Ejercicio práctico en escenario real

Este es un **ejercicio práctico y completo** para aprender **Infrastructure as Code (IaC) con Terraform**, integrando:

- AWS (como proveedor de infraestructura)
- Un aplicativo web sencillo (por ejemplo, Node.js o Python Flask)
- Jenkins para la automatización (CI/CD)

🧪 Ejercicio Práctico: Despliegue de una Aplicación Web con Terraform, AWS y Jenkins

📘 Escenario

Eres parte del equipo DevOps de una empresa emergente que está lanzando su primer MVP: una aplicación web de registro de usuarios desarrollada en Node.js. El equipo necesita desplegar esta aplicación en la nube (AWS), asegurando que la infraestructura sea reproducible y gestionada como código (IaC). Además, quieren automatizar el proceso de despliegue con Jenkins.

Tu misión es **diseñar, implementar y automatizar** toda la infraestructura y el pipeline de despliegue usando **Terraform** y **Jenkins**.

🎯 Objetivos del Ejercicio

1. **Crear la infraestructura en AWS con Terraform:**
    - Una **VPC** con subred pública.
    - Un **Security Group** que permita tráfico HTTP y SSH.
    - Una **EC2** con Amazon Linux 2.
    - Una **instancia RDS (opcional)** para base de datos MySQL o PostgreSQL.
    - Un **Load Balancer (opcional)** si decides escalar más allá de una instancia.
2. **Provisionar la aplicación web:**
    - Instalar Node.js (o Python Flask).
    - Subir el código de la aplicación a la EC2.
    - Configurar Nginx o usar pm2 para mantener la app en ejecución.
    - (Opcional) Configurar HTTPS con Let's Encrypt.

3. **Automatizar con Jenkins:**
    - Crear un **pipeline declarativo (Jenkinsfile)**.
    - Al hacer push a la rama main, debe:
        - Ejecutar pruebas básicas (si existen).
        - Aplicar cambios en Terraform (infraestructura).
        - Conectar vía SSH a la EC2 para hacer pull de código y reiniciar la app.

🧱 Requerimientos

- Terraform v1.0+
- Jenkins instalado (puede ser en otra EC2 o local)
- AWS CLI configurado
- GitHub o GitLab para el repositorio del código
- Docker (opcional para empaquetar Jenkins o la app)
- Lenguaje backend (Node.js, Flask, etc.)

📁 Estructura sugerida del proyecto

```
web-app-iac/
├── app/
│   ├── index.js
│   ├── package.json
├── terraform/
│   ├── main.tf
│   ├── variables.tf
│   ├── outputs.tf
│   └── user_data.sh
├── jenkins/
│   └── Jenkinsfile
└── README.md
```

📌 Tips para Implementación

- Usa módulos en Terraform para separar recursos (red, compute, DB).
- Para provisionar la app, puedes usar user_data en EC2 o herramientas como Ansible.
- Configura el provider AWS con variables para región, credenciales, etc.
- Haz pruebas con terraform plan y terraform apply en entorno de pruebas.
- Usa remote-exec provisioners o mejor aún, pipelines en Jenkins para deploy continuo.

✅ Criterios de Éxito

- La aplicación web es accesible desde un navegador usando la IP pública del EC2 o el Load Balancer.
- Toda la infraestructura se puede destruir y volver a crear con terraform destroy y terraform apply.
- El pipeline de Jenkins ejecuta el flujo completo tras un git push.

📝 README.md

Incluye:

- Cómo generar claves SSH
- Cómo configurar AWS CLI
- Cómo correr terraform apply
- Cómo configurar Jenkins con tus credenciales