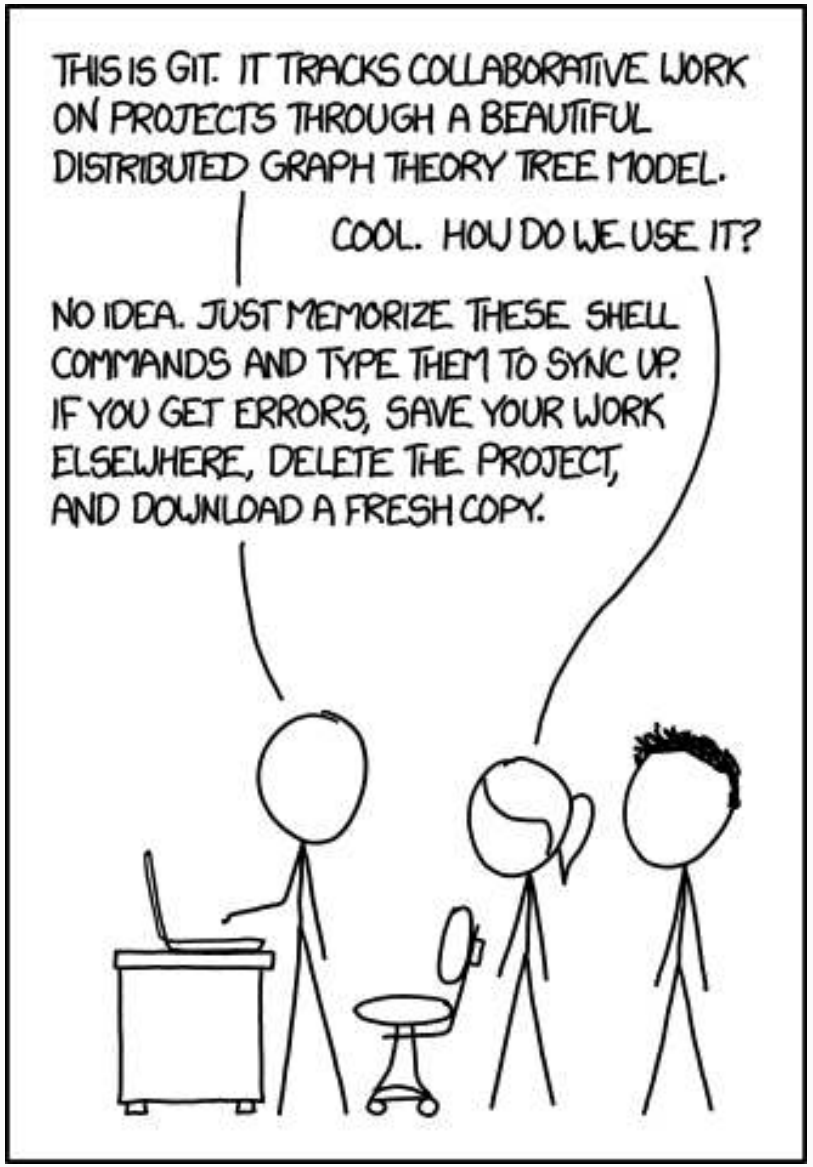
Introdução ao uso de dados geoespaciais no R

1 Controle de versão, git e GitHub

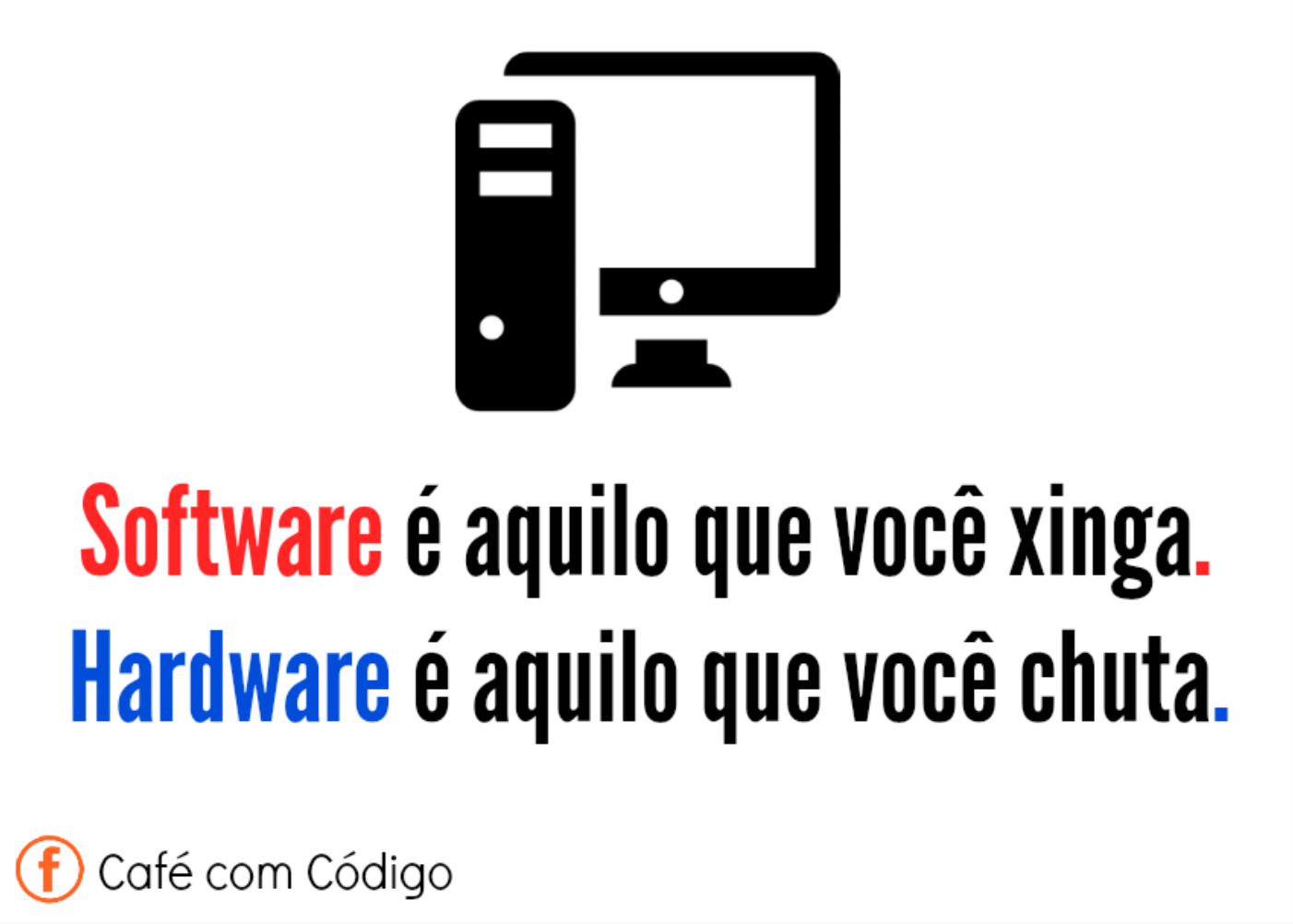
Maurício H. Vancine
Milton C. Ribeiro
UNESP - Rio Claro
Laboratório de Ecologia Espacial e Conservação (LEEC)
25/10/2021-05/11/2021


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 MEMORIZE 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



1. Conferir os computadores

R (4.1.1)

R

1. Conferir os computadores

RStudio (2021.09.0-351)

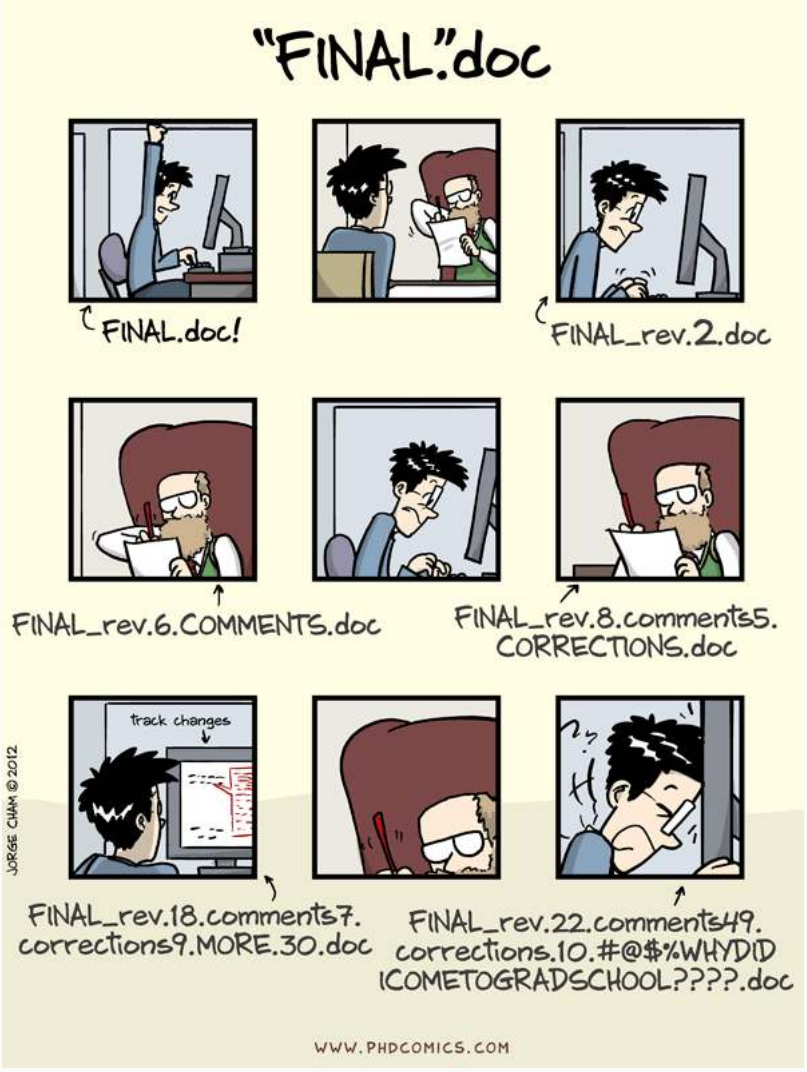
RStudio

1. Conferir os computadores

git (2.33)

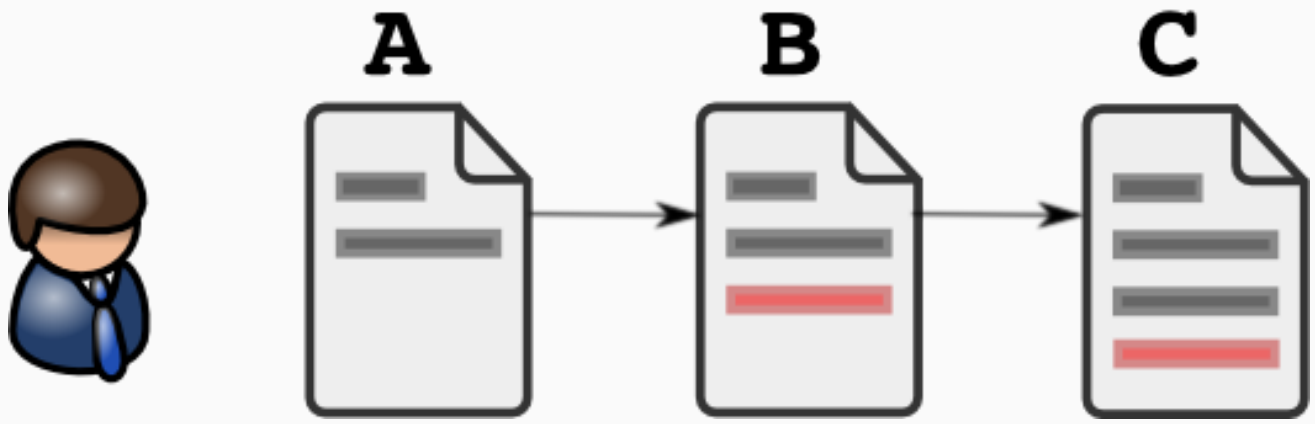
Tudo instalado? Então bora!

2. Controle de versão



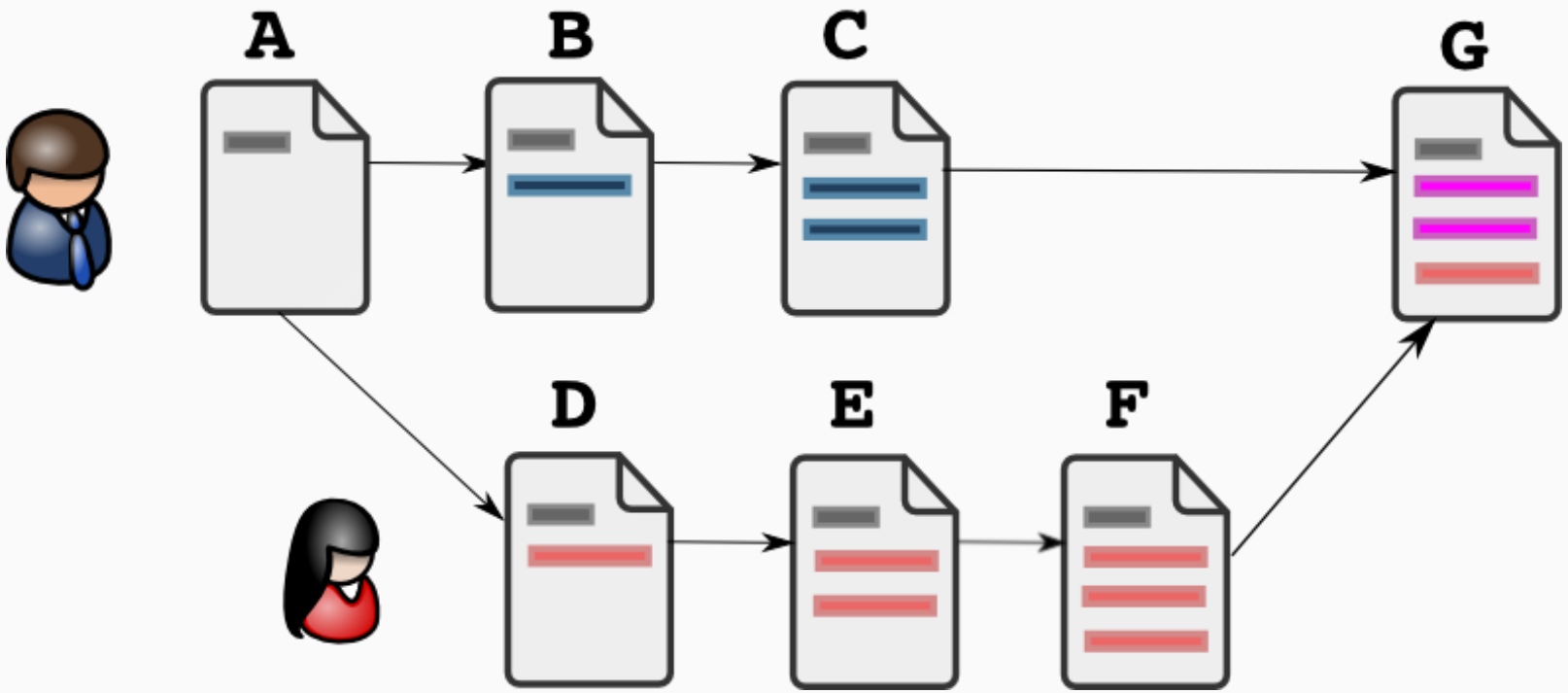
2. Controle de versão

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



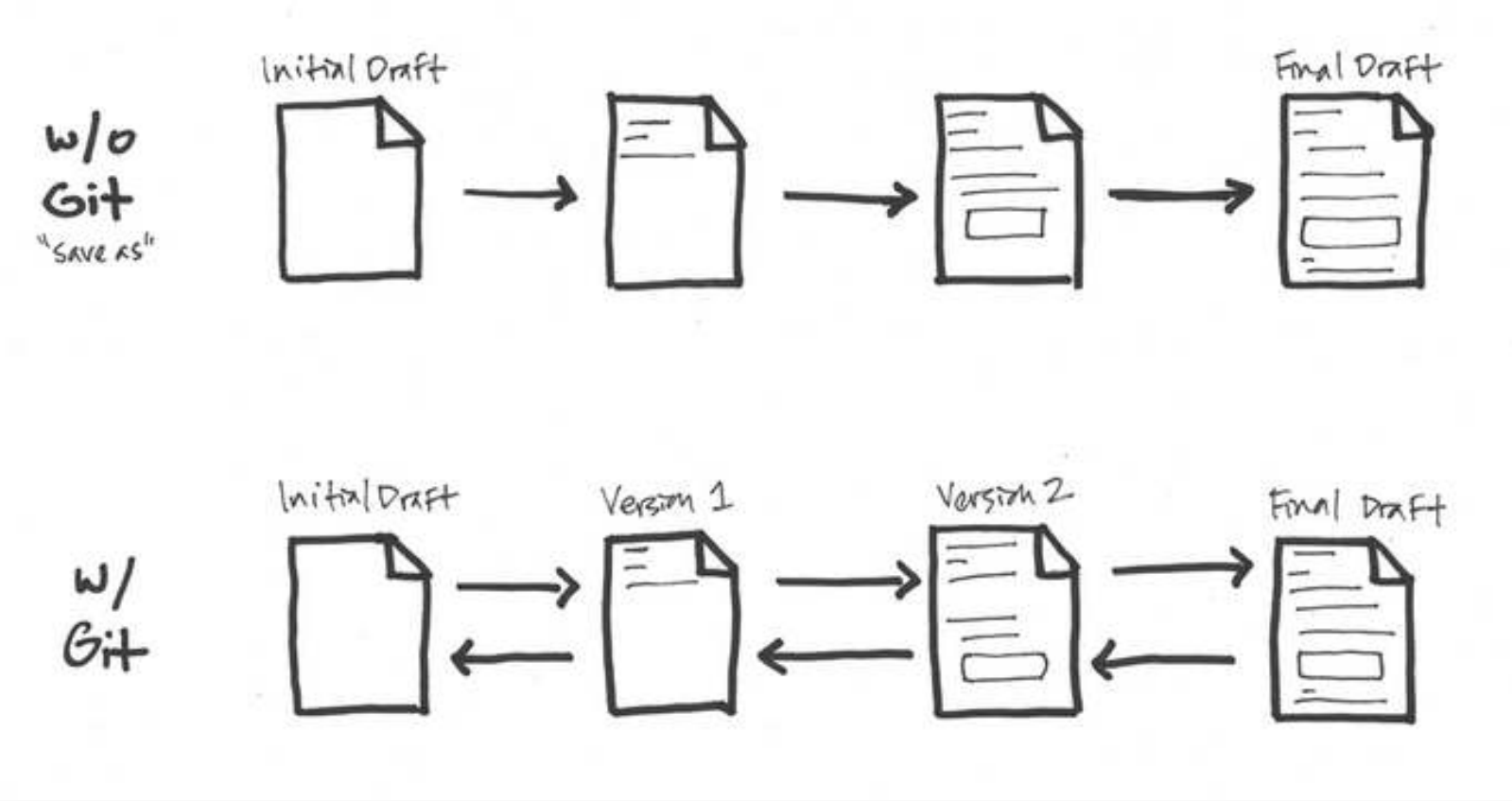
2. Controle de versão

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

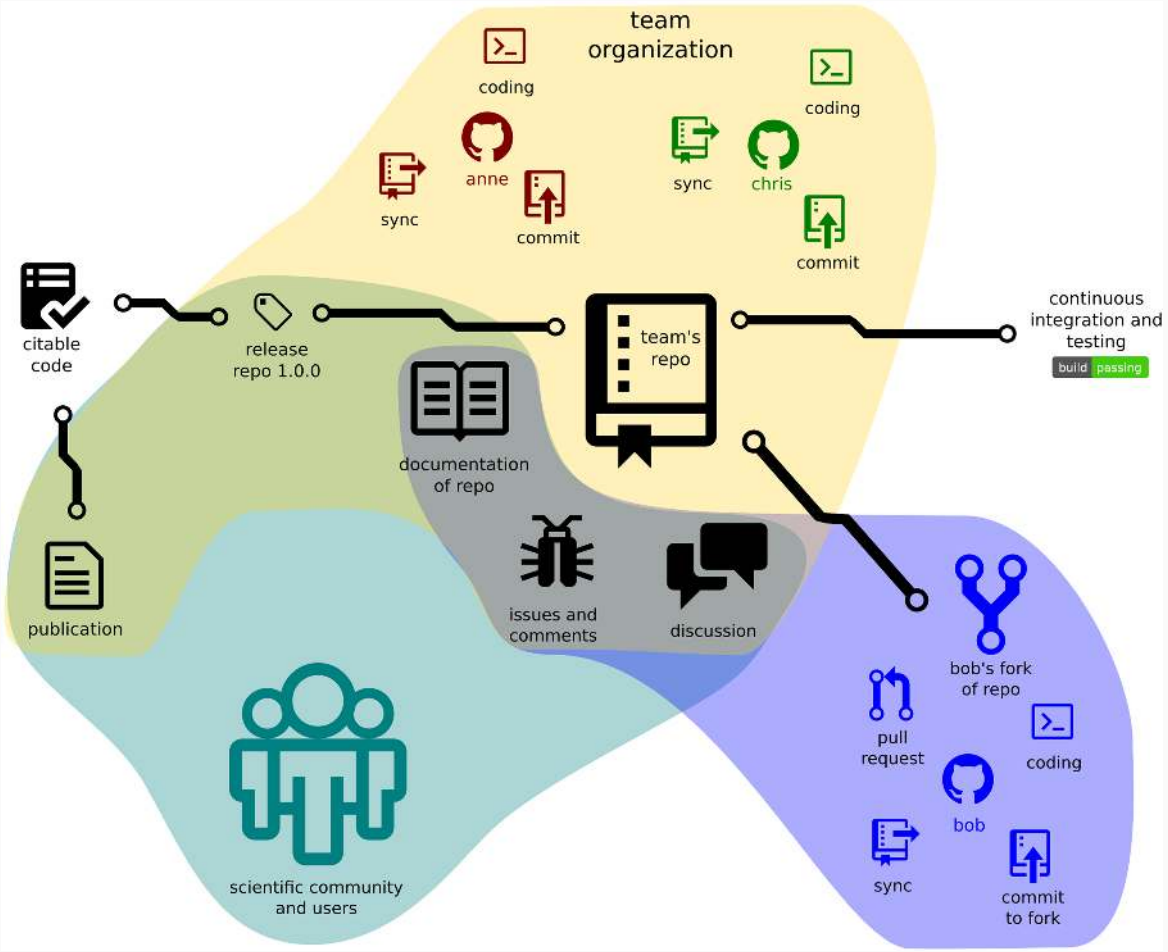
EDITORIAL

Ten Simple Rules for Taking Advantage of Git and GitHub

Yasset Perez-Riverol[1]*, Laurent Gatto[2], Rui Wang[1], Timo Sachsenberg[3], Julian Uszkoreit[4], Felipe da Veiga Leprevost[5], Christian Fufezan[6], Tobias Ternent[1], Stephen J. Eglen[7], Daniel S. Katz[8], Tom J. Pollard[9], Alexander Konovalov[10], Robert M. Flight[11], Kai Blin[12], 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 GitVersion 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



Perez-Riverol et al. (2016)

3. git e GitHub

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

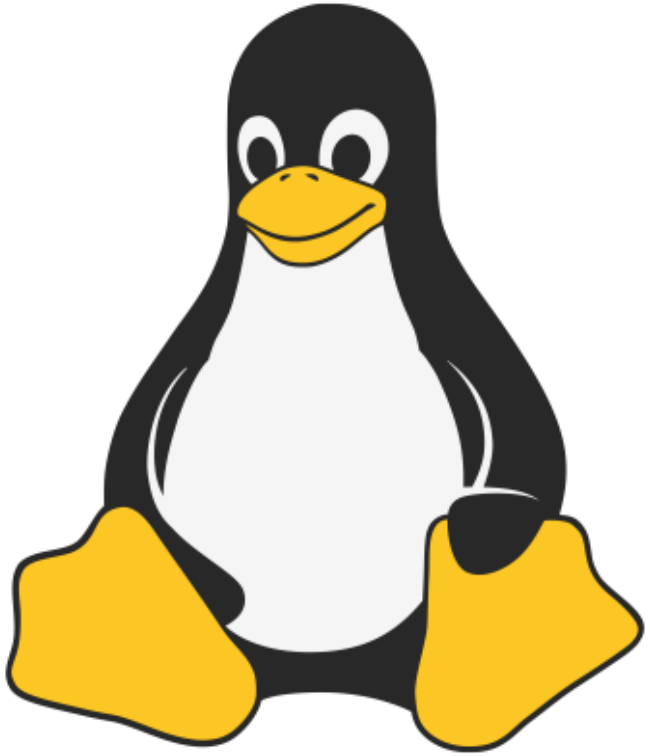
Maneja os **repositórios locais** (computador) e **remotos** (e.g. GitHub)

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



The Linux Kernel Archives

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

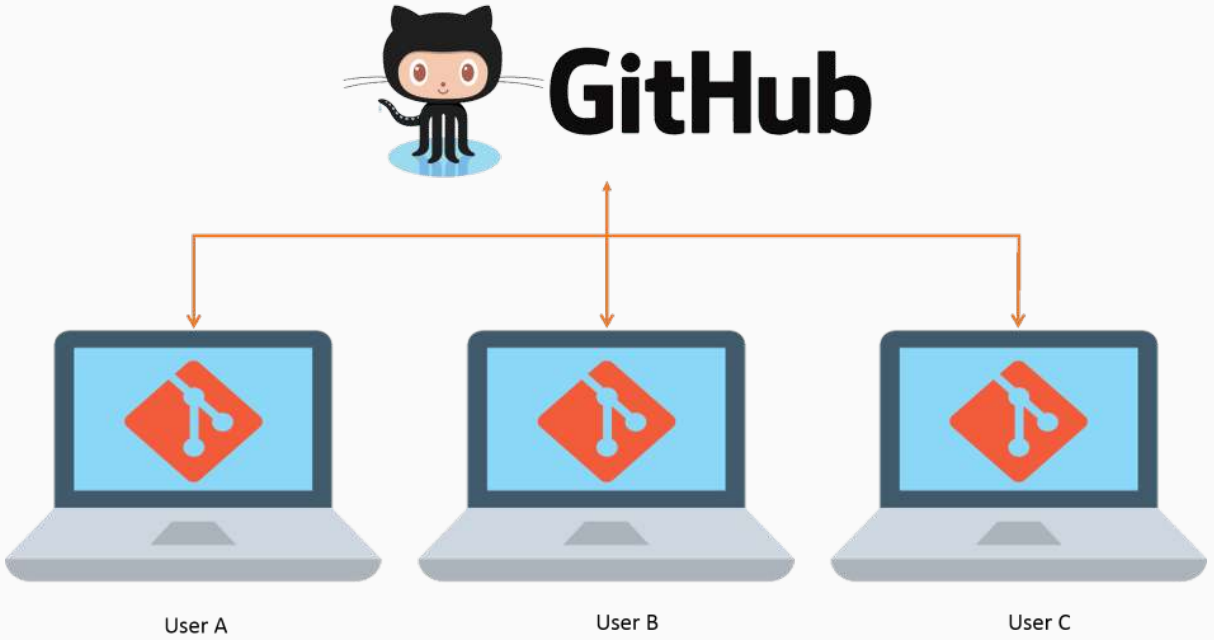
GitHub: repositório remoto que hospeda os arquivos versionados

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)

GitHub

Salvem ou recuperem essas informações!!!

username: mauriciovancine

email: mauricio.vancine@gmail.com

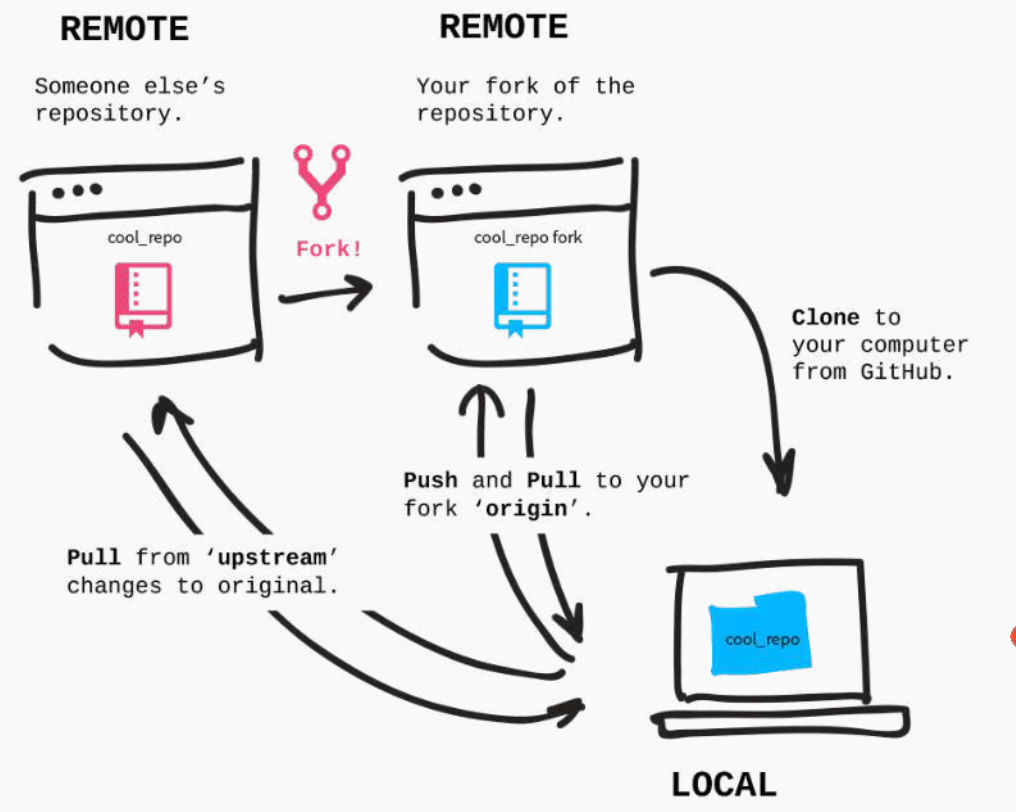
senha: !@#$%¨&*+

Tudo bem até aqui?

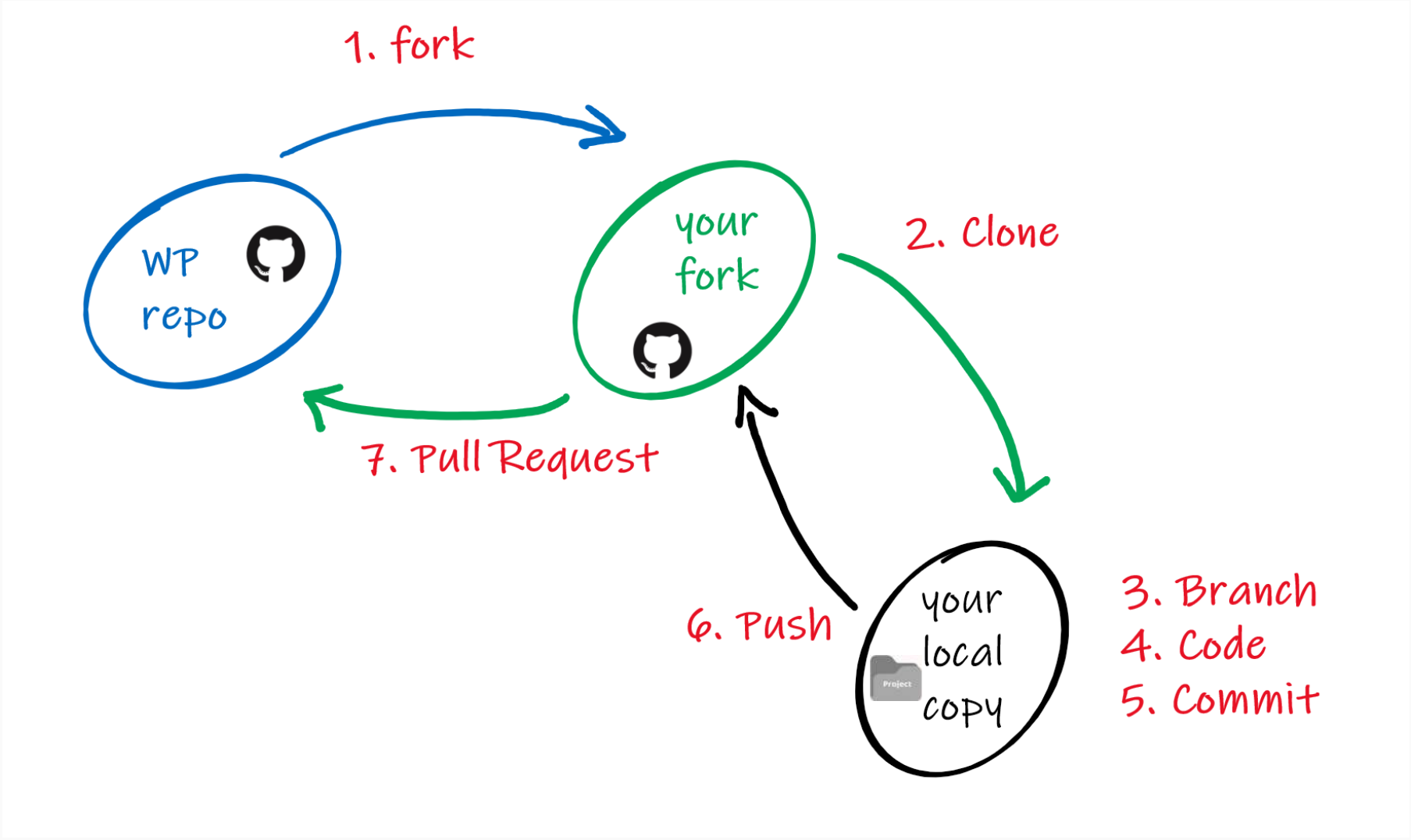
4. Funcionamento do controle de versão

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

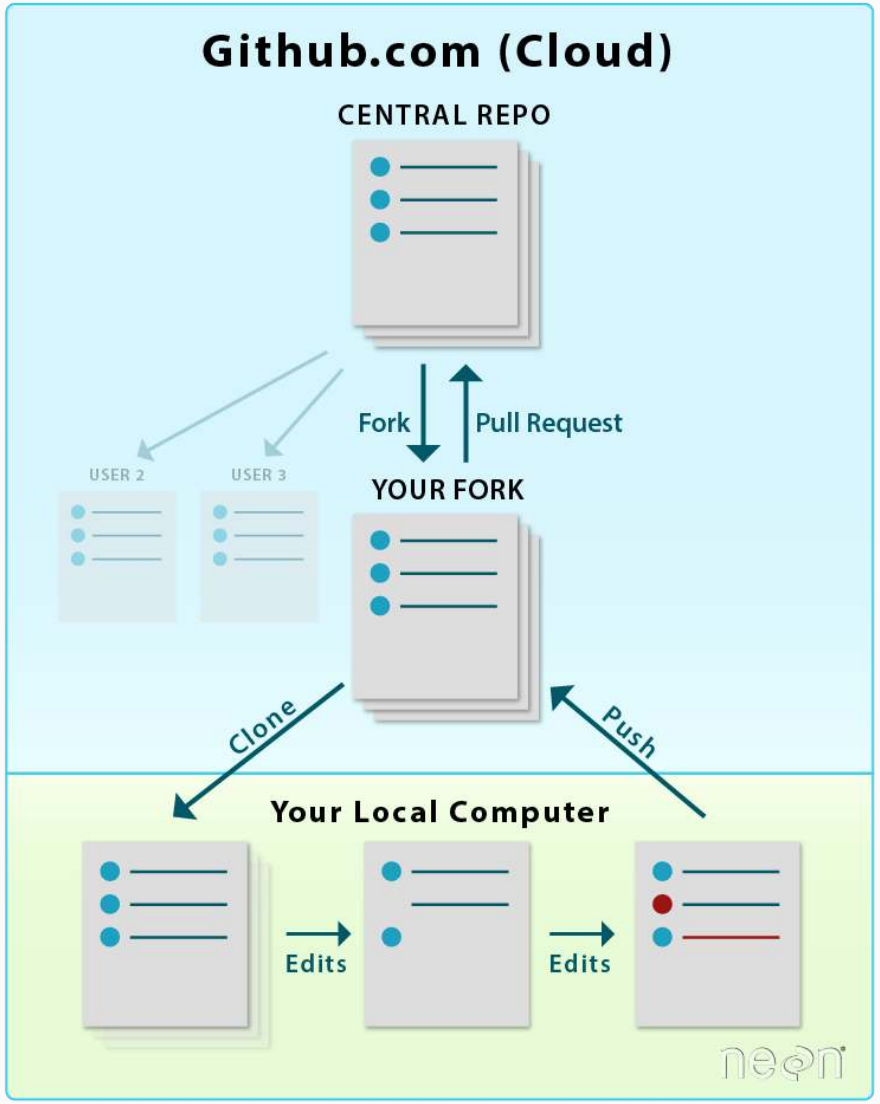
1. Iniciando um projeto em nosso computador (git)
2. Iniciando por um repositório remoto (GitHub) ["forkando" de alguém ou criando um repositório próprio]



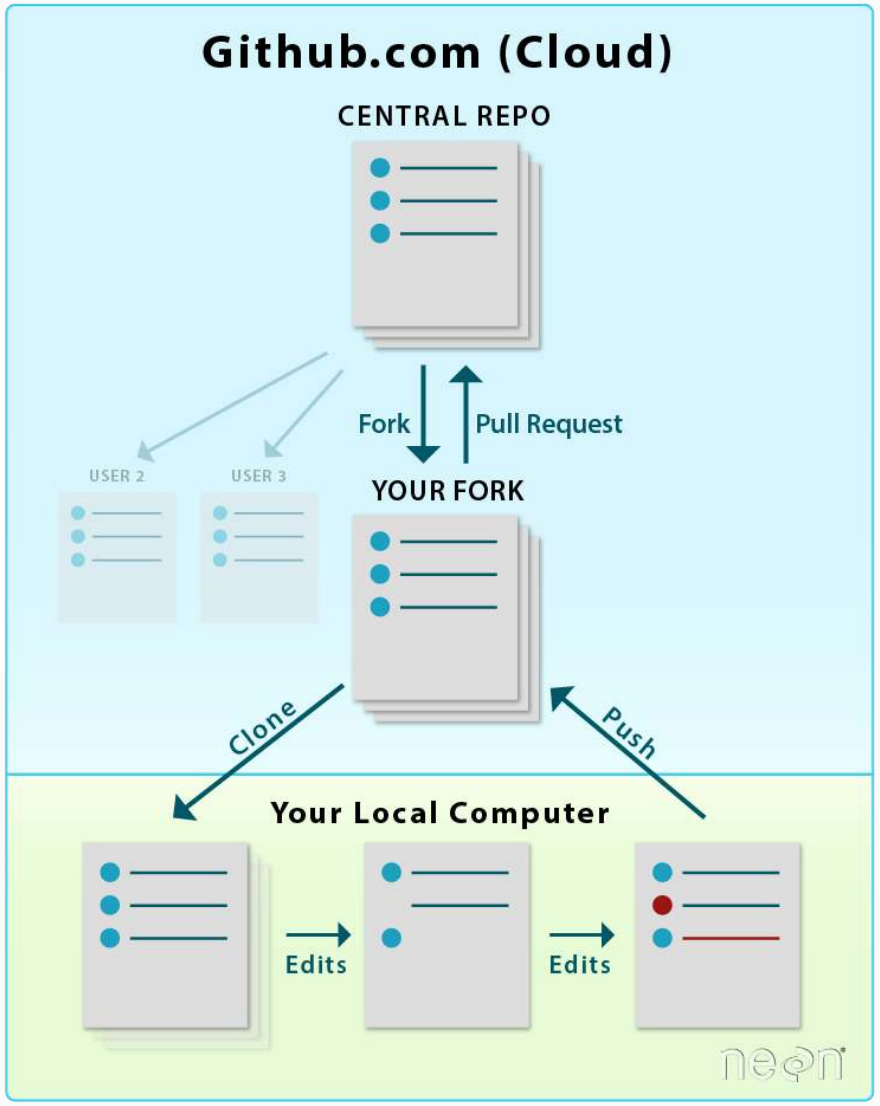
4. Funcionamento do controle de versão



4. Funcionamento do controle de versão

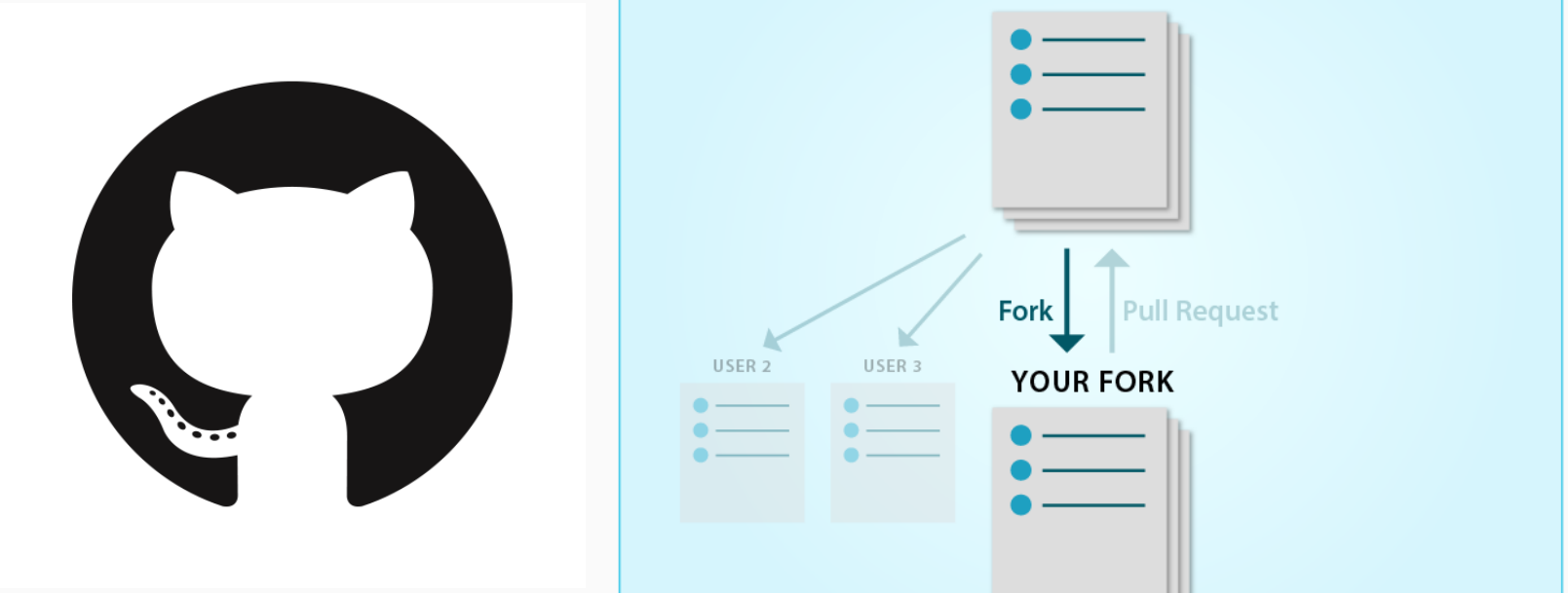


4. Funcionamento do controle de versão



5. GitHub: Fork

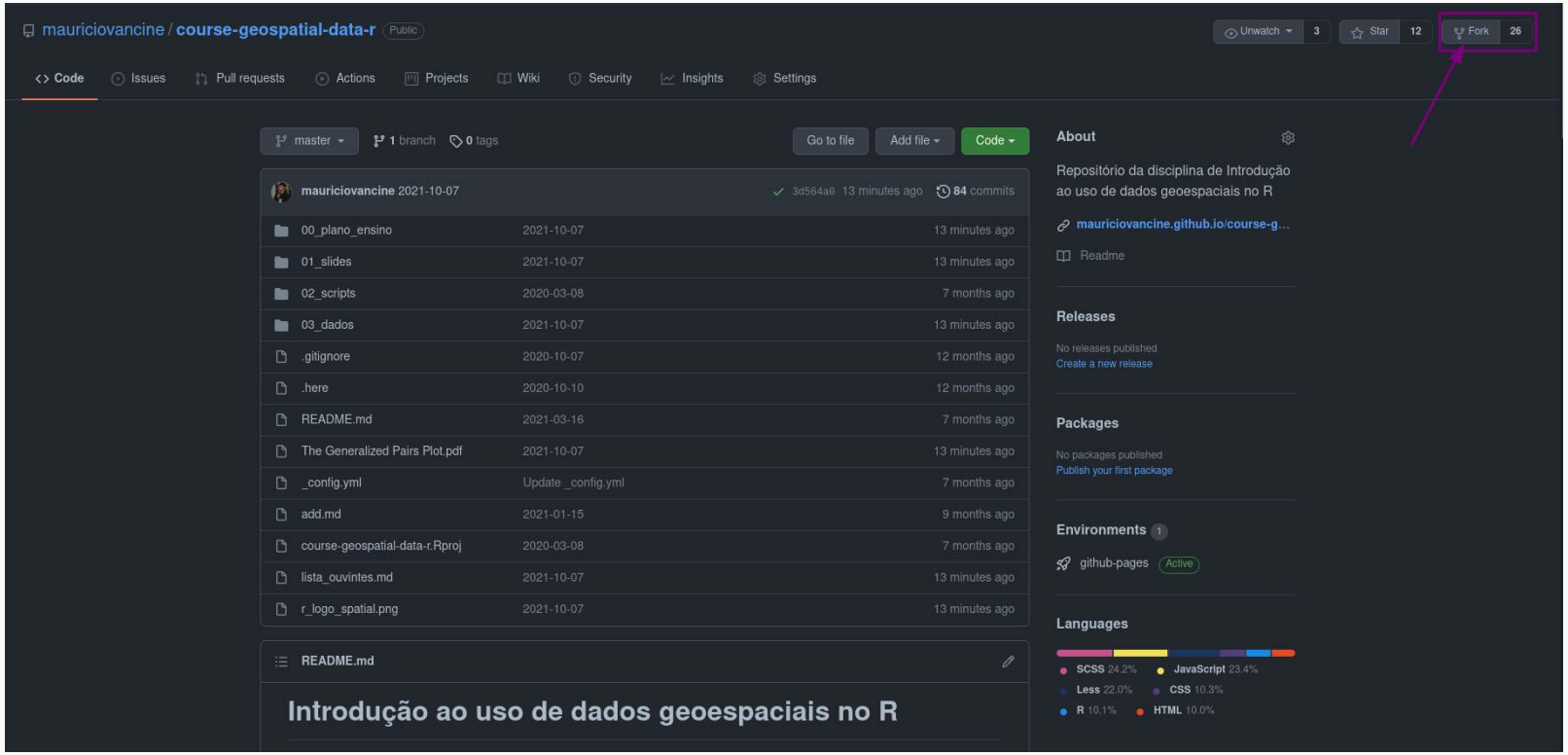
Fork: cópia de um repositório remoto para o seu GitHub



5. GitHub: Fork

Forken o repositório do curso

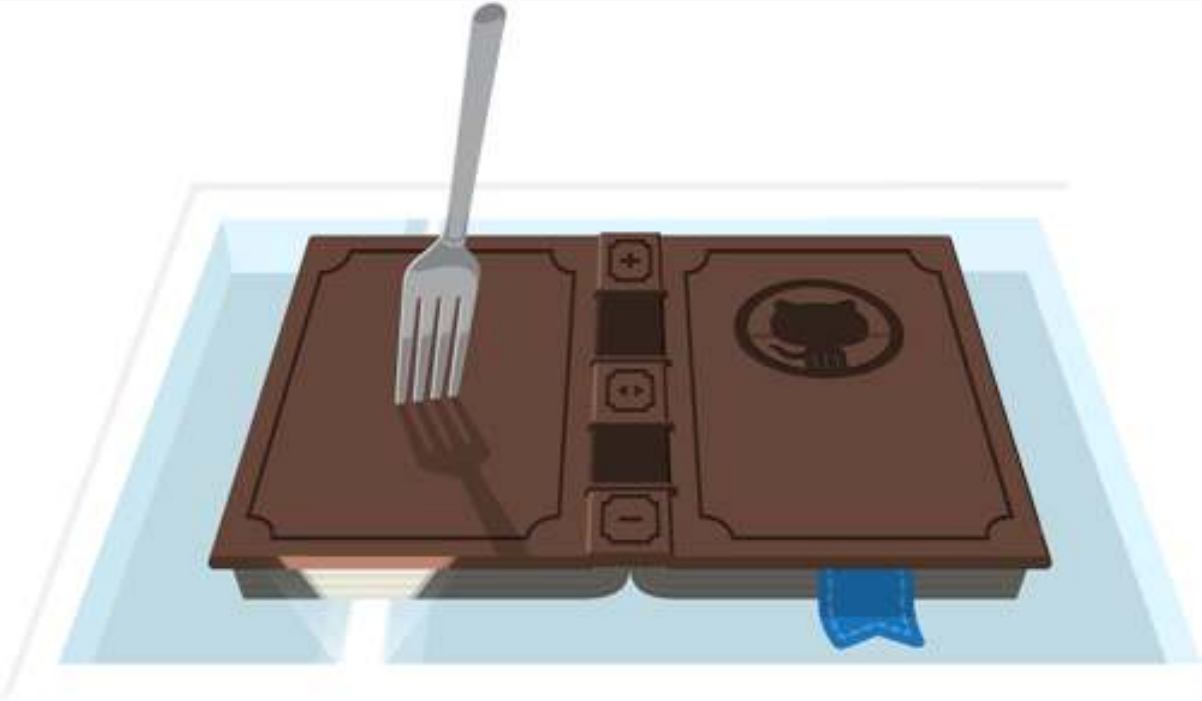
https://github.com/mauriciovancine/course-geospatial-data-r



5. GitHub: Fork

Forken o repositório do curso

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



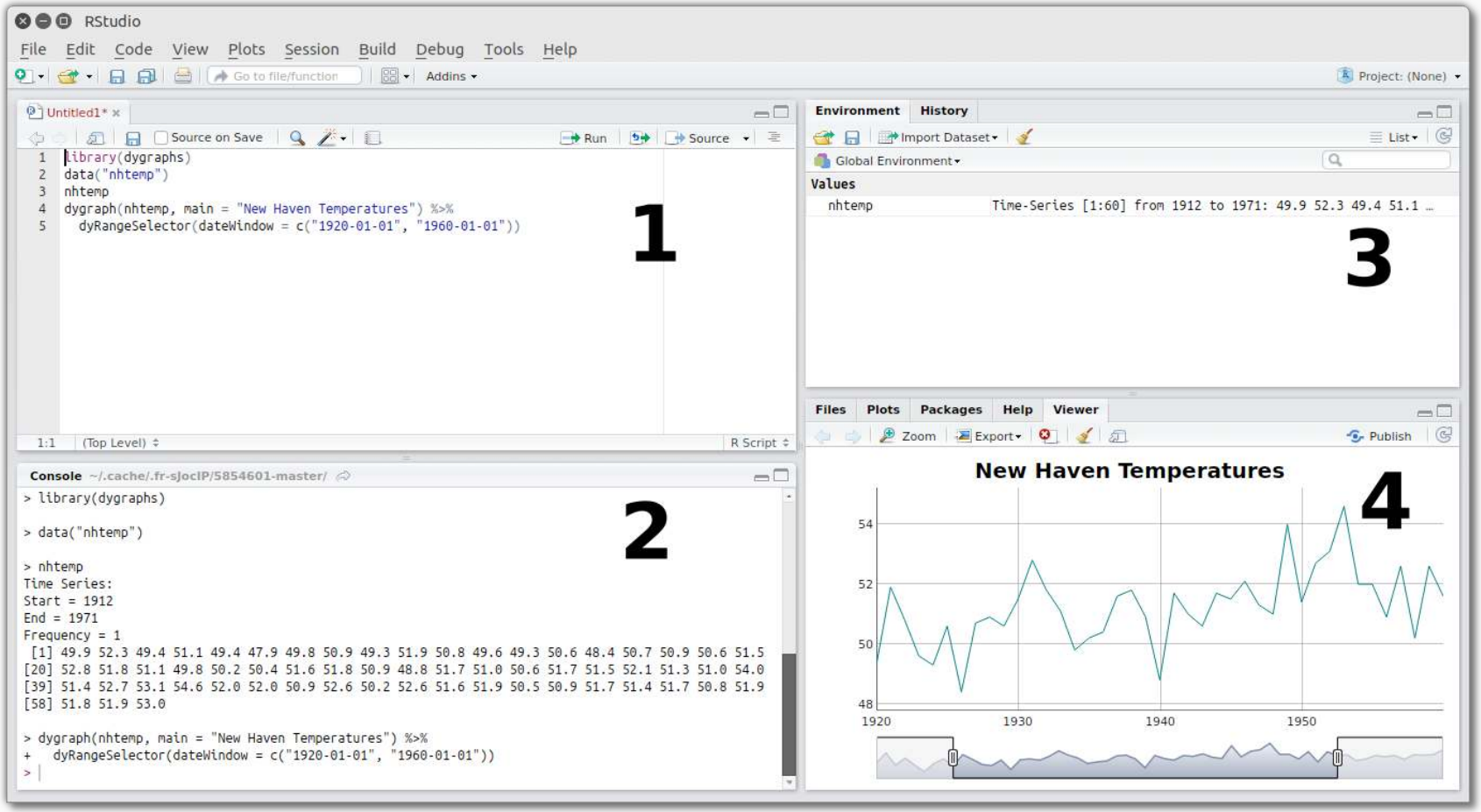
Agora o repositório do curso, que estava no meu GitHub, foi copiado ("garfado") para o GitHub de vocês

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

Vamos abrir o RStudio

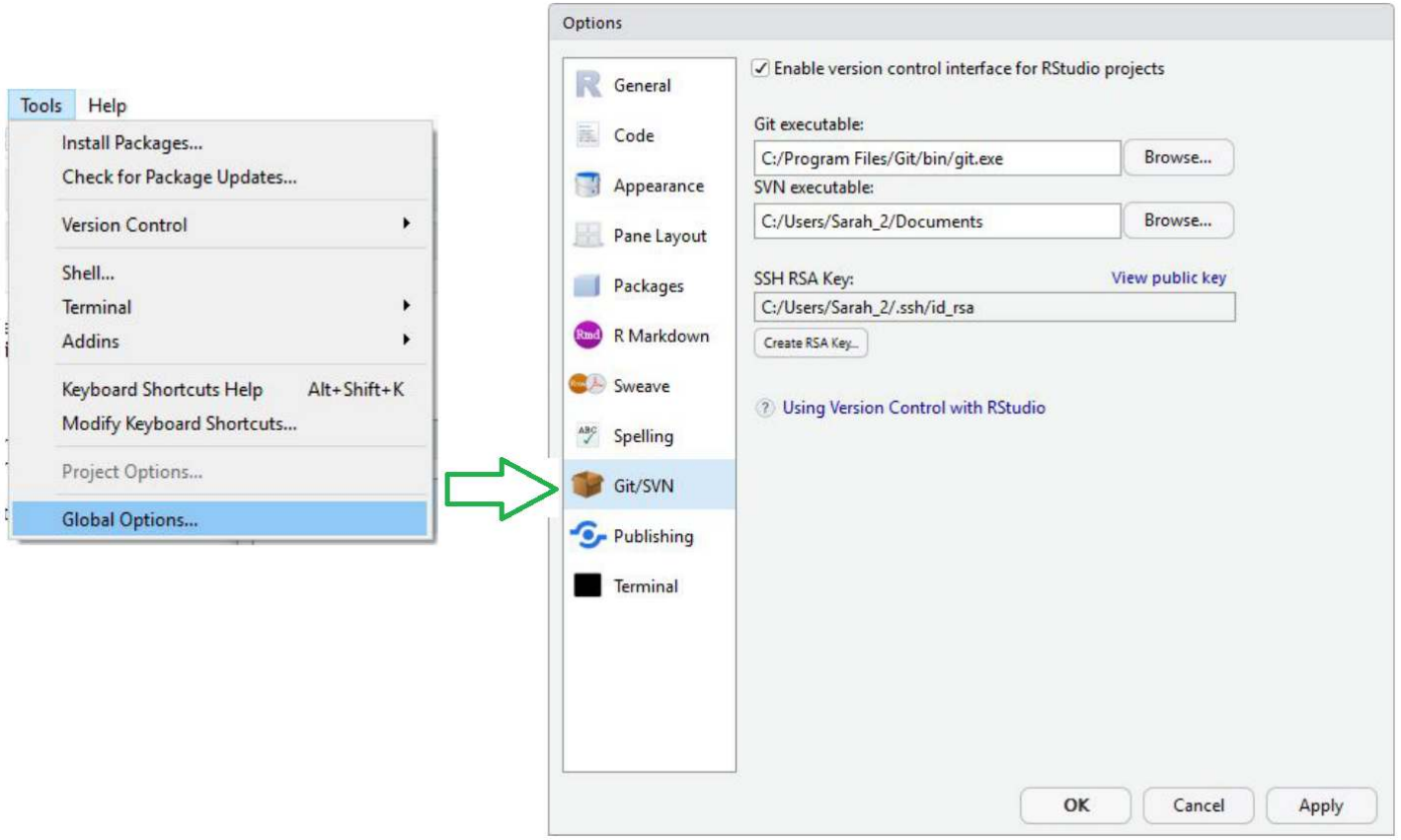
6. Configuração: RStudio, git, GitHub

IDE RStudio



Indicar o caminho do git para o RStudio

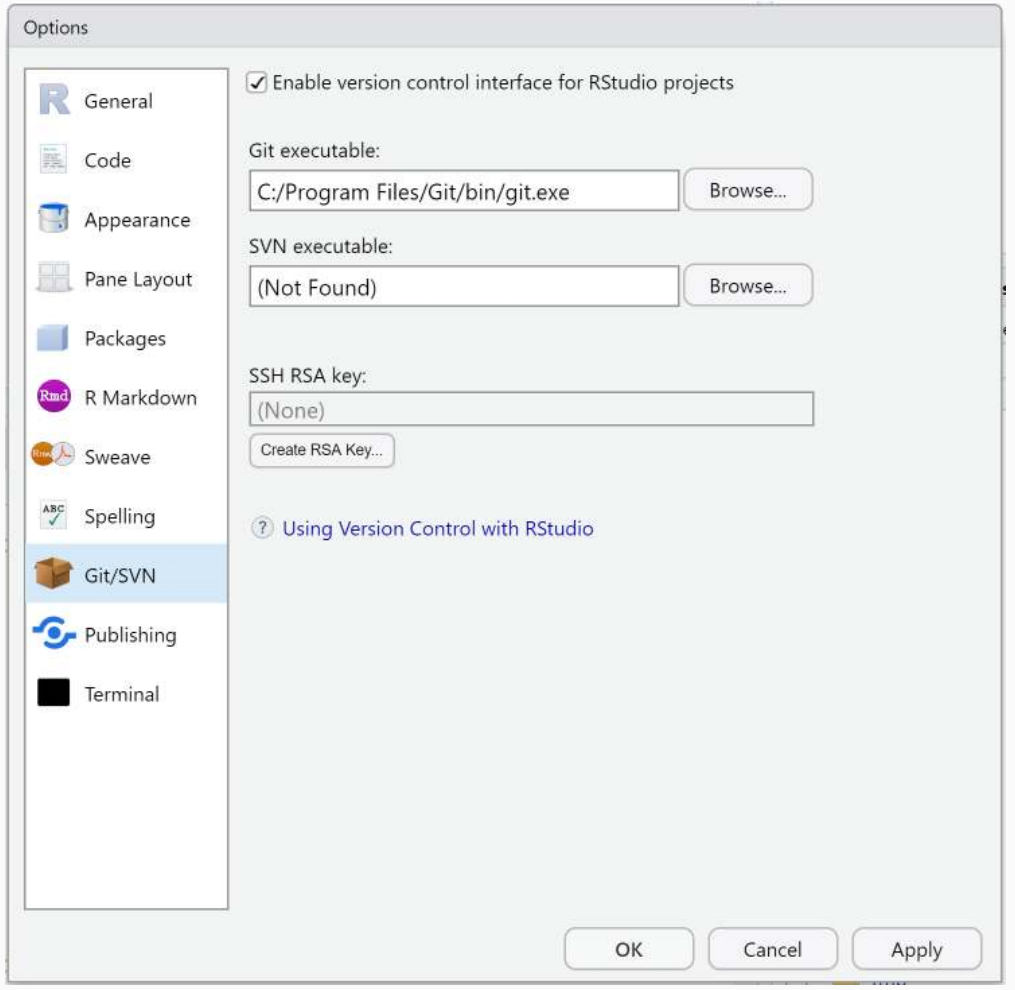
> Tools > Global Options > Git/SVN



6. Configuração: RStudio, git, GitHub

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

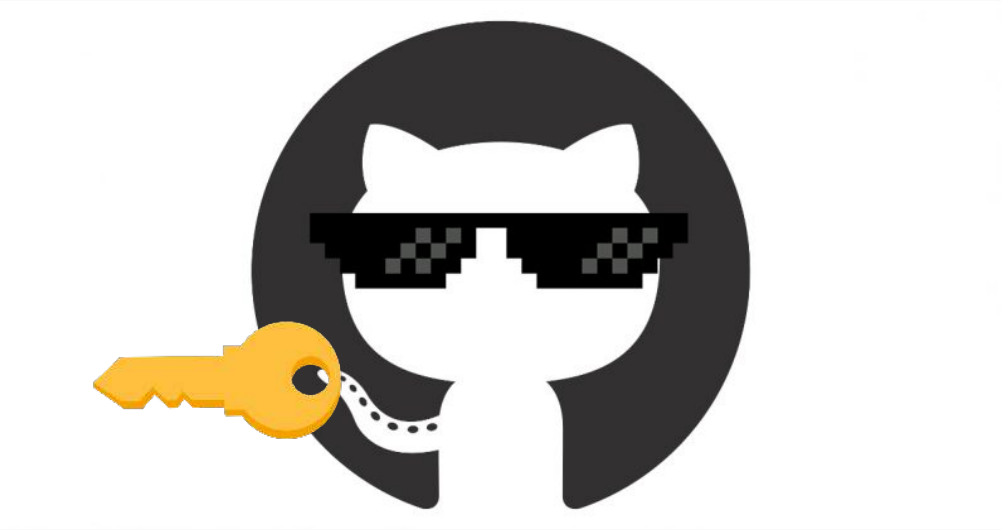
- Windows: `C:/Program Files/Git/bin/git.exe`
- GNU/Linux: `/usr/bin/git`
- MacOS: `???`

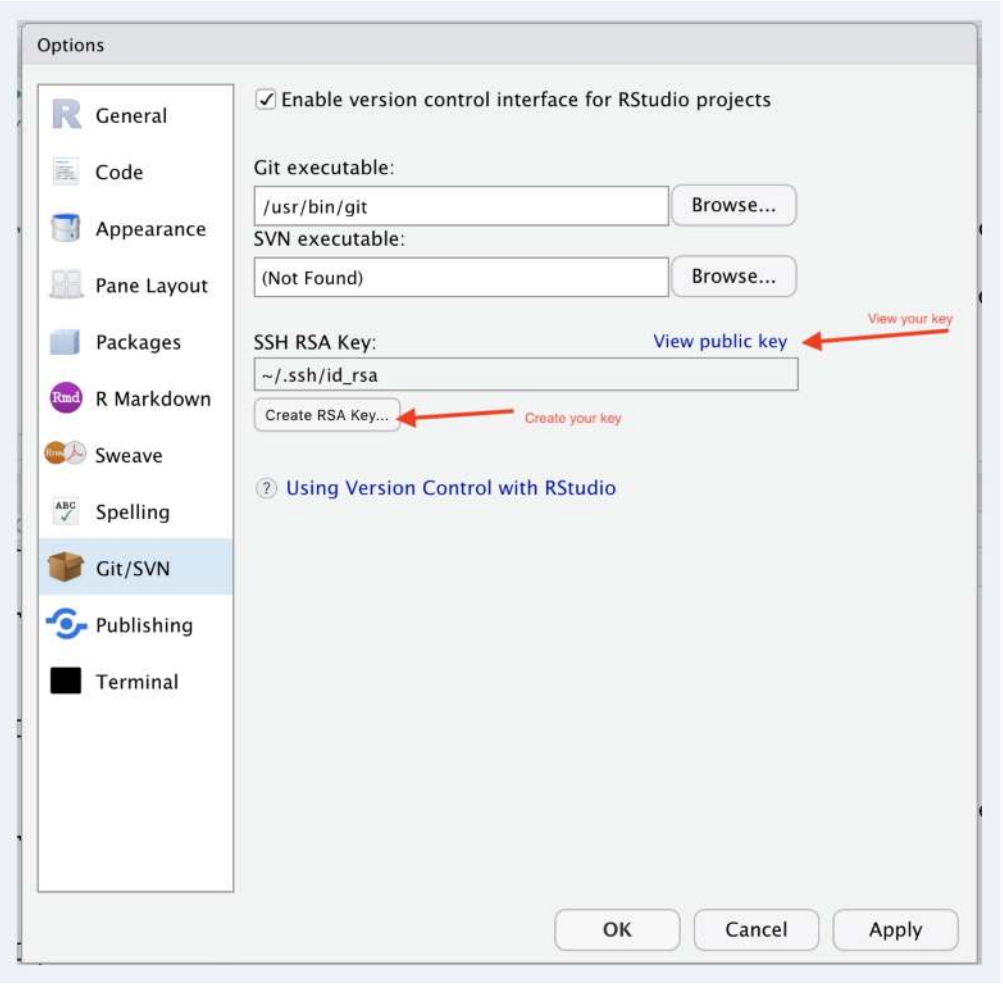


6. Configuração: RStudio, git, GitHub

Criar uma chave SSH

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



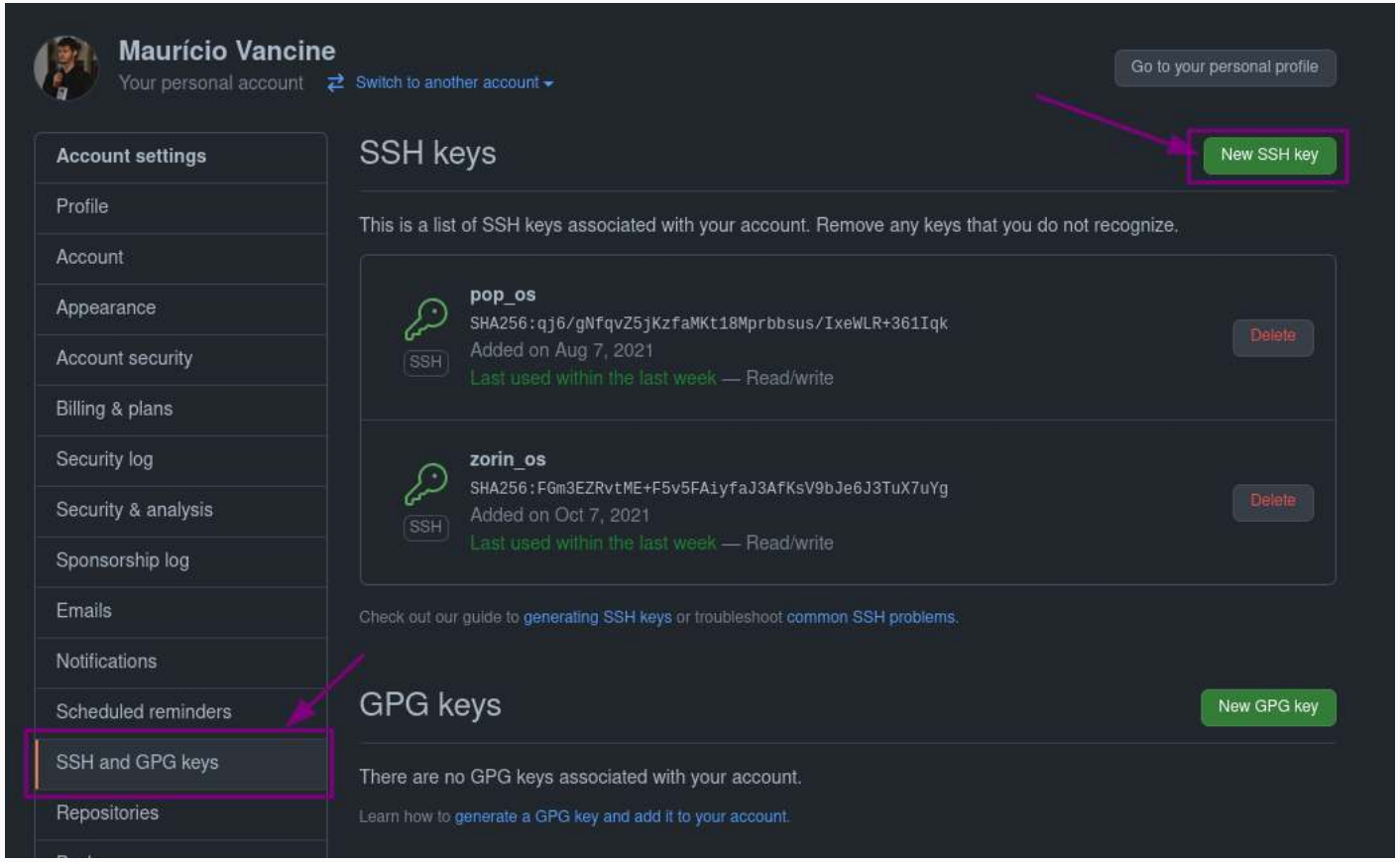


Agora vamos ao GitHub

6. Configuração: RStudio, git, GitHub

No GitHub

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

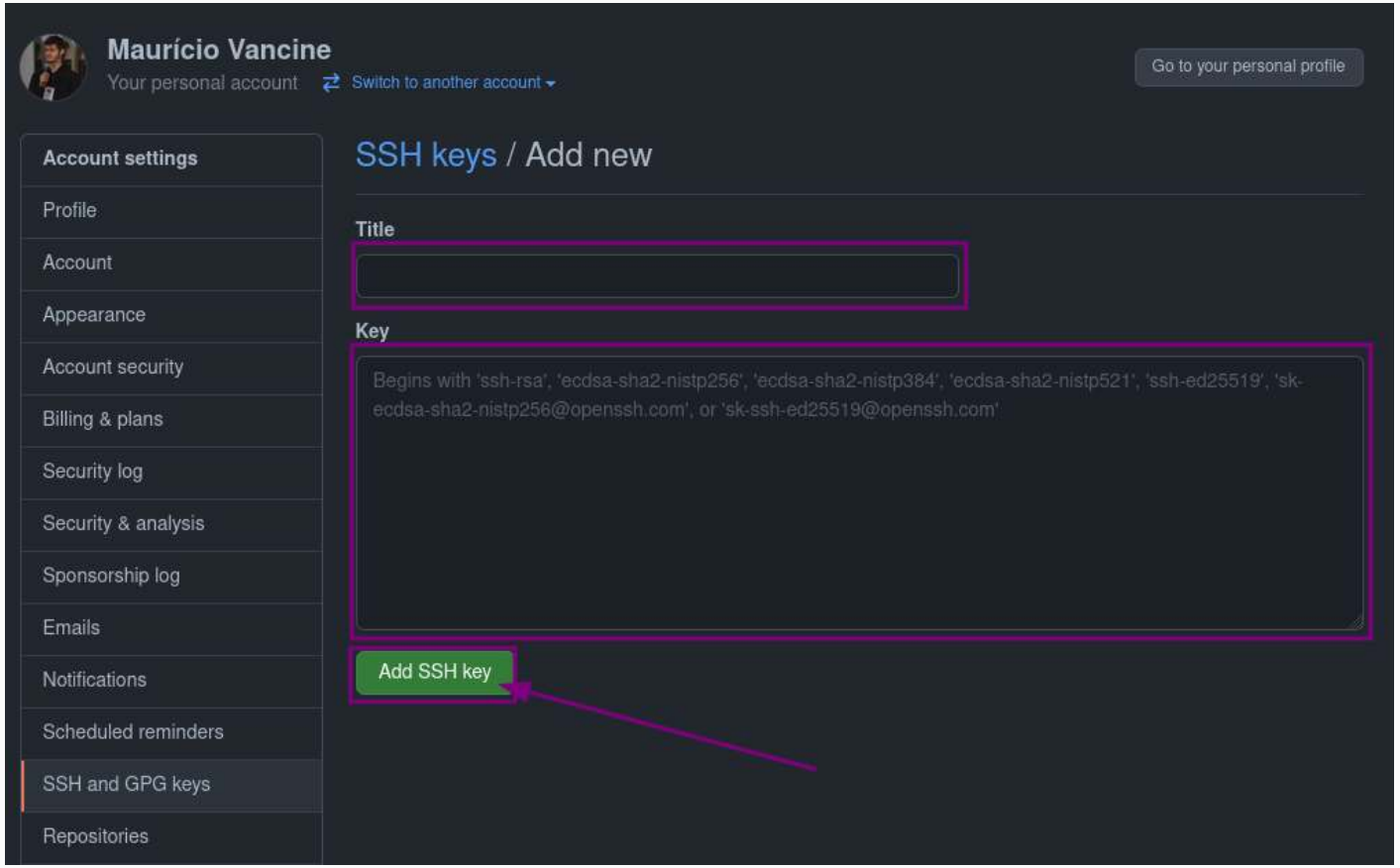


[Add ao GitHub]

6. Configuração: RStudio, git, GitHub

No GitHub

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

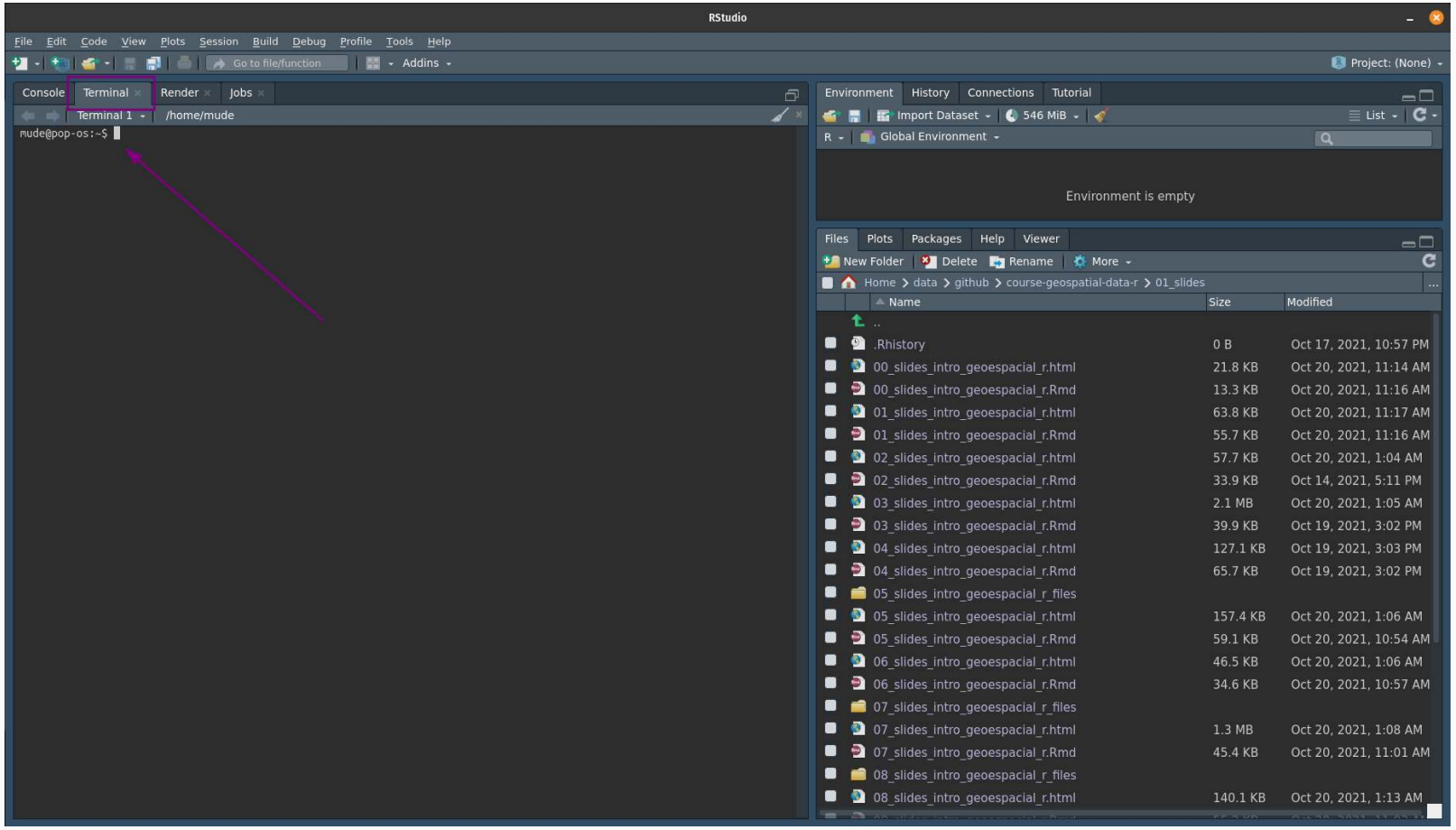


[Add ao GitHub]

Antes de avançarmos, vamos configurar o git

6. Configuração: RStudio, git, GitHub

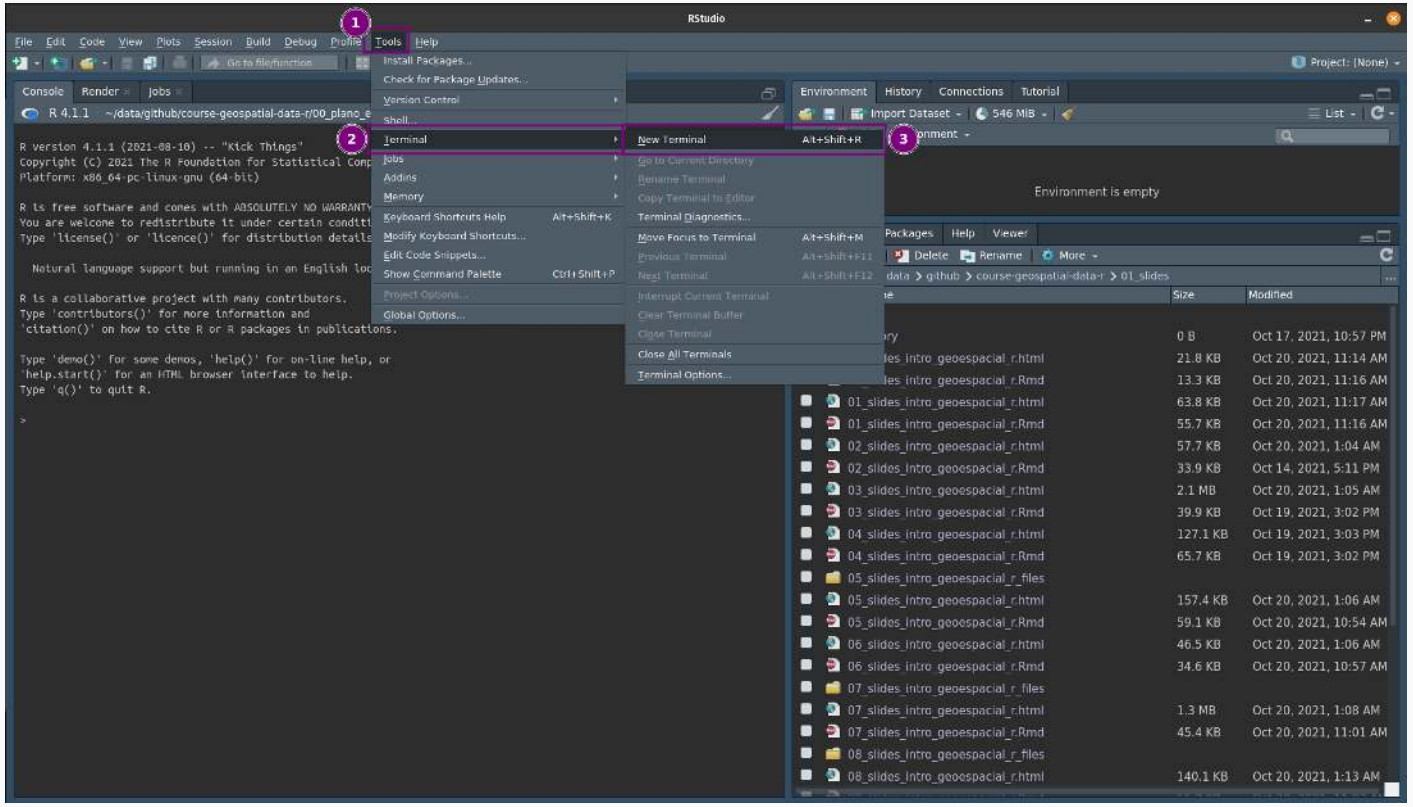
Terminal



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
```

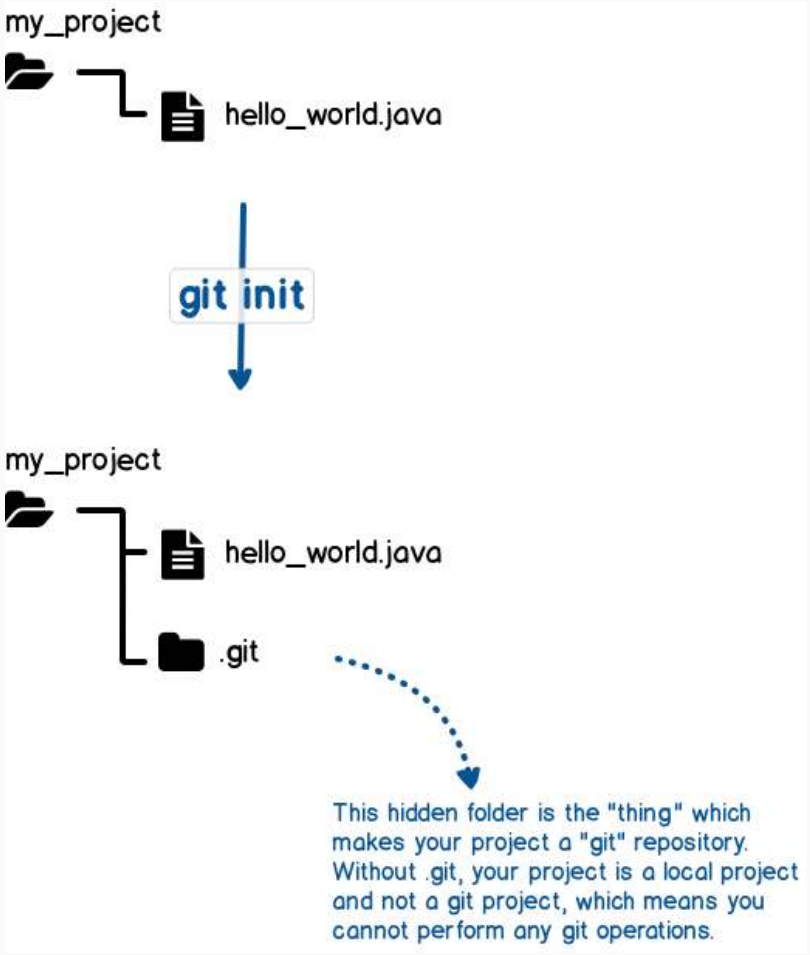
Tudo certo até aqui?

Agora podemos iniciar o versionamento

1. Podemos começar pelo repositório local

7. Iniciando localmente: git init

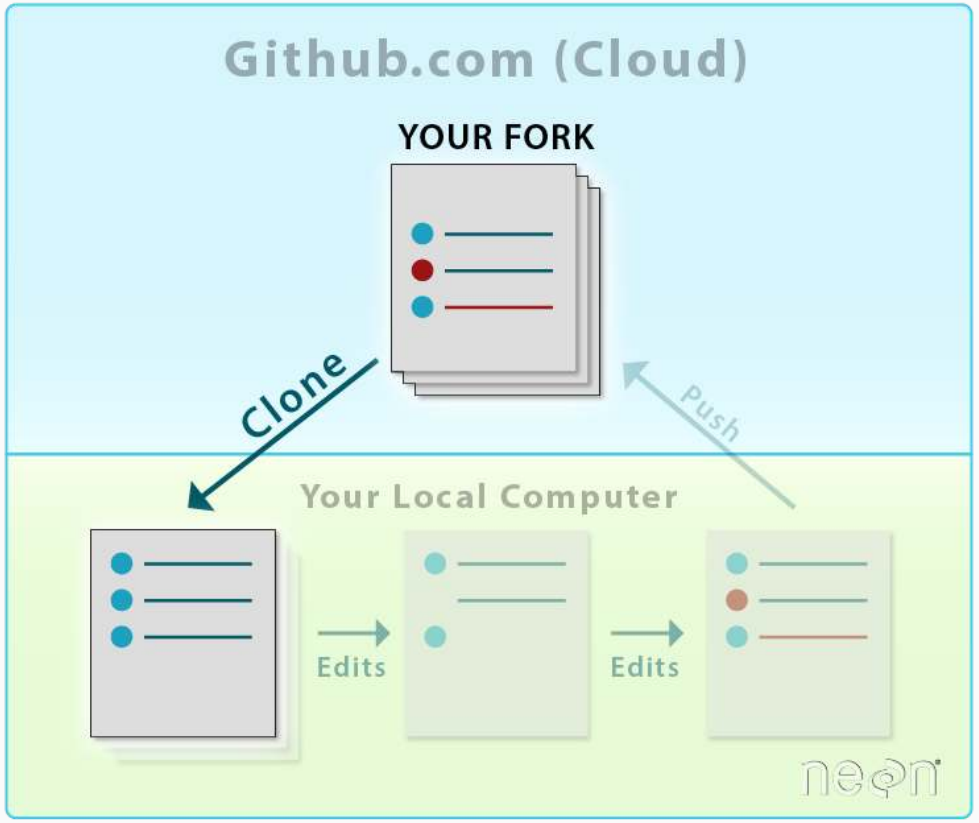
git init: inicia o versionamento de um repositório local



2. Podemos começar pelo repositório remoto

8. Iniciando remotamente: git clone

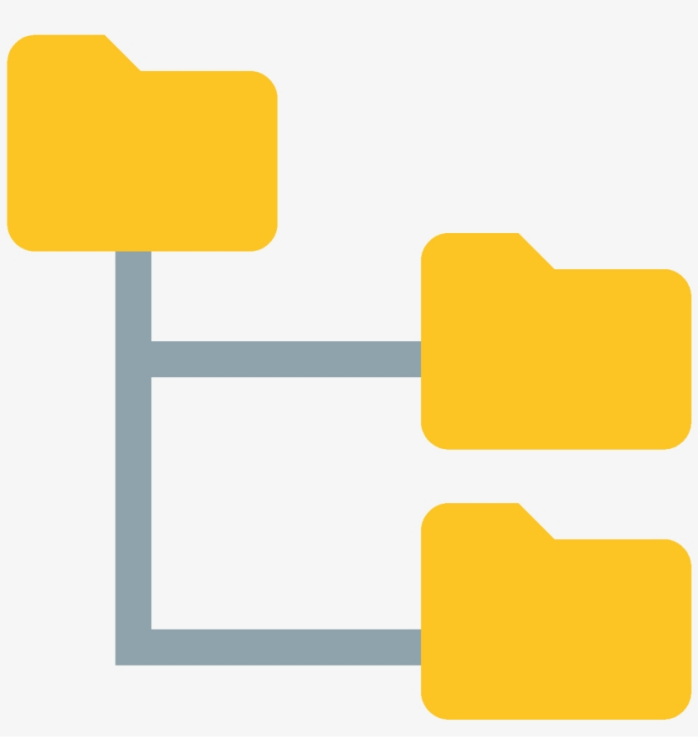
git clone: faz o download de um repositório remoto (e.g. GitHub) para o seu computador (repositório local)



8. Iniciando remotamente: git clone

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

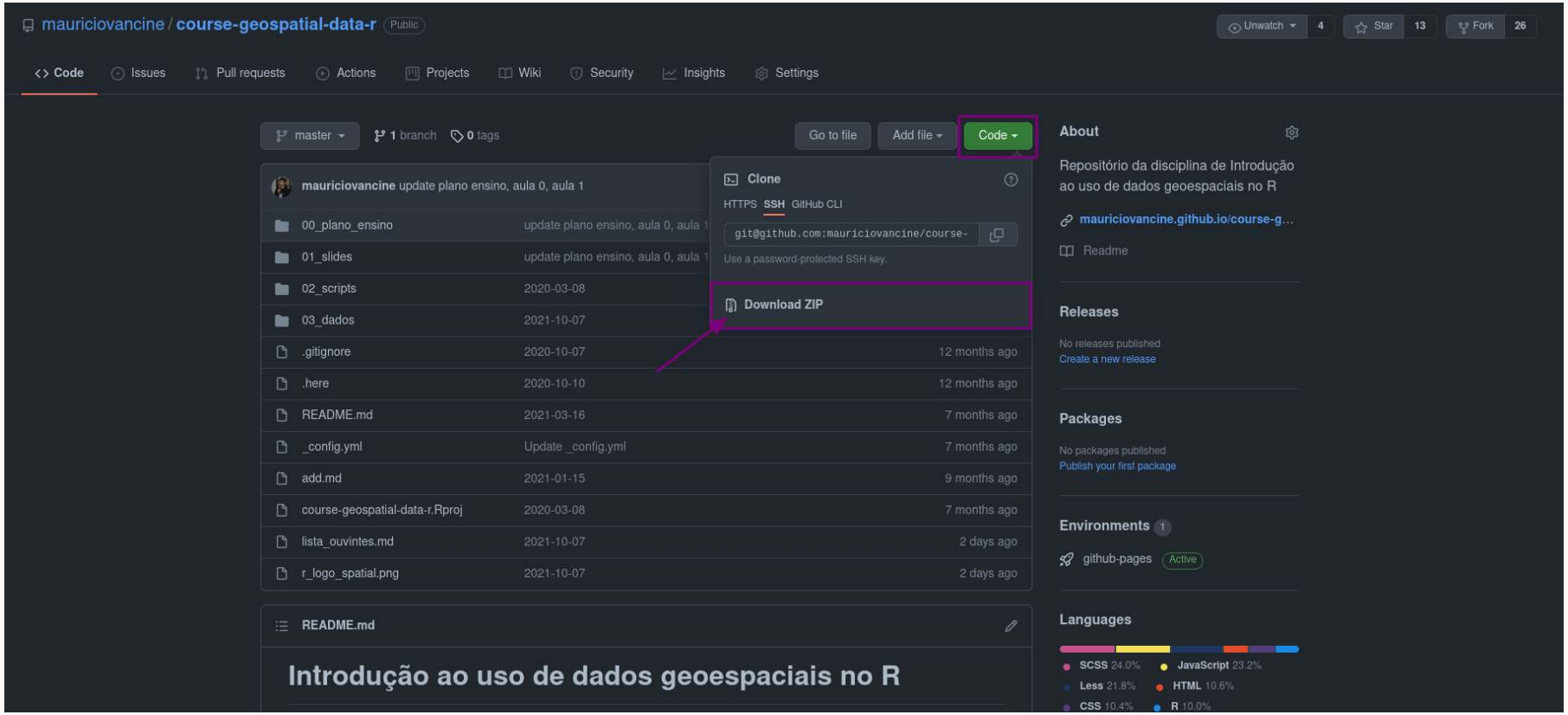
```
".                          # raiz"
"    ├── home/              # home"
"        ├── data/          # dados"
"            ├── github/    # todos os repositorios"
```



E como podemos fazer o clone?

8. Iniciando remotamente: git clone

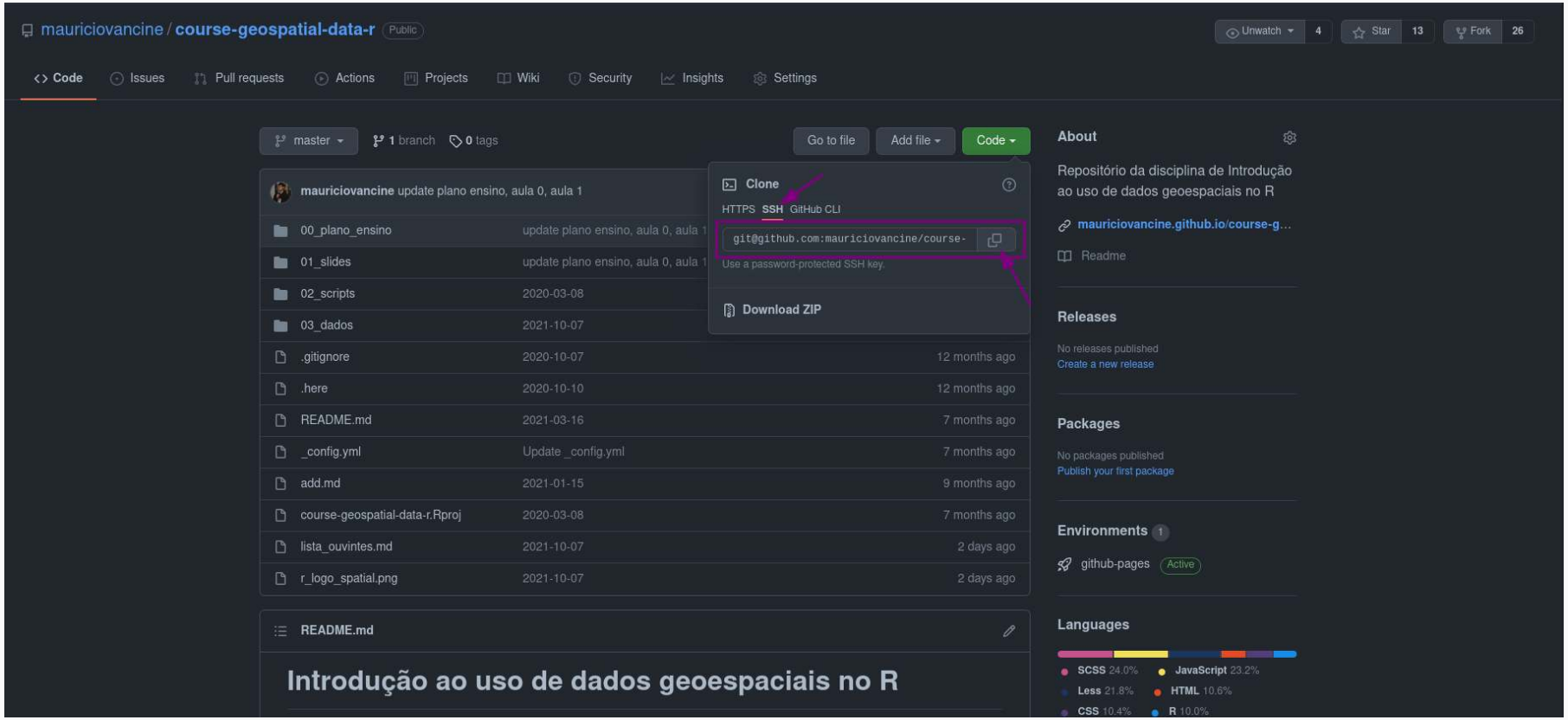
1. **Download diretamente** do repositório remoto no formato **.zip**



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
```



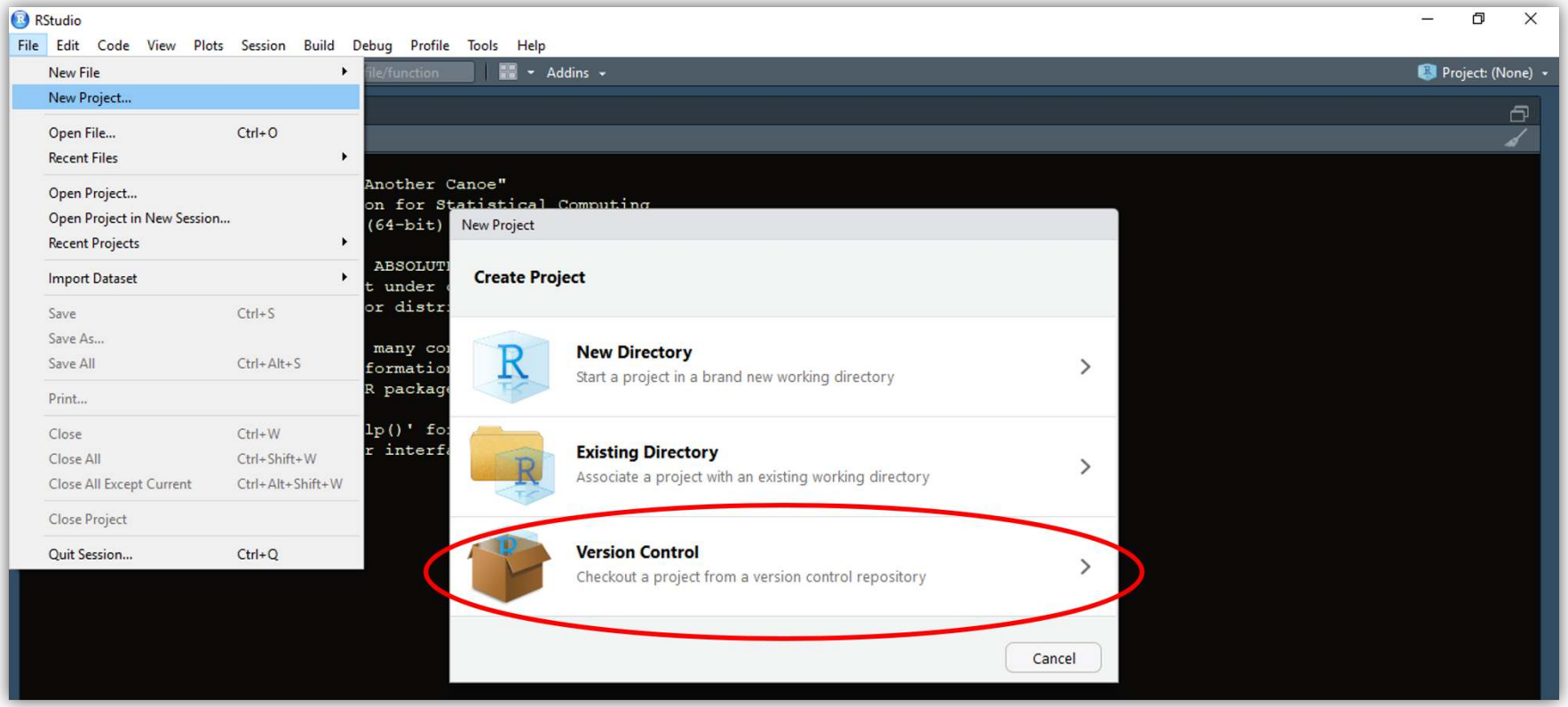
8. Iniciando remotamente: git clone

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

8. Iniciando remotamente: git clone

Criar um **Projeto R** com controle de versão

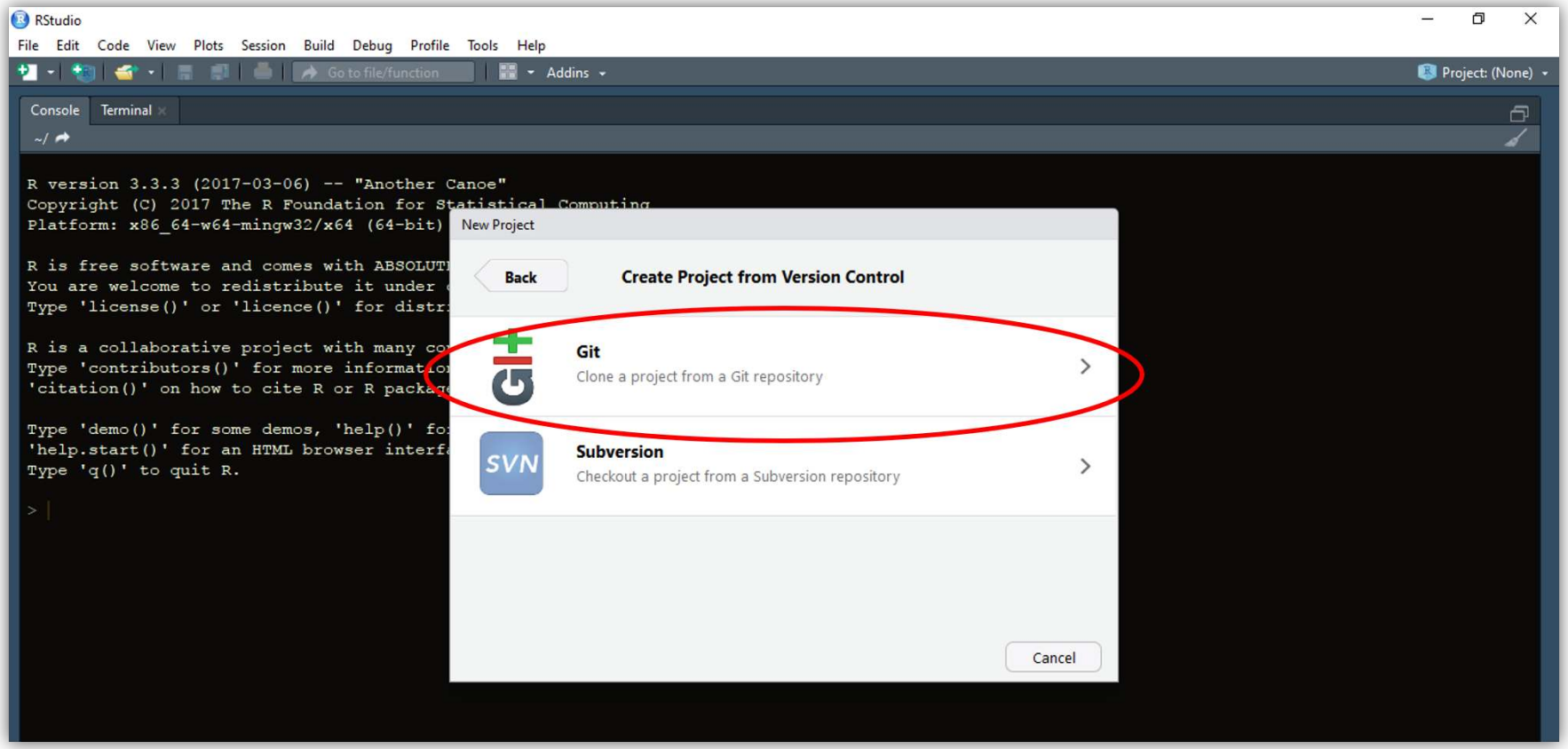
> File > New Project > Version Control



8. Iniciando remotamente: git clone

Escolher **clonar repositório** do GitHub

> Git



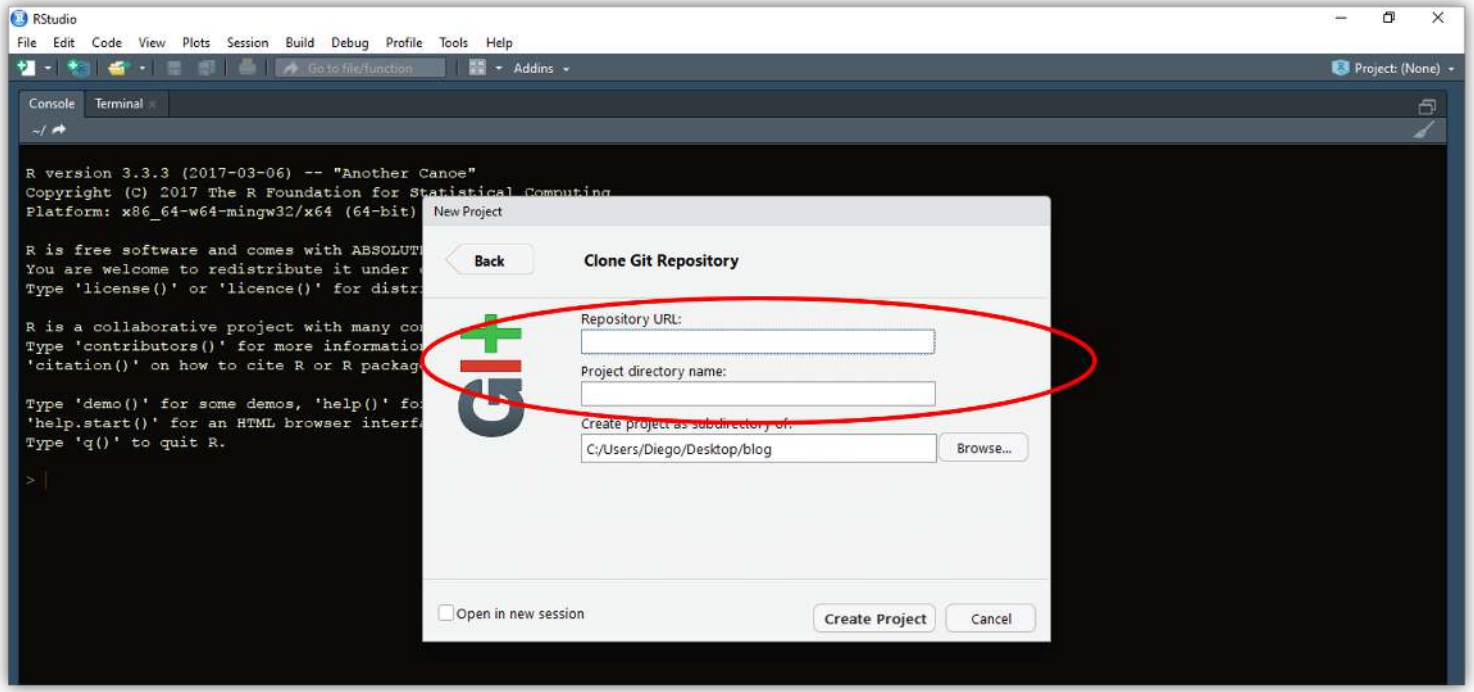
8. Iniciando remotamente: git clone

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`

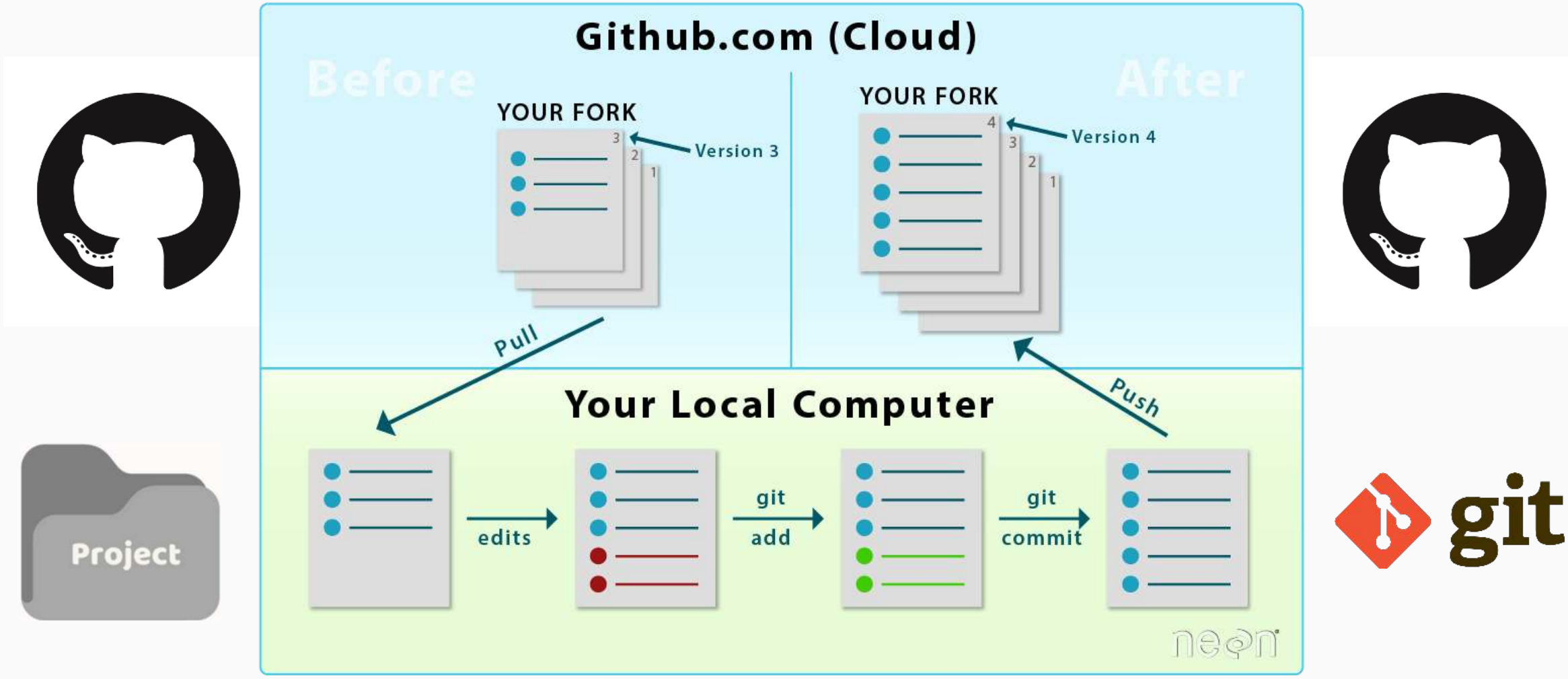


Se uma janela aparecer, basta digitar "yes" para criar uma autentificação

Vamos aguardar o download...

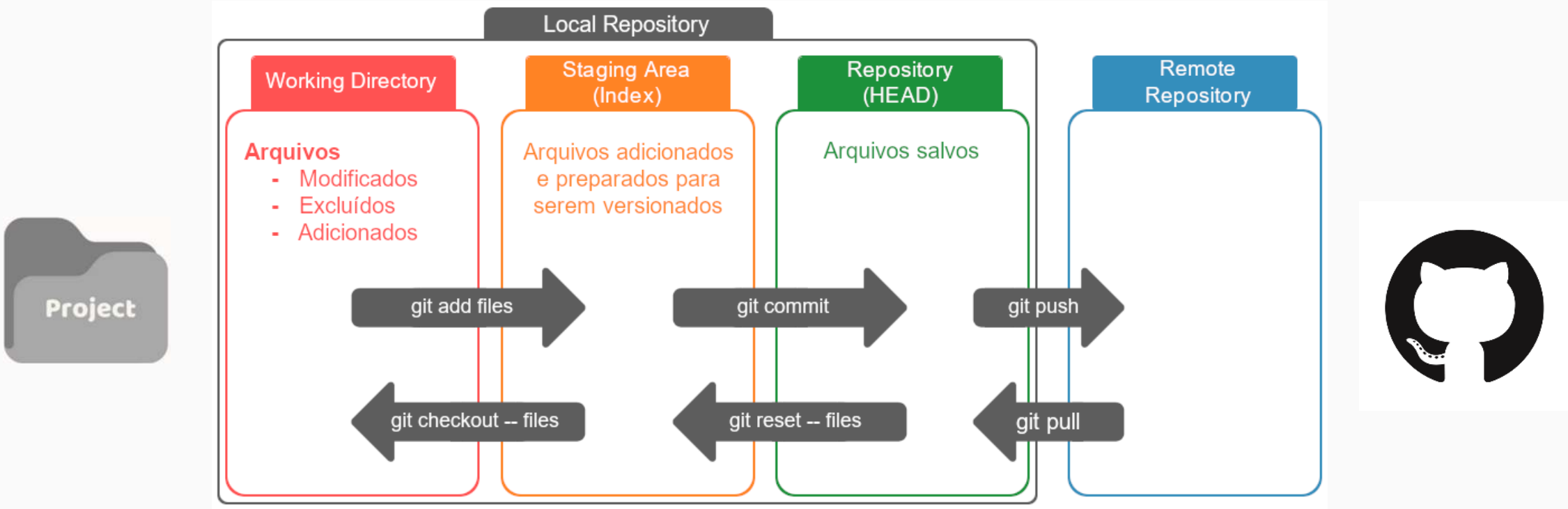
Agora sim, vamos começar o versionamento propriamente dito

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

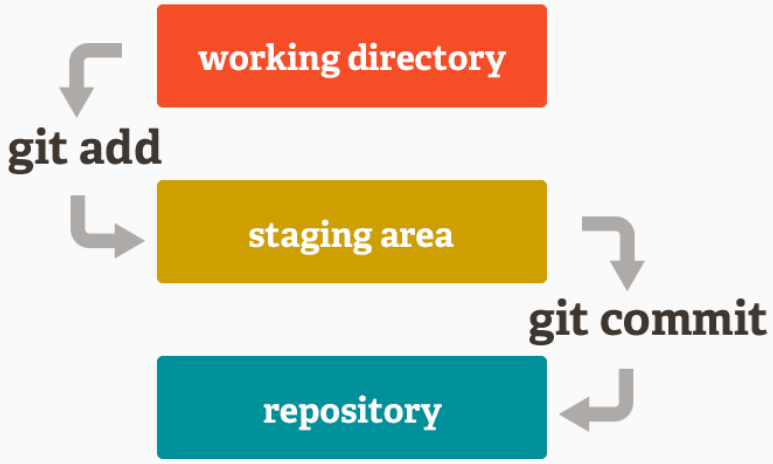
9. Versionamento: git add, git commit e git status

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*)



[Git — Os três estados]

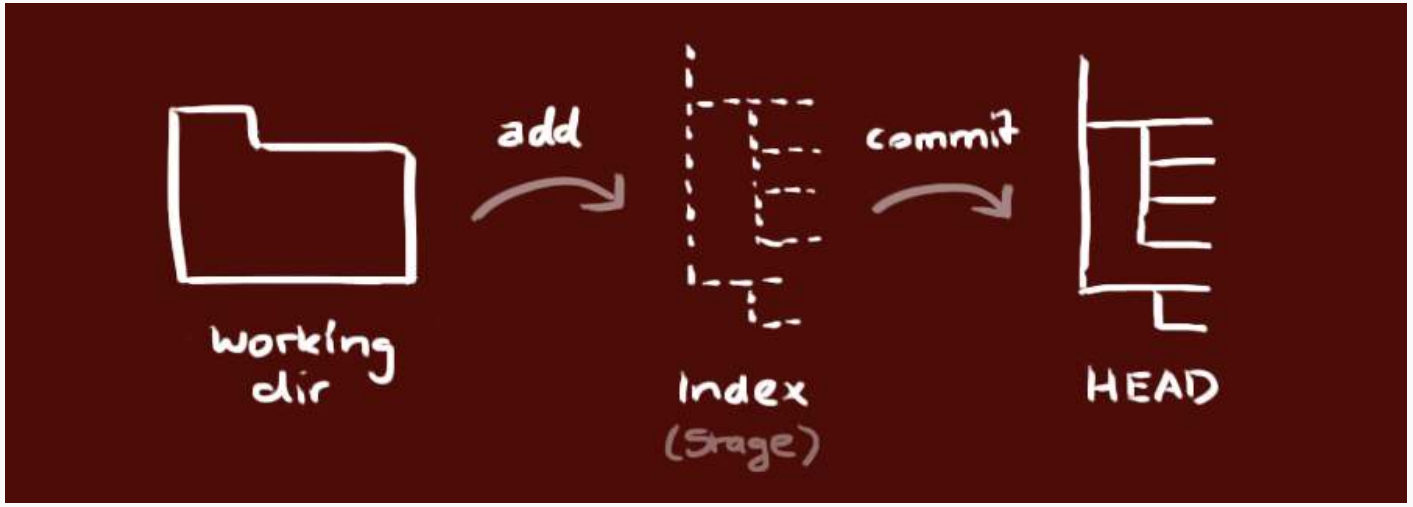
9. Versionamento: git add, git commit e git status

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*)

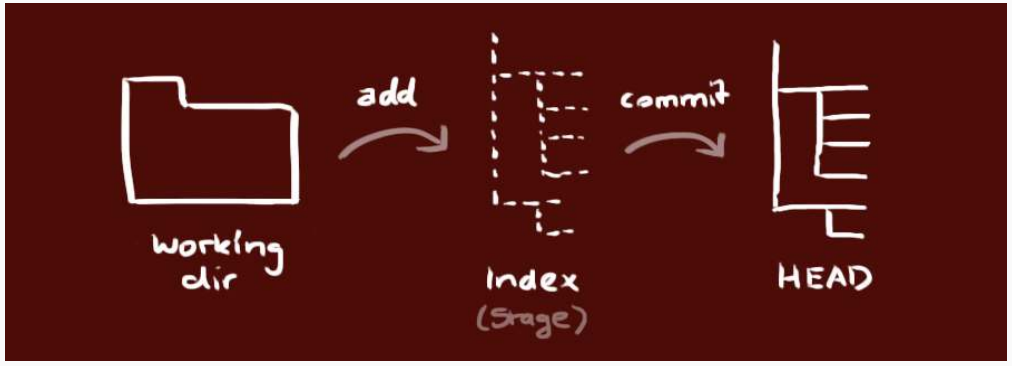
2. **Preparado (*staged*)**: arquivos adicionados e preparados para serem consolidados na versão final (*staged area*)

3. **Consolidado (*committed*)**: criação da versão consolidada de todos os arquivos que estavam no *staged* (não considera os que estão no *modified*) (*.git*)



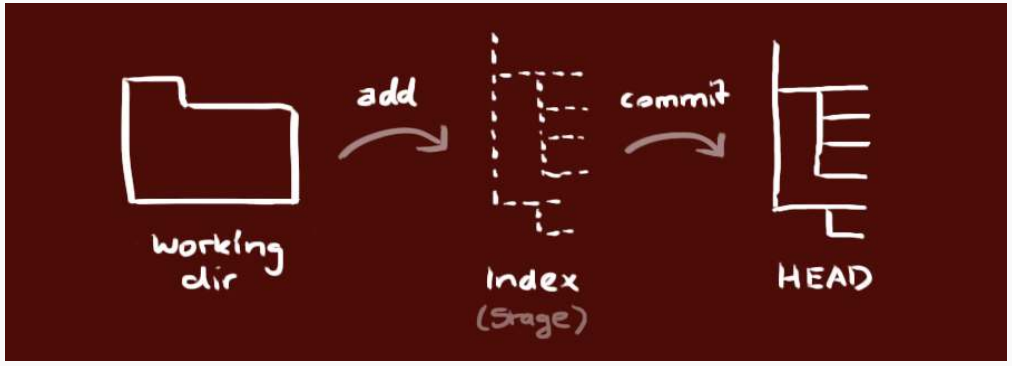
9. Versionamento: git add, git commit e git status

Estados dos arquivos para o git



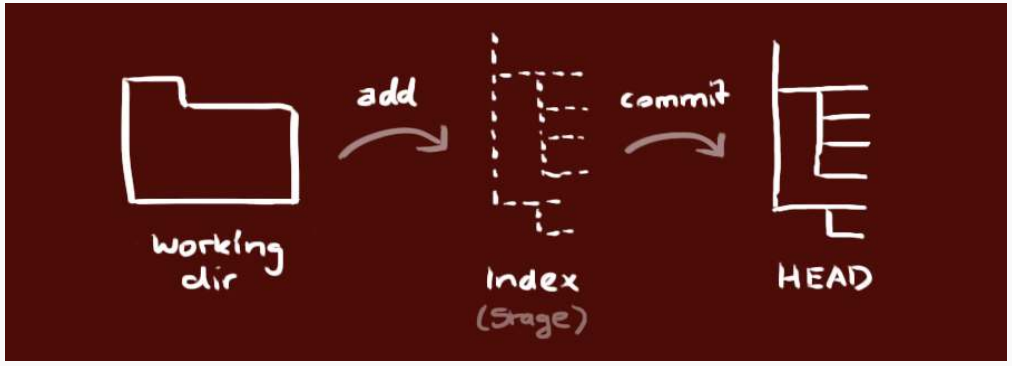
9. Versionamento: git add, git commit e git status

Estados dos arquivos para o git



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

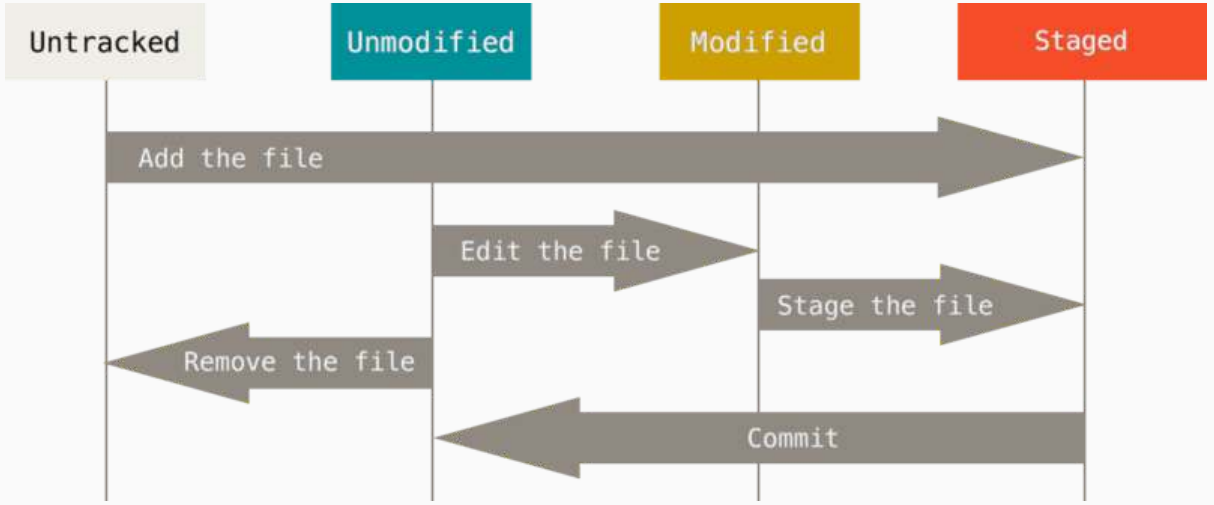
Estados dos arquivos para o git



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

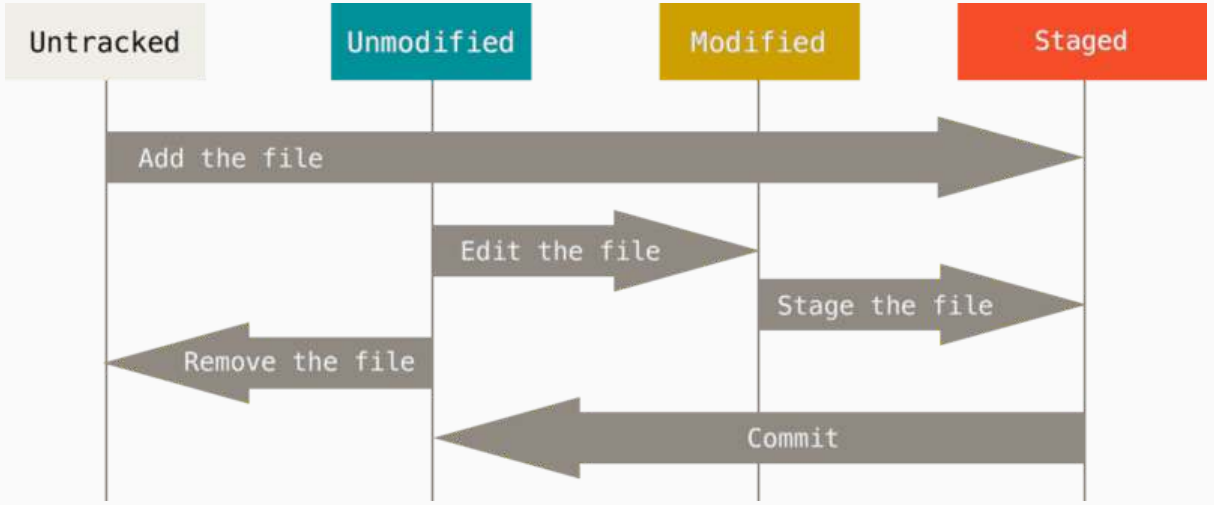


9. Versionamento: git add, git commit e git status

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

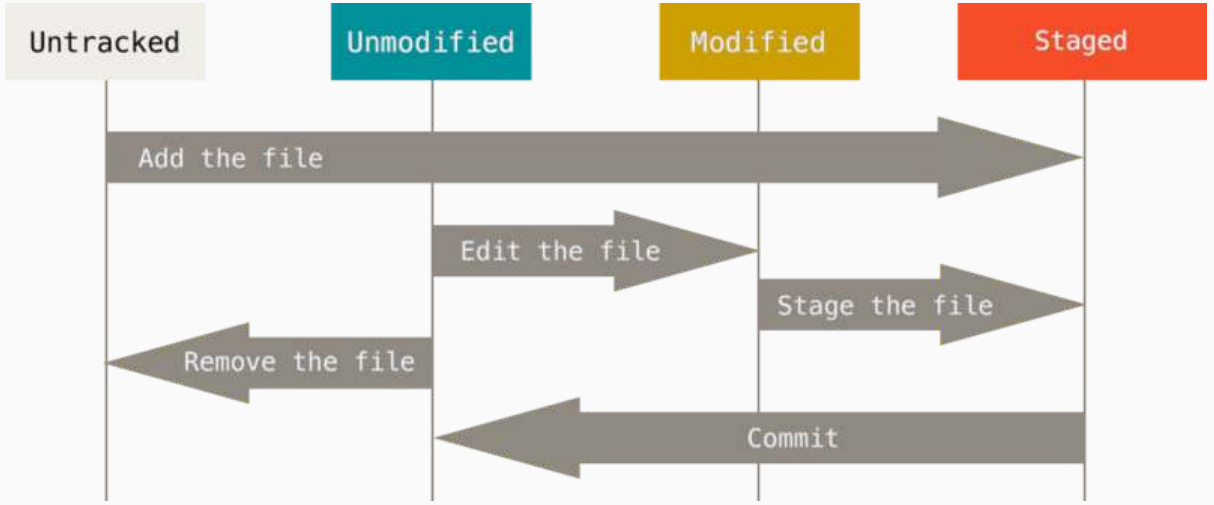
```
# criar um arquivo
> teste.txt
```

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)
```

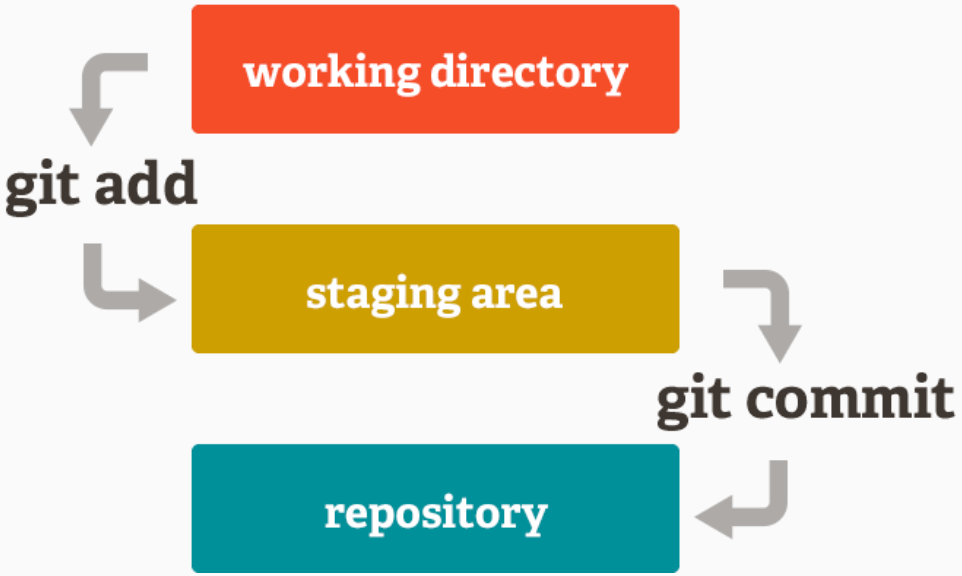


9. Versionamento: git add, git commit e git status

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

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

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

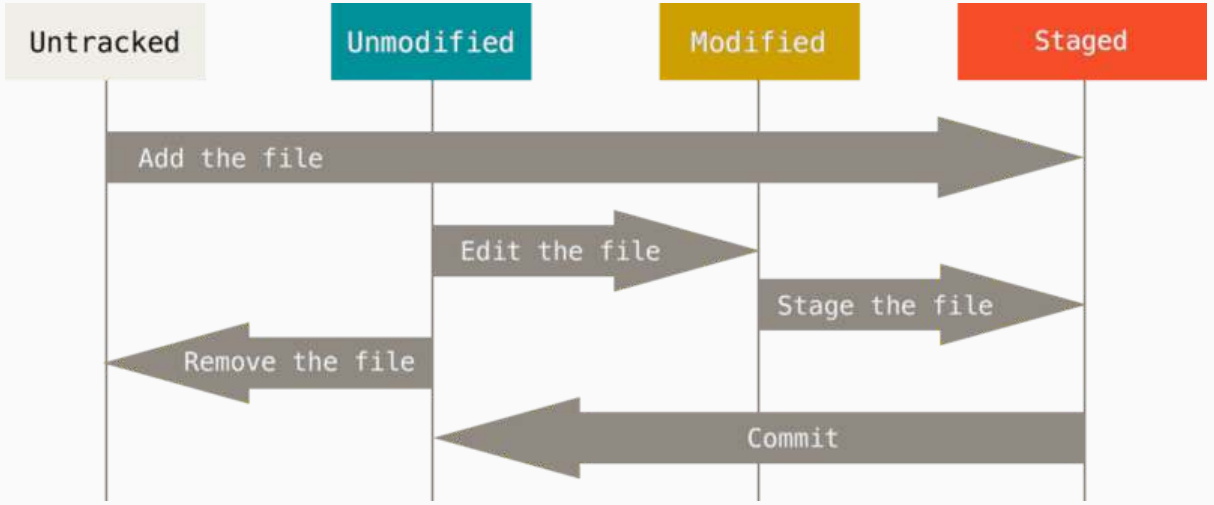


9. Versionamento: git add, git commit e git status

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
```

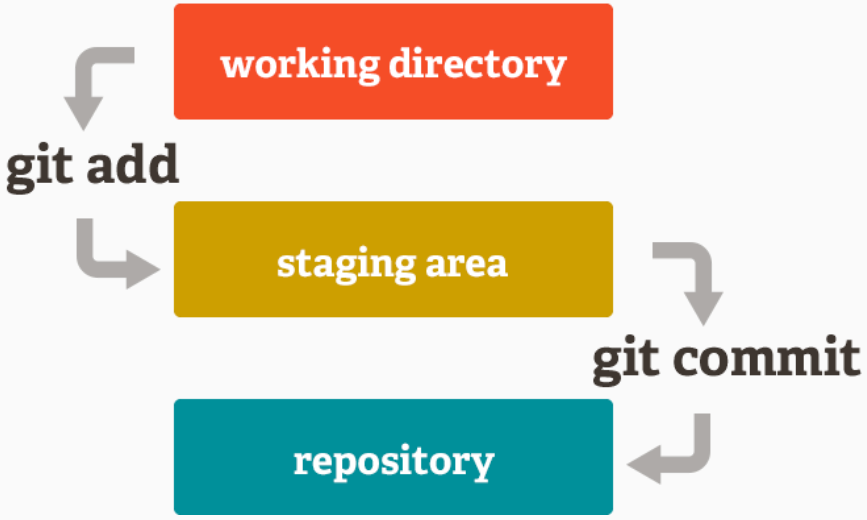


9. Versionamento: git add, git commit e git status

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
```

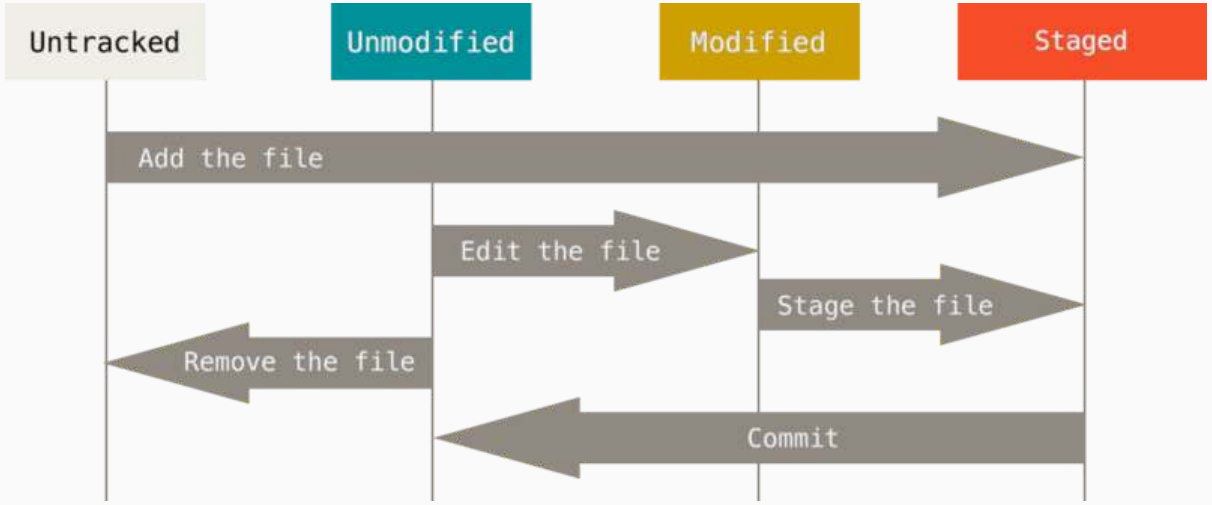


9. Versionamento: git add, git commit e git status

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**

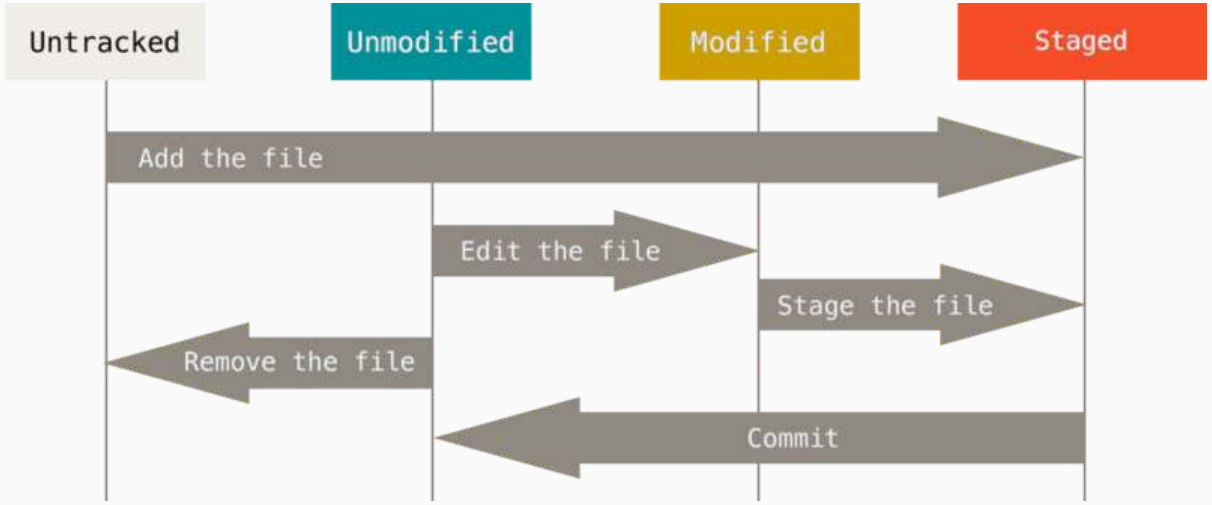
```
Acabo de inserir uma edição ao meu arquivo
```

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")
```

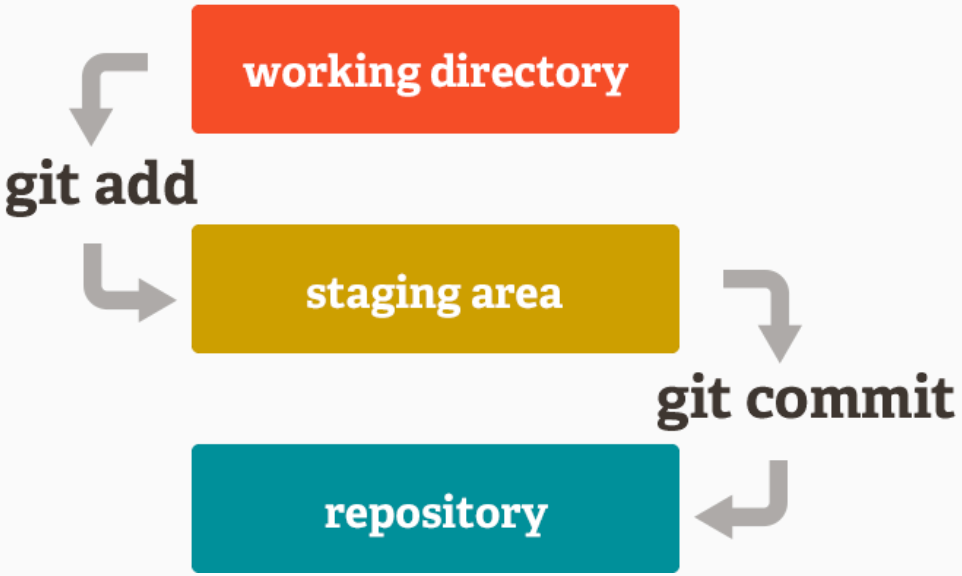


9. Versionamento: git add, git commit e git status

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

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

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

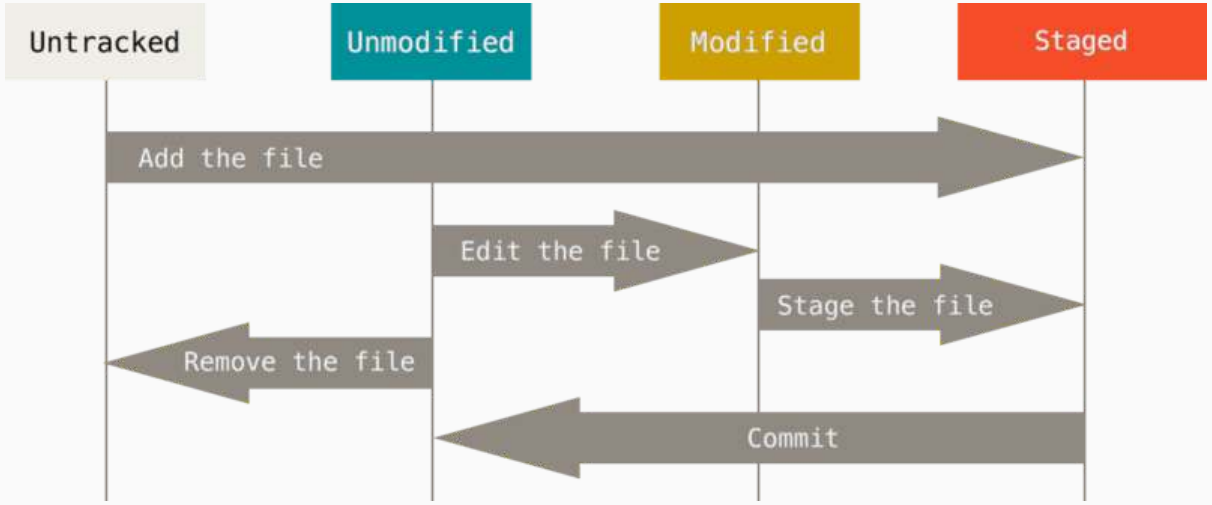


9. Versionamento: git add, git commit e git status

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
```

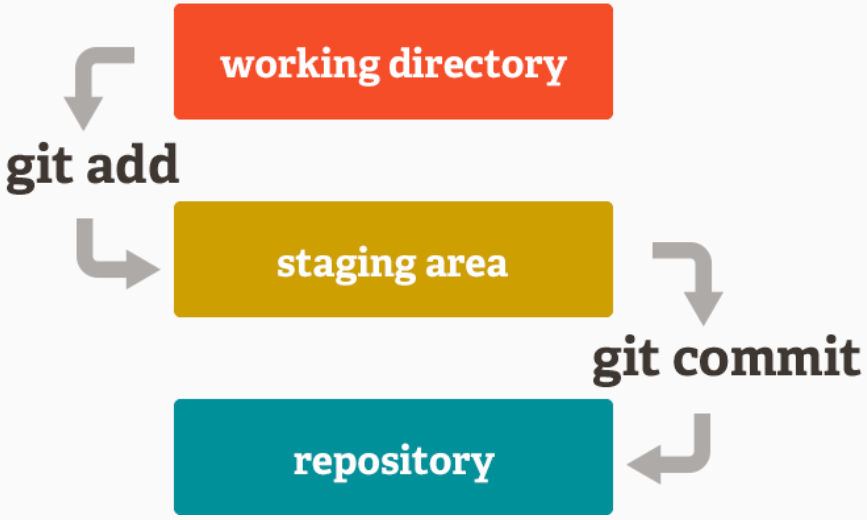


9. Versionamento: git add, git commit e git status

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

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

```
[master dcbd894] 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!

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

Usando Git Direito | Limpando seus Commits! - Fabio Akita

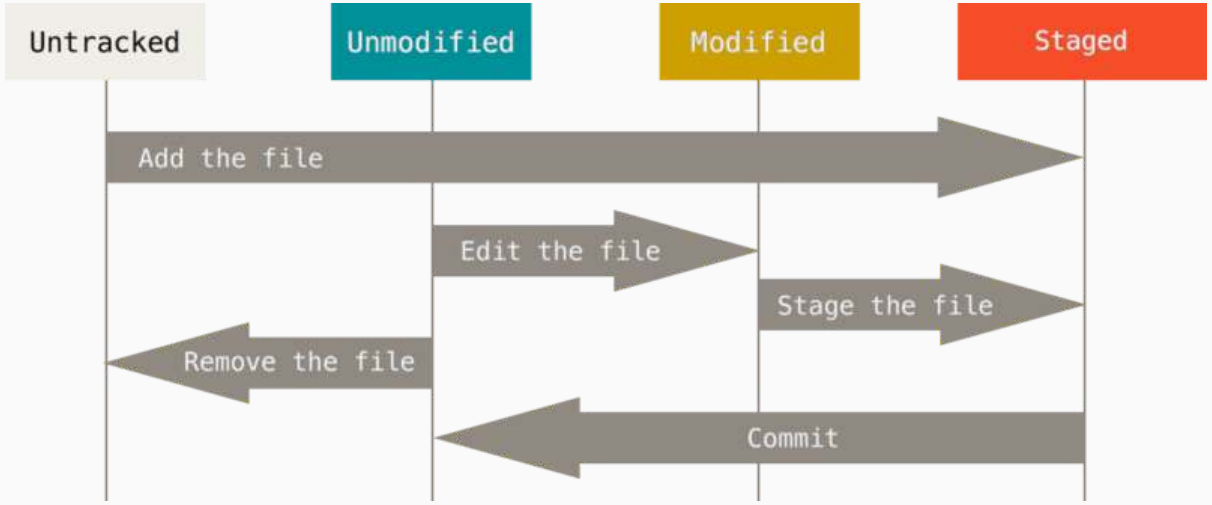
9. Versionamento: git add, git commit e git status

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

- raster.tif

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)

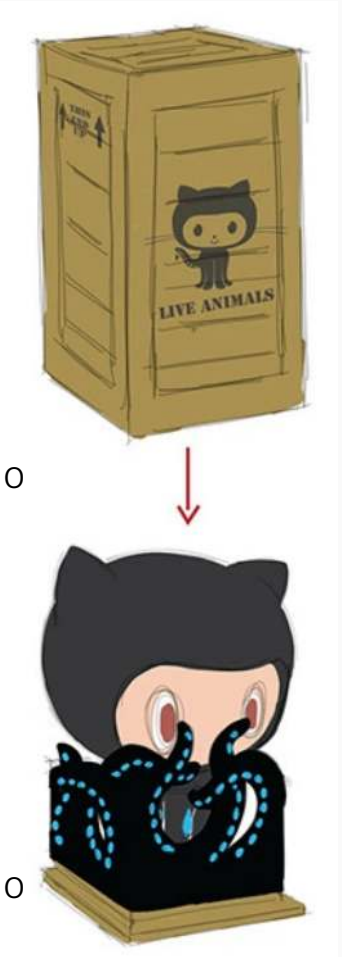
- Arquivos adicionados via browser é limitado a 25 Mb

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

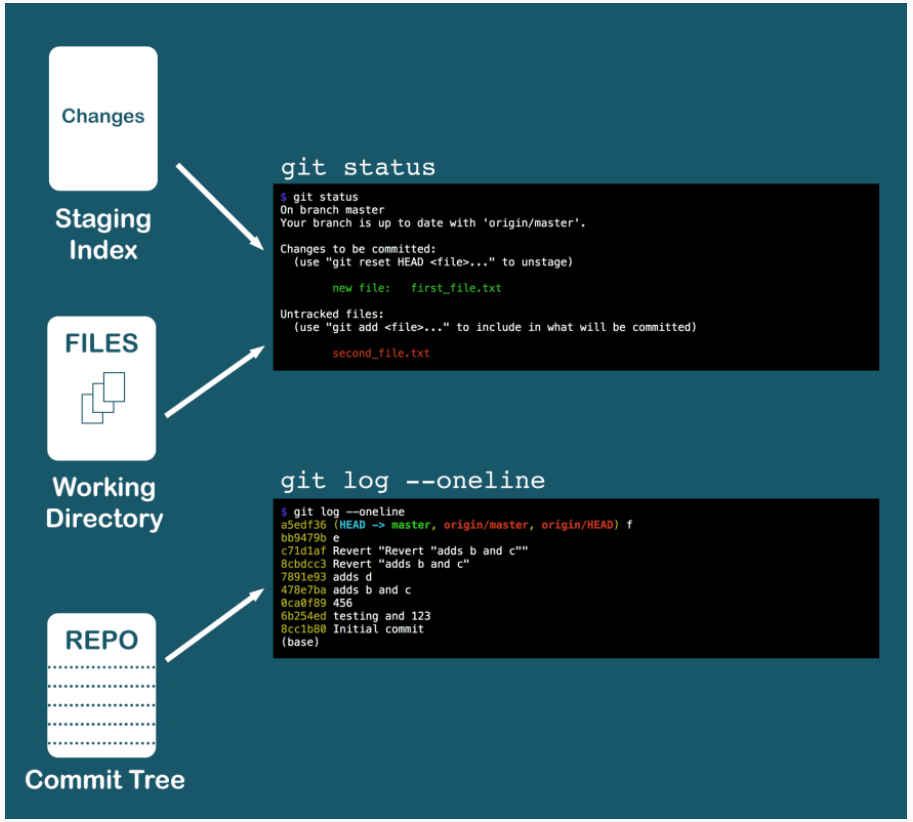
Dúvidas?

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 - dcbd894)
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

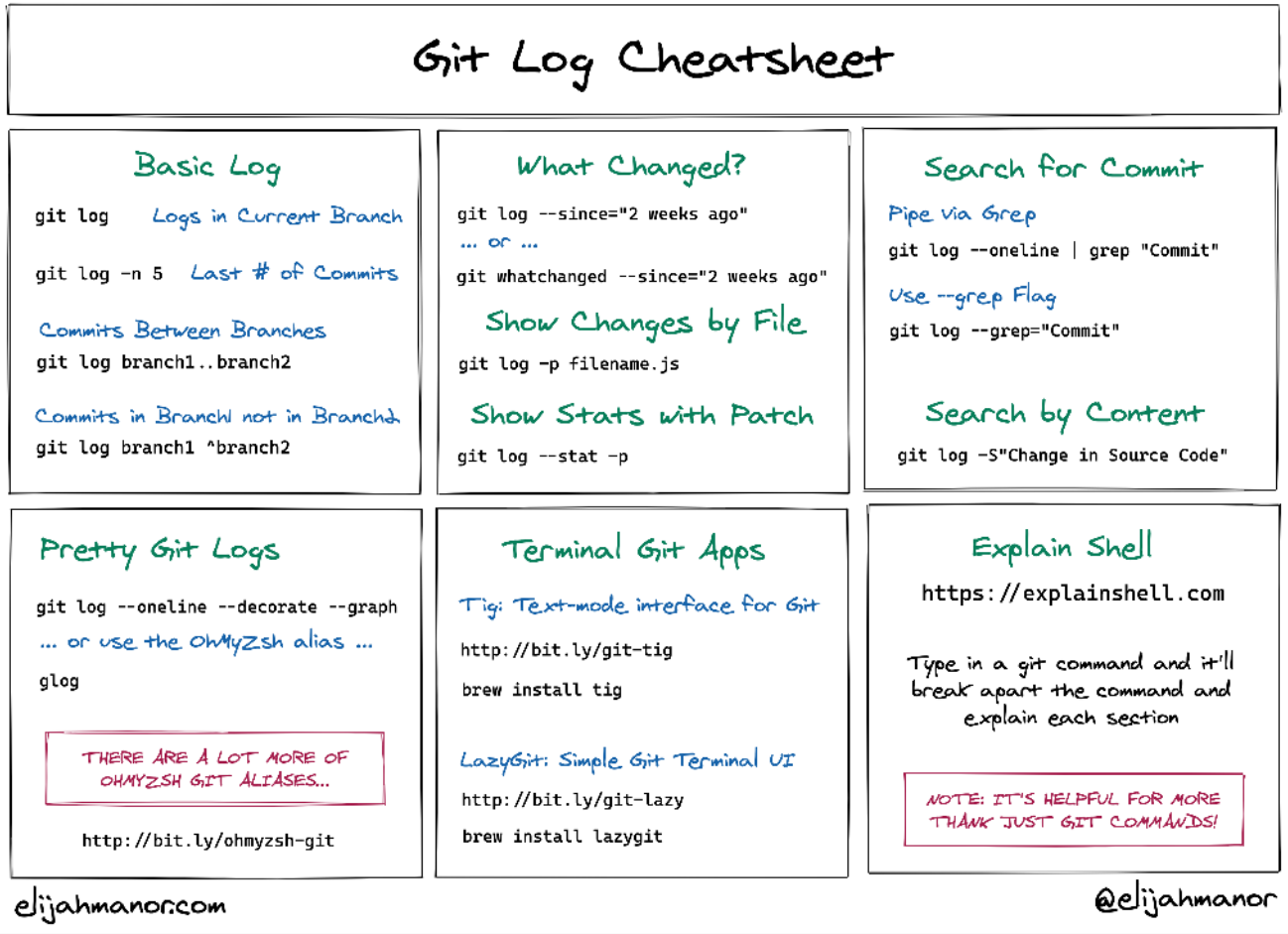
git log --oneline: registro de todo o histórico de commits simplificado

```
git log --oneline
```

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

11. Histórico: git log e git show

git log: cheatsheet



11. Histórico: git log e git show

git show: visualização das alterações realizadas nos arquivos

```
git show
```

```
commit dcbd894ade9a85f78e1b0d6ed08cbc76dcc2448f (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
```

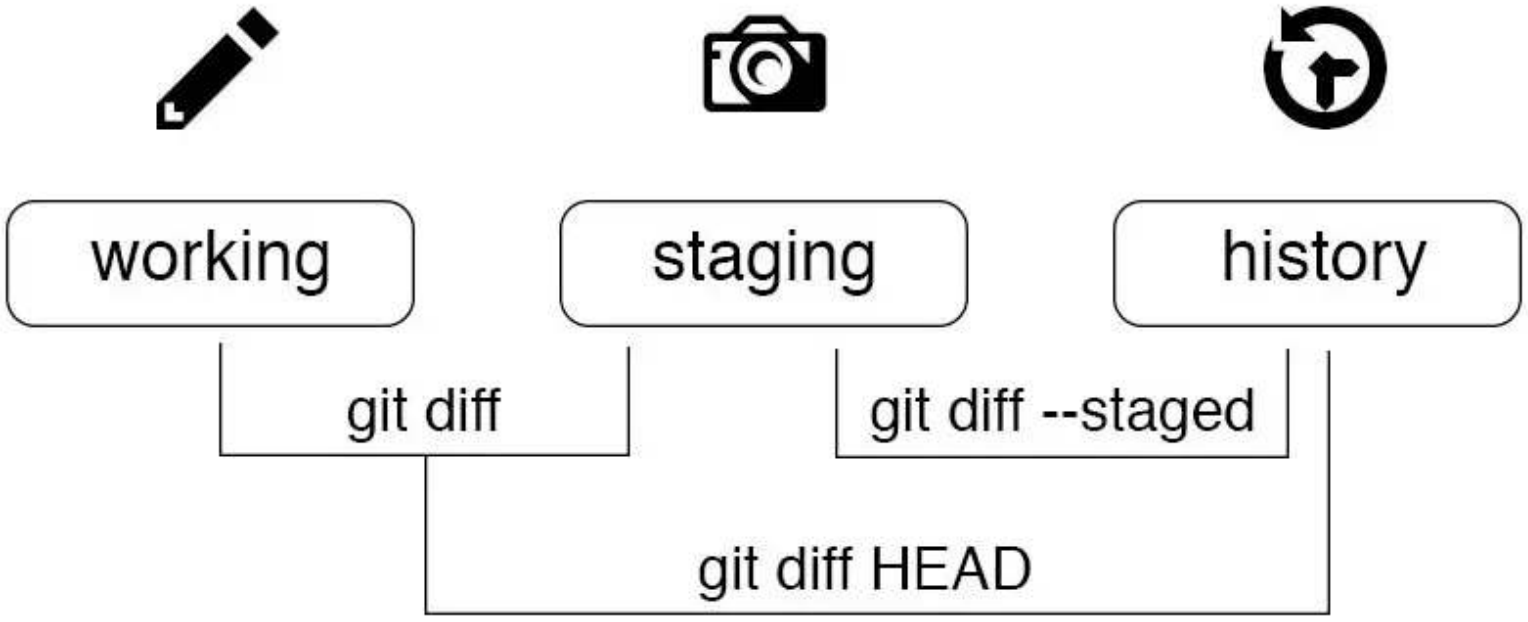
```
commit 8b33bc57187a2bcb59172c75ce7266fbddb64eff
Author: mauriciovancine <mauricio.vancine@gmail.com>
Date:  Mon Mar 15 11:02:04 2021 -0300

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*

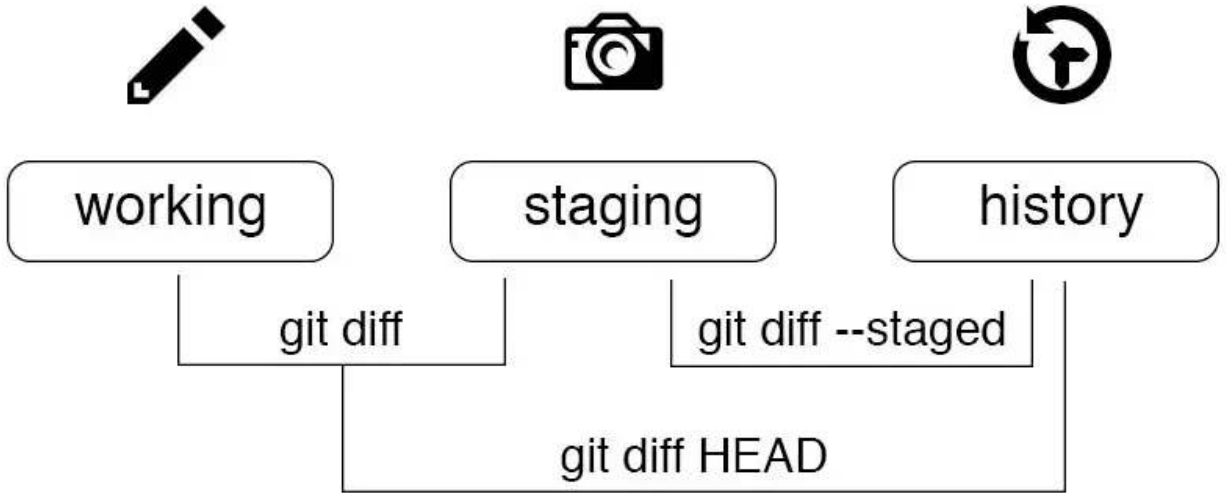


12. Diferenças: git diff

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?

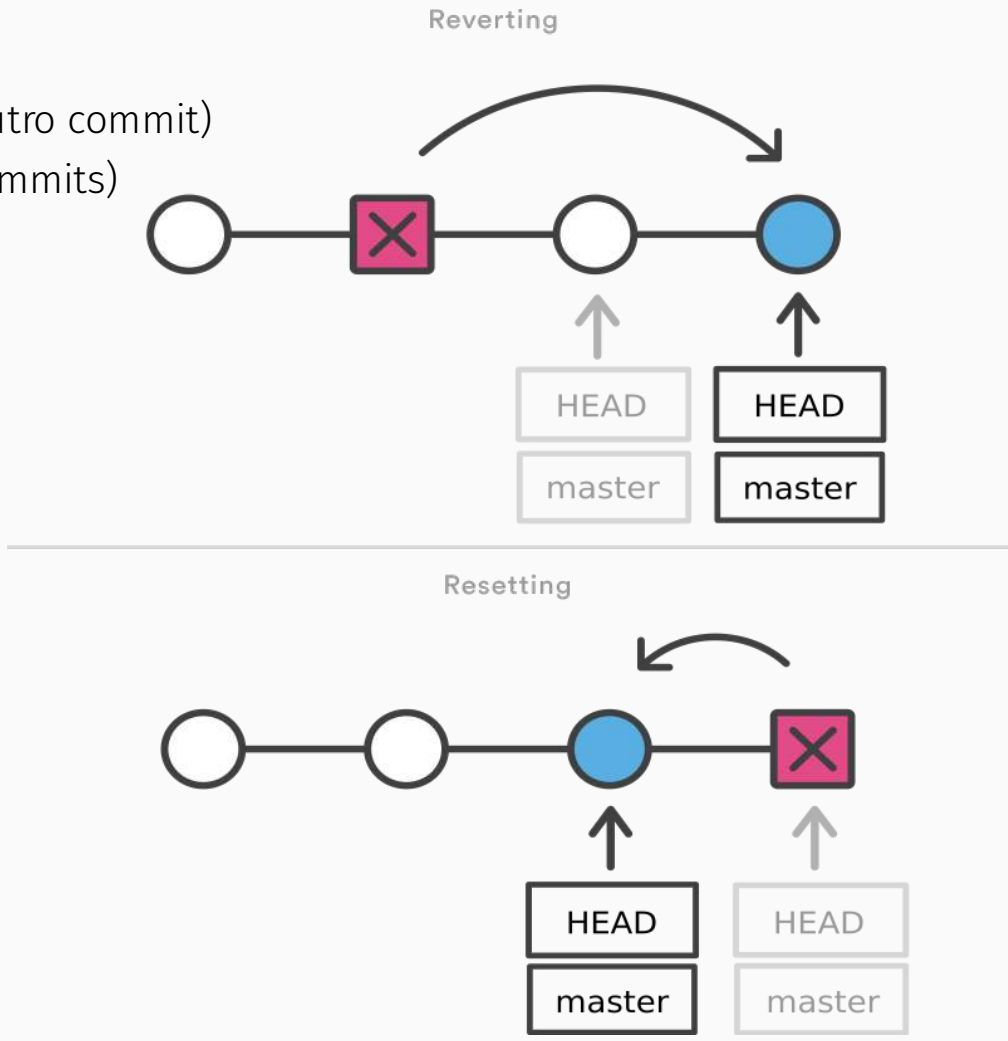
E como voltar no tempo?


DMC
DMC

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 especifico
git revert <hash do commit>
```

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

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

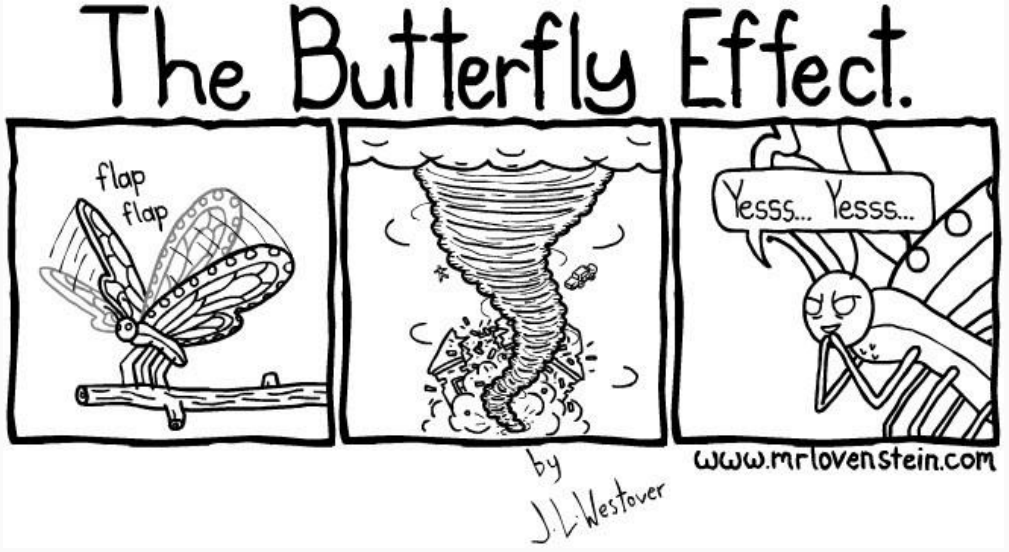
13. Desfazer: git revert e git reset

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 f9f2e9c867cfd549d0efcab1519a5ec9d86e7d0d.

# 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

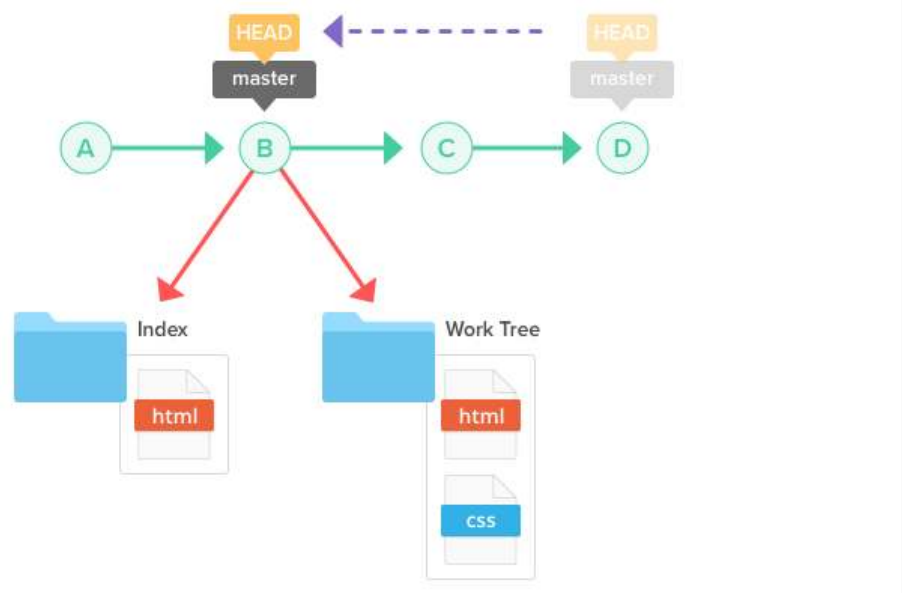
```
git reset [modo] [arquivo|commit]
```

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**

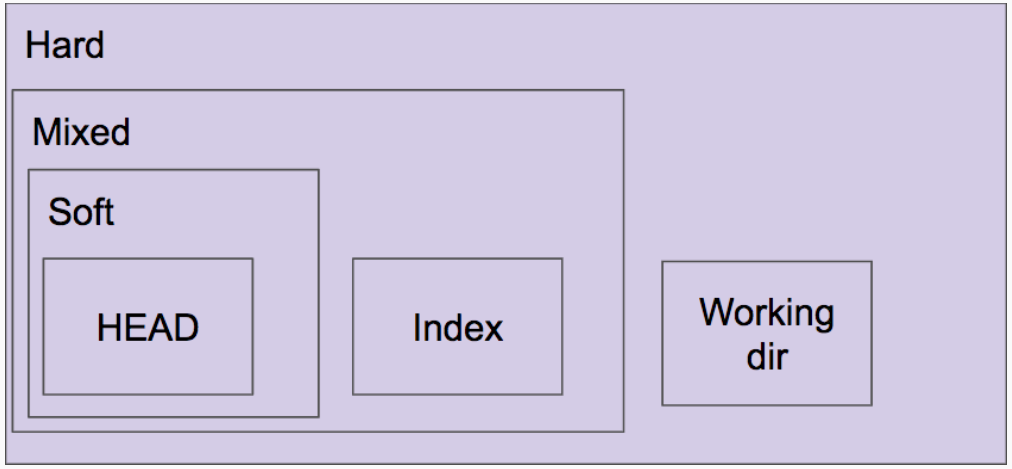


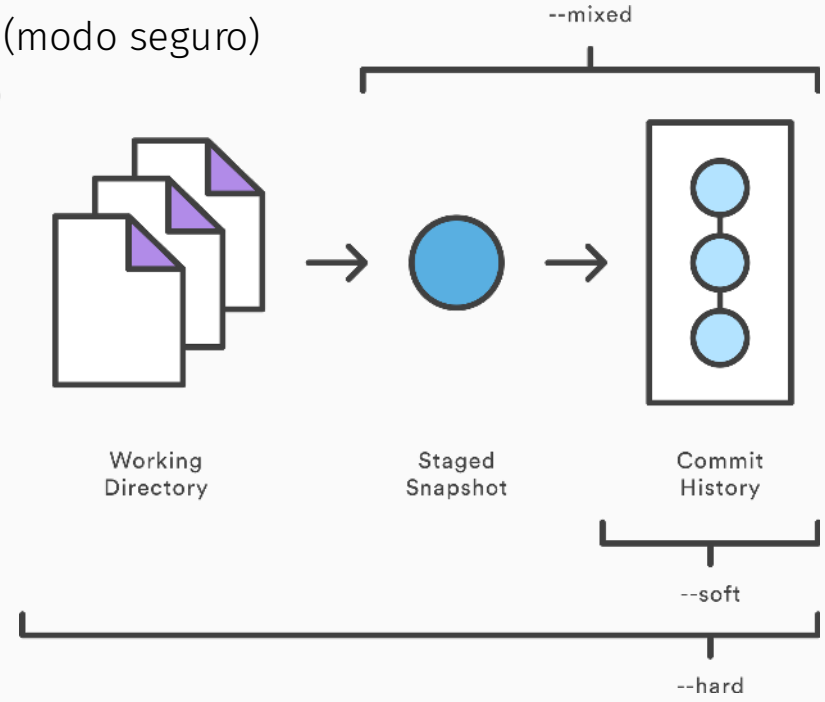
13. Desfazer: git revert e git reset

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 sobrescritas

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
```

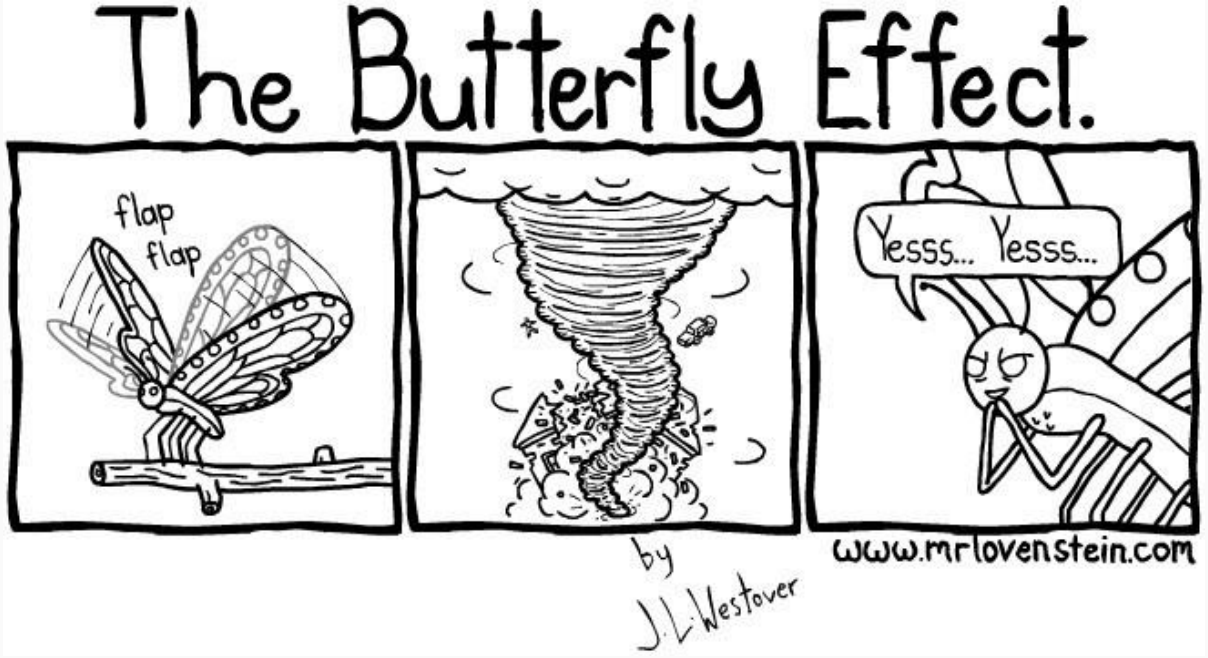
```
f9f2e9c (HEAD → master) mod2 teste.txt
dcbd894 mod teste.txt
8b33bc5 add teste.txt
```

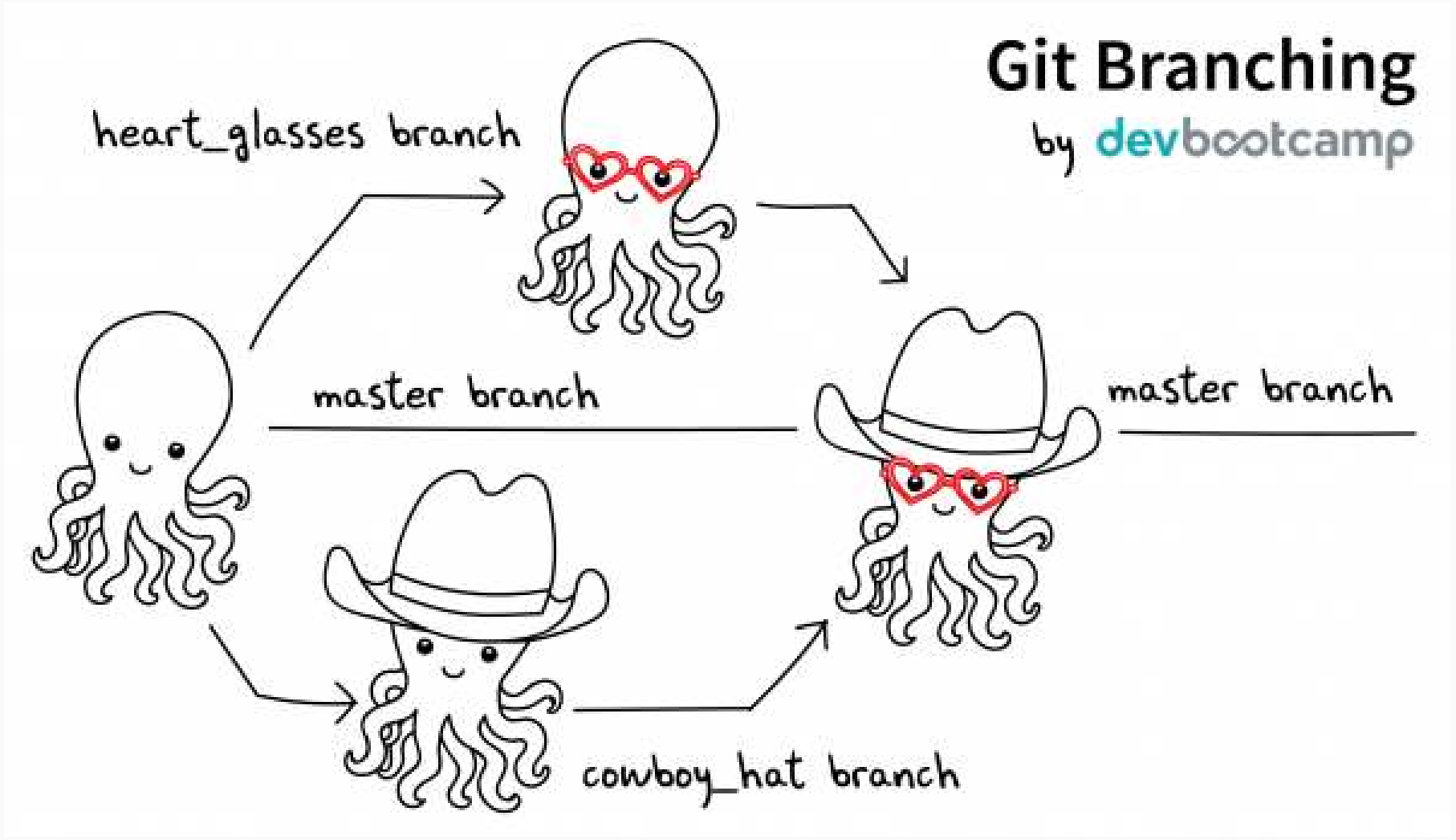
13. Desfazer: git revert e git reset

Desfazendo operações no git

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

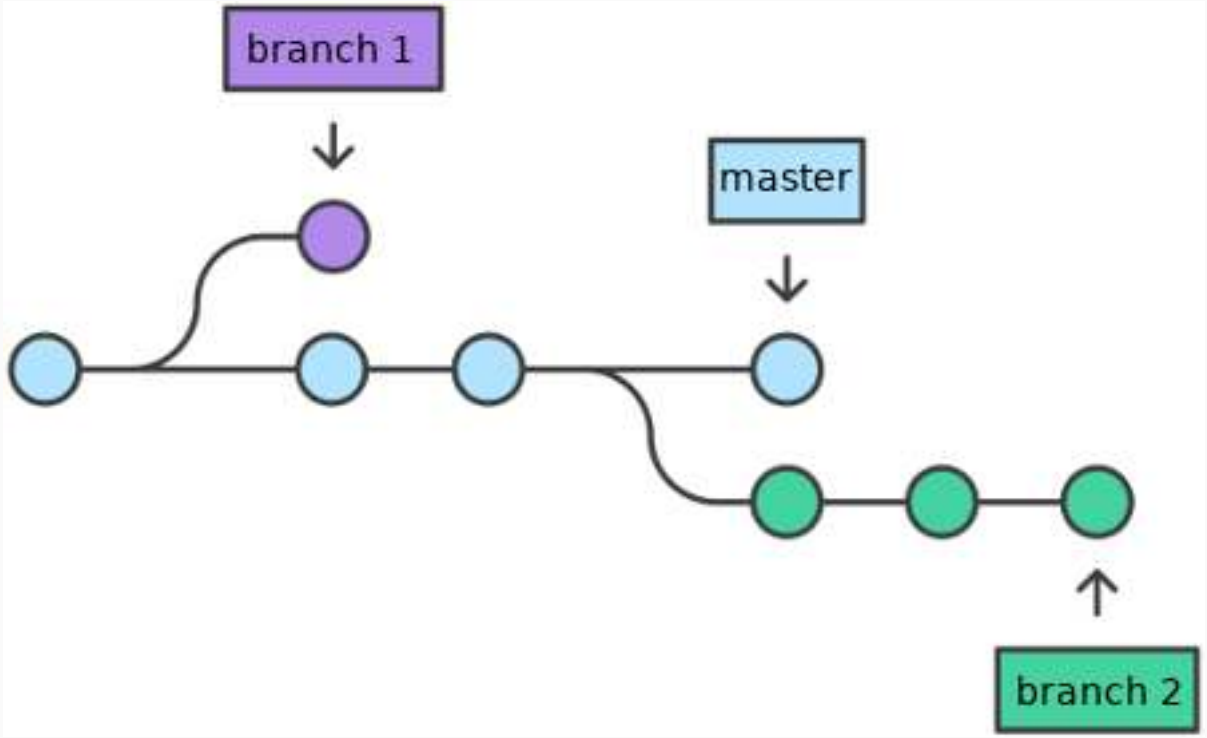
2. **git reset**: desfaz um commit **deletando o histórico** (deletando commits)




Git Branching
by devbootcamp
heart_glasses branch
master branch
master branch
cowboy_hat branch

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

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

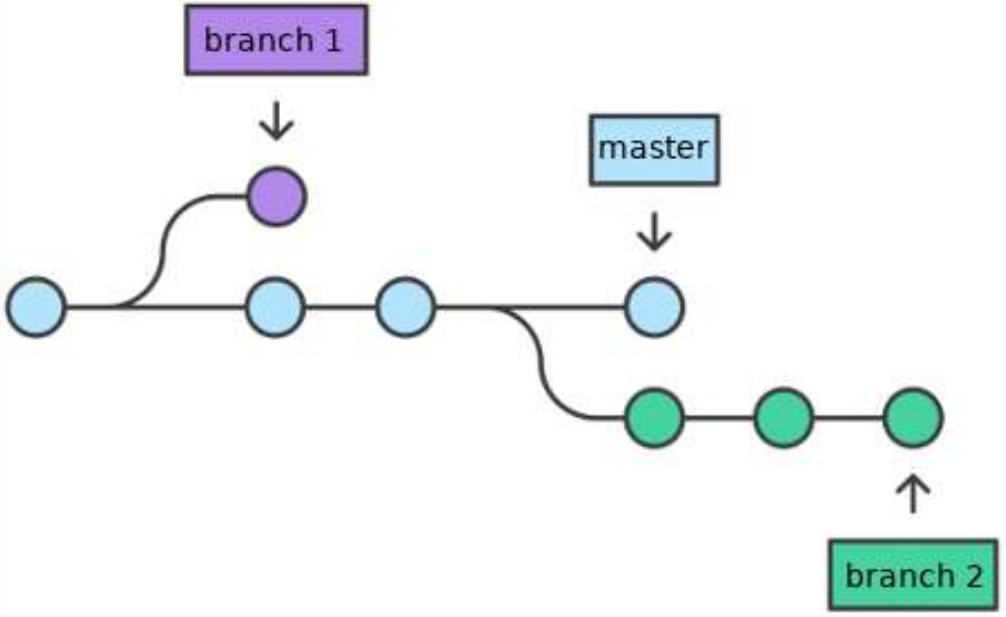


14. Ramificações: git branch, git switch e git merge

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

```
git branch
```

```
* master
```



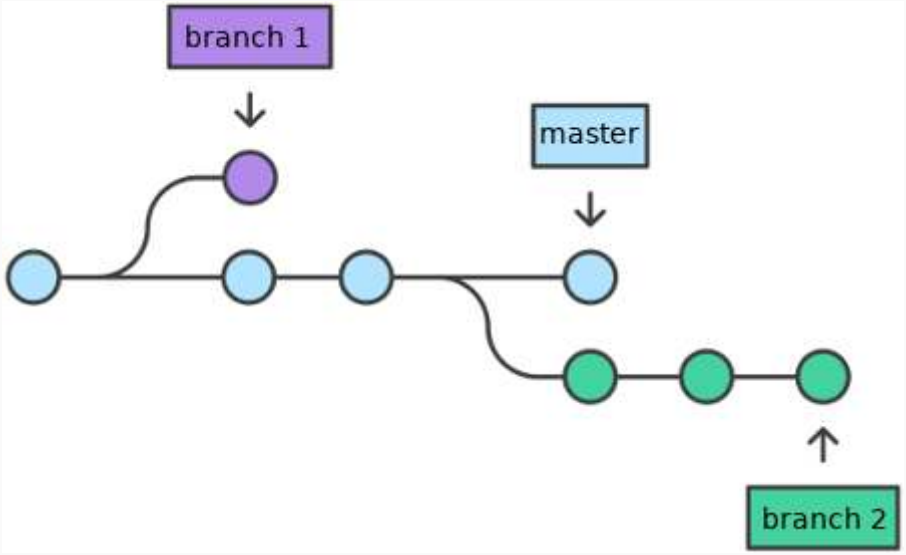
14. Ramificações: git branch, git switch e git merge

git branch: cria uma ramificação

```
git branch branch-1
```

```
git branch
```

```
branch-1
* master
```

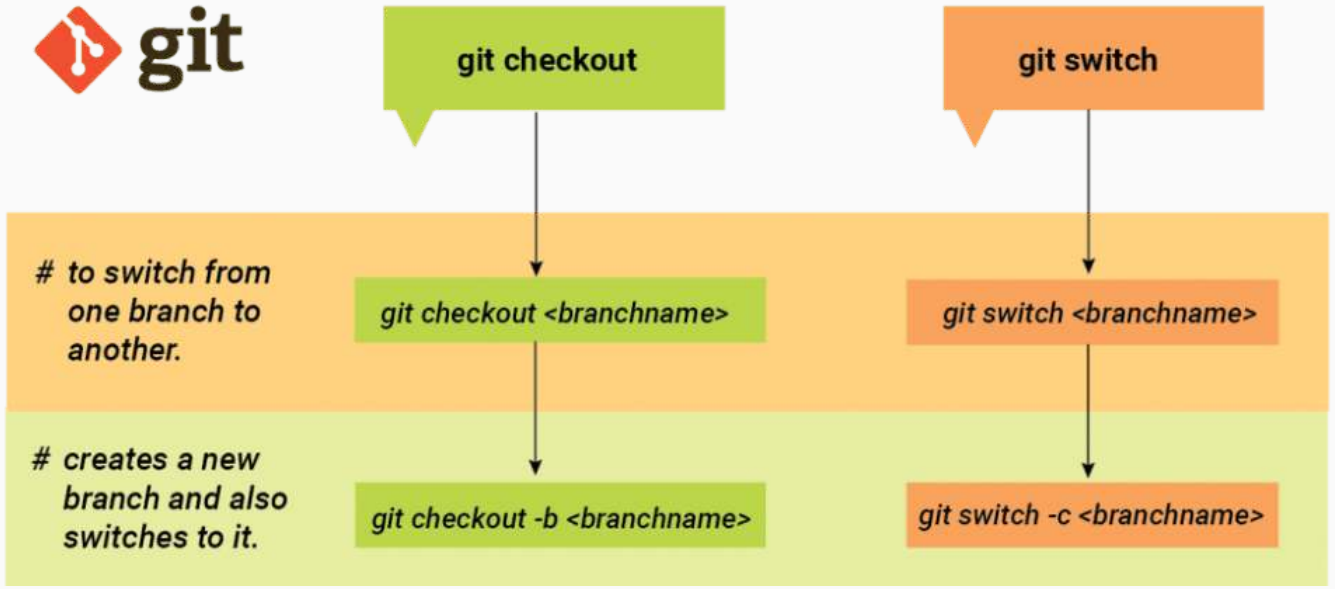


14. Ramificações: git branch, git switch e git merge

git switch: troca a raficação de edição

```
git switch branch-1
```

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

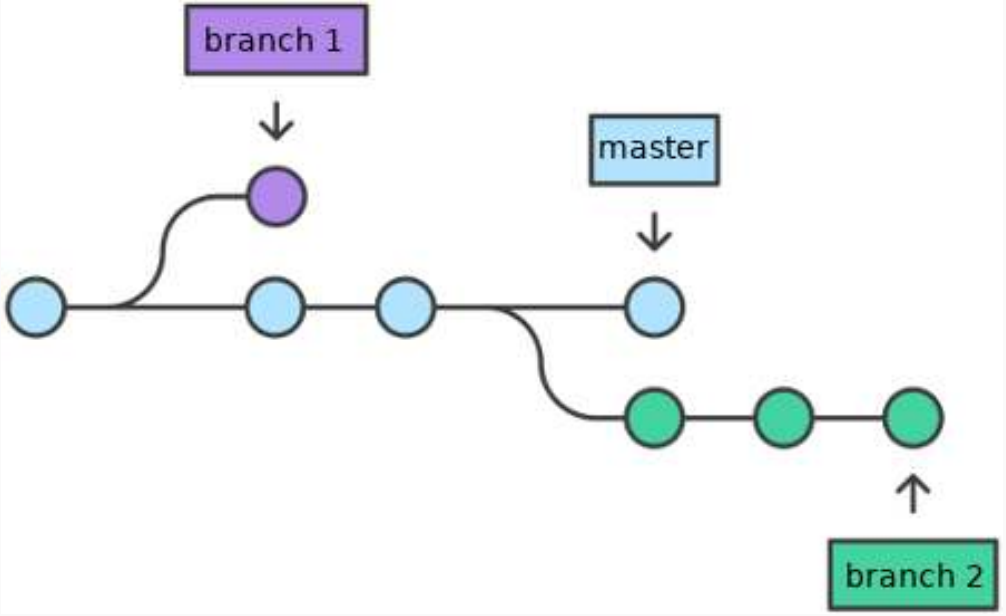


14. Ramificações: git branch, git switch e git merge

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

```
git branch
```

```
* branch-1
master
```



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

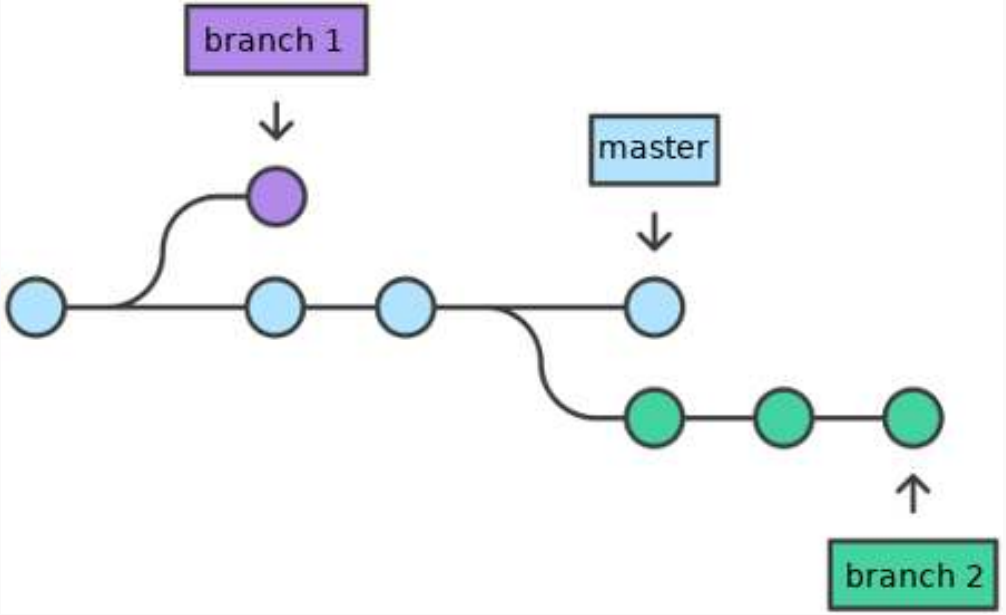
Vamos criar um arquivo no diretório

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

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

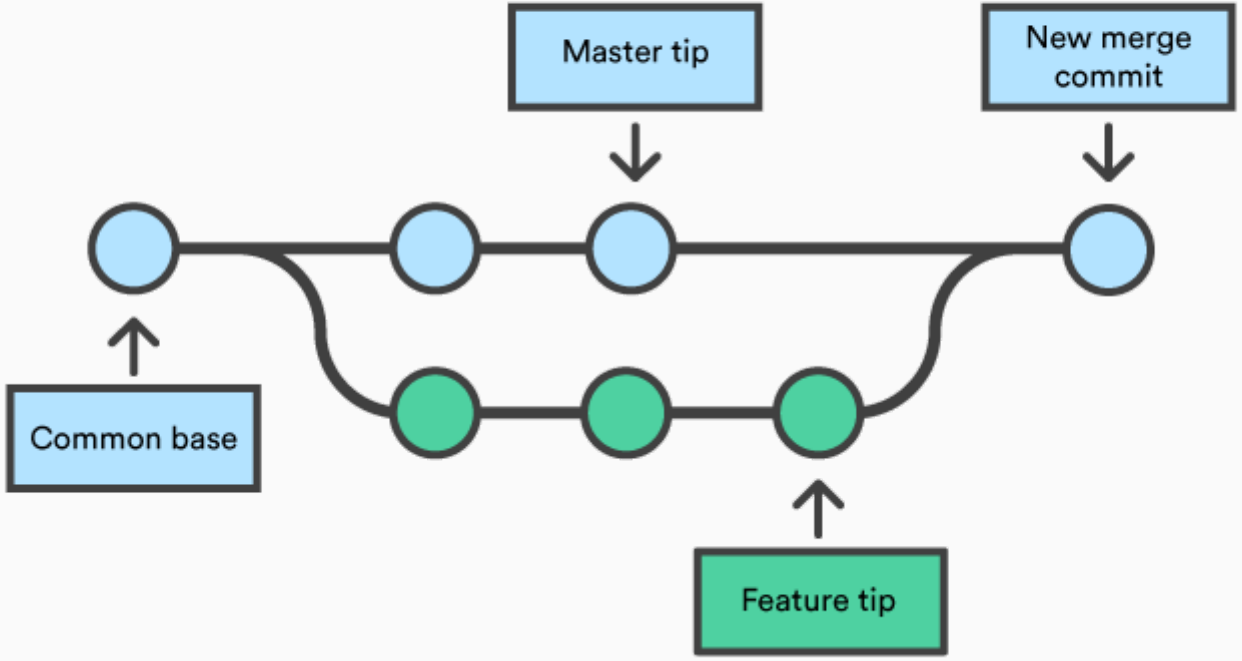
switch: troca a raficação de edição

```
git switch master
```



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

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

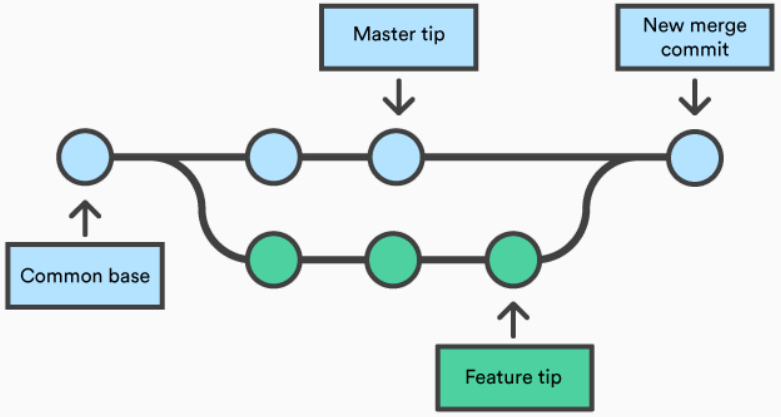


14. Ramificações: git branch, git switch e git merge

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
```

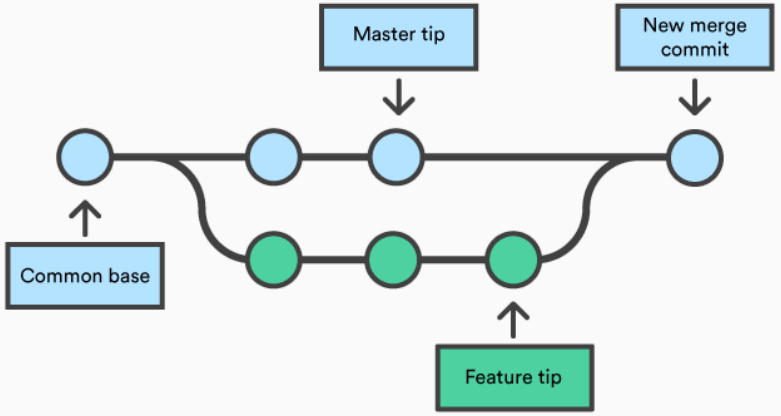


14. Ramificações: git branch, git switch e git merge

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
```

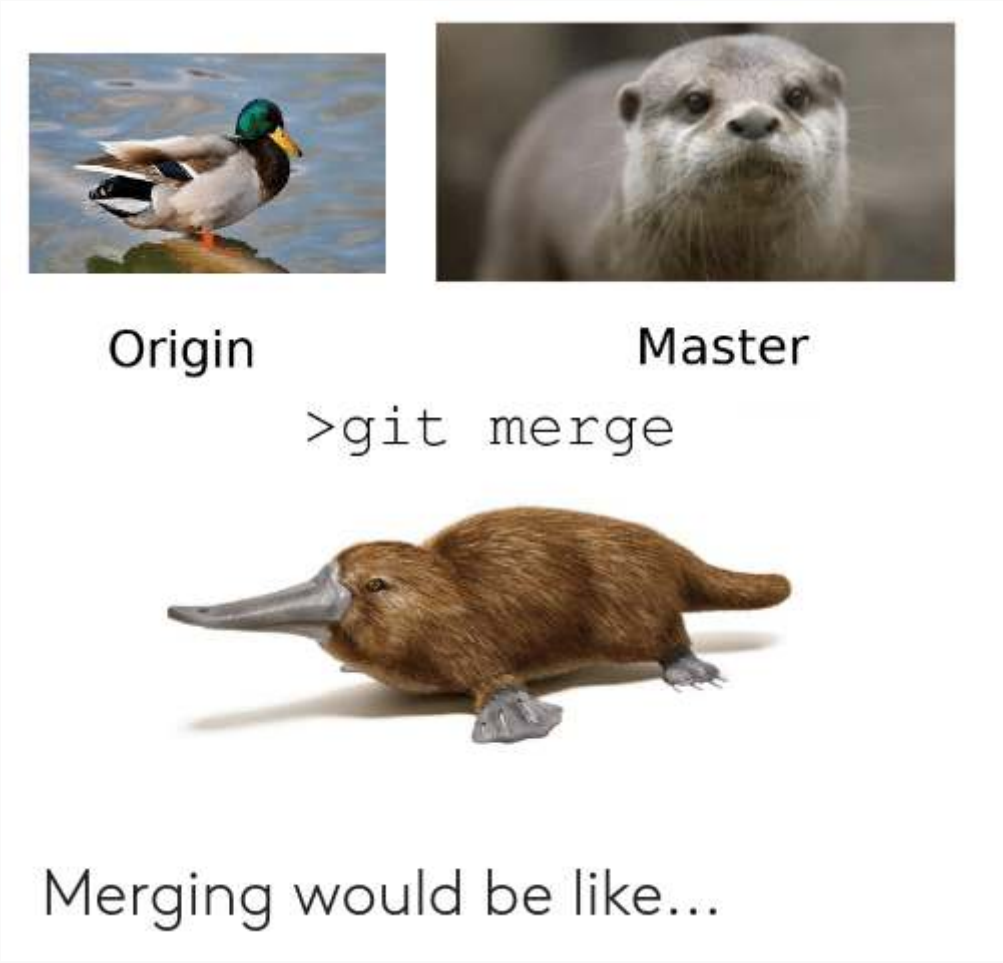


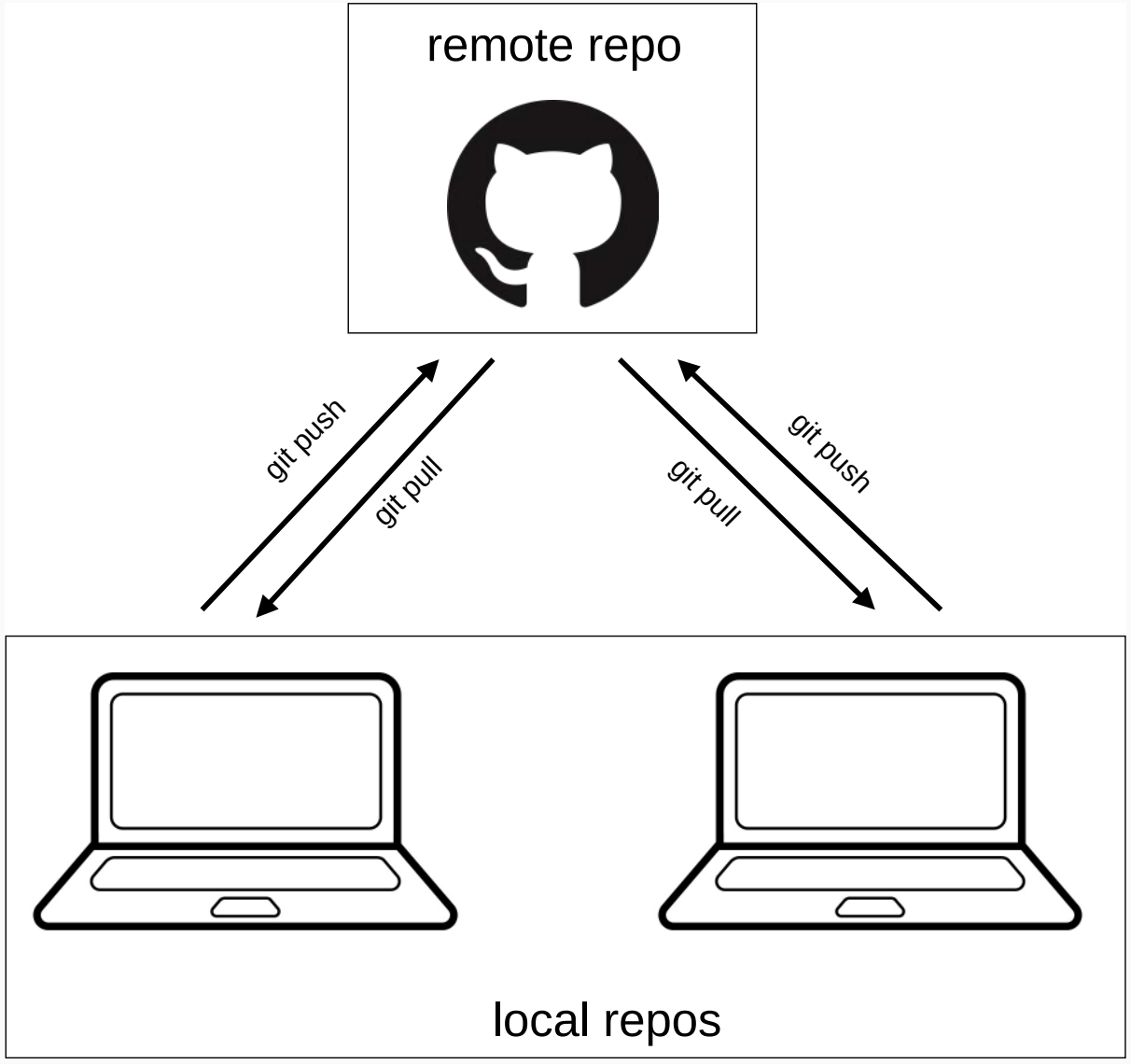
14. Ramificações: git branch, git switch e git merge

CUIDADO!

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

CUIDADO!



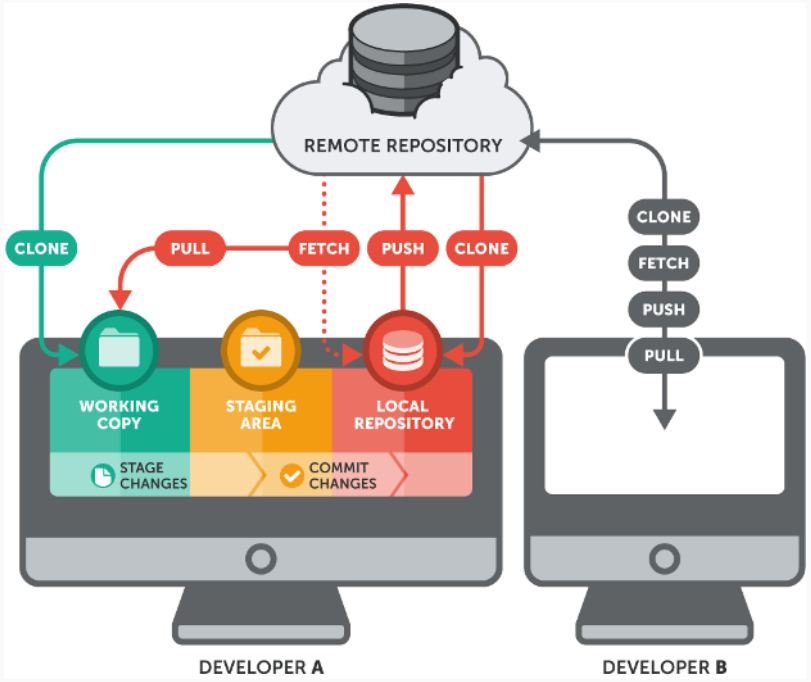

remote repo
git push
git pull
git pull
git push
local repos

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)
```

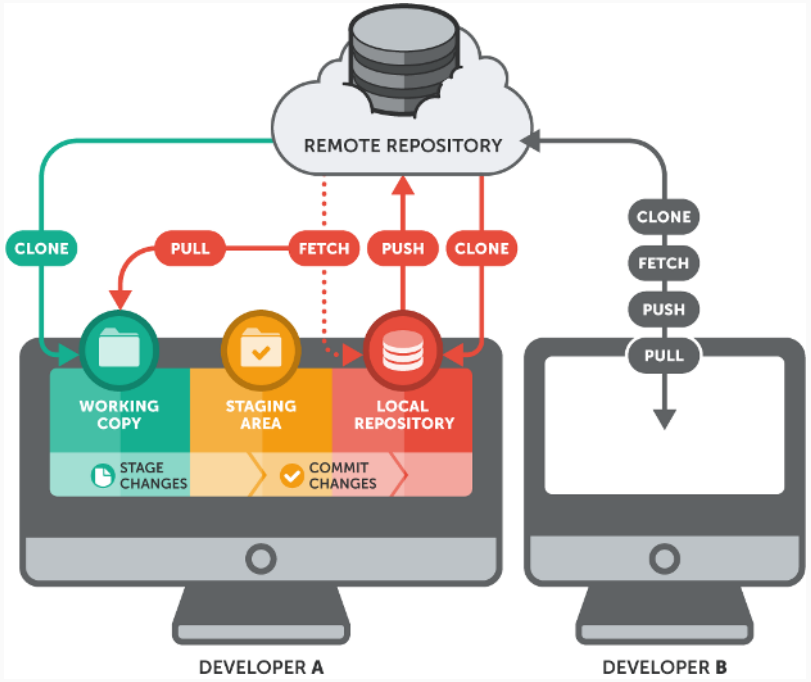


15. Remoto: git remote, git push e git pull

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

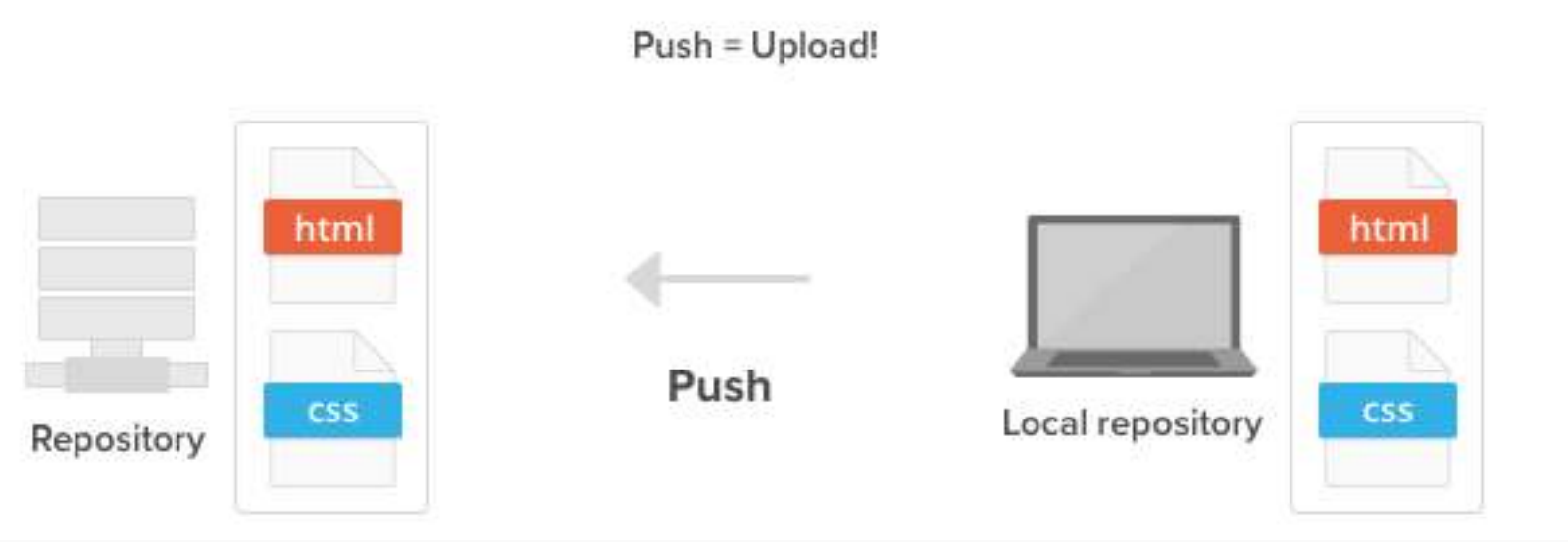
Observação: Usar o SSH para não precisar digitar sua senha!

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



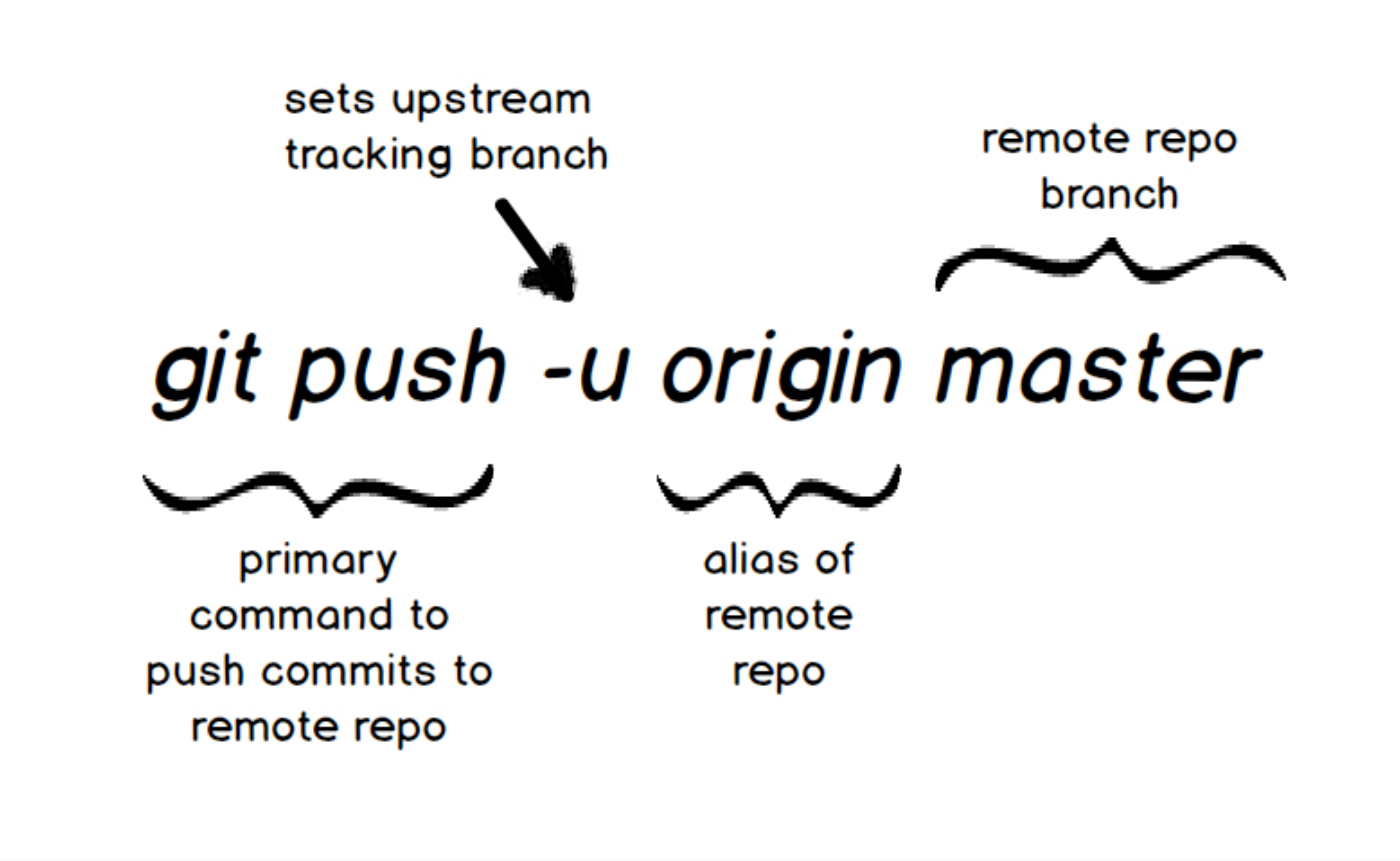
15. Remoto: git remote, git push e git pull

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

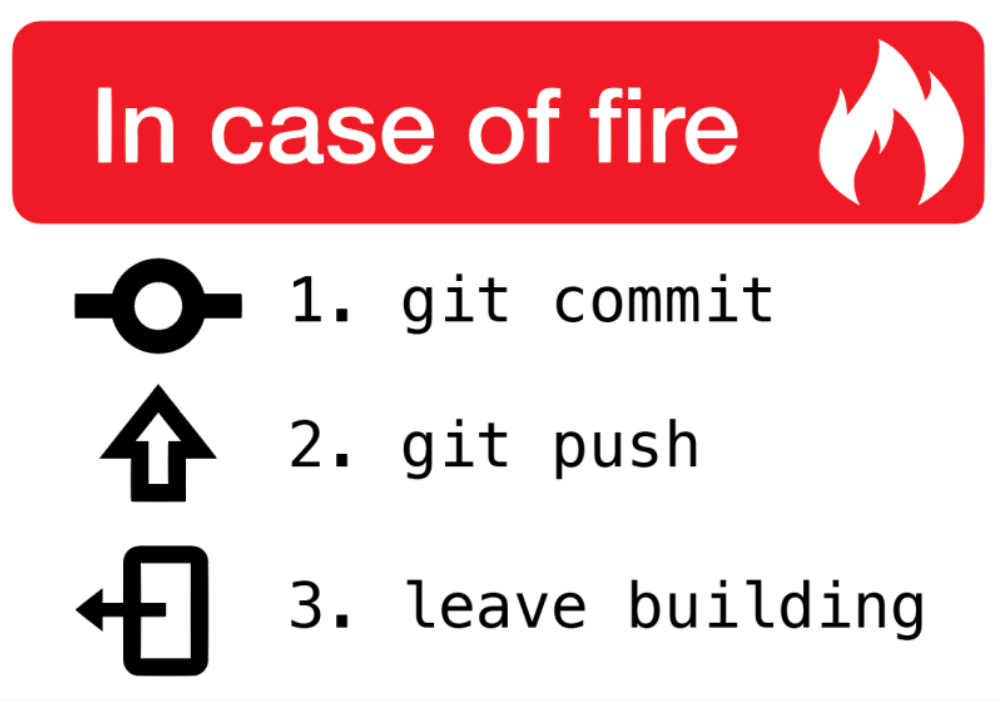
git push: empurra uma nova versão do repositório local para o repositório remoto (GitHub)

```
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...

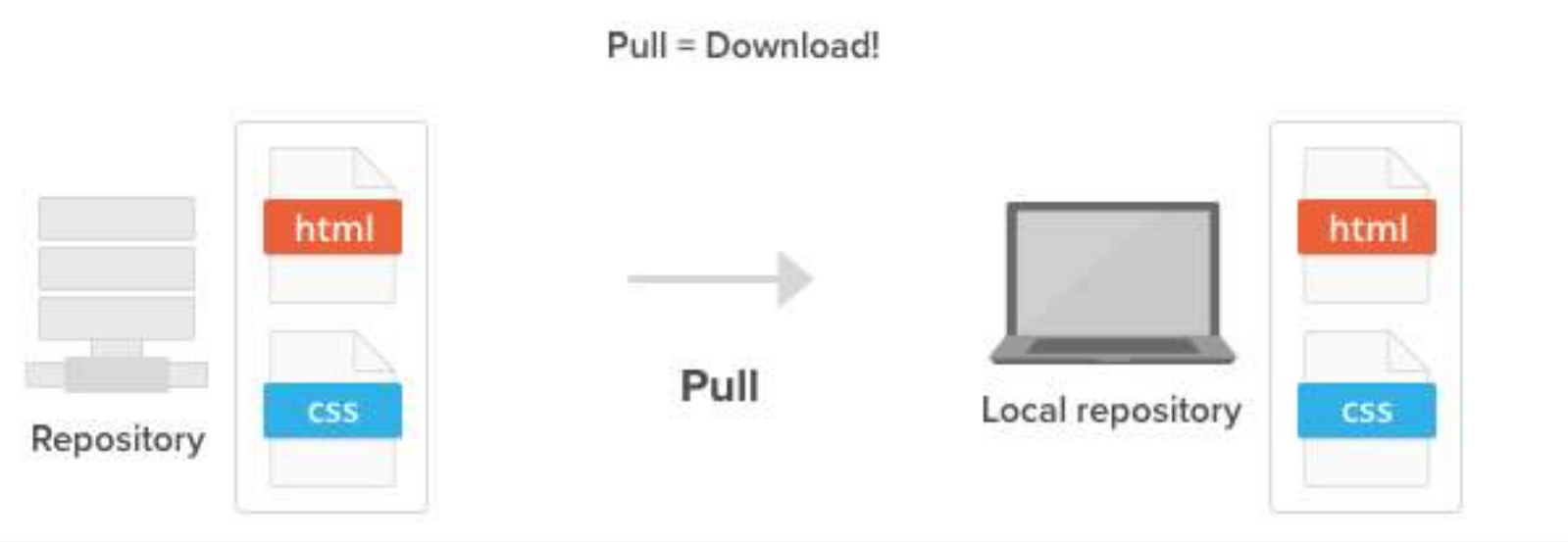


15. Remoto: git remote, git push e git pull

CUIDADO!

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

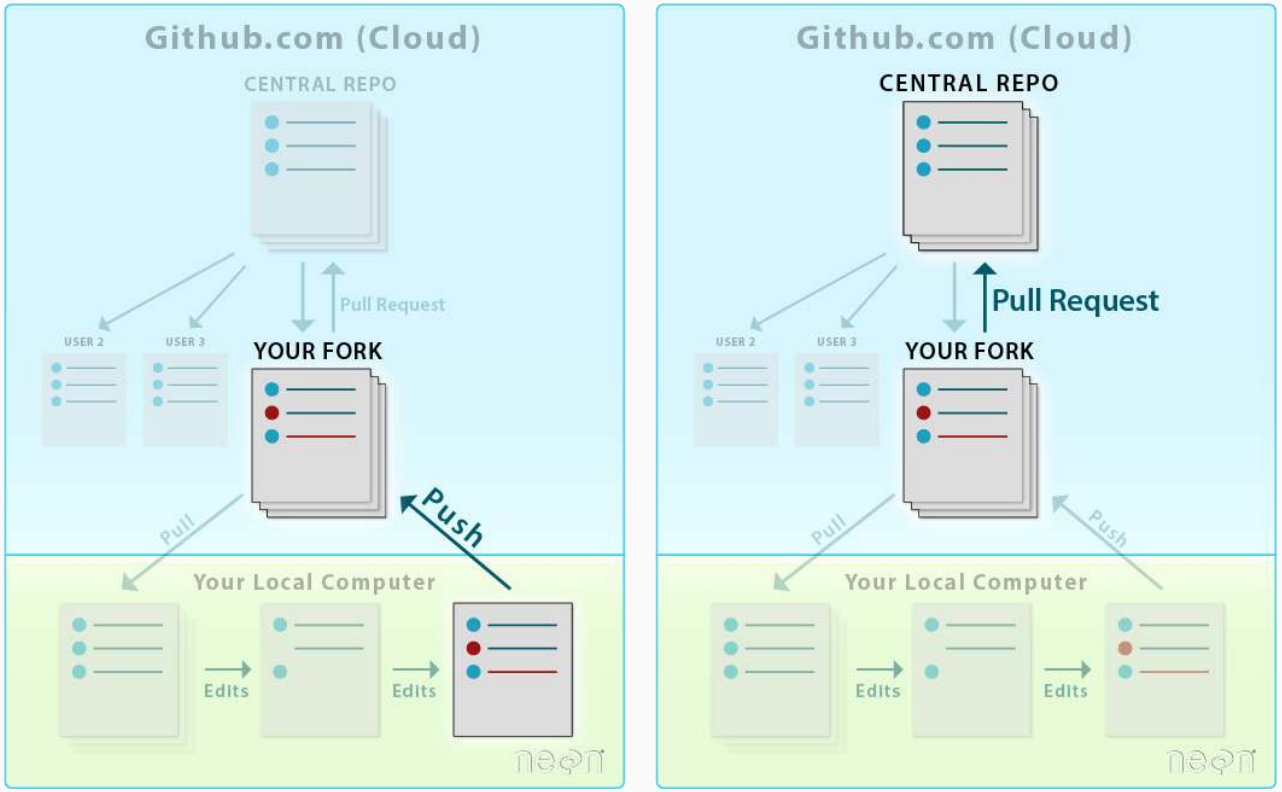
git pull: puxa uma nova versão do repositório remoto (GitHub) para o repositório local



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

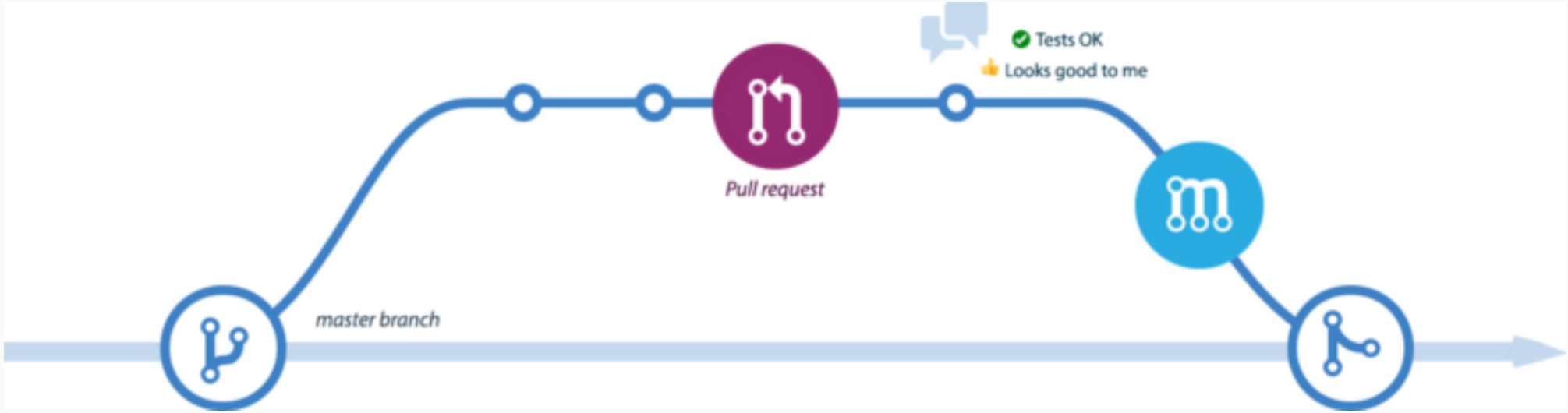
16. GitHub: Pull request

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



16. GitHub: Pull request

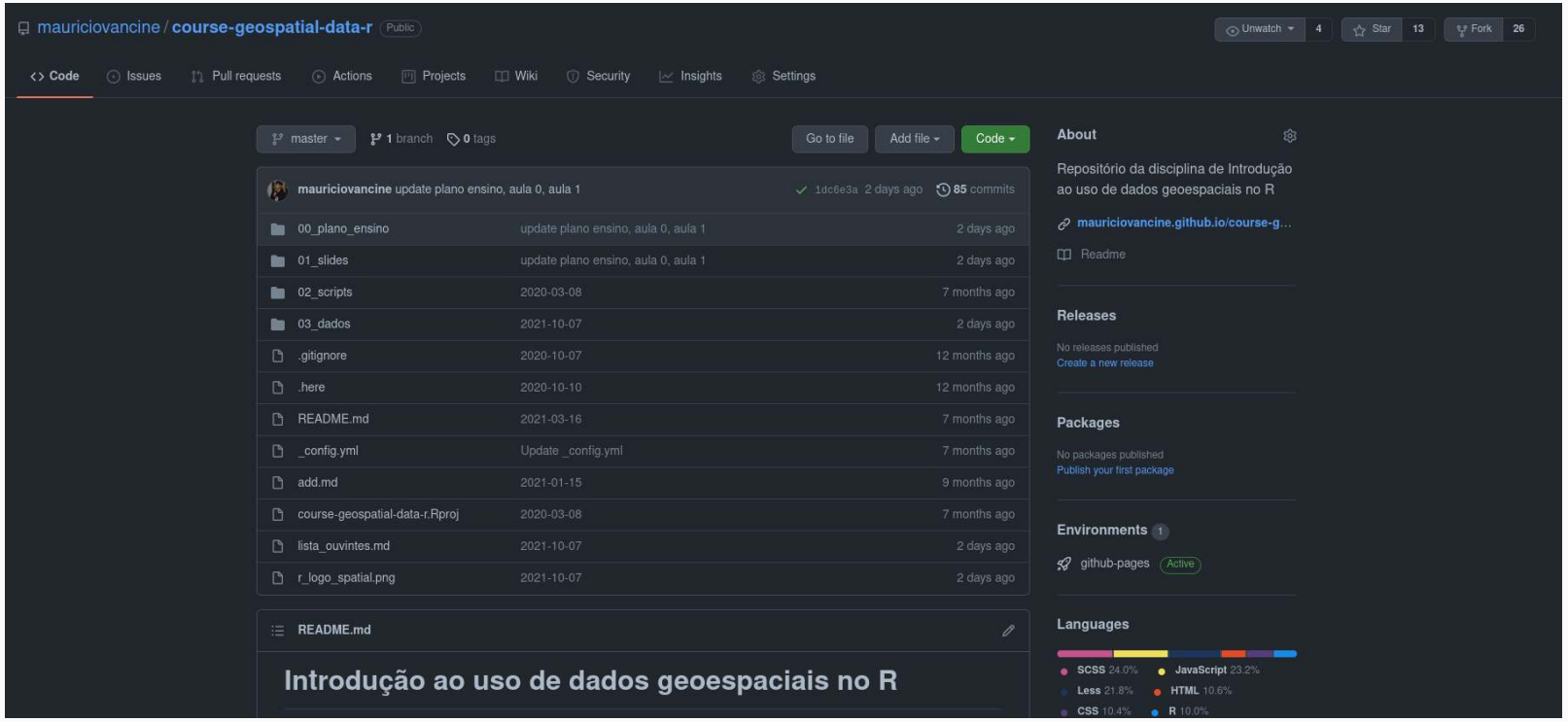
Pull Request: solicita que o repositório central (remoto) aceite (pull - 'puxe') as alterações realizadas do nosso fork



17. Detalhes do repositório do GitHub

Vamos detalhar alguns pontos dos repositórios no GitHub

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

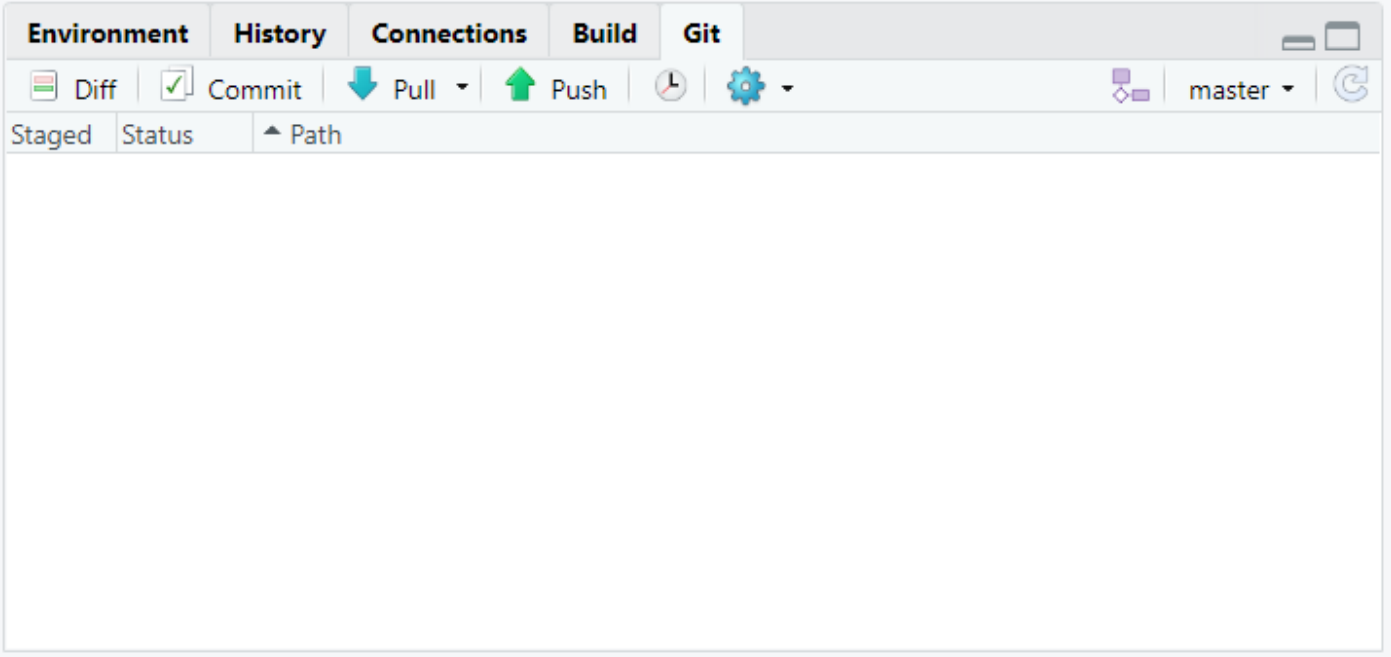


Agradecimento à Beatriz Milz pelas figuras ~~surrupiadas~~
(eu pedi antes =])

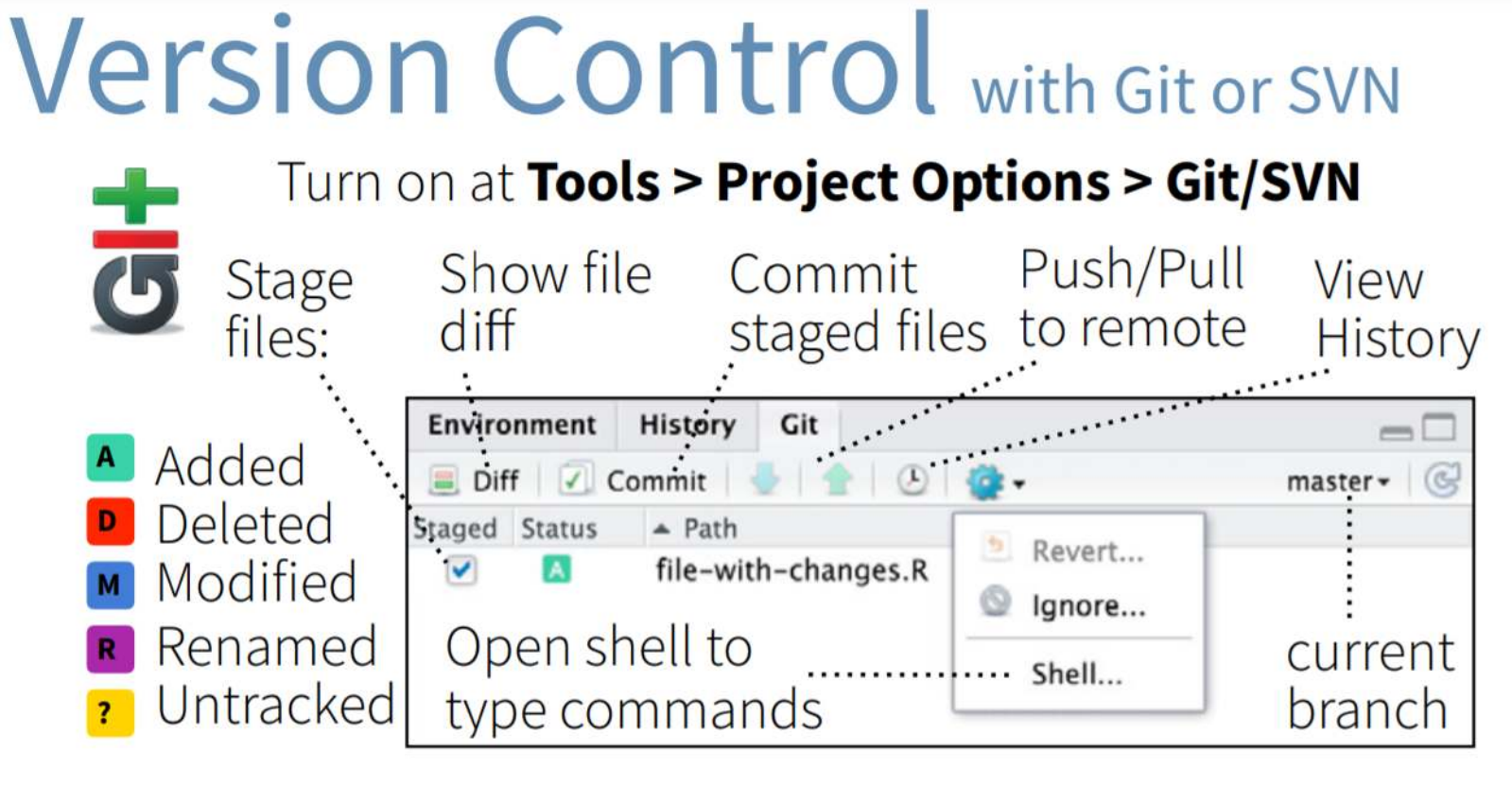
18. Interface Gráfica do RStudio

Git Panel

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

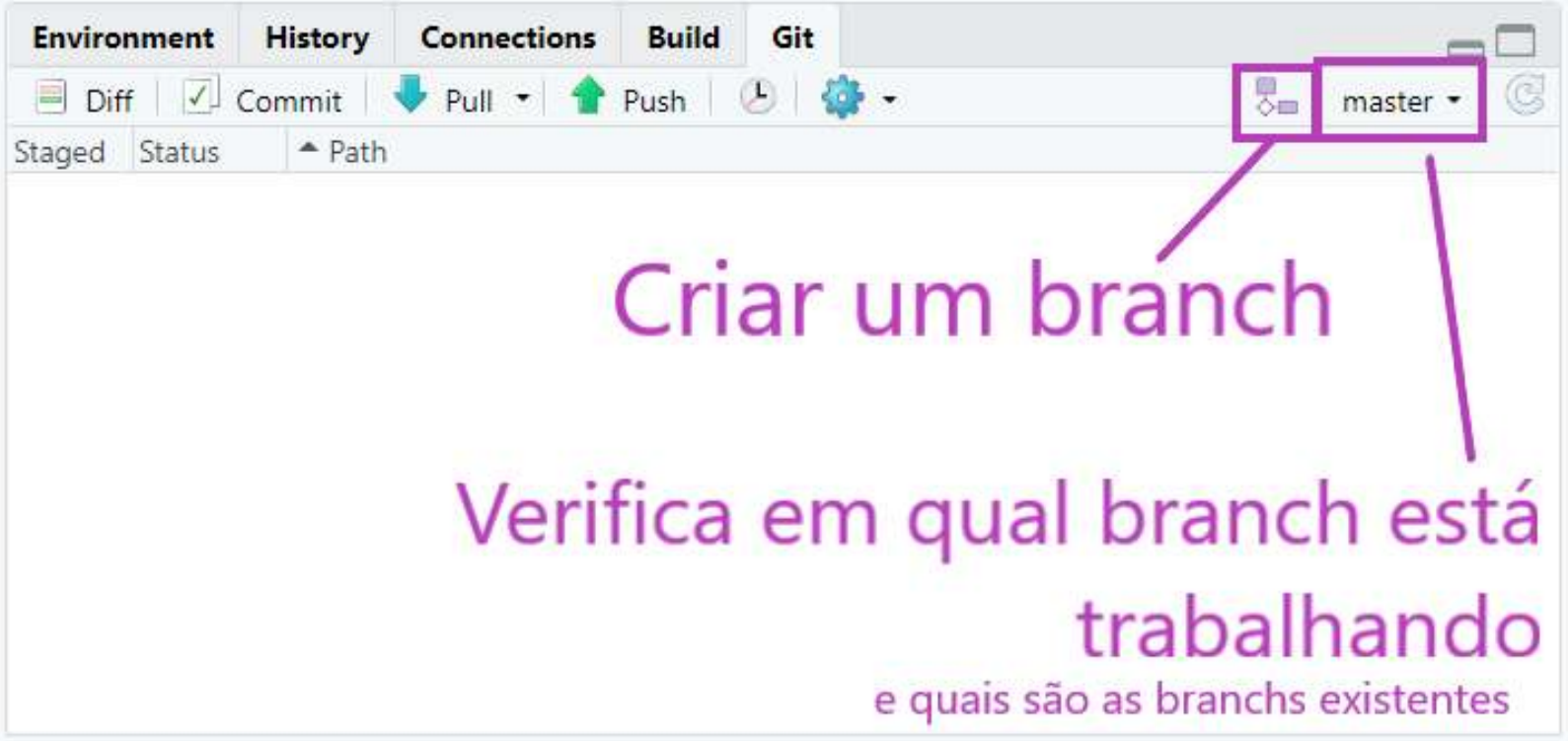


Git Panel - Detalhes



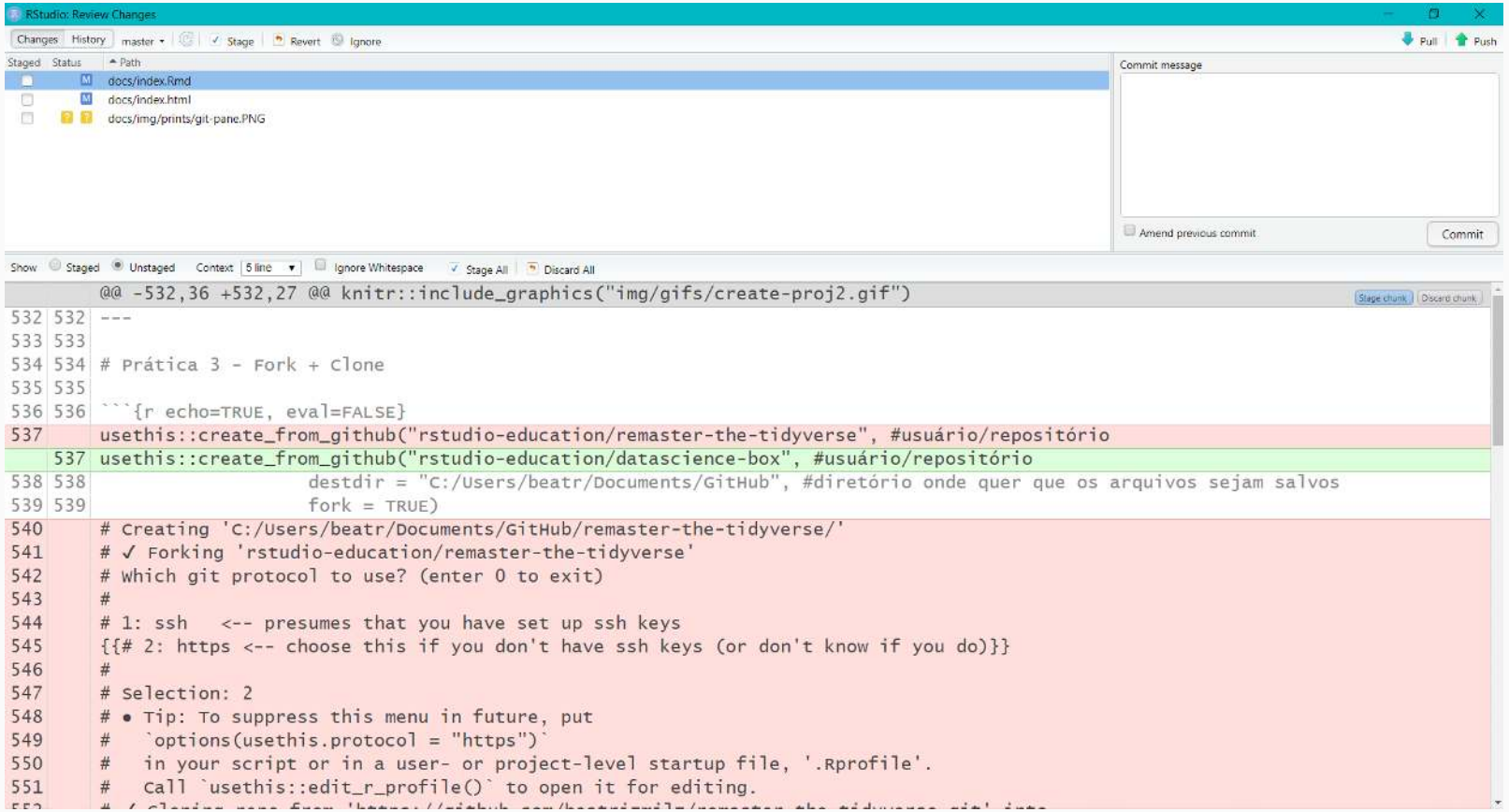
18. Interface Gráfica do RStudio

Git Panel - Branches



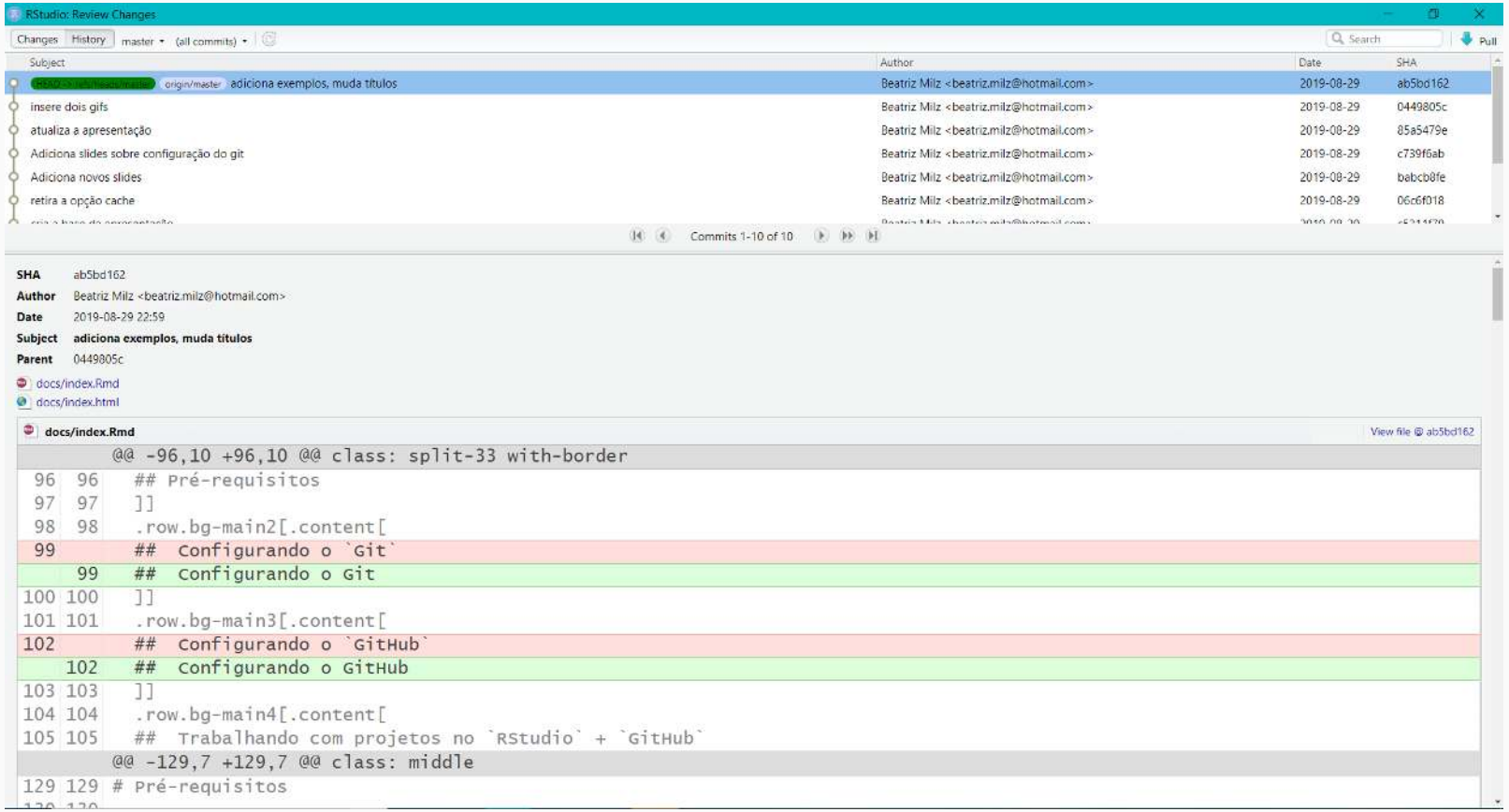
18. Interface Gráfica do RStudio

Git Panel - Diff - Changes: Revisar mudanças



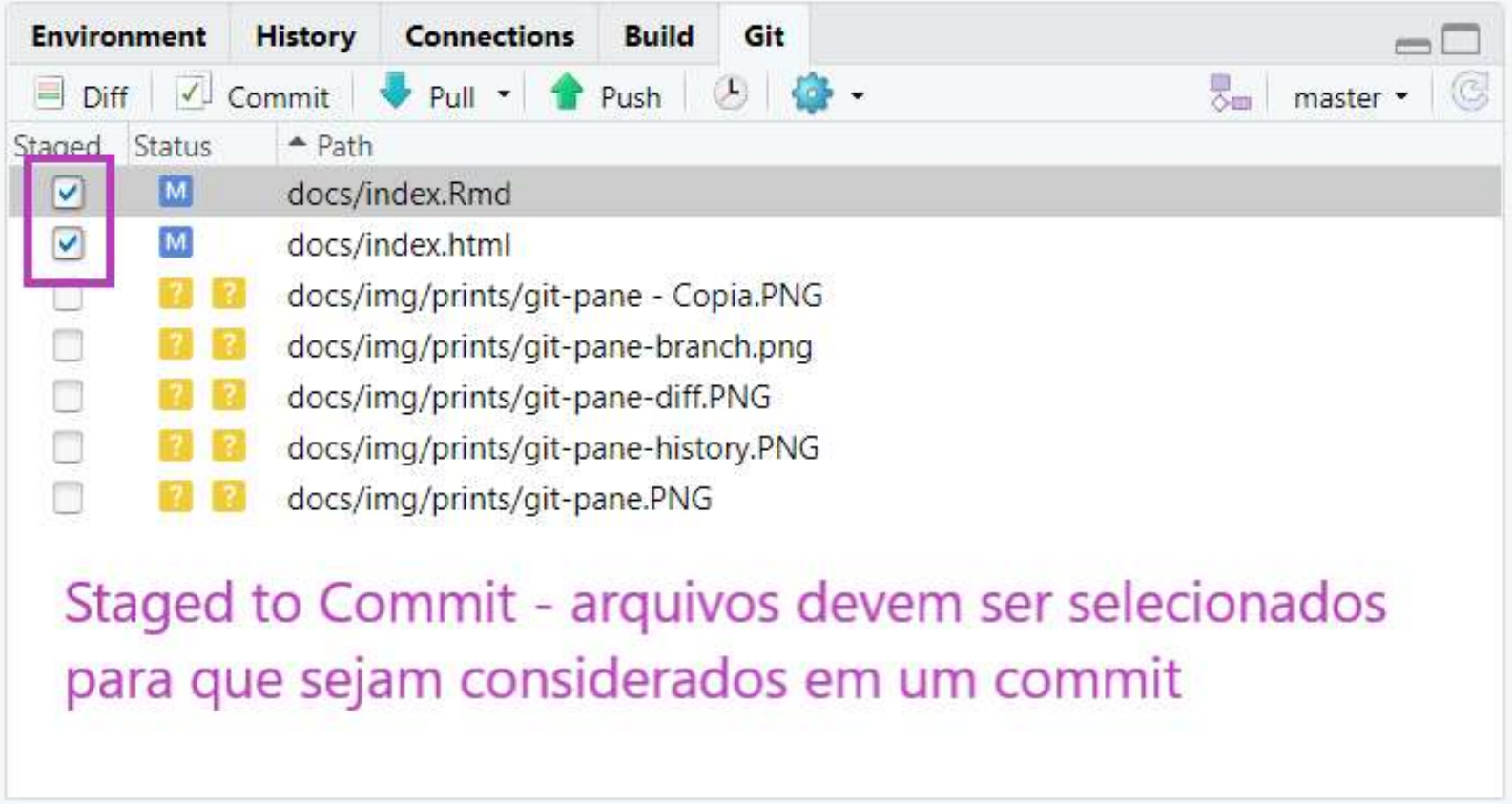
18. Interface Gráfica do RStudio

Git Panel - Diff - History: Histórico de mudanças

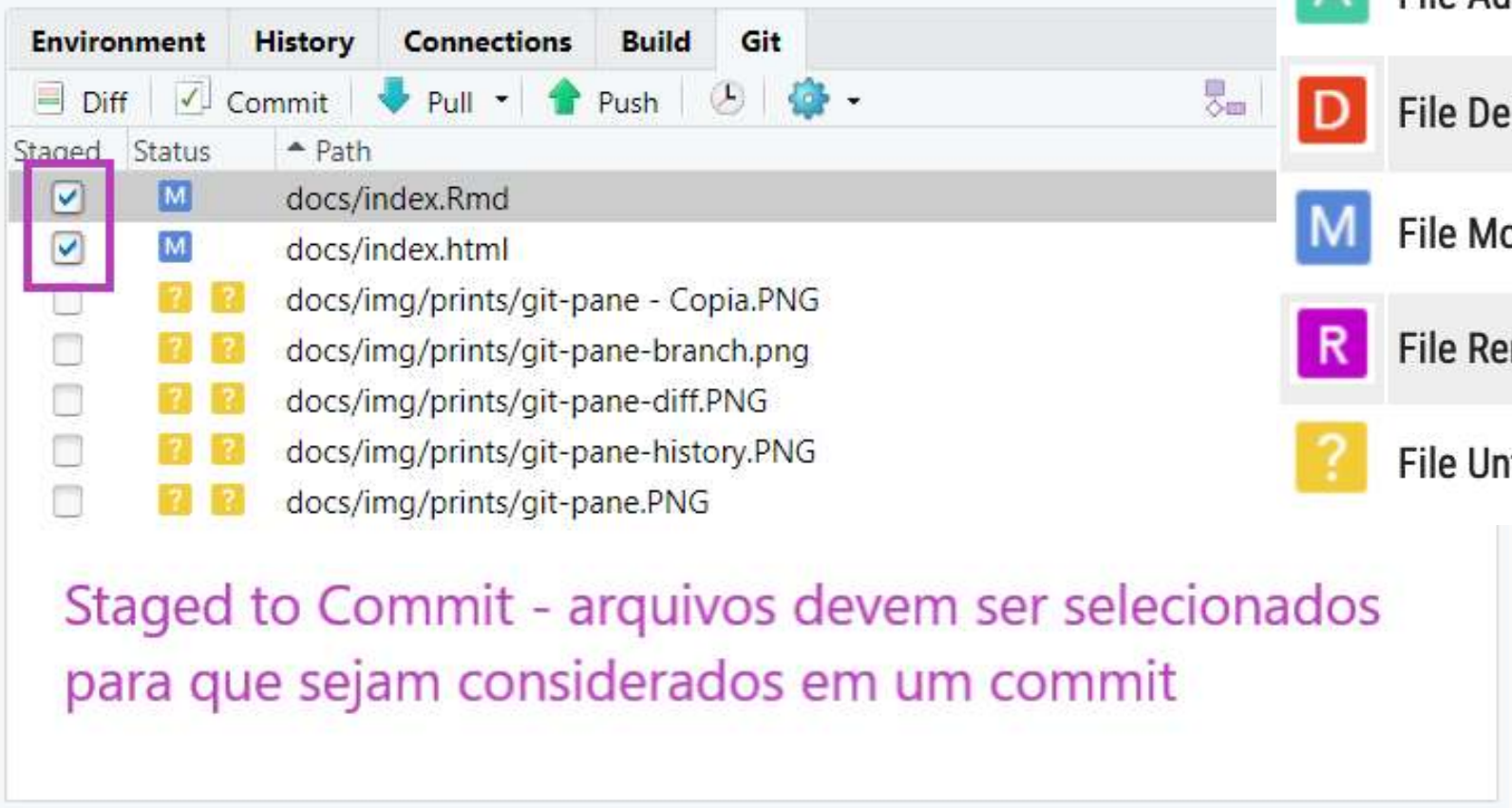


18. Interface Gráfica do RStudio

Git pane - Staged

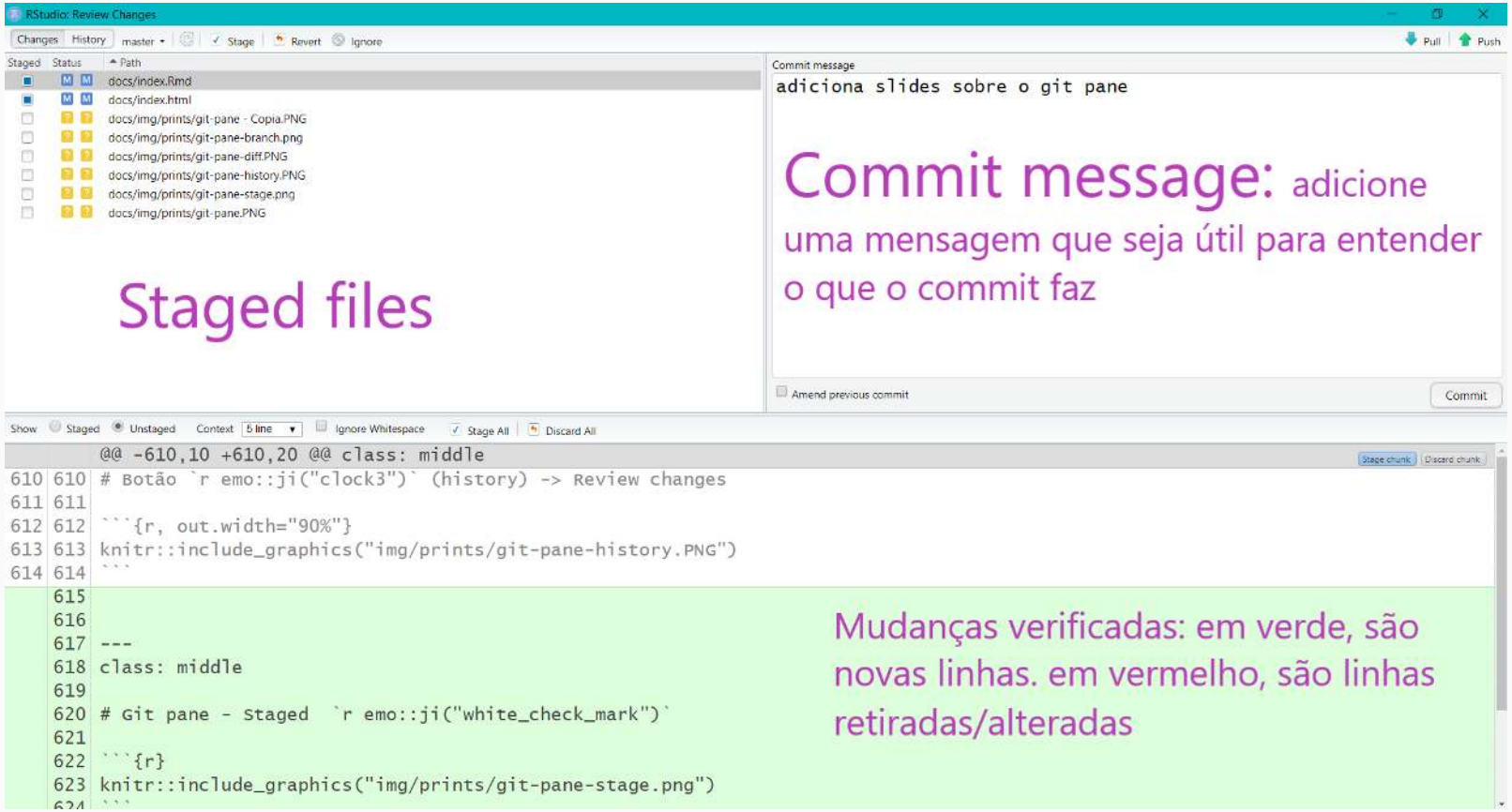


Git pane - Staged e File status



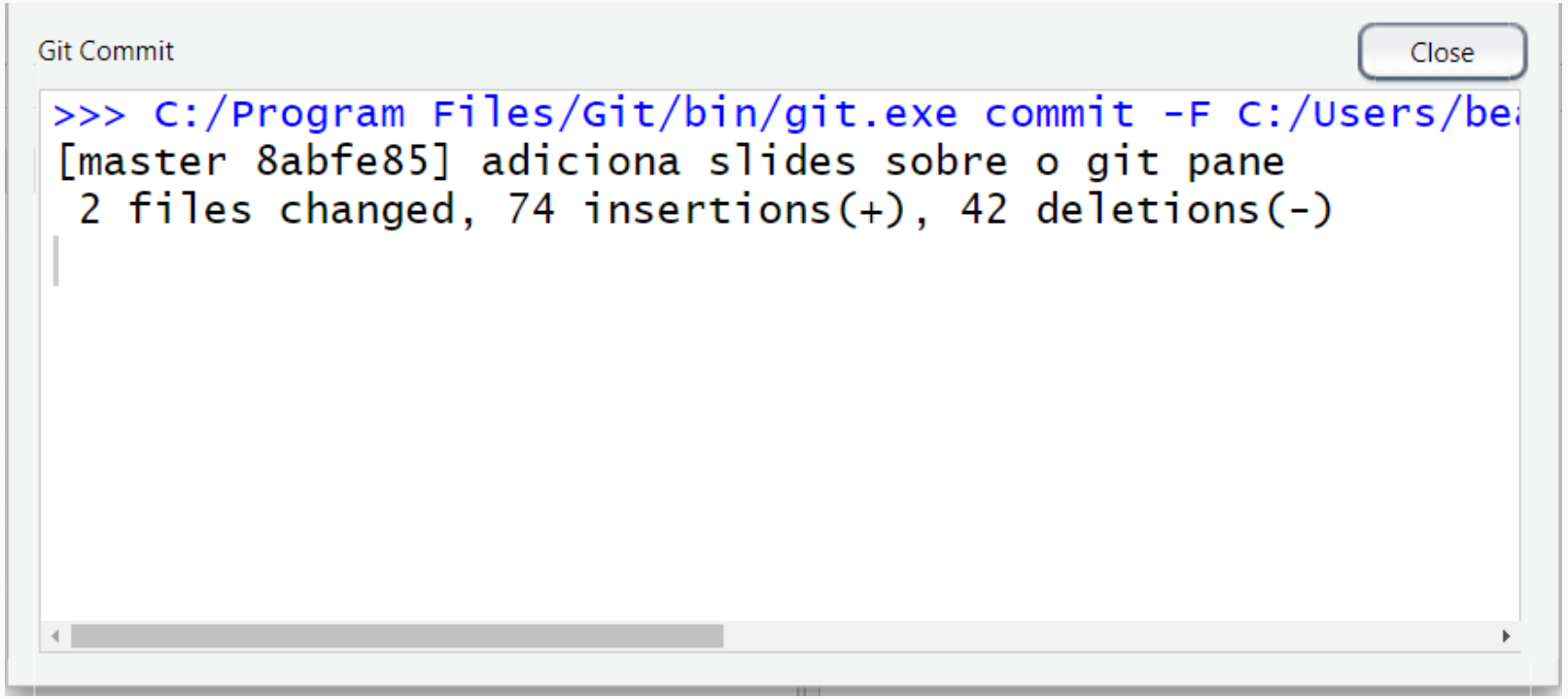
18. Interface Gráfica do RStudio

Git pane - Commit



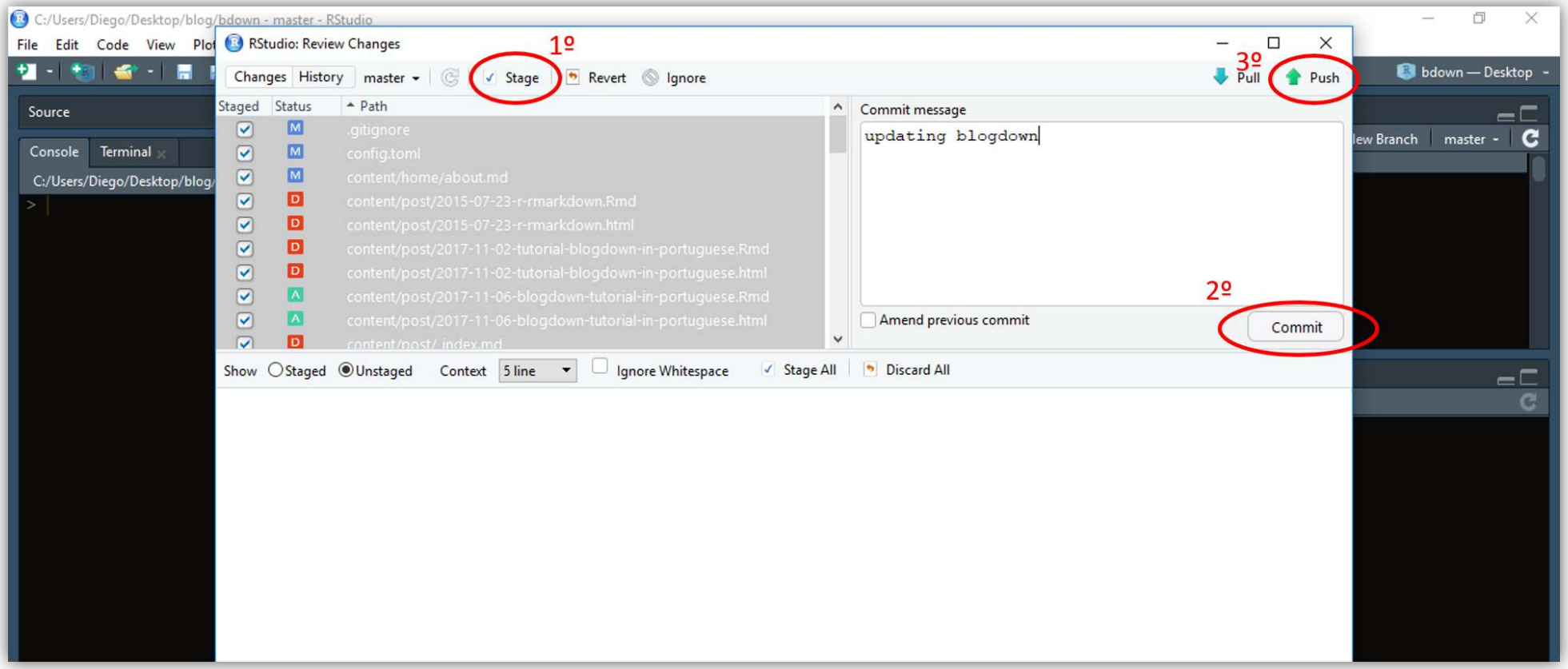
18. Interface Gráfica do RStudio

Git pane - Commit



18. Interface Gráfica do RStudio

Git pane - Push e Pull



19. Principal material de estudo

Git Cheat Sheets

GitHub

GIT CHEAT SHEET

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

INSTALL GIT

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Git for All Platforms
http://git-scm.com

CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```
Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```
Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```
Enables helpful colorization of command line output

CREATE REPOSITORIES

Start a new repository or obtain one from an existing URL

```
$ git init [project-name]
```
Creates a new local repository with the specified name

```
$ git clone [url]
```
Downloads a project and its entire version history

MAKE CHANGES

Review edits and craft a commit transaction

```
$ git status
```
Lists all new or modified files to be committed

```
$ git diff
```
Shows file differences not yet staged

```
$ git add [file]
```
Snapshots the file in preparation for versioning

```
$ git diff --staged
```
Shows file differences between staging and the last file version

```
$ git reset [file]
```
Unstages the file, but preserve its contents

```
$ git commit -m "[descriptive message]"
```
Records file snapshots permanently in version history

GROUP CHANGES

Name a series of commits and combine completed efforts

```
$ git branch
```
Lists all local branches in the current repository

```
$ git branch [branch-name]
```
Creates a new branch

```
$ git checkout [branch-name]
```
Switches to the specified branch and updates the working directory

```
$ git merge [branch]
```
Combines the specified branch's history into the current branch

```
$ git branch -d [branch-name]
```
Deletes the specified branch

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

mauriciovancine

mauriciovancine.github.io

Slides criados via pacote xaringan e tema Metropolis. Animação dos sapos por @probzz.