



Primeiros Passos com Git Bash

Seu Guia para o Controle de Versão

O que é Git e por que usar o Git Bash?



O que é Git?

Sistema de controle de versão distribuído (DVCS) mais popular do mundo. É essencial para rastrear mudanças no código-fonte e coordenar o trabalho entre múltiplos desenvolvedores de forma eficiente e organizada.



O que é Git Bash?

É um aplicativo para ambientes Windows que fornece uma camada de emulação de shell Unix. Permite executar comandos Git e comandos comuns do Linux (como `ls`, `cd`, `mkdir`) diretamente no Windows.

Oferece uma experiência de linha de comando consistente, independentemente do sistema operacional.



Por que é importante?

Facilita a colaboração entre desenvolvedores, o rastreamento completo do histórico de alterações e a reversão de erros de forma segura. É o padrão da indústria para desenvolvimento de software moderno.

Configuração Essencial: Quem é Você para o Git?

Importante: Antes de qualquer commit, o Git precisa saber quem está fazendo a alteração. Esta configuração é feita apenas uma vez.

1. Configurar Nome de Usuário

```
git config --global user.name "Seu Nome Completo"
```

Define o nome que será anexado aos seus commits. O parâmetro `--global` aplica a configuração a todos os seus projetos Git na máquina.

2. Configurar Email

```
git config --global user.email "seu.email@exemplo.com"
```

Define o endereço de e-mail que será anexado aos seus commits. É crucial que este e-mail seja o mesmo usado em plataformas como GitHub, GitLab ou Bitbucket.

3. Verificar Configurações

```
git config --list
```

Lista todas as configurações atuais do Git para confirmar que tudo foi configurado corretamente.

Os Pilares do Git: Commit e Branch

Commit

Definição

É um "instantâneo" (snapshot) do seu repositório em um momento específico no tempo.

Função

Registra as alterações no histórico do projeto, criando um ponto de referência que pode ser acessado ou restaurado posteriormente.

Mensagem

Cada commit deve ter uma mensagem clara e concisa que descreva as alterações feitas.

Branch (Ramo)

Definição

Uma linha de desenvolvimento independente. Pense nela como um ponteiro móvel para um commit.

Função

Permite trabalhar em novos recursos ou correções de bugs isoladamente, sem afetar a linha principal de desenvolvimento.

Melhor Prática

O branch principal (`main` ou `master`) deve sempre conter código estável e pronto para produção.

Iniciando um Repositório e Verificando o Status



Inicializar Repositório

```
git init
```

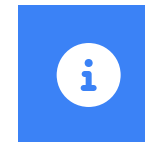
Transforma o diretório atual em um repositório Git. Cria a pasta oculta `.git` que contém todo o histórico e configurações.



Clonar Repositório

```
git clone <URL>
```

Baixa uma cópia completa de um repositório remoto para sua máquina local, incluindo todo o histórico de commits.



Verificar Status

```
git status
```

Mostra o estado dos arquivos no diretório de trabalho e na área de staging. Indica modificações, arquivos prontos para commit e o branch atual.

Preparando e Registrando Alterações

Área de Staging (Index)

Definição

Uma área intermediária onde você prepara as alterações que deseja incluir no próximo commit.

Função

Permite que você faça commits de apenas um subconjunto das suas alterações, dando controle total sobre o que será registrado no histórico.

Comandos Práticos

Adicionar Arquivos à Staging

```
git add <nome_do_arquivo>
```

```
git add .
```

Move as alterações do diretório de trabalho para a área de staging. O ponto (.) adiciona todos os arquivos modificados/novos.

Realizar Commit

```
git commit -m "Mensagem descritiva"
```

Registra as alterações da área de staging no histórico do repositório. A mensagem é obrigatória e deve ser clara e descritiva.

Criando e Navegando entre Branches



Listar Branches

```
git branch
```

Mostra todos os branches locais. O branch atual é marcado com um asterisco (*****).



Mudar para um Branch

```
git checkout <nome_do_branch>
```

Move o ponteiro **HEAD** para o branch especificado, alterando os arquivos no diretório de trabalho.



Criar um Novo Branch

```
git branch <nome_do_novo_branch>
```

Cria um novo branch, mas não muda para ele automaticamente.



Criar e Mudar (Atalho)

```
git checkout -b <nome_do_novo_branch>
```


Cria um novo branch e muda para ele imediatamente. Comando mais usado na prática!

Sincronizando com o Mundo: Push e Pull

Repositório Remoto

Definição: A versão do seu repositório hospedada em um servidor (ex: GitHub, GitLab, Bitbucket).

Função: Centraliza o código para colaboração entre equipes e serve como backup seguro do projeto.

 O nome padrão do repositório remoto é **origin**. Ele é criado automaticamente quando você clona um repositório.

↑ Enviar Alterações (Push)

```
git push origin <nome_do_branch>
```

Envia seus commits locais para o repositório remoto, tornando-os disponíveis para outros colaboradores.

↓ Puxar Alterações (Pull)

```
git pull origin <nome_do_branch>
```

Baixa as alterações do repositório remoto e as mescla automaticamente com seu branch local.

Adicionar um Remote

```
git remote add origin <URL_do_repositorio>
```

Conecta seu repositório local a um repositório remoto pela primeira vez.

Tabela de Comandos Essenciais

Comando	Função Principal
<code>git config</code>	Configura o usuário e o ambiente Git.
<code>git init</code>	Inicializa um novo repositório Git.
<code>git clone</code>	Baixa um repositório remoto.
<code>git status</code>	Mostra o estado dos arquivos.
<code>git add .</code>	Adiciona todos os arquivos à área de <i>staging</i> .
<code>git commit -m "msg"</code>	Registra as alterações no histórico.
<code>git branch</code>	Lista, cria ou deleta branches.
<code>git checkout</code>	Muda entre branches ou restaura arquivos.
<code>git push</code>	Envia commits para o repositório remoto.
<code>git pull</code>	Baixa e mescla alterações do repositório remoto.

Onde Continuar a Jornada Git

1

Merge e Rebase

Aprenda a mesclar branches usando `git merge` para combinar históricos de desenvolvimento e `git rebase` para reescrever o histórico de commits de forma linear e organizada.

2

Ignorando Arquivos

Use o arquivo `.gitignore` para especificar quais arquivos e pastas o Git deve ignorar, como dependências, arquivos de configuração local e arquivos temporários.

3

Plataformas de Hospedagem

Explore plataformas como [GitHub](#), [GitLab](#) e [Bitbucket](#) para hospedar seus projetos, colaborar com outros desenvolvedores e aproveitar recursos como pull requests, issues e CI/CD.



Dúvidas?

Comece a praticar hoje mesmo!
O Git é uma habilidade fundamental para todo desenvolvedor.

Obrigado!