Flink Applications Powered by Java on a locally running Apache Flink Cluster in Docker

Discover how Apache Flink® can transform your data pipelines! Explore hands-on examples of Flink applications using the DataStream API and Table API in Java. You'll see how these technologies integrate seamlessly with AWS Services, Apache Kafka, and Apache Iceberg.

Curious about the differences between the DataStream API and Table API? Click here to learn more and find the best fit for your next project.

Table of Contents

1.0 Power up the Apache Flink Docker containers

> **Prerequisite**
>
> Before you can run `scripts/run-flink-locally.sh` Bash script, you need to install the `aws2-wrap` utility. If you have a Mac machine, run this command from your Terminal:
>
> ```
> brew install aws2-wrap
> ```
>
> If you do not, make sure you have Python3.x installed on your machine, and run this command from your Terminal:
>
> ```
> pip install aws2-wrap
> ```

This section guides you through the local setup (on one machine but in separate containers) of the Apache Flink cluster in Session mode using Docker containers with support for Apache Iceberg. Run the `bash` script below to start the Apache Flink cluster in Session Mode on your machine:

```
./deploy-flink.sh <on | off> --profile=<AWS_SSO_PROFILE_NAME>
                             --chip=<amd64 | arm64>
                             --flink-language=java
```

Argument placeholder	Replace with

Argument placeholder	Replace with
`<DOCKER_SWITCH>`	`on` to start up your very own local Apache Cluster running in Docker containers, otherwise `off` to stop the Docker containers.
`<AWS_SSO_PROFILE_NAME>`	your AWS SSO profile name for your AWS infrastructue that host your AWS Secrets Manager.
`<CHIP>`	if you are running on a Mac with M1, M2, or M3 chip, use `arm64`. Otherwise, use `amd64`.
`<FLINK_LANGUAGE>`	`java` to specify Java is the language base of the Flink Apps you plan on running. Otherwise, specifiy `python` if the language base of the Flink Apps are Python.

To learn more about this script, click here.

2.0 Discover What You Can Do with These Flink Apps

To access the Flink JobManager (`apache_flink-kickstarter-jobmanager-1`) container, open the interactive shell by running:

```
docker exec -it -w /opt/flink/java_apps/app/build/libs apache_flink-
kickstarter-jobmanager-1 /bin/bash
```

This command drops you right into the container, giving you full control to execute commands, explore the file system, or handle any tasks directly.

Finally, to launch one of the **pre-complied** Flink applications, choose your app and use the corresponding `flink run` command listed below. Let's have some fun with Flink!

2.1 Avro formatted data

Flink App	Flink Run Command
`AvroDataGeneratorApp`	`flink run --class kickstarter.AvroDataGeneratorApp apache_flink-kickstarter-dev-SNAPSHOT.jar --service-account-user <SERVICE_ACCOUNT_USER>`
`AvroFlightConsolidatorApp`	`flink run --class kickstarter.AvroFlightConsolidatorApp apache_flink-kickstarter-dev-SNAPSHOT.jar --service-account-user <SERVICE_ACCOUNT_USER>`
`AvroFlyerStatsApp`	`flink run --class kickstarter.AvroFlyerStatsApp apache_flink-kickstarter-dev-SNAPSHOT.jar --service-account-user <SERVICE_ACCOUNT_USER>`

Argument placeholder	Replace with

> | Argument placeholder | Replace with |
> |---|---|
> | `<SERVICE_ACCOUNT_USER>` | specify the name of the service account user, used in the the AWS Secrets and Parameter Store Path name. |

2.1.1 Avro Java Classes Special Consideration

Whenever any of the Flink Apps `Avro models` need to be updated, the `avro-tools-1.12.0.jar` must be used afterwards to generate the respective Java class. This is necessary to ensure that the Avro schema is in sync with the Java class. To generate the Java class, run the following command from the `apache_flink-kickstarter-jobmanager-1` Docker container:

```
java -jar /path/to/avro-tools-1.12.0.jar compile -string schema
app/src/main/java/kickstarter/model/avro/AirlineAvroData.avsc .

java -jar /path/to/avro-tools-1.12.0.jar compile -string schema
app/src/main/java/kickstarter/model/avro/FlightAvroData.avsc .

java -jar /path/to/avro-tools-1.12.0.jar compile -string schema
app/src/main/java/kickstarter/model/avro/FlyerStatsAvroData.avsc .
```

Then copy the generated Java class to the `app/src/main/java/kickstarter/model/` directory:

```
cp kickstarter/model/AirlineAvroData.java
app/src/main/java/kickstarter/model/
```

> You can download the `avro-tools-1.12.0.jar` here.

2.2 JSON formatted data

Flink App	Flink Run Command
`JsonDataGeneratorApp`	`flink run --class kickstarter.JsonDataGeneratorApp apache_flink-kickstarter-dev-SNAPSHOT.jar --service-account-user <SERVICE_ACCOUNT_USER>`
`JsonFlightConsolidatorApp`	`flink run --class kickstarter.JsonFlightConsolidatorApp apache_flink-kickstarter-dev-SNAPSHOT.jar --service-account-user <SERVICE_ACCOUNT_USER>`
`JsonFlyerStatsApp`	`flink run --class kickstarter.JsonFlyerStatsApp apache_flink-kickstarter-dev-SNAPSHOT.jar --service-account-user <SERVICE_ACCOUNT_USER>`

> | Argument placeholder | Replace with |
> |---|---|

> | Argument placeholder | Replace with |
> |---|---|
> | `<SERVICE_ACCOUNT_USER>` | specify the name of the service account user, used in the the AWS Secrets and Parameter Store Path name. |