

Git é um sistema distribuído open source de controlo de versões que permite interagir com o GitHub no seu portátil ou desktop. Este folheto de ajuda resume as instruções frequentemente usadas na linha de comando do Git para referência rápida.

INSTALAR O GIT

O GitHub disponibiliza clientes desktop que incluem um interface gráfico para as ações mais comuns. Estes clientes também incluem ferramentas para a linha de comandos para cenários mais avançados. Em ambos os casos, os clientes são atualizados automaticamente.

GitHub para Windows

<https://windows.github.com>

GitHub para Mac

<https://mac.github.com>

Distribuições de Git para Linux e sistemas POSIX estão disponíveis no site oficial do Git SCM.

Git para todas as plataformas

<http://git-scm.com>

CONFIGURAR O GIT

Configurar os detalhes de utilizador ao nível do sistema:

```
$ git config --global user.name "[nome]"
```

Configura o nome que ficará associado aos Git commits

```
$ git config --global user.email "[endereço-de-email]"
```

Configura o email que ficará associado aos Git commits

CRIAR REPOSITÓRIOS

Criar um novo repositório ou obter um já existente a partir do URL:

```
$ git init [nome-do-projeto]
```

Cria um novo repositório local com o nome de projecto especificado

```
$ git clone [url]
```

Faz download de um projeto, incluindo toda a sua história

FAZER ALTERAÇÕES

Ver estado das alterações e criar um commit

```
$ git status
```

Lista todos os ficheiros modificados no projecto atual

```
$ git diff
```

Mostra as diferenças linha a linha dos ficheiros alterados (pre-stage)

```
$ git add [ficheiro]
```

Adiciona um ficheiro à área de preparação para que possa ser incluído em commits (staged)

```
$ git diff --staged
```

Mostra as diferenças linha a linha dos ficheiros preparados e alterados (pre-stage)

```
$ git reset [ficheiro]
```

Remove o ficheiro da área de preparação (staged) sem o remover do sistema de ficheiros

```
$ git commit -m "[mensagem descritiva]"
```

Guarda as alterações preparadas permanentemente no histórico de versões

BRANCHING (HISTÓRIA PARALELA)

Crie uma versão paralela do histórico de versões para trabalhar em isolamento.

```
$ git branch
```

Lista todos os branches locais no repositório atual

```
$ git branch [nome-do-branch]
```

Cria um novo branch

```
$ git checkout [nome-do-branch]
```

Muda para o branch especificado e atualiza o diretório de trabalho

```
$ git merge [nome-do-branch]
```

Combina a história do branch especificado com o branch atual

```
$ git branch -d [nome-do-branch]
```

Remove o branch especificado



GITHUB FOLHA DE DICAS DE GIT

ALTERAÇÃO DE FICHEIROS

Mover e remover os ficheiros já versionados

```
$ git rm [ficheiro]
```

Marca o ficheiro para remoção do repositório local e remove o ficheiro do sistema de ficheiros (staged)

```
$ git rm --cached [ficheiro]
```

Remove o ficheiro do controlo de versão mas preserva-o no diretório de trabalho

```
$ git mv [ficheiro-original] [ficheiro-renomeado]
```

Muda o nome do ficheiro e o prepara-o para o commit

IGNORAR FICHEIROS

Ignore ficheiros e directórios temporários

```
*.log  
build/  
temp-*
```

Um ficheiro `.gitignore` na raiz do projecto previne o versionamento acidental de ficheiros e directórios correspondentes aos padrões especificados

```
$ git ls-files --other --ignored --exclude-standard
```

Lista todos os ficheiros e directórios ignorados neste projeto

GUARDAR ALTERAÇÕES TEMPORARIAMENTE

Guarde e restaure mudanças temporárias

```
$ git stash
```

Armazena temporariamente todos os ficheiros modificados

```
$ git stash pop
```

Restaura os ficheiros previamente guardados na stash

```
$ git stash list
```

Lista todos os conjuntos de alterações na stash

```
$ git stash drop
```

Remove os últimos ficheiros guardados na stash

NAVEGAR O HISTÓRICO DE VERSÕES

Navegue e inspecione a história do projeto

```
$ git log
```

Lista o histórico de versões para o branch atual

```
$ git log --follow [ficheiro]
```

Lista o histórico de versões para um ficheiro, incluindo mudanças de nome

```
$ git diff [primeiro-branch]...[segundo-branch]
```

Mostra a diferença de conteúdo entre dois branches

```
$ git show [commit]
```

Mostra as mudanças de conteúdo e metadada do commit especificado

MANIPULAR A HISTÓRIA DE VERSÕES

Remove commits e refazer a história de versões

```
$ git reset [commit]
```

Aponta o ambiente de trabalho para o commit especificado, preservando alterações locais e descartando o histórico de versões posteriores.

```
$ git reset --hard [commit]
```

Aponta o ambiente de trabalho para o commit especificado, sem preservar alterações e descartando o histórico de versões posteriores.

SINCRONIZAR ALTERAÇÕES

Atualiza e combina alterações entre repositórios

```
$ git fetch [remote]
```

Faz download de todo o histórico de um repositório remoto

```
$ git merge [remote]/[branch]
```

Combina a história do branch especificado com o branch atual.

```
$ git push [remote] [branch]
```

Envia todos os commits do branch local para o GitHub

```
$ git pull
```

Download e merge num só comando. Igual a executar:
`git fetch`
`git merge`

GitHub Training

Aprenda mais sobre GitHub e Git. Envie um email para a equipa de serviços profissionais ou visite o nosso site para ver a agenda de eventos ou a disponibilidade de cursos específicos.

✉ training@github.com

🌐 training.github.com