2021/05/03 - AULA 07.1

Laboratório de Sistemas Computacionais Complexos

https://uclab.xyz/sistemas-complexos-2021-aula07-1

Renato Cordeiro Ferreira
renatocf@ime.usp.br

João Francisco Daniel
joaofran@ime.usp.br

Alfredo Goldman
gold@ime.usp.br

Thatiane de Oliveira Rosa
thatiane@ime.usp.br

Em caso de dúvidas

Acessem slido.com com #complexos

ou

Agenda

Tema da aula:
Git Avançado

1. Git Básico
2. Branching
3. Merge vs. Rebase
4. Git Workflows
5. Trunk-Based Development

Git Básico

git add

git commit

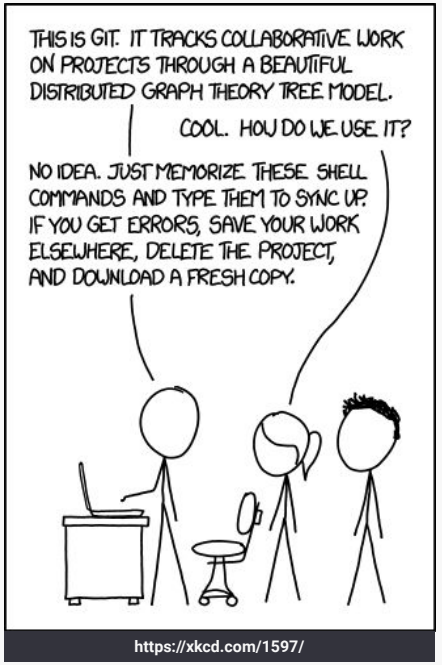
git push

git clone

git pull

"The Mindless three"

Começando os trabalhos



Branching

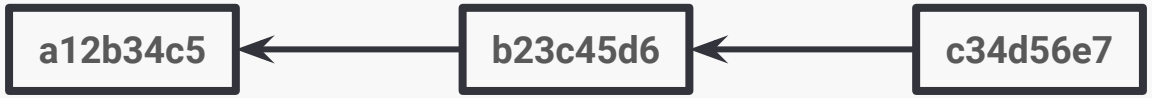
Commits são "caixinhas" que guardam metadados sobre as modificações do código

a12b34c5

Cada commit é identificado por checksum SHA-1
que representa um cabeçalho + as mudanças do commit

Branching

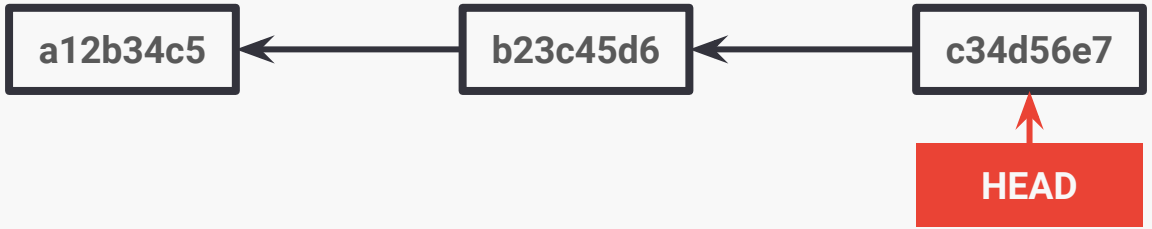
Commits se ligam como numa lista, apontando para o commit do qual se originaram



Novos commits são criados com a combinação
de comandos git add + git commit

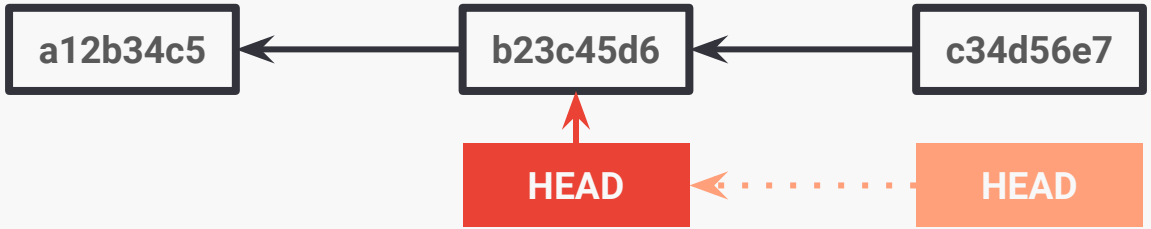
Branching

git checkout permite ver versões antigas do código ao mover a HEAD para outros commits



Branching

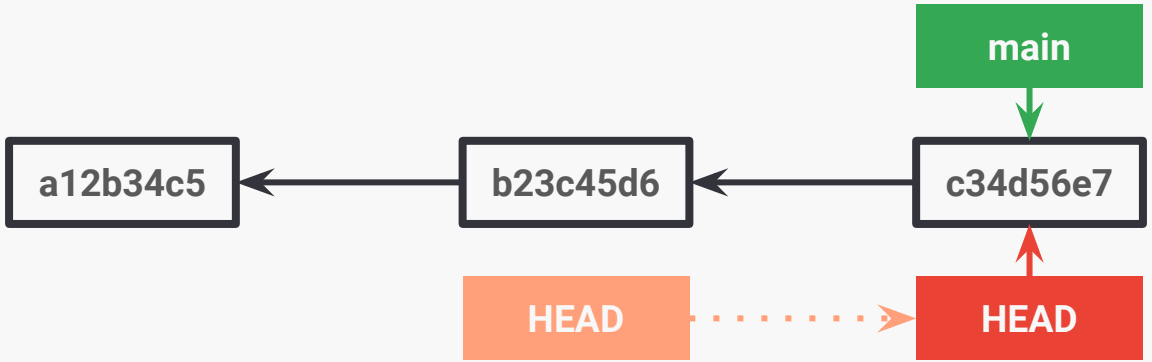
A cabeça dessa lista (HEAD) representa o que você visualiza no repositório no momento




```
git checkout b23c45d6
git checkout HEAD^1
```

Branching

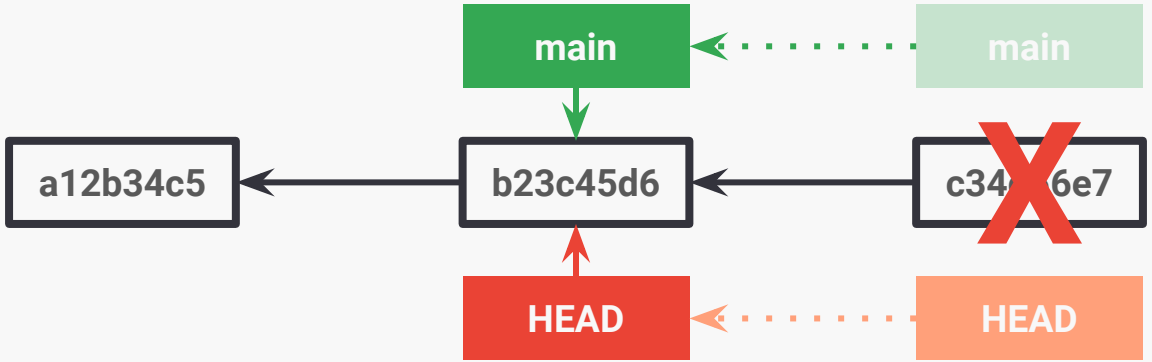
`git checkout` pode voltar para a ponta da lista pois ela é identificada por padrão como `main`



`git checkout master`

Branching

git reset modifica a main e a HEAD juntas, tornando um commit inacessível

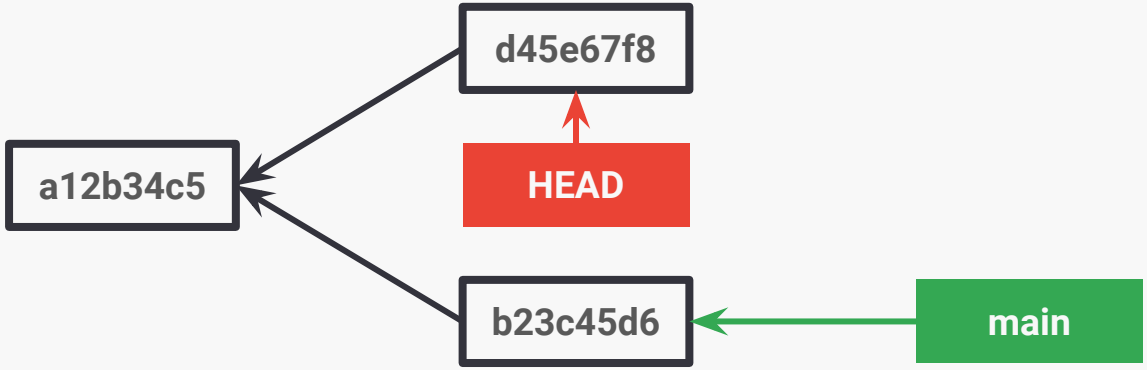


git reset b23c45d6

git reset HEAD^1

Branching

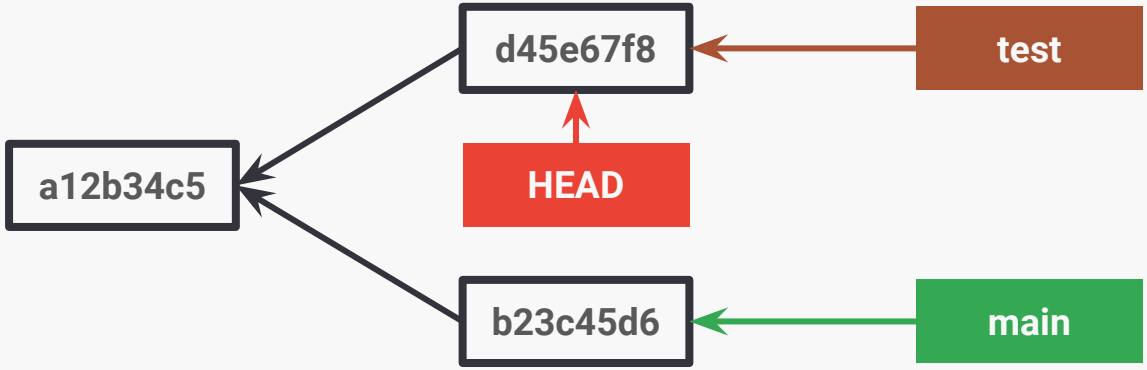
git commit com a HEAD desconectada da main gera uma árvore de commits



git commit

Branching

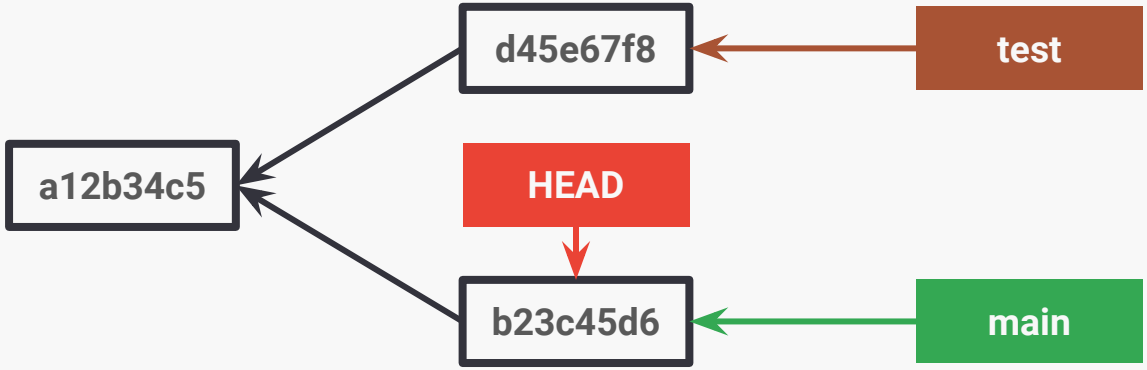
git checkout -b permite criar um novo nome para referenciar esse novo galho (branch)



git commit

Branching

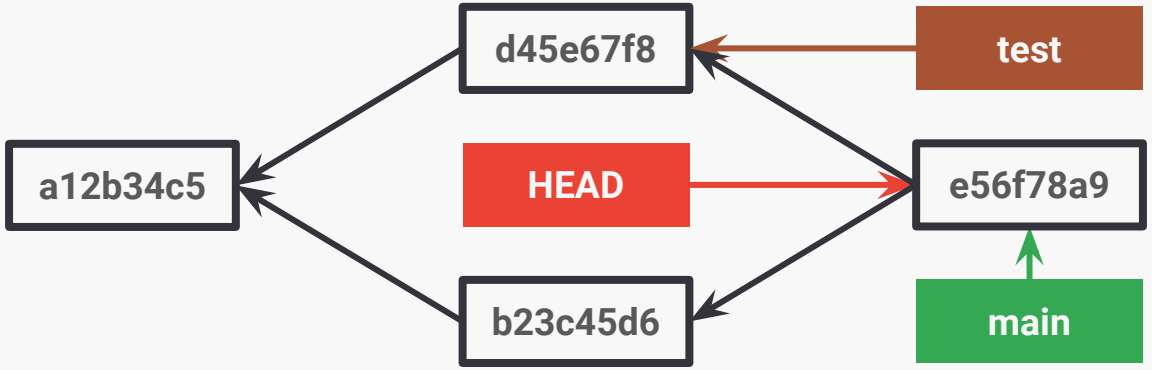
`git checkout` ainda permite navegar entre esses commits como desejado



`git checkout master`

Merge vs. Rebase

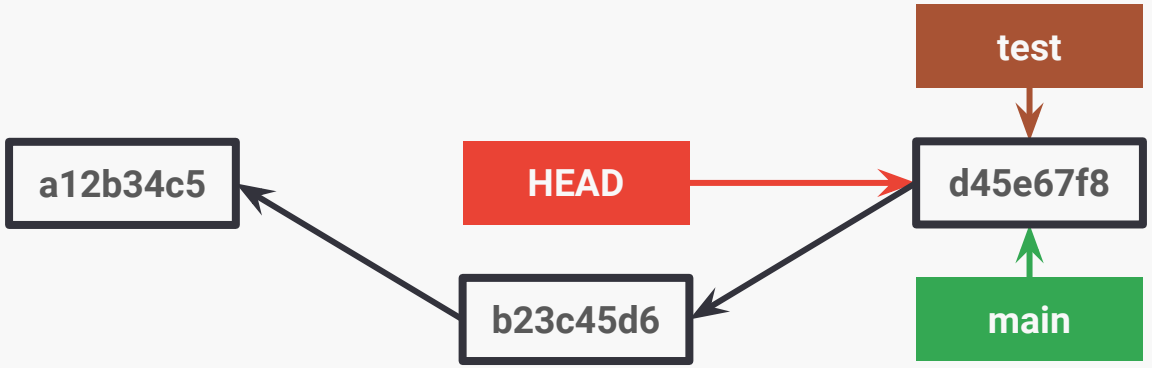
`git merge` permite juntar duas branches gerando um commit que registra a junção



`git merge test`

Merge vs. Rebase

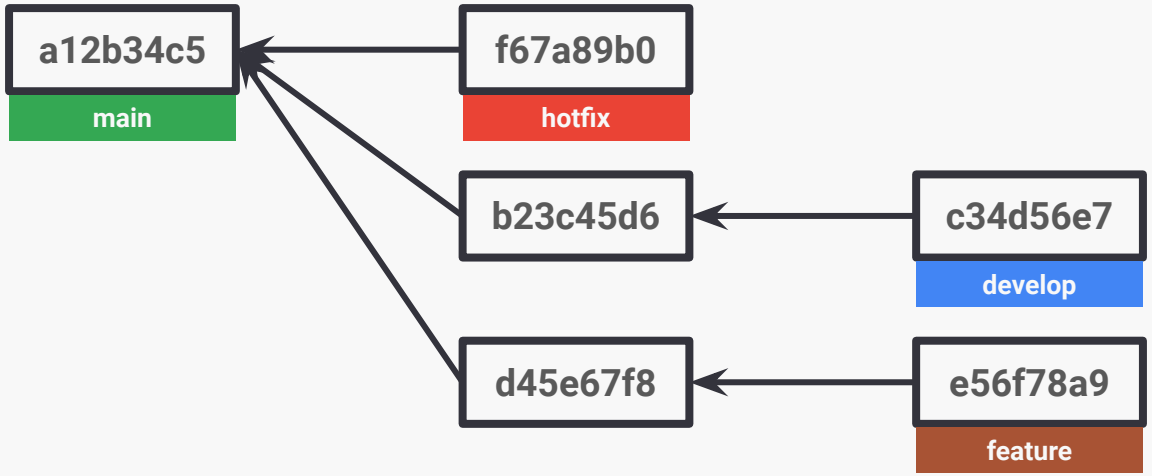
`git rebase` permite reaplicar uma branch sob outra gerando um histórico linear



`git rebase test`

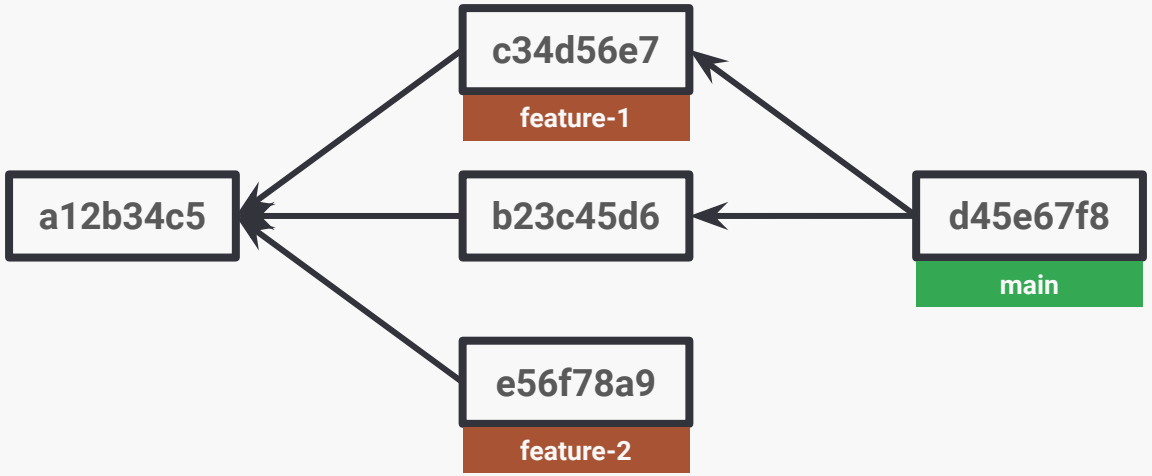
Git Flow

Git Flow propõe uma branch estável, uma de desenvolvimento, várias para funcionalidades e algumas de urgência



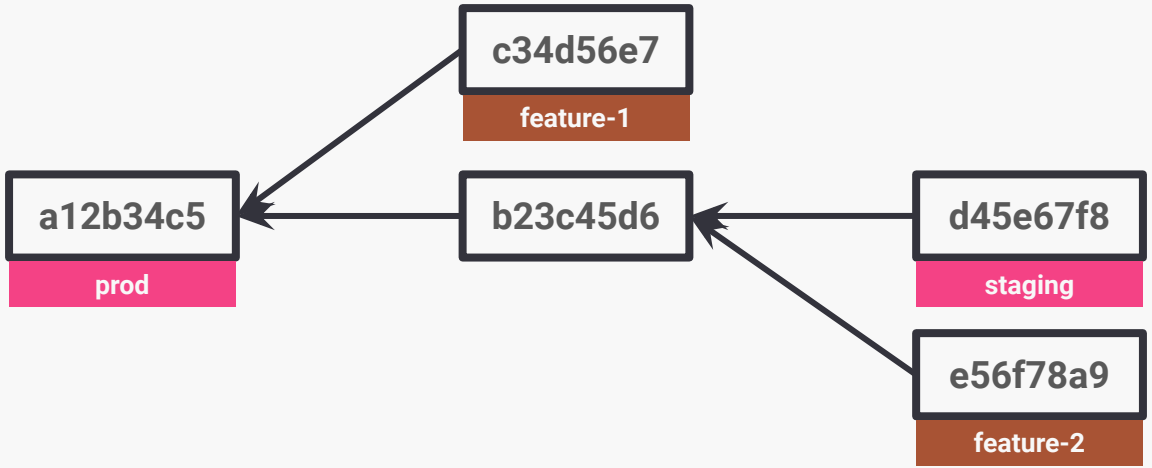
GitHub Flow

GitHub Flow propõe uma branch estável
e várias para funcionalidades que são integradas via pull requests



GitLab Flow

GitHub Flow propõe uma branch para cada ambiente de deploy
e para funcionalidades que são disponibilizadas via merge requests



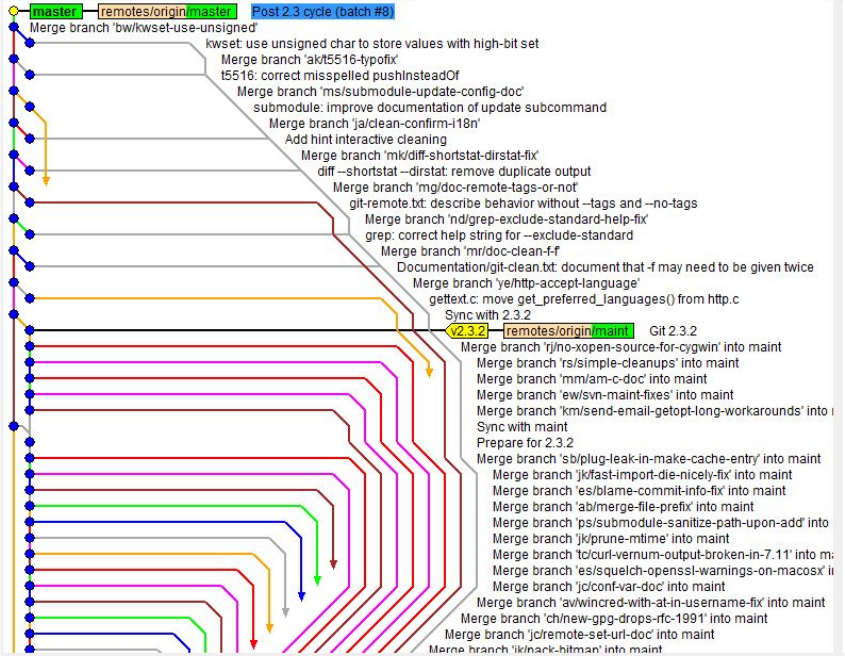
Qual estratégia utilizar?

Resposta curta:

NENHUMA

Resposta longa:

Depende, mas no geral, nenhuma...



Trunk-Based Development

Contra o branching hell, aplique trunk-based development

git add

git commit

git push

"The Mindless three"

Se você só tem a branch main e faz commits com frequência,
você dilui merges necessários a cada commit

Trunk-Based Development

Single Source of Truth

+

Continuous Integration

+

Continuous Code Review

+

Feature Flags

=

Trunk-Based Development

Mas... Mas... Mas...

Não pode ser tão simples assim!

Trunk-Based Development

Single Source of Truth

+

Continuous Integration

+

Continuous Code Review

+

Feature Flags

=

Trunk-Based Development

Resposta curta:

Realmente não é...

Resposta longa:

Se o seu projeto é de software livre, ou seu time é muito grande, ou sua revisão de código é longa, ou seu código integra vários repositórios...

Licença

Créditos

Imagens usadas nesta apresentação são provenientes de: freepik.com

Frequência

Senha do Estudante: **8dag6w**

2021/05/03 - AULA 07.1

Laboratório de Sistemas Computacionais Complexos

https://uclab.xyz/sistemas-complexos-2021-aula07-1

Renato Cordeiro Ferreira
renatocf@ime.usp.br

João Francisco Daniel
joaofran@ime.usp.br

Alfredo Goldman
gold@ime.usp.br

Thatiane de Oliveira Rosa
thatiane@ime.usp.br