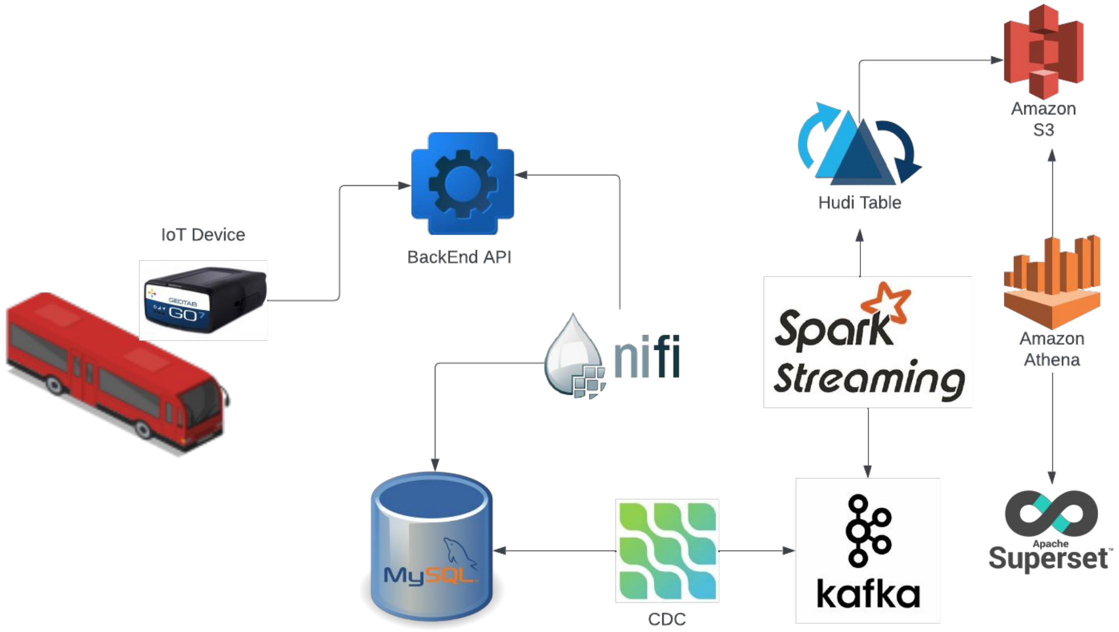
WeCloud Data Engineering Final Presentation

Nathan Ling
EST Batch 7

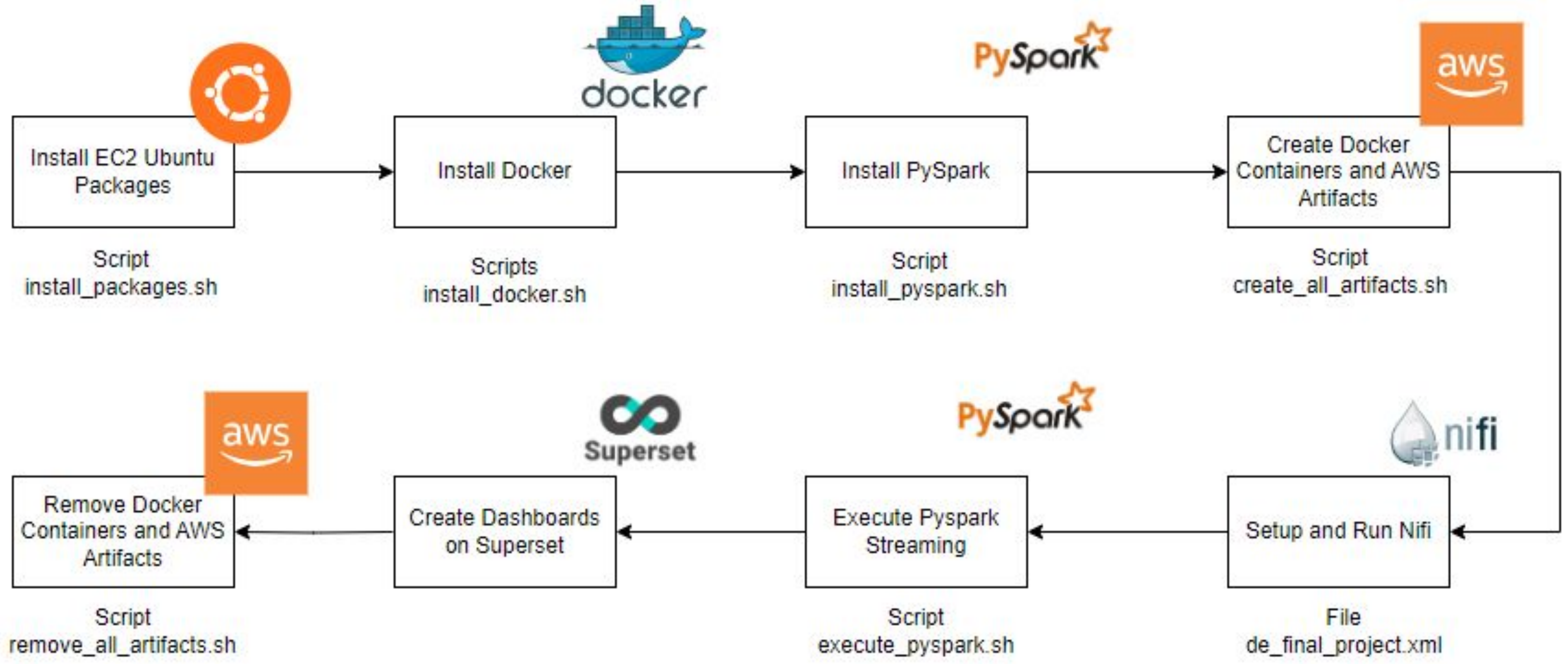
Data Architecture

Objective is to collect, store, and analyze TTC real time bus data using dashboards



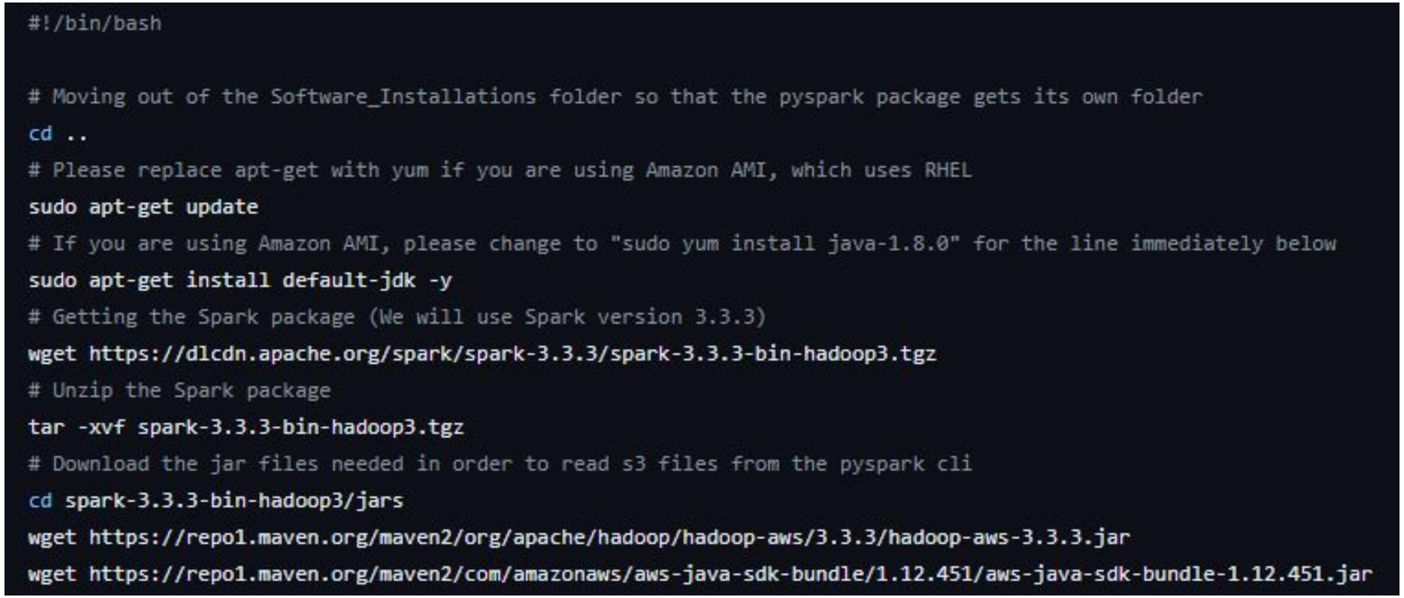
User Journey

Minimize user work



EC2 Ubuntu Installations

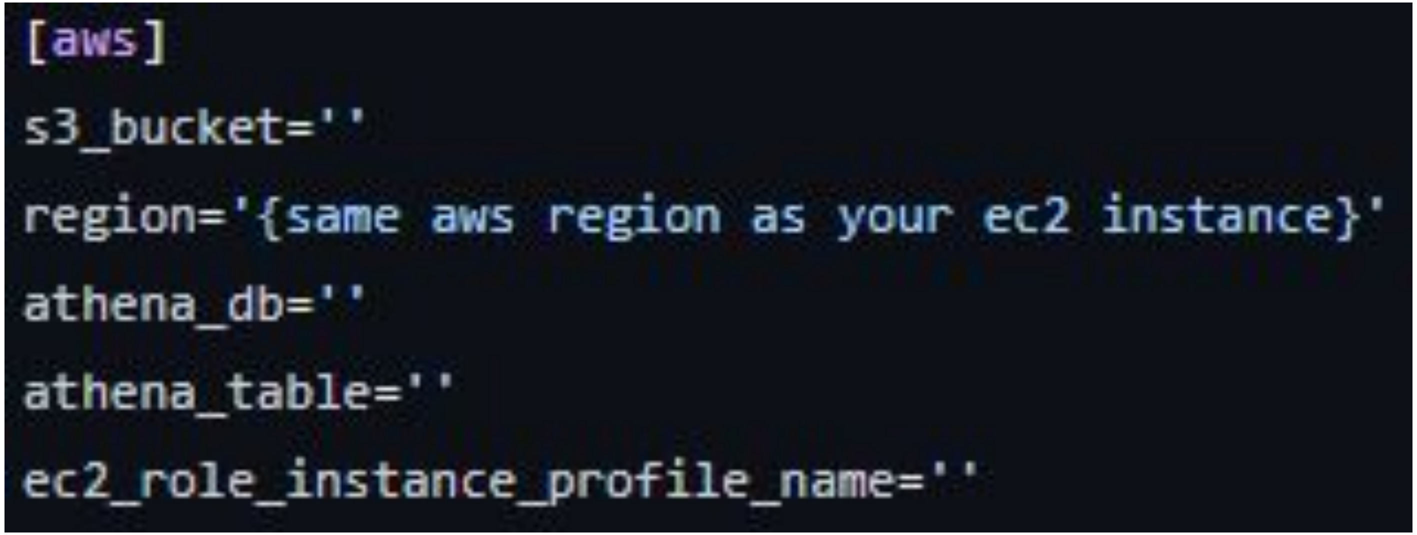
- Prevent errors downstream
- Minimize the pain of setting up the packages and programs, which include
  - Pyspark
  - Docker
  - Boto3
  - Awscli
  - Toml



```bash
#!/bin/bash

# Moving out of the Software_Installations folder so that the pyspark package gets its own folder
cd ..
# Please replace apt-get with yum if you are using Amazon AMI, which uses RHEL
sudo apt-get update
# If you are using Amazon AMI, please change to "sudo yum install java-1.8.0" for the line immediately below
sudo apt-get install default-jdk -y
# Getting the Spark package (We will use Spark version 3.3.3)
wget https://dlcdn.apache.org/spark/spark-3.3.3/spark-3.3.3-bin-hadoop3.tgz
# Unzip the Spark package
tar -xvf spark-3.3.3-bin-hadoop3.tgz
# Download the jar files needed in order to read s3 files from the pyspark cli
cd spark-3.3.3-bin-hadoop3/jars
wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/3.3.3/hadoop-aws-3.3.3.jar
wget https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk-bundle/1.12.451/aws-java-sdk-bundle-1.12.451.jar
```

Config Files

1. Ensures all the code is referencing the same source of truth
2. Minimizes user work
   a. Will be tedious to change each .sh or .py file to manually edit variables

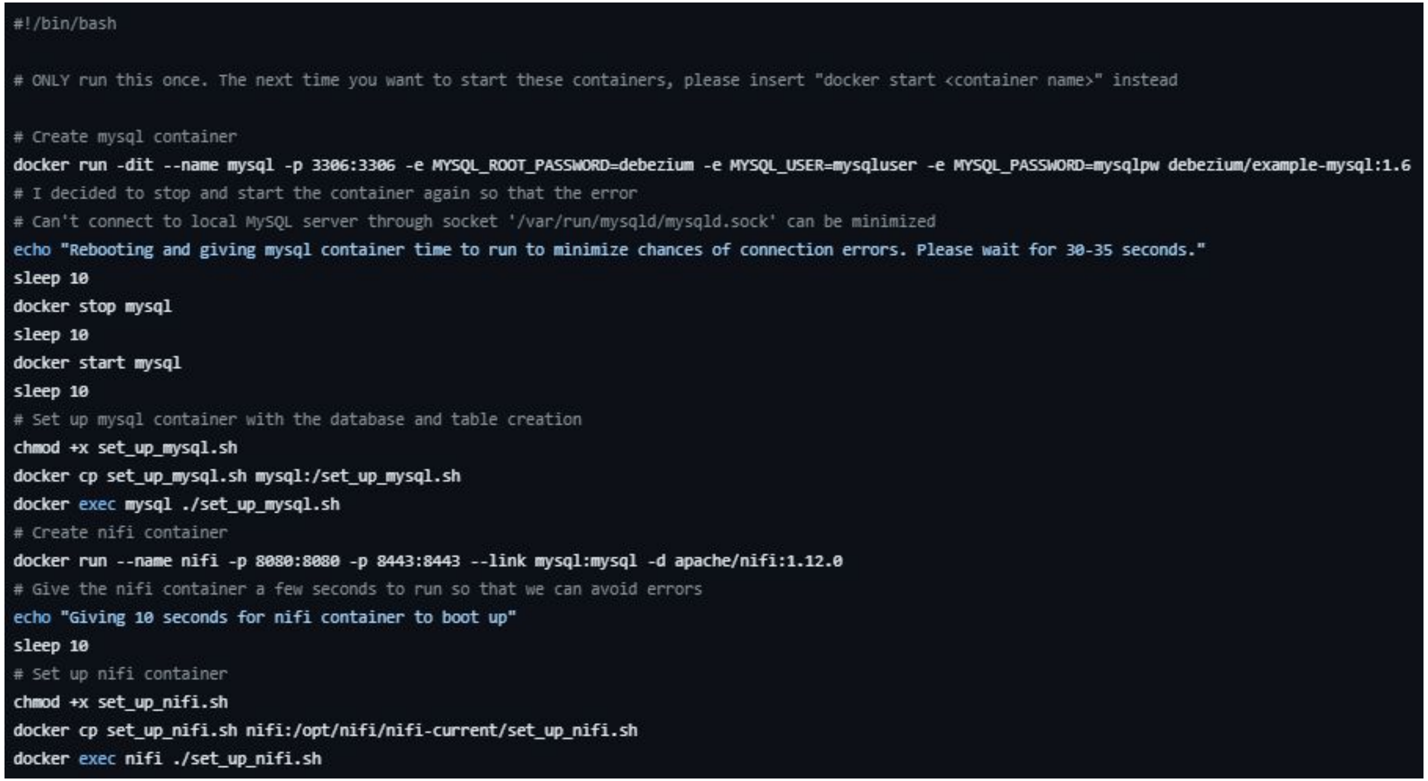


```
[aws]
s3_bucket=''
region='{same aws region as your ec2 instance}'
athena_db=''
athena_table=''
ec2_role_instance_profile_name=''
```

Create Docker Containers

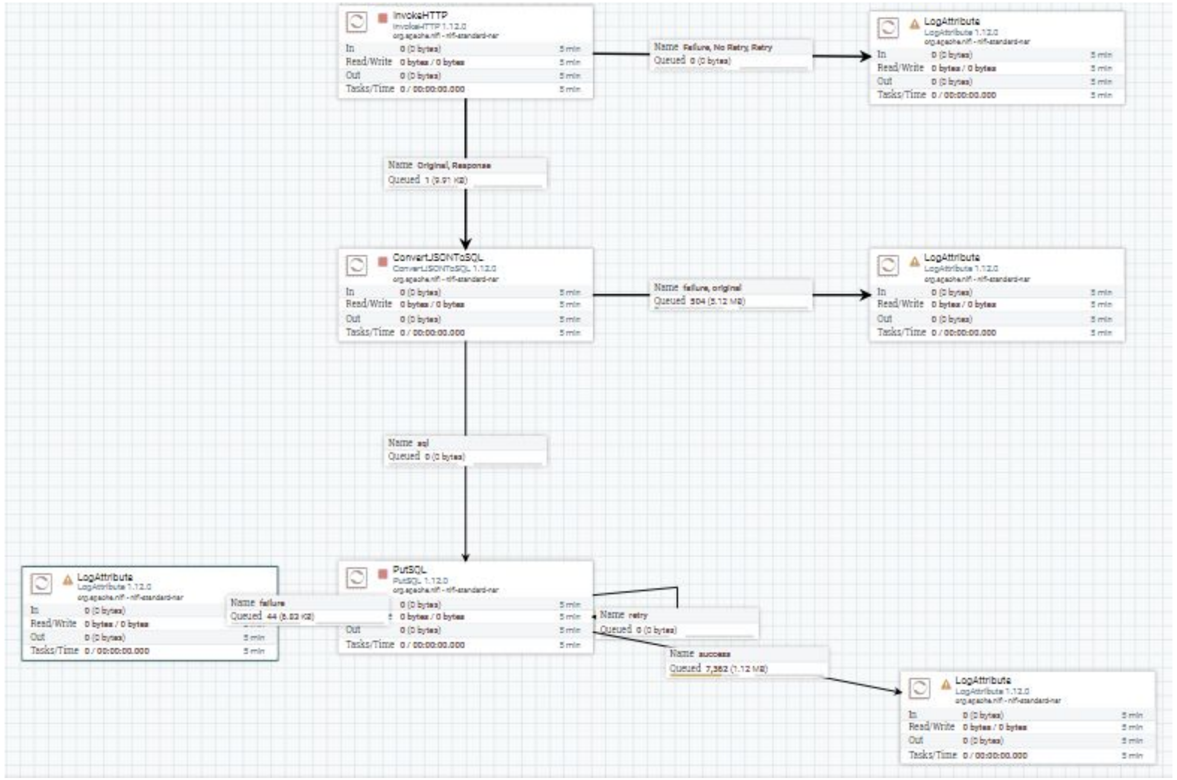
Container	Purpose
MySQL	Store streaming data coming in
Nifi	Capture, transform, and insert TTC streaming data to MySQL
Zookeeper	Metadata storage for Kafka
Kafka	Consume data changes from MySQL
Connect	Capture changes from MySQL database and pass this to Kafka
Superset	Create dashboards

Create Docker Containers Cont.



```
#!/bin/bash

# ONLY run this once. The next time you want to start these containers, please insert "docker start <container name>" instead

# Create mysql container
docker run -dit --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=debezium -e MYSQL_USER=mysqluser -e MYSQL_PASSWORD=mysqlpw debezium/example-mysql:1.6
# I decided to stop and start the container again so that the error
# Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock' can be minimized
echo "Rebooting and giving mysql container time to run to minimize chances of connection errors. Please wait for 30-35 seconds."
sleep 10
docker stop mysql
sleep 10
docker start mysql
sleep 10
# Set up mysql container with the database and table creation
chmod +x set_up_mysql.sh
docker cp set_up_mysql.sh mysql:/set_up_mysql.sh
docker exec mysql ./set_up_mysql.sh
# Create nifi container
docker run --name nifi -p 8080:8080 -p 8443:8443 --link mysql:mysql -d apache/nifi:1.12.0
# Give the nifi container a few seconds to run so that we can avoid errors
echo "Giving 10 seconds for nifi container to boot up"
sleep 10
# Set up nifi container
chmod +x set_up_nifi.sh
docker cp set_up_nifi.sh nifi:/opt/nifi/nifi-current/set_up_nifi.sh
docker exec nifi ./set_up_nifi.sh
```

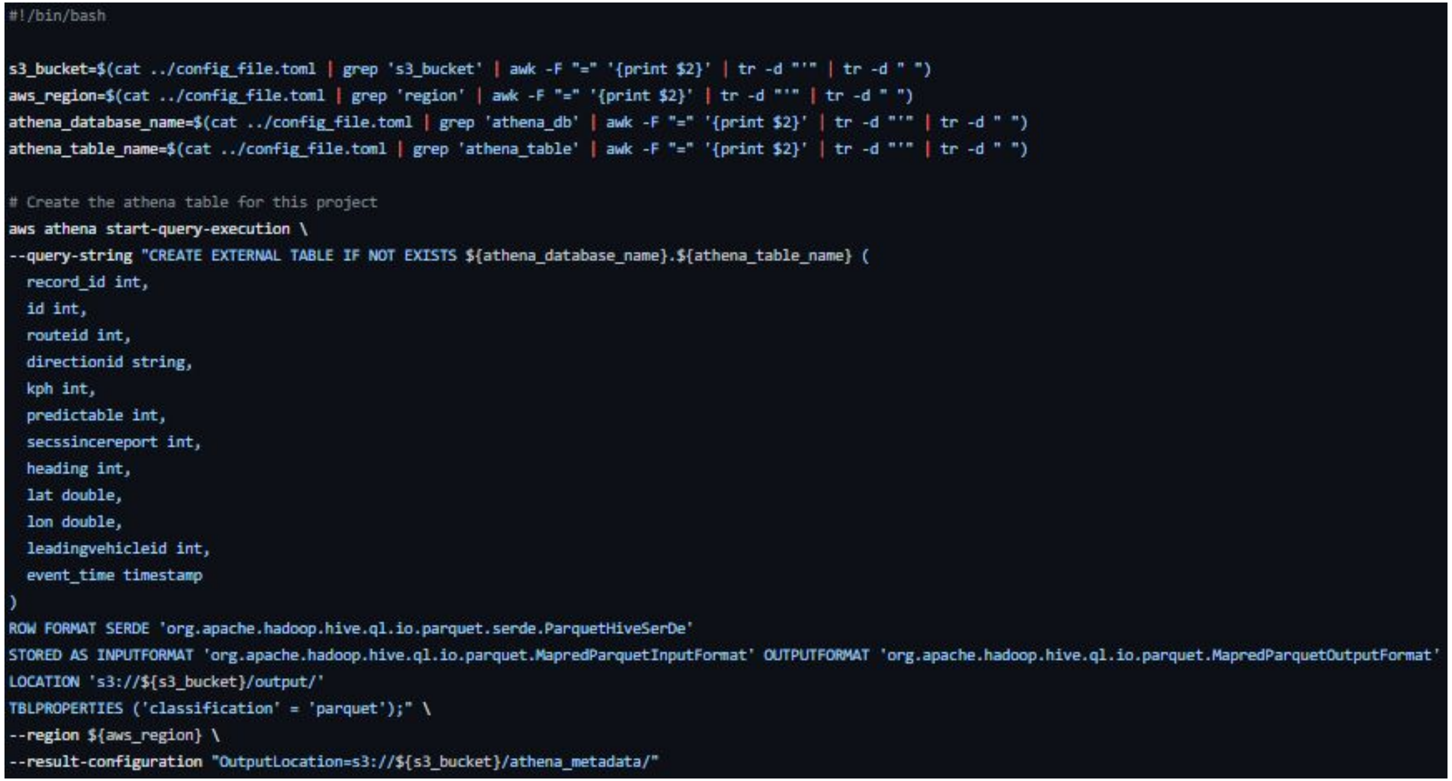
Nifi Workflow



Create AWS Artifacts

Microservice	**Purpose**
S3	Store streaming data in a data lake
Athena	Create a database and table from the S3 data
IAM Instance Profile	Enable spark streaming to read from and populate S3 bucket

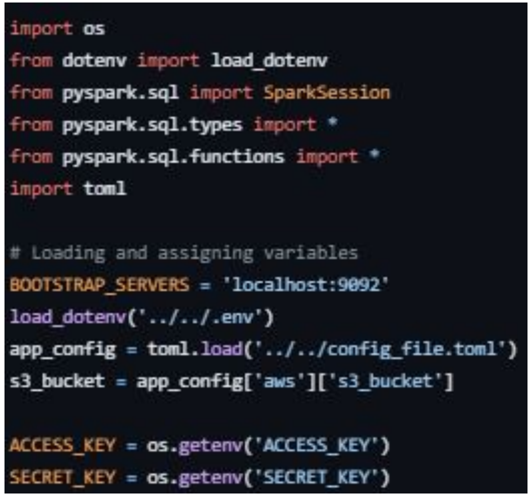
Create AWS Artifacts Cont.



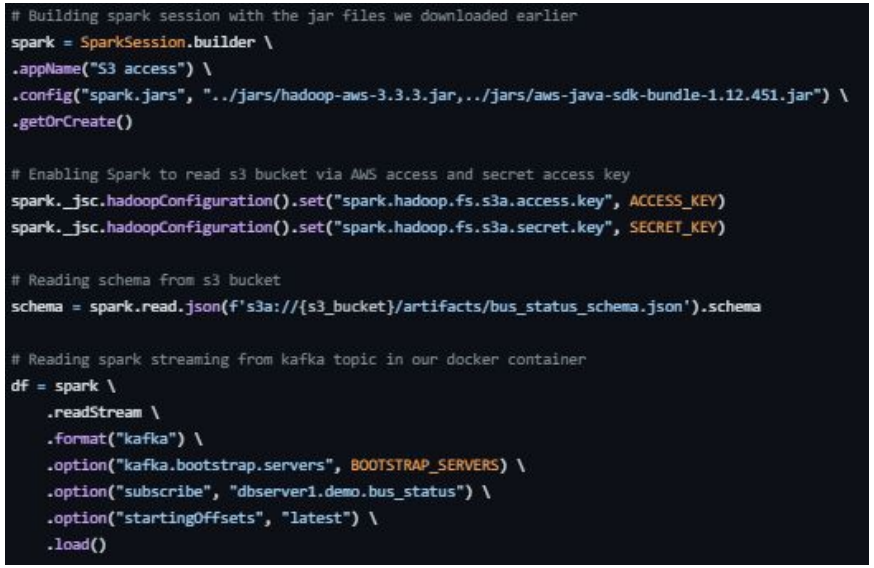
```
#!/bin/bash

s3_bucket=$(cat ../config_file.toml | grep 's3_bucket' | awk -F "=" '{print $2}' | tr -d "'" | tr -d " ")
aws_region=$(cat ../config_file.toml | grep 'region' | awk -F "=" '{print $2}' | tr -d "'" | tr -d " ")
athena_database_name=$(cat ../config_file.toml | grep 'athena_db' | awk -F "=" '{print $2}' | tr -d "'" | tr -d " ")
athena_table_name=$(cat ../config_file.toml | grep 'athena_table' | awk -F "=" '{print $2}' | tr -d "'" | tr -d " ")

# Create the athena table for this project
aws athena start-query-execution \
--query-string "CREATE EXTERNAL TABLE IF NOT EXISTS ${athena_database_name}.${athena_table_name} (
  record_id int,
  id int,
  routeid int,
  directionid string,
  kph int,
  predictable int,
  secssincereport int,
  heading int,
  lat double,
  lon double,
  leadingvehicleid int,
  event_time timestamp
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://${s3_bucket}/output/'
TBLPROPERTIES ('classification' = 'parquet');" \
--region ${aws_region} \
--result-configuration "OutputLocation=s3://${s3_bucket}/athena_metadata/"
```

Spark Streaming

1. Capture streaming data from Kafka topic
2. Parse and transform streaming data
3. Output streaming data to S3 bucket in Hudi open table format

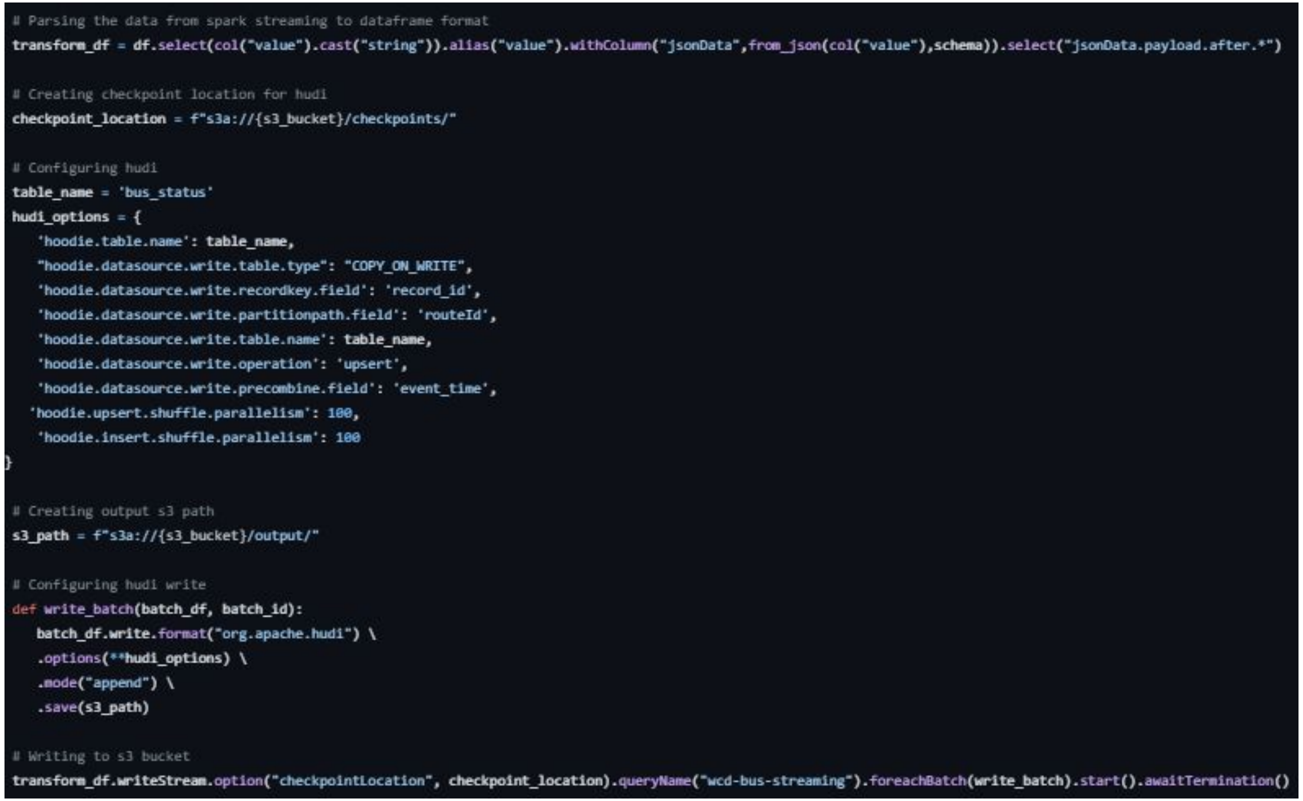


```
import os
from dotenv import load_dotenv
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *
import toml

# Loading and assigning variables
BOOTSTRAP_SERVERS = 'localhost:9092'
load_dotenv('../../.env')
app_config = toml.load('../../config_file.toml')
s3_bucket = app_config['aws']['s3_bucket']

ACCESS_KEY = os.getenv('ACCESS_KEY')
SECRET_KEY = os.getenv('SECRET_KEY')
```



```
# Building spark session with the jar files we downloaded earlier
spark = SparkSession.builder \
.appName("S3 access") \
.config("spark.jars", "../jars/hadoop-aws-3.3.3.jar,../jars/aws-java-sdk-bundle-1.12.451.jar") \
.getOrCreate()

# Enabling Spark to read s3 bucket via AWS access and secret access key
spark._jsc.hadoopConfiguration().set("spark.hadoop.fs.s3a.access.key", ACCESS_KEY)
spark._jsc.hadoopConfiguration().set("spark.hadoop.fs.s3a.secret.key", SECRET_KEY)

# Reading schema from s3 bucket
schema = spark.read.json(f's3a://{s3_bucket}/artifacts/bus_status_schema.json').schema

# Reading spark streaming from kafka topic in our docker container
df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", BOOTSTRAP_SERVERS) \
    .option("subscribe", "dbserver1.demo.bus_status") \
    .option("startingOffsets", "latest") \
    .load()
```

Spark Streaming Cont.



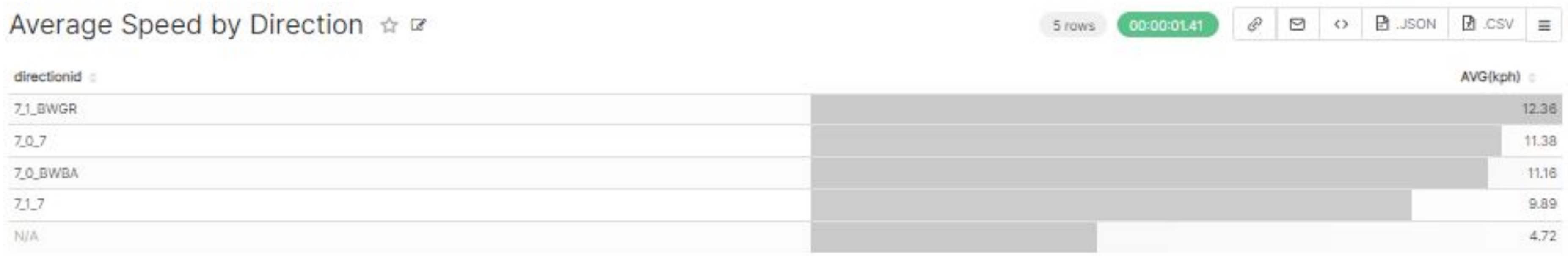
```
# Parsing the data from spark streaming to dataframe format
transform_df = df.select(col("value").cast("string")).alias("value").withColumn("jsonData",from_json(col("value"),schema)).select("jsonData.payload.after.*")

# Creating checkpoint location for hudi
checkpoint_location = f"s3a://{s3_bucket}/checkpoints/"

# Configuring hudi
table_name = 'bus_status'
hudi_options = {
    'hoodie.table.name': table_name,
    "hoodie.datasource.write.table.type": "COPY_ON_WRITE",
    'hoodie.datasource.write.recordkey.field': 'record_id',
    'hoodie.datasource.write.partitionpath.field': 'routeId',
    'hoodie.datasource.write.table.name': table_name,
    'hoodie.datasource.write.operation': 'upsert',
    'hoodie.datasource.write.precombine.field': 'event_time',
   'hoodie.upsert.shuffle.parallelism': 100,
    'hoodie.insert.shuffle.parallelism': 100
}

# Creating output s3 path
s3_path = f"s3a://{s3_bucket}/output/"

# Configuring hudi write
def write_batch(batch_df, batch_id):
    batch_df.write.format("org.apache.hudi") \
    .options(**hudi_options) \
    .mode("append") \
    .save(s3_path)

# Writing to s3 bucket
transform_df.writeStream.option("checkpointLocation", checkpoint_location).queryName("wcd-bus-streaming").foreachBatch(write_batch).start().awaitTermination()
```

Superset Dashboards

- Created a Superset container through Docker
- Connected to AWS Athena
- Passed a mapbox token into the Superset container as an environment variable to enable mapbox features in dashboard visuals
  - You need to create a mapbox account to retrieve the token
- Streamed on **September 20, 2023 (Wednesday)** between ~1:00pm to 4:59pm (EST)

Average Speed by Direction

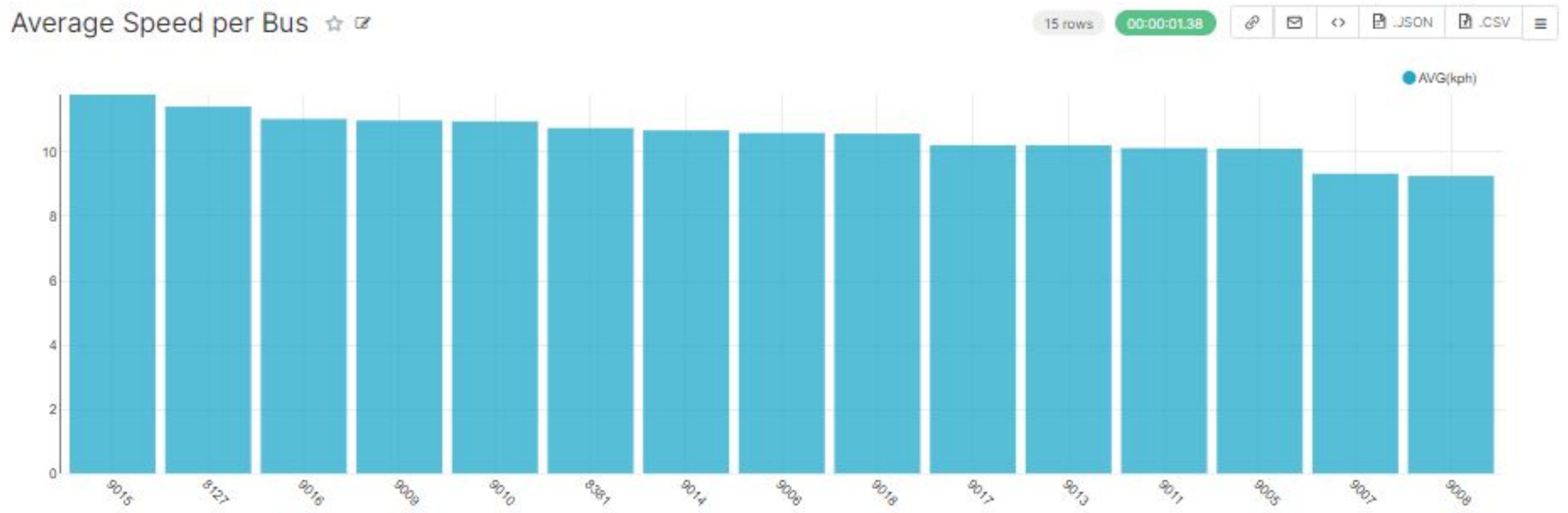
- Not counting “BW” directions, 7_0_7 is faster than 7_1_7 direction


Average Speed by Direction

directionid	AVG(kph)
7_1_BWGR	12.36
7_0_7	11.38
7_0_BWBA	11.16
7_1_7	9.89
N/A	4.72

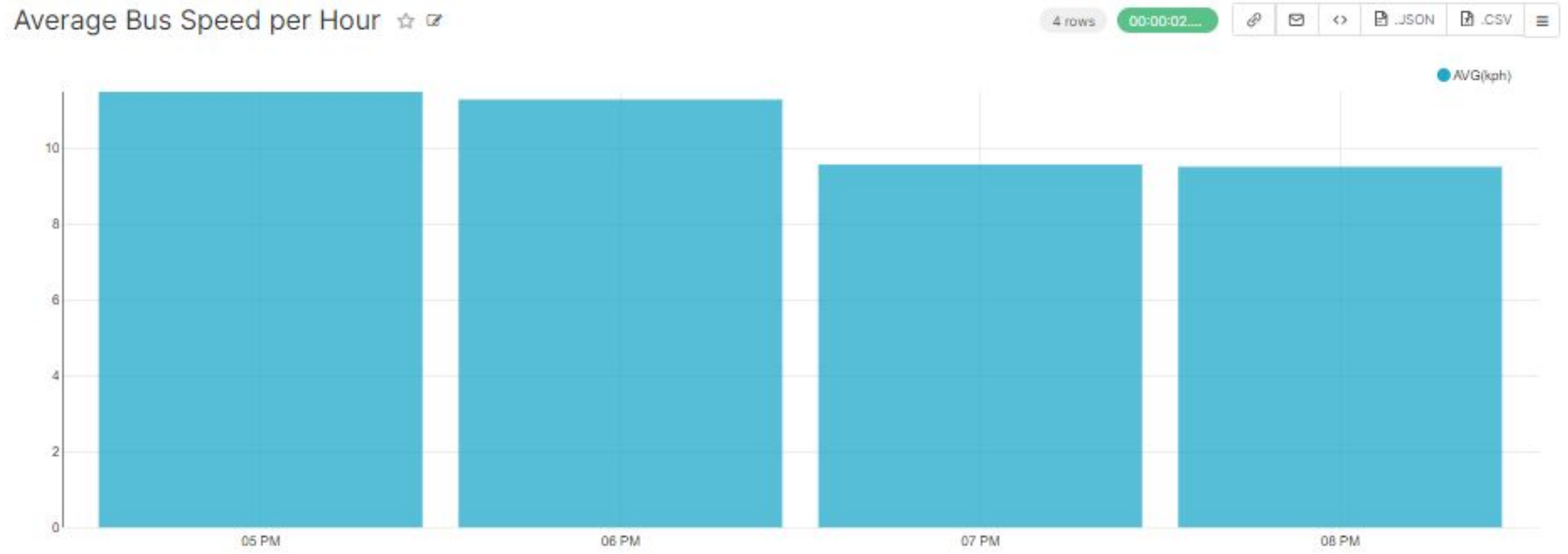
Average Speed per Bus

- 9015 is the fastest bus while 9008 is the slowest



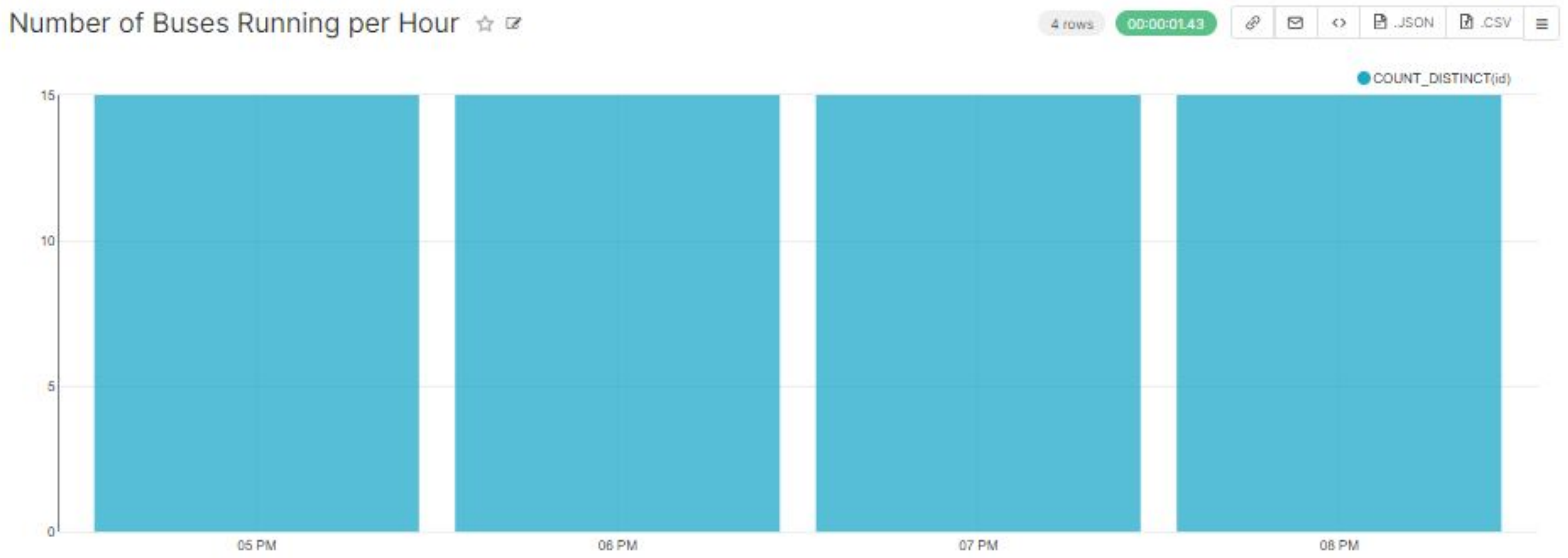
Average Bus Speed per Hour

- From 1-5pm, the average speed slows down. Could be due to the rush hour as people start getting off work from 3-5pm



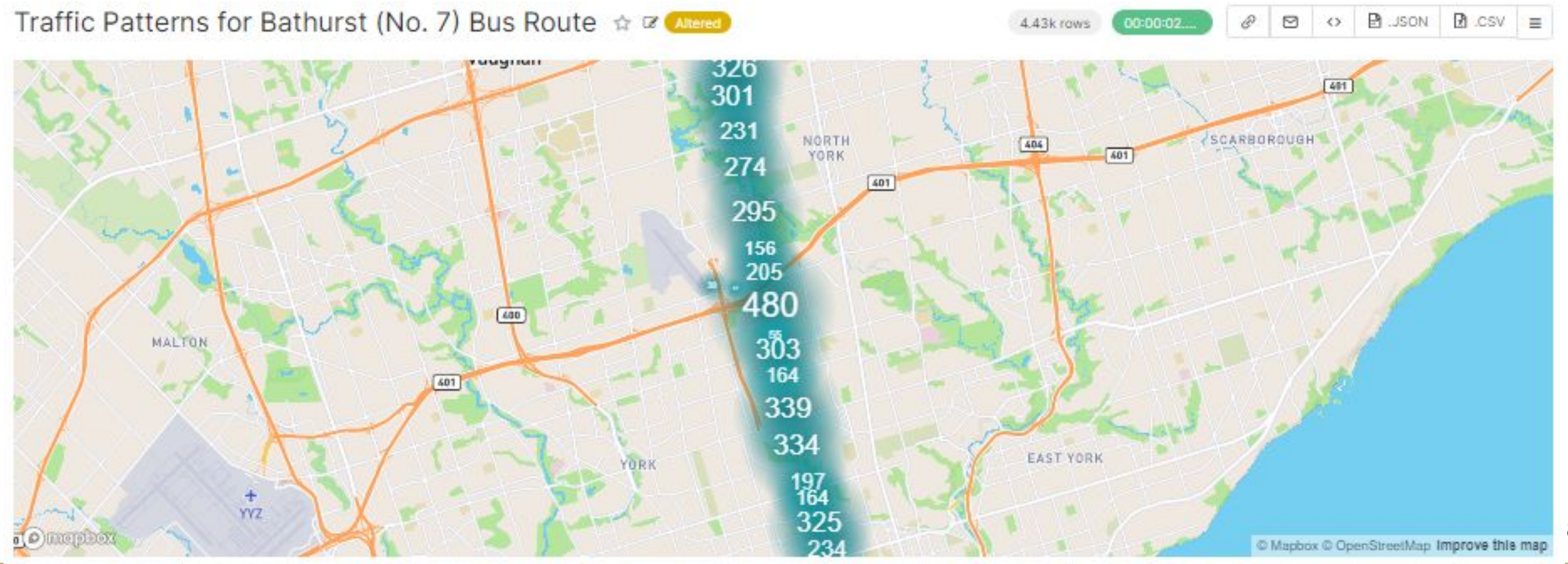
Number of Buses Running per Hour

- From 1-5pm, all the buses are in full service. May drop at night time



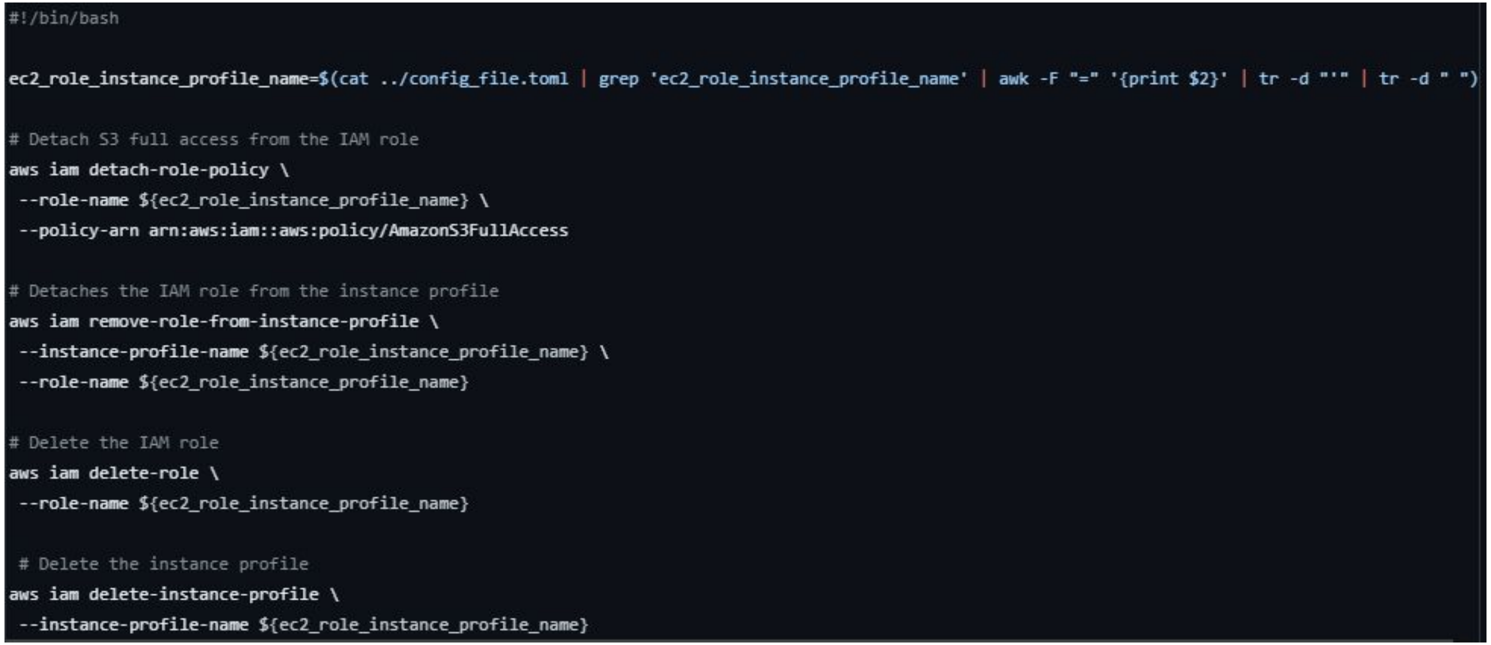
Traffic Patterns for Bathurst (No. 7) Bus Route

- Higher traffic near Bathurst and Highway 401. Probably caused by a high number of vehicles merging into and out of the highway



Remove Docker Container and AWS Artifacts

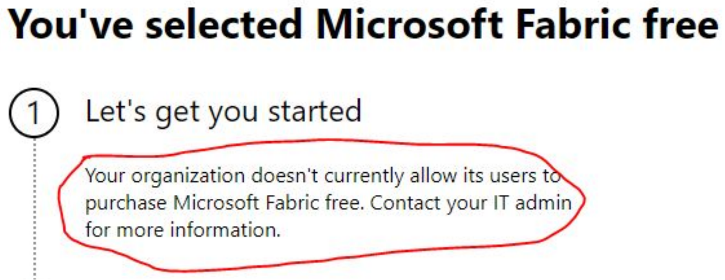
- Removes all the Docker Containers and AWS artifacts that were created
- Keeps the AWS account and EC2 instance clean



```
#!/bin/bash

ec2_role_instance_profile_name=$(cat ../config_file.toml | grep 'ec2_role_instance_profile_name' | awk -F "=" '{print $2}' | tr -d "'" | tr -d " ")

# Detach S3 full access from the IAM role
aws iam detach-role-policy \
 --role-name ${ec2_role_instance_profile_name} \
 --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

# Detaches the IAM role from the instance profile
aws iam remove-role-from-instance-profile \
 --instance-profile-name ${ec2_role_instance_profile_name} \
 --role-name ${ec2_role_instance_profile_name}

# Delete the IAM role
aws iam delete-role \
 --role-name ${ec2_role_instance_profile_name}

 # Delete the instance profile
aws iam delete-instance-profile \
 --instance-profile-name ${ec2_role_instance_profile_name}
```

Challenges

1. Dashboards
   a. Had to create an account in mapbox to generate key for mapbox visualizations
   b. Unable to find free tier for PowerBI
2. Hudi
   a. Sometimes received an error message from Hudi complaining record_id is null when running the pyspark_streaming.py function
   b. Fix was to re-create mysql docker container and error proof the docker container creation
3. Resources
   a. Compute and storage costs can add up if we want to collect a lot of data
   b. Removing duplicate rows for large amounts of streaming data will be computationally costly


You've selected Microsoft Fabric free

1 Let's get you started

Your organization doesn't currently allow its users to purchase Microsoft Fabric free. Contact your IT admin for more information.

Next Steps

1. Explore whether we can use shell script to automate the following in nifi:
    i. Import and use the template xml file
    ii. Start and stop processes
    iii. Set DBCP Connection Pool password
    iv. Enable the DBCP Connection Pool
2. Figure out how to use a toml or .env file to pass the mapbox token into the Superset container
    a. Currently there are formatting issues that make it difficult to pass the mapbox token in a config file
3. Stream the data for a longer time period to uncover more enriched insights
4. Figure out how to stream data to s3 with unique record_id rows

Thank You