# Básico de Git

O Git permite acompanhar alterações feitas em um projeto. O Git registra as alterações feitas em um projeto.

## Git Workflow

Podemos separar o Git Workflow em três partes:

1. **Diretório de Trabalho:** Onde é feito toda a organização do trabalho, como criar, editar ou excluir arquivos.
2. **Área de Preparação:** Onde serão listadas todas as alterações feitas no **Diretório de Trabalho**.
3. **Repositório:** Onde o Git armazena permanentemente todas as alterações e versões do projeto.

## Inicializando o Git

O comando **‘git init’** inicializa o Git, habilitando todas as ferramentas necessárias.

## ‘git status’

Mostra os status. Arquivos verdes são arquivos novos. Os arquivos vermelhos são arquivos não “comitados”.

## ‘git add’

Adiciona o arquivo à área de preparação. A sintaxe é a seguinte:

**‘git add <nome\_arquivo>’**

## ‘git diff’

Compara um arquivo em sua versão na área de trabalho com o mesmo arquivo na área de preparação, mostrando suas diferenças.

A sintaxe é a seguinte:

**‘git diff <nome\_arquivo>’**

## ‘git commit’

Armazena permanentemente as alterações da área de preparação para o repositório.

A sintaxe é a seguinte:

**‘git commit -m “Mensagem sobre as alterações”’**

A mensagem deve respeitar as seguintes regras:

* Deve estar entre aspas.
* Resume o que foi feito naquela versão.
* Deve possuir 50 caracteres ou menos.
* Deve vir depois do **‘-m’**.

O parâmetro **‘-a’**, ou **‘--all’**, adiciona automaticamente todas as alterações já rastreadas (tracked) à área de preparação antes de fazer o commit.

## ‘git log’

Permite consultar as versões anteriores do projeto. Todos os commits são armazenados em ordem cronológica no repositório, e o comando **‘git log’** permite visualizá-los.

Sua sintaxe é:

**‘git log’**

Esse comando retornará algumas informações:

* Um texto em laranja que possui 40 caracteres chamado *SHA*. Esse texto especifica o commit.
* O autor do commit.
* A data e o horário do commit.
* A mensagem do commit.

# Repositório Remoto

## ‘master’ para ‘main’

Use **‘git branch -M main’** para converter a branch principal de **‘master’** para **‘main’**.

## Adicionar Repositório Remoto

É necessário vincular o repositório remoto ao repositório local. Para isso, use:

**‘git remote add origin <URL\_repositorio>’**

## ‘git push’

Envia todas as alterações ao repositório remoto.

A sintaxe é a seguinte:

**‘git push -u origin <nome\_branch>’**

O parâmetro **‘-u’** é usado apenas no primeiro push de um branch específico e cria uma associação entre o branch local e o branch remoto.

Caso esteja trabalhando na branch principal, use ‘main’ como nome da branch. Dessa forma:

**‘git push -u origin main’**

# Branches

O comando **‘git branch <nome\_da\_nova\_branch>’** cria uma nova branch com o nome especificado.

## Alternância de Branch

Para alternar para outra branch, use **‘git checkout <nome\_branch>’**.

Caso deseje criar uma nova branch e ao mesmo tempo já alternar para ela, use o parâmetro **‘-b’**.

**‘git checkout -b <nome\_da\_nova\_branch>’**

## Listar Branches

O comando **‘git branch’** lista todas as branches. A branch marcada com um asterisco (**‘\*’**) é a branch selecionada atualmente.

## Deletar Branch

Para excluir uma branch, use:

**‘git branch -d <nome\_da\_branch>’**

Caso haja alterações não “comitadas”, não será possível excluir essa branch com ‘-d’, ao invés disso, use **‘-D’**, que forçará a exclusão.

## Merge

Mescla uma branch à outra. Por exemplo:

**‘git merge <nome\_da\_branch>’**

A branch que deve receber as novas alterações deve ser a selecionada no momento do merge.

## Rebase

Replica as alterações da branch atual em outra branch, fazendo que a nova branch seja exatamente igual a essa.

**‘git rebase <branch\_a\_ser\_replicada>’**

Lembre-se que o comando deve ser executado na branch a receber as alterações, e não a que vai dá-las.