

Trabajo Práctico 4

Infraestructuras Computacionales para Procesamiento de
Datos Masivos

Irene J. Ventura Farias

Resumen

Este trabajo práctico busca que nos familiaricemos con entornos en la nube y los posibles servicios que ofrecen. Además de ello, conocer como podemos trabar con los servicios fundamentales en la ingeniería de datos (EC2, EMR y S3).

El objetivo principal era lanzar un bechmark sobre un clúster, para evaluar diferentes tareas en EMR. Luego los datos obtenidos por cada tarea, los logs del clúster y ejecutable del benchamark (aplicación) se almacenaban en S3. Y también acceder a la instancia master generada por el clúster a través de SSH.

En el apartado siguiente explicaré paso a paso el desarrollo de la práctica.

Desarrollo de la Práctica

1. Selección del repositorio: <https://github.com/ehiggs/spark-terasort>
2. Crear un Key Pair para acceder por SSH
 - a. EC2 > Network & Security> Key Pairs> Create Key Pair

Create key pair [Info](#)

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type [Info](#)

☒ RSA

☐ ED25519

Private key file format

☒ .pem
For use with OpenSSH

☐ .ppk
For use with PuTTY

Tags (Optional)

No tags associated with the resource.

Add tag

You can add 50 more tags.

Cancel

Create key pair

3. [Crear Bucket y Folders](#)

Hemos creado un bucket para la práctica, la cual tendrá los siguientes directorios, los cuales cubrirán diferentes cosas:

- **application:** contiene el benchmark y el ejecutable
- **generate_data, sort_data y validate_data:** cada uno almacena los valores generados por los steps ejecutados (Paso 6)
- **logs:** logs generados por el clúster

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Amazon S3 > tp4-storage

tp4-storage [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

Objects (5)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

< 1 > ⚙

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	application/	Folder	-	-	-
<input type="checkbox"/>	generate_data/	Folder	-	-	-
<input type="checkbox"/>	logs/	Folder	-	-	-
<input type="checkbox"/>	sort_data/	Folder	-	-	-
<input type="checkbox"/>	validate_data/	Folder	-	-	-

4. [Creamos el Clúster en EMR](#)

5. Accedemos al Clúster a través de SSH:

```
ssh -i tp4-access.pem hadoop@ec2-<IP>.compute-1.amazonaws.com
```

- Instalar desde el server ssh:
 - Para clonar el repositorio del benchmark: [Git](#)
 - Para obtener el ejecutable del benchmark: [Maven](#)
- Clonamos el repositorio y accedemos al directorio principal del repositorio:
- Ejecutamos `mvn install` sobre directorio principal del benchmark y obtenemos los ejecutables, en nuestro caso vamos a trabajar con el .jar que incluye las dependencias

```
[hadoop@tp-172-31-6-159 target]$ ls
analysis  classes  jars      maven-status  spark-terasort-1.2-SNAPSHOT.jar  spark-terasort-1.2-SNAPSHOT-javadoc.jar
archive-tmp  generated-sources  maven-archiver  site          spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar
```

- Copiamos la Aplicación a S3 mediante [AWS CLI](#)

6. Creación de los Steps para probar el Benchmark:

Para los 3 pasos se le asigno su nombre respectivo, el jar-with-dependencies (`s3://tp4-storage/application/spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar`) que generamos del benchmark y el spark-submit correspondiente.

- Generate Data:** Se le indico que generara 1000MB de datos aleatorios y la ruta de s3 en donde volcara los datos `s3://tp4-storage/generate_data/`

The screenshot shows the 'Add step' dialog in the AWS EMR console. The 'Step type' is set to 'Custom JAR'. The 'Name' is 'Generate Data'. The 'JAR location' is `s3://tp4-storage/application/spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar`. The 'Arguments' field contains the command: `spark-submit --class com.github.ehiggs.spark.terasort.TeraGen s3://tp4-storage/application/spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar 1g s3://tp4-storage/generate_data/`. The 'Action on failure' is set to 'Continue'. There are 'Cancel' and 'Add' buttons at the bottom right.

- Sort Data:** Sobre los datos generados en el paso anterior obtendremos los datos ordenados. `s3://tp4-storage/sort_data/`

The screenshot shows the 'Add step' dialog in the AWS EMR console. The 'Step type' is set to 'Custom JAR'. The 'Name' is 'Sort Data'. The 'JAR location' is `s3://tp4-storage/application/spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar`. The 'Arguments' field contains the command: `spark-submit --class com.github.ehiggs.spark.terasort.TeraSort s3://tp4-storage/application/spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar s3://tp4-storage/generate_data/ s3://tp4-storage/sort_data/`. The 'Action on failure' is set to 'Continue'. There are 'Cancel' and 'Add' buttons at the bottom right.

c. **Validate Data:** valida que los datos del paso anterior estén ordenados

Add step

Step type

Custom JAR

Name*

Custom JAR

JAR location*

s3://tp4-storage/application/spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar

JAR location may be a path into S3 or a fully qualified java class in the classpath.

Arguments

```
spark-submit --class com.github.ehiggs.spark.terasort.TeraValidate
s3://tp4-storage/application/spark-terasort-1.2-SNAPSHOT-jar-with-dependencies.jar
s3://tp4-storage/sort_data/ s3://tp4-storage/validate_data/
```

OR, if the test file you want to use is in the classpath.

Action on failure

Continue

What happens if the step fails

Cancel

Add

Opinión sobre la TP4

En líneas generales me parece que la práctica abarca lo fundamental para tratar datos, aplicaciones de Spark en la Nube. Sin embargo, considero que el contenido teórico no cubre lo suficiente para poder abordar el problema de esta práctica, ya muchas de las soluciones se obtuvieron a través de internet y sin saber con certeza si que lo que se estaba haciendo iba por buen camino. Además, la mayoría de los benchmarks propuestos tienen problemas de versiones con Spark o incluso con sus lenguajes de programación, en mi caso particular pese a que el benchmark sugería usar Spark 2.4.4 tuve que usar una versión más actualizada para poder lanzarla, así como también tener conocimientos previos de Scala para abordar ciertas modificaciones al código.