USA or Neither. Each variable table will be followed by a statement of evidence (number of items that agree with your conclusion) as to what the definition of North America is according to the OECD. After all variables have been accessed, conclude with a statement of the total evidence (number of items that agree with your conclusion) as to what the definition of North America is according to the OECD. Figure 1 shows what the output is to look like except that only one variable has values. Also realize that you must only show the variables that are in common between northamerica, canada, usa, and mexico (hint: canada, usa, and mexico have more variables than northamerica does).

```
Commodity: Wheat

Variable: Area Harvested
...

Variable: Exports
...

Variable: Feed
...

Variable: Food
...

Variable: Imports
...

Variable: Human consumption per capita
...

Variable: Consumption
...

Variable: Production
Year  North America   Canada        USA           Mexico       CAN+USA     CAN+USA+MEX  NA Defn
2010    83365312.0    23299600.0    60065712.0    3676708.0    83365312.0  87042020.0   CAN+USA
2011    79692784.0    25288000.0    54404784.0    3627511.0    79692784.0  83320295.0   CAN+USA
2012    88536432.0    27246000.0    61290432.0    3274337.0    88536432.0  91810769.0   CAN+USA
2013    95695260.0    37589100.0    58106160.0    3357307.0    95695260.0  99052567.0   CAN+USA
2014    84581716.0    29442100.0    55139616.0    3669814.0    84581716.0  88251530.0   CAN+USA
2015    83766792.0    27647400.0    56119392.0    3710706.0    83766792.0  87477498.0   CAN+USA
2016    94981644.0    32139900.0    62841744.0    3862914.0    94981644.0  98844558.0   CAN+USA
2017    77733140.0    30377300.0    47355840.0    3503521.0    77733140.0  81236661.0   CAN+USA
2018    83503260.0    32201100.0    51302160.0    2943445.0    83503260.0  86446705.0   CAN+USA
2019    84602620.0    32347900.0    52254720.0    3290323.0    84602620.0  87892943.0   CAN+USA
2020    85508623.0    32635538.0    52873086.0    3355839.0    85508624.0  88864463.0   Neither
2021    86098913.0    33098503.0    53000410.0    3432850.0    86098913.0  89531763.0   CAN+USA
2022    86187905.0    33154342.0    53033564.0    3488550.0    86187906.0  89676456.0   Neither
2023    86323601.0    33257160.0    53066441.0    3548114.0    86323601.0  89871715.0   CAN+USA
2024    86629478.0    33488605.0    53140873.0    3617330.0    86629478.0  90246808.0   CAN+USA
2025    86997189.0    33778979.0    53218210.0    3688626.0    86997189.0  90685815.0   CAN+USA
2026    87350102.0    34077603.0    53272499.0    3762019.0    87350102.0  91112121.0   CAN+USA
2027    87643518.0    34316293.0    53327225.0    3832980.0    87643518.0  91476498.0   CAN+USA
2028    87949358.0    34578994.0    53370364.0    3902849.0    87949358.0  91852207.0   CAN+USA
2029    88273394.0    34862678.0    53410716.0    3973522.0    88273394.0  92246916.0   CAN+USA
North America Definition Results:  18  CAN+USA,  0  CAN+USA+MEX,  2  Neither
Therefore we conclude North America =  CAN+USA

Variable: Yield
...

Overall North America Definition Results: 150  CAN+USA,  0 CAN+USA+MEX, 30  Neither
Conclusion for all Wheat variables, North America = CAN+USA
```

Figure 1: Query/Scan Output

The commodity can be input either on the commandline or as STDIN. The variables that will be displayed are those that all region files have in common. Here are different ways that the program can be run:

$ python3 queryOECD.py WT
or
python3 queryOECD.py Wheat

$ python3 queryOECD.py
Enter the code for the commodity> WT
or
$ python3 queryOECD.py
Enter the label for the commodity> Wheat

Errors

All errors must be properly handled. Errors can include failure to create a table, incorrect table name, table already exists (so you can't create it again), specified commodity or variable does not exist, etc.

Testing

When the table creation and loading is tested it will be done in the TA's account so please do not authenticate within your program.

Appendix

commodity
"BD","Biodiesel"
"BT","Butter"
"BV","Beef and veal"
"CA","Casein"
"CH","Cheese"
"CT","COTTON"
"DDG","Distiller's dry grains"
"ET","Ethanol"
"FDP","Fresh dairy products"
"FH","Fish"
"FHA","Fish from aquaculture"
"FHC","Fish from capture"
"FL","Fish oil"
"FM","Fish meal"
"FT","Fertilizer"
"HFCS","High fructose corn syrup"
"MA","Maize"
"MK","Milk"
"MOL","Molasses"
"OCG","Other coarse grains"
"OIL","Oil"
"OOS","Other oilseeds"
"PK","Pigmeat"
"PM","Protein meals"
"PS","PULSES"
"PT","Poultry meat"
"RI","Rice"
"RT","ROOTS AND TUBERS"
"SB","Soybean"
"SBE","Sugar beet"
"SCA","Sugar cane"
"SH","Sheepmeat"
"SMP","Skim milk powder"
"SU","Sugar"
"SUR","Raw sugar"
"SUW","White sugar"
"VL","Vegetable oils"
"WMP","Whole milk powder"
"WT","Wheat"
"WYP","Whey powder"

variable
"AH","Area harvested"
"BF","Biofuel use"
"CI","Cow inventory"
"CR","Crush"
"EX","Exports"
"FE","Feed"
"FO","Food"
"IM","Imports"
"NT","Trade balance"
"OU","Other use"
"PC","Human consumption per capita"
"PP","Producer price"
"QC","Consumption"
"QP","Production"
"QP__MA","Ethanol production from maize"
"QP__SCA","Ethanol production from sugar cane"
"QP__VL","Biodiesel production from vegetable oil"
"ST","Ending stocks"
"XP","World Price"
"YLD","Yield"

unit
"HA","Hectares"
"KG_HAB","Kilograms per capita"
"LT","Litres"
"NATCUR_HL","National currency per hectolitre"
"NATCUR_TONNE","National currency per tonne"
"NBR","Number"
"NONE","No units"
"TONNE","Tonnes"
"TONNE_ANIMAL","Tonnes per animal"
"TONNE_HA","Tonnes per hectare"
"USD_BAR","US dollars per barrel"
"USD_HL","US Dollar per hectolitre"
"USD_TONNE","US Dollar per tonne"

mfactor
"0","Units"
"3","Thousands"
"6","Millions"