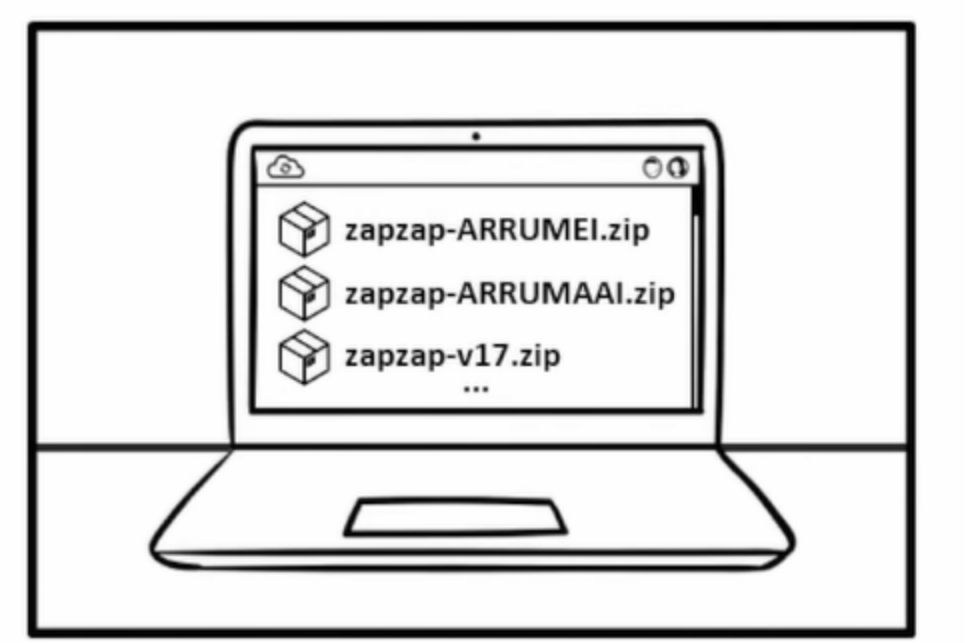
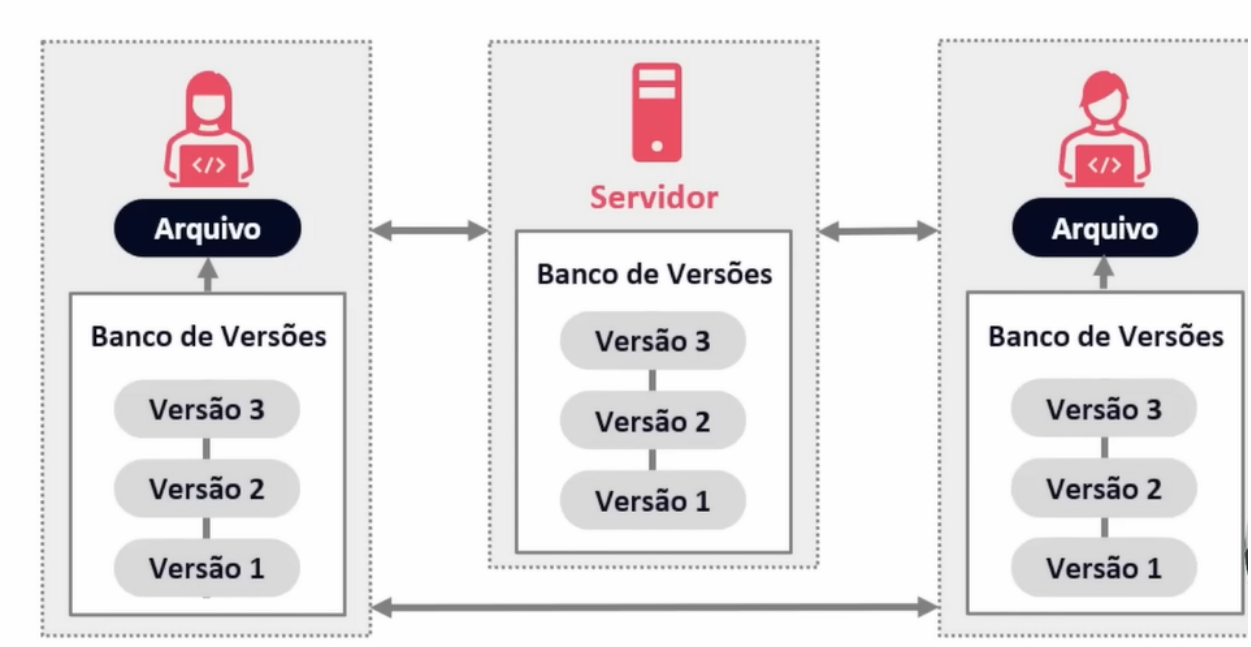
Versionamento de Código



- controle de versões de um arquivo ao longo do tempo
  - Registra o histórico de atualizações de um arquivo
  - Gerencia quais foram as alterações, a data, autor, etc
  - Organização, controle e segurança
- Sistema de Versão Distribuído:



- Clona o repositório completo, o que inclui o histórico de versões
  - Cada clone é como um backup
  - Possibilita um fluxo de trabalho flexível
  - Possibilidade de trabalhar sem conexão à rede.

O que é Git?

- Sistema de Controle de versão distribuído
  - Gratuíto e Open Source
  - Ramificações (branching) e fusões (merging) eficientes
  - Leve e Rápido.

Criando e Clonando Repositórios

- inicializar um repositório git: *git init*
- clonar um repositório: *git clone url (http ou ssh)*
  - você pode alterar o nome da pasta que irá criar o clone, por exemplo: *git clone https://clone.com repo-clonado*

    - Nesse caso, repo-clonado será o nome da pasta do repositório clonado.
- verificar quais repositórios remotos o git local está contectado: *git remote -v*
- criar uma nova ***conexão*** com um repositório remoto: *git remote add origin url (http ou ssh)*
- Também é possível clonar apenas uma branch do repositório remoto: *git clone URL — branch develop —single-branch.*
    - Caso você não indique a branch, o clone baixa apenas o conteúdo da branch principal (main ou master)

Salvando alterações no repositório local

- verificar as atualizações do repositório local com o repositório remoto: *git status*
- criar um arquivo readme: *trouch README.md*
- Untracked files: arquivos que estã na sua máquina, mas não foram commitados pelo git
    - adicionar à área de preparo: *git add .*
    - ou *git add nome_do_file*
- indicação de site editor de markdown: *readme.so*
- commitar os arquivos - salvar no git: *git commit -m “commit inicial”*
- histórico de commits: *git log*
- adicionando arquivos ao gitignore: *echo arquivos/ > .gitignore*
    - nesse caso, a pasta “arquivos” não estará sendo rastreada pelo git.

Desfazendo alterações no repositório local

- remover a inicialização do git: *rm -rf .git*
- restaurar a última versão anterior de um arquivo: *git restore README.md*
- alterar a mensagem de um commit: *git commit —amend -m “nova mensagem alterada”*
    - verifica se foi realmente alterado com o *git log*

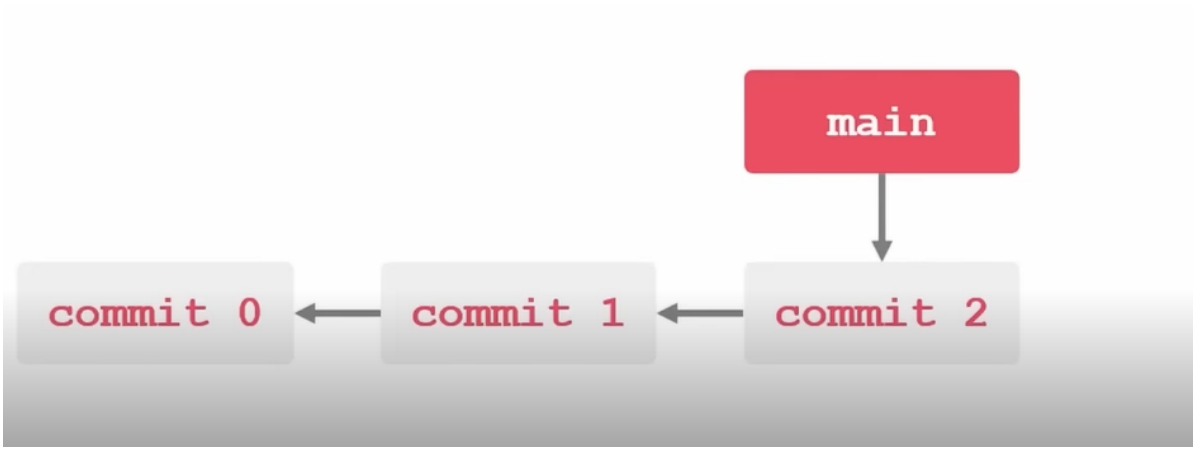
- Desfazer um commit e voltar para o anterior:
  - *git reset —soft [hash do commit a ser desfeito]*
  - *git reset —mixed [hash do commit a ser desfeito]*
  - *git reset —hard [hash do commit a ser desfeito]*
- visualizando alterações nos commit de forma mais precisa: *git reflog*
- remover arquivos da área de preparo: *git restore —staged pasta/arquivo*

Enviando e baixando alterações com o repositório remoto

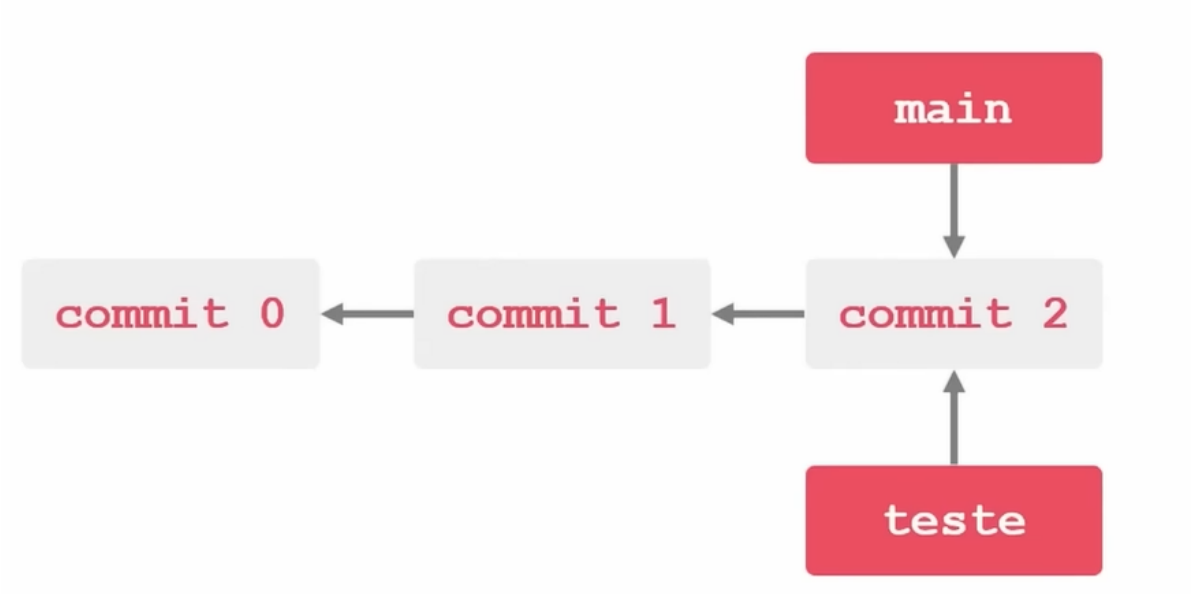
- enviar alterações para o repositório remoto: *git push -u origin main*
- para abrir o editor do visual studio code no github basta apertar o atalho . do teclado.
- baixar alterações do repositório remoto para o repositório local: *git pull*

Trabalhando com Branchs

- É uma ramificação do seu projeto
- É um ponteiro móvel para um commit no histórico do repositório



- Quando você cria uma nova branch a partir de outra existente, a nova se inicia apontando para o mesmo commit da branch que estava quando criada.



- criar e mudar o “checkout” para a nova branch: git *checkout -b teste*
- mesclar duas branchs: *git checkout main* na branch que deseja receber as alterações
  - *git merge teste*
  - agora a branch main está atualizada com a branch teste
- lista todas as branchs do repositório: *git branch*
- excluir uma branch local: *git branch -d teste*

Comandos Úteis no dia a dia

- *git pull* é a junção de: *git fetch* (baixa as alterações) + *git merge* (mescla as alterações)
- arquivar uma modificação: *git stash*