



UNIVERSIDADE
FEDERAL DE
MATO GROSSO DO SUL



Gerência de Configuração de Software

Aula 05 – Git - local

Prof. Dr. Awdren de Lima Fontão
awdren.fontao@ufms.br





- Uma ferramenta de controle de versão baseada em velocidade e integridade dos dados
- Pode-se usar git para hospedar seus projetos em repositórios no GitHub, GitLab, BitBucket, Codeplane.



Downloads



macOS



Windows



Linux/Unix

Older releases are available and the [Git source repository](#) is on GitHub.



GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).

Conceitos básicos: help!

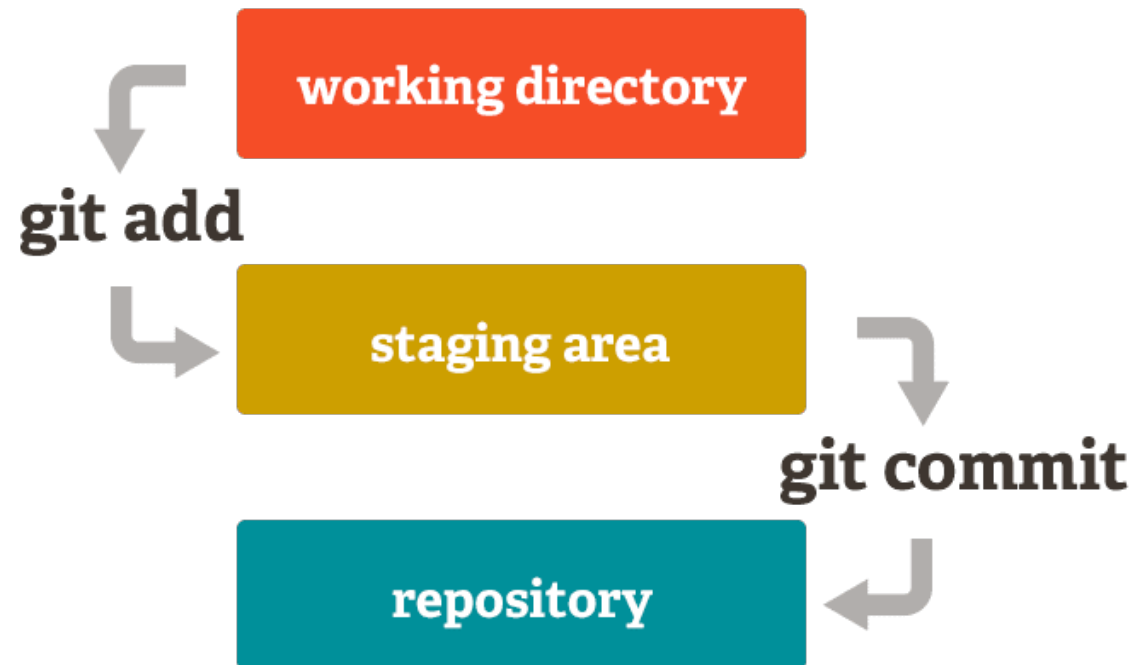
- git help
 - Oferece ajuda geral sobre o git
- Git help <commando>
 - Oferece ajuda sobre um comando específico do git
- Demais comandos dão dicas do que pode ser feito
 - Sempre leia com atenção ;)

Conceitos básicos: quem é você?

- `git config -global user.name <seu nome>`
 - Configura o nome de usuário
- `git config -global user.email <seu email>`
 - Configura o email do usuário

Conceitos básicos: STAGING AREA!

- É a área onde são colocados os arquivos que pretendemos enviar para o repositório



Conceitos básicos: commit id

- Cada sistema de controle de versão usa uma estratégia diferente para identificar commits
 - Número sequencial por arquivo (CVS)
 - Número sequencial por repositórios (subversion)
 - Hash (Git e Mercurial)

Um commit em um repositório git registra uma fotografia (snapshot) de todos os arquivos no seu diretório.
É a realização de um conjunto de mudanças provisórias permanentes, marcando o fim de uma transação

Commit semântico

fix: correct minor typos in code
see the issue for details
on typos fixed.
Reviewed-by: Elisandro Mello
Refs #133

1. **build:** Alterações que afetam o sistema de construção ou dependências externas (escopos de exemplo: gulp, broccoli, npm),
2. **ci:** Changes to our CI configuration files and scripts (example scopes: Travis, Circle, BrowserStack, SauceLabs);
3. **docs:** referem-se a inclusão ou alteração somente de arquivos de documentação;
4. **feat:** Tratam adições de novas funcionalidades ou de quaisquer outras novas implantações ao código;
5. **fix:** Essencialmente definem o tratamento de correções de bugs;
6. **perf:** Uma alteração de código que melhora o desempenho;

Commit semântico

```
fix: correct minor typos in code  
see the issue for details  
on typos fixed.  
Reviewed-by: Elisandro Mello  
Refs #133
```

- 7. refactor:** Tipo utilizado em quaisquer mudanças que sejam executados no código, porém não alterem a funcionalidade final da tarefa impactada;
- 8. style:** Alterações referentes a formatações na apresentação do código que não afetam o significado do código, como por exemplo: espaço em branco, formatação, ponto e vírgula ausente etc.);
- 9. test:** Adicionando testes ausentes ou corrigindo testes existentes nos processos de testes automatizados (TDD);
- 10. chore:** Atualização de tarefas que não ocasionam alteração no código de produção, mas mudanças de ferramentas, mudanças de configuração e bibliotecas que realmente não entram em produção;
- 11. env:** basicamente utilizado na descrição de modificações ou adições em arquivos de configuração em processos e métodos de integração contínua (CI), como parâmetros em arquivos de configuração de containers.



Conceitos básicos: apelidos

- A versão base do seu espaço de trabalho
 - HEAD
- O ramo principal do seu repositório
 - master ou main
- O repositório do qual seu repositório foi clonado
 - origin

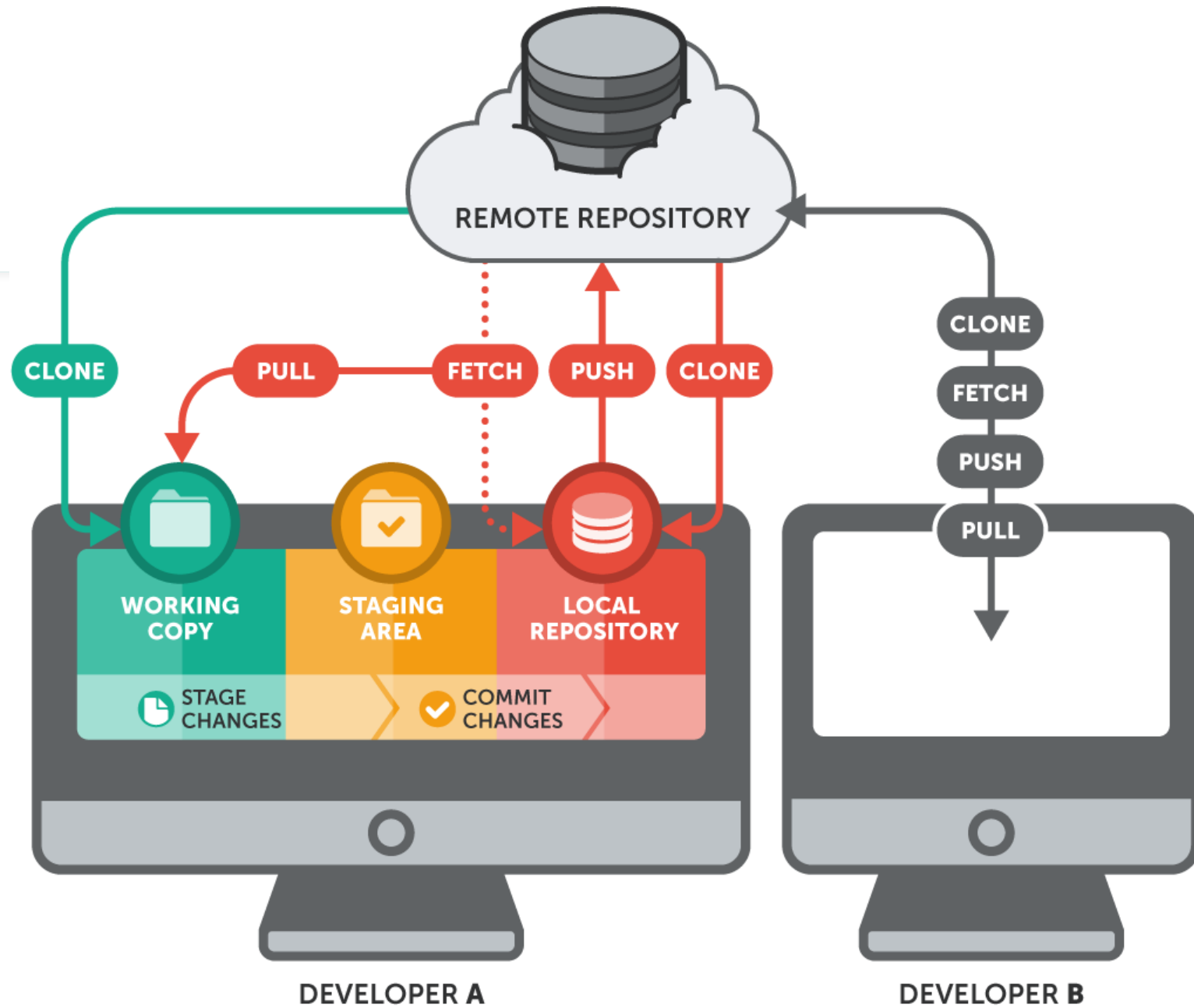
Repositório local

- `git init <nome>`
 - Cria um repositório Git no diretório
- `git add`
 - Adiciona um arquivo na *staging area* para ser enviado ao repositório no próximo *commit*
- `git commit -m <mensagem>`
 - Envia os arquivos que estão na *staging area* para o repositório

`close, closes, closed, fixes, fixed`

Repositório remoto

- Git clone <url> <diretório>
 - Cria um repositório local copiando o histórico de um repositório remoto
- Git pull
 - Atualiza o repositório local e o espaço de trabalho em relação a um repositório remoto
- Git push
 - Atualiza o repositório remoto em relação ao repositório local



Inspecionando mudanças

- Git status
 - Inspeciona o espaço de trabalho
- Git log [--graph] [--decorate=short] [--name-status]
 - Inspeciona o histórico do repositório local
- Git show
 - Inspeciona um commit
- Git diff
 - Compara o espaço de trabalho com a *staging area* ou com alguma versão do repositório

Demarcando versões especiais


- Git tag
 - Lista os rótulos existentes
- Git tag <nome do rótulo> [commit id]
 - Cria um rótulo sobre um dado commit (HEAD por default)
- Git tag -d <nome do rótulo>
 - Remove um rótulo

Repositório local com ramos


- `git branch -all -v`
 - Lista os ramos existentes no repositório
- `git branch <nome do ramo>`
 - Cria um ramo à partir da versão indicada no HEAD
- `git branch -d <nome do ramo>`
 - Remove um ramo
- `git checkout <commit id ou nome do ramo>`
 - Troca a versão base do espaço de trabalho
- `git merge <nome do ramo>`
 - Combina um ramo com o ramo corrente




Como você trabalha com git?




```
git add .  
git commit -m  
git pull  
git push
```



```
git add .  
git commit -m  
git pull  
git push
```



```
git add .  
git commit -m  
git pull  
git push
```



```
git add .  
git commit -m  
git pull  
git push
```

**O que realmente
estamos fazendo?**



```
git add .
```

```
git commit -m "Msg do commit"
```

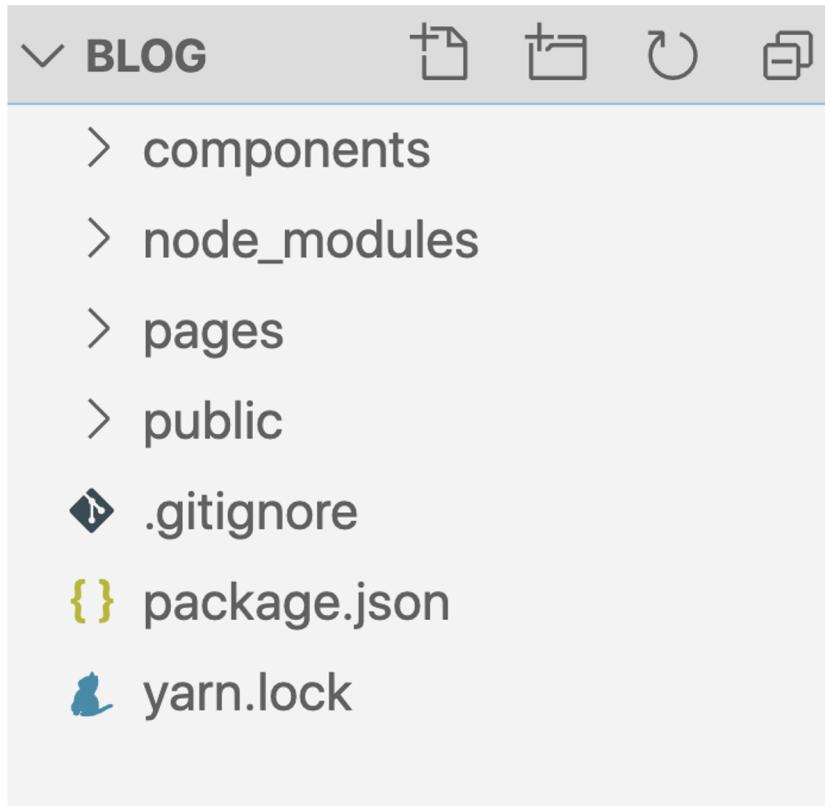
```
git pull
```

```
git push
```

"Estamos fechando um pacote de alterações"



WORKING DIRECTORY



→ **blog git:(master)** x git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore
components/
package.json
pages/
public/
yarn.lock

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

→ **blog git:(master)** x █



--all

-A

./dir/*

.

→ **blog git:(master)** x git status

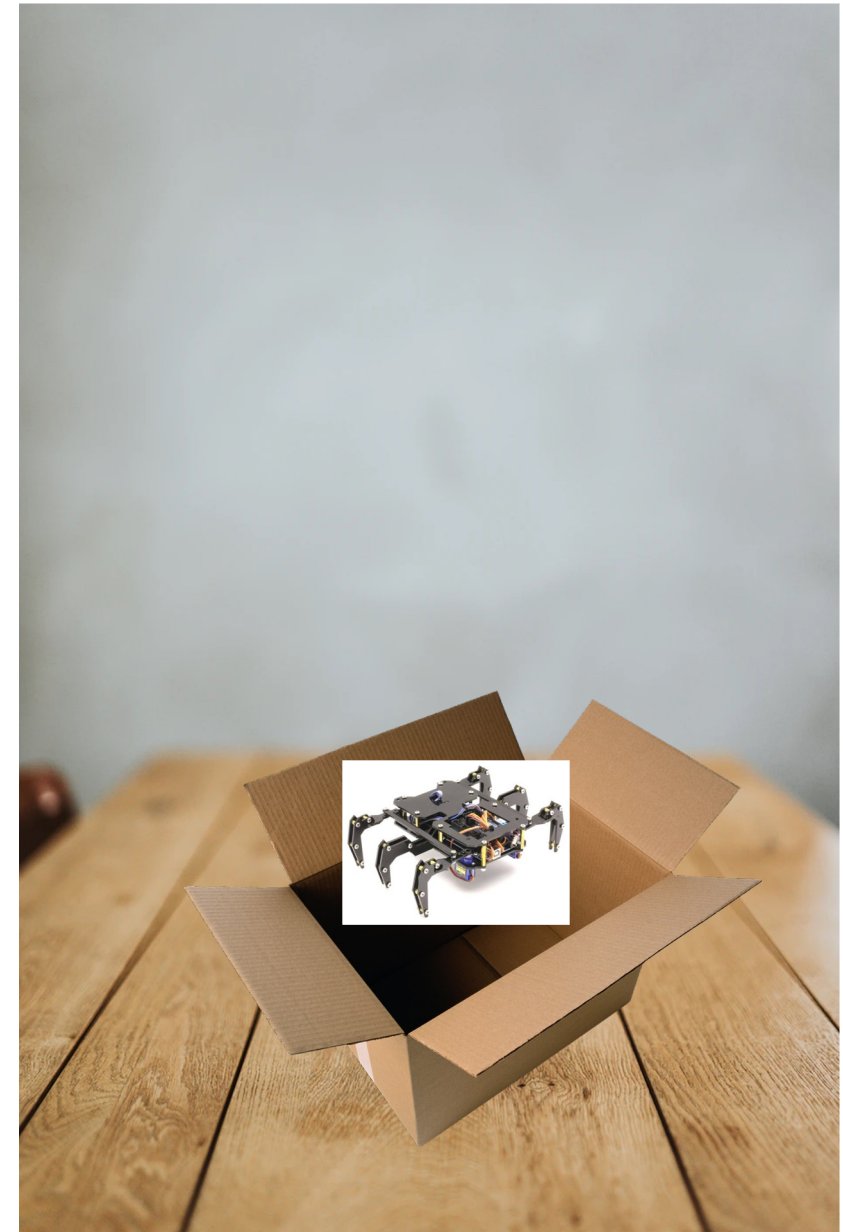
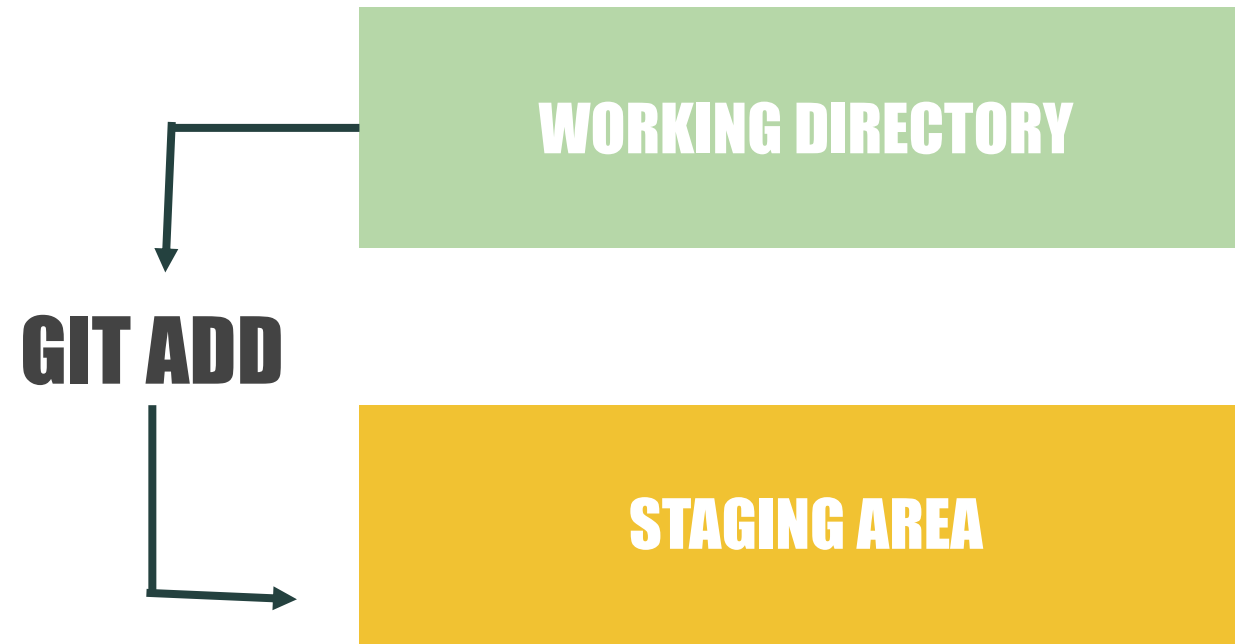
On branch master

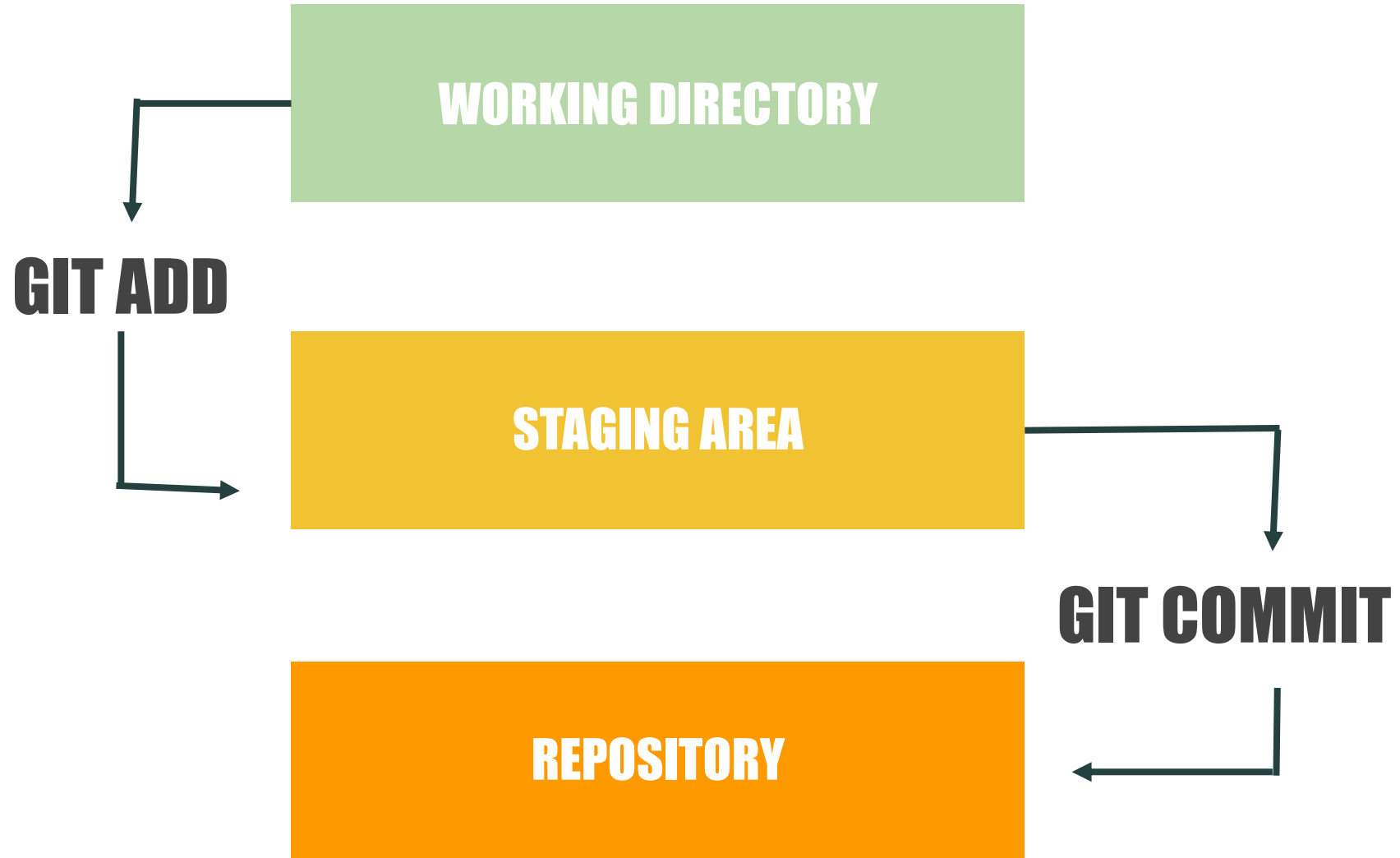
No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)


```
new file:   .gitignore
new file:   components/nav.js
new file:   package.json
new file:   pages/index.js
new file:   public/favicon.ico
new file:   yarn.lock
```





"Estamos fechando um pacote de alterações"

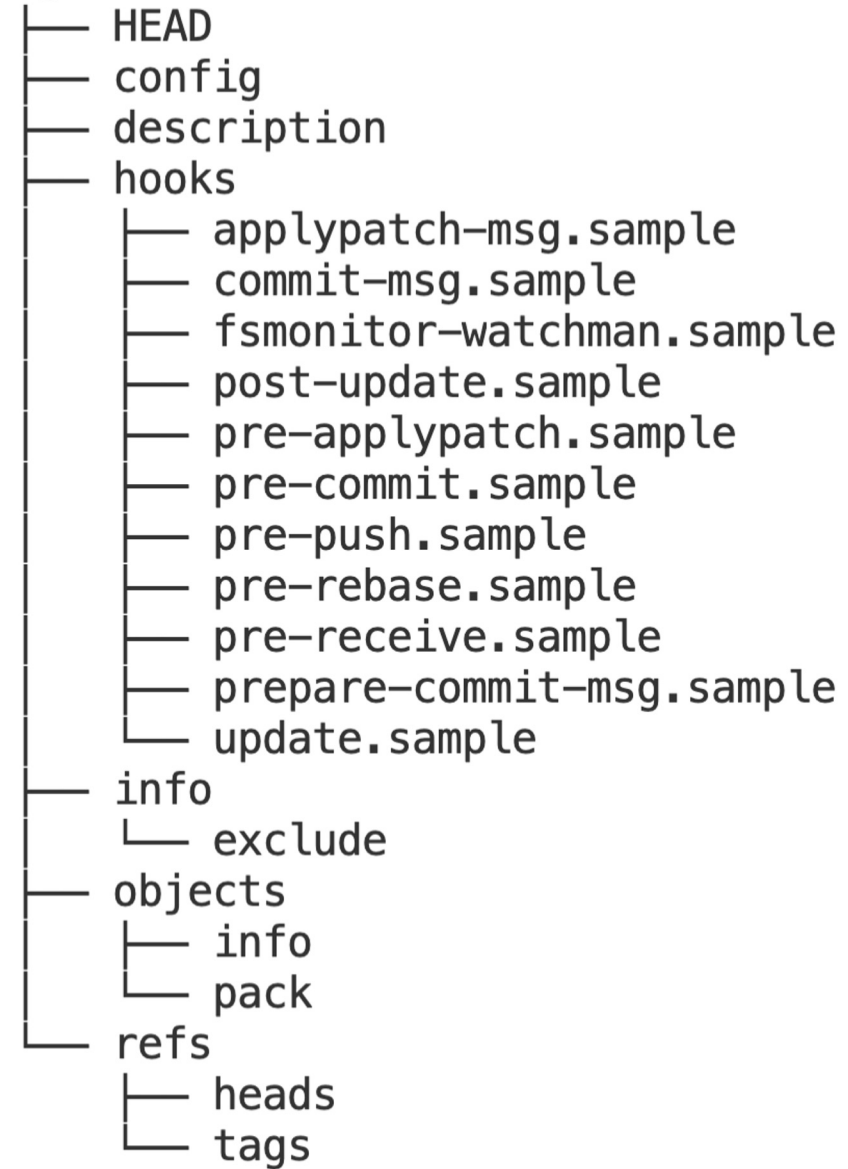




```
git add .  
git commit -m  
git pull  
git push
```

Como o git guarda isso?

.git



→ **artigos** **git:(master)** x git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: artigo-1.md

new file: artigo-2.md

tree .git

```
→ artigos git:(master) x tree .git
.git
├── HEAD
├── config
├── description
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── fsmonitor-watchman.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   ├── pre-receive.sample
│   ├── prepare-commit-msg.sample
│   └── update.sample
├── index
├── info
│   └── exclude
├── objects
│   ├── b7
│   │   └── 4edb1bd1a3c1be3814ac2bd11452102058cce7
│   ├── b9
│   │   └── 2c360502530526cae64d4869ff8c80a2b17c48
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags
```

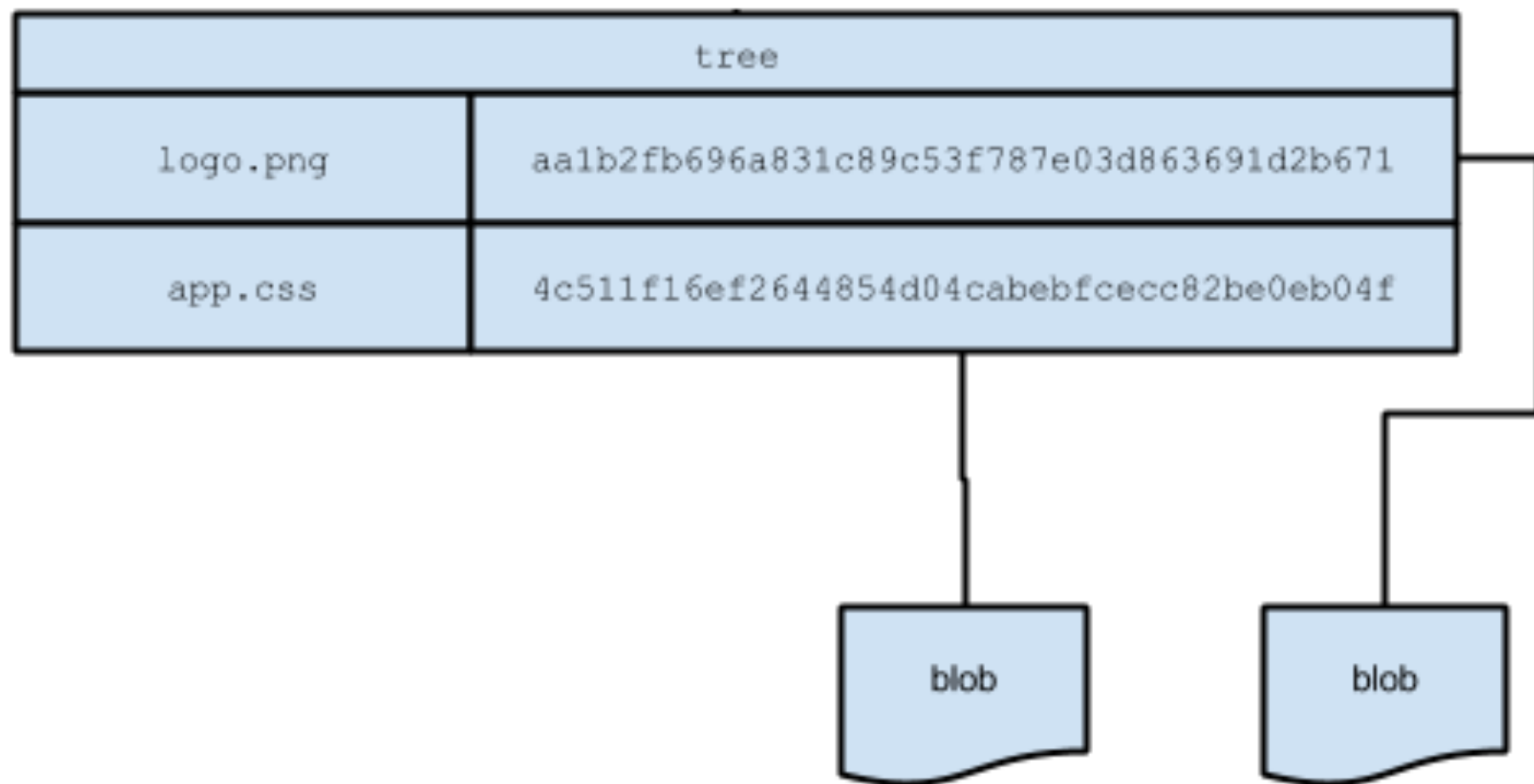
Blobs

tree .git

Um blob Git (objeto binário grande) é o tipo de objeto usado para armazenar o conteúdo de cada arquivo em um repositório.

O hash SHA-1 do arquivo é calculado e armazenado no objeto blob.

```
→ artigos git:(master) x tree .git
.git
├── HEAD
├── config
├── description
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── fsmonitor-watchman.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   ├── pre-receive.sample
│   ├── prepare-commit-msg.sample
│   └── update.sample
├── index
├── info
├── exclude
├── objects
│   ├── b7
│   │   └── 4edb1bd1a3c1be3814ac2bd11452102058cce7
│   ├── b9
│   │   └── 2c360502530526cae64d4869ff8c80a2b17c48
│   ├── info
│   └── pack
├── refs
│   ├── heads
│   └── tags
```



Blobs

```
blob b92c6  
## Titulo 02
```

```
blob b74ed  
## Titulo 01
```

→ **artigos** **git:(master)** x git commit -m "Add artigos"
[master (root-commit) 52febca] Add artigos
2 files changed, 2 insertions(+)
create mode 100644 artigo-1.md
create mode 100644 artigo-2.md

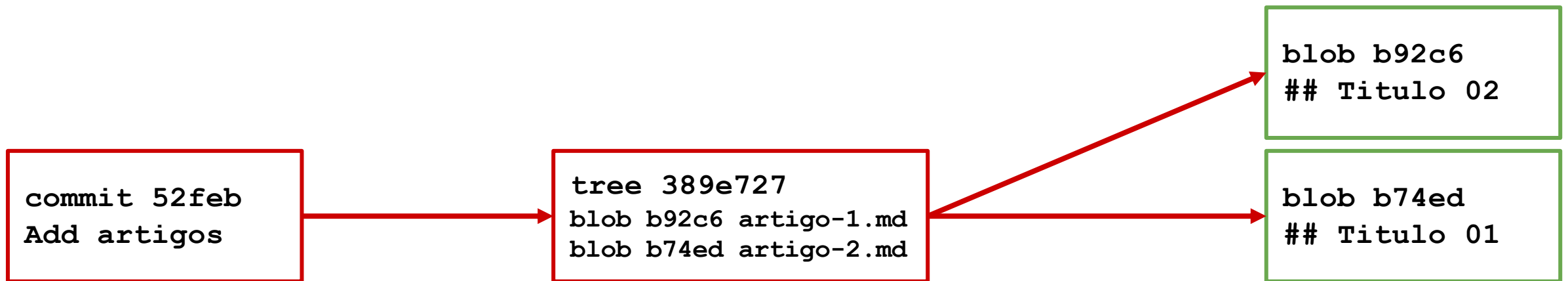
Commit

```
commit 52feb  
Add artigos
```

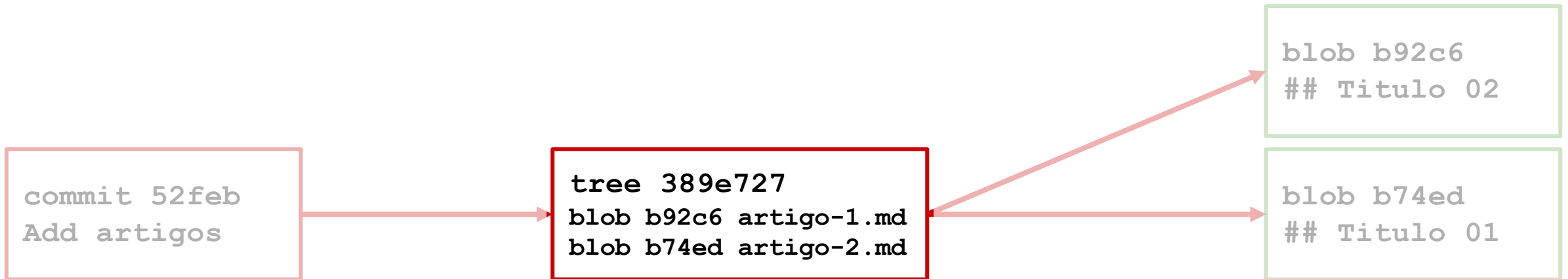
```
blob b92c6  
## Titulo 02
```

```
blob b74ed  
## Titulo 01
```

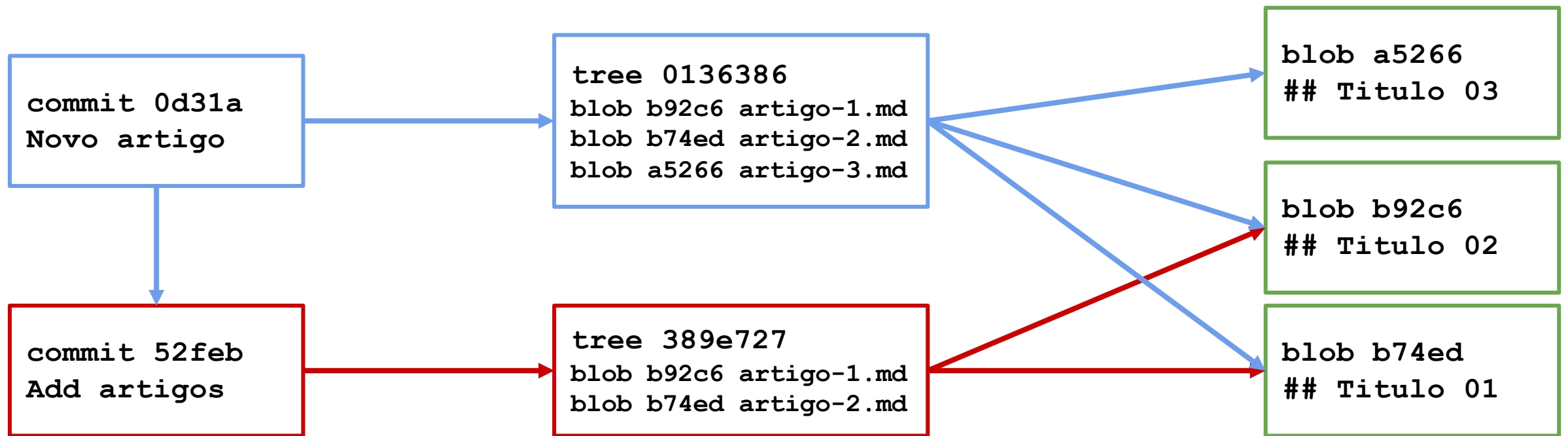
Commit

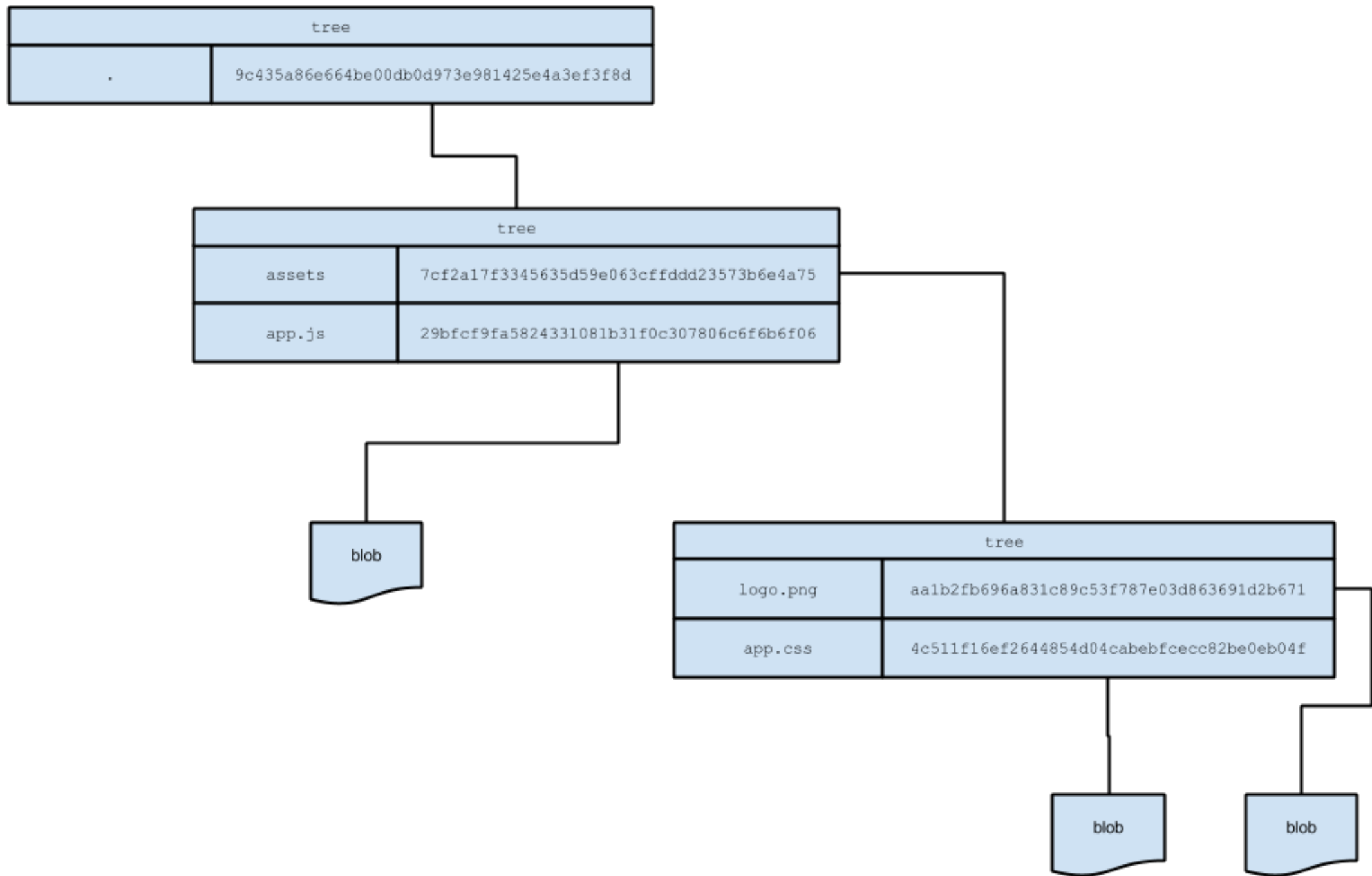


Tree = Snapshot/Foto do estado atual do projeto



```
→ artigos git:(master) x git commit -m "Novo artigo"
[master 0d31af2] Novo artigo
1 file changed, 1 insertion(+)
create mode 100644 artigo-3.md
→ artigos git:(master) █
```

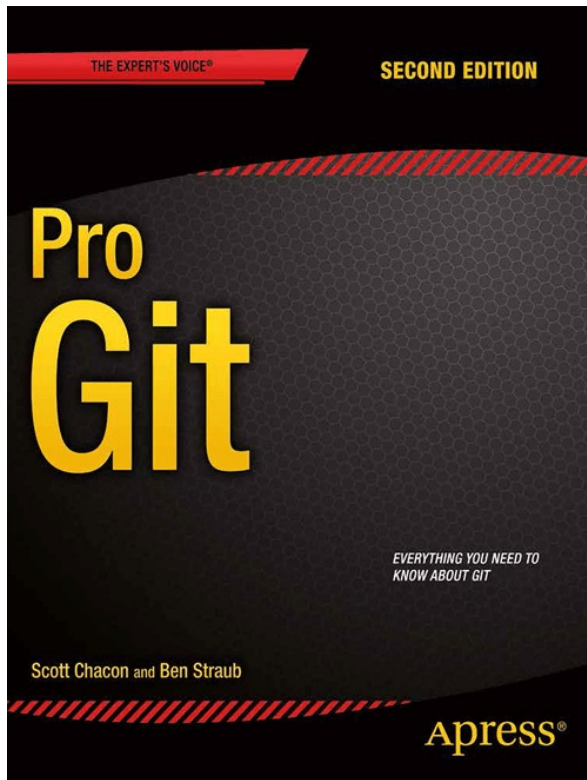




Exercício (Entrega das evidências pelo AVA)

- https://learngitbranching.js.org/?locale=pt_BR
- <http://git-school.github.io/visualizing-git/>

Referência



- <https://git-scm.com/book/en/v2>