```
(base) [bigdata@localhost bin]$ ./start-scala-shell.sh --help
Flink Scala Shell
Usage: start-scala-shell.sh [local|remote|yarn] [options] <args>...

Command: local [options]
Starts Flink scala shell with a local Flink cluster
  -a, --addclasspath <path/to/jar>
                        Specifies additional jars to be used in Flink
Command: remote [options] <host> <port>
Starts Flink scala shell connecting to a remote cluster
  <host>                Remote host name as string
  <port>                Remote port as integer

  -a, --addclasspath <path/to/jar>
                        Specifies additional jars to be used in Flink
Command: yarn [options]
Starts Flink scala shell connecting to a yarn cluster
  -n, --container arg      Number of YARN container to allocate (= Number of TaskManagers)
  -jm, --jobManagerMemory arg
                        Memory for JobManager container
  -nm, --name <value>      Set a custom name for the application on YARN
  -qu, --queue <arg>       Specifies YARN queue
  -s, --slots <arg>        Number of slots per TaskManager
  -tm, --taskManagerMemory <arg>
                        Memory per TaskManager container
  -a, --addclasspath <path/to/jar>
                        Specifies additional jars to be used in Flink
  --configDir <value>      The configuration directory.
  -h, --help               Prints this usage text

(base) [bigdata@localhost bin]$ ./start-scala-shell.sh local
Starting Flink Shell:
log4j:WARN No appenders could be found for logger
(org.apache.flink.configuration.GlobalConfiguration).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Starting local Flink cluster (host: localhost, port: 8082).

Connecting to Flink cluster (host: localhost, port: 8082).

            F L I N K - S C A L A - S H E L L

NOTE: Use the prebound Execution Environments to implement batch or streaming programs.

  Batch - Use the 'benv' variable

    * val dataSet = benv.readTextFile("/path/to/data")
    * dataSet.writeAsText("/path/to/output")
    * benv.execute("My batch program")
```

HINT: You can use print() on a DataSet to print the contents to the shell.

Streaming - Use the 'senv' variable

* val dataStream = senv.fromElements(1, 2, 3, 4)
* dataStream.countWindowAll(2).sum(0).print()
* senv.execute("My streaming program")

HINT: You can only print a DataStream to the shell in local mode.

```
scala> val text = benv.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,")

scala> val counts = text.flatMap { _.toLowerCase.split("\\W+") }.map { (_, 1) }.groupBy(0).sum(1)
counts: org.apache.flink.api.scala.AggregateDataSet[(String, Int)] =
org.apache.flink.api.scala.AggregateDataSet@50ccd6a

scala> counts.print()
(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)
scala> :q
 good bye ..
scala> val text = benv.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,")
scala> val counts = text.flatMap { _.toLowerCase.split("\\W+") }.map { (_, 1) }.groupBy(0).sum(1)
```

```
scala> counts.print()
```

```
(base) [bigdata@localhost bin]$ start-cluster.sh
Starting cluster.
Starting standalonesession daemon on host localhost.localdomain.
Starting taskexecutor daemon on host localhost.localdomain.
(base) [bigdata@localhost flink]$ ./bin/flink run ./examples/batch/WordCount.jar
Starting execution of program
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
(a,5)
(action,1)
(after,1)
(against,1)
(all,2)
(and,12)
(arms,1)
(arrows,1)
(awry,1)
(ay,1)
(bare,1)
(be,4)
(bear,3)
(bodkin,1)
(bourn,1)
(but,1)
(by,2)
(calamity,1)
(cast,1)
(coil,1)
(come,1)
(conscience,1)
(consummation,1)
(contumely,1)
(country,1)
(cowards,1)
(currents,1)
(d,4)
(death,2)
(delay,1)
(despis,1)
(devoutly,1)
(die,2)
(does,1)
(dread,1)
(dream,1)
(dreams,1)
```

(end,2)
(enterprises,1)
(er,1)
(fair,1)
(fardels,1)
(flesh,1)
(fly,1)
(for,2)
(fortune,1)
(from,1)
(give,1)
(great,1)
(grunt,1)
(have,2)
(he,1)
(pause,1)
(perchance,1)
Program execution finished
Job with JobID 6888eccaa01ea3648699ae7e720840e8 has finished.
Job Runtime: 10559 ms
Accumulator Results:
- 9babfa1ec971204531922c414bad35a8 (java.util.ArrayList) [170 elements]

(base) [bigdata@localhost flink]$ export JOB_ID="6888eccaa01ea3648699ae7e720840e8"
(base) [bigdata@localhost flink]$ ./bin/flink list
Waiting for response...
No running jobs.
No scheduled jobs.
(base) [bigdata@localhost flink]$ ./bin/flink cancel 6888eccaa01ea3648699ae7e720840e8
(base) [bigdata@localhost flink]$ ./bin/flink savepoint \da6369ffb61e4548e53a6a0442bfe7f5 \ /tmp/flink-savepoints
Triggering savepoint for job da6369ffb61e4548e53a6a0442bfe7f5.
Waiting for response...
Savepoint completed. Path: file:/tmp/flink-savepoints/savepoint-da6369-20fa00003bb2
You can resume your program from this savepoint with the run command.
(base) [bigdata@localhost flink]$ flink run -d examples/streaming/TopSpeedWindowing.jar
Starting execution of program
Executing TopSpeedWindowing example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
Job has been submitted with JobID e180a9cfd5b9246e27656c196711156f

(base) [bigdata@localhost flink]$ ./bin/taskmanager.sh start
[INFO] 1 instance(s) of taskexecutor are already running on localhost.localdomain.
Starting taskexecutor daemon on host localhost.localdomain.
(base) [bigdata@localhost flink]$ flink run -d examples/streaming/WordCount.jar --input /home/bigdata/shakespeare.txt --output /home/bigdata/flink/output
Starting execution of program
Job has been submitted with JobID 1047c823107e3a55ea362e9436d0b1dc
Check your flink directory for output

```
(base) [bigdata@localhost flink]$ stop-cluster.sh
Stopping taskexecutor daemon (pid: 7224) on host localhost.localdomain.
Stopping standalonesession daemon (pid: 3688) on host localhost.localdomain.
(base) [bigdata@localhost flink]$ ./bin/taskmanager.sh stop
Stopping taskexecutor daemon (pid: 4154) on host localhost.localdomain.

export JOB_ID="d2d75f9ecd1995d0b8ee3684df213a81"

./bin/flink list
./bin/flink cancel d2d75f9ecd1995d0b8ee3684df213a81
./bin /flink run -d examples/streaming/ StateMachineExample.jar
./bin/flink savepoint \da6369ffb61e4548e53a6a0442bfe7f5 \ /tmp/flink-savepoints
flink run -d examples/streaming/WordCount.jar
flink list -m 127.0.0.1:8081
flink cancel -m 127.0.0.1:8081 16df10697a3774545ffb71cd2d5ddcd1
flink savepoint -m 127.0.0.1:8081 ec53edcfaeb96b2a5dadbfbe5ff62bbb /tmp/savepoint
```

Application Mode

To start a Flink JobManager with an embedded application, we use the bin/standalone-job.sh script. We demonstrate this mode by locally starting the TopSpeedWindowing.jar example, running on a single TaskManager.

The application jar file needs to be available in the classpath. The easiest approach to achieve that is putting the jar into the lib/ folder:

```
$ cp ./examples/streaming/TopSpeedWindowing.jar lib/
```

Then, we can launch the JobManager:

```
$ ./bin/standalone-job.sh start --job-classname org.apache.flink.streaming.examples.windowing.TopSpeedWindowing
```

The web interface is now available at localhost:8081. However, the application won't be able to start, because there are no TaskManagers running yet:

```
$ ./bin/taskmanager.sh start
```

Note: You can start multiple TaskManagers, if your application needs more resources.

Stopping the services is also supported via the scripts. Call them multiple times if you want to stop multiple instances, or use stop-all:

```
$ ./bin/taskmanager.sh stop
$ ./bin/standalone-job.sh stop
```

```
(base) [bigdata@localhost bin]$ ./start-scala-shell.sh --help
Flink Scala Shell
Usage: start-scala-shell.sh [local|remote|yarn] [options] <args>...
```

```
Command: local [options]
Starts Flink scala shell with a local Flink cluster
  -a, --addclasspath <path/to/jar>
                        Specifies additional jars to be used in Flink
Command: remote [options] <host> <port>
Starts Flink scala shell connecting to a remote cluster
  <host>                Remote host name as string
  <port>                Remote port as integer

  -a, --addclasspath <path/to/jar>
                        Specifies additional jars to be used in Flink
Command: yarn [options]
Starts Flink scala shell connecting to a yarn cluster
  -n, --container arg      Number of YARN container to allocate (= Number of TaskManagers)
  -jm, --jobManagerMemory arg
                        Memory for JobManager container
  -nm, --name <value>      Set a custom name for the application on YARN
  -qu, --queue <arg>       Specifies YARN queue
  -s, --slots <arg>        Number of slots per TaskManager
  -tm, --taskManagerMemory <arg>
                        Memory per TaskManager container
  -a, --addclasspath <path/to/jar>
                        Specifies additional jars to be used in Flink
  --configDir <value>      The configuration directory.
  -h, --help               Prints this usage text
(base) [bigdata@localhost bin]$ ./start-scala-shell.sh local
Starting Flink Shell:
log4j:WARN No appenders could be found for logger
(org.apache.flink.configuration.GlobalConfiguration).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Starting local Flink cluster (host: localhost, port: 8081).
Connecting to Flink cluster (host: localhost, port: 8081).

            F L I N K - S C A L A - S H E L L

NOTE: Use the prebound Execution Environments to implement batch or streaming programs.

  Batch - Use the 'benv' variable

    * val dataSet = benv.readTextFile("/path/to/data")
    * dataSet.writeAsText("/path/to/output")
    * benv.execute("My batch program")

    HINT: You can use print() on a DataSet to print the contents to the shell.
```

Streaming - Use the 'senv' variable

* val dataStream = senv.fromElements(1, 2, 3, 4)
* dataStream.countWindowAll(2).sum(0).print()
* senv.execute("My streaming program")

HINT: You can only print a DataStream to the shell in local mode.

```
scala> val text = benv.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,")
text: org.apache.flink.api.scala.DataSet[String] = org.apache.flink.api.scala.DataSet@10466c55

scala> val counts = text.flatMap { _.toLowerCase.split("\\W+") }.map { (_, 1) }.groupBy(0).sum(1)
counts: org.apache.flink.api.scala.AggregateDataSet[(String, Int)] = org.apache.flink.api.scala.AggregateDataSet@4a29520e

scala> counts.print()
(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)
```

```
scala> :q
 good bye ..
```