Configurando

Configurando o git

> use para configurar o git na sua maquina

```
git config --global user.name "Seu nome"
git config --global user.email "seu@email"
```

Desconectar do git

> use para remover suas credenciais do git

```
git config --global --unset user.name
git config --global --unset user.email
ou
git config --global --unset-all
```

Criando um repositório local

> Use pra criar inicializar um repositorio

```
git init
```

- Verificando os arquivos dentro de .git

```
tree .git
```

Para isso e necessário que a ferramenta tree esteja instalada

---

Ignorar arquivos, que não serão commitados

- Crie um arquivo com o nome .**gitignore**
- coloque os arquivos ou pasta, um por linha

Conectando o github ao pc

1 - Criar um par de chave ssh com **ssh-keygen** não prencha nada

2 - Entre na pasta onde a chave foi criada abra o arquivo **id_rsa.pub** copie a chave

3 - va ate o github em ssh-key e insira a chave e um nome para ela

---

Trabalhando com arquivos

Vendo o status dos arquivos

```
git status
```

Adicionado na stage área

```
git add .  //adiciona tudo
git add *.txt //Adiciona todos os arquivos txt
git add <nome_do_arquivo> //adiciona apenas o arquivo
```

Commitando arquivos

```
git commit -m <Mensagem do commit>
```

Ver Alterações e Histórico

Ver o que foi alterando antes de adicionar a stage area

```
git diff
```

Depois que esta na stage area

```
git diff --staged
```

Ver o histórico do commit

```
git log
git log --oneline //mostra commit em uma só linha, facilitando a visualização

git log --oneline  --decorate  //mostra commit em uma só linha, mostrando o branch e o head do checkout atual

git log --oneline  --decorate  --all //mostra commit em uma só linha, mostrando o branch e o head de todos

git log --oneline  --decorate  --all   --graph //mostra commit em uma só linha, mostrando as branchs estilo gráfico, mostrando a ramificação

git log -p  //ver histórico do commit com detalhes do que foi alterado
```

Ver histórico pela interface

```
gitk
```

Reset de arquivo local

Remove alterações do arquivo

```
git restore <nome arquivo>
```

Limpa arquivo modified para situação antes da alteração local:

```
git checkout <diretório ou arquivo>
```

Retirar arquivo do staging:

```
git reset HEAD <diretório ou arquivo>
```

Desfazer commit local, mantém alterações e arquivos ficam como staged:

```
git reset --soft <hash do commit anterior>
```

Desfazer commit local, mantém alterações e arquivos ficam como modified:

```
git reset --mixed
```

Desfaz o commit local e as alterações realizadas nos arquivos:

```
git reset --hard
```

Criando Ramificações

Verificar as branchs

```
git branch
```

Criando nova branch

```
git branch <Nome_da_nova_branch>

git checkout -b <nome_da_nova_branch>  //cria um branch a partir da branch que você esta
```

Mudando da branch

```
git checkout <nome_da_branch>
```

Deletando uma branch

```
git branch -d <nome_da_branch>
```

Para apagar o branch remotamente:

```
git push <nome do origin> <nome do branch> --delete
ou
git branch -dr origin/nome-do-branch       -> -dr  : d- de delete r- de remote
```

Pega um commit expecifico e aplica numa branch

git cherry-pick

Merge

```
git merge <nome_da_branch_que_voce_deseja_os_dados>
```

No Merge, os commits do outro branch são aplicados por cima dos commits do branch atual.

Conflitos, caso esteja usando o ambiente da amil use:

```
git mergetool -t meld
```

Rebase

```
git rebase <nome_da_branch_que_voce_deseja_os_dados>
```

No Rebase, os seus commits (acima da base) são temporariamente apagados, o branch atual fica exatamente igual ao outro branch e seus commits são aplicados um a um no branch atual.

---

stash -> esconderijo

Salvar arquivos no stash

```
git stash save "texto indentificador"
```

Listar arquivos do stash

```
git stash list
```

Recuperando arquivo stash

```
git stash apply stash@{<referencia_do_arquivo>}  //recupera a referencia do
arquivo porem não a apaga do stash

git stash pop   //recupera e arquivo e apaga do stash
```

Apagar do stash

o comando anterior recupera arquivos do stash porem não apaga eles, caso for necessário use

```
git stash drop stash@{<referencia_do _arquivo>}
```

Criar uma branch a partir de um stash

```
git stash branch <nome_da_nova_branch>
```

Repositório remotos

Clonando repositório remoto

```
git clone <endereço_do_repositorio_remoto> [nome da pasta caso queira, e opcional]
```

Adicionando um repositório remoto a um repositório local

```
git remote -v  //Lista os repositório remotos

git remote add <nome_do_repositorio_remoto><endereço_do_repositorio_remoto>
/Adiciona um novo repositório


ex:  git remote add origin https://github.com/Uniliva/typescript.git
```