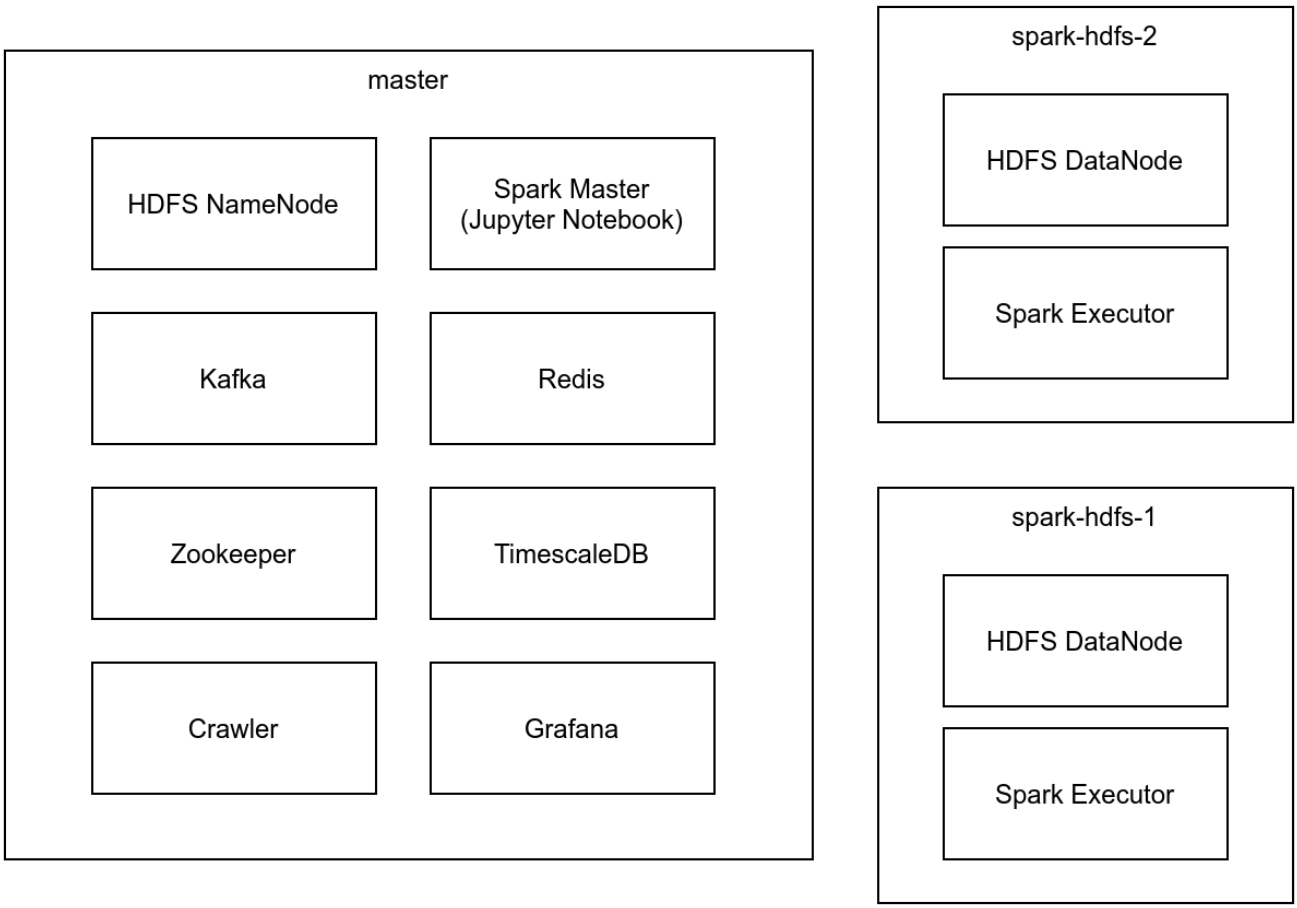
Hướng dẫn cụ thể triển khai hệ thống

Hệ thống sử dụng trong luận văn được cài đặt trên ba máy chủ ảo của Microsoft Azure, trong đó một máy chịu trách nhiệm chạy các tác vụ điều khiển và cơ sở dữ liệu, hai máy còn lại chạy tiến trình Spark Executor và HDFS DataNode. Chi tiết các tác vụ được trình bày trong hình sau:



Triển khai các tác vụ

Thiết lập các thông số cần thiết

Ở tệp tin ssh/constants.cfg, điều chỉnh các thông số cho phù hợp với môi trường triển khai hệ thống. Trong đó:

- `MASTER_ADDRESS`: Địa chỉ public của máy master.
- `MASTER_INTERNAL_ADDRESS`: Địa chỉ nội bộ của máy master. Các máy worker cần phân giải được địa chỉ này và truy cập được tất cả các port sử dụng địa chỉ này.
- `WORKER_NUM`: Số lượng máy worker.
- `WORKER_ADDRESS_<i>`: Địa chỉ máy worker thứ `i`.
- `WORKER_INTERNAL_ADDRESS_<i>`: Địa chỉ nội bộ của máy worker thứ `i`. Máy master cần phân giải được địa chỉ này và truy cập được tất cả các port sử dụng địa chỉ này.
- `SSH_USERNAME`: Tên tài khoản dùng để truy cập các máy chủ.
- `SSH_KEY_PATH`: Đường dẫn đến tập tin khoá bảo mật (private key) để truy cập đến các máy chủ. Trên các máy chủ cần có các tệp khóa công khai (public key) để xác nhận sự truy cập.

Xem thêm cách tạo khóa và truy cập máy chủ từ xa tại: Hướng dẫn cách truy cập các máy chủ tử xa sử dụng ssh và khóa.

Cuối cùng, copy tập tin ssh/constants.cfg lên các máy chủ:

```
./ssh/copy_to_all.sh ssh/constants.cfg .
```

Cài đặt các công cụ cần thiết

Java

Các máy chủ cần có Java với phiên bản từ `11.0.11` trở lên.

```
./ssh/copy_to_all.sh libs/jdk-11.0.11_linux-x64_bin.tar.gz libs/
./ssh/run_command_on_all.sh "
    cd libs && \
    tar -xf jdk-11.0.11_linux-x64_bin.tar.gz && \
    rm jdk-11.0.11_linux-x64_bin.tar.gz
"
```

Python 3 và pip3

Máy chủ master cần có Python 3 với phiên bản từ `3.6.9` trở lên. Chi tiết cài đặt có thể xem ở đây: Hướng dẫn cài đặt Python 3 và pip3 trên Ubuntu Linux

Docker

Máy master sử dụng Docker để chạy các tác vụ cơ sở dữ liệu và Grafana.

- Cài đặt `docker` cho Ubunbu: Hướng dẫn cài đặt Docker Engine trên Ubuntu.
- Thiết lập để việc sử dụng `docker` không cần phải có lệnh `sudo`: Quản lý Docker như một người dùng không phải root.
- Cài đặt `docker-compose`: Hướng dẫn cài đặt Docker Compose.

Triển khai cụm Spark Standalone

Cụm Spark Standalone bao gồm một Spark Master chạy ở máy master, và một Spark Executor chạy ở mỗi máy worker.

- Chạy Spark Master:

  ```
  ./ssh/copy_to_master.sh scripts/start_spark_master.sh .
  ./ssh/run_command_on_master.sh start_spark_master.sh
  ```

- Chạy các Spark Worker:

  ```
  ./ssh/copy_to_workers.sh scripts/start_spark_worker.sh .
  ./ssh/run_command_on_workers.sh start_spark_worker.sh
  ```

Triển khai Jupyter Notebook

Luận văn sử dụng Jupyter Notebook để làm công cụ chỉnh sửa mã nguồn.

- Cài đặt thư viện `jupyter`:

  ```
  ./ssh/run_command_on_master.sh "pip3 install jupyter"
  ```

- Cài đặt thư viện `findspark`:

```
./ssh/run_command_on_master.sh "pip3 install findspark"
```

- Chạy Jupyter Notebook:

```
./ssh/copy_to_master.sh scripts/start_jupyter_notebook.sh .
./ssh/run_command_on_master.sh start_jupyter_notebook.sh
```

- Nếu máy master không có public IP, cần phải chạy lệnh:

```
./ssh/forward_master_port_to_local.sh 8888
```

- Nhấn vào đường dẫn hiện lên để truy cập Notebook.

NOTE Đường dẫn để truy cập Notebook cần phải kèm theo token. Ví dụ: `http://localhost:8888?token=<token>`

Triển khai hệ thống quản lý tập tin phân tán HDFS

HDFS gồm một tác vụ NameNode trên máy master và một tác vụ DataNode trên mỗi máy worker.

Để triển khai HDFS trên các DataNode, máy chủ master cần phải truy cập được vào các máy chủ worker bằng SSH.

- Tạo ra một cặp khóa:

```
ssh-keygen -b 4096 -t rsa -f ssh/id_rsa -q -N ""
```

- Copy khóa bảo mật lên máy master:

```
./ssh/copy_to_master.sh "ssh/id_rsa ssh/id_rsa.pub" .ssh/
./ssh/run_command_on_master.sh "cat .ssh/id_rsa.pub >> .ssh/authorized_keys"
```

- Copy khóa công khai lên các máy worker và cho phép master truy cập:

```
./ssh/copy_to_workers.sh ssh/id_rsa.pub .ssh/
./ssh/run_command_on_workers.sh "cat .ssh/id_rsa.pub >> .ssh/authorized_keys"
```

- Config các máy chủ:

```
./ssh/copy_to_all.sh scripts/config_hdfs.sh .
./ssh/run_command_on_all.sh config_hdfs.sh
```

- Chạy HDFS:

```
./ssh/copy_to_master.sh scripts/start_hdfs.sh .
./ssh/run_command_on_master.sh start_hdfs.sh
```

- Dùng port-forwarding để điều hướng các yêu cầu đến master:

```
./ssh/forward_master_port_to_local.sh 9870
```

Sau đó truy cập http://localhost:9870 để vào trang quản lý của HDFS.

Triển khai Apache Kafka

Hệ thống sử dụng Apache Kafka như một trung gian truyền dữ liệu.

```
./ssh/copy_to_master.sh scripts/start_kafka.sh .
./ssh/run_command_on_master.sh start_kafka.sh
```

Chạy crawler

Hệ thống sử dụng NodeJS để viết một chương trình liên tục nhận dữ liệu từ trang http://api.metro.net/agencies/lametro/vehicles/, sau đó gửi vào trong Kafka.

```
./ssh/copy_to_master.sh source/crawler .
./ssh/copy_to_master.sh scripts/start_crawler.sh .
./ssh/run_command_on_master.sh start_crawler.sh
```

Cơ sở dữ liệu

Hệ thống sử dụng TimescaleDB và Redis chạy trên nền Docker để làm cơ sở dữ liệu phục vụ truy vấn.

```
./ssh/copy_to_master.sh docker/databases .
./ssh/run_command_on_master.sh databases/start.sh
```

Grafana

Hệ thống sử dụng Grafana để theo dõi trạng thái hoạt động của hệ thống, độ trễ, ...

Triển khai Grafana

- Chạy Grafana

  ```
  ./ssh/copy_to_master.sh docker/grafana .
  ./ssh/run_command_on_master.sh grafana/start.sh
  ```

- Sử dụng port-forwarding để chuyển hướng các yêu cầu đến port 3000 ở máy local đến máy master.

  ```
  ./ssh/forward_master_port_to_local.sh 3000
  ```

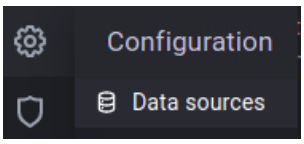
  Giữ cửa sổ lệnh này mở để truy cập vào Grafana ở local.

Thiết lập Grafana

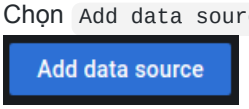
Truy cập http://localhost:3000, đăng nhập bằng tên tài khoản và mật khẩu `admin`.

Tạo nguồn dữ liệu

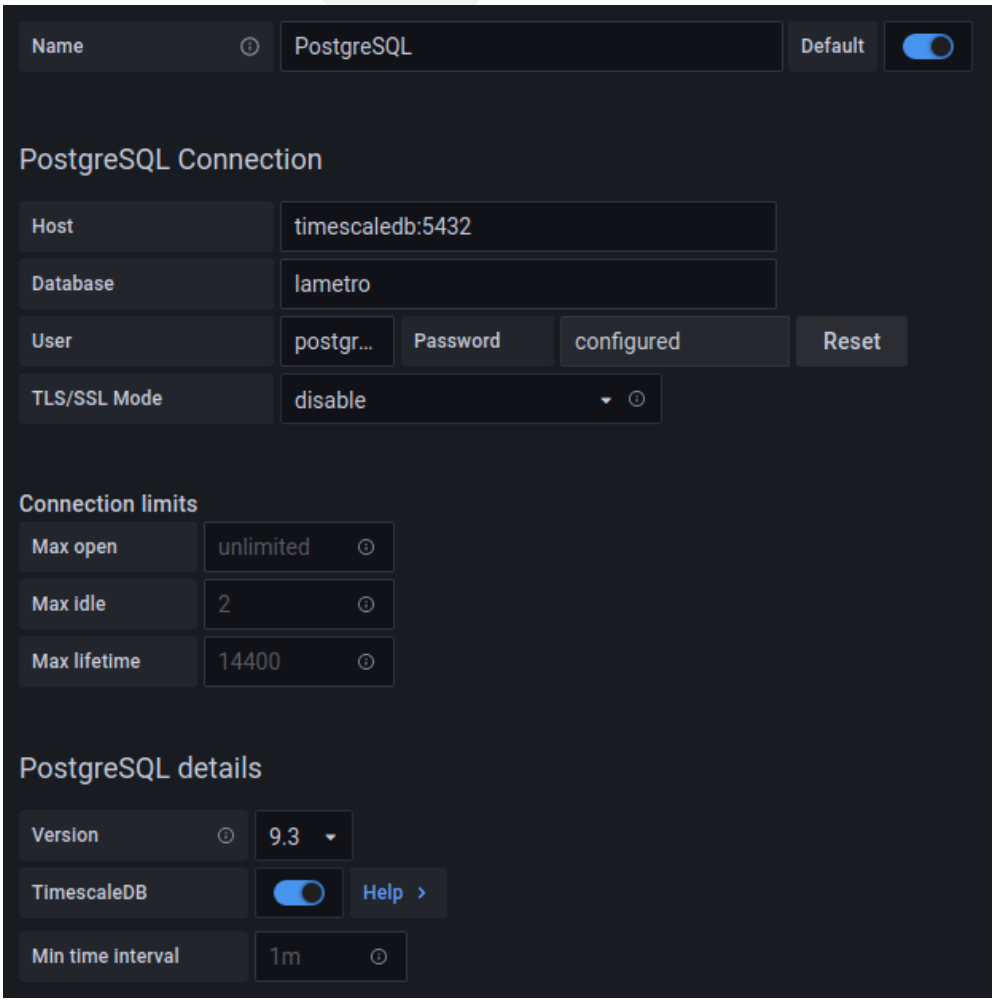
- Ở thanh công cụ bên trái màn hình, chọn biểu tượng bánh răng, sau đó chọn `Data sources`.

- Chọn `Add data source`.

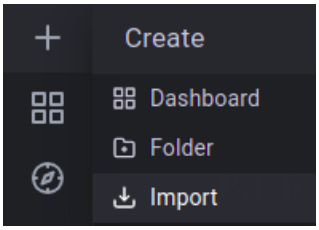
  

- Điền thông tin như hình với mục `password` là `8zr7E3SV` (được thiết lập trong docker/databases/docker-compose.yml), sau đó chọn `Save & Test`.

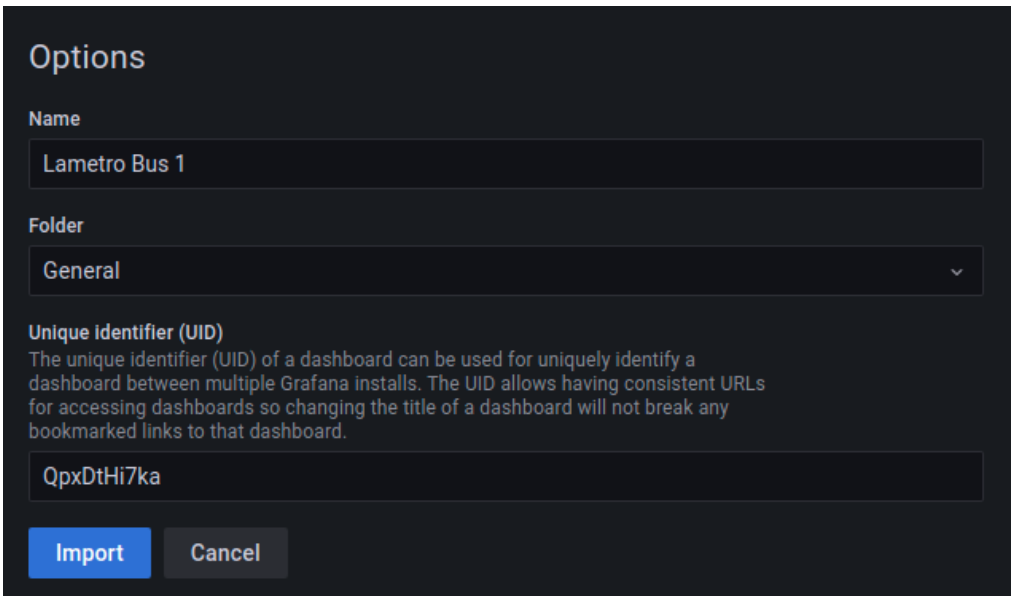


Tạo trang quản lý:

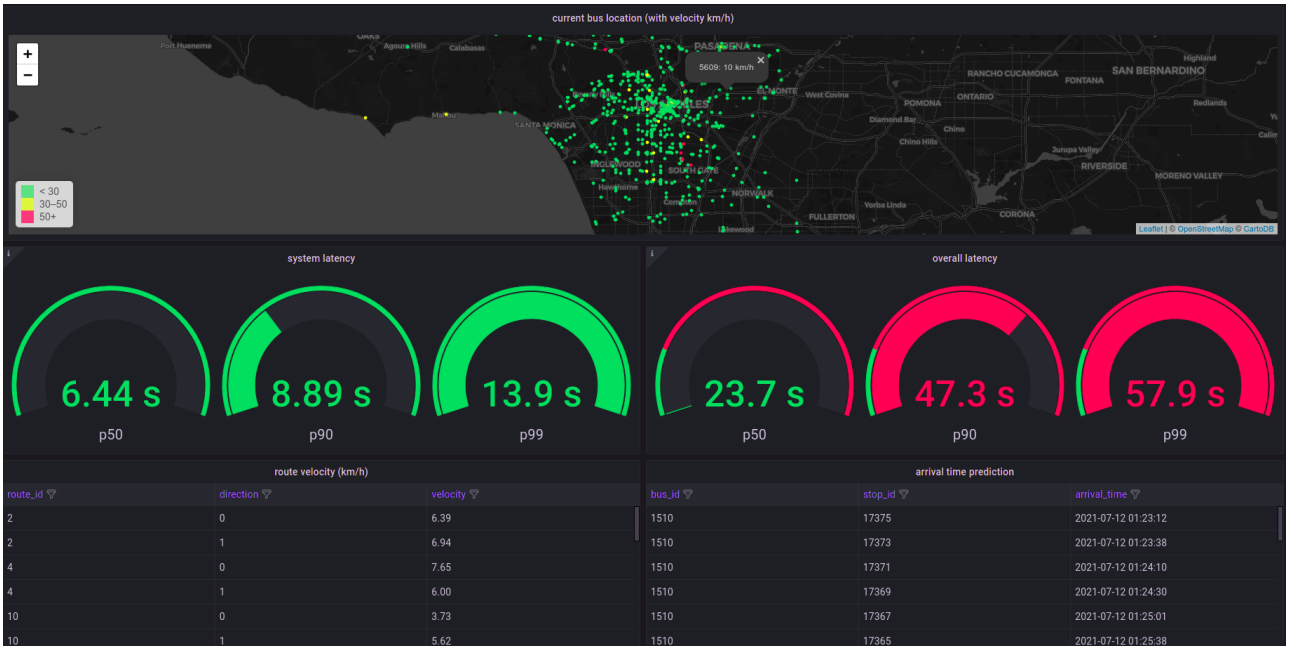
- Ở thanh công cụ bên trái màn hình, chọn biểu tượng dấu cộng, sau đó chọn `Import`.



- Chọn `Upload JSON file` và chọn tập tin nằm ở docker/grafana/dashboard.json. Nhấn `Import` để tạo trang quản lý.



- Giao diện trang quản lý sẽ tương tự như sau:



Chạy mã nguồn

Các thiết lập cần thiết

- Sao chép dữ liệu lên HDFS:

```
./ssh/copy_to_master.sh source/data.zip .
./ssh/copy_to_master.sh scripts/copy_data_to_hdfs.sh .
./ssh/run_command_on_master.sh copy_data_to_hdfs.sh
```

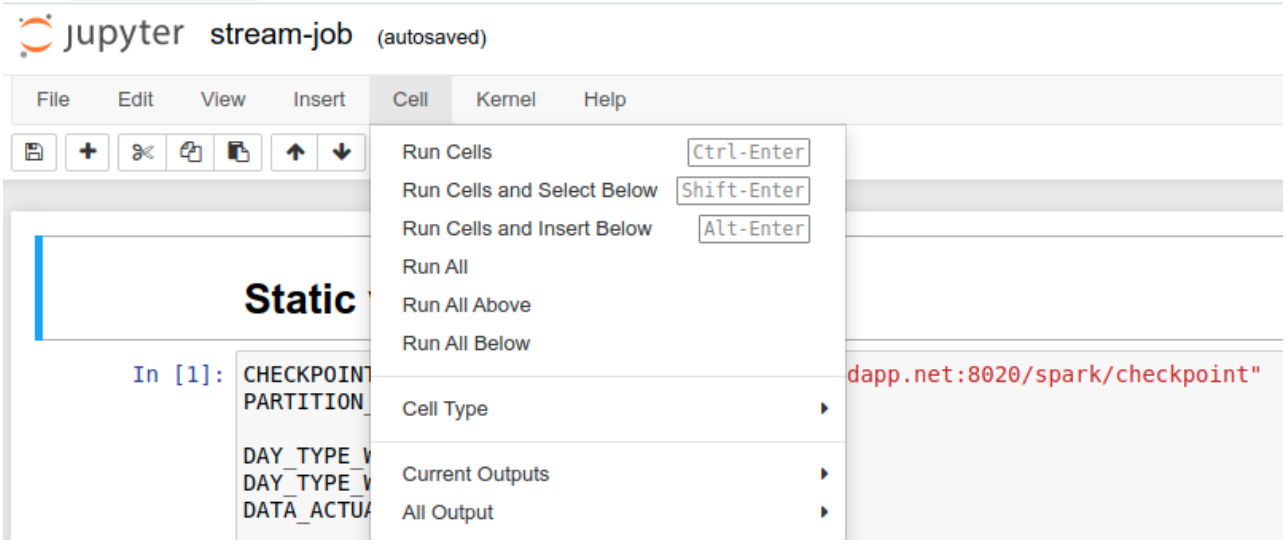
- Sao chép thư mục chứa mã nguồn lên master:

```
./ssh/copy_to_master.sh source/notebooks .
```

- Sao chép các thư viện cần thiết cho việc chạy mã nguồn lên master:

```
./ssh/copy_to_master.sh libs/third-party-jars libs/spark-3.1.1-bin-hadoop3.2/
```

- Truy cập Jupyter Notebook, sau đó mở tập tin `stream-job.ipynb`. Sau đó, ở thanh công cụ phía trên màn hình, chọn `Cell > Run All`.



NOTE Cần điều chỉnh hostname của các biến tĩnh trong tập tin cho phù hợp với môi trường thực thi trước khi chạy.

```
CHECKPOINT_DIR = "hdfs://master.internal.cloudapp.net:8020/spark/checkpoint"
PARTITION_NUMBER = 4

DAY_TYPE_WEEKDAY = 0
DAY_TYPE_WEEKEND = 1
DATA_ACTUAL_TIMEZONE = "America/Los_Angeles"

BOOTSTRAP_SERVER = "master.internal.cloudapp.net:9092"
TOPIC = "buses-location"

POSTGRES_URL = "jdbc:postgresql://master.internal.cloudapp.net:5432/lametro"
POSTGRES_TABLE_BUS_VELOCITY = "bus_velocity"
POSTGRES_TABLE_BUS_ARRIVAL = "bus_arrival"
POSTGRES_USERNAME = "postgres"
POSTGRES_PASSWORD = "8zr7E3SV"

REDIS_HOST = "master.internal.cloudapp.net"
REDIS_PORT = "6379"
REDIS_PASSWORD = "8zr7E3SV"

STATIC_DATA_DIR = "hdfs://master.internal.cloudapp.net:8020/ola/static_data/"
HISTORICAL_DATA_DIR = "hdfs://master.internal.cloudapp.net:8020/ola/historical_data/"
AGGREGATED_DATA_DIR = "hdfs://master.internal.cloudapp.net:8020/ola/aggregated_data/"
TEMP_DIR = "hdfs://master.internal.cloudapp.net:8020/temp"
```
