📝 Plan de Demo: Seguridad de IaC con Snyk y Terraform

1. Objetivo del Demo

Demostrar cómo las herramientas SAST, específicamente **Snyk IaC**, identifican vulnerabilidades de configuración en archivos de Infraestructura como Código (IaC) *antes* de que la infraestructura sea desplegada en la nube.

Principio Reforzado: Shift Left (Mover a la Izquierda): Encontrar fallos de seguridad en la fase de código (IaC) es más rápido y mucho más económico que corregirlos en un entorno de producción (Nube).

2. Escenario de Vulnerabilidad Recomendado

Vulnerabilidad: Bloqueo de Acceso Público Deshabilitado en un Bucket de Almacenamiento S3 (AWS).

Impacto: Permite que datos sensibles sean accedidos públicamente si el bucket no está configurado correctamente, un error extremadamente común y riesgoso.

Herramienta de IaC: Terraform (AWS).

3. Guía Paso a Paso para la Demostración

Paso 1: Preparación (El Código Vulnerable)

Crea un archivo Terraform simple (`main.tf`) con una configuración de AWS S3 que *omite* o *deshabilita* intencionalmente las mejores prácticas de seguridad.

Ejemplo de Código Vulnerable (`main.tf`):

```
# Código de Terraform con vulnerabilidad intencional
resource "aws_s3_bucket" "datos_sensibles" {
  bucket = "mi-bucket-de-demo-sensible-987654"
  # NOTA: Omitimos la configuración 'public_access_block'
}

resource "aws_s3_bucket_acl" "example" {
  bucket = aws_s3_bucket.datos_sensibles.id
  # Uso de una ACL anticuada y potencialmente insegura
  acl    = "public-read"
}
```

Paso 2: Ejecución del Análisis SAST (Snyk IaC)

1. **Instalar Snyk CLI** (si no está instalado).
2. **Ejecutar el comando de análisis de IaC** en la carpeta que contiene el archivo `main.tf` .

```
snyk iac test
```

3. **Mostrar Resultados:** El escaneo de Snyk detectará inmediatamente las configuraciones inseguras.
   - **Resultado Típico:** Snyk reportará un fallo de severidad alta o crítica (ej. `AWS S3 bucket allows public access`).
   - **Enfoque:** Resalta el **código exacto** y la **línea** donde se encuentra el problema, mostrando cómo Snyk conecta la seguridad directamente con el código fuente (similar al feedback de Snyk Code en tu Canvas de SAST).

Paso 3: Remedio y Validación

1. **Corregir el Código:** Modifica el archivo `main.tf` para añadir el bloque de acceso público y aplicar la mejor práctica de seguridad (Shift Left aplicado).

   Ejemplo de Código Corregido (`main.tf`):

```
resource "aws_s3_bucket" "datos_sensibles" {
  bucket = "mi-bucket-de-demo-sensible-987654"
}

# Bloque de acceso público (Public Access Block)
resource "aws_s3_bucket_public_access_block" "datos_sensibles_block" {
  bucket = aws_s3_bucket.datos_sensibles.id
  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

# Eliminar/corregir la ACL insegura si es necesario
resource "aws_s3_bucket_acl" "example" {
  bucket = aws_s3_bucket.datos_sensibles.id
  acl    = "private"
}
```

2. **Re-ejecutar el Análisis:**

```
snyk iac test
```

3. **Mostrar el Resultado "Limpio":** Snyk ahora mostrará "No issues found" o solo fallos de menor severidad, demostrando que la vulnerabilidad se ha resuelto en el código, *antes* de la implementación.

4. Integración en un Pipeline CI/CD (DevSecOps)

Para un curso de TI, es crucial demostrar que la seguridad no es un paso manual, sino un **Quality Gate** automatizado.

Punto de Inserción:

El escaneo con Snyk IaC debe ocurrir *después* de que se valide el código (ej. `terraform validate`) y *antes* de que se ejecute cualquier comando de despliegue (`terraform apply`).

Mecanismo de Falla (Quality Gate):

Snyk CLI está diseñado para devolver un **código de salida (exit code) distinto de cero** si encuentra vulnerabilidades que superan un nivel de umbral definido (ej. Alta o Crítica). Un código de salida distinto de cero hace que el script de CI/CD falle, deteniendo automáticamente el despliegue de la infraestructura vulnerable.

Ejemplo de Etapa de CI/CD (Pseudocódigo):

Añade este paso a tu pipeline (ya sea Jenkins, GitLab, GitHub Actions, etc.) para forzar la verificación de seguridad.

```
# Esta es una etapa típica de 'Test' o 'Security Scan' en un CI/CD

- name: Security Scan - Snyk IaC Check
  script: |
    # 1. Autenticar Snyk (requiere el token de Snyk, típicamente una variable de entorr
    snyk auth $SNYK_TOKEN

    # 2. Ejecutar el escaneo IaC y configurar el Quality Gate
    # --severity-threshold=high detendrá el pipeline si se encuentra AL MENOS una vulne
    echo "Ejecutando Snyk IaC Test con Quality Gate HIGH..."
    snyk iac test --severity-threshold=high

    # NOTA: Si snyk iac test devuelve un error (debido a la vulnerabilidad),
    # el pipeline se detiene aquí y nunca llega al Paso 'terraform apply'.
    echo "¡Escaneo de seguridad aprobado! Procediendo al despliegue..."

- name: Deployment
  script: |
    # Este paso solo se ejecuta si la etapa anterior (Security Scan) fue exitosa
    terraform apply -auto-approve
```

5. Conclusión para el Curso de Infraestructura TI

Este demo es efectivo porque:

- **Es Realista:** Muestra cómo un pequeño error en la configuración del IaC puede resultar en una brecha de seguridad grave en la nube.

- **Alineación con DevSecOps:** Ilustra el valor del "Shift Left" para el equipo de infraestructura, no solo para los desarrolladores de aplicaciones.
- **Visibilidad Inmediata:** Proporciona *feedback* instantáneo, demostrando que el proceso de seguridad no tiene por qué ser un cuello de botella.
- **Automatización:** Demuestra el uso de un **Quality Gate** para prevenir automáticamente despliegues inseguros, una práctica fundamental en TI moderna.