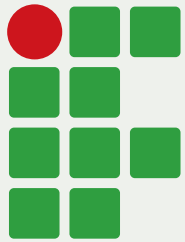
Git – sistema de controle de versão

POO29004 – Engenharia de Telecomunicações

Prof. Emerson Ribeiro de Mello

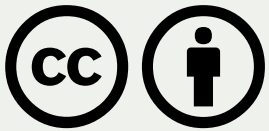
mello@ifsc.edu.br

09 de outubro de 2021



INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Licenciamento



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

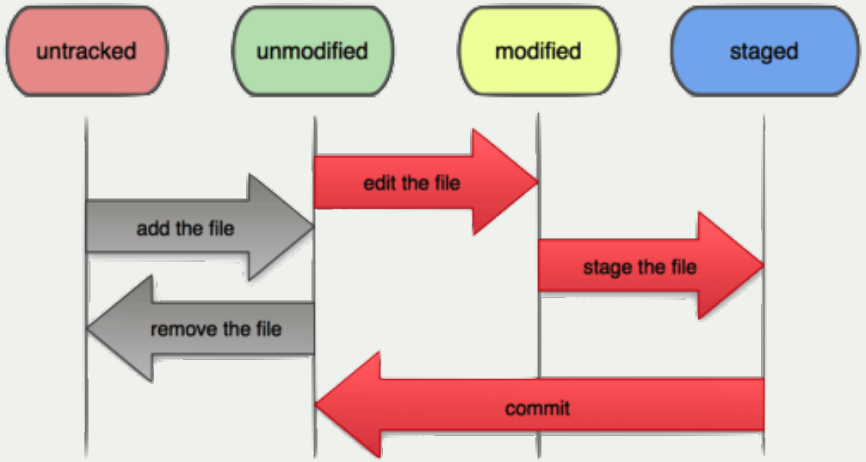
- Ao iniciar um repositório git, será criado um subdiretório `.git` e este armazenará os dados e metadados de todas as alterações feitas no repositório

```
1 mkdir projeto
2
3 cd projeto
4
5 git init
```

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

```
1 echo "Olá mundo" >> arquivo.txt
2
3 git add arquivo.txt
4
5 git commit -m "Adicionando arquivo.txt"
```

Ciclo de vida de arquivos em um repositório git




```
1  # arquivo novo fica como untracked
2  echo "olá" >> novo.txt
3  # arquivo é marcado (staged) para ir para o próximo commit
4  git add novo.txt
5  # mudanças persistidas
6  git commit -m "adicionando arquivo"
7  # arquivo sai do staged para modified
8  echo "mundo" >> novo.txt
9  # marcando (staged) todos os arquivos modificados
10 git add .
11 # mudanças persistidas
12 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

```
1 git branch
```

- Criando um novo ramo

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

- Alterando para outro ramo

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

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

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

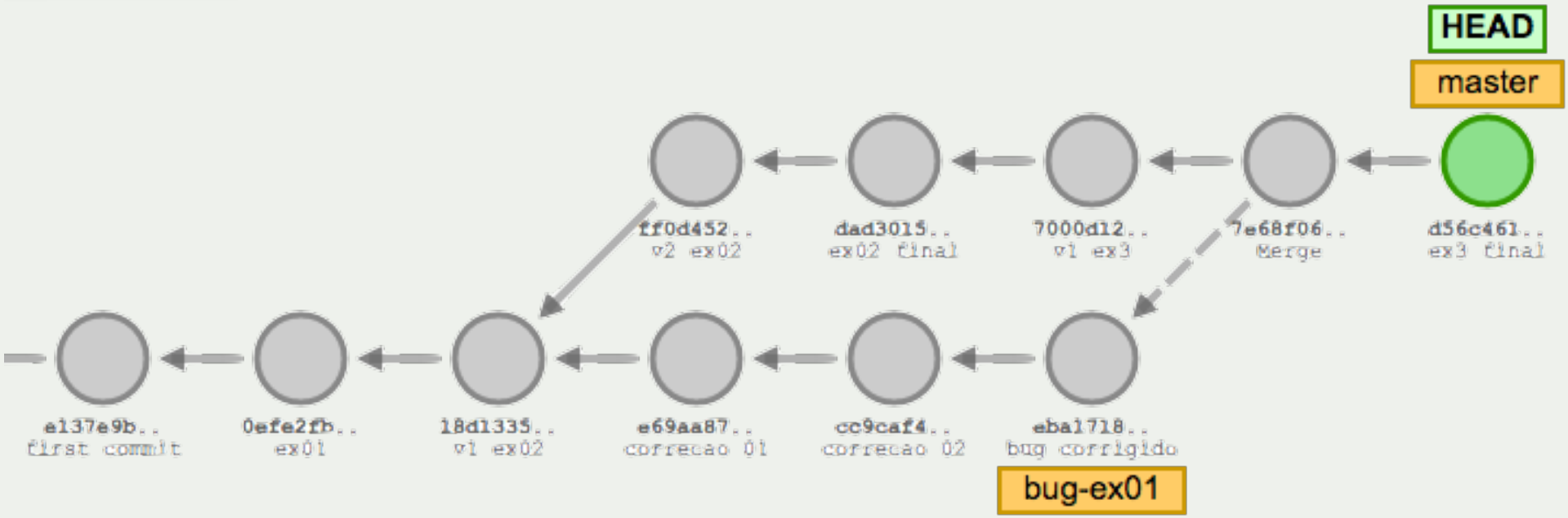
- Mesclando o conteúdo do ramo com o diretório de trabalho atual

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


Prática com ramos

Prática com ramos

- 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

Trabalhando com repositórios remotos

- Criando repositório local

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

- 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

```
# Obtém os arquivos do repositório remoto, porém não mescla
git fetch origin
# obtém 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

```
1 git clone https://github.com/emersonmello/modelos-latex
```


Lista com principais comandos

Lista com principais comandos

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

```
1 git add arquivo.md
```

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

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

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

```
1 git diff
```

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

```
1 git diff --staged
```

Lista com principais comandos

- Visualizando o histórico de commits

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

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

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

- Restaurando a versão de um arquivo de um *commit* específico

```
1 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!)

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


Lista com principais comandos

- Excluindo um arquivo do diretório de trabalho e do repositório

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

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

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

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

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


Referências

- https://training.github.com/downloads/pt_BR/github-git-cheat-sheet
- https://docs.github.com/pt/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax