



Help Deploy



## UNIVERSIDAD PRIVADA DE TACNA

### FACULTAD DE INGENIERÍA

### Escuela Profesional de Ingeniería de Sistemas

### Proyecto “Help Deploy - Asistente para Automatización de Terraform y Git”

Curso: *CALIDAD Y PRUEBAS DE SOFTWARE*

Docente: ING. PATRICK JOSE CUADROS QUIROGA

Integrantes:

**AUGUSTO JOAQUIN RIVERA MUÑOZ  
JEFFERSON ROSAS CHAMBILLA**

**(2022073505)  
(2021072618)**

**Tacna – Perú  
2025 II**



## ÍNDICE

<b>ÍNDICE.....</b>	<b>2</b>
<b>Proyecto.....</b>	<b>3</b>
<b>Visión.....</b>	<b>3</b>
<b>Requisitos de Arquitectura.....</b>	<b>3</b>
<b>Estructura de Proyecto (lógica).....</b>	<b>3</b>
<b>Componentes Principales.....</b>	<b>4</b>
<b>Vistas y Contribuciones.....</b>	<b>5</b>
<b>Integraciones Externas.....</b>	<b>5</b>
<b>Flujos de Secuencia (texto).....</b>	<b>5</b>
<b>Datos y Persistencia.....</b>	<b>6</b>
<b>Decisiones de Arquitectura (ADR).....</b>	<b>6</b>
<b>Seguridad.....</b>	<b>6</b>
<b>Despliegue y Packaging.....</b>	<b>7</b>
<b>Observabilidad.....</b>	<b>7</b>
<b>Escalabilidad y Extensibilidad.....</b>	<b>7</b>
<b>Riesgos.....</b>	<b>7</b>
<b>Roadmap sugerido.....</b>	<b>7</b>



## Proyecto

- Nombre: Help Deploy
- Versión: 1.0.9
- Tipo: Extensión de Visual Studio Code

## Visión

- Integrar en el IDE flujos de Git y Terraform mediante asistentes guiados, con trazabilidad vía historial y mínima fricción operativa.

## Requisitos de Arquitectura

- Calidad: usabilidad, mantenibilidad, compatibilidad, seguridad.
- Integración: VS Code API, Git CLI, Terraform CLI.
- Observabilidad: feedback inmediato en terminal y registro en historial.

## Estructura de Proyecto (lógica)

- **src/extension.ts**: punto de entrada, registro de comandos, proveedores de vista.
- **dist/extension.js**: bundle generado por **esbuild**.
- **images/\***: iconos del contenedor y recursos visuales.

## Componentes Principales



- Registro de comandos
  - o **gitClone**, **gitCreateBranch**, **createReadme**, **crearMainTf**,  
**desplegarTerraform**, **gitPush**, **gitStashAssistant**, **gitStashRecover**.
- Terminales
  - o **getGitTerminal** y **getTerraformTerminal**: reutilización de sesiones y  
ajuste de **cwd**.
- Generador de plantillas
  - o **buildMainTfTemplate**: crea **main.tf** según proveedor y parámetros.
- Actualización de **.gitignore**
  - o **ensureGitignoreTerraform**: añade entradas faltantes post-**apply**.
- Historial
  - o **HistoryStore**: persistencia en **globalState** (máx. 100 entradas).
  - o **HistoryProvider**: **TreeDataProvider** para la vista **Historial**.
  - o Acciones: limpiar (**help-deploy.history.clear**) y copiar  
(**help-deploy.history.copyEntry**).



## Help Deploy



- Activity Bar
  - o Contenedor **help-deploy-history** con título **Help Deploy** e ícono **images/btnHistorial(2).png**.
- Vista **helpDeployHistoryView**
  - o Lista de entradas con **label**, **detail**, **time** y tooltips.
  - o Menús: botón de limpiar en título, copiar en menú contextual del ítem.

## Integraciones Externas

- Git CLI
  - o **execFile** para obtener rama, **sendText** para operaciones de flujo.
- Terraform CLI
  - o **sendText** para **init**, **plan**, **apply** y manejo interactivo.

## Flujos de Secuencia (texto)

- Push convencional
  - o Usuario elige tipo → ámbito → descripción → **git add** → **git commit -m** → **git push** → registro en historial.
- Despliegue Terraform
  - o Selección de carpeta → crear/abrir **main.tf** → **init** → **plan** → confirmar → **apply** → actualizar **.gitignore** → registro en historial.

## Datos y Persistencia



# Help Deploy



- Estructura de entrada de historial: { **label: string, detail?: string, time: number** }.
- Almacenamiento: **globalState** clave **help-deploy.history**, límite 100 entradas, orden inverso.

## Decisiones de Arquitectura (ADR)

- Uso de Terminal VS **execFile**
  - o Git: lectura con **execFile** (e.g., rama), operaciones en terminal para feedback y control.
  - o Terraform: siempre en terminal por naturaleza interactiva.
- No almacenar secretos
  - o Credenciales se aplican como variables de entorno de la sesión de terminal.
- Reutilización de terminales
  - o Evita múltiples sesiones; mejora rendimiento y UX.
- Icono único para contenedor

## Seguridad

- Sanitización de parámetros y comillas (**quotePath**).
- Sin persistencia de secretos; uso temporal de entorno.

## Despliegue y Packaging



# Help Deploy



- **esbuild** para bundle.
- **vsce package** para generar .vsix.
- **eslint** y **tsc --noEmit** en prepublish.

## Observabilidad

- Feedback en terminal.
- Historial navegable con copiar/limpiar.

## Escalabilidad y Extensibilidad

- Nuevos comandos de Git/Terraform siguiendo patrones existentes.
- Filtros y búsqueda en historial.
- Webviews para flujos más ricos.

## Riesgos

- Diferencias entre shells (PowerShell vs bash).
- **cwd** inválido al lanzar terminal.
- API de VS Code cambiando entre versiones.

## Roadmap sugerido

- Filtros por tipo y fecha en Historial.
- Previsualización de **plan** en panel dedicado.
- Soporte multi-workspace con contexto por carpeta.