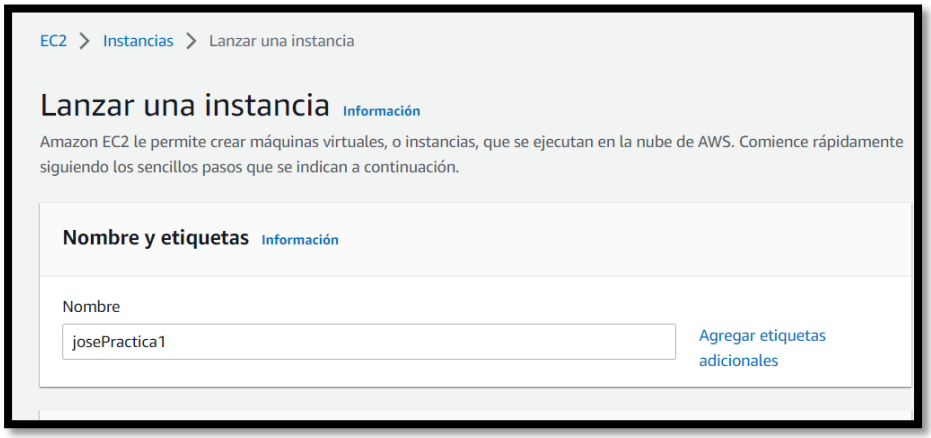
José Ortega Martínez

DESPLIEGUE DE APLICACIONES WEB

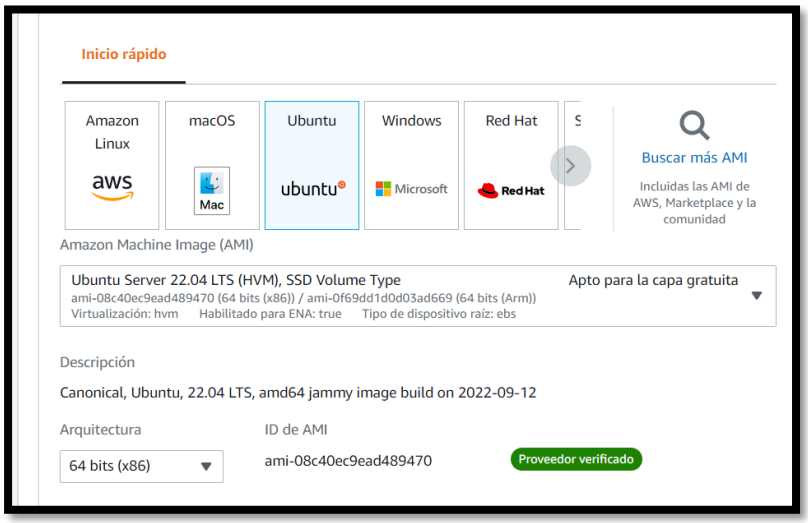
PRÁCTICA 1. SSH + AWS

En primer lugar creamos la instancia Linux EC2. Estos tres últimos caracteres hacen referencia al espacio que te otorga Amazon Academy (AWS) para alojar datos en su nube. De esta forma podremos levantar una máquina virtual Linux en el servidor.

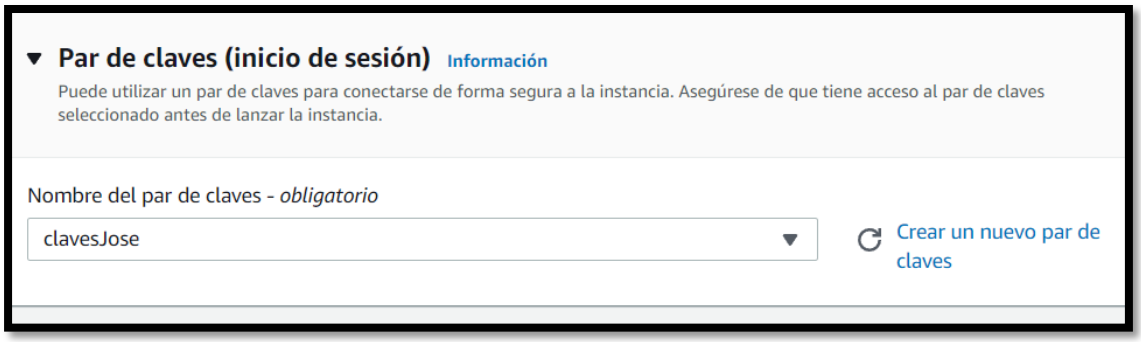
1. Una vez en la consola de Amazon EC2, desde el panel se puede lanzar una instancia. Habrá que nombrarla.



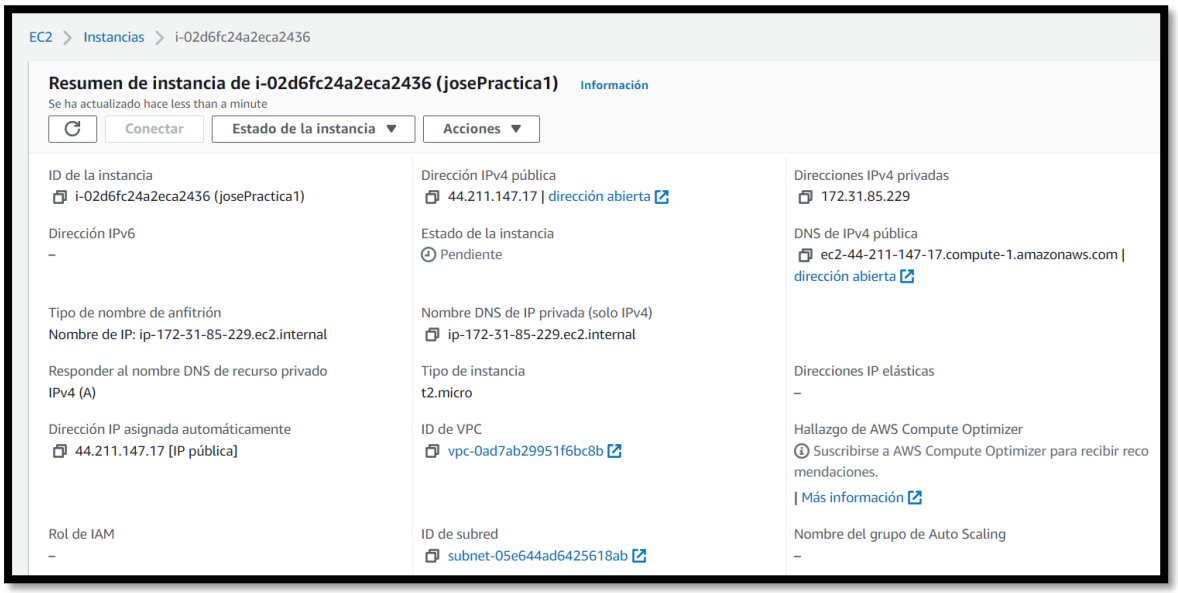
2. Seleccionamos también el tipo de instancia que queremos crear. En nuestro caso, de Linux Ubuntu.



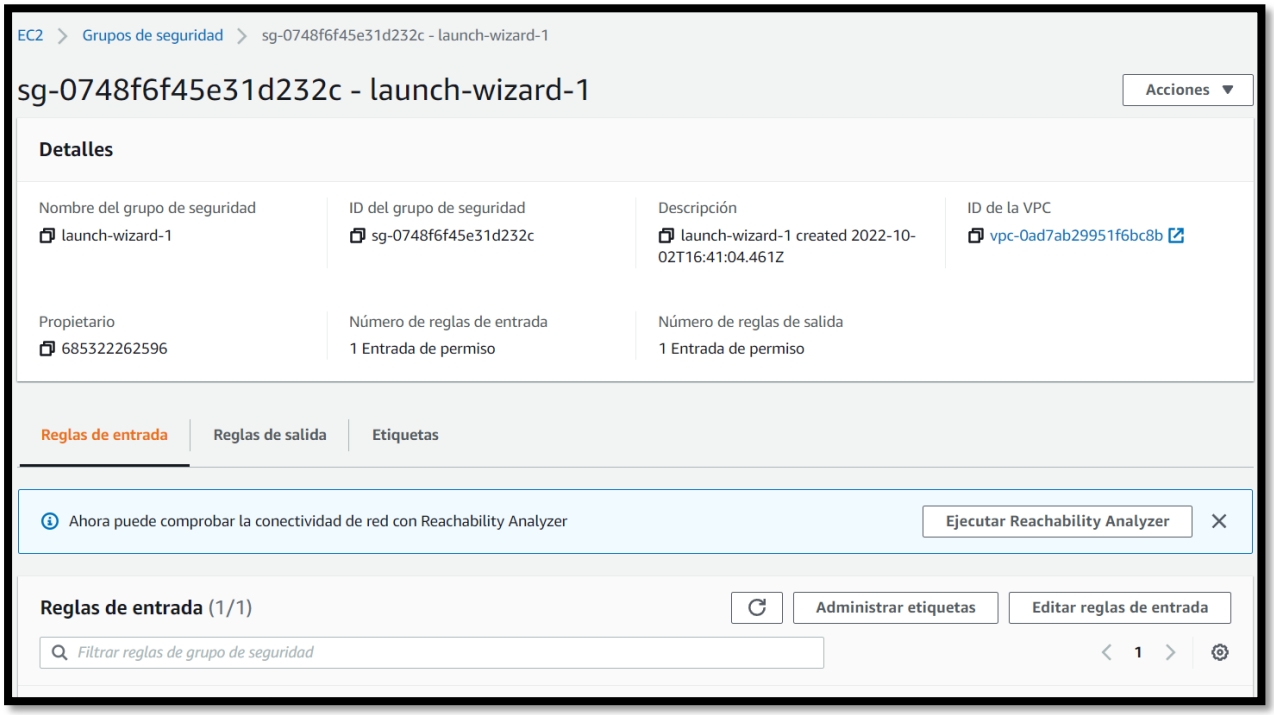
3. Le damos a ‘crear un nuevo par de claves’, que nos genera un archivo .pem. Este archivo sería suficiente si accediéramos al servicio SSH desde la consola de Ubuntu, pero como lo vamos a hacer con Windows, este archivo tendrá que pasar posteriormente por un generador de claves de Putty, que nos dará el par de claves definitivo.

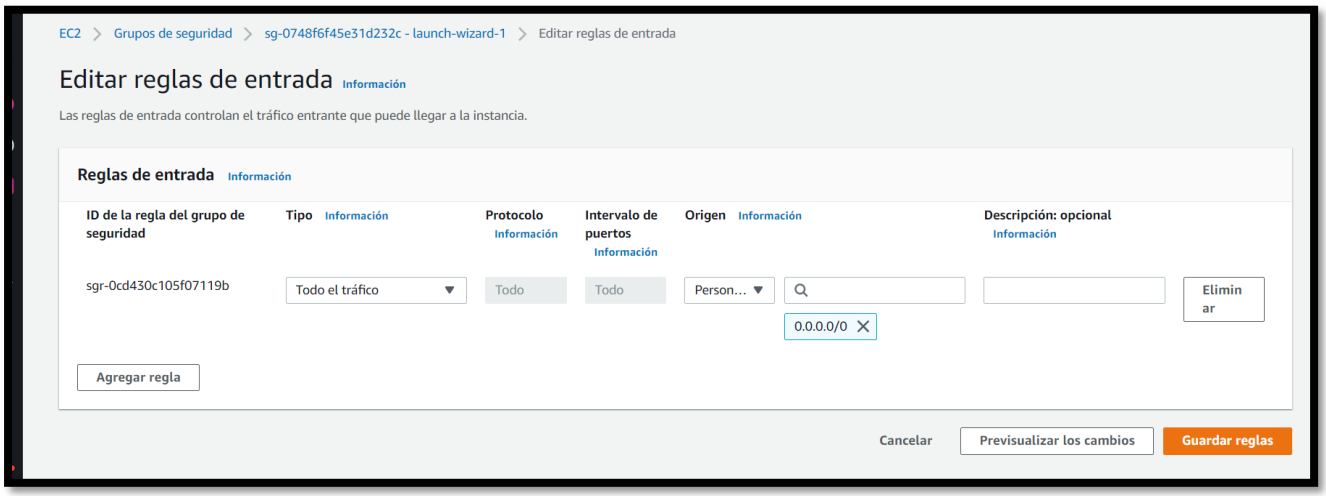


4. Así queda el resumen de la instancia recién creada.

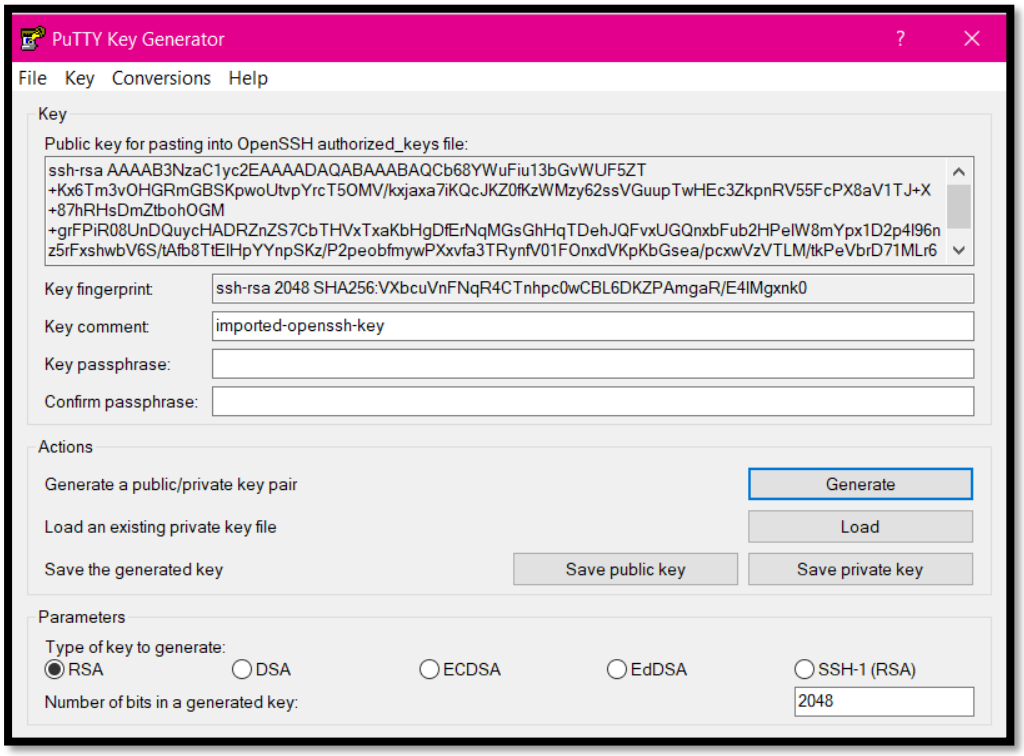


5. En la pestaña de seguridad hay un enlace con el que podemos acceder a la configuración y editar las reglas de entrada. En el tipo de entrada que tienen acceso a la instancia, seleccionamos ‘todo el tráfico’.

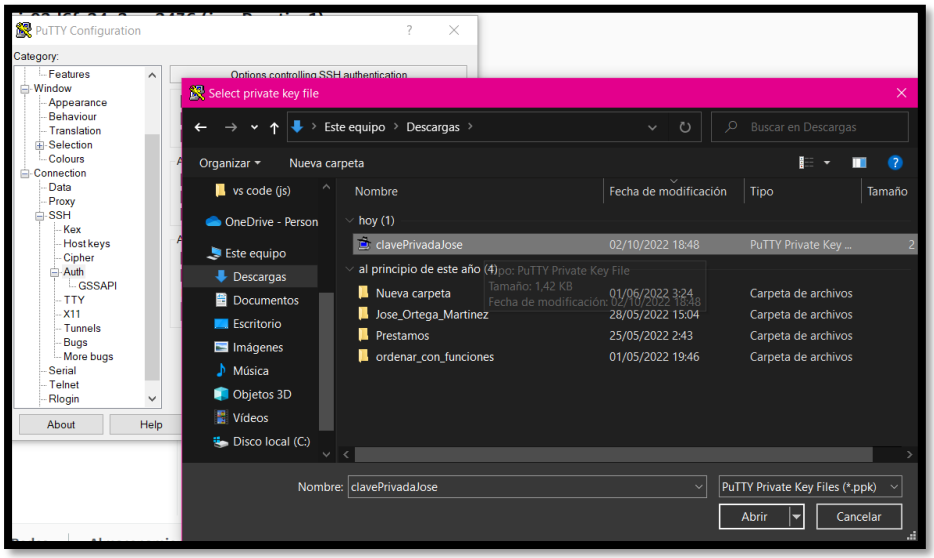




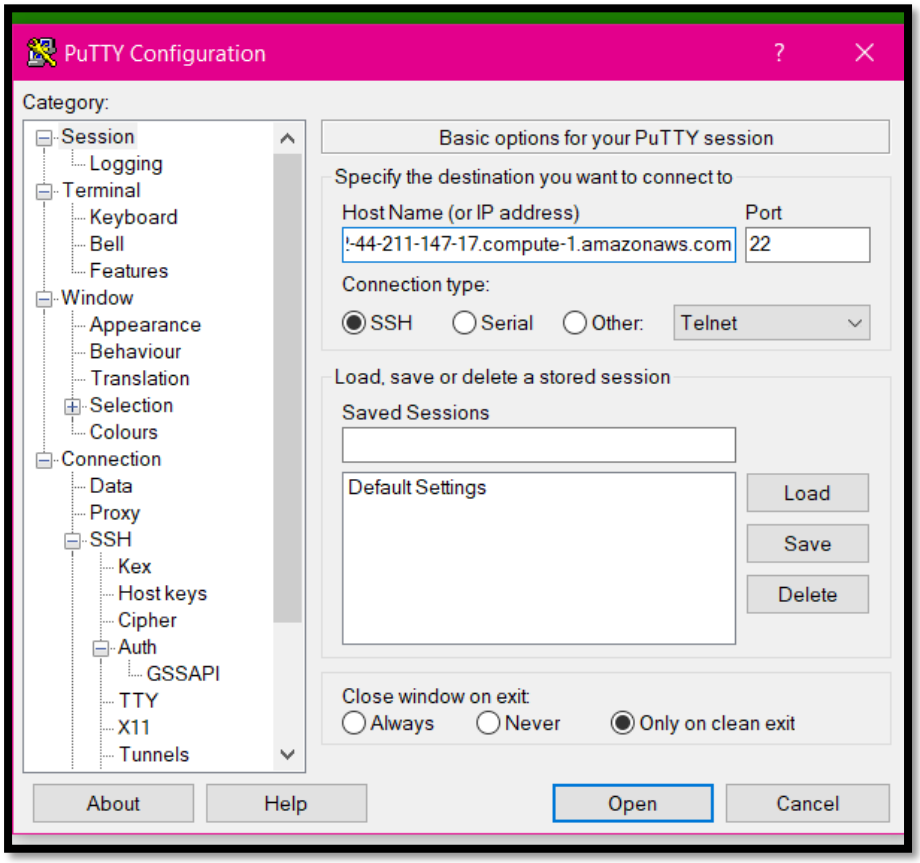
6. Ahora nos centramos en la parte del cliente. Con el Putty Key Generator importamos el archivo generado anteriormente para poder crear el par de claves (pública y privada).



7. En la configuración de Putty nos situamos en la pestaña ‘Auth’, y llamamos a la clave recién generada para que nos permita el acceso.

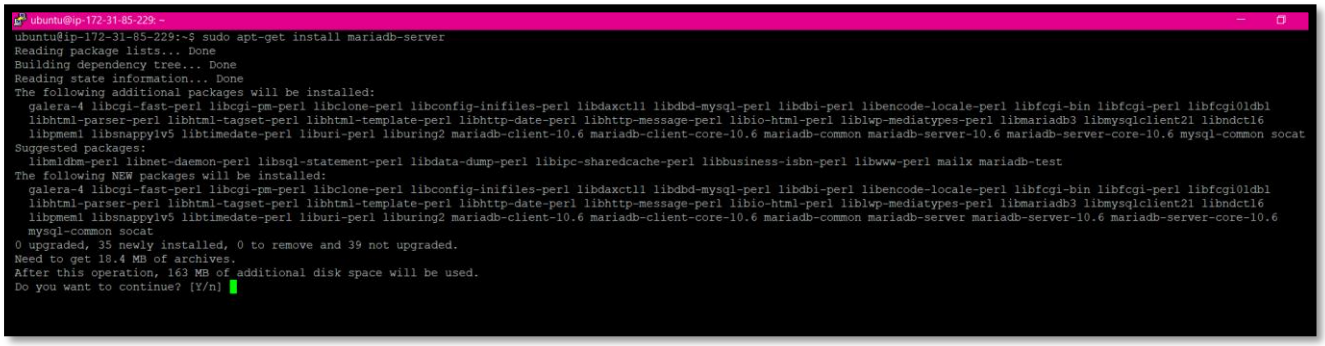


8. En la pestaña ‘Session’ introducimos el nombre del dominio, que lo obtenemos del resumen de la instancia, en ‘DNS de IPv4 pública’, y ya podemos abrir para acceder a la terminal de comandos.

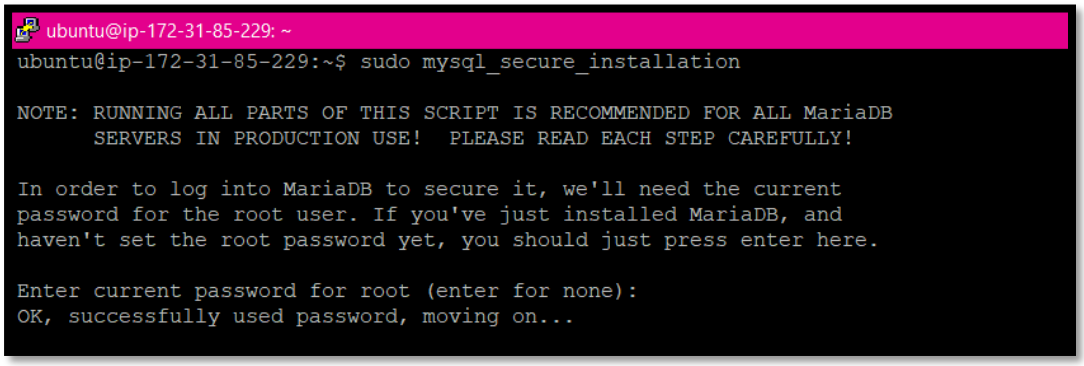


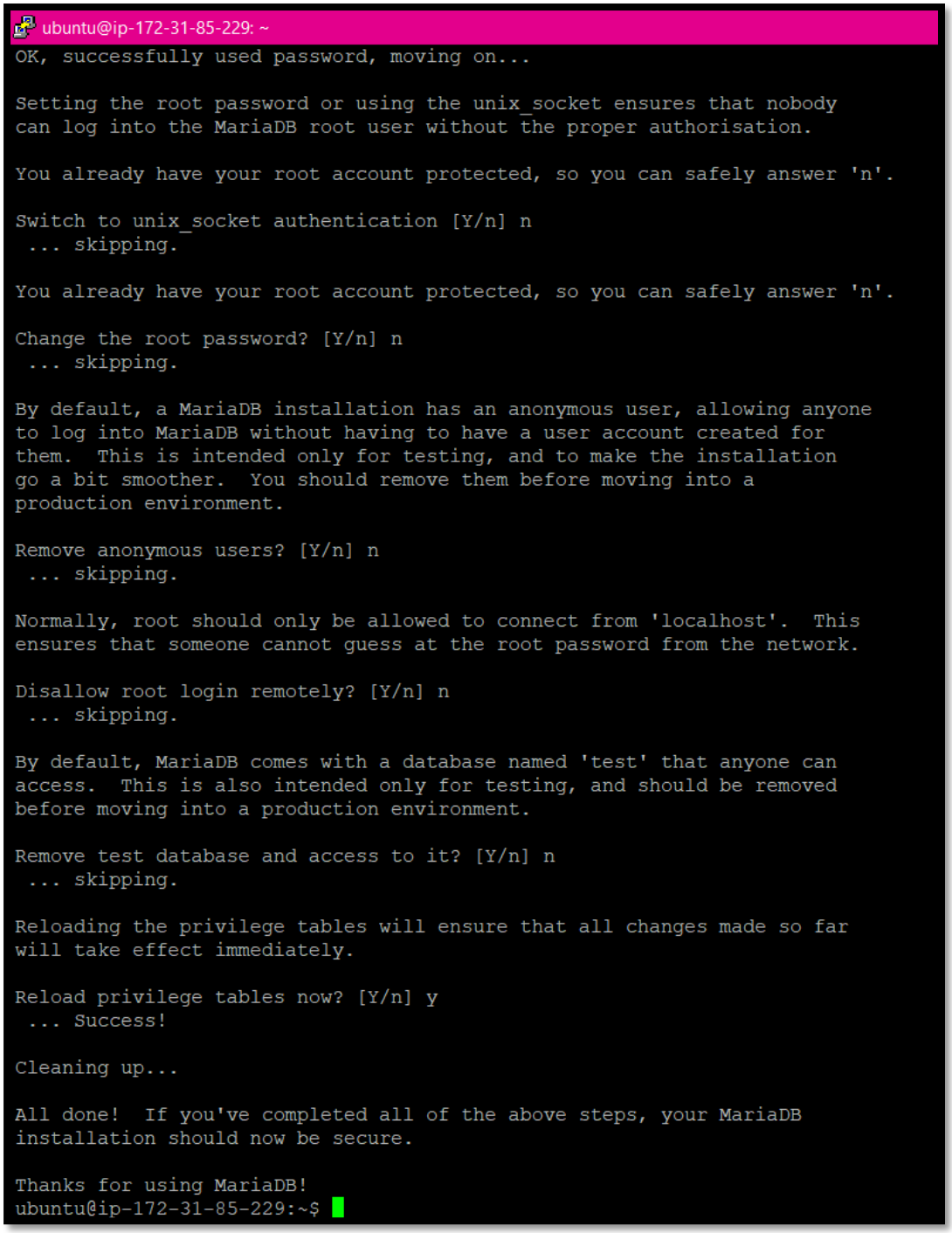
9. Introduciendo ‘ubuntu’ como nombre de usuario tenemos acceso al servicio SSH. Ahora ya estamos conectados remotamente con la máquina virtual.

10. Ahora tenemos que instalar el servidor de base de datos ‘mariadb’ en nuestra máquina virtual. Actualizamos primero el repositorio de archivos y después introducimos ‘sudo apt-get install mariadb-server'.



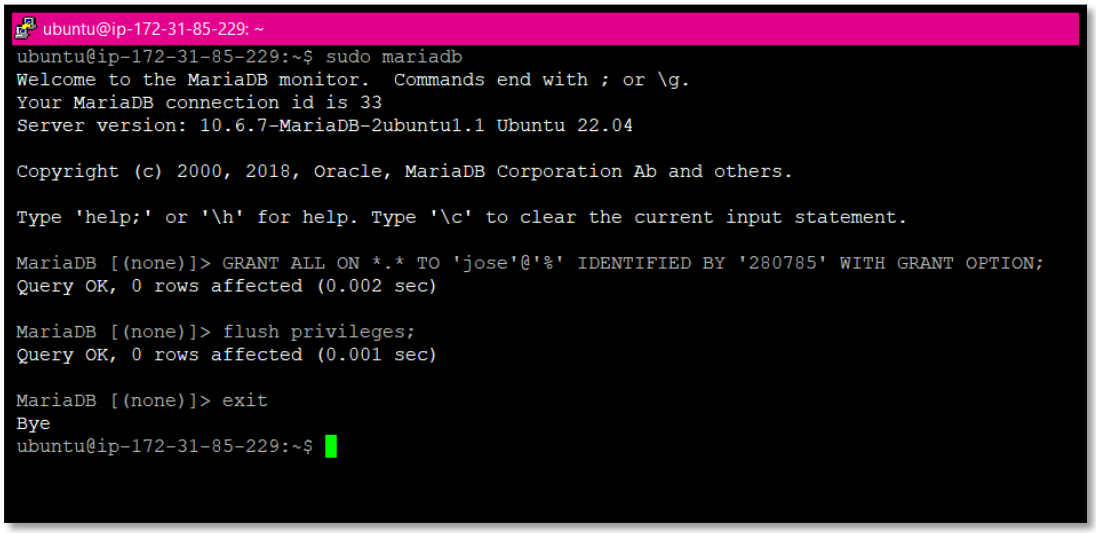
11. Ejecutamos ahora un script de seguridad para poder configurar el acceso al servidor de base de datos. Primero te pide la contraseña actual (como aún no tenemos presionamos ‘Enter’) y también habrá que elegir una serie de opciones. Nos preguntará si queremos establecer alguna contraseña. Es aconsejable no cambiar nada en cuanto al acceso, ya que podría generar problemas para la ejecución del mantenimiento, que accede de manera automática. Solo contestamos que sí a la última pregunta, para actualizar así los privilegios seleccionados.



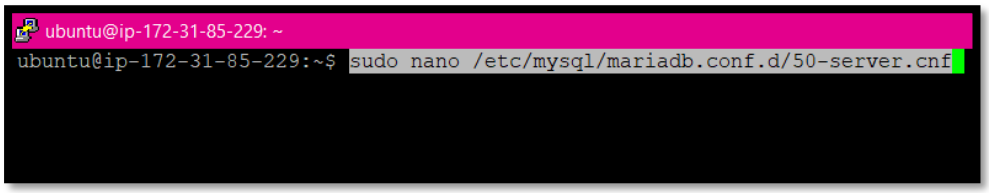

ubuntu@ip-172-31-85-229: ~
OK, successfully used password, moving on...
Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.
You already have your root account protected, so you can safely answer 'n'.
Switch to unix_socket authentication [Y/n] n
... skipping.
You already have your root account protected, so you can safely answer 'n'.
Change the root password? [Y/n] n
... skipping.
By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.
Remove anonymous users? [Y/n] n
... skipping.
Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.
Disallow root login remotely? [Y/n] n
... skipping.
By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.
Remove test database and access to it? [Y/n] n
... skipping.
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.
Reload privilege tables now? [Y/n] y
... Success!
Cleaning up...
All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.
Thanks for using MariaDB!
ubuntu@ip-172-31-85-229:~$

12. Si estamos ejecutando el servidor mariadb en ubuntu, como es el caso, el sistema de autenticación de usuario no se hace con contraseña, sino con un plugin que usa el socket como identificador. Esto proporciona una gran seguridad para el acceso del usuario, pero puede resultar un problema para el acceso de cualquier otro programa (por ejemplo un gestor de bases de datos) o un tercero. Esto podemos solucionarlo creando una cuenta extra. Tendrá los mismos privilegios que el 'root', pero estableceremos una contraseña para ella.

    Accedemos al servidor mariadb y, con la línea de comandos que se puede ver en la captura creamos el nuevo usuario 'jose', con contraseña '280785'.



13. Accedemos al archivo de configuración de conexiones al servidor, y cambiamos el bind address por 0.0.0.0. De esta forma permitimos que mariadb acepte peticiones de otros dominios, o sea, conexiones remotas.

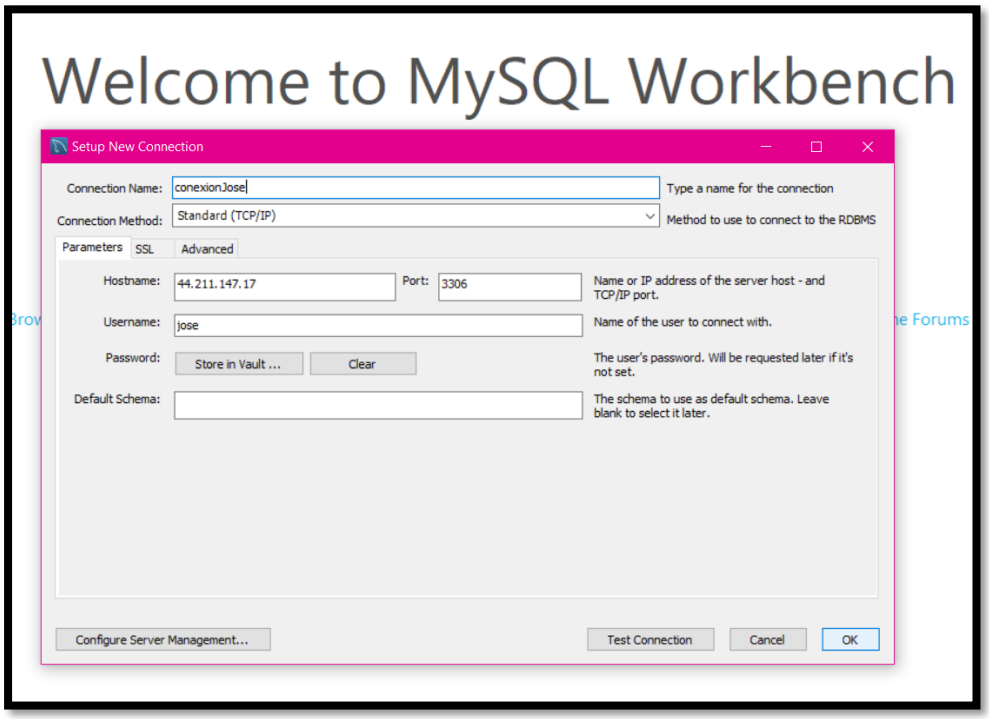


```
ubuntu@ip-172-31-85-229: ~
  GNU nano 6.2                                                    /etc
#
# These groups are read by MariaDB server.
# Use it for options that only the server (but not clients) should see

# this is read by the standalone daemon and embedded servers
[server]

# this is only for the mysqld standalone daemon
[mysqld]

#
# * Basic Settings
#

#user                    = mysql
pid-file                = /run/mysqld/mysqld.pid
basedir                 = /usr
#datadir                 = /var/lib/mysql
#tmpdir                  = /tmp

# Broken reverse DNS slows down connections considerably and name resolve is
# safe to skip if there are no "host by domain name" access grants
#skip-name-resolve

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0

#
# * Fine Tuning
#

#key_buffer_size         = 128M
#max_allowed_packet      = 1G
#thread_stack            = 192K
#thread_cache_size       = 8
# This replaces the startup script and checks MyISAM tables if needed
# the first time they are touched
#myisam_recover_options = BACKUP
#max_connections         = 100
#table_cache             = 64

#
# * Logging and Replication
#
```

14. Reiniciamos el servicio para aplicar los cambios.

```
ubuntu@ip-172-31-85-229: ~
ubuntu@ip-172-31-85-229:~$ sudo systemctl restart mariadb
ubuntu@ip-172-31-85-229:~$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.6.7 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-10-02 17:00:19 UTC; 21s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 2502 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (code=exited, status=0/SUCCESS)
    Process: 2504 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
    Process: 2506 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= ||   VAR=`cd /usr/bin/..; /usr/bin/galera_recovery`; [ $? -eq 0 ]   && systemctl set-environment _WSREP_>
    Process: 2545 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
    Process: 2547 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
   Main PID: 2535 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 11 (limit: 1143)
     Memory: 61.5M
        CPU: 335ms
     CGroup: /system.slice/mariadb.service
             └─2535 /usr/sbin/mariadbd

Oct 02 17:00:19 ip-172-31-85-229 mariadbd[2535]: Version: '10.6.7-MariaDB-2ubuntu1.1'  socket: '/run/mysqld/mysqld.sock'  port: 3306  Ubuntu 22.04
Oct 02 17:00:19 ip-172-31-85-229 systemd[1]: Started MariaDB 10.6.7 database server.
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2549]: Upgrading MySQL tables if necessary.
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2552]: Looking for 'mysql' as: /usr/bin/mysql
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2552]: Looking for 'mysqlcheck' as: /usr/bin/mysqlcheck
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2552]: This installation of MariaDB is already upgraded to 10.6.7-MariaDB.
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2552]: There is no need to run mysql_upgrade again for 10.6.7-MariaDB.
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2552]: You can use --force if you still want to run mysql_upgrade
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2560]: Checking for insecure root accounts.
Oct 02 17:00:19 ip-172-31-85-229 /etc/mysql/debian-start[2564]: Triggering myisam-recover for all MyISAM tables and aria-recover for all Aria tables
lines 1-28/28 (END)
```

15. Nos vamos al Workbench de MySQL y establecemos una nueva conexión. Le ponemos un nombre, cogemos la IP pública de nuestra máquina virtual y la colocamos en el hostname, y le damos el usuario y la contraseña para poder acceder al servidor mariadb.



16. Creamos una base de datos llamada 'despliegue_jose_ortega'. Refrescamos y ya nos aparece a la izquierda.