

# Práctica 2 - Base de datos, balanceo y escalado

---

**Asignatura:** Computación en la Nube

**Fecha:** 24 de octubre de 2021

**Autor:** Francisco Javier López-Dufour Morales



# Índice

- Práctica 2 - Base de datos, balanceo y escalado
  - Índice
  - 1. Introducción
  - 2. Objetivos
  - 3. Actividades
    - 3.1. Despliegue y configuración de las instancias EC2
    - 3.2. Despliegue de un "load balancer"
    - 3.3. Despliegue de un "auto scaling group"
    - 3.4. Despliegue de una base de datos "relacional"
    - 3.5. Diagrama de la infraestructura desplegada
    - 3.6. Presupuesto y estimación de gasto de los recursos desplegados
  - 4. Conclusiones
  - 5. Bibliografía
  - 6. Anexos

# 1. Introducción

El objetivo de esta práctica es explorar y experimentar con las herramientas de balanceo de carga y escalado proporcionadas por AWS, aplicando los conceptos teóricos aprendidos en clase para desplegar una infraestructura web escalable y altamente disponible.

## 2. Objetivos

- Desplegar y configurar instancias EC2 que funcionen como servidores web independientes.
- Implementar un balanceador de carga que distribuya el tráfico de manera equitativa entre las instancias.
- Configurar un grupo de autoescalado para mantener la disponibilidad y ajustar la capacidad según la demanda.
- Desplegar una base de datos relacional utilizando Amazon RDS.
- Realizar un diagrama de la infraestructura y estimar los costos asociados a los recursos utilizados.

### 3. Actividades

#### 3.1. Despliegue y configuración de las instancias EC2

**Requisito:** Despliega dos instancia en EC2 con un servidor web que muestre una pagina similar pero que se pueda reconocer que es un servidor distinto. Estos servidores deben poder ser accedidos con un navegador desde fuera.

Configuración de las instancias EC2:

- AMI seleccionada
  - Amazon Linux 2023 AMI 2023.5.2 ([ami-0ebfd941bbafe70c6](#))
- Tipo de instancia
  - **t2.nano** (seleccionada por su equilibrio entre costo y rendimiento para pruebas)
- Grupo de seguridad:
  - **SG-SSH-HTTP-HTTPS** (configurado para permitir tráfico SSH, HTTP y HTTPS)

Configuración del grupo de seguridad:

Nombre	ID de la regla del grupo de seguridad	Versión de IP	Tipo	Protocolo	Intervalo de puertos	Origen	Descripción
	sgr-07861fea6d1e6cb9e	IPv4	HTTPS	TCP	443	0.0.0.0/0	
	sgr-002bcf0775e6ab4ee	IPv4	SSH	TCP	22	0.0.0.0/0	
	sgr-0a8dce6066c4ab592	IPv4	TCP personalizado	TCP	0	0.0.0.0/0	
	sgr-048cf1efa02df6e89	IPv4	HTTP	TCP	80	0.0.0.0/0	

Instancias EC2 ejecutándose:

Se inició correctamente la detención de i-042e190818ee2811a,i-022b352c4271ee914

Instancias (2/4) Información

Última actualización Hace less than a minute

Conectar

Estado de la instancia

Acciones

Lanzar instancias

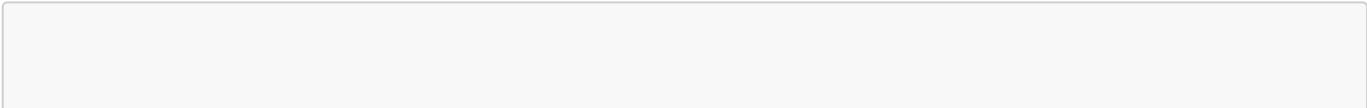
Buscar Instancia por atributo o etiqueta (case-sensitive)

Todos los e...

	Name	ID de la instancia	Estado de la i...	Tipo de inst...	Comprobación de estado	Estado de la al...
<input checked="" type="checkbox"/>	my-server-1	i-09e36353a518a84e7	En ejecución	t2.nano	Inicializando	Ver alarmas
<input checked="" type="checkbox"/>	my-server-2	i-0a53fd7b805eb00bc	En ejecución	t2.nano	Inicializando	Ver alarmas
<input type="checkbox"/>	MyWebServer...	i-042e190818ee2811a	Deteniéndose	t2.micro	2/2 comprobaciones superadas	Ver alarmas
<input type="checkbox"/>	SSH_Gate	i-022b352c4271ee914	Deteniéndose	t2.micro	2/2 comprobaciones superadas	Ver alarmas

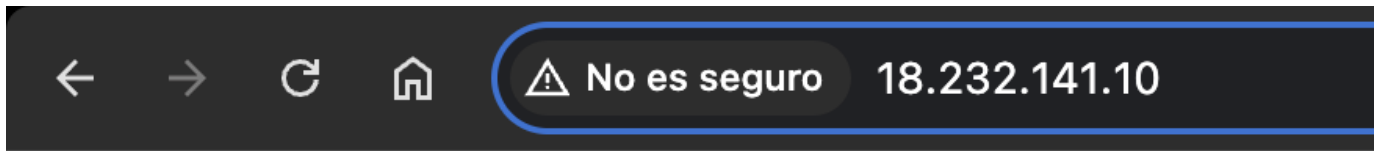
Pasos realizados

1. **Conexión a las instancias:** Utilizamos SSH para conectarnos a cada instancia EC2.
2. **Ejecución del script de configuración:** Subimos y ejecutamos el script **setup\_server.sh** que automatiza la instalación de Node.js, configuración del servidor Express y Nginx como proxy inverso.



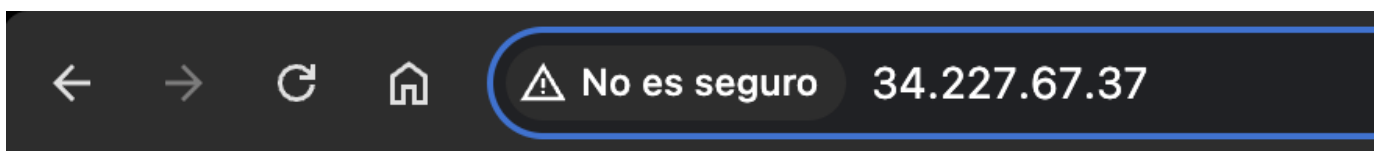
```
chmod +x setup_server.sh
sudo ./setup_server.sh
```

3. **Verificación del servidor web:** Accedemos a la IP pública de cada instancia desde un navegador para comprobar que el servidor web está operativo y muestra el hostname, permitiendo distinguir entre ambos servidores.



# Express Server on AWS EC2

Hostname: ip-172-31-45-209.ec2.internal



# Express Server on AWS EC2

Hostname: ip-172-31-33-205.ec2.internal

## 3.2. Despliegue de un "load balancer"

**Requisito:** Despliega un "load balancer" que distribuya las peticiones entre los dos servidores a partes iguales.

Antes de nada, es necesario crear un grupo de destino (target group) para los servidores web.

Desde el menu de EC2, accedemos a "Grupos de destino".

Pasos realizados:

1. Creación del grupo de destino (**Target Group**)\*\*:
  - Tipo de destino: Instancias
  - Nombre: `lb-pr2-p2-tg`
  - Protocolo: HTTP (Puerto 80)
  - Configuración de comprobación de estado: Ruta /

## Revisar destinos



### Destinos (2)

Eliminar todos los pendientes

 Filtrar destinos

☐ Mostrar solo pendientes

< 1 > 

ID de instancia ▾	Nombre ▾	Puerto ▾	Estado ▾
i-09e36353a518a84e7	my-server-1	80	 Ejecutando
i-0a53fd7b805eb00bc	my-server-2	80	 Ejecutando

✓ Se creó correctamente el grupo de destinos: **lb-pr2-p2-tg**. La detección de anomalías se aplica automáticamente a todos los destinos registrados. El estado se puede ver en la pestaña **Destinos**.


[EC2](#) > [Grupos de destino](#) > lb-pr2-p2-tg

### lb-pr2-p2-tg

Acciones ▾

#### Detalles

 `arn:aws:elasticloadbalancing:us-east-1:491250998585:targetgroup/lb-pr2-p2-tg/46fe0242fff8cb84`

Tipo de destino Instancia	Protocolo : Puerto HTTP: 80	Versión del protocolo HTTP1	VPC <a href="#">vpc-01678e3128d2f6638</a> 
Tipo de dirección IP IPv4	Balanceador de carga <a href="#">Ninguno asociado</a>		

2 Destinos totales	 0 En buen estado	 0 En mal estado	 2 Sin utilizar	 0 Inicial	 0 Vaciado
	0 Anómalo				

#### ► Distribución de destinos por zona de disponibilidad (AZ)

Seleccione los valores de esta tabla para ver los filtros correspondientes aplicados a la tabla Destinos registrados que aparece a continuación.

## 2. Creación del **balanceador de carga** de aplicaciones (ALB):

- Nombre: **lb-pr2-p2**
- Esquema: Expuesto a Internet
- Subredes: **us-east-1a**, **us-east-1b**
- Grupos de seguridad: **SG-SSH-HTTP-HTTPS**
- Configuración de listeners: Protocolo HTTP en el puerto 80, encaminado al grupo de destino **lb-pr2-p2-tg**

## Resumen

Revise y confirme las configuraciones. [Costo estimado](#)

### Configuración básica [Editar](#)

lb-pr2-p2

- Expuesto a Internet
- IPv4

### Grupos de seguridad [Editar](#)

- SG-Ssh-HTTP-HTTPS  
[sg-04ea111d0258785f7](#)

### Mapeo de red [Editar](#)

VPC [vpc-01678e3128d2f6638](#)

- us-east-1a  
[subnet-08b194394f7f92fce](#)
- us-east-1b  
[subnet-0051ffccb5958745a](#)

### Agentes de escucha y direccionamiento [Editar](#)

- HTTP:80 valor predeterminado para [lb-pr2-p2-tg](#)

### Integraciones de servicios [Editar](#)

AWS WAF: Ninguno

AWS Global Accelerator: Ninguno

### Etiquetas [Editar](#)

Ninguno

### Atributos

Algunos atributos predeterminados se aplicarán al balanceador de carga. Puede verlos y editarlos después de crear el balanceador de carga.

## Balanceador de carga: lb-pr2-p2

us-east-1 | 491250998585



## Verificación

Utilizamos `curl` para enviar peticiones al DNS del balanceador de carga y observamos que las respuestas provienen de ambas instancias, confirmando la distribución equitativa del tráfico.

```
lb-pr2-p2-1198883516.us-east-1.elb.amazonaws.com
```

Comprobamos que el balanceador de carga distribuye las peticiones entre los servidores web.

```
(base) fran.@MacBook ~ % curl lb-pr2-p2-1198883516.us-east-1.elb.amazonaws.com
<h1>Express Server on AWS EC2</h1><p>Hostname: ip-172-31-45-209.ec2.internal</p>%

(base) fran.@MacBook ~ % curl lb-pr2-p2-1198883516.us-east-1.elb.amazonaws.com
<h1>Express Server on AWS EC2</h1><p>Hostname: ip-172-31-33-205.ec2.internal</p>%
```

## 3.3. Despliegue de un "auto scaling group"

**Requisito:** Configurar un Auto Scaling Group (ASG) con un mínimo de 1 instancia y un máximo de 2, utilizando un template de instancia.

Para crear un Auto Scaling Group (ASG) necesitamos un Launch Template.

Desde el menu de EC2, accedemos a EC2 > ... > "Crear plantilla de lanzamiento".



## 1. Crear un nuevo **Launch Template**:

- Nombre: **lt-pr2-p2**
- Descripción: Launch template for EC2 instances
- Versión: 1
- Tipo de instancia: **t2.nano**
- AMI: **Amazon Linux 2023 AMI 2023.5.2**
- Tipo de almacenamiento: **EBS**
- Tamaño de almacenamiento: **8 GiB**
- Grupo de seguridad: **SG-SSH-HTTP-HTTPS**
- Clave de par:
- Tipo de red: **default**
- Datos de usuario: **setup\_server.sh** (script de configuración de la instancia)

**lt-pr2-p2 (lt-0754f1f098e5cd448)**

Acciones ▼Eliminar plantilla

**Detalles de la plantilla de lanzamiento**

ID de la plantilla de lanzamiento lt-0754f1f098e5cd448	Nombre de la plantilla de lanzamiento lt-pr2-p2	Versión predeterminada 1	Propietario arn:aws:sts::491250998585:assumed-role/voclabs/user3502361=francisco_l
---	--	-----------------------------	---

Detalles

Versiones

Etiquetas de la plantilla

**Detalles de la versión de la plantilla de lanzamiento**

Acciones ▼Eliminar versión de plantilla

Versión 1 (predeterminado) ▼	Descripción Launch template for EC2 instances	Fecha de creación 2024-10-14T23:44:43.000Z	Creada por arn:aws:sts::491250998585:assumed-role/voclabs/user3502361=francisco_l
---------------------------------	--	---	--

Detalles de la instancia

Almacenamiento

Etiquetas de recursos

Interfaces de red

Detalles avanzados

ID de AMI ami-0ebfd941bbafe70c6	Tipo de instancia t2.nano	Zona de disponibilidad -	Nombre del par de claves vockey
Grupos de seguridad -	ID de grupo de seguridad sg-04ea111d0258785f7		

## 2. Una vez creado el Launch Template, creamos un **Auto Scaling Group**.

- Nombre: **asg-pr2-p2**
- Plantilla de lanzamiento: **lt-pr2-p2**
- Configuración de grupo:
  - Capacidad deseada: 1 instancia
  - Capacidad mínima: 1 instancia
  - Capacidad máxima: 2 instancias
  - Redes: **default**
  - Subredes: **us-east-1a, us-east-1b**
  - Grupo de destino: **lb-pr2-p2-tg**
  - Grupo de equilibrador de carga: **lb-pr2-p2**
  - Configuración de escalado:
    - Escalado de destino:
      - Grupo de destino: **lb-pr2-p2-tg** (integración con el balanceador de carga)

- Política de mantenimiento de instancias:
  - Comporamiento mixto: Sin política.

asg-pr2-p2

Detalles

Actividad

Escalado automático


Administración de instancias

Monitoreo

Actualización de instancias




Detalles del grupo

Editar

Nombre del grupo de Auto Scaling asg-pr2-p2	Capacidad deseada 1	Tipo de capacidad deseado Unidades (número de instancias)	Nombre de recurso de Amazon (ARN)  arn:aws:autoscaling:us-east-1:491250998585:autoScalingGroup:18d06861-a703-4a15-b95b-1ff2fa292162:autoScalingGroupName/asg-pr2-p2
Fecha de creación Tue Oct 15 2024 00:51:44 GMT+0100 (hora de verano de Europa occidental)	Capacidad mínima 1	Estado -	
	Capacidad máxima 1		

Plantilla de lanzamiento

Editar

Plantilla de lanzamiento  lt-0754f1f098e5cd448 lt-pr2-p2	ID de AMI  ami-0ebfd941bbafe70c6	Tipo de instancia t2.nano	Propietario arn:aws:sts::491250998585:assumed-role/voclabs/user3502361=francisco_l
Versión Default	Grupos de seguridad -	ID de grupos de seguridad  sg-04ea111d0258785f7	Hora de creación Tue Oct 15 2024 00:44:43 GMT+0100 (hora de verano de Europa occidental)
Descripción Launch template for EC2 instances	Almacenamiento (volumenes) -	Nombre del par de claves vockey	Solicitar instancias de spot No

Balance de carga

Editar

Grupos de destino del balanceador de carga <a href="#">lb-pr2-p2-tg</a>	Balanceadores de carga clásicos -
--	--------------------------------------

Opciones de integración de VPC Lattice

Editar

Grupos de destino de VPC Lattice -
---------------------------------------

Comprobaciones de estado

Editar

Tipo de comprobación de estado EC2, ELB	Periodo de gracia de la comprobación de estado 300
--	---

Política de mantenimiento de instancias

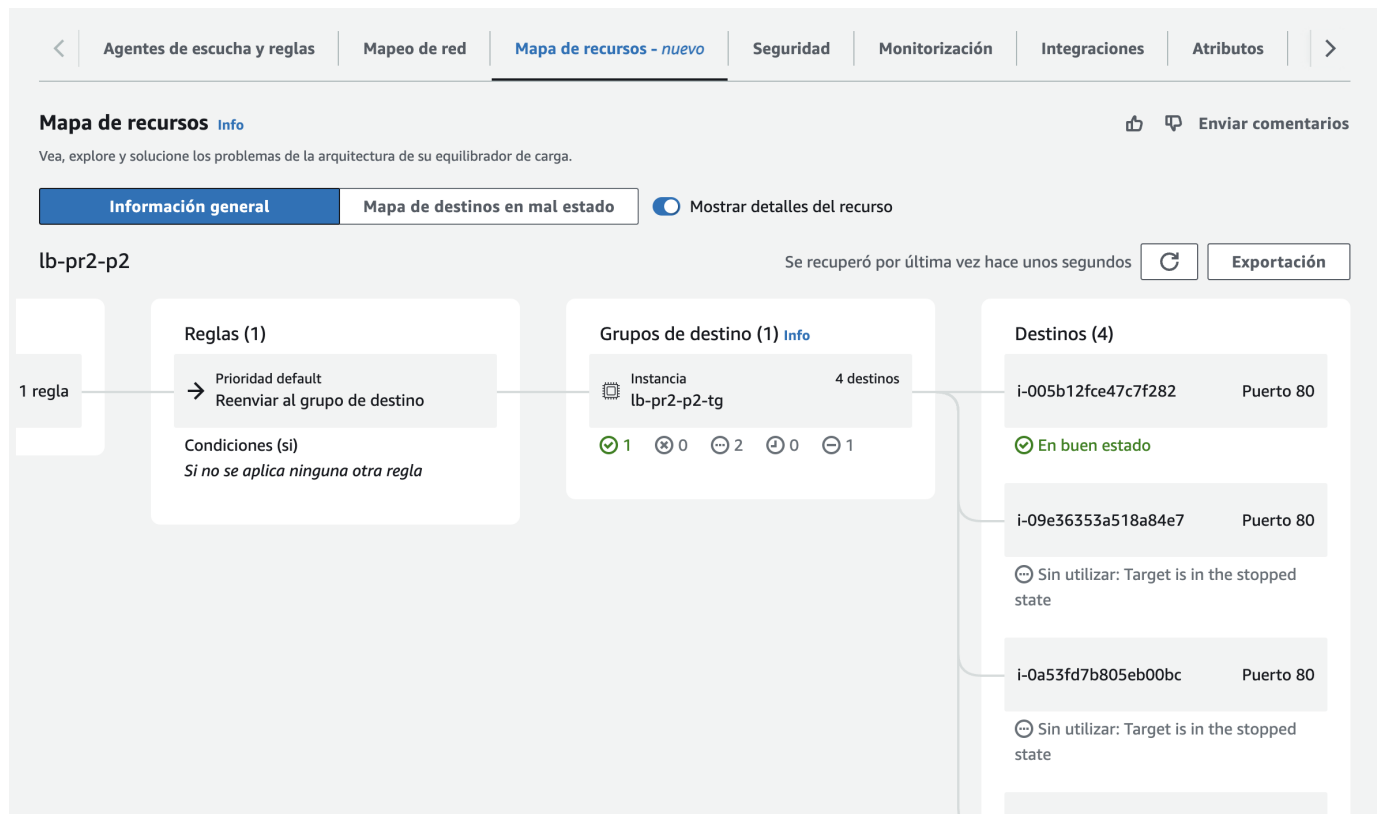
Editar

Comportamiento de reemplazo Sin política	Porcentaje mínimo de buen estado -	Porcentaje máximo de buen estado -
---	---------------------------------------	---------------------------------------

Una vez creado el Auto Scaling Group, detenemos todas las instancias y comprobamos que mantiene al menos una instancia viva de forma automática.

```
(base) franciscoj. MacBook ~ % curl lb-pr2-p2-1198883516.us-east-1.elb.amazonaws.com
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Francisco Javier López-Dufour Morales
```

Y comprobamos el estado del Load Balancer.



### 3.4. Despliegue de una base de datos "relacional"

**Requisito:** Desplegar una base de datos de elección dentro de AWS, demostrar su funcionamiento y estimar el costo de utilización.

AWS ofrece varios servicios de bases de datos relacionales mediante Amazon RDS (Relational Database Service).

En este caso, vamos a desplegar una base de datos MySQL.

Desde el menu de RDS, accedemos a "Bases de datos" y creamos una nueva base de datos.

Pasos realizados:

1. Creación de una instancia de base de datos en Amazon RDS:

- Motor: MySQL (versión 8.0.25)
- Tipo de instancia: **db.t4g.micro** (dentro del Free Tier)
- Almacenamiento: **20 GiB SSD**
- Configuración de credenciales: Usuario admin y una contraseña segura.

2. Configuración de seguridad:

- Modificamos el grupo de seguridad para permitir conexiones desde las instancias EC2 y nuestra IP local para administración.

3. Verificación:

- Conectamos a la base de datos desde nuestra máquina local utilizando el cliente MySQL:

```
mysql -h pr2-p2-db.cntejlv13d16.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 8.0.35 Source distribution

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Resumen:

pr2-p2-db

Modificar

Acciones

Resumen

Identificador de base de datos pr2-p2-db CPU -	Estado ⌚ Backing-up Clase db.t4g.micro	Rol Instancia Actividad actual	Motor MySQL Community Región y AZ us-east-1a	Recomendaciones
---	---	--------------------------------------	---	-----------------

<

Conectividad y seguridad

Supervisión

Registros y eventos

Configuración

Integraciones sin extracción, transformación y carga (ETL)

>

Conectividad y seguridad

Punto de enlace y puerto Punto de enlace pr2-p2-db.cntejlv13d16.us-east-1.rds.amazonaws.com Puerto 3306	Redes Zona de disponibilidad us-east-1a VPC vpc-01678e3128d2f6638 Grupo de subredes default-vpc-01678e3128d2f6638 Subredes	Seguridad Grupos de seguridad de la VPC default (sg-0f66ed311d4e085b1) Activo SG-SSH-HTTP-HTTPS (sg-04ea111d0258785f7) Activo Accesible públicamente Sí
---	---	--

Coste de utilización:

Resumen de la estimación		
Costo inicial	Costo mensual	Costo total de 12 months
0,00 USD	27,96 USD	335,52 USD
		Incluye el costo inicial

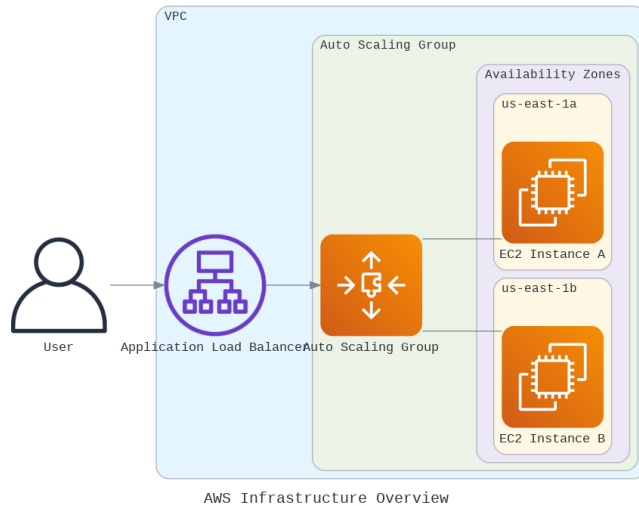
Estimación detallada

Nombre	Grupo	Región	Costo inicial	Costo mensual
Amazon RDS for MySQL	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	27,96 USD
Estado: -				
Descripción: pr2-p2-db				
Resumen de la configuración: Cantidad de almacenamiento (20 GB), Almacenamiento para cada instancia RDS (SSD de uso general (gp2)), Nodos (1), Tipo de instancia (db.t4g.micro), Utilización (solo bajo demanda) (100 %Utilized/Month), Opción de implementación (Multi-AZ), Modelo de precios (OnDemand)				

- Costo mensual: \$27.96.

- Costo anual: \$335.52.

### 3.5. Diagrama de la infraestructura desplegada



### 3.6. Presupuesto y estimación de gasto de los recursos desplegados

El costo mensual estimado de los recursos desplegados es el siguiente:

- **Instancias EC2:**
  - 2 instancias **t2.nano** en la región Norte de Virginia (**us-east-1**).
  - Costo mensual estimado: \$4.23.
  - Costo anual estimado: \$50.81.
- **Auto Scaling Group (ASG):**
  - 1 grupo de autoescalado en la región Norte de Virginia (**us-east-1**).
  - Costo mensual estimado: \$0.00.
  - Costo anual estimado: \$0.00.
- **Application Load Balancer (ALB):**
  - 1 balanceador de carga en la región Norte de Virginia (**us-east-1**).
  - Costo mensual estimado: \$60.23.
  - Costo anual estimado: \$722.76.

Costo total mensual estimado: \$64.46.

Costo total anual estimado: \$773.52.

*Nota:* Esta estimación es un cálculo preliminar basado en el uso constante de los recursos mencionados. El costo real puede variar en función del uso real, la configuración regional y otros factores. Además, no se incluyen los impuestos aplicables.

## 4. Conclusiones

En esta práctica, hemos logrado desplegar una infraestructura web escalable y altamente disponible utilizando los servicios de AWS. La experiencia nos permitió comprender cómo integrar diferentes componentes como EC2, balanceadores de carga, grupos de autoescalado y bases de datos relacionales para construir soluciones robustas en la nube.

## 5. Bibliografía

- [Amazon EC2](#)
- [Amazon RDS](#)
- [Amazon Web Services In Action](#)

## 6. Anexos

```
#!/bin/bash

# Update the system
sudo yum update -y

# Install Node Version Manager (nvm)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash

# Load nvm
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"

# Install Node.js
nvm install 16

# Create a new directory for the application
mkdir ~/myapp
cd ~/myapp

# Initialize a new Node.js project and install Express
npm init -y
npm install express

# Create a simple Express server
cat << EOF > app.js
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('<h1>Express Server on AWS EC2</h1><p>Hostname: ' +
    require('os').hostname() + '</p>');
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
EOF

# Install PM2 globally
npm install pm2 -g

# Start the Express server with PM2
pm2 start app.js

# Configure PM2 to start on system boot
pm2 startup
pm2 save

# Install and configure nginx as a reverse proxy
sudo yum install nginx

# Configure nginx
sudo tee /etc/nginx/conf.d/myapp.conf > /dev/null <<EOF
```



```
server {
    listen 80;
    server_name _;

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade \${http_upgrade};
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host \${host};
        proxy_cache_bypass \${http_upgrade};
    }
}
EOF
```

```
# Remove the default nginx configuration
sudo rm /etc/nginx/conf.d/default.conf
```

```
# Start and enable nginx
sudo systemctl start nginx
sudo systemctl enable nginx
```

```
# Print the public IP address
echo "Setup complete. Your server's public IP address is:"
curl -s http://169.254.169.254/latest/meta-data/public-ipv4
```

```
echo "You can access your Express server by visiting this IP address in your web browser."
```