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).

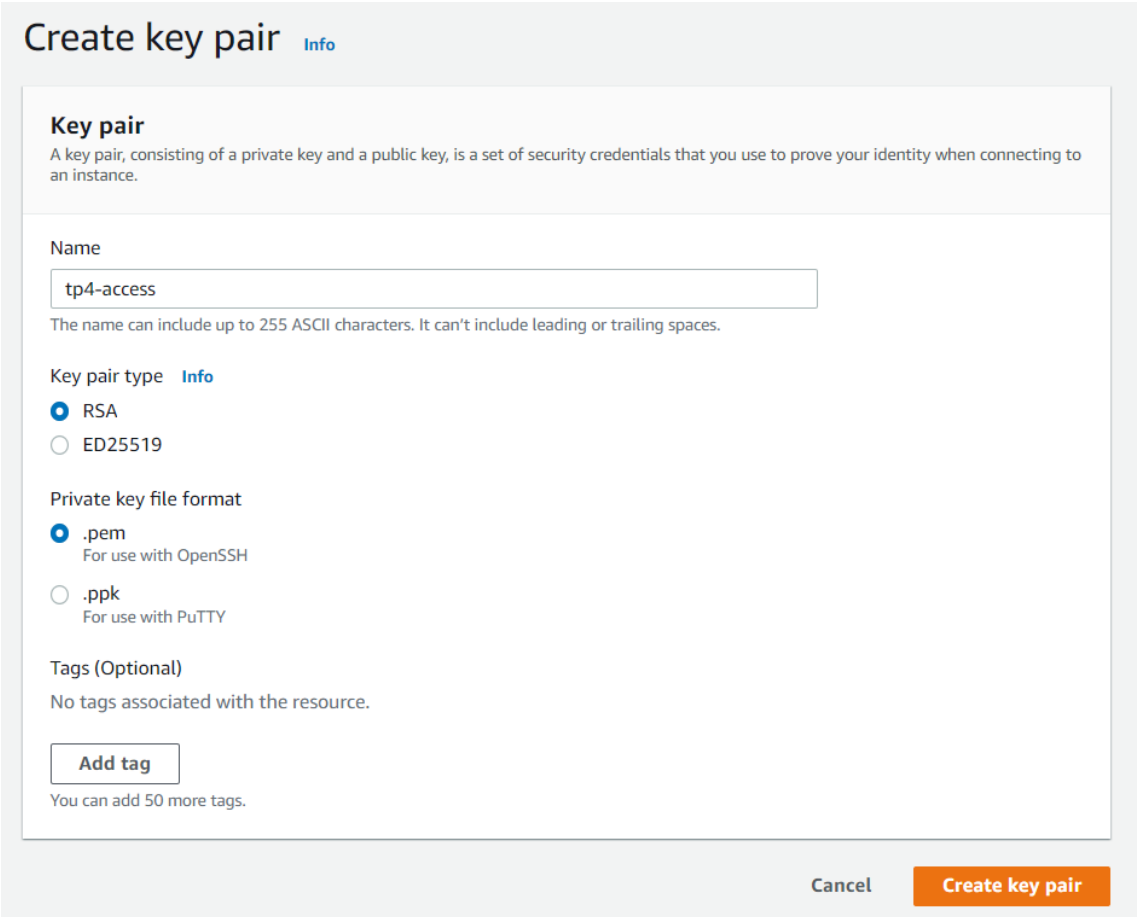
El objetivo principal era lanzar un bechmark sobre un clúster, para evaluar diferentes tareas en EMR. Luego los datos obtenidos por cada tarea, los logs del clúster y ejecutable del benchamark (aplicación) se almacenaban en S3. Y también acceder a la instancia master generada por el clúster a través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

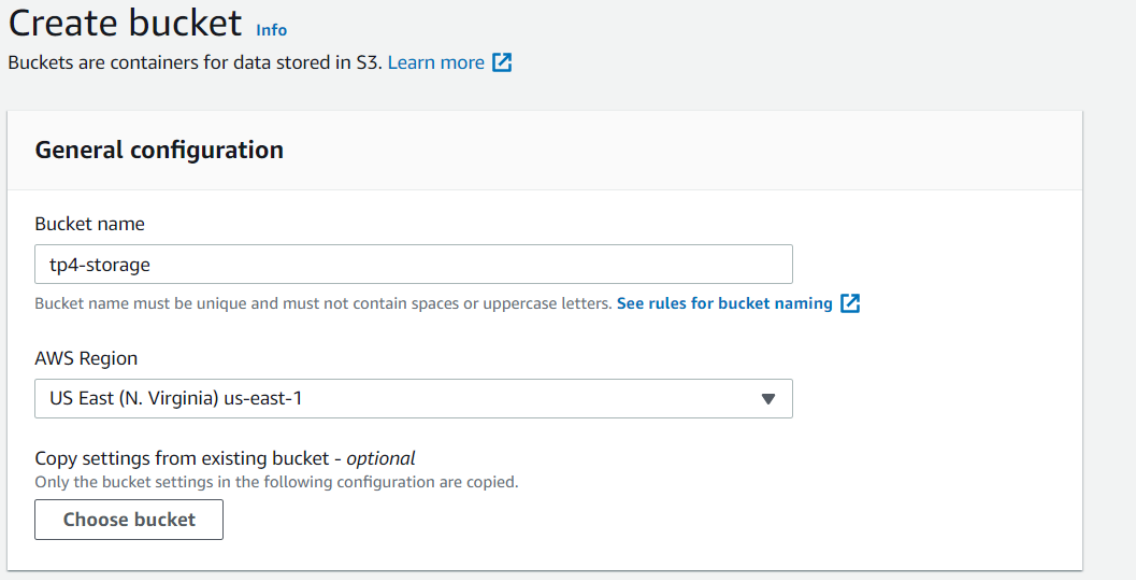
2. Crear un Key Pair para acceder por SSH
    a. EC2 > Network & Security> Key Pairs> Create Key Pair

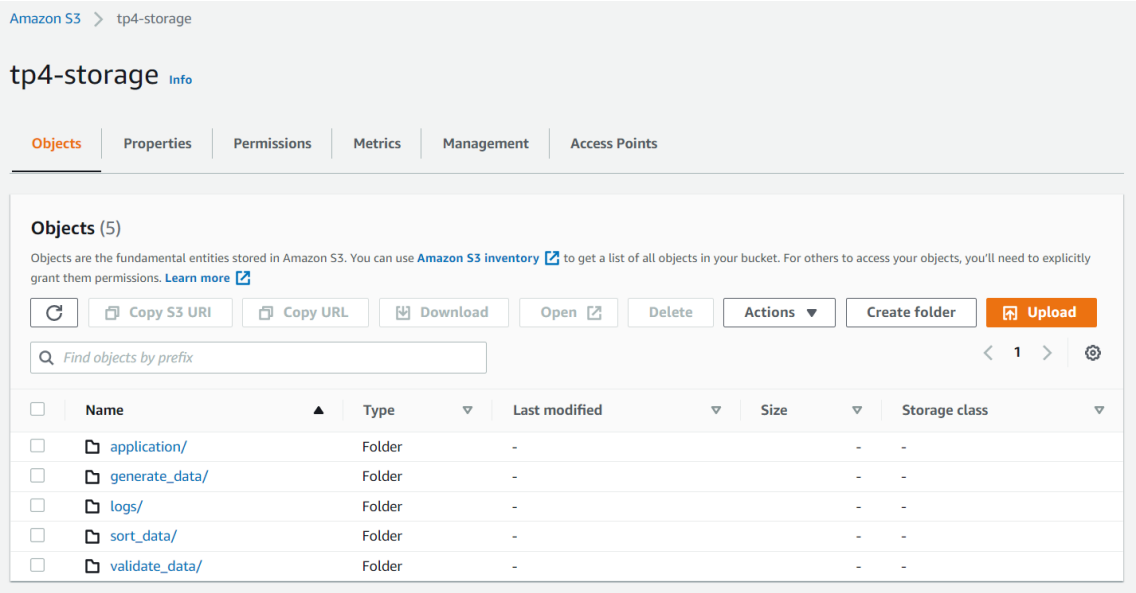


3. Crear Bucket y Folders

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)
- **log**s: logs generados por el clúster





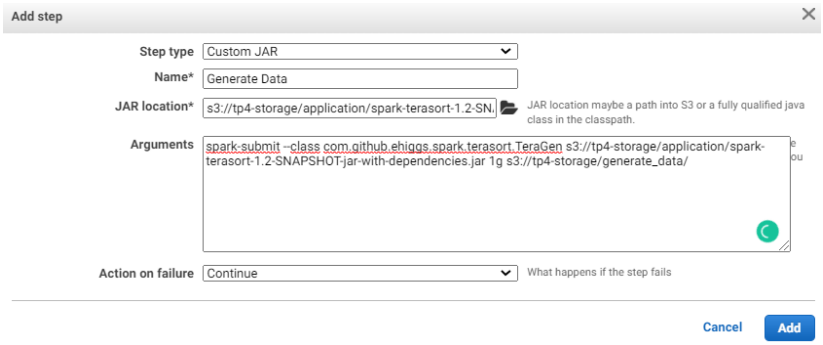
4. Creamos el Clúster en EMR
5. Accedemos al Clúster a través de SSH:
   ssh -i tp4-access.pem hadoop@ec2-<IP>.compute-1.amazonaws.com
   a. Instalar desde el server ssh:
      i. Para clonar el repositorio del benchmark: Git
      ii. Para obtener el ejecutable del benchmark: Maven
   b. Clonamos el repositorio y accedemos al directorio principal del repositorio:
   c. Ejecutamos mvn install sobre directorio principal del bechmark y obtenemos los ejecutables, en nuestro caso vamos a trabajar con el .jar que incluye las dependencias

```
[hadoop@ip-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
[hadoop@ip-172-31-6-159 target]$
```

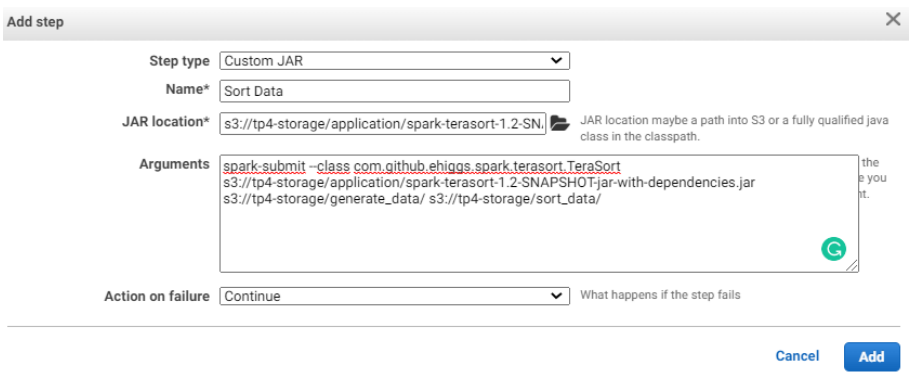
   d. Copiamos la Aplicación a S3 mediante AWS CLI

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.

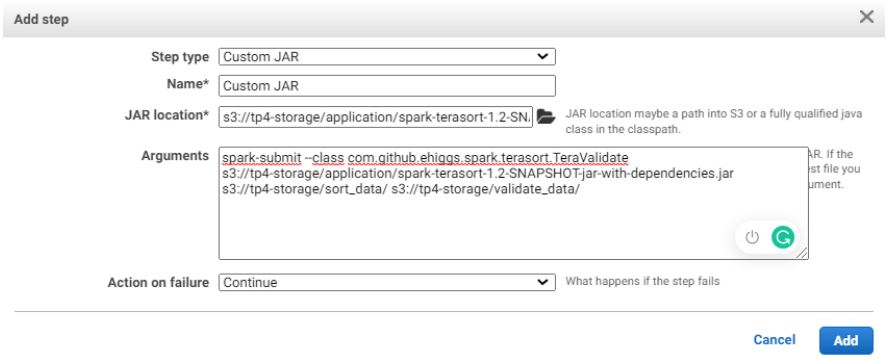
   a. **Generate Data:** Se le indico que generara 1000MB de datos aletorios y la ruta de s3 en donde volcara los datos s3://tp4-storage/generate_data/



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



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



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 sí 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.