Proyecto CloudGram (Docker + EC2 + RDS + Bastion)

Proyecto CloudGram - Despliegue con Docker y AWS

Introducción

Hola, soy Juan. Este documento es un resumen completo, en primera persona, del proyecto llamado CloudGram. El objetivo de este proyecto fue desplegar una aplicación web Flask usando contenedores Docker sobre una infraestructura clásica de AWS: EC2 para el servidor web, RDS para la base de datos PostgreSQL y un Bastion Host para conectarme de forma segura. Todo el proyecto fue desplegado con Terraform.

1 Infraestructura desplegada

- VPC personalizada
- Subredes públicas y privadas (2 y 2)
- Internet Gateway y NAT Gateway
- Tabla de rutas públicas y privadas
- Bastion Host (EC2 Amazon Linux 2)
- Web Server (EC2 Ubuntu 24.04 con Docker)
- RDS PostgreSQL
- Grupos de seguridad: acceso SSH desde mi IP, conexión segura entre Bastion y RDS, y entre Web Server y RDS

2 Proceso paso a paso

1. Escribí el código Terraform para desplegar toda la infraestructura descrita.
2. Generé una clave SSH pública y privada para acceder a las instancias.
3. Usé Terraform para importar la clave a AWS y lanzar las instancias EC2.
4. Verifiqué conectividad desde PowerShell y accedí por SSH a las instancias (Web y Bastion).
5. Instalé Docker en la instancia Ubuntu (Web Server).
6. Preparé el proyecto Flask localmente, creé un Dockerfile y comprimí los archivos en .zip.
7. Usé SCP para transferir el proyecto a la EC2 del Web Server y descomprimí allí.
8. Construí la imagen Docker: `docker build -t cloudgram-app .`

Proyecto CloudGram (Docker + EC2 + RDS + Bastion)

9. Ejecuté el contenedor con: `docker run -d -p 5000:5000 cloudgram-app`
10. Desde la app Flask se configuró la conexión directa a RDS usando su endpoint.
11. Verifiqué conectividad a la base de datos desde el Web Server y Bastion Host.
12. Ajusté las reglas del Security Group para permitir tráfico desde Web Server a RDS.
13. Accedí al Bastion, instalé PostgreSQL client y conecté vía `psql` al RDS.
14. Creé la base de datos `cloudgram` y la tabla `users`.
15. Posteriormente creé también la tabla `publicaciones` para almacenar comentarios e imágenes.
16. Después de actualizar el `app.py` en local, volví a subirlo, reconstruí la imagen y reinicié el contenedor.
17. Finalmente verifiqué que la app funcionaba correctamente: se pueden registrar usuarios, iniciar sesión, subir comentarios e imágenes.

Conclusión

Este proyecto demuestra cómo desplegar una aplicación contenida en Docker sobre una infraestructura tradicional de AWS. Aprendí a integrar varios servicios (EC2, RDS, SSH, grupos de seguridad, VPC, subredes y más) y asegurar que todos se comuniquen correctamente. También consolidé mis conocimientos en Docker, gestión de imágenes y contenedores, y el uso de Terraform para automatizar todo el proceso.