

Lectura 15

Infraestructura como Código

- Útil para automatizar y versionar la provisión y configuración de infraestructuras

¿Qué es infraestructura?: Son los componentes necesarios para ejecutar aplicaciones y servicios:

- **Computo**: servidores físicos o VM
- **Red**: VPC, subredes,平衡adores
- **Almacenamiento**: buckets
- **Seguridad**: políticas
- **Servicios complementarios**: bd, colas, DNS

Tradicionalmente esto se gestionaba de forma manual, un operador ingresaba al sistema de consolas web y ejecutaba los comandos de CLI y esto tendía a errores humanos y ser lento

- Ejecutar comandos implicaba hacer click y sin un registro de lo hecho.
- con Iac se traslada la definición a archivos de texto, en los cuales se describe de forma declarativa qué recursos queremos, no como crearlos. Esto se almacena en archivos en un repositorio git (**versionar, revisar, automatizar**)

¿Qué no es Iac?

- Ejecutar scripts
- Configurar servidores manualmente

Principios de Iac

1. Reproducibilidad: Un archivo de Iac debe permitir recrear un entorno idéntico cada vez que se aplique. El repositorio y sus versiones etiquetadas garantizan entornos reproducibles

2. Idempotencia: Aplicar varias veces el mismo código no debe cambiar el estado si ya está en el resultado deseado, no recreará el recurso

3. Composabilidad: Definir módulos o bloques reutilizables que puedan combinarse para construir infraestructuras complejas. Cada módulo encapsula una pieza de infraestructura red o computo y se combinan sin duplicar código.

4. Evolutividad: Facilita la extensión y adaptación de la configuración a medida que cambian los requisitos.

5. Aplicación de Principios:

1. Separación de responsabilidades
2. Parametrización
3. Portabilidad con Docker

¿Porque usar IaC?

1. Gestión de cambios:

- **Rastro de auditoría:** Cada modificación en la infraestructura queda registrada como un commit de git
- **Revisión por pares:** Antes de aplicar un cambio, se abre un pull request que incluye un **terraform plan**
- **Roll back instantáneo:** Si un despliegue automático introduce un error, basta con revertir un commit con **git revert <SHA>**, terraform deshacerá todos los cambios no deseados en minutos en lugar de horas haciendolo de forma manual.

2. Retorno de Inversión (ROI) de tiempo:

- **Despliegues exprés**
- **Pipeline automatizado:** Integra IaC en github actions para que se ejecuten plan, apply, etc.

3. Compartir conocimiento:

- **Documentación viva en el código:** Variables con nombres claros, comentarios en módulo y ejemplos en el README.md
- **Onboarding acelerado:** Al clonar un repositorio y usar **docker-compose up -build**, se levanta un entorno de pruebas idéntico al de producción
- **Bibliotecas de módulos reutilizables:** Almacena módulos genéricos, el equipo crea un catálogo interno de bloques IaC, esto evita reinventar la rueda

4. Seguridad:

- Gestión centralizada de secretos: integra vault, AWS SSM o Azure Key Vault
- Revisión de políticas: Definir roles y permisos de IAM como código
- Principio de menor privilegio: Al versionar documentos que permisos exactos necesita cada componente.

Herramientas:

1. Aprovisionamiento: Es la etapa de orquestar o crear los recursos de infraestructura, En Devops son herramientas para versionado, idempotencia, multi-proveedor AWS, GCP, Azure
 - pulumi up crea o actualiza recursos
2. Gestión de configuración: Se encarga del estado ideal: instalar paquetes, copiar archivos de configuración, gestionar servicios
3. Construcción de imágenes: crea artefactos inmutables, contenedores docker o imágenes VM con todo pre-instalado para un arranque rápido, reproducibilidad e inmutabilidad (si falla un nodo se descarta y lanza otra igual)