PUC Minas

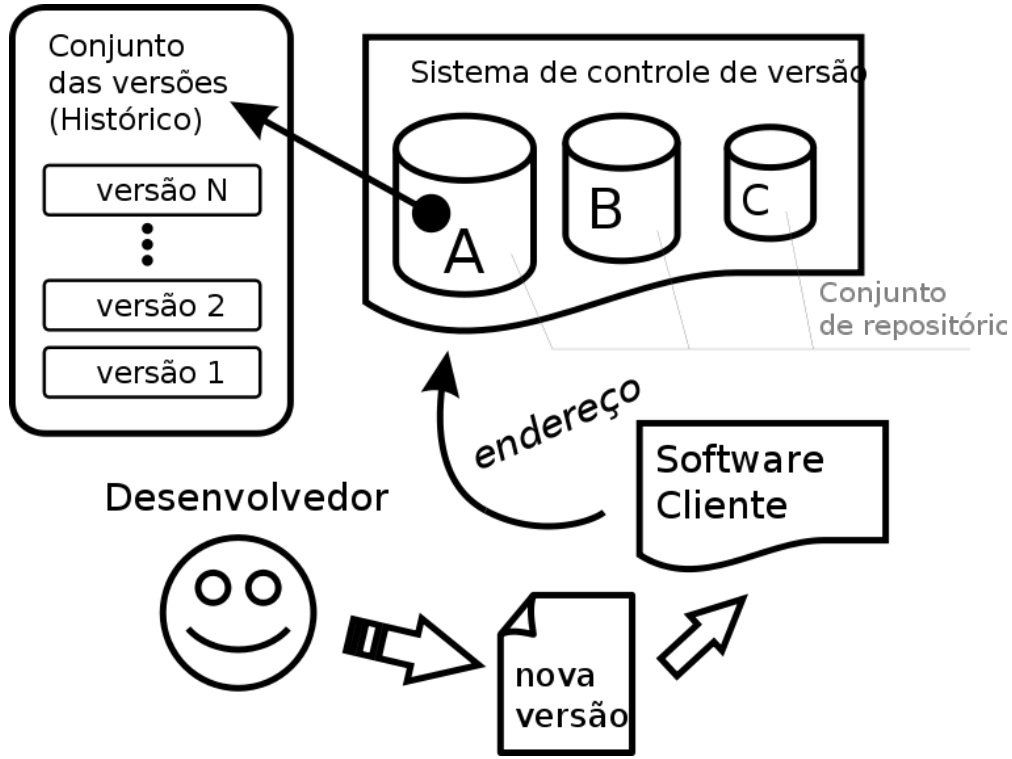
Trabalho Interdisciplinar

Controle de Versão (Source Control)

Profª. Leila Ribeiro de Oliveira

Controle de Versão

Quando um projeto de software cresce, o controle de versão se torna essencial.



"O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou em um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas".

(Scott Chacon e Ben Strub, 2 Edição, 2014)

Controle de Versão

Os sistemas de Controle de Versão são a melhor forma de se controlar o histórico de desenvolvimento de uma aplicação principalmente quando este é realizado por diversos desenvolvedores, pois auxiliam os desenvolvedores a rastrear e gerenciar alterações no código de um projeto de software de forma rápida e segura.

GIT

O **git** é um sistema de controle de versões distribuído

- Toda a base de código e histórico fica disponível no computador de todos os desenvolvedores
- Criado por Linus Torvalds em 2005.

Referência: https://git-scm.com/

GITHUB

O **github** é um repositório remoto, público e privado (pago).

- Plataforma de hospedagem (de código-fonte) que usa o Git como sistema de controle de versão.
- Local na Web para armazenamento de projetos.
- Também usado como rede social.

Referência: https://github.com/

Conceitos importantes

Para trabalharmos com o controle de versão precisamos primeiramente entender alguns conceitos importantes, como o que é um ***branch***, um ***commit*** e uma ***merge***.

COMMIT

Um **commit** é como se você fizesse uma cópia atual de todo o diretório, arquivando essa cópia junto com uma chave única de referência a esse **commit** e uma cópia da chave única do ***commit*** anterior em um **banco de dados**

Cada **commit** fica com uma referência ao **commit** anterior.

- estrutura de um grafo direcionado que começa no primeiro **commit** caminhando até o atual.
- somente o primeiro **commit** não possui uma referência a um **commit** anterior

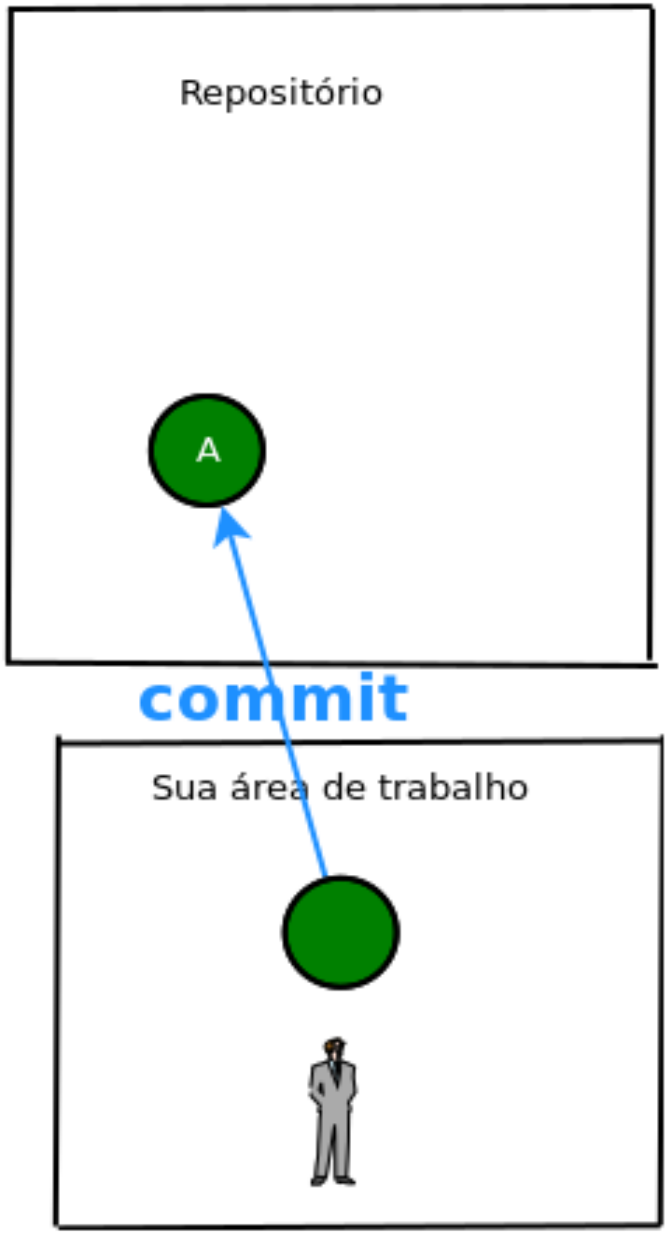
COMMIT

Exemplo:

Pense em você desenvolvendo uma aplicação Web:

- primeiro você constrói o acesso ao banco de dados,
- você testa esse acesso,
- conclui que está ok e então você o publica no seu controle de versão. Ou seja, você realiza um ***commit***.

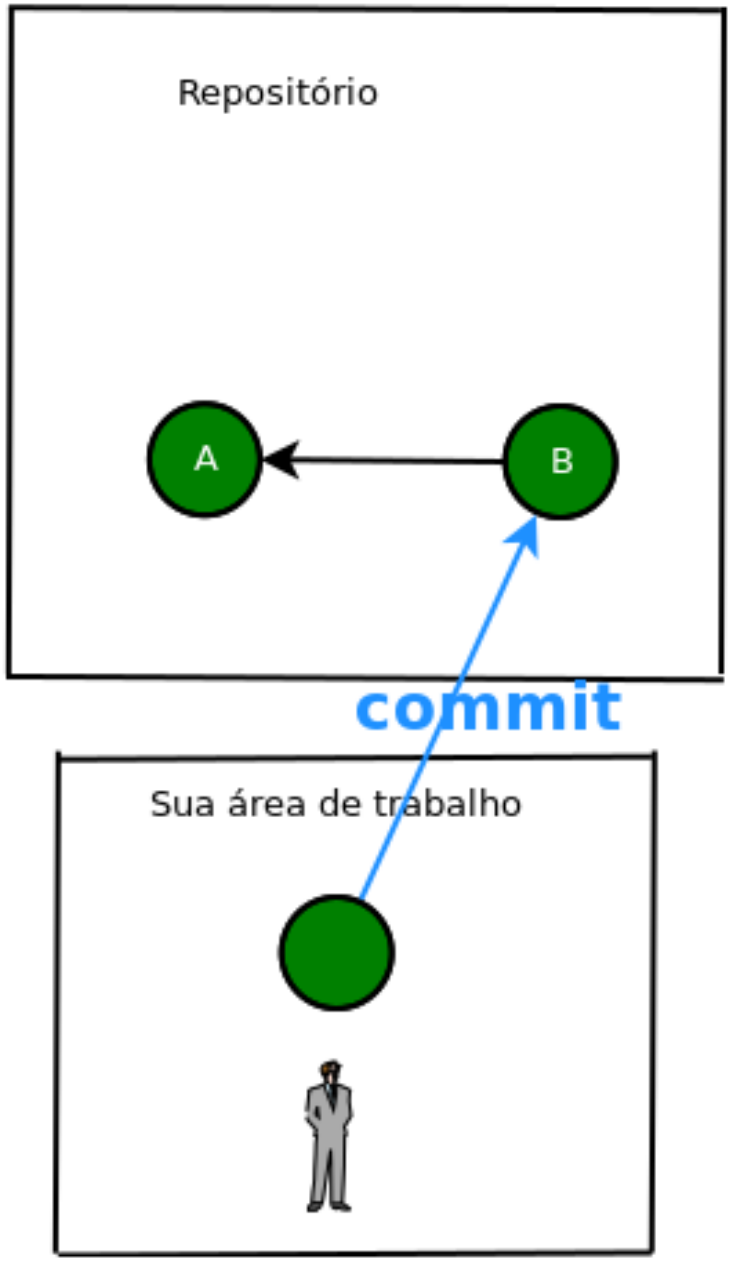
No grafo a direita, esse seria o círculo **A**



COMMIT

Após isso:

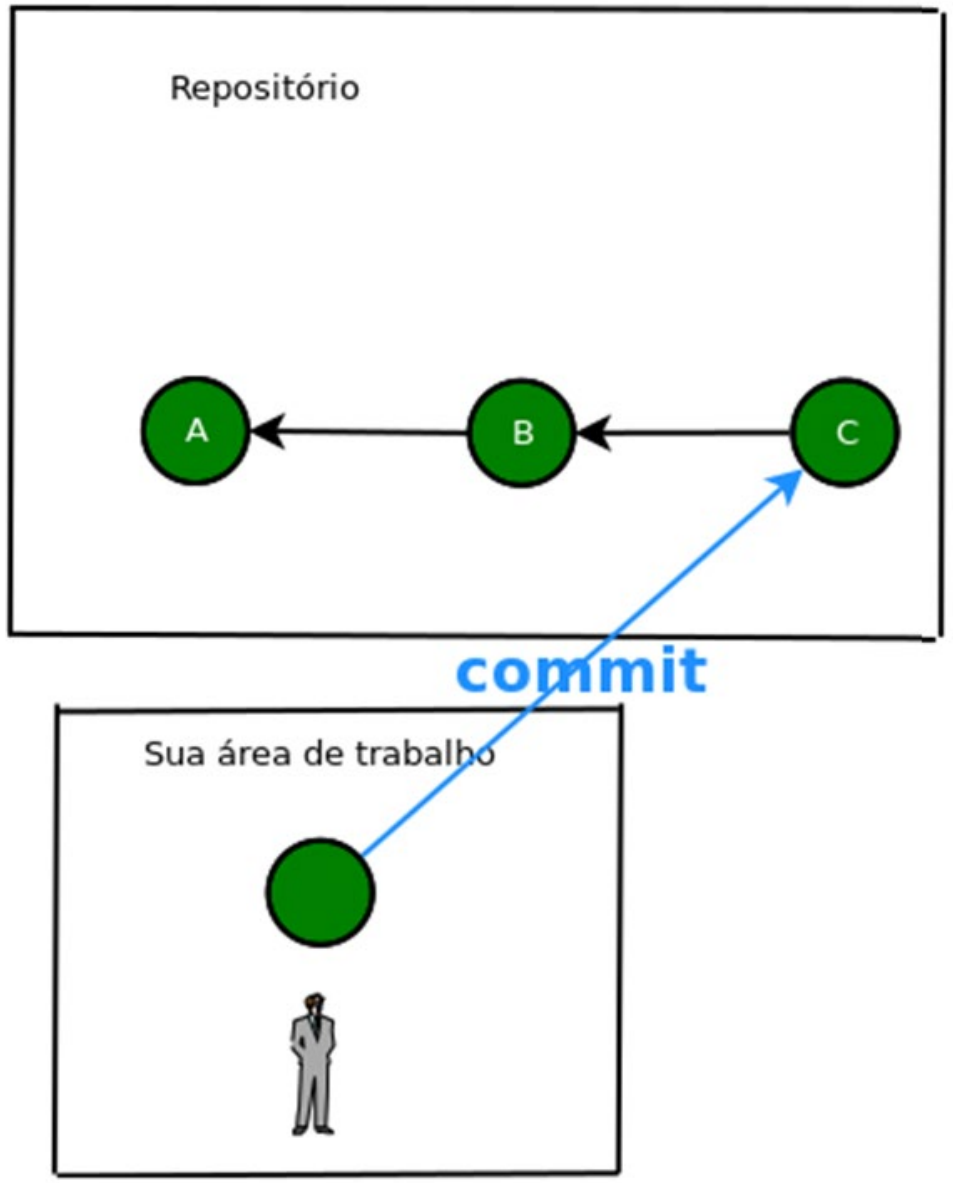
- você continua seu desenvolvimento criando uma parte do ***backend*** de sua aplicação,
- novamente você testa, e ao se dar por satisfeito realiza um ***commit*** no seu sistema de controle de versão, criando agora o estagio B:



COMMIT

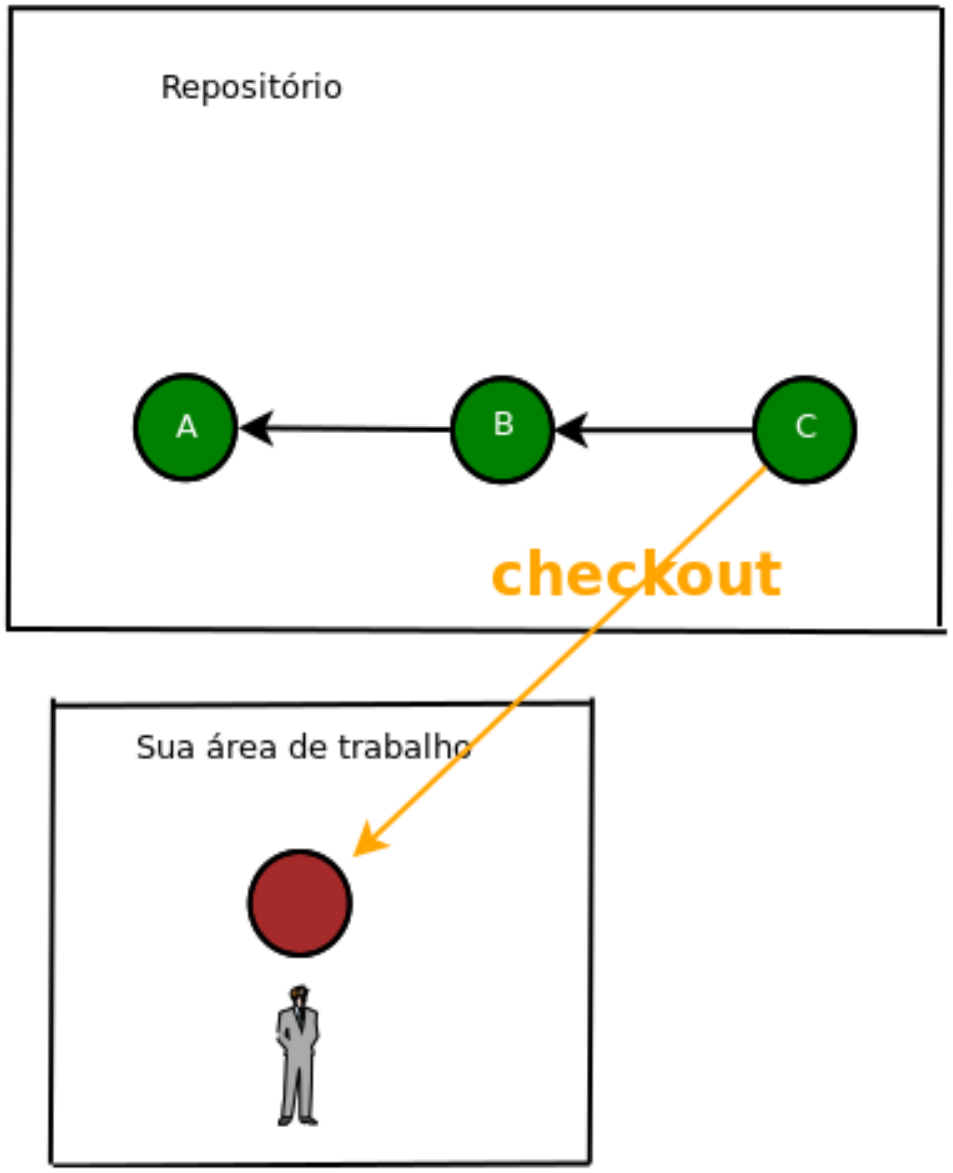
Você segue seu desenvolvimento, mas agora você recebe uma dica de como melhorar o acesso ao banco de dados e você deseja testá-la, mas, sem mexer na sua linha original de desenvolvimento.

- primeiro você salva a linha de desenvolvimento atual realizando um ***commit***,



COMMIT

- Depois você realiza uma operação chamada ***checkout*** e cria uma nova linha de desenvolvimento:
- Realizar um ***commit*** para ***checkout***



COMMIT

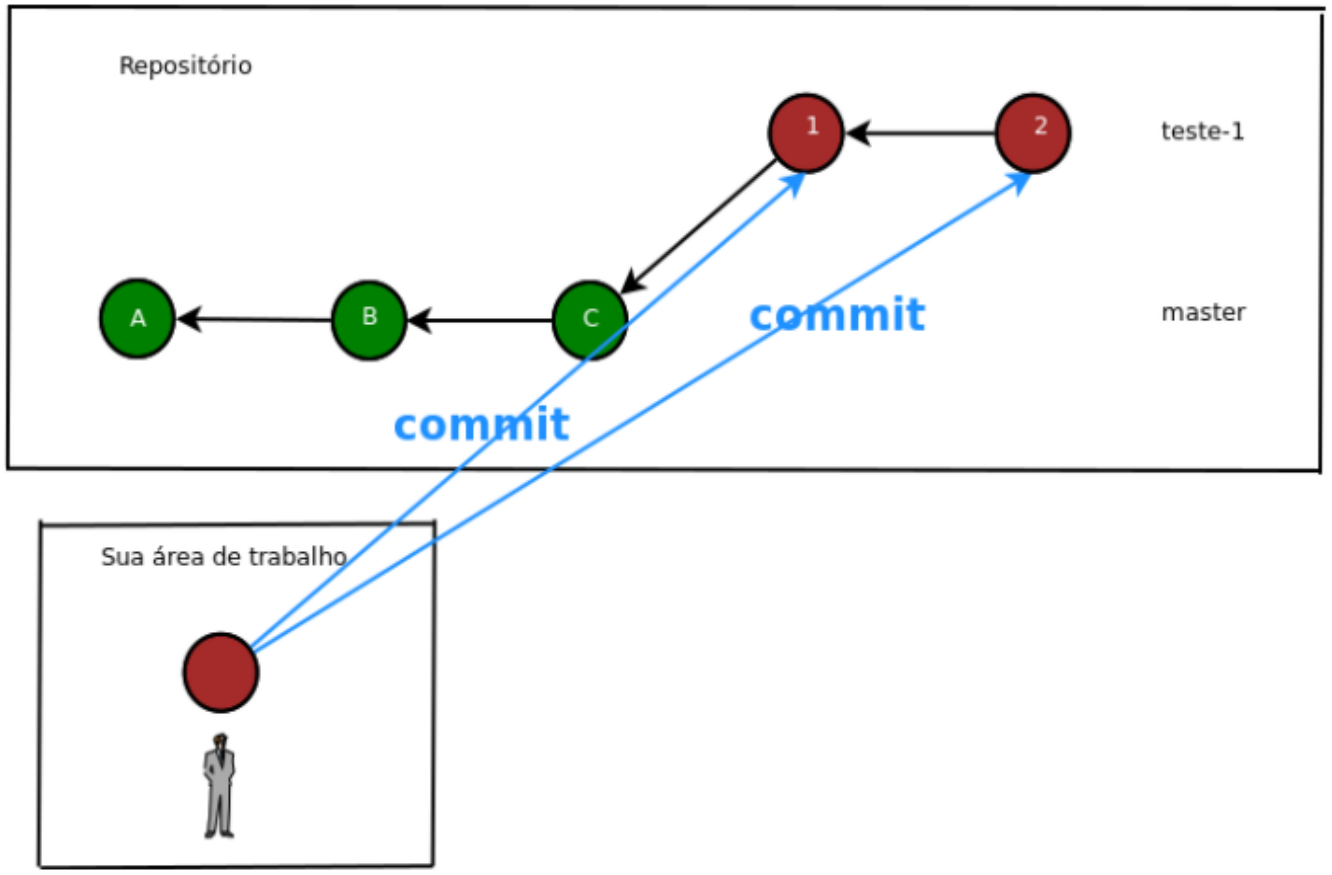
A cada linha de desenvolvimento é dado o nome de ***branch*** (Versionamento de um repositório de código fonte), a linha principal é normalmente chamada de ***branch master***

- apesar se poder alterar esse nome, não é recomendado, uma vez que muitos sistemas de controle de versão usam esse nome como padrão em algumas operações.
- os demais ***branchs*** podem ser nomeadas de acordo com as necessidades do projeto.

COMMIT

Suponha que temos duas **branchs: master e teste-1.**

O **branch teste-1** está vazio pois não foi realizado nenhum **commit** ainda.



Agora suponha que você testou a nova linha de desenvolvimento, e realizou dois **commits** criando assim o círculo 1 e 2 ou estágio 1 e 2 desse **branch**:

COMMIT

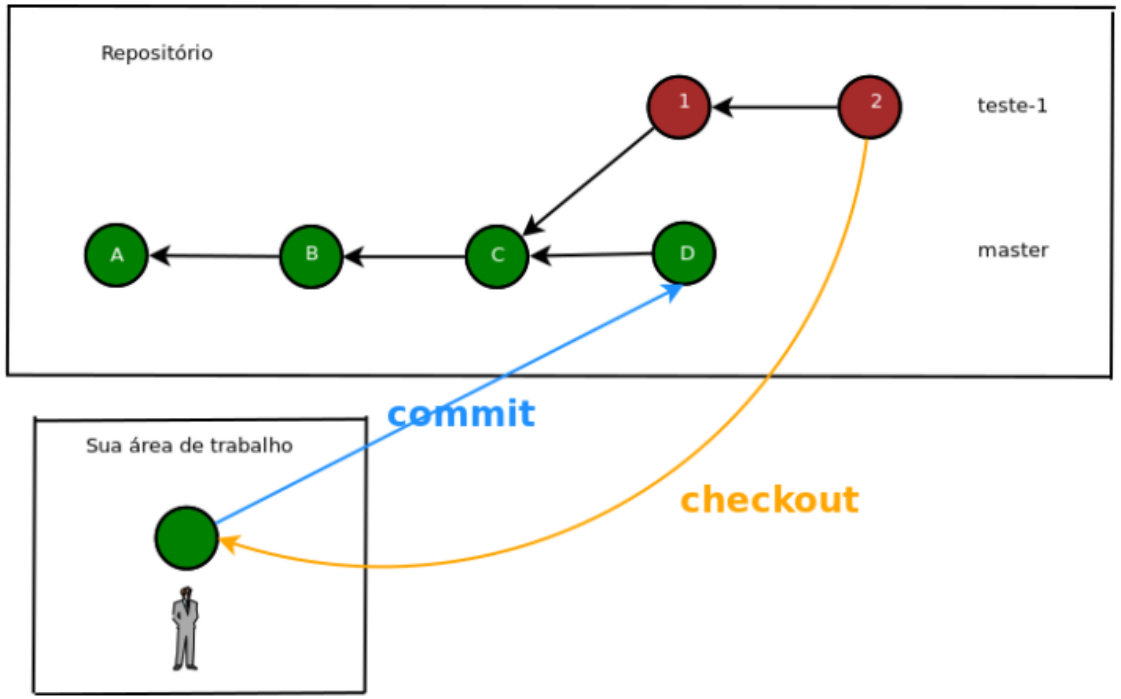
Suponha agora que surgiram algumas dúvidas a respeito da nova linha de desenvolvimento e você quer voltar à antiga linha.

COMO PROCEDER?

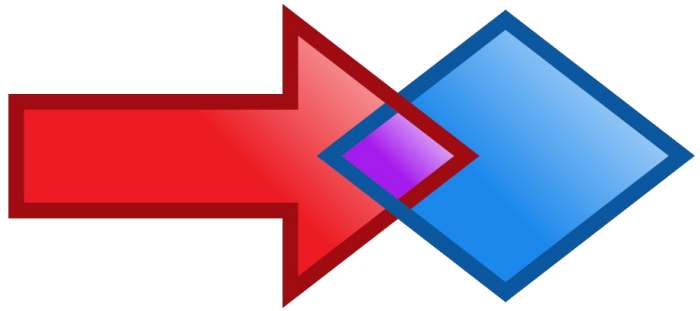
COMMIT

- Sem problemas!!
  - basta dar um novo **checkout** para você retornar ao **branch master**.

- Realizado o **checkout**
  - basta voltar a trabalhar no seu **branch master**
  - realiza um novo **commit**
  - como resultado obtemos a figura original do nosso repositório:



MERGE



O que é **merge**?

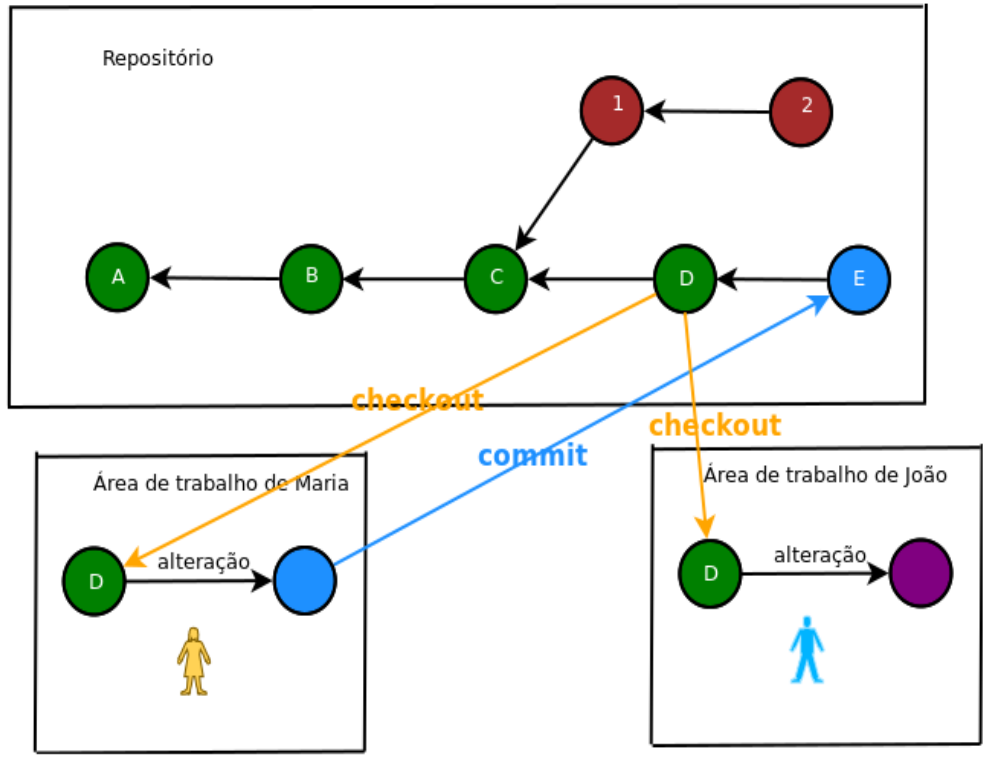
Merge significa fusão, integração.

- Trata-se de uma operação que permite que você pegue as linhas de desenvolvimento independentes e as unifique em um único branch.
- O **merge** é necessário quando mais de um desenvolvedor está trabalhando no mesmo projeto.

MERGE

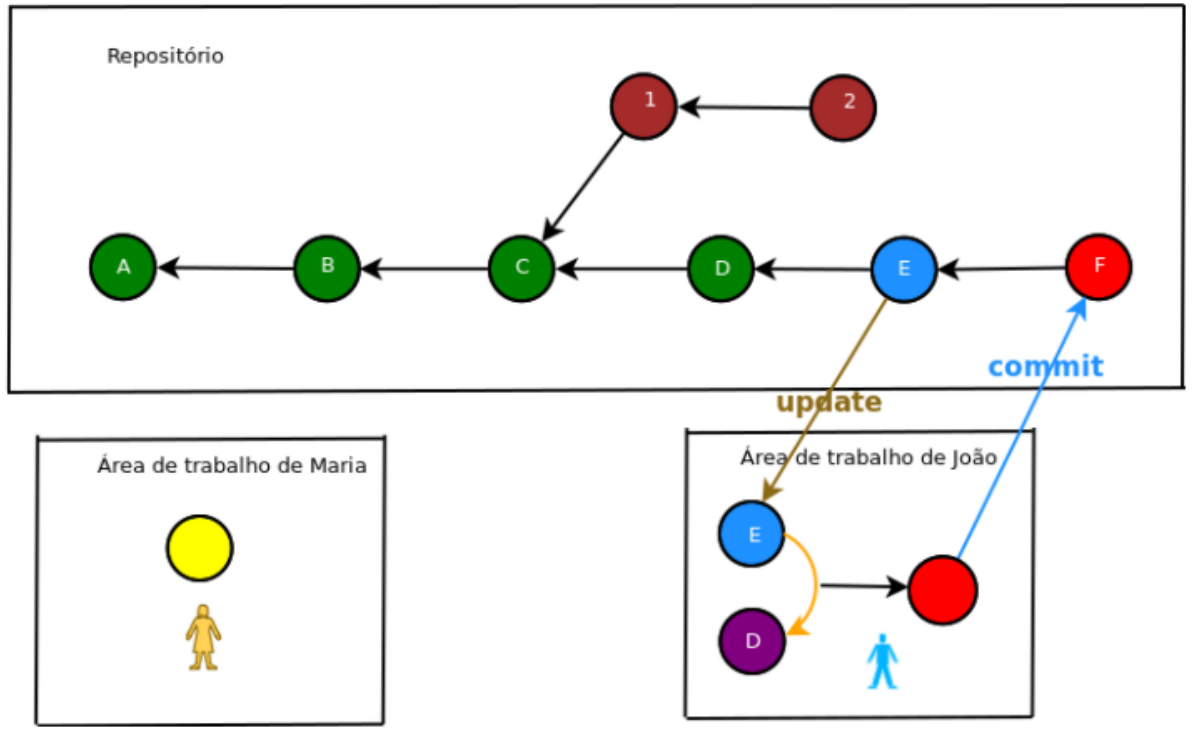
Exemplo: Suponha que João e Maria são desenvolvedores e ambos trabalham em um mesmo projeto.

- Para isso, primeiro os dois precisam realizar uma operação de **checkout** no repositório atual o que irá trazer para suas áreas de trabalho somente uma parte do repositório.
- Feitos os respectivos **checkouts** João e Maria agora tem uma parte do repositório para trabalhar e por acaso ambos vão trabalhar no mesmo estágio.
- Maria sendo mais rápida que João terminou suas revisões primeiro e faz um ***commit*** para o repositório



MERGE

- Quando João for realizar o seu commit esse será recusado pelo sistema de controle de versão avisando a João que Maria já modificou o repositório.
- João tem então que fazer um update na sua área de trabalho. Ao fazer isso, se não existir nenhum problema, o sistema de controle de versão automaticamente mescla qualquer arquivo alterado por Maria com as mudanças de João, isso é denominado MERGE.
- João então testa suas alterações estando tudo certo ele agora faz o seu commit.
- Enquanto isso Maria continua trabalhando com outras alterações na sua área de trabalho.



Referência Bibliográfica

- **Pro Git**
  Scott Chacon e Ben Strub
  Apress, 2 Edição, 2014.

- **Como funcionam os sistemas de controle de versão**
  Francisco Cunha Neto
  Disponível em: https://cadernoscicomp.com.br/tutorial/como-funcionam-os-sistemas-de-controle-de-versao-scv/

- **Git Merge (Atlasian Bitbucket)**
  Disponível em: https://www.atlassian.com/br/git/tutorials/using-branches/git-merge

Obrigado!