

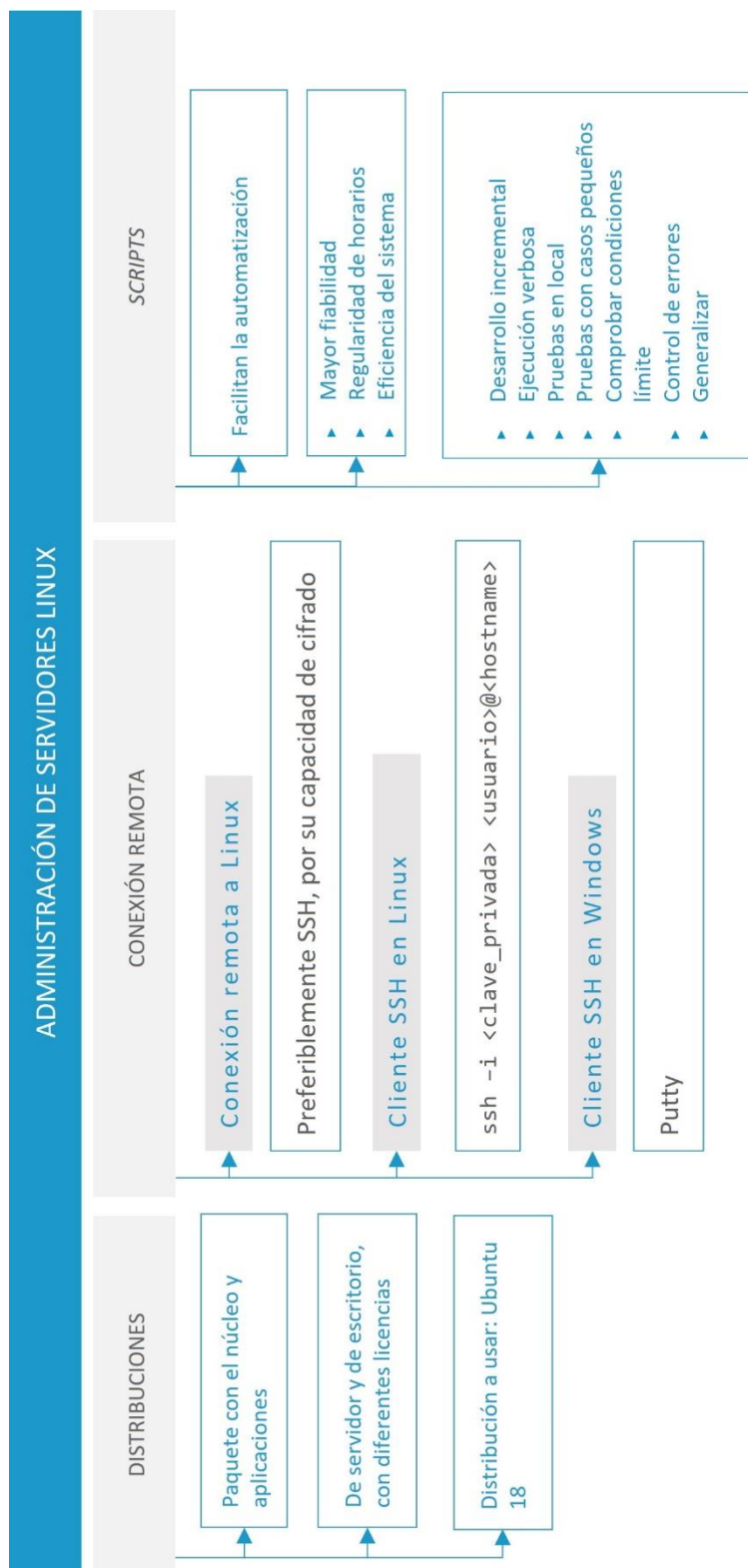
Administración de Sistemas en la Cloud

Administración de servidores Linux

Índice

Esquema	3
Ideas clave	4
2.1. Introducción y objetivos	4
2.2. Distribuciones	4
2.3. Acceso remoto	5
2.4. Servidor SSH	7
2.5. Cliente SSH	8
2.6. SSH con clave privada	14
2.7. Automatización de instalación y configuración	17
2.8. Acceso remoto en la nube	19
2.9. Referencias bibliográficas	26
A fondo	28
Test	29

Esquema



2.1. Introducción y objetivos

Este tema presentará el concepto de distribución y cubrirá en detalle el acceso remoto a Linux.

Los **objetivos** que se pretenden conseguir son:

- ▶ Entender el concepto de distribución y su relación con el sistema operativo Linux.
- ▶ Conocer las diferentes formas de acceso remoto que ofrece Linux.
- ▶ Entender los conceptos básicos de automatización e instalación.

2.2. Distribuciones

Una distribución de Linux es un paquete que consiste en el **núcleo del sistema operativo** y una **colección de paquetes y aplicaciones** (Dulaney, 2018). Dada una misma versión del núcleo, una aplicación escrita para Linux debería ser ejecutable en cualquier distribución con ese núcleo. No obstante, dado que la colección de aplicaciones abarca desde el instalador al entorno gráfico y el gestor de paquetes, la manera de interactuar con cada distribución puede ser muy diferente. La licencia también varía de una distribución a otra: aunque Linux es de código abierto, hay distribuciones comerciales que se ofrecen con contratos de soporte. Compañías como SUSE o RedHat no cobran por el sistema operativo, sino por el soporte que ofrecen a sus clientes.

Algunas distribuciones están enfocadas al usuario final. A tal efecto, incluyen entorno gráfico, aplicaciones ofimáticas, reproductores de audio y vídeo, etc. Aquí nos

centraremos en **distribuciones orientadas a servidores**. Algunas de ellas, como Ubuntu, mantienen ediciones de escritorio y de servidor para una misma versión de la distribución.

Entre las aplicaciones incluidas en casi todas las distribuciones, están las **utilidades GNU**. Hay utilidades de manipulación de ficheros y del propio contenido de los ficheros, de información del sistema operativo, de la sesión, de búsqueda, editores de texto, etc. Estas utilidades son esenciales a la hora de administrar servidores en la línea de comandos y para automatizar tareas.

Los ejemplos y ejercicios que utilizaremos asumirán que se usa una instalación de [Ubuntu 18.04](#), salvo que se indique lo contrario. Es una distribución basada en [Debian](#), que a su vez es una de las distribuciones más utilizadas como base para otras. Ubuntu es muy habitual en entornos de escritorio y relativamente habitual en entornos de servidor. Otras distribuciones muy habituales en entornos corporativos son SUSE, [RedHat Enterprise Linux](#) (RHEL) y su variante gratuita [CentOS](#). AWS (Amazon Web Services) ofrece [Amazon Linux 2](#) como imagen base para instancias Linux, así como imágenes para VirtualBox y otros entornos de virtualización para pruebas en local.

2.3. Acceso remoto

Uno de los primeros aspectos a considerar para utilizar un servidor en la nube es el acceso remoto al mismo. Al tratarse de un servidor en la nube, será necesario contar con las **herramientas necesarias** que lo permitan:

- ▶ Telnet.
- ▶ SSH (Secure Shell).
- ▶ Otros.

Telnet

Telnet es un protocolo de capa de aplicación, utilizado en Internet o redes de área local, para proporcionar acceso remoto mediante una conexión de terminal virtual. La utilización de Telnet está desaconsejada, porque el tráfico no se transmite cifrado. Este hecho permitiría a un usuario malintencionado acceder al contenido del tráfico, si fuera capaz de interceptarlo. Para poder utilizar Telnet, es necesario que el servidor lo tenga activado y contar con un cliente de Telnet, que habitualmente está incluido en todos los sistemas operativos.

SSH

Secure Shell (SSH) es un protocolo de red de cifrado para servicios de red de operación segura a través de una red no segura. Su aplicación más habitual es la de inicio de sesión remoto a servidores Linux. SSH proporciona un canal seguro mediante una arquitectura cliente-servidor. Los sistemas operativos Linux cuentan habitualmente con un cliente de SSH. En el caso de los clientes Windows, el cliente más utilizado es el llamado [Putty](#). Se trata de un cliente de SSH de código abierto y libre (Van Vugt, 2015).

SSH es la forma más habitual de acceder a los servidores, tanto aquellos alojados en la nube como servidores tradicionales en un centro de datos propio. SSH soporta el acceso con usuario y *password* y el uso de certificados (también llamados pareja de claves o *key pair*). En muchos de los proveedores de nube pública, el uso de certificados es obligatorio.

Otros

Existen otras formas de acceder a los servidores de forma remota, la mayoría de ellas específicas para entornos gráficos, por lo que serán menos usadas a nivel de automatización y administración. **Algunas** de ellas son:

- ▶ **VNC.** Virtual Network Computing (VNC) es un sistema de control remoto que utiliza el protocolo *remote framebuffer* (RFB) para controlar de forma remota otro ordenador. Transmite los eventos de teclado y ratón de un ordenador a otro y las actualizaciones de pantalla de vuelta en la otra dirección. VNC es independiente de la plataforma, por lo que puede usarse para conectarse también a equipos Windows. El código fuente VNC original y muchos derivados modernos son de código abierto, bajo la licencia pública general, GNU por sus siglas en inglés.
- ▶ **TeamViewer.** Se trata de un desarrollo comercial y propietario, pero es posible usarlo de forma gratuita en una de sus ediciones. Puede ser útil en ciertos casos para acceder a equipos de sobremesa para soporte remoto o como sustituto de una VPN, pero rara vez se usa para tareas de administración.
- ▶ **Logmein.** Se trata de un *software* SaaS que permite la gestión, acceso y monitorización de la PC (*personal computer*) o servidores de forma multiplataforma y mediante su interfaz web.

2.4. Servidor SSH

En este apartado, se detallará paso a paso **cómo conectarse a un servidor Linux mediante SSH**. El sistema operativo de una instancia en la nube estará preparado para aceptar conexiones SSH por defecto. Sin embargo, en un entorno local, es posible que sea necesario activar el servicio SSH. Por ejemplo, en una instalación de Ubuntu 18 en una máquina virtual, SSH no está instalado por defecto.

A continuación, en el vídeo *Instalación de Ubuntu y acceso remoto al servidor*, se podrá ver una demostración de instalación de Ubuntu 18 Server y cómo es el acceso al servidor.



Accede al vídeo

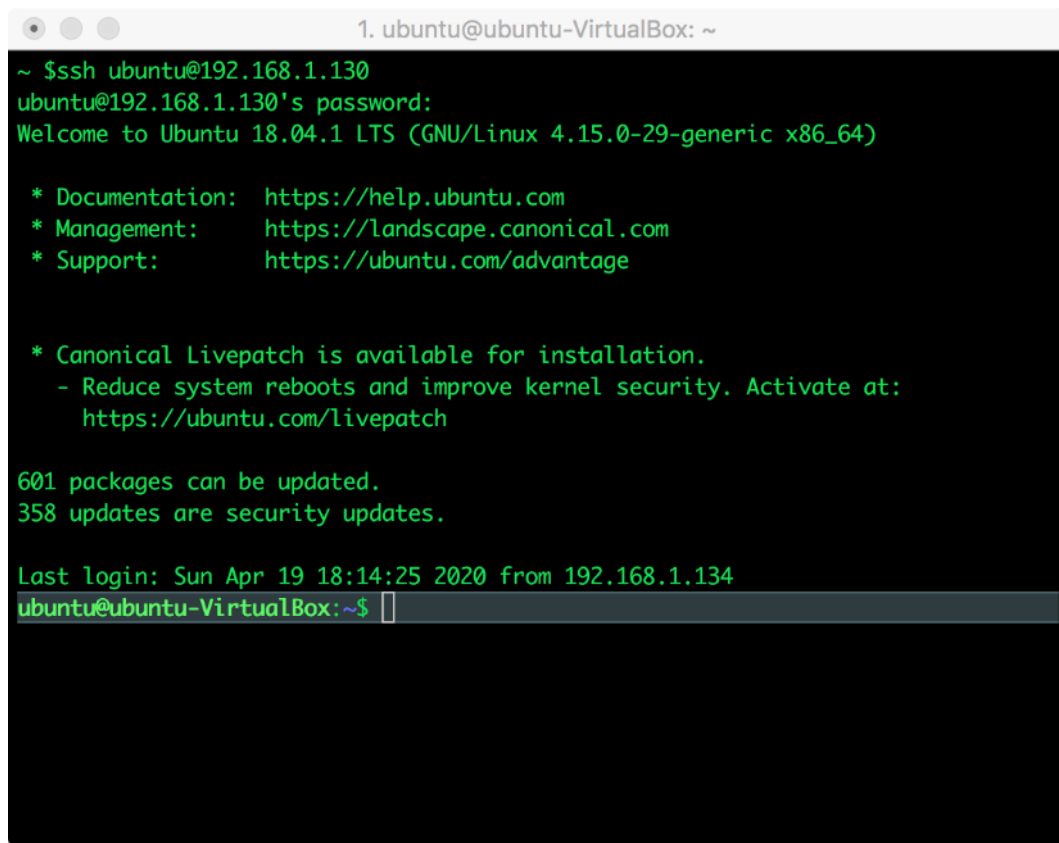
Para instalarlo, es necesario seguir estos **pasos**:

- ▶ Ejecutar `sudo apt-get update && sudo apt-get install -y openssh-server`.
- ▶ Asegurarse de que el servicio esté arrancado con `sudo service ssh start`.
- ▶ Editar el archivo de configuración con `sudo vim /etc/ssh/sshd_config` o con cualquier otro editor de texto. El fichero debe contener al menos las siguientes opciones. Estas opciones no son las más seguras, pero servirán para poder hacer pruebas en entornos locales:
 - Port 22.
 - ListenAddress 0.0.0.0.
 - PasswordAuthentication yes.
- ▶ Recargar la configuración con `sudo service sshd reload`.
- ▶ Averiguar la IP (*internet protocol*) del equipo con `ip address`.

En este punto, ya sería posible usar un cliente para conectarse y, en un entorno local, no será necesario más configuración. En un entorno de nube, será necesario haber habilitado el puerto 22 en el grupo de seguridad de la instancia.

2.5. Cliente SSH

Asumiendo que el servidor SSH ha sido configurado correctamente, se podría usar el cliente SSH desde un equipo Linux o MacOS con SSH `<nombre_usuario>@<ip_servidor>`, por ejemplo: `ssh ubuntu@192.168.1.130`.

A terminal window titled '1. ubuntu@ubuntu-VirtualBox: ~' showing an SSH session. The user has successfully logged into an Ubuntu 18.04.1 LTS system. The terminal displays the Ubuntu welcome message, links for documentation, management, and support, information about Canonical Livepatch, and update statistics (601 packages can be updated, 358 are security updates). The last login was on Sun Apr 19 18:14:25 2020 from 192.168.1.134. The prompt is 'ubuntu@ubuntu-VirtualBox:~\$' with a cursor.

```
1. ubuntu@ubuntu-VirtualBox: ~  
~ $ssh ubuntu@192.168.1.130  
ubuntu@192.168.1.130's password:  
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-29-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
* Canonical Livepatch is available for installation.  
  - Reduce system reboots and improve kernel security. Activate at:  
    https://ubuntu.com/livepatch  
  
601 packages can be updated.  
358 updates are security updates.  
  
Last login: Sun Apr 19 18:14:25 2020 from 192.168.1.134  
ubuntu@ubuntu-VirtualBox:~$
```

Figura 1. Inicio de sesión en Ubuntu con SSH. Fuente: elaboración propia.

Para acceder desde un equipo Windows, se podría usar **Putty**. Es el cliente SSH recomendado para entornos Windows, está disponible para su descarga y es gratuito. Tras ejecutar Putty, se presentará la primera pantalla de configuración (ver Figura 2).

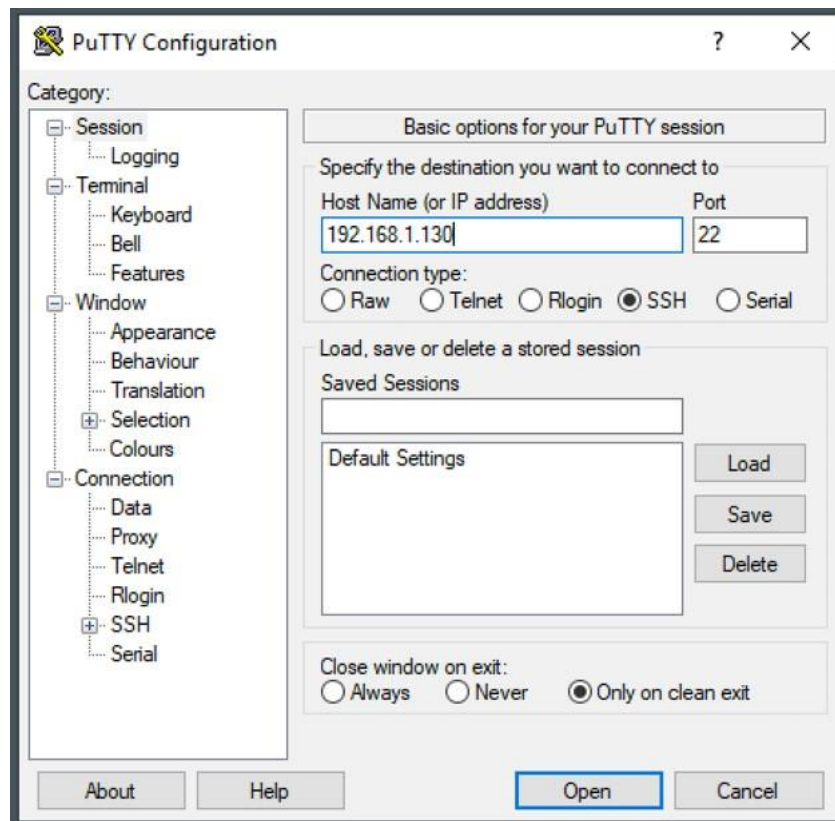


Figura 2. Pantalla de inicio de Putty. Fuente: elaboración propia.

En esta pantalla, será posible configurar la información para memorizar diferentes sesiones de conexión. En el caso más sencillo, será necesario marcar *connection type* a **SSH** e introducir la IP en el campo **Host Name (o IP address)**. Tras presionar el botón Open, se abrirá una ventana de terminal donde se solicitará el usuario y la contraseña para intentar el inicio de sesión. En el primer intento de conexión, aparecerá una advertencia que el equipo es desconocido. Una vez aceptado este aviso, no volverá a aparecer, a menos que se use el mismo nombre de equipo o la misma IP para conectarse a un equipo diferente.



Figura 3. Advertencia en Putty durante la primera conexión a un equipo. Fuente: elaboración propia.

Una vez introducido el usuario y la *password* correcta, Putty ofrecerá una consola en la máquina remota, como muestra la Figura 4.

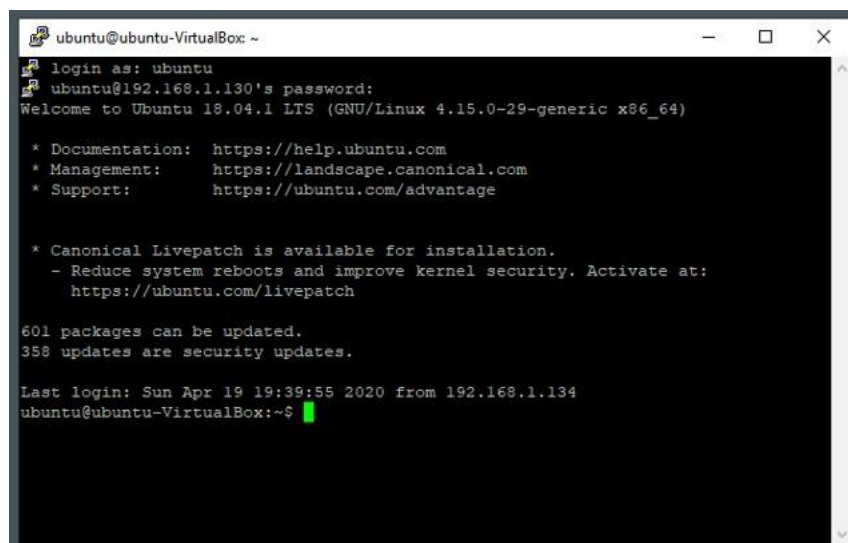


Figura 4. Sesión remota en Ubuntu con Putty. Fuente: elaboración propia.

Putty dispone de más opciones de configuración, accesibles desde el icono de la ventana (ver Figura 5).

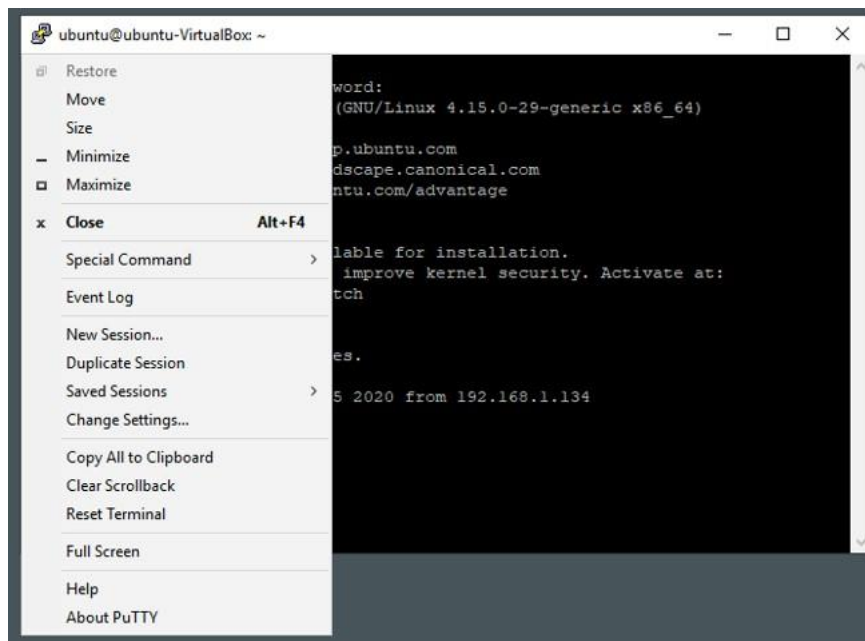


Figura 5. Menú contextual de Putty. Fuente: elaboración propia.

- ▶ **Special Command:** muestra una lista de los comandos más frecuentes en esa máquina, que pueden ser enviados directamente por Putty como una forma de ayudar al usuario a realizar tareas de forma más rápida.
- ▶ **Event Log:** registro de todas las operaciones realizadas en esa máquina durante la sesión de conexión. Esta opción resulta especialmente útil en procesos de automatización, donde será habitual el probar manualmente los comandos que instalan o configuran un componente *software*. Este registro puede usarse como primer borrador del *script*.
- ▶ **New Session:** permite realizar una nueva conexión a este u otro servidor diferente.
- ▶ **Duplicate Session:** permite abrir de forma automática una nueva sesión contra la misma máquina.
- ▶ **Saved Sessions:** permite acceder a las conexiones guardadas.
- ▶ **Change Settings:** esta opción da acceso al cambio de parámetros de la conexión actual.

- ▶ **Copy All to Clipboard:** esta funcionalidad copia todo el contenido del Buffer de la sesión actual al portapapeles. Esta opción es muy útil a la hora de generar *scripts* a partir de los resultados de una sesión interactiva.
- ▶ **Clear Scrollback:** limpia el contenido de la pantalla de texto.
- ▶ **Reset Terminal:** reinicia el terminal de texto, borrando la pantalla y limpiando el *buffer*.
- ▶ **Full Screen:** ejecuta Putty en modo pantalla completa.

Putty permite guardar los comandos tecleados en la consola y la salida de estos en un archivo de *log*. El fichero de salida se selecciona en la ventana de configuración.

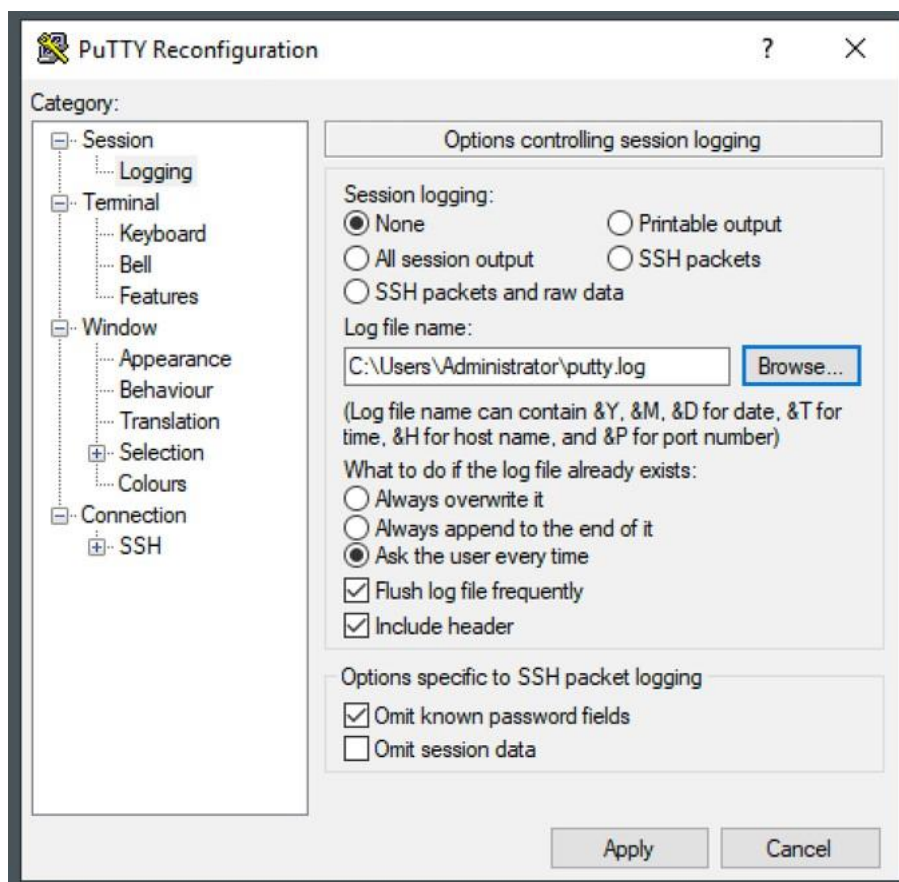


Figura 6. Configuración de *log* de Putty. Fuente: elaboración propia.

2.6. SSH con clave privada

Los ejemplos anteriores son útiles para pruebas locales, pero, en escenarios reales, es más seguro **no utilizar contraseñas para los inicios de sesión**. SSH soporta autenticación basada en claves: el usuario crea una pareja de claves, pública y privada, y copia la clave pública a aquellos servidores a los que se va a conectar. Al conectar, selecciona la clave privada en el cliente SSH. El servidor, al recibir la conexión, comprobará que la clave privada corresponde con la clave pública (Matotek et al., 2017).

Los proveedores de nube suelen ofrecer la opción de generar una **pareja de claves** y copiar la clave pública en los servidores. El usuario deberá guardar la clave privada y proporcionarla al cliente de SSH en cada conexión. Por ejemplo, si la clave privada está alojada en `key.pem` y la instancia ha sido desplegada en AWS con la imagen base de Amazon Linux, el comando sería parecido a:

```
ssh -i key.pem ec2-user@34.123.123.123
```

Es posible generar la pareja de **claves localmente** y copiar la clave pública a los servidores por algún otro método o proporcionarla al proveedor de nube antes de arrancar las instancias. En una máquina Linux, se podría generar la pareja de claves de la siguiente manera (Matotek et al., 2017):

```
ssh-keygen -t rsa -b 4096 -f./security_key
```

La clave pública se guardará en el fichero `security_key.pub` y la clave privada en `security_key`. Por defecto, las rutas serían `~/.ssh/id_rsa.pub` y `~/.ssh/id_rsa`. Además, el comando solicita una contraseña (o *passphrase*) con la que proteger el fichero de clave privada. A continuación, es necesario copiar la clave pública al equipo remoto con el siguiente comando:

```
ssh-copy-id -i ./security_key.pub <usuario>@<equipo_remoto>
```

A partir de ese momento, ya es posible conectarse indicando el fichero de clave privada. El cliente solicitará la clave del fichero, pero no hace falta indicar la contraseña del usuario:

```
ssh -i ./security_key <usuario>@<equipo_remoto>
```

También es posible generar y usar claves públicas y privadas en Windows con **Puttygen**, una utilidad disponible en la misma web que Putty. Haciendo *click* en Generate, se obtienen las claves que habrá que guardar en dos ficheros con Save public key y Save private key (ver Figura 7).

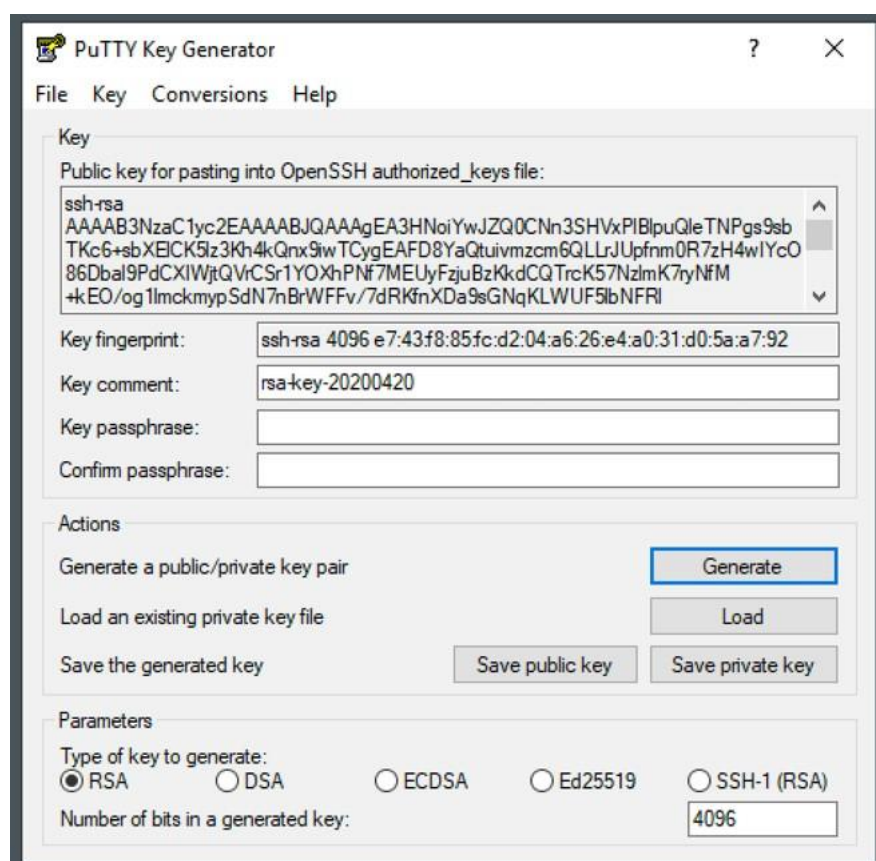


Figura 7. Interfaz de Puttygen. Fuente: elaboración propia.

Para copiar la clave pública, es necesario conectarse al equipo con contraseña, abrir el fichero ~/.ssh/authorized_keys y pegar en una línea nueva el contenido de la ventana de Puttygen que empieza por ssh-rsa. Por ejemplo, usando el editor nano, el fichero quedaría como muestra la Figura 8.

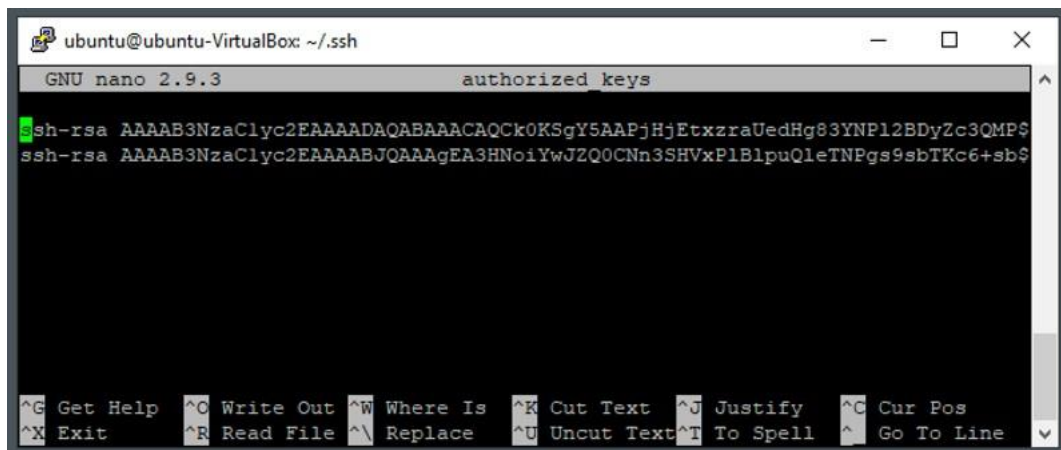


Figura 8. Edición de authorized_keys con nano. Fuente: elaboración propia.

La primera línea es la de la clave pública generada en el equipo Linux y que el comando `ssh-copy-id` se encargó de copiar automáticamente. La segunda ha sido copiada a mano desde Puttygen. El último paso es configurar Putty para usar el fichero de clave privada guardado anteriormente. Se especifica en **Connection > SSH > Auth > Private key file for authentication**.

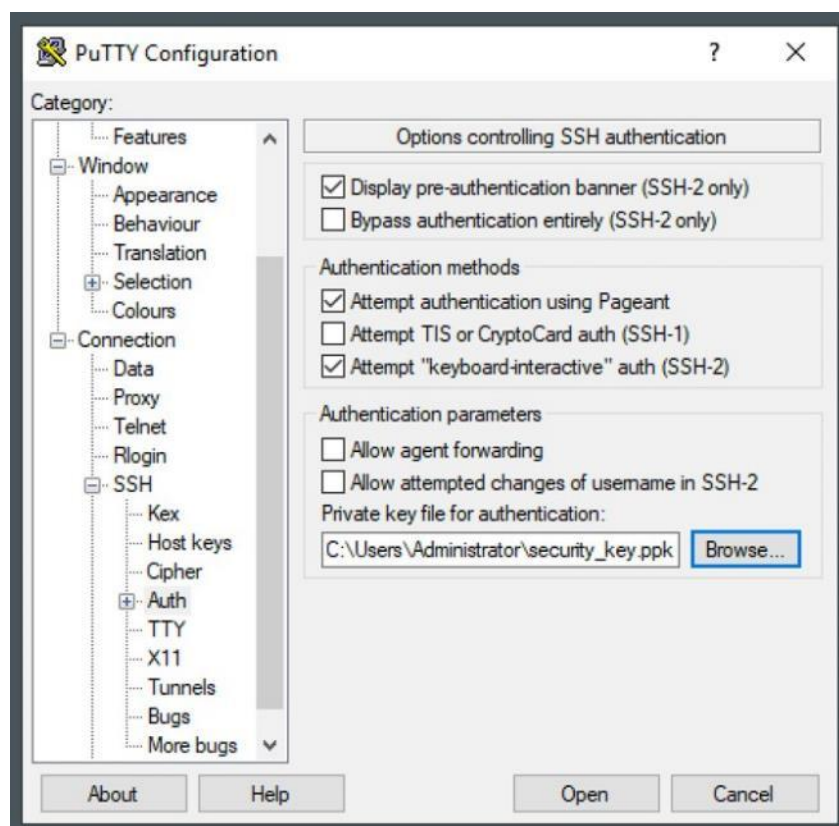


Figura 9. Selección de clave privada en Putty. Fuente: elaboración propia.

Los pasos que se han detallado hasta ahora pueden parecer tediosos, pero son fácilmente automatizables, bien en la línea de comandos o bien con herramientas de despliegue como Ansible.

2.7. Automatización de instalación y configuración

La automatización es fundamental para un administrador de sistemas. Aunque el perfil de un administrador no es el mismo que el de un programador, la capacidad de escribir *scripts* para **automatizar tareas** es una gran herramienta en el arsenal del *sysadmin* (administrador de sistemas).

Un *script* será una pieza de código que implementa una tarea concreta. Un *script* útil para un administrador estará escrito en un lenguaje interpretado y que esté disponible en el sistema de destino. Aquí se tratarán principalmente *scripts* en Bash, ya que esta *shell* está disponible en muchas distribuciones Linux, pero un *script* puede estar escrito en Perl, en Python o en cualquier otro lenguaje, siempre que el administrador esté cómodo con él.

En general, la automatización mediante *scripts* presenta las siguientes **ventajas** sobre la ejecución manual (Frisch, 2002):

- ▶ Mayor fiabilidad: una vez se ha comprobado que un *script* funciona bien, funcionará de la misma manera todas las veces.
- ▶ Regularidad de horarios: las tareas automatizadas pueden programarse en base a un horario concreto o a eventos externos; es decir, no es necesaria la presencia del administrador.
- ▶ Eficiencia del sistema: las tareas administrativas intensivas se pueden ejecutar en horarios de mejor carga del sistema.

A continuación, se indican algunas **estrategias** para escribir *scripts* útiles (Frisch, 2002):

- ▶ **Escribir el *script* incrementalmente.** Conviene escribir una versión sencilla del *script* e ir añadiendo funcionalidad poco a poco. Por ejemplo, la primera versión puede no aceptar argumentos en invocación del *script* para no añadir ruido al código de la funcionalidad que se quiere implementar.
- ▶ **Invocar los *scripts* en modo verboso.** Por ejemplo, los *scripts* de *shell* se pueden invocar con el parámetro `-v`, que muestra cada línea antes de ejecutarla.
- ▶ **Probar en archivos locales.** Antes de ejecutar el *script* en un entorno real, conviene probar con archivos de prueba.
- ▶ **Empezar a probar con casos pequeños.** Por ejemplo, si el *script* va a operar sobre una lista de elementos, conviene probar que la tarea sobre un elemento funciona correctamente antes de introducir la lógica del bucle.
- ▶ **No olvidar probar las situaciones límite.** Siguiendo con el caso anterior, conviene probar que el *script* puede operar con un elemento, con muchos elementos y que no falla si la lista es de cero elementos.
- ▶ **Asumir que algo va a ir mal.** Cuanto más control de errores incorpore el *script*, más estable será. El control de errores implica tanto detectarlos como la lógica necesaria para tratar el error: un mensaje de error legible para el usuario, deshacer las acciones llevadas a cabo hasta el momento, etc.
- ▶ **Generalizar.** Un *script* más general es más fácil de extender a otros casos, probablemente será más fácil de probar y, por tanto, más robusto.

2.8. Acceso remoto en la nube

Los servidores en la nube no se instalan, **se despliegan**. Las instancias, como generalmente se denominan en muchos proveedores, se pueden arrancar de diversas maneras: en una consola web, con una herramienta de línea de comandos o a través de aplicaciones propias que usen un SDK del proveedor. Esta versatilidad ofrece posibilidades a los administradores para automatizar los despliegues al máximo. Los siguientes apartados muestran un despliegue manual en AWS EC2 con el objetivo de familiarizarse con los conceptos de este servicio.

EC2, o Elastic Compute Cloud, permite que los usuarios alquilen recursos de computación por segundos a partir de máquinas virtuales en varios sistemas operativos. Estas instancias se pueden personalizar para ejecutar cualquier aplicación que el usuario necesite.

La consola de administración de AWS es la cara pública de EC2 y de todos los servicios de AWS. Para desplegar una nueva instancia en la consola, hay que abrir la sección de EC2 en el menú Services y hacer *click* en Launch Instance. Se abrirá un asistente de despliegue en el que se podrá seleccionar el sistema operativo, tal como muestra la Figura 10.

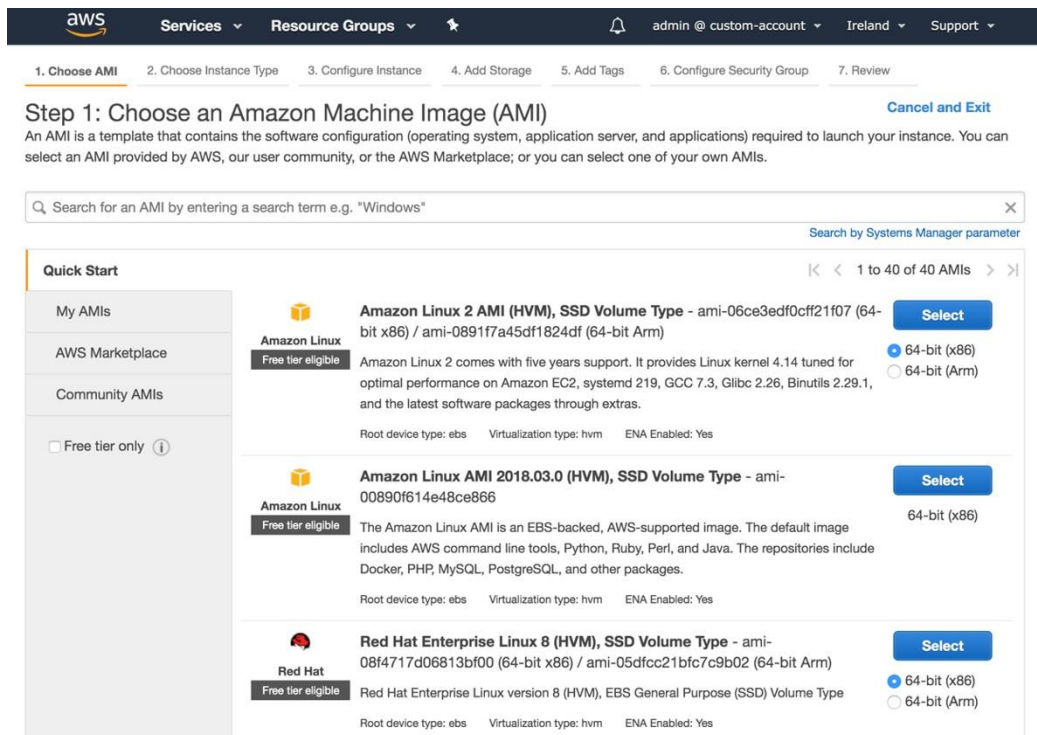


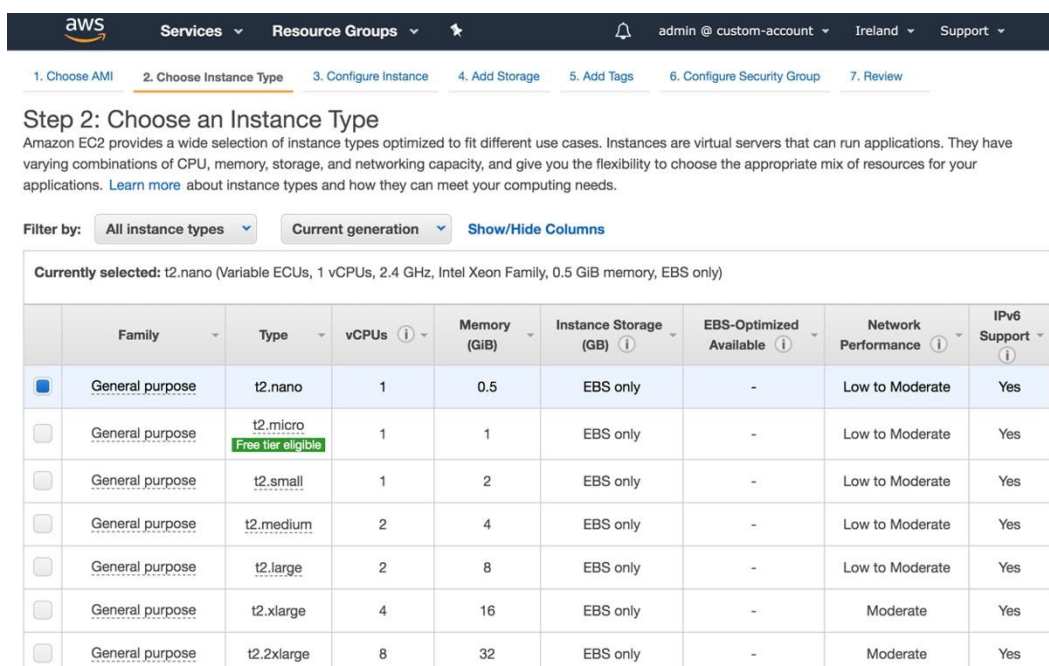
Figura 10. Consola de EC2, selección de sistema operativo. Fuente: elaboración propia.

Este paso no ofrece un sistema operativo como tal, sino una imagen de máquina o **AMI** (*Amazon machine image*). Las AMI se usan para arrancar máquinas, ya tienen los paquetes de *software* instalados, configurados y listos para ejecutarse. Amazon provee AMI para diversos sistemas operativos y la oferta crece en las pestañas del AWS Marketplace y Community AMIs.

Por ejemplo, Canonical proporciona imágenes soportadas oficialmente de varias versiones de Ubuntu, mientras que el AWS Marketplace ofrece *appliances* virtuales o máquina con *software* preinstalado para ejecutar, por ejemplo, un *router* virtual o un CMS corporativo. Algunas de estas AMI son gratuitas (es decir, sin coste por encima del precio de la instancia de AWS), mientras que otras tienen un sobrecoste. Este asistente también permite escoger una AMI propia.

Para demostrar la conexión remota a un equipo Linux, se escoge la imagen de **Amazon Linux 2**. El siguiente paso consiste en la elección del tipo de instancia. Las instancias EC2 pueden tener diversas configuraciones y especificaciones de *hardware* virtual. Además de varios tamaños de memoria y CPU (*central processing unit*), los

tipos de instancias, también denominados *flavors* informalmente, ofrecen diferentes configuraciones de velocidad de almacenamiento, rendimiento de tarjeta de red, GPU e incluso procesadores propios de AWS optimizados para, por ejemplo, la inferencia en aplicaciones de aprendizaje automático. Para este ejemplo, se selecciona el tamaño más pequeño, t2.nano, como en la Figura 11.



Step 2: Choose an Instance Type

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.nano (Variable ECUs, 1 vCPUs, 2.4 GHz, Intel Xeon Family, 0.5 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input checked="" type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

Figura 11. Tipo de instancia o *flavor*. Fuente: elaboración propia.

Una vez seleccionado el tamaño, el siguiente paso permite configurar detalles de la instancia. En la parte superior, se selecciona la red virtual y la subred a la que conectar la instancia. Además, y dado que el objetivo es demostrar el acceso remoto, la opción de asignar una IP pública está habilitada. Las redes virtuales suelen tener rangos de direcciones privadas, personalizables por el usuario, pero no enrutables desde Internet. Esto no suele ser un problema, y de hecho suele ser deseable. AWS ofrece varios servicios para exponer una aplicación a Internet (por ejemplo, API Gateway), pero, en este caso, es suficiente con que la instancia tenga una IP pública. La tarjeta de red de la instancia seguirá teniendo una IP privada y AWS configurará un NAT (*network address translation*) entre la IP pública y la privada.

Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 1 [Launch into Auto Scaling Group](#)

Purchasing option: ☐ Request Spot instances

Network: vpc-08a3dd8391df4d996 | custom [Create new VPC](#)

Subnet: subnet-0b72d64a627e30068 | subnet1 | eu-west-1c [Create new subnet](#)
251 IP Addresses available

Auto-assign Public IP: **Enable**

Placement group: ☐ Add instance to placement group

Capacity Reservation: Open [Create new Capacity Reservation](#)

IAM role: None [Create new IAM role](#)

Figura 12. Selección de red. Fuente: elaboración propia.

En la parte inferior de este diálogo, aparece una sección importante: **User data** (ver Figura 13). Es una característica fundamental de EC2, ya que proporciona la base para la automatización de despliegues: el contenido del campo User data se ejecutará durante el arranque, por lo que se puede personalizar una AMI genérica para que funcione como, por ejemplo, un servidor web o un servidor de base de datos, simplemente arrancando una instancia. Las AMI de Ubuntu y Amazon Linux soportan *scripts* de *shell* en este campo, mientras que las AMI de Windows soportan PowerShell. En el ejemplo de la Figura 13, no se hace más que escribir un mensaje en un archivo, pero cualquier tarea que se haya automatizado en un *script* podría ejecutarse desde el User data.

▼ Advanced Details

Metadata accessible: Enabled

Metadata version: V1 and V2 (token optional)

Metadata token response hop limit: 1

User data: ☒ As text ☐ As file ☐ Input is already base64 encoded

```
mkdir -p /opt/app
echo "Hello World" > /opt/app/message
```

Figura 13. Apartado User data. Fuente: elaboración propia.

Los apartados de almacenamiento y etiquetas no aportan demasiado valor para el ejemplo de acceso remoto, así que no se van a comentar en detalle. El paso de los grupos de seguridad, sin embargo, sí es importante.

Los **grupos de seguridad**, o *security groups*, son *firewalls* a nivel de instancia que se ejecutan a nivel de hipervisor, por lo que liberan al sistema operativo de ejecutar un *firewall* interno. Los grupos de seguridad de AWS bloquean todo el tráfico entrante y permiten todo el tráfico saliente por defecto. Dado que el objetivo es el acceso remoto a un equipo Linux, es necesario añadir al menos una regla para permitir el SSH desde el equipo de origen. Se puede abrir el puerto a cualquier IP, aunque es recomendable restringirlo a la IP que detecta la consola web, por ejemplo, para limitar más el acceso, tal como muestra la Figura 14.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group ☐ Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	My IP 139.47.100.40/32	e.g. SSH for Admin Desktop

Figura 14. Grupos de seguridad. Fuente: elaboración propia.

El último paso es una mera revisión de las opciones seleccionadas hasta el momento. En el momento de hacer *click* en Launch. Sin embargo, el asistente ofrece la opción de **escoger un par de claves**, o *key pair*. Las claves permiten el acceso seguro a la instancia, tal como se ha visto. Las AMI oficiales no tienen contraseña por defecto y la única manera de conseguir acceso remoto es mediante un par de claves. La clave privada se descarga en el momento de la creación y AWS no la guarda, por lo que no es posible recuperarla si se pierde. AWS solo mantiene la parte pública y se encarga de copiarla a la instancia durante el arranque. Varias instancias pueden compartir un mismo par de claves.

En caso de crear una clave nueva al vuelo, es necesario descargar el archivo .pem y guardarlo en el disco, tal como describe la Figura 15. Una vez descargado, hay que configurar los permisos del archivo solo como lectura; por ejemplo, con `chmod 400 fichero.pem`, en MacOS o en Linux.

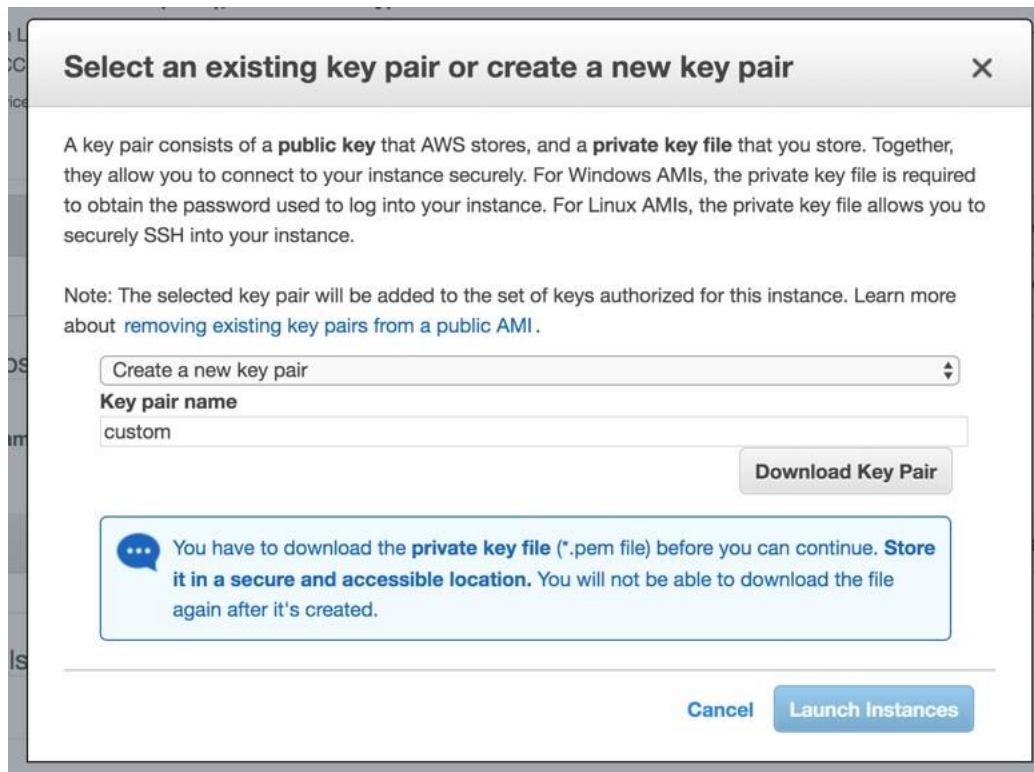


Figura 15. Creación de par de claves. Fuente: elaboración propia.

Una vez seleccionada la clave, o tras crear una nueva y descargar el archivo .pem, AWS inicia el despliegue de la instancia. En cuestión de segundos, cambiará a estado *running* y será posible obtener la IP pública. En la Figura 16, la IP pública es 63.35.201.236., mientras que la IP privada, que no es necesario conocer para este acceso, es 172.31.12.135.

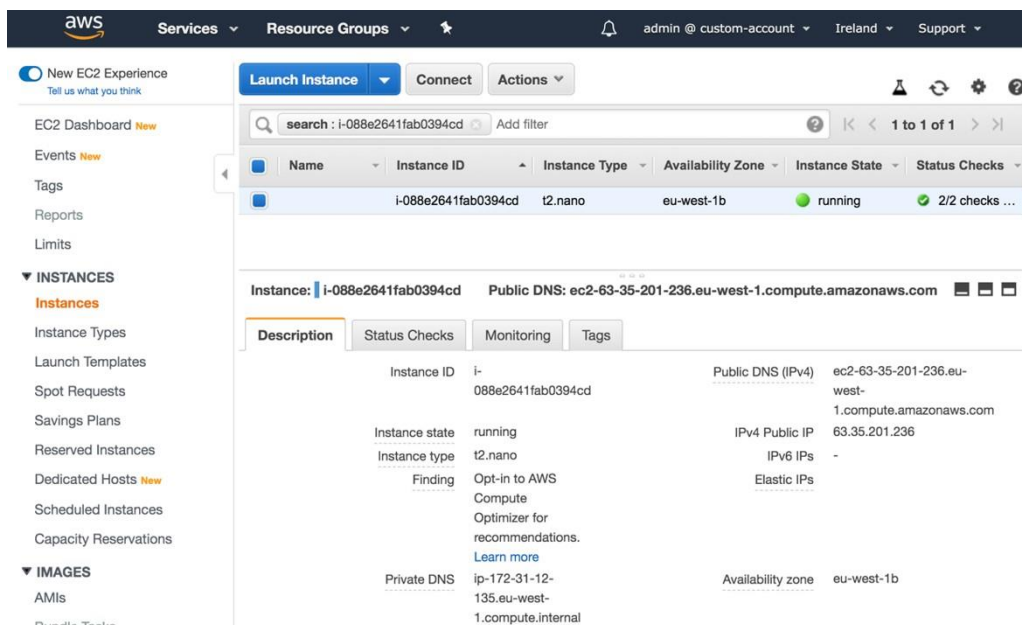


Figura 16. Instancia EC2 arrancada. Fuente: elaboración propia.

Con la instancia arrancada, ya es posible acceder por SSH con el usuario `ec2-user`. Este usuario existe por defecto en las AMI de Amazon Linux y está disponible en la documentación. Otras AMI tendrán otros nombres de usuario. El comando para iniciar sesión será `ssh -i custom.pem ec2-user@63.35.201.236`. Tal como muestra la Figura 17, será necesario aceptar la **firma de la clave**, al igual que al acceder a la instalación de Ubuntu de los apartados anteriores.

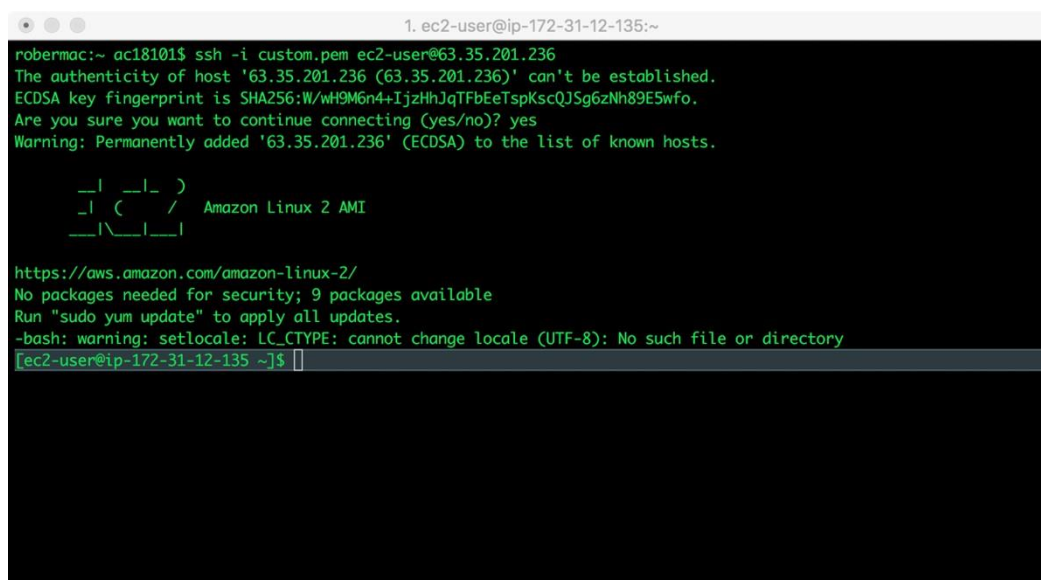
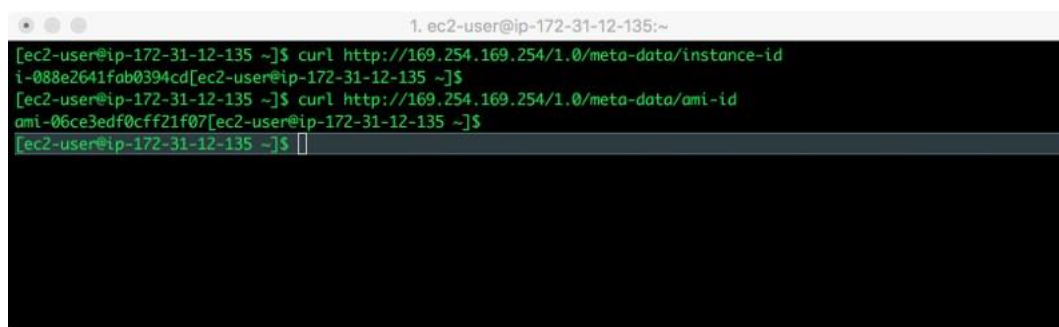


Figura 17. Conexión SSH. Fuente: elaboración propia.

Las instancias de AWS ofrecen un servicio de metadatos integrado en una pequeña API (*application programming interface*) estática accesible por HTTP. Desde la propia instancia, se pueden recuperar datos, como el ID de la instancia o de la AMI, con una simple llamada de `curl`, como muestra la Figura 18.



```
1. ec2-user@ip-172-31-12-135:~  
[ec2-user@ip-172-31-12-135 ~]$ curl http://169.254.169.254/1.0/meta-data/instance-id  
i-088e2641fab0394cd[ec2-user@ip-172-31-12-135 ~]$  
[ec2-user@ip-172-31-12-135 ~]$ curl http://169.254.169.254/1.0/meta-data/ami-id  
ami-06ce3edf0cff21f07[ec2-user@ip-172-31-12-135 ~]$  
[ec2-user@ip-172-31-12-135 ~]$
```

Figura 18. Acceso a los metadatos de la instancia. Fuente: elaboración propia.

2.9. Referencias bibliográficas

AWS. (s. f.). *Amazon Linux 2*. <https://aws.amazon.com/es/amazon-linux-2/>

Debian. (s. f.). *Información sobre Debian «buster»*.

<https://www.debian.org/releases/stable/>

Dulaney, E. (2018). *Linux All-in-one for Dummies* (6.ª ed.; cap. I, pp. 7-57). John Wiley & Sons.

Frisch, Æ. (2002). *Essential System Administration* (3.ª ed.; cap. XV, pp. 885-942). O'Reilly.

Matotek, D., Turnbull, J. y Lieverdink, P. (2017). *Pro Linux System Administration: Learn to Build Systems for Your Business Using Free and Open Source Software* (2.ª ed.; cap. X, pp. 417-471). Apress.

Página de CentOS (<https://www.centos.org/>).

Putty. (s. f.). *Download PuTTY*. <https://www.putty.org/>

RedHat Customer Portal. (s. f.). *Product Documentation for Red Hat Enterprise Linux*
8. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/

Ubuntu Releases. (s. f.). *Ubuntu 18.04.5 LTS (Bionic Beaver)*.
<http://releases.ubuntu.com/18.04/>

Van Vugt, S. (2015). *Beginning the Linux Command Line* (2.ª ed.; cap. I, pp. 1-26).
Apress.

Infraestructura de clave pública

Katz, J. y Yehuda, L. (2015). *Introduction to Modern Cryptography* (2.ª ed., parte III). Chapman & Hall Book.

El cifrado y la autenticación con claves públicas tienen una complejidad matemática importante. Para su uso práctico, no es necesario profundizar en exceso, pero una pequeña base ayuda a entender la solución a la privacidad que ofrecen las claves públicas. El Capítulo 8 de este libro es una intensa pero breve introducción al tema.

Distribuciones Linux

Página de Distrowatch (<https://distrowatch.com/>).

Distrowatch mantiene un *ranking* de las distribuciones más utilizadas, las más novedosas, así como información básica sobre cada una.

Historia de Linux

Van Vugt, S. (2015). *Beginning the Linux Command Line* (2.ª ed., cap. I, pp. 1-26). Apress.

En el primer capítulo de este libro, el autor hace un pequeño repaso de la historia de Linux, las principales distribuciones y el primer inicio de sesión en una consola Linux.

1. ¿Qué diferencia hay entre una distribución Linux y el *kernel* (núcleo) Linux?
 - A. El núcleo es la base del sistema operativo y todas las distribuciones lo incluyen, además incluyen aplicaciones adicionales.
 - B. Ninguna, son la misma cosa.
 - C. El *kernel* es una distribución más de Linux, la oficial.
 - D. El *kernel* es un paquete más, las distribuciones pueden incluirlo o no.

2. ¿Qué lenguaje de *scripting* es mejor? Selecciona dos opciones.
 - A. Bash, porque lo incluyen todas las distribuciones Linux.
 - B. Aquel que esté disponible en los sistemas en los que tiene que correr el *script*.
 - C. Uno interpretado para poder editarlo rápidamente.
 - D. Perl, porque es el más potente.

3. ¿Por qué es recomendable usar SSH en vez de Telnet?
 - A. Es incorrecto, Telnet es más recomendable.
 - B. Porque SSH es más moderno.
 - C. Porque SSH cifra el tráfico para asegurar la privacidad del contenido en caso de que sea interceptado.
 - D. Porque la implementación OpenSSH es de código abierto.

4. ¿Qué comando se puede utilizar en Linux para generar un par de clave privada y pública?
 - A. `ssh-keygen`.
 - B. `ssh`.
 - C. `ssh-copy-id`.
 - D. Ninguno de los anteriores.

5. ¿Qué tipo de autenticación soporta AWS por defecto en sus imágenes Amazon Linux?
- A. Usuario y contraseña y clave pública.
 - B. Usuario y contraseña, con una contraseña por defecto conocida.
 - C. Clave pública.
 - D. Ninguno, no es posible iniciar sesión de manera interactiva.
6. ¿Qué tipo de autenticación soporta Putty?
- A. Usuario y contraseña y clave pública.
 - B. Usuario y contraseña únicamente.
 - C. Clave pública únicamente.
 - D. Clave pública, que debe haber sido generada con Puttygen.
7. ¿Cuál de las siguientes distribuciones Linux se puede descargar sin coste alguno?
- A. Ubuntu.
 - B. RHEL.
 - C. CentOS.
 - D. Todas las anteriores.
8. ¿Qué comando habría que usar para conectarse por SSH al equipo *host1.local* con el usuario *localuser* y una clave privada alojada en *secure.key*?
- A. `ssh localuser@host1.local`.
 - B. `ssh -i secure.key localuser@host1.local`.
 - C. `putty localuser@host1.local`.
 - D. Ninguno de los anteriores.

9. ¿Qué significa escribir un *script* pensando que todo puede ir mal?
- A. Que hay que detectar todos los errores posibles.
 - B. En caso de error, informar al usuario de lo que ha ocurrido.
 - C. Controlar el error de manera que el *script* lo solucione o termine de una manera controlada.
 - D. Todas las anteriores.
10. ¿Qué puerto hay que abrir en un grupo de seguridad para permitir conexiones SSH?
- A. 22.
 - B. 23.
 - C. 443.
 - D. Ninguno de los anteriores.