Virginia Tech

Bradley Department of Electrical and Computer Engineering

ECE 5984 Data Engineering Project
Fall 2023

Assignment 1
Data Pipelines

Please note the following:

- Solutions must be clear and presented in the order assigned. Solutions must show the work needed, as appropriate, to derive your answers.
- Data being processed in both the lab and homework should be in the correct format and location and any other deliverables should also be completed as mentioned in the modules.
- For the Homework, the submission process requires that all code files should be uploaded to Canvas before the deadline.
- Submit your Homework using the respective area of the class website by 11:55 p.m. on the due date.
- When a PDF file must be submitted, include at the top of the first page: your name (as recorded by the university), email address, and the assignment name (e.g., "**ECE 5984, Homework/Project 1**"). Submit a single file unless an additional file is explicitly requested.

Lab 1

1.1 Perform Batch ingestion to store data on an S3 bucket (data lake) using Apache Airflow and Python APIs

We will be performing batch ingestion of data from a mock dataset. The dataset that we will be using is the historical stock market data. We will be using Python APIs like pandas datareader (https://pandas-datareader.readthedocs.io/en/latest/) and the Yahoo finance API (https://pypi.org/project/yfinance/)
Since these APIs are dataset specific we have to install the Python libraries necessary for them manually.

Code on your local machine

1. Open a new project in pycharm on your local machine
2. Install the same Python packages needed for the dataset being used. In our case the pandas-datareader and the yfinance libraries
3. Save the dag.py file and the batch_ingest.py file provided into the project location
4. Open the batch_ingest.py file and go through the code. This code retrieves the stock market data from the past year for 3 companies namely Apple, Google and Amazon. You can choose different companies as well.
5. After retrieving the data it also adds some noise to the dataset to simulate a real-world scenario where you interact with messy data. And then after retrieving this data the Python code pushes the data to a S3 bucket.
6. Create the correct S3 bucket to push the data into. Make a folder labeled ‘Lab1’ inside the S3 bucket provided to you and copy its S3 URL. This should be set as the DIR variable in the code. Refer to the companion video labeled ‘Access S3 bucket.mp4’

Push the code onto the cloud and execute the lab

7. Spin up a docker container (shown in Lab 0, step 2.2)
8. Install the needed packages for the dataset running the following commands
   a. `pip install pandas-datareader`
   b. `pip install yfinance`
9. Navigate to the airflow/dags folder by typing the following command:
   a. `cd airflow/dags`
10. Create a new dag.py and copy all the code from your local machine dag.py onto it. To do so enter the command:
    a. `sudo nano dag.py`
11. This will open a command line based text editor. Copy the contents of your local dag.py file to the newly created one inside your container (Tip: use ctrl+shift+V to paste content directly if using the web browser)
12. Save by pressing ctrl+x and then hit Enter
13. Similarly, create a batch_ingest.py file and copy code from local machine batch_ingest.py onto it using the same steps mentioned earlier
14. Enter command `cd ..` to come out of the dag folder you are currently in
15. Enter command `cat standalone_admin_password.txt` to display the contents of the text file. The contents of the text file is your password and user is as admin (explained in Lab 0)
16. Now access your airflow GUI as described in step 2.4: Launching and accessing airflow GUI of lab 0. Not since you are already inside your docker container there is no need to perform step 1 of spinning up a new container
17. You should see a **batch_ingest_dag**. Click it and go to graph. On the right side click the play button and press trigger DAG

18. This triggers the dag we just uploaded which inturn runs the ingest_data function from batch_ingest.py and you are able to see the whole process
19. After the DAG finishes running you should be able to navigate to your S3 bucket directory and check to see the data now there in a pickle file format (Companion video: Access S3 bucket.mp4

1.2 Perform streaming ingestion to store data on an S3 bucket (data lake) using Apache Kafka and Python APIs

We will be performing streaming ingestion now using Apache Kafka. In order to run Apache Kafka we need a bunch of other services working as well. We need to have zookeeper running first and then we need our kafka broker to be running. We also need to create a Topic manually. In our lab we will be creating a Topic called 'StockData' where we will be storing real-time stock market data of 3 companies namely Apple, Google and Amazon (similar to lab 1.1). We will use our local machine to produce said data and push it to our kafka server simulating a real time finance api that pushes data continuously. Next we will use our airflow service to start a consuming DAG that will push this data to our datalake (S3 bucket) where we store it in a .json file format

Code on your local machine

1. Open a new project in pycharm on your local machine
2. Install the same python packages needed for the dataset being used. In our case the yfinance library.
3. Save the dag_stream.py file, produce_stream.py file and the stream_ingest.py file provided into the project location
4. Open produce.py file and go through it. This is a Python script to make a Kafka producer that pushes data acquired from the yfinance library in real time. By default it produces data for a minute when run but this can be changed within the code by changing the length of the while loop.
5. Please notice that in one of the lines of the code, you need to fill out the ip and port number indicated by their place holders <ip> and <prt> in the bootstrap server variable. This we will fill later when our kafka server is up and running and we are ready to push and receive real time data. There we suggest to just keep this file open and come back to it later.
6. Open stream_ingest.py and dag_stream.py file and go through it. These files will control our airflow session. Fill in the S3 bucket location provided in the stream_ingest.py file. Here again we need to fill <ip> and <prt> which we will come to know in the upcoming steps. Save any and all changes made

Push the code onto the cloud and execute the lab

1. Spin up a docker container (shown in Lab 0, step 2.2) however instead of the command mentioned in step 2 of the mentioned procedure use the following command instead:
   a.
   ```
   docker run --rm -it --entrypoint bash -v /home/ubuntu/efs-mount-point/students/<pid>/root:/root -p 8080-8131:8080 -p 9092-9143:9092 --name <pid> <image-name-or-id>
   ```

   This time we are adding additional ports to be mapped so as to access the kafka server we are about to deploy
2. Install the needed packages for the dataset using the following commands: (optional if already done in lab 1.1 and you are using the same container)
   a.
   ```
   pip install yfinance
   ```
3. Navigate to the dags folder using `cd airflow/dags`
4. Open a new browser tab and connect back to your EC2 instance (shown in Step 2.1: Accessing the EC2 instance)
5. Run the command `docker ps`
6. Note down the mapped airflow and Kafka port numbers. The mapped airflow port number is the 4 digit port number left of '->8080' and the mapped kafka port number is the 4 digit number left of '->9092'
7. Also note down the Public IPv4 address of the EC2 instance shown in the details tab of the EC2 instance.

8. Go back to your produce.py and stream_ingest.py file and fill the <ip> and <prt> place holders where <ip> is the Public IPv4 address of the EC2 instance and <prt> is the mapped **kafka** port and then save it.
9. Create a new dag_stream.py and copy all the code from your local machine dag_stream.py onto it as we did earlier in lab 1.1 step 10
10. Similarly create a stream_ingest.py file and copy code from local machine onto it using the same steps
11. Start your airflow service and access the GUI

Start up Kafka

12. Open a new tab and connect to the EC2 instance
13. Type the following command:
    a. `docker exec -it <pid> bash`

    This lets you enter the container you have already launched
14. Type the following command
    a. `cd /home/kafka_2.13-3.5.0/`
15. Next type the following commands:
    a. `export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"`
    b. `sudo nano config/server.properties`
16. Navigate to the line in the file that says:

    #advertised.listeners=PLAINTEXT://0.0.0.0:9092

    And change it to the following:

    advertised.listeners=PLAINTEXT://<ip>:<prt>

    Where <ip> and <prt> should be replaced by the ip address along with the port number for Kafka (refer step 7)
17. Press ctrl+x and enter to save and exit
18. Next we need to start zookeeper. Zookeeper is what Kafka uses to coordinate a variety of distributed tasks, including but not limited to storing metadata, tracking the status of Kafka Brokers, etc. Run the following command to start Zookeeper:
    a. `bin/zookeeper-server-start.sh config/zookeeper.properties`
19. Open up a new browser tab and repeat steps 12, 13 and 14
20. Run the following command:
    a. `bin/kafka-server-start.sh config/server.properties`

    This should start up your kafka broker
21. Open up another browser tab and repeat steps 12, 13 and 14 again.
22. Run the following command:
    a. `bin/kafka-topics.sh --create --topic StockData --bootstrap-server <ip>:<prt> --replication-factor 1 --partitions 1`

    Where 'StockData' is the Topic name and <ip>:<prt> is the ip address with port number of the Kafka server
23. **Optional:**

    You can open up a similar consumer we plan to deploy on airflow in the console. This can be done by opening yet another browser tab and following steps 12, 13 and 14 and the running the following command:

    ```
    bin/kafka-console-consumer.sh --topic StockData --bootstrap-server <ip>:<prt>
    ```

    This command prompt can be useful for debugging and seeing if the consumer is working with your producer
24. Open another browser tab and follow steps 12 and 13
25. Insert command `airflow standalone` to start up airflow
26. Access the airflow GUI as was down in lab 0 (Step 2.4: Launching and access the airflow GUI)

27. You should see a **stream_ingest_dag in the list of dags shown**. Click it and go to graph. On the right side click the play button and press trigger DAG
28. At the same time Go back to your local machine and run the script produce_stream.py. You should now be producing real time stock market data for a specific amount of time (default 1 min) from your local machine. If you had done the optional step you should see the real time data flowing into the command prompt aswell
29. After produce_stream.py has done running you should be able to navigate to your S3 bucket directory and see that you have ingested stock market data directly and saved in json format
30. After checking your data reached the right destination go ahead and close all the services and the container you were working out of

Homework 1

1.1 Batch Ingestion

Perform the same batch ingestion as performed in Lab 1.1 to store data on an S3 bucket (data lake) using Apache Airflow and Python APIs but using a different dataset than the one provided in Lab 1.1

Some example datasets that can be used:

1. https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs
2. https://www.kaggle.com/datasets/bharatnatrayn/movies-dataset-for-feature-extracion-prediction

Deliverables:

1. The dag.py file and the batch_ingest.py for the homework should be uploaded to canvas.
2. Stock data from homework should be in the appropriate S3 bucket location.
3. Data from the lab should be in the appropriate S3 bucket.

1.2 - Streaming Ingestion

Perform the same streaming ingestion as performed in Lab 1.2 to store data on an S3 bucket (data lake) using Apache Kafka and Python APIs but using a different dataset than the one provided in Lab 1.2.

Some example datasets that can be used:

1. https://developer.twitter.com/en/docs/tutorials/stream-tweets-in-real-time
2. https://openweathermap.org/api

Deliverables

1. The dag.py file, produce_stream.py file and the stream_ingest.py file for the homework should be uploaded to Canvas.
2. Stock data from homework should be in the appropriate S3 bucket location.
3. Data from the lab should be in the appropriate S3 bucket.