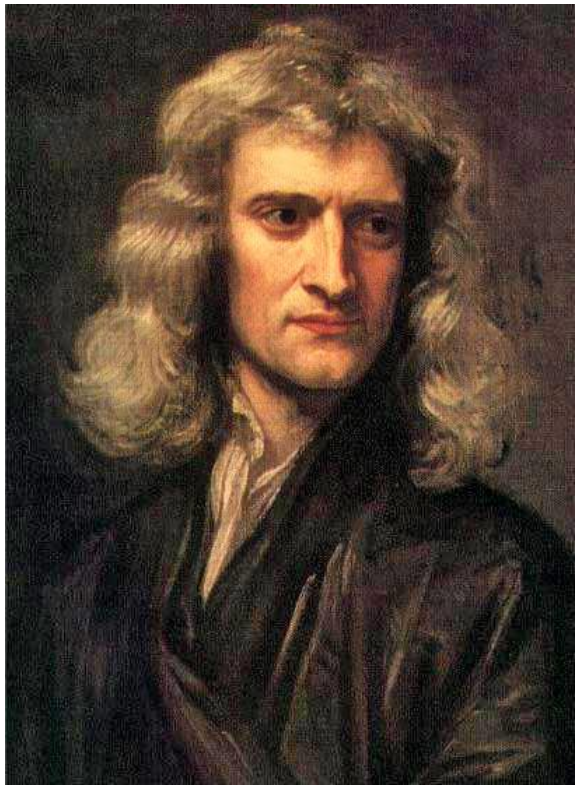
Uso de Git no apoio à ciência aberta

Instituto de Computação

Troy Costa Kohwalter

tkohwalter@ic.uff.br

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

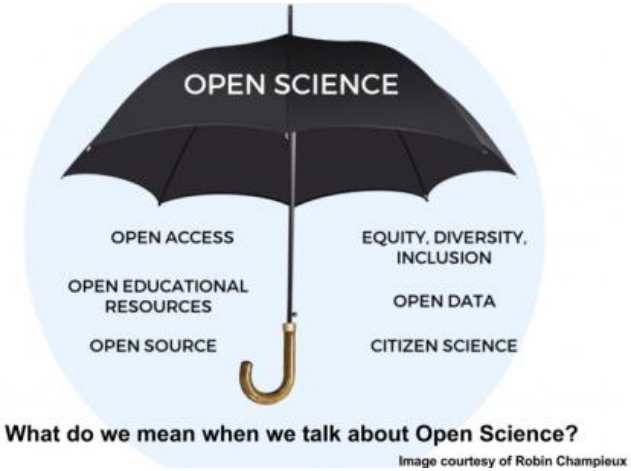
Qual caminho seguir?

Ciência Aberta

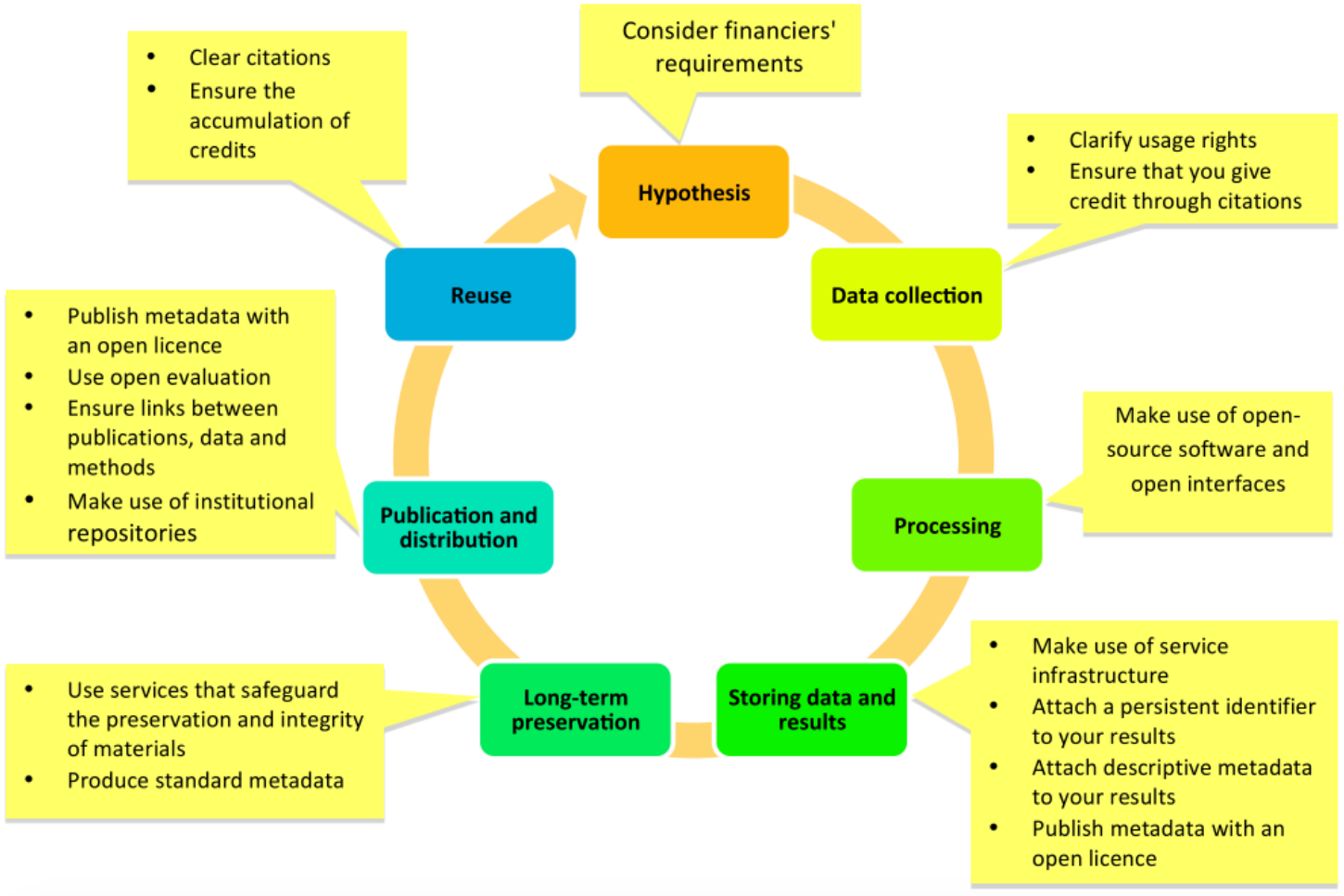
- Disponibilizar conhecimento
- Universidades
- Artigo cientifico
- Mestrado e doutorado

Ciência Fechada

- Pesquisa fechada
- Empresa privada
  - Investimento privado
- Patentes

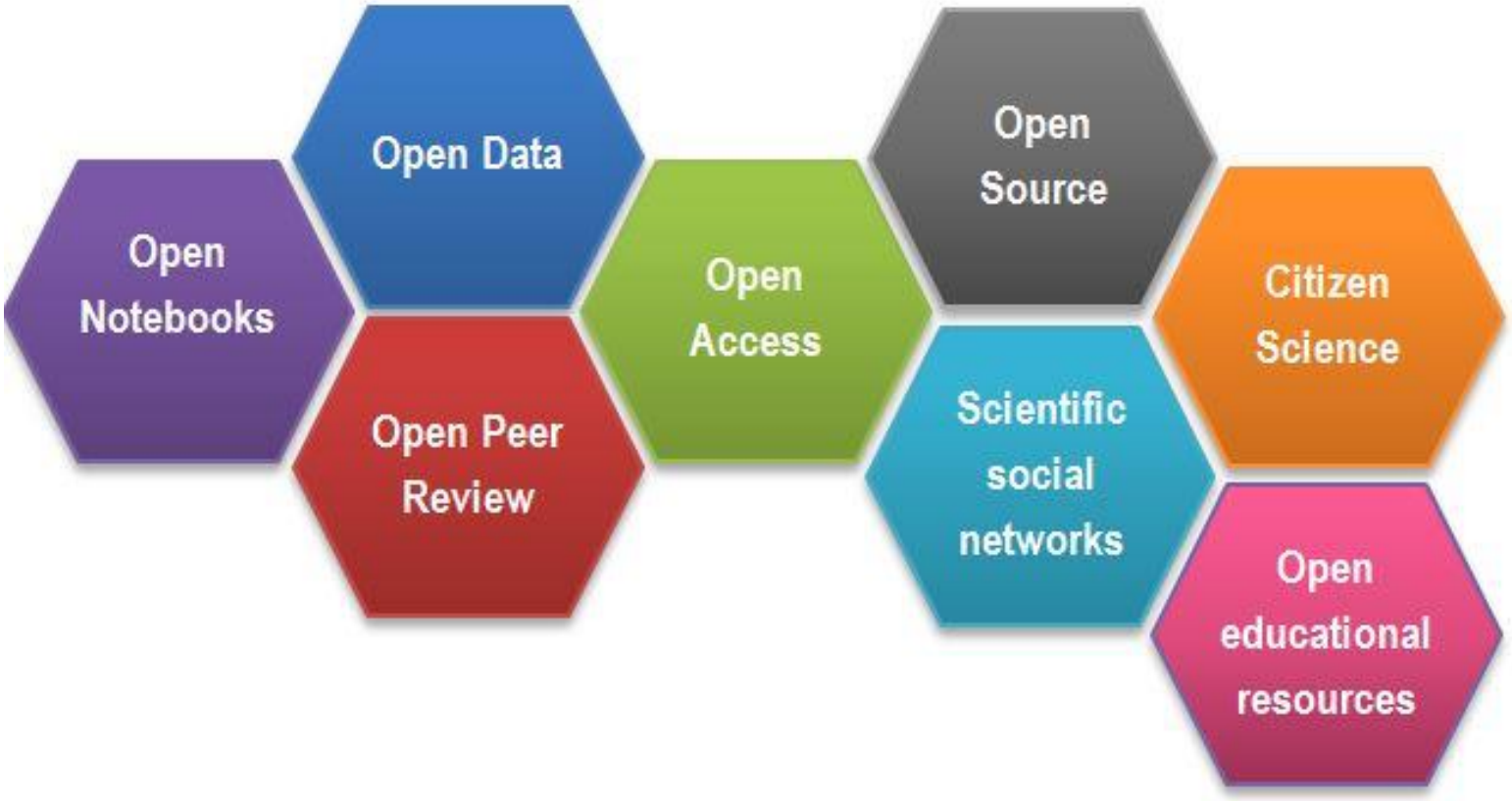


Ciência aberta...



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

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

Licença MIT

- Copyright <YEAR> <COPYRIGHT HOLDER>
- Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
- The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
- THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

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

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

noworkflow

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

Jupyter Notebook MIT 19 87 48 1 Updated 19 days ago

Top languages

Java Python Jupyter Notebook JavaScript Ruby

People 1 >

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
- 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

Hatchback

Honda Civic

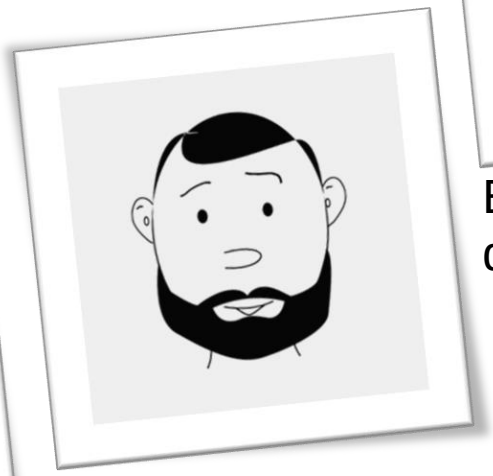
Sedan

Coupe

Cooperação (versões rascunho)

Versão base

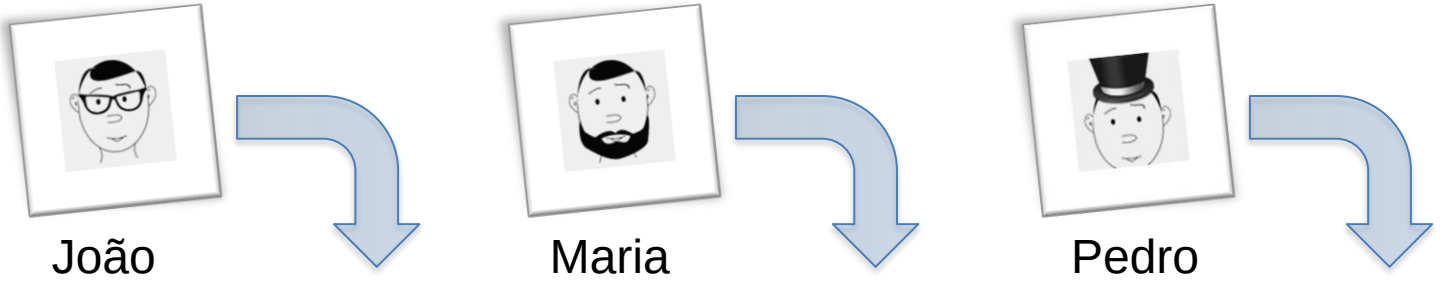
Espaço de trabalho do Pedro

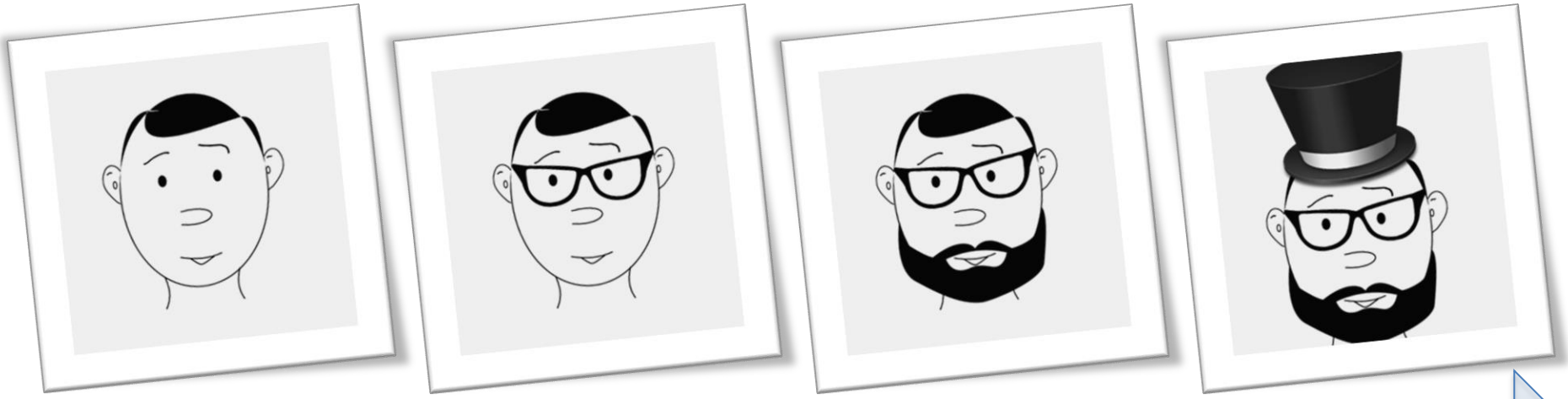


Espaço de trabalho da Maria

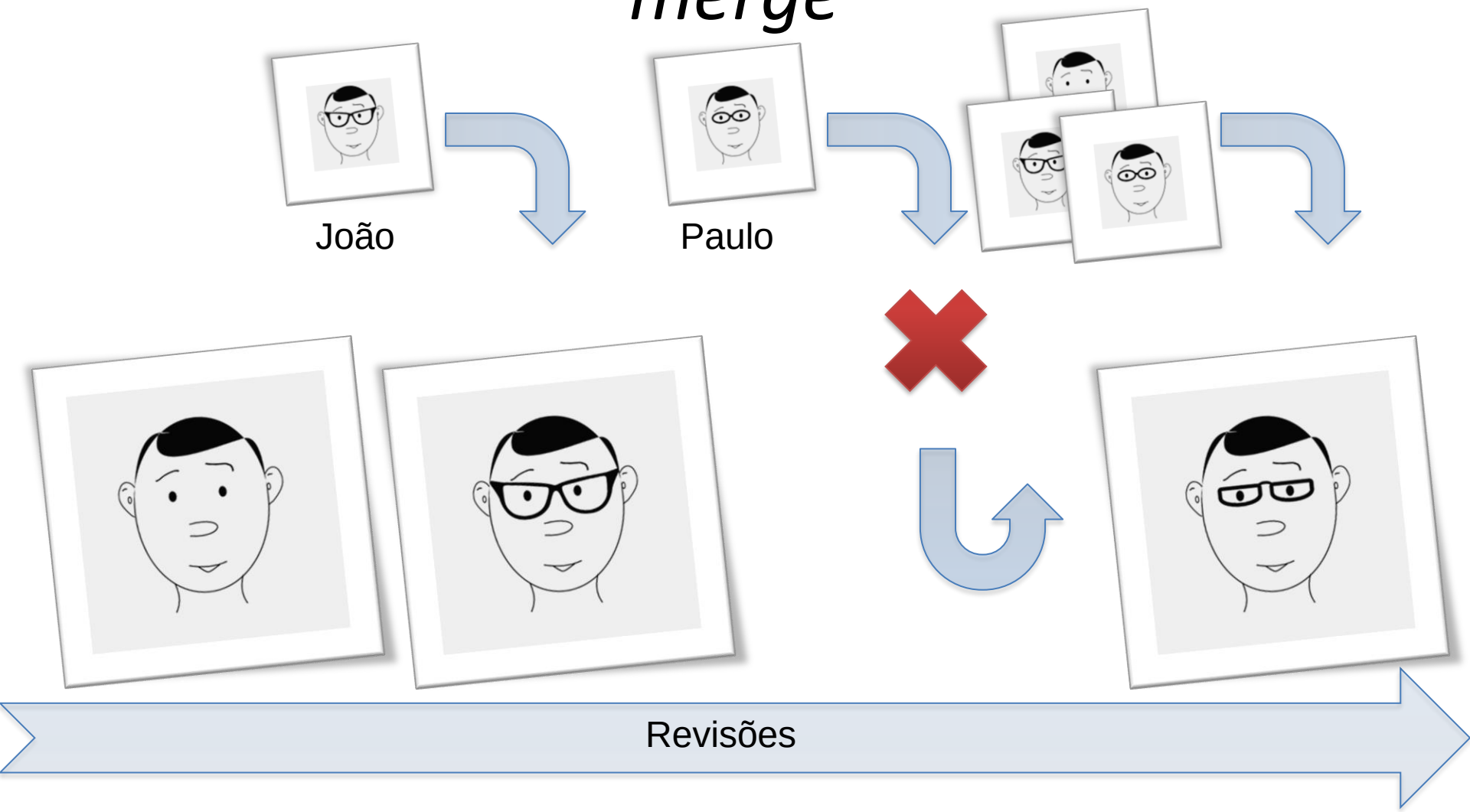
Espaço de trabalho do João

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

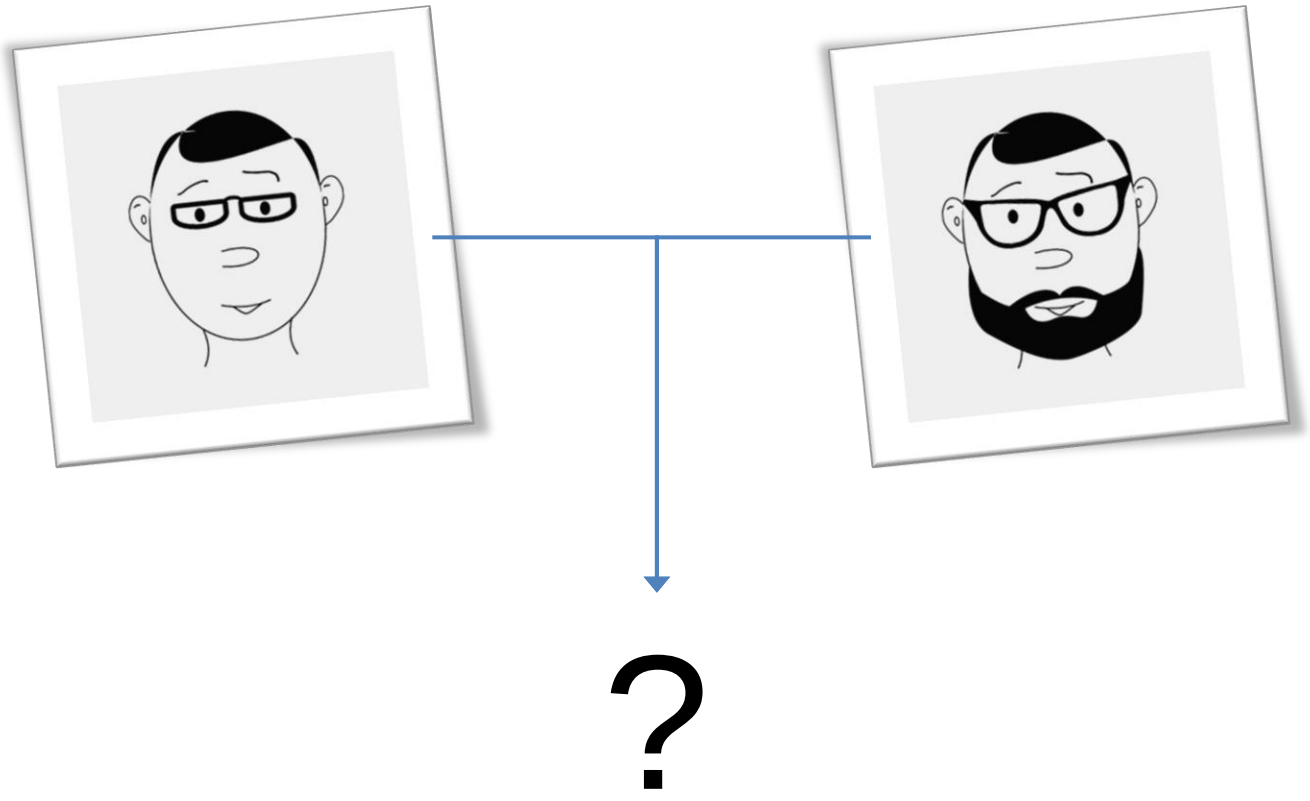




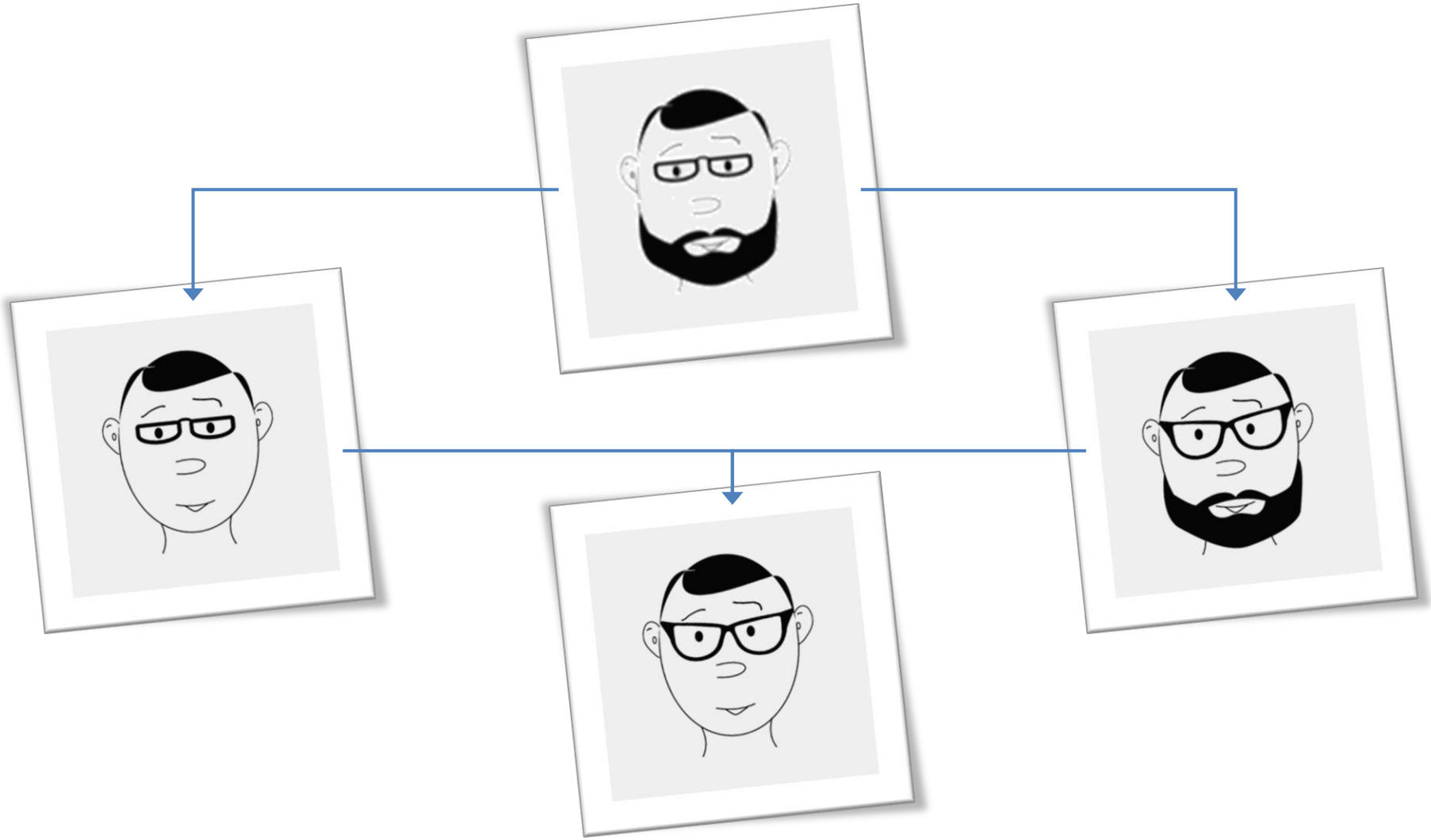
Conflitos podem ocorrer durante o *merge*



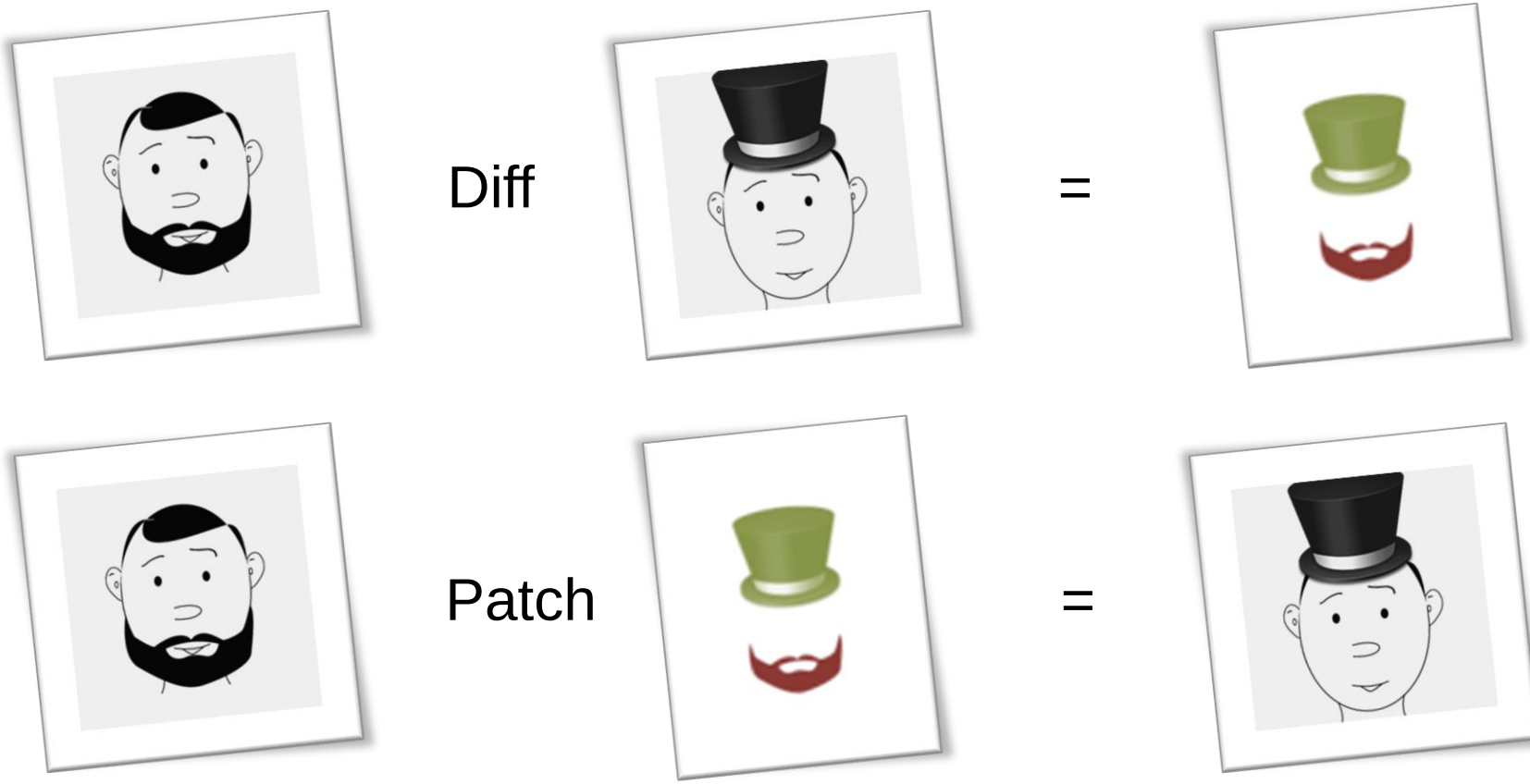
2-way merge



3-way merge



Outras duas operações importantes...

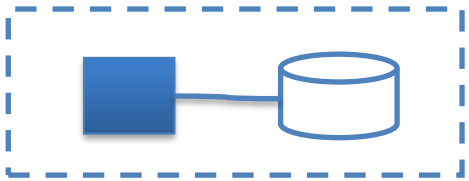


... para guardar, transferir e compreender versões.

Histórico dos sistemas de controle de versões (VCS)

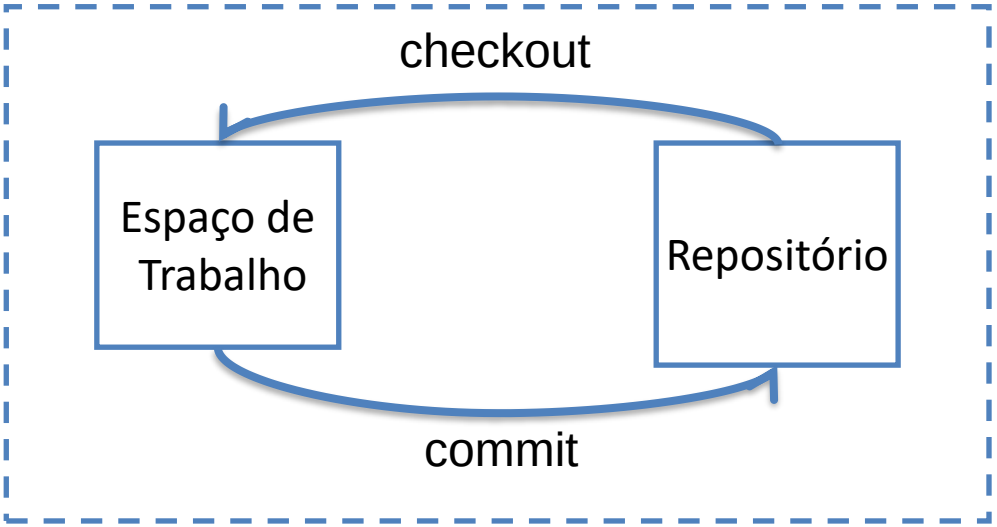
- Anos 70/80 – Sistemas locais
  - SCCS (1972)
  - RCS (1982)

- Repositório e espaço de trabalho estão na mesma máquina



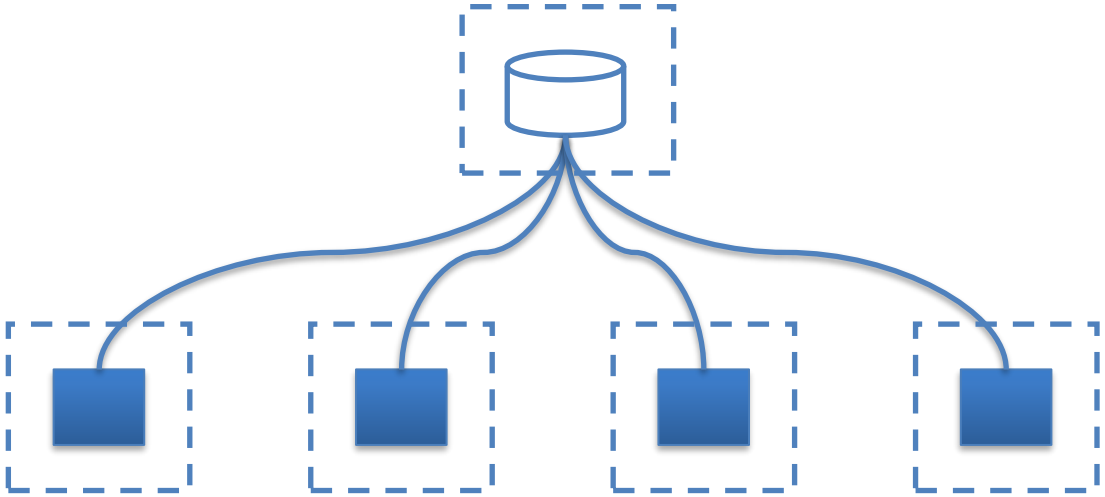
Histórico dos sistemas de controle de versões (VCS)

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



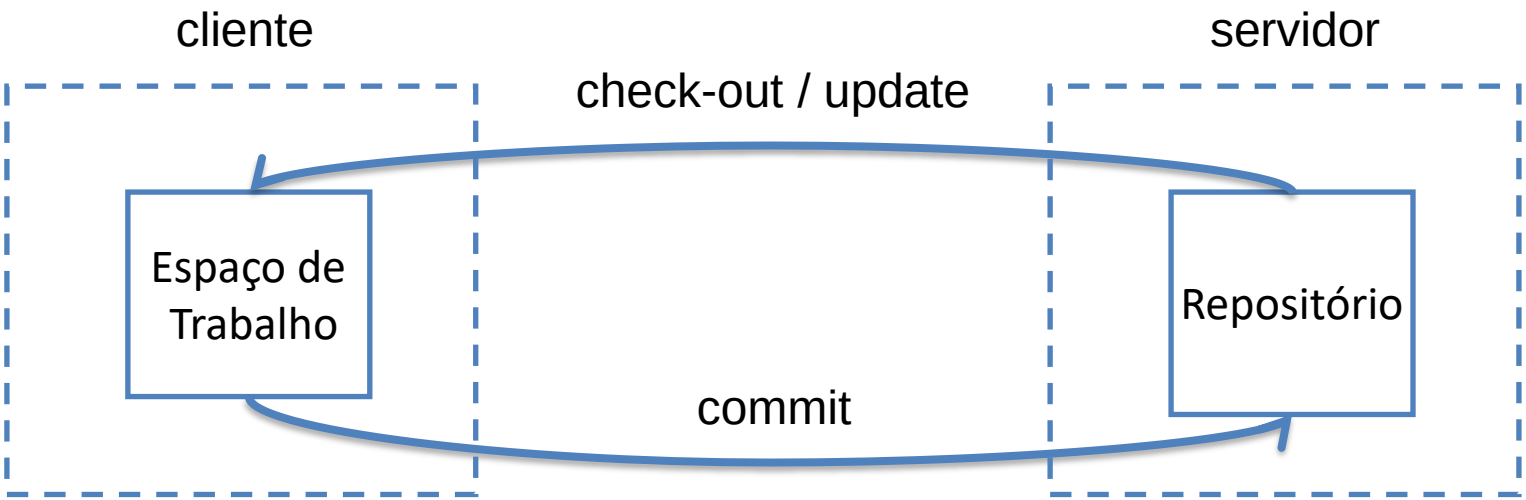
Histórico dos sistemas de controle de versões (VCS)

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



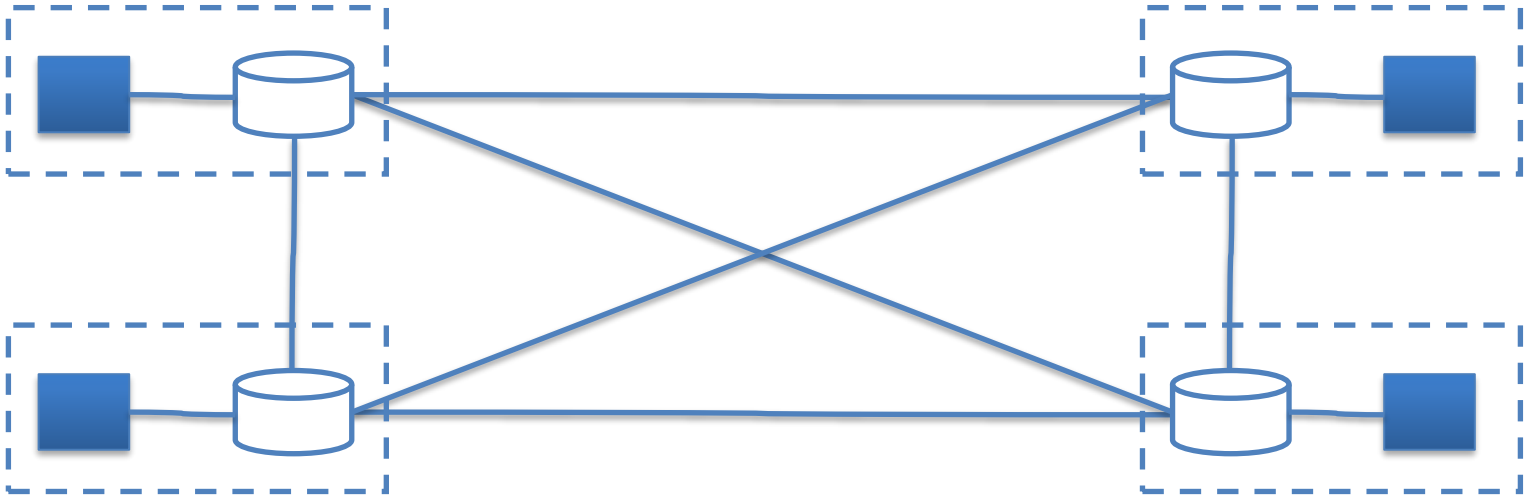
Histórico dos sistemas de controle de versões (VCS)

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



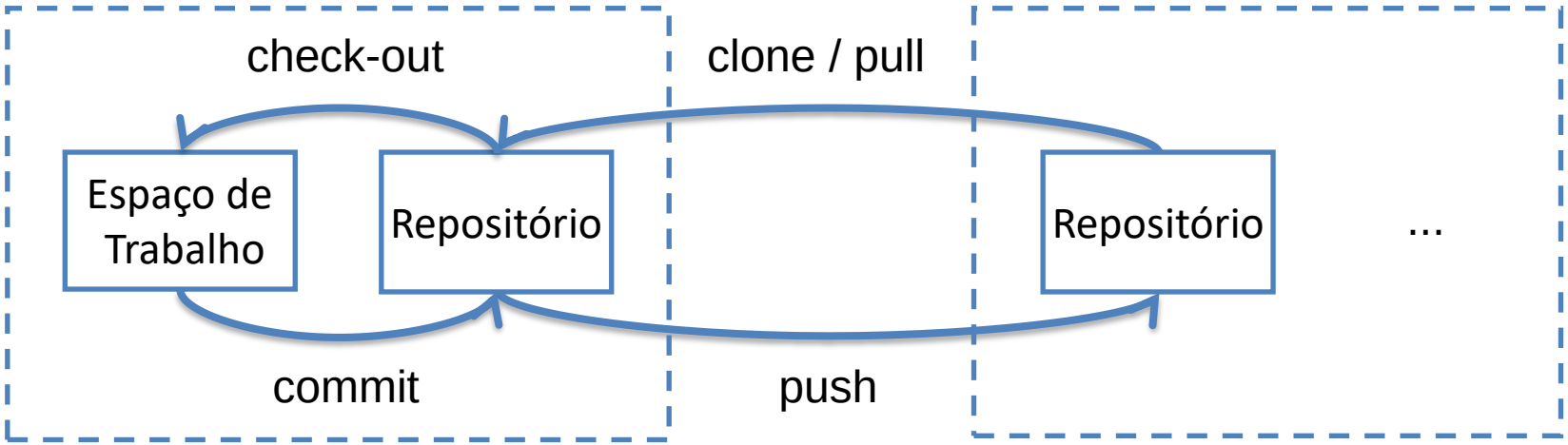
Histórico dos sistemas de controle de versões (VCS)

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

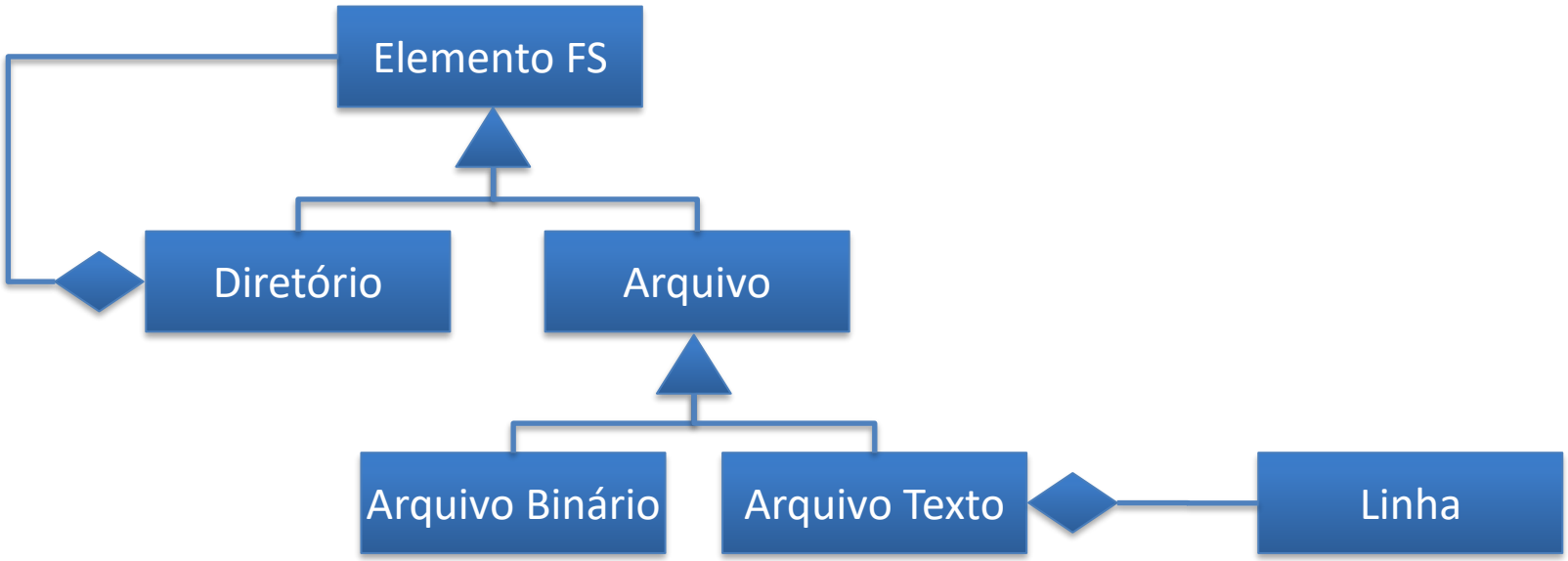


Histórico dos sistemas de controle de versões (VCS)

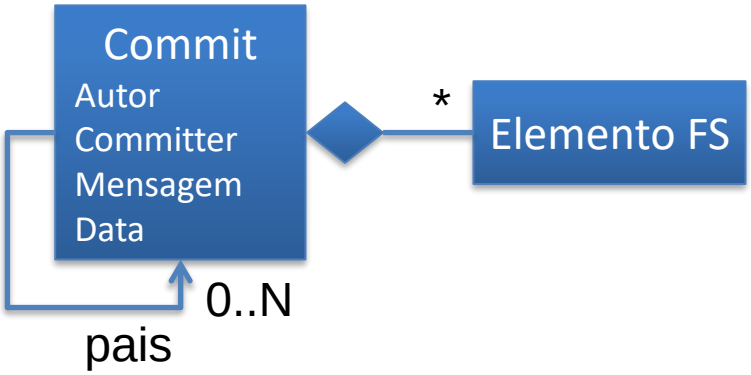
- Anos 2000 – Sistemas peer-to-peer
  - Git (2005)
  - Mercurial (2005)

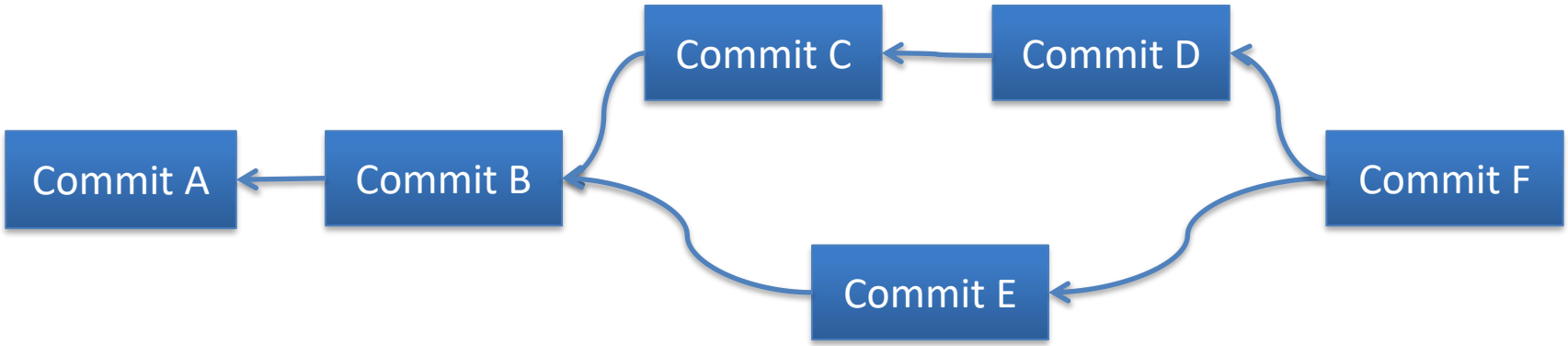


O que é versionado?



Como é versionado?

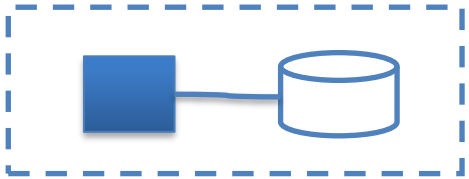




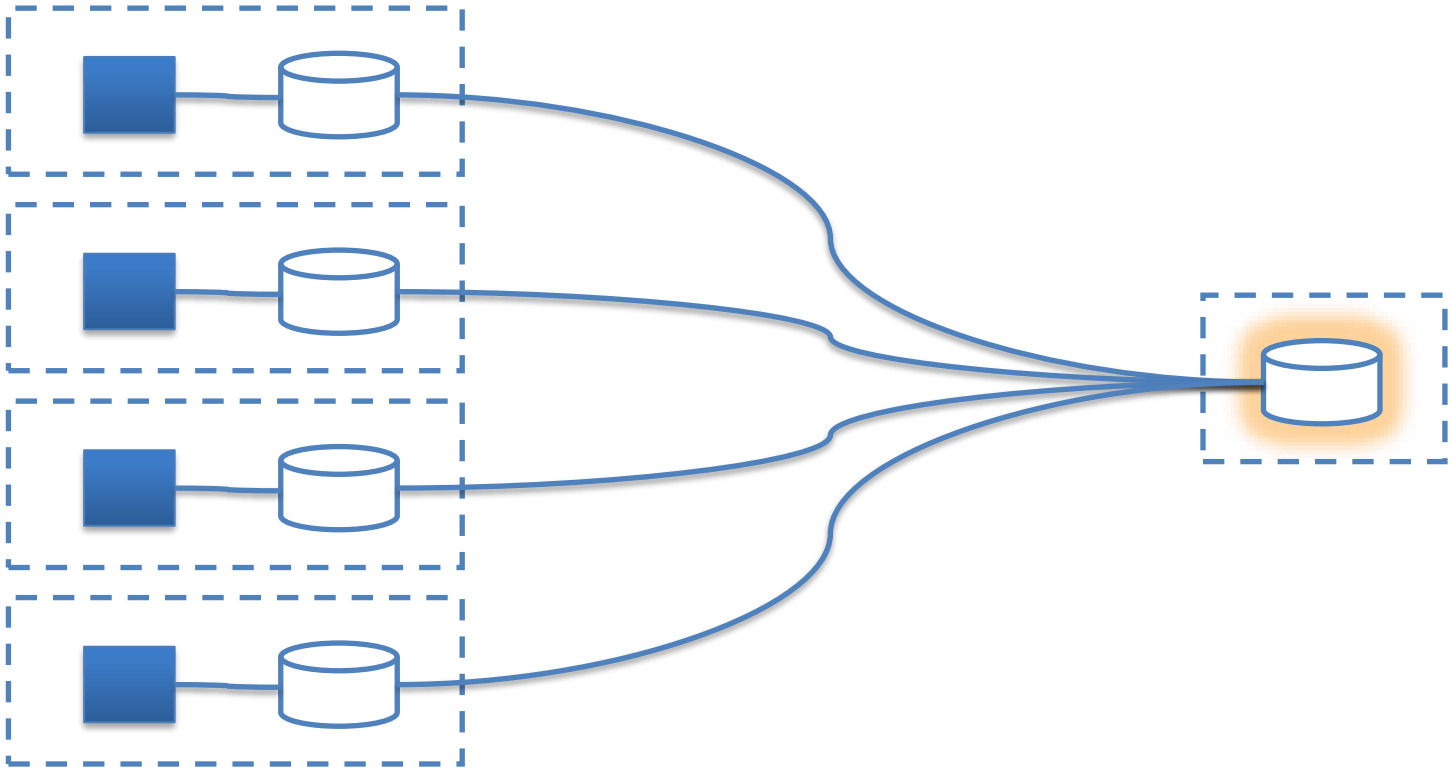
Formas de adoção

- 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

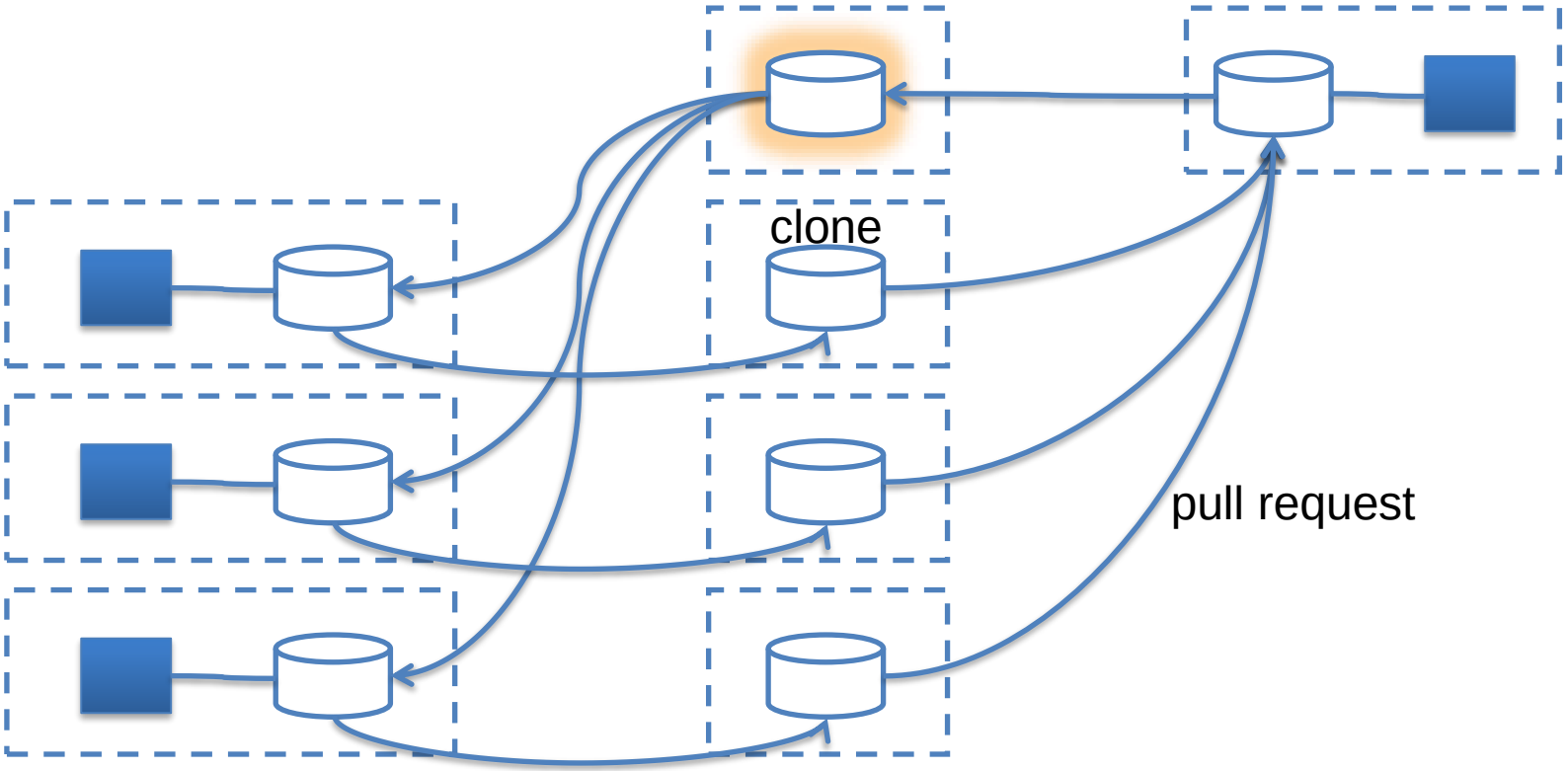
Individual



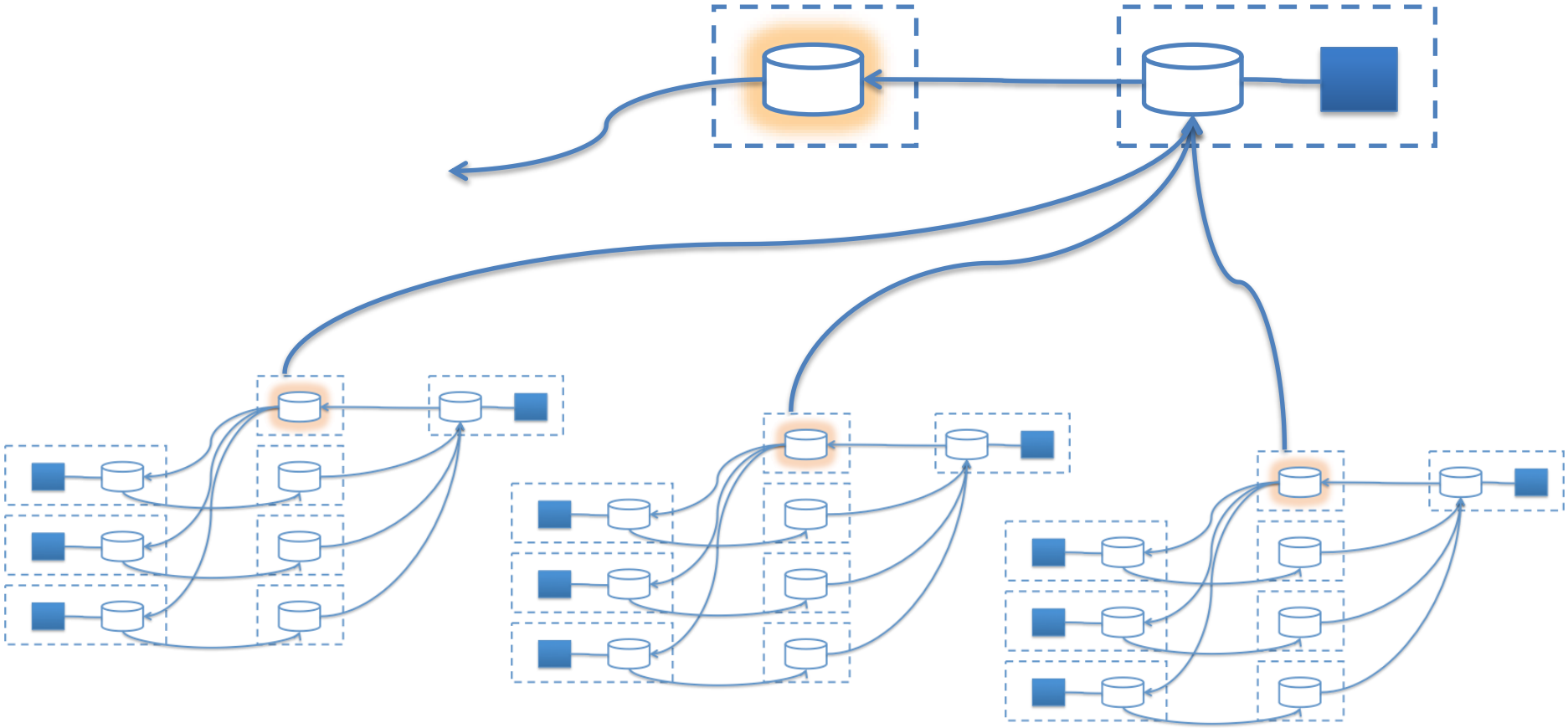
Cliente-servidor



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
  - Inspecionando 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

- Existem 3 níveis: Local, System, Global
  - A resolução é sempre o mais especifico

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 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

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.

Agradecimentos

- Gostaria de agradecer ao Prof. Leonardo Murta pelo material fornecido para esta apresentação

Uso de Git no apoio à pesquisa

Instituto de Computação

Troy Costa Kohwalter

tkohwalter@ic.uff.br