Project 2 – 2023 – Big Data – Flink DataStream

Author: Daniel Zdancewicz indeks 145317

Description

Processing stream data using Flink DataStream API with Kafka as a message broker.

Dataset

Dataset used for this exercise is a modified Stock Data Dataset from Kaggle.

Example cloud environment:

- Google Cloud Platform
- Google Cloud Storage
- Google Cloud Dataproc
- setup-cluster.sh - Script file to set up the cluster on Google Cloud Platform using Google Cloud Dataproc.

```
gcloud dataproc clusters create "${CLUSTER_NAME}" \
  --enable-component-gateway --bucket "${BUCKET_NAME}" \
  --region "${REGION}" --subnet default --zone "${ZONE}" \
  --master-machine-type n1-standard-4 --master-boot-disk-size 50 \
  --num-workers 2 \
  --worker-machine-type n1-standard-2 --worker-boot-disk-size 50 \
  --image-version 2.0-debian10 \
  --project "${PROJECT_ID}" --max-age=3h \
  --optional-components=ZEPPELIN,DOCKER,ZOOKEEPER \
  --metadata "run-on-master=true" \
  --initialization-actions \
  gs://goog-dataproc-initialization-actions-"${REGION}"/kafka/kafka.sh
```

Variables

- `${CLUSTER_NAME}` is the name of the cluster
- `${BUCKET_NAME}` is the name of the bucket
- `${REGION}` is the region where the cluster will be created
- `${ZONE}` is the zone where the cluster will be created
- `${PROJECT_ID}` is the project id

Flags

- `-enable-component-gateway` is the component gateway to access the cluster
- `-bucket ${BUCKET_NAME}` is the bucket to store the data
- `-region ${REGION}` is the region where the cluster will be created
- `-subnet default` is the subnet of the cluster
- `-zone ${ZONE}` is the zone where the cluster will be created
- `-master-machine-type n1-standard-4` is the machine type of the master node
- `-num-workers 2` is the number of workers
- `-master-boot-disk-size 50` is the boot disk size of the master node
- `-worker-machine-type n1-standard-2` is the machine type of the worker node
- `-worker-boot-disk-size 50` is the boot disk size of the worker node
- `-image-version 2.0-debian10` is the image version of the cluster
- `-project ${PROJECT_ID}` is the project id
- `-max-age=3h` is the maximum age of the cluster
- `-optional-components=ZEPPELIN,DOCKER,ZOOKEEPER` is the optional components to install on the cluster
  - `ZEPPELIN` is the Zeppelin notebook which is used to run the Flink DataStream application
  - `DOCKER` is the Docker to run the Kafka
  - `ZOOKEEPER` is the Zookeeper to run the Kafka
- `-metadata "run-on-master=true"` is the metadata to run Kafka on the master node

- `-initialization-actions` is the initialization script for Kafka

Kafka broker

```
gs://goog-dataproc-initialization-actions-${REGION}/kafka/kafka.sh
```

Scripts

All scripts are located in the scripts directory. ALl scripts (except the setup-cluster.sh) should be used from the cwd (to ensure the loading of all required variables located in setup-variables.sh) of the project like so:

```
source ./scripts/{script_name}
```

Parameters

- Cloud parameters
  - `CLUSTER_NAME` – Name of the cluster
  - `BUCKET_NAME` – Name of the bucket
  - `HADOOP_CLASSPATH` – hadoop classpath
  - `HADOOP_CONF_DIR` – hadoop configuration directory
  - `INPUT_FILE_PATH` – path to the input file of dataset
  - `INPUT_DIRECTORY_PATH` – path to the input directory of dataset
- Kafka parameters
  - `KAFKA_PRODUCER_SLEEP_TIME` – kafka producer sleep time
  - `KAFKA_CONTENT_TOPIC_NAME` – kafka content topic name
  - `KAFKA_ANOMALY_TOPIC_NAME` – kafka anomaly topic name
  - `KAFKA_BOOTSTRAP_SERVERS` – kafka bootstrap servers
  - `KAFKA_GROUP_ID` – kafka group id
- JDBC parameters

    - `JDBC_URL` – url to the database
    - `JDBC_USERNAME` – username to the database
    - `JDBC_PASSWORD` – password to the database
- Flink parameters
    - `FLINK_DIRECTORY` – flink directory
- Anomaly parameters
    - `ANOMALY_STOCK_DAYS_RANG` – anomaly stock days range
    - `ANOMALY_STOCK_PERCENT_FLUCTUATION` – anomaly stock percent fluctuation
- `PROCESSING_TYPE` – “historical” | “realtime” – content processing type

Setup scripts:

- setup-variables.sh is used to set up the variables.
- setup-environment.sh is used to set up the environment.
- setup-cluster.sh is used to set up the cluster.
- setup-database.sql is used to set up the database.
- setup-database-unsafe.sh is used to set up the database for preview purposes.
- setup-bucket.sh is used to set up the bucket with the data.

Run scripts:

- run-consumer-database.sh is used to run the consumer to store the data in the database.
- run-consumer-kafka.sh is used to run the consumer to print the data.
- run-producer-kafka.sh is used to run the producer to send the data.
- run-processor-kafka.sh is used to run the processor to process the data.

Step-by-step setup

1. Create google cloud cluster on google cloud console using setup-cluster.sh.
2. Create necessary bucket with the name as the `${BUCKET_NAME}` variable defined in the setup-variables.sh.

3. Create necessary mysql instance with the google cloud (Sql menu / create instance). Or skip and use unsafe current host with setup-database-unsafe.sh and `sudo mysql -u root < ./scripts/setup-database.sql`, goto step 6.
4. Create connection to the cluster master vm.
5. Connect sql instance to the cluster master vm.
6. Run database setup query `mysql -h sql_ip -p < ./scripts/setup-database.sql`.
7. Access ssh of master-machine in the created cluster (should be in the dataproc menu under the cluster view in the virtual-machine tab named as `${CLUSTER_NAME}-m`).
8. Download this repository onto the machine.
9. Access the repository's directory. ex. `cd ~/pbd-2023-flink-streams`.
10. Run setup bucket script to download dataset to the bucket using setup-bucket.sh.
11. Run setup environment script using setup-environment.sh to set up dataset download from the bucket to the master, download all updates to the machine, install required sbt/scala, install required flink version, build application jars, create kafka topics, and create database.

After all that you should be able to run all the run scripts (remember to use them from the cwd of the project to ensure all variables are exported).

To test the set-up environment, I boot up the kafka processor, consumer database, consumer kafka, producer kafka.

```
source ./scripts/run-consumer-database.sh
```

```
source ./scripts/run-consumer-kafka.sh
```

```
source ./scripts/run-processor-kafka.sh
```

```
source ./scripts/run-producer-kafka.sh
```