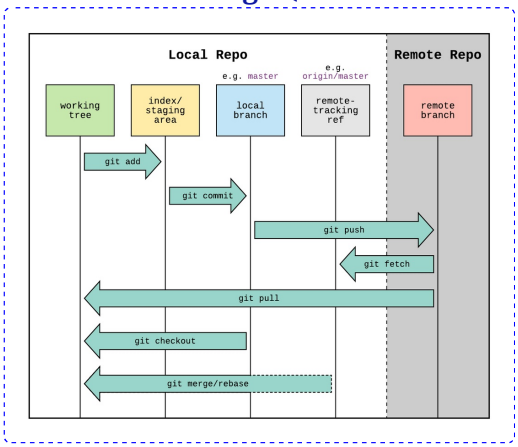
Uso de git en local

Documentación de interese

- git cheat sheet education
- Git and Git Flow Cheat Sheet
- Pro Git book
- Fundamentos de git (ver Áreas de traballo)



- Learn Git Branching

Práctica

Crear o directorio do proxecto

$ mkdir probando-git # Crear unha carpeta chamada probando-git como o directorio do proxecto

$ cd probando-git # Entrar na carpeta do proxecto

Inicializar o repositorio

$ git init # Inicializar un novo repositorio de Git

> Historicamente, a rama principal por defecto chamábase master, pero a partir de setembro de 2020, Git mudou a súa configuración predeterminada para usar main como o nome da rama principal nos novos repositorios creados con git init. Esta decisión foi tomada para promover unha linguaxe máis inclusiva. Se estás creando novos proxectos emprega main, que é o estándar actual en Git. É o que moitas plataformas, como GitHub, configuran por defecto.
> A rama predeterminada pódese definir coa seguinte configuración: `git config --global init.defaultBranch main`
>
> Despois de executar git init, o comando git branch non mostra ningunha rama porque aínda non hai commits no repositorio. En Git, unha rama non existe ata que se crea polo menos un commit.
> Se queres cambiar a rama principal dun proxecto xa existente de **master** a **main**, podes seguir estes pasos:
>
> 1. Renomea a rama local: `git branch -m master main`
> 2. Actualiza o nome no repositorio remoto: `git push origin main`
> 3. Cambia a rama principal por defecto no repositorio remoto (por exemplo, en GitHub, podes facelo desde a interface web).
> 4. Elimina a antiga rama **master** do remoto (opcional): `git push origin --delete master`

Configurar usuario para este proxecto

$ git config user.name "Teu Nome" # Configurar o nome do usuario para definir o nome do usuario que se asociará cos commits que realices no repositorio. Este nome aparecerá como parte da autoría de cada commit.

$ git config user.email "teu.email@example.local" # Configurar o email do usuario. Soe combinarse co comando anterior para configurar a información de autoría dos teus commits.

$ git config -l # Amosa todas as configuracións activas (local, global, e de sistema). Ficheiros: .git/config, ~/.gitconfig, /etc/gitconfig

$ git config --local -l # Amosa as configuracións locais para o repositorio actual. Ficheiro: .git/config

$ git config --global -l # Amosa as configuracións globais. Ficheiro: ~/.gitconfig

$ git config --system -l # Amosa as configuracións do sistema. Ficheiro: /etc/gitconfig

> **Orde de prevalencia**: **Local → Global → System**
>
> **Explicación:**
> As configuracións locais (para o repositorio actual) teñen prioridade sobre as globais e de sistema. Isto significa que, se hai unha configuración tanto no ficheiro local como no global ou de sistema, será a configuración local a que se aplique.
> As configuracións globais teñen prioridade sobre as de sistema, pero só se non hai unha configuración no nivel local.
> **Exemplo:**
> Se configuras un *user.name* de forma global e logo estableces un *user.name* diferente no repositorio actual, a configuración local será a que se aplique para ese repositorio.

Crear un ficheiro e engadilo ao repositorio

$ git status # Ver o estado actual do repositorio

$ echo "Primeiro contido" > ficheiro.txt # Crear un ficheiro con contido inicial no repositorio

$ git status # Ver que hai un novo ficheiro sen rastrexar

$ git add ficheiro.txt # Engadir o ficheiro ao staging area

$ git status # Ver que o ficheiro está en preparación para o commit

$ git rm --cached ficheiro.txt # Elimina o ficheiro do staging area (do control de versións de Git), pero permanece no sistema de arquivos. Se se eliminara do comando a opción --cached eliminariase o ficheiro noo sistema de arquivos como no índice (staging area).

> $ git rm -f ficheiro.txt # Se se eliminara do comando a opción --cached e se engadira a opción -f eliminariase o ficheiro tanto no sistema de arquivos como no índice (staging area).

$ git status # Ver que hai un novo ficheiro sen rastrexar

$ git add ficheiro.txt # Engadir o ficheiro ao staging area

$ git status # Ver que o ficheiro está en preparación para o commit

$ git commit -m "Primeiro commit: Engadir ficheiro.txt" # Crear o primeiro commit

$ git status # Ver que non hai cambios pendentes

Verificar o historial de commits

$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.

Cada commit en Git é identificado de forma única mediante un hash SHA-1 de 40 caracteres, que actúa como un identificador único para ese commit. Este hash é xerado a partir da información do commit, que inclúe:

```
Contido dos ficheiros
Autor do commit
Data e hora do commit
Mensaxe do commit
Hash do commit pai (se existe)
                   (versión curta:
                     . Os primeiros 7 caracteres
                     . Soe ser suficiente para identificar un commit na maioría dos casos)
```

Crear unha nova rama para desenvolver unha funcionalidade

$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.

$ git branch nova-funcionalidade # Crear unha rama chamada nova-funcionalidade

$ git checkout nova-funcionalidade # Cambiar á nova rama

Os 2 anteriores comandos son análogos ao comando: *git checkout -b nova_funcionalidade*

$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.

Modificar o ficheiro e facer outro commit

$ echo "Engadir nova funcionalidade" >> ficheiro.txt # Engadir liña ao ficheiro

$ git status # Ver que hai un novo ficheiro sen rastrexar

$ git add ficheiro.txt # Engadir os cambios ao staging area

$ git status # Ver que o ficheiro está en preparación para o commit

$ git commit -m "Nova funcionalidade: Actualizar ficheiro.txt" # Crear un commit na nova rama

$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.

Volver á rama principal

$ git checkout master # Cambiar á rama principal

Fusionar os cambios da nova rama na rama principal

$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.

$ git merge nova-funcionalidade # Fusionar a rama nova-funcionalidade en main

Verificar o historial de commits

$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.

$ git log -2 # Amosa os últimos 2 commits do historial de Git. Se queres ver máis ou menos commits, podes cambiar o número

$ git log -2 --name-only # Amosa os últimos 2 commits xunto cos nomes dos ficheiros modificados en cada un deles, sen mostrar os detalles completos do commit.

$ git log -1 -p # Amosa o diff do último commit, é dicir, amosa os cambios feitos no ficheiro.

$ git log -- ficheiro.txt # Ver os commits que fan referencia unicamente ao ficheiro ficheiro.txt

Crear un novo ficheiro e movelo

$ echo "Contido adicional" > outro.txt # Crear un segundo ficheiro

$ git status # Ver que hai un novo ficheiro sen rastrexar

$ git add outro.txt # Engadir o ficheiro ao staging area

$ git status # Ver que o ficheiro está en preparación para o commit

$ git commit -m "Engadir outro ficheiro" # Crear un commit

$ git mv outro.txt cambiado.txt # Renomear o ficheiro

$ git status # Ver que hai un ficheiro renomeado que está en preparación para o commit

$ git commit -m "Renomear outro.txt a cambiado.txt" # Commit do cambio de nome

Amosar as diferencias entre commits

$ git diff HEAD~1 HEAD # Ver as diferenzas co commit anterior
$ git diff c0d3fde 520a484 # Ver as diferenzas entre 2 commits (o commit c0d3fde e o commit 520a484)

Simular un cambio crítico e correxilo

$ git checkout -b hotfix # Crear unha rama hotfix para correccións urxentes
$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.
$ echo "Corrección crítica" >> ficheiro.txt # Engadir unha corrección
$ git status # Ver que hai un novo ficheiro sen rastrexar
$ git add ficheiro.txt # Engadir os cambios
$ git status # Ver que o ficheiro está en preparación para o commit
$ git commit -m "Corrección crítica na rama hotfix" # Crear un commit de corrección
$ git checkout master # Cambiar de volta á rama principal
$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.
$ git merge hotfix # Fusionar a corrección na rama principal

Eliminar ramas que xa non se necesitan

$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.
$ git branch -d nova-funcionalidade # Eliminar a rama nova-funcionalidade
$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.
$ git branch -d hotfix # Eliminar a rama hotfix
$ git branch # Lista todas as ramas locais. A rama actual aparece marcada con * antes do nome.

Reverter un ficheiro a un commit específico

$ cat ficheiro.txt # Ver o contido do ficheiro.txt
$ git log -- ficheiro.txt # Ver os commits que fan referencia unicamente ao ficheiro ficheiro.txt
$ git restore --source *commit-hash* -- ficheiro.txt # Isto non afecta a outros ficheiros, só restaura ficheiro.txt ao estado que tiña no commit especificado
$ cat ficheiro.txt # Ver o contido do ficheiro.txt
$ git status # Ver que hai un novo ficheiro sen rastrexar

Reverter un ficheiro ao último commit confirmado se o ficheiro non foi agregado ao staging area

$ cat ficheiro.txt # Ver o contido do ficheiro.txt
$ git restore ficheiro.txt # Útil cando se fixeron cambios per aínda non foi executado o comando: git add
$ cat ficheiro.txt # Ver o contido do ficheiro.txt
$ git status # Ver o estado actual do repositorio

Reverter un ficheiro se o ficheiro xa foi agregado ao staging area pero aínda non foi confirmado

$ cat ficheiro.txt # Ver o contido do ficheiro.txt
$ git restore --source *commit-hash* -- ficheiro.txt # Isto non afecta a outros ficheiros, só restaura ficheiro.txt ao estado que tiña no commit especificado
$ cat ficheiro.txt # Ver o contido do ficheiro.txt
$ git add ficheiro.txt # Engadir o ficheiro ao staging area
$ git status # Ver que o ficheiro está en preparación para o commit
$ git reset HEAD ficheiro.txt # Saca o ficheiro do staging area
$ git status # Ver que hai un novo ficheiro sen rastrexar
$ git restore ficheiro.txt # Reverte os cambios locais. (Agora estamos no exemplo anterior)
$ cat ficheiro.txt # Ver o contido do ficheiro.txt

Desfacer todos os cambios a un commit específico

$ ls -l # Listar o contido do directorio co formato extendido.
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
$ git reset --hard commit_hash # Este comando é destructivo e elimina todos os cambios posteriores ao commit especificado
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
$ ls -l # Listar o contido do directorio co formato extendido.

Desfacer todos os cambios a un commit específico sen perder os cambios no código nin no staging area

```
$ git status # Ver o estado actual do repositorio
$ echo "Engadir nova entrada" >> ficheiro.txt # Engadir liña ao ficheiro
$ git status # Ver que hai un novo ficheiro sen rastrexar
$ git add ficheiro.txt # Engadir os cambios ao staging area
$ git status # Ver que o ficheiro está en preparación para o commit
$ git commit -m "Nova entrada no ficheiro: Actualizar ficheiro.txt" # Crear un commit
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
$ echo "Engadir outra entrada" >> ficheiro.txt # Engadir liña ao ficheiro
$ git status # Ver que hai un novo ficheiro sen rastrexar
$ git add ficheiro.txt # Engadir os cambios ao staging area
$ git status # Ver que o ficheiro está en preparación para o commit
$ git commit -m "Outra entrada no ficheiro: Actualizar ficheiro.txt" # Crear un commit
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
$ git reset --soft commit_hash # Borra os commits despois de commit-hash, pero deixa os cambios en staging
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
$ git status # Ver o estado actual do repositorio
```

Reverter o último commit sen perder os cambios

```
$ git status # Ver o estado actual do repositorio
$ echo "Engadir entrada de novo" >> ficheiro.txt # Engadir liña ao ficheiro
$ git status # Ver que hai un novo ficheiro sen rastrexar
$ git add ficheiro.txt # Engadir os cambios ao staging area
$ git status # Ver que o ficheiro está en preparación para o commit
$ git commit -m "Entrada de novo no ficheiro: Actualizar ficheiro.txt" # Crear un commit
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
$ echo "Engadir unha entrada máis" >> ficheiro.txt # Engadir liña ao ficheiro
$ git status # Ver que hai un novo ficheiro sen rastrexar
$ git add ficheiro.txt # Engadir os cambios ao staging area
$ git status # Ver que o ficheiro está en preparación para o commit
$ git commit -m "Unha entrada máis no ficheiro: Actualizar ficheiro.txt" # Crear un commit
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
$ git reset --soft HEAD~1 # Reverter soamente o último commit. O código segue en staging, listo para facer outro commit.
$ git status # Ver o estado actual do repositorio
$ git log --oneline --graph # Amosa o historial de commits de forma resumida e gráfica, indicando a estrutura de branches e merges.
```

Ricardo Feijoo Costa