

**GIT**

**Semana Acadêmica da Engenharia de Computação -  
SAECOMP**

**Prof. Julio Saraçol** [juliodomingues@unipampa.edu.br](mailto:juliodomingues@unipampa.edu.br)



# História



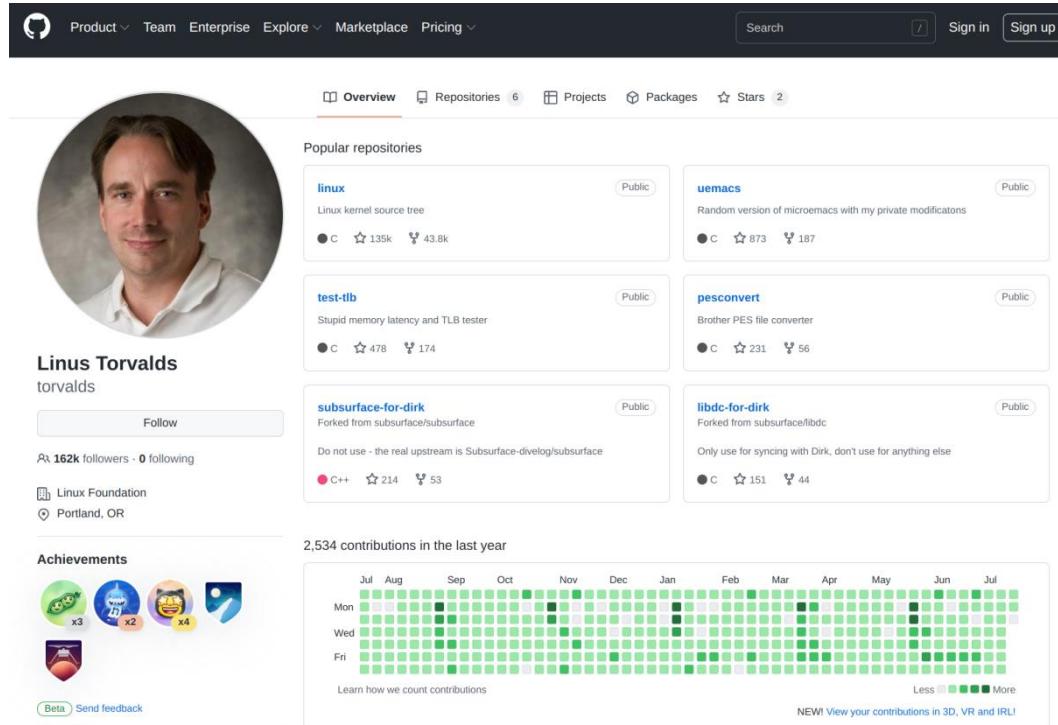
- Motivação:
  - Linus Torvalds estava insatisfeito com o BitKeeper, ferramenta de controle de versão que ele utilizava para desenvolver o kernel do Linux.
  - Após estudar as outras opções ...  
***“The end result was I decided I can write something better than anything out there in two weeks, and I was right.”***

[Tech Talk: Linus Torvalds on git](#)



# Motivação

- Atualmente o desenvolvimento do kernel Linux é auxiliado exclusivamente pelo Git.



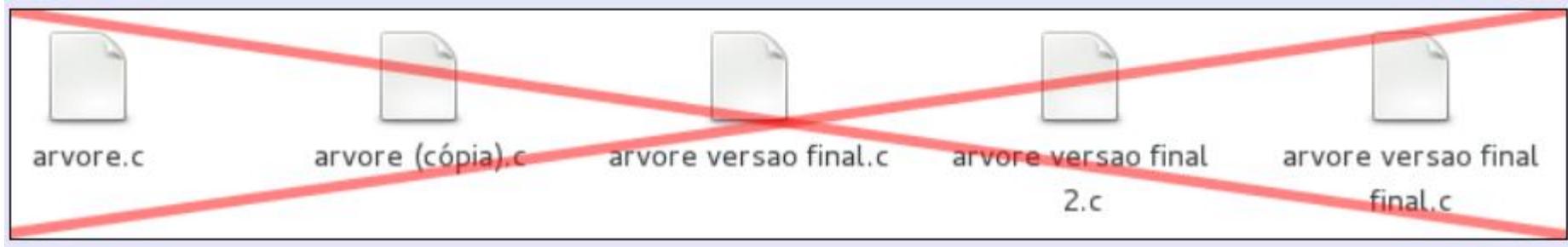
- [github.com/torvalds](https://github.com/torvalds)



# Mas por que eu usaria um controle de versão?

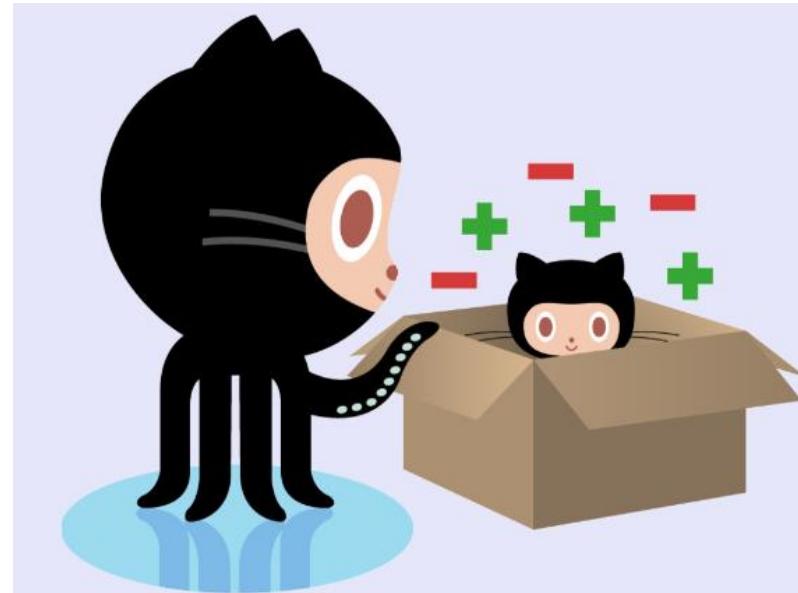
# Organizar o desenvolvimento de software

- **Gerenciar manualmente** as versões do seu software **não será mais necessário**, o Git gerencia de uma forma mais organizada e eficiente para você.



# Organizar o desenvolvimento de software

- O Git oferece **controle total do projeto** ao desenvolvedor para, entre outras coisas:
  - Visualizar as **mudanças ocorridas** em cada arquivo;
  - Visualizar o **estado do projeto** em etapas anteriores;
  - **Desfazer** mudanças;
  - Desenvolver funcionalidades em **paralelo**.



# Compartilhar projetos

- Desenvolver projetos colaborativos nem sempre é fácil.
- Utilizar DropBox, pen drives ou afins para compartilhar código muitas vezes resulta em dor de cabeça.

- V1\_final.c
- V1\_finalv2.c
- V1\_final\_agoraVai.c
- V1\_final\_agoraVai\_alternativa2.c





## Problema em ferramentas de armazenamento de arquivos??

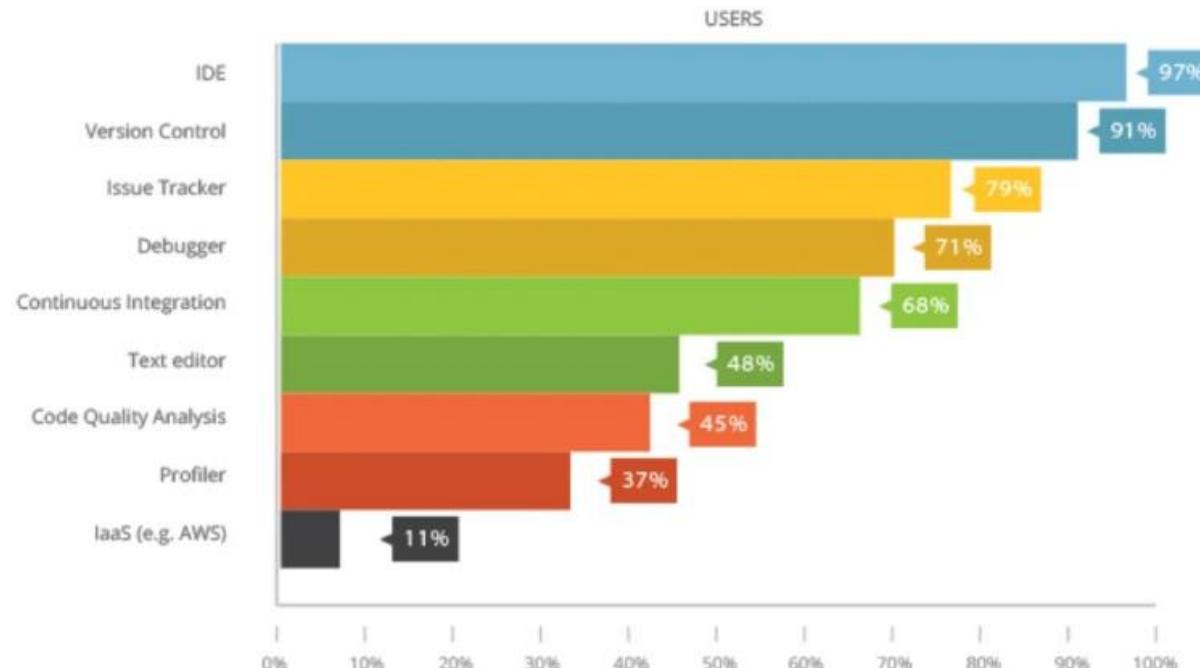
- Estas ferramentas **foram projetadas para fins genéricos**, não oferecendo aspectos importantes para uma equipe de desenvolvimento, como **histórico de ações de cada colaborador**, consistência entre as versões dos integrantes e **fácil identificação e correção de bugs**.
- Mais uma vez o **Git cuida disso para você**.

# Independente de Plataforma



- O mercado utiliza muito !!!!
- Desenvolvimento profissional é feito assim :)

Tool type usage

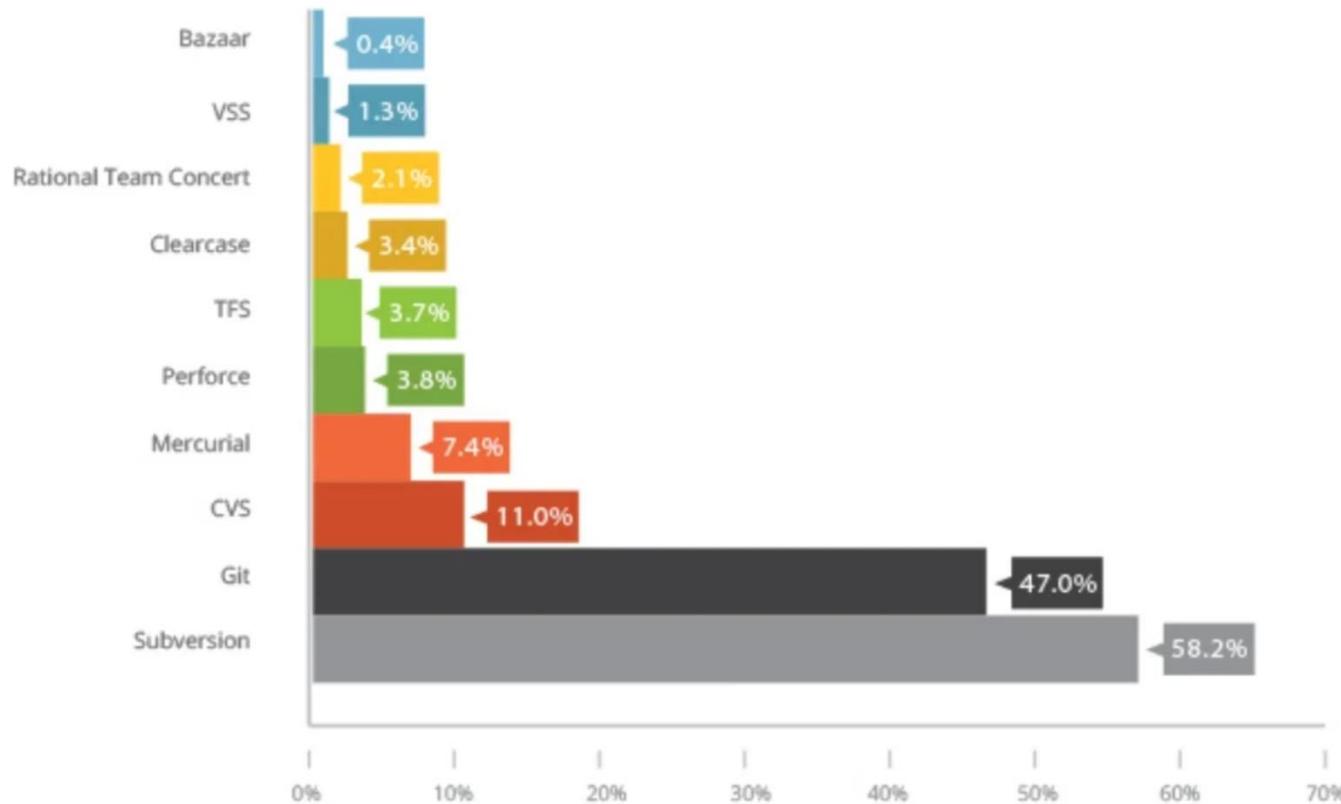


rebellabs

# O mercado utiliza muito !!!!

- E o Git está entre os mais populares.

**Popularity of Version Control Systems (VCS) used by respondents**



# Quem usa Git?

## Companies & Projects Using Git

Google

FACEBOOK

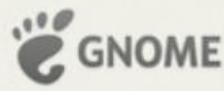
Microsoft



LinkedIn NETFLIX



PostgreSQL



## Vagas de emprego!

- Experience with subversion, GIT - branching, and merging

Apple

- Experience with code/asset repositories such as SVN, Git, Perforce, TFS, and Alienbrain

Blizzard

. You will be responsible for everything from maintaining Google Docs and Sites aimed at  
Android partners to developing API references and managing source HTML in a Git repository.

Android/Google

# O que é?



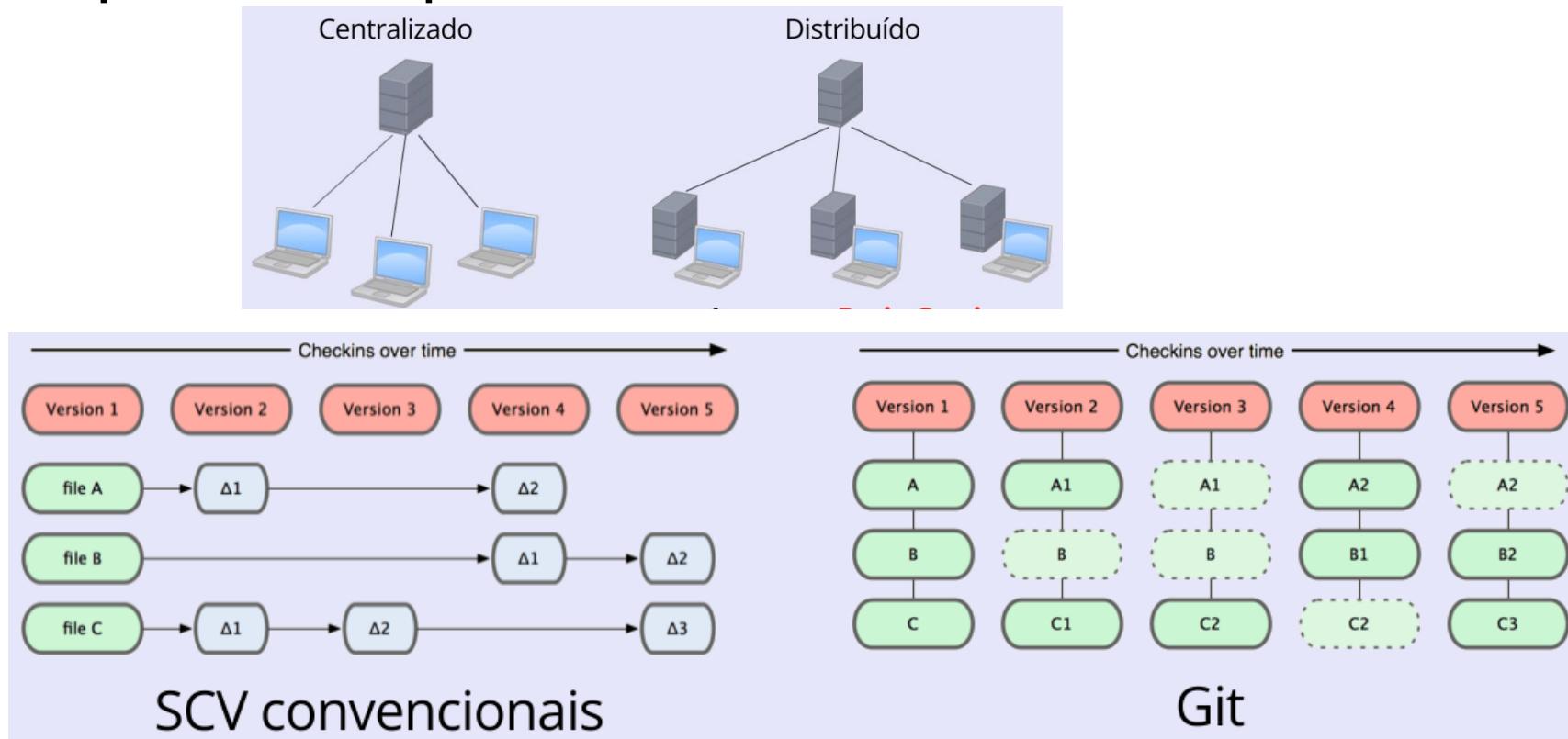
- Git:
  - É um sistema de **controle de versão de arquivos**;
  - Através dele podemos desenvolver projetos nos quais diversas pessoas podem contribuir simultaneamente, ou simplesmente controlarmos versões do nosso software (ou só arquivos);

# GIT:

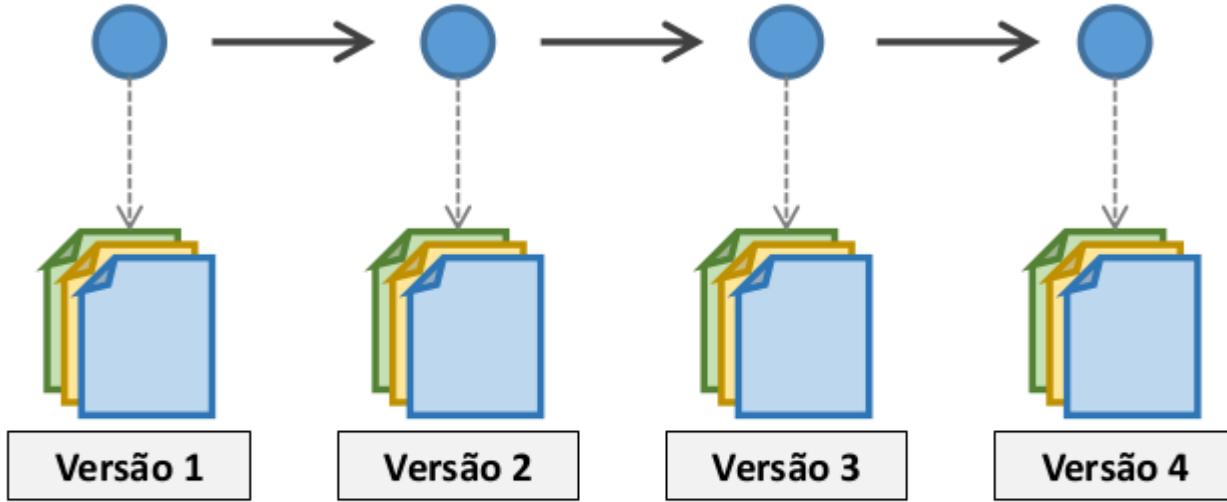
- SISTEMAS DE CONTROLE DE VERSÃO:
  - Controle de histórico;
  - Trabalho em equipe;
  - Marcação e resgate de versões estáveis;
  - Ramificação do projeto;

# Como o Git funciona?

- Baseado em Snapshots e não em lista de alterações.
- No Git, todos os clientes possuem uma cópia completa do repositório remoto.



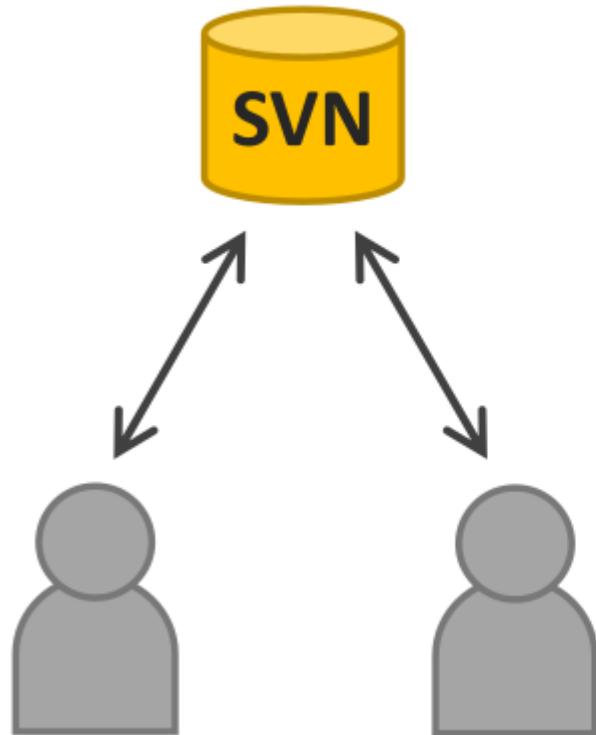
# Snapshots



Cada versão é uma “foto” do diretório de trabalho e será representado por um círculo azul denominado *commit*.

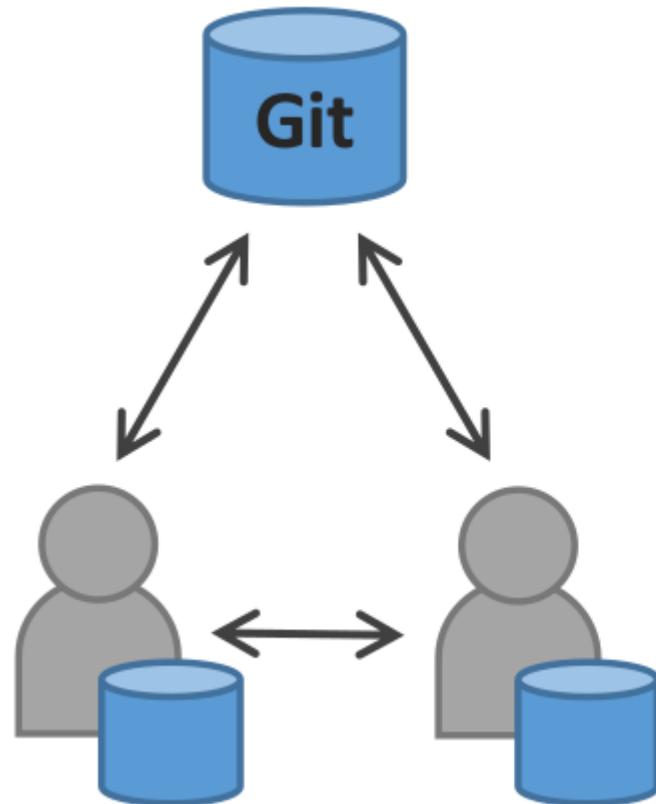
*bismarckiunior@outlook.com*

# Sistema de Controle de Versão Centralizado



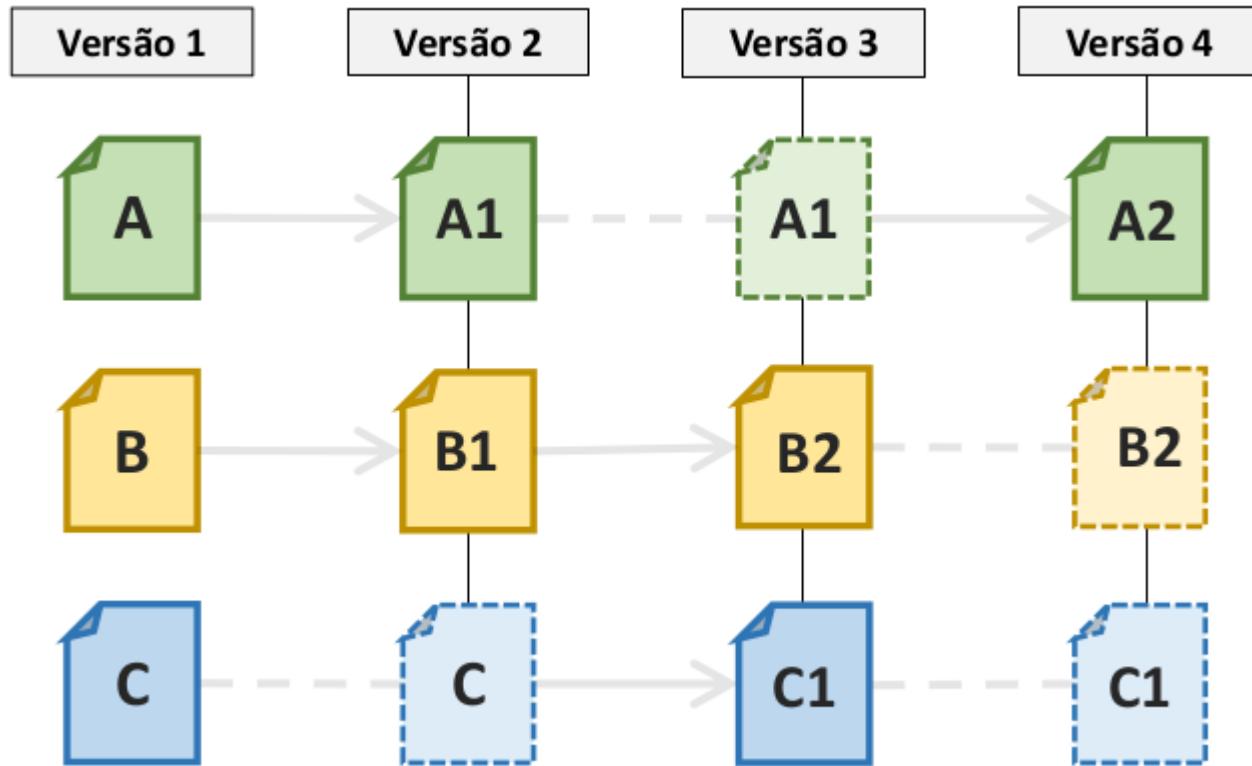
- Pouca autonomia  
Ações necessitam de acesso ao servidor.
- Trabalho privado limitado  
Versiona apenas arquivos no repositório.
- Risco de perda de dados  
Tudo em um único repositório.

# Sistema de Controle de Versão Distribuído

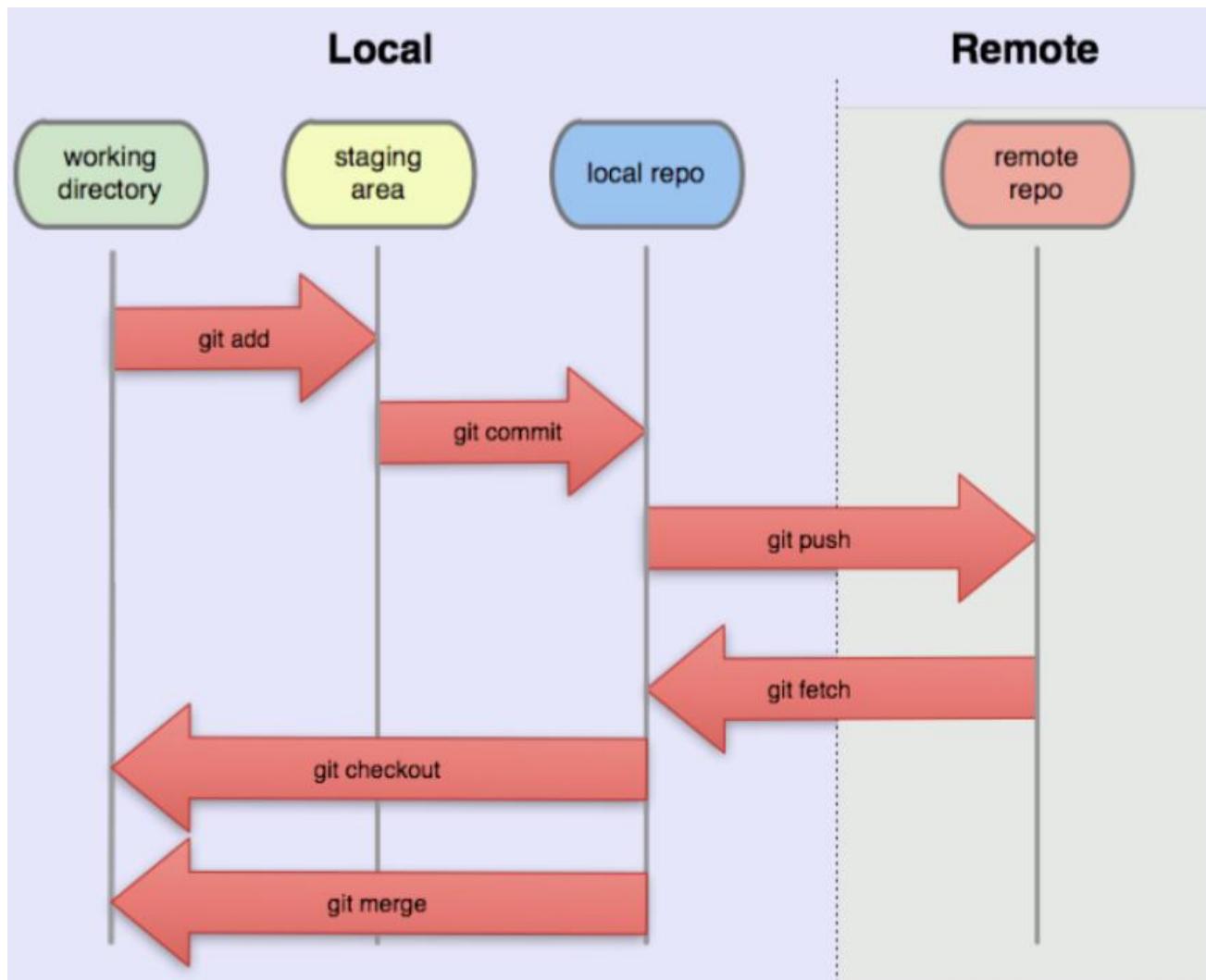


- Autonomia  
Ações básicas “off-line”.
- Rapidez  
Processos são locais.
- Trabalho privado  
Trabalho local não afeta os demais.
- Confiabilidade  
Todo repositório é um *backup*, ou seja, uma cópia completa do repositório, incluindo versões anteriores e histórico.

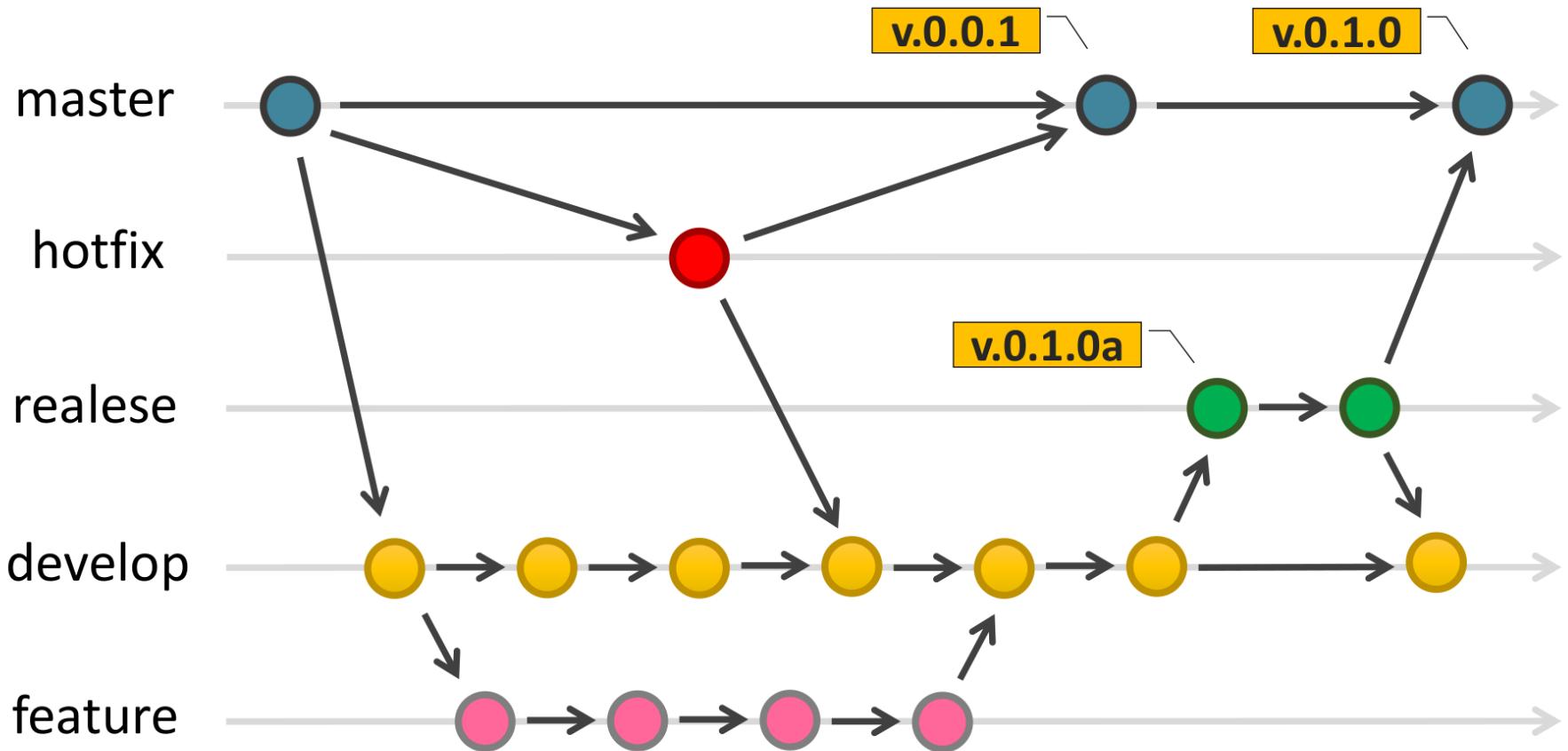
# Sistema de Controle do Git



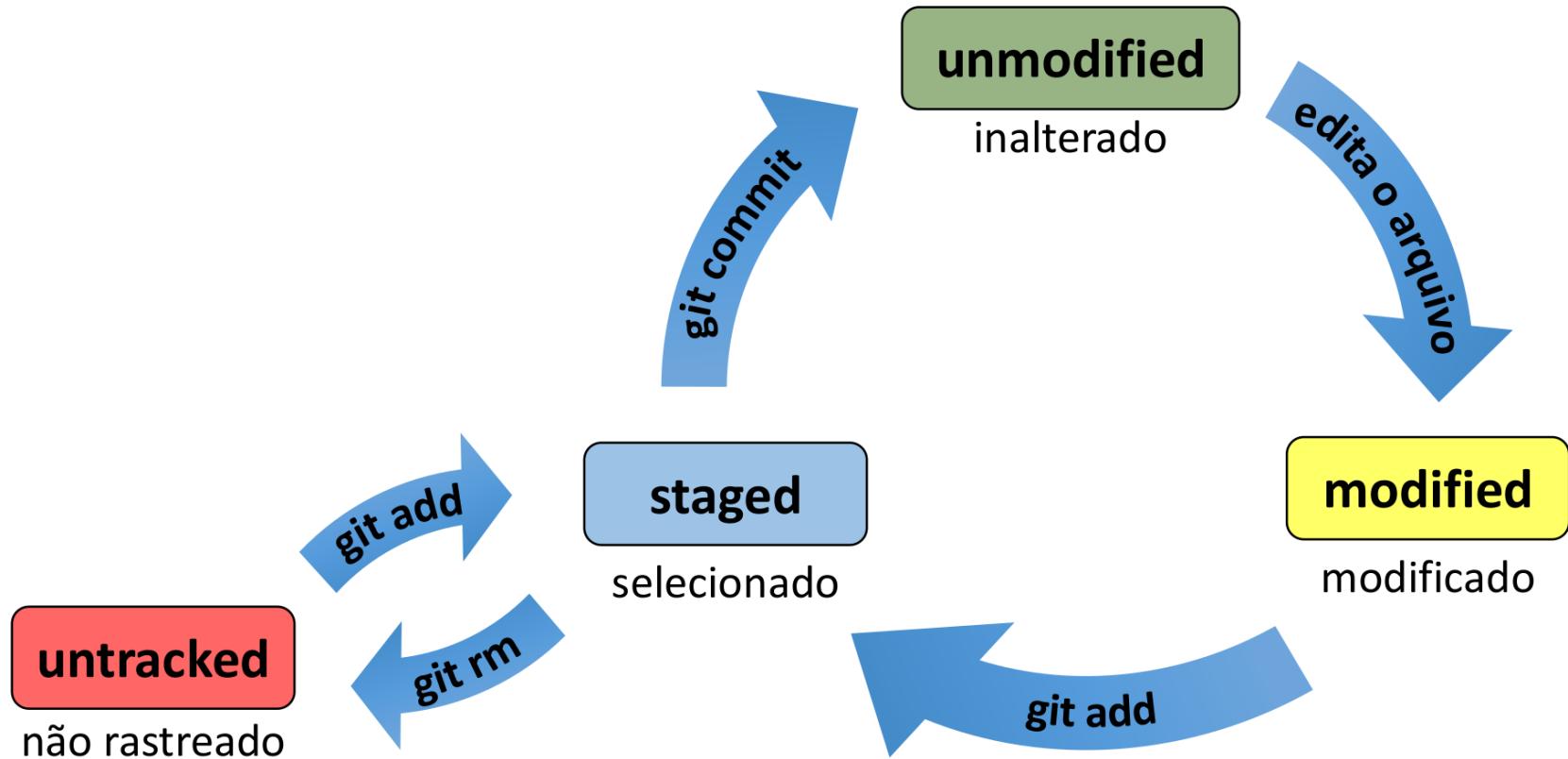
# Fluxo de Trabalho - Dev



# Fluxo de Trabalho - Projeto



# Tipos de Estado de um Arquivo



# Configurando Ambiente

# Instalando

- Ambiente Linux
  - sudo apt-get install git-all
  - sudo yum install git-all
- Ambiente Windows:
  - <https://git-scm.com/download/win>
- Ambiente MAC:
  - <http://git-scm.com/download/mac>.
- Mais informações: <https://git-scm.com/download>

# Instalando II [opcional]

Não é obrigatório mas a forma com mais produtividade seria utilizando **ssh** (chave pública e privada) no **caso de repositórios remotos** (falaremos disso mais tarde).

- Linux pacote ssh nativo
  - ssh-keygen
- Windows
  - <https://tecnoblog.net/responde/windows-10-ativar.openssh-cliente/>

# Sistema de Controle de Versão com SSH

O próprio Bitbucket/Github/Gitlab orienta como criar o projeto na sua máquina e sincronizá-lo com o servidor de versionamento remoto.

Lembrando:  
Para a utilização é preciso cadastrar chave SSH para realizar autenticação por SSH.

Leia:

<https://www.vivaolinux.com.br/artigo/Conexoes-SSH-sem-senha-facil-e-descomplicado>

<https://goo.gl/YhD6jE>



Engenharia de Computação  
SAECOMP

## Add some code

You can start a brand new project or push up an existing repo to Bitbucket. What are you doing?

I'm starting from scratch

I have an existing project to push up

## Clone your new repo

Set up Git on your machine if you haven't already.

```
$ mkdir /path/to/your/project
$ cd /path/to/your/project
$ git init
$ git remote add origin git@bitbucket.org:...
```

Visit [Bitbucket 101](#) for more help getting set up.

Next

# Sistema de Controle de Versão – Terminal Linux

- \$ sudo apt-get install git-all
- ssh-keygen -t rsa -C "comment"
- sudo apt-get install xclip
- cat ~/.ssh/id\_rsa.pub | xclip -sel clip
- ssh git@github.com
- git config --global user.name "Your Name"
- git config --global user.email codexico@gmail.com
- ---projeto-----
- \$ mkdir nomedoprojeto
- \$ cd nomedodiretorio
- \$ git init
- \$ git remote add origin git@github.com:usuario/tutorial-github.git

# Comandos Básicos do Git

## Trabalhando Localmente

# Consultas Rápidas

- Caso não se lembre de algum comando no decorrer do minicurso ou queira uma explicação mais detalhada ...

**git help  
man git**

# Configurando o ambiente

- Identificação

**git config --global user.name “nome”**

**git config --global user.email “email”**

- Editor

**git config --global core.editor “editor”**

- Verificar configurações atuais

**git config –list**

Todas informações de configuração ficam na pasta oculta **.git** em diferentes arquivos (por exemplo

# Sistema de Controle de Versão



Início

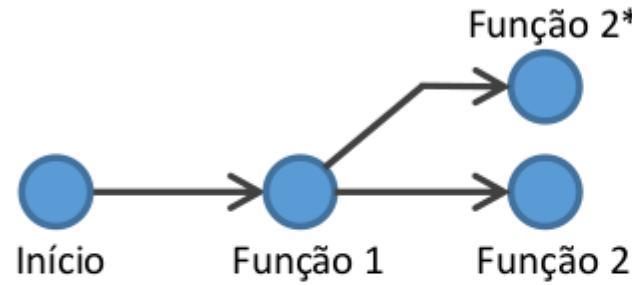


**commit**

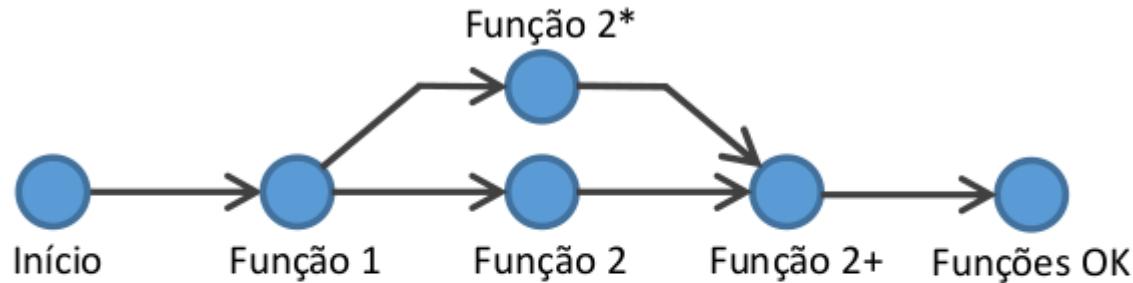
# Sistema de Controle de Versão



# Sistema de Controle de Versão

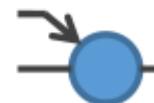


# Sistema de Controle de Versão

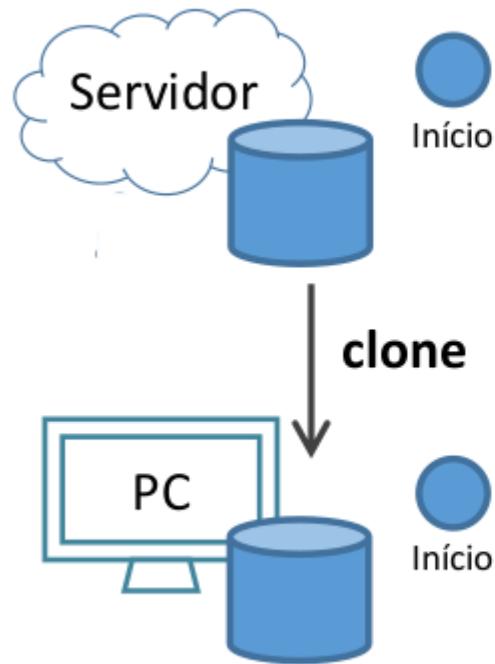


 commit

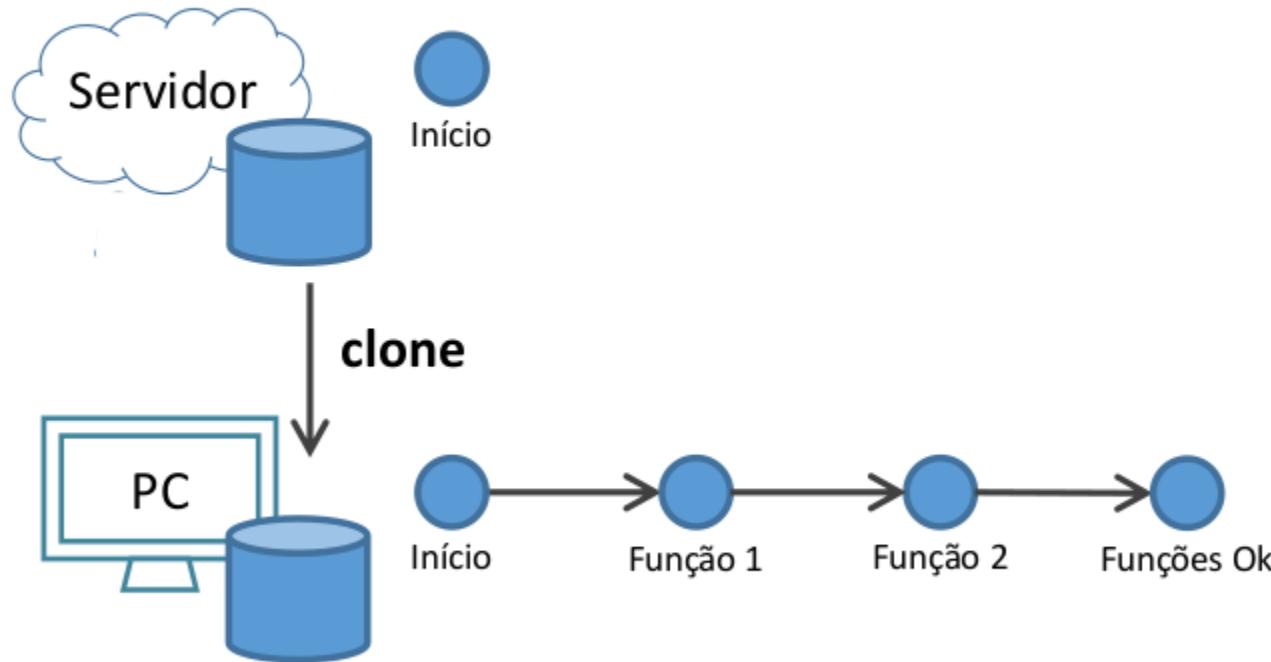
 branch

 merge

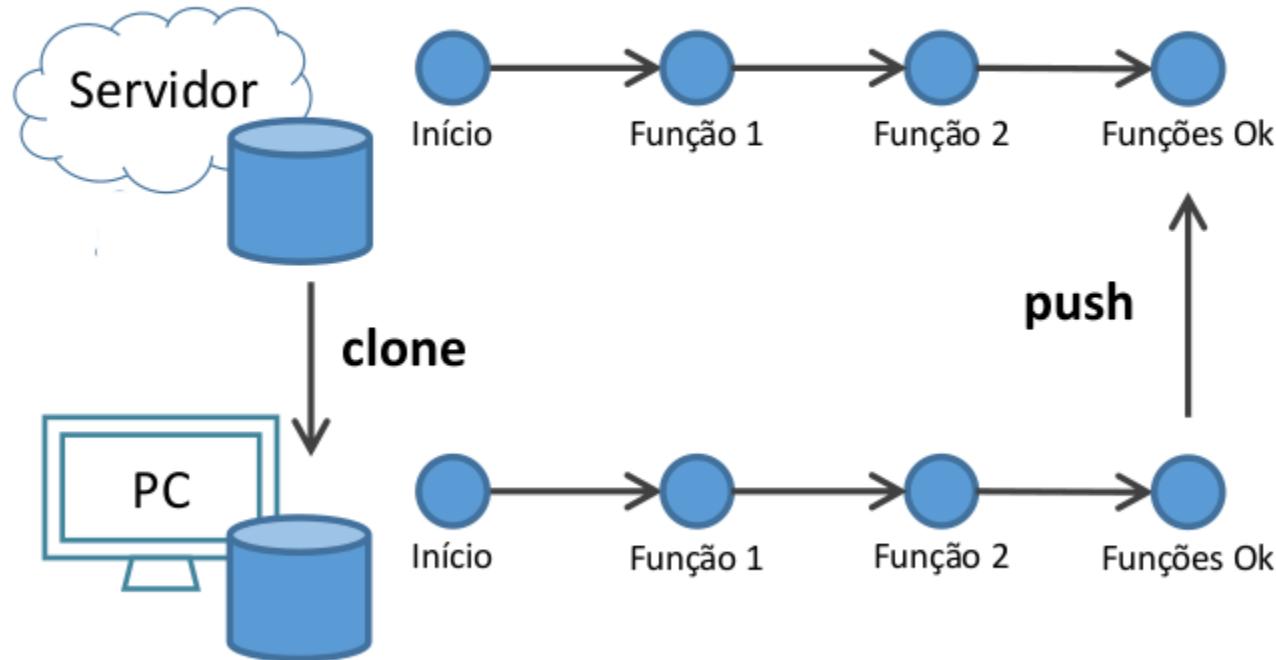
# Sistema de Controle de Versão



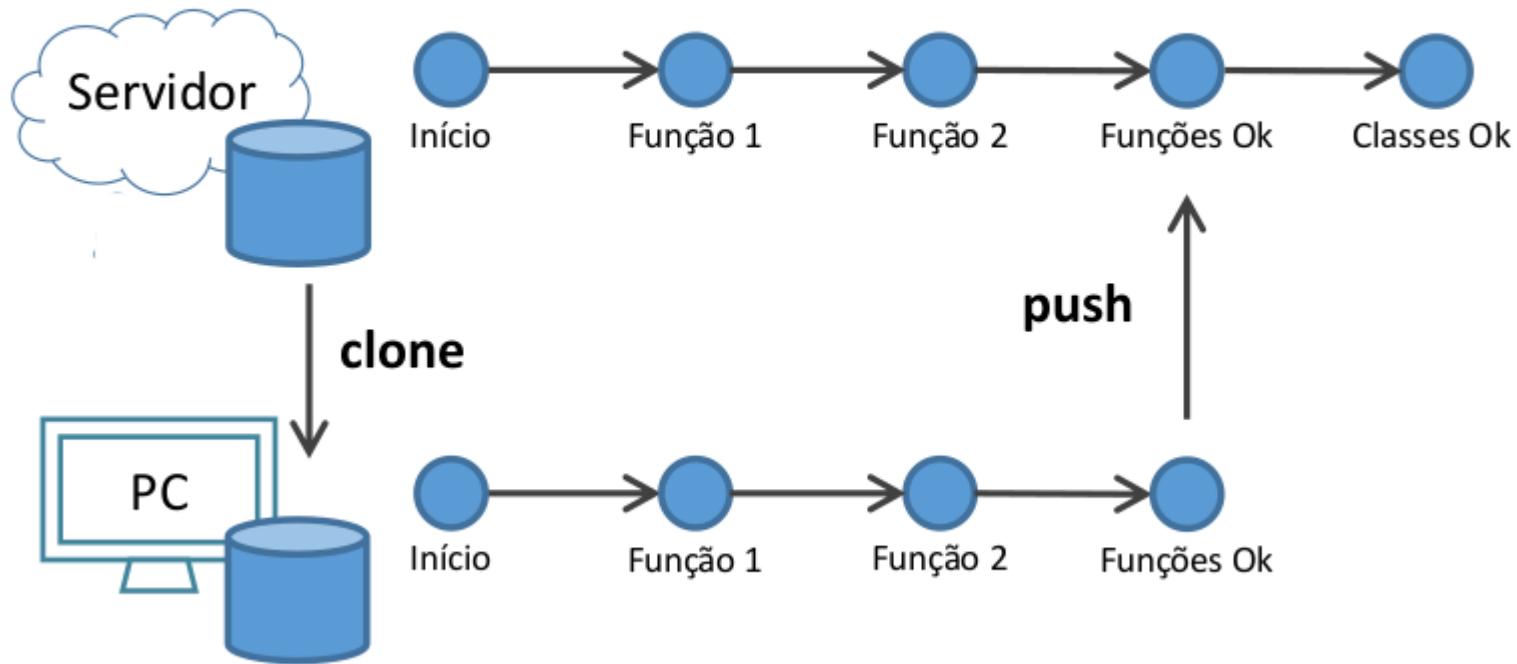
# Sistema de Controle de Versão



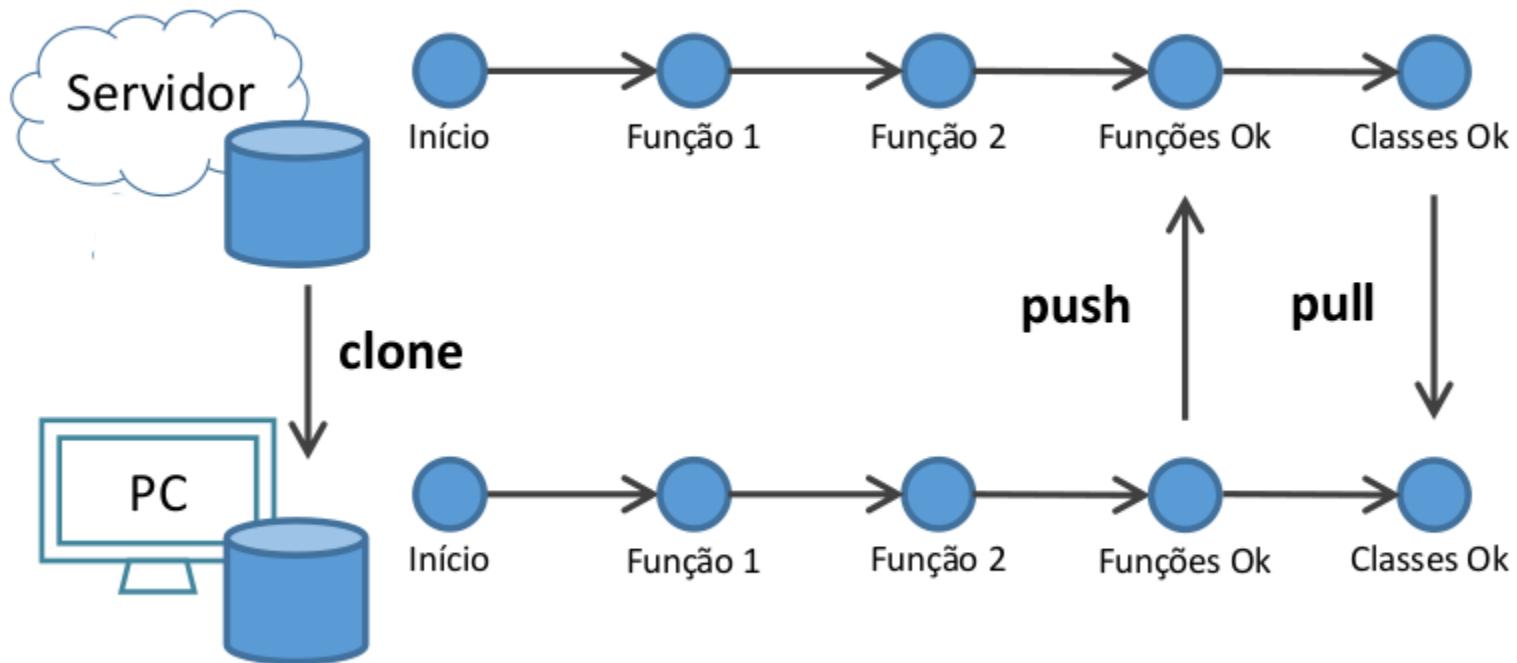
# Sistema de Controle de Versão



# Sistema de Controle de Versão



# Sistema de Controle de Versão



# Sistema de Controle de Versão

## Criando um Repositório



```
$ git init
```

Transforma a diretório atual em um repositório git, criando o subdiretório “.git”.

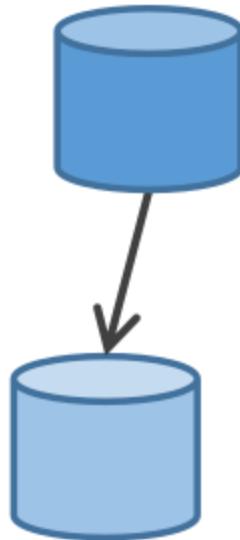


```
$ git init <dir>
```

Cria o diretório <dir> e transforma em um repositório git.

# Sistema de Controle de Versão

## Clonando um Repositório



```
$ git clone <repo>
```

Clona o repositório <repo> para a máquina local.

```
$ git clone <repo> <dir>
```

Clona o repositório <repo> para o diretório <dir>.

```
$ git clone git@github.com:user/Project.git
```

```
$ git clone https://github.com/user/Project.git
```

# Sistema de Controle de Versão

## Configuração Inicial do Git

```
$ git config --global user.name <nome>
```

Atribui <nome> ao nome do usuário.

```
$ git config --global user.email <email>
```

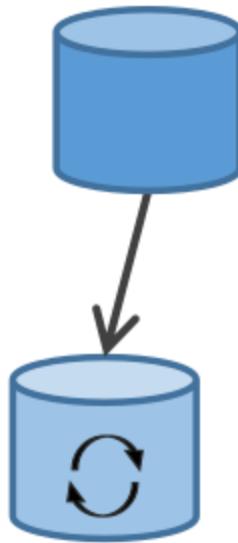
Atribui <email> ao e-mail do usuário.

```
$ git config --global core.editor <editor>
```

Atribui <editor> como editor padrão. Ex.: notepad, emacs ...

# Sistema de Controle de Versão

## Atualizando o Repositório Local



```
$ git fetch [<repo>]
```

Baixa todos os dados do repositório <repo>.

```
$ git fetch [<repo>] [<branch>]
```

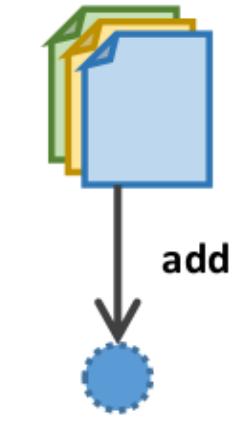
Baixa todos os dados do branch <branch> do repositório <repo>.

```
$ git pull [<repo>]
```

Atualiza todos os dados do repositório <repo>, ou seja, realiza um *fetch* seguido de um *merge*.

# Sistema de Controle de Versão

## Preparando Para Salvar Alterações



Stage Area  
(Index)

```
$ git add <arquivo | dir>
```

Adiciona as mudanças do arquivo *<arquivo>* ou do diretório *<dir>* para o próximo *commit*. O arquivo passa a ser rastreado.

```
$ git reset <arquivo>
```

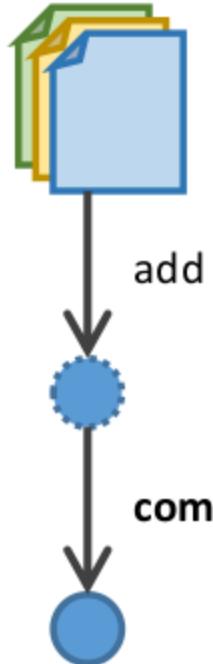
Remove as mudanças do arquivo *<arquivo>* para o próximo *commit*.

```
$ git rm --cached <arquivo>
```

Para de rastrear o arquivo *<arquivo>*.

# Sistema de Controle de Versão

## Salvando Alterações



```
$ git commit
```

Realiza o *commit* e abre o editor para inserir uma mensagem.

```
$ git commit -a
```

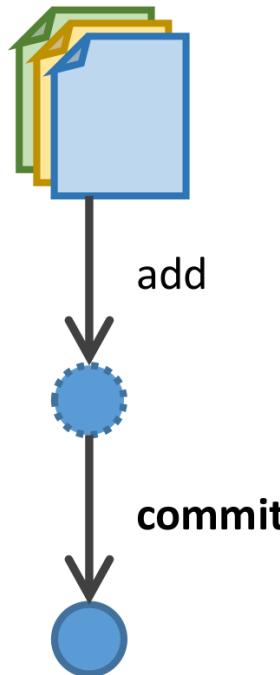
Adiciona as mudanças dos arquivos já rastreados e realiza o *commit*. O editor será aberto.

```
$ git commit -m "<msg>"
```

Realiza o *commit*, com a mensagem <msg>.

# Sistema de Controle de Versão

## Salvando Alterações



```
$ git commit -am <msg>
```

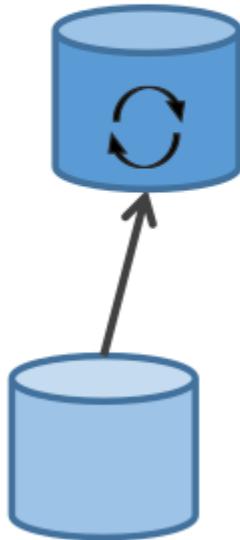
Adiciona as mudanças dos arquivos já rastreados e realiza o *commit* com a mensagem <msg>.

```
$ git commit --amend -m <msg>
```

Substitui o último commit e altera a mensagem para <msg>.

# Sistema de Controle de Versão

## Enviando Para o Repositório



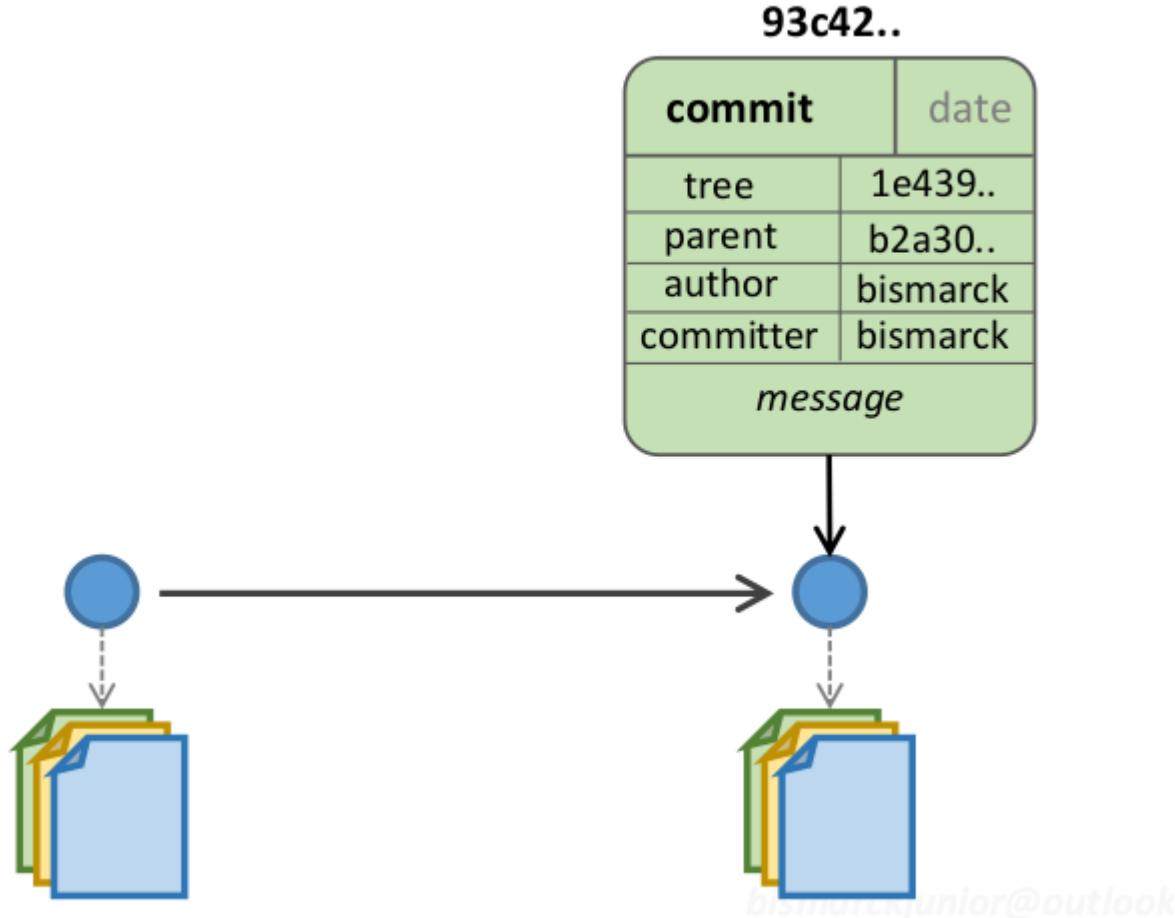
```
$ git push [<repo>] [<branch>]
```

Envia o branch *<branch>* para o repositório *<repo>*.  
Por padrão *<repo>* é *origin* e *<branch>* é o *branch* atual, mas pode ser configurado\*.

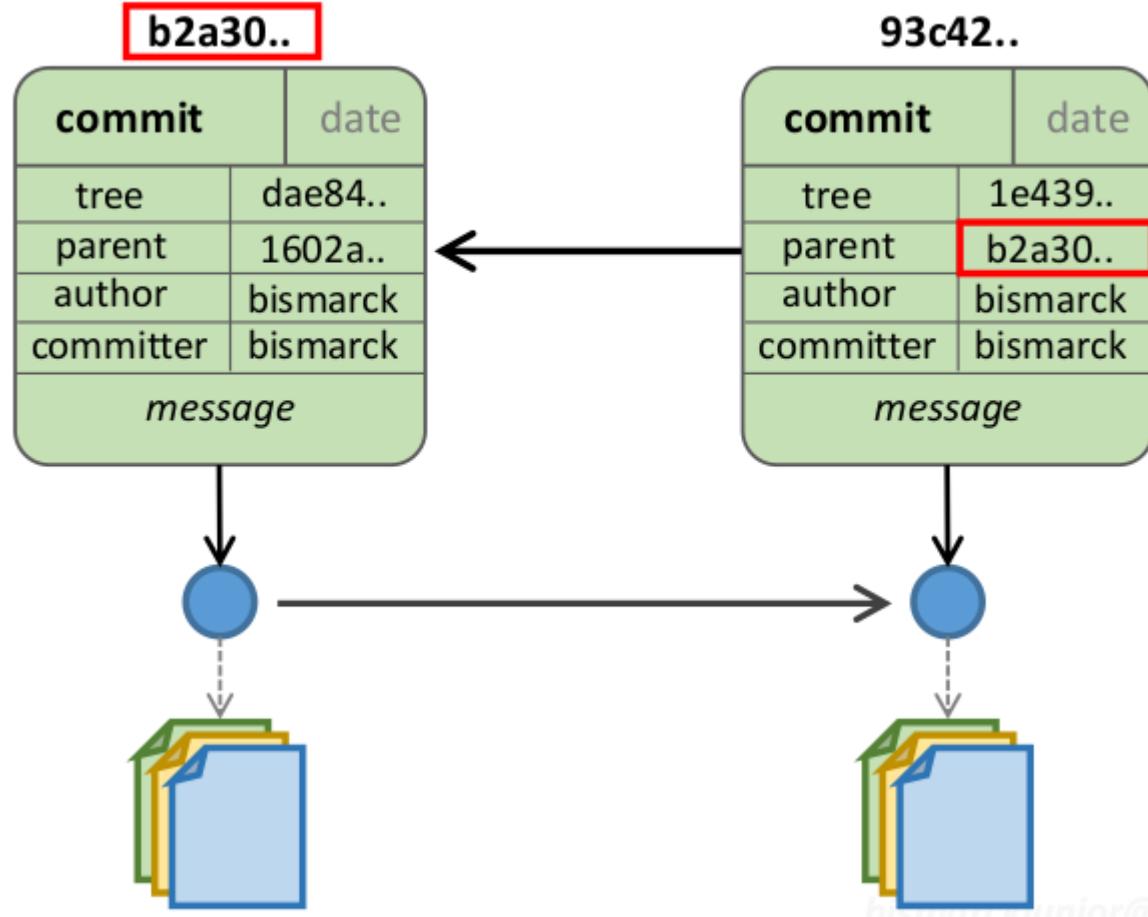
```
$ git push [<repo>] --all
```

Envia todos os *branches* para o repositório *<repo>*.

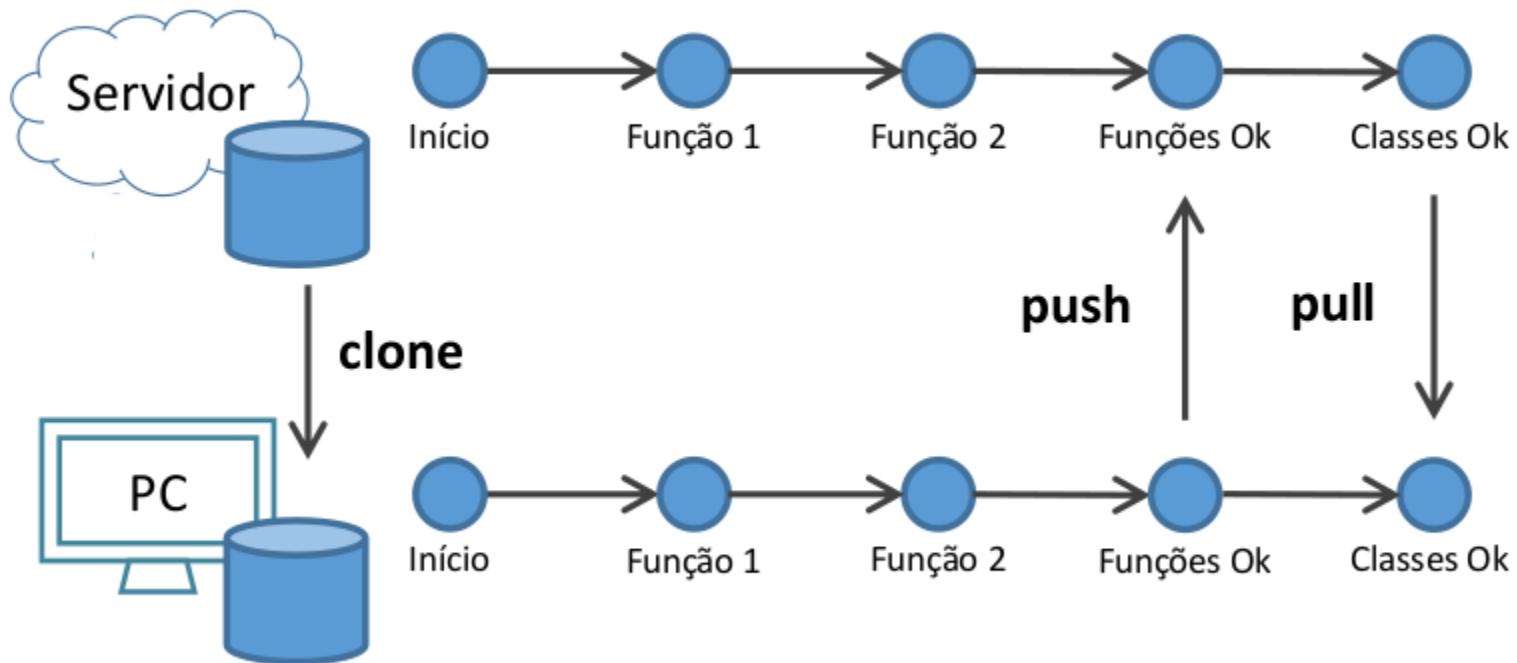
# Sistema de Controle de Versão



# Sistema de Controle de Versão



# Sistema de Controle de Versão



# Analisando os Arquivos na Área Transitória



```
$ git status
```

Lista os arquivos que estão e que não estão na área transitória, e os arquivos que não estão sendo rastreados.

```
$ git status -s
```

Lista os arquivos de uma forma simplificada.

# Tagging



```
$ git tag
```

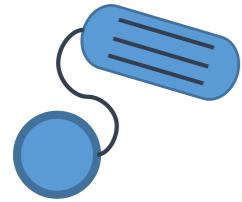
Lista as *tags* existentes.

```
$ git tag -l <tag>
```

Procura pela *tag* <tag>.

```
$ git tag -l 'v.0.*'
```

# Tagging



```
$ git tag <tag> [<commit>]
```

Cria a *tag* *<tag>* para o último *commit* ou para o *commit* *<commit>*.



```
$ git tag -a <tag>
```

Cria a *tag* *<tag>* completa para o último *commit* e abre o editor para inserir uma mensagem.

```
$ git tag -a <tag> -m <msg>
```

Cria a *tag* *<tag>* completa para o último *commit* com a mensagem *<msg>*.

bismarckiunior@outlook.com

# Versionamento

v.0.1.0

**v [major] . [minor] . [patch]**

**[patch]**: correção de *bugs*.

**[minor]**: incrementos de funcionalidades compatíveis com versões anteriores.

**[major]**: incrementos de funcionalidades incompatíveis com versões anteriores.

**Versões teste**: alpha (a), beta (b)

**Ex:** v0.1.9 < v0.1.10 < v0.2.0a < v0.2.0b < v0.2.0

# Referência a Commit

## <sha1>

*Hash SHA-1 referente ao commit.* Pode-se usar os primeiros caracteres.

Ex: b230 = b230e84a4c90d2f11ba85404e5fba93ce0a...

## <tag>

*Tag referente ao commit.*

Ex: v0.1.2

## <branch>

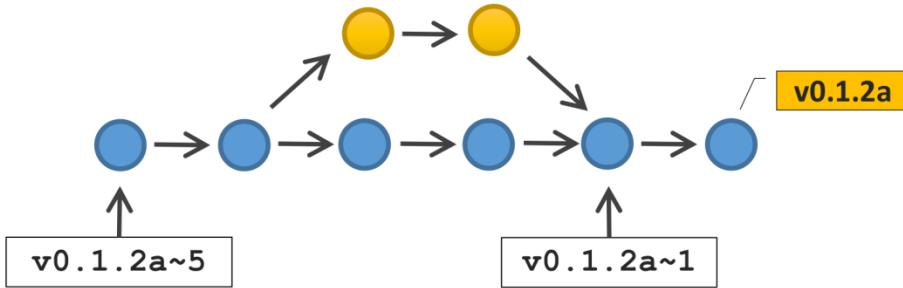
*Último commit do branch <branch>.*

Ex: master

# Referência a Commit

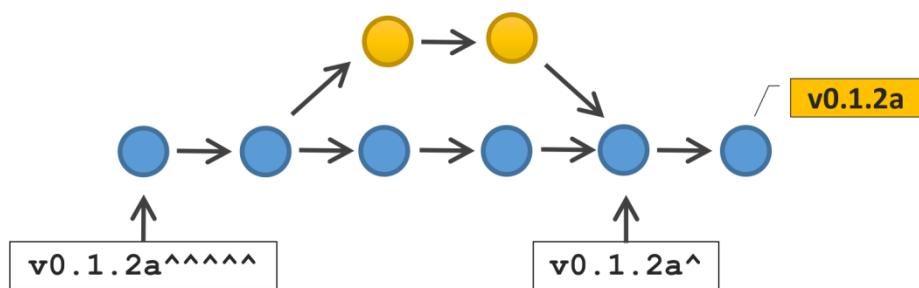
**<commit><sup>~n</sup>**

O n-ésimo percussor do commit <commit>.



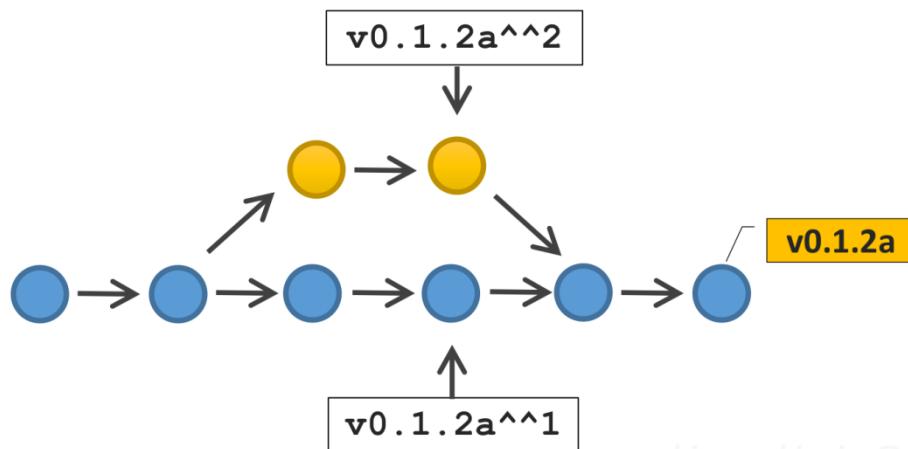
**<commit><sup>^1</sup> ou <commit><sup>^</sup> ou <commit><sup>~1</sup>**

O primeiro percussor do commit <commit>.



**<commit><sup>^2</sup>**

O segundo percussor do commit <commit>. Utilizado em commits resultantes de um *merge*.



# Analizando Commits



```
$ git show
```

Exibe o último *commit*.

```
$ git show <commit>
```

Exibe o *commit* referenciado por <commit>.

```
$ git show <commit>:<arquivo>
```

Exibe o arquivo <arquivo> no *commit* <commit>.

# Analizando um Arquivo



```
$ git blame <arquivo>
```

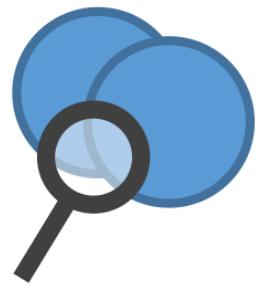
Exibe quem modificou cada linha do arquivo *<arquivo>*, incluindo data e *commit*.

Capturar uma nova imagem

```
$ git blame -L <n>,<m> <arquivo>
```

Exibe quem modificou as linhas de *<n>* a *<m>* do arquivo *<arquivo>*, incluindo data e *commit*.

# Diferença Entre Commits



```
$ git diff <commit>
```

Exibe a diferença nos arquivos entre o *commit* <commit> e o diretório de trabalho.

```
$ git diff --cached <commit>
```

Exibe a diferença nos arquivos entre o *commit* <commit> e a área transitória.

# .gitignore

- Arquivo que contém os arquivos que não serão visíveis pelo git.
- Arquivo .gitignore (exemplo)

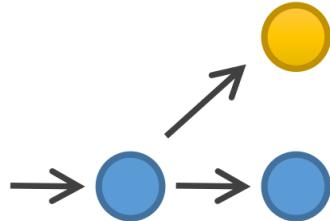
```
Thumbs.db      #Arquivo específico  
*.html        #Arquivos que terminam com “.html”  
!index.html    #Exceção, esse arquivo será visível ao git  
log/           #Diretório específico  
*/tmp          #Qualquer diretório nomeado de “tmp”
```

- Arquivos que já estavam sendo rastreados não são afetados.

# Comandos avançados

# **BRANCHES**

# Criando Ramificações



```
$ git branch [-a]
```

Exibe os *branches* existentes. Na forma completa, exibe também os *branches* remotos.

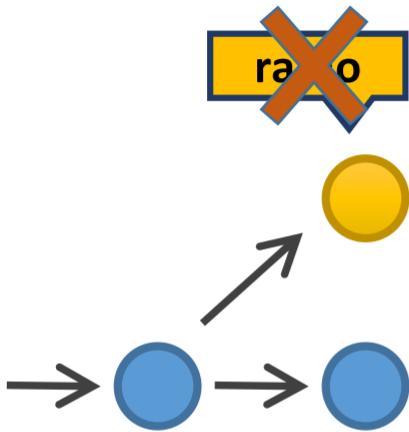
```
$ git branch <branch> [<base>]
```

Cria o *branch* *<branch>* a partir do *commit* *<base>*.

```
$ git checkout -b <branch>
```

Cria o *branch* *<branch>* e altera para ele.

# Excluindo Ramificações



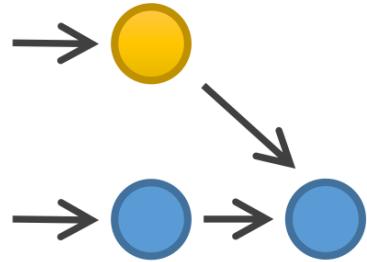
```
$ git branch -d <branch>
```

Exclui o *branch* <branch>. O *branch* já deve ter sido mesclado.

```
$ git branch -D <branch>
```

Exclui o *branch* <branch> mesmo não tendo sido mesclado.

# Mesclando Commits



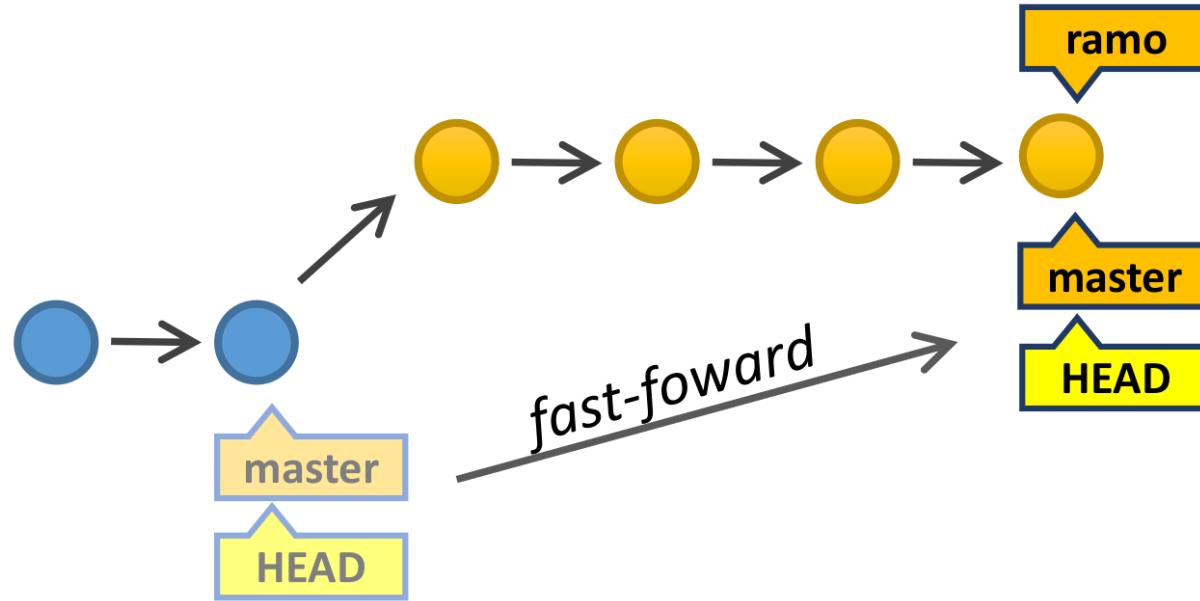
```
$ git merge <branch>
```

Mescla os *commits* do *branch* `<branch>` para o *branch* atual.

```
$ git merge <branch> --no-ff
```

Mescla os *commits* do *branch* `<branch>` para o *branch* atual sem *fast-forward*.

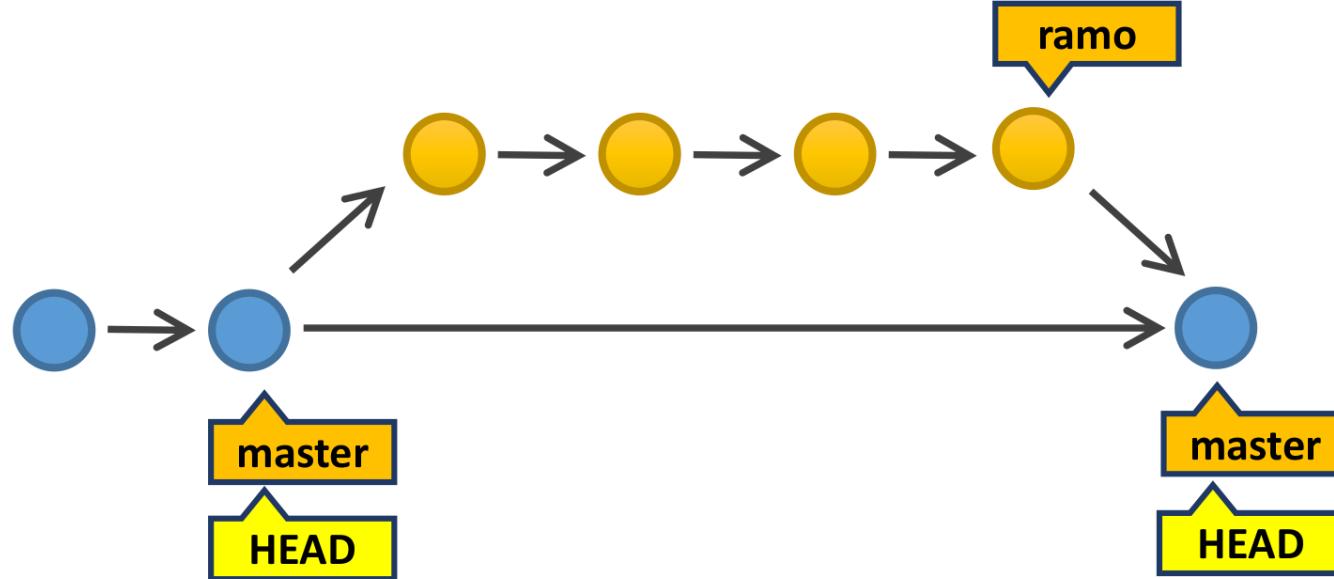
# Mesclando Commits com Fast-forward



```
$ git merge ramo
```

Neste caso, não é necessário nenhum *commit* para realizar a mesclagem. Ocorre apenas um avanço rápido (*ff*).

# Mesclando Commits sem Fast-forward

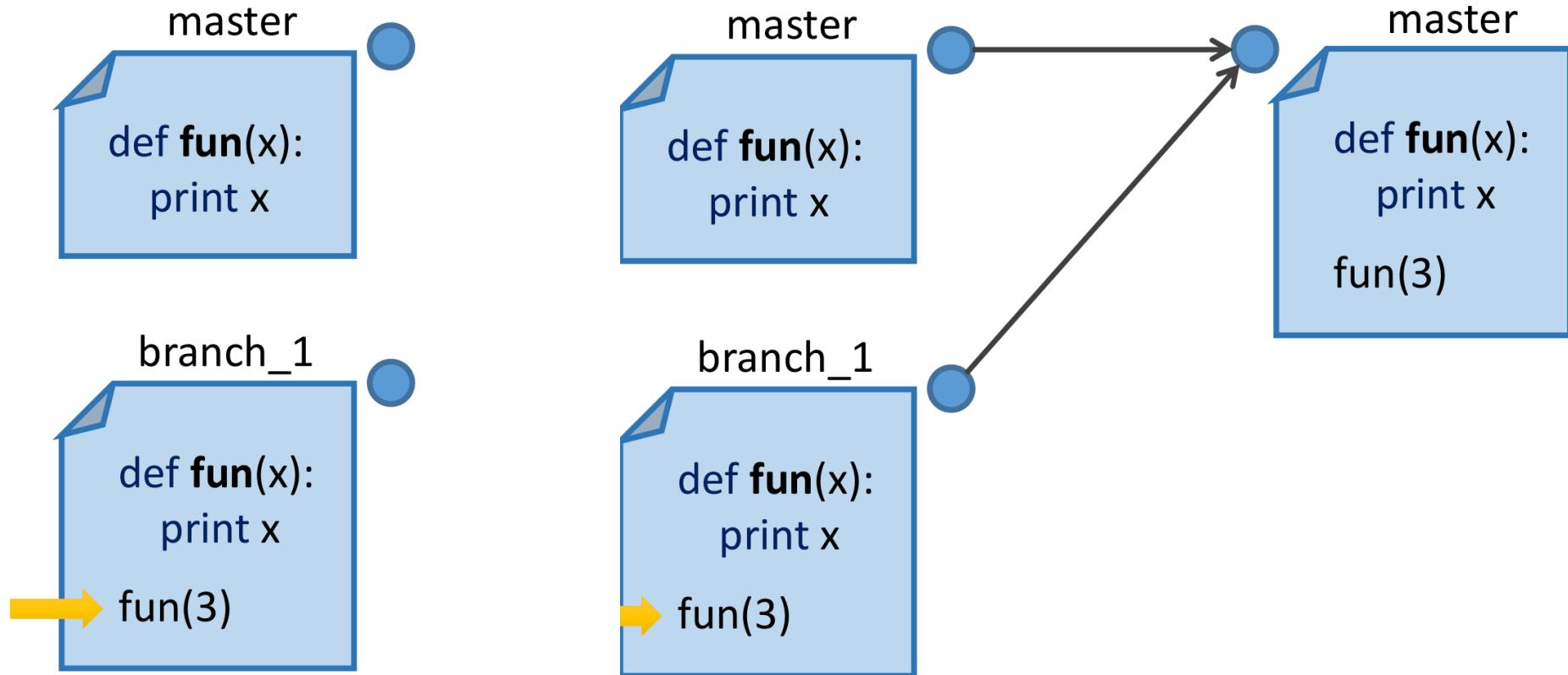


```
$ git merge ramo --no-ff
```

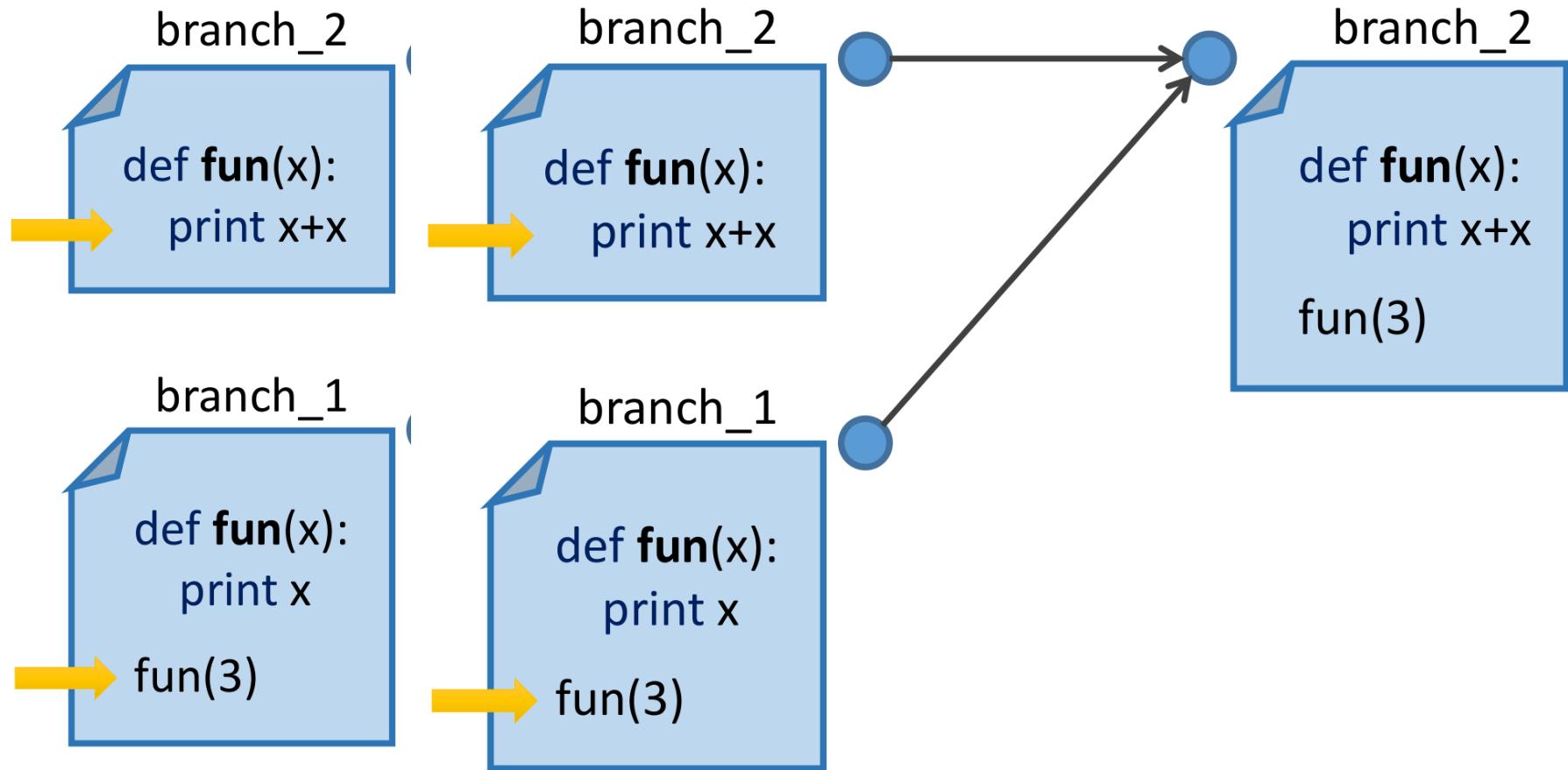
Realiza um *merge* com um *commit* obrigatoriamente.  
Possibilita uma melhor visualização no histórico.

*distinckj@outlook.com*

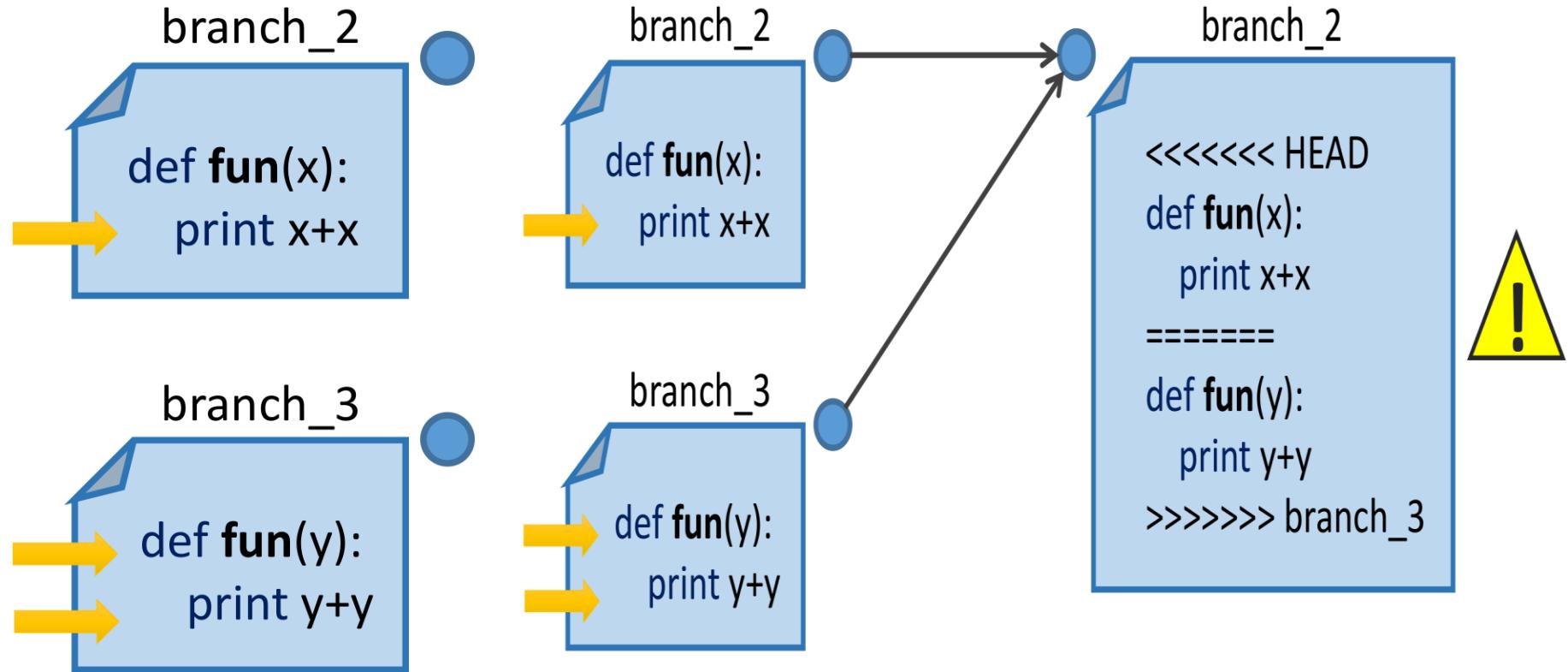
# Mesclando Commits - Conflitos



# Mesclando Commits - Conflitos

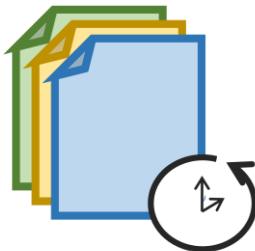


# Mesclando Commits - Conflitos



# Comandos avançados **DESFAZENDO AÇÕES**

# Recuperando Arquivos



```
$ git checkout [--] <arquivo>
```

Recupera o arquivo <arquivo> do último *commit*.

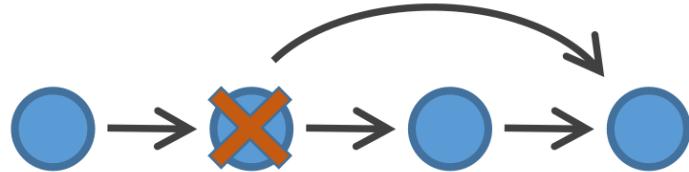
```
$ git checkout <commit> <arq>
```

Recupera o arquivo <arq> do *commit* <commit>.

```
$ git checkout <commit>
```

Recupera os arquivos do *commit* <commit>.

# Revertendo Commits

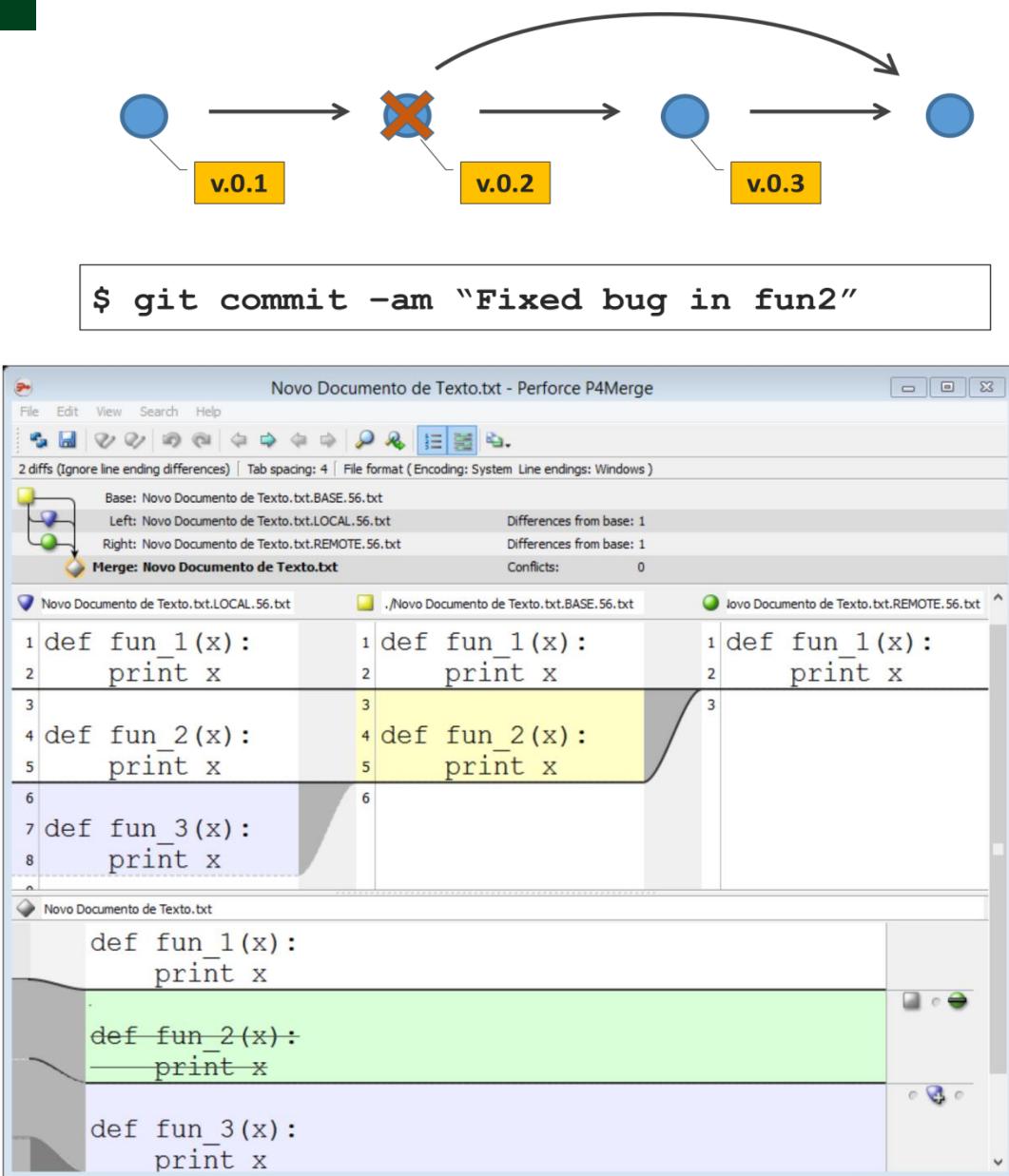
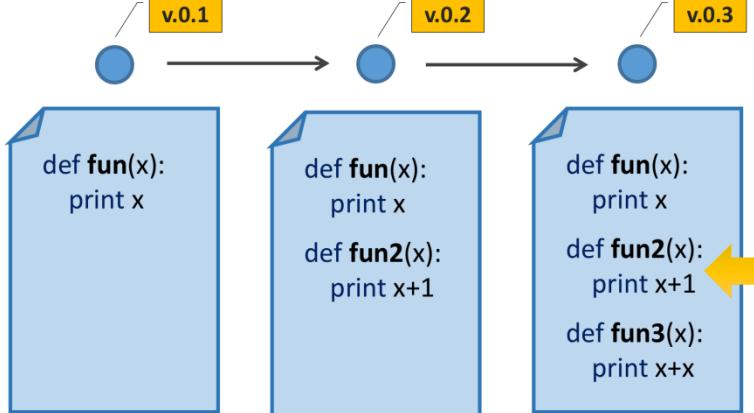


```
$ git revert <commit>
```

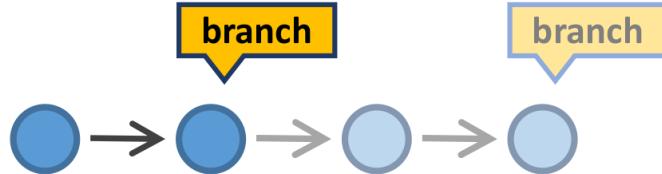
Cria um novo *commit* no *branch* atual que desfaz o que foi introduzido no *commit* <commit>.

- Consertar um *bug* introduzido por um *commit*.
- Não remove o *commit* <commit>

# Revertendo Commits

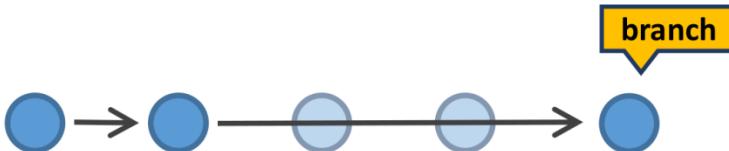


# “Excluindo” Commits



```
$ git reset --soft <commit>
```

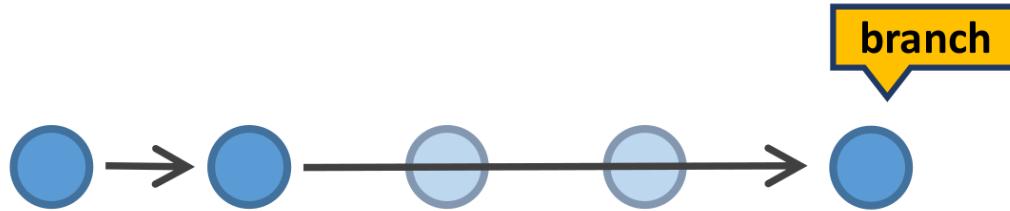
Altera apenas o HEAD para o *commit* <commit>. Não altera a área transitória nem o diretório de trabalho.



```
$ git reset --soft <commit>
$ git commit
```

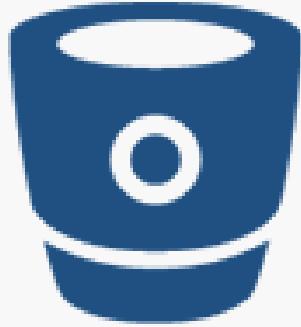
Substitui os *commits* por um único *commit*. O diretório de trabalho não é alterado.

# “Excluindo” Commits



```
$ git reset <commit>  
$ git add <arquivos>  
$ git commit
```

Mantém os arquivos <arquivos> do diretório.



**Bitbucket**



# Repositórios Remotos

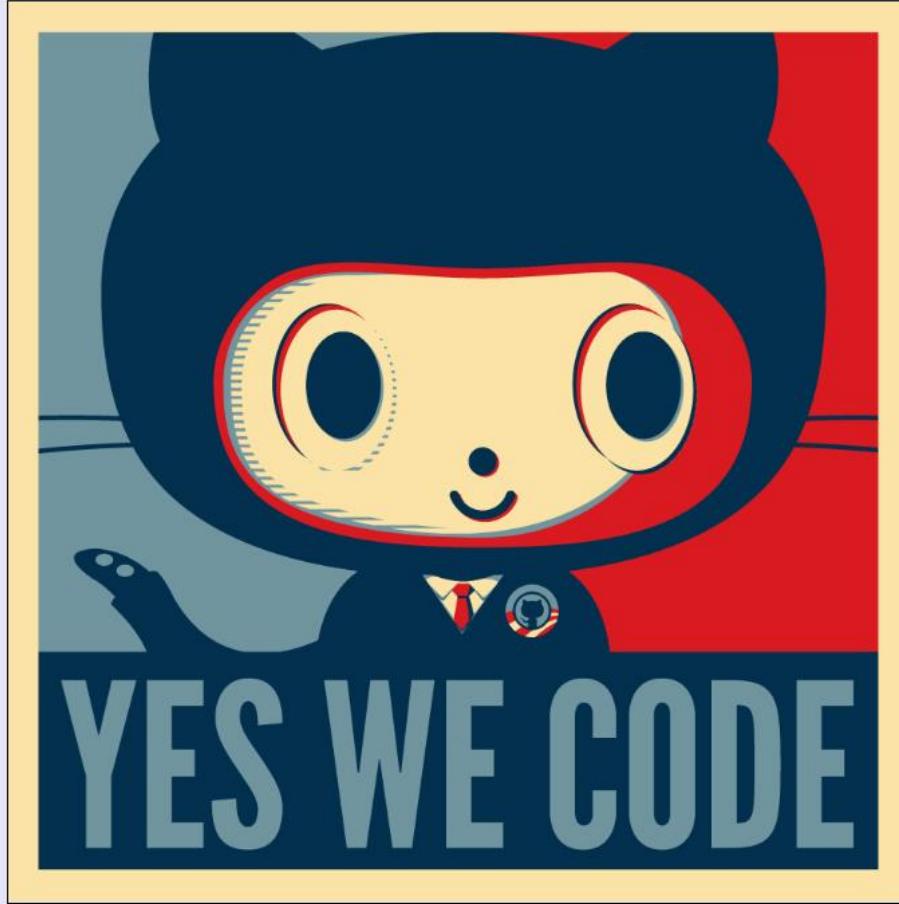


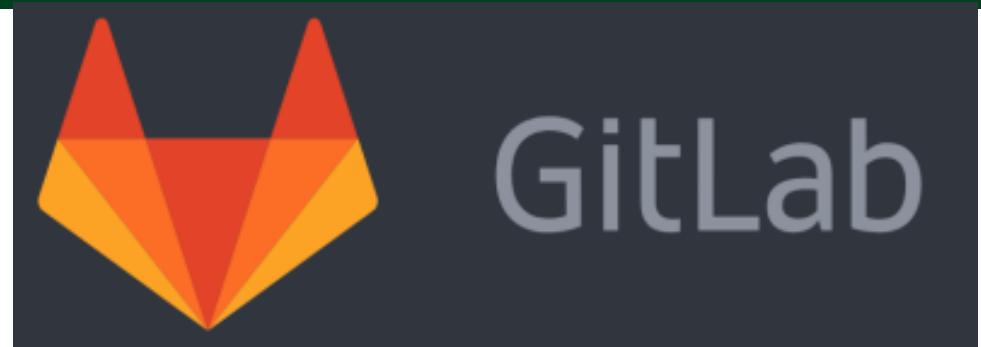
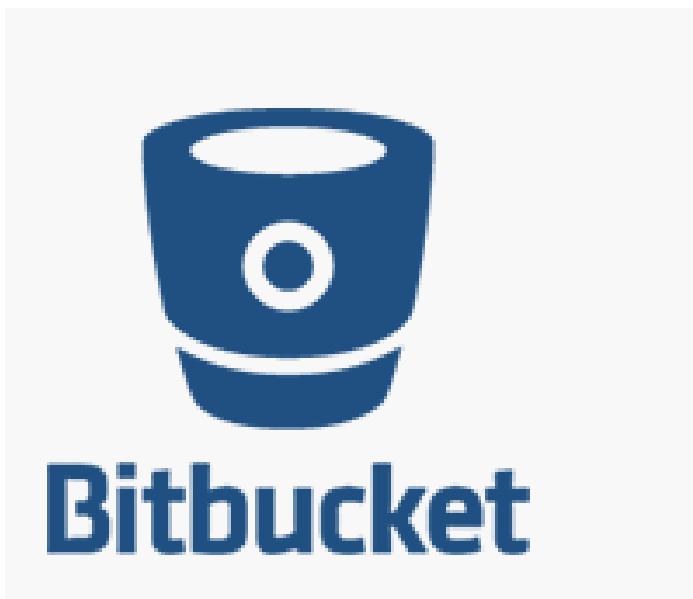
# Alternativas com GIT

- Bitbucket: <https://bitbucket.org/product/>
- Github: <https://github.com/>
- Gitlab: <https://about.gitlab.com/>
- Todos utilizam GIT como linguagem;
- Ferramentas de gerência de projetos;



# Hands-on!

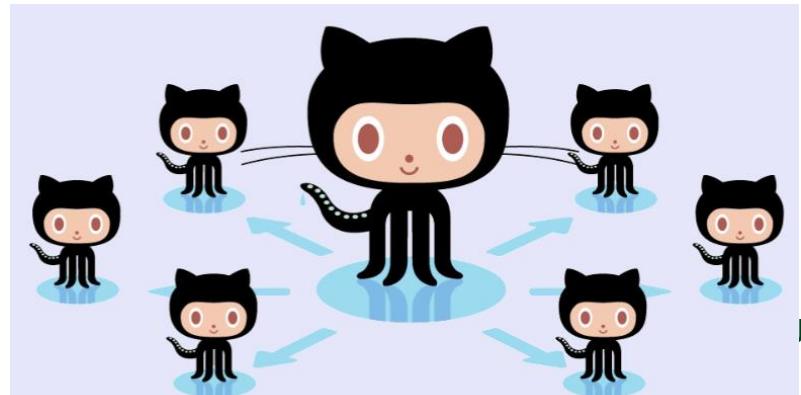




# CONFIGURANDO REPOSITÓRIO REMOTO

# Repositórios Remotos

- Muitas vezes(para não dizer na maioria) não desenvolvemos projetos sozinhos.
- Uma forma de facilitar o compartilhamento de projetos é disponibiliza-lo em um local de fácil acesso entre os integrantes da equipe, além de permitir que o repositório possua sempre a versão mais atualizada dentre os integrantes.



# Repositórios Remotos

- Muitas empresas possuem servidores dedicados para suas equipes, mas há algumas soluções no mercado que cumprem a mesma função.



- Mas daremos foco ao case mais bem sucedido.



- Criar um user github
- Cadastrar uma ssh ou acessar pelo login e senha
- Criar repositorio
- Fazer um add remote
- Fazer um clone direto
- Criar funcao
- Fazer commit e push
- Criar branch
- Fazer merge

# Let's go

- Vamo criar uma conta no github
- Acesse [github.com](https://github.com)
  - <https://github.com/signup?source=login>
  - Acesse informações da conta e *settings*

The screenshot shows a GitHub user profile for 'juliosaracol'. The profile includes a circular profile picture of a man with dark hair and a beard, wearing a black polo shirt. Below the picture is the username 'juliosaracol' and a 'Edit profile' button. To the right, there are sections for 'Popular repositories' showing 'FlexMap', 'minicurso\_stl', 'juliosaracol', 'migoritho', and 'Istools-showcase'. Below these is a 'Contributions in the last year' chart showing activity from July to July of the following year. On the far right, a sidebar shows the user is signed in as 'juliosaracol' and provides links to 'Your profile', 'Your repositories', 'Your codespaces', 'Your projects', 'Your stars', 'Your gists', 'Upgrade', 'Feature preview', 'Help', 'Settings' (which is highlighted with a red box), and 'Sign out'.

# Configurando github

- SSH and GPG Keys
- Click em New SSH KEY

The screenshot shows the GitHub profile page for the user 'juliosaracol'. The left sidebar has a red box around the 'SSH and GPG keys' link. The main content area has two sections: 'SSH keys' and 'GPG keys'. The 'SSH keys' section lists one key: 'julio@julio-RedmiBook-Pro' (SHA256: //0oQPUKFUtErFZPxtSrvn8KwEL0Lj9PYUfga0HMpQ), which was added on 18 Jan 2022 and last used 7 months ago. The 'GPG keys' section indicates there are no GPG keys associated with the account. The 'Vigilant mode' section has a checkbox for 'Flag unsigned commits as unverified'.

Search or jump to... 7 Pull requests Issues Marketplace Explore Go to your profile

juliosaracol Your personal account

Public profile Account Appearance Accessibility Notifications

Access Billing and plans Emails Password and authentication **SSH and GPG keys**

Organizations Moderation

Code, planning, and automation

Repositories Packages GitHub Copilot Pages Saved replies

Security

Code security and analysis

**SSH keys**

This is a list of SSH keys as

**julio@julio-RedmiBook-Pro**  
SHA256: //0oQPUKFUtErFZPxtSrvn8KwEL0Lj9PYUfga0HMpQ  
Added on 18 Jan 2022  
Last used within the last 7 months — Read/write Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

**GPG keys**

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and add it to your account.

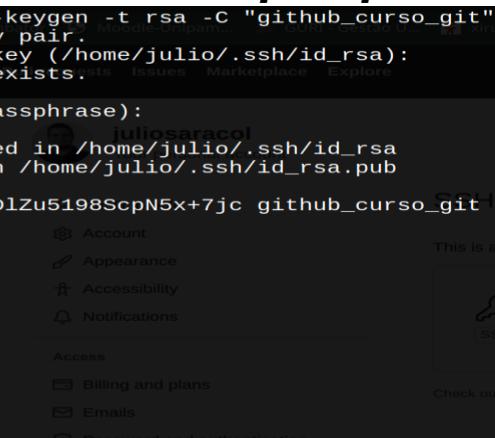
**Vigilant mode**

**Flag unsigned commits as unverified**  
This will include any commit attributed to your account but not signed with your GPG or S/MIME key.  
Note that this will include your existing unsigned commits.

[Learn about vigilant mode.](#)

# Gerando SSH para cadastro no Github

- No terminal da máquina execute:
  - `ssh-keygen -t rsa -C "comment"`
  - `cat ~/.ssh/id_rsa.pub`
  - **Copie o conteúdo da chave do arquivo `id_rsa.pub` para o campo no github.**



```
julio@julio-RedmiBook-Pro:~$ ssh-keygen -t rsa -C "github_curso_git"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/julio/.ssh/id_rsa):
/home/julio/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/julio/.ssh/id_rsa
Your public key has been saved in /home/julio/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:eYqIXX+5x7wqvy+IGUT+QhLCmDlZu5198ScpN5x+7jc github_curso_git
The key's randomart image is:
+---[RSA 3072]---+
| B.
| * o...
| ... +
| .o.o+ = o
| .o+o.S X .
| o oo+.* =
| . o ..=o..+o.
| o o...+ + E
| o=B=o .
+---[SHA256]---+
```

- Testando conexão e login pelo terminal:
  - `ssh git@github.com`



```
julio@julio-RedmiBook-Pro:~/curso_git$ ssh git@github.com
PTY allocation request failed on channel 0
Hi juliosaracol! You've successfully authenticated, but GitHub does not provide shell access.
Connection to github.com closed.
```

# Gerando SSH para cadastro no Github

- No [SSH keys / Add new](#)

The screenshot shows the GitHub 'SSH keys' page under 'Add new'. A new key is being added with the following details:

- Title:** github\_curso\_git
- Type:** ssh-rsa
- Key:** AAAAB3NzaC1yc2EAAAQABAAQOCzu1KcyW1W9ik4rt2Oevq/RDtEVOLz4Fq1Wxmt4cfdlVQLW0oZHM5HijBV

A large gray box covers the key text area. At the bottom right of this box is a small green circular icon with a white letter 'G'.

[Add SSH key](#)

- Test

- **ssh git@github.com**

```
julio@julio-RedmiBook-Pro:~/curso_git$ ssh git@github.com
PTY allocation request failed on channel 0
Hi juliosaracol! You've successfully authenticated, but GitHub does not provide shell access.
Connection to github.com closed.
```

# Criando repositório

- Na página inicial vá em [NEW]

The screenshot shows the GitHub homepage. At the top, there is a dark header bar with the GitHub logo, a search bar containing "Search or jump to...", and navigation links for "Pull requests", "Issues", "Marketplace", and "Explore". Below the header, there are three main sections: "Recent Repositories" (listing repositories like "juliosaracol/migortho" and "juliosaracol/FlexMap"), "Following" (highlighted with a red border), and "For you" (Beta). A red box highlights the "New" button in the "Recent Repositories" section. To the right, there is a "Introduce yourself" section with a sample README.md file from "juliosaracol".

Recent Repositories

New

Following For you (Beta)

Introduce yourself

The easiest way to introduce yourself on GitHub is by creating a RE/

juliosaracol / README.md

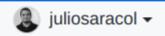
1 - 🌟 Hi, I'm @juliosaracol  
2 - 🌐 I'm interested in ...  
3 - 🌱 I'm currently learning ...  
4 - ❤️ I'm looking to collaborate on ...  
5 - 📧 How to reach me ...  
6

# Criando repositório

## 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 and memorable. Need inspiration? How about [crispy-engine](#)?

Description (optional)

Public

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

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

You are creating a public repository in your personal account.

[Create repository](#)



# It's Done!!

juliosaracol / curso\_de\_git Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Code main 1 branch 0 tags Go to file Add file Code

juliosaracol Initial commit now 1 commit 72a7db1

LICENSE Initial commit now

README.md Initial commit now

README.md

curso\_de\_git

curso da semana academica saecomp2022

About

Readme 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

# Clonando o repositório remoto

- É possível baixar como.zip
- Pelo terminal executar o comando selecionado abaixo:
  - git clone <endereço do repositório>.git

The screenshot shows a GitHub repository page for a repository named 'curso\_de\_git'. The page includes a sidebar with repository statistics (main branch, 1 branch, 0 tags), a file list (LICENSE, README.md), and a main content area with the README.md file's content. A modal window titled 'Clone' is open, showing three cloning options: HTTPS (selected), SSH, and GitHub CLI. The HTTPS URL is highlighted with a blue box: `git@github.com:juliosaracol/curso_de_git`. Below the URL, there is a note about using a password-protected SSH key and a 'Download ZIP' button.

main ▾ 1 branch 0 tags

juliosaracol Initial commit

LICENSE Initial commit

README.md Initial commit

README.md

**curso\_de\_git**

curso da semana academica saecomp2022

Clone

HTTPS SSH GitHub CLI

git@github.com:juliosaracol/curso\_de\_git

Use a password-protected SSH key.

Download ZIP

About

curso da semana academica saecomp2022

Readme 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

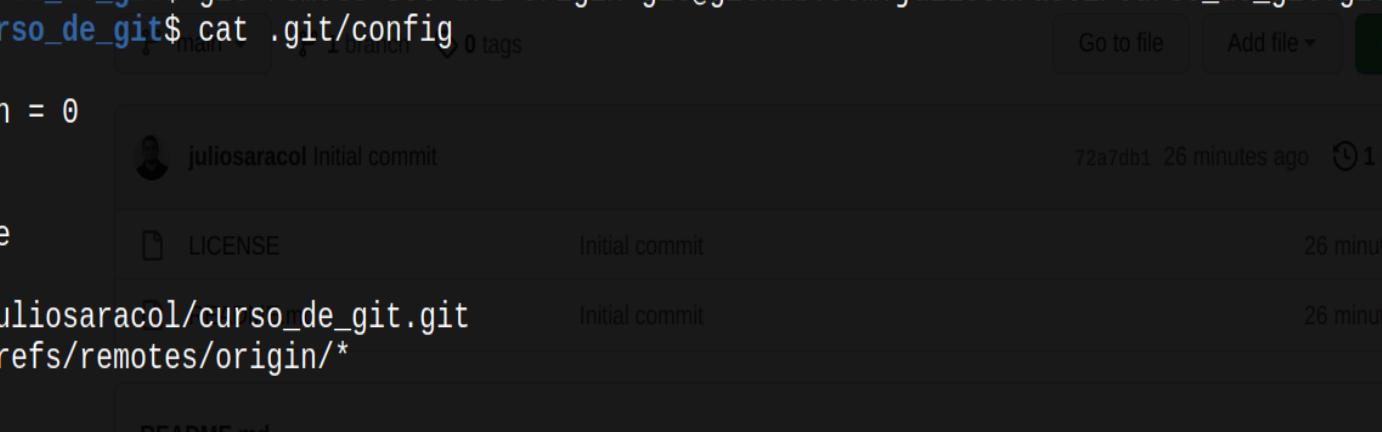
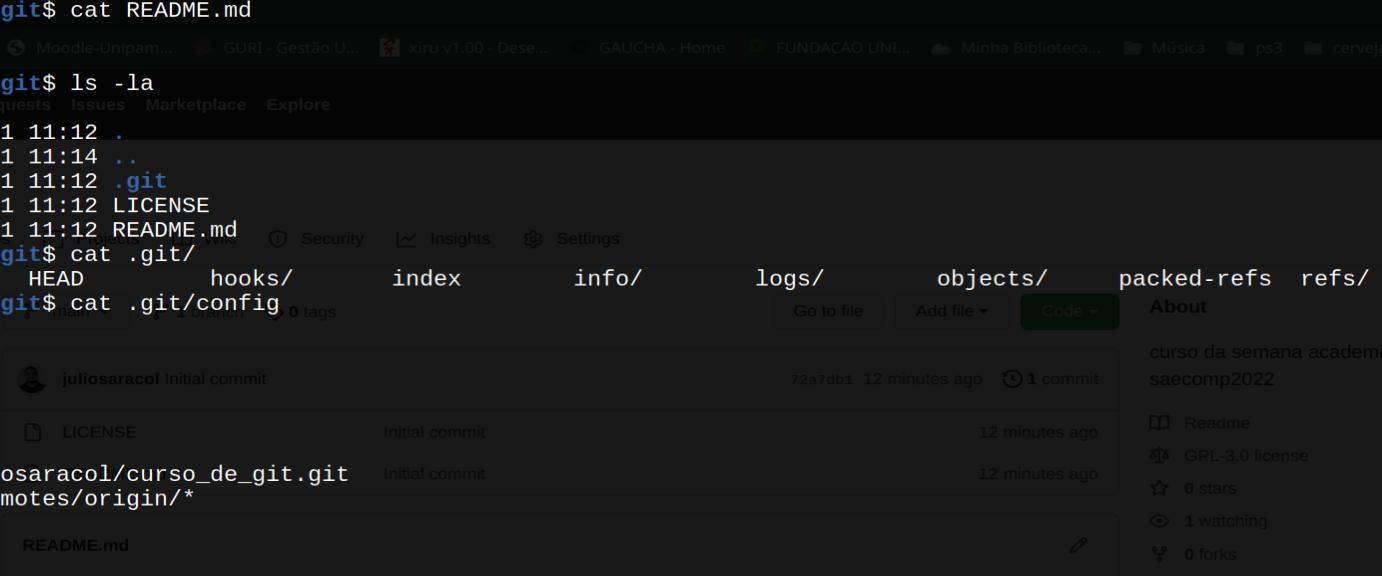
# Clonando na máquina local

```
julio@julio-RedmiBook-Pro:~$ git clone https://github.com/juliosaracol/curso_de_git.git
Cloning into 'curso_de_git'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 12.48 KiB | 319.00 KiB/s, done.
julio@julio-RedmiBook-Pro:~$
```

```
julio@julio-RedmiBook-Pro:~$ ls
Android           'Área de Trabalho'   Documentos    Dropbox      magic_exemplos  ProgramasRFB  tools
arduino_projects curso_de_git       Downloads     fiction      migortho2      snap          'VirtualB
julio@julio-RedmiBook-Pro:~$ cd curso_de_git/
julio@julio-RedmiBook-Pro:~/curso_de_git$ ls
LICENSE  README.md
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat README.md
# curso_de_git
curso da semana academica saecomp2022
julio@julio-RedmiBook-Pro:~/curso_de_git$
```

# Verificando informações do clone

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ ls +  
LICENSE README.md  
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat README.md  
# curso_de_git  
curso da semana academica saecomp2022  
julio@julio-RedmiBook-Pro:~/curso_de_git$ ls -la  
total 52  
drwxrwxr-x 3 julio julio 4096 jul 21 11:12 .  
drwxr-xr-x 57 julio julio 4096 jul 21 11:14 ..  
drwxrwxr-x 8 julio julio 4096 jul 21 11:12 .git  
-rw-rw-r-- 1 julio julio 35149 jul 21 11:12 LICENSE  
-rw-rw-r-- 1 julio julio 53 jul 21 11:12 README.md  
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat .git/  
branches/ config description HEAD hooks/ index info/ logs/ objects/ packed-refs refs/  
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat .git/config  
[core]  
repositoryformatversion = 0  
filemode = true  
bare = false  
logallrefupdates = true  
[remote "origin"]  
url = https://github.com/juliosaracol/curso_de_git.git  
fetch = +refs/heads/*:refs/remotes/origin/*  
[branch "main"]  
remote = origin  
merge = refs/heads/main  
julio@julio-RedmiBook-Pro:~/curso_de_git$ █  
julio@julio-RedmiBook-Pro:~/curso_de_git$ git remote set-url origin git@github.com:juliosaracol/curso_de_git.git  
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat .git/config  
[core]  
repositoryformatversion = 0  
filemode = true  
bare = false  
logallrefupdates = true  
[remote "origin"]  
url = git@github.com:juliosaracol/curso_de_git.git  
fetch = +refs/heads/*:refs/remotes/origin/*  
[branch "main"]  
remote = origin  
merge = refs/heads/main
```



# Verificando informações do clone

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ ls
LICENSE README.md
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat README.md
# curso_de_git
curso da semana academica saecomp2022
julio@julio-RedmiBook-Pro:~/curso_de_git$ ls -la
total 52
drwxrwxr-x  3 julio julio  4096 jul 21 11:12 .
drwxr-xr-x  57 julio julio  4096 jul 21 11:14 ..
drwxrwxr-x  8 julio julio  4096 jul 21 11:12 .git
-rw-rw-r--  1 julio julio 35149 jul 21 11:12 LICENSE
-rw-rw-r--  1 julio julio   53 jul 21 11:12 README.md
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat .git/
branches/ config      description HEAD      hooks/    index     info/    logs/    obj/
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat .git/config
[core]
  repositoryformatversion = 0
  filemode = true
  bare = false
  logallrefupdates = true
[remote "origin"]
  url = https://github.com/juliosaracol/curso_de_git.git
  fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
  remote = origin
  merge = refs/heads/main
julio@julio-RedmiBook-Pro:~/curso_de_git$
```

- Informações da “origin” e o branch atual “main”

# Trabalhando Localmente

- **pull origin master** para verificar se estamos atualizados com o repositório remoto.
- Crie um arquivo “ `exemplo.c`” dentro da pasta do repositório “`curso_de_git`”
- Crie um “olá mundo”

# Enviando o arquivo ao repositório

main ▾ [curso\\_de\\_git / exemplo.c](#)



juliosaracol primeiro commit

1 contributor

9 lines (6 sloc) | 88 Bytes

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5
6     printf("ola mundo\n");
7
8     return 0;
9 }
```

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ cat > exemplo.c
#include <stdio.h>
#include <stdlib.h>

int main(){
    printf("ola mundo\n");
    return 0;
}

{julio@julio-RedmiBook-Pro:~/curso_de_git$ cat exemplo.c
#include <stdio.h>
#include <stdlib.h>

int main(){
    printf("ola mundo\n");
    return 0;
}
{julio@julio-RedmiBook-Pro:~/curso_de_git$ git add exemplo.c
{julio@julio-RedmiBook-Pro:~/curso_de_git$ git commit -am 'primeiro commit'
[main 11457eb] primeiro commit
 1 file changed, 9 insertions(+)
 create mode 100644 exemplo.c
{julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin main
Username for 'https://github.com': 
```

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 382 bytes | 382.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:juliosaracol/curso_de_git.git
    72a7db1..11457eb  main -> main
julio@julio-RedmiBook-Pro:~/curso_de_git$ 
```



# Enviando o arquivo ao repositório

main ▾ curso\_de\_git / exemplo.c

juliosaracol primeiro commit

1 contributor

9 lines (6 sloc) | 88 Bytes

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5
6     printf("ola mundo\n");
7
8     return 0;
9 }
```

# Modificando o arquivo original

- Faça modificações no arquivo,
  - Faça um git add, git commit e git push

The screenshot shows a terminal session on a Mac OS X system. The user is in a directory named 'curso\_de\_git'. They open a file named 'exemplo.c' with 'nano 4.8'. In the code editor, they change the printf statement to print a message in Portuguese: 'ola mundo seja bem vindo a SAECOMP e o curso de git'. After saving the file, they run several Git commands:

```
GNU nano 4.8
#include <stdio.h>
#include <stdlib.h>

int main(){

printf("ola mundo seja bem vindo a SAECOMP e o curso de git\n");

return 0;
}

julio@julio-RedmiBook-Pro:~/curso_de_git$ nano exemplo.c
julio@julio-RedmiBook-Pro:~/curso_de_git$ git add *
julio@julio-RedmiBook-Pro:~/curso_de_git$ git commit -am 'novo commit'
[main 9f05e43] novo commit
 1 file changed, 2 insertions(+), 2 deletions(-)
julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 361 bytes | 361.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:juliosaracol/curso_de_git.git
   include <stdio.h>
      9f05e43  main -> main
    11457eb..9f05e43  main -> main
   #include <stdlib.h>
```

## novo commit

main



juliosaracol committed 2 minutes ago

Showing 1 changed file with 2 additions and 2 deletions.

exemplo.c

...

@@ -3,7 +3,7 @@

3 3

4 4 int main(){

5 5

6 - printf("ola mundo\n");

6 + printf("ola mundo seja bem vindo a SAECOMP e o curso de git\n");

7 7

8 8 return 0;

9 - } ⊖

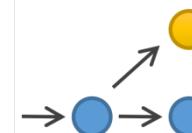
9 + }

# Criando um novo branch

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ git status  
No ramo main  
Your branch is up to date with 'origin/main'.  
  exemplo.c  
nothing to commit, working tree clean
```

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ git checkout -b novoBranch  
Switched to a new branch 'novoBranch'  
julio@julio-RedmiBook-Pro:~/curso_de_git$ git status  
No ramo novoBranch  
nothing to commit, working tree clean  
julio@julio-RedmiBook-Pro:~/curso_de_gits$
```

- Vamos criar uma nova função ola\_mundo



\$ git branch [-a]

Exibe os *branches* existentes. Na forma completa, exibe também os *branches* remotos.

\$ git branch <branch> [<base>]

Cria o *branch* <branch> a partir do *commit* <base>.

\$ git checkout -b <branch>

Cria o *branch* <branch> e altera para ele.

GNU nano 4.8

```
#include <stdio.h>
#include <stdlib.h>/juliosaracol/curso_de_git

void ola_mundo(void);
int main(){
    ola_mundo();

    return 0;
}

void ola_mundo(void){
printf("ola mundo seja bem vindo a SAECOMP e o curso de git\n");
}
```

Gmail Facebook Google Keep Muambator Moodle-Unipam... GURI - Gestão U... xiru

Search or jump to... Pull requests Issues Marketplace Explore

juliosaracol / curso\_de\_git (Public)

Issues Pull requests Actions Projects Wiki Security In

main ▾ 1 branch 0 tags

juliosaracol novo commit

LICENSE Initial co

README.md Initial co

exemplo.c novo co

^G Obter Ajuda ^O Gravar ^W Onde está? ^K Recort txt ^J J mingues Jr.
^X Sair ^R Ler o arq ^\ Substituir ^U Colar txt ^T Ve

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ git add *
julio@julio-RedmiBook-Pro:~/curso_de_git$ git commit -am 'adicionando função o
[novoBranch e995cbe] adicionando função ola mundo no branch
 1 file changed, 5 insertions(+), 1 deletion(-)
julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin main
^C
julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin
HEAD juliosaracol/curso_de_git  main          novoBranch      origin/HEAD    origin/main
julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin
HEAD > main           novoBranch      origin/HEAD    origin/main
julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin novoBranch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads27 novoBranch had recent pushes less than a minute ago
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 405 bytes | 405.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)  □ main ▾  □ 1 branch  □ 0 tags
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'novoBranch' on GitHub by visiting:
remote:     https://github.com/juliosaracol/curso_de_git/pull/new/novoBranch
remote:
To github.com:juliosaracol/curso_de_git.git
 * [new branch]      novoBranch -> novoBranch
julio@julio-RedmiBook-Pro:~/curso_de_git$
```

## All branches

**main**  Updated 12 minutes ago by juliosaracol

**novoBranch**  Updated 1 minute ago by juliosaracol

-o 1 commit

1 file changed

1 contributor

-o Commits on Jul 21, 2022

adicionando função ola mundo no branch

 juliosaracol committed 2 minutes ago



e995c

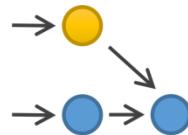
 Showing 1 changed file with 5 additions and 1 deletion.

Split

▼ 6  exemplo.c 

```
... ... @@ -1,9 +1,13 @@
1 1 #include <stdio.h>
2 2 #include <stdlib.h>
3 3
4 + void ola_mundo(void);
5 int main(){
6 - printf("ola mundo seja bem vindo a SAECOMP e o curso de git\n");
7 + ola_mundo();
8
9 return 0;
10+
11+ void ola_mundo(void){
12+ printf("ola mundo seja bem vindo a SAECOMP e o curso de git\n");
9 13 }
```

# Merge



```
$ git merge <branch>
```

Mescla os *commits* do *branch* <branch> para o *branch* atual.

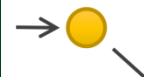
```
$ git merge <branch> --no-ff
```

Mescla os *commits* do *branch* <branch> para o *branch* atual sem *fast-forward*.

- Vamos ao branch main
- Vamos fazer o merge com o branch novoBranch

```
julio@julio-RedmiBook-Pro:~/curso_de_git$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
julio@julio-RedmiBook-Pro:~/curso_de_git$ git merge novoBranch
Updating 9f05e43..e995cbe
Fast-forward
  exemplo.c | 6 +++++-
  1 file changed, 5 insertions(+), 1 deletion(-)
julio@julio-RedmiBook-Pro:~/curso_de_git$ git add *
julio@julio-RedmiBook-Pro:~/curso_de_git$ git commit -m 'merge dos branchs'
No ramo main
Seu ramo está à frente de 'origin/main' por 1 submissão.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
julio@julio-RedmiBook-Pro:~/curso_de_git$ git push origin main
Total 0 (delta 0), reused 0 (delta 0)
To github.com:juliosaracol/curso_de_git.git
  9f05e43..e995cbe  main -> main
julio@julio-RedmiBook-Pro:~/curso_de_git$
```



## novo commit

main



juliosaracol committed 18 minutes ago

Showing 1 changed file with 2 additions and 2 deletions.

⌄ ⌂ 4 ████ exempo.c ⌂

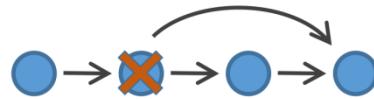
↑...	@@ -3,7 +3,7 @@
3	3
4	4 int main(){
5	5
6	- printf("ola mundo\n");
6	+ printf("ola mundo seja bem vindo a SAECOMP e o curso de git\n");
7	7
8	8 return 0;
9	- } ⊖
9	+ }

0 comments on commit 9f05e43

To github.com:juliosaracol/curso\_de\_git.git Write Preview  
9f05e43..e995cbe main -> main

Prof. Julio S. Domingues Jr.

julio@julio-RedmiBook-Pro:~/curso\_de\_git\$ □ Leave a comment



# Revertendo

- Voltando ao estado inicial

```
$ git revert <commit>
```

Cria um novo *commit* no *branch* atual que desfaz o que foi introduzido no *commit* <commit>.

- Corrigir um *bug* introduzido por um *commit*.
- Não remove o *commit* <commit>

The screenshot shows a GitHub repository interface. At the top, there are navigation buttons: 'main' (with a dropdown arrow), '2 branches', '0 tags', 'Go to file', 'Add file', and 'Code'. Below this, a list of commits is shown:

- juliosaracol** Revert "Initial commit" ... bc28930 5 minutes ago 5 commits
- juliosaracol** exemplo.c adicionando função ola mundo no branch 17 minutes ago

At the bottom of the repository view, there is a light blue call-to-action bar with the text "Help people interested in this repository understand your project by adding a README." and a green button labeled "Add a README".

# Thanks!

juliosaracol@gmail.com

juliiodomingues@unipampa.edu.br

SAECOMP 2022



# Referências

- <https://git-scm.com/book/en/v2>
- <https://minicursogit.github.io/#/9/12>
- Aprendendo Git - Bismarck Gomes Souza Júnior (<https://pt.slideshare.net/bismarckjunior/aprendendo-git>)
- Cursos:
  - <http://try.github.io/>
  - <http://gitreal.codeschool.com/>