TRABAJO DE GRADO
Opción Seminario-Diplomado.

La capa gratuita de AWS y algunos componentes útiles.

Corporación Universitaria Remington.
Facultad de Ingeniería
Ingeniería de sistemas

Jonatan Emilio Ramírez Alzate.
Nombre del Tutor del trabajo de grado (Juan Pablo Berrio López).
Opción de Trabajo de grado Seminario-Diplomado.
2025.

Dedicatoria

Mi dedicatoria será para el amor de mi vida, mi hija Luciana.

Agradecimientos

Hay tantas gracias que dar, iniciando con Dios.
Agradecer a mi familia (madre, padre y hermanas) a mi esposa Paola, a la familia de mi esposa que asimismo es mi familia, a incontables amigos y compañeros, y siempre al alma de mi vida que es mi hija amada. Mil y mil gracias a todos.

Tabla de Contenidos

Resumen

Para iniciar se dirá que el trabajo trata de Amazon Web Service y algunos de los servicios que puede ofrecer y como configurarlos para mostrar lo útil que puede llegar a ser el manejo de estas herramientas. Se crearán instancias o EC2, una red privada virtual o VPC por sus siglas, un balanceador de carga o Load Balancer y configuraremos un Grupo de escalado automático O Auto Scaling Group. Al igual que se ejecutarán contenedores en Docker sobre instancias en sistemas operativos Linux propios de la nube de AWS, servicios de almacenamiento, servicios de bases de datos como los RDS y otros servicios más que son de mucha ayuda a la hora de administrar recursos en la nube y montar una infraestructura estructurada ya sea por ocio o aprendizaje, para pequeñas, medianas o grandes empresas.

Palabras clave

(AWS, Ec2, Balanceador de carga, VPC, Contenedor)

Marco conceptual y contextual

1. Introducción a Amazon Web Services (AWS)

Considerado el primer proveedor de servicios en la nube iniciando en el 2002, AWS se estableció como el mayor participante de este negocio por dar los primeros pasos, seguido por Microsoft AZURE y Google Cloud Platform (GCP).

AWS ofrece casi todo tipo de servicios en la nube para cualquier necesidad, desde una página web muy básica hasta infraestructura que soporte empresas multinacionales.

Las mayores ventajas que ofrece la computación en la nube es su flexibilidad y escalabilidad ya que con solo dar unos cuántos clics podría desplegar toda una infraestructura y servicios para una aplicación bastante robusta, con servidores para las aplicaciones, servidores para las bases de datos, servicios que apoyen la continuidad de los negocios con su redundancia (para el caso una segunda infraestructura) y todo esto en cuestión de muy poco tiempo, haciendo comparación con una construcción de cero de manera local.

2. Servicios clave de AWS.

Este trabajo se estudia algunos de los principales servicios de AWS y configuración. Entre los servicios relevantes se encuentran:

Instancias EC2 2.1.

Esto es la unidad básica de toda infraestructura en la nube para AWS llamada instancia no es más que un servidor o computador virtual con diferentes configuraciones de hardware y software.

Red privada virtual (VPC) 2.2

Este servicio permite crear una red asilada dentro de AWS para mejorar la seguridad y control del tráfico de datos.

Balanceador de carga (Load Balancer) 2.3

Reparte las peticiones o solicitudes entendiendo que esto, es todo aquello que consultamos a las aplicaciones o bases de datos, como un informe, una imagen o un archivo. Entonces el load balancer se encarga de distribuir entre varias o muchas instancias las peticiones para que los servicios no se sobrecarguen en una sola máquina.

Grupo de escalado automático (Auto Scaling Group) 2.4

Como su nombre lo indica es una función programada que automáticamente crea más instancias de acuerdo con las necesidades, entonces, por ejemplo, si tengo 2 máquinas y por alguna razón las aplicaciones tienen más trabajo y se están quedando cortas para responder las peticiones el auto scaling group crea otro servidor casi instantáneamente para ayudar en la operación y en conjunto con el balanceador repartirán la carga para que la continuidad del negocio nunca se vea comprometida.

Docker en AWS 2.5

Esta función que empaqueta toda clase de soluciones es muy útil para el despliegue rápido de aplicaciones y servicios. Esta tecnología mayormente utilizada en ambientes con sistemas operativos Linux. De esta manera un Docker es un contenedor de varios o muchas funciones que necesitarían unos recursos en términos de capacidad de las instancias ya sea memoria RAM o procesamiento CPU menores a los que necesitarían con una instalación y configuración tradicional. También tienen la ventaja que se pueden desplegar mucho más rápido en cualquier momento y lugar.

Servicios de almacenamiento 2.6

Conocido para el entorno de AWS como S3 (Simple Storage Service), este servicio de almacenamiento tiene varias maneras de uso, puede utilizarse como almacenamiento tradicional donde se tiene un disco duro que almacena aplicaciones e información. Pero también tiene un modo de uso por objetos, que quiere decir que si lo utilizo de esta manera una carpeta con muchos archivos o aplicaciones no toma en cuenta su peso total (eso sí, máximo 5GB por objeto) o la cantidad, sino que lo toma como una sola pieza. Puede almacenar objetos que casi no usará con este servicio desde otra de sus funcionalidades S3 Glacier para generar costos menores al del almacenamiento normal.

Bases de datos gestionadas (RDS – Relational Database Service) 2.7

Este servicio puede ayudar a desplegar una base de datos relacional sin la necesidad de instalar o configurar nada para el motor de bases de datos, no se tendría la necesidad de encargarse de la infraestructura del servidor. Este servicio de acuerdo con la necesidad y configuración que se le dé podría ser tan potente como se requiera.

3. Aplicaciones y usos

Todo este tipo de servicios puede ser aplicados en múltiples propósitos, como:

Aprendiza y desarrollo 3.1

Con su capa gratuita se puede tener un entorno de práctica para comprender los principios de la computación en la nube.

Empresas de cualquier tamaño 3.2

Definitivamente cualquier tipo de individuo o empresa puede tener el beneficio de la flexibilidad y la escalabilidad que ofrecen los entornos en la nube. Desde tener una

solo instancia con una página web de muestra hasta tener toda una super aplicación mundial.

Optimización de infraestructura 3.3

Ya se ha dicho que puede automatizar o aumentar todo lo que desea para mejorar la infraestructura requerida tanto como desee o se necesite, pero de igual modo puede reducir todo en su ambiente, como decir que tiene 10 instancias o servidores porque se necesitaba en el momento, pero puede llegar el momento en que no necesite eso sino tan solo, una única instancia y de esta manera reducir costos operativos sin desperdiciar recursos, tiempo ni dinero.

Desarrollo e implementación del aprendizaje

Orígenes de la virtualización y la computación en la nube: línea de tiempo

Introducción:

La virtualización y la computación en la nube han cambiado a lo largo del tiempo, dando paso al desarrollo de las nuevas tecnologías que hoy en día se usan a diario. En la tabla 1 línea de tiempo, veremos hitos relevantes y algunos eventos notables que llevaron a alcanzar el nivel tecnológico de los servicios de la nube actuales.

Tabla 1

Hitos en la historia de la virtualización

Año	Hito	Descripción
Década de 1960	Primeros pasos: virtualización de mainframe	IBM desarrolla la primera tecnología de virtualización en ordenadores mainframe, permitiendo que múltiples usuarios compartan el mismo sistema sin interferencias.
Década de 1970	Sistema IBM VM/370	IBM presenta el sistema operativo VM/370, uno de los primeros ejemplos reales de virtualización para informática empresarial en mainframes.
Década de 1980	Memoria virtual en computadoras personales	El concepto de memoria virtual, que permite al sistema operativo utilizar el espacio del disco duro como RAM "virtual", está ganando popularidad, especialmente en las computadoras personales.
década de 1990	El auge de la virtualización x86	Se comienza a desarrollar software de virtualización para sistemas basados en x86, lo que permite que las computadoras personales ejecuten múltiples instancias del sistema operativo en la misma máquina.
década de 2000	Virtualización de servidores y VMware	VMware crea uno de los primeros productos comerciales de virtualización x86, que permite a los administradores de TI ejecutar múltiples instancias del sistema operativo en un solo servidor. Esto impulsa la adopción de la virtualización en los centros de datos.

2005-2010	Computación en la nube y virtualización	El crecimiento de las plataformas de computación en la nube como Amazon Web Services (AWS) convierte la virtualización en una práctica común para entornos de nube públicos y privados.
década de 2010	Contenerización y Docker	La contenerización (a través de Docker) se convierte en una tendencia importante, ofreciendo una virtualización liviana que es más rápida y eficiente que las máquinas virtuales tradicionales.
década de 2020	Integración de IA y aprendizaje automático con virtualización	Las tecnologías de inteligencia artificial y aprendizaje automático se integran con plataformas de virtualización para mejorar la gestión de recursos, la automatización y la eficiencia en entornos virtuales.
Década de 2020 (en curso)	Computación cuántica y virtualización	La investigación en computación cuántica impacta el futuro de la virtualización, prometiendo capacidades potencialmente transformadoras en la computación.

Nota. Hitos o eventos que marcaron el camino hacia los avances que produjeron al nivel tecnológico presente. Buscar en la sección referencias.

Tabla 2

Comparación nombres y servicios entre proveedores más populares.

Categoría	Descripción	AWS	Azure	GCP
Computación	Servicios que proporcionan máquinas virtuales para ejecutar aplicaciones en la nube.	EC2 (Elastic Compute Cloud)	Virtual Machines (VMs)	Compute Engine
Plataforma como Servicio (PaaS)	Soluciones para desplegar y administrar aplicaciones sin preocuparse por la infraestructura.	Elastic Beanstalk	Azure App Service	App Engine
Contenedores	Servicios para la gestión y orquestación de contenedores Docker y Kubernetes.	ECS (Elastic Container Service) / EKS (Elastic Kubernetes Service)	Azure Kubernetes Service (AKS)	Kubernetes Engine (GKE)

Computación sin Servidor (Serverless)	Ejecuta funciones sin administrar servidores, cobrando solo por el tiempo de ejecución.	Lambda	Functions	Cloud Functions
Almacenamiento de Archivos	Servicios para almacenar y recuperar objetos, como imágenes, videos o backups.	S3 (Simple Storage Service)	Azure Blob Storage	Cloud Storage
Bases de Datos Relacionales	Bases de datos administradas como MySQL, PostgreSQL y SQL Server.	RDS (Relational Database Service)	Azure SQL Database	Cloud SQL
Bases de Datos NoSQL	Bases de datos escalables para documentos, claves-valor y tiempo real.	DynamoDB	Cosmos DB	Firestore / Bigtable
Redes y CDN	Distribución de contenido globalmente y redes privadas en la nube.	CloudFront (CDN)	Azure CDN	Cloud CDN
Balanceo de Carga	Distribuye tráfico entre instancias para alta disponibilidad y rendimiento.	Elastic Load Balancer (ELB)	Azure Load Balancer	Cloud Load Balancing
Seguridad y Gestión de Identidad	Control de accesos, autenticación y permisos en la nube.	IAM (Identity and Access Management)	Azure Active Directory (AAD)	IAM
Análisis de Datos y Big Data	Plataformas para procesar grandes volúmenes de datos.	Redshift	Azure Synapse Analytics	BigQuery
Mensajería y Colas	Comunicación asíncrona entre servicios con colas y eventos.	SQS (Simple Queue Service) / SNS (Simple Notification Service)	Azure Service Bus / Event Grid	Pub/Sub

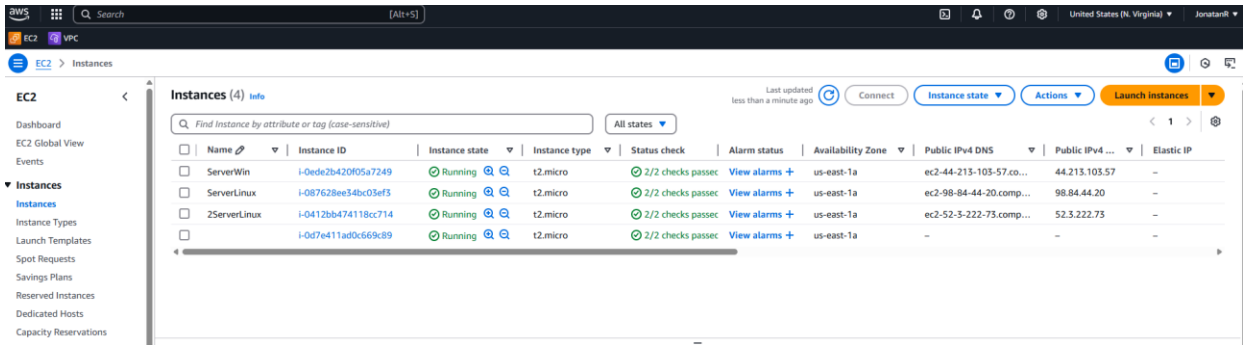
Nota. * Información obtenida de las páginas oficiales de AWS, AZURE y Google Cloud.

Buscar en la sección referencias.

PRÁCTICA

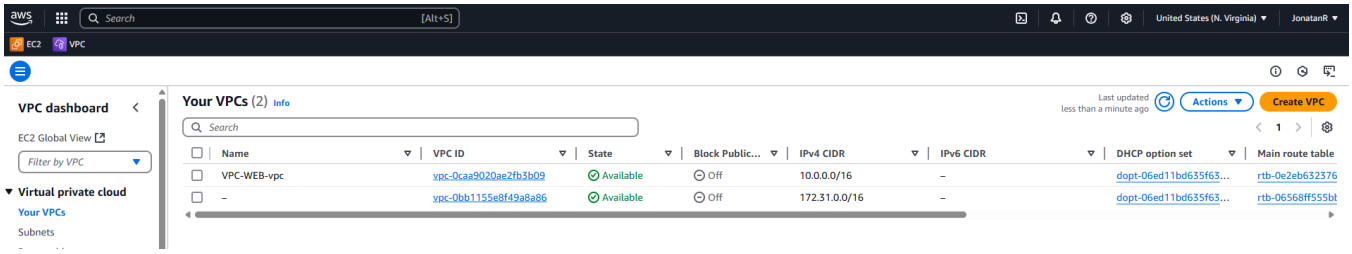
Instancias creadas

Se crean instancias para realizar la práctica mediante el menú de Launch instances.



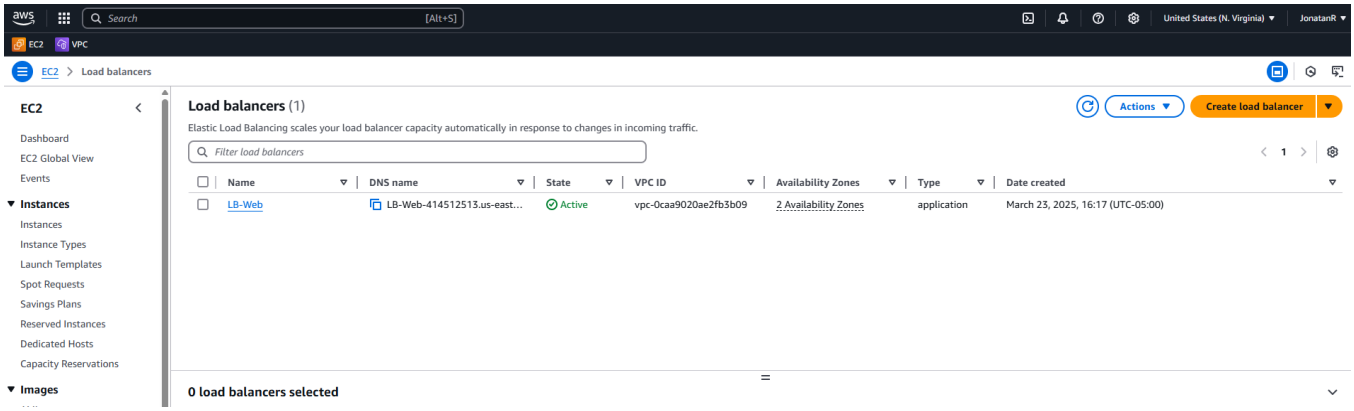
VPC

Función que crea una red virtual privada para la práctica



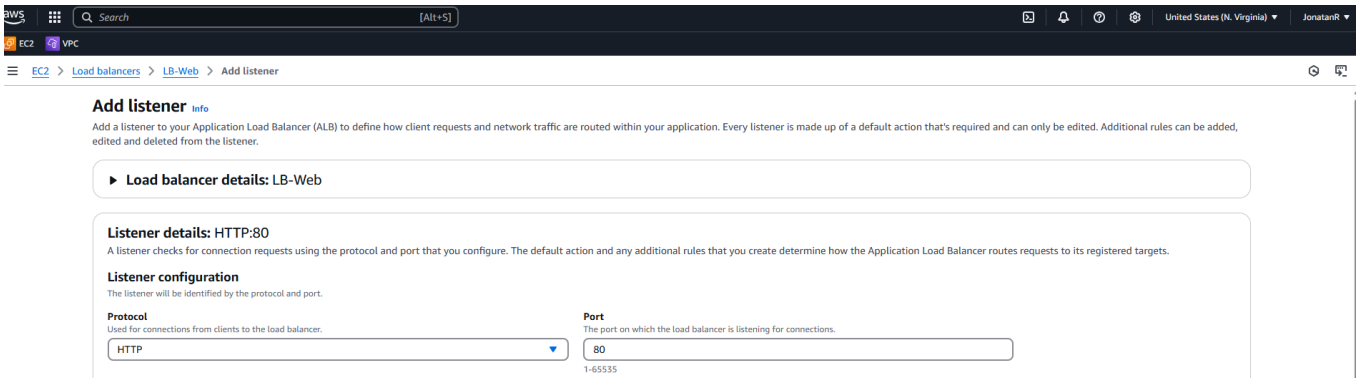
Load balancer

Se crea un Balanceador de carga de aplicaciones para que las instancias se les reparta la carga equitativamente de acuerdo con la cantidad de peticiones.



Listener

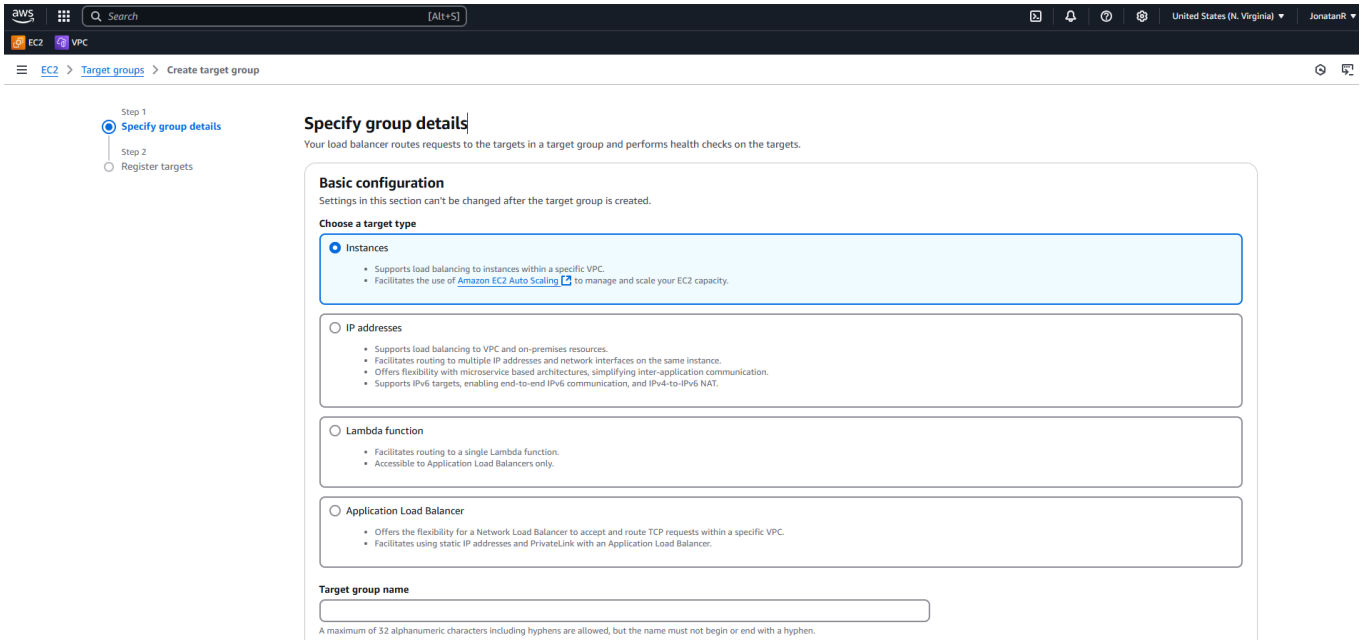
Al cual le modificamos el Listener para que reciba el tráfico por el puerto 80 y que reciba las peticiones provenientes de internet y para el caso de la correspondiente prueba

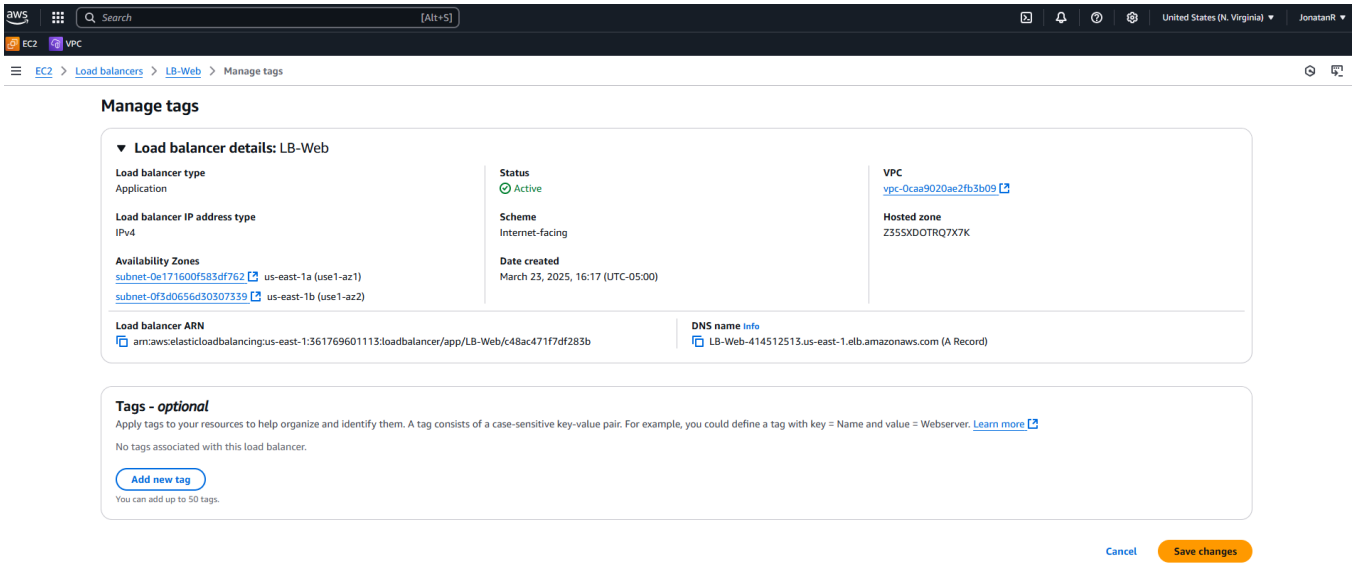


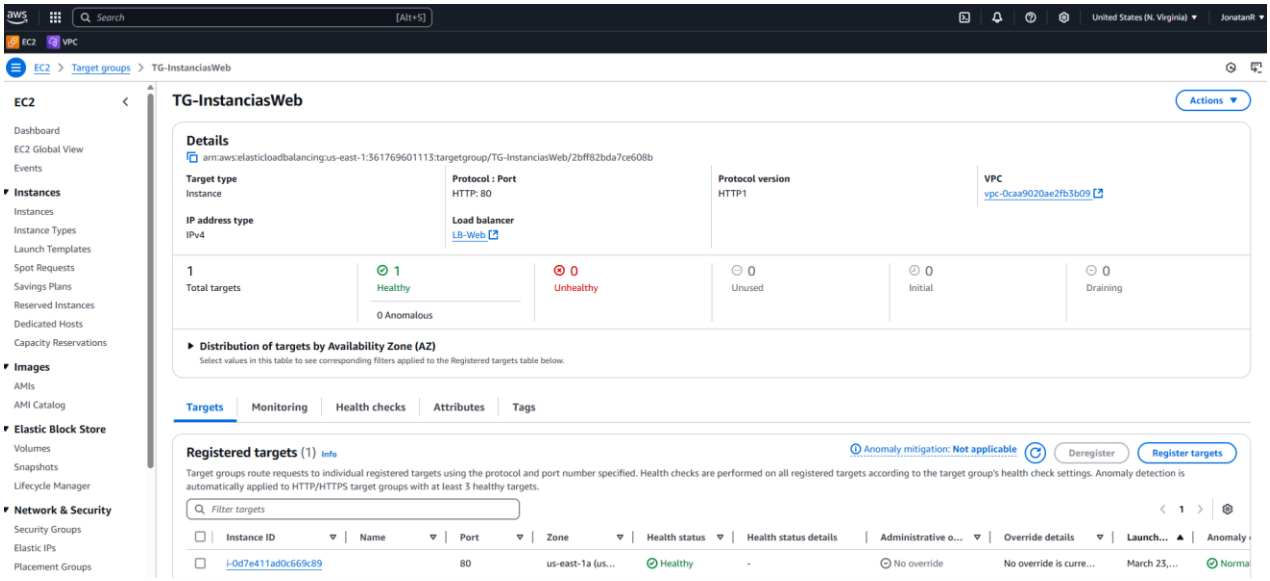
Target group

Se configura el target group por instancia y debe quedar en el mismo VPC que siempre se ha estado utilizando para que tenga comunicación con las instancias.

HEALTH CHECKS son los tiempos en los cuales el balanceador va a estar revisando que las instancias están disponibles para enviar las peticiones

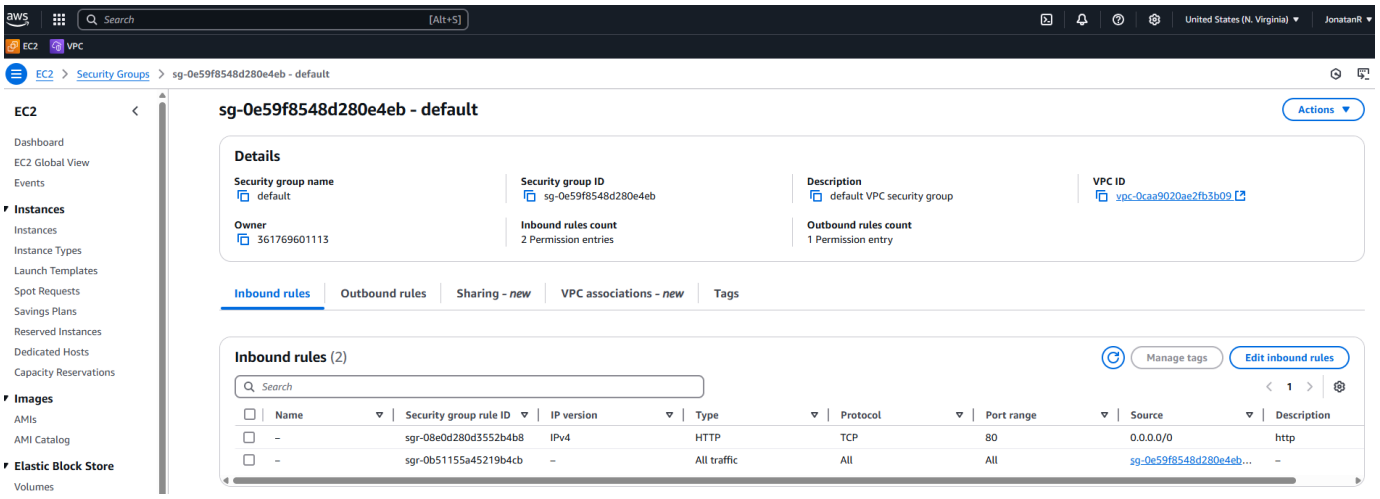

aws
Search
[Alt+S]
United States (N. Virginia)
JonatanR
EC2
VPC
EC2 > Target groups > Create target group
Step 1
Specify group details
Step 2
Register targets
Specify group details
Your load balancer routes requests to the targets in a target group and performs health checks on the targets.
Basic configuration
Settings in this section can't be changed after the target group is created.
Choose a target type
Instances
Supports load balancing to instances within a specific VPC.
Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.
IP addresses
Supports load balancing to VPC and on-premises resources.
Facilitates routing to multiple IP addresses and network interfaces on the same instance.
Offers flexibility with microservice based architectures, simplifying inter-application communication.
Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
Lambda function
Facilitates routing to a single Lambda function.
Accessible to Application Load Balancers only.
Application Load Balancer
Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.
Target group name
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.


aws
Search
[Alt+S]
United States (N. Virginia)
JonatanR
EC2
VPC
EC2 > Load balancers > LB-Web > Manage tags
Manage tags
Load balancer details: LB-Web
Load balancer type
Application
Load balancer IP address type
IPv4
Availability Zones
subnet-0e171600f583df762 us-east-1a (use1-az1)
subnet-0f3d0656d30307339 us-east-1b (use1-az2)
Status
Active
Scheme
Internet-facing
Date created
March 23, 2025, 16:17 (UTC-05:00)
VPC
vpc-0caa9020ae2fb3b09
Hosted zone
Z35SXDOTRQ7X7K
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:361769601113:loadbalancer/app/LB-Web/c48ac471f7df283b
DNS name Info
LB-Web-414512513.us-east-1.elb.amazonaws.com (A Record)
Tags - optional
Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more
No tags associated with this load balancer.
Add new tag
You can add up to 50 tags.
Cancel
Save changes



Security group de balanceador

Configurado para permitir que tenga habilitado el tráfico por el puerto 80



Auto scaling

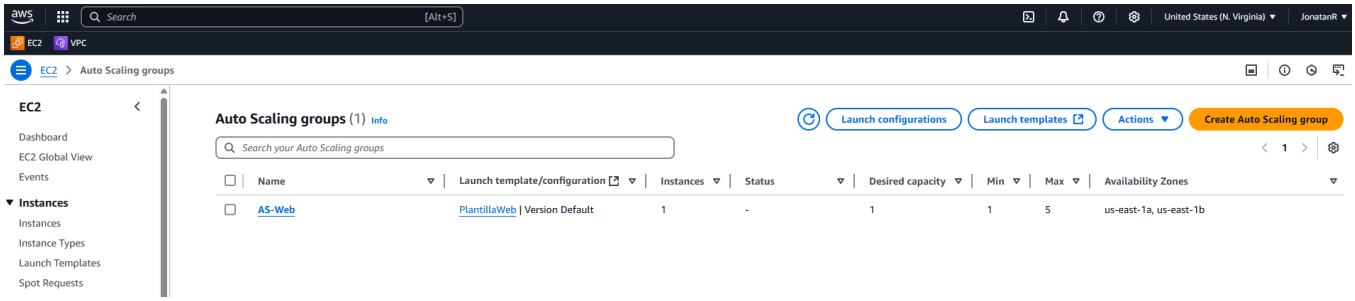
Lanza instancias automáticas con el performance ya predefinido con el launch Template y se le define una plantilla la cual puede tener varias versiones de acuerdo con cambios que se le hagan con configuraciones diferentes de recursos.

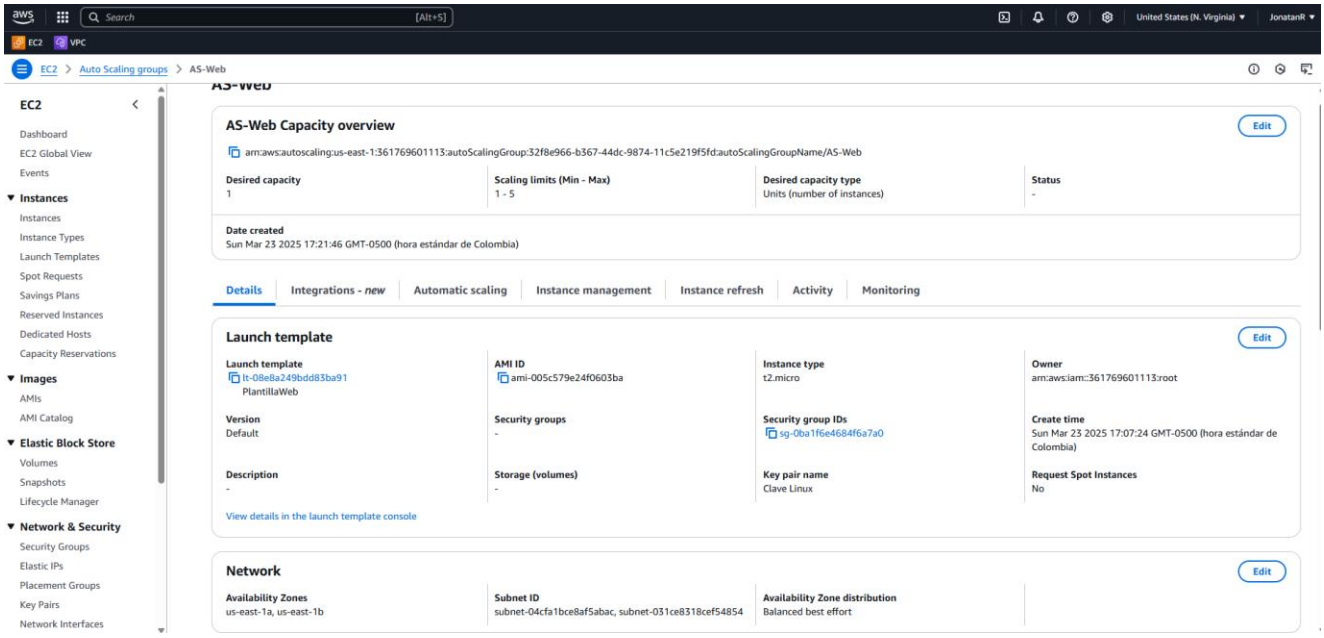
En la configuración de red se deja en la subred privada por buenas prácticas de seguridad y diseño y con la estrategia para repartir las nuevas instancias eligiendo entre datacenters por zona de distribución al balanceo con el mejor esfuerzo.

Se debe asociar el launch template al balanceador que se creó mediante el target group (TG-InstanciasWeb).

El monitoreo de salud lo chekeamos para que realice el monitoreo de la CPU para este caso, que se actualiza cada 5 min porque se encuentra en la capa gratuita y reparte la carga según el promedio entre las instancias activas y dependiendo de la carga crea o termina otras instancias.

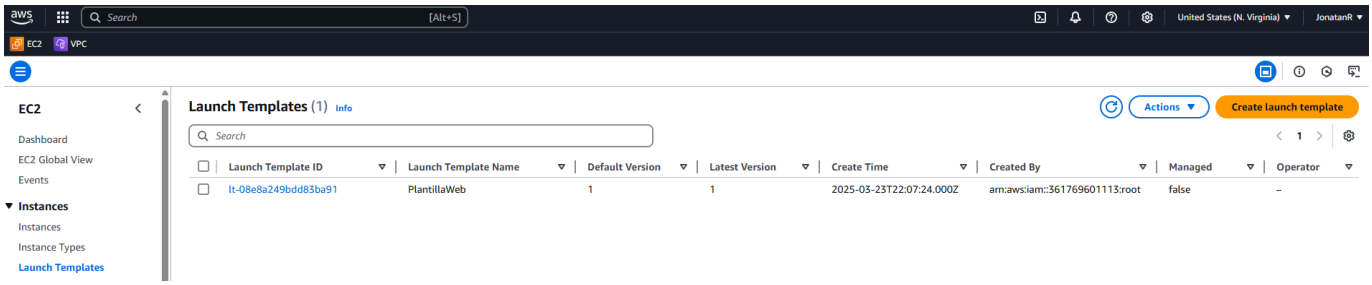
Se configura el valor deseado, valor mínimo y valor máximo de las instancias que lanzaría el auto scaling.

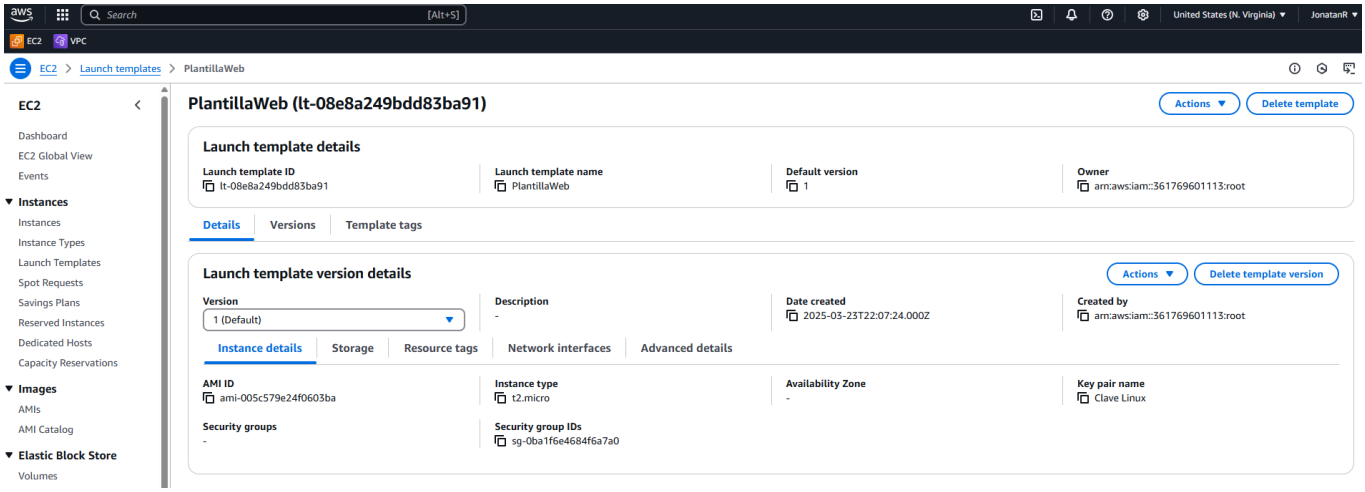




Plantilla WEB

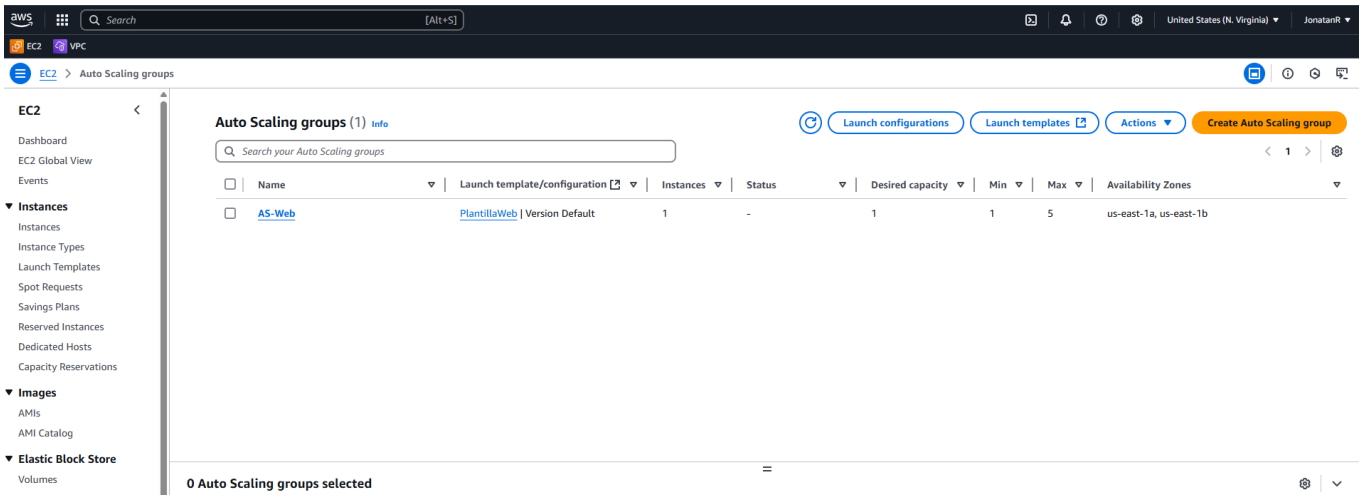
Se Configura la plantilla de instancias para como requisito previo a la configuración del launch template.





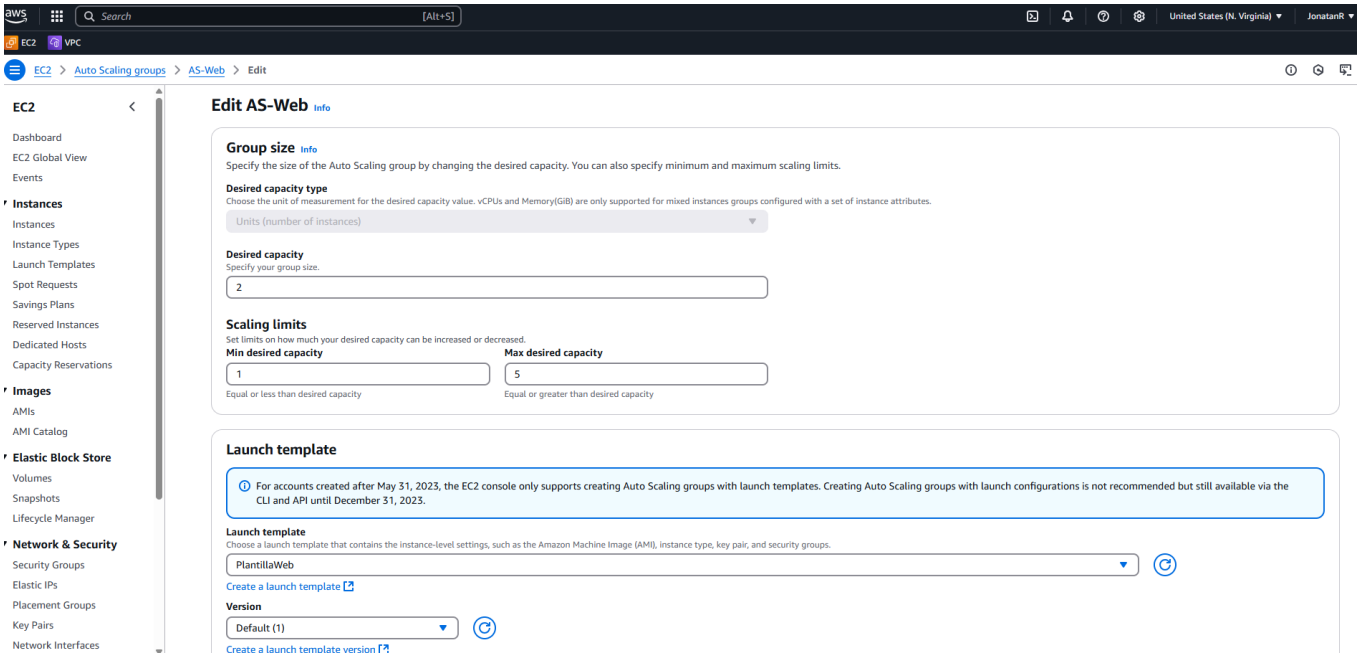
Launch template

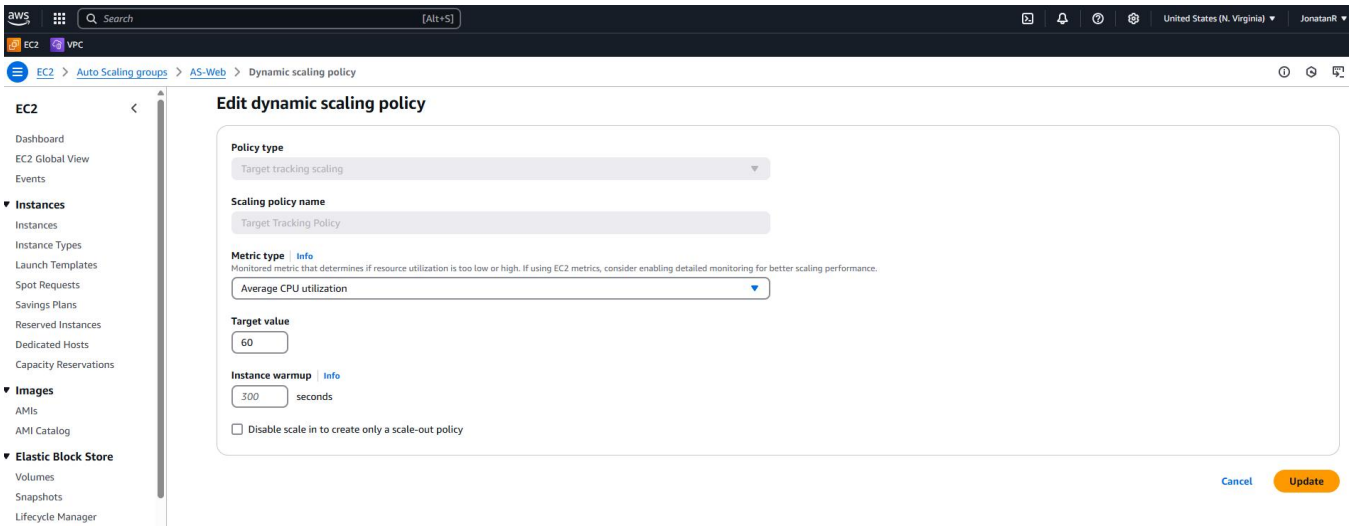
Se asocia la plantilla anterior al launch template para configurar el auto scaling



Configuración de la política dinámica

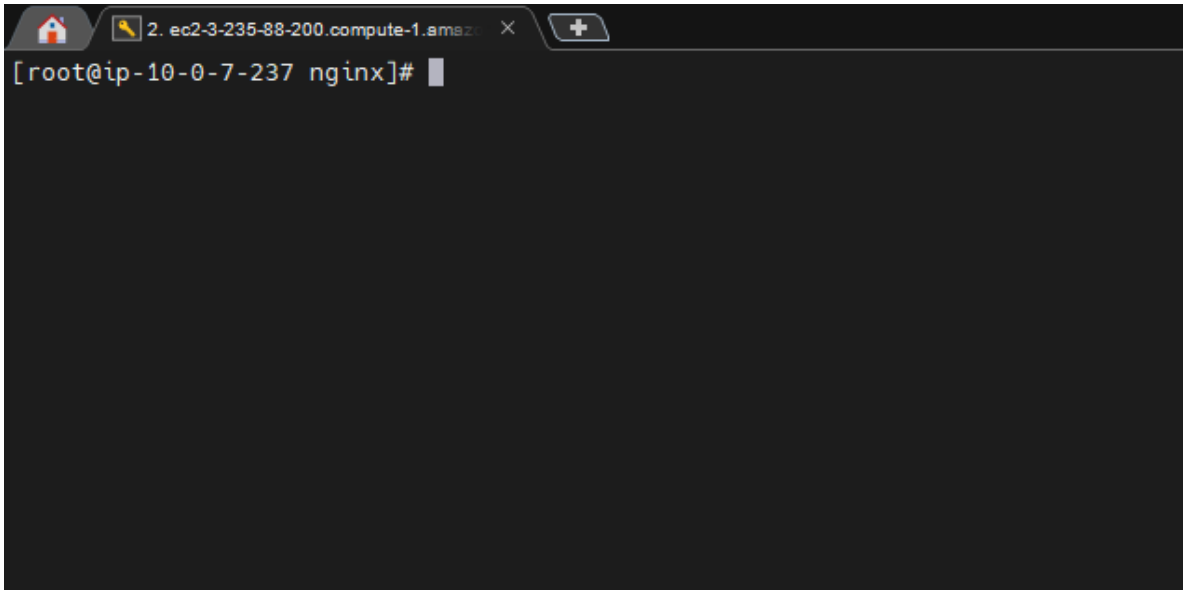
Se configura para tener definida la capacidad mínima o máxima de instancia lanzadas por el auto scaling con la configuración de la plantilla contenida en el launch template.



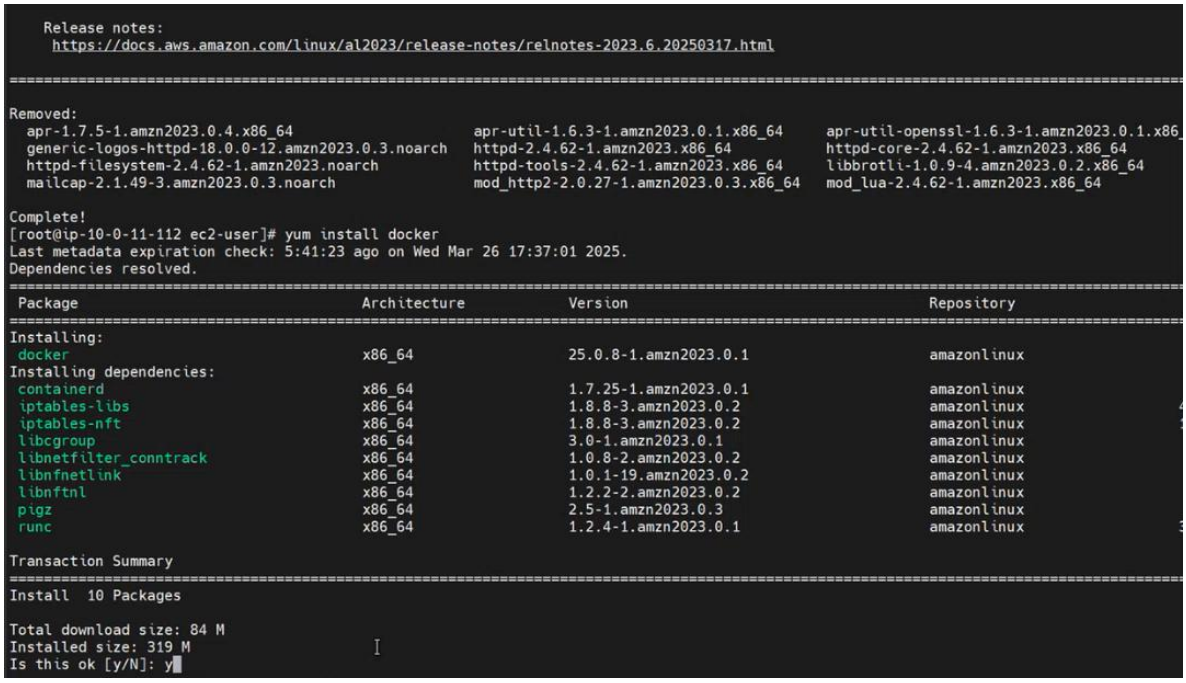


Creación de contenedores

Se ingresa a la instancia Linux con la llave a través de conexión SSH y se pasa al usuario a root con el comando sudo su.

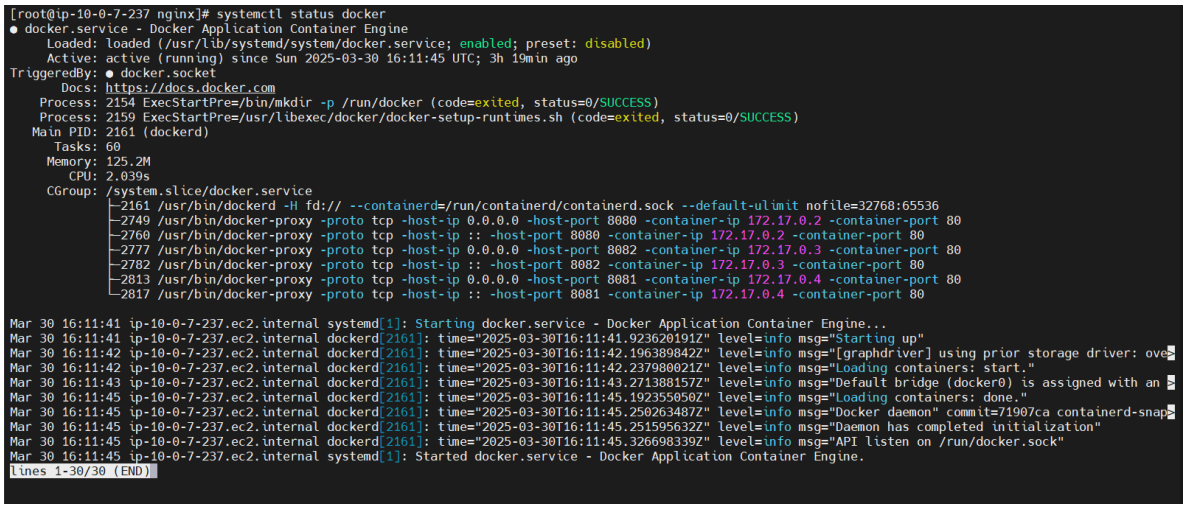


Se instala docker con el comando “yum install docker” y se escribe Yes



Se valida que el servicio docker ya se está ejecutando con el comando systemctl status docker y se inicializa el servicio con el comando systemctl start docker y se deja

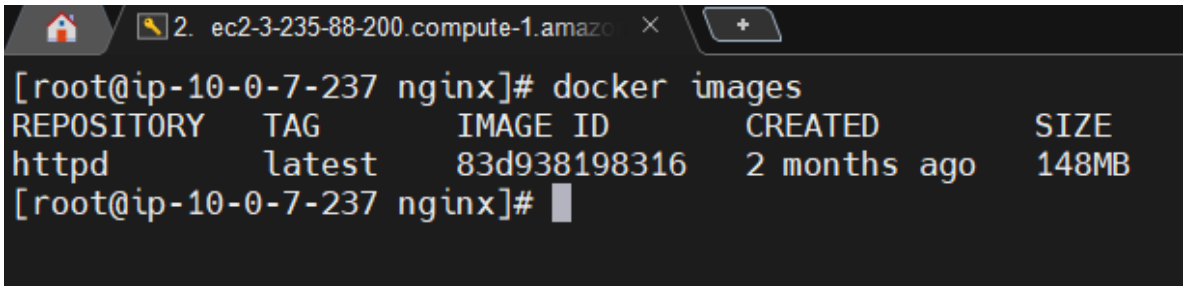
para que arranque automáticamente con systemctl enable docker cuando inicie la instancia.



Descarga de imagen

Se descarga la imagen que se utilizará para los dockers por lo que se va a la página https://hub.docker.com/.

Se descarga la imagen con docker pull httpd y se confirma cual fue la imagen que se descargó con el comando docker images

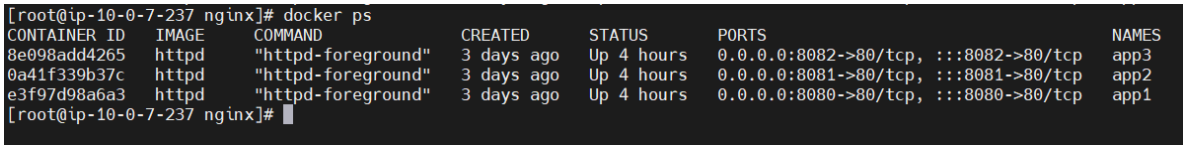


Se crean los contenedores a partir de la imagen que se descargó con el comando "docker run -dit --name app1 -d --restart always -p 8080:80 -v /home/ec2-user/app1/:/usr/local/apache2/htdocs/ httpd"

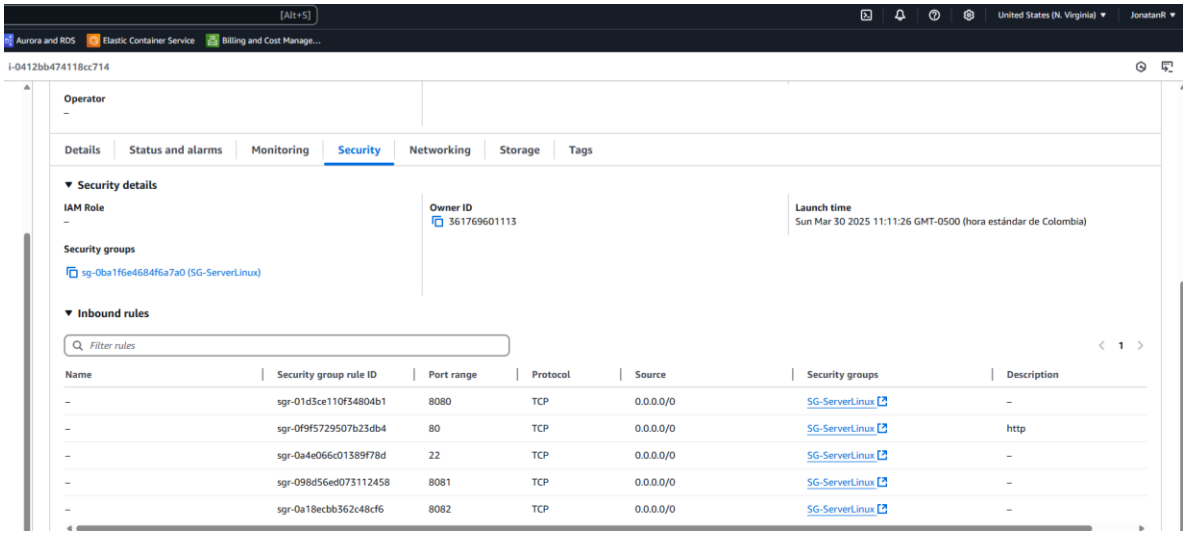
Los parámetros -d que significa demon para que se ejecute en segundo plano, it porque es interactivo y permitiría el ingreso al contenedor para realizar cualquier otra acción. Se pondrá –-name para el nombre y se configura -d –-restart always para que se inicie automáticamente con la instancia si se detiene o apaga con -p se configura el puerto 8080:80. El puerto 8080 es de la instancia y el 80 del contenedor. Se utiliza el parámetro -v de volumen y se genera el **enlace simbólico**, inicialmente con la ruta donde se encuentra la carpeta local o verdadera **/home/ec2-user/app1/** y la ruta que el apache busca por defecto sería ":**/usr/local/apache2/htdocs/",** y al final la imagen que se llama httpd.

NOTA: Tener en cuenta que al crear otro contenedor se debe cambiar el nombre de la aplicación o docker, el puerto al cual se conectaría y la ubicación donde quedaría si se desea que conecte a otro contenedor ejemplo la carpeta app2. Y se debe agregar el puerto al security group de la instancia

Con "docker ps" se ven los contenedores que se están ejecutando



Se debe validar que los puertos de los contenedores están en el security group de la instancia y que estén permitidos



Se crea una carpeta que, para el caso, que se llama app1 con permisos (chmod -R 777 app1) con el parámetro -R se da permisos a la carpeta y a los archivos que contenga. Se crea un archivo index.html con un texto simple para la primera aplicación que estará en el primer contenedor.

```
[root@ip-10-0-7-237 ec2-user]# ls
Carpeta1  Linux.pem  app1  app2  app3  file1  file2  fileCopia
[root@ip-10-0-7-237 ec2-user]# cd app1
[root@ip-10-0-7-237 app1]# ls
app2  index.html
[root@ip-10-0-7-237 app1]# nano index.html
[root@ip-10-0-7-237 app1]#
```

Se crean un par de carpetas más con el comando mkdir app2 y app3, se ingresa en las carpetas y se descarga una plantilla de un sitio web. Utilizando el comando wget y la URL de la descarga por ejemplo wget https://html5up.net/minimaxing/download y se descomprime el archivo download con el comando Unzip.

```
[root@ip-10-0-7-237 ec2-user]# ls -l
total 32
drwxr-xr-x. 2 root     root         6 Mar 23 00:25 Carpeta1
-rw-r--r--. 1 ec2-user ec2-user  1679 Mar 24 20:40 Linux.pem
drwxrwxrwx. 3 root     root        36 Mar 27 03:22 app1
drwxr-xr-x. 4 root     root       105 Mar 27 03:35 app2
drwxr-xr-x. 4 root     root     16384 Mar 27 03:42 app3
-rw-r--r--. 1 root     root        14 Mar 23 00:27 file1
-rw-r--r--. 1 root     root        31 Mar 23 00:29 file2
-rw-r--r--. 1 root     root        14 Mar 23 00:35 fileCopia
[root@ip-10-0-7-237 ec2-user]#
```

Instalación Servidor web Nginx

Se instalará un servicio en la instancia que es un servidor web llamado ngnix con el comando “dnf install nginx” el cual ingresa a trabajar por defecto en el puerto 80.

```
[root@ip-10-0-11-112 app2]# dnf install nginx
Last metadata expiration check: 6:14:41 ago on Wed Mar 26 17:37:01 2025.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                      Repository      Size
================================================================================
Installing:
 nginx                   x86_64        1:1.26.3-1.amzn2023.0.1      amazonlinux     33 k
Installing dependencies:
 generic-logos-httpd     noarch        18.0.0-12.amzn2023.0.3       amazonlinux     19 k
 gperftools-libs         x86_64        2.9.1-1.amzn2023.0.3         amazonlinux    308 k
 libunwind               x86_64        1.4.0-5.amzn2023.0.2         amazonlinux     66 k
 nginx-core              x86_64        1:1.26.3-1.amzn2023.0.1      amazonlinux    670 k
 nginx-filesystem        noarch        1:1.26.3-1.amzn2023.0.1      amazonlinux    9.6 k
 nginx-mimetypes         noarch        2.1.49-3.amzn2023.0.3        amazonlinux     21 k

Transaction Summary
================================================================================
Install  7 Packages

Total download size: 1.1 M
Installed size: 3.6 M
Is this ok [y/N]:
```

Configuración de Nginx como proxy reverso

Esto se puede configurar como un proxy reverso, por lo que, se crea copia del archivo de configuración por seguridad nginx.conf. Para acceder al archivo se utiliza la ruta cd /etc /nginx, que es la ruta por defecto donde se aloja.

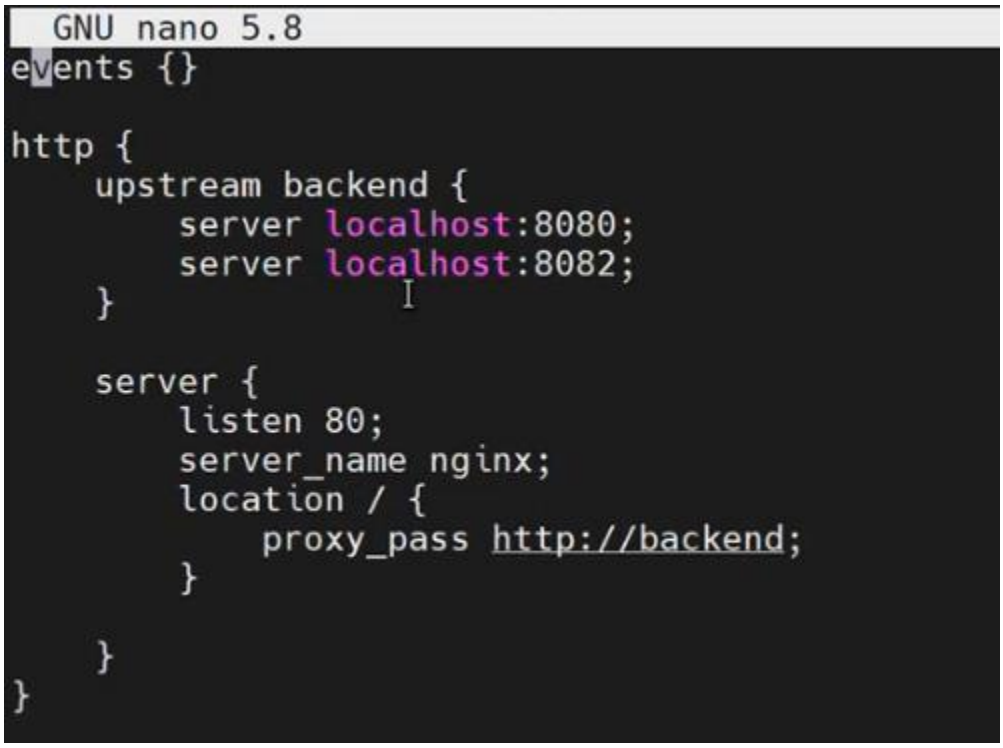
```
[root@ip-10-0-7-237 nginx]# ls
conf.d        fastcgi.conf.default  koi-utf     mime.types.default  nginx.conf.default   uwsgi_params
default.d     fastcgi_params        koi-win     nginx.conf          scgi_params          uwsgi_params.default
fastcgi.conf  fastcgi_params.default  mime.types  nginx.conf.BK       scgi_params.default  win-utf
[root@ip-10-0-7-237 nginx]#
```

Se modifica con el comando nado y con las siguientes líneas de configuración.

Dentro se tienen las secciones UPSTREAM y SERVER.

Server que escucha por el puerto 80 y el servidor se llama nginx. Y tiene una localización que es la raíz que envía las peticiones a la dirección http://backend.

Por definición de las palabras reservadas al indicar que es un proxy pass va y busca la función upstream que son las peticiones entrantes y de esta manera balancea las peticiones. Y se debe reiniciar el servicio de nginx ya que se modificó un archivo de configuración con, systemctl restart nginx



Configuración de Nginx para consultar por DNS

Para el ejercicio se va a modificar el archivo nginx.conf con las siguientes líneas lo cual permitiría consultar las páginas que se tienen en los contenedores con DNS y no

con IP’s y puertos directamente. Para ello también se debe modificar el archivo host que funcionaría como un DNS local.

```
  GNU nano 5.8
events {}

http {
    server {
        listen 80;
        server_name www.exito.com;
        location / {
            proxy_pass http://localhost:8080;
        }
    }

    server {
        listen 80;
        server_name www.milocal.com;
        location / {
            proxy_pass http://localhost:8081;
        }
    }

    server {
        listen 80;
        server_name www.miapplocal.com;
        location / {
            proxy_pass http://localhost:8082;
        }
    }
}
```

Luego de modificar el archivo host del equipo local se realizan pruebas ingresando a los contenedores que tienen las páginas configuras a través de los DNS’s configurados y no por IP ya que los usuarios no utilizan estás maneras sino por medio de un nombre.

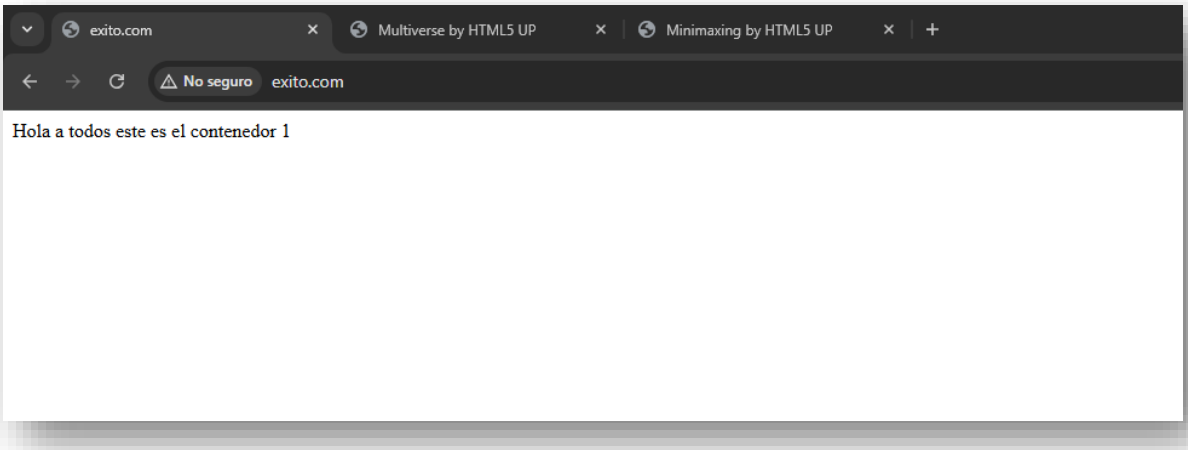
Configuración del archive host

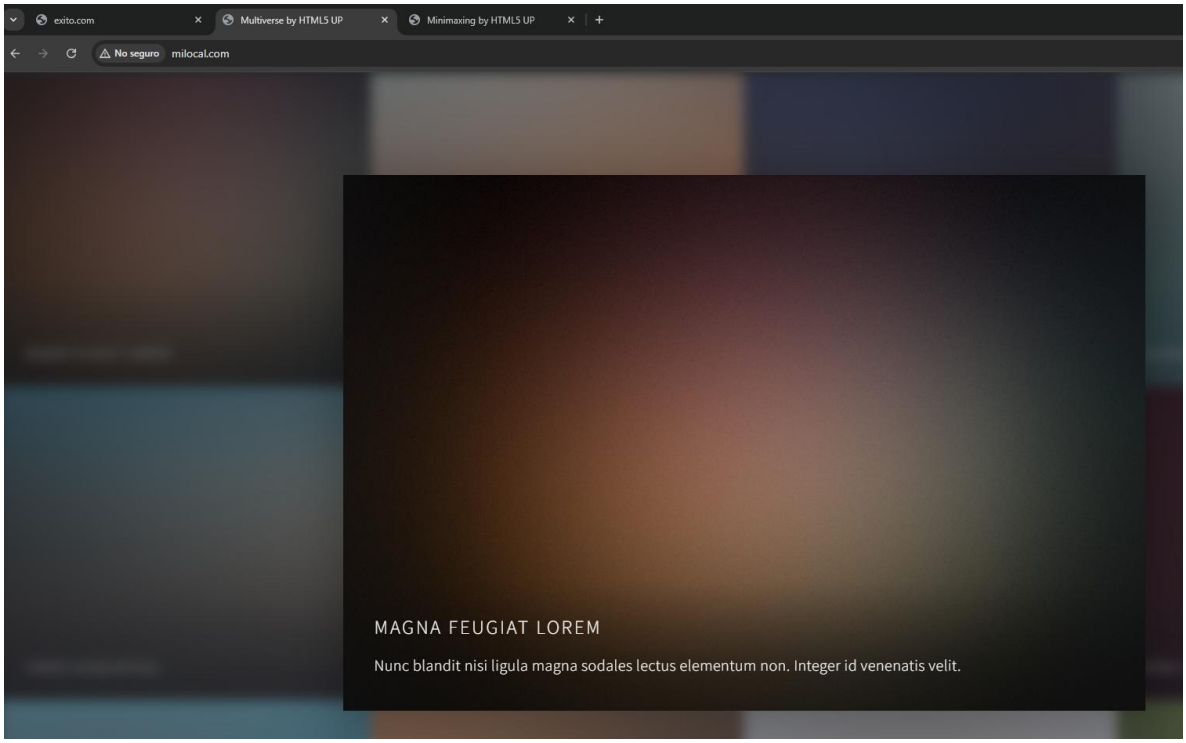
hosts

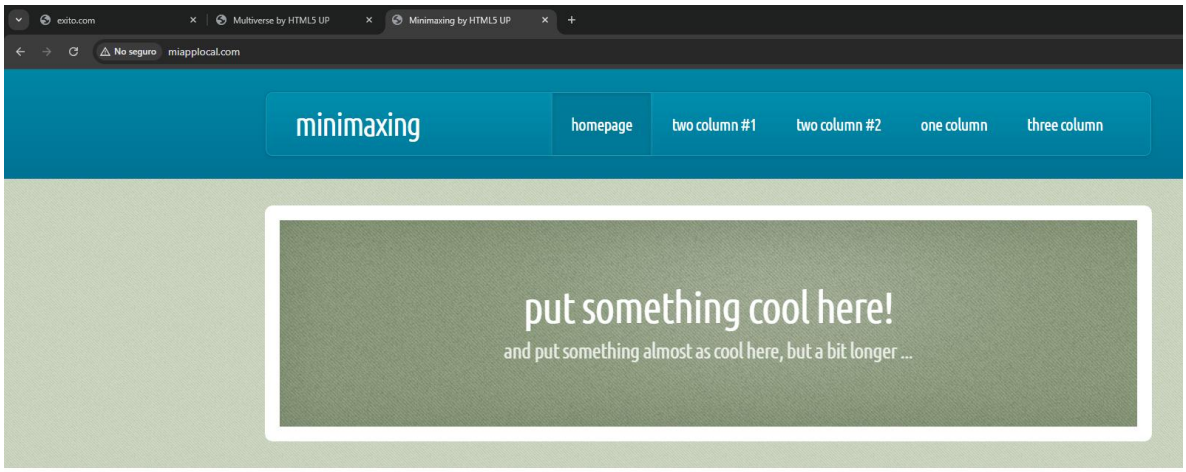
Archivo Editar Ver

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost

3.235.88.200 www.exito.com
3.235.88.200 www.milocal.com
3.235.88.200 www.miapplocal.com
```

Consulta de los sitios por DNS




exito.com
Multiverse by HTML5 UP
Minimaxing by HTML5 UP
No seguro
milocal.com
MAGNA FEUGIAT LOREM
Nunc blandit nisi ligula magna sodales lectus elementum non. Integer id venenatis velit.


exito.com
Multiverse by HTML5 UP
Minimaxing by HTML5 UP
No seguro
miapplocal.com
minimaxing
homepage
two column #1
two column #2
one column
three column
put something cool here!
and put something almost as cool here, but a bit longer ...


welcome to minimaxing!
This is Minimaxing, a fully responsive HTML5 site template designed by AJ and released for free by HTML5 UP. It features a simple, lightweight design, solid HTML5 and CSS3 code, and full responsive support for desktop, tablet, and small displays.
more cool designs ...
who are you guys?
Jane Anderson
Varius nibh. Suspendisse vitae magna eget et amet mollis justo facilisis amet quis.
James Doe
Vitae magna eget odio amet mollis justo facilisis amet quis. Sed sagittis consequat.
how about some links?
Sed neque nisi consequat
Dapibus sed mattis blandit
Quis accumsan lorem
Suspendisse varius ipsum
Eget et amet consequat
Quis accumsan lorem
Sed neque nisi consequat
Eget et amet consequat
Dapibus sed mattis blandit
Vitae magna sed dolore

Conclusiones

Definitivamente lo más relevantes es la utilización de contenedores, claro está a título personal. Lo comento por la manera en la que una sola instancia puede correr varias aplicaciones o contenedores sin llegar al punto en que se bloquee por exceso de consumos en los recursos. Es cierto que eran ejercicios con páginas de pruebas sencillas pero que una solo instancia con 1 gigabyte pueda ejecutar hasta 3 contenedores se me hizo una verdadera locura.

El poder montar toda una infraestructura desde la capa gratuita de AWS es en verdad algo que permite tener una gran experiencia, conocimiento y sin duda una grata sensación de lograr mucho con poco. Comprender de una manera práctica y contextual varios conceptos de la nube y a nivel técnico es naturalmente algo que mereció mucho la pena.

La manera en que un balanceador de carga puede ayudar a que la infraestructura permanezca estable es un ejercicio que siempre se debe tener en cuenta a la hora de planear la base para la construcción de una aplicación, base de datos y toda clase de infraestructura tecnológica.

Los servicios de almacenamiento tradicionales siguen siendo una opción, sin embargo, el poder contar con alternativas tan sencillas y potentes como el servicio S3 (Simple Storage Service) es en verdad una joya, y poder integrar inteligencia artificial al análisis y que ayudan a ver la manera en la que se almacena la información marca una diferencia en los costos y de cómo se realizar el propio cronograma de backups de las compañías.

La globalización rige las tendencias en los usos de los servicios de la nube, por las bondades a la hora de iniciar una infraestructura completa con apenas unos varios clics. Estas formas de prestar un servicio que se puede controlar completamente en costos responsabilidades y usabilidad son unas de las razones por las que migrar a la nube es una gran alternativa. Personalmente tengo un pero, y es el hecho de que se

Referencias

Nayak, S. (6 de febrero de 2025). Understanding Virtualization: A Comprehensive Guide. CloudOptimo. https://www.cloudoptimo.com/blog/understanding-virtualization-a-comprehensive-guide/

Amazon Web Services. (s.f.). *Cloud Products.* Amazon. Recuperado el 05 de abril de 2025, de https://aws.amazon.com/products/

Google Cloud. (s.f.). *Google Cloud Products.* Google. Recuperado el 05 de abril de 2025, de https://cloud.google.com/products

Microsoft Azure. (s.f.). *Azure Products.* Microsoft. Recuperado el 05 de abril de 2025, de https://azure.microsoft.com/en-us/products/