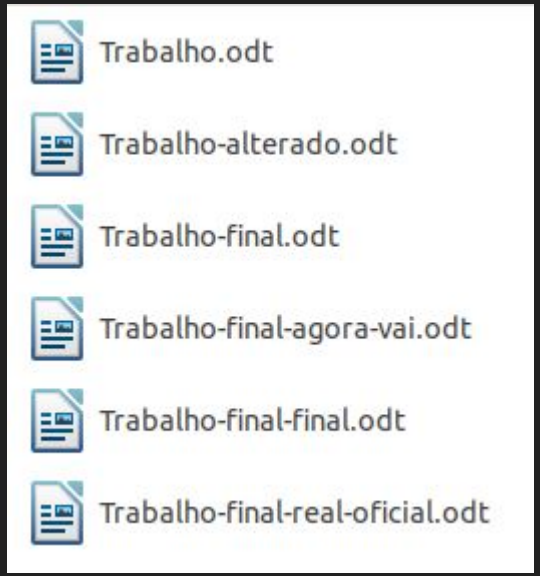
Trabalho em equipe com

Git

IPDEC

Versões de um arquivo



Nova Feature ?

Sem problemas...

Trabalho em equipe é tranquilo

Pessoas diferentes, pensamentos diferentes.

Vamos falar de Git

O que é o Git?

É um sistema do controle de versão.

Commit

Um commit é a confirmação de que suas alterações devem ser armazenadas no git.

Branch

Um branch é uma ramificação do seu repositório, assim você pode realizar alterações sem interferir no estado em que o repositório se encontra.

HEAD

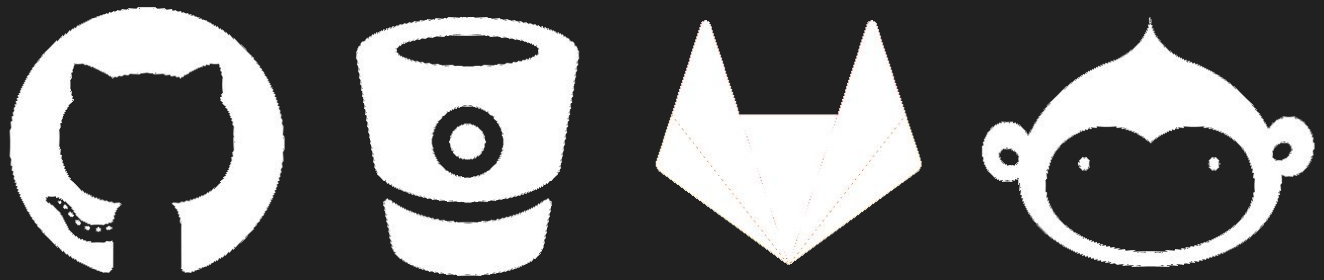
É um ponteiro especial do Git que aponta para a branch local em que você está no momento.

Ignore

O arquivo .gitignore contém os arquivos que o git não deve monitorar.

Remoto

Repositórios remotos são versões do seu projeto que estão hospedados na internet.



Iniciando um novo repositório:

```
git init
```

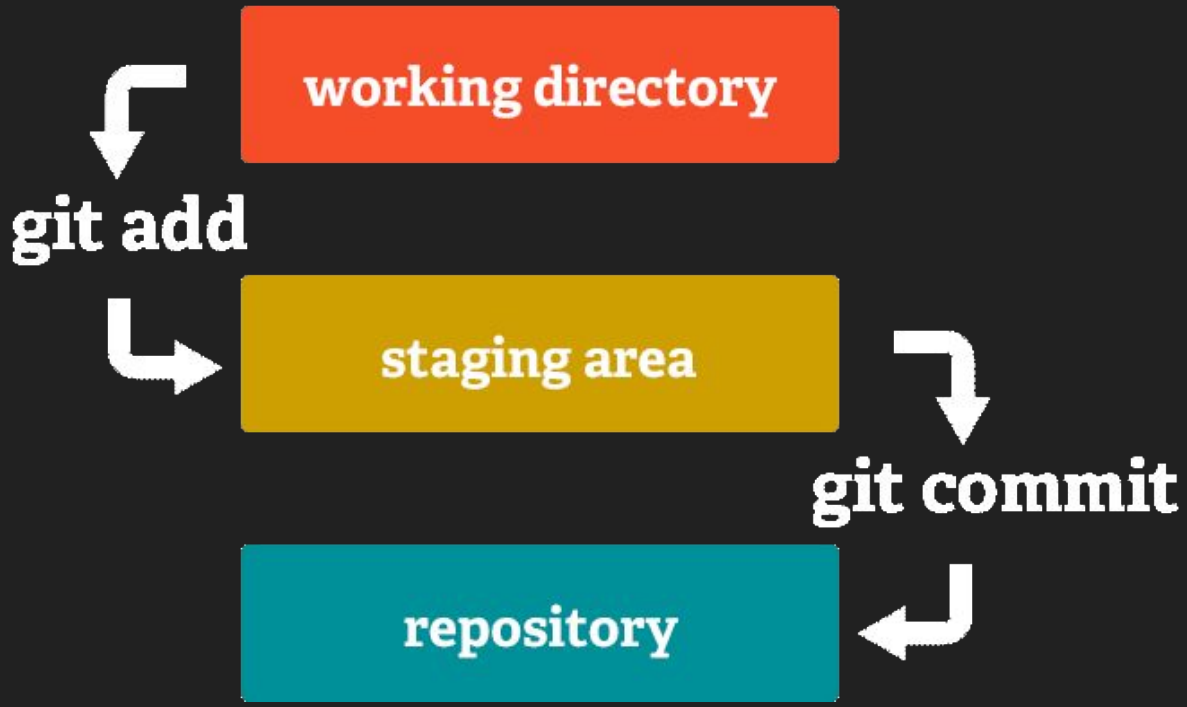
Onde é armazenado?

diretório .git

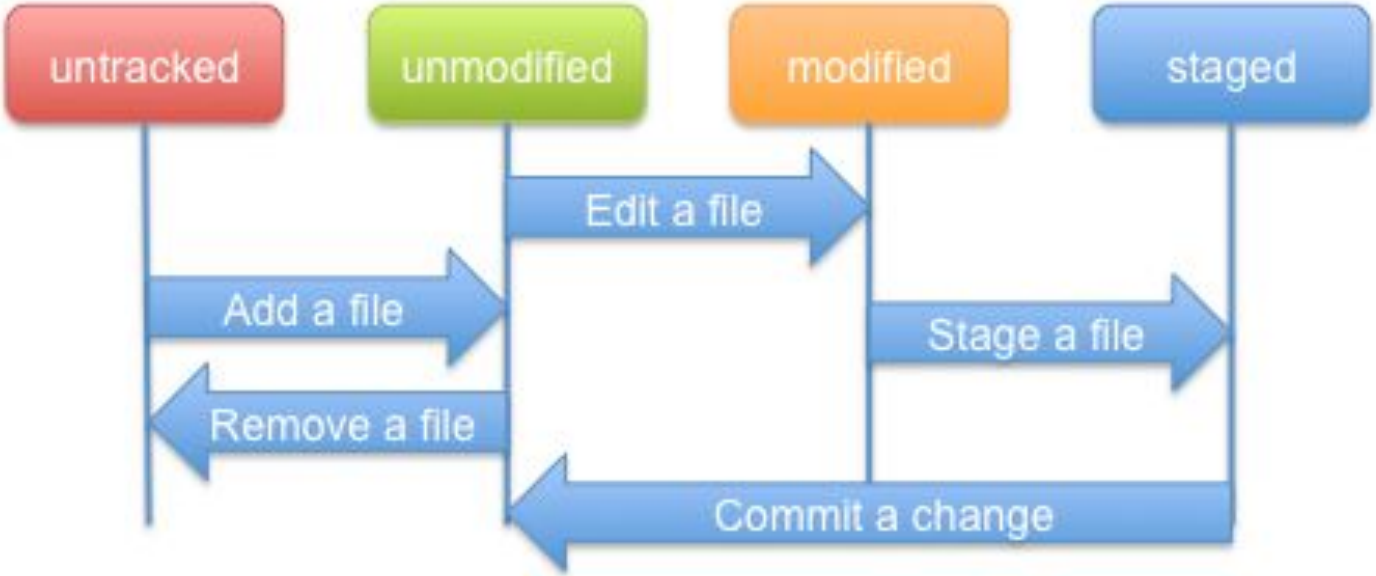
Vendo o status do repositório:

```
git status
```

Seus arquivos podem estar em três áreas:



Estados de um arquivo:



Adiciona um arquivo na staging area:

```
git add <nome_arquivo>
```

Adiciona todos arquivos modificados na staging area:

```
git add . ou -A ou --all
```

Removendo um arquivo do monitoramento:

```
git rm --cached <nome_arquivo>
```

Removendo todos arquivos do monitoramento:

```
git rm -r --cached .
```

Commit:

```
git commit -m “sua_msg”
```

Substituindo último Commit:

git commit --amend -m “sua_msg”

O --amend substitui seu último commit e sua msg por este.

Ver log e histórico de commits:

```
git log
```

Ver um commit:

git show

Ver alterações realizadas:

```
git diff
```

Removendo um arquivo na staging area:

```
git reset HEAD <nome_arquivo>
```

Removendo todos arquivos da staging area:

```
git reset HEAD .
```

descartando alterações de um arquivo:

```
git checkout <nome_arquivo>
```

Use . para descartar todas as alterações da working directory

alterando de branch:

```
git checkout <nome_branch>
```

Cuidado para não perder suas modificações ao trocar de branch

Resetando commit:

git reset --soft HEAD~

O --soft reverte o commit e deixa as alterações dele na staging area. HEAD~ diz para reverter o último commit.
Use no lugar de HEAD~ a hash do commit para retornar para um commit específico

Resetando commit:

git reset --mixed HEAD~

O --mixed reverte o commit e deixa as alterações dele na working directory.
HEAD~ diz para reverter o último commit.
Use no lugar de HEAD~ a hash do commit para retornar para um commit específico

Resetando commit:

git reset --hard HEAD~

O --hard reverte o commit e reverte os arquivos para o estado do commit anterior.
HEAD~ diz para reverter o último commit.
Use no lugar de HEAD~ a hash do commit para retornar para um commit específico

Revertendo commit:

git revert HEAD~

O revert cria um novo commit com a reversão das alterações realizadas. Desta forma o histórico é mantido.
Use no lugar de HEAD~ a hash do commit para reverter até um commit específico

Clonando um repositório do remoto:

```
git clone <url_remoto>
```

Configurando credenciais:

```
git config --global user.name “seu_usr”
```

```
git config --global user.email “seu_email”
```

Adicionando e removendo endereço do remoto:

```
git remote add origin <url_remoto>

git remote rm origin
```

Atualizando seu local com o remoto:

```
git pull origin <nome_branch>
```

Atualizando referências e branchs do local:

```
git fetch origin
```

Enviando dados para o remoto:

```
git push origin <nome_branch>
```

Criando e entrando numa branch:

```
git branch <nome_branch>
git checkout <nome_branch>
```

ou

```
git checkout -b <nome_branch>
```

Apagando uma branch:

```
git branch -D <nome_branch>
```

Salvando alterações inacabadas:

```
git stash save <nome_stash>
```

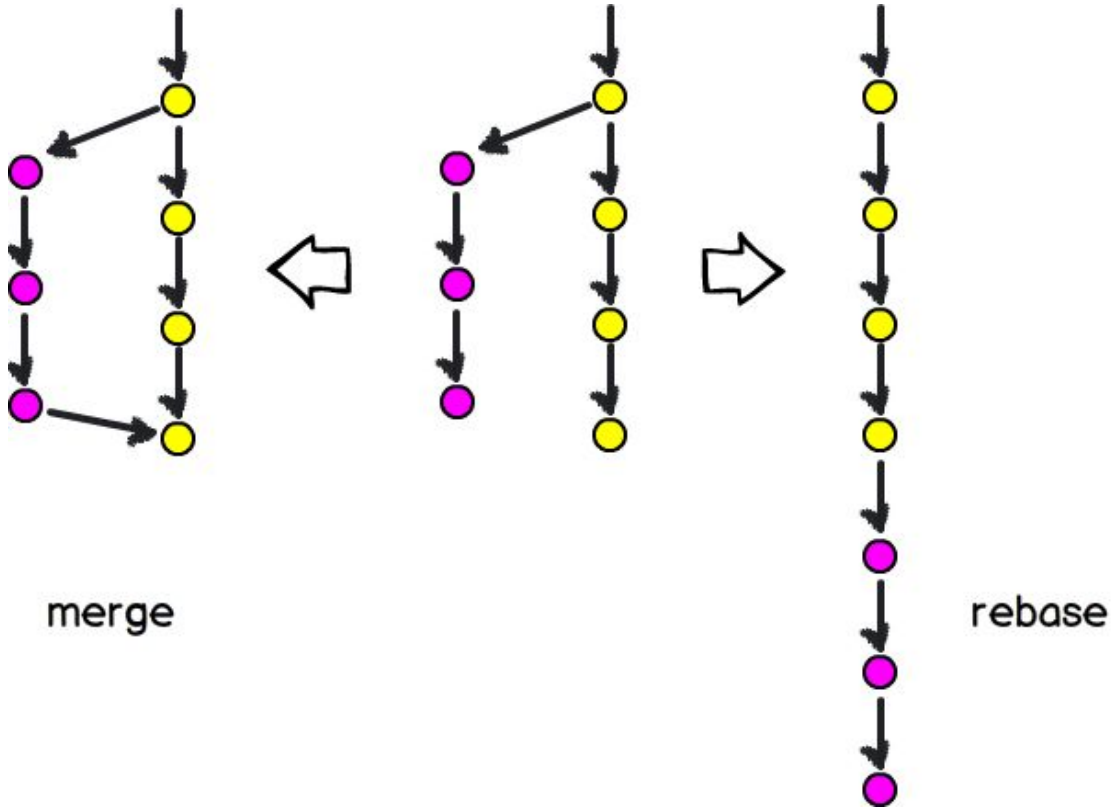
Listando stashs:

```
git stash list
```

Aplicando um stash:

```
git stash apply stash{<nro_stash>}
```

Unindo Branchs - Rebase e Merge:

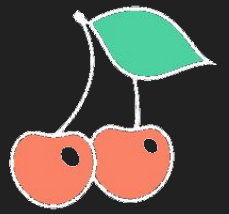


git merge <nome_branch>

git rebase <nome_branch>

git rebase --continue

Cherry-pick



git cherry-pick <hash_commit>

Criando alias para diminuir comandos

git config --global alias.<nome_alias> <comando>

ex:

git config --global alias.c checkout

agora o comando git c representa o comando git checkout

Versões e releases

git tag <tag_nome>

Uma tag representa uma release ou versão do seu projeto.

git submodule

Submódulo é o “import” de outro repositório dentro do seu projeto.

E as equipes ?

Problemas comuns

Noob

Falta de comunicação

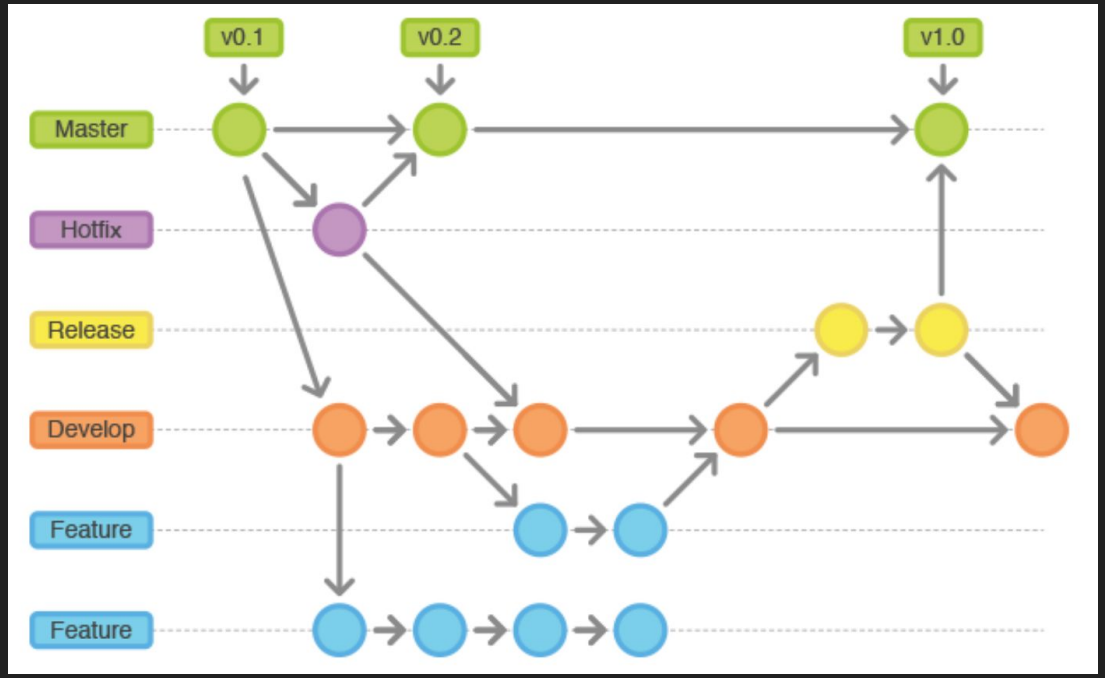
Conflitos

Perda de código

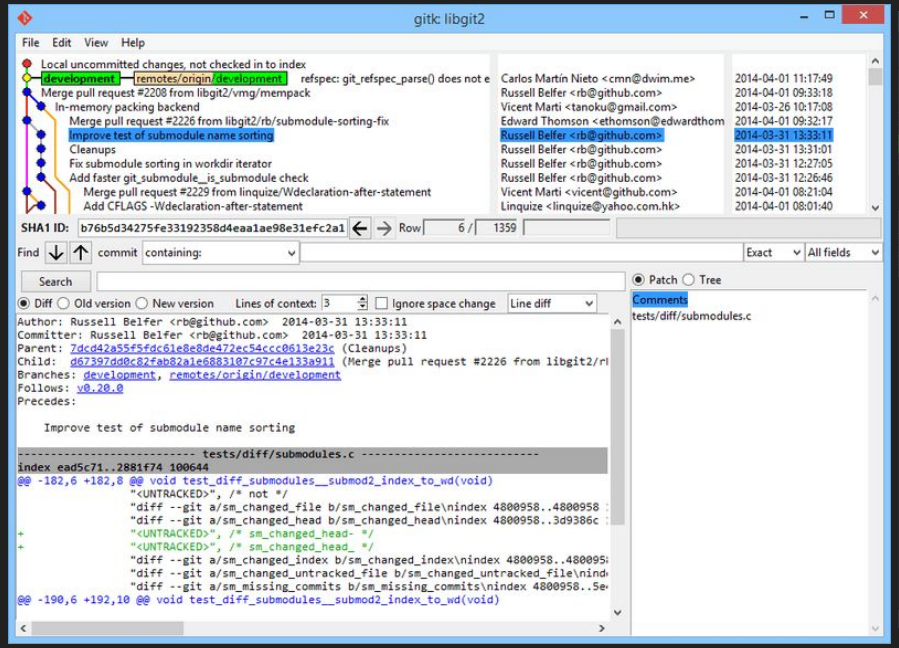
Bisonhice

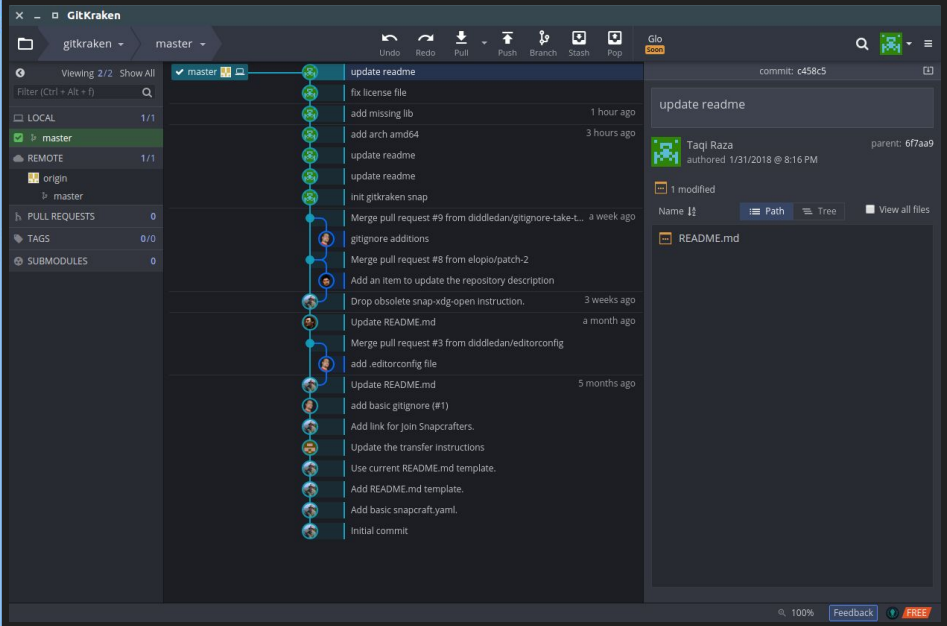
Problemas entre a equipe

Gitflow



GitK e GitKraken - ferramentas visuais





Vamos praticar

Obrigado !!!!

Leon Manickchand Junior
leon.junior@ipdec.org

https://gitlab.com/manickchand/curso-git