Sumário

[**Configuração Inicial**](#_n1cyq5efpq1y) **1**

[**Básicos**](#_cpvtcry0fga0) **2**

[Apagando arquivos não rastreados](#_usqbhu5rqs9m) 3

[Visualização de Histórico](#_nt4tavsfu071) 3

[**Manipulação de Branches**](#_hgd2g1m9kd70) **3**

[Branch](#_nc8uwctdzuzv) 3

[Checkout](#_7h6d9jx65t9k) 4

[Merge](#_t2pfzo5l16hh) 4

[Ferramentas para resolução de conflitos](#_9gxnrw98pqob) 5

[KDiff3](#_gsqfg5nwrbwc) 5

[Windows](#_2je2ekecyy0f) 5

[Linux (Ubuntu e derivados)](#_z095koxnrvop) 5

[**Repositórios remotos**](#_9snaaue6rt95) **6**

[Push](#_ilrxmw7vro85) 6

[Sync](#_y69ec2qqx594) 6

[Rebase](#_4zvyozohmry8) 6

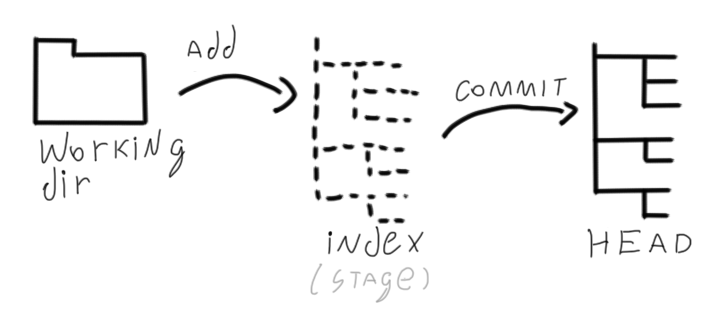
[**Tag**](#_fec1m56dksak) **6**

[**Stash**](#_9ysowi8idcks) **7**

[**Alias**](#_ho4m53tpfp8p) **7**

[**Reset**](#_hp2la0o8ef2h) **8**

Working Flow



Comandos Git

# **Configuração Inicial**

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git config --global user.name “<usuário>” | Configura o nome do usuário.  **<usuário>**  DanielCastriani |
| git config --global user.email “<email>” | Configura o e-mail.  **<email>**  dcastriani@gmail.com |
| git config --global user.name | Exibe nome do usuário previamente configurado. |
| git config --global user.email | Exibe e-mail previamente configurado. |
| git config --global --list | Exibe usuário e email. |
| git config --global --add merge.tool <ferramenta de merge> | Adiciona uma ferramenta de merge. |
| git config --global --add mergetool.<ferramenta>.path “<path>” | **<path>** caminho do executável .  **<ferramenta>** Ferramenta adicionada |
| git config color.ui true | Usar os comandos do git com coloração |
| git config format.pretty oneline | Mostrar o log em uma única linha por padrão |
| git config --global --unset <configuração> | Remove configuração  **<configuração>** user.name |

# **Básicos**

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git init | Inicializar um repositório local(cria o diretório .git com vários arquivos de configuração, .git está oculto). |
| git remote add <nome> <Link> | Adicionar um repositório remoto ao local  nome a ser utilizado **<nome>**  origin  Link do repositório **<link>**  https://github.com/DanielCastriani/8Puzzle.git |
| git clone <path> | Clona o repositório **<path>**  Clone de repositório **local**:  C:/pasta/repositório  Clone de repositório **remoto**:  https://github.com/DanielCastriani/8Puzzle.git |
| git status | Verificar status dos arquivos, sendo eles não rastreados, modificados, entre outros. |
| git add <arquivo> | Adiciona um arquivo/diretório ao rastreio do git  **<arquivo>**  .  *adiciona todos arquivos/diretórios*  --all  adiciona *todos arquivos/diretórios*  arquivo.txt  *adiciona um arquivo/diretório específico, neste caso arquivo.txt* |
| git rm -r --cached <arquivo> | remove um arquivo/diretório ao rastreio do git  **<arquivo>**  .  remove todos arquivos/diretórios  arquivo.txt  remove um arquivo/diretório específico, neste caso arquivo.txt |
| git commit -m “<mensagem>” | Realiza o commit (a confirmação) das alterações. |

## Apagando arquivos não rastreados

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git clean -f | Apaga todos os arquivos que não foram rastreados. |
| git clean -n | Exibe uma lista de todos os arquivos que não foram rastreados e serão apagados pelo comando **git clean -f** |
| git clean -i | Usuário seleciona arquivos não rastreados e que serão apagados. |

## Visualização de Histórico

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git log | Exibe histórico de commits exibindo **hash** (código do commit), autor, email, data, hora, mensagem, ... |
| git log --stat | Mesmo que **git log** com a quantidade de alterações realizadas. |
| git diff <hash1> <hash2> | Exibe as alterações realizadas entre dois commits. |
| git log --author=”Daniel” | Mostra todos os commits realizados por um determinado autor do repositório. |
| git log --oneline | Log em uma linha |

# M**anipulação de Branches**

## **Branch**

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git branch | Exibe todas as branches locais. |
| git branch -r | Exibe todas as branches remotas. |
| git branch -a | Exibe todas as branches, sendo elas remotas ou locais. |
| git branch <nome> | Cria uma branch a partir da branch atual.  Quando for criado uma branch localmente, é necessário enviá la para o servidor com o comando push. |
| git branch -d <nome> | Deleta a branch **<nome>** localmente, se a branch estiver a frente, é obrigado a colocar **-D** |
| git branch -m <Novo Nome> | Renomeia branch atual para **<Novo Nome>** |
| git branch -m <Nome> <Novo Nome> | Renomeia branch com no nome **<Nome>** para **<Novo Nome>** |

## **Checkout**

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git checkout <arquivo> | Retorna o arquivo ao estado HEAD. |
| git checkout <hash> | Vai para o commit relacionado a hash, não altera o estado HEAD. |
| git checkout <branch> | Se **<branch>** for a selecionada, retorna a branch para o estado HEAD, se não, seleciona outra branch. |
| git checkout -b <nome> | Criar uma branch com o nome **<nome>**, e realiza checkout.  Mesmo que git branch <nome> + git checkout **<nome>** |
| git checkout -b <branch> <tag> | Cria uma branch a partir de uma tag. |

## **Merge**

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git merge <branch> | Realiza a mesclagem do branch atual, e a branch **<branch>** trazendo todos os commits da branch mesclada. |
| git rebase <branch> | Realiza a mesclagem do branch atual e a branch **<branch>** trazendo apenas um commit de merge e ignorando os commits realizados na branch **<branch>** |
| git mergetool | Executa ferramenta para resolução de conflito configurada no *merge.tool* |

Quando ocorre conflito, o arquivo será editado

|  |
| --- |
| <<<<<<< HEAD  Conflito1  =======  Conflito 2  >>>>>>> branch |

Basta editar o arquivo removendo

|  |
| --- |
| <<<<<<< HEAD  =======  >>>>>>> branch |

Uma forma mais fácil de se resolver conflito é utilizando ferramentas de merge

## Ferramentas para resolução de conflitos

### KDiff3

Para fazer a instalação da ferramenta, é necessário seguir os passos:

#### Windows

Acesse o link: [kdiff3.sourceforge.net](https://sourceforge.net/projects/kdiff3/files/) e faça o download e instale a ferramenta para poder utilizá-la.

#### Linux (Ubuntu e derivados)

Execute no terminal do seu Ubuntu o seguinte comando: **sudo apt install kdiff3**

É necessário configurar utilizando os comandos:

|  |
| --- |
| **git config --global --add merge.tool kdiff3**  **git config --global --add mergetool.kdiff3.path "<caminho\_para\_o\_executavel>"** |

No Linux não é necessário colocar o caminho completo para o executável, somente o nome da aplicação, no caso, kdiff3.

# 

# Repositórios remotos

## Push

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git push --set-upstream origin <branch> | Envia commit para repositório remoto, e seta o rastreio. |
| git push -u origin <branch> | Mesmo que git push --set-upstream origin <branch> |
| git push origin --delete <branch> | Remove branch **<branch>** do repositório remoto. |

## 

## Sync

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git remote update | Atualiza as referências do repositório remoto |
| git status -uno | Busca todos os commits da branch atual que não estão no repositório local |
| git pull | Baixa todos os commits da branch atual que não estão no repositório local |

# Tag

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git tag | Exibe a lista de tags já criadas |
| git tag <nome> | Cria uma tag **<nome>** localmente |
| git tag -d <nome> | Remove tag **<nome>** local |
| git push origin --tags | Envia tags criadas localmente para o servidor, e relaciona com o estado head |
| git push origin --delete <tag> | Remove tag so servidor |
| git tag <nome> <hash> | Envia tag para o servidor e relaciona a um commit específico |

# Stash

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git stash | Guarda as alterações realizadas em uma pilha, e retorna o branch ao estado HEAD, assim possibilitando realizar o **checkout** para outra branch |
| git stash apply | Retorna os arquivos alterados do topo da pilha.  Não retira da pilha |
| git stash apply stash@{<index>} | Retorna os arquivos alterados a posição <index>.  Não retira da pilha |
| git stash pop | Retorna os arquivos alterados do topo da pilha.  Retira da pilha |
| git drop | Retira da pilha.  Não altera os arquivos |
| git stash list | Exibe pilha |

# Alias

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git config --global alias.<novo comando> <comando> | Define um apelido para representar um comando já existente do git.  **git config --global alias.st status**  Para executar o comando status:  **git st** |
| git config --global --unset alias.<comando> | Remove comando do alias |
| git config --get-regexp alias | Exibe lista de alias |

# Reset

|  |  |
| --- | --- |
| **Comando** | **Descrição** |
| git reset --hard origin/master | Retorna todos arquivos rastreados e que foram alterados para os estado em que se encontra o branch origin/master. |