Guía de Git Cómo trabajar en equipo en proyectos: Aprende a usar Git para controlar versiones, colaborar con otros desarrolladores y mantener tu código organizado.

Edison Achalma
Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga

Resumen

Primer parrafo de abstrac

Palabras Claves: keyword1, keyword2

Tabla de contenidos

Edison Achalma https://orcid.org/0000-0001-6996-3364
El autor no tiene conflictos de interés que revelar. Los roles de autor se clasificaron utilizando la taxonomía de roles de colaborador (CRediT; https://credit.niso.org/) de la siguiente manera: Edison Achalma: conceptualización, redacción
La correspondencia relativa a este artículo debe dirigirse a Edison Achalma, Email: elmer.achalma.09@unsch.edu.pe

Guía de Git Cómo trabajar en equipo en proyectos

Guía esencial de Git y GitHub

Esta guía te introduce a los fundamentos de Git y GitHub, desde la instalación hasta la gestión avanzada de proyectos. Ideal tanto para principiantes como para quienes buscan perfeccionar sus habilidades en control de versiones.

Git es un sistema de control de versiones (SCV) esencial para rastrear cambios en el código, colaborar en equipo y experimentar con nuevas ideas mediante ramas. Plataformas como GitHub potencian esta colaboración al hospedar repositorios y facilitar el intercambio de código.

¿Cómo funciona Git?

Git organiza los proyectos en tres áreas principales: - **Directorio de trabajo**: Donde editas tus archivos. - **Área de preparación (staging)**: Donde preparas los cambios antes de confirmarlos. - **Directorio Git**: Almacena las instantáneas confirmadas de tu proyecto.

Disponible en Linux, Windows y macOS, Git tiene una curva de aprendizaje inicial, pero su dominio abre un mundo de posibilidades para gestionar proyectos eficientemente.

Comandos básicos de Git

Aquí tienes los comandos fundamentales para empezar:

1. `git init`: Inicia un nuevo repositorio en el directorio actual.
2. `git clone [url]`: Copia un repositorio existente a tu máquina.
3. `git add [file]`: Añade un archivo al área de preparación.
4. `git commit -m "mensaje"`: Guarda los cambios con un mensaje descriptivo.
5. `git status`: Muestra el estado actual del repositorio.
6. `git log`: Lista el historial de commits.
7. `git diff`: Compara cambios no confirmados con el último commit.
8. `git branch`: Muestra las ramas existentes.
9. `git checkout [branch]`: Cambia a una rama específica.
10. `git merge [branch]`: Fusiona una rama con la actual.
11. `git config --global user.name "tu-nombre"`: Configura tu nombre de usuario.
12. `git config --global user.email "tu-email@example.com"`: Configura tu correo.

Instalación de Git en Ubuntu

Método 1: Paquetes predeterminados (rápido y estable)

1. Verifica si Git está instalado:

```
git --version
```

   Ejemplo de salida: `git version 2.34.1`

2. Si no está instalado, actualiza e instala con APT:

```
sudo apt update
sudo apt install git
```

3. Configura tu identidad:

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu.correo@example.com"
```

4. Verifica la configuración:

```
git config --list
```

Método 2: Desde la fuente (versión más reciente)

1. Instala las dependencias:

```
sudo apt update
sudo apt install libz-dev libssl-dev libcurl4-gnutls-dev libexpat1-dev gettext cmake gc
```

2. Descarga y descomprime la versión deseada (ejemplo: 2.34.1):

```
mkdir tmp && cd tmp
curl -o git.tar.gz https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.34.1.tar.
tar -zxf git.tar.gz
cd git-*
```

3. Compila e instala:

```
make prefix=/usr/local all
sudo make prefix=/usr/local install
exec bash
```

4. Confirma la instalación:

```
git --version
```

Configuración de claves SSH para GitHub

Generar una clave SSH

1. Verifica claves existentes:

```
ls -al ~/.ssh
```

   Si no hay claves, crea el directorio: `mkdir ~/.ssh`.

2. Genera un par de claves:

```
ssh-keygen -t rsa -b 4096 -C "tu.email@example.com"
```

   Acepta el nombre predeterminado y añade una contraseña (opcional).

Añadir la clave a ssh-agent

1. Inicia el agente:

```
eval "$(ssh-agent -s)"
```

2. Añade la clave privada:

```
ssh-add ~/.ssh/id_rsa
```

Vincular la clave a GitHub

1. Copia la clave pública:
   - Linux/Mac: `cat ~/.ssh/id_rsa.pub`
   - Windows: `clip < ~/.ssh/id_rsa.pub`
2. En GitHub, ve a *Settings* > *SSH and GPG keys* > *New SSH key*, pega la clave y guárdala.
3. Prueba la conexión:

```
ssh -T git@github.com
```

Resultado esperado: `Hi tu_usuario! You've successfully authenticated...`

Crear un repositorio local

1. Inicia un repositorio:

```
git init [nombre-del-proyecto]
```

2. Añade archivos y haz un commit:

```
git add .
git commit -m "Primer commit"
```

Clonar un repositorio

Clonación básica

Clona un repositorio remoto:

```
git clone https://github.com/usuario/repositorio.git
```

Clonación en carpeta específica

```
git clone https://github.com/usuario/repositorio.git /ruta/deseada
```

Clonación superficial

Solo las últimas n confirmaciones:

```
git clone --depth=1 https://github.com/usuario/repositorio.git
```

Clonar una rama específica

```
git clone --branch=nombre-rama https://github.com/usuario/repositorio.git
```

Subir un proyecto a GitHub

1. Crea un repositorio en GitHub (público o privado).
2. En tu proyecto local:

```
git init
git add .
git commit -m "Inicial"
git remote add origin git@github.com:usuario/repositorio.git
git push -u origin main
```

3. Si aparece el error remote origin already exists:

```
git remote rm origin
```

Luego repite el paso 2.

Observar el repositorio

- `git status`: Muestra el estado actual.
- `git diff`: Compara cambios no confirmados.
- `git log --oneline`: Historial compacto. Ejemplo:

```
7e320e8 update
```

- git blame [archivo]: Autores y fechas de cambios.

Trabajar con ramas

1. Crea y cambia a una rama:

```
git checkout -b nueva-rama
```

2. Fusiona ramas:

```
git checkout main
git merge nueva-rama
```

3. Etiqueta un commit:

```
git tag v1.0.0
git push origin main --tags
```

Sincronización

- `git fetch origin`: Descarga cambios remotos sin fusionarlos.
- `git pull origin main`: Descarga y fusiona cambios.
- `git push origin main`: Envía cambios locales al remoto.

Conclusión

Dominar Git y GitHub es clave para gestionar proyectos de desarrollo. Practica estos comandos y consulta `git --help` para más detalles.

Publicaciones Similares

Si te interesó este artículo, te recomendamos que explores otros blogs y recursos relacionados que pueden ampliar tus conocimientos. Aquí te dejo algunas sugerencias:

1. Comandos De Informacion Windows
2. Adb
3. Limpieza Y Optimizacion De Pc
4. Usando Apk En Windown 11
5. Gestionar Versiones De Jdk En Kubuntu
6. Instalar Tor Browser
7. Crear Enlaces Duros O Hard Link En Linux
8. Comandos Vim
9. Guia De Git Y Github
10. 00 Primeros Pasos En Linux
11. 01 Introduccion Linux
12. 02 Distribuciones Linux
13. 03 Instalacion Linux
14. 04 Administracion Particiones Volumenes
15. Atajos De Teclado Y Comandos Para Usar Vim
16. Instalando Specitify

Esperamos que encuentres estas publicaciones igualmente interesantes y útiles. ¡Disfruta de la lectura!