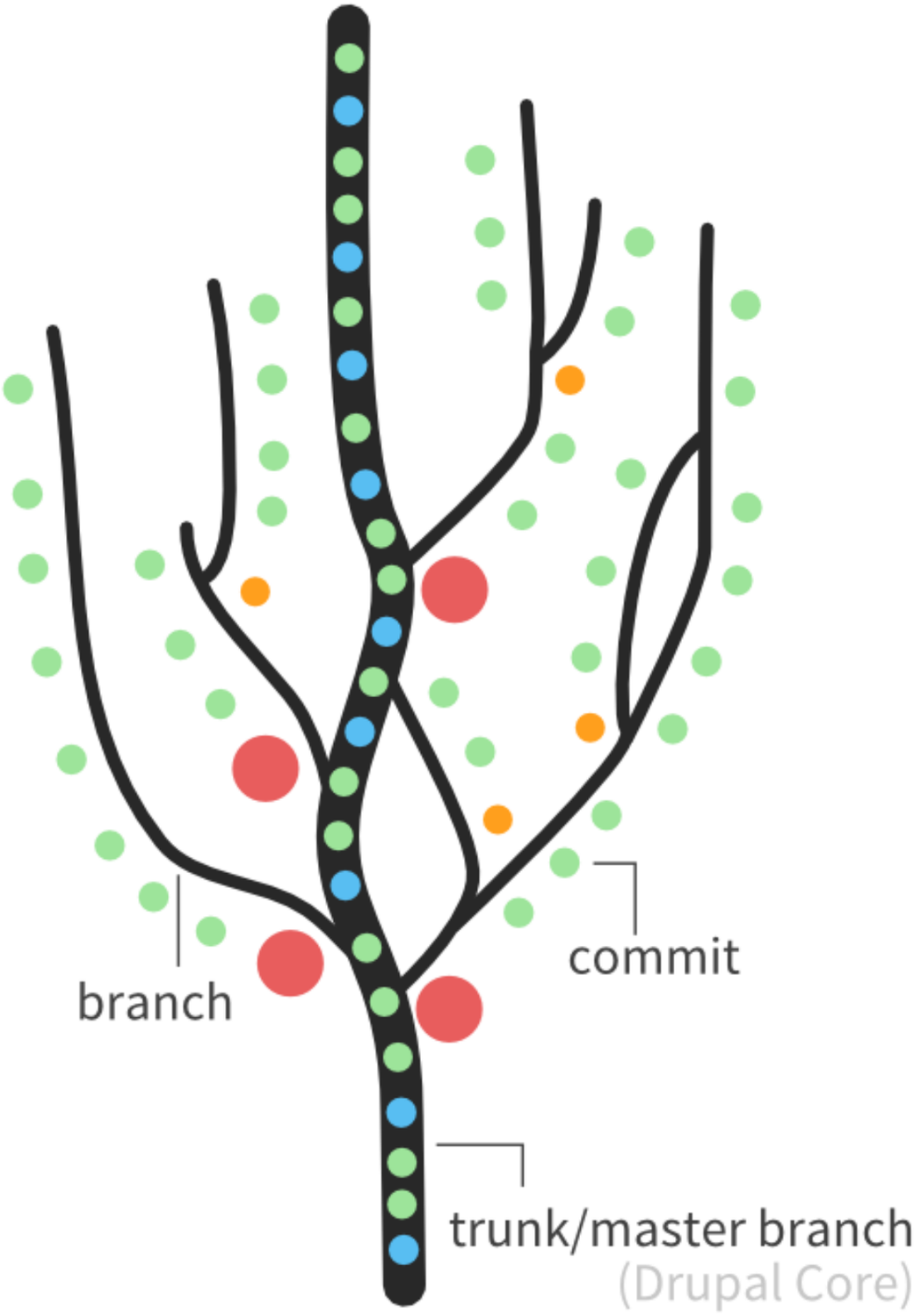
Faculdade IMPACTA TECNOLOGIA

Comandos essenciais no *bash*

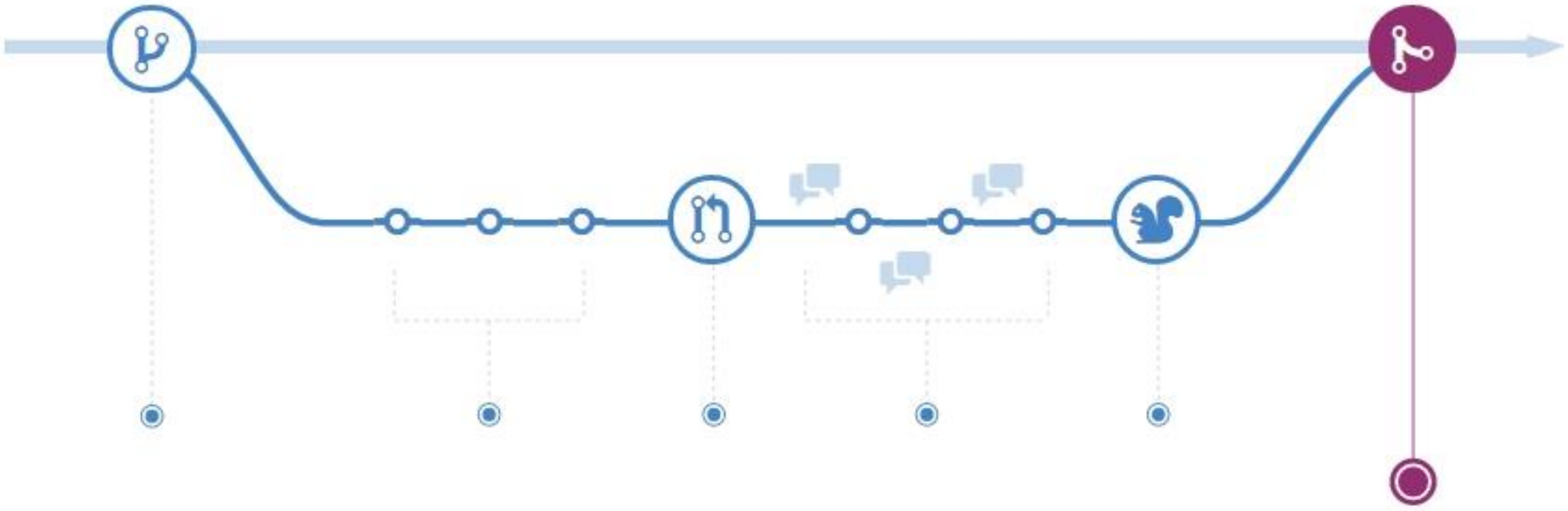
Aula 06 - Parte 1
Disciplina: Ambiente de Desenvolvimento e Operação
Curso: Sistemas da Informação
Turma: 2A
Prof. Edson Benites Silva
2º semestre de 2017

Árvore de versões

Fonte: http://www.drupal.org/node/991716

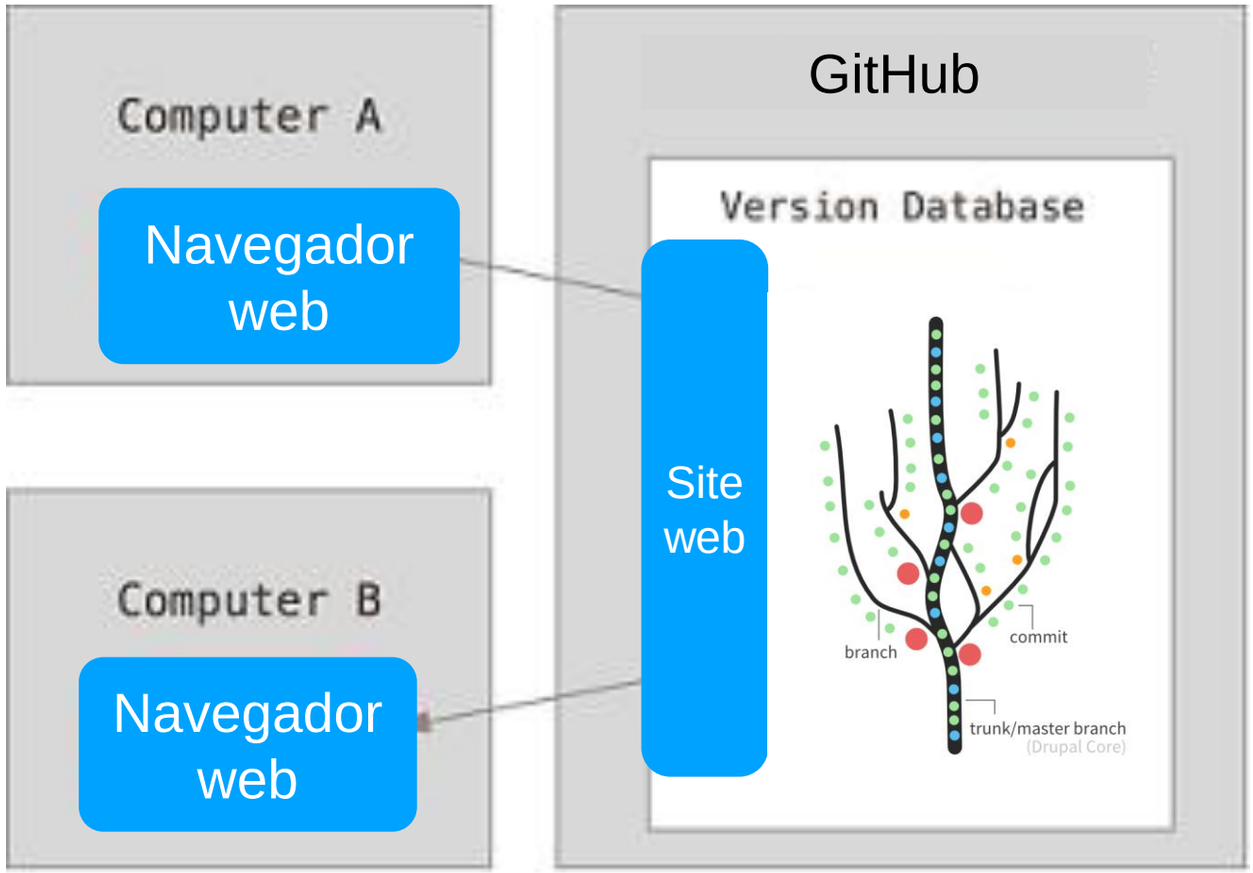


Workflow Github

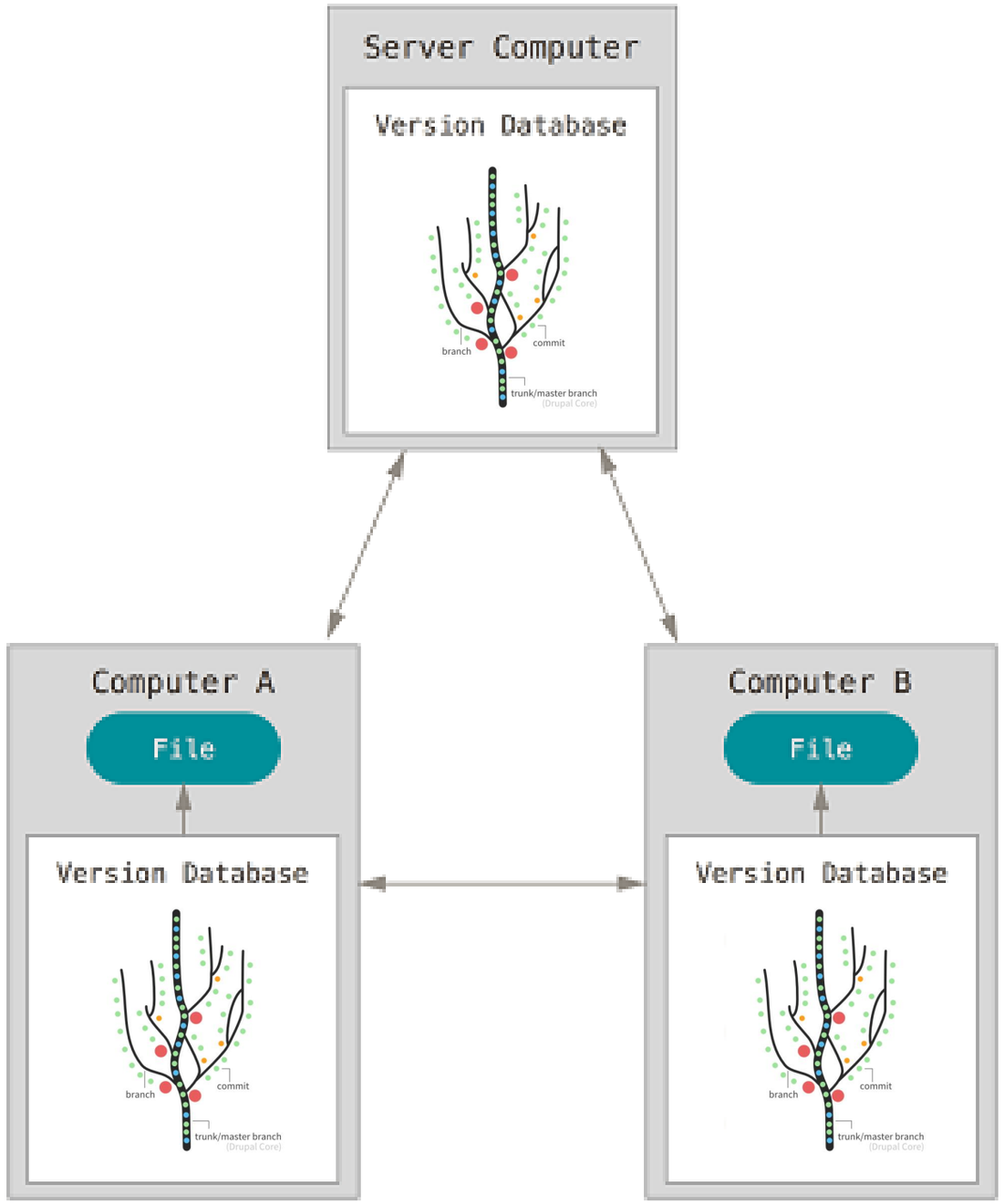


Fonte: https://guides.github.com/introduction/flow

GitHub pela Web

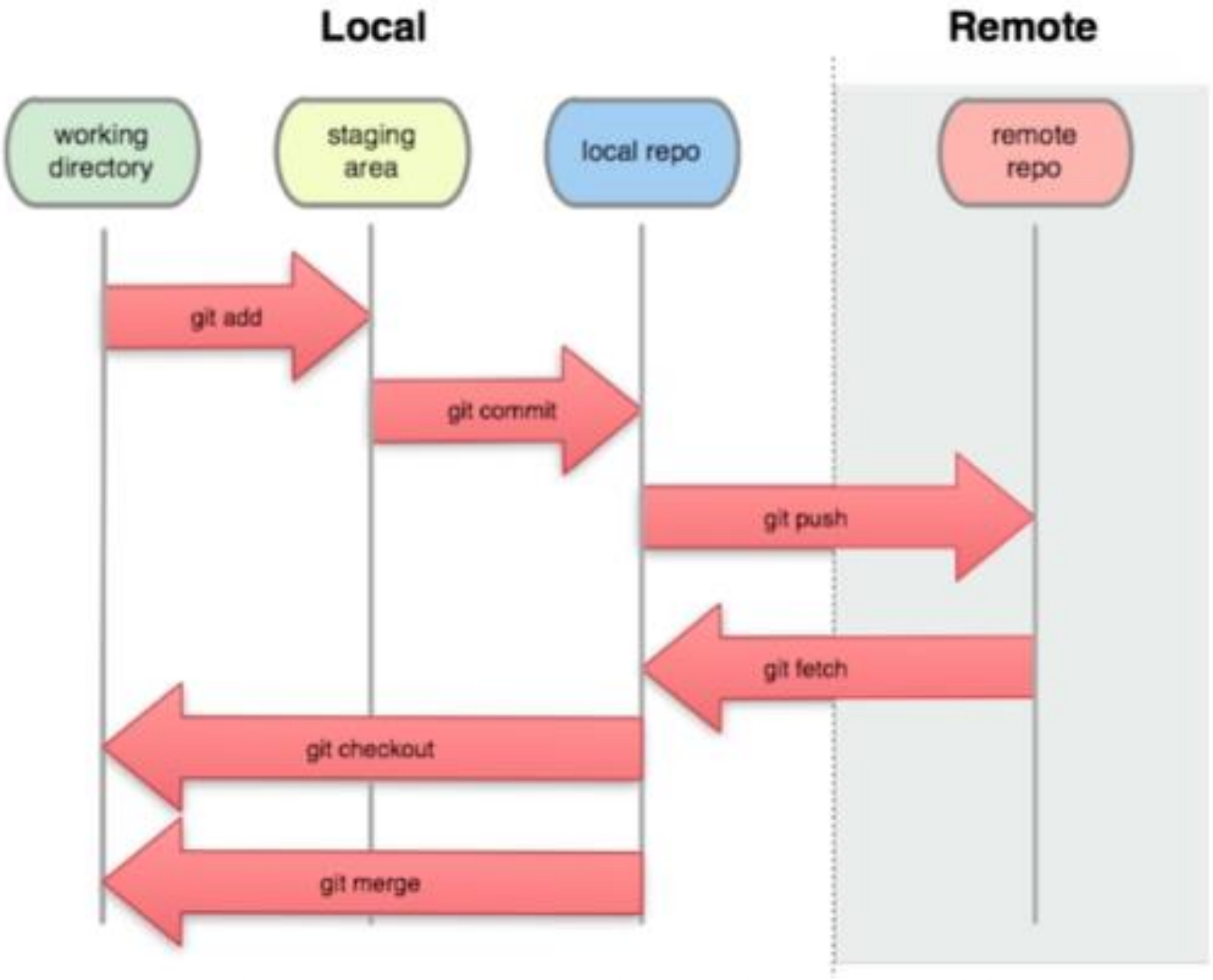


Git Distruido



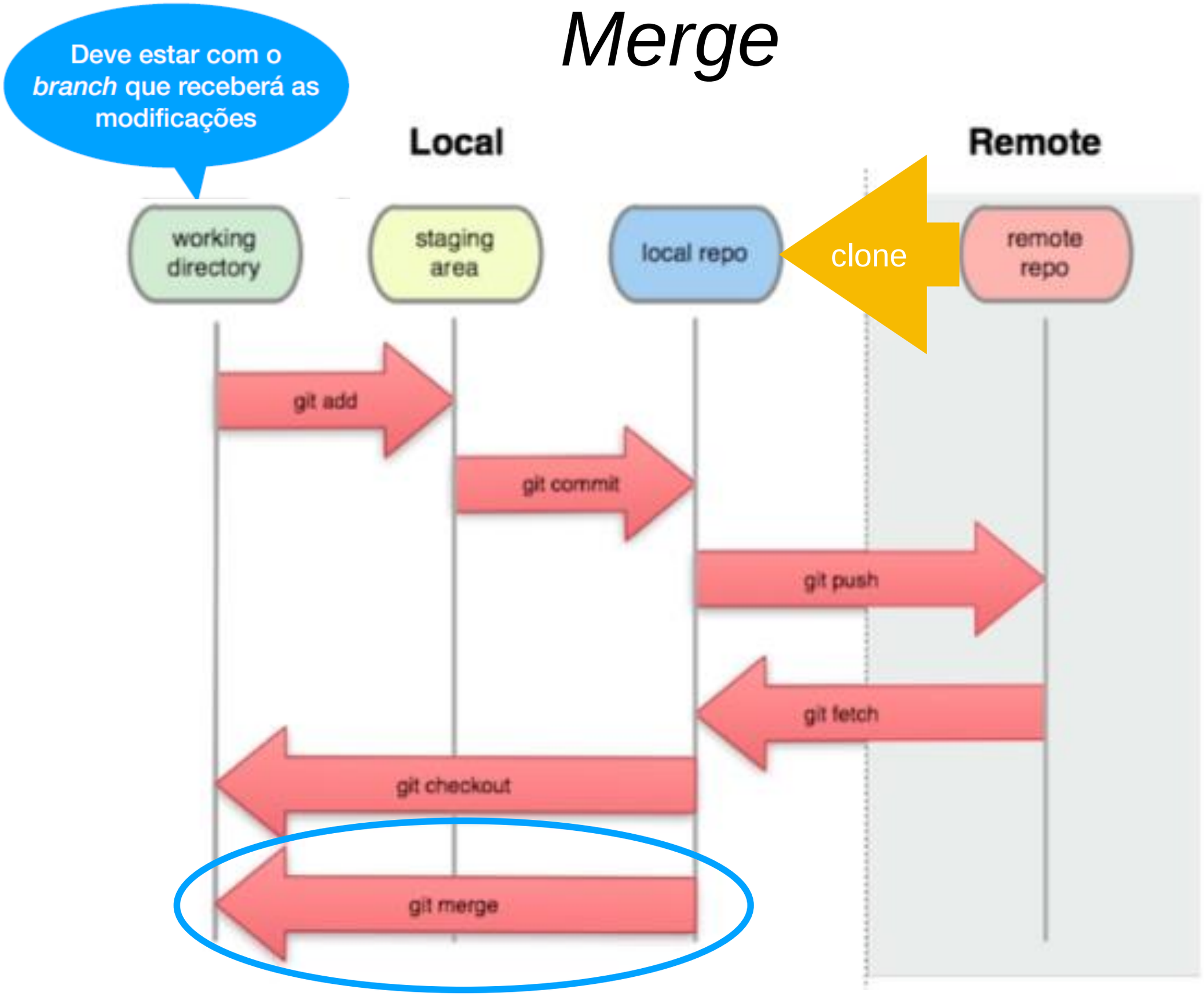
Fonte: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Operações nos repositórios Git



Fonte: https://greenido.wordpress.com/2013/07/22/git-101-useful-commands/

Merge


Deve estar com o *branch* que receberá as modificações
Local
Remote
working directory
staging area
local repo
clone
remote repo
git add
git commit
git push
git fetch
git checkout
git merge

Listagem do conteúdo do diretório

ls

Criação e remoção de diretórios

mkdir ***diretorio***

rmdir ***diretorio***

Mudança do diretório local

cd ***diretorio***

cd ..

cd ***diretorio1/diretorio2***

cd ***/***

cd ***/diretorio***

Criação de arquivo vazio

```
touch arquivo
```

Abrir arquivo com vi

```
vi arquivo
```

Sair do vi sem salvar

Executar a sequência abaixo sem errar (se errar, começar a sequência novamente desde o início):

- Pressionar a tecla ESC
- Digitar :q!
- Pressionar ENTER

Adicionar texto com vi

Executar a sequência abaixo sem errar (se errar, começar a sequência novamente desde o início):

- Pressionar a tecla ESC
- Levar o cursor até o ponto onde o texto será inserido.
- Pressionar a tecla **i**
- Digitar o texto.
- Pressionar a tecla ESC

Apagar texto com **vi**

Executar a sequência abaixo sem errar (se errar, começar a sequência novamente desde o início):

- Pressionar a tecla ESC
- Levar o cursor até o ponto onde o texto será apagado.
- Pressionar a tecla **del** ou a tecla **x** para apagar o caractere à direita do cursor.

Sair do vi salvando mudanças

Executar a sequência abaixo sem errar (se errar, começar a sequência novamente desde o início):

- Pressionar a tecla ESC
- Digitar :wq!
- Pressionar ENTER

Exercício
Passo 1

- Acesse pela web a sua conta no GitHub.

- Crie um novo repositório chamado **devops-aula06** e inicialize-o com um arquivo chamado **README.md**.

- Crie um novo *branch* com o nome **v1**.

Passo 2

- Faça o **clone** do repositório no seu computador local (consulte o material da aula 4 para se lembrar como executar a operação de **clone**).

- Faça o **checkout** do *branch* **v1** (consulte o material da aula 4 para se lembrar como executar a operação de **checkout**).

Passo 3

- (Faça o exercício usando o bash)
- No seu diretório de trabalho (que está com o *branch* **v1**), crie as pastas **docs** e **src**.
- Crie os arquivos:
    - docs/requisitos.md
    - docs/arquitetura.md
    - src/jogovelha.py
    - src/testes.py

Passo 4

- (Faça o exercício usando o bash)

- Edite o conteúdo do arquivo **docs/requisitos.md** para que fique com o conteúdo abaixo:

```
# Requisitos

* Gerar uma estrutura de dados para manter o estado de cada
uma das casas de um jogo da velha.

* Cada casa do jogo da velha poderá estar vazia, ocupada
pelo 1o jogador ou ocupada pelo 2o jogador.
```

Passo 5

- (Faça o exercício usando o bash)

- Edite o conteúdo do arquivo **docs/arquitetura.md** para que fique com o conteúdo abaixo:

```
# Arquitetura

* As funções relacionadas ao gerenciamento das casas do
jogo da velha ficarão no módulo **jogovelha.py**.

* O estado de cada casa do jogo será representada por uma
string: "." para casa vazia; "X" para casa ocupada pelo 1o
jogador; "O" para casa ocupada pelo 2o jogador

* A função inicializar() retornará uma lista 3x3, onde cada
posição conterá uma string para indicar o estado de uma
casa do jogo. A função retornará todas as casas
inicialmente vazias.
```

Passo 6

- (Faça o exercício usando o bash)

- Edite o conteúdo do arquivo **src/jogovelha.py** para que fique com o conteúdo abaixo:

```
def inicializar():
    tab = []
    for i in range(3):
        linha = []
        for j in range(3):
            linha.append(".")
        tab.append(linha)
    return tab

def main():
    jogo = inicializar()
    print(jogo)

if __name__ == "__main__":
    main()
```

Passo 7

- (Faça o exercício usando o bash)

- Edite o conteúdo do arquivo **src/testes.py** para que fique com o conteúdo abaixo:

```
import jogovelha
import sys

erroInicializar = False

jogo = jogovelha.inicializar()

if len(jogo) != 3:
    erroInicializar = True
else:
    for linha in jogo:
        if len(linha) != 3:
            erroInicializar = True
        else:
            for elemento in linha:
                if elemento != '.':
                    erroInicializar = True

if erroInicializar:
    sys.exit(1)
else:
    sys.exit(0)
```

Passo 8

- (Faça o exercício usando o bash)

- Adicione todos os arquivos ao ***staging area*** (consulte o materialda aula 4 para se lembrar como executar a operação de **add**).

- Faça o **commit** com o comentário “versão 1” (consulte o material da aula 4 para se lembrar como executar a operação de **commit**).

- Faça o **push** para o repositório remoto (consulte o material da aula 4 para se lembrar como executar a operação de **push**).

- Confira no site **GitHub** se as modificações foram enviadas para o*branch* **v1**.

Faculdade

IMPACTA

TECNOLOGIA