TRAD2026

Treinamento em reprodutibilidade para análise de dados

Módulo I: Git

Versionamento e colaboração

LNBio

lnbr

CNPEM Centro Nacional de Pesquisa em Energia e Materiais

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

Visão geral

- Reprodutibilidade computacional
- Controle de versão
- Git: conceitos e funcionamento
- Branches e trabalho paralelo
- GitHub e colaboração
- Boas práticas
- Atividade prática: comandos git
- Mini-projeto: colaborando em equipe

Objetivos de aprendizado

- Compreender os princípios do controle de versão e sua relação com reprodutibilidade computacional
- Criar, clonar e sincronizar repositórios Git
- Registrar alterações e consultar o histórico do projeto
- Criar e gerenciar branches e merges
- Identificar e resolver conflitos de merge
- Colaborar por meio de forks e Pull Requests
- Aplicar boas práticas de versionamento em projetos científicos

Reprodutibilidade computacional

» Capacidade de reproduzir uma análise a partir do mesmo conjunto de dados, métodos, parâmetros e ambiente computacional, obtendo resultados consistentes† com os originalmente obtidos.

Git

Controle de versão

Histórico, rastreabilidade e colaboração

Containers

Controle de ambiente

Dependências, software e configurações padronizados

Nextflow

nextflow

Orquestração de pipelines

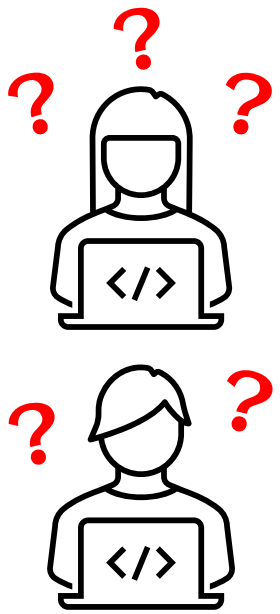
Parâmetros, tarefas e fluxo automatizados

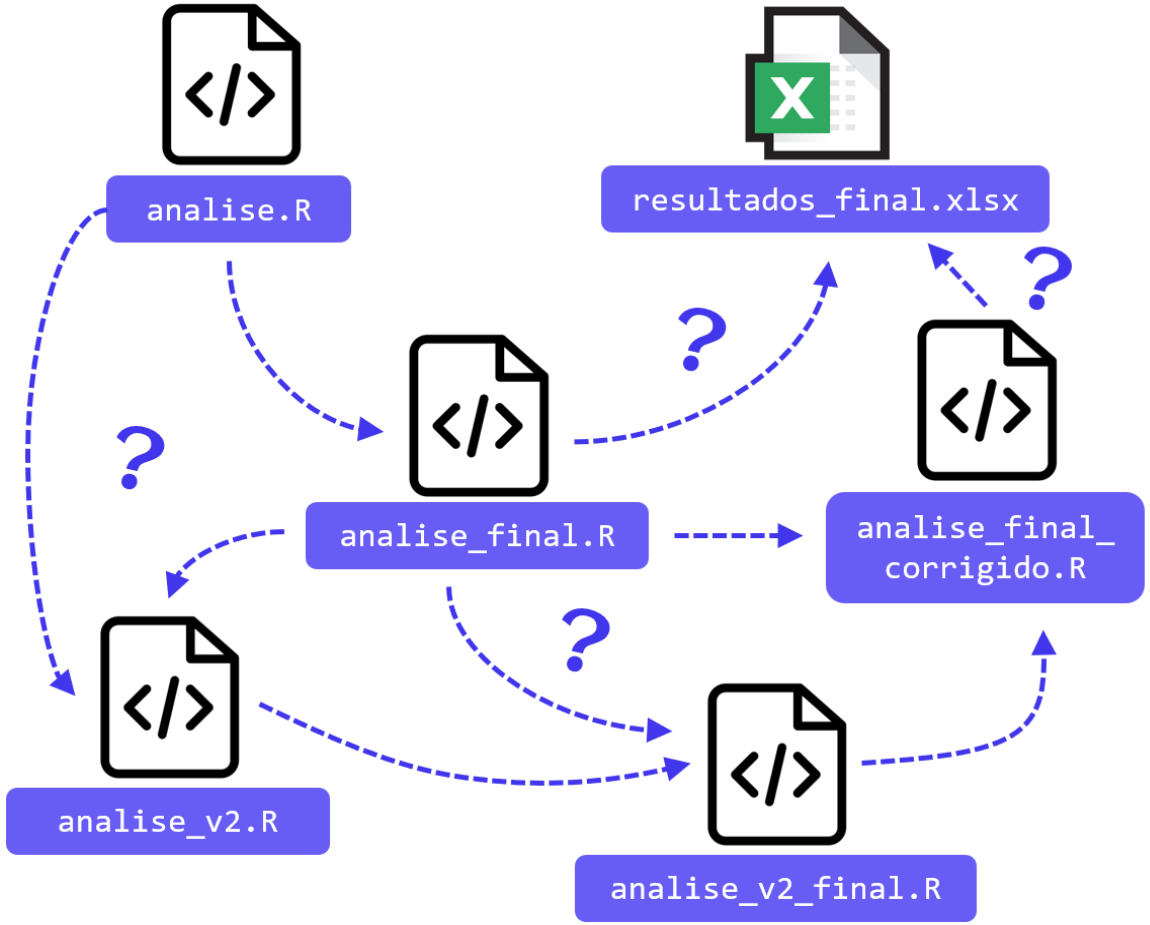
REPRODUTIBILIDADE COMPUTACIONAL

† componentes estocásticos (e.g., amostragem, inicialização aleatória, algoritmos probabilísticos, paralelismo)

Cenário comum: arquivos, versões e incertezas

» Sem controle de versão, arquivos se espalham, versões se confundem e o histórico se perde, dificultando a reprodutibilidade computacional ao longo do tempo.





Perguntas frequentes

- Qual versão gerou os resultados?
- O que mudou entre as análises?
- Quem fez as alterações?
- Como recuperar uma versão que funcionava?
- Como reproduzir em outra máquina?

O que é controle de versão?

» Controle de versão é o processo de **registrar, organizar e recuperar alterações** realizadas em arquivos ao longo do tempo, onde cada alteração é salva em um **histórico** com **autoria, data e descrição**.

Histórico completo
Acompanha todas as alterações

Rastreabilidade
Saiba quem alterou, quando e o que

Trabalho em paralelo
Desenvolva diferentes funcionalidades

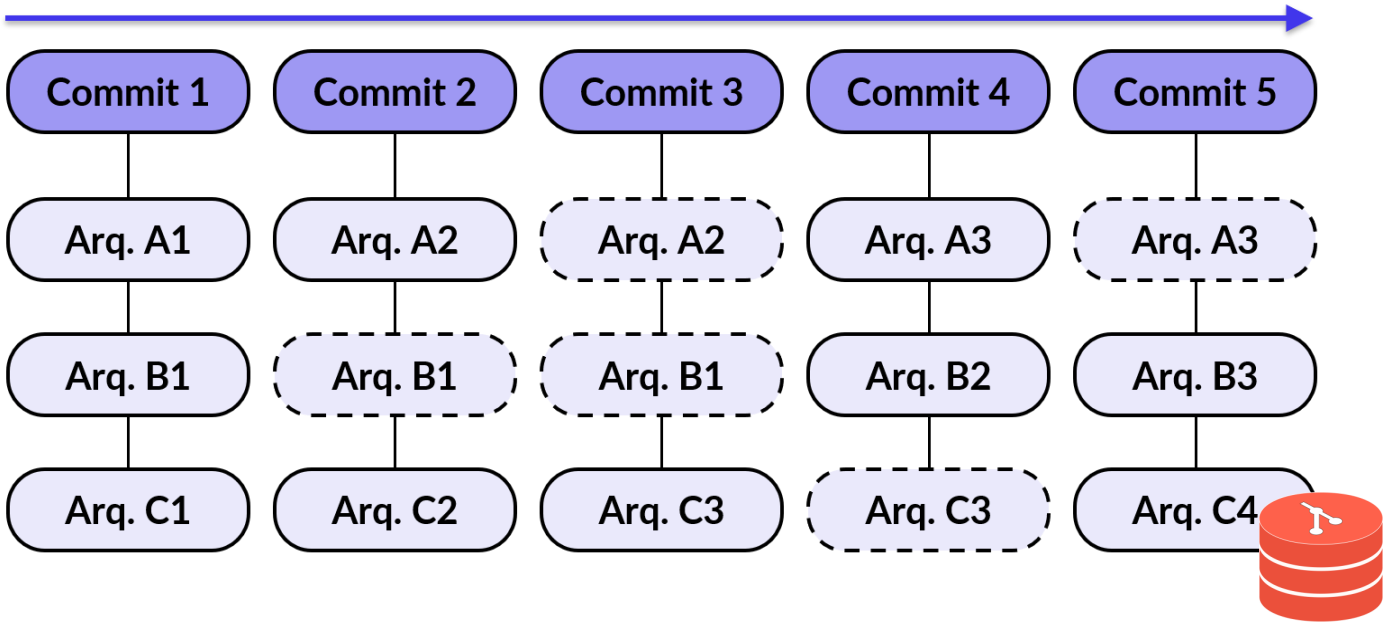
Recuperação
Volte a versões anteriores

Colaboração
Integre contribuições de diferentes pessoas

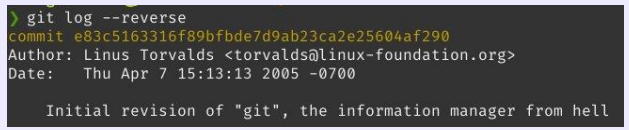
O que é git?

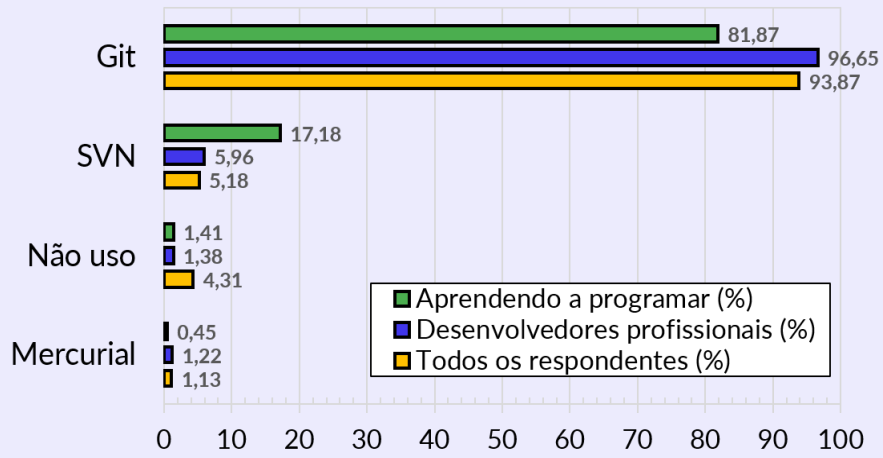
Git é um sistema de controle de versão **distribuído** e de **código aberto.**

A evolução do projeto é registrada por meio de **commits**, que funcionam como **snapshots** dos arquivos ao longo da sua história.



Criado por Linus Torvalds em 2005 para o desenvolvimento do kernel do Linux



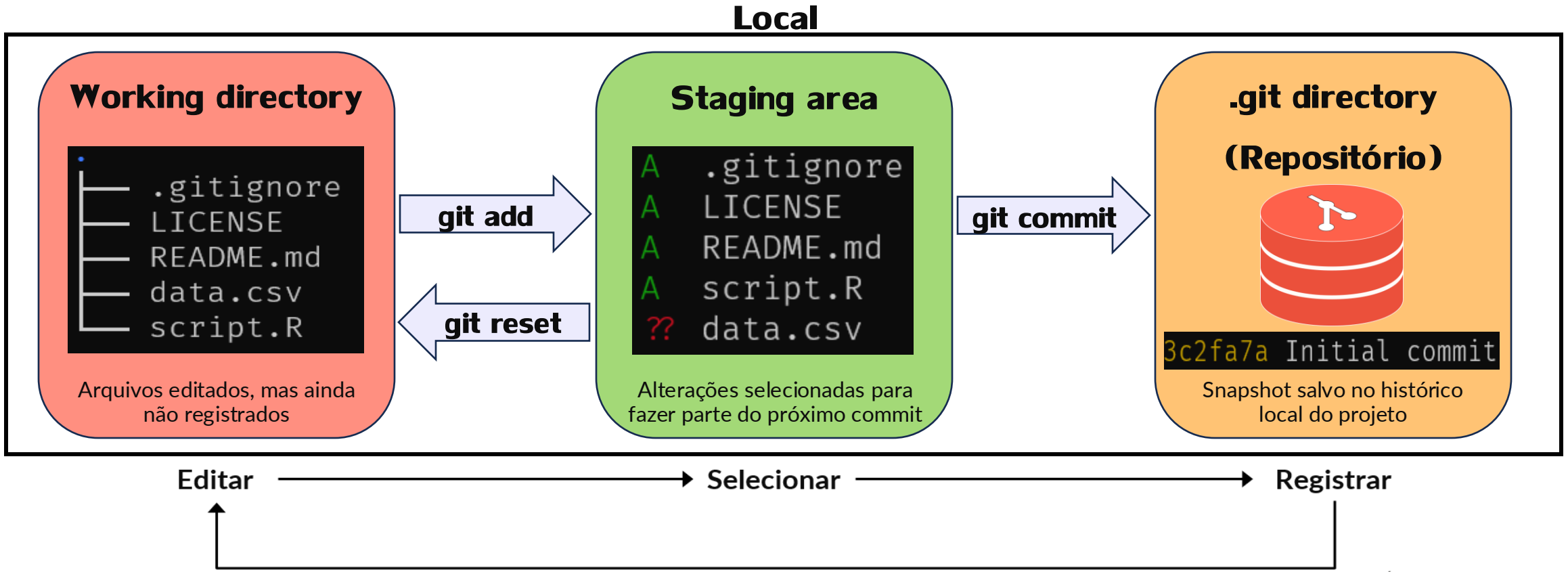


Fonte: Stack Overflow Developer Survey 2022, seção Version Control System.

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

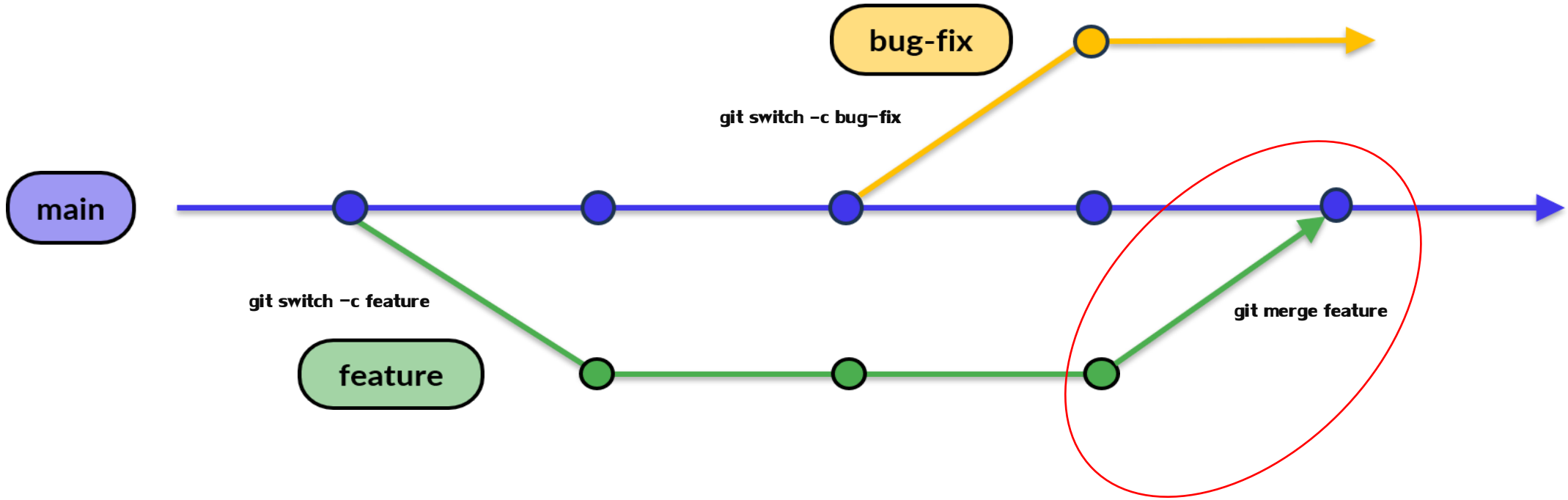
Como o git funciona?

» O Git organiza os arquivos em três estados principais, associados a três áreas de trabalho: **Working Directory**, **Staging Area** e **Repositório**.



Como trabalhar em paralelo no git?

» Uma **branch** é uma linha independente de desenvolvimento, que permite modificar parte do projeto sem afetar diretamente a versão principal (main), mantendo-a **funcional** e **estável**.



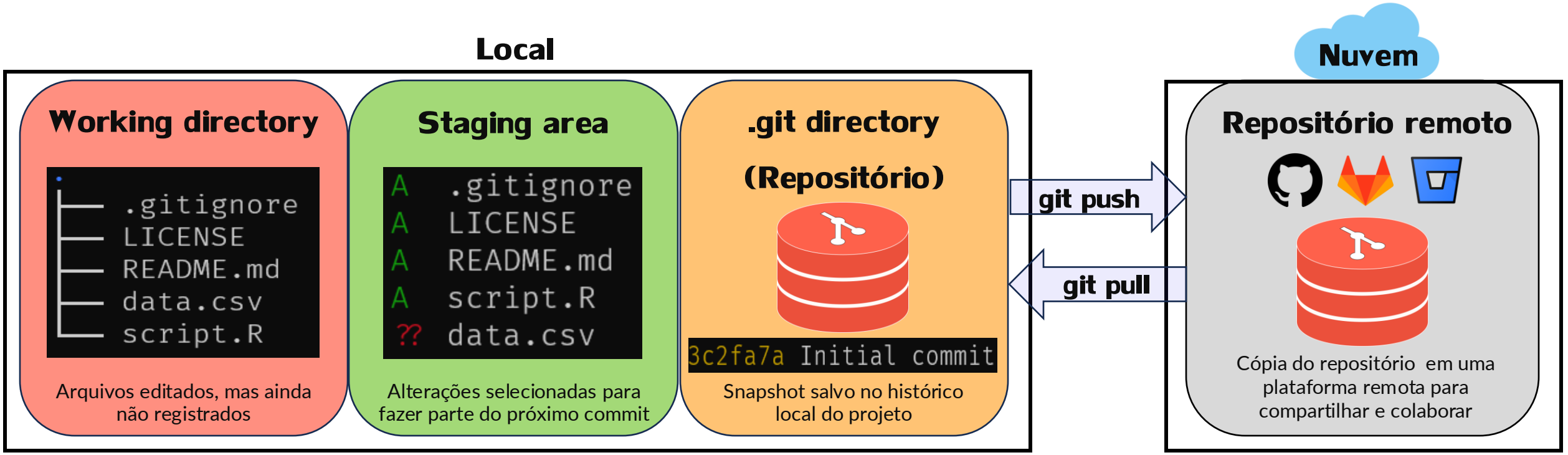
Merge conflict: quando mudanças feitas em branches diferentes afetam a mesma parte de um arquivo.

```
❯ git merge feature
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

```
<<<<<<< HEAD
resultado = "controle"
=======
resultado = "tratamento"
>>>>>>> feature
```

Como colaborar com git?

» O **git** permite criar commits, branches e merges **localmente**. Para trabalhar de forma colaborativa, usamos **repositórios remotos** hospedados em plataformas como **GitHub, GitLab e Bitbucket**, que oferecem recursos para compartilhamento, revisão, discussão e integração de mudanças.



Plataformas colaborativas para git

» **GitHub**, **GitLab** e **Bitbucket** hospedam repositórios git e adicionam recursos de **colaboração**.

GitHub

Projetos públicos e colaboração ampla

Hospedagem
github.com
Ex: github.com/cnpem

Revisão de código
Pull Requests

Automação
GitHub Actions

Comunidade
Ampla adoção em projetos de código aberto

GitLab

Projetos internos, DevOps e gestão institucional

Hospedagem
gitlab.com ou auto-hospedagem
Ex: gitlab.cnpem.br

Revisão de código
Merge Requests

Automação
CI/CD integrado com GitLab CI

Bitbucket

Integração com ecossistema Atlassian (Jira, Confluence, etc)

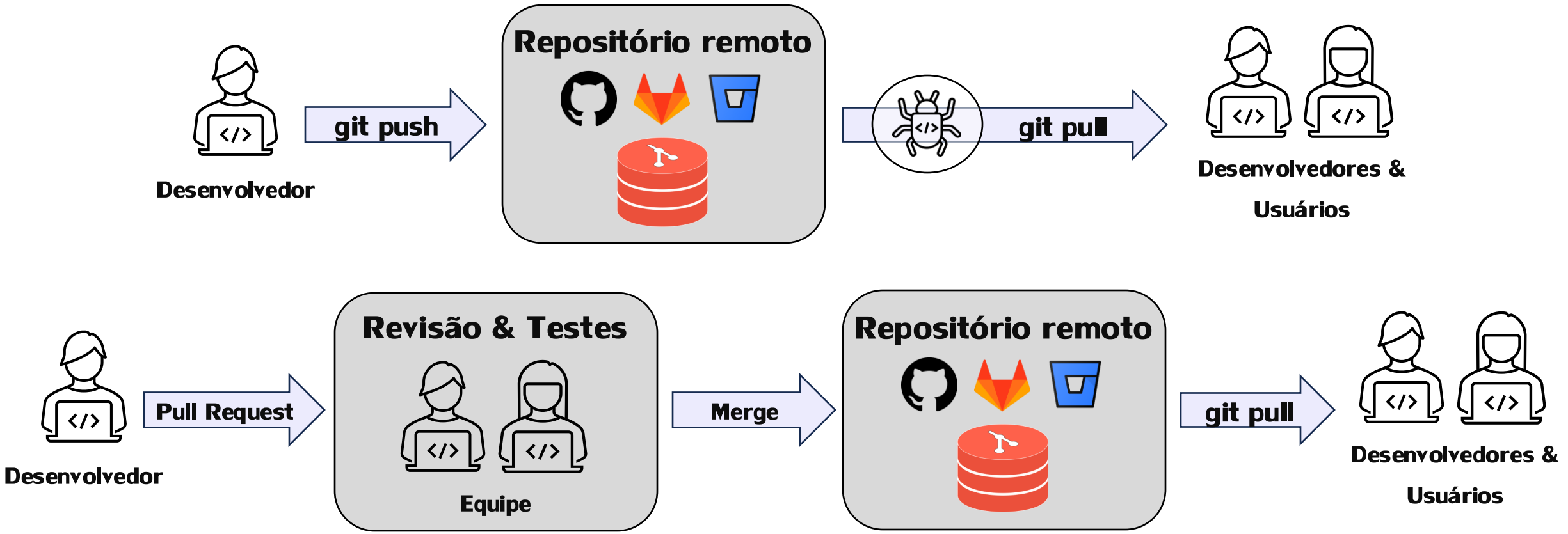
Hospedagem
Bitbucket Cloud
ou auto-hospedagem

Revisão de código
Pull Requests

Integrações
Jira, Confluence e pipelines corporativos

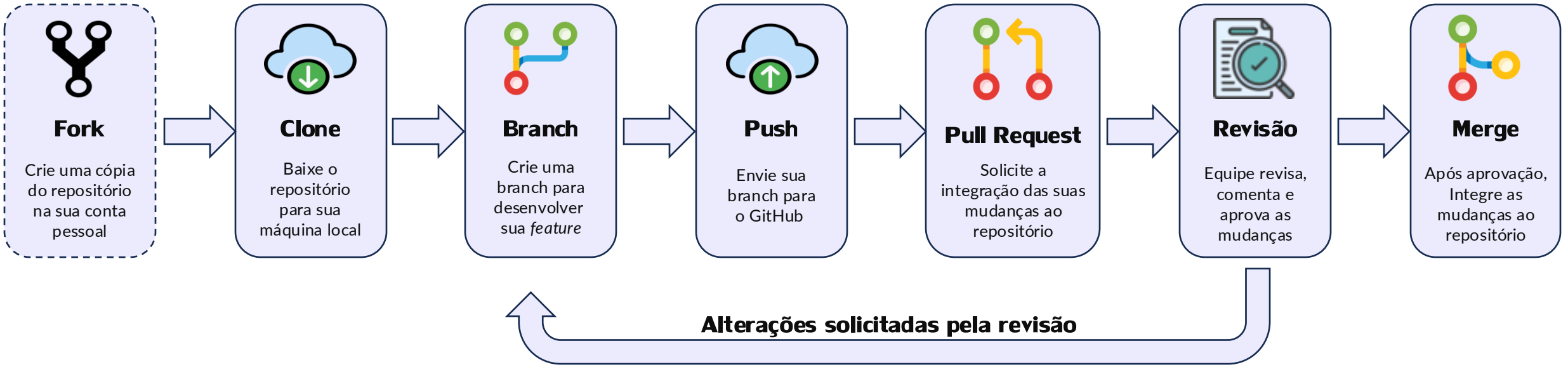
Workflow colaborativo

» É essencial revisar e testar as mudanças antes de integrá-las à branch **main** do repositório remoto



Workflow colaborativo com Git e GitHub

» Cada desenvolvedor trabalha em uma branch, envia suas contribuições e intrega ao repositório remoto após revisão



Boas práticas para Git

» Boas práticas ajudam a manter o projeto organizado, reduzem conflitos e tornam o histórico mais útil para a equipe.

Repositório

- Defina padrões e convenções por projeto
- Crie um **README.md** claro
- Configure um **.gitignore** apropriado ao seu projeto
- Defina uma estrutura clara de diretório e arquivos
- Crie **testes** unitários e de integração **automatizados**
- **Documente** demandas e discussões em **Issues**
- **Não** armazene dados brutos ou sensíveis

Branches

- Crie uma **branch** por tarefa ou feature
- Mantenha sua branch sempre **atualizada** com o **main**

Pull Request

- **Uma** mudança por Pull Request (atomicidade)
- Sempre **revise** as alterações
- Sempre **teste** o efeito das mudanças
- Aplicar squash de commits quando for apropriado para manter o histórico limpo

Commits

- Faça commits **frequentes**
- **Uma** mudança por commit
- Escreva mensagens de commit **claras** e **descritivas**
- Evite commits direto na branch **main**

OBRIGADO

João V. S. Guerra
joao.guerra@lnbio.cnpem.br
@jvsguerra

Inscreva-se para receber comunicados **sobre o CNPEM e suas unidades**

cnpem.br

Material do curso

https://github.com/cnpem/TRAD2026

Laboratório Nacional de Biociências

João V. S. Guerra (EDB)
José G. C. Pereira (EDB)
Nilson A. R. Coimbra (MAS)
Rick H. Hokama (EDB)

Laboratório Nacional de Biorrenováveis

Joaquim M. Junior
Monyque K. P. Silva

CNPEM
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO