

# Conhecendo o Github

---



Universidade Estadual Paulista, Júlio de Mesquita Filho - UNESP

[clayton.pereira@unesp.br](mailto:clayton.pereira@unesp.br)

- ☐ Github References
- ☐ Atlassian Tutorials
- ☐ Git Book
- ☐ Git Install
- ☐ Chaves SSH
- ☐ Manual Git
- ☐ Na linha de comando: `git help comando`



# Sistemas de Controle de Versão

---

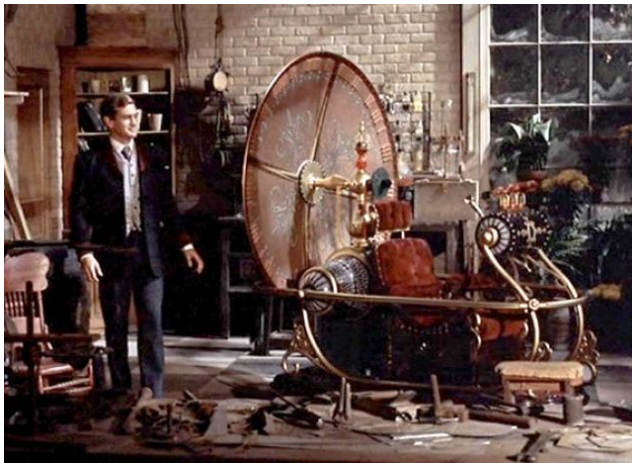


Figure: The time Machine Movie (1960)



# Sistemas de Controle de Versão

---



Figure: Controle de Versões



# Sistemas de Controle de Versão

---

## O que é um Sistema de Controle de Versão



Figure: Sistema de Controle de Versões



## Motivação

- ☐ Gerência de alterações no código-fonte **com o passar do tempo**;
- ☐ Mantém registro de todas as modificações no código;
  - Utiliza alguma forma especial de Banco de dados
- ☐ Uma máquina do tempo:
  - Comparar versões anteriores
- ☐ Rastreando de cada alteração individual feita por cada contribuinte, **evitando conflito com trabalhos simultâneos**;

### Importante

As alterações feitas em uma parte do software podem ser **incompatíveis** com aquelas feitas por **outro desenvolvedor trabalhando ao mesmo tempo**. Esse problema deve ser descoberto e resolvido **sem bloquear o trabalho do restante da equipe**.

## Motivação

- ☐ Compatível com o fluxo de trabalho dos desenvolvedores;
- ☐ Não fazer imposições sobre o modo de trabalho;
- ☐ Independência de plataforma;
- ☐ Parte essencial do dia a dia de equipe distribuídas de software;
- ☐ Rastreabilidade: capacidade de identificar quando um determinado bug foi inserido no código;
- ☐ Controle de Ramificações;



Git

---

- ☐ Conceitos
- ☐ Comandos
- ☐ Github
- ☐ Fluxos para Gerenciamento de Código

## O que é o Git?

- ☐ É um sistema de controle de versão de arquivos que rastreia o histórico de mudanças;
- ☐ Sistema de controle de versão moderno mais usado no mundo hoje;
- ☐ Desenvolvido em 2005 por Linus Torvalds;
- ☐ Arquitetura distribuída;
  - Em contraste com sistemas centralizados como *CVS* e *Subversion*
  - Toda cópia de trabalho é um repositório;
- ☐ Se concentra no conteúdo do arquivo em vez de nomes;
- ☐ Projetado com desempenho, segurança e flexibilidade em mente

## Características elementares do Git

- ☐ Provê um controle de versionamento completamente distribuído.
- ☐ Não precisa de uma conexão constante com um repositório central.
  - Desenvolvedores podem trabalhar em qualquer lugar e colaborar de **forma assíncrona**;
- ☐ Permite gerenciar as alterações localmente, antes de enviar para uma instância remota;

## Durante o desenvolvimento do software, buscamos saber:

- ☐ Quando e o que foi mudado?
- ☐ Por que foi mudado?
- ☐ Quem fez a mudança?
- ☐ Essa mudança poderá ser reproduzida?

## Quatro pontos básicos

- ☐ Identificação
- ☐ Documentação
- ☐ Auditoria
- ☐ Controle



**ONE DOES NOT SIMPLY**

**UNDERSTAND GIT**

MemesHappen

## Exemplo:

- ☐ Estou trabalhando em um código que está no meu Dropbox;
- ☐ As alterações no arquivo são executadas com sucesso.
- ☐ Salvo o arquivo no Dropbox



Figure: Salvando alterações no arquivo no Dropbox

## Exemplo:

- ☐ Seu amigo também quer editar o código;
- ☐ O mesmo arquivo é baixado pelos dois ao mesmo tempo;
- ☐ Você edita e salva o arquivo no Dropbox;
- ☐ Seu amigo faz a mesma coisa e sobrescreve suas alterações;

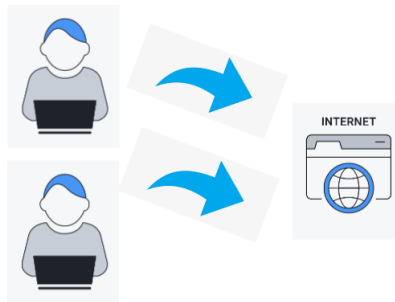


Figure: Salvando alterações no arquivo no Dropbox

## Exemplo:

- Controle de versão faz o **"merge"** entre as alterações

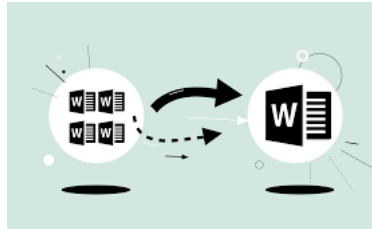


Figure: Merge in files

## Git é Distribuído

- Um repositório no servidor e cada desenvolvedor possui uma cópia do arquivo

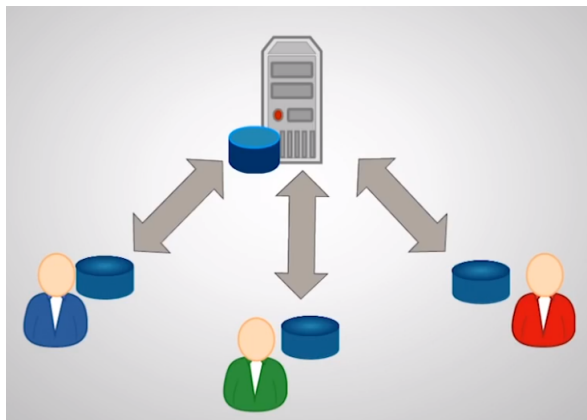


Figure: Repositório Git

# Controle de Versão

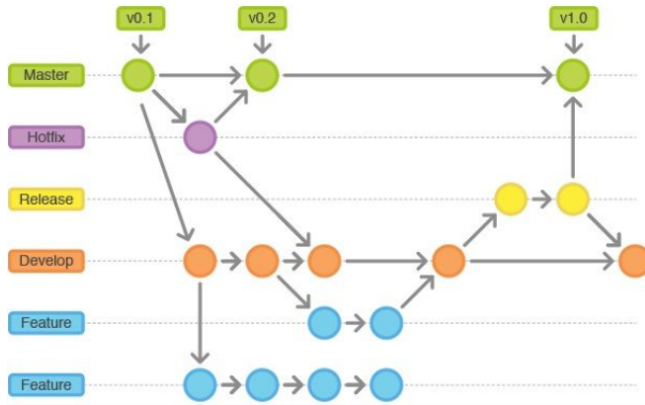


Figure: Estados do código (commits)

## Versionamento do Git

- ❑ **Master:** É o *branch* (ramo) que possui a última versão produtiva do código.
- ❑ **Release:** É o *branch* que contém os novos recursos finalizados que estão sendo desenvolvidos para o próximo release para que ao iniciar um novo você possa baixar todos os anteriores caso tenham alguma dependência.
- ❑ **Develop:** É o *branch* que contém as funcionalidades em desenvolvimento em uma iteração, este *branch* posteriormente fará parte do Release através de um *pull request*.
- ❑ **Features:** É o *branch* que contém o recurso que você está trabalhando pessoalmente (vários desenvolvedores podem trabalhar em um recurso), ele deve ser enviado para desenvolvimento por meio de uma solicitação completa, geralmente aprovada pelo líder técnico.
- ❑ **Hotfix:** É o *branch* que contém alterações urgentes no *master* que permitem corrigir um *bug* ou resolver um erro, deve ser enviado ao *master* e todos os desenvolvedores devem ser notificados para que possam atualizar seus *branches*.

## Setup Necessário - Chaves SSH

- ☐ Secure Shell Protocol
- ☐ Transferência remota de arquivos, gerenciamento de rede e acesso remoto ao sistema operacional.
- ☐ Usa um par de chaves para iniciar um handshake seguro entre partes remotas;
- ☐ Criptografia Assimétrica;

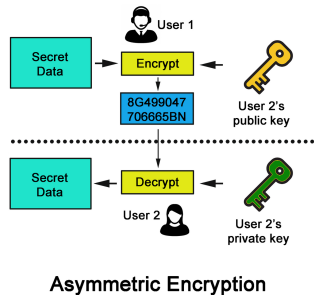


Figure: Criptografia Assimétrica



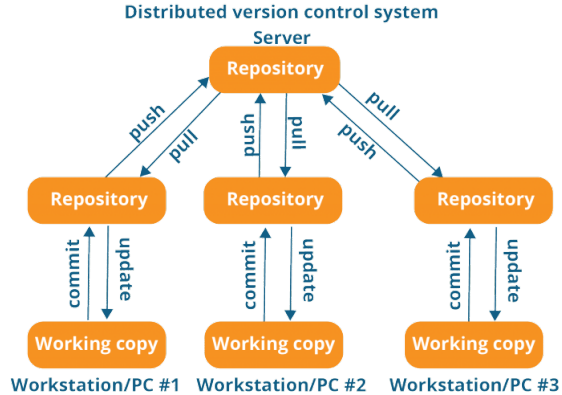


Figure: Repositórios Distribuídos

## O que é um repositório Git?

- ☐ Armazenamento virtual para projetos;
- ☐ Diretório especial `.git`;
  - contém todos os metadados Git necessários para o novo repositório;
  - subdiretórios para objetos, referências;
  - Arquivo HEAD que aponta para o commit em uso no momento;

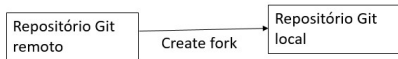


# Github

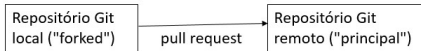
---

<https://github.com>

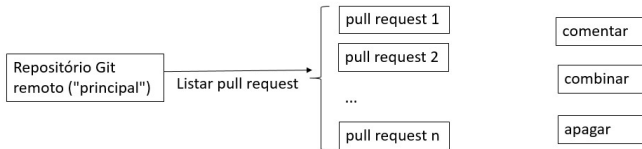
**Fork:** Criar um novo repositório a partir de um existente



**Pull request:** Enviar um pedido de alteração de um repositório replicado para o repositório original



**Aceitar, comentar ou apagar os pedidos de alteração enviados**



## O que é o GitHub?

- ☐ Servidor de repositório Git;
- ☐ Criado em 2008;
- ☐ +10 milhões de repositórios;
- ☐ +10 milhões de usuários



Figure: GitHub



Figure: Principais empresas mundiais de tecnologia.

# Empresas que usam o Git



Figure: Principais empresas mundiais de tecnologia.

claytontey / MachineLearning

Public

Unwatch 1

Star 2

Fork 0

<> Code

**Issues**

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Label issues and pull requests for new contributors

Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

Go to Labels

Filters is:issue is:open


Labels 7


Milestones 0


New issue

Welcome to issues!



 claytontey / **MachineLearning** Public Unwatch

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) 



Anotações

Write

Preview

H

B

I

≡

<>

🔗

☰

☷


☑


@


📎

↶


Implementação utilizando bibliotecas públicas.



Attach files by dragging & dropping, selecting or pasting them. 

 Styling with Markdown is supported


Submit new issue

 claytontey / **MachineLearning** PublicUnwatch


[Code](#) [Issues 1](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

## Anotações #1

Open claytontey opened this issue 1 minute ago · 0 comments

 claytontey commented 1 minute ago Owner 😊 ⋮

Implementação utilizando bibliotecas públicas.

 claytontey / MachineLearning

Public

Unwatch

1

Star

2

Fork

0

<> Code

Issues 1

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

master


1 branch

0 tags





Go to file


Add file


Code

 claytontey Update README.md

7facc8b on Sep 29, 2020 7 commits

 DataAnalysis-Spiral.ipynb	Add files via upload	2 years ago
 Meander_HandPD.csv	Add files via upload	2 years ago
 README.md	Update README.md	13 months ago
 Spiral_HandPD.csv	Add files via upload	2 years ago

 README.md



About

Introdução ao processo de Machine Learning para auxílio ao diagnóstico do mal de Parkinson


Readme

Releases

No releases published


Create a new release


# Github - Pull Request


 ai2-education-fiep-turma-4 / **git-katas** Public


forked from [eficode-academy/git-katas](#)


<> Code


 **Pull requests**

 Actions

 Wiki

 Security

 Insights

 Settings

Watch 0

Star 0

Fork 399


Filters ▾

Q is:pr is:open

Labels 9

Milestones 0

New pull request



## Welcome to pull requests!

Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should [create a pull request](#).

# Github - Pull Request

base repository: eficode-academy/git-katas

base: master

←

head repository: ai2-education-fiep-turma-4/git-k...

compare: master

✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

1 commit

1 file changed

0 comments

1 contributor

Commits on Oct 14, 2021

Update info

Verified d6d6a54

Showing 1 changed file with 2 additions and 0 deletions.

Unified Split

2 README.md

... @@ -1,6 +1,8 @@

1 ---

2 maintainer: randomsort

3 ---

4 +

5 + \*\*AI2 - Turma 4\*\*

6 # Git Katas

7

## Instalando o Git

□ Para instalação do Git precisamos do seguinte comando:

```
1 > apt-get update && apt-get install -y git
2 > git --version
```

## Inicializando um Repositório Git

- Definindo nome de usuário e email globalmente:

```
1 > git config --global user.name <name>
2 > git config --global user.email <email>
```

## Listando as chaves de SSH

□ Comando:

```
Terminal
(base) clayton@clayton-note:~$ ls -al ~/.ssh
total 68
drwx----- 2 clayton clayton 4096 out  4 07:34 .
drwxr-xr-x 108 clayton clayton 12288 out  3 18:22 ..
-rw----- 1 clayton clayton 1766 nov 18 2020 google_compute_engine
-rw-r--r-- 1 clayton clayton 402 nov 18 2020 google_compute_engine.pub
-rw-r--r-- 1 clayton clayton 222 nov 18 2020 google_compute_known_hosts
-rw----- 1 clayton clayton 464 out  4 07:31 id_ed25519
-rw-r--r-- 1 clayton clayton 102 out  4 07:31 id_ed25519.pub
-rw----- 1 clayton clayton 2655 jan 24 2022 id_rsa
-rw-r--r-- 1 clayton clayton 574 jan 24 2022 id_rsa.pub
-rw-r--r-- 1 clayton clayton 3552 set 19 10:19 known_hosts
-rw----- 1 clayton clayton 11238 dez  2 2021 known_hosts.old
-rw----- 1 clayton clayton 1766 abr 13 2020 pfsense
-rw-r--r-- 1 clayton clayton 402 abr 13 2020 pfsense.pub
(base) clayton@clayton-note:~$
```



## Conteúdo da chaves de SSH

□ Comando:

```
1 > cat ~/.ssh/id_ed25519.pub
```

## Inicializando um Repositório Git

### □ `git init`

- Converter um projeto existente e não versionado em um repositório do Git ou
- Inicializar um novo repositório vazio;
- cria um subdiretório `.git`

### □ `git clone`

- Maneira mais comum para os usuários obterem uma cópia de desenvolvimento;
- Em geral uma operação única;
- O repositório original pode estar localizado no sistema de arquivos local ou em protocolos com suporte a acesso por máquinas remotas;
  - Por exemplo, o protocolo SSH.

```
1 > git clone https://github.com/ai2-education-fiep-turma-5/resident-profiles.git
2 Cloning into 'meurepo'...
```

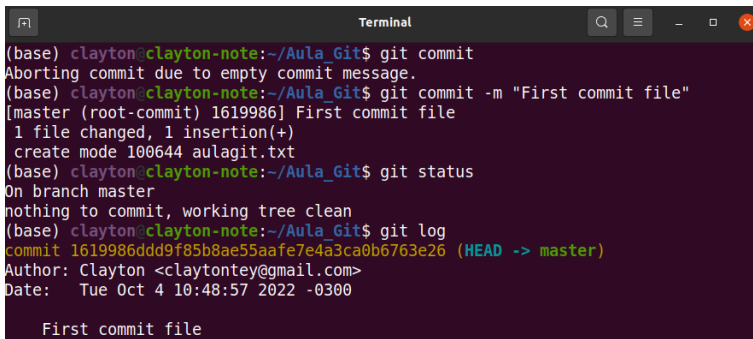
## Inicializando um Repositório Git

- Iniciando versionamento no Git:

```
(base) clayton@clayton-note:~/Aula_Git$ git init
Initialized empty Git repository in /home/clayton/Aula_Git/.git/
(base) clayton@clayton-note:~/Aula_Git$ touch aulagit.txt
(base) clayton@clayton-note:~/Aula_Git$ ls -la
total 20
drwxrwxr-x  3 clayton clayton  4096 out  4 10:34 .
drwxr-xr-x 110 clayton clayton 12288 out  4 10:33 ..
-rw-rw-r--  1 clayton clayton    0 out  4 10:34 aulagit.txt
drwxrwxr-x  7 clayton clayton  4096 out  4 10:34 .git
(base) clayton@clayton-note:~/Aula_Git$
```

## Inicializando um Repositório Git

- ❑ Iniciando versionamento no Git:

A terminal window titled "Terminal" with a dark background and light-colored text. It shows the execution of several Git commands in a shell. The prompt is "(base) clayton@clayton-note:~/Aula\_Git\$". The first command is "git commit", which fails with the message "Aborting commit due to empty commit message.". The second command is "git commit -m 'First commit file'", which succeeds, showing the commit hash [master (root-commit) 1619986], the commit message, and file changes. The third command is "git status", which shows the current state on the master branch. The fourth command is "git log", which shows the commit history. The output of "git log" shows a single commit with the message "First commit file".

```
(base) clayton@clayton-note:~/Aula_Git$ git commit
Aborting commit due to empty commit message.
(base) clayton@clayton-note:~/Aula_Git$ git commit -m "First commit file"
[master (root-commit) 1619986] First commit file
1 file changed, 1 insertion(+)
create mode 100644 aulagit.txt
(base) clayton@clayton-note:~/Aula_Git$ git status
On branch master
nothing to commit, working tree clean
(base) clayton@clayton-note:~/Aula_Git$ git log
commit 1619986ddd9f85b8ae55aafe7e4a3ca0b6763e26 (HEAD -> master)
Author: Clayton <claytontey@gmail.com>
Date: Tue Oct 4 10:48:57 2022 -0300

    First commit file
```

## Salvando Alterações

- ☐ O conceito de "salvar" é um processo com mais nuances do que salvar em um processador de texto;
- ☐ "salvar" é sinônimo do termo Git "*commit*"
- ☐ Os comandos: `git add`, `git status` e `git commit` são usados em combinação para salvar uma fotografia do estado atual de um projeto Git

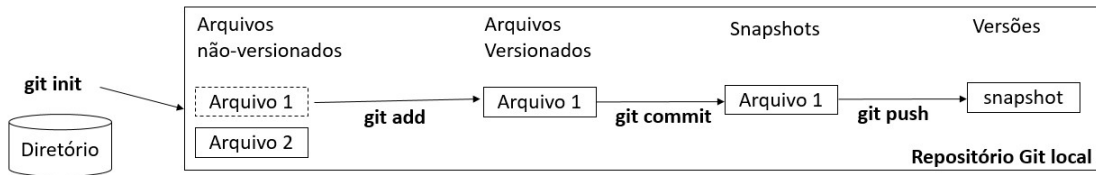
## Salvando Alterações

- ❑ O Comando `git add`:
  - Adiciona uma mudança no diretório de trabalho à área de teste
  - Informa ao Git que você deseja incluir atualizações para um arquivo específico no próximo *commit*;
  - Não afeta o repositório até que seja realizado o `git commit`
- ❑ Utilizado em conjunto com o comando `git status`;
- ❑ Um terceiro comando `git push` é essencial para um fluxo de trabalho Git colaborativo
  - Utilizado para enviar as alterações confirmadas para repositórios remotos para colaboração;
- ❑ `git pull` atualiza o repositório local com atualizações do repositório remoto;

## `git status`

Retorna:

- Lista de arquivos não-versionados
- Alterações realizados nos arquivos versionados



**Replica um repositório remoto**



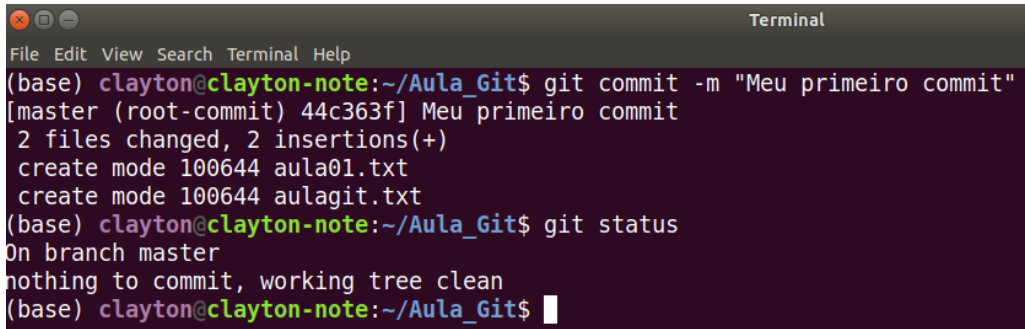
**Atualiza repositório local a partir de um repositório remoto**





## Inicializando um Repositório Git

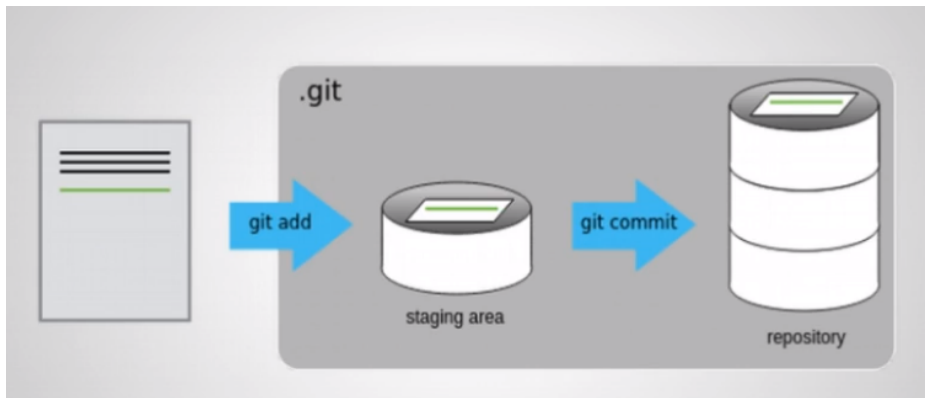
- Nosso primeiro *Commit*:



```
Terminal
File Edit View Search Terminal Help
(base) clayton@clayton-note:~/Aula_Git$ git commit -m "Meu primeiro commit"
[master (root-commit) 44c363f] Meu primeiro commit
2 files changed, 2 insertions(+)
create mode 100644 aula01.txt
create mode 100644 aulagit.txt
(base) clayton@clayton-note:~/Aula_Git$ git status
On branch master
nothing to commit, working tree clean
(base) clayton@clayton-note:~/Aula_Git$
```

## Inicializando um Repositório Git

- Como funciona?



## Visualizando Diferenças

- ❑ Utilizando o comando `git diff`:
  - Compara as mudanças específicas no diretório de trabalho com o índice de objetos (HEAD)
  - Pode ser especificado apenas um arquivo;
  - Mostrando as mudanças que ainda não foram *staged*;
  - Opção `--cached`, o *diff* irá comparar as mudanças *staged* com o repositório local;
- ❑ Combinado com o comando `git log` para mostrar a diferença entre dois *commits* específicos.

```
1 > git log --oneline
2 611dc29 (HEAD -> master, origin/master, origin/HEAD) Added Section 2
3 85cadb5 Added Section 1
4 42b045b Added README.md
5 > git diff 42b0 611c
```

## Desfazendo Mudanças

- Para desfazer mudanças, podemos utilizar o comando `git checkout` como uma forma de navegar na história de *commits*

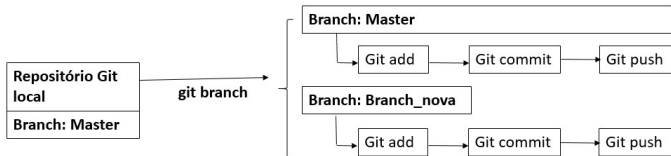
```
1 > git log --oneline
2 611dc29 (HEAD -> master, origin/master, origin/HEAD) Added Section 2
3 85cadb5 Added Section 1
4 42b045b Added README.md
5 > git checkout 42b045b README.md
6 > git commit "Revert to previous README.md"
```

- Usar o comando `git revert` quando mudanças já foram publicadas

## Branches

- ☐ `git branch` mostra a *branch* local atual.
- ☐ `git branch (nome)` cria uma nova *branch*.
- ☐ `git checkout (nome-branch)` troca para outra *branch*;
- ☐ `git merge` é usado para combinar alterações feitas em duas *branches* distintas;

## Criando uma ramificação do repositório



## Trocando contexto entre ramificações



## Combinando duas ramificações em uma existente



## Cuidando de conflitos

- ❑ Conflitos são situações em que o Git não consegue atualizar um arquivo com uma nova versão da *branch* automaticamente
- ❑ Nesses casos o Git avisa essa situação e pede para que o desenvolvedor resolva os conflitos

```
1  CONFLICT (content): Merge conflict in README.md
2  Automatic merge failed; fix conflicts and then commit the result.
3  > cat README.md
4  # README.md
5
6  <<<<<<< HEAD
7  ## Section 1
8  =====
9  ## Section 3
10 >>>>>>> conflict
```

## Criando e sincronizando Repositório

- ☐ Acesse o site do GitHub e sua conta

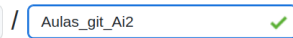
### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*



Repository name \*



Great repository names are short, simple, and unique. **Aulas\_git\_Ai2** is available. Need inspiration? How about **refactored-computing-machine**?

Description (optional)



**Public**

Anyone on the internet can see this repository. You choose who can commit.



## Vamos praticar

- ☐ Crie um arquivo README com seu nome no diretório **aula3***git* contendo suas informações como, *formação acadêmica, experiência profissional*
- ☐ Clone o diretório *DSUnesp*
- ☐ Haverá um novo diretório também onde, iremos colocar nossos trabalhos.
- ☐ **Obs:** todos os slides e códigos de aulas estarão no diretório do git para nossas aulas futuras.

**obrigado.**