



Quem se prepara, não para.

Arquitetura de Aplicações Web

5º período

Professora: Michelle Hanne

Sumário

Git

Git Bash – Instalação

Git Bash – Principais comandos

O Git é um sistema de controle de versão gratuito e de código aberto criado por Linus Torvalds em 2005. Diferente de outros sistemas de controle de versão centralizados, como SVN e CVS, **o Git é distribuído: todo desenvolvedor tem o histórico completo de seu repositório de códigos local.**

CVS e SVN

CVS (Concurrent Versions System): É um sistema de controle de versões OpenSource e Multiplataforma. É um software de versionamento de código o qual utiliza uma arquitetura de Cliente-Servidor para armazenar as versões do projeto atual. **CVS por ser um sistema centralizado**, precisa de um servidor central ao qual os desenvolvedores precisam sincronizar-se para realizar as suas alterações e não é possível dar "checkouts" reservados.

SVN: Apache Subversion é um sistema de controle de versão e revisão de software de código aberto sob a licença Apache. Gerenciou arquivos e pastas que estão presentes no repositório. Ele pode operar através da rede, o que o permite e é usado por pessoas em computadores diferentes.

É um software de **versionamento de código** o qual também usa uma **arquitetura de cliente-servidor** para armazenar versões do projeto. Diferente do **CVS** o **git** funciona sem um **servidor central**, permitindo aos desenvolvedores terem seus próprios repositórios enquanto há um repositório público o qual podem ser enviadas as alterações.

- **Sendo um sistema de controle de versão descentralizado**, commits podem ser feitos mais frequentemente, permitindo a implementação de funcionalidades através de diversos commits.

CVS e SVN

GIT	SVN
Git é um sistema de vice-controle distribuído de código aberto desenvolvido por Linus Torvalds em 2005. Ele enfatiza a velocidade e a integridade dos dados	Apache Subversion é uma versão de software de código aberto e um sistema de controle de revisão sob a licença Apache.
Git tem um modelo distribuído.	SVN tem um modelo centralizado.
No git, cada usuário tem sua própria cópia de código em seu local, como sua própria ramificação.	No SVN existe um repositório central com uma cópia de trabalho que também faz alterações e confirma no repositório central.
No git, não exigimos nenhuma rede para executar a operação git.	No SVN, exigimos Rede para executar a operação SVN.
Git é mais difícil de aprender. Tem mais conceitos e comandos.	O SVN é muito mais fácil de aprender em comparação com o git.
O Git lida com um grande número de arquivos, como arquivos binários, que mudam rapidamente, por isso se tornam lentos.	SVN controla facilmente o grande número de arquivos binários.
No git, criamos apenas o diretório .git.	No SVN criamos o diretório .svn em cada pasta.
Não possui uma boa interface do usuário em comparação com o SVN.	SVN tem interface de usuário simples e melhor.

CVS e SVN

GIT	SVN
<p>Características do GIT:</p> <ul style="list-style-type: none">• Sistema distribuído.• Ramificação.• Compatibilidade.• Desenvolvimento não linear.• Leve.• Código aberto.	<p>Características do SVN:</p> <ul style="list-style-type: none">• Os diretórios são versionados• Copiar, excluir e renomear.• Metadados com versão de formato livre.• Compromissos atômicos.• Ramificação e marcação.• Mesclar rastreamento.• Bloqueio de arquivo.

Git - Conceitos

Repositório: Lugar onde ficam guardados os arquivos sob controle do CVS que compõem os diversos módulos. Um repositório pode ser local (armazenado na própria estação de trabalho) ou em um servidor remoto.

Aqui está uma visão geral básica de como o Git funciona:

- Crie um "repositório" (projeto) com uma ferramenta de hospedagem de git (como o Bitbucket)
- Copie (ou clone) o repositório na sua máquina local
- Adicione o arquivo ou seu repositório local e faça "commit" (salve) as alterações
- "Coloque" suas alterações na sua ramificação principal
- Faça uma alteração no seu arquivo com uma ferramenta de hospedagem de git e faça commit
- "Puxe" as alterações para a sua máquina local
- Crie uma "ramificação" (versão), faça uma alteração, faça commit da alteração
- Abra uma "solicitação pull" (proponha alterações na ramificação principal)
- "Mescle" sua ramificação com a principal

Git - Instalação

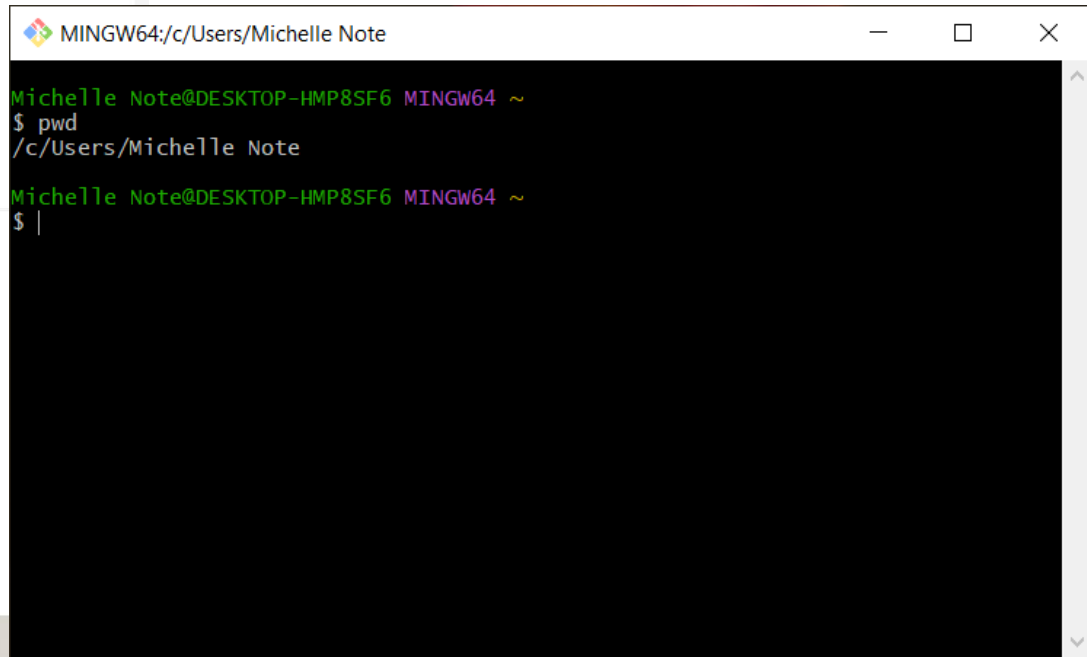
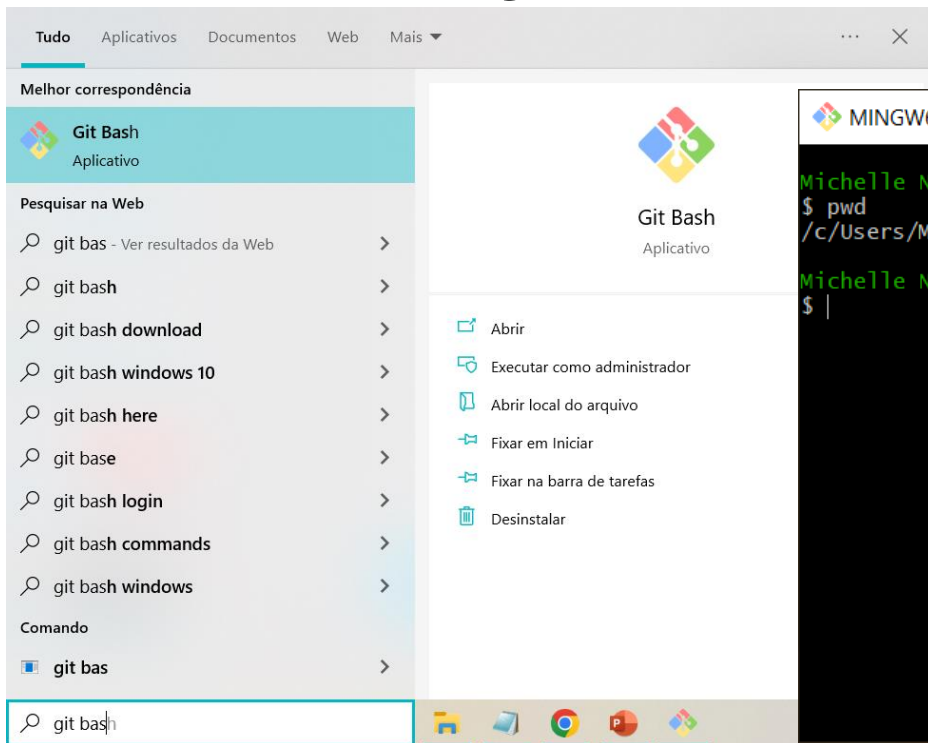
O primeiro passo será abrir o terminal, se estiverem no Mac ou Linux. Se estiverem utilizando Windows, podem utilizar o Git Bash, que vem junto com o Git.

Acessar o site abaixo, faça o download, em seguida proceder a instalação:

<https://gitforwindows.org/>

O Git Bash é o aplicativo para ambientes do **Microsoft Windows** que **oferece a camada de emulação para a experiência de linha de comando Git**. Bash é acrônimo para "*Bourne Again Shell*". Shells são aplicativos terminais usados como interface em sistemas operacionais por meio de comandos gravados.

Git - Instalação



Git Bash

O comando do Bash **pwd** é usado para exibir o "**diretório de trabalho atual**". O **pwd** é equivalente ao executar o **cd** em terminais DOS. Essa é a pasta ou o caminho no qual está a sessão de Bash atual.

O comando do Bash **ls** é usado para listar o conteúdo do diretório de trabalho atual. O **ls** é equivalente ao **DIR** em terminais de host do console do Windows.

O Bash e o host do console do Windows têm o comando **cd** (Change Directory). **O cd é chamado junto com o nome do diretório.**

Git Bash – Configuração Inicial

Configuração LOCAL:

Acessar o Git Bash e configurar a sua conta de nome e e-mail:

```
git config --global user.name "seu nome"
```

```
git config --global user.email "seu-email@email.com"
```

Git Bash – Configuração Inicial

Configuração para conexão com o GitHub

1- Abrir o GIT BASH e digite:

ssh-keygen -t rsa -C "seu-email@dominio.com"

ATENÇÃO: será solicitado o usuário e senha do seu GitHub.

2- Será criada a chave **SSH no diretórios C:\Users\nome-do-seu-usuario.ssh\id_rsa.pub.**

Acessar este arquivo e copiar o conteúdo da chave.

3- Acessar o **github, vá em configurações - SSH Keys - Add SHH Keys.**
Colar o SSH copiado e atribuir um título para a chave.

Git Bash – Configuração Inicial

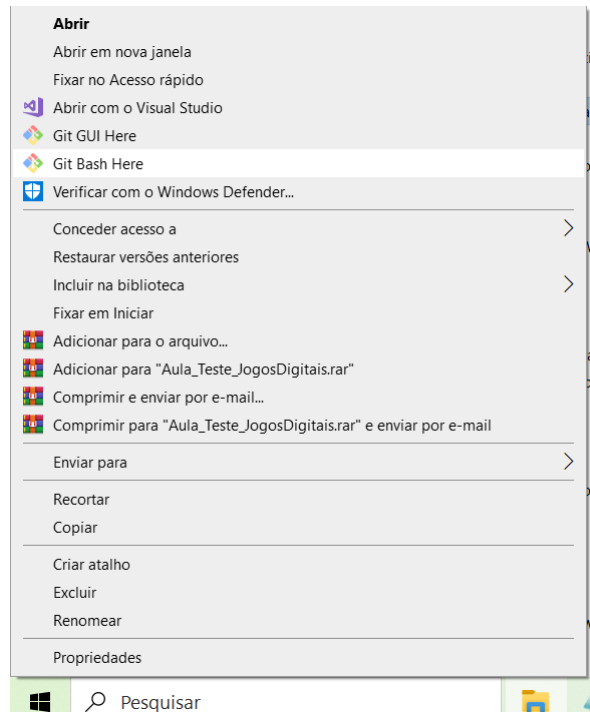
Depois de **instalar o Git...**

Crie uma pasta vazia onde será seu projeto ou clique em uma pasta que já existe para utilizar.

Para indicar que a pasta utilizará o serviço de Git, devemos inicializar:

git init

```
Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Testes/JogosDigitais
$ git init
Initialized empty Git repository in C:/Users/Michelle Note/Documents/GitHub/Aula_Testes/JogosDigitais/.git/
Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Testes/JogosDigitais
```



Clonar Repositório do GitHb

Git clone: é uma comando para baixar o código-fonte existente de um repositório remoto (ex: Github). Em outras palavras, **git clone**, basicamente, faz uma cópia idêntica da versão mais recente de um projeto em um repositório e a salva em seu computador.

`git clone <https://link-com-o-nome-do-repositório>`

git clone https://github.com/mihanne/Exemplos_POO.git

Branch

2. Git branch

Branches (algo como ramificações, em português) são altamente importantes no mundo do git. Usando as branches, vários desenvolvedores conseguem trabalhar em paralelo no mesmo projeto simultaneamente.

Podemos criar branches:

git branch <nome-da-branch>

Esse comando criará uma branch em seu local de trabalho. Para fazer o **push (algo como enviar)** da nova branch para o repositório remoto, você precisa usar o comando a seguir:

git push -u <local-remoto> <nome-da-branch>

Branch

Como ver as branches:

git branch ou git branch --list

Como excluir uma branch:

git branch -d <nome-da-branch>

Git Checkout

Para trabalhar em uma **branch**, primeiro, é preciso "**entrar**" nela. Usamos **git checkout**, na maioria dos casos, **para trocar de uma branch para outra**. Também podemos usar o comando **para fazer o checkout de arquivos e commits**.

git checkout <nome-da-branch>

Também existe um comando de atalho que permite criar e automaticamente trocar para a branch criada ao mesmo tempo:

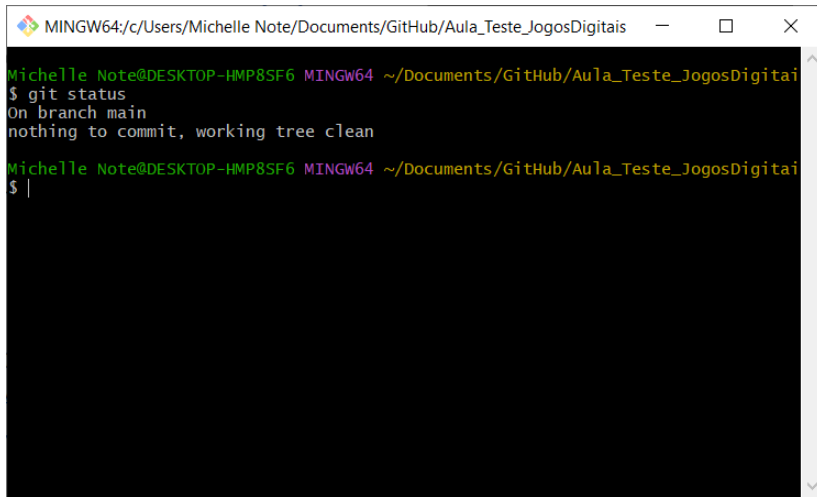
git checkout -b <nome-da-branch>

Esse comando cria a branch em seu espaço de trabalho local (a flag -b representa a branch) e faz o checkout na nova branch logo após sua criação.

Git Status

O comando **git status** nos dá todas as informações necessárias sobre a **branch atual**.

git status



```
MINGW64:/c:/Users/Michelle Note/Documents/GitHub/Aula_Teste_JogosDigitais
Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Teste_JogosDigitais
$ git status
On branch main
nothing to commit, working tree clean

Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Teste_JogosDigitais
$ |
```

Git Add

Ao criarmos, modificarmos ou excluirmos um arquivo, essas alterações acontecerão em nosso espaço de trabalho local e não serão incluídas no **próximo commit (a menos que alteremos as configurações)**.

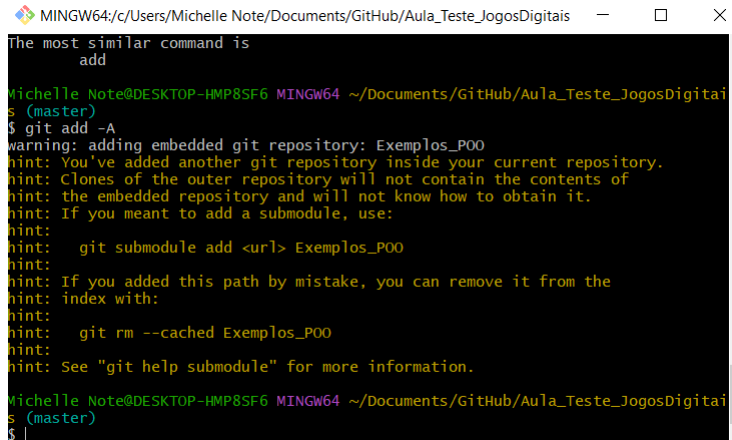
Precisamos usar o comando **git add** para incluir as alterações de um ou vários arquivos em nosso **próximo commit**.

Para adicionar um único arquivo:

git add <arquivo>

Para adicionar tudo ao mesmo tempo:

git add -A



```
MINGW64:/c/Users/Michelle Note/Documents/GitHub/Aula_Teste_JogosDigitais
The most similar command is
add
Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Teste_JogosDigitais (master)
$ git add -A
warning: adding embedded git repository: Exemplos_POO
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:   git submodule add <url> Exemplos_POO
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:   git rm --cached Exemplos_POO
hint: See "git help submodule" for more information.
Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Teste_JogosDigitais (master)
$
```

Git Commit

Importante: o comando git add não altera o repositório. As alterações não são salvas até que se use o git commit.

Quando chegamos a determinado ponto em desenvolvimento, queremos salvar nossas alterações (talvez após uma tarefa ou resolução de problema específica). Git commit é como definir um ponto de verificação no processo de desenvolvimento.

git commit -m "mensagem do commit"

Importante: git commit salva suas alterações no espaço de trabalho local.

Git Push

Após fazer o commit de suas alterações, a próxima coisa a fazer é enviar suas alterações ao servidor remoto. **Git push** faz o upload dos seus **commits** no **repositório remoto**.

```
git push <repositório-remoto> <nome-da-branch>
```

```
git push -u origin main
```

Enviando arquivos para o GitHub

- 1- Criar um repositório no GitHub
- 2- Criar uma pasta local e acessar este diretório no Git Bash. Clicar com o botão direito na pasta e escolher a opção **Git Bash Here**.

- 3- Inicializar o Git na pasta:

git init

- 4- Criar/copiar os arquivos para esta pasta

- 5- Adicionar arquivos ao repositório.

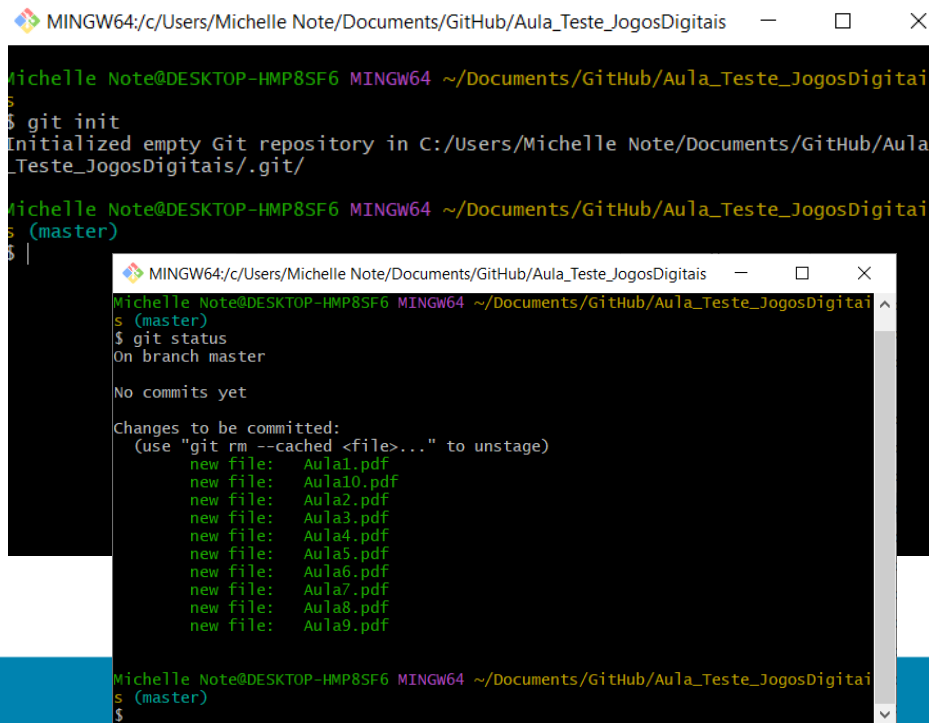
git add .

- 6- Verificar se os arquivos foram adicionados

git status

- 7- Adicionando os arquivos à Branch

git branch -M main



```
MINGW64/c:/Users/Michelle Note/Documents/GitHub/Aula_Teste_JogosDigitais
Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Teste_JogosDigitais
$ git init
Initialized empty Git repository in C:/Users/Michelle Note/Documents/GitHub/Aula_Teste_JogosDigitais/.git/

Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Teste_JogosDigitais
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   Aula1.pdf
    new file:   Aula10.pdf
    new file:   Aula2.pdf
    new file:   Aula3.pdf
    new file:   Aula4.pdf
    new file:   Aula5.pdf
    new file:   Aula6.pdf
    new file:   Aula7.pdf
    new file:   Aula8.pdf
    new file:   Aula9.pdf

Michelle Note@DESKTOP-HMP8SF6 MINGW64 ~/Documents/GitHub/Aula_Teste_JogosDigitais
$
```


Enviando arquivos para o GitHub

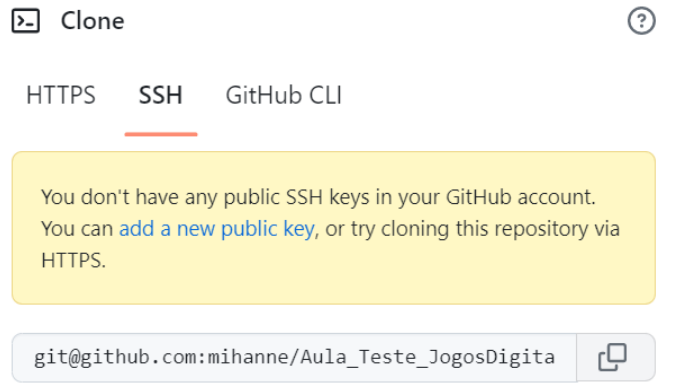
8- Conectando-se com seu repositório no github:

Acessar o Github e copiar o endereço de SSH

git remote add origin [git@github.com:mihanne/Aula_Testes_JogosDigitais.git](https://github.com/mihanne/Aula_Testes_JogosDigitais.git)

9- Enviando os arquivos selecionados para o GitHub

git push -u origin main



Clonando um repositório e enviando arquivos para o GitHub

1- Clonar o Repositório

git clone <https://github.com/YOUR-USERNAME/YOUR-REPOSITORY>

2- No seu computador, mova o arquivo do qual deseja fazer upload para o GitHub, no diretório local que foi criado quando o repositório foi clonado.

Abra Git Bash.

Mude o diretório de trabalho atual para o seu repositório local.

Prepare a arquivo para commit em seu repositório local.

\$ git add .

Clonando um repositório e enviando arquivos para o GitHub

3- Faça commit do arquivo que você preparou no repositório local.

\$ git commit -m "Add existing file"

4- Efetue push das alterações no repositório local para o GitHub.com.

\$ git push origin YOUR_BRANCH

Referências

- Atlassian - <https://www.atlassian.com/br/git>
- Full Cycle - <https://fullcycle.com.br/git-e-github/>
- https://pt.wikiversity.org/wiki/CVS_vs_Git
- <https://docs.github.com/pt/repositories>