

A dark, moody photograph of a person sitting at a desk, working on a laptop. Their hands are visible, holding a smartphone. The image is dimly lit, with the primary light source coming from the laptop screen, which is partially visible on the left. The overall tone is professional and focused.

SHIFT

FIAP

GIT & GITHUB - HANDS ON

DO BÁSICO AO AVANÇADO

Agenda

- **Apresentação**
- **Módulo 1** - Introdução aos sistemas de controle de versão
- **Módulo 2** - Instalação e Configuração
- **Módulo 3** - Inicializando um Repositório Local
- **Módulo 4** - Navegação entre as versões
- **Módulo 5** - Desfazendo as alterações
- **Módulo 6** - Trabalhando com Branch e Tag
- **Módulo 7** - Ignorar arquivos no repositório
- **Módulo 8** - Trabalhando com Repositórios Remotos
- **Módulo 9** - Boas Práticas



WHO AM I?

Uma breve apresentação...

Who Am I?

Douglas Cabral

Especialista em Desenvolvimento de Aplicações e Games para Dispositivos Móveis e Internet das Coisas.

Graduado em Análise e Desenvolvimento de Sistemas.

Desenvolvedor responsável pelos sites e portal EAD da FIAP.

Professor de Java, Android e IOT dos cursos de graduação na FIAP.

Mais de 10 anos de experiência em desenvolvimento de sistemas.

CONTATO

<https://www.linkedin.com/in/douglascabral/>



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

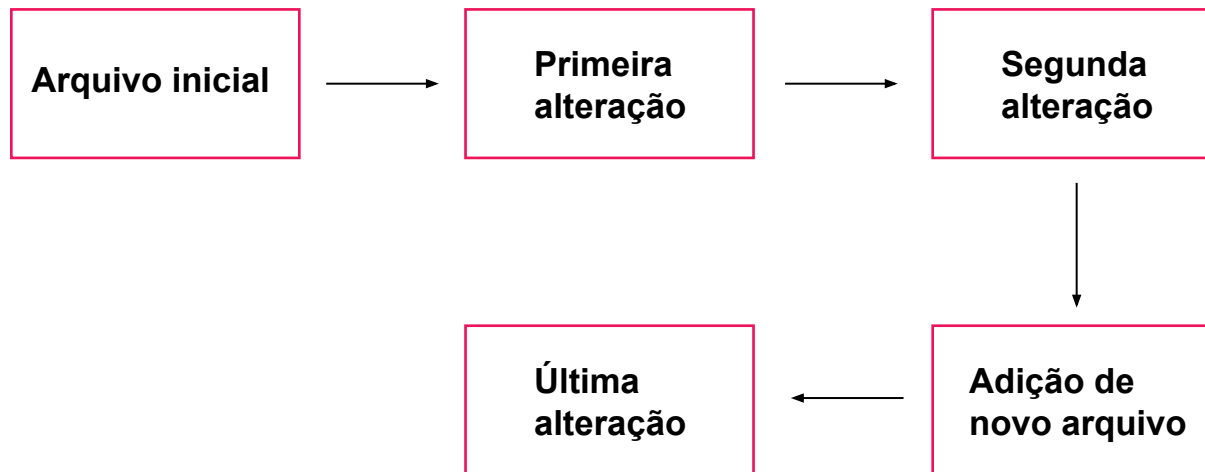
INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

- O que é um sistema de controle de versão
- Tipos de controle de versão
- Controle de versão local
- Controle de versão centralizado
- Controle de versão distribuído
- Diferença entre GIT e GitHub

INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

O QUE É UM SISTEMA DE CONTROLE DE VERSÃO?

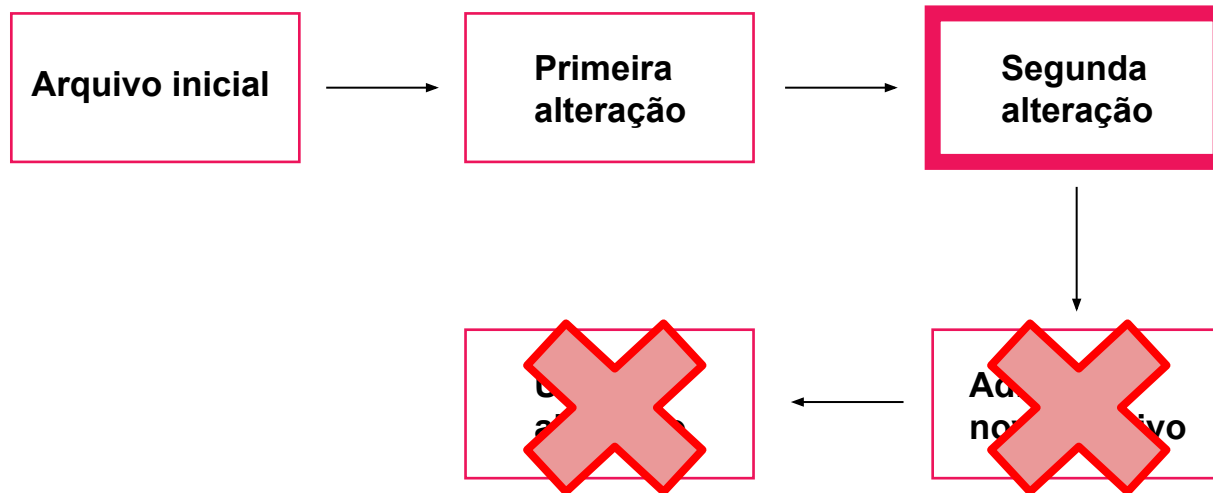
É um sistema com uma inteligência de registrar as alterações feitas em arquivos ao longo do tempo de um projeto.



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

O QUE É POSSÍVEL COM UM SISTEMA DE CONTROLE DE VERSÃO?

- Desfazer uma ou mais alterações;
- Saber quem alterou e quando alterou;
- Por que alterou;
- Quais arquivos foram alterados;



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

QUÊM PODE USAR?

Os principais adeptos são:

- Programadores
- Web Designers
- Designers Gráficos
- Redatores
- Editores de conteúdo

Além de todos os profissionais citados, nada impede de utilizar para outra finalidade!

INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

FERRAMENTAS EXISTENTES



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

FERRAMENTAS QUE POSSUEM ALGUM TIPO DE VERSIONAMENTO DE ALTERAÇÕES



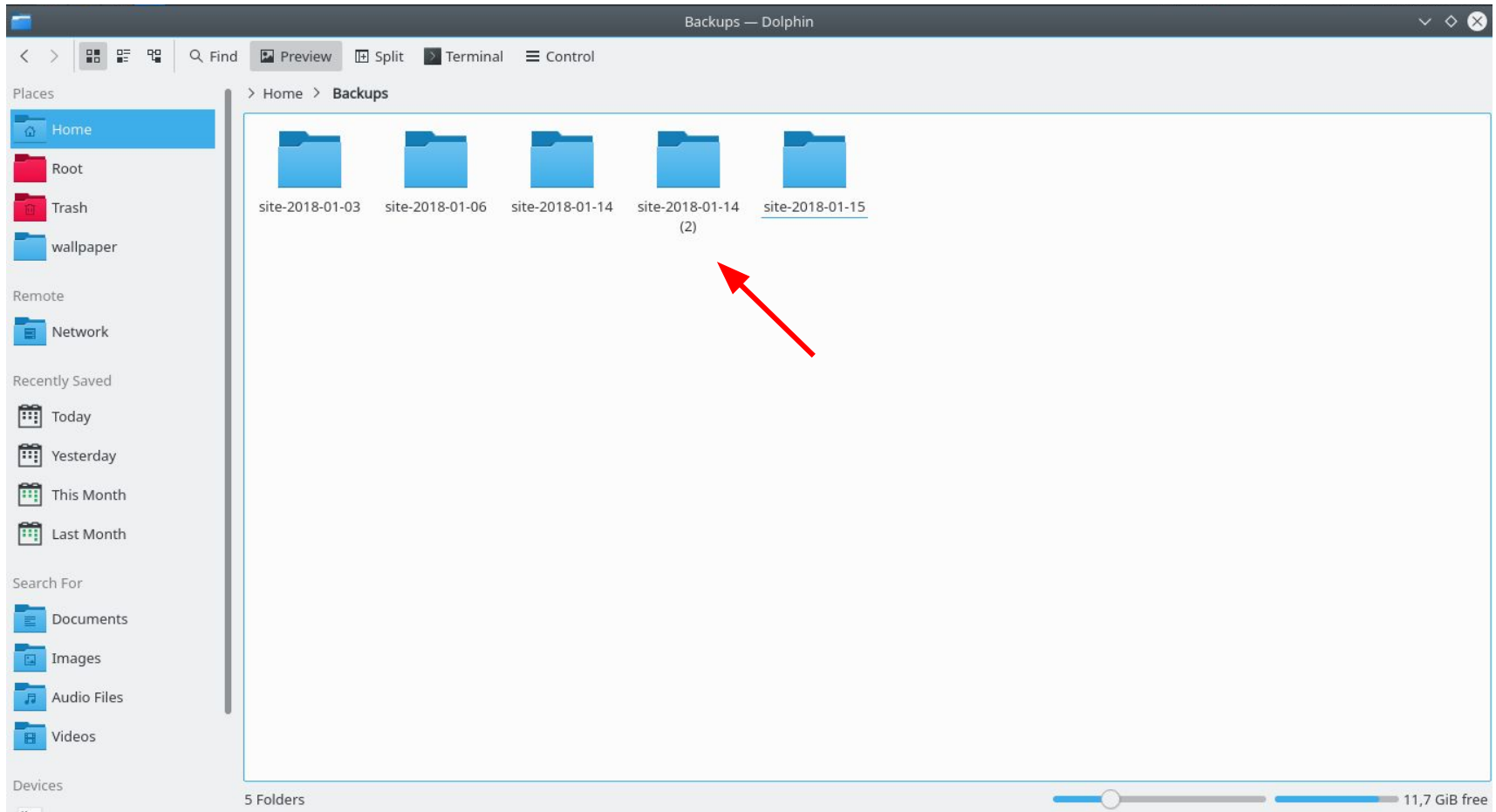
INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

VANTAGENS

- Histórico de alterações
 - Quem fez
 - Por que fez
 - Quando fez
- Trabalho em equipe
- Trabalho remoto (quando a ferramenta permite)
- Resgate de versões
- Ramificação do projeto
- Segurança
- Rastreabilidade
- Organização

INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

MÉTODO PREFERIDO DE CONTROLE DE VERSÃO POR MUITAS PESSOAS



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

MÉTODO PREFERIDO DE CONTROLE DE VERSÃO POR MUITAS PESSOAS

Nome ^	Tamanho	Tipo	Modificado	Permissões	Proprietário
..					
old		Pasta	16-03-2018 ...	drwxr-xr-x	root root
public		Pasta	26-05-2018 ...	drwxr-xr-x	www-dat...
_bkp--gdf_2015.zip	5.398.657	zip-arquivo	07-06-2015 ...	-rwxr-xr-x	root root

1 arquivo e 2 pastas. Tamanho: 5.398.657 bytes

INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

MÉTODO PREFERIDO DE CONTROLE DE VERSÃO POR MUITAS PESSOAS



classes

classes(10-06-2017)

Nome ^	Tamanho	Tipo
..		
admin		Pasta
classes		Pasta
classes(10-06-2017)		Pasta
classes(11-06-2017)		Pasta
classes-novo		Pasta
classes2		Pasta
config		Pasta
css		Pasta

Selecionadas 5 pastas.

Arquivo remoto/local	Direção	Arquivo remoto	Tamanho	Prioridad	Status
----------------------	---------	----------------	---------	-----------	--------



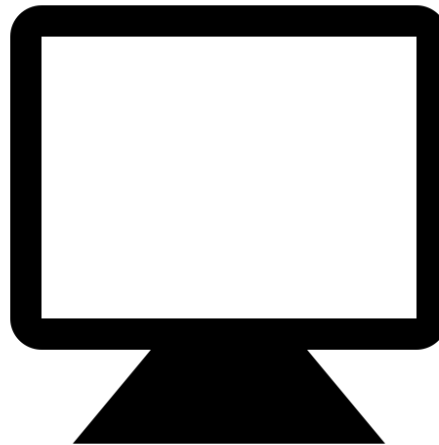
Qual desses diretórios contém os arquivos mais recentes?
Quem alterou? O que alterou? Por que alterou?

INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

CONTROLE DE VERSÃO LOCAL

Em um sistema de controle de versão local, o histórico de modificações estará no computador do usuário apenas.

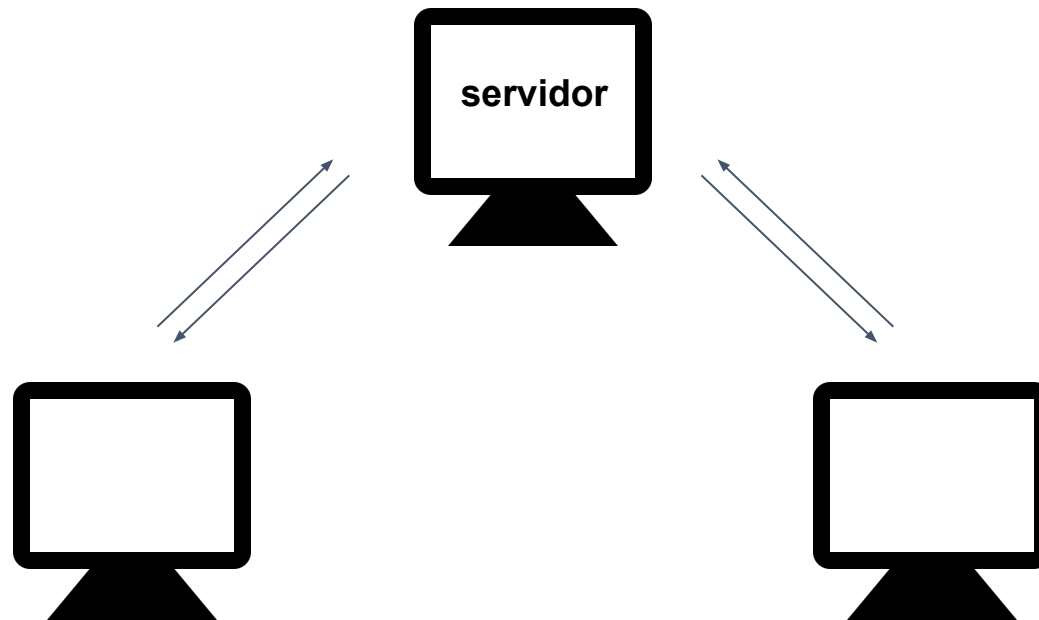
Isso dificulta o trabalho em equipe e fica suscetível a perda da informação em caso de falhas no computador.



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

CONTROLE DE VERSÃO CENTRALIZADO

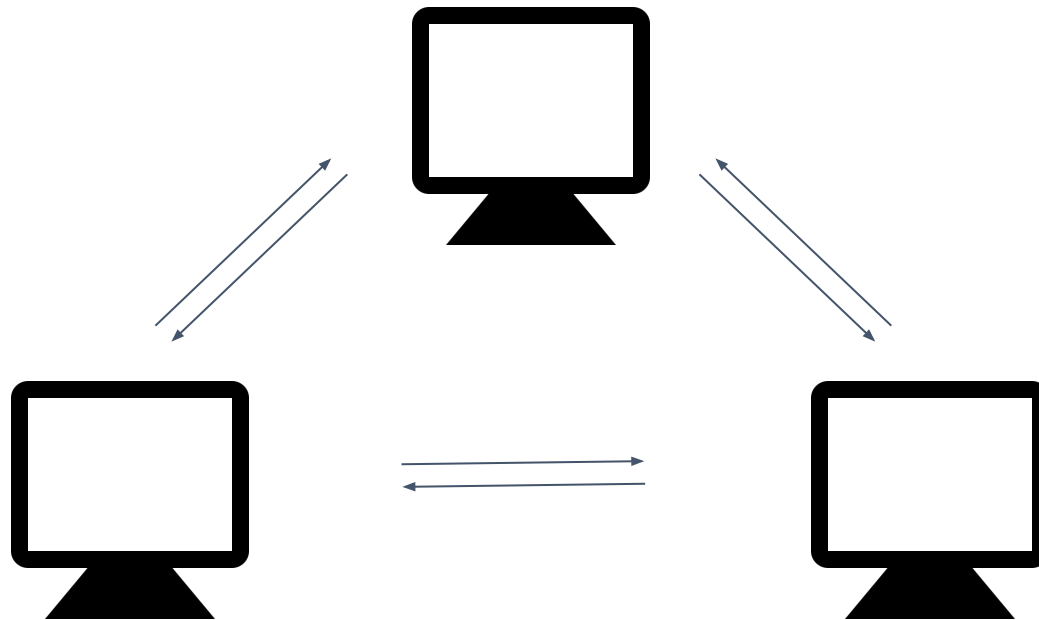
Para resolver o problema de se trabalhar em equipes, há o modelo centralizado, onde um servidor mantém os arquivos e os clientes podem resgatar ou enviar as modificações para este servidor.



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

CONTROLE DE VERSÃO DISTRIBUÍDO

Neste esquema, todos possuem uma cópia completa do repositório.



INTRODUÇÃO AOS SISTEMAS DE CONTROLE DE VERSÃO

NOSSA ESCOLHA...



Em nosso curso vamos trabalhar com a ferramenta GIT, por ser um modelo distribuído, gratuito e open source, além de estar disponível para as principais plataformas: Linux, Mac e Windows.



GIT

UMA BREVE HISTÓRIA DO GIT

GIT

UMA BREVE HISTÓRIA DO GIT

- Surgiu com a necessidade de versionar o código-fonte do Kernel Linux
- Nos primeiros 10 anos, o Kernel era versionado usando patches de tarballs
- Foi versionado posteriormente pelo BitKeeper (software proprietário)
- Após o acesso gratuito do BitKeeper ser removido, Linus decidiu criar o GIT.
- Há histórias de que, dias após iniciar o desenvolvimento o GIT já versionava seu próprio código-fonte.
- O GIT não armazena as diferenças, mas sim, o próprio arquivo.
- A ideia inicial era ser um software bem melhor que o CVS e derivados.



DIFERENÇAS ENTRE GIT E GITHUB

MUITAS PESSOAS ACREDITAM QUE AMBOS SEJAM A MESMA COISA



Software que possui a inteligência para versionamento e controle de versão de arquivos em um projeto.

GitHub

Rede social para compartilhamento de projetos versionados utilizando o GIT.

Aquisição do GitHub pela Microsoft:

<https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/>

INSTALAÇÃO E CONFIGURAÇÃO

INSTALAÇÃO E CONFIGURAÇÃO

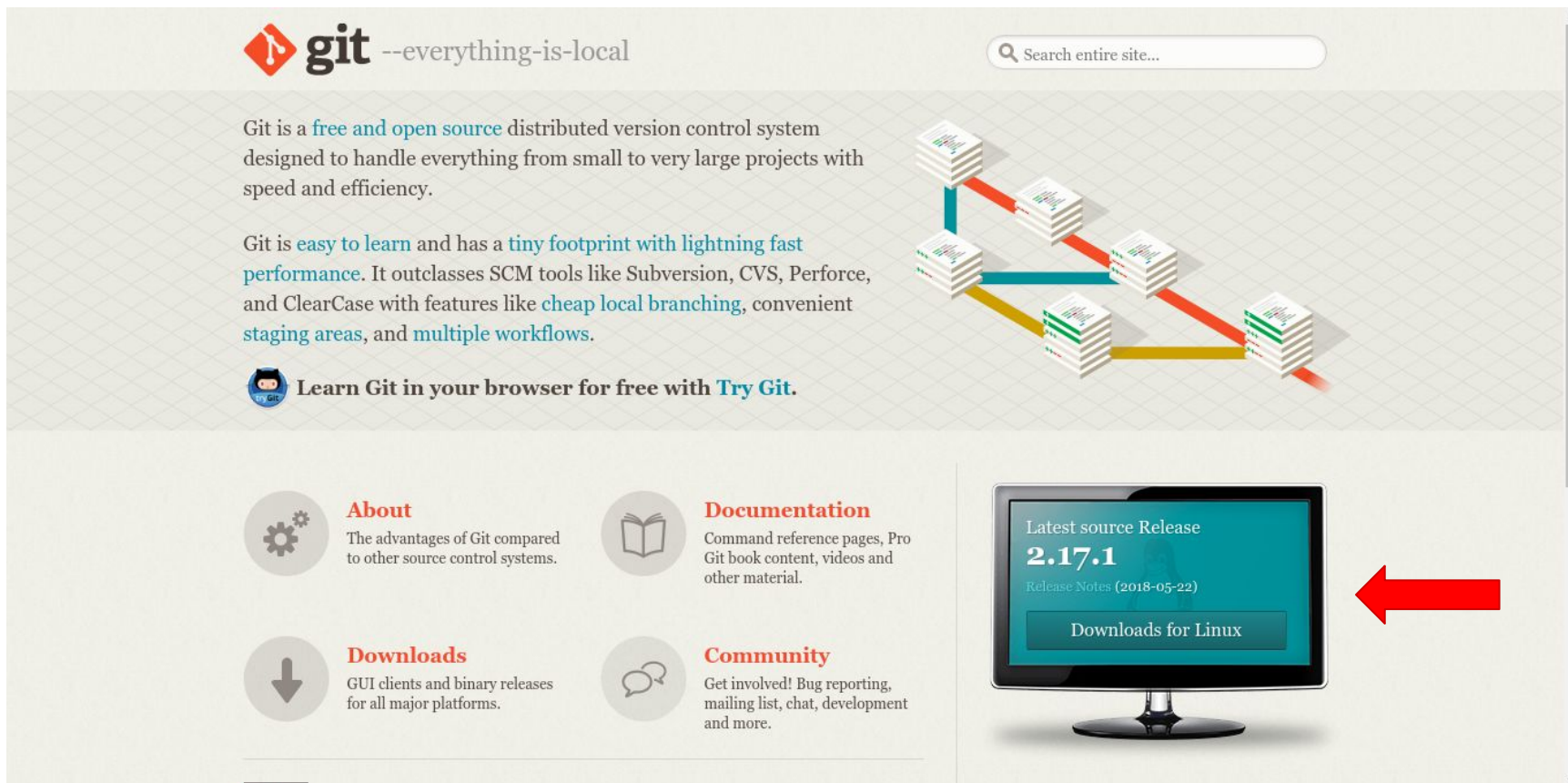
- Instalação do GIT nas plataformas Windows, Linux e Mac OS
- Configurações no momento da instalação
- Configurações pós instalação
- Entendendo o Bash

INSTALAÇÃO E CONFIGURAÇÃO

DOWNLOAD

Para download, basta acessar o link <https://git-scm.com/>.

A última versão disponível para o seu Sistema Operacional será exibido da seguinte forma:




The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--everything-is-local". A search bar is located in the top right corner. The main content area describes Git as a "free and open source distributed version control system" and highlights its "easy to learn" nature and "tiny footprint with lightning fast performance". A diagram on the right illustrates a branching model with multiple stacks of code connected by colored lines. Below the main text, there are four sections: "About", "Documentation", "Downloads", and "Community", each with an icon and a brief description. On the right side, a monitor displays the "Latest source Release 2.17.1" with a link to "Release Notes (2018-05-22)" and a button for "Downloads for Linux". A large red arrow points to this monitor.

git --everything-is-local

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

 Learn Git in your browser for free with **Try Git**.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.17.1
Release Notes (2018-05-22)
Downloads for Linux

INSTALAÇÃO E CONFIGURAÇÃO

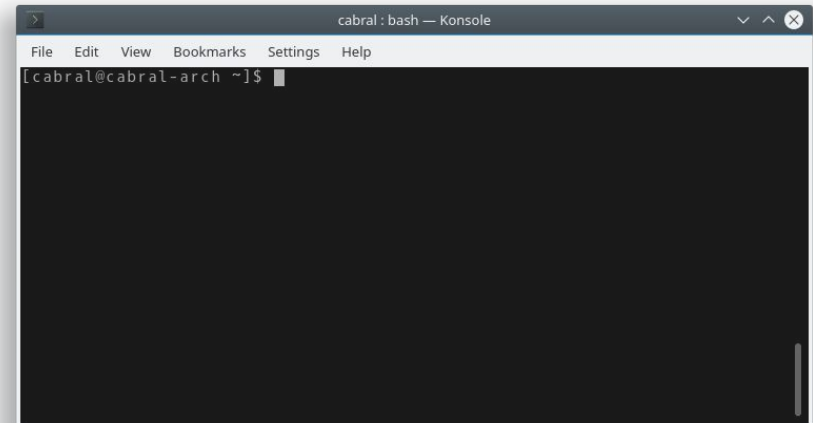
CONFIGURAÇÃO E USO DO TERMINAL / PROMPT DO COMANDO

Após a instalação concluída para o seu sistema operacional preferido, vamos trabalhar com o **GIT** utilizando o **terminal** ou **prompt do comando**.

No caso do **Windows**, o **GIT** pode instalar também o **GIT Bash**, uma ferramenta de terminal que facilita o uso do **git** na forma de comandos.

Abra o terminal na pasta onde deseja iniciar um repositório e versionar os arquivos.

(Windows: botão direito > Git Bash Here)



INSTALAÇÃO E CONFIGURAÇÃO

CONFIGURAÇÃO E USO DO TERMINAL / PROMPT DO COMANDO

A primeira configuração que vamos realizar, é informar a identificação do usuário do computador (Nome e E-mail), que será usado para descrever quem realizou determinado versionamento.

Essa configuração pode ser realizada de forma **local** (**apenas no projeto atual**) ou de forma **global** (**uma única vez, não necessitando ser realizada em cada projeto seguinte**).

Let's Go para o Hands-On!



A person with short hair and glasses is sitting at a desk, working on a laptop. The background is dark and out of focus, suggesting an office or home workspace. The person is wearing a dark top and has their hands on the laptop keyboard.

`git config --global user.name "Nome aqui"`

Aqui será informado, de forma global (opcional) para o GIT, o nome do desenvolvedor que está trabalhando neste computador.

Sempre use aspas duplas!

git config user.name

Para confirmar qual o nome do usuário setado no git para o projeto.



A person with short hair and glasses is sitting at a desk, working on a laptop. The background is dark and out of focus, suggesting an office environment. The person is wearing a dark top and has their hands on the laptop keyboard.

`git config --global user.email "email@aqui"`

Aqui será informado, de forma global (opcional) para o GIT, o e-mail do desenvolvedor que está trabalhando neste computador.

Sempre use aspas duplas!

git config user.email

Para confirmar qual o e-mail do usuário setado no git para o projeto.



git config --list

Recupera todas as informações de configurações do repositório.





git <comando> --help

Exibe uma descrição de ajuda com todas as opções de determinado comando.

INICIALIZANDO UM REPOSITÓRIO LOCAL

INICIALIZANDO UM REPOSITÓRIO LOCAL

- Criando o workspace para o nosso projeto
- Inicializando o repositório
- Entendendo o estado de arquivo Untracked
- Entendendo o estado de arquivo Unmodified
- Entendendo o estado de arquivo Modified
- Entendendo o estado de arquivo Staged
- Adicionando arquivos para serem versionados
- Entendendo o que são commits
- Primeiro commit

git init

Inicializa (cria) um repositório local.

Repare que é criado também um diretório oculto chamado .git

Nunca alterar manualmente nenhum arquivo do diretório .git !!!



git status

Exibe o estado das alterações existentes (ou não) no repositório.

untracked ⇒ Arquivo ainda não versionado

staged ⇒ Arquivo preparado para ser versionado

modified ⇒ Arquivo que teve seu conteúdo alterado

deleted ⇒ Arquivo excluído do projeto

A person with short blonde hair and glasses is sitting at a desk, working on a laptop. They are wearing a dark top and a watch. The background is a blurred office environment. The text is overlaid on the left side of the image.

git add <nome_do_arquivo>

Prepara um arquivo para ser versionado, ou seja, coloca este arquivo na staging area.

A person with short hair and glasses is sitting at a desk, working on a laptop. The scene is dimly lit, with the primary light source coming from the laptop screen. The person is wearing a dark top and has several bracelets on their left wrist. The background is out of focus, showing what appears to be a modern office environment.

git add <nome1> <nome2> <nome3...>

Prepara diversos arquivos para serem versionados, ou seja, adiciona os arquivos na staging area.



git add .

Coloca todos os arquivos alterados dentro do projeto
na staging area.



git add *

Coloca todos os arquivos alterados dentro do projeto
na staging area.

git commit -m “Mensagem explicando...”

Versiona todos os arquivos que estão na staging area do repositório.

Boas práticas: sempre colocar uma mensagem explicando a alteração.

OBS: Se o parâmetro **-m** não for informado, um editor de texto (no terminal) será aberto.

NAVEGAÇÃO ENTRE AS VERSÕES

NAVEGAÇÃO ENTRE AS VERSÕES

- Visualização do histórico
- Entendendo o histórico
- Entendendo o que é HEAD
- Navegação entre as versões
- Comparando as alterações

A woman with short blonde hair and glasses is sitting at a desk, working on a laptop. She is wearing a dark top and has several bracelets on her left wrist. The background is dark and out of focus, suggesting an office or workspace at night.

git log

Exibe a lista dos commits, com a descrição e detalhando quem fez e quando fez na ordem decrescente (do mais recente para o mais antigo).

OBS: Observe a marcação **HEAD**. Esta marcação mostra o ponto atual no histórico em que você está!

Se a lista for muito grande, para sair do modo scroll, pressione a letra “q”

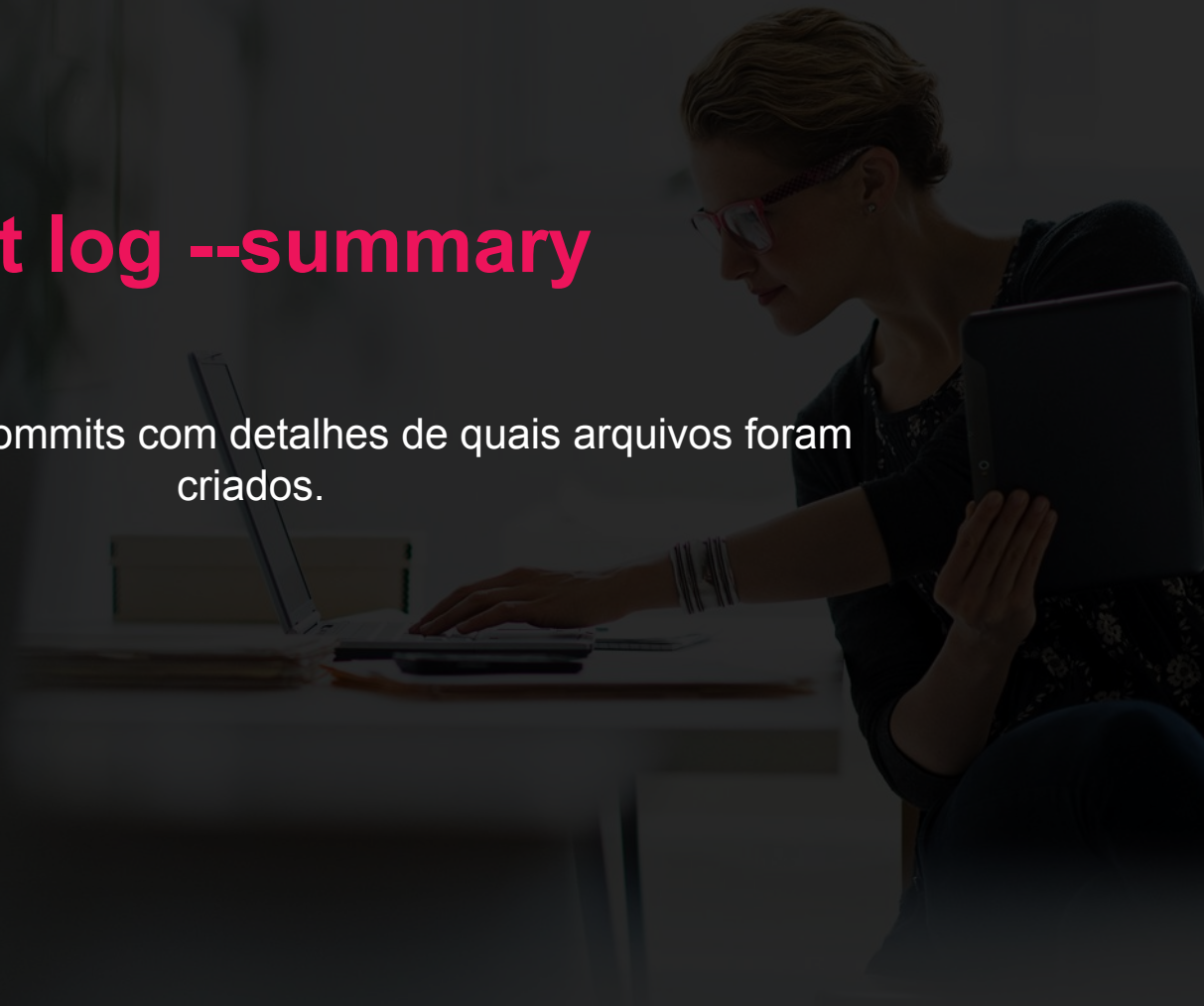
git log --reverse

Exibe a lista dos commits na ordem inversa
(do mais antigo para o mais recente).



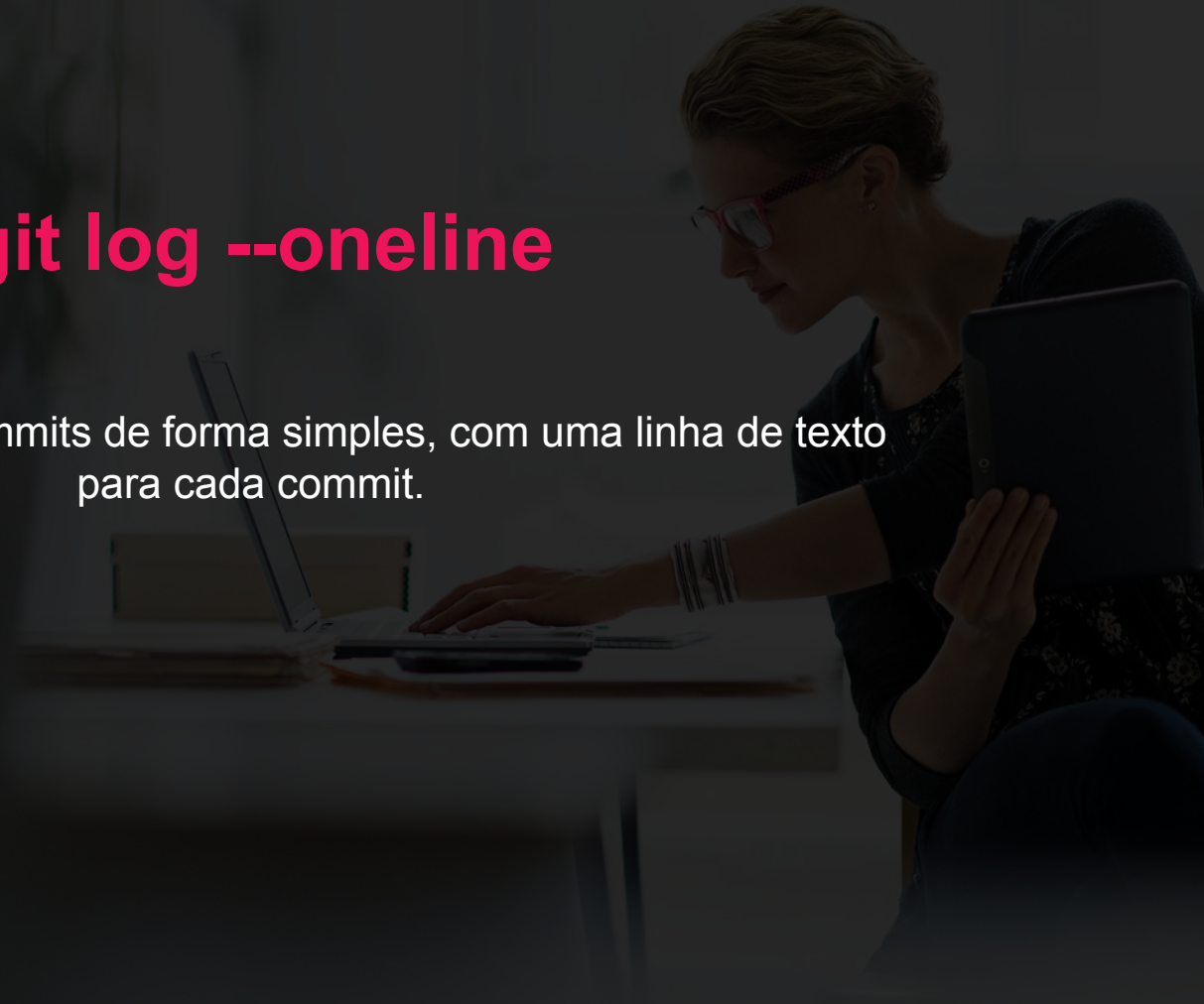
git log --summary

Exibe a lista dos commits com detalhes de quais arquivos foram criados.



git log --oneline

Exibe a lista dos commits de forma simples, com uma linha de texto para cada commit.



git log --graph

Exibe a lista dos commits desenhando um gráfico com todas as ramificações.



git checkout <identificador>

Navega entre as versões dos arquivos através de um identificador da versão.

OBS: Para voltar para a última versão da branch principal (master), basta digitar: git checkout master

git diff

Exibe as diferenças ocorridas nos arquivos.

OBS: Se a lista for muito grande, para sair do modo scroll, pressione a letra “q”

A woman with short blonde hair and glasses is sitting at a desk, working on a laptop. She is wearing a dark top and has several bracelets on her left wrist. The room is dimly lit, with light coming from the laptop screen and a window in the background. The text is overlaid on the image.

`git diff <identificador>`

Exibe as diferenças ocorridas nos arquivos do HEAD para o commit do identificador.

OBS: Se a lista for muito grande, para sair do modo scroll, pressione a letra “q”



```
git diff <identificador>..<identificador>
```

Exibe as diferenças ocorridas nos arquivos entre os identificadores.

OBS: Se a lista for muito grande, para sair do modo scroll, pressione a letra “q”

DESFAZENDO AS ALTERAÇÕES

DESFAZENDO AS ALTERAÇÕES

- Checkout de arquivos
- Reset
- Reset Soft
- Reset Mixed
- Reset Hard
- Revert
- Stash

git checkout <arquivo>

Desfaz as edições no arquivo, deixando-o como o commit mais próximo.



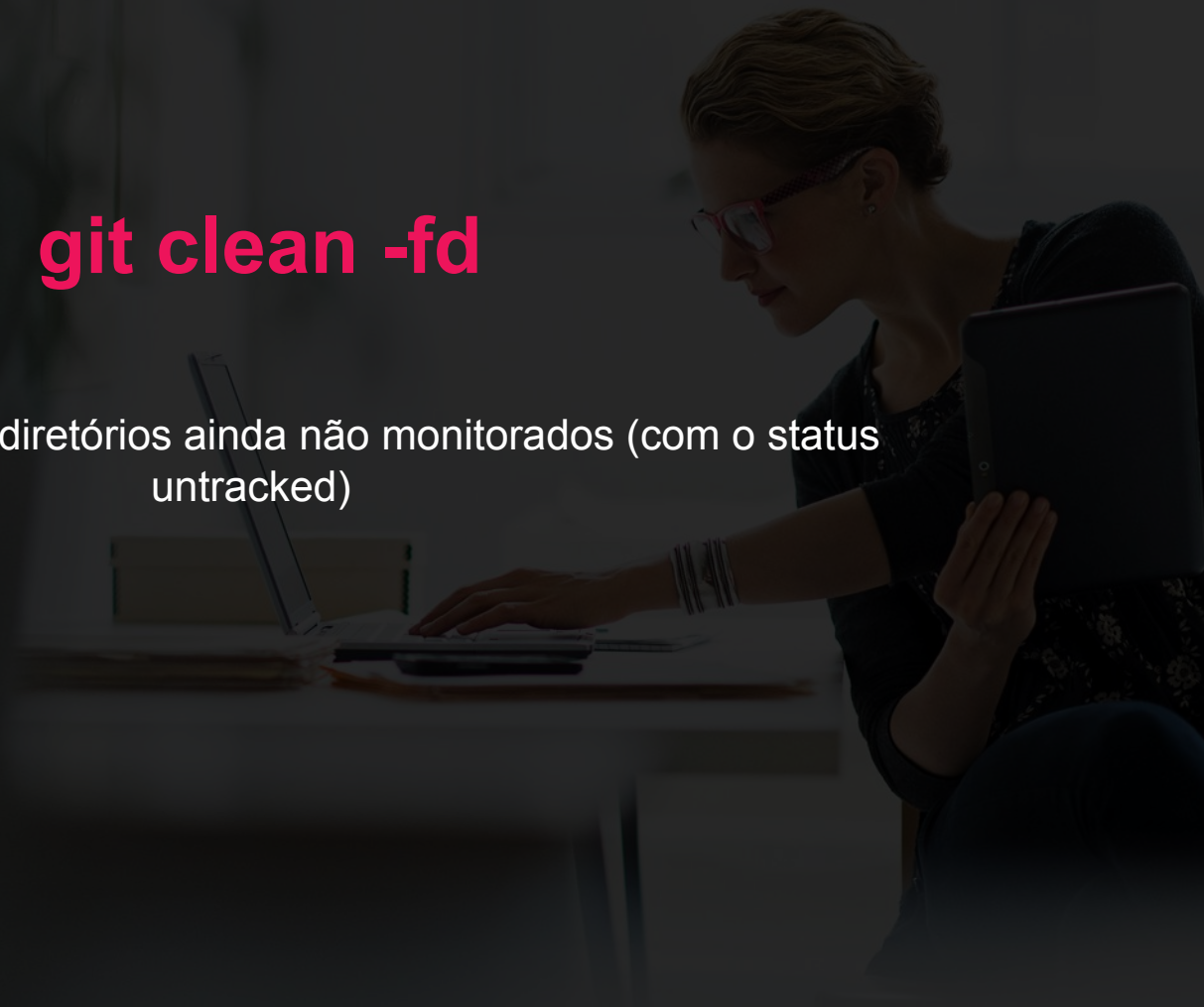
A person with short hair and glasses is sitting at a desk, working on a laptop. They are wearing a dark top and have several bracelets on their left wrist. The background is a blurred office setting.

`git checkout <commit> <arquivo>`

Desfaz as edições no arquivo, deixando-o como a versão presente no commit informado.

git clean -fd

Remove arquivos e diretórios ainda não monitorados (com o status untracked)





git reset HEAD <arquivo>

Remove um arquivo da staging area.

git reset --hard HEAD~<número>

Volta o projeto para um commit específico e “mata” todas as alterações.

OBS: O <número> representa a posição do commit no log, iniciando em 0, do mais recente para o mais antigo, a partir do HEAD.



```
git reset --soft HEAD~<número>
```

Volta para um commit específico e deixa as alterações (diferenças entre os commits desfeitos) na staging area.

OBS: O <número> representa a posição do commit no log, iniciando em 0, do mais recente para o mais antigo, a partir do HEAD.

A woman with short brown hair and glasses is sitting at a desk, working on a laptop. She is wearing a dark patterned top. The background is a blurred office environment with a whiteboard and some plants. The overall lighting is dim, with the primary light source being the laptop screen and some ambient light from the background.

`git reset --mixed HEAD~<número>`

Volta para um commit específico e deixa os arquivos com alterações com o status modified apenas.

OBS: O <número> representa a posição do commit no log, iniciando em 0, do mais recente para o mais antigo, a partir do HEAD.

git reflog

Exibe todo o histórico de commits do GIT, incluindo commits “apagados” com os comandos anteriores.



A woman with short blonde hair and glasses is sitting at a desk, working on a laptop. She is wearing a dark top and has her hands on the keyboard. The background is dark and out of focus, suggesting an office environment.

`git revert HEAD~<número>`

Volta para um commit específico e cria um commit registrando esta alteração.

OBS: O <número> representa a posição do commit no log, iniciando em 0, do mais recente para o mais antigo, a partir do HEAD.

git stash

Guarda as modificações ainda não comitadas em uma pilha temporária para serem recuperadas posteriormente.



git stash list

Lista a pilha de stash ainda não aplicada no repositório.



git stash apply

Aplica as alterações realizadas no último stash.



A person with short blonde hair and glasses is sitting at a desk, working on a laptop. They are wearing a dark top and have several bracelets on their left wrist. The background is a blurred office environment. The text is overlaid on the left side of the image.

`git stash apply <identificador>`

Aplica as alterações realizadas no stash especificado.

git stash apply --index

Aplica as alterações realizadas no último stash, incluindo arquivos na staging area.



git stash pop

Remove o stash mais recente da pilha.



TRABALHANDO COM BRANCH E TAG

TRABALHANDO COM BRANCH E TAG

- Entendendo o que são Tags
- Criação/Exclusão de Tags
- Comparação de Tags
- Entendendo o que são Branches
- Criação/Exclusão de Branches
- Comparação de Branches
- Merge de Branches
- Forçando um conflito de arquivo
- Correção de conflitos de arquivos
- Blame

A woman with short blonde hair, wearing glasses and a headband, is sitting at a desk and working on a laptop. She is wearing a dark top and has several bracelets on her left wrist. The background is a blurred office setting. The text 'git tag <identificador>' is overlaid on the image in a bright pink color.

git tag <identificador>

Cria um “apelido” para o commit, facilitando encontrá-lo e referenciá-lo.

git tag

Lista todas as tags existentes no repositório.





```
git tag -d <identificador>
```

Remove uma tag do repositório.

OBS: Isso não apaga o commit



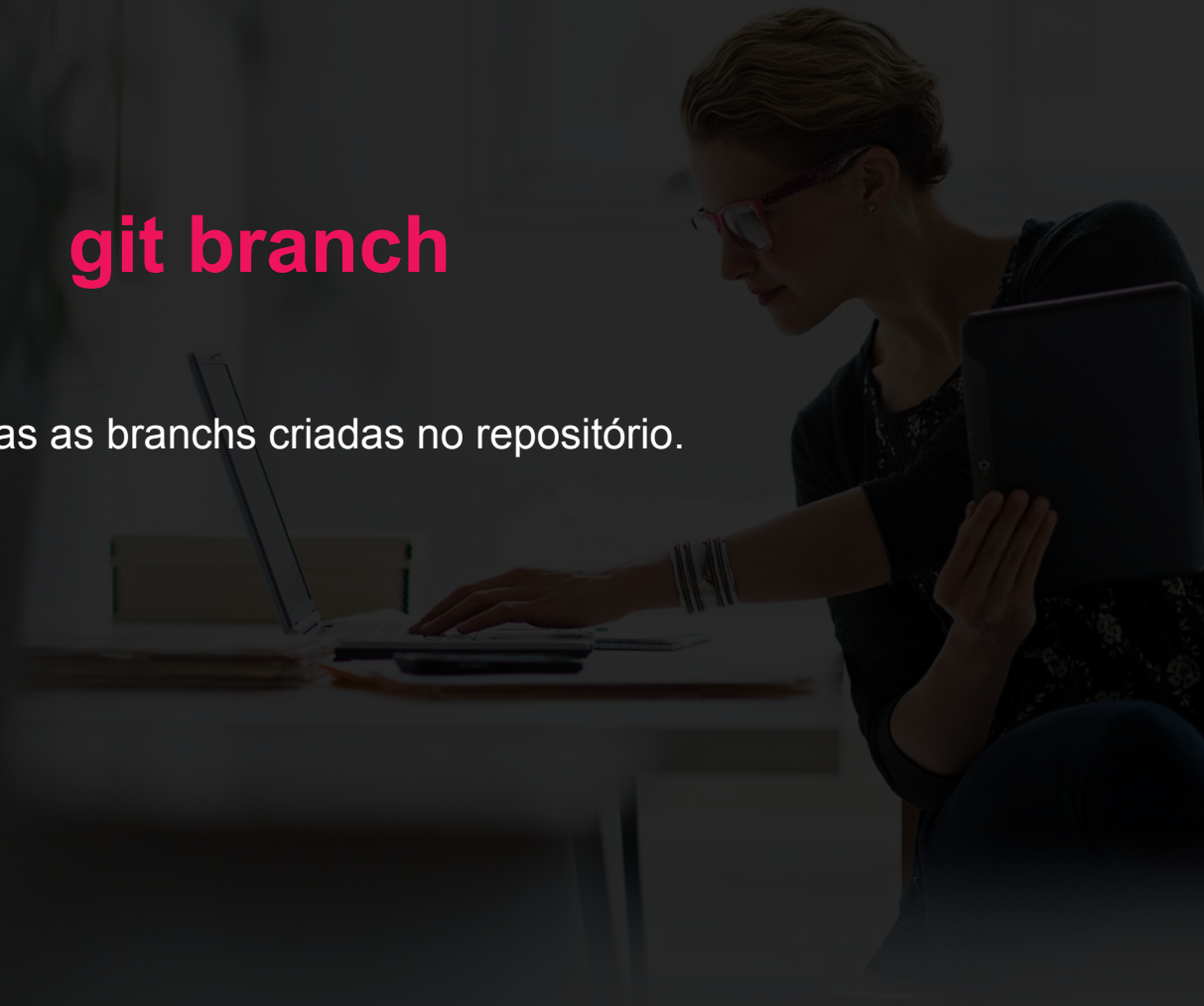
git branch <identificador>

Cria uma nova branch ramificada a partir do HEAD com o identificador informado.

OBS: apenas cria, não faz o checkout para ela.

git branch

Exibe todas as branches criadas no repositório.





git branch -d <identificador>

Remove uma branch do repositório,
caso não haja risco de perder modificações.



`git branch -D <identificador>`

Remove uma branch do repositório,
mesmo que haja perdas de modificações.

A person with short blonde hair and glasses is sitting at a desk, working on a laptop. They are wearing a dark top and have several bracelets on their left wrist. The background is a blurred office environment with plants and a window. The text is overlaid on the left side of the image.

`git checkout -b <identificador>`

Atalho para criar uma branch e já realizar o checkout para a mesma.

git merge <identificador>

Mescla as alterações entre o HEAD e a branch informada pelo identificador.



git cherry-pick <identificador>

Mescla apenas um commit informado na branch atual.

OBS: Isso altera o identificador do commit mesclado, devido sua data de aplicação diferente.





git blame <arquivo>

Exibe quem realizou a última modificação em cada linha do arquivo.

IGNORAR ARQUIVOS NO REPOSITÓRIO

IGNORAR ARQUIVOS NO REPOSITÓRIO

- Quando e por que ignorar alguns arquivos?
- Arquivo .gitignore
- Gerador de .gitignore

IGNORAR ARQUIVOS NO REPOSITÓRIO

COMO FAZER?

- Para ignorar um arquivo, basta criar um arquivo de texto com o nome “.gitignore” em alguma pasta do projeto.
 - Dentro deste arquivo deve conter o caminho dos arquivos que deverão ser ignorados no projeto.
 - Arquivos ignorados não serão mais monitorados pelo git.
 - O **ideal** é ignorar o arquivo quando ele **ainda está untracked**. Após isso é necessário remover o arquivo do cache do git usando o comando: **git rm –cached <arquivo>**

IGNORAR ARQUIVOS NO REPOSITÓRIO

EXEMPLO

```
1  #Exemplo de comentário no arquivo .gitignore
2
3  #Ignorar qualquer arquivo ou diretorio com o seguinte nome,
4  #em qualquer diretorio do projeto:
5  node_modules
6
7  #Ignorar somente diretorio com o seguinte nome,
8  #em qualquer diretorio do projeto
9  diretorio_ignorado/
10
11 #Ignorar somente arquivo com o seguinte nome,
12 #a partir da raiz do projeto
13 /ignorar_esse_arquivo/arquivo_ignorado.txt
14
15 #Ignorar qualquer arquivo zip
16 *.zip
```

touch .gitignore

Modo via terminal para se criar um arquivo chamado **.gitignore** .

OBS: Nada impede de criar através de um editor de texto, ou de qualquer outra maneira.





<https://github.com/github/gitignore>

Gerador de **.gitignore** online.

Basta colocar o tipo do projeto, sistema operacional, IDE, linguagem de programação, etc.

TRABALHANDO COM REPOSITÓRIOS REMOTOS

TRABALHANDO COM REPOSITÓRIOS REMOTOS

- Como funciona um repositório remoto
- GitHub
- Outras opções de repositórios remotos
- Realizando o Clone
- Realizando o Pull
- Realizando o Push
- Realizando o Fetch
- Entendendo o que é Remote
- Entendendo o que é Origin
- Merge Request
- Fork de projetos
- Gist

TRABALHANDO COM REPOSITÓRIOS REMOTOS

COMO FUNCIONA UM REPOSITÓRIO REMOTO

Em um repositório remoto podemos:

- Criar um novo repositório, geralmente enviando arquivos de um repositório local já existente.
- Clonar um repositório remoto para uma máquina local.

TRABALHANDO COM REPOSITÓRIOS REMOTOS

GITHUB

[Features](#)[Business](#)[Explore](#)[Marketplace](#)[Pricing](#)[Sign in](#) or [Sign up](#)

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

Username

Email

Password

Use at least one letter, one numeral, and seven characters.

[Sign up for GitHub](#)

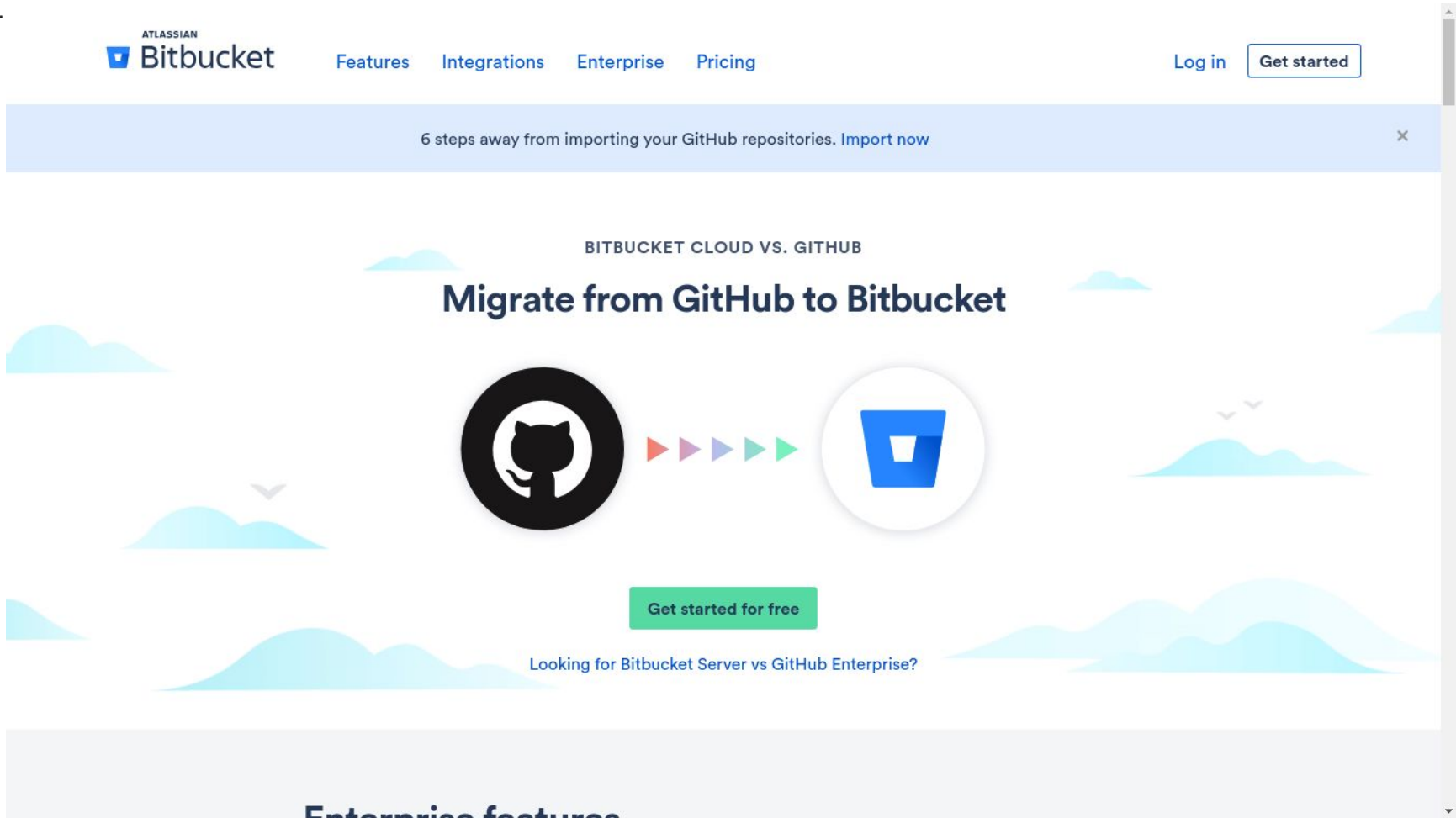
By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

<https://github.com/>

SHIFT | FIAP

TRABALHANDO COM REPOSITÓRIOS REMOTOS

BITBUCKET



<https://bitbucket.org/>

TRABALHANDO COM REPOSITÓRIOS REMOTOS

GITLAB



Product

Pricing

Install

Community

Blog

Contact



Explore GitLab.com

Sign In/Register

GitLab Installation

Introduction

We strongly recommend the Omnibus package installation since it is quicker to install, easier to upgrade, and it contains features to enhance reliability not found in other methods. We also strongly recommend at least 4GB of free memory to run GitLab.

Omnibus package installation (recommended)



Ubuntu



Debian



CentOS

<https://gitlab.com/>

git clone <endereço-do-repositório>

Realiza uma cópia do repositório remoto para a máquina local.

Cria automaticamente um diretório com o mesmo nome do repositório remoto.

GIT URLS:

```
ssh://[user@]host.xz[:port]/path/to/repo.git/  
git://host.xz[:port]/path/to/repo.git/  
http[s]://host.xz[:port]/path/to/repo.git/  
ftp[s]://host.xz[:port]/path/to/repo.git/
```



git clone <endereço> <diretório>

Realiza uma cópia do repositório remoto para a máquina local.

Cria um diretório com o nome informado no final do comando.

A person with short blonde hair and glasses is sitting at a desk, working on a laptop. They are wearing a dark top and several bracelets on their left wrist. The background is a blurred office environment with plants and a window. The overall image has a dark, muted color palette.

`git pull <remote-name> <branch-name>`

Recupera alterações presentes no repositório remoto para o repositório local.

A person with short blonde hair and glasses is sitting at a desk, working on a laptop. They are wearing a dark top and several bracelets on their left wrist. The background is a blurred office environment with a plant and a window. The overall image has a dark, moody aesthetic with a semi-transparent dark overlay.

`git push <remote-name> <branch-name>`

Envia alterações existentes no repositório local para o repositório remoto.

A person with short blonde hair and glasses is sitting at a desk, working on a laptop. They are wearing a dark top and have several bracelets on their left wrist. The background is a blurred office environment with plants and a window. The text is overlaid on the left side of the image.

`git push <remote-name> --all`

Envia alterações de todas as branches existentes no repositório local para o repositório remoto.



```
git push <remote-name> --tags
```

Realiza o envio de todas as tags
do repositório local para o repositório remoto.



```
git fetch <remote-name>
```

Realiza o download de objetos e referências do repositório remoto.

git remote -v

Exibe todos os repositórios remotos referenciados no repositório local.





```
git remote add <remote-name> <caminho>
```

Adiciona um novo repositório remoto.

Após isso executar o seguinte comando:
`git push -u <remote-name> <branch-name>`

Isso cria a associação das referências da branch atual com a branch remota.




```
git remote remove <remote-name>
```

Remove um repositório remoto antes configurado.


GITHUB

GIST


GitHubGist [All gists](#) [GitHub](#) [Sign up for a GitHub account](#) [Sign in](#)

 **Douglas Cabral**
dougascabral [View dougascabral on GitHub](#) [Sort: Recently created ▾](#)


[All gists 42](#) [Forked 3](#) [Starred 14](#)

 **dougascabral / upper-to-lower.sh** [1 file](#) [0 forks](#) [0 comments](#) [0 stars](#)
Created a month ago


```
1 echo "FRASE" | tr 'A-Z' 'a-z'
```

 **dougascabral / biggest-folders.sh** [1 file](#) [0 forks](#) [0 comments](#) [0 stars](#)
Created 2 months ago

```
1 sudo du -hsx * | sort -rh | head -10
```

 **dougascabral / .htaccess** [1 file](#) [0 forks](#) [0 comments](#) [0 stars](#)
Created 2 months ago
Parâmetros de segurança para o Apache

```
1 #bloqueia o acesso ao git
2 RedirectMatch 404 /\.\.git(/|$)
3
4 #bloqueia o acesso em arquivos e pastas ocultas
5 RedirectMatch 404 (?i)/\.\.+
6
7 #não permite listagem de pastas sem index
8 Options -Indexes
```

 **dougascabral / clone-db-mysql.sh** [1 file](#) [0 forks](#) [0 comments](#) [0 stars](#)
Created 2 months ago

<https://gist.github.com/>

BOAS PRÁTICAS

BOAS PRÁTICAS

- Não clonar repositórios dentro de outros repositórios.
- Não manter versionado arquivos compilados (.dll, .exe, etc).
- Não manter versionado arquivos com dados sensíveis (Arquivos de configuração, senha, ambiente).
- Ignorar arquivo antes do mesmo já estar versionado.

BOAS PRÁTICAS

- Deixar bem documentado as mensagens do commit e evitar mensagens genéricas como “Alterações no arquivo.txt”.
- Seguir algum modelo de workflow com branches (Ex: Branch por ambiente de desenvolvimento, ou branches por funcionalidades, ou ambos) .

Tks!

DOUGLAS CABRAL



<https://www.linkedin.com/in/douglascabral/>

Copyright © 2018 | Professor Douglas Cabral

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem consentimento formal, por escrito, do professor/autor.