GIT

Informações utilitárias:

- Para criar uma chave ssh no GitHub
- Documentação Oficial

Comandos básicos:

```
git clone <URL> (baixa o repositório remoto)
git init (cria um repositório na pasta em que se encontra);
git init repository (cria uma pasta e um repositório na mesma);
```

Adiciona arquivos ao repositório:

```
git add Arquivo ( . todos)
git commit -m "Ciando um arquivo."
```

Alteração de commit

```
git commit --amend (altera o último commit: mensagem de commit; adiciona arquivos)
```

Status

```
git status (para ver o status do repositório e alterações)
git log (ver os últimos commits no repositório)
```

Envio/recebimento de alterações

```
git push -u origin master (Enviar alterações, commits, de uma branch para o repositório remoto)
git push (mandar alterações para Branch na qual se encontra)
git pull (baixa as alterações do repositório remoto)
```

Verificar alterações

Verificar alterações

```
git diff HEAD~1 (verifica alterações ocorridas no último commit)

git checkout <commit> <file> (permite ver como um arquivo ou todo o repositório estava em um determinado commit {para voltar para a principal use git checkout master})
```

Descartes

```
git checkout -- <arquivo> <.> (para descartar mudanças antes de passar pelo git add, descartar um arquivo modificado ou todos os arquivos modificados)
git checkout HEAD -- <arquivo> <.> (para descartar mudanças depois de passar pelo git add, descartar um arquivo modificado ou todos os arquivos modificados)
```

Reset

```
git revert <commit> (irá criar um novo commit que desfaz as alterações do commit especificado)
git reset <commit> (resetar o repositório para um determinado commit)
git reset --hard <commit> (resetar e remover todas as alterações)
git reset HEAD~1 --hard (resetar e remover todas as alterações do último commit)
```

Branchs

```
git branch (mostra todas as branchs)
git branch <nova_branch> (cria uma nova branch)
git branch -d <branch> (exclui uma nova branch)
git checkout <branch> (muda para a branch)
```

Mesclagem

```
git merge <branch> (aplica os commits de uma branch na branch atual)
git fetch (baixa as atualizações do remote porém não aplica elas no repositório)
```

TAG, útil para definir versões estáveis do projeto. Semelhante a branch porém não recebe mais commits, guarda um estado do repositório

```
git tag [nome da tag]
git push <remote/origin> <tag>
```

FORK

- Copia um repositório de outro usuário para seu usuário no GitHub. É assim que começa a contribuição para outros projetos. Você teria uma cópia independente do repositório original, podendo fazer quaisquer alterações.

PULL REQUEST

- Grande simbolo da colaboração. É quando você solicita que suas alterações sejam unidas a uma branch no mesmo repositório ou a um repositório que sofreu fork.

```
Checkout em Pull Request: git fetch origin pull/ID/head:BRANCH
```

Armazenamento

```
git stash (guarda as alterações do Working Directory. Permite fazer rebase, merge, trocar de branch sem a necessidade de fazer um commit)
git stash list
git stash pop
Aplica o último stash armazenado
```

Visualizar alterações

```
git cherry-pick <commit/branch>** (aplica as alterações de um commit na branch atual)
git blame** (mostra as alterações feitas em um arquivo por linha, o autor e o commit que foi feito naquela linha)
git show <commit>** (mostra a alteração realizada pelo commit escolhido)
```

.gitignore

```
Configura arquivos que devem ser ignorados
.project (o arquivo)
node_module/ (a pasta e tudo que estiver dentro dela)
```

Estado dos Arquivos

```
[untracked] ao se criar um arquivo
[modified] quando edita um arquivo que já está no repositório
[staged] quando faz add
[commited] quando se commita um arquivo
```

[Macro]

```
git status
git add
git commit -m "Ciando um arquivo."
git push
```