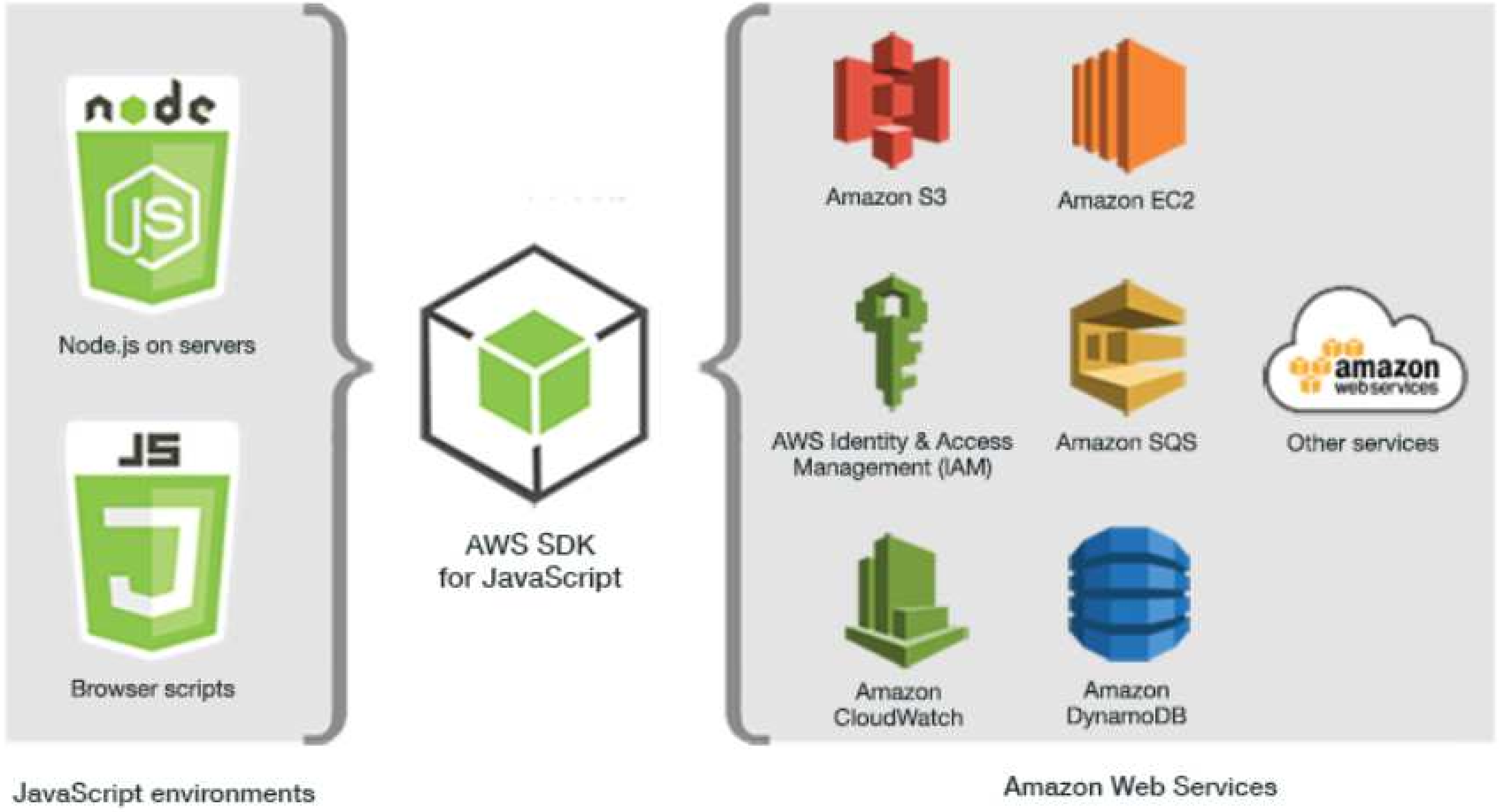
4.2

AWS CLI/SDK

Mondragon Unibertsitatea
Goi Eskola Politeknikoa

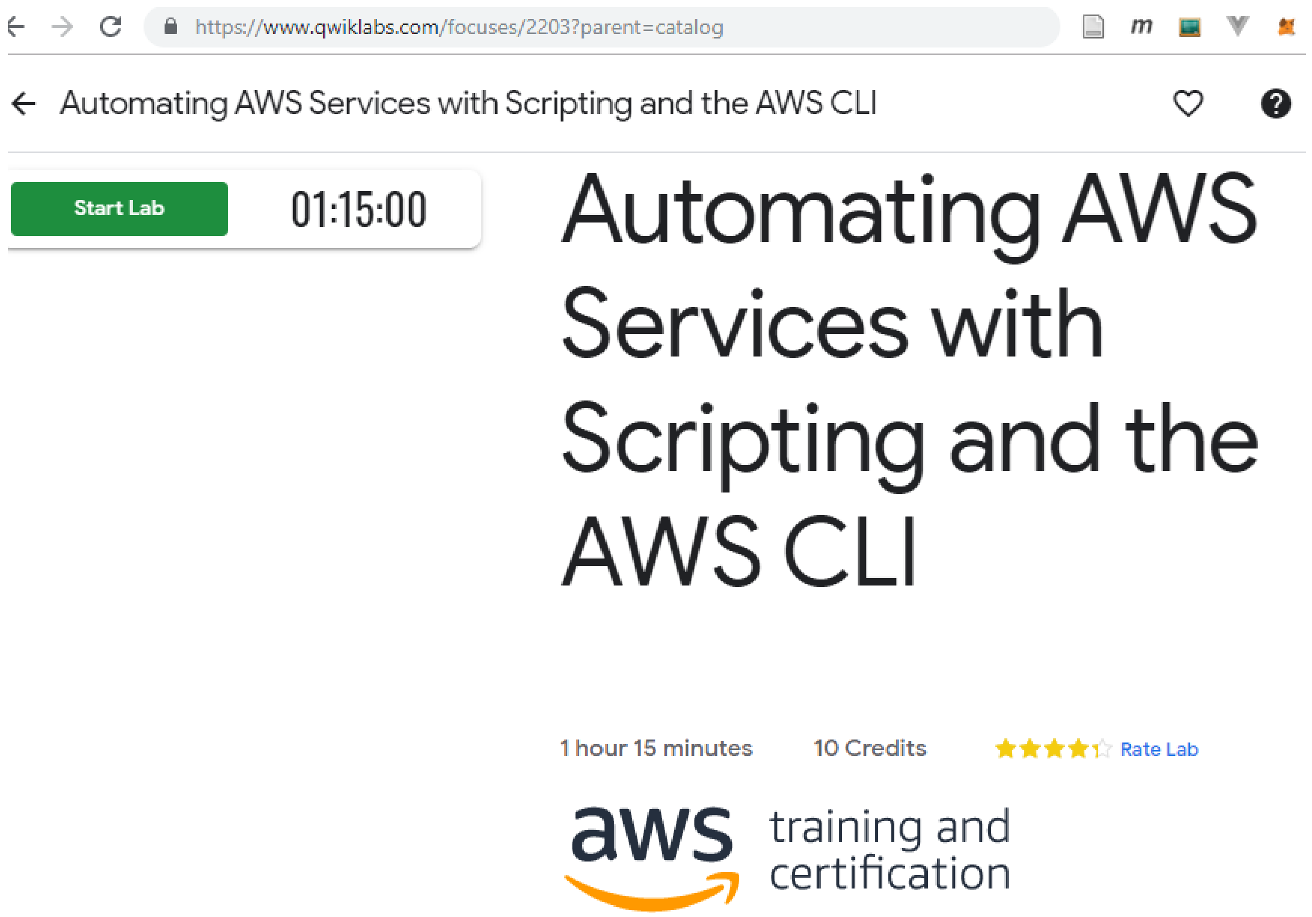
Scripting : AWS CLI

- Scripting : AWS SDK
- https://github.com/awsdocs/aws-doc-sdk-examples/tree/master/python/example_code/ec2



Scripting : AWS CLI

- QWIKSLAB: AWS CLI



https://www.qwiklabs.com/focuses/2203?parent=catalog

Automating AWS Services with Scripting and the AWS CLI

Start Lab 01:15:00

Automating AWS Services with Scripting and the AWS CLI

1 hour 15 minutes 10 Credits Rate Lab

aws training and certification

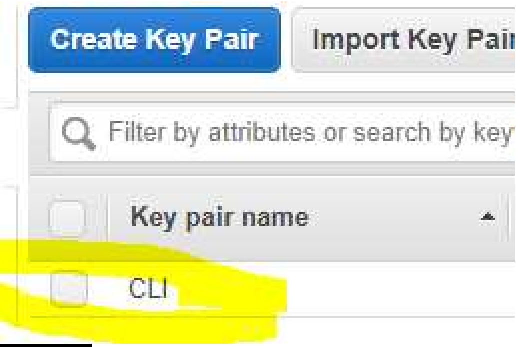
Scripting : AWS CLI

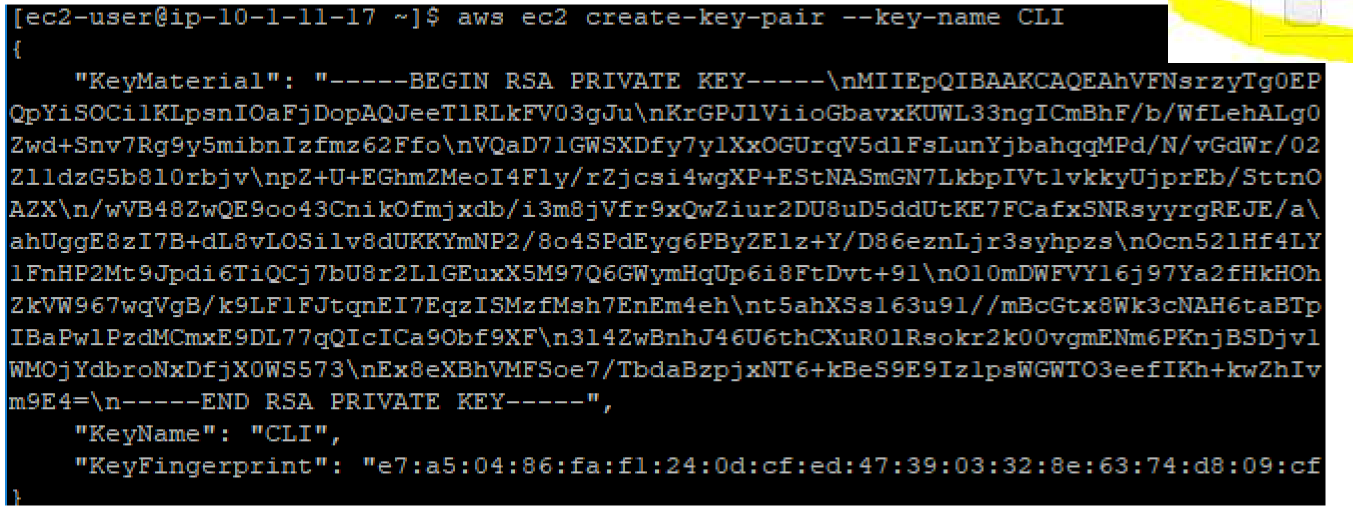
- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
  - Gestionar recursos mediante CLI
  - Gestionar recursos mediante el SDK
  - Configurar aspectos de seguridad mediante CLI

Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Creación de un par claves

```
aws ec2 create-key-pair --key-name CLI
```

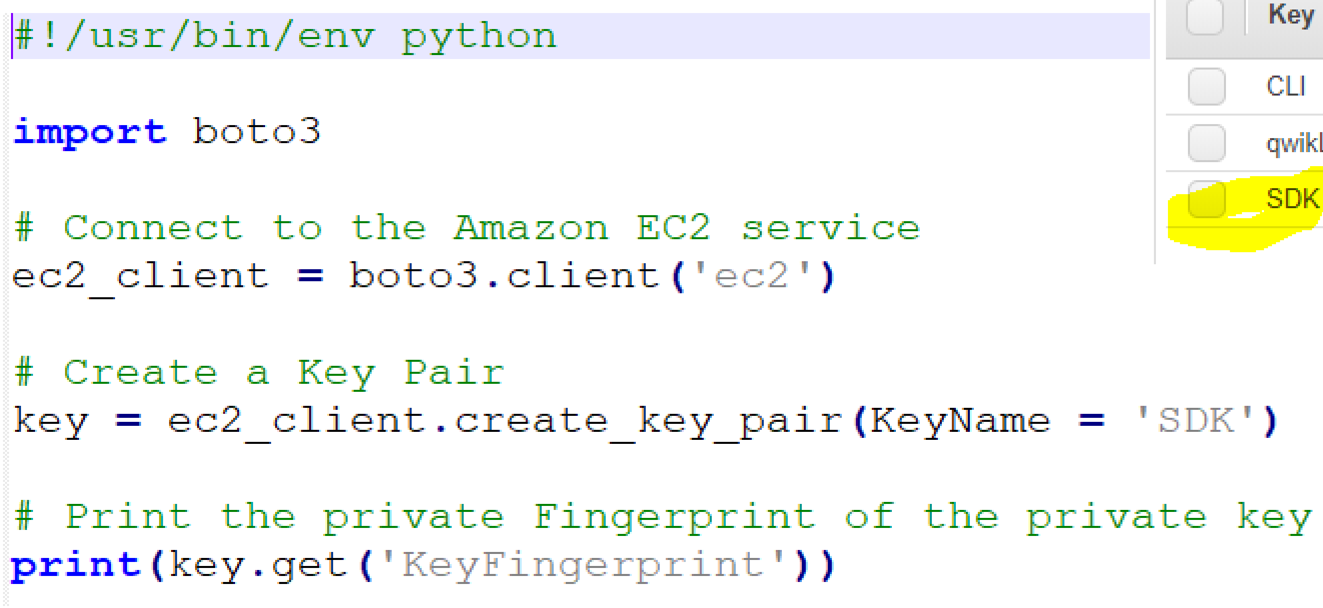




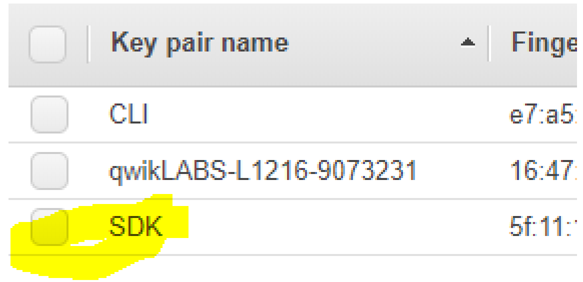
```
[ec2-user@ip-10-1-11-17 ~]$ aws ec2 create-key-pair --key-name CLI
{
    "KeyMaterial": "-----BEGIN RSA PRIVATE KEY-----\nMIIEpQIBAAKCAQEAhVFNsrzyTg0EP
QpYiSOCilKLpsnIOaFjDopAQJeeTlRLkFV03gJu\nKrGPJlViioGbavxKUWL33ngICmBhF/b/WfLehALg0
Zwd+Snv7Rg9y5mibnIzfmz62Ffo\nVQaD7lGWSXDfy7ylXxOGUrqV5dlFsLunYjbahqqMPd/N/vGdWr/02
ZlldzG5b810rbjv\npZ+U+EGhmZMeoI4Fly/rZjcsi4wgXP+EStNASmGN7LkbpIVtlvkkyUjprEb/SttnO
AZX\n/wVB48ZwQE9oo43CnikOfmjxdb/i3m8jVfr9xQwZiur2DU8uD5ddUtKE7FCafxSNRsyyrgREJE/a\
ahUggE8zI7B+dL8vLOSilv8dUKKYmNP2/8o4SPdEyg6PByZElz+Y/D86eznLjr3syhpzs\nOcn521Hf4LY
1FnHP2Mt9Jpdi6TiQCj7bU8r2LlGEuxX5M97Q6GWymHqUp6i8FtDvt+91\nOl0mDWFVY16j97Ya2fHkHOh
ZkVW967wqVgB/k9LFlFJtqnEI7EqzISMzfMsh7EnEm4eh\nt5ahXSsl63u91//mBcGtx8Wk3cNAH6taBTp
IBaPwlPzdMCmxE9DL77qQIcICa9Obf9XF\n314ZwBnhJ46U6thCXuR0lRsokr2k00vgmENm6PKnjBSDjvl
WMOjYdbroNxDfjX0WS573\nEx8eXBhVMFSoe7/TbdaBzpjxNT6+kBeS9E9IzlpsWGWTO3eefIKh+kwZhIv
m9E4=\n-----END RSA PRIVATE KEY-----",
    "KeyName": "CLI",
    "KeyFingerprint": "e7:a5:04:86:fa:f1:24:0d:cf:ed:47:39:03:32:8e:63:74:d8:09:cf
}
```

Scripting : AWS CLI

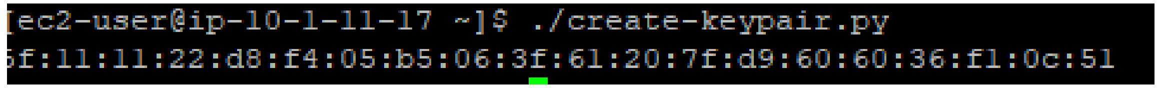
- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Creación de un par claves via SDK



```
#!/usr/bin/env python

import boto3

# Connect to the Amazon EC2 service
ec2_client = boto3.client('ec2')

# Create a Key Pair
key = ec2_client.create_key_pair(KeyName = 'SDK')

# Print the private Fingerprint of the private key
print(key.get('KeyFingerprint'))
```



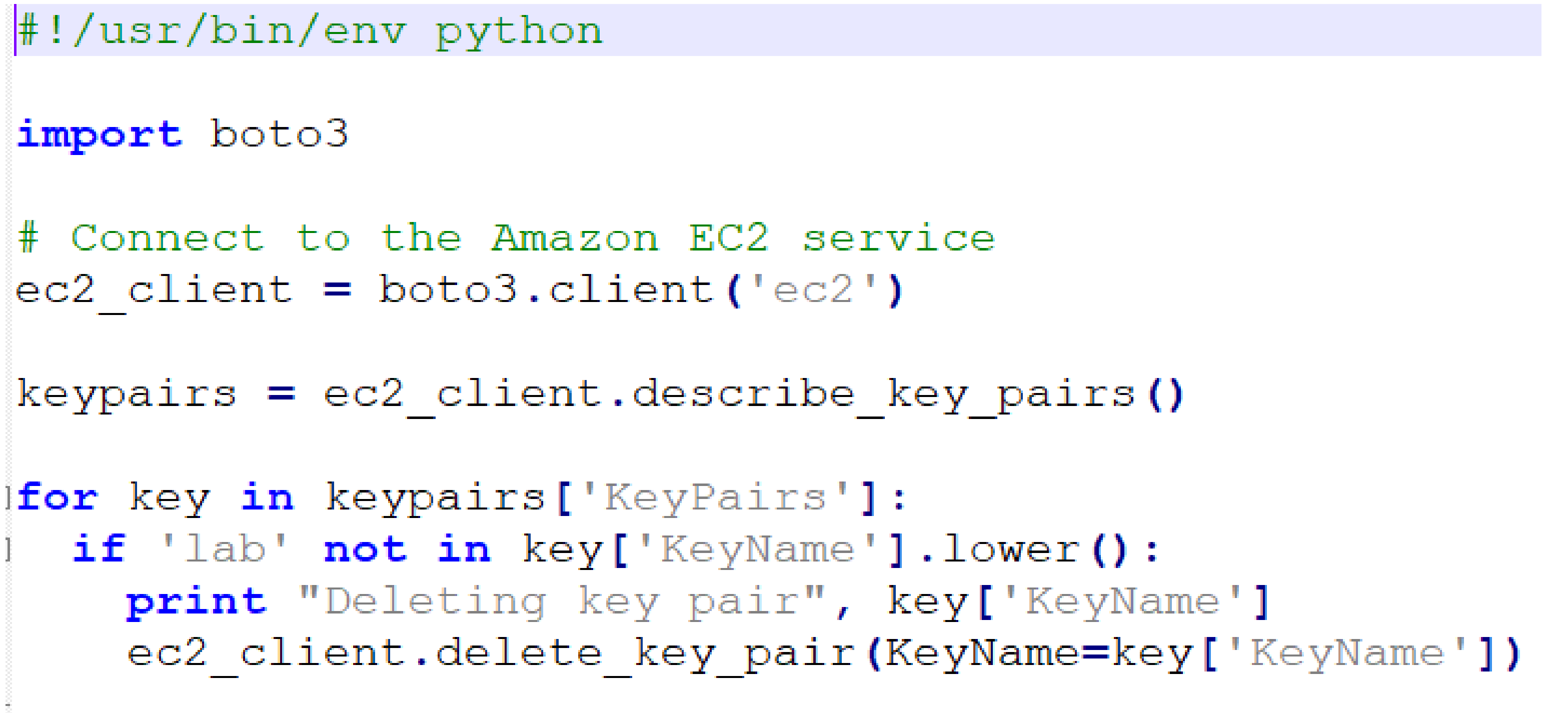
Key pair name	Finge
CLI	e7:a5
qwikLABS-L1216-9073231	16:47
SDK	5f:11:



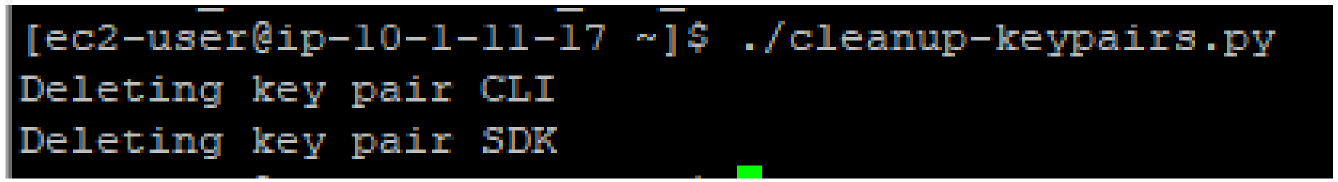
```
[ec2-user@ip-10-1-11-17 ~]$ ./create-keypair.py
5f:11:11:22:d8:f4:05:b5:06:3f:61:20:7f:d9:60:60:36:f1:0c:51
```

Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Borrando los recursos vía SDK



```python
#!/usr/bin/env python

import boto3

# Connect to the Amazon EC2 service
ec2_client = boto3.client('ec2')

keypairs = ec2_client.describe_key_pairs()

for key in keypairs['KeyPairs']:
  if 'lab' not in key['KeyName'].lower():
    print "Deleting key pair", key['KeyName']
    ec2_client.delete_key_pair(KeyName=key['KeyName'])
```

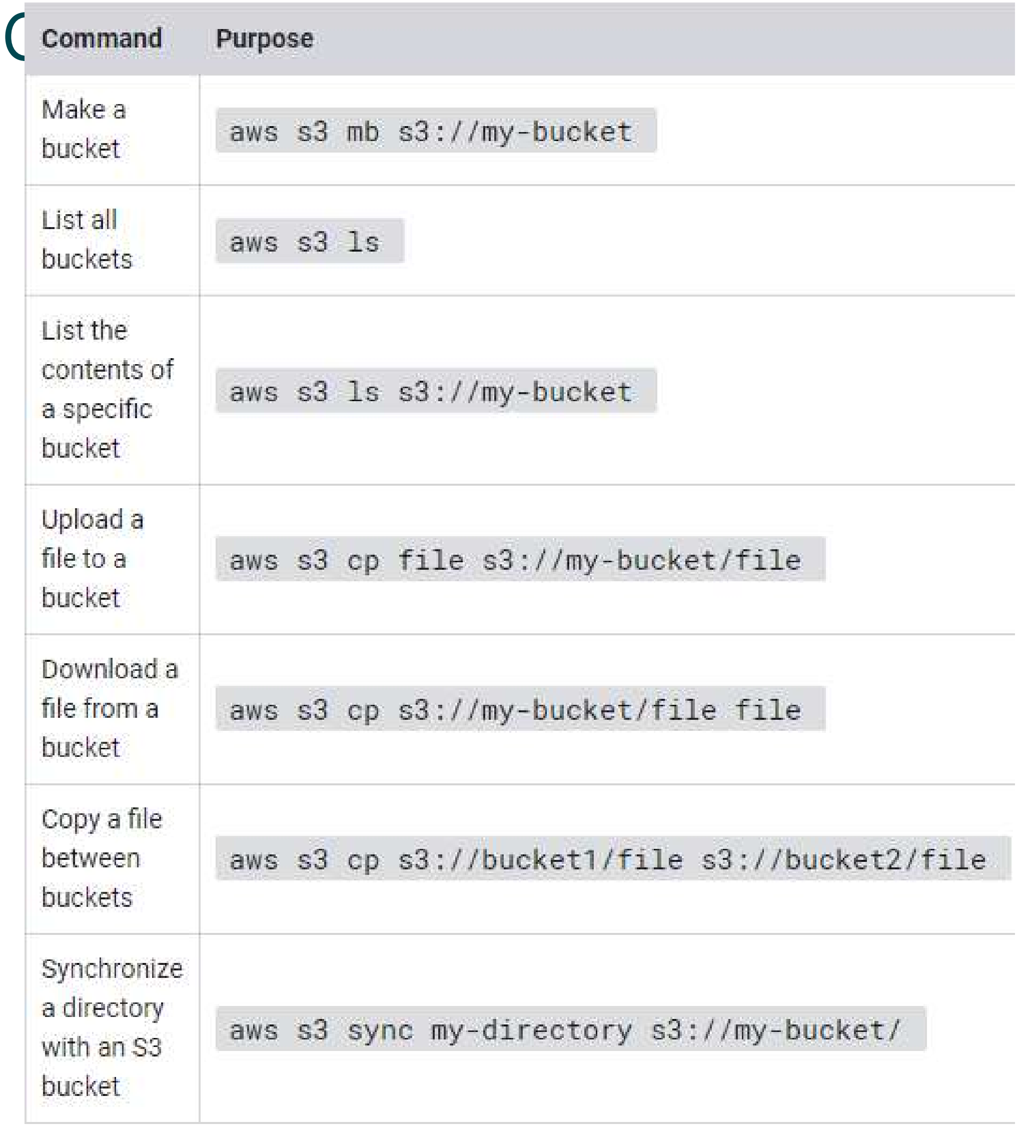


```
[ec2-user@ip-10-1-11-17 ~]$ ./cleanup-keypairs.py
Deleting key pair CLI
Deleting key pair SDK
```

Mondragon Unibertsitatea
Goi Eskola Politeknikoa

Scripting : AWS CLI

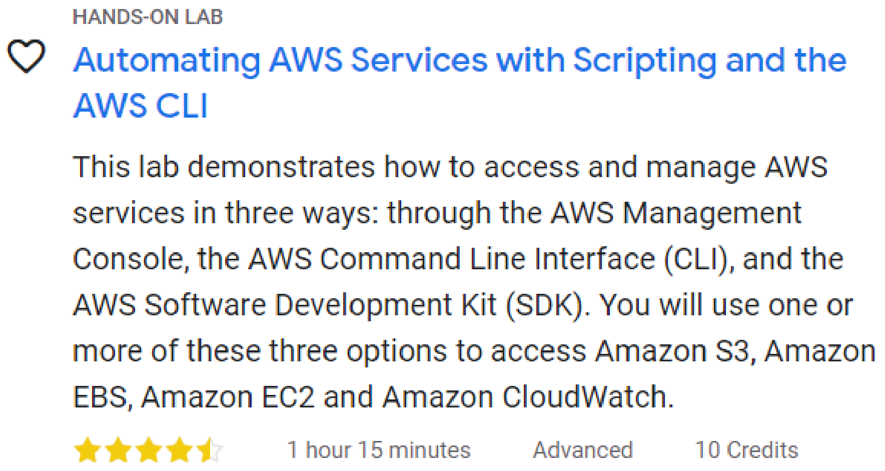
- QWIKSLAB : Automating AWS Services with Scripting and the AWS C
- S3 via CLI



Command	Purpose
Make a bucket	`aws s3 mb s3://my-bucket`
List all buckets	`aws s3 ls`
List the contents of a specific bucket	`aws s3 ls s3://my-bucket`
Upload a file to a bucket	`aws s3 cp file s3://my-bucket/file`
Download a file from a bucket	`aws s3 cp s3://my-bucket/file file`
Copy a file between buckets	`aws s3 cp s3://bucket1/file s3://bucket2/file`
Synchronize a directory with an S3 bucket	`aws s3 sync my-directory s3://my-bucket/`

Scripting : AWS CLI

- QWIKSLAB :Automating AWS Services with Scripting and the AWS CLI
  - Crear un bucket
  - Copiar ficheros al bucket
  - Listar bucket
  - Sincronizar carpeta con S3
  - Listar bucket
  - Borrar bucket



HANDS-ON LAB

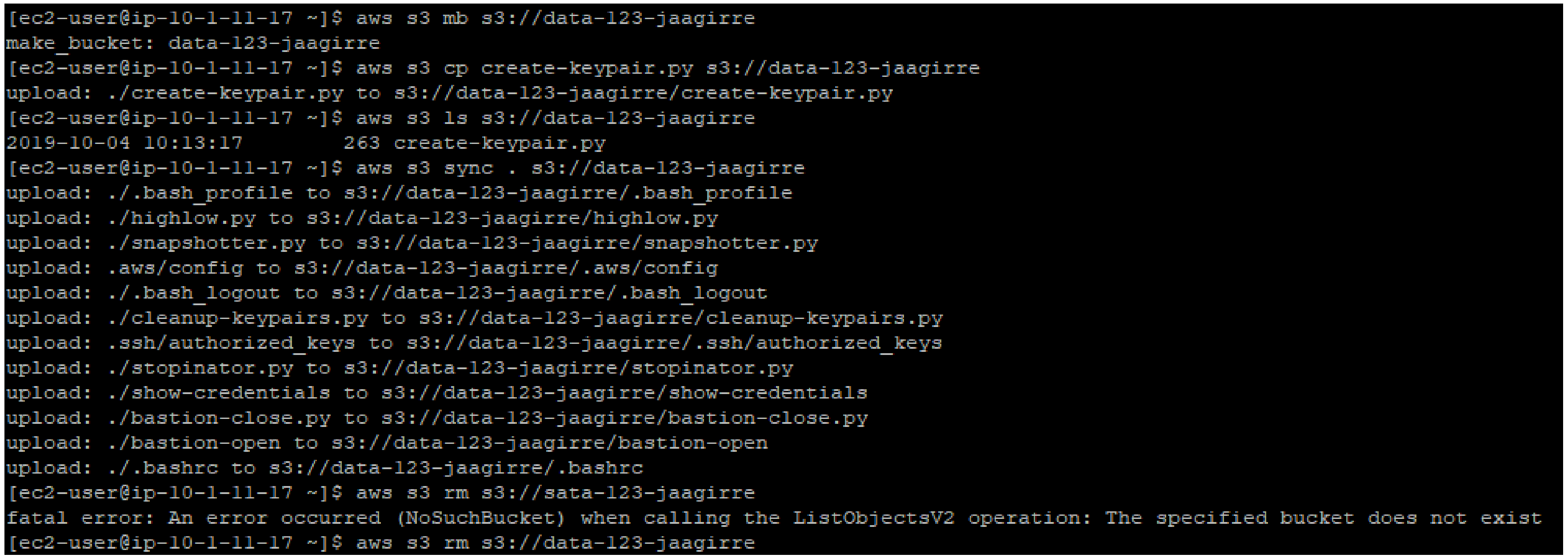
Automating AWS Services with Scripting and the AWS CLI

This lab demonstrates how to access and manage AWS services in three ways: through the AWS Management Console, the AWS Command Line Interface (CLI), and the AWS Software Development Kit (SDK). You will use one or more of these three options to access Amazon S3, Amazon EBS, Amazon EC2 and Amazon CloudWatch.

1 hour 15 minutes Advanced 10 Credits

Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI

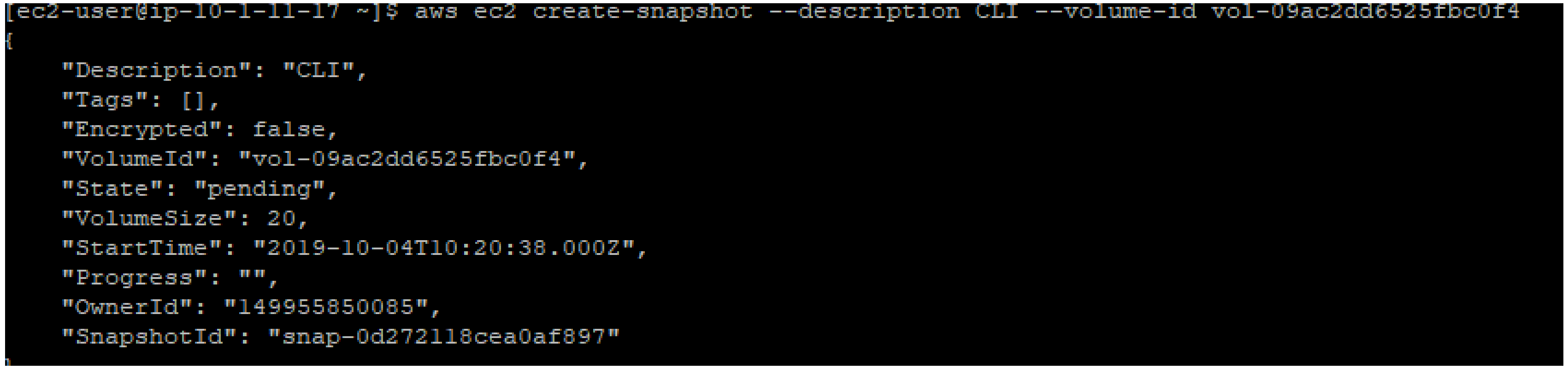


```
[ec2-user@ip-10-1-11-17 ~]$ aws s3 mb s3://data-123-jaagirre
make_bucket: data-123-jaagirre
[ec2-user@ip-10-1-11-17 ~]$ aws s3 cp create-keypair.py s3://data-123-jaagirre
upload: ./create-keypair.py to s3://data-123-jaagirre/create-keypair.py
[ec2-user@ip-10-1-11-17 ~]$ aws s3 ls s3://data-123-jaagirre
2019-10-04 10:13:17        263 create-keypair.py
[ec2-user@ip-10-1-11-17 ~]$ aws s3 sync . s3://data-123-jaagirre
upload: ./.bash_profile to s3://data-123-jaagirre/.bash_profile
upload: ./highlow.py to s3://data-123-jaagirre/highlow.py
upload: ./snapshotter.py to s3://data-123-jaagirre/snapshotter.py
upload: .aws/config to s3://data-123-jaagirre/.aws/config
upload: ./.bash_logout to s3://data-123-jaagirre/.bash_logout
upload: ./cleanup-keypairs.py to s3://data-123-jaagirre/cleanup-keypairs.py
upload: .ssh/authorized_keys to s3://data-123-jaagirre/.ssh/authorized_keys
upload: ./stopinator.py to s3://data-123-jaagirre/stopinator.py
upload: ./show-credentials to s3://data-123-jaagirre/show-credentials
upload: ./bastion-close.py to s3://data-123-jaagirre/bastion-close.py
upload: ./bastion-open to s3://data-123-jaagirre/bastion-open
upload: ./.bashrc to s3://data-123-jaagirre/.bashrc
[ec2-user@ip-10-1-11-17 ~]$ aws s3 rm s3://sata-123-jaagirre
fatal error: An error occurred (NoSuchBucket) when calling the ListObjectsV2 operation: The specified bucket does not exist
[ec2-user@ip-10-1-11-17 ~]$ aws s3 rm s3://data-123-jaagirre
```

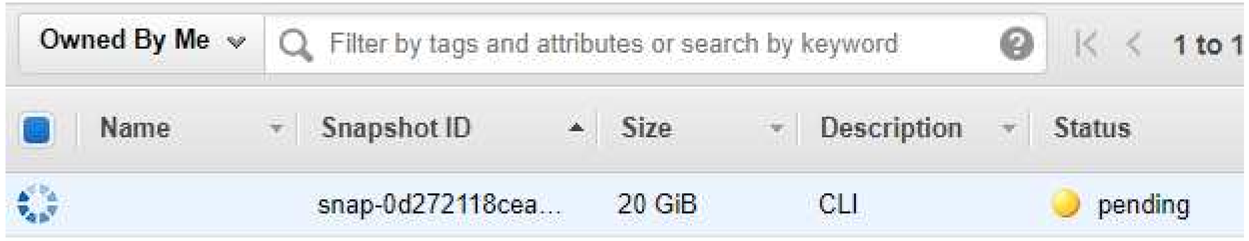
Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Backups de EBS

```
aws ec2 create-snapshot --description CLI --volume-id
YOUR-VOLUME-ID
```



```
[ec2-user@ip-10-1-11-17 ~]$ aws ec2 create-snapshot --description CLI --volume-id vol-09ac2dd6525fbc0f4
{
    "Description": "CLI",
    "Tags": [],
    "Encrypted": false,
    "VolumeId": "vol-09ac2dd6525fbc0f4",
    "State": "pending",
    "VolumeSize": 20,
    "StartTime": "2019-10-04T10:20:38.000Z",
    "Progress": "",
    "OwnerId": "149955850085",
    "SnapshotId": "snap-0d272118cea0af897"
```



Name	Snapshot ID	Size	Description	Status
	snap-0d272118cea...	20 GiB	CLI	pending

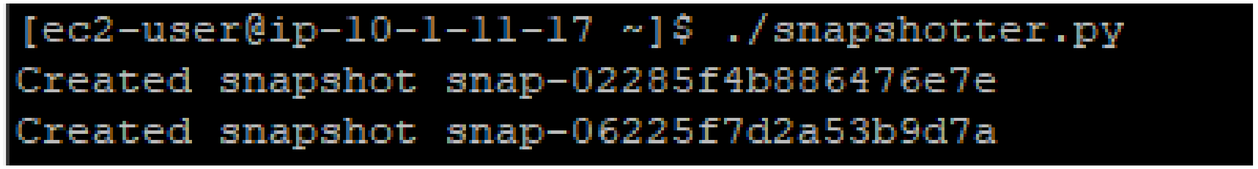
Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Backups de EBS via SDK

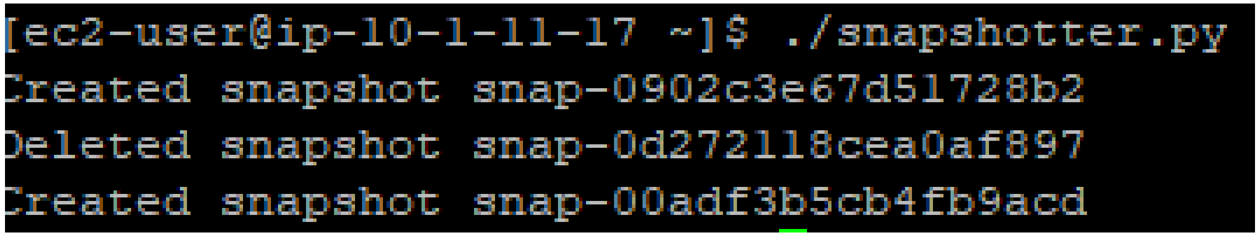
```
import boto3
import datetime

MAX_SNAPSHOTS = 2    # Number of snapshots to keep

# Connect to the Amazon EC2 service
ec2 = boto3.resource('ec2')

# Loop through each volume
for volume in ec2.volumes.all():

  # Create a snapshot of the volume with the current time as a Description
  new_snapshot = volume.create_snapshot(Description = str(datetime.datetime.now()))
  print ("Created snapshot " + new_snapshot.id)

  # Too many snapshots?
  snapshots = list(volume.snapshots.all())
  if len(snapshots) > MAX_SNAPSHOTS:

    # Delete oldest snapshots, but keep MAX_SNAPSHOTS available
    snapshots_sorted = sorted([(s, s.start_time) for s in snapshots], key=lambda k: k[1])
    for snapshot in snapshots_sorted[:-MAX_SNAPSHOTS]:
      print ("Deleted snapshot " + snapshot[0].id)
      snapshot[0].delete()
```

Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Backups de EBS via SDK



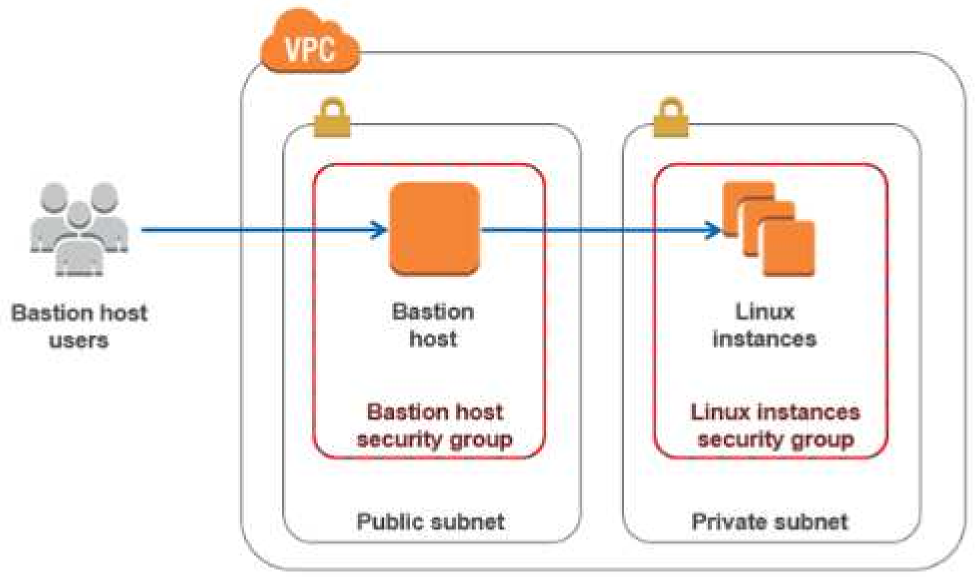
```
[ec2-user@ip-10-1-11-17 ~]$ ./snapshotter.py
Created snapshot snap-02285f4b886476e7e
Created snapshot snap-06225f7d2a53b9d7a
```



```
[ec2-user@ip-10-1-11-17 ~]$ ./snapshotter.py
Created snapshot snap-0902c3e67d51728b2
Deleted snapshot snap-0d272118cea0af897
Created snapshot snap-00adf3b5cb4fb9acd
```

Scripting : AWS CLI

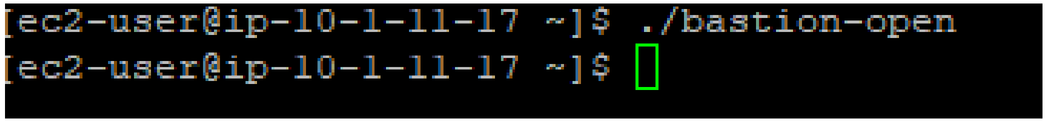
- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Host BASTION



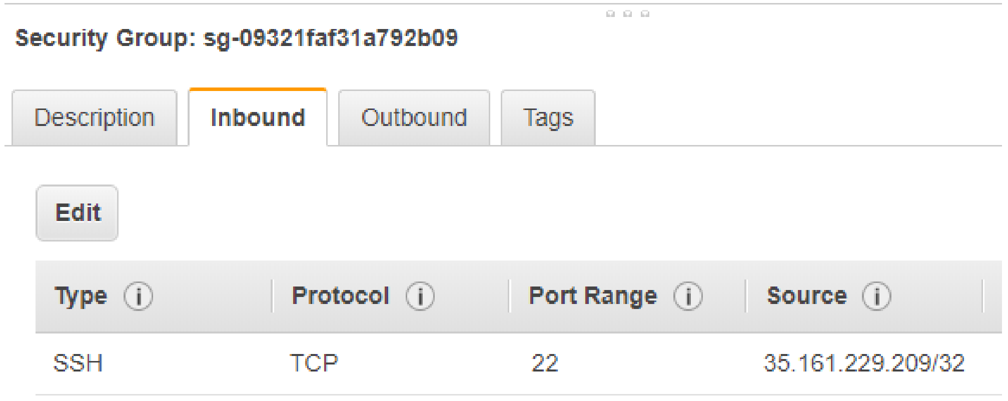
Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Host BASTION
  - Crear desde la consola un grupo de seguridad llamado BASTION en la VPC por defecto
  - Agregar una regla vía CLI

```
IP=`curl -s checkip.amazonaws.com`
aws ec2 authorize-security-group-ingress --group-name
    "Bastion" --protocol tcp --port 22 --cidr $IP/32
```



```
[ec2-user@ip-10-1-11-17 ~]$ ./bastion-open
[ec2-user@ip-10-1-11-17 ~]$ 
```

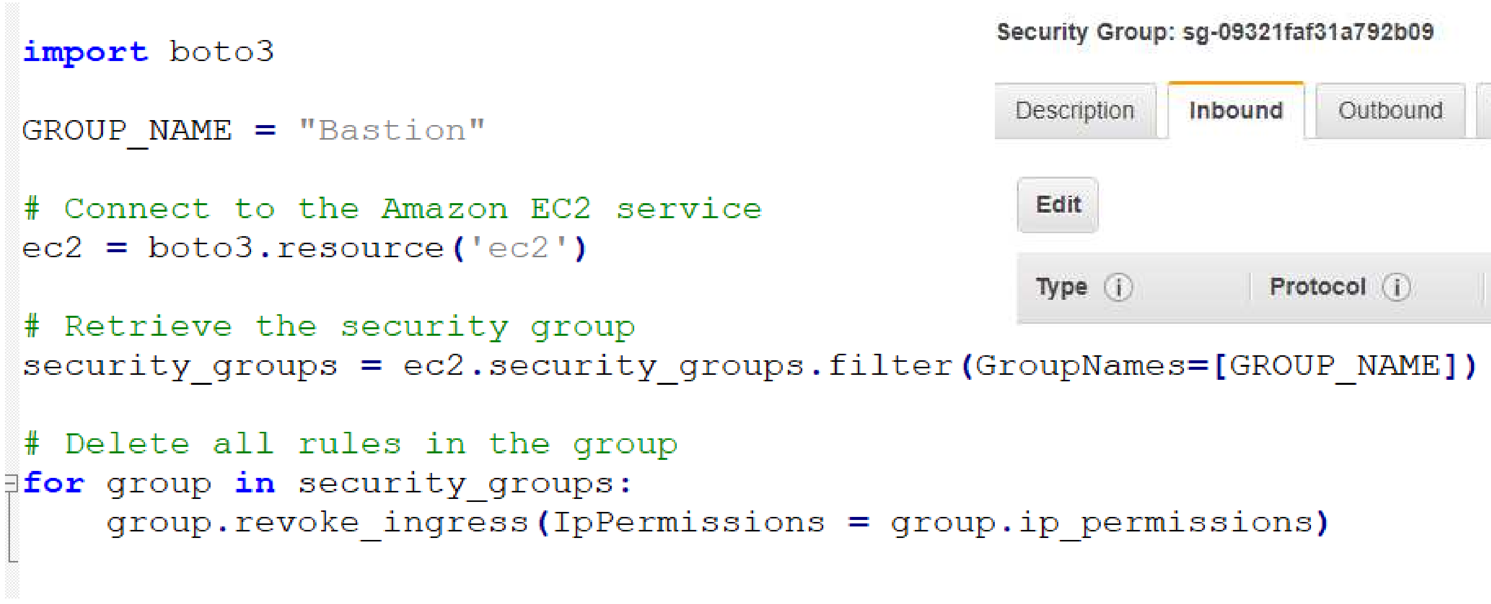


Security Group: sg-09321faf31a792b09

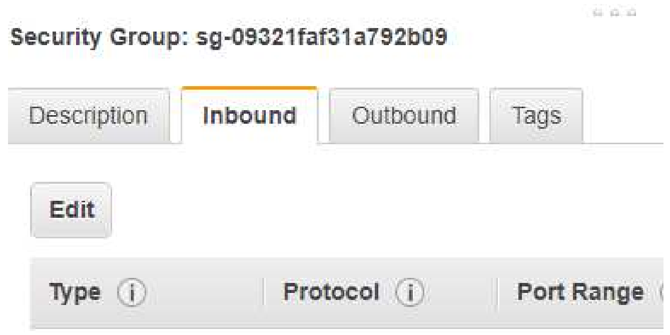
Type	Protocol	Port Range	Source
SSH	TCP	22	35.161.229.209/32

Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Host BASTION
  - Borrar todas las reglas con permisos a nivel de IP

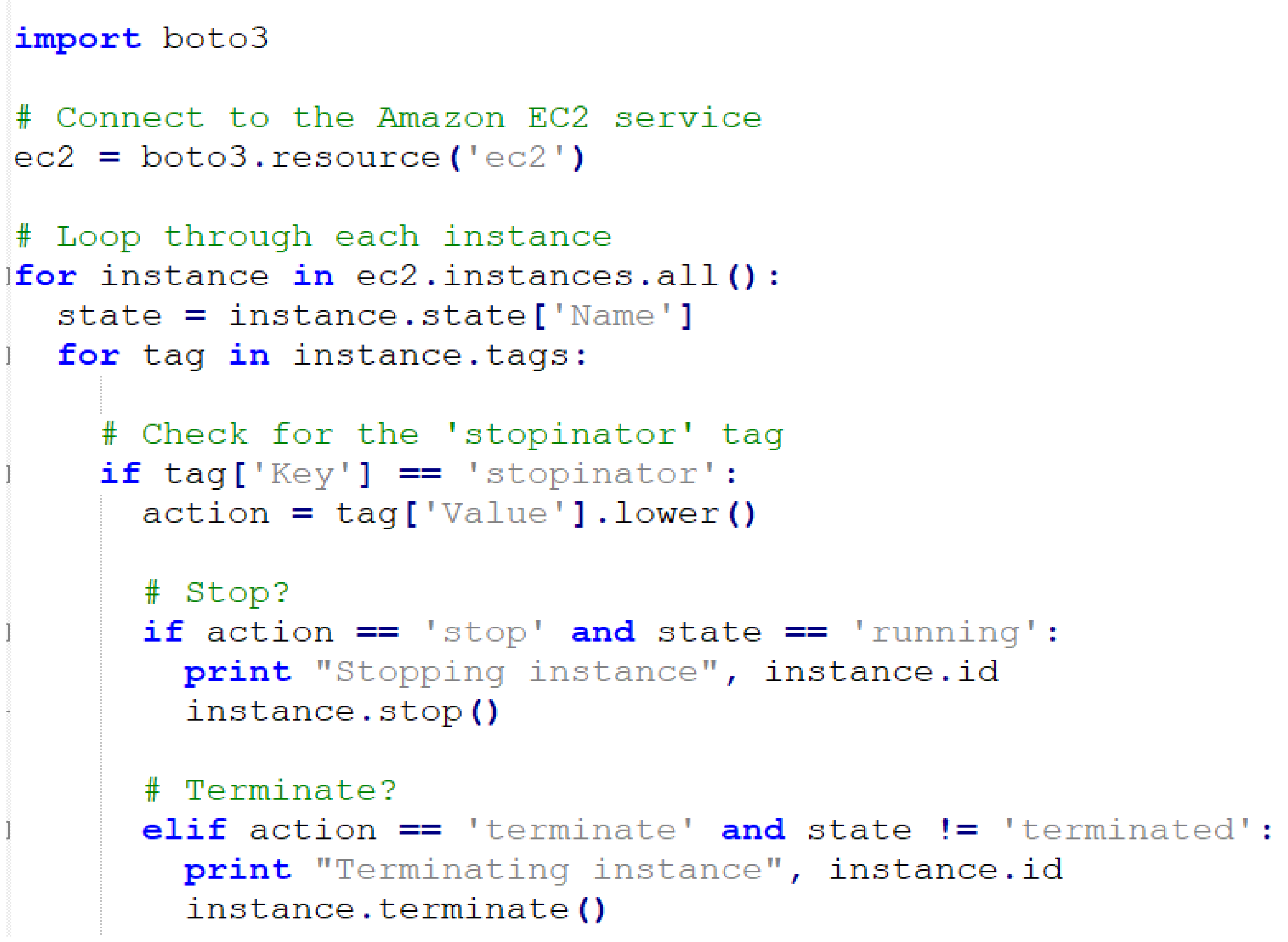


```
import boto3

GROUP_NAME = "Bastion"

# Connect to the Amazon EC2 service
ec2 = boto3.resource('ec2')

# Retrieve the security group
security_groups = ec2.security_groups.filter(GroupNames=[GROUP_NAME])

# Delete all rules in the group
for group in security_groups:
    group.revoke_ingress(IpPermissions = group.ip_permissions)
```



Security Group: sg-09321faf31a792b09

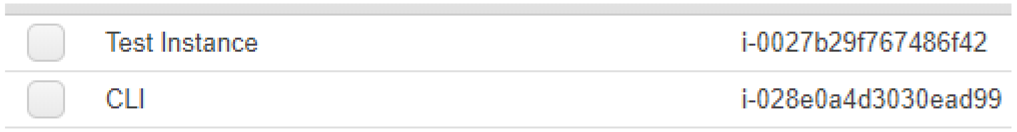
Scripting : AWS CLI

- QWIKSLAB : Automating AWS Services with Scripting and the AWS CLI
- Ejercicio Scripting : AWS SDK Boto3 : Stopinator
- 

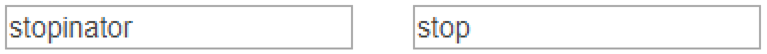
```python
import boto3

# Connect to the Amazon EC2 service
ec2 = boto3.resource('ec2')

# Loop through each instance
for instance in ec2.instances.all():
  state = instance.state['Name']
  for tag in instance.tags:

    # Check for the 'stopinator' tag
    if tag['Key'] == 'stopinator':
      action = tag['Value'].lower()

      # Stop?
      if action == 'stop' and state == 'running':
        print "Stopping instance", instance.id
        instance.stop()

      # Terminate?
      elif action == 'terminate' and state != 'terminated':
        print "Terminating instance", instance.id
        instance.terminate()
```

Scripting : AWS CLI

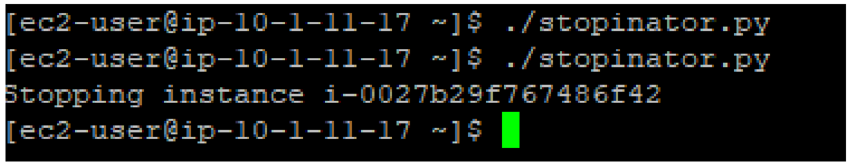
- QWIKSLAB : AWS SDK Boto3 : Stopinator
  - Script para apagar todas las instancias EC2
  - Crear Tag



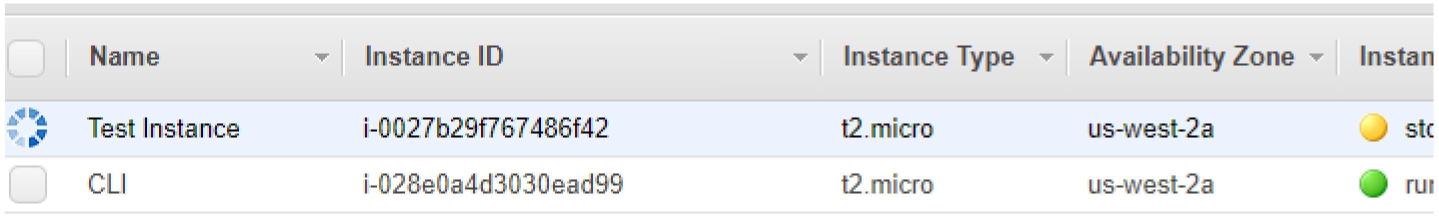
☐	Test Instance	i-0027b29f767486f42
☐	CLI	i-028e0a4d3030ead99



stopinator	stop



```
[ec2-user@ip-10-1-11-17 ~]$ ./stopinator.py
[ec2-user@ip-10-1-11-17 ~]$ ./stopinator.py
Stopping instance i-0027b29f767486f42
[ec2-user@ip-10-1-11-17 ~]$ 
```

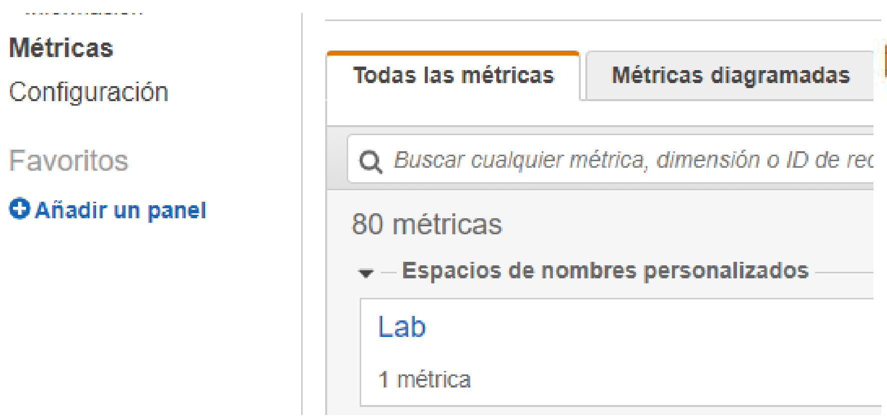


☐	Name	Instance ID	Instance Type	Availability Zone	Instan
	Test Instance	i-0027b29f767486f42	t2.micro	us-west-2a	● stc
☐	CLI	i-028e0a4d3030ead99	t2.micro	us-west-2a	● rur

Scripting : AWS CLI

- QWIKSLAB : AWS CLI
- Script para métricas Cloudwath

```
aws cloudwatch put-metric-data --namespace Lab --metric-
name YOUR-INITIALS --value 42
```



Métricas
Configuración
Favoritos
Añadir un panel

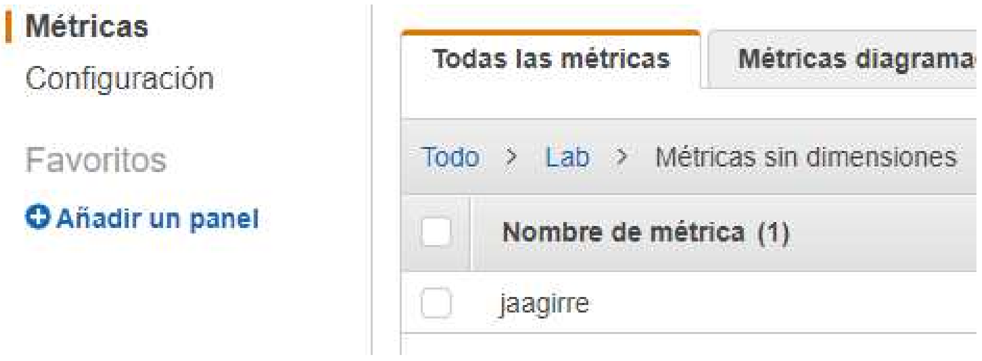
Todas las métricas | Métricas diagramadas

Buscar cualquier métrica, dimensión o ID de rec

80 métricas

Espacios de nombres personalizados

Lab
1 métrica



Métricas
Configuración
Favoritos
Añadir un panel

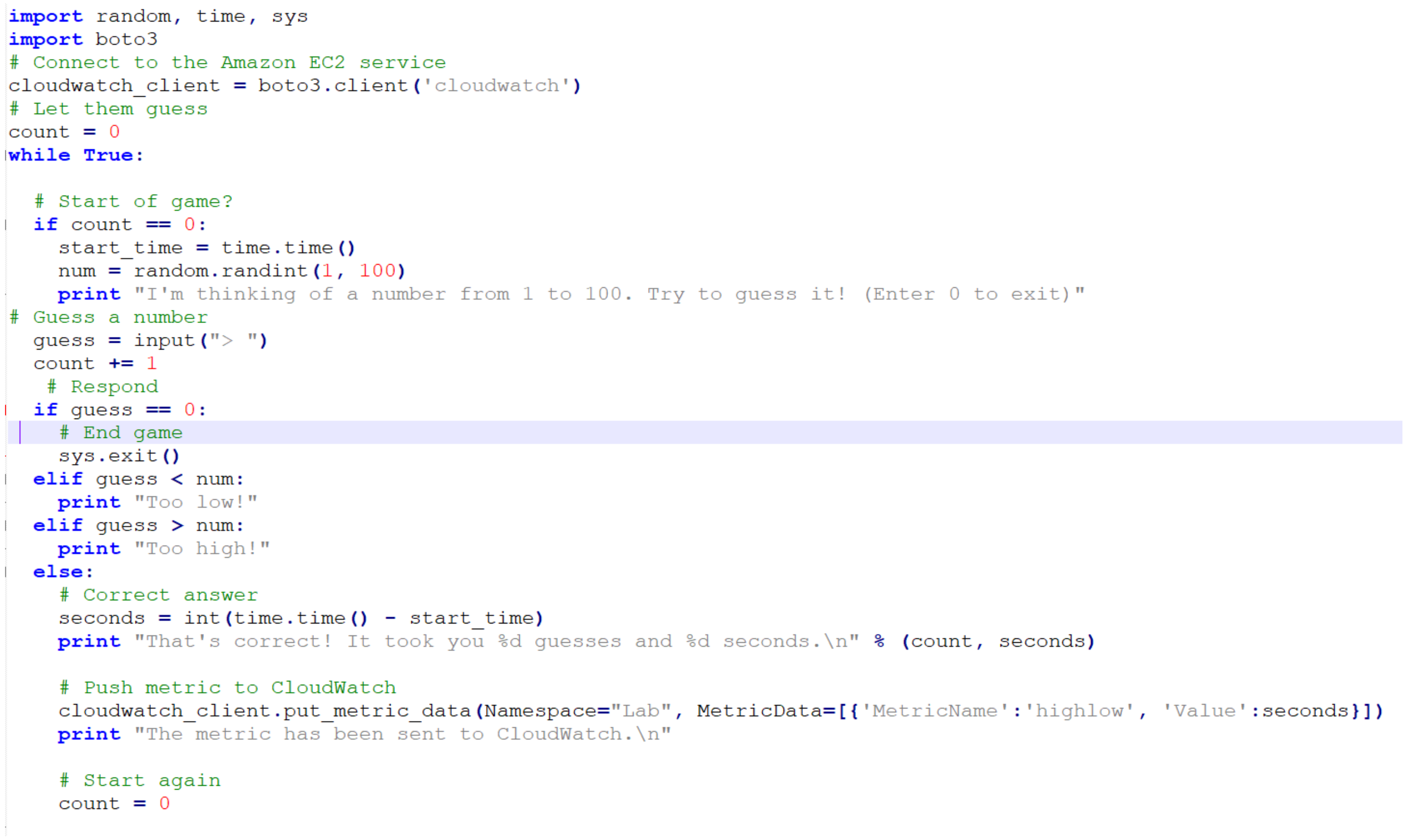
Todas las métricas | Métricas diagrama

Todo > Lab > Métricas sin dimensiones

Nombre de métrica (1)
jaagirre

Scripting : AWS CLI

- QWIKSLAB : AWS CLI
- Script para métricas Cloudwath : Script para crear métricas

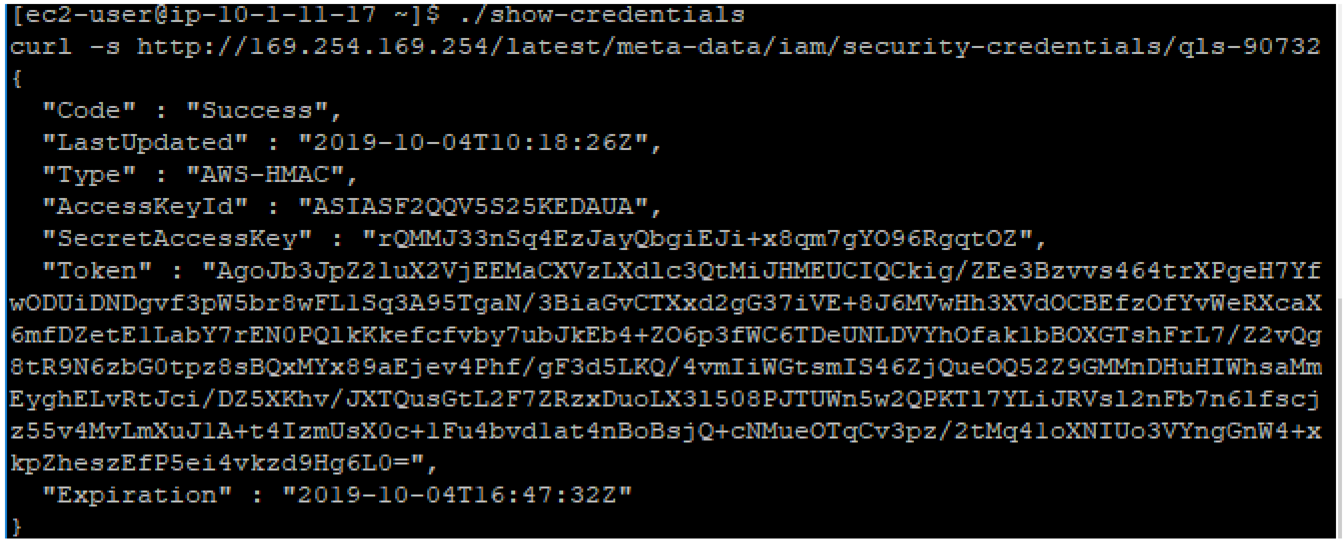


```
import random, time, sys
import boto3
# Connect to the Amazon EC2 service
cloudwatch_client = boto3.client('cloudwatch')
# Let them guess
count = 0
while True:

  # Start of game?
  if count == 0:
    start_time = time.time()
    num = random.randint(1, 100)
    print "I'm thinking of a number from 1 to 100. Try to guess it! (Enter 0 to exit)"
# Guess a number
  guess = input("> ")
  count += 1
   # Respond
  if guess == 0:
    # End game
    sys.exit()
  elif guess < num:
    print "Too low!"
  elif guess > num:
    print "Too high!"
  else:
    # Correct answer
    seconds = int(time.time() - start_time)
    print "That's correct! It took you %d guesses and %d seconds.\n" % (count, seconds)

    # Push metric to CloudWatch
    cloudwatch_client.put_metric_data(Namespace="Lab", MetricData=[{'MetricName':'highlow', 'Value':seconds}])
    print "The metric has been sent to CloudWatch.\n"

    # Start again
    count = 0
```

Scripting : AWS CLI

- QWIKSLAB : AWS CLI
- Script para credenciales de seguridad

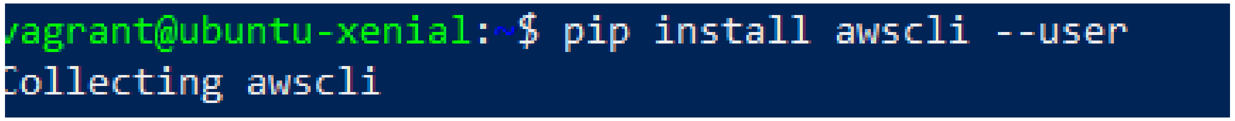
```
ROLE=`curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/`
echo curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/$ROLE
curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/$ROLE
echo
```



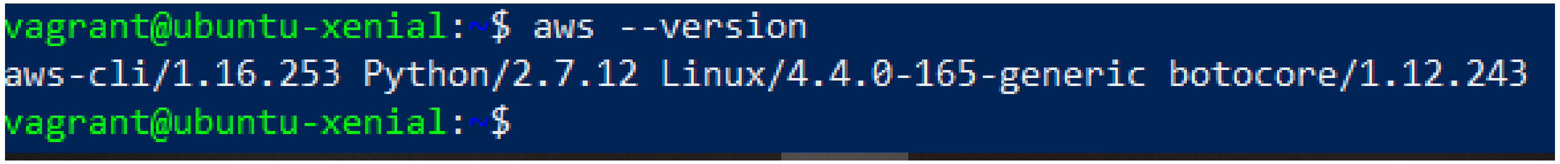
```
[ec2-user@ip-10-1-11-17 ~]$ ./show-credentials
curl -s http://169.254.169.254/latest/meta-data/iam/security-credentials/qls-90732
{
  "Code" : "Success",
  "LastUpdated" : "2019-10-04T10:18:26Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIASF2QQV5S25KEDAUA",
  "SecretAccessKey" : "rQMMJ33nSq4EzJayQbgiEJi+x8qm7gYO96RgqtOZ",
  "Token" : "AgoJb3JpZ2luX2VjEEMaCXVzLXdlc3QtMiJHMEUCIQCkig/ZEe3Bzvvs464trXPgeH7Yf
wODUiDNDgvf3pW5br8wFL1Sq3A95TgaN/3BiaGvCTXxd2gG37iVE+8J6MVwHh3XVdOCBEfzOfYvWeRXcaX
6mfDZetElLabY7rEN0PQlkKkefcfvby7ubJkEb4+ZO6p3fWC6TDeUNLDVYhOfaklbBOXGTshFrL7/Z2vQg
8tR9N6zbG0tpz8sBQxMYx89aEjev4Phf/gF3d5LKQ/4vmIiWGtsmIS46ZjQueOQ52Z9GMMnDHuHIWhsaMm
EyghELvRtJci/DZ5XKhv/JXTQusGtL2F7ZRzxDuoLX31508PJTUWn5w2QPKTl7YLiJRVsl2nFb7n6lfscj
z55v4MvLmXuJlA+t4IzmUsX0c+lFu4bvdlat4nBoBsjQ+cNMueOTqCv3pz/2tMq4loXNIUo3VYngGnW4+x
kpZheszEfP5ei4vkzd9Hg6L0=",
  "Expiration" : "2019-10-04T16:47:32Z"
}
```

Scripting : AWS CLI

- Ejercicio Scripting : Ejemplo Finalizar todas las instancias : Bash & AWS CLI & Jq & Xargs
- Instalar

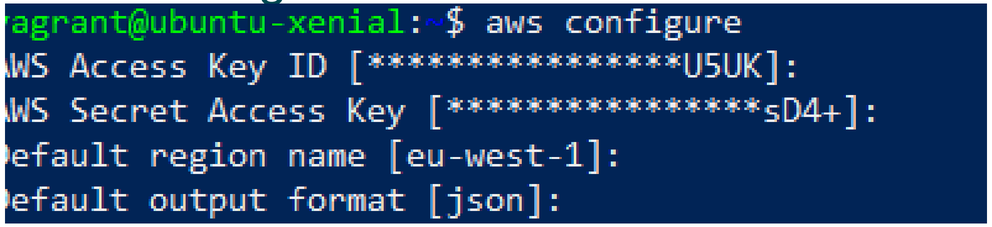


```
vagrant@ubuntu-xenial:~$ pip install awscli --user
Collecting awscli
```



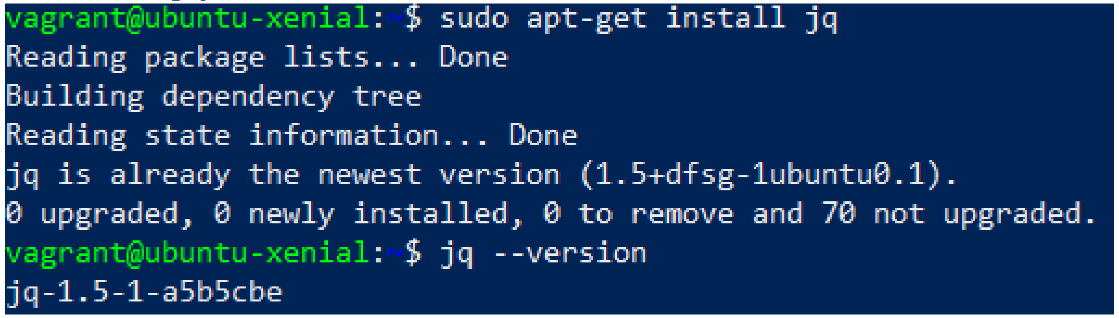
```
vagrant@ubuntu-xenial:~$ aws --version
aws-cli/1.16.253 Python/2.7.12 Linux/4.4.0-165-generic botocore/1.12.243
vagrant@ubuntu-xenial:~$
```

- Configurar credenciales



```
agrant@ubuntu-xenial:~$ aws configure
WS Access Key ID [****************U5UK]:
WS Secret Access Key [****************sD4+]:
efault region name [eu-west-1]:
efault output format [json]:
```

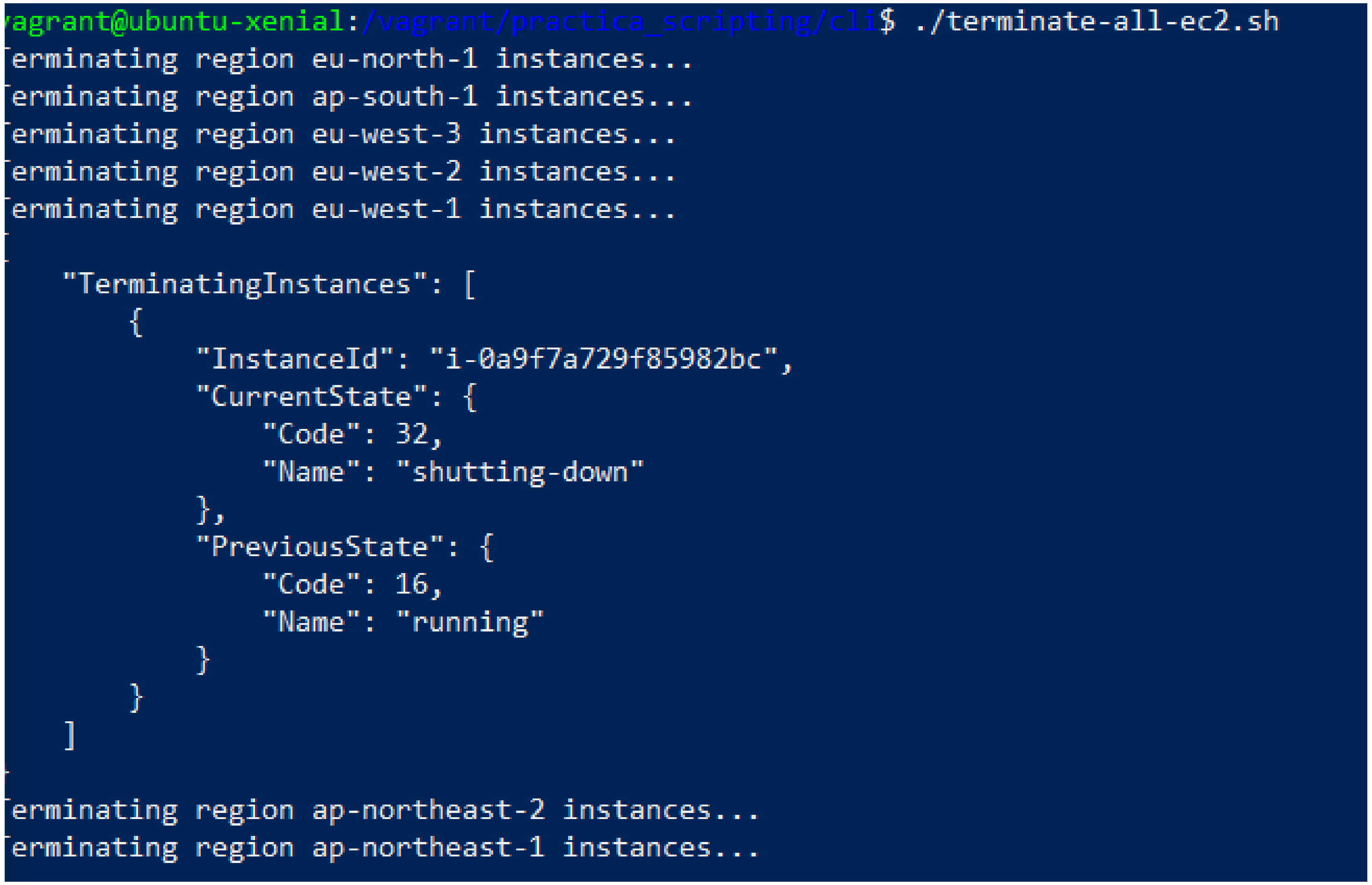
- **Jq** para lectura de formato JSON



```
vagrant@ubuntu-xenial:~$ sudo apt-get install jq
Reading package lists... Done
Building dependency tree
Reading state information... Done
jq is already the newest version (1.5+dfsg-1ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 70 not upgraded.
vagrant@ubuntu-xenial:~$ jq --version
jq-1.5-1-a5b5cbe
```

Scripting : AWS CLI

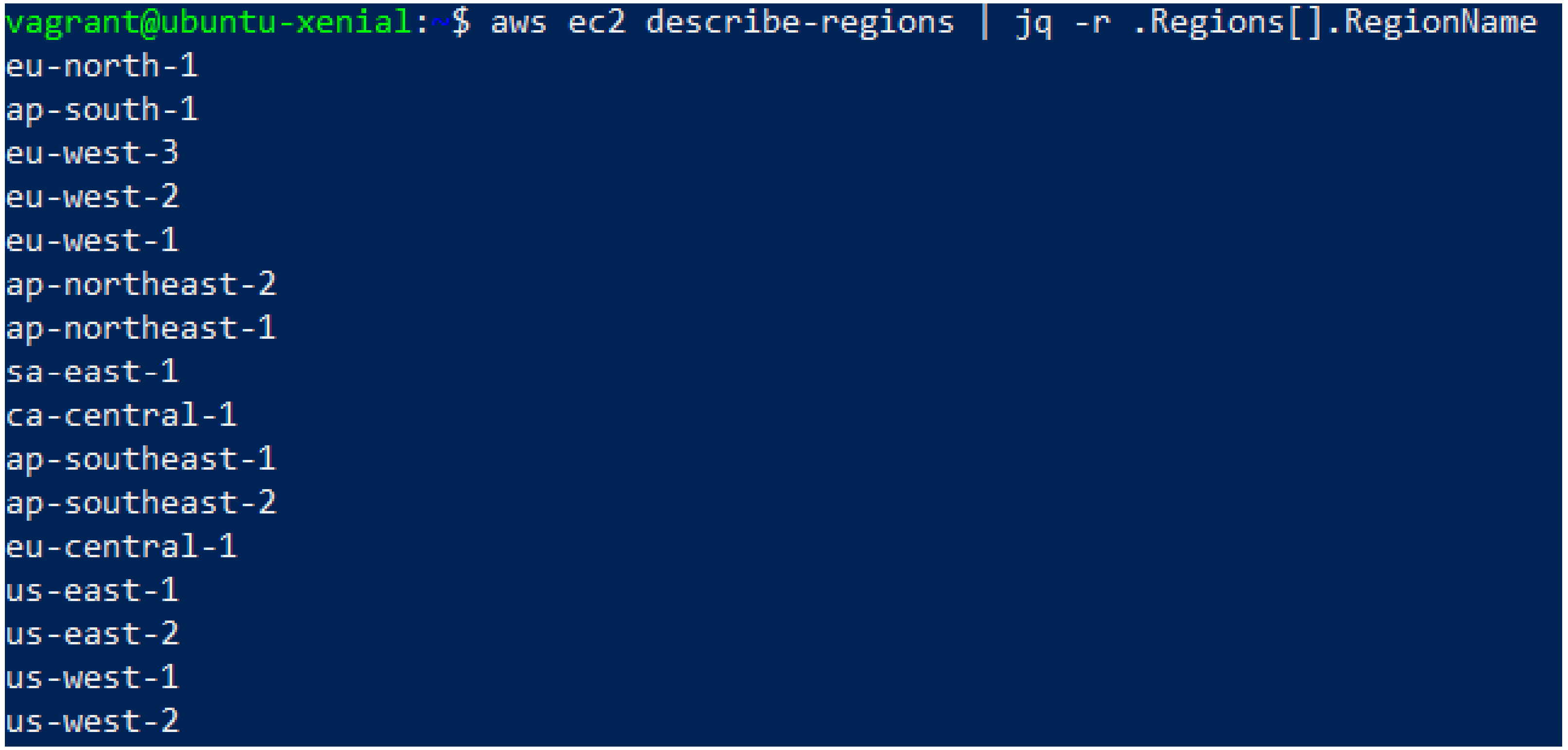
- Ejemplo Finalizar todas las instancias : Bash & AWS CLI & Jq & Xargs



```
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./terminate-all-ec2.sh
erminating region eu-north-1 instances...
erminating region ap-south-1 instances...
erminating region eu-west-3 instances...
erminating region eu-west-2 instances...
erminating region eu-west-1 instances...

    "TerminatingInstances": [
        {
            "InstanceId": "i-0a9f7a729f85982bc",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]

erminating region ap-northeast-2 instances...
erminating region ap-northeast-1 instances...
```

Scripting : AWS CLI

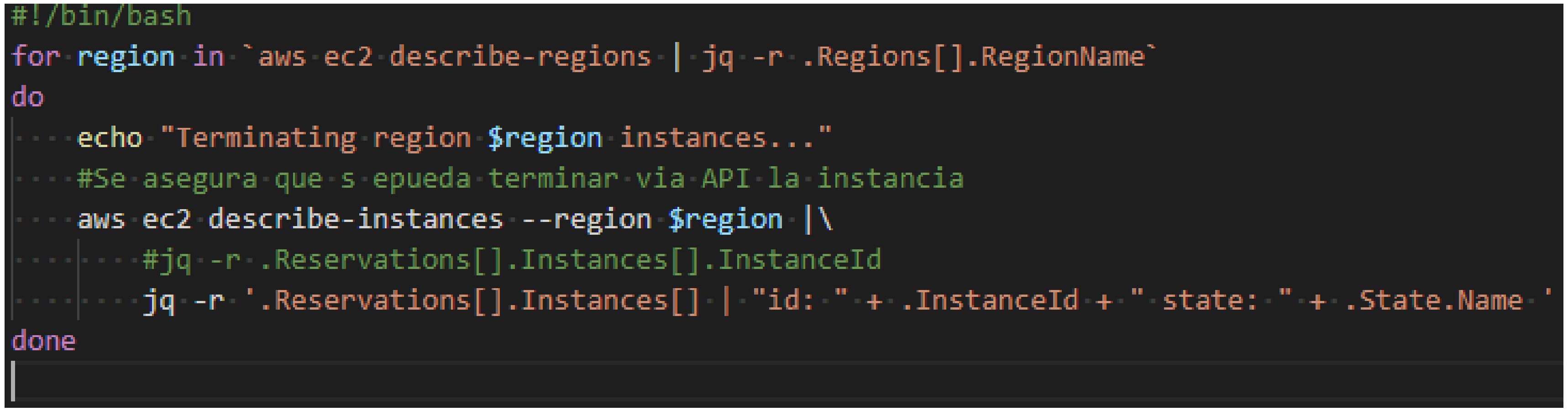
- Scripting : AWS SDK
- Probar entorno AWS CLI & Jq



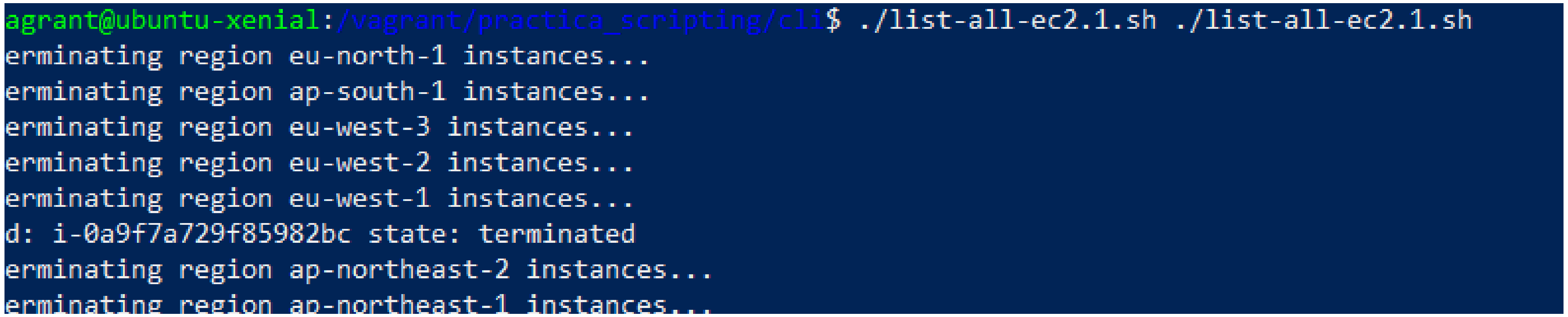
```
vagrant@ubuntu-xenial:~$ aws ec2 describe-regions | jq -r .Regions[].RegionName
eu-north-1
ap-south-1
eu-west-3
eu-west-2
eu-west-1
ap-northeast-2
ap-northeast-1
sa-east-1
ca-central-1
ap-southeast-1
ap-southeast-2
eu-central-1
us-east-1
us-east-2
us-west-1
us-west-2
```

Scripting : AWS CLI

- Ejemplo Finalizar todas las instancias : Bash & AWS CLI & Jq
- Listar las instancias existentes



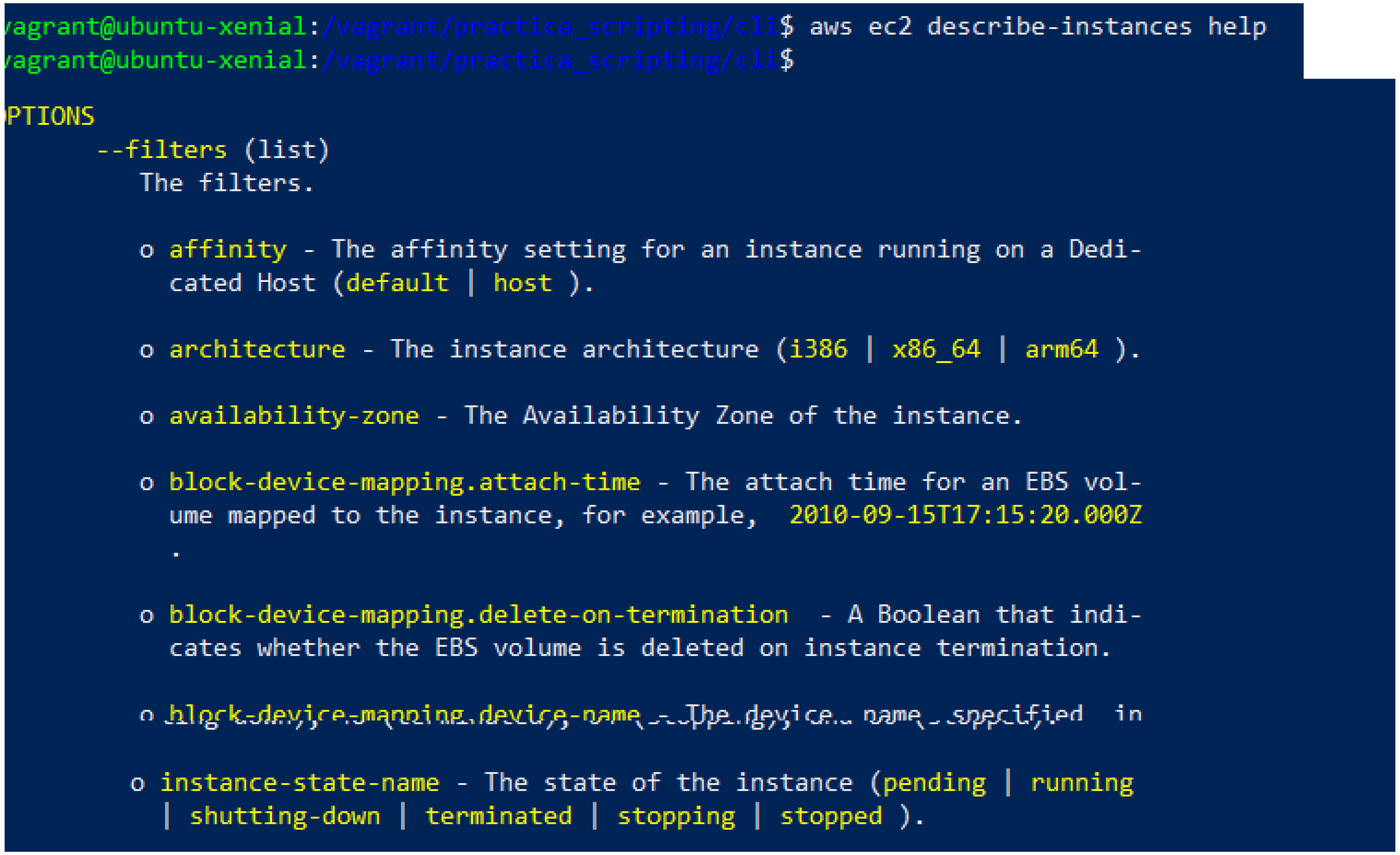
```
#!/bin/bash
for region in `aws ec2 describe-regions | jq -r .Regions[].RegionName`
do
    echo "Terminating region $region instances..."
    #Se asegura que s epueda terminar via API la instancia
    aws ec2 describe-instances --region $region |\
        #jq -r .Reservations[].Instances[].InstanceId
        jq -r '.Reservations[].Instances[] | "id: " + .InstanceId + " state: " + .State.Name '
done
```



```
agrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./list-all-ec2.1.sh ./list-all-ec2.1.sh
erminating region eu-north-1 instances...
erminating region ap-south-1 instances...
erminating region eu-west-3 instances...
erminating region eu-west-2 instances...
erminating region eu-west-1 instances...
d: i-0a9f7a729f85982bc state: terminated
erminating region ap-northeast-2 instances...
erminating region ap-northeast-1 instances...
```

Scripting : AWS CLI

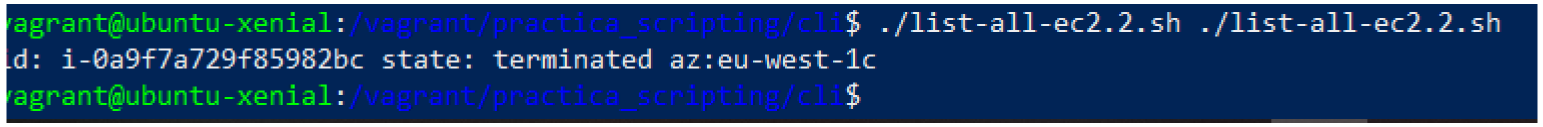
- Ejemplo Finalizar todas las instancias : Bash & AWS CLI & Jq
- Filtrar en los comandos AWS : Ejemplo



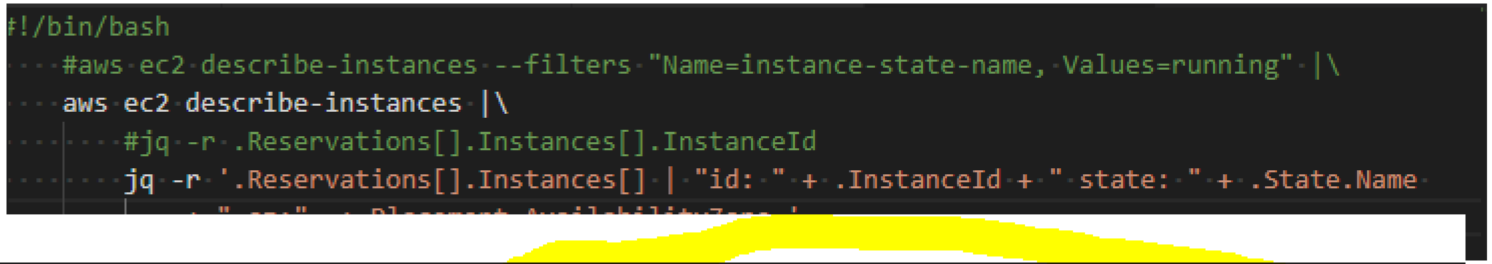
```
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ aws ec2 describe-instances help
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$

PTIONS
      --filters (list)
         The filters.

           o affinity - The affinity setting for an instance running on a Dedi-
             cated Host (default | host ).

           o architecture - The instance architecture (i386 | x86_64 | arm64 ).

           o availability-zone - The Availability Zone of the instance.

           o block-device-mapping.attach-time - The attach time for an EBS vol-
             ume mapped to the instance, for example,  2010-09-15T17:15:20.000Z
             .

           o block-device-mapping.delete-on-termination  - A Boolean that indi-
             cates whether the EBS volume is deleted on instance termination.

           o block-device-mapping.device-name - The device name specified  in

          o instance-state-name - The state of the instance (pending | running
            | shutting-down | terminated | stopping | stopped ).
```

Scripting : AWS CLI

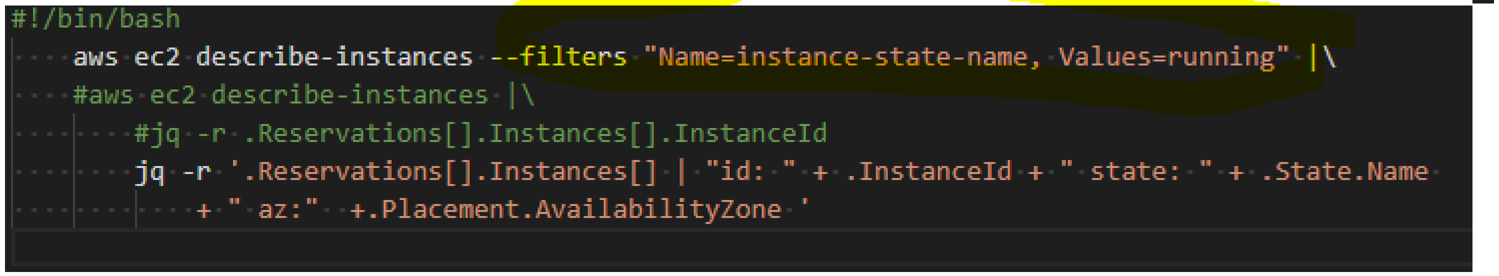
- Ejemplo Finalizar todas las instancias : Bash & AWS CLI & Jq
- Filtrar en los comandos AWS : Ejemplo



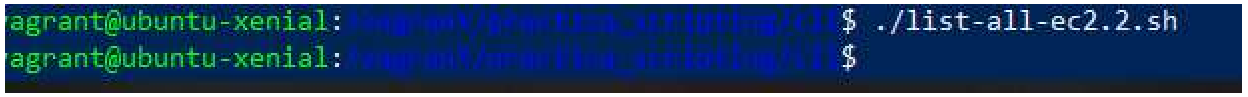
```
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./list-all-ec2.2.sh ./list-all-ec2.2.sh
id: i-0a9f7a729f85982bc state: terminated az:eu-west-1c
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$
```



```
#!/bin/bash
    #aws ec2 describe-instances --filters "Name=instance-state-name, Values=running" |\
    aws ec2 describe-instances |\
        #jq -r .Reservations[].Instances[].InstanceId
        jq -r '.Reservations[].Instances[] | "id: " + .InstanceId + " state: " + .State.Name
            + " az:" + .Placement.AvailabilityZone '
```



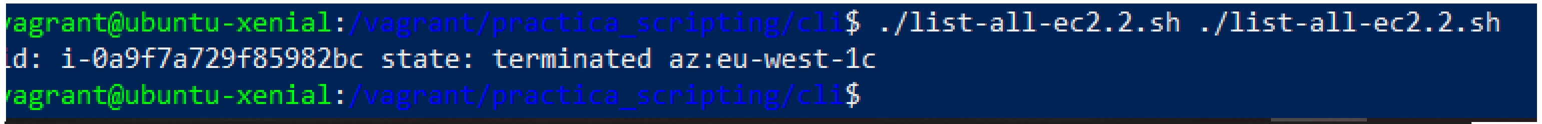
```
#!/bin/bash
    aws ec2 describe-instances --filters "Name=instance-state-name, Values=running" |\
    #aws ec2 describe-instances |\
        #jq -r .Reservations[].Instances[].InstanceId
        jq -r '.Reservations[].Instances[] | "id: " + .InstanceId + " state: " + .State.Name
            + " az:" +.Placement.AvailabilityZone '
```



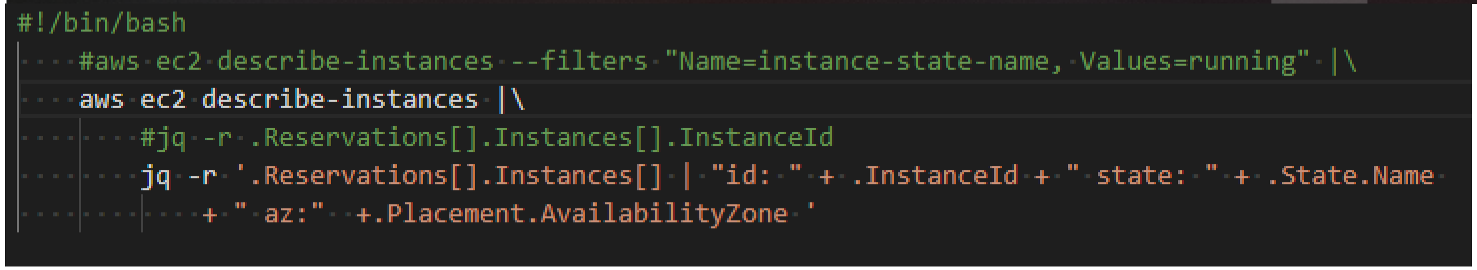
```
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./list-all-ec2.2.sh
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$
```

Scripting : AWS CLI

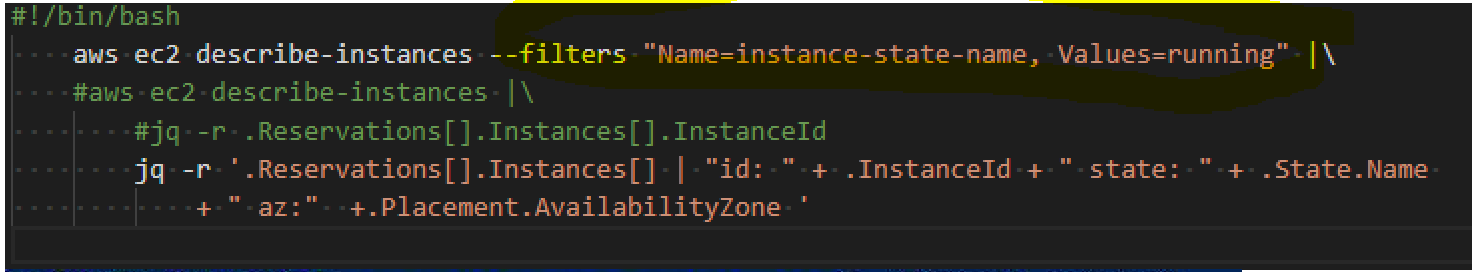
- Ejemplo Finalizar todas las instancias : Bash & AWS CLI & Jq
- Filtrar en los comandos AWS : Ejemplo



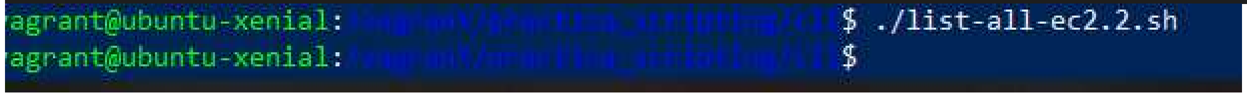
```
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./list-all-ec2.2.sh ./list-all-ec2.2.sh
id: i-0a9f7a729f85982bc state: terminated az:eu-west-1c
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$
```



```
#!/bin/bash
    #aws ec2 describe-instances --filters "Name=instance-state-name, Values=running" |\
    aws ec2 describe-instances |\
        #jq -r .Reservations[].Instances[].InstanceId
        jq -r '.Reservations[].Instances[] | "id: " + .InstanceId + " state: " + .State.Name
            + " az:"  +.Placement.AvailabilityZone '
```



```
#!/bin/bash
    aws ec2 describe-instances --filters "Name=instance-state-name, Values=running" |\
    #aws ec2 describe-instances |\
        #jq -r .Reservations[].Instances[].InstanceId
        jq -r '.Reservations[].Instances[] | "id: " + .InstanceId + " state: " + .State.Name
            + " az:"  +.Placement.AvailabilityZone '
```



```
agrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./list-all-ec2.2.sh
agrant@ubuntu-xenial:/vagrant/practica_scripting/cli$
```

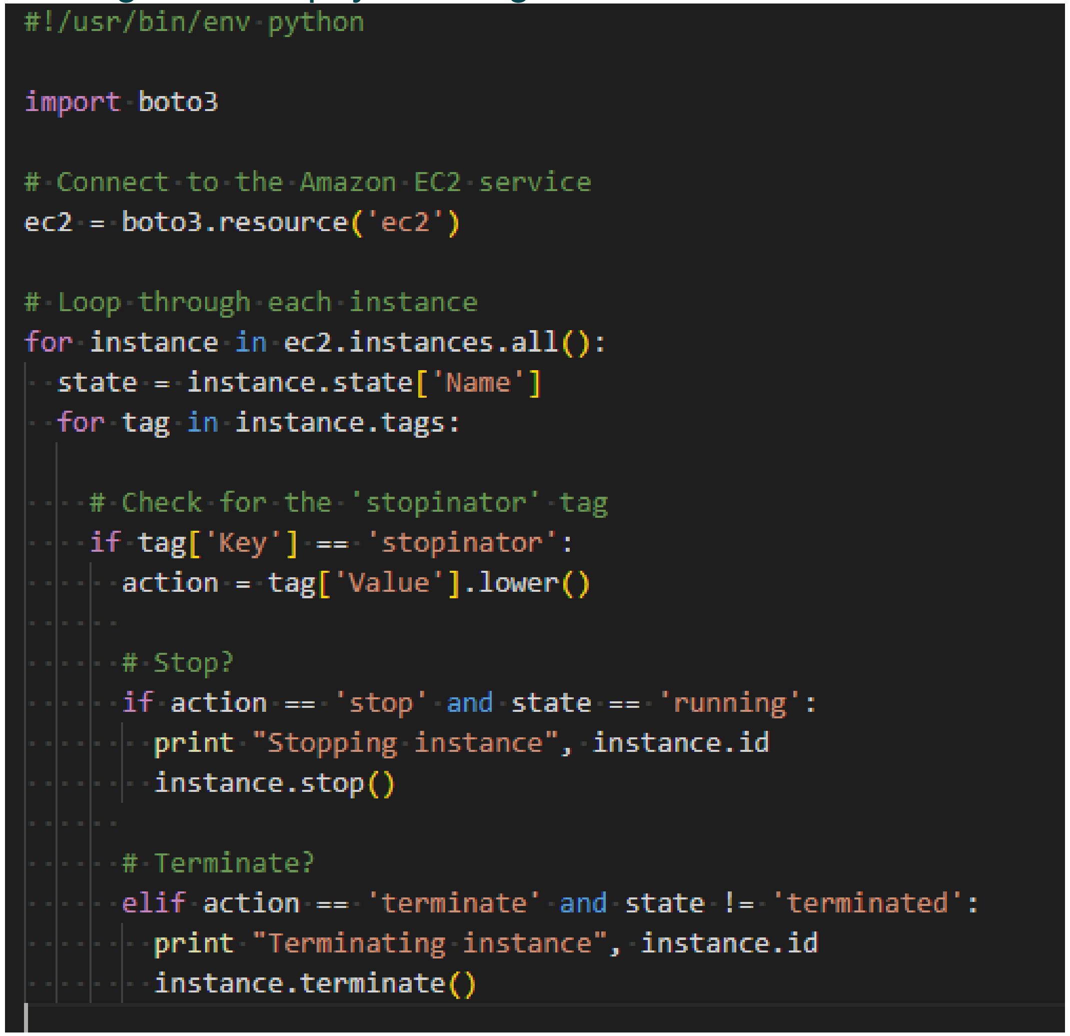
Scripting : AWS CLI

- Ejemplo Finalizar todas las instancias : Bash & AWS CLI & Jq
- Filtrar en los comandos AWS : Ejemplo

```
#!/bin/bash
for region in `aws ec2 describe-regions | jq -r .Regions[].RegionName`
do
  echo "Terminating region $region instances..."
  #Se asegura que s epueda terminar via API la instancia
  aws ec2 describe-instances --region $region | \
    jq -r .Reservations[].Instances[].InstanceId |\
      xargs -L 1 -I {} aws ec2 modify-instance-attribute \
      --region $region \
      --no-disable-api-termination \
      --instance-id {}
  #y se termina la instancia
  aws ec2 describe-instances --region $region |\
    jq -r .Reservations[].Instances[].InstanceId |\
      xargs -L 1 -I {} aws ec2 terminate-instances \
      --region $region \
      --instance-id {}
done
```

Scripting : AWS CLI

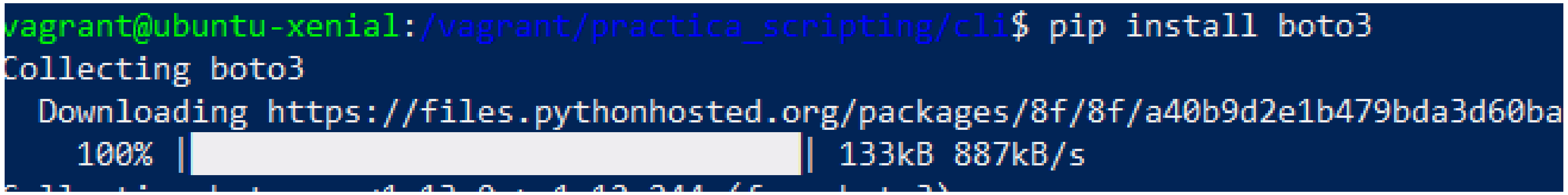
- Ejemplo Finalizar todas las instancias : AWS SDK Python Boto3
- Eliminar aquellas instancias que tengan el TAG stopinator
  - Si el tag vale ‘stop’ y si el tag=‘terminate’



```python
#!/usr/bin/env python

import boto3

# Connect to the Amazon EC2 service
ec2 = boto3.resource('ec2')

# Loop through each instance
for instance in ec2.instances.all():
  state = instance.state['Name']
  for tag in instance.tags:

    # Check for the 'stopinator' tag
    if tag['Key'] == 'stopinator':
      action = tag['Value'].lower()

      # Stop?
      if action == 'stop' and state == 'running':
        print "Stopping instance", instance.id
        instance.stop()

      # Terminate?
      elif action == 'terminate' and state != 'terminated':
        print "Terminating instance", instance.id
        instance.terminate()
```

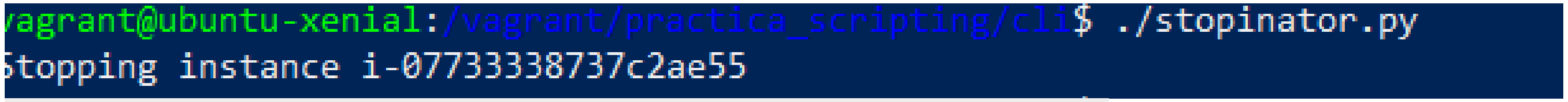
Scripting : AWS CLI

- Ejemplo Finalizar todas las instancias : AWS SDK Python Boto3
- Eliminar aquellas instancias que tengan el TAG stopinator
  - Instalar librería boto3

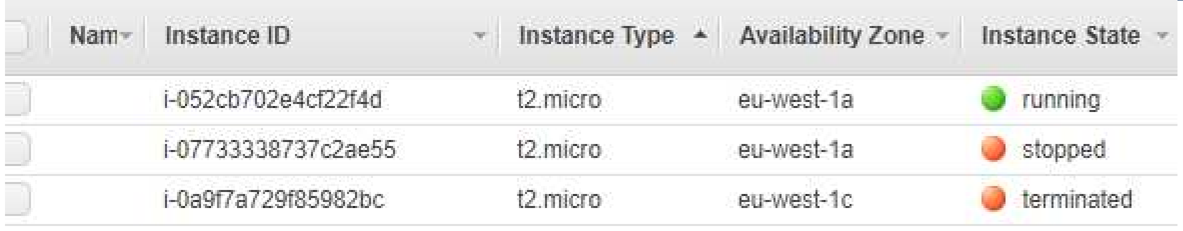


```
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ pip install boto3
Collecting boto3
  Downloading https://files.pythonhosted.org/packages/8f/8f/a40b9d2e1b479bda3d60ba
    100% |████████████████████████████████| 133kB 887kB/s
```

  - Tres instancias
    - eu-west-1 : tag stopinator=stop
    - eu-west-1 : tag cost=master
    - uvags-east-1 : tag stopinator=terminate



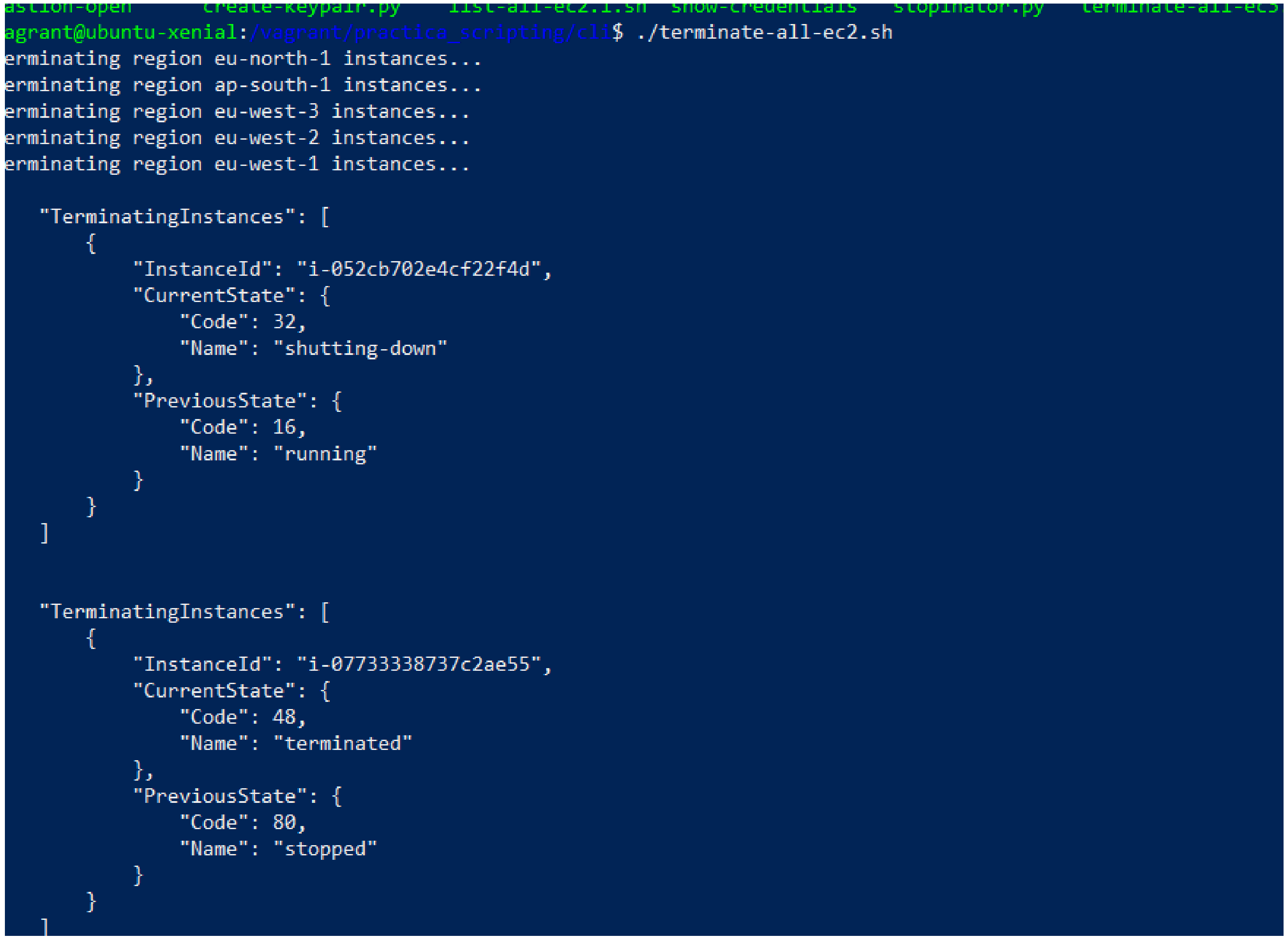
```
vagrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./stopinator.py
Stopping instance i-07733338737c2ae55
```



Name	Instance ID	Instance Type	Availability Zone	Instance State
	i-052cb702e4cf22f4d	t2.micro	eu-west-1a	🟢 running
	i-07733338737c2ae55	t2.micro	eu-west-1a	🔴 stopped
	i-0a9f7a729f85982bc	t2.micro	eu-west-1c	🔴 terminated

Scripting : AWS CLI

- Ejemplo Finalizar todas las instancias : AWS SDK Python Boto3
- Ahora ejecutamos con el shellscript



```
astion-open      create-keypair.py    list-all-ec2.1.sh  show-credentials   stopinator.py    terminate-all-ec3
agrant@ubuntu-xenial:/vagrant/practica_scripting/cli$ ./terminate-all-ec2.sh
erminating region eu-north-1 instances...
erminating region ap-south-1 instances...
erminating region eu-west-3 instances...
erminating region eu-west-2 instances...
erminating region eu-west-1 instances...

   "TerminatingInstances": [
       {
           "InstanceId": "i-052cb702e4cf22f4d",
           "CurrentState": {
               "Code": 32,
               "Name": "shutting-down"
           },
           "PreviousState": {
               "Code": 16,
               "Name": "running"
           }
       }
   ]


   "TerminatingInstances": [
       {
           "InstanceId": "i-07733338737c2ae55",
           "CurrentState": {
               "Code": 48,
               "Name": "terminated"
           },
           "PreviousState": {
               "Code": 80,
               "Name": "stopped"
           }
       }
   ]
```

Scripting : AWS CLI

- Scripting : AWS CLI

https://github.com/awsdocs/aws-doc-sdk-examples/

https://github.com/awsdocs/aws-doc-sdk-examples/blob/master/python/