Amazon Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR) es un servicio web para la configuración y despliegue de un cluster basado en instancias de máquinas en el servicio Amazon Elastic Compute Cloud (Amazon EC2) y que es gestionado mediante Hadoop. También se puede ejecutar en Amazon EMR otros marcos de trabajo distribuidos como Spark, e interactuar con los datos en otros almacenes de datos como Amazon S3.

Creación de un cluster con EMR

Un cluster EMR suele tener un ciclo de vida totalmente automatizado y que se establece en el momento de su creación. El proceso general sería:

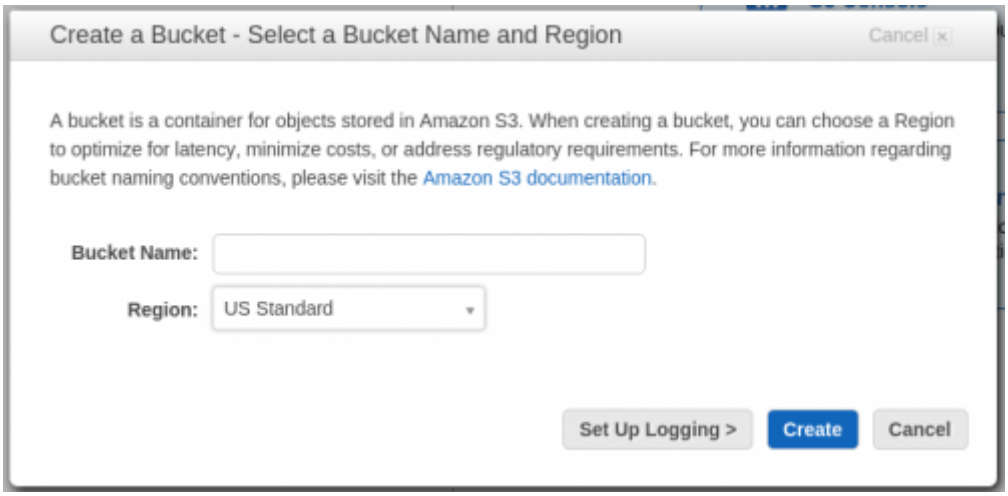
- Lanzamiento de las instancias EC2 de las que se compone el cluster
- Ejecución de los scripts de instalación, tanto automáticos de amazon (como las imágenes preconfiguradas AMI) como los añadidos por el usuario en las acciones de inicialización (Bootstrap actions).
- Trabajos a realizar (Steps) normalmente consistentes en carga de datos de entrada, procesamiento de los mismos, y almacenado de los resultados.
- Apagado automático del cluster una vez se han terminado todos los steps.

En las siguientes subsecciones se explica todo lo básico para poder lanzar un cluster EMR y analizar los resultados de las ejecuciones.

Almacenamiento con S3

Amazon EMR puede hacer uso de Amazon S3 como almacenamiento de los datos de entrada, los ficheros de log y los datos de salida. Para más información sobre este tipo de sistema de almacenamiento visita la wiki de amazon.

Para crear un nuevo contenedor de datos S3 (bucket), solamente es necesario entrar en el servicio S3 y pulsar “Create Bucket” rellenando el nombre del nuevo contenedor y la región donde estará el mismo (es importante que esta sea la misma región utilizada para desplegar el cluster EMR).



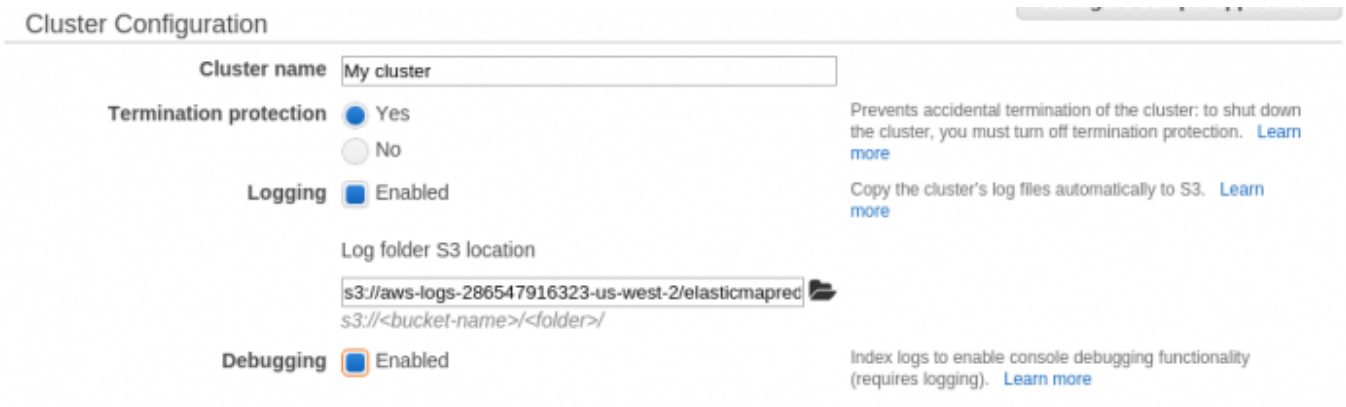
Una vez creado el contenedor, suele ser una buena práctica organizarlo de la siguiente manera:

- Crear una carpeta `log` donde guardar los logs de los despliegues de máquinas EC2, así como de las ejecuciones de los diferentes trabajos.
- Crear una carpeta `input` para tener almacenados todos los datos de entrada.
- Crear una carpeta `output` que servirá para guardar los resultados de las ejecuciones.

Además, será necesario tener en este contenedor todo lo necesario para el trabajo que se vaya a ejecutar en el cluster, así como los diferentes scripts de configuración (tal como se comenta en la siguiente sección).

Configuración del cluster

Una vez se dispone de un contenedor S3, ya es posible lanzar un cluster EMR plenamente útil. Después de pulsar en `Create cluster` en la consola de EMR, lo primero que hay que hacer es la configuración general del cluster: nombre del cluster y donde almacenar los logs (directorio `log` que hemos creado previamente en el bucket S3).

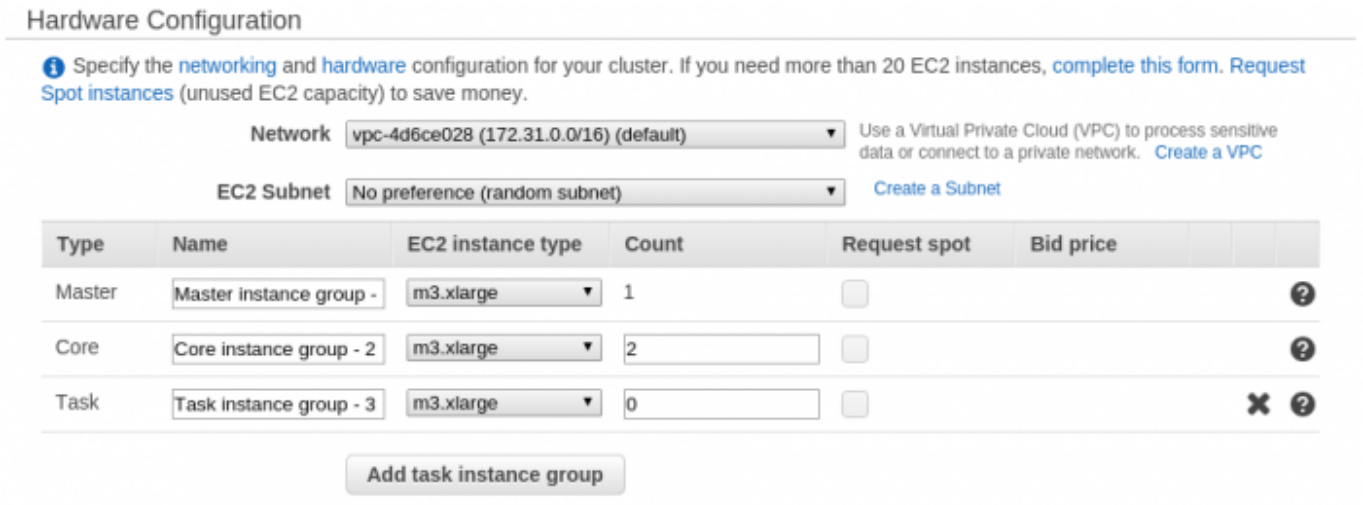


El siguiente paso es la configuración del software que estará disponible en el cluster. En primer lugar se elige la distribución de Hadoop preconfigurada por amazon (versión mayor que emr-4.0.0). Además, es posible añadir software adicional que proporciona Amazon. Es importante que entre este software se encuentre Spark.

Es importante configurar Spark para que use todos los recursos disponibles en los nodos. Para ello, una configuración básica para introducir en `Edit software settings`:

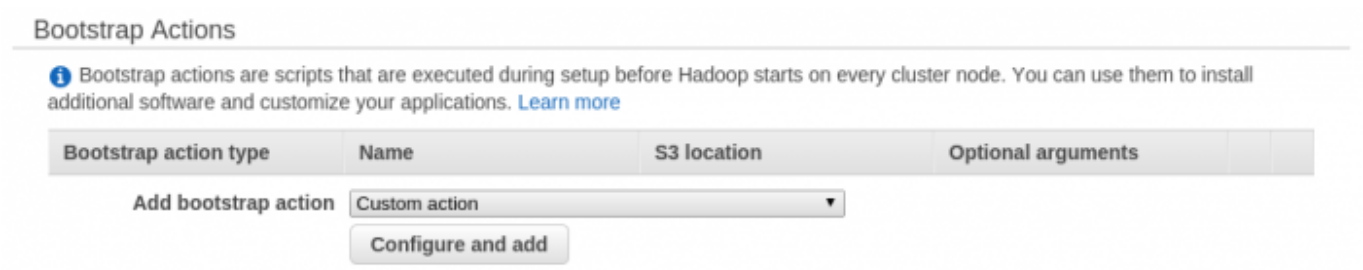
```
[{"classification":"spark","properties":{"maximizeResourceAllocation":"true"}},{"classification":"spark-defaults","properties":{"spark.network.timeout":"500s"}}]
```

Una vez configurado el software, se continúa con la configuración del hardware. La configuración más típica se compone por un nodo `Master` donde se lanzarán los trabajos y 1 o más instancias `Core` que harán de workers dentro del cluster hadoop (para realizar, por ejemplo, las tareas de mapper). Dependiendo del tipo de necesidad, amazon pone a disposición varios tipos de instancias EC2.

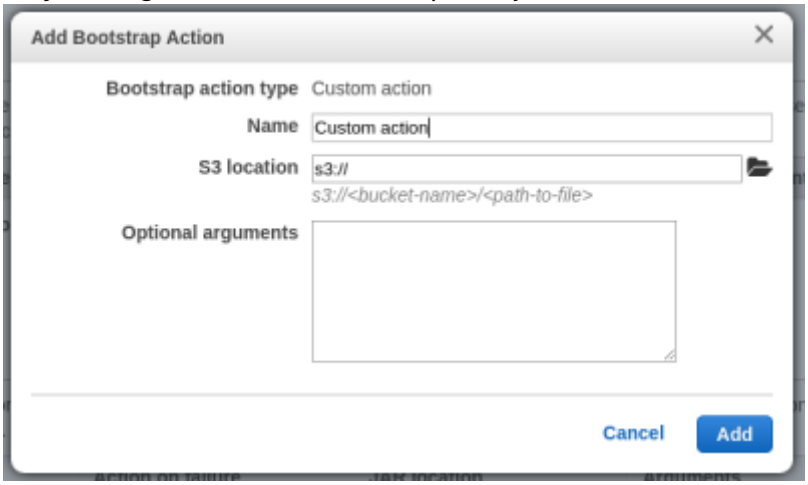


También es posible añadir un par de claves de acceso previamente generadas para poder acceder al master mediante ssh.

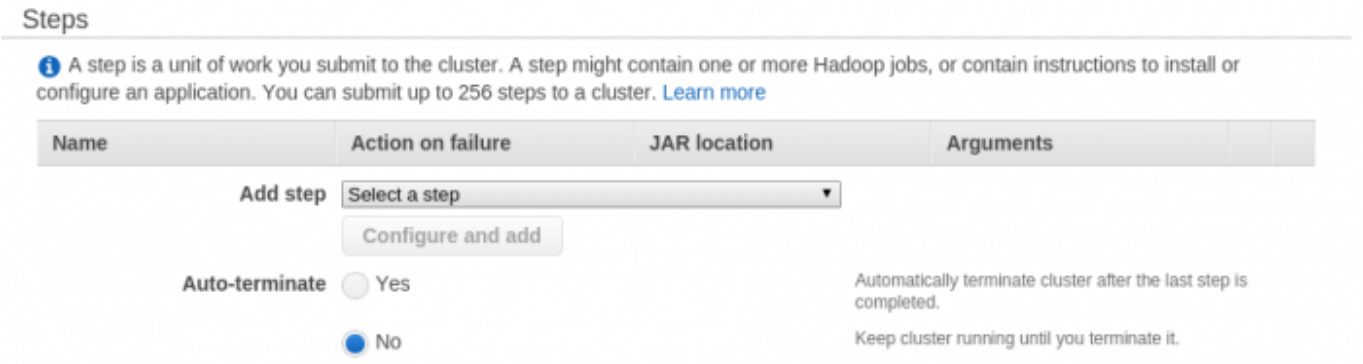
Además del software preconfigurado por amazon, se pueden realizar más acciones de instalación de software o configuración mediante acciones de lanzamiento (Bootstrap actions).



Para añadir una nueva acción de lanzamiento, es necesario indicar donde está almacenado el script dentro de un bucket S3 y los argumentos necesarios para ejecutar la acción.



Por último, aunque es posible añadir trabajos una vez desplegado el cluster, el procedimiento habitual y más seguro es añadir los trabajos (`Steps`) a realizar en el cluster antes de su lanzamiento.



Por ejemplo, para añadir la ejecución de un archivo `jar` de Java mediante Hadoop, es necesario indicar donde está almacenado el programa dentro de un bucket S3 y los argumentos del mismo. La acción a realizar al terminar el trabajo suele ser el apagado del cluster. Sin embargo, si existiesen varios steps, esta acción sería utilizada solo por el último step.

Logs

Durante el despliegue y ejecución del cluster, se irán generando una serie de logs que serán guardados en la carpeta indicada dentro del contenedor S3 (más información).

De entre los logs generados, cabe destacar:

- `/<clusterID>/node/` contiene los logs de ejecución de los bootstrap actions para cada uno de los nodos, así como el estado de las instancias.
- `/<cluasterID>/steps/` contiene los logs generados por ejecutar cada uno de los trabajos añadidos como steps. Dentro de estas carpetas se pueden observar los siguientes archivos de log:
  - controller — Información sobre el procesamiento del trabajo.
  - syslog — Describe la ejecución del trabajo mediante hadoop.
  - stderr — La salida estandar de error del trabajo (en Spark suele ser aquí donde están los logs generados por la ejecución del trabajo)
  - stdout — La salida estandar del trabajo.

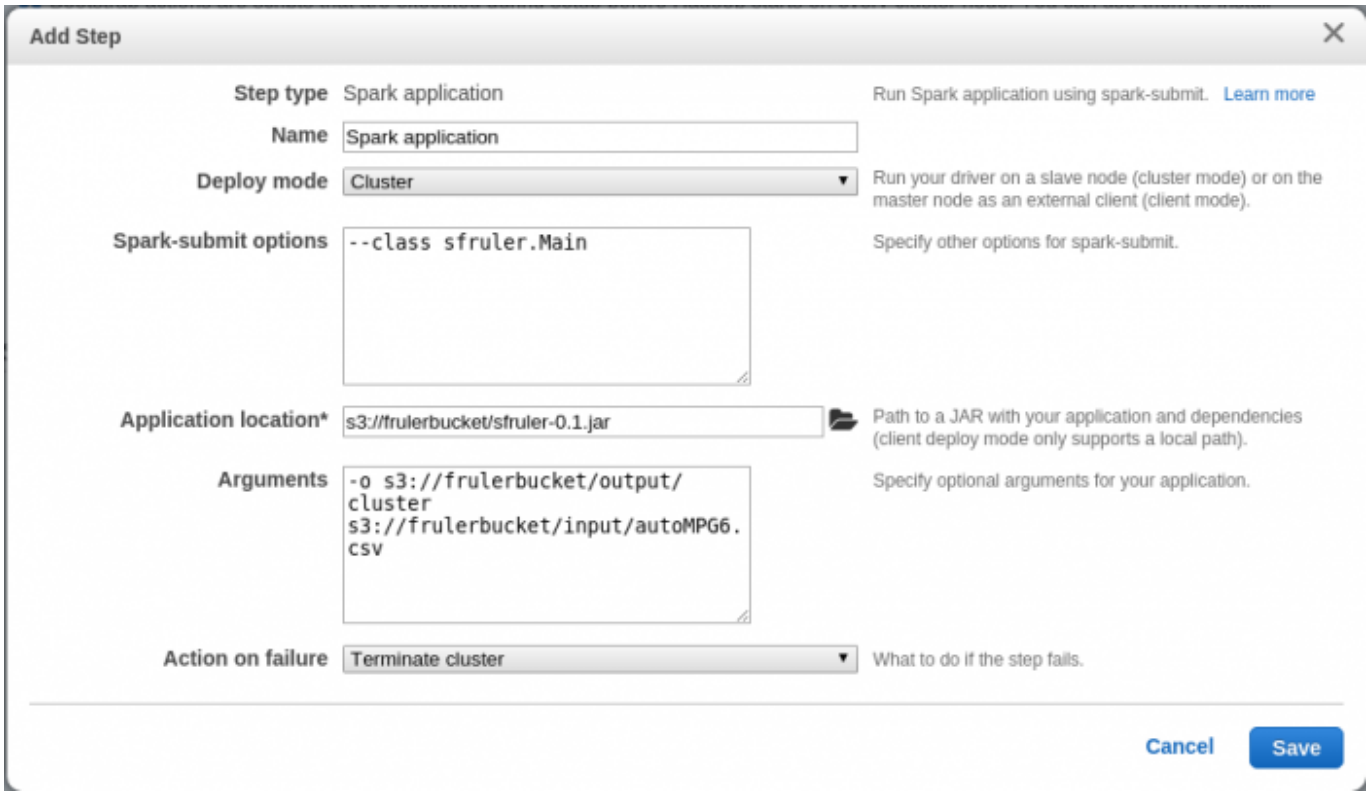
Spark sobre EMR

Desde la versión 4 de la imagen de Hadoop de Amazon (`emr-4.0.0`), Spark está totalmente integrado y solamente es necesario añadir como `Step` un `Spark application`.

Ejecutar un trabajo

Para ejecutar un trabajo sobre Spark, las opciones son las siguientes (sustituir los argumentos entre <> por sus valores reales):

- **Step type**: Spark application
- **Deploy mode**: Cluster
- **Spark-submit options**: `–class <MainClass>`
- **Applicaiton location**: Ruta al archivo en una unidad S3.
- **Arguments**: Argumentos necesarios del programa JAR anterior.



Lectura y escritura en S3

Spark es capaz de leer y escribir mediante el protocolo s3 sin necesidad de realizar cambios.

Java 8 en EMR

La última versión de EMR (emr-4.0.0) contiene como versión java 7. Para aquellos que usan java 8 (algo común en este tipo de entorno, por el uso de funciones lambda), pueden instalarlo en el cluster mediante un script ejecutado como bootstrap action. Para ello, solo es necesario almacenar en el contenedor S3 el siguiente script [1)], para luego añadirlo como acción bootstrap a la hora de lanzar un nuevo cluster:

```
# Check java version
JAVA_VER=$(java -version 2>&1 | sed 's/java version
"\(.*\)\.\(.*\)\..*"/\1\2/; 1q')

if [ "$JAVA_VER" -lt 18 ]
then
    # Figure out how many versions of Java and javac we currently have
    NR_OF_JRE_OPTIONS=$(echo 0 | alternatives --config java 2>/dev/null |
grep 'There ' | awk '{print $3}' | tail -1)
    NR_OF_SDK_OPTIONS=$(echo 0 | alternatives --config javac 2>/dev/null |
grep 'There ' | awk '{print $3}' | tail -1)

    # Silent install javac (includes jre)
    sudo yum -y install java-1.8.0-devel
```

```
    echo "Found $NR_OF_JRE_OPTIONS existing versions of java. Adding new
version."
    echo "Found $NR_OF_SDK_OPTIONS existing versions of javac. Adding new
version."

    # Make java 8 the default
    echo $(($NR_OF_JRE_OPTIONS + 1)) | sudo alternatives --config java
    echo $(($NR_OF_SDK_OPTIONS + 1)) | sudo alternatives --config javac

    # Fix wrong links
    sudo rm /etc/alternatives/java_sdk_openjdk;sudo ln -s
/usr/lib/jvm/java-1.8.0-openjdk.x86_64 /etc/alternatives/java_sdk_openjdk
    sudo rm /etc/alternatives/java_sdk_openjdk_exports;sudo ln -s
/usr/lib/jvm-exports/java-1.8.0-openjdk.x86_64
/etc/alternatives/java_sdk_openjdk_exports

fi

# Check java version again
JAVA_VER=$(java -version 2>&1 | sed 's/java version
"\(.*\)\.\(.*\)\..*"/\1\2/; 1q')

echo ""
echo "Java version is $JAVA_VER!"
echo "JAVA_HOME: $JAVA_HOME"
echo "JRE_HOME: $JRE_HOME"
echo "PATH: $PATH"
```

1)

Bootstrap script for installing Java 8 on an Amazon Elastic MapReduce instance (AMI 3.0.1)