Mitchell Van Braeckel (1002297)

CIS*4010: Cloud Computing

Assignment 4

(Extended) Dec 4th, 2020

# Part 2 – Self Evaluation: What I Did, Why I did it, & What I Learned

Instructions:

"In this class we have looked at 3 cloud platforms: AWS, Azure, and Google Cloud. We have also looked at some of the popular functions on those platforms including storage, VM instances, serverless functions, and some PaaS offerings from AWS and Azure (image processing, AI, text processing, etc.)."

"For this assignment to end the assignment portion of the course, you are to explore the Cloud landscape looking for other platforms and/or other services on the three platforms studied that were not mentioned in the course."

- If you are using a **new platform**, then you should either:
  - Adapt one of the previous assignment to work on this new platform.
  - If the system is very specialized like Dwave, then take one of their tutorials or demos, get it working, and then add to it to do something interesting.
- If you are using one of the three platforms covered in class, but are using a service or feature that we did not cover, then you should either:
  - Take one of their tutorials or demos, get it working, and then add to it to do something interesting.
  - Use this service or feature to do something that you always wanted to do, or something that fits into some other project or assignment that you are doing independently or in another course.

"Yes, there is a great deal of choice in this assignment and you are probably thinking, 'Just what does she want?' But honestly, the assignment will be graded generously if you document what you are doing, and you show some imagination and/or interest in what you have chosen to do."

"Trust yourself and follow your interests and explore!"

- What do you need to hand in?
  - In the A4 GitLab repo, put all source code/scripts and a README file that explains how to run your code.
  - Also, put a small document (text or PDF) explaining:
    - What you did,
    - Why you did it, and
    - What you learned.
  - This report does not need to be more than 2 pages in length.

Answer: See the next 2 pages, which answer the 3 questions overall:

**Approach:** Using AWS: S3 Glacier Vault + SNS (Simple Notification Service)

I did not follow any tutorials, but instead looked into documentation of how Glacier and SNS worked, as well as API documentation for Python 3 Boto3 Glacier and AWS CLI Glacier. I just wanted to try this out because it piqued my curiosity. This was initially interesting to me because it has a cool name, we dealt with other storage S3 buckets and other containers before, it is an immutable archival storage system (i.e. after an archive is uploaded, both its description and itself cannot be edited).

**How I got the Idea**

I was looking at how memory space on my laptop is slowly getting full of lots of old schoolwork stuff and was thinking about alternative cloud solutions to store all my stuff away besides just uploading to my Google Drive or One Drive or something. Although, I decided I won't be using Glacier to do this for myself (for storing old school stuff), I thought it would be fun to try using it and seeing what an immutable storage system is like that is purposed for actual archives. To clarify, I think of archives as super old info people will barely ever need to access, but copies still need to be kept. FUN FACT: After all, "Glacier" is a very fitting name as glacier ice tens to be nearly 100 years old, with the oldest Alaskan glacier ice ever recovered is about 30,000 years old (from a basin between Mt. Bona and Mt. Churchill). This also plays along with how I interrupt archives (explained above), and vaults (durable safekeeping for long periods of time where things are stored but rarely ever withdrawn). I figured that Glacier could be used in many different ways, or rather implemented/integrated for many different things, as an archival storage system for old data has passed a certain date-time threshold to become archived, and then these archives are deleted completely after a even longer amount of time has passed.

I was also intrigued by the AWS SNS notifications, but one of my friends had already mostly finished an A4 using this, so it didn't feel right to do the same thing. By using Glacier, I still got to investigate SNS and how it works since I could make life easier and design the Glacier vault email notifications are sent out when a retrieval operation finishes. This was done by connecting a SNS topic to the Glacier vault and enabling notifications for both retrieval operations, then adding email subscriptions to the topic so they would receive notifications when messages are automatically published to the SNS topic when Glacier vault retrieval operation(s) complete. Plus, as I was looking at different cloud services, I might be able to use for A4, Glacier stood out because it is a cool word!

**Takeaways**

I can definitely see how this would be useful for archival purposes; however, I am personally very uneasy about the large costs for expedited tier and the very long 3-5+ hours retrieval operations (inventory, archives) and that inventory updates only once a day. The few times one may want to withdraw something, it becomes very difficult. For example, with my idea of storing old school stuff, if I was doing work next semester and wanted to reference something from a previous semester, I would have to wait 3-5 hours before actually being able to download it again. From my personal view, this seems like something I would use to throw things away and forget about them (or at least because you believe you will never need to access it again).

Until inventory updates (approximately the next day), you won't know it's present in the vault inventory unless you keep track of things added yourself. Also, if the user doesn't add descriptions to things, they won't even know what each archive is later on. Moreover, it's unlikely for someone to save

all the Archive IDs, so waiting for inventory retrieval operation is required. In addition, Job IDs only last 24 hours, so the user would need to do a retrieval operation again if they still needed that JobID info the next day. Overall, I mostly discovered the pain of working with immutable archives with an extremely long (multiple hours, nearing 1/4 of the day). This made developing and testing kind of pointless and I was better off thoroughly researching and planning, then coding and testing at the end rather than incrementally developing and testing like how I usually do. So, on the bright side, I became much more resourceful in this regard and was able to figure out how to do everything on my own, especially because nobody else (none of my friends) did the same thing and there wasn't even bouncing of ideas or links/resources. To continue, there was not much documentation of how to do this in python. The AWS website for Glacier stuff only had examples for Java and .NET SDKs, so I just logically learned about Glacier and how it works before diving into APIs for Boto3 Glacier and AWS CLI Glacier. Consequently, I mostly just kept searching the internet and googling things to find out how to do specific things, and then put everything together and test, making small fixes and adjustments as I saw fit. Fortunately, this approach payed off and there were no big issues because I researched a lot more than usual beforehand. Although it was very interesting, I don't think I would want to deal with this kind of stuff and believe using Glacier is suited for a purely archival purpose. For example, old transactions could be stored as archives after a certain number of years, then fully deleted even later once I designated threshold of time is reached.

Furthermore, regarding the pain of working with Glacier and the lack of python documentation, I really want to mention AWS Console. I had taken the console for granted, when it's actually really useful and helpful. I learned that I was severely underappreciating the convenience of the AWS console while working with Glacier, since it barely has any support. It can only Create and Delete vaults, manage some settings, and display a very limited amount of information. In order to upload files as archives, find out what archives are actually in the vault inventory, or download archives as files, you need to use code/scripts and/or AWS CLI for Glacier: First, uploaded archives don't appear until the vault inventory updates, which happens once per day and you can't force the inventory to update somehow either. Also, the first uploads don't show up and you aren't able to retrieve anything until a half to a full day has passed. Second, retrieving inventory or archives from the vaults requires 2 operations each: initiating a retrieval, waiting for it to finish/complete and then getting the job output; where the retrieval job takes ~3-5 hours to complete (but expedited tier retrieval operations, even if only 1-5min long, are VERY costly: $100USD/month to buy a provisioned capacity, and the use rate fee is also many times more expensive). Third, you need to track Archive IDs and Job IDs yourself -- If you forget, you need to retrieve inventory again, and make another archive retrieval job, each which will take a long time to complete. Fourth, you can't even see your inventory or archives unless you start an inventory retrieval job (which takes a long time), even then, you still don't really know what's in the vault (unless you have really good descriptions on everything). Even if you have good descriptions, you can't actually see the archives (eg. file contents, folder contents, image previews, etc.)

Although, I could've just used AWS CLI only, felt like I would've been taking it to easy (even if I made something like bash scripts that used AWS CLI Glacier commands), so I used Python 3 Boto3 like in most of the other AWS Cloud assignments. After dealing with multiple AWS services using Python3 Boto3, I feel pretty confident using this now (or at least much more confident in practical application compared to using other cloud services, eg. Azure, Google Cloud, etc.). In summary, I really appreciate the convenience of the AWS Console, as well as the ability to use AWS CLI to just test out services and view things a little easier before developing my own scripts -- without these, it becomes a lot more difficult and frustrating when you misinterpret things and find out later on and then need to change/redo things.