

Sincronizando seu projeto



- Até aqui, moleza, mas o que queremos de fato no nosso dia a dia é criar novos arquivos em nosso projeto, realizar alterações, versionarmos toda movimentação no cenário de nosso projeto através do comando **git commit** (se estivermos localmente) ou ainda com o comando **commit** se estivermos no GitHub e sincronizamos com nosso repositório no GitHub
- Então vamos ao trabalho, para isso, vamos ao Visual Studio Code, selecione a pasta do seu projeto e vamos adicionar um novo arquivo, veja como ...

Sincronizando seu projeto



- Uma vez selecionada a pasta de trabalho criemos um novo arquivo

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with sections for 'OPEN EDITORS' (containing 'Welcome') and 'PROJETOLAB' (containing '.gitignore' and 'README.md'). The center is the 'OPEN EDITORS' view, which lists 'Welcome' and 'index.htm' (the latter being the active editor). The right pane displays the contents of 'index.htm':

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9
10 </body>
11 </html>
```

Sincronizando seu projeto



- A partir daí podemos trabalhar normalmente, veja o fluxo padrão a seguir ...

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index.htm

nothing added to commit but untracked files present (use "git add" to track)

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git add index.htm

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   index.htm
```

Sincronizando seu projeto



- Arquivo adicionado na Stage, estamos prontos para realizar um commit no nosso repositório local
- Vamos agora realizar um **git commit -m 'mensagem'**

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git commit -m 'arquivo index.htm criado'
[master 81f3a2b] arquivo index.htm criado
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.htm

accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Sincronizando seu projeto



- Observe no, entanto, que nossa ação ficou restrita à nossa máquina, veja nosso repositório no GitHub
- Apenas nosso primeiro commit realizado ainda no ambiente GitHub estará presente

A screenshot of a GitHub repository page. At the top, there are summary statistics: 1 commit, 1 branch, 0 packages, 0 releases, and 1 contributor. Below this, a button for 'New pull request' is visible. On the left, a large hand cursor icon points towards the commit count. A red box highlights the '1 commit' statistic. The main area shows a single commit by 'accolombinifake' with the message 'Initial commit'. This commit was made 2 hours ago and includes files '.gitignore' and 'README.md'. A green 'Clone or download' button is located at the top right of the commit list.

File	Message	Time
.gitignore	Initial commit	2 hours ago
README.md	Initial commit	2 hours ago

Sincronizando seu projeto



- Agora no meu ambiente local, lembra, nós clonamos nosso repositório do GitHub, veja o que acontece quando usamos o comando **git log**
- Ele trará dois commits, um realizado no GitHub (referenciado como origin/master, origin/HEAD) e o outro realizado em nosso ambiente local, observe(referenciado como HEAD → master) ...

A screenshot of a terminal window titled "TERMINAL". The window shows the command \$ git log and its output. Two commits are listed. The first commit is highlighted with a green rounded rectangle and labeled "Initial commit". The second commit is also highlighted with a green rounded rectangle and labeled "arquivo index.htm criado".

```
DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL  
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)  
$ git log  
commit 81f3a2b64beb1ccc0c5ac5394827f0566cce00fd (HEAD -> master)  
Author: accolombinifake <accolombinifake@hotmail.com>  
Date:   Wed Mar 18 17:03:20 2020 -0300  
  
        arquivo index.htm criado  
  
commit 8d51b1123f3ed9cf869d6ce97915a0fa3dc96d4e (origin/master, origin/HEAD)  
Author: Angelo <59621376+accolombinifake@users.noreply.github.com>  
Date:   Wed Mar 18 15:12:49 2020 -0300  
  
        Initial commit
```

Sincronizando seu projeto



- Para fazermos o sincronismo dessas alterações com o repositório do GitHub, vamos precisar de mais um comando, o **git push**
- Ao usar esse comando pela primeira vez, será aberta uma caixa de diálogo, é preciso que você tenha permissão para publicar no repositório. Preencha com seus dados

A screenshot of a terminal window. On the left, a GitHub login dialog box is displayed, prompting for 'Username or email' and 'Password'. Below the dialog, there are links for 'Login' and 'Cancel', and options for 'Sign up' and 'Forgot your password?'. On the right side of the terminal, a git commit history is shown:

```
" content="width=device-width, initial-scale=1"> >
" ERMINAL
users/Angelo/GIT/projetolab (master)
7f0566cce0fd (HEAD -> master)
nifake@hotmail.com>
Date: Wed Mar 18 17:03:20 2020 -0300
arquivo index.htm criado
commit 8d51b1123f3ed9cf869d6ce97915a0fa3dc96d4e (origin/master, origin/HEAD)
Author: Angelo <59621376+accolombinifake@users.noreply.github.com>
Date: Wed Mar 18 15:12:49 2020 -0300
Initial commit
accol@DESKTOP-TMBPLKD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git push
```

A large yellow curved arrow points from the top-left towards the GitHub login dialog. A red rectangular box highlights the command `$ git push` at the bottom of the terminal window.

Sincronizando seu projeto



Nota: qualquer pessoa pode clonar um repositório do GitHub público, mas para você publicar no diretório é preciso que tenha permissão, falaremos mais sobre isso nas próximas aulas

- Uma vez tendo a permissão, observe o que acontece...

```
acco1@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 318 bytes | 318.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/acco1ombinifake/projetolab.git
  8d51b11..81f3a2b  master -> master
```

Sincronizando seu projeto



- Para conferir se tudo está ok, vamos agora ao nosso repositório remoto no GitHub, observe ...
- Temos agora dois commits e nosso novo arquivo se encontra no repositório

The screenshot shows a GitHub repository page for the user 'accolombinifake' named 'projetolab'. The page displays the following information:

- Code tab:** Selected tab.
- Issues:** 0
- Pull requests:** 0
- Actions:** 0
- Projects:** 0
- Wiki:** 0
- Security:** 0
- Insights:** 0
- Settings:** Link

Project description: Projeto para ações práticas em laboratório, sincronizando Git com GitHub. [Edit](#)

Statistics:

- 2 commits (highlighted with a red box)
- 1 branch
- 0 packages
- 0 releases
- 1 contributor

Branch: master [New pull request](#)

Commits:

File	Type	Message	Time
.gitignore	Initial commit		2 hours ago
README.md	Initial commit		2 hours ago
index.htm	arquivo index.htm criado	Latest commit 81f3a2b 26 minutes ago	26 minutes ago
README.md			

Sincronizando seu projeto



- Vamos agora fazer o caminho inverso, queremos a partir de alterações realizadas no nosso repositório remoto, atualizar nosso diretório local, para isso, vamos precisar do comando **git pull**

Nota: antes de fazer o git push você precisa ter a certeza de que seu diretório está atualizado com as últimas modificações do seu repositório remoto GitHub, daí a necessidade de usar o **git pull** para assegurar o sincronismo entre o diretório local e o repositório remoto

Sincronizando seu projeto



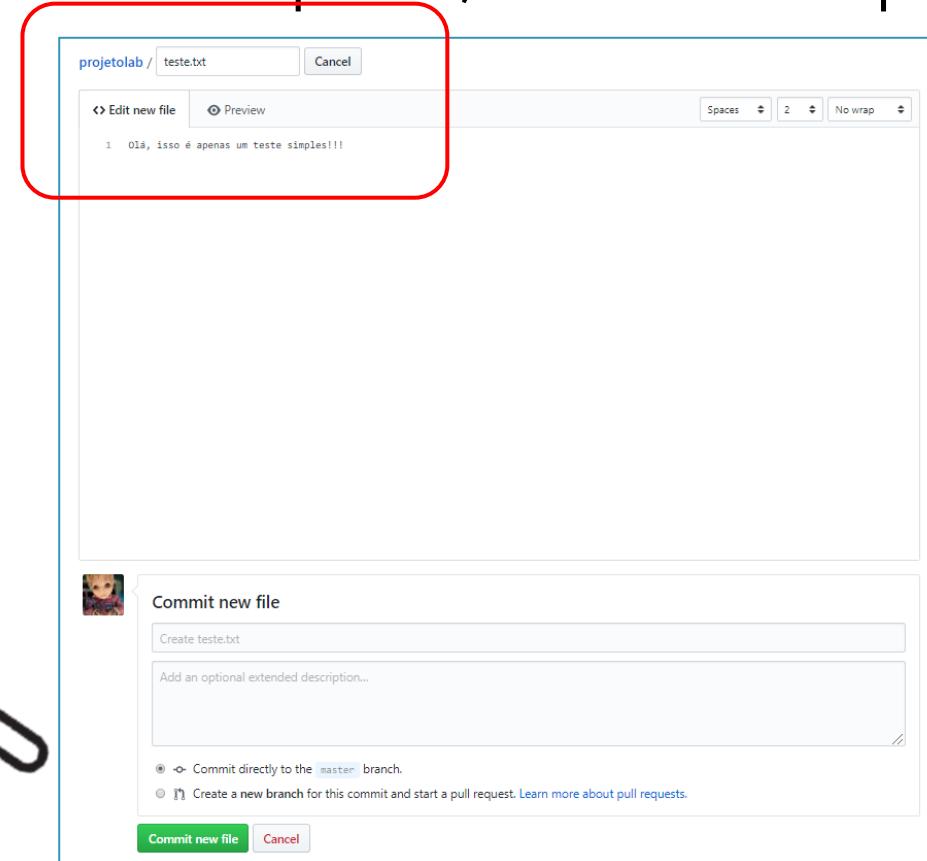
- Para nosso laboratório, vamos simular esse evento promovendo alterações em nosso repositório na nuvem GitHub
- Vamos criar um novo arquivo, para isso clicamos em criar novo arquivo, veja ...

The screenshot shows a GitHub repository page for 'acolombinifake / projetolab'. The page displays basic repository statistics: 2 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. Below these stats, there are buttons for 'Branch: master', 'New pull request', 'Create new file' (which is highlighted with a red box and has a hand cursor pointing at it), 'Upload files', 'Find file', and 'Clone or download'. The repository's history is listed, showing the creation of 'index.htm' and 'README.md' files, and a commit for '.gitignore'. The most recent commit was made 2 hours ago.

Sincronizando seu projeto



- Uma vez criado o arquivo, não se esqueça, faça o **commit**



Sincronizando seu projeto



- Observe que este novo arquivo está no GitHub, mas não se encontra ainda no repositório local

A screenshot of a GitHub repository page for 'acolombinifake / projetolab'. The page shows a commit history with the following details:

Author	Commit Message	Time Ago
acolombinifake	Create teste.txt	Latest commit 891579b 26 seconds ago
.gitignore	Initial commit	4 hours ago
README.md	Initial commit	4 hours ago
index.htm	arquivo index.htm criado	2 hours ago
teste.txt	Create teste.txt	26 seconds ago

- Observe que este arquivo não se encontra em nosso diretório local

A screenshot of a file explorer window titled 'GIT > projetolab'. The table lists the contents of the repository:

Nome	Data de modificação	Tipo
.git	18/03/2020 17:03	Pasta de arquivos
.gitignore	18/03/2020 15:51	Documento de Texto
index.htm	18/03/2020 16:13	Chrome HTML Documento
README.md	18/03/2020 15:51	Arquivo MD



Sincronizando seu projeto



- Para sincronizarmos nosso diretório local, basta acessarmos nosso terminal e usar o comando **git pull**

Observe que ele trouxe um arquivo, sendo este adicionado ao diretório local

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 691 bytes | 4.00 KiB/s, done.
From https://github.com/accolombinifake/projetolab
  81f3a2b..891579b  master      -> origin/master
Updating 81f3a2b..891579b
Fast-forward
  teste.txt | 1 +
  1 file changed, 1 insertion(+)
  create mode 100644 teste.txt
```

Sincronizando seu projeto



- Observe agora quando fazemos uso do comando **git log** como está nosso diretório atual

Temos em nosso diretório local os três commits realizado e estamos exatamente como nosso repositório no GitHub

```
accol@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)
$ git log
commit 891579bfaff5811ea7df824afd284d72e36bcfeb (HEAD -> master, origin/master, origin/HEAD)
Author: Angelo <59621376+accolombinifake@users.noreply.github.com>
Date:   Wed Mar 18 18:42:36 2020 -0300

    Create teste.txt

commit 81f3a2b64beb1ccc0c5ac5394827f0566cce00fd
Author: accolombinifake <accolombinifake@hotmail.com>
Date:   Wed Mar 18 17:03:20 2020 -0300

    arquivo index.htm criado

commit 8d51b1123f3ed9cf869d6ce97915a0fa3dc96d4e
Author: Angelo <59621376+accolombinifake@users.noreply.github.com>
Date:   Wed Mar 18 15:12:49 2020 -0300

    Initial commit
```

Uma das ferramentas mais
importantes da plataforma
GitHub

O que são as
branches?



Branches - introdução



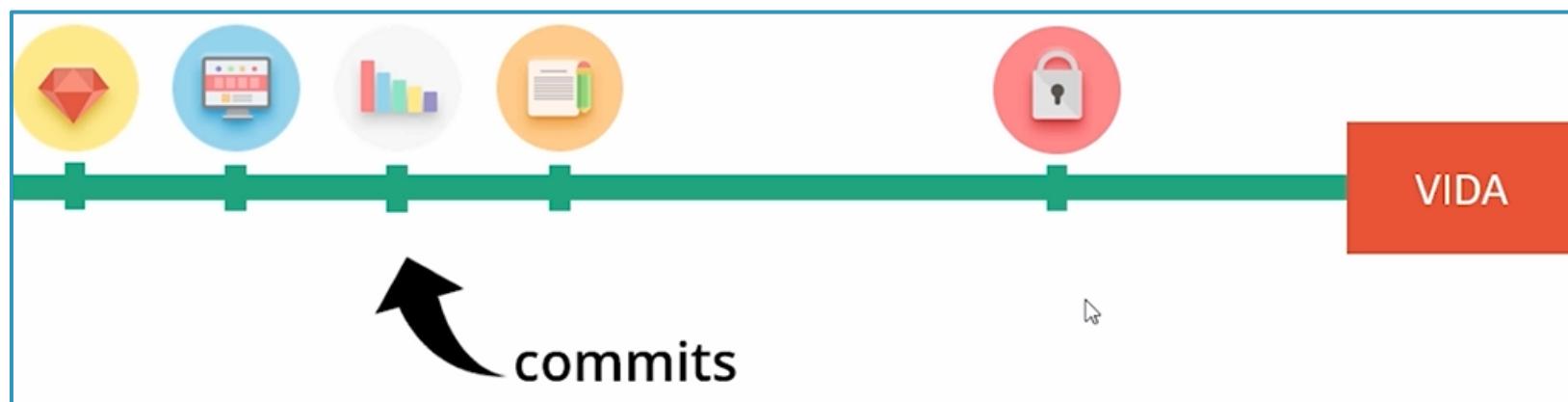
- Para facilitar podemos pensar em Branches como sendo linhas do tempo, isso mesmo no plural, podemos seguir por caminhos diferentes → são as Branches. Até agora, trabalhamos apenas com a nossa **Branch master**
- Em algum momento devemos convergir todas as Branches para a linha do tempo de nosso projeto, sim esse deve ser única
- As Branches são um recurso poderoso que nos ajuda a tentarmos alternativas ou alterações em nosso projeto num ambiente separado (uma branch separada), isso nos permite manter um controle mais assertivo e minimizar os riscos de nosso projeto



Branches - introdução



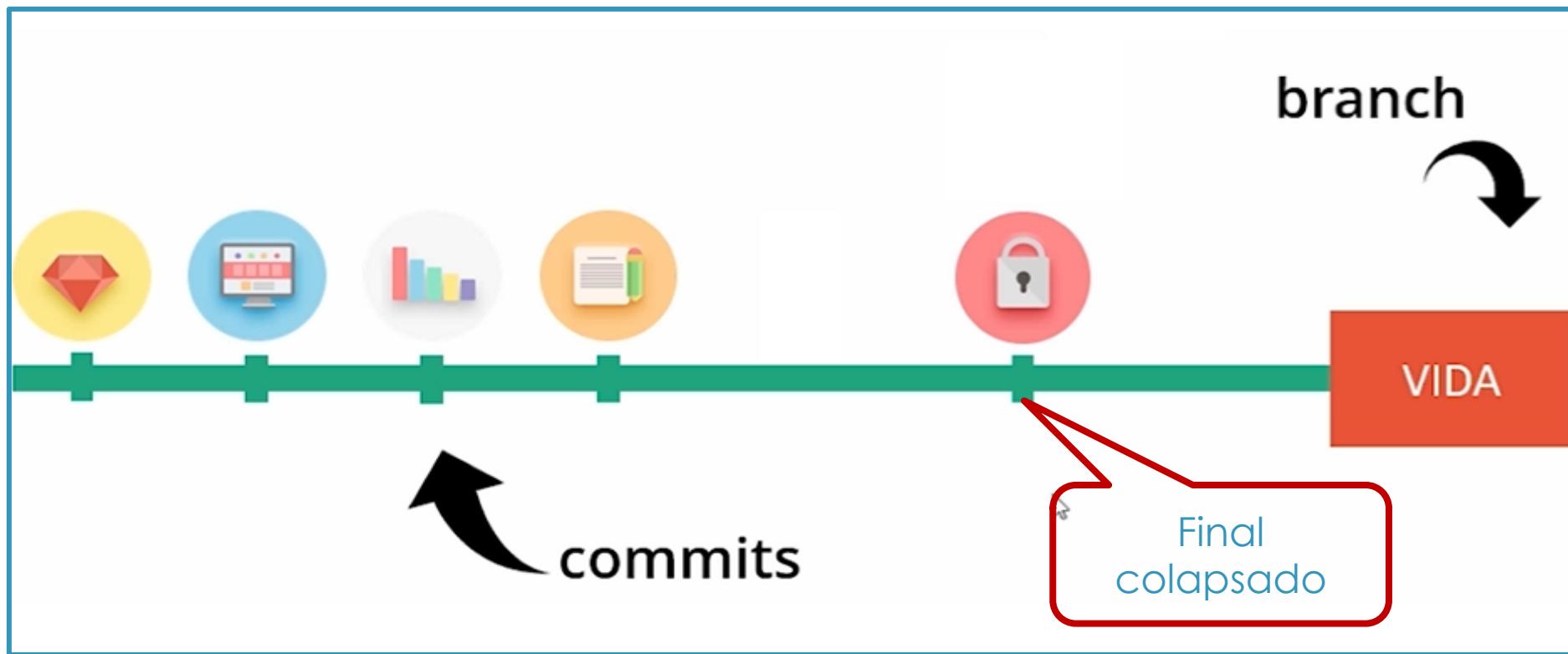
- Observe a linha do tempo a seguir, podemos imaginar que para cada evento nela assinalado temos um commit realizado



Branches - introdução



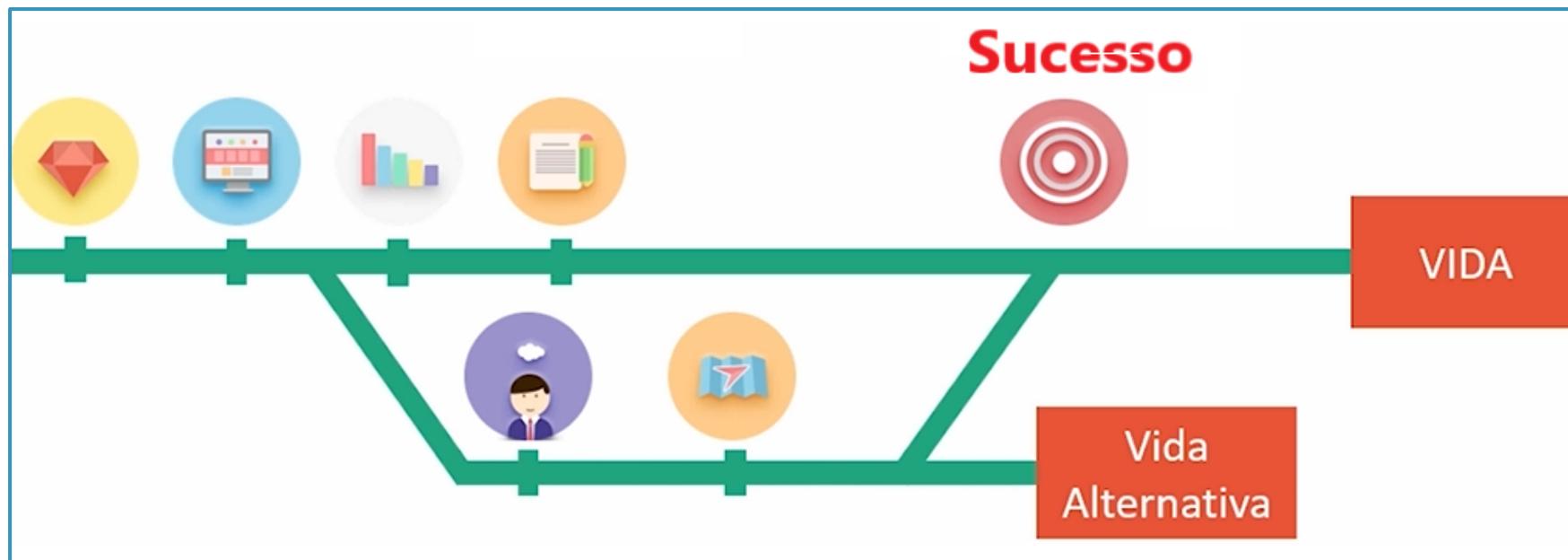
- A **linha do tempo** para o Git recebe o nome de **Branch**



Branches - introdução



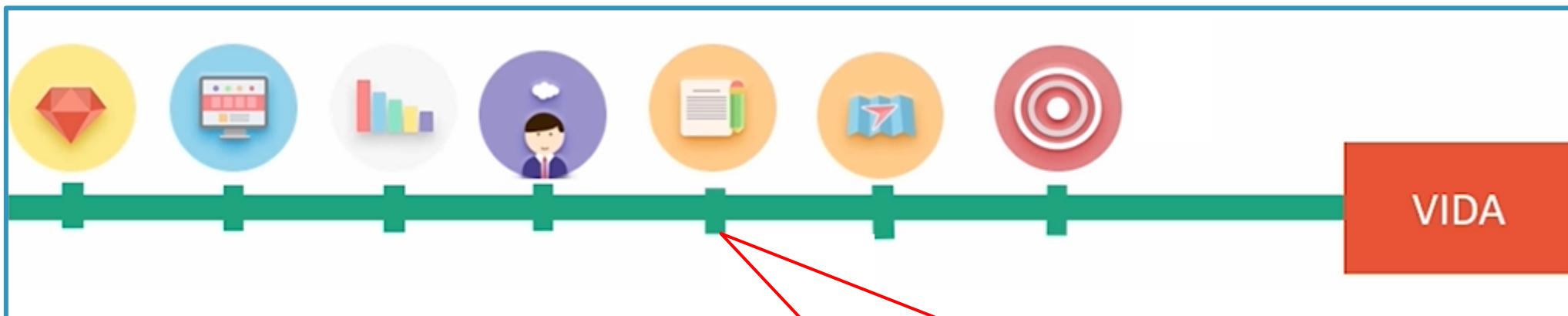
- Ao que parece nossa linha do tempo não está convergindo para um bom fim (final colapsado)
- Felizmente para o Git podemos alterar esse final, criando Branches alternativas



Branches - introdução



- Ao realizar os ajustes usando uma Branch alternativa (vida alternativa), o Git permite que você integre todos os commits realizados na Branch alternativa à sua branch master (vida). No final das contas, tudo irá parecer que seu projeto transcorreu de forma linear



Branch master restabelecida e todo histórico (commits) assegurados

Branches - introdução



- Vamos falar um pouco sobre a **Branch Master**, ela nos acompanhou até aqui, mas pouco falamos a seu respeito
- Chegou o momento de conhecê-la um pouco mais de perto



```
acco1@DESKTOP-TMBP1KD MINGW64 /d/Users/Angelo/GIT/projetolab (master)  
$ []
```

A screenshot of a terminal window showing a command-line interface. The prompt shows the user's name, the computer name, the operating system (MINGW64), the directory path, and the branch name "(master)". A red circle highlights the "(master)" text.

Sim, isso mesmo, é dela que estamos falando

Branches - introdução

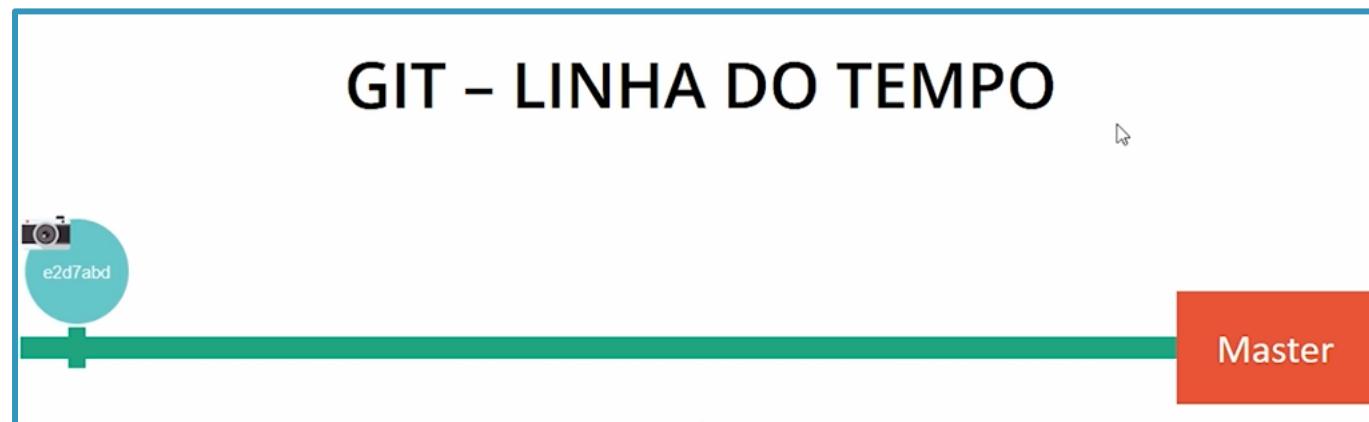


- Como já perceberam sempre que iniciamos nosso Git (ela representa a linha do tempo do nosso projeto) com o comando **git init** é criada a **Branch Master**
- É na Branch Master que todos os nossos commits serão registrados, a menos que, em algum momento você crie uma Branch alternativa
- Cada vez que efetuamos um commit é como se estivéssemos tirando uma foto do status da linha do tempo do nosso projeto naquele instante

Branches - introdução



- Para o Git a linha do tempo do nosso projeto é fator crítico de sucesso, podemos navegar por ela revisitando os commits realizados sempre que houver uma demanda
- Ainda é uma forma de assegurar que todos os eventos relevantes (marcos) que geraram commits não se percam



Branches - introdução



- Falamos muito dos commits, mas afinal, o que são esses commits?
- O que são os códigos nele registrados? Olhando um commit por dentro



O nosso Snapshot nos traz:

1. Commit id → e2d7a...
2. Autor → accolombinifake
3. Data → Thursday March 19 ...
4. Message → 'first commit'

Branches - introdução



Nota: A partir deste commit inicial, o Git cria uma árvore que rastreia todos os nossos commits, o que lhe permitirá acesso a qualquer desses ramos no instante em que desejar

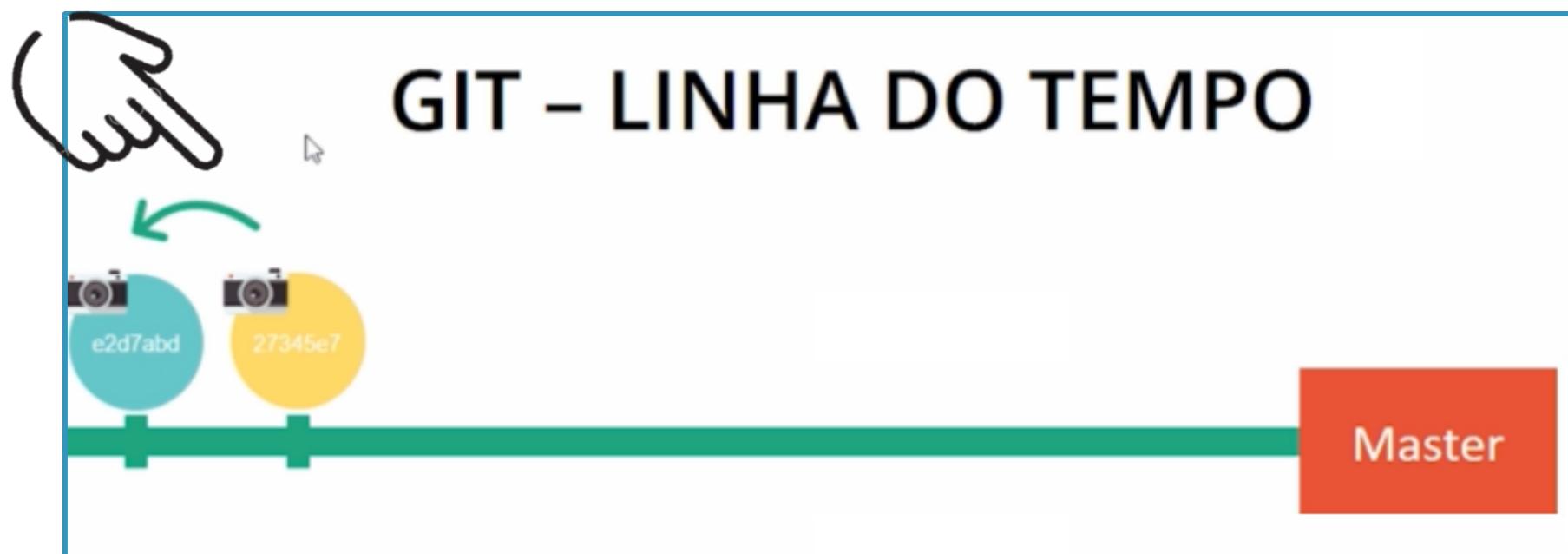
- Podemos executar vários **commits**, sempre que o **Git** adiciona um novo commit ele cria um código de rastreio que nos diz que o commit que estamos criando possui um pai, e aponta para seu antecessor



Branches - introdução



- Observe na figura a seguir que um segundo commit foi gerado. Em seguida vamos abrir esse commit (snapshot) para observá-lo mais de perto



Branches - introdução



- Observe na imagem a seguir que um novo campo surge. Esse campo, **parent**, aponta para o commit pai, ou seja, para o commit imediatamente anterior na árvore de rastreio do Git

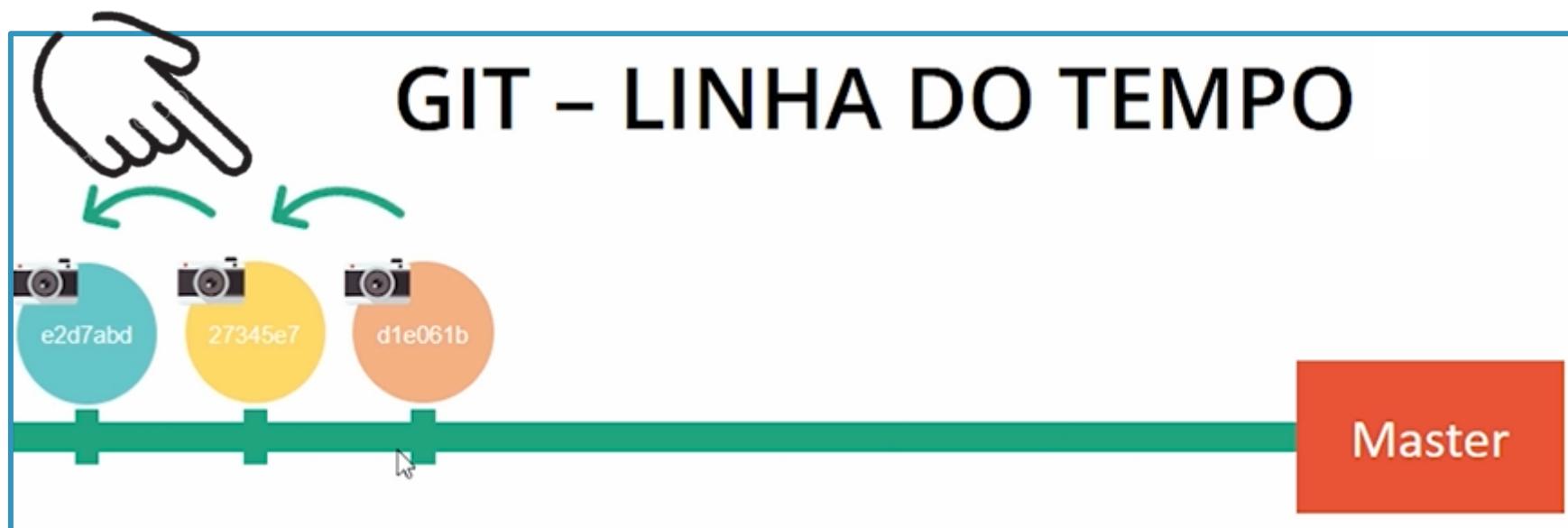


É desta forma que o Git consegue caminhar por seu projeto, rastreando cada etapa registrada em um snapshot

Branches - introdução



- Observe que esse processo se repete, a linha do tempo (nossa Branch Master) é contínua e outros commits serão adicionados, sempre com a mesma dinâmica, ou seja, apontando para o commit anterior (pai)



Branches – FAST FORWARD

Até aqui trabalhamos apenas com uma Branch, a Branch Master, vamos agora subir um pouco nosso nível de trabalho, vamos criar novas Branches para garantir um controle mais preciso de nosso projeto

Vamos fazer a seguinte simulação: criamos três commits em nossa Branch Master, e em seguida resolvemos criar uma nova Branch com o propósito de desenvolver novas alterações no projeto sem comprometer nossa Branch Master (imagine um caso em que esteja testando uma nova hipótese, algo que poderá contribuir muito com seu projeto) são muitas as possibilidades, ok

....

git

Git



Branches – FAST FORWARD

- Uma grande vantagem em usar esse recurso é o seguinte:
- Imagine um projeto sendo desenvolvido por um time remoto. Manter a Branch Master intacta garante que todos que estão trabalhando remotamente na Branch Master não sejam afetados por um possível teste, ou alteração que não deu certo. Estes testes estão na sua nova Branch, que ainda não foi incorporada na Branch Master
- Em outras palavras, a Branch Master não muda, veja a figura a seguir

Branches – FAST FORWARD



Esta linha não muda!



Criar nova branch neste ponto

Branches – FAST FORWARD

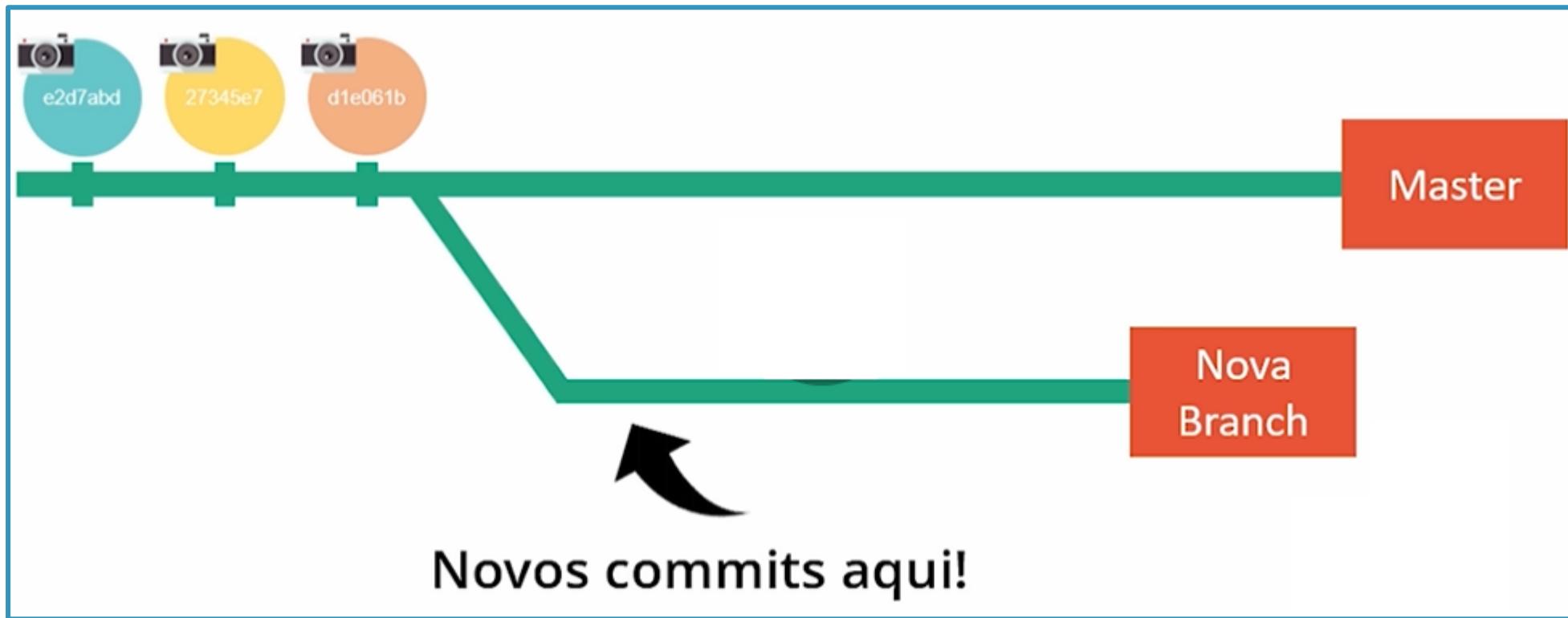
- Como fica nosso projeto então, veja na figura a seguir. Note que você irá trabalhar na nova Branch criando commits tudo como se estivesse na Branch Master.
- Você está apenas resguardando a Branch Master para que ela não receba nenhum commit desnecessário, ou até mesmo, lá na frente você poderá concluir que essas alterações não devam fazer parte de seu projeto, devendo ser até mesmo descartadas



Branches – FAST FORWARD



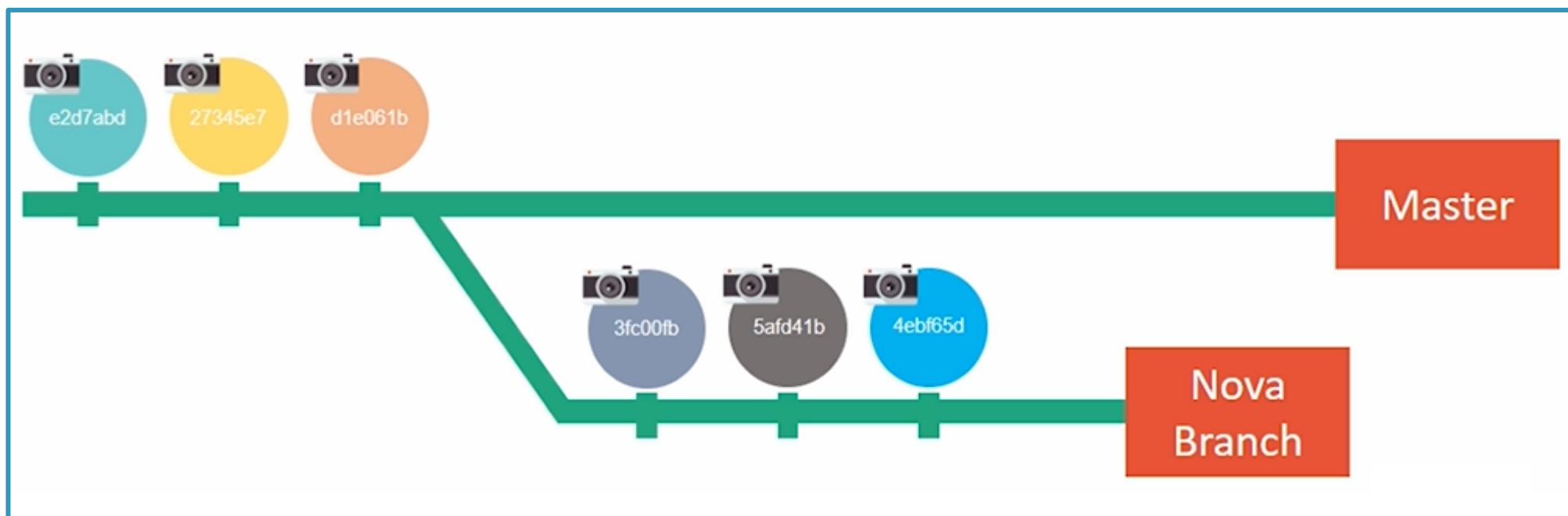
- Observe:



Branches – FAST FORWARD



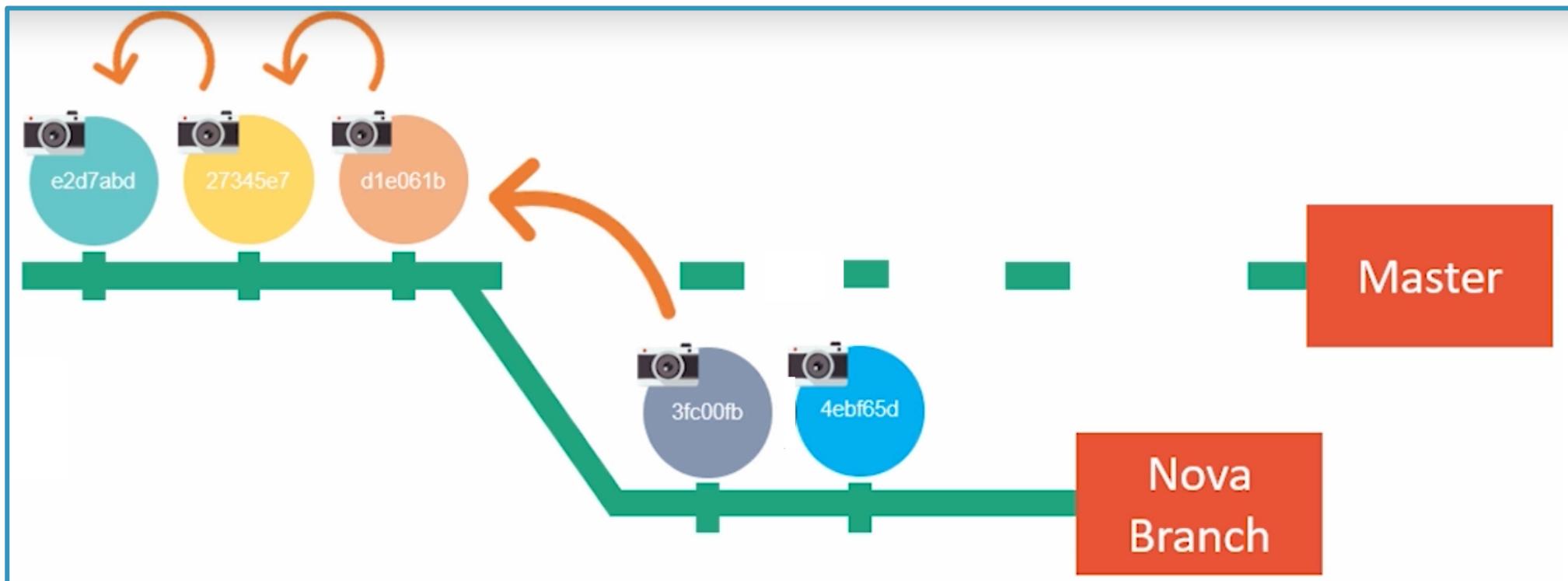
- Novos commits serão adicionados, e a lógica continua a mesma, ou seja, cada commit aponta para o anterior



Branches – FAST FORWARD



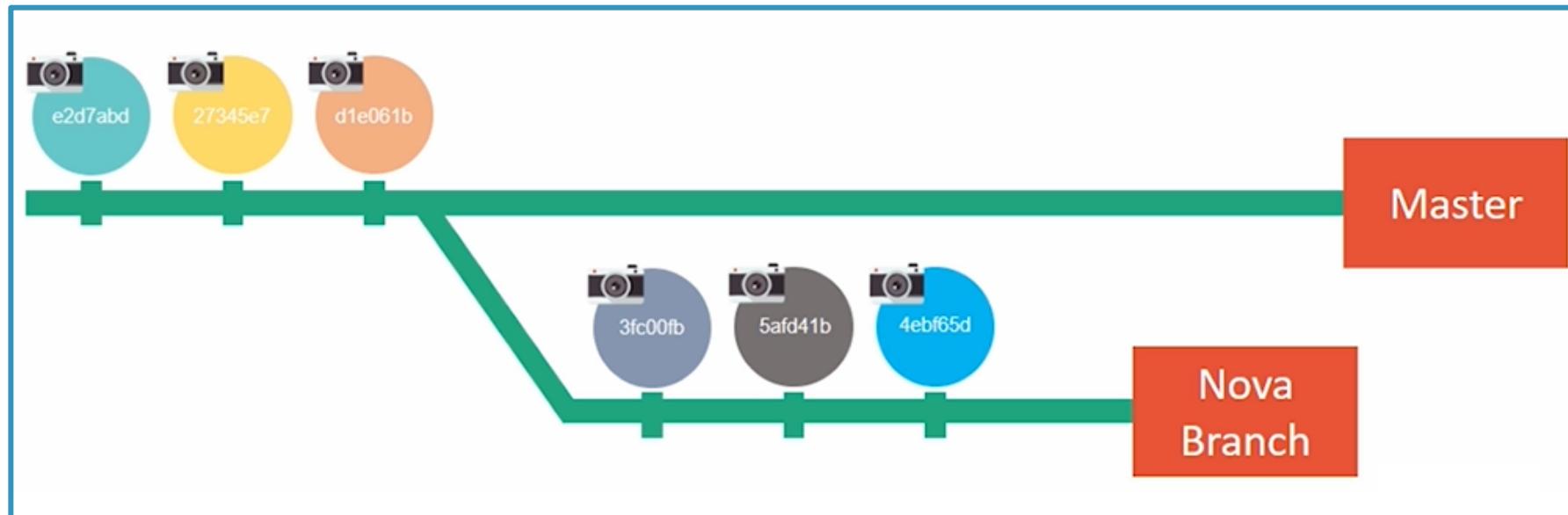
Importante: a nova Branch é composta dos três commits anteriores e os dois commits na Nova Branch, assim, poderemos trabalhar na Nova Branch como se estivéssemos na Branch Master



Branches – FAST FORWARD



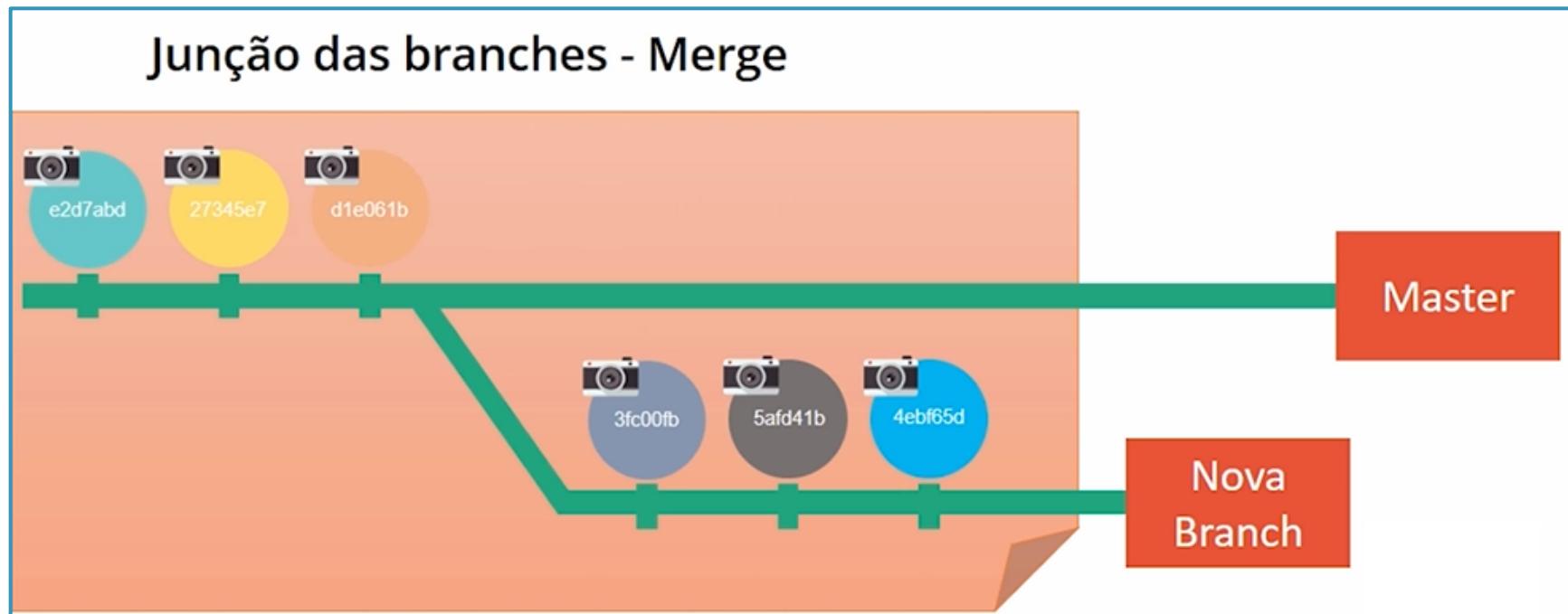
- Vamos avaliar algumas possibilidades:
 - No primeiro caso, vamos imaginar que três commits foram realizados na Branch Master
 - Criamos uma Nova Branch e realizamos novos commits
 - Neste caso, nenhum commit aconteceu na Branch Master enquanto os trabalhos aconteceram na Nova Branch



Branches – FAST FORWARD



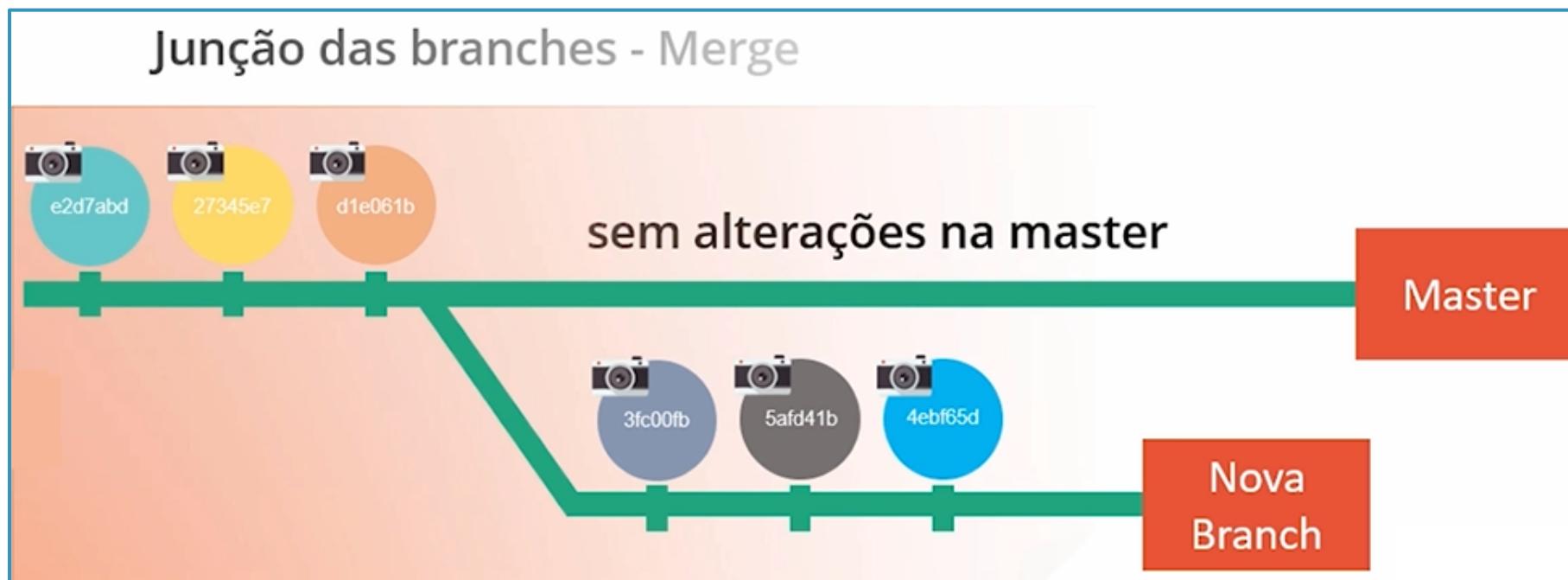
- Imagine que os testes na Nova Branch foram ok e nós queremos agora voltar todo histórico (commits) para a Branch Master, a isso, dá se o nome de Junção das Branches ou Merge



Branches – FAST FORWARD



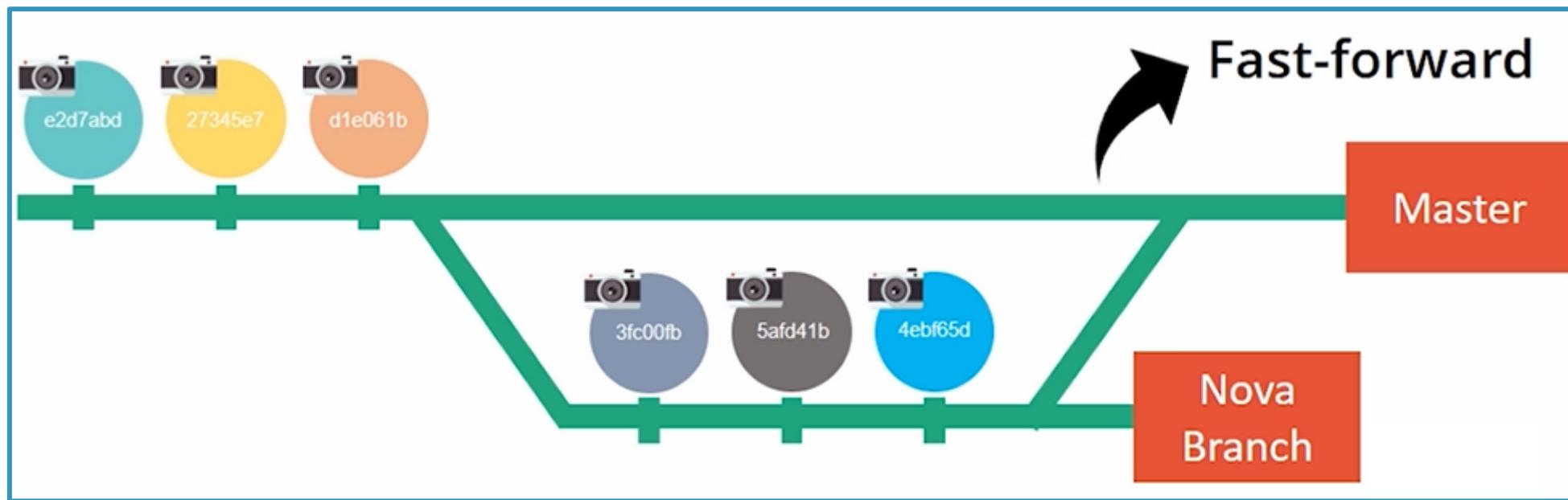
- Em outras palavras, o que queremos é restabelecer a Branch Master, assegurando a plenitude do histórico do seu projeto e neste caso, note que não houveram novos commits na Branch Master



Branches – FAST FORWARD



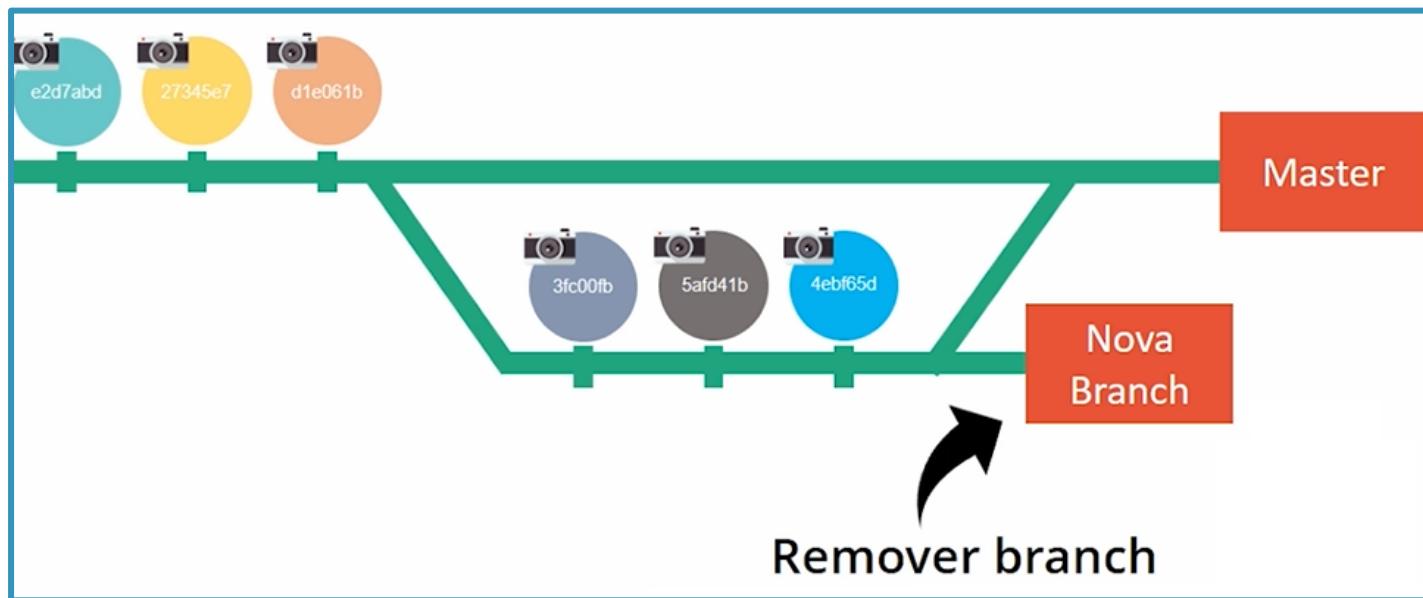
- A este tipo de Merge, dá se o nome de Fast-forward, veja na figura
- Nesta Merge os commits da Nova Branch serão inseridos na linha do tempo da Branch Master na sequência correta



Branches – FAST FORWARD



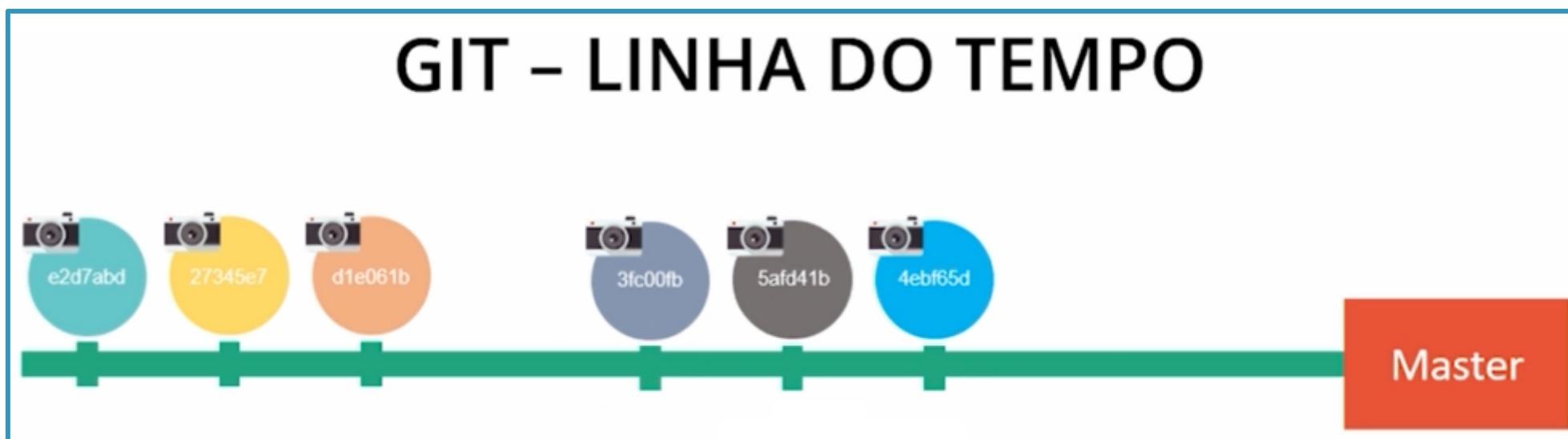
- Nesta opção de Merge o Git reunirá os novos commits na Branch Master sem qualquer problema, sem precisar fazer maiores alterações, sem lançar mão de qualquer recurso adicional, no caso, a criação de novos commits
- Uma vez feito o Merge, seu histórico de projeto estará restabelecido e você, caso queira (recomendo) poderá excluir a Nova Branch (aquele que criou)



Branches – FAST FORWARD



- Uma vez feito o Merge teremos a impressão que todos os commits foram realizados na Branch Master, veja a figura, note que a linha do tempo do projeto se mantém intacta





Processo de trabalho FriendsLab

Rotina



Rotina de trabalho

1. Faça o clone do repositório do github Friendslab20
2. Gere as credenciais de trabalho – use git config user.name e git config user.email
3. Gere uma nova branch, sua branch de trabalho deverá ter um nome sugestivo, por exemplo, branch previsor, caso esteja trabalhando no módulo previsor e assim por diante
 1. Para criar um branch: git branch <nome_da_branch>
4. Você precisará ir para a branch criada, faça: git checkout <nome_da_branch>



Rotina de trabalho

5. Na nova branch, primeiro confira onde esta: git branch → o nome da nova branch deverá estar destacado com um asterisco * .
6. Na nova branch trabalhe normalmente, altere, crie, modifique, realize commits etc., enfim tudo o que aprendeu.
7. Por fim, quando estiver convicto de que seu trabalho está ok, volte para a branch master: git checkout master

Rotina de trabalho

8. Na branch master verifique: git branch você deverá estar na branch master, lembra do asterisco.
9. Agora você deverá fazer o merge da branch que estava trabalhando com a branch master: git merge <nome_da_branch_de_trabaho>
10. Apague a branch de trabalho: git d <nome_da_branch_de_trabalho>
11. Você agora deverá sincronizar com o repositório remoto: git pull origin master



Rotina de trabalho

- 12.Uma vez sincronizado é hora de você enviar toda sua produção para o repositório remoto: git push
- 13.A partir daí siga para seu próximo trabalho, sempre se lembrando de criar uma nova branch para trabalhar.



GitHub



Esta Foto de Autor Desconhecido está licenciado em [CC BY-NC](#)

Sucesso a todos!!!

Com o tempo
teremos mais ações
a serem
incorporadas em
nossa rotina

