TRABAJO DE GRADO
Opción Seminario-Diplomado.

Implementación en la nube Amazon Web Services (AWS)

Corporación Universitaria Remington.
Nombre de la facultad: Ingeniería de Sistemas
Nombre del programa académico:
Tecnología en Desarrollo de Software

Nombres de los estudiantes autores del trabajo de grado.
Sergio Andrés Giraldo Herrera
Brayan Alexander Pantoja Mueses
Nombre del Tutor del trabajo de grado (docente del seminario o diplomado).
Juan Pablo Berrio López

Opción de Trabajo de grado Seminario-Diplomado.
Año de presentación del trabajo de grado.
2025

Tabla de Contenidos

Resumen

El seminario de Amazon Web Services fue realizado por Sergio Andrés Giraldo Herrera y Brayan Alexander Pantoja Mueses con la tutoría de Juan Pablo Berrio López, se centra en el uso de Amazon Web Services (AWS) en el seminario de la Corporación Universitaria Remington. AWS ofrece servicios de computación en la nube. Se tiene como objetivo compartir lo que se aprendió sobre cómo utilizar las herramientas básicas disponibles para desarrollar soluciones tecnológicas seguras, accesibles y eficientes.

Se realizaron actividades prácticas, como el uso de máquinas virtuales que funcionan a través de Internet durante el seminario. También se aprendió a usar balanceadores de carga, que están diseñados para distribuir el trabajo entre muchas máquinas para mejorar el rendimiento y prevenir las sobrecargas.

Se trabajo con instancias como servidor virtual para la ejecución en la nube de la aplicación y almacenamiento de los datos, permitiendo tener control sobre el entorno de ejecución de la aplicación, escalando recursos según la configuración adecuada a las necesidades que sean flexibles y escalables, y que nos permita ajustar los recursos de la aplicación.

Un tema importante fue la creación de redes virtuales privadas VPCs (Virtual Private Cloud), donde se configuraron subredes públicas y privadas. Esto permitió comprender cómo organizar los recursos dentro de una red y controlar el acceso para mayor seguridad, esto nos permite tener el control y la configuración en la nube que sea personalizable, garantizando la privacidad y seguridad de datos y aplicaciones.

Se trabajó con grupos de auto escalado, que permiten que el sistema agregue o quite recursos de manera automática según la demanda. Además, se utilizó el servicio de almacenamiento S3 (Simple Storage Services), que permite guardar archivos en la nube y acceder a ellos desde cualquier lugar y en cualquier momento, este servicio lo usamos para poder respaldar datos, para los sitios web estáticos y distribución de contenido.

Finalmente, se revisaron buenas prácticas para responder ante fallos o situaciones de emergencia, lo cual es clave para mantener los servicios disponibles y protegidos.

Palabras clave

(Incluya 5 palabras clave que representen su trabajo de grado)

- Balanceadores de carga
- Amazon web services
- Máquinas virtuales
- Redes VPC y subredes
- Almacenamiento en la nube

Marco conceptual y contextual

La computación en la nube es un modelo que permite el acceso a la red bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de gestión o interacción con el proveedor de servicios.

Esto significa que los usuarios pueden utilizar estos servicios donde y cuando los necesiten, en lugar de estar limitados a dispositivos físicos dentro de un edificio o centro de datos, sin mencionar que se paga solo por lo que se usa.

Amazon Web Services (AWS) es una de las plataformas más populares para este tipo de servicio, permitiendo despliegues de soluciones rápidas, seguras y escalables.

Conceptos claves

Conceptos fundamentales de las soluciones en la nube, incluyendo su arquitectura, servicios de computación, modelos de seguridad en la nube y mejores prácticas para gestionar recursos, fueron cubiertos en el seminario de AWS:

- **Instancias EC2 (Elastic Compute Cloud):** Máquinas virtuales personalizables que permiten ejecutar aplicaciones en la nube y pueden escalarse dinámicamente para satisfacer las necesidades del sistema. Estas instancias también pueden usarse para desarrollo, pruebas o como entorno de producción, brindando flexibilidad y control sobre el entorno. Puedes seleccionar el sistema operativo, capacidad de procesamiento, memoria y almacenamiento según las necesidades del proyecto.

- **VPC (Virtual Private Cloud):** Son redes virtuales privadas que permiten ser configurables dentro de AWS, permitiendo organizar los recursos de forma segura y segmentada, incluyendo subredes públicas y privadas para una gestión eficiente de la infraestructura en la nube. Estas redes permiten tener control sobre cosas como las direcciones IP, las rutas por donde pasa la información, las reglas de seguridad y las conexiones hacia internet. Gracias a eso, se pueden crear entornos personalizados según lo que necesite el sistema, y también se puede controlar quién accede a qué recurso dentro de la red.

- **Autoescalado (Auto Scaling Groups):** es un servicio de Amazon Web Services (AWS) para ajustar el número de instancias EC2 según la demanda del sistema, incrementando o disminuyendo su cantidad.

- **Almacenamiento S3 (Simple Storage Service):** un servicio de almacenamiento de objetos que permite un almacenamiento seguro y escalable de archivos que se pueden acceder desde cualquier lugar. La opción de almacenamiento lista para usar

que permite el acceso y proporciona almacenamiento en la nube a un gran volumen de datos.

- **Balanceadores de carga (Load Balancers):** servicios que distribuyen de manera equitativa el tráfico entre varias instancias de servidor para aumentar la disponibilidad y el rendimiento de las aplicaciones. Los balanceadores de carga desempeñan un papel crucial en el rendimiento de las aplicaciones al distribuir eficientemente la carga entre varios servidores, evitando que un solo servidor se sobrecargue y garantizando una experiencia sin interrupciones para los usuarios, mejorando así la escalabilidad y fiabilidad de las aplicaciones.

Concepto de aplicación

Este trabajo de grado se realizó como parte del seminario sobre Amazon Web Services (AWS) en la Corporación Universitaria Remington, con el propósito de aplicar los conocimientos adquiridos en la implementación del uso de la tecnológica en la nube. La simulación de escenarios de implementación en AWS fortalece las competencias técnicas en infraestructuras escalables y seguras, integrando áreas como redes, administración de servidores, seguridad informática y desarrollo, bajo un enfoque de arquitectura en la nube o cloud. Esta experiencia fue clave para comprender el funcionamiento de los servicios en la nube y su aplicabilidad en proyectos reales.

Desarrollo e implementación del aprendizaje

En la primera entrega del seminario AWS se investigó sobre la virtualización y la computación en la nube, se creó una línea de tiempo con los diferentes hitos que han existido y que hoy en día existen a nivel tecnológico en la nube. Se implemento un balanceador de carga con autoscaling en el sistema Linux, con una aplicación web básica que se pueda cargar automáticamente al crearse la instancia, demostrando también el funcionamiento del autoscaling al terminar instancias manualmente, Se crea también una política de autoscaling y terminamos con la investigación de cómo se puede hacer que con los nombres de los dominios de forma local este sea enviado a un contenedor diferente.

Desarrollo seminario AWS

1. Investigar cuáles fueron los orígenes de la virtualización y la computación en la nube, haga una línea de tiempo con los diferentes hitos para que hoy en día exista el nivel tecnológico que hemos alcanzado en la nube.

Virtualización:

Aunque la virtualización y la computación en la nube son conceptos relativamente nuevos, sus raíces pueden rastrearse hasta una variedad de tendencias tecnológicas a lo largo de los años, incluyendo:

En la década de 1960, IBM fue pionera en la virtualización en los mainframes para ejecutar múltiples sistemas operativos en un solo hardware y mejorar la utilización de los recursos.

En la década de 1970, IBM desarrolló las primeras máquinas virtuales verdaderas con VM/370, que sentaron las bases para la virtualización moderna al permitir que múltiples sistemas operativos compartieran los mismos recursos.

En los años 90, la virtualización x86 fue introducida por la empresa VMware, permitiendo la consolidación de servidores y la creación de entornos eficientes.

La virtualización explotó en la década de 2000 con lanzamientos de primer nivel como el ESX Server de VMware y el Virtual PC de Microsoft, junto con soluciones de código abierto como Xen y KVM, que allanaron el camino para opciones robustas pero accesibles entre los entornos empresariales y de desarrollo.

Computación en la nube:

Una nueva era comenzó con el lanzamiento en 2006 de Amazon Web Services (AWS), revolucionando así la computación en la nube. A través de AWS, las empresas podían consumir recursos de computación de manera elástica y a escala de internet sin tener que invertir en hardware físico.

A esto le siguieron empresas como Microsoft con Azure y Google con Google Cloud Platform, que desarrollaron aún más los servicios de computación en la nube. Estos fueron desarrollos históricos en la historia de la virtualización y la computación en la nube, revolucionando la forma en que se gestionan los recursos de computación y se ofrecen los servicios empresariales.

Linea de Tiempo:

En los años 1950-1960, los primeros pasos en cuanto a la virtualización se dieron de la siguiente manera:

- IBM fue pionera en el tiempo compartido en 1959, permitiendo que múltiples usuarios pudieran acceder a la misma computadora al mismo tiempo, haciendo un uso eficiente de los recursos.
- IBM inventó la virtualización en 1960 para hacer que los mainframes más grandes (y más costosos) fueran lo más eficientes posible.

- En 1964, el desarrollo del sistema CP-40 por parte de IBM, la base para el CP-67, fue el primer paso hacia el funcionamiento de múltiples sistemas operativos en una sola máquina.

La virtualización avanzó notablemente en la década de 1970:

- IBM creó las primeras máquinas virtuales con la capacidad de implementar múltiples sistemas operativos simultáneamente en un solo hardware en 1970.
- En 1974, se publicó un documento clave sobre hipervisores, que son programas esenciales para la virtualización, allanando el camino para futuros avances.

Algunos desarrollos importantes de las décadas de 1980 y 1990 que hicieron más generalizado el uso de la computación distribuida y la virtualización fueron:

- La idea del modelo cliente-servidor nació en la década de 1980, lo que permitió intercambiar recursos a través de redes y fue el comienzo del concepto de la nube.
- IBM lanzó la primera PC en 1981, lo que disminuyó la dependencia de las empresas de los mainframes.
- En la década de 1990, el lanzamiento de la virtualización x86 por VMware transformó fundamentalmente el mercado, permitiendo la consolidación de servidores y la optimización de costos para las empresas.
- Salesforce ofreció una de las primeras instancias de Software como Servicio (SaaS) en 1999, demostrando la posibilidad técnica de entrega de software por Internet.

La computación en la nube cobró relevancia en la década de 2000:

- Lo destacado fue en 2003 cuando VMware comercializó el servidor para la virtualización, lo que permitió a las empresas operar máquinas virtuales.
- En 2006, AWS introdujo EC2 y S3: la computación en la nube moderna comenzó aquí, cambiando todo.
- La infraestructura basada en la nube: en 2008, OpenStack un proyecto de código abierto para crear infraestructuras de nube fue lanzado.
- Microsoft entró al campo de los proveedores de servicios en la nube con su propia plataforma de servicios, Azure, en 2009.

La computación en la nube creció y maduró en la década de 2010:

- Docker revolucionó las cosas en 2010 con contenedores que hicieron el despliegue más fácil que nunca.
- En 2013, Google open-source Kubernetes, que es por mucho la plataforma de orquestación de contenedores más popular en entornos de nube híbrida.

- La computación sin servidor (AWS Lambda, Google Cloud Functions, Azure Functions, etc.) se consolidó en 2019, sirviendo código sin complicaciones de servidores.

Tendencias en inteligencia artificial y nube híbrida en la década de 2020:

- La adopción de la nube se aceleró en 2020 cuando la pandemia de COVID-19 forzó a la mayoría de las empresas a trabajar de forma remota y muchas otras a mudarse en línea.
- La nube se integró con la IA en 2021, allanando el camino para el entrenamiento de modelos avanzados en plataformas como AWS, Google Cloud y Azure.
- En 2023, surgieron estrategias de nube híbrida y multinube, combinando los beneficios de las nubes públicas y privadas.
- La nube se acerca a la computación cuántica y a la computación en el borde entre 2024 y 2025, acercando el procesamiento de datos a los usuarios finales.

comparación de nombres de servicios entre AWS, Azure y GCP.

Categoría	AWS	Azure	GCP
Computación	EC2 (elastic compute cloud)	Virtual Machines (VMs)	Compute Engine
	Lambda (serverless)	Azure functions(serverless)	Cloud functions (serverless)
Almacenamiento	S3 (simple storage service)	Blob storage	Cloud storage
	EBS (elastic block store)	Managed disks	Persistent disks
Bases de Datos	RDS (relational database service)	Azure SQL database	Cloud SQL
	DynamoBD (NoSQL)	Cosmos BD (NoSQL)	Firestore (NoSQL)
Redes	VPC (virtual private cloud)	Virtual network (vnet)	Virtual private cloud (vpc)
	Route 53(DNS)	Azure DNS	Cloud DNS
Big data y analytics	EMR (elastic MapReduce)	HDInsight	Dataproc
	Athena (consultas en s3)	Azure data lake analytics	bigQuery (consultas en datos)
Machine learning	sageMaker	Azure machine learning	IA platform

	Recognition (visión por computadora)	Computer vision	Vision AI
Contenedores y Kubernetes	ECS (elastic container service)	Azure Kubernetes service (AKS)	Google Kubernetes engine (GKW)
	Eks (elastic Kubernetes service)	Aks (azure Kubernetes service)	Gke (google Kubernetes engine)
Seguridad	IAM (identity and access management)	Azure active directory (AD)	IAM (identity and access management)
	KMS (key management service)	Key vault	Cloud key management service (KMS)
DevOps	CodeBuild (integración continua)	Azure devOps	Cloud Build
	CodePipeline (entrega continua)	Azure pipelines	Cloud build (integración y entrega)
Monitorización	CloudWatch	Azure Monitor	Cloud monitoring
	x-Ray (trazabilidad)	Application insight	Cloud trace
Mensajería y colas	SQS (Simple Queue Service)	Queue Storage	Cloud Pub/Sub
	SNS (Simple Notification Service)	Service Bus	Cloud Pub/Sub

Al observar las ofertas de los grandes proveedores de servicios en la nube. AWS, Azure y GCP, se puede ver fácilmente que, a pesar de los diferentes enfoques sobre cómo brindan los servicios y las diferentes terminologías, en realidad, todos hacen lo mismo en áreas clave como computación, almacenamiento, bases de datos, aprendizaje automático y seguridad.

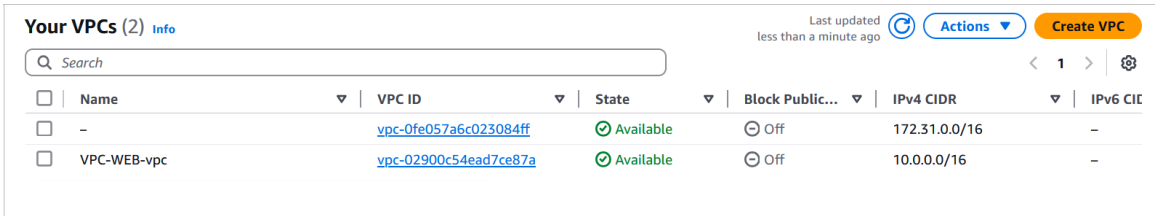
AWS es como el más veterano del grupo: utiliza nombres muy directos y fue el primero en lanzar servicios como EC2 y S3, lo que lo ha hecho el más popular. Sin embargo, Azure es el que mejor se integra con las herramientas de Microsoft, con designaciones simples (piensa en Máquinas Virtuales o Azure SQL Database). Es el más adecuado para empresas que ya utilizan Windows u Office. GCP, en comparación, es el innovador: se destaca en campos como el análisis de datos con BigQuery y la inteligencia artificial con AI Platform, e integra muy bien con otras herramientas de Google.

En resumen, la decisión de elegir entre AWS, Azure o GCP depende en gran medida de las necesidades de la organización, su tecnología actual y los mercados en los que se centran.

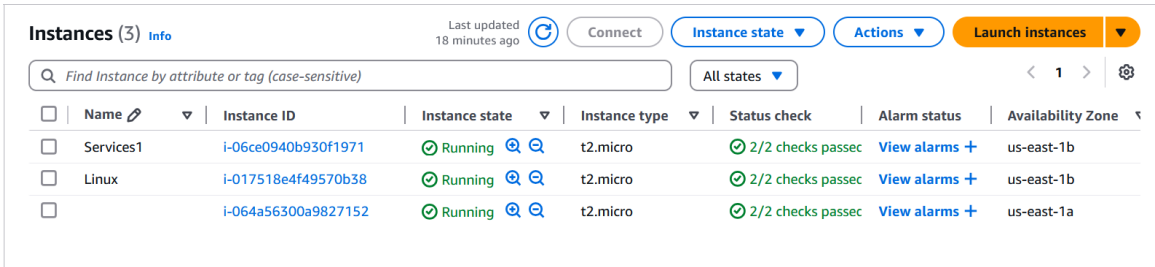
AWS es ideal para aquellos que buscan un mercado maduro, características y servicios; Azure es el mejor para las empresas con infraestructuras existentes de Microsoft; y GCP es el mejor para los usuarios interesados en innovación y análisis de datos. Por lo tanto, cada uno tiene sus ventajas, y muchas empresas adoptan el enfoque de combinar las tres plataformas para maximizar las ventajas de cada una.

2. Implemente un balanceador de carga con un autoscaling, las instancias deben ser Linux, debe tener una aplicación web básica que cargue automáticamente al crearse la instancia. Se debe mostrar cómo funciona el autoscaling al terminar instancias manualmente.

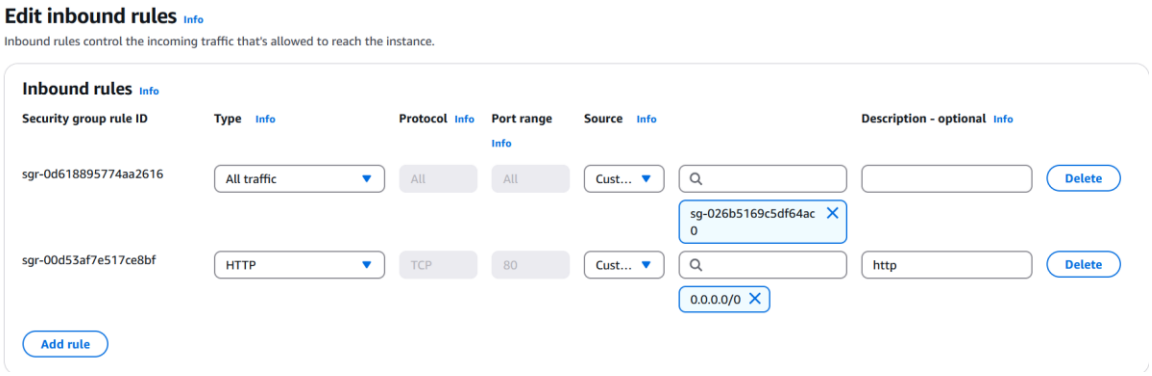
Se crea un VPC (Virtual Private Cloud) configurable dentro de la infraestructura en la nube de AWS, que nos permite crear reglas de conectividad, seguridad y direccionamiento IP personalizadas como ejecutar instancias, configuración de subredes públicas y privadas, como también configurar balanceadores de carga.



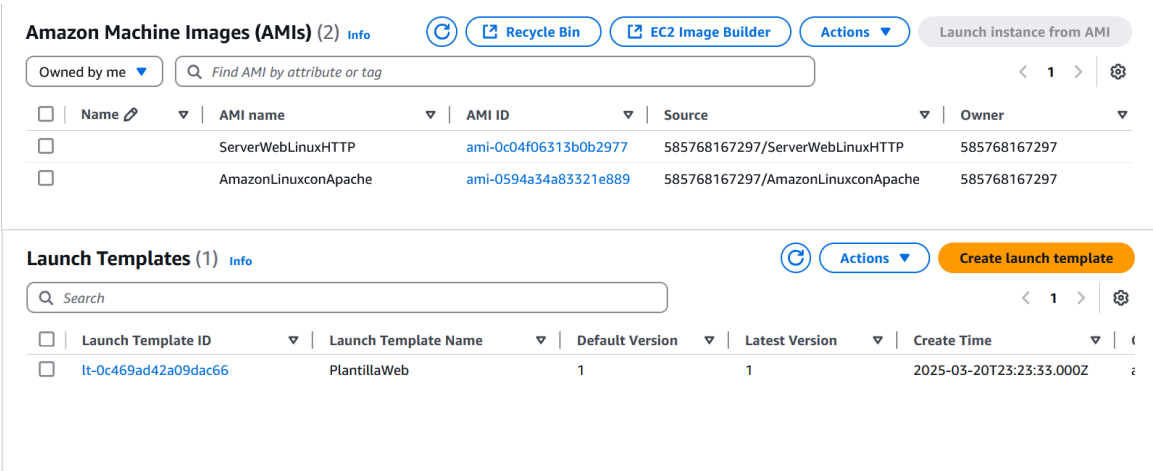
Se crea la instancia, se escoge el AMI de Linux sistema operativo con el cual se va a trabajar, tipo de instancia en este caso un t2.micro, cantidad de instancias, se configura la VPC y subred publica y configuración de políticas, configuración de almacenamiento por defecto son 8GB, se configura las reglas de seguridad (security groups), se elige la clave SSH (Key Pair).



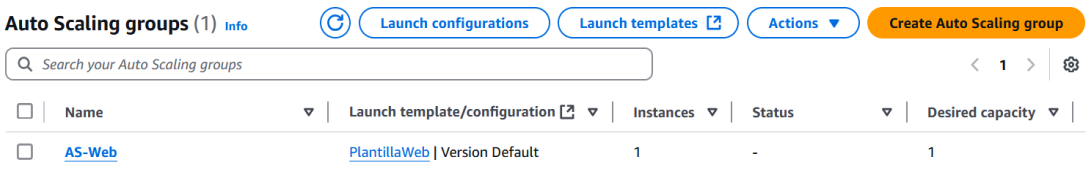
Se configura el cortafuego virtual en los security groups para las instancias EC2, estos son reglas de seguridad que controlan el tráfico de entrada a las instancias u otros recursos. Estas reglas definen qué tipo de tráfico está permitido desde la dirección IP o redes específicas.



Se selecciona la instancia y se crea una imagen y plantilla, se crea la imagen y se le asigna un nombre (Amazon Linux con Apache).

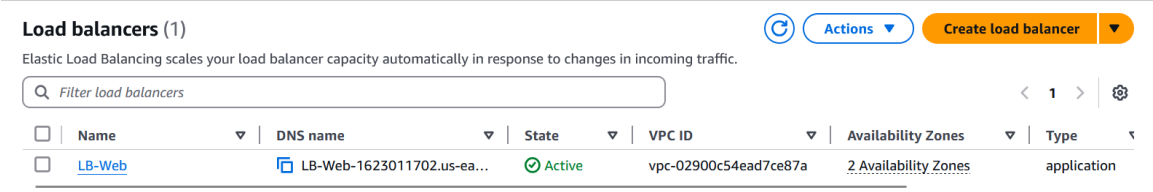


Se crea el Auto Scaling Group con el launche template que se creó usando la AMI, se configura el tamaño del grupo como es el desired capacity, mínimum capacity, máximum capacity, se elige la red y subnets, se configura políticas, configurar notificaciones y crear.



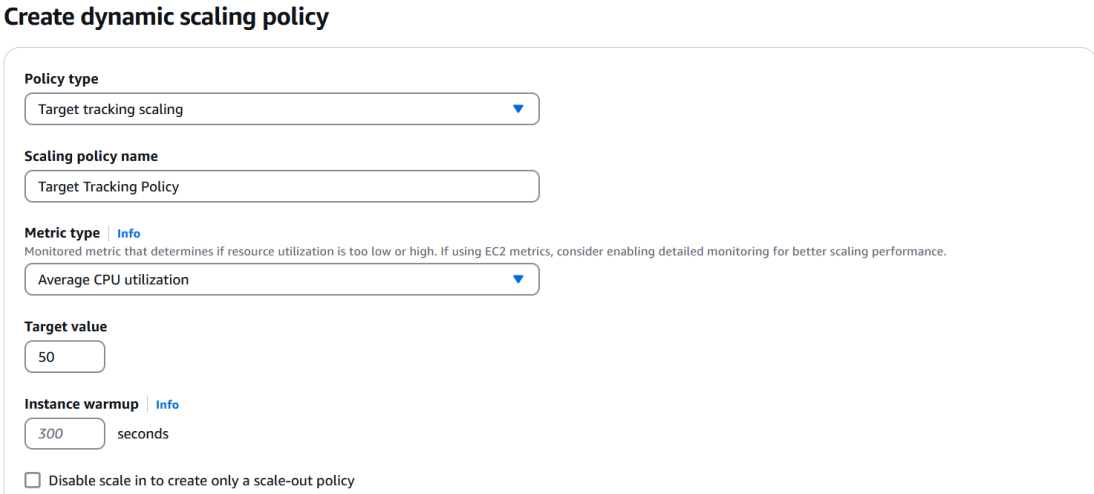
Se crea el load balancer para distribuir automáticamente el tráfico de red o de aplicación entre múltiples instancias, contenedores o direcciones IP y una o varias zonas disponibles, el objetivo principal es mejorar la disponibilidad, escalabilidad y tolerancia a fallos de una

aplicación. Lo que hace la distribución de carga es repartir el tráfico de manera eficiente entre varios servidores, evitando la sobrecarga de una sola instancia, entonces si una instancia falla, el Load Balancer redirige el tráfico a otras instancias disponibles; con la escalabilidad automática se integra con el Auto Scaling Groups, agregando o eliminando instancias según la demanda.



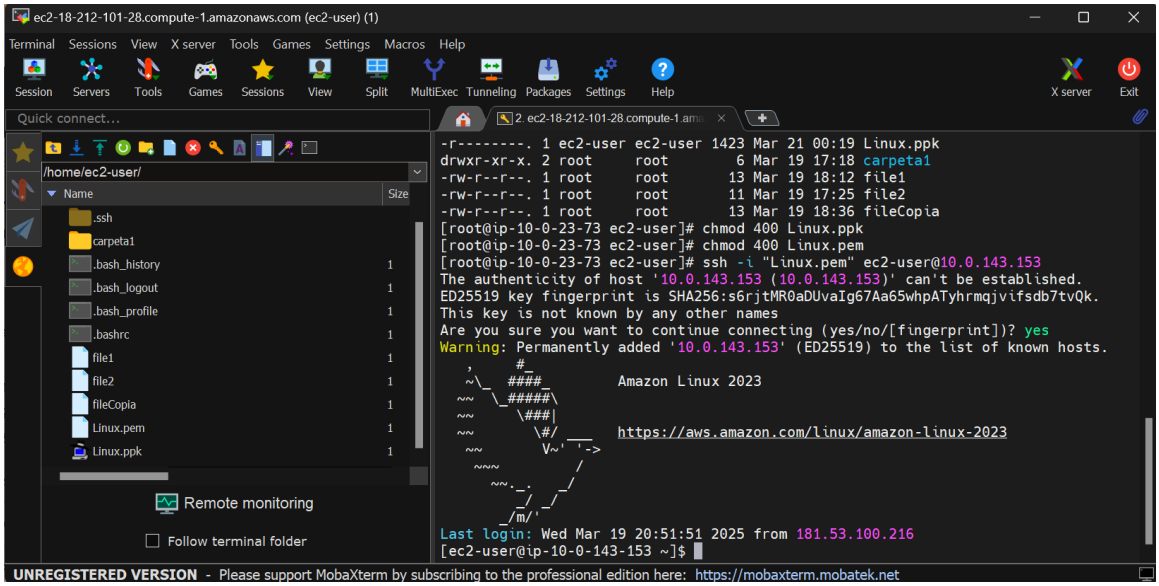
3. Se debe crear una política de autoscaling al tener un aumento de cpu de un umbral definidio. Ej: 50%

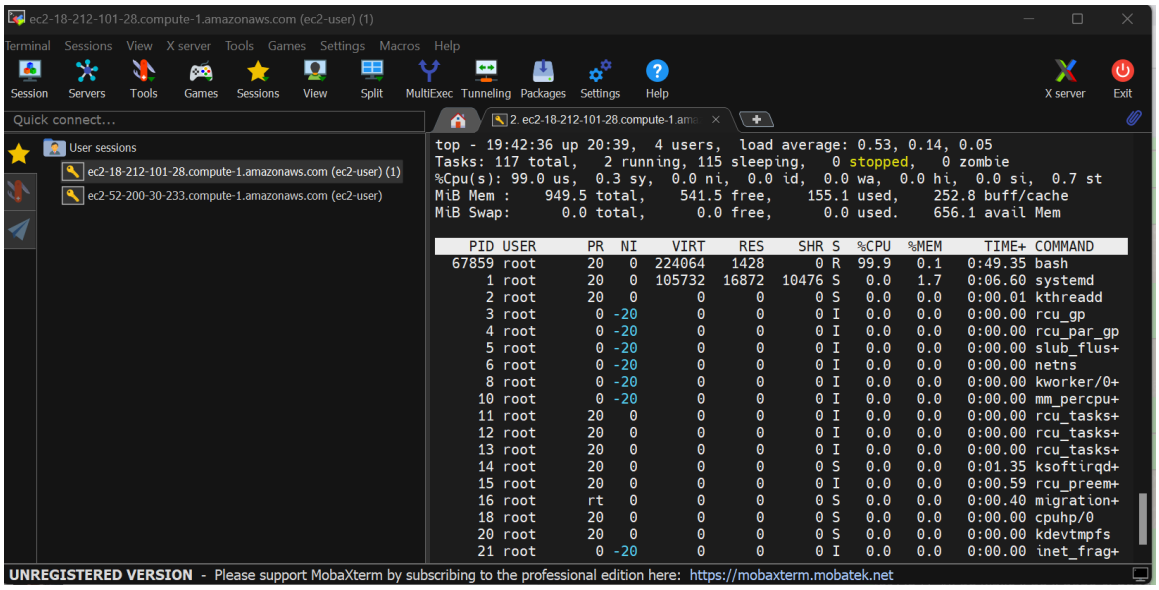
Se crea el Auto Scaling Group con políticas basadas en la CPU con un porcentaje del 50%. Esto permite que la infraestructura escale dinámicamente y mantenga un rendimiento óptimo sin intervención manual.



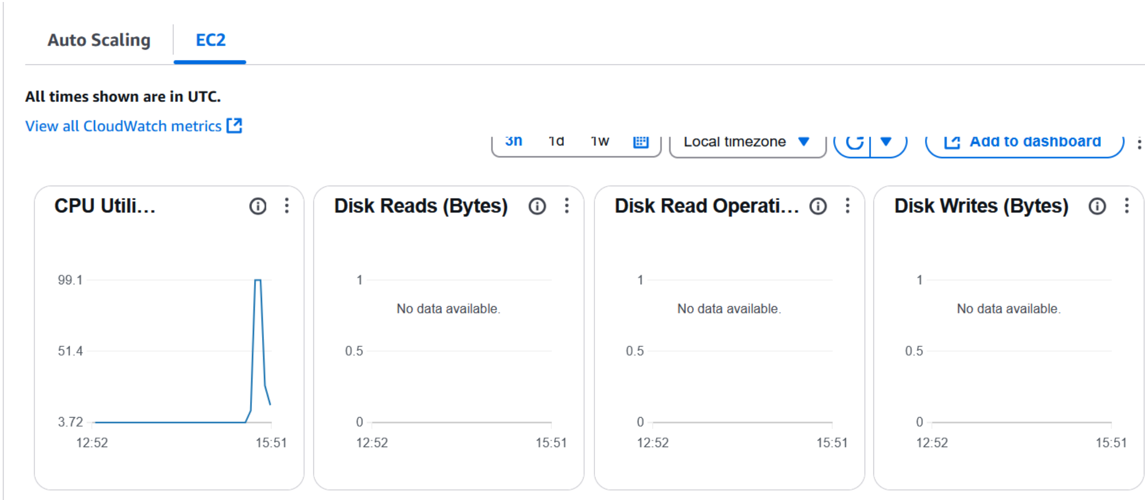
Funcionamiento

Desde la consola hacemos la prueba de estrés de CPU





En la parte de Auto Scaling Groups podemos ver el monitoreo y la actividad.

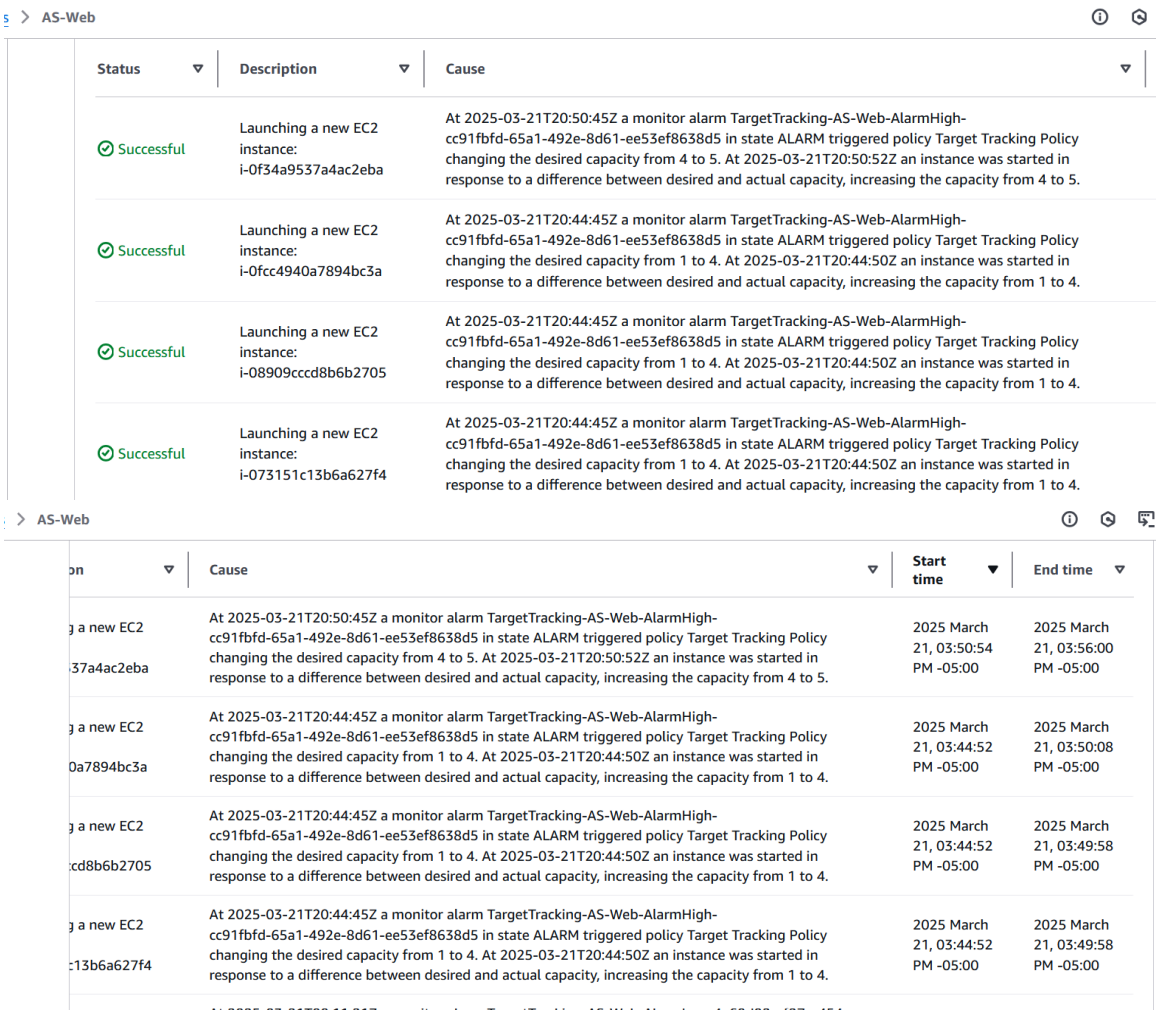


AS-Web

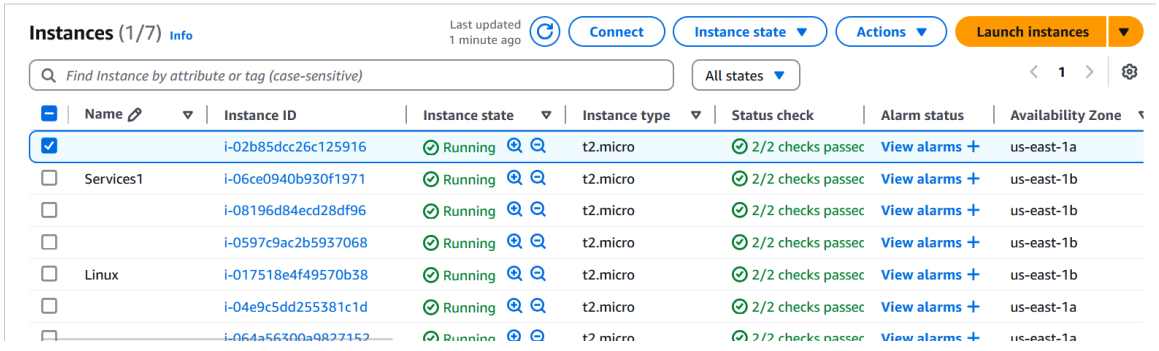
Status	Description	Cause
Waiting for instance warm up	Launching a new EC2 instance: i-0fcc4940a7894bc3a	At 2025-03-21T20:44:45Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-cc91fbfd-65a1-492e-8d61-ee53ef8638d5 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 4. At 2025-03-21T20:44:50Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 4.
Waiting for instance warm up	Launching a new EC2 instance: i-08909cccd8b6b2705	At 2025-03-21T20:44:45Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-cc91fbfd-65a1-492e-8d61-ee53ef8638d5 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 4. At 2025-03-21T20:44:50Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 4.
Waiting for instance warm up	Launching a new EC2 instance: i-073151c13b6a627f4	At 2025-03-21T20:44:45Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-cc91fbfd-65a1-492e-8d61-ee53ef8638d5 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 4. At 2025-03-21T20:44:50Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 4.
Successful	Terminating EC2 instance: i-07408d3c2b1c77ca6	At 2025-03-21T00:11:21Z a monitor alarm TargetTracking-AS-Web-AlarmLow-4e60d02a-f27a-454a-ace2-9b1c0b323707 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 1. At 2025-03-21T00:11:27Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-03-21T00:11:28Z instance i-07408d3c2b1c77ca6 was selected for termination.

AS-Web

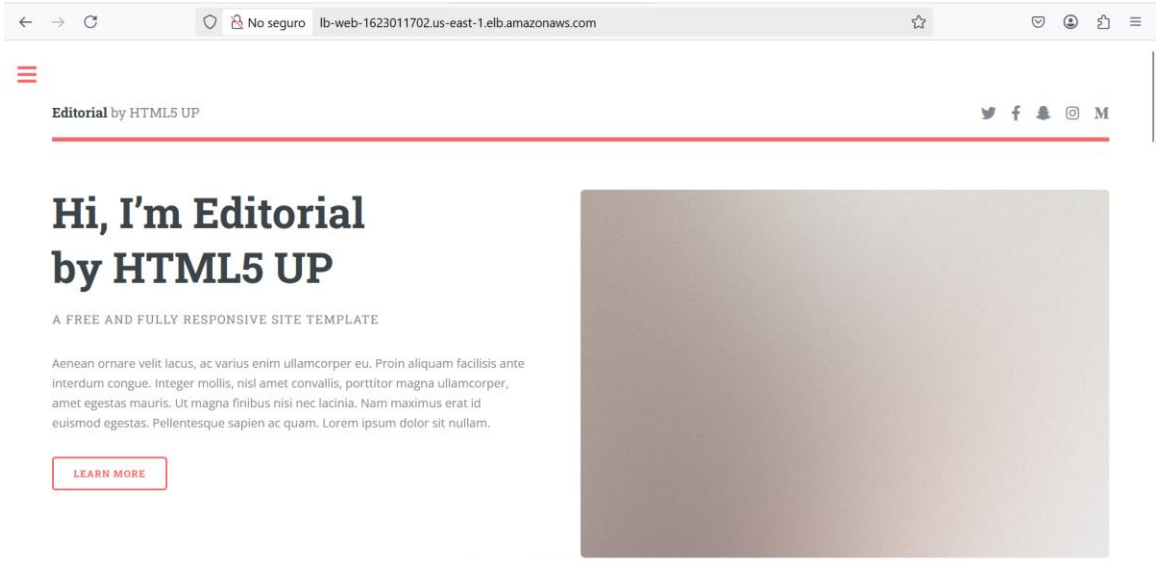
on	Cause	Start time	End time
g a new EC2 0a7894bc3a	At 2025-03-21T20:44:45Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-cc91fbfd-65a1-492e-8d61-ee53ef8638d5 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 4. At 2025-03-21T20:44:50Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 4.	2025 March 21, 03:44:52 PM -05:00	
g a new EC2 cd8b6b2705	At 2025-03-21T20:44:45Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-cc91fbfd-65a1-492e-8d61-ee53ef8638d5 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 4. At 2025-03-21T20:44:50Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 4.	2025 March 21, 03:44:52 PM -05:00	
g a new EC2 :13b6a627f4	At 2025-03-21T20:44:45Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-cc91fbfd-65a1-492e-8d61-ee53ef8638d5 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 4. At 2025-03-21T20:44:50Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 4.	2025 March 21, 03:44:52 PM -05:00	
ng EC2 3c2b1c77ca6	At 2025-03-21T00:11:21Z a monitor alarm TargetTracking-AS-Web-AlarmLow-4e60d02a-f27a-454a-ace2-9b1c0b323707 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 1. At 2025-03-21T00:11:27Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-03-21T00:11:28Z instance i-07408d3c2b1c77ca6 was selected for termination.	2025 March 20, 07:11:28 PM -05:00	2025 March 20, 07:17:10 PM -05:00



En las instancias podemos ver cómo se van inicializando y eliminando.

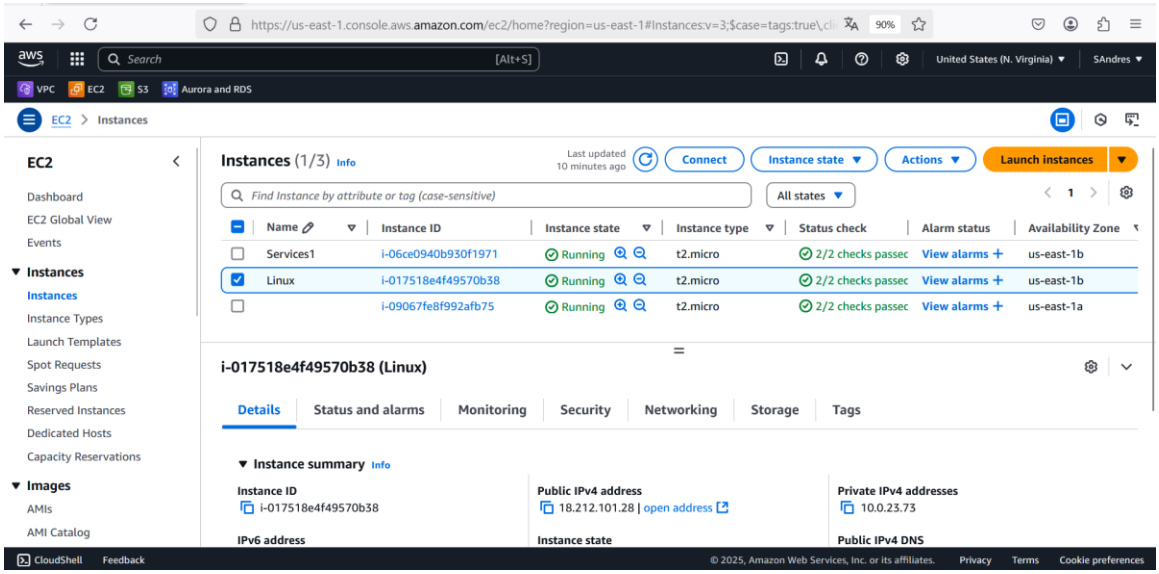


Y si copiamos la dirección en el navegador se debe mantener siempre la misma dirección si se refresca.

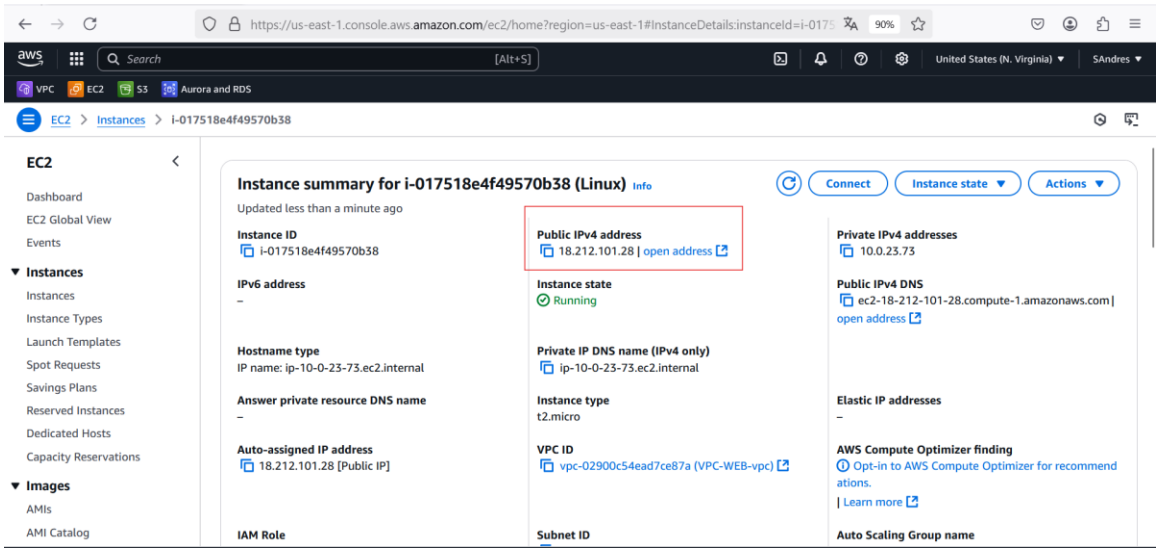


4. Investigar cómo hacer, que con los nombres de un dominio de forma local este se envié a un contenedor diferente.

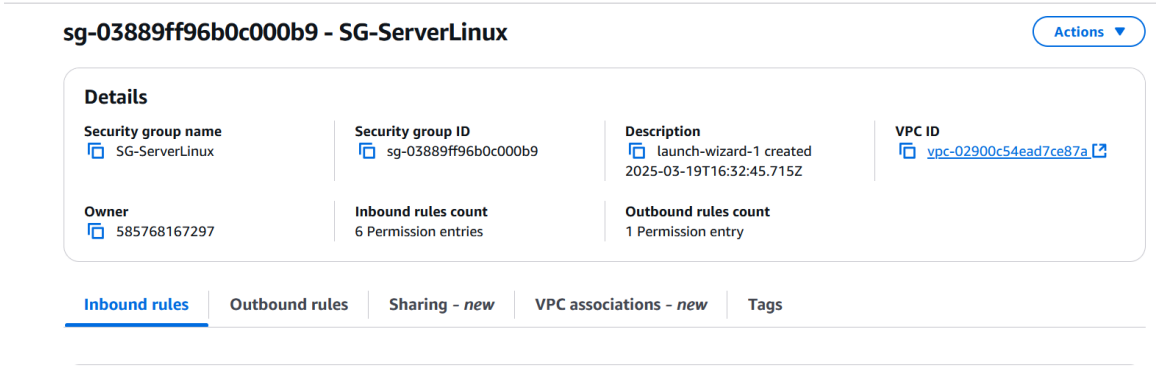
En este ejercicio se va mostrar pantallazos de lo que se hizo para poder mostrar los dominios de forma local y que se conecten a un contenedor diferente.
En nuestro caso escogemos la instancia con el nombre Linux que ya teníamos creada, esta nos sirve para ejecutar servidores web, bases de datos y aplicaciones en la nube, lo usamos también con el servidor Nginx, todo esto como hospedaje de aplicaciones, también están configuradas las VPN y redes privadas.



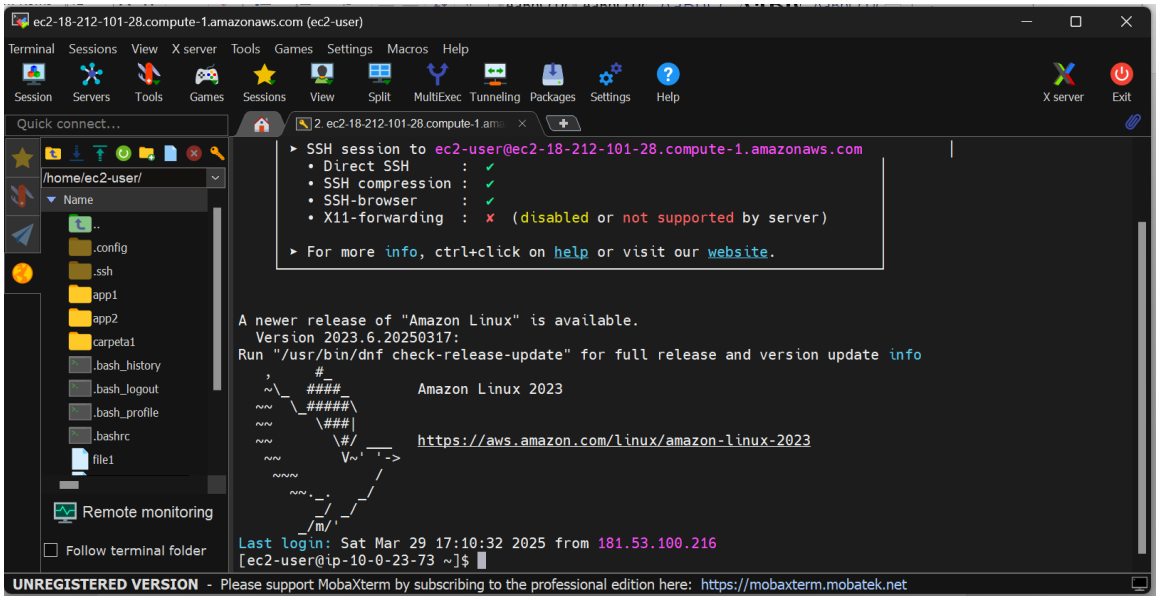
En la siguiente imagen tenemos la IPv4 Publica 18.212.101.28 que vamos a configurar con cada nombre de dominio.



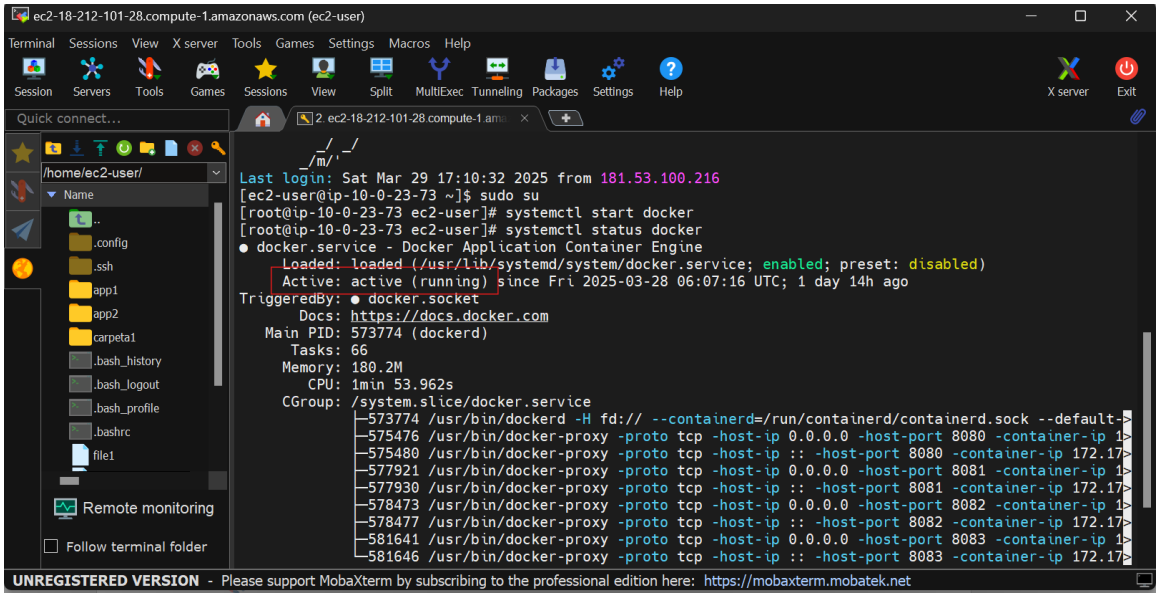
En el security group que tenemos, con el cuál controla el tráfico de red de entrada y salida de la instancia y de los contenedores.



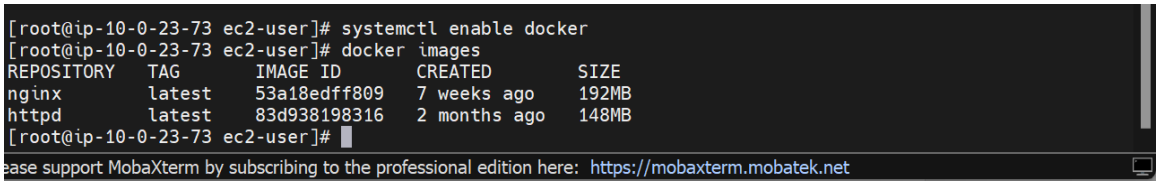
Se instala Docker el cual nos va a servir para crear los contenedores para el desarrollo de la actividad, se usa MobaXterm como terminar de Linux para poder instalar Docker, la cual iniciamos con Session, SSH, Remote host (18.212.101.28), specify username (ec2-user), advanced SSH settings, use private key(Linux.pem) y activamos el terminar.



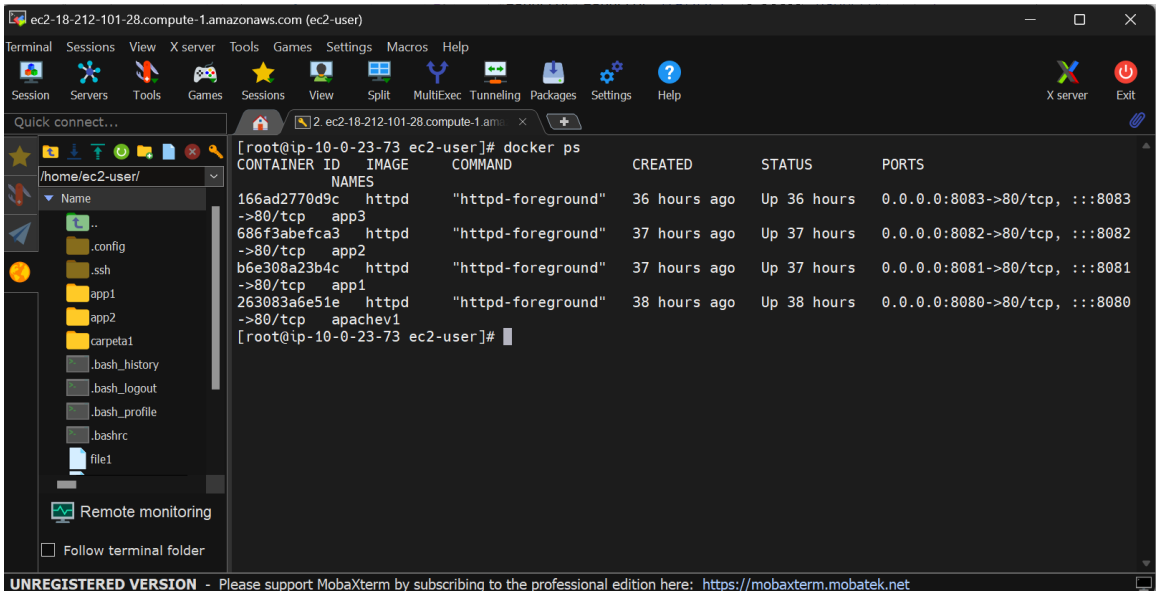
Se desinstala Apache, para poder trabajar con Docker y Nginx.
Como ya lo tenemos instalado Docker se verifica que “Active” se encuentre activado.



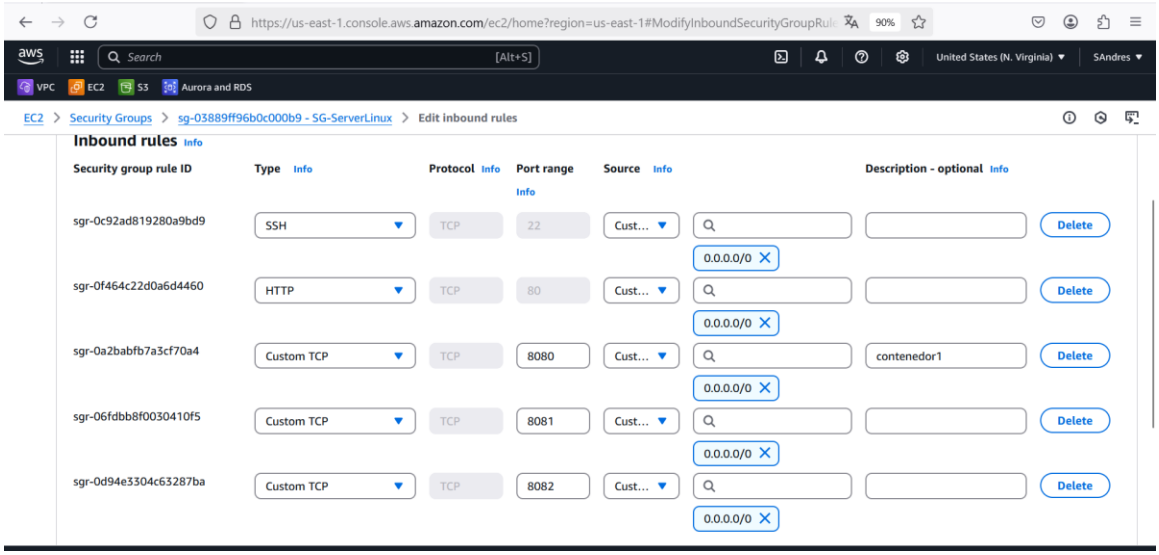
Las imágenes que se han montado en Docker se pueden observar en el IMAGE ID



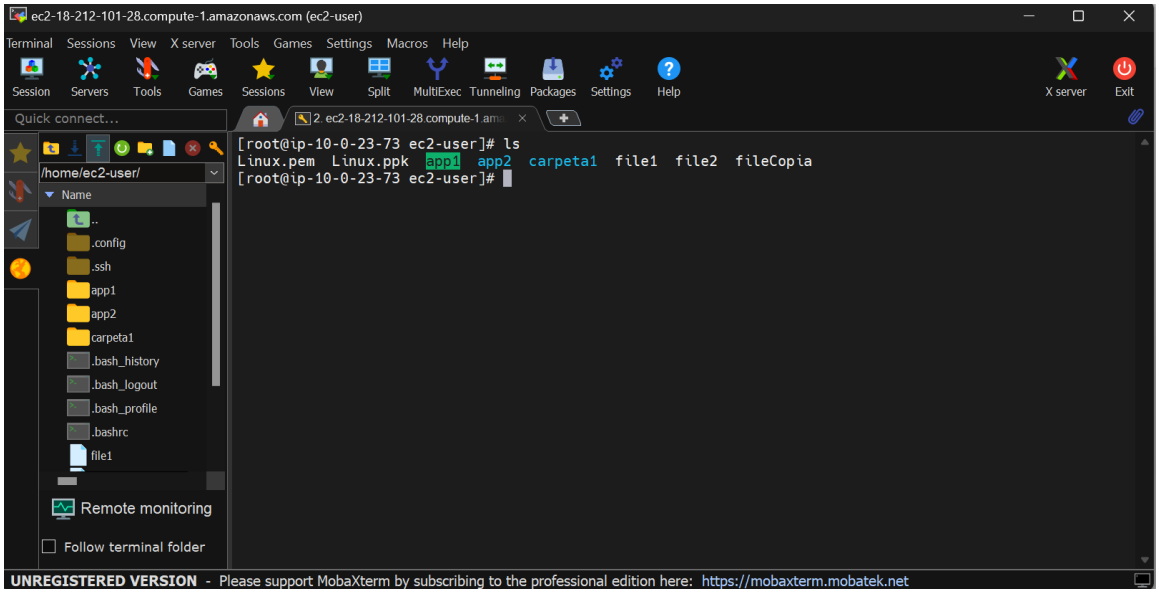
Para ver los contenedores creados ingresamos con la entrada docker ps. Podemos ver el ID del contenedor, los puertos y los nombres.



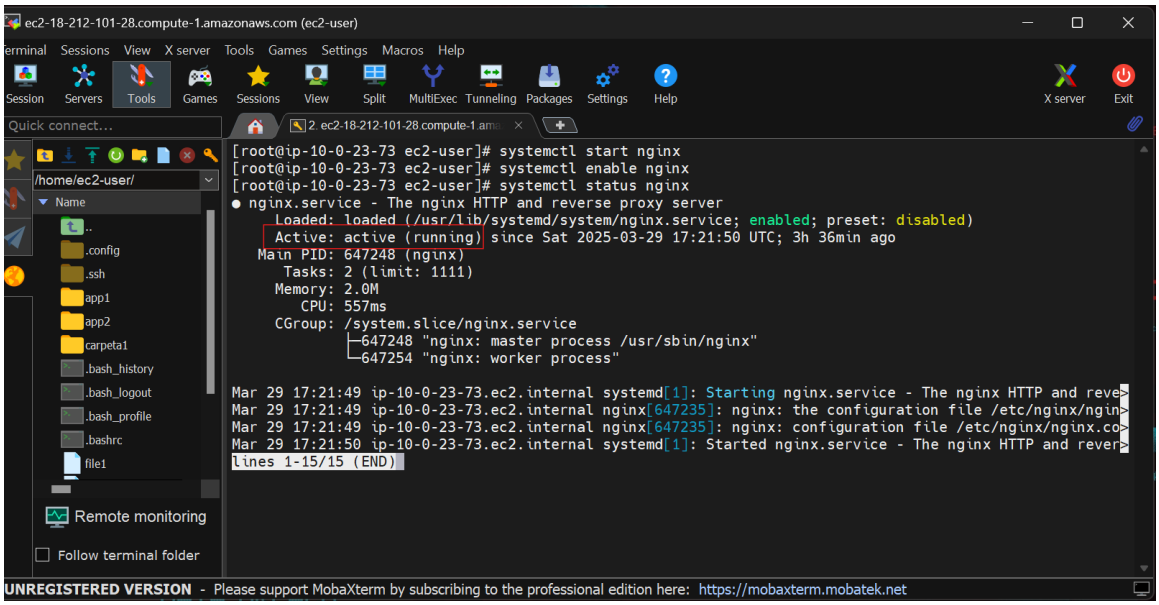
Los puertos se editan en el Inbound rules en AWS.



Se crean las carpetas de cada app. Usamos las entradas correspondientes para la creación como mkdir app1, nos ingresamos al directorio con cd app1/ y cargamos el contenido, en este caso un index.html (con la palabra “Hola a todos este es el contenedor 1”) el cual se va ver reflejado en la dirección IP del navegador y lo creamos. Creamos otra carpeta con el nombre app2 y el contenedor con Docker run, y después de crear los contenedores en el Inboun rules modificamos los puertos.



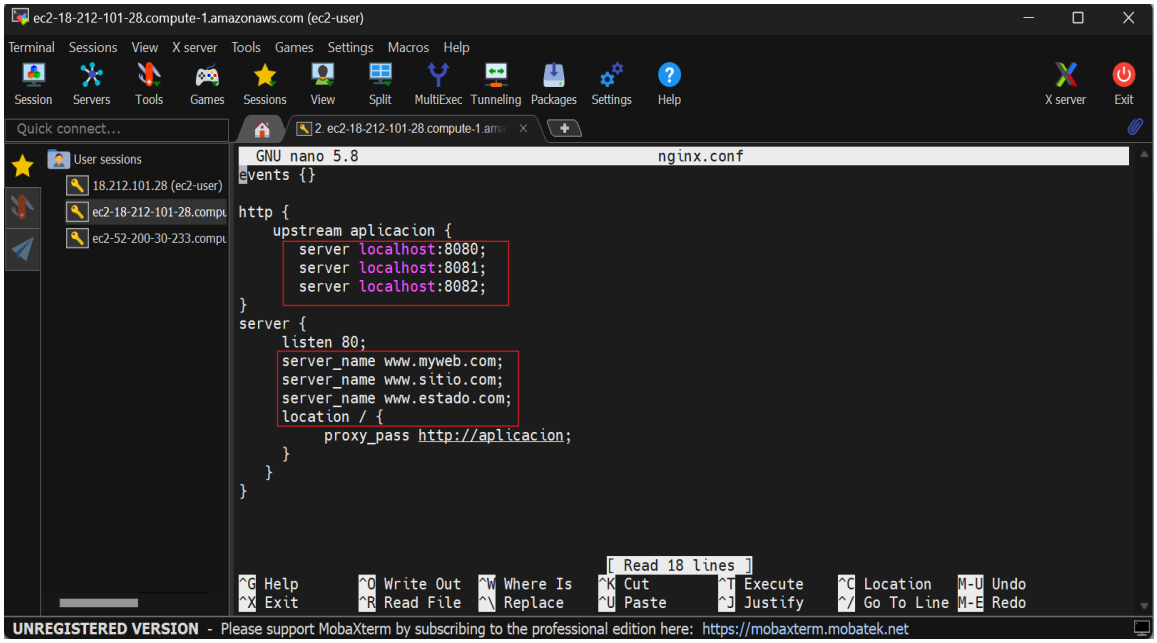
En el siguiente paso se instala Nginx, se verifica que este activo.



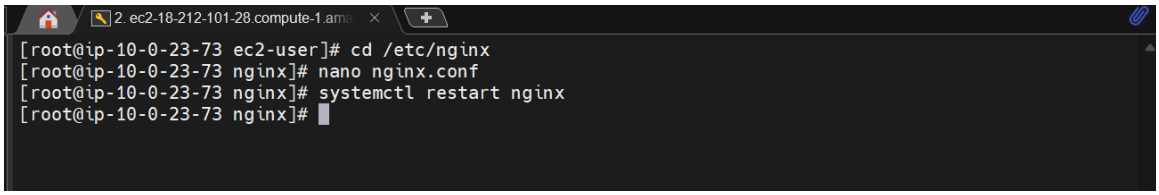
Ingresamos al directorio Nginx con cd /etc/nginx/

Desde ahí podemos modificar un archivo con la entrada nano nginx.conf.

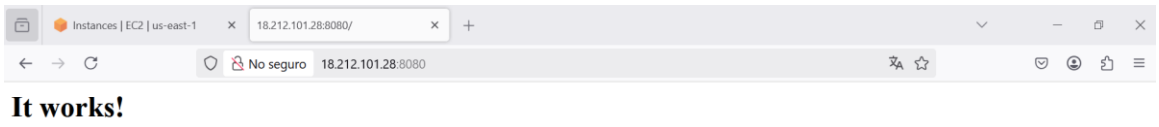
Con esta entrada puedo asegurar que cada contenedor se direccione correctamente al dominio.



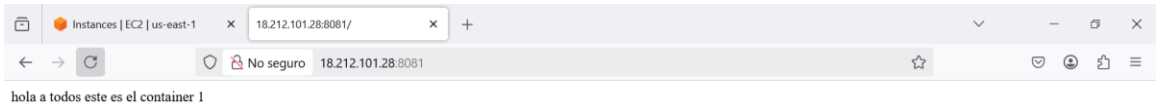
Se modifican los puertos y digitamos en el sistema un systemctl restart nginx. Para volverlo a iniciar con la nueva configuración.



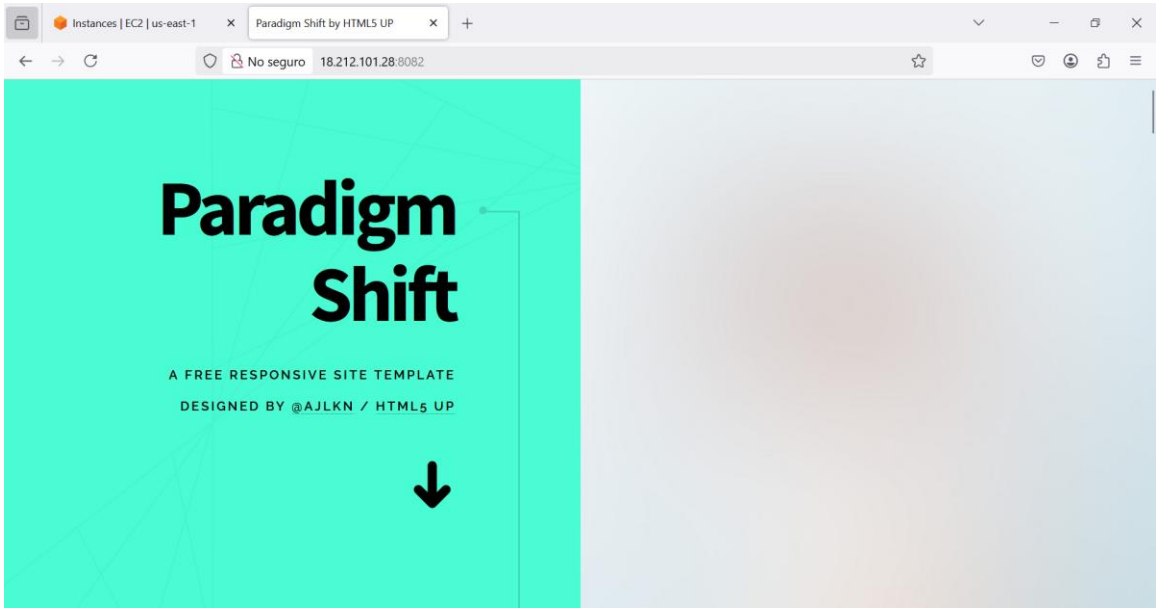
Probando la IPv4 y el puerto: 18.212.101.28:8080



Probando la IPv4 y el puerto: 18.212.101.28:8081

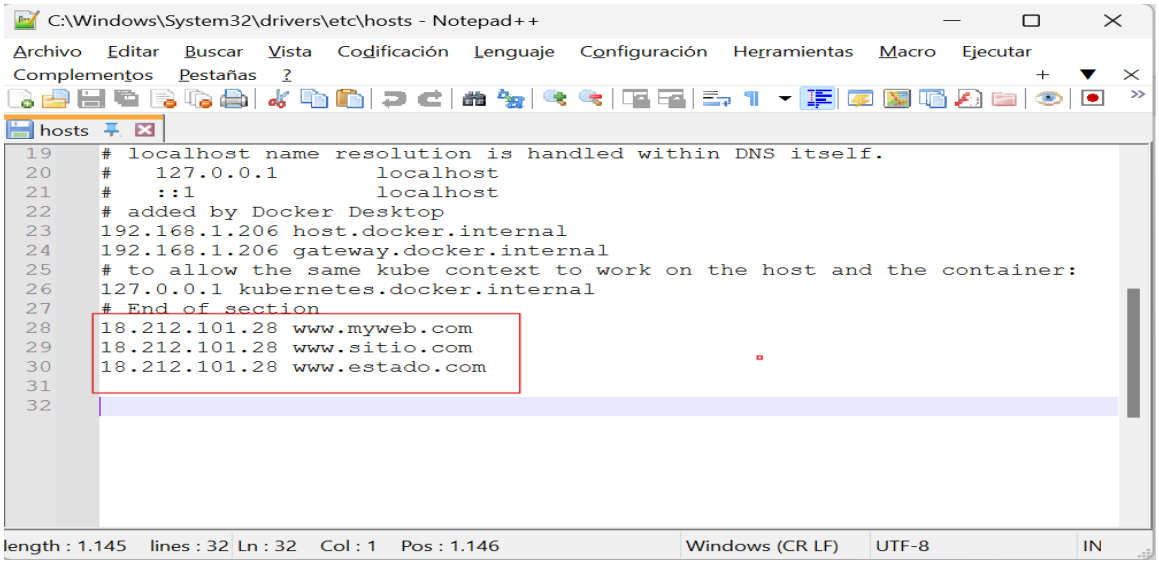


Probando la IPv4 y el puerto: 18.212.101.28:8082



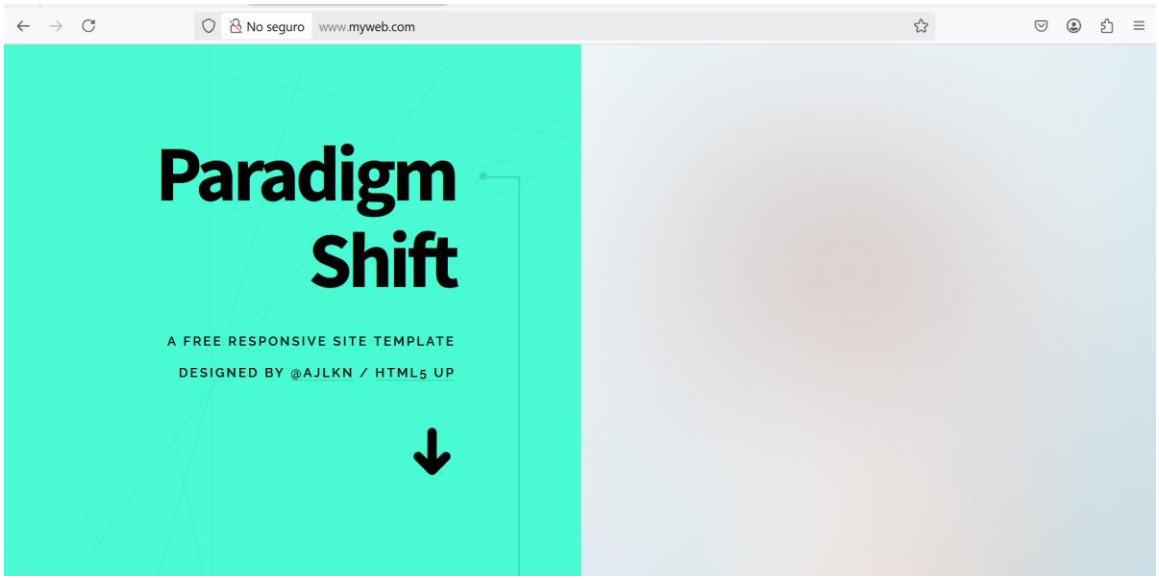
Ahora para poder hacer que el dominio tenga un nombre y que este de forma local apunte al nombre del dominio es decir si en la URL coloco www.myweb.com esta me abra la imagen o el sitio de este contenedor, para hacer esto lo que se modifico fue el archivo interno de Windows llamado host, nos dirigimos a la carpeta de: Windows\System32\drivers\etc\hosts

Sobre este archivo lo editamos en nuestro caso con Notepad ++ debe decir added by Docker Desktop para poder configurar las direcciones IP al final.



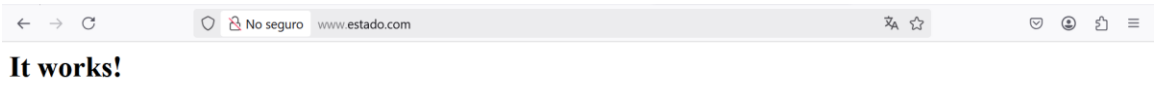
Se debe guardar e iniciar como administrador, se recomienda borrar la cache del sistema antes de probar cada dirección.

Probando el dominio www.myweb.com



Probando el dominio www.sitio.com

Probando el dominio www.estado.com



It works!

Conclusiones

A lo largo del seminario, el servicio más relevante que fue cubierto con mayor amplitud fue Amazon EC2, que permitía el despliegue y la gestión de máquinas virtuales en la nube. Esta herramienta fue fundamental para adquirir conocimientos sobre cómo elegir sistemas operativos, tipos de instancia, configuraciones de almacenamiento y gestionar el acceso remoto.

Los equilibradores de carga para distribuir el tráfico entre instancias EC2 fueron otra parte importante de la solución implementada. Esto no solo aumentó el rendimiento de la aplicación, sino también su disponibilidad, evitando cuellos de botella y fallos por sobrecarga. Su configuración demostró cómo, incluso en una situación de alta carga, se puede asegurar la estabilidad.

Ayuda a que el número de instancias se añada o elimine según el usuario o el uso, y también trabajamos con grupos de autoescalado para eso. Esto nos mostró cómo podríamos ahorrar en recursos y mantener el rendimiento sin la necesidad de intervención manual. Fue una gran experiencia para nosotros conocer y entender de manera efectiva cómo funciona la elasticidad en la nube y por qué es tan esencial para que una aplicación responda correctamente todo el tiempo.

Los archivos y datos se mantuvieron utilizando AWS S3 como un servicio de almacenamiento en la nube. Este servicio era perfecto para copias de seguridad y el servicio de sitios estáticos, e incluso la distribución de contenido en general, además de permitir el control de permisos de la información.

Finalmente, la computación en la nube es una forma mucho más práctica y moderna de trabajar con tecnología. Soluciones utilizando AWS, que pueden ser soluciones sin servidores, pero también permitirán construir soluciones robustas que crecen sin la necesidad de construir servidores físicos. Aprovechar esta gran oportunidad de asistir y participar en este seminario nos permitió aprender mejor cómo implementar estos servicios en la vida real y, junto con ello, nos proporcionó todo el conocimiento inicial necesario para ganar una base más sólida para continuar y practicar estas habilidades en el futuro profesional.

Referencias

Fernández Morales, M. (2012). Computación en la nube para automatizar unidades de información. Revista Bibliotecas. Vol. 30, No. 1, 2012: (ed.). Red Universidad Nacional de Costa Rica. https://elibro.net/es/lc/remington/titulos/23422.

Goldberg, R. P. (1974). Survey of Virtual Machine Research. IEEE Computer. https://ieeexplore.ieee.org/abstract/document/6323581

Hernandez, N. Florez, A. (2014). Computación en la nube. Revista mundo FESC. Universidad de Pamplona. file:///C:/Users/seran/Downloads/Dialnet-ComputacionEnLaNube-5109245.pdf

Google Cloud. (s.f.). Google Cloud Documentation. Recuperado de https://cloud.google.com/docs?hl=es-419

Microsoft Azure. (s.f.). Azure Documentation. Recuperado de https://learn.microsoft.com/en-us/azure/?product=popular

Amazon Web Services. (s.f.). AWS Documentation. Recuperado de https://docs.aws.amazon.com/

Docker. (2025). Docker hub. Recuperado de https://hub.docker.com/

Nginx. (2025). F5 NGINX. Recuperado de https://www.f5.com/go/product/welcome-to-nginx