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 ***(2022073505)***
JEFFERSON ROSAS CHAMBILLA ***(2021072618)***

Tacna – Perú
2025 II

ÍNDICE

Proyecto

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

Propósito

- Proveer asistentes guiados para tareas comunes de Git y despliegues con Terraform directamente desde VS Code, reduciendo fricción operativa y errores.

Alcance

- Asistentes de Git: clonar repos, crear ramas, crear README, commits convencionales y push, gestión de stash.
- Asistentes de Terraform: creación de **main.tf** y ejecución de **init**/**plan**/**apply** con actualización automática de **.gitignore**.
- Historial integrado: registro de ejecuciones con acciones de limpiar y copiar.

Definiciones

- SRS: Software Requirements Specification.
- CLI: Command Line Interface.
- ConvCommits: Convenciones de commits tipo **feat(scope): description**.

Actores

- Desarrollador: usa los asistentes desde VS Code para operar Git y Terraform.
- Equipo DevOps: aprovecha la estandarización de flujos y mensajes de commit.

Requisitos Funcionales

- Clonar repositorio (**help-deploy.gitClone**)
    - Entrada: URL del repositorio.
    - Proceso: abrir terminal Git en carpeta destino, ejecutar **git clone**.
    - Salida: repositorio clonado en la ruta seleccionada.
- Crear rama (**help-deploy.gitCreateBranch**)
    - Entrada: nombre de la nueva rama, rama base opcional.
    - Proceso: **git fetch**, **git checkout -b**, **git push --set-upstream**.
    - Salida: rama creada y publicada en remoto.
- Crear README (**help-deploy.createReadme**)
    - Entrada: título y descripción breve.
    - Proceso: generar **README.md** con secciones estándar.
    - Salida: archivo abierto en el editor.
- Crear **main.tf** (**help-deploy.crearMainTf**)
    - Entrada: proveedor (AWS/Azure/GCP/Genérico) y parámetros.
    - Proceso: generar plantilla acorde.
    - Salida: archivo **main.tf** creado y abierto.
- Desplegar con Terraform (**help-deploy.desplegarTerraform**)
    - Entrada: carpeta con **main.tf** y credenciales opcionales.
    - Proceso: **terraform init**, **terraform plan**, confirmación modal, **terraform apply**.
    - Postproceso: actualización de **.gitignore** con entradas de Terraform.

- Pushear cambios (**help-deploy.gitPush**)
    - Entrada: tipo de commit, ámbito opcional, descripción corta, rama destino.
    - Proceso: **git add**, **git commit -m**, **git push origin <rama>**.
    - Salida: commit y push con mensaje convencional.
- Guardado temporal (stash) (**help-deploy.gitStashAssistant**)
    - Entrada: nombre del stash.
    - Proceso: **git stash save "<nombre>"**.
- Recuperar stash (**help-deploy.gitStashRecover**)
    - Entrada: selección de stash y acción (**apply**/**pop**).
    - Proceso: **git stash <acción> <ref>**.
- Historial
    - Registro: guardar entrada **{label, detail, time}** en **globalState** (máx. 100).
    - Acciones: limpiar historial, copiar detalles.

Requisitos de Interfaz de Usuario

- Paleta de comandos: accesible por **Ctrl+Shift+P**.
- Barra lateral: contenedor **Help Deploy** con vista **Historial** e icono **images/btnHistorial(2).png**.
- Terminales: reutilización de **Git** y **Terraform** con **cwd** correcto.
- Prompts: **QuickPick** e **InputBox** para flujos guiados.

Requisitos de Datos

- Historial: persistencia en **globalState** con clave **help-deploy.history**.
- **.gitignore**: añadir entradas Terraform si faltan.

Interfaces Externas

- Git CLI: **git clone**, **rev-parse**, **checkout**, **commit**, **push**, **stash**.
- Terraform CLI: **init**, **plan**, **apply**.

Requisitos No Funcionales

- Usabilidad: flujos breves y autoexplicativos; mensajes claros.
- Rendimiento: comandos disparados en terminal, respuesta inmediata visual.
- Confiabilidad: manejo de errores y validaciones (e.g., ARN de AWS, ramas vacías).
- Seguridad: no se almacenan secretos; credenciales sólo en sesión de terminal.
- Compatibilidad: VS Code **^1.105.0**, Windows PowerShell, Git/Terraform en **PATH**.
- Mantenibilidad: TypeScript + ESLint + TSC, empaquetado con **esbuild** y **vsce**.

Suposiciones y Dependencias

- Usuario con permisos y credenciales válidas para Git y proveedor cloud.
- Terraform y Git instalados y accesibles.

Criterios de Aceptación

- Todos los comandos disponibles y funcionando según flujo.
- Mensajes de commit cumplen ConvCommits.
- Historial persiste y soporta limpiar/copiar.
- **.gitignore** se actualiza tras **apply**.

Riesgos y Mitigaciones

- **cwd** inválido en terminal: ajustar con **cd** y opciones **cwd**.
- Inyección de parámetros: uso de comillas y sanitización (**quotePath**).
- Diferencias de shell: priorizar PowerShell y comandos seguros.