



UNIVERSIDADE
FEDERAL DE
MATO GROSSO DO SUL



Gerência de Configuração de Software

Aula 03 – GCS Centralizado x distribuído

Prof. Dr. Awdren de Lima Fontão
awdren.fontao@ufms.br



Resumo aula anterior



Revisão dos conceitos iniciais



Itens de Configuração de Software



Versões

revisões e variantes



Configuração x Versão

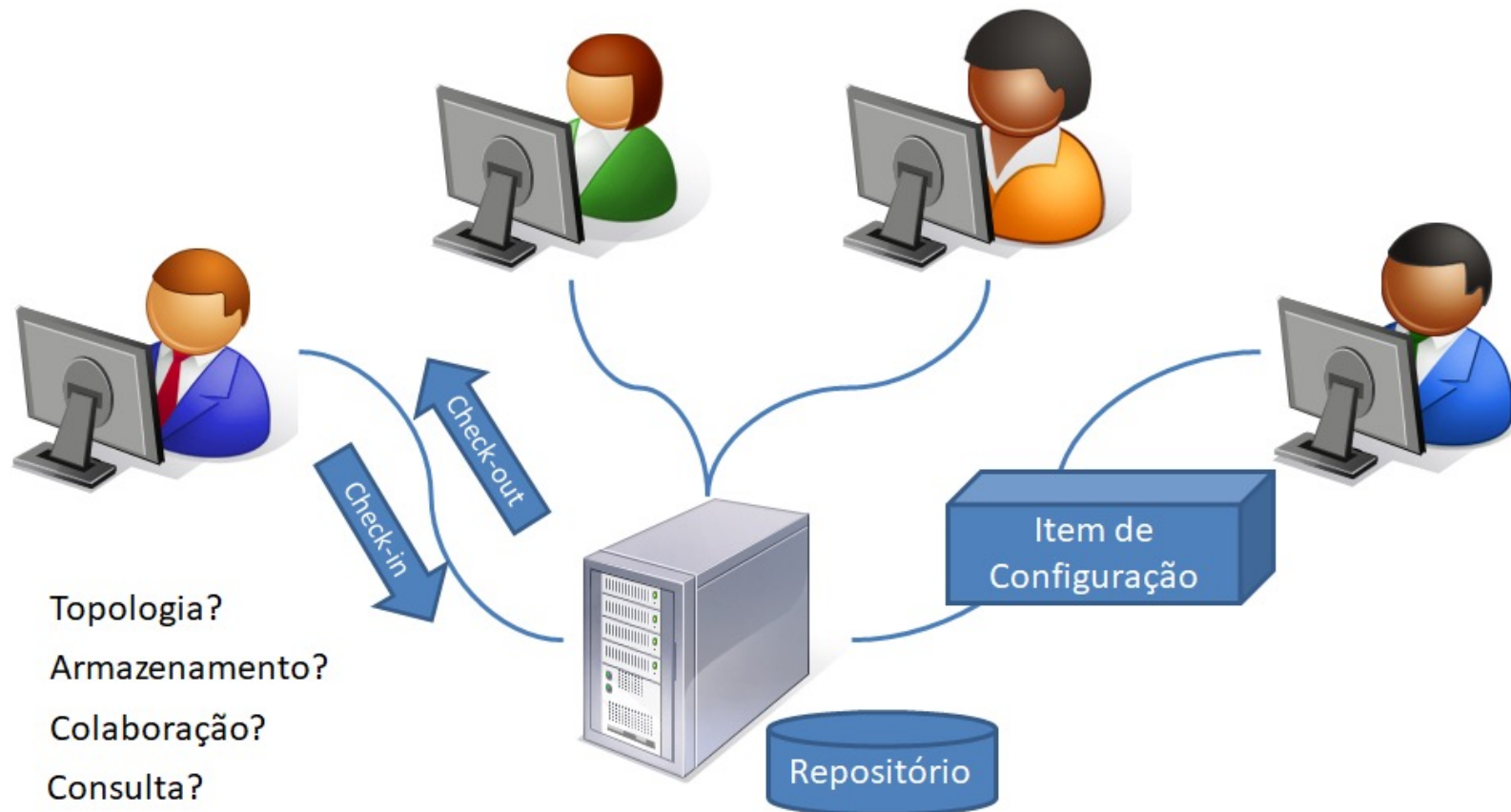


Etiquetas (tags)

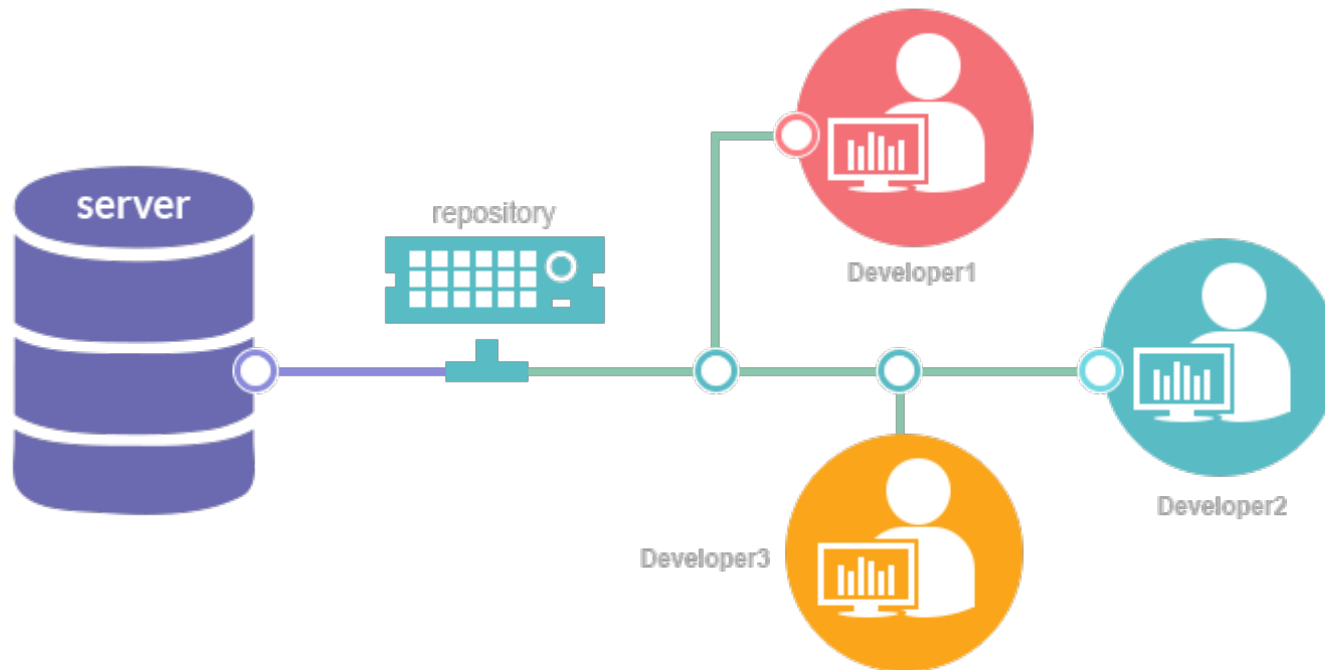


Controle de Versão


Controle de versão



Como Funciona o Controle de Versão?



- O controle de versão é composto de duas partes:
 - Repositório
 - Área de Trabalho




Controle de Versão: Repositório

Armazena todo o histórico de evolução do projeto

Registra toda e qualquer alteração feita em cada item versionado

O desenvolvedor não trabalha diretamente nos arquivos do repositório



Controle de Versão: Área de Trabalho

Contém a cópia dos arquivos do projeto e que é monitorada para identificar as mudanças realizadas

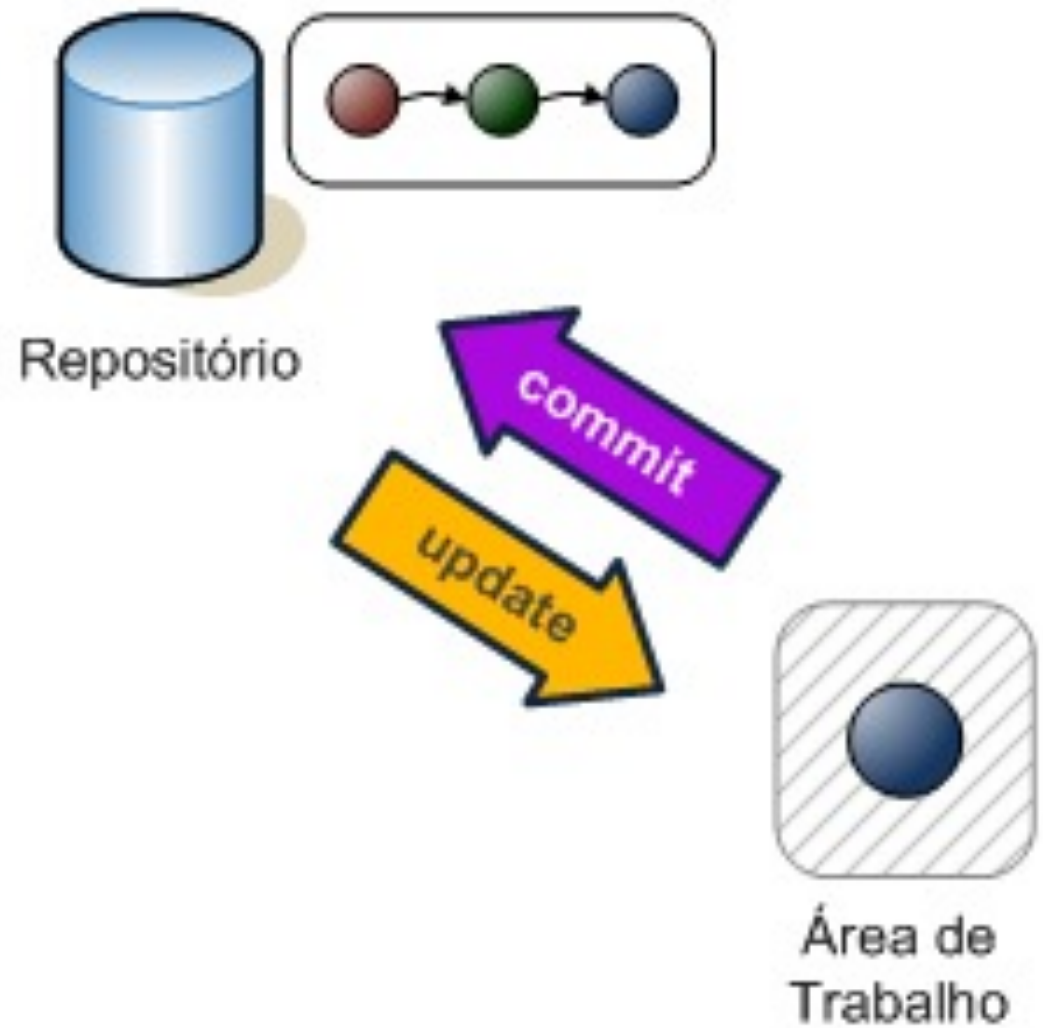
É individual e isolada das demais áreas de trabalho

Controle de Versão de Software - Centralizado e Distribuído

- Muitos problemas de desenvolvimento de software são causados por falta de controle de versão.

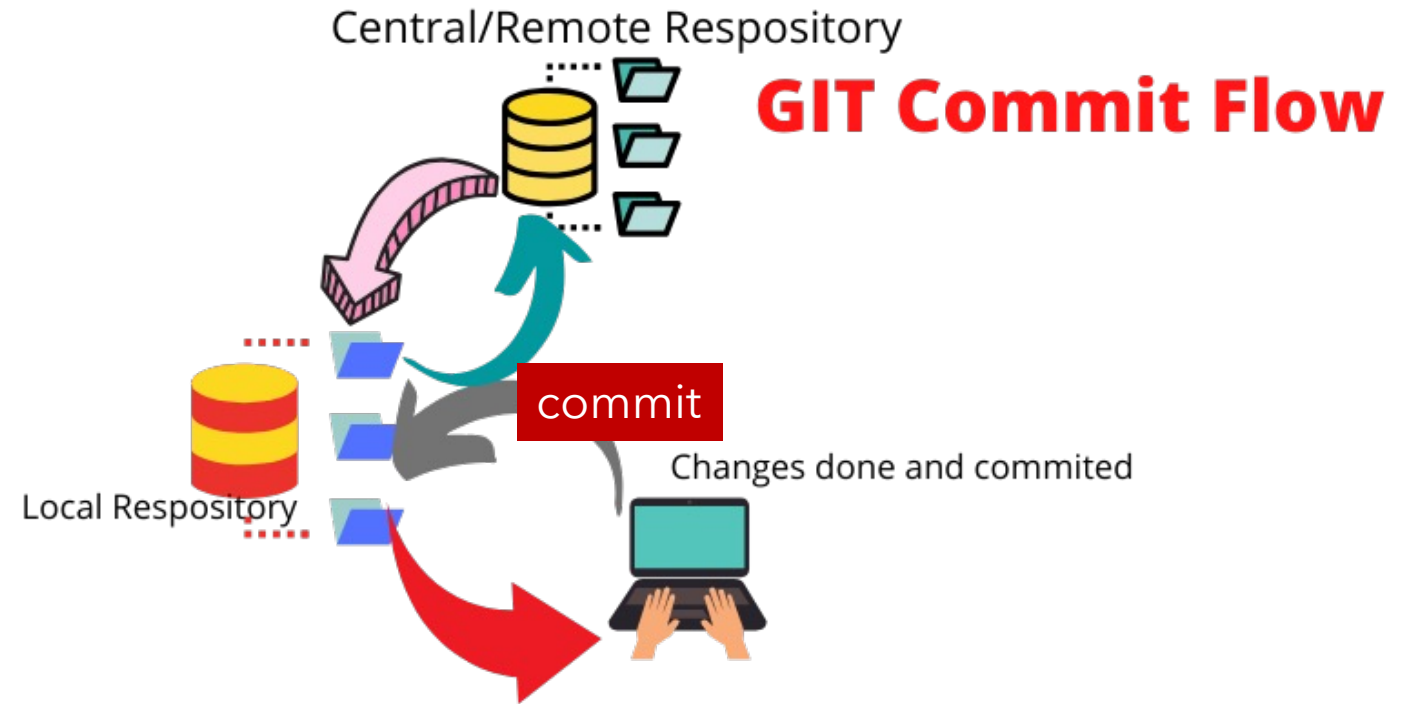


Repositório e Área de Trabalho



Commit

- Realização de um conjunto de mudanças provisórias permanentes, marcando o fim de uma transação;
- Envia alterações para o repositório local, criando uma revisão.



Controle de Versão de Software - Centralizado e Distribuído

Faça uma avaliação rápida da situação da sua equipe de desenvolvimento:

Alguém já sobrescreveu o código de outra pessoa por acidente e acabou perdendo as alterações?

Tem dificuldades em saber quais as alterações efetuadas em um programa, quando foram feitas e quem fez?

Tem dificuldade em recuperar o código de uma versão anterior que está em produção?

Tem problemas em manter variações do sistema ao mesmo tempo?



Controle Centralizado x Distribuído

Controle de versão centralizado:

Com sistemas de controle de versão centralizados, você tem uma única cópia "central" do seu projeto em um servidor e confirma suas alterações nesta cópia central.

Você puxa os arquivos que precisa, mas nunca tem uma cópia completa do seu projeto localmente. Alguns dos sistemas de controle de versão mais comuns são centralizados, incluindo o Subversion (SVN) e o Perforce.

Syncro SVN Client

File Edit Repository Working copy Compare History Tools Options Window Help

Working copy

www.syncrosvncient.com All Files Modified Incoming Outgoing Conflicts

Name	Remote date	Remot...	Remote author	Size	Type
D:\Projects\www.syncrosvncient.com					File Folder
InstData					File Folder
img					File Folder
mb_bt_screenshots.png				2 KB	PNG File
include					File Folder
site-commons					File Folder
css					File Folder
site-commons-v13.css	Jan 3, 2012	16920	mihai	41 KB	Cascading ...
php	Dec 28, 2011	16899	mihai		File Folder
xml					File Folder
thirdparty					File Folder
xml					File Folder
site.xml	Dec 19, 2011	16839	mihai	261 KB	XML Docum...

Repositories Working copy History

D:\Projects\www.syncrosvncient.com\site-commons\css\site-commons-v13.css

site-commons-v13.css [mihai]

```

101 table.buy,table.buy_specials{width:726px;t
102 float:right}
103 table.buy_specials{margin:20px 0 0 20px;}
104 .buy_l .strikeout-fl {color: #ff3300;text-
105 text-align:right;padding-right:3px; /*font
106 font-size:0.9em; font-style:italic; positi
107 }
108 a.buy_product:hover {text-decoration:under
109 td.edition {height:30px;text-align:left;te

```

site-commons-v13.css@Revision 16920 [mihai]

```

96 .strikeout {color: #ff3300;text-decoration:
97
98 .buy_l .strikeout-fl {color: #ff3300;text-d
99 }
100
101 td.edition {height:30px;text-align:left;tex
102
103 td.buy_vsplit{border-right:1px solid #E8E9E
104 #buy_dif .buy_l{margin-left:90px;}

```

Console

```

--revision 16772:16920
www.syncrosvncient.com
/site-commons/css/site-
commons-v13.css
[10:31:57] - Operation
successful
[10:32:49] - Operation
successful
[10:32:49] - svn status
--verbose
www.syncrosvncient.com
/site-commons/css/site-
commons-v13.css

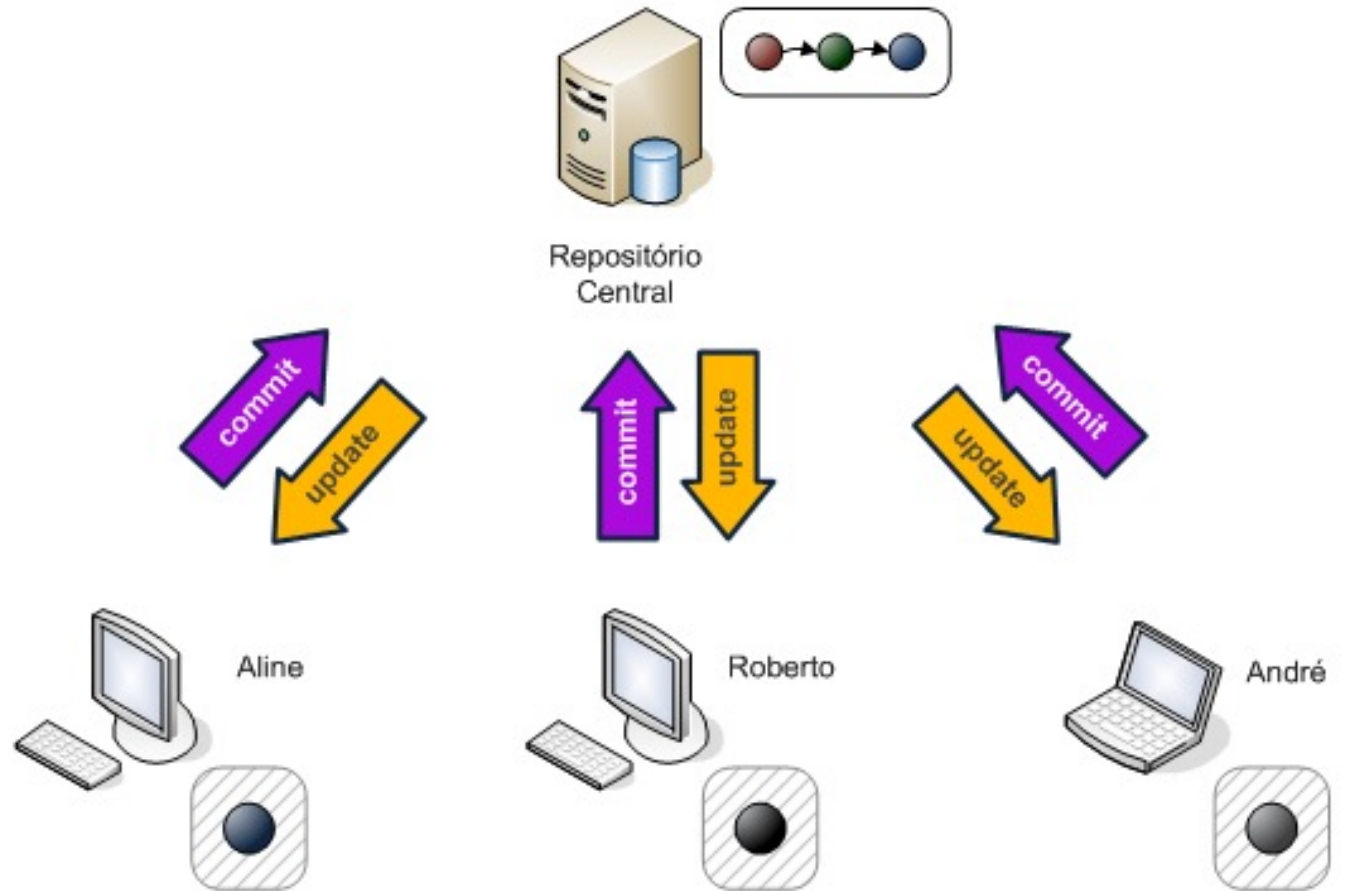
```

Operation successful

74 2 1

Controle Centralizado

- No controle de versão **centralizado** há um único repositório e várias cópias de trabalho que se comunicam apenas através do repositório central.



Controle Centralizado



Vantagens:

Controle de acesso
Cópia de segurança
Controle de qualidade



Desvantagens:

Dependência do repositório central
Ponto único de falha

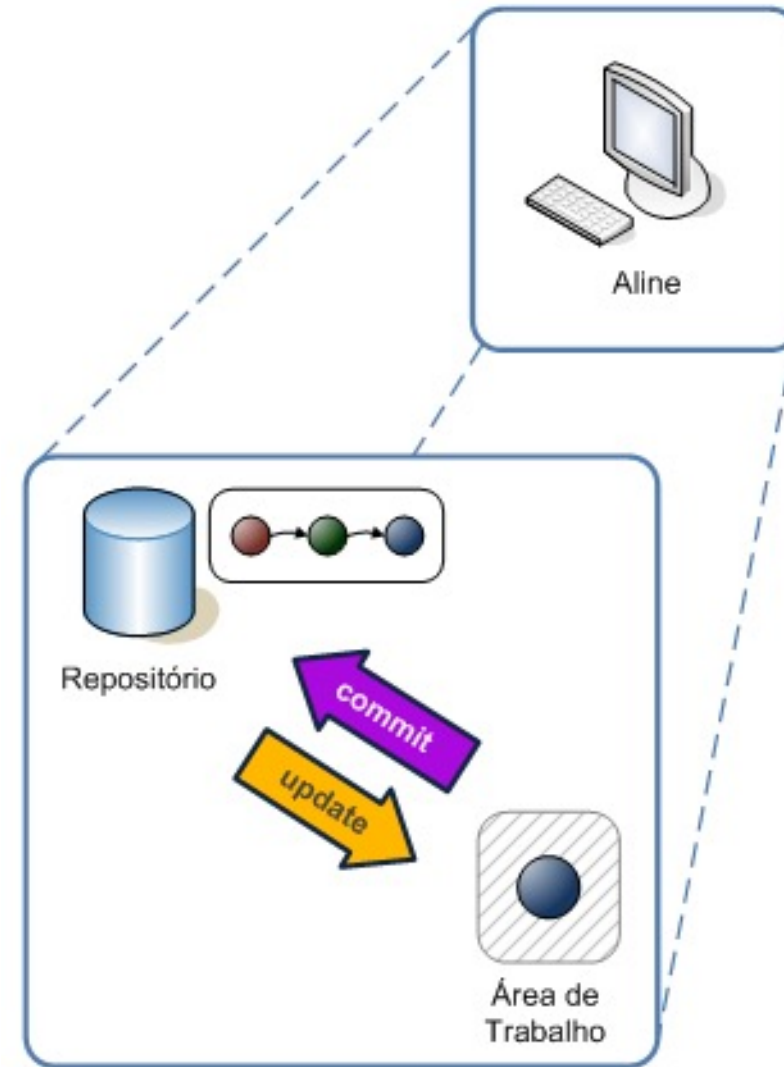


Controle Centralizado x Distribuído

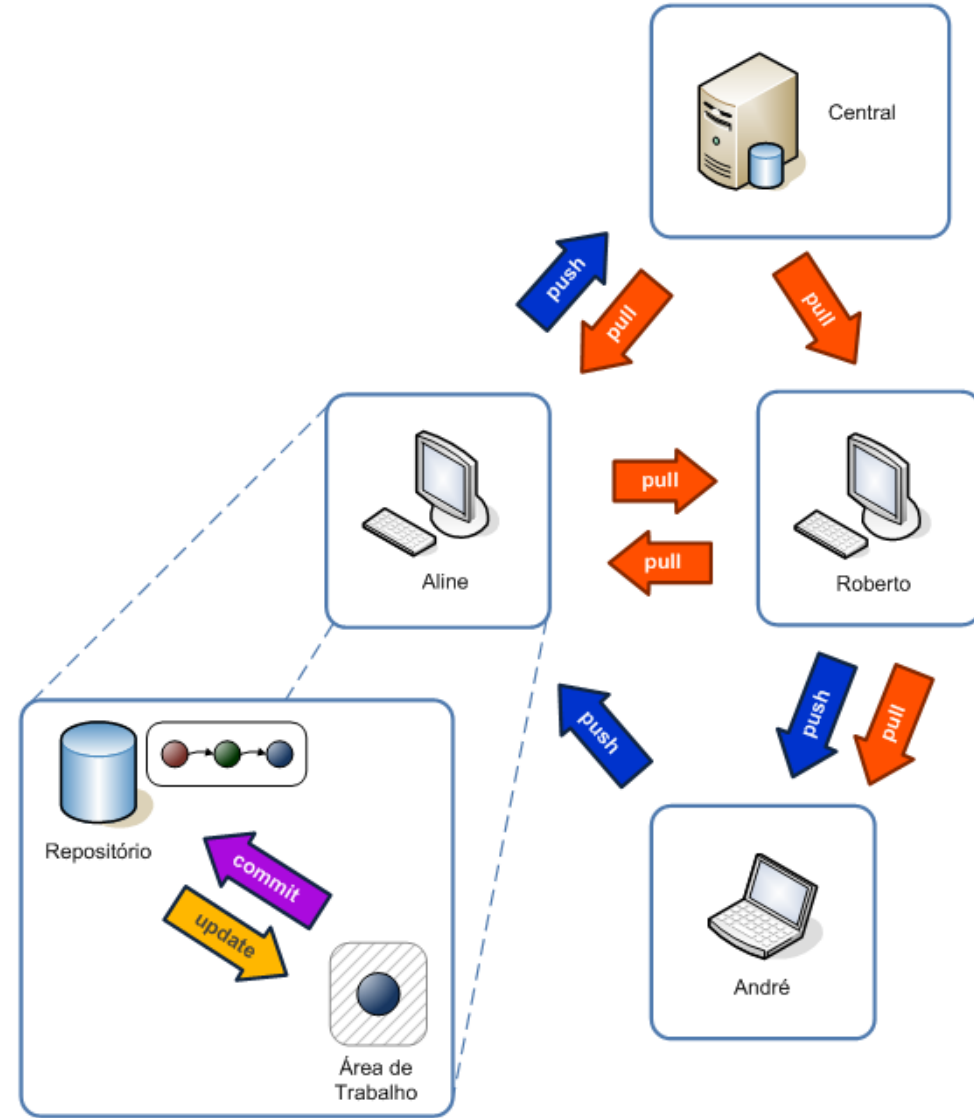
- **Controle de versão distribuído:**
 - Com os sistemas de controle de versão distribuído (DVCS), você não depende de um servidor central para armazenar todas as versões dos arquivos de um projeto. Em vez disso, você clona uma cópia de um repositório localmente para que você tenha o histórico completo do projeto. Dois sistemas comuns de controle de versão distribuída são o Git e o Mercurial.

Controle Distribuído

- No controle de versão **distribuído** cada desenvolvedor possui um repositório próprio acoplado a uma área de trabalho.



Controle Distribuído



Controle Distribuído

- Vantagens do Ponto de Vista do Desenvolvedor:
 - **Rapidez:**
 - As operações são processadas localmente. Não é necessário passar pela rede e contatar o servidor central para fazer um commit, log ou diff por exemplo.
 - **Autonomia:**
 - A conexão com a rede só é necessária para trocar revisões com outros repositórios. Fora isso, trabalha-se desconectado e em qualquer lugar, como num cliente por exemplo.
-




Controle Distribuído

Vantagens do Ponto de Vista da Gerência/Coordenação:
Parte das decisões gerenciais envolve manter livre o caminho da equipe para que possam trabalhar da melhor maneira possível. Outras decisões importantes são sobre redução de custos.

Nestes dois casos específicos, o modelo distribuído oferece as seguintes vantagens:

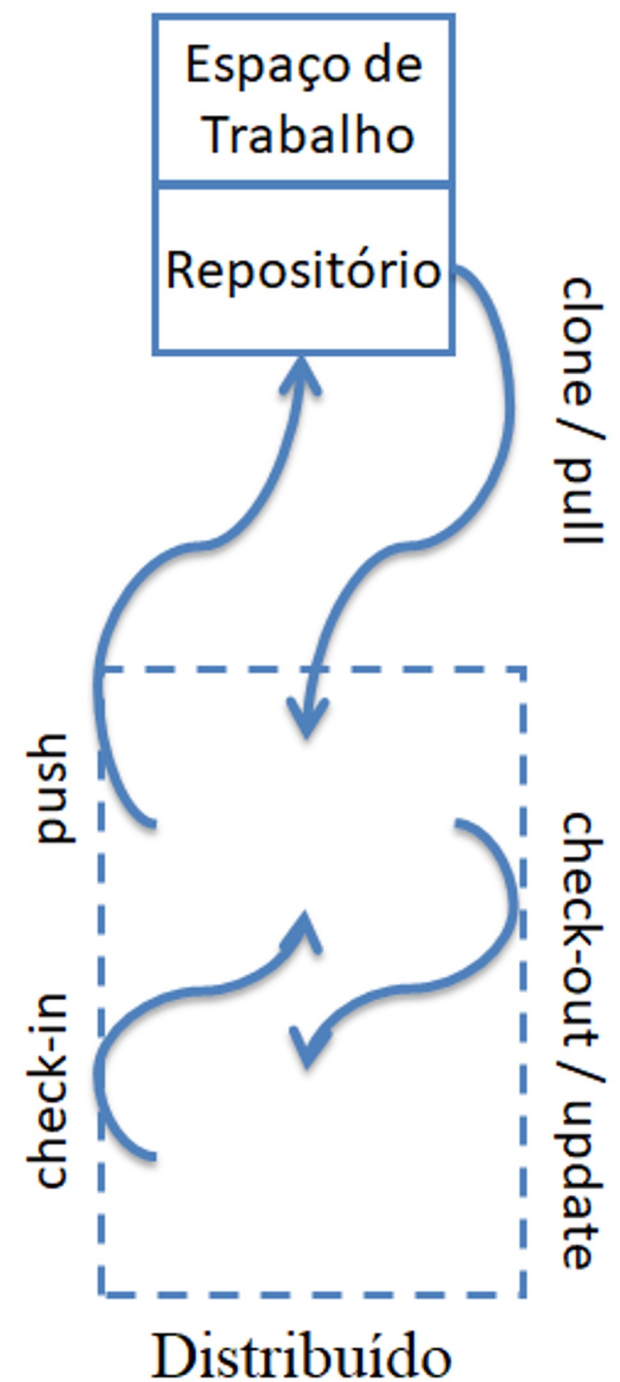
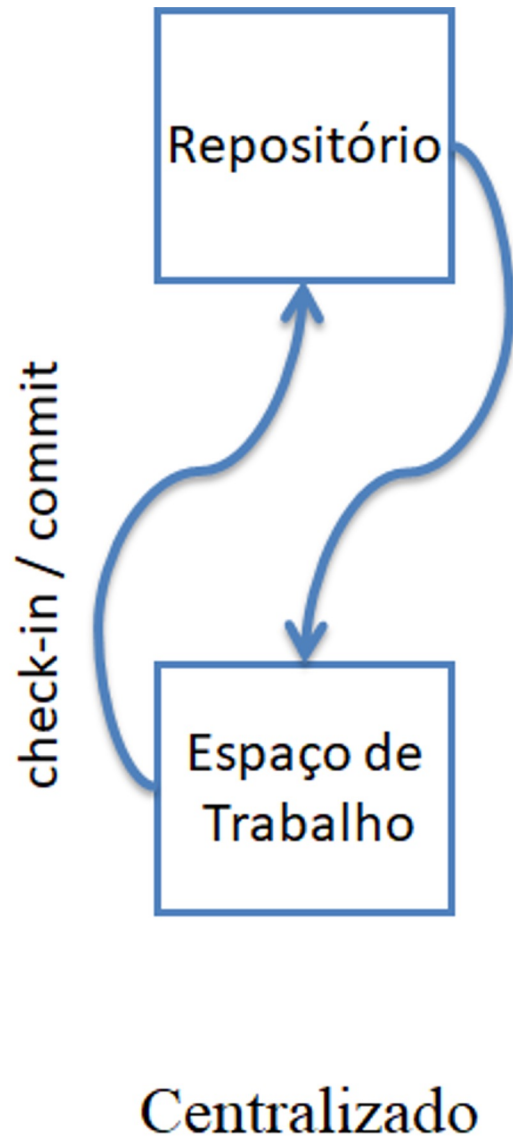
- **Confiabilidade:**
 - No sistema centralizado, uma pane no servidor interrompe todo o desenvolvimento. Já no sistema distribuído, além de a equipe poder continuar seu trabalho, os repositórios dos desenvolvedores funcionam como cópias de backup de todo o projeto.
- **Redução de custos com servidor:**
 - A carga de processamento fica distribuída nas próprias máquinas dos desenvolvedores. O repositório "central", quando existe, tem o papel do repositório "oficial" e não como processador central das requisições.

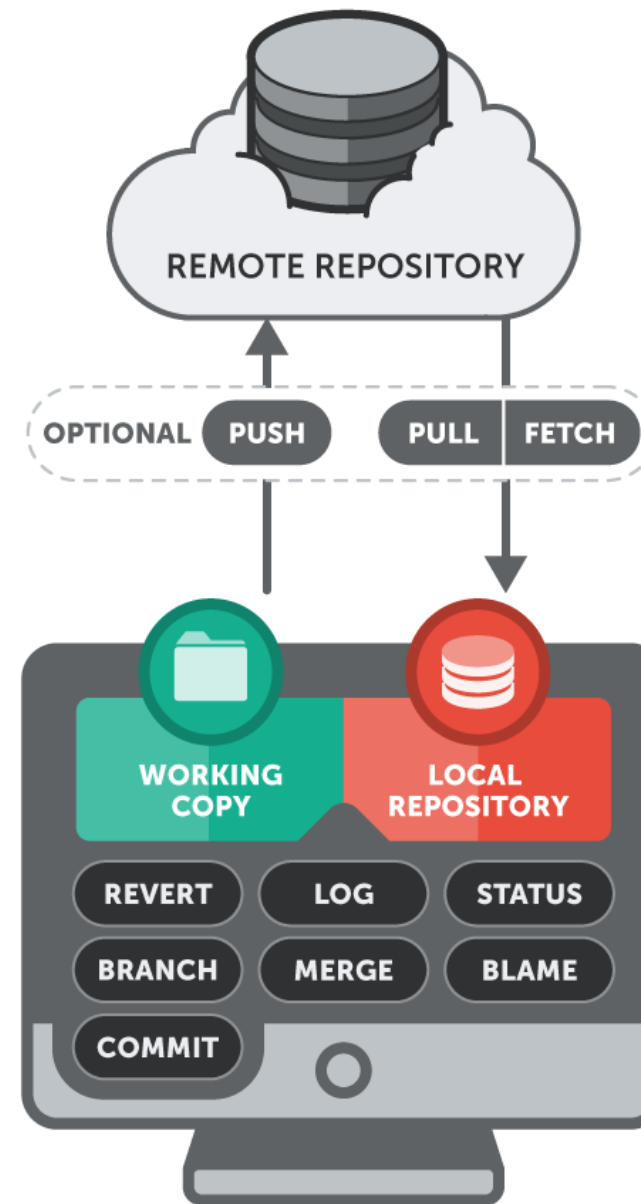
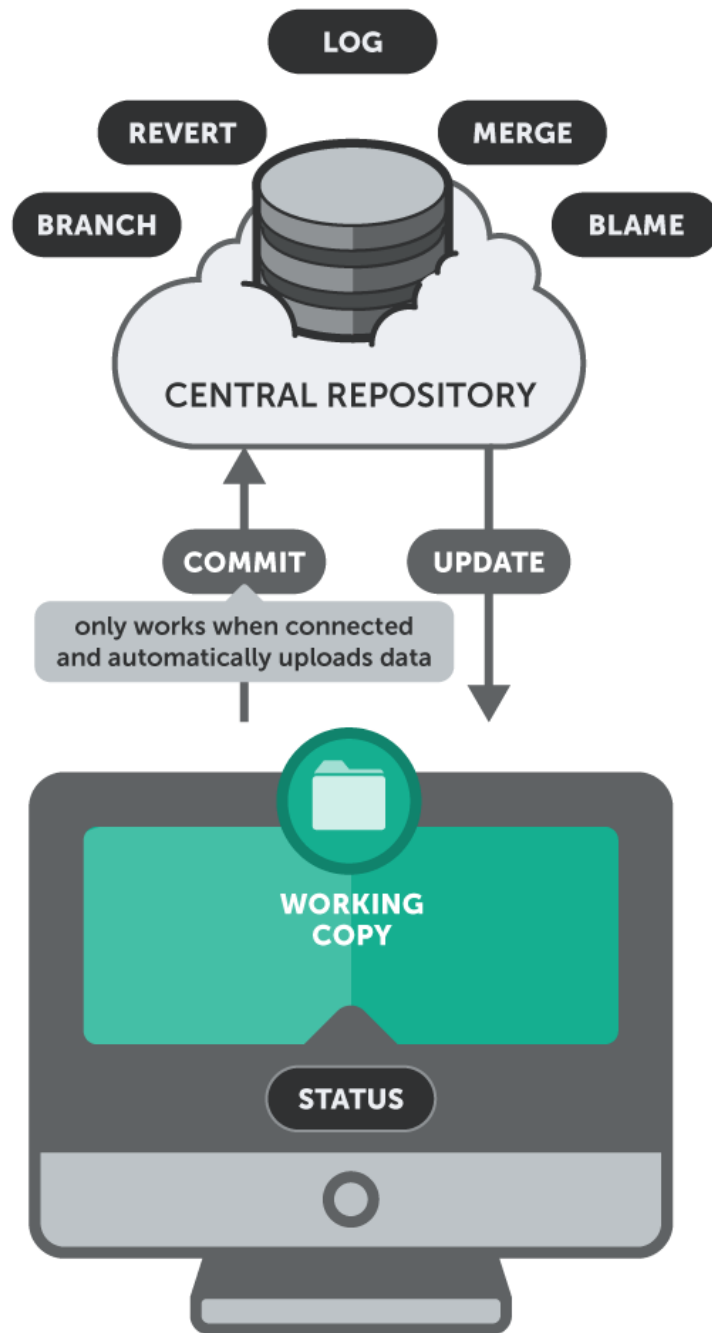



Operações Básicas dos Controles de Versão Centralizado e Distribuídos

Centralizado	Distribuído	Descrição
checkout	clone	Criação da cópia de trabalho/repositório
commit	commit	Envia alterações para o repositório, criando uma revisão
update	update	Atualiza a cópia/área de trabalho em uma revisão
	pull	Importa revisões feita em outro repositório
	push	Envia revisões locais para outro repositório

Topologia







Exemplo de ferramentas de controle de versões

Software Livre	Software Comercial
<ul style="list-style-type: none">→ Git→ Mercurial→ Subversion	<ul style="list-style-type: none">→ BitKeeper (BitMover)→ ClearCase (IBM Rational)→ Perforce→ PVCS (Serena)→ StarTeam (Borland)→ Synergy/CM (Telelogic)→ Team Foundation Server (Microsoft)



Identificação de Revisões

Uma revisão precisa de uma identificação única

Identificação de Revisão: Centralizado

- Cada revisão produzida recebe um número inteiro sequencial.
 - Ex: 1, 2, 3,
- Como existe um repositório, a numeração de revisão é a mesma para todos os desenvolvedores

Committed Changes

	Rev	Scores	Commit log message
★	r5356		Extra import lint from r5314 cl
★	r5355		Issue 34804: Fix javadoc tag
★	r5354		Adds HTMLTable.clear(boolean)
★	r5353		Fixes issue 3647: Long.parse
★	r5352		Fixed checkstyle errors. Patc
★	r5351		Fixed checkstyle errors. Patc
★	r5350		BUILD FIX: restore draftComp
★	r5349		BUILD FIX: missed a necessa

Identificação de Revisão: Distribuído

- Os repositórios são autônomos e portanto, não há como definir uma numeração sequencial compartilhada para todos.
- Solução:
 - Identificar cada revisão com uma numeração que nunca se repita em qualquer repositório
 - Uso de um *hash* SHA-1, que produz um número de 160 bits (40 dígitos na forma hexadecimal)
 - Esse um número é tão grande e específico que torna extremamente improvável a colisão com um *hash* produzido por outro repositório.
 - A equipe define uma nomenclatura para a realização da rastreabilidade.
 - **nome_desenvolvedor#numero_tarefa**
 - **fontao#1234**

Identificação de Revisão: Distribuído

Committed Changes

	Rev	Scores	Commit log message
☆	9b43fe121f		Merge in the erratic
☆	b11b659298		Merge in the un
☆	2e610b4d9c		Shorten the
☆	3b5d7ecd51		Make an un:
☆	921d335549		Trim the READI
☆	1682c71ea3		Erratic branch F
☆	af55c85092		Added a README

Sincronização de Mudanças Concorrentes

Uma das responsabilidades do controle de versão é:

possibilitar o trabalho paralelo e
concorrente de vários
desenvolvedores sobre os mesmo
arquivos

evitar que um desenvolvedor
sobrescreva o código de outro, o que
resultaria no reaparecimento de
defeitos e perda de funcionalidades.



Em parte, isto é conseguido através da área de trabalho, que fornece a impressão de que o desenvolvedor é o único dono de todo o projeto. Mas só a área de trabalho não resolve todo o problema. Ainda é necessário um jeito de sincronizar os esforços dos membros da equipe.

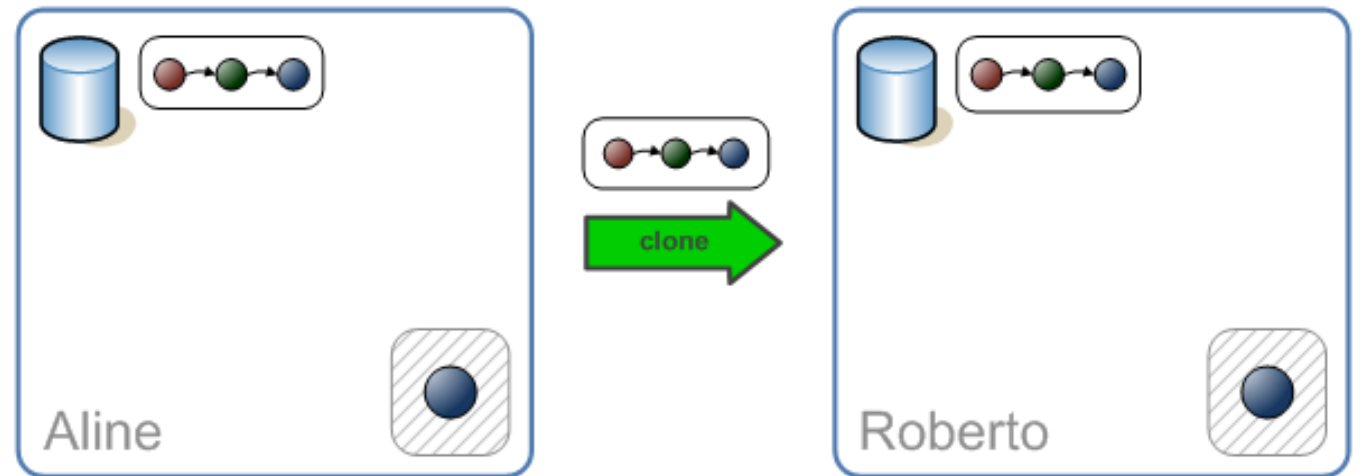
Sincronização de Mudanças Concorrentes

- A sincronização é feita combinando-se revisões concorrentes em uma única resultante.
- Essa operação é conhecida como **merge** (mesclagem) e, apesar de parecer complexa, acontece sem conflitos na maioria das vezes – aliás, essa é a ideia.



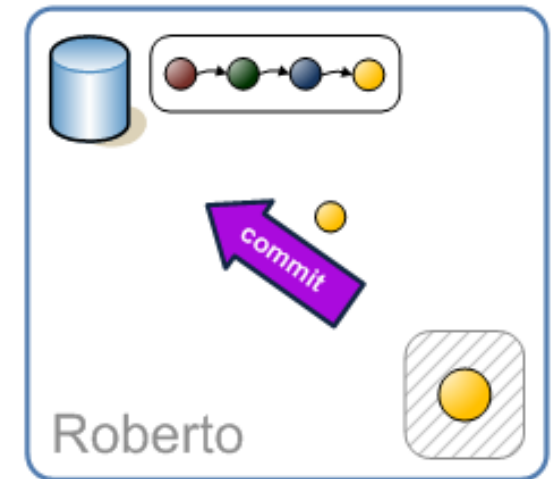
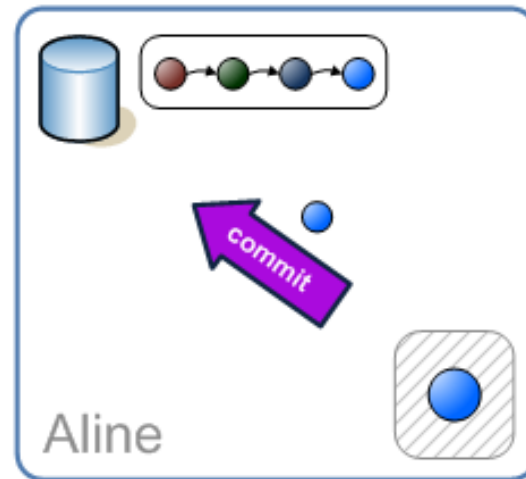
Sincronização no Controle de Versão Distribuído

- 1. Roberto clona o repositório de Aline. Agora, ambos partem do mesmo ponto.



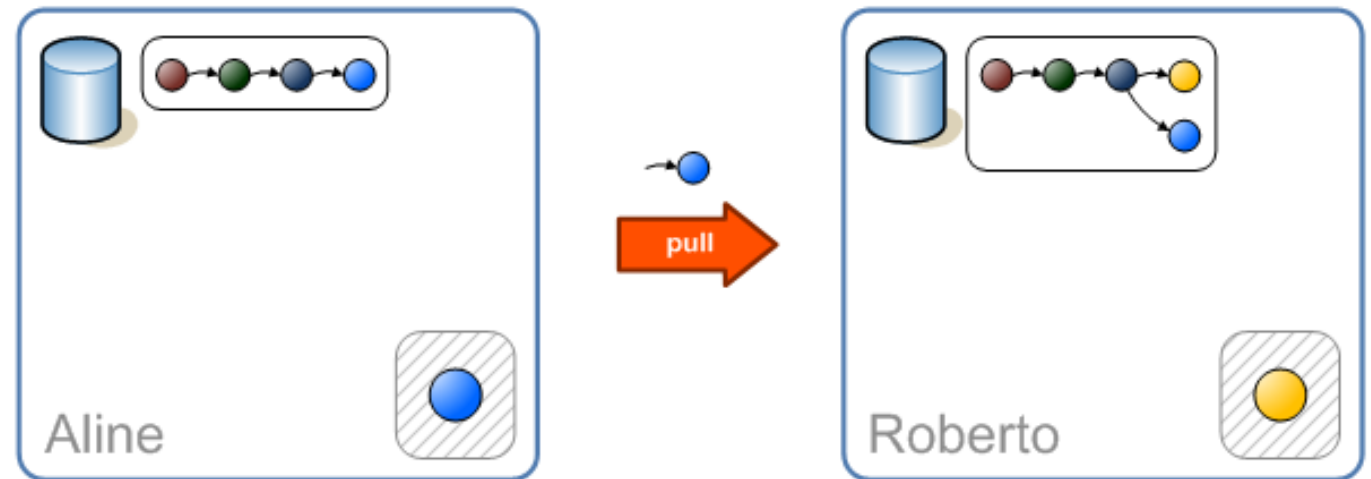
Sincronização no Controle de Versão Distribuído

- 2. Aline e Roberto publicam suas alterações nos seus respectivos repositórios, sem interferir no repositório um do outro.



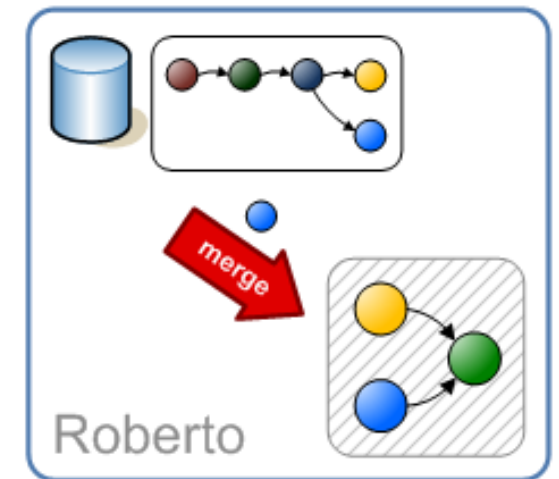
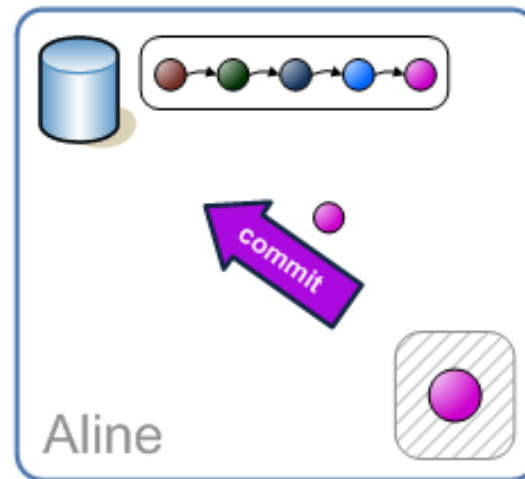
Sincronização no Controle de Versão Distribuído

- 3. Roberto sincroniza seu repositório com as revisões publicadas por Aline. Sua área de trabalho não é afetada pela sincronização.



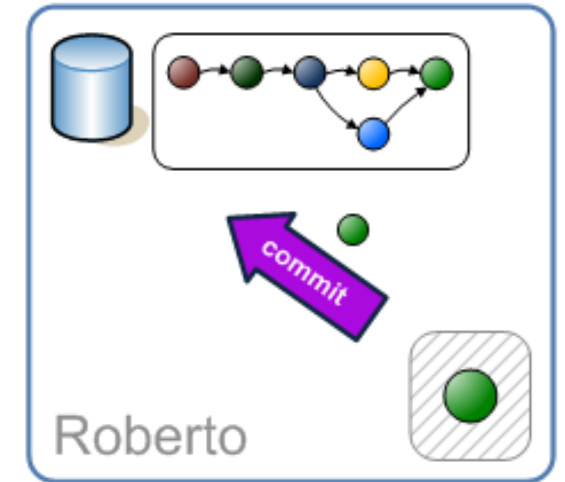
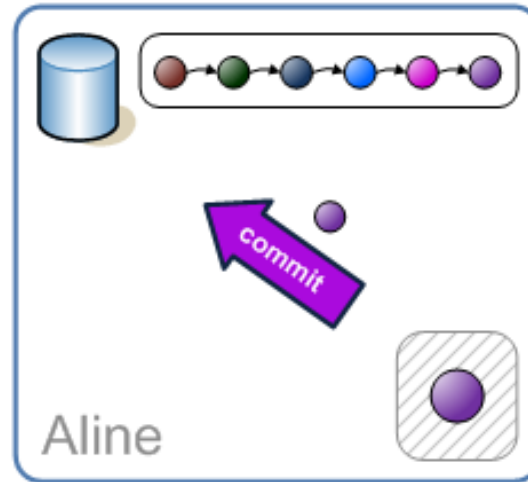
Sincronização no Controle de Versão Distribuído

- 4. A mesclagem entre as revisões de Aline e Roberto é feita explicitamente na área de trabalho de Roberto através de um comando *merge*. Enquanto isso, Aline já gera outra revisão no seu repositório.



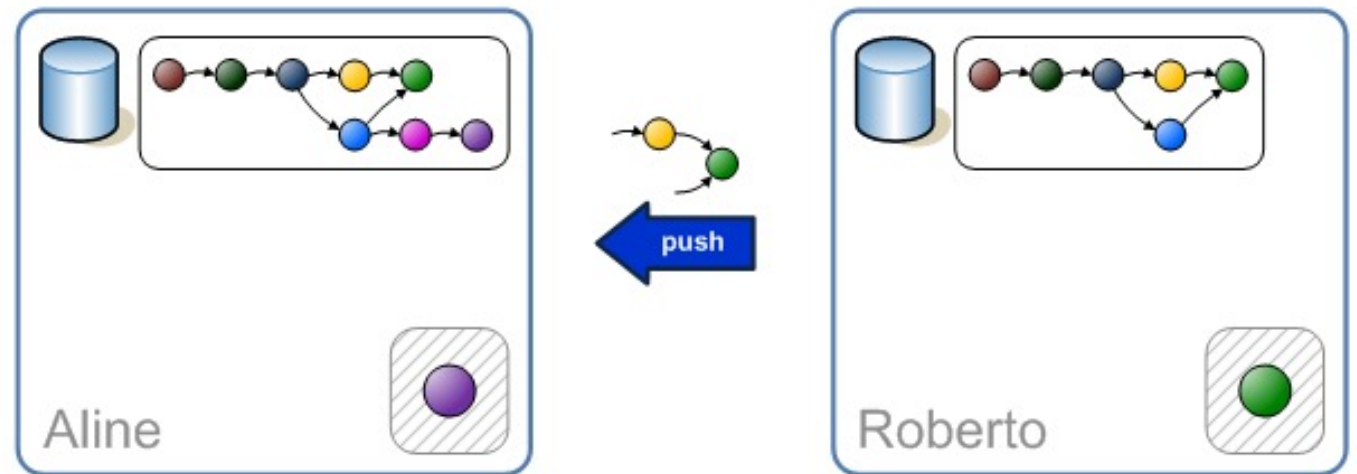
Sincronização no Controle de Versão Distribuído

- 5. Após conferir se a mesclagem produziram o resultado desejado, Roberto envia as mudanças ao seu repositório. Paralelamente, Aline publica mais uma vez no seu repositório.



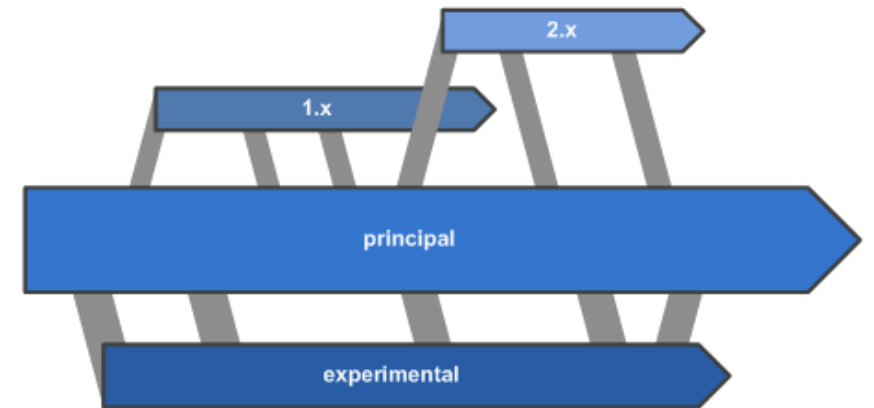
Sincronização no Controle de Versão Distribuído

- 6. Roberto envia suas revisões ao repositório de Aline, que as combina com o histórico de revisões já existente.



Diferentes versões de projeto

- Muitos projetos precisam de variações específicas.
 - Um caso típico é para customizações feitas para atender determinados clientes que precisam de adaptações particulares.
 - Outro caso comum é a criação de um ramo para experimentações no projeto, sem comprometer a linha principal de desenvolvimento.
- O controle de versão oferece funcionalidades que facilitam a coordenação de ramos diferentes de desenvolvimento em um mesmo projeto.



Considerações Finais

Controle de versão resolve diversos problemas intrínsecos ao desenvolvimento de software.

É uma prática de engenharia de software comprovadamente eficaz.

Faz parte das exigências para melhorias do processo de desenvolvimento de certificações, tais como: **CMMi**, **MPS-Br**.