

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:

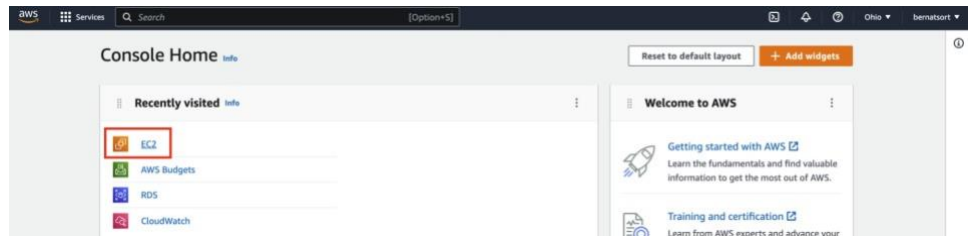
- **Hosting de webs y aplicaciones.**
EC2 puede utilizarse para alojar aplicaciones web, incluidos sitios y servicios web dinámicos, proporcionando recursos informáticos escalables y fiables para estas cargas de trabajo.
- **Procesamiento y análisis de datos.**
EC2 se puede utilizar para ejecutar cargas 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

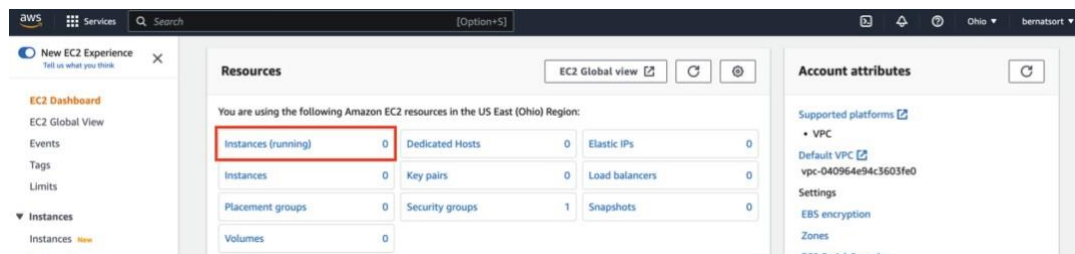
Una vez visto y entendido que es AWS EC2, vamos a desplegar una aplicación Flask en una instancia Amazon EC2. Por tanto, vamos a configurar una máquina remota que nos permitirá visualizar el sitio web.

1.3.1 Crear una cuenta en AWS

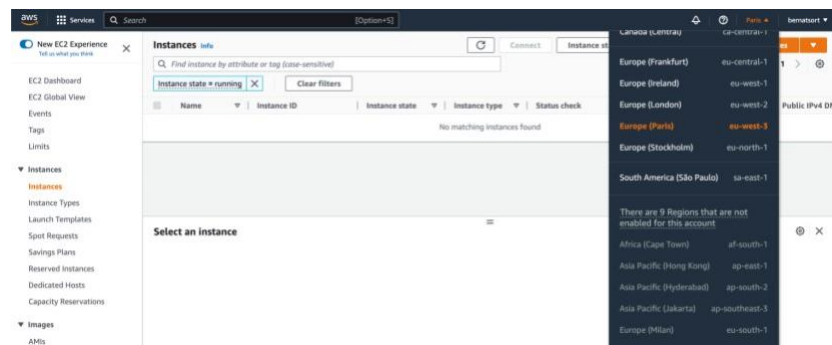
Lo primero que haremos será crear una cuenta en AWS. Una vez creada, se nos abre la *Console Home* y seleccionamos EC2.



Observamos que no tenemos ninguna instancia corriendo:

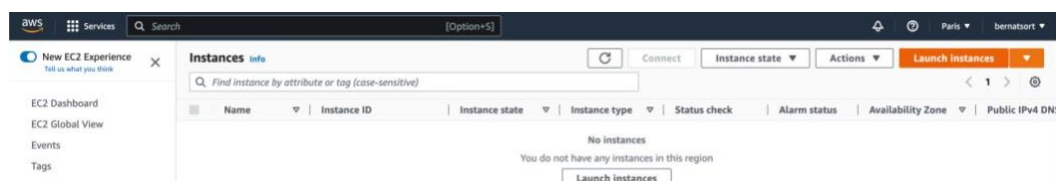


Seleccionamos la región (la que esté más cerca de nuestra ubicación). En nuestro caso es Europa (París), ya que España no está disponible para nuestro tipo de cuenta

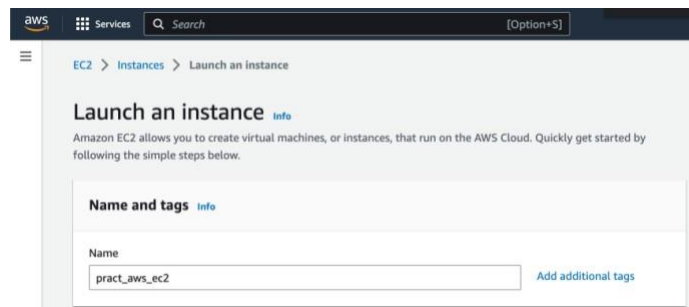


1.3.2 Lanzar una instancia de Amazon EC2

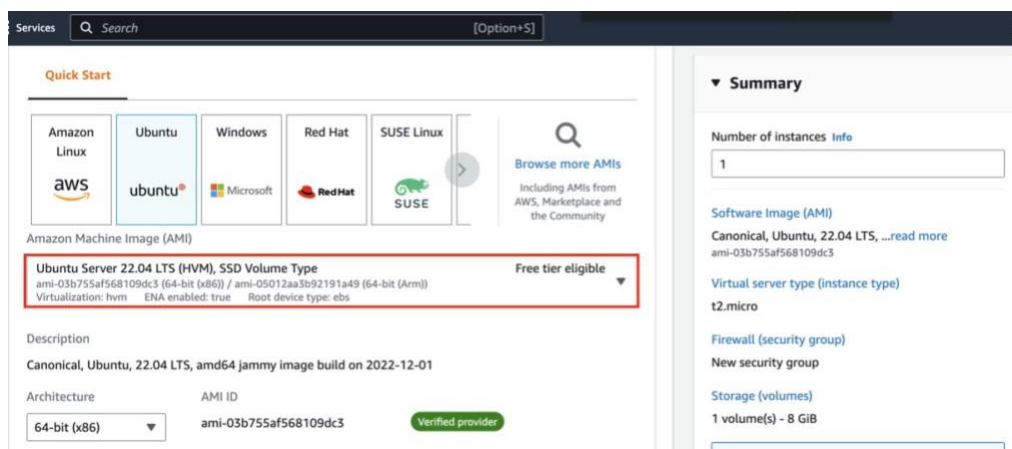
Queremos lanzar una instancia. Por tanto, vamos a la EC2 Dashboard y en *Instances* seleccionamos *Launch instance*.



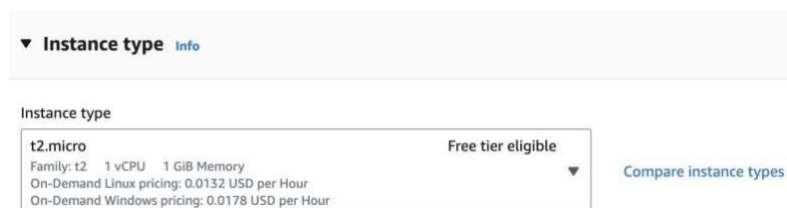
Le asignamos un nombre. En nuestro caso se llamará *pract_aws_ec2*.



Seleccionamos una *Amazon Machine Image (AMI)*, por ejemplo, Ubuntu Server 22.04. Nos debemos asegurar que es *free tier eligible*: 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:

Create key pair

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.**
[Learn more](#)

Key pair name

pract_aws_ec2

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

☒ RSA
RSA encrypted private and public key pair

☐ ED25519
ED25519 encrypted private and public key pair (Not supported for Windows instances)

Private key file format

☒ .pem
For use with OpenSSH

☐ .ppk
For use with PuTTY

Cancel

Create key pair

aws

Name	Date Modified
gr02_aws	Today at 21:56
pract_aws_ec2.pem	Today at 21:57

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

▼ Network settings

Info

Edit

Network
Info
vpc-0f34d4f16f4507dda

Subnet
Info
No preference (Default subnet in any availability zone)

Auto-assign public IP
Info
Enable

Firewall (security groups)
Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group
☐ Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

☒ Allow SSH traffic from
Helps you connect to your instance

Anywhere
0.0.0.0/0

☒ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

⚠

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

×

Configuramos el almacenamiento. Al cumplir con los requisitos de la capa gratuita (*free tier eligible*) 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.

5

▼ **Configure storage** [Info](#) Advanced

1x GiB Root volume (Not encrypted)

ℹ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage ✕

[Add new volume](#)

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

0 x File systems [Edit](#)

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.

Free Tier details

Filter by:
[Clear all filters](#)

▼ Tier Type

- ☐ Featured
- ☐ 12 Months Free
- ☐ Always Free
- ☐ Trials

▼ Product Categories

- ☐ Analytics
- ☐ Application Integration
- ☐ Business Productivity
- ☐ Compute
- ☐ Containers

COMPUTE

Free Tier 12 MONTHS FREE

Amazon EC2

750 Hours

per month

Resizable compute capacity in the Cloud.

750 hours per month of Linux, RHEL, or SLES t2.micro or t3.micro instance dependent on region

750 hours per month of Windows t2.micro or t3.micro instance dependent on region

En resumen, tenemos lo siguiente:

▼ **Summary**

Number of instances [Info](#)

Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2022-12-01
ami-03b755af568109dc3

Virtual server type (instance type)

t2.micro

[Firewall \(security group\)](#)

New security group

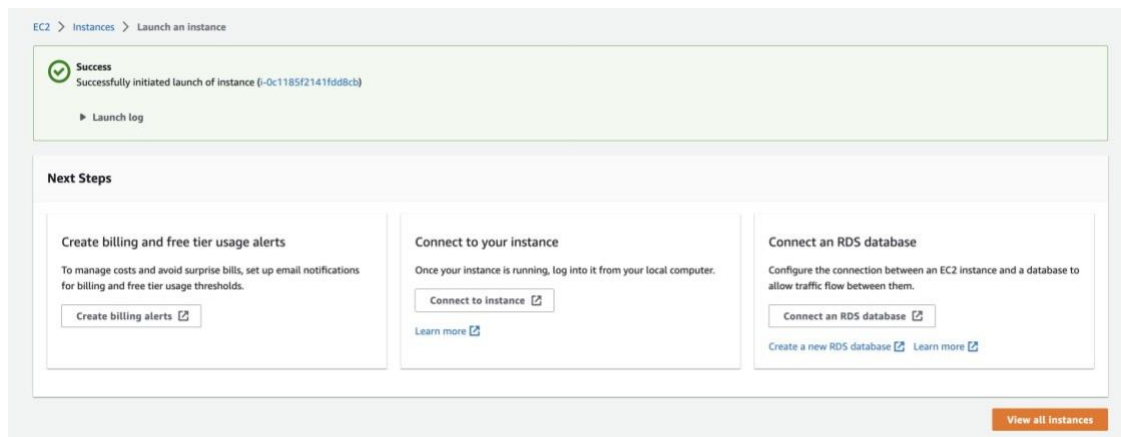
[Storage \(volumes\)](#)

1 volume(s) - 8 GiB

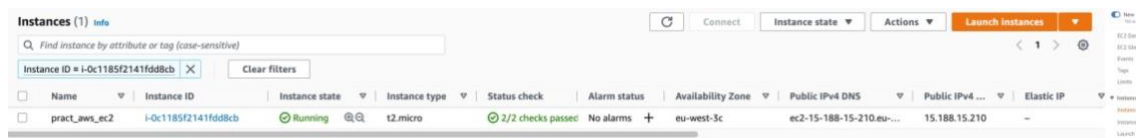
ℹ **Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet. ✕

[Cancel](#) [Launch instance](#)

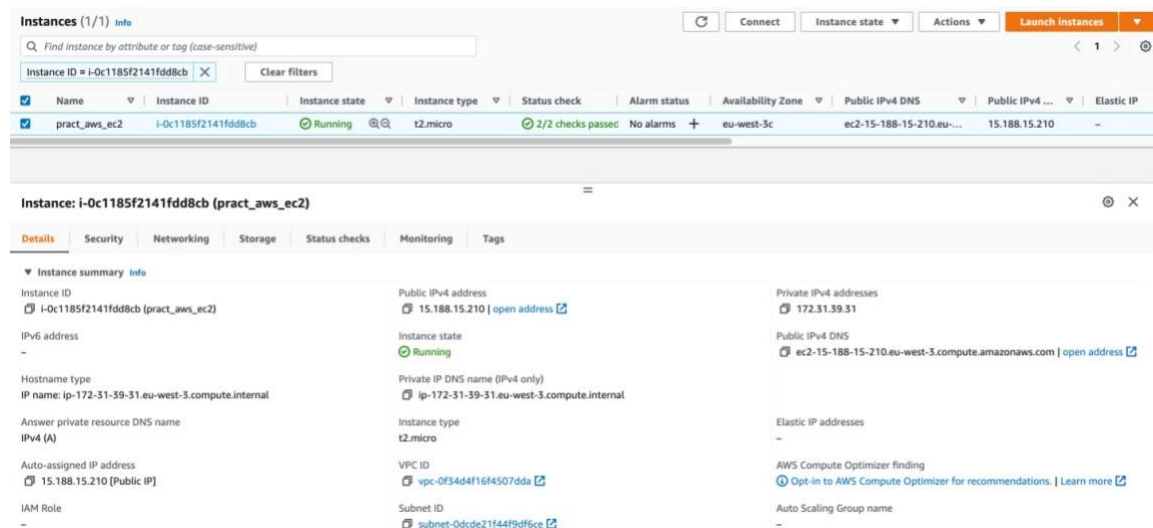
Finalmente, lanzamos la instancia:



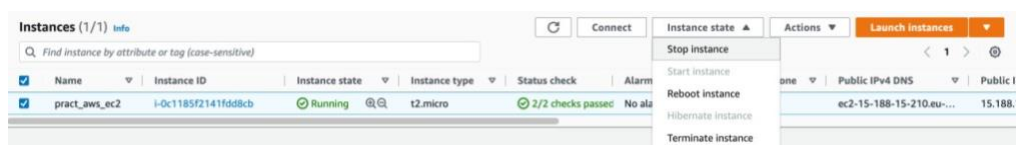
Nuestra instancia ya está corriendo:



La seleccionamos y nos aparece información del servidor:



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

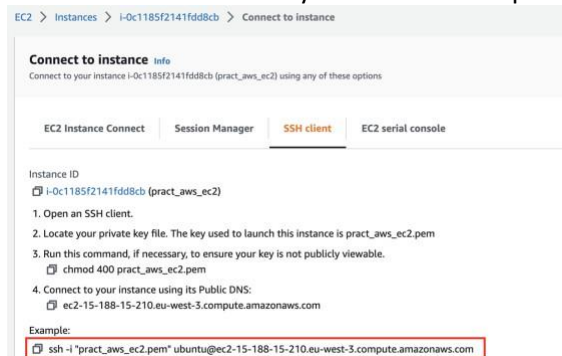


1.3.3 Conectarse a una instancia

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:

```
~/Documents/MASTER_DS/compus/aws  
chmod 400 pract_aws_ec2.pem
```

- 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.



```
~/Documents/MASTER_DS/compus/aws  
ssh -i "pract_aws_ec2.pem" ubuntu@ec2-15-188-15-210.eu-west-3.compute.amazonaws.com  
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1026-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Sun Dec 18 02:24:57 UTC 2022  
  
System load:  0.0          Processes:      97  
Usage of /:   21.5% of 7.57GB  Users logged in: 0  
Memory usage: 23%          IPv4 address for eth0: 172.31.39.31  
Swap usage:   0%  
  
* Ubuntu Pro delivers the most comprehensive open source security and  
compliance features.  
  
https://ubuntu.com/aws/pro  
  
0 updates can be applied immediately.  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-39-31:~$
```

- 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:

```
~ /Documents/MASTER_DS/compus/flask_ETL_sw
git init
Initialized empty Git repository in /Users/bernatsort/Documents/MASTER_DS/compus/flask_ETL_sw/.git/

~ /Documents/MASTER_DS/compus/flask_ETL_sw on git main ?5
git remote add origin https://github.com/bernatsort/StarWars_ETL_flask.git

~ /Documents/MASTER_DS/compus/flask_ETL_sw on git main ?5
git add -A

~ /Documents/MASTER_DS/compus/flask_ETL_sw on git main +29
git commit -a

~ /Documents/MASTER_DS/compus/flask_ETL_sw on git main
git status
On branch main
nothing to commit, working tree clean

~ /Documents/MASTER_DS/compus/flask_ETL_sw on git main
git push -u origin main
Username for 'https://github.com': bernatsort
Password for 'https://bernatsort@github.com':
Enumerating objects: 36, done.
Counting objects: 100% (36/36), done.
Delta compression using up to 8 threads
Compressing objects: 100% (36/36), done.
Writing objects: 100% (36/36), 100.81 KiB | 20.16 MiB/s, done.
Total 36 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), done.
To https://github.com/bernatsort/StarWars_ETL_flask.git
* [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Ahora todo está en nuestro GitHub. Borraremos 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.

```
ubuntu@ip-172-31-39-31: ~  
~/Documents/MASTER_DS/compus/flask_ETL_sw on main  
ssh -i pract_aws_ec2.pem ubuntu@ec2-15-188-15-210.eu-west-3.compute.amazonaws.com  
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1026-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Sun Dec 18 03:39:24 UTC 2022  
  
System load:  0.0          Processes:            104  
Usage of /:   21.5% of 7.57GB Users logged in:       1  
Memory usage: 24%         IPv4 address for eth0: 172.31.39.31  
Swap usage:   0%  
  
* Ubuntu Pro delivers the most comprehensive open source security and  
  compliance features.  
  https://ubuntu.com/aws/pro  
  
0 updates can be applied immediately.  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
Last login: Sun Dec 18 02:51:15 2022 from 83.44.250.202  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
ubuntu@ip-172-31-39-31:~$
```

Ya hemos sido capaces de iniciar sesión en nuestra instancia.

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

```
ubuntu@ip-172-31-39-31:~$ git clone https://github.com/bernatsort/StarWars_ETL_flask.git  
Cloning into 'StarWars_ETL_flask'...  
remote: Enumerating objects: 41, done.  
remote: Counting objects: 100% (41/41), done.  
remote: Compressing objects: 100% (32/32), done.  
remote: Total 41 (delta 10), reused 36 (delta 8), pack-reused 0  
Receiving objects: 100% (41/41), 102.00 KiB | 4.08 MiB/s, done.  
Resolving deltas: 100% (10/10), done.  
ubuntu@ip-172-31-39-31:~$ ls  
StarWars_ETL_flask  
ubuntu@ip-172-31-39-31:~$
```

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.

Creemos un *security group*:

EC2 > Security Groups > Create security group

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: pract_ec2_aws
Name cannot be edited after creation.

Description: make IP accessible through browser

VPC: vpc-0f3d4f16f4507dda

Inbound rules

Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	8006	Anywhere...	make IP accessible through browser

Outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	Custom	make IP accessible through browser

Tags - optional

No tags associated with the resource.

Create security group

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

EC2 > Instances > i-0c1185f2141fdd8cb > Change security groups

Change security groups

Amazon EC2 evaluates all the rules of the selected security groups to control inbound and outbound traffic to and from your instance. You can use this window to add and remove security groups.

Instance details

Instance ID: i-0c1185f2141fdd8cb (pract_aws_ec2)
Network interface ID: eni-0644cd7e798fcaa9f

Associated security groups

Add one or more security groups to the network interface. You can also remove security groups.

Search: sg-01913ceae0f7a83ca

Security groups associated with the network interface (eni-0644cd7e798fcaa9f)

Security group name	Security group ID	Action
launch-wizard-1	sg-0cdd15e9323112c43	Remove
pract_ec2_aws	sg-01913ceae0f7a83ca	Remove

Add security group

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 proyecto:

```
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:

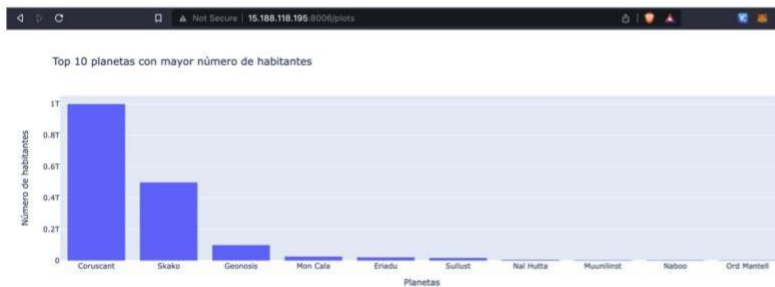
```
^Cubuntu@ip-172-31-39-31:~/StarWars_ETL_flask/src$ ls
__pycache__ app.py db extract.py load.py models.py plots templates transform.py
ubuntu@ip-172-31-39-31:~/StarWars_ETL_flask/src$ vim app.py

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

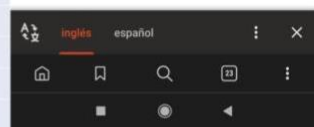
ubuntu@ip-172-31-39-31:~/StarWars_ETL_flask/src$ python3 app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8006
* Running 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



[Volver al índice](#)



[Volver al índice](#)

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>