



**TRABAJO DE GRADO**  
**Opción Seminario-Diplomado.**

**Implementación de servidores web con Docker, Apache, NGINX y balanceadores de carga  
en AWS**

Corporación Universitaria Remington.  
Facultad de Ingenierías  
Ingeniería de Sistemas

Santiago Pulgarín Ocampo  
Tutor: Juan Pablo Berrío López  
Seminario-Diplomado.  
2025

**Tabla de Contenidos**

Resumen.....	3
Marco conceptual y contextual .....	4
Desarrollo e implementación del aprendizaje.....	5
Conclusiones.....	18
Referencias.....	19

## **Resumen**

En este proyecto realizado bajo la infraestructura de AWS, directamente desde su plataforma, se crearon múltiples instancias de conexión pública y privada en la arquitectura Linux. Donde igualmente se implementaron balanceadores de carga y algunas políticas de Auto Scaling. Las instancias fueron la base principal para el despliegue de servicios web en un entorno local, garantizando el correcto acceso en todo momento a estos mismos sitios.

También se realizó un cambio en la implementación y se optó por crear y configurar algunos contenedores Docker para el acceso independiente de cada sitio web. Todo esto dentro de una misma instancia base, así permitiendo más flexibilidad y acceso de las aplicaciones.

Para dar un complemento a la arquitectura, se procedió con la instalación de Nginx dentro de esta misma instancia para que actuara como proxy inverso y distribuyera el tráfico hacia los diferentes Docker, según los dominios personalizados. Esto además de dar un buen funcionamiento y organización al entorno, también permite a futuro un fácil crecimiento y mantenimiento.

En todo momento se tuvo en cuenta la correcta configuración de los grupos de seguridad, puertos y políticas de acceso. Todo en un enfoque automatizado, facilitando futuros aumentos de los servicios.

Este proyecto nos permite demostrar como la computación en la nube nos permiten obtener infraestructuras modernas y eficientes, adaptadas a las necesidades actuales de las organizaciones.

## **Palabras clave**

Balanceadores de carga, arquitectura, Nginx, Auto Scaling, contenedores Docker.

## **Marco conceptual y contextual**

### Balancedores de carga

Un mecanismo como los balancedores de carga en la nube nos permiten la distribución automática del tráfico entrante a los servicios, según el rendimiento y la carga de los recursos de AWS garantizando la alta disponibilidad y eficiencia.

(Amazon Web Services, 2025).

### Contenedores Docker

Podríamos definir los contenedores Docker como un software que nos permite empaquetar dependencias necesarias para que los servicios que estamos usando se ejecuten de manera óptima y confiable. Estos contenedores incluyen una imagen que contiene el código, tiempo de ejecución, herramientas del sistema, configuración, entre otros elementos necesarios para el correcto funcionamiento.

(Docker Inc., 2025).

### Auto Scaling AWS

Nos referimos a Auto Scaling como la herramienta que nos proporciona AWS a los administradores para configurar y ejecutar de manera automática la creación o eliminación de instancias según la carga de los servicios, así optimizando recursos y garantizando estabilidad.

(Amazon Web Services, 2025).

### Instancias EC2

Las instancias EC2 que nos brinda AWS permiten la creación de los múltiples servidores virtuales que se requieran. Siendo totalmente configurables, tanto en seguridad, redes y almacenamiento, además de ser escalables. Cada instancia puede ofrecer una combinación diferente de recursos computacionales.

(Amazon Web Services, 2025).

### NGINX

El proxy inverso de NGINX es básicamente la configuración de un archivo interno que permite balancear el tráfico entre varios servidores o contenedores Docker. La principal funcionalidad es aliviar las transacciones entrantes y optimizar el rendimiento, además gestionando las solicitudes bajo distintos protocolos de HTTP.

(NGINX Inc., 2025).

## Desarrollo e implementación del aprendizaje

### Balancedador de carga con Auto Scaling

1. Como primera parte, vamos a crear las instancias Dinamica1 y Dinamica2, ambas bajo la configuración Amazon Linux en su versión Amazon Linux 2023 AMI. Para el tipo de instancia seleccionaremos una configuración específica de los recursos del servidor, que será t2, 1vCPU y 1GB de memoria.

Luego, generamos el certificado de seguridad (Key pair), bajo el nombre Linux y en formato .ppk.

En la configuración de red, seleccionaremos el vpc por defecto previamente creado VPC-WEB. Dado que serán publicaciones de accesos públicas, elegimos la subred pública y activamos la auto asignación de ip pública. El resto de configuración la dejamos por defecto.

**Launch an instance** [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

**Name and tags** [Info](#)

Name  
Dinamica2 [Add additional tags](#)

▼ **Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents **Quick Start**

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debian

aws Mac ubuntu Microsoft Red Hat SUSE Linux debian

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

Amazon Linux 2023 AMI  
ami-08b5b3e93ed654d19 (64-bit (x86), uefi-preferred) / ami-0eae2a0fc13b15fce (64-bit (Arm), uefi)  
Virtualization: hvm ENA enabled: true Root device type: ebs [Free tier eligible](#) ▼

### ▼ Instance type [Info](#) | [Get advice](#)

**Instance type**

t2.micro Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour On-Demand RHEL base pricing: 0.026 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

All generations [Compare instance types](#)

**Additional costs apply for AMIs with pre-installed software**

### ▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

**Key pair name - required**

Linux ↕ [Create new key pair](#)

### ▼ Network settings [Info](#)

**VPC - required** [Info](#)

vpc-0b83c899cd0fbc288 (VPC-WEB-vpc) ↕ [Refresh](#)

10.0.0.0/16

**Subnet** [Info](#)

subnet-07186e3feac54b166 VPC-WEB-subnet-public2-us-east-1b ↕ [Create new subnet](#)

VPC: vpc-0b83c899cd0fbc288 Owner: 513686310065 Availability Zone: us-east-1b

Zone type: Availability Zone IP addresses available: 4091 CIDR: 10.0.16.0/20

**Auto-assign public IP** [Info](#)

Enable ↕

**Additional charges apply when outside of free tier allowance**

### Firewall (security groups) [Info](#)

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

Create security group  [Select existing security group](#)

**Common security groups** [Info](#)

Select security groups ↕ [Compare security group rules](#)

SG-Dinamica1 sg-0009ef6b004ffd492 ✕

VPC: vpc-0b83c899cd0fbc288

Security groups that you add or remove here will be added to or removed from all your network interfaces.

**▶ Advanced network configuration**

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

1x  GiB  Root volume, 3000 IOPS, Not encrypted

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

[Add new volume](#)

ℹ [Click refresh to view backup information](#) ↕

The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems Edit

2. Como segunda parte, nos conectamos a la administración de los servidores mediante MobaXTerm y procedemos a instalar el servidor web Apache en cada una de las instancias con el comando `dnf install httpd`. Luego, iniciamos el servicio de Apache en cada servidor con el comando `systemctl start httpd` y lo habilitamos para que se inicie automáticamente usando el comando `systemctl enable httpd`.

Posterior a esto, configuramos nuestras vistas (archivos `index.html`) dentro de la ruta `/var/www/html`

```
[root@ip-10-0-31-166 ec2-user]# dnf install httpd
Last metadata expiration check: 0:05:52 ago on Thu Mar 20 03:35:57 2025.
Dependencies resolved.
Package                               Architecture      Version           Repository        Size
-----
Installing:
httpd                                  x86_64            2.4.62-1.amzn2023    amazonlinux        48 k
Installing dependencies:
apr                                     x86_64            1.7.5-1.amzn2023.0.4    amazonlinux        129 k
apr-util                               x86_64            1.6.3-1.amzn2023.0.1    amazonlinux        98 k
generic-logos-httpd                   noarch            18.0.0-12.amzn2023.0.3    amazonlinux        19 k
httpd-core                             x86_64            2.4.62-1.amzn2023      amazonlinux        1.4 M
httpdfilesystem-2.4.62-1.amzn2023     noarch            2.4.62-1.amzn2023      amazonlinux        14 k
httpd-tools                             x86_64            2.4.62-1.amzn2023      amazonlinux        81 k
libbrotli                              x86_64            1.0.9-4.amzn2023.0.2    amazonlinux        315 k
mailcap                                 noarch            2.1.49-3.amzn2023.0.3    amazonlinux        33 k
Installing weak dependencies:
apr-util-openssl                       x86_64            1.6.3-1.amzn2023.0.1    amazonlinux        17 k
mod_http2                              x86_64            2.0.27-1.amzn2023.0.3    amazonlinux        166 k
mod_lua                                 x86_64            2.4.62-1.amzn2023      amazonlinux        61 k

Transaction Summary
-----
Install 12 Packages
Total download size: 2.3 M
Installed size: 6.9 M
Is this ok [y/N]: y
Downloading Packages:
(1/12): apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64.rpm           314 kB/s | 17 kB  00:00
(2/12): apr-1.7.5-1.amzn2023.0.4.x86_64.rpm                       2.1 MB/s | 129 kB  00:00
(3/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rpm                  1.5 MB/s | 98 kB  00:00
(4/12): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm     1.0 MB/s | 19 kB  00:00
(5/12): httpd-2.4.62-1.amzn2023.x86_64.rpm                        2.4 MB/s | 48 kB  00:00
(6/12): httpdfilesystem-2.4.62-1.amzn2023.noarch.rpm              654 kB/s | 14 kB  00:00
(7/12): httpd-tools-2.4.62-1.amzn2023.x86_64.rpm                  2.0 MB/s | 81 kB  00:00
(8/12): httpd-core-2.4.62-1.amzn2023.x86_64.rpm                  2.1 MB/s | 1.4 MB  00:00
(9/12): mailcap-2.1.49-3.amzn2023.0.3.noarch.rpm                  1.4 MB/s | 33 kB  00:00
(10/12): libbrotli-1.0.9-4.amzn2023.0.2.x86_64.rpm                6.0 MB/s | 315 kB  00:00
(11/12): mod_http2-2.0.27-1.amzn2023.0.3.x86_64.rpm              3.0 MB/s | 166 kB  00:00
(12/12): mod_lua-2.4.62-1.amzn2023.x86_64.rpm                     1.3 MB/s | 61 kB  00:00
-----
Total: 11 MB/s | 2.3 MB  00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction:
  Preparing                : apr-1.7.5-1.amzn2023.0.4.x86_64                1/12
  Installing                : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64    1/12
  Installing                : apr-util-1.6.3-1.amzn2023.0.1.x86_64          2/12
  Installing                : mailcap-2.1.49-3.amzn2023.0.3.noarch           3/12
  Installing                : httpd-tools-2.4.62-1.amzn2023.x86_64          4/12
  Installing                : libbrotli-1.0.9-4.amzn2023.0.2.x86_64         5/12
  Installing                : libbrotli-1.0.9-4.amzn2023.0.2.x86_64         6/12
  Running scriptlet: httpdfilesystem-2.4.62-1.amzn2023.noarch              7/12
  Installing                : httpdfilesystem-2.4.62-1.amzn2023.noarch       7/12
```

```
Installed:
apr-1.7.5-1.amzn2023.0.4.x86_64      apr-util-1.6.3-1.amzn2023.0.1.x86_64      apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64      generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
httpd-2.4.62-1.amzn2023.x86_64      httpd-core-2.4.62-1.amzn2023.x86_64      httpdfilesystem-2.4.62-1.amzn2023.noarch      httpd-tools-2.4.62-1.amzn2023.x86_64
libbrotli-1.0.9-4.amzn2023.0.2.x86_64  mailcap-2.1.49-3.amzn2023.0.3.noarch      mod_http2-2.0.27-1.amzn2023.0.3.x86_64      mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[root@ip-10-0-31-166 ec2-user]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
   Active: inactive (dead)
     Docs: man:httd.service(8)
[root@ip-10-0-31-166 ec2-user]# systemctl start httpd
[root@ip-10-0-31-166 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
   Active: active (running) since 2025-03-20 03:50:10 UTC; 45 ago
     Docs: man:httd.service(8)
  Main PID: 27679 (httpd)
    Status: "Started, listening on: port 80"
     Tasks: 177 (limit: 1111)
   Memory: 12.0M
     CPU: 55ms
   CGroup: /system.slice/httpd.service
           └─27678 /usr/sbin/httpd -DFOREGROUND
             └─27683 /usr/sbin/httpd -DFOREGROUND
               └─27685 /usr/sbin/httpd -DFOREGROUND
                 └─27686 /usr/sbin/httpd -DFOREGROUND
                   └─27687 /usr/sbin/httpd -DFOREGROUND

Mar 20 03:50:10 ip-10-0-31-166.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Mar 20 03:50:10 ip-10-0-31-166.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Mar 20 03:50:10 ip-10-0-31-166.ec2.internal httpd[27678]: Server configured, listening on: port 80
[root@ip-10-0-31-166 ec2-user]#
```

```

[ec2-44-201-136-197.compute-1] x [ec2-174-129-144-107.compute-1] x
[root@ip-10-0-31-166 ec2-user]# cd /var/www/html/
[root@ip-10-0-31-166 html]# wget https://html5up.net/halcyonic/download
--2025-03-20 04:03:01-- https://html5up.net/halcyonic/download
Resolving html5up.net (html5up.net)... 172.67.195.190, 104.21.76.136, 2606:4700:3030:6815:4c88, ...
Connecting to html5up.net (html5up.net)|172.67.195.190|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-zip]
Saving to: 'download'

download [ <=>]
2025-03-20 04:03:01 (60.2 MB/s) - 'download' saved [116503]

[root@ip-10-0-31-166 html]# ls
download
[root@ip-10-0-31-166 html]# unzip download
Archive:  download
  inflating: LICENSE.txt
   creating: assets/
   creating: assets/js/
  inflating: assets/js/breakpoints.min.js
  inflating: assets/js/jquery.min.js
  inflating: assets/js/util.js
  inflating: assets/js/main.js
  inflating: assets/js/browser.min.js
   creating: assets/css/
  inflating: assets/css/main.css
   creating: assets/css/images/
  inflating: assets/css/images/button-big-hover.svg
  inflating: assets/css/images/icon-checkmark.png
  inflating: assets/css/images/bg03.jpg
  inflating: assets/css/images/mobileUI-site-nav-opener-bg.svg
  inflating: assets/css/images/bg02.jpg
  inflating: assets/css/images/button-big-active.svg
  inflating: assets/css/images/bg01.jpg
  inflating: assets/css/images/button-big.svg
  inflating: assets/css/images/bg04.png
  inflating: assets/css/images/bg04.jpg
   creating: assets/sass/
  inflating: assets/sass/main.scss
   creating: assets/sass/libs/
  inflating: assets/sass/libs/_functions.scss
  inflating: assets/sass/libs/_mixins.scss
  inflating: assets/sass/libs/_vars.scss
  inflating: assets/sass/libs/_html-grid.scss
  inflating: assets/sass/libs/_vendor.scss
  inflating: assets/sass/libs/_breakpoints.scss
  inflating: README.txt
  inflating: threecolumn.html
  inflating: index.html
  inflating: twocolumn1.html
  inflating: onecolumn.html
   creating: images/
  inflating: images/pic05.jpg
  inflating: images/pic08.jpg
  inflating: images/pic07.jpg
  inflating: images/pic06.jpg
  inflating: images/pic03.jpg

```

3. En la tercera parte procedemos a crear el balanceador de cargas externo HTTPS para los servidores con IPV4, seleccionando el VPC que ya habíamos creado. Posteriormente creamos el target group para las instancias y muy importante, configurando los health checks y los tiempos de validación. Agregamos las instancias al balanceador.

### 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**

TG-Dinamica

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Protocol : Port**

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation.

HTTP 80

**Health check port**

The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.

**Traffic port**

**Override**

**Healthy threshold**

The number of consecutive health checks successes required before considering an unhealthy target healthy.

2

2-10

**Unhealthy threshold**

The number of consecutive health check failures required before considering a target unhealthy.

2

2-10

**Timeout**

The amount of time, in seconds, during which no response means a failed health check.

5 seconds

2-120

**Interval**

The approximate amount of time between health checks of an individual target.

20 seconds

5-300

**Success codes**

The HTTP codes to use when checking for a successful response from a target. You can specify multiple values (for example, "200,202") or a range of values (for example, "200-299").

200

**Available instances (4/4)**

Filter instances

<input checked="" type="checkbox"/>	Instance ID	Name	State	Security groups	Zone	Private IPv4 address	Subnet
<input checked="" type="checkbox"/>	i-02b9e9dee4ace3ac4	Dinamica4	Running	SG-Dinamica1	us-east-1a	10.0.8.4	subnet
<input checked="" type="checkbox"/>	i-083defdecf1dcaf2f	Dinamica3	Running	SG-Dinamica1	us-east-1a	10.0.7.191	subnet
<input checked="" type="checkbox"/>	i-0dd0ffb60428ba60c	Dinamica2	Running	SG-Dinamica1	us-east-1b	10.0.31.166	subnet
<input checked="" type="checkbox"/>	i-0b6ef16aa11bdd078	Dinamica1	Running	SG-Dinamica1	us-east-1a	10.0.0.50	subnet

4 selected

**Ports for the selected instances**  
Ports for routing traffic to the selected instances.

80

1-65535 (separate multiple ports with commas)

[Include as pending below](#)

**LB-Dinamica**

Details

<b>Load balancer type</b> Application	<b>Status</b> Provisioning	<b>VPC</b> vpc-0b83c899cd0fbc288	<b>Load balancer IP address type</b> IPv4
<b>Scheme</b> Internet-facing	<b>Hosted zone</b> Z35SXDOTRQ7X7K	<b>Availability Zones</b> subnet-0e9f28f9f399e013 us-east-1a (use1-az2) subnet-07186e3feac54b166 us-east-1b (use1-az4)	<b>Date created</b> March 24, 2025, 14:36 (UTC-05:00)
<b>Load balancer ARN</b> arn:aws:elasticloadbalancing:us-east-1:513686310065:loadbalancer/app/LB-Dinamica/b1f8ad1ca34c9cda	<b>DNS name info</b> LB-Dinamica-1352301026.us-east-1.elb.amazonaws.com (A Record)		

Listeners and rules (1)

Filter listeners

<input type="checkbox"/>	Protocol/Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
<input type="checkbox"/>	HTTP:80	Forward to target group • TG-Dinamica (1 (100%)) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable	Not applic

**sg-0293f832e052495ed - default**

Details

<b>Security group name</b> default	<b>Security group ID</b> sg-0293f832e052495ed	<b>Description</b> default VPC security group	<b>VPC ID</b> vpc-0b83c899cd0fbc288
<b>Owner</b> 513686310065	<b>Inbound rules count</b> 2 Permission entries	<b>Outbound rules count</b> 1 Permission entry	

Inbound rules (2)

Search

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-074babe672690111d	-	All traffic	All	All	sg-0293f832e052495ed...	-
<input type="checkbox"/>	-	sgr-055b37a64922a7aa5	IPv4	HTTP	TCP	80	0.0.0.0/0	http

4. Procedemos a crear el Auto Scaling para la creación automática de instancias. Para verificar esto, vamos a eliminar una de las instancias para que luego el Auto Scaling cree de manera automática una instancia nueva.

### Name

**Auto Scaling group name**  
Enter a name to identify the group.

Must be unique to this account in the current Region and no more than 255 characters.

### Launch template Info

[Switch to launch configuration](#)

**Launch template**  
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

DinamicaWeb

[Create a launch template](#)

**Version**  
Default (1)

[Create a launch template version](#)

<b>Description</b> -	<b>Launch template</b> DinamicaWeb <a href="#">↗</a> lt-0a88e98d6610a8feb	<b>Instance type</b> t2.micro
<b>AMI ID</b> ami-061455286e5ba0731	<b>Security groups</b> -	<b>Request Spot Instances</b> No
<b>Key pair name</b> Linux	<b>Security group IDs</b> sg-0009ef6b004ffd492 <a href="#">↗</a>	

**Additional details**

<b>Storage (volumes)</b>	<b>Date created</b>	
--------------------------	---------------------	--

### Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**  
Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0b83c899cd0fbc288 (VPC-WEB-vpc)  
10.0.0.0/16

[Create a VPC](#)

**Availability Zones and subnets**  
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

us-east-1a | subnet-06aaec1c4508d2cf8 (VPC-WEB-subnet-private1-us-east-1a)   
10.0.128.0/20

us-east-1b | subnet-00d613c42eae48ea (VPC-WEB-subnet-private2-us-east-1b)   
10.0.144.0/20

[Create a subnet](#)

**Availability Zone distribution - *new***  
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

**Balanced best effort**  
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.

**Balanced only**  
If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

Instances (6) info

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
	i-0e9bf4741fd234b5e	Running	t2.micro	Initializing	View alarms +	us-east-1b	-	-	-
Dinamica1	i-0b6ef16aa11bd0078	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-201-136-197.co...	44.201.136.197	-
	i-004419bf45366a189	Running	t2.micro	Initializing	View alarms +	us-east-1a	-	-	-
Dinamica3	i-083defdec1dcaf2f	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-52-207-220-211.co...	52.207.220.211	-
Dinamica4	i-02b9e9dee4ace3ac4	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-202-143-139.co...	44.202.143.139	-
Dinamica2	i-0dd0f860428ba60c	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-174-129-144-107.c...	174.129.144.107	-

Select an instance

Instances (1/6) info

Last updated 10 minutes ago

Find Instance by attribute or tag (case-sensitive)

All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
	i-0e9bf4741fd234b5e	Running	t2.micro	Initializing	View alarms +	us-east-1b	-	-	-
Dinamica1	i-0b6ef16aa11bd0078	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-201-136-197.co...	44.201.136.197	-
	i-004419bf45366a189	Running	t2.micro	Initializing	View alarms +	us-east-1a	-	-	-
Dinamica3	i-083defdec1dcaf2f	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-52-207-220-211.co...	52.207.220.211	-
Dinamica4	i-02b9e9dee4ace3ac4	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-202-143-139.co...	44.202.143.139	-
Dinamica2	i-0dd0f860428ba60c	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-174-129-144-107.c...	174.129.144.107	-

Stop instance  
Start instance  
Reboot instance  
Hibernate instance  
Terminate (delete) instance

Activity history (6)

Filter activity history

Status	Description	Cause	Start time	End time
Successful	Launching a new EC2 instance: i-0a73ff67c5630a403	At 2025-03-24T20:27:54Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 March 24, 03:27:56 PM -05:00	2025 March 24, 03:28:02 PM -05:00
Connection draining in progress	Terminating EC2 instance: i-004419bf45366a189 - Waiting For ELB Connection Draining.	At 2025-03-24T20:27:54Z an instance was taken out of service in response to an ELB system health check failure.	2025 March 24, 03:27:54 PM -05:00	-
Successful	Launching a new EC2 instance: i-08aef07035867c44d	At 2025-03-24T20:21:45Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 March 24, 03:21:47 PM -05:00	2025 March 24, 03:21:53 PM -05:00
Successful	Terminating EC2 instance: i-0e9bf4741fd234b5e	At 2025-03-24T20:21:45Z an instance was taken out of service in response to an ELB system health check failure.	2025 March 24, 03:21:45 PM -05:00	2025 March 24, 03:27:27 PM -05:00
Successful	Launching a new EC2 instance: i-004419bf45366a189	At 2025-03-24T20:15:36Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-03-24T20:15:46Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2025 March 24, 03:15:48 PM -05:00	2025 March 24, 03:15:54 PM -05:00
Successful	Launching a new EC2 instance: i-0e9bf4741fd234b5e	At 2025-03-24T20:15:36Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-03-24T20:15:46Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2025 March 24, 03:15:47 PM -05:00	2025 March 24, 03:15:54 PM -05:00

Successfully initiated termination (deletion) of i-0e9bf4741fd234b5e

Instances (7) info

Last updated 1 minute ago

Find Instance by attribute or tag (case-sensitive)

All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
	i-08aef07035867c44d	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	-	-	-
	i-0e9bf4741fd234b5e	Terminated	t2.micro	-	View alarms +	us-east-1b	-	-	-
Dinamica1	i-0b6ef16aa11bd0078	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-201-136-197.co...	44.201.136.197	-
	i-004419bf45366a189	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	-	-	-
Dinamica3	i-083defdec1dcaf2f	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-52-207-220-211.co...	52.207.220.211	-
Dinamica4	i-02b9e9dee4ace3ac4	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-44-202-143-139.co...	44.202.143.139	-
Dinamica2	i-0dd0f860428ba60c	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-174-129-144-107.c...	174.129.144.107	-

Select an instance

5. Por último creamos la política de Scaling para un umbral en este caso del 75%

### Create dynamic scaling policy

**Policy type**  
Target tracking scaling

**Scaling policy name**  
Target Tracking Policy

**Metric type** Info  
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.  
Average CPU utilization

**Target value**  
75

**Instance warmup** Info  
300 seconds

Disable scale in to create only a scale-out policy

[Cancel](#) [Create](#)

## Implementación NGINX

1. Como primer punto vamos a seleccionar la instancia sobre la que queremos trabajar y la ip publica que esta misma nos brindará.

EC2 > Instances

Instances (4) Info

Find Instance by attribute or tag (state-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Dinamica4	i-02b9e9dee4ace3ac4	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-143-139.co...	44.202.143.139	-
Dinamica3	i-083d4fdedcf1dca2f1	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-52-207-220-211.co...	52.207.220.211	-
Dinamica2	i-0d409860428ba6f0c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-174-129-144-107.c...	174.129.144.107	-
Dinamica1	i-086ef16aa11bd4078	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-201-136-197.co...	44.201.136.197	-

Select an instance

### Instance summary for i-02b9e9dee4ace3ac4 (Dinamica4) Info

Updated 10 minutes ago

<b>Instance ID</b> i-02b9e9dee4ace3ac4	<b>Public IPv4 address</b> 44.202.143.139   open address	<b>Private IPv4 addresses</b> 10.0.8.4
<b>IPv6 address</b> -	<b>Instance state</b> Running	<b>Public IPv4 DNS</b> ec2-44-202-143-139.compute-1.amazonaws.com   open address
<b>Hostname type</b> IP name: ip-10-0-8-4.ec2.internal	<b>Private IP DNS name (IPv4 only)</b> ip-10-0-8-4.ec2.internal	<b>Elastic IP addresses</b> -
<b>Answer private resource DNS name</b> -	<b>Instance type</b> t2.micro	<b>AWS Compute Optimizer finding</b> Opt-in to AWS Compute Optimizer for recommendations.   Learn more
<b>Auto-assigned IP address</b> 44.202.143.139 [Public IP]	<b>VPC ID</b> vpc-0b83c899cd0fbc288 (VPC-WEB-vcpc)	<b>Auto Scaling Group name</b> -
<b>IAM Role</b> -	<b>Subnet ID</b> subnet-0e9f28f9f399e013 (VPC-WEB-subnet-public1-us-east-1a)	<b>Managed</b> false
<b>IMDSv2</b> Required	<b>Instance ARN</b> arn:aws:ec2:us-east-1:513686310065:instance/i-02b9e9dee4ace3ac4	
<b>Operator</b> -		



4. Posteriormente instalamos la imagen httpd del Docker con el comando docker pull httpd.

```
[root@ip-10-0-8-4 ec2-user]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-10-0-8-4 ec2-user]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
6e909acdb790: Pull complete
9f9b03a66afb: Pull complete
4f4fb700ef54: Pull complete
09bf08b13dbd: Pull complete
084c58879b9a: Pull complete
c61868f0ad74: Pull complete
Digest: sha256:391a8eb0c1ed464163da46099606a5ec293705118f3054d6c60f5957e2485bd0
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[root@ip-10-0-8-4 ec2-user]#
```

5. Procedemos a crear las carpetas dgh (Dinámica Gerencial Hospitalaria) donde se van a alojar los index de cada uno de los sitios web.

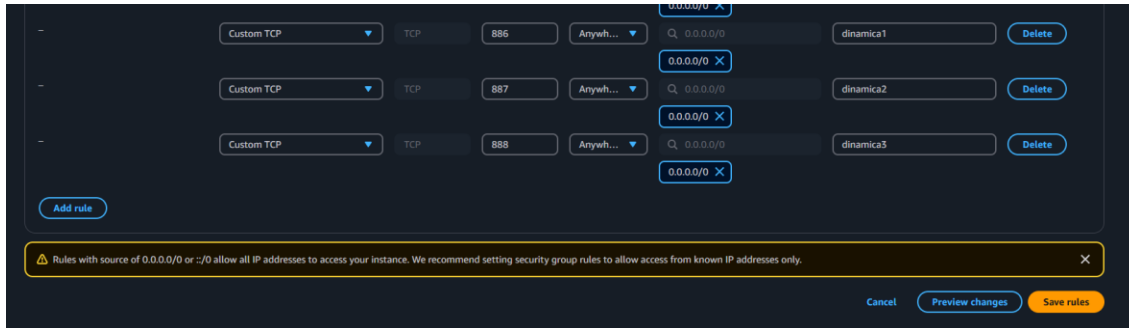
```
[root@ip-10-0-8-4 ec2-user]# mkdir dgh1 dgh2 dgh3
[root@ip-10-0-8-4 ec2-user]#
[root@ip-10-0-8-4 ec2-user]#
[root@ip-10-0-8-4 ec2-user]#
[root@ip-10-0-8-4 ec2-user]#
[root@ip-10-0-8-4 ec2-user]#
```

```
[root@ip-10-0-8-4 ec2-user]# cd dgh1
[root@ip-10-0-8-4 dgh1]# nano index.html
[root@ip-10-0-8-4 dgh1]# ls
index.html
[root@ip-10-0-8-4 dgh1]# cd ..
[root@ip-10-0-8-4 ec2-user]# cd dgh2
[root@ip-10-0-8-4 dgh2]# nano index.html
[root@ip-10-0-8-4 dgh2]# ls
index.html
[root@ip-10-0-8-4 dgh2]# cd ..
[root@ip-10-0-8-4 ec2-user]# cd dgh3
[root@ip-10-0-8-4 dgh3]# nano index.html
```

6. El siguiente paso es crear los contenedores de cada sitio con el comando docker run -dit ---name (nombre docker) -d --restart always -p (puerto para cada sitio):80 -v /home/ec2-user/(nombre carpeta)/:/usr/local/apache2/htdocs/ httpd

```
[root@ip-10-0-8-4 ec2-user]# chmod -R 777 dgh1 dgh2 dgh3
[root@ip-10-0-8-4 ec2-user]# docker run -dit --name dgh1 -d --restart always -p 8086:80 -v /home/ec2-user/dgh1:/usr/local/apache2/htdocs/ httpd
d10f32591495df69c7ef11712beda9e4b13915c0a3bee72425f0db5168b10a2
[root@ip-10-0-8-4 ec2-user]# docker run -dit --name dgh2 -d --restart always -p 8087:80 -v /home/ec2-user/dgh2:/usr/local/apache2/htdocs/ httpd
a37ead10f5d80b4b8c8b1fcff2a930bcd5978d2d3b4b324b6f259975838973a
[root@ip-10-0-8-4 ec2-user]# docker run -dit --name dgh3 -d --restart always -p 8088:80 -v /home/ec2-user/dgh3:/usr/local/apache2/htdocs/ httpd
bb73f8b4709a9131239efd0e023e769d30c3e32a2ad8cdb63dd0af2d55dc3cc4
[root@ip-10-0-8-4 ec2-user]#
```

7. Comprobamos que los puertos estén habilitados en el Security Group, si no procedemos a crearlos



Se realiza corrección, ya que les faltaba un cero a los puertos.

8. Instalamos el nginx en la instancia

```
[root@ip-10-0-0-4 ec2-user]#
[root@ip-10-0-0-4 ec2-user]# dnf install nginx
Last metadata expiration check: 0:46:48 ago on Mon Mar 31 03:05:54 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-2.9.1-1.amzn2023.0.3  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]: y
Downloading Packages:
(1/7): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm 489 kB/s | 19 kB 00:00
(2/7): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rpm             1.5 MB/s | 66 kB 00:00
(3/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm      6.0 MB/s | 308 kB 00:00
(4/7): nginx-1.26.3-1.amzn2023.0.1.x86_64.rpm               1.5 MB/s | 33 kB 00:00
(5/7): nginx-core-1.26.3-1.amzn2023.0.1.x86_64.rpm           22 MB/s | 670 kB 00:00
(6/7): nginx-filesystem-1.26.3-1.amzn2023.0.1.noarch.rpm    383 kB/s | 9.6 kB 00:00
(7/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm      957 kB/s | 21 kB 00:00
-----
Total: 9.6 MB/s | 1.1 MB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing: 1/1
Running scriptlet: nginx-filesystem-1:1.26.3-1.amzn2023.0.1.noarch 1/7
Installing: 1/7
Installing:  : nginx-filesystem-1:1.26.3-1.amzn2023.0.1.noarch 1/7
Installing:  : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 2/7
Installing:  : libunwind-1.4.0-5.amzn2023.0.2.x86_64 3/7
Installing:  : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 4/7
Installing:  : nginx-core-1:1.26.3-1.amzn2023.0.1.x86_64 5/7
Installing:  : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 6/7
Installing:  : nginx-1:1.26.3-1.amzn2023.0.1.x86_64 7/7
Running scriptlet: nginx-1:1.26.3-1.amzn2023.0.1.x86_64 7/7
Verifying:  : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 1/7
Verifying:  : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 2/7
Verifying:  : libunwind-1.4.0-5.amzn2023.0.2.x86_64 3/7
Verifying:  : nginx-1:1.26.3-1.amzn2023.0.1.x86_64 4/7
Verifying:  : nginx-core-1:1.26.3-1.amzn2023.0.1.x86_64 5/7
Verifying:  : nginx-filesystem-1:1.26.3-1.amzn2023.0.1.noarch 6/7
Verifying:  : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 7/7
=====
WARNING:
=====

Complete!
[root@ip-10-0-0-4 ec2-user]# systemctl start nginx
[root@ip-10-0-0-4 ec2-user]# systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[root@ip-10-0-0-4 ec2-user]# █
```

9. Se crea un archivo de configuración `/etc/nginx/conf.d/dgh.conf` por separado ya que así nos permite una configuración modular, limpia y escalable del servidor, ya que estamos gestionando múltiples sitios web en una misma instancia.

Se definen bloques `server` que permiten enrutar HTTP a diferentes sitios web, para este caso con dominios personalizados (`dghweb1,dghweb2,dghweb3`) y estos apuntan a diferentes Docker puestos en los puertos (8086,8087,8088).

```

GNU nano 5.8 /etc/nginx/conf.d/dgh.conf
server {
    listen 80;
    server_name www.dghweb1.com;

    location / {
        proxy_pass http://localhost:8086;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

server {
    listen 80;
    server_name www.dghweb2.com;

    location / {
        proxy_pass http://localhost:8087;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

server {
    listen 80;
    server_name www.dghweb3.com;

    location / {
        proxy_pass http://localhost:8088;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

```

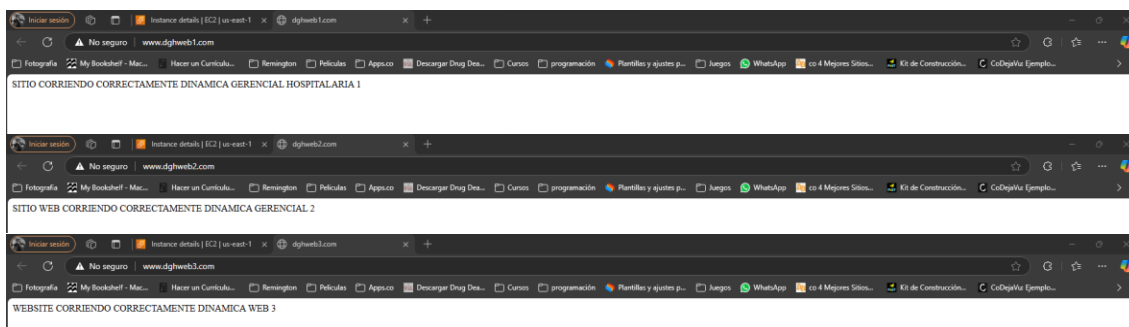
`proxy_set_header Host $host;`

Mantiene el encabezado y es útil para saber a qué dominio ingresó.

`proxy_set_header X-Real-IP $remote_addr;`

Envía información de la ip original de la app en el contenedor.

Luego de esto validamos que el archivo este correcto con el comando `nginx -t` y si esta todo correcto procedemos a reiniciar `nginx`



## Conclusiones

1. La aplicación de balanceadores de carga en nuestros servicios, bien sean en AWS u otras infraestructuras, nos permiten asegurar una correcta distribución de transacciones hacia diferentes instancias o servidores, esto garantiza un buen rendimiento y disponibilidad de los sitios web.
2. Los contenedores Docker dentro de las instancias, nos facilitan la administración y configuración de los servicios. Además, asegura que cada aplicación o sitio web se ejecute de manera independiente dentro de cada imagen.
3. Usando una función como Auto Scaling propia de AWS se garantiza una correcta adaptación de la infraestructura en la automatización dinámica, según las cargas del sistema o aplicación, aprovechando todos los recursos y reduciendo tiempos de carga, todo esto sin ajustes manuales.
4. La infraestructura que nos ofrece AWS en la configuración de las instancias EC2 con servicios web, nos da una gran ventaja y facilita la administración de estos mismos en la seguridad, la red, rendimiento y el almacenamiento. También la eficiente gestión de transacciones brindándonos herramientas como el Auto Scaling y los propios balanceadores de carga.
5. NGINX usado como proxy inverso y configurado con archivos internos es esencial si requerimos una buena gestión del tráfico hacia múltiples Docker dentro de una misma instancia. Esto permite la administración de dominios de una manera más organizada.
6. Las arquitecturas basadas en la nube, en este caso AWS, son una solución moderna que además de ser muy potentes, nos pueden ayudar con una buena escalabilidad y facilidad a la hora del despliegue de servicios o aplicaciones web. Esto junto con balanceadores de carga, contenedores Docker, NGINX y la herramienta Auto Scaling nos consolida infraestructuras con fácil acceso, disponibilidad constante, gran rendimiento y un buen mantenimiento. Esto además nos demuestra el avance y el progreso que cada día se obtiene con la computación en la nube y su importancia actual dentro de las organizaciones y empresas actuales.

## Referencias

Amazon Web Services. (2025). *Elastic Load Balancing documentation*.  
<https://aws.amazon.com/es/elasticloadbalancing/>.

Docker Inc. (2025). *What is a container?*  
<https://www.docker.com/resources/what-container/>.

Amazon Web Services. (2025). *What is AWS Auto Scaling?*  
[https://docs.aws.amazon.com/en\\_us/autoscaling/](https://docs.aws.amazon.com/en_us/autoscaling/).

Amazon Web Services. (2025). *Amazon EC2 documentation*.  
[https://docs.aws.amazon.com/en\\_us/ec2/](https://docs.aws.amazon.com/en_us/ec2/).

NGINX Inc. (2025). *NGINX as a reverse proxy*.  
<https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>.