Técnicas de Programação e Análise de Algoritmos

Prof. Dr. Lucas Rodrigues Costa

Aula PP: Git - Sistema de controle de versão

lucas.costa@idp.edu.br

@lucasrodri

www.linkedin.com/in/lucas-rodri

OBJETIVOS

- Compreender os conceitos básicos de Git
- Compreender e conhecer as boas práticas de uso do Git
- Conhecendo os principais comando git
- Praticar os comandos git com interação na criação e deleção de ramos

idp

Boas práticas para desenvolvimento de software & git

idp

Boas práticas para desenvolvimento de software & git

- **Escreva mensagens claras para os commits**
  - Faça um resumo daquilo que resolveu ou desenvolveu em uma única sentença
- **Faça commits frequentemente**
  - Commits ficarão pequenos e facilitará o compartilhamento e resolução de possíveis conflitos durante as mesclagens (merges)
- **Nunca faça commit de um trabalho inacabado**
  - Divida seu trabalho em pequenos problemas, resolva-os e teste-os. Somente após isso faça um commit.
- **Faça uso de ramos (branchs)**
  - No ramo main estará somente o código validado e funcional
  - Novas funcionalidades ou correções de bugs devem ser feitas em ramos separados

Iniciando um repositório

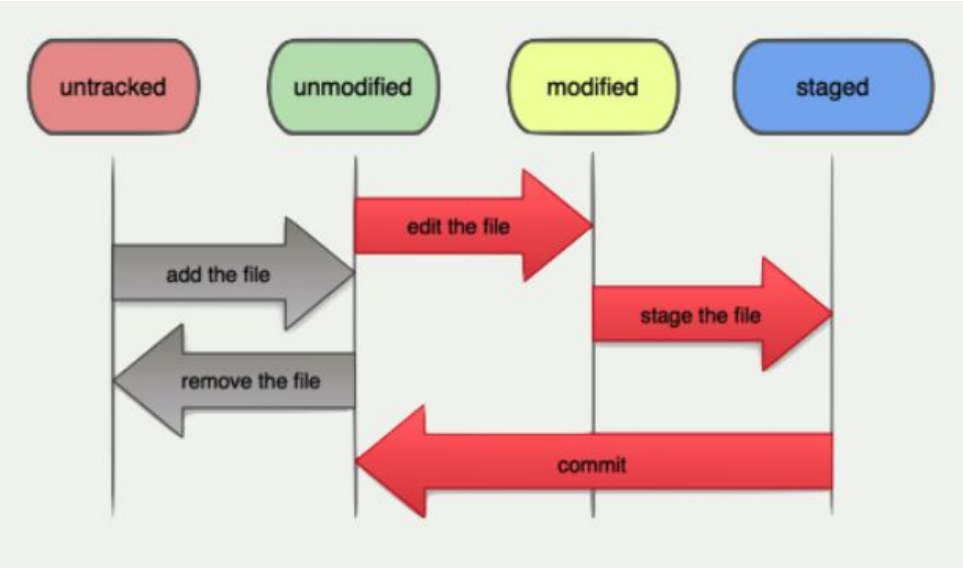
- Ao iniciar um repositório git, será criado um subdiretório .git e este armazenará os dados e metadados de todas as alterações feitas no repositório

```
mkdir projeto
cd projeto
git init
```

- Criando um arquivo, adicionando ele no git e persistindo as mudanças

```
echo "Olá mundo" >> arquivo.txt
git add arquivo.txt
git commit -m "Adicionando arquivo.txt"
```

Ciclo de vida de arquivos em um repositório git




```
#arquivo novo fica como untracked
echo "olá" >> novo.txt
# arquivo é marcado (staged) para ir para o próximo commit
git add novo.txt
# mudanças persistidas
git commit -m "adicionando arquivo"
# arquivo sai do staged para modified
echo "mundo" >> novo.txt
# marcando (staged) todos os arquivos modificados
git add .
# mudanças persistidas
git commit -m "adicionando mundo"
```

Markdown – extensão de arquivo .md

- Linguagem de marcação geralmente usada para documentar projetos no Github
  - https://guides.github.com/features/mastering-markdown/
  - https://www.markdownguide.org

```
# Título 1
## Título 2

- Lista de itens
- Segundo item
```

Ramos - branches

Comandos para trabalhar com ramos (branch)

- **Listando os ramos existentes**

```
git branch
```

- **Criando um novo ramo**

```
git branch [nome do ramo]
```

- **Alterando para outro ramo**

```
git checkout [nome do ramo]
```

- **Criando um novo ramo a partir de um commit específico**

```
git checkout [commit] -b [nome do ramo]
```

- **Mesclando o conteúdo do ramo com o diretório de trabalho atual**

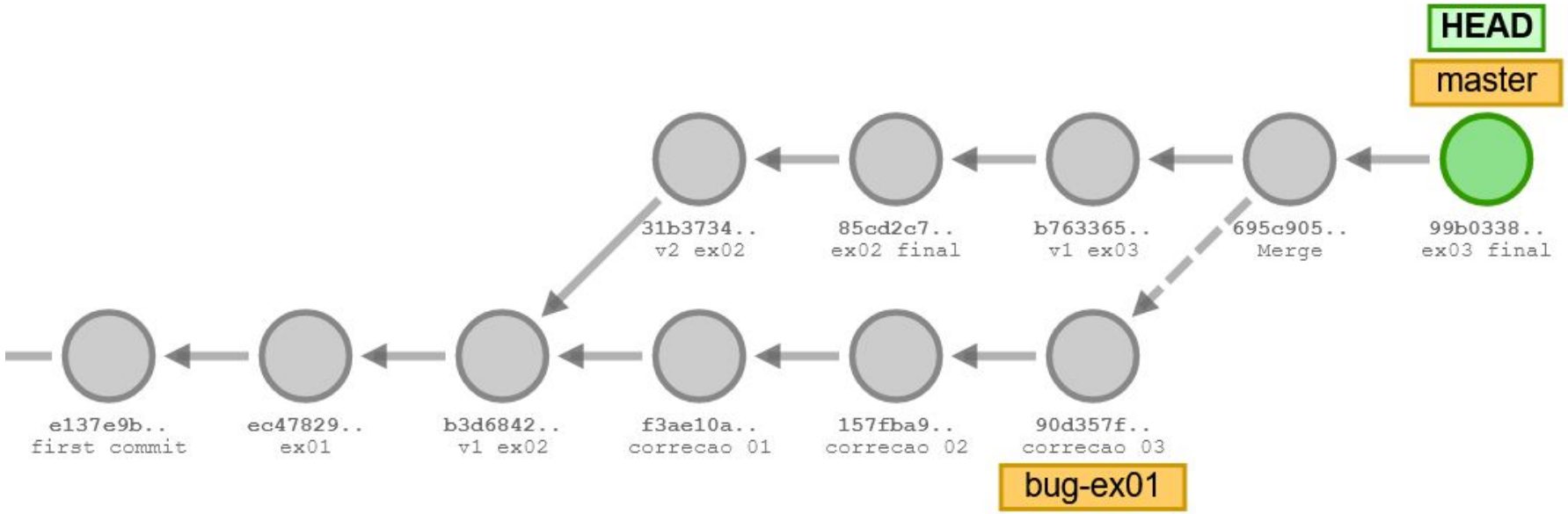
```
git merge [nome do ramo] # ou git rebase [nome do ramo]
```

idp

Práticas com ramos

Iniciando um repositório

- Faça uso do site https://git-school.github.io/visualizing-git/#free e gere a seguinte árvore de commits
  - Use somente os comandos: **commit**, **branch**, **checkout** e **merge**



Repositórios remotos

idp

Trabalhando com repositórios remotos

➔ **Criando repositório local**

```
git init
git add arquivo.md
git commit -m "Iniciando repositorio"
```

➔ **Adicionando repositório remoto com o apelido “origin”**

```
git remote add origin https://site-remoto/repositorio.git
```

➔ **Enviando os commits locais para o repositório remoto**

```
git push -u origin main
```

➔ **Sincronizando repositório local a partir do repositório remoto**

```
# Obtémm os arquivos do repositório remoto, porém não mescla
git fetch origin
# obtémm e mescla qualquer commit de qualquer ramo no repositório remoto
git pull
```

Clonando um repositório remoto

- É possível clonar um repositório remoto em seu computador pessoal

```
git clone https://github.com/lucasrodri/idp_TPAA.git
```

idp

Lista com principais comandos

Lista com principais comandos

- **Marca arquivo** para ir para o próximo commit

```
git add arquivo.md
```

- **Desmarca um arquivo**, mantendo as mudanças no diretório de trabalho (oposto ao git add)

```
git reset HEAD arquivo.md
```

- Mostra o que **foi alterado** no diretório de trabalho e que **ainda não foi marcado**

```
git diff
```

- Mostra o que **foi alterado e marcado**, ou seja, o que vai para o próximo commit

```
git diff --staged
```

Lista com principais comandos

- Visualizando o histórico de commits

```
git log --oneline --graph --decorate --all
```

- Desfazendo alterações de um arquivo (voltar para o último commit)

```
git checkout -- arquivo.md
```

- Restaurando a versão de um arquivo de um commit específico

```
git checkout [commit] -- arquivo.md
```

- Limpa a área de marcação e reescreve toda a árvore do diretório de trabalho a partir do commit especificado **(cuidado!)**

```
git reset --hard [commit]
```

Lista com principais comandos

- **Excluindo um arquivo do diretório de trabalho e do repositório**

```
git rm arquivo.md
git commit -m "Removendo arquivo"
```

- **Excluindo um arquivo do repositório, porém mantendo-o no diretório de trabalho**

```
git rm --cache arquivo.md
git commit -m "Removendo arquivo do índice, porém mantendo-o no
diretório"
```

- **Renomeando arquivo e marcando ele para ir para o próximo commit**

```
git mv nome-antigo nome-novo
git commit -m "Renomeando arquivo"
```

idp

Perguntas?

Referências

- BACKES, A. Ricardo. Algoritmos e estruturas de dados em linguagem C. 1. ed. Rio de Janeiro: LTC, 2023.
- Prof. Emerson Ribeiro de Mello; IF Santa Catarina
- I. C. Corrêa, C. C. Araujo, A. M. Medina; TUTORIAL Git; Universidade Federal de Santa Maria; 2016
- https://git-school.github.io/visualizing-git/#free

INSTITUTO BRASILEIRO DE ENSINO,
DESENVOLVIMENTO E PESQUISA

lucas.costa@idp.edu.br

@lucasrodri

www.linkedin.com/in/lucas-rodri