MINICURSO DE GIT E GITHUB

Version Control

@allythy

- Cursando Gestão da Tecnologia da Informação
- Colaborador e autor do SempreUpdate
- Coordenador do PotiLivre
- Fundador e Colaborador do PHP-RN
- Coordenador do GruPy-RN
- Colaborador do Debian
- Software Livre
- Segurança da informação

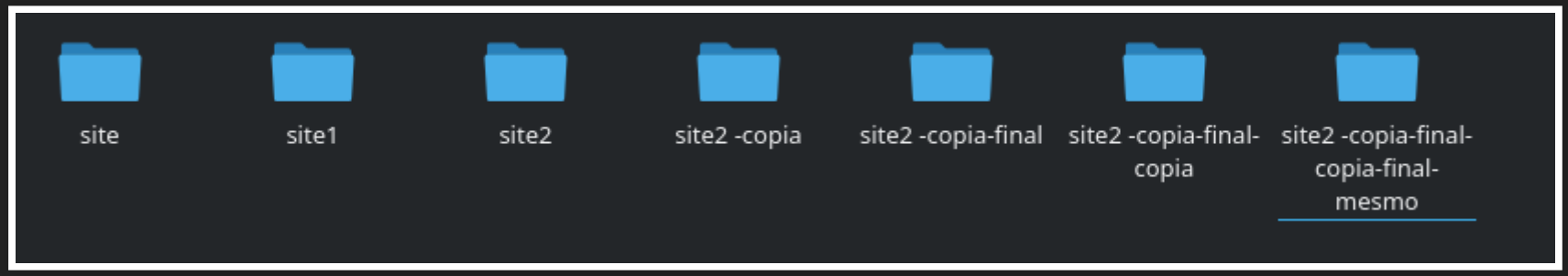
O QUE É UM SISTEMA DE VERSÃO DE CONTROLE (VCS) ?

- Reverter um um arquivo ou projeto

- Comparar as mudanças

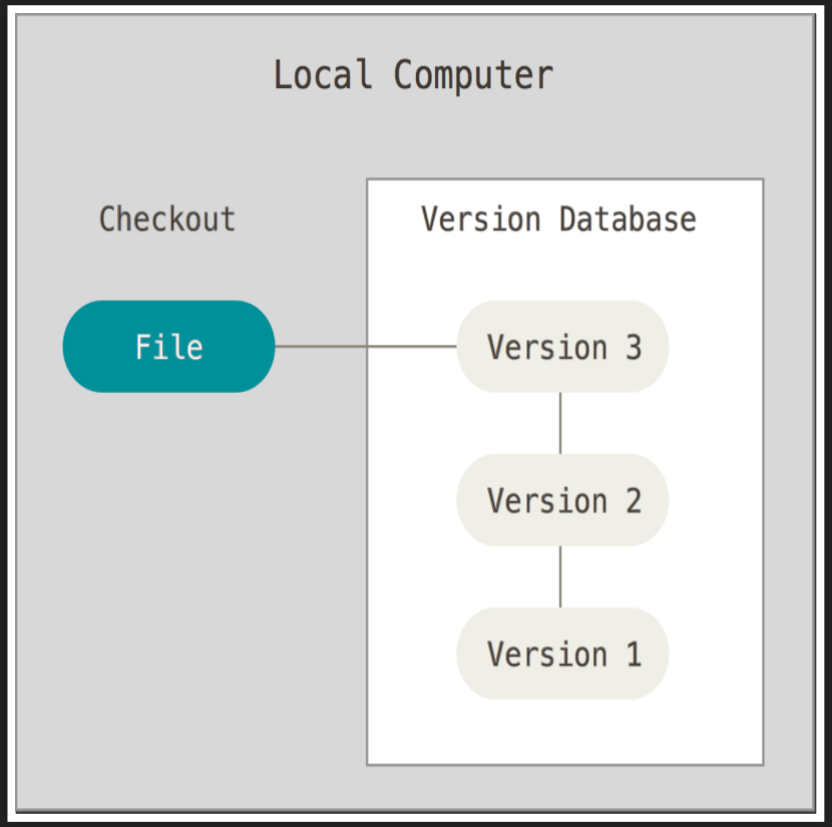
- Saber quem fez alguma alteração

POR QUE EU DEVERIA USAR ?

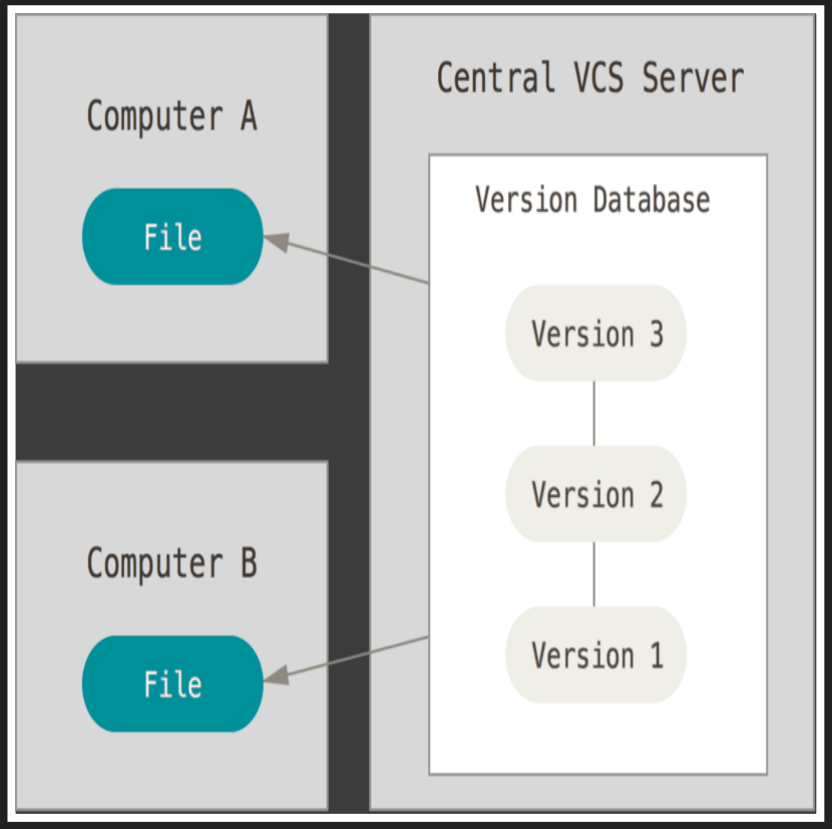


TIPOS DE SISTEMA DE CONTROLE DE VERSÃO

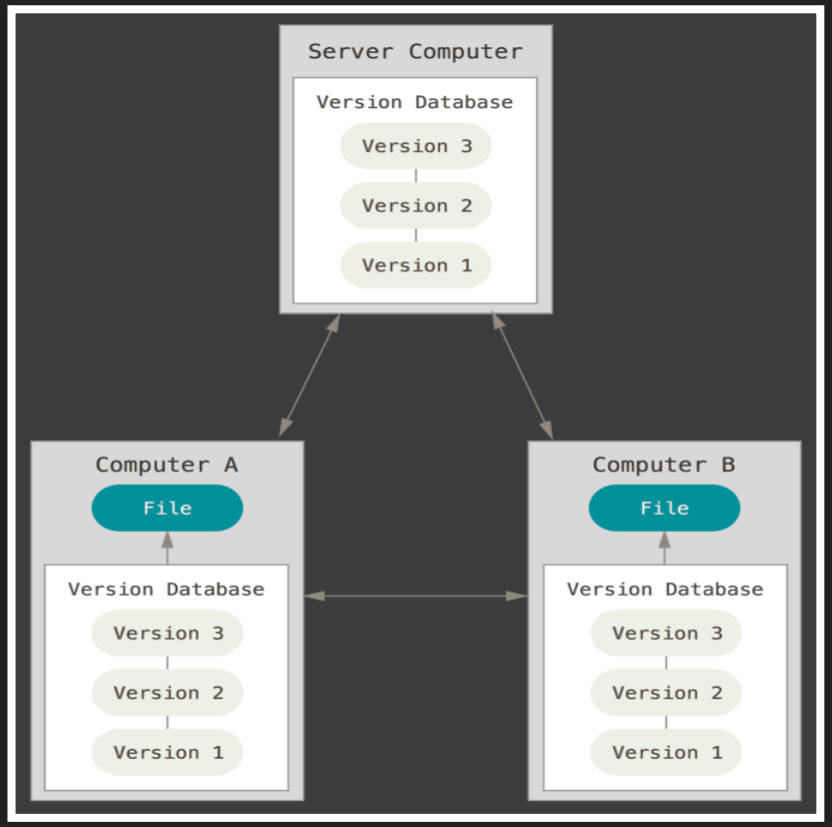
SISTEMAS DE CONTROLE DE VERSÃO LOCAL



SISTEMAS DE CONTROLE DE VERSÃO CENTRALIZADO (CVCS)

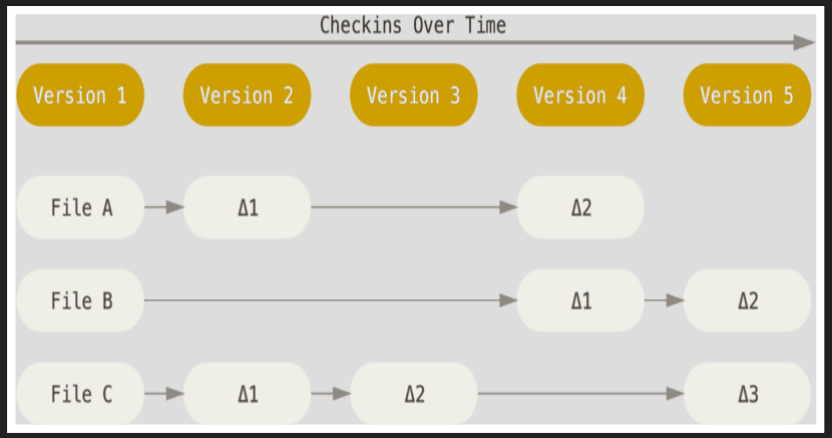


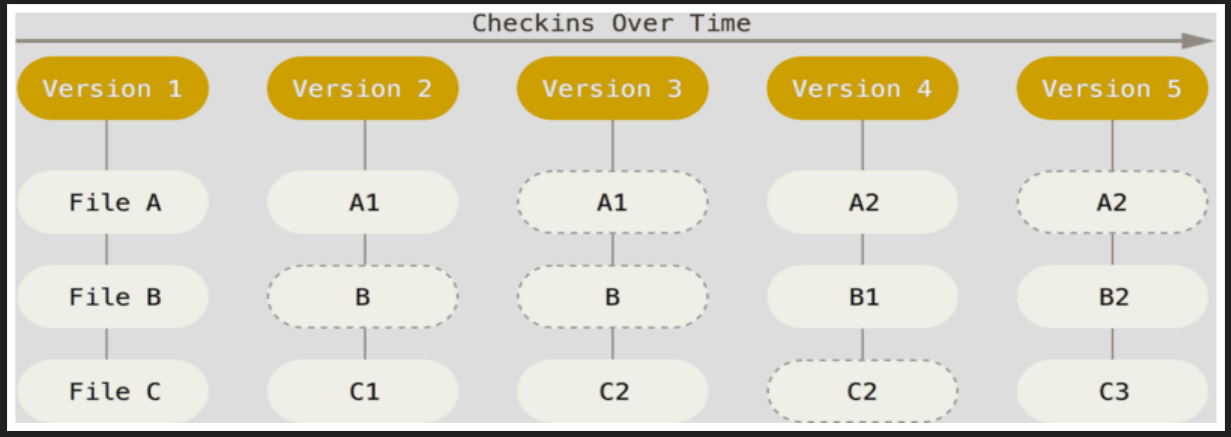
SISTEMAS DE CONTROLE DE VERSÃO DISTRIBUÍDO (DVCS)



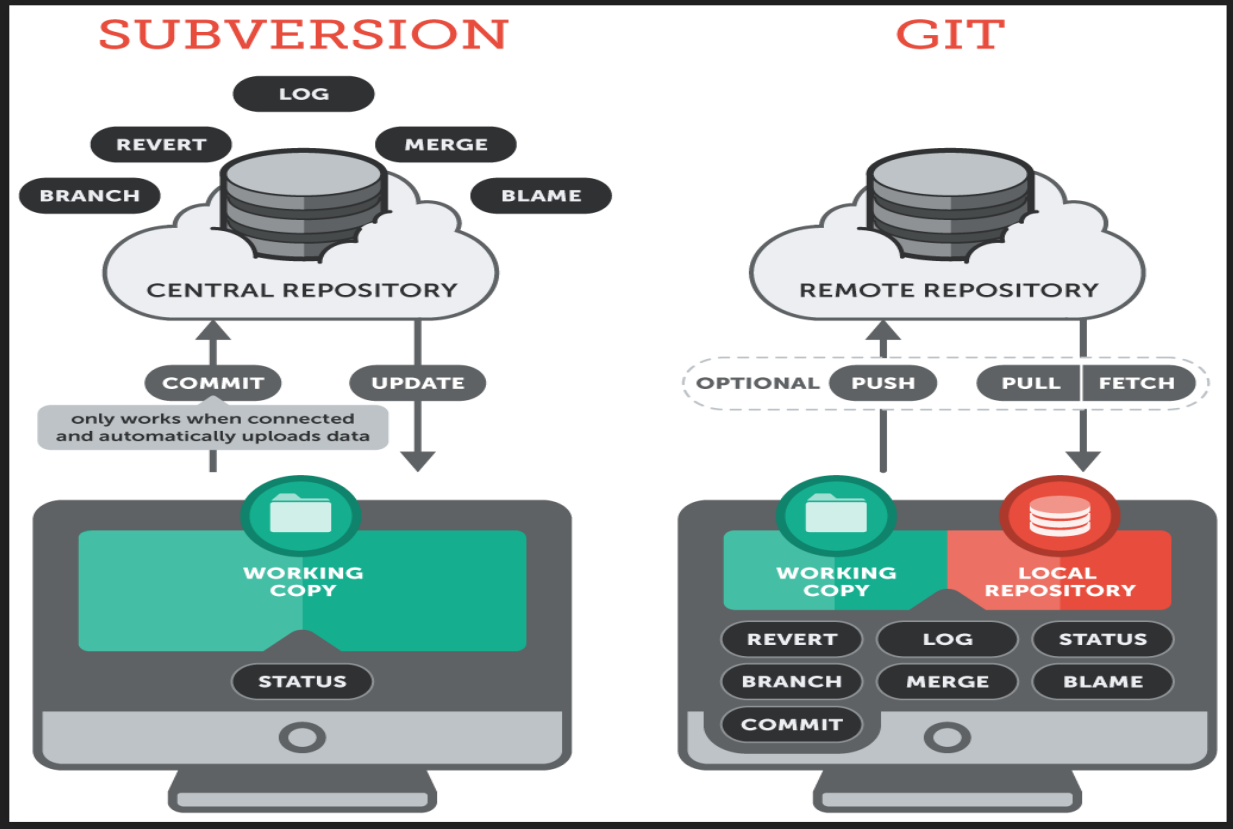
POR QUE EU DEVERIA USAR O GIT ?

A FORMA QUE ELE TRATA OS DADOS





QUASE TODAS AS OPERAÇÕES SÃO LOCAIS



POSSUI INTEGRIDADE

SHA-1

```
74a39bb962c5f40b0c6ab1db1704405d33814b50
7e9c515f547087584976ed4d6c65442cd2b7c7bf
5396e0a2f5c8062fba49286484ac8348204b7015
f977cdaaeb22b6e14a06ae59cc623272a7ac1c68
1d8b9eedf87021ea2b17b75c7f297ea93fbfdda0
```

QUEM ESTÁ USANDO ?

QUEM CRIOU ?

Linus Torvalds - 2005

INSTALAÇÃO DO GIT NO GNU/LINUX

No Debian e derivados:

```
sudo apt install git
```

No Fedora:

```
yum install git-core
```

CONFIGURAÇÕES INICIAS

Define o nome do úsuario:

```
 git config --global user.name "John Doe"
```

Define o endereço dee-mail :

```
git config --global user.email johndoe@example.com
```

Define um editor de texto padrão:

```
git config --global core.editor vim
```

Verificando Suas Configurações:

```
git config --list
```

OBTENDO AJUDA

Documentação Online: Pro Git

Através da linha de comando:

```
git help comando
```

```
man git
```

CRIANDO UM PROJETO LOCAL

INICIALIZANDO UM REPOSITÓRIO:

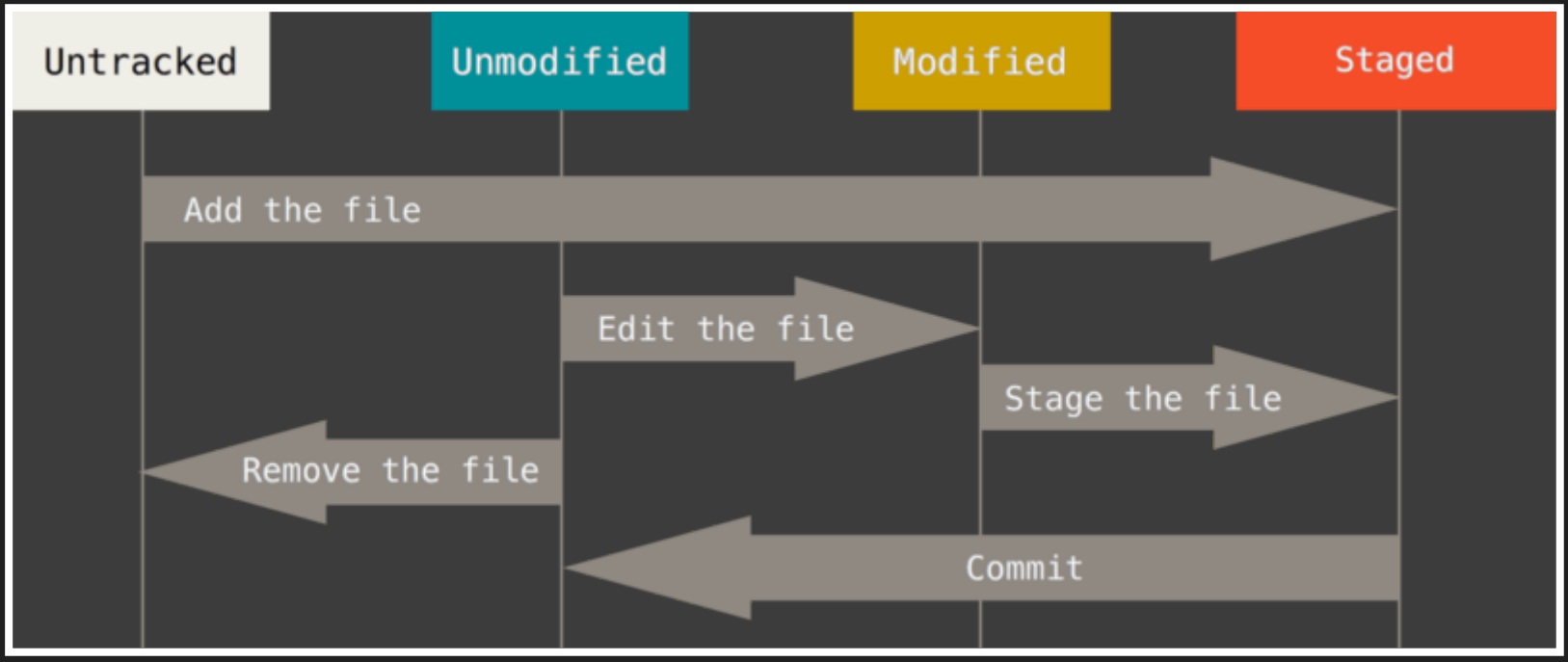
```
git init
```

Crie um arquivo chamado aula1:

```
touch aula1
```

Digite o comando abaixo para ver o estado do repositório:

```
git status
```

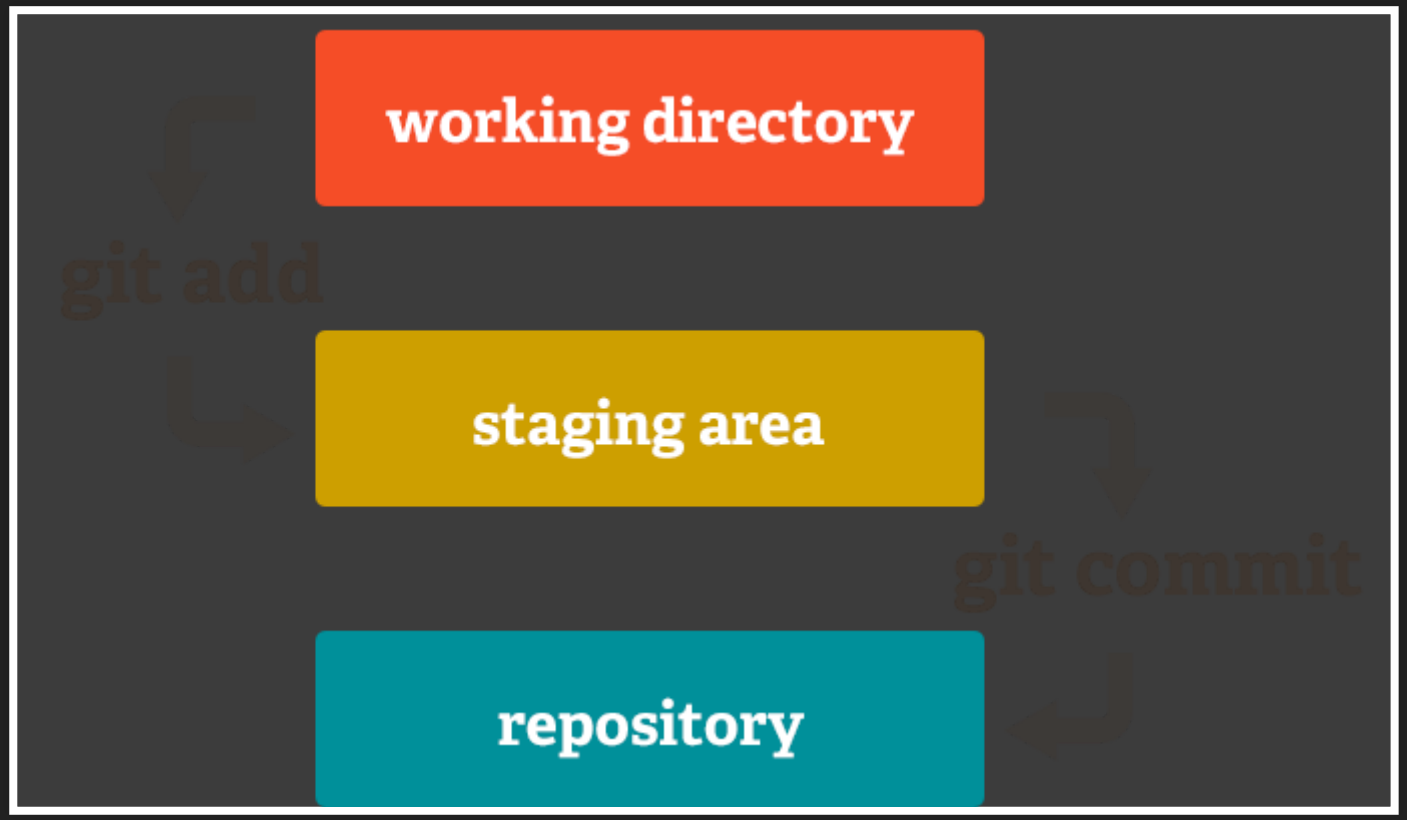

Untracked
Unmodified
Modified
Staged
Add the file
Edit the file
Stage the file
Remove the file
Commit

MONITORANDO OS ARQUIVOS:

```
git add arquivo1 arquivo2
```

ou

```
git add .
```


working directory
git add
staging area
git commit
repository

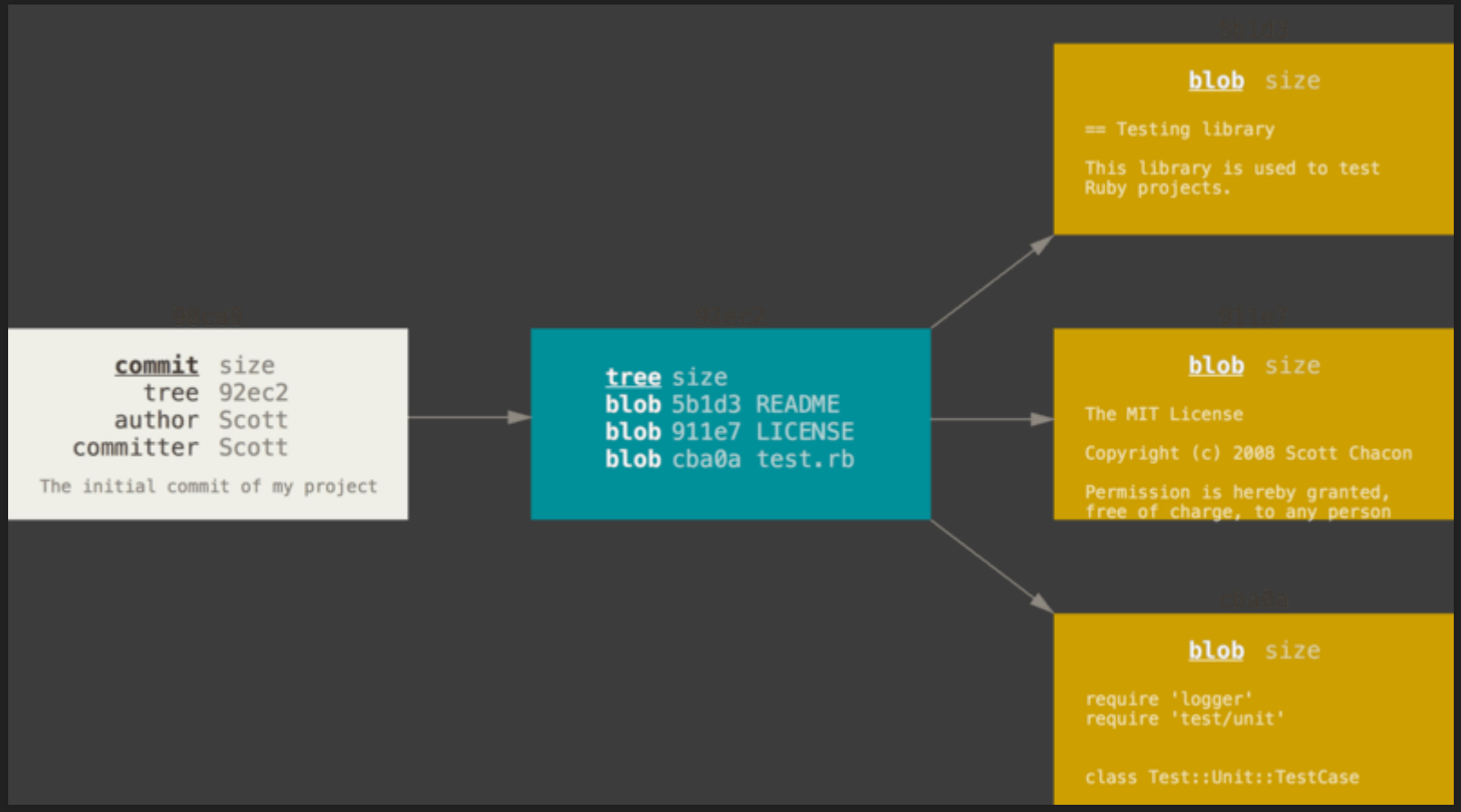
CONSOLIDANDO AS ALTERAÇÕES:

```
git commit
```

ou

```
git commit -m "Descrição do que você fez nesse commit"
```

O QUE O GIT TÁ FAZENDO POR BAIXO




commit size
tree 92ec2
parent
author Scott
committer Scott
The initial commit of my project
commit size
tree 184ca
parent 98ca9
author Scott
committer Scott
Fixed bug #1328 – stack overflow under certain conditions
commit size
tree 0de24
parent 34ac2
author Scott
committer Scott
add feature #32 – ability to add new formats to the central interface
Snapshot A
Snapshot B
Snapshot C

VISUALIZANDO O HISTÓRICO DE COMMITS:

```
git log
```

Mostrando o histórico em uma única linha:

```
git log --oneline
```

Mostrando o commit de autor:

```
git log --author=allythy
```

Mostrando o commit de autor:

```
git shortlog
```

FERRAMENTA VISUAL PARA VER O HISTÓRICO DE COMMITS

Instale o gitk:

```
sudo apt install gitk
```

Como usar:

```
gitk &
```

DESFAZENDO ALTERAÇÕES

Modificar a mensagem do último commit:

```
git commit --amend
```

Você fez o um commit e esqueceu de colocar um arquivo que estava na staging area:

```
git commit -m 'primeiro commit'
git add novoArquivo
git commit --amend
```

DESFAZENDO UM ARQUIVO MODIFICADO

Cenário:

```
touch arquivo
git add arquivo
git status
echo "mensagem" > arquivo
git status
```

Quero descartar essa modificação:

```
git checkout -- arquivo
```

TIRANDO UM ARQUIVO DA STAGING AREA

```
git reset arquivo
```

DESFAZENDO UM COMMIT

```
--soft
--mixed
--hard
```

DESFAZ UM COMMIT, MAS DEIXA OS ARQUIVOS NA STAGING AREA

```
git reset --soft HEAD~1
```

Desfaz um commit, mas deixa os arquivos na working directory

```
git reset --mixed HEAD~1
```

Desfaz o commit deixando igual ao commit que você informou, sem nada na Staging Area ou no working directory

```
git reset --hard HEAD~1
```

TEM COMO DESFAZER UM RESET ?

```
git reflog
```

```
git reset --hard número-do-commit
```

TRABALHANDO COM BRANCH

O QUE É UMA BRANCH ?

Uma branch é uma novo linha de desenvolvimento que permite isolar o código de uma nova funcionalidade, matendo a linha base estáve, evitando o represamento de código. No git, o uma branch é apenas uma referência para um commit.

Criando uma branch

```
git branch nome-da-branch
```

Exibindo as branch do repositório

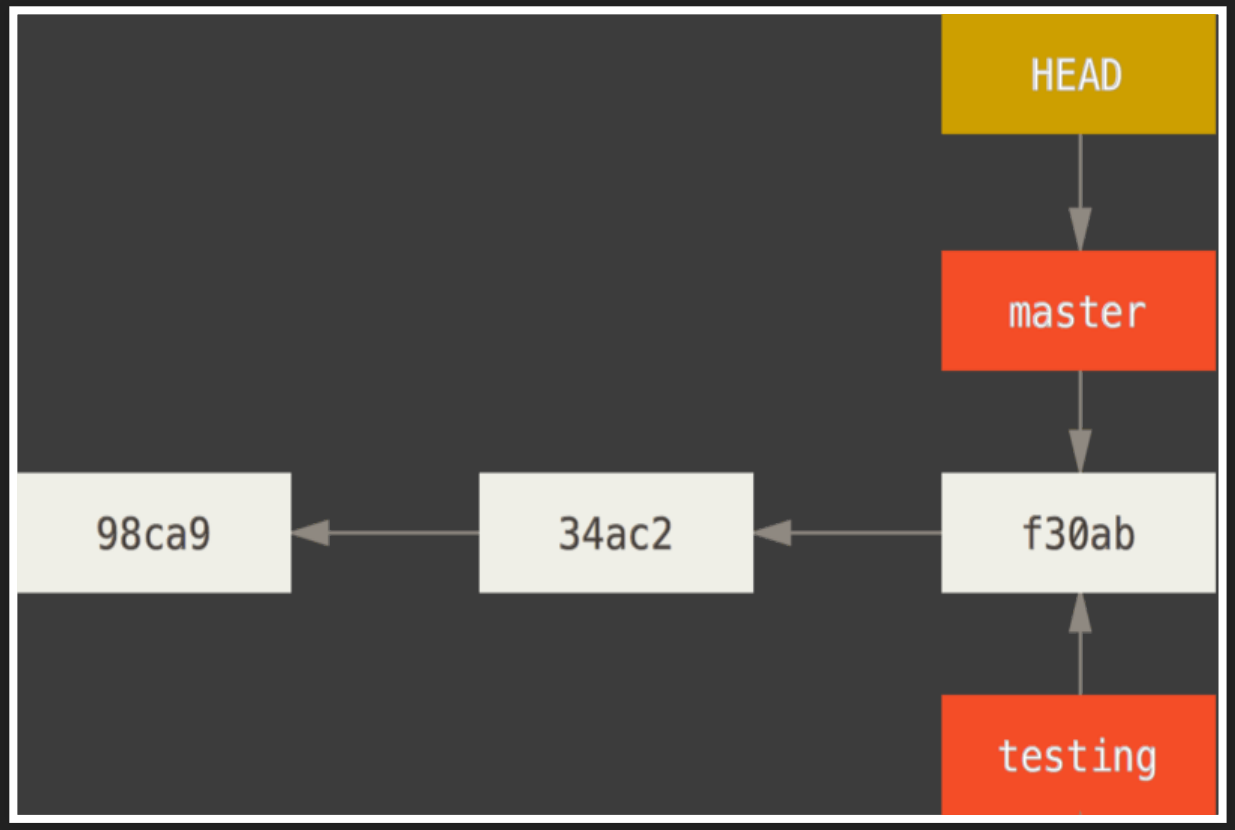
```
git branch
```

Trocando de brach

```
git checkout nome-da-branch
```

COMO O GIT SABE O BRANCH EM QUE VOCÊ ESTÁ ATUALMENTE ?

Ele mantém um ponteiro especial chamado HEAD



Cenário:

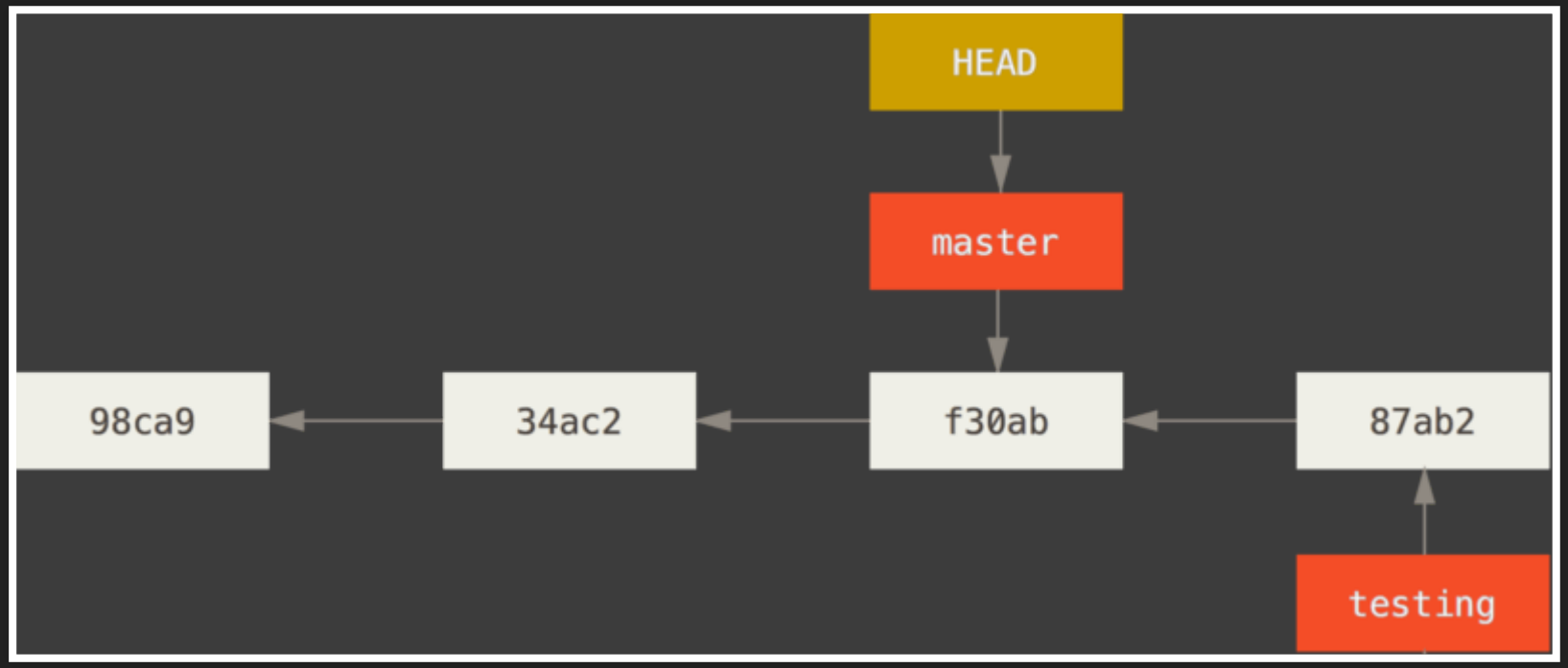
```
touch arquivo1
git add .
git commit -m" branch testing"
```

```
git checkout master
```

FAZENDO O MERGE

```
git merge nome-da-branch
```

O QUE É UM MERGE FAST FORWARD ?

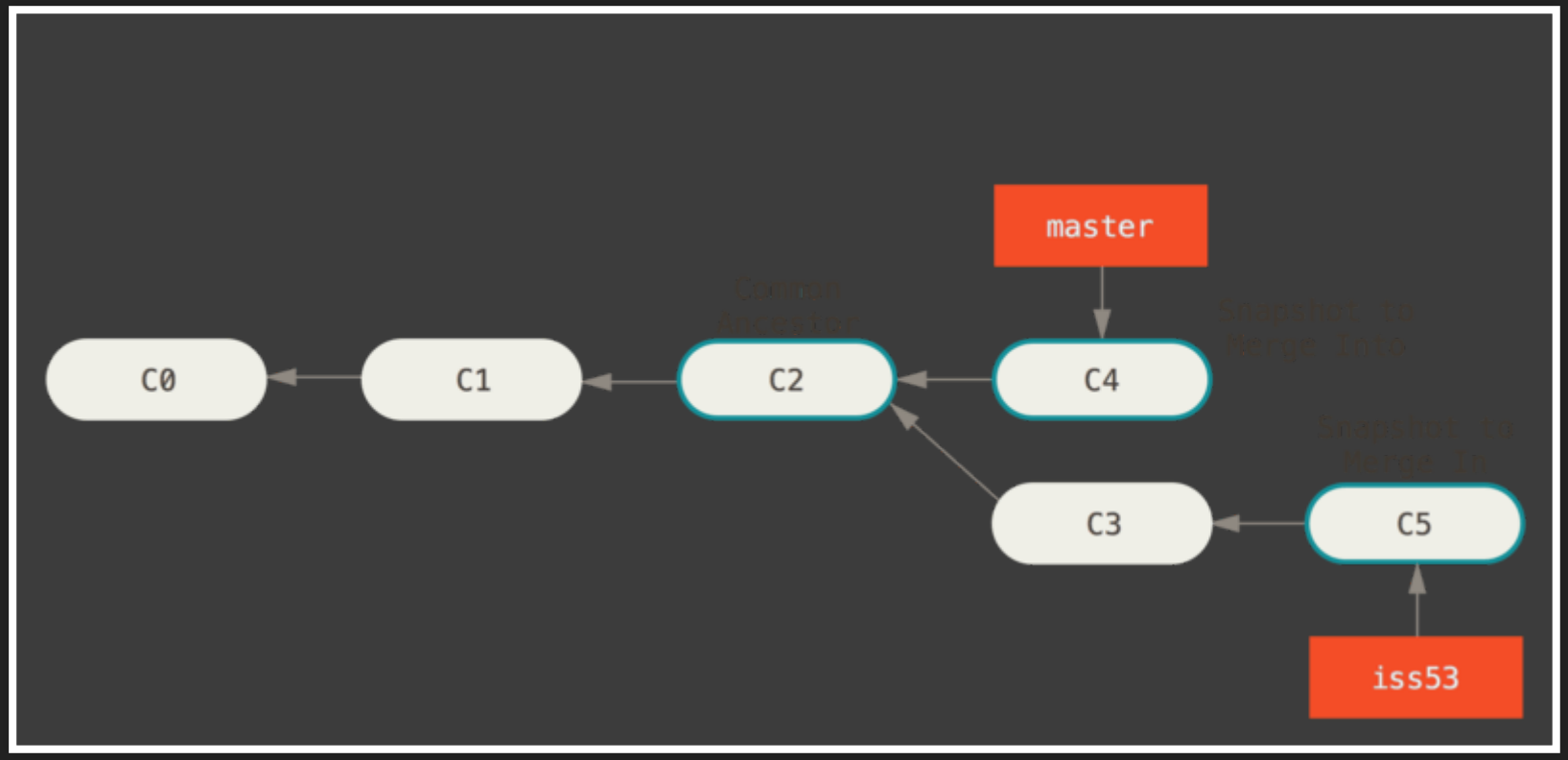


CRIANDO UMA DIVERGÊNCIA ENTRE AS BRANCH

Cenário:

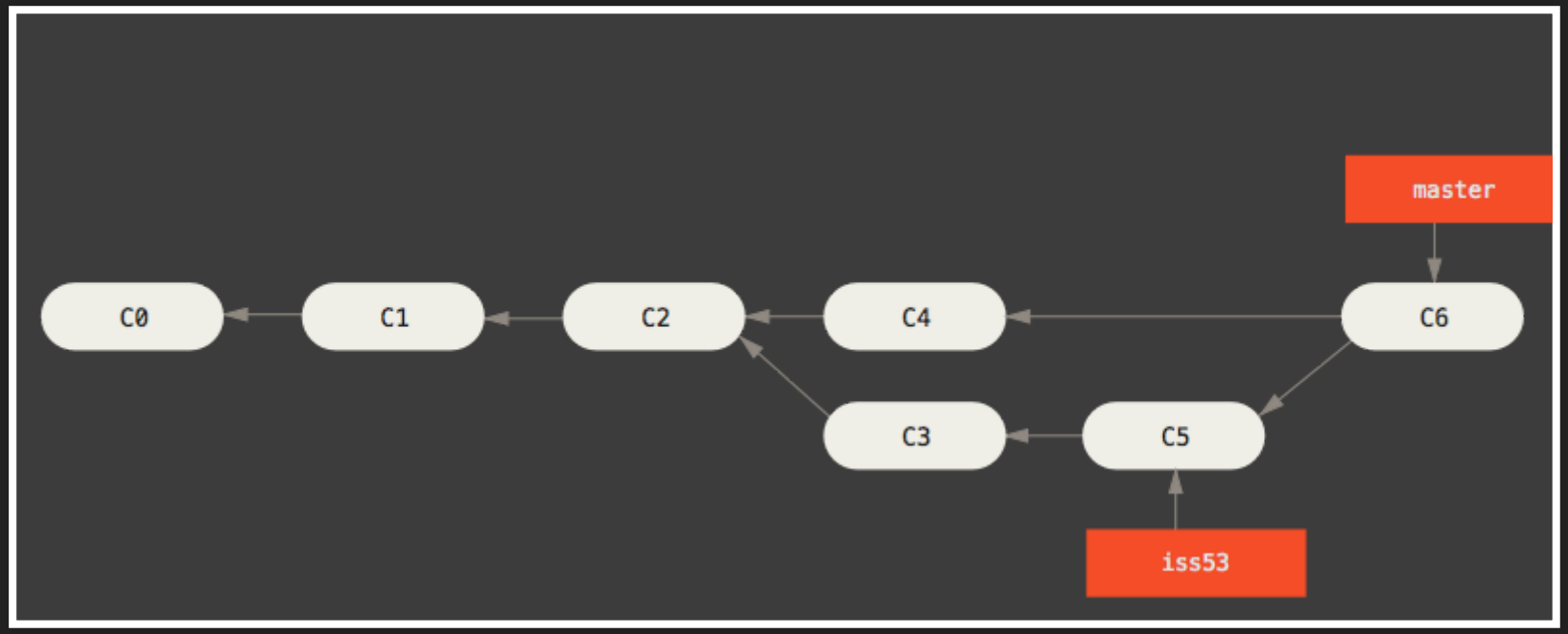
```
touch arquivo3
git add .
git commit -m" branch master"
git checkout testing
```

```
touch arquivo2
git add .
git commit -m" branch testing"
git checkout master
```


master
C0
C1
C2
C4
C3
C5
iss53

Fazendo o merge

```
git merge testing
```


master
C0
C1
C2
C4
C6
C3
C5
iss53

GERANDO UM CONFLITO

Cenário:

```
vim arquivo1
git add .
git commit -m" branch master"
git checkout testing
```

```
vim arquivo1
git add .
git commit -m" branch testing"
git checkout master
```

RESOLVENDO UM CONFLITO

```
git status
```

RESOLVENDO UM CONFLITO

```
<<<<<<< HEAD
<h1>titulo</h1>
=======
<h1>outro titulo</h1>
>>>>>>> testing
```

```
git add .
```

```
git merge --continue
```

TRABALHANDO COM REMOTOS

EXIBINDO OS REPOSITÓRIOS REMOTOS

```
git remote
```

```
git remote -v
```

ADICIONANDO UM REPOSITÓRIO REMOTO

Sintaxe:

```
git remote add [nome] [url]
```

Exemplo:

```
git remote add site git://github.com/allythy/Minicurso.git
```

ENVIANDO ARQUIVOS PARA O REPOSITÓRIO REMOTO

Sintaxe:

```
git push [nome-remoto] [branch]
```

Exemplo:

```
git push origin master
```

BAIXANDO ARQUIVOS DO REPOSITÓRIO REMOTO

```
git pull
```

ou

Sintaxe:

```
git fetch [nome-remoto]
```

Exemplo:

```
git fetch origin
```

```
git merge origin/master
```

RENOMEANDO REMOTOS

Sintaxe:

```
git remote rename [nome-atual] [novo-nome]
```

Exemplo:

```
git remote rename site blog
```

APAGANDO O REPOSITÓRIO REMOTO:

Sintaxe:

```
git remote rm [name]
```

Exemplo:

```
git remote rm site
```

CRIANDO UMA CONTA NO GITHUB

Acessem:

https://github.com

CONTRIBUINDO COM UM PROJETO

Acessem:

https://github.com/allythy/Minicurso-de-git-e-Github


OBRIGADO

CONTATOS

Telegram: https://t.me/allythy

GitHub: https://github.com/allythy

Site: https://allythy.github.io/

E-mail: allythy@protonmail.com