

Introdução ao uso de dados geoespaciais no R

1 Controle de versão, git e GitHub

Maurício H. Vancine

Milton C. Ribeiro

UNESP - Rio Claro

Laboratório de Ecologia Espacial e Conservação (LEEC)

25/10/2021-05/11/2021

THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZIZE THESE SHELL
COMMANDS AND TYPE THEM TO SYNC UP.
IF YOU GET ERRORS, SAVE YOUR WORK
ELSEWHERE, DELETE THE PROJECT,
AND DOWNLOAD A FRESH COPY.



1 Controle de versão: git, GitHub e RStudio

Conteúdo

1 Conferir os computadores

2 Controle de versão

3 git e GitHub

4 Funcionamento do controle de versão

5 GitHub: Fork

6 Configuração: RStudio, git, GitHub
7 Iniciando localmente:
git init

8 Iniciando remotamente: git clone

9 Versionamento: git add, git commit e git status

10 Ignorando: .gitignore

11 Histórico: git log e git show

12 Diferenças: git diff

13 Desfazer: git revert e git reset

14 Ramificações: git branch, git switch e git merge

15 Remoto: git remote, git push e git pull

16 GitHub: Pull request

17 Detalhes do repositório do GitHub

18 Interface gráfica do RStudio

19 Principal material de estudo



1. Conferir os computadores



Software é aquilo que você xinga.
Hardware é aquilo que você chuta.



Café com Código

1. Conferir os computadores

R (4.1.1)



1. Conferir os computadores

RStudio (2021.09.0-351)



1. Conferir os computadores

git (2.33)



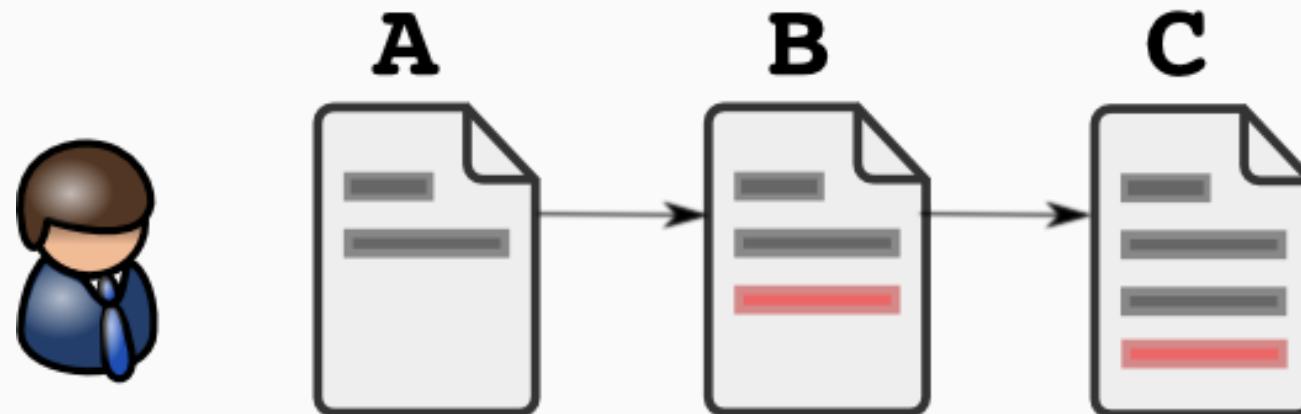
Tudo instalado? Então bora!

2. Controle de versão



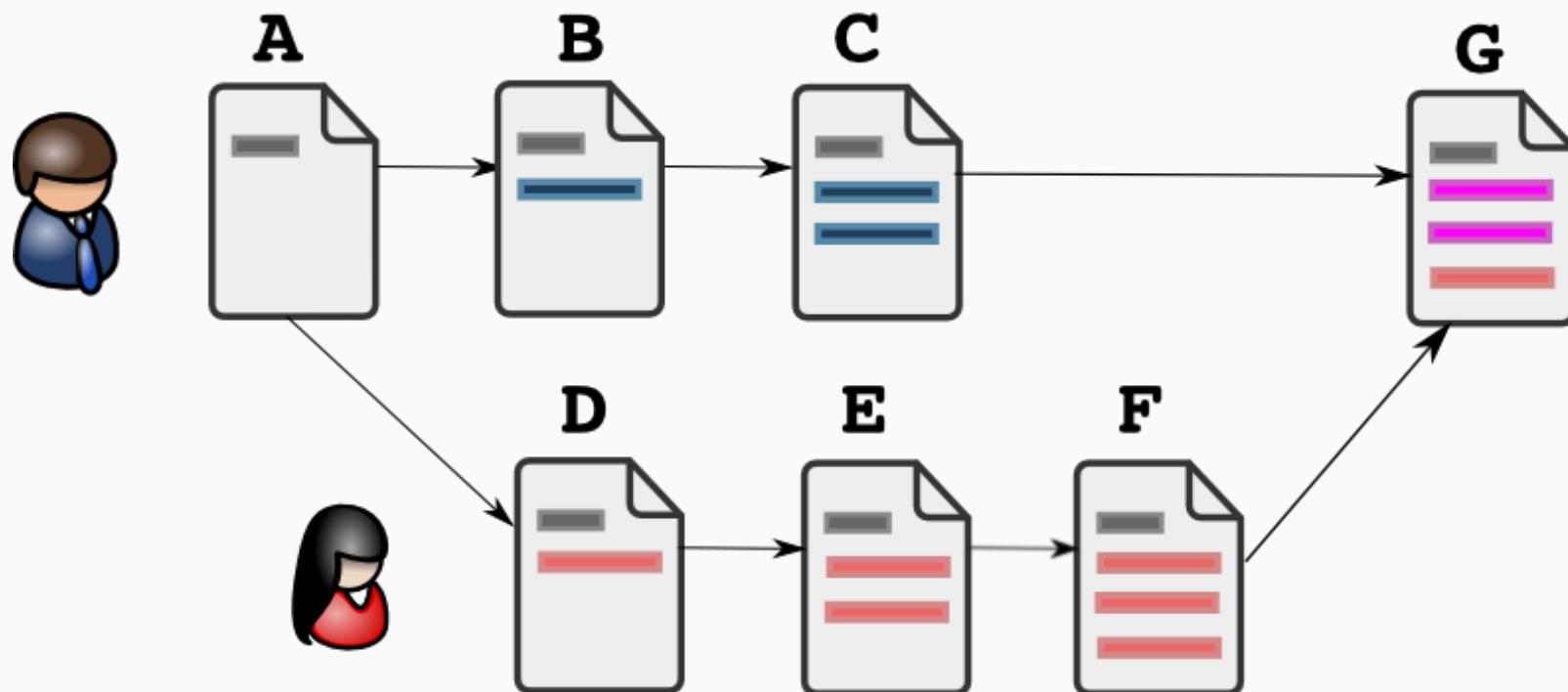
2. Controle de versão

Manejar projetos (diretório com vários arquivos) **individualmente**



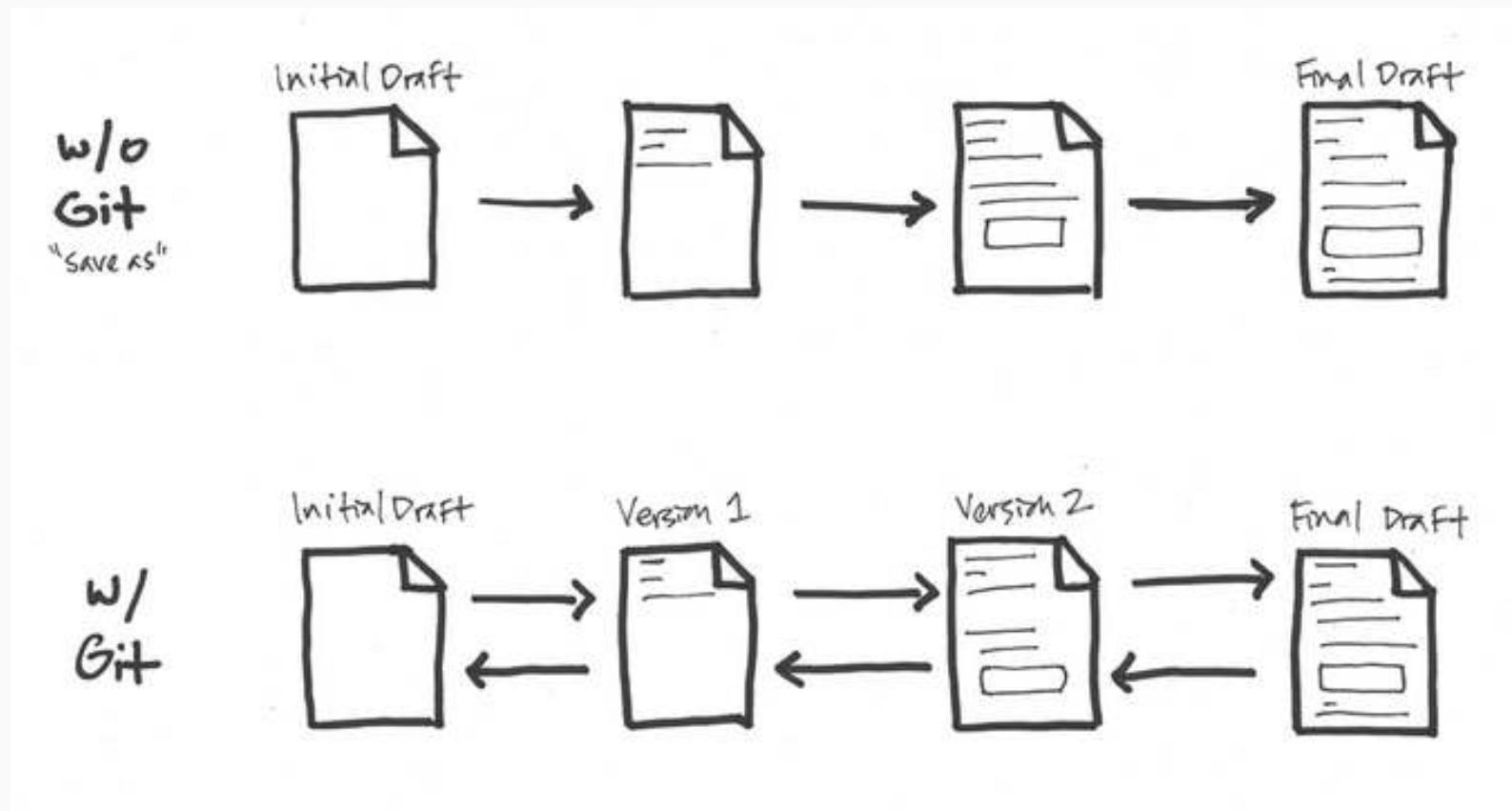
2. Controle de versão

Manejar projetos (diretório com vários arquivos) **compartilhados**



2. Controle de versão

Manejar projetos (diretório com vários arquivos) **no tempo**



2. Controle de versão

Principal **ferramenta** utilizada em grandes projetos

PLOS COMPUTATIONAL BIOLOGY

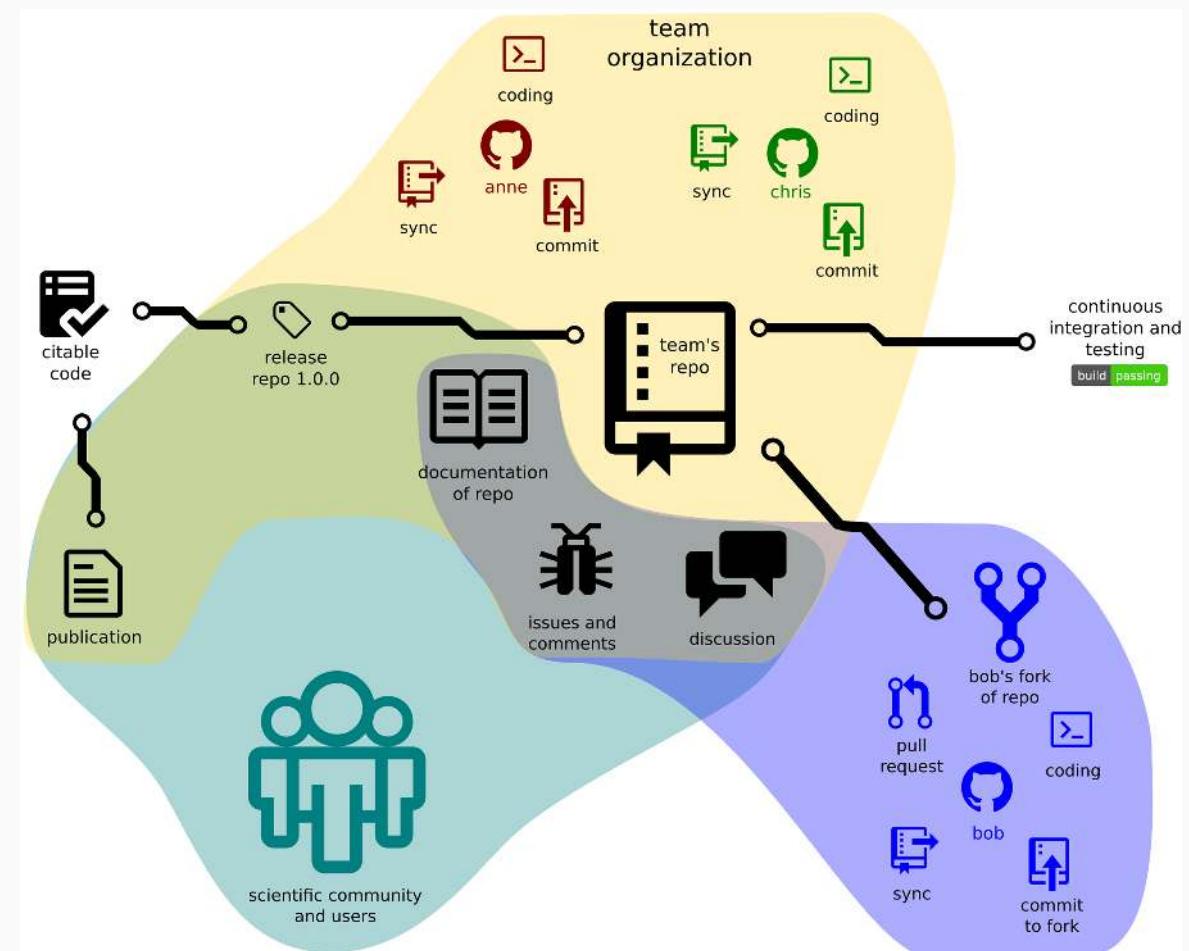
EDITORIAL

Ten Simple Rules for Taking Advantage of Git and GitHub

Yasset Perez-Riverol^{1*}, Laurent Gatto², Rui Wang¹, Timo Sachsenberg³, Julian Uszkoreit⁴, Felipe da Veiga Leprevost⁵, Christian Fufezan⁶, Tobias Ternent¹, Stephen J. Eglen⁷, Daniel S. Katz⁸, Tom J. Pollard⁹, Alexander Konovalov¹⁰, Robert M. Flight¹¹, Kai Blin¹², Juan Antonio Vizcaíno^{1*}

Name of the Material	URL
Git help and Git help -a	Document, installed with Git
Karl Broman's Git/Github Guide	http://kbroman.org/github_tutorial/
Version Control with Git/Version Control with Git	http://swcarpentry.github.io/git-novice/
Introduction to Git	http://git-scm.com/book/ch1-3.html
Github Training	https://training.github.com/
Github Guides	https://guides.github.com/
Good Resources for Learning Git and GitHub	https://help.github.com/articles/good-resources-for-learning-git-and-github/
Software Carpentry: Version Control with Git	http://swcarpentry.github.io/git-novice/

doi:10.1371/journal.pcbi.1004947.t002



3. git e GitHub

git: software (app) que faz o **controle de versão**

Maneja os **repositórios locais** (computador) e **remotos** (e.g. GitHub)

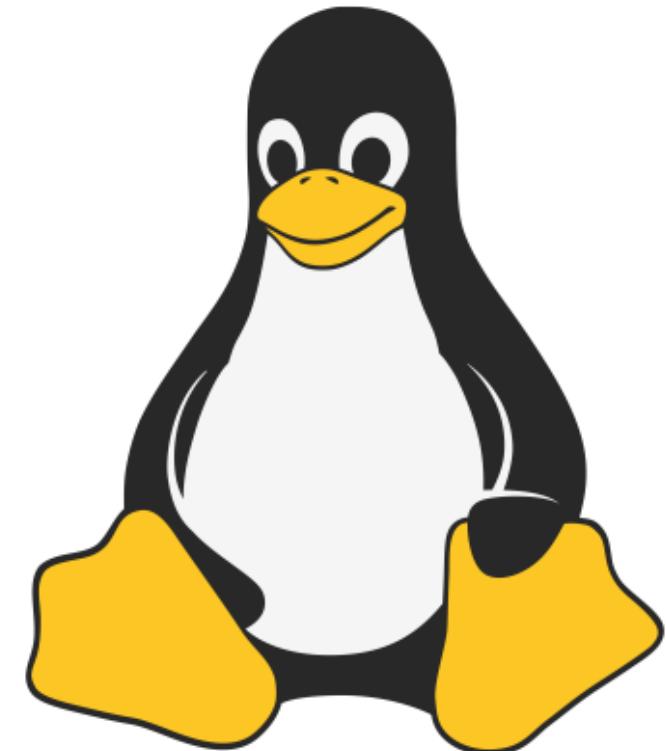


3. git e GitHub

git: software (app) que faz o **controle de versão**

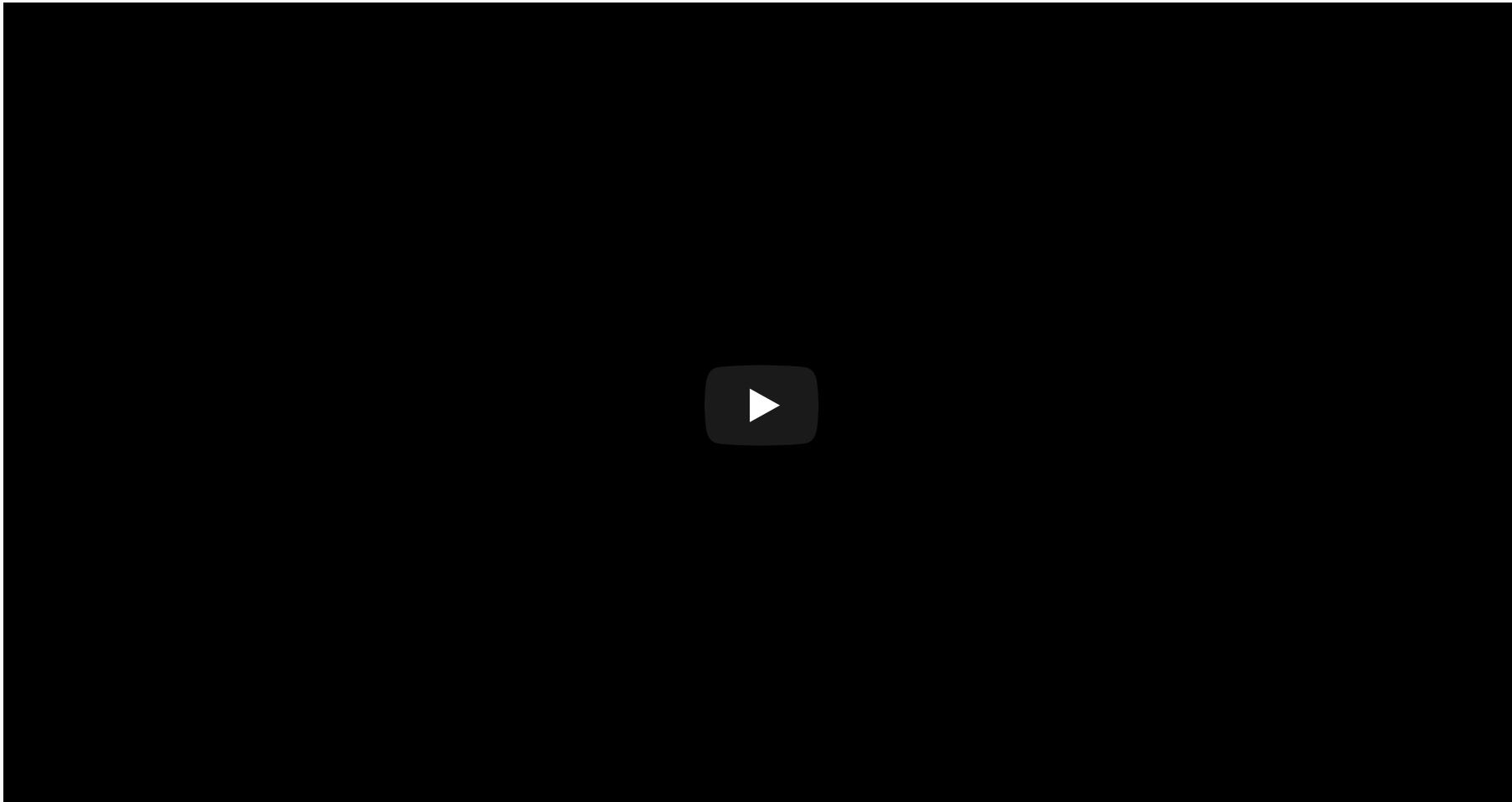
Criado por Linus Torvalds em 2005 para substituir o BitKeeper SCM no gerenciamento do Kernel Linux

Software livre, distribuído sob a licença GNU GPL v.2



3. git e GitHub

Entendendo git - Fabio Akita



3. git e GitHub

GitHub: repositório remoto

Plataforma de hospedagem de códigos com controle de versão



3. git e GitHub

ATENÇÃO: dois conceitos!

git: software (app) que faz o controle de versão

GitHub: repositório remoto que hospeda os arquivos versionados

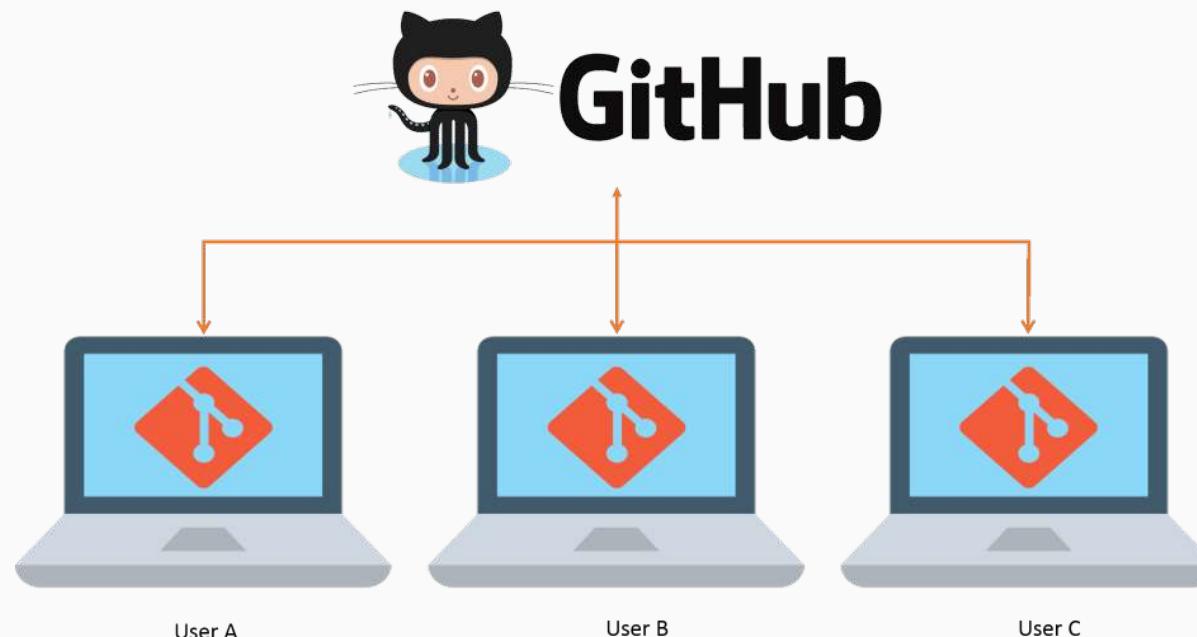


3. git e GitHub

ATENÇÃO: dois conceitos!

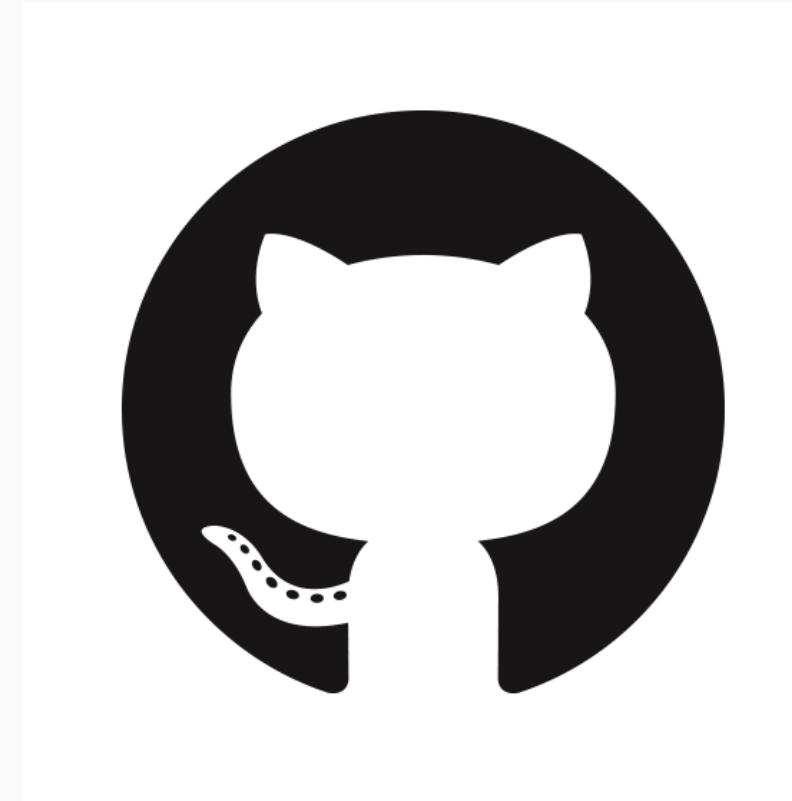
git: software (app) que faz o controle de versão

GitHub: repositório remoto que hospeda os arquivos versionados



3. git e GitHub

Vamos **criar uma conta no GitHub** (caso não possuam)



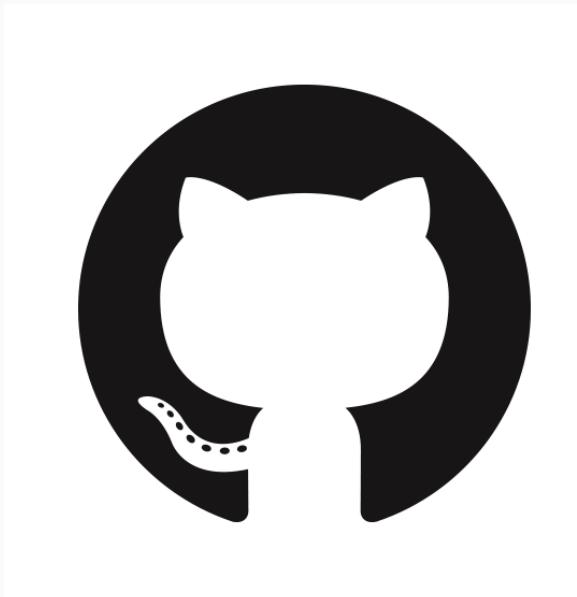
3. git e GitHub

Salvem ou recuperem essas informações!!!

username: mauriciovancine

email: mauricio.vancine@gmail.com

senha: !@#\$%^&*+

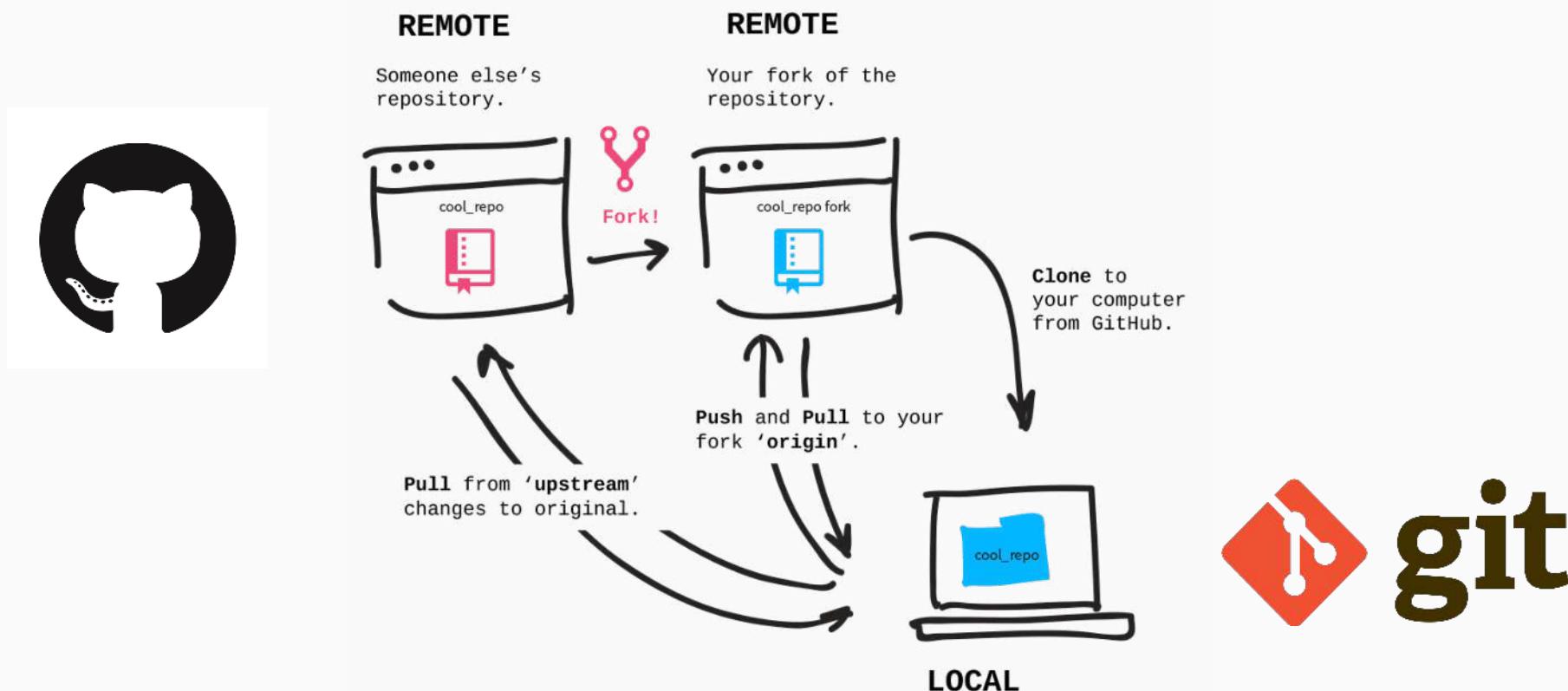


Tudo bem até aqui?

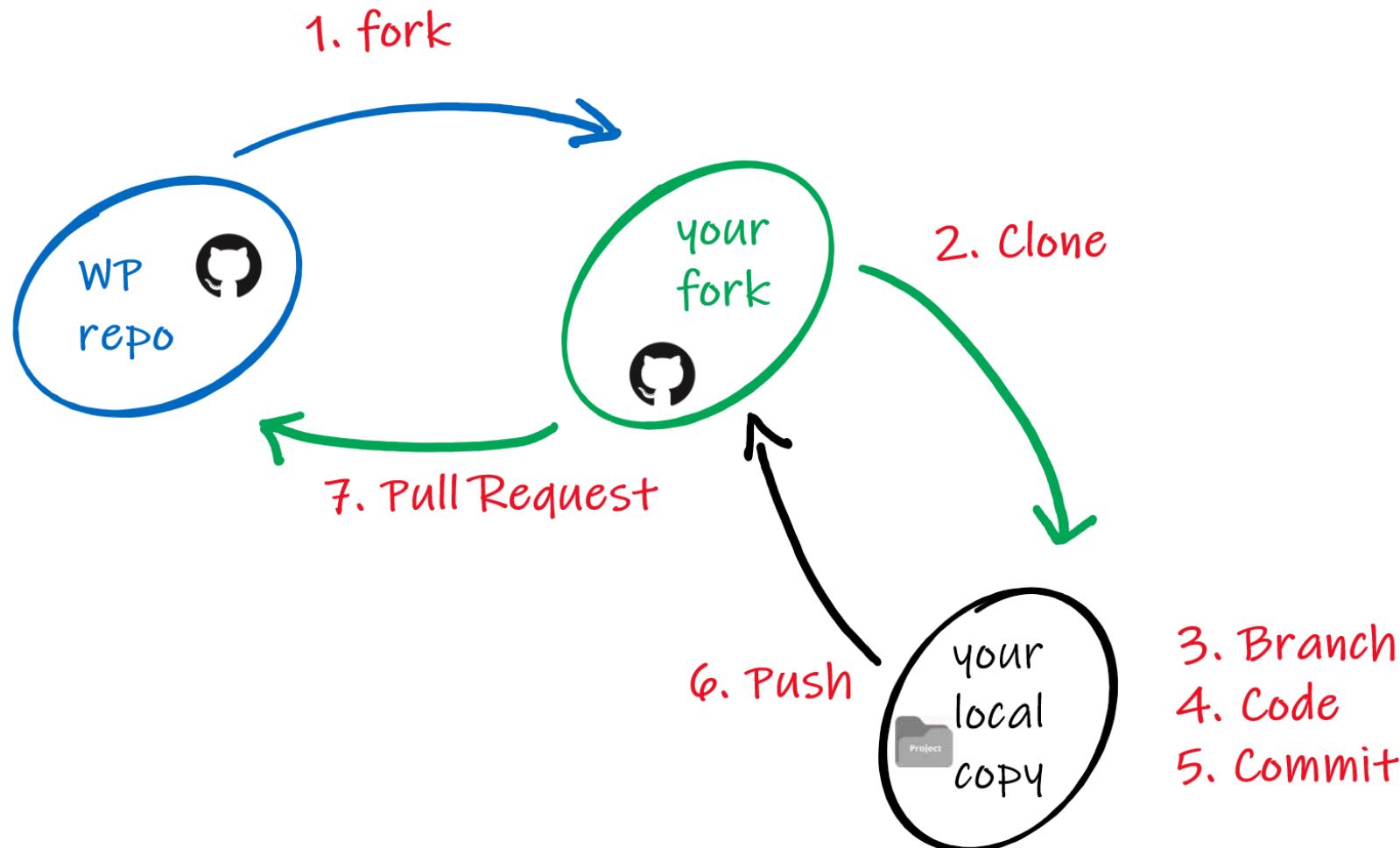
4. Funcionamento do controle de versão

Há duas formas de trabalhar com o git e GitHub para o versionamento

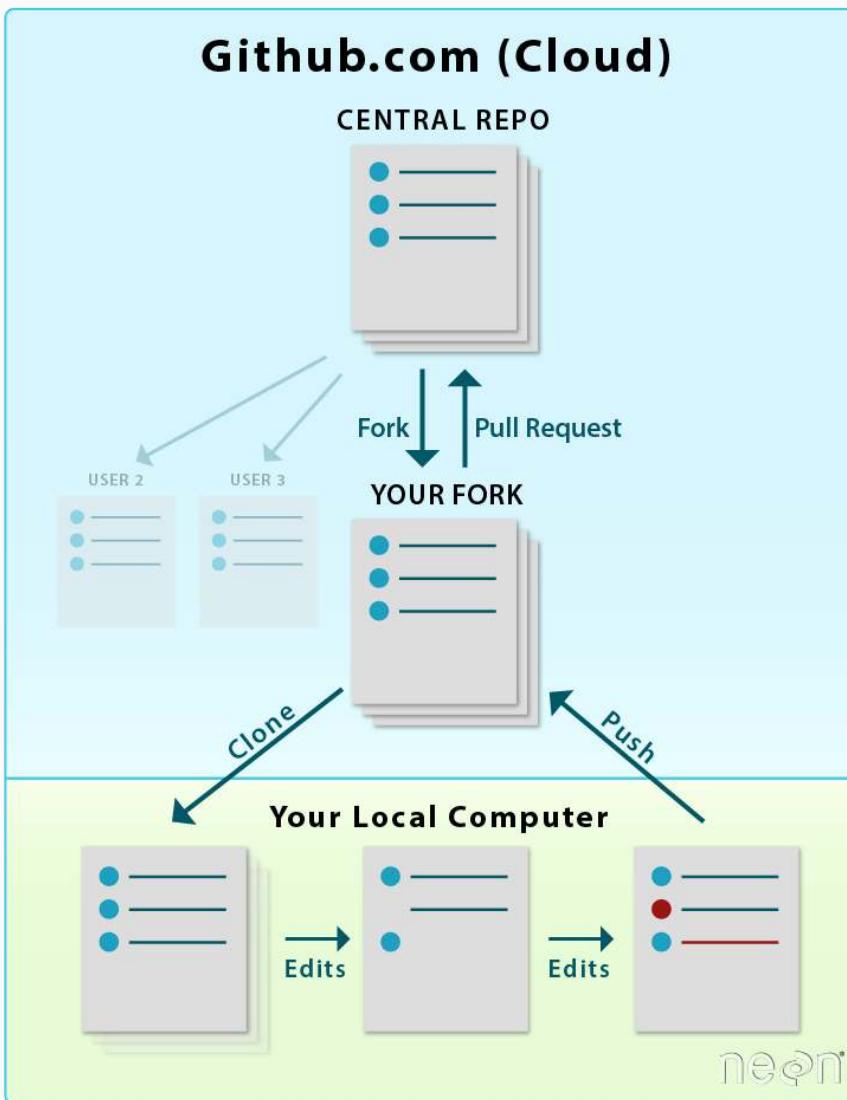
1. Iniciando um projeto em nosso computador (git)
2. Iniciando por um repositório remoto (GitHub) ["forkando" de alguém ou criando um repositório próprio]



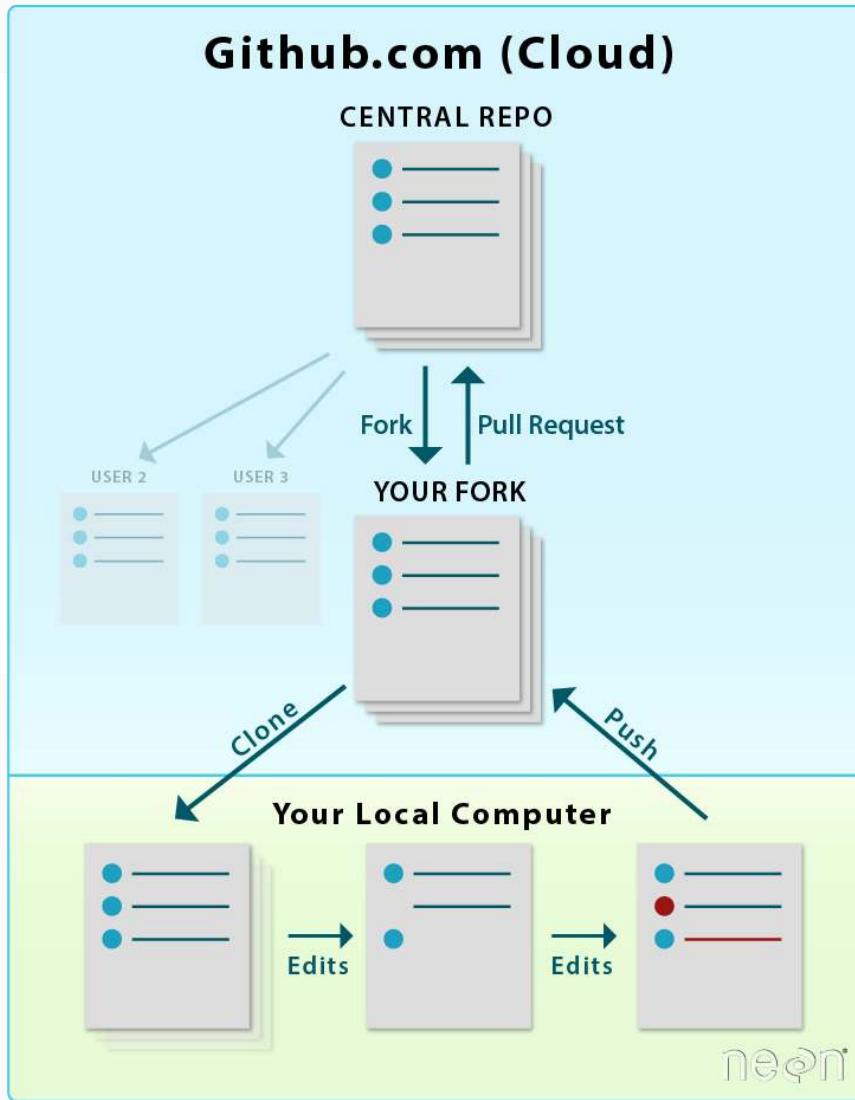
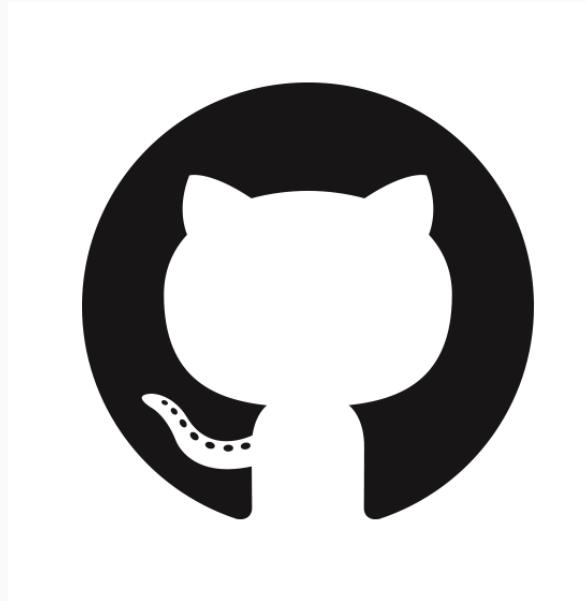
4. Funcionamento do controle de versão



4. Funcionamento do controle de versão

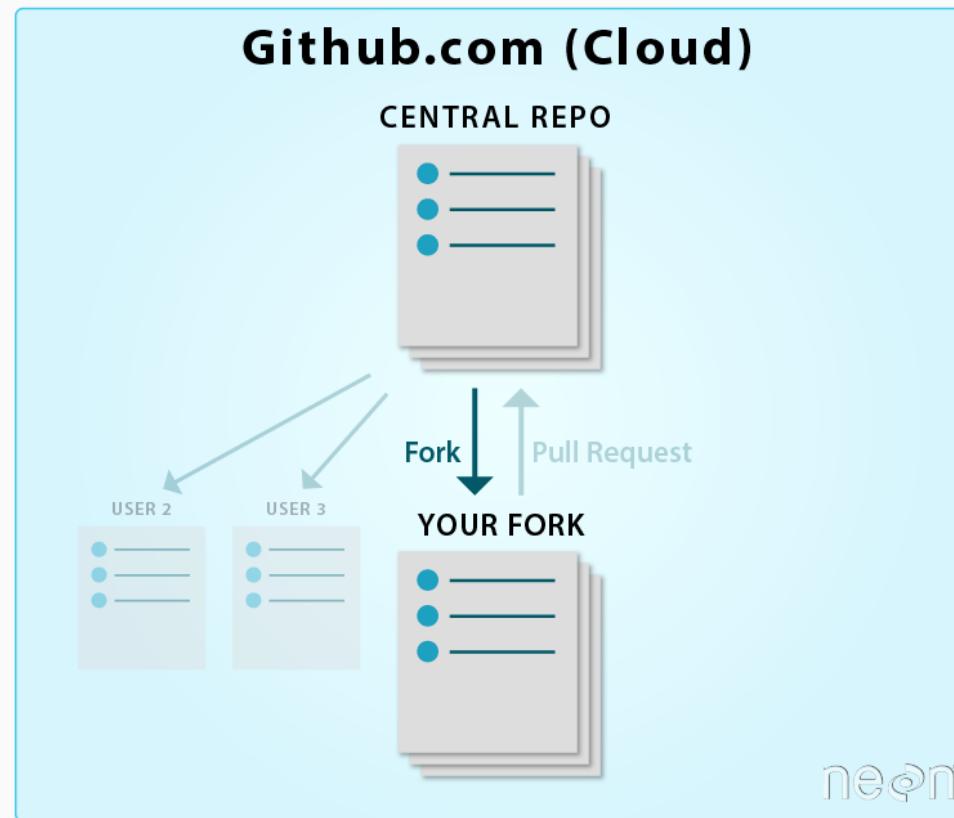
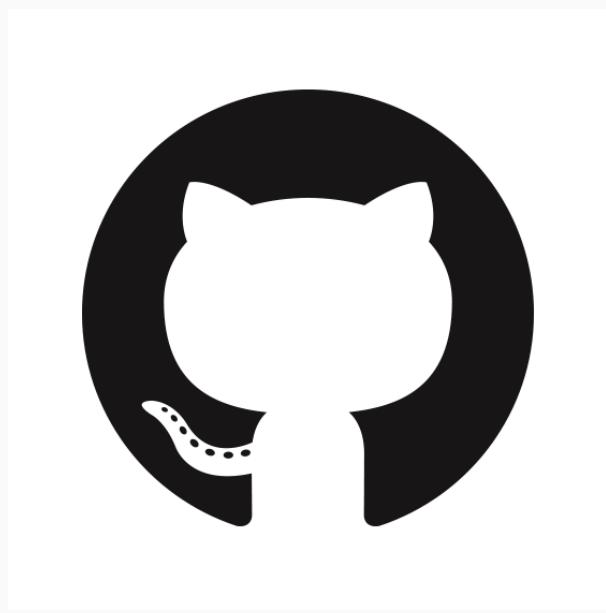


4. Funcionamento do controle de versão



5. GitHub: Fork

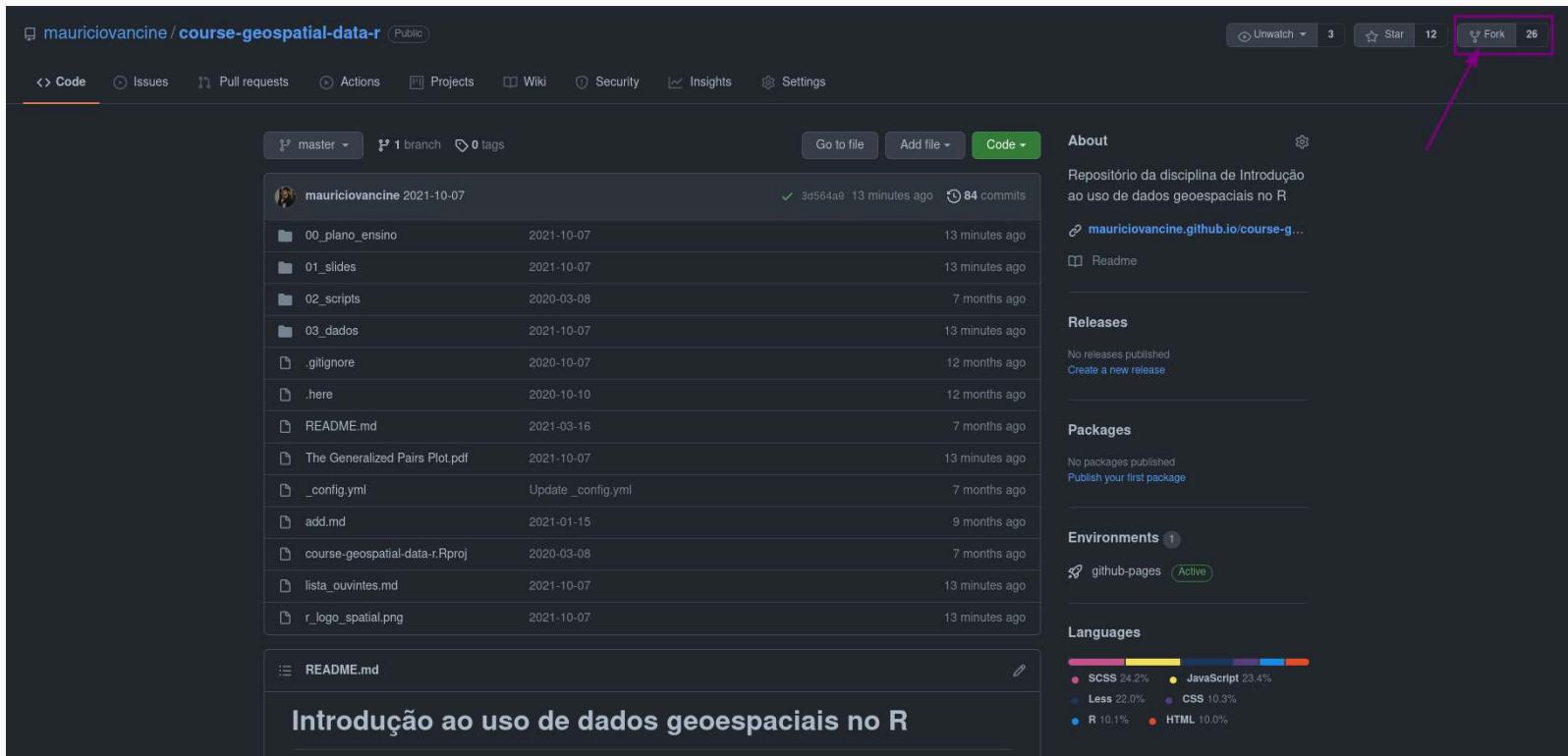
Fork: cópia de um repositório remoto para o seu GitHub



5. GitHub: Fork

Forken o repositório do curso

<https://github.com/mauriciovancine/course-geospatial-data-r>



The screenshot shows the GitHub repository page for 'mauriciovancine/course-geospatial-data-r'. The repository is public and has 84 commits. The 'Code' tab is selected. The 'Fork' button in the top right corner is highlighted with a pink box and arrow. The repository description is 'Repositório da disciplina de Introdução ao uso de dados geoespaciais no R'. It includes sections for About, Releases, Packages, Environments, and Languages.

About
Repositório da disciplina de Introdução ao uso de dados geoespaciais no R
[mauriciovancine.github.io/course-g...](#)

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

Environments 1
[github-pages](#) Active

Languages

Linguagem	Porcentagem
SCSS	24.2%
JavaScript	23.4%
Less	22.0%
CSS	10.3%
R	10.1%
HTML	10.0%

5. GitHub: Fork

Forken o repositório do curso

<https://github.com/mauriciovancine/course-geospatial-data-r>



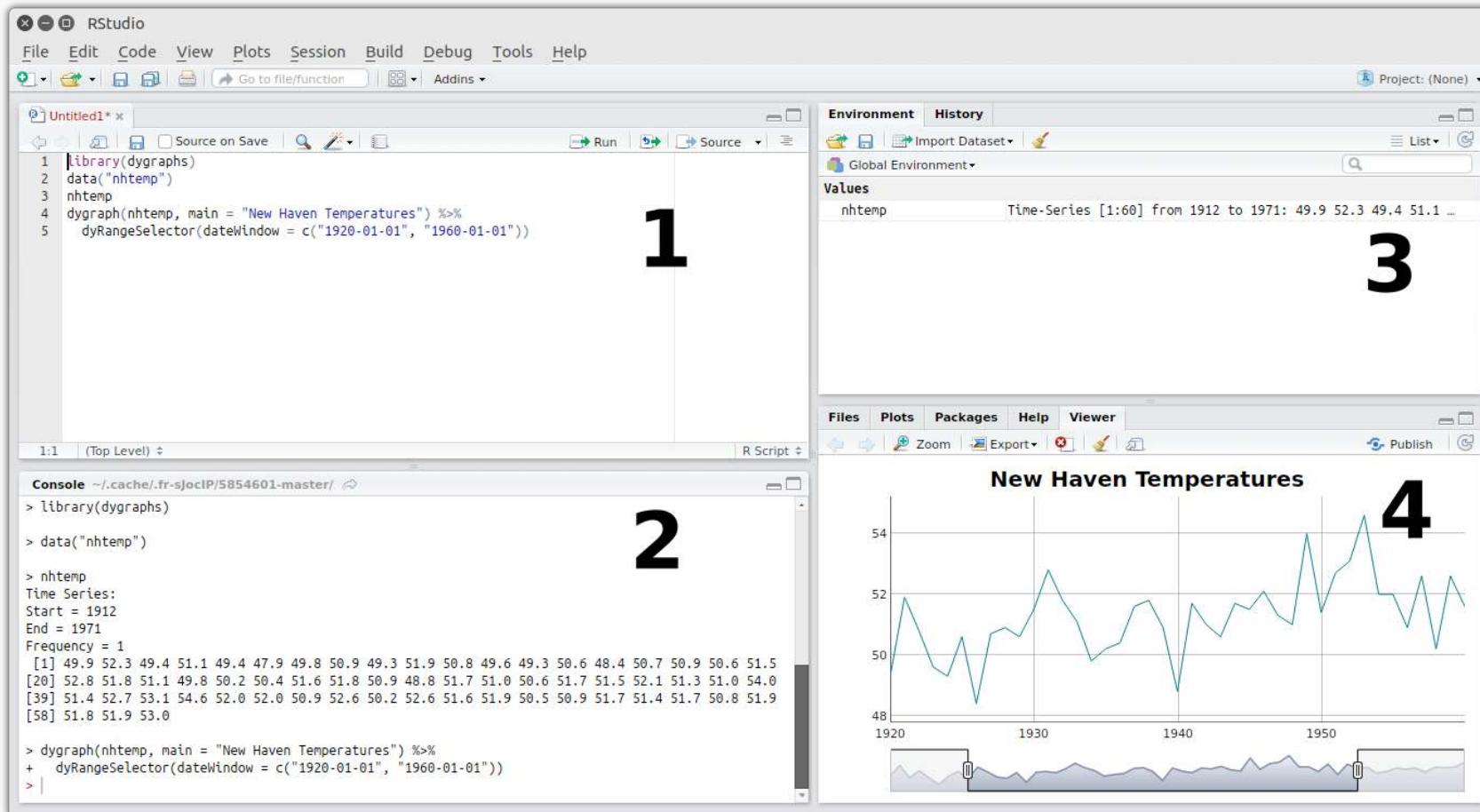
Agora o repositório do curso, que estava no meu GitHub, foi copiado ("garfado") para o GitHub de vocês

Antes de continuar, vamos configurar o RStudio, git e
GitHub

Vamos abrir o RStudio

6. Configuração: RStudio, git, GitHub

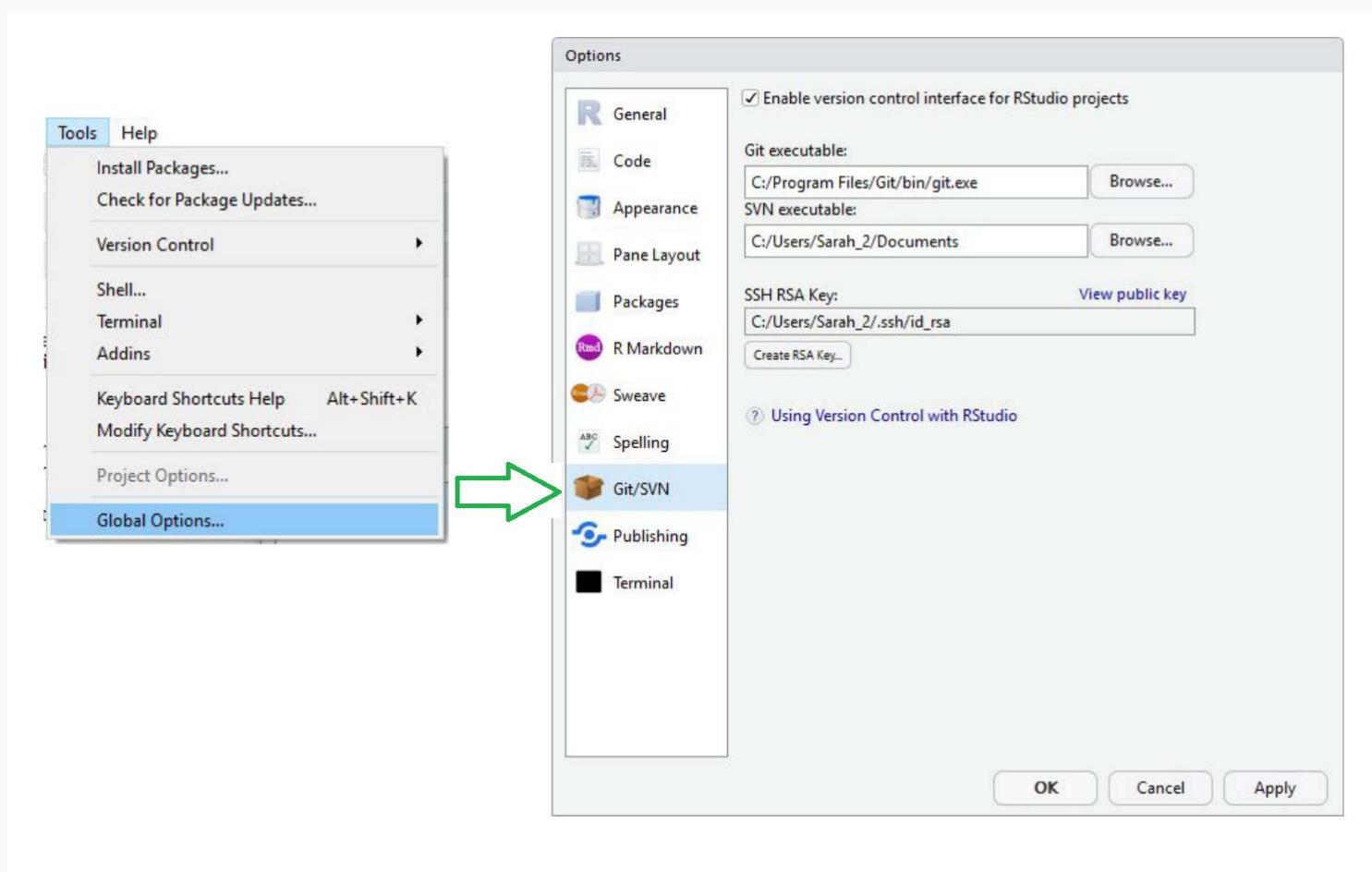
IDE RStudio



6. Configuração: RStudio, git, GitHub

Indicar o caminho do git para o RStudio

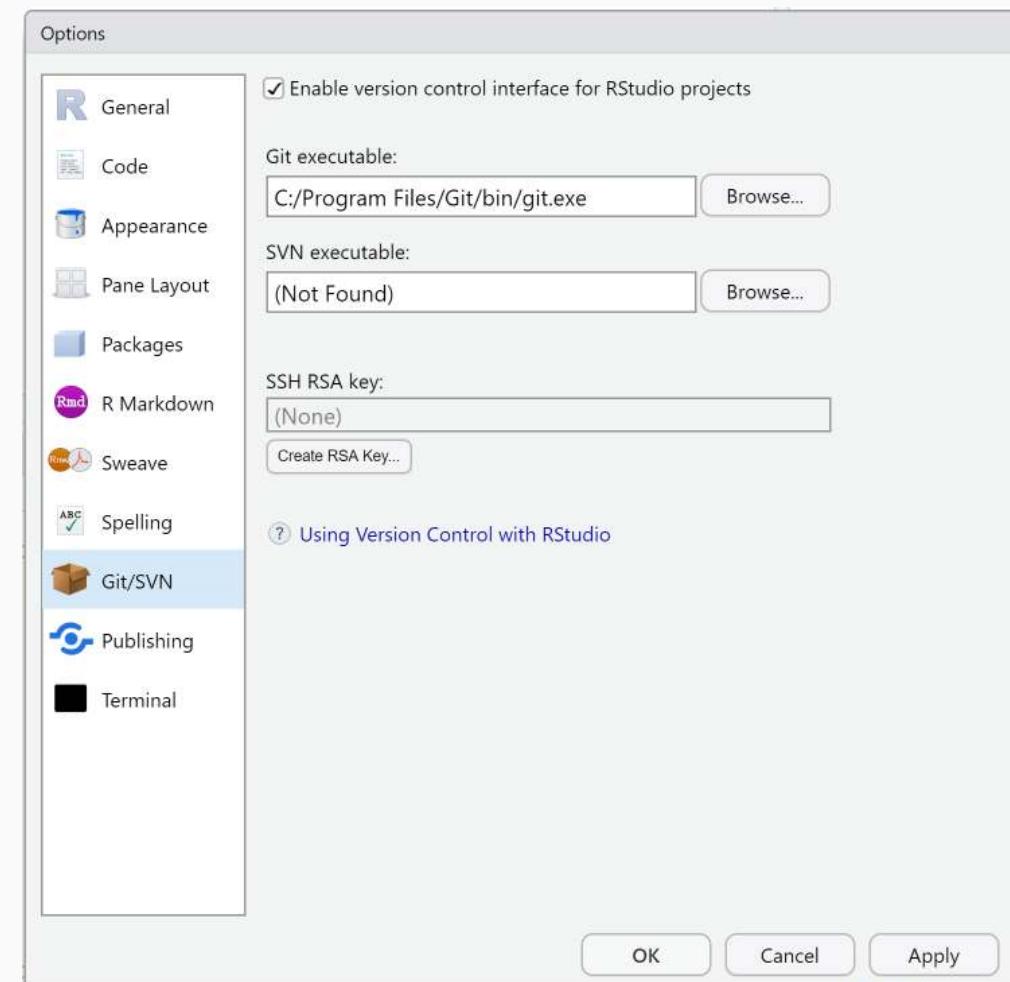
Tools > Global Options > Git/SVN



6. Configuração: RStudio, git, GitHub

Marcar **Enable version control interface...** e em **Git executable**

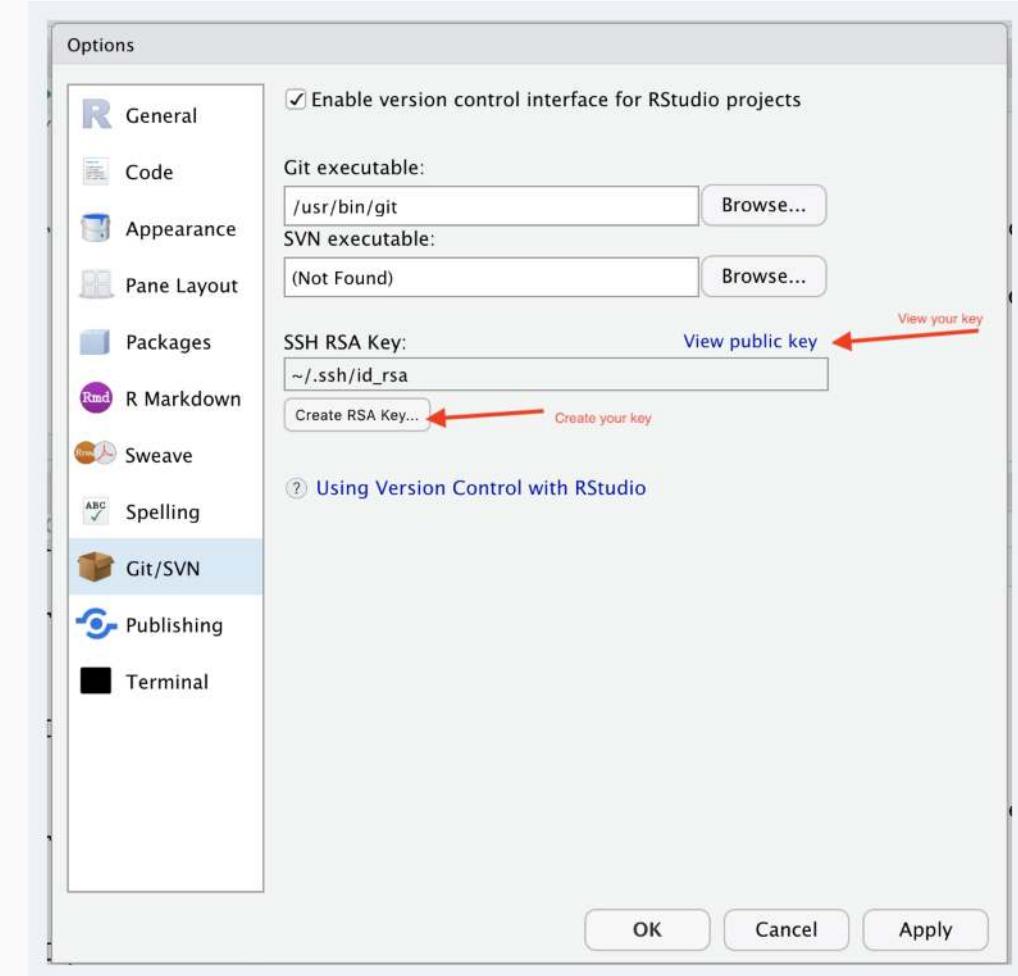
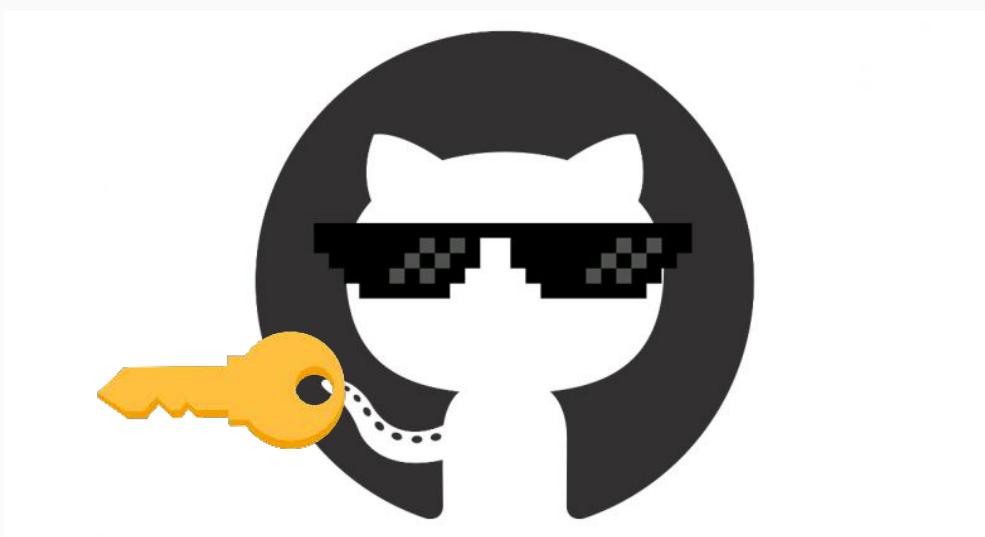
- Windows: C:/Program Files/Git/bin/git.exe
- GNU/Linux: /usr/bin/git
- MacOS: ???



6. Configuração: RStudio, git, GitHub

Criar uma chave SSH

- Create RSA key
- View public key
- Copiar: Ctrl + C

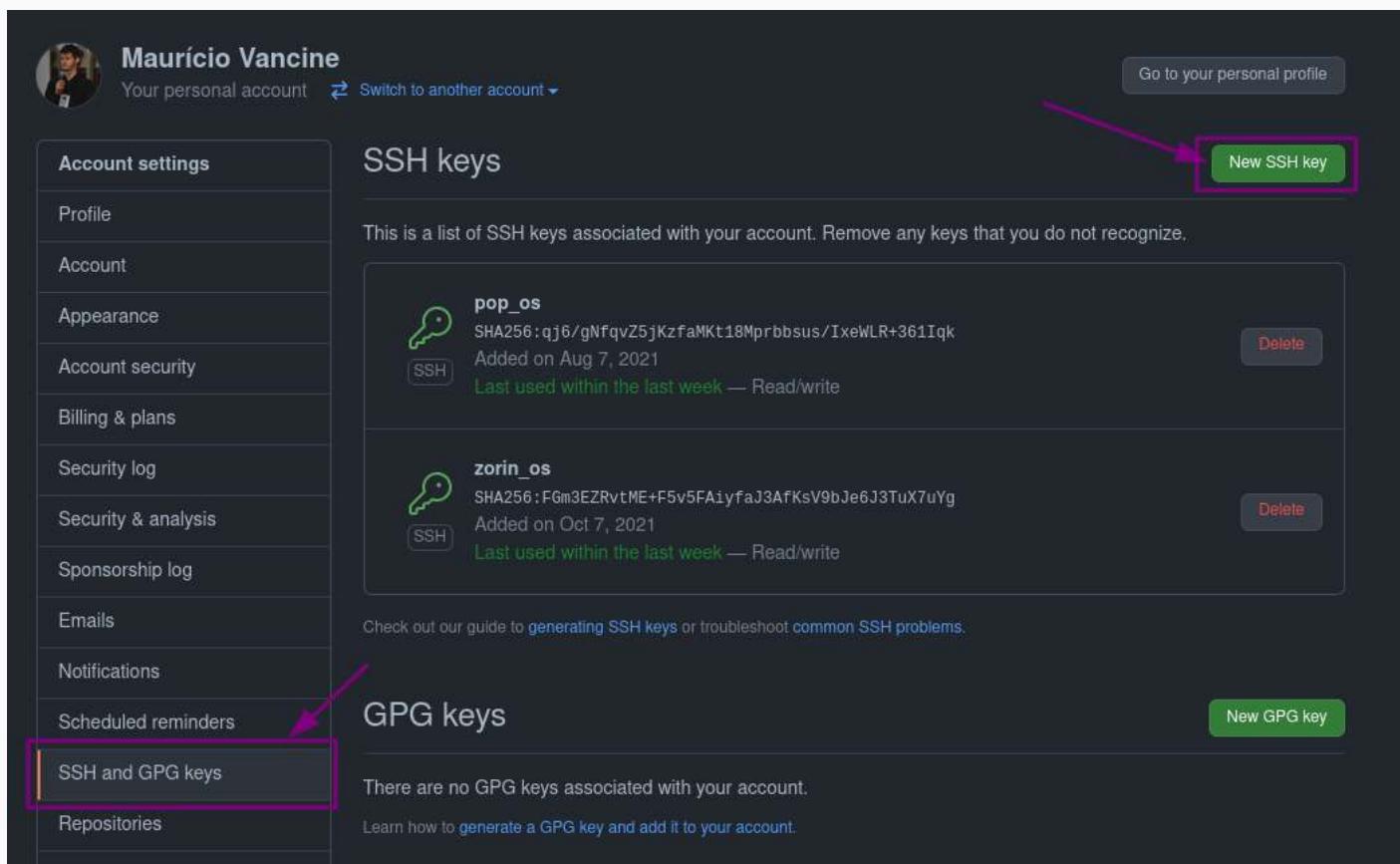


Agora vamos ao GitHub

6. Configuração: RStudio, git, GitHub

No GitHub

Perfil > Settings > SSH and GPG keys > New SSH key



The screenshot shows the GitHub account settings for Maurício Vancine. The left sidebar lists various account settings, with 'SSH and GPG keys' highlighted by a pink rectangle and arrow. The main area displays 'SSH keys' and 'GPG keys'. Under 'SSH keys', two entries are listed: 'pop_os' and 'zorin_os', each with a key icon, SHA256 fingerprint, date added, and a 'Delete' button. Under 'GPG keys', it says 'There are no GPG keys associated with your account.' A green 'New SSH key' button is located at the top right of the SSH keys section.

Maurício Vancine
Your personal account [Switch to another account](#)

Account settings

- Profile
- Account
- Appearance
- Account security
- Billing & plans
- Security log
- Security & analysis
- Sponsorship log
- Emails
- Notifications
- Scheduled reminders
- SSH and GPG keys**
- Repositories

SSH keys

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

Key	Name	SHA256	Date Added	Last used	Action
SSH	pop_os	SHA256:qj6/gNfqvZ5jKzfaMkt18Mprbbus/IxeWLR+361Iqk	Added on Aug 7, 2021	Last used within the last week — Read/write	Delete
SSH	zorin_os	SHA256:FGm3EZRvtME+F5v5FAiyfaJ3AfKsV9bJe6J3TuX7uYg	Added on Oct 7, 2021	Last used within the last week — Read/write	Delete

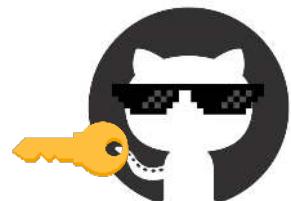
GPG keys

New GPG key

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account.



[Add ao GitHub](#)

6. Configuração: RStudio, git, GitHub

No GitHub

- Title: um nome qualquer
- Key: colar (Ctrl + V)
- Add SSH key



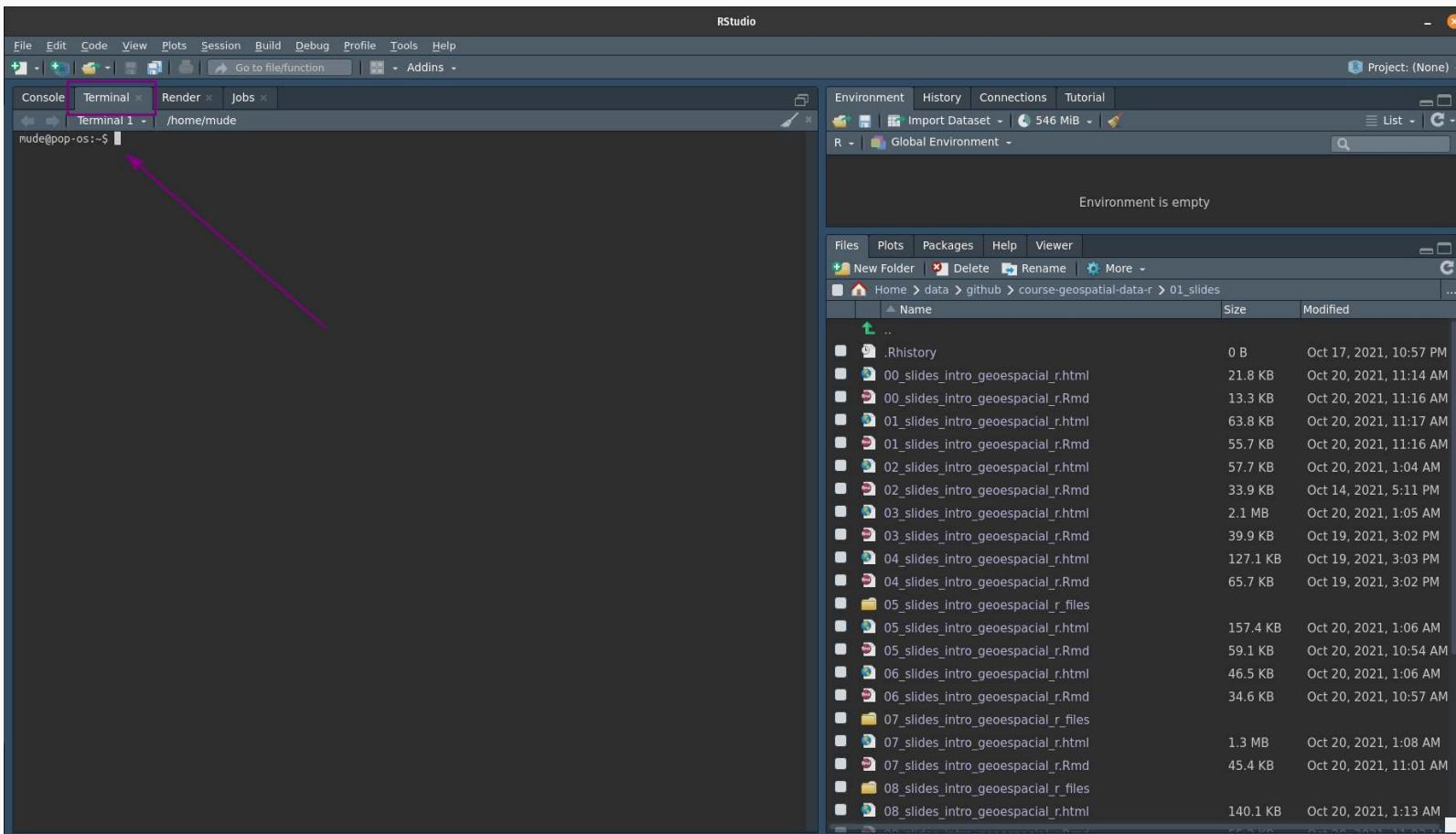
The screenshot shows the GitHub account settings page for Maurício Vancine. The left sidebar has a dark theme with white text. The 'SSH and GPG keys' option is highlighted with a red border. The main area is titled 'SSH keys / Add new'. It contains fields for 'Title' (with a placeholder 'My First Key') and 'Key' (containing a long string of characters). A green button labeled 'Add SSH key' is at the bottom. A pink arrow points from the 'Add SSH key' button to the 'Add SSH key' item in the list above.

[Add ao GitHub](#)

Antes de avançarmos, vamos configurar o git

6. Configuração: RStudio, git, GitHub

Terminal

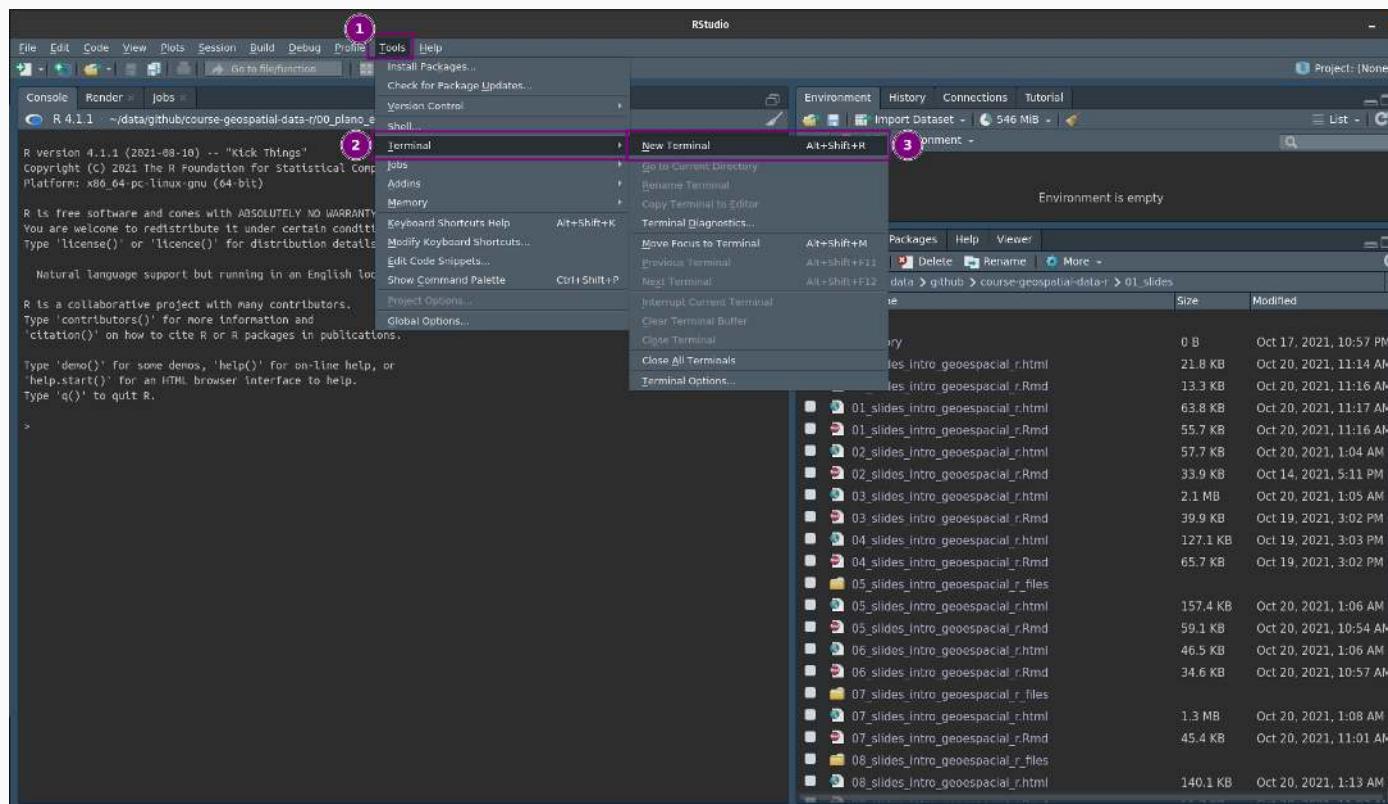


6. Configuração: RStudio, git, GitHub

Terminal

Tools > Terminal > New Terminal

Atalho: Alt + Shift + R



6. Configuração: RStudio, git, GitHub

git config: definir as configurações de usuário do git

```
# config
git config --list
git config --global user.name "seu nome"
git config --global user.email "email@dominio.com"
git config --list

# ssh
ssh-keyscan -t rsa github.com >> ~/.ssh/known_hosts
```



Git Config

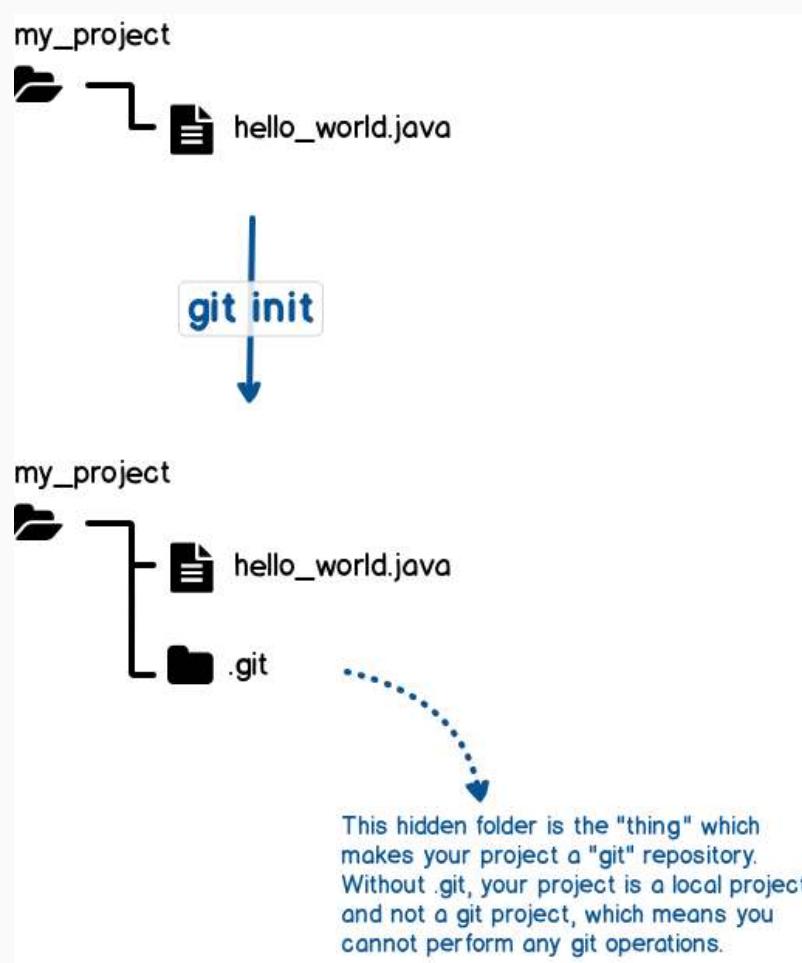
Tudo certo até aqui?

Agora podemos iniciar o versionamento

1. Podemos começar pelo repositório local

7. Iniciando localmente: git init

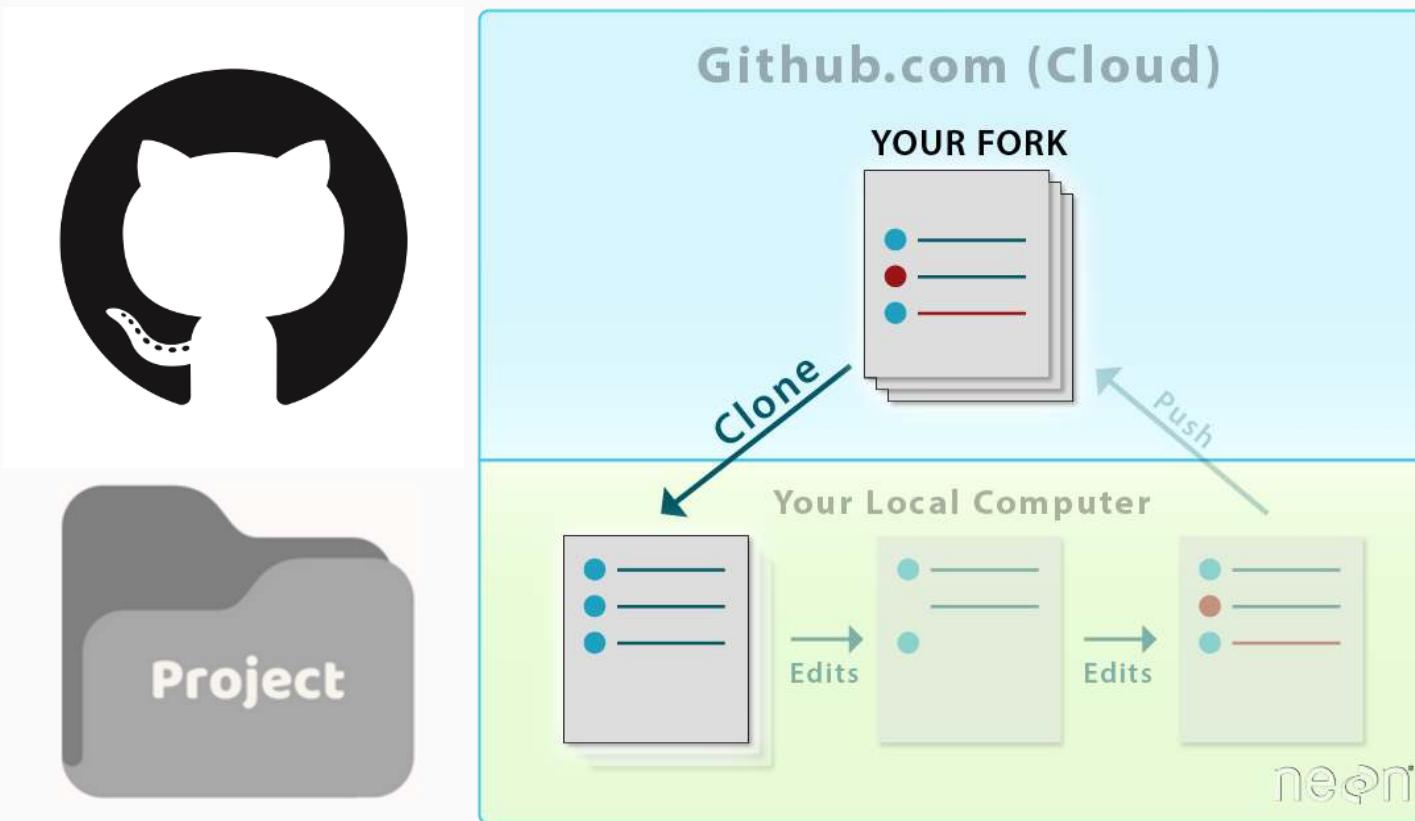
git init: inicia o versionamento de um repositório local



2. Podemos começar pelo repositório remoto

8. Iniciando remotamente: git clone

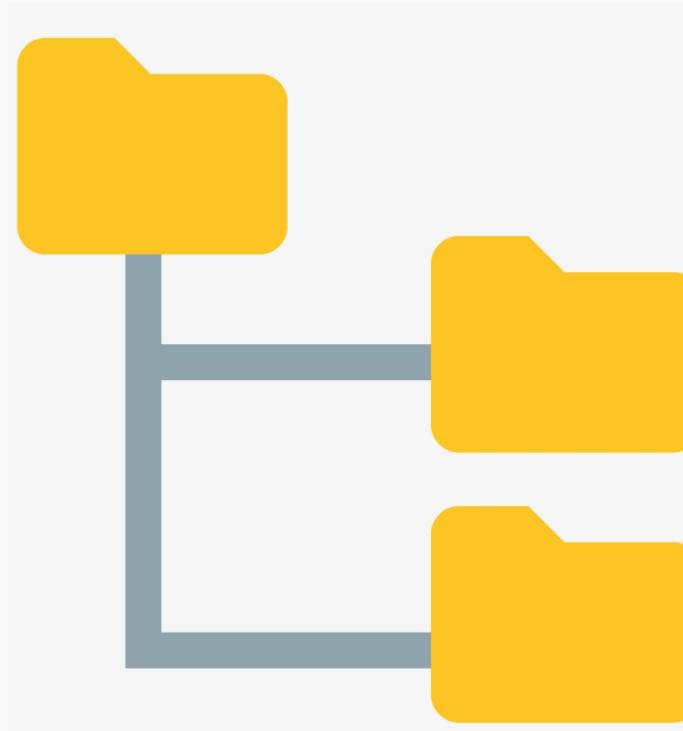
git clone: faz o download de um repositório remoto (e.g. GitHub) para o seu computador (repositório local)



8. Iniciando remotamente: git clone

Primeiro, vamos **criar um diretório** chamado `github`

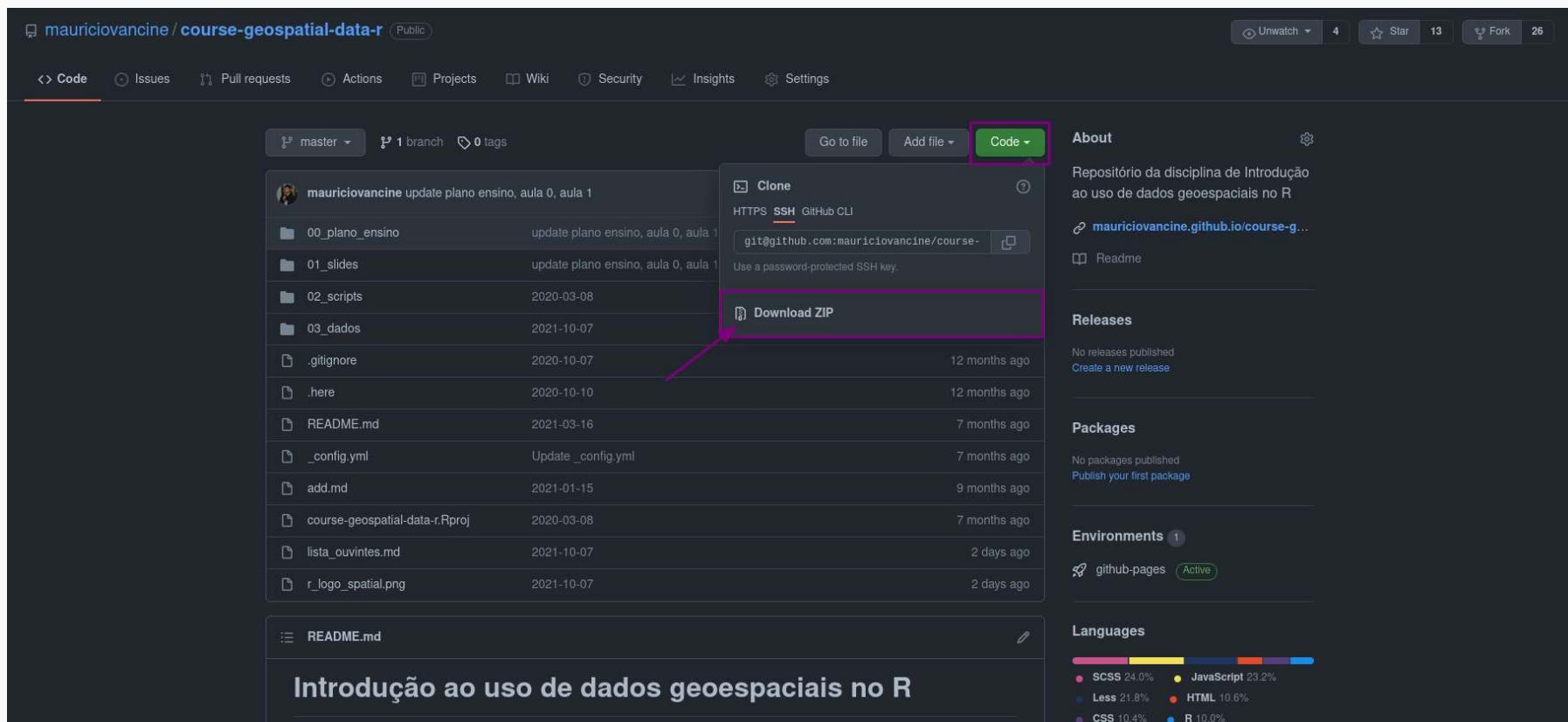
```
".          # raiz"  
"   ┌── home/      # home"  
"   │   ┌── data/    # dados"  
"   │   └── github/  # todos os repositorios"
```



E como podemos fazer o clone?

8. Iniciando remotamente: git clone

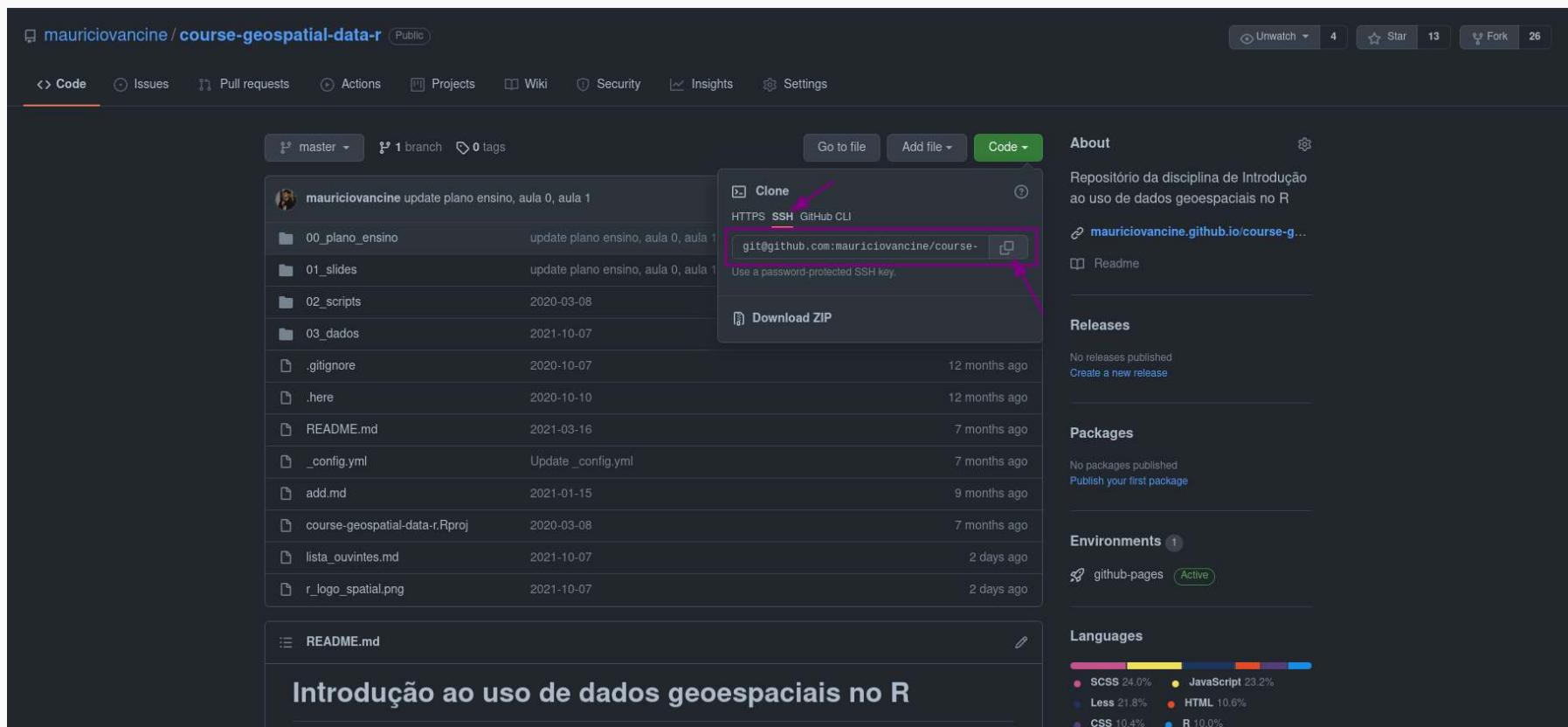
1. Download diretamente do repositório remoto no formato .zip



8. Iniciando remotamente: git clone

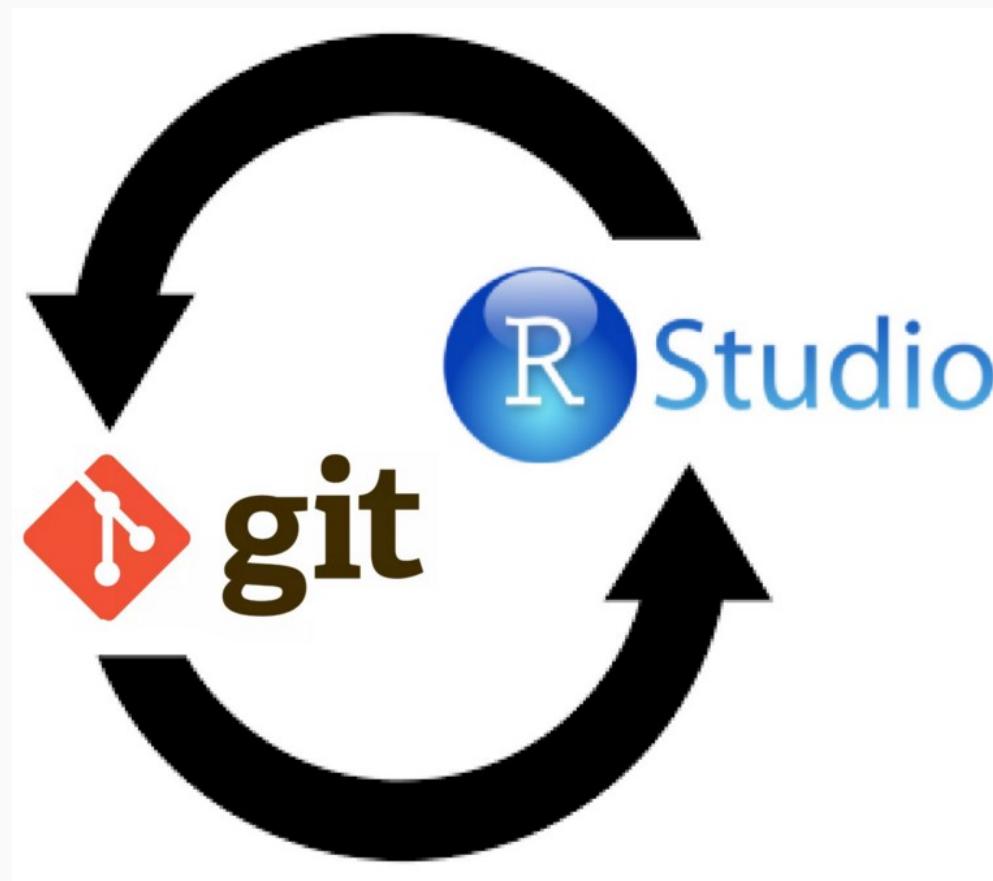
2. Usando o **terminal** para clonar pelo **SSH** (configurado anteriormente)

```
# terminal  
git clone git@github.com:mauriciovancine/course-geospatial-data-r.git
```



8. Iniciando remotamente: git clone

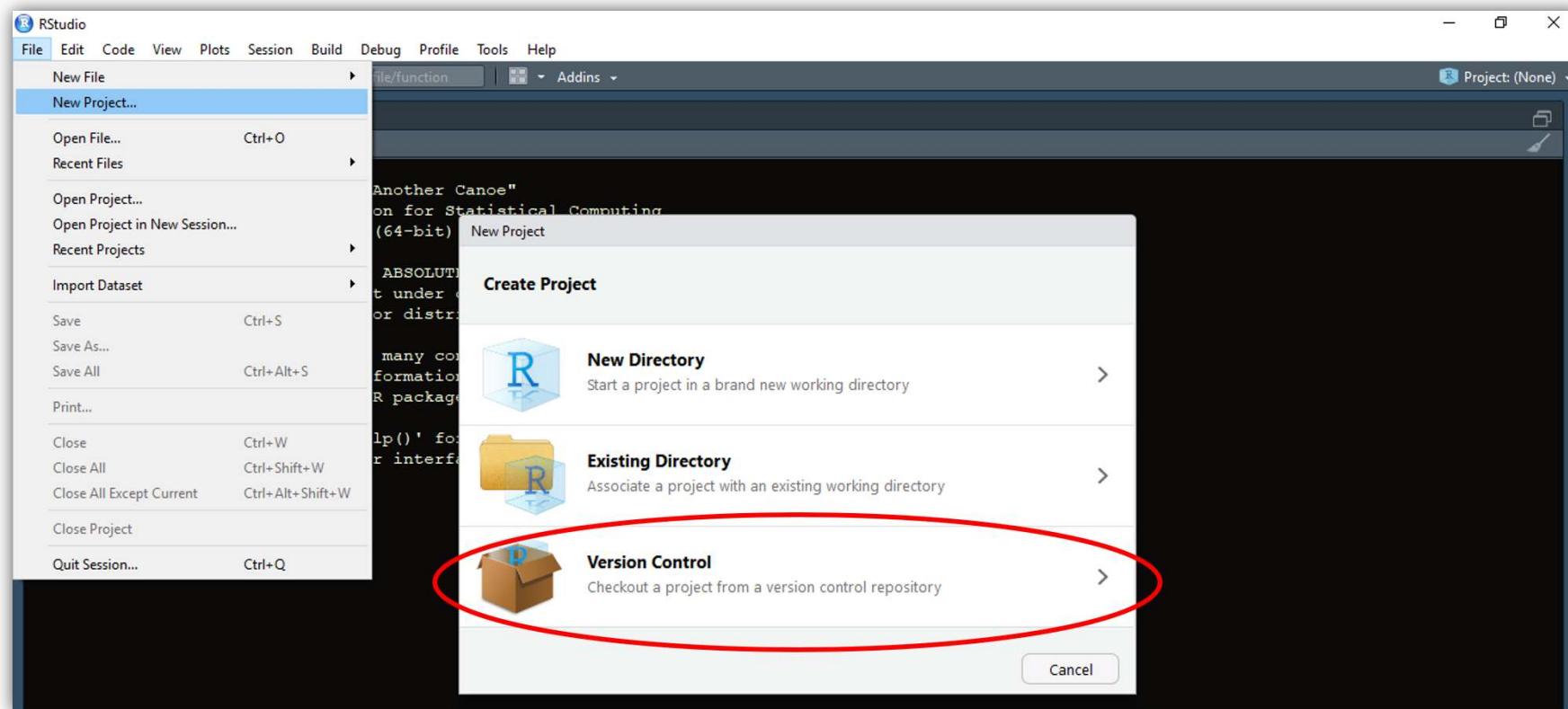
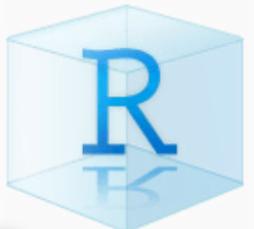
3. Usando o **RStudio** para clonar pelo **SSH** (configurado anteriormente)



8. Iniciando remotamente: git clone

Criar um Projeto R com controle de versão

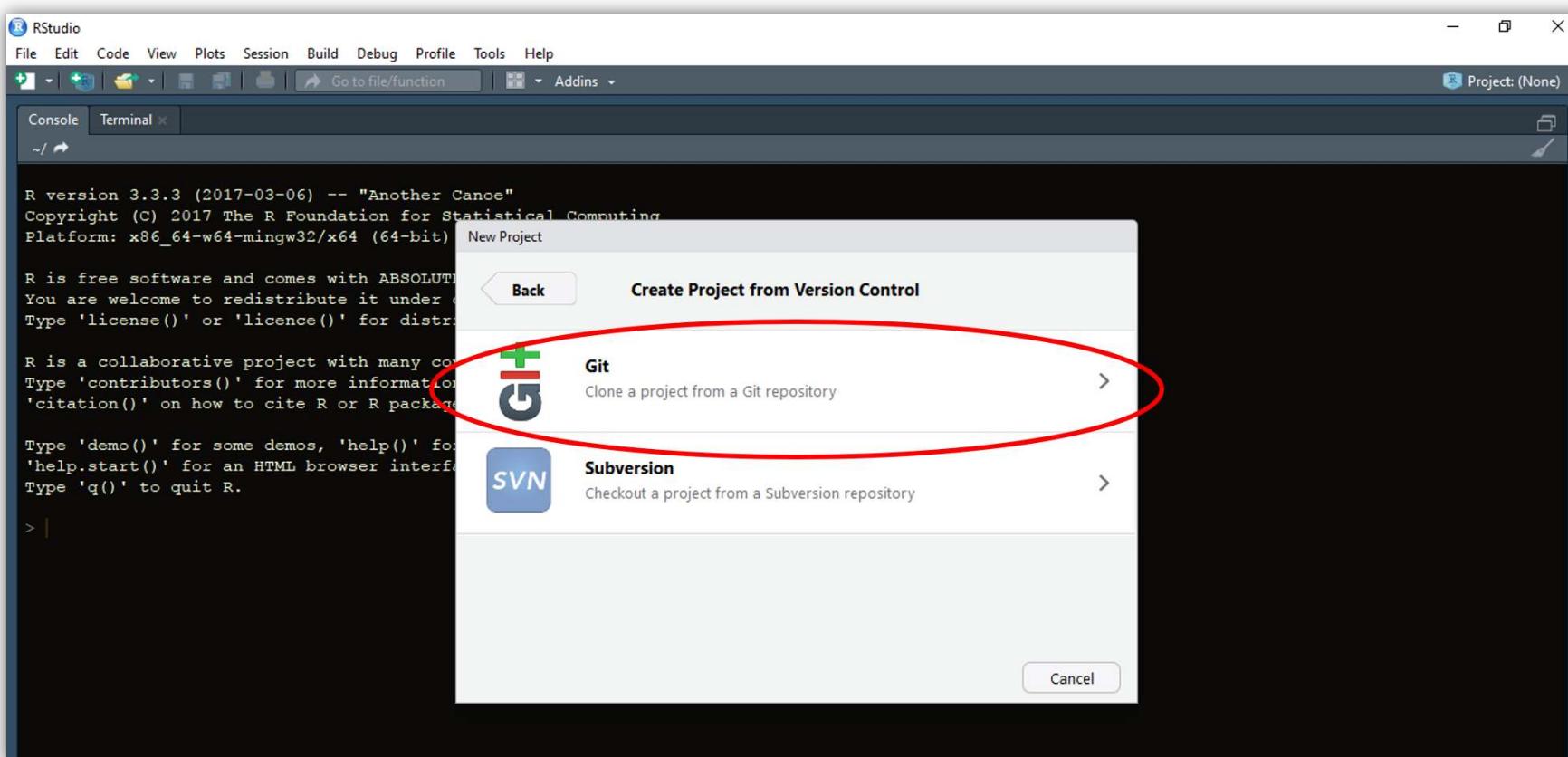
File > New Project > Version Control



8. Iniciando remotamente: git clone

Escolher **clonar repositório** do GitHub

Git



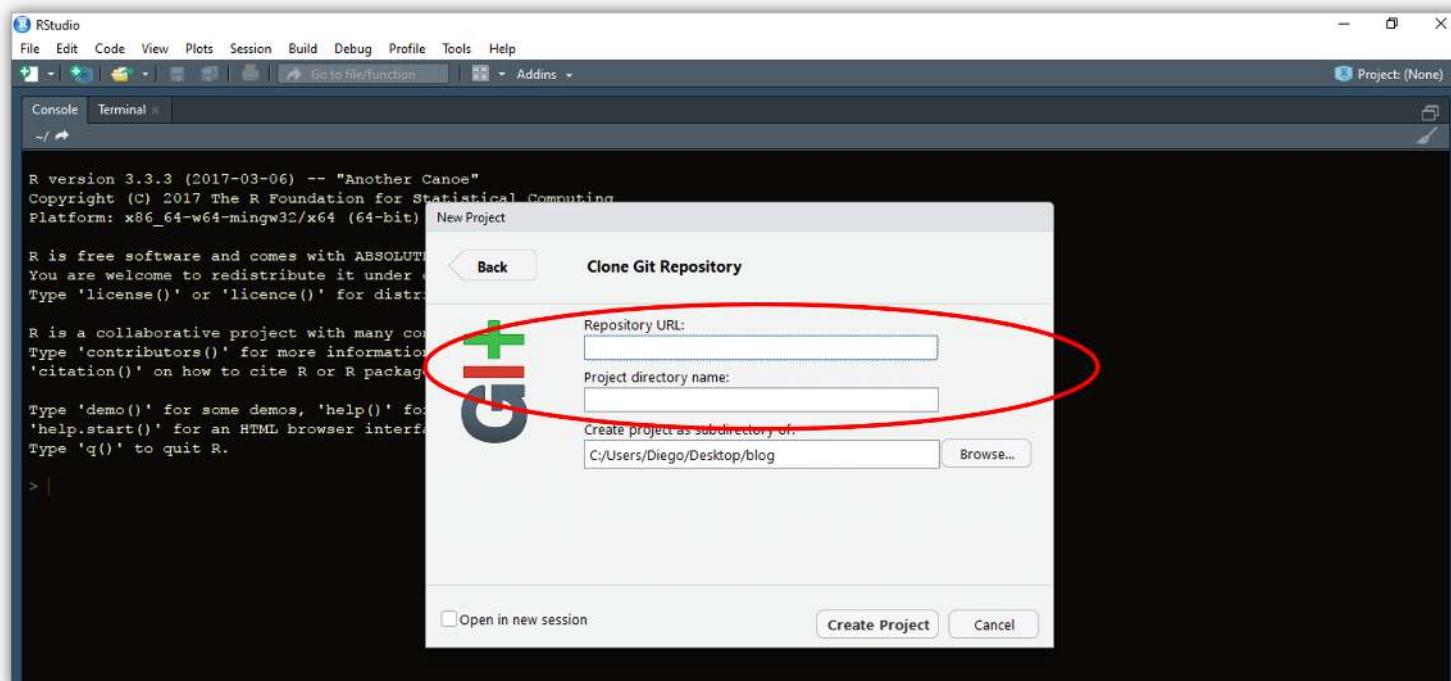
8. Iniciando remotamente: git clone

Preencher:

Repository URL: `git@github.com:mauriciovancine/course-geospatial-data-r.git`

Project directory name: `course-geospatial-data-r`

Create project as subdirectory of: `/home/mude/data/github`

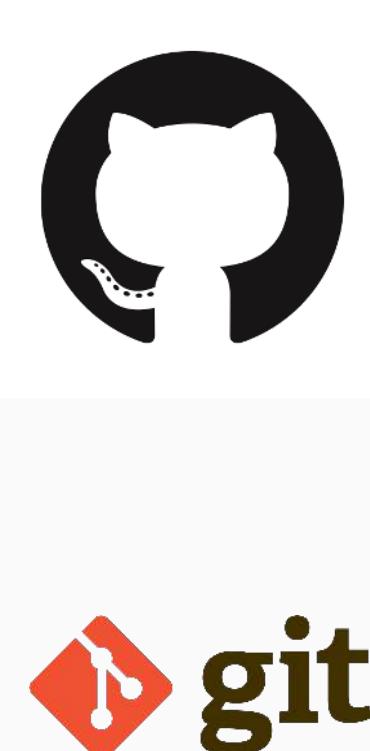
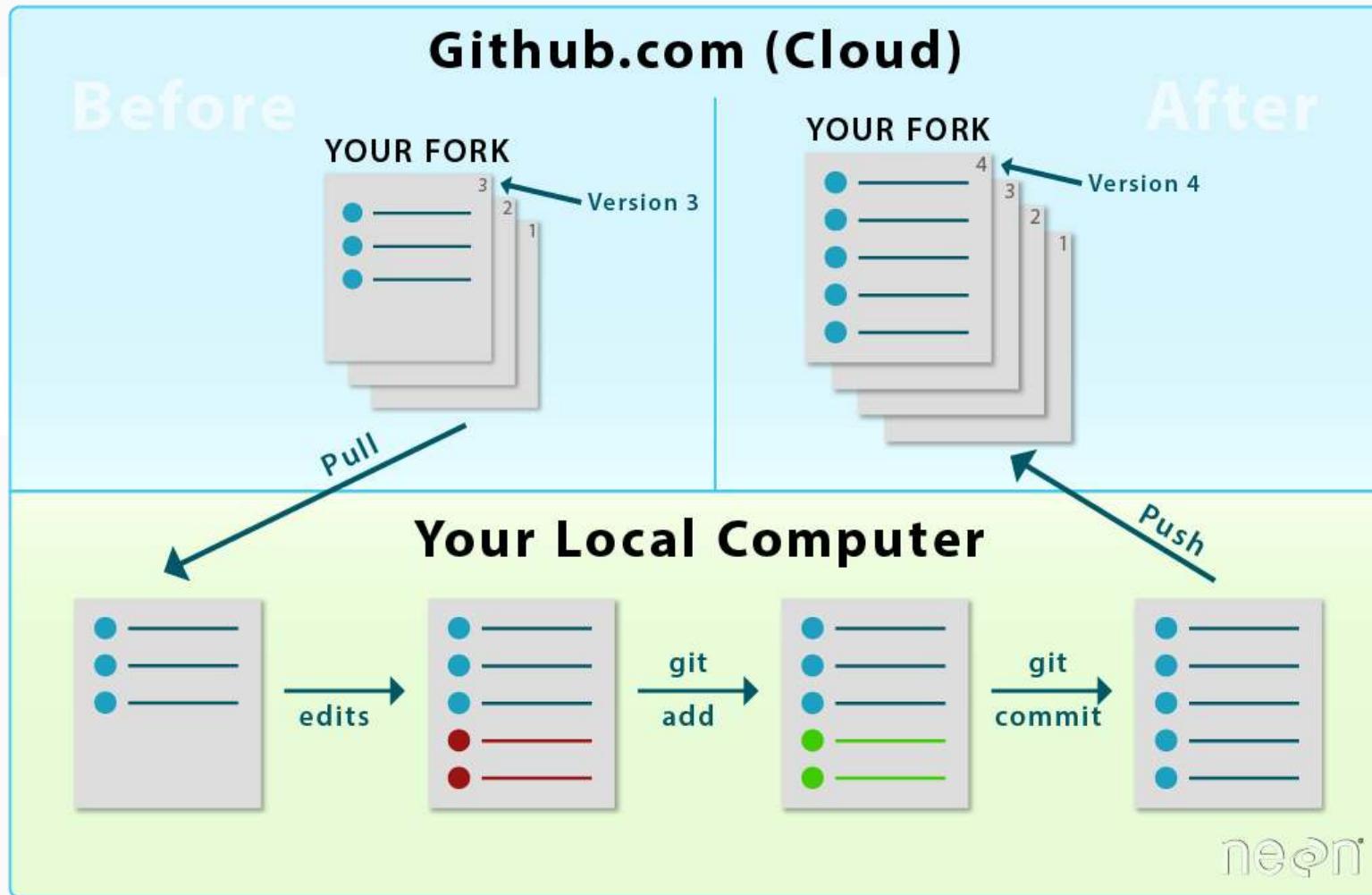
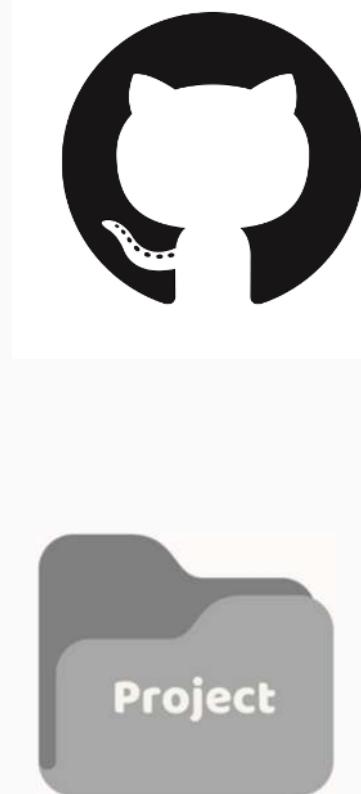


Se uma janela aparecer, basta digitar "yes" para criar
uma autentificação

Vamos aguardar o download...

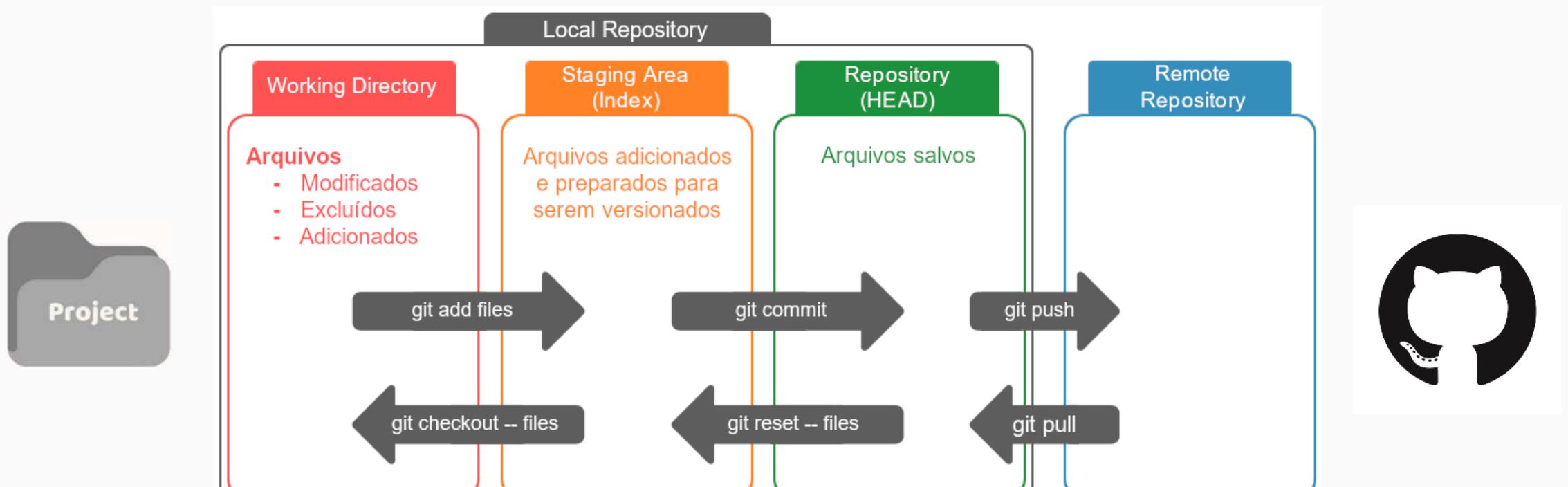
Agora sim, vamos começar o versionamento
propriamente dito

9. Versionamento: git add, git commit e git status



9. Versionamento: git add, git commit e git status

Comandos git para o manejo do repositório local e remoto



Agora prestem bastante atenção, porque complica...

9. Versionamento: git add, git commit e git status

Estados de um arquivo no diretório

1. Modificados
2. Deletados
3. Adicionados



9. Versionamento: git add, git commit e git status

Estados dos arquivos para o git

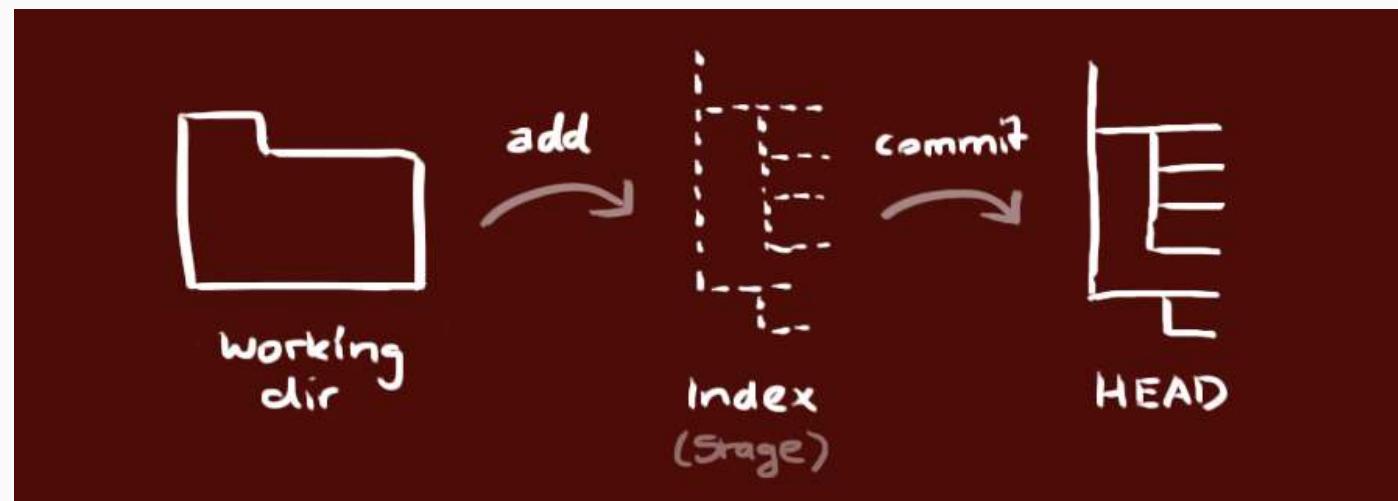
1. **Modificado (modified)**: qualquer arquivo modificado, removido ou adicionado, mas ainda não informado que fará parte da versão consolidada, ou seja, ainda não preparado (*working directory*)
2. **Preparado (staged)**: arquivos adicionados e preparados para serem consolidados na versão final (*staged area*)
3. **Consolidado (committed)**: criação da versão consolidada de todos os arquivos que estavam no *staged* (não considera os que estão no *modified*) (*.git*)



9. Versionamento: git add, git commit e git status

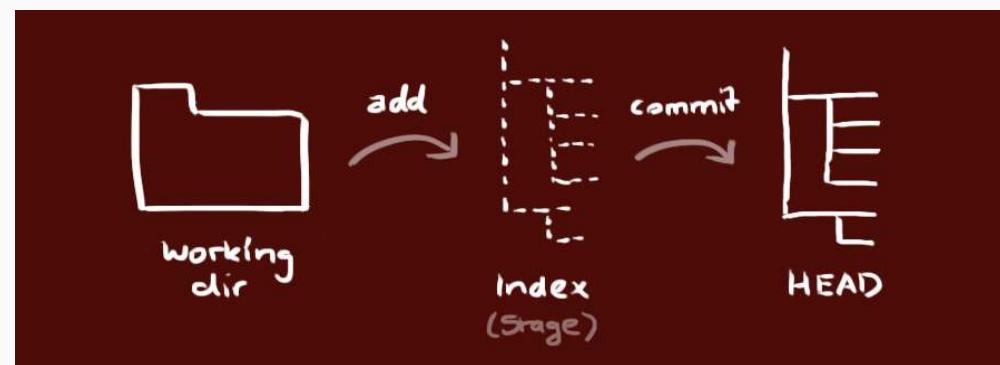
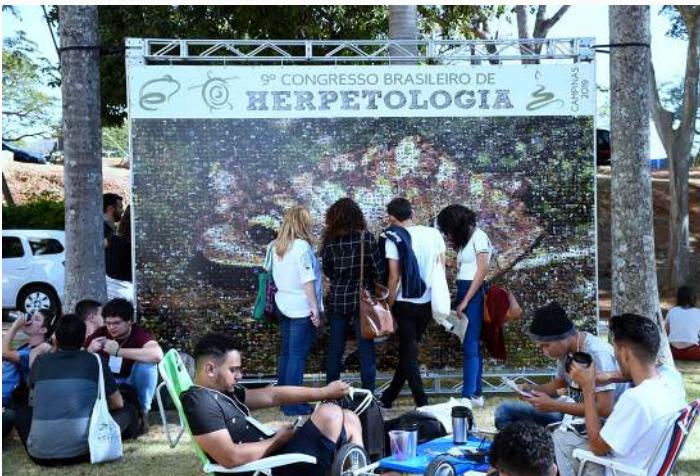
Estados dos arquivos para o git

1. **Modificado (modified)**: qualquer arquivo modificado, removido ou adicionado, mas ainda não informado que fará parte da versão consolidada, ou seja, ainda não preparado (*working directory*)
2. **Preparado (staged)**: arquivos adicionados e preparados para serem consolidados na versão final (*staged area*)
3. **Consolidado (committed)**: criação da versão consolidada de todos os arquivos que estavam no *staged* (não considera os que estão no *modified*) (*.git*)



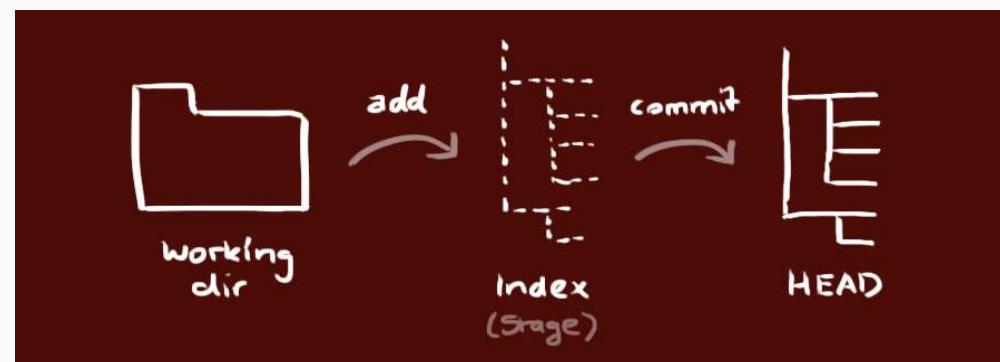
9. Versionamento: git add, git commit e git status

Estados dos arquivos para o git



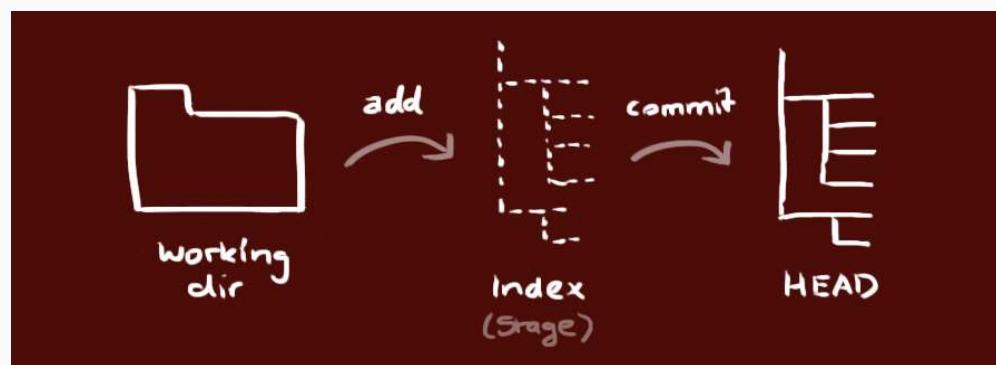
9. Versionamento: git add, git commit e git status

Estados dos arquivos para o git



9. Versionamento: git add, git commit e git status

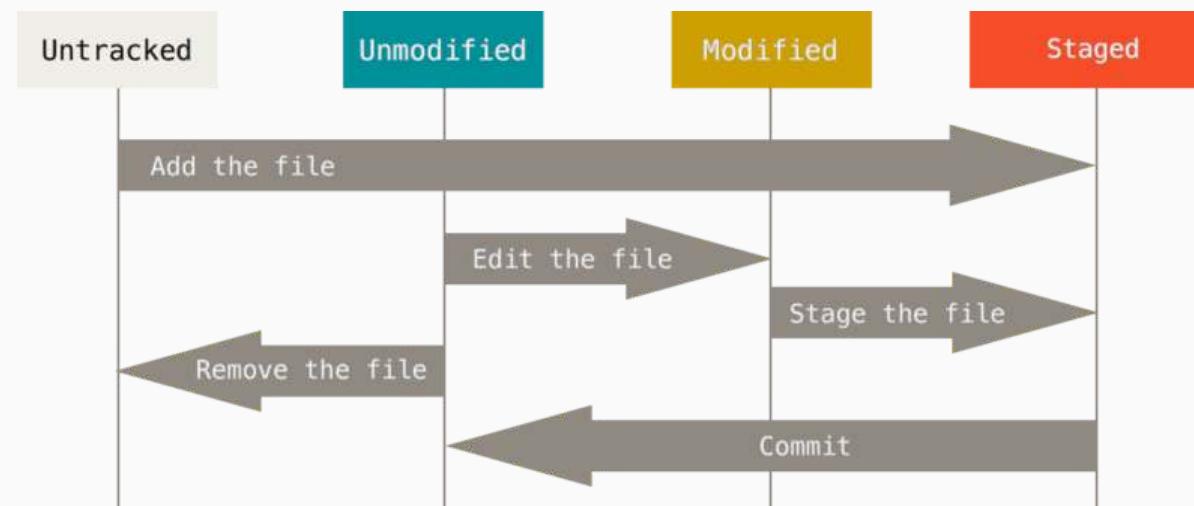
Estados dos arquivos para o git



9. Versionamento: git add, git commit e git status

Estados do repositório local do git

1. **Untracked**: não registrou os arquivos adicionados ou deletados do diretório local
2. **Unmodified**: marcou os arquivos como não-modificados após o commit
3. **Modified**: marcou os arquivos modificados depois da edição dos arquivos
4. **Staged**: marcou os arquivos adicionados ou modificados para ir para o commit

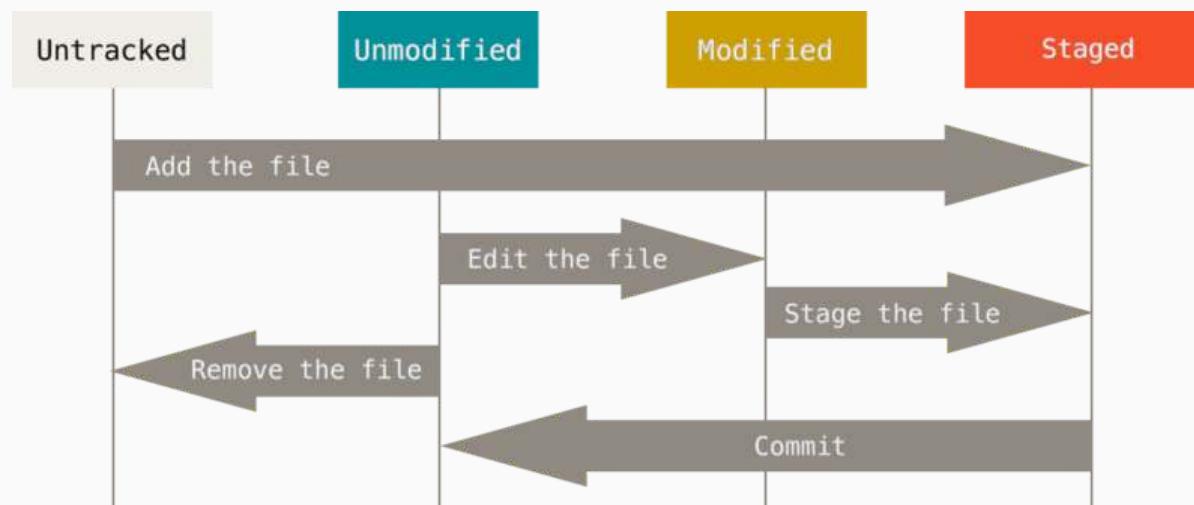


9. Versionamento: git add, git commit e git status

git status: mostra o estado do repositório

```
# status  
git status
```

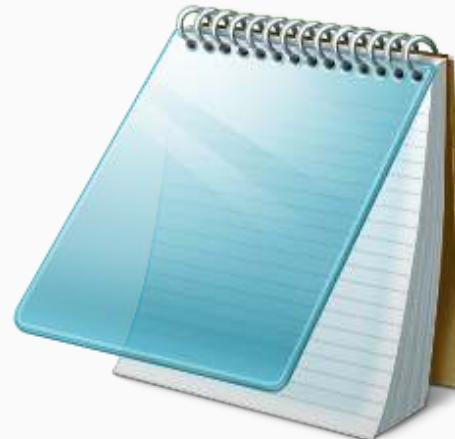
```
On branch master  
No commits yet  
nothing to commit (create/copy files and use "git add" to track)
```



Vamos criar um arquivo no diretório

Usando a aba **Terminal** do RStudio

```
# criar um arquivo  
> teste.txt
```

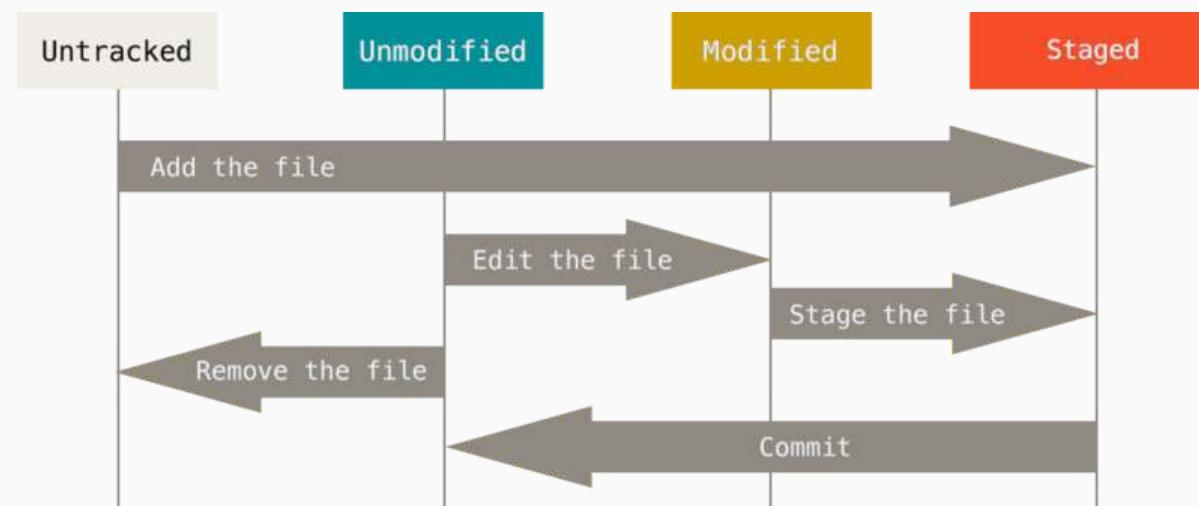


9. Versionamento: git add, git commit e git status

git status: mostra o estado do repositório

```
git status
```

```
On branch master
No commits yet
Untracked files:
  (use "git add <file> ..." to include in what will be committed)
    teste.txt
nothing added to commit but untracked files present (use "git add" to track)
```

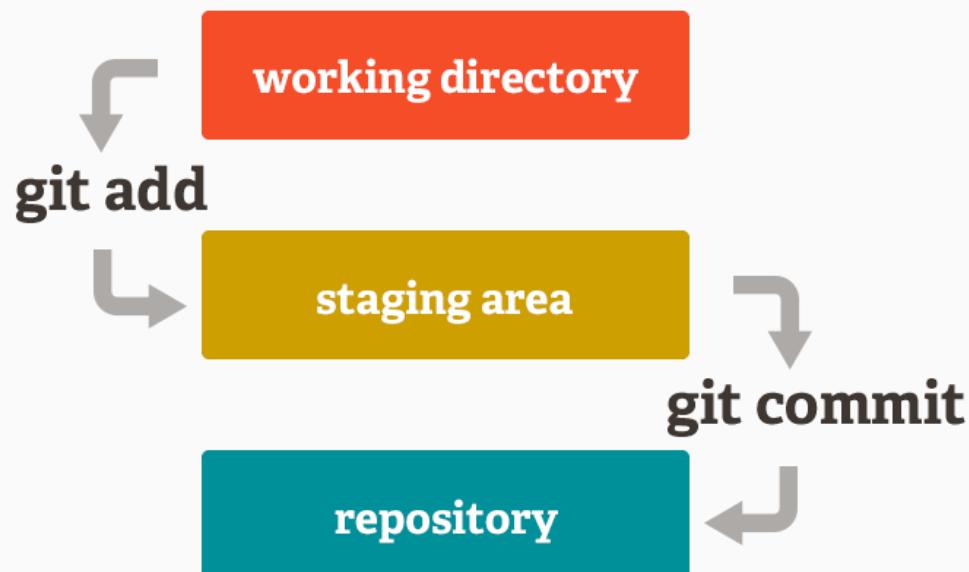


9. Versionamento: git add, git commit e git status

git add: adiciona mudanças após edições (*staging area*)

```
# adicionar ao staging area  
git add -Av
```

```
add 'teste.txt'
```

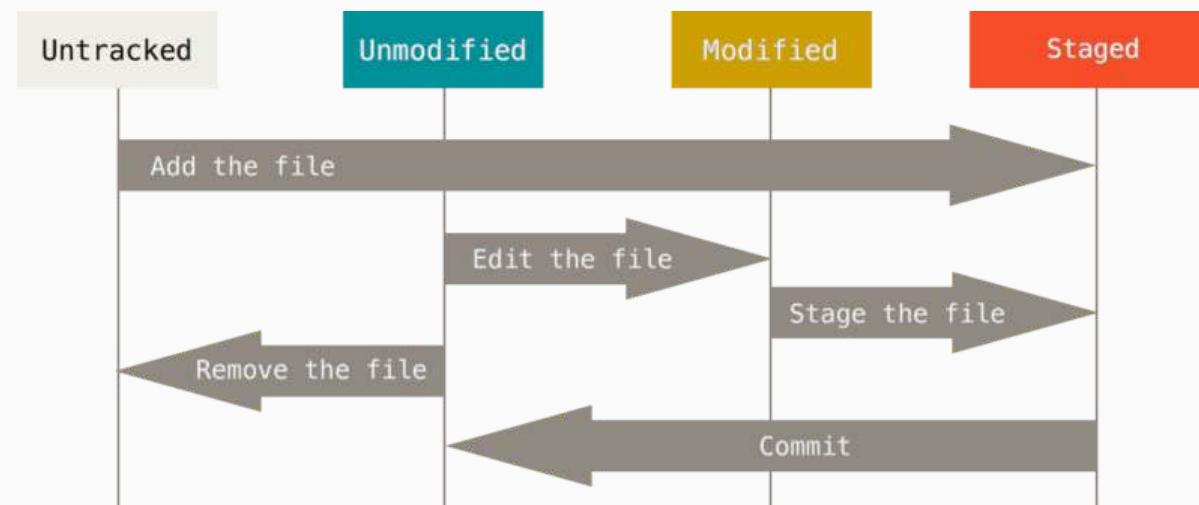


9. Versionamento: git add, git commit e git status

git status: mostra o estado do repositório

```
git status
```

```
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file> ..." to unstage)
    new file: teste.txt
```

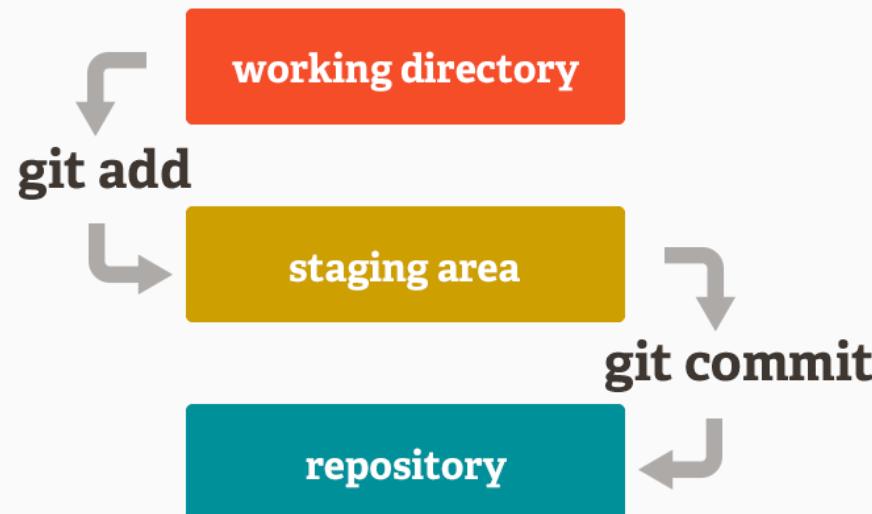


9. Versionamento: git add, git commit e git status

git commit: armazena as mudanças com uma descrição, criando nova versão do repositório (*repository*)

```
git commit -m "add teste.txt"
```

```
[master (root-commit) 8b33bc5] add teste.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 teste.txt
```

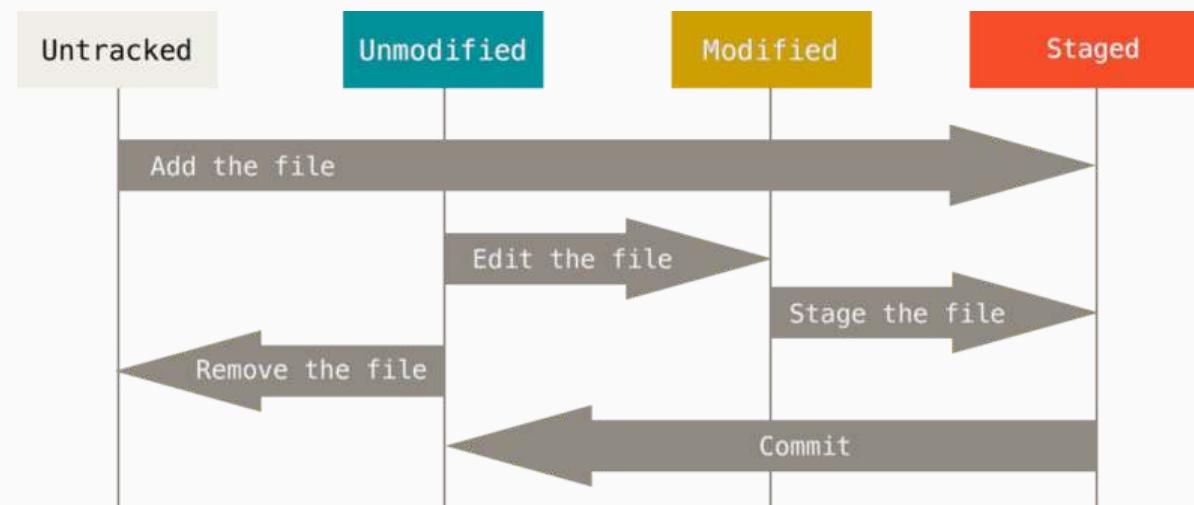


9. Versionamento: git add, git commit e git status

git status: mostra o estado do repositório

```
git status
```

```
On branch master  
nothing to commit, working tree clean
```



9. Versionamento: git add, git commit e git status

Usando a aba **Files** do RStudio, vamos abrir e editar o arquivo **teste.txt**

Acabo de inserir uma edição ao meu arquivo

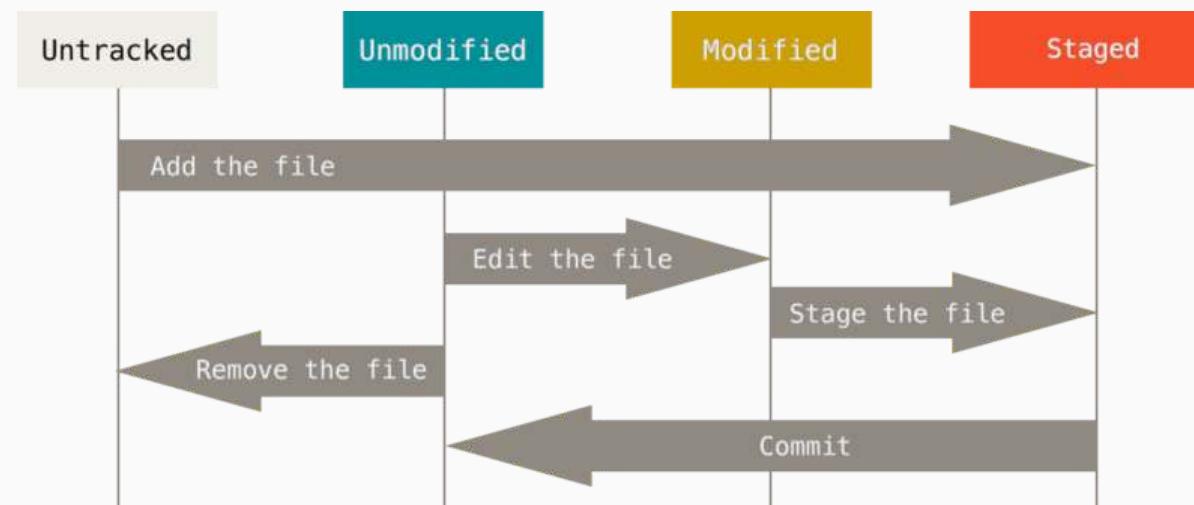


9. Versionamento: git add, git commit e git status

git status: mostra o estado do repositório

```
git status
```

```
Changes not staged for commit:  
(use "git add <file> ..." to update what will be committed)  
(use "git restore <file> ..." to discard changes in working directory)  
modified: teste.txt  
no changes added to commit (use "git add" and/or "git commit -a")
```

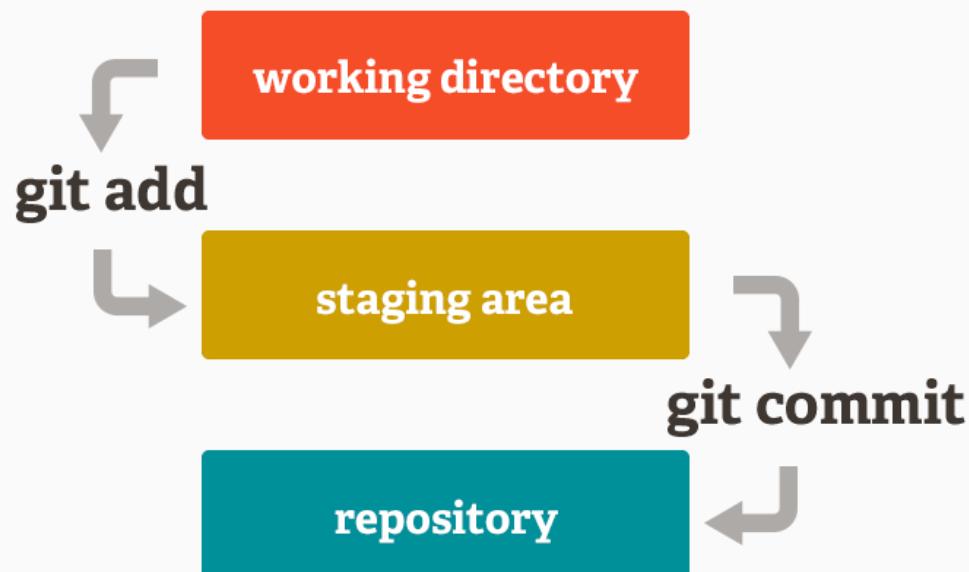


9. Versionamento: git add, git commit e git status

git add: adiciona mudanças após edições (*staging area*)

```
# terminal  
git add -Av
```

```
add 'teste.txt'
```

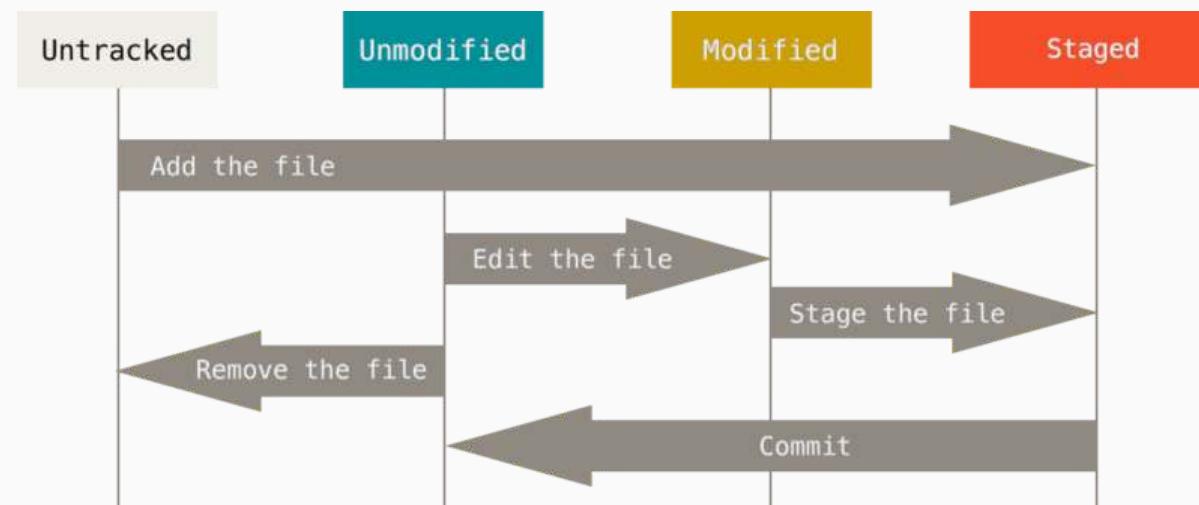


9. Versionamento: git add, git commit e git status

git status: mostra o estado do repositório

```
git status
```

```
On branch master
Changes to be committed:
(use "git restore --staged <file> ..." to unstage)
modified: teste.txt
```

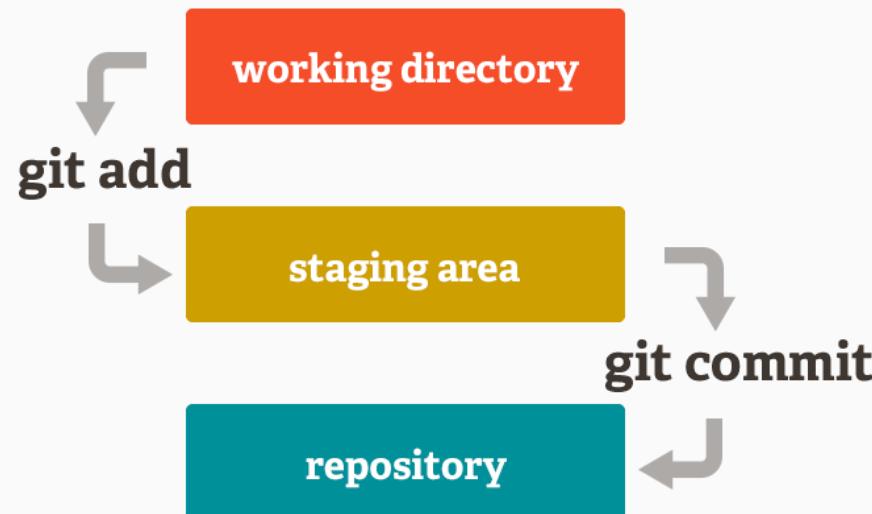


9. Versionamento: git add, git commit e git status

git commit: armazena as mudanças com uma descrição, criando nova versão do repositório (*repository*)

```
git commit -m "mod teste.txt"
```

```
[master dcdbd894] mod teste.txt  
1 file changed, 1 insertion(+)
```



9. Versionamento: git add, git commit e git status

ATENÇÃO!

As mensagens no commit são fundamentais!

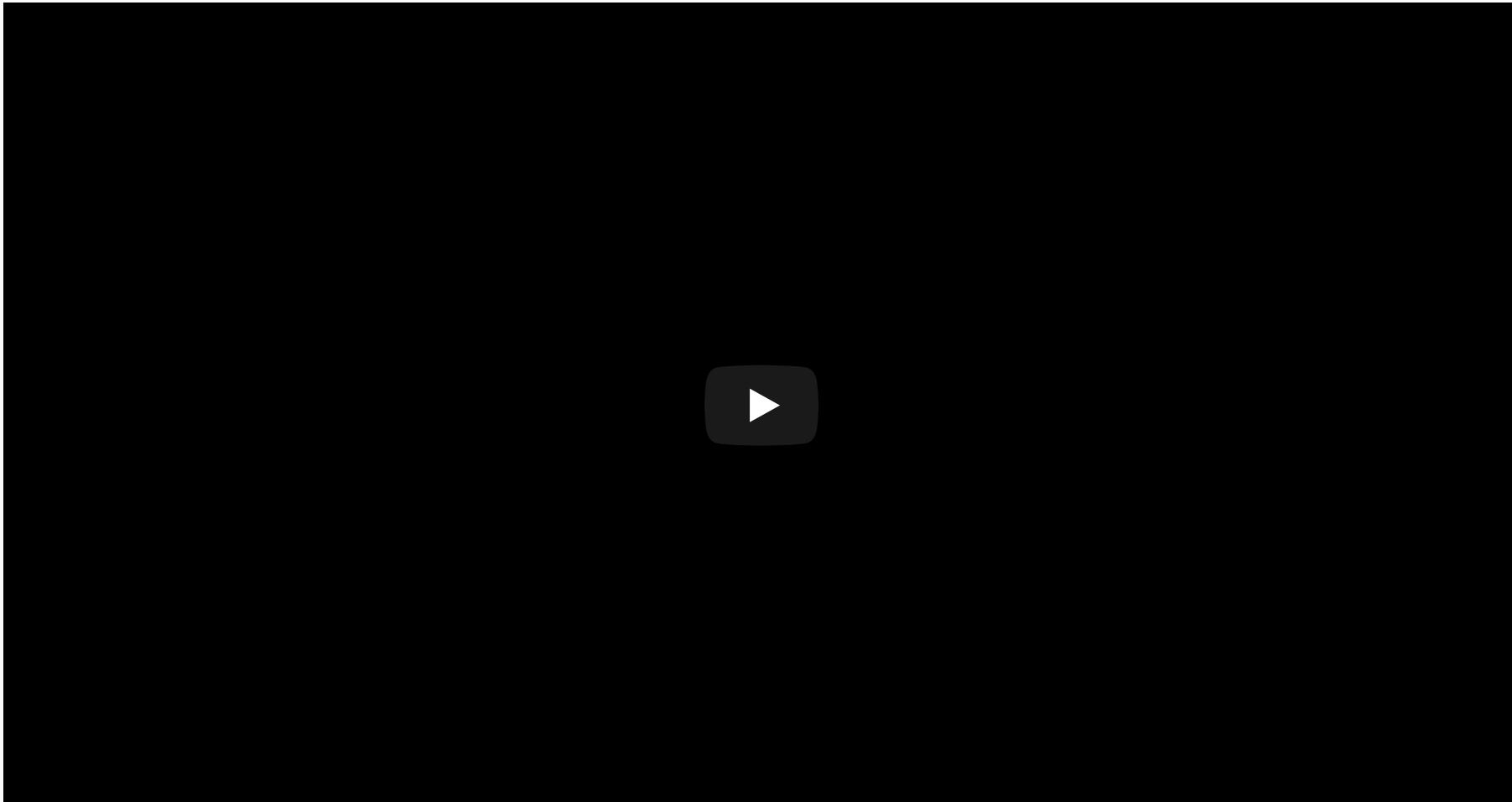
Devem ser curtas e indicar as mudanças feitas nos arquivos do diretório!

	COMMENT	DATE
O	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
O	ENABLED CONFIG FILE PARSING	9 HOURS AGO
O	MISC BUGFIXES	5 HOURS AGO
O	CODE ADDITIONS/EDITS	4 HOURS AGO
O	MORE CODE	4 HOURS AGO
O	HERE HAVE CODE	4 HOURS AGO
O	AAAAAAA	3 HOURS AGO
O	ADKFJSLKDFJSDFKJ	3 HOURS AGO
O	MY HANDS ARE TYPING WORDS	2 HOURS AGO
O	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

9. Versionamento: git add, git commit e git status

Usando Git Direito | Limpando seus Commits! - Fabio Akita



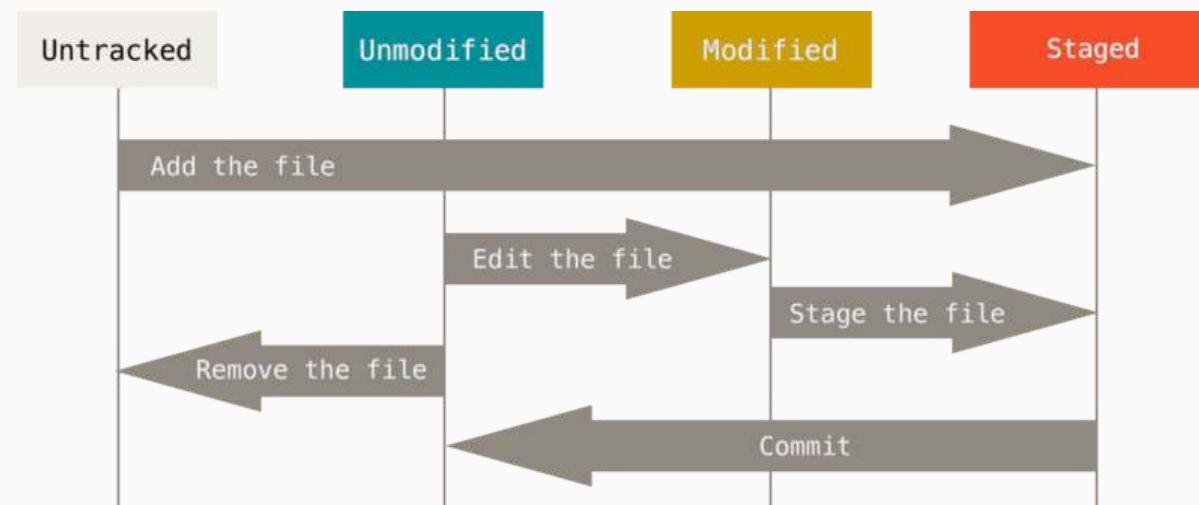
9. Versionamento: git add, git commit e git status

git status: mostra o estado do repositório

```
git status
```

```
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```



E se houver arquivos que eu não quero versionar?

.GITIGNORE



YOU FOOLS

memegenerator.net

10. Ignorando: `.gitignore`

Arquivo `.gitignore`

O **`.gitignore`** é um arquivo especial que indica ao git que determinados arquivos ou diretórios devem ser **ignorados e não comitados**

O **`.gitignore`** é um **arquivo de texto simples** em que cada linha especifica um padrão de arquivos ou diretórios a serem ignorados

Geralmente colocado no **diretório raiz** de um repositório

Atentar para o **ponto**, pois é um arquivo oculto [**`.gitignore`**]

10. Ignorando: .gitignore

Arquivo .gitignore

```
# criar um arquivo  
touch ~/.gitignore
```

Nome de arquivos

- rastertif

Diretórios

- temp/

Caracteres-curinga

- Padrão: *.tif (todos com .tif)
- Negação: !.tif (negar ignore)
- Comentário: # (não negar)

E se houver arquivos [enormes] que eu queira
versionar?

10. Ignorando: .gitignore

Limites

O tamanho máximo de arquivos é 25 Mb (browser)

- Arquivos adicionados via browser é limitado a 25 Mb

O tamanho máximo de arquivos é 100 Mb (terminal)

- Arquivos adicionados via terminal é limitado a 25 Mb
- Se o arquivo exceder esse limite, ocorrerá uma mensagem de erro e o envio ao GitHub será bloqueado

O tamanho máximo de um repositório é 5 GB

- O tamanho total do repositório é limitado a 5 Gb
- Idealmente um repositório dever ser menor que 1 Gb
- Mensagens de aviso informam que o tamanho do repositório se aproxima do limite
- Se o repositório exceder o limite, ocorrerá uma mensagem de erro e o envio ao GitHub será bloqueado



10. Ignorando: .gitignore

Git Large File Storage (LFS)

[Git Large File Storage \(LFS\)](#) substitui arquivos grandes por ponteiros de texto dentro do git

Armazena o conteúdo desses arquivos em um servidor remoto como GitHub.com ou GitHub Enterprise



Dúvidas?

Vamos ver como acessar todo o histórico de commits
do git?

11. Histórico: git log e git show

git log: mostra o registro de todo o histórico de commits

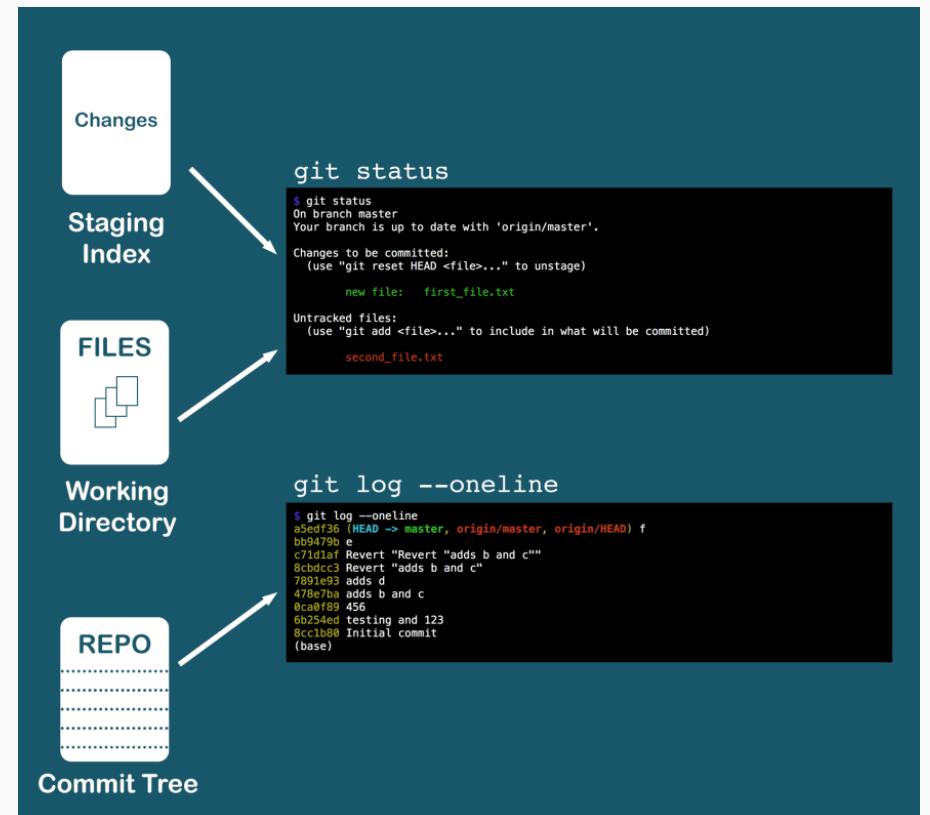
1. Nome do commit (hash ou tag - dcdbd894)

2. Autor

3. E-mail

4. Data

5. Descrição (mensagem do commit)



11. Histórico: git log e git show

git log: mostra o registro de todo o histórico de commits

```
git log
```

```
commit dcbd894ade9a85f78e1b0d6ed08cbc76dcc2448f (HEAD → master)
```

```
Author: mauriciovancine <mauricio.vancine@gmail.com>
```

```
Date: Mon Mar 15 11:03:26 2021 -0300
```

```
mod teste.txt
```

```
commit 8b33bc57187a2bcb59172c75ce7266fbddb64eff
```

```
Author: mauriciovancine <mauricio.vancine@gmail.com>
```

```
Date: Mon Mar 15 11:02:04 2021 -0300
```

```
add teste.txt
```

11. Histórico: git log e git show

git log [arquivo]: registro de todo o histórico de commits para arquivos

```
git log teste.txt
```

```
commit dcbd894ade9a85f78e1b0d6ed08cbc76dcc2448f (HEAD → master)
```

```
Author: mauriciovancine <mauricio.vancine@gmail.com>
```

```
Date: Mon Mar 15 11:03:26 2021 -0300
```

```
mod teste.txt
```

```
commit 8b33bc57187a2bcb59172c75ce7266fbddb64eff
```

```
Author: mauriciovancine <mauricio.vancine@gmail.com>
```

```
Date: Mon Mar 15 11:02:04 2021 -0300
```

```
add teste.txt
```

11. Histórico: git log e git show

git log --oneline: registro de todo o histórico de commits simplificado

```
git log --oneline
```

```
dcbd894 (HEAD → master) mod teste.txt  
8b33bc5 add teste.txt
```

11. Histórico: git log e git show

git log: cheatsheet

Git Log Cheatsheet		
<h3>Basic Log</h3> <pre>git log Logs in Current Branch git log -n 5 Last # of Commits Commits Between Branches git log branch1..branch2 Commits in Branch1 not in Branch2 git log branch1 ^branch2</pre>	<h3>What Changed?</h3> <pre>git log --since="2 weeks ago" ... or ... git whatchanged --since="2 weeks ago"</pre> <h3>Show Changes by File</h3> <pre>git log -p filename.js</pre> <h3>Show Stats with Patch</h3> <pre>git log --stat -p</pre>	<h3>Search for Commit</h3> <p>Pipe via Grep</p> <pre>git log --oneline grep "Commit"</pre> <p>Use --grep Flag</p> <pre>git log --grep="Commit"</pre> <h3>Search by Content</h3> <pre>git log -S"Change in Source Code"</pre>
<h3>Pretty Git Logs</h3> <pre>git log --oneline --decorate --graph ... or use the OhMyZsh alias ... glog</pre> <div style="border: 1px solid red; padding: 5px; width: fit-content;">THERE ARE A LOT MORE OF OHMYZSH GIT ALIASES...</div> <pre>http://bit.ly/ohmyzsh-git</pre>	<h3>Terminal Git Apps</h3> <p>Tig: Text-mode interface for Git</p> <pre>http://bit.ly/git-tig brew install tig</pre> <p>LazyGit: Simple Git Terminal UI</p> <pre>http://bit.ly/git-lazy brew install lazygit</pre>	<h3>Explain Shell</h3> <pre>https://explainshell.com</pre> <p>Type in a git command and it'll break apart the command and explain each section</p> <div style="border: 1px solid red; padding: 5px; width: fit-content;">NOTE: IT'S HELPFUL FOR MORE THAN JUST GIT COMMANDS!</div>

elijahmanor.com

@elijahmanor

11. Histórico: git log e git show

git show: visualização das alterações realizadas nos arquivos

```
git show
```

```
commit dcdbd894ade9a85f78e1b0d6ed08cbc76dcc2448f (HEAD → master)
Author: mauriciovancine <mauricio.vancine@gmail.com>
Date: Mon Mar 15 11:03:26 2021 -0300

mod teste.txt

diff --git a/teste.txt b/teste.txt
index e69de29..f5201c0 100644
--- a/teste.txt
+++ b/teste.txt
@@ -0,0 +1 @@
+Acabo de inserir uma edição ao meu arquivo
\ No newline at end of file
```

11. Histórico: git log e git show

git show: visualização das alterações realizadas nos arquivos

```
git show 8b33bc5
```

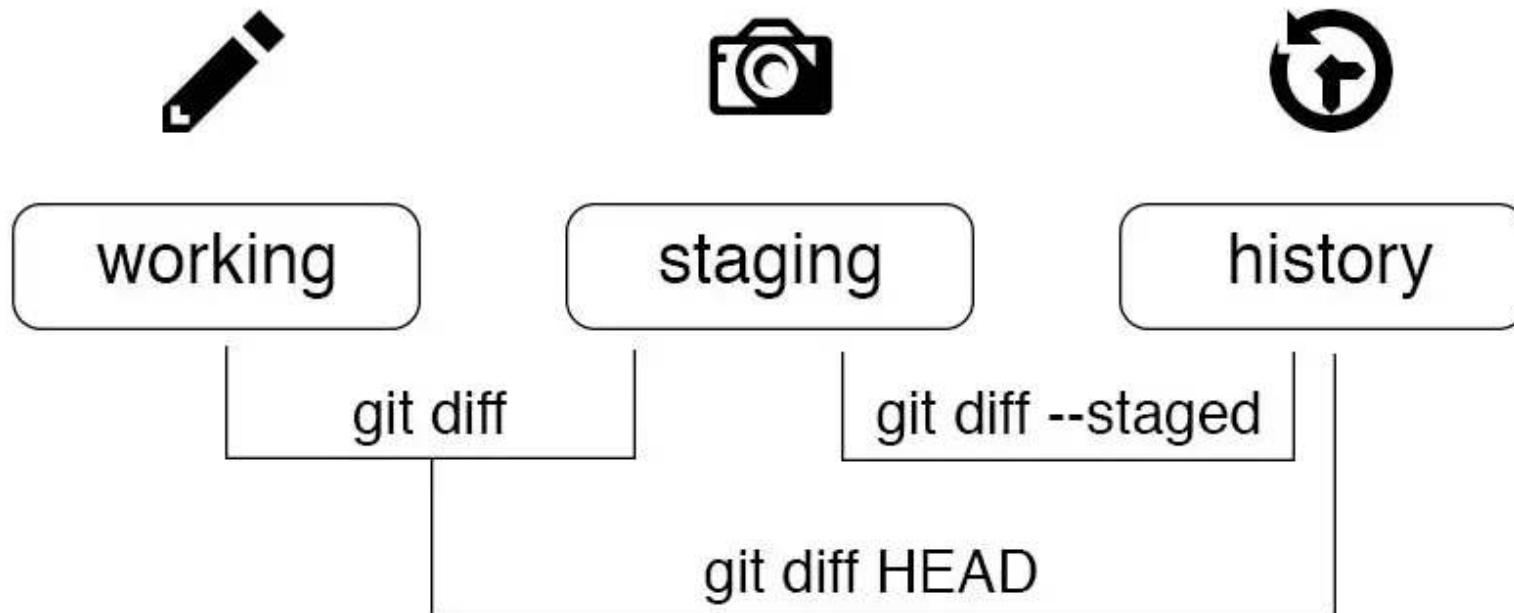
```
commit 8b33bc57187a2bcb59172c75ce7266fbddb64eff
Author: mauriciovancine <mauricio.vancine@gmail.com>
Date: Mon Mar 15 11:02:04 2021 -0300

add teste.txt

diff --git a/teste.txt b/teste.txt
new file mode 100644
index 0000000..e69de29
```

12. Diferenças: git diff

git diff: mostra as diferenças dos arquivos no *working directory*

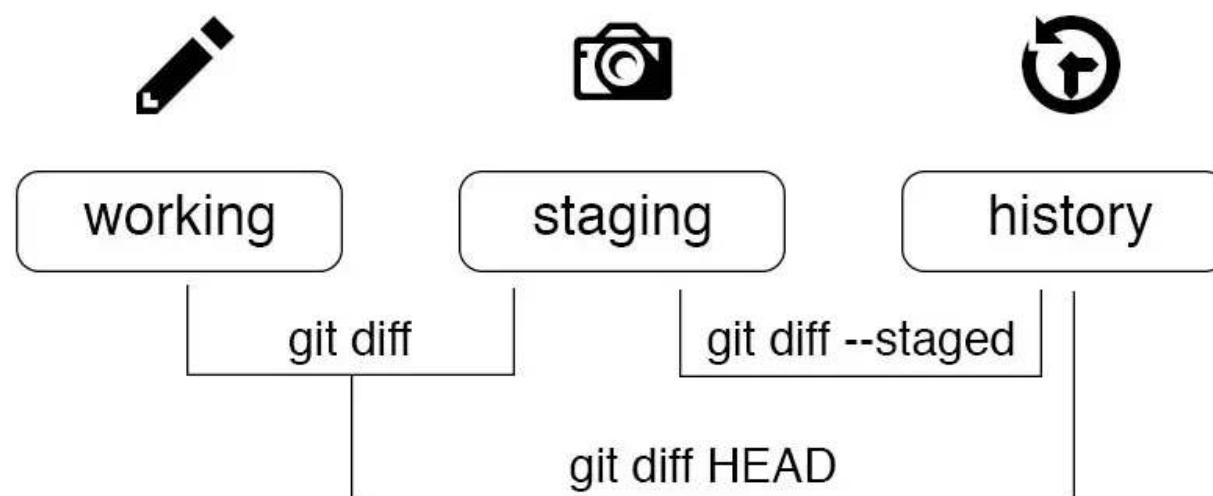


12. Diferenças: git diff

git diff: mostra as diferenças dos arquivos no *working directory*

```
git diff
```

Observação: Não retornou nenhuma diferença...



12. Diferenças: git diff

git diff: mostra as diferenças dos arquivos no *working directory*

Vamos editar o arquivo "teste.txt"

```
Outra linha
```



12. Diferenças: git diff

git diff: mostra as diferenças dos arquivos no *working directory*

```
git diff
```

```
diff --git a/teste.txt b/teste.txt
index f5201c0..71f35ef 100644
--- a/teste.txt
+++ b/teste.txt
@@ -1 +1,3 @@
-Acabo de inserir uma edição ao meu arquivo
\ No newline at end of file
+Acabo de inserir uma edição ao meu arquivo
+
+Outra linha
\ No newline at end of file
```

12. Diferenças: git diff

git diff [arquivo]: mostra as diferenças para o arquivo no *working directory*

```
git diff teste.txt
```

```
diff --git a/teste.txt b/teste.txt
index f5201c0..71f35ef 100644
--- a/teste.txt
+++ b/teste.txt
@@ -1 +1,3 @@
-Acabo de inserir uma edição ao meu arquivo
\ No newline at end of file
+Acabo de inserir uma edição ao meu arquivo
+
+Outra linha
\ No newline at end of file
```

12. Diferenças: git diff

git diff --staged: mostra as diferenças dos arquivos na *staged area*

```
git diff --staged
```

```
diff --git a/teste.txt b/teste.txt
index f5201c0..71f35ef 100644
--- a/teste.txt
+++ b/teste.txt
@@ -1 +1,3 @@
-Acabo de inserir uma edição ao meu arquivo
\ No newline at end of file
+Acabo de inserir uma edição ao meu arquivo
+
+Outra linha
\ No newline at end of file
```

Até aqui, alguma dúvida?

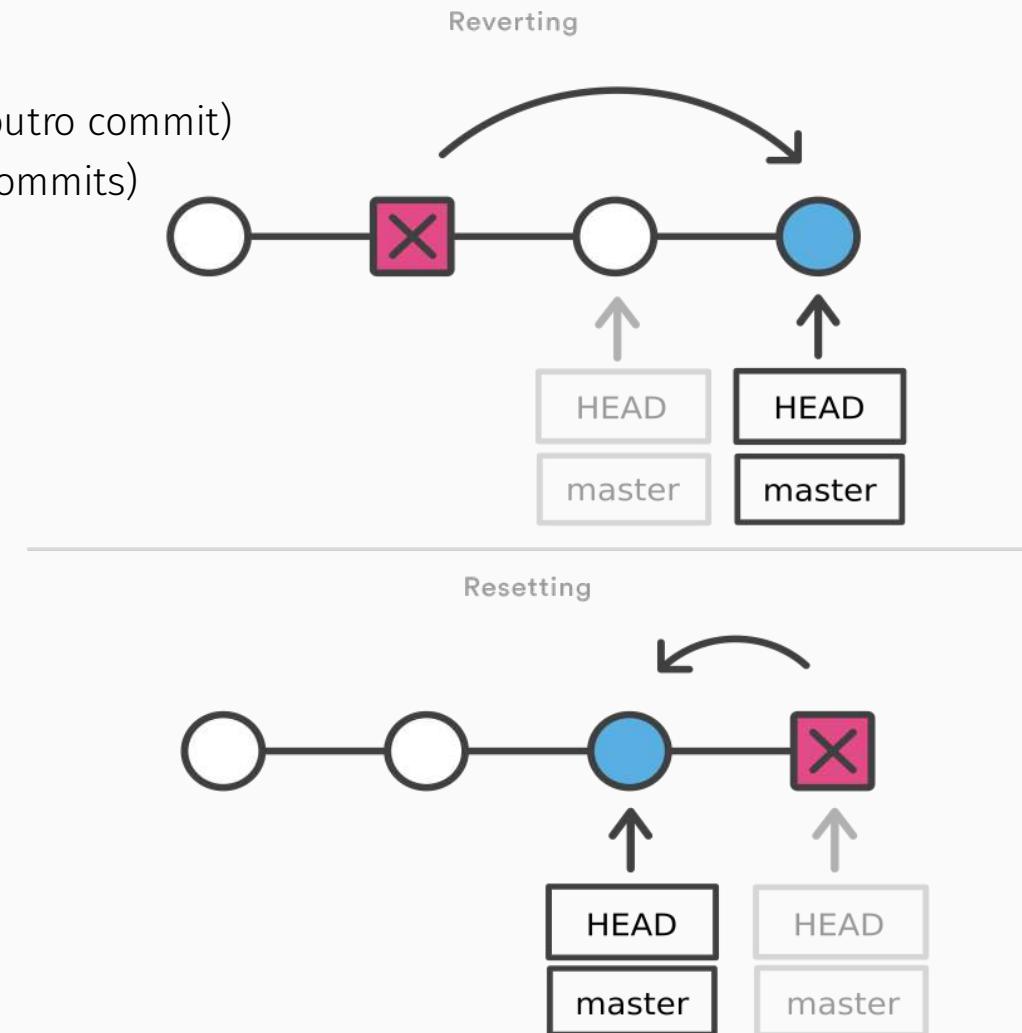
E como voltar no tempo?



13. Desfazer: git revert e git reset

Desfazendo operações no git

1. **git revert**: desfaz um commit **preservando o histórico** (criando outro commit)
2. **git reset**: desfaz um commit **deletando o histórico** (deletando commits)



13. Desfazer: git revert e git reset

1. **git revert**: desfaz um commit preservando o histórico (criando outro commit)

Sintaxe

```
# reverte um commit específico  
git revert <hash do commit>
```

```
# reverter o último commit  
git revert HEAD
```

```
# cancelar uma reversão  
git revert --abort
```

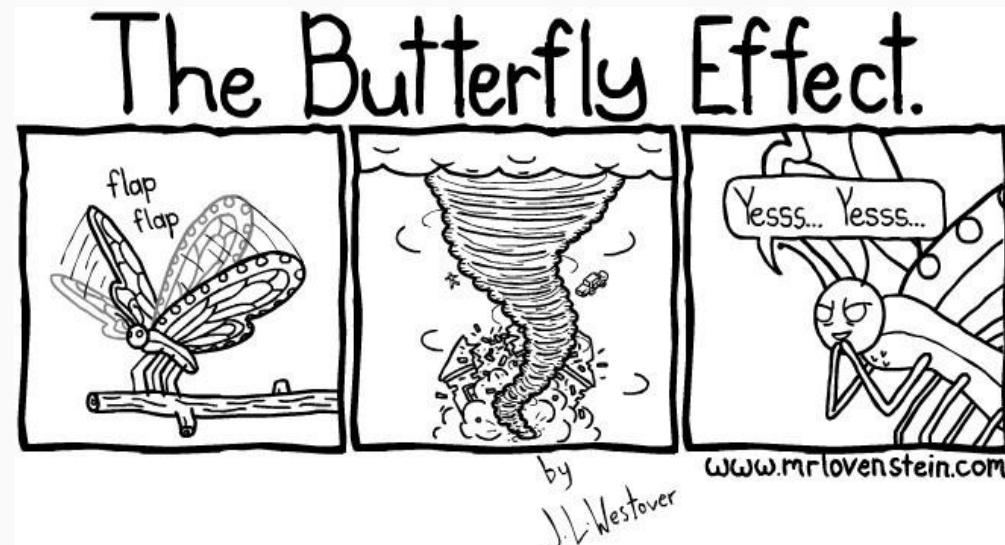
13. Desfazer: git revert e git reset

1. **git revert**: desfaz um commit preservando o histórico (criando outro commit)

IMPORTANTE!

Deve-se trabalhar em um diretório de trabalho limpo. Executar sempre **git add** e **git commit** antes de tentar reverter um commit

2. Tente não **mudar** o passado!



13. Desfazer: git revert e git reset

1. **git revert**: desfaz um commit preservando o histórico (criando outro commit)

Estado

```
git status
```

```
On branch master  
nothing to commit, working tree clean
```

Histórico

```
git log --oneline
```

```
f9f2e9c (HEAD → master) mod2 teste.txt  
dcbd894 mod teste.txt  
8b33bc5 add teste.txt
```

13. Desfazer: git revert e git reset

1. **git revert**: desfaz um commit preservando o histórico (criando outro commit)

Reverter - mensagem

```
git revert f9f2e9c
```

```
Revert "mod2 teste.txt"
```

```
This reverts commit f9f2e9c867cf549d0efcab1519a5ec9d86e7d0d.
```

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       modified:  teste.txt
#
```

13. Desfazer: git revert e git reset

1. **git revert**: desfaz um commit preservando o histórico (criando outro commit)

Reverter

```
git revert f9f2e9c
```

```
[master fca02a2] Revert "mod2 teste.txt"  
1 file changed, 1 insertion(+), 3 deletions(-)
```

Estado

```
git status
```

```
On branch master  
nothing to commit, working tree clean
```

13. Desfazer: git revert e git reset

1. **git revert**: desfaz um commit preservando o histórico (criando outro commit)

Histórico

```
git log --oneline
```

```
fca02a2 (HEAD → master) Revert "mod2 teste.txt"  
f9f2e9c mod2 teste.txt  
dcbd894 mod teste.txt  
8b33bc5 add teste.txt
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Sintaxe

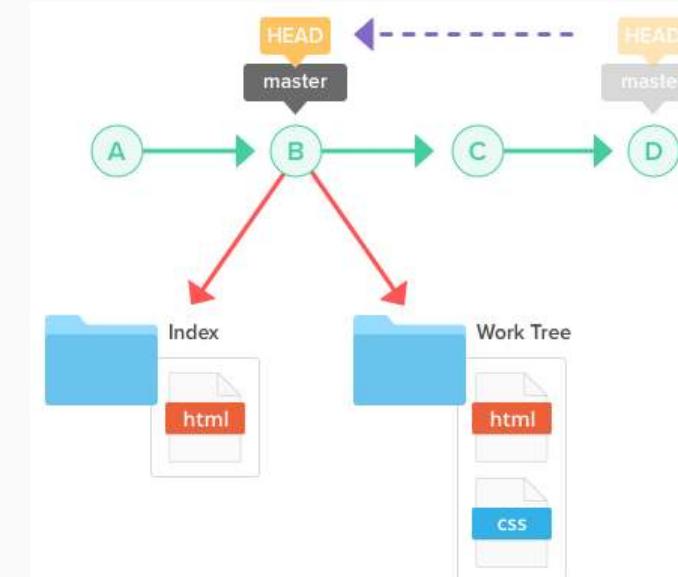
```
git reset [modo] [arquivo|commit]
```

Aplicações

- descartar commits em um branch privado
- desfazer alterações não comitadas (branch privado)
- tirar arquivos da área de staged

Observações

- descartar commits de um branch público: **git revert**
- descartar alterações no diretório de trabalho: **git checkout**

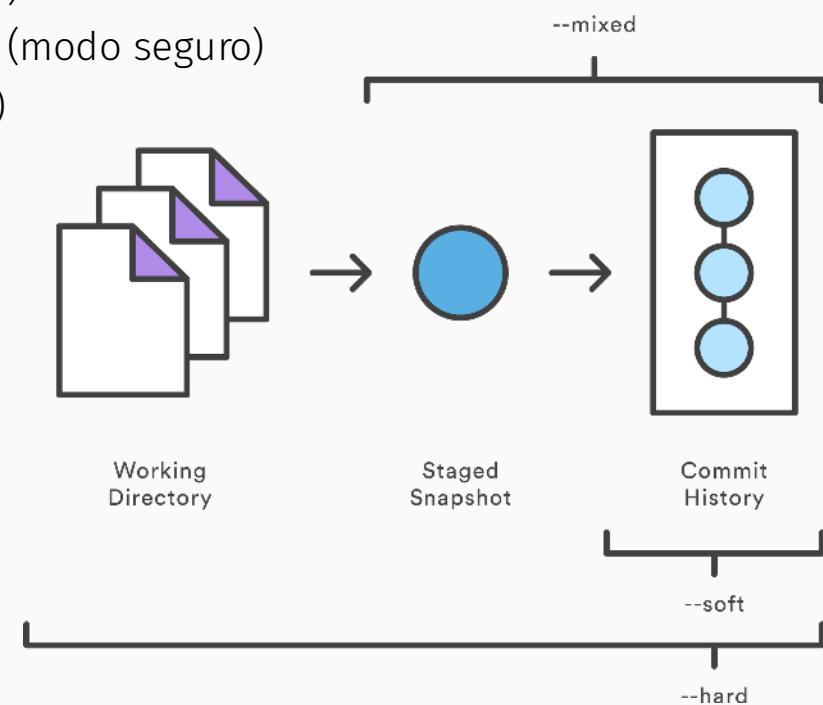
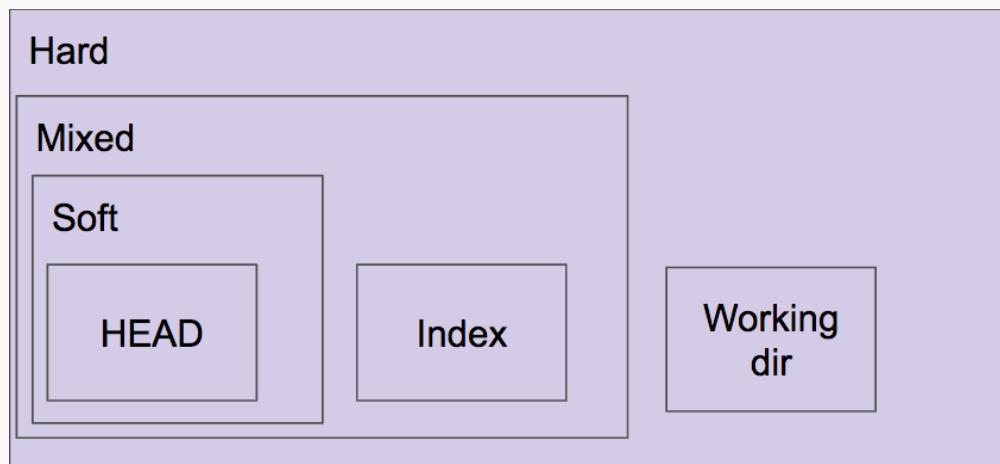


13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Modos principais

- **--soft**: não reseta o índice ou o diretório de trabalho (modo seguro)
- **--mixed (default)**: reseta o índice, mas não o diretório de trabalho (modo seguro)
- **--hard**: reseta o índice e o diretório de trabalho (modo não seguro)



13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Exemplos

git reset

Reseta a *staging area* (retira os arquivos) para corresponder ao commit mais recente, sem modificar o *working directory*

git reset teste.txt

Retira o teste.txt da *staging area*, sem modificar o *working directory*

git reset --hard

Reseta a *staging area* e o *working directory* para corresponder ao commit mais recente. Todas as alterações no diretório são sobreescritas

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Exemplos

git reset

Move a versão para o commit indicado, reseta a *staging area* para corresponder a ele, mas não modifica o *working directory*

git reset --hard

Move a versão para o commit indicado e reseta tanto a *staging area*, quanto o *working directory* para corresponder ao commit especificado

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Estado

```
git status
```

```
On branch master  
nothing to commit, working tree clean
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Vamos editar o arquivo "teste.txt"

```
Acabo de inserir uma edição ao meu arquivo  
Viagens no tempo podem ser perigosas ...
```



13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Estado

```
git status
```

```
On branch master
Changes not staged for commit:
  (use "git add <file> ..." to update what will be committed)
  (use "git restore <file> ..." to discard changes in working directory)
modified:  teste.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Adicionar

```
git add -Av
```

```
add 'teste.txt'
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Estado

```
git status
```

```
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified: teste.txt
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Remover do staged area

Resetar

```
git reset teste.txt
```

```
Unstaged changes after reset:
```

```
M  teste.txt
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Estado

```
git status
```

```
On branch master
Changes not staged for commit:
  (use "git add <file> ..." to update what will be committed)
  (use "git restore <file> ..." to discard changes in working directory)
modified:  teste.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Ver o conteúdo do arquivo

```
cat teste.txt
```

```
Acabo de inserir uma edição ao meu arquivo  
Viagens no tempo podem ser perigosas ...
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Histórico

```
git log --oneline
```

```
fca02a2 (HEAD → master) Revert "mod2 teste.txt"  
f9f2e9c mod2 teste.txt  
dcbd894 mod teste.txt  
8b33bc5 add teste.txt
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Adicionar

```
git add -Av
```

```
add 'teste.txt'
```

Commitar

```
git commit -m "mod3 teste.txt"
```

```
[master 4b0c4e8] mod3 teste.txt
1 file changed, 2 insertions(+), 1 deletion(-)
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Histórico

```
git log --oneline
```

```
4b0c4e8 (HEAD → master) mod3 teste.txt
fca02a2 Revert "mod2 teste.txt"
f9f2e9c mod2 teste.txt
dcbd894 mod teste.txt
8b33bc5 add teste.txt
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Resetar

```
git reset --hard f9f2e9c
```

```
HEAD is now at f9f2e9c mod2 teste.txt
```

Histórico

```
git log --oneline
```

```
f9f2e9c (HEAD → master) mod2 teste.txt  
dcbd894 mod teste.txt  
8b33bc5 add teste.txt
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Remover arquivo

```
rm teste.txt
```

Adicionar

```
git add -Av
```

Commitar

```
git commit -m "rem text.txt"
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Histórico

```
git log --oneline
```

```
6a417c5 (HEAD → master) rem text.txt
f9f2e9c mod2 teste.txt
dcbd894 mod teste.txt
8b33bc5 add teste.txt
```

13. Desfazer: git revert e git reset

2. **git reset**: desfaz um commit deletando o histórico (deletando commits)

Resetar

```
git reset --hard f9f2e9c
```

```
HEAD is now at f9f2e9c mod2 teste.txt
```

Histórico

```
git log --oneline
```

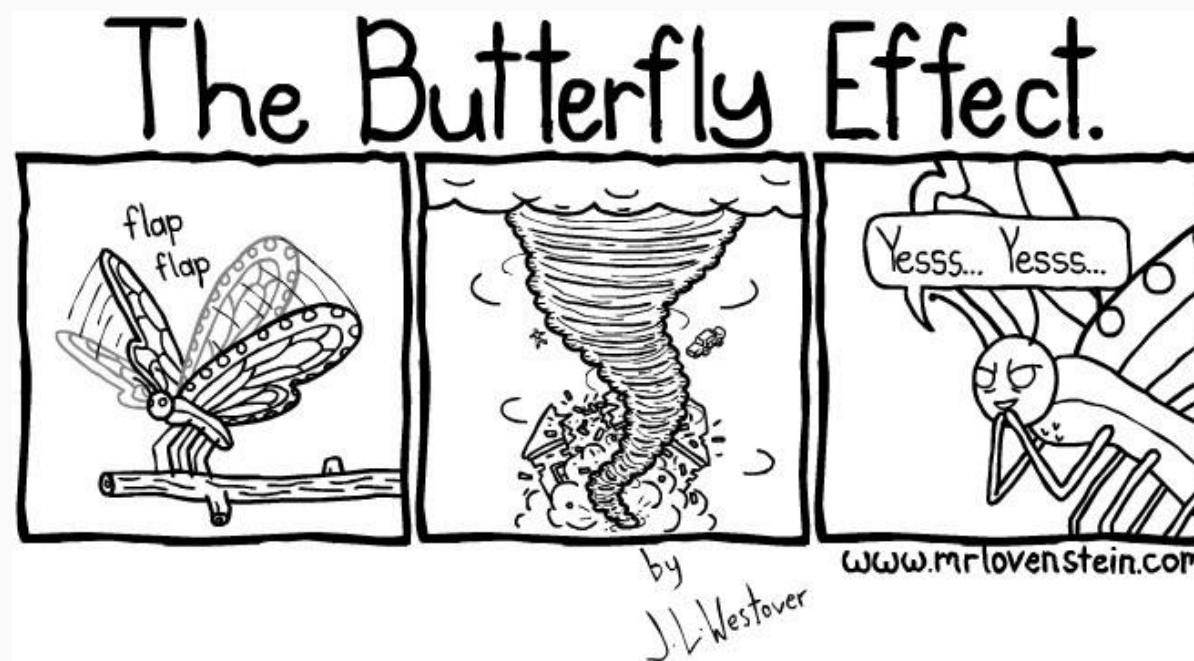
```
f9f2e9c (HEAD → master) mod2 teste.txt  
dcbd894 mod teste.txt  
8b33bc5 add teste.txt
```



13. Desfazer: git revert e git reset

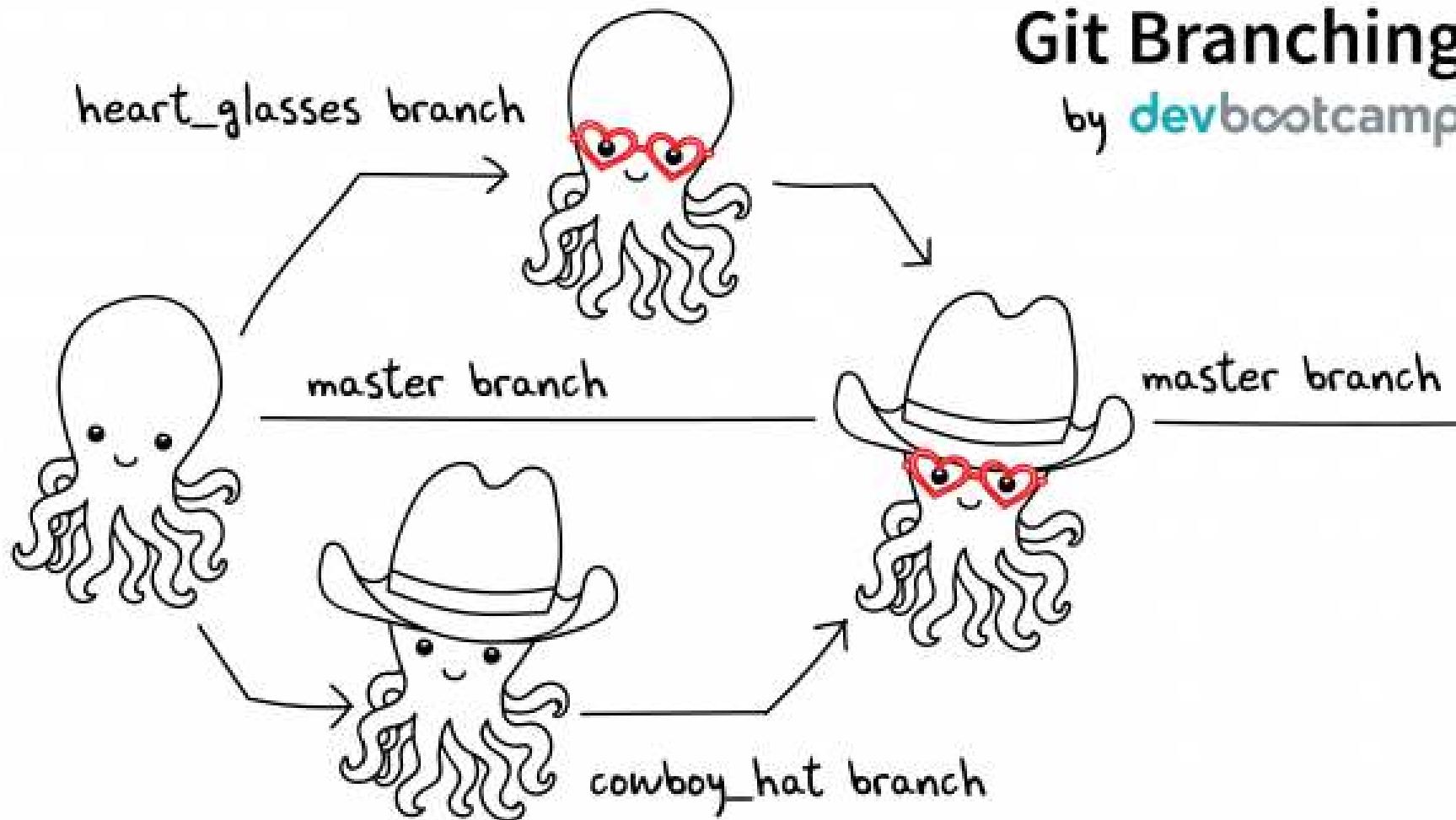
Desfazendo operações no git

1. **git revert**: desfaz um commit **preservando o histórico** (criando outro commit)
2. **git reset**: desfaz um commit **deletando o histórico** (deletando commits)



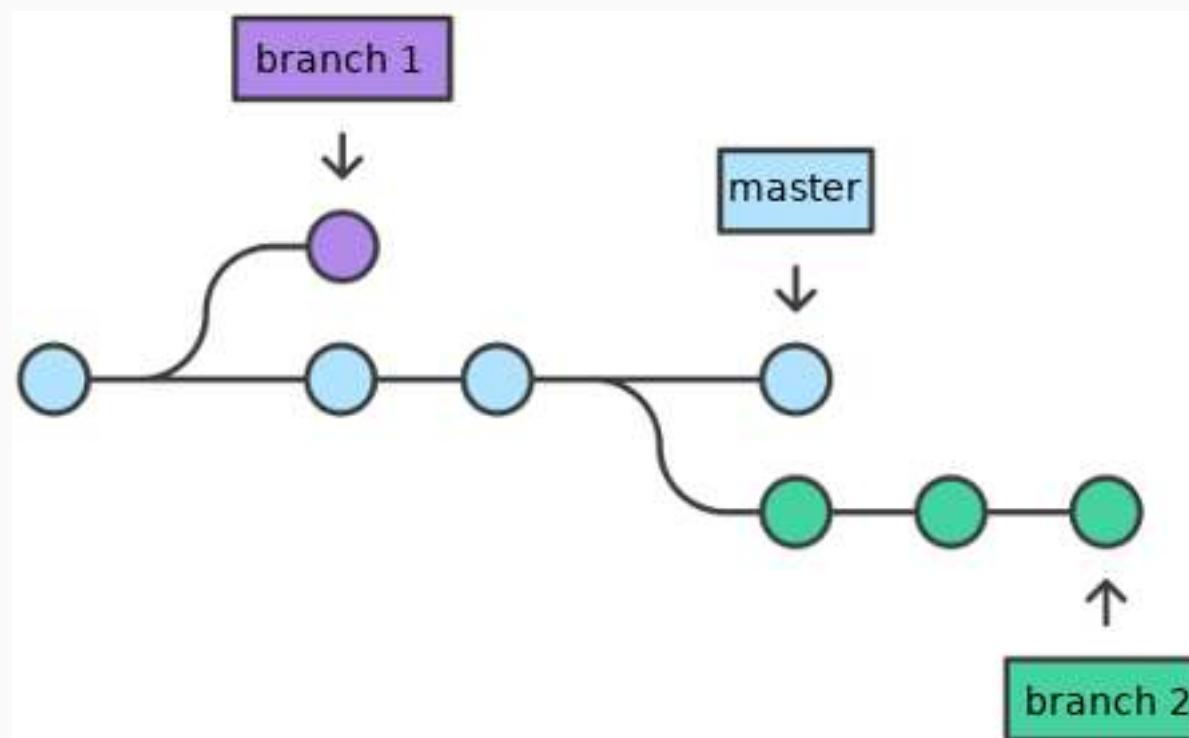
Git Branching

by **devbootcamp**



14. Ramificações: git branch, git switch e git merge

git branch: cria uma ramificação (**linha independente**) de desenvolvimento

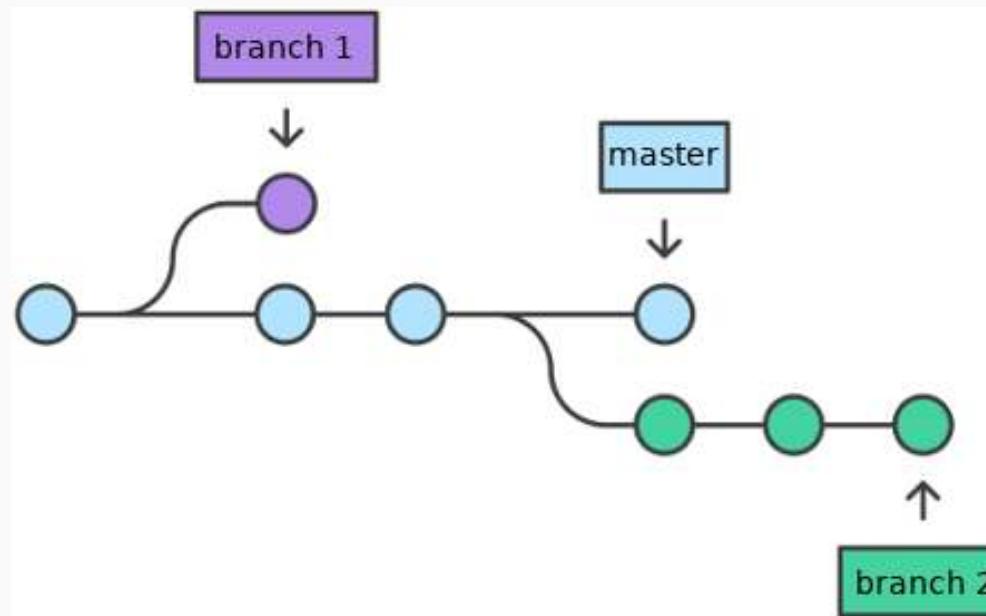


14. Ramificações: git branch, git switch e git merge

git branch: lista os branches e verifica o branch de edição

```
git branch
```

```
* master
```



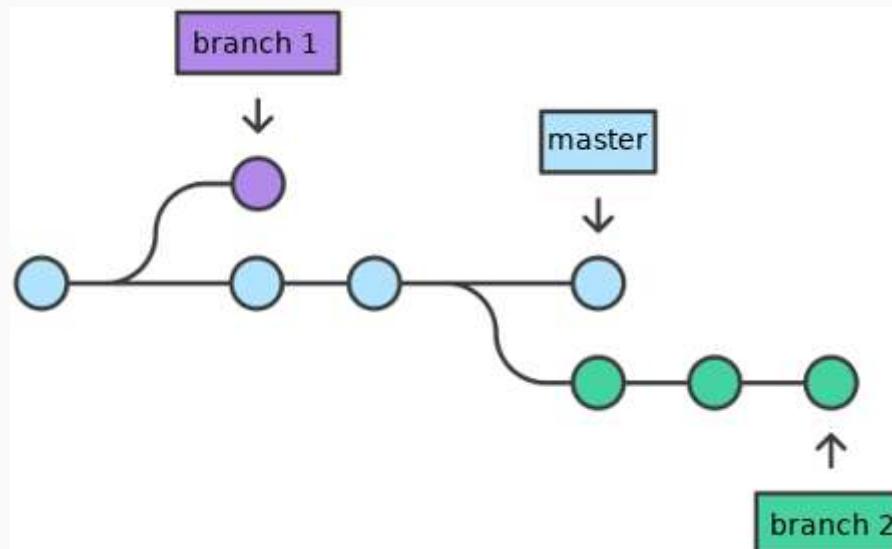
14. Ramificações: git branch, git switch e git merge

git branch: cria uma ramificação

```
git branch branch-1
```

```
git branch
```

```
branch-1  
* master
```

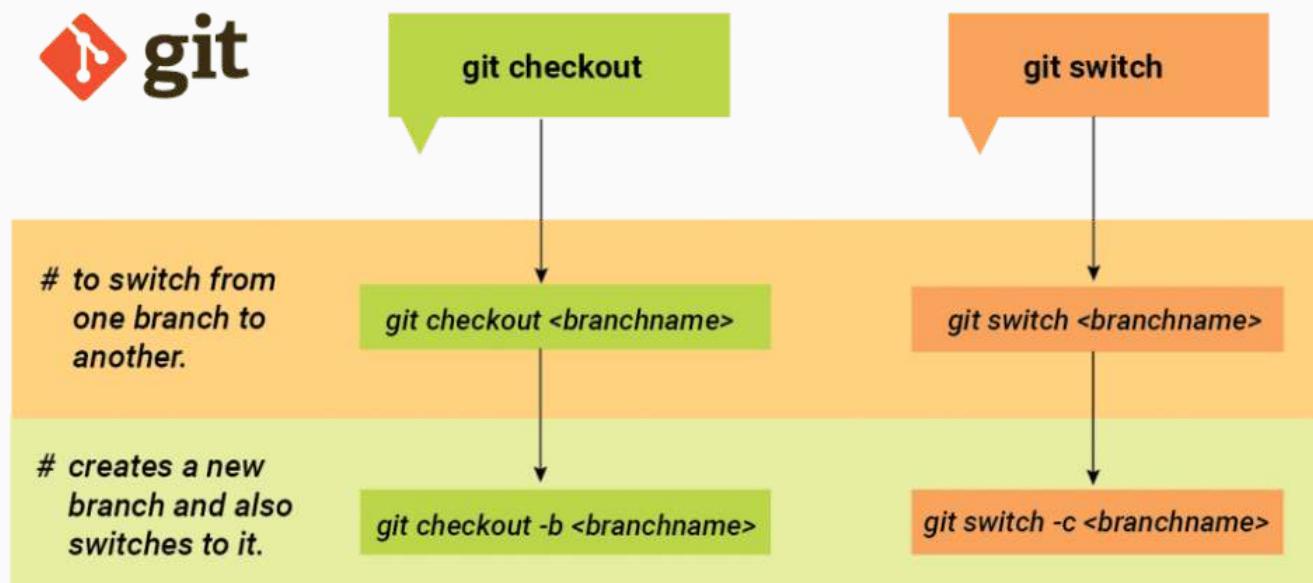


14. Ramificações: git branch, git switch e git merge

git switch: troca a raficação de edição

```
git switch branch-1
```

```
Switched to branch 'branch-1'
```

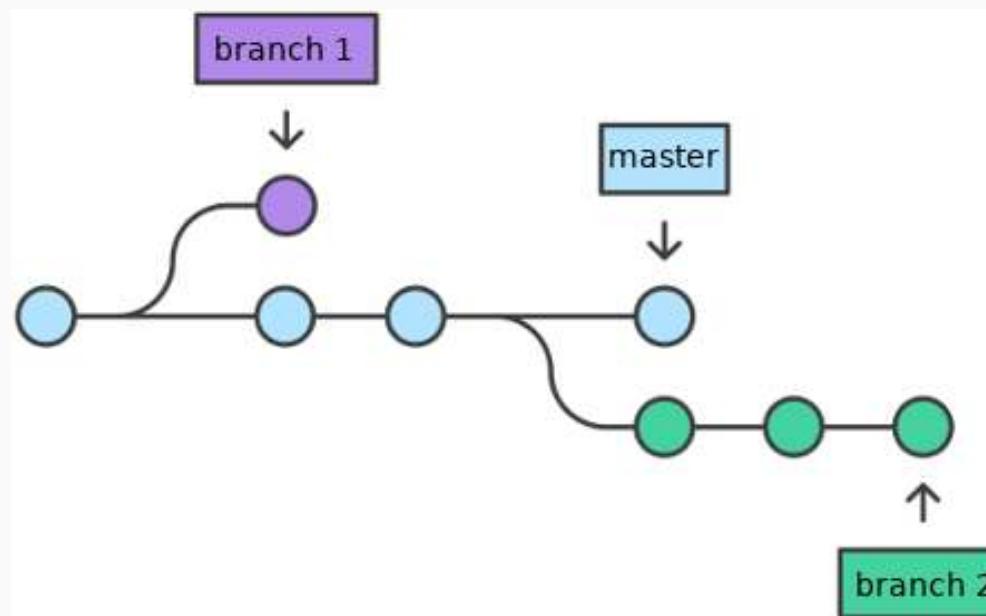


14. Ramificações: git branch, git switch e git merge

git branch: lista os branches e verifica o branch de edição

```
git branch
```

```
* branch-1  
master
```



14. Ramificações: git branch, git switch e git merge

Vamos criar um arquivo no diretório

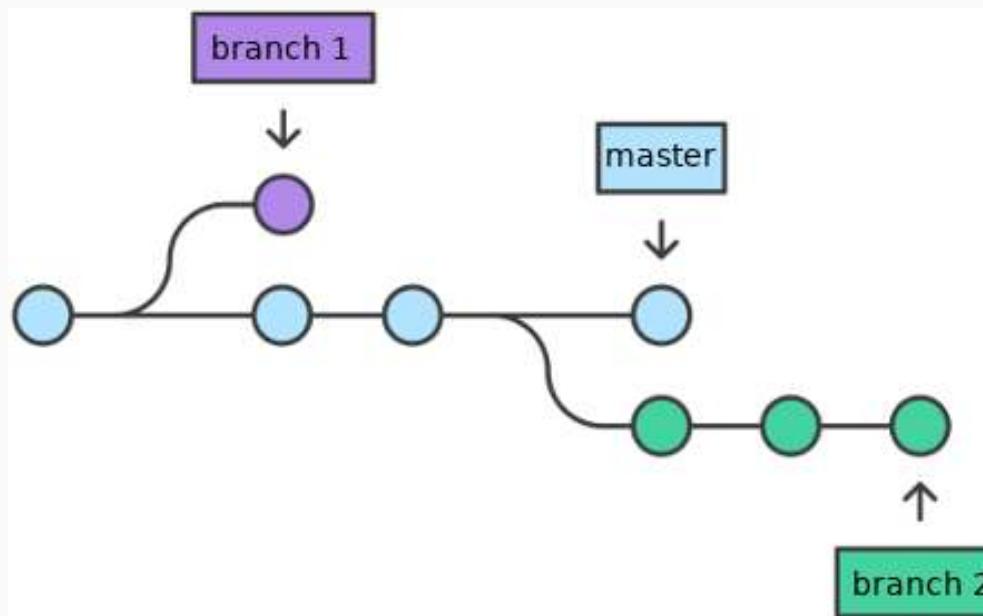
```
# terminal  
touch teste_branch1.txt  
git add -Av  
git commit -m "add teste_branch1.txt"
```



14. Ramificações: git branch, git switch e git merge

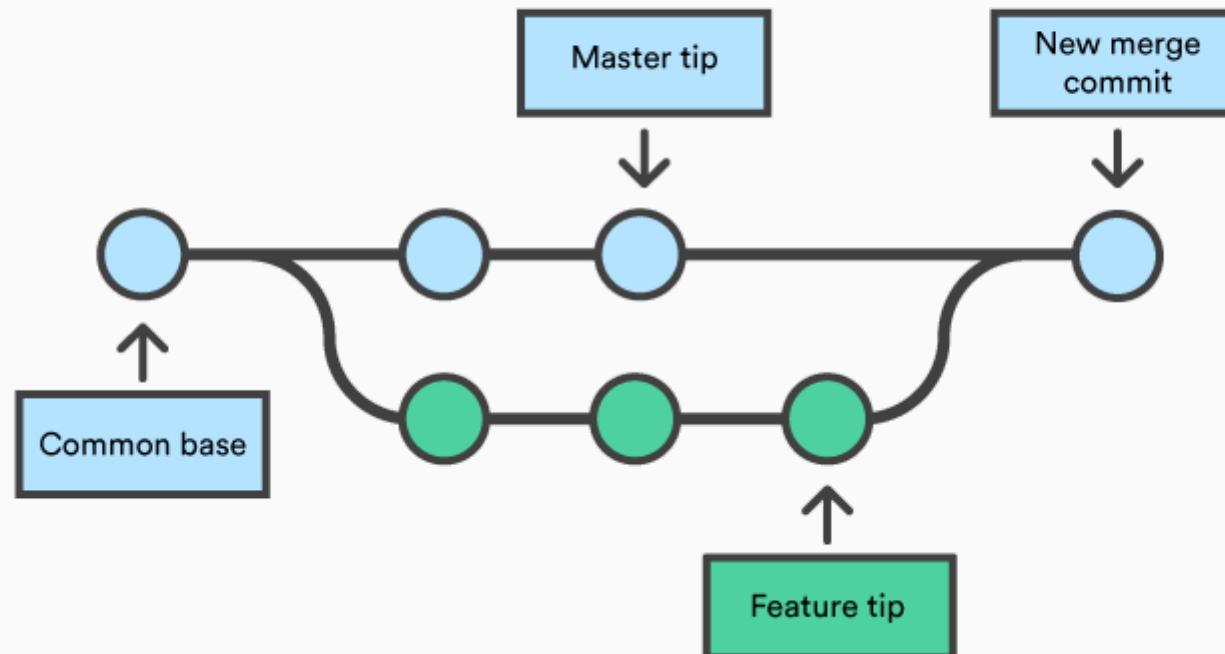
switch: troca a raficação de edição

```
git switch master
```



14. Ramificações: git branch, git switch e git merge

git merge: mescla as linhas de desenvolvimento independentes em um único branch

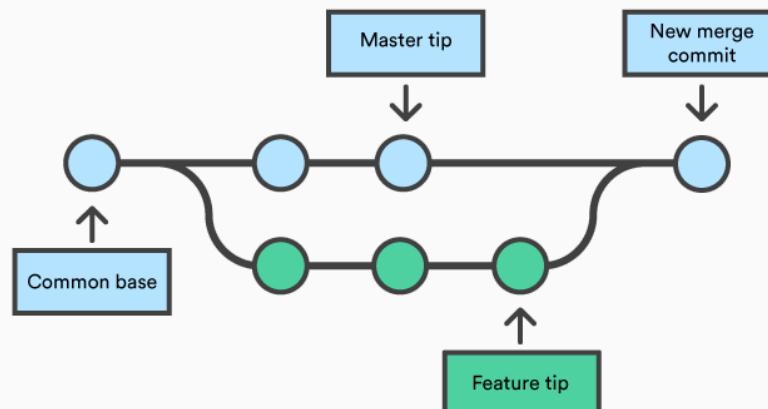


14. Ramificações: git branch, git switch e git merge

git merge: mescla as linhas de desenvolvimento independentes em um único branch

```
git merge branch-1
```

```
Updating f9f2e9c..7aeb06d
Fast-forward
  teste_branch1.txt | 0
  1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 teste_branch1.txt
```

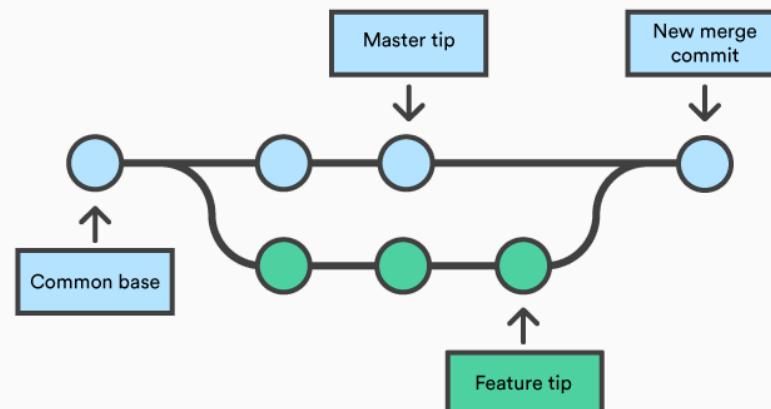


14. Ramificações: git branch, git switch e git merge

git merge: mescla as linhas de desenvolvimento independentes em um único branch

```
git log --oneline
```

```
7aeb06d (HEAD → master, branch-1) add teste_branch1.txt  
f9f2e9c mod2 teste.txt  
dcbd894 mod teste.txt  
8b33bc5 add teste.txt
```



14. Ramificações: git branch, git switch e git merge

CUIDADO!



14. Ramificações: git branch, git switch e git merge

CUIDADO!



Origin

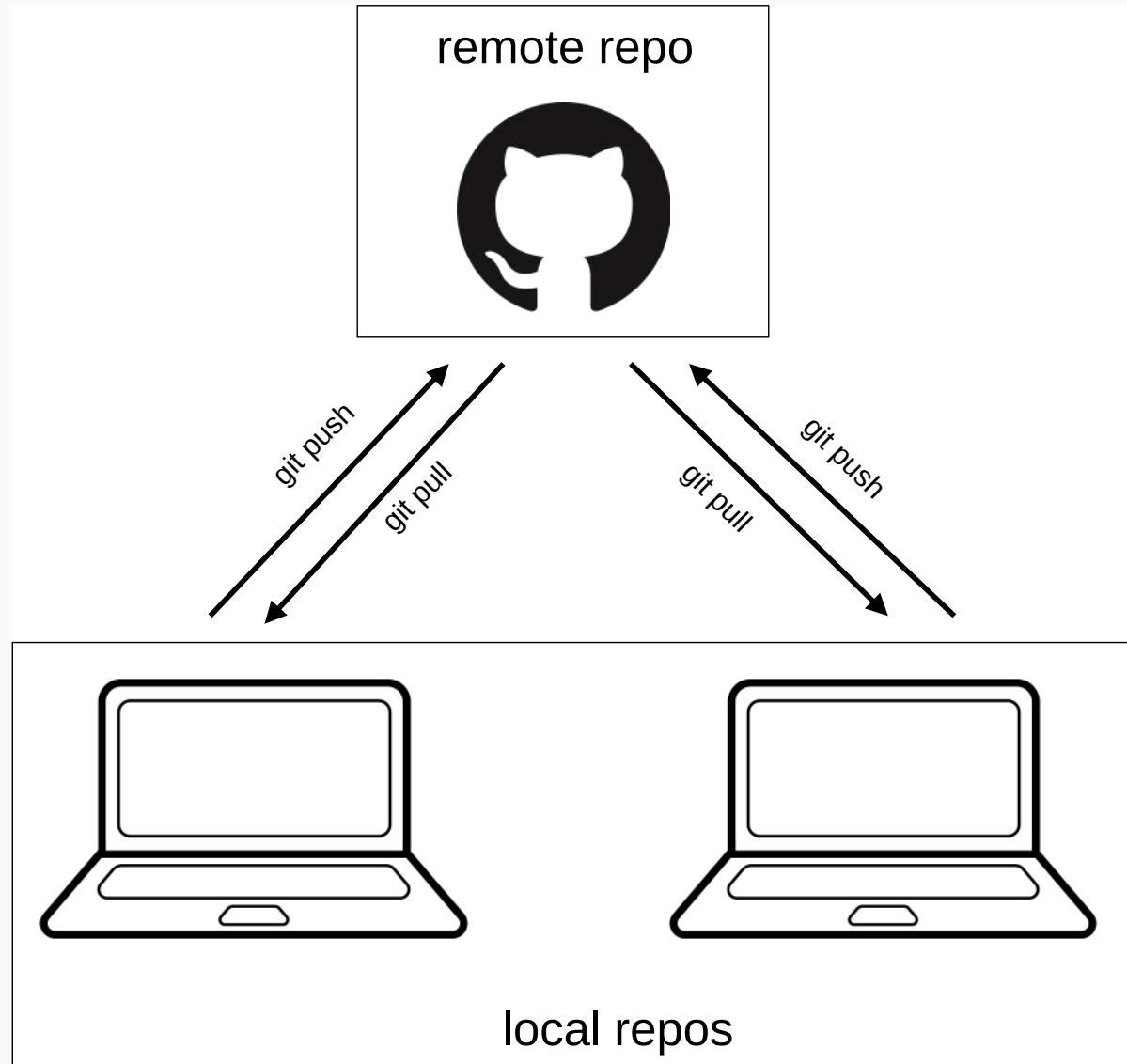


Master

>git merge



Merging would be like...

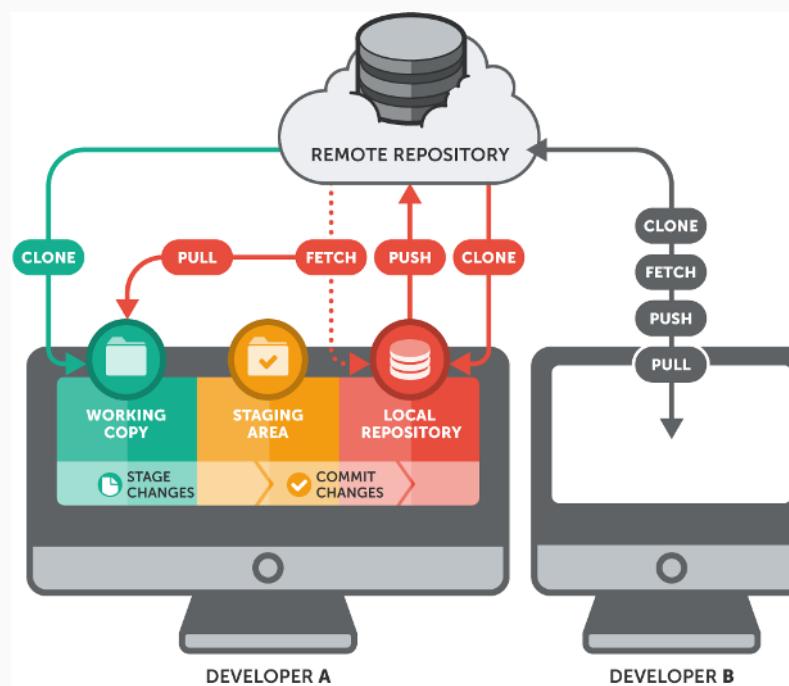


15. Remoto: git remote, git push e git pull

git remote: exibe os repositórios remotos (GitHub)

```
git remote -v
```

```
origin  git@github.com:mauriciovancine/course-geospatial-data-r (fetch)
origin  git@github.com:mauriciovancine/course-geospatial-data-r (push)
```

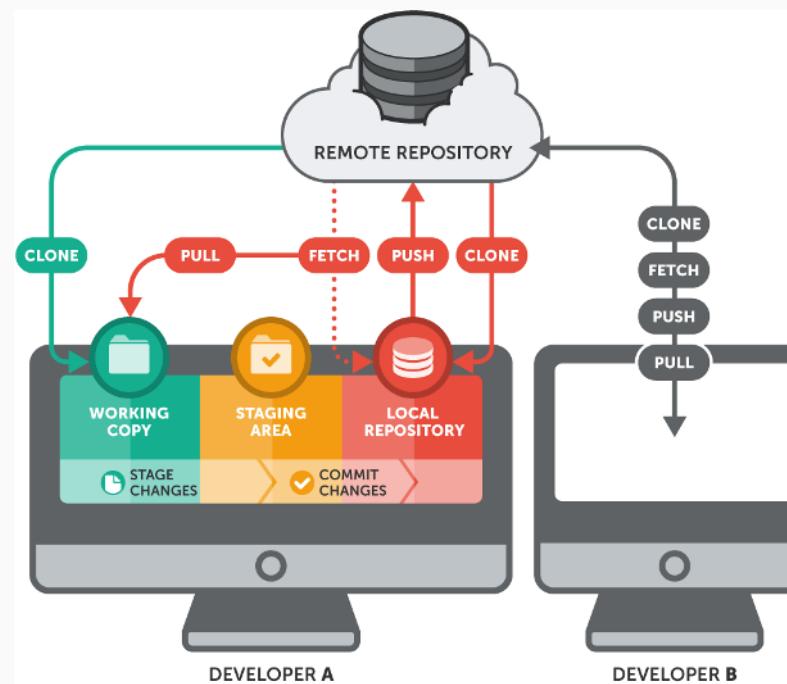


15. Remoto: git remote, git push e git pull

git remote add: faz a ligação com um repositório remoto (GitHub)

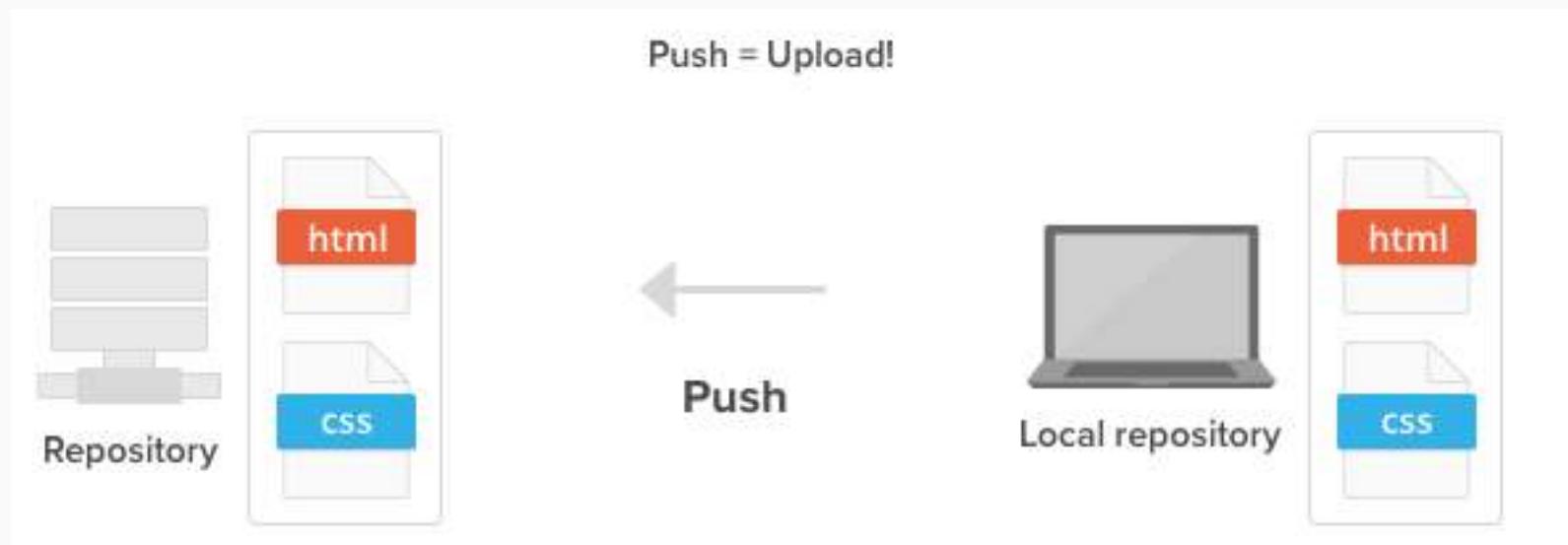
Observação: Usar o SSH para não precisar digitar sua senha!

```
git remote add origin git@github.com:mauriciovancine/course-geospatial-data-r
```



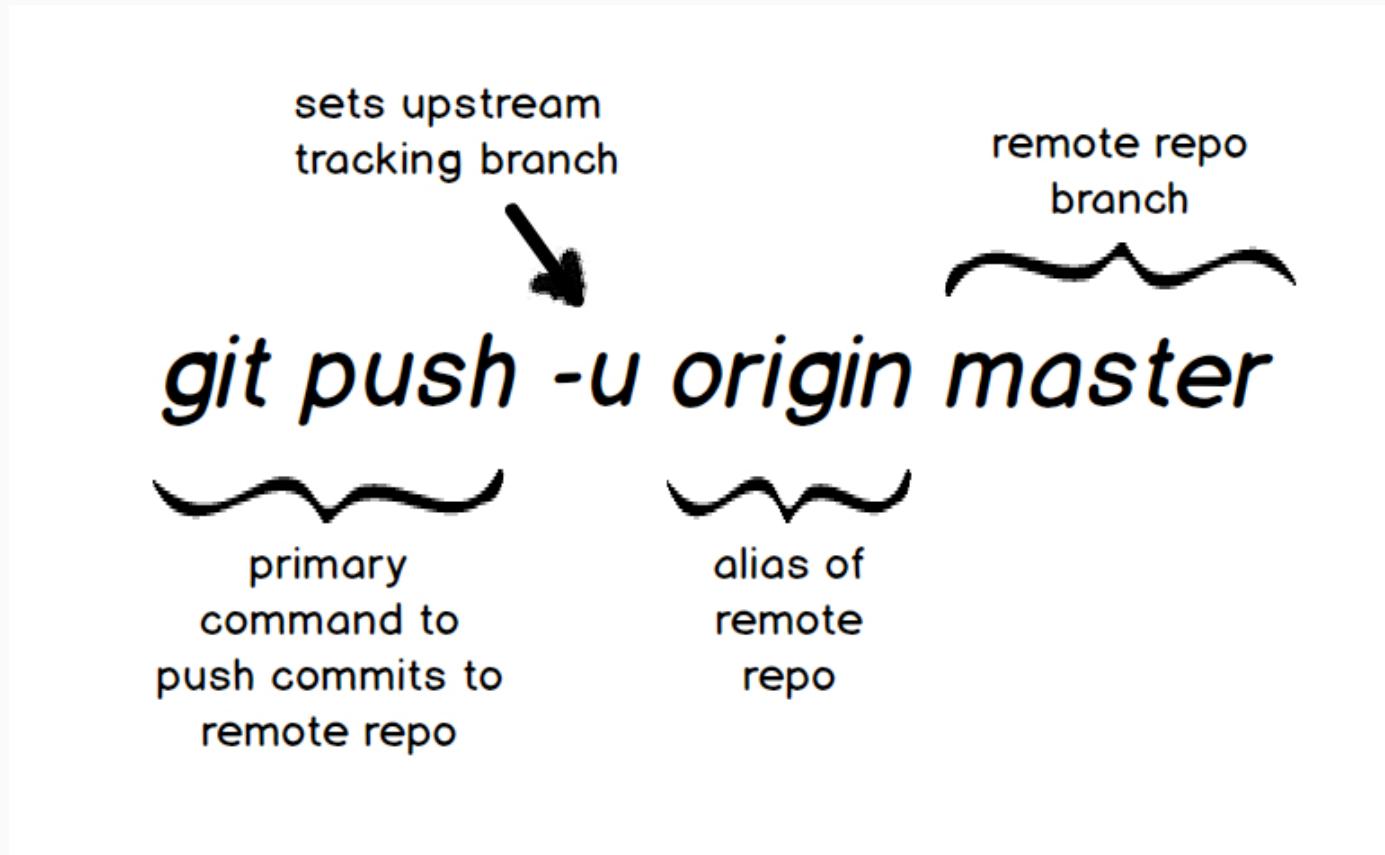
15. Remoto: git remote, git push e git pull

git push: empurra uma nova versão do repositório local para o repositório remoto (GitHub)



15. Remoto: git remote, git push e git pull

git push: empurra uma nova versão do repositório local para o repositório remoto (GitHub)



15. Remoto: git remote, git push e git pull

git push: empurra uma nova versão do repositório local para o repositório remoto (GitHub)

```
git push -u origin master
```

```
Enumerating objects: 41, done.  
Counting objects: 100% (41/41), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (35/35), done.  
Writing objects: 100% (35/35), 4.99 MiB | 155.00 KiB/s, done.  
Total 35 (delta 6), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.  
To github.com:mauriciovancine/course-geospatial-data-r.git  
accc9db..8f70e96 master → master
```

15. Remoto: git remote, git push e git pull

Em caso de incêndio...

In case of fire



1. git commit



2. git push



3. leave building

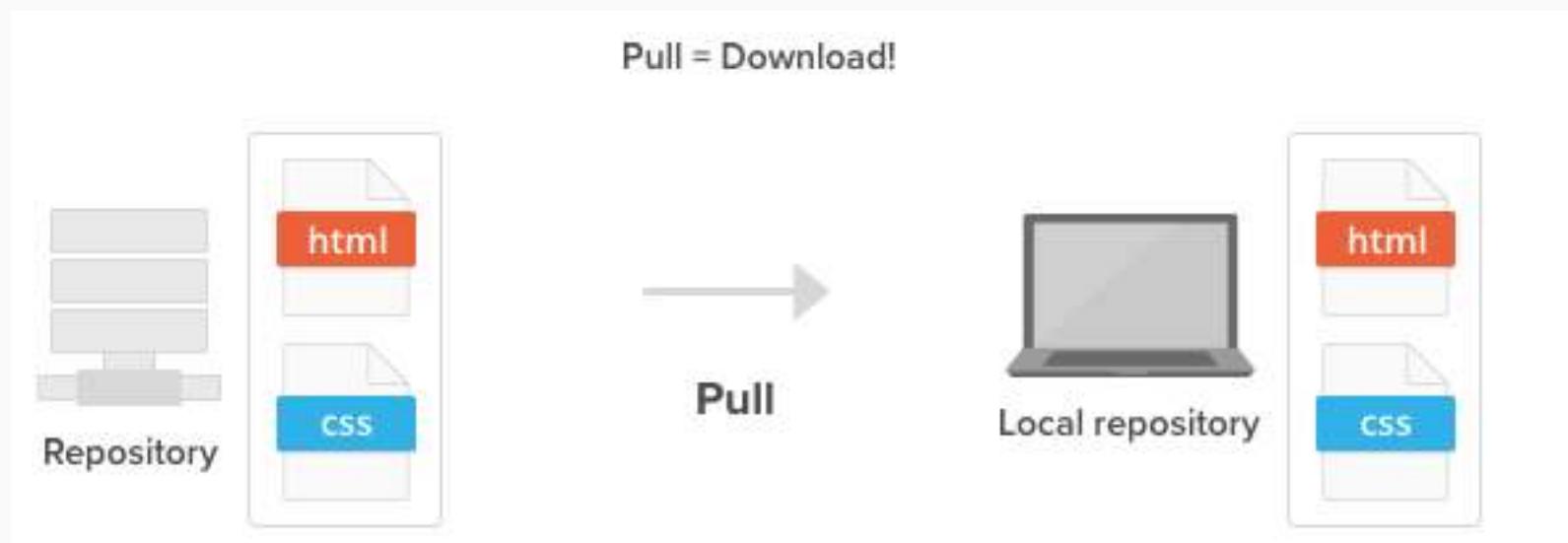
15. Remoto: git remote, git push e git pull

CUIDADO!



15. Remoto: git remote, git push e git pull

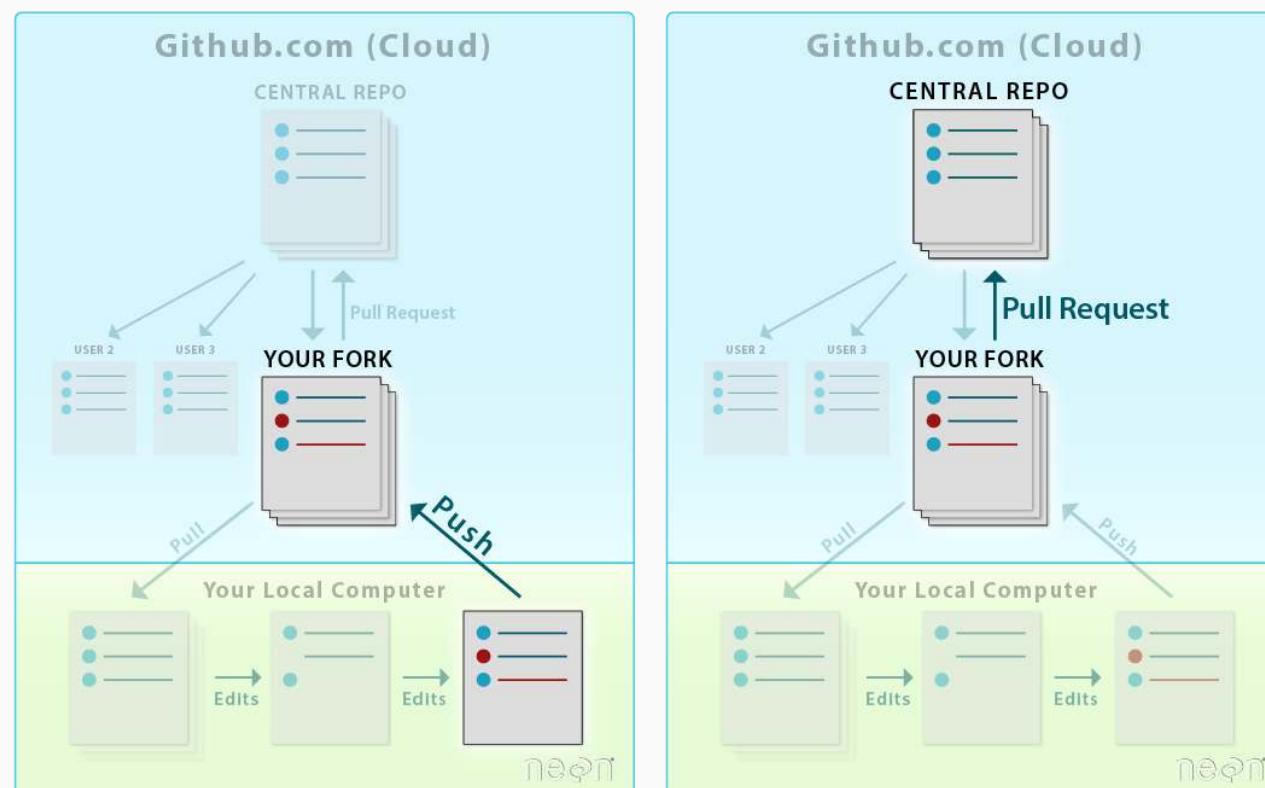
git pull: puxa uma nova versão do repositório remoto (GitHub) para o repositório local



Por fim, fazermos a requisição da mudança do nosso repositório remoto para o repositório remoto original com o **Pull request** (pedimos para que "pxuem" nossas mudanças)

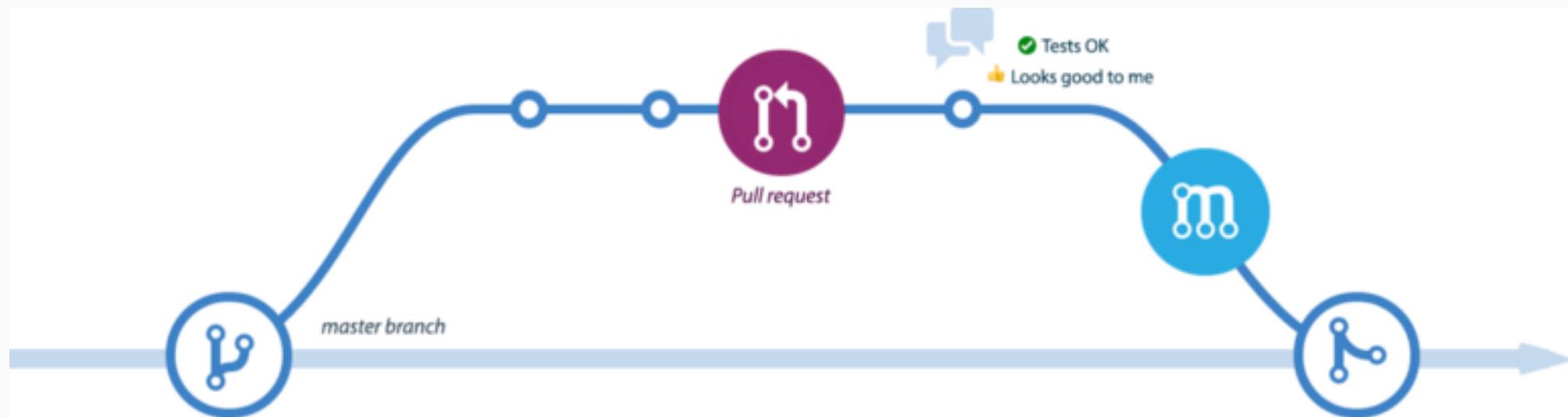
16. GitHub: Pull request

Pull Request: solicita que o repositório central (remoto) aceite (pull - 'puxa') as alterações realizadas do nosso fork



16. GitHub: Pull request

Pull Request: solicita que o repositório central (remoto) aceite (pull - 'puxa') as alterações realizadas do nosso fork



17. Detalhes do repositório do GitHub

Vamos detalhar alguns pontos dos repositórios no GitHub

<https://github.com/mauriciovancine/course-geospatial-data-r>

The screenshot shows the GitHub repository page for 'course-geospatial-data-r'. The repository is public and has 1 branch and 0 tags. The master branch has 85 commits. The repository description is 'Repositório da disciplina de Introdução ao uso de dados geoespaciais no R'. It includes links to the README and a GitHub Pages link. There are sections for Releases (none published), Packages (none published), Environments (github-pages active), and Languages (SCSS 24.0%, JavaScript 23.2%, Less 21.8%, HTML 10.6%, CSS 10.4%, R 10.0%).

mauriciovancine / course-geospatial-data-r Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

mauriciovancine update plano ensino, aula 0, aula 1 1dc6e3a 2 days ago 85 commits

00_plano_ensino update plano ensino, aula 0, aula 1 2 days ago

01_slides update plano ensino, aula 0, aula 1 2 days ago

02_scripts 2020-03-08 7 months ago

03_dados 2021-10-07 2 days ago

.gitignore 2020-10-07 12 months ago

.here 2020-10-10 12 months ago

README.md 2021-03-16 7 months ago

_config.yml Update _config.yml 7 months ago

add.md 2021-01-15 9 months ago

course-geospatial-data-r.Rproj 2020-03-08 7 months ago

lista_ouvintes.md 2021-10-07 2 days ago

r_logo_spatial.png 2021-10-07 2 days ago

README.md

Introdução ao uso de dados geoespaciais no R

About

Repositório da disciplina de Introdução ao uso de dados geoespaciais no R

mauriciovancine.github.io/course-g...

Readme

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Environments 1

github-pages Active

Languages

SCSS 24.0% JavaScript 23.2%

Less 21.8% HTML 10.6%

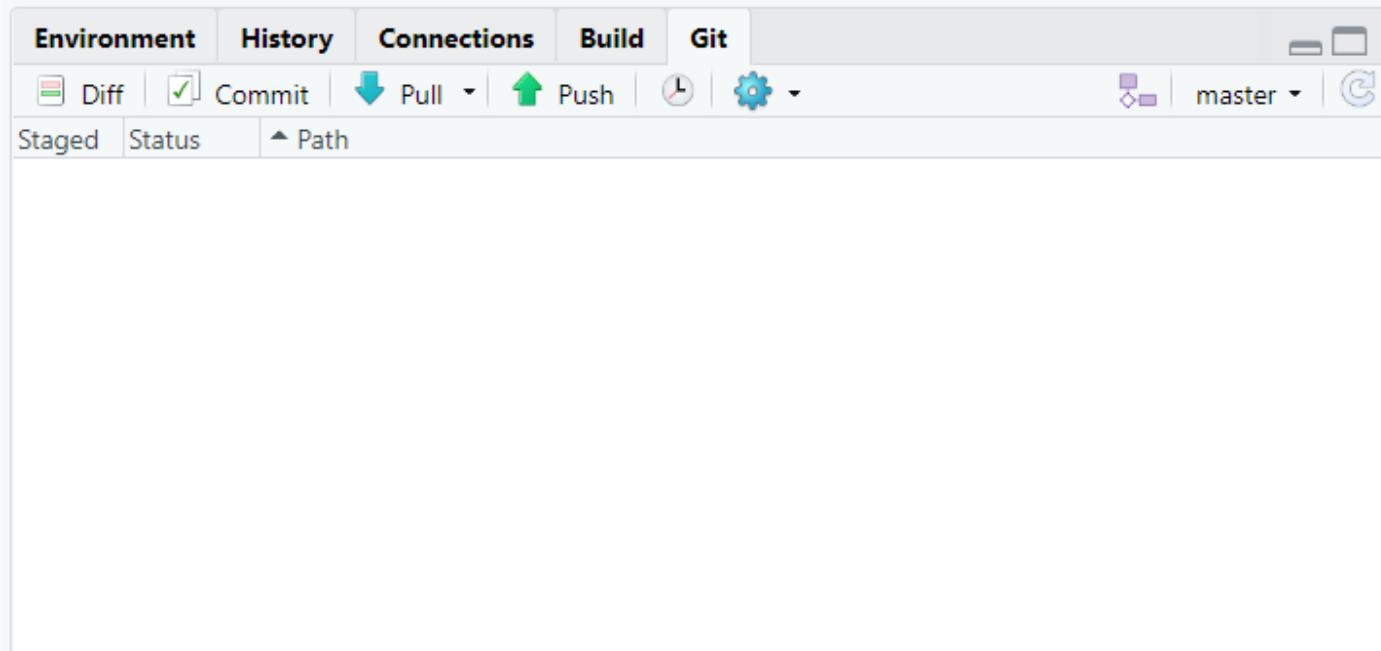
CSS 10.4% R 10.0%

Agradecimento à Beatriz Milz pelas figuras ~~surrepticiadas~~
(eu pedi antes =])

18. Interface Gráfica do RStudio

Git Panel

- RStudio tem um **cliente Git** na aba "Git"
- Esse painel aparece em **projetos** que estejam **versionados com git**

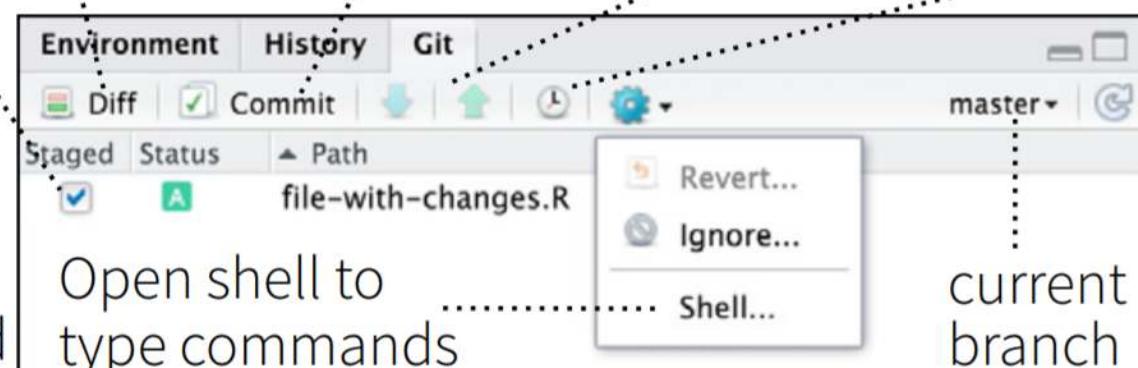


18. Interface Gráfica do RStudio

Git Panel - Detalhes

Version Control with Git or SVN

Turn on at **Tools > Project Options > Git/SVN**



The screenshot shows the RStudio interface with the 'Git' tab selected in the top navigation bar. Below it, a dropdown menu is open over a file named 'file-with-changes.R'. The menu items are 'Revert...', 'Ignore...', and 'Shell...'. A tooltip 'Open shell to type commands' points to the 'Shell...' option.

Stage files: 

Show file diff

Commit staged files

Push/Pull to remote

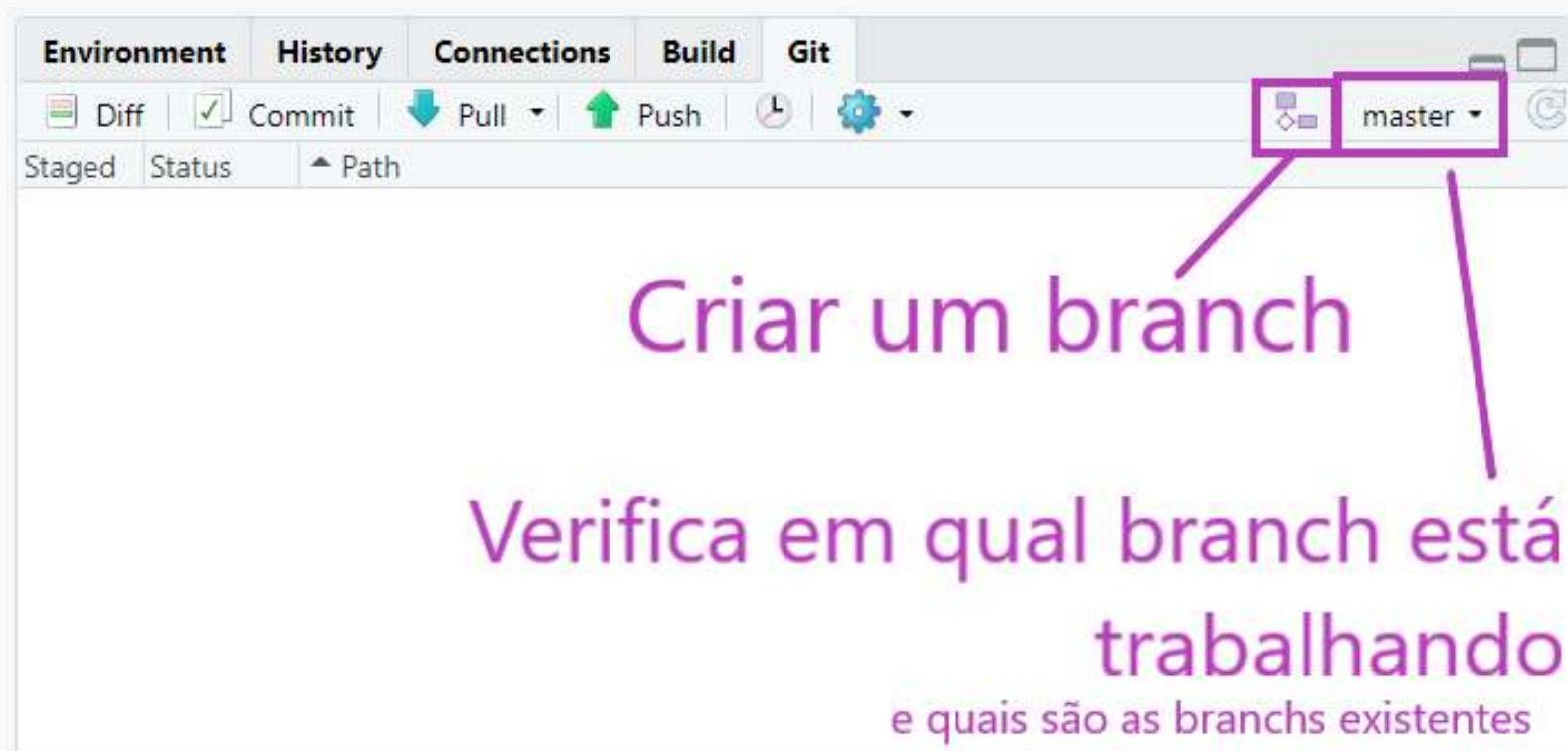
View History

Legend:

- A** Added
- D** Deleted
- M** Modified
- R** Renamed
- ?** Untracked

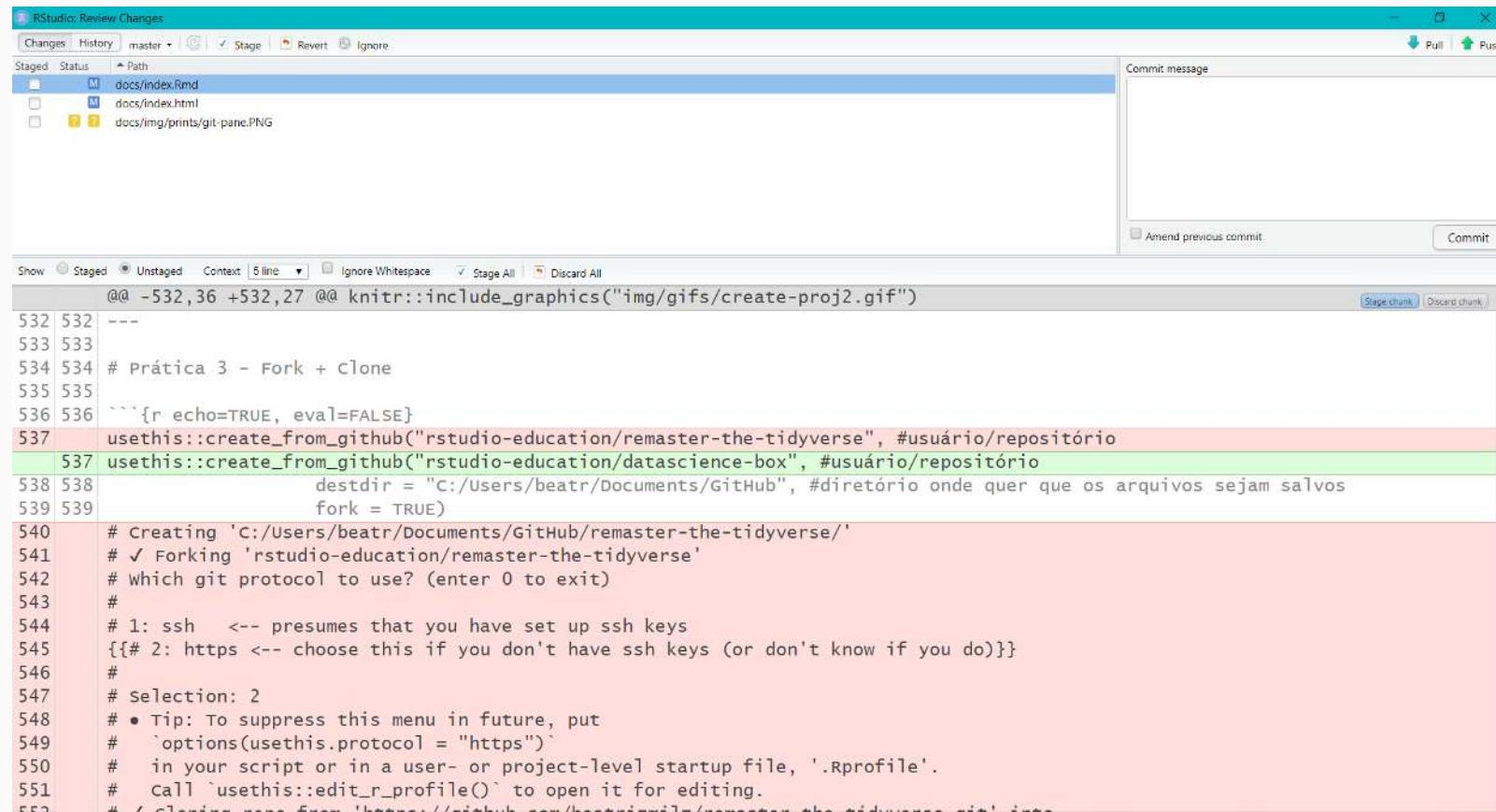
18. Interface Gráfica do RStudio

Git Panel - Branches



18. Interface Gráfica do RStudio

Git Panel - Diff - Changes: Revisar mudanças



18. Interface Gráfica do RStudio

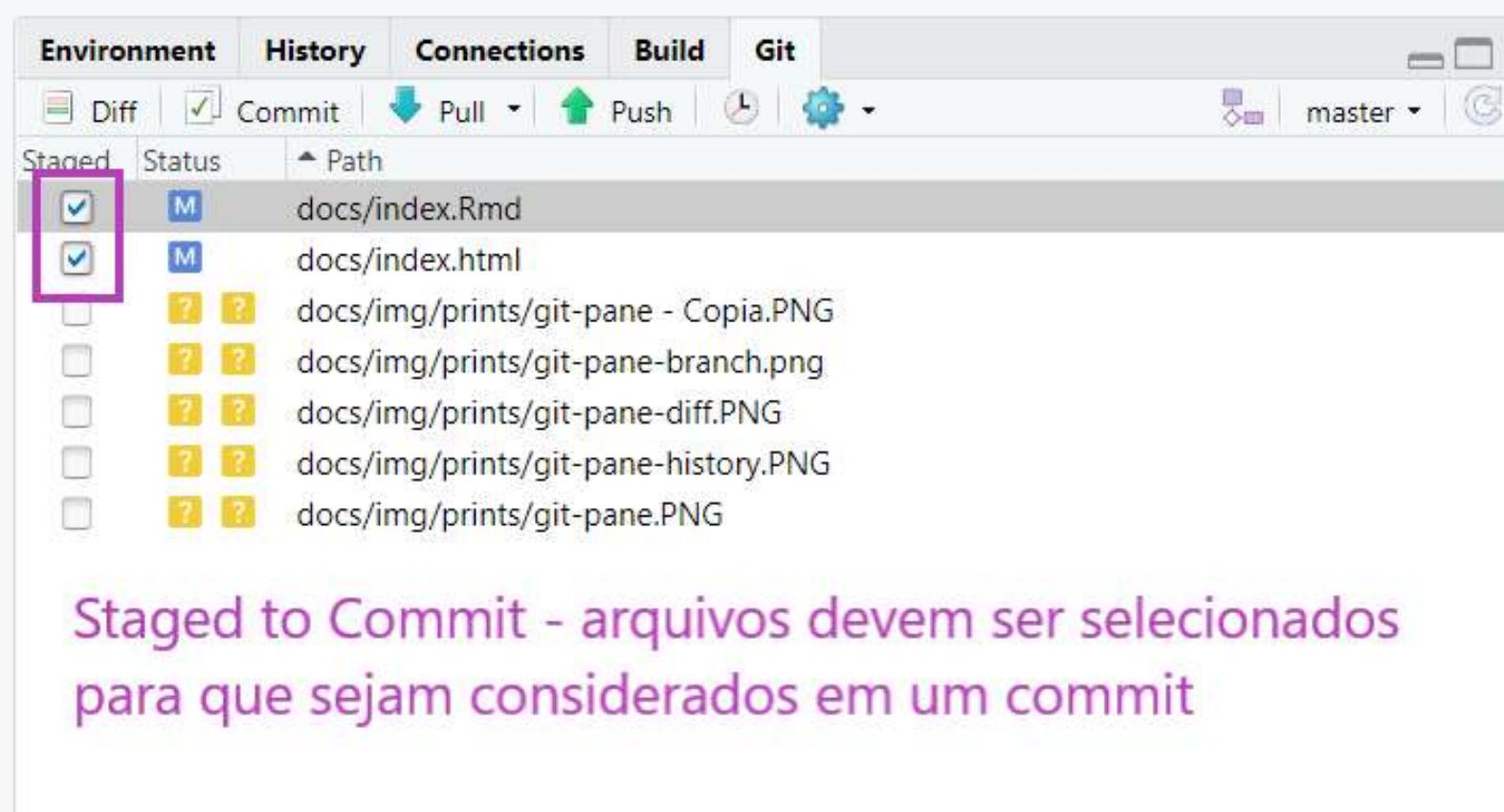
Git Panel - Diff - History: Histórico de mudanças

The screenshot shows the RStudio interface with the Git Panel open. The History tab is selected, displaying a list of commits for the master branch. The most recent commit is highlighted, showing its details: SHA ab5bd162, Author Beatriz Milz <beatriz.milz@hotmail.com>, Date 2019-08-29 22:59, Subject adiciona exemplos, muda títulos, and Parent 0449805c. Below this, the Diff tab is active, showing the changes made in the file docs/index.Rmd. The changes are color-coded: red for deletions and green for additions. The code in the diff view is as follows:

```
@@ -96,10 +96,10 @@ class: split-33 with-border
96 96 ## Pré-requisitos
97 97 ]]
98 98 .row.bg-main2[.content[
99 99 ## Configurando o `Git`
100 100 ]]
101 101 .row.bg-main3[.content[
102 102 ## Configurando o `GitHub`
103 103 ]]
104 104 .row.bg-main4[.content[
105 105 ## Trabalhando com projetos no `Rstudio` + `GitHub`
@@ -129,7 +129,7 @@ class: middle
129 129 # Pré-requisitos
130 130
```

18. Interface Gráfica do RStudio

Git pane - Staged



The screenshot shows the RStudio interface with the 'Git' tab selected in the top navigation bar. Below the navigation bar is a toolbar with icons for Diff, Commit, Pull, Push, and a gear icon. To the right of the toolbar is a dropdown menu showing 'master'. The main area is the 'Git pane' titled 'Staged'. It displays a list of files with their status: 'M' for modified, 'A' for added, 'D' for deleted, and '?' for untracked. The first two files, 'docs/index.Rmd' and 'docs/index.html', have a checked checkbox next to them, indicating they are staged for commit. A purple box highlights these two staged files.

Staged	Status	Path
<input checked="" type="checkbox"/>	M	docs/index.Rmd
<input checked="" type="checkbox"/>	M	docs/index.html
<input type="checkbox"/>	?	docs/img/prints/git-pane - Copia.PNG
<input type="checkbox"/>	?	docs/img/prints/git-pane-branch.png
<input type="checkbox"/>	?	docs/img/prints/git-pane-diff.PNG
<input type="checkbox"/>	?	docs/img/prints/git-pane-history.PNG
<input type="checkbox"/>	?	docs/img/prints/git-pane.PNG

Staged to Commit - arquivos devem ser selecionados para que sejam considerados em um commit

18. Interface Gráfica do RStudio

Git pane - Staged e File status

The screenshot shows the RStudio interface with the 'Git' tab selected in the top navigation bar. Below the tabs, there are buttons for 'Diff', 'Commit', 'Pull', 'Push', and a gear icon. The main area displays a list of files with their status: 'Staged' or 'Status' (indicated by a small icon) and the file path. Two files, 'docs/index.Rmd' and 'docs/index.html', are highlighted with a purple border and have a checkmark icon in their 'Staged' column, indicating they are ready to be committed. To the right of the list is a legend mapping icons to git operations:

Icon	Description
A	File Added
D	File Deleted
M	File Modified
R	File Renamed
?	File Untracked by Git

Staged to Commit - arquivos devem ser selecionados para que sejam considerados em um commit

18. Interface Gráfica do RStudio

Git pane - Commit

The screenshot shows the RStudio interface with the 'Review Changes' window open. The left pane displays a tree view of files under the 'Staged' tab, with several files highlighted in blue. The right pane shows a commit message input field containing the text: 'adiciona slides sobre o git pane'. Below the message is a note: 'Commit message: adicione uma mensagem que seja útil para entender o que o commit faz'. At the bottom of the pane are buttons for 'Amend previous commit' and 'Commit'. The bottom section of the window shows a diff view with code changes, where new lines are green and deleted lines are red.

Staged files

Commit message:
adiciona slides sobre o git pane
Commit message: adicione uma mensagem que seja útil para entender o que o commit faz

Commit

@@ -610,10 +610,20 @@ class: middle

610 # Botão `r emoji::ji("clock3")` (history) -> Review changes

611

612 ```{r, out.width="90%"}
613 knitr::include_graphics("img/prints/git-pane-history.PNG")
614```

615

616

617 ---

618 class: middle

619

620 # Git pane - Staged `r emoji::ji("white_check_mark")`

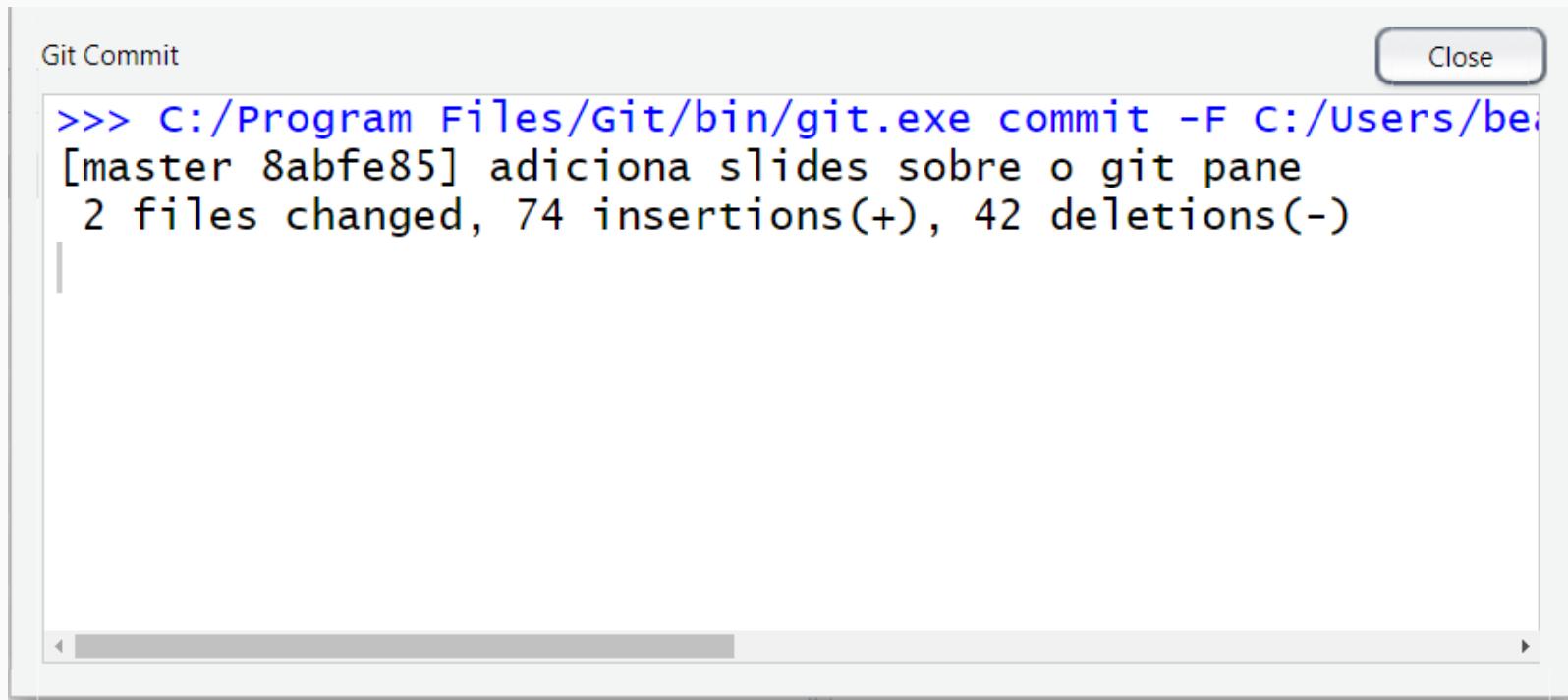
621

622 ```{r}
623 knitr::include_graphics("img/prints/git-pane-stage.png")
624```

Mudanças verificadas: em verde, são novas linhas. em vermelho, são linhas retiradas/alteradas

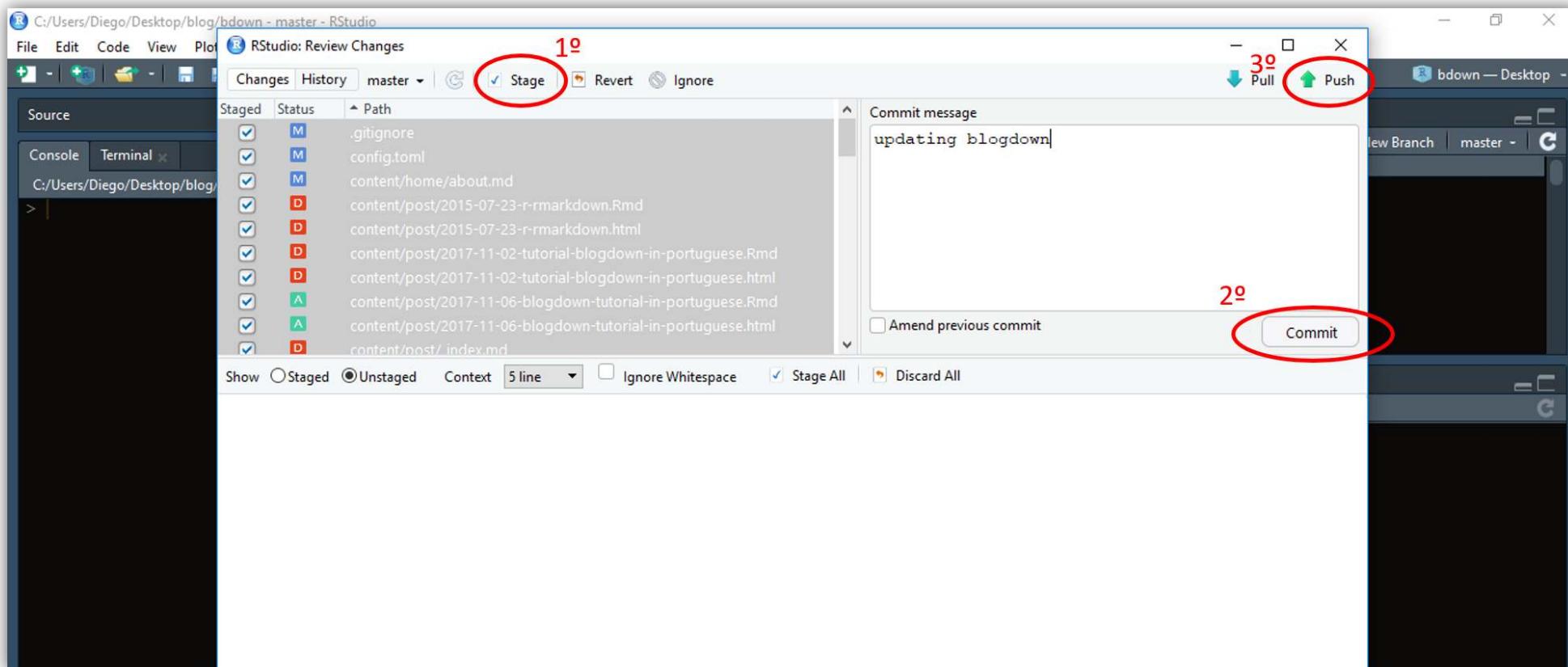
18. Interface Gráfica do RStudio

Git pane - Commit



18. Interface Gráfica do RStudio

Git pane - Push e Pull



19. Principal material de estudo

Git Cheat Sheets

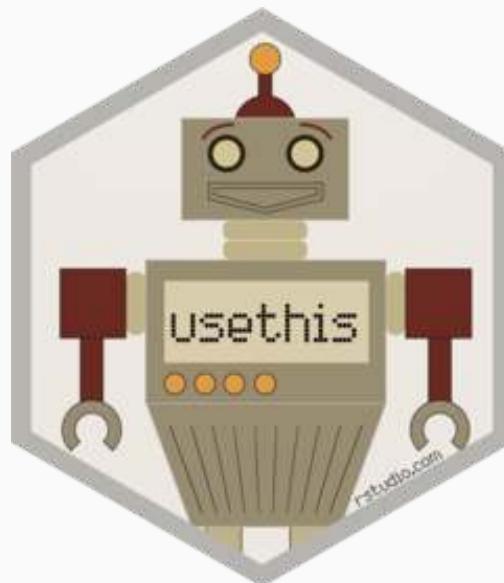


19. Principal material de estudo

Pacote usethis

Comandos direto no R para criar e versionar repositórios

```
install.packages("usethis")
devtools :: install_github("r-lib/usethis")
```



19. Principal material de estudo

Livros

[Pro Git](#) - Scott Chacon e Ben Straub

[Beginning Git and GitHub](#) - Mariot Tsitoara

[Happy Git and GitHub for the useR](#) - Jenny Bryan

[Zen do R](#) - Caio Lente + Curso-R

19. Principal material de estudo

Material

[Primeiros passos utilizando o Git e GitHub no RStudio](#) - Beatriz Milz

[Git e GitHub no RStudio](#) - Beatriz Milz

[Torne-se um guru do git](#)

[RStudio e Github no dia a dia](#)

19. Principal material de estudo

Material

[Oh Shit, Git!?! - Katie Sylor-Miller](#)

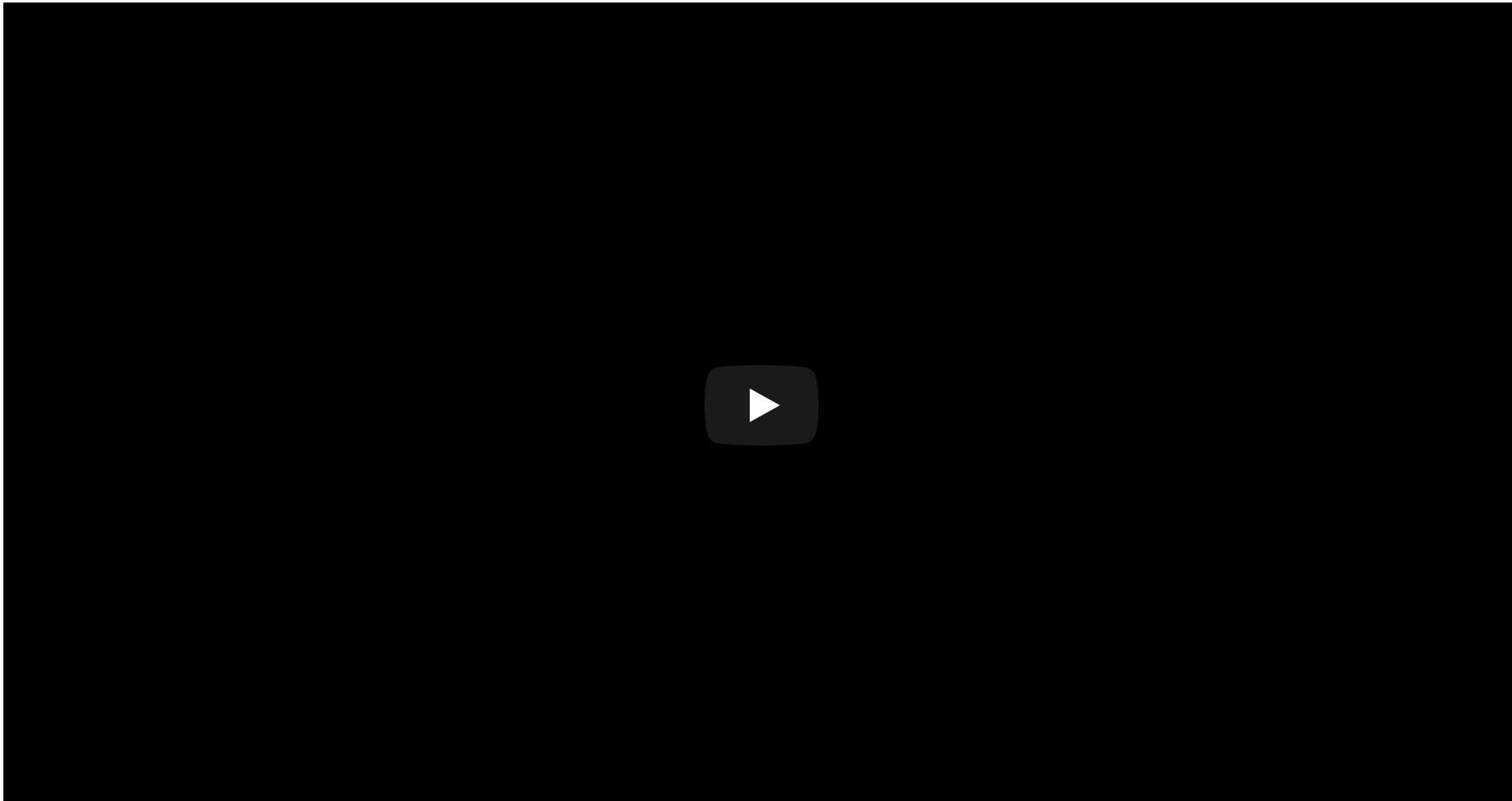
[Dangit, Git!?! - Katie Sylor-Miller](#)

[Version Control with GitHub](#)

[Git Cheat Sheet – 50 Git Commands You Should Know](#)

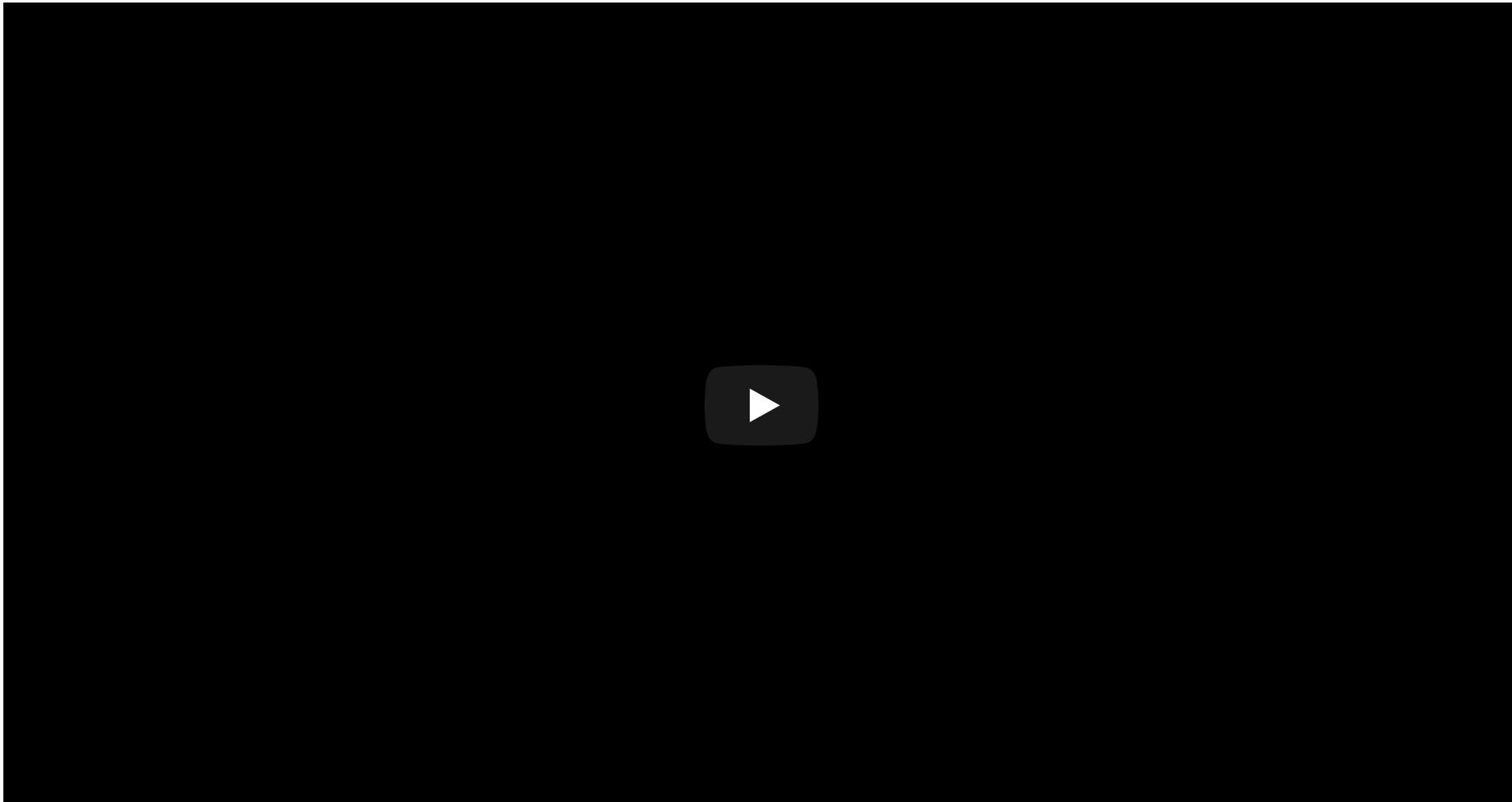
19. Principal material de estudo

Git e Github para iniciantes - Willian Justen de Vasconcellos



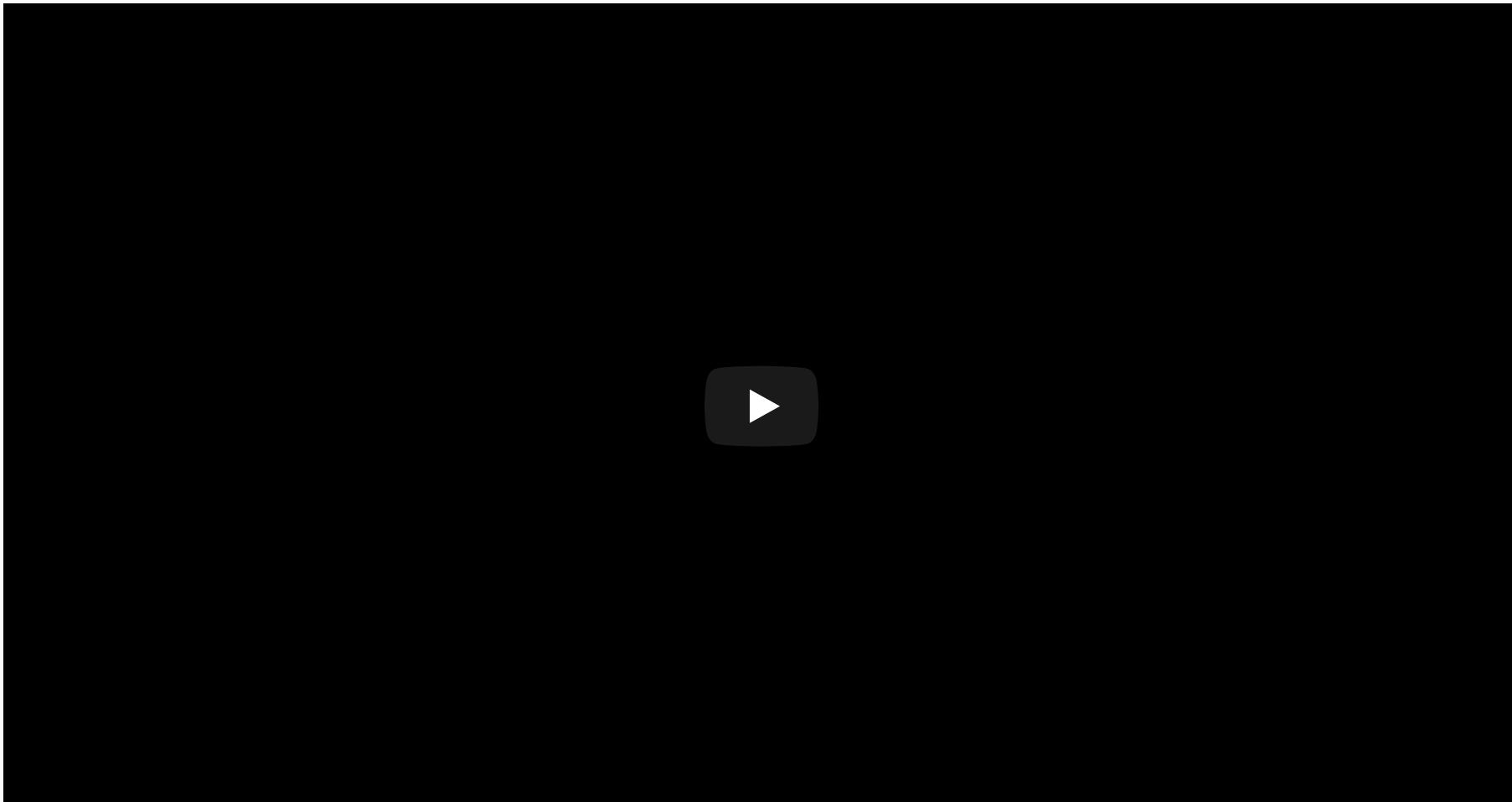
19. Principal material de estudo

Curso de Git e GitHub - Curso em Vídeo



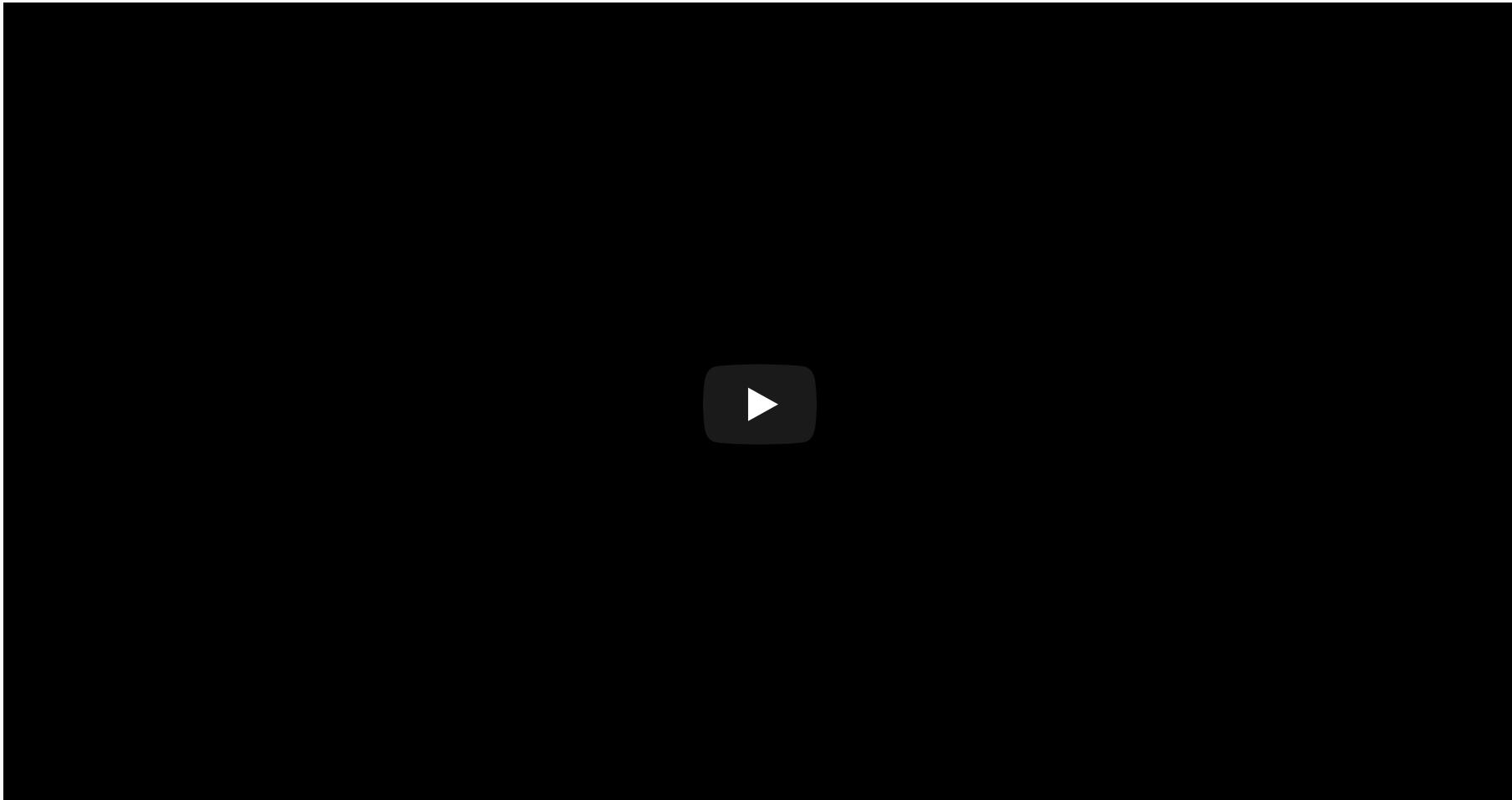
19. Principal material de estudo

Curso de Git - Professor Aquino



19. Principal material de estudo

Curso de Git - Bóson Treinamentos



Dúvidas?

Maurício Vancine

Contatos:

✉ mauricio.vancine@gmail.com

🐦 [@mauriciovancine](https://twitter.com/mauriciovancine)

🐙 [mauriciovancine](https://github.com/mauriciovancine)

🔗 mauriciovancine.github.io



Slides criados via pacote [xaringan](#) e tema [Metropolis](#). Animação dos sapos por [@probzz](#).