免費電子書

學習

apache-flink

Free unaffiliated eBook created from
Stack Overflow contributors.

#apache-flink

1: apache-flink

apache-flink。

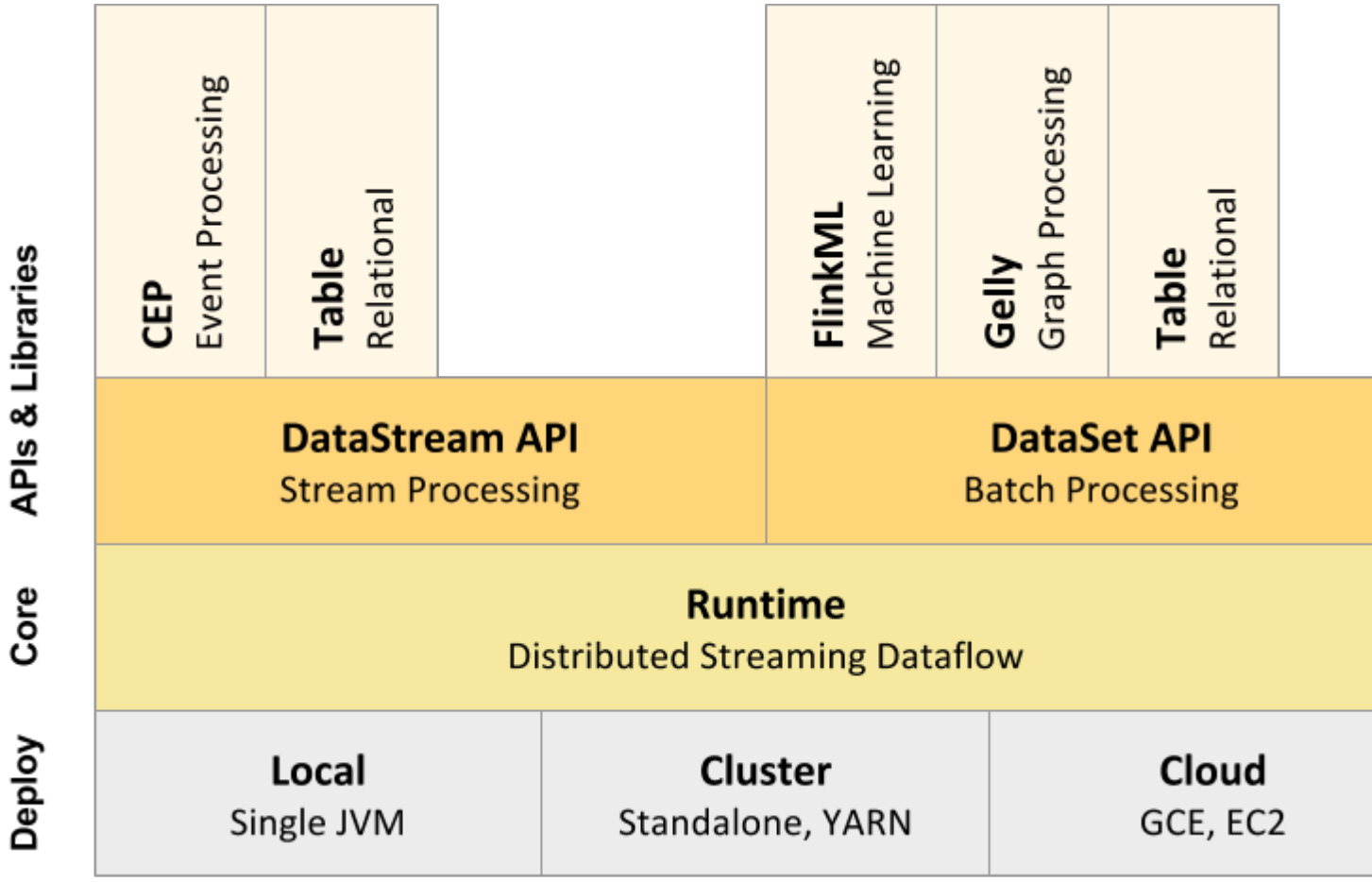
apache-flink。 apache-flink。

Examples

Flink

Apache HadoopApache Spark Apache Flink。 FlinkJavaScalaJavaPython API。

- UNIXLinuxMac OS XCygwin;
- Java 6.X;
- [] Maven 3.0.4。



Apache Flink **Hadoop MapReduce** 。 MapReduce。 Hadoop。

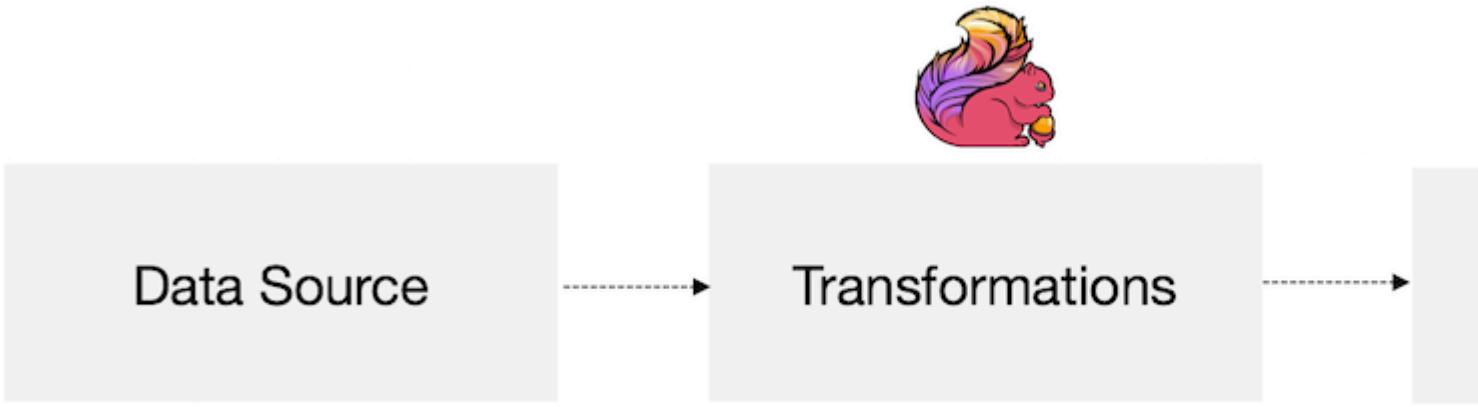
`ExecutionEnvironmentExecutionEnvironment`。 。

1. *JVM* FlinkJavaIDEFlink。 maven。
2. FlinkIDEFlink。 。
3. Flink。 。 mportant__`2.11`*scala* 。

Flink。 API

- **DataStream API** 。
- **DataSet API** 。
- **API** SQLSpark。

Flinks。



Flink

- Flink
- Flink
- Flink

/ HDFS。 。

0. java 6`JAVA_HOME`。
1. flink

   ```
   wget flink-XXXX.tar.gz
   ```

   Hadoophadoop 1。 scalamaven。
2. flink

```
tar xzvf flink-XXXX.tar.gz
./flink/bin/start-local.sh
```

Flink。 flink`flink/log/http://localhost:8081`flink jobManager。

3. flink

```
./flink/bin/stop-local.sh
```

Flink

IDEflinkEclipseIntellij IDEApreffered `flink-java` / `flink-scalaflink-clients` 20162。 MavenSBT JARSscala。

- **Maven**

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-java</artifactId>
    <version>1.1.4</version>
</dependency>

<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-clients_2.11</artifactId>
    <version>1.1.4</version>
</dependency>
```

- **SBT**=“”

```
version := "1.0"

scalaVersion := "2.11.8"

libraryDependencies ++= Seq(
  "org.apache.flink" %% "flink-scala" % "1.2.0",
  "org.apache.flink" %% "flink-clients" % "1.2.0"
)
```

`2.11`*scala* 。

WordCount - API

*WordCount*Table API。 *WordCount* 。

Maven

Table API`flink-table`maven

```
<dependency>
```

```
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table_2.11</artifactId>
    <version>1.1.4</version>
</dependency>
```

```
public class WordCountTable{

    public static void main( String[] args ) throws Exception{

        // set up the execution environment
        final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        final BatchTableEnvironment tableEnv = TableEnvironment.getTableEnvironment( env );

        // get input data
        DataSource<String> source = env.fromElements(
                "To be, or not to be,--that is the question:--",
                "Whether 'tis nobler in the mind to suffer",
                "The slings and arrows of outrageous fortune",
                "Or to take arms against a sea of troubles"
        );

        // split the sentences into words
        FlatMapOperator<String, String> dataset = source
                .flatMap( ( String value, Collector<String> out ) -> {
                    for( String token : value.toLowerCase().split( "\\W+" ) ){
                        if( token.length() > 0 ){
                            out.collect( token );
                        }
                    }
                } )
                // with lambdas, we need to tell flink what type to expect
                .returns( String.class );

        // create a table named "words" from the dataset
        tableEnv.registerDataSet( "words", dataset, "word" );

        // word count using an sql query
        Table results = tableEnv.sql( "select word, count(*) from words group by word" );
        tableEnv.toDataSet( results, Row.class ).print();
    }
}
```

Java <8lambda

```
FlatMapOperator<String, String> dataset = source.flatMap( new FlatMapFunction<String,
String>(){
        @Override
        public void flatMap( String value, Collector<String> out ) throws Exception{
            for( String token : value.toLowerCase().split( "\\W+" ) ){
                if( token.length() > 0 ){
                    out.collect( token );
                }
            }
        }
    } );
```

Maven

`flink-javaflink-client` *JVM*。

```
public class WordCount{

    public static void main( String[] args ) throws Exception{

        // set up the execution environment
        final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();

        // input data
        // you can also use env.readTextFile(...) to get words
        DataSet<String> text = env.fromElements(
                "To be, or not to be,--that is the question:--",
                "Whether 'tis nobler in the mind to suffer",
                "The slings and arrows of outrageous fortune",
                "Or to take arms against a sea of troubles,"
        );

        DataSet<Tuple2<String, Integer>> counts =
                // split up the lines in pairs (2-tuples) containing: (word,1)
                text.flatMap( new LineSplitter() )
                        // group by the tuple field "0" and sum up tuple field "1"
                        .groupBy( 0 )
                        .aggregate( Aggregations.SUM, 1 );

        // emit result
        counts.print();
    }
}
```

`LineSplitter.java`

```
public class LineSplitter implements FlatMapFunction<String, Tuple2<String, Integer>>{

    public void flatMap( String value, Collector<Tuple2<String, Integer>> out ){
        // normalize and split the line into words
        String[] tokens = value.toLowerCase().split( "\\W+" );

        // emit the pairs
        for( String token : tokens ){
            if( token.length() > 0 ){
                out.collect( new Tuple2<String, Integer>( token, 1 ) );
            }
        }
    }
}
```

Java 8lambda.`flatmap(new LineSplitter())`

```
DataSet<Tuple2<String, Integer>> counts = text
    // split up the lines in pairs (2-tuples) containing: (word,1)
    .flatMap( ( String value, Collector<Tuple2<String, Integer>> out ) -> {
        // normalize and split the line into words
        String[] tokens = value.toLowerCase().split( "\\W+" );

        // emit the pairs
        for( String token : tokens ){
            if( token.length() > 0 ){
```

```
                out.collect( new Tuple2<>( token, 1 ) );
            }
        }
    } )
    // group by the tuple field "0" and sum up tuple field "1"
    .groupBy( 0 )
    .aggregate( Aggregations.SUM, 1 );
```

IDE IDE。 FlinkJVM。

flink

1. flink `flink/bin/start-local.sh` ;
2. jar `maven package` ;
3. `flink`flink`bin`

   ```
   flink run -c your.package.WordCount target/your-jar.jar
   ```

   `-c`。 jar/。

```
(a,1)
(against,1)
(and,1)
(arms,1)
(arrows,1)
(be,2)
(fortune,1)
(in,1)
(is,1)
(mind,1)
(nobler,1)
(not,1)
(of,2)
(or,2)
(outrageous,1)
(question,1)
(sea,1)
(slings,1)
(suffer,1)
(take,1)
(that,1)
(the,3)
(tis,1)
(to,4)
(troubles,1)
(whether,1)
```

WordCount - API

*WordCount*Table API。*WordCount* 。

Maven

Streaming API`flink-streaming`maven

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java_2.11</artifactId>
    <version>1.1.4</version>
</dependency>
```

```
public class WordCountStreaming{

    public static void main( String[] args ) throws Exception{

        // set up the execution environment
        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

        // get input data
        DataStreamSource<String> source = env.fromElements(
                "To be, or not to be,--that is the question:--",
                "Whether 'tis nobler in the mind to suffer",
                "The slings and arrows of outrageous fortune",
                "Or to take arms against a sea of troubles"
        );

        source
                // split up the lines in pairs (2-tuples) containing: (word,1)
                .flatMap( ( String value, Collector<Tuple2<String, Integer>> out ) -> {
                    // emit the pairs
                    for( String token : value.toLowerCase().split( "\\W+" ) ){
                        if( token.length() > 0 ){
                            out.collect( new Tuple2<>( token, 1 ) );
                        }
                    }
                } )
                // due to type erasure, we need to specify the return type
                .returns( TupleTypeInfo.getBasicTupleTypeInfo( String.class, Integer.class ) )
                // group by the tuple field "0"
                .keyBy( 0 )
                // sum up tuple on field "1"
                .sum( 1 )
                // print the result
                .print();

        // start the job
        env.execute();
    }
}
```

apache-flink https://riptutorial.com/zh-TW/apache-flink/topic/5798/apache-flink

2:

“” flink。 Flink 1.2。

Examples

ab。

-
- Flink
-
- ...

1.3

- 。

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
env.enableCheckpointing(checkpointInterval);
```

```
java.lang.IllegalStateException: Checkpointing disabled. You can enable it via the
execution environment of your job
```

- ;
- **uid** 。 FlinkUID;
- ID。 ID[a - > b - > c]IDbc。 。 IDID。

。

`flink/conf/flink-conf.yaml` Mac OSX`/usr/local/Cellar/apache-flink/1.1.3/libexec/conf/flink-conf.yaml` 。

Flink <1.2 。 Flink。

```
# Supported backends: filesystem, <class-name-of-factory>
savepoints.state.backend: filesystem
```

```
# Use "hdfs://" for HDFS setups, "file://" for UNIX/POSIX-compliant file systems,
# (or any local file system under Windows), or "S3://" for S3 file system.
# Note: must be accessible from the JobManager and all TaskManagers !
savepoints.state.backend.fs.checkpointdir: file:///tmp/flink-backend/savepoints
```

jobmanager 。 。

Flink 1.2+ jirajobmanager。 Flink 1.2。

```
# Default savepoint target directory
state.savepoints.dir: hdfs:///flink/savepoints
```

ID

ID。 ID`flink list`ID

```
flink list
Retrieving JobManager.
Using address localhost/127.0.0.1:6123 to connect to JobManager.
------------------ Running/Restarting Jobs -------------------
17.03.2017 11:44:03 : 196b8ce6788d0554f524ba747c4ea54f : CheckpointExample (RUNNING)
--------------------------------------------------------------
No scheduled jobs.
```

`flink savepoint <jobID>`

```
flink savepoint 196b8ce6788d0554f524ba747c4ea54f
Retrieving JobManager.
Using address /127.0.0.1:6123 to connect to JobManager.
Triggering savepoint for job 196b8ce6788d0554f524ba747c4ea54f.
Waiting for response...
Savepoint completed. Path: file:/tmp/flink-backend/savepoints/savepoint-a40111f915fc
You can resume your program from this savepoint with the run command.
```

`flink/bin/flink-conf.yamlflink/bin/flink-conf.yaml` ◦

Flink 1.2+`-s`

```
flink cancel -s 196b8ce6788d0554f524ba747c4ea54f # use default savepoints dir
flink cancel -s hdfs:///savepoints 196b8ce6788d0554f524ba747c4ea54f # specify target dir
```

`flink run-s [savepoint-dir]`

```
flink run -s /tmp/flink-backend/savepoints/savepoint-a40111f915fc app.jar
```

UID

UID。 UID.`uid(<name>)`

```
env
    .addSource(source)
    .uid(className + "-KafkaSource01")
    .rebalance()
    .keyBy((node) -> node.get("key").asInt())
    .flatMap(new StatefulMapper())
    .uid(className + "-StatefulMapper01")
    .print();
```

Flink 1.2+

1.2///。 1.2。

1. 。
2. 。 。

;。

Flink。 。 。

[externalized]`flink-conf.yaml`

```
# path to the externalized checkpoints
state.checkpoints.dir: file:///tmp/flink-backend/ext-checkpoints
```

。 state.bachend.fs.checkpointdir。

`getCheckpointConfig()`

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
// enable regular checkpoints
env.enableCheckpointing(5000); // every 5 sec.
// enable externalized checkpoints
env.getCheckpointConfig()

.enableExternalizedCheckpoints(CheckpointConfig.ExternalizedCheckpointCleanup.RETAIN_ON_CANCELLATION);
```

`ExternalizedCheckpointCleanup`

- `RETAIN_ON_CANCELLATION` ;。
- `DELETE_ON_CANCELLATION` 。

。

```
flink run -s /tmp/flink-backend/ext-checkpoints/savepoint-02d0cf7e02ea app.jar
```

https://riptutorial.com/zh-TW/apache-flink/topic/9466/

3:

KafkaRabbitMQJava。

Examples

SerializationSchemaDeserializationSchema。

```
public class MyMessageSchema implements DeserializationSchema<MyMessage>,
SerializationSchema<MyMessage> {

    @Override
    public MyMessage deserialize(byte[] bytes) throws IOException {
        return MyMessage.fromString(new String(bytes));
    }

    @Override
    public byte[] serialize(MyMessage myMessage) {
        return myMessage.toString().getBytes();
    }

    @Override
    public TypeInformation<MyMessage> getProducedType() {
        return TypeExtractor.getForClass(MyMessage.class);
    }

    // Method to decide whether the element signals the end of the stream.
    // If true is returned the element won't be emitted.
    @Override
    public boolean isEndOfStream(MyMessage myMessage) {
        return false;
    }
}
```

MyMessage

```
public class MyMessage{

    public int id;
    public String payload;
    public Date timestamp;

    public MyMessage(){}

    public static MyMessage fromString( String s ){
        String[] tokens = s.split( "," );
        if(tokens.length != 3) throw new RuntimeException( "Invalid record: " + s );

        try{
            MyMessage message = new MyMessage();
            message.id = Integer.parseInt(tokens[0]);
            message.payload = tokens[1];
            message.timestamp = new Date( Long.parseLong(tokens[0]));
            return message;
        }catch(NumberFormatException e){
```

```
            throw new RuntimeException("Invalid record: " + s);
        }
    }

    public String toString(){
        return String.format("%d,%s,%dl", id, payload, timestamp.getTime());
    }
}
```

https://riptutorial.com/zh-TW/apache-flink/topic/9004/--

4:

Examples

KafkaConsumer

`FlinkKafkaConsumer`kafka。

kafka。

- `FlinkKafkaConsumer08` Kafka`SimpleConsumer` API。 Flinkzookeeper。
- `FlinkKafkaConsumer09` KafkaConsumer API。
- `FlinkKafkaProducer010` Kafka。

flink

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kafka-0.${kafka.version}_2.10</artifactId>
  <version>RELEASE</version>
</dependency>
```

-
- Flink/
- kafka。“”。
    - `bootstrap.servers` ipportKafka。 8`zookeeper.connect` zookeeper
    - `group.id` IDkafka

Java

```
Properties properties = new Properties();
properties.put("group.id", "flink-kafka-example");
properties.put("bootstrap.servers", "localhost:9092");

DataStream<String> inputStream = env.addSource(
        new FlinkKafkaConsumer09<>(
            kafkaInputTopic, new SimpleStringSchema(), properties));
```

scala

```
val properties = new Properties();
properties.setProperty("bootstrap.servers", "localhost:9092");
properties.setProperty("group.id", "test");

inputStream = env.addSource(
        new FlinkKafkaConsumer08[String](
            "topic", new SimpleStringSchema(), properties))
```

kafka`enable.auto.commit=falseauto.offset.reset=earliest`pogram。

FlinkFlink Kafka ConsumerKafka。FlinkKafka。

。

Kafka`enableCheckpointing`

```
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
env.enableCheckpointing(5000); // checkpoint every 5 seconds
```

SimpleStringSchema `SimpleStringSchema`。 。

```
new FlinkKafkaConsumer09<>(kafkaInputTopic, new SimpleStringSchema(), prop);
```

JSONDeserializationSchema

`JSONDeserializationSchema`*jackson*json`com.fasterxml.jackson.databind.node.ObjectNode`。

`.get("property")`。 。

```
new FlinkKafkaConsumer09<>(kafkaInputTopic, new JSONDeserializationSchema(), prop);
```

JSONKeyValueDeserializationSchema

`JSONKeyValueDeserializationSchema`json。

```
boolean fetchMetadata = true;
new FlinkKafkaConsumer09<>(kafkaInputTopic, new
JSONKeyValueDeserializationSchema(fetchMetadata), properties);
```

`ObjectNode`

- `key` `key`
- `value`
- `metadata` `offset` `partition`topic `true`。

```
kafka-console-producer --broker-list localhost:9092 --topic json-topic \
    --property parse.key=true \
    --property key.separator=|
{"keyField1": 1, "keyField2": 2} | {"valueField1": 1, "valueField2" : {"foo": "bar"}}
^C
```

```
{
    "key":{"keyField1":1,"keyField2":2},
    "value":{"valueField1":1,"valueField2":{"foo":"bar"}},
    "metadata":{
        "offset":43,
        "topic":"json-topic",
        "partition":0
    }
}
```

KafkaFlink

kafka。 。 flinkflink1。

1. **kafka== flink parallelism** 。 flink;
2. **kafka<flink parallelism** flink。 `rebalance`

```
inputStream = env.addSource(new FlinkKafkaConsumer10("topic", new SimpleStringSchema(),
properties));

inputStream
    .rebalance()
    .map(s -> "message" + s)
    .print();
```

3. **kafka> flink parallelism** 。 `rebalance`。

https://riptutorial.com/zh-TW/apache-flink/topic/9003/

5:

Flink 1.2

Flink。Flink。。

。

。。

/flink 1.2+。

Examples

。。。

- `MemoryStateBackend` JobManager/ ZooKeeper。5 MBKafka。
- `FsStateBackend` TaskManagersHDFSDS3.....。。
- `RocksDBStateBackend` RocksDBTaskManager。RocksDB。FsStateBackend。

/。

`main`

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
env.setStateBackend(new FsStateBackend("hdfs://namenode:40010/flink/checkpoints"));
```

`flink/conf/flink-conf.yaml`

```
# Supported backends:
#  - jobmanager (MemoryStateBackend),
#  - filesystem (FsStateBackend),
#  - rocksdb (RocksDBStateBackend),
#  - <class-name-of-factory>
state.backend: filesystem

# Directory for storing checkpoints in a Flink-supported filesystem
# Note: State backend must be accessible from the JobManager and all TaskManagers.
# Use "hdfs://" for HDFS setups, "file://" for UNIX/POSIX-compliant file systems,
# "S3://" for S3 file system.
state.backend.fs.checkpointdir: file:///tmp/flink-backend/checkpoints
```

```
long checkpointInterval = 5000; // every 5 seconds

StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
env.enableCheckpointing(checkpointInterval);
```

◦

```
env.enableCheckpointing(checkpointInterval, CheckpointingMode.AT_LEAST_ONCE);
```

◦ ◦ EXACTLY_ONCE /“”◦ AT_LEAST_ONCE ◦

flinkInteger◦ checkpointEnable checkpointIntervalcheckpointMode

```
public class CheckpointExample {

    private static Logger LOG = LoggerFactory.getLogger(CheckpointExample.class);
    private static final String KAFKA_BROKER = "localhost:9092";
    private static final String KAFKA_INPUT_TOPIC = "input-topic";
    private static final String KAFKA_GROUP_ID = "flink-stackoverflow-checkpointer";
    private static final String CLASS_NAME = CheckpointExample.class.getSimpleName();


    public static void main(String[] args) throws Exception {

        // play with them
        boolean checkpointEnable = false;
        long checkpointInterval = 1000;
        CheckpointingMode checkpointMode = CheckpointingMode.EXACTLY_ONCE;

        // -------------------------------------------------

        LOG.info(CLASS_NAME + ": starting...");
        final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

        // kafka source
        // https://ci.apache.org/projects/flink/flink-docs-release-
1.2/dev/connectors/kafka.html#kafka-consumer
        Properties prop = new Properties();
        prop.put("bootstrap.servers", KAFKA_BROKER);
        prop.put("group.id", KAFKA_GROUP_ID);
        prop.put("auto.offset.reset", "latest");
        prop.put("enable.auto.commit", "false");

        FlinkKafkaConsumer09<String> source = new FlinkKafkaConsumer09<>(
                KAFKA_INPUT_TOPIC, new SimpleStringSchema(), prop);

        // checkpoints
        // internals: https://ci.apache.org/projects/flink/flink-docs-
master/internals/stream_checkpointing.html#checkpointing
        // config: https://ci.apache.org/projects/flink/flink-docs-release-
1.3/dev/stream/checkpointing.html
        if (checkpointEnable) env.enableCheckpointing(checkpointInterval, checkpointMode);

        env
                .addSource(source)
                .keyBy((any) -> 1)
                .flatMap(new StatefulMapper())
                .print();

        env.execute(CLASS_NAME);
    }
```

```
        /* ****************************************************************
         * Stateful mapper
         * (cf. https://ci.apache.org/projects/flink/flink-docs-release-
1.3/dev/stream/state.html)
         * ****************************************************************/

    public static class StatefulMapper extends RichFlatMapFunction<String, String> {
        private transient ValueState<Integer> state;

        @Override
        public void flatMap(String record, Collector<String> collector) throws Exception {
            // access the state value
            Integer currentState = state.value();

            // update the counts
            currentState += 1;
            collector.collect(String.format("%s: (%s,%d)",
                    LocalDateTime.now().format(ISO_LOCAL_DATE_TIME), record, currentState));
            // update the state
            state.update(currentState);
        }

        @Override
        public void open(Configuration parameters) throws Exception {
            ValueStateDescriptor<Integer> descriptor =
                    new ValueStateDescriptor<>("CheckpointExample",
TypeInformation.of(Integer.class), 0);
            state = getRuntimeContext().getState(descriptor);
        }
    }
}
```

cluster ◦ flink/binstart-cluster.sh

```
start-cluster.sh
Starting cluster.
[INFO] 1 instance(s) of jobmanager are already running on virusnest.
Starting jobmanager daemon on host virusnest.
Password:
Starting taskmanager daemon on host virusnest.
```

flink

```
mvn clean package
flink run target/flink-checkpoints-test.jar -c CheckpointExample
```

```
kafka-console-producer --broker-list localhost:9092 --topic input-topic
a
b
c
^D
```

flink/logs/flink-<user>-jobmanager-0-<host>.out◦

```
tail -f flink/logs/flink-Derlin-jobmanager-0-virusnest.out
2017-03-17T08:21:51.249: (a,1)
```

```
2017-03-17T08:21:51.545: (b,2)
2017-03-17T08:21:52.363: (c,3)
```

```
# killing the taskmanager
ps -ef | grep -i taskmanager
kill <taskmanager PID>

# starting a new taskmanager
flink/bin/taskmanager.sh start
```

taskmanager`flink/logs/flink-<user>-jobmanager-1-<host>.out` ◦

- ◦
- ◦
-

https://riptutorial.com/zh-TW/apache-flink/topic/9465/

6: API

Examples

Maven

Table API`flink-table`maven`flink-clientsflink-core`

```
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table_2.11</artifactId>
    <version>1.1.4</version>
</dependency>
```

scala2.11。

CSV

CSV`peoples.csv`

```
1,Reed,United States,Female
2,Bradley,United States,Female
3,Adams,United States,Male
4,Lane,United States,Male
5,Marshall,United States,Female
6,Garza,United States,Male
7,Gutierrez,United States,Male
8,Fox,Germany,Female
9,Medina,United States,Male
10,Nichols,United States,Male
11,Woods,United States,Male
12,Welch,United States,Female
13,Burke,United States,Female
14,Russell,United States,Female
15,Burton,United States,Male
16,Johnson,United States,Female
17,Flores,United States,Male
18,Boyd,United States,Male
19,Evans,Germany,Male
20,Stephens,United States,Male
```

+

```
public class TableExample{
    public static void main( String[] args ) throws Exception{
        // create the environments
        final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        final BatchTableEnvironment tableEnv = TableEnvironment.getTableEnvironment( env );

        // get the path to the file in resources folder
        String peoplesPath = TableExample.class.getClassLoader().getResource( "peoples.csv"
).getPath();
```

```
        // load the csv into a table
        CsvTableSource tableSource = new CsvTableSource(
                peoplesPath,
                "id,last_name,country,gender".split( "," ),
                new TypeInformation[]{ Types.INT(), Types.STRING(), Types.STRING(),
Types.STRING() } );
        // register the table and scan it
        tableEnv.registerTableSource( "peoples", tableSource );
        Table peoples = tableEnv.scan( "peoples" );

        // aggregation using chain of methods
        Table countriesCount = peoples.groupBy( "country" ).select( "country, id.count" );
        DataSet<Row> result1 = tableEnv.toDataSet( countriesCount, Row.class );
        result1.print();

        // aggregation using SQL syntax
        Table countriesAndGenderCount = tableEnv.sql(
                "select country, gender, count(id) from peoples group by country, gender" );

        DataSet<Row> result2 = tableEnv.toDataSet( countriesAndGenderCount, Row.class );
        result2.print();
    }
}
```

```
Germany,2
United States,18

Germany,Male,1
United States,Male,11
Germany,Female,1
United States,Female,7
```

`peoples.csv` *CSV* CSV。

`sales.csv` people_idproduct_id

```
19,5
6,4
10,4
2,4
8,1
19,2
8,4
5,5
13,5
4,4
6,1
3,3
8,3
17,2
6,2
1,2
3,5
15,5
3,3
6,3
13,2
20,4
```

```
20,2
```

`products.csv` idnameprice

```
1,Loperamide,47.29
2,pain relief pm,61.01
3,Citalopram,48.13
4,CTx4 Gel 5000,12.65
5,Namenda,27.67
```

40

```
public class SimpleJoinExample{
    public static void main( String[] args ) throws Exception{

        final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
        final BatchTableEnvironment tableEnv = TableEnvironment.getTableEnvironment( env );

        String peoplesPath = TableExample.class.getClassLoader().getResource( "peoples.csv"
).getPath();
        String productsPath = TableExample.class.getClassLoader().getResource( "products.csv"
).getPath();
        String salesPath = TableExample.class.getClassLoader().getResource( "sales.csv"
).getPath();

        Table peoples = csvTable(
                tableEnv,
                "peoples",
                peoplesPath,
                "pe_id,last_name,country,gender",
                new TypeInformation[]{ Types.INT(), Types.STRING(), Types.STRING(),
Types.STRING() } );

        Table products = csvTable(
                tableEnv,
                "products",
                productsPath,
                "prod_id,product_name,price",
                new TypeInformation[]{ Types.INT(), Types.STRING(), Types.FLOAT() } );

        Table sales = csvTable(
                tableEnv,
                "sales",
                salesPath,
                "people_id,product_id",
                new TypeInformation[]{ Types.INT(), Types.INT() } );

        // here is the interesting part:
        Table join = peoples
                .join( sales ).where( "pe_id = people_id" )
                .join( products ).where( "product_id = prod_id" )
                .select( "last_name, product_name, price" )
                .where( "price < 40" );

        DataSet<Row> result = tableEnv.toDataSet( join, Row.class );
        result.print();

    }//end main
```

```
    public static Table csvTable( BatchTableEnvironment tableEnv, String name, String path,
String header,
                                  TypeInformation[]
                                          typeInfo ){
        CsvTableSource tableSource = new CsvTableSource( path, header.split( "," ), typeInfo);
        tableEnv.registerTableSource( name, tableSource );
        return tableEnv.scan( name );
    }

}//end class
```

flink“”。

```
Burton,Namenda,27.67
Marshall,Namenda,27.67
Burke,Namenda,27.67
Adams,Namenda,27.67
Evans,Namenda,27.67
Garza,CTx4 Gel 5000,12.65
Fox,CTx4 Gel 5000,12.65
Nichols,CTx4 Gel 5000,12.65
Stephens,CTx4 Gel 5000,12.65
Bradley,CTx4 Gel 5000,12.65
Lane,CTx4 Gel 5000,12.65
```

TableSink。 `BatchTableSink` `StreamTableSink` 。

flink`CsvTableSink`。

```
DataSet<Row> result = tableEnv.toDataSet( table, Row.class );
result.print();
```

```
TableSink sink = new CsvTableSink("/tmp/results", ",");
// write the result Table to the TableSink
table.writeToSink(sink);
// start the job
env.execute();
```

`/tmp/results`flink。 44。

`env.execute()` flink `print()`。

API https://riptutorial.com/zh-TW/apache-flink/topic/8966/api

7:

Flinklog4j。

Examples

`slf4jpom.xml`

```
<properties>
    <slf4j.version>1.7.21</slf4j.version>
</properties>

<!-- ... -->

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>${slf4j.version}</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>${slf4j.version}</version>
</dependency>
```

logger

```
private Logger LOGGER = LoggerFactory.getLogger(FlinkApp.class);
```

`RichMapFunctionLOGGERtransient`

```
private transient Logger LOG = LoggerFactory.getLogger(MyRichMapper.class);
```

`LOGGER。 {}`

```
LOGGER.info("my app is starting");
LOGGER.warn("an exception occurred processing {}", record, exception);
```

IDE`log4j log4j.properties`。 maven`src/main/resourceslog4j.properties`。

```
log4j.rootLogger=INFO, console

# patterns:
#  d = date
#  c = class
#  F = file
#  p = priority (INFO, WARN, etc)
#  x = NDC (nested diagnostic context) associated with the thread that generated the logging
event
#  m = message
```

```
# Log all infos in the console
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{dd/MM/yyyy HH:mm:ss.SSS} %5p [%-10c] %m%n

# Log all infos in flink-app.log
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.file=flink-app.log
log4j.appender.file.append=false
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{dd/MM/yyyy HH:mm:ss.SSS} %5p [%-10c] %m%n

# suppress info messages from flink
log4j.logger.org.apache.flink=WARN
```

`jar`。 Flink。

Flink

- `log4j-cli.properties` Flink`flink run`
- `log4j-yarn-session.properties` YARNFlink `yarn-session.sh`
- `log4j.properties` / TaskmanagerYARN

`${log.file}flink/log`。 `env.log.dirflink-conf.yaml`

`env.log.dir`Flink。 。

JobManager / Taskmanager。

Yarn YarnFlinkHadoop YARN。 YARN 。 `yarn-site.xml fileyarn.log-aggregation-enabletrue`。 YARN

```
yarn logs -applicationId <application ID>
```

。

Flink 1.2。

flink*one-yarn-cluster-per-job* `flink run -m yarn-cluster` ...

1. `conf`
2. `flink/conf`

   ```
   mkdir conf
   cd conf
   ln -s flink/conf/* .
   ```

3. `log4j.properties`
4. ```
   export FLINK_CONF_DIR=/path/to/my/conf
   ```
   ```

flink`flink/bin/config.sh`◦

```
FLINK_CONF_DIR=$FLINK_ROOT_DIR_MANGLED/conf
```

```
if [ -z "$FLINK_CONF_DIR" ]; then
    FLINK_CONF_DIR=$FLINK_ROOT_DIR_MANGLED/conf;
fi
```

Flink-on-Yarnrsyslog

Yarn◦

`rsyslog` Linux◦

`/etc/rsyslog.conf`udp

```
$ModLoad imudp
$UDPServerRun 514
```

`log4j.properties SyslogAppender`

```
log4j.rootLogger=INFO, file

# TODO: change package logtest to your package
log4j.logger.logtest=INFO, SYSLOG

# Log all infos in the given file
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.file=${log.file}
log4j.appender.file.append=false
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=bbdata: %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p %-60c %x
- %m%n

# suppress the irrelevant (wrong) warnings from the netty channel handler
log4j.logger.org.jboss.netty.channel.DefaultChannelPipeline=ERROR, file

# rsyslog
# configure Syslog facility SYSLOG appender
# TODO: replace host and myTag by your own
log4j.appender.SYSLOG=org.apache.log4j.net.SyslogAppender
log4j.appender.SYSLOG.syslogHost=10.10.10.102
log4j.appender.SYSLOG.port=514
#log4j.appender.SYSLOG.appName=bbdata
log4j.appender.SYSLOG.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.SYSLOG.layout.conversionPattern=myTag: [%p] %c:%L - %m %throwable %n
```

rsyslog◦ stacktraces◦ /“”`rsyslog.conf`

```
$EscapeControlCharactersOnReceive off
```

`conversionPatternmyTag:◦ rsyslog.conf`

```
if $programname == 'myTag' then /var/log/my-app.log
& stop
```

https://riptutorial.com/zh-TW/apache-flink/topic/9713/

S. No		Contributors
1	apache-flink	Community, Derlin, vdep
2		Derlin
3		Derlin
4		alpinegizmo, Derlin
5		Derlin
6	API	Derlin
7		Derlin