

Práctica 2 - Base de datos, balanceo y escalado

Francisco Javier López-Dufour Morales

Índice

1. Introducción	2
2. Objetivos	3
3. Actividades	4
3.1. Despliegue y configuración de las instancias EC2	4
3.2. Despliegue de un “load balancer”	6
3.3. Despliegue de un “auto scaling group”	8
3.4. Despliegue de una base de datos “relacional”	10
3.5. Diagrama de la infraestructura desplegada	12
3.6. Presupuesto y estimación de gasto de los recursos desplegados	13
4. Conclusiones	14
5. Bibliografía	15
6. Anexos	16

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

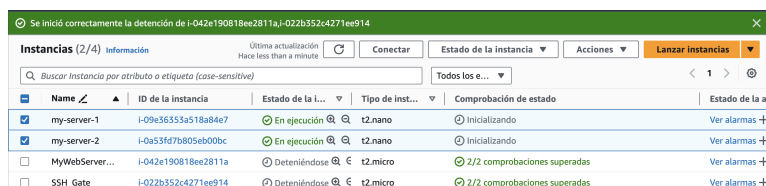
3.1. Despliegue y configuración de las instancias EC2

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)



Seleccionar	Nombre	ID de la instancia	Estado de la instancia	Tipo de instancia	Comprobación de estado	Estado de la alarma
<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 +

Figura 1: Panel de control EC2 mostrando las instancias en ejecución

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.

```
1 chmod +x setup_server.sh
2 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.



Figura 2: Servidor web 1 en funcionamiento



Figura 3: Servidor web 2 en funcionamiento

3.2. Despliegue de un “load balancer”

Requisito

Despliega un “load balancer” que distribuya las peticiones entre los dos servidores a partes iguales.

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 /

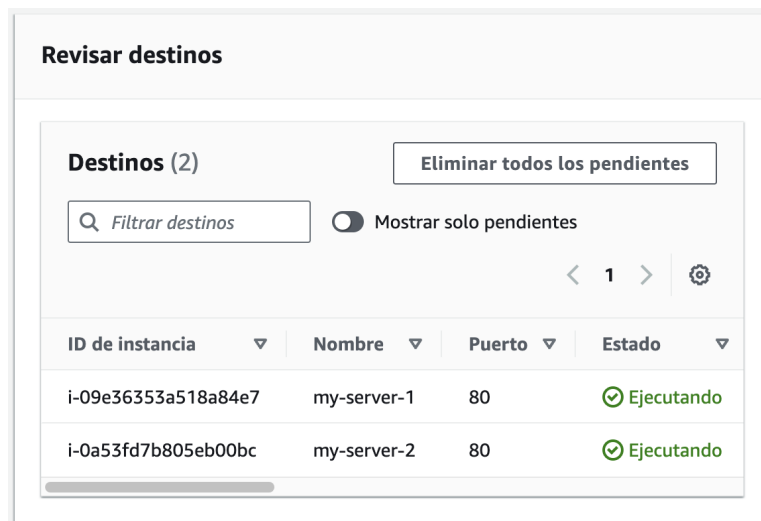


Figura 4: Grupo de destino

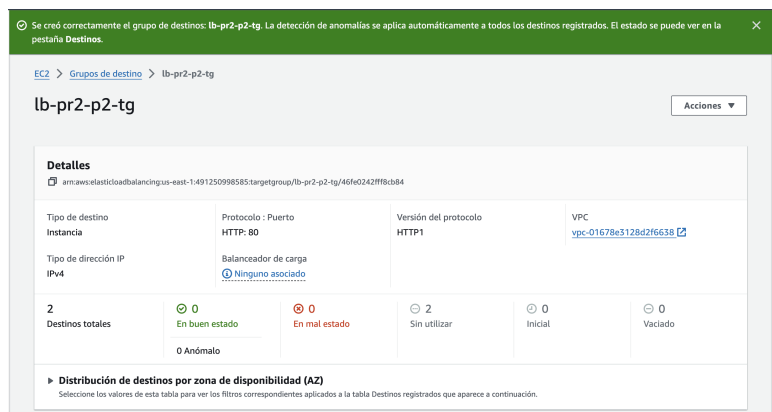


Figura 5: Detalles del grupo de destino

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



Figura 6: Detalles del balanceador de carga

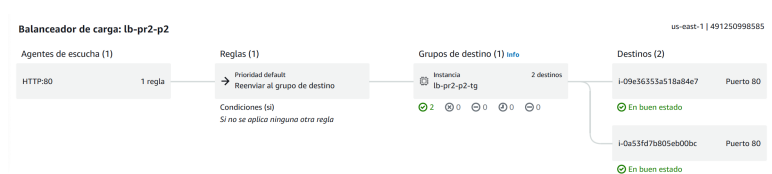


Figura 7: Detalles adicionales del balanceador de carga

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.

Configuración

1. Crear Launch Template:

- Nombre: `lt-pr2-p2`
- Tipo de instancia: `t2.nano`
- AMI: Amazon Linux 2023 AMI 2023.5.2
- Almacenamiento: 8 GiB EBS
- Grupo de seguridad: `SG-SSH-HTTP-HTTPS`

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/user:3502361-francisco_...
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/user:3502361-francisco_...
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-04ea11d0258785f7		

Figura 8: Plantilla de lanzamiento

2. Crear Auto Scaling Group:

- Nombre: `asg-pr2-p2`
- Capacidad: Min 1, Max 2 instancias
- Subredes: `us-east-1a`, `us-east-1b`
- Integración con Load Balancer: `lb-pr2-p2-tg`

asg-pr2-p2			
<div>DetallesActividadEscalado automáticoAdministración de instanciasMonitoreoActualización de instancias</div>			
Detalles del grupo <div>Editar</div>			
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-11f2fa292162: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 <div>Editar</div>			
Plantilla de lanzamiento la-0754f180985c4448 lb-pr2-p2	ID de AMI ami-0ebfd941bbafe70dc	Tipo de instancia t2.nano	Propietario arn:aws:sts::491250998585:assumed-role/vordlabs/user3502361-francisco_j
Versión Default	Grupos de seguridad -	ID de grupos de seguridad sg-04ea1160258785f7	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

Figura 9: Grupo de auto escalado

Balance de carga <div>Editar</div>			
Grupos de destino del balanceador de carga lb-pr2-p2-tg	Balanceadores de carga clásicos -		
Opciones de integración de VPC Lattice <div>Editar</div>			
Grupos de destino de VPC Lattice -			
Comprobaciones de estado <div>Editar</div>			
Tipo de comprobación de estado EC2, ELB	Periodo de gracia de la comprobación de estado 300		
Política de mantenimiento de instancias <div>Editar</div>			
Comportamiento de reemplazo Sin política	Porcentaje mínimo de buen estado -	Porcentaje máximo de buen estado -	

Figura 10: Detalles adicionales del grupo de auto escalado

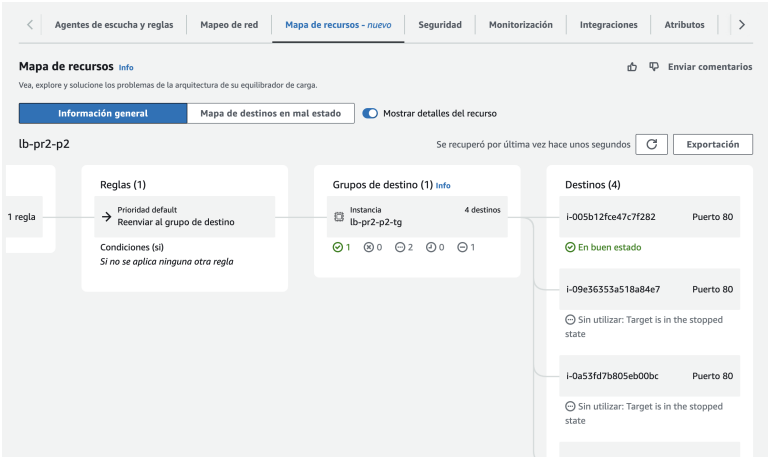


Figura 11: Estado de las instancias en el balanceador de carga

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.

Configuración

- Motor: MySQL (versión 8.0.25)
- Tipo de instancia: `db.t4g.micro`
- Almacenamiento: 20 GiB SSD
- Costo mensual estimado: \$27.96
- Costo anual estimado: \$335.52

The screenshot displays the AWS RDS console for an instance named 'pr2-p2-db'. The interface includes a top navigation bar with 'Modificar' and 'Acciones' buttons. Below this is a 'Resumen' (Summary) section with a table of instance details:

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

Below the summary is a horizontal navigation bar with tabs: 'Conectividad y seguridad' (selected), 'Supervisión', 'Registros y eventos', 'Configuración', and 'Integraciones sin extracción, transformación y carga (ETL)'. The 'Conectividad y seguridad' section is expanded, showing three columns:

Punto de enlace y puerto	Redes	Seguridad
Punto de enlace pr2-p2-db.cntejlv3d16.us-east-1.rds.amazonaws.com	Zona de disponibilidad us-east-1a	Grupos de seguridad de la VPC default (sg-0f66ed311d4e085b1) ● Activo
Puerto 3306	VPC vpc-01678e3128d2f6638	SG-SSH-HTTP-HTTPS (sg-04ea111d0258785f7) ● Activo
	Grupo de subredes default-vpc-01678e3128d2f6638	Accesible públicamente Sí
	Subredes	

Figura 12: Base de datos RDS

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)				

Figura 13: Costo estimado de la base de datos RDS

3.5. Diagrama de la infraestructura desplegada

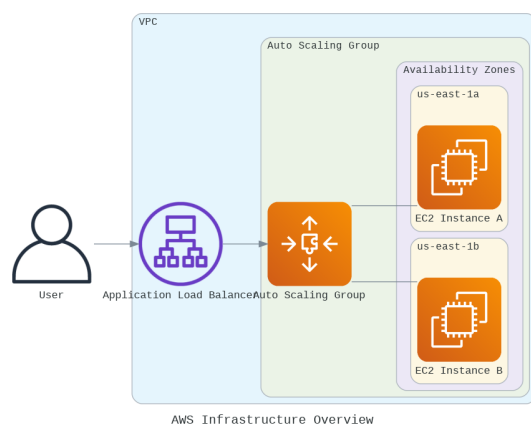


Figura 14: Diagrama de la infraestructura

3.6. Presupuesto y estimación de gasto de los recursos desplegados

- **Instancias EC2:**
 - 2 instancias `t2.nano`
 - Costo mensual: \$4.23
 - Costo anual: \$50.81
- **Auto Scaling Group (ASG):**
 - Sin costo adicional
- **Application Load Balancer (ALB):**
 - Costo mensual: \$60.23
 - Costo anual: \$722.76
- **Total:**
 - Costo mensual: \$64.46
 - Costo anual: \$773.52

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: <https://aws.amazon.com/ec2/>
- Amazon RDS: <https://aws.amazon.com/rds/>
- Amazon Web Services In Action (Manning Publications)

6. Anexos

Listing 1: setup_server.sh

```
1  #!/bin/bash
2
3  # Update the system
4  sudo yum update -y
5
6  # Install Node Version Manager (nvm)
7  curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0
   .39.3/install.sh | bash
8
9  # Load nvm
10 export NVM_DIR="$HOME/.nvm"
11 [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
12
13 # Install Node.js
14 nvm install 16
15
16 # Create a new directory for the application
17 mkdir ~/myapp
18 cd ~/myapp
19
20 # Initialize a new Node.js project and install Express
21 npm init -y
22 npm install express
23
24 # Create a simple Express server
25 cat << EOF > app.js
26 const express = require('express');
27 const app = express();
28 const port = 3000;
29
30 app.get('/', (req, res) => {
31   res.send('<h1>Express_Server_on_AWS_EC2</h1><p>Hostname
   :_ + require('os').hostname() + '</p>');
32 });
33
34 app.listen(port, () => {
35   console.log(`Server running at http://localhost:${
   port}`);
36 });
37 EOF
38
39 # Install PM2 globally
40 npm install pm2 -g
41
42 # Start the Express server with PM2
```



```

43 pm2 start app.js
44
45 # Configure PM2 to start on system boot
46 pm2 startup
47 pm2 save
48
49 # Install and configure nginx as a reverse proxy
50 sudo yum install nginx
51
52 # Configure nginx
53 sudo tee /etc/nginx/conf.d/myapp.conf > /dev/null <<EOF
54 server {
55     listen 80;
56     server_name _;
57
58     location / {
59         proxy_pass http://127.0.0.1:3000;
60         proxy_http_version 1.1;
61         proxy_set_header Upgrade \$http_upgrade;
62         proxy_set_header Connection 'upgrade';
63         proxy_set_header Host \$host;
64         proxy_cache_bypass \$http_upgrade;
65     }
66 }
67 EOF
68
69 # Remove the default nginx configuration
70 sudo rm /etc/nginx/conf.d/default.conf
71
72 # Start and enable nginx
73 sudo systemctl start nginx
74 sudo systemctl enable nginx
75
76 # Print the public IP address
77 echo "Setup complete. Your server's public IP address is:
78 "
79 curl -s http://169.254.169.254/latest/meta-data/public-
80     ipv4
81
82 echo "You can access your Express server by visiting this
83     IP address in your web browser."

```