

Practica 6 AWS. El modo consola



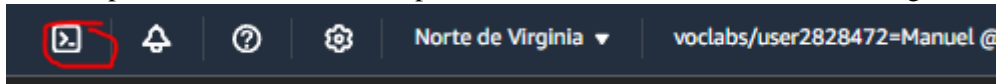
Contenido

Introduccion a AWS CLI	3
1. Acceso a la consola desde entorno gráfico de AWS.....	3
2. Instalación de AWS CLI en el SO y configurar AWS CLI en nuestro ordenador	6
3. Listar todas las VPC	7
4. Crear grupo de seguridad desde la consola de AWS (online)	7
5. Ver instancias	9
6. Crear 1 instancia EC2 en el grupo de seguridad anterior (online)	9
7. Lo anterior, pero con script Linux en vez de comando en consola	10
8. Modifica el anterior script para que lanzar 2 instancias y las actualice al lanzar	12
9. Elimina las instancias creadas desde PS.....	12

Introducción a AWS CLI

1. Acceso a la consola desde entorno gráfico de AWS

Esta práctica vamos a trabajar el modo consola de AWS. Usaremos la Shell de aws para diferentes operaciones. Comenzamos por conectarnos a la cli de aws desde el logo de la Shell



Vamos a configurar las **credenciales y parámetros básicos** para que la **CLI de AWS (Command Line Interface)** pueda interactuar con los servicios de AWS de forma autenticada y personalizada.

Para ello escribiremos el comando `aws configure`. Este comando crea el archivo de configuración. Para configurar, el sistema te pedirá una serie de valores que se almacenarán en archivos de configuración dentro de tu máquina local. Estos valores incluyen:

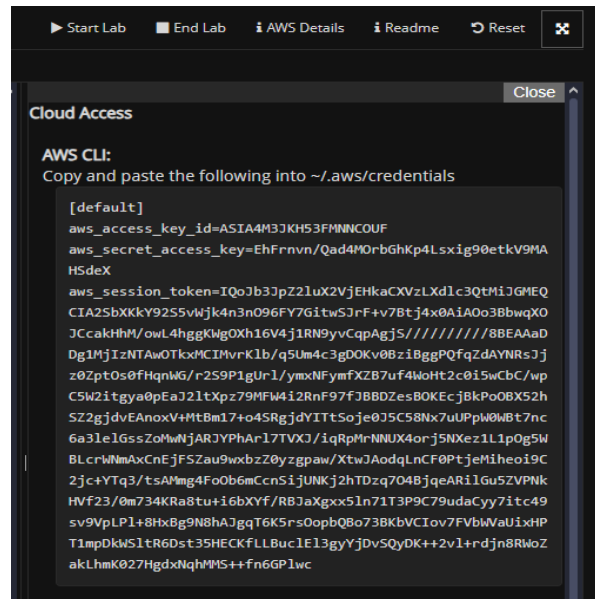
1. **AWS Access Key ID** - Este es el **ID de clave de acceso** asociado a tu cuenta de AWS, que se utiliza para autenticar solicitudes desde la CLI.
2. **AWS Secret Access Key** - Es la **clave secreta** que se utiliza junto con el Access Key ID para firmar tus solicitudes de AWS.

Ambas claves (Access Key ID y Secret Access Key) se generan desde la **Consola de Gestión de AWS** en la sección de **IAM** (Identity and Access Management) cuando creas un usuario o clave de acceso. Nosotros las copiamos desde la pantalla del lanzamiento del laboratorio.

3. **Default region name** - Define la **región por defecto** en la que se realizarán las solicitudes si no especificas una región en los comandos. En el laboratorio us-east-1 (puede cambiar)
4. **Default output format - formato de salida** de los comandos de AWS CLI. Los formatos más comunes son: `json`, `text`, `table`

Para ello vamos a AWS details->Cloud Access AWS CLI

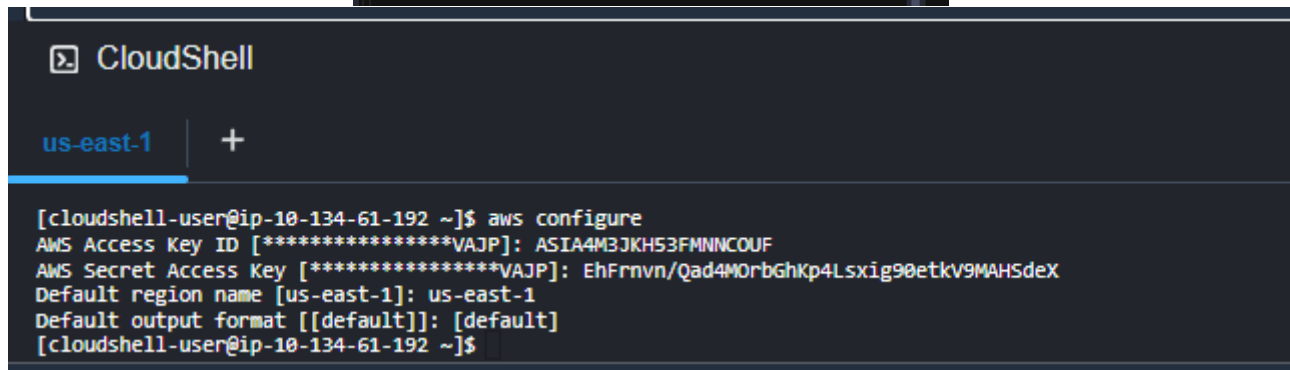
Le damos en details y show aws cli y copiamos y pegamos en la consola



Cloud Access

AWS CLI:
Copy and paste the following into ~/.aws/credentials

```
[default]
aws_access_key_id=ASIA4M3JKH53FMNNCOUF
aws_secret_access_key=EhFrnnv/Qad4M0rbGhKp4Lsxig90etkv9MAHSdex
aws_session_token=IQo7b3pZ2LwX2VjEHkaCXVzLXd1c3QtMiJGMtEQ
CIA2SbXKkY92S5vWjk4n3n096FY7G1tw5JrF+v7Btj4x0AIA0o3BbwqXO
JCcakHhM/owL4hggKWgOXh16V4j1RN9yvCqpAgjS/////////8BEAAAD
Dg1MjIzNTAwOTkxMCIMvrK1b/q5Um4c3gDOKv0BziBggPQfqZdAYNRs7j
z0Zpt0s0fhqNwG/r2S9P1gUr1/ymxNFymfXZB7uf4WoHt2c0i5wCbC/wp
CSW2itgya0pEaJ2ItXpz79MFw4i2RnF97fJ8BDZesB0KEc-jBkPo0BX52h
SZ2gjdVdEAnoxV+MtBm17+o4SRgjdYITtSoje0J5C58Nx7uUPplw0Bt7nc
6a3le1GssZoMwMjARJYPhAr17TVXJ/iqRpMrNNUX4orj5NXez1L1p0g5W
BLcrWmAxCnEjFSZau9wxbzZ0yzgpaw/XtwJAodqLnCF0PtjeMiheo19C
2jc+YTq3/tsAMmg4FoOb6mCcnSijUNKj2hTDzq704BjqeARilGu5ZVPnk
HVF23/0m734KR8tu+i6bXYf/RBjaXgxx5ln71T3P9C79udaCyy7itc49
sv9VpLPl+8HxBg9N8hA7gqT6K5rs0opbQ8o73BkbVCIov7FVbWuUixHP
T1mpDkWSltR6DSt35HECKFLBuc1E13gyYjDv5QyDK++2v1+rdjn8RwoZ
akLhmK027HgdXNqhMMS++fn6GP1wc
```



CloudShell

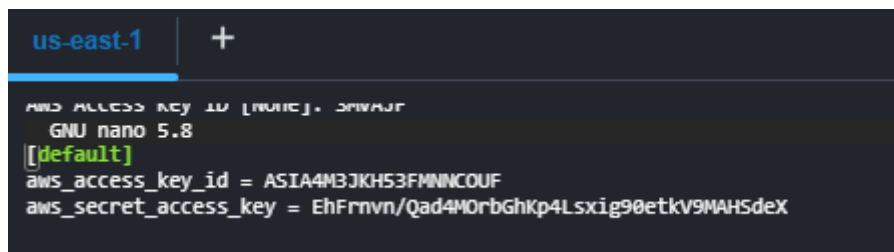
us-east-1 +

```
[cloudshell-user@ip-10-134-61-192 ~]$ aws configure
AWS Access Key ID [*****VAJP]: ASIA4M3JKH53FMNNCOUF
AWS Secret Access Key [*****VAJP]: EhFrnnv/Qad4M0rbGhKp4Lsxig90etkv9MAHSdex
Default region name [us-east-1]: us-east-1
Default output format [[default]]: [default]
[cloudshell-user@ip-10-134-61-192 ~]$
```

Una vez configurada la CLI de AWS, puedes ejecutar comandos de AWS directamente desde la terminal.

```
[cloudshell-user@ip-10-134-54-46 ~]$ nano .aws/credentials
```

Y sustituimos con las credenciales que hemos copiado en AWS detail



us-east-1 +

```
AWS ACCESS KEY ID [VAJP]. 3P1W4JF
GNU nano 5.8
[default]
aws_access_key_id = ASIA4M3JKH53FMNNCOUF
aws_secret_access_key = EhFrnnv/Qad4M0rbGhKp4Lsxig90etkv9MAHSdex
```

2. Instalación de AWS CLI en el SO y configurar AWS CLI en nuestro ordenador

Instalación en Linux Para realizar la instalación de AWS CLI en un sistema operativo Linux descargamos un archivo .zip con la aplicación AWS CLI.

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

```
alumno@alumno:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %          Dload  Upload  Total   Spent    Left   Speed
 45  63.0M    45  28.9M    0     0   7956k      0  0:00:08  0:00:03  0:00:05  7954k
```

Descomprimos el archivo que acabamos de descargar y ejecutamos el script de instalación y por ultimo comprobamos que la instalación se ha realizado de forma correcta.

```
unzip awscliv2.zip
```

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

```
aws --version
```

```
alumno@alumno:~/Descargas/awscliv2/aws$ sudo ./install
You can now run: /usr/local/bin/aws --version
alumno@alumno:~/Descargas/awscliv2/aws$ aws --version

usage: aws [options] <command> [<subcommand> [<subcommand> ...] [parameters]]
To see help text, you can run:

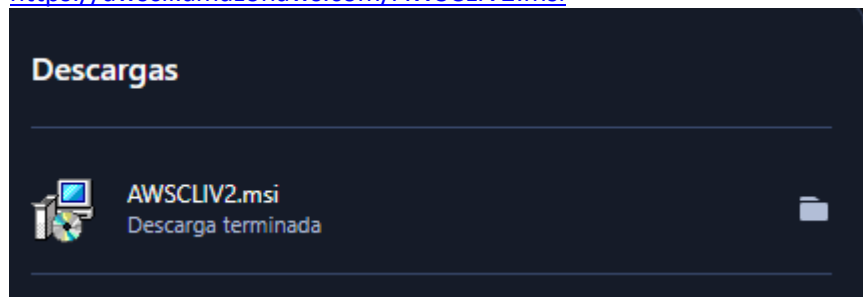
    aws help
    aws <command> help
    aws <command> <subcommand> help

aws: error: the following arguments are required: command

alumno@alumno:~/Descargas/awscliv2/aws$ aws help
[1]+  Detenido                  aws help
alumno@alumno:~/Descargas/awscliv2/aws$ aws --version
aws-cli/2.18.5 Python/3.12.6 Linux/6.2.0-26-generic exe/x86_64.ubuntu.22
alumno@alumno:~/Descargas/awscliv2/aws$
```

Instalación en Windows descárgatelo desde el navegador y después instálalo. Puedes buscar “cliente AWS cli” o cogerlo de los siguientes enlaces. Queda instalado en PowerShell

<https://awscli.amazonaws.com/AWSCLIV2.msi>

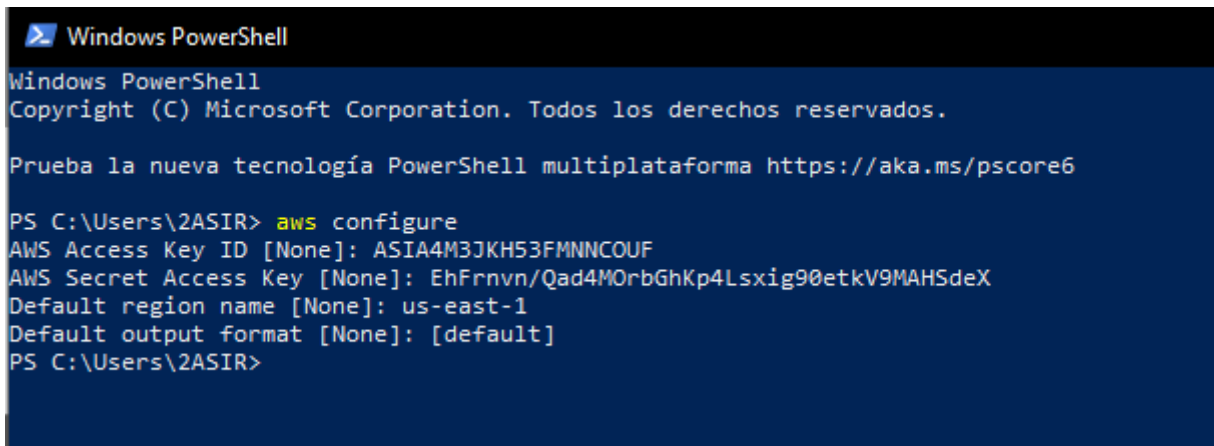


o en cmd, escribir: [msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi](https://awscli.amazonaws.com/AWSCLIV2.msi)

Si todo ha ido bien puedes ejecutar el comando **aws configure** desde PowerShell en tu equipo.

Este comando nos preguntará estos datos:

```
1 AWS Access Key ID [None]:  
2 AWS Secret Access Key [None]:  
3 Default region name [None]:  
4 Default output format [None]:
```

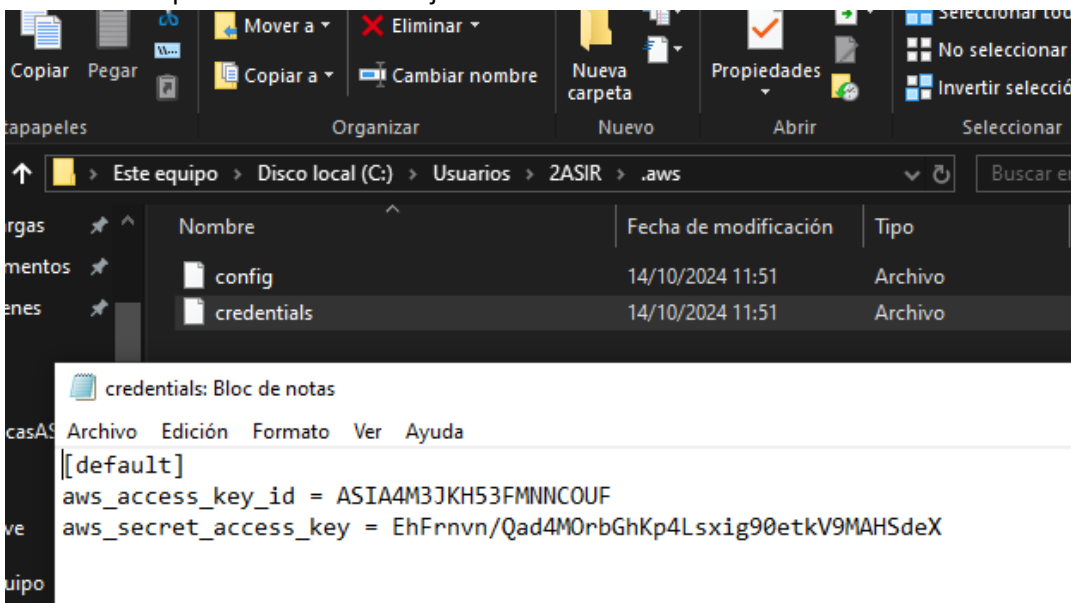


```
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos los derechos reservados.  
  
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6  
  
PS C:\Users\2ASIR> aws configure  
AWS Access Key ID [None]: ASIA4M3JKH53FMNNCOUF  
AWS Secret Access Key [None]: EhFrnvN/Qad4MOrbGhKp4Lsxig90etkV9MAHSdeX  
Default region name [None]: us-east-1  
Default output format [None]: [default]  
PS C:\Users\2ASIR>
```

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`.

Pega el token de AWS details, en el archivo `.aws/credentials`, editándolo con nano. Ahora ya sabe CLI con que usuario vamos a ejecutar los comandos.

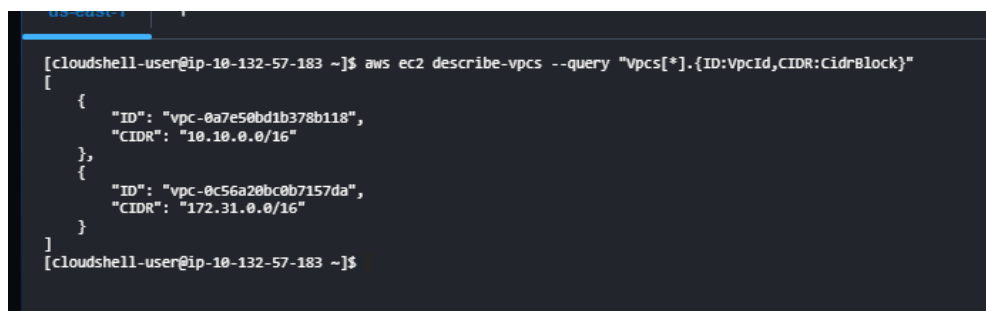


Si queremos utilizar las credenciales de **AWS Academy** solo tenemos que copiar en el archivo `~/.aws/credentials` los datos que nos aparecen en el apartado **AWS Details** -> **Cloud Access** -> **AWS CLI**, dentro del **Learner Lab** de AWS Academy.

En nuestro caso seleccionaremos: Región: `us-east-1` Formato de salida: `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`

3. Listar todas las VPC

```
aws ec2 describe-vpcs --query "Vpcs[*].{ID:VpcId,CIDR:CidrBlock}"
```



Aparecen dos en mi lista

4. Crear grupo de seguridad desde la consola de AWS (online)

- Crea un grupo de seguridad para las máquinas del Backend con el nombre practica6.
Descripción: “Reglas backend”
- 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.

EL ID de la VPC lo tomas de una de las que tengas en tu red (ver en comando anterior)

```

    "CIDR": "172.31.0.0/16"
  }
}
[cloudshell-user@ip-10-132-57-183 ~]$ aws ec2 create-security-group --group-name backend-sg --description "Reglas backend" --vpc-id vpc-0a7e50bd1b378b118
{
  "GroupId": "sg-06a2e571e0cf346f7"
}
[cloudshell-user@ip-10-132-57-183 ~]$ aws ec2 create-security-group --group-name practica6 --description "Reglas backend" --vpc-id vpc-0c56a20bc0b7157da
{
  "GroupId": "sg-03c4fc1db59833572"
}
[cloudshell-user@ip-10-132-57-183 ~]$

```

"GroupId": "sg-04b8a9f26cdd93173"

■	-	sg-03c4fc1db59833572	practica6	vpc-0c56a20bc0b7157da [?]	Reglas backend	852235009910
■	-	sg-06a2e571e0cf346f7	backend-sg	vpc-0a7e50bd1b378b118 [?]	Reglas backend	852235009910

Si vamos a la consola comprobamos que se han agregado. También podemos verlo así con el comando de listar los grupos de seguridad y verificar que el grupo practica6 está presente:

```
aws ec2 describe-security-groups --filters Name=group-name,Values=practica6 --query "SecurityGroups[*].{ID:GroupId,Name:GroupName,Description:Description}"
```

```

[cloudshell-user@ip-10-132-57-183 ~]$ aws ec2 describe-security-groups --filters Name=group-name,Values=practica6 --query "SecurityGroups[*].{ID:GroupId,Name:GroupName,Description:Description}"
[
  {
    "ID": "sg-03c4fc1db59833572",
    "Name": "practica6",
    "Description": "Reglas backend"
  }
]
[cloudshell-user@ip-10-132-57-183 ~]$

```

sg-03c4fc1db59833572

Ahora abrimos los puertos. Puedes hacerlo usando el nombre del grupo o el group id. Copia el group id anterior que has obtenido.

```
aws ec2 authorize-security-group-ingress --group-id sg-04b8a9f26cdd93173 --protocol tcp --port 22 --cidr 0.0.0.0/0
```

```
aws ec2 authorize-security-group-ingress --group-id sg-04b8a9f26cdd93173 --protocol tcp --port 3306 --cidr 0.0.0.0/0
```

```
]
[cloudshell-user@ip-10-132-57-183 ~]$ aws ec2 authorize-security-group-ingress --group-id sg-03c4fc1db59833572 --protocol tcp --port 22 --cidr 0.0.0.0/0
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-03e60cd6e611756ab",
      "GroupId": "sg-03c4fc1db59833572",
      "GroupOwnerId": "852235009910",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 22,
      "ToPort": 22,
      "CidrIpv4": "0.0.0.0/0"
    }
  ]
}
```

5. Ver instancias

Comando para ver instancias: `aws ec2 describe-instances`

```
}
[cloudshell-user@ip-10-132-57-183 ~]$ aws ec2 describe-instances
{
  "Reservations": []
}
```

Para solo las instancias running, usa el comando con un filtro: `aws ec2 describe-instances --filters "Name=instance-state-name,Values=running"`

Otros parámetros:

`--query 'Reservations[*].Instances[*].[InstanceId,State.Name]'`: Selecciona solo la ID de las instancias y su estado.

`--output table`: Formatea la salida en una tabla legible.


6. Crear 1 instancia EC2 en el grupo de seguridad anterior (online)

Características:

- Nombre: `tu_nombreP6server` (ej `RuthP6server`)
AMI: `ami-08e637cea2f053dfa`. Esta AMI es un [Red Hat Enterprise Linux 9 \(HVM\)](#).
- Tipo de instancia: `t2.micro` – Clave privada: `vockey`
- – Grupo de seguridad: `--group-id sg-03c4fc1db59833572` (pon el tuyo)
- Subred: debes buscar el identificador de la subred (probablemente haya varias en el laboratorio) donde creaste el grupo de seguridad anterior (probablemente el de por defecto): `sg-04b8a9f26cdd93173`, si no probablemente dé problemas Ej: Subred: `subnet-02e2c527d313554c5`

```
aws ec2 run-instances --image-id ami-053b0d53c279acc90 --count 1 --instance-type t2.micro -
key-name vockey --security-group-ids sg-03c4fc1db59833572 --subnet-id
subnet02e2c527d313554c5 --tag-specifications
'ResourceType=instance,Tags=[{Key=Name,Value=ManuP6Server}]'
```

Comprobamos que se ha lanzado en la terminal y también online: `aws ec2 describe-instances`

<input type="checkbox"/>	Name 	ID de la instancia
<input type="checkbox"/>	ManuP6Server	i-097879f476547d499

7. Lo anterior, pero con script Linux en vez de comando en consola

Podríamos hacerlo en la línea de comandos directamente, pero vamos a empezar a automatizar, creamos un script para lanzar dos instancias, una tras otra. Llama a la instancia `TunombreP6ServerII`

```
aws ec2 run-instances --image-id ami-053b0d53c279acc90 --count 1 --instance-type t2.micro -
key-name vockey --security-group-ids sg-0eee16c03163d3e97 --subnet-id
subnet02e2c527d313554c5 --tag-specifications
'ResourceType=instance,Tags=[{Key=Name,Value=RuthP6ServerII}]'
```

Script:

```
#!/bin/bash
# Parámetros para la instancia

IMAGE_ID="ami-053b0d53c279acc90"

INSTANCE_TYPE="t2.micro"

KEY_NAME="vockey"

SECURITY_GROUP_ID="sg-0eee16c03163d3e97"

SUBNET_ID="subnet-02e2c527d313554c5"

TAG_SPECIFICATIONS='ResourceType=instance,Tags=[{Key=Name,Value=RuthP6ServerII}]'

# Ejecutar el comando para lanzar la instancia

echo "Lanzando la instancia EC2..."

INSTANCE_ID=$(aws ec2 run-instances \
    --image-id $IMAGE_ID \
    --count 1 \
    --instance-type $INSTANCE_TYPE \
    --key-name $KEY_NAME \
    --security-group-ids $SECURITY_GROUP_ID \
    --subnet-id $SUBNET_ID \
    --tag-specifications "$TAG_SPECIFICATIONS" \
    --query 'Instances[0].InstanceId' \
    --output text)

# Comprobar si la instancia fue lanzada exitosamente if [ -z
"$INSTANCE_ID" ]; then    echo "Error al lanzar la instancia."; exit 1
else    echo "Instancia lanzada con éxito. ID de la instancia:
$INSTANCE_ID"
fi
```

Guarda el script en un archivo llamado lanzar_instancia.sh.. Dale permisos de ejecución al archivo

```
chmod +x lanzar_instancia.sh
./lanzar_instancia.sh
```

8. Modifica el anterior script para que lanzar 2 instancias y las actualice al lanzar

Llama el script lanzar_instancia2.sh Nota: Para los datos finales, bien lo pones en la opción `--user-data` “a pelo”, o bien creas un segundo script llamado “actualiza.sh”.


```
#!/bin/bash

ami_id="ami-053b0d53c279acc90"
instance_type="t2.micro"
key_name="vockey"
security_group_id="sg-00966f12139b5804c" #aqui hay que poner el id de tu grupo de seguridad

num_instancias=2

for ((i=1; i<=num_instancias; i++))
do
    nombre_instancia="Script$i" # Nombre de la instancia
    echo "Lanzando la instancia $nombre_instancia..."

    instance_id=$(aws ec2 run-instances \
        --image-id "$ami_id" \
        --instance-type "$instance_type" \
        --key-name "$key_name" \
        --security-group-ids "$security_group_id" \
        --query 'Instances[0].InstanceId' \
        --output text \
        --user-data "sudo apt update && sudo apt upgrade -y" \
```



La otra opción, mejor si los comandos al final son varios, es crear un segundo script con los comandos que quieres ejecutar. En este ejemplo el script lo llamaremos “actualiza.sh” y el contenido del script es el siguiente:

```
1 #!/bin/bash
2 sudo apt update
3 sudo apt upgrade
```

9. Elimina las instancias creadas desde PS