

Politécnica
UFRJ



UFRJ
UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

Politécnica
UFRJ

Introdução ao Git

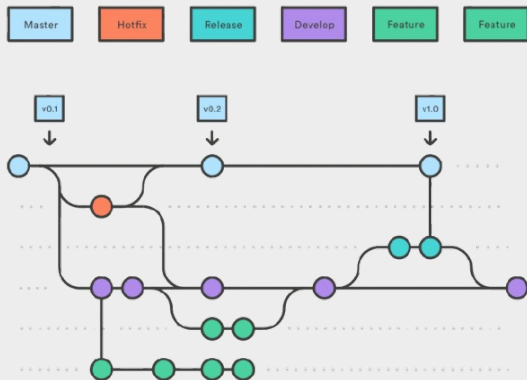
Autor: **Elizeu Rodrigues Sena** (ellizeurs@poli.ufrj.br)

11 de Setembro de 2025

O que é Git?

- ▶ Sistema de controle de versão distribuído.
- ▶ Permite acompanhar alterações no código ao longo do tempo.
- ▶ Facilita colaboração entre desenvolvedores.
- ▶ Principais benefícios:
 - ▶ Histórico de mudanças.
 - ▶ Trabalho em equipe sem sobrescrever código.
 - ▶ Possibilidade de ramificação (*branching*) e fusão (*merge*).

Branches em Git



Branch = linha independente de desenvolvimento.

Branches em Git (cont.)

- ▶ master (ou main) – Código estável, pronto para produção.
- ▶ develop – Integração das features em desenvolvimento.
- ▶ feature/* – Novas funcionalidades, criadas a partir da develop. Ex.:
feature/login
- ▶ release/* – Preparação para nova versão, criada a partir da develop. Ex.:
release/1.0.0
- ▶ hotfix/* – Correções críticas feitas diretamente a partir da master. Ex.:
hotfix/1.0.1

Comando git init

- ▶ Cria um novo repositório Git na pasta atual.
- ▶ Gera a pasta oculta .git, onde ficam os metadados.
- ▶ É o primeiro passo para versionar um projeto.

Exemplo

```
git init
```

Comando git clone

- ▶ Usado para copiar um repositório remoto para a máquina local.
- ▶ Sintaxe:

Exemplo

```
git clone https://github.com/usuario/repositorio.git
```

- ▶ Cria uma pasta com todo o histórico do projeto.
- ▶ Normalmente o primeiro passo para começar a colaborar em um projeto existente.

Comando git add

- ▶ Prepara os arquivos para serem versionados (coloca na **staging area**).
- ▶ Sintaxe:

Exemplo

```
git add arquivo.txt  
git add . (todos os arquivos modificados)
```

- ▶ Não salva nada no histórico ainda, apenas “marca” o que vai ser incluído no próximo commit.

Comando git commit

- ▶ Registra no histórico as alterações que foram adicionadas com add.
- ▶ Cada commit é como uma “foto” do projeto em um momento específico.
- ▶ Sintaxe:

Exemplo

```
git commit -m "Mensagem explicando a mudança"
```

- ▶ Dicas:
 - ▶ Escreva mensagens curtas e claras.
 - ▶ Um commit deve representar uma alteração lógica/coesa.

Comando git pull

- ▶ Atualiza o repositório local com mudanças do remoto.
- ▶ Combina dois passos:
 1. `git fetch` (baixa as alterações).
 2. `git merge` (mescla no seu branch atual).
- ▶ Sintaxe:

Exemplo

```
git pull origin main
```

- ▶ Usado para manter seu repositório sincronizado.

Comando git push

- ▶ Envia as alterações locais (commits) para o repositório remoto.
- ▶ Fluxo comum:
 1. `git add` (seleciona arquivos).
 2. `git commit` (registra alterações).
 3. `git push` (envia para o remoto).
- ▶ Sintaxe:

Exemplo

```
git push origin main
```

Comando git status

- ▶ Mostra o estado atual dos arquivos no repositório.
- ▶ Indica arquivos modificados, adicionados ou não rastreados.
- ▶ Útil para verificar antes de commitar.

Exemplo

```
git status
```

Comando git log

- ▶ Exibe o histórico de commits.
- ▶ Mostra hash, autor, data e mensagem.
- ▶ Opções úteis: -oneline, -graph.

Exemplo

```
git log -oneline -graph
```

Comando git branch

- ▶ Lista, cria ou exclui branches.
- ▶ Permite desenvolver funcionalidades isoladas.

Exemplo

```
git branch nova-feature
```

Comando git switch / git checkout

- ▶ Troca de branch ou recupera versões anteriores.
- ▶ git switch é a forma mais atual recomendada.

Exemplo

```
git switch nova-feature
```

Comando git merge

- ▶ Mescla mudanças de uma branch em outra.
- ▶ Usado geralmente para integrar features na main.
- ▶ Pode gerar conflitos que devem ser resolvidos.

Exemplo

```
git merge nova-feature
```


Comando git tag

- ▶ Marca versões específicas no histórico.
- ▶ Muito usado para releases.

Exemplo

```
git tag v1.0  
git push origin v1.0
```

Comando git stash

- ▶ Guarda alterações não finalizadas sem commitar.
- ▶ Permite trocar de branch sem perder progresso.

Exemplo

```
git stash  
git stash pop
```

Comando git revert

- ▶ Reverte um commit específico sem apagar histórico.
- ▶ Cria um novo commit desfazendo as mudanças.

Exemplo

```
git revert abc123
```

Comando git rebase

- ▶ Reorganiza o histórico aplicando commits em outro branch.
- ▶ Mantém histórico linear e mais limpo.

Exemplo

```
git rebase main
```

O que são Issues?

- ▶ Recurso disponível em plataformas como GitHub/GitLab.
- ▶ Funcionam como **tarefas** ou **tickets** que organizam o trabalho.
- ▶ Podem ser usadas para:
 - ▶ Relatar **bugs**.
 - ▶ Sugerir **melhorias**.
 - ▶ Discutir e planejar **novas funcionalidades**.
- ▶ Geralmente associadas a **branches** e **Pull Requests**.

Workflow para Resolver Issues

1. Leia atentamente a descrição da issue.
2. Crie uma branch relacionada:

Exemplo

```
git checkout -b feature/23-login-google
```

3. Faça as alterações necessárias e registre com `git commit`.
4. Envie a branch para o repositório remoto:

Exemplo

```
git push origin feature/23-login-google
```

5. Abra um Pull Request e associe à issue (closes #23).
6. Após revisão e merge, a issue é fechada automaticamente.

Gerando uma chave SSH (padrão)

- ▶ Comando básico:

Exemplo

```
ssh-keygen -t ed25519 -C "seu_email@exemplo.com"
```

- ▶ Gera dois arquivos:
 - ▶ **Privada:** ~/.ssh/id_ed25519 (não compartilhar).
 - ▶ **Pública:** ~/.ssh/id_ed25519.pub (adicionar no GitHub/GitLab).
- ▶ Testar conexão:

Exemplo

```
ssh -T git@github.com
```

Gerando uma chave SSH com nome específico

- ▶ Útil quando você tem múltiplas contas ou serviços.
- ▶ Sintaxe:

Exemplo

```
ssh-keygen -t ed25519 -C "seu_email@exemplo.com-f  
~/.ssh/id_ed25519_github
```

- ▶ Resulta em:
 - ▶ **Privada:** id_ed25519_github
 - ▶ **Pública:** id_ed25519_github.pub

Usando uma chave SSH específica no Git

- Configure o arquivo `~/.ssh/config`:

Exemplo

```
Host github.com
  HostName github.com
  User git
  IdentityFile ~/.ssh/id_ed25519_github
```

- Depois, ao clonar:

Exemplo

```
git clone git@github.com:usuario/repositorio.git
```

- O Git usará automaticamente a chave configurada para o host.

- ▶ Git é essencial para versionamento e colaboração.
- ▶ Principais comandos:
 - ▶ `clone`, `add`, `commit`, `pull`, `push`.
- ▶ Branches permitem desenvolvimento paralelo.
- ▶ Issues ajudam a organizar o trabalho em equipe.
- ▶ SSH garante autenticação segura.

Obrigado!

Politécnica
UFRJ



UFRJ
UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO