Guia Prático de Git: Comandos Essenciais para Iniciantes

Este guia oferece um conjunto de exercícios práticos para iniciantes em Git, abordando desde a configuração inicial até a colaboração remota. Através de exemplos detalhados e desafios, você aprenderá a utilizar os comandos essenciais do Git no Windows e Git Bash, preparando-se para o controle de versão eficiente e o trabalho colaborativo em projetos de software.

por Wellington Oliveira

Configuração Inicial do Git

Antes de começar a usar o Git, é necessário configurar seu nome de usuário e endereço de e-mail. Essas informações serão associadas a cada commit que você fizer, permitindo que outros colaboradores saibam quem fez cada alteração.

1. Abra o Git Bash.
2. Execute os seguintes comandos, substituindo "Seu Nome" e "seu@email.com" pelas suas informações:

```
git config --global user.name "Seu Nome"
git config --global user.email "seu@email.com"
```

Para verificar se as configurações foram aplicadas corretamente, execute o comando:

```
git config --global --list
```

Este comando exibirá uma lista de todas as configurações globais do Git, incluindo seu nome de usuário e e-mail.

Inicialização e Status do Repositório

Para começar a controlar as versões de um projeto, você precisa inicializar um repositório Git. Siga os passos abaixo:

1. Crie uma nova pasta para o seu projeto e acesse-a no Git Bash:

```
mkdir meu-projeto && cd meu-projeto
```

1. Inicialize um repositório Git dentro da pasta:

```
git init
```

Este comando cria uma pasta oculta chamada ".git" dentro do diretório do seu projeto. Essa pasta contém todos os metadados e o histórico de versões do seu repositório.

Para verificar o status do repositório recém-criado, use o comando:

```
git status
```

Este comando mostra quais arquivos foram modificados, adicionados ou removidos, e se há alguma alteração que ainda não foi commitada.

Adicionando e Confirmando Arquivos

Agora que você tem um repositório Git inicializado, pode começar a adicionar arquivos e confirmar as alterações. Veja como:

1. Crie um arquivo de texto chamado "arquivo1.txt":

```
touch arquivo1.txt
```

1. Adicione algum conteúdo ao arquivo, usando o comando echo ou editando manualmente:

```
echo "Meu primeiro arquivo no Git" > arquivo1.txt
```

1. Verifique novamente o status do repositório:

```
git status
```

Você verá que o arquivo "arquivo1.txt" aparece como "Untracked files" (arquivos não rastreados).

1. Adicione o arquivo ao stage (área de preparação):

```
git add arquivo1.txt
```

1. Confirme a mudança no repositório com uma mensagem de commit:

```
git commit -m "Adicionando arquivo1.txt ao repositório"
```

A mensagem de commit deve ser concisa e descrever as alterações que você fez.

Histórico de Commits e Modificações

O Git permite que você acompanhe o histórico de todas as alterações feitas no seu projeto. Para visualizar o histórico de commits, use o comando:

```
git log --oneline
```

Este comando exibe uma lista de commits, mostrando o hash (identificador único) e a mensagem de cada commit.

Para ver as diferenças entre a versão atual de um arquivo e a versão anterior, siga estes passos:

1. Modifique o conteúdo do arquivo "arquivo1.txt":

```
echo "Modificação no arquivo" >> arquivo1.txt
```

1. Visualize a diferença:

```
git diff
```

O comando "git diff" mostra as linhas que foram adicionadas, removidas ou modificadas.

1. Adicione e confirme as mudanças:

```
git add arquivo1.txt
git commit -m "Atualizando arquivo1.txt"
```

Trabalhando com Branches

Branches são ramificações do seu projeto que permitem que você trabalhe em novas funcionalidades ou correções de bugs sem afetar a branch principal (geralmente chamada de "main").

Para criar uma nova branch chamada "feature-x" e alternar para ela, execute os seguintes comandos:

```
git branch feature-x
git checkout feature-x
```

Agora você está trabalhando na branch "feature-x". Faça as alterações que precisar, adicione os arquivos e confirme as mudanças:

```
touch arquivo2.txt
echo "Novo arquivo na branch feature-x" > arquivo2.txt
git add arquivo2.txt
git commit -m "Adicionando arquivo2.txt na branch feature-x"
```

Para retornar para a branch principal ("main"), use o comando:

```
git checkout main
```

Mesclagem de Branches e Repositórios Remotos

Após concluir o trabalho na branch "feature-x", você pode mesclá-la na branch principal ("main"). Para isso, execute o comando:

```
git merge feature-x
```

Se houver conflitos, o Git avisará e você precisará resolvê-los manualmente. Após resolver os conflitos, adicione os arquivos modificados e confirme a mesclagem.

Depois de mesclar a branch, você pode excluí-la:

```
git branch -d feature-x
```

Para conectar seu repositório local a um repositório remoto (como no GitHub), siga estes passos:

1. No GitHub, crie um novo repositório chamado "meu-projeto-remoto".
2. Conecte seu repositório local ao GitHub:

```
git remote add origin https://github.com/seu-usuario/meu-projeto-remoto.git
```

1. Envie os commits locais para o repositório remoto:

```
git push -u origin main
```

A opção "-u" configura a branch "main" como a branch de rastreamento upstream.

Clonando e Atualizando Repositórios

Para clonar um repositório remoto para sua máquina, use o comando:

```
git clone https://github.com/usuario/repo-exemplo.git
```

Isso cria uma cópia local do repositório remoto, incluindo todo o histórico de commits e branches.

Para atualizar seu repositório local com as últimas alterações do repositório remoto, use o comando:

```
git pull origin main
```

Isso busca as alterações do repositório remoto e as mescla na sua branch local "main".

Com estes exercícios, você terá uma base sólida para usar o Git em seus projetos e colaborar com outros desenvolvedores. Experimente e pratique para se familiarizar com os comandos e os fluxos de trabalho do Git.