



## MANUAL DO INICIANTE GIT/GITHUB

### Comandos do banco de dados

### Comandos citados

### Nomenclatura criada para teste

#### Primeiro acesso

Para começa a usar o GIT é necessário definir um usuário e um E-mail, o E-mail deve ser igual a usado na conta GITHUB para unificação.

O comando a seguir defini o usuário GIT da máquina.

```
git config --global user.name "JD"
```

O comando a seguir definir o e-mail do usuário GIT da máquina.

```
git config --global user.email "JD@EXAMPLE.COM"
```

Validando informações das configurações pode ser usado o comando **GIT CONFIG** porém para que tenha dados especifico deve informar o parâmetro em seguida.

```
git config user.name
```

#### Navegação com GIT

Comando Windows para navegação entre pastas

DIR: é usado para ver o conteúdo da pasta acessada.

```
dir
```

CD + Nome da pasta: é usado para entrar na pasta.

```
cd PASTA
```

CD + .. : é usando para sai da pasta.

```
cd ..
```

MKDIR + nome: usado para criar uma pasta.

```
mkdir NOME_PASTA
```

#### Criando e acessando diretório

Diretório GIT é uma pasta onde todo seu conteúdo é monitorado referente a alterações.

Para definir um repositório é usado dentro da pasta desejada o comando **GIT INIT** será criado uma pasta oculta dentro das mesmas com nomenclatura (. GIT) e ao acessar a pasta que foi criado um repositório GIT será sinalizado no GITBASH como MASTER.

```
git init
```

#### Acessando arquivos

Para acessar no GIT pelo VIM utiliza comando **VIM nome desejado e sua extensão** caso o arquivo não exista será criado no local que foi executado será aberto a tela do vim no GIT onde usa comandos **I** para inserir dados **ESC** para sai do modo INSERT digita ":" mais **W** para salvar alterações no modo dois pontos **Q** para sai da edição podendo usar **WQ** junto.



## Fases do controle de versão

O comando **GIT STATUS** serve para verificar como está seus arquivos quando sofre uma alteração ou está na área para commitar dentro do seu diretório.

### GIT ADD

Quando os arquivos são novos ou sofreram alterações eles estão no status: “Untracked files - arquivos não rastreado” e no GITBASH é identificado com a cor vermelha.

Para adicionar o arquivo no próximo status, “Changes to be committed-staged”, é utilizado o comando **GIT ADD**.

Para comentar a arquivos únicos é utilizado comando git add mais o nome do arquivo.

```
git add index.php
```

Em casos onde deseja enviar apenas um formato de arquivo, exemplo PHP, pode ser usado o comando a seguir:

```
git add *.php
```

Caso deseje enviar todos os objetos utilizar o comando a seguir.

```
git add .
```

Caso deseje voltar o arquivo para a primeira área utiliza o comando **GIT RESET HEAD** e nome do arquivo com extensão ou ao invés do nome do arquivo utilizar comando **FULL**.

```
git reset head full
```

### GIT DIFF

Para consultar a diferença de arquivo na área não rastreada para área STAGE é utilizada o comando GIT DIFF.

Para consultar a diferença de arquivo na área STAGED para o último commit é utilizada o comando GIT DIFF --STAGED.

### MODIFY

Para reverter alterações feitas no arquivo que esteja na primeira área utiliza comando **GIT CHECKOUT** em seguida do nome do arquivo e sua extensão (para reverter apenas aquele arquivo).

### COMMIT

**GIT COMMIT** seguido do nome do arquivo com extensão (para especificar qual arquivo fazer commit) serve para salvar as alterações em versões, seguindo o mesmo exemplo do GIT ADD.

Para adicionar comentário no commit faz uso do **-m** e a mensagem entre aspas duplas

```
git commit FILE.SQL -m "V1.1"
```

O commit pode ser feito já na primeira etapa de alteração na “untracked files - arquivos não rastreado”, através do comando:

```
git commit FILE.SQL -a -m "V1.1"
```

Caso deseje alterar o último commit utilizar o comando **--AMEND**, com esse comando o commit a seguir irá alterar ou mesclar ao último commit.

```
git commit - -amend -m "nova mensagem".
```



## LOG

Para lista os commit feito no diretório é usado o comando **GIT LOG** onde vai trazer detalhes dos commit como comentários, autor, chave e etc.

```
git log
```

Caso deseje vê apenas as chaves e o comentário utiliza comando

```
git log --pretty:oneline
```

Para filtra apenas por commit de um autor utilizar comando

```
git log author="name"
```

O **GIT LOG -P** busca todos commit e detalhe do que foi alterado em ordem cronológica e decrescente podendo também definir quantos commit em você deseja buscar definido depois de traço a quantidade **GIT LOG -P -2**.

```
git log -P -2
```

O **GIT SHORTLOG** agrupa o log por autor ordenando em ordem alfabética e trazendo apenas informações do comentário em ordem crescente dentro da organização de cada autor caso deseje apenas a quantidade de commit por autor utilizar comando - -SN

```
git shortlog - -sn.
```

## DELETE MR

Para deletar um arquivo utiliza comando **GIT RM** em seguida do nome do arquivo e sua extensão (para especificar o arquivo que deseja deletar), quando deletado o arquivo fica na segunda área o (STAGED) para que seja feito o commit do delete podendo fazer o commit ou voltar para primeira área com o comando **GIT RESTORE - -STAGED** voltando para primeira área pode ser feito o restore com o comando **GIT RESTORE**.

Deletando commit

Git reset --soft --mixed --hard

## TAG

As TAGS servem como se fosse uma etiqueta que é colocada nos commit para se referência para criar uma TAG utiliza o comando **GIT TAG** junto com o nome da TAG desejada caso queira adicionar a informações de data hora e autor entre outras informações utiliza a **-A** antes do nome da TAG podendo também ser adicionado comentário após a criação da TAG com o comando **-M**.

```
git tag -a v1.0 -m "teste de versão"
```

Para criar uma TAG me um log anterior basta usar mesmo comando com a chave do log após o nome da TAG

```
git tag -a v1.0 4feddd6a5919fc770ab82ef3a612ac4d7b002019 -m "teste de versão"
```

Para verificar as TAGS disponíveis naquele diretório basta utilizar comando **GIT TAG** caso precise de detalhes sobre a TAG utiliza o comando **GIT SHOW** seguido do nome da TAG onde vai trazer todas informações da TAG e do log o qual está vinculado.

Para voltar um script para versão anterior através da TAG utiliza comando

```
git checkout nome_da_tag
```

Dessa forma o script volta a ser o mesmo do commit que a TAG está vinculada para voltar para última TAG criada só utilizar **GIT CHECKOUT MASTER**.

Para deletar uma TAG utiliza comando **-D** seguido do nome da TAG



```
git tag -d v1.0
```

## BRANCH

BRANCH são ramificações da linha de compilação para que seja feito alterações sem mexer com o código existente ele funciona paralelo a linha de compilação original.

Para criar um BRANCH utiliza o comando GIT BRANCH e o nome desejado logo após utiliza o comando GIT CHECKOUT e nome do BRANCH assim ele vai está acessando uma cópia de quando o BRANCH foi criado sem alterar o código da linha MASTER.

```
git branch teste  
git checkout teste
```

Para trazer as alterações que fora feito commit no BRANCH para linha principal MASTER utilizar comando MENGE assim as alterações que foram feitas serão copiadas para o código MASTER.

```
git munge teste
```

Para deletar um BRANCH utiliza o comando **-D** seguido do nome do **BRANCH**

```
git branch -d teste
```

rebase

## ACESSANDO DIRETÓRIO REMOTO

Para criar um repositório compartilhado no servidor local para que seja acesso por outras pessoas na rede utiliza o comando **GIT INIT - -BARE** será criado várias pastas dentro do repositório ao invés de apenas a pasta (. GIT) e ao acessar a pasta que foi criado um repositório será sinalizado no GIT como BARE: MASTER.

A pasta do repositório deve estar compartilhada.

Para acessar um repositório remoto em uma máquina cliente utiliza o comando

```
GIT CLONE FILE:/// HOST_NAME ou IP/NOME_DA_PASTA_COMPARTILHADA/NOME_DO_DIRETORIO NOME DIRETORIO DESEJADO
```

Este comando vai clonar tudo que existir no diretório que foi direcionado para máquina cliente com a nomenclatura do diretório de acordo com que foi descrito no final do código.

Por padrão a conexão remota entre as máquinas recebe o nome de ORIGEN.

Para atualizar os que foi feito em sua máquina para o servidor faz uso do comando **PUSH** definindo a conexão (ORIGEN) e o BRANCH que está saindo as alterações.

```
GIT PUSH ORIGEN MASTER
```

Para atualizar seu diretório com as informações existente no servidor faz uso do comando **PULL** definindo o nome da conexão e do BRANCH que vai ser atualizado assim o diretório local recebera as atualizações feita no servidor.

```
GIT PULL ORIGEN MASTER
```

Git fetch origin branch moneclatura

## GITHUB

GITHUB é um repositório remoto para conectar a ele é necessário criar uma chave de acesso dentro de sua máquina utilizando o comando

```
SSH-KEYGEN -T RSA -B 4096 -C "YOUR_EMAIL@EXAMPLE.COM"
```

Definindo E-mail de acordo com que foi configura no GIT será criado uma pasta dentro do usuário com nomenclatura de (. SSH) onde a chave será criada com nomenclatura de (ID\_RAS.PUB) onde dentro dela vai conter a chave que irá conectar com GITHUB.



Signed in as  
**JailtonDPaula13**

Set status

Your profile  
Your repositories  
Your projects  
Your stars  
Your gists

Help  
**Settings**  
Sign out



Personal settings

**Profile**  
Account  
Security  
Emails  
Notifications  
Billing  
**SSH and GPG keys**  
Blocked users  
Repositories  
Organizations  
Saved replies  
Applications

### SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 **DELL**  
  
SSH Added on 24 Aug 2019  
Last used within the last week — Read/write

[Delete](#)

Definindo um titulo e em KEY adicionando uma chave ele irá ter uma conexão com seu GIT local



SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

Existir duas maneira de conectar com um repositório no GITHUB uma delas é criar um repositório na maquina e transferindo pro GIT ou outra é criar um repositório no GITHUB ou pegar um já existente.

Para mandar um repositorio do GIT pro GITHUB primeiro tem que criar um repositorio dentro do GITHUB onde ira receber o repositório do GIT, ao criar repositorio deixa a opção (**INITIALIZE THIS REPOSITORY WITH A README**) desmarcada

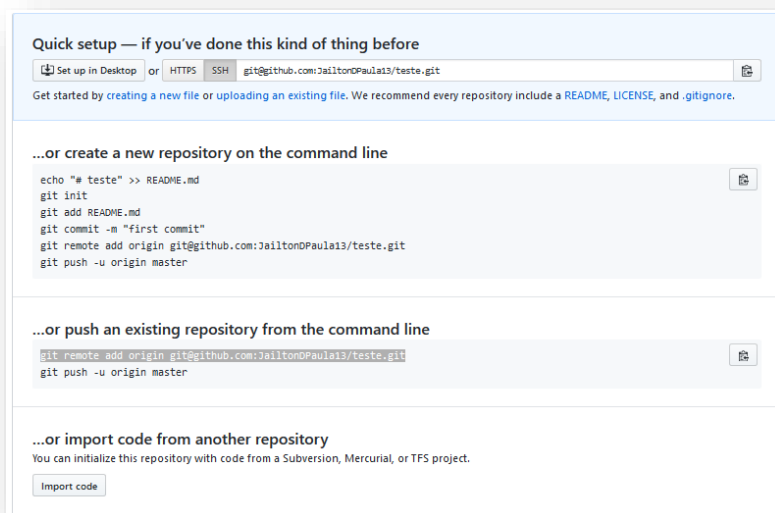
Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾ | Add a license: None ▾ ⓘ

Create repository

Será criado um repositório vazio com instruções para conectar



Para importar do repositório do GIT para o GITHUB é preciso utilizar o comando **GIT REMOTE** seguido do nome que deseja para conexão e o link do diretório.

**GIT REMOTE ADD TESTE GIT@GITHUB.COM:JAILTONDPAULA13/TESTE.GIT**

Para conectar um repositório do GITHUB para GIT utiliza o comando **GIT CLONE** seguido do link de conexão e nome que deseja para pasta no diretório do GIT.

**GIT CLONE GIT@GITHUB.COM:JAILTONDPAULA13/TESTE.GIT PROJETOS**

Para fazer DOWNLOAD e UPLOAD do diretório no GITHUB funciona da mesma forma que acesso remoto utilizado comando **PUSH** para upload ou **PULL** para download seguido do nome da conexão com o nome do **BRANCH**.

**GIT .IGNORE**