Git

Nomenclatura de versões

- alfa = possui funcionalidades mas falta recursos.
- beta = todas funcionalidades, mas pode ter erro, teste pelos usuários.
- release candidate = todas funcionalidades, falta últimos testes do desenvolvedor.
- release = versão final.

- major.minor.patch - versao = 1.0.0 - Beta
- major = incremento com grande atualização
- minor = incremento com novas funcionalidades
- patch = incremento conforme correção de bugs

Configuração inicial para todos os Gits

```
git config --global user.email "seuEmail"
git config --global user.name <nickName>
```

Vendo configuração local

```
git config --list
```

Alguns dos principais comandos do Git

Verificando mudanças no projeto

Vendo status dos arquivos

```
git status
```

verificando o log

```
git log

para sair do log apertar Ctrl c ou Ctrl q
```

log resumido

```
git shorting
```

reflog mais completo que o log

```
git reflog
```

Vendo diferenças entre área de trabalho e master

```
git diff
```

Vendo diferenças entre branch

```
git diff <commit> <commit>

ver diferença para ultimo commit
git diff HEAD~1

ver diferença para n commit
git diff HEAD~n
```

Vendo diferenças entre arquivos

```
git diff <arquivoA> <arquivoB>
```

Mostrar alterações por linha para cada arquivo

```
git blame <nomeArquivo>
```

Commits

Se salva alterações no projeto através dos commits

Adicionar arquivo para commit

```
git add <nomeArquivo.extensao> ou *(vai adicionar todos arquivos)
```

Retirar arquivo da zona de preparação

```
git restore --staged <nomeArquivo.extensao>
```

Retirar arquivo

```
git reset <nomeArquivo.extensao>
```

Submeter o commit

Arquivos específicos, sem e com msg

```
git commit <nomeArquivo.extensao>

git commit <nomeArquivo.extensao> -m "descrição do commit sem caracter especial"
```

Vários arquivos, sem e com msg

```
git commit -a
git commit -a -m "descrição do commit sem caracter especial"
```

Enviando uma mensagem junto

```
git commit -m "descrição do commit sem caracter especial"
```

Desfazendo alterações

Voltar arquivo para estado anterior

```
git checkout <nomeArquivo.extensao>
```

navegar no histórico

```
git checkout <commit>
git checkout <commit> <file>
```

Desfazer o commit

```
git revert <commit>
```

Alterar comentario do último commit

```
git commit --amend "descrição do commit sem caracter especial"
```

Renomeando ou movendo arquivo

```
git mv <nomeArquivo.Extensao> <novoNomeArquivo.Extensao>
ou
git mv <nomeArquivo.Extensao> <novoCaminho/nomeArquivo.Extensao>
ou
git mv <nomeArquivo.Extensao> <novoCaminho/NovoNomeArquivo.Extensao>
```

Limpando arquivos untracked

```
git clean

ou para forçar
git clean -f
```

Ignorando arquivos no projeto

criar arquivo .gitignore na raiz do projeto e dentro colocar o nome do que precisa ser ignorado.

Repositórios

Criando um repositório

criar repositório local na pasta que esta no cmd.

```
git init
```

Enviando pela primeira vez para repositório do GitHub

Já com repositório local criado

```
git branch -m main (criando a master)
git remote add <nomeEscolhidoPorMim> <endereço do repositorio>
git -u push <nomeEscolhidoAcima> <branch> (nome do ramo de desenvolvimento)

modo mais usado abaixo:
git branch -m main (criando a master)
git remote origin <endereço do repositorio>
git -u push origin main
```

Enviando para repositório do GitHub

```
git push <apelido> <branch>

git push <branch>
```

Na Main

```
git push
```

Buscando arquivos do repositório do GitHub

```
git pull <apelido> <branch>
ou
git pull <branch>
```

Na Main

```
git pull
```

Removendo a origem do GitHub

listando as origens

Removendo a origem

```
git remove -v
git remote rm <apelido>
ou
git remove -v
git remote rm origin
```

Colocando novo endereço do repositóio remoto

```
git branch -m main (criando a master)
git remote add <nomeEscolhidoPorMim> <endereço do repositorio>
git -u push <nomeEscolhidoAcima> <branch> (nome do ramo de desenvolvimento)

modo mais usado abaixo:
git remote origin <endereço do repositorio>
git -u push origin main
```

Clonando Repositório

```
git clone <endereçoPegoNoGitHub> "nome do repositorio opcional"
ou
git clone <endereçoPegoNoGitHub>
ou clonar no diretorio atual
git clone <endereçoPegoNoGitHub> .
```

Removendo arquivos do repositório

```
git rm <nomeArquivo.Extensao>
git commit -a -m "Deletando arquivo desnecessário"
git push
```

Desfazendo Alterações

desfazendo todas alterações

```
git reset --hard <branch>
```

resetar o repositório para um determinado commit

```
git reset <commit>
git reset HEAD~n (onde n é o numero de commits para voltar)
```

```
git reset --soft HEAD~1 (volta estado de preparação)
git reset --hard HEAD~1 (volta sem a preparação)
```

Otimizando o repositório

```
git gc
```

Checando integridade de arquivos

```
git fsck
```

Transformando o repositório para arquivo

```
git archive --format zip --output master_files.zip
```

Submódulos

É a maneira que se tem de possuir dois ou mais projetos em um só repositório.

Verificando

```
git submodule
```

Adicionando submódulo

```
git submodule add <linkRepositorio> <nomeSubmodulo>
```

Atualizando submódulos

Para atualizar um submódulo, primeiro devemos commitar as mudanças.

```
git push --recurse-submodules=on-demand
```

Branchs

- Main (antigamente Master) = principal, onde fica versão estável.
- Hotfix = onde se conserta um erro da master.
- Release = onde se testa e vê se tem bug ou falta uma funcionalidade.
- Develop = onde junta as novas funções.
- Feature = nova função de cada usuário.

Detalhes da branch

```
git show
```

Visualizando branchs

```
git branch
```

Criando branchs

```
git branch <nomeBranch>
```

Excluir branch

```
git branch -d <nomeBranch>
ou
git branch --delete <nomeBranch>
```

Mudar branch

```
git checkout <nomeBranch>
```

Mudar e já criar nova branch

```
git checkout -b <nomeNovaBranch>
```

Juntar duas branchs

Uni a branch atual com a escolhida.

1 opção mantem os commits originais.

2 opção uni todos os commits em um.

```
git merge <nomeDaBranchQueSeraUnida>
ou
git rebase <nomeDaBranchQueSeraUnida>
```

Encontrando banchs remotas

```
git fech
```

Stash

Podemos salvar as modificações atuais para prosseguir com uma outra abordagem de solução e não perder o código atual.

Mostrar stachs

```
git stach list
```

Salvar temporário sem commitar.

```
git stach
```

Recuperando a stach

n é o número que identifica a stach

```
git stach apply n
```

Recuperando a última stach

```
git stach pop
```

Mostrando o que foi alterado no arquivo

n é o número que identifica a stach

```
git stach show -p n
```

Removendo todas stach

```
git stach clear
```

Removendo stach específica

n é o número que identifica a stach

```
git stach drop n
```

Tags

Precisa commitar para salvar alterações.

Listar tags

```
git tag
```

Verificando uma tag

```
git show <nomeTag>
```

Criando versão

```
git tag -a <nomeTag> -m "descrição da versão"
normalmente usada como abaixo, com versões.
git tag -a v1.0.0 -m "descrição da versão"
```

Alterando entre tags

```
git checkout <nomeTag>
```

Enviando e compartilhando tags

Uma tag

```
git push origin <nomeTag>
ou
git push <apelido> <nomeTag>
```

Todas as tags

```
git push origin --tags
ou
git push <apelido> --tags
```