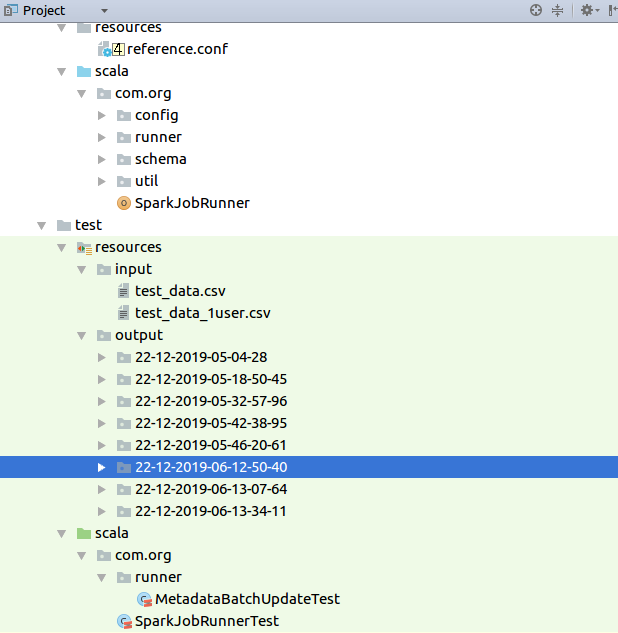
**Solution**

This is the Intellij scala maven project, The structure given below,



Main Code is in src/main/scala, Configuration file is in resource folder.

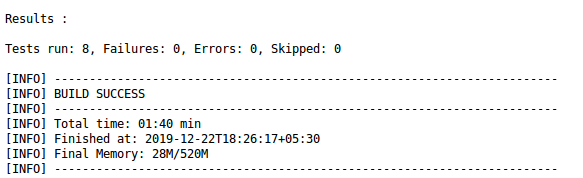
Unit tests are in src/test/scala, Input and Output folders for tests are in src/test/resources/ folder.

Entry point is **SparkJobRunner** class, It invokes **JobResolverFactory** factory class to resolve between stream and batch mode.

Batch mode job is in **MetadataBatchUpdate** class and stream mode job is in **MetadataStreamUpdate** class.

All the configurations are in **reference.conf** file which will have project related configurations of Spark, Kafka, Utilitites etc

Unit test output given below,



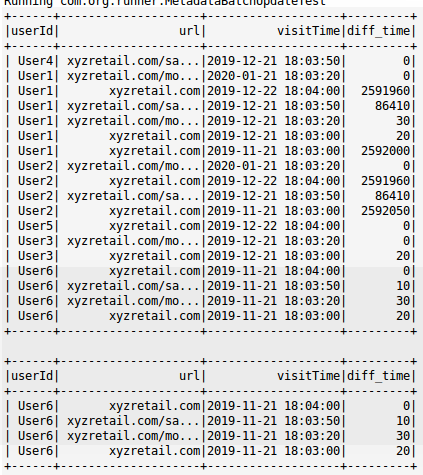
Output of test executions are stored in src/test/resources/output folder.

The build of the jar is located in target folder, Using the build we run the jobs in the production scenario with the command given below,

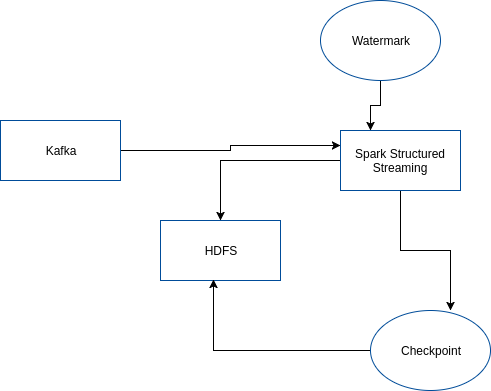
**spark-submit –class com.org.SparkJobRunner –master yarn –deploy-mode cluster --executor-memory 18G --num-executors 29 –executor-cores 5 user-metadata-1.0-SNAPSHOT.jar true hdfs:///home/app/input/ hdfs:///home/app/output/**

Here Number of executors , Memory and cores are choosen as a trade off between fat (1 Executor per node) and tiny (1 per core) executors.

Sample Output from Unit Tests for multiple and single user Id’s,



**STREAMING LIVE SOLUTION**



**MetadataStreamUpdate** class has the logic of Ingesting data from Spark , Writing the dstreams in the **appended** mode to hdfs and then updating the results of the current state using **Structured Streaming** feature.

Advantages

1. Reliable and handle late data, Watermark etc

2. Checkpoint feature

3. Continous updation of results, No overwrite.

Disadvantages

1. Structured streaming are pretty new and slight complex.