

$x_2$

# T7 : AWS II

## IAM: Gestión de Identidad

### Identity and Access Management

- Facilita el acceso seguro a los servicios de AWS
- Admin usuarios A y grupos A/G
- Podemos def. condiciones específicas (hora del día, IP origen, ...)
- Autenticación Múltifactor (MFA)

### Estado de Seguridad

- + No usar usuario root/root para admin AWS
- + Proteger con <ARPA> sin claves, de acceso asociadas
- + Crear A/IAM con privilegios de Admin
- + A/IAM: solo permisos necesarios
- + Crear A/G y asociar A, recomendado
- + Asociar políticas de contraseñas robustas → A
- + Activar CloudTrail (monitoreo la actividad de los A/I)

### Crear usuarios

- + Acceso por consola (web) y API
- + Si no seleccionamos ninguna opción de Permisos → A acción recursos
- + En ese momento de obtención de KEY ID + URL acceso consola

### Grupos de Usuarios A/G

- + Baja los > 5 perfiles de A/G
  - + Facilita gestión de permisos → nuevo perfil estructurado
- Buena Práctica:** A no tiene permisos A/G ni (políticas)

### Políticas de IAM

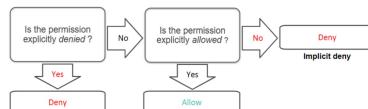
Conjunto de Reglas + condiciones → acciones

QUÉN    QUÉ    CÓMO

Políticas autorizadas  
en identidad recursos

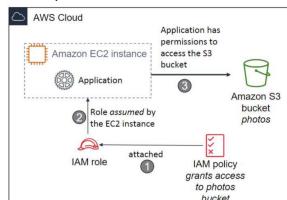


### Determinación de permisos IAM:



### Roles en AWS

- Permisos asociados a Recursos (no a A)
- Rol → A, API, ...
- No tienen ninguna credencial asociada
- Permiten asociar permisos temporales
- Permiten asociar permisos a una entidad sobre otra (ej. instancia EC2 → S3)



## CLI AWS. Instalación Linux

? LabUser → info para descargarte una identidad IAM en tu otro terminal

**CLI** → Acceso directo a los APIs públicos servicios AWS  
Basada en AWS SDK Python

### Instalación Plinux

Sincronizar la hora del sistema, manualmente o con NTP  
systemctl restart chronyd

Instalar los clientes AWS (comando)

```

$ curl https://awscli.amazonaws.com/AWSCLILinux-x86_64.rpm
$ sudo awscli-x86_64.rpm
$ sudo awscli-x86_64.rpm
$ ls -l /var/lib/dnf/yumcache/00000000000000000000000000000000
$ aws --version
$ aws s3 ls
$ aws s3 sync /tmp /var/www/html
$ aws configure
$ AWS_ACCESS_KEY_ID=AKIAJ4WQH8CZPZV5K3A
$ AWS_SECRET_ACCESS_KEY=3LqBzXfFkDwvIYUuR9GZdXnXfTmEJLx
$ AWS_DEFAULT_REGION=eu-west-1
$ AWS_DEFAULT_PROFILE=labuser
$ aws ec2 describe-regions
$ aws ec2 describe-instances
$ aws ec2 describe-keypairs
  
```

### Archivos de config y credenciales

La configuración introducida se almacena en los archivos config y credentials del directorio ~/.aws

```

$ cat ~/.aws/config
[default]
region = eu-west-1
output = json
[ec2]
region = eu-west-1
profile = labuser
  
```

Es posible crear otros perfiles de usuario (para usar otros perfiles se añade la cláusula --profile usuario a los comandos)

```

$ aws configure --profile usuario
  
```

### Config. y Precedencia

AWS & credenciales en el siguiente orden:  
Ajustes config

- Opciones de líneas de comandos  
perfil A, región / formato salida
- Variables de entorno : AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY, AWS\_SESSION\_TOKEN
- Archivos de credenciales del CLI  
Credenciales almacenadas en .aws (Linux, Mac OS X) o C:\Users\USUARIO\aws (Windows). Se admiten credenciales que contienen múltiples perfiles de usuario.
- Credenciales del tokenizador Amazon BlobStorage Tokenizer Service
- Credenciales de perfil de instancia configuradas mediante metadatos instancias EC2

### Uso de la interfaz de consola

Los comandos AWS tienen la siguiente estructura:

\$ aws <comando> <subcomando> [opciones y parámetros]

Por ejemplo, para listar todos los buckets S3 existentes:

\$ aws s3 ls

### Uso de la ayuda

\$ aws help  
\$ aws <comando> help  
\$ aws <comando> <subcomando> help

Para activar la depuración en la ejecución de un comando

\$ aws --debug <comando> <subcomando>

Un comando se puede ejecutar con un perfil distinto del predeterminado añadiendo la opción --profile

\$ aws ec2 describe-instances --profile user2

Si vamos a usar un perfil con nombre en varios comandos, podemos establecerlo en la variable de entorno AWS\_PROFILE

\$ export AWS\_PROFILE=user2

También podemos especificar el formato de salida y la región:

\$ aws ec2 describe-instances --output table --region us-west-1

Para habilitar el autocompleteo de comandos aws (en bash):

\$ complete -C '/usr/local/bin/aws\_completer' aws

Nota: Puede añadirse a ~/.bashrc

### Comandos → Servicio EC2

Crear un grupo de seguridad y una pareja de claves

\$ aws ec2 create-security-group --group-name prueba-sg --description "Grupo de seguridad de prueba. Acceso por SSH"

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

\$ aws ec2 create-key-pair --key-name prueba-key --query 'KeyMaterial' --output text > prueba-key.pem

\$ chmod 400 prueba-key.pem

Listar grupos de seguridad y claves (comandos describe). Podemos probar distintos formatos de salida.

\$ aws ec2 describe-security-groups --output json

\$ aws ec2 describe-key-pairs --output table

# T7 : AWS II

## (CLI AWS . Instalación Linux)

(comandos → servicio EC2)

Lanzamiento de instancia EC2:

la imagen (ami-xxxxxx)  
Antes: selecciona identif. grupo de sg (sg-xxxxxx)  
identif. subnet (subnet-xxxxxx)

```
$ aws ec2 describe-images --owners self amazon --filters
"Name=architecture,Values=x86_64" --query
"Images[*].ImageId,VirtualizationType"
$ aws ec2 describe-instances --filters
"Name=ImageId,Values=ami-2cni-kernel-x86_64-gp2"
"Name=Value,Values=available" --output text | jq -r ".Images |
sort_by(.CreationDate) | last().ImageId"
$ aws ec2 describe-security-groups
$ aws ec2 describe-subnets
$ yum install jq
```

La instancia creada tiene por defecto una IP pública asociada. Podemos conocer su valor partiendo del ID de la instancia (l-xxxxx)

```
$ aws ec2 describe-instances --query
'Reservations[*].Instances[*].[Placement,AvailabilityZone,
State.Name,InstanceId]' --output text
eu-west-1a
running
1-00b9bbb44ae6fa9e
```

Establecemos una conexión SSH con la instancia creada:

```
$ ssh -i pruebas-key.pem ec2-user@52.214.203.84
```

También podemos detener y volver a arrancar la instancia:

```
$ aws ec2 stop-instances --instance-ids l-00b9bbb44ae6fa9e
$ aws ec2 describe-instances --query
'Reservations[*].Instances[*].[Placement,AvailabilityZone,
State.Name,InstanceId]' --output text
eu-west-1a
stopping
```

Por último, podemos eliminar la instancia:

```
$ aws ec2 terminate-instances --instance-ids l-00b9bbb44ae6fa9e
$ aws ec2 describe-instances --query
'Reservations[*].Instances[*].[Placement,AvailabilityZone,
State.Name,InstanceId]' --output text
eu-west-1a
shutting-down
1-00b9bbb44ae6fa9e
```

## Comandos → S3

```
$ aws s3 mb s3://prueba-s3
$ aws s3 ls
$ aws s3 cp file.txt s3://prueba-s3 --grant-full-control uri=http://acs.amazonaws.com/global/AllUsers
$ aws s3 rb s3://prueba-s3 --force
```

## Filtros de búsqueda (-filters & --query)

Ejemplo: Vista instancias EC2

Podemos establecer filtros sobre propiedades de la instancia (mediante Name+Value):

```
$ aws ec2 describe-instances --filters "Name=instance-
Type,Values=t2.micro" --query "Reservations[*].Instances[*].medium"
"Name=availability-zone,Values=eu-west-1a"
```

Lo mismo, pero en formato JSON:

```
$ aws ec2 describe-instances --filters "[{"Name": "instance-
Type", "Values": ["t2.micro"]}]" --query "Reservations[*].Instances[*].medium"
```

Otro ejemplo: por tipo de imagen AMI:

```
$ aws ec2 describe-instances --filters "Name=image-
Id,Values=ami-03420b006acbf69d" --output json
```

Para limpiar la información visualizada (-query). Encadenamos claves hasta alcanzar el elemento necesario (para todas las instancias):

```
$ aws ec2 describe-instances --query "Reservations[*].Instances[*].imageId"
```

Información de una sola instancia, la número 1 (están numeradas a partir de la 0):

```
$ aws ec2 describe-instances --query "Reservations[1]"
```

Visualizar solo las propiedades deseadas, con un nombre de propiedad asociado:

```
$ aws ec2 describe-instances --query
'Reservations[*].Instances[*].[ARCHitecture,
cpu_type,instance_name]'
```

Otras servicios

Sintaxis comando

aws [comando] <command> <subcommand>  
[subcommands,...] [parámetros]

VPC2, RDS, Route53, ...

## SDK para AWS . Boto3

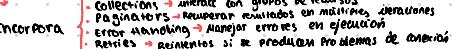
Boto3 facilita integración APPS servers Python para servicios AWS  
Proporciona 2 niveles de API

### Low-level Client

- Bajo nivel
- Acceso directo

### Resources

- Alto nivel
- Recopilaciones y obj de recursos
- Acción

Incorpora  Collection → interact con grupos de recursos  
Paginator → Recuperar resultados en múltiples iteraciones  
Error handling → Manejar errores en ejecución  
Retries → Resientes si se producen problemas de conexión

## Instalación Boto3

Podemos instalar boto3 con el gestor de paquetes de python (pip):

```
$ pip install boto3 --user
```

Una vez instalado, boto3 usa los mismos archivos de configuración y credenciales de aws (-~aws/credentials y ~aws/configure)

Para verificar su funcionamiento con el intérprete de Python:

```
$ python3
>>> import boto3
>>> ec2 = boto3.client('ec2')
>>> ec2.describe_instances()['Reservations'][0]['Instances'][0]
{'ImageId': 'ami-0742d4e679307206d',
 'InstanceId': 'i-00b9bbb44ae6fa9e',
 'InstanceState': 'stopped',
 'InstanceType': 't2.micro'}
>>> exit()
```

## Algunos métodos (client)

Gestión de instancias EC2

- describe\_instances
- create\_instances
- start\_instances / stop\_instances / reboot\_instances

Gestión de Buckets (cliente S3)

- create\_bucket
- upload\_file
- download\_file / download\_fileobj
- get\_bucket\_policy / put\_bucket\_policy / delete\_bucket\_policy
- get\_bucket\_acl
- put\_bucket\_website / delete\_bucket\_website

## Scripts en Python

Listar instancias EC2 (ListarInstancias.py)

```
#!/usr/bin/python
import boto3
ec2 = boto3.resource('ec2')
for instance in ec2.instances.all():
    print(instance.id, instance.state)
```

Crear una instancia EC2 (CrearInstancia.py)

```
#!/usr/bin/python
import boto3
ec2 = boto3.resource('ec2')
ec2.create_instances(ImageId='ami-0742d4e679307206d',
                     MinCount=1,
                     MaxCount=1,
                     InstanceType='t2.micro')
print(instance.id, instance.state)
```

Terminar instancias EC2 (TerminarInstancia.py)

Nota: pasamos como parámetro el id de la(s) instancia(s)

```
#!/usr/bin/python3
import sys
import boto3
ec2 = boto3.resource('ec2')
for instance_id in sys.argv[1:]:
    instance = ec2.Instance(instance_id)
    response = instance.terminate()
    print(response['TerminatingInstances'][0]['CurrentState']['Name'])
```