Git Básico

Arturo Silvelo

Try New Roads

Ejercicios Prácticos

Git Hooks y CI/CD

Objetivos

Al finalizar estos ejercicios serás capaz de:

- Entender el problema de los hooks nativos de Git
- Configurar hooks nativos en `.git/hooks/`
- Implementar hooks con Husky
- Comparar ambas soluciones
- Configurar GitHub Actions para CI/CD

Ejercicio 1: Hooks Nativos

Explorando el problema

Archivos necesarios: `hooks.zip`

1. **Extraer y explorar:**

```
unzip hooks.zip
cd hooks-ejemplo
ls -la .git/hooks/
```

2. **Probar los hooks:**

```
# Restaurar permisos (se pierden al comprimir)
chmod +x .git/hooks/*
```

Ejercicio 1

Pre-commit hook:

- Debe fallar si hay archivos muy grandes

```
#!/bin/sh
# .git/hooks/pre-commit

echo "🔍 Ejecutando validaciones pre-commit..."

# Verificar que no hay archivos grandes
for file in $(git diff --cached --name-only); do
    if [ -f "$file" ]; then
        size=$(wc -c < "$file")
        if [ $size -gt 1048576 ]; then  # 1MB
            echo "❌ Archivo muy grande: $file ($size bytes)"
            echo "   Los archivos no pueden superar 1MB"
            exit 1
        fi
    fi
done

echo "✅ Validaciones pre-commit completadas"
exit 0
```

```
commit_file="$1"
commit_msg=$(cat "$commit_file")

echo "📝 Validando mensaje de commit..."

# Verificar formato: tipo(scope): descripción
commit_pattern="^(feat|fix|docs|style|refactor|test|chore)(\(.+\))?: .{1,50}"

if ! echo "$commit_msg" | grep -qE "$commit_pattern"; then
    echo "❌ Formato de commit inválido"
    echo ""
    echo "Formato correcto:"
    echo "  tipo(scope): descripción"
    echo ""
    echo "Tipos válidos:"
    echo "  feat, fix, docs, style, refactor, test, chore"
    echo ""
    echo "Ejemplos:"
    echo "  feat(auth): add login functionality"
    echo "  fix: resolve memory leak in parser"
    echo ""
    exit 1
fi

echo "✅ Formato de commit válido"
exit 0
```

Pre-push hook:

- Debe rechazar push directo a main/master

```
#!/bin/sh
# .git/hooks/pre-push

echo "🚀 Ejecutando validaciones pre-push..."

# Obtener rama actual
current_branch=$(git rev-parse --abbrev-ref HEAD)
echo "📍 Rama actual: $current_branch"

# Prevenir push directo a main/master
protected_branches="main master"
for branch in $protected_branches; do
```