HTML Edgar Scraping

Team 8

Goal:

The goal of this assignment is to scrap the 10K/10Q files from the Edgar website. A pipeline has to be created to fetch the data from the HTML, scrap the data into informed csv files and store the data into Amazon S3. Our Team has chosen python to implement this scraping.

Given:

We have been given the company with CIK (company ID) XXX (omitting leading zeroes) and document accession number YYY (acc-no on search results) and we need to programmatically generate the url to get data.

Steps to process:

1. We first create a Dockerfile in a working directory with a base image and install python and list all the required packages in the Dockerfile
2. We set the Working directory for the container in the Dockerfile. This Working directory is used to store files in the container
3. We then start building a python script to scrape the HTML data into tables, which will further be saved as individual CSV files
4. We first check the File name whether it is 10q or 10K
5. After checking the file, we then move on to scraping all the usable tables from the HTML page.
6. We then clean the Tables and remove the extra spaces and ‘$’ signs from the Table
7. Once, these tables are clean, we then store these individual tables in CSV’s.
8. These CSV’s are then zipped from the working directory and pipelined to be sent to Amazon S3 storage.
9. In order to access amazon S3 we need to have an amazon access and secret key.
10. Parameterized S3 Keys are present, so that you can store the Files on any eligible S3 machine
11. Amazon S3 storage needs a Bucket- which stores the information that we send.
12. The python script contains the code to check the URL, scrap the tables and stores the tables on S3
13. We then add this python script to the Dockerfile and also add the python script in the Docker working directory.
14. Once the Dockerfile is updated and the python script is placed in the working directory, we then build the image on docker
15. A build image can be viewed by docker ps -a
16. We then run the image and start the container
17. After the container runs, we then set the parameters in the container and run the python script
18. Check the zipped files in the S3 bucket