Infraestructura como Código (IaC) con AWS y GitHub

GitTogether Medellín

David Arias
Cloud DevOps, Solutions Architect

Hora y Fecha:
Jueves, 20 Febrero 2025, Eafit, Bloque 18, Piso 2 en la sala percepción

Registro:
https://www.meetup.com/gittogether-medellin/events/306021996

WITH THE SUPPORT OF

Rockwell Automation · NODO · EAFIT

// DAVID ARIAS FUENTES

Sobre mi:

Traductor de inglés por casi 10 años, debido a la pandemia me reinventé, autodidacta, estudié muchas rutas de Backend y de nube, actualmente trabajo como DevOps Engineer en una empresa de USA en HealthCare.

Comunidades

Fui instructor de Javascript de la comunidad de Women Who Code Medellín por 3 años, líder de AWS User Group Medellín desde 2022, AWS Community Builder en la categoría de DevTools, speaker en diferentes eventos de AWS y organizador de eventos.

Certificaciones

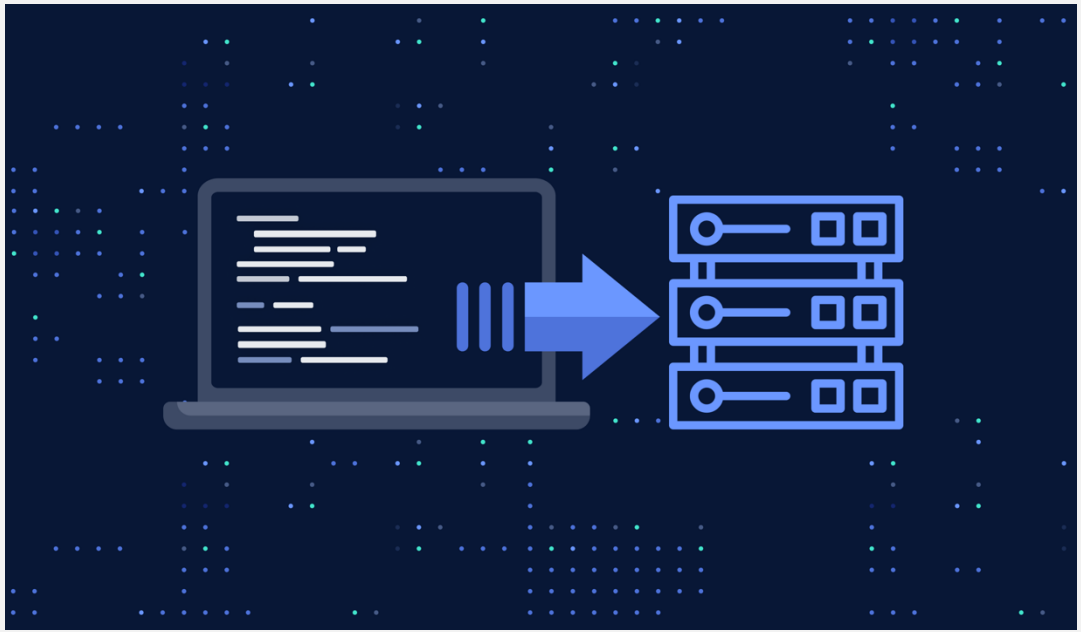
- AWS Cloud Practitioner
- AWS Developer Associate
- AWS Solutions Architect Associate
- AWS Cloud AI Practitioner
- AWS DevOps Engineer Professional
- HashiCorp Certified: Terraform Associate

// Agenda

1. Qué es IaC?
2. Terraform
3. Fundamentos de Terraform
4. Integración con AWS
5. Integración con GitHub
6. Crear un pipeline con GitHub Actions

// Qué es IaC?

IaC (Infrastructure as Code) es una práctica en la que la infraestructura de TI se gestiona y aprovisiona mediante código en lugar de procesos manuales. Con IaC, los servidores, redes, bases de datos y otros recursos se definen mediante archivos de configuración o scripts, lo que permite automatizar la implementación y administración de la infraestructura.



// Qué es IaC?

Beneficios de IaC:

•**Automatización y eficiencia**: Reduce el trabajo manual y los errores humanos.

•**Consistencia**: Se evitan configuraciones inconsistentes al definir la infraestructura en código.

•**Escalabilidad**: Facilita la creación y destrucción de entornos de manera rápida.

•**Versionado y trazabilidad**: Al usar control de versiones (como Git), se pueden rastrear cambios y revertir configuraciones fácilmente.

•**Reproducibilidad**: Permite crear entornos idénticos en distintas fases (desarrollo, prueba, producción).

// Qué es IaC?

Herramientas populares de IaC:

•**Terraform**: Utiliza un lenguaje declarativo para definir infraestructura en múltiples proveedores (AWS, Azure, GCP).

•**Ansible**: Usa YAML para automatizar configuraciones y despliegues.

•**CloudFormation**: Servicio de AWS para gestionar infraestructura mediante plantillas en JSON o YAML.

•**Pulumi**: Permite definir infraestructura con lenguajes de programación como Python, TypeScript y Go.

// Razones para elegir Terraform.

1. Multi-nube y multi-proveedor
2. Lenguaje declarativo (HCL - HashiCorp Configuration Language)
3. Planificación antes de aplicar cambios (terraform plan)
4. Estado de infraestructura (terraform state)
5. Modularidad y reutilización
6. Automatización y CI/CD
7. Gran comunidad y soporte
8. Gestión de dependencias
9. Open Source y Enterprise

// Fundamentos de Terraform

1. Lenguaje Declarativo (HCL)

2. Estado de la Infraestructura (terraform.tfstate)

3. Ciclo de Vida de Terraform
    - Paso 1: Inicialización (terraform init)
    - Paso 2: Planificación (terraform plan)
    - Paso 3: Aplicación (terraform apply)
    - Paso 4: Destrucción (terraform destroy)

4. Recursos y Proveedores
5. Variables y Salidas
6. Módulos y Reutilización
7. Backend Remoto y Bloqueo de Estado
8. Gestión de Dependencias
9. Seguridad y Buenas Prácticas

// Integración con AWS

Terraform facilita la gestión de infraestructura en AWS con código, asegurando que los entornos sean reproducibles, escalables y automatizados.

1. ✅ **Definir infraestructura en código (main.tf)**
2. ✅ **Automatizar despliegues con GitHub Actions**
3. ✅ **Usar backend remoto en S3 para trabajo en equipo**
4. ✅ **Proteger credenciales con GitHub Secrets**

// Integración con GitHub

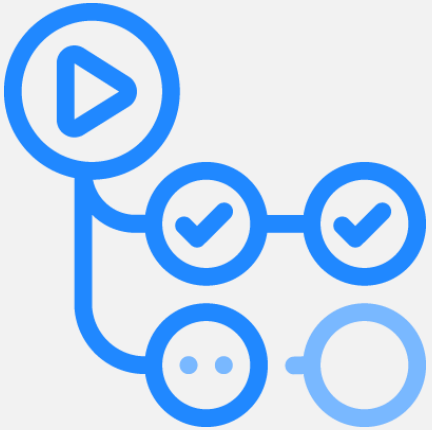
Integrar Terraform con GitHub y GitHub Actions permite:

- ✅ **Automatizar despliegues de infraestructura.**
- ✅ **Revisar cambios antes de aplicarlos (terraform plan).**
- ✅ **Usar secretos de GitHub para proteger credenciales.**
- ✅ **Almacenar el estado en un backend para trabajo en equipo.**
- ✅ **Mejorar la seguridad con Pull Requests.**

// Buenas prácticas

- [ ] Estructura y Organización del Código
- [ ] Uso de GitHub para Control de Versiones
- [ ] Seguridad y Gestión de Credenciales
- [ ] Automatización y CI/CD
- [ ] Gestión de Estados y Backends
- [ ] Manejo de Entornos
- [ ] Monitoreo y Auditoría
- [ ] Gestión de Costos

// Crear un despliegue con GitHub Actions



¿Preguntas?

GRACIAS