CONTROLE DE VERSÃO

Por Sediane Carmem Lunardi Hernandes

Agenda

- O que é um sistema de controle de versões?
- Ferramentas para o controle de versões
- Operações em controle de versões
- Tipos de repositórios
- Características em controle de versões
- Tipos de versionamento
- A ferramenta Git
- Características
- Principais termos

Controle de versão

- Um sistema de controle de versão, também conhecido como VCS (*Version Control System*) ou SCM (*Source Code Management*) possui “como finalidade gerenciar várias versões no desenvolvimento de um documento qualquer.” (MONTEIRO et. all., 2021, p. 23)
- Em um projeto, possibilita:
  - Acompanhar o histórico de desenvolvimento
  - Atividades paralelas
  - Customização de determinada versão
    - Sem necessidade de alterar o projeto principal ou recuperar versão anterior

Controle de versão

- Importante!
  - Arquivos de um projeto são armazenados em um repositório
    - O histórico de suas versões é registrado
    - Programadores podem:
      - acessar e recuperar a última versão disponível
      - realizar uma cópia local (para alterações)
      - submeter cada alteração ao repositório
      - recuperar as atualizações feitas por outros membros da equipe

Controle de versão

- Suporta o desenvolvimento de algumas formas:
  1) **Registrando o histórico**: armazena toda a evolução do projeto, ou seja, toda alteração realizada é registrada no repositório (identificação de autor, data e origem das alterações). É possível a reconstrução de determinada revisão específica do código-fonte, sempre que necessário.
  2) **Colaborando concorrentemente**: permite que mais de um programador realize alterações em paralelo sobre um mesmo código-fonte, sem sobrescrever as modificações de outro membro da equipe.
  3) **Variações no projeto**: proporciona a manutenção de versões diferentes de evolução do mesmo projeto. Ou seja, a versão 1.0 é a oficial, enquanto se prepara a versão 2.0.

Controles de versões

- Os controles de versões são compostos por duas partes:
  - O **repositório** (servidor)
    - armazena todo o histórico de ajustes do projeto, registrando todas as alterações realizadas nos itens versionados;
  - A **estação de trabalho**
    - possui uma cópia dos arquivos do projeto vinculada ao servidor para a identificação de possíveis modificações. Cada estação de trabalho é considerada individual e isolada das demais

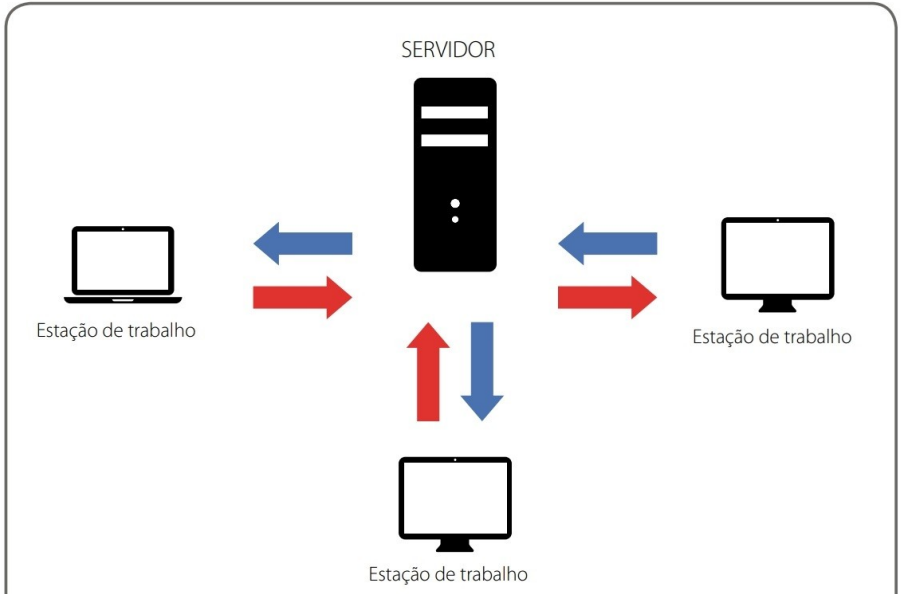
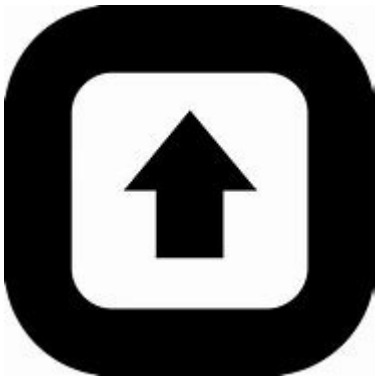


Figura 1. Arquitetura básica do controle de versão.

Controle de versões

- O processo de sincronização entre a **estação de trabalho** e o **repositório** é realizado por meio dos comandos `commit` e `update`.

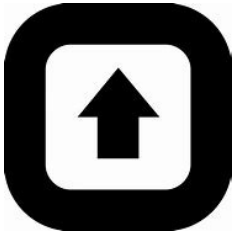


`commit`

`update`

Controle de versões

- O comando `commit` **submete** um pacote com as modificações feitas pela estação de trabalho (origem) ao repositório (destino).



- O comando `update` realiza o processo inverso, ou seja, disponibiliza as alterações submetidas pelas demais estações de trabalho ao repositório (origem) para a estação de trabalho (destino), que deseja atualizar o projeto.

Controle de versões

- Todo `commit` cria uma **revisão** no servidor
  - A **revisão** (fotografia) contém:
    - as alterações (arquivos e diretórios);
    - a data, e;
    - o usuário responsável

  O **conjunto dessas revisões** é o histórico de alterações do projeto

Ferramentas para o controle de versões

- Gratuitas
  - Git
  - Redmine
  - Subversion
  - Mercurial
  - Darcs
  - Bazaar
- Proprietárias
  - IBM Rational ClearCase
  - Microsoft Visual SourceSafe

Operações em controle de versões

- Para compreender melhor o funcionamento dos sistemas de controle de versão, é importante conhecer as principais operações que os envolvem:
  1) `Commit` (*Checkin*): **criação** de uma nova versão do projeto.
  2) `Checkout`: **recuperação** de uma versão específica do projeto ou arquivo.
  3) `Revert`: possibilita ao desenvolvedor **descartar** as mudanças realizadas em estação local, recuperando a mesma versão do repositório.
  4) `Diff`: garante a possibilidade de **comparação** do arquivo na estação local com qualquer outra versão do repositório.
  5) `Delete`: permite a **exclusão** de um arquivo do repositório. Quando as demais estações de trabalho realizarem um `update`, o arquivo será efetivamente excluído do repositório.
  6) `Lock`: possibilita o **travamento** de determinado arquivo, de forma que nenhum outro usuário o modifique.

Tipos de repositórios

Repositório de versões: realiza o armazenamento de todas as versões dos arquivos sob o controle versões.

1) **Repositório de versões centralizado**: cada estação de trabalho local contém apenas uma versão específica da árvore de versões do repositório central, e todos os usuários realizam as operações de controle de versões nesse repositório.

2) **Repositório de versões distribuído**: cada estação local possui um repositório acoplado, de modo que o usuário possui um repositório próprio para realizar o controle de versões. As operações realizadas sobre os arquivos são feitas no repositório local do usuário, e operações específicas dos repositórios distribuídos são utilizadas para sincronizar repositórios diferentes.

Árvore de revisões ou de versões: estrutura lógica que mapeia todas as versões armazenadas no repositório para determinado arquivo ou conjunto de arquivos.

Tipos de versionamento

A) Versionamento centralizado

- Arquitetura cliente servidor
  - único servidor central
  - várias estações de trabalho
- Estações de trabalho precisam consultar o servidor para a comunicação
- Vantagens:
  - Maior controle do projeto
  - Segurança
- Desvantagens:
  - Baixa escalabilidade
  - Necessidade de conexão com a Internet

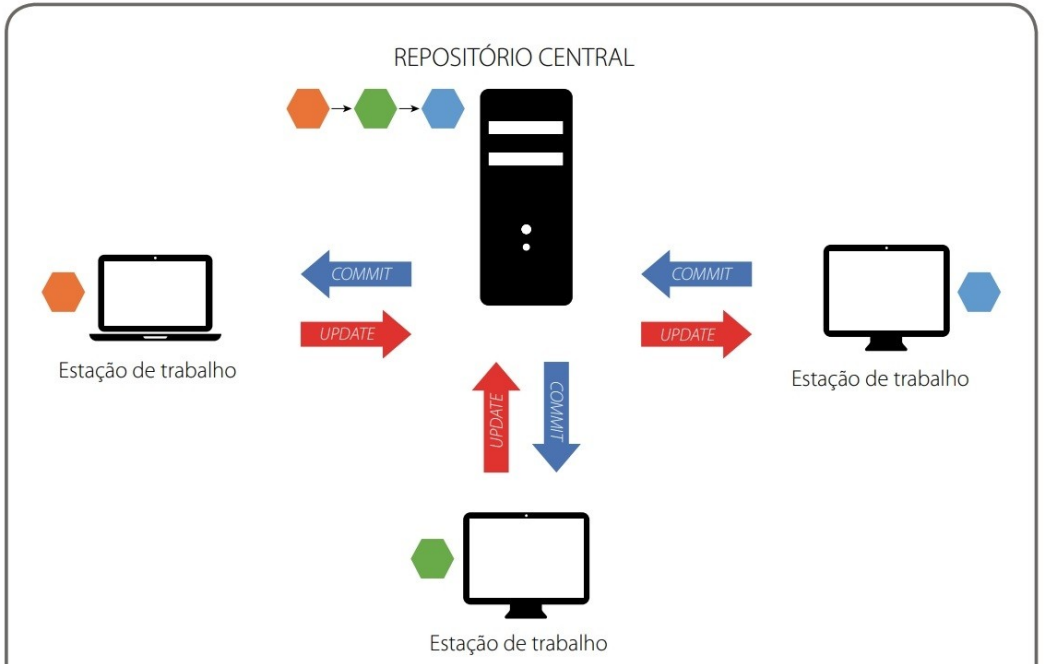


Figura 10. Sistema de controle de versão centralizado.

Tipos de versionamento

B) Versionamento distribuído

- Possui vários repositórios autônomos e independentes
  - Cada repositório tem sua própria estação de trabalho acoplada
    - Operações `commit` e `update` ocorrem localmente, bem como `checkin` e `checkout`
  - As estações podem se comunicar entre si
  - Sistema oferece servidor remoto para que o projeto seja hospedado

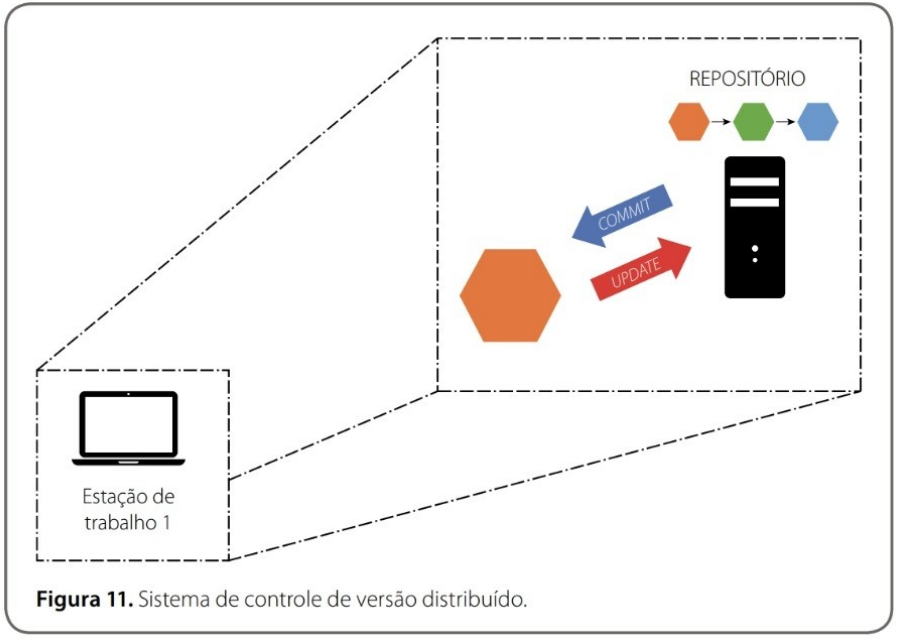


Figura 11. Sistema de controle de versão distribuído.

Tipos de versionamento

B) Versionamento distribuído (cont.)

- O processo de comunicação entre o servidor principal e as estações de trabalho funciona por meio de duas operações:
  - uma para atualizar (puxar - `pull`); e;
  - uma para mesclar o projeto (empurrar – `push`)
- Vantagens:
  - Replicação do repositório
  - Rapidez e autonomia (alterações *off-line*)
- Desvantagens:
  - Complexidade do fluxo de trabalho
  - Dificuldade de bloqueio de arquivos específicos

A ferramenta Git

- O Git é um sistema de controle de versão com controle de código-fonte distribuído
- Criado por Linus Torvalds para o desenvolvimento do kernel Linux
- O Git é livre e de código aberto

Termos básicos

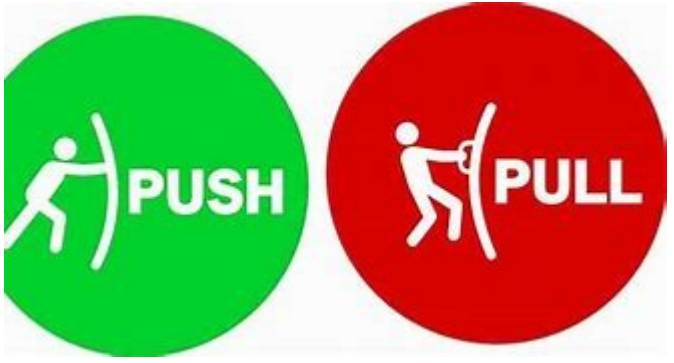
- Características:
  - Restauração de porções de código removido ou modificado
  - Organização do código fonte pela equipe de desenvolvimento
  - Criação de históricos de funcionalidades
  - Backup do código utilizado

Quadro 1. Principais termos do Git

Termo	Descrição
Repositório	O repositório é o **diretório onde os arquivos do projeto ficam armazenados**, também denominado **diretório Git**, o qual pode estar presente de forma local ou remota (em outro servidor; CHACON; STRAUB, 2014).
Commit	O *commit* é o **ponto no histórico do projeto** que indica um conjunto de modificações realizadas em um ou mais arquivos do projeto naquele momento (CHACON; STRAUB, 2014). Além disso, uma descrição é utilizada para identificar as alterações realizadas nesse determinado ponto.
Diretório de trabalho	Diretório onde **os arquivos de uma determinada versão do repositório** do projeto ficam disponíveis para acesso e manipulação por parte dos usuários (CHACON; STRAUB, 2014). Esse conteúdo é, na verdade, uma simples cópia dos arquivos contidos no **diretório Git**.

Operações estação local-servidor

- As máquinas passam a se comunicar com o servidor para a atualização dos arquivos e a versão do projeto por meio das operações
  - `push` (realizar o envio dos arquivos locais para a versão do projeto no servidor remoto);
  - `pull` (baixar os arquivos do servidor na máquina local e atualizar com os arquivos existentes).



A ferramenta Git

- Importante saber:
  - Para o gerenciamento dos repositórios na web, os desenvolvedores utilizam, em sua maioria, o `github` ou o `bitbucket`.

- Foco da disciplina em relação ao Git é a instalação, via terminal do Git, e uso de alguns comandos básicos.

Referências

- MONTEIRO, Eduarda R.; CERQUEIRA, Marcos V B.; SERPA, Matheus da S.; et al. **DevOps**. Porto Alegre: Grupo A, 2021. E-book. ISBN 9786556901725. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786556901725/. Acesso em: 12 mai. 2023.
- CHACON, Scott; STRAUB, Ben. **Pro Git**. Appress, 2ª. ed. Disponível por www em: https://git-scm.com/book/en/v2. Acesso em 12 de maio de 2023.