ServiceMix

**Abstract:**

Team: 4Spartans

Saloni Sharad Bhalerao 012466623 Github  
Anu Sebastian 012496276 Github  
Tarun Shetty 012449697 Github  
 Hemambujam Veeraraghavan 012482392 [Github](https://github.com/hemambujamV/Servicemix)

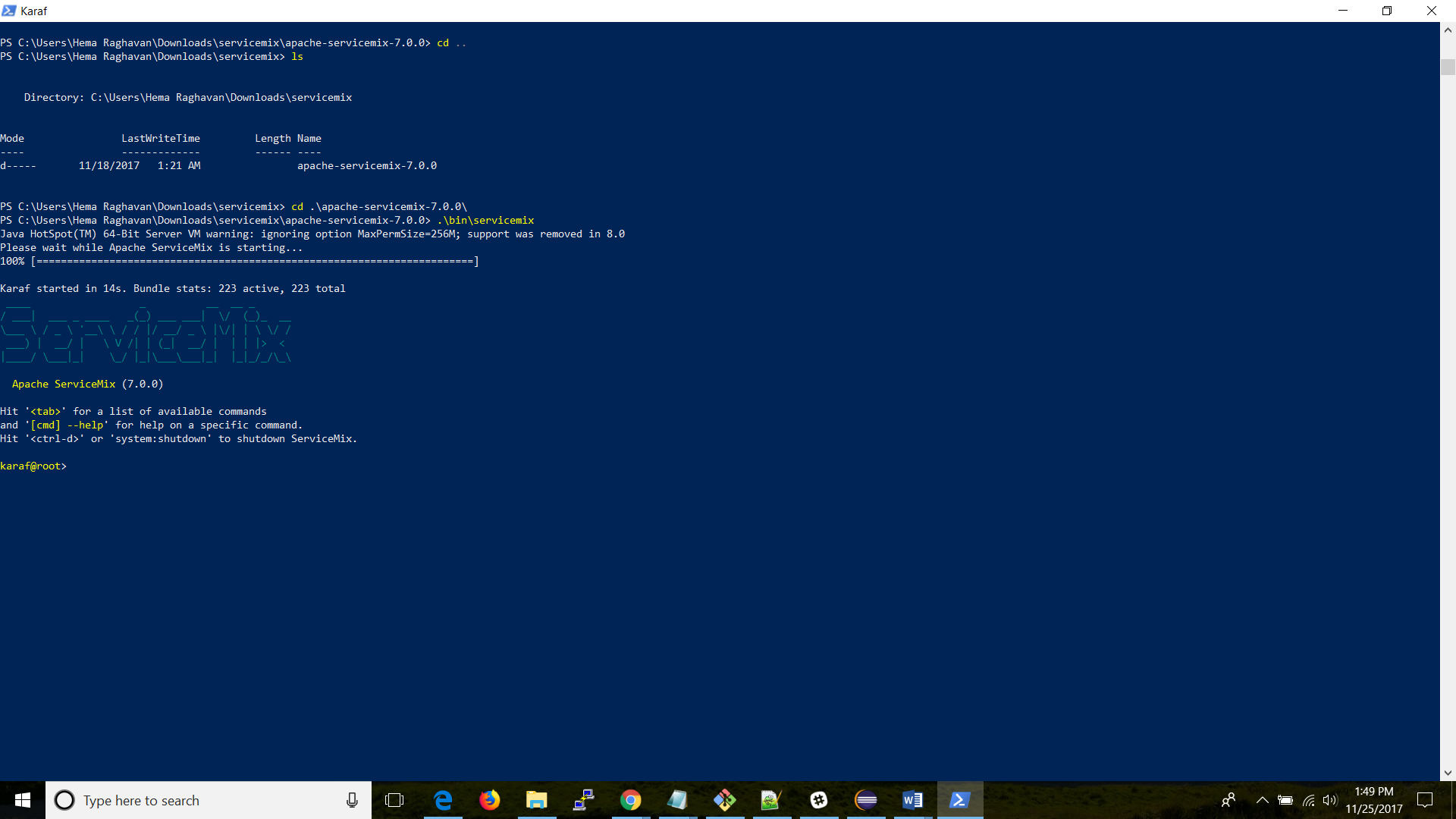
Apache ServiceMix is a flexible, open-source integration container that unifies the features and functionality of Apache ActiveMQ, Camel, CXF, and Karaf into a powerful runtime platform you can use to build your own integrations solutions. It provides a complete, enterprise ready ESB exclusively powered by OSGi.

**Download & Install ServiceMix:**

* Apache ServiceMix 7.0.0-SNAPSHOT is available under the Apache License v2 and can be downloaded from <http://servicemix.apache.org/downloads.html>
* Installing Apache ServiceMix is as simple as uncompressing the downloaded archive on your hard disk

**Starting ServiceMix:**

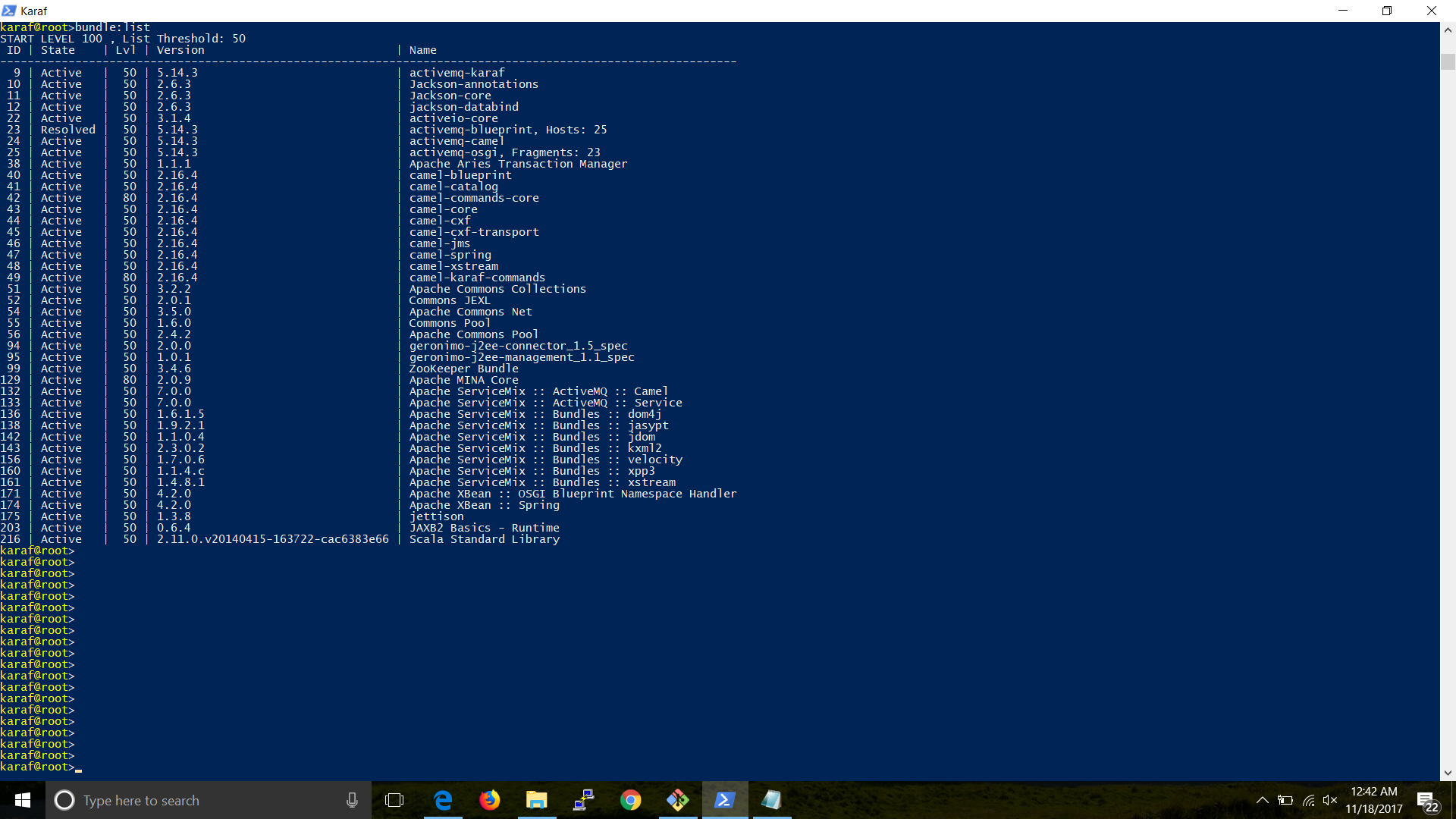
* To start ServiceMix, navigate to the directory where you extracted ServiceMix and run the bin\servicemix.bat file:



**Apache ServiceMix Console:**

The ServiceMix console lets you manage your ServiceMix instance, add and remove bundles, install optional features, etc. We’ll have a look at a few of them in detail below:

* **Working with bundles:**

When ServiceMix is first started, a whole set of bundles providing the core features for the product are being installed. The main information provided by a feature is the set of OSGi bundles that defines the application. Such bundles are URLs pointing to the actual bundle jars. The *bundle:list* command can be used to get a list of all bundles currently installed: 

For every bundle, you see:

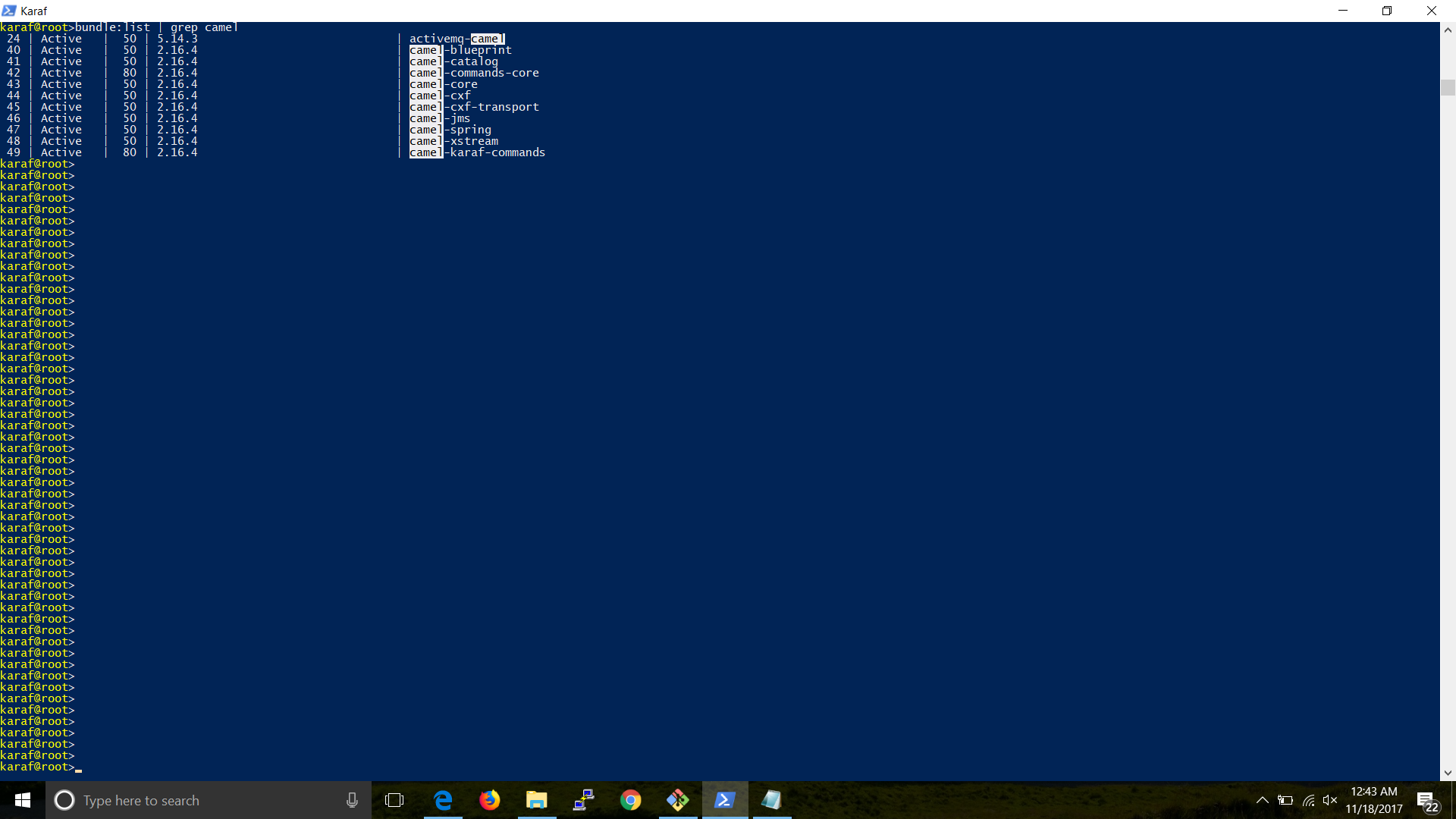
• the bundle id

• the bundle state

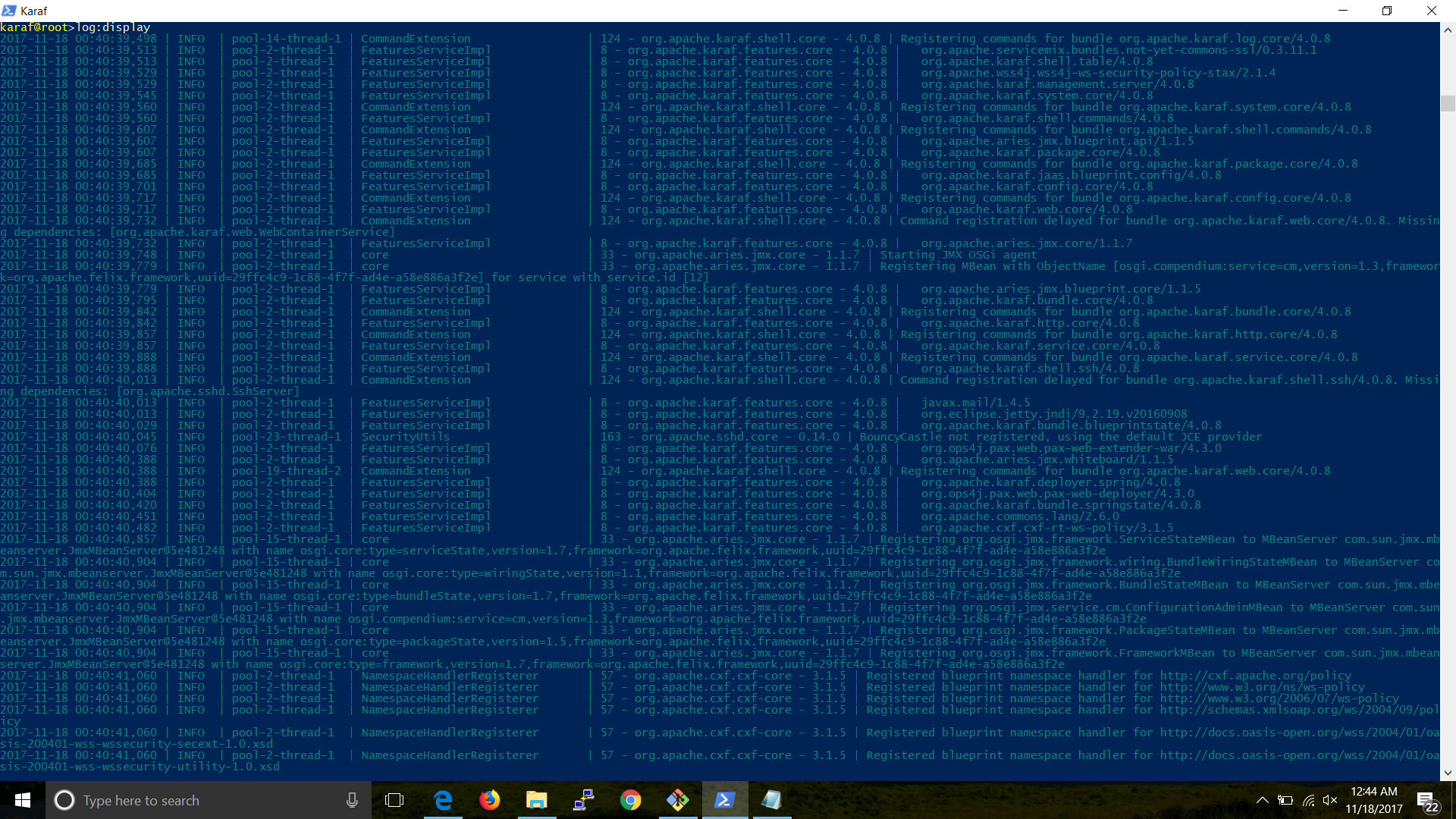
• the bundle start level

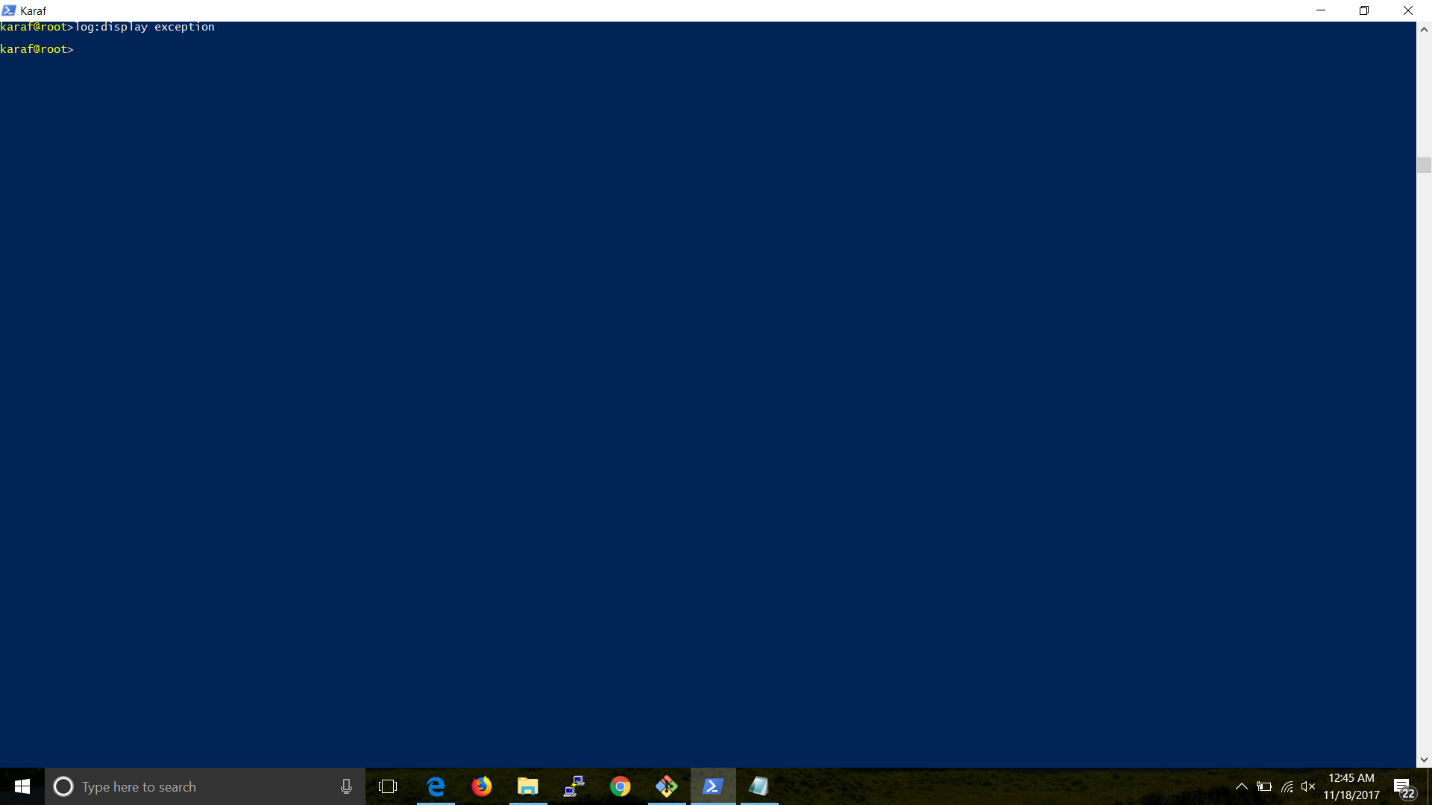
• the bundle name and version

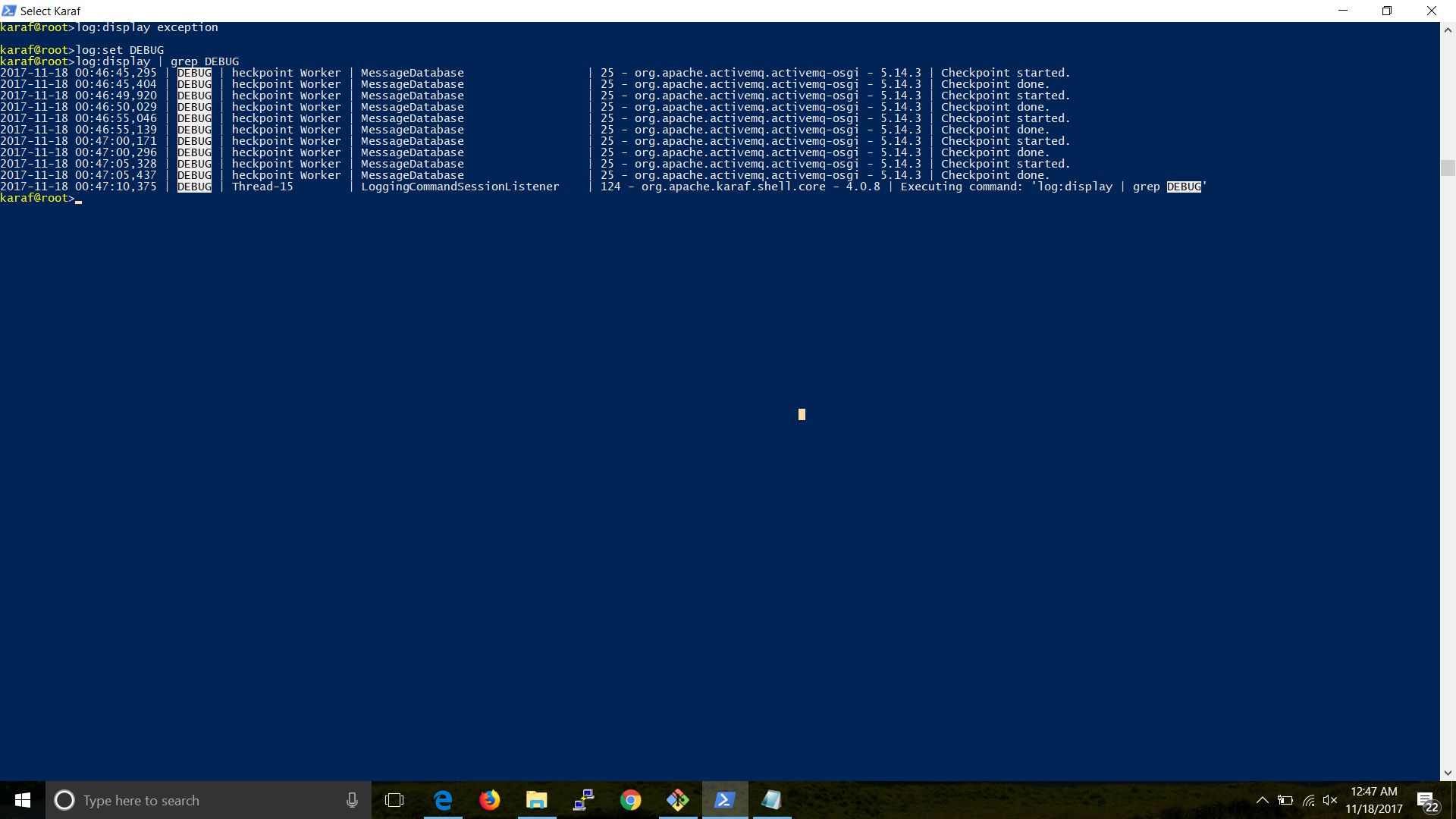
To access something specific, you can use unix-like pipes and utilities to help you. For example: the command *karaf@root> bundle:list | grep camel* will list those bundles related to camel:

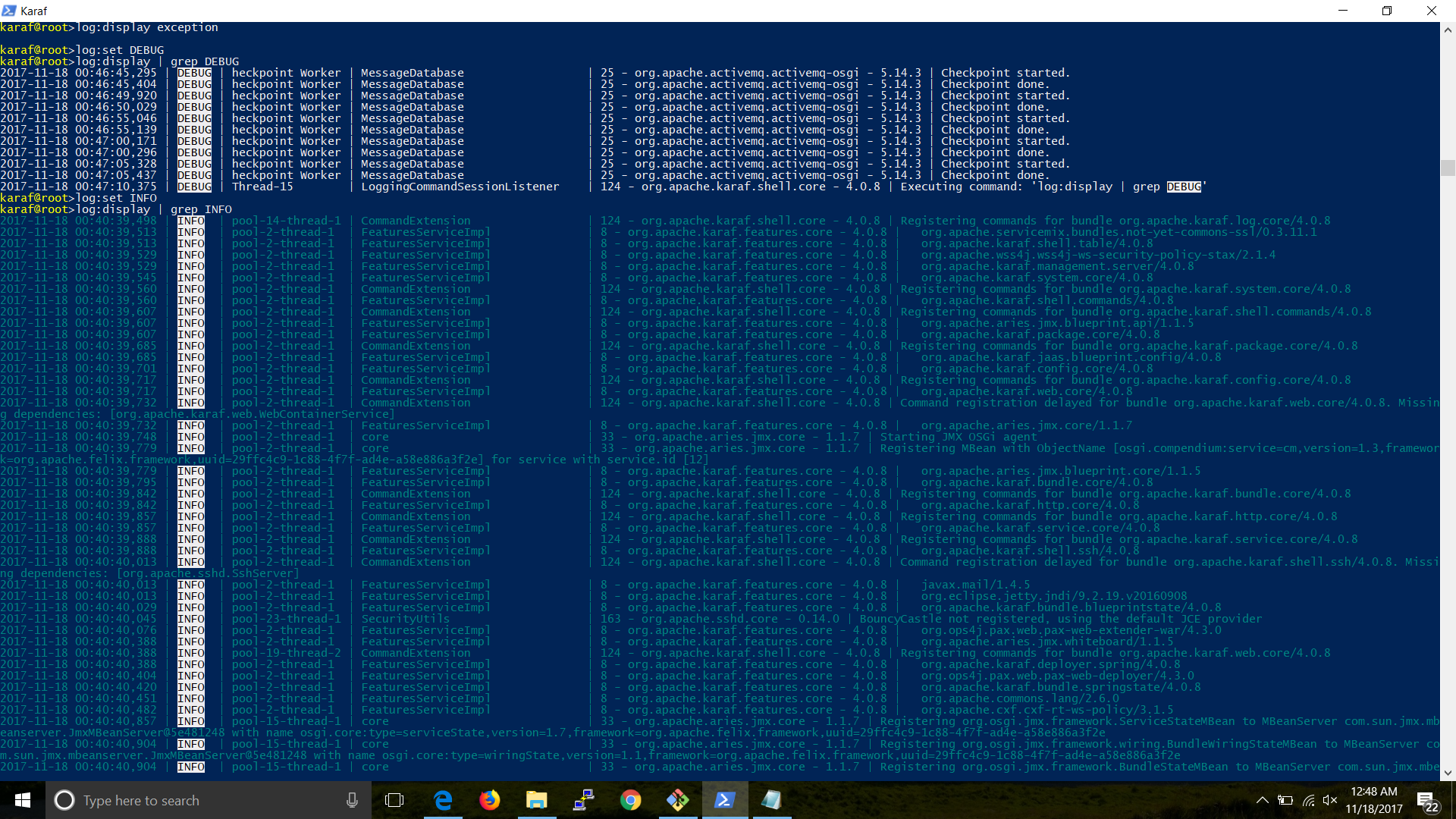


* **Working with logging:**

Many of the applications we write must have some form of log output. To access these messages in the log file, we can use the *log:diplay* command: 

To filter logs for exceptions: (since I dint have any exception in my log, its returned empty)

To set the log level to DEBUG, use the command *log:set DEBUG* and the access the logs:

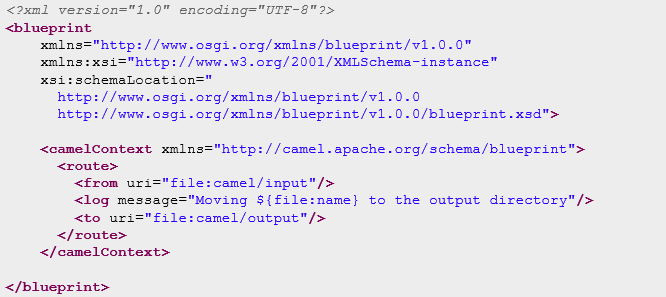
To set the log level to INFO, use the command *log:set INFO*:

**Camel & ServiceMix:**

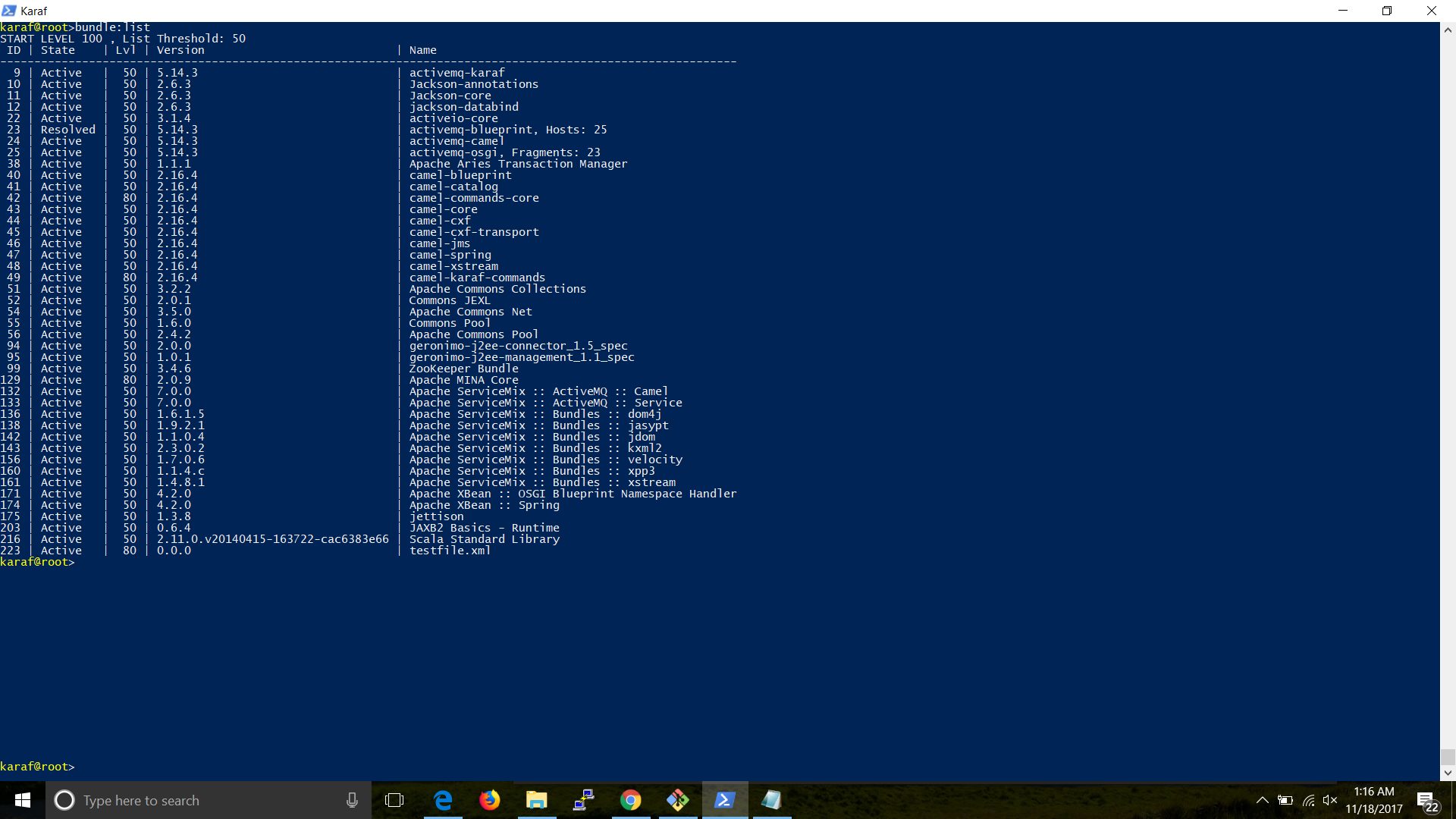
“Camel is smart routing and mediation engine which implements the [Enterprise Integration Patterns](http://camel.apache.org/enterprise-integration-patterns.html) and is designed to be used either inside an ESB like ServiceMix, in a Message Broker like ActiveMQ or in a smart endpoint or web services framework like CXF. ServiceMix is an ESB, a JBI container and an integration platform. So they both address different needs though they are both designed to work great together.” [1]

**Defining routes:**

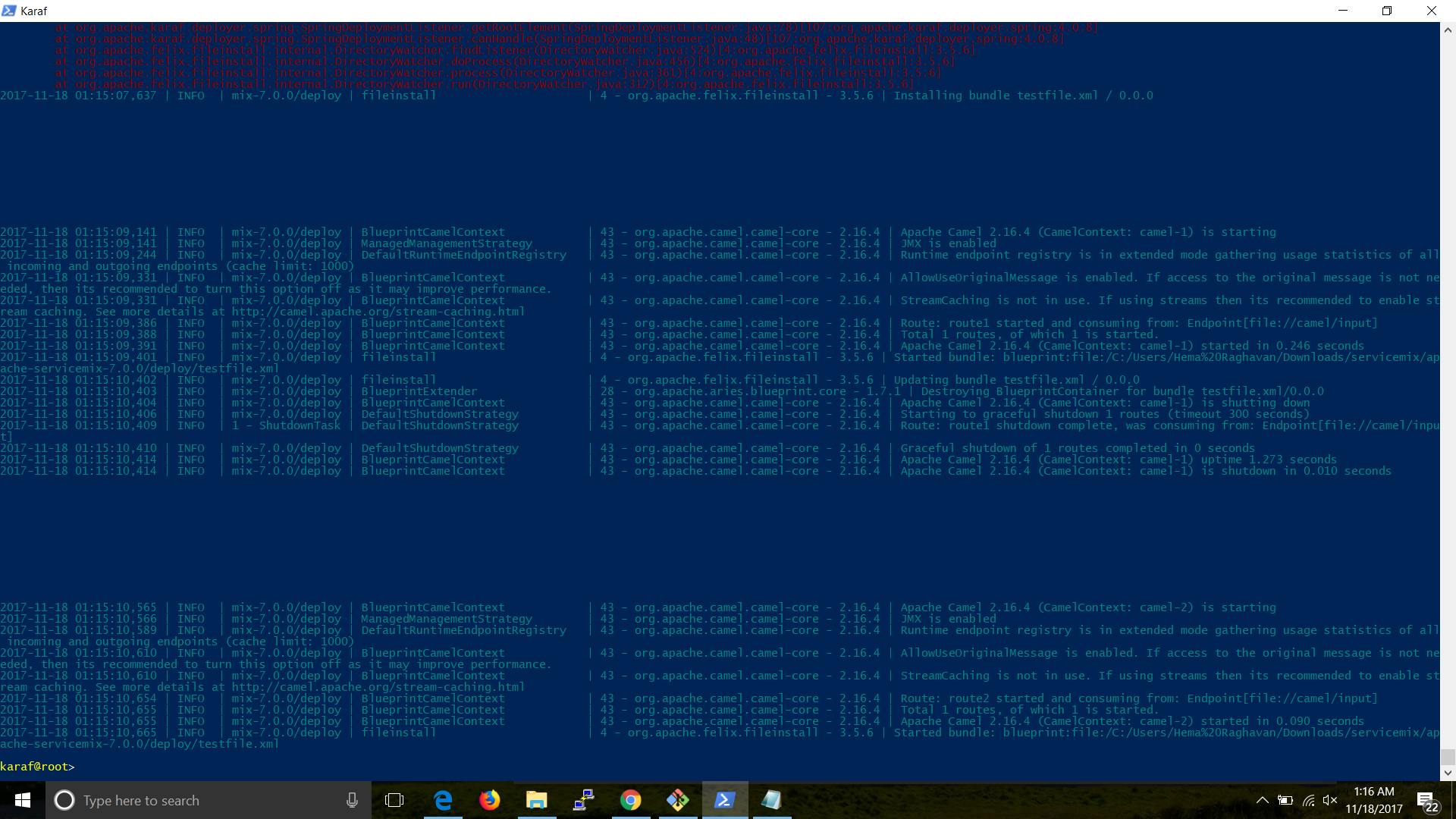
To perform actions like moving files from one directory to another, we can use routes. The easiest way to define a route is using a blueprint xml file, such as below:

****

Using the route mentioned above, we have now directed the file’s input and output path. We can start testing the route, by adding a file to the input path. In our case, we have called the input file “testfile.xml”. Once we’ve added the file, it appears in the *bundle: list* as “**Active**”



Once we have the routes defined, we can have them verified in the logs:

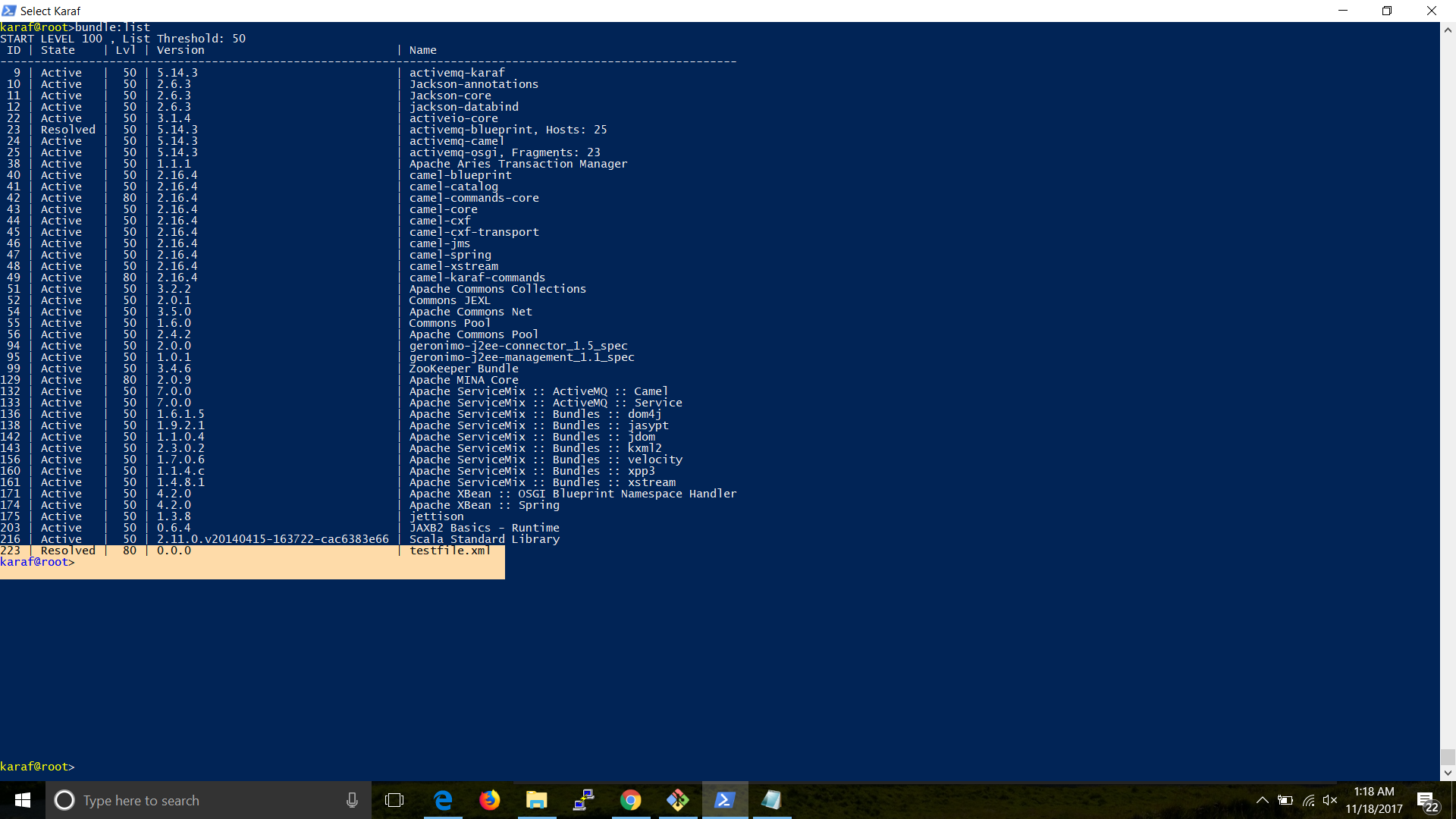


**Accessing ActiveMQ using ServiceMix:**

In this example, we are trying to move files between directories.

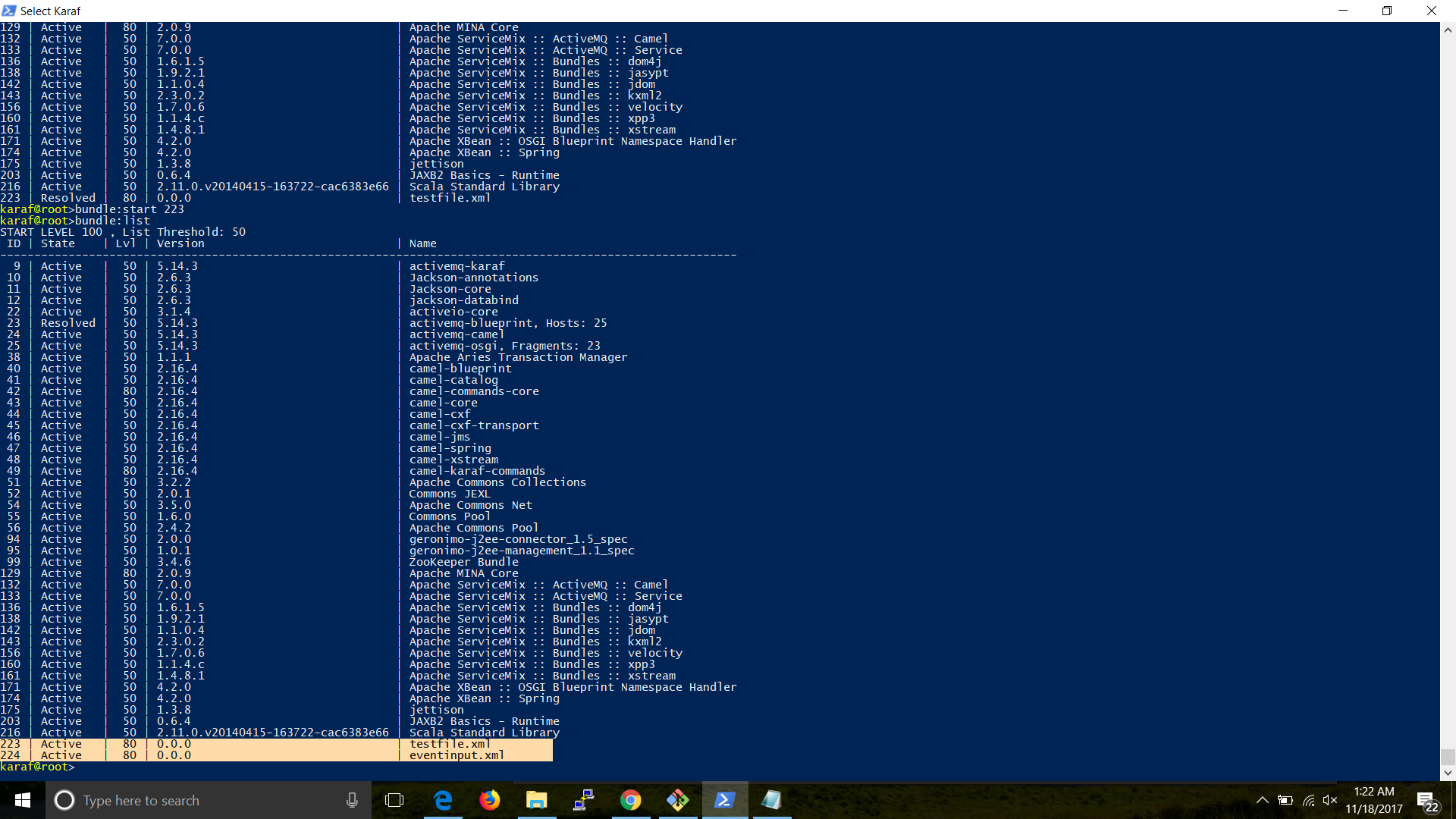
The first Blueprint XML file we'll create contains a Camel route that moves the files from activemq/input to the activemq/output directory. Afterwards, it will generate an event message and send that to an ActiveMQ queue called events. The first Blueprint XML file we'll create contains a Camel route that moves the files from activemq/input to the activemq/output directory.

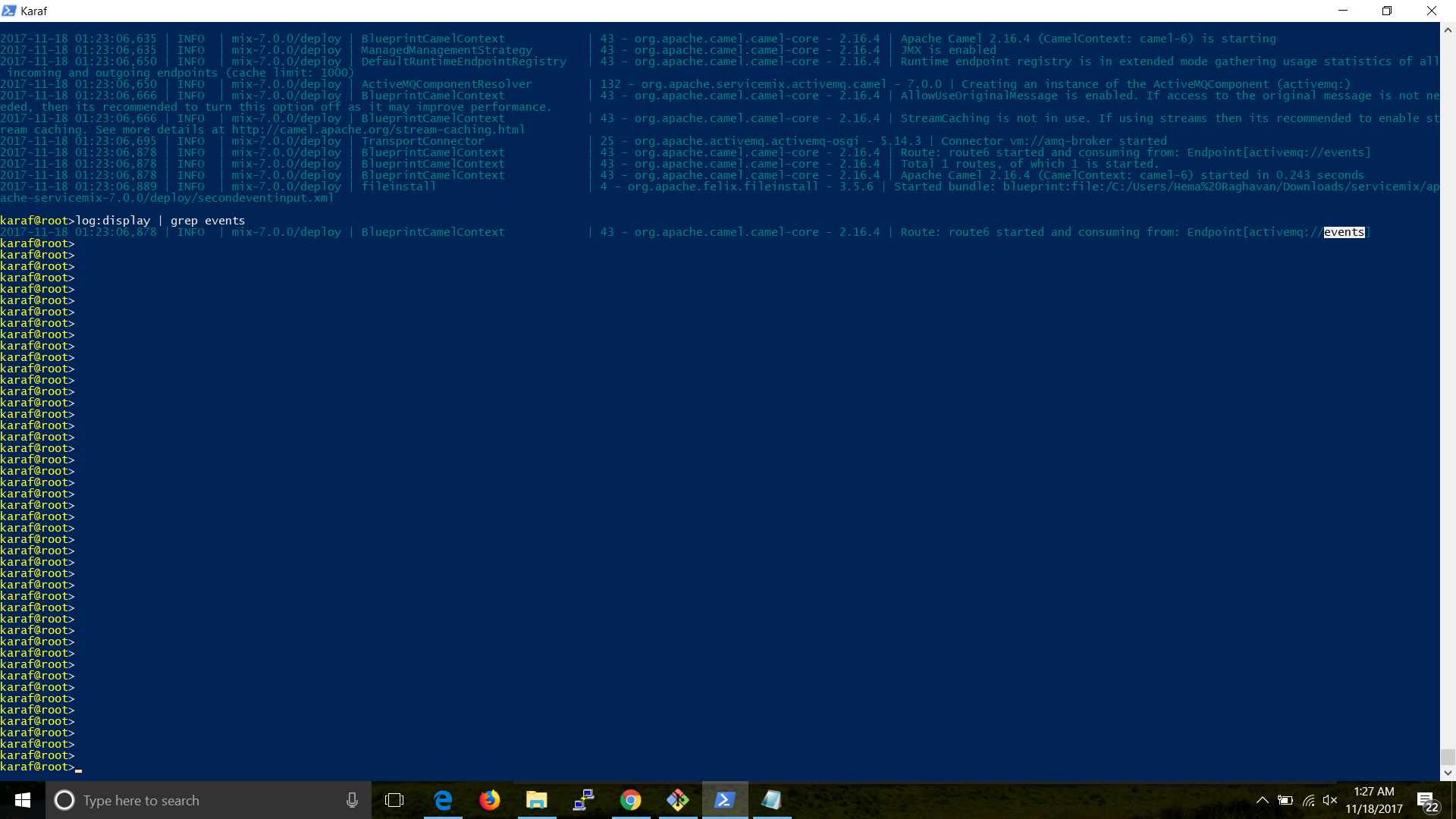


Once we have the route generated, we’ll be able to see it in the logs as: 

Now that we have created a file, and have added the input route, we now have to create a output blueprint file for the output route:



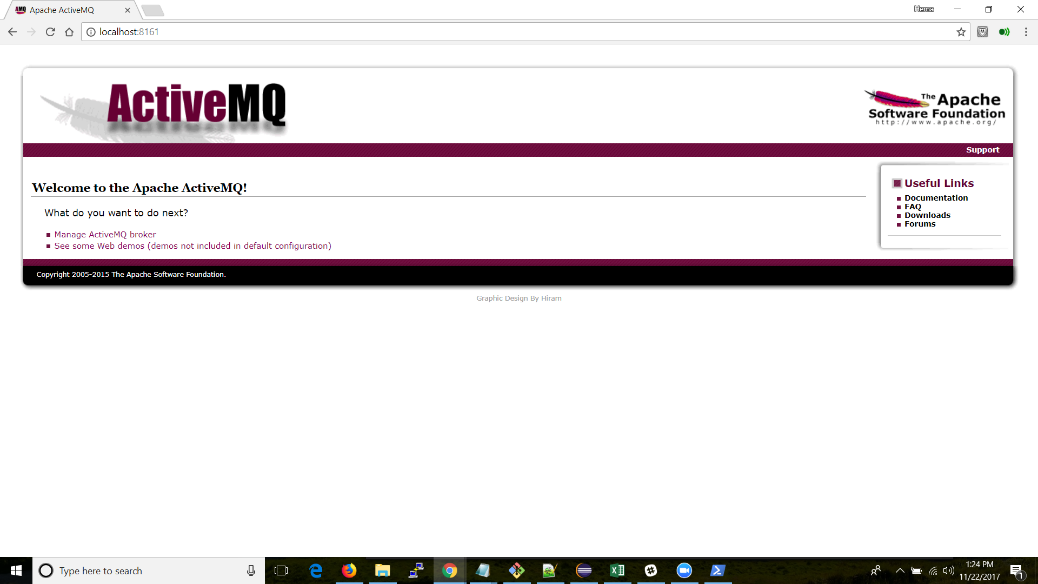
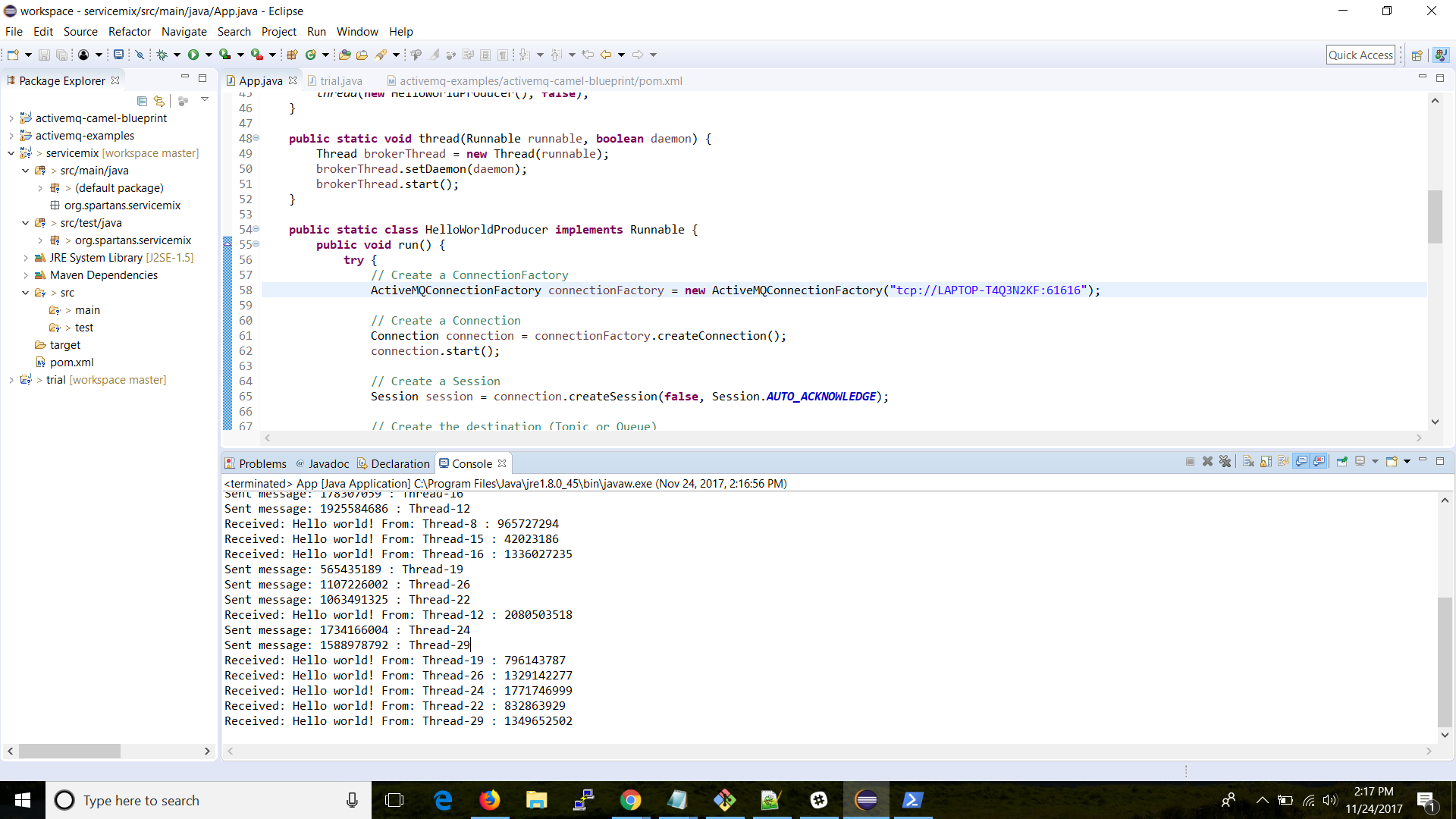
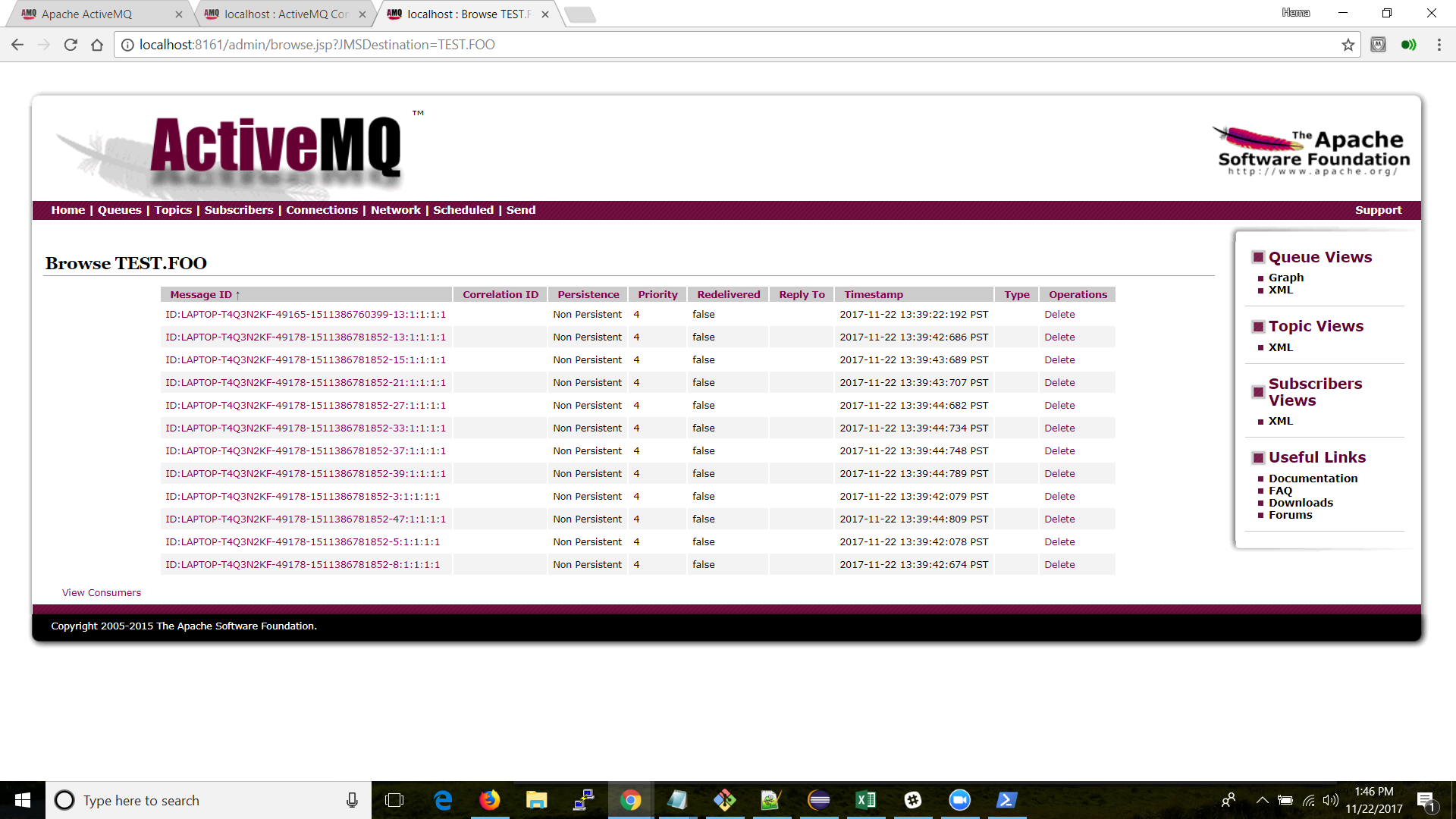
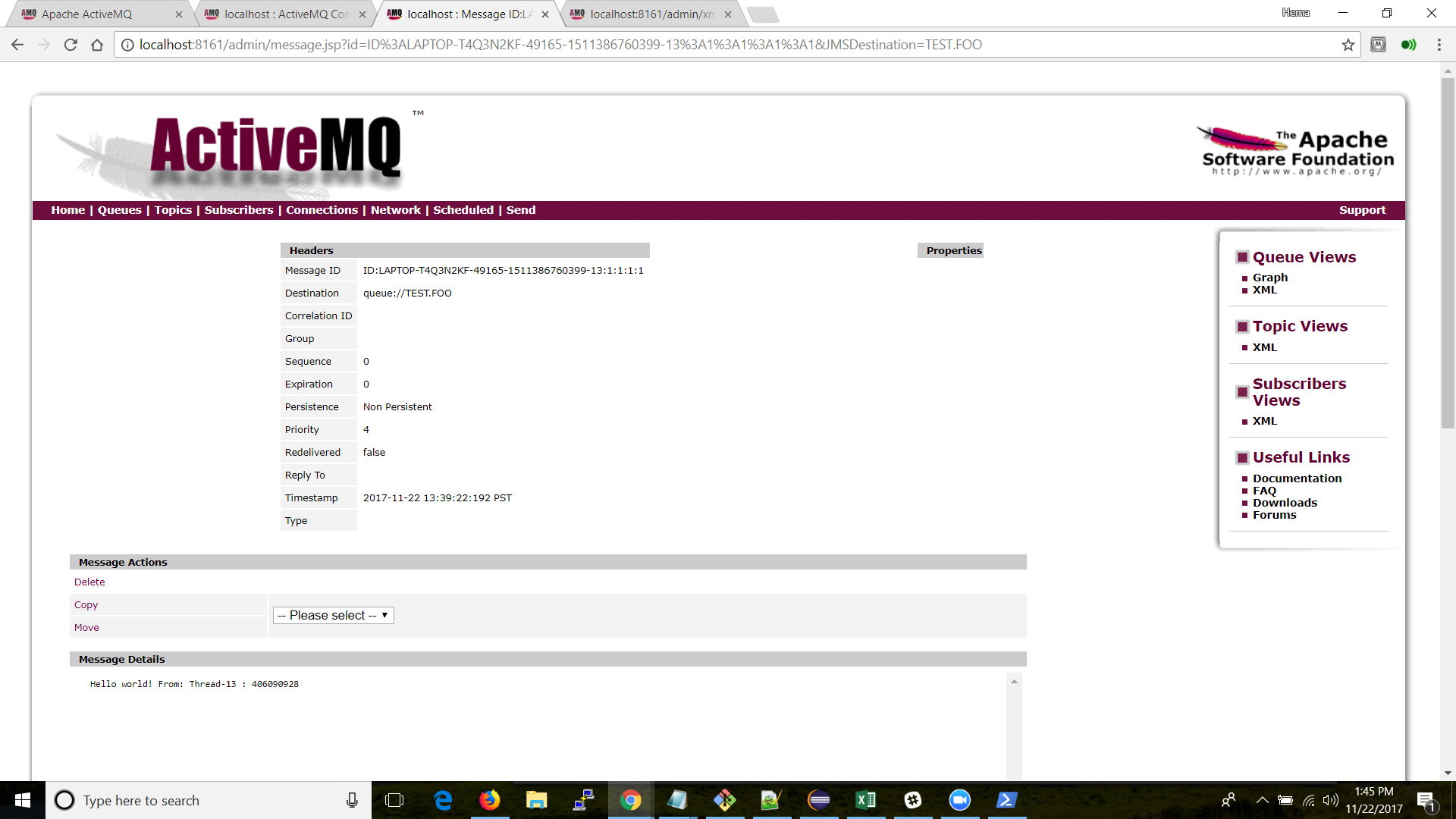
Once we have the output route, we can see from them in the active state in bundle list: 

From the logs, we can see the event and the file transfer:

**Setup a simple JMS application**

Using [ActiveMQ](http://activemq.apache.org/hello-world.html) as reference, we are to build a simple JMS application to display “Hello World”.

**Steps**:

1. Download ActiveMQ and have it started in the command prompt.
2. Once the ActiveMQ is up, verify it in the browser: 
3. With ActiveMQ up and running, open eclipse and set-up the ActiveMQ workspace and update the **app.java** code:
4. On the ActiveMQ portal, We will be able to see the message created using the code as thus: 
5. Each of these message can be viewed by clicking on the Message IDs: 

**References**:

1. <http://camel.apache.org/how-does-camel-compare-to-servicemix.html>
2. <http://servicemix.apache.org/docs/7.x/quickstart/quickstart.pdf>