AWS CLI (Command Line Interface)

Implantación de Aplicaciones Web

José Juan Sánchez Hernández

Curso 2023/2024

Índice

Índice de figuras

Índice de cuadros

1 AWS CLI (Command Line Interface)

En esta práctica vamos a aprender a utilizar algunos comandos básicos de la utilidad AWS CLI (Command Line Interace).

1.1 ¿Qué es AWS CLI?

AWS CLI (Command Line Interface) es una herramienta que se ejecuta desde la línea de comandos que permite gestionar todos los servicios de Amazon Web Services.

AWS CLI proporciona acceso directo a la API pública de los servicios de AWS, de forma que todas las funcionalidades que se pueden realizar desde la consola de admnistración web, también se pueden realizar con esta herramienta desde la línea de comandos.

Esta herramienta nos permite crear scripts de shell para automatizar la creación y la administración de los servicios de AWS.

1.2 Instalación de AWS CLI

La herramienta AWS CLI está disponible para:

- Linux
- macOS
- Windows
- Docker

Se recomienda consultar la documentación oficial para obtener los detalles de la instalación para cada una de las opciones disponibles.

1.2.1 Instalación en Linux

Para realizar la instalación de AWS CLI en un sistema operativo Linux sobre una arquitectura `x86` de 64 bits, tenemos que ejecutar los siguientes comandos.

Paso 1. Descargamos un archivo `.zip` con la aplicación AWS CLI.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

Paso 2. Descomprimimos el archivo que acabamos de descargar.

```
unzip awscliv2.zip
```

Paso 3. Ejectuamos el script de instalación.

```
sudo ./aws/install
```

Paso 4. Comprobamos que la instalación se ha realizado de forma correcta.

```
aws --version
```

1.2.2 Instalación en macOS

- Instalación de AWS CLI desde la interfaz de usuario.
- Instalación de AWS CLI desde la línea de comandos.

1.2.3 Instalación en Windows

- Instalación de AWS CLI desde la interfaz de usuario.

1.2.4 Uso de AWS CLI con un contenedor Docker

En Docker Hub está disponible la imagen amazon/aws-cli, que es una imagen verificada por Amazon. Esta imagen nos permite utilizar la herramienta AWS CLI sin necesidad de tener que instalarla en nuestro equipo.

En este caso, cada vez que se ejecute un comando se creará un contenedor Docker, se ejecutará el comando dentro del contenedor y luego se eliminará el contenedor.

El comando de Docker que tenemos que utilizar es el que se muestra a continuación.

```
docker run \
    -it \
    --rm \
    -v ~/.aws:/root/.aws \
    amazon/aws-cli \
    <COMMAND>
```

Vamos a explicar cada uno de los parámetros que aparecen en este comando:

- `-it` se utiliza para crear un contenedor en modo interactivo.
- `--rm` se utiliza para eliminar el contenedor cuando se detenga.
- `-v ~/.aws:/root/.aws` es para crear un *bind mount* entre el directorio `~/.aws` del host con el directorio `/root/.aws` del contenedor. El directorio del host `~/.aws` tiene que almacenar las credenciales de AWS.
- `amazon/aws-cli` es el nombre del repositorio de la imagen. Como no se ha indicado una etiqueta se utilizará la imagen que está etiquetada como `latest`.
- `<COMMAND>` tendrá que reemplazarse por el comando de AWS CLI que queremos ejecutar.

1.3 Configuración de AWS CLI

1.3.1 Opción 1. Con el comando `aws configure`

Para configurar AWS CLI ejecutaremos el siguiente comando.

```
aws configure
```

Este comandos nos preguntará estos datos:

```
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

Y creará un archivo de texto llamado `credentials` dentro del directorio home del usuario.

- En Linux/macOs el archivo estará en la ruta: `~/.aws/credentials`.
- En Windows estará en la ruta: `C:\Users\usuario\.aws\credentials`.

1.3.2 Opción 2. Copiando el contenido del archivo `credentials` de AWS Academy

Si queremos utilizar las credenciales de **AWS Academy** solo tenemos que copiar en el archivo `~/.aws/credentials` los datos que nos aparecen en el apartado **AWS Details** -> **Cloud Access** -> **AWS CLI**, dentro del **Learner Lab** de AWS Academy.

Ejemplo de un archivo `credentials`

```
[default]
aws_access_key_id=BSICYBW38QHVAVV7P365
aws_secret_access_key=DE/CB2FGCx8EV34x0EtuZfJg39E7hapZ9suhXBzF
aws_session_token=
    FwoGZXIvYXdzECkaDAGFN5okYCHateFXISK9AYwT4gghzz5hIr6TZs9X8pQaOYPZvTCNSrLtJoaw
    /fEwz9...XBzH
```

Después de copiar el contenido del archivo tenemos que ejecutar el comando:

```
aws configure
```

para **configurar la región de AWS que vamos a utilizar** y el **formato de salida** de los mensajes que nos devuelve la ejecución de cada comando.

En nuestro caso seleccionaremos:

- Región: `us-east-1`
- Formato de salida: `json`

```
AWS Access Key ID [****************P365]:
AWS Secret Access Key [****************XBzH]:
Default region name [None]: us-east-1
Default output format [None]: json
```

Los valores de configuración de la región y el formato de salida se almacenan en un archivo de texto llamado `config` dentro del directorio `.aws`.

Ejemplo de un archivo `config`

```
[default]
region = us-east-1
output = json
```

1.4 Creación de un grupo de seguridad

Sintaxis

```
aws ec2 create-security-group
        --description <value>
        --group-name <value>
        [--vpc-id <value>]
        [--tag-specifications <value>]
        [--dry-run | --no-dry-run]
        [--cli-input-json | --cli-input-yaml]
        [--generate-cli-skeleton <value>]
```

Documentación oficial para `create-security-group`.

Ejemplo

En este ejemplo vamos a crear un grupo con los siguientes parámetros:

- Nombre del grupo (`--group-name`): `frontend-sg`.
- Descripción (`--description`): `"Reglas para el frontend`".

```
aws ec2 create-security-group \
    --group-name frontend-sg \
    --description "Reglas para el frontend"
```

1.5 Obtener información de todos los grupos de seguridad

Sintaxis

```
aws ec2 describe-security-groups
        [--filters <value>]
        [--group-ids <value>]
        [--group-names <value>]
        [--dry-run | --no-dry-run]
        [--cli-input-json | --cli-input-yaml]
        [--starting-token <value>]
        [--page-size <value>]
        [--max-items <value>]
        [--generate-cli-skeleton <value>]
```

Documentación oficial para `describe-security-groups`.

Ejemplo

```
aws ec2 describe-security-groups
```

1.6 Obtener información de un grupo de seguridad

1.6.1 Utilizando el nombre del grupo

Ejemplo

Obtener información del grupo de seguridad que tiene el nombre `frontend-sg`.

```
aws ec2 describe-security-groups \
    --group-name frontend-sg
```

1.6.2 Utilizando el identificador

Ejemplo

Obtener información del grupo de seguridad que tiene el identificador `sg-0c0946d3ed5a9e3df`.

```
aws ec2 describe-security-groups \
    --group-ids sg-0c0946d3ed5a9e3df
```

1.7 Añadir reglas de entrada al grupo de seguridad

Sintaxis

```
aws ec2 authorize-security-group-ingress
        [--group-id <value>]
        [--group-name <value>]
        [--ip-permissions <value>]
        [--dry-run | --no-dry-run]
        [--tag-specifications <value>]
        [--protocol <value>]
        [--port <value>]
        [--cidr <value>]
        [--source-group <value>]
        [--group-owner <value>]
        [--cli-input-json | --cli-input-yaml]
        [--generate-cli-skeleton <value>]
```

Documentación oficial para `authorize-security-group-ingress`.

1.7.1 Acceso SSH desde cualquier dirección IP

Ejemplo

Añadir una regla al grupo `frontend-sg` donde se permite el acceso al puerto `22/tcp` desde cualquier dirección IP.

```
aws ec2 authorize-security-group-ingress \
    --group-name frontend-sg \
    --protocol tcp \
    --port 22 \
    --cidr 0.0.0.0/0
```

1.7.2 Acceso HTTP desde cualquier dirección IP

Ejemplo

Añadir una regla al grupo `frontend-sg` donde se permite el acceso al puerto `80/tcp` desde cualquier dirección IP.

```
aws ec2 authorize-security-group-ingress \
    --group-name frontend-sg \
    --protocol tcp \
    --port 80 \
    --cidr 0.0.0.0/0
```

1.7.3 Acceso HTTPS desde cualquier dirección IP

Ejemplo

Añadir una regla al grupo `frontend-sg` donde se permite el acceso al puerto `443/tcp` desde cualquier dirección IP.

```
aws ec2 authorize-security-group-ingress \
    --group-name frontend-sg \
    --protocol tcp \
    --port 443 \
    --cidr 0.0.0.0/0
```

1.8 Eliminar un grupo de seguridad

Sintaxis

```
aws ec2 delete-security-group
        [--group-id <value>]
        [--group-name <value>]
        [--dry-run | --no-dry-run]
        [--cli-input-json | --cli-input-yaml]
        [--generate-cli-skeleton <value>]
```

Documentación oficial para `delete-security-group`.

Ejemplo

En este ejemplo vamos a eliminar el grupo de seguridad con el nombre `frontend-sg`.

```
aws ec2 delete-security-group \
        --group-name frontend-sg
```

1.9 Eliminar todos los grupos de seguridad

```
aws ec2 delete-security-group \
        --group-id $(aws ec2 describe-security-groups \
                        --query "SecurityGroups[*].GroupId" \
                        --output text)
```

```
# Guardamos una lista con todos los identificadores de los grupos de seguridad
SG_ID_LIST=$(aws ec2 describe-security-groups \
            --query "SecurityGroups[*].GroupId" \
            --output text)

# Recorremos la lista de ids y eliminamos los grupos
for ID in $SG_ID_LIST
do
    echo "Eliminando $ID ..."
    aws ec2 delete-security-group --group-id $ID
done
```

1.10 Crear instancias EC2

Sintaxis

```
aws ec2 run-instances
        [--block-device-mappings <value>]
        [--image-id <value>]
        [--instance-type <value>]
        [--ipv6-address-count <value>]
        [--ipv6-addresses <value>]
        [--kernel-id <value>]
        [--key-name <value>]
        [--monitoring <value>]
        [--placement <value>]
        [--ramdisk-id <value>]
        [--security-group-ids <value>]
        [--security-groups <value>]
        [--subnet-id <value>]
        [--user-data <value>]
        [--additional-info <value>]
        [--client-token <value>]
        [--disable-api-termination | --enable-api-termination]
```

```
19          [--dry-run | --no-dry-run]
20          [--ebs-optimized | --no-ebs-optimized]
21          [--iam-instance-profile <value>]
22          [--instance-initiated-shutdown-behavior <value>]
23          [--network-interfaces <value>]
24          [--private-ip-address <value>]
25          [--elastic-gpu-specification <value>]
26          [--elastic-inference-accelerators <value>]
27          [--tag-specifications <value>]
28          [--launch-template <value>]
29          [--instance-market-options <value>]
30          [--credit-specification <value>]
31          [--cpu-options <value>]
32          [--capacity-reservation-specification <value>]
33          [--hibernation-options <value>]
34          [--license-specifications <value>]
35          [--metadata-options <value>]
36          [--enclave-options <value>]
37          [--count <value>]
38          [--secondary-private-ip-addresses <value>]
39          [--secondary-private-ip-address-count <value>]
40          [--associate-public-ip-address | --no-associate-public-ip-address]
41          [--cli-input-json | --cli-input-yaml]
42          [--generate-cli-skeleton <value>]
```

Documentación oficial para `run-instances`.

Ejemplo:

En este ejemplo vamos a crear una instancia EC2 con los siguientes parámetros:

- Identificador de la AMI: `ami-050406429a71aaa64`. Esta AMI se corresponde con la imagen `Debian 11 (HVM)`.
- Número de instancias: 1
- Tipo de instancia: `t2.micro`
- Clave privada: `vockey`
- Grupo de seguridad: `frontend-sg`
- Nombre de la instancia: `frontend-01`

```
1 aws ec2 run-instances \
2     --image-id ami-050406429a71aaa64 \
3     --count 1 \
4     --instance-type t2.micro \
5     --key-name vockey \
6     --security-groups frontend-sg \
7     --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=frontend
        -01}]"
```

1.10.1 Ejecutar comandos durante la creación de una instancia

Es posible ejecutar comandos durante la creación de una instancia EC2. Para ello, se puede utilizar el parámetro `--user-data` e indicar con una cadena de texto o con un script cuáles son los comandos que queremos ejecutar durante la creación de la instancia.

Ejemplo:

El siguente ejemplo crea una instancia EC2 y ejecuta los comandos `sudo apt update && sudo apt install -y nginx` durante la creación de la instancia.

```
aws ec2 run-instances \
    --image-id ami-050406429a71aaa64 \
    --count 1 \
    --instance-type t2.micro \
    --key-name vockey \
    --security-groups frontend-sg \
    --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=frontend
        -01}]" \
    --user-data "sudo apt update && sudo apt install -y nginx"
```

Ejemplo:

En este ejemplo vamos ejecutar un script de bash durante la creación de la instancia. En este ejemplo el script se llama `install_nginx.sh` y el contenido del script es el siguiente:

```
#!/bin/bash
sudo apt update
sudo apt install -y nginx
```

El comando de creación de la instancia es el siguiente:

```
aws ec2 run-instances \
    --image-id ami-050406429a71aaa64 \
    --count 1 \
    --instance-type t2.micro \
    --key-name vockey \
    --security-groups frontend-sg \
    --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=frontend
        -01}]" \
    --user-data file://install_nginx.sh
```

1.11 Obtener la lista de instancias EC2

Sintaxis

```
aws ec2 describe-instances
        [--filters <value>]
        [--instance-ids <value>]
        [--dry-run | --no-dry-run]
        [--cli-input-json | --cli-input-yaml]
        [--starting-token <value>]
        [--page-size <value>]
        [--max-items <value>]
        [--generate-cli-skeleton <value>]
```

Documentación oficial para `describe-instances`.

Ejemplo

```
aws ec2 describe-instances
```

1.12 Obtener los datos de una instancia a partir de su nombre

Ejemplo

En este ejemplo vamos a obtener todos los datos de la instancia que tiene el valor `frontend-01` en la etiqueta `Name`.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=frontend-01"
```

1.12.1 Ip Privada

Ejemplo

En este ejemplo vamos a obtener la dirección IP privada de la instancia que tiene el valor `frontend-01` en la etiqueta `Name`.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=frontend-01" \
    --query "Reservations[*].Instances[*].PrivateIpAddress" \
    --output text
```

1.12.2 Ip Pública

Ejemplo

En este ejemplo vamos a obtener la dirección IP pública de la instancia que tiene el valor `frontend-01` en la etiqueta `Name`.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=frontend-01" \
    --query "Reservations[*].Instances[*].PublicIpAddress" \
    --output text
```

1.12.3 PublicDnsName

Ejemplo

En este ejemplo vamos a obtener el nombre públic DNS de la instancia que tiene el valor `frontend-01` en la etiqueta `Name`.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=frontend-01" \
```

```
    --query "Reservations[*].Instances[*].PublicDnsName" \
    --output text
```

1.12.4 InstanceId

Ejemplo

En este ejemplo vamos a obtener el identificador de la instancia que tiene el valor `frontend-01` en la etiqueta `Name`.

```
aws ec2 describe-instances \
    --filters "Name=tag:Name,Values=frontend-01" \
    --query "Reservations[*].Instances[*].InstanceId" \
    --output text
```

1.13 Terminar una instancia EC2

Sintaxis

```
aws ec2 terminate-instances
        --instance-ids <value>
        [--dry-run | --no-dry-run]
        [--cli-input-json | --cli-input-yaml]
        [--generate-cli-skeleton <value>]
```

Documentación oficial para `terminate-instances`.

Ejemplo

En este ejemplo vamos a eliminar la instancia que tiene el identificador `i-0dd20d987f751ae56`.

```
aws ec2 terminate-instances \
    --instance-ids i-0dd20d987f751ae56
```

1.14 Terminar todas las instancias EC2

Para Linux/macOS

```
aws ec2 terminate-instances \
    --instance-ids $(aws ec2 describe-instances \
                     --query "Reservations[*].Instances[*].InstanceId" \
                     --output text)
```

1.15 Crear una IP elástica

Sintaxis

```
aws ec2 allocate-address
        [--domain <value>]
        [--address <value>]
        [--public-ipv4-pool <value>]
        [--network-border-group <value>]
        [--customer-owned-ipv4-pool <value>]
        [--dry-run | --no-dry-run]
        [--tag-specifications <value>]
        [--cli-input-json <value>]
        [--generate-cli-skeleton <value>]
```

Documentación oficial para `allocate-address`.

Ejemplo

En este ejemplo vamos a crear una dirección IP elástica y la vamos a mostrar en texto plano.

```
aws ec2 allocate-address --query PublicIp --output text
```

1.16 Scripts de ejemplo

1.16.1 terminate_all_instances.sh

Este script elimina todas las instancias EC2 que existan.

```
#!/bin/bash
set -x

# Deshabilitamos la paginación de la salida de los comandos de AWS CLI
# Referencia: https://docs.aws.amazon.com/es_es/cli/latest/userguide/cliv2-
    migration.html#cliv2-migration-output-pager
export AWS_PAGER=""

# Eliminamos todas las intancias
aws ec2 terminate-instances --instance-ids $(aws ec2 describe-instances --query
     "Reservations[*].Instances[*].InstanceId" --output text)
```

1.16.2 delete_all_security_groups.sh

Este script elimina todos los grupos de seguridad que existan.

```
#!/bin/bash
set -x

# Deshabilitamos la paginación de la salida de los comandos de AWS CLI
# Referencia: https://docs.aws.amazon.com/es_es/cli/latest/userguide/cliv2-
    migration.html#cliv2-migration-output-pager
export AWS_PAGER=""

# Guardamos una lista con todos los identificadores de las instancias EC2
```

```
SG_ID_LIST=$(aws ec2 describe-security-groups \
            --query "SecurityGroups[*].GroupId" \
            --output text)

# Recorremos la lista de ids y eliminamos las instancias
for ID in $SG_ID_LIST
do
    echo "Eliminando $ID ..."
    aws ec2 delete-security-group --group-id $ID
done
```

1.17 Repositorio con scripts de ejemplo

En GitHub puede encontrar un repositorio con scripts de ejemplo que le pueden ser de utilidad.

- https://github.com/josejuansanchez/practica-aws-cli

1.18 Ejercicios

1.18.1 Ejercicio 1

- Crea un grupo de seguridad para las máquinas del Backend con el nombre `backend-sg`.
- Añada las siguientes reglas al grupo de seguridad:
  - Acceso SSH (puerto `22/TCP`) desde cualquier dirección IP.
  - Acceso al puerto `3306/TCP` desde cualquier dirección IP.

1.18.2 Ejercicio 2

- Crea una instancia EC2 para la máquina del Backend con las siguientes características.
  - Identificador de la AMI: `ami-08e637cea2f053dfa`. Esta AMI se corresponde con la imagen `Red Hat Enterprise Linux 9 (HVM)`.
  - Número de instancias: 1
  - Tipo de instancia: `t2.micro`
  - Clave privada: `vockey`
  - Grupo de seguridad: `backend-sg`
  - Nombre de la instancia: `backend`

1.18.3 Ejercicio 3

- Crear un script para crear la infraestructura de la práctica propuesta por el profesor.
- Crear un script para eliminar la infraestructura de la práctica propuesta por el profesor.

1.18.4 Ejercicio 4

Modifique los scripts del repositorio de ejemplo:

- https://github.com/josejuansanchez/practica-aws-cli

para que utilicen la AMI de la última versión de `Ubuntu Server`.

También tendrá que modificar los scripts para que se ejecute el siguiente comando en las instancias durante el inicio.

```
$ sudo apt update && sudo apt upgrade -y
```

En la documentación oficial puede encontrar más información sobre cómo ejecutar comandos en una instancia durante el inicio.

1.18.5 Ejercicio 5

Escriba un script de bash que muestre el nombre de todas instancias EC2 que tiene en ejecución junto a su dirección IP pública.

1.19 Referencias

- Documentación de AWS Command Line Interface

2 Licencia