Git 101 - MAC0110

Matheus Tavares

matheus.bernardino@usp.br

Pedro Teos

pedro.teotonio.sousa@usp.br

Agenda

1. Controle de versões
2. O que é Git? / Por que Git?
3. Conceitos básicos
    - Commits e HEAD
4. Exemplo prático com comandos fundamentais
    - Setup
    - Salvando modificações
    - Visualizando e recuperando versões
5. Boas práticas de commits
6. Remotes e GitLab

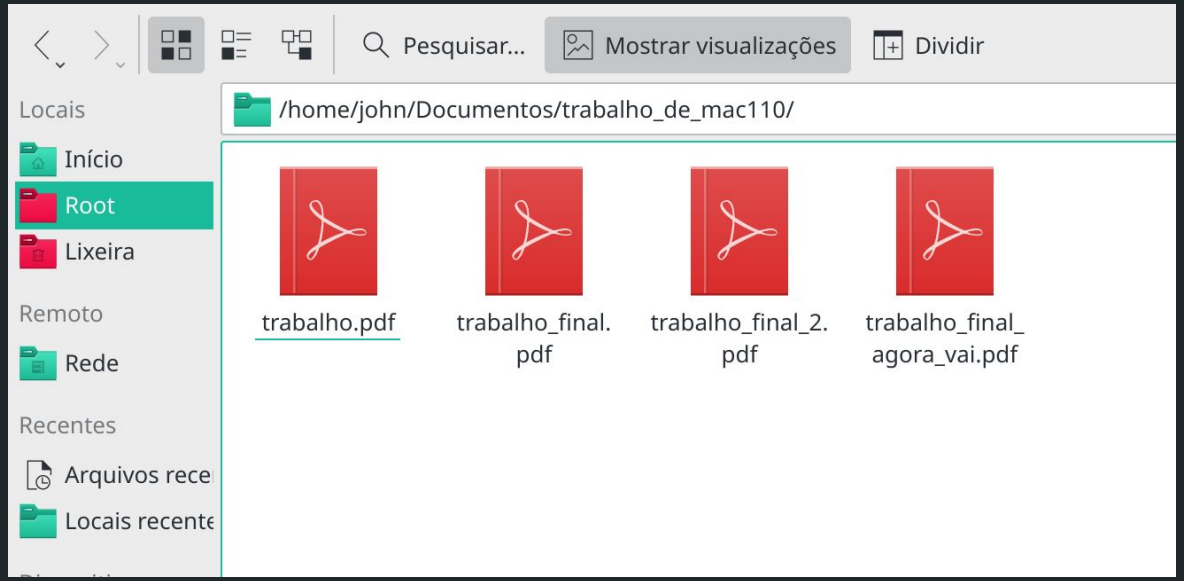
O que é controle de versões?

Controle de Versões

“Controle de versão é um sistema que **registra alterações** em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa **lembrar versões específicas** mais tarde.”

- Git Pro Book

Controle de Versões



O que é *Git*?
E como ele funciona?

Git, GitHub, GitLab, ...

≠

Git, GitHub, GitLab, ...

≠

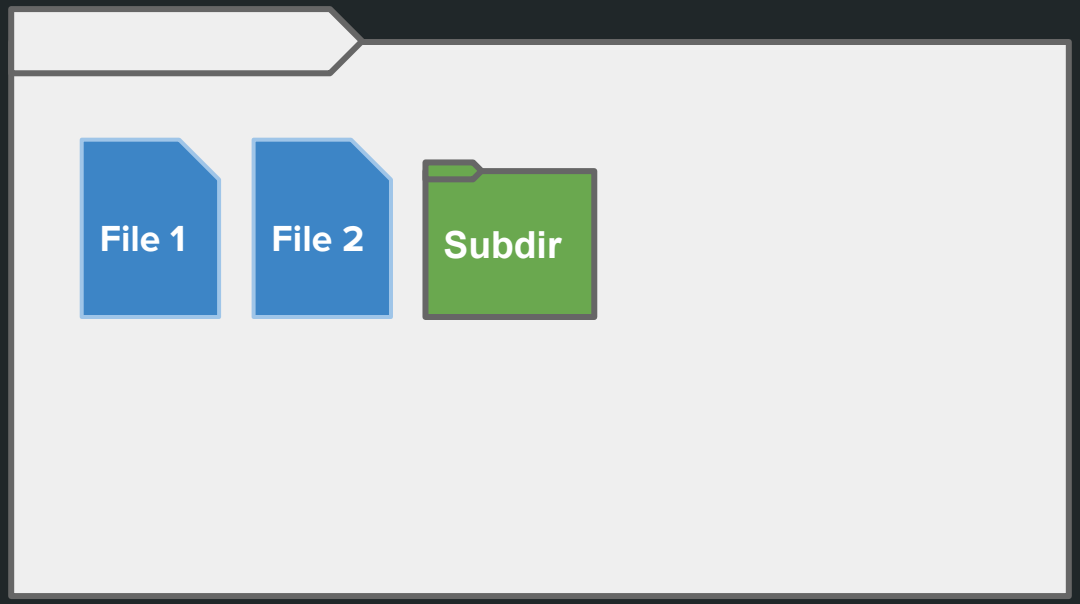
Sistema de controle de versões
(o software que você roda na sua máquina)

Serviços para hospedagem de repositórios online

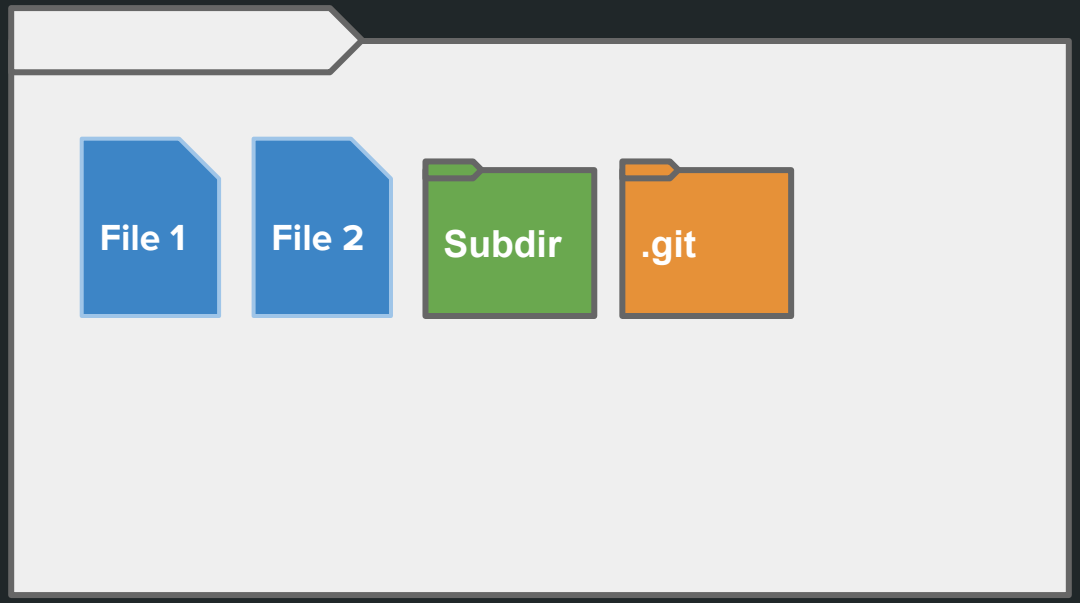
O que é Git?

- Criado em 2005, por Linus Torvalds
  - Revogação da licença grátis do BitKeeper, antigo VCS usado no kernel Linux.
  - Linus cria o Git como uma alternativa aberta.
- Quase **90% dos desenvolvedores** usam Git para versionamento, hoje. [3]
- Gerencia desde projetos de poucos MBs até projetos como o Windows, com aproximadamente 300 GBs.

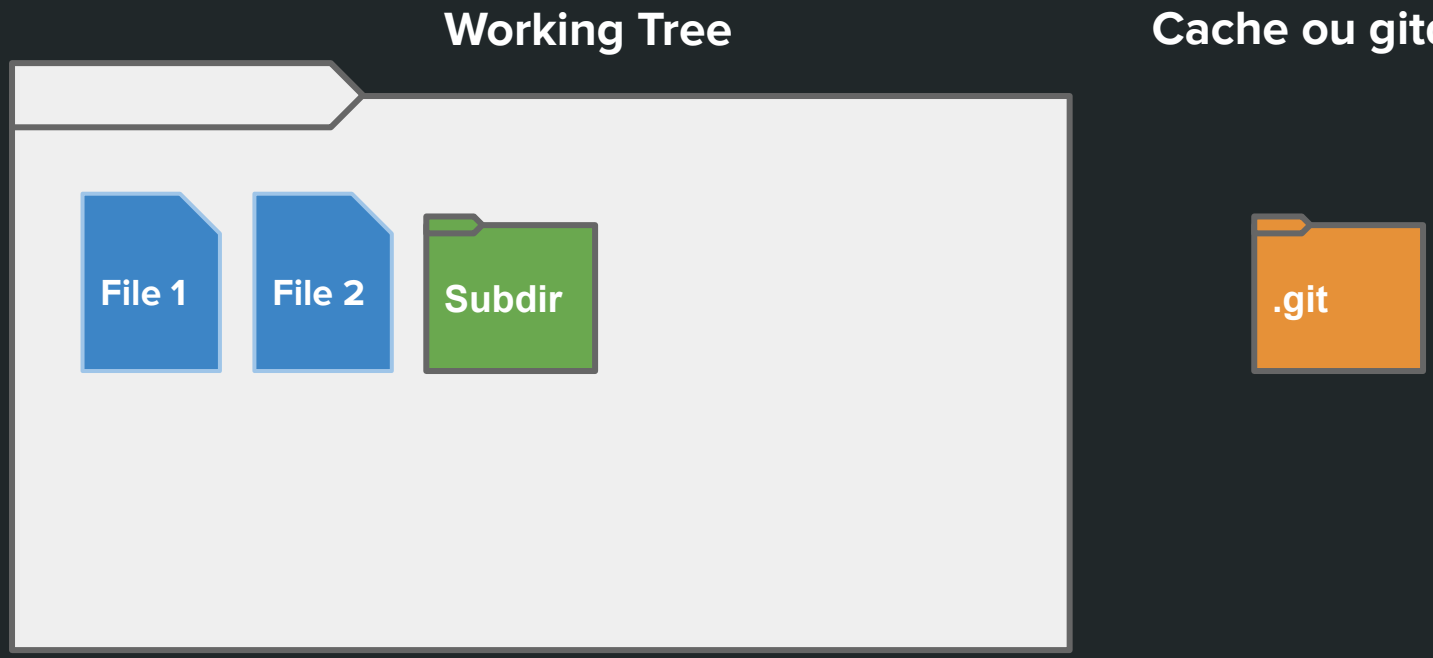
Repositório (ou repo, pros íntimos)



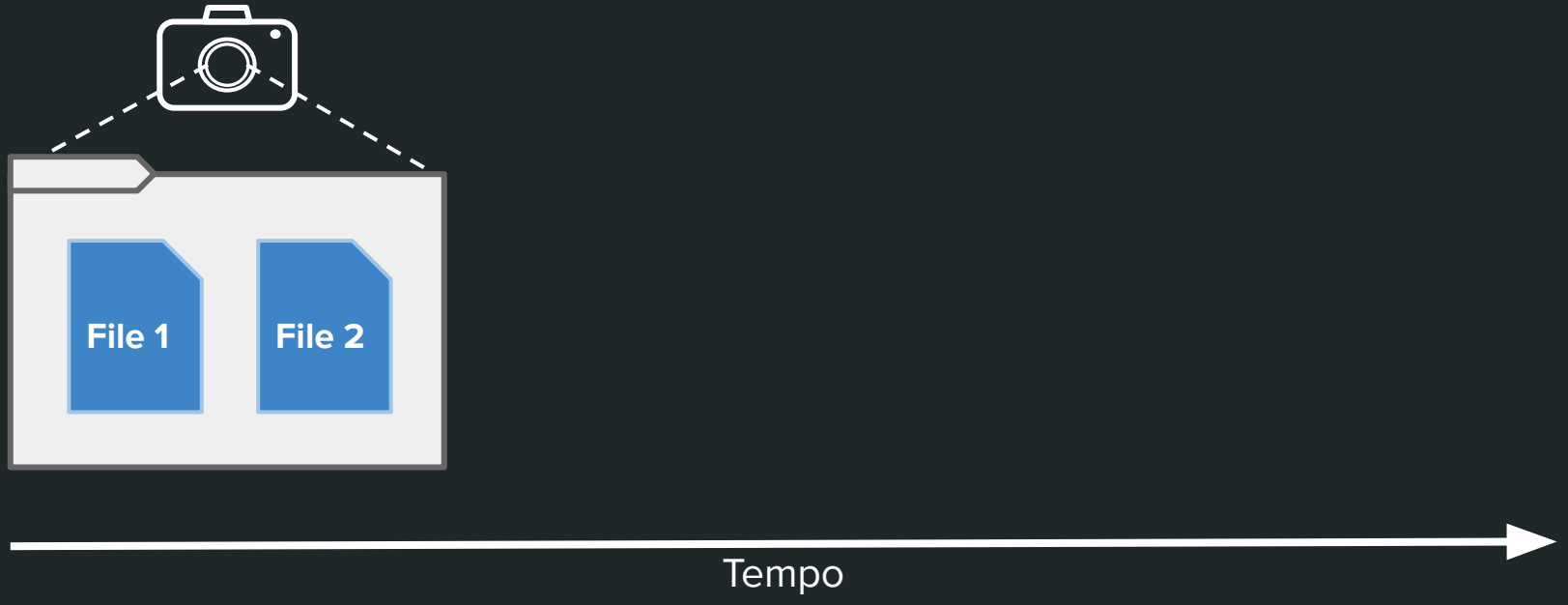
Repositório (ou repo, pros íntimos)



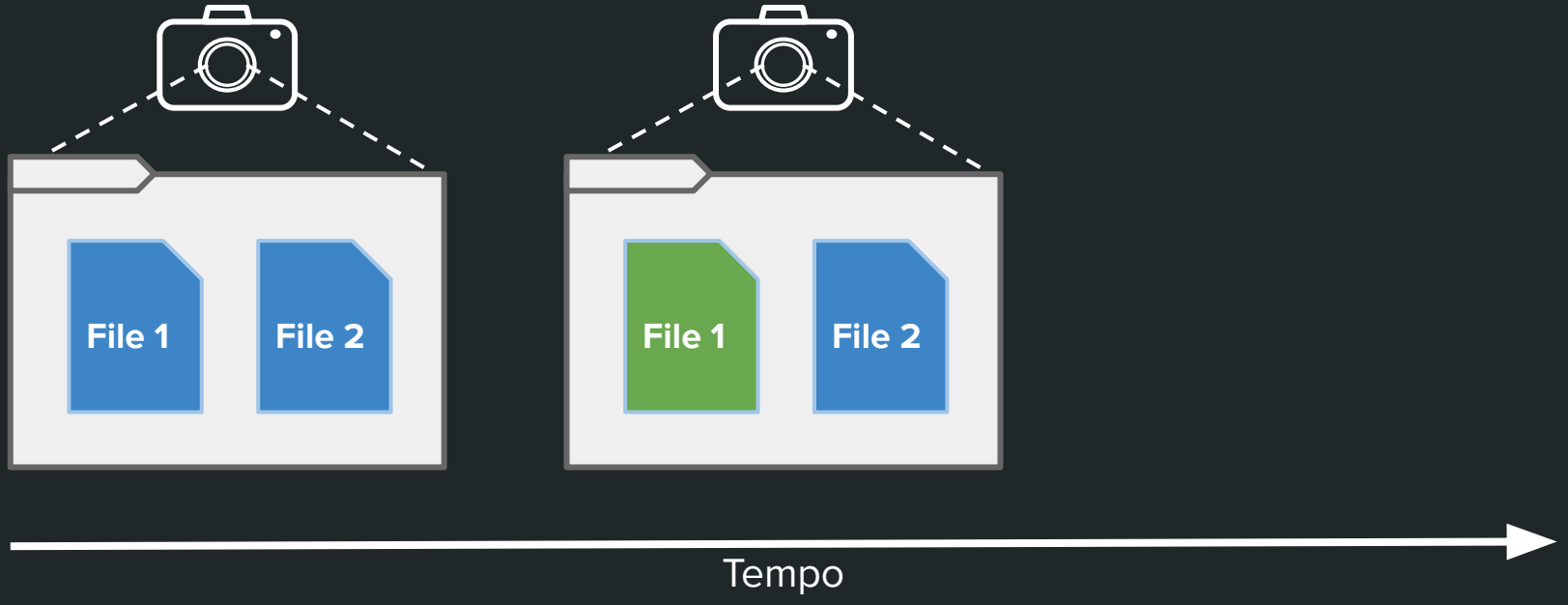
Repositório (ou repo, pros íntimos)



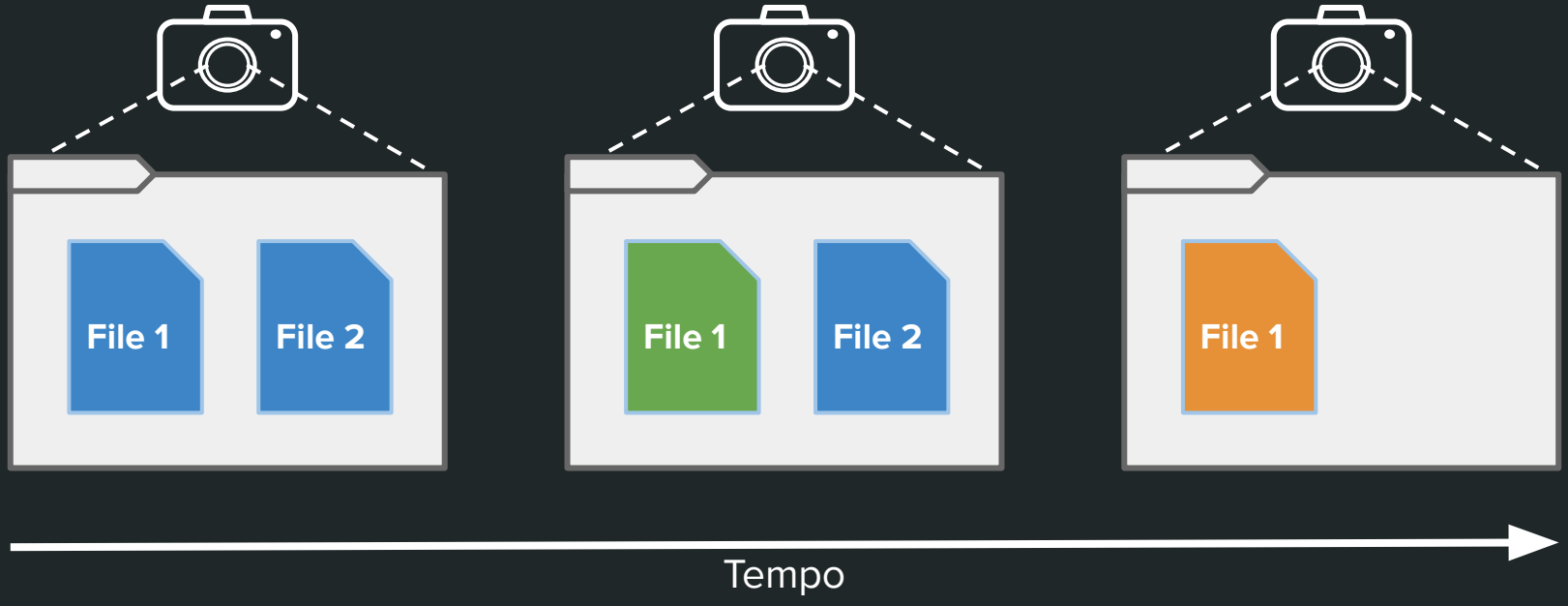
“Fotos” de um diretório



“Fotos” de um diretório



“Fotos” de um diretório

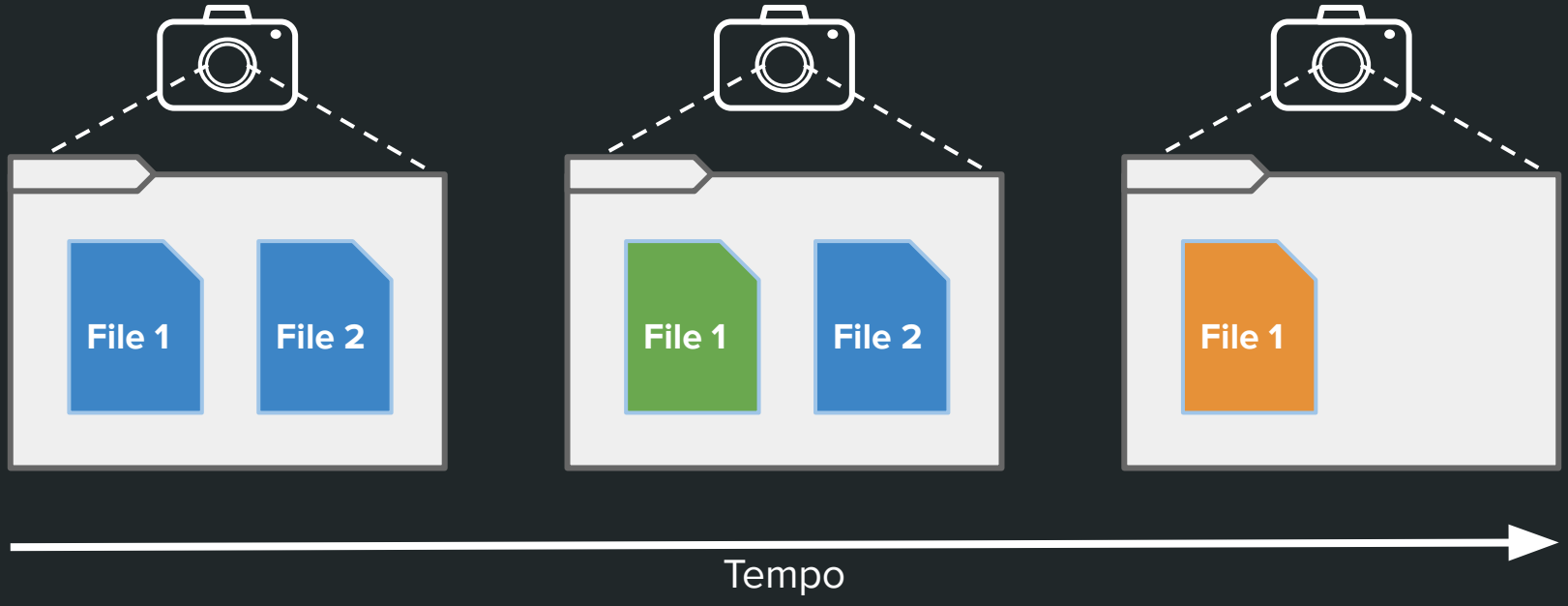


“Fotos” de um diretório

“Adicione notas da aula 1 de MAC0110 e foto da lousa”

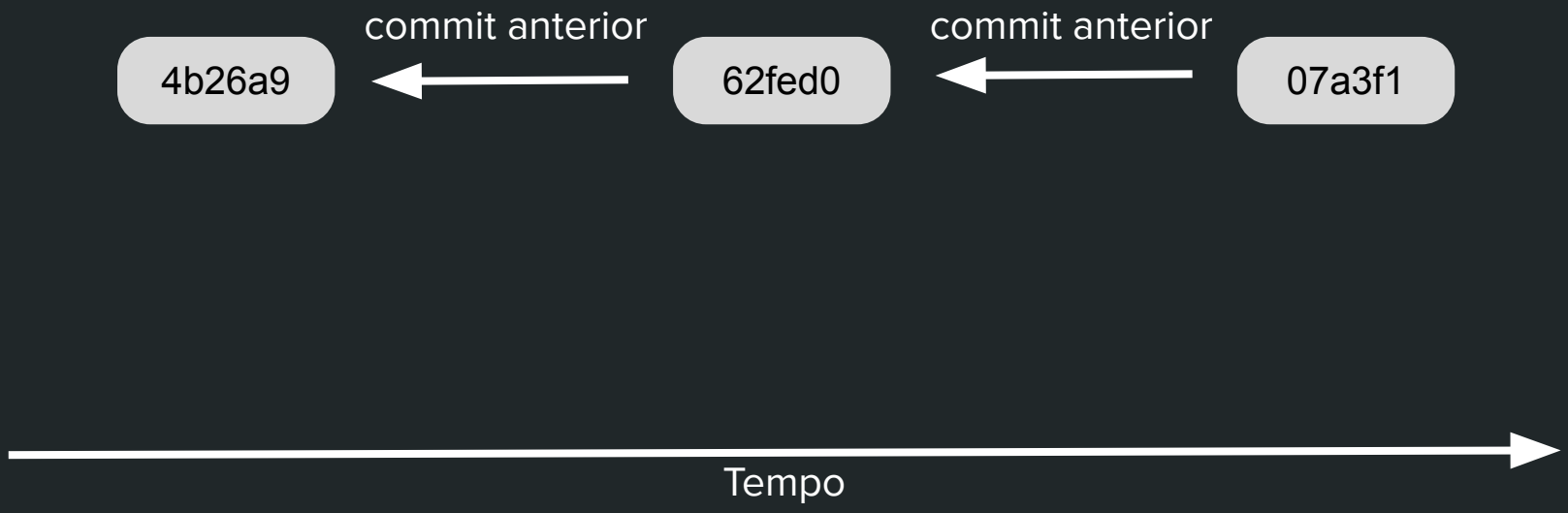
“Coloque as palavras inglesas do arquivo 1 em *itálico*”

“Transcreva os conteúdos da foto da lousa e remova-a”



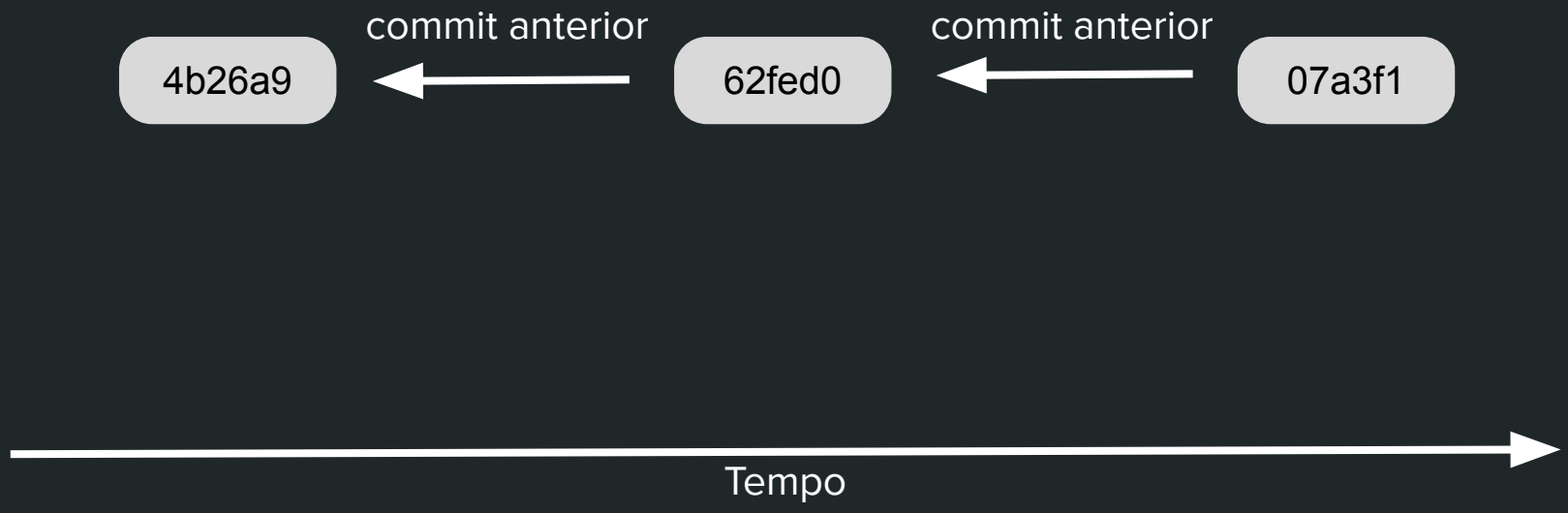
Registro de *“fotos com informações”*

Registro de ***commits***



Registro de ***commits***

- Também apelidados de *versions* ou *revisions*

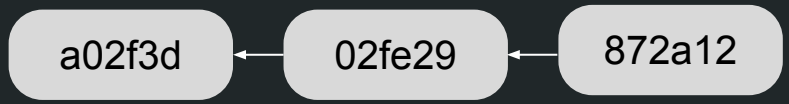


O que posso fazer com isso?

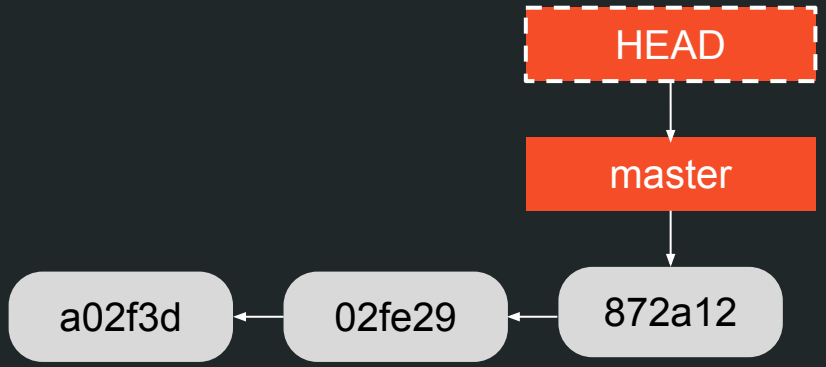
- O que mudou entre uma versão e outra?
- Como eram os arquivos X e Y a um mês atrás?
- Quem alterou por último a linha L? E porquê?
- Qual mudança introduziu o bug #21? Vamos reverter essa mudança.
- etc.

Referências

Histórico de commits

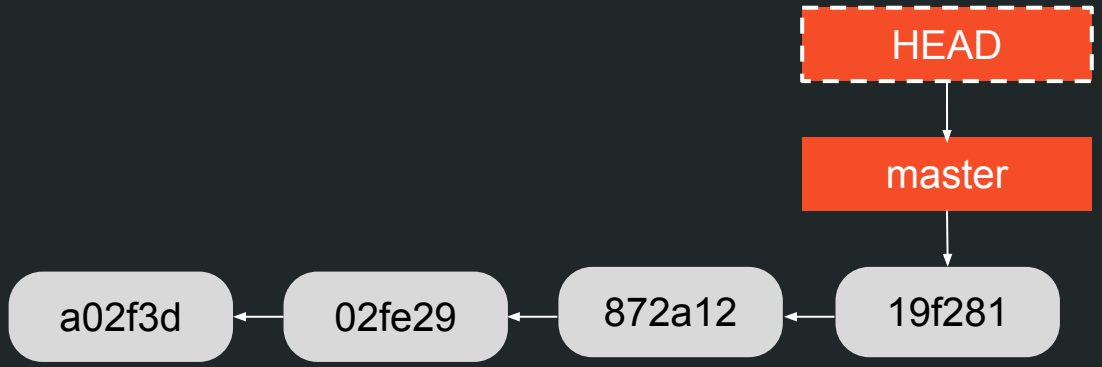


Referências



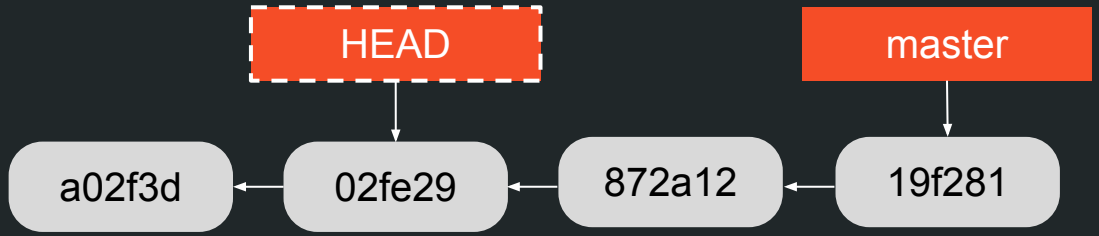
Referências

$ git commit



Referências

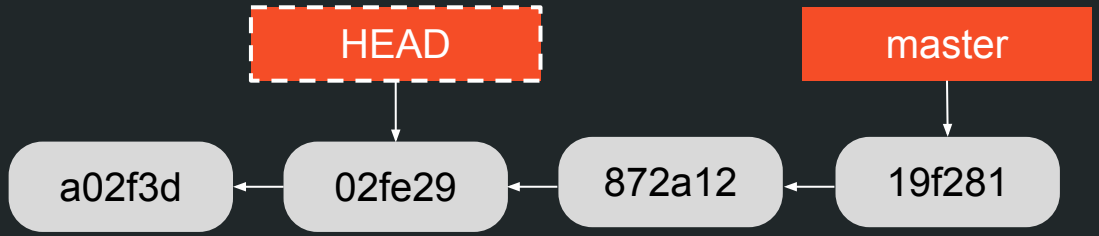
$ git checkout 02fe29



Referências

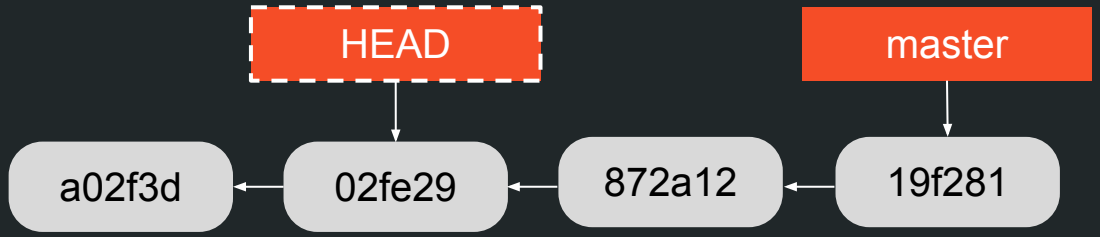
$ git checkout 02fe29

You are in “detached HEAD” state.



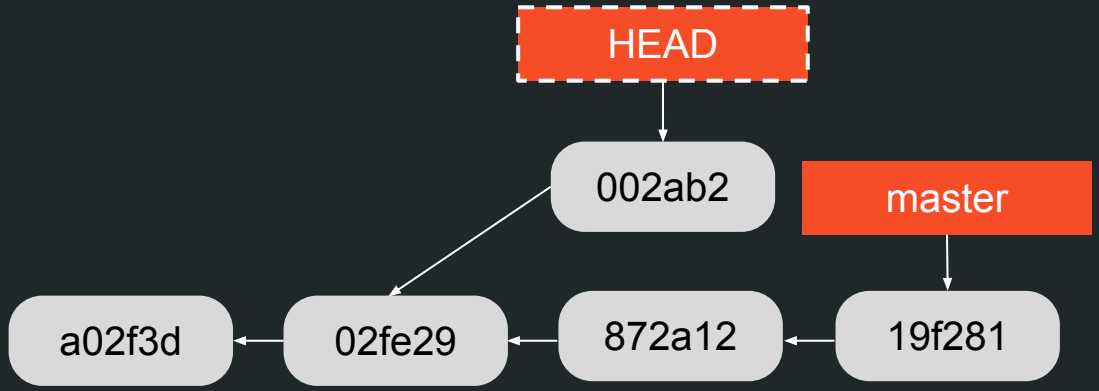
Referências

$ git commit



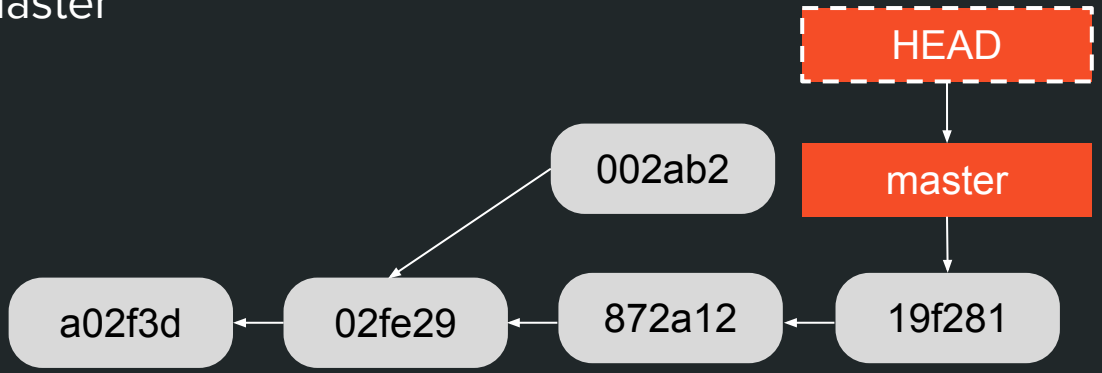
Referências

$ git commit



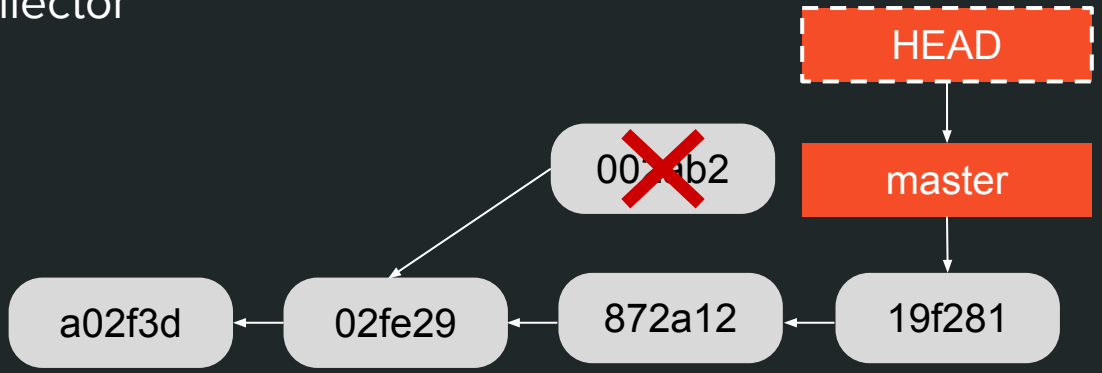
Referências

$ git checkout master



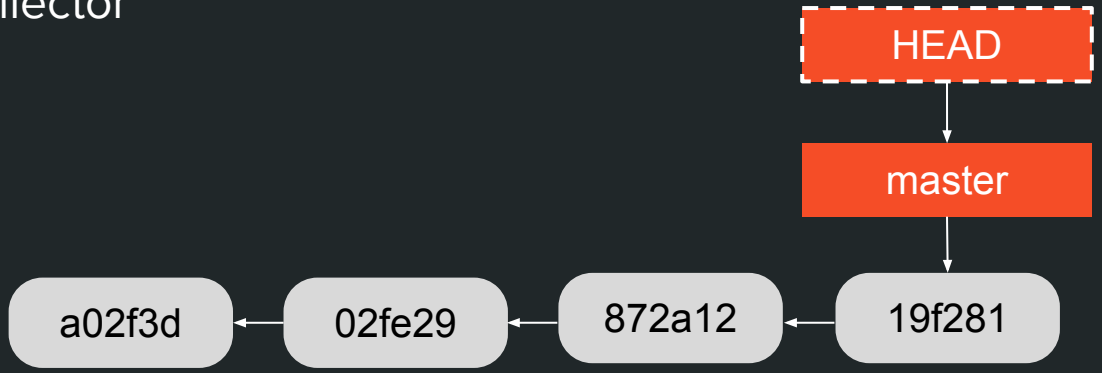
Referências

Git’s garbage collector



Referências

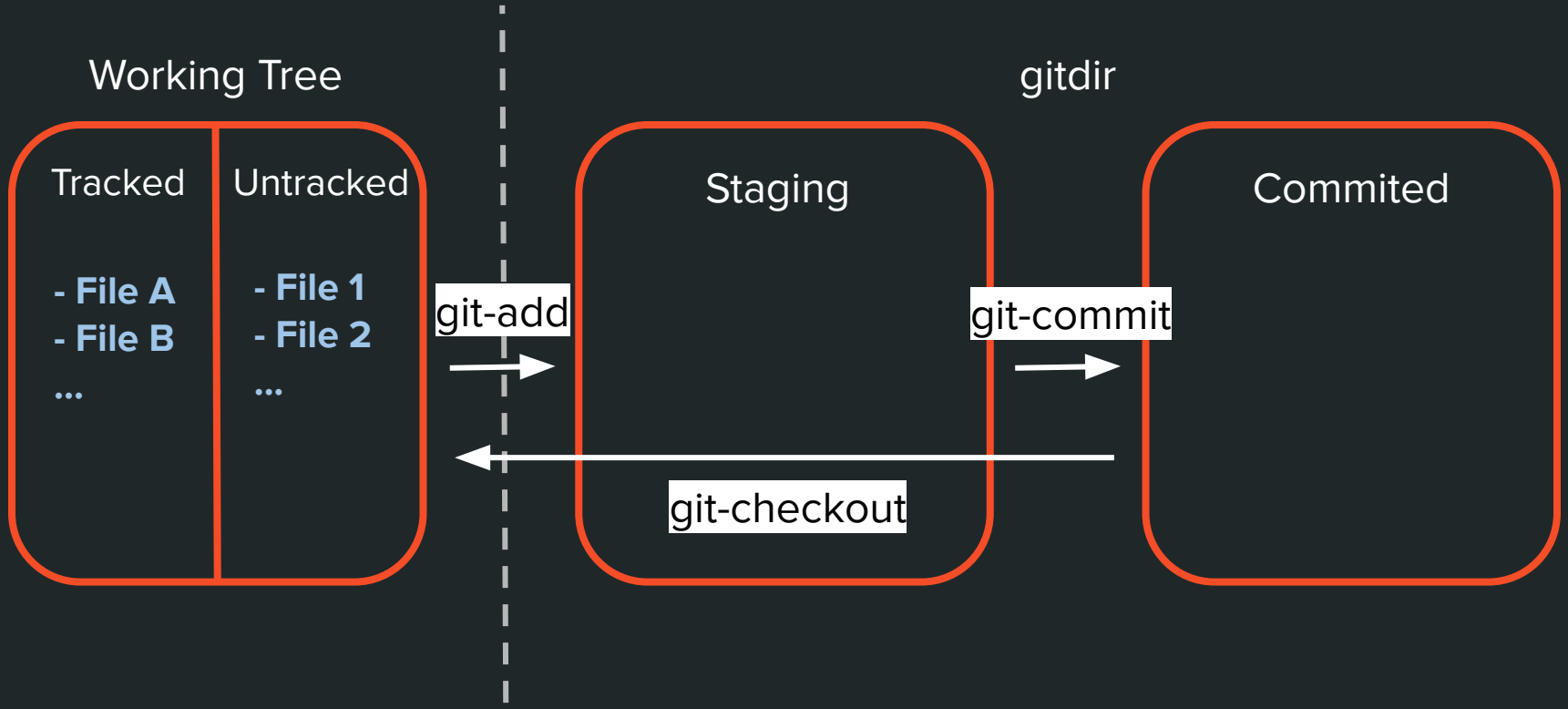
Git’s garbage collector



Comandos básicos

Hora de um exemplo prático!

Add e Commit: mais detalhes



Dica

git help glossary: lista de nomenclaturas e conceitos usados no Git, com definições.

Boas práticas de *Commits*

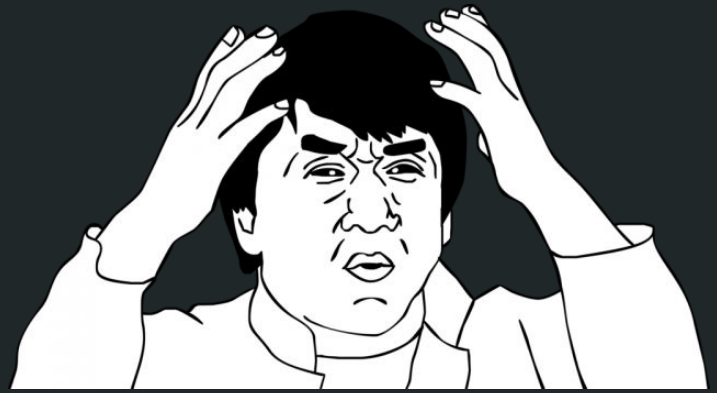
Parece uma boa ideia...

$ vim file.c

$ git add .

$ git commit -m “Blá”

... até que você precise relembrar o que fez.



https://blog.tiagopariz.com/wp-content/uploads/2018/10/jackie-chan-meme.png

```
commit 5e2faaea31dceaad23a3331cd9ad20afad6c719b
Author: joh123 <>

    Minor changes

commit ae95825114ccac3a996251bd1de4da5e83c95172
Author: joh123 <>

    Fixed some bugs

commit 6e639b014f82521138025091fac660cd2474b7a0
Author: joh123 <>

    Now it compiles!

commit f717b52ddd1e95dbbdd5a1476051b842468cb195
Author: jonh123 <>

    Actually it didn't. NOW IT DOES

commit 90384aec06d44a5d40d9c299f082334503943b32
Author: john123 <>

    Update README
```

Dissecando um bom commit

1. Mudanças **não correlacionadas** pertencem a **commits separados**.
2. **Não *commitar*** blocos de trabalho **incompletos**.
3. Invista na escrita de mensagens de commit **informativas**.

1. Mudanças **não correlacionadas** pertencem a **commits separados**.

- Mais fácil de **revisar** (➜ melhores revisões ➜ melhor código)
- Mais fácil de **reverter**
- Mais fácil de **integrar** com outros commits e branches (menos conflitos)

1. Mudanças **não correlacionadas** pertencem a **commits separados**.

2. **Não *commitar*** blocos de trabalho **incompletos**.

- Commits devem ser justificáveis por si só. (Embora um commit possa depender de outro)
- É legal garantir que **cada commit** seja compilável e passe os testes. (➜ mais fácil encontrar bugs / mudanças de comportamento)

2. Não *commitar* blocos de trabalho incompletos.

```
commit c6444bb0d21d625311d8f06935acd6ea2cf3a8bd
Author: A U Thor <author@example.com>

    Iniciando implementação do método de euler

diff --git a/a b/a
new file mode 100644
index 0000000..d8ab891
--- /dev/null
+++ b/a
@@ -0,0 +1,6 @@
+
+def euler(x0, v0, a, dt, N):
+       x, v = x0, v0
+       for i in range(N):
+               x = x + v*dt
+               v =
```

```
commit 1a3f223c6341684ee78d04760a9188563397b028
Author: A U Thor <author@example.com>

    Finalizando implementação do método de euler

diff --git a/a b/a
index d8ab891..96be0ca 100644
--- a/a
+++ b/a
@@ -3,4 +3,5 @@ def euler(x0, v0, a, dt, N):
        x, v = x0, v0
        for i in range(N):
                x = x + v*dt
-               v =
+               v = v + a*dt
+       return x
```

Esses commits podem ser unidos em um.

3. Invista na escrita de mensagens de commit **informativas**.

> “a well-crafted Git commit message is the best way to communicate context about a change to fellow developers (and indeed to [your future self]). A diff will tell you **what changed**, but only the commit message can **properly tell you why**.”
>
> - Chris Beams (https://chris.beams.io/posts/git-commit/)

3. Invista na escrita de mensagens de commit **informativas**.

- **Descreva o problema**: o que não está legal no código atual?
- **Justifique como as mudanças resolvem o problema**: porque o estado do projeto após este commit é melhor do que o atual?
- **Alternativas descartadas [opcional]**: existem outros modos de implementar a mudança? Se sim, porque este foi escolhido?

Dica:
git commit -v ou
git config --global commit.verbose true

O formato da mensagem também é importante

```
Título resumindo as mudanças em até 50 chars

Corpo, separado do título por uma linha em branco, e
justificado em 72 colunas. Explica a mudança em mais
detalhes, como descrito no slide anterior. (Para mudanças
triviais, pode ser omitido.)

O corpo pode ter múltiplos parágrafos e também:

  - Bullet points
  - Tabelas ou outros
```

Normalmente:

- No imperativo
- Não pontuado

Dissecando um bom commit

Exemplo

```
commit 7655b4119d49844e6ebc62da571e5f18528dbde8
Author: René Scharfe <l.s.r@web.de>
Date:   Tue Mar 3 21:55:34 2020 +0100

remote-curl: show progress for fetches over dumb HTTP

Fetching over dumb HTTP transport doesn't show any progress, even with
the option --progress.  If the connection is slow or there is a lot of
data to get then this can take a long time while the user is left to
wonder if git got stuck.

We don't know the number of objects to fetch at the outset, but we can
count the ones we got.  Show an open-ended progress indicator based on
that number if the user asked for it.
```

Trabalhando com *Remotes*

Remote

- Um repositório *remoto* relativo ao mesmo projeto.
  - Fazer backup na nuvem
  - Compartilhar código com outros (ou fazer download de código de outros)
  - Desenvolvimento colaborativo
  - etc.

Remote

- Você pode ter **vários** remotes (e.g. seu backup, o de um colega, ...)
  - **$ git remote add** john https://github.com/john/repo.git
- Atualizar branches do repo remoto:
  - **$ git push <remote> <branch>**
- Atualizar sua branch local:
  - **$ git pull <remote> <branch>**

Remote

- E baixamos um repositório remoto com:
  - **$ git clone <url>**

Remote

Outro exemplo prático? :)

Referências

1. **Pro Git**, Scott Chacon and Ben Straub: https://git-scm.com/book/en/v2
2. **Git Docs:** https://git-scm.com/docs/
3. **Stack Overflow Developer Survey Results from 2018:** https://insights.stackoverflow.com/survey/2018#work-_-version-control
4. **How to Write a Git Commit Message**, Criss Beans: https://chris.beams.io/posts/git-commit/
5. **Developer Tip: Keep Your Commits "Atomic"**, Sean Patterson: https://www.freshconsulting.com/atomic-commits/

Obrigado!

https://matheustavares.gitlab.io