Apache Flink Installation Tutorial

Jan 2020

This tutorial covers installation local cluster mode of Apache Flink and running two example projects:

1. **A java streaming project connected to Apache Kafka:**

   We want to send a stream of sentences and get the count of each word in the sentences in the specified time windows.

2. **A batch processing project example with Python:**

   Again, a word count example, but not a streaming one. We want to give some text files and get the count of each word existing in the files.

Build and Run a Flink Java Project Connected to Kafka

In order to build, compile and run your Java code in Flink, you should use Maven or Gradle. We recommend it to use Maven because most of the official documentation on the Flink website is based on Maven.

Initially, we will discuss the basics of creating and writing a java program and then we will explain how to install, run and connect the Apache Kafka to your Flink project as the source (steps 6-8 in the following).

Prerequisites:
You need to install Java on your Linux and set the required environment variables (e.g. "JAVA_HOME").

1. Install Apache Maven

Download the binary archive of Maven from its website: https://maven.apache.org

Then, extract it in an appropriate place. Finally, add it to "PATH" environment variable as follow:

```
export PATH=/[path of extracted folder]/apache-maven-3.6.3/bin:$PATH
```

2. Create Project

Create a new folder (e.g. "flink_java_example") then open a terminal in it and run the following commands:

```
mvn archetype:generate                                    \
      -DarchetypeGroupId=org.apache.flink                   \
      -DarchetypeArtifactId=flink-quickstart-java           \
      -DarchetypeVersion=1.9.0
```

After that, it will ask you the names of your project such as "groupid", "artifactid" etc. You need to just give them any names you want.
Finally, it will construct your project with two template Java files named "BatchJob.java" and "StreamingJob.java". You can put your source codes instead.

3. Import Project in an IDE (optional)

Flink creators recommend importing the project in an IDE. They introduced IntelliJ IDEA and Eclipse however based on our experience and existing documents, IntelliJ IDEA is the best.
To perform it, open IntelliJ IDEA and import the project folder (where the “src” folder and “pom.xml” file).
First, run the code then go to “Edit Configuration” and check the “Include dependencies with “Provided” scope” checkbox.
We can create an test our projects here locally but we will explain how to run it on Flink’s cluster.

4. Write your Java Code

Here are the java code and configurations you need.

```java
import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.common.functions.ReduceFunction;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.api.windowing.time.Time;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer;
import org.apache.flink.util.Collector;

import java.util.Properties;


public class StreamingJob {

 public static void main(String[] args) throws Exception {
 // set up the streaming execution environment
 final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
 Properties properties = new Properties();
 properties.setProperty("bootstrap.servers", "localhost:9092");
 properties.setProperty("zookeeper.connect", "localhost:2181");
 properties.setProperty("group.id", "test");
 DataStream<String> stream = env.addSource(new FlinkKafkaConsumer<>("news", new
SimpleStringSchema(), properties));
 DataStream<WordWithCount> windowCounts = stream
 .flatMap(new FlatMapFunction<String, WordWithCount>() {
 @Override
 public void flatMap(String value, Collector<WordWithCount> out) {
 for (String word : value.split("\\s")) {
 out.collect(new WordWithCount(word, 1L));
 }
 }
 })
 .keyBy("word")
 .timeWindow(Time.seconds(5))
 .reduce(new ReduceFunction<WordWithCount>() {
 @Override
```

```
                    public WordWithCount reduce(WordWithCount a, WordWithCount b) {
                        return new WordWithCount(a.word, a.count + b.count);
                    }
                });
        // print the results with a single thread, rather than in parallel
        windowCounts.print().setParallelism(1);
        // execute program
        env.execute("Flink Streaming Java API Skeleton");
    }

    // Data type for words with count
    public static class WordWithCount {

        public String word;
        public long count;

        public WordWithCount() {
        }

        public WordWithCount(String word, long count) {
            this.word = word;
            this.count = count;
        }

        @Override
        public String toString() {
            return word + " : " + count;
        }
    }
}
```

5. Download Apache Kafka

Download the latest binary version of Kafka from its official website then extract it.

    https://kafka.apache.org/downloads

6. Download Flink

Download the latest binary version of Flink from its official website then extract it.

https://flink.apache.org/downloads.html

7. Start Apache Kafka

We are going to start Kafka as the data producer.

Go to kafka folder, open terminal and start a ZooKeeper server:

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

Now run another instance of the terminal in the Kafka folder and start the Kafka server:

```
bin/kafka-server-start.sh config/server.properties
```

You should create a topic on another terminal again. If you name it "News", type this command:

```
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --
replication-factor 1 --partitions 1 --topic News
```

You can see the topic by this command:

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

The final step here is running the producer hence open another terminal and type:

```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic News
```

Then you can type your messages, line by line in the terminal, these messages will be received by your Flink program.

8. Run on Flink's cluster

Before running the Java code, you should add a dependency in the "pom.xml" file so that you can import and use Kafka library in the code, and then build the project. The dependency is:

```
<dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kafka_2.11</artifactId>
      <version>1.9.0</version>
</dependency>
```

Change the version numbers as you want to use in the above dependency. Use the following webpage on the official Flink website:
https://ci.apache.org/projects/flink/flink-docs-stable/dev/connectors/kafka.html

Open a terminal in Flink folder and start the job with the following command:

```
./bin/start-cluster.sh
```

You need to run the JAR file created in step 4, copy the path of JAR file and use it in the following command:

```
./bin/flink run /<your JAR File path>/<your JAR File name>.jar
```

You can track your job status on the Flink web UI available on localhost:8081
In order to stop the job run this command:

```
./bin/stop-cluster.sh
```

9. Build Project

Go to your project folder i.e. where the "src" folder and "pom.xml" file are located then open terminal in there and run the "`mvn clean package`" command. A JAR file will be created "target" folder with the following structure: "<artifact-id>-<version>.jar"

10. Get the Results

The result of the program (the "print" function in the mentioned java code) will be saved in the following file path: "`<your flink folder>/log/*taskexcutor*.out`". The "*"s mean wildcard.

Build and Run a Flink Python Project (Batch Word_Count)

The only thing you need is to run your Python code with this command:

```
./bin/flink run -py ./<your python code>.py
```