TRABALHO SOBRE GIT

RODRIGO MAIA SANTOS

21/06/2014

Sumário

[CONCEITOS SIMPLIFICADOS DE PALAVRAS UTILIZADAS NO GIT 3](#_Toc391141977)

[LINK PARA PRATICAR O BÁSICO DO GIT: 3](#_Toc391141978)

[COMANDOS DO GIT 3](#_Toc391141979)

[INICIAR UM REPOSITÓRIO 3](#_Toc391141980)

[VER STATUS 3](#_Toc391141981)

[CLONANDO UM REPOSITÓRIO 3](#_Toc391141982)

[CONFIGURAÇÕES PÓS INSTALAÇÃO 4](#_Toc391141983)

[HISTÓRICO 4](#_Toc391141984)

[ADICIONANDO SNAPSHOTS 4](#_Toc391141985)

[CONFIRMAÇÃO 4](#_Toc391141986)

[NAVEGAR NO LOG 4](#_Toc391141987)

[CHECKOUT 4](#_Toc391141988)

[REVERT 5](#_Toc391141989)

[RESET 5](#_Toc391141990)

[LIMPANDO ARQUIVOS NÃO RASTREADOS 5](#_Toc391141991)

[BRANCHES 5](#_Toc391141992)

[CHECKOUT E BRANCH 6](#_Toc391141993)

[STASH 6](#_Toc391141994)

[MERGE 6](#_Toc391141995)

[REMOTE 6](#_Toc391141996)

[FETCH 7](#_Toc391141997)

[PULL 7](#_Toc391141998)

[PUSH 7](#_Toc391141999)

[ADICIONANDO EM REPOSITÓRIO REMOTO: 7](#_Toc391142000)

[PROCURAR POR DIFERENÇAS 7](#_Toc391142001)

[CHECKOUT 7](#_Toc391142002)

[REMOVENDO ARQUIVOS 7](#_Toc391142003)

[BIBLIOGRAFIA 8](#_Toc391142004)

## CONCEITOS SIMPLIFICADOS DE PALAVRAS UTILIZADAS NO GIT

**HEAD**: Última versão.

**BRANCH**: Cópia do projeto.

**COMMIT**: envio de alterações

**LOG**: registro de atividades

**DIFF**: Diferença entre uma versão e outra

**ROLLBACK**: Pontos de retorno para determinada versão

**WORKING** **DIRECTORY**: Onde vai trabalhar com os arquivos

**STAGE** **AREA**: Onde guarda as alterações, aguardando confirmação.

**GIT** **DIRECTORY**: Onde guarda o projeto

## LINK PARA PRATICAR O BÁSICO DO GIT:

<https://try.github.io/levels/1/challenges/1>

## COMANDOS DO GIT

### INICIAR UM REPOSITÓRIO

Quando estiver no diretório repositório

git init

git init –bare (usado no servidor)

### VER STATUS

git status

### CLONANDO UM REPOSITÓRIO

git clone *e-mail:login/novorepositório*.git

### CONFIGURAÇÕES PÓS INSTALAÇÃO

git config --global user.name “*Rodrigo Maia*”

git config --global user.email *roderictus@gmail.com*

### HISTÓRICO

git log

--oneline (simplifica o log)

--graph

### ADICIONANDO SNAPSHOTS

git add *arquivo2.extensão*

(aguardando confirmação)

### CONFIRMAÇÃO

git commit –m “*comentário*”

### NAVEGAR NO LOG

git checkout *código do commit desejado*

git checkout master (vai para a versão mais recente)

### CHECKOUT

checkout serve para 3 tarefas:

* voltar commits,
* arquivos
* e branches

git checkout master

git checkout <commit ou HEAD> <file>

git checkout <commit ou HEAD>

HEAD é aponta para a última alteração do arquivo

HEAD~1 (volta 1 commit, por exemplo)

O HEAD sempre aponta para o atual

E o master para o final

### REVERT

Desfaz um commit, mas não deleta, ele cria um novo commit posterior.

git revert <commit ou HEAD>

### RESET

Usar com cuidado, pois não tem retorno. É um revert permanente.

Jamais fazer de histórico público. Problemas em trabalho em equipe.

git reset <file> (apenas retira da stage área)

git reset (apenas retira tudo da stage área)

git reset –hard (retira da stage área e apaga as alterações nos arquivos)

git reset <commit> (concatena os commits num commit citado)

git reset –hard <commit> (descarta os commits posteriores ao informado)

### LIMPANDO ARQUIVOS NÃO RASTREADOS

git clean

git clean –n (simula sem remover)

git clean –f (remove arquivo não rastreados)

git clean –f <caminho> (remove arquivo não rastreados de lugar determinado)

git clean –fd (remore arquivos e diretórios)

git clean –xf (remove arquivos não rastreados e ignorados)

### BRANCHES

git branch (lista de brach e diz em que branch está)

git branch <nome> (dar nome)

git branch –d <nome> (deleta nome)

git branch –d <branch> (deleta branch)

git branch –m <branch> (renomeia branch)

### CHECKOUT E BRANCH

git checkout –b <branch> (permite criar um novo branch e trocar por ele)

git checkout <branch> (mudar de um branch para outro)

git checkout –b <new-branch> <atual-branch> (cópia de branch)

### STASH

Arquivo que devem ser commitadas... mas são “commitadas temporariamente”.

git stash (grava)

git stash pop (retorna)

### MERGE

Mesclagem entre branch

git merge <branch>

*A mesclagem não ocorre se entre os arquivos comuns tiverem alterações nas mesmas linha.*

### REMOTE

git remote (mostra apelidos)

git remote –v (lista todos os apelidos com as origens)

git remote add <name> <url> (apelido)

git remote rm <name> (remove apelido)

git remote rename <old-name> <new-name> (troca de apelido)

### FETCH

Importa os commit de repositórios.

Para ver antes de commitar

git fetch apelido branch

### PULL

Baixa as alterações mesclando no repositório

git pull apelido branch

### PUSH

Envia para repositórios remotos

Não funciona se no destino tiver alterações que não estiver no remetente.

### ADICIONANDO EM REPOSITÓRIO REMOTO:

git remote add origin https://github.com/try-git/try\_git.git

git push –u origin máster (o –u coloca os parâmetros seguintes na memória para não precisar repeti-los na próxima vez).

### PROCURAR POR DIFERENÇAS

git diff HEAD

git diff - - staged

### CHECKOUT

git checkout – ARQUIVO (volta para último commit do arquivo)

### REMOVENDO ARQUIVOS

git rm ‘\*.txt’ (exemplo)

## BIBLIOGRAFIA

http://www.4javacursos.com.br/blog/todo-programador-deve-usar-git

https://www.youtube.com/watch?v=4ROWC8upplM

https://www.youtube.com/watch?v=u9WEkx8cuGU

https://www.youtube.com/watch?v=SMWk\_Ccyuoo