

GITHUB

CRIAR UMA CHAVE SSH:

→ Abrir o terminal e utilizar o comando:

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

→ Após utilizado, ir para:

```
cd ~/.ssh/
```

→ Utilize o CAT para ler a chave pub do SSH

→ Copie essa chave e adicione no seu GitHub.

CONFIGURE A FERRAMENTA:

Configure informações de usuário para todos os repositórios locais

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

Configura o nome que você quer ligado às suas transações de commit

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

Configura o email que você quer ligado às suas transações de commit

Crie repositórios

Inicie um novo repositório ou obtenha de uma URL existente

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

Cria um novo repositório local com um nome especificado

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

Baixa um projeto e seu histórico de versão inteiro

REFATORE NOMES DE ARQUIVOS

Mude e remova os arquivos versionados

```
$ git rm [arquivo]
```

Remove o arquivo do diretório de trabalho e o prepara a remoção

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

Remove o arquivo do controle de versão mas preserva o arquivo localmente

\$ git mv [arquivo-original] [arquivo-renomeado]

Muda o nome do arquivo e o prepara para o commit

SUPRA O MONITORAMENTO:

Ignore arquivos e diretórios temporários

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

Um arquivo de texto chamado .gitignore suprime o versionamento acidental de arquivos e diretórios correspondentes aos padrões especificados

\$ git ls-files --others --ignored --exclude-standard

Lista todos os arquivos ignorados neste projeto

SALVE FRAGMENTOS:

Archive e restaure mudanças incompletas

\$ git stash

Armazena temporariamente todos os arquivos monitorados modificados

\$ git stash pop

Restaura os arquivos recentes em stash

\$ git stash list

Lista todos os conjuntos de alterações em stash

\$ git stash drop

FAÇA MUDANÇAS:

Revise edições e crie uma transação de commit

\$ git status

Lista todos os arquivos novos ou modificados para serem commitados

\$ git diff

Mostra diferenças no arquivo que ainda não foram preparadas

\$ git add [arquivo]

Faz o snapshot de um arquivo na preparação para versionamento

\$ git diff --staged

Mostra a diferença entre arquivos preparados e suas últimas versões

\$ git reset [arquivo]

Retira o arquivo da área de preparação, mas preserva seu conteúdo

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

Grava o snapshot permanentemente do arquivo no histórico de versão

MUDANÇA EM GRUPO:

Nomeie uma série de commits e combine os esforços completos

\$ git branch

Lista todos os branches locais no repositório atual

\$ git branch [nome-do-branch]

Cria um novo branch

\$ git switch -c [nome-do-branch]

Muda para o branch especificado e atualiza o diretório de trabalho

\$ git merge [nome-do-branch]

Combina o histórico do branch especificado ao branch atual

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

Exclui o branch especificado

Descarta os conjuntos de alterações mais recentes em stash

REVISE O HISTÓRICO:

Navegue e inspecione a evolução dos arquivos do projeto

\$ git log

Lista o histórico de versões para o branch atual

\$ git log --follow [arquivo]

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

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

Mostra a diferença de conteúdo entre dois branches

\$ git show [commit]

Retorna mudanças de metadata e conteúdo para o commit especificado

DESFAÇA COMMITS:

Apague enganos e crie um histórico substituto

\$ git reset [commit]

Desfaz todos os commits depois de [commit], preservando mudanças locais

\$ git reset --hard [commit]

Descarta todo histórico e mudanças para o commit especificado

SINCRONIZE MUDANÇAS:

Registre um repositório remoto e troque o histórico de versão

\$ git fetch [nome-remoto]

Baixe todo o histórico de um repositório remoto

\$ git merge [nome-remoto]/[branch]

Combina o branch remoto ao branch local atual

\$ git push [alias] [branch]

Envia todos os commits do branch local para o GitHub

\$ git pull

Baixa o histórico e incorpora as mudanças