TRABAJO DE GRADO
Opción Seminario-Diplomado.

Implementación de servicios AWS

Corporación Universitaria Remington.
Facultad de Ingeniería de Sistemas
SEMINARIO AMAZON WEB SERVICES (AWS)

Jhonatan Sneider Santos Ortiz - Luis Angel Pérez Polo - Néstor Navarro Ibarra
Juan Pablo Beriros López
Opción de Trabajo de grado Seminario-Diplomado.
Año de presentación del trabajo de grado.

2025

Tabla de Contenidos

Resumen

En el siguiente trabajo de proyecto se demostrará de manera detallada la implementación de servicios en la nube del proveedor *AWS (Amazon Web Services)*, siendo este uno de los más relevantes a nivel global.
Se explicará de manera detallada la metodología para acceder y utilizar servicios como la creación de máquinas virtuales en función de buscar optimización de recursos y sacar el máximo potencial de AWS en relación con el costo de sus servicios, todo esto desde nuestra propia VPC (Virtual Private Cloud).

La creación de contenedores Docker para el despliegue de aplicaciones se encuentra incluida como actividad de este proyecto, además de la configuración de balanceadores de carga que permitan optimizar y asegurar el correcto flujo de peticiones a las aplicaciones desplegadas en los servicios de AWS (EC2 y ECS), añadiendo el servicio de Auto escalado *AWS Auto Scaling* aplicado tanto a instancias EC2 como a contenedores Docker dentro de un Cluster, sirviéndonos como práctica visual de cómo al saturarse una instancia o contenedor en función de políticas de rendimiento que nosotros mismos asignamos, el servicio de Auto Scaling genera una instancia o una tarea (contenedor) nueva con el fin de mantener activa para los usuarios la aplicación desplegada.
Entre los entornos de seguridad se dará muestra de cómo se asignaron los Firewalls de los balanceadores de carga para permitir la conexión a las instancias creadas y el diseño de rangos de puerto permitidos en el balanceador de carga de los contenedores para permitir conexión con las tareas creadas por el Auto Scaling con puertos dinámicos.

Palabras clave

- Despliegue de aplicaciones en la nube.
- Máquinas virtuales.
- Contenedores Docker.
- Balanceadores de carga.
- Auto Scaling.
- Servicios de AWS.

Marco conceptual y contextual

Implementación de servicios en la nube

Para implementar los servicios que se utilizaron en este proyecto seminario primero se debe comprender cuáles fueron y para qué sirven:

- VPC: Virtual Private Cloud, trata de una red virtual aislada que nos permite crear un entorno unificado de todos los servicios de Amazon que deseemos activar,

por lo que podemos tener múltiples aplicaciones relacionadas a una sola cuenta AWS, pero separadas por diferentes VPC.

- Amazon Elastic Computed Cloud (EC2): Amazon Elastic Computed Cloud es un servicio de infraestructura de hardware escalable que permite a los usuarios crear instancias (servidores virtuales), a gusto y en función de poder reasignar el hardware deseado para el funcionamiento del servidor.

- Amazon EC2 Auto Scaling: Este servicio le permite al empleador tener la certeza de que su aplicación alojada en los servicios de AWS cuenta con la infraestructura de instancias necesaria y óptima para su correcto funcionamiento, el número de instancias es proporcionado manualmente y por medio de políticas de rendimiento que permiten tener una vista gráfica del funcionamiento y gasto de recursos de las instancias.

- Amazon Elastic Container Service (ECS): Creado con el fin de implementar el servicio de terceros *Docker* y su ágil implementación de contenedores para el despliegue de aplicaciones, éstas ofrecen un gran aprovechamiento de recursos de una Instancia debido a la posible inclusión de *balanceadores de carga* que permiten ofrecer a los clientes un mismo punto de conexión mediante un DNS designado, además del control de tráfico de peticiones a múltiples contenedores designados por políticas manuales, el servicio Auto Scalling de tareas y la asignación de múltiples servicios que podemos controlar a conveniencia.

- Load Balancer (balanceador de carga): Herramienta de AWS que permite controlar el flujo de las solicitudes de los clientes a una aplicación, el balanceador de carga hace de único punto de llegada, a este se le puede determinar una única dirección y se encargará de redirigir las peticiones a distintas instancias EC2 o a distintos contenedores Docker dependiendo de la disponibilidad al momento de recibir las peticiones.

- Amazon Machine Image (AMI): Imagen de máquina de Amazon es una imagen determinada de un sistema previamente seleccionado con características de software como sistema operativo definido y requerimientos de hardware fijos, las AMI proporcionan una imagen de máquina fijo que genera facilidad en la replicación de instancias o contenedores, ya que permite al servicio de Auto Scaling que tipo de instancia o contenedor crear y con qué características.

- Grupos de seguridad: Los grupos de seguridad en AWS funcionan cómo control de seguridad por medio de la asignación de reglas de permisión o denegación de puertos, son aplicables tanto a instancias cómo a balanceadores de carga y servicios de Auto Scaling.

- Target Groups: Los Target Groups son grupos de instancias regidas por protocolos o puertos que tienen como intención ser el objetivo de un balanceador de carga.

Desarrollo e implementación del aprendizaje

Software de terceros requerido para el desarrollo del seminario:

- MobaXterm, Mobatek. (s. f.). *MobaXterm*. https://mobaxterm.mobatek.net/
- Putty, *Download PuTTY - a free SSH and telnet client for Windows*. (s. f.). https://www.putty.org/
- HeidiSQL, Becker, A. (s. f.). *HeidiSQL - MariaDB/MySQL, MSSQL, PostgreSQL, SQLite and Interbase/Firebird made easy*. https://www.heidisql.com/

Entrega 1

En el proyecto de grado Seminario Amazon Web Services (AWS) se hizo un principal enfoque en conocer los servicios que AWS nos ofrece para alojar nuestras aplicaciones, con el fin de que como ingenieros desarrollemos la habilidad de determinar qué servicio es el óptimo a utilizar en función de nuestros requerimientos, o los requerimientos de la empresa o cliente para quien trabajemos.
Además de comprender los beneficios del entorno del Cloud Computing y explorar posibles nuevos horizontes profesionales.

A continuación, procedemos a explicar el entorno AWS y los procedimientos de trabajo realizados en el presente proyecto.

Dashboard de AWS

En la siguiente imagen se muestra el Dashboard de AWS luego de ingresar a nuestra cuenta, en este Dashboard podemos ver con mayor detalle los servicios que tengamos activos al momento, además de ser la pantalla desde dónde podemos dirigirnos con solo un clic a cualquiera de los servicios que AWS ofrece (Sección izquierda de la pantalla).

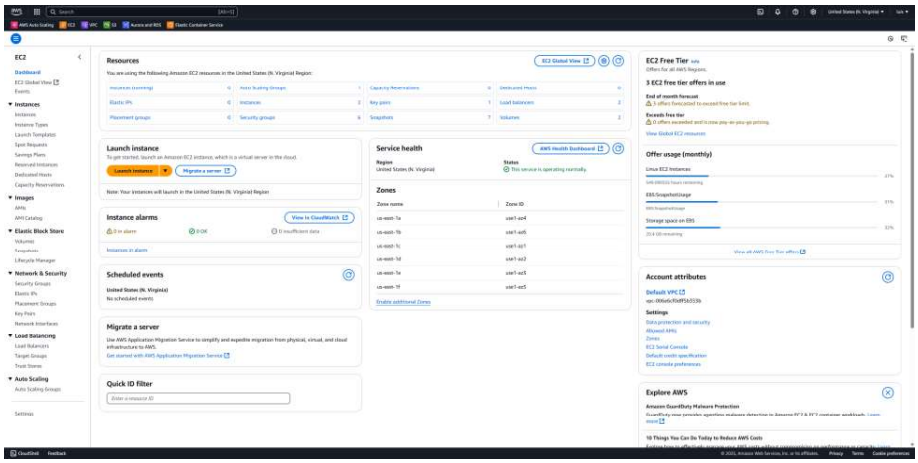


Imagen 1, Dashboard de AWS

Creación de una Instancia EC2

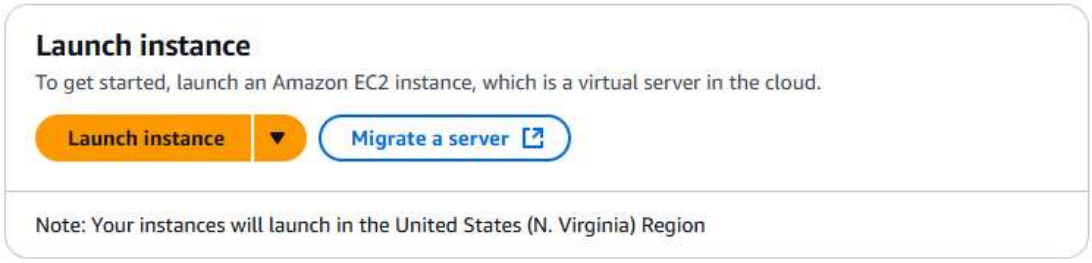


Imagen 2, Botón para crear instancias EC2

Desde el Dashboard podemos encontrar el botón con el texto Launch Instance, una vez presionado seremos redirigidos a la siguiente pantalla.

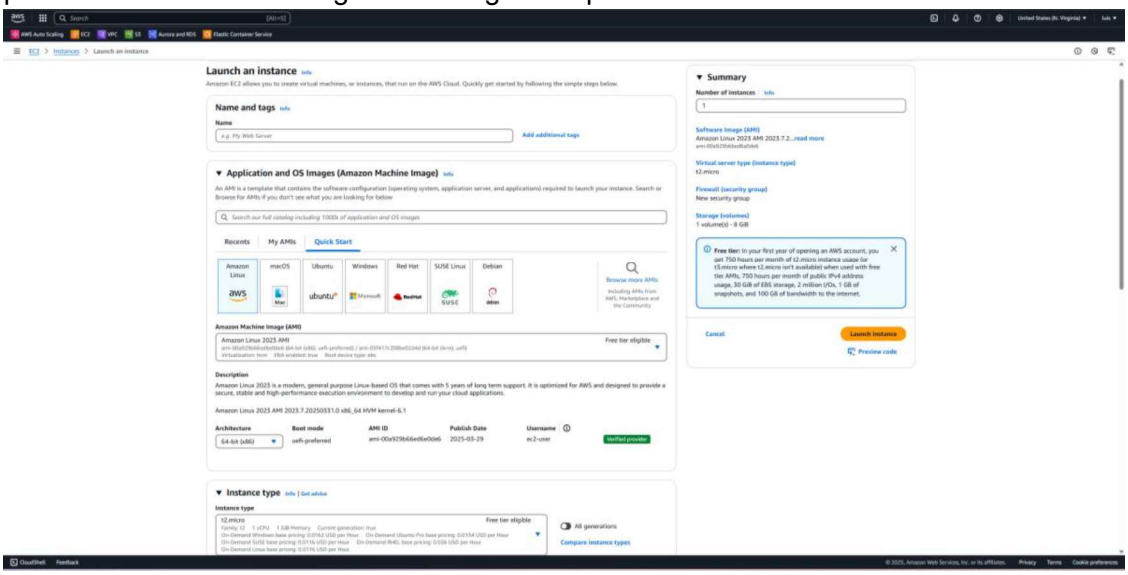


Imagen 3, pantalla de creación de Instancias EC2

En la pantalla representada por la imagen 3 nos encontraremos con el formulario necesario para definir las características que deseamos asignarle a la instancia, tales como sistema operativo, hardware deseado, asignarle grupo de seguridad y hasta las opciones de red.

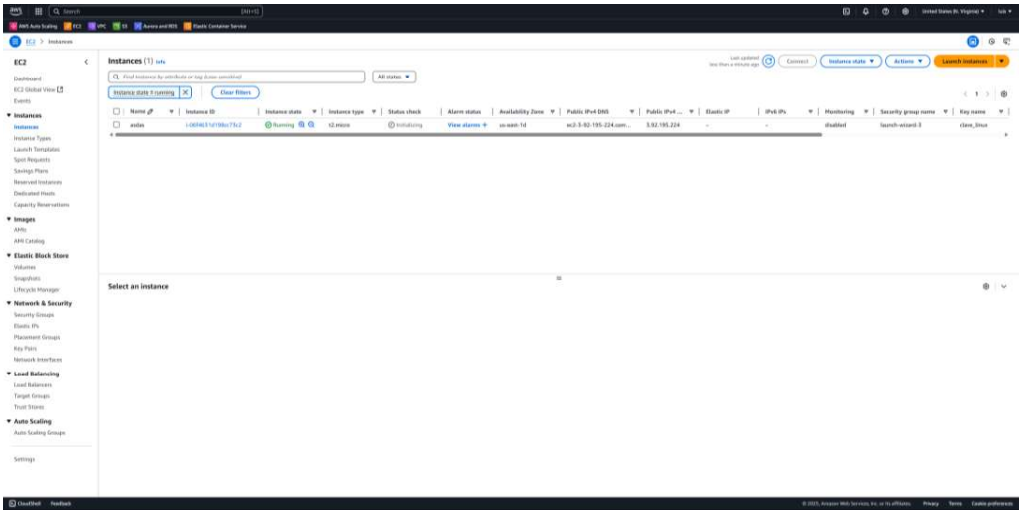


Imagen 4, Instancia creada y funcionando

Una vez la instancia es creada recurrimos al botón Connect para poder conectarnos a la instancia desde nuestra máquina física.

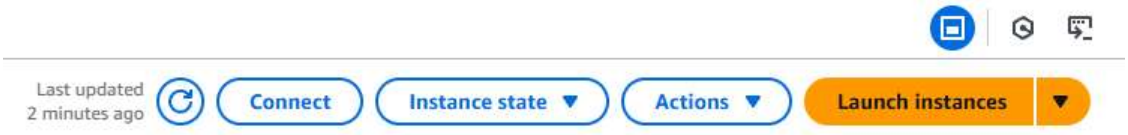


Imagen 5, botón Conectar

Una vez presionado el botón Connect seremos redirigidos a la siguiente pantalla (Imagen 6) dónde nos encontraremos con la información necesaria para conectarnos a la instancia.

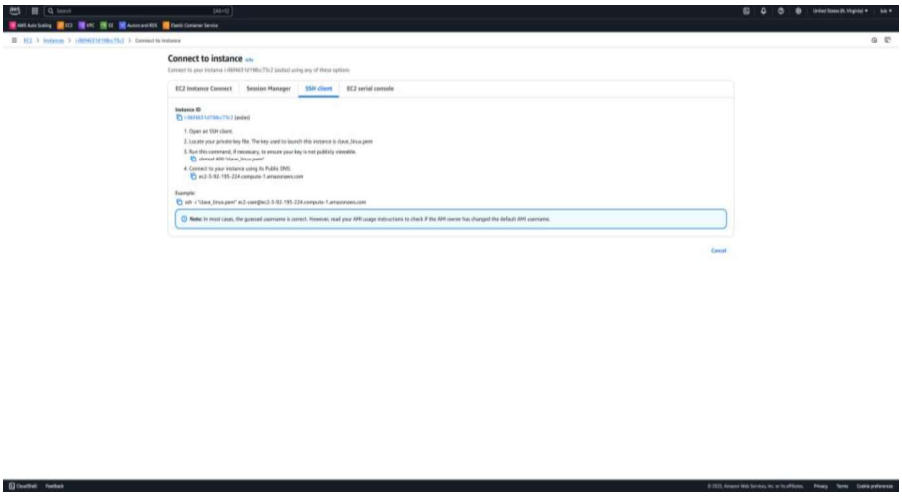


Imagen 6, Pantalla de conexión a la instancia EC2

Para este proyecto le asignamos a las Instancias creadas el sistema operativo Amazon Linux, por tanto, el método de conexión es de tipo SSH, es por ello qué en pantalla (Imagen 6) nos encontramos ubicados en la sección SSH Client.

Para completar la conexión haremos uso del software MobaXterm mencionado en el *software de terceros requerido para el desarrollo del seminario.*

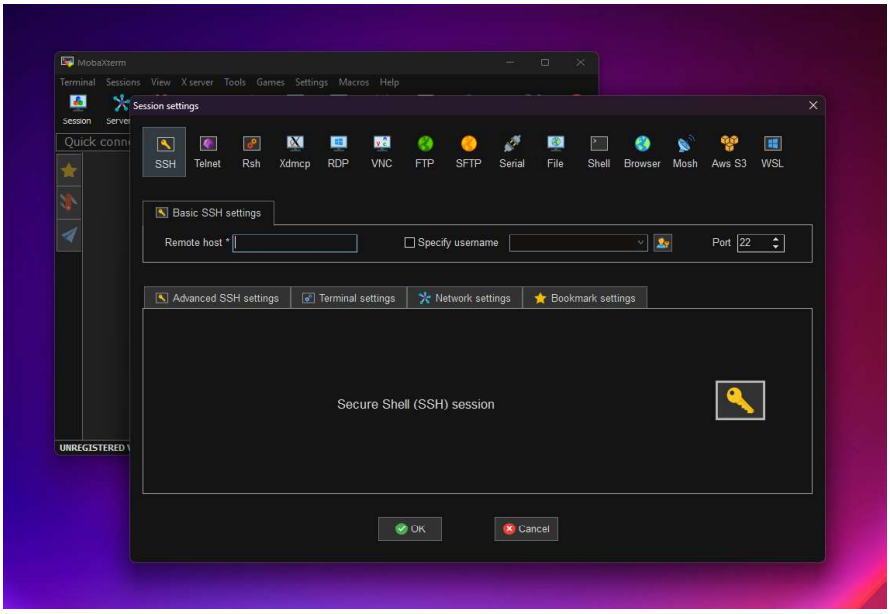


Imagen 7, configuración de MobaXterm

Una vez abierto el software MobaXterm procedemos a rellenar la información requerida:

- Remote host:
  Extraemos la cadena de conexión que provee AWS
  “ssh -i "clave_linux.pem" ec2-user@ec2-3-92-195-224.compute-1.amazonaws.com”, de la cual extraemos los caracteres después del @; ese sería el dato a rellenar en Remote Host.

- Marcamos la casilla Specify username e ingresamos: ec2-user.

- Luego presionamos en Advance SSH settings y en use private Key importamos nuestro archivo .ppk

Los datos deberían quedar de la forma ilustrada en la siguiente imagen (Imagen 8).

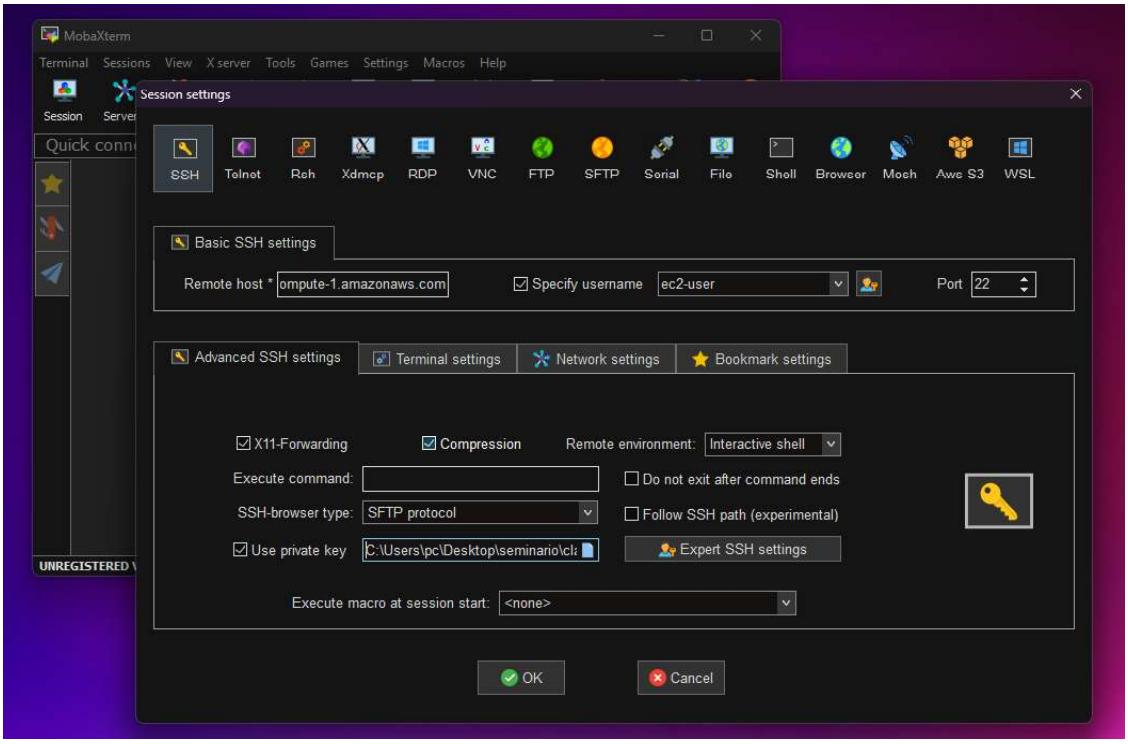


Imagen 8, configuración completa de MobaXterm

Una vez presionamos OK deberíamos ver la siguiente pantalla (Imagen 9) en MobaXterm para confirmar que la configuración del software y la conexión con la Instancia EC2 es correcta:

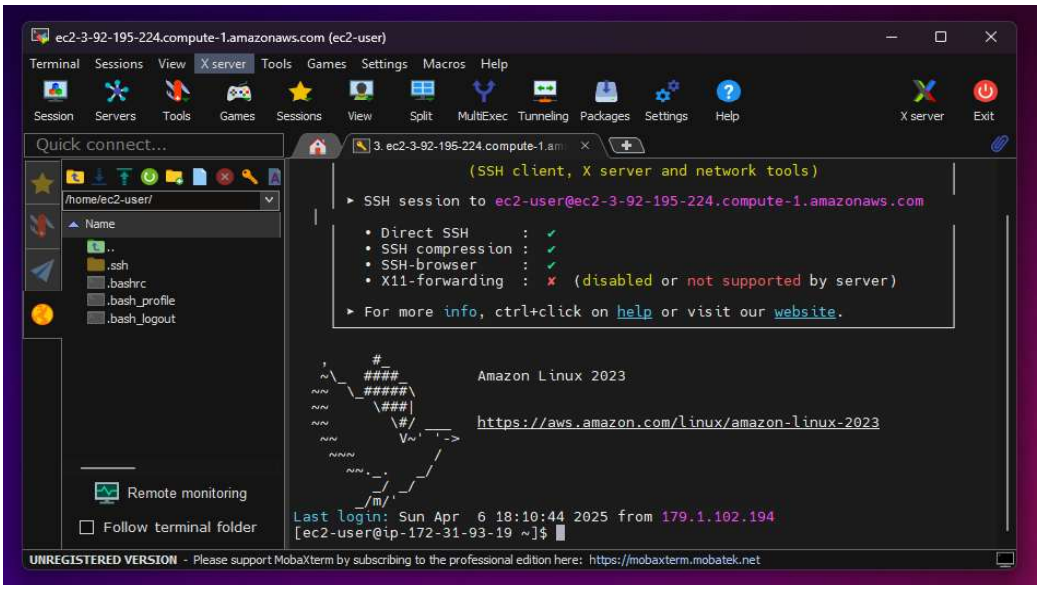


Imagen 9, Conexión exitosa entre MobaXterm y la Instancia EC2

Una vez conectados a la instancia EC2 procedemos a instalar Apache server para tener una especie de aplicativo funcional en nuestra instancia.

1. dnf install httpd => Instalación del server de Apache
2. Systemctl status httpd => verifica el estado del servidor
3. Systemctl start httpd => Arranca el servidor en la máquina virtual [port: 80]

Una vez realizamos los 3 pasos mencionados deberíamos poder ver una pantalla como la siguiente (Imagen 10)

Imagen 10, conexión con el servidor de apache exitosa

Una vez vemos la pantalla (Imagen 10) podemos concluir que el servidor de apache montado en nuestra máquina virtual (Instancia EC2) está funcionando correctamente.

Creación del Load Balancer

En la sección de Load Balancer podemos crear, editar y/o eliminar balanceadores de carga, además de observar las características de los ya existentes.

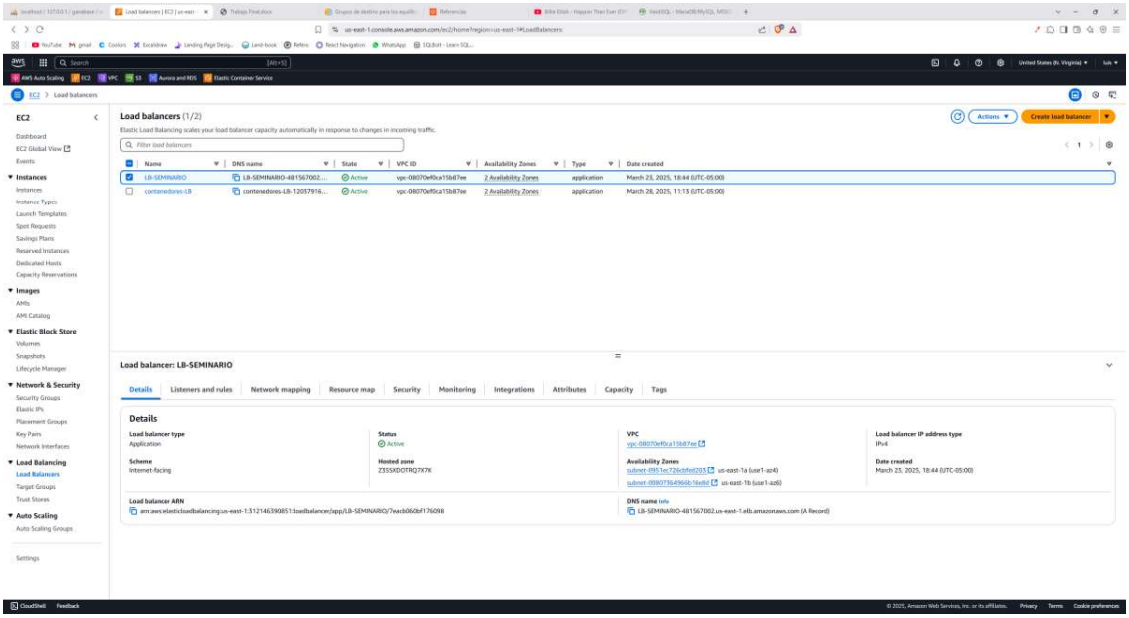


Imagen 11, Dashboard de Load Balancers

Imagen 12, Botón de creación del Load Balancer

Una vez presionado el botón ilustrado en la Imagen 3 seremos redirigidos a la siguiente pantalla.

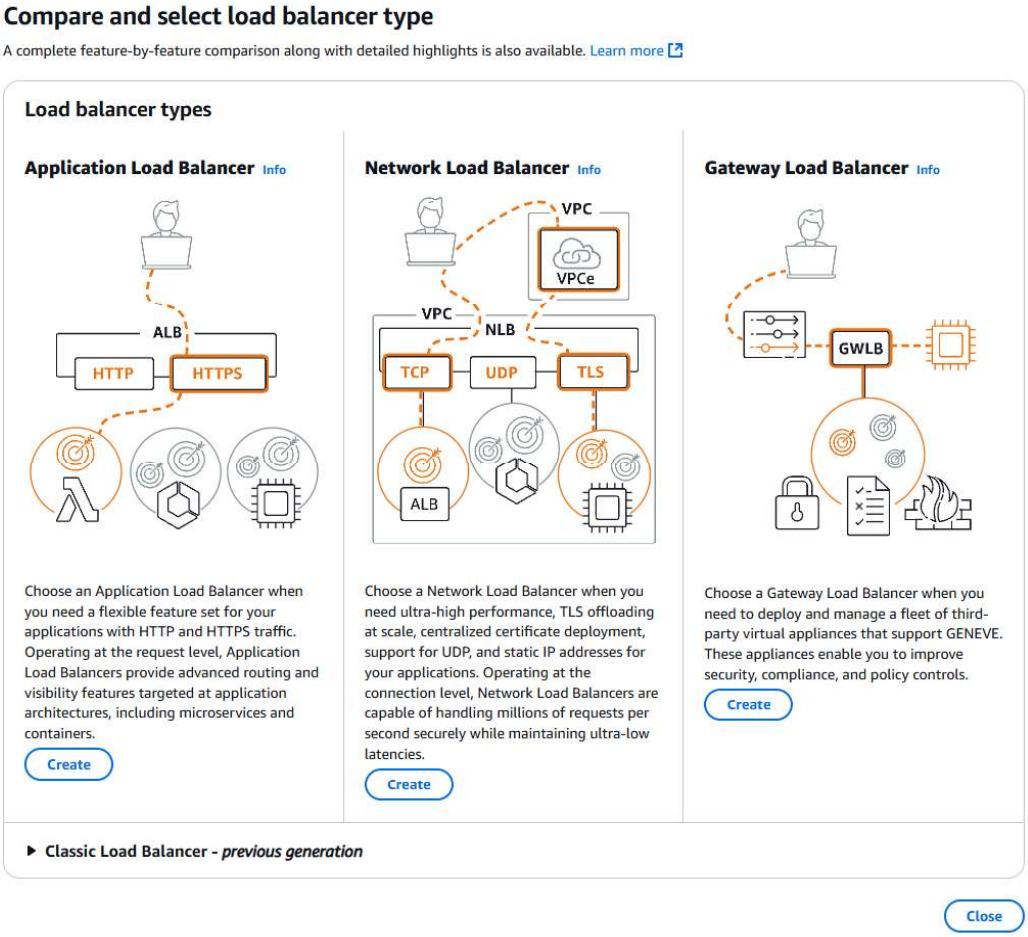


Imagen 13, Selección del tipo de balanceador de carga

El balanceador de carga utilizado en este proyecto fue el Application Load Balancer, por ello seleccionamos el botón Create indicado.

Lo anterior nos redireccionará a la siguiente pantalla.

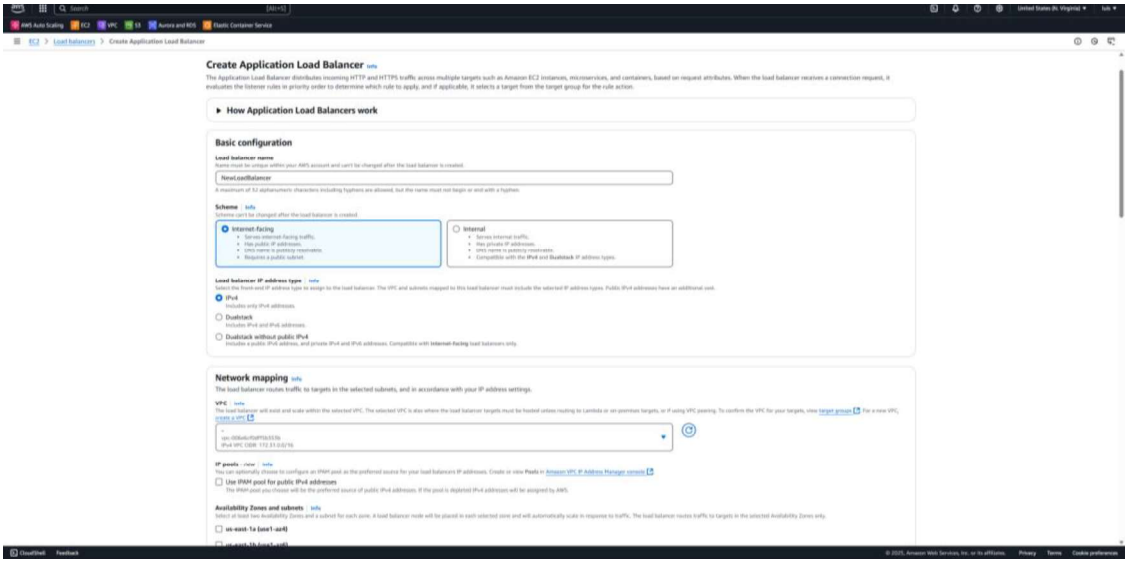


Imagen 14, configuración del Load Balancer

Una vez creado el Load Balancer debería aparecer en la sección Load Balancers.

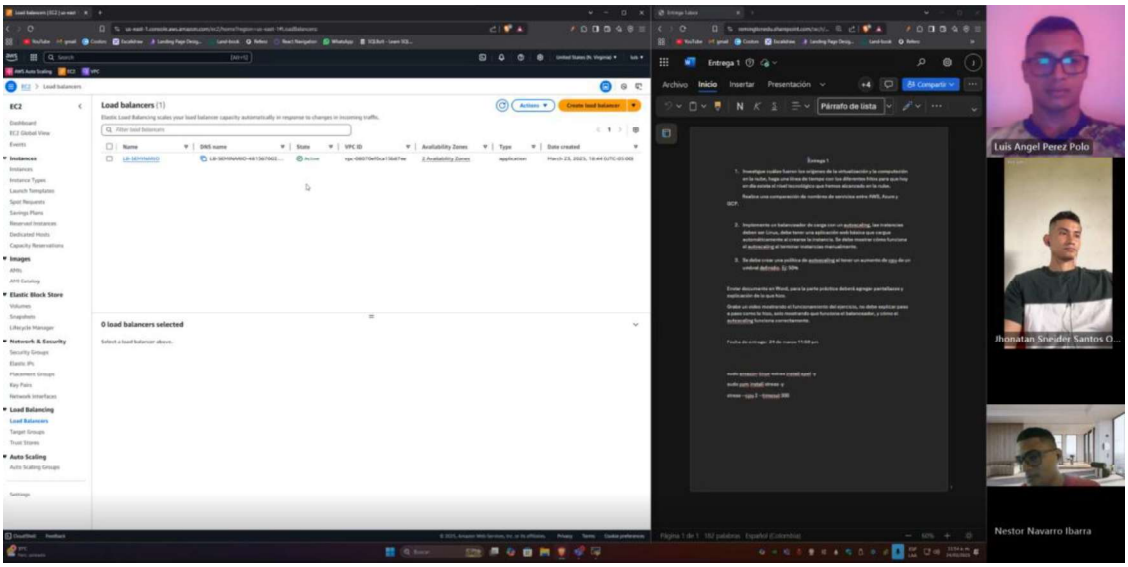


Imagen 15, Load Balancer creado correctamente

Cómo punto final de la primera entrega se realizó la ya ilustrada implementación del balanceador de carga, entonces ahora procedemos a explicar el funcionamiento de este con un Auto Scaling.

Para esto nos dirigimos al buscador y seleccionamos el servicio AWS Auto Scaling, nos aparecerá la siguiente pantalla (Imagen 16).

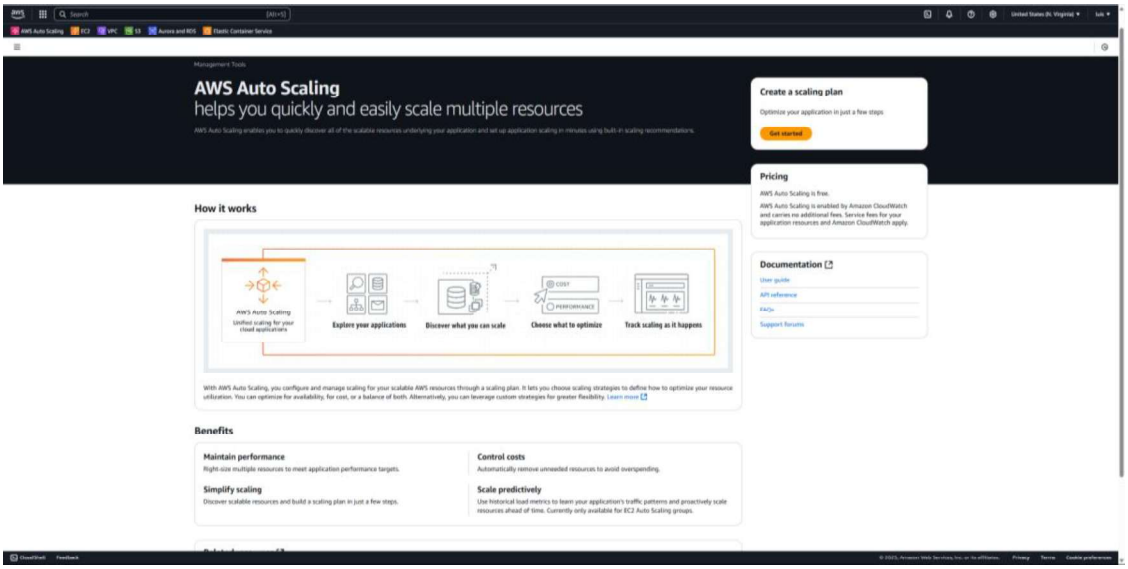


Imagen 16, Auto Scaling Dashboard

Previamente a crear el plan de Auto Escalado debemos tener previamente listos algunos complementos, tales como:

- Un Instance Template (archivo JSON que contiene la infraestructura de la Instancia)
- Un Grupo de Auto Scaling (Normas de crecimiento y decrecimiento del número de instancias)

Para crear un Instance Template nos dirigimos a la sección Launch Template mostrado en la ilustración (Imagen 17).

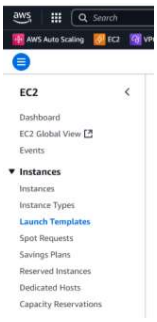


Imagen 17, Launch Template

Una vez ubicados en la sección Launch Template se nos mostrará la siguiente pantalla:

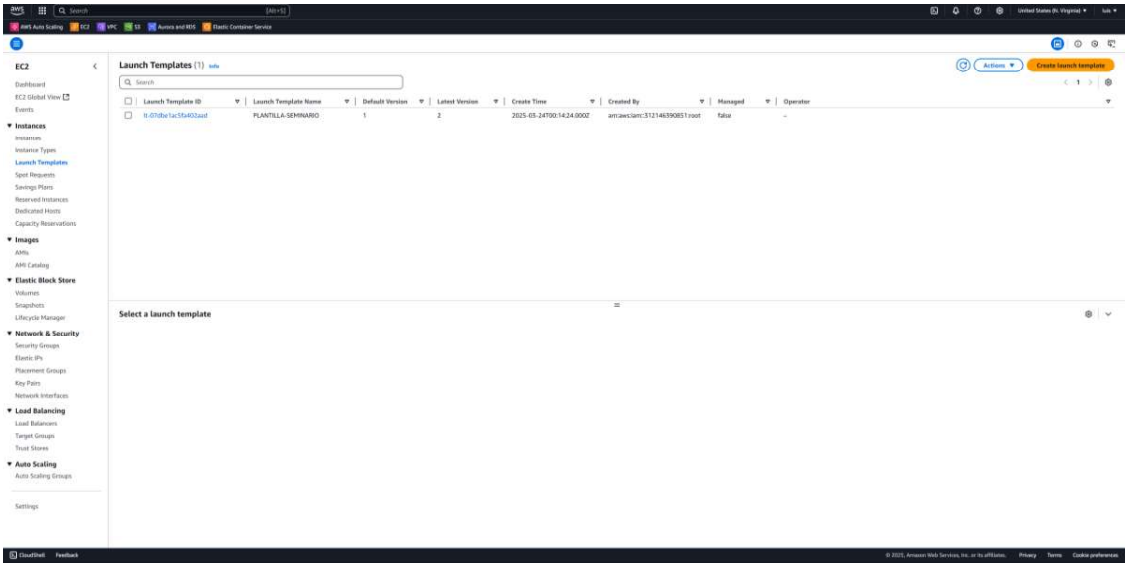


Imagen 18, pantalla de Launch Instance Template

Para crear un Launch Instance Template presionamos en el botón Create launch template de color naranja de la ilustración (Imagen 18).

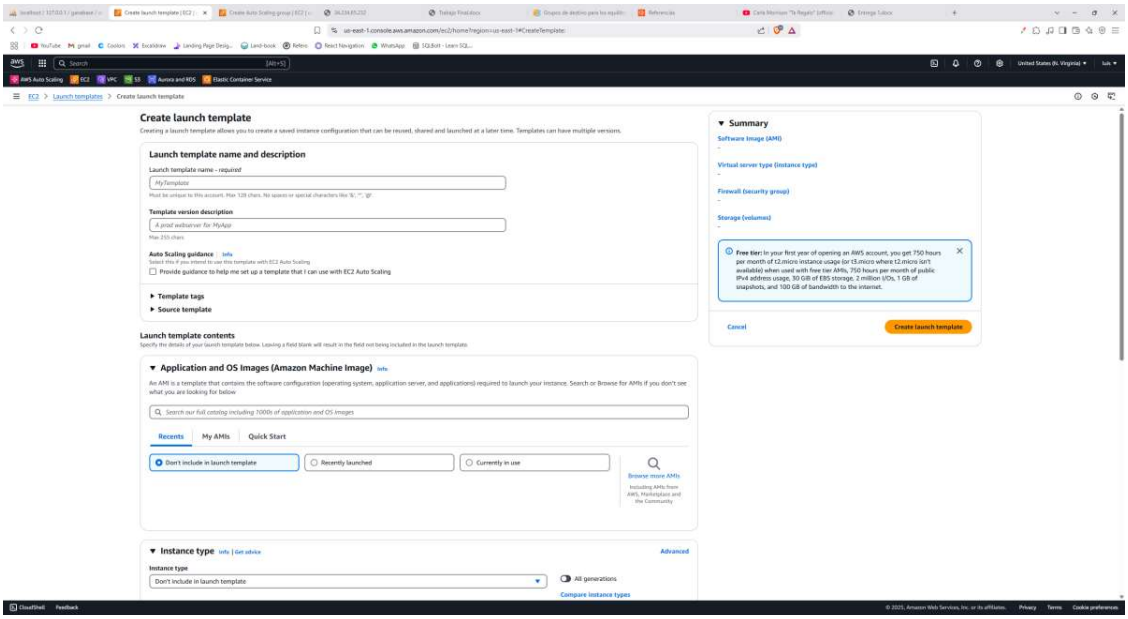


Imagen 19, formulario Create Launch Template

Rellenamos el formulario con la AMI y demás características deseadas, para este proyecto se creó una AMI previamente a partir de la instancia Amazon Linux que creamos al comienzo del proyecto.

Una vez creado el Template podemos crear el Auto Scaling Group. Para ello nos dirigimos al servicio EC2, luego en la parte inferior izquierda de la ilustración (Imagen 20) encontraremos el servicio Auto Scaling Group, es ahí donde nos dirigiremos.

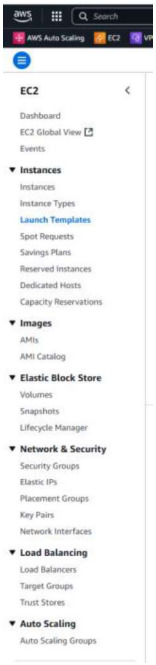


Imagen 20, Auto Scaling

Veremos el siguiente Dashboard.

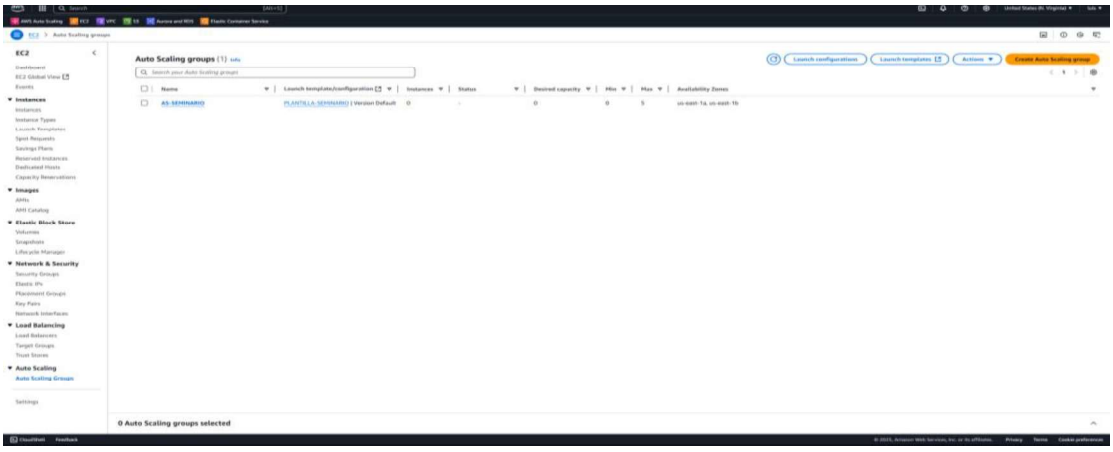


Imagen 21, Auto Scaling Groups Dashboard

Procedemos a crear un Auto Scaling Group por medio del botón Create Auto Scaling group de color naranja de la ilustración (Imagen 21).

Veremos la siguiente pantalla.

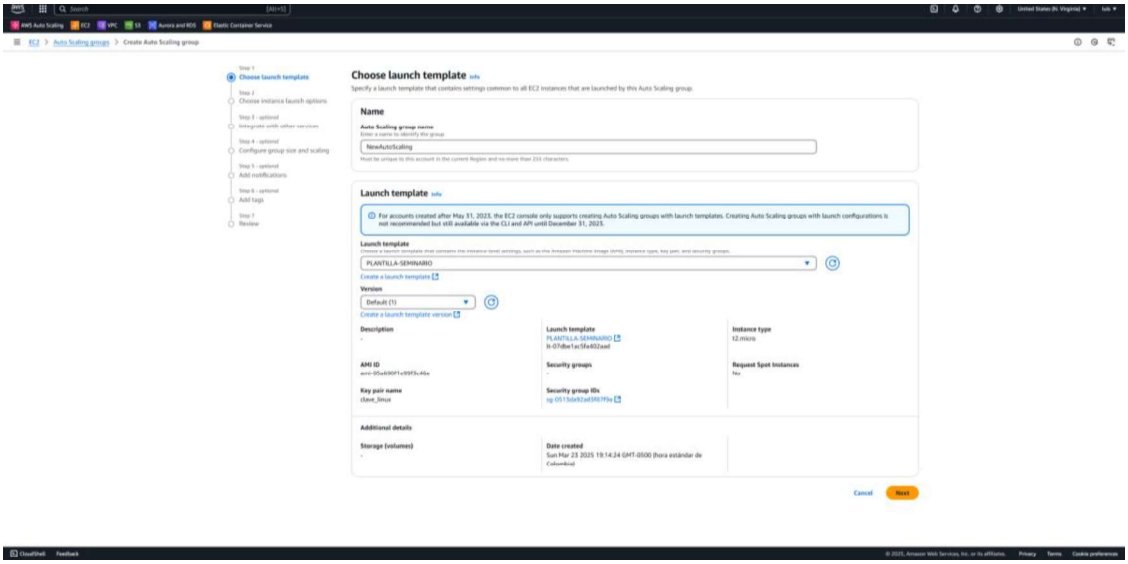


Imagen 22, Formulario de Auto Scaling group

Configuración de las políticas del Auto Scaling, se procedió a asignar el límite de uso de CPU que el Auto Scaling debe detectar en el monitoreo para efectuar la creación de nuevas instancias y el máximo de éstas que queremos crear.

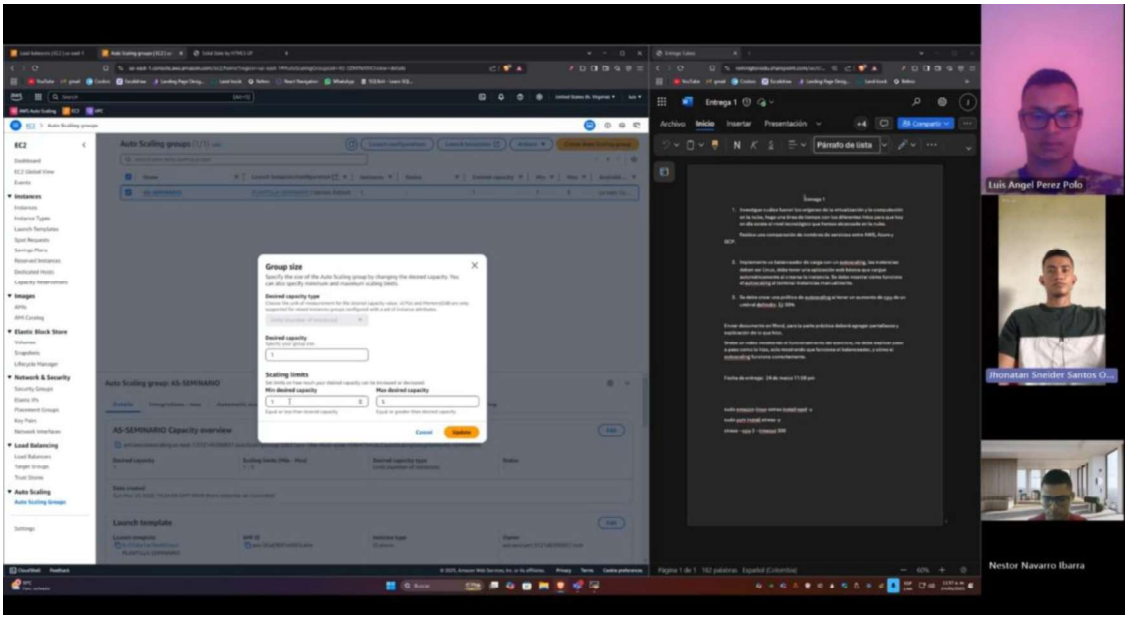


Imagen 23, Políticas del Auto Scaling Group

Luego nos dirigimos al servicio AWS Auto Scaling y procedemos a crear un plan de Auto Escalado, para esto presionamos el botón Get started de color naranja ilustrado en la parte superior derecha de la ilustración *Auto Scaling Dashboard* (Imagen 16).

Una vez presionado el botón seremos dirigidos a la siguiente pantalla.

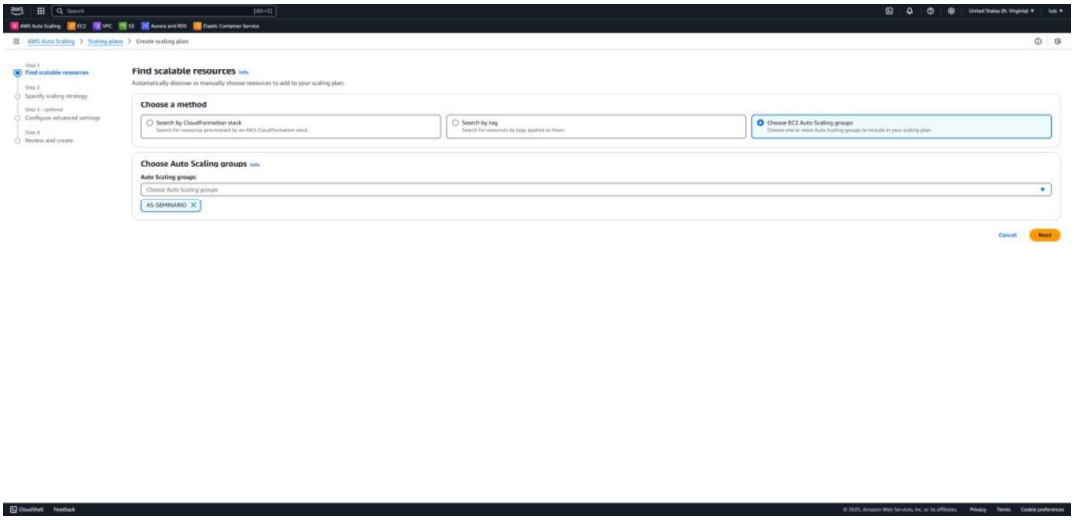


Imagen 24, Formulario del servicio AWS Auto Scaling

Procedemos a conectarnos a la instancia creada por el Auto Scaling Group a través de la Instancia principal a la cual tenemos acceso, luego saturamos la instancia del Auto Scaling para comprobar si crea nuevas instancias.

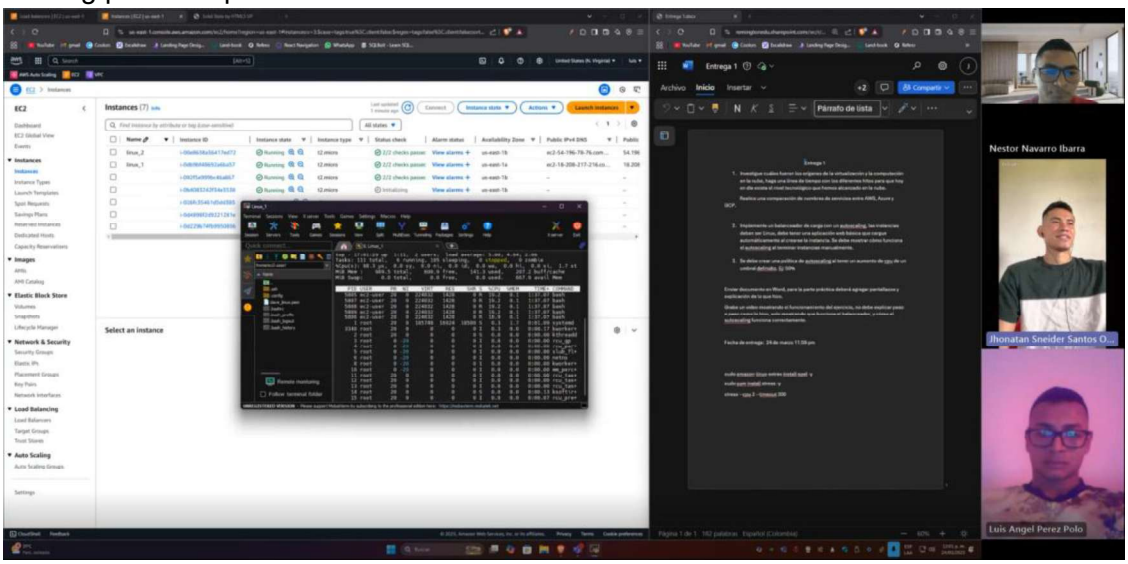


Imagen 25, saturación de la Instancia para poner a prueba el Auto Scaling

En la siguiente imagen (Imagen 26) se muestra el número de instancias corriendo al momento, 3 (1 instancias principal (AMI) y 2 creadas por el servicio de Auto Scaling group y la política de la preferencia de 2 instancias en caso de correcto funcionamiento)

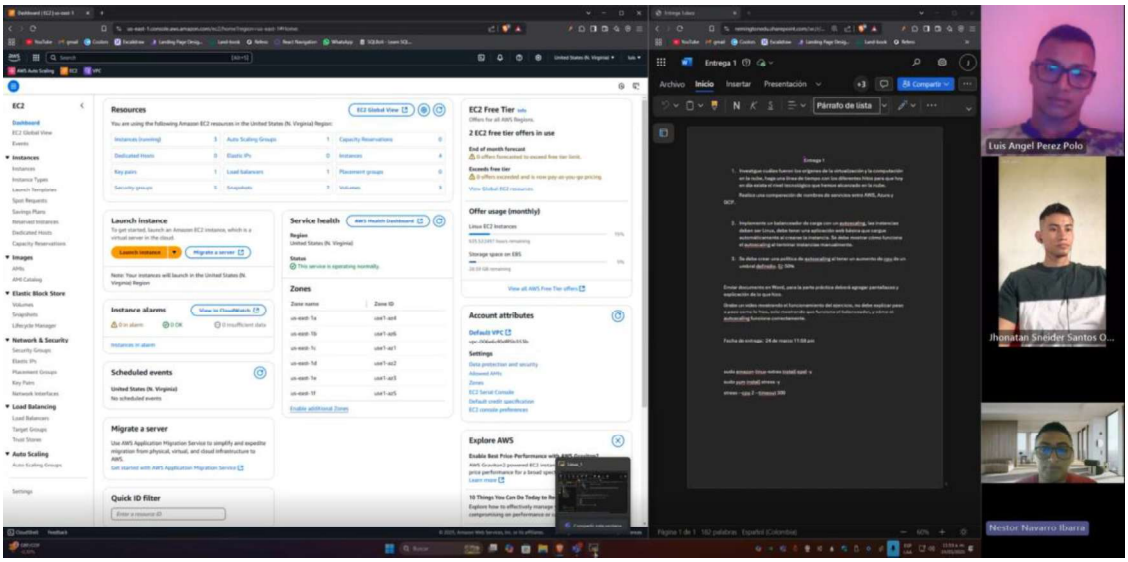


Imagen 26, Imagen previa al funcionamiento del Auto Scaling

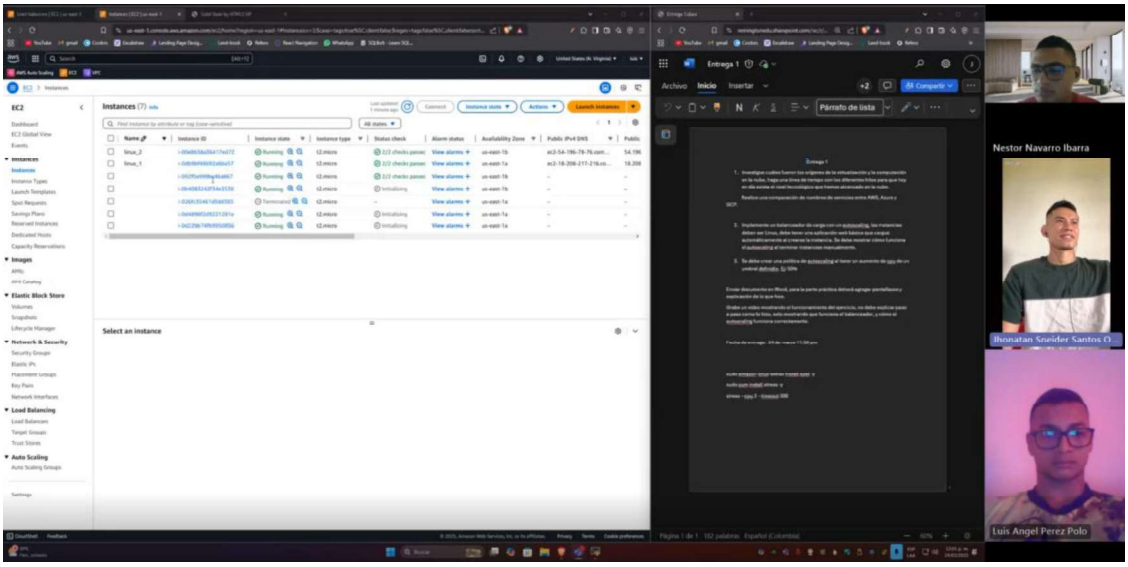


Imagen 27, Imagen posterior al funcionamiento del Auto Scaling

Como se puede observar en la Ilustración (Imagen 27) se encuentran corriendo 7 Instancias de las cuales 5 son del Auto Scaling, lo que demuestra que luego de la saturación de una de las dos máquinas del Auto Scaling, este crea automáticamente máquinas para compensar el mal funcionamiento de la instancia saturada.

Entrega 2

Para esta entrega se realizó la creación de un Cluster mediante la activación del servicio Amazon Elastic Container Service, la creación de contenedores dentro del mismo (servicios) y 2 tareas dentro de éste para poder asignarle un Load Balancer que direccione a ambos contenedores con diferente puerto dentro de cada contenedor. Para esto creamos un Cluster desde el servicio Elastic Container Service.

En esta parte del ejercicio establecimos un clúster ya que este nos va a permitir gestionar múltiples tareas.

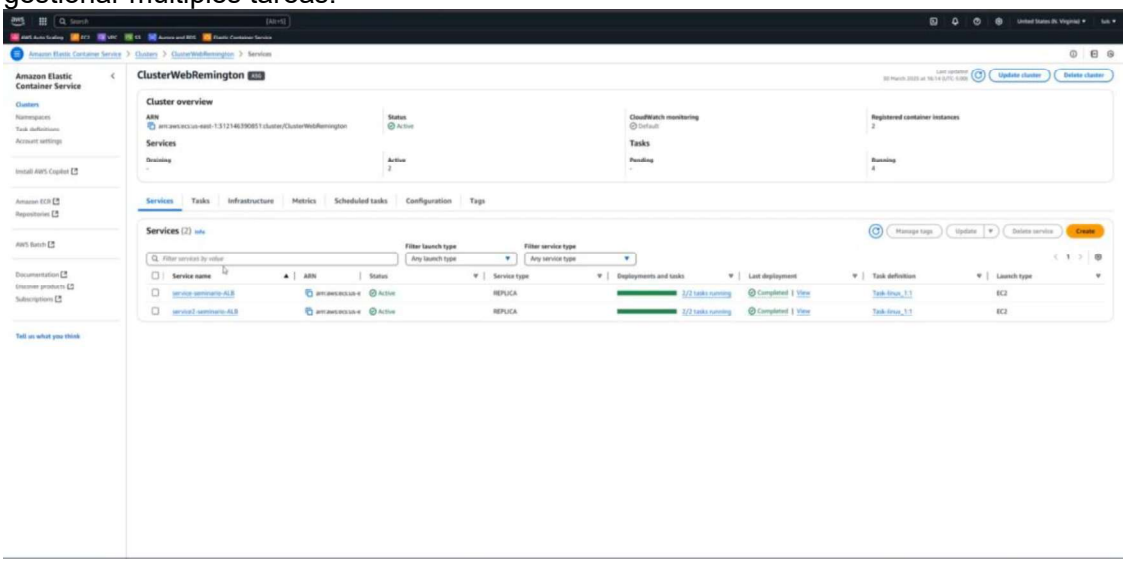


Imagen 28, Cluster DashBoard

Una vez creado el Cluster (ClusterWebRemington) procedimos a asignarle 2 servicios para poder testear el redireccionamiento del Load Balancer que se le va a asignar, los servicios podemos verlos en la parte inferior de la ilustración (Imagen 28).

Mientras tanto en nuestra máquina virtual creada en la primera entrega preparamos la Instancia acceder a la aplicación desde los contenedores y ya no desde la aplicación httpd instalada en la instancia.

Primero eliminamos la aplicación httpd de la instancia por medio del comando:

- sudo dnf remove httpd

Una vez removido httpd procedemos a instalar Docker

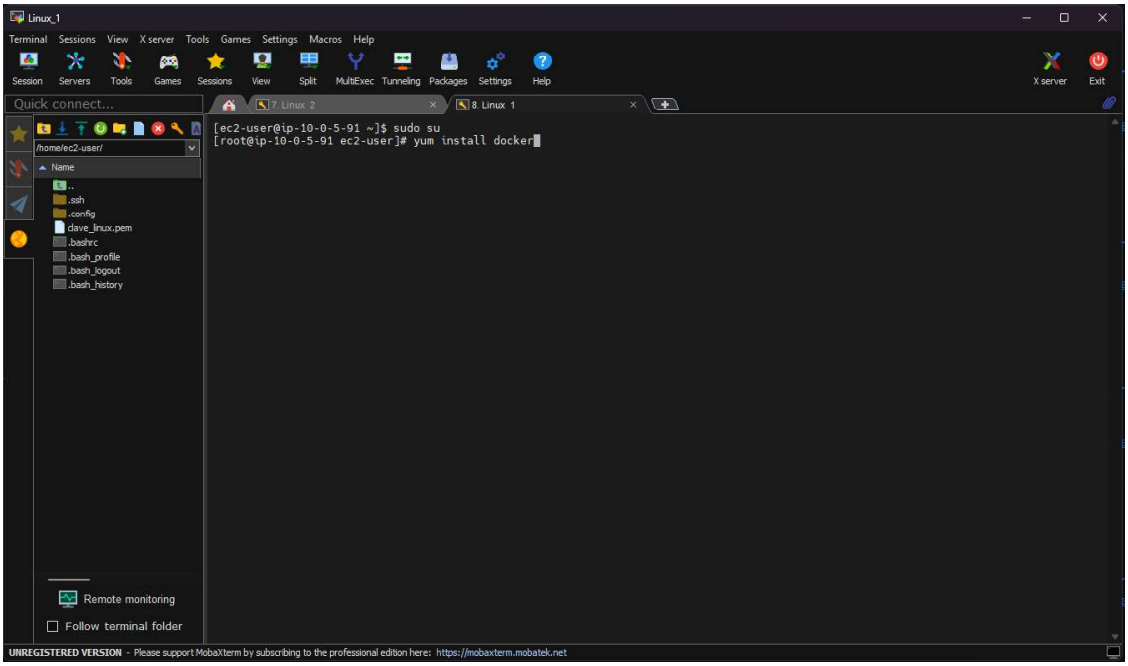


Imagen 29, Instalación de Docker en la instancia

Una vez instalado Docker, lo siguiente es buscar la imagen de httpd en DockerHub Docker Hub Container Image Library | App Containerization. (s. f.). https://hub.docker.com/ .

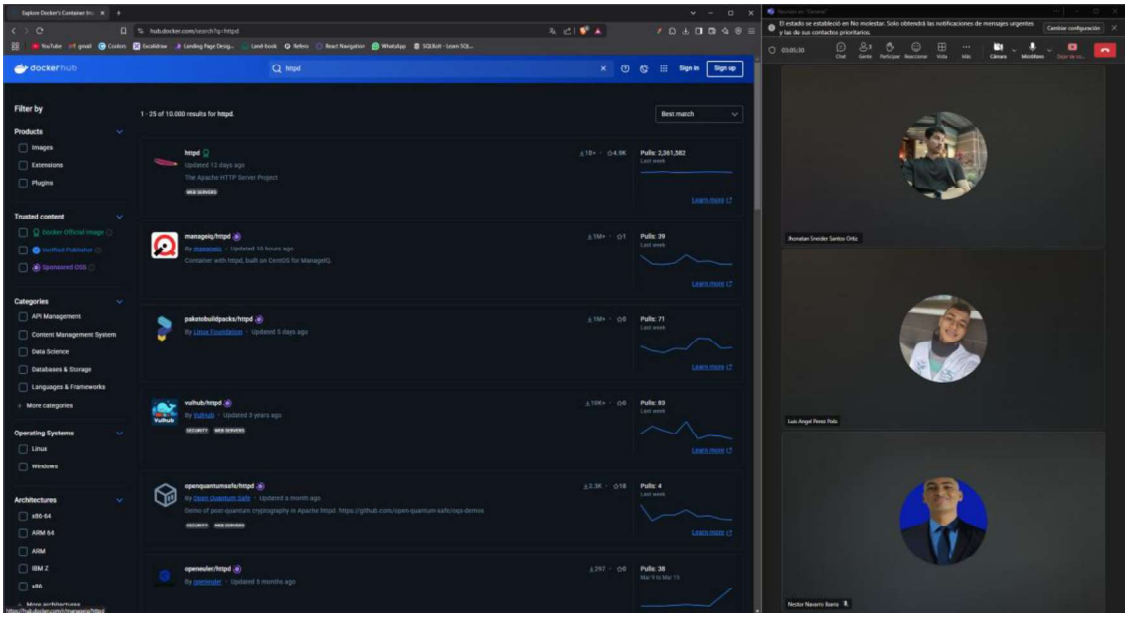


Imagen 30, Búsqueda de la imagen httpd en DockerHub

Creación de los contenedores en la instancia.

En la siguiente ilustración se muestra el comando utilizado para crear uno de los contenedores utilizados durante la realización de este proyecto para la entrega número 2.

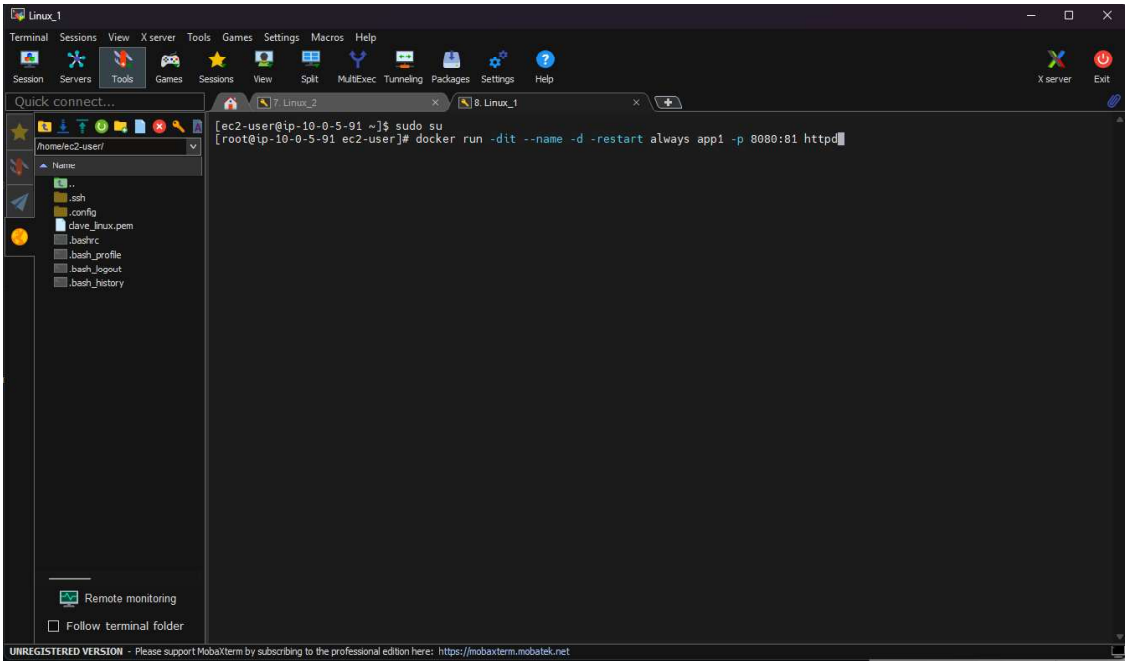


Imagen 31, Creación del contenedor Docker

De la ilustración (Imagen 31) podemos extraer porciones del código que nos permiten entender su funcionamiento, tales como:

-dit proporciona permisos al docker para ejecutarse en segundo plano

-restart always permite que el contenedor intente reiniciarse cada vez que por algún motivo deje de funcionar correctamente.

-p 8080:81 determina el puerto mediante el cual el contenedor recibirá las peticiones.

Httpd al final del comando determina la imagen que utilizaremos en el contenedor.

Asignación de un Load Balancer al Cluster

Se utilizó un Load Balancer para asignar un único punto de llegada a los contenedores, el Load Balancer determinará a que contenedor enviar la solicitud del cliente dependiendo del tráfico de solicitudes.

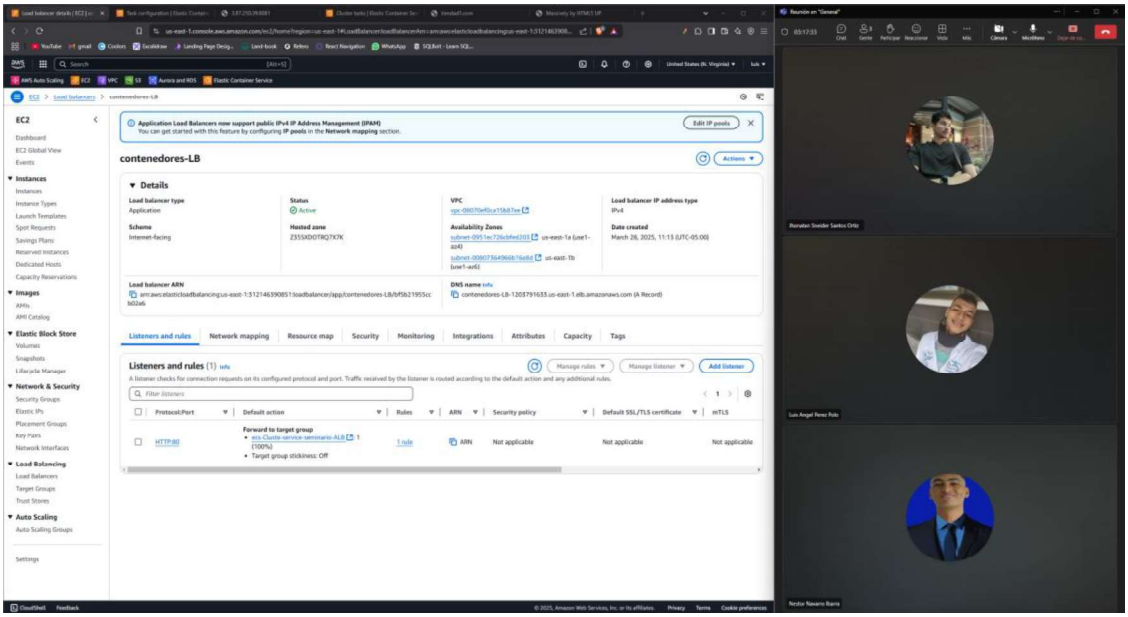


Imagen 32, Load Balancer designado al Cluster

Una forma de acceder y probar el funcionamiento de los contenedores es accediendo a la ip pública del Load Balancer del Cluster, y añadiendo el puerto nos permite la conexión al contenedor con el puerto escrito.

Para el desarrollo de este proyecto realizamos esta prueba, ilustraciones (Imagen 33 e imagen 34)

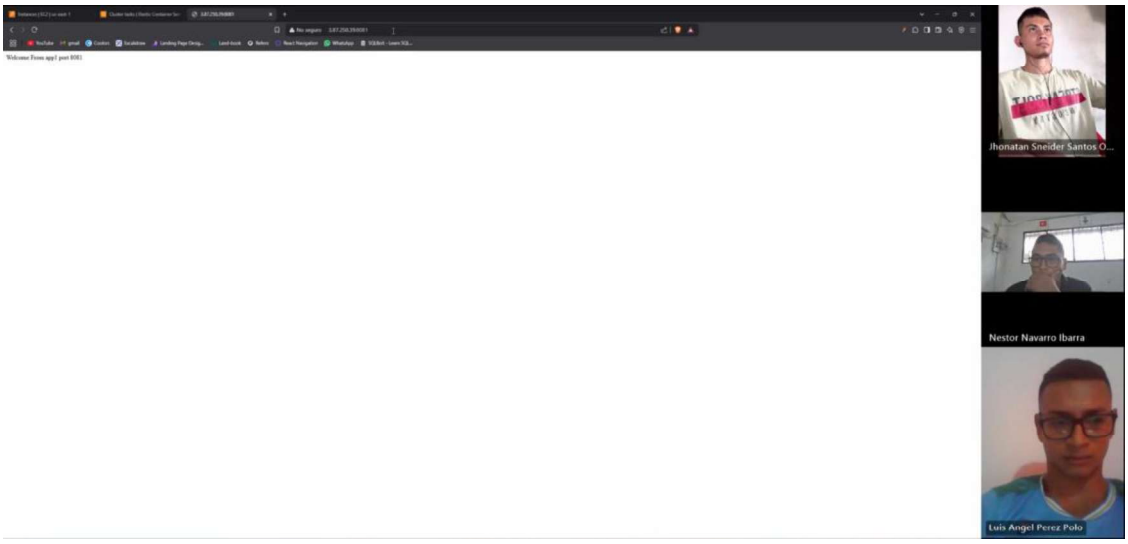


Imagen 33, Conexión manual al contenedor con el puerto 8081

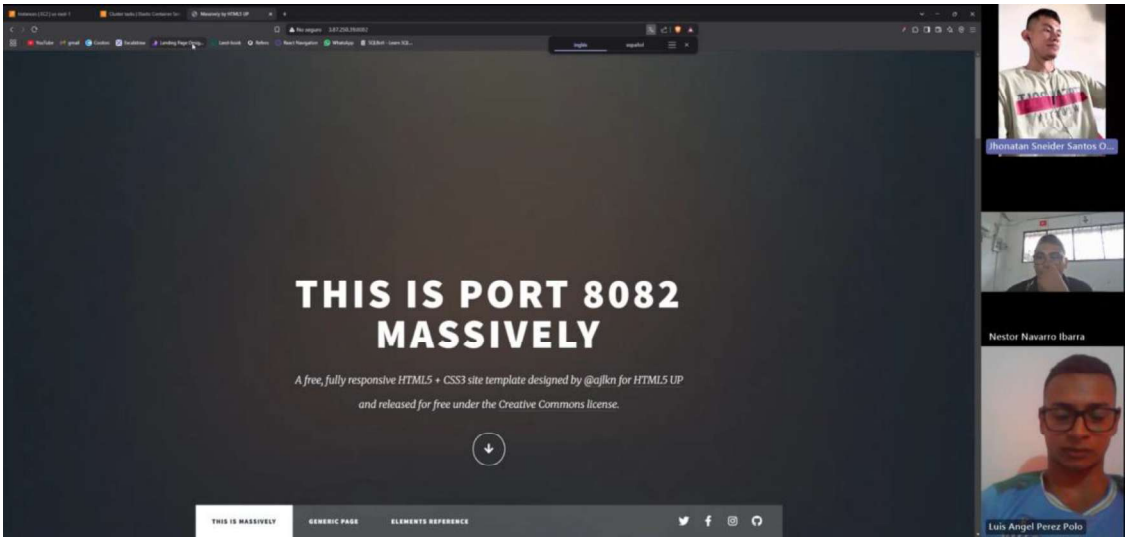


Imagen 34, Conexión manual al contenedor con el puerto 8082

Una vez demostrado el acceso correcto a los contenedores desde una ip pública a través del Load Balancer, procedimos a insertar manualmente en la máquina física dns a la dirección del Load Balancer asignado al Cluster, esto con el fin de verificar que el Cluster funciona correctamente junto con el Load Balancer y que el acceso es permitido incluso mediante un DNS (lo que se considera es la forma correcta de servir de una dirección a los clientes).
Para simular el funcionamiento de un DNS apuntando a nuestro Load Balancer ajustamos manualmente el archivo gestor de DNS del sistema operativo de la máquina física.

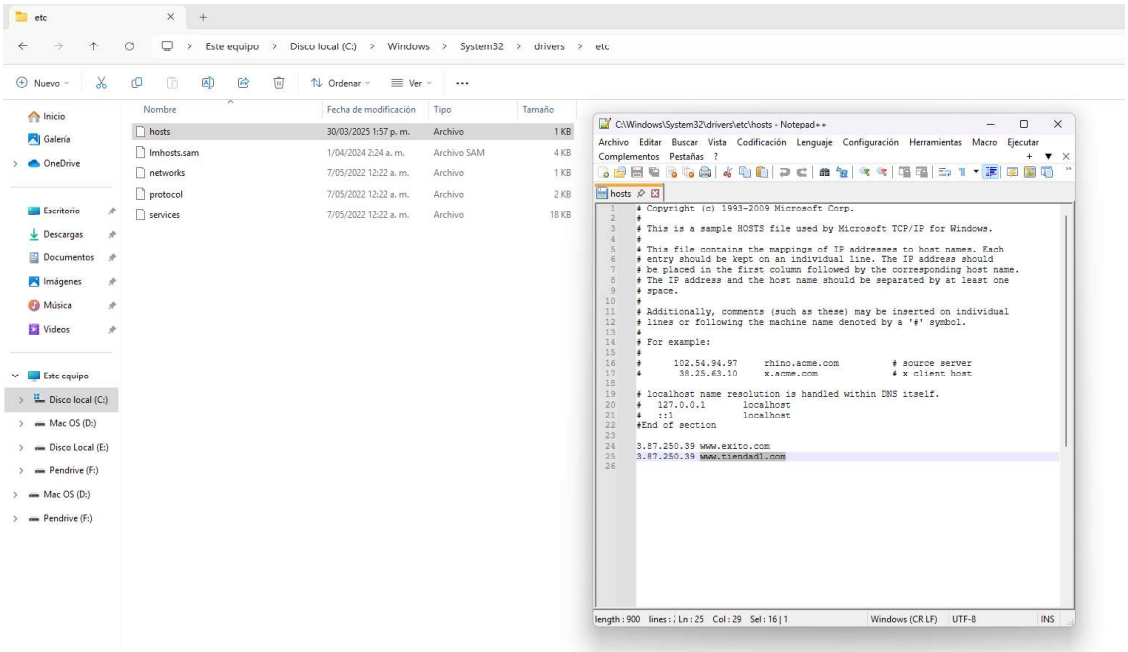
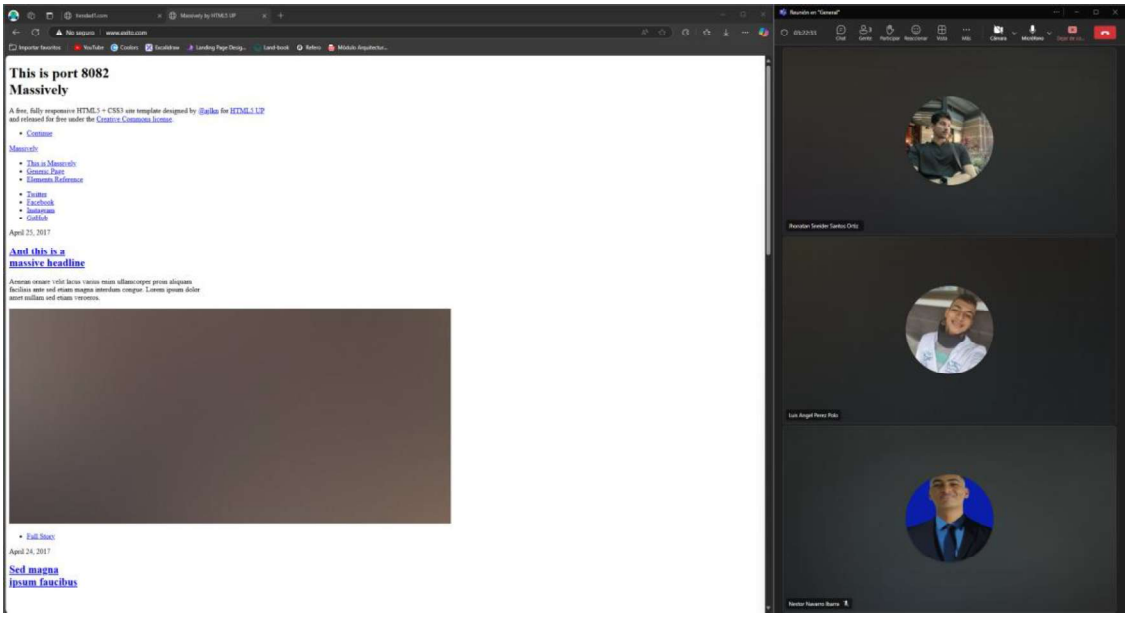


Imagen 35, asignación manual DNS

En la siguiente ilustración (Imagen 36) se demuestra como al introducir la dirección especificada en el archivo host de la máquina física, esta nos redirecciona a los contenedores y concluye con nuestra demostración de proyecto en la segunda entrega.



Imágen 36, Funcionamiento del redireccionamiento de la URL asignada.

Conclusiones

- Mediante este diplomado pudimos desarrollar la habilidad de determinar qué servicios de AWS deberíamos utilizar para suplir las necesidades que presente nuestro proyecto, aplicación, cliente o empresa que necesite de llevar su negocio al extenso y benéfico mundo del Cloud Computing.

- El entender como el Cloud Computing provee de facilidad de escalabilidad en cuanto a hardware y mantenibilidad de recursos nos impulsa a orientar nuestro conocimiento en la tecnología que sin duda ha revolucionado la forma de mantener la disponibilidad de modelos de negocio a nivel global.

- Orientar el perfil de Ingeniero de Sistemas al conocimiento de arquitectura AWS abre puertas a oportunidades para generar crecimiento profesional.

- Conocer y saber aprovechar los recursos que AWS ofrece nos permite generar una gran cantidad de oportunidades para gestionar nuestros propios proyectos, e incluso extender las habilidades profesionales para nuestro CV.

Referencias

- *¿Qué es Amazon VPC? - Amazon Virtual Private Cloud*. (s. f.). https://docs.aws.amazon.com/es_es/vpc/latest/userguide/what-is-amazon-vpc.html
- *¿Qué es Amazon EC2? - Amazon Elastic Compute Cloud*. (s. f.). https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html
- *What is Amazon Elastic Container Service? - Amazon Elastic Container Service*. (s. f.). https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html
- *Imágenes de máquina de Amazon (AMI) en Amazon EC2 - Amazon Elastic Compute Cloud*. (s. f.). https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AMIs.html
- *Grupos de destino para los equilibradores de carga de aplicaciones - Elastic Load Balancing*. (s. f.). https://docs.aws.amazon.com/es_es/elasticloadbalancing/latest/application/load-balancer-target-groups.html
- Mobatek. (s. f.). *MobaXterm*. https://mobaxterm.mobatek.net/
- [1] «jessup,» [En línea]. Available: https://jessup.edu/blog/engineering-technology/what-is-the-future-of-cloud-computing/.
- [2] «BMC,» [En línea]. Available: https://www.bmc.com/blogs/aws-vs-azure-vs-google-cloud-platforms/.
- [3] F5, «F5.COM,» [En línea]. Available: https://www.f5.com/es_es/glossary/load-balancer#:~:text=Un%20equilibrador%20de%20carga%20es,la%20fiabilidad%20de%20las%20aplicaciones..
- [4] universidadCesuma, «cesuma,» cesumamx, 22 febrero 2022. [En línea]. Available: https://www.cesuma.mx/blog/cloud-computing-historia-y-propiedades.html#Desarrollo_del_cloud_computing.