









# **Tutorial** **Controle de Versão** **de Software com Git**



**Henrique Linhares**

Mais informações no site: <http://ev-ppgc.ic.uff.br/2020-2/tutoriais.html>

# Introdução

Nome	Tipo	Tamanho
 trabalho	Arquivo do WinRAR	2.729 KB
 trabalho_final	Arquivo do WinRAR	2.729 KB
 trabalho_final_final	Arquivo do WinRAR	2.729 KB
 trabalho_final_final_para_entregar	Arquivo do WinRAR	2.729 KB
 trabalho_final_final_para_entregar2	Arquivo do WinRAR	2.729 KB
 trabalho_final_final2	Arquivo do WinRAR	2.729 KB
 trabalho_final2	Arquivo do WinRAR	2.729 KB
 trabalho_final3	Arquivo do WinRAR	2.729 KB

- Softwares evoluem
- Como controlar as mudanças em um software?

# Controle de Versão?

- Sistema que registra alterações em arquivos



# Por que usar um VCS?

- Recuperar versões antigas do seu programa
- Comparar versões
- Colaborar
- Criar ramificações
- Compartilhar o código fonte

# Por que usar um VCS?

- Recuperar versões antigas do seu programa
- Comparar versões
- Colaborar
- Criar ramificações

Ciência aberta!!

- Compartilhar o código fonte

# Por exemplo...



## Grupo de Evolução e Manutenção de Software (GEMS)

Universidade Federal Fluminense (UFF) <http://gems.ic.uff.br/>

Repositories 36

Packages

People 1

Projects

Find a repository...

Type: All ▾

Language: All ▾

### merge-nature

Companion website for the paper "On the Nature of Software Merge Conflicts"

Java 0 0 0 0 Updated 17 days ago

#### Top languages

Java Python Jupyter Notebook  
JavaScript Ruby

### noworkflow

Supporting infrastructure to run scientific experiments without a scientific workflow management system.

Jupyter Notebook MIT 19 87 48 1 Updated 19 days ago

#### People

1 >



<https://github.com/gems-uff>

# Git vs GitHub



<https://www.amarinfotech.com/gitlab-vs-github-vs-bitbucket.html>

# Git vs GitHub

## Git

- Instalado localmente
- Lançado em 2005
- Linux Foundation
- Controle de versão e compartilhamento de código
- Command-line
- Desktop interface Git Gui
- Compete com Mercurial, Subversion, IBM ClearCase
- Licença Open source

## Github

- Nuvem
- Lançado em 2008
- Comprado pela Microsoft em 2018
- Hospedagem de código de fonte centralizada
- Administrado pela web
- Desktop interface GitHub
- Compete com Atlassian Bitbucket e GitLab
- Tem versão gratuita e paga



# Git vs GitHub

## Git

- Um sistema de controle de versão de arquivos

## Github

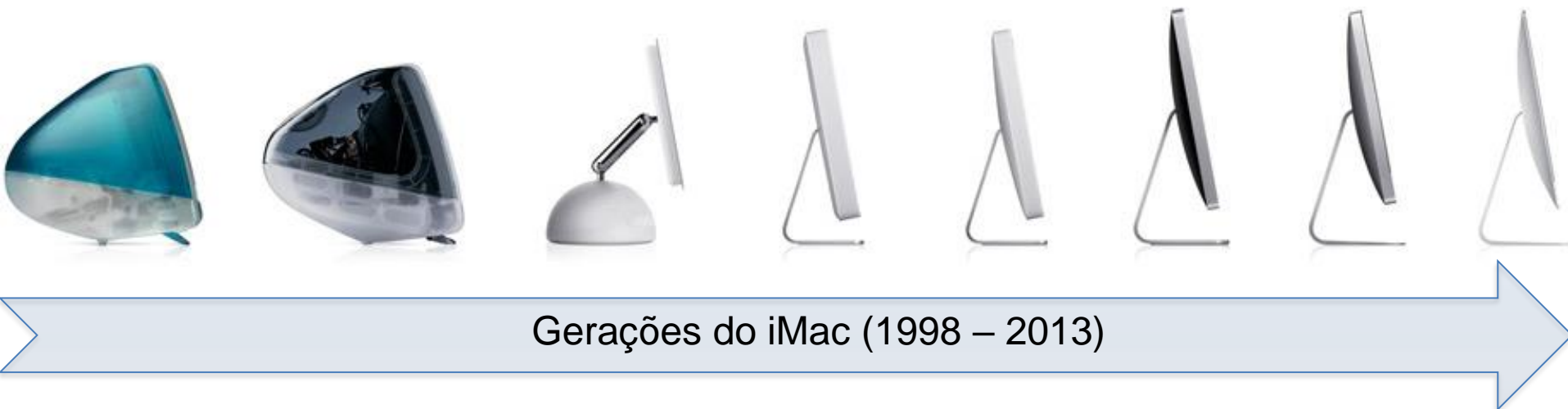
- Um serviço web que oferece diversas funcionalidades extras aplicadas ao git

# Mas afinal, o que são versões?



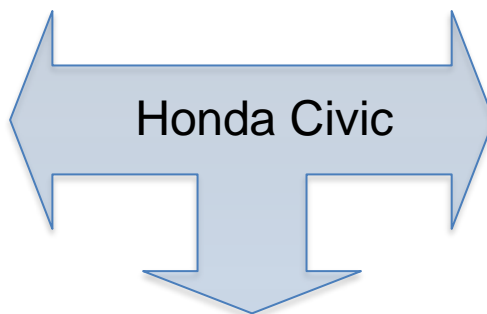
(Conradi and Westfechtel 1998)

# Revisões

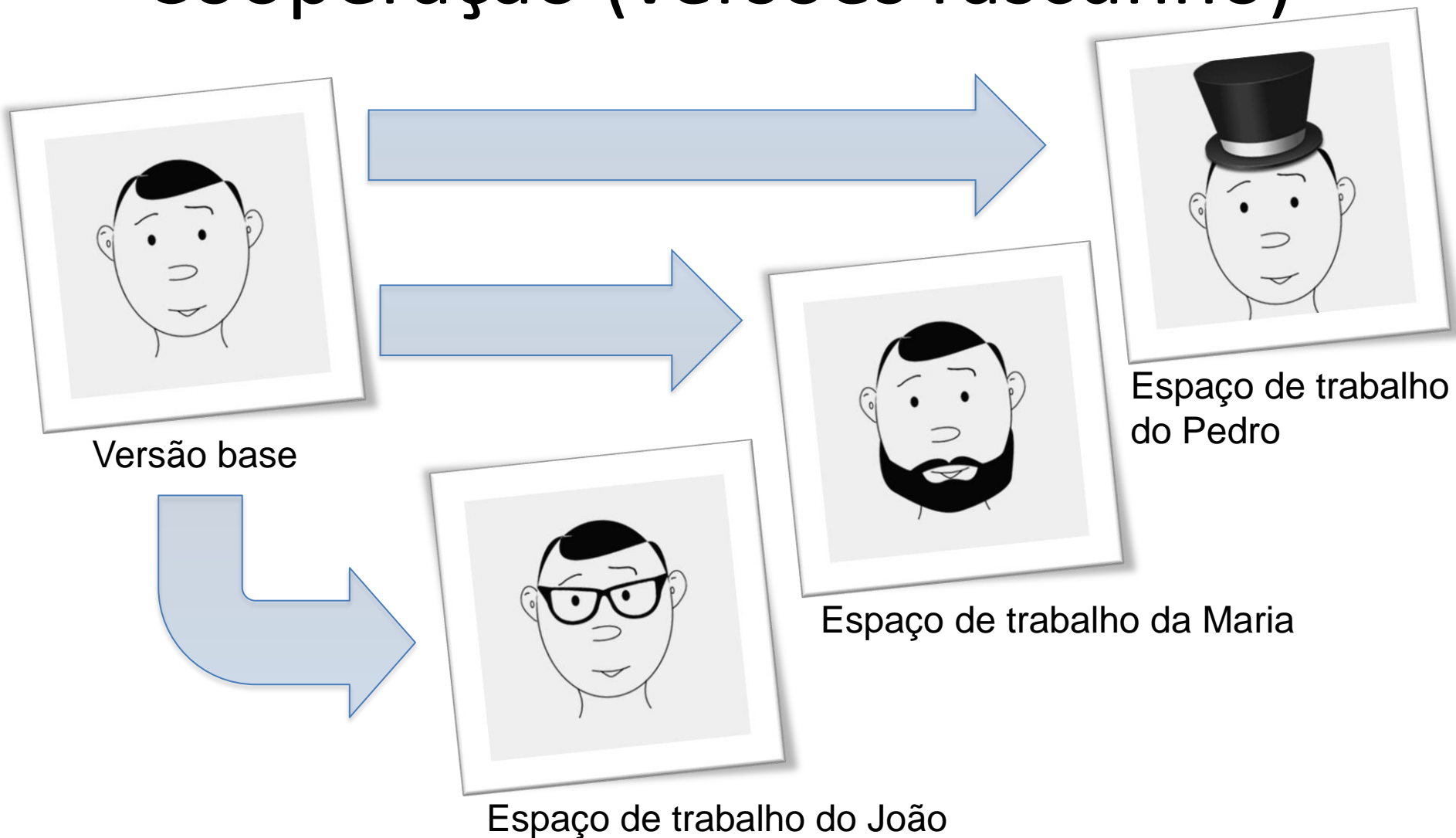


Gerações do iMac (1998 – 2013)

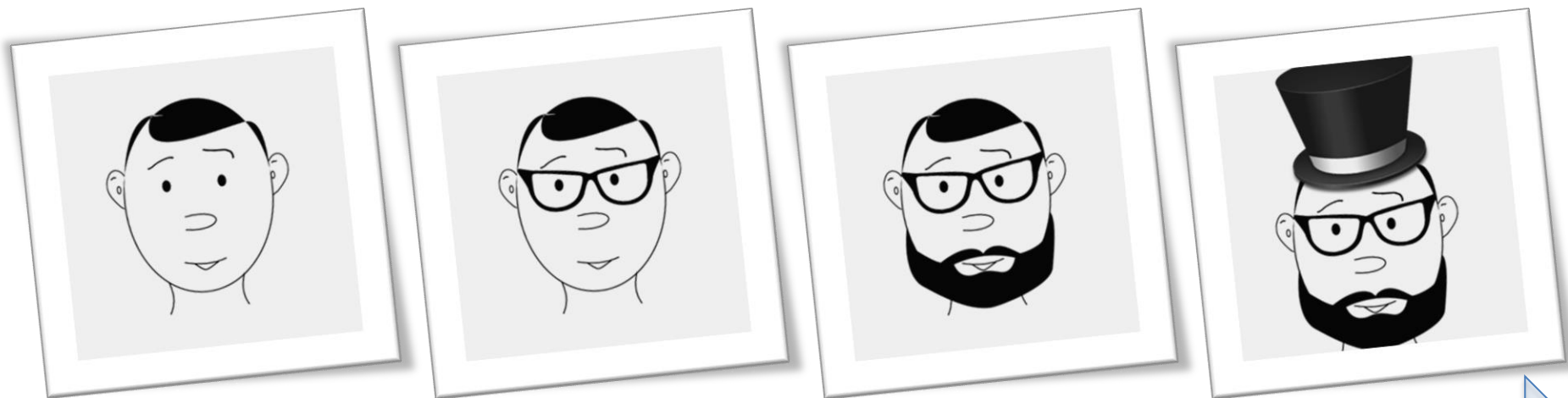
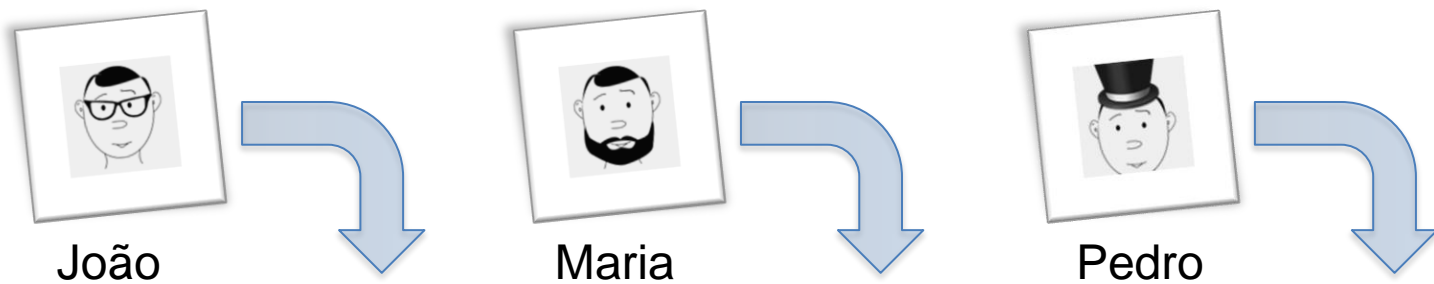
# Variantes



# Cooperação (versões rascunho)

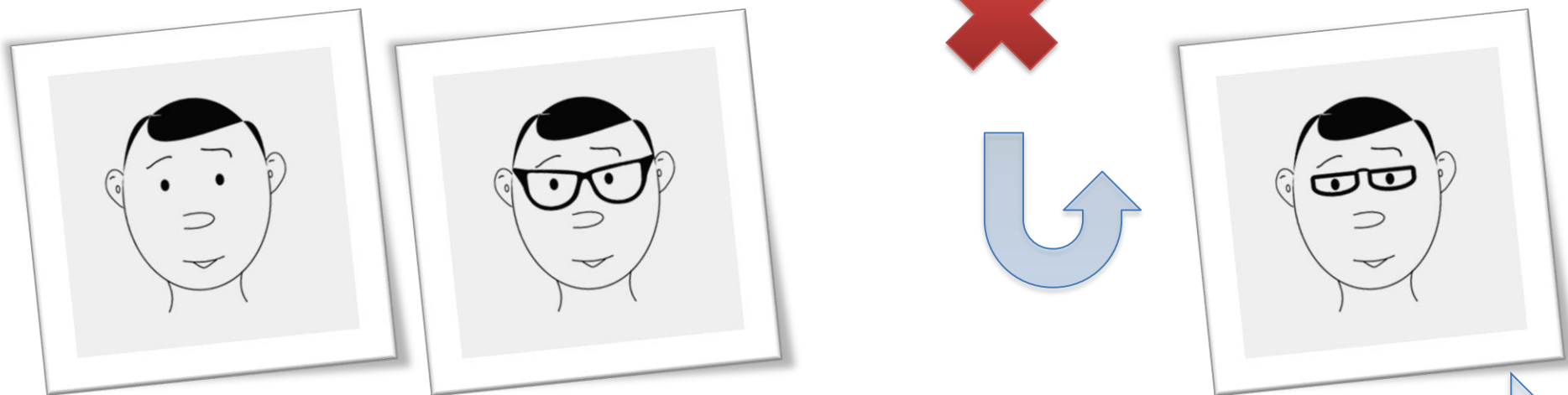
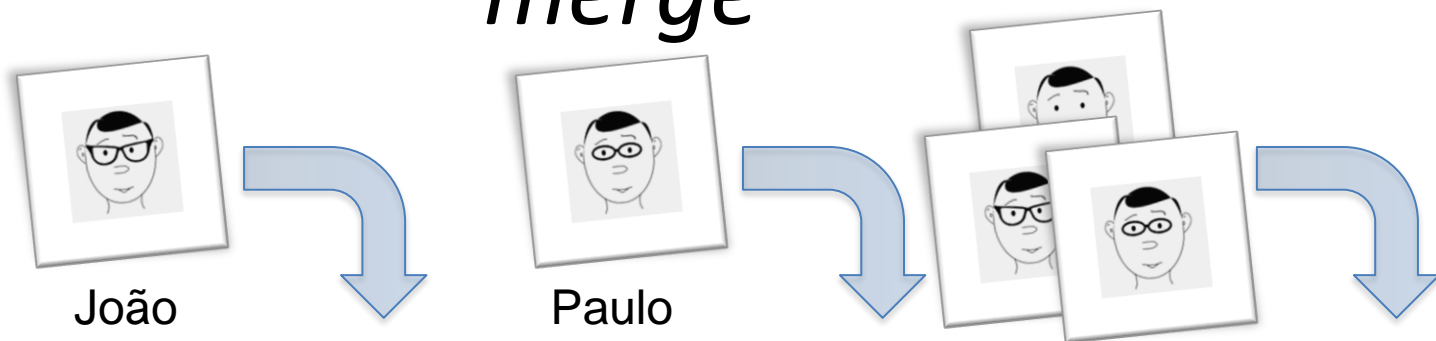


# Versões de rascunho podem ser combinadas (operação de *merge*)



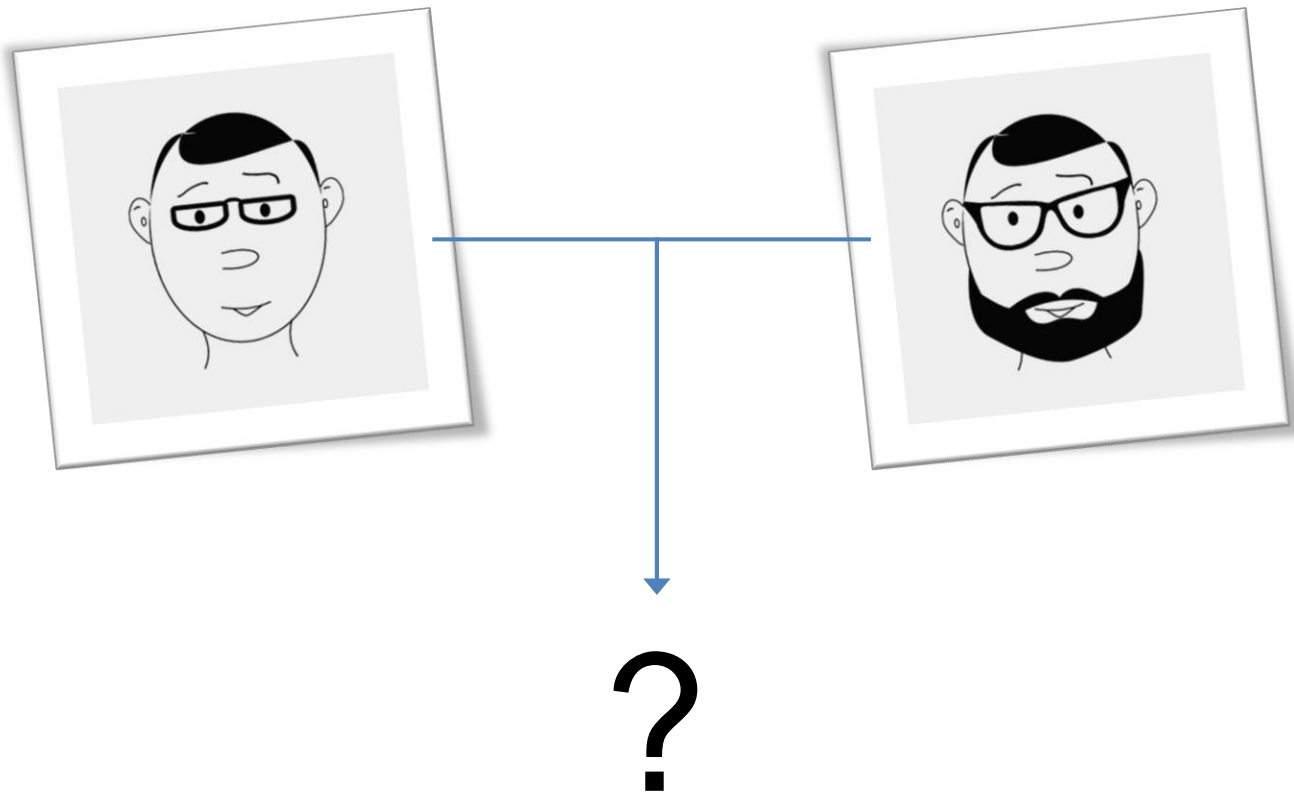
Revisões

# Conflitos podem ocorrer durante o *merge*



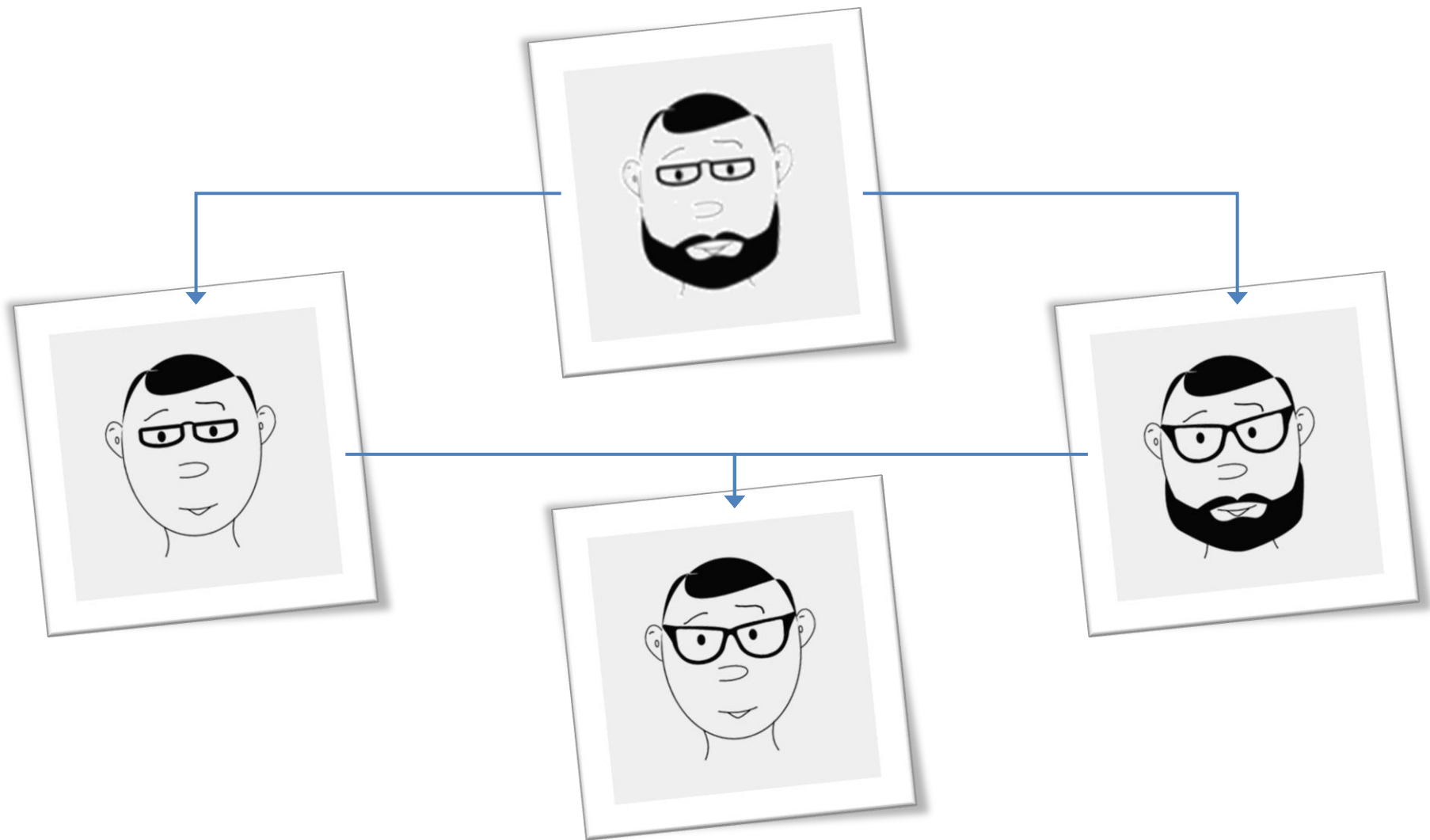
Revisões

# 2-way merge

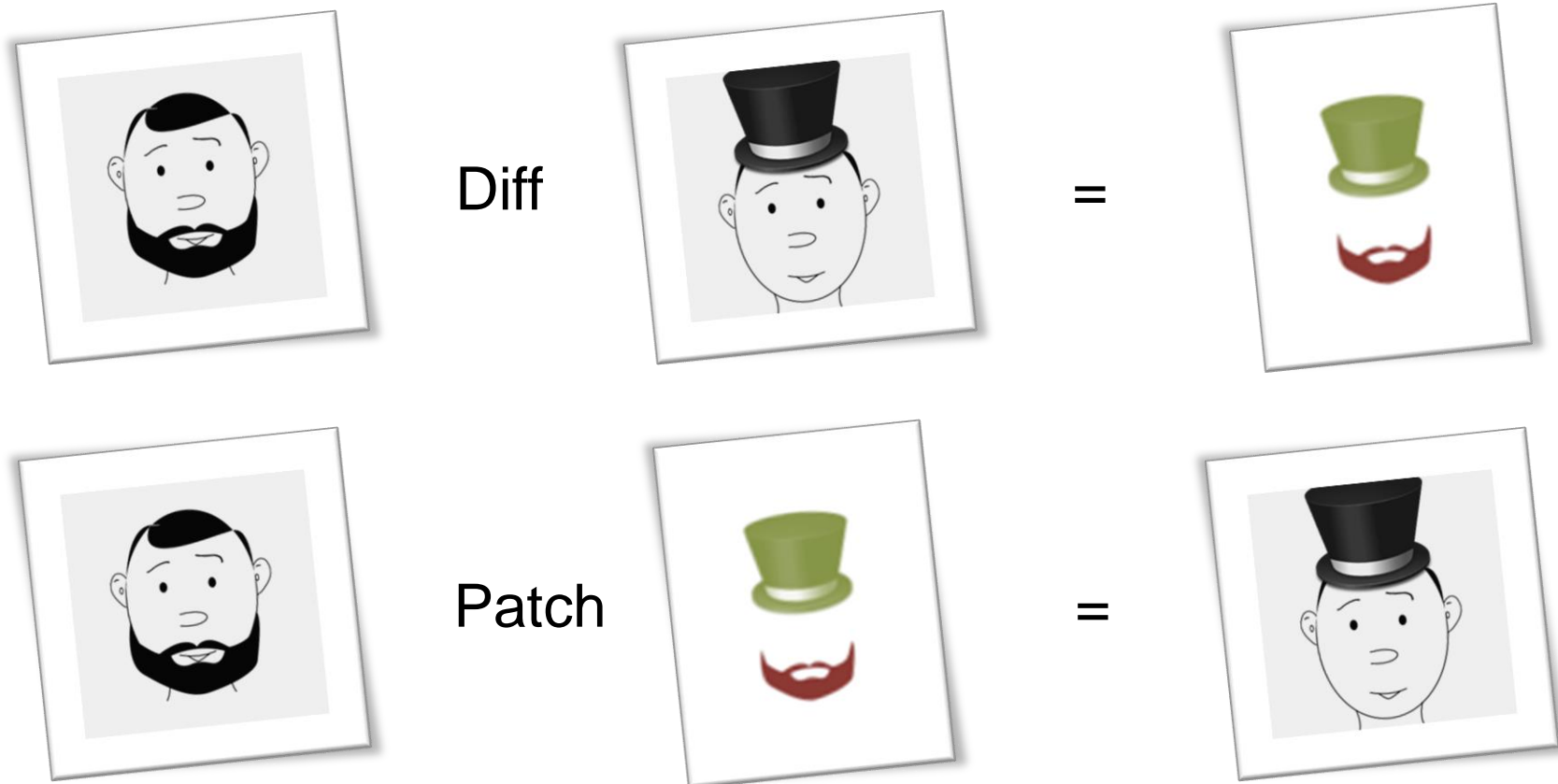




# 3-way merge



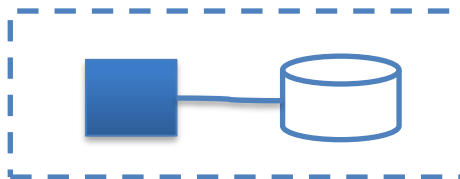
# Outras duas operações importantes...



... para guardar, transferir e compreender versões.

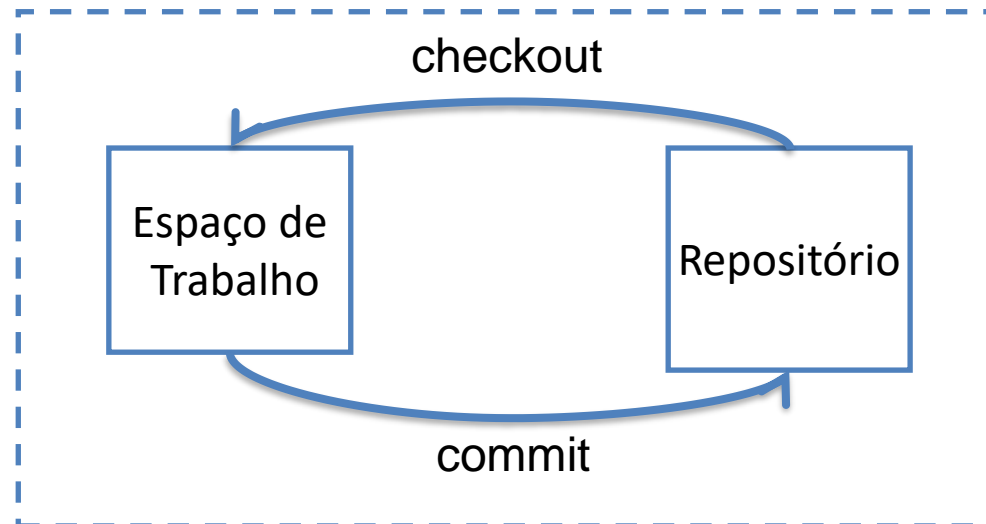
# Histórico dos sistemas de controle de versões (VCS)

- Anos 70/80 – Sistemas locais
  - SCCS (1972)
  - RCS (1982)
- Repositório e espaço de trabalho estão na mesma máquina



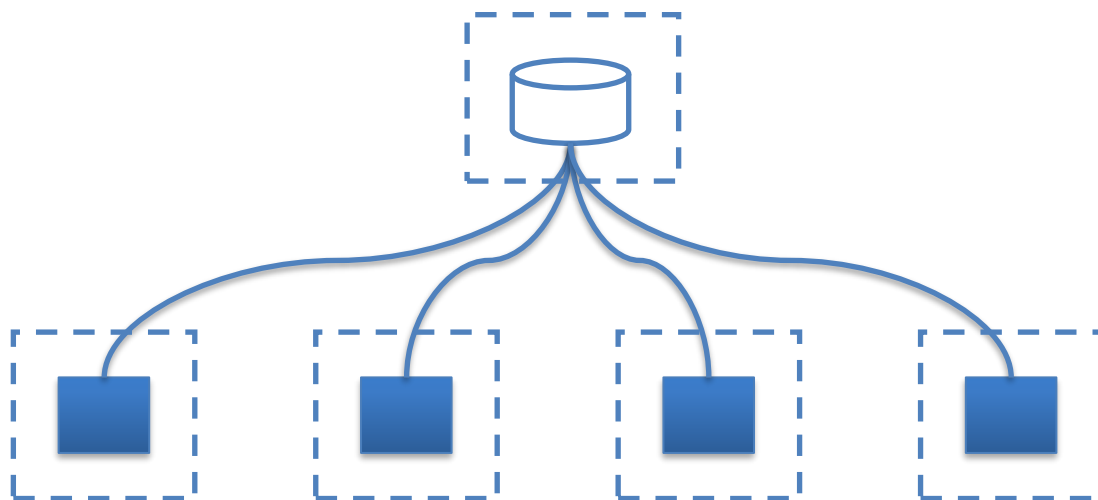
# Histórico dos sistemas de controle de versões (VCS)

- Anos 70/80 – Sistemas locais
  - SCCS (1972)
  - RCS (1982)



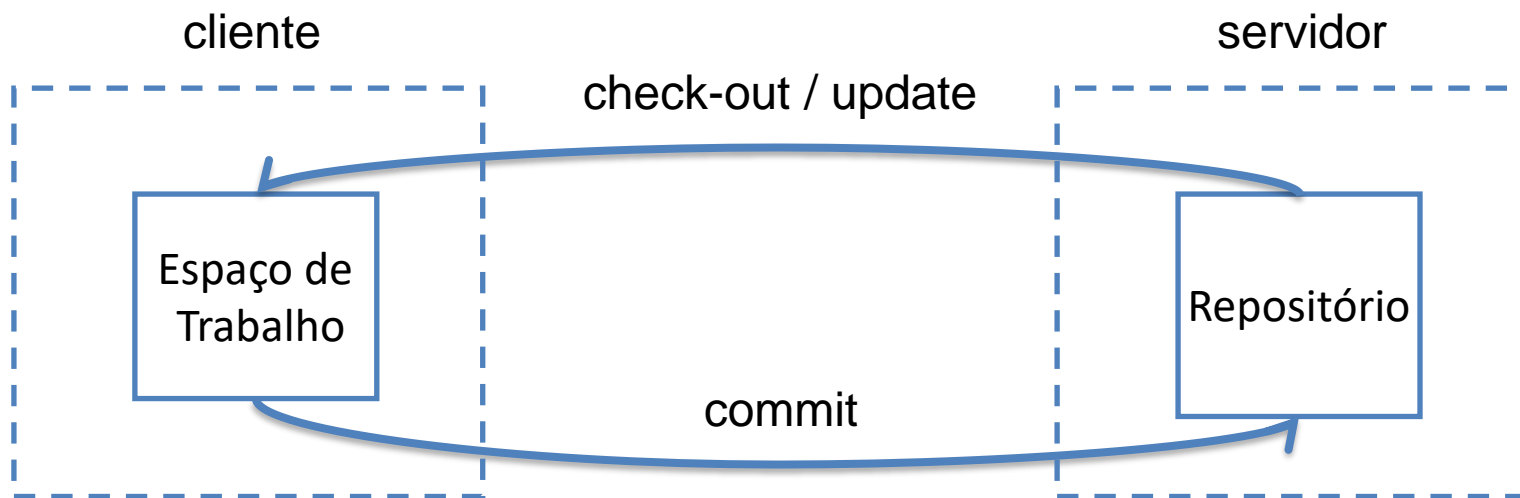
# Histórico dos sistemas de controle de versões (VCS)

- Anos 80/90 – Sistemas cliente-servidor
  - CVS (1986)
  - Subversion (2000)



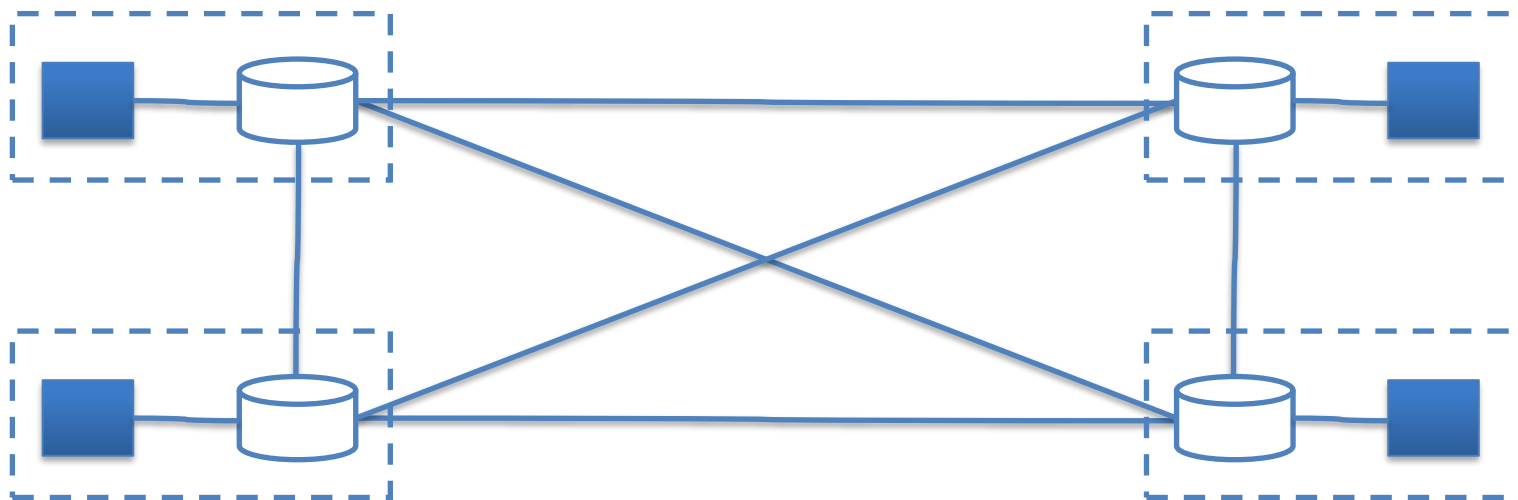
# Histórico dos sistemas de controle de versões (VCS)

- Anos 80/90 – Sistemas cliente-servidor
  - CVS (1986)
  - Subversion (2000)



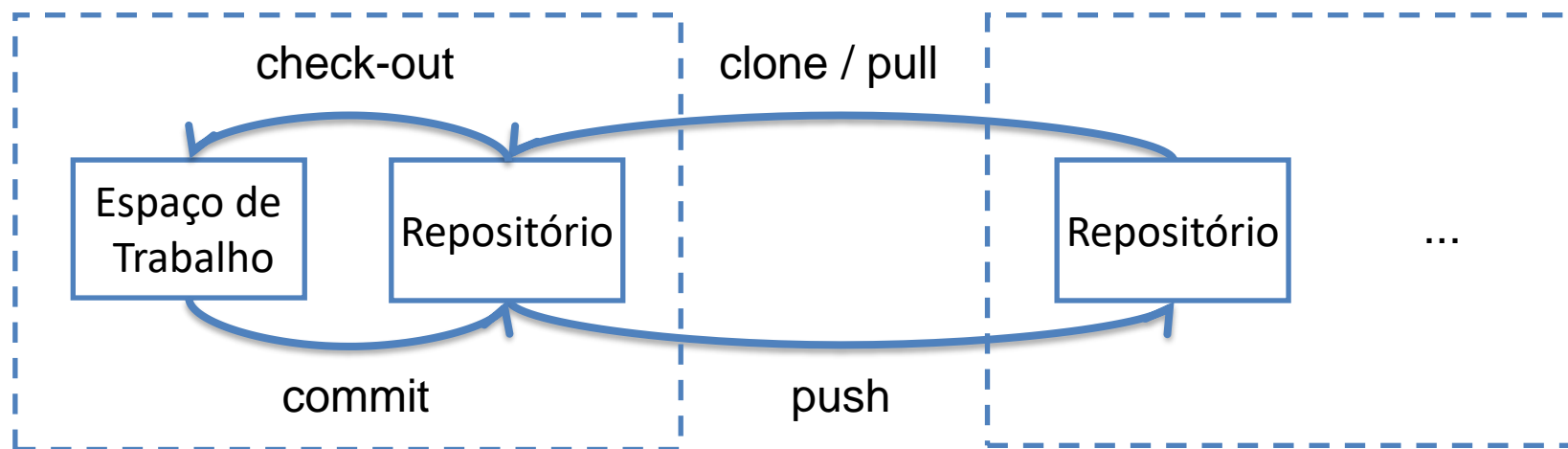
# Histórico dos sistemas de controle de versões (VCS)

- Anos 2000 – Sistemas peer-to-peer
  - Git (2005)
  - Mercurial (2005)



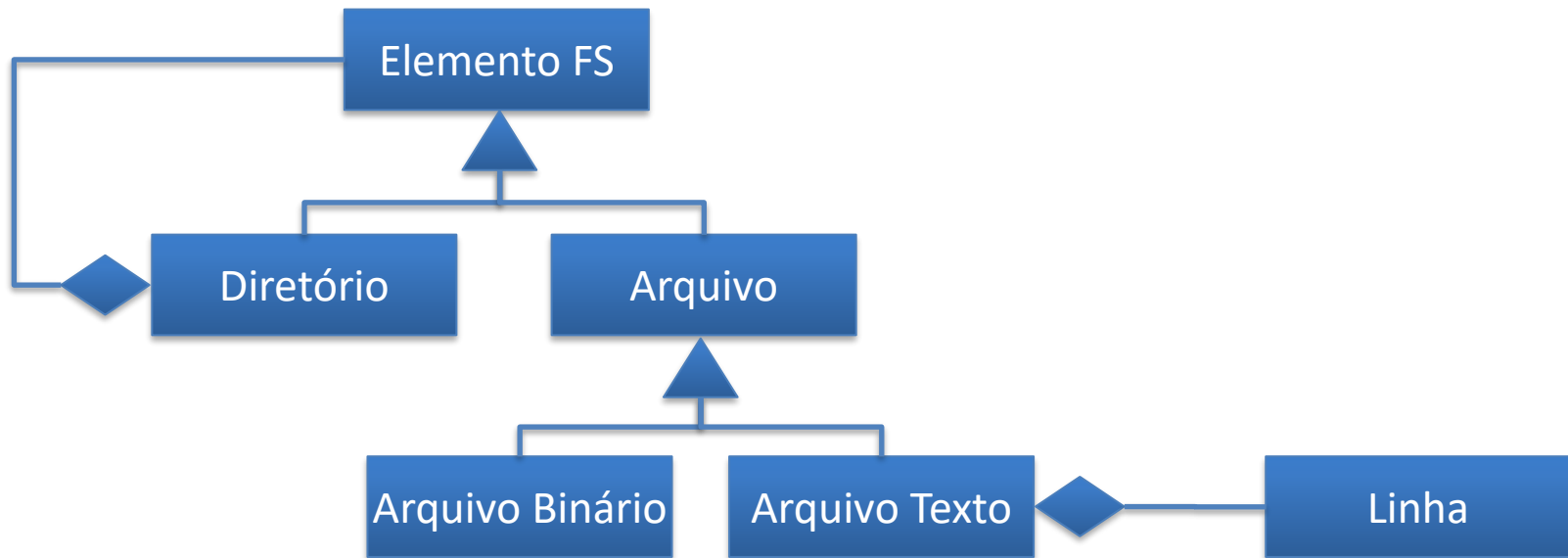
# Histórico dos sistemas de controle de versões (VCS)

- Anos 2000 – Sistemas peer-to-peer
  - Git (2005)
  - Mercurial (2005)

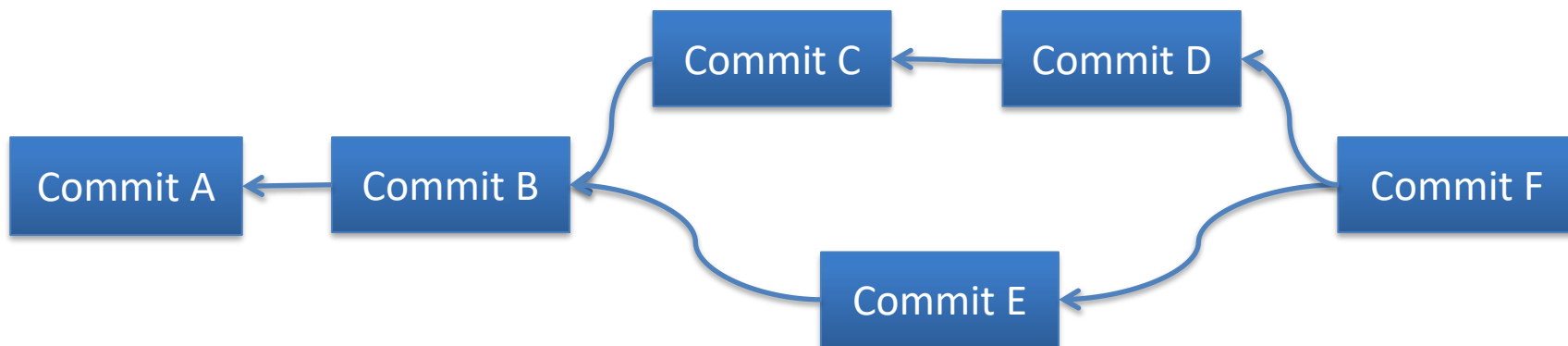
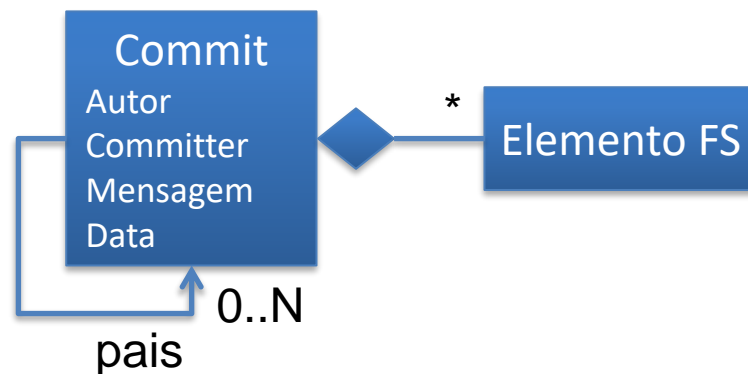




# O que é versionado?



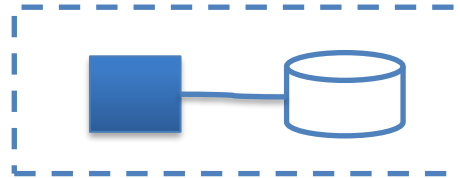
# Como é versionado?



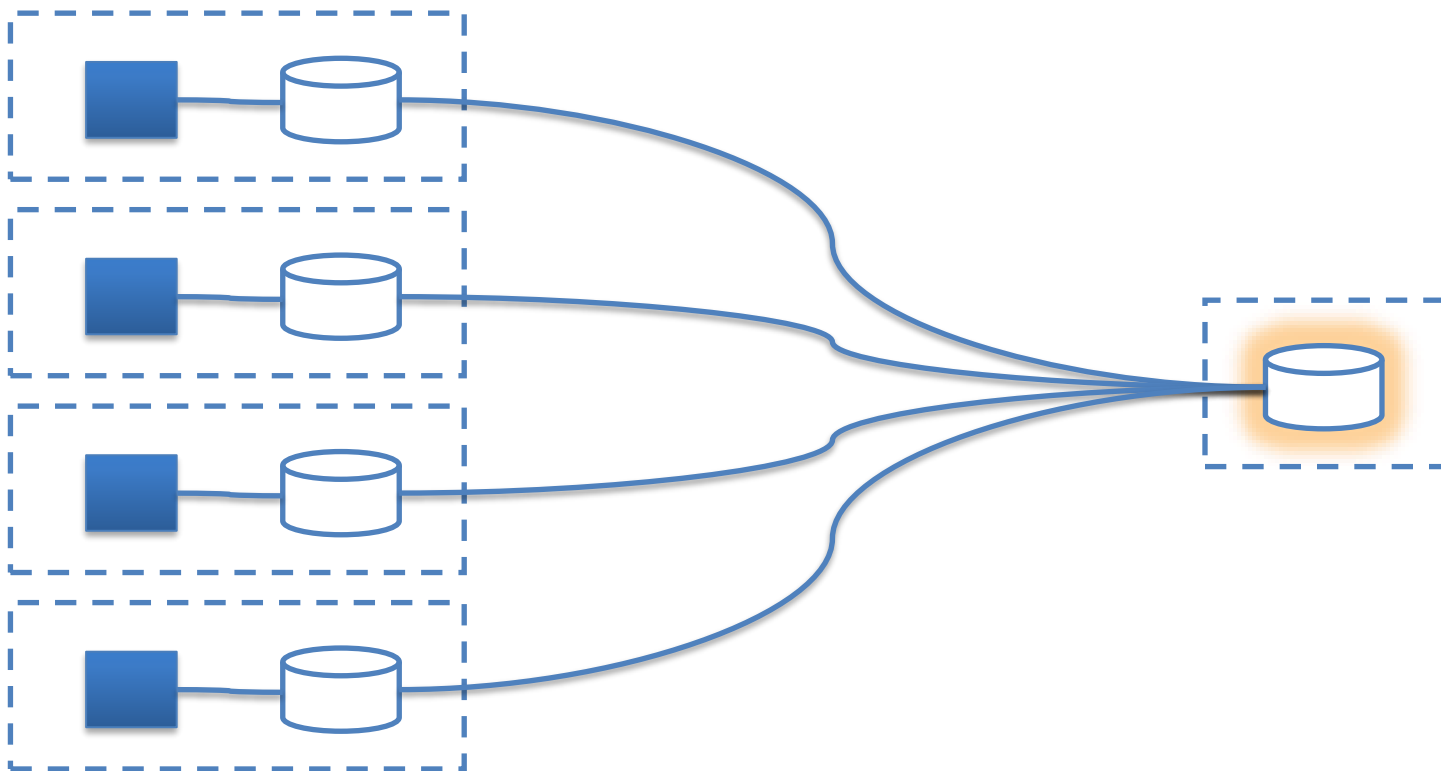
# Formas de adoção

- Apesar de ser peer-to-peer, normalmente é definido um “workflow” para adoção de DVCS (*Distributed Version Control Systems*) em função de características do projeto
  - Individual
  - Cliente-servidor
  - Gerente de integração
  - Ditador/tenentes

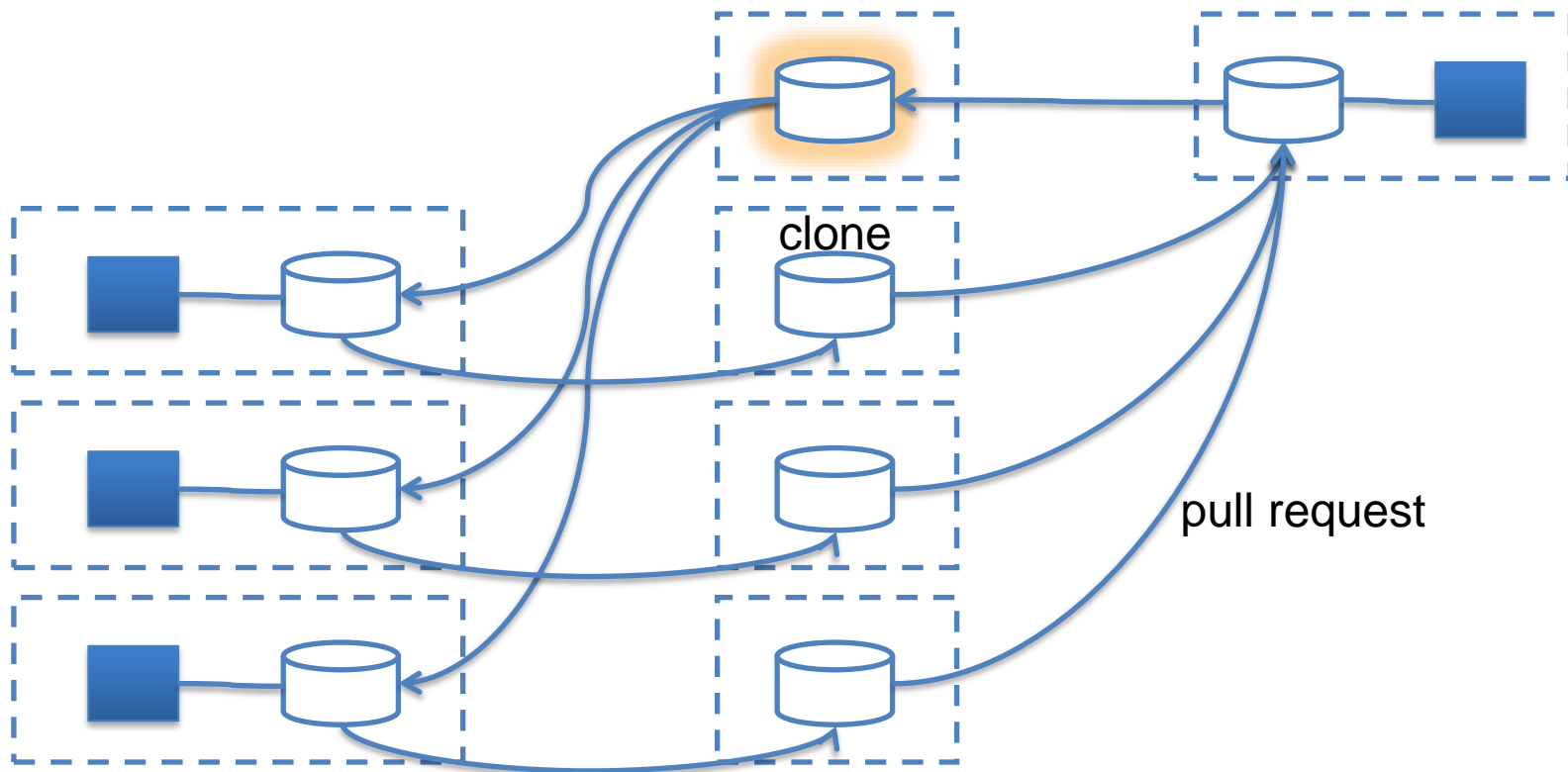
# Individual



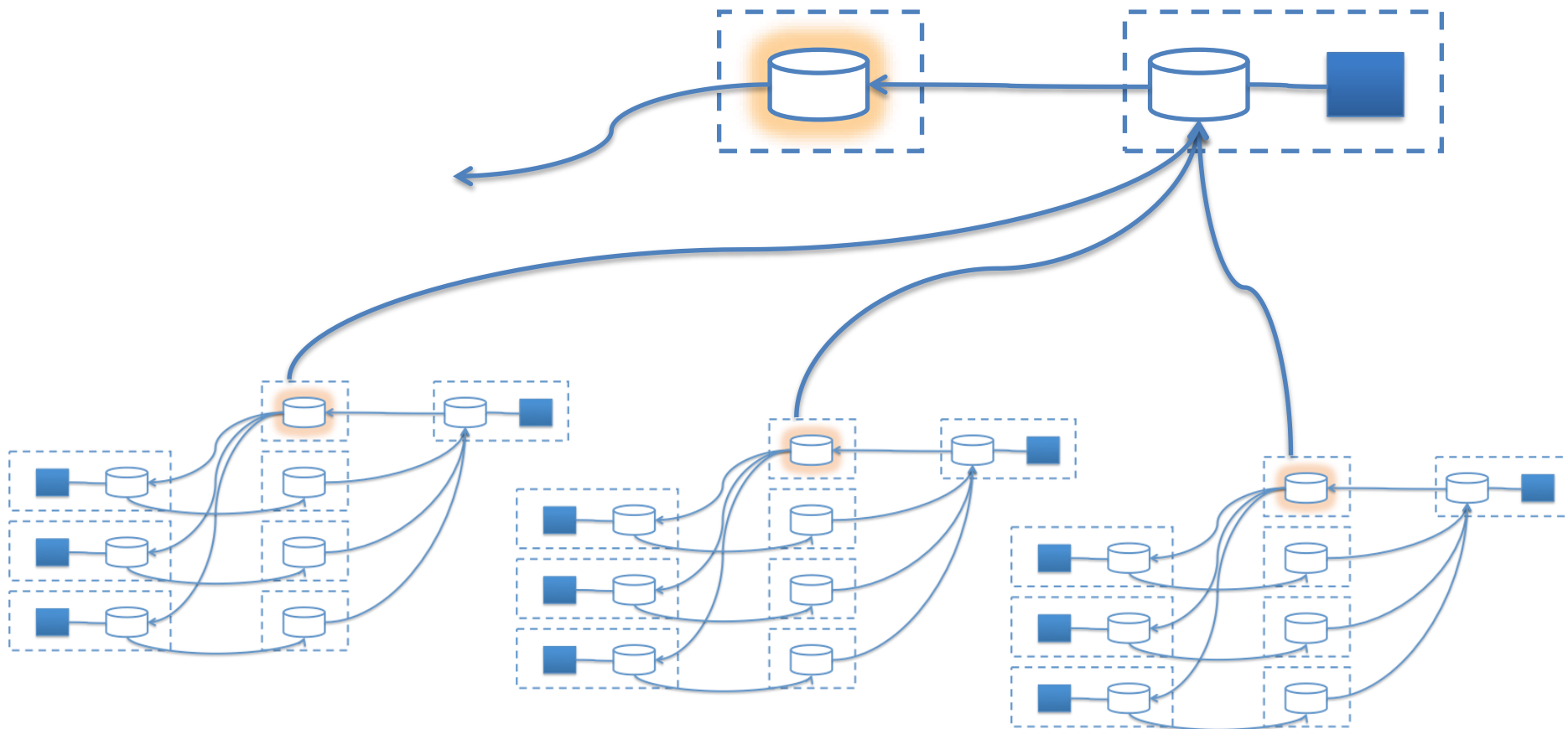
# Cliente-servidor



# Gerente de integração (fork + pull request)



# Ditador/tenentes (pull request em cascata)



# Passo a passo

- Vamos utilizar o Git gradualmente em diferentes situações
  - Conceitos básicos
  - Repositório local
  - Inspeccionando mudanças
  - Demarcando versões especiais
  - Repositório local com ramos
  - Repositório remoto



# Conceitos básicos: *help*!

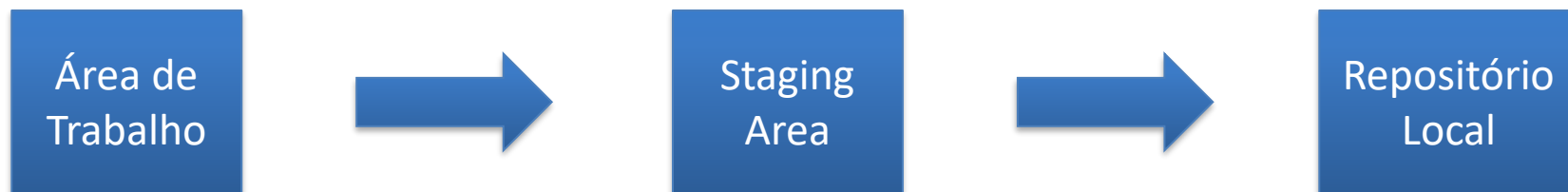
- `git help`
  - Oferece ajuda geral sobre o git
- `git help <comando>`
  - Oferece ajuda sobre um comando específico do git
- Demais comandos dão dicas do que pode ser feito (leia com atenção as saídas dos comandos!)

# Conceitos básicos: quem sou eu?

- `git config --global user.name <seu nome>`
  - Configura o nome do usuário
- `git config --global user.email <seu email>`
  - Configura o email do usuário
- Existem 3 níveis: Local, System, Global
  - A resolução é sempre o mais específico

# Conceitos básicos: *staging area*

- Área onde são colocados os arquivos que pretendemos enviar para o repositório



# Conceitos básicos: *commit* id

- Cada sistema de controle de versão usa uma estratégia diferente para identificar *commits*
  - Número sequencial por arquivo (CVS)
  - Número sequencial por repositório (Subversion)
  - Hash (Git e Mercurial)

# Conceitos básicos: *commit* id

- Hash (Git e Mercurial)
  - O ponteiro para os pais é na verdade o Hash dos pais
- Isso gera um efeito interessante!
  - Se eu altero um commit no inicio da historia...
  - Eu inviabilizo toda a historia!
- Hash do commit mais recente aponta para o Hash dos commits anteriores
  - É possível detectar fraude no histórico

# Conceitos básicos: apelidos

- A versão base do seu espaço de trabalho
  - *HEAD*
- O ramo principal do seu repositório
  - *master*
- O repositório do qual seu repositório foi clonado
  - *origin*

# Repositório local

- `git init <nome>`
  - Cria um repositório Git no diretório
- `git status`
  - Inspecciona o espaço de trabalho
- `git add <arquivo>`
  - Adiciona um arquivo na *staging area* para ser enviado ao repositório no próximo *commit*
- `git commit -m <mensagem>`
  - Envia os arquivos que estão na *staging area* para o repositório

# Inspecionando mudanças

- `git log [--graph] [--decorate=short] [--name-status]`
  - Inspeciona o histórico do repositório local
- `git show`
  - Inspeciona um *commit*
- `git diff`
  - Compara o espaço de trabalho com a staging area ou com alguma versão do repositório

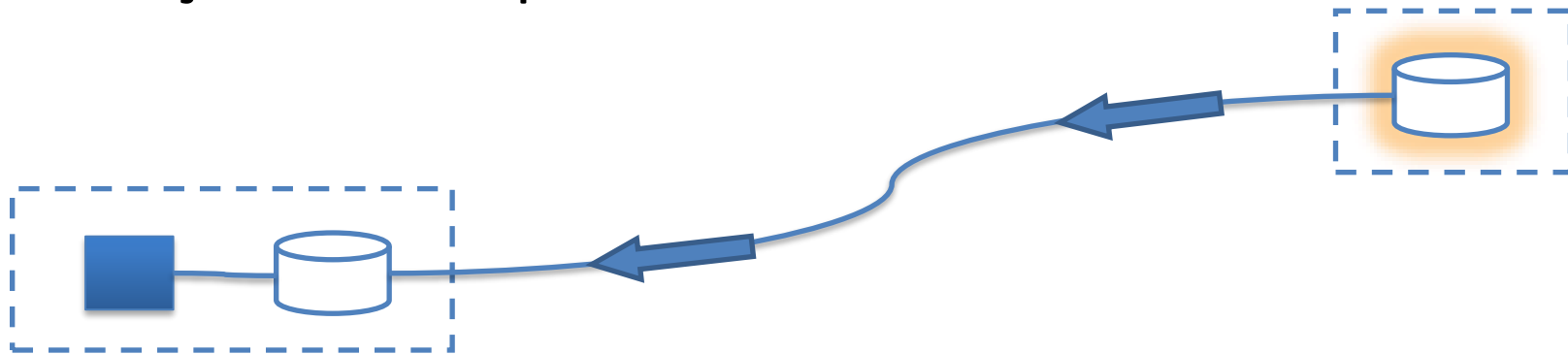


# Repositório local com ramos

- `git branch --all -v`
  - Lista os ramos existentes no repositório
- `git branch <nome do ramo>`
  - Cria um ramo à partir da versão indicada no HEAD
- `git branch -d <nome do ramo>`
  - Remove um ramo
- `git checkout <commit id ou nome do ramo>`
  - Troca a versão base do espaço de trabalho
- `git merge <nome do ramo>`
  - Combina um ramo com o ramo corrente

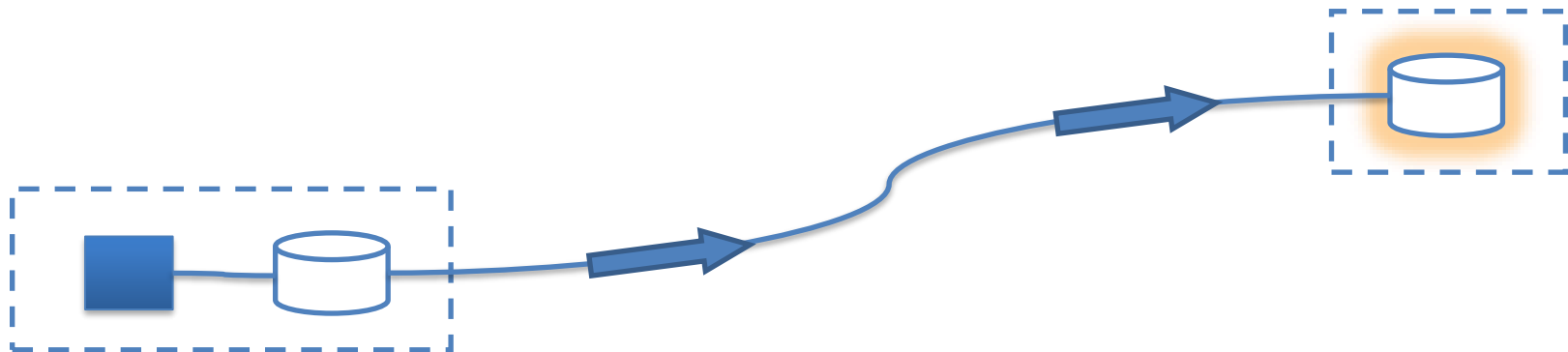
# Repositório remoto

- `git clone <url> <diretório>`
  - Cria um repositório local copiando o histórico de um repositório remoto
- `git pull <url>`
  - Atualiza o repositório local e o espaço de trabalho em relação a um repositório remoto



# Repositório remoto

- `git push <url> <branch>`
  - Atualiza o repositório remoto em relação ao repositório local

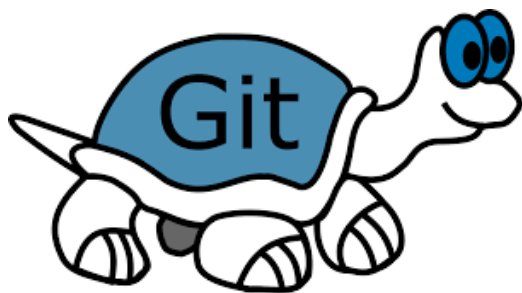


# Interface gráfica

- É possível fazer todos esses passos de forma visual



axosoft  
**GitKraken**



TortoiseGit



# Principais referências bibliográficas

- Conradi, R. and Westfechtel, B. Version Models for Software Configuration Management. ACM Computing Surveys, v.30, n.2, p. 232-282, 1998.
- Chacon, S. Pro Git. Apress, 1ª edição, 2009.

# Controle de Versão de Software com Git



**<https://tinyurl.com/y65wuhj3>**

# Múltiplos repositórios remotos

- `git remote -v`
  - Listar os repositórios remotos cadastrados
- `git remote add <nome> <url>`
  - Adiciona um novo repositório remoto
- `git remote remove <nome>`
  - Remove um repositório remoto existente

# Marcando versões especiais

- `git tag`
  - Lista os rótulos existentes
- `git tag <nome do rótulo> [commit id]`
  - Cria um rótulo sobre um dado commit (HEAD por default)
- `git tag -d <nome do rótulo>`
  - Remove um rótulo