

UNIVERSIDAD PRIVADA DE TACNA



Help Deploy

Asistente para Automatización de Terraform y Git

Ingeniería de Sistemas | Calidad y Pruebas de
Software

Augusto Joaquin Rivera
Muñoz Código: 2022073505

Jefferson Rosas
Chambilla Código:
2021072618

Docente: Ing. Patrick Jose Cuadros Quiroga

Contexto del Proyecto

¿Qué es Help Deploy?

Es una extensión innovadora para **Visual Studio Code** diseñada para asistir a desarrolladores en la gestión de Infraestructura como Código (IaC).

El proyecto busca cerrar la brecha entre el desarrollo de software y las operaciones (DevOps), para automatizar flujos de trabajo en **Terraform** y **Git**.



Planteamiento del Problema

Curva de Aprendizaje

Terraform posee una sintaxis compleja (HCL) que representa una barrera de entrada alta para desarrolladores junior.



Errores Manuales

La configuración manual de archivos de infraestructura es propensa a errores de sintaxis que fallan en tiempo de despliegue.

Ineficiencia

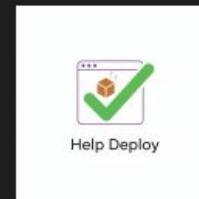
El cambio constante de contexto entre la documentación oficial y el editor de código reduce significativamente la productividad.

Nuestra Solución

Asistente Inteligente Integrado

Help Deploy vive dentro del IDE, proporcionando:

- 🕒 **Generación de Scripts:** Creación automática de archivos main.tf basados en lenguaje natural.
- ⚡ **Validación en Tiempo Real:** Detección proactiva de errores antes del terraform apply.
- > **Automatización Git:** Flujos de commit y push simplificados desde la misma interfaz.



DETAILS FEATURES CHANGELOG

Help Deploy

Augusto Joaquin Rivera Muñoz | 5 | ★★★★★

Asistente para Terraform y Git: clonar repos, crear ramas, plantilla README, commits guiados y stash.

Auto Update

Help Deploy

Asistente para Terraform y Git desde Visual Studio Code: clonar repos, crear ramas, plantilla README, commits guiados y stash.

Comandos

- Clonar repositorio (Git): `help-deploy.gitClone`
- Crear rama (Git): `help-deploy.gitCreateBranch`
- Crear README.md (Plantilla): `help-deploy.createReadme`
- Crear main.tf: `help-deploy.createMainTf`
- Desplegar proyecto con Terraform: `help-deploy.desplegarTerraform`
- Pushear cambios (Git) con wizard de commits: `help-deploy.gitPush`
- Guardado temporal (stash) (Git): `help-deploy.gitStashAssistant`
- Recuperar stash (Git): `help-deploy.gitStashRecover`

Requisitos

- Tener `Terraform` instalado y en el `PATH` (`terraform -v`).
- Credenciales/configuración del proveedor si usas `plan`/`apply`.
- Tener `git` disponible en el `PATH` para los asistentes de Git.

Uso rápido

- Terraform:
 - Crear `main.tf` para una plantilla básica.
 - Desplegar proyecto con Terraform ejecuta `init` → `plan` → confirmación → `apply`.
- Git:
 - Clonar `repositorio` para traer el repo.
 - Crear `rama` para crear y publicar una nueva rama.
 - Pushear `cambios` con wizard que genera mensajes `feat(scope): description`.
 - Guardado temporal crea un `stash` nombrado, `Recuperar stash` lo aplica o hace `pop`.

Stack Tecnológico

Core & Frontend

- **Visual Studio Code API:** Para la integración nativa con el editor.
- **TypeScript:** Lenguaje principal para lógica robusta y tipada.
- **Webviews:** Para interfaces de usuario personalizadas dentro del IDE.

Inteligencia & Backend

- **Terraform CLI:** `init` , `plan` , `apply` desde el IDE; actualización automática de `.gitignore` .
- **Git CLI:** `clone` , `checkout -b` , `wizard de commits`, `push` , `stash` y recuperación.
- **Git CLI:** `clone` , `checkout -b` , `wizard de commits`, `push` , `stash` y recuperación.

Objetivos del Proyecto

Objetivo General

Desarrollar un asistente que centralice y optimice la creación de infraestructura y control de versiones.

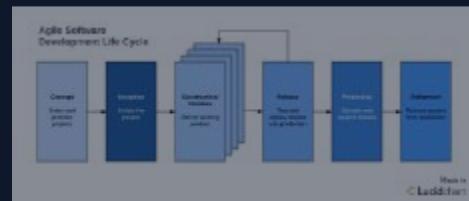
Objetivos Específicos

- ✓ Reducir el tiempo de configuración de entornos en la nube mediante automatización.
- ✓ Minimizar la tasa de errores de sintaxis en archivos HCL (HashiCorp Configuration Language).
- ✓ Facilitar la adopción de prácticas DevOps en estudiantes y desarrolladores junior.



Cronograma de Ejecución (2025-II)

AGO	SEP	OCT	NOV
Inicio Definición de alcance, identificación de actores y planificación inicial.	Elaboración Diseño de arquitectura, modelado de datos y prototipado UI/UX.	Construcción Implementación del módulo integración con Terraform y Git.	Transición Pruebas de calidad (QA), corrección de errores y despliegue final.



Análisis de Factibilidad



Técnica

El equipo posee conocimientos en TypeScript y VS Code API. Las herramientas base (Git/Terraform) son estándares abiertos.



Económica

Uso de capas gratuitas en servicios cloud y APIs de bajo costo. No requiere infraestructura física costosa.



Operativa

Alta demanda de herramientas que simplifiquen DevOps. Interfaz intuitiva reduce la resistencia al cambio.

Impacto Esperado: Tiempo de Despliegue

Comparativa estimada de tiempo para desplegar una instancia EC2 básica + configuración de red.



**Reducción estimada del 78% en tiempo operativo gracias a la generación automática de código.*

| Conclusión

“

"Help Deploy no solo automatiza tareas; democratiza el acceso a la infraestructura como código, permitiendo que los desarrolladores se enfoquen en innovar, no en configurar."

Equipo de Desarrollo - Help Deploy



¿Preguntas?

Gracias por su atención