



UNIVERSIDAD DE LAS FUERZAS ARMADAS “ESPE”

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN



TALLER 2

INGENIERIA EN TECNOLOGÍAS DE LA INFORMACIÓN

APLICACIÓN DE SISTEMAS OPERATIVOS

INTEGRANTES:

CEVALLOS JUAN

PACHACAMA FREDDY LEONEL

QUIMUÑA KEVIN

SANGOQUIZA WILSON

GRUPO #4

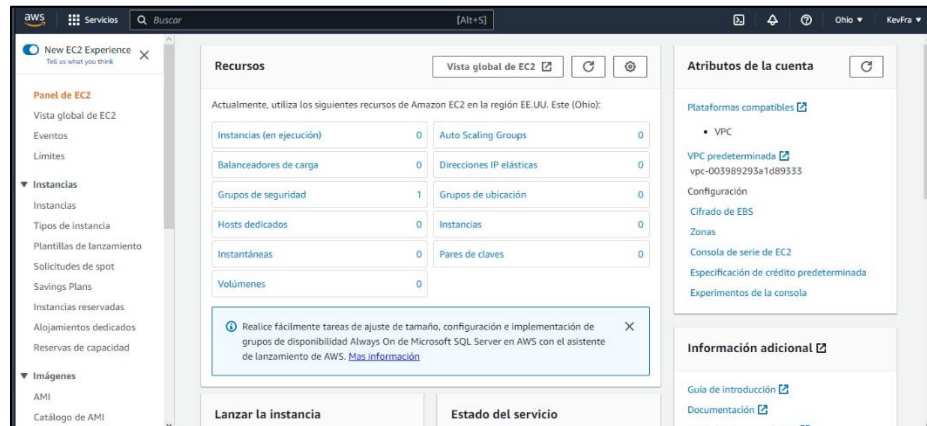
ING. ANDREA LÓPEZ

NRC: 10035

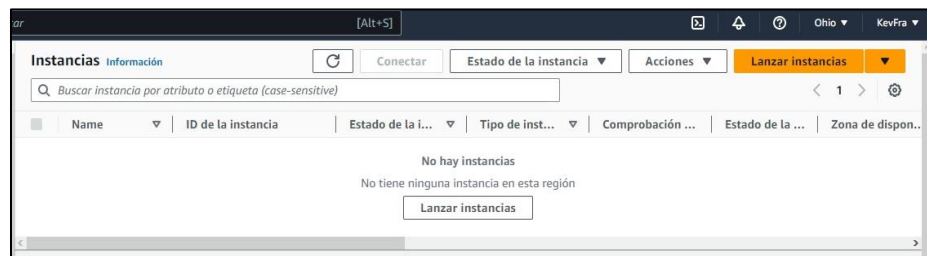
SANGOLQUÍ, 29 DE MAYO DEL 2023

- **Creación de instancia Linux**

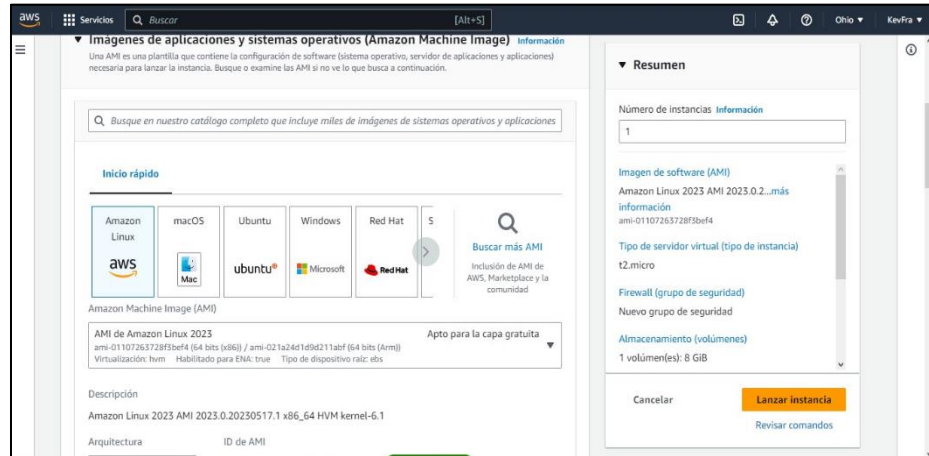
1. Buscar el servicio de EC2.



2. En la consola de EC2, dirigirse a la sección Instancias y crear una nueva instancia en Launch instances:



3. Seleccionar una AMI (Amazon Machine Image). Para el ejemplo se selecciona AMI de Amazon Linux 2023.



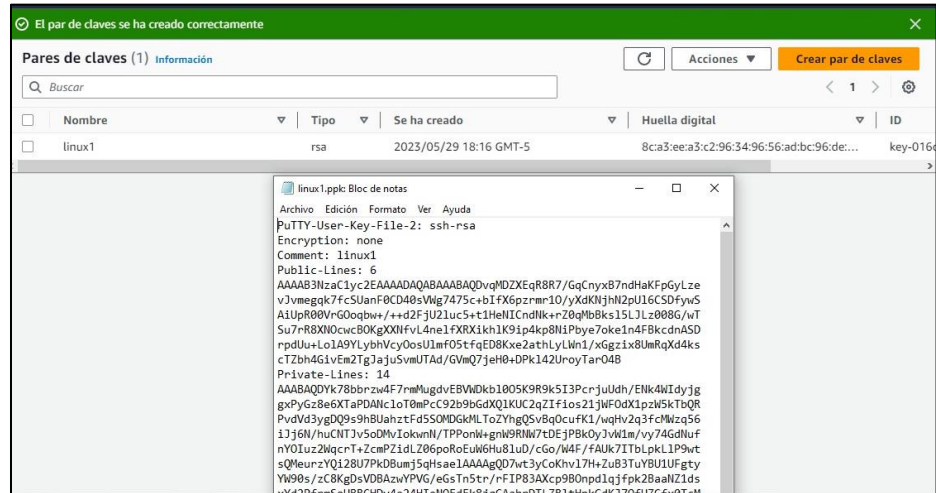
4. Seleccionar las características de hardware para la instancia en creación.



5. Si todo está según lo requerido puede proceder a iniciar su instancia.



6. Es importante crear una clave que permita la conexión de la instancia. Se debe tomar en cuenta que es el único momento para descargar esta key ya que no se podrá tener acceso más adelante. Una vez descargada la llave proceder con el lanzamiento de la instancia Launch.

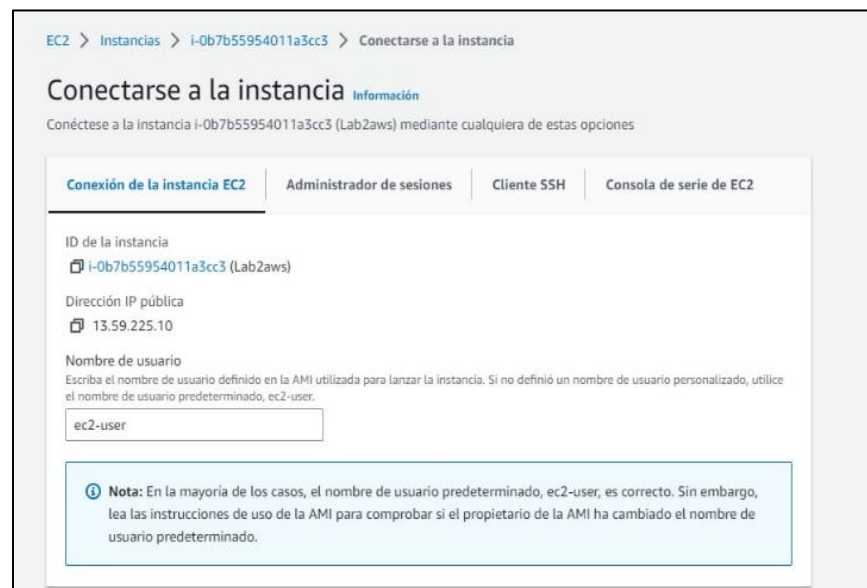


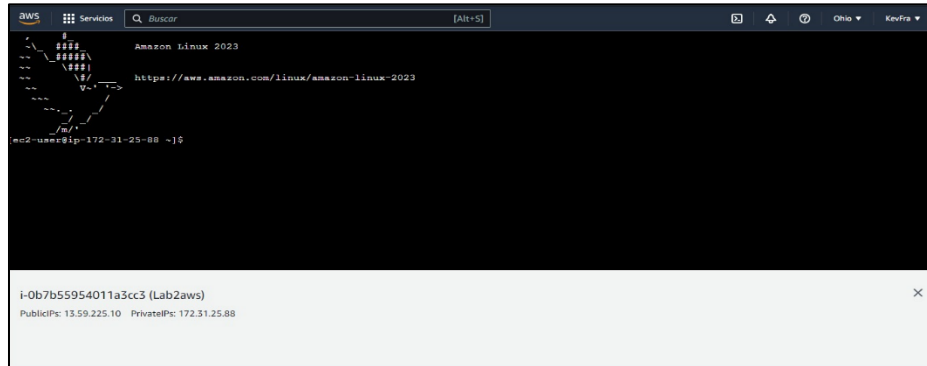
- En la consola de AWS puede verificar la creación de su instancia y los detalles asignados en configuración, almacenamiento, redes seguridad etc.



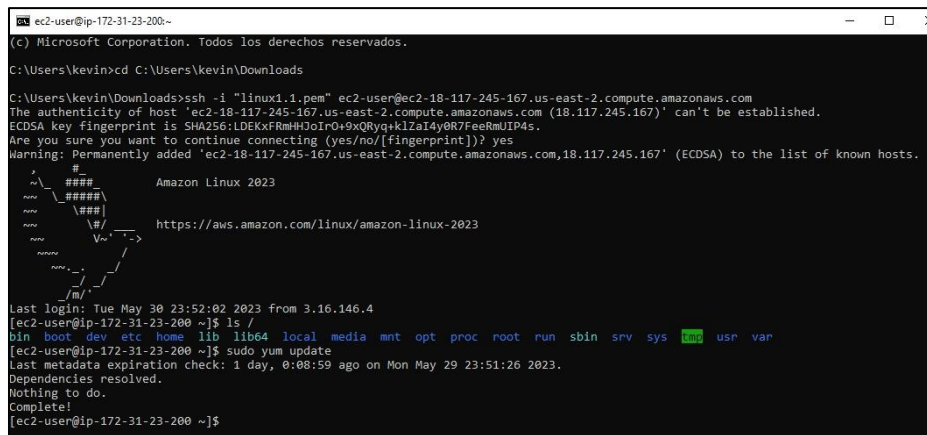
- Conexión de la instancia**

- Para conectar desde la consola seleccionar la opción





9. En la consola del sistema cmd ubicamos la carpeta en la que se guardo la llave de la instancia y pegamos la dirección proporcionada en la pestaña SSH de la instancia.



- **Configuración de un servidor WEB**

10. Mediante el comando de gestor de paquetes de actualizaciones comprobamos si requiere la instancia nuevas actualizaciones o instalación completa.

```

[ec2-user@ip-172-31-23-200 ~]$ sudo yum install git
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-23-200 ~]$ sudo yum install git
Last metadata expiration check: 1 day, 0:11:13 ago on Mon May 29 23:51:26 2023.
Dependencies resolved.
===== Package
chitecture      Version      Repository      Size      -----Installing:
git             x86_64      2.40.1-1.amzn2023.0.1      amazonlinux      57 k
Installing dependencies:
git-core       x86_64      2.40.1-1.amzn2023.0.1      amazonlinux      4.3 M
git-core-doc   noarch     2.40.1-1.amzn2023.0.1      amazonlinux      2.6 M
perl-error     noarch     1:0.17029-5.amzn2023.0.2    amazonlinux      41 k
perl-file-find noarch     1.37-477.amzn2023.0.4      amazonlinux      26 k
perl-git       noarch     2.40.1-1.amzn2023.0.1      amazonlinux      45 k
perl-termreadkey x86_64     2.38-9.amzn2023.0.2        amazonlinux      36 k
perl-lib       x86_64     0.65-477.amzn2023.0.4      amazonlinux      15 k
Transaction Summary
-----Install 8 Packages

Total download size: 7.1 M
Installed size: 34 M
Is this ok [y/N]: y
Downloading Packages:
(1/8): git-2.40.1-1.amzn2023.0.1.x86_64.rpm      780 kB/s | 57 kB  00:00
(2/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm 470 kB/s | 36 kB  00:00
(3/8): perl-error-0.17029-5.amzn2023.0.2.noarch.rpm 1.3 MB/s | 41 kB  00:00
(4/8): perl-lib-0.65-477.amzn2023.0.4.x86_64.rpm 313 kB/s | 15 kB  00:00
(5/8): perl-file-find-1.37-477.amzn2023.0.4.noarch.rpm 859 kB/s | 26 kB  00:00
(6/8): git-core-2.40.1-1.amzn2023.0.1.x86_64.rpm 20 MB/s | 4.3 MB  00:00
(7/8): perl-git-2.40.1-1.amzn2023.0.1.noarch.rpm 682 kB/s | 45 kB  00:00
(8/8): git-core-doc-2.40.1-1.amzn2023.0.1.noarch.rpm 15 MB/s | 2.6 MB  00:00
-----Total
20 MB/s | 7.1 MB  00:00

```

11. Instalamos un repositorio Git hub para exportar el proyecto de Nodejs. Se uso y para la confirmación de la instalación.

```

[ec2-user@ip-172-31-23-200 ~]$ sudo yum install git
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-23-200 ~]$ sudo yum install git
Last metadata expiration check: 1 day, 0:11:13 ago on Mon May 29 23:51:26 2023.
Dependencies resolved.
===== Package
chitecture      Version      Repository      Size      -----Installing:
git             x86_64      2.40.1-1.amzn2023.0.1      amazonlinux      57 k
Installing dependencies:
git-core       x86_64      2.40.1-1.amzn2023.0.1      amazonlinux      4.3 M
git-core-doc   noarch     2.40.1-1.amzn2023.0.1      amazonlinux      2.6 M
perl-error     noarch     1:0.17029-5.amzn2023.0.2    amazonlinux      41 k
perl-file-find noarch     1.37-477.amzn2023.0.4      amazonlinux      26 k
perl-git       noarch     2.40.1-1.amzn2023.0.1      amazonlinux      45 k
perl-termreadkey x86_64     2.38-9.amzn2023.0.2        amazonlinux      36 k
perl-lib       x86_64     0.65-477.amzn2023.0.4      amazonlinux      15 k
Transaction Summary
-----Install 8 Packages

Total download size: 7.1 M
Installed size: 34 M
Is this ok [y/N]: y
Downloading Packages:
(1/8): git-2.40.1-1.amzn2023.0.1.x86_64.rpm      780 kB/s | 57 kB  00:00
(2/8): perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64.rpm 470 kB/s | 36 kB  00:00
(3/8): perl-error-0.17029-5.amzn2023.0.2.noarch.rpm 1.3 MB/s | 41 kB  00:00
(4/8): perl-lib-0.65-477.amzn2023.0.4.x86_64.rpm 313 kB/s | 15 kB  00:00
(5/8): perl-file-find-1.37-477.amzn2023.0.4.noarch.rpm 859 kB/s | 26 kB  00:00
(6/8): git-core-2.40.1-1.amzn2023.0.1.x86_64.rpm 20 MB/s | 4.3 MB  00:00
(7/8): perl-git-2.40.1-1.amzn2023.0.1.noarch.rpm 682 kB/s | 45 kB  00:00
(8/8): git-core-doc-2.40.1-1.amzn2023.0.1.noarch.rpm 15 MB/s | 2.6 MB  00:00
-----Total
20 MB/s | 7.1 MB  00:00

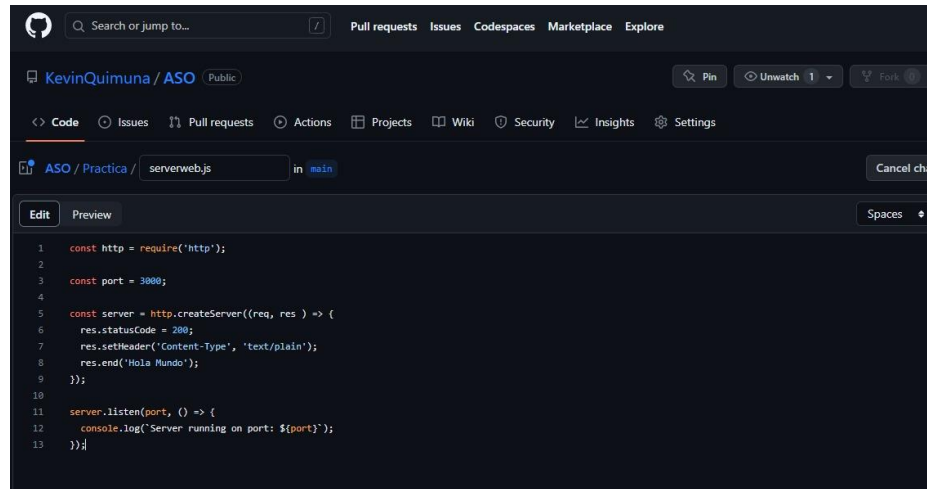
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                : 1/1
  Installing               : git-core-2.40.1-1.amzn2023.0.1.x86_64 1/8
  Installing               : git-core-doc-2.40.1-1.amzn2023.0.1.noarch 2/8
  Installing               : perl-file-find-1.37-477.amzn2023.0.4.noarch 3/8
  Installing               : perl-error-1:0.17029-5.amzn2023.0.2.noarch 4/8
  Installing               : perl-lib-0.65-477.amzn2023.0.4.x86_64 5/8
  Installing               : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 6/8
  Installing               : perl-git-2.40.1-1.amzn2023.0.1.noarch 7/8
  Installing               : git-2.40.1-1.amzn2023.0.1.x86_64 8/8
  Running scriptlet       : git-2.40.1-1.amzn2023.0.1.x86_64 8/8
  Verifying                : git-core-2.40.1-1.amzn2023.0.1.x86_64 1/8
  Verifying                : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 2/8
  Verifying                : git-2.40.1-1.amzn2023.0.1.x86_64 3/8
  Verifying                : perl-lib-0.65-477.amzn2023.0.4.x86_64 4/8
  Verifying                : perl-error-1:0.17029-5.amzn2023.0.2.noarch 5/8
  Verifying                : git-core-doc-2.40.1-1.amzn2023.0.1.noarch 6/8
  Verifying                : perl-file-find-1.37-477.amzn2023.0.4.noarch 7/8
  Verifying                : perl-git-2.40.1-1.amzn2023.0.1.noarch 8/8

Installed:
git-2.40.1-1.amzn2023.0.1.x86_64      git-core-2.40.1-1.amzn2023.0.1.x86_64      git-core-doc-2.40.1-1.amzn2023.0.1.noarch
perl-error-1:0.17029-5.amzn2023.0.2.noarch perl-file-find-1.37-477.amzn2023.0.4.noarch perl-git-2.40.1-1.amzn2023.0.1.noarch
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64 perl-lib-0.65-477.amzn2023.0.4.x86_64

Complete!
[ec2-user@ip-172-31-23-200 ~]$

```

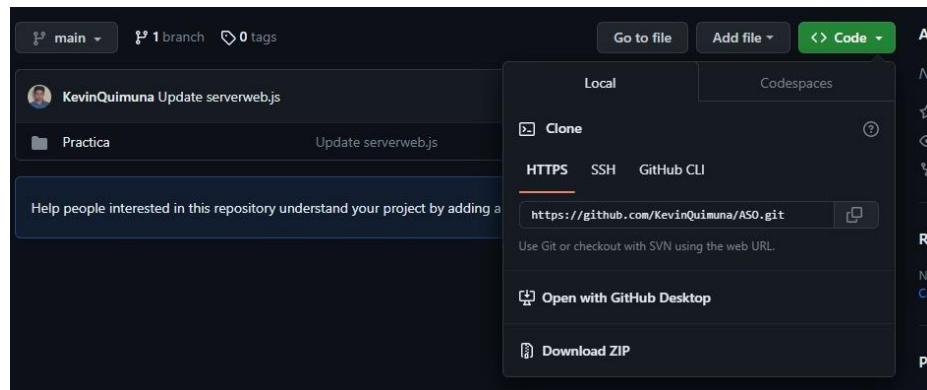
12. Creamos un repositorio en github con el código del servidor para conectar en el puerto 300.



The screenshot shows the GitHub interface for a repository named 'ASO' by user 'KevinQuimuna'. The file 'serverweb.js' is selected, and the code is displayed in a dark-themed editor. The code is a simple HTTP server using Node.js's built-in 'http' module, listening on port 3000 and responding with 'Hola Mundo'.

```
1  const http = require('http');
2
3  const port = 3000;
4
5  const server = http.createServer((req, res) => {
6    res.statusCode = 200;
7    res.setHeader('Content-Type', 'text/plain');
8    res.end('Hola Mundo');
9  });
10
11 server.listen(port, () => {
12   console.log('Server running on port: ${port}');
13 });
```

13. Una vez creado el documento del código del servidor, se deberá clonarlo. En la opción “Code” luego copiamos la dirección HTTPS.



14. En la consola de la instancia pegamos la dirección HTTPS usando el comando “git clone”

```
ec2-user@ip-172-31-23-200:~  
Verifying      : perl-Error-1:0.17029-5.amzn2023.0.2.noarch  
Verifying      : git-core-doc-2.40.1-1.amzn2023.0.1.noarch  
Verifying      : perl-File-Find-1.37-477.amzn2023.0.4.noarch  
Verifying      : perl-Git-2.40.1-1.amzn2023.0.1.noarch  
  
Installed:  
git-2.40.1-1.amzn2023.0.1.x86_64      git-core-2.40.1-1.amzn2023.0.1.x86_64  
perl-Error-1:0.17029-5.amzn2023.0.2.noarch  perl-File-Find-1.37-477.amzn2023.0.4.noarch  
perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64  perl-lib-0.65-477.amzn2023.0.4.x86_64  
  
Complete!  
[ec2-user@ip-172-31-23-200 ~]$ git clone https://github.com/KevinQuimuna/ASO.git  
Cloning into 'ASO'..  
remote: Enumerating objects: 8, done.  
remote: Counting objects: 100% (8/8), done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (8/8), done.  
[ec2-user@ip-172-31-23-200 ~]$
```

15. Instalamos NVM. Copiamos el comando desde el link <https://github.com/nvm/nvm#installing-and-updating>

```
README.md  
nvm is a version manager for node.js, designed to be installed per-user, and invoked per-shell. nvm works on any  
POSIX-compliant shell (sh, dash, ksh, zsh, bash), in particular on these platforms: unix, macOS, and windows WSL.  
  
🔗 Installing and Updating  
  
Install & Update Script  
  
To install or update nvm, you should run the install script. To do that, you may either download and run the script  
manually, or use the following cURL or Wget command:  
  
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash  
  
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash  
  
Running either of the above commands downloads a script and runs it. The script clones the nvm repository to
```

16. Corremos el comando en la consola.


```

ec2-user@ip-172-31-23-200:~
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
[ec2-user@ip-172-31-23-200 ~]$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 15916 100 15916 0 0 177k 0 --:--:-- --:--:-- --:--:-- 178k
=> Downloading nvm from git to '/home/ec2-user/.nvm'
=> Cloning into '/home/ec2-user/.nvm'...
remote: Enumerating objects: 360, done.
remote: Counting objects: 100% (360/360), done.
remote: Compressing objects: 100% (306/306), done.
remote: Total 360 (delta 40), reused 170 (delta 28), pack-reused 0
Receiving objects: 100% (360/360), 219.95 KiB | 4.49 MiB/s, done.
Resolving deltas: 100% (40/40), done.
* (HEAD detached at FETCH_HEAD)
master
=> Compressing and cleaning up git repository

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[ec2-user@ip-172-31-23-200 ~]$

```

17. Una vez instalado verificamos la versión de nvm.

```

ec2-user@ip-172-31-23-200:~
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 15916 100 15916 0 0 177k 0 --:--:-- --:--:-- --:--:-- 178k
=> Downloading nvm from git to '/home/ec2-user/.nvm'
=> Cloning into '/home/ec2-user/.nvm'...
remote: Enumerating objects: 360, done.
remote: Counting objects: 100% (360/360), done.
remote: Compressing objects: 100% (306/306), done.
remote: Total 360 (delta 40), reused 170 (delta 28), pack-reused 0
Receiving objects: 100% (360/360), 219.95 KiB | 4.49 MiB/s, done.
Resolving deltas: 100% (40/40), done.
* (HEAD detached at FETCH_HEAD)
master
=> Compressing and cleaning up git repository

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[ec2-user@ip-172-31-23-200 ~]$ nvm -v
-bash: nvm: command not found
[ec2-user@ip-172-31-23-200 ~]$ source ~/.bashrc
[ec2-user@ip-172-31-23-200 ~]$ nvm -v
0.39.3
[ec2-user@ip-172-31-23-200 ~]$

```

18. Instalamos la ultima versión estable de nodejs.

```

ec2-user@ip-172-31-23-200:~
Receiving objects: 100% (360/360), 219.95 KiB | 4.49 MiB/s, done.
Resolving deltas: 100% (40/40), done.
* (HEAD detached at FETCH_HEAD)
master
=> Compressing and cleaning up git repository

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[ec2-user@ip-172-31-23-200 ~]$ nvm -v
-bash: nvm: command not found
[ec2-user@ip-172-31-23-200 ~]$ source ~/.bashrc
[ec2-user@ip-172-31-23-200 ~]$ nvm -v
0.39.3
[ec2-user@ip-172-31-23-200 ~]$ nvm install node
Downloading and installing node v20.2.0...
Downloading https://nodejs.org/dist/v20.2.0/node-v20.2.0-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v20.2.0 (npm v9.6.6)
Creating default alias: default -> node (-> v20.2.0)
[ec2-user@ip-172-31-23-200 ~]$

```

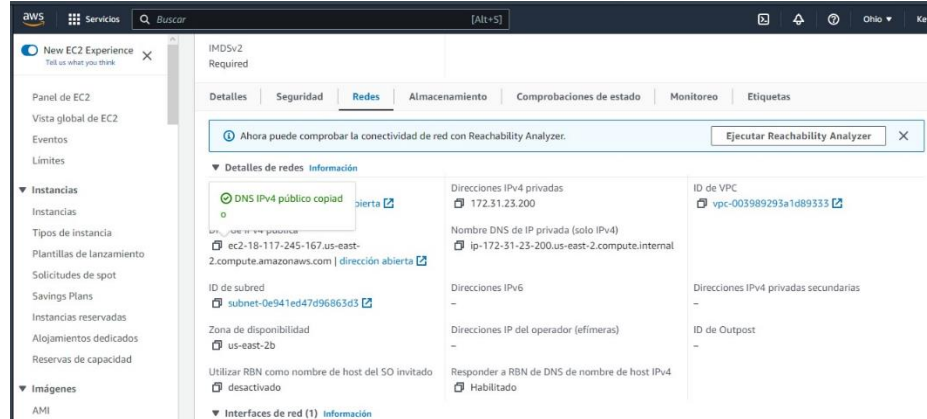
19. Finalmente ejecutamos el servidor.

```

ec2-user@ip-172-31-23-200:~
}
Node.js v20.2.0
[ec2-user@ip-172-31-23-200 ~]$ ls -l
total 0
drwxr-xr-x. 4 ec2-user ec2-user 34 May 31 00:22 ASO
[ec2-user@ip-172-31-23-200 ~]$ ls -l Practica
ls: cannot access 'Practica': No such file or directory
[ec2-user@ip-172-31-23-200 ~]$ ls -l ASO
total 0
drwxr-xr-x. 2 ec2-user ec2-user 26 May 31 00:22 Practica
[ec2-user@ip-172-31-23-200 ~]$ ls -l Practica
ls: cannot access 'Practica': No such file or directory
[ec2-user@ip-172-31-23-200 ~]$ node ASO/Practica/serverweb.js
Server running on port: 3000

```

20. Copiamos la dirección pública de la instancia de la consola de AWS



21. Colocamos la dirección en el navegador y conectamos el puerto en el que se está ejecutando.



Preguntas:

1. ¿Qué es EC2? ¿Cuáles son las características principales de este servicio de AWS?

Amazon Elastic Compute Cloud (EC2) es un servicio web de Amazon Web Services (AWS) que proporciona capacidad de computación segura y redimensionable en la nube. Está diseñado para facilitar a los desarrolladores el escalado web y el control de los recursos informáticos.

Algunas de las características principales de Amazon EC2 incluyen:

- **Infraestructura global:** Amazon EC2 ofrece la posibilidad de colocar instancias en distintas ubicaciones, compuestas por regiones y zonas de disponibilidad.
- **Optimización de costos y capacidad:** Pago en función del uso.
- **Almacenamiento:** Almacenamiento óptimo para cada carga de trabajo.
- **Redes:** Alto rendimiento de paquetes por segundo y baja latencia con una red mejorada.
- **Sistemas operativos y software:** Elección de sistemas operativos y software.

2. ¿Por qué los tipos de instancia que se pueden añadir de hardware se conocen como t2.nano, t2.micro, t2.medium, t2.small? Describa las características de estos tipos de instancias.

Los tipos de instancia t2.nano, t2.micro, t2.small y t2.medium son parte de la familia de instancias T2 de Amazon EC2. Estas instancias están diseñadas para reducir drásticamente

los costos para aplicaciones que no utilizan la capacidad completa de la CPU continuamente, pero que ocasionalmente necesitan ráfagas de rendimiento.

Estos tipos de instancia tienen un procesador Intel Xeon Scalable de hasta 3.3 GHz. Las características específicas de cada tipo de instancia son las siguientes:

t2.nano: 1 vCPU y 0.5 GiB de memoria
t2.micro: 1 vCPU y 1 GiB de memoria
t2.small: 1 vCPU y 2 GiB de memoria
t2.medium: 2 vCPU y 4 GiB de memoria

Estas instancias son ideales para aplicaciones que requieren ráfagas ocasionales de rendimiento, como servidores web, pequeñas bases de datos y entornos de desarrollo.

3. ¿Qué es un servidor Nodejs?

Node.js es un entorno en tiempo de ejecución multiplataforma y de código abierto para la capa del servidor (aunque no se limita a ello) basado en el lenguaje de programación JavaScript. Es asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Node.js fue creado con el objetivo de ser útil en la creación de programas de red altamente escalables, como por ejemplo servidores web. Permite a los desarrolladores crear toda clase de herramientas y aplicaciones del lado del servidor en JavaScript y ejecutarlas fuera del contexto de un navegador web

4. ¿Cuáles son los beneficios de crear una AMI personalizada (customized)?

Una AMI personalizada (customized) es una imagen de máquina de Amazon (AMI) que ha sido configurada de acuerdo a las necesidades específicas de un usuario. Hay varios beneficios al crear una AMI personalizada:

Empaquetado previo de paquetes concretos: Puedes realizar un empaquetado previo de paquetes concretos en lugar de instalarlos una vez que la instancia se ha iniciado.

Control de actualizaciones: Puedes controlar cuándo se producirán las actualizaciones de paquetes para proporcionar a la capa una imagen base coherente.

Arranque rápido: Las instancias basadas en carga en particular pueden arrancar lo más rápido posible.

Crear una AMI personalizada también te permite elegir el sistema operativo que necesites, agregar recursos como volúmenes de Elastic Block Store (EBS), cambiar la configuración, cargar datos o instalar el software que se aplicará a todas las implementaciones que crees.

5. ¿Qué clase de modelo de servicio fue implementado en el taller mediante el uso del servicio EC2 IaaS, SaaS, PaaS? Justifique su respuesta.

EC2, que significa Elastic Compute Cloud, es un servicio de computación en la nube ofrecido por Amazon Web Services. EC2 es parte de la gama de servicios de infraestructura como servicio (IaaS) de AWS y proporciona capacidad informática escalable en la nube.

EC2 permite a los usuarios alquilar máquinas virtuales (instancias) en la nube para ejecutar aplicaciones y realizar tareas de cómputo. Estas instancias se pueden configurar y personalizar según las necesidades del usuario, incluyendo la elección del sistema operativo, los recursos computacionales, la capacidad de almacenamiento y la ubicación geográfica.