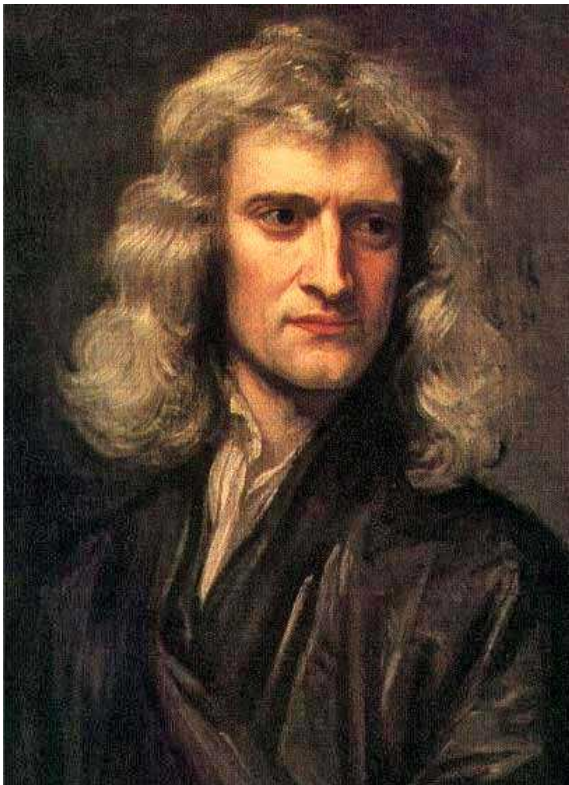


# Uso de Git no apoio à ciência aberta



# Em 1675 o Isaac já dizia...



"Se eu vi mais longe, foi por estar sobre ombros de gigantes."

Isaac Newton

# Qual caminho seguir?

Ciência  
aberta



Ciência  
fechada

# Ciência aberta...



<https://www.fosteropenscience.eu/content/what-open-science-introduction>

# Como seguir na direção de ciência aberta?

- Adotar um serviço de hospedagem de código
  - Ex. GitHub
- Escolher uma licença apropriada
  - Ex. MIT
- Disponibilizar todos os materiais
  - Ex. código, dados, protocolos, exemplos, etc.
- Documentar adequadamente o projeto
- Facilitar o processo de uso/instalação
- Acolher potenciais usuários

# Por exemplo...

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

# Git vs GitHub



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

# Mas afinal, o que são versões?



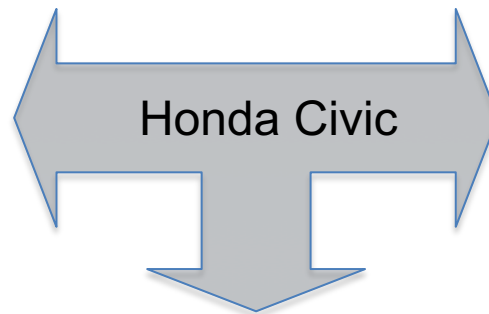
(Conradi and Westfechtel 1998)



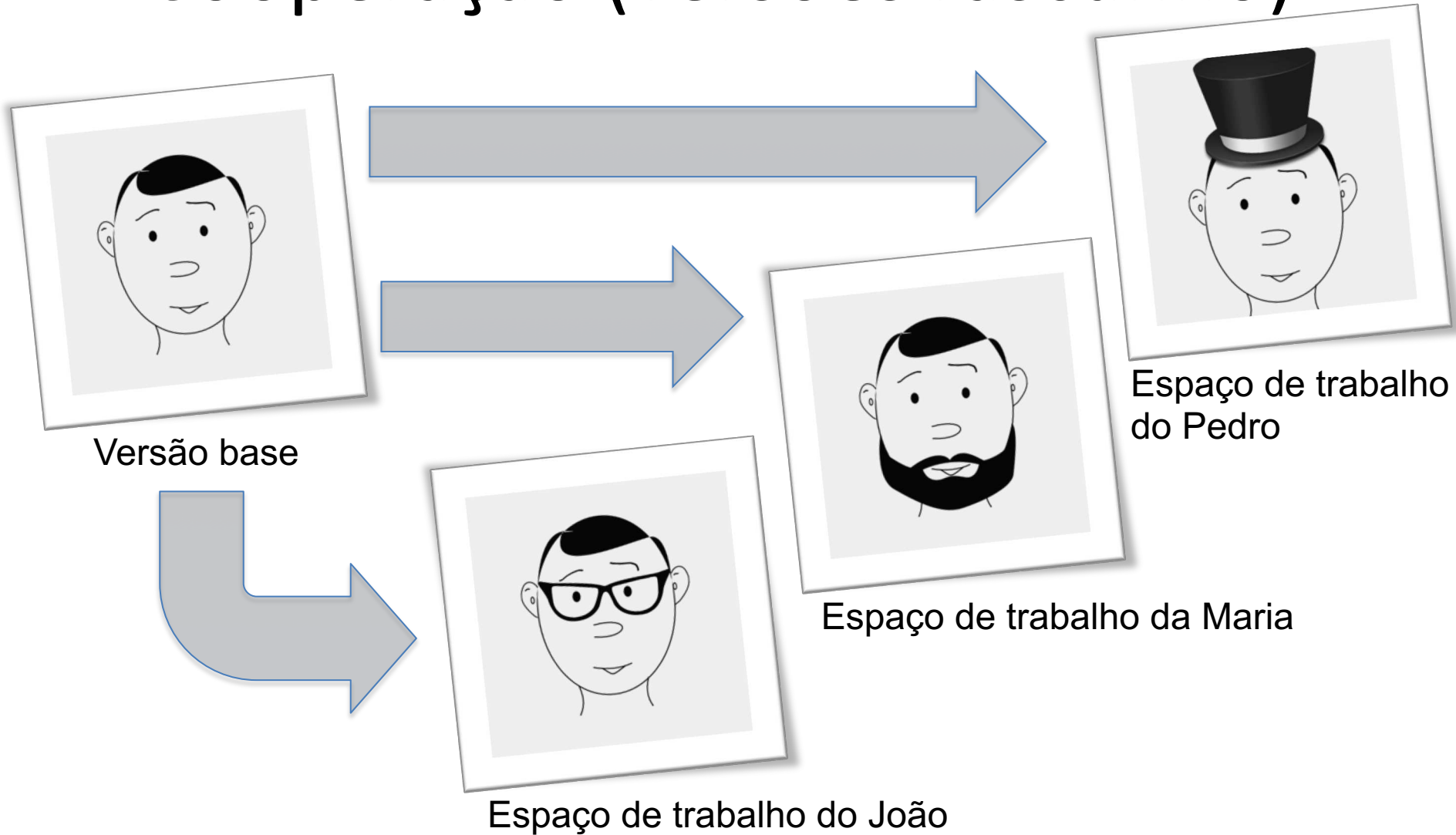
# Revisões



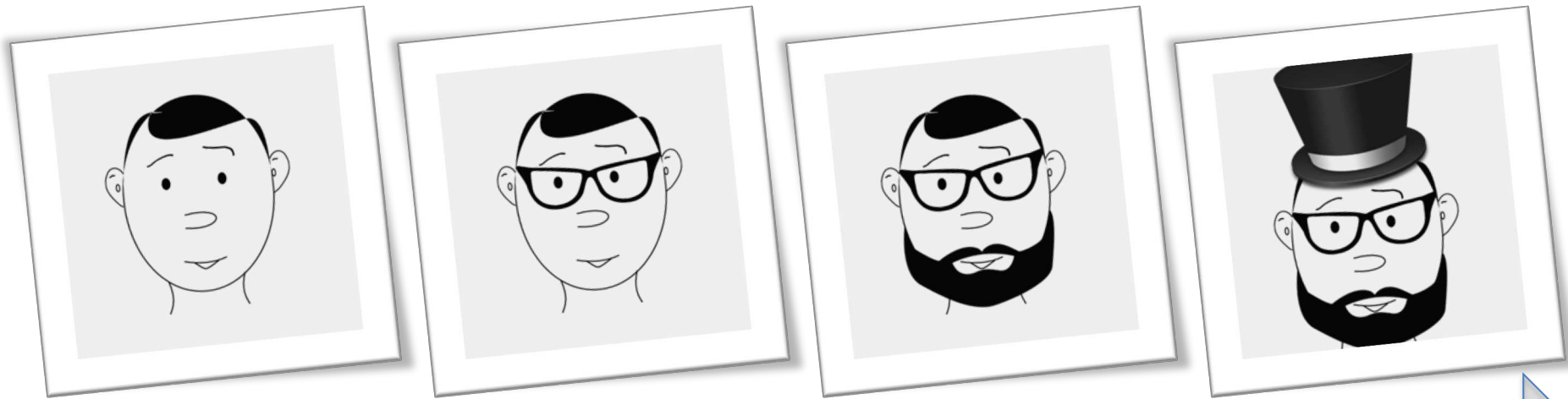
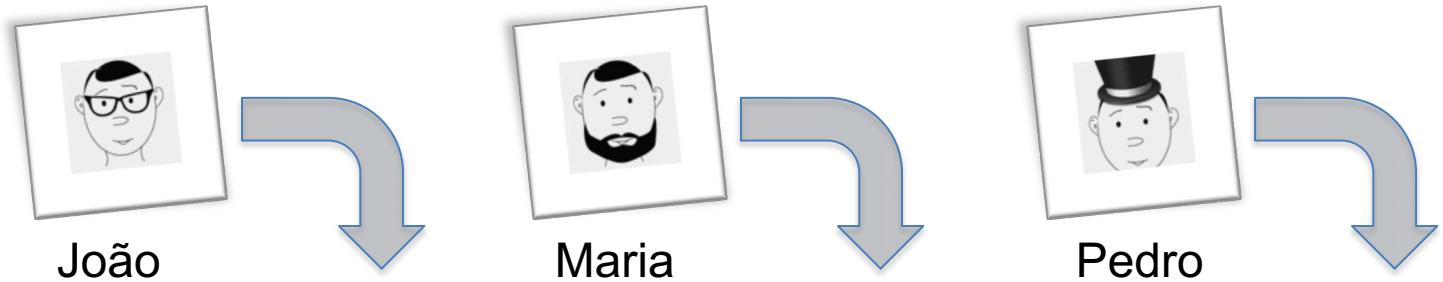
# Variantes



# Cooperação (versões rascunho)

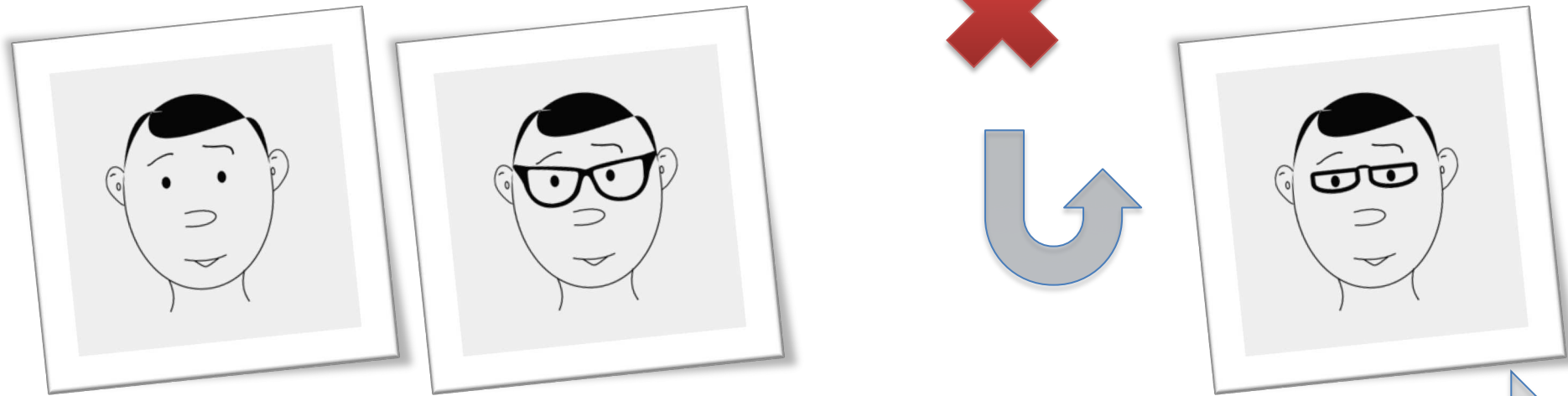
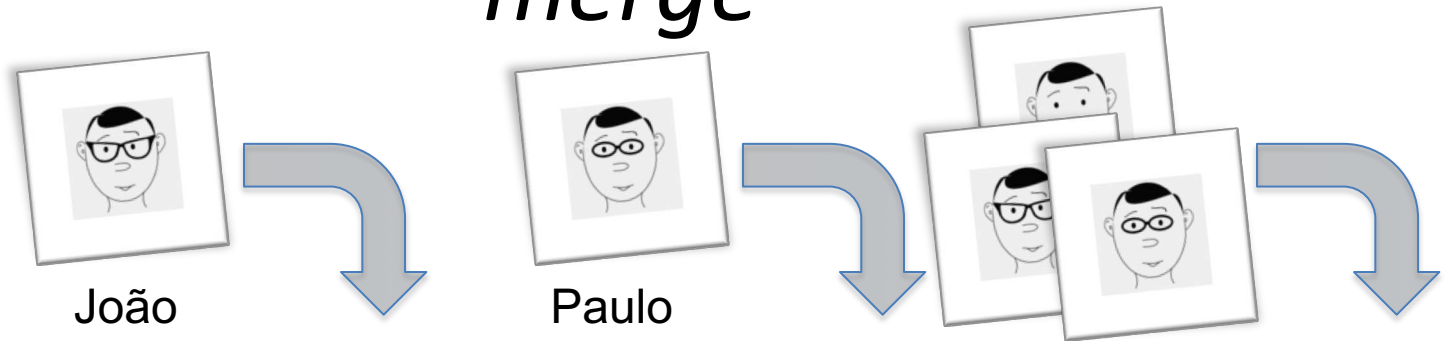


# Versões de rascunho podem ser combinadas (operação de *merge*)

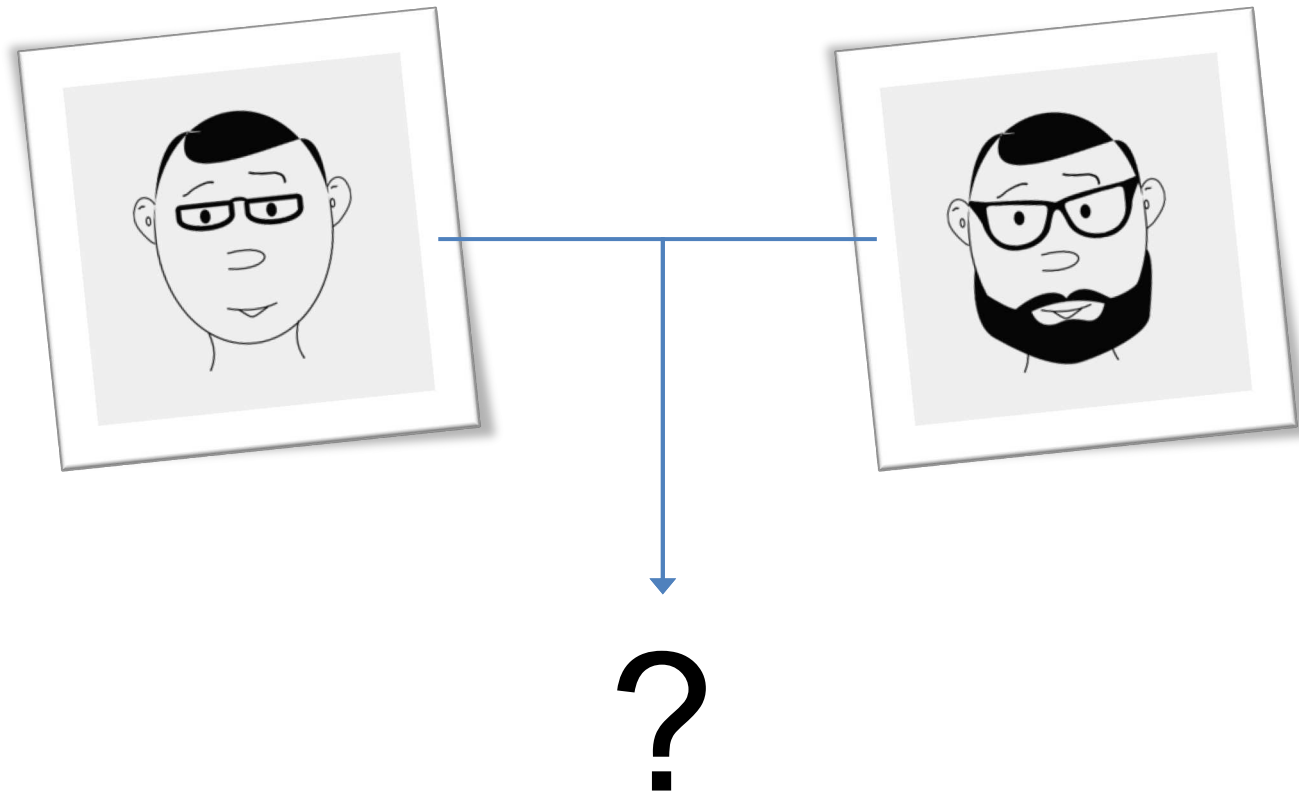


Revisões

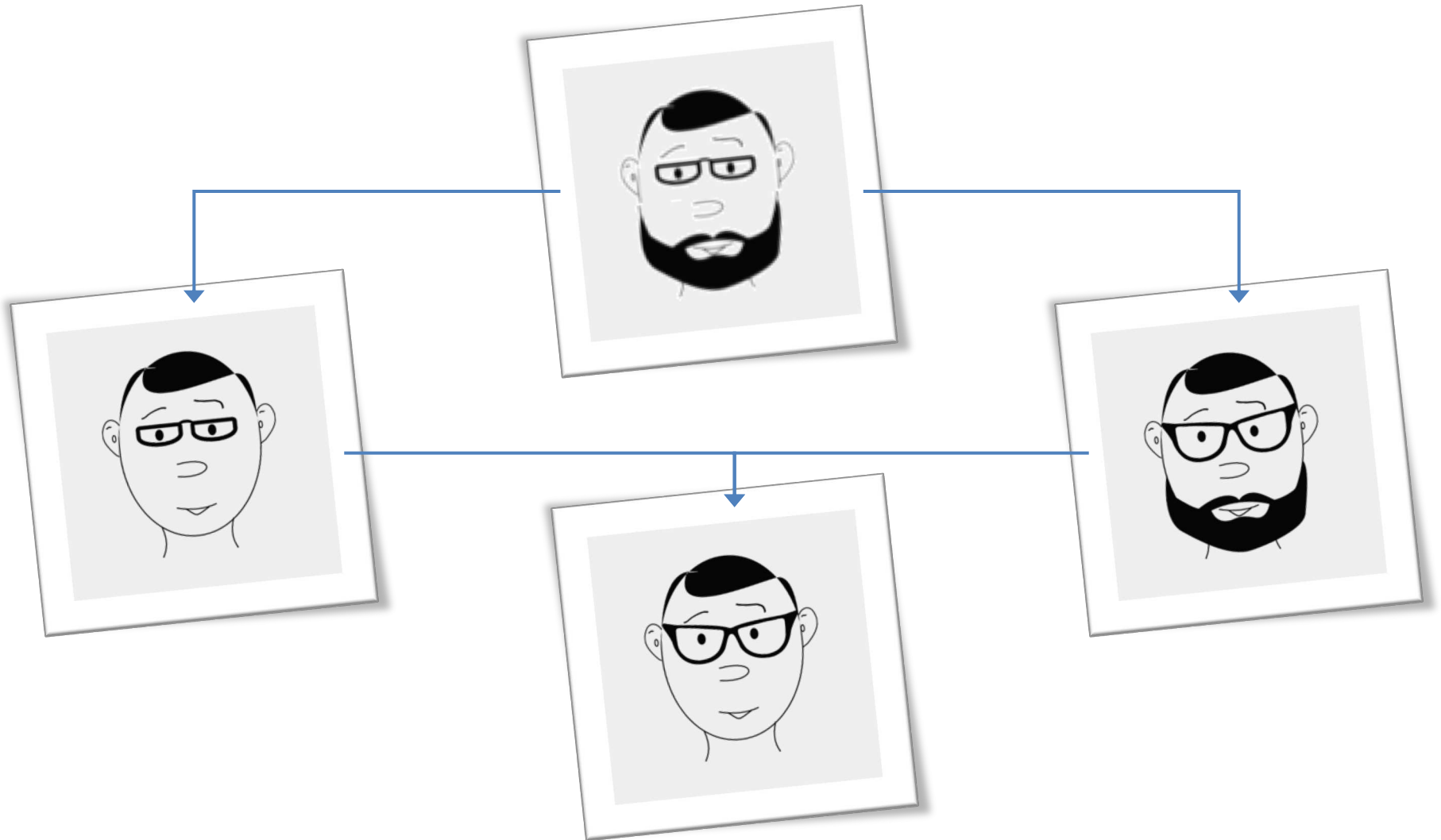
# Conflitos podem ocorrer durante o *merge*



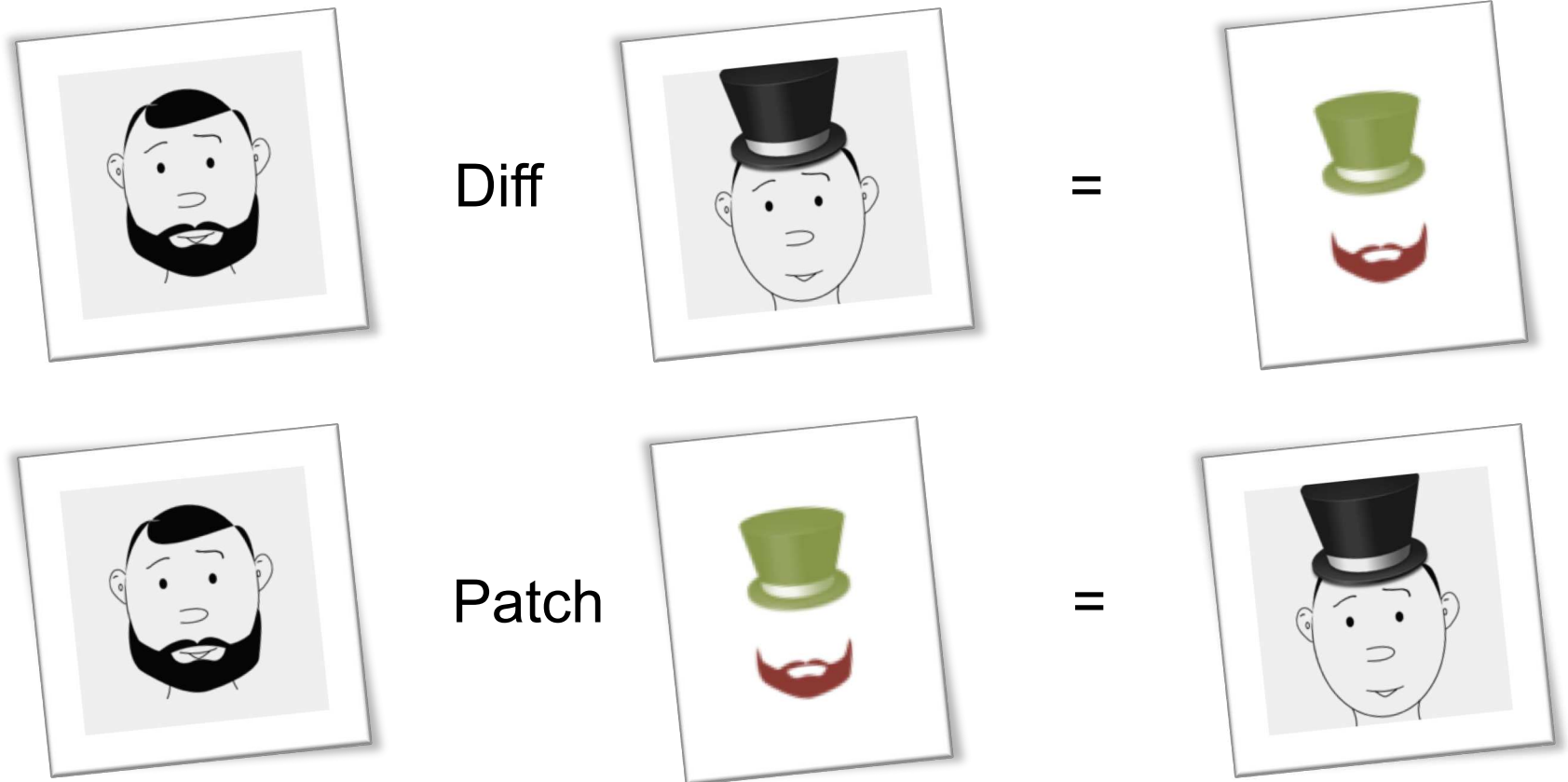
# 2-way merge



# 3-way merge



# Outras duas operações importantes...

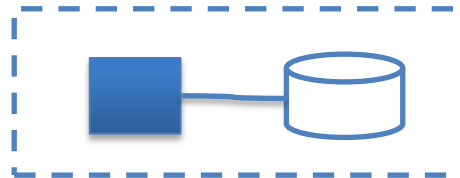


... para guardar, transferir e compreender versões.



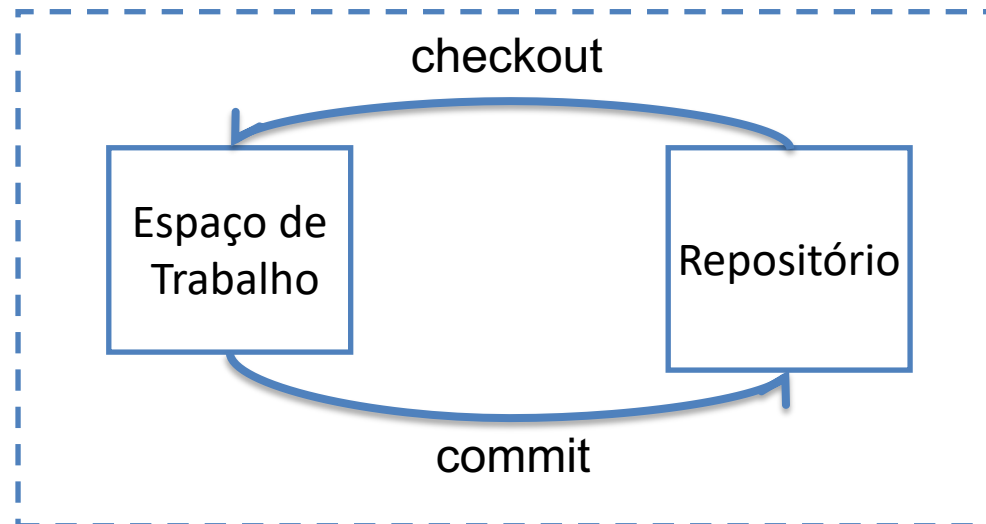
# Histórico dos sistemas de controle de versões (VCS)

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



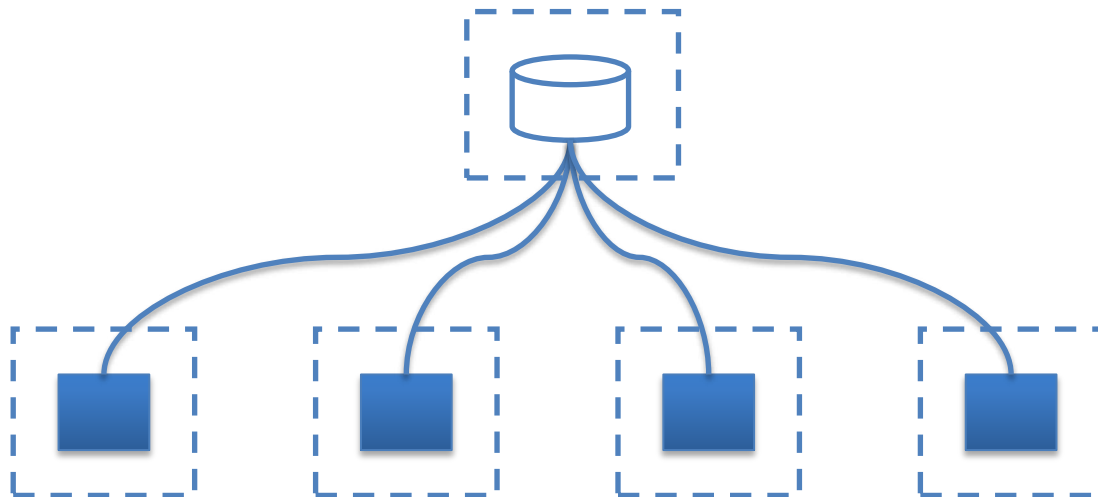
# Histórico dos sistemas de controle de versões (VCS)

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



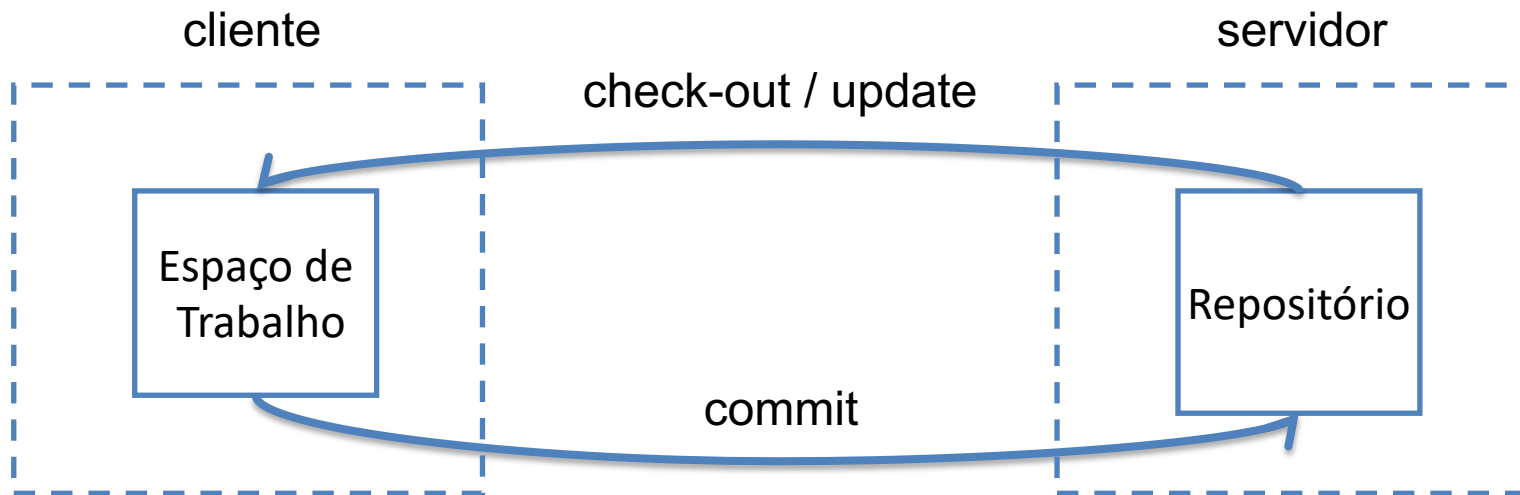
# Histórico dos sistemas de controle de versões (VCS)

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



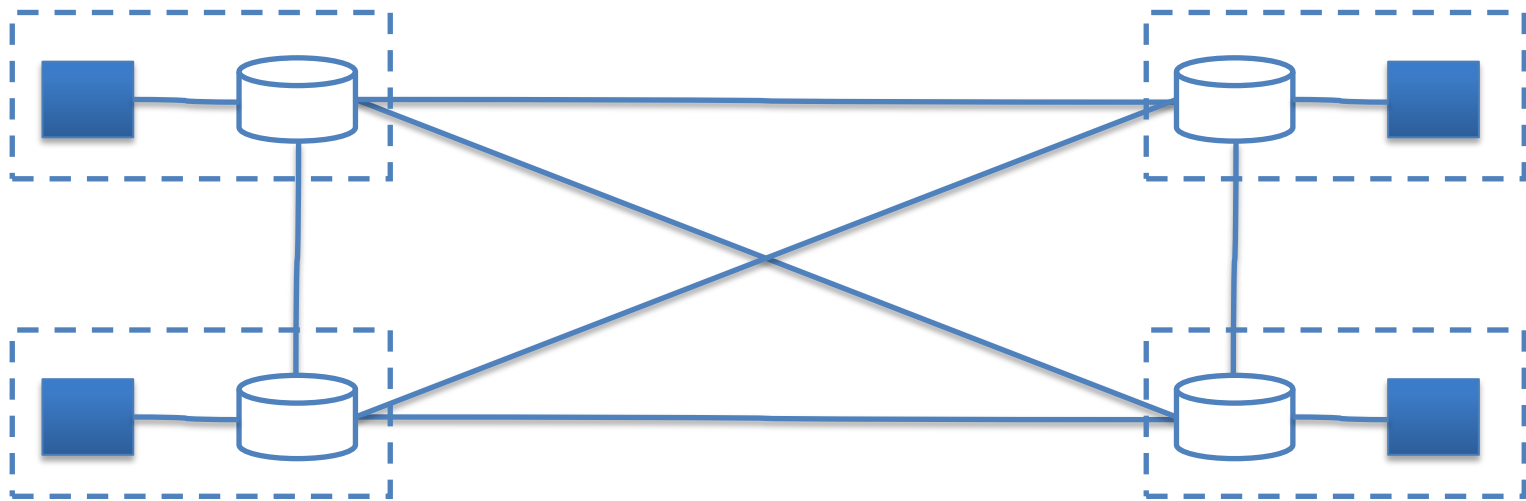
# Histórico dos sistemas de controle de versões (VCS)

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



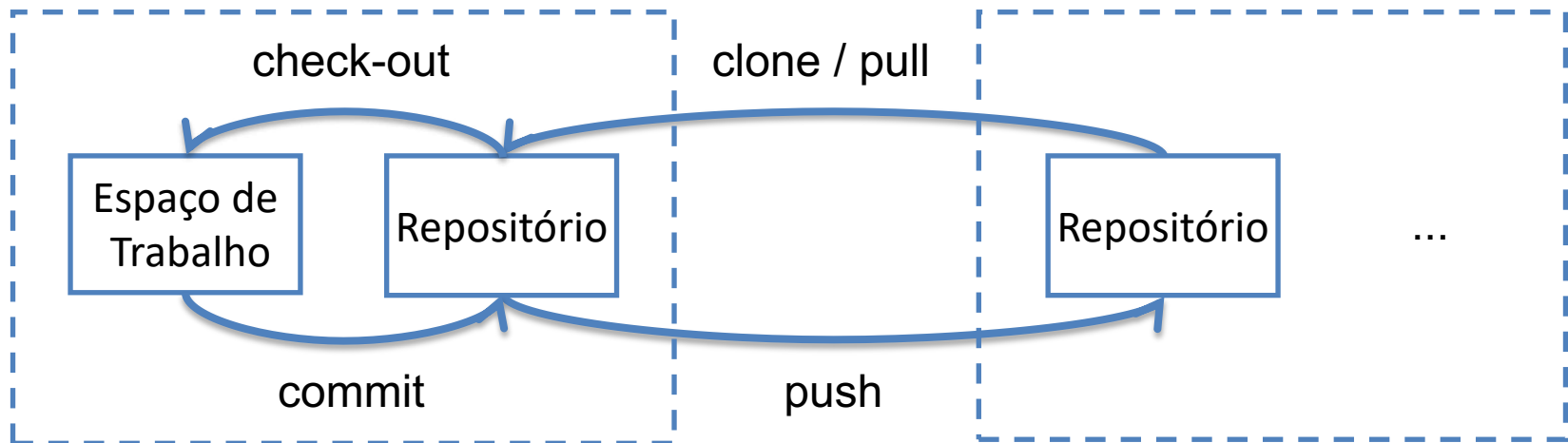
# Histórico dos sistemas de controle de versões (VCS)

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

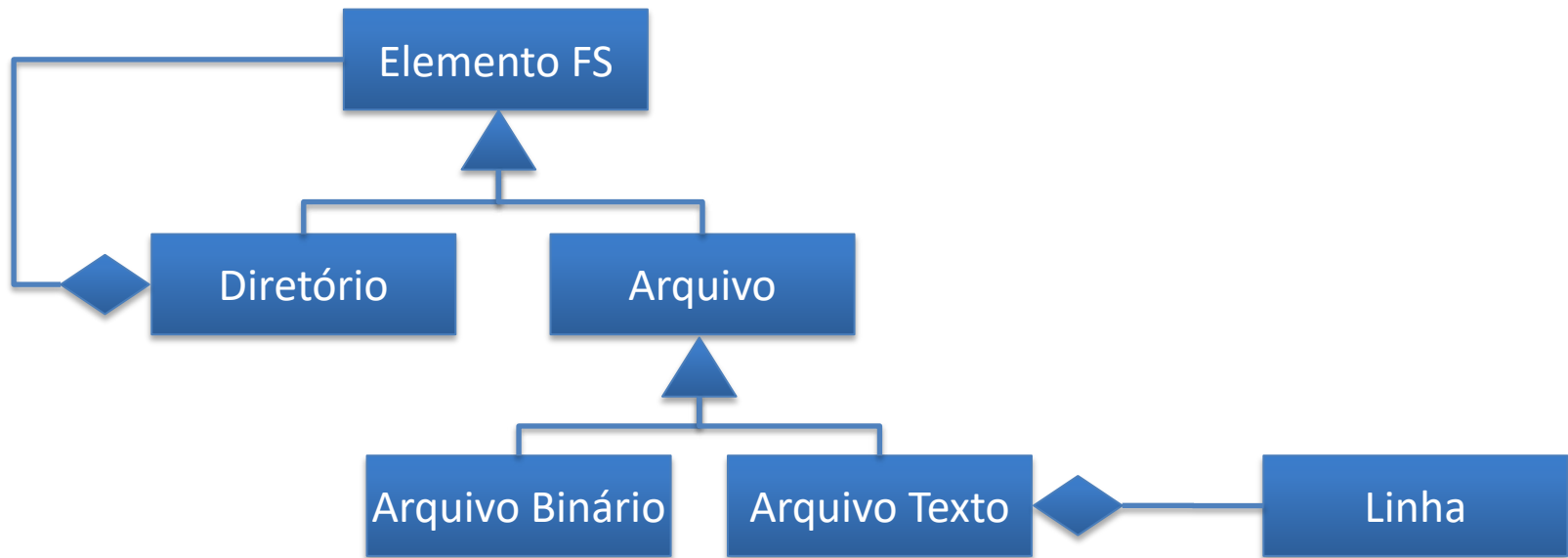


# Histórico dos sistemas de controle de versões (VCS)

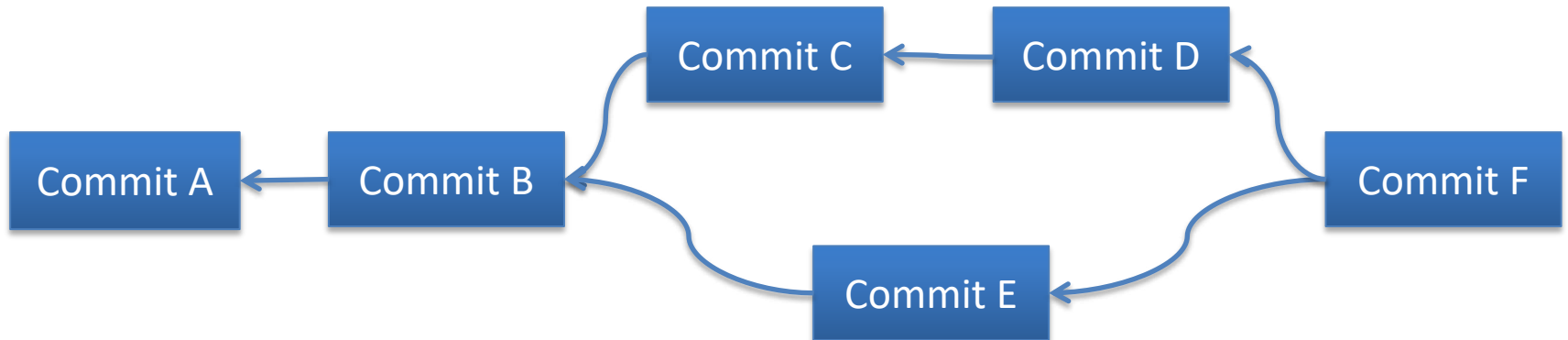
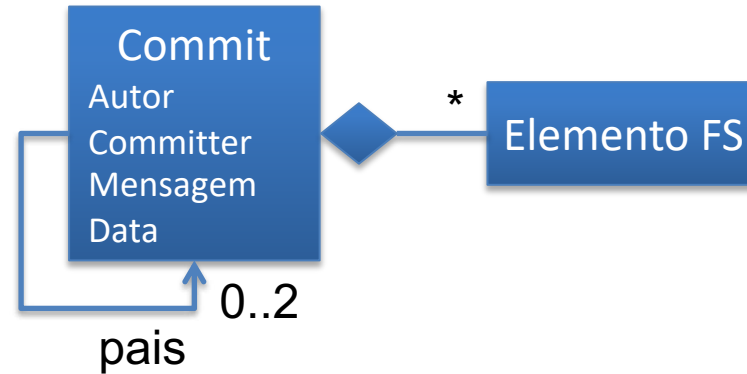
- Anos 2000 – Sistemas peer-to-peer
  - Git (2005)
  - Mercurial (2005)



# O que é versionado?



# Como é versionado?

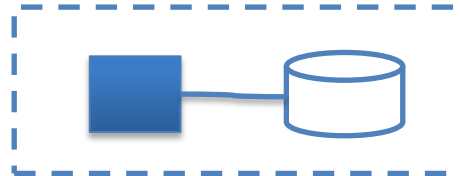




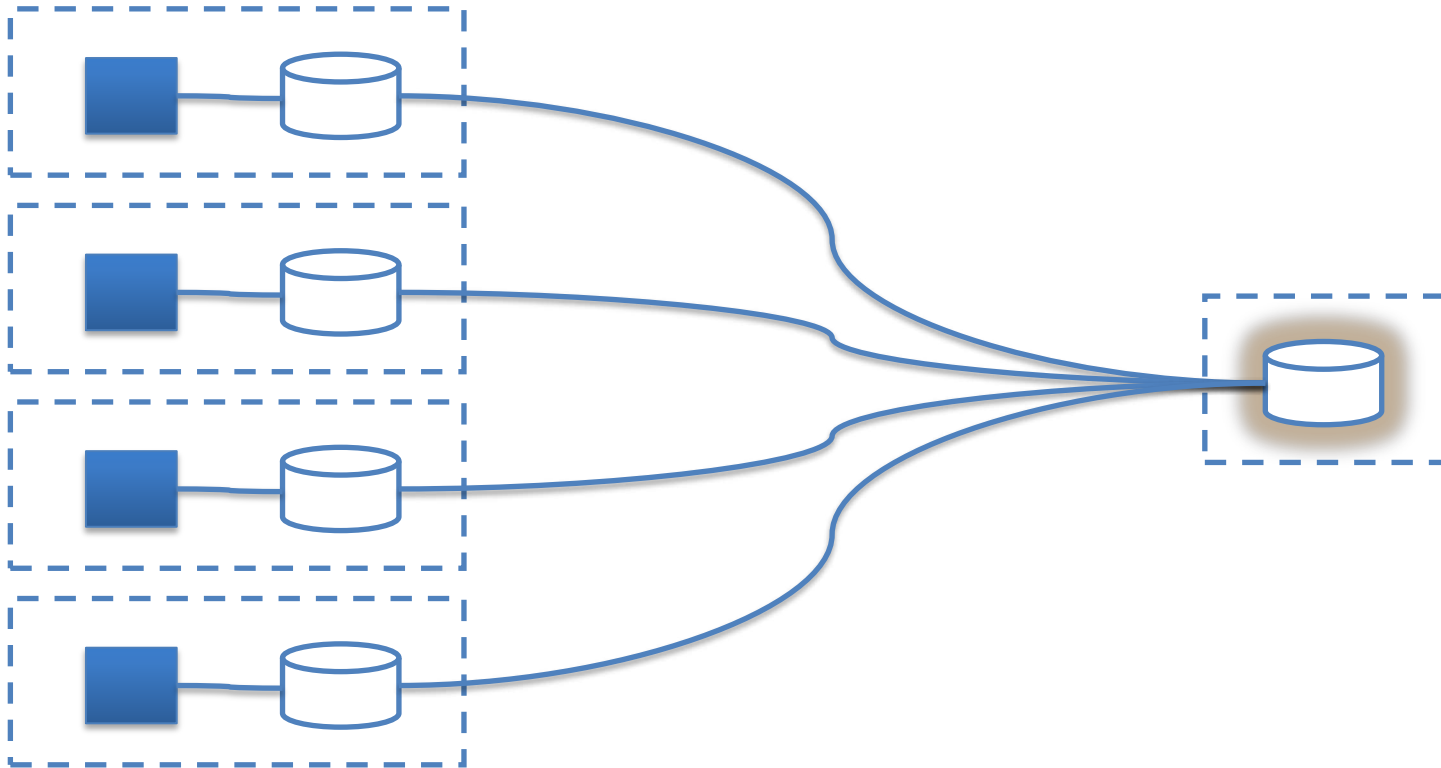
# Formas de adoção

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

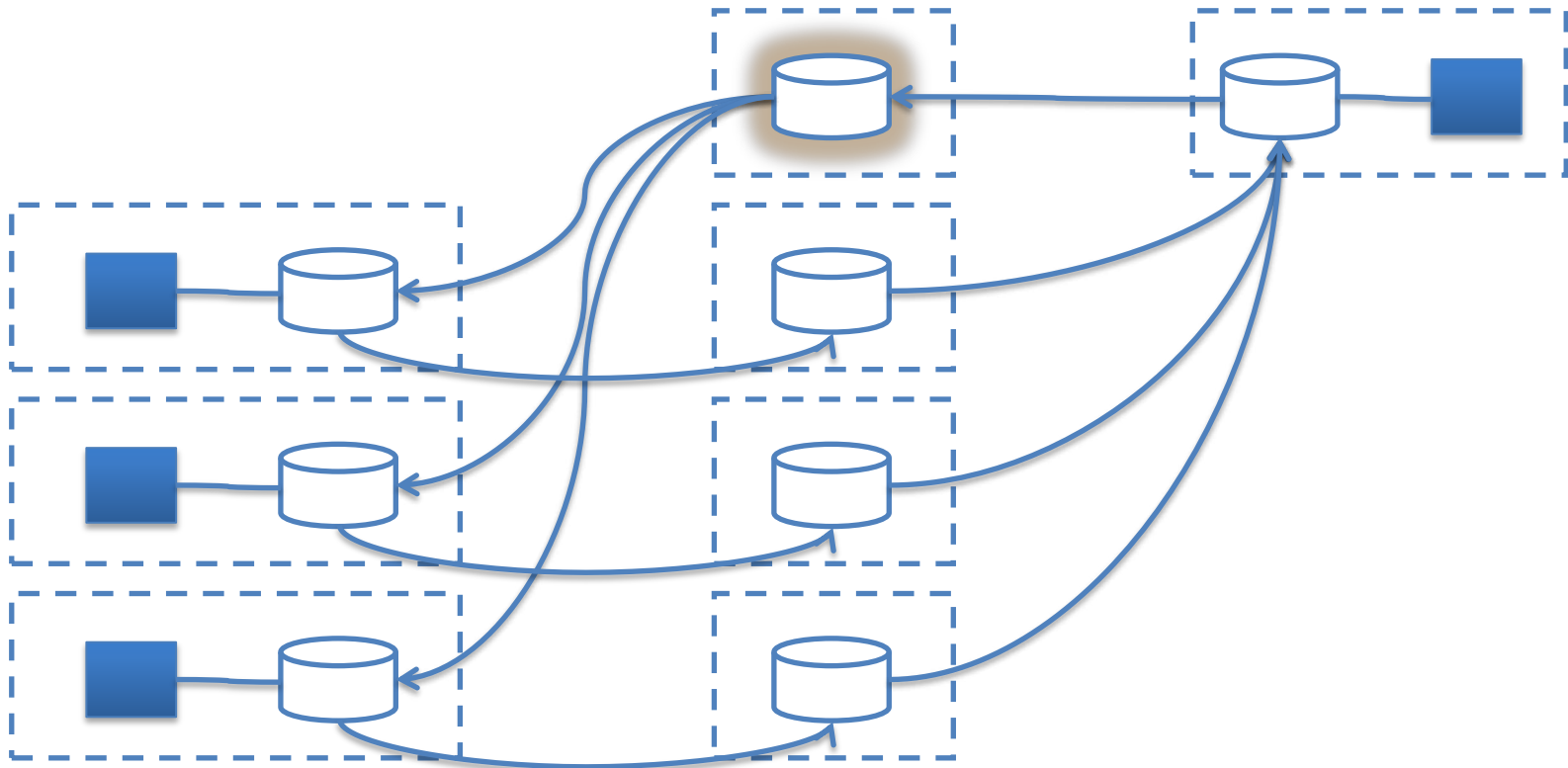
# Individual



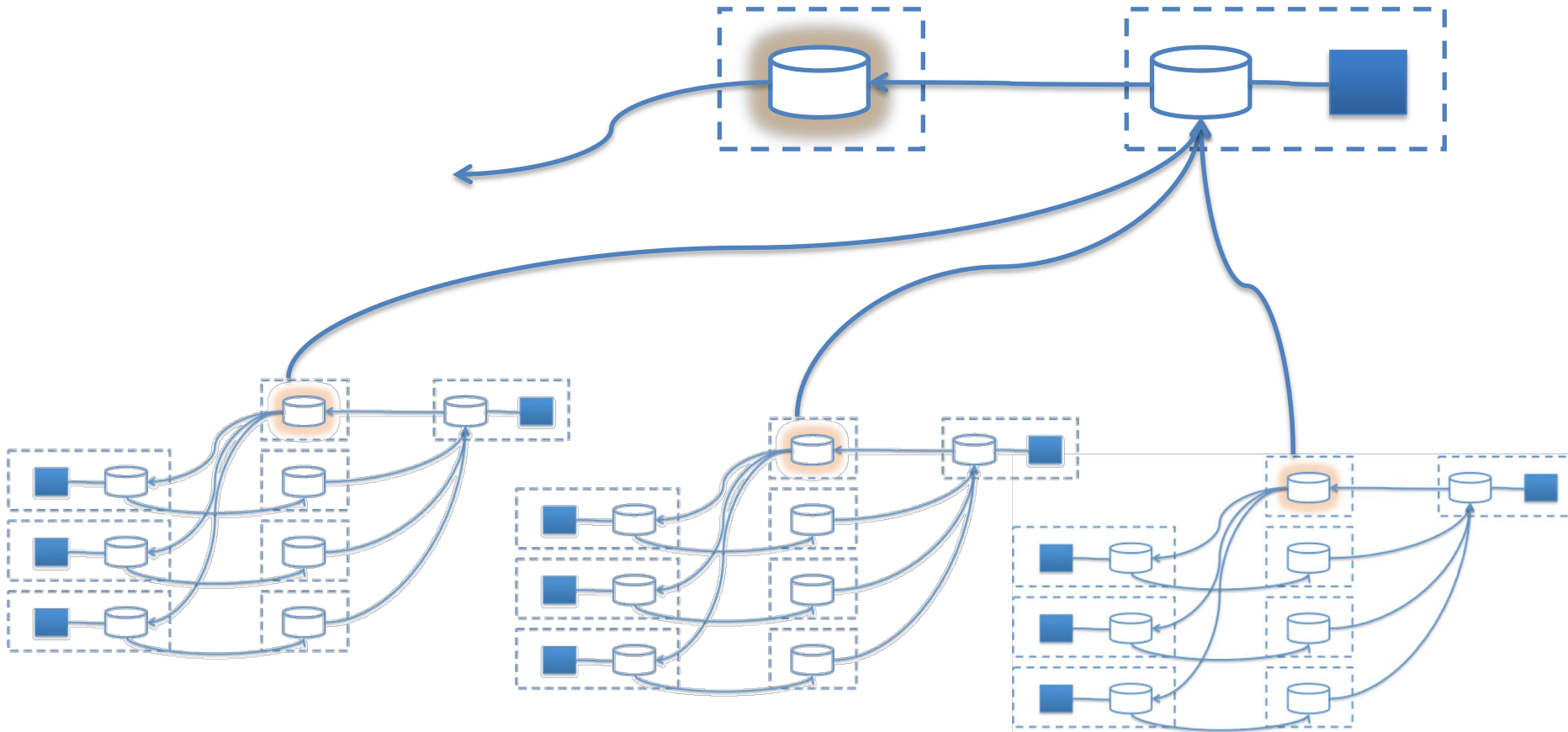
# Cliente-servidor



# 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
  - Múltiplos repositórios remotos

# 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!)

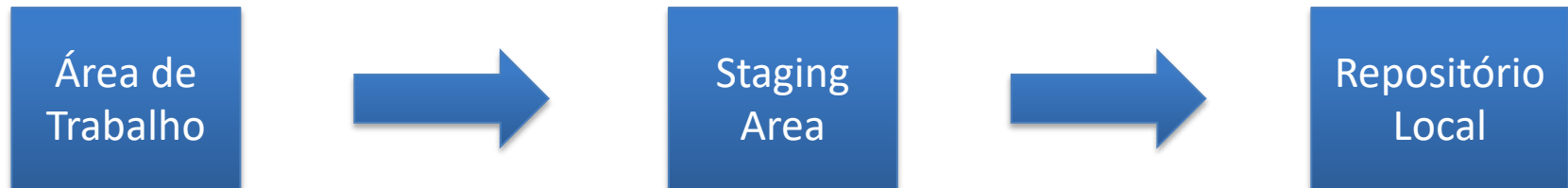
# Conceitos básicos: quem sou eu?

- `git config --global user.name <seu nome>`
  - Configura o nome do usuário
- `git config --global user.email <seu email>`
  - Configura o email do usuário



# 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: 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 add`
  - 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 status`
  - Inspeciona o espaço de trabalho
- `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

# Interface gráfica

- É possível fazer todos esses passos de forma visual
- Dentre várias ferramentas, vamos praticar com...



# 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

# 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`
  - Atualiza o repositório local e o espaço de trabalho em relação a um repositório remoto
- `git push`
  - Atualiza o repositório remoto em relação ao repositório local

# 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

# 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.

# Uso de Git no apoio à pesquisa

