

Plataformas y APIS en la nube
© EDICIONES ROBLE, S.L.

Indice

Plataformas y apis en la nube	3
I. Introducción	3
II. Objetivos	4
III. Introducción a los servicios en la nube	4
3.1. Ventajas de los servicios en la nube	6
IV. Configuración inicial de AWS	7
4.1. AWS: acceso y seguridad (IAM)	7
4.2. AWS CLI (Command Line Interface)	10
4.3. VPC (Virtual Private Cloud)	12
V. Instancias, configuración EC2 (Elastic compute cloud) y servicios básicos	20
5.1. S3 (Simple Cloud Storage Service)*	30
5.2. RDS (Relational Database Service)*	32
VI. Herramientas de configuración y utilidades	36
6.1. Elastic Beanstalk*	36
6.2. SQS (Simple Queue Service)*	48
6.3. AWS Kinesis*	53
6.3.1. Amazon Kinesis Data Streams	54
6.3.2. Amazon Kinesis Data Firehose	54
6.3.3. Amazon Kinesis Data Analytics	54
VII. Resumen	55
Ejercicios	56
Caso práctico	56
Solución	56
Recursos	58
Enlaces de Interés	58
Bibliografía	58
Glosario.	58

Plataformas y apis en la nube

I. Introducción

AVISO IMPORTANTE: durante la unidad el alumno activará instancias y servicios en la nube desde AWS; una vez terminada la unidad, el caso práctico y el ejercicio de la práctica final, es **obligatorio** que el alumno pare todos los servicios e instancias activadas, ya que una vez superado el crédito gratuito AWS comienza a cobrar por el uso de sus servicios.

A lo largo de la unidad, se irán detallando los puntos donde debéis terminar o parar los servicios o instancias para no tener problemas con AWS.

Como se ha podido ver en módulos anteriores, las máquinas virtuales son una herramienta importante para solucionar algunos de los problemas ligados a la instalación y configuración tanto de los sistemas operativos necesarios como de las aplicaciones que se utilizan en la recogida y análisis de datos. Sin embargo, sigue planteándose la siguiente pregunta: ¿dónde se instalarán estas máquinas virtuales? Después de todo, no todos los equipos/servidores son compatibles con todas las máquinas virtuales, o puede que los recursos disponibles sean menores que los necesarios para que funcionen las aplicaciones dentro de la máquina virtual; al fin y al cabo, debemos tener en cuenta cosas como arquitectura de CPU, memoria, espacio físico, etc.

Posiblemente, también se deberán tener en cuenta detalles como la configuración y el mantenimiento del sistema operativo junto con la aplicación encargada de gestionar las máquinas virtuales, así como la red en la que se encuentre el servidor e, incluso, el mantenimiento de este.

En este punto, para intentar solucionar estos y otros problemas, es en el que entran en juego los servicios en la nube, el objeto de estudio de esta unidad. Más en concreto, se estudiarán los servicios web de Amazon, que denominaremos a partir de ahora AWS.

Es importante destacar que se utilizará AWS como ejemplo para dar a conocer las posibilidades de los servicios en la nube, pero que existen otros proveedores (como Microsoft Azure, IBM Bluemix y Google Cloud Platform, por citar algunos) con funcionalidades similares y con sus propios puntos fuertes y débiles.

En esta unidad, primero se explicarán los distintos tipos de servicios en la nube y después las ventajas de su utilización en contraposición con los equipos físicos. Para terminar, se estudiará la creación y configuración de máquinas virtuales, junto con otros servicios útiles que se podrán encontrar dentro de AWS.

Además, también se puede utilizar **Amazon CloudWatch** para monitorizar los cargos por el servicio de AWS y recibir alertas de facturación, que pueden ser usadas para notificar a un usuario **si accidentalmente excede el uso cubierto por la capa gratuita** o Free Tier. Se puede ampliar información en los siguientes enlaces:

- <https://aws.amazon.com/cloudwatch/>
- <https://aws.amazon.com/cloudwatch/getting-started/>



Importante: en esta unidad, el código informático está en la plataforma mediante capturas de pantalla. Pero, además, de cara a que los alumnos puedan copiar-pegar el código que requieran, se facilita esta unidad en formato Word en el siguiente enlace.



II. Objetivos

Los objetivos que se deben alcanzar con esta unidad son:

- Conocer los distintos tipos de servicios en la nube que podemos encontrar.
- Comprender las ventajas del uso de servicios en la nube.
- Saber utilizar AWS, tanto por consola como por la aplicación web, para la creación y configuración de máquinas virtuales y otras utilidades.

III. Introducción a los servicios en la nube

Hace muchos años, si una empresa necesitaba electricidad para desarrollar sus actividades, la única alternativa era tener su propia fuente de energía eléctrica. Esto conllevaba costes de instalación, mantenimiento, infraestructuras y personal, entre otros, algo que podía resultar prohibitivo para empresas pequeñas. Con el tiempo, este modelo fue cambiando. Empezó a ser más productivo, y por tanto más económico, tener una única planta energética de gran capacidad antes que muchas pequeñas. De esta forma, las empresas ya no necesitaban hacer una gran inversión para tener este tipo de infraestructuras; ya podían disponer de electricidad bajo demanda.

Esto es una analogía de lo que está ocurriendo actualmente con la informática. Ahora no es necesario hacer una gran inversión para disponer de un CPD en el que guardar un número finito de servidores, ni de los servidores para desplegar las aplicaciones, ahora se puede disponer de todo eso bajo demanda y según las necesidades de cada proyecto.

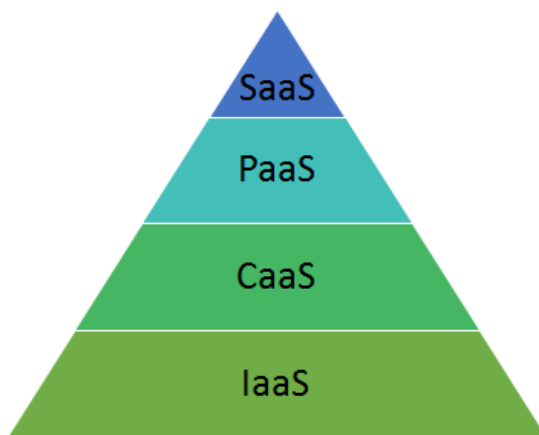


Figura 1. Pirámide de representación de los distintos modelos de servicios.

Fuente: elaboración propia.

Hasta hace poco se podían distinguir tres modelos principales de servicios en la nube: IaaS, PaaS y SaaS; sin embargo, desde hace un tiempo se ha añadido un nuevo modelo denominado CaaS. Estos modelos se tienden a representar en una pirámide (figura 1.) en la que, cuanto más nos acercamos a la base, más control tenemos sobre los recursos.

SaaS

Software-as-a-Service

Es un modelo de distribución de *software* donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación (TIC), a los que se accede por medio de internet desde un cliente. En este tipo de servicios se accederá normalmente a través del navegador sin atender al *software* y todo el desarrollo, mantenimiento, actualizaciones y copias de seguridad es responsabilidad del proveedor.

En este caso se tiene poco control, los usuarios se sitúan en la parte más arriba de la capa del servicio. Si el servicio se cae es responsabilidad de proveedor hacer que vuelva a funcionar.



Google Docs, DropBox, Microsoft Office 365 o Salesforce son algunos de los ejemplos que podemos encontrar de este modelo.

PaaS

Platform-as-a-Service

Es un entorno basado en la nube que se puede usar para desarrollar, probar, ejecutar y administrar aplicaciones. Este enfoque permite la prestación del entorno de desarrollo, sin la complejidad de comprar, crear o administrar ningún tipo de infraestructura.

Este es un modelo que reduce bastante la complejidad a la hora de desplegar y mantener aplicaciones ya que las soluciones PaaS gestionan automáticamente la escalabilidad usando más recursos si fuera necesario. Los desarrolladores, aun así, tienen que preocuparse de que sus aplicaciones estén lo mejor optimizadas posible para consumir menos recursos, pero todo ello sin entrar en las máquinas.



Ejemplos de este modelo son: Heroku, Google App Engine, Django.

IaaS**Infraestructure-as-a-Service**

En este modelo se tiene mucho más control que con PaaS, aunque a cambio de eso será necesario encargarse de la gestión de infraestructura. La idea básica es la externalización de servidores para espacio en disco, bases de datos o tiempo de computación, en lugar de tener servidores *in house*. Con una infraestructura como servicio (IaaS) lo que se tiene es una solución basada en virtualización en la que se paga por consumo de recursos: espacio en disco utilizado, tiempo de CPU, espacio en base de datos, transferencia de datos.

En este caso también será necesario encargarse de escalar las aplicaciones según las necesidades del servicio, además de preparar todo el entorno en las máquinas, aunque en algunos casos se tendrán plantillas predefinidas con las opciones más comunes para una rápida configuración. Gracias a estas plantillas, en algunos casos, este modelo se puede comportar como los otros modelos.



Google Cloud Platform, Amazon Web Services, Microsoft Azure e IBM Bluemix entran dentro de esta categoría.

CaaS**Container-as-a-Service**

Es un modelo emergente, entre IaaS y PaaS, que permite a los usuarios controlar y desplegar contenedores, aplicaciones y clústers a través de virtualización basada en contenedores. Esto permite a los departamentos de IT y desarrollo construir aplicaciones seguras y escalables con mayor comodidad y control.

Con los contenedores se podrán controlar los recursos que se asignan a las aplicaciones, aunque también es necesaria la correcta configuración de los mismos para asegurar su funcionamiento. Por otro lado, gracias a las propiedades de los contenedores, se pueden realizar despliegues rápidos en distintos entornos e incluso usando diferentes proveedores de servicios *cloud*. En otras palabras, se puede tener un entorno local de desarrollo y pruebas para las aplicaciones y, una vez finalizadas, subir los contenedores a cualquiera de las empresas que brindan este modelo y poner a funcionar las aplicaciones al momento.



Los ejemplos más comunes utilizados con este modelo son Dockers y Kubernetes.

3.1. Ventajas de los servicios en la nube

1

Las ventajas de estos servicios dependen en muchos casos de las necesidades del proyecto que se esté abordando; por ejemplo, en una empresa que tenga un servicio que usan unos pocos empleados y que es fácil de mantener, puede que sea más económico y más eficiente realizar una instalación en un servidor local. Sin embargo, ¿qué ocurre si después de un tiempo la empresa necesita habilitar el servicio para que sea usado desde cualquier parte del mundo y no solo desde la sede donde está instalado? ¿Y si los usuarios pasan de ser unos pocos a unos miles? ¿O si el servicio necesita estar activo 24x7 y como los datos son muy importantes es necesario realizar un *backup* constante de ellos?

2

Para estos problemas hay muchas soluciones, como por ejemplo: preparar el servicio para que pueda estar accesible fuera de la red, posiblemente contratando un nuevo servicio para el acceso a internet, adquirir un servidor más potente, o mejor, varios servidores para tener redundancia y una unidad de cintas para realizar los *backups*. Todas estas soluciones, aunque efectivas, llevan tiempo y en varios casos gastos adicionales. Para contratar una mejora o una nueva línea de salida a internet es necesario ponerse en contacto con el proveedor del servicio y, normalmente, esperar unos días o unas semanas hasta que actualicen el servicio. En el caso de los servidores, salvo excepciones, entre que se solicitan, se reciben y se configuran pasa un tiempo, por no hablar de que, en caso de varios servidores, será necesario comprar un RACK y, en el peor de los casos, preparar un sitio con refrigeración adicional para guardarlo, además de adquirir SAls para prevenir problemas por apagones y picos de tensión. En función de la magnitud y las necesidades, la actualización del servicio puede tardar entre varios días y varios meses.

3

Otra de las formas de resolver estos problemas es contratar un proveedor IaaS y subir el servicio a la nube. De esta forma la actualización del servicio pasaría a estar operativa, como máximo, en unas horas. En este ejemplo se pueden encontrar algunas de las facetas más importantes de los servicios en la nube: flexibilidad, agilidad, confiabilidad, disponibilidad y reducción de costes.

4

Otro ejemplo bastante común en el que se pueden ver con facilidad estas ventajas lo encontramos cuando nuestro servicio tiene picos de cargas de trabajo a lo largo del tiempo. Por ejemplo, un servicio que durante el fin de semana o durante algún evento concreto multiplique entre 2 y 10 veces el número de usuarios o procesos simultáneos. En este caso, si usamos equipos físicos, tendremos que decidir entre adquirir equipos mejores para soportar los picos de trabajo, teniendo en cuenta que estarán infrutilizados la mayor parte del tiempo, o por el contrario dejar de dar servicio a los usuarios que sobrepasen cierto umbral o provocar que se ralenticen los procesos. Sin embargo, si nuestro servicio se encuentra en la nube, el escalado tanto hacia arriba (para soportar los picos de trabajo) como hacia abajo (para desconectar las máquinas que ya no se utilizan o reducir sus recursos) se podrá gestionar con facilidad e incluso realizar configuraciones para que estos procesos se ejecuten de forma automática.

IV. Configuración inicial de AWS

4.1. AWS: acceso y seguridad (IAM)



Para acceder a la consola de administración de AWS, escribimos la siguiente URL en el navegador: <https://console.aws.amazon.com>.

En caso de no tener cuenta, se puede crear una accediendo desde la URL <https://portal.aws.amazon.com/billing/signup#/start> o registrándose como estudiante en <https://aws.amazon.com/es/education/awseducate>.

Registrarse en esta última es recomendable, independientemente de si se tiene una cuenta de AWS o no, ya que si nos aprueban la solicitud tendremos acceso a más documentación, entre otras ventajas.

IMPORTANTE: para poder acceder usando la interfaz de línea de comando, es necesario tener una cuenta personal registrada. Cuando se solicite el registro como estudiante, tendremos la opción de añadir el ID de la cuenta de una cuenta de AWS existente; es decir, una de las opciones es asociar una cuenta de AWS existente y entonces se añaden los créditos a esa cuenta, aunque una vez realizado el registro este tarda unos días en completarse.

El ID de la cuenta se podrá encontrar accediendo a los datos de la cuenta (figuras A y B):

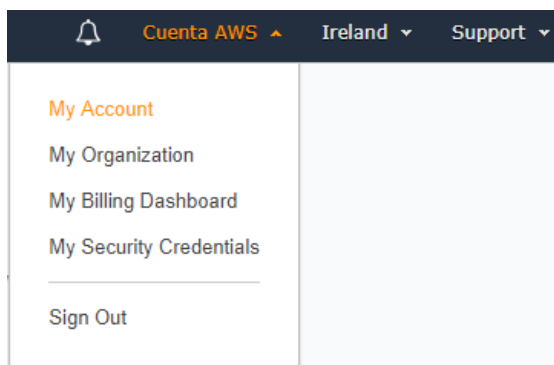


Figura A. Captura de pantalla de la consola de administración AWS.

▼ Configuración de cuenta

ID de cuenta: [redacted]
 Vendedor: AWS Inc.
 Nombre de cuenta: Cuenta AWS
 Contraseña: *****

Figura B. Captura de pantalla de la consola de administración AWS.

ADVERTENCIA: es muy importante señalar que, una vez finalizados los ejercicios o después de realizar pruebas, las instancias, aplicaciones o buckets creados deben ser destruidos para evitar incurrir en gastos en la cuenta. Si se ha registrado la cuenta como estudiante se tendrá un número de créditos que se utilizarán antes de imputar cargos a la tarjeta configurada. También, durante el primer año de vida de la cuenta, AWS dispone de una capa gratuita que permite realizar pruebas sin generar gastos; sin embargo, las condiciones dependen del servicio y, normalmente, están indicadas en la consola de administración. Por ejemplo, cuando se intenta crear una instancia en EC2, se indica que, en las instancias del tipo t2.micro, "Free tier eligible" indica que se encuentran dentro de la capa gratuita siempre que no se usen más de 750 horas de estas instancias por mes (figura C).

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-
<input checked="" type="checkbox"/>	General purpose	t2.micro	1	1	EBS only	-
<input type="checkbox"/>	General purpose					
<input type="checkbox"/>	General purpose	t2				
<input type="checkbox"/>	General purpose					

Micro instances are eligible for the AWS free usage tier. For the first 12 months following your AWS sign-up date, you get up to 750 hours of micro instances each month. When your free usage tier expires or if your usage exceeds the free tier restrictions, you pay standard, pay-as-you-go service rates.

[Learn more](#) about free usage tier eligibility and restrictions

Figura C. Captura de pantalla de la consola de administración AWS.

1

Una vez dentro de la consola de administración, se podrá acceder a los distintos servicios proporcionados por AWS. Para ello, pulsando en la opción superior derecha "Services" se podrá ver la lista de estos servicios, además de poder buscarlos en la barra de búsqueda.

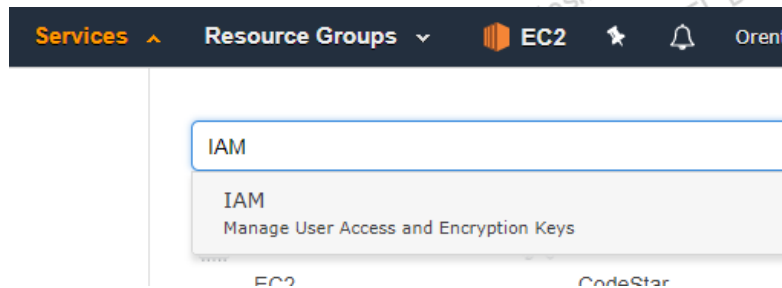


Figura 2. Captura de pantalla de la consola de administración AWS.

2

Ahora es necesario buscar y acceder al servicio IAM (figura 2). Desde aquí se podrá controlar el acceso a los recursos de AWS. Esta parte es importante para poder crear usuarios con los permisos necesarios para usar la interfaz de línea de comando de AWS.

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[+ Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* ☒ **Programmatic access**

Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☐ **AWS Management Console access**

Enables a **password** that allows users to sign-in to the AWS Management Console.

Figura 3. Captura de pantalla de la consola de administración AWS.

3

Una vez dentro del servicio IAM, en el menú hay que seleccionar “Users” y el botón “Add user”. Dentro de esta ventana se necesita el nombre del nuevo usuario y el tipo de acceso, en este caso “Programmatic access” ya que solo se utilizarán para acceder por la línea de comandos (figura 3).

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonEC2FullAccess
Managed policy	AmazonVPCFullAccess

Figura 4. Captura de pantalla de la consola de administración AWS.

4

En la siguiente ventana se administran los permisos del usuario. Para este caso, se selecciona “Attach existing policies directly” y en la barra de búsqueda se buscan las políticas de permisos según el usuario. Para el usuario con acceso a EC2, se necesitan las políticas “AmazonEC2FullAccess” y “AmazonVPCFullAccess”. Para el usuario con acceso a S3, “AmazonS3FullAccess”. RDS, “AmazonRDSFullAccess”. Y para el usuario con el que se accederá a SQL, “AmazonSQSFullAccess”. Una vez seleccionadas las políticas se continúa para revisar la configuración (figura 4).

[Download .csv](#)

User	Access key ID	Secret access key
▶ user_ec2_master	AKIAIRV7VMXGHLSPAXNA	***** Show

Figura 5. Captura de pantalla de la consola de administración AWS.

Como los usuarios se han creado con el tipo de acceso “Programmatic access” se mostrará al finalizar las llaves de acceso de CLI (figura 5).

4.2. AWS CLI (Command Line Interface)

La interfaz de línea de comando se utiliza para administrar la mayoría de los productos y servicios que se pueden encontrar en AWS. Con esta herramienta se pueden crear tareas automáticas mediante *scripts* para realizar trabajos repetitivos.

Aunque se pueda instalar tanto en sistemas Linux como Unix, Mac y Windows, para utilizar AWS CLI en la unidad y la práctica final, utilizaremos para implementar todos los comandos de la unidad cualesquiera de las máquinas virtuales proporcionadas en la unidad 1 de este módulo basadas en Ubuntu 18.04.

Para su instalación es necesario tener Python 2 versión 2.6.5+ o Python 3 versión 3.3+, además del sistema de gestión de paquetes de Python (pip). Por último, debido a que la salida de la mayoría de los comandos del CLI se realiza con el formato *json* se instalará *jq*, debido a la necesidad de tener una herramienta para el procesamiento de este formato en la línea de comandos.

```
$ sudo apt-get install python3 python3-pip jq
```

Para instalar AWS CLI se usará el gestor de paquetes de Python:

```
$ sudo apt install awscli
```

```
$ pip3 install awscli --upgrade --user
```

Se comprueba la versión:

```
$ aws --version
aws-cli/1.14.32 Python/3.5.2 Linux/4.4.0-112-generic botocore/1.8.36
```

Para configurar el CLI se utilizarán las llaves de acceso generadas previamente:

```
$ aws configure
AWS Access Key ID [None]: AKIAIRV7VMXGHLSPAXNA
AWS Secret Access Key [None]: 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
Default region name [None]: eu-west-1
Default output format [None]:
```

Al utilizar usuarios con distintos permisos, es necesario configurar distintos perfiles en la herramienta, como por ejemplo el de S3:

```
$ aws configure --profile S3
AWS Access Key ID [None]: AKIAIHBARCAE6RTXUZ2Q
AWS Secret Access Key [None]: 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
Default region name [None]: eu-west-1
Default output format [None]:
```

Para utilizar los perfiles solo hay que añadir la opción “—profile” y el nombre del perfil al ejecutar el comando, con el siguiente formato: `aws --profile <perfil> <servicio> <comando> <opciones>`. Por ejemplo, para listar el contenido de S3 usamos

```
$ aws --profile S3 s3 ls
```

4.3. VPC (Virtual Private Cloud)

Antes de adentrarnos en la parte lógica de la red que podemos configurar en AWS, tenemos que entender los términos de regiones y zonas de disponibilidad que conforman la estructura física de AWS.



Las regiones son áreas geográficas independientes en las que se hospeda todo lo que tenga que ver con las instancias que podemos crear y sus configuraciones, entre otros servicios. Algunas de las regiones que tenemos disponibles son: EE. UU. Este, EE. UU. Oeste, UE (Fráncfort), UE (Irlanda), UE (Londres) o UE (París) entre otras. Al crear nuestras instancias podemos escoger la región que más nos interese según la ubicación de los usuarios (cuanto más cerca se encuentre el servicio con respecto a los usuarios, mejor será la latencia), o incluso por restricciones debidas a leyes de seguridad o protección de datos (al manejar datos confidenciales podemos tener la restricción de solo poder usar servidores dentro de la Unión Europea).

Dentro de las regiones encontramos las zonas de disponibilidad. Cada zona es el equivalente a un CPD independiente del resto. Cada región puede tener una o más zonas. Si tiene más de una zona y distribuimos los servicios entre varias de ellas, en el improbable caso que una de ellas falle, el resto de las zonas seguirán funcionando con normalidad.

A diferencia de las regiones y zonas, que son separaciones físicas entre nuestros recursos, las VPC son separadores lógicos dentro de la nube de AWS. Una VPC es el equivalente a una red física como puede ser la de una oficina, la universidad o incluso nuestra casa. Para crearla, al igual que una red física, es necesario indicar el CIDR, que se compone por rango de IP y la máscara, por ejemplo 10.0.0.0/16.

Las VPC se dividen a su vez en subredes, igual que en una red física. Las subredes nos permiten dividir y gestionar mejor las instancias, ya que podemos configurar subredes públicas (con acceso a internet) y privadas (solo accesibles dentro de la nube de AWS). Incluso podemos restringir el tráfico entre cada una de ellas para poder aislarlas y protegerlas.

Cada VPC de AWS está asociada a una región. Las subredes de esa VPC tienen asociada, además, una zona en concreto dentro de la región. Aunque una subred no puede existir en más de una zona, dentro de una zona pueden coexistir varias subredes. Todas las instancias dentro de una misma zona, aunque se encuentren en subredes distintas, no generan gastos adicionales por transferencia de datos entre ellas y presentan unas tasas de transferencia mayores (figura 6).

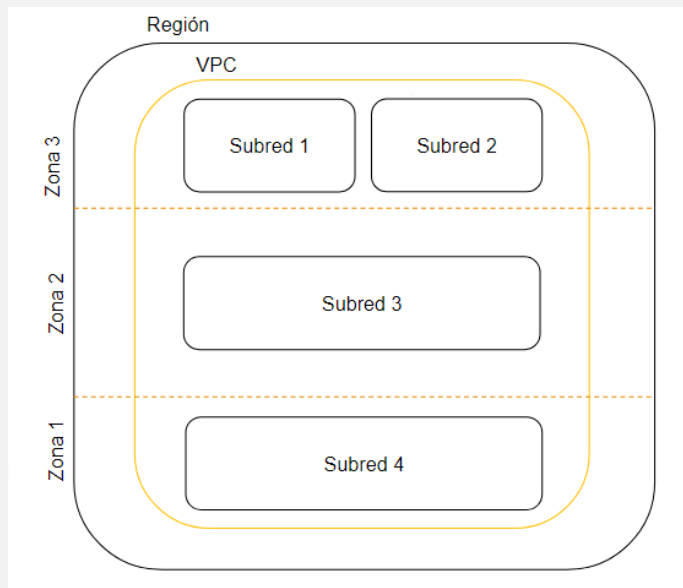


Figura 6. Ejemplo de configuración de VPC y subredes. *Fuente:* elaboración propia.

1

Para trabajar con las VPC usando CLI, trabajaremos usando el servicio ec2. Para crear una VPC usaremos el comando “create-vpc” indicando el CIDR:

```
$ aws ec2 create-vpc --cidr-block 10.0.0.0/16
{
  "Vpc": {
    "VpcId": "vpc-e761a181",
    "Tags": [],
    "CidrBlock": "10.0.0.0/16",
    "State": "pending",
    "DhcpOptionsId": "dopt-f1406096",
    "CidrBlockAssociationSet": [
      {
        "CidrBlock": "10.0.0.0/16",
        "AssociationId": "vpc-cidr-assoc-0f948c67",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ],
    "InstanceTenancy": "default",
    "Ipv6CidrBlockAssociationSet": [],
    "IsDefault": false
  }
}
```

Al ejecutar el comando, nos devuelve el ID de la VPC en el campo "VpcId". También nos indica que su estado es "pending", por lo que tendremos que esperar unos minutos para que se termine de configurar la VPC en la infraestructura de AWS.

Para ver el estado de la VPC usamos el comando "describe-vpcs" junto con la ID (si no especificamos la ID, nos mostrará todas las VPCs disponibles).

```
$ aws ec2 describe-vpcs --vpc-ids vpc-e761a181
{
  "Vpcs": [
    {
      "DhcpOptionsId": "dopt-f1406096",
      "CidrBlock": "10.0.0.0/16",
      "VpcId": "vpc-e761a181",
      "State": "available",
      "IsDefault": false,
      "InstanceTenancy": "default",
      "CidrBlockAssociationSet": [
        {
          "CidrBlock": "10.0.0.0/16",
          "AssociationId": "vpc-cidr-assoc-0f948c67",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ]
    }
  ]
}
```

Ahora se puede ver que la VPC está disponible y puede utilizarse.

3

Una vez creada y disponible, para poder usarla es necesario añadir las subredes. Esto lo hacemos con el comando “create-subnet”. Las opciones que debemos especificar a la hora de crearla son el ID de la VPC a la que estará asociada y el CIDR, el cual tiene que estar dentro del CIDR de la VPC y además no puede solaparse con el CIDR de otra subred. Otra opción importante es la zona de disponibilidad, aunque si no se especifica será escogida automáticamente por AWS.

```
$ aws ec2 create-subnet --availability-zone eu-west-1b --cidr-block
10.0.0.0/24 --vpc-id vpc-e761a181
{
  "Subnet": {
    "State": "pending",
    "CidrBlock": "10.0.0.0/24",
    "MapPublicIpOnLaunch": false,
    "AvailabilityZone": "eu-west-1b",
    "SubnetId": "subnet-4a775b11",
    "AssignIpv6AddressOnCreation": false,
    "AvailableIpAddressCount": 251,
    "VpcId": "vpc-e761a181",
    "DefaultForAz": false,
    "Ipv6CidrBlockAssociationSet": []
  }
}
```

Al igual que al crear la VPC, el comando nos devuelve la ID de la subred en el campo “SubnetID” y nos indica que su estado es pendiente.

4

Un dato importante al crear una VPC es que esta se encuentra completamente aislada de otras VPCs y de internet. Para permitir que el tráfico salga es necesario crear un *gateway* de salida de internet, para lo que usamos el comando: “create-internet-gateway”:

```
$ aws ec2 create-internet-gateway
{
  "InternetGateway": {
    "Attachments": [],
    "Tags": [],
    "InternetGatewayId": "igw-899127ee"
  }
}
```

La creación y configuración del *gateway* se realiza automáticamente en AWS. Ahora solo hay que añadirlo a la VPC. Para esto usamos el comando: “attach-internet-gateway” con su ID y el ID de la VPC:

```
$ aws ec2 attach-internet-gateway --internet-gateway-id igw-899127ee --vpc-id
vpc-e761a181
```

5

Ahora las subredes de esta VPC tienen la capacidad de enviar tráfico a internet; sin embargo, por defecto, el único tráfico que tienen permitido es entre otras subredes de su propia VPC. Para modificar este comportamiento es necesario modificar la tabla de rutas asociada a la VPC o a la subred. Al crear una subred, por defecto se usa la tabla de la VPC. Para modificarla, primero se busca cuál es la tabla asociada con el comando “describe-route-tables” y la opción de filtro para que muestre solo aquellas asociadas a la VPC:

```
$ aws ec2 describe-route-tables --filters "Name=vpc-id, Values=vpc-e761a181"
{
  "RouteTables": [
    {
      "Associations": [
        {
          "Main": true,
          "RouteTableAssociationId": "rtbassoc-683af310",
          "RouteTableId": "rtb-1d71647b"
        }
      ],
      "RouteTableId": "rtb-1d71647b",
      "PropagatingVgws": [],
      "Routes": [
        {
          "GatewayId": "local",
          "State": "active",
          "DestinationCidrBlock": "10.0.0.0/16",
          "Origin": "CreateRouteTable"
        }
      ],
      "VpcId": "vpc-e761a181",
      "Tags": []
    }
  ]
}
```

6

La tabla de rutas por defecto es aquella que tenga el campo “Main” con el valor “true”. Como se puede ver, solo tiene una ruta configurada y que podemos traducir como “todo el tráfico con destino a la red 10.0.0.0/16 es enviado al *gateway* local”; de esta forma, toda subred que creemos y que no tenga su propia tabla de rutas podrá intercambiar información con otras subredes de la VPC.

Para añadir una nueva ruta a esta tabla usaremos el comando “create-route” y en las opciones el ID de la tabla, la red de destino (en este caso todas) y el *gateway*.

```
$ aws ec2 create-route --route-table-id rtb-1d71647b --destination-cidr-block
0.0.0.0/0 --gateway-id igw-899127ee
{
  "Return": true
}
```

Al devolver “true” indica que se ha creado correctamente. Podemos comprobarlo usando nuevamente “describe-route-tables”:


```
$ aws ec2 describe-route-tables --filters "Name=vpc-id, Values=vpc-e761a181"
{
  "RouteTables": [
    {
      "Routes": [
        {
          "GatewayId": "local",
          "DestinationCidrBlock": "10.0.0.0/16",
          "Origin": "CreateRouteTable",
          "State": "active"
        },
        {
          "GatewayId": "igw-899127ee",
          "DestinationCidrBlock": "0.0.0.0/0",
          "Origin": "CreateRoute",
          "State": "active"
        }
      ],
      "Tags": [],
      "PropagatingVgws": [],
      "Associations": [
        {
          "Main": true,
          "RouteTableAssociationId": "rtbassoc-683af310",
          "RouteTableId": "rtb-1d71647b"
        }
      ],
      "VpcId": "vpc-e761a181",
      "RouteTableId": "rtb-1d71647b"
    }
  ]
}
```

Ahora aparece una nueva ruta que indica que todo el tráfico sale por el *gateway* igw-899127ee. Hay que tener en cuenta que, al usar las rutas, estas se comprueban en orden de más específica a más genérica. Por ejemplo, en este caso, cualquier transmisión de datos a la IP 10.0.2.36 se enviará al *gateway* local porque el CIDR de esa ruta es el más parecido a esta IP; por el contrario, una transmisión a la IP 54.80.24.12 saldrá por el *gateway* igw-899127ee porque, aunque es genérico, es el único con el que tiene concordancia. }

Con esta configuración, todas las subredes que no tengan una tabla propia tendrán acceso a internet; sin embargo, si se desea crear una red privada que solo pueda ver las subredes locales, se deberá crear una tabla nueva usando el comando “create-route-table” junto con el ID de la VPC.

```
$ aws ec2 create-route-table --vpc-id vpc-e761a181
{
  "RouteTable": {
    "Tags": [],
    "VpcId": "vpc-e761a181",
    "RouteTableId": "rtb-06effa60",
    "PropagatingVgws": [],
    "Associations": [],
    "Routes": [
      {
        "State": "active",
        "Origin": "CreateRouteTable",
        "GatewayId": "local",
        "DestinationCidrBlock": "10.0.0.0/16"
      }
    ]
  }
}
```

Como se puede observar, por defecto todas las tablas tienen configurado permitir el tráfico entre subredes. Ahora que ya se ha creado, podemos asociarla a la subred con "associate-route-table":

```
$ aws ec2 associate-route-table --route-table-id rtb-06effa60 --subnet-id
subnet-4a775b11
{
  "AssociationId": "rtbassoc-325f974a"
}
```

Si se ejecuta nuevamente el descriptor de las tablas, se podrá ver la nueva tabla asociada a la subred.

```
$ aws ec2 describe-route-tables --filters "Name=vpc-id, Values=vpc-e761a181"
{
  "RouteTables": [
    {
      "Tags": [],
      "RouteTableId": "rtb-1d71647b",
      "VpcId": "vpc-e761a181",
      "Associations": [
        {
          "Main": true,
          "RouteTableId": "rtb-1d71647b",
          "RouteTableAssociationId": "rtbassoc-683af310"
        }
      ],
      "PropagatingVgws": [],
      "Routes": [
        {
          "State": "active",
          "Origin": "CreateRouteTable",
          "DestinationCidrBlock": "10.0.0.0/16",
          "GatewayId": "local"
        },
        {
          "State": "active",
          "Origin": "CreateRoute",
          "DestinationCidrBlock": "0.0.0.0/0",
          "GatewayId": "igw-899127ee"
        }
      ]
    },
    {
      "Tags": [],
      "RouteTableId": "rtb-06effa60",
      "VpcId": "vpc-e761a181",
      "Associations": [
        {
          "Main": false,
          "RouteTableId": "rtb-06effa60",
          "SubnetId": "subnet-4a775b11",
          "RouteTableAssociationId": "rtbassoc-325f974a"
        }
      ],
      "PropagatingVgws": [],
      "Routes": [
        {
          "State": "active",
          "Origin": "CreateRouteTable",
          "DestinationCidrBlock": "10.0.0.0/16",
          "GatewayId": "local"
        }
      ]
    }
  ]
}
```

V. Instancias, configuración EC2 (*Elastic compute cloud*) y servicios básicos

En AWS las instancias son el equivalente a las máquinas virtuales y forman parte del núcleo de todo el sistema.*

* **AVISO IMPORTANTE:** una vez terminada la unidad, el ejercicio práctico y la práctica final, se recomienda **parar la instancia** para no agotar el crédito gratuito que AWS pone a disposición de las cuentas con licencia de estudiante, ya que de lo contrario comenzará a cobrarnos por su uso, aunque no hagamos nada en la instancia (se recomienda ver el apartado de *billing*, facturas, de AWS para conocer el estado de nuestro crédito).

Al crear una instancia, lo primero que necesitamos saber es la imagen que vamos a utilizar para crearla, lo que a partir de ahora denominaremos AMI (*Amazon Machine Images*). Las AMI son similares a los ficheros ova utilizados con la aplicación VirtualBox para crear máquinas virtuales preconfiguradas, en este caso AWS nos permite seleccionar entre varios sistemas operativos, o incluso entre sistemas operativos con aplicaciones ya instaladas y configuradas (Jenkins, Esri ArcGIS, Pentaho, Tableau Server, entre otras muchas). Hay que tener en cuenta que las AMI con aplicaciones preinstaladas pueden suponer un coste adicional, sobre todo aquellas que requieren licencia. Un detalle muy importante sobre las AMI es la posibilidad de crear nuestras propias AMI basadas en una instancia ya configurada. Esto nos da la capacidad de crear una instancia con un sistema operativo de base, instalarle las aplicaciones que necesitemos o generar *scripts* en la instancia para ejecutar los procesos que necesitemos y crear una AMI. De esta forma, si necesitamos una instancia idéntica solo hay que utilizar esa AMI sin tener que instalar todo nuevamente.

Una vez que conozcamos la AMI que vamos a utilizar para crear la instancia, es necesario saber el tipo de instancia que crearemos. Esto es, cuánta memoria, CPU, uso de red y acceso a disco usaremos. Aunque no podemos configurar exactamente esos valores, AWS nos facilita una buena variedad de configuraciones distintas que podemos utilizar.

Tipos de instancias

Los tipos de instancias de los que disponemos son:

- Uso general: M3, M4, M5, T1 y T2. Presentan un equilibrio entre todas las especificaciones.
- Optimizadas para cómputo: C3, C4 y C5. Normalmente tienen más y mejores CPUs que el resto de tipos.
- Optimizadas para uso de memoria: R3, R4, X1 y X1e. La memoria configurada en este tipo de instancias es mayor y más eficiente que en el resto.
- Optimizadas para acceso a disco: D2, I3, H1. Este tipo de instancia añade un disco adicional con mejores características y mayor número de operaciones de entrada y salida que el resto.
- Optimizadas para el uso de GPU y aceleradores: F1, G3, P2 y P3. A diferencia del resto, este tipo de instancias tiene acceso a GPUs.

Dentro de cada tipo de instancia tendremos que especificar el tamaño de la misma; por ejemplo, en el caso de las T2, entre las opciones disponibles están las t2.small (1 CPU y 2 GB de memoria) y las t2.medium (2 CPU y 4 GB de memoria).

Una de las grandes ventajas que tenemos es la posibilidad de cambiar el tipo de instancia después de haberla creado; gracias a esto, en caso de no conocer de forma específica los recursos que necesitaremos para el correcto funcionamiento de nuestro servicio, siempre podemos empezar con una aproximación y monitorizar la instancia para poder ajustar el tipo y el tamaño correctamente.



Aunque estos dos datos, la AMI y el tipo, son suficientes para crear una instancia, podemos especificar otros detalles para que la instancia se adecue lo mejor posible a nuestras necesidades. Por ejemplo, si tenemos configurada alguna subred (esto lo veremos más adelante), podemos especificarla, o la cantidad de discos y el tamaño de los mismos que tendrá inicialmente, incluso podemos especificar si queremos que al apagarse se destruya o, por el contrario, si queremos proteger la instancia ante una destrucción accidental. Otro de los detalles que podemos configurar son los grupos de seguridad.

Grupos de seguridad

Los grupos de seguridad (a partir de ahora SG) son similares a la configuración de un cortafuego: si un puerto no está especificado en el SG, no podremos acceder por él. En estos SG no solo configuramos los puertos a los que podemos acceder, también especificamos desde dónde usando la nomenclatura básica de redes (IP/Mascara).



Por ejemplo, un SG que permita el acceso al puerto 80 desde cualquier parte del mundo tendría una configuración de entrada como esta: Protocol: TCP, Port Range: 80, Source: 0.0.0.0/0. Si además tiene configurado el acceso SSH solo desde su red, tendría además una configuración similar a: Protocol: TCP, Port Range: 22, Source: 10.0.0.0/24 (figura 7).

Security Group: sg-7197680b

Description	Inbound	Outbound	Tags
-------------	---------	----------	------

Edit

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
SSH	TCP	22	10.0.0.0/24	

Figura 7. Captura de pantalla de la consola de administración AWS.

Una instancia puede tener entre uno y varios SG, y un mismo SG puede ser usado por varias instancias. Esto nos permite crear SG y organizarlos según consideremos conveniente. Por ejemplo, si tenemos una instancia con un servicio WEB (puerto 80 o 443), una base de datos Mysql (puerto 3306) y además tenemos que acceder por SSH (puerto 22), podemos configurar un SG específico para esta instancia o crear un SG por cada servicio de forma individual y asignárselos todos. O podemos crear un SG para un servicio en particular que sabemos que usarán todas nuestras instancias y después un SG para cada una de ellas.

1

Para crear un SG solo hay que ejecutar el comando “create-security-group”, indicando la VPC a la que se asociará, el nombre y la descripción:

```
$ aws ec2 create-security-group --description "Acceso por SSH" --group-name  
AccesoSSH --vpc-id vpc-b2a65dd4  
{  
  "GroupId": "sg-7197680b"  
}
```

2

Acto seguido, es necesario configurar los puertos a los que se permitirá el acceso y el origen con el comando “authorize-security-group-ingress”:

```
$ aws ec2 authorize-security-group-ingress --group-id sg-7197680b --protocol  
tcp --port 22 --cidr 0.0.0.0/0
```

Usando el comando “describe-security-groups” se podrá comprobar la configuración del SG:

```
$ aws ec2 describe-security-groups --group-id sg-7197680b
{
  "SecurityGroups": [
    {
      "IpPermissions": [
        {
          "IpProtocol": "tcp",
          "Ipv6Ranges": [],
          "ToPort": 22,
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "UserIdGroupPairs": [],
          "FromPort": 22,
          "PrefixListIds": []
        }
      ],
      "IpPermissionsEgress": [
        {
          "IpProtocol": "-1",
          "Ipv6Ranges": [],
          "PrefixListIds": [],
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "UserIdGroupPairs": []
        }
      ],
      "VpcId": "vpc-b2a65dd4",
      "GroupName": "AccesoSSH",
      "Description": "Acceso por SSH",
      "GroupId": "sg-7197680b",
      "OwnerId": "700226713106"
    }
  ]
}
```

El último paso al crear una instancia es especificar la llave que se usará para el acceso a la misma. Tanto si ya hemos creado alguna anteriormente como si no lo hemos hecho, en el momento de lanzar la instancia usando la consola nos da la opción de crear una nueva. La llave consta de una llave pública y una llave privada, esta última solo se puede descargar, si se crea usando la consola AWS, o ver si se usa CLI, al crear el par.

Para crear la llave usando CLI, ejecutamos el comando "create-key-pair". Como la llave privada solo se podrá ver al crear el par, redirigimos la salida del comando a un fichero:

```
$ aws ec2 create-key-pair --key-name "LlaveMaestra" --query
'{{KeyMaterial.KeyMaterial}}' --output text > llaveMaestra.pem
```

Este es el contenido del fichero creado:

```
$ cat llaveMaestra.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAhbo/XRP6R0FmgY3PzJasLSuBMEWmWi6JR+AoWikOJV6N7GMw4XoSfEuS1WY35
SpNhqjKv6jbyAoS48aB3XgiTNmiUhCRCD94EbWnBYwk6jvQszzEB+hFwniXbAj32mmV+hLxR4S7R
P3cuH9Uzxs+dKLWv0GiZzXvUvCHR5coK8EAYTNynY7TqfplgcwpYUDigiWkICXoYbhsHISci2zm9
FsRCek1EgNBAoU/nQnZEA6sObEyBHy9rMNHsrh4Fw8He0fLI+Me7SgWcqIK7QIzmdPO01vcFmdtV
TjIfX2PlwRbUkU+plfV/3vAEP6Dr5xemYw0ApjwRR3B+03+HeoNzWQIDAQABAoIBAAGQ6NuDZfha
UJ+e9sB5eiVxP1SbS7JW13tfmGZ0Lmcr3DPttYS1+SZZLzq0npHJ+/0Zf/Nfpw9R1Bn29Nw+JisE
+dhSm417dZiU5W5E0zxmLAG9Y+IihSL03RXwiNxdPNlixomPml16odsXtozKXw9i8nteo9BKJNXI
MHhs0uYU0LOZDeuGgZazFgC3chSiHIjLGe0wU5dEZ/MPH4agJnZs2hN94SiXAE78P3FZga6bCphq
tWjO3+tbxuo/ZiW5KX+3qEnmKTb8sYlHdgRjbQawYIvXDb1LTyz04yJhA+N73a57bwQ0zEtmTkIe
DxRABvNZ+R2Lcdp8dRlt5RmrVQECgYEA6bsYqAGyektamavXCx27jbQwAYORWRRGW/SOVLMBEj7
DTnQMia9sKJ5Hj1fYvLYWkxtBq2qxRyBCRH28MHCCRwc/WN/kNVKfPsa5rvk0k4xZTtQoQi+PwXN
mqANCKGBXyQjKDTs2GbaZ7zKD8yovVHG9EZTjjrOno4iJWID3mECgYEAknf7LmTlawGHpt0gLy0C
odTLA8jKC8/1HCKrvXQ4YRME8yxqYOBgPeEJh7PAvrWyy7uJ+iY/X0fSEchtXoJ3C5/rnF5md3J4
YtYTXNmHirijqe95/nl5ArUv3EEsdDyzGoGe5+csYpggS6d993ZKfZ0F0aUQ+zLUUZnaEyc5bh/kC
gYAB+6Bd7IXDG9iM7TRJ19q7fQfb026FIYC+ooA7XnKSqkNW/WKSyllZokc8xmEunDRc0yJffew
7Gj71rctm7c1tIU8cRen9udG4Cp+QqHSVu98WGB6vUQ/7Kct6yWxLJUYZYow0TvoshawQp3EPIxB
7uutLtuOnVkbZ0FK5+X4gQKBgHuIvI7NxH2zBaGkQV/ou56Ypj3j3R2HYTLGNEEPQ/oEG5mp7XMV
67ZLfi+hWUDAaTPSUjZsiein37Ll9EiIgZWHXsXWnN+z6XubCSu3wM0sm8VZWPBbrcet39cKFR9Y
jzwMIFxhgsGOB71XpYX7A30IRNIAIGTQd2tQLQE0jhipAoGBAL7N2s4fQT2SGMgJmAMRTkvTOjO7
SNoGePdPljRRPlhwU9hWxKeProXblwLbeDldnzBZIAxXqy61l399NwbOaayii8/aO06qYd6IcAeu
aJGwWM+3Uuxy4vtXoylHE6+3ZoydkHGr9kwnTayHQY40PQxtuhFr9UZukh6+Obu1/xX4
-----END RSA PRIVATE KEY-----
```


5

Para poder usar la llave con el comando de Linux “ssh”, es necesario modificar los permisos del fichero para que solo el usuario pueda acceder a la llave:

```
$ chmod 600 llaveMaestra.pem
$ ls -lisa llaveMaestra.pem
259021 4 -rw----- 1 master master 1671 Jan 28 03:11 llaveMaestra.pem
```

Podemos ver las llaves que tenemos usando el comando “describe-key-pairs”:

```
$ aws ec2 describe-key-pairs
{
  "KeyPairs": [
    {
      "KeyName": "LlaveMaestra",
      "KeyFingerprint":
"ed:f2:0a:b2:78:62:09:45:aa:02:da:81:f7:67:27:9e:b1:47:10:94"
    }
  ]
}
```

6

Ahora que se tiene la configuración de red, el grupo de seguridad y la llave, solo falta conocer el ID de la AMI que se utilizará para crear la instancia. En caso de no tener el id, es necesario buscarlo con el comando “describe-images”:

```
$ aws ec2 describe-images --filters "Name=virtualization-type,Values=hvm"
"Name=is-public,Values=true" --query 'Images[*].{ID:ImageId,
Description:Description, Name:Name, CreationDate:CreationDate}.sort_by(@,
&CreationDate)' --output text | grep -v testing | grep -v None | grep
"ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server"
.
.
.
2018-01-26T22:07:39.000Z          Canonical, Ubuntu, 16.04 LTS, amd64 xenial
image build on 2018-01-26      ami-1b791862      ubuntu/images/hvm-ssd/ubuntu-
xenial-16.04-amd64-server-20180126
```

Este comando nos filtra todas las imágenes con el tipo de virtualización hvm, usado para las redes VPC, y que sean públicas (--filters "Name=virtualization-type,Values=hvm" "Name=is-public,Values=true"). Después, con la opción “query” solo se muestra el ID de la imagen, la descripción, el nombre y la fecha de creación, además ordenando todo por esta última, de menor a mayor. Se utiliza también la opción “--output text” para que el formato de salida sea solo texto en lugar de *json*, como se ha visto en el resto de comandos. De esta forma podemos usar las herramientas propias del sistema operativo para manejar flujos de comandos. Por último, usamos el comando “grep” del sistema para mostrar en pantalla las imágenes que no sean de test y no tengan valores nulos, además de que contengan la frase “ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server”. De todas las que se muestran en pantalla, solo se ha puesto la última en el ejemplo, ya que al estar ordenadas esa es la más actual.

7

Nuevamente usando el comando “describe-images” con la opción del ID de la AMI podemos ver sus características:

```
$ aws ec2 describe-images --image-id ami-1b791862
{
  "Images": [
    {
      "ImageType": "machine",
      "ImageLocation": "099720109477/ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20180126",
      "CreationDate": "2018-01-26T22:07:39.000Z",
      "BlockDeviceMappings": [
        {
          "Ebs": {
            "VolumeSize": 8,
            "DeleteOnTermination": true,
            "Encrypted": false,
            "VolumeType": "gp2",
            "SnapshotId": "snap-04c2ac19a43fd1dc0"
          },
          "DeviceName": "/dev/sda1"
        },
        {
          "VirtualName": "ephemeral0",
          "DeviceName": "/dev/sdb"
        },
        {
          "VirtualName": "ephemeral1",
          "DeviceName": "/dev/sdc"
        }
      ],
      "Description": "Canonical, Ubuntu, 16.04 LTS, amd64 xenial image build on 2018-01-26",
      "VirtualizationType": "hvm",
      "Name": "ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20180126",
      "Hypervisor": "xen",
      "Public": true,
      "SriovNetSupport": "simple",
      "ImageId": "ami-1b791862",
      "EnaSupport": true,
      "OwnerId": "099720109477",
      "Architecture": "x86_64",
      "RootDeviceName": "/dev/sda1",
      "State": "available",
      "RootDeviceType": "ebs"
    }
  ]
}
```

8

Ahora que tenemos el ID de la AMI, se puede crear la instancia con el comando “run-instances”:

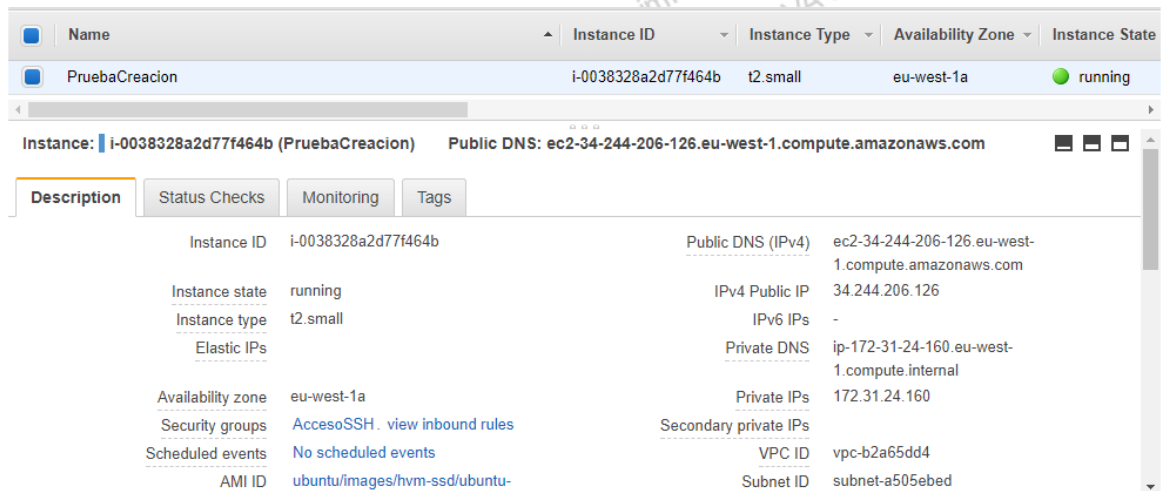
```
$ aws ec2 run-instances --image-id ami-lb791862 --count 1 --instance-type
t2.micro --key-name "LlaveMaestra" --security-group-ids sg-7197680b --subnet-
id subnet-a505ebed --instance-initiated-shutdown-behavior terminate --
associate-public-ip-address --tag-specifications
'ResourceType=instance,Tags=[{Key=Name,Value=PruebaCreacion}]'
{
  "ReservationId": "r-03762d0b62986fae8",
  "Groups": [],
  "OwnerId": "700226713106",
  "Instances": [
    {
      "SubnetId": "subnet-a505ebed",
      "AmiLaunchIndex": 0,
      "ProductCodes": [],
      "NetworkInterfaces": [
        {
          "SubnetId": "subnet-a505ebed",
          "VpcId": "vpc-b2a65dd4",
          "PrivateIpAddresses": [
            {
              "PrivateDnsName": "ip-172-31-24-160.eu-west-
1.compute.internal",
              "Primary": true,
              "PrivateIpAddress": "172.31.24.160"
            }
          ],
          "Attachment": {
            "Status": "attaching",
            "DeviceIndex": 0,
            "DeleteOnTermination": true,
            "AttachTime": "2018-01-28T02:44:03.000Z",
            "AttachmentId": "eni-attach-68dfc313"
          },
          "MacAddress": "06:02:c8:81:d5:f4",
          "NetworkInterfaceId": "eni-3af6ad0a",
          "Ipv6Addresses": [],
          "SourceDestCheck": true,
          "PrivateDnsName": "ip-172-31-24-160.eu-west-
1.compute.internal",
```

```
      "Status": "in-use",
      "Groups": [
        {
          "GroupName": "AccesoSSH",
          "GroupId": "sg-7197680b"
        }
      ],
      "PrivateIpAddress": "172.31.24.160",
      "OwnerId": "700226713106",
      "Description": ""
    }
  ],
  "RootDeviceName": "/dev/sda1",
  "VirtualizationType": "hvm",
  "StateReason": {
    "Code": "pending",
    "Message": "pending"
  },
  "InstanceId": "i-0038328a2d77f464b",
  "ImageId": "ami-lb791862",
  "PublicDnsName": "",
  "Architecture": "x86_64",
  "PrivateIpAddress": "172.31.24.160",
  "Placement": {
    "GroupName": "",
    "AvailabilityZone": "eu-west-1a",
    "Tenancy": "default"
  },
}
```

```
"SecurityGroups": [
  {
    "GroupName": "AccesoSSH",
    "GroupId": "sg-7197680b"
  }
],
"BlockDeviceMappings": [],
"Tags": [
  {
    "Value": "PruebaCreacion",
    "Key": "Name"
  }
],
"SourceDestCheck": true,
"RootDeviceType": "ebs",
"State": {
  "Code": 0,
  "Name": "pending"
},
"EbsOptimized": false,
"VpcId": "vpc-b2a65dd4",
"Hypervisor": "xen",
"InstanceType": "t2.micro",
"LaunchTime": "2018-01-28T02:44:03.000Z",
"PrivateDnsName": "ip-172-31-24-160.eu-west-1.compute.internal",
"Monitoring": {
  "State": "disabled"
},
"StateTransitionReason": "",
"ClientToken": "",
"KeyName": "LlaveMaestra"
}
}
```

9

Se puede ver que el comando tiene algunas opciones adicionales al ID de la AMI, la llave usada, el tipo de instancia, el grupo de seguridad y la subred. La opción “count” se usa para indicarle al comando cuántas instancias se van a crear con esta configuración. La opción “instance-initiated-shutdown-behavior” permite configurar el comportamiento de la instancia al realizar un apagado desde el sistema operativo. Por defecto la instancia solo se apagará, pero se puede modificar para que se destruya, tal y como se indica en esta ocasión. Al crear la instancia, a esta se le asigna una IP privada con la que se comunicará con el resto de los servicios e instancias de AWS; sin embargo, para que pueda ser accesible desde el exterior es necesario asociarle una IP pública, y esto se hace con la opción “associate-public-ip-address”. Para finalizar, se ha añadido la opción “tag-specifications” para poder identificar la instancia de forma más fácil.



Name	Instance ID	Instance Type	Availability Zone	Instance State
PruebaCreacion	i-0038328a2d77f464b	t2.small	eu-west-1a	running

Instance: **i-0038328a2d77f464b (PruebaCreacion)** Public DNS: **ec2-34-244-206-126.eu-west-1.compute.amazonaws.com**

Description		Status Checks	Monitoring	Tags
Instance ID	i-0038328a2d77f464b			
Instance state	running			
Instance type	t2.small			
Elastic IPs				
Availability zone	eu-west-1a			
Security groups	AccesoSSH , view inbound rules			
Scheduled events	No scheduled events			
AMI ID	ubuntu/images/hvm-ssd/ubuntu-			
Public DNS (IPv4)	ec2-34-244-206-126.eu-west-1.compute.amazonaws.com			
IPv4 Public IP	34.244.206.126			
IPv6 IPs	-			
Private DNS	ip-172-31-24-160.eu-west-1.compute.internal			
Private IPs	172.31.24.160			
Secondary private IPs				
VPC ID	vpc-b2a65dd4			
Subnet ID	subnet-a505ebed			

Figura 8. Captura de pantalla de la consola de administración AWS.

Podemos ver en la figura 8 cómo se muestra la instancia ya creada en la consola de AWS.

10

Para acceder a esta instancia, tanto para comprobar que está funcionando como para trabajar en ella, tendremos que conocer la IP pública, o su IP privada si estamos accediendo desde otra instancia en su VPC. Como la IP pública se crea después de iniciar la instancia, en la descripción inicial no aparece, por lo que se tendrá que usar el comando “describe-instances” para obtener esta información. En este caso, se usa la opción “query” para mostrar solo este dato:

```
$ aws ec2 describe-instances --instance-id i-0038328a2d77f464b --query
'Reservations[*].Instances[*].{IPPublica:PublicIpAddress}' --output text
34.243.94.248
```

Ahora que ya se conoce la IP pública, se puede usar el comando del sistema “ssh” para acceder a la instancia con la llave que se ha creado (hay que recordar que esto se puede hacer gracias a que se ha configurado el SG para que se pueda acceder por el puerto 22).

```
$ ssh ubuntu@34.243.94.248 -i llaveMaestra.pem
The authenticity of host '34.243.94.248 (34.243.94.248)' can't be established.
ECDSA key fingerprint is SHA256:2dnppOBD40vgm0+8MuuRDgQjgazMkKp+ecC8SJm6Qxk.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '34.243.94.248' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1049-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-31-241:~$
```

Si queremos detener la instancia, usamos el comando “stop-instances”:

```
$ aws ec2 stop-instances --instance-id i-0038328a2d77f464b
{
  "StoppingInstances": [
    {
      "InstanceId": "i-0038328a2d77f464b",
      "PreviousState": {
        "Name": "running",
        "Code": 16
      },
      "CurrentState": {
        "Name": "stopping",
        "Code": 64
      }
    }
  ]
}
```

En este caso, aunque la instancia tenga la opción de destruirse al apagarse, esta opción solo es válida si se apaga desde el sistema operativo (por ejemplo, usando el comando de sistema “halt”); por lo tanto, la instancia solo se detendrá y podremos iniciarla nuevamente cuando queramos. Si la intención es destruir la instancia, usaremos el comando “terminate-instances”.

```
$ aws ec2 terminate-instances --instance-id i-0038328a2d77f464b
{
  "TerminatingInstances": [
    {
      "InstanceId": "i-0038328a2d77f464b",
      "PreviousState": {
        "Name": "stopping",
        "Code": 64
      },
      "CurrentState": {
        "Name": "shutting-down",
        "Code": 32
      }
    }
  ]
}
```

Con esto la instancia se destruirá y, si se necesita nuevamente, será necesario crear una nueva.

5.1. S3 (Simple Cloud Storage Service)*

Adicionalmente a los discos virtuales que podemos crear como parte de las instancias, AWS ofrece otros métodos para almacenar información.

* **AVISO IMPORTANTE:** una vez terminada la unidad, el caso práctico y el ejercicio práctico final, se recomienda eliminar el bucket de S3.

Aunque en este apartado nos centraremos en S3, otros dos servicios de almacenamiento que tenemos a nuestra disposición son EFS (Amazon Elastic File System) y Glacier.

EFS

EFS es un sistema de archivos fácil de configurar e integrar con las instancias de EC2, ya que se comporta como una unidad de red; en otras palabras, es el equivalente a un NAS dentro de la nube. Otra de sus características importantes es su espacio casi ilimitado y, además, su coste depende del espacio que estemos utilizando, a diferencia de un disco virtual cuyo coste dependerá del tamaño que hemos reservado. Sin embargo, el precio por GB es casi tres veces superior que el de un disco SSD de uso general. Aun así, en casos en los que no conocemos cuánta capacidad necesitamos, o si necesitamos compartir la información entre varias instancias, EFS puede ser una solución.

Glacier

Glacier está pensado para almacenar *backups* o datos de acceso poco frecuentes. Su coste es muy inferior al del resto de los sistemas de almacenamiento; sin embargo, para acceder a los datos podemos tardar, según el servicio contratado, entre algunos minutos y varias horas.

S3

S3 permite almacenar datos a bajo coste y, aunque presenta algunas limitaciones a la hora de acceder a los datos desde las instancias, la capacidad que nos brinda para poder acceder a los datos desde la consola de AWS o utilizando enlaces web hace de S3 un servicio de almacenamiento muy útil. Además de esto, muchas aplicaciones permiten utilizar accesos a S3 directamente, lo que nos permite usar los datos sin tener que descargarlos primero al disco virtual de la instancia.

Para utilizar S3, primero hay que crear al menos un *bucket* (cubo), en el que se guardarán los datos que queramos. Hay que tener en cuenta que el nombre del *bucket* es único en toda la red de AWS.

Utilizar las opciones básicas de S3 desde la línea de comandos es muy simple. Hay que utilizar el servicio "s3" y los comandos básicos son los siguientes.

Crear un bucket (mb)

```
$ aws --profile S3 s3 mb s3://masters3bucket
make_bucket: masters3bucket
```

Mostrar el contenido (ls)

```
$ aws --profile S3 s3 ls
2018-01-28 16:45:52 masters3bucket
```

Copiar ficheros (cp)

En este caso, al copiar los ficheros, si el directorio de destino dentro de S3 no existe, se crea:

```
$ aws --profile S3 s3 cp fichero_prueba.txt
s3://masters3bucket/pruebacopia/fichero_prueba.txt
upload: ./fichero_prueba.txt to
s3://masters3bucket/pruebacopia/fichero_prueba.txt

$ aws --profile S3 s3 ls s3://masters3bucket/pruebacopia/
2018-01-28 16:50:58          69 fichero_prueba.txt
```

Borrar ficheros (rm)

```
$ aws --profile S3 s3 rm s3://masters3bucket/pruebacopia/fichero_prueba.txt
delete: s3://masters3bucket/pruebacopia/fichero_prueba.txt
```

Borrar bucket (rb)

```
$ aws --profile S3 s3 rb s3://masters3bucket
remove_bucket: masters3bucket
```

En la figura 9 se puede ver el fichero antes de borrarlo en la consola de AWS.

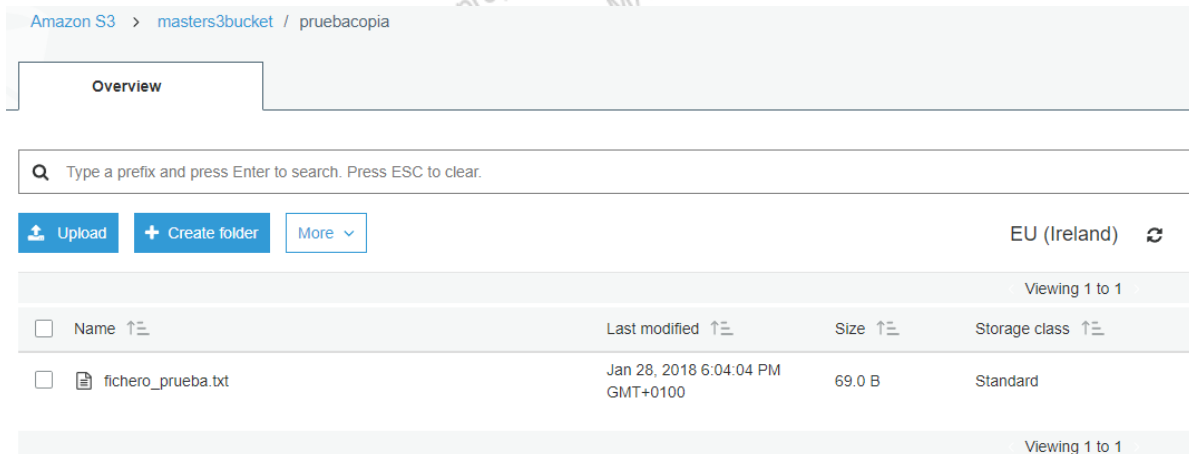


Figura 9. Captura de pantalla de la consola de administración AWS.

También podemos habilitar el acceso desde la web a los ficheros al copiarlos usando la opción “--acl public-read”.

```
$ aws --profile S3 s3 cp fichero_prueba.txt
s3://masters3bucket/pruebacopia/fichero_prueba.txt --acl public-read
upload: ./fichero_prueba.txt to
s3://masters3bucket/pruebacopia/fichero_prueba.txt
```

Ahora, a este fichero se puede acceder desde la URL: https://s3-eu-west-1.amazonaws.com/masters3bucket/pruebacopia/fichero_prueba.txt

Los ficheros compartidos de esta forma usan la nomenclatura: <https://s3-<region>.amazonaws.com/<bucket>/<ruta y fichero>>.

De todas formas, para una mejor configuración de S3 y el uso de opciones más complejas (versionado, *logs*, permisos, etc.), es preferible usar la consola de AWS.

5.2. RDS (Relational Database Service)*

Con un funcionamiento muy similar a EC2, tenemos también RDS. Desde esta herramienta podemos crear instancias especiales con motores de BB. DD. preinstalados y configurados para poder trabajar con ellos desde el primer momento. Los distintos motores son: Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle y Microsoft SQL Server.

* **AVISO IMPORTANTE:** una vez terminada la unidad, el caso práctico y la práctica final, se recomienda parar la instancia en RDS.



Las instancias de RDS están optimizadas para mejorar el uso con las BB. DD., y al igual que en EC2 tenemos distintos tipos:

- Instancias estándar: M3, M4 y T2.
- Optimizadas para uso de memoria: R3 y R4.

Y cada una de ellas con distintos tamaños, desde micro a 16xlarge. La nomenclatura usada en estas instancias es db.<tipo>.<tamaño>, por ejemplo db.m4.2xlarge.

Esta herramienta, además de permitir crear instancias con un motor de BB. DD. preinstalado, realizar cambios de tipo de instancia y actualizaciones del motor, como parches o mejoras de versión, también permite configurar sistemas de *backup*, aumentar la capacidad de las BB. DD. e incluso crear réplicas de lectura tanto en la misma región y zona como en otra distinta, lo cual nos permite tener un sistema de alta disponibilidad y mejorar el acceso a los datos si los clientes de las BB. DD. se encuentran en otras zonas geográficas.

“create-db-instance”

Para crear una instancia de RDS con la línea de comandos, usamos el comando “create-db-instance” con el servicio “rds”. Las opciones obligatorias son: “db-instance-identifier”, que será el que usemos para identificar esta instancia; “db-instance-class”, con el tipo de la instancia; y “engine”, con el motor de la BB. DD.

El resto de parámetros son opcionales.

Además, según el motor pueden tener distintos valores, por ejemplo: “allocated-storage” en el motor de Amazon Aurora no se aplica, en MySQL para un disco SSD tiene un valor entre 20 y 16384, para SQLServer un valor entre 200 y 16384.

Nota: Del mismo modo que hemos configurado el usuario para S3, hemos de crear el RDS (aws configure --profile RDS).



Proceso: “create-db-instance”

```

$ aws --profile RDS rds create-db-instance --db-instance-identifier master-
rds-mysql --allocated-storage 20 --db-instance-class db.t2.small --engine mysql
--master-username mastermyuser --master-user-password mastermypass
{
  "DBInstance": {
    "LicenseModel": "general-public-license",
    "DBInstanceArn": "arn:aws:rds:eu-west-1:700226713106:db:master-rds-
mysql",
    "CopyTagsToSnapshot": false,
    "DBInstanceIdentifier": "master-rds-mysql",
    "PubliclyAccessible": true,
    "IAMDatabaseAuthenticationEnabled": false,
    "DBSecurityGroups": [],
    "BackupRetentionPeriod": 1,
    "DbiResourceId": "db-JUBFOF3EU2TEFMHTFMVI6UI5CY",
    "MasterUsername": "mastermyuser",
    "VpcSecurityGroups": [
      {
        "Status": "active",
        "VpcSecurityGroupId": "sg-3358ba49"
      }
    ],
    "PreferredBackupWindow": "03:58-04:28",
    "DBSubnetGroup": {
      "SubnetGroupStatus": "Complete",
      "Subnets": [
        {
          "SubnetAvailabilityZone": {
            "Name": "eu-west-1a"
          },
          "SubnetStatus": "Active",
          "SubnetIdentifier": "subnet-d47d949c"
        },
        {
          "SubnetAvailabilityZone": {
            "Name": "eu-west-1a"
          },
          "SubnetStatus": "Active",
          "SubnetIdentifier": "subnet-a505ebed"
        }
      ],
      "SubnetAvailabilityZone": {
        "Name": "eu-west-1b"
      },
      "SubnetStatus": "Active",
      "SubnetIdentifier": "subnet-37d6f16c"
    },
    {
      "SubnetAvailabilityZone": {
        "Name": "eu-west-1c"
      },
      "SubnetStatus": "Active",
      "SubnetIdentifier": "subnet-9f6c76f8"
    }
  ],
  "DBSubnetGroupDescription": "default",
  "DBSubnetGroupName": "default",
  "VpcId": "vpc-b2a65dd4"
},
  "DBInstanceClass": "db.t2.small",
  "PerformanceInsightsEnabled": false,
  "MonitoringInterval": 0,
  "CACertificateIdentifier": "rds-ca-2015",
  "AllocatedStorage": 20,
  "StorageEncrypted": false,
  "ReadReplicaDBInstanceIdentifiers": [],
  "DbInstancePort": 0,
  "Engine": "mysql",

```

```

    "DBParameterGroups": [
      {
        "DBParameterGroupName": "default.mysql5.6",
        "ParameterApplyStatus": "in-sync"
      }
    ],
    "StorageType": "standard",
    "OptionGroupMemberships": [
      {
        "Status": "in-sync",
        "OptionGroupName": "default:mysql-5-6"
      }
    ],
    "DBInstanceStatus": "creating",
    "AutoMinorVersionUpgrade": true,
    "PreferredMaintenanceWindow": "wed:01:06-wed:01:36",
    "DomainMemberships": [],
    "MultiAZ": false,
    "PendingModifiedValues": {
      "MasterUserPassword": "****"
    },
    "EngineVersion": "5.6.37"
  }
}

```

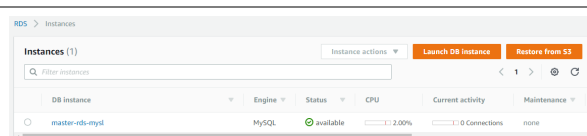


Figura 10. Captura de pantalla de la consola de administración AWS.

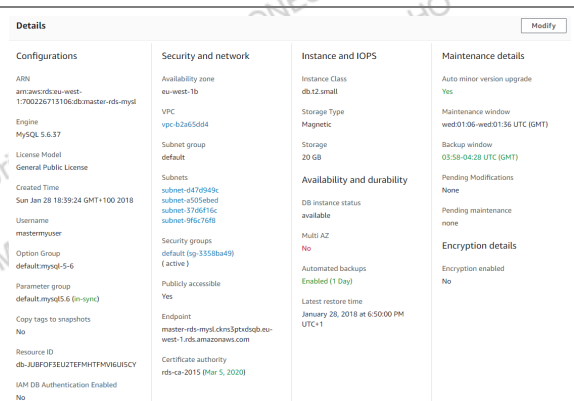


Figura 11. Captura de pantalla de la consola de administración AWS.

Podemos ver el resultado en las figuras 10 y 11: los valores por defecto que ha usado el comando. El acceso a la instancia, por defecto, se configura en todas las subredes y en todas las zonas de la región para permitir alta disponibilidad. También se configura un *backup* diario y se habilita un acceso público.

"describe-db-instances"

Para ver este acceso, desde CLI usamos el comando "describe-db-instances".

```

$ aws --profile RDS rds describe-db-instances --db-instance-identifier master-
rds-mysql --query 'DBInstances[*].Endpoint'
[
  {
    "HostedZoneId": "Z29XKXDKYMONMX",
    "Port": 3306,
    "Address": "master-rds-mysql.ckns3ptxdsqb.eu-west-1.rds.amazonaws.com"
  }
]

```

"Endpoint"

En el apartado "Endpoint" se puede encontrar la dirección de acceso y el puerto, en este caso:

"master-rds-mysql.ckns3ptxdsqb.eu-west-1.rds.amazonaws.com" y 3306.

"delete-db-instance"

Para destruir la instancia, solo hay que utilizar el comando "delete-db-instance"; sin embargo, es necesario crear una copia de seguridad antes del borrado e identificarla con la opción "final-db-snapshot-identifier" o añadir la opción "skip-final-snapshot".

```
$ aws --profile RDS rds delete-db-instance --db-instance-identifier master-rds-mysql --skip-final-snapshot
```

VI. Herramientas de configuración y utilidades

6.1. Elastic Beanstalk*

* **AVISO IMPORTANTE:** una vez terminada la unidad, el caso práctico y la práctica final, se recomienda **parar** todos los servicios levantados desde Elastic Beanstalk (load balancing, aplicación creada en Elastic Beanstalk y grupos de auto-escalado).

Aunque AWS es IaaS, esta herramienta se comporta como PaaS, ya que es una herramienta de configuración que permite integrar varios servicios de AWS con el fin de desplegar aplicaciones o servicios web.

Actualmente, sin usar Elastic Beanstalk, si queremos desplegar una aplicación web en AWS, tendremos que crear una instancia, instalar y configurar las librerías y programas necesarios para su funcionamiento y, por último, copiar nuestro código en ella. Esto en principio no es muy complejo, pero si nuestra aplicación web tiene un gran número de visitas, posiblemente sea necesario crear un grupo de escalado que nos permita incrementar o reducir el número de instancias de forma dinámica. Para esto, primero tendremos que crear una AMI de nuestra instancia y después configurar el grupo:

1

Para la AMI se usará la instancia creada anteriormente:

```
$ aws ec2 create-image --instance-id i-0038328a2d77f464b --name "masterAMI"
{
  "ImageId": "ami-985737e1"
}
```

Nota: Si el siguiente comando da fallo o no reconoce el id, podemos volver a ejecutar comando anterior.

```
$ aws ec2 run-instances --image-id ami-1b791862 --count 1 --instance-type
t2.small --key-name "LlaveMaestra" --security-group-ids sg-7197680b --subnet-
id subnet-a505ebed --instance-initiated-shutdown-behavior terminate --
associate-public-ip-address --tag-specifications
'ResourceType=instance,Tags=[{Key=Name,Value=PruebaCreacion}]'
```

Ahora podemos ver la AMI con el comando "describe-images" y la opción "owners self":

```
$ aws ec2 describe-images --owners self
{
  "Images": [
    {
      "ImageLocation": "700226713106/masterAMI",
      "OwnerId": "700226713106",
      "RootDeviceType": "ebs",
      "RootDeviceName": "/dev/sda1",
      "EnaSupport": true,
      "Hypervisor": "xen",
      "Architecture": "x86_64",
      "ImageType": "machine",
      "Public": false,
      "CreationDate": "2018-01-28T19:03:29.000Z",
      "VirtualizationType": "hvm",
      "SriovNetSupport": "simple",
      "ImageId": "ami-985737e1",
      "Name": "masterAMI",
      "State": "available",
      "BlockDeviceMappings": [
        {
          "Ebs": {
            "SnapshotId": "snap-09f961a7bfd4777f8",
            "VolumeSize": 8,
            "VolumeType": "gp2",
            "Encrypted": false,
            "DeleteOnTermination": true
          },
          "DeviceName": "/dev/sda1"
        },
        {
          "VirtualName": "ephemeral0",
          "DeviceName": "/dev/sdb"
        },
        {
          "VirtualName": "ephemeral1",
          "DeviceName": "/dev/sdc"
        }
      ]
    }
  ]
}
```

3

Ahora, para crear el grupo de autoescalado, en la consola dentro del servicio EC2 en el apartado “AUTO SCALING” – “Auto Scaling Groups” pulsamos en “Create Auto Scaling group”. En este punto nos saldrá un aviso indicando que primero se creará la configuración y después el grupo de escalado.

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

Create Launch Configuration

Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

My AMIs

AWS Marketplace

Community AMIs

Search my AMIs

masterAMI - ami-985737e1

Root device type: ebs Virtualization type: hvm Owner: 700226713106 64-bit

Select

Figura 12. Captura de pantalla de la consola de administración AWS.

4

En la configuración, lo primero es escoger la AMI. La encontraremos dentro de “My AMIs” (figura 12).

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

Create Launch Configuration

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate

Figura 13. Captura de pantalla de la consola de administración AWS.

Después, seleccionaremos el tipo de instancia (figura 13).

1

En los pasos siguientes solo hay que poner nombre a la configuración, configurar el espacio de almacenamiento, añadir el grupo de seguridad, comprobar que la configuración es correcta, añadir la llave y aceptar.

1. Configure Auto Scaling group details

2. Configure scaling policies

3. Configure Notifications

4. Configure Tags

5. Review

Create Auto Scaling Group

Cancel and Exit

Launch Configuration ⓘ

masterLaunchConfig

Group name ⓘ

masterASG

Group size ⓘ

Start with 2 instances

Network ⓘ

vpc-b2a65dd4 (172.31.0.0/16) (default)

Create new VPC

Subnet ⓘ

subnet-a505ebed(172.31.16.0/20) | Default in eu-west-1a x

subnet-37d6f16c(172.31.32.0/20) | Default in eu-west-1b x

subnet-d47d949c(172.31.221.0/24) | prueba | eu-west-1a

subnet-9f6c76f8(172.31.0.0/20) | Default in eu-west-1c

Create new subnet

a public IP address. ⓘ

Figura 14. Captura de pantalla de la consola de administración AWS.

2

Ahora que la configuración básica del grupo de autoescalado está creada, se creará el grupo propiamente dicho. Lo primero es nombrar el grupo y añadir las distintas subredes en las que se pueden crear las instancias y las instancias iniciales (figura 14).

● Use scaling policies to adjust the capacity of this group

Scale between 2 and 4 instances. These will be the minimum and maximum size of your group.

Scale Group Size

✕

Name: Scale Group Size

Metric type: Average CPU Utilization ▼

Target value: 60

Instances need: 300 seconds to warm up after scaling

Disable scale-in: ☐

Scale the Auto Scaling group using step or simple scaling policies ⓘ

Figura 15. Captura de pantalla de la consola de administración AWS.

3

Ahora le indicamos al sistema que mantenga el grupo entre 2 y 4 instancias y con una media del 60 % de uso de CPU (figura 15).

<input type="checkbox"/>	Name	Launch Configuration /	Instances	Desired	Min	Max	Availability Zones
<input checked="" type="checkbox"/>	masterASG	masterLaunchConfig	0 ⓘ	2	2	4	eu-west-1b, eu-west-1a

Figura 16. Captura de pantalla de la consola de administración AWS.

4

Revisamos que la configuración es la correcta y creamos el grupo. En la figura 16 se puede ver el grupo creado y en la figura 17 las instancias generadas por el grupo de autoescalado.

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>		i-064a5cb58f5cbfc98	t2.micro	eu-west-1a	● running	⌚ Initializing
<input type="checkbox"/>		i-09de23cadbae5eb8c	t2.micro	eu-west-1b	● running	⌚ Initializing

Figura 17. Captura de pantalla de la consola de administración AWS.

Además de esto, tendremos que crear un balanceador de carga para que nuestras instancias tengan un solo punto de acceso.

5

En la consola de EC2, en el menú "LOAD BALANCING" – "Load Balancers" podemos crear nuestro balanceador de carga pulsando en "Create Load Balancer". Primero nos permite escoger el tipo, entre http/https, TCP y de antigua generación. En este caso se escogerá la primera opción.

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Security Groups 4. Configure Routing 5. Register Targets 6. Review

Step 1: Configure Load Balancer

Name ⓘ masterLoadBalancer

Scheme ⓘ ☒ internet-facing ☐ internal

IP address type ⓘ ipv4

Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
HTTP	80

Add listener

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

VPC ⓘ vpc-b2a65dd4 (172.31.0.0/16) (default)

<input type="checkbox"/>	Availability Zone	Subnet ID	Subnet IPv4 CIDR	Name
<input type="checkbox"/>	eu-west-1a	Select a subnet...		
<input checked="" type="checkbox"/>	eu-west-1b	subnet-37d6f16c	172.31.32.0/20	
<input checked="" type="checkbox"/>	eu-west-1c	subnet-9f6c76f8	172.31.0.0/20	

Figura 18. Captura de pantalla de la consola de administración AWS.

1

Como se puede ver en la figura 18, se puede configurar el balanceador indicando el nombre, los puertos por los que escucha y las zonas de disponibilidad.

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Security Groups 4. Configure Routing 5. Register Targets 6. Review

Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify, and performs health checks on the targets using these health check settings. Note that each target group can be associated with only one load balancer.

Target group

Target group ⓘ New target group

Name ⓘ masterLBTarget

Protocol ⓘ HTTP

Port ⓘ 80

Target type ⓘ instance

Health checks

Protocol ⓘ HTTP

Path ⓘ /

► Advanced health check settings

Figura 19. Captura de pantalla de la consola de administración AWS.

2

En las siguientes pestañas configuramos los grupos de seguridad y después el tipo de enrutamiento que usará el balanceador (figura 19).

Registered targets

To deregister instances, select one or more registered instances and then click Remove.

Remove

<input type="checkbox"/>	Instance	Name	Port	State	Security groups	Zone
<input type="checkbox"/>	i-064a5cb58f5cbfc98		80	running	AccesoSSH	eu-west-1a
<input type="checkbox"/>	i-09de23cadbae5eb8c		80	running	AccesoSSH	eu-west-1b

Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

Add to registered on port 80

Search Instances

<input type="checkbox"/>	Instance	Name	State	Security	Zone	Subnet ID	Subnet CIDR
<input checked="" type="checkbox"/>	i-064a5cb58f...		running	AccesoSSH	eu-west-1a	subnet-a505ebed	172.31.16.0/20
<input checked="" type="checkbox"/>	i-09de23cadb...		running	AccesoSSH	eu-west-1b	subnet-37d6f16c	172.31.32.0/20

Figura 20. Captura de pantalla de la consola de administración AWS.

3

En la última pestaña de configuración, escogemos los objetivos del balanceador (figura 20).

<input checked="" type="checkbox"/>	Name	DNS name	State	VPC ID
<input checked="" type="checkbox"/>	masterLoadBalancer	masterLoadBalancer-1614955034....	active	vpc-b2a65dd4

Figura 21. Captura de pantalla de la consola de administración AWS.

Aunque el balanceador está funcionando y redirigiendo el tráfico a las instancias seleccionadas, si el grupo de autoescalado crea nuevas instancias, estas no estarán en el balanceador. Para resolverlo, nuevamente en "AUTO SCALING" – "Auto Scaling Groups", seleccionamos el grupo y en la pestaña "Details" pulsamos "Edit". En "Target Groups" añadimos la configuración de objetivos previamente creada (se modificará también el número de instancias mínimas para comprobar el funcionamiento) (figura 22). Al cabo de unos minutos, se puede comprobar que se ha creado una nueva instancia (figura 23) y en "LOAD BALANCING" – "Target Groups", en la pestaña "Targets", se pueden ver las tres instancias (figura 24).

6

Para terminar, revisamos la configuración y creamos el balanceador (figura 21).

Auto Scaling Group: masterASG

Details Activity History Scaling Policies Instances Monitoring Notifications Tags Scheduled Actions

Launch Configuration masterLaunchConfig

Launch Template

Launch Template Version

Load Balancers

Target Groups masterLBTarget x

Desired 3 **Availability Zone(s)** eu-west-1a, eu-west-1b

Min 3 **Subnet(s)** subnet-a505ebed (Default in eu-west-1)

Figura 22. Captura de pantalla de la consola de administración AWS.

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>		i-064a5cb58f5cbfc98	t2.micro	eu-west-1a	● running	✓ 2/2 checks ...
<input type="checkbox"/>		i-06fabe44ad7dcb8f0	t2.micro	eu-west-1b	● running	✓ 2/2 checks ...
<input type="checkbox"/>		i-0dfec609624504920	t2.micro	eu-west-1b	● running	✓ 2/2 checks ...

Figura 23. Captura de pantalla de la consola de administración AWS.

Con esto ya tendríamos nuestra aplicación web funcionando y, en caso de necesidad, se crearían más instancias con nuestra AMI para hacer frente a picos de trabajo. Sin embargo, recordemos que las AMI son imágenes estáticas. ¿Qué ocurre si tenemos que cambiar el código de nuestra aplicación web? En este caso, tendremos que crear una nueva AMI con la mejora y reconfigurarla todo de nuevo. Otra opción, para no tener que repetir este proceso en cada cambio de código, es configurar la instancia para que al iniciar copie el código que dejaremos en S3 o en un repositorio de código; de esta forma siempre tendremos nuestra aplicación actualizada.

Ahora que conocemos la complejidad de desarrollar y administrar una aplicación o servicio web, podemos comprobar las ventajas que nos brinda esta herramienta. Desde Elastic Beanstalk, primero tendremos que crear los entornos de nuestra aplicación, indicar el tipo de instancias, el tipo de aplicación (entre los tipos tenemos Java, .NET, PHP, Node.js, Python, Ruby, Go y Docker), indicar algunos datos para crear el grupo de autoescalado y el balanceador de carga y preparar un fichero con nuestro código. Todo esto nos permitirá desplegar nuestra aplicación, monitorizar su estado y administrarla fácilmente.

Desde la página del servicio de Elastic Beanstalk se pulsa en “Create New Application”, se escribe el nombre y la descripción y se crea la aplicación. Ahora habrá que crear los entornos de trabajo: para ello se pulsa en “Actions” – “Create environment”. Para el caso de ejemplo, escogemos “Web server environment”.

The screenshot shows the AWS Elastic Beanstalk console. At the top, there is a search bar and a table with columns: Name, Port, Protocol, and Target type. The table contains one entry: 'masterLBTarget' with Port 80, Protocol HTTP, and Target type instance. Below this, there is a section for 'Target group: masterLBTarget' with tabs for Description, Targets, Health checks, Monitoring, and Tags. The 'Targets' tab is selected, showing a description: 'The load balancer starts routing requests to a newly registered target as soon as the registration target passes the initial health checks. If demand on your targets increases, you can register additional targets. If demand decreases, you can deregister targets.' There is an 'Edit' button. Below the description is a section for 'Registered targets' with a table showing three targets:

Instance ID	Name	Port	Availability Zone
i-064a5cb58f5cbfc98		80	eu-west-1a
i-06fabe44ad7dcb8f0		80	eu-west-1b
i-0dfec609624504920		80	eu-west-1b

Figura 24. Captura de pantalla de la consola de administración AWS.

En la configuración hay que dar un nombre al entorno (si solo hay uno, se puede dejar el nombre por defecto; en caso contrario, es mejor un nombre descriptivo), la descripción y el dominio. Si este último no se especifica, AWS generará el suyo propio. Además, este tiene que ser único dentro de la región.

Más abajo, como se muestra en la figura 25, es donde se configura el tipo de plataforma (en este caso PHP) y, si se añade el código, que puede ser un ejemplo de AWS, una versión de código ya subida o un nuevo fichero comprimido con el código de la aplicación. Más abajo tenemos la opción “Configure more options”. En esta nueva ventana, seleccionando la opción “High availability” o “Custom configuration” podremos configurar los grupos de autoescalado (en la parte de “Capacity”) y el balanceador, entre otras cosas (figura 26).

La creación del entorno tardará unos minutos ya que está creando las instancias, cargando el código, creando los grupos de autoescalado y el balanceador de cargas.

Una vez terminado, se podrán ver los entornos en la aplicación (figura 27). Dentro del entorno tenemos un menú adicional que muestra los eventos y el estado del entorno y que además permite, dentro del menú “Configuration”, volver a acceder al panel que se ve en la figura 26.

Base configuration

Tier Web Server ([Choose tier](#))

Platform ☒ Preconfigured platform
Platforms published and maintained by AWS Elastic Beanstalk.

PHP

☐ Custom platform NEW
Platforms created and owned by you. [Learn more](#)

-- Choose a custom platform --

Application code ☒ Sample application
Get started right away with sample code.

☐ Existing version
Application versions that you have uploaded for masterWebApp.

-- Choose a version --

☐ Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

[Upload](#) ZIP or WAR

[Cancel](#) [Configure more options](#) [Create environment](#)

Figura 25. Captura de pantalla de la consola de administración AWS.

Elastic Beanstalk masterWebApp

[All Applications](#) > masterWebApp

Environments

Application versions

Saved configurations

Masterwebapp-env
Environment tier: Web Server
Platform: 64bit Amazon Linux 2017.09 v2.6.4 running PHP 7.1
Running versions: Sample Application
Last modified: 2018-01-28 21:31:22 UTC+0...
URL: Masterwebapp-env.zme9t3kwt3.eu-wes...

Figura 26. Captura de pantalla de la consola de administración AWS.

Configuration presets ☐ Low cost (*Free Tier eligible*) ☒ High availability ☐ Custom configuration

Platform 64bit Amazon Linux 2017.09 v2.6.4 running PHP 7.1 [Change platform configuration](#)

Software Rotate logs: disabled (default) Log streaming: disabled (default) Environment properties: 0 Modify	Instances EC2 instance type: t1.micro EC2 image ID: ami-ac9a03d5 Root volume type: container default Root volume size (GB): container default Root volume IOPS: container default Modify	Capacity Environment type: load balancing, auto scaling Availability Zones: Any Instances: 1-4 Modify
Load balancer Port: HTTP on port 80 Secure port: disabled Cross-zone load balancing: enabled Connection draining: enabled Modify	Rolling updates and deployments Deployment policy: All at once Rolling updates: disabled Health check: enabled Modify	Security Service role: aws-elasticbeanstalk-service-role Virtual machine key pair: -- Virtual machine instance profile: aws-elasticbeanstalk-ec2-role Modify
Monitoring	Notifications	Network

Figura 27. Captura de pantalla de la consola de administración AWS.

9

En la figura 28 se puede ver la instancia creada con el nombre del entorno y, en las figuras 29 y 30, el balanceador y el grupo de autoescalado, respectivamente.

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input type="checkbox"/>	MasterwebappEnv	i-067d9ee94c331dfae	t1.micro	eu-west-1b	running	2/2 checks ...

Figura 28. Captura de pantalla de la consola de administración AWS.

<input checked="" type="checkbox"/>	Name	DNS name	State	VPC ID
<input checked="" type="checkbox"/>	awseb-e-b-AWSEBLoa-1VO8LMBZUJ9TP	awseb-e-b-AWSEBLoa-1VO8LMBZ...		vpc-b2a65dd4

Load balancer:	awseb-e-b-AWSEBLoa-1VO8LMBZUJ9TP	
----------------	----------------------------------	--

Figura 29. Captura de pantalla de la consola de administración AWS.

Otro detalle importante es que AWS no carga tarifas adicionales por el uso de Elastic Beanstalk, solo por los recursos de AWS que se necesiten para almacenar y ejecutar las aplicaciones.

<input checked="" type="checkbox"/>	Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones
<input checked="" type="checkbox"/>	awseb-e-b7wg...	awseb-e-b7wgggqhb...	1	1	1	4	eu-west-1b, eu-west-1c, eu-...

Auto Scaling Group:	awseb-e-b7wgggqhb-stack-AWSEBAutoScalingGroup-WJJOEAXEZJS4	
---------------------	--	--

Figura 30. Captura de pantalla de la consola de administración AWS.

6.2. SQS (Simple Queue Service)*

* **AVISO IMPORTANTE:** una vez terminada la unidad, el caso práctico y práctica final se recomienda **eliminar** cualquier servicio y aplicación creada en SQS.

Esta herramienta nos permite crear y gestionar colas de mensajes que pueden usar otras aplicaciones y servicios. Los dos tipos de colas que se pueden crear son:

Estándar

Este tipo de colas ofrece una capacidad de procesamiento máxima y permite realizar un número de transacciones casi ilimitado por segundo. Estas colas, además, garantizan que cada mensaje se entrega al menos una vez. Sin embargo, no garantizan completamente que los mensajes se entreguen en el orden de llegada, aunque los algoritmos que utilizan intentan minimizar este comportamiento.

FIFO (first-in, first-out)

La característica más importante que presentan estas colas es la de garantizar que los mensajes se entregan en el orden de llegada (primero en entrar, primero en salir). Cada mensaje se envía una vez y permanece disponible hasta que es procesado y eliminado. La desventaja con respecto a las colas estándar se encuentra en su capacidad de procesamiento, pues solo puede realizar 300 transacciones por segundo.

Con SQS podemos, por ejemplo, crear aplicaciones basadas en mensajes, crear integraciones con otros servicios de AWS para que las aplicaciones sean más flexibles y fiables o crear colas de trabajo en las que cada mensaje es como una tarea que ha de completar un proceso, de tal forma que uno o varios equipos lean las tareas de la cola de mensajes y las procesen.

Un ejemplo del uso de SQS lo podemos encontrar en el FAQ oficial de SQS¹.

Para un sitio web de transcodificación de vídeos:

Los usuarios finales envían vídeos para que se transcodifiquen en el sitio web.

Los vídeos se almacenan en Amazon S3 y se coloca un mensaje de solicitud en una cola de Amazon SQS, con un puntero dirigido al vídeo y al formato de vídeo de destino en el mensaje.

El motor de transcodificación, que se ejecuta en un grupo de instancias de Amazon EC2, lee el mensaje de solicitud de la cola de entrada, recupera el vídeo de Amazon S3 utilizando el puntero y transcodifica el vídeo al formato de destino.

El vídeo convertido se vuelve a colocar en Amazon S3 y se coloca otro mensaje de respuesta en otra cola de salida de Amazon SQS con un puntero dirigido al vídeo convertido.

Al mismo tiempo, los metadatos del vídeo (formato, fecha de creación, duración, etc.) se indexan en Amazon DynamoDB para realizar consultas.

Durante este flujo de trabajo, una instancia de Auto Scaling puede monitorizar constantemente la cola de entrada. En función de la cantidad de mensajes en la cola de entrada, la instancia de Auto Scaling ajusta dinámicamente la cantidad de instancias de Amazon EC2 de transcodificación para satisfacer los requisitos de tiempo de respuesta de los clientes del sitio web². Recordemos, además, que un grupo de autoescalado utiliza una sola AMI para crear instancias idénticas, pero gracias a los mensajes de SQS estas instancias realizan trabajos distintos.

¹Página web de AWS. "Preguntas frecuentes sobre Amazon SQS. P: ¿Puede proporcionarme un ejemplo de un caso de uso de Amazon SQS?". [En línea] URL disponible en: <https://aws.amazon.com/es/sqs/faqs/>

²Ibidem.

Para poder integrar SQS con nuestras aplicaciones, AWS pone a nuestra disposición SDKs de distintos lenguajes y plataformas, con librerías y códigos de ejemplo. Además, también podemos usar SQS con AWS CLI.

1

Utilizando el servicio SQS, las colas se podrán crear con el comando “create-queue”. Para crear una cola estándar basta con usar la opción “queue-name” con el nombre de la cola; sin embargo, para las colas FIFO hay que especificarlas con la opción “attributes”, además de añadirle al nombre de la cola la terminación “.fifo”.

```
$ aws --profile SQS sqs create-queue --queue-name masterfifoqueue.fifo --
attributes {"FifoQueue\":"true\"}
{
  "QueueUrl": "https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo"
}
```

Este comando devuelve una URL que se usará para la comunicación con la cola.

2

Para añadir un mensaje a la cola, usamos el comando “send-message”. Para este comando la opción es “queue-url”, que es la URL antes mencionada, “message-body” con el cuerpo del mensaje y los parámetros “message-group-id” y “message-deduplication-id”. También es importante señalar que el mensaje puede tener más atributos, no solo el cuerpo, añadidos con la opción “message-attributes”.

```
$ aws --profile SQS sqs send-message --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --message-body "Primer
mensaje de prueba" --message-group-id 1 --message-deduplication-id 1
{
  "MessageId": "8d160eba-5f02-4210-8c5b-26a9ae28b6e2",
  "SequenceNumber": "18835140618058991616",
  "MD5OfMessageBody": "ea686eb13cd34d70a9ee1e9414f0f6d8"
}
```

Se puede saber el número de mensajes en la cola con facilidad con el comando “get-queue-attributes” y la opción “attribute-names” con el valor “ApproximateNumberOfMessages”.

```
$ aws --profile SQS sqs get-queue-attributes --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --attribute-names
"ApproximateNumberOfMessages" --query 'Attributes' --output text
1
```

Para recibir mensajes, usamos el comando “receive-message”. Con la opción “attribute-names”, podemos solicitar más datos referentes al mensaje (en este caso, cuántas veces se ha visto el mensaje). Si el servicio no encuentra ningún mensaje en la cola, con la opción “wait-time-seconds” se mantiene en espera los segundos indicados a que llegue un nuevo mensaje a la cola. La opción “visibility-timeout” está relacionada con la opción “message-group-id” al enviar el mensaje. Esta opción provoca que todos los mensajes con el mismo ID de grupo que el mensaje recibido dejen de estar disponibles en la cola por un intervalo de segundos igual al especificado en la opción.

```
$ aws --profile SQS sqs receive-message --queue-url https://eu-west-1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --attribute-names ApproximateReceiveCount --visibility-timeout 5 --wait-time-seconds 20
{
  "Messages": [
    {
      "MessageId": "8d160eba-5f02-4210-8c5b-26a9ae28b6e2",
      "ReceiptHandle": "AQEBmCXEq+NAKWHhBKsf1fI9dCvM4PsZ9fsqMrRE0nPvVftuJrEBBcF0lvAJtkaDILxtBCKOQGi8gOYprusi4blRNNKxtroylVH5MYBjEgNo6Tf0YxSO2MsXP66VuCU7+4aj33pBHFva3nNqOZO3sJSQWL8SXs1f5JK9HBPLvryRgBZA9C04I3PyVZ676qW4Okoe7Ch5qJF9J81k2Q/yfeTmDnEs1boL//hqQ96zaw/TYL+vC0v5uSocPPopsUBExORkI4ZwhpPFJZSp5/kYqD77sPXkqfDXrhx4kRIY9/uDcU=",
      "Attributes": {
        "ApproximateReceiveCount": "1"
      },
      "MD5OfBody": "ea686eb13cd34d70a9ee1e9414f0f6d8",
      "Body": "Primer mensaje de prueba"
    }
  ]
}
```

En la salida del comando se puede ver el mensaje en el atributo “Body”. Otro atributo importante es “ReceiptHandle”, ya que es el código que se necesita para borrar los mensajes ya leídos.

4

Para borrar los mensajes de la cola se usa el comando “delete-message” con el “receipt-handle” recibido al leer el mensaje. Sin embargo, esto se tiene que hacer antes de que transcurra el tiempo indicado con “visibility-timeout”; de lo contrario, dará un error:

```
$ aws --profile SQS sqs delete-message --queue-url https://eu-west-1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --receipt-handle AQEBmCXEq+NAKWHhBKsf1fI9dCvM4PsZ9fsqMrRE0nPvVftuJrEBBcF0lvAJtkaDILxtBCKOQG8gOYprusi4blRNNKxtroylVH5MYBjEgNo6Tf0YxSO2MsXP66VuCU7+4aj33pBHFva3nNqOZo3sJSQWL8SXs1f5JK9HBPLvryRgBZA9C04I3PyVZ676qW4Ocoe7Ch5qJF9J8lk2Q/yfeTmDnEs1boL//hqQ96zaw/TYL+vC0v5uSocPPopsUBExORkI4ZwhpPFJZSp5/kYqD77sPXkqfDXrhx4kRIY9/uDcU=
```

An error occurred (InvalidParameterValue) when calling the DeleteMessage operation: Value

```
AQEBmCXEq+NAKWHhBKsf1fI9dCvM4PsZ9fsqMrRE0nPvVftuJrEBBcF0lvAJtkaDILxtBCKOQG8gOYprusi4blRNNKxtroylVH5MYBjEgNo6Tf0YxSO2MsXP66VuCU7+4aj33pBHFva3nNqOZo3sJSQWL8SXs1f5JK9HBPLvryRgBZA9C04I3PyVZ676qW4Ocoe7Ch5qJF9J8lk2Q/yfeTmDnEs1boL//hqQ96zaw/TYL+vC0v5uSocPPopsUBExORkI4ZwhpPFJZSp5/kYqD77sPXkqfDXrhx4kRIY9/uDcU= for parameter ReceiptHandle is invalid. Reason: The receipt handle has expired.
```

Para solucionar este problema se puede incrementar el tiempo de “visibility-timeout” o utilizar *scripts* para realizar el proceso rápidamente. Para crear *scripts*, primero se deberá guardar el mensaje en una variable para usarlo más fácilmente:

```
$ MENSAJE=$(aws --profile SQS sqs receive-message --queue-url https://eu-west-1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --visibility-timeout 5 --wait-time-seconds 20)
$ echo $MENSAJE
{ "Messages": [ { "ReceiptHandle": "AQEBUQVMCKut3TzO27Uoi37zgu8j4j9JIVfcL/GTii8mMvvzU0v7JkAing13ZlUPlxUownFLujvkzpqfWwAHuUCLIsbMcmLGa+k4qe8BjQnRmz+ulTTPv5+612IMOzEX3On7FPFHriplAIP+o0yhsEaFJdaex6YuEQx+OYz7Zw9o2YlQifaj5lQwzDzL/7S5F3bHmXlnfPTTG0/2P6L7vreceRMXdcXnnFn/dh1T8ECFi9d49HsOYrnGeFDvtR2g6GfmAu4GRT6V/coh4YrsG3+4aVLqzY4Xv4tFIIfvgAW+X8A=", "MessageId": "8d160eba-5f02-4210-8c5b-26a9ae28b6e2", "MD5OfBody": "ea686eb13cd34d70a9ee1e9414f0f6d8", "Body": "Primer mensaje de prueba" } ] }
```

Como el mensaje se guarda con el formato json, será necesario usar la herramienta instalada previamente, “jq”. Para acceder al cuerpo del mensaje usamos:

```
$ echo $MENSAJE | jq '.Messages[] | .Body'
"Primer mensaje de prueba"
```

5

Para acceder al dato "ReceiptHandle":

```
$ echo $MENSAJE | jq '.Messages[] | .ReceiptHandle'
"AQEBUQVMCKut3TzO27UOi37zgu8j4j9JIVfcL/GTIi8mMvvzU0v7JkAing13ZlUPlxUownFLujvkz
pqfWwAHuUCLIsbMcmLGa+k4qe8BjQnRmz+uLTpV5+612IMOzEX3On7FPFHriplAIP+o0yhsEaFJd
aeX6YuEQx+OYz7Zw9o2YlQifaJ5lQwzDzL/7S5F3bHMxLNfPTTG0/2P6L7vreceRMXdcXnnFn/dh1T
8ECFi9d49HsOYrnGeFDvtR2g6GfmAu4GRT6V/coh4YrsG3+4aVLqzY4Xv4tFIfvgAW+X8A="
```

Ahora, usando el comando del sistema "xargs" podemos usar esa salida como parte de otro comando:

```
$ aws --profile SQS sqs get-queue-attributes --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --attribute-names
"ApproximateNumberOfMessages" --query 'Attributes' --output text
1

$ MENSAJE=$(aws --profile SQS sqs receive-message --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --visibility-timeout 5
--wait-time-seconds 20)

$ echo $MENSAJE | jq '.Messages[] | .ReceiptHandle' | xargs aws --profile
SQS sqs delete-message --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --receipt-handle

$ echo $MENSAJE | jq '.Messages[] | .Body'
"Primer mensaje de prueba"

$ aws --profile SQS sqs get-queue-attributes --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo --attribute-names
"ApproximateNumberOfMessages" --query 'Attributes' --output text
0
```

6

Se puede ver que al principio había un mensaje en la cola, ese mensaje se guardó en una variable y fue eliminado de la cola. Además, en la variable se almacena toda la información del mensaje para poder trabajar con ella.

Por último, si se necesita limpiar una cola, pero sin borrarla, se usa el comando "purge-queue":

```
$ aws --profile SQS sqs purge-queue --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo
```

Pero si la intención es borrarla, se usa "delete-queue":

```
$ aws --profile SQS sqs delete-queue --queue-url https://eu-west-
1.queue.amazonaws.com/700226713106/masterfifoqueue.fifo
```

6.3. AWS Kinesis*

* **Nota:** Como AWS Kinesis no tiene capa gratuita y según el tipo de pruebas que se ejecuten el coste podría llegar a ser elevado, solo se explica su uso de forma teórica sin entrar en la práctica.

Kinesis es una herramienta de AWS para la recopilación, el procesamiento y el análisis de datos de *streaming*. Esta herramienta se divide en tres partes.

6.3.1. Amazon Kinesis Data Streams

Esta herramienta de Kinesis se usa para crear aplicaciones que procesen y analicen los datos. Kinesis Data Stream puede registrar y almacenar terabytes de datos por hora procedentes de miles de orígenes (*logs*, redes sociales, transacciones financieras, secuencias de clics, etc.). Para esto, AWS pone a disposición de los desarrolladores la Kinesis Producer Library (KPL), para introducir los datos del usuario en *stream* de datos, y la Kinesis Client Library (KCL) para poder consumirlos generando alertas, implementando anuncios dinámicos o alimentando paneles en tiempo real, por poner algunos ejemplos. También se podrá emitir datos desde Kinesis Data Streams a otros servicios de AWS, como Amazon S3, Amazon Redshift, Amazon EMR o AWS Lambda. Otra de las capacidades de Kinesis Data Stream es permitir que varias aplicaciones de Kinesis puedan procesar la misma transmisión de manera simultánea.

6.3.2. Amazon Kinesis Data Firehose

A diferencia de Amazon Kinesis Data Stream, con esta herramienta no es necesario escribir aplicaciones para manejar los recursos. Solo es necesario configurar los productores de datos para que envíen los datos a Kinesis Data Firehose y este automáticamente entrega los datos al destino que se haya especificado. Adicionalmente, Kinesis Data Firehose puede transformar los datos antes de entregarlos. Se podrán registrar, transformar y enviar datos a otras herramientas de AWS como S3, Redshift o Elasticsearch y, desde estas, usar aplicaciones y herramientas de análisis e inteligencia empresarial para analizar los datos de *streaming*.

La característica más importante de Kinesis Data Firehose es su facilidad de uso, ya que permite registrar y cargar datos de *streaming* o configurar un preprocesado para convertir los datos de origen a los formatos requeridos por los *datastores* de destino y todo con unos pocos clics en la consola de administración de AWS. AWS se encarga automáticamente de aprovisionar, administrar y escalar los recursos informáticos de memoria y red para poder cargar los datos de *streaming*.

6.3.3. Amazon Kinesis Data Analytics

Amazon Kinesis Data Analytics permite procesar los datos de *streaming* en tiempo real usando SQL estándar. Esta herramienta se encarga de administrar todos los recursos informáticos para ejecutar las consultas de forma continua, escalando automáticamente y adaptándose al volumen y capacidad de procesamiento de los datos de entrada, con la intención de obtener latencias de procesamiento inferiores a un segundo.

Con Amazon Kinesis, se pueden "realizar análisis en tiempo real de datos que tradicionalmente se analizaban con procesamiento en lotes en almacenes de datos o con marcos Hadoop. Los casos de uso más comunes incluyen lagos de datos, ciencia de datos y aprendizaje automático. Se puede usar Kinesis Firehose para cargar datos de *streaming* de manera continua en lagos de datos de S3. También se pueden actualizar modelos de aprendizaje automático con mayor frecuencia a medida que se pongan a disposición nuevos datos, lo que garantiza la precisión y fiabilidad de los resultados"³.

Nota: Como AWS Kinesis no tiene capa gratuita y según el tipo de pruebas que se ejecuten el coste podría llegar a ser elevado, solo se explica su uso de forma teórica sin entrar en la práctica.

³ Página web de AWS. "Amazon Kinesis". [En línea] URL disponible en: <https://aws.amazon.com/es/kinesis/>



En la figura 31, se puede ver un ejemplo de uso de Kinesis.



Figura 31. Ejemplo de uso de Amazon Kinesis.

Fuente: <https://aws.amazon.com/es/kinesis/>

VII. Resumen



En esta unidad, hemos hecho una introducción a los modelos de servicios en la nube y la utilidad de estos en comparación con los actuales entornos físicos.

Una vez explicadas las ventajas de los servicios en la nube, se ha empezado a estudiar AWS como ejemplo del modelo IaaS, para presentar las capacidades, servicios y herramientas que podemos encontrar en este tipo de modelos ya que, aunque se ha basado el estudio en AWS, el resto de proveedores presentan unas características similares.

Para poder utilizar la interfaz de línea de comando de AWS con las distintas herramientas, se ha explicado cómo dar acceso a los usuarios y su instalación.

Usando la interfaz y la consola de administración, se ha hecho un recorrido por los distintos servicios que podemos encontrar en AWS, como la creación de instancias, de redes, BB. DD., colas de mensaje, almacenamiento y análisis de datos en *streaming*.

Ejercicios

Caso práctico



Crear un *script* para, por línea de comando, configurar una VPC y dos subredes con las siguientes características:

- Subred 1: subred pública con acceso a internet.
- Subred 2: subred privada. Solo se puede acceder a ella desde la subred 1.

Crear dos instancias de AWS usando la AMI `ami-4d46d534`, una en cada subred y con un grupo de seguridad para permita el acceso por el puerto 22 a las dos instancias.

Solución

Para conseguir el objetivo, primero creamos la VPC, guardando la información del ID en una variable:

```
VPCID=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --query 'Vpc.VpcId' --output text)
```

Ahora creamos las subredes. Como no se ha especificado en el enunciado, escogeremos un CIDR cualquiera y usaremos la misma zona en las dos.

```
SUBNETID1=$(aws ec2 create-subnet --availability-zone eu-west-1a --cidr-block 10.0.0.0/24 --vpc-id $VPCID --query 'Subnet.SubnetId' --output text)
SUBNETID2=$(aws ec2 create-subnet --availability-zone eu-west-1a --cidr-block 10.0.1.0/24 --vpc-id $VPCID --query 'Subnet.SubnetId' --output text)
```

Creamos el grupo de seguridad:

```
SECGRID=$(aws ec2 create-security-group --description "Acceso por SSH" --group-name AccesoSSH --vpc-id $VPCID --query 'GroupId' --output text)
```

Como por defecto las tablas de rutas de una VPC permiten que todas las subredes se comuniquen entre ellas, solo es necesario crear una tabla de rutas para la subred 1, pero primero hay que crear el *gateway*:

```
GATEWAYID=$(aws ec2 create-internet-gateway --query 'InternetGateway.InternetGatewayId' --output text)

aws ec2 attach-internet-gateway --internet-gateway-id $GATEWAYID --vpc-id $VPCID

ROUTETID=$(aws ec2 create-route-table --vpc-id $VPCID --query 'RouteTable.RouteTableId' --output text)

aws ec2 associate-route-table --route-table-id $ROUTETID --subnet-id $SUBNETID1

aws ec2 create-route --route-table-id $ROUTETID --destination-cidr-block 0.0.0.0/0 --gateway-id $GATEWAYID
```


Y ahora, para terminar, creamos las dos instancias (a la instancia de la subred 1 le agregaremos una IP pública):

```
aws ec2 run-instances --image-id ami-4d46d534 --count 1 --instance-type  
t2.micro --key-name "LlaveMaestra" --security-group-ids $SECGRID --subnet-id  
$SUBNETID1 --associate-public-ip-address  
  
aws ec2 run-instances --image-id ami-4d46d534 --count 1 --instance-type  
t2.micro --key-name "LlaveMaestra" --security-group-ids $SECGRID --subnet-id  
$SUBNETID2
```

Recursos

Enlaces de Interés



https://imfformacion-my.sharepoint.com/:u:/g/personal/masterbigdata_imf_com/Eb184ATgYltCmk88W32BVf8Be6hGsvkUIQHcxKg7Lb1eGg?e=xKzadU

https://imfformacion-my.sharepoint.com/:u:/g/personal/masterbigdata_imf_com/Eb184ATgYltCmk88W32BVf8Be6hGsvkUIQHcxKg7Lb1eGg?e=xKzadU

Ubuntu.ova



http://imfformacion-my.sharepoint.com/personal/masterbigdata_imf_com/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fmasterbigdata_imf_com%2FDocuments%2FM6_INFRAESTRUCTURA_BIG_DATA%2FUbuntu%2Eova&parent=%2Fpersonal%2Fmasterbigdata_imf_com%2FDocuments%2FM6

http://imfformacion-my.sharepoint.com/personal/masterbigdata_imf_com/_layouts/15/onedrive.aspx?id=/personal/masterbigdata_imf_com/Documents/M6_INFRAESTRUCTURA_BIG_DATA/Ubuntu.ova&parent=/personal/masterbigdata_imf_com/Documents/M6_INFRAESTRUCTURA_BIG



https://imfformacion-my.sharepoint.com/personal/masterbigdata_imf_com/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fmasterbigdata_imf_com%2FDocuments%2FM6_INFRAESTRUCTURA_BIG_DATA%2FUbuntu%2Eova&parent=%2Fpersonal%2Fmasterbigdata_imf_com%2FDocuments%2FM6

https://imfformacion-my.sharepoint.com/personal/masterbigdata_imf_com/_layouts/15/onedrive.aspx?id=/personal/masterbigdata_imf_com/Documents/M6_INFRAESTRUCTURA_BIG_DATA/Ubuntu.ova&parent=/personal/masterbigdata_imf_com/Documents/M6_INFRAESTRUCTURA_BIG



https://imfformacion-my.sharepoint.com/:u:/g/personal/masterbigdata_imf_com/Eb184ATgYltCmk88W32BVf8Be6hGsvkUIQHcxKg7Lb1eGg?e=xKzadU

https://imfformacion-my.sharepoint.com/:u:/g/personal/masterbigdata_imf_com/Eb184ATgYltCmk88W32BVf8Be6hGsvkUIQHcxKg7Lb1eGg?e=xKzadU

Máquina Virtual

Bibliografía

- **AWS CLI documentation.** : Documentación obtenida de la ayuda en la interfaz de línea de comando de AWS (AWS Help).
- **AWS documentation.** : [En línea] URL disponible en: <https://aws.amazon.com/documentation>

Glosario.

- **AMI:** Las AMI (Amazon Machine Image) son imágenes de instancias con la información requerida para lanzar una instancia nueva idéntica a la original.
- **Autoescalado:** Capacidad de aumentar o disminuir los recursos informáticos disponibles según la carga de trabajo y de forma automática.
- **AWS CLI:** Interfaz de línea de comando usada para administrar los servicios de AWS.
- **Container-as-a-Service (CaaS):** Nuevo modelo de servicios en la nube basado en el despliegue de contenedores, como Docker o Kubernetes.

- **EC2 (Elastic Compute Cloud):** Uno de los servicios principales de AWS, ya que desde él se administran las instancias.
- **Infraestructure-as-a-Service (IaaS):** Modelo de servicios en la nube en los que el usuario tiene un control casi completo sobre la infraestructura virtual suministrada por el proveedor.
- **Instancias:** Nombre que utiliza AWS para denominar las máquinas virtuales creadas en sus sistemas.
- **Kinesis:** Servicio de AWS para el control, procesamiento y análisis de datos en streaming.
- **Platform-as-a-Service (PaaS):** Modelo de servicios en la nube usado principalmente por desarrolladores para desplegar sus aplicaciones sin tener que dedicarse a configurar la infraestructura.
- **S3 (Simple Storage Service):** Sistema básico de almacenamiento en AWS.
- **Software-as-a-Service (SaaS):** Modelo básico de servicio en la nube en el que el usuario solo tiene acceso a las aplicaciones creadas por el proveedor.
- **SQS (Simple Queue Service):** Servicio de administración de colas de mensajes.
- **VPC (Virtual Private Cloud):** Red lógica en AWS que permite aislar y proteger las instancias desplegadas en ella.