Spring Cloud Data Flow Lab

# Setting up the environment

1. Install RabbitMQ following the instructions in **Lab 0 - Install RabbitMQ.**
2. Now let’s setup the lab environment:
   1. Using git
      1. From a your local terminal or command prompt change directory to a clean working directory.
      2. Now execute: git clone https://github.com/cppwfs/DNDataflow.git
      3. Now cd DNDataflow
   2. Using Thumbdrive
      1. Copy the **DNDataflow** directory from the thumbdrive to a location on your laptop hard drive
      2. Now from a terminal or command prompt **cd** to the **DNDataflow** directory you just created on your hard drive.

# Creating your first stream

* 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
* Register applications from Shell application

**(1) Source**

app register --name http --type source --uri file:////<FOLDER>/DNDataflow/labs/jars/http-source-rabbit-1.1.2.RELEASE.jar

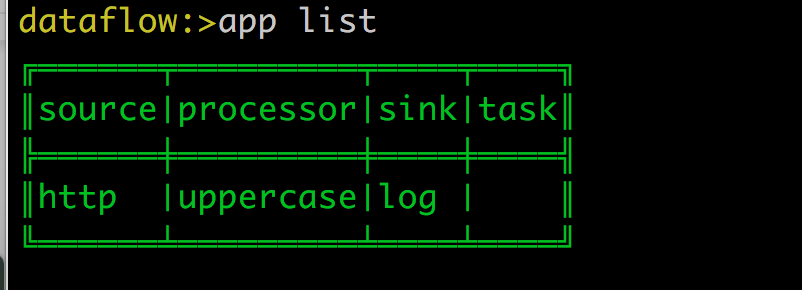
**(2) Processor**

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

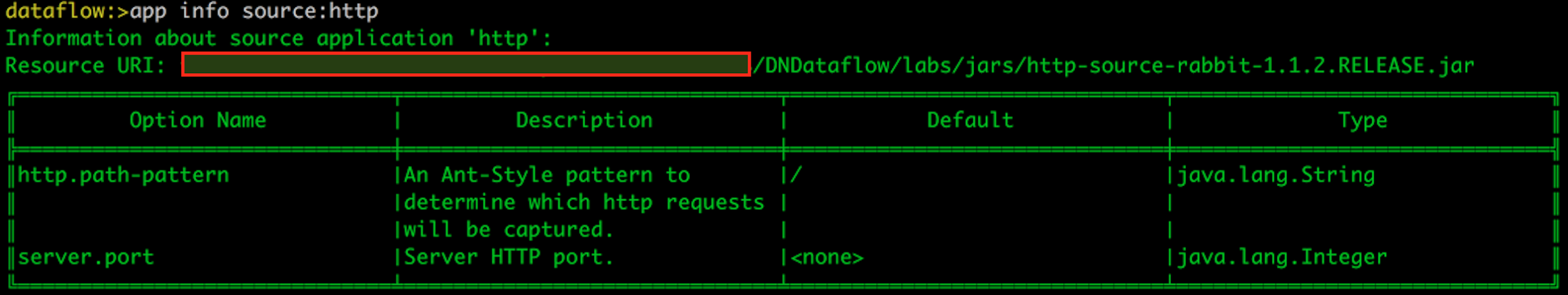
**(3) Sink**

app register --name log --type sink --uri file:////<FOLDER>/DNDataflow/labs/jars/log-sink-rabbit-1.1.1.RELEASE.jar

* List the registered applications



* Verify the registered applications

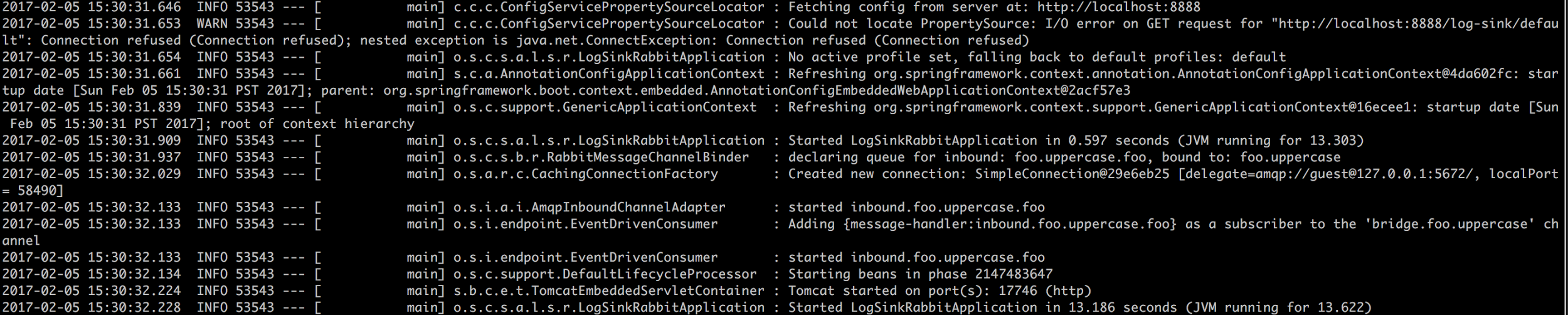


* Create a stream

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

* Tail the log-sink logs; for example:

tail -f /var/folders/c3/ctx7\_rns6x30tq7rb76wzqwr0000gp/T/spring-cloud-dataflow-3545000607490975505/foo-1486337156762/foo.log/stdout\_0.log



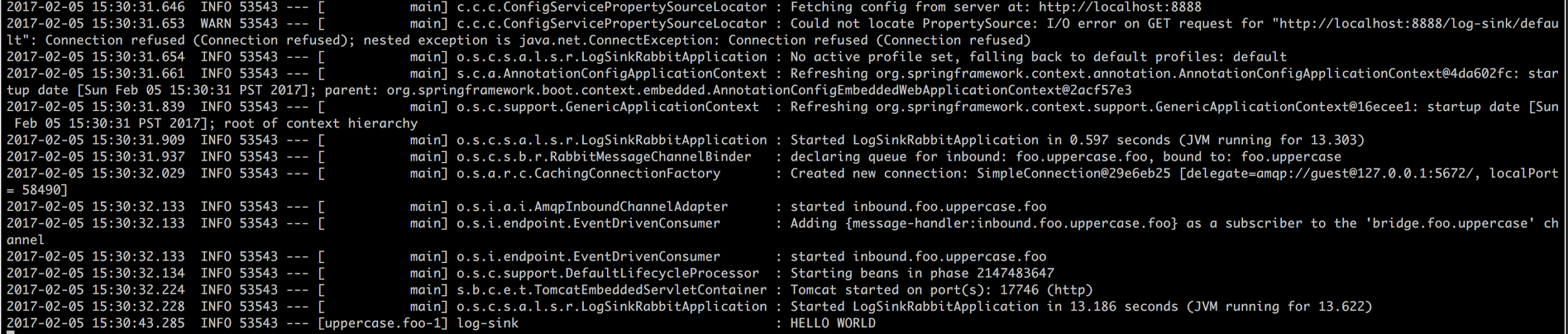
* Post some data against the target <http://localhost:9001>

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”



# Partitioned Stream

* Create a simple partitioned stream

stream create --name words --definition "http --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:

tail -f /var/folders/c3/ctx7\_rns6x30tq7rb76wzqwr0000gp/T/spring-cloud-dataflow-3545000607490975505/words-1486337773441/words.log/stdout\_0.log

tail -f /var/folders/c3/ctx7\_rns6x30tq7rb76wzqwr0000gp/T/spring-cloud-dataflow-3545000607490975505/words-1486337773441/words.log/stdout\_1.log

* Post the following data

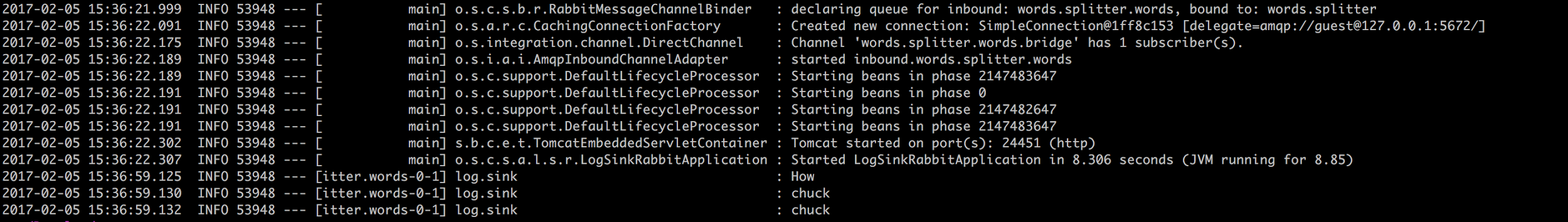
dataflow:>http post --target http://localhost:9900 --data "How much wood would a woodchuck chuck if a woodchuck could chuck wood"

> 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

