Universitat Oberta de Catalunya

Big Data y Seguridad

Carlos de la Cruz Pinto

PFM: Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones

Contenidos

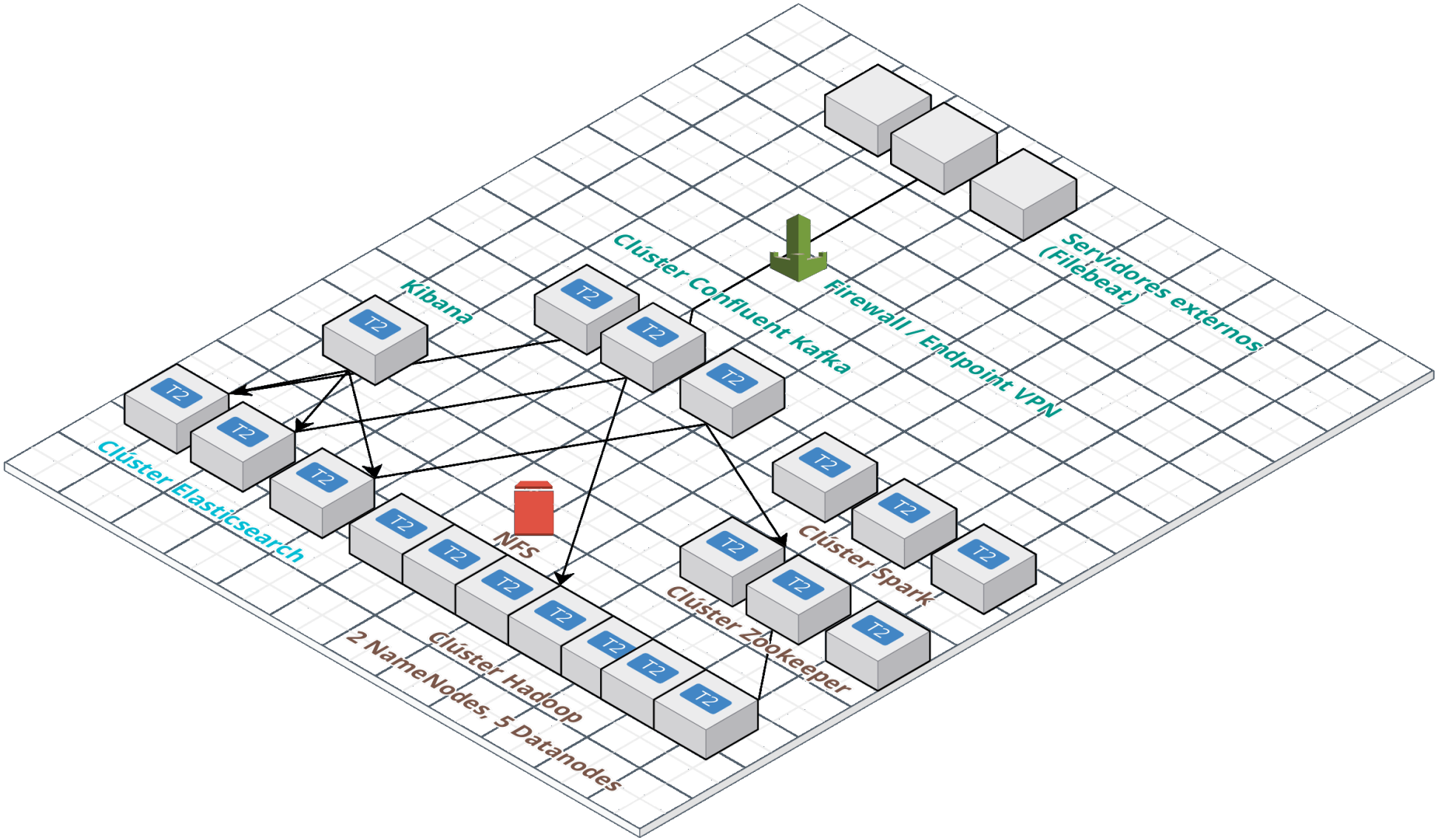
Universitat Oberta de Catalunya

- Supuesto trabajado
- Arquitectura propuesta
- Herramientas que componen la arquitectura
- Detalles de implementación
- Prueba de concepto vs. Implementación final
- Posibles futuras mejoras

Supuesto trabajado

- Gran volumen de información de seguridad generada en una red con alrededor de 3000 servidores en notarías en toda España.
- Necesidad de garantizar confidencialidad e integridad de los datos en los servidores.
- Se desea un conjunto de herramientas, elementos hardware y configuraciones que permitan realizar un análisis de los datos de seguridad en tiempo real y de forma escalable.
- Se desea crear un data lake en el que se permita almacenar la información a largo plazo.

Vista general de la arquitectura



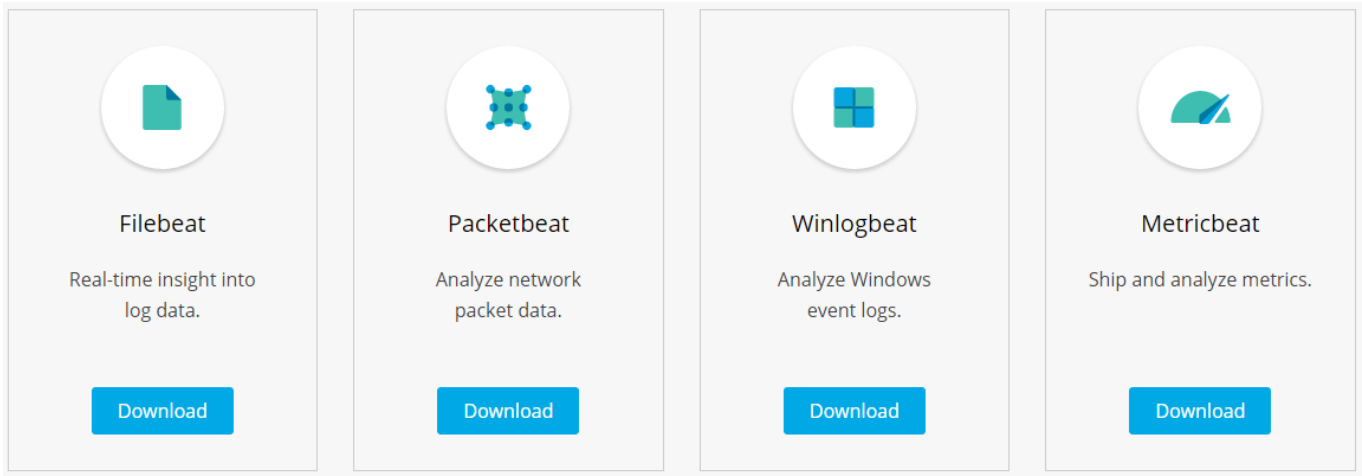
Subsistemas / Herramientas

Universitat Oberta de Catalunya

Extracción de los datos

Universitat Oberta de Catalunya

- Tradicionalmente, se ha venido usando rsyslogd y herramientas collectd
- Elastic proporciona agentes propios con su plataforma ‘beats’. Estos agentes son Open Source y de libre disposición.
- Integrables con Apache Kafka directamente.



Capa de streaming/buffering

Universitat Oberta de Catalunya

Redis

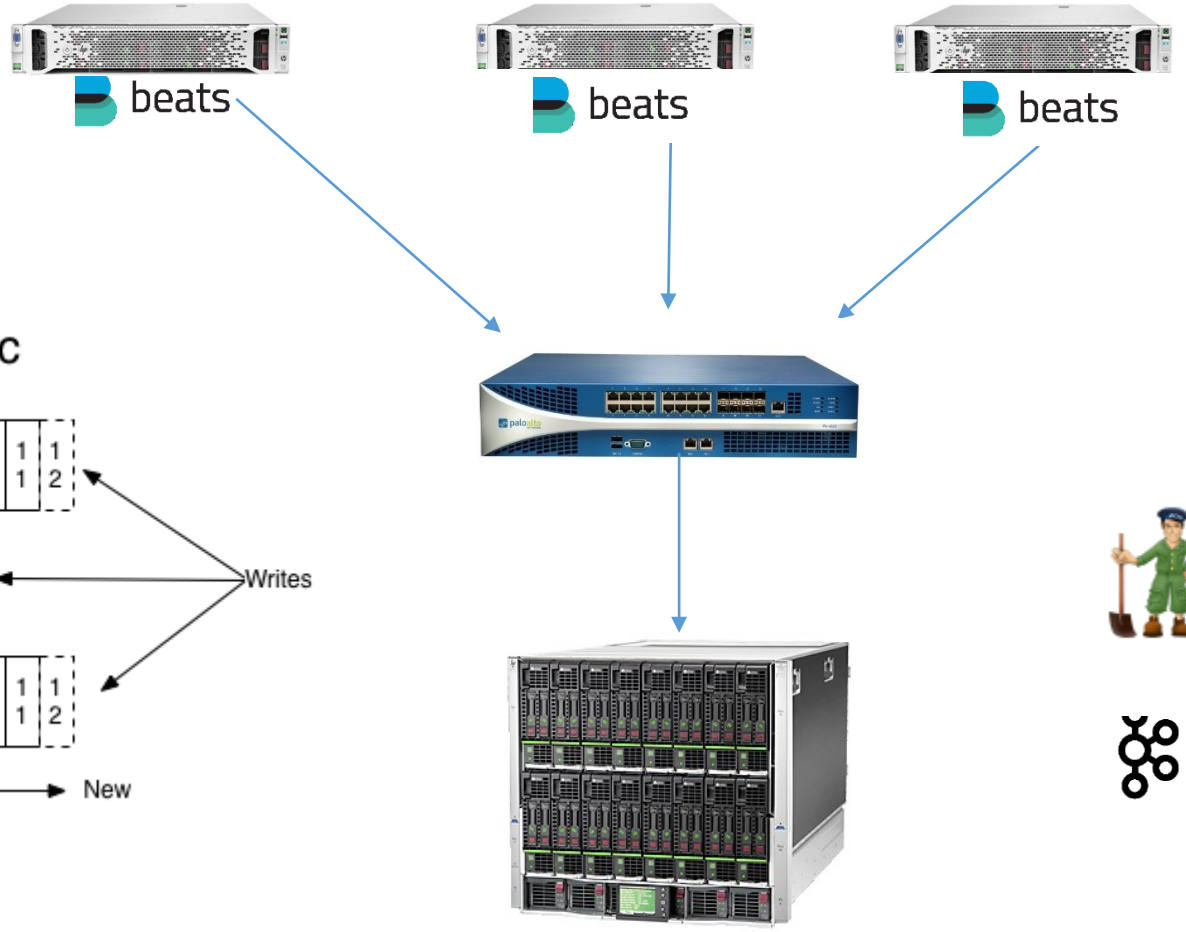
- Gran rendimiento y fácilmente escalable
- Pensado para caching en memoria
- No permite cifrado de las comunicaciones sin componentes adicionales (stunnel)

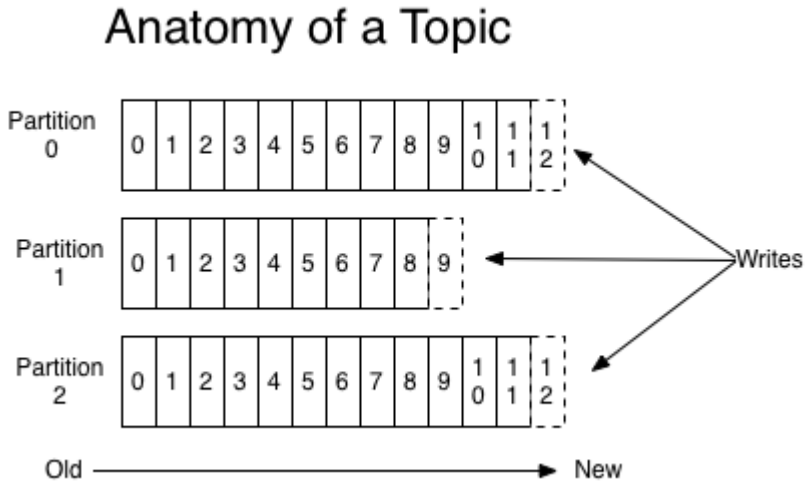
Kafka

- Gran rendimiento y fácilmente escalable
- Permite almacenar y reconstruir las secuencias de mensajes durante un tiempo de retención
- Orientado a ‘topics’
- Permite cifrado de las comunicaciones
- Amplio número de conectores disponibles

Capa de streaming/buffering

Universitat Oberta de Catalunya





Qué es Apache Kafka?

Universitat Oberta de Catalunya

- Según su sitio web: Plataforma de Streaming distribuido.
  - Similar a los sistemas tradicionales empresariales de emisor-suscriptor.

- Características:
  - Almacenar información temporalmente a la espera de ser persistida
  - Entregar mensajes a varios servicios a la vez
  - Realizar transformaciones sobre grandes volúmenes de datos en tiempo real.
  - Se garantiza la entrega de los mensajes en orden.
  - El stream es reproducible desde el principio, pero a la vez se proporcionan mecanismos para “recordar” hasta dónde han sido consumidos los datos.
  - Soporte para cifrado SSL y autenticación SASL
  - Muy baja latencia
  - Proporciona API’s para trabajar con streams en múltiples lenguajes de programación (Kafka Streams)

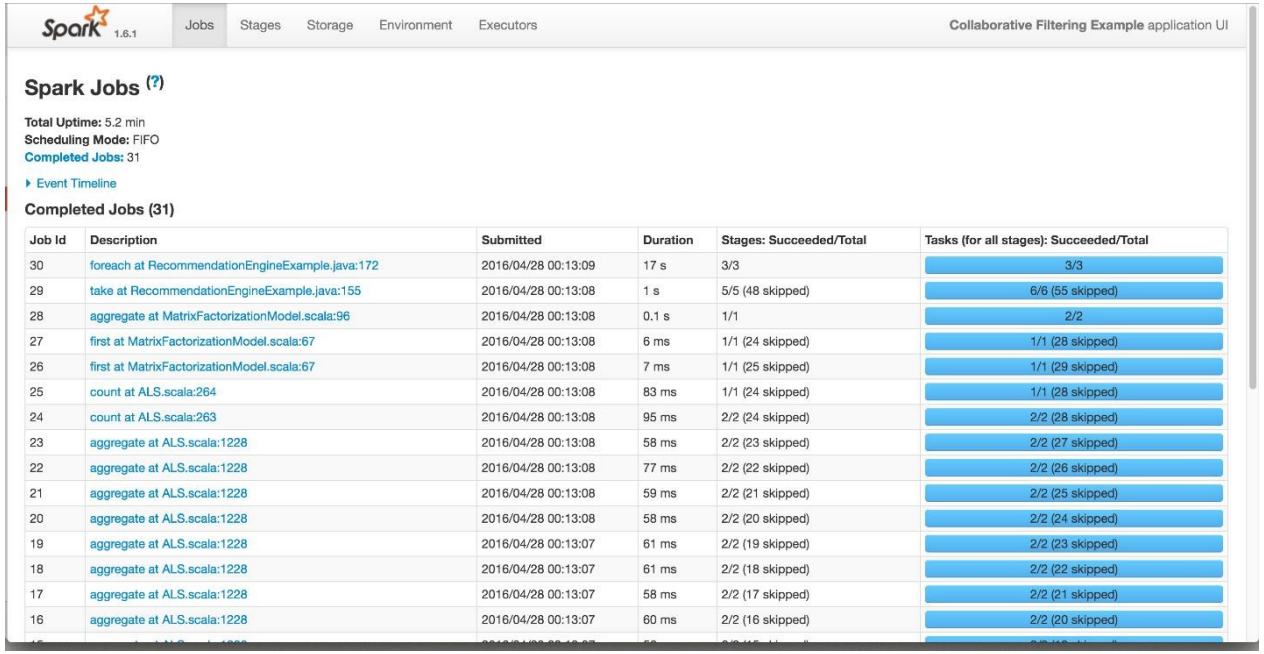
Capa de transformación y procesado

Universitat Oberta de Catalunya

- SMT, single message transforms (Confluent / Kafka Connect)
- Kafka streams
- Spark Streams & Spark ML

confluent

Spark



Spark 1.6.1 — Jobs | Stages | Storage | Environment | Executors — Collaborative Filtering Example application UI

Spark Jobs (?)

Total Uptime: 5.2 min
Scheduling Mode: FIFO
Completed Jobs: 31

▸ Event Timeline

Completed Jobs (31)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
30	foreach at RecommendationEngineExample.java:172	2016/04/28 00:13:09	17 s	3/3	3/3
29	take at RecommendationEngineExample.java:155	2016/04/28 00:13:08	1 s	5/5 (48 skipped)	6/6 (55 skipped)
28	aggregate at MatrixFactorizationModel.scala:96	2016/04/28 00:13:08	0.1 s	1/1	2/2
27	first at MatrixFactorizationModel.scala:67	2016/04/28 00:13:08	6 ms	1/1 (24 skipped)	1/1 (28 skipped)
26	first at MatrixFactorizationModel.scala:67	2016/04/28 00:13:08	7 ms	1/1 (25 skipped)	1/1 (29 skipped)
25	count at ALS.scala:264	2016/04/28 00:13:08	83 ms	1/1 (24 skipped)	1/1 (28 skipped)
24	count at ALS.scala:263	2016/04/28 00:13:08	95 ms	2/2 (24 skipped)	2/2 (28 skipped)
23	aggregate at ALS.scala:1228	2016/04/28 00:13:08	58 ms	2/2 (23 skipped)	2/2 (27 skipped)
22	aggregate at ALS.scala:1228	2016/04/28 00:13:08	77 ms	2/2 (22 skipped)	2/2 (26 skipped)
21	aggregate at ALS.scala:1228	2016/04/28 00:13:08	59 ms	2/2 (21 skipped)	2/2 (25 skipped)
20	aggregate at ALS.scala:1228	2016/04/28 00:13:08	58 ms	2/2 (20 skipped)	2/2 (24 skipped)
19	aggregate at ALS.scala:1228	2016/04/28 00:13:07	61 ms	2/2 (19 skipped)	2/2 (23 skipped)
18	aggregate at ALS.scala:1228	2016/04/28 00:13:07	61 ms	2/2 (18 skipped)	2/2 (22 skipped)
17	aggregate at ALS.scala:1228	2016/04/28 00:13:07	58 ms	2/2 (17 skipped)	2/2 (21 skipped)
16	aggregate at ALS.scala:1228	2016/04/28 00:13:07	60 ms	2/2 (16 skipped)	2/2 (20 skipped)

Capa de transformación y procesado (Kafka Streams)

Universitat Oberta de Catalunya

- Kafka Streams permite transformar un topic Kafka en una fuente de datos en tiempo real, sobre la que se pueden aplicar transformaciones de forma muy eficiente.
- Una aplicación Kafka streams se ejecuta de forma ‘Stand Alone’. No requiere un clúster de procesamiento.
- Ideal para operaciones sencillas como:
  - Enriquecimiento de datos (P.Ejemplo, añadir datos GeoIP)
  - Transformaciones de formato
  - Anonimización de los datos

Capa de transformación y procesado (Kafka streams)

Universitat Oberta de Catalunya

Ejemplo sencillo. Reemplazar delimitador del texto que se recibe en un topic y escribir la cadena transformada en otro con Kafka streams

```
1  KStreamBuilder builder = new KStreamBuilder();
2
3  KStream<String, UserActivityMessage> userActivity = builder
4      .stream("notarias-es-mad");
5
6  userActivity
7      .map((key, value) -> {
8          return new KeyValue<>(key, value.replace(',', value.getDelimiter()));
9      })
10     .to("notarias-es-mad-normalizado");
11
12 KafkaStreams streams = new KafkaStreams(builder, getSettings());
13 streams.start();
```

Capa de transformación y procesado (Spark)

Universitat Oberta de Catalunya

- Spark permite crear streams desde varias fuentes. No solo apache Kafka:
  - Twitter
  - Ficheros CSV
  - HDFS….
- Su función en el pipeline sería hacer un procesamiento más avanzado que con Kafka streams.
  - Predicciones basadas en información en tiempo real relacionada con la información histórica almacenada en el almacén HDFS
  - Entrenamiento de modelos de ML para la detección de anomalías
  - Posibilidad de realizar informes con Spark SQL, que nos permite realizar consultas SQL sobre streams, pudiendo hacer consultas sobre ventanas de tiempo de las métricas y datos adquiridos previamente.

Capa de transformación y procesado (Spark)

Universitat Oberta de Catalunya

- Spark es un framework que trabaja con micro-batches en lugar de tiempo real. La duración con la que se procesan estos micro-batches es variable
- Spark está diseñado para funcionar en clúster. Para ejecutar trabajos soporta:
  - Su propio módulo de ejecución (Standalone)
  - Hadoop YARN
  - Mesos
  - Kubernetes

Capa de almacenamiento (HDFS)

Universitat Oberta de Catalunya

- Para el almacenamiento de los datos a largo plazo se utilizará HDFS
- HDFS es el sistema de archivos *distribuido* de Hadoop
- Diferentes tipos de nodo
  - Máster
  - DataNodes
- Interactuamos con él mediante el comando hdfs dfs <subcomando>
  - -ls
  - -mkdir …
- Soporta usuarios y permisos como en otros sistemas de archivos corrientes.

Consultas ad-hoc

Universitat Oberta de Catalunya

Elasticsearch + Kibana

- Se indexa la información a medio plazo, lo cual permite hacer búsquedas y realizar gráficas.
- Sencillo de escalar, ya que soportan clústering.
- Necesidad de seguridad externa en versión Open Source.

Apache Drill

- Posibilidad de hacer consultas SQL de forma muy potente sobre gran cantidad de ficheros almacenados en HDFS

Prueba de concepto

Universitat Oberta de Catalunya

Droplets

Search by Droplet name

Droplets | Volumes

Name	IP Address	Created ▲	Tags	
docker 1 GB / 25 GB Disk / AMS3 - Ubuntu Docker 17.12.0~ce on 1...	188.166.88.208	7 days ago		More
elk-01 3 GB / 25 GB Disk / AMS3 - Ubuntu 16.04.4 x64	167.99.35.198	13 days ago		More
hadoop-0 1 GB / 25 GB Disk / AMS3 - Ubuntu 16.04.4 x64	167.99.215.158	16 days ago		More
hadoop-1 2 GB / 25 GB Disk / AMS3 - Ubuntu 16.04.4 x64	206.189.3.98	16 days ago		More
hadoop-2 2 GB / 25 GB Disk / AMS3 - Ubuntu 16.04.4 x64	174.138.11.120	16 days ago		More
kafka-02 4 GB / 60 GB Disk / AMS3 - Ubuntu 16.04.4 x64	206.189.11.15	21 days ago		More
kafka-01 3 GB / 60 GB Disk / AMS3 - Ubuntu 16.04.4 x64	206.189.3.210	21 days ago		More

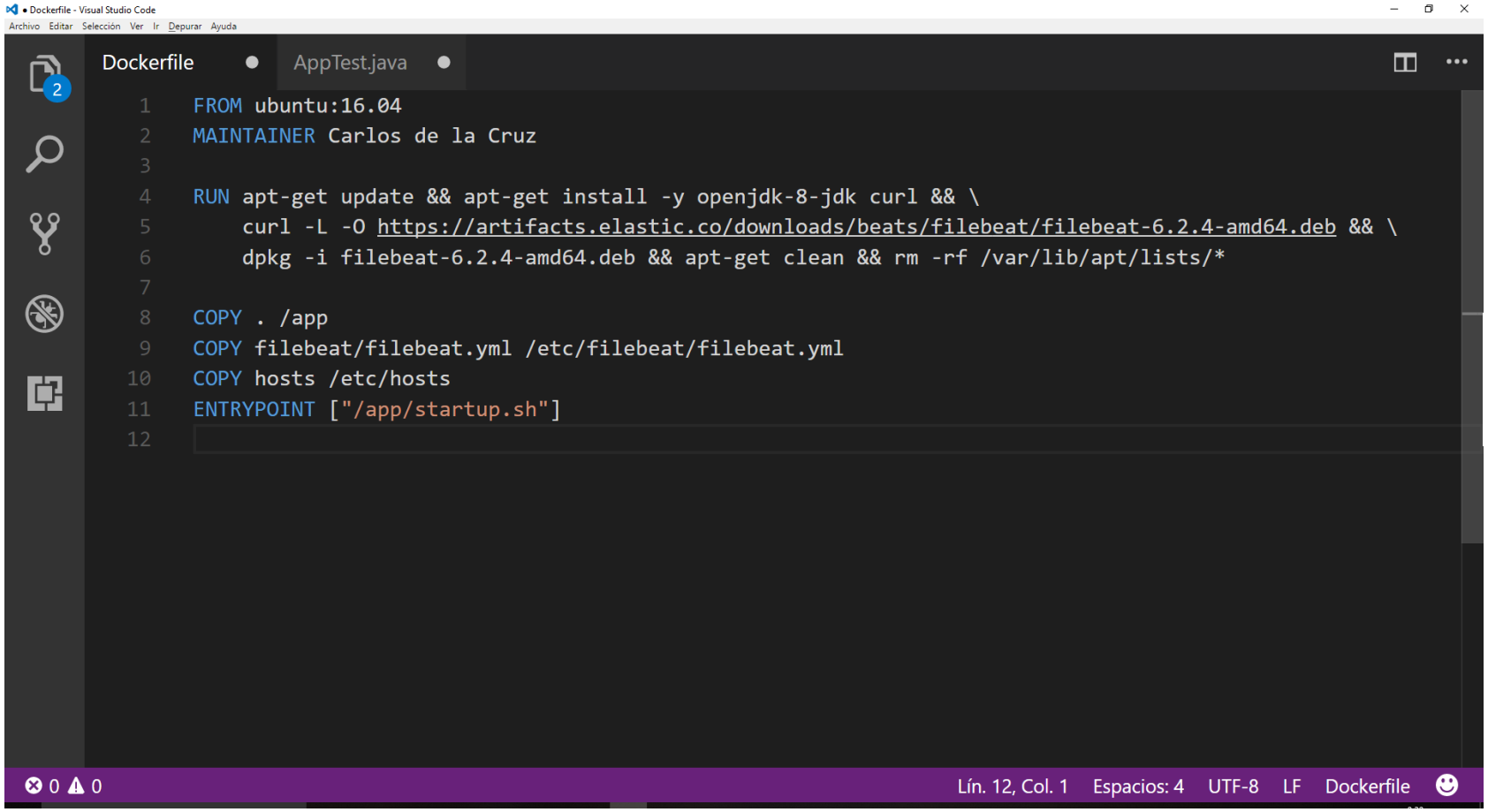
docker

HashiCorp Terraform

ANSIBLE

Prueba de concepto
Generando mensajes

Universitat Oberta de Catalunya



docker

beats

Prueba de concepto: Provisionamiento de la infraestructura

Universitat Oberta de Catalunya

- Se han generado playbooks para Ansible para:
  - Configuración del clúster Hadoop
  - Configuración del clúster Confluent / Kafka
  - Instalación de Filebeat en los clientes
  - Instalación de Elastic Stack

- Se ha generado un fichero modelo para Terraform

con el proveedor para DigitalOcean

```
variable "digitalocean_ssh_fingerprint"   { }
provider "digitalocean" {
    token = "<digital ocean API token>"
}



resource "digitalocean_droplet" "terraform" {
    image = "ubuntu-16-04-x64"
    count = "3"
    name = "hadoop-${count.index}"
    region = "ams3"
    size = "s-2vcpu-4gb"
    ssh_keys = [ "${var.digitalocean_ssh_fingerprint}"]
    provisioner "remote-exec" {
        inline = [ "apt-get update; apt-get install -y python"]
    }
    provisioner "local-exec" {
        command = "sleep 120; ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -u root
        --private-key ./deployer.pem -i inventory hadoop,' master.yml"
    }

}

```

ANSIBLE

HashiCorp Terraform

Paso a producción

Universitat Oberta de Catalunya

- Uso de playbooks para configurar cada servidor con los agentes apropiados.
- Cambio del código terraform para desplegar las máquinas virtuales en VMWare Vsphere.
- Dimensionamiento apropiado de número de particiones Kafka y del número de nodos
- Valoración de la necesidad de alta disponibilidad. Necesidad de almacenamiento compartido en este caso para el clúster Hadoop.
- Introducción de un sistema para gestionar las keys SSL (Hashicorp Vault, o Chef Vault).
- Transformación de los datos enviados a Elasticsearch a un formato más rico (campos separados, más metainformación…)
- Configuración del Heap de Java acorde a la memoria de los nodos en las distintas piezas de software.