Spring Cloud DataFlow and Stream Processing

Spring Cloud Data Flow Lab

Partitioned Stream with SCDF for PCF

We will use a new type of processor in this exercise. A splitter-processor, as you might have assumed, it splits the payload by the specified character. Let's register this Application. If you're comparing with the local mode we have done a bulk load of all of the app-starters so there is no need to register apps as you see in the local development mode examples. (Note: Even in local mode you can bulk load in the stream and task starter apps and avoid these registration steps).

● Create a simple partitioned stream

```
stream create --name words --definition "http  | splitter
--expression=payload.split(' ') | log"
```

● Deploy the partitioned stream with 2 instances of log-sink

```
stream deploy words --properties
"app.splitter.producer.partitionKeyExpression=payload,app.log.count=2"
```

● Tail both the log-sink instances; for example:

```
cf logs aag8JT6-words-log-v2
Retrieving logs for app aag8JT6-words-log-v2 in org stubhub / space lab1 as
wayne...



   2019-03-07T16:49:08.48-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.483
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
How
   2019-03-07T16:49:08.54-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.546
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
```

```
much
   2019-03-07T16:49:08.54-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.548
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
wood
   2019-03-07T16:49:08.55-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.550
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
would
   2019-03-07T16:49:08.55-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.552
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    : a
   2019-03-07T16:49:08.55-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.554
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
woodchuck
   2019-03-07T16:49:08.55-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.555
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
chuck
   2019-03-07T16:49:08.55-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.557
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    : if
   2019-03-07T16:49:08.55-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.558
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    : a
   2019-03-07T16:49:08.55-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.559
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
woodchuck
   2019-03-07T16:49:08.56-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.560
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
could
   2019-03-07T16:49:08.56-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.562
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
chuck
   2019-03-07T16:49:08.56-0800 [APP/PROC/WEB/0] OUT 2019-03-08 00:49:08.563
INFO 20 --- [itter.words-0-1] aag8JT6-words-log-v2                    :
wood
```

● Post the following data

```
dataflow:>http post --target http://localhost:9900 --data "How much
wood would a woodchuck chuck if a woodchuck could chuck wood"
> POST (text/plain;Charset=UTF-8) http://localhost:9900 How much wood
would a woodchuck chuck if a woodchuck could chuck wood
> 202 ACCEPTED
```

● Observe the log-sink logs

```
words.log instance 0
words.log instance 1
```

Setting up the environment (local mode)

1. Install RabbitMQ following the instructions in the document <FOLDER>/DNDataflow/labs/InstallRabbitMQ.pdf.
2. Now let's setup the lab environment:
    a. Using git
        i. From a your local terminal or command prompt change directory to a clean working directory.
        ii. Now execute:

```
git clone https://github.com/cppwfs/DNDataflow.git
```

        iii. iii. Now cd DNDataflow
    b. Using Thumbdrive
        i. i. Copy the DNDataflow directory from the thumbdrive to a location on your laptop hard drive
        ii. ii. Now from a terminal or command prompt cd to the DNDataflow directory you just created on your hard drive.
3. Creating your first stream
    a. If you haven't already completed all the installation steps from `lab3`, please proceed there to set up Spring Cloud Data Flow "Server" and "Shell" applications
    b. Register applications from Shell application

(1) Source

```
app register --name http --type source --uri
file:///<FOLDER>/DNDataflow/labs/jars/http-source-rabbit-2.0.3.RELEASE.jar
```

(2) Processor

```
app register --name uppercase --type processor --uri
file:///<FOLDER>/DNDataflow/labs/jars/streamlab-0.0.1-SNAPSHOT.jar
```

(3) Sink

```
app register --name log --type sink --uri
file:///<FOLDER>/DNDataflow/labs/jars/log-sink-rabbit-2.0.2.RELEASE.jar
```

● List the registered applications

```
app list
```

● Verify the registered applications

```
app info source:http
```

● Create a stream

```
stream create foo --definition "http --port=9001 | uppercase | log"
--deploy
```

● Tail the log-sink logs; for example:

In the server console, you will see both the http-source and log-sink logs being logged to at a special directory. Copy the path for “foo.log” application and use the “tail” command to review the logs like the following. If you are a Windows user, you could either use something like Cygwin or open the file in a text-editor and that could load and refresh the contents continuously.

```
tail -f
/var/folders/c3/ctx7_rns6x30tq7rb76wzqwr0000gp/T/spring-cloud-dataflo
w-3545000607490975505/foo-1486337156762/foo.log/stdout_0.log
```

● Post some data against the target http://localhost:9001

```
dataflow:>http post --target http://localhost:9001 --data "hello world"
> POST (text/plain;Charset=UTF-8) http://localhost:9001 hello world
> 202 ACCEPTED
```

● Verify the log-sink logs for “HELLO WORLD”

● Destroy the stream

```
dataflow:>stream destroy foo
```

Partitioned Stream

We will use a new type of processor in this exercise. A splitter-processor, as you might have assumed, it splits the payload by the specified character. Let’s register this application.

● Create a simple partitioned stream

```
stream create --name words --definition "http splitter --SERVER.PORT=9900 |
SPLITTER --expression=payload.split(' ') | log"
```

● Deploy the partitioned stream with 2 instances of log-sink

```
stream deploy words --properties
"app.splitter.producer.partitionKeyExpression=payload,app.log.count=2"
```

● Tail both the log-sink instances; for example:

In the server console, you will see both the http-source and log-sink logs being logged to a special directory. Copy the path for “ words.log ” application and use the “tail” command to review the logs like the following. There will be 2 instances of this log file; one from each of the log-sink application instance. If you are a Windows user, you could either use something like Cygwin or open the file in a text-editor and that could load and refresh the contents continuously.

tail -f

```
/var/folders/c3/ctx7_rns6x30tq7rb76wzqwr0000gp/T/spring-cloud-dataflo
w-3545000607490975505/words-1486337773441/words.log/stdout_0.log
tail -f
/var/folders/c3/ctx7_rns6x30tq7rb76wzqwr0000gp/T/spring-cloud-dataflo
w-3545000607490975505/words-1486337773441/words.log/stdout_1.log
```

● Post the following data

```
dataflow:>http post --target http://localhost:9900 --data "How much
wood would a woodchuck chuck if a woodchuck could chuck wood"
> POST (text/plain;Charset=UTF-8) http://localhost:9900 How much wood
would a woodchuck chuck if a woodchuck could chuck wood
> 202 ACCEPTED
```

● Observe the log-sink logs

```
words.log instance 0
words.log instance 1
```