## Ramos — Branchs

Para equipes grandes é comum que os desenvolvedores estejam trabalhando em diferentes partes do sistema ao mesmo tempo. Imagine que o sistema principal é como uma arvore, e os desenvolvedores trabalham em pequenas modificações (galhos de arvore) que são inseridas na arvore principal quando estão prontos. Assim, desenvolvedores podem trabalhar paralelamente e entregar com qualidade.

considerar que o branch principal, ou seja a arvore principal, é master. Além disso, o HEAD é uma referência para o branch atual, ou seja, onde o código está alocado no momento. Por padrão, o HEAD aponta para o branch principal, o master.

**Branch — Comandos principais**

* git branch novo-branch - Cria o branch novo-branch.
* git checkout -b novo-branch - Criar o branch e mudar o código para o branch novo-branch .
* git checkout novo-branch - Muda o código para o branch novo-branch.
* git checkout master - Muda o código para a arvore. Branch principal: master / main
* git branch -d novo-branch - Deleta o branch novo-branch .
* git branch - Lista os branchs criados .
* git branch -v - Listar os branches criados com os logs de commit.
* git push origin novo-branch - Criando um branch remoto com o mesmo nome .
* git push origin novo-branch:new-branch - Criando um branch remoto com nome diferente .
* git checkout -b novo-branch origin/novo-branch - Baixar um branch remoto para edição.
* git merge novo-branch - Realiza o merge entre os branchs, ou seja, junta o ramo na arvore principal

.**gitignore**

O arquivo com nome .gitignore desabilita o envio de outros arquivos para o repositório .git. Em alguns projetos existem arquivos muito importantes, que possuem configurações específicas ou autenticações, esses arquivos devem ser ignorados para a demonstração no repositório. Além disso, arquivos de compilação e biblioteca devem ser inuteis no git, somente enxendo o repositório de informações desnecessárias.

**Adicionar arquivo/diretório**

No git, primeiro você adiciona os arquivos para depois enviar as modificações para o programa remoto. As seguintes variações de adição são interessantes:

* git add . - Adiciona todos os arquivos/diretórios modificados.
* git add meu\_programa.py - Adiciona somente o arquivo meu\_programa.py.
* git add meu\_diretório - Adiciona somente o diretorio meu\_diretorio.
* git add -f meu\_programa\_gitignore.py - Adiciona um arquivo que está no .gitignore.

Remover arquivo/diretório

* git rm meu\_arquivo.txt - Remover arquivo.
* git rm -r diretorio - Remover diretório.

Comitar arquivo/diretório

* git commit meu\_programa.py - Comitar um arquivo.
* git commit meu\_arquivo.txt meu\_outro\_arquivo.txt - Comitar vários arquivos.
* git commit meuarquivo.txt -m “minha mensagem de commit” - Comitar informando mensagem.

Verificando modificações/estado

* gitk - modificações totais do projeto.
* git diff - modificações antes de enviar as modificações para o repositório remoto (commit).
* git status - estado dos arquivos/diretório.

**Histórico**

* git log - Exibe histórico dos ultimos commits.
* git log -p -3 - Exibe histórico com diff dos ultimos 3 commits.
* git log - - <caminho\_do\_arquivo> - Exibir histório do arquivo no caminho\_do\_arquivo.
* git log - - pretty=oneline - Exibe histórico de commits com informações resumidas em uma linha. É possível realizar a exibição com formatação específica, **verifique aqui.**
* git log - - diff-filter=M - -<caminho\_do\_arquivo> - Exibe histórico de modificações de um arquivo.
* git log - -author=usuario - Exibir histório de um determinado autor

**Reescrevendo o histórico**

* git commit -- amend -m “Minha nova mensagem” - Alterando mensagens de commit já realizado.
* git rebase -i HEAD~3 - Alterar últimos commits, modificando as mensagens.

**Stash**

O Stash é um armazenamento temporário para modificações que ainda não podem ser inseridas no commit. Se você precisar mudar o branch, e já tiver feito as modificações, você realiza o stash para salvar temporariamente as modificações e inserir no branch desejado.

* git stash - Cria um stash, salva temporariamente as modificações .
* git stash list - Lista os stashes criados .
* git stash apply - Volta ao último stash .
* git stash apply stash@{2} - Volta ao stash com índice 2 .
* git stash branch meu\_branch - Criar um branch a partir de um stash.