

ERIC BRASIL  
(IHL-UNILAB e IHC - NOVA  
FCSH / IN2PAST)

# GIT PARA HISTORIADORES E HISTORIADORAS





# OBJETIVOS

Caracterizar sistemas de controle de versões (SCV) e analisar as possibilidades de seu uso para a pesquisa em história.

Git: instalar o Git, suas funções básicas, seu fluxo de trabalho e como recuperar informações.

O foco é demonstrar na prática as potencialidades do Git como um programa de suporte metodológico.

# ESTRUTURA DO WORKSHOP

## O QUE É UM SCV?

- . Porque controle de versões?
- . Centralizado ou Distribuído?
- . Vantagens do Git

## O QUE É O GIT?

- . História
- . Fluxo de trabalho
- . Instalação

## UTILIZANDO O GIT

- . Configurações gerais
- . Comandos básicos
- . Estágios de um ficheiro
- . Recuperando informações

# "FINAL".doc



↑ FINAL.doc!

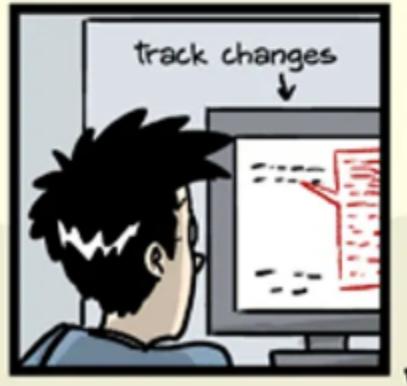


↑ FINAL\_rev.2.doc



↑ FINAL\_rev.6.COMMENTS.doc

↑ FINAL\_rev.8.comments5.  
CORRECTIONS.doc



↑ FINAL\_rev.18.comments7.  
corrections9.MORE.30.doc

↑ FINAL\_rev.22.comments49.  
corrections.10.#@\$%WHYDID  
ICOMETOGRAD SCHOOL?????.doc

# JÁ FEZ ISSO?

"FINAL".doc de [PhDComics](#)



# QUAL NOSO MÉTODO DE CONTROLE DE VERSÕES?

James Baker, "Preservar os seus dados de investigação", traduzido por Márcia T. Cavalcanti, Programming Historian em português 1 (2021),  
<https://doi.org/10.46430/phpt0001>



# SISTEMA DE CONTROLE DE VERSÃO [SCV]

Um SCV consiste em um sistema que registra as mudanças de um ficheiro ou conjunto de ficheiros ao longo do tempo, cada uma dessas mudanças é acompanhada de um conjunto de metadados (ou seja, informações sobre os dados), e te permite recuperar tanto esses dados quanto o estado em que se encontrava o seu projeto há época.

Ao utilizar um SCV você é capaz de acompanhar, documentar, recuperar e corrigir as etapas do projeto de pesquisa. Também é possível acompanhar trabalhos de alunos ou equipe que compõe um projeto

# SCV CENTRALIZADO

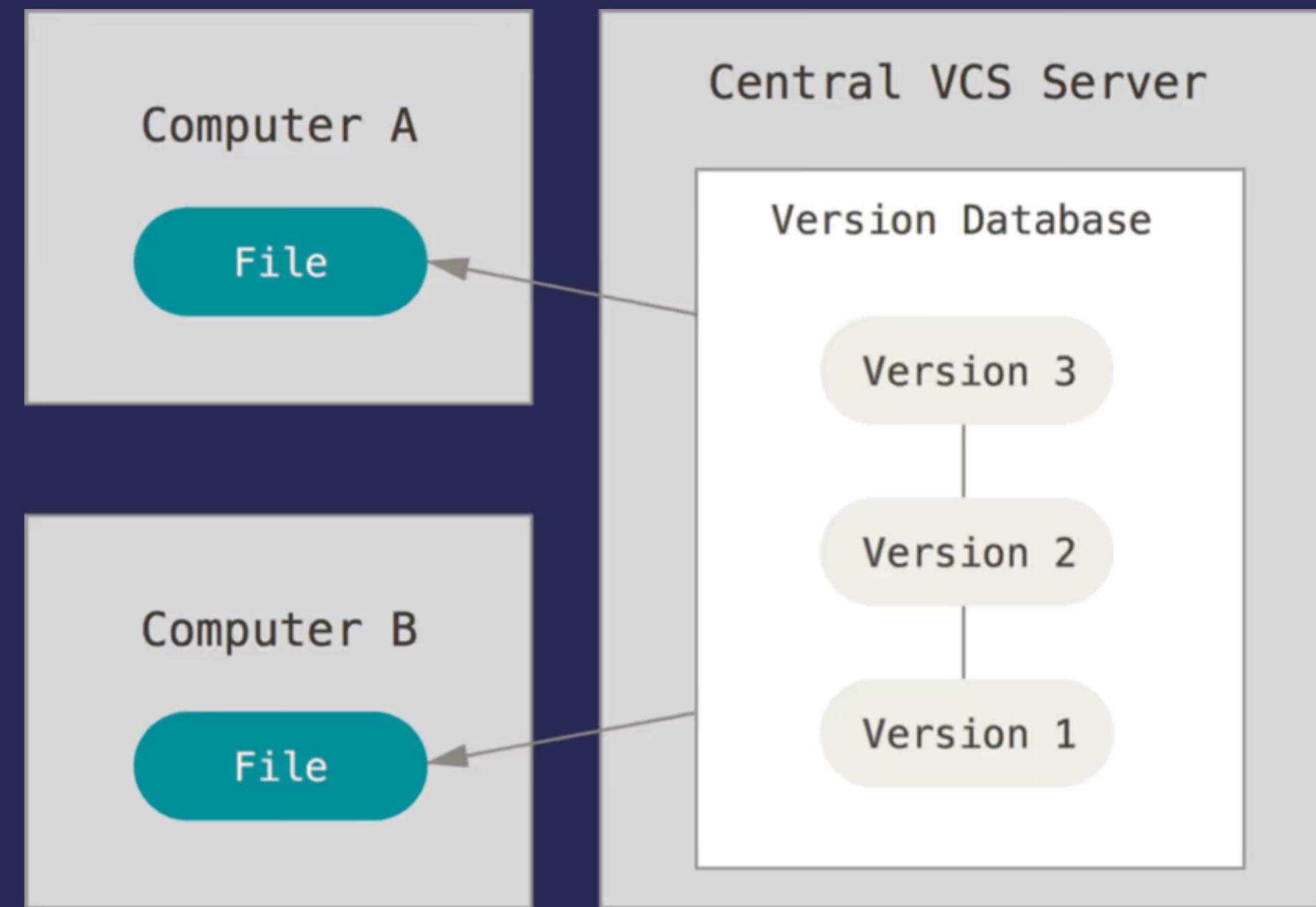
repositório principal era hospedado em um único servidor que armazenava todos os ficheiros versionados.



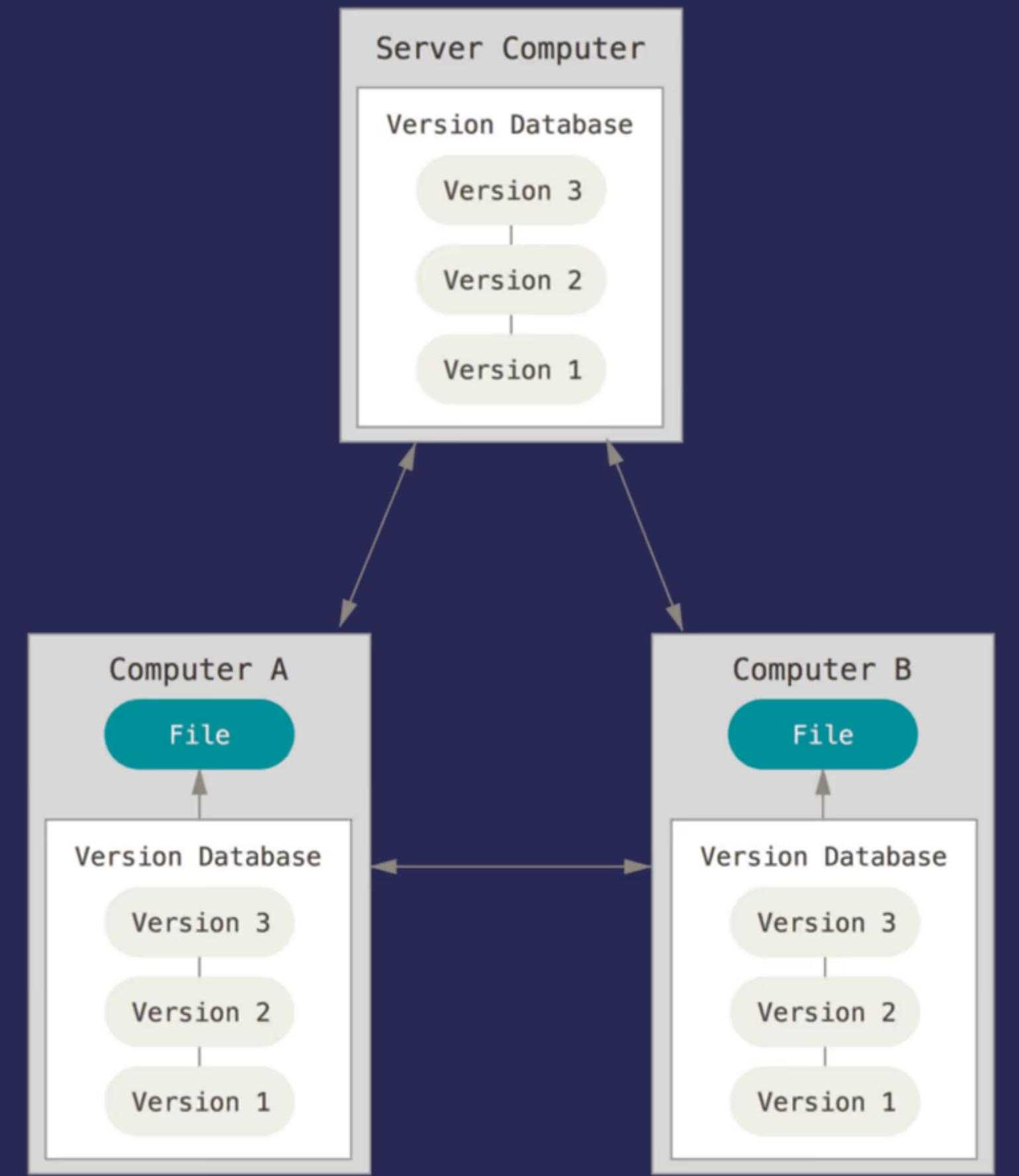
# SCV DISTRIBUÍDO

"cada clone [de um repositório de SCV distribuído] é realmente um backup completo de todos os dados" (Chacon e Straub, 2014, p. 12)

# SCV CENTRALIZADO



# SCU DISTRIBUIDO





## LIVRE

Livre, gratuito e com  
uma vasta  
comunidade de  
usuários.  
Documentação  
extensa

## DISTRIBUÍDO

Compreende seus  
dados como "uma  
série de snapshots  
de um sistema de  
arquivos em  
miniatura" (Chacon e  
Straub, 2014, p.15)

## METODOLOGIA

Metadados e linha do  
tempo.  
Leve.  
Integridade dos  
dados.  
Segurança.  
Registros e acesso  
às informações.  
Controle das etapas.

# ATENÇÃO PARA AS LIMITAÇÕES DO GIT

- 
1. Curva de aprendizado
  2. Dificuldades com arquivos compactados
  3. Ficheiros muito grandes



# FLUXO DE TRABALHO COM GIT

1. Modifica um ficheiro no seu diretório de trabalho (working tree);
2. Selecciona as mudanças que pretende submeter para o histórico do Git (ou o repositório local);
3. Envia as mudanças para a área de preparação (staging area);
4. Realiza a submissão (commit), com uma mensagem associada às mudanças.
5. Os ficheiros da área de preparação são armazenados permanentemente no seu repositório local do Git, juntamente com o conjunto de metadados associado ao commit (um snapshot).

## DIRETÓRIO DE TRABALHO (WORKING DIRECTORY)

## ÁREA DE PREPARAÇÃO (STAGING AREA)

## REPOSITÓRIO LOCAL (DIRETÓRIO .GIT)

Prepara as alterações

Submete as alterações

# INSTALANDO O GIT

**WINDOWS:**

Download Git

**LINUX:**

```
sudo apt install git
```

**MACOS:**

```
brew install git
```

# ou

```
sudo port install git
```

# CONFIGURAÇÃO GLOBAL

```
git config --global user.name "Edward Palmer Thompson"
```

```
git config --global user.email ephompson@hist.com
```

```
git config --global core.editor "vim"
```

```
git config --global init.defaultbranch main
```



# GIT INIT

Inicia um diretório  
como um repositório  
local Git.



# GIT STATUS

Informa a situação  
atual do  
repositório

# GIT ADD

Insere ficheiros na  
área de preparação



# GIT COMMIT

Submete as  
alterações  
preparadas no  
repositório local.  
Sempre deve vir  
acompanhada de  
uma mensagem.

## DIRETÓRIO DE TRABALHO (WORKING DIRECTORY)

## ÁREA DE PREPARAÇÃO (STAGING AREA)

## REPOSITÓRIO LOCAL (DIRETÓRIO .GIT)

`git add <file>`

Prepara as alterações

`git commit`

Submete as alterações

# ESTÁGIOS DE UM FICHEIRO



- Não monitorado (untracked)
- Monitorado (tracked)
- Modificado (modified)
- Preparado para submissão  
(Staged)
- Submetido (Committed)

## DIRETÓRIO DE TRABALHO (WORKING DIRECTORY)



não monitorado  
↓  
novo ficheiro

monitorado  
↓  
modificado

## ÁREA DE PREPARAÇÃO (STAGING AREA)

Prepara as alterações



preparado

## REPOSITÓRIO LOCAL (DIRETÓRIO .GIT)

Submete as alterações



monitorados  
↓  
committed



# COMO ESCREVER UMA MENSAGEM DE COMMIT EFICIENTE?

A melhor forma de escrever a mensagem de commit é utilizar `git commit` sem a opção `-m`, pois nos permite escrever mensagens mais longas do que 50 caracteres e incluir quebras de linha e um título para nossa mensagem.

O `git commit` abre o editor de texto padrão do seu sistema operacional - ou o editor que você configurou no Git - para que você possa escrever a mensagem de commit.

# MENSAGEM DE COMMIT



Atualização dos dados da lição

- Inclusão do nome do Programming Historian no README.md
- Atualização do texto em resumos.txt

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# No ramo main
# Mudanças a serem submetidas:
#       modified: README.md
#       modified: resumo.txt
#
```





# RECUPERANDO INFORMAÇÕES COM GIT LOG

# GIT LOG



```
git log --oneline
```

```
git log --pretty=format:"%h,%an,%ad,%s"
```

```
git log --pretty=format:"%h,%an,%ad,'%s','%b'" > log.csv
```

# EXEMPLO DE PLANILHA DE LOG

A5037c2	Eric Brasil	Tue Apr 11 18:03:44 2023 -0300	'Criação do arquivo tabular com dados do log'
0f4bde9	Eric Brasil	Tue Apr 11 18:00:10 2023 -0300	'Inclusão da apresentação'
49fc794	Eric Brasil	Tue Apr 11 17:45:14 2023 -0300	'Criação de versão em docx da estrutura'
3b10dc1	Eric Brasil	Tue Apr 11 10:00:44 2023 -0300	'Criação do arquivo com estrutura do workshop'
a718565	Eric Brasil	Tue Apr 11 09:36:56 2023 -0300	'Initial commit'

# REVERTER COMMITs



```
git reset --hard <hash do commit>
```

```
# apaga os commits até o commit indicado  
# no comando
```

# CONCLUSÃO

O uso consciente e sistemático do Git, apesar de sua curva de aprendizado mais acentuada, permite que pesquisadores e equipes possam trabalhar de forma segura e controlada, integrando ao processo de pesquisa/escrita os procedimentos metodológicos de documentação e registro de metadados e decisões tomadas.

Ao mesmo tempo, garante a criação de uma linha do tempo de todo o processo, permitindo a recuperação das informações e restauração de ficheiros.





# REFERENCES

- Bird, Christian, Peter C. Rigby, Earl T. Barr, David J. Hamilton, Daniel M. German, e Prem Devanbu. "The promises and perils of mining git". Em 2009 6th IEEE International Working Conference on Mining Software Repositories, 1-10, 2009. <https://doi.org/10.1109/MSR.2009.5069475>.
- Baker, Baker, "Preservar os seus dados de investigação", traduzido por Márcia T. Cavalcanti, Programming Historian em português 1(2021), <https://doi.org/10.46430/phpt0001>.
- Brasil, Eric. Criação, manutenção e divulgação de projetos de História em meios digitais: git, GitHub e o Programming Historian. Zenodo (2022). <https://doi.org/10.5281/zenodo.6566754>
- Bryan, Jennifer. "Excuse Me, Do You Have a Moment to Talk About Version Control?" *The American Statistician* 72, nº1(2 de janeiro de 2018): 20-27. <https://doi.org/10.1080/00031305.2017.1399928>.
- Chacon, Scott, e Ben Straub. *Pro Git*. 2º edição. Apress, 2014.
- Loeliger, Jon, e Matthew McCullough. *Version Control with Git: Powerful tools and techniques for collaborative software development*. 2º edição. Sebastopol, CA: O'Reilly Media, 2012.
- Ram, Karthik. "Git can facilitate greater reproducibility and increased transparency in science". *Source Code for Biology and Medicine* 8, nº1(28 de fevereiro de 2013): 7. <https://doi.org/10.1186/1751-0473-8-7>.

# CONTATOS

Emails:

[ericbrasiln@proton.me](mailto:ericbrasiln@proton.me)

[profericbrasil@unilab.edu.br](mailto:profericbrasil@unilab.edu.br)

GitHub:

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

Twitter:

[www.twitter.com/ericbrasiln](https://www.twitter.com/ericbrasiln)

Site:

<https://ericbrasiln.github.io/>



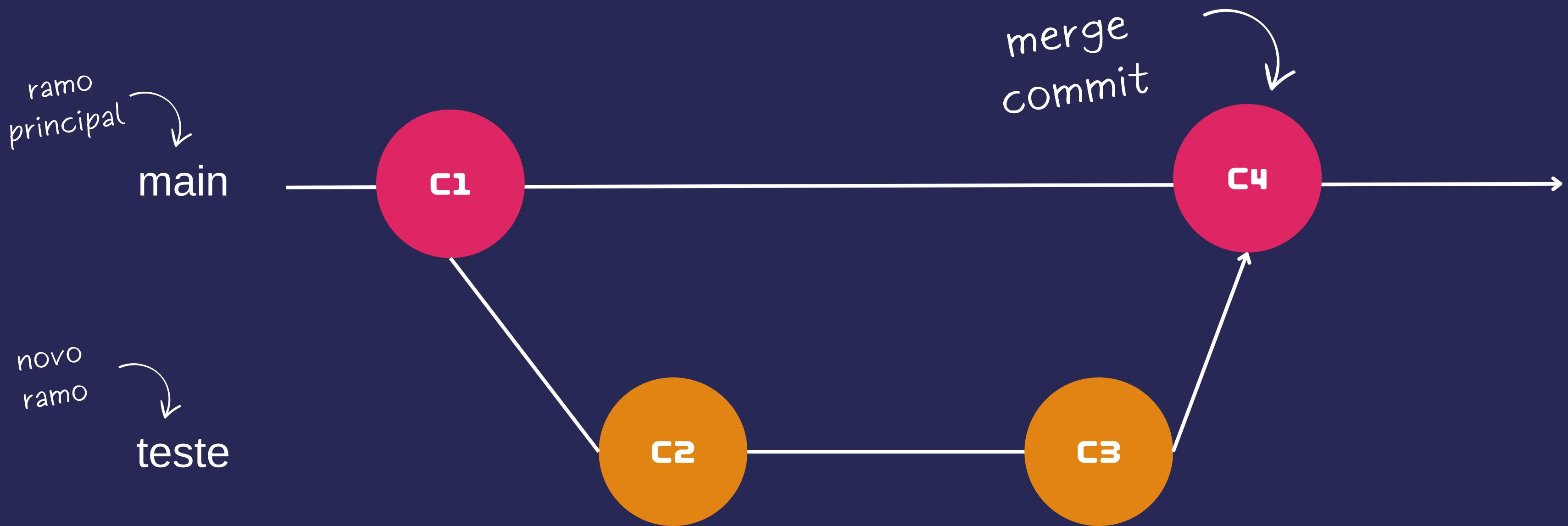
# OBRIGADO!





**EXTRA!**

# RAMIFICAÇÕES (BRANCHES)



# GIT BRANCH



```
git branch <nome do novo branch> #Cria novo branch
```

```
git checkout <nome do branch> #Muda para um branch
```

```
git merge <nome do branch> # Faz a mescla do branch nomeado com o branch atual
```

