WEB11 (Git) —— Aula 2

CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO WEB COM *FRAMEWORKS* MODERNOS

UTFPR

PROF. RUI PIMENTEL LEITE

PRODUZINDO CÓDIGO VERSIONADO

- Na aula anterior, vimos:
  - `git clone` traz uma cópia de um repositório
  - `git log` exibe um histórico de *commits*
  - `git checkout` aplica uma versão
  - `gitk &` mostra o histórico de forma gráfica

PRODUZINDO CÓDIGO VERSIONADO

- Mas como criar um repositório?
- Como nasce um *commit*?
- O que era mesmo um *commit*?
  - Quais campos o compõem?
- O que torna um diretório um repositório?

PRODUZINDO CÓDIGO VERSIONADO

- Teste: o que acontece ao executar `git log`...
  - No diretório `flexbox-boilerplate/`?
  - Em um diretório vazio?
  - Em um diretório qualquer com arquivos?

PRODUZINDO CÓDIGO VERSIONADO

- O diretório oculto `.git/` é o que transforma um simples diretório em um repositório Git
  - Em `.git/` se encontram importantes metadados para catalogar as versões de nosso código
  - Por isso, ao destruí–lo, essencialmente achatamos nosso diretório a uma pasta comum, com os arquivos na versão de `checkout` atual

PRODUZINDO CÓDIGO VERSIONADO

- Inicialização de um repositório (criação do diretório `.git/`):
  - `mkdir novo-repositorio`
  - `cd novo-repositorio/`
  - `git init`

PRODUZINDO CÓDIGO VERSIONADO

- Criação do *commit* inicial:
  - `touch meu-arquivo-{1..3}`
  - `git add .`
  - `git commit -m ‘Commit inicial’`

PRODUZINDO CÓDIGO VERSIONADO

- Criação de mensagem de *commit* via editor:
  - `rm meu-arquivo-2`
  - `git add .`
  - `git commit`

PRODUZINDO CÓDIGO VERSIONADO

- Na prática, as mensagens de *commit* são frequentemente usadas da seguinte maneira:
  - A primeira linha consiste de um resumo; é quase sempre exibida junto ao *hash* SHA-1
  - A segunda linha fica em branco
  - As demais linhas descrevem o que foi feito e por qual razão

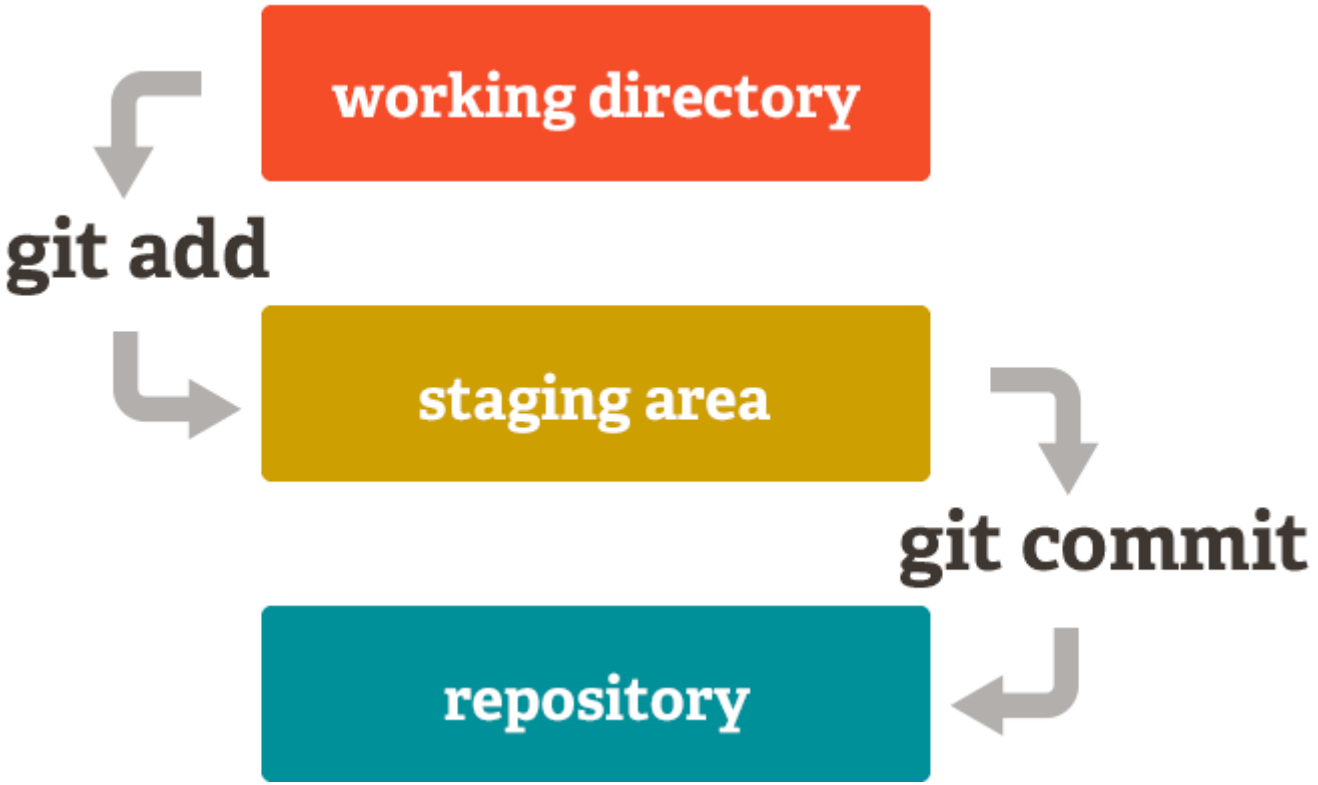
PRODUZINDO CÓDIGO VERSIONADO

- Alterando o editor de mensagem de *commit*:
  - `git config --global core.editor "vim"`

PRODUZINDO CÓDIGO VERSIONADO

- O que faz a etapa `git add .`?
  - `git add` adiciona alterações num **índice** (conhecido como *index* ou *staging area*)
  - Apenas o que estiver nesse **índice** será incluído no *commit*

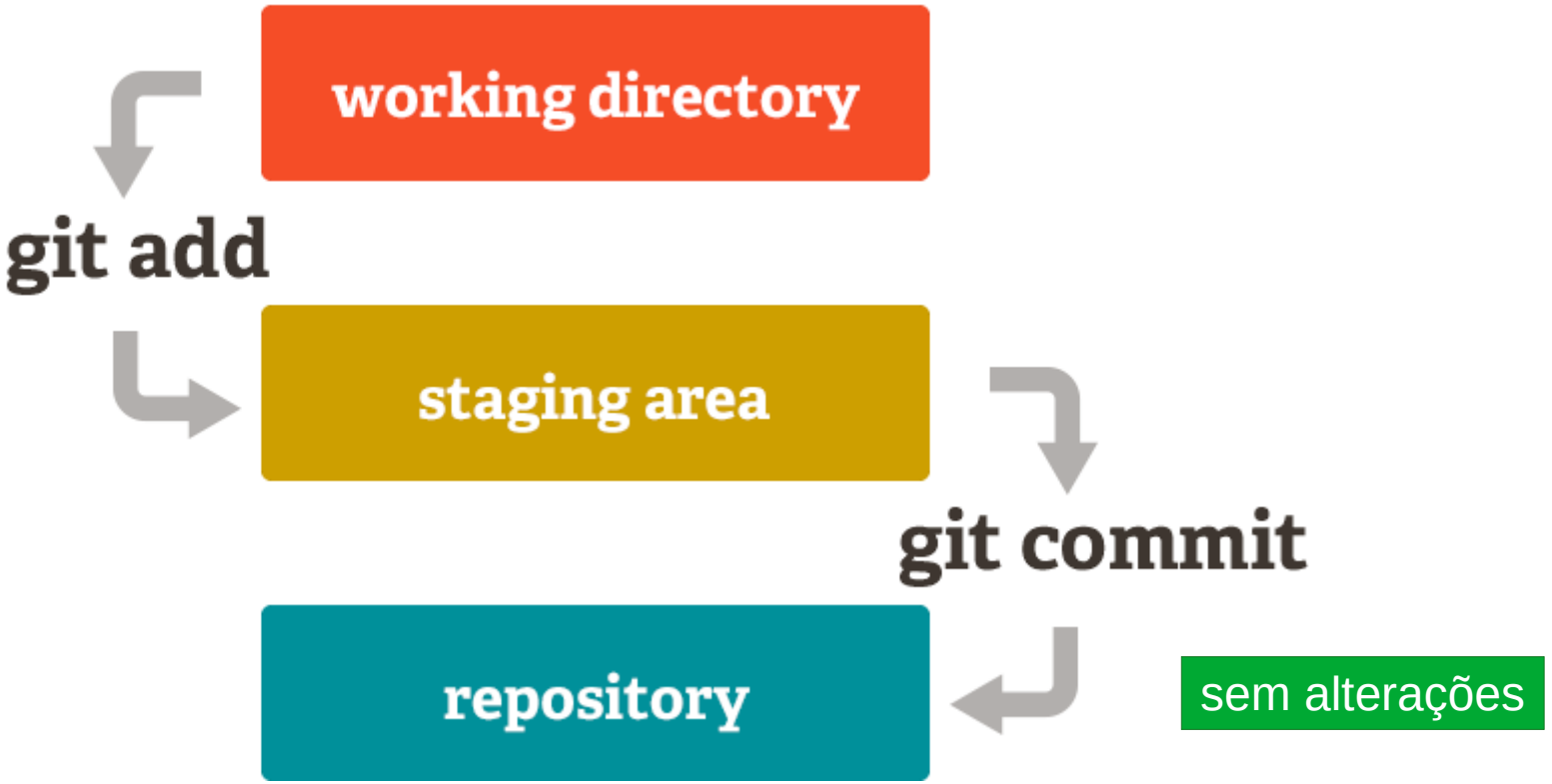
PRODUZINDO CÓDIGO VERSIONADO



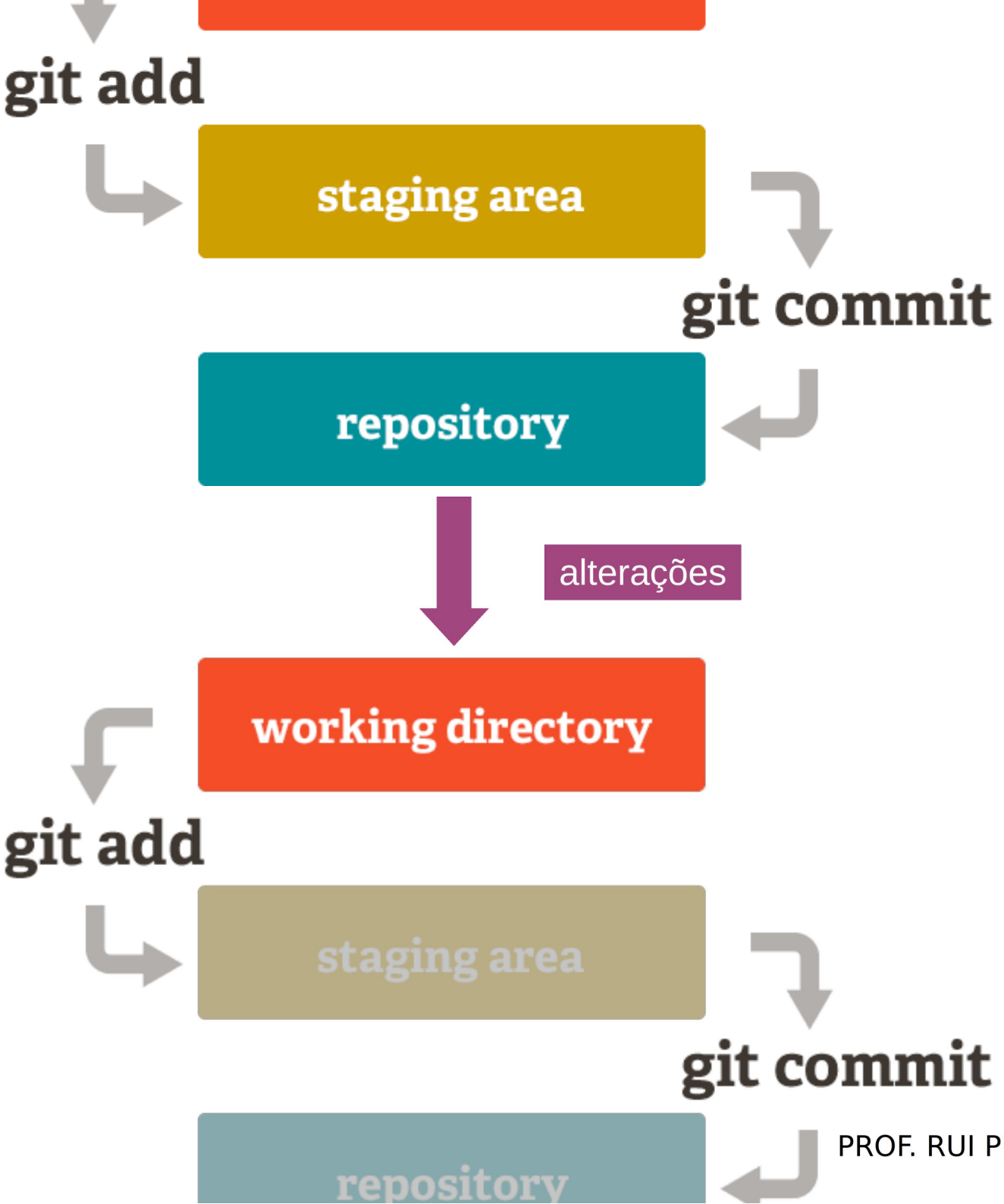
PRODUZINDO CÓDIGO VERSIONADO

- O comando `git status` exibe a situação atual do repositório

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$ git status
On branch master
nothing to commit, working tree clean
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$
```


working directory
git add
staging area
git commit
repository
sem alterações


working directory
git add
staging area
git commit


git add
staging area
git commit
repository
alterações
working directory
git add
staging area
git commit
repository

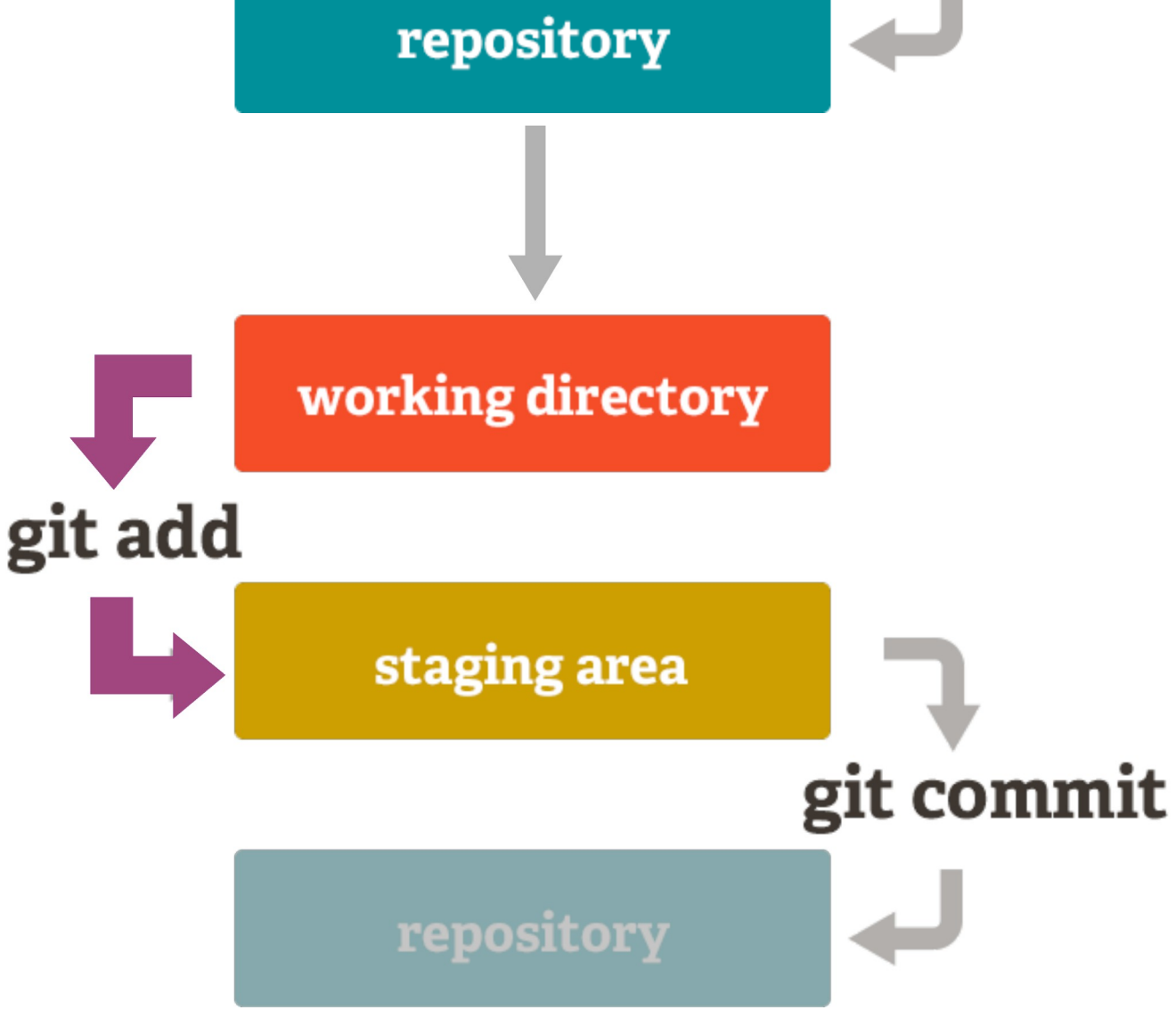
PRODUZINDO CÓDIGO VERSIONADO

- Exercício: altere o arquivo `meu-arquivo-1` com o `vim`, e então verifique o status do repositório:
  - `vim meu-arquivo-1`
  - `git status`

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   meu-arquivo-1

no changes added to commit (use "git add" and/or "git commit -a")
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$
```


repository
working directory
git add
staging area
git commit
repository

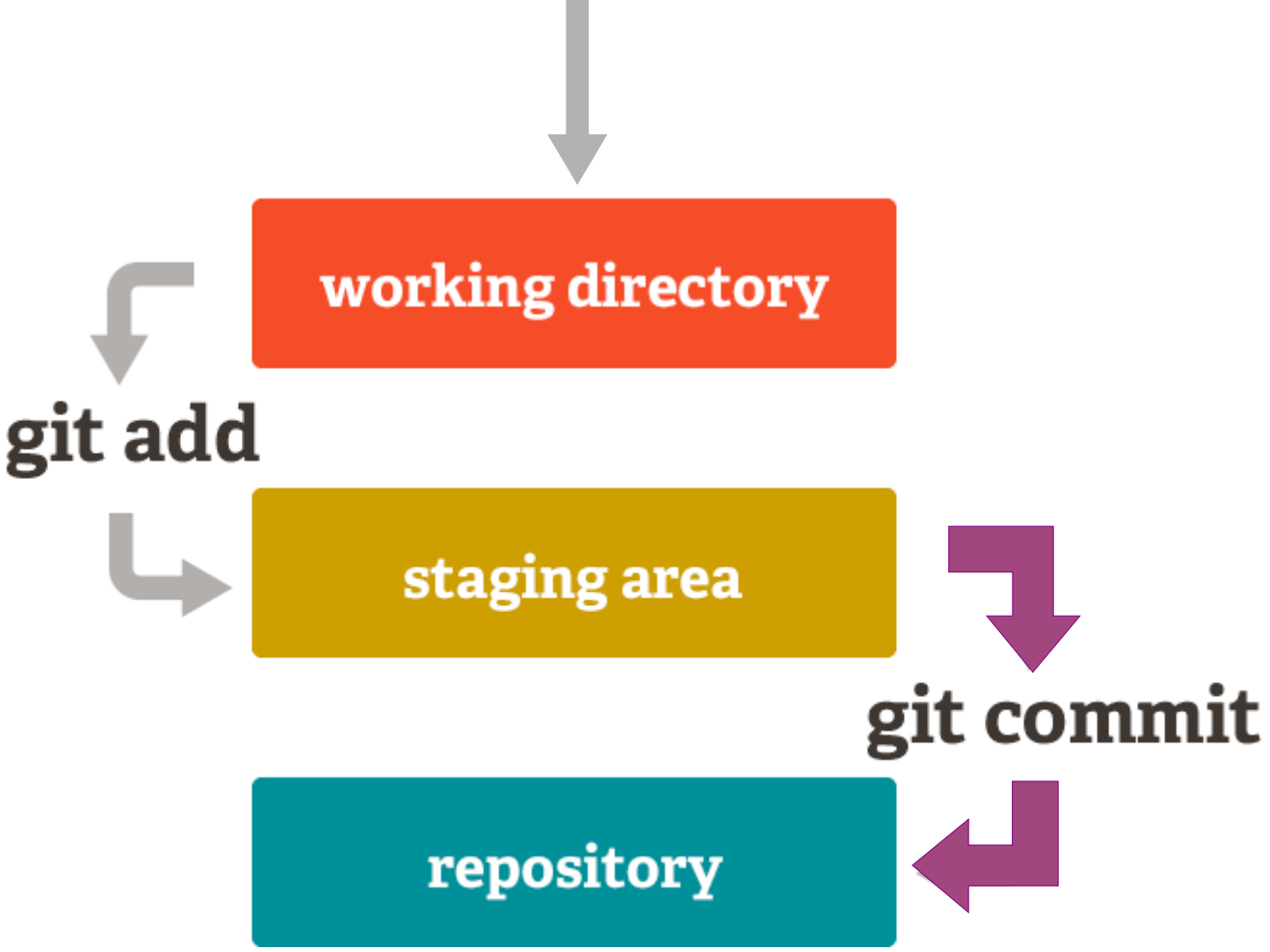
PRODUZINDO CÓDIGO VERSIONADO

- Exercício: adicione a modificação no arquivo ao índice, e então verifique novamente o status:
  - `git add .`
  - `git status`

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   meu-arquivo-1

rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$
```


working directory
git add
staging area
git commit
repository

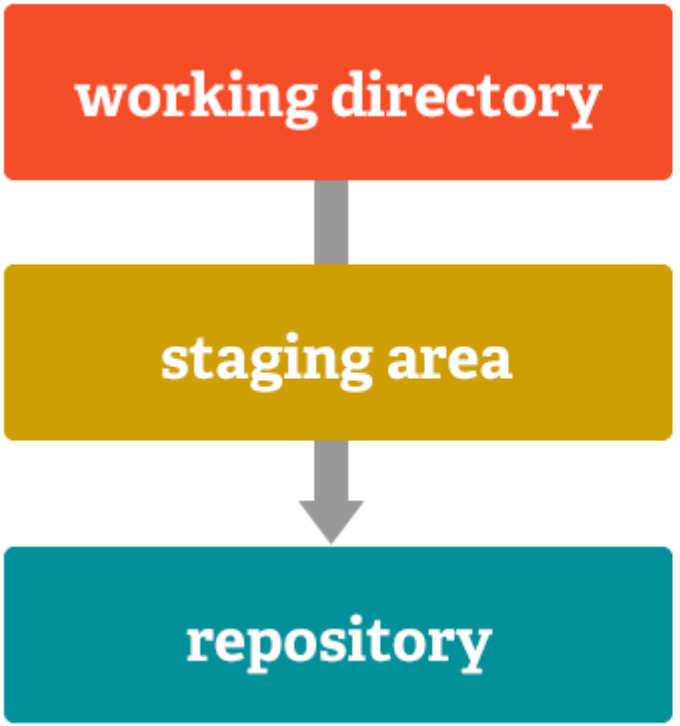
PRODUZINDO CÓDIGO VERSIONADO

- Exercício: faça *commit* da modificação, e então verifique mais uma vez o status:
  - `git commit -m ‘Altera meu-arquivo-1’`
  - `git status`

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$ git status
On branch master
nothing to commit, working tree clean
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$
```

PRODUZINDO CÓDIGO VERSIONADO

- `git commit -a -m ‘Gera commit com modificações em arquivos existentes’`



Funciona apenas para arquivos já incluídos em *commits* anteriores!

git commit -a

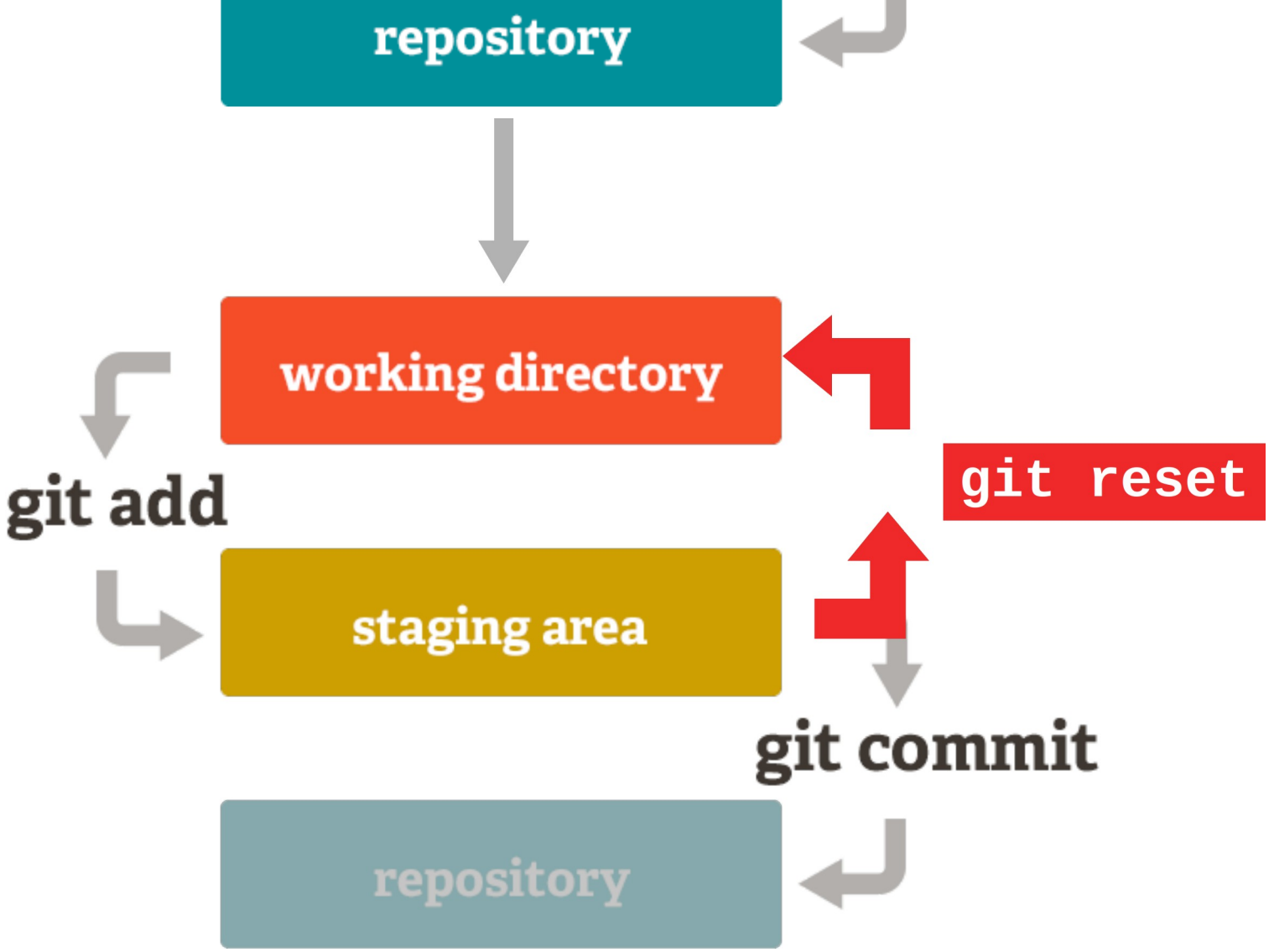
PRODUZINDO CÓDIGO VERSIONADO

- Para restringir os arquivos adicionados ao índice, pode-se enumerar os caminhos:
  - `git add meu-arquivo-1`
  - `git add meu-arquivo-*`

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   meu-arquivo-1

rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$
```


repository
working directory
git add
git reset
staging area
git commit
repository

PRODUZINDO CÓDIGO VERSIONADO

- Para remover arquivos do índice, pode-se utilizar o comando `reset`:
  - `git reset meu-arquivo-1`
  - `git reset meu-arquivo-*`

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   meu-arquivo-1

rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$
```

PRODUZINDO CÓDIGO VERSIONADO

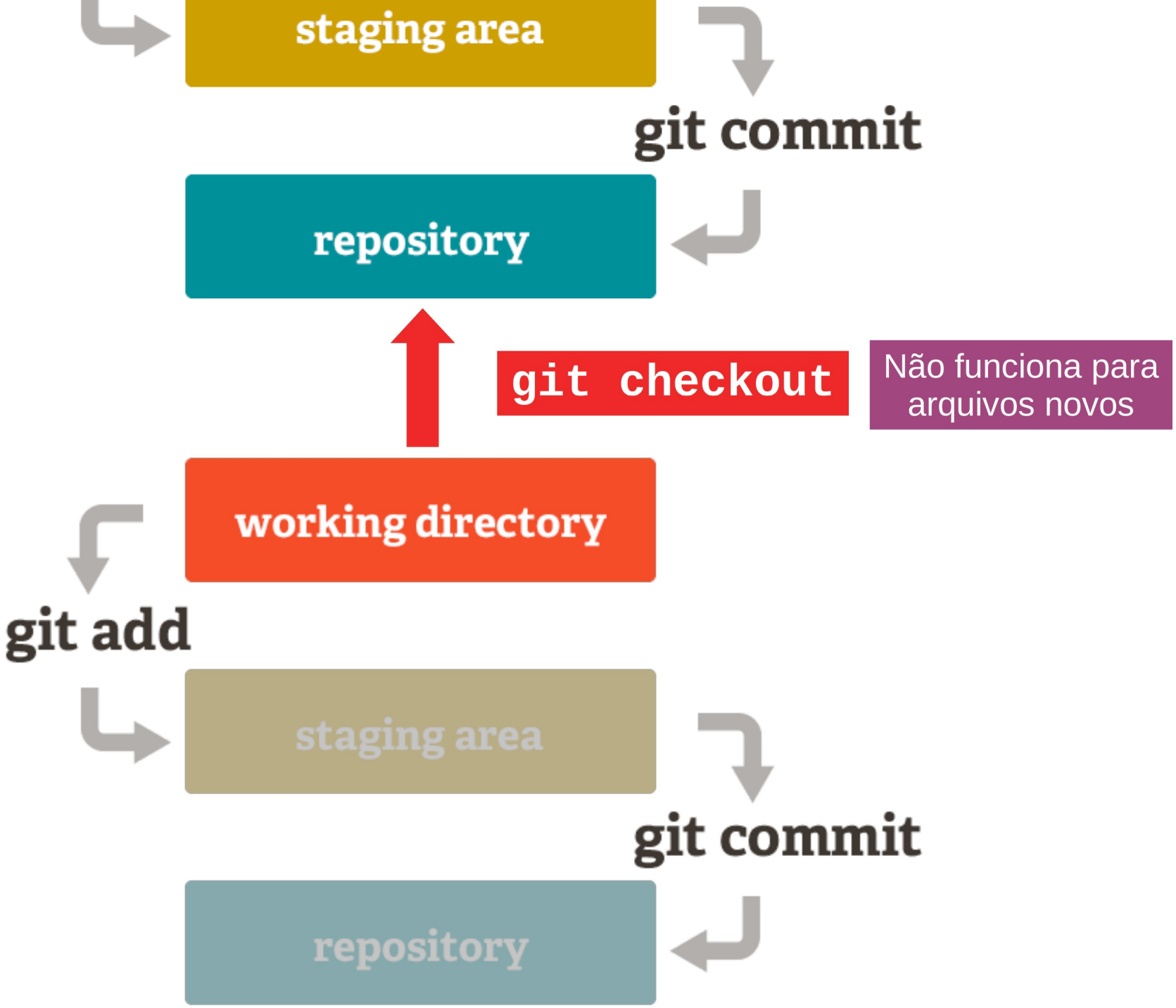
- Para adicionar ou remover todos os arquivos do índice, pode-se utilizar o `.`, que nesse caso representa o diretório atual:
  - `git add .`
  - `git reset .`

PRODUZINDO CÓDIGO VERSIONADO

- Para selecionar de modo interativo os arquivos a serem adicionados ou removidos do índice, pode-se utilizar o parâmetro `-i`:
  - `git add -i`

```
rui@gotham:~/Documents/Projetos/CEFWM/temp/novo-repo$ git add -i
           staged     unstaged path
  1:        +3/-0      nothing meu-arquivo-2
  2:        +1/-0      nothing meu-arquivo-3

*** Commands ***
  1: status       2: update       3: revert       4: add untracked
  5: patch        6: diff         7: quit         8: help
What now>
```

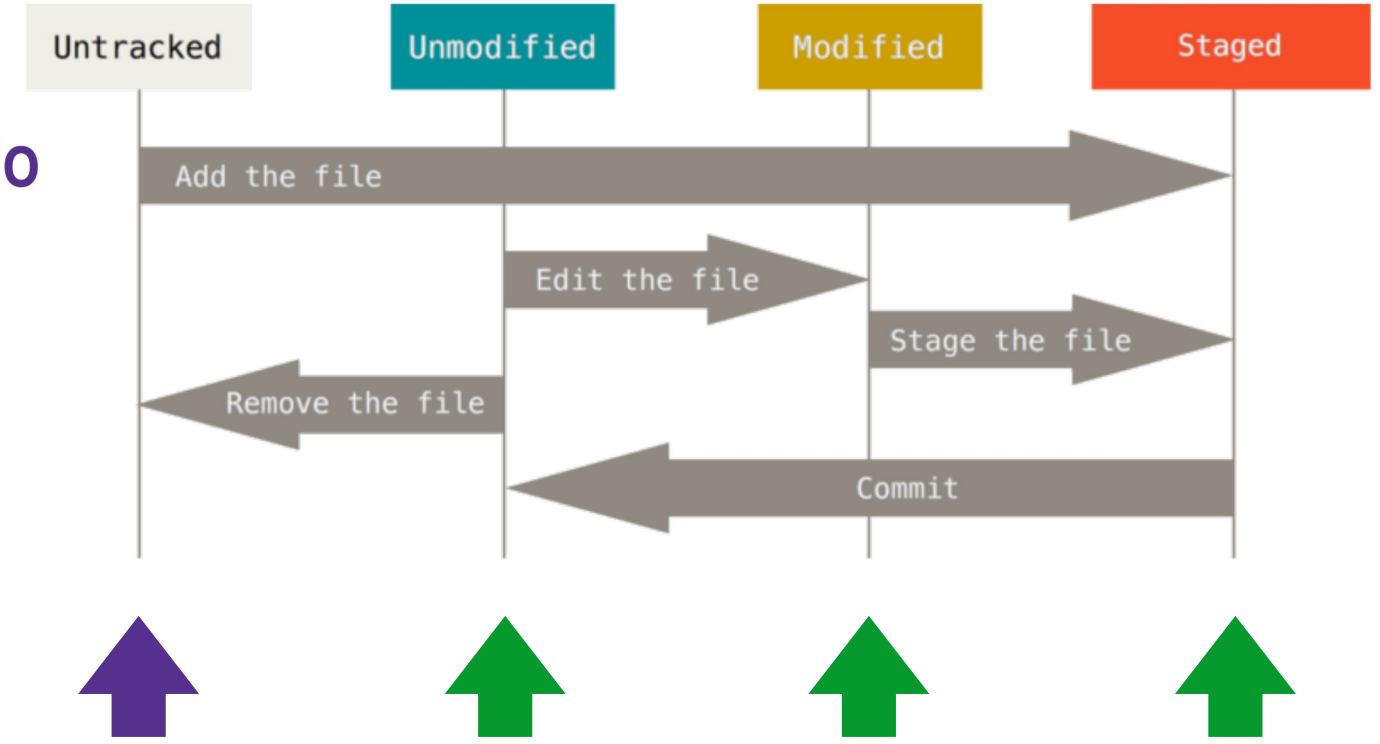

staging area
git commit
repository
git checkout
Não funciona para arquivos novos
working directory
git add
staging area
git commit
repository

PRODUZINDO CÓDIGO VERSIONADO

- Para desfazer modificações em arquivos do *working directory*, pode-se utilizar o comando `checkout`:
  - `git checkout -- meu-arquivo-3`
  - `git checkout -- .`

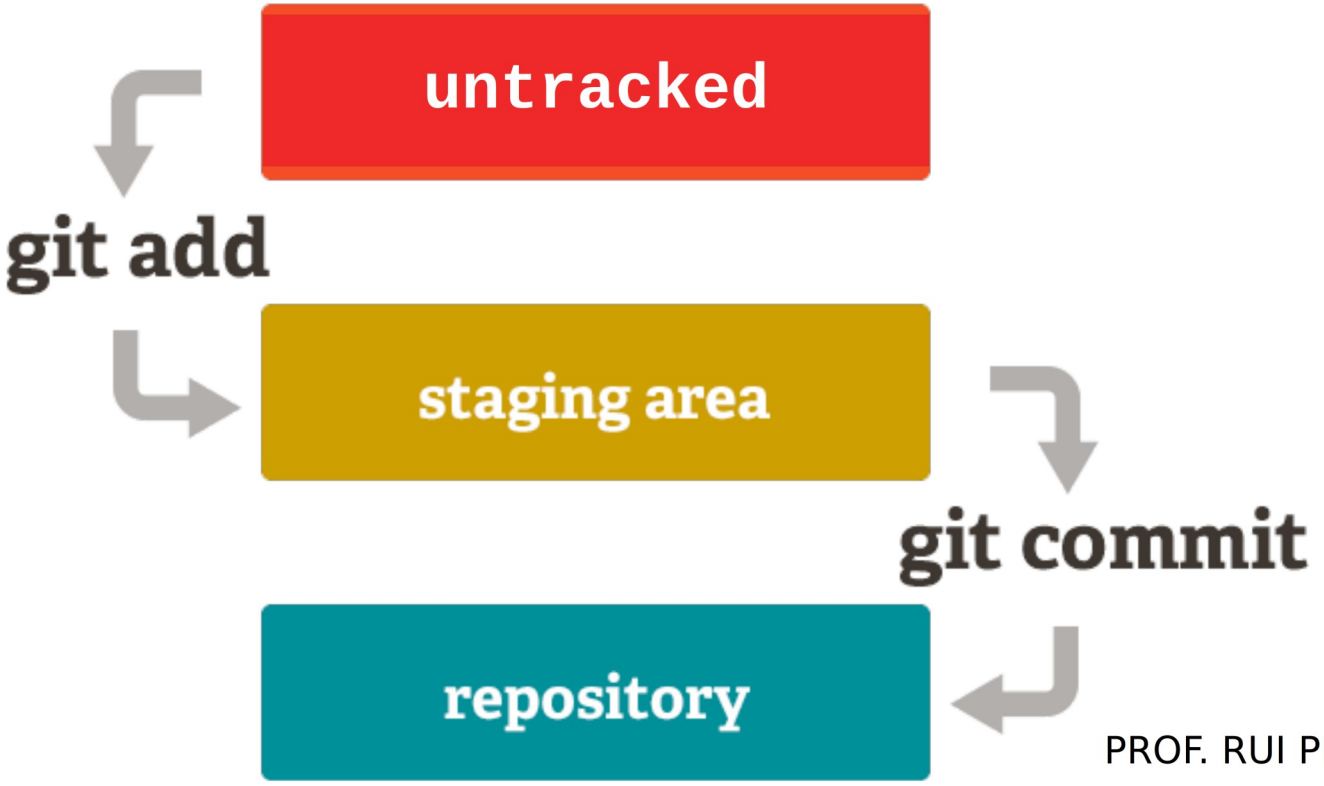
PRODUZINDO CÓDIGO VERSIONADO

- No Git, cada arquivo pode estar em um de dois estados:
  - **Não rastreado**
  - **Rastreado**



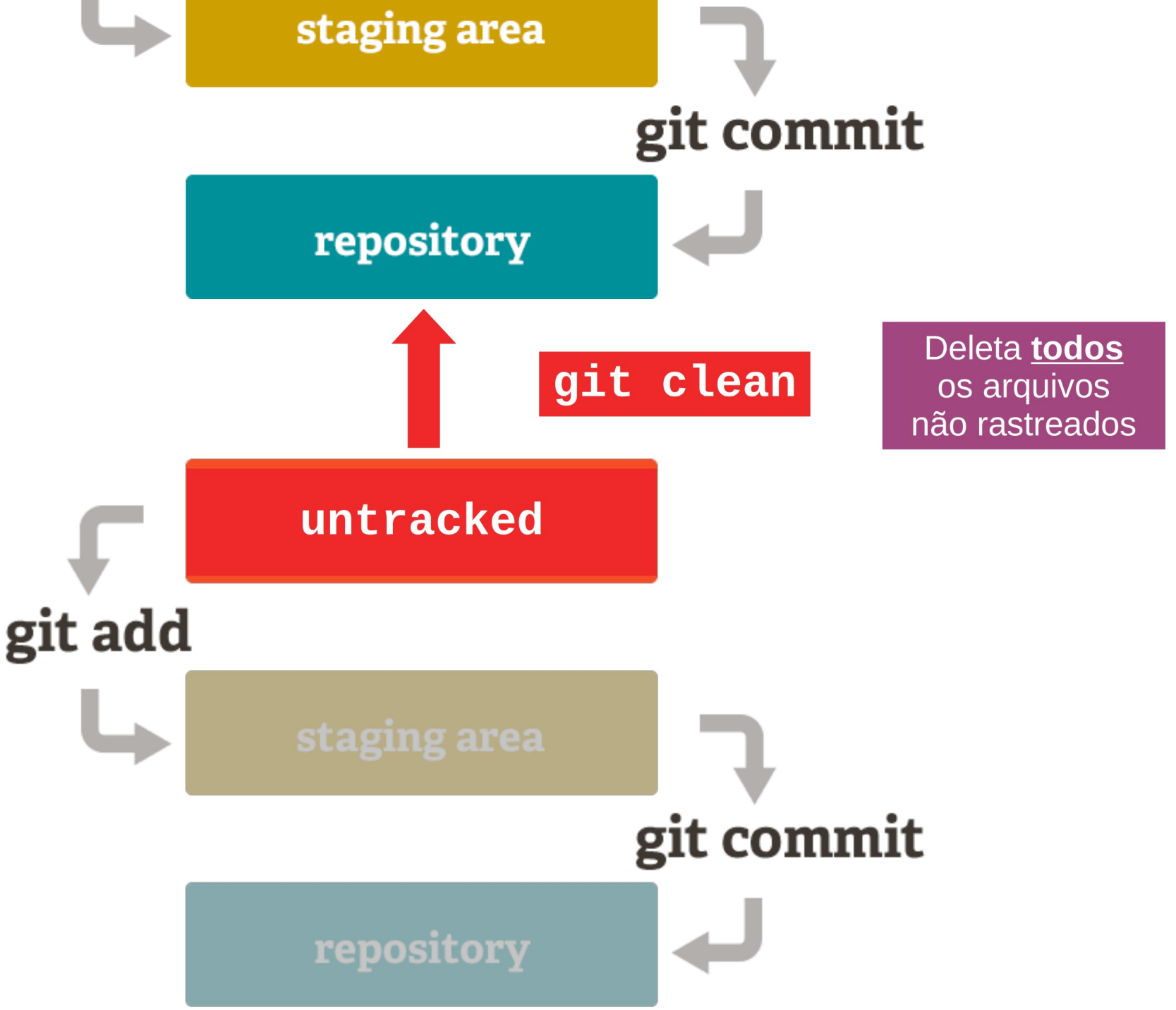
PRODUZINDO CÓDIGO VERSIONADO

- Para adicionar arquivos **não rastreados** ao repositório, procede-se da mesma maneira que para as alterações dos arquivos já **rastreados**



PRODUZINDO CÓDIGO VERSIONADO

- Para deletar de modo seletivo, pode-se utilizar o comando disponível no próprio SO:
  - `rm meu-arquivo-2`
- Para deletar **todos** os arquivos **não rastreados**, utiliza-se o comando `clean`:
  - `git clean -df`


staging area
git commit
repository
git clean
Deleta **todos** os arquivos não rastreados
untracked
git add
staging area
git commit
repository

PRODUZINDO CÓDIGO VERSIONADO

- Como já vimos, o `git add .` adiciona todos os arquivos ao índice. O que não sabíamos é que ele adiciona inclusive arquivos não rastreados
- Arquivos não rastreados também são adicionados ao utilizar *wildcards*, como `meu-arquivo-*`

PRODUZINDO CÓDIGO VERSIONADO

- Exercício: crie novos arquivos (`meu-arquivo-4`, `meu-arquivo-5` e `meu-arquivo-6`) e avalie o efeito dos comandos a seguir:
  - `git add .`
  - `git reset .`
  - `git add meu-arquivo-*`
  - `git reset meu-arquivo-*`

PRODUZINDO CÓDIGO VERSIONADO

- Na realidade, é muito comum adicionar arquivos não rastreados ao índice de modo acidental; por isso, pode-se criar o arquivo `.gitignore`, que indica caminhos a serem ignorados
- Atenção: arquivos rastreados (desde o momento de sua adição ao índice) não sofrem efeito do `.gitignore`

PRODUZINDO CÓDIGO VERSIONADO

- Para listar os arquivos não rastreados ignorados, pode-se utilizar uma variação do `git status`:
  - `git status --ignored`

PRODUZINDO CÓDIGO VERSIONADO

- Exercício: adicione `meu-arquivo-5` aos arquivos ignorados, e então reavalie o efeito dos comandos a seguir:
  - `git add .`
  - `git reset .`
  - `git add meu-arquivo-*`
  - `git add meu-arquivo-5`
  - `git add -f meu-arquivo-5`

PRODUZINDO CÓDIGO VERSIONADO

- Frequentemente, um arquivo não rastreado acaba sendo acidentalmente incluído em um *commit*
- A opção mais fácil e segura para torná–lo não rastreado novamente é criar um *commit* adicional com sua remoção, de modo que o Git não mais o considere parte do repositório
- A única desvantagem desse método é a criação de um novo *commit*

PRODUZINDO CÓDIGO VERSIONADO

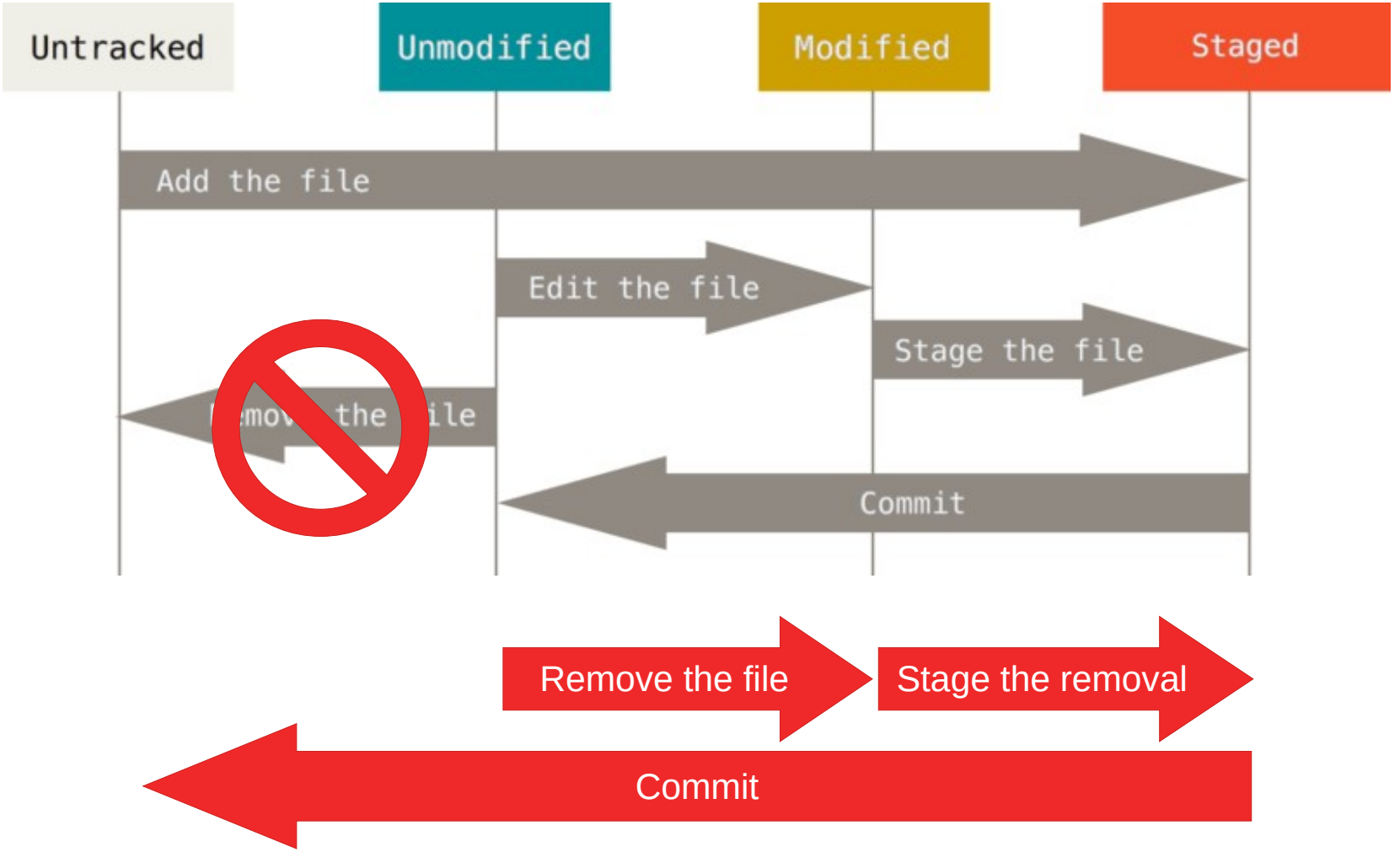
- Para remover um arquivo do repositório e do disco:
  - `git rm meu-arquivo-2`
  - `git commit`

Equivalente à remoção do arquivo do disco, seguida de um **`git add`**

- Para removê-lo somente do repositório:
  - `git rm --cached meu-arquivo-2`
  - `git commit`

Equivalente ao recorte do arquivo do disco, seguido de um **`git add`**, e então da colagem do arquivo novamente

PRODUZINDO CÓDIGO VERSIONADO



PRODUZINDO CÓDIGO VERSIONADO

- Tarefas:
  - Para pensar: suponha um projeto que contenha código fonte, e artefatos compilados a partir desse código fonte. Devemos fazer *commit* desses arquivos compilados?
  - Investigue a diferença entre os comandos `ls`, `git ls-files` e `git status`. Dica: crie ao menos um arquivo ignorado e então teste os 3 comandos.

PRODUZINDO CÓDIGO VERSIONADO

- Resumo até agora:
  - `git init` cria o diretório `.git/`, transformando a pasta em um repositório Git
  - Arquivos iniciam não rastreados
  - Arquivos não rastreados podem ser ignorados
  - Arquivos se tornam rastreados com o `git add`

PRODUZINDO CÓDIGO VERSIONADO

- Resumo até agora:
  - Cada arquivo rastreado pode se encontrar não modificado, modificado ou adicionado ao índice
  - `git add` e `git reset` são usados para acrescentar e remover arquivos do índice
  - `git checkout` altera no disco o conteúdo de arquivos rastreados (restaurando-os ao estado não modificado)

PRODUZINDO CÓDIGO VERSIONADO

- Resumo até agora:
  - `git clean` remove do disco todos os arquivos não rastreados
  - `git add` necessita de *flag* `-f` para adicionar arquivos ignorados ao índice
  - `git rm` remove (ou não) um arquivo rastreado do disco e adiciona a remoção ao índice

PRODUZINDO CÓDIGO VERSIONADO

- Já somos capazes de:
  - Inicializar um repositório
  - Criar novos *commits*
  - Adicionar e remover arquivos do repositório
  - Desfazer alterações indesejadas
- Mas como fazer backup do repositório?
- Como trabalhar colaborativamente?