

Sistemas de Controle de Versão

Juliano F. Ravasi

Setembro / 2008
<http://juliano.info/>

Conteúdo

- Parte 1:
Controle de Versões
- Parte 2:
Trabalhando com Subversion
- Parte 3:
Trabalhando com Mercurial



Controle de Versões



O que é controle de versões?

- Gerenciamento de múltiplas revisões
 - Documentos, projetos, software, etc.
 - Histórico de alterações sofridas
 - Permitir consultar revisões anteriores
 - Permitir comparações entre revisões
 - Permitir trabalho cooperativo



O que é controle de versões?

- Vários nomes:
 - *Revision control system (RCS)*
 - *Version control system (VCS)*
 - *Software Configuration Management (SCM)*
 - *Source Code Management*
 - *Source Control*
- Controle de versão não é exclusivo para gerenciamento de software.



Por que controle de versões?

- Software é algo caro de ser produzido
 - Consome muito tempo
 - Exige cooperação, organização, disciplina
 - É importante armazenar tudo que é feito



Onde é utilizado?

- Sistemas de arquivos
- Suítes de escritório
- Ambientes colaborativos
- Gerenciamento de *software*

Importante para qualquer desenvolvedor ou empresa de desenvolvimento de *software*.



Motivação

Onde é utilizado?

Trabalho em equipe

O sistema ~~de controle de versão~~ também ~~é~~ ~~são~~ ~~bastante~~ ~~útil~~ ~~convenientes~~ quando ~~se trabalha com vários~~ ~~diversos~~ ~~desenvolvedores~~ ~~trabalham sobre o mesmo projeto simultaneamente~~ ~~de modo simultâneo~~, resolvendo ~~de modo muito~~ ~~satisfatório~~ ~~eventuais~~ conflitos ~~de versões~~ entre as alterações. A maioria dos sistemas ~~possuem~~ ~~possui~~ diversos recursos como ~~mesclagem e divisão de ramo de desenvolvimento~~ ~~ramificação e mesclagem de histórico~~ para auxiliar nessas tarefas.

Para que seja possível o trabalho em equipe, o sistema de controle de versão pode possuir um mini sistema de **controle de usuários** embutido ou pode utilizar algum outro sistema de autenticação separado. Assim, é possível identificar cada usuário, que geralmente fica protegido por uma senha pessoal, ou alguma senha criada pelo administrador de sistemas.

No CVS, por exemplo, é possível escolher o método de autenticação a ser usado, dentre várias opções. No caso do SVN, por exemplo, se ele estiver rodando via Apache, o controle de usuários poderá ser feito pela autenticação padrão do Apache. Embora menos comum, é possível também configurar o SVN para utilizar o usuário do sistema, como os usuários do Linux ou do Windows.

Mesclagens Otimistas vs Edições Exclusivas

Formatado: Fonte: Não Negrito



Motivação

Onde é utilizado?

Sistema de controle de versão - Wikipédia, a enciclopédia livre - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

W http://pt.wikipedia.org/w/index.php?title=Sistema_de_controle_de_versão&diff=113535 W Wikipédia (pt)

W Sistema de controle de versão - ...

artigo discussão editar história mover desinteressar-se

Sistema de controle de versão

Origem: Wikipédia, a enciclopédia livre.
Diferença entre revisões

Revisão de 20h34min de 9 de Junho de 2008 (editar)
200.198.201.78 (discussão)
(→ *Funcionamento básico*)
← Ver a alteração anterior

Revisão actual (00h21min de 5 de Julho de 2008)
(editar) (desfazer)
Juliano R (discussão | contribs)
m (→ *Trabalho em equipe: melhorando a gramática da frase*)
[Marcar como verificado]

Linha 65:

===Trabalho em equipe===

O sistema também é bastante útil quando se trabalha com vários [[programador|desenvolvedores]] de modo simultâneo, resolvendo de modo muito satisfatório conflitos de versões. A maioria dos sistemas possuem diversos recursos como "mesclagem" e "divisão de ramo" do desenvolvimento para auxiliar nessas tarefas.

Para que seja possível o trabalho em equipe, o sistema de controle de versão pode possuir um mini sistema de "controle de usuários" embutido ou pode utilizar algum outro sistema de autenticação separado. Assim, é possível identificar cada usuário, que geralmente fica protegido por uma senha pessoal, ou alguma senha criada pelo [[sysop|administrador de sistemas]].

Linha 65:

===Trabalho em equipe===

Sistemas de controle de versão também são convenientes quando diversos [[programador|desenvolvedores]] trabalham sobre o mesmo projeto simultaneamente, resolvendo eventuais conflitos entre as alterações. A maioria dos sistemas possui diversos recursos como "ramificação" e "mesclagem" de histórico para auxiliar nessas tarefas.

Para que seja possível o trabalho em equipe, o sistema de controle de versão pode possuir um mini sistema de "controle de usuários" embutido ou pode utilizar algum outro sistema de autenticação separado. Assim, é possível identificar cada usuário, que geralmente fica protegido por uma senha pessoal, ou alguma senha criada pelo [[sysop|administrador de sistemas]].

javascript:dismissNotice();

WIKIPÉDIA
A enciclopédia livre

navegação

- Página principal
- Os melhores artigos
- Eventos atuais
- Página aleatória
- Portais

colaboração

- Portal comunitário
- Mudanças recentes
- Ajuda
- Donativos

busca

Ir Pesquisa

ferramentas

- Artigos afluentes
- Novidades relacionadas
- Carregar arquivo
- Páginas especiais

CC BY NC SA

Registro de revisões

- Toda alteração realizada é registrada
 - Quem, quando, o que e por quê?
 - Nada é perdido para sempre
 - Descartar código ruim sem remorso
- Rápido acesso a revisões anteriores
 - Determinar introdução de defeitos
 - Manutenção de código legado



Auditoria

- Comparação entre versões do projeto, mostrando diferenças linha-a-linha
- Apontar desenvolvedores responsáveis por cada trecho de código do projeto
- Automação de testes de estabilidade



Ramificações

- Múltiplas linhas de desenvolvimento dentro do mesmo projeto
- Permite divergência e reconvergência do desenvolvimento



Trabalho cooperativo

- Vários desenvolvedores trabalhando sobre o mesmo projeto
- Mescla das alterações dos diversos desenvolvedores ou ramificações



Segurança

- Autenticação criptográfica de histórico
- Controle de acesso sobre o repositório
- Cópias de segurança (*backup*)



Modelos

- Centralizado (cliente-servidor)
 - Um repositório central de revisões
 - Desenvolvedores obtém cópias de trabalho do repositório central
- Distribuído
 - Cada desenvolvedor tem seu repositório
 - Desenvolvedores copiam repositórios e alterações de outros desenvolvedores



Centralizado (cliente-servidor)

- Vantagens:
 - Controle de acesso
 - Cópia de segurança
 - Controle de qualidade
- Desvantagens:
 - Dependência do repositório central
 - Ponto único de falha



- Vantagens:
 - Permite submissões particulares, *offline*
 - Melhor suporte a ramificação e mescla
 - Independência da rede, mais rápido
- Desvantagens:
 - Estimula o isolamento de desenvolvedores
 - Questões de privacidade e segurança



Centralizado vs. distribuído

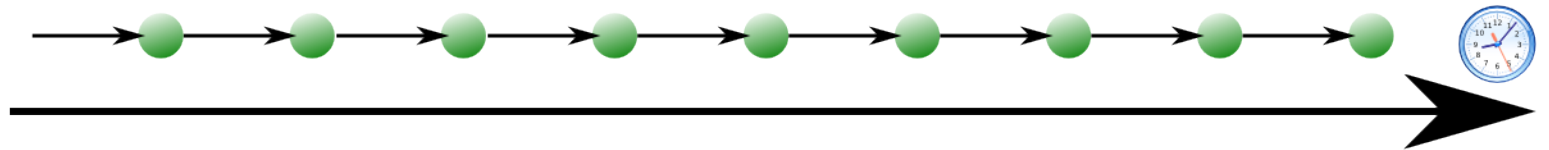
- Assunto “quente” nos últimos anos
 - Opiniões inflamadas, guerra de egos
- Há sobreposição entre os modelos
 - Ferramentas distribuídas podem ser usadas no modelo centralizado, quando necessário
- Ainda não existe a ferramenta perfeita
 - Cada modelo de desenvolvimento exige um modelo de controle de versões



Conceitos

Repositório

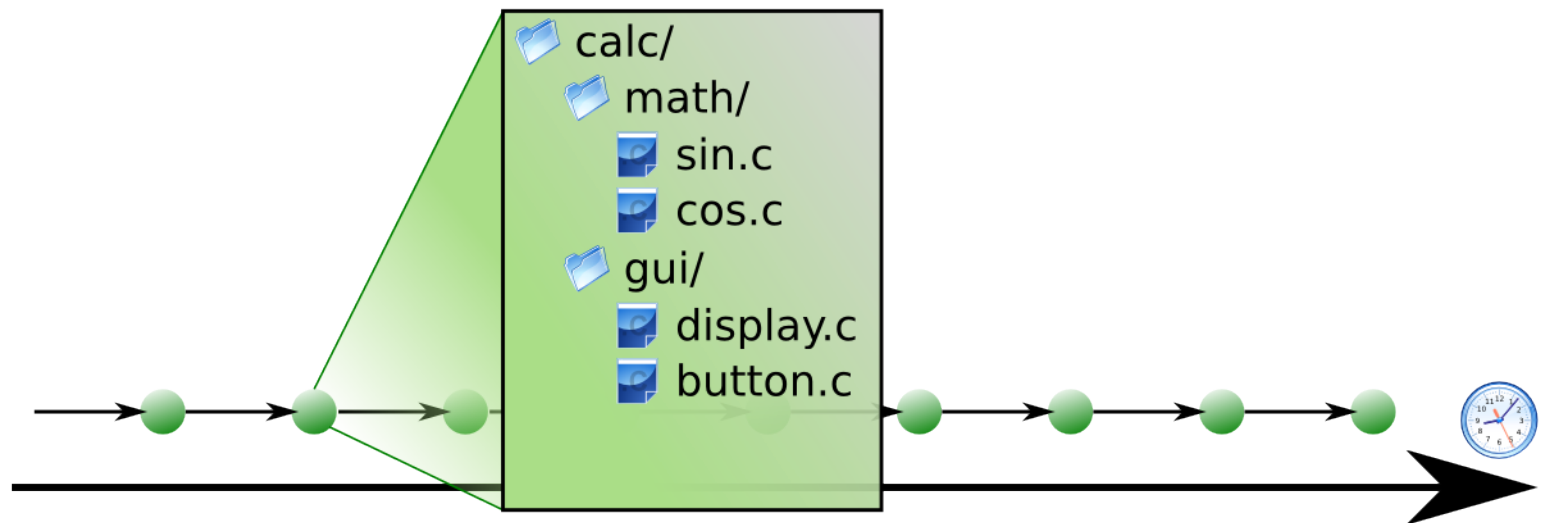
- Núcleo do controle de versões
- Possui uma “linha do tempo” embutida
 - Coletânea de revisões do projeto



Conceitos

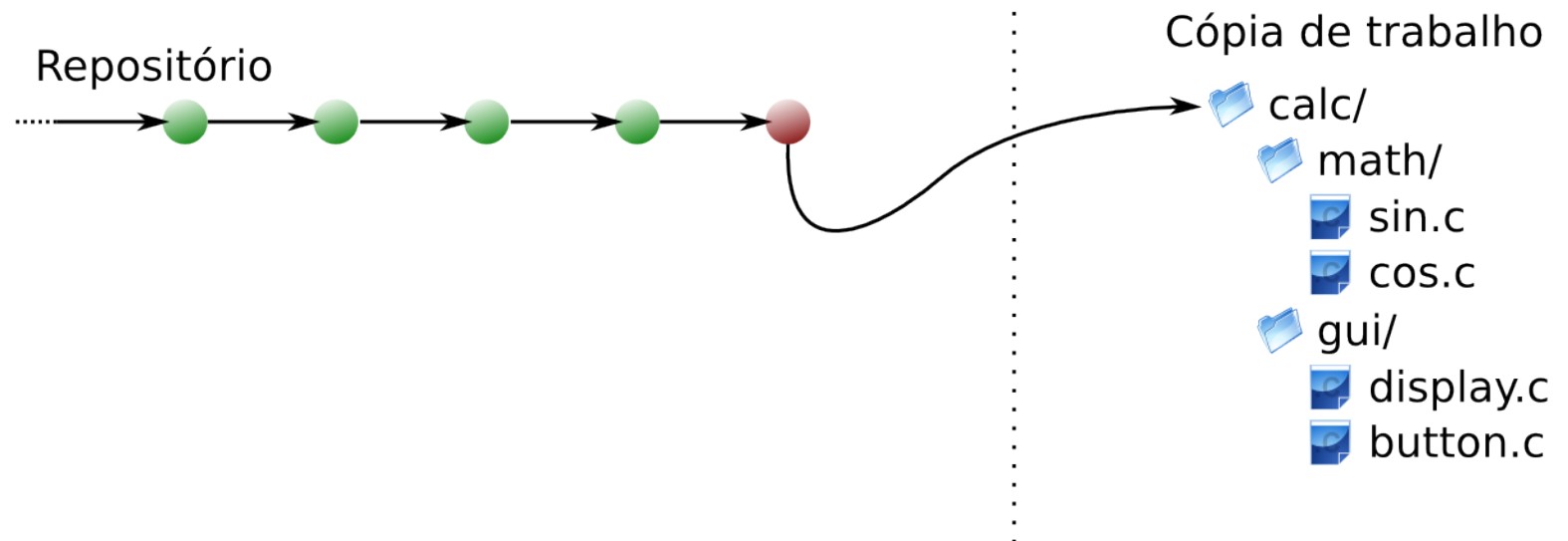
Repositório

- Revisão
 - Estado em um determinado instante
 - Imutável após criada



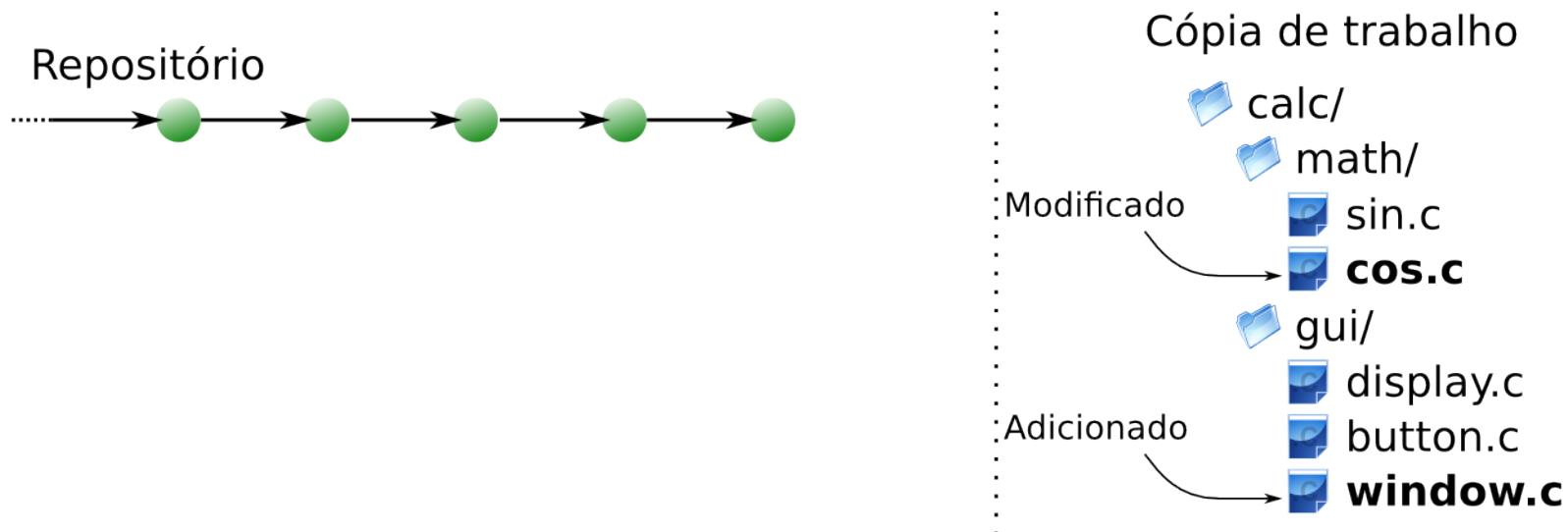
Cópia de trabalho

- Cópia do repositório em certa revisão
 - Geralmente a mais recente
 - *Checkout* (obtenção de uma cópia)



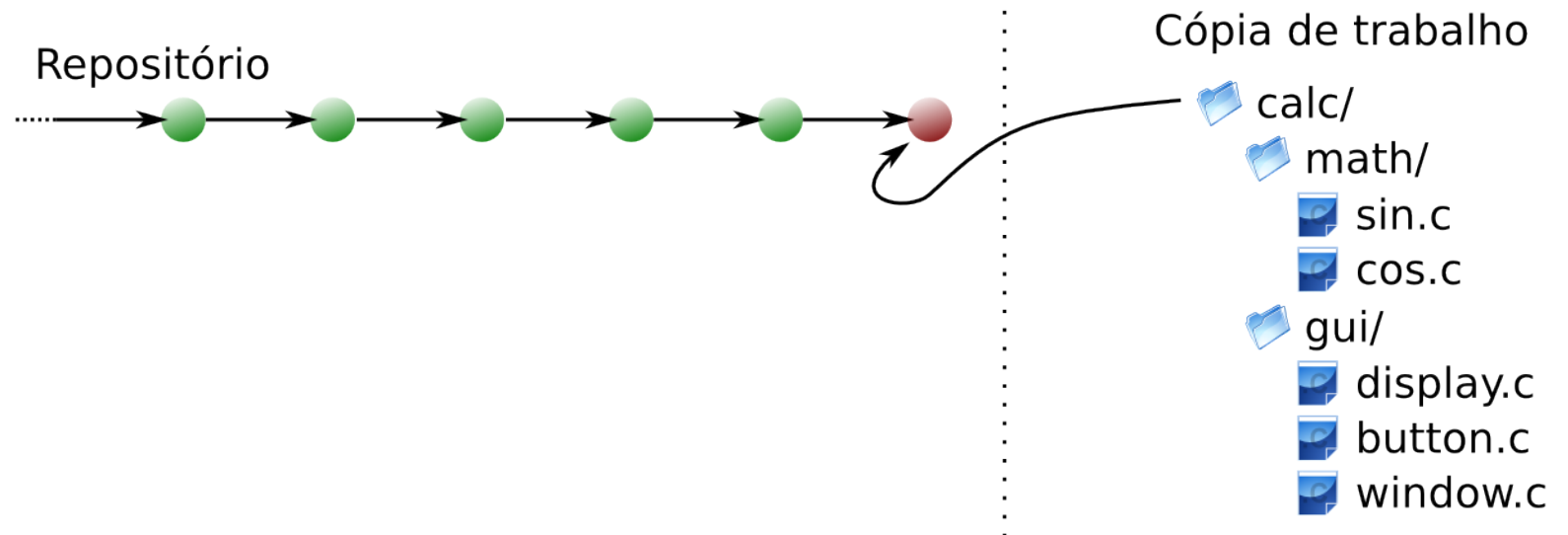
Cópia de trabalho

- Onde ocorre o desenvolvimento
 - O sistema reconhece as alterações feitas
 - Algumas operações devem ser explícitas
 - Adição, remoção, moção e cópia de arquivos



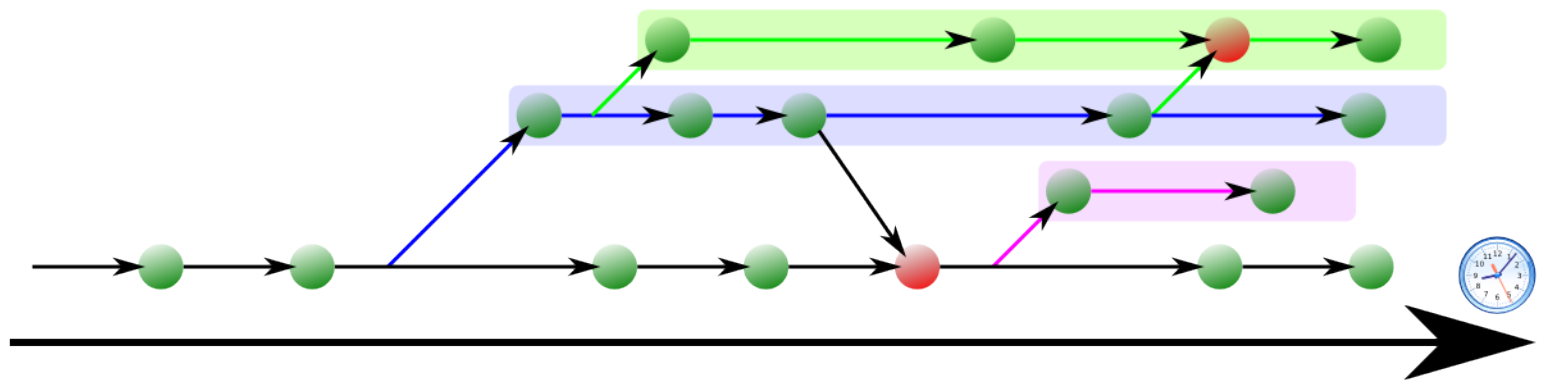
Cópia de trabalho

- Submissão (*commit*)
 - Alterações são registradas em uma nova revisão do repositório



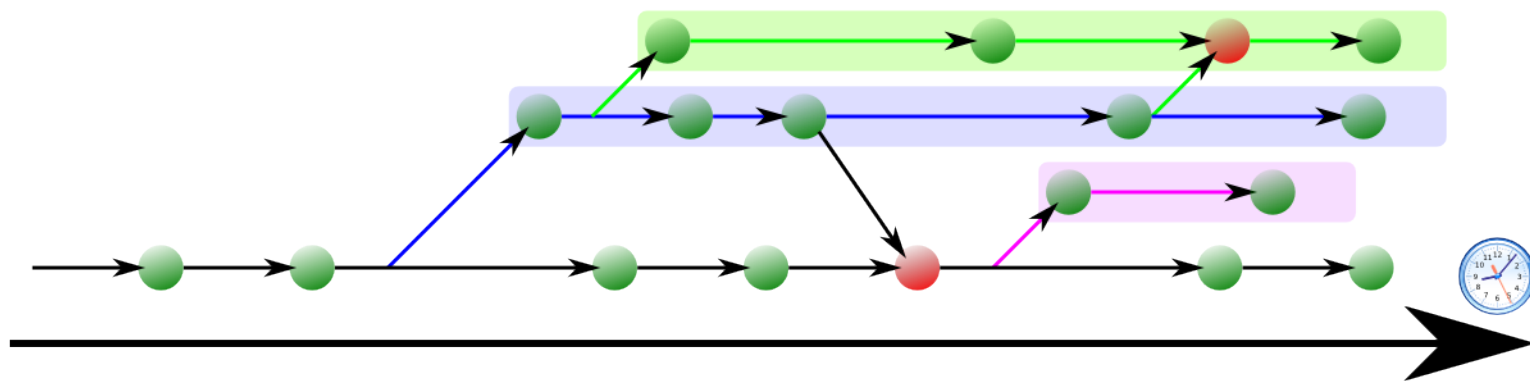
Ramificações (branches)

- Linhas alternativas de desenvolvimento
 - Explícitas
 - Implícitas



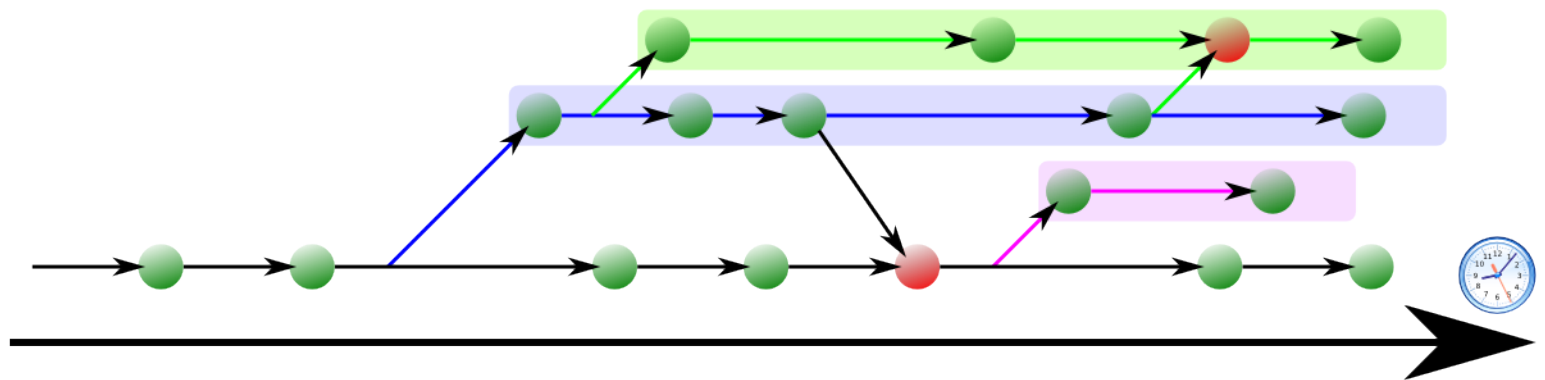
Ramificações (branches)

- Explícitas
 - Manutenção de versões legadas
 - Implementação de novos recursos
 - Experiências no código do projeto



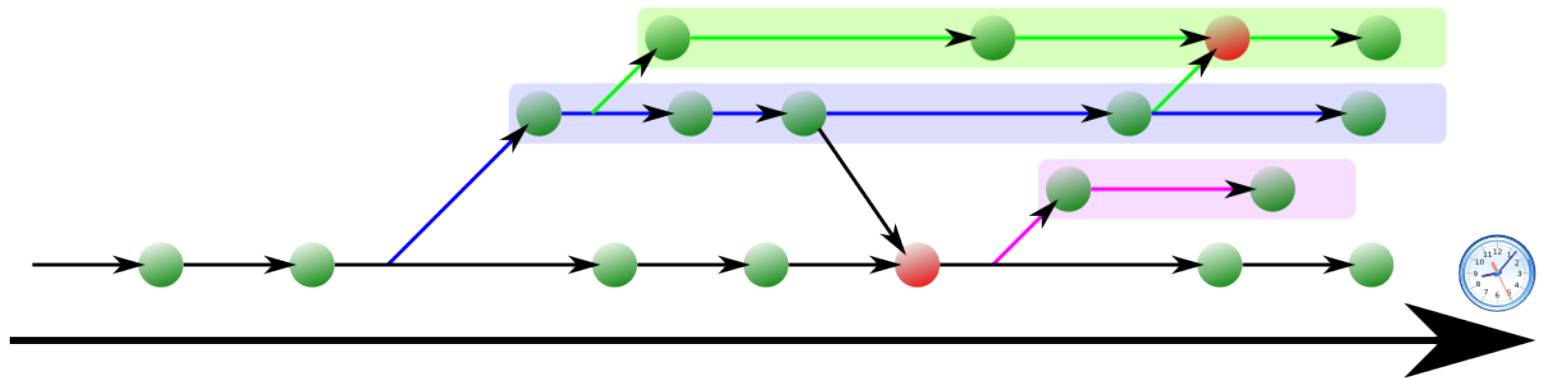
Ramificações (branches)

- Implícitas
 - Cópia de trabalho
 - Múltiplos desenvolvedores



Mescla (merge)

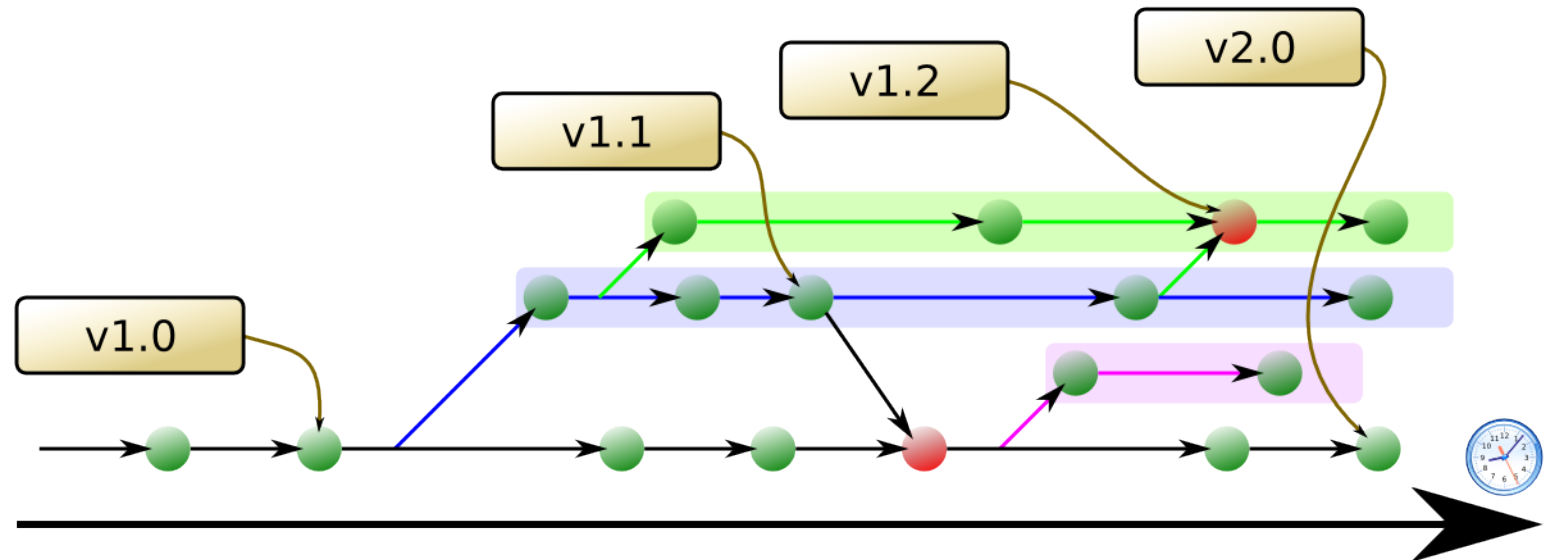
- Reintegração de ramificações
 - Em grande parte é automatizado
 - Conflitos podem ocorrer
 - O desenvolvedor pode precisar interagir



Conceitos

Rótulos (tags)

- Nomes atribuídos a revisões



Gerenciamento de software

Fluxos de trabalho

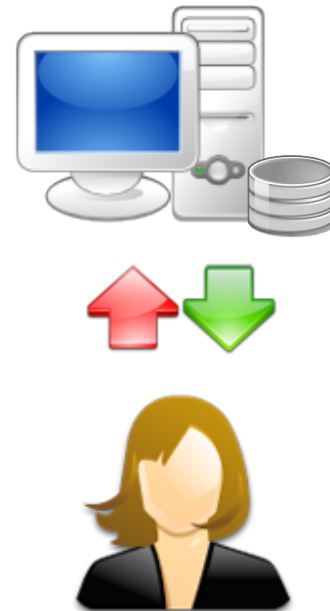
- Solitário
- Centralizado
 - Repositório local
 - Repositório remoto
- Distribuído
 - Parceiro
 - Equipe
 - Hierárquico



Gerenciamento de software

Fluxos de trabalho

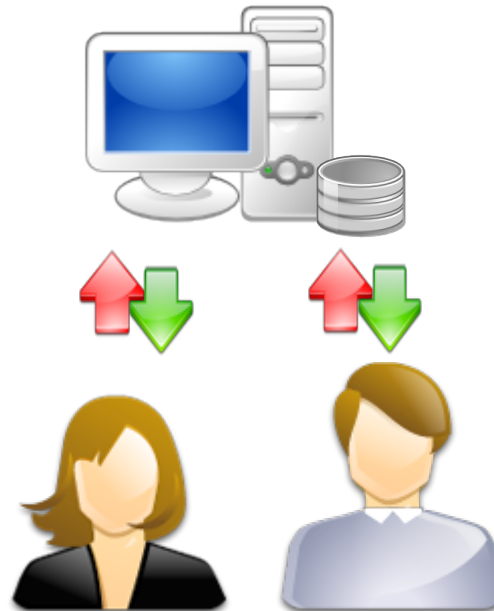
- Solitário



Gerenciamento de software

Fluxos de trabalho

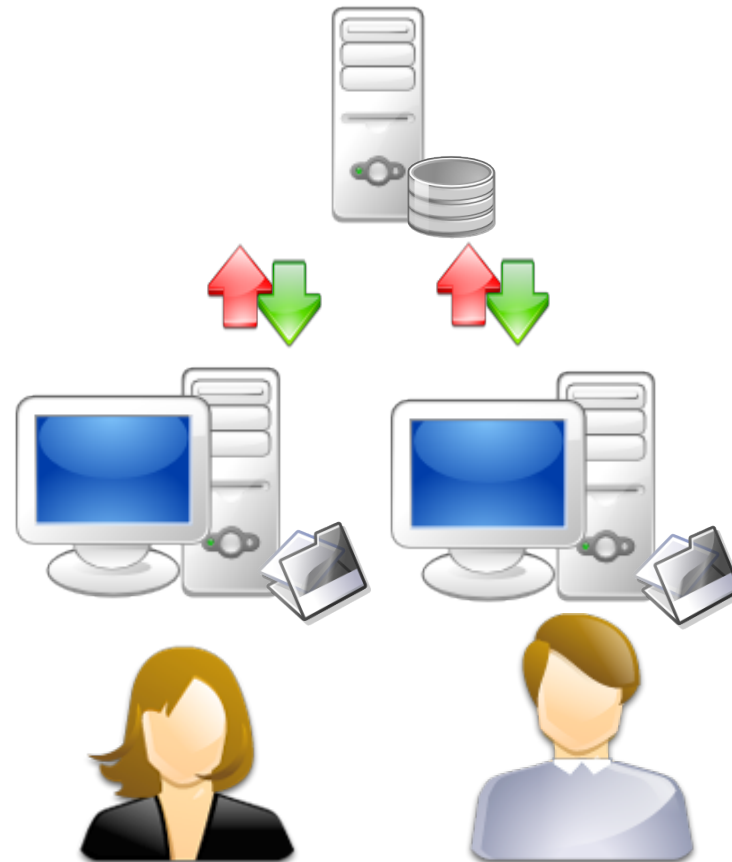
- Centralizado, repositório local



Gerenciamento de software

Fluxos de trabalho

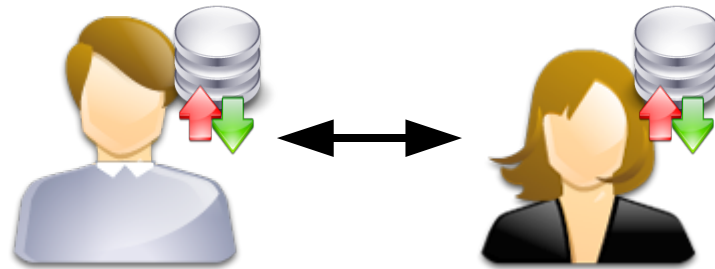
- Centralizado, repositório remoto



Gerenciamento de software

Fluxos de trabalho

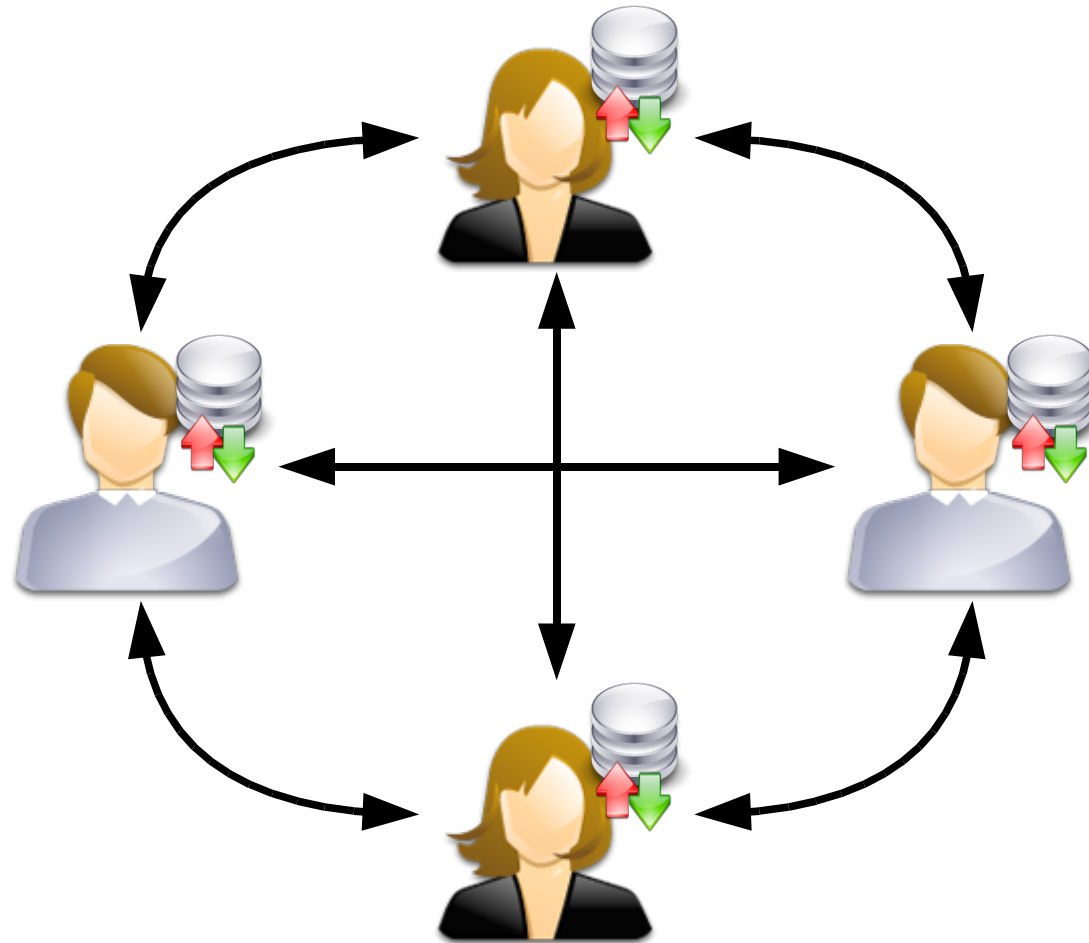
- Distribuído, parceiro



Gerenciamento de software

Fluxos de trabalho

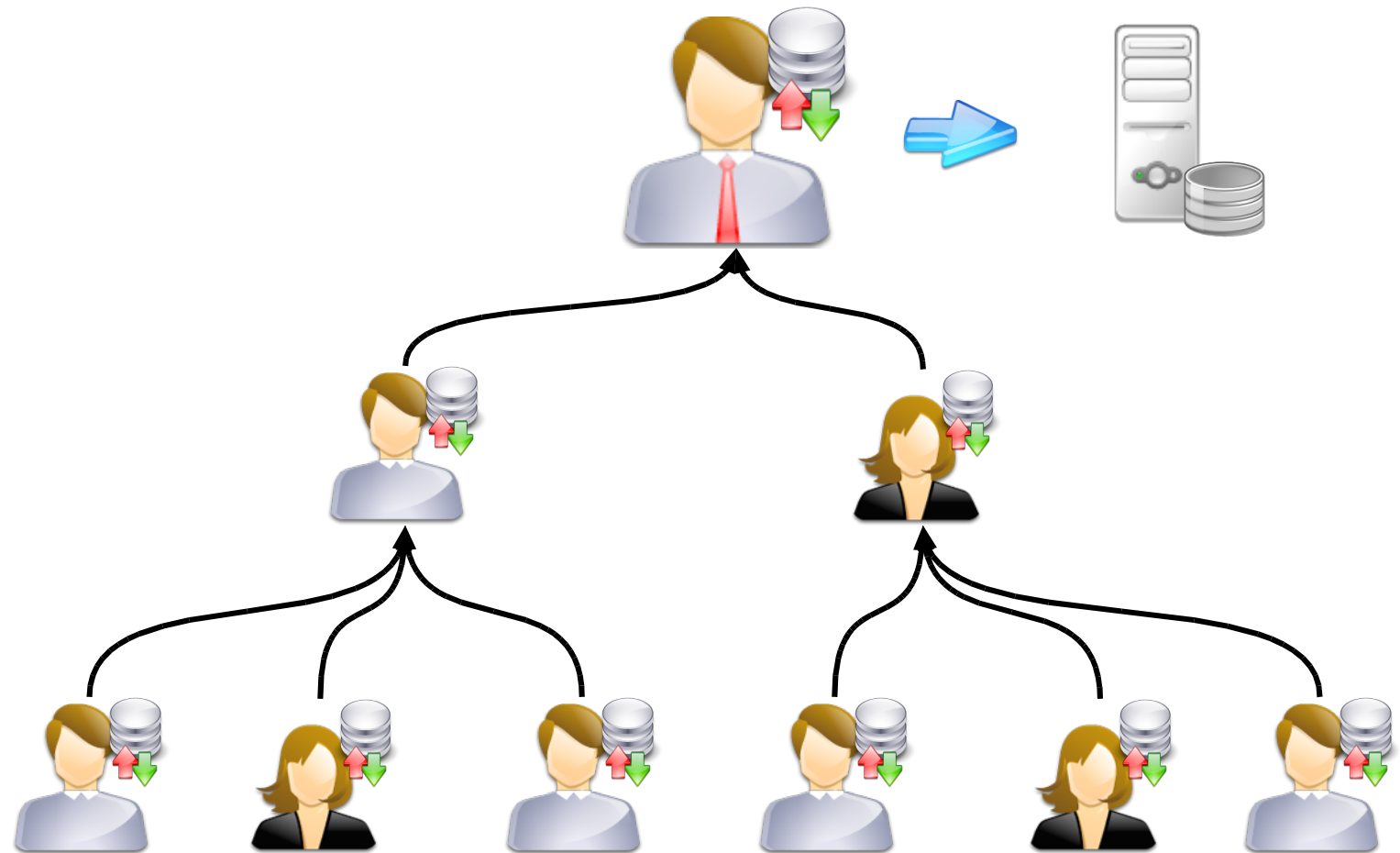
- Distribuído, equipe



Gerenciamento de software

Fluxos de trabalho

- Distribuído, hierárquico



Gerenciamento de software

Sistemas de controle de versões

Vault SourceAnyware Aldon FileHamster
MOG darcs Git DesignSync Evolution
ArX Surround AllFusion Fossil QVCS
FtpVC Serena SVK TeamWare PureCM
PlasticSCM Store codeBeamer Team Coherence
CVS Stellation MKS SourceHaven AVS OpenCVS
ClearCase Perforce SourceSafe CVSNT LibreSource
Vesta Code Co-op DCVS Subversion AlienBrain
AccuRev GNU arch Mercurial StarTeam Monotone
CodeVille RCS BitKeeper Bazaar
Aegis Team Foundation Server Synergy



Gerenciamento de software

Sistemas de controle de versões

	Centralizado	Distribuído
Aberto	CVS OpenCVS Subversion CVSNT Stellation Vesta	Aegis ArX arch Bazaar CodeVille darcs DCVS Fossil Git LibreSource Mercurial Monotone SVK
Proprietário	AccuRev Aldon AlienBrain AllFusion AVS ClearCase codeBeamer DesignSync Evolution FileHamster FirePublish FtpVC MKS MOG Perforce PureCM QVCS Serena SourceAnyware SourceHaven SourceSafe StarTeam Store Surround Synergy Team Coherence Vault Team Foundation Server	BitKeeper Code Co-op TeamWare PlasticSCM



Sistemas de controle de versões

- Visão geral dos VCSs abertos:
 - Centralizados:
 - CVS <http://www.nongnu.org/cvs/>
 - ▶ Subversion <http://subversion.tigris.org/>
 - Distribuídos:
 - Git <http://git.or.cz/>
 - ▶ Mercurial <http://www.selenic.com/mercurial/>
 - Bazaar <http://bazaar-vcs.org/>
 - ...
- Mais:
 - <http://linuxmafia.com/faq/Apps/vcs.html>
 - http://en.wikipedia.org/wiki/List_of_revision_control_software



CVS

- Criado para substituir o RCS (1980s)
- Obsoleto, desenvolvimento estagnado
- Modelo centralizado
- Possui grande base de usuários
- Possui defeitos e limitações de projeto
- Escrito em C, monolítico



Subversion

- “Sucessor” do CVS
 - Projetado para contornar os seus defeitos
- Desenvolvido por CollabNet Inc.
- Modelo centralizado
- Robusto e maduro
- Ênfase em qualidade, larga escala
- Sem restrições a tipos de arquivos
- Escrito em C, modular



Subversion

- Projeto em camadas
 - Repositório
 - BorlandDB, fsfs
 - Acesso ao repositório
 - Local, WebDAV, svnservice
 - Cópia de trabalho
 - Biblioteca de cliente
 - *Bindings*: C, Python, Perl, Java, Ruby
 - Cliente
 - svn, TortoiseSVN, Subclipse, KDESvn, Trac, ...



Git

- Criado para gerenciar o kernel do Linux
 - Após controvérsia sobre o BitKeeper
 - Inspirado no Monotone
- Inicialmente por Linus Torvalds
- Modelo distribuído
- Ênfase em desenvolvimento não-linear
- Autenticação criptográfica do histórico
- Gerencia conteúdo, ao invés de arquivos
- Escrito em C, monolítico



Mercurial

- Criado simultaneamente ao Git
 - Inspirado no Monotone
- Desenvolvido por Matt Mackall
- Modelo distribuído
- Autenticação criptográfica do histórico
- Ênfase em uso por grandes projetos
- Escrito em Python, modular



Bazaar

- Desenvolvido por Canonical Inc.
- Modelo distribuído
- Ênfase em facilidade, flexibilidade
 - Suporte a múltiplos fluxos de trabalho
- Desempenho ruim
- Escrito em Python, modular



O que armazenar?

- Arquivos produzidos pelo desenvolvedor:
 - ✓ código fonte
 - ✓ *scripts* de automação
 - ✓ documentação escrita
 - ✓ figuras, imagens e ícones
 - possivelmente apenas os originais, usando ferramentas automáticas para converter entre formatos e tamanhos
 - ✓ “*makefiles*”
 - a menos que sejam criados por um processo automático (por ex: *./configure*)



O que não armazenar?

- Arquivos gerados automaticamente:
 - × código-objeto
 - × programas compilados
 - × documentação automática
- Arquivos com configurações locais:
 - × credenciais de acesso a banco-de-dados
- Arquivos criados acidentalmente:
 - × “*core dumps*”
 - × arquivos temporários



Trabalhando com Subversion



Trabalhando com Subversion

Características gerais

- Repositório vs. cópia de trabalho
 - Armazenados em lugares distintos
- Ramificações, rótulos
 - Implícito, parte da árvore do repositório
 - Criados através de cópias leves
- Características únicas:
 - Propriedades
 - Diretórios
 - WebDAV



Trabalhando com Subversion

Características gerais

- Recomendação
 - Usar versão 1.5 ou superior.
- Informações
 - Projeto: <http://subversion.tigris.org/>
 - Manual: <http://svnbook.red-bean.com/>
 - Clientes:
 - Windows: <http://tortoisesvn.tigris.org/>
 - Eclipse: <http://subclipse.tigris.org/>
 - KDE: <http://kdesvn.alwins-world.de/>



Trabalhando com Subversion

Repositório

- Sistema de arquivos
 - Baseado em ligações
 - Cópias leves, *copy-on-write*
 - Banco-de-dados transacional
 - Revisões numéricas, seriais
 - Revisão zero: repositório em branco
- Sugestões de organização:
 - Um repositório por projeto (incluindo subprojetos)



Trabalhando com Subversion

Repositório

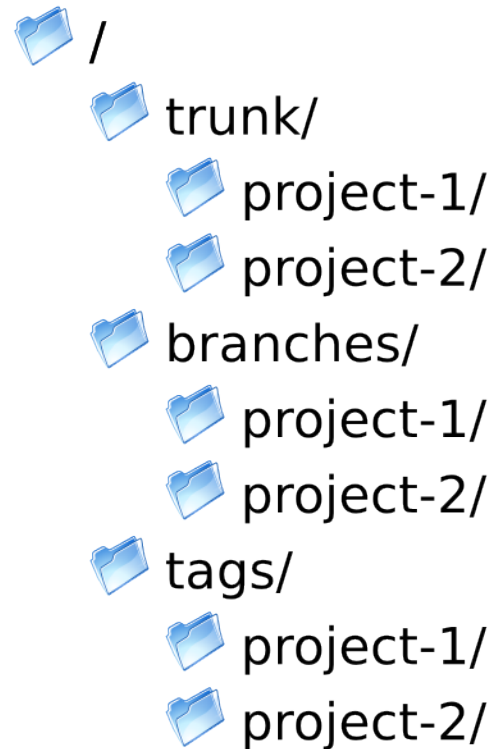
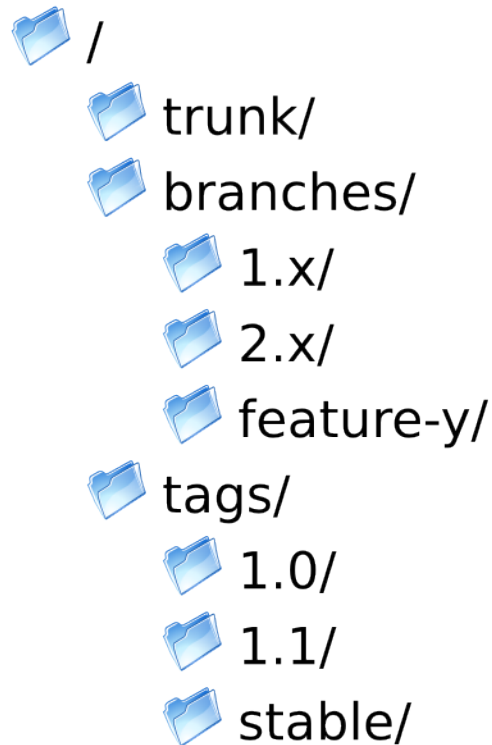
- Criação do repositório
 - `svnadmin create diretório`
 - Cria um novo repositório Subversion em branco (revisão zero) em '*diretório*'.
- Convenções de hierarquia
 - /
 - trunk/ ← tronco
 - branches/ ← ramificações
 - tags/ ← rótulos



Trabalhando com Subversion

Repositório

- Convenções de hierarquia



Trabalhando com Subversion

Repositório

- Criação da hierarquia inicial

- Método 1:

```
svn checkout file:///home/user/svn/repo
cd repo
svn mkdir trunk branches tags
svn commit -m "Directory hierarchy."
```

- Método 2 (bash):

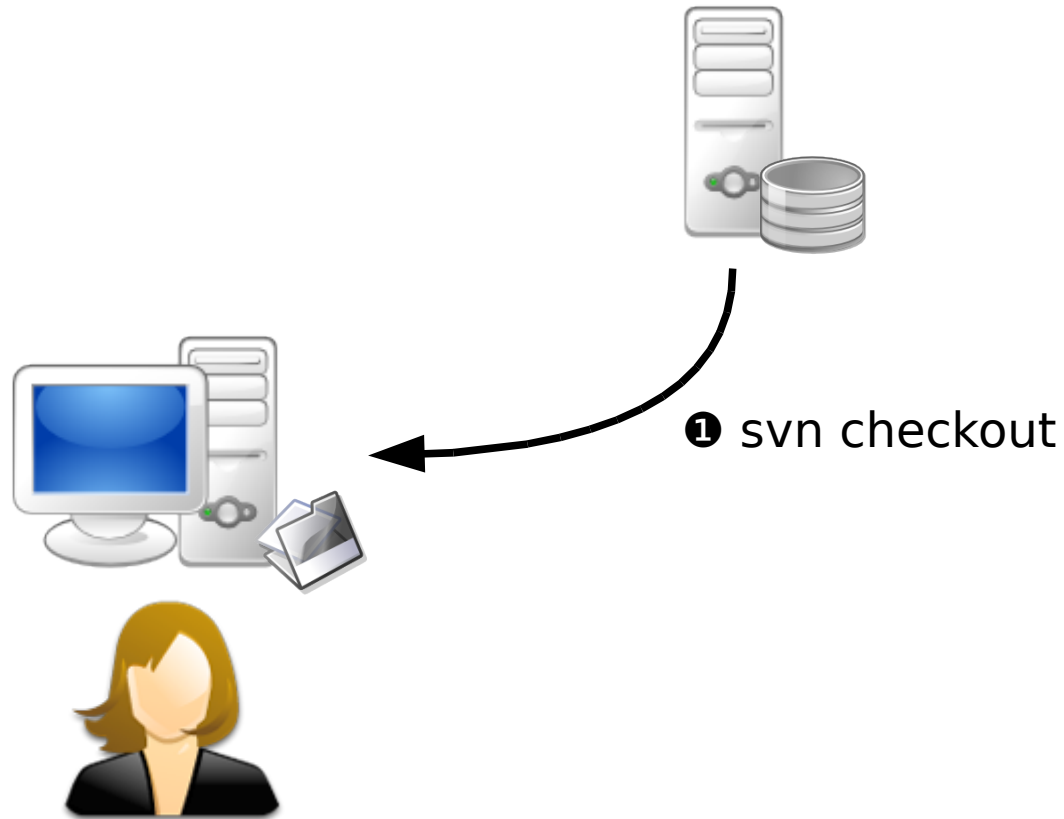
```
r=file:///home/user/svn/repo
svn mkdir $r/{trunk,branches,tags} -m "..."
```



Trabalhando com Subversion

Ciclo de trabalho

- 1. Obtenção (*checkout*)
 - svn checkout



Trabalhando com Subversion

Ciclo de trabalho

- 2. Desenvolvimento



② (edição)



Trabalhando com Subversion

Ciclo de trabalho

- 3. Comparação, reversão
 - svn status
 - svn diff
 - svn revert

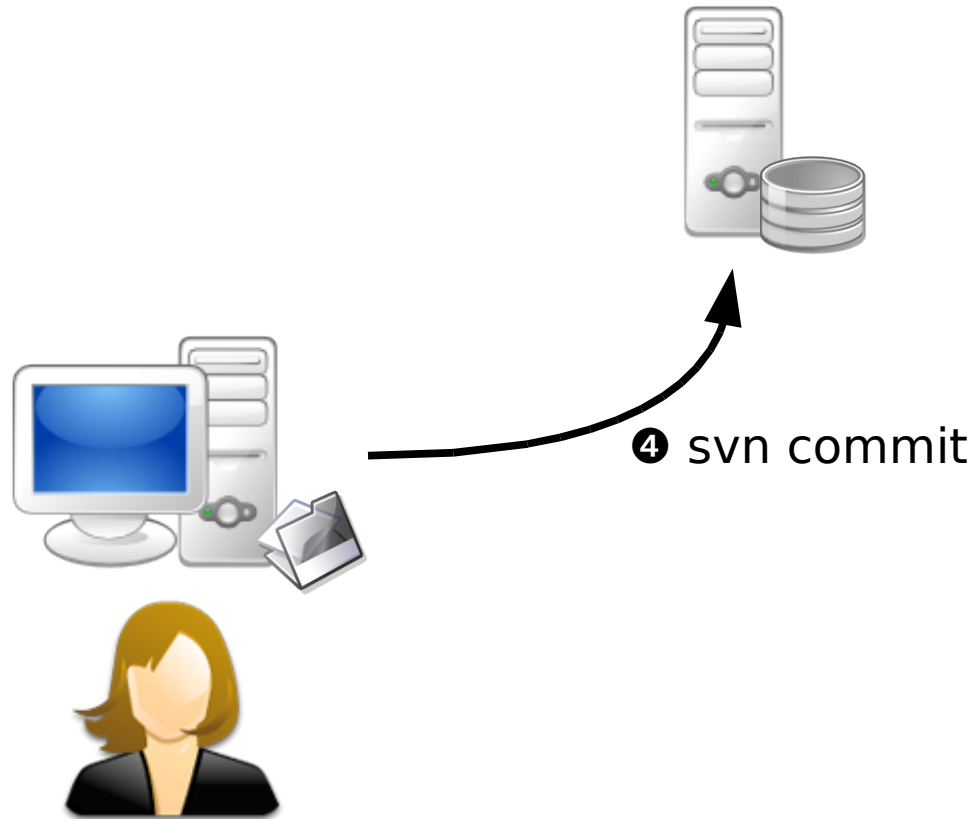


③ svn diff
svn revert

Trabalhando com Subversion

Ciclo de trabalho

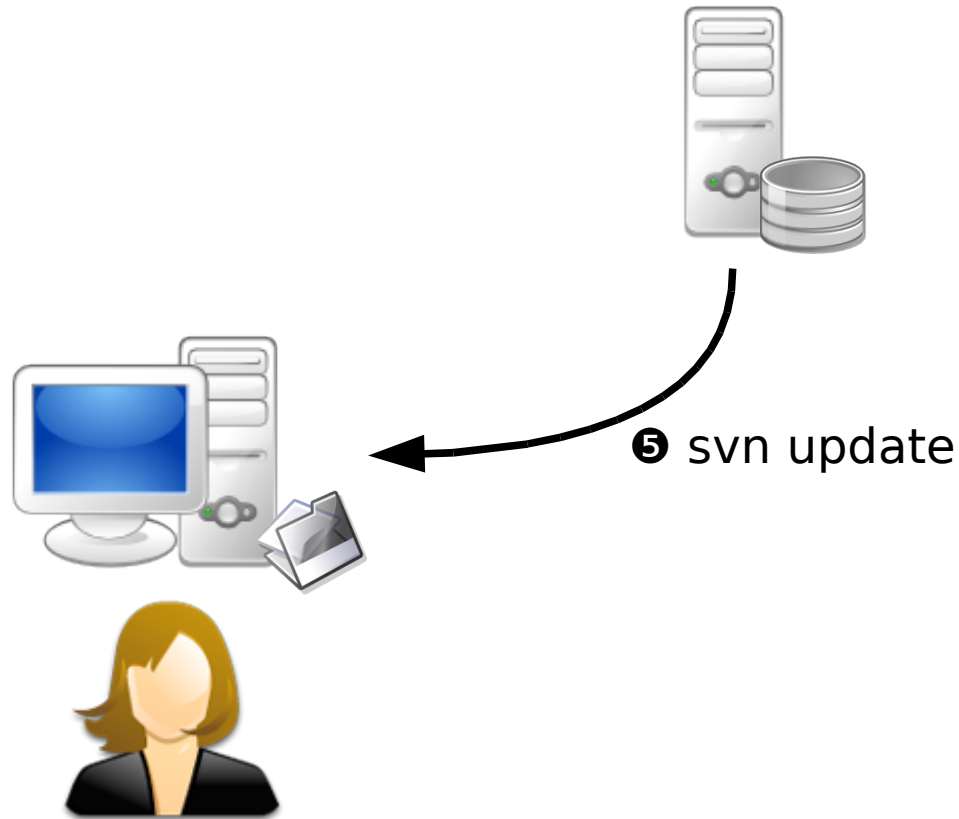
- 4. Submissão
 - svn commit



Trabalhando com Subversion

Ciclo de trabalho

- 5. Atualização
 - svn update



Trabalhando com Subversion

Comandos

- Ferramentas principais:
 - svn
 - Cliente padrão, cópia de trabalho.
 - svnadmin
 - Manutenção do repositório.
- Outras ferramentas:
 - svndumpfilter
 - svnlook
 - svnservice
 - svnsync



Trabalhando com Subversion

Comandos

- Acesso ao repositório
 - Local
 - *file:///home/user/svn/repo/...*
 - WebDAV sobre HTTP
 - *http://svn.example.com/svn/repo/...*
 - *https://svn.example.com/svn/repo/...*
 - svnservice
 - *svn://svn.example.com/svn/repo/...*
 - svnservice sobre SSH
 - *svn+ssh://svn.example.com/svn/repo/...*



Trabalhando com Subversion

Comandos

- Acesso ao repositório
 - Composição da URL

svn://svn.example.com/svn/project/subproject/trunk/

Endereço do
servidor Subversion

Endereço do
repositório

Caminho armazenado
no repositório



Trabalhando com Subversion

Comandos

- Importação
 - `svn import [diretório] URL`
 - Importa um projeto existente em um repositório.
- Obtenção
 - `svn checkout URL [diretório]`
 - Obtém uma cópia de trabalho.
 - `svn export URL [diretório]`
 - Obtém uma cópia limpa do projeto.
 - `svn update [diretório]`
 - Atualiza a cópia de trabalho com o repositório.



Trabalhando com Subversion

Comandos

- Edição
 - `svn add alvo`
 - Adiciona um arq./dir. ao controle de versão.
 - `svn delete alvo`
 - Remove um arq./dir. do controle de versão.
 - `svn mkdir diretório`
 - Cria um diretório no controle de versão.
 - `svn copy orig dest`
 - Cria uma cópia de um arquivo ou diretório.
 - `svn move orig dest`
 - Move ou renomeia um arquivo ou diretório.



Trabalhando com Subversion

Comandos

- Consulta e comparação
 - `svn status [alvo]`
 - Mostra arquivos alterados, adicionados, etc.
 - `svn diff [alvo]`
 - Mostra alterações realizadas em arquivos.
- Reversão
 - `svn revert [alvo]`
 - Reverte alterações realizadas em arquivos.



Trabalhando com Subversion

Comandos

- Submissão
 - `svn update`
 - Atualiza a cópia de trabalho antes de submeter, opcional.
 - Podem ocorrer conflitos ao mesclar as alterações do repositório com as suas.
 - `svn commit [alvo]`
 - Submete o estado atual da cópia de trabalho para o repositório, criando uma nova revisão.
 - Nesse processo, é necessário fornecer uma breve descrição das alterações realizadas.



Trabalhando com Subversion

Comandos

- Resolução de conflitos
 - `svn resolve --accept arg [alvo]`
 - Resolução simples, informa qual versão do alvo deverá ser preservada.
 - `svn resolved alvo`
 - Informa que o conflito foi manualmente resolvido, e destrava a cópia de trabalho.
 - `svn status`
 - `svn commit [alvo]`
 - Após resolver os conflitos, tentar submeter novamente.



Trabalhando com Subversion

Comandos

- Informação e auditoria
 - `svn info [alvo]`
 - Mostra informações diversas sobre a cópia de trabalho ou o alvo fornecido.
 - `svn log [alvo]`
 - Mostra registro de alterações de um arquivo, diretório ou do projeto.
 - `svn blame arquivo`
 - Acusa os desenvolvedores que alteraram por último cada linha de um arquivo.



Trabalhando com Subversion

Consulta de histórico (svn log)

```
r3 | juliano | 2008-07-18 00:41:18 -0300 (Fri, 18 Jul 2008) | 8 lines  
Changed paths:
```

```
  M /trunk  
  A /trunk/CalculatorForm.cpp  
  A /trunk/CalculatorForm.h  
  A /trunk/CalculatorForm.ui  
  M /trunk/calculator.cpp  
  M /trunk/calculator.pro
```

User interface design.

Designed the user interface of the calculator, using Qt Designer.
No functions have been implemented yet. The calculator is
non-functional.

Added some patterns for generated files to svn:ignore.



Atualização (`svn update`)

- Para visitar revisões passadas
 - `svn update -r revisão`
 - `svn update -r {data}`
- Para retornar à última revisão
 - `svn update`
- Legenda
 - A: adicionado (Added)
 - D: apagado (Deleted)
 - U: atualizado (Updated)
 - C: conflito (Conflicted)
 - G: mesclado (merGed)
 - E: já existente (Existed)



Trabalhando com Subversion

Comparação (svn diff)

- Alterações na cópia de trabalho
 - `svn diff [alvo]`
- Comparar revisões arbitrárias
 - `svn diff -r x:y [alvo]`
 - Alterações entre revisões 'x' e 'y'.
 - `svn diff -c x [alvo]`
 - Alterações entre revisões 'x'-1 e 'x'.



Trabalhando com Subversion

Comparação (svn diff)

Index: CalculatorForm.h

```
-----  
--- CalculatorForm.h      (revision 6)  
+++ CalculatorForm.h      (revision 7)  
@@ -27,6 +27,7 @@
```

```
private slots:  
    void on_buttonClear_clicked();  
+   void on_buttonErase_clicked();  
    void on_button0_clicked();  
    void on_button1_clicked();  
    void on_button2_clicked();
```

Index: CalculatorForm.ui

```
-----  
--- CalculatorForm.ui     (revision 6)  
+++ CalculatorForm.ui     (revision 7)  
@@ -171,7 +171,7 @@  
    </widget>  
  </item>  
  <item row="1" column="3" >  
-   <widget class="QToolButton" name="buttonUnassigned3" >  
+   <widget class="QToolButton" name="buttonErase" >  
    <property name="sizePolicy" >  
      <sizepolicy vsizetype="Preferred" hsizetype="Preferred" >  
        <horstretch>0</horstretch>  
@@ -179,7 +179,7 @@  
    </sizepolicy>  
  </property>  
  <property name="text" >  
-   <string>...</string>  
+   <string><</string>  
  </property>  
  </widget>  
</item>
```



Consulta de estado (svn status)

- Legenda

- A: adicionado (Added)
- C: em conflito (Conflicted)
- D: apagado (Deleted)
- I: ignorado (Ignored)
- M: modificado (Modified)
- R: substituído (Replaced)
- X: item externo (eXternal)
- ?: item desconhecido, não controlado
- !: item controlado, porém ausente

- Mais detalhes

- svn help status



Trabalhando com Subversion

Ramificações

- Comandos
 - `svn copy repo/trunk repo/branches/branch`
 - Cria uma nova ramificação.
 - `svn switch repo/trunk`
 - `svn switch repo/branch/branch`
 - Alterna entre ramificações.



Mescla e reversão

- Mescla entre ramificações
 - `svn merge URL`
 - Mescla alterações de uma ramificação diferente na ramificação atual da cópia de trabalho.
 - `svn merge --reintegrate URL`
 - Reintegra alterações de uma ramificação ao tronco.
 - `svn merge -c rev URL`
 - Mescla apenas revisão 'rev'.



Trabalhando com Subversion

Mescla e reversão

- Reversão
 - `svn merge -c -rev URL`
 - Desfaz as alterações da revisão '*rev*'.
 - Pode ser usado na própria ramificação.
 - É uma mescla “ao contrário”.



Rótulos

- Caso especial de ramificação
 - São usados apenas como referência
 - Devem ser “somente-leitura” após criados.
- Comando
 - `svn copy repo/trunk repo/tags/tag`
 - Cria um novo rótulo.



Trabalhando com Subversion

Propriedades

- Comandos
 - `svn proplist [alvo]`
 - Lista propriedades.
 - `svn propget prop alvo`
 - Recupera conteúdo de uma propriedade.
 - `svn propset prop cont alvo`
 - Configura o conteúdo de uma propriedade.
 - `svn propedit prop alvo`
 - Altera uma propriedade (no editor externo).
 - `svn propdel prop alvo`
 - Apaga uma propriedade.



Trabalhando com Subversion

Propriedades

- Propriedades padrão
 - Diretórios e arquivos
 - `svn:eol-style`
 - `svn:executable`
 - `svn:externals`
 - `svn:ignore`
 - `svn:keywords`
 - `svn:mime-type`
 - `svn:needs-lock`
 - Revisões
 - `svn:author`
 - `svn:date`
 - `svn:log`



Trabalhando com Mercurial



Trabalhando com Mercurial

Características gerais

- Distribuído
 - Cópia de trabalho **contém** o repositório:
São armazenados juntos
- Ramificações, rótulos
 - Explícitos, possuem tratamento especial
 - Cada cópia de trabalho é uma ramificação



Trabalhando com Mercurial

Características gerais

- Informações
 - Projeto: <http://www.selenic.com/mercurial/>
 - Manual: <http://hgbook.red-bean.com/>
 - Clientes:
 - Windows: <http://tortoisehg.sourceforge.net/>
 - Eclipse:
<http://www.vectrace.com/mercurialeclipse/>



Trabalhando com Mercurial

Repositório

- Sistema de arquivos
 - Armazenamento eficiente, seguro e rápido
 - Revisões:
 - Numéricas, seriais
 - *Hash SHA-1 do changeset*
- Sugestões de organização:
 - Um repositório por subprojeto
 - Projeto: vários subprojetos agrupados



Trabalhando com Mercurial

Ciclo de trabalho

- 1. Início do projeto (*init*)
 - hg init



① hg init

Trabalhando com Mercurial

Ciclo de trabalho

- 2. Desenvolvimento, submissão local
 - hg status, diff, revert, commit



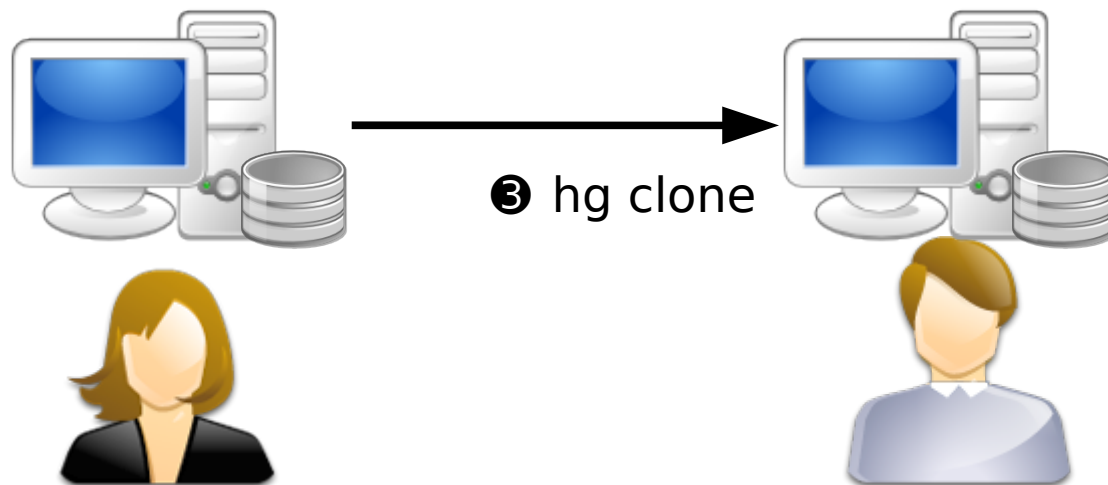
② hg status
hg diff
hg revert
hg commit



Trabalhando com Mercurial

Ciclo de trabalho

- 3. Ramificação
 - hg clone



Trabalhando com Mercurial

Ciclo de trabalho

- 4. Mais desenvolvimento



④ hg status
hg diff
hg revert
hg commit



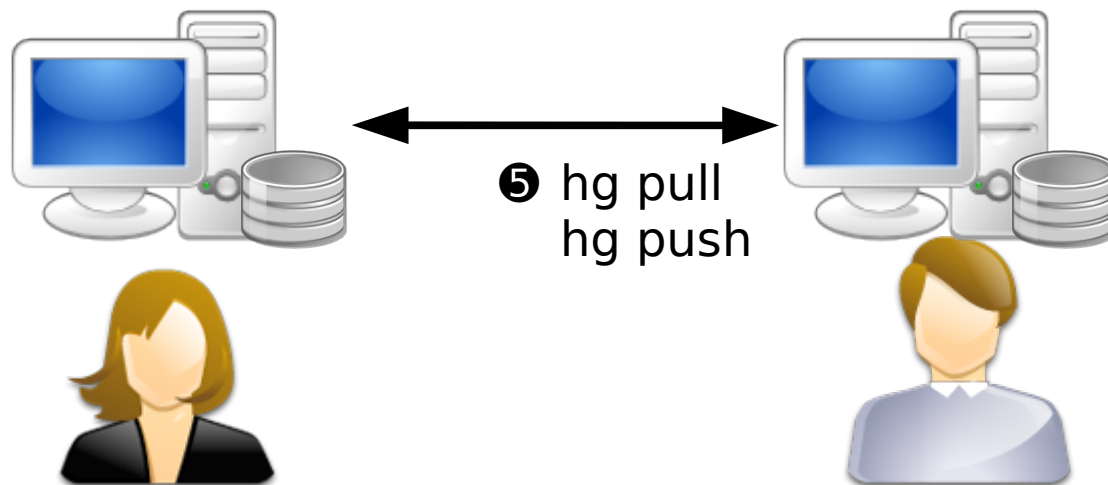
④ hg status
hg diff
hg revert
hg commit



Trabalhando com Mercurial

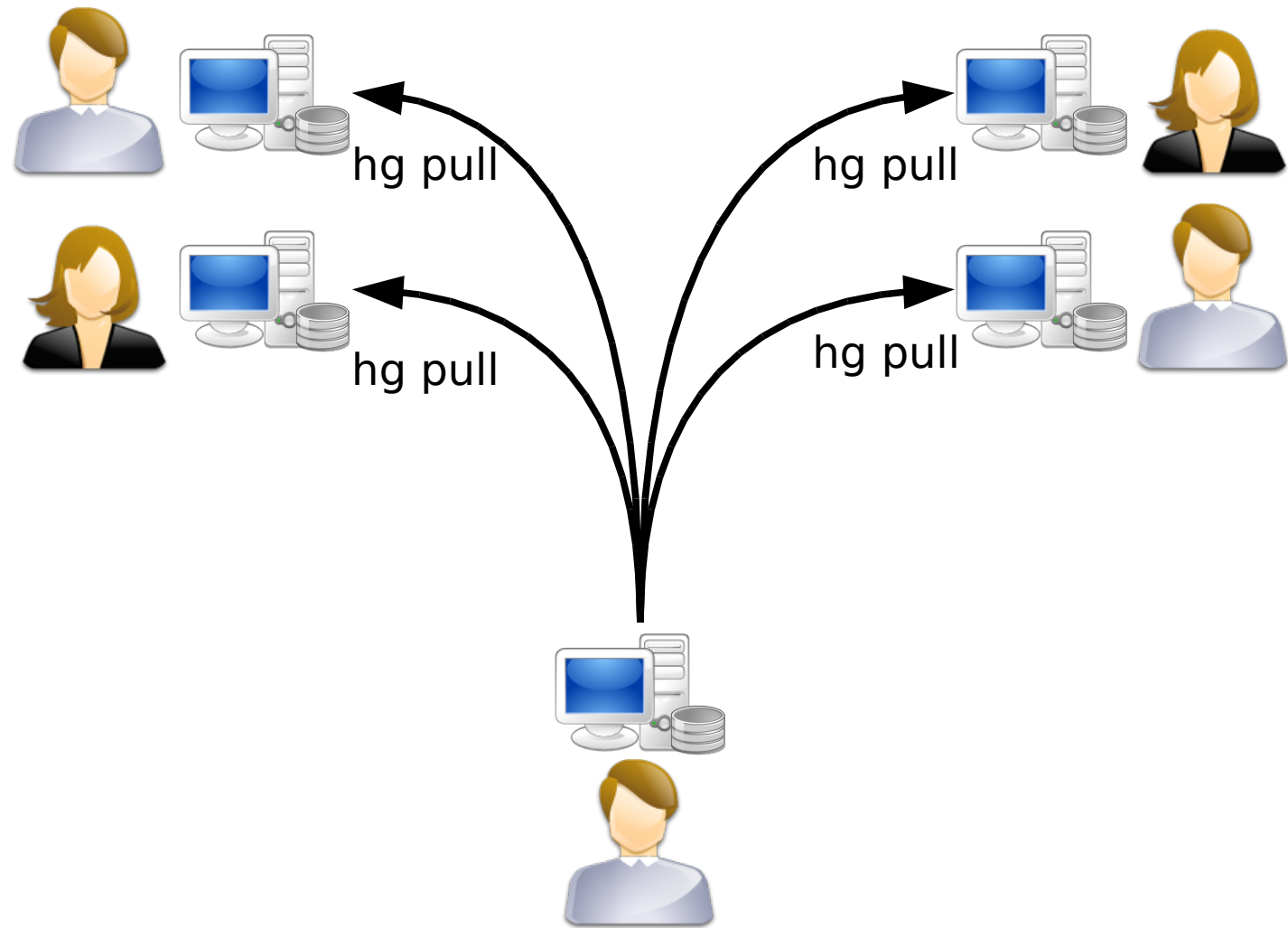
Ciclo de trabalho

- 5. Mescla
 - hg pull, push



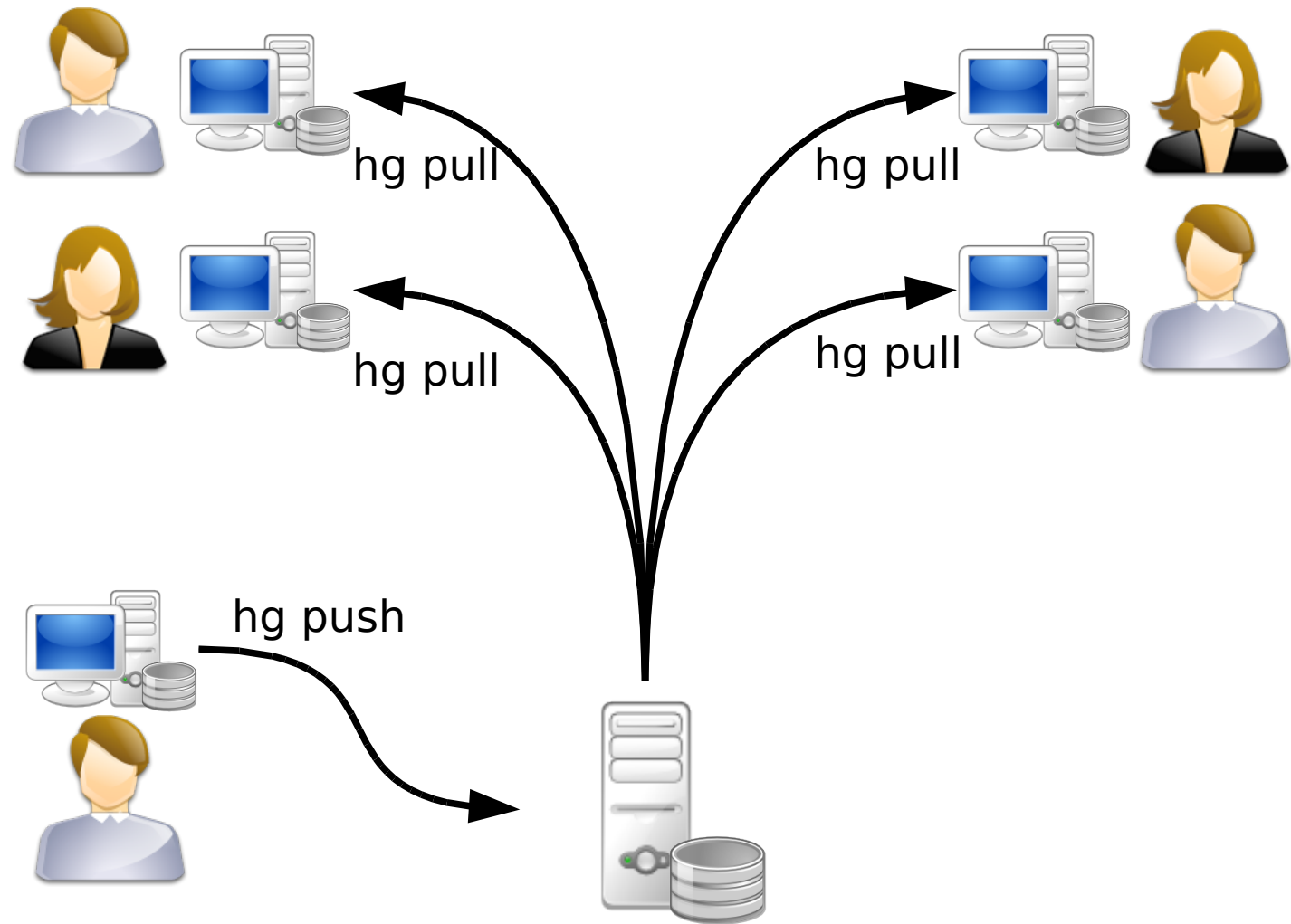
Trabalhando com Mercurial

Ciclo de trabalho



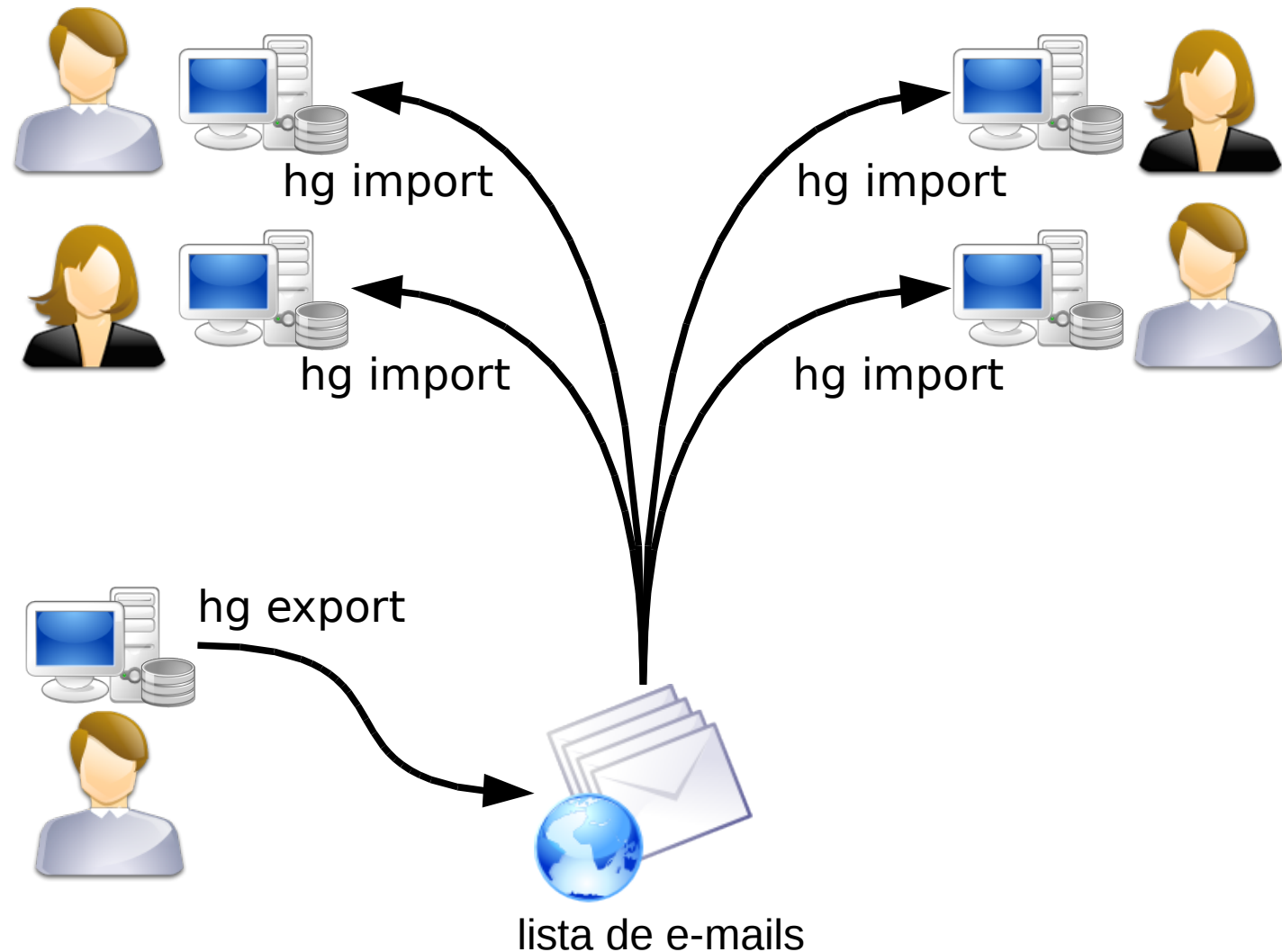
Trabalhando com Mercurial

Ciclo de trabalho



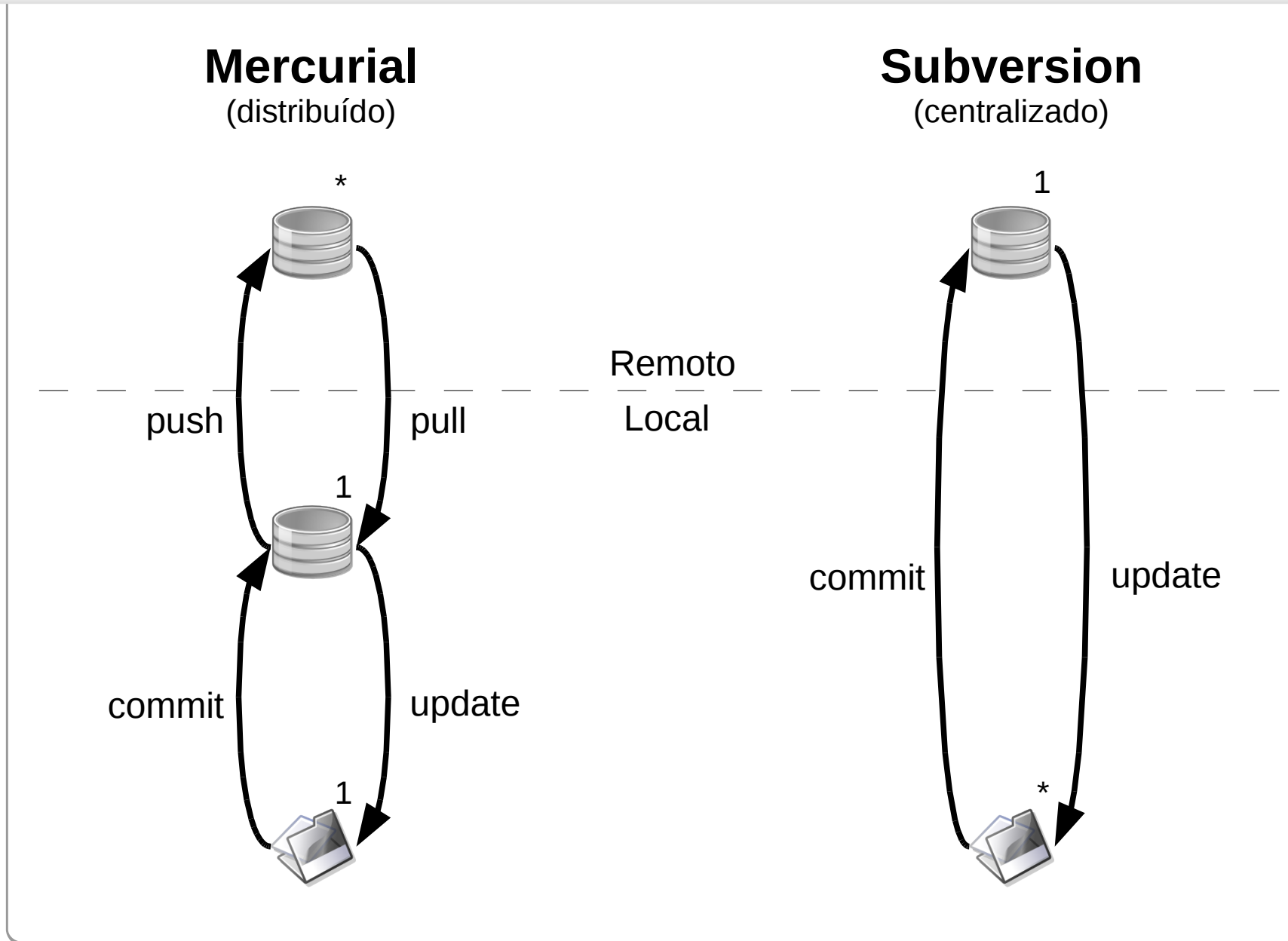
Trabalhando com Mercurial

Ciclo de trabalho



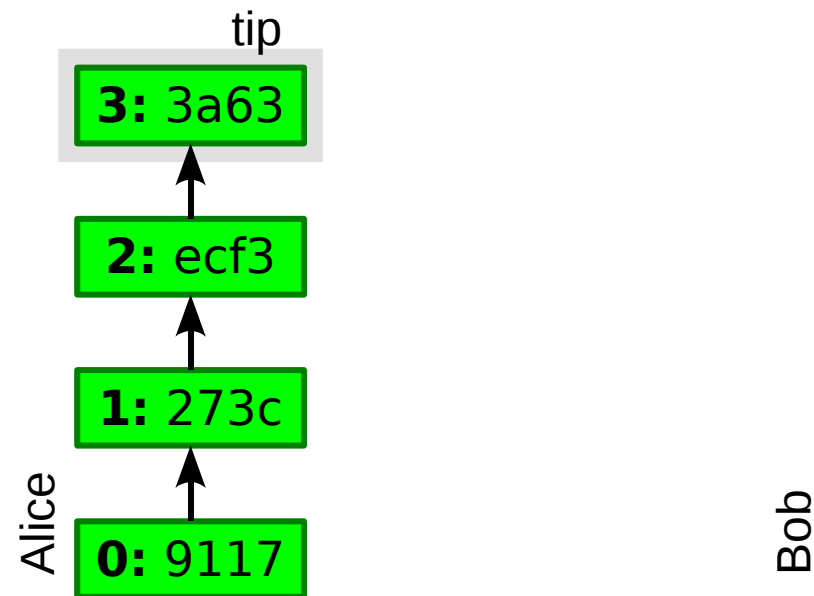
Trabalhando com Mercurial

Ciclo de trabalho



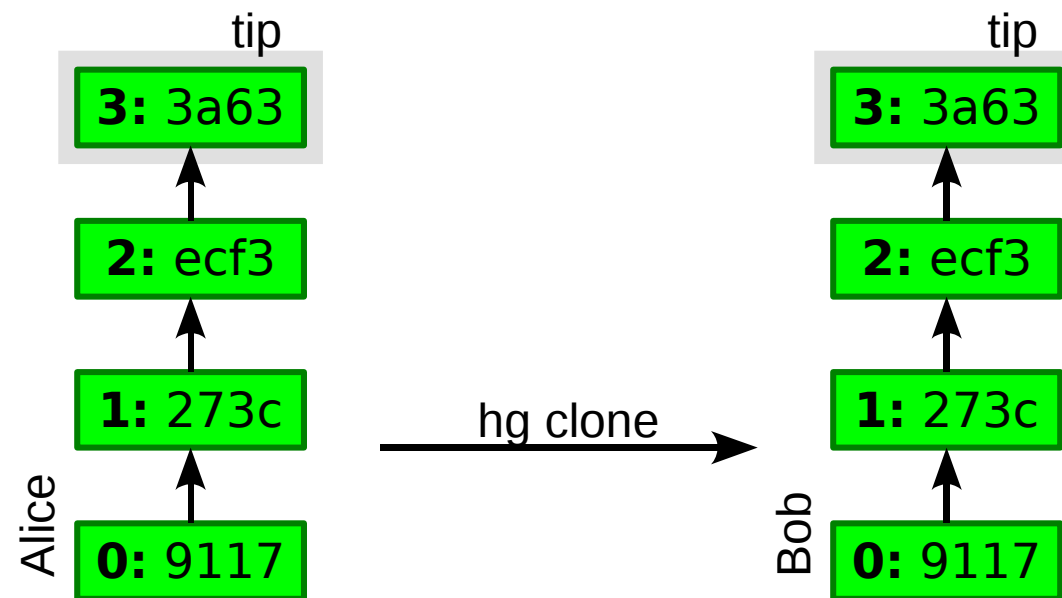
Trabalhando com Mercurial

Grafo de revisões



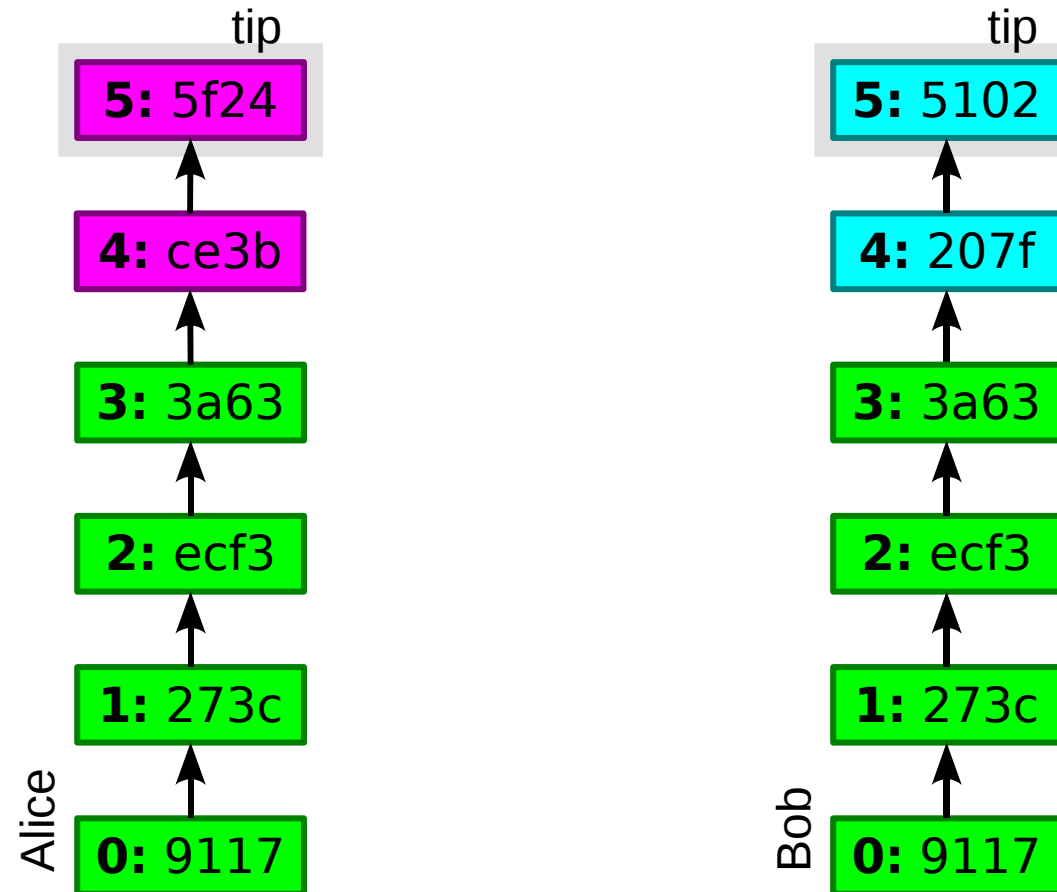
Trabalhando com Mercurial

Grafo de revisões



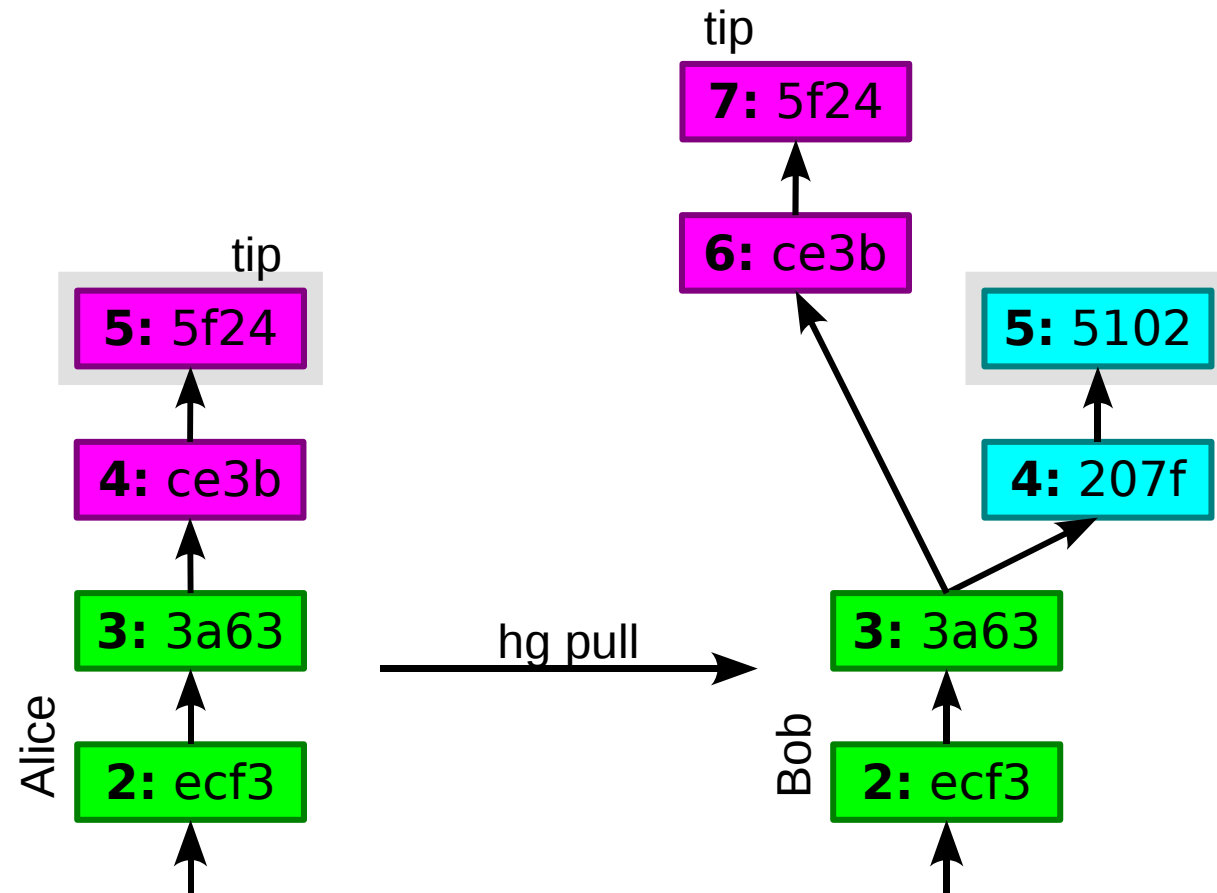
Trabalhando com Mercurial

Grafo de revisões



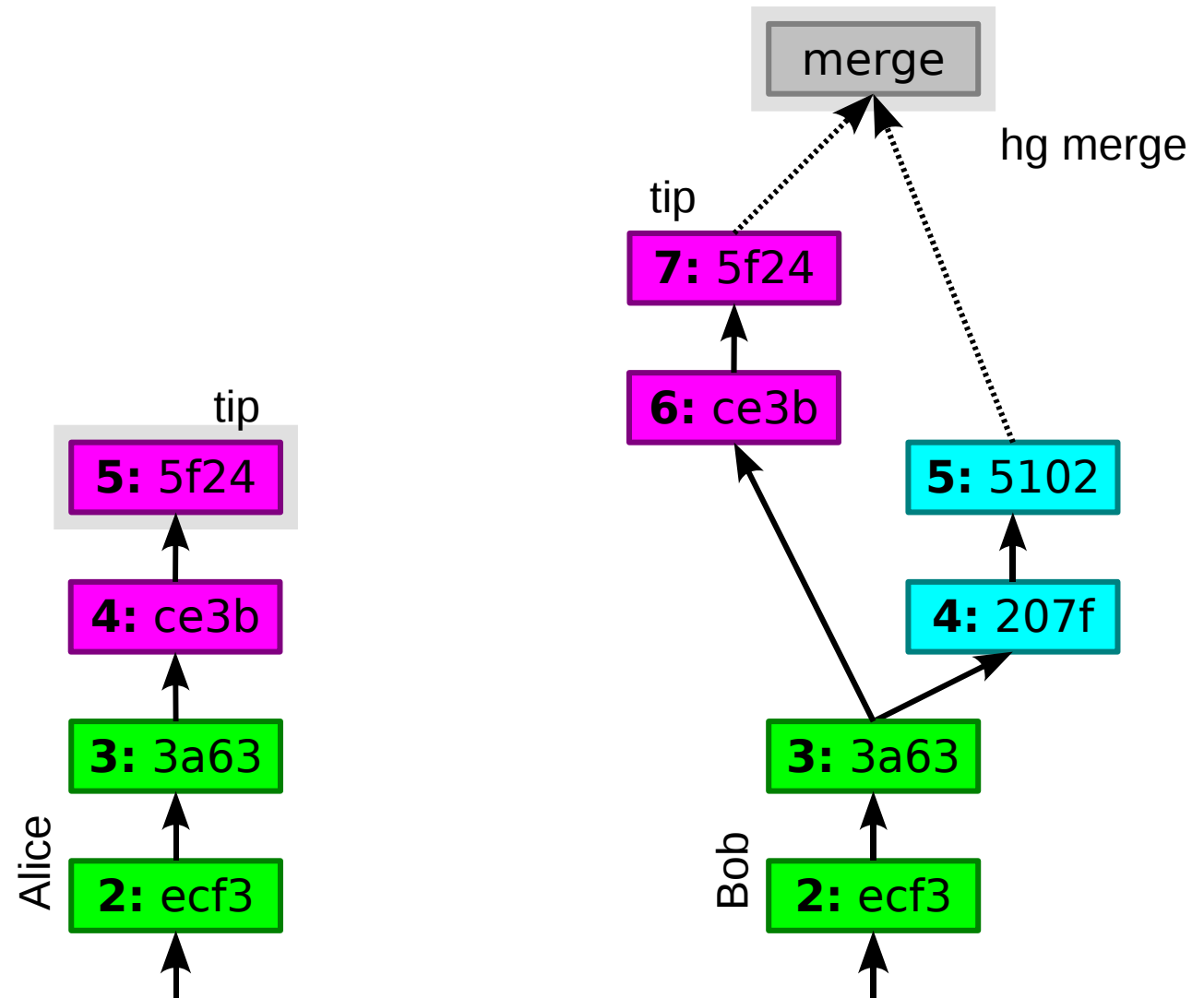
Trabalhando com Mercurial

Grafo de revisões



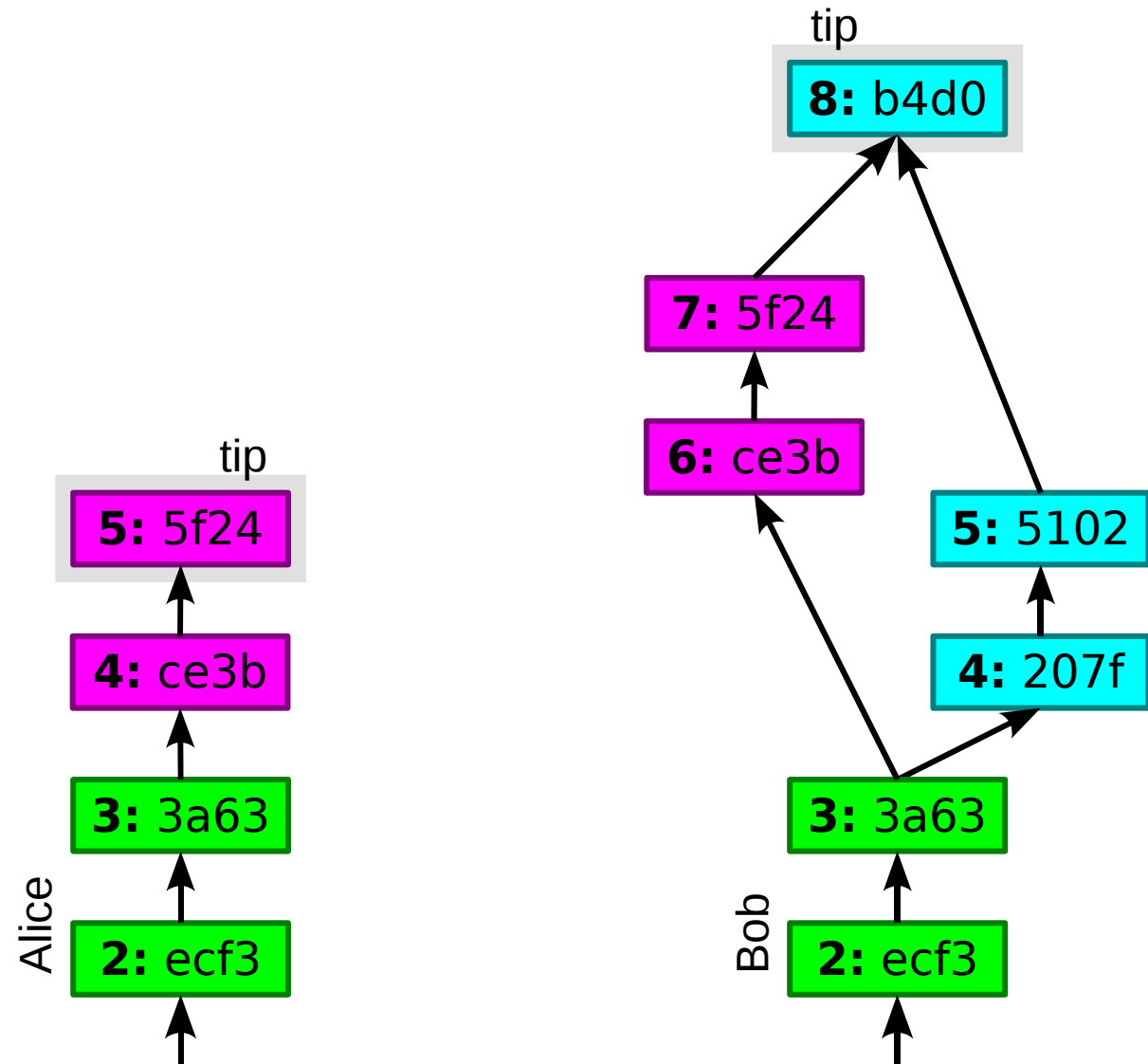
Trabalhando com Mercurial

Grafo de revisões



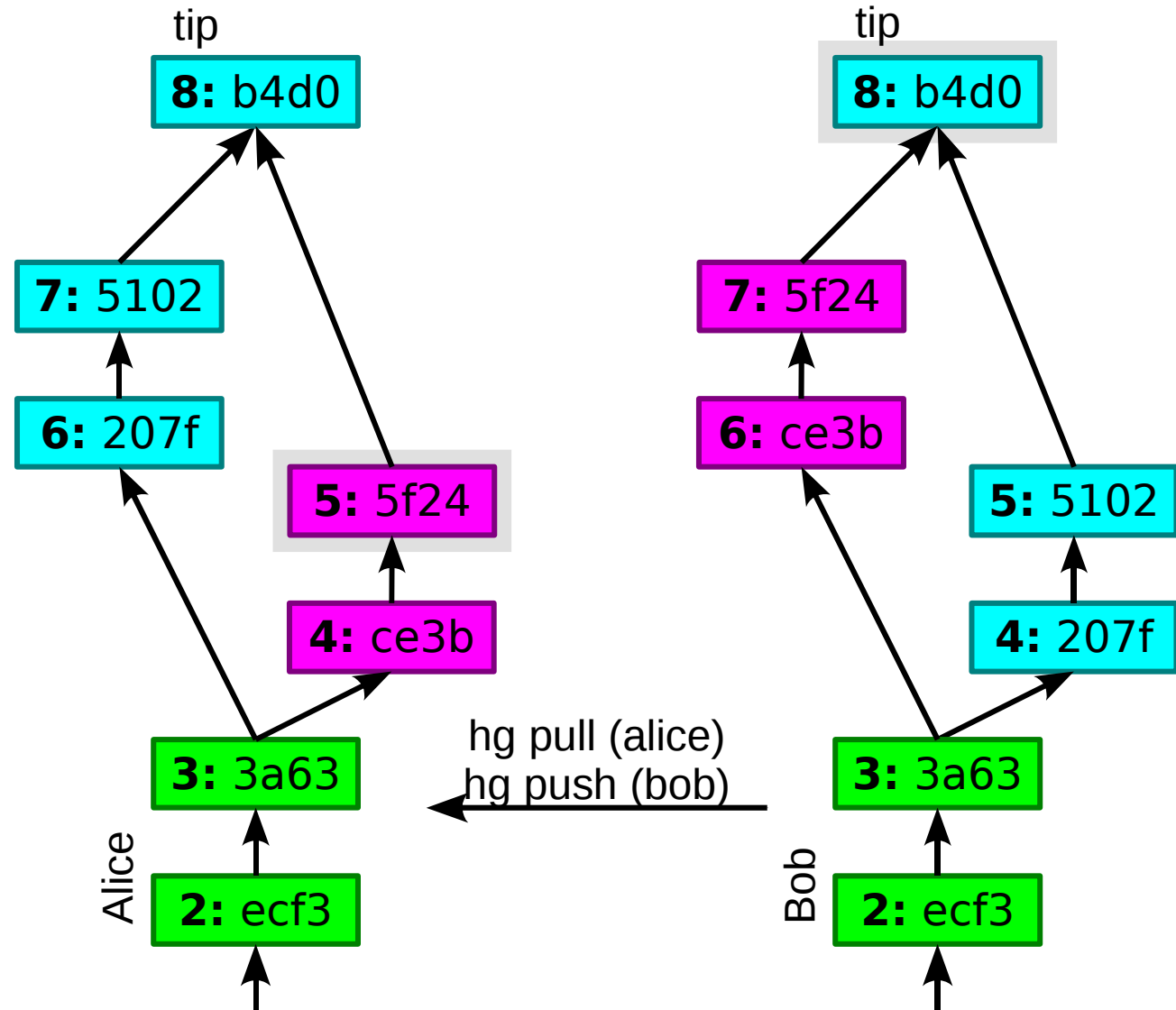
Trabalhando com Mercurial

Grafo de revisões



Trabalhando com Mercurial

Grafo de revisões



Trabalhando com Mercurial

Comandos

- Comandos comuns

- hg add (svn add)
- hg annotate (svn blame)
- hg commit (svn commit)
- hg copy (svn copy)
- hg diff (svn diff)
- hg help (svn help)
- hg log (svn log)
- hg remove (svn delete)
- hg rename (svn move)
- hg revert (svn revert)
- hg status (svn status)
- hg update (svn update)



Trabalhando com Mercurial

Comandos

- Acesso a repositórios
 - Local
 - */home/user/hg/project/*
 - *file:///home/user/hg/project/*
 - HTTP, Mercurial
 - *http://hg.example.com/hg/project/*
 - *https://hg.example.com/hg/project/*
 - SSH, Mercurial
 - *ssh://hg.example.com/hg/project/*



Trabalhando com Mercurial

Criação e clonagem

- Criação do projeto
 - hg init [diretório]
 - Transforma o diretório atual (ou o diretório informado) em um repositório Mercurial
- Clonagem
 - hg clone origem [diretório]
 - Cria uma cópia do repositório de origem
 - É, implicitamente, uma ramificação
 - Hg lembra seu repositório de origem



Trabalhando com Mercurial

Trabalho distribuído

- Trazer (puxar) submissões
 - `hg pull [URL]`
 - Recupera as diferenças entre o repositório indicado e o seu, e aplica as alterações.
 - Se 'URL' for omitido, seu repositório de origem (fornecido ao 'hg clone') é considerado.
 - Atualiza apenas o repositório, use 'hg update' para atualizar a cópia de trabalho.
 - `hg incoming [URL]`
 - Mostra o que há no repositório indicado que não há no seu, e pode ser trazido com 'hg pull'.



Trabalhando com Mercurial

Trabalho distribuído

- Levar (empurrar) submissões
 - hg push [URL]
 - Determina as diferenças entre o seu repositório e o indicado, e as aplica remotamente.
 - Se ‘URL’ for omitido, seu repositório de origem (fornecido ao ‘hg clone’) é considerado.
 - Não cria ramificações remotas, seu repositório precisa estar compatível (via ‘hg pull’).
 - hg outgoing [URL]
 - Mostra o que há no seu repositório que não há no indicado, e pode ser levado com ‘hg push’.



Trabalhando com Mercurial

Trabalho distribuído

- Mescla
 - hg heads [*rev*]
 - Mostra as “revisões-cabeça” do grafo de revisões (todas as ramificações).
 - hg merge [*rev*]
 - Mescla as alterações de uma determinada cabeça à cópia de trabalho.
 - Caso ‘*rev*’ seja omitido, e só houver uma outra cabeça possível de ser mesclada (não ambígua), tal cabeça será selecionada.



Trabalhando com Mercurial

Consulta de estado

- Comandos
 - hg status
 - Mostra arquivos alterados, adicionados, etc.
 - hg identify [-i] [-n] [-b] [-t]
 - Mostra qual a revisão atual da cópia de trabalho.
 - hg parents [-r rev] [alvo]
 - Mostra as revisões ascendentes da revisão atual (ou a revisão fornecida).



Trabalhando com Mercurial

Consulta ao histórico

- Histórico
 - hg log [-v]
 - Descrição textual do histórico.
- Visualização
 - hg view
 - Visualiza o histórico interativamente.
 - Grafo de revisões.



Trabalhando com Mercurial

Ramificações

- Clonagem
 - Criadas implicitamente
 - Basta clonar a cópia de trabalho
 - Eficiente: utiliza *hardlinks* quando possível
- Locais, anônimas
 - Criadas implicitamente
 - Submissão sobre revisão intermediária
- Locais, explícitas
 - Criadas através de 'hg branch'
 - Armazenadas no próprio repositório



Rótulos

- Referenciam determinados *changesets*
- Fazem parte do controle de versões
- Comandos
 - hg tag [-l] [-r rev] nome
 - Cria um novo rótulo '*nome*' para a revisão '*rev*'.
 - Parâmetro '-l': rótulo local.
 - hg tag [-l] --remove nome
 - Apaga um rótulo.



Obrigado!

Juliano F. Ravasi

<http://juliano.info/>

Esta apresentação:

<http://juliano.info/pt/vcs>

