Guia de Git

Inicializando o repositório

Criar um novo repositório

Se você deseja iniciar um novo repositório local, use o comando:

```
it init
```

Isso cria um repositório Git vazio no diretório atual.

Clonar um repositório existente

Para clonar um repositório remoto para seu ambiente local:

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

Fluxo básico de trabalho

1. Verificar o status do repositório

Antes de começar a fazer mudanças, sempre é bom verificar o status do seu repositório:

```
git status
```

Isso mostra quais arquivos foram modificados, quais estão prontos para commit, e se há alterações que não foram rastreadas.

2. Adicionar arquivos para o commit

Após fazer mudanças em seus arquivos, você precisa adicioná-los ao *staging area* antes de fazer um commit:

```
git add nome-do-arquivo.txt
```

Ou para adicionar todos os arquivos modificados de uma vez:

```
git add .
```

3. Fazer commit das mudanças

Depois de adicionar os arquivos ao *staging*, você pode gravar essas mudanças com um commit:

```
git commit -m "Mensagem explicativa do commit"
```

O `-m` é usado para adicionar uma mensagem descritiva ao commit. Mantenha as mensagens claras e concisas.

Trabalhando com branches (ramificações)

1. Criar uma nova branch

Branches permitem que você trabalhe em funcionalidades ou correções sem interferir no código principal:

```
git branch minha-nova-branch
```

2. Mudar para uma branch existente

Depois de criar ou se quiser trabalhar em uma branch já existente:

```
git checkout minha-nova-branch
```

No Git moderno, você pode usar o comando `switch` para evitar confusões:

```
git switch minha-nova-branch
```

3. Criar e mudar para a branch de uma vez

Uma forma eficiente de criar e já mudar para a nova branch:

```
git checkout -b minha-nova-branch
```

4. Mesclar branches

Depois de concluir seu trabalho em uma branch, você pode mesclar suas mudanças na branch principal (geralmente a `main` ou `master`):

```
git checkout main
git merge minha-nova-branch
```

Isso aplica as mudanças da branch de trabalho na branch principal.

Trabalhando com repositórios remotos

1. Verificar os repositórios remotos

Você pode verificar os repositórios remotos associados ao seu repositório local com:

```
git remote -v
```

2. Puxar mudanças do repositório remoto

Antes de começar a trabalhar, é uma boa prática puxar as mudanças mais recentes do repositório remoto:

```
git pull origin main
```

Isso sincroniza sua branch local com as mudanças mais recentes da branch remota `main`.

3. Enviar mudanças para o repositório remoto

Após fazer commit das suas mudanças localmente, envie-as para o repositório remoto:

```
git push origin minha-nova-branch
```

Isso envia as mudanças da branch atual para o repositório remoto.

Desfazer mudanças

1. Desfazer mudanças não comitadas

Se você deseja desfazer mudanças em um arquivo modificado, mas que ainda não foi adicionado ao *staging*, use:

```
git checkout nome-do-arquivo.txt
```

2. Resetar o *staging*

Se você já adicionou o arquivo ao *staging* mas ainda não fez o commit, você pode removê-lo do *staging*:

```
git reset nome-do-arquivo.txt
```

3. Reverter um commit

Se você cometeu um erro em um commit, você pode desfazê-lo com:

```
git revert id-do-commit
```

Isso cria um novo commit que reverte as mudanças do commit anterior, sem alterar o histórico do repositório.

Comandos úteis adicionais

Ver o log de commits

Você pode ver o histórico dos commits com:

```
git log
```

Para um histórico mais compacto:

```
git log --oneline
```

Comparar mudanças entre commits

Você pode ver as diferenças entre commits com:

```
git diff commit1 commit2
```

Remover um arquivo do Git (sem deletá-lo do sistema)

Se você quer parar de rastrear um arquivo, mas mantê-lo no diretório:

```
git rm --cached nome-do-arquivo.txt
```
