1. Amazon Elastic Compute Cloud (Amazon EC2)

Elastic Compute Cloud (EC2) proporciona capacidad informática escalable en la nube de Amazon Web Services (AWS) [1]. Es decir, nos permite tener capacidad de cómputo en la nube y lanzar una instancia o máquina virtual aprovisionándola de capacidad de procesamiento, de capacidad de memoria y de capacidad de almacenamiento. El uso de Amazon EC2 elimina la necesidad de invertir en hardware por adelantado, por lo que permite desarrollar e implementar aplicaciones más rápidamente, lanzar tantos servidores virtuales como se necesite, configurar la seguridad y las redes, administrar el almacenamiento, y escalar hacia arriba o hacia abajo para gestionar cambios en los requisitos o picos de popularidad, lo que reduce la necesidad de prever el tráfico [1].

1.1 Características

Amazon EC2 proporciona las siguientes características [1]:

- **Instancias**: entornos informáticos virtuales. Por ejemplo, una instancia de Amazon EC2 podría ser un servidor web que se ejecuta en un sistema operativo Linux y que se utiliza para alojar un sitio web.
- **Plantillas preconfiguradas para las instancias**, conocidas como *Amazon Machine Images (AMI)*, que empaquetan los bits que se necesitan para un servidor (incluido el sistema operativo y el software adicional). Por ejemplo, una *AMI* podría incluir un sistema operativo Linux, un servidor web y una aplicación web determinada.
- **Tipos de instancia** (*instance types*): varias configuraciones de CPU, memoria, almacenamiento y capacidad de red para las instancias. De esta manera, se puede escoger el tipo de instancia que se adapte mejor a los requisitos de cada aplicación.
- **Información** de **inicio** de **sesión seguro para** tus **instancias** mediante pares de claves (*key pairs*): AWS almacena la clave pública y tú almacenas la clave privada en un lugar seguro en tu equipo local.
- **Volúmenes de almacenamiento de instancias** (*instance store volumes*)*:* volúmenes de almacenamiento para datos temporales que se eliminan cuando detienes, hibernas o finalizas tu instancia. Por ejemplo, se pueden utilizar volúmenes de almacenamiento de instancias para almacenar los archivos temporales generados por una aplicación.
- **Volúmenes de Amazon EBS** (*Amazon EBS volumes*): volúmenes de almacenamiento persistente para tus datos mediante Amazon Elastic Block Store (Amazon EBS). Se utilizan para almacenar datos que deban persistir incluso cuando la instancia se detenga, hiberne o finalice. Por ejemplo, se podrían utilizar para almacenar los archivos de base de datos de una aplicación.
- Varias ubicaciones físicas para tus recursos, como instancias y volúmenes de Amazon EBS, conocidas como **regiones y zonas de disponibilidad** (*Regions and Availability Zones*). Poder lanzar instancias de Amazon EC2 y volúmenes de Amazon EBS en diferentes regiones y zonas de disponibilidad permite protegerse frente a errores en una ubicación específica y ayuda a garantizar la durabilidad de las aplicaciones.
- Un ***firewall*** que permite especificar los protocolos, puertos y rangos de IP de origen que pueden llegar a tus instancias mediante grupos de seguridad (*security groups*). Por ejemplo, se pueden utilizar grupos de seguridad para controlar el tráfico de red entrante y saliente a las instancias de Amazon EC2.
- **Direcciones IP elásticas** (*elastic IP addresses*): direcciones IPv4 estáticas para la computación dinámica en la nube. Por ejemplo, utilizar las direcciones IP elásticas para asignar una dirección IPv4 estática a una instancia de Amazon EC2 puede ser útil si se necesita acceder a la instancia utilizando una dirección IP estática, o si se necesita reasignar la dirección a una instancia diferente.

- **Metadatos**, conocidos como etiquetas (*tags*), que puedes crear y asignar a tus recursos de Amazon EC2. Por ejemplo, se pueden utilizar etiquetas para identificar la finalidad, el propietario o el entorno de un recurso, o para agrupar recursos con características similares y administrarlos como una única entidad.
- **Nubes virtuales privadas** (*virtual private clouds* o VPCs): se pueden crear redes virtuales que están lógicamente aisladas del resto de la nube de AWS. Opcionalmente, se puede conectar la VPC a una red propia, utilizando una conexión VPN o AWS Direct Connect, para crear un entorno de nube híbrida. Esto puede resultar útil si se desea ampliar una red local a la nube o si se desea aislar los recursos en la nube de la Internet pública.

1.2 Casos de uso

Teniendo en cuenta todas las características anteriores, podemos concluir que Amazon EC2 es un servicio de computación en nube versátil y potente que puede utilizarse para una amplia gama de aplicaciones y cargas de trabajo. Existen muchos casos de uso potenciales para Amazon EC2. Algunos de los más importantes son:

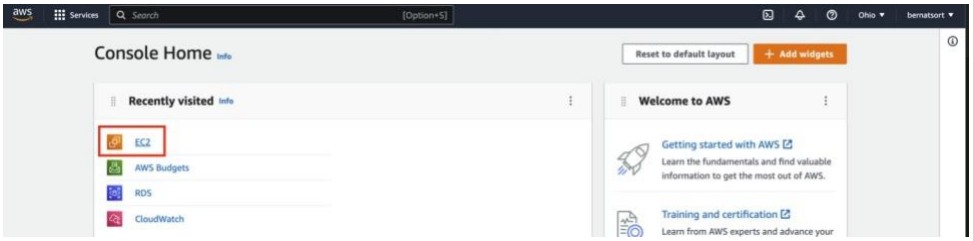
- ***Hosting* de webs y aplicaciones.**
  EC2 puede utilizarse para alojar aplicaciones web, incluidos sitios y servicios web dinámicos, proporcionando recursos informáticos escalables y fiables para estas cargas de trabajo.
- **Procesamiento y análisis de datos.**
  EC2 se puede utilizar para ejecutar cargas de trabajo de procesamiento y análisis de datos a gran escala, lo que permite a las organizaciones obtener información de sus datos de forma rápida y sencilla.
- ***Machine Learning* e Inteligencia Artificial (IA).**
  Otro caso en que se puede utilizar EC2 es para entrenar e implementar modelos de aprendizaje automático (*Machine Learning)* y para ejecutar aplicaciones y servicios impulsados por IA.
- ***Hosting* de bases de datos.**
  EC2 también se puede utilizar para alojar bases de datos relacionales y no relacionales, como MySQL, PostgreSQL y MongoDB, proporcionando almacenamiento escalable y confiable para estas cargas de trabajo.
- ***DevOps* e integración/entrega continua.**
  Con EC2 se pueden ejecutar canalizaciones de integración continua y entrega continua (*CI/CD*), que automatizan el proceso de creación, prueba y despliegue de software, lo que permite a las organizaciones lanzar nuevas versiones de sus aplicaciones de forma rápida y eficiente.
- **Recuperación de desastres.**
  EC2 puede utilizarse como parte de una estrategia de recuperación de desastres para configurar rápidamente un nuevo entorno en la nube, lo que permite a las organizaciones recuperarse de un desastre o interrupción y volver a poner en funcionamiento rápidamente los sistemas y datos. Por ejemplo, se puede crear una *AMI* del entorno de producción y almacenarla en Amazon S3, para que en caso de desastre o interrupción se pueda lanzar rápidamente una nueva instancia EC2 utilizando la *AMI* y se recupere ese entorno más rápidamente.

1.3 Deployment de una aplicación Flask en AWS EC2

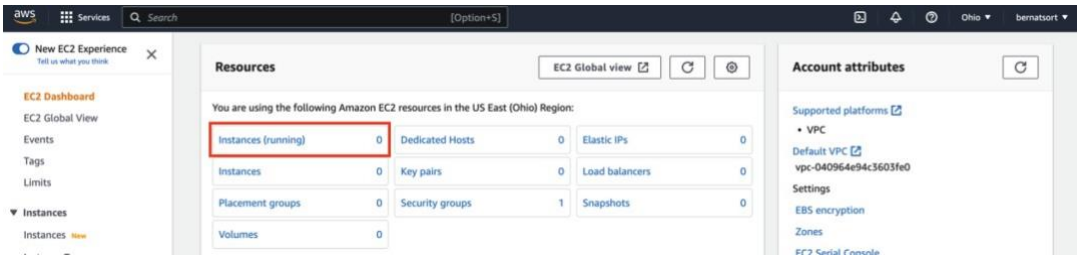
Una vez visto y entendido que es AWS EC2, vamos a desplegar una aplicación Flask en una instancia Amazon EC2. Por tanto, vamos a configurar una máquina remota que nos permitirá visualizar el sitio web.

1.3.1 Crear una cuenta en AWS

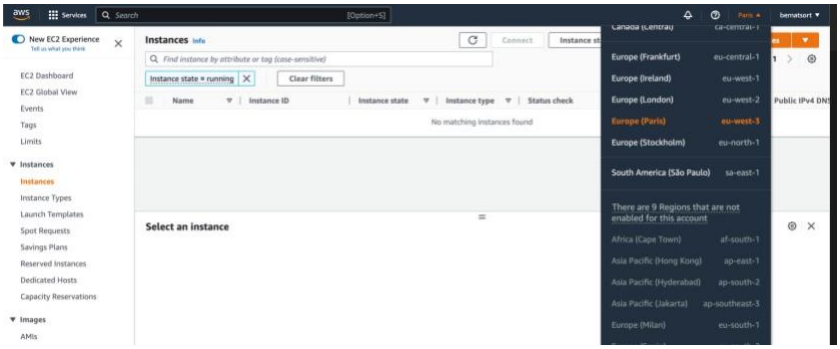
Lo primero que haremos será crear una cuenta en AWS. Una vez creada, se nos abre la *Console Home* y seleccionamos EC2.



Observamos que no tenemos ninguna instancia corriendo:

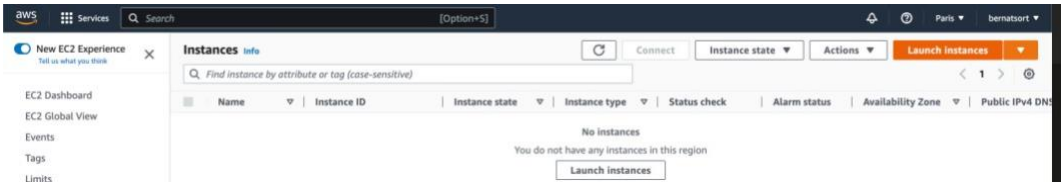


Seleccionamos la región (la que esté más cerca de nuestra ubicación). En nuestro caso es Europa (París), ya que España no está disponible para nuestro tipo de cuenta

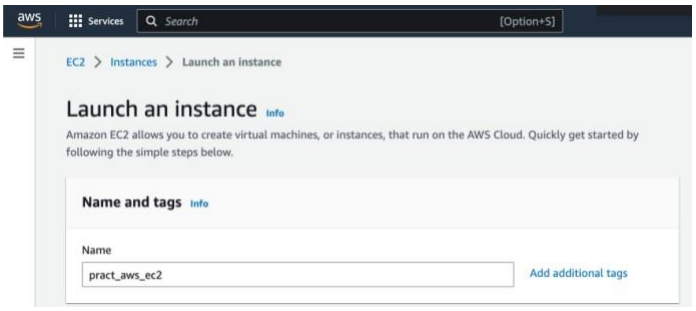


1.3.2 Lanzar una instancia de Amazon EC2

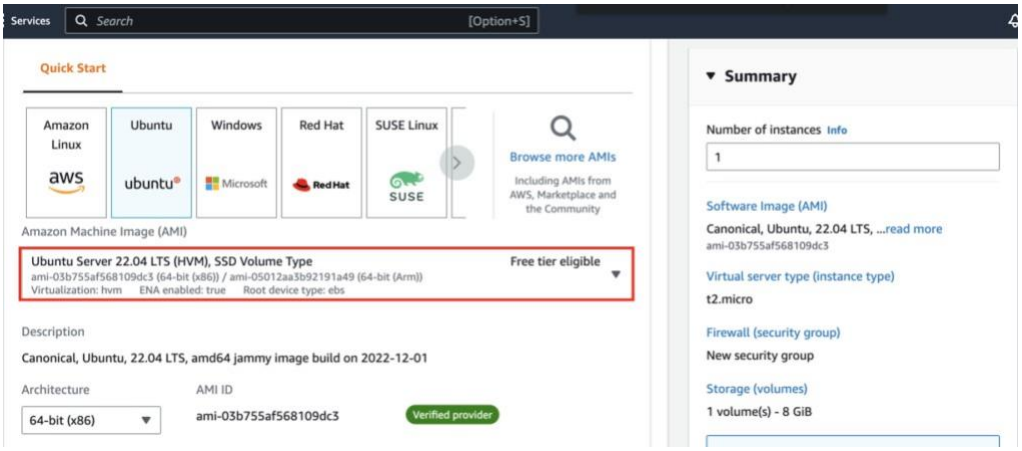
Queremos lanzar una instancia. Por tanto, vamos a la EC2 Dashboard y en *Instances* seleccionamos *Launch instance.*



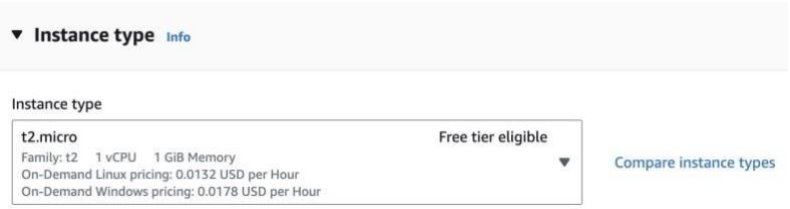
Le asignamos un nombre. En nuestro caso se llamará *pract_aws_ec2*.



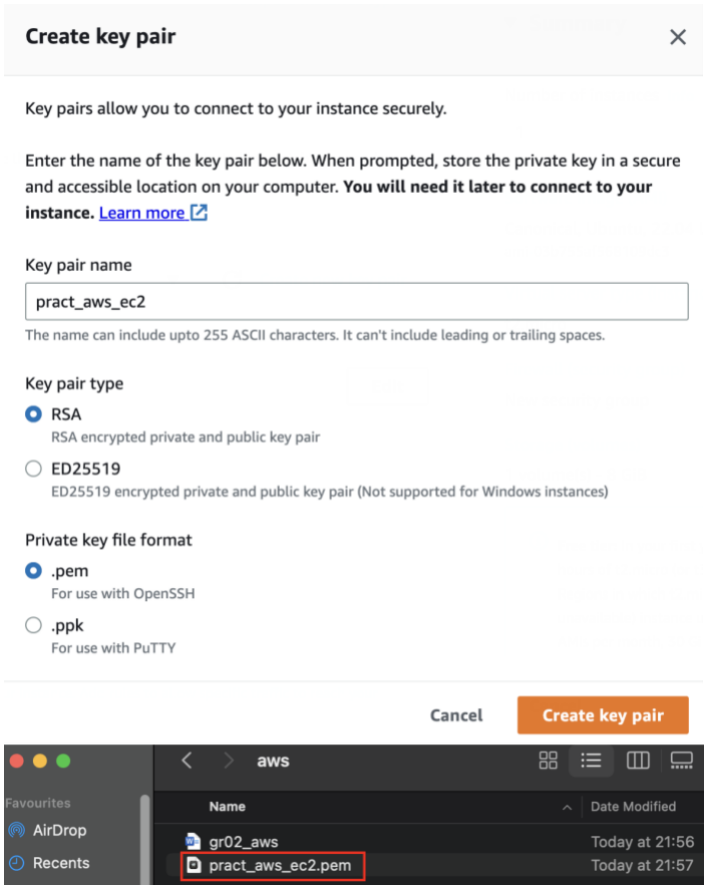
Seleccionamos una *Amazon Machine Image (AMI)*, por ejemplo, Ubuntu Server 22.04. Nos debemos asegurar que es *free tier elegible:* podemos tener un servidor en marcha durante un año entero si pagar nada.



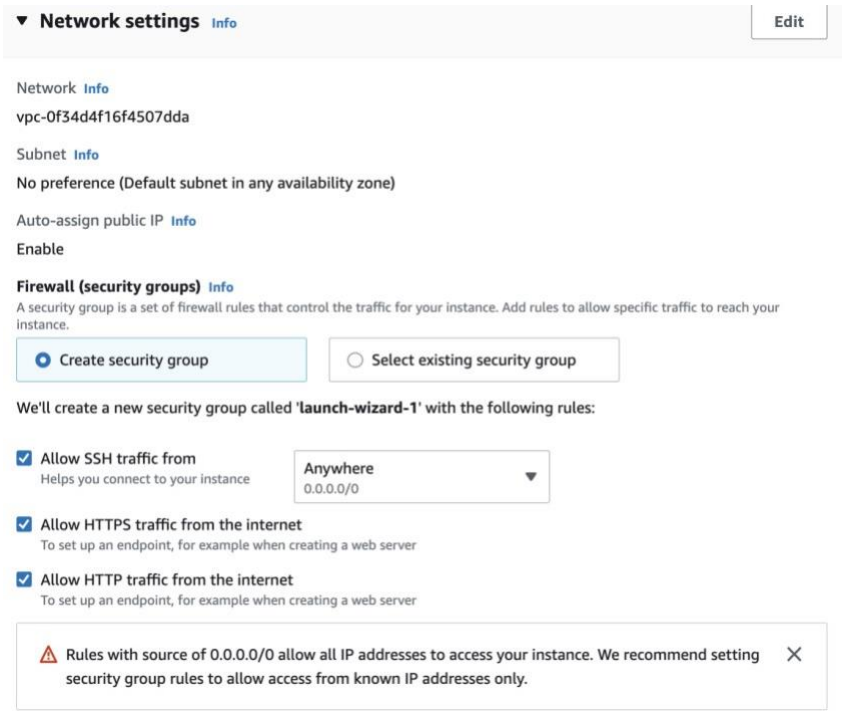
El tipo de instancia va a ser *t2.micro*, ya que al ser máquinas pequeñas no usarán mucha memoria ni potencia computacional y por tanto serán muy baratas. No obstante, deberemos asegurarnos de apagar nuestras instancias cuando no las estemos usando.



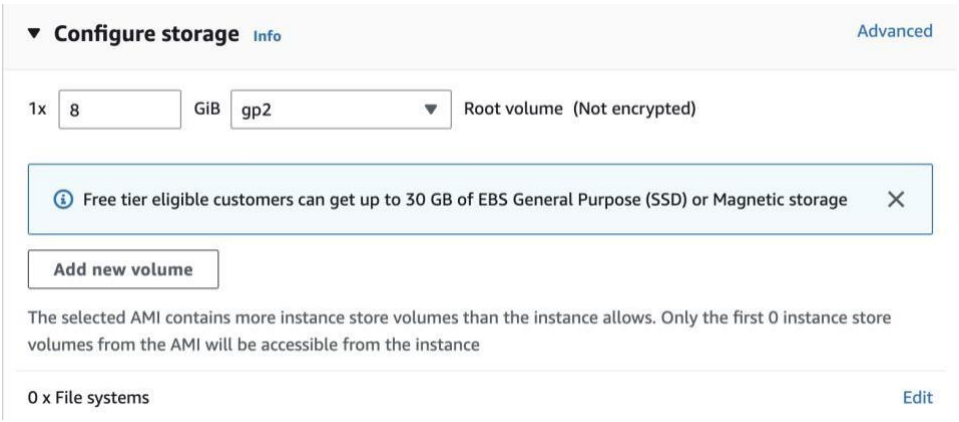
Para conectarnos de forma segura a nuestra instancia necesitamos una *key pair.* Creamos una nueva *key pair* y la guardamos en nuestra máquina local:



Una vez creada nuestra clave privada, seleccionamos las configuraciones de red:

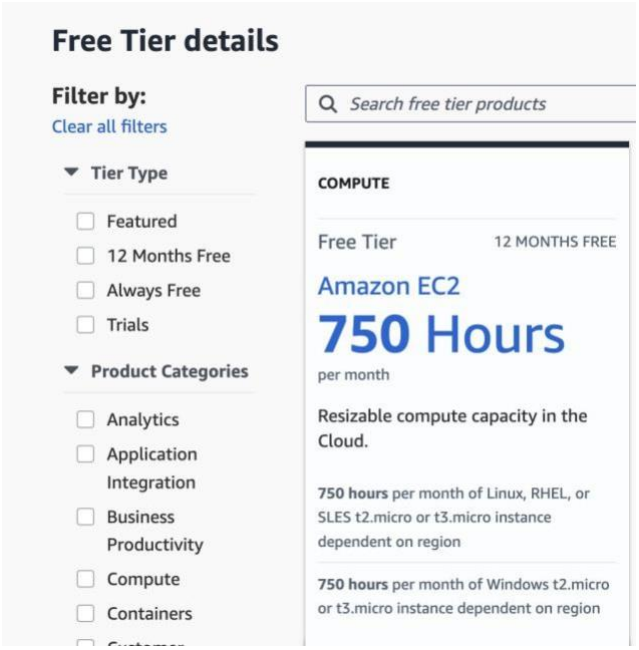


Configuramos el almacenamiento. Al cumplir con los requisitos de la capa gratuita (*free tier elegible*) podemos obtener hasta 30 GB de almacenamiento EBS de propósito general (SSD) o magnético, pero para el propósito de esta práctica lo dejaremos en 8 GB.

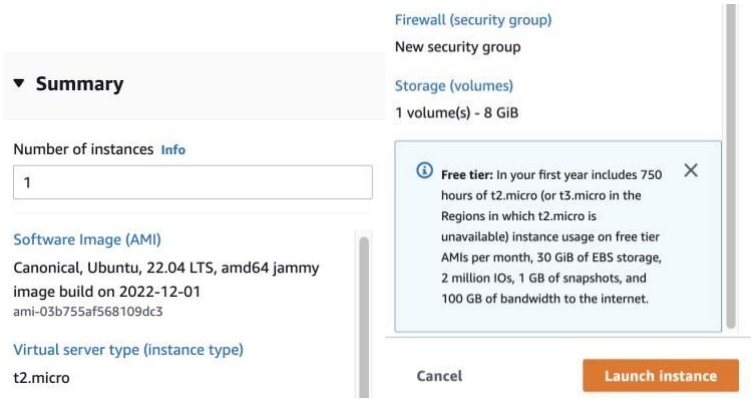


En este enlace podemos encontrar la información y los detalles de la *free tier:* https://aws.amazon.com/free.

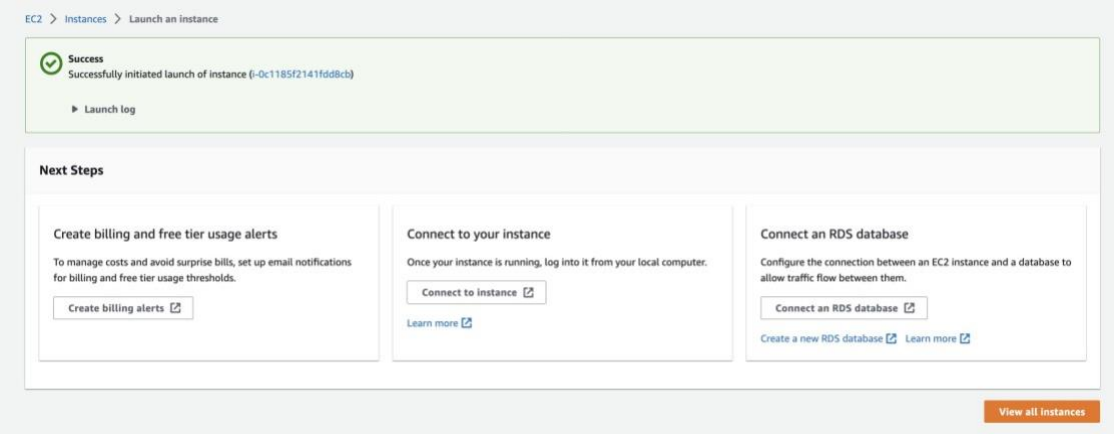
Observamos que para Amazon EC2 tenemos 750 horas al mes de instancias Windows *t2.micro* o *t3.micro* dependiendo de la región y 750 horas al mes de instancias Linux, RHEL o SLES *t2.micro* o *t3.micro* dependiendo de la región.



En resumen, tenemos lo siguiente:

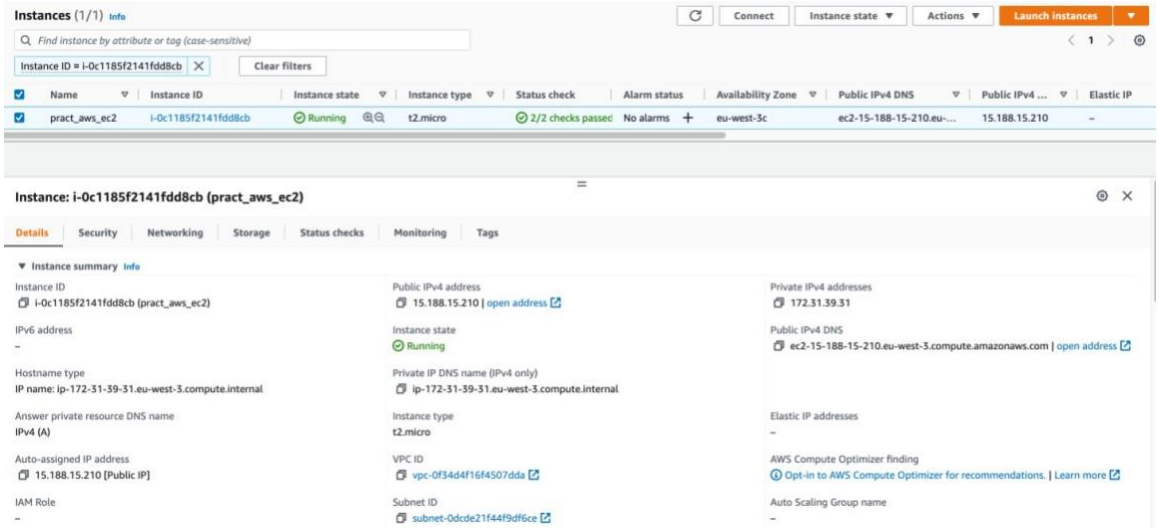


Finalmente, lanzamos la instancia:

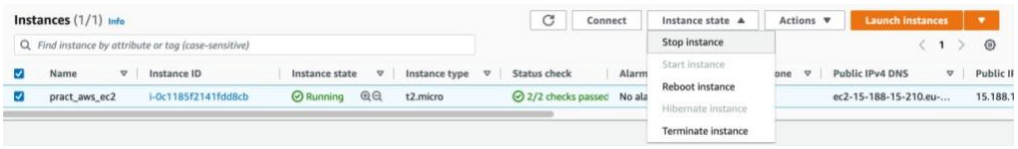


Nuestra instancia ya está corriendo:

La seleccionamos y nos aparece información del servidor:



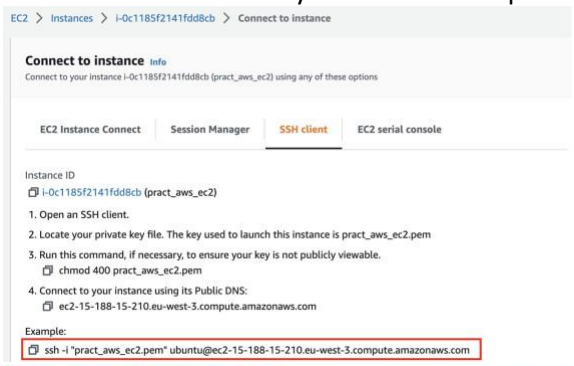
Si queremos detener la instancia, en *Instance rate* clicamos *Stop instance.*

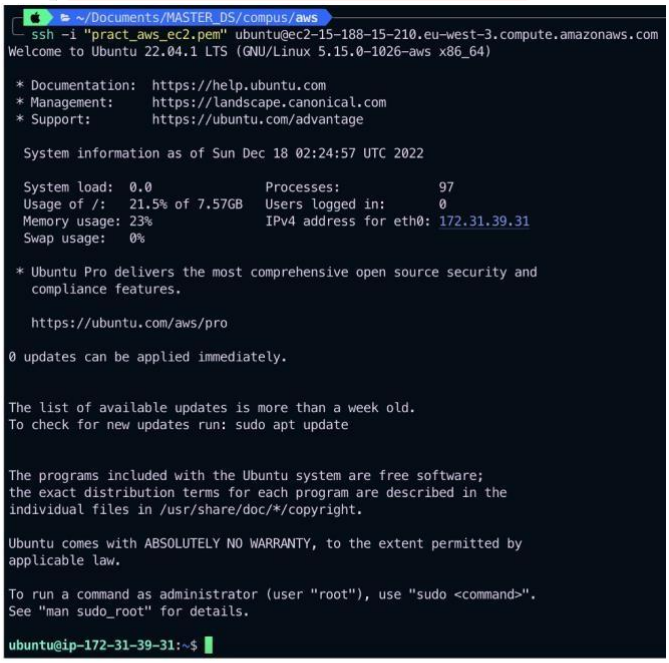


1.3.3 Conectarse a una instancia

Ahora que tenemos la instancia corriendo, nos podemos conectar a esa instancia. Nos conectaremos usando la terminal. En la terminal vamos a la carpeta donde hayamos guardado la clave privada y ejecutamos los siguientes comandos:

- Comando que se utiliza para cambiar los permisos de un archivo llamado *pract_aws_ec2.pem*. El comando *chmod* se utiliza para cambiar los permisos de un archivo o directorio en Linux y otros sistemas operativos tipo Unix.





- **ssh** para conectarnos a la instancia.
- **-i** para indicar el nombre del archivo: **pract_aws_ec2.pem**.
- **ubuntu**: el nombre del servidor que hemos usado.
- Añadimos una **@**.
- Proporcionamos nuestro DNS público o IPv4: **ec2-15-188-15-210.eu-west-3.compute.amazonaws.com**

Ahora ya estamos conectados a un nuevo ordenador.

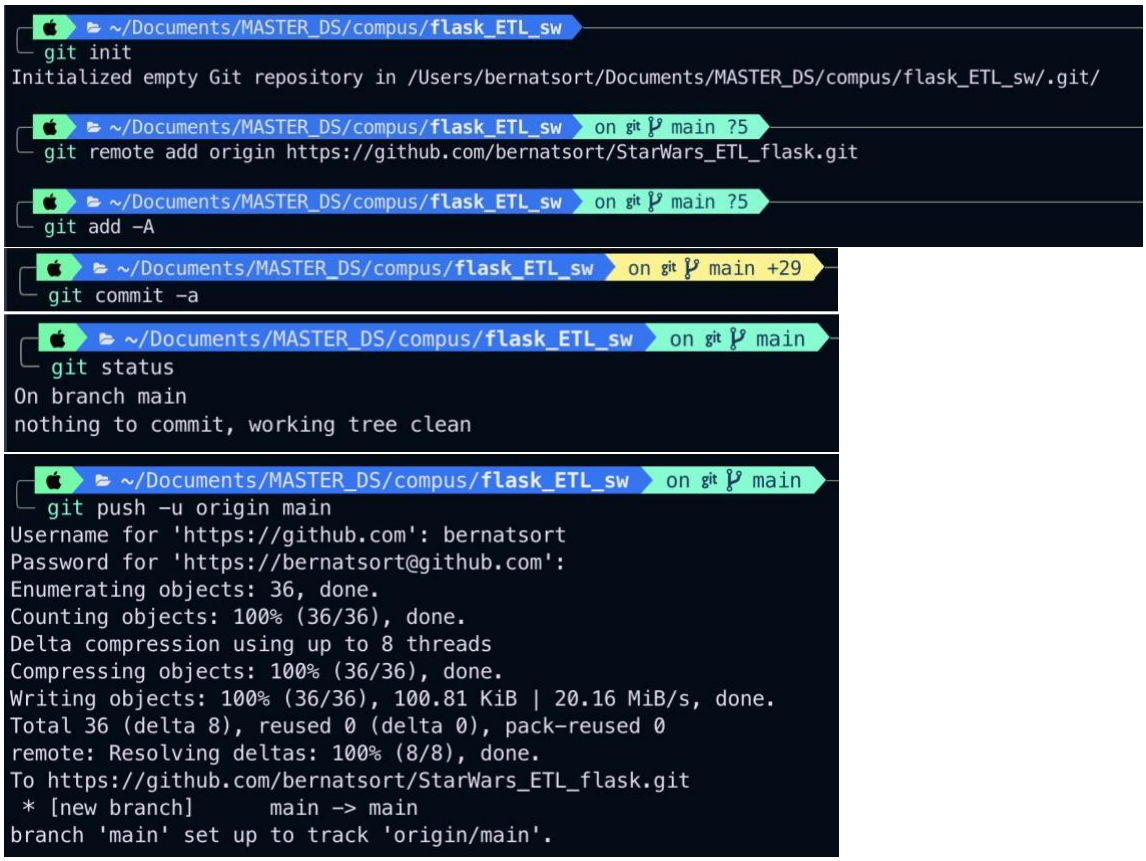
1.3.4 Despliegue de la aplicación en Amazon EC2

Ahora que tenemos nuestra instancia en ejecución en AWS y tenemos todos nuestros archivos configurados, es el momento de poner todos nuestros archivos online.

1.3.4.1 Transferencia de archivos a AWS – Git

Queremos subir todos nuestros archivos Flask a nuestra instancia EC2. Para ello, vamos a GitHub y creamos un nuevo repositorio llamado StarWars_ETL_ flask.

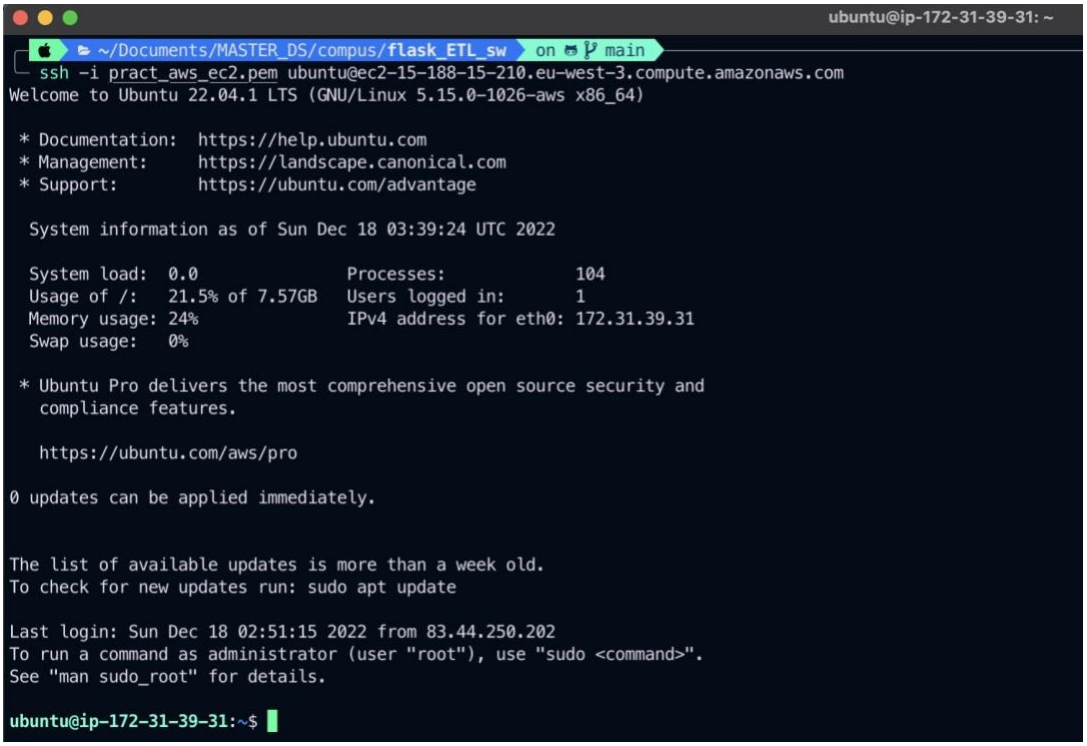
Copiamos el link: https://github.com/bernatsort/StarWars_ETL_flask.git y subimos el proyecto a GitHub:



Ahora todo está en nuestro GitHub. Borramos la clave privada del repositorio ya que al guardarla en la misma carpeta del proyecto se subió a GitHub.

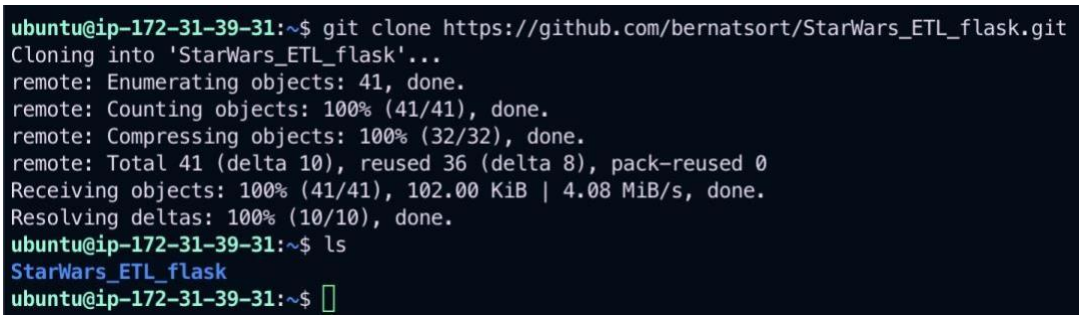
1.3.4.2 SSH

Ahora que hemos transferido todos nuestros archivos desde nuestro ordenador a GitHub, queremos acceder a nuestro GitHub a través de nuestra instancia EC2. Esto significa que necesitamos SSH.



Ya hemos sido capaces de iniciar sesión en nuestra instancia.

Clonamos el repositorio y podemos ver como tenemos nuestra carpeta del proyecto:

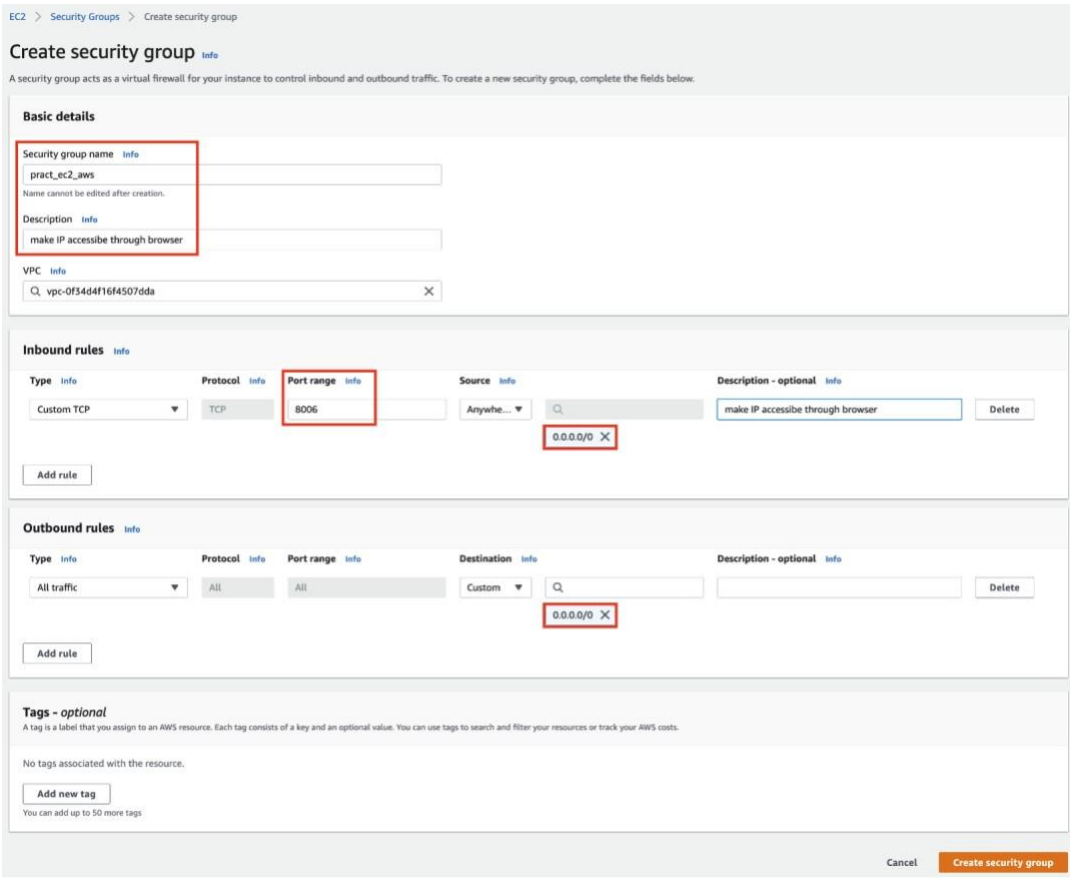


Llegados a este punto, hemos tomado con éxito nuestros archivos de nuestra máquina local, los hemos subido a GitHub y hemos sido capaces de obtenerlos de GitHub utilizando SSH en nuestra instancia.

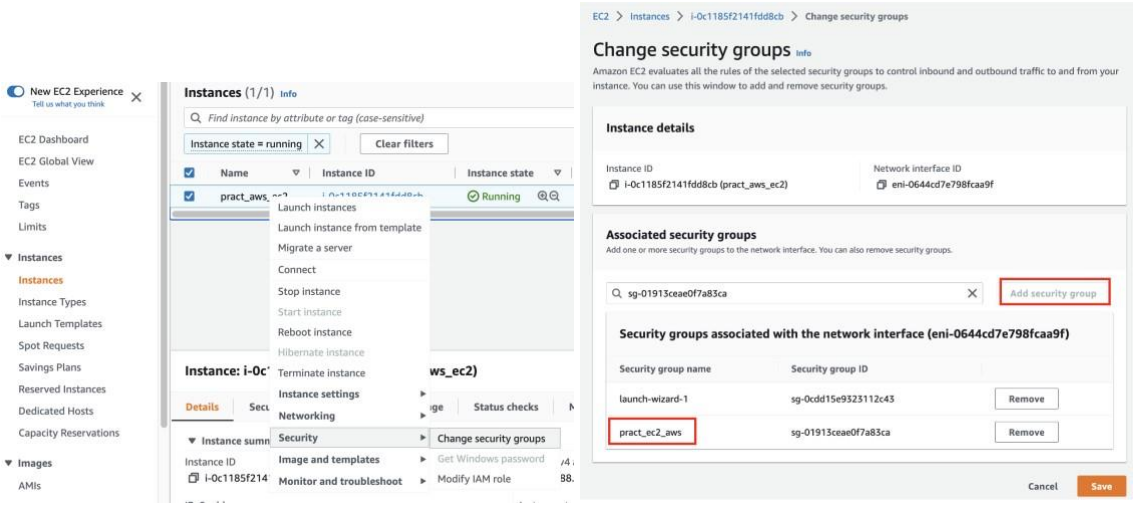
1.3.4.3 Nuestra aplicación web online

Ahora estamos listos para tener nuestro proyecto Flask en vivo en Internet con nuestra IP pública.

Creamos un *security group*:



Una vez creado, volvemos a nuestra instancia, cambiamos el *security group* y añadimos el que hemos creado:



Actualizamos lo necesario para instalar lo que necesitemos en nuestra instancia:

```
ubuntu@ip-172-31-39-31:~$ sudo apt-get update
```

Instalamos Python 3:

```
ubuntu@ip-172-31-39-31:~$ sudo apt-get install python3-pip
```

Instalamos las dependencias del propyecto:

```
ubuntu@ip-172-31-39-31:~/StarWars_ETL_flask$ pip3 install -r requirements.txt
```

Modificamos el puerto acorde con el que hemos especificado en el *security group* usando vim:

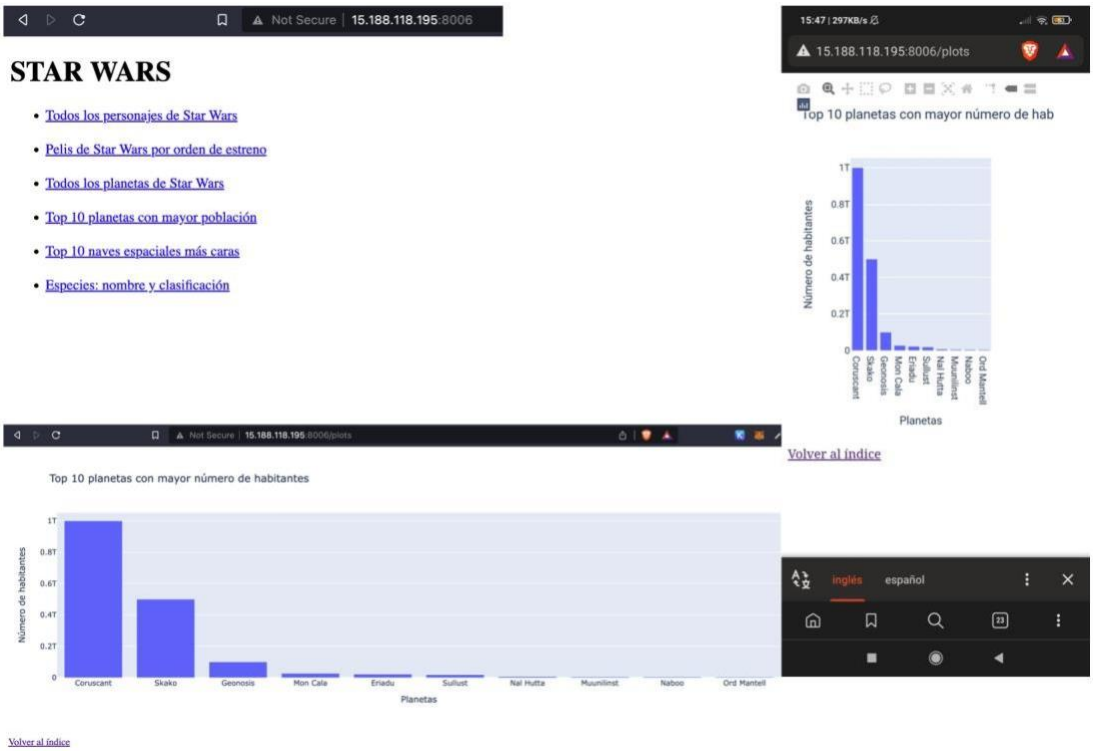
```
^Cubuntu@ip-172-31-39-31:~/StarWars_ETL_flask/src$ ls
__pycache__  app.py  db  extract.py  load.py  models.py  plots  templates  transform.py
ubuntu@ip-172-31-39-31:~/StarWars_ETL_flask/src$ vim app.py
```

```
if __name__ == '__main__':
    # llamamos al método app.run() para iniciar la
    app.run(host="0.0.0.0", port=8006, debug=True)
```

```
ubuntu@ip-172-31-39-31:~/StarWars_ETL_flask/src$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8006
 * Running on http://172.31.39.31:8006
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 868-566-258
83.44.250.202 - - [18/Dec/2022 14:41:17] "GET / HTTP/1.1" 200 -
83.44.250.202 - - [18/Dec/2022 14:41:17] "GET /favicon.ico HTTP/1.1" 404 -
```

El enlace es el siguiente: http://15.188.118.195:8006/.

- IP pública (dirección IPv4): 15.188.118.195
- Puerto: 8006



Podemos observar que todo funciona correctamente, tanto en el ordenador como en un *smartphone*. Tenemos un sitio web que funciona y que la gente puede visitar desde cualquier parte del mundo.

Por el momento tenemos que utilizar el comando *python3 app.py*, y solo funciona mientras se está ejecutando en nuestra terminal. Así que si hacemos CTR + C no va a funcionar porque el sitio web ya no se está ejecutando el servidor.

Si quisiéramos que siempre estuviese funcionando como una página web normal en cualquier navegador, podríamos usar *Gunicorn*.

2. Referencias

[1] Amazon Web Services, Inc. What is Amazon EC2? Retrieved December 15, 2022, from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html