

Eric Brasil

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!



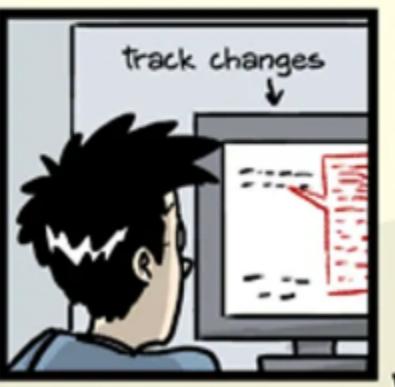
↑ FINAL_rev.2.doc



↑ FINAL_rev.6.COMMENTS.doc



↑ FINAL_rev.8.comments5.CORRECTIONS.doc



JORGE CHAM © 2012

FINAL_rev.18.comments7.corrections9.MORE.30.doc

FINAL_rev.22.comments49.corrections.10.#@\$%WHYDIDICOMETOGRAD 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

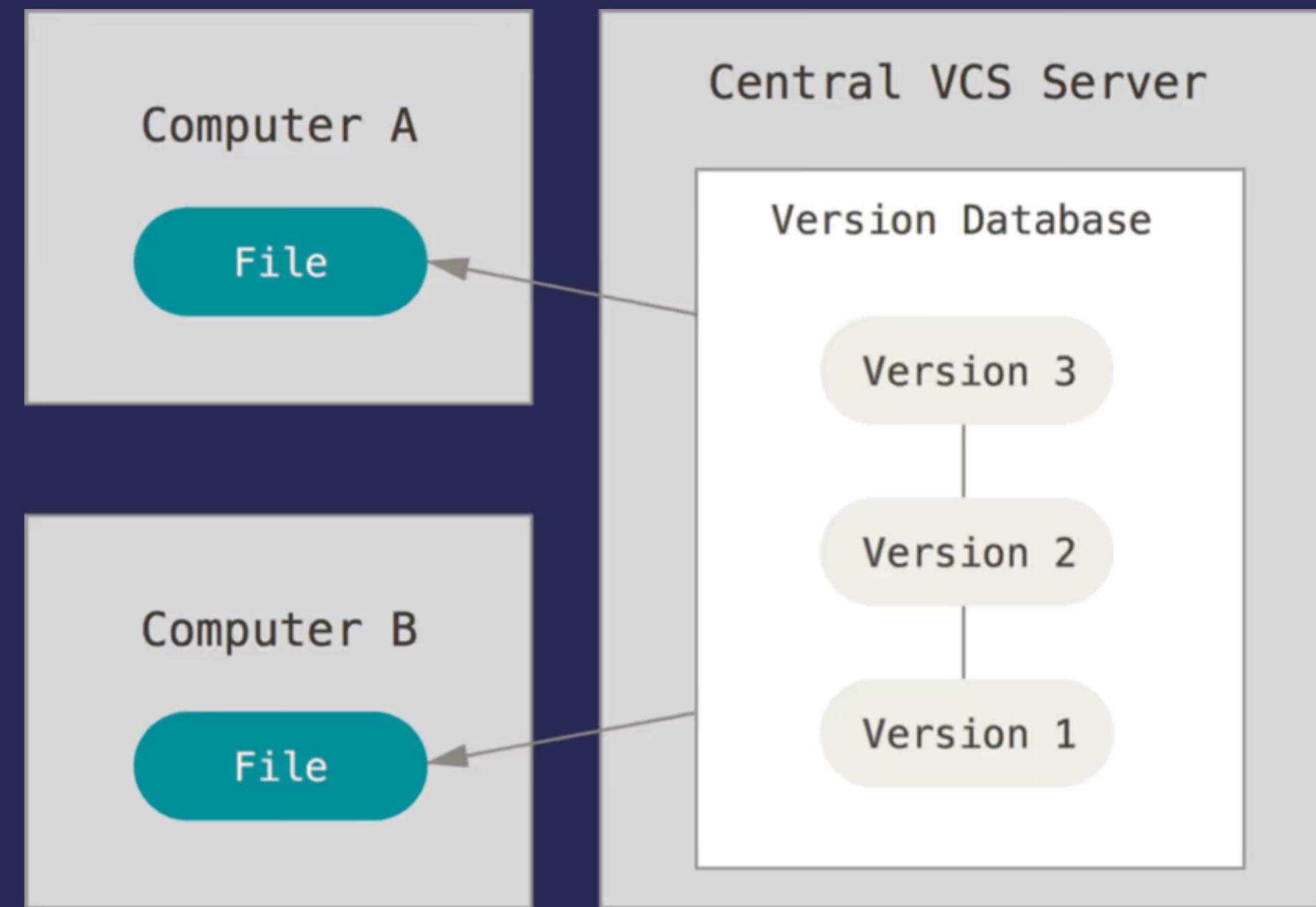
repositório principal era hospedado em um único servidor que armazenava todos os ficheiros versionados.



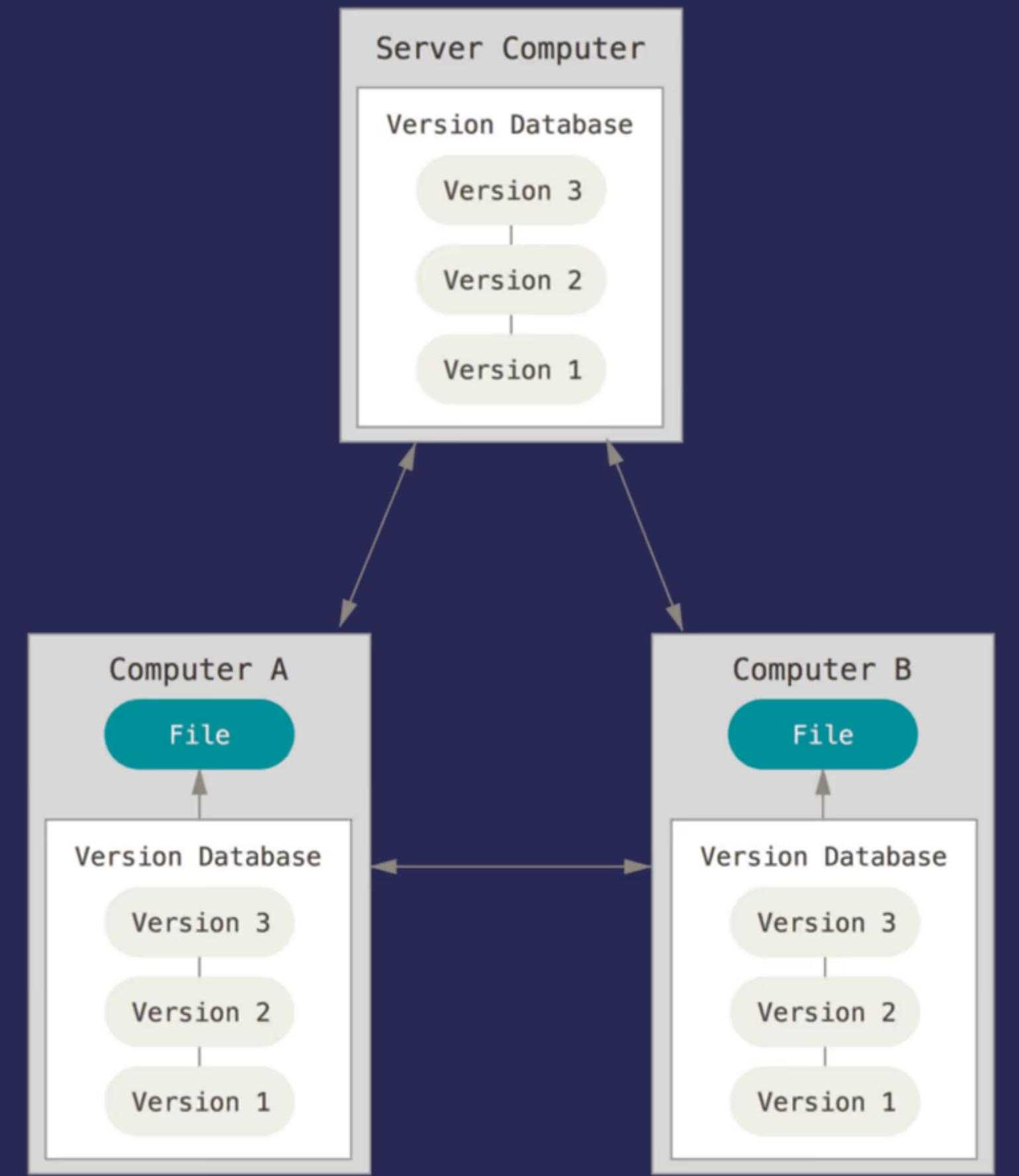
SCV DISTRIBUÍDO

"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 SUAS LIMITAÇÕES



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



FLUXO DE TRABALHO com GIT

1. Você modifica algum ficheiro no seu diretório de trabalho (working tree);
2. Você seleciona 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. Você realiza a submissão (commit), incluindo uma mensagem explicativa associada às mudanças realizadas.
5. O Git então pega os ficheiros exatamente como estão na área de preparação e armazena esse snapshot permanentemente no seu repositório local do Git, juntamente com o conjunto de metadados associado ao commit.

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.

ESTÁGIOS DE UM ARQUIVO



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

REVERTER COMMITs

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

- Ketut Susilo (2017)
- Rosa Maria Aguado (2019)
- Yanis Petros (2022)

RESOURCE PAGE

