# Adquisición de Datos - Práctica Final

[Adquisición de Datos - Práctica Final](#_wbgmvcx8ecoj)

[Ejercicio 1](#_e22ooh26oihz)

[Pasos Previos](#_68b4dr71gdaw)

[Ejercicio](#_e5734s813x2x)

[Resultado esperado](#_851xdoo5pxf9)

[Ejercicio 2](#_5drqm8m726b5)

[Ejercicio](#_tvj52jr0hu8d)

[Resultado esperado](#_4mos8nimkp8l)

[Ejercicio 3](#_ejdviu63brtk)

[Resultado esperado](#_ocony69ptqeh)

# Ejercicio 1

En este ejercicio vamos a utilizar Flume para consumir eventos syslog y almacenarlos en HDFS:



## Pasos Previos

* Ejecutar el siguiente comando para evitar que ejecuciones de anteriores ejercicios afecten a este:

rm -rf ~/.flume/file-channel/\*

* Ejecutar el siguiente comando para descargar un pequeño script que utilizaremos en este ejercicio:

wget https://gist.github.com/fjavieralba/ab129e84241623916acd/raw/fd0e221e22a6355453a18b213c74e386c58d9a5f/syslog\_generator.py

## Ejercicio

1. Crear un nuevo fichero de configuración de Flume (en el directorio /etc/flume/conf/) que tenga como source eventos Syslog en el puerto UDP 5140 y como destino el directorio /tmp/flume/syslog de HDFS

**Nota**: Lo más fácil es copiar una de las configuraciones utilizadas en ejercicios de clase (por ejemplo, spooldir-to-hdfs.properties) y modificarla convenientemente.

Se recomienda consultar [http://flume.apache.org/FlumeUserGuide.html#syslog-udp-source](http://flume.apache.org/FlumeUserGuide.html) para ver los parámetros de configuración necesarios y un ejemplo.

1. Ejecutar el agente Flume con la configuración creada. Se pueden revisar las diapositivas o ejercicios de Flume para recordar el comando a utilizar. Recuerda que tendrás que utilizar el “*command*” agent y los parámetros “--conf”, “--conf-file” y “--name” para especificar el nombre del agente Flume
2. En otra consola, ir al directorio ﻿donde se descargó el script syslog\_generator.py y ejecutar el siguiente comando, para que se generen eventos syslog:

python syslog\_generator.py <TU\_NOMBRE>

Por ejemplo:

python syslog\_generator.py Javier

## Resultado esperado

Se deberían crear en HDFS varios ficheros que contengan los mensajes syslog generados (que deberán incluir tu nombre).

Puedes comprobar si se han generado los ficheros en HDFS consultando la interfaz web de HUE ([http://localhost:8000/filebrowser/#/tmp](http://localhost:8000/filebrowser/%23/tmp))

Los ficheros resultado se escribirán en el directorio HDFS que hayas especificado en la configuración del sink en tu configuración de Flume (se recomienda /tmp/flume/syslog).

Si no ves ningún fichero generado, revisa el fichero de configuración, por si te faltara algo, y revisa también el log de flume (/var/log/flume/flume.log), ya que si encuentra algún error lo reflejará allí.

Recuerda que en tu solución deberás indicar el fichero de configuración que has utilizado y los comandos que has ejecutado, así como una foto (**sólo una**) de tu ordenador con el resultado final (tienes más instrucciones de entrega en [este documento](https://docs.google.com/a/campusciff.net/document/d/11rFvdW0H-G1sq_i3AoKkZqzKMh09nKqVFXVqBIRWd8w/edit?usp=sharing))

## Solucion

Creamos el fichero syslog-to.hdfs.conf:

agent1.sources = syslog1

agent1.channels = syslogchannel1

agent1.channels.syslogchannel1.type = file

agent1.sources.syslog1.type = syslogudp

agent1.sources.syslog1.port = 5140

agent1.sources.syslog1.host = localhost

agent1.sources.syslog1.channels = syslogchannel1

agent1.sinks = syssink1

agent1.sinks.syssink1.channel = syslogchannel1

agent1.sinks.syssink1.type = hdfs

agent1.sinks.syssink1.hdfs.path = /tmp/flume/ejercicio1

agent1.sinks.syssink1.hdfs.filePrefix = syslog

agent1.sinks.syssink1.hdfs.inUsePrefix = \_

agent1.sinks.syssink1.hdfs.fileType = DataStream

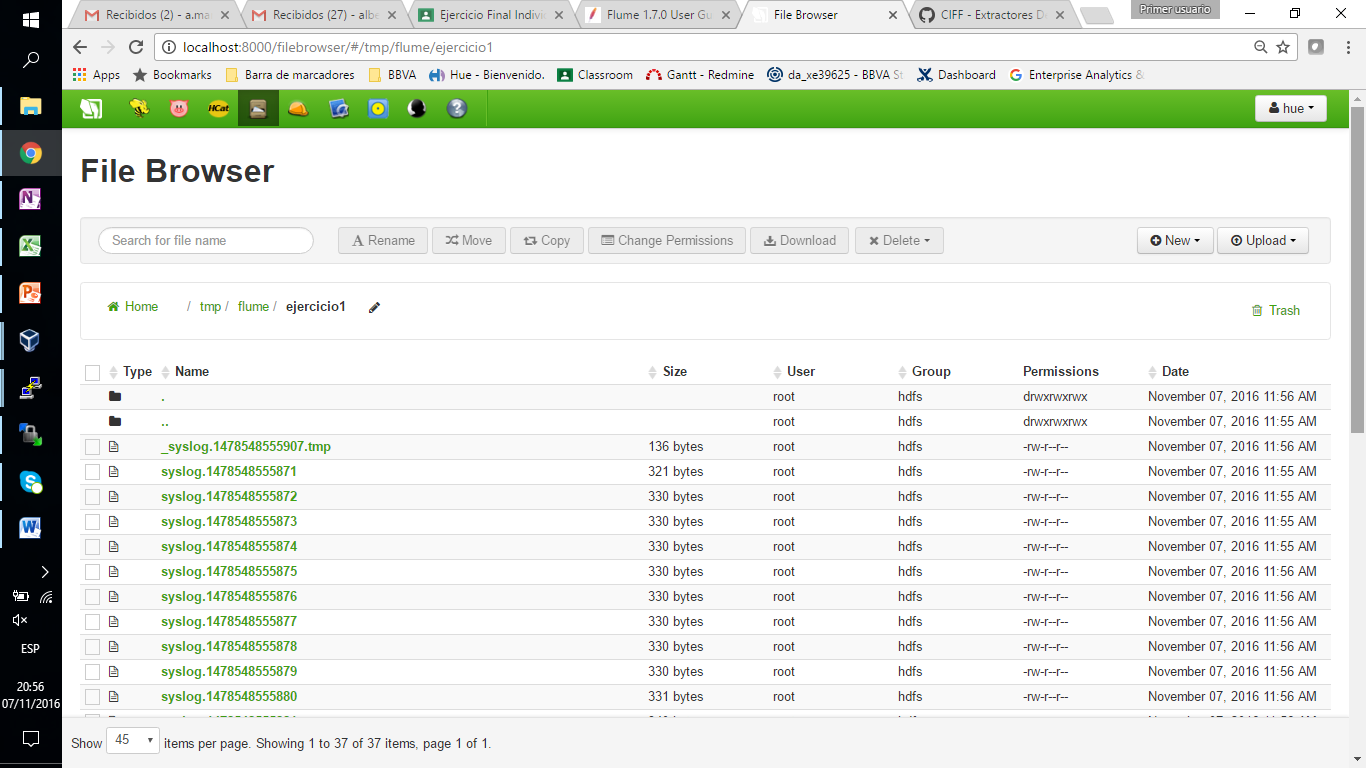
Ejecutamos el agente de flume:

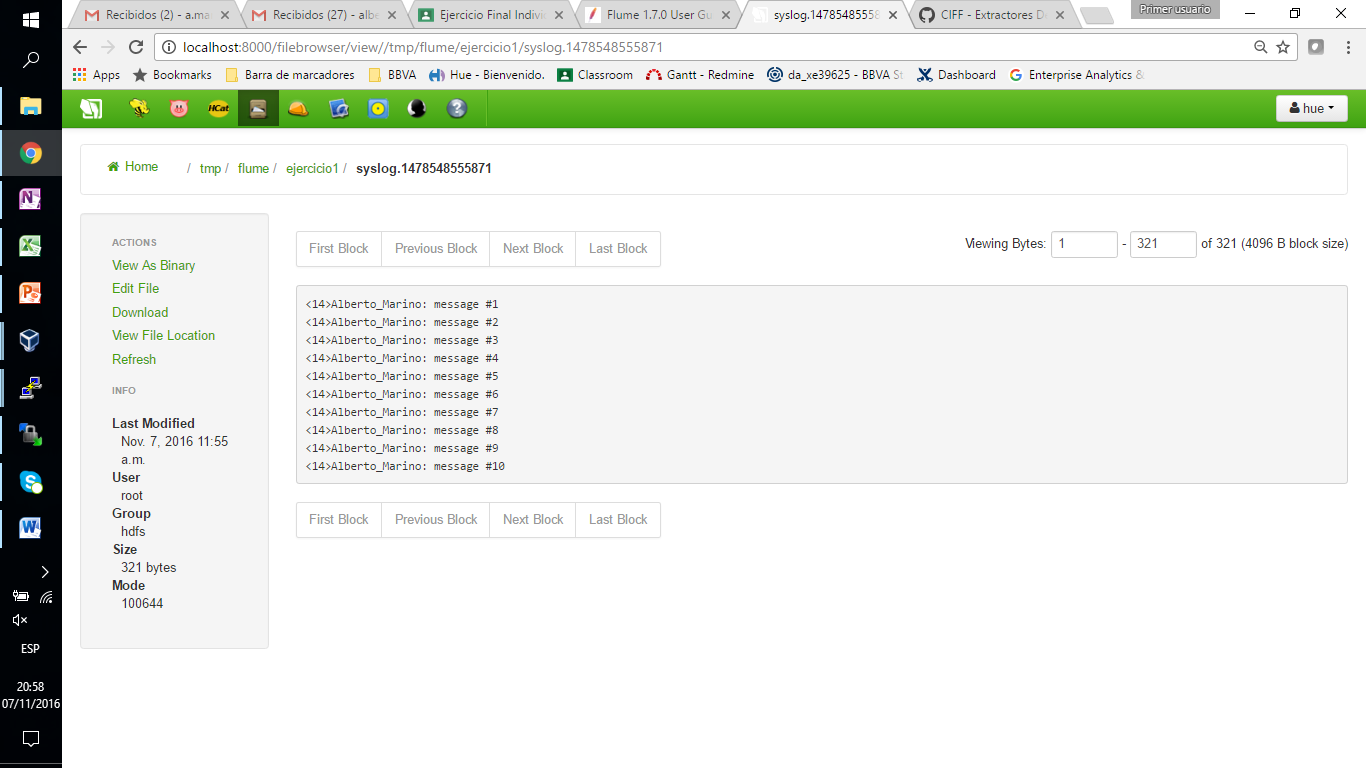
flume-ng agent --conf-file /etc/flume/conf/ syslog-to-hdfs.properties --name agent1 --conf /etc/flume/conf/ -Dflume.root.logger=INFO,console

A continuacion ejecutamos el script bajado de syslog:

python syslog\_generator.py Alberto\_Marino

Revisamos vía File Browser de HDFS la ruta configurada para ver que están precisamente los eventos generados.





# Ejercicio 2

En este ejercicio vamos a comprobar que Kafka nos proporciona tolerancia a fallos en los nodos de nuestro cluster.

## Ejercicio

1. Inicia un cluster de Kafka que consista en 3 brokers (recuerda que tendrás que usar un fichero de configuración diferente para cada uno, al igual que hicimos en los ejercicios de clase)
2. Crea un topic “practica” que tenga 3 particiones y factor de replicación 3
3. Usa la opción --describe del comando kafka-topics.sh para comprobar que el tópico practica que acabas de crear tiene 3 particiones
4. Usando el console-producer que incluye Kafka (y que usamos en algunos ejercicios de clase), inserta algunos mensajes en el topic, asegúrate de que uno de los mensajes sea tu nombre y apellidos. Una vez publicados los mensajes, ya puedes matar al console-producer. (Usando las teclas Ctrl+C)
5. Usando el console-consumer que incluye Kafka, consume **todos los mensajes** del topic practica
6. Mata uno de los brokers (Usando Ctrl+C)
7. Vuelve a lanzar el console-consumer y consume todos los mensajes del topic practica
8. Mata otro broker (Usando Ctrl+C)
9. Vuelve a lanzar el console-consumer y consume todos los mensajes del topic practica

## Resultado esperado

El resultado esperado en este ejercicio es comprobar que con un factor de replicación 3, en un cluster de 3 brokers, podemos perder 2 de ellos sin pérdida de datos.

Recuerda que en tu solución deberás indicar los comandos que has ejecutado y su salida, así como una foto (sólo una) de tu ordenador con el resultado final (tienes más instrucciones de entrega en [este documento](https://docs.google.com/a/campusciff.net/document/d/11rFvdW0H-G1sq_i3AoKkZqzKMh09nKqVFXVqBIRWd8w/edit?usp=sharing))

## Solucion

Creamos 3 ficheros server.properties con propiedades port y log\_dir y bróker.id diferentes para que arranquen los 3 demonios.

cd /usr/hdp/current/kafka-broker/

bin/kafka-server-start.sh config/server1.properties

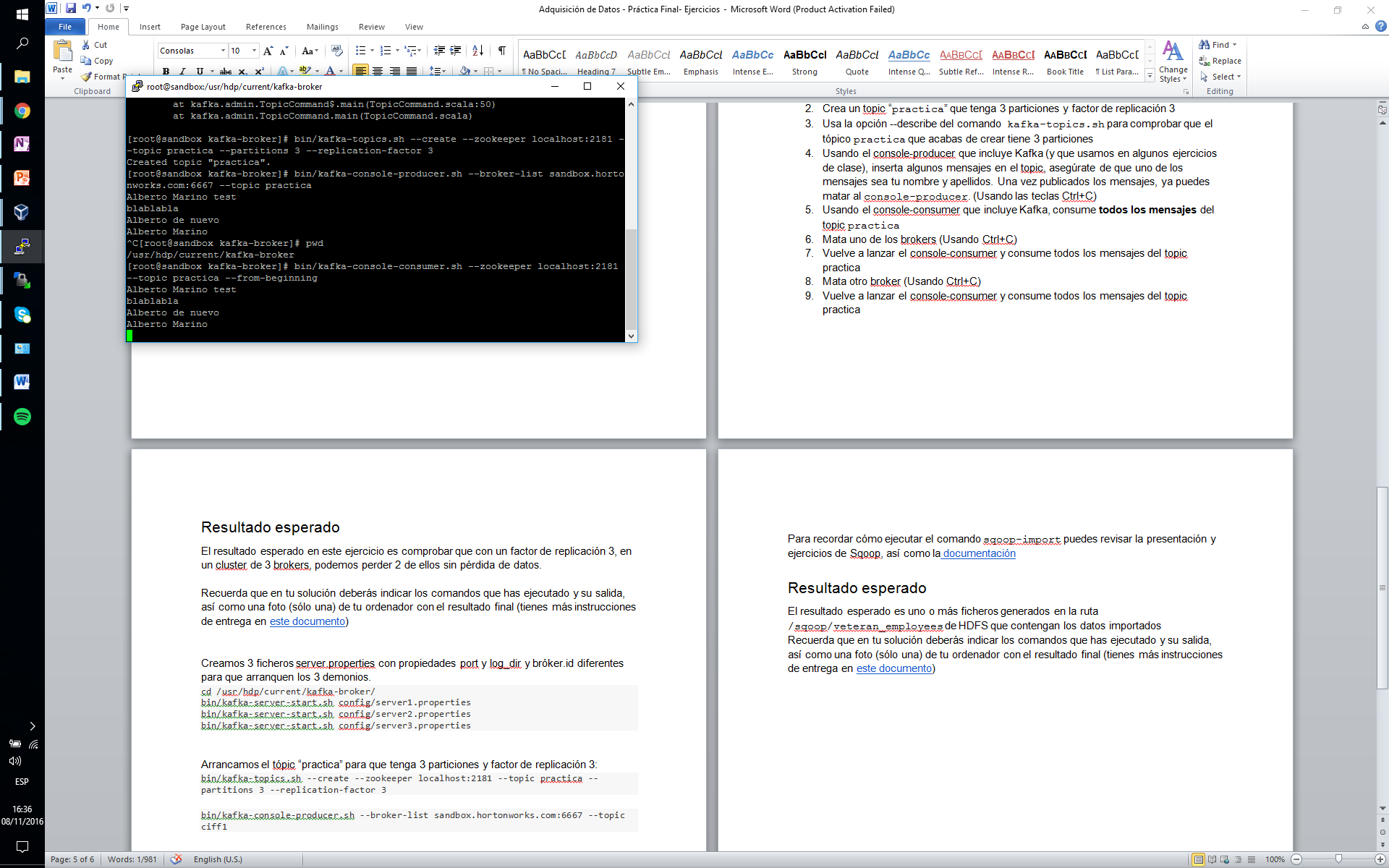
bin/kafka-server-start.sh config/server2.properties

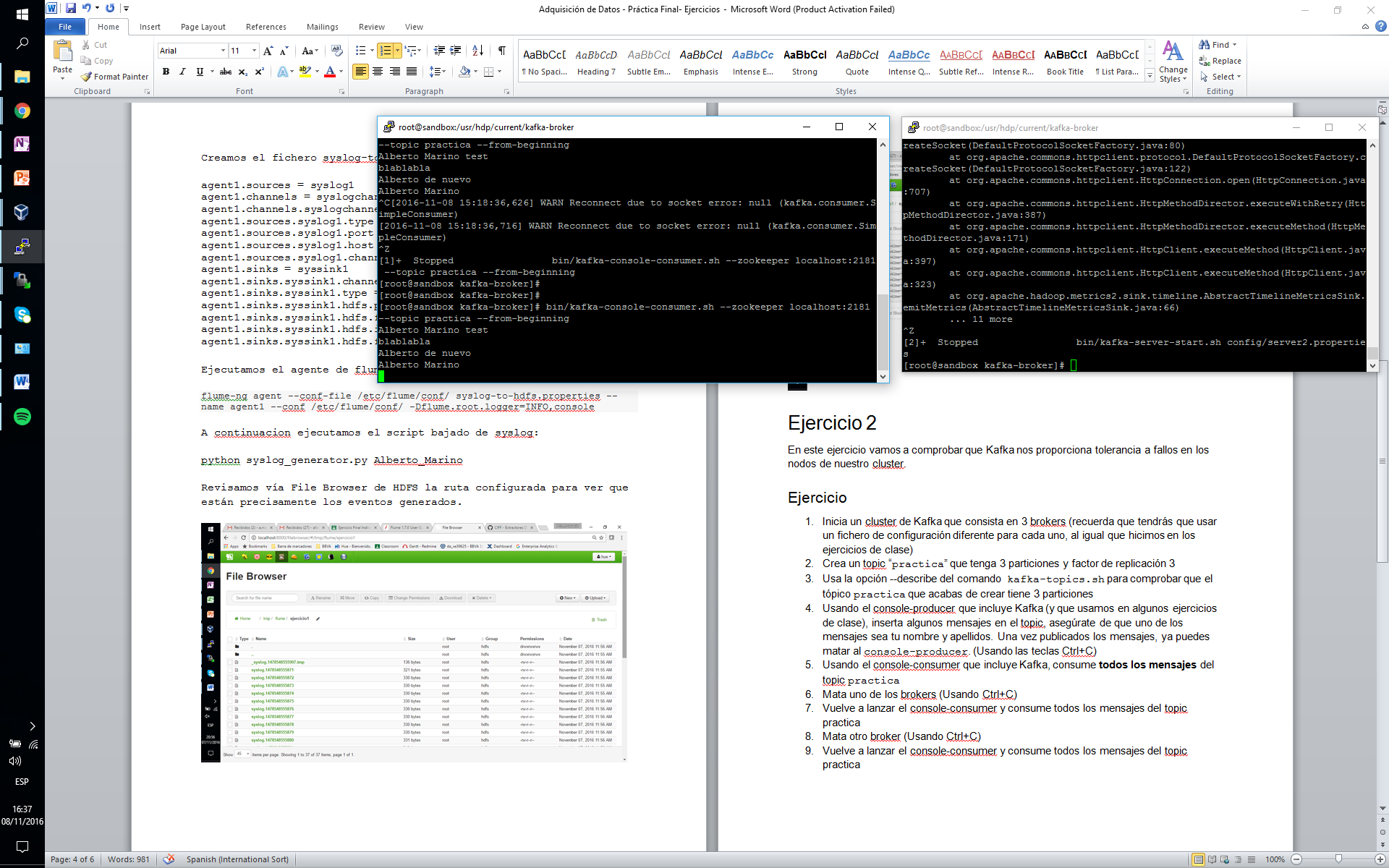
bin/kafka-server-start.sh config/server3.properties

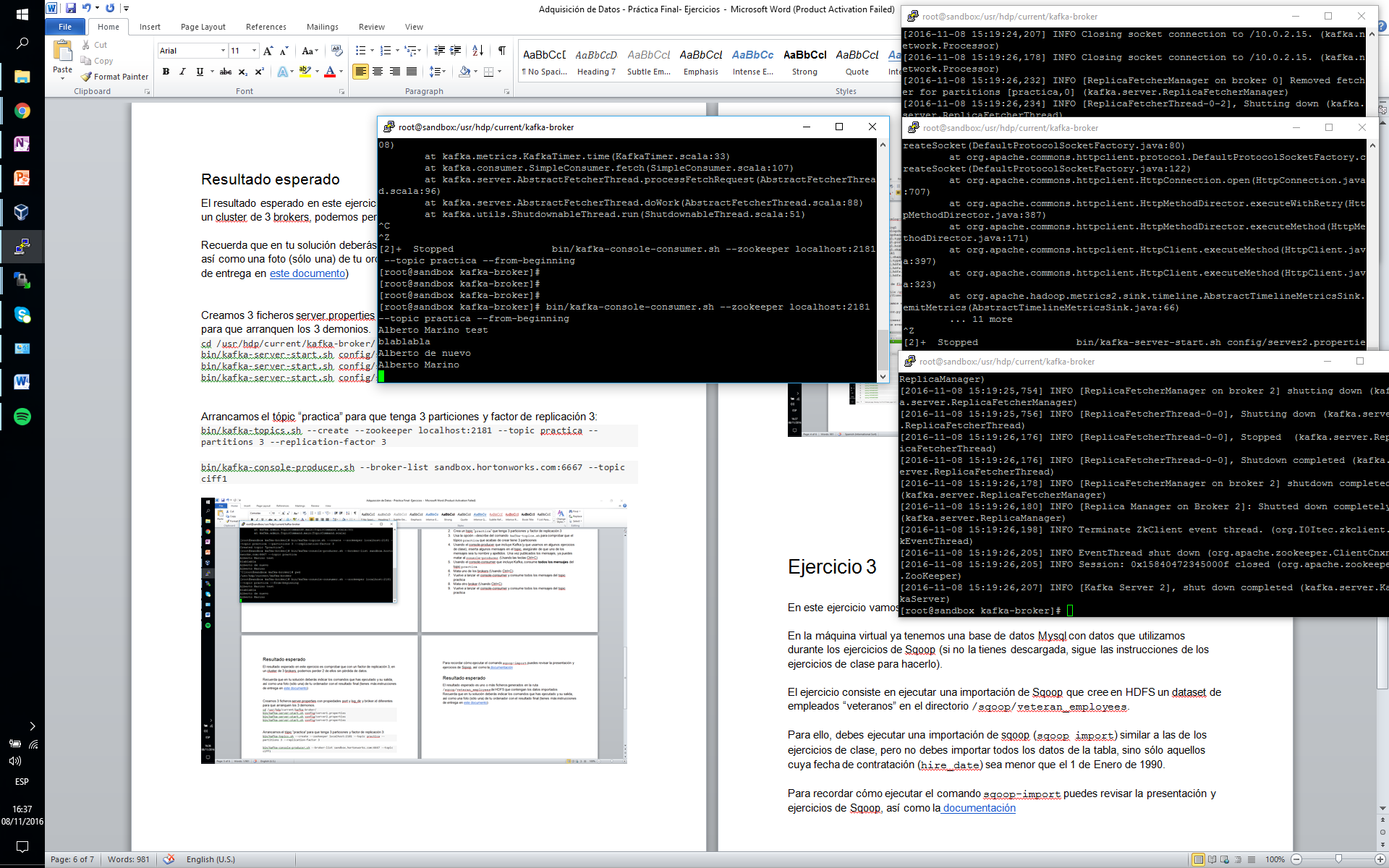
Arrancamos el tópic “practica” para que tenga 3 particiones y factor de replicación 3:

bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic practica --partitions 3 --replication-factor 3

bin/kafka-console-producer.sh --broker-list sandbox.hortonworks.com:6667 --topic ciff1







# Ejercicio 3

En este ejercicio vamos a utilizar Sqoop para importar datos de una tabla de Mysql a HDFS.

En la máquina virtual ya tenemos una base de datos Mysql con datos que utilizamos durante los ejercicios de Sqoop (si no la tienes descargada, sigue las instrucciones de los ejercicios de clase para hacerlo).

El ejercicio consiste en ejecutar una importación de Sqoop que cree en HDFS un dataset de empleados “veteranos” en el directorio /sqoop/veteran\_employees.

Para ello, debes ejecutar una importación de sqoop (sqoop import) similar a las de los ejercicios de clase, pero no debes importar todos los datos de la tabla, sino sólo aquellos cuya fecha de contratación (hire\_date) sea menor que el 1 de Enero de 1990.

Para recordar cómo ejecutar el comando sqoop-import puedes revisar la presentación y ejercicios de Sqoop, así como la [documentación](http://sqoop.apache.org/docs/1.4.5/SqoopUserGuide.html)

## Resultado esperado

El resultado esperado es uno o más ficheros generados en la ruta /sqoop/veteran\_employees de HDFS que contengan los datos importados

Recuerda que en tu solución deberás indicar los comandos que has ejecutado y su salida, así como una foto (sólo una) de tu ordenador con el resultado final (tienes más instrucciones de entrega en [este documento](https://docs.google.com/a/campusciff.net/document/d/11rFvdW0H-G1sq_i3AoKkZqzKMh09nKqVFXVqBIRWd8w/edit?usp=sharing))

## Solucion

sqoop import --connect jdbc:mysql://localhost/employees --table employees --where " hire\_date < 1990-01-01" --fields-terminated-by "\t" --warehouse-dir /sqoop/veteran\_employees

