

Implementación Automatizada con Terraform Cloud, GitHub Actions y AWS

Autor: Acosta Hernan Andres

Fecha: 31/03/2025

Objetivo

Este proyecto demostró la integración de:

HCP Terraform (gestión centralizada de estado)

GitHub Actions (CI/CD automatizado)

AWS (proveedor cloud)

-----///-----

[Desarrollador]

↓ (Push código)

[Repositorio GitHub]

↓ (Trigger automático)

[GitHub Actions]

↓ (Ejecuta)

[Terraform Plan] → [Muestra cambios en Pull Request]

↓ (Tras aprobación)

[Merge a rama principal]

↓ (Trigger automático)

[GitHub Actions Apply]

↓ (Aplica cambios)

[HCP Terraform]

↓ (Gestiona infra)

[Recursos en AWS]

-----////////-----

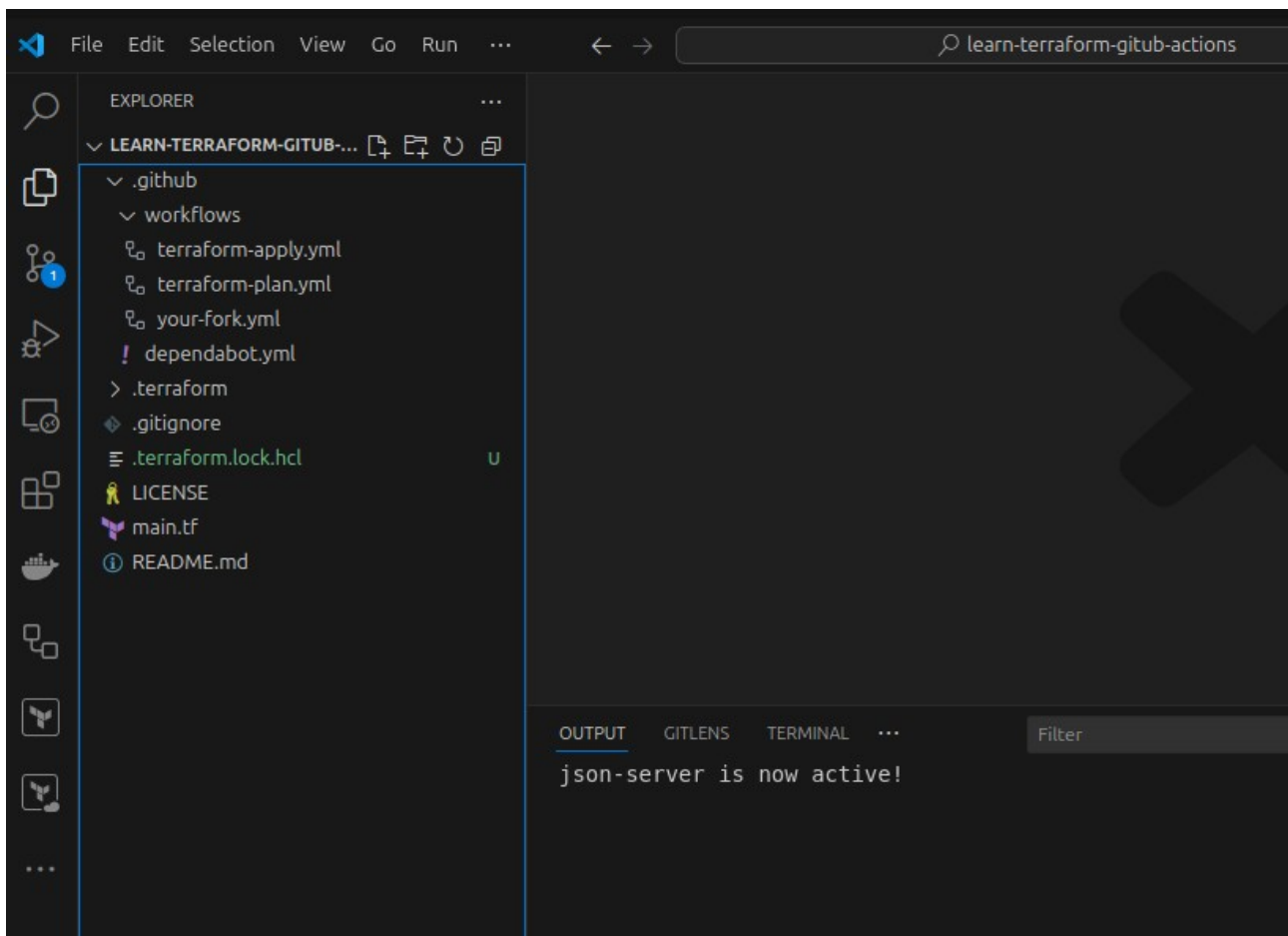
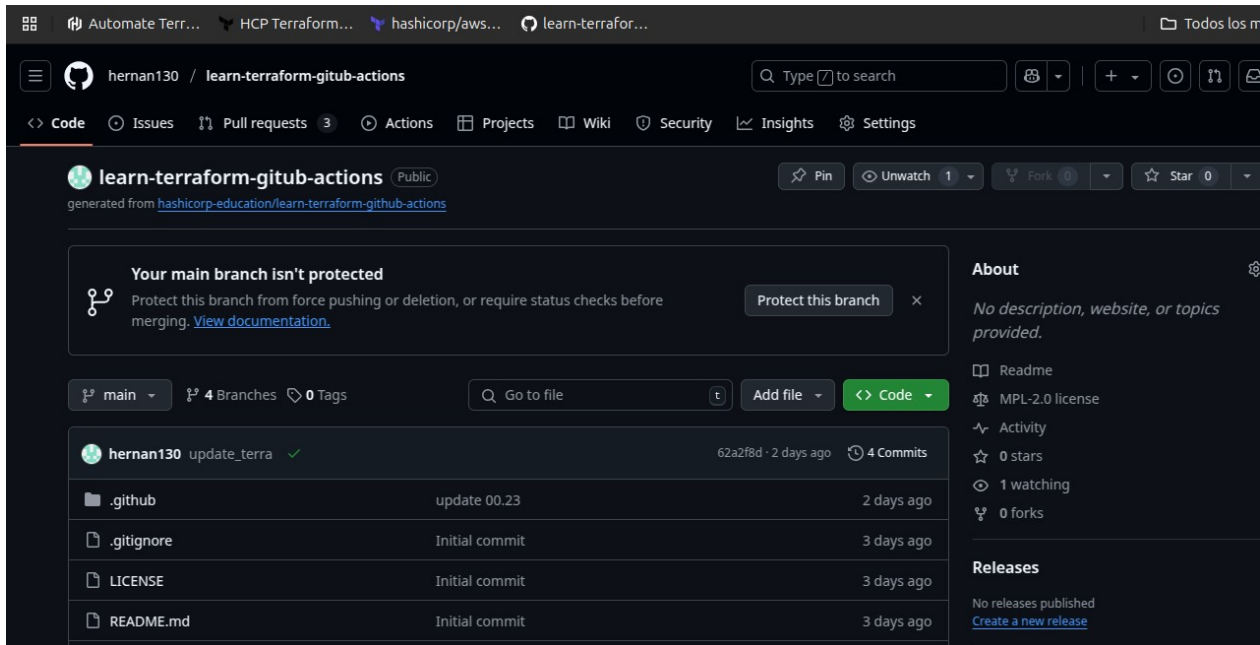
Flujo de trabajo implementado:

1. El desarrollador realiza un Push al repositorio de GitHub.
2. GitHub Actions detecta los cambios y ejecuta un Terraform Plan.
3. Se revisa el resultado en el Pull Request.
4. Si se aprueba, se realiza el merge a la rama main .
5. GitHub Actions ejecuta Terraform Apply y despliega los cambios en AWS.

como primera medida se utilizo la plantilla del repositorio propuesto((Post Hashicorp) -

<https://developer.hashicorp.com/terraform/tutorials/automation/github-actions>)

para crear un repositorio personal y luego clonarlo a nivel localmente



Configuración Realizada

3.1 HCP Terraform

Team Tokens

Team API tokens are used by services, for example a CI/CD pipeline, to perform plans and applies on a workspace. Treat this token like a password, as it is used to access your account without your username, password, or two-factor authentication.

Team Tokens (1)

[Create a team token](#)

Team name ↕	Permission	Expiry date ↑	Created by ↕	Last used ↕
owners	Members can manage	Apr 27, 2025	hernan_acosta	34 minutes ago

Creación de workspace y token de API:

Variables de AWS configuradas como sensibles:

Llave	Valor	Categoría	Comp
ID DE CLAVE DE ACCESO DE AWS Sensible	Sensible - solo escritura	entorno	...
CLAVE DE ACCESO SECRETA DE AWS Sensible	Sensible - solo escritura	entorno	...

[+ Agregar variable](#)

Conjuntos de variables (0)

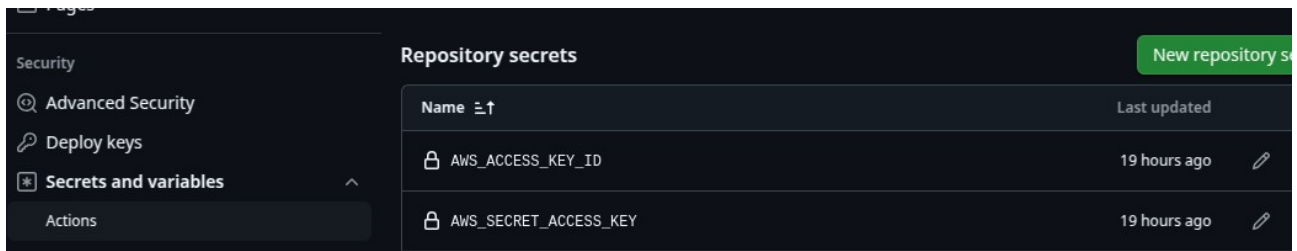
Secrets del repositorio con TF_API_TOKEN:

The screenshot shows the GitHub 'Repository secrets' page. On the left, a sidebar contains links for 'Pages', 'Security', 'Advanced Security', 'Deploy keys', 'Secrets and variables', and 'Actions'. The main area is titled 'Repository secrets' and includes a 'New repository secret' button. A table lists the secrets, with one entry visible: 'TF_API_TOKEN' created '2 hours ago'. The table has columns for 'Name' and 'Last updated'.

Configuración de Secrets en GitHub

Se almacenaron las credenciales de AWS en Settings > Secrets > Actions:

- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY



- Workflows en .GitHub/Workflows/:
terraform-plan.yml
- name: "Terraform Plan"

on:

Pull_Request:

env:

TF_CLOUD_ORGANIZATION: "mi-org-tf"

TF_API_TOKEN: "\${{ secrets.TF_API_TOKEN }}"

TF_WORKSPACE: "learn-terraform-github-actions"

CONFIG_DIRECTORY: "./"

jobs:

terraform:

if: github.repository != 'hashicorp-education/learn-terraform-github-actions'

name: "Terraform Plan"

runs-on: ubuntu-latest

permissions:

so GitHub can check out this repo using the default github.token

contents: read

pull-requests: write

steps:

- name: Checkout

uses: actions/checkout@v3

- name: Upload Configuration

uses: hashicorp/tfc-workflows-github/actions/upload-configuration@v1.0.0

id: plan-upload

with:

workspace: "\${{ env.TF_WORKSPACE }}"

directory: "\${{ env.CONFIG_DIRECTORY }}"

speculative: true

- name: Create Plan Run

uses: hashicorp/tfc-workflows-github/actions/create-run@v1.0.0

id: plan-run

with:

workspace: "\${{ env.TF_WORKSPACE }}"

configuration_version: "\${{ steps.plan-upload.outputs.configuration_version_id }}"

plan_only: true

```

- name: Get Plan Output
uses: hashicorp/tfc-workflows-github/actions/plan-output@v1.0.0
id: plan-output
with:
plan: ${{ fromJSON(steps.plan-run.outputs.payload).data.relationships.plan.data.id }}

- name: Update PR
uses: actions/github-script@v6
id: plan-comment
with:
github-token: ${{ secrets.GITHUB_TOKEN }}
script: |
// 1. Retrieve existing bot comments for the PR
const { data: comments } = await github.rest.issues.listComments({
owner: context.repo.owner,
repo: context.repo.repo,
issue_number: context.issue.number,
});
const botComment = comments.find(comment => {
return comment.user.type === 'Bot' && comment.body.includes('Terraform Cloud Plan Output')
});
const output = `#### Terraform Cloud Plan Output
\\`\\`\\`
Plan: ${{ steps.plan-output.outputs.add }} to add, ${{ steps.plan-output.outputs.change }} to
change, ${{ steps.plan-output.outputs.destroy }} to destroy.
\\`\\`\\`
[Terraform Cloud Plan](${{ steps.plan-run.outputs.run_link }})
`;
// 3. Delete previous comment so PR timeline makes sense
if (botComment) {
github.rest.issues.deleteComment({
owner: context.repo.owner,
repo: context.repo.repo,
comment_id: botComment.id,
});
}
github.rest.issues.createComment({
issue_number: context.issue.number,
owner: context.repo.owner,
repo: context.repo.repo,
body: output
});

```

- terraform-apply.yml)
name: "Terraform Apply"

on:
push:
branches:

- main

env:

TF_CLOUD_ORGANIZATION: "mi-org-tf"

TF_API_TOKEN: "\${{ secrets.TF_API_TOKEN }}"

TF_WORKSPACE: "learn-terraform-github-actions"

CONFIG_DIRECTORY: "./"

jobs:

terraform:

if: github.repository != 'hashicorp-education/learn-terraform-github-actions'

name: "Terraform Apply"

runs-on: ubuntu-latest

permissions: # granular permissions

so GitHub can check out this repo using the default github.token

contents: read

steps:

- name: Checkout

uses: actions/checkout@v3

- name: Upload Configuration

uses: hashicorp/tfc-workflows-github/actions/upload-configuration@v1.0.0

id: apply-upload

with:

workspace: "\${{ env.TF_WORKSPACE }}"

directory: "\${{ env.CONFIG_DIRECTORY }}"

- name: Create Apply Run

uses: hashicorp/tfc-workflows-github/actions/create-run@v1.0.0

id: apply-run

with:

workspace: "\${{ env.TF_WORKSPACE }}"

configuration_version: "\${{ steps.apply-upload.outputs.configuration_version_id }}"

- name: Apply

uses: hashicorp/tfc-workflows-github/actions/apply-run@v1.0.0

if: fromJSON(steps.apply-run.outputs.payload).data.attributes.actions.IsConfirmable

id: apply

with:

run: "\${{ steps.apply-run.outputs.run_id }}"


comment: "Apply Run from GitHub Actions CI \${{ github.sha }}"

4. Flujo de Trabajo

4.1 Terraform Plan (PR)



Ejecución actual

**Activado desde Terraform Cloud CI por el autor (hernan130) para SHA (62a2f8d...**
run-zFowMGmRVDy3sae3 | Integración de API activada mediante API

ACTUAL

✓ Planificado y terminado

Hace 21 minutos


Lista de ejecución

[Todo](#) 3 [⚠ Necesita atención](#) 0 [✖ Error](#) 0 [🔄 Correr](#) 0 [⏸ En espera](#) 0 [✅ Éxito](#) 3

⌵ Estado

⌵ Operación


⌵ Fuente

**Activado desde Terraform Cloud CI por el autor (hernan130) para SHA (62a2f8d...**
run-zFowMGmRVDy3sae3 | Integración de API activada mediante API

ACTUAL

✓ Planificado y terminado

Hace 21 minutos

**Activado desde Terraform Cloud CI por el autor (hernan130) para SHA (137106b...**

✓ Planificado y terminado

hace una hora

1. Se creó un Pull Request con cambios :

GitHub Actions

Terraform Apply #4: Commit 62a2f8d pushed by hernan130 35s

✓ "Update TFC organization" main 2 days ago 33s ...

Terraform Apply #3: Commit 137106b pushed by hernan130

Recursos creados en AWS:

Instancia EC2 t2.micro con acceso HTTP:

EC2

Dashboard

EC2 Global View

Events

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Instance summary for i-0beda5f6445428d30

Updated 5 minutes ago

Instance ID

i-0beda5f6445428d30

IPv6 address

-

Hostname type

IP name: ip-172-31-36-152.us-west-2.compute.internal

Answer private resource DNS name

-

Auto-assigned IP address

34.218.209.185 [Public IP]

Public IPv4 address

34.218.209.185 | open address

Instance state

Running

Private IP DNS name (IPv4 only)

ip-172-31-36-152.us-west-2.compute.internal

Instance type

t2.micro

VPC ID

vpc-0f774ed0f3aeef536

Private IPv4 addresses

172.31.36.152

Public IPv4 DNS

ec2-34-218-209-185.us-west-2.compute.amazonaws.com | open address

Elastic IP addresses

-

AWS Compute Optimizer finding

Opt-in to AWS Compute Optimizer findings. | Learn more

Conclusión

La implementación fue exitosa, logrando:

- ♦ Automatización completa del ciclo de vida de IaC
- ♦ Integración entre Terraform Cloud y GitHub
- ♦ Creación verificable de recursos en AWS

Lo que mejoraría en esta implementación es la estructura

Estructura original

learn-terraform-github-actions/

```
├── .github/
│   ├── workflows/          # Directorio de flujos de GitHub Actions
│   │   ├── terraform-plan.yml # Workflow para ejecutar terraform plan
│   │   └── terraform-apply.yml # Workflow para ejecutar terraform apply
│   └── dependabot.yml       # Configuración para actualizaciones automáticas de
dependencias
├── .gitignore               # Archivos excluidos del control de versiones
├── LICENSE                  # Licencia del proyecto (probablemente MIT)
├── README.md                # Documentación principal del proyecto
└── main.tf                  # Archivo principal de configuración de Terraform
```

Estructura con modificaciones

learn-terraform-github-actions/

```
├── .github/
│   ├── workflows/
│   │   ├── terraform-plan.yml
│   │   └── terraform-apply.yml
│   └── dependabot.yml
├── modules/                # Módulos reutilizables (opcional)
│   └── example-module/
│       ├── main.tf
│       ├── variables.tf
│       └── outputs.tf
├── environments/           # Configuración por entorno
│   ├── dev/
│   │   ├── main.tf         # Config específica de dev
│   │   ├── terraform.tfvars # Variables para dev
│   │   └── backend.tf       # Backend config para dev
│   └── prod/
│       ├── main.tf
│       ├── terraform.tfvars
│       └── backend.tf
```



```
|— main.tf          # Configuración principal
|— providers.tf     # Configuración de proveedores
|— variables.tf     # Variables globales
|— outputs.tf       # Outputs globales
|— terraform.tfvars.example # Ejemplo de variables (sin valores sensibles)
|— .gitignore
|— LICENSE
└— README.md
```

Mejoras clave:

1. **Separación clara de componentes:**

- providers.tf para la configuración de proveedores
- variables.tf para declaración de variables
- outputs.tf para outputs

2. **Estructura por entornos (dev/prod) con:**

- Configuraciones específicas por entorno
- Variables separadas
- Backend configuration independiente

3. **Módulos opcionales para componentes reutilizables**

Ejemplo de terraform.tfvars sin valores reales para documentación

4. **README.md debería explicar:**

- Cómo usar los workflows de GitHub Actions
- Estructura del proyecto
- Requisitos previos
- Ejemplos de uso