Sistemas en nube pública: AWS

Rubén Gómez Olivencia

2023-2024

- Github: https://github.com/yuki

Puedes encontrar la última versión de este libro en formato **HTML** en el siguiente link, así como otros libros que he creado. Para descargar el código fuente en formato **Markdown** visita el repositorio en GitHub.

Información

Por favor, ponte en contacto conmigo si encuentras algún fallo, falta de ortografía o quieres mejorar de alguna manera este libro. Gracias.

I Cloud Computing

II AWS

III VPC

IV EC2

Cloud Computing

1. Introducción

Los servicios que utilizamos al navegar por internet, al consultar páginas web o al utilizar *apps* en el móvil, realmente están alojados en un conjunto de ordenadores (denominados **servidores**) que están almacenados en un **CPD** (centro de procesamiento de datos).

Hace unos años había empresas pequeñas que contaban con sus propios servidores en sus oficinas, abaratando parte del coste del mantenimiento de los servidores, ya que no tenían que alquilar el espacio o el propio servidor. Si se quedaba pequeño, compraban uno nuevo, o reajustaban los recursos.

Con el avance de internet, han ido surgiendo distintos proveedores que ofrecen servicios para que las empresas puedan contratarlos en condiciones más favorables y en CPDs profesionales. Veremos diferentes tipos de computación y diferenciaremos distintos tipos de "computación en la nube".

2. Sistemas de computación propios

Dado que tener el servidor en una oficina puede suponer distintos riesgos (pérdida de electricidad, robo de datos, acceso al servidor por parte de personas no autorizadas, tener la necesidad de tener IP estática en la oficina,...), muchas empresas deciden delegar la parte pública de sus servicios en empresas que cumplen con estándares de seguridad para tener servidores de manera segura.

Información

Se siguen teniendo servidores en las oficinas, pero son para tareas internas, de desarrollo o para el control de dominios. No es habitual tener servicios externos.

Para los servicios externos, dependiendo del tamaño de la empresa, hoy en día lo habitual es delegar parte del mantenimiento o de los servicios en un proveedor externo.

Algunos ejemplos de sistemas de computación propios (cuando nos referimos a *hardware* propio) puede ser:

- **CPD propio**: Esto sólo está al alcance de empresas muy grandes, que necesitan de muchos servidores y que tienen la necesidad de montar su propio CPD parar sus propios servicios.
- **Alquiler de un *rack* en un CPD profesional**: Existen empresas que se encargan de crear CPDs donde alquilan armarios donde poder colocar nuestros servidores físicos. La empresa nos proporcionará el espacio, el ancho de banda que contratemos, y puede que otros servicios extra de red (*firewall* perimetral, sistemas para apaciguar ataques de denegación de servicios...). Por otro lado, el *hardware* del servidor lo proporcionaremos nosotros, y cualquier posible rotura del mismo será cosa nuestra.

  En este tipo de sitios se necesita pedir cita previa para acceder a los servidores, y sólo ciertas personas pre-autorizadas previamente podrán entrar.
- **Alquiler de máquinas virtuales**: Con el ***boom*** de la virtualización, surgieron compañías que ofrecen

la posibilidad de contratar unos recursos (con computación, RAM y almacenamiento limitados) que están virtualizados dentro de la infraestructura del proveedor. Esto hace que sea barato de contratar, pero toda la gestión del sistema operativo y del software sea por nuestra cuenta.

Dependiendo de las necesidades de la empresa, se hará uso de un sistema u otro. El problema en estos casos, es que normalmente es necesario un equipo de informáticos que tengan los conocimientos adecuados dependiendo del tipo de infraestructura que tengamos.

Aunque hoy en día este tipo de computación sigue siendo barato y funcional, el término "computación en la nube" ha hecho que este tipo de contrataciones hayan quedado en parte relegadas.

3. Computación en la nube

La computación en la nube se puede resumir como el uso de una red de servidores al que de manera remota se puede solicitar una serie de recursos que nos proporcionará de manera casi instantánea, y sin la gestión activa por nuestra parte. Normalmente **la gestión de todo se puede realizar a través de un panel web de configuración**.

En lugar de solicitar un recurso físico como tal, "la nube", que está compuesta por un conjunto de servidores que ha sido configurada como "un todo", es capaz de proporcionarnos distintos tipos de recursos dependiendo de las necesidades que tengamos en ese momento.

Por poner sólo unos ejemplos:

- **Software as a Service** (SaaS): En este caso se contrata lo necesario para poder hacer uso de un software y todo los datos que se vayan a almacenar con él. El proveedor nos da acceso a una instancia del software que está configurada y que permitirá el uso con unos recursos limitados.

  De esta manera, no nos tendremos que preocupar del *hardware* en el que está instalado, o la propia instalación del software. **Ejemplos**: servicio blog; sistema ERP para una empresa; sistema de base de datos con alta disponibilidad y *backups* incluidos...

- **Platform as a service** (PaaS): Se nos proporcionará una plataforma donde podremos desplegar o correr el servicio que configuremos por nuestra cuenta. El ejemplo más habitual es el de obtener una máquina virtual donde podremos desplegar el servicio que nos interese instalar.

  La ventaja del PaaS es que no nos tenemos que preocupar en qué *hardware* está corriendo, y que en caso de necesidad, podríamos ampliar los recursos de manera sencilla. Como desventaja, es que el coste puede volverse muy elevado y en algunos casos complejo de calcular.

- **Infraestructure as a Service** (IaaS): Cuando nos referimos a "Infraestructura" hace referencia a un conjunto de servicios como son la red, el almacenamiento, servidores, virtualización, servicios... El proveedor nos proporcionará un conjunto de servicios que es equiparable a tener una red propia con nuestros propios servidores y servicios.

  A la hora de mantener y desplegar este tipo de arquitectura, debemos tener los conocimientos

adecuados para saber qué se está realizando. Los proveedores facilitan la labor de realizar los despliegues, pero es tarea de quien realiza la acción de conocer exactamente qué se está haciendo y el coste que tendrá.

4. Tipos de computación en la nube

A la hora de hablar de computación en la nube debemos de distinguir principalmente entre dos tipos. Dependiendo del tamaño de nuestra empresa, del departamento de informática que tengamos, de los conocimientos que tengamos y/o del tiempo que tengamos para aprender, deberemos optar por una o por otra.

4.1. Nube privada

La gestión de una nube privada es aquella que está diseñada para una sola organización, y que será administrada por la propia empresa o con ayuda de terceros. Crear una infraestructura propia para la gestión de una nube privada no es una tarea sencilla, ya que va a requerir de una inversión inicial de dinero para la compra de bastante *hardware*, conocimiento para administrarlo, mantenimiento, posibles cambios a futuro...

¡Cuidado!

Crear y mantener una nube privada no es una tarea al alcance de cualquier empresa. Es una labor compleja y cara.

Para poder crear una nube privada existe distinta tecnología de orquestación de servicios que se puede utilizar:

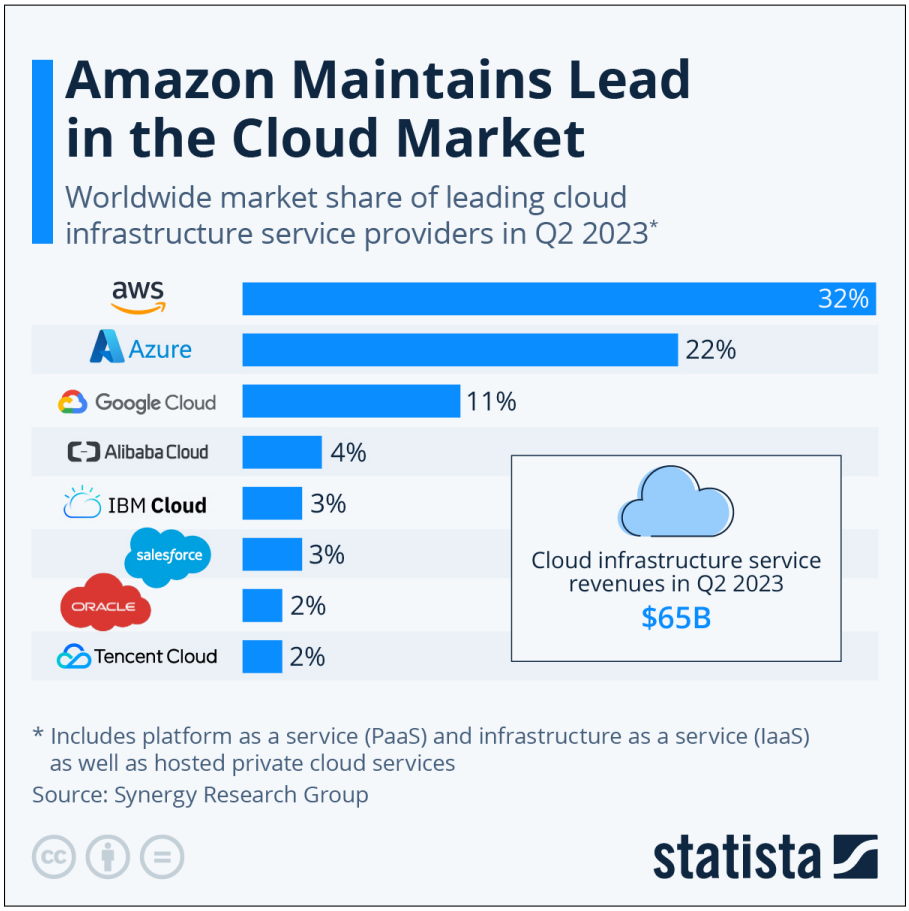
- Openstack: Nos permite realizar una infraestructura de nube completa, diferenciando entre nodos de computación, de almacenamiento, ...
- Apache CloudStack: permite la gestión de grandes redes de máquinas virtuales para la gestión en alta disponibilidad y alta escabilidad.
- Proxmox: Proxmox no es un sistema de nube privada como tal, ya que es un sistema de virtualización. Por otro lado, es un sistema que permite crear clústers de nodos de virtualización de manera sencilla, por lo que puede ser utilizado como un primer punto de partida para el inicio de una nube privada.

4.2. Nube pública

Un proveedor de nube pública es aquel que ofrece los servicios de su propia infraestructura para la gestión y creación de una infraestructura propia a un cliente a través de una suscripción de pago o a través de un sistema de “pago por servicio”.

Hoy en día los proveedores de nube pública son grandes compañías que ofrecen decenas de servicios que son fácilmente contratables a través de un interfaz de administración web, o también a través de sistemas

API para poder automatizar tareas.



Fuente: Statista

Entre los proveedores más conocidos de nube pública están:

- **Amazon Web Services** (AWS): Aunque el negocio inicial de Amazon era el de la venta de libros *online*, durante la creación de un negocio para la creación de tiendas online a terceros se dieron cuenta que sus sistemas internos no tenían la velocidad adecuada.

  Decidieron reimplementar toda la infraestructura utilizando estándares abiertos y el sistema REST para la comunicación. En el 2002 lanzó los primeros servicios pero el auge fue en el 2006 cuando lanzó **EC2** para la generación de máquinas virtuales.

- **Microsoft Azure**: Es la segunda empresa de nube pública. Aunque es la plataforma de Microsoft, Linux es el sistema más utilizado dentro de la virtualización de servicios.

- **Google Cloud Platform**: Google no se podía quedar atrás y también ofrece su sistema de nube pública. Ofrece muchos servicios y zonas de computación, y con la creación del sistema de orquestación de contenedores Kubernetes en 2015, consiguió buena cuota de mercado.

- **IBM / Softlayer**: El sistema de nube público actual de IBM fue inicialmente conocido como SoftLayer, ya que posteriormente lo compró IBM. Aunque se inició en 2005, hoy en día su cuota de mercado no es muy alta.

- **Alibaba Cloud**: Aunque en occidente es posible que no lo conozcamos, Alibaba es un gigante

tecnológico en Asia que está detrás de AliExpress, entre otros servicios. Su cuota de mercado es baja a nivel global, pero muy alta en Asia.

II

AWS

1. Antes de empezar

Hay que tener en cuenta que lo que se va a explicar en este documento es el acceso a AWS Academy, por lo que existe una pequeña diferencia con una cuenta de AWS estándar, como la que se obtendría al crear una cuenta real y al introducir los datos de la tarjeta de crédito.

AWS Academy ofrece a las instituciones de educación superior un plan de estudios de computación en la nube gratuito y la posibilidad de crear laboratorios personalizados, o libres, para poder realizar despliegues en un sistema de nube pública.

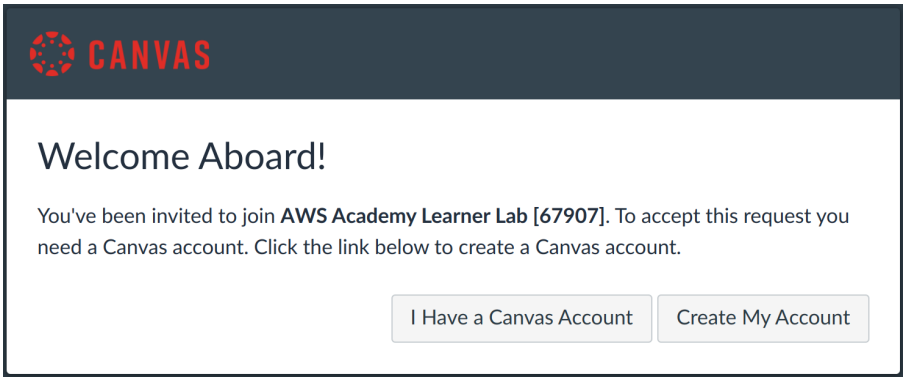
La ventaja de AWS Academy es que va a permitir a los alumnos a hacer uso del interfaz y de la infraestructura de AWS sin tener que hacer uso de una tarjeta de crédito. Ahora bien, **existen ciertas limitaciones** que no nos encontraríamos en una cuenta de AWS real.

Como vamos a hacer uso de la infraestructura real de AWS, para que no se abuse de ello, los laboratorios estarán limitados a una cantidad de 100$ y de 4 horas de uso límite. A medida que vamos realizando despliegues o vamos utilizando servicios, la cantidad de dinero irá disminuyendo.

Por otro lado, para evitar que los servicios se queden corriendo (y por tanto, gasten dinero), los laboratorios tendrán un contador que al llegar a 4 horas se apagarán. **No existe problema en volver a iniciar el laboratorio, o resetear el contador**.

2. Acceder a los cursos

Para acceder a los cursos en los que el profesor nos ha matriculado, nos habrá llegado un mail que nos mandará a una página donde tendremos que indicar si tenemos ya una cuenta o tenemos que crearla:



Debemos introducir una contraseña y aceptar los terminos para poder crear la cuenta. Una vez creada, nos mandará directamente al curso al que nos han matriculado.

Si ya tenemos una cuenta creada previamente, podemos ir al panel de acceso y elegir la opción **Student Login**. En ese punto nos aparecerá el formulario en el que debemos introducir el nombre de usuario y la contraseña.

3. Acceder al laboratorio

Una vez logueado veremos los posibles cursos en los que estamos matriculados, y debemos ir al curso correspondiente. Una vez en el curso, luego hay que ir a "Módulos → Iniciar el Laboratorio de aprendizaje de AWS Academy", y aparecerá una página en el que abajo del todo tendréis un botón para aceptar las condiciones de uso, y luego veremos algo similar a la siguiente pantalla:

En esta pantalla podemos ver distinta información del propio laboratorio y su estado:

- **AWS** ●: Estado visual del laboratorio. En este caso, el círculo aparece en rojo porque el laboratorio está apagado. Al darle a iniciar, se pondrá en color amarillo (●) mientras el laboratorio se está preparando. Cuando esté listo para ser utilizado se pondrá en verde (●).
- **Used $5 of $100**: Cuánto dinero nos hemos gastado en el laboratorio de los 100 dólares que tenemos. Este gasto no se actualiza en tiempo real, y pueden pasar algunas horas hasta que se refleje el gasto real.
- **00:00**: El tiempo que le queda al laboratorio encendido. Es una cuenta atrás de 4 horas.

- ▶ **Start Lab**: Iniciar el laboratorio.
- ■ **End Lab**: Parar el laboratorio. Todo lo realizado se para, pero hay ciertos servicios que se siguen cobrando (almacenamiento de ciertos espacios).
- ℹ **AWS Details**: Detalles del laboratorio. **Más adelante hablaremos de este apartado**.
- ℹ **Readme**: Tutoriales y manuales sobre el laboratorio.
- ↺ **Reset**: Resetea el laboratorio y borrará todo el contenido realizado. Lo deja como si estuviese recién inicializado.

El terminal que aparece es un interfaz de administración para el laboratorio, y desde el que podríamos realizar administración del laboratorio, o incluso levantar nuevos servicios a través del CLI **aws**. Por otro lado, no lo vamos a utilizar.

¡Atención!

La primera vez que encendamos el laboratorio puede tardar entre 2 y 5 minutos.

Cuando tenemos el laboratorio encendido la imagen será la siguiente

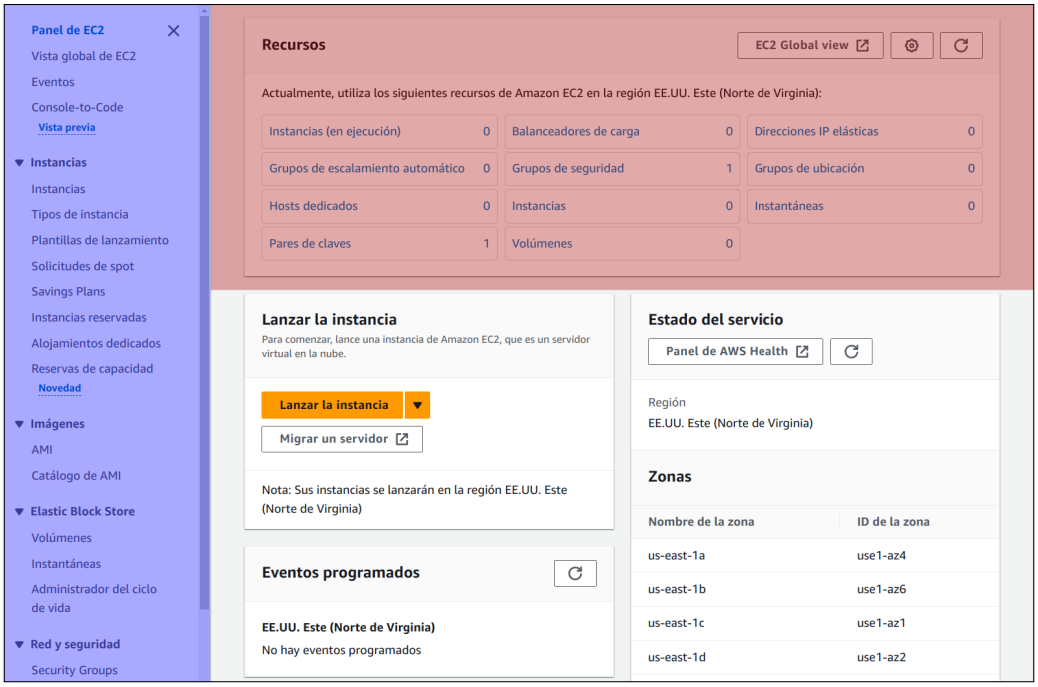
Para entrar en el laboratorio debemos hacer click en "AWS ●", lo que nos abrirá una nueva pestaña en la que veremos el panel de administración (o "Consola" llamada por AWS) donde tendremos un resumen de los servicios que hemos visitado anteriormente, aplicaciones que tengamos desplegadas, coste... La idea es tener a simple vista lo que más estemos utilizando.

4. Servicios básico de AWS

AWS cuenta hoy en día con infinidad de servicios que podemos utilizar y desplegar. Es imposible abarcar todos en un curso, por lo que aprenderemos a utilizar y a entender sólo de unos pocos. Entre los que vamos a utilizar están:

- **VPC**: Es la parte que nos proporciona la red virtual en la nube de Amazon, y todo lo que tiene que ver con direccionamiento de red, acceso a internet, ...
- **EC2**: Parte central de AWS que se encarga de la computación virtual y otros servicios que tienen que ver con máquinas virtuales.
- **RDS**: Es el servicio para crear bases de datos relacionales de Amazon. Podemos crear servicios con distintos motores, con distinta configuración, escalado, ...

Cuando entremos a algunos de estos servicios, el interfaz web que nos vamos a encontrar, por norma general, suele ser muy parecido y tiene el siguiente aspecto:



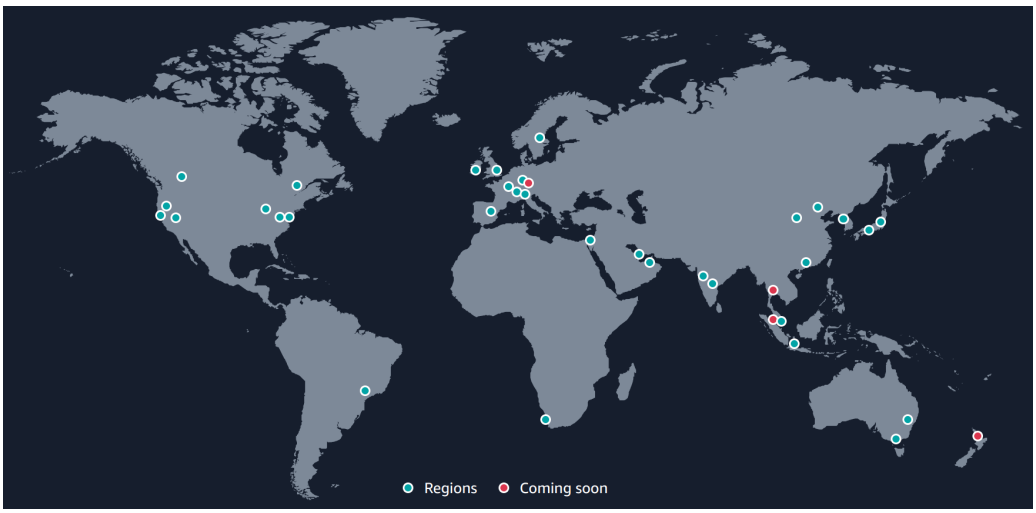
Podemos diferenciar dos apartados en la vista:

- **Panel lateral izquierdo**: En la imagen resaltado de color azul, es un menú donde podemos ver distintos apartados diferenciados por grupos. Estos apartados nos permitirán crear o configurar servicios que se nos desplegarán en la vista principal.
- **Vista principal**: En este caso sólo se ha resaltado la parte superior, ya que al entrar a cada tipo de servicio, en esta parte aparece un resumen de los recursos que tenemos contratados.

  Esta vista central cambiará teniendo en cuenta la opción seleccionada del panel lateral.

5. Regiones y zonas de disponibilidad

Dado que AWS ofrece un servicio mundial, los servidores que nos ofrecen están desplegados a lo largo de distintos países, para que el acceso desde cada zona sea lo más rápida posible. Esta infraestructura global, actualmente está diferenciada de la siguiente manera tal como aparece en la documentación oficial:



- **Regiones**: AWS tiene el concepto de una región, que es una ubicación física en todo el mundo donde se agrupan los centros de datos. Se llama a cada grupo de centros de datos lógicos "zona de disponibilidad".

  Ejemplo de Regiones en Europa: España (situada en Aragón), Irlanda, Fráncfort, Londres, París, Estocolmo, Milán, Zúrich.

- **Zonas de disponibildiad**: Cada región de AWS consta de un mínimo de tres zonas de disponibilidad con acceso aisladas y físicamente separadas dentro de un área geográfica.

  El diseño múltiple de zonas de disponibilidad de cada región de AWS ofrece ventajas para los clientes. Cada zona de disponibilidad tiene alimentación, refrigeración y seguridad física independientes y está conectada a través de redes redundantes de latencia ultrabaja.

¡Atención!

AWS Academy está limitado sólo a la región "Norte de Virginia" de Estados Unidos.

Por limitaciones de AWS Academy, todo lo que creemos sólo va a ser creado en la regiónd "Norte de Virginia" de Estados Unidos. Esto puede hacer que haya algo más de latencia desde España, pero el resultado de aprendizaje será el mismo.

VPC

1. Introducción

VPC, por sus siglas en inglés *Virtual Private Cloud*, es la infraestructura de red privada virtual que se va a crear en la nube de AWS para nosotros, donde se podrán crear los servicios y añadir los recursos que posteriormente vamos a ver.

Podemos tener distintas redes virtuales VPC, y en cada una de ellas alojar los servicios que nos interesen. De esta manera, podemos diferenciar proyectos o clientes, a través de distintos VPC, ya que cada uno de ellos está separado completamente del resto.

Aunque no vamos a realizar modificaciones al VPC creado por defecto en el laboratorio, es interesante conocer qué características existen y las posibilidades que podemos tener.

2. Características

El VPC creado en nuestro laboratorio cuenta con un direccionamiento privado con la red **172.31.0.0/16** que está situado en la zona de disponibilidad "**us-east-1**" (Norte de Virginia de Estados Unidos, tal como se ha dicho previamente). Una de las limitaciones de AWS Academy es que no podremos crear un VPC en otra zona de disponibilidad.

Información

En una cuenta AWS normal se pueden crear VPC en distintas zonas geográficas.

Aunque no vamos a profundizar, conviene entender distintas características que nos podemos encontrar y que podemos contratar o modificar.

¡Cuidado!

El apartado VPC puede ser uno de los más complejos de entender dentro de todo AWS. Es por eso que no vamos a profundizar en ello.

2.1. Subredes, tablas enrutamiento y acceso a internet

Para entender y modificar este apartado hay que tener conocimientos profundos de redes, por lo que hay que tenerlo en cuenta si se decide realizar algún tipo de modificación en estos apartados.

- **Subredes**: El VPC que tenemos por defecto consta de distintas subredes, cada una de ellas creada en una zona de disponibilidad diferente dentro de la región. Esto permite la alta disponibilidad de nuestros servicios.

- **Enrutamiento**: Básicamente nos indica qué se debe hacer con las rutas dentro de la red. Como de la red local de casa, tenemos dos apartados:

    - **La propia red**: Es decir, el direccionamiento del propio VPC, 172.31.0.0/16. Es como si fuese la red de casa.
    - **0.0.0.0**: Es el resto de direcciones que no coinciden con la red local, y que lo que hará con ellas será redirigirlas al "*internet gateway*". Sucede lo mismo en nuestra casa, que las direcciones que no son la propia red se mandan al router.

- **Internet *gateway***: Es el sistema que permite la conexión entre el VPC e internet.

¡Cuidado!

No vamos a realizar ninguna modificación de la configuración que tenemos por defecto.

IV

EC2

1. Introducción

EC2, contracción de las siglas en inglés *Elastic Compute Cloud*, es la parte central de AWS que tiene que ver con la computación, donde podremos crear máquinas virtuales y gestionar todo lo relacionado con ellas.

Dentro del apartado EC2 podremos crear las máquinas virtuales (llamadas **instancias**) donde después alojaremos o desplegaremos los servicios que nos interese. También podremos crear backups de ellas, crear imágenes para levantar instancias de manera más rápida, securizarlas...

2. Instancias

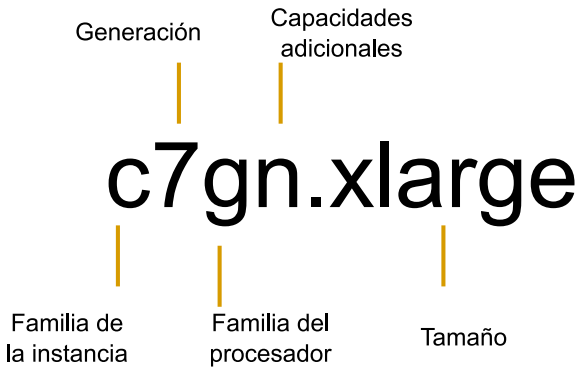
Una instancia es cómo llama AWS a una máquina virtual, por lo que los conocimientos previos que tengamos sobre máquinas virtuales se pueden asociar una instancia EC2 de AWS. Podemos realizar distintas acciones sobre las instancias que ya tengamos creadas.

Antes de crear una instancia debemos entender distintos apartados ya que a la hora de lanzar una instancia (crearla), se nos pedirá que elijamos entre distintas opciones disponibles. Es similar a lo que sucede cuando creamos una máquina virtual con Virtualbox.

2.1. Tipos de instancia

El tipo de instancia hace referencia a parte del hardware que va a tener disponible la máquina virtual al realizar el despliegue. Existe una infinidad de tipos de instancias, y es por eso que conviene leer la documentación oficial ya que en ella se detalla mucho más para qué sirve cada una de ellas.

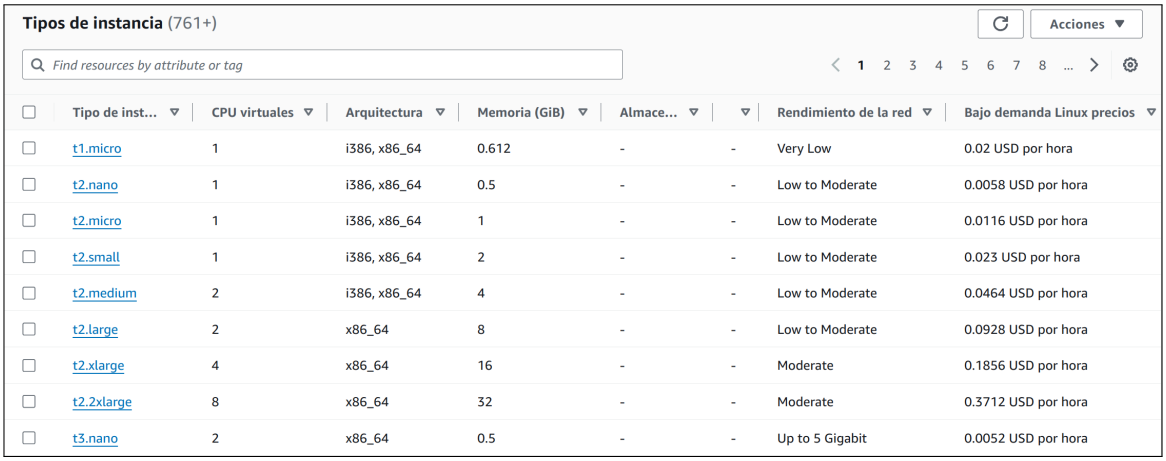
Los nombres de las instancias siguen un convenio que viene explicado en la documentación oficial.



Dado que el número de tipos de instancias actualmente es mayor que 700, puede resultar complejo saber exáctamente de primera mano cuál debemos elegir. Por otro lado, en el listado que podemos ver en el panel de EC2, nos aparece la información que a simple vista nos puede dar una idea de cómo es el tipo de instancia.

Información

Para asegurar que se adecúa a nuestras necesidades, es mejor confirmar el tipo de instancia que necesitamos mirando la documentación oficial.



Tipos de instancia (761+)

Acciones

Find resources by attribute or tag

1 2 3 4 5 6 7 8 ...

Tipo de inst...	CPU virtuales	Arquitectura	Memoria (GiB)	Almace...		Rendimiento de la red	Bajo demanda Linux precios
t1.micro	1	i386, x86_64	0.612	-	-	Very Low	0.02 USD por hora
t2.nano	1	i386, x86_64	0.5	-	-	Low to Moderate	0.0058 USD por hora
t2.micro	1	i386, x86_64	1	-	-	Low to Moderate	0.0116 USD por hora
t2.small	1	i386, x86_64	2	-	-	Low to Moderate	0.023 USD por hora
t2.medium	2	i386, x86_64	4	-	-	Low to Moderate	0.0464 USD por hora
t2.large	2	x86_64	8	-	-	Low to Moderate	0.0928 USD por hora
t2.xlarge	4	x86_64	16	-	-	Moderate	0.1856 USD por hora
t2.2xlarge	8	x86_64	32	-	-	Moderate	0.3712 USD por hora
t3.nano	2	x86_64	0.5	-	-	Up to 5 Gigabit	0.0052 USD por hora

Los valores que podemos identificar a simple vista son:

- **CPU virtuales**: El número de cores virtuales que el procesador virtual tendrá a su disposición. Dependiendo de nuestras necesidades de cómputo y procesamiento, deberemos elegir un número que se adecúe al rendimiento que esperamos. Actualmente podemos elegir tipos de instancias que tengan entre 1 y 448 CPUs.

- **Arquitectura**: Es el tipo de arquitectura del procesador con el que se desplegará la máquina virtual. Entre las arquitecturas que podemos elegir están:

  - **i386**: Arquitectura clásica de equipos de 32 bits.

  - **x86_64**: Arquitectura actual para equipos y servidores.

  - **x86_64_mac**: Arquitectura utilizada por los equipos Mac con procesadores Intel.

  - **arm64**: Arquitectura ARM de 64 bits.

  - **arm64_mac**: Arquitectura ARM de los nuevos procesadores M* de Apple.

- **Memoria**: Tamaño de la memoria RAM medido en GiB.

- **Rendimiento de la red**: Velocidad de la conexión de red que tendrá la instancia.

- **Precio de la instancia**: El coste de la instancia cuando está en funcionamiento se mide en doláres americanos (USD) por hora. El precio varía si el sistema operativo que va a correr la instancia es Linux o Windows, y por supuesto del resto de componentes virtuales.

¡Atención!

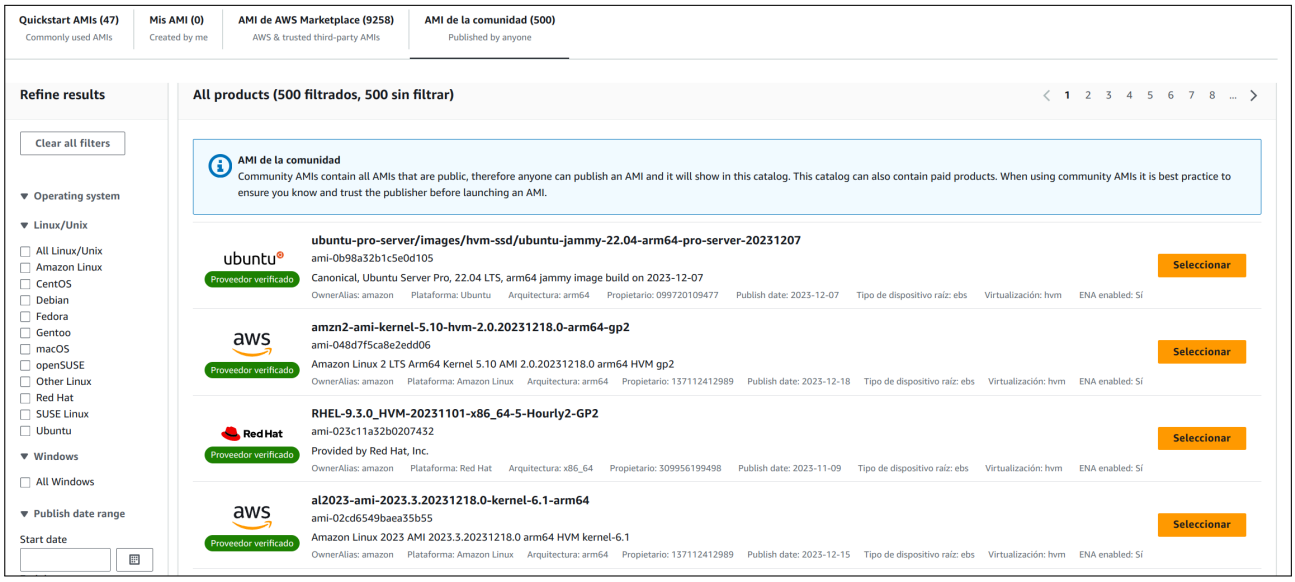
Se puede cambiar el tipo de instancia, pero para ello se debe parar la instancia}.

2.2. Imágenes AMI

Una imagen AMI (*Amazon Machine Image*)es una plantilla que contiene la configuración del software (el sistema operativo, los servicios y aplicaciones) que son necesarias para correr dentro de nuestra instancia. Existen imágenes AMI públicas creadas por proveedores de software verificados entre las que podremos elegir.

Hay un catálogo de AMI entre las que podemos elegir:

- **Quickstart AMIs**: Imágenes que normalmente suelen ser el sistema operativo con el mínimo software necesario para hacerlo funcionar. Para correr el resto de servicios, necesitaremos instalar lo que necesitemos.
- **Mis AMI**: Podemos crear nuestras propias imágenes para poder reusarlas cuando queramos.
- **AMI de AWS Marketplace**: Son imágenes creadas por empresas que pueden contener servicios de alto rendimiento, y **que en muchos casos pueden ser de pago (por licencia o uso)**.
- **AMI de la comunidad**: Imágenes que pueden ser creadas por proveedores oficiales o por comunidades que crean imágenes con software específico.



Información

Podemos crear nuestras propias imágenes para poder reusarlas cuando queramos.

Para empezar, podemos crear nuestra instancia con una imagen AMI de un proveedor oficial, para

posteriormente utilizarla como base para crear una AMI propia.

2.3. Crear una instancia

A continuación se va a explicar cómo crear una instancia de tipo Linux teniendo en cuenta todas las posibles opciones que podemos elegir. Para crearla haremos click en **Lanzar instancias** y seguiremos las indicaciones para adecuarla a nuestras necesidades.

Las respuestas que debemos contestar y tener en cuenta son:

- **Nombre y etiquetas**: Es el nombre que le vamos a dar a la instancia (que debería ser un nombre significativo), y posibles etiquetas para identificarla.
- **Application and OS Images (Amazon Machine Image)**: Imagen AMI que queremos desplegar en la instancia.
- **Arquitectura**: Arquitectura del procesador.
- **Tipo de instancia**: El tipo de instancia que queremos desplegar para las necesidades que tenemos.
- **Par de claves (inicio de sesión)**: El par de claves asimétricas (**pública-privada**) que vamos a utilizar para realizar la conexión a la instancia. Más adelante explicaremos cómo acceder a la instancia.

  ¡Atención!

  Debemos elegir el par de claves "vockey" para poder acceder a la instancia.

- **Configuraciones de red**: Ajustar las configuraciones de red. Por defecto podemos elegir el **grupo de seguridad** que queremos asignar a la instancia. Como opciones avanzadas, podemos desactivar el tener IP pública o asignar la instancia a una subred.

  ¡Cuidado!

  La IP pública que se asigna no es "eterna". Cuando se apague la instancia y se vuelva a levantar obtendrá otra IP pública.

- **Configurar almacenamiento**: El disco duro principal de la instancia.
- **Opciones avanzadas**: Distintas opciones de configuraciones que podemos modificar o de hardware que podemos habilitar.

A la derecha del asistente tenemos tenemos un resumen en el que nos aparece las distintas opciones seleccionadas. En este resumen podemos elegir el número de instancias que nos interese crear, por si queremos más de una.

Una vez le damos a "Lanzar instancia" la instancia se creará y empezará el despliegue. Desde el panel de instancias, si seleccionamos la instancia recién creada veremos todos los detalles de la misma:

Tal como se puede ver, la información de la instancia está separada en distintas pestañas y en cada una de ellas nos aparecerá mucha información sobre la misma. Parte de esta información será la necesaria para poder acceder a la instancia por SSH.

2.4. Acceder a una instancia

Para acceder a una instancia desplegada tenemos distintas opciones que podemos visualizar desde el botón Conectar cuando esté la instancia seleccionada. Entre las opciones, vamos a destacar dos:

- **Conexión de la instancia EC2**: Es el método más sencillo para acceder a una instancia a través de SSH, ya que vamos a poder realizar la conexión a través del propio navegador web. Esto, por otro lado, no nos va a permitir hacer uso de todas las características que tiene una terminal real.
- **Cliente SSH**: Al igual que sucede al conectarnos a una máquina virtual normal, podemos acceder a la instancia a través de un cliente SSH de consola. Para ello, necesitamos conocer:
  - **Nombre de usuario**: Dependiendo de la AMI que hayamos desplegado, el usuario de conexión será distinto. Por ejemplo, para instancias Ubuntu el usuario es "**ubuntu**"; para instancias Debian el usuario es "**admin**"; para instancias Amazon Linux, el usuario es "**ec2-user**". **Estos usuarios tienen la posibilidad de convertirse en "root" usando el comando "sudo"**.

    Información

    Dependiendo de la AMI elegida, el usuario de acceso será distinto.

  - **DNS público / IP pública**: Lógicamente, para poder acceder a la instancia, debemos conocer cuál es su IP pública. También se puede hacer uso del DNS público, que está asociado a la IP, pero de nuevo, este cambia.

    ¡Cuidado!

    La IP pública que se asigna no es "eterna". Cuando se apague la instancia y se vuelva a levantar obtendrá otra IP pública.

- **Clave privada de acceso**: El usuario por defecto de la instancia no cuenta con contraseña, por lo que para acceder necesitamos hacer uso de una autenticación basada en **clave asimétrica** (conocidas también como "**clave pública-privada**").

> **¡Atención!**
>
> **Para poder acceder a la instancia es necesario usar claves asimétricas.**

2.4.1. Obtener clave privada

Tal como se ha dicho previamente, para poder acceder a las instancias recién creadas debemos hacer uso del sistema de claves asimétricas, ya que es el método más seguro de conexión, y así no dependemos de contraseñas.

Al crear el laboratorio se nos ha creado un par de claves llamadas "**vockey**", que sólo podemos descargar desde el panel del curso, en el apartado " **AWS Details**", dándole al botón Download PEM , que nos descargará un fichero llamado **labuser.pem**.

Este fichero de clave debe tener permisos especiales en sistemas GNU/Linux. El fichero debe tener sólo permisos de lectura para el usuario, dejando el resto de permisos sin asignar. Por lo tanto, en formato binario, **400**. Para ello debemos hacer:

Cambios de permisos en el fichero

```
ruben@vega:~$ chmod 400 labuser.pem
```

Para realizar la conexión a la instancia, y teniendo en cuenta los datos comentados previamente, el acceso se realizará utilizando el siguiente comando, sustituyendo los datos por los correspondientes para la instancia:

Conexión a una instancia mediante cliente SSH

```
ruben@vega:~$ ssh -i labsuser.pem admin@52.91.155.75
admin@ip-172-31-32-168:~$ sudo su
root@ip-172-31-32-168:/home/admin#
```

De esta manera, ya estaremos dentro de la instancia desplegada en AWS, y podremos empezar a desplegar el servicio que nos interese.

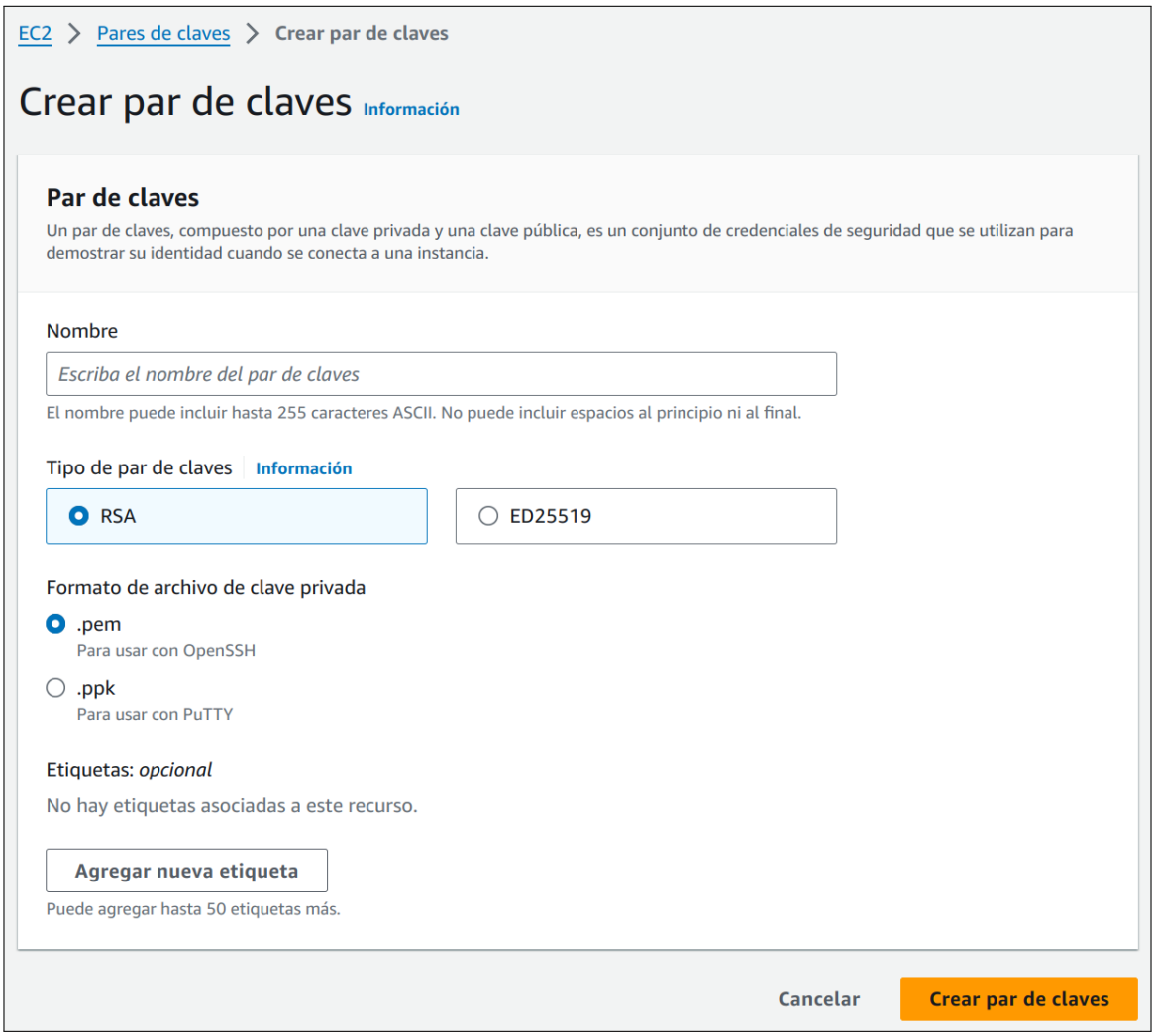
2.4.2. Crear nuevo par de claves

En entornos profesionales **es habitual crear distintas par de claves para cada proyecto, o incluso para cada instancia**, ya que de conseguir una (o en sistemas basados en contraseñas), se obtendría acceso a un gran número de instancias, **lo que es un fallo de seguridad de extrema gravedad**.

¡Cuidado!

En entornos profesionales nunca se debería reutilizar claves, y menos entre distintos proyectos.

Si queremos crear un nuevo par de claves lo podemos realizar desde el menú lateal, en el apartado: "**Red y seguridad → Pares de claves**". Ahí podremos hacer click sobre el icono Crear par de claves , que nos redirigirá a un formulario donde crear las nuevas claves.



Una vez rellenado los datos, al darle al botón de crear, se nos descargará el fichero **que nunca más podremos volver a descargar**, por lo que deberemos guardarla a buen recaudo. A partir de ahora, al crear una nueva instancia, podremos hacer uso de esta nueva clave para el poder realizar el acceso con ella.

¡Cuidado!

¡Cuidado con el certificado descargado! No lo podremos volver a descargar.

3. Direcciones IP elásticas

Una IP elástica es una **IPv4 de direccionamiento público** que podemos asignar a una instancia EC2 (o a un interfaz de red de una EC2) que hayamos creado. Esta dirección IP estará asociado a nuestro VPC aunque la instancia EC2 no esté en funcionamiento.

¡Atención!

Las direcciones IP elásticas se cobran aparte. Es mejor no contratar una IP elástica hasta que el servicio no se vaya a poner en producción.

Las instancias EC2 no cuentan siempre con la misma IP pública, y es por eso que si queremos que nuestro servicio públicamente siempre tenga la misma IP debemos asociarle una IP elástica.

4. Security Groups

Los grupos de seguridad, o ***security groups***, actúan como un **firewall** virtual que controla el tráfico de una o varias instancias. Al crear una instancia se puede especificar uno o varios grupos de seguridad, o se pueden añadir más adelante. Por defecto, **con cada instancia se nos creará un security group nuevo**.

Lo ideal es crear security groups que se reutilicen entre varias instancias, ya que si después modificamos las reglas de este security group, se aplicará a todas las instancias a las que esté asignado. Los security group cuentan con **reglas de entrada y de salida**.

4.1. Crear security group

Los security group se pueden crear desde el panel de **VPC** o de **EC2**. En este último caso dentro del apartado "**Red y seguridad → Security Groups**" y dándole al botón Crear grupo de seguridad , que nos llevará a un formulario donde podemos rellenar:

- Nombre del grupo de seguridad
- Descripción
- VPC al que pertenece
- Reglas de entrada
- Reglas de salida

4.2. Reglas de entrada

Tal como hemos dicho, un security group es un **firewall**, por lo que **si no tenemos reglas de entrada, por defecto todo el tráfico estará bloqueado**. Esta es la manera más segura para que no haya ninguna conexión no controlada que llegue a la instancia.

Información

Las reglas de entrada son el tráfico que queremos permitir entrar a la instancia.

A la hora de crear una regla de entrada, **que será el tráfico que queremos permitir**, tendremos que elegir entre:

- **Tipo de regla**: Nos aparece un menú desplegable en el que podremos elegir entre distintos tipos de reglas ya creadas. Dependiendo de la regla seleccionada, no nos permitirá elegir ni protocolo ni puertos, ya que por defecto los habrá rellenado la regla elegida.
- **Protocolo**: Es el tipo de protocolo que queremos permitir. Sólo podemos elegir entre TCP, UDP o ICMP.
- **Intervalo de puertos**: El puerto al que llegará la comunicación.
- **Origen**: Para mayor seguridad, podemos hacer que la comunicación sólo se acepte si viene desde un origen concreto. En algunos casos quizá nos interese que sea permitido desde cualquier sitio (internet incluído), en otros sólo desde la red privada del VPC, u otras sólo desde una IP concreta (pública o privada).

  ¡Atención!

  Para mayor seguridad, tenemos la opción de limitar el origen de la comunicación.

- **Descripción**: Es interesante poner una descripción a las reglas, para entender por qué la hemos creado o qué es lo que hacen.

Un security group puede tener varias reglas de entrada, lo que nos servirá para poder reutilizarlos entre distintas instancias.

Por defecto, al crear una instancia de tipo Linux se nos crea un security group que **permite el acceso desde cualquier IP del mundo al puerto 22**, para así poder realizar la conexión SSH.

4.3. Reglas de salida

En este caso lo que queremos es habilitar qué tipo de tráfico se va a permitir hacia fuera de la instancia. **Por defecto existe la regla de permitir todo el tráfico**, por lo que cualquier tipo de comunicación desde dentro de la instancia hacia fuera estará permitido.

A veces puede parecer más complejo pensar por qué interesaría bloquear tráfico en salida, pero muchas veces suele ser para evitar que si el servidor ha tenido un fallo de seguridad, se pueda explotar para realizar actos contraproducentes, como por ejemplo:

- Conectarse a páginas web externas para descargarse software malicioso.
- Enviar spam.

- Utilizar la máquina para realizar otros saltos.

V

RDS

1. Introducción

RDS, contracción de las siglas en inglés *Relational Database Service*, es el **servicio de bases de datos relacional** que podemos crear y ejecutar en la nube de AWS sin que tengamos que realizar nosotros la creación e instalación de un sistema operativo en el que se ejecute.

Este servicio facilita el despliegue del servicio, permitiendo que podamos crear servicios en alta disponibilidad sin tener que conocer los aspectos de configuración que hay detrás de ellos, ya que Amazon se encarga de realizar la configuración total del servicio.

Actualmente existen distintos sistemas gestores de bases de datos relacionales que podemos desplegar a través del servicio RDS, entre las que podemos destacar:

- **MySQL**: Es la base de datos relacional con licencia de Software Libre mas conocida a día de hoy.
- **MariaDB**: Fork de MySQL y compatible con ella.
- **PostgreSQL**: Antes del boom de MySQL era la base de datos de referencia en el mundo del Software Libre, siendo más compatible que la anterior.
- **Aurora (MySQL Compatible)**: Es una versión de MySQL modificada por Amazon para tener mayor rendimiento.
- **Aurora (PostgreSQL Compatible)**: Es una versión de PostgreSQL modificada por Amazon para tener mayor rendimiento.
- **Oracle**: Sistema de base de datos privativo desarrollado por la empresa del mismo nombre.
- **SQL Server**: Gestor de base de datos de Microsoft.

Dependiendo del proyecto que queramos desplegar, deberemos elegir entre el sistema que más se adecúe al proyecto.

2. Características

A la hora de desplegar una instancia RDS existen distintas características que podemos tener en cuenta, que son muy interesantes y en principio no importa qué SGBD elijamos, ya que están disponibles para todos ellos.

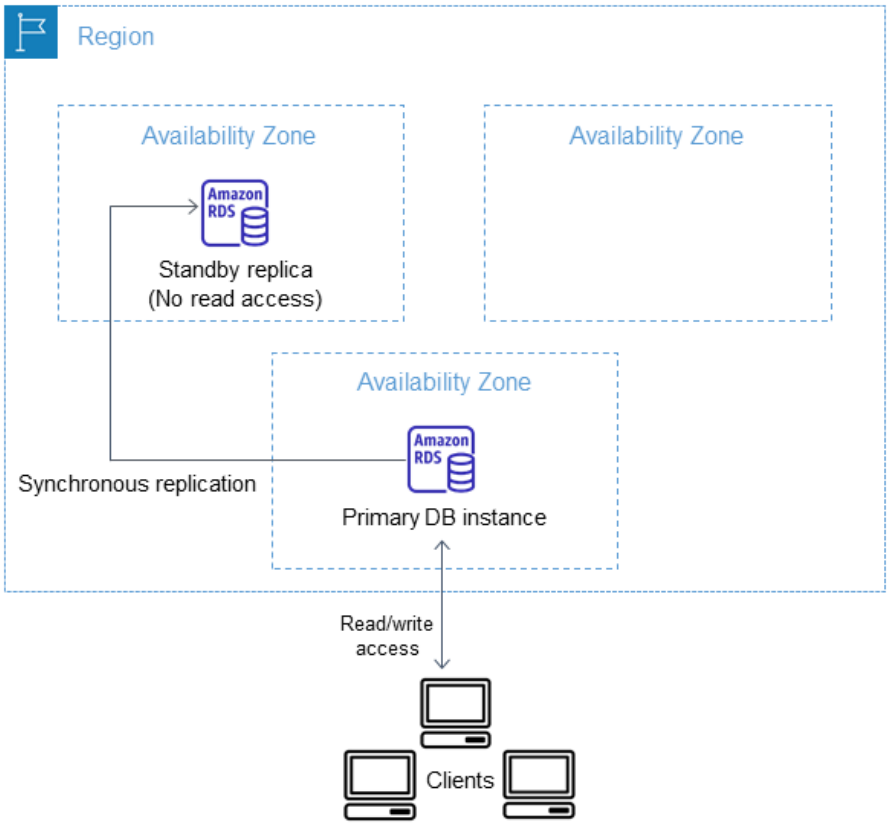
2.1. Instancia única

“Instancia de base de datos única” es la opción más sencilla a la hora de realizar un despliegue. En este caso tendremos una única instancia de base de datos que aceptará lecturas y escrituras.

El problema de este tipo de instancias es que de existir un error, o de caerse la zona de disponibilidad, perderemos acceso a la base de datos.

2.2. Instancia de base de datos Multi-AZ

Amazon RDS provee alta disponibilidad y soporte de *failover* en caso de fallo utilizando despliegues Multi-AZ. En este caso se crea una segunda instancia que es réplica de la primera en otra zona de disponibilidad, tal como se puede ver en el siguiente esquema:



Fuente: AWS

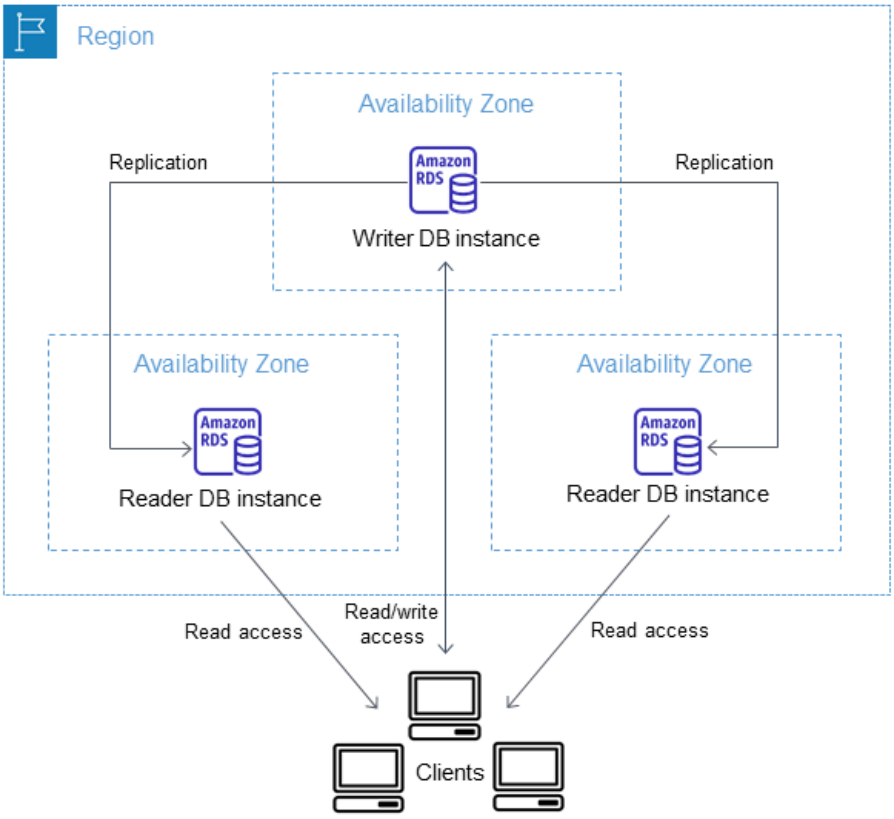
En caso de fallo, las conexiones automáticamente cambiarán a la réplica en la otra zona de disponibilidad. Este cambio puede durar entre 60 y 120 segundos tal como dice la documentación oficial.

2.3. Clúster de base de datos Multi-AZ

Marcando esta opción RDS nos creará un sistema clúster del sistema de base de datos que hayamos elegido teniendo en cuenta las siguientes características:

- **Instancia primaria**: Es la instancia que permite lectura y escritura.
- **Instancias en espera con capacidad de lectura**: Estas instancias las desplegará en otras zonas de disponibilidad diferente a donde se encuentra la instancia primaria. **Esto proporciona alta disponibilidad y redundancia de datos**. También podremos realizar cargas de trabajo de lectura sobre estas instancias.

El esquema sería el siguiente:



Fuente: AWS

Se puede crear el sistema Multi-AZ posterior al despliegue de una instancia RDS, ya que quizá no nos interese realizarlo de inicio ya que supone un coste adicional.

2.4. Copias de seguridad automáticas

Durante la creación de la base de datos podemos indicar que se realicen copias de seguridad de manera automática de la instancia de base de datos. De esta manera, podemos delegar parte de este sistema en el propio servicio.

Por otro lado, estas copias de seguridad posteriormente las podremos utilizar para levantar nuevas instancias de pruebas o para realizar la restauración de la base de datos a una copia concreta.

2.5. Implementación azul-verde

Un despliegue azul-verde (o en inglés *blue-green deployment*) es una técnica que se utiliza cuando queremos realizar modificaciones en nuestra aplicación o servicio, que trata de minimizar los riesgos de un pase de producción tradicional.

La idea es hacer uso de dos servidores (o grupos de servidores) idénticos. Normalmente el servicio actual es el denominado **blue**, y de él se realiza una réplica. En este nuevo entorno de réplica se realizarán las modificaciones necesarias para el nuevo despliegue o las nuevas características.

Cuando se haya comprobado que el nuevo entorno funciona de manera correcta, está en modo **green** (se le da luz verde para pasar a producción), se alterará a cuál de ellos se estará enviando el tráfico real, pasando

del entorno antiguo *blue* al nuevo entorno *green*.

En caso de que haya algún tipo de error, se puede dar marcha atrás, ya que es reversible, volviendo a enviar las conexión al entorno anterior *blue*.

3. Crear base de datos MySQL

Desde el panel de RDS tenemos el botón para crear una base de datos, **Crear base de datos** , y tendremos que tener en cuenta las necesidades y lo comentado previamente. Entre las opciones que nos aparecen en la nueva página, y los distintos apartados del formulario podemos destacar:

- **Método de creación de base de datos**
  - **Creación estándar**: Es el método en el que nos permite elegir más opciones a la hora de crear la instancia de base de datos.
  - **Creación sencilla**: Es la manera para crear la base de datos sin que aparezcan todas las opciones que sí aparecen con el método anterior. Algunas configuraciones se pueden modificar después y otras no.
- **Opciones del motor**: Podremos elegir el SGBD y la versión concreta que nos interese. Elegiremos **mysql**.
- **Plantillas**: Existen plantillas creadas por AWS, y dependiendo de cuál se elija, por defecto se seleccionarán otras opciones que detallaremos más adelante.
  - **Producción**: Para disfrutar de una alta disponibilidad y de un rendimiento rápido y constante.
  - **Desarrollo y pruebas**: Esta instancia se ha diseñado para su uso en desarrollo, fuera de un entorno de producción.
  - **Capa gratuita**: Para desarrollar nuevas aplicaciones, probar aplicaciones existentes o adquirir experiencia práctica.
- **Disponibilidad y durabilidad**: Opciones para crear alta disponibilidad.
  - **Clúster de base de datos multi-AZ: nuevo**: Crea un clúster de base de datos con una instancia de base de datos primaria y dos instancias de base de datos en espera con capacidad de lectura, con cada instancia de base de datos en una zona de disponibilidad (AZ) diferente.
  - **Instancia de base de datos Multi-AZ**: Crea una instancia de base de datos primaria y una instancia de base de datos en espera en una zona de disponibilidad diferente.
  - **Instancia de base de datos única**: Sólo existe una instancia de la base de datos.
- **Configuración**:
  - **Nombre de la instancia**: El nombre para poder identificar a la instancia de base de datos.
  - **Nombre del usuario maestro**: Es el nombre del usuario para administrar la base de datos.

    - **Contraseña del usuario**: Nos permite elegir la contraseña de acceso.

- **Configuración de la instancia**: Similar al caso de las instancias de computación, es para saber cuántas vCPUs y memoria RAM tendrá la instancia de base de datos.
- **Almacenamiento**:
    - **Tipo de almacenamiento**: Dónde se alojan los datos.
    - **Almacenamiento**: Cantidad de GiB para guardar los datos. Se deberá ajustar a nuestras expectativas.
    - **IOPS provisionadas**: Número solicitado de operaciones de E/S por segundo que la instancia de base de datos puede admitir.
- **Conectividad**:
    - **Recurso de computación**: Crea las "*security group*" necesarias para poder permitir la conexión desde una instancia concreta.
    - **Acceso público**: Si queremos asignar una IP pública a la base de datos y así permitir el acceso a la base de datos desde fuera del VPC.

Tal como se puede ver, hay muchas opciones entre las que se pueden elegir, pero afortunadamente, **al lado de cada opción existe un enlace para obtener más información**.

¡Atención!

Es recomendable leer la documentación de las distintas opciones que se pueden elegir durante la creación de la instancia de base de datos.

El despliegue de la base de datos puede tardar alrededor de 5 minutos, ya que crea la instancia, realiza un backup y otras tareas de configuración.

4. Conexión a la base de datos

Para realizar la conexión a la base de datos, ya sea desde una aplicación o desde un programa de acceso, necesitaremos conocer los siguientes datos:

- **Punto de enlace**: Es una dirección DNS para poder acceder a la base de datos del estilo **database-1.cpicwmm0inx1.us-east-1.rds.amazonaws.com**.
- **Puerto**: Para MySQL es **3306** salvo que lo hayamos cambiado.
- **Usuario administrador**: El usuario que hayamos elegido durante el despliegue.
- **Contraseña**: Elegida durante el despliegue.

Si tenemos el **cliente de mysql** instalado en la instancia de computación, podremos realizar la conexión de la siguiente manera:

Acceso a la base de datos desde consola

```
admin@ip-172-31-32-168:~$ mysql -u admin \
   -h prueba-gratuita.cpicwmm0inx1.us-east-1.rds.amazonaws.com  -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.0.35 Source distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MySQL [(none)]>
```

VI

Alta Disponibilidad y Arquitectura de sistemas

1. Alta Disponibilidad

La alta disponibilidad en servidores se puede definir como el diseño de infraestructura (y su implantación) que asegura la continuidad del servicio y que no tiene un único punto de fallo.

Es lógico entender que un servicio debe de ser contínuo en el tiempo, ya que debe de dar servicio de manera continuada para que los usuarios puedan acceder a él. Pero para que esta premisa sea efectiva, y para asegurarnos que así sea, **la infraestructura debe de estar redundada y carecer de puntos de fallo únicos en su diseño**.

Esto quiere decir, que de cada servicio y para cada posible punto de fallo deberá haber al menos dos de ellos, para que en caso de que uno deje de funcionar el servicio siga funcionando (dos tomas eléctricas separadas, dos servidores que otorguen el servicio, dos conexiones a internet, ...).

Es habitual que un sistema en Alta Disponibilidad deba de estar pensado desde el diseño. Algunos tipos de servicios pueden empezar como un único servidor y posteriormente realizar un escalado horizontal, formando la alta disponibilidad, mientras que para otros **el diseño en alta disponibilidad debe de estar pensado desde el comienzo** (habitualmente en algunos tipos de clusters).

1.1. Importancia de un sistema en Alta Disponibilidad

Como se ha citado previamente, la alta disponibilidad nos va a asegurar al menos tres grandes ventajas:

- Una continuidad en el servicio
- Un diseño libre de puntos de fallos únicos, gracias a la redundancia.
- Mejorar el rendimiento global.

La redundancia permitirá que en caso de fallo de algún equipamiento/servicio, al estar redundando, no afecte al servicio. Gracias al sistema de monitorización seremos capaces de ver el problema y solventarlo lo antes posible. De estar el diseño correcto, el servicio mantendrá su actividad, mientras que por el contrario, si ha habido algún fallo en el diseño de infraestructura (o el problema es más grave de lo esperado) el servicio se verá afectado.

1.2. Tipos de Alta Disponibilidad

Existen muchos tipos de alta disponibilidad dependiendo de en qué capa de infraestructura estemos hablando. Por poner unos ejemplos:

- **Redundancia eléctrica**: Los servidores normalmente cuentan con doble fuente de alimentación, por lo que cada fuente de alimentación debe de estar conectada a una toma eléctrica completamente separada de la otra.
- **Redundancia de conectividad física**: El acceso a internet debe de ser redundado.
- **Redundancia de conectividad LAN**: El acceso a la LAN/DMZ/red de servicio debe de estar redundado (stacks de switches, LACPs configurados en switches y servidores, …).

- **Redundancia de servidores**: Debe de existir una redundancia de servidores para asegurar que el servicio funcione en más de un servidor físico.
- **Redundancia de servicio**: El servicio que se ofrece debe de estar redundado entre los distintos servidores.

La alta disponibilidad también se puede diferenciar como:

- **Alta disponibilidad real**: En caso de que haya algún problema el servicio continúa como si no hubiese pasado nada, gracias a la redundancia completa de servicios/dispositivos.
- **Alta disponibilidad pasiva**: En caso de error, los servidores activos serían los que reciben el impacto del problema y hay que escalar los servidores pasivos a modo activo para que comiencen a funcionar otorgando el servicio. Como se puede presuponer, esta modificación puede ser realizada de manera automática o de manera manual (lo que llevaría algo de tiempo, y por tanto el servicio se vería afectado).

2. Arquitectura de instalación

A la hora de realizar la instalación de un sistema de información, y teniendo en cuenta que es un pilar fundamental de la empresa, habrá que tomar ciertas decisiones desde el punto de vista de sistemas hardware.

De estas decisiones se encargará el ***sysadmin***, o administrador de sistemas, pero tendrá que tener ayuda de los especialistas de la aplicación de sistemas de información, así como de determinar una decisión desde el punto de vista empresarial.

Entre las tareas que hay que tener en cuenta, se podrían destacar las siguientes:

- **Hardware**: Determinar el hardware en donde se va a realizar la instalación. Hoy en día existen distintas alternativas, como son:
  - **Hardware dedicado**: Un servidor propio para el sistema, donde se realizará la instalación sólo para este servicio.
  - **Máquina Virtual**: El servicio será instalado en una máquina virtual a través de un sistema de virtualización profesional. El servicio es agnóstico al hardware, por lo que no sabrá si está virtualizado o no. Hoy en día suele ser la opción más común dadas las ventajas que ofrecen.
- **Elección del sistema de información**: Esta es una tarea importante y que no se puede dejar de lado, ya que la decisión de optar por una herramienta u otra puede suponer un problema a futuro.

  Es por eso que se debe realizar un estudio de mercado entre las distintas posibilidades y tener en cuenta, al menos, las siguientes situaciones:
  - **Estado actual de la herramienta**: Es importante saber si la herramienta analizada cuenta con un desarrollo continuado, si existe una empresa o grupo de desarrollo por detrás que la apoye; que no esté abandonada; que sea una herramienta con buena aceptación y críticas...

- **Coste de licencia**: Es una herramienta que cuenta con una licencia a perpetuidad bajo un coste determinado**,** tiene licencia por el número de usuarios que acceden a ella**,** es una herramienta de Software Libre ...
- **Seguridad, actualizaciones y parches**: La herramienta cuenta con actualizaciones de seguridad periódicas; no ha habido fallos de seguridad graves en las últimas versiones; cuando se detectan fallos las actualizaciones aparecen de manera rápida y efectiva; las actualizaciones y/o parches son gratuitos o de pago...
- **Coste de mantenimiento**: Existe un coste asociado al mantenimiento de la aplicación, pero este puede ser por parte del sistema (realizar actualizaciones, aumento de recursos...) o por pago de licencias anuales, por versiones...
- **Posibilidades de personalización**: Existe la posibilidad de personalizar la herramienta; parametrizar opciones propias que se ajustan a la empresa; creación de módulos/plugins propios para mejorar/expandir la funcionalidad de la herramienta, ...
- **Conocimientos sobre la herramienta**: Dentro de la organización se cuenta con conocimientos acerca del uso/instalación/administración de la herramienta, debe ser subcontratado o existe la posiblidad de adquirir conocimiento mediante cursos o manuales.

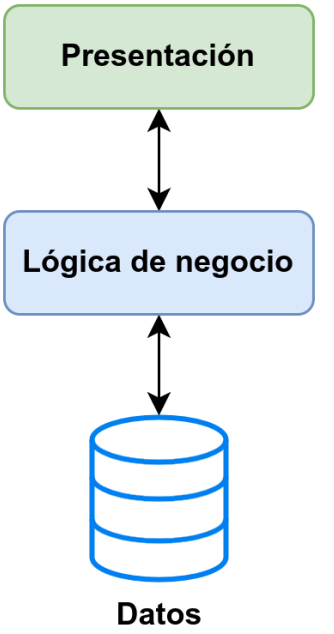
- **Sistema operativo**: Dependiendo del sistema de información elegido, se deberá instalar en un sistema operativo u otro. En este punto se pueden tener en cuenta también los puntos anteriores sobre el conocimiento para la toma de decisiones.
- **Método de instalación**: Hoy en día existen distintas posibilidades a la hora de instalar servicios, por lo que es importante realizar una buena decisión:
  - **Tradicional**: Vamos a llamar sistema tradicional a aquel que se realiza mediante un instalador que realiza la instalación en el sistema operativo, que no suele dar demasiadas opciones de configuración durante el proceso.
  - **Contenedores**: Hoy día existen servicios que podemos instalar a través de sistemas de contenedores (como puede ser Docker), los cuales suelen facilitar la instalación, así como la posibilidad de que también sea un sistema multicapa.
  - **Por capas**: La instalación multicapa puede resultar un poco más compleja y la aplicación/servicio debe poder permitir realizarlo. Aunque inicialmente pueda suponer un poco más de esfuerzo, pero a la larga puede suponer una gran ventaja como es la **alta disponibilidad**.

    Pasar de un sistema “monolítico” a un sistema por capas es posible, pero una vez más dependeremos de la aplicación. Por otro lado, si desde el inicio se ha creado un sistema multicapa, escalarlo será más sencillo que realizar la migración cuando ya esté en uso.

2.1. Arquitectura multicapa

Un sistema informático multicapa es aquel que hace uso de una arquitectura **cliente-servidor** en las que existe una separación lógica y/o física entre las distintas funciones que tiene una aplicación o servicio.

Normalmente se suele representar como una arquitectura en tres niveles, siendo estos:



- **Capa de presentación**: Es la que ve el usuario (también se la denomina «capa de usuario»), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato).

  También es conocida como interfaz gráfica y debe tener la característica de ser «amigable» (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

  Hoy en día lo habitual es que hagamos uso de servicios web, por lo que la capa de presentación es **la web que estamos visualizando**. En el caso de aplicaciones móviles, es **la propia aplicación que tenemos instalada en el dispositivo**.

- **Capa de negocio**: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse.

  Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

  En este tipo de arquitecturas, esta capa es la que se denomina ***backend***, y lo habitual es que sea un sistema al que llamamos a través de una API (del inglés, *application programming interface*, o interfaz de programación de aplicaciones).

- **Capa de datos**: es donde residen los datos y es la encargada de acceder a los mismos. Está formada

por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

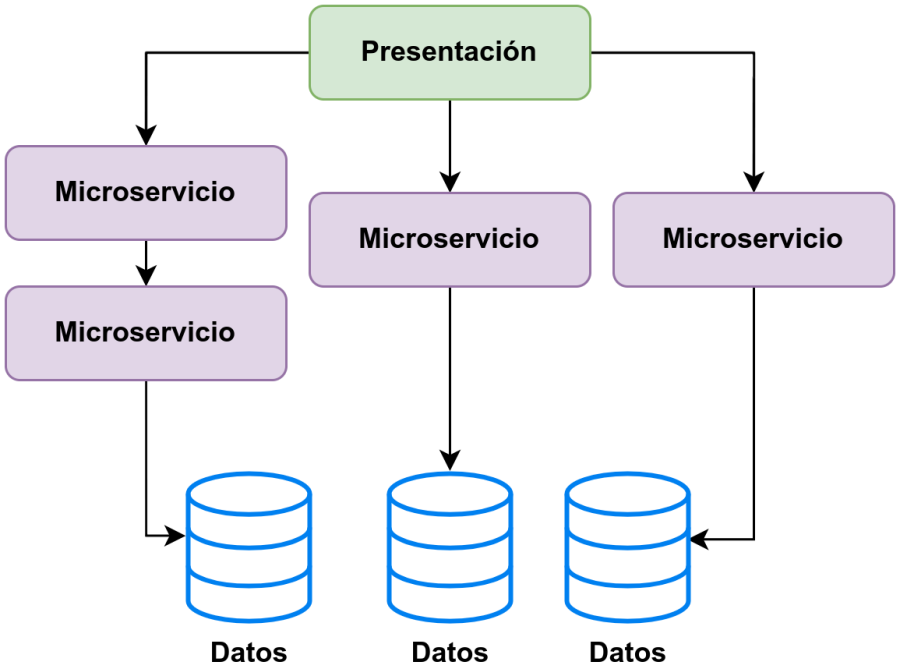
Información

Las aplicaciones web se pueden separar en dos capas: aplicación y base de datos.

En aplicaciones web es posible crear una arquitectura en capas aunque la aplicación no esté 100 % pensado para ello: la propia aplicación y la base de datos.

2.2. Arquitectura de microservicios

Para poder crear una arquitectura de microservicios el enfoque debe darse desde el primer momento del desarrollo de software. Es decir, antes de realizar ningún tipo de programación la aplicación se planteará como pequeños servicios que podrán interactuar entre sí.



Cada servicio se encargará de implementar una única funcionalidad. En caso de necesitar alguna característica que se repita en varios, se debería crear un microservicio que proporcione dicha característica o funcionalidad.

Información

Podríamos comparar una arquitectura de microservicios como una librería de programación, en la que existen funciones que realizan una única función.

Cada microservicio se desplegará de manera independiente, e incluso cada uno podrá estar programado en distintos lenguajes de programación. De esta manera **se puede hacer uso del lenguaje y la tecnología más adecuada para cada funcionalidad**.

3. Escalabilidad

Teniendo en cuenta todo lo dicho hasta ahora, cuando un sistema empieza a tener problemas de rendimiento deberemos abordar el problema y plantearnos cómo solucionarlo. De no hacerlo, se corre el peligro de que el servicio se vea interrumpido y por tanto perder tiempo de trabajo.

Antes de realizar ninguna modificación habría que analizar qué es lo que está sucediendo (para ello es importante tener un buen sistema de monitorización), y de esta manera saber en qué punto existe el problema y así poder solucionarlo.

Dependiendo de las decisiones tomadas durante la instalación, y tras lo visto previamente, podremos abordarlo de dos maneras diferentes.

Información

"Escalabilidad" no existe en el diccionario de la RAE, pero se usa como anglicismo de la palabra *scalability*.

3.1. Escalado vertical

Cuando se escala verticalmente un sistema lo que se va a realizar es **añadir más recursos al nodo que está teniendo problemas**. Tras el análisis previo realizado se añadirán los recursos necesarios (más RAM, discos duros más rápidos, aumentar el número de procesadores/cores).

Comúnmente también se dice "meter más hierro", porque antiguamente lo que se hacía era incrementar los recursos hardware del sistema. Hoy en día en sistemas virtualizados, estos recursos se pueden modificar, dependiendo del virtualizador, en "caliente", por lo que no sería necesario reiniciar el servicio.

Es el sistema más simple, ya que incrementando los recursos se espera que el problema se apacigüe o desaparezca, aunque esto no tiene por qué ser siempre así.

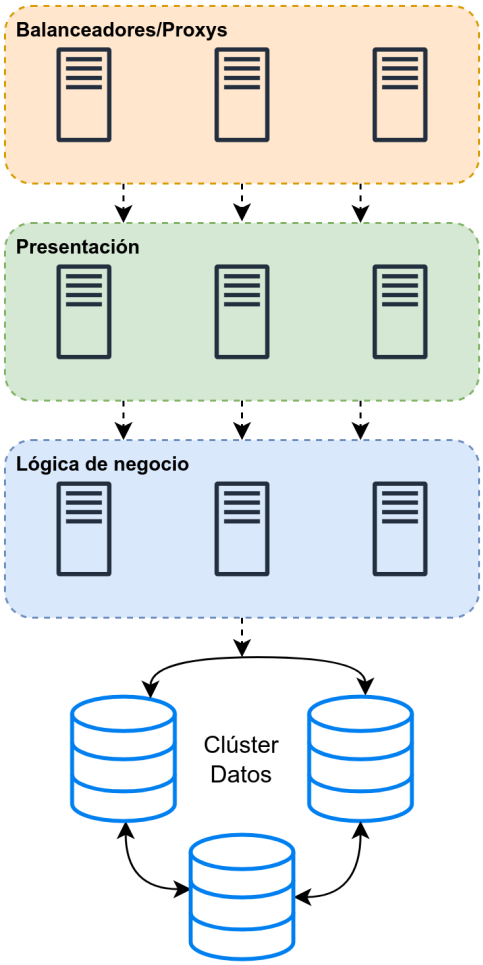
3.2. Escalado horizontal

El escalado horizontal trata de solventar el problema repartiendo la carga entre más nodos. Este proceso de escalado es más complejo y dependerá de la modularidad del servicio ofrecido. Es decir, la lógica de la aplicación debería haberse pensado desde el inicio para un sistema que escalará de manera horizontal en el futuro.

En principio, al realizar un escalado horizontal no existe una limitación de crecimiento, ya que siempre que se pueda repartir la carga entre servidores, no importará el número de ellos.

Dentro del escalado horizontal podríamos diferenciar dos tipos:

- **Escalado horizontal por capas**: Teniendo en cuenta lo visto previamente acerca de la arquitectura multicapa, en este modelo lo que se conseguirá es escalar cada capa de manera independiente. Podría realizarse de la siguiente manera:



- **Servidores frontales** (proxys) reciben las peticiones de la aplicación o del navegador y que balancean la carga y la mandan a servidores con la capa de presentación.
- Servidores con la **capa de presentación** que realizan las peticiones de negocio a los servidores correspondientes.
- Servidores de **lógica de negocio** que procesan las peticiones y piden los datos a un clúster de bases de datos.
- **Clúster de base de datos**.

- **Escalado horizontal de microservicios**: En el caso del escalado de microservicios, será similar al escalado horizontal, pero en este caso sólo se escalarán los microservicios necesarios.

  Debido a que todo es mucho más modular, es posible que quizá sólo sea necesario realizar el escalado de algunos microservicios, que quizá sean los más utilizados o los que requieren más tiempo de ejecución.

VII

Anexos

1. Administración remota

En informática no siempre tenemos los equipos que administramos en nuestra oficina. Pueden estar en otro edificio, la oficina de un cliente, en internet … por lo tanto no siempre es posible acceder de manera física a ellos, y por tanto entra en juego la **administración remota**.

Podemos definir la administración remota como el sistema que nos permite realizar ciertas acciones "lanzadas" desde nuestro equipo local pero que serán ejecutadas en un equipo remoto.

Se pueden diferenciar varios tipos de sistemas dentro de la administración remota, pero nos vamos a centrar en los siguientes:

- **Cliente remoto**: Lanzamos una acción a ejecutar desde un equipo remoto a través de algún tipo de interfaz o comando (que viajará a través de un protocolo securizado) y esperaremos a la respuesta.
- **Acceso remoto**: En este caso lo que hacemos es conectarnos al equipo a través de un protocolo que nos va a permitir administrarlo como si estuviésemos delante de él.

Todos estos sistemas pueden ser complementarios, y puede que podamos administrar un mismo servicio de todas estas maneras, por lo que queda a nuestra disposición elegir el mejor método en cada momento.

Por otro lado, dependiendo de qué tipo de administración vayamos a llevar a cabo, o el protocolo que utilice, tendremos que tener acceso al servidor de alguna manera (ya sea conexión directa o mediante VPN).

Información

Dependiendo de la administración remota que realicemos, necesitaremos conexión directa o mediante VPN al equipo que nos queremos conectar.

Por último, también debemos de conocer el tipo de protocolo que vamos a utilizar al realizar la conexión remota y por dónde va a pasar esa comunicación. Siempre hay que premiar la seguridad de la comunicación, y más cuando esta puede pasar por redes no controladas. Por lo tanto, deberemos asegurar que el protocolo utilizado es seguro, o en caso contrario, securizarlo de alguna manera.

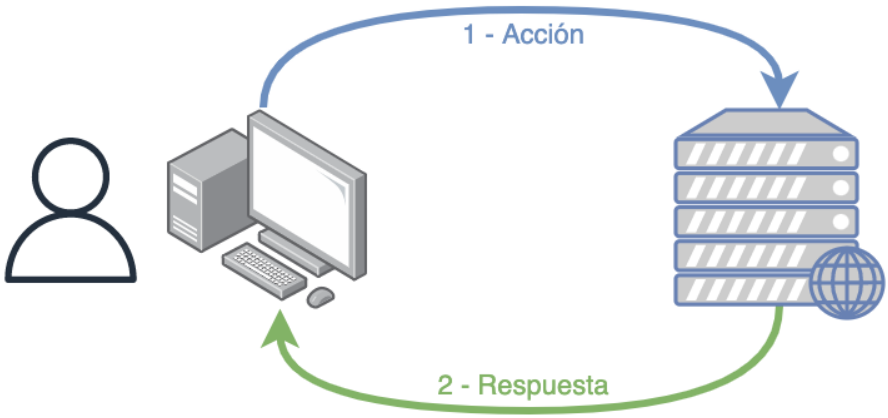
¡Cuidado!

Siempre debemos confirmar que la comunicación que se realiza para la administración remota viaja cifrada.

Más adelante veremos cómo securizar una comunicación no segura realizando un túnel mediante el protocolo SSH en entornos GNU/Linux.

1.1. Cliente remoto

Este sistema de administración permite enviar acciones al equipo remoto a través de un protocolo establecido, y dependiendo de la acción ejecutada se esperará una respuesta o no.



Hoy en día es muy habitual este tipo de sistemas a través de disintos **CLI** (*client line interface*) o **GUI** (*graphic user interface*) que nos permiten administrar servicios remotos. Por ejemplo:

- **https://www.mysql.com/MySQL**: El sistema gestor de base de datos MySQL cuenta con un cliente para realizar la conexión, ya sea desde el propio equipo o desde un equipo remoto.

  Este cliente se puede ejecutar desde línea de comandos, aunque también viene integrado en distintos interfaces gráficos como MySQL Benchmark, Dbeaver, ...

- **AWS CLI**: Es el interfaz de línea de comandos para poder administrar de manera remota los servicios contratados en la nube AWS de Amazon.

- **Gcloud CLI**: Similar al caso anterior pero esta vez para Google Cloud.

- **Remote Server Administration Tools for Windows 10**: En este caso se trata de un interfaz gráfico (**GUI**) que nos permite administrar un Windows Server desde un equipo Windows 10.

Antes de poder realizar la conexión remota con el cliente **debemos configurar un sistema de autenticación** para que el servicio remoto acepte las peticiones enviadas. En algunos casos será usando unos sistemas de certificados y en otros introduciendo un usuario y contraseña que establecerá una sesión temporal.

En el caso de AWScli y GCloud no nos estamos conectando directamente a nuestros servidores alojados en esas nubes, si no que lanzamos la orden a un "proxy" que verificará nuestros credenciales, verá los permisos que tenemos y después realizará la acción solicitada.

1.2. Acceso remoto

Este sistema permite acceder al sistema y podremos administrarlo como si nos encontrásemos delante. Dependiendo del sistema la conexión nos permitirá interactuar de alguna de las siguientes maneras:

- **CLI**: Mediante una conexión de línea de comandos. Es el caso más habitual en servidores GNU/Linux y la conexión se hace a través del protocolo seguro **SSH**.

- **GUI**: Podremos obtener un interfaz gráfico con el que veremos lo que está ocurriendo en pantalla en ese momento. En este caso, dependiendo del sistema, existirán distintas opciones, pero vamos a nombrar dos de ellas:

- **RDP**: Es el protocolo de escritorio remoto de Microsoft que transmite la información gráfica que el usuario debería ver por la pantalla, la transforma en el formato propio del protocolo, y la envía al cliente conectado. El problema es que este sistema desconecta al usuario que está logueado para poder hacer uso del escritorio remoto.
- **VNC**: En inglés *Virtual Network Computing*, es un servicio con estructura **cliente-servidor** que permite visualizar el escritorio del servidor desde un programa cliente. En este caso, no existe desconexión del usuario que está logueado y por tanto podrá ver lo que le están realizando de manera remota.

  Es muy habitual que los equipos de usuarios ya tengan la instalación del servidor hecha, para que de esta manera, en caso de incidencia, poder realizar la conexión remota sin que el usuario tenga que realizar ninguna acción.

A continuación se va a detallar algunos de los métodos mencionados.

1.3. SSH

SSH es un protocolo de comunicación segura mediante cifrado cuya función principal es el acceso remoto a un servidor. La arquitectura que utiliza es la de cliente-servidor.

Aunque el uso más habitual de SSH es el acceso remoto, también se puede utilizar para:

- Securizar protocolos no seguros mediante la realización de túneles.
- Acceder a un equipo saltando a través de otro.

Estas funcionalidades las veremos más adelante.

1.3.1. Servidor SSH

En el servidor al que nos queramos conectar deberá estar instalado el demonio/servicio SSH, conocido como **sshd**. Es habitual que ya esté instalado en sistemas GNU/Linux, pero de no ser así deberemos usar el sistema de paquetes de nuestra distribución para hacer la instalación. El nombre suele ser **openssh-server**.

Este servicio por defecto se pondrá a la escucha en el puerto 22/TCP:

SSHd escuchando en puerto 22

```
ruben@vega:~$ sudo ss -pntaln
State   Recv-Q   Send-Q   Local Address:Port   Peer Address:Port   Process
LISTEN  0        128      0.0.0.0:22           0.0.0.0:*           users:(("sshd",pid=1122,fd=3))
LISTEN  0        128      [::]:22              [::]:*              users:(("sshd",pid=1122,fd=4))
```

La configuración del servicio se realiza a través de un fichero de configuración que está situado en la ruta `/etc/ssh/sshd\_config`. Las distribuciones de GNU/Linux ya traen una configuración predeterminada que suele constar de las siguientes directivas (aunque hay muchas más):

- **Port**: Normalmente viene comentada, ya que el puerto por defecto es el 22. En caso de querer cambiar el puerto, podremos modificar esta línea, asegurando que no esté comentada.

- **ListenAddress**: Por defecto SSH se pondrá a la escucha en todos los interfaces que tengamos configurados. Si sólo nos interesa escuchar en alguna de las IPs que tengamos configuradas, deberemos modificar esta configuración.
- **PermitRootLogin**: Para evitar problemas de seguridad, esta directiva suele estar configurada a "**No**", para evitar que se puedan usar los credenciales de root para hacer el login.
- **PubkeyAuthentication**: Esta directiva permite realizar la conexión a través de unas claves públicas/privadas que podemos crear. Se explicará más adelante.

Hoy día también se puede instalar en Windows 10 y posteriores a través de un comando, siendo administrador de PowerShell:

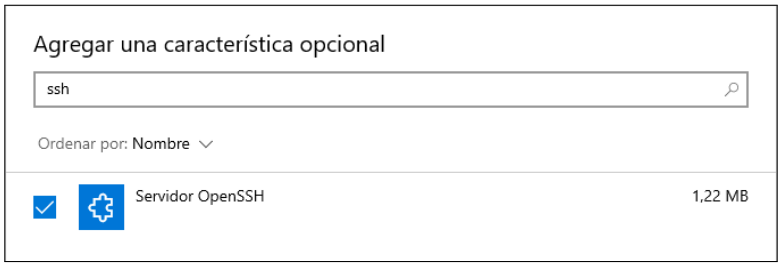
Instalando OpenSSH Server en Windows 10

```
PS C:\Windows\System32> Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0

PS C:\Windows\System32> Start-Service sshd

PS C:\Windows\System32> Set-Service -Name sshd -StartupType 'Automatic'
```

O desde el interfaz gráfico a través de las "**Características opcionales**", buscando por ssh:



1.3.2. Cliente SSH

El cliente SSH es aquel programa que a través del protocolo SSH se puede conectar a un servidor SSH. Existen distintos tipos de clientes que podemos utilizar:

- **CLI**: El cliente de consola es el más habitual. Está instalado de forma habitual en todas las distribuciones de GNU/Linux (normalmente el paquete se llama **openssh-client**). También lo encontramos instalado por defecto en MacOS.

Hoy día también está instalado en Windows 10, y de no estar, se puede instalar a través de las "**Características Opcionales**".

- **GUI**: Existen distintos interfaces gráficos que nos puede interesar utilizar:
  - **Putty**: Un cliente muy habitual en entornos Windows.
  - **Kitty**: Una versión mejorada del anterior.
  - **Termius**: Cuenta con versión de escritorio y móvil.

Para realizar la conexión al servidor SSH debemos conocer:

- **Dirección del servidor**: Ya sea mediante IP o nombre FQDN (*fully qualified domain name*) que se resuelva.
- **Puerto**: Ya hemos comentado que por defecto el puerto es 22.
- **Usuario**: Para realizar el sistema de autenticación, necesitamos un usuario que exista en el sistema.
- **Contraseña**: Los credenciales de acceso del usuario.

Para realizar la conexión desde un cliente de consola ejecutaremos:

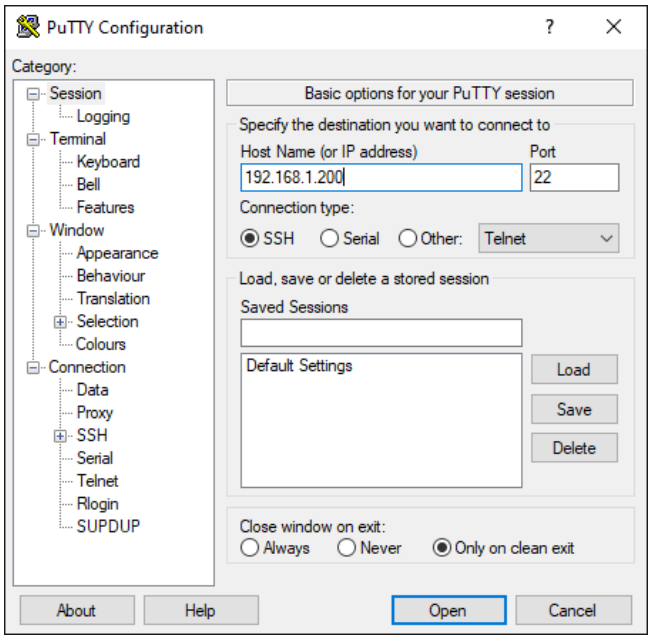
Conexión SSH

```
ruben@vega:~$  ssh usuario@192.168.1.200 -p 22
```

En el comando anterior podemos identificar:

- **ssh**: el cliente de consola
- **usuario**: el nombre del usuario con el que nos queremos conectar al servidor remoto.
- **@**: la arroba en inglés significa “at”, que indica “usuario en el servidor X”.
- **192.168.1.200**: La IP del servidor al que nos queremos conectar.
- **-p 22**: Estos dos parámetros van juntos, “-p” indica que vamos a indicar el puerto de conexión y “22” que nos queremos conectar a ese puerto. Debido a que 22 es el puerto por defecto, podríamos no poner estas opciones si sabemos que el servidor escucha en el puerto 22.

Si realizamos la conexión a través de un cliente de interfaz, como es putty, el aspecto será el siguiente, donde sólo podremos introducir la IP del servidor. Cuando se comience con la conexión nos pedirá los credenciales de acceso.



Si es la primera vez que nos conectamos a un servidor mediante SSH nos saldrá un mensaje como el siguiente:

Conexión SSH

```
ruben@vega:~$  ssh usuario@192.168.1.200 -p 22

The authenticity of host '192.168.1.200 (192.168.1.200)' can't be established.
ECDSA key fingerprint is SHA256:uK9MOl0gLDhTtCrlcafc1zVObVA/vnOMn6TWFsQb23o.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Este "key fingerprint" es un identificador que está relacionado con el fichero de "**clave pública**" del servidor. Es como el DNI del servidor. La primera vez se nos guarda ese *fingerprint*, y en caso de que en una próxima conexión varíe, nos avisará. No suele ser habitual que este identificador cambie.

1.3.3. Conexión mediante certificados de clave pública/clave privada

Existe una alternativa a la hora de realizar una conexión SSH para que no nos pida la contraseña del usuario, y es **hacer uso de los certificados de clave pública y clave privada**. Este concepto de "clave pública y clave privada" viene de la **criptografía asimétrica**.

Este sistema de criptografía asimétrica hace uso de dos claves que están asociadas entre sí:

- **Clave privada**: Es la base del sistema criptográfico, y como su nombre indica, se debe de mantener en privado. **Nunca se debe de compartir**, ya que entonces se podrían hacer pasar por nosotros.
- **Clave pública**: Asociada a la clave privada, la clave pública puede ser compartida y enviada a otros ordenadores para poder realizar la conexión.

Para generar el par de claves se realiza con el siguiente comando:

Crear par de claves pública/privada

```
ruben@vega:~$  ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ruben/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):
Enter same passphrase again:

Your identification has been saved in /home/ruben/.ssh/id_rsa
Your public key has been saved in /home/ruben/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:SPqPOYBmPb8PCFhcZgqcWZPZzaL5RNfMeKmHqebvC7E ruben@vega
The key's randomart image is:
+---[RSA 3072]----+
|o +oB o = .      |
| * B.+ = *       |
```

```
|  + + + =        |
| o o + = .       |
|. .o+.o S        |
|  +.+*o          |
| o  +Eo          |
|     .+=         |
|      *B+        |
+----[SHA256]-----+
```

El comando muestra los siguientes pasos:

1. Creación de la pareja de claves pública/privada haciendo uso del sistema criptográfico **RSA**.
2. Lugar donde se va a guardar la clave privada. Por defecto en `\~/.ssh/id\_rsa`.
3. Contraseña para securizar la clave privada. De esta manera, para poder usarla habrá que introducir dicha contraseña. Dado que nosotros queremos evitar introducir contraseñas, lo dejaremos en blanco.
4. Lugar donde se va a guardar la clave pública. Por defecto en `\~/.ssh/id\_rsa.pub`

Una vez tenemos nuestro par de claves, podemos copiar la clave pública al usuario del servidor que nos interese mediante el siguiente comando:

Crear par de claves pública/privada

```
ruben@vega:~$  ssh-copy-id user@servidor_remoto
```

Para ello es imprescindible conocer previamente la contraseña del usuario en el servidor. El comando `ssh-copy-id` realizará una conexión SSH y copiará el contenido de la **clave pública**, `\~/.ssh/id\_rsa.pub`, dentro del fichero `\~/.ssh/authorized\_keys` del usuario en el servidor remoto. Este paso se puede realizar a mano (con un editor de texto).

¡Cuidado!

Windows no tiene el comando "ssh-copy-id", por lo que deberemos hacer el paso a mano, tal como se ha explicado.

Al realizar la siguiente conexión, ya no necesitaremos introducir la contraseña del usuario, ya que el sistema remoto podrá autenticarnos a través del sistema clave pública/privada.