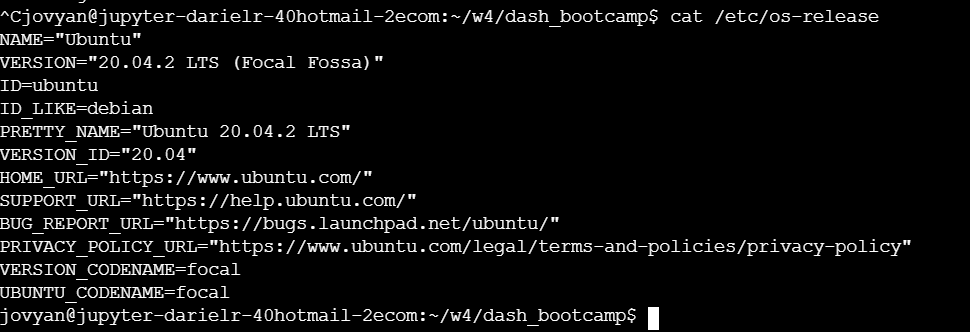
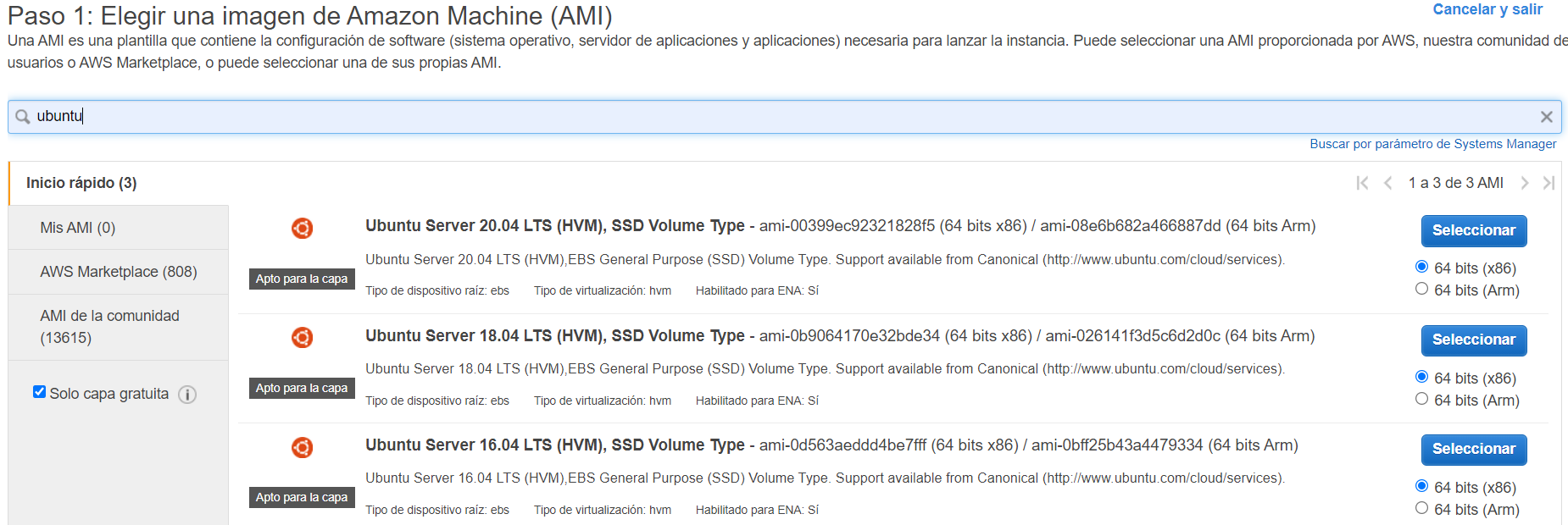
Versión de nuestro OS work Space

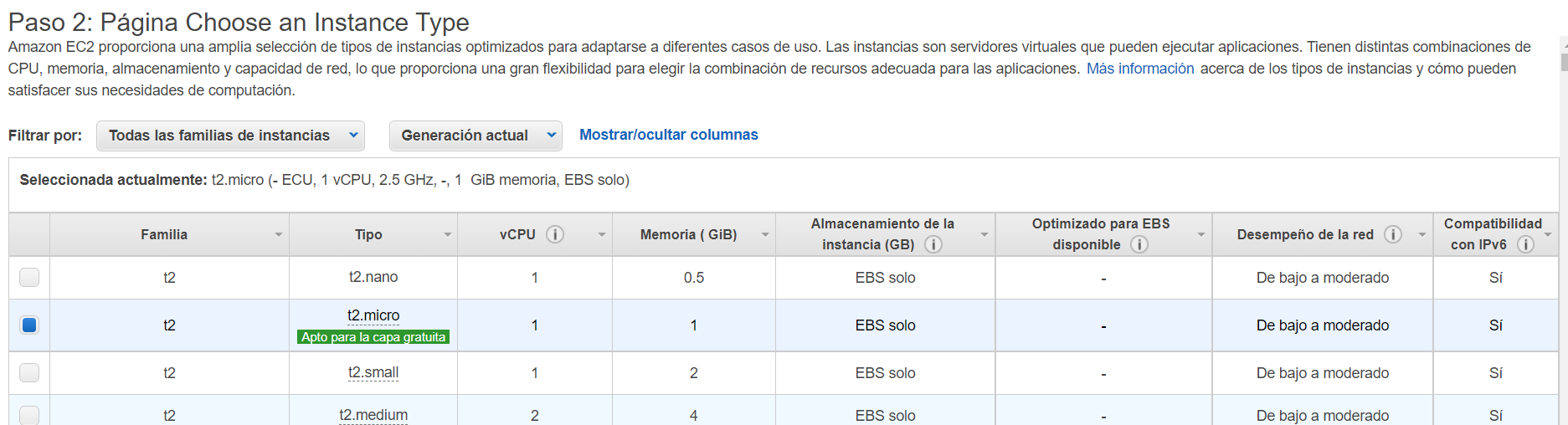
cat /etc/os-release



Seleccionar SO



Se selecciona el tipo de maquina

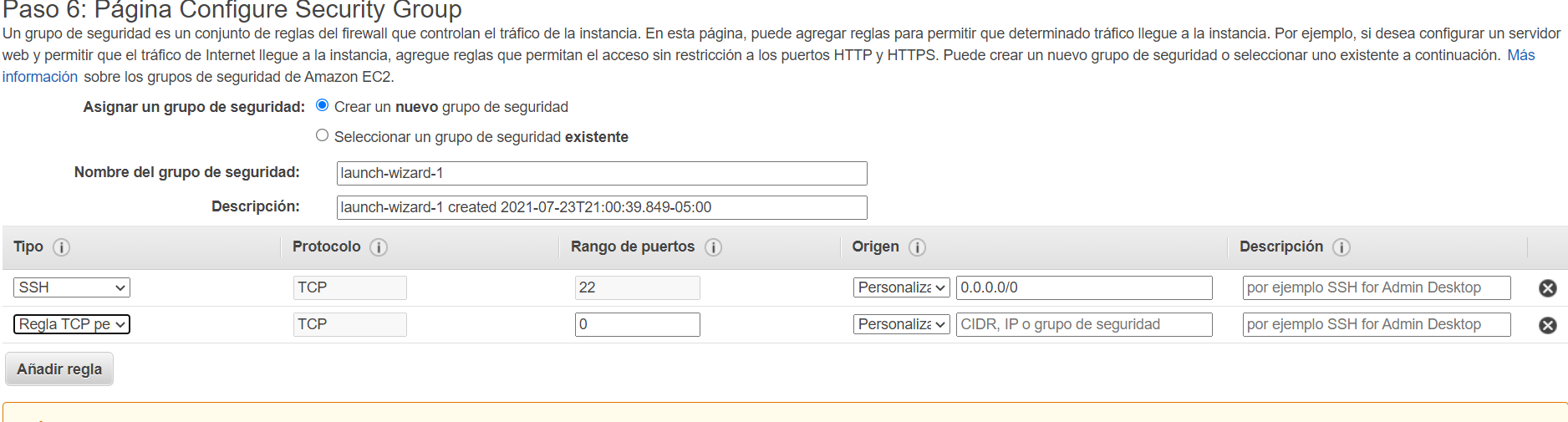




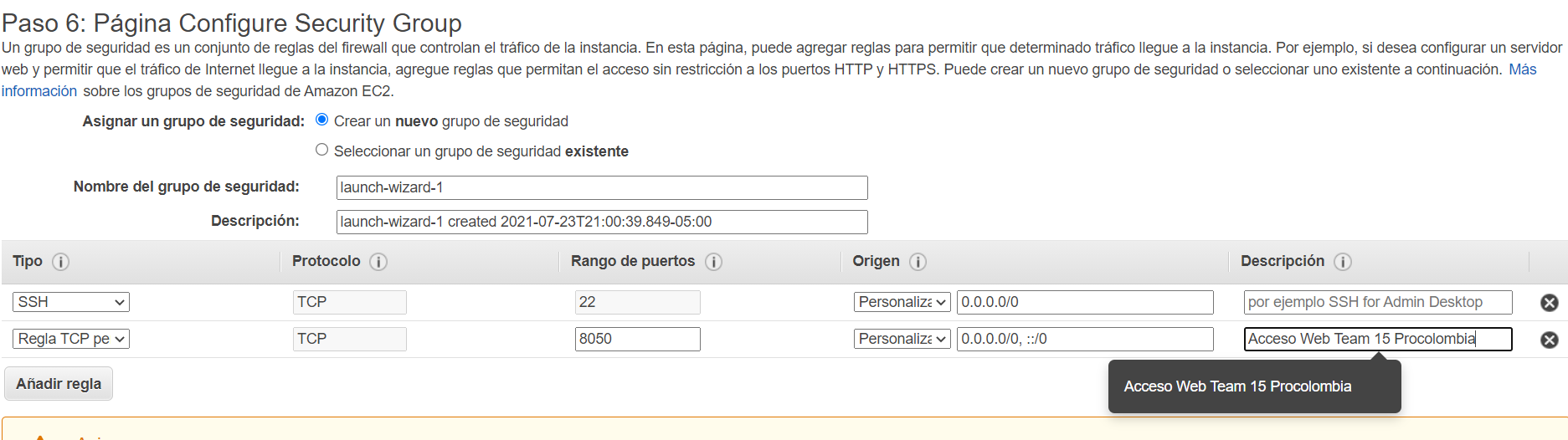
Editar grupos de seguridad



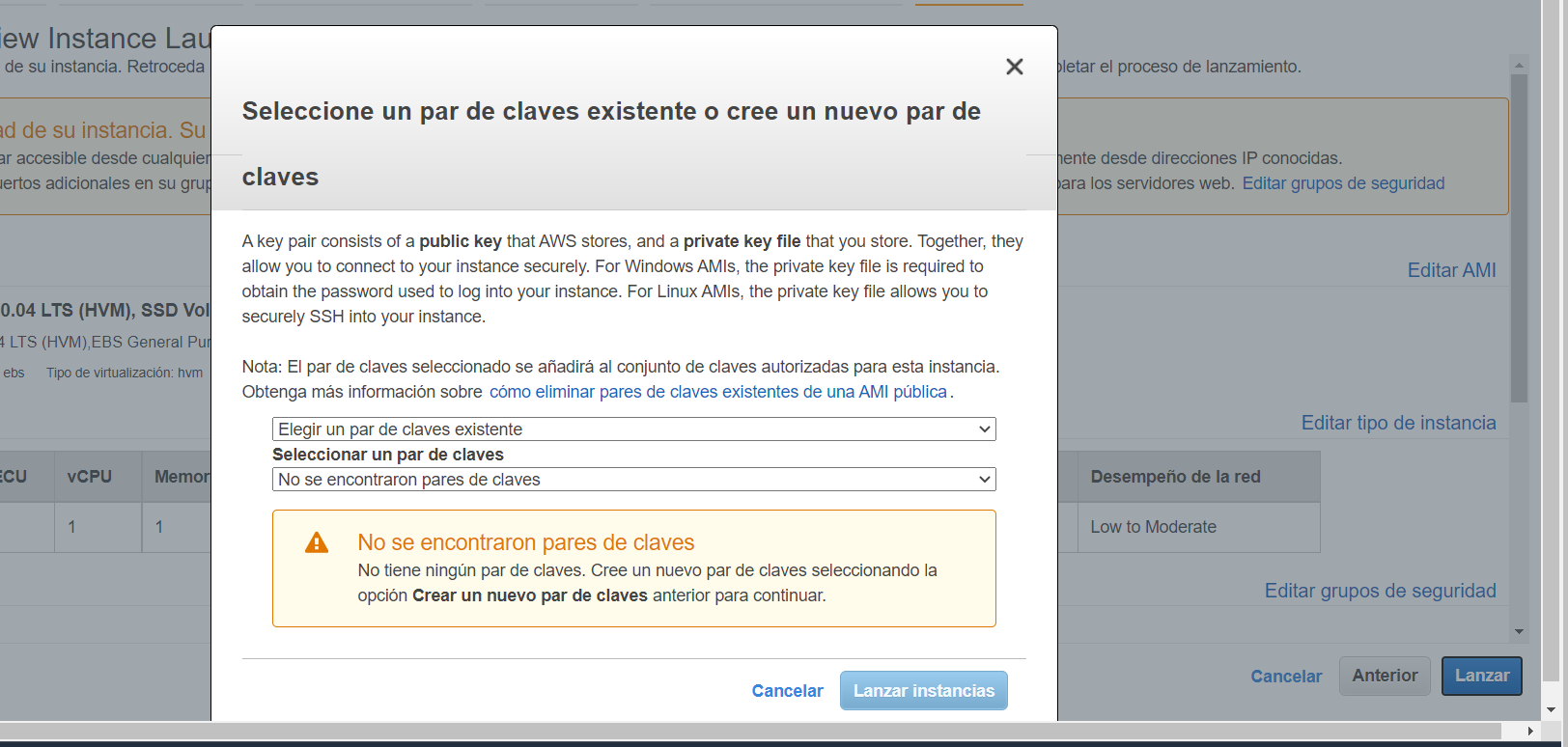
Añadimos una regla:

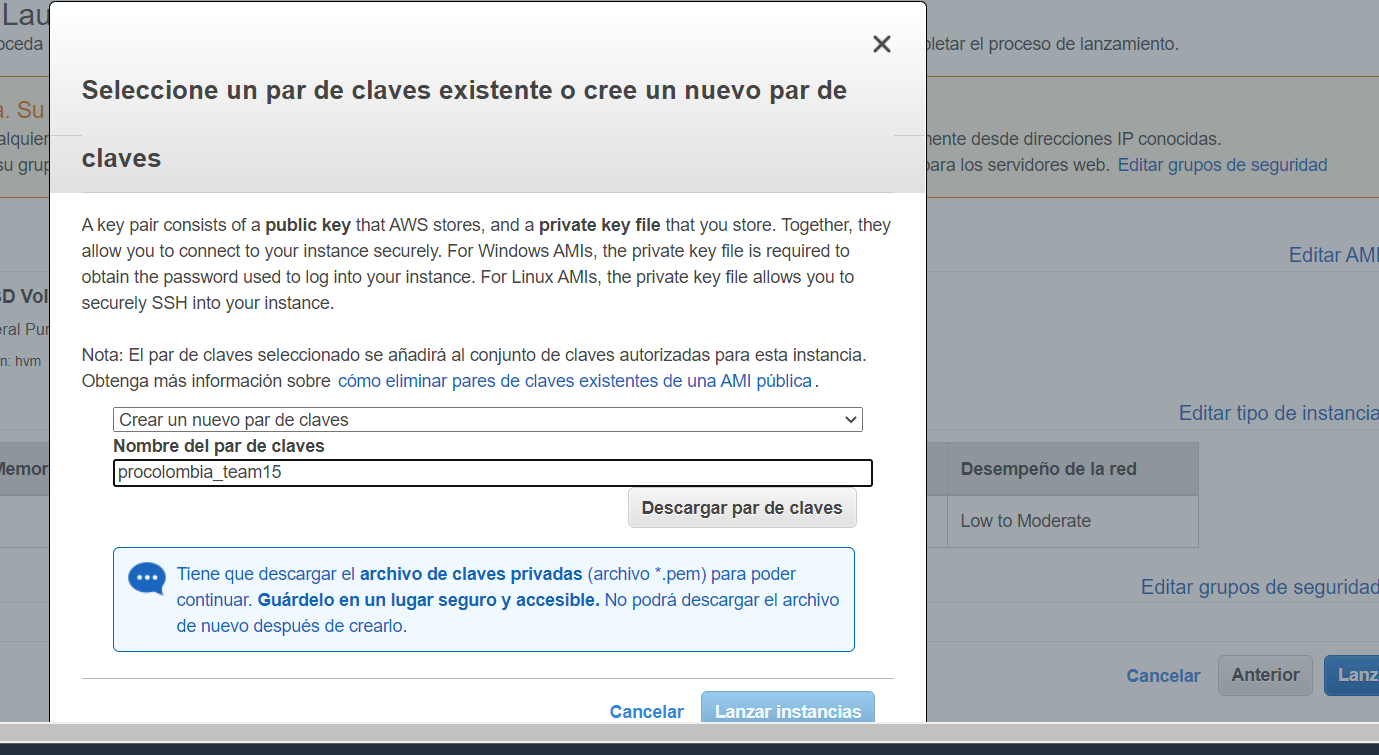


Seleccionamos tipo de protocolo TCP personalizado y puerto 8050:



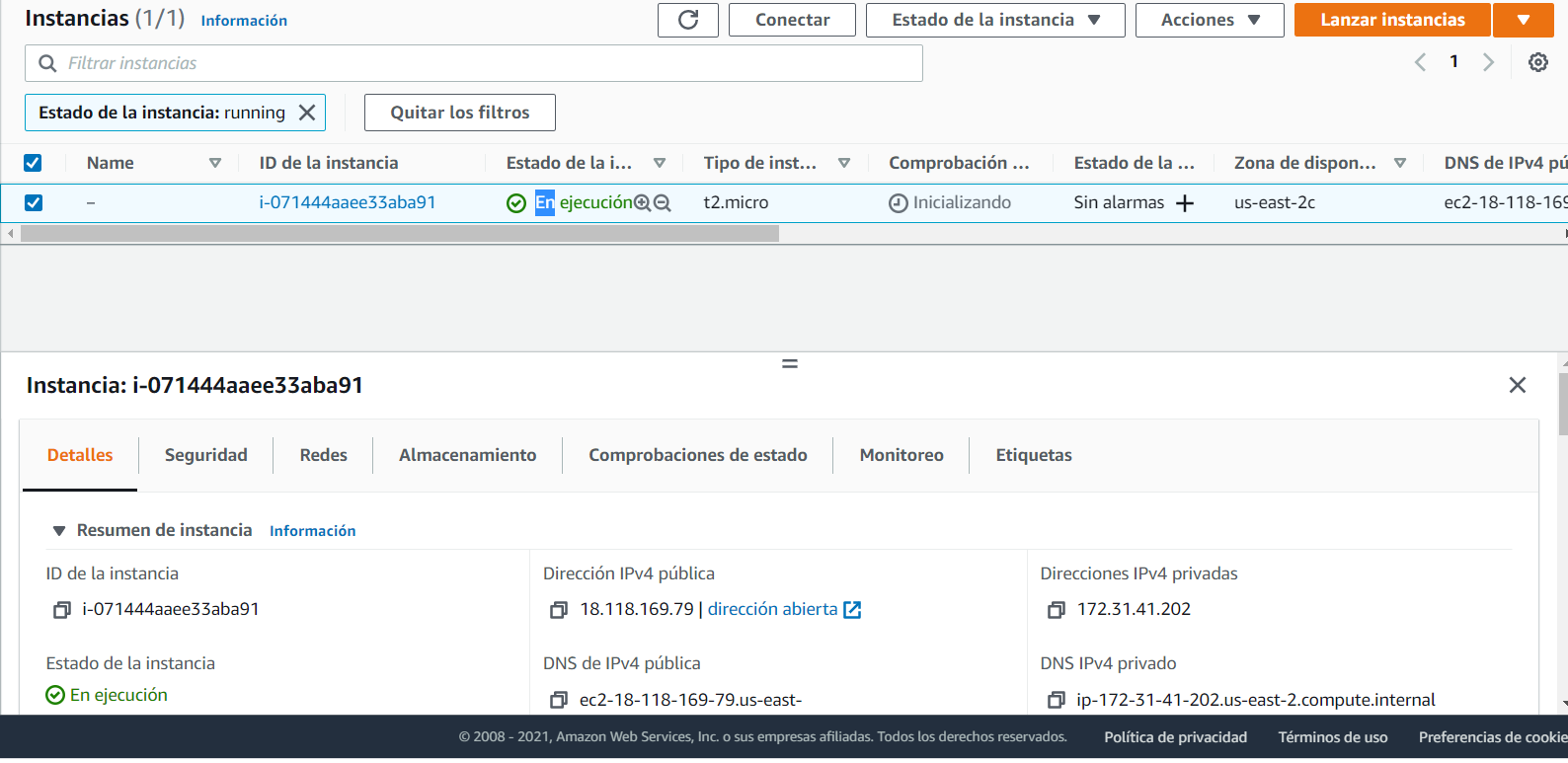
Al darle click en lanzar guardar claves de acceso:





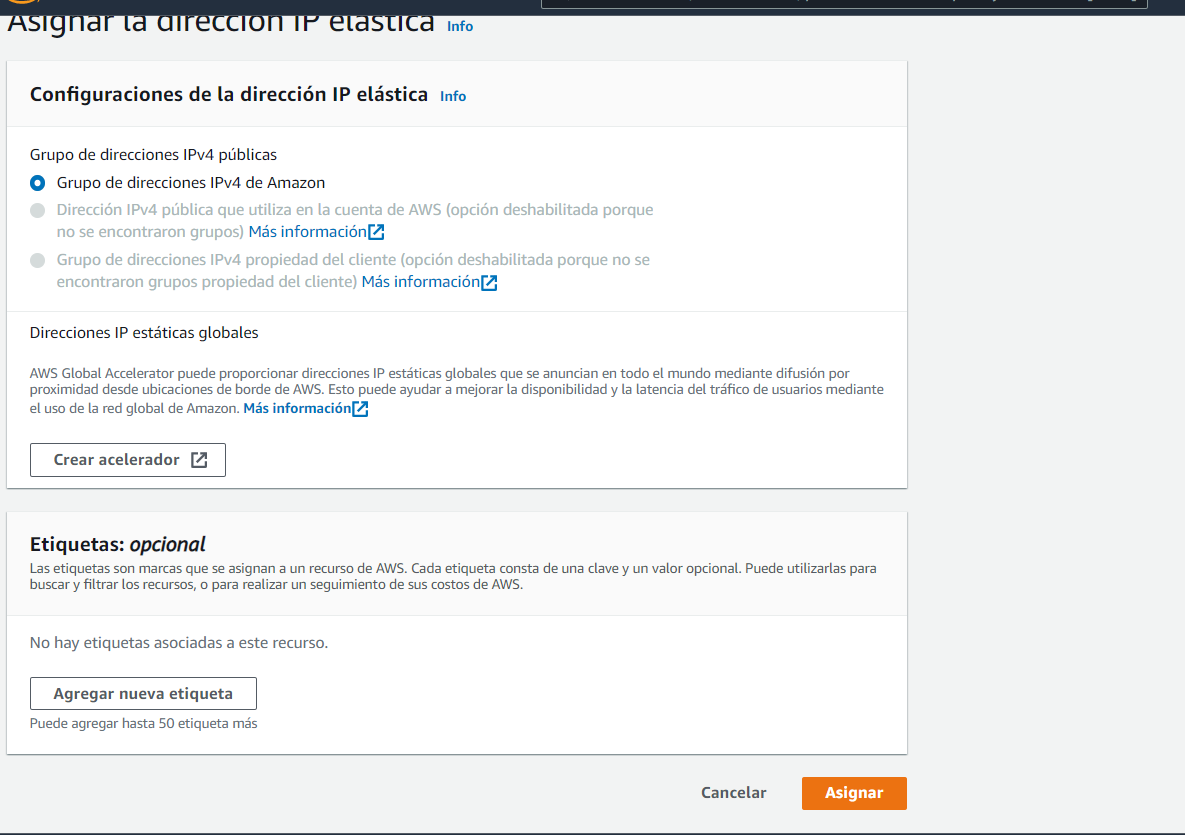
Descargamos claves (.pem) y lanzamos instancia.

Instancia en ejecución.

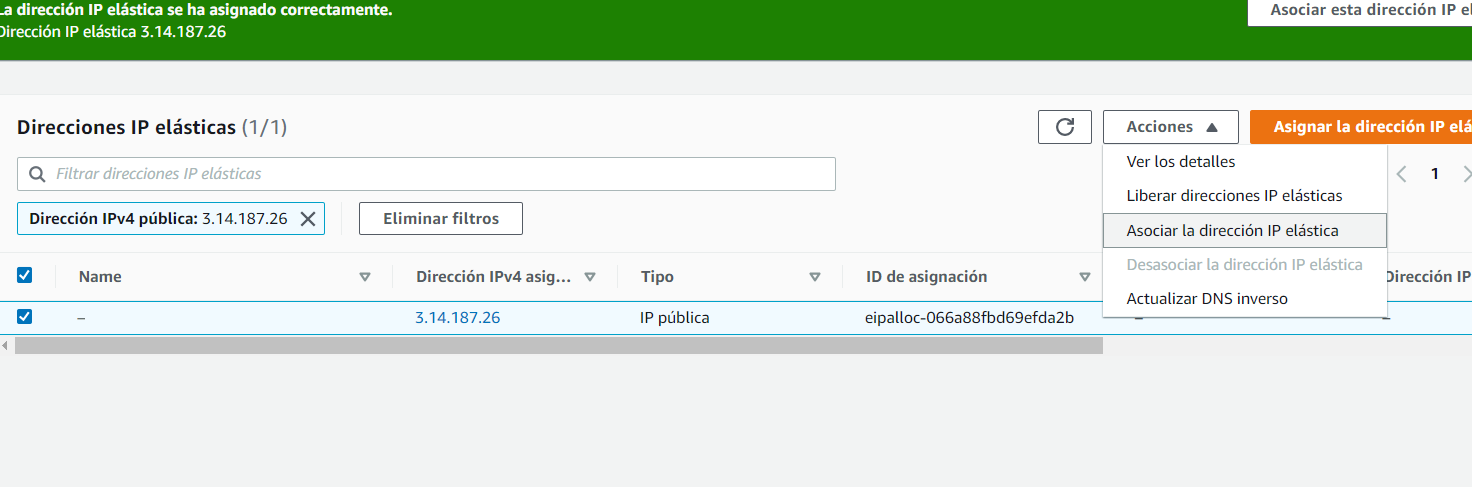


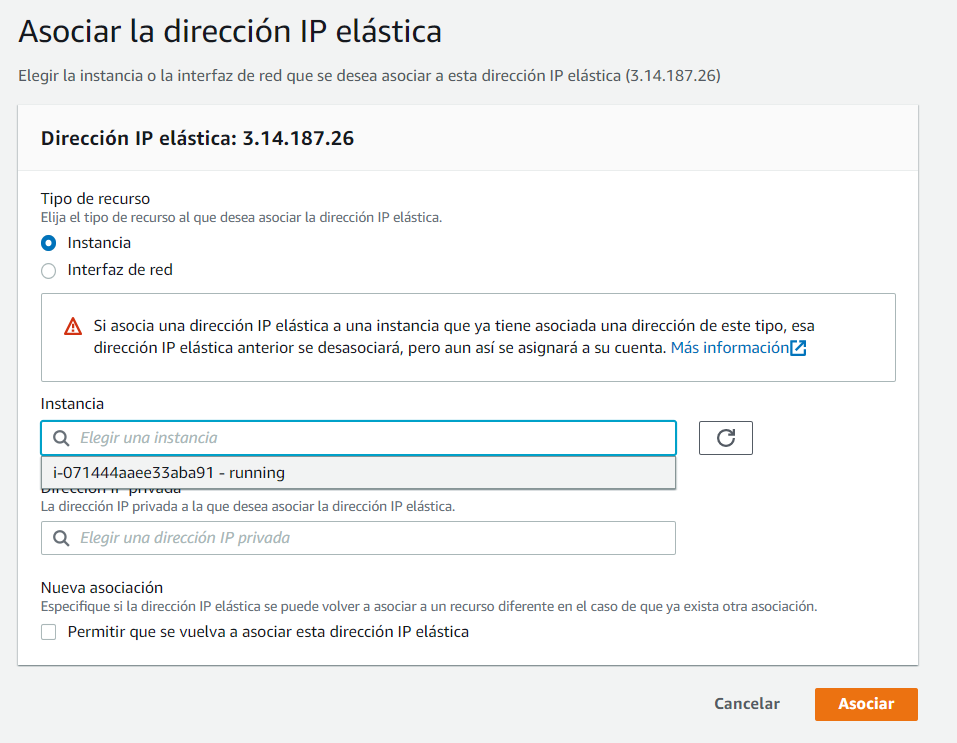
Reservemos una IP para que no cambie cada vez que baje la instancia de mi máquina, Direcciones ip elásticas:



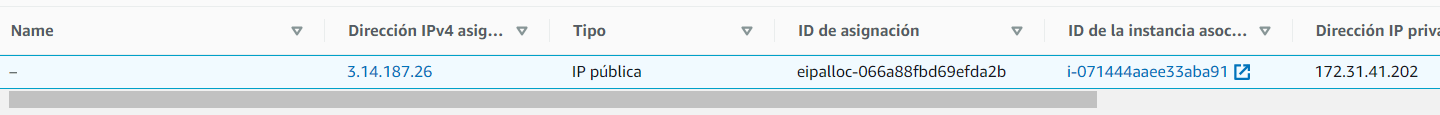


Asociamos la IP generada por AMZ a nuestra maquina.





Aquí se ve la asociación:



IP: 3.14.187.26

Configurar Conexión a la maquina:

Download MobaXTerm

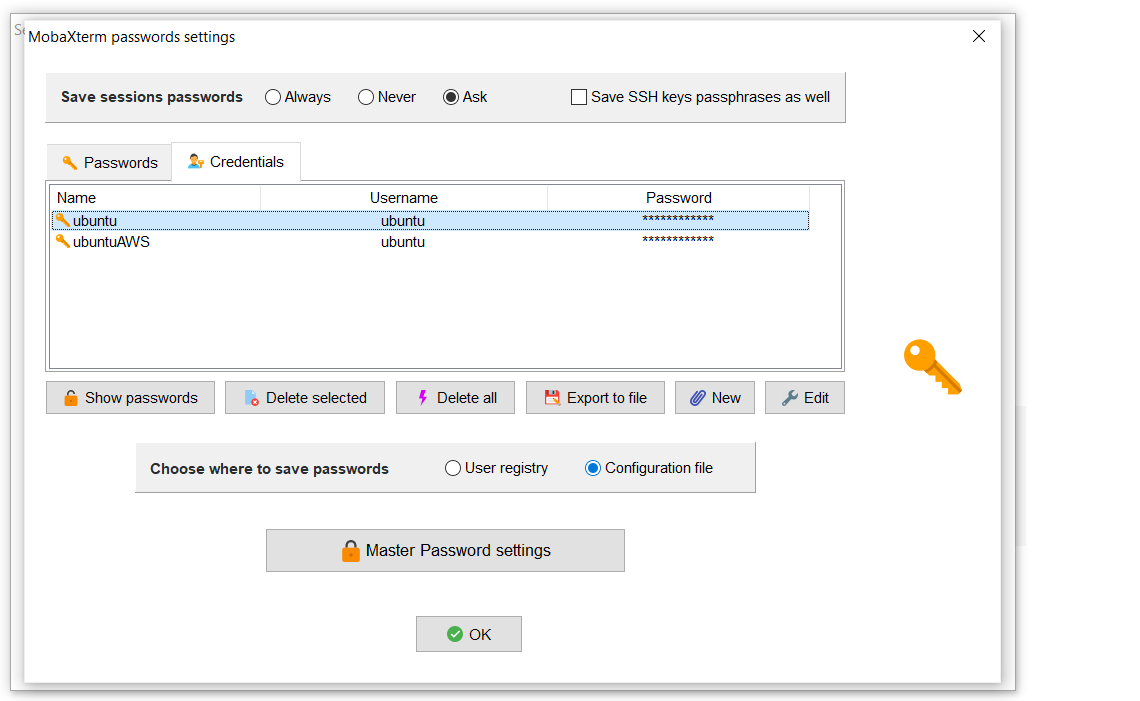
Datos del host de la maquina:



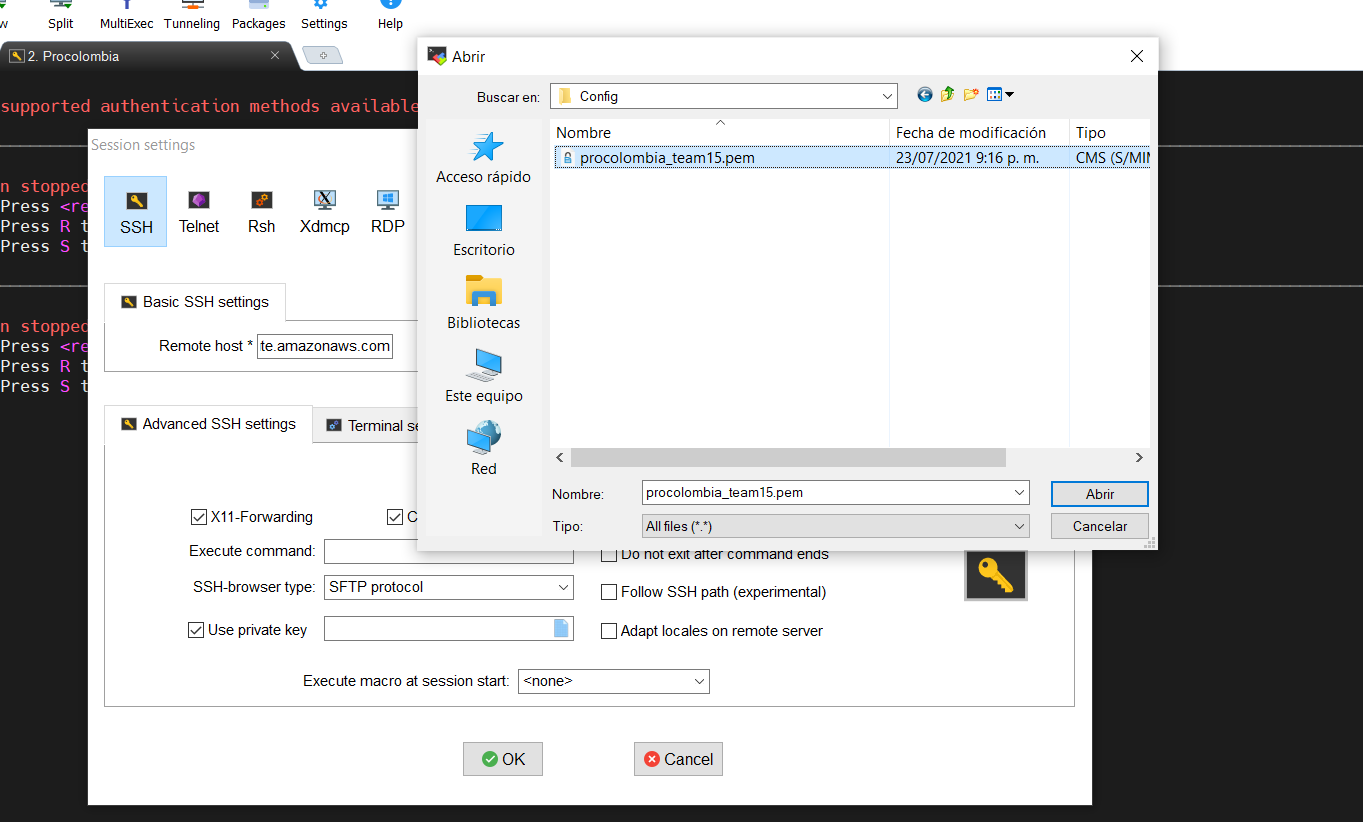
ec2-3-14-187-26.us-east-2.compute.amazonaws.com

Crear usuario ubuntu

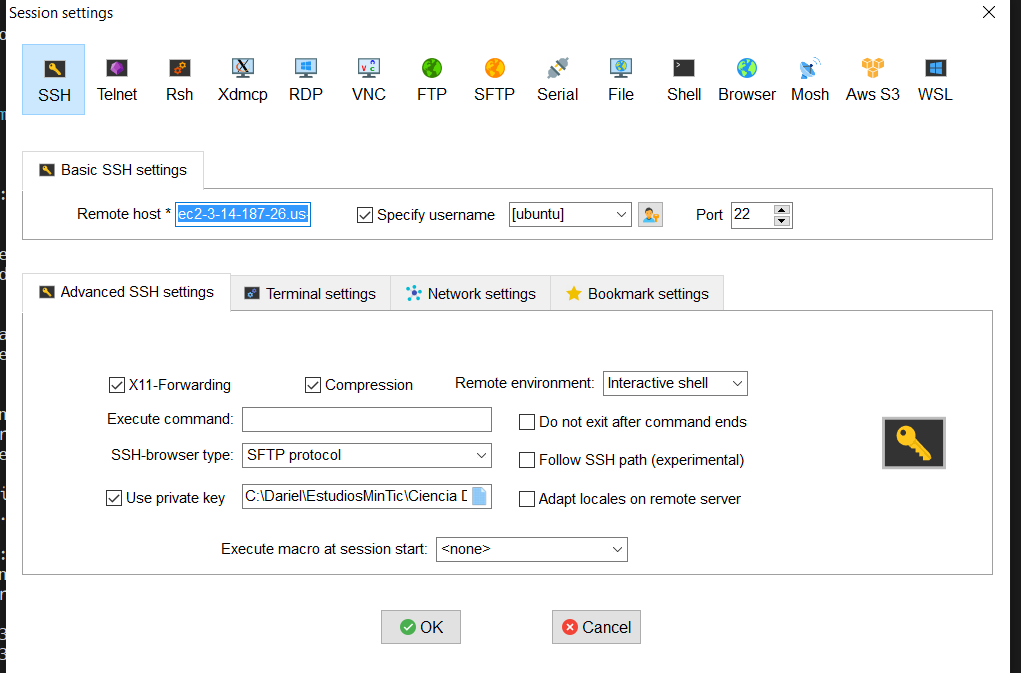
Clave ubuntu



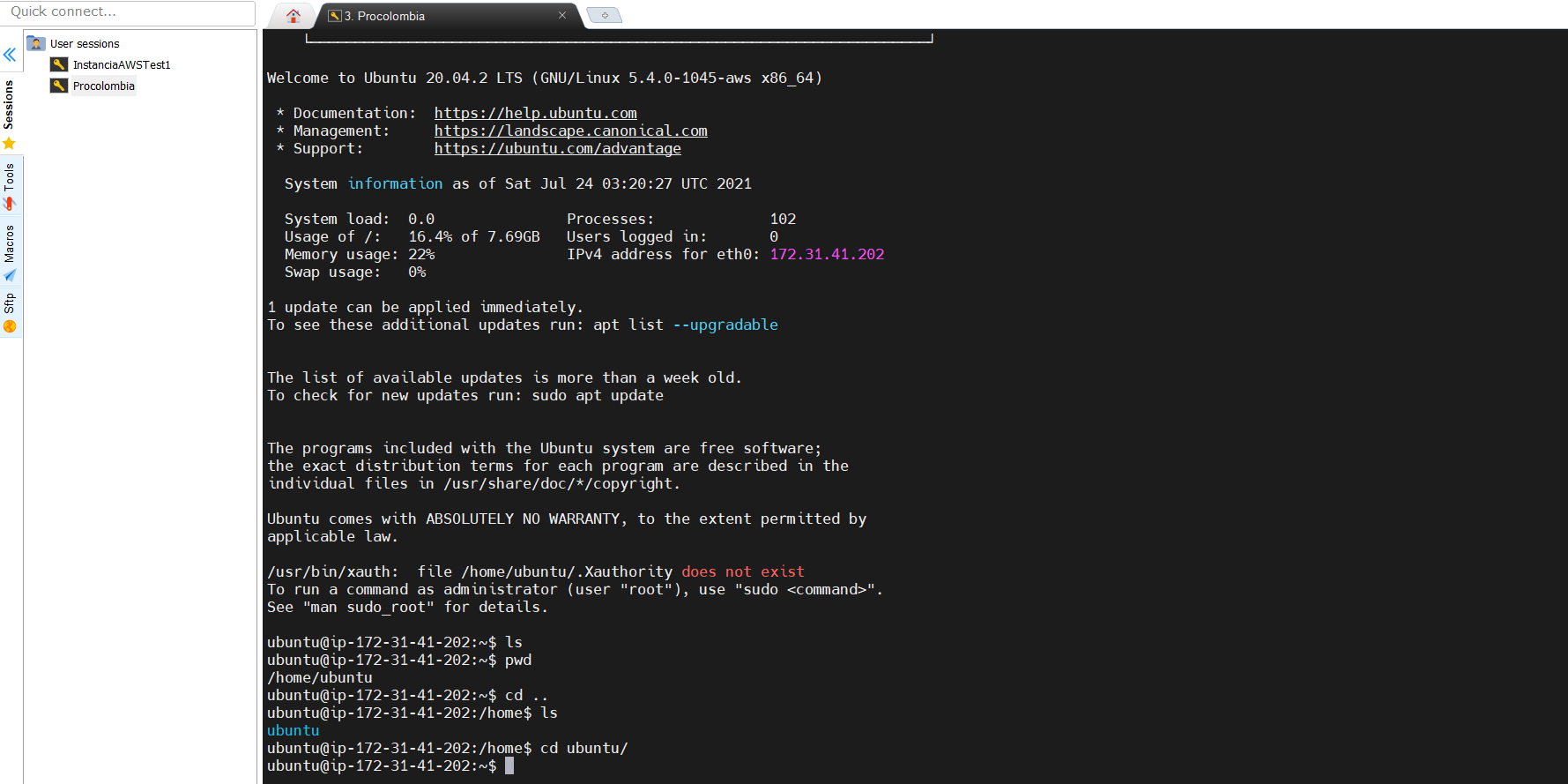
Seleccionar la llave privada:



Configuración final:



Conexión realizada a Procolombia:



Paso opcional si se desea hacer conexión con el workspace hacia la maquina de AWS:

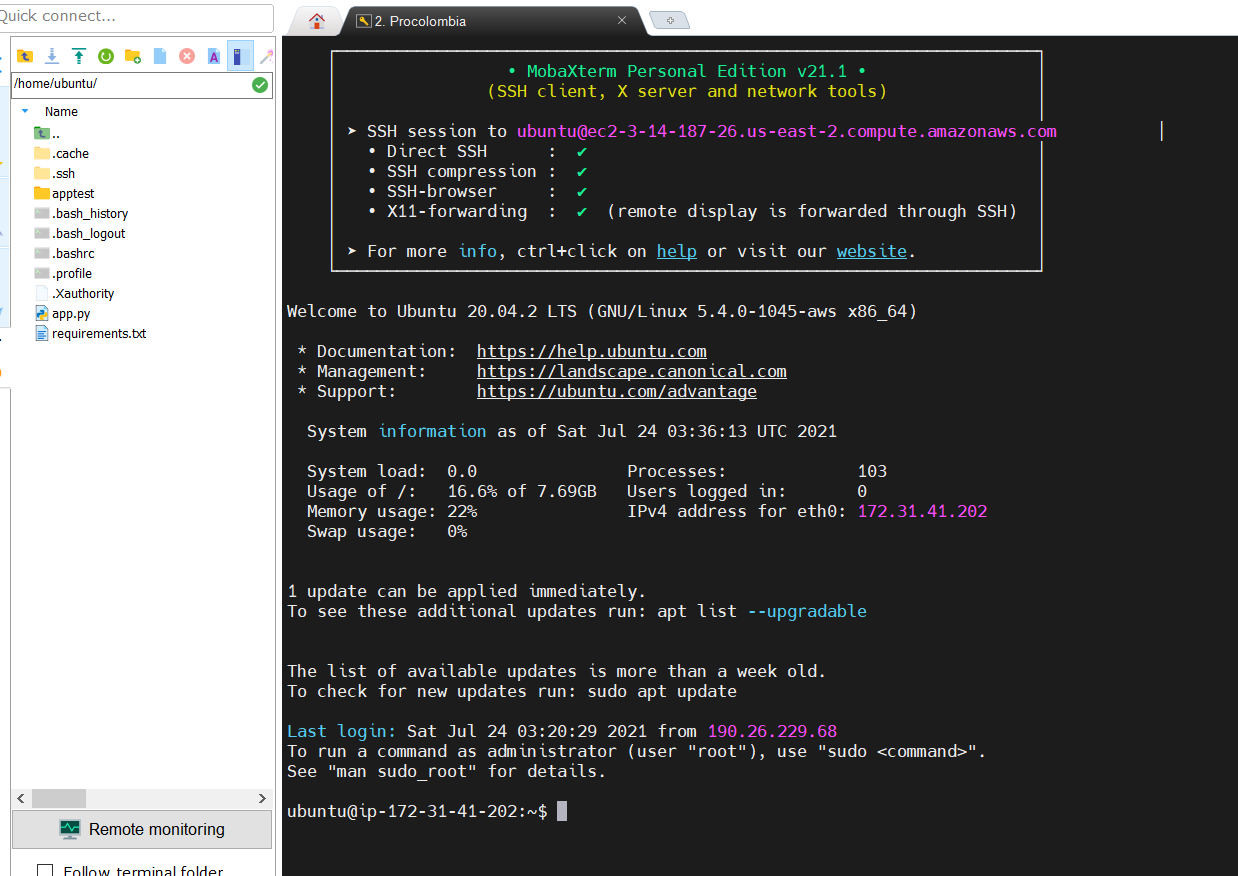
1. Se sube el archivo .pem
2. Se corre el siguiente comando:
3. Modificar el archivo *chmod 600 “nombre de archivo”*
4. Pasar la aplicación nombreAPP.py al servidor en amz:

scp -i nombre.pem app.py ubuntu@ 3.14.187.26;/home/Ubuntu

lo siguiente es Y

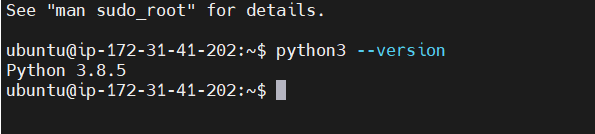
Ubicando la APP.py en la maquina:

Nos conectamos a la maquina y dejamos los archivos de requirement y el archivo .py:



Verificar versión de Python en el servidor:

python3 --version



Instalación de pip para Python:

sudo apt-get update

sudo apt-get upgrade

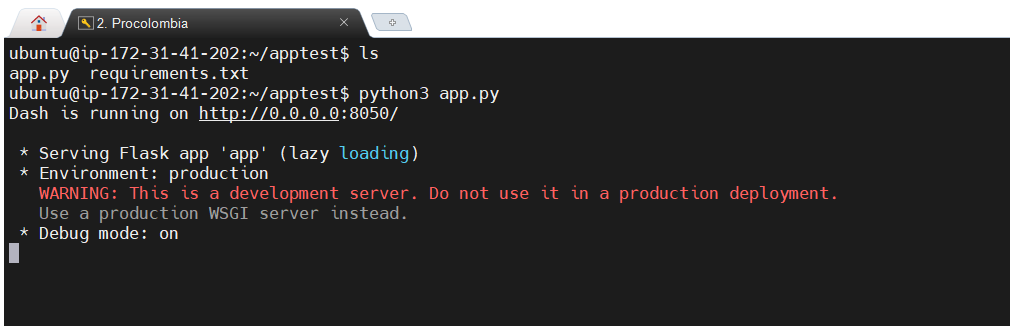
sudo apt install python3-pip

cargar librerias para la app:

pip install -r requirements.txt

Correr la app.py:

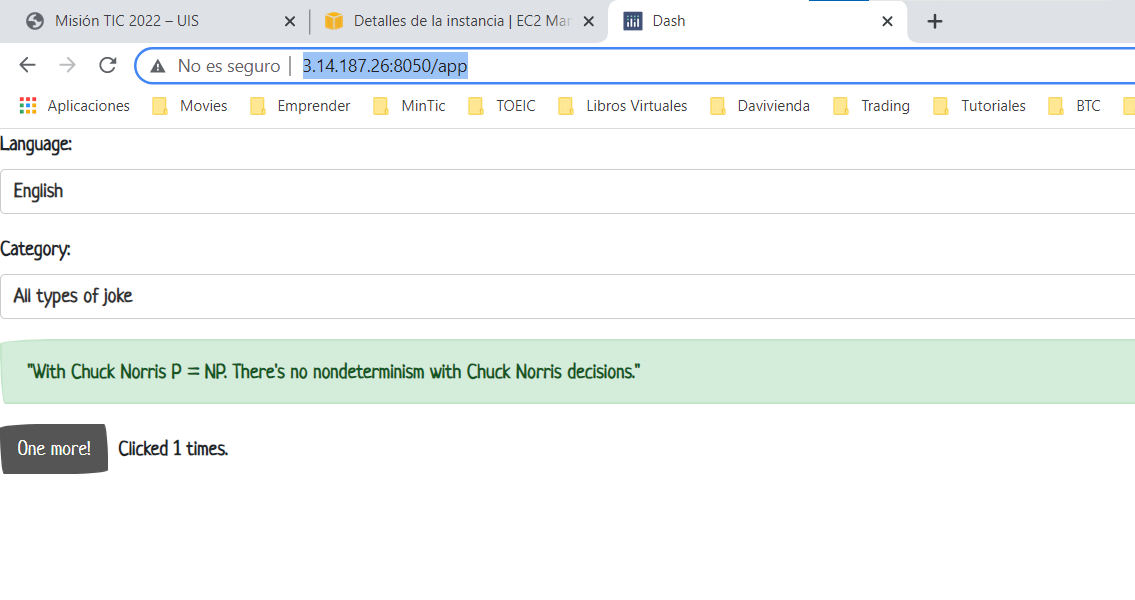
python3 app.py



Aplicación corriendo

<http://3.14.187.26:8050/app>

se ve la app:



## Installing the screen command

sudo apt install screen

*screen*

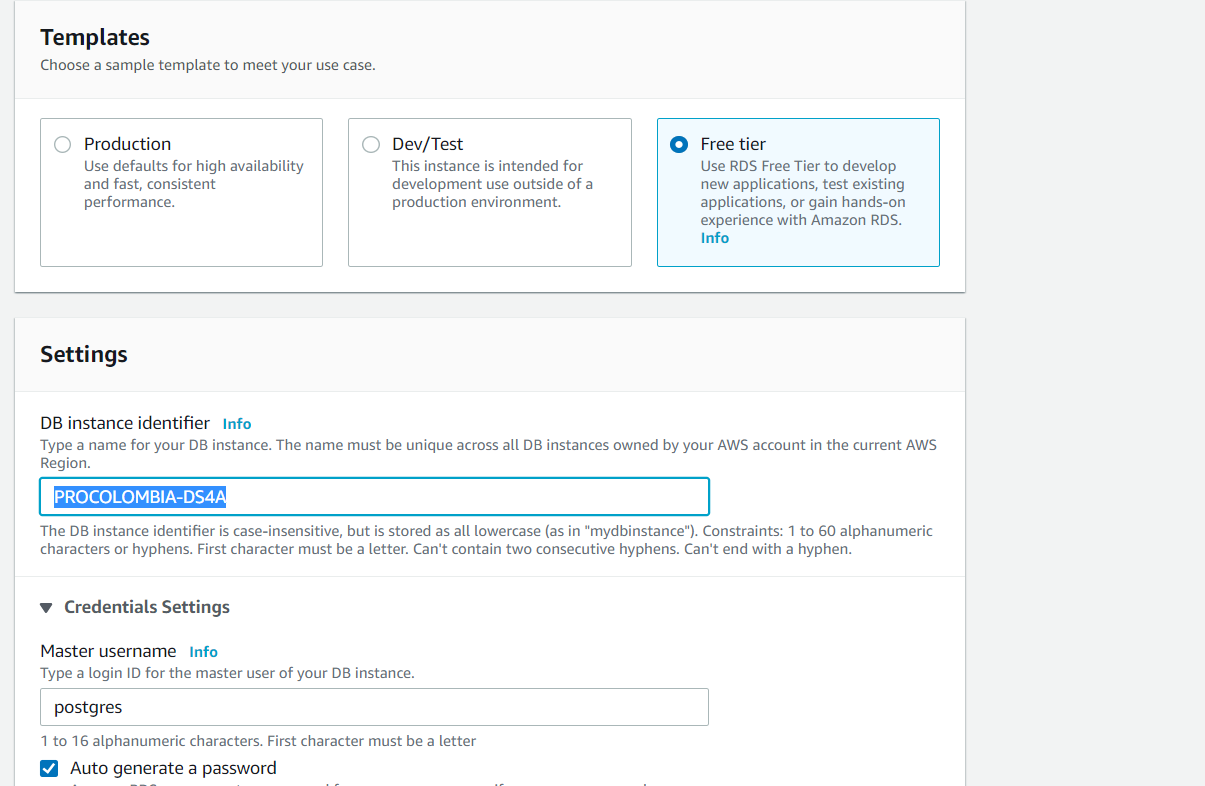
* *Ctrl a c* - Creates a new *screen* session so that you can use more than one *screen* session at once.
* *Ctrl a n* - Switches to the **n**ext *screen* session (if you use more than one).
* *Ctrl a p* - Switches to the **p**revious *screen* session (if you use more than one).
* *Ctrl a d* - Detaches a *screen* session (without killing the processes in it - they continue).
* To close a screen session where all tasks are finished you can type
* *exit*
* Now let's play around with it a little bit. In our *screen* window, we run the command
* *top*
* Now let's create another *screen* session by typing

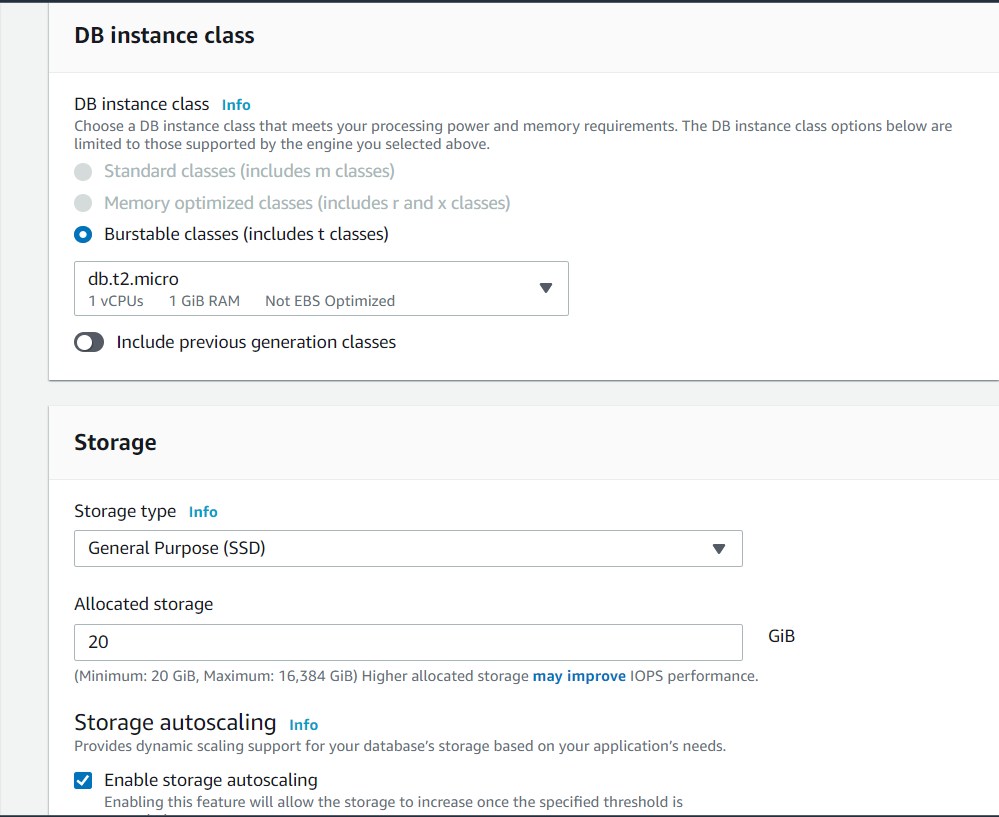
*Ctrl a c*

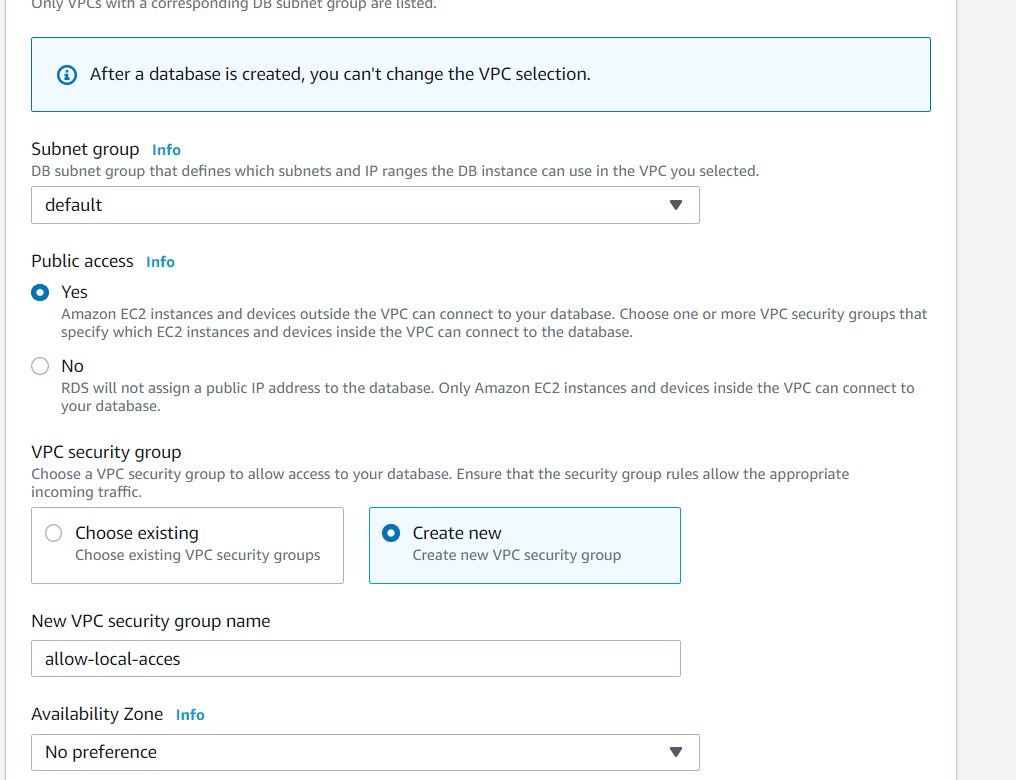
*tail -f /var/log/mail.log*

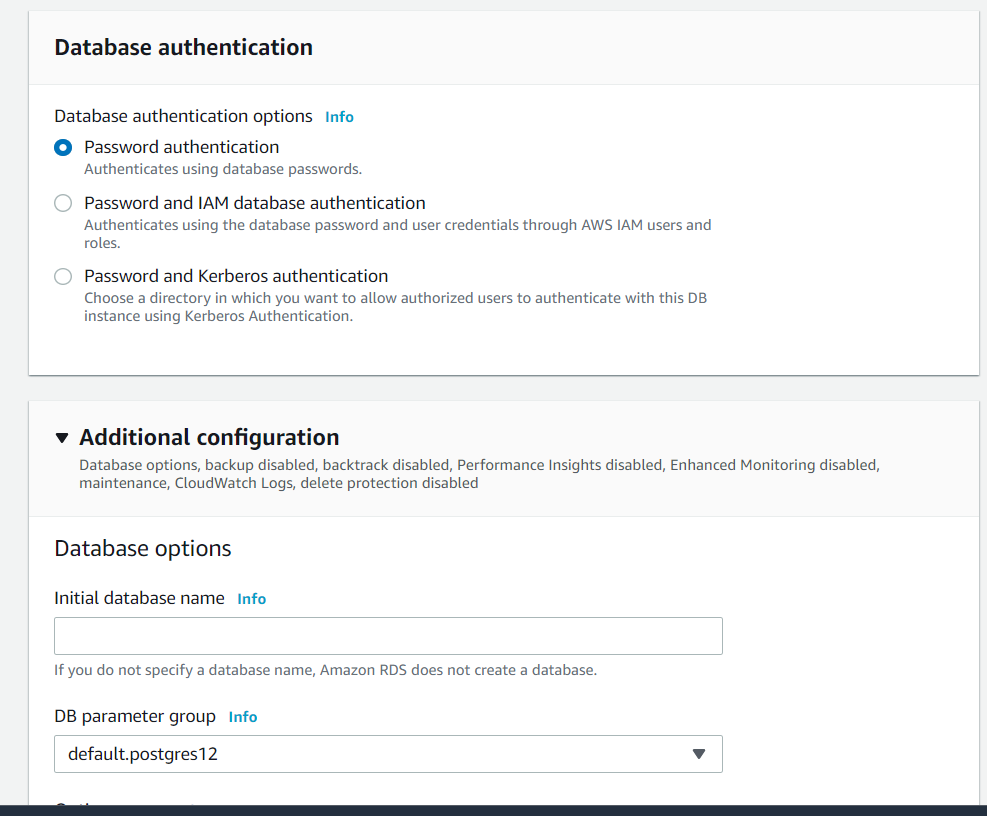
DataBase :

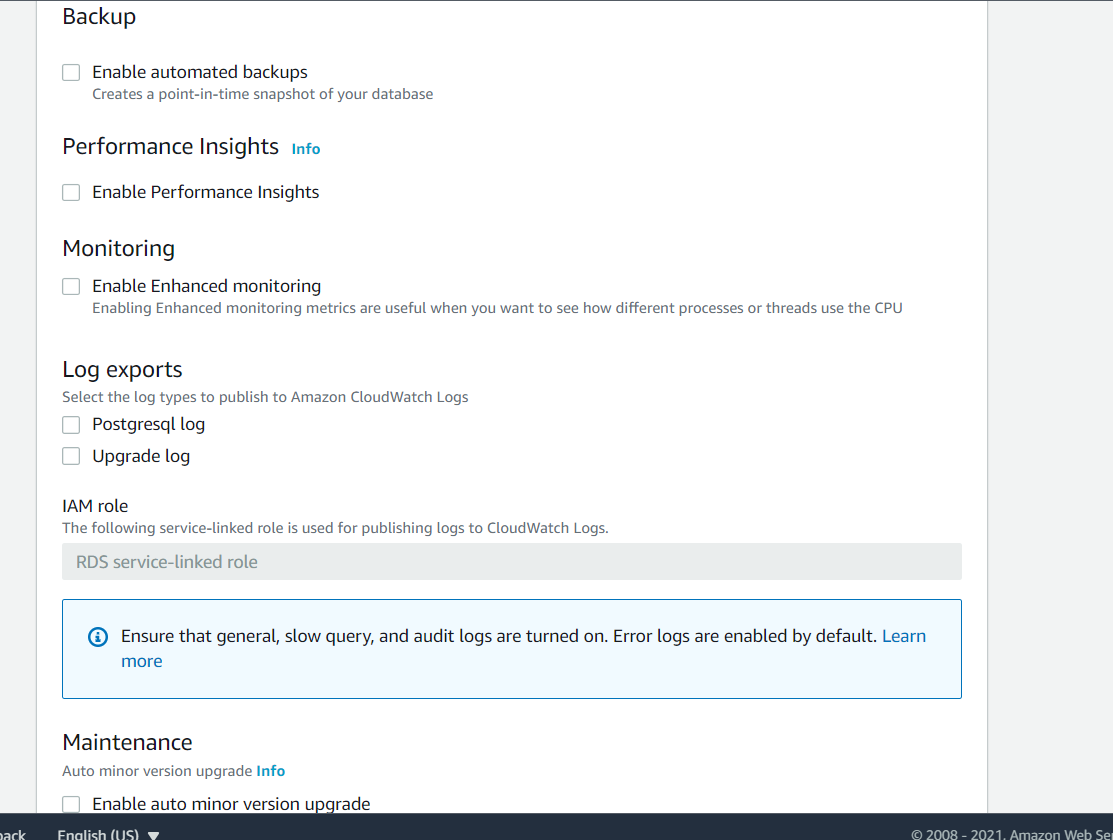
PROCOLOMBIA-DS4A







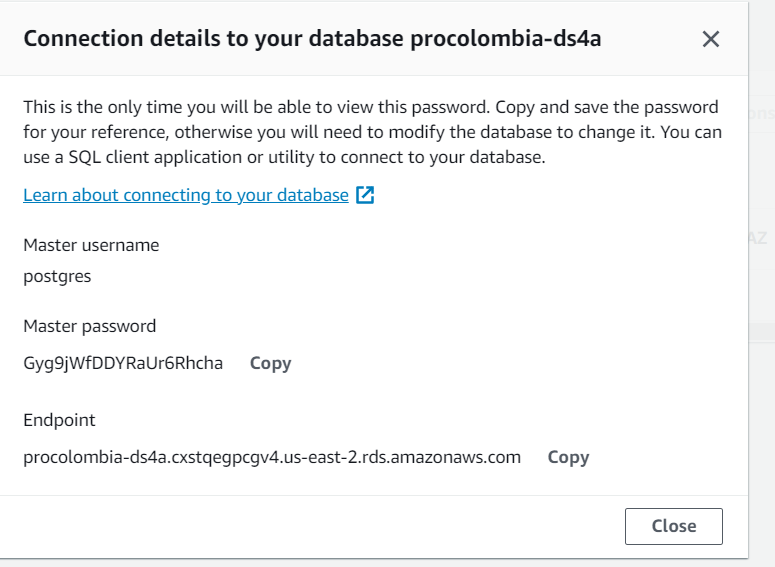




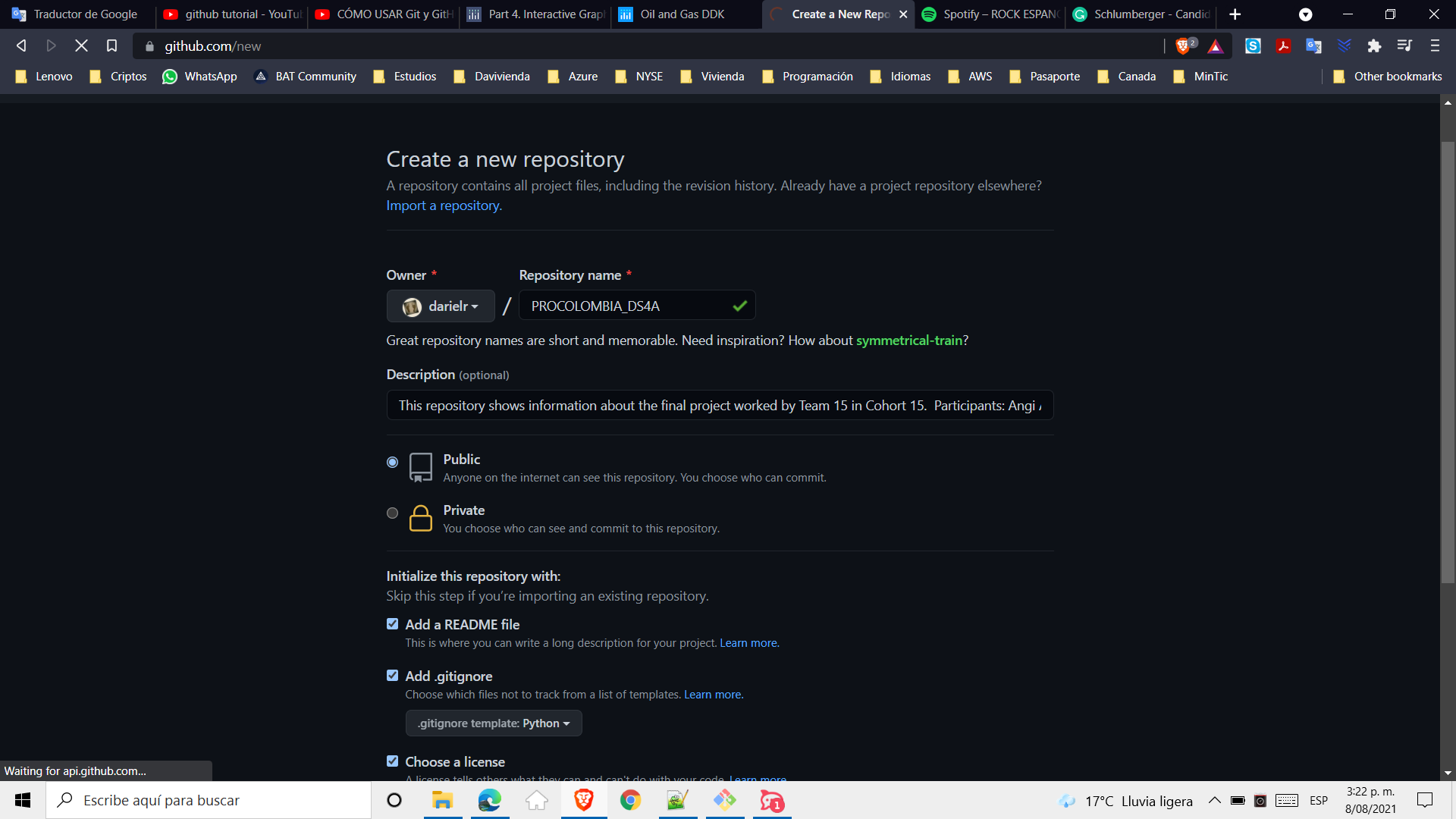
Gyg9jWfDDYRaUr6Rhcha

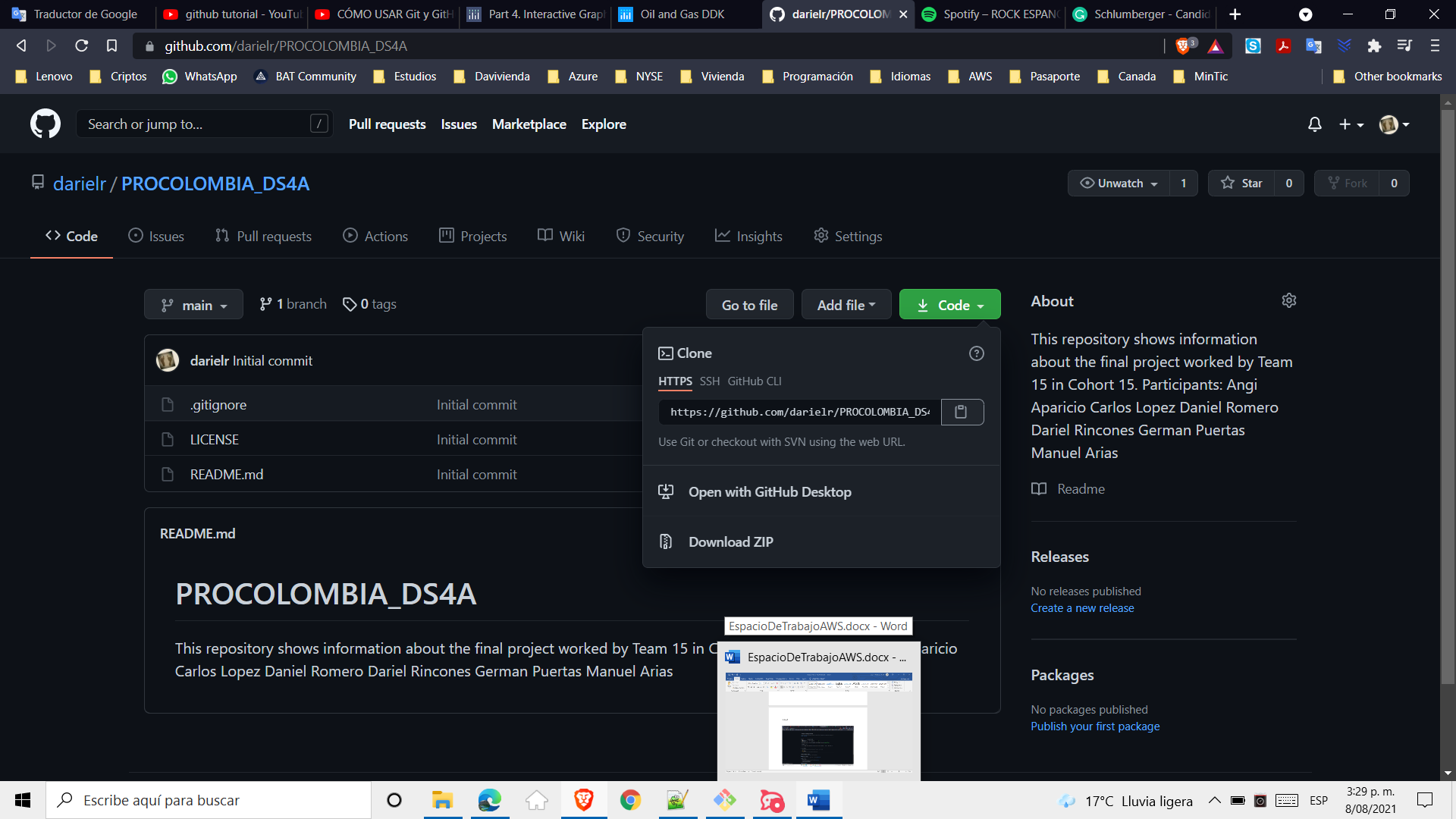
procolombia-ds4a.cxstqegpcgv4.us-east-2.rds.amazonaws.com

postgres



GITHUB





<https://github.com/darielr/PROCOLOMBIA_DS4A.git>

git clone <https://github.com/darielr/PROCOLOMBIA_DS4A.git>

git add “nombre de archivo.txt” -- adiciona los elementos al staging área local

git add . – adiciona todos los archivos modificados o nuevos

git commit – m ‘mis cambios’ -- sube los cambios con comentarios al repositorio local

git push --sube al repositorio de github

git diff -- muestra diferencias de los cambios a subir, si existen

git status -- muestra archivos con cambios o nuevos que no se han subido al repositorio

PENDIENTES:

* When you start your application, Dash shows this warning, understand what it implies:  
  WARNING: This is a development server. Do not use it in a production deployment.
* In a production environment, you must include digital certificates SSL, a domain name DNS, a web server like NGINX as a proxy, or a Web Server Gateway Interface like Gunicorn.
* Some additional security needs to be implemented on Dash to avoid unauthorized access to your dashboard and information. A basic way to do it is to include auth = dash\_auth.BasicAuth(app,USERNAMEINFO) [click here for more information](https://dash.plotly.com/authentication)
* Keep your server clean and organized by using [environments](https://pypi.org/project/pipenv/) and a clear folder and file structure.
* app.py only keeps running if you don't close the terminal running it on EC2. Be sure that your production deployment has a server configuration to keep running even if you lose your connection. E.g. sudo apt install screen

DASH AUTH

Para usuario contraseña:

