Academia DevOps & Cloud · @jemjaf

ANTIPATRONES COMUNES EN TERRAFORM (EVÍTALOS)

> Para **code review** y entrevistas ("¿qué está mal aquí?"). Cada antipatrón tiene su señal y su corrección directa.

SEGURIDAD Y ESTADO

Antipatrón	Por qué duele	Cómo corregirlo
Secretos en `.tf` o `tfvars` commiteado	Acaban en Git, historial y filtraciones	Variables con `TF_VAR_*`, secretos en Vault / SSM / Key Vault; `.gitignore` en los `*.tfvars` reales
Estado en el repo sin proteger	Filtra IDs, ARNs, IPs; colisiones si dos personas aplican a la vez	Backend remoto con locking (S 3 + DynamoDB, GCS, Terraform Cloud) y cifrado en reposo
`apply -auto-approve` en producción	Destruyes lo que no querías sin revisar el plan	Solo en entornos desechables o CI con aprobación previa; prod: `plan` → revisión → `apply` manual

ESTRUCTURA Y CÓDIGO

Antipatrón	Por qué duele	Cómo corregirlo
Un solo mega-módulo	Imposible de testear, reusar o entender; cualquier cambio toca todo	Módulos pequeños por función (red, cómputo, IAM); la raíz solo orquesta
`count` / `for_each` sin pensar	Cambios de índice o clave destruyen y recrean recursos equivocados	`for_each` con claves de mapa estables; `count` solo para toggle on/off simple
Hardcodear IDs de cuenta/región	Rompe al reutilizar en otra cuenta o entorno	Variables + `data sources` (`aws_caller_identity`, `aws_region`) para valores dinámicos
Copiar-pegar sin variables	El mismo bug en diez archivos; cada parche hay que repetirlo	Variables con validación; extraer módulo cuando el bloque se repite 3 + veces

OPERACIÓN

Antipatrón	Por qué duele	Cómo corregirlo
Modificar infra a mano tras `apply`	El estado queda desincronizado; el próximo `plan` da diffs fantasma	Todo por código; si es urgente, arreglar a mano y luego `terraform import` o `state rm`
Sin pin de versión en providers	Actualización silenciosa en CI rompe comportamiento existente	`required_providers` con `version = "~> X.Y"` en todos los módulos
Un workspace / cuenta para todo	Blast radius máximo: un `destroy` equivocado puede borrar producción	Cuentas o proyectos separados por entorno (dev / staging / prod) con estado independiente

Señal rápida en code review: `*` en una política IAM, un `.tfstate` en el repo, o un módulo de 5 0 0 + líneas — para y pregunta antes de aprobar.

🌟 COMUNIDAD SKOOL — SIGUIENTE PASO

¿LISTO PARA PRACTICARLO DE VERDAD?

Esta guía es el mapa; la diferencia suele estar en **hacer labs sin quedarte atrapado ante un error** durante tres días. En mi comunidad de Skool aprendes con **cursos** dentro de la plataforma, **sesiones en vivo** y **soporte técnico** cuando tu entorno no levanta o tu despliegue falla.

👥 **Entra en la comunidad — skool.com/jemjaf** ↗️

Qué hay dentro (según la oferta vigente en Skool):

- 📚 **Cursos** grabados para ir a tu ritmo, más **lives** temáticos (cloud y DevOps)
- **Sandboxes temporales en AWS, Azure y GCP**: practica en nube **real** sin el miedo a una factura sorpresa
- 🛠️ **Soporte directo** para destrabar laboratorios, código y dudas puntuales
- 📈 **Career Center** para revisión de CV y LinkedIn con mirada técnica
- 📂 Materiales prácticos y guías que acompañan lo que enseño en clase

👉 **Únete en skool.com/jemjaf** ↗️ — Es formación hands-on y acompañamiento; sin prometer empleos automáticos. El objetivo es que **lleves práctica defendible**.