



VERSIONAMENTO

GIT – SISTEMA DE VERSIONAMENTO DE CÓDIGO

O que é controle de versão e por quê ele é importante?

O sistema de controle de versão é responsável por gerenciar as diferentes versões de um ou vários documentos

Apaguei um arquivo importante e limpei a lixeira, e agora?



Projeto portfólio v1

Projeto portfólio v2

- Alterar o nome
- Alterar animação
- Alterar cores

Projeto portfólio v3

- Alterar fundo
- Alterar título

Projeto portfólio v4

- Ajustar tags



VERSIONAMENTO

GIT – SISTEMA DE VERSIONAMENTO DE CÓDIGO

Controlam as versões de um arquivo ao longo do tempo.



Registra o histórico de atualizações de um arquivo;



Gerencia quais foram as alterações, a data, autor, etc.;



Organização, controle e segurança.



VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO

Linha do tempo



Versão 1

index.html

style.css

fundo.png

Versão 2

index.html (V2)

style.css

fundo.png

Versão 3

index.html (V3)

style.css (V2)

fundo.png



git



VERSIONAMENTO

GIT – SISTEMA DE VERSIONAMENTO DE CÓDIGO

Tipos de Sistemas de Controle de Versão

Dentre os Sistemas de Controle de Versão (VCS), temos:



VCS Centralizado (CVCS)

Ex.: CVS, Subversion.



VCS Distribuído (DVCS)

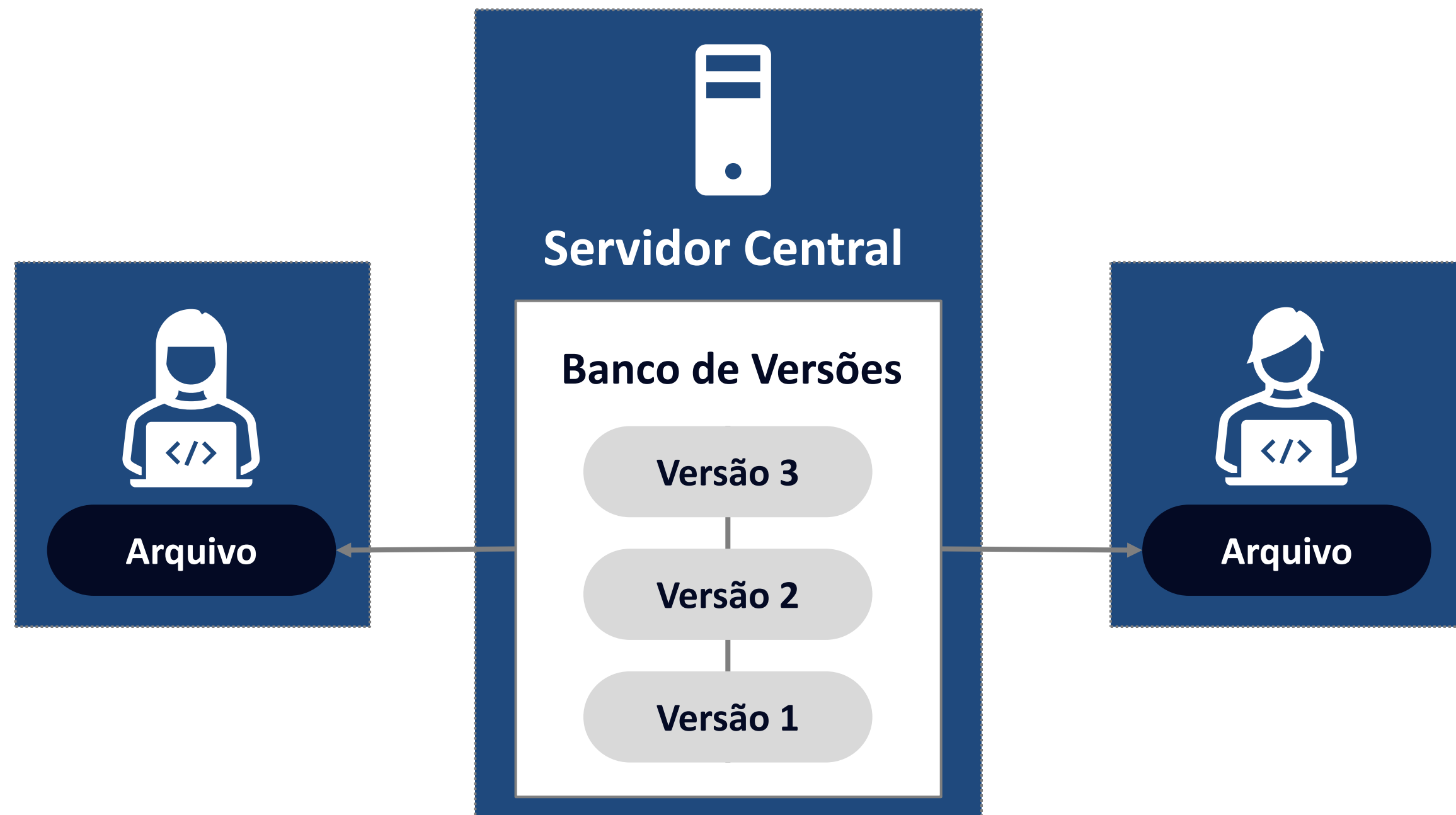
Ex.: Git, Mercurial.



VERSIONAMENTO

GIT – SISTEMA DE VERSIONAMENTO DE CÓDIGO

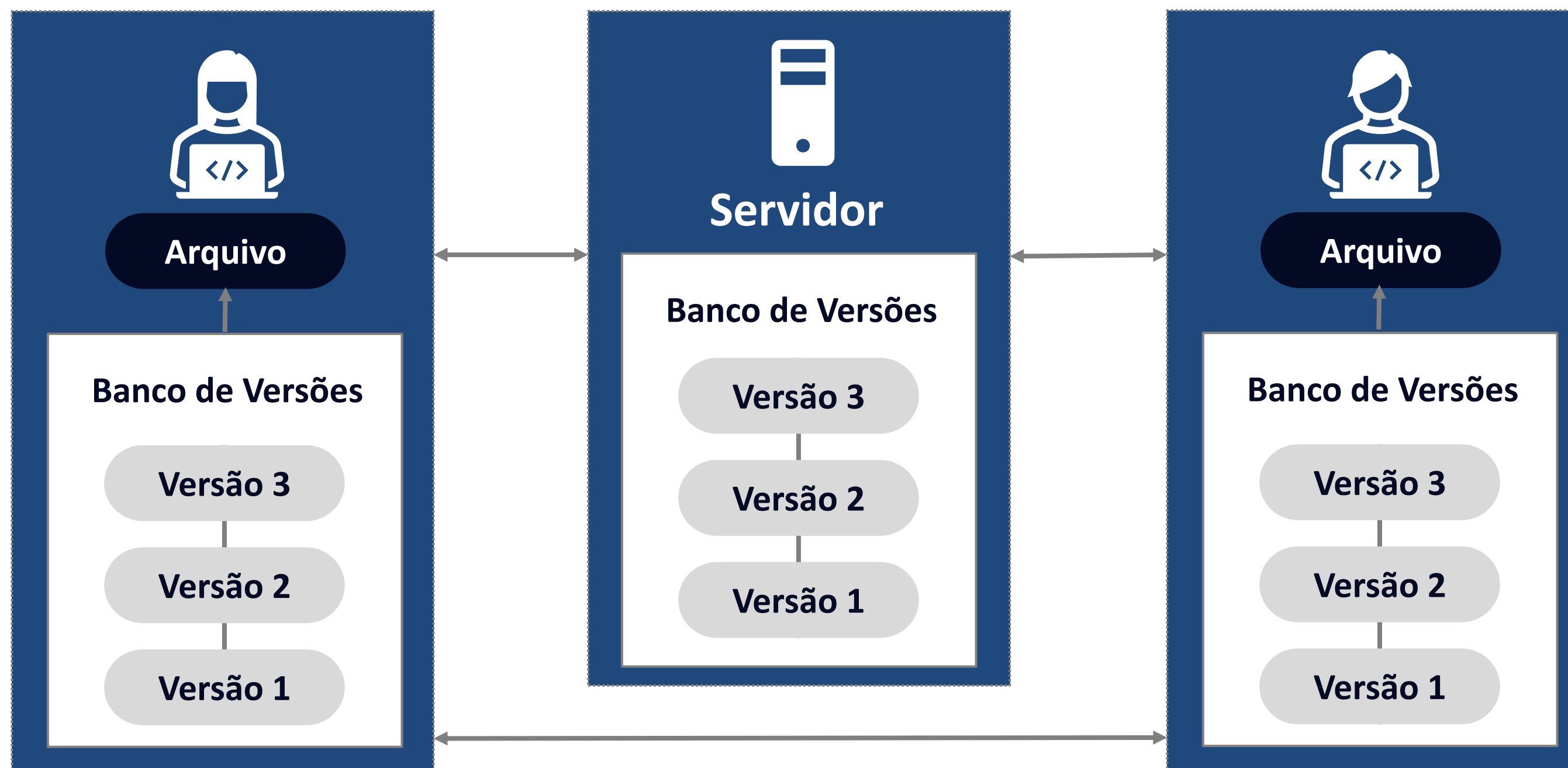
VCS Centralizado (CVCS)



VERSIONAMENTO

GIT – SISTEMA DE VERSIONAMENTO DE CÓDIGO

VCS Distribuído (DVCS)

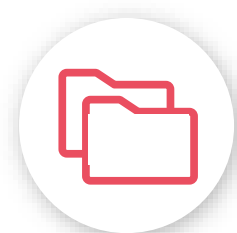


VERSIONAMENTO

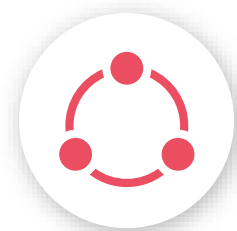
GIT – SISTEMA DE VERSIONAMENTO DE CÓDIGO

VCS Distribuído (DVCS)

Clona o repositório completo, o que inclui o histórico de versões.



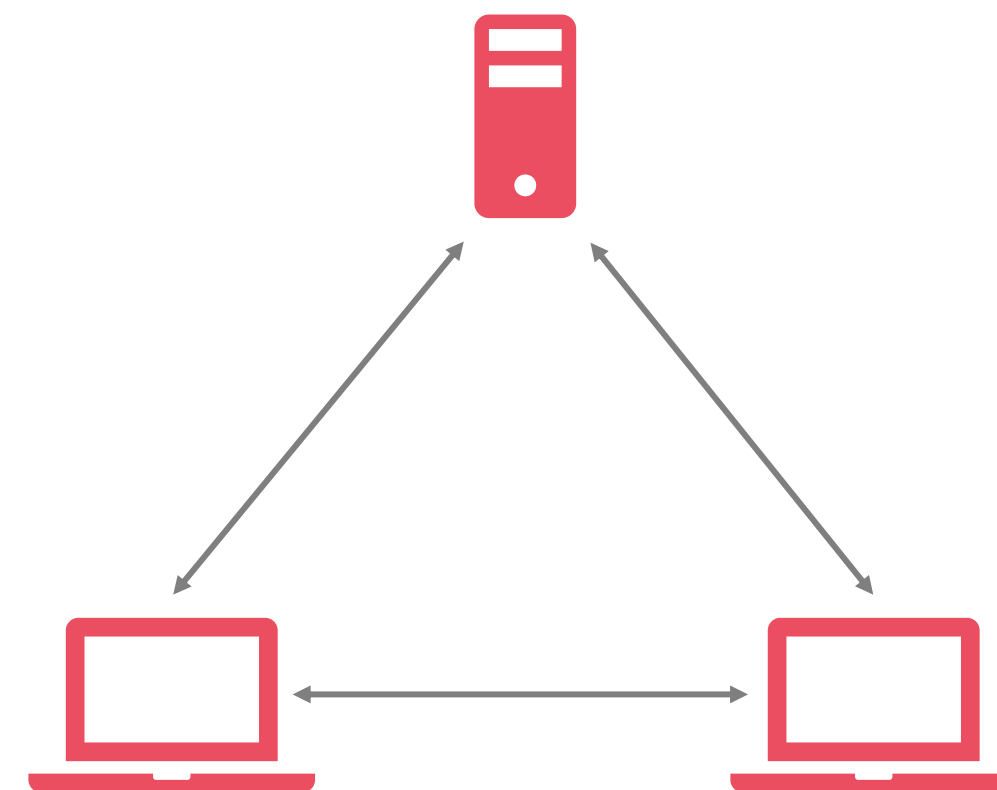
Cada clone é como um backup;



Possibilita um fluxo de trabalho flexível;



Possibilidade de trabalhar sem conexão à rede.



VERSIONAMENTO

GIT – SISTEMA DE VERSIONAMENTO DE CÓDIGO

O que é Git?

Sistema de Controle de Versão Distribuído.



{...}

Gratuito e Open Source (Código Aberto);



Ramificações (branching) e fusões (merging) eficientes;



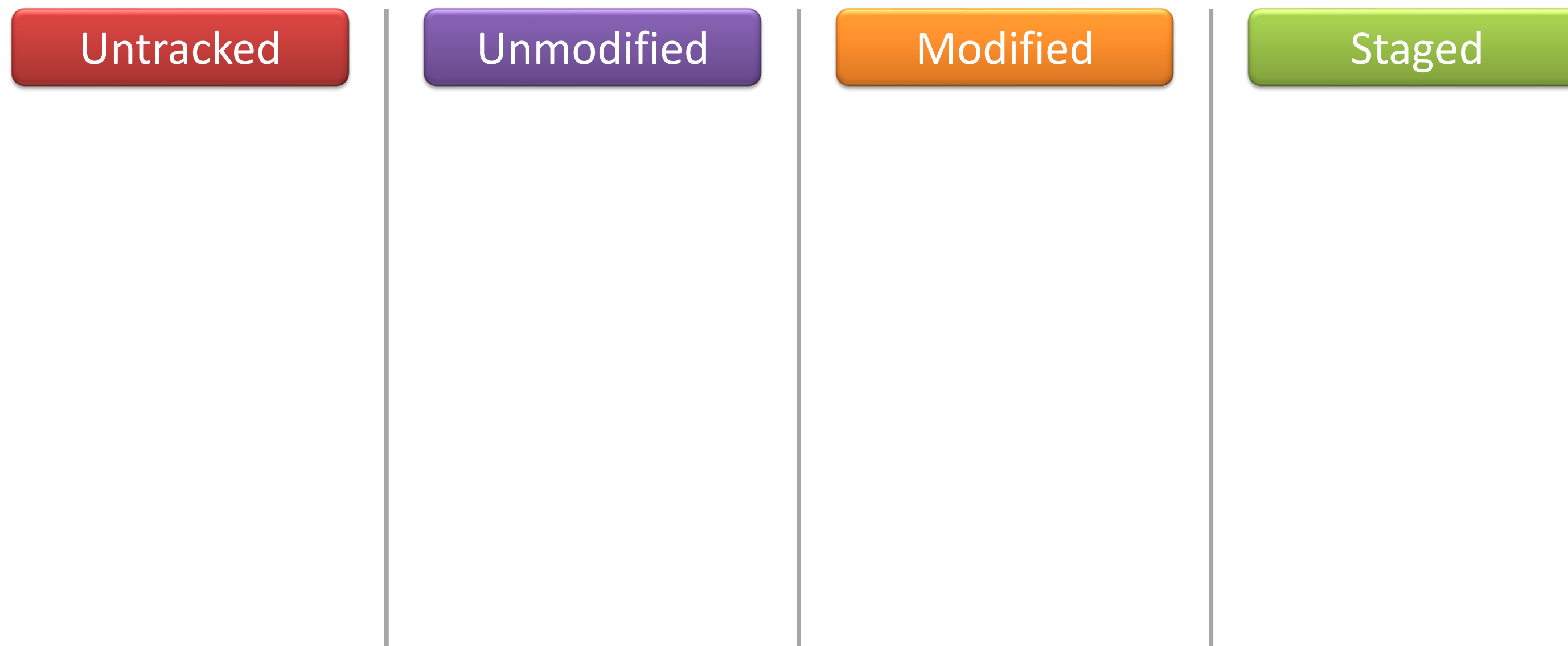
Leve e rápido.



VERSIONAMENTO

GIT – CICLO DE VIDA DOS ARQUIVOS

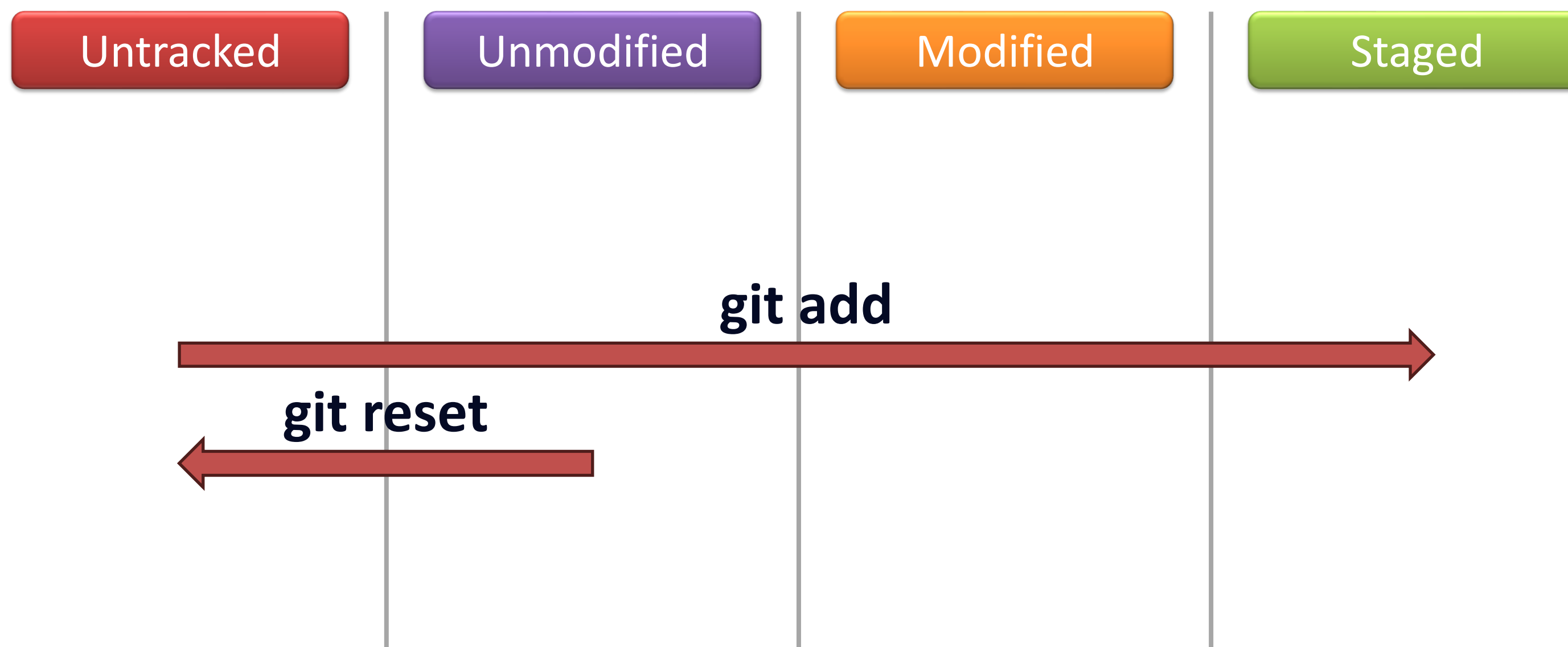
O Git possui 4 áreas de “armazenamento” dos arquivos



VERSIONAMENTO

GIT – CICLO DE VIDA DOS ARQUIVOS

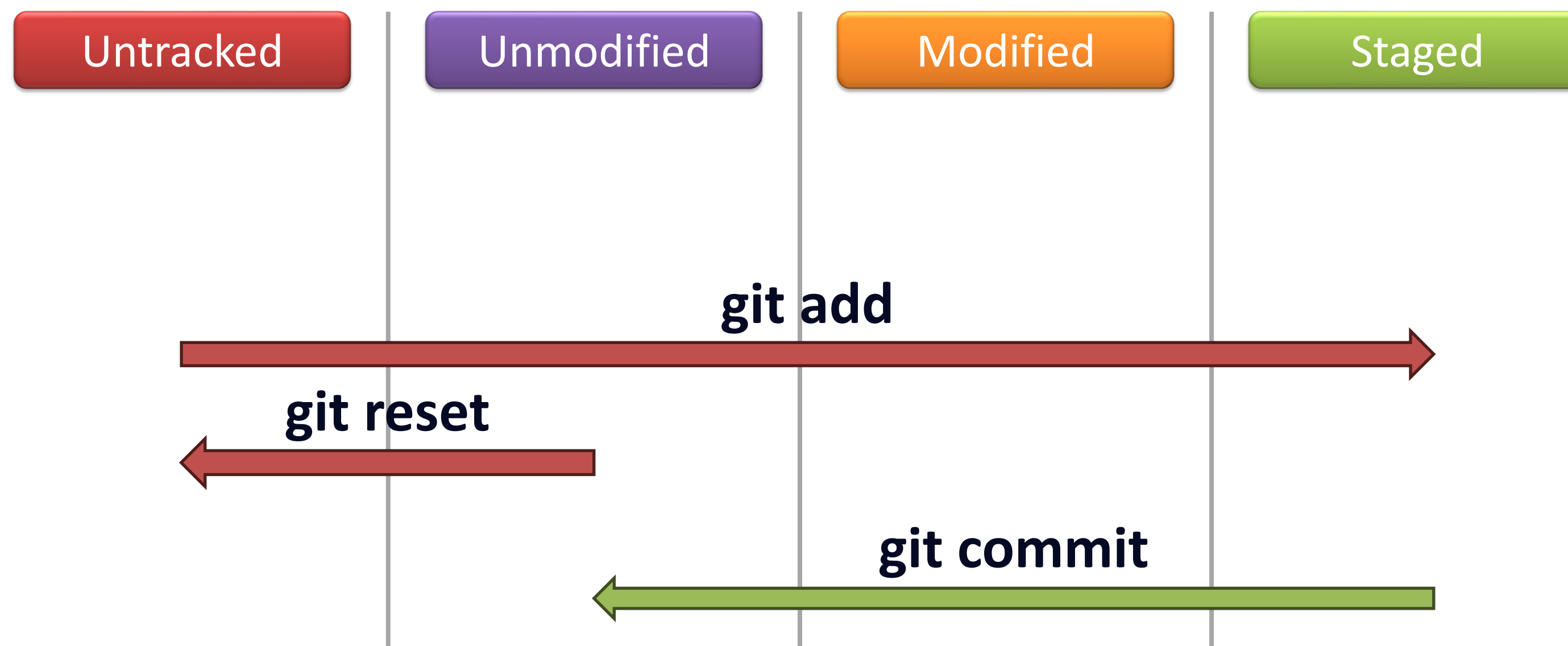
Untracked – o Git ainda não conhece a existência do arquivo



VERSIONAMENTO

GIT – CICLO DE VIDA DOS ARQUIVOS

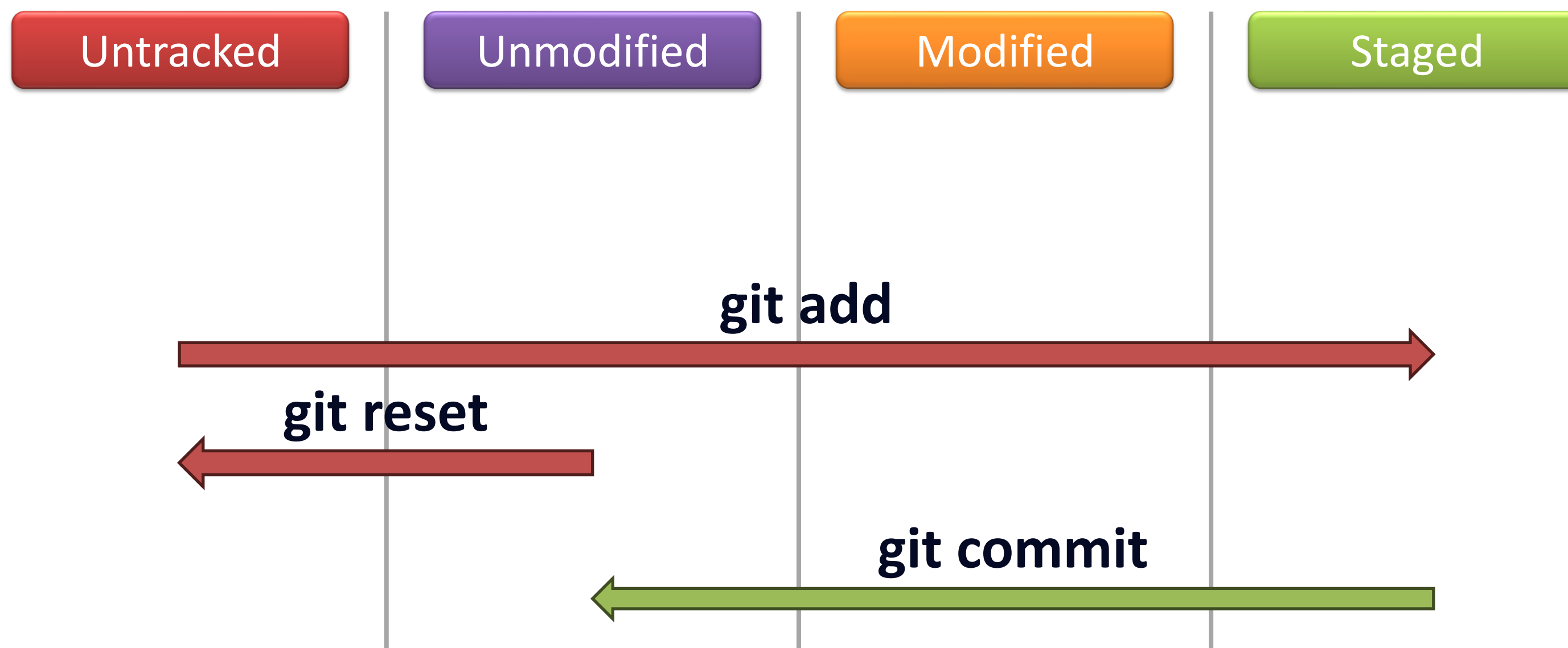
Staged – ficam os arquivos que estão prontos para serem inseridos na nova versão



VERSIONAMENTO

GIT – CICLO DE VIDA DOS ARQUIVOS

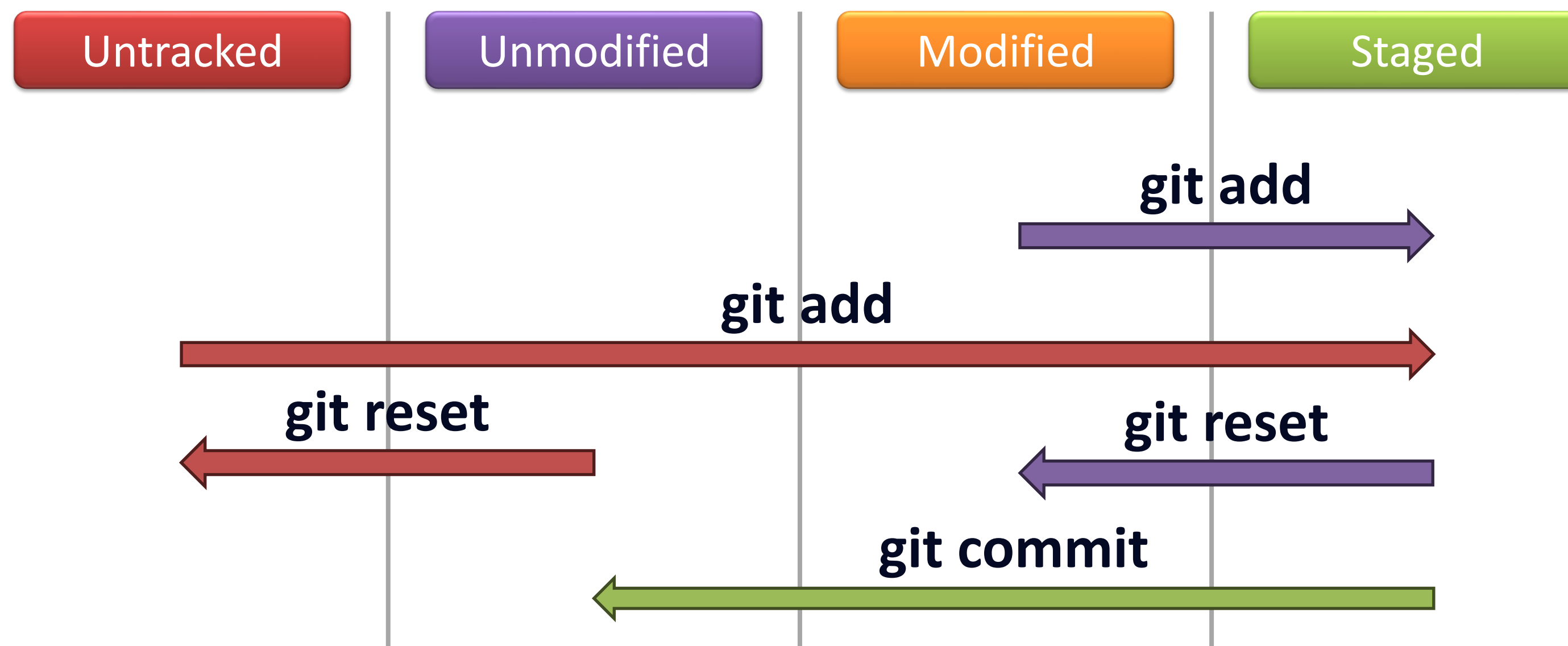
Unmodified – ficam os arquivos que não sofreram nenhuma alteração em relação a última versão



VERSIONAMENTO

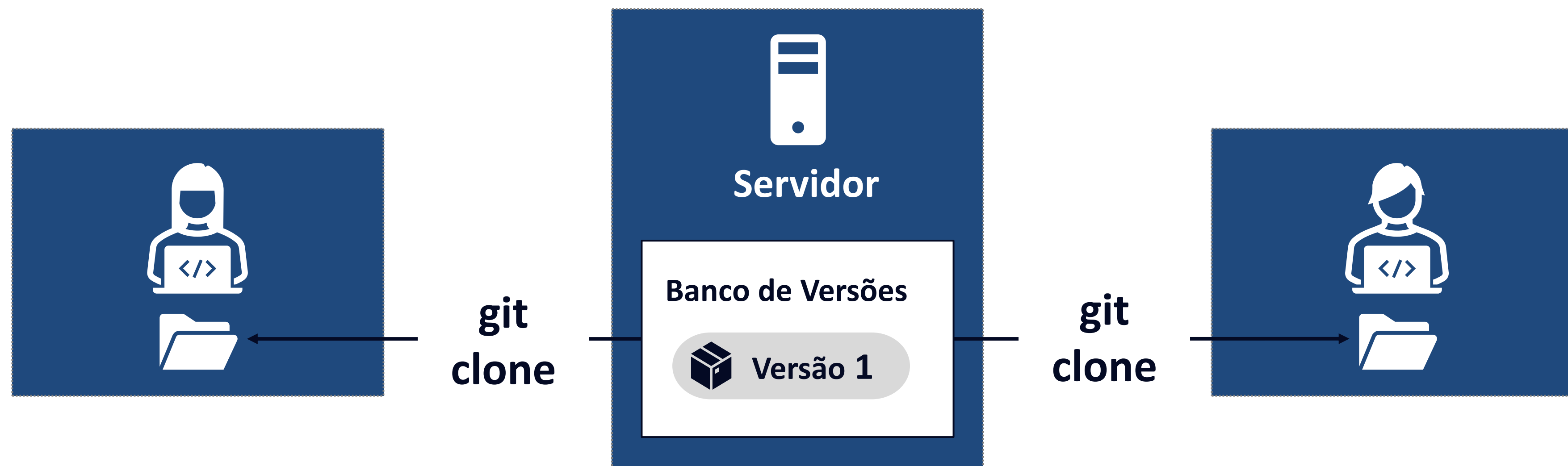
GIT – CICLO DE VIDA DOS ARQUIVOS

Modified – ficam os arquivos que sofreram alteração em relação a última versão



VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



git clone → clona um repositório Git existente para um novo diretório (pasta) local.



VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



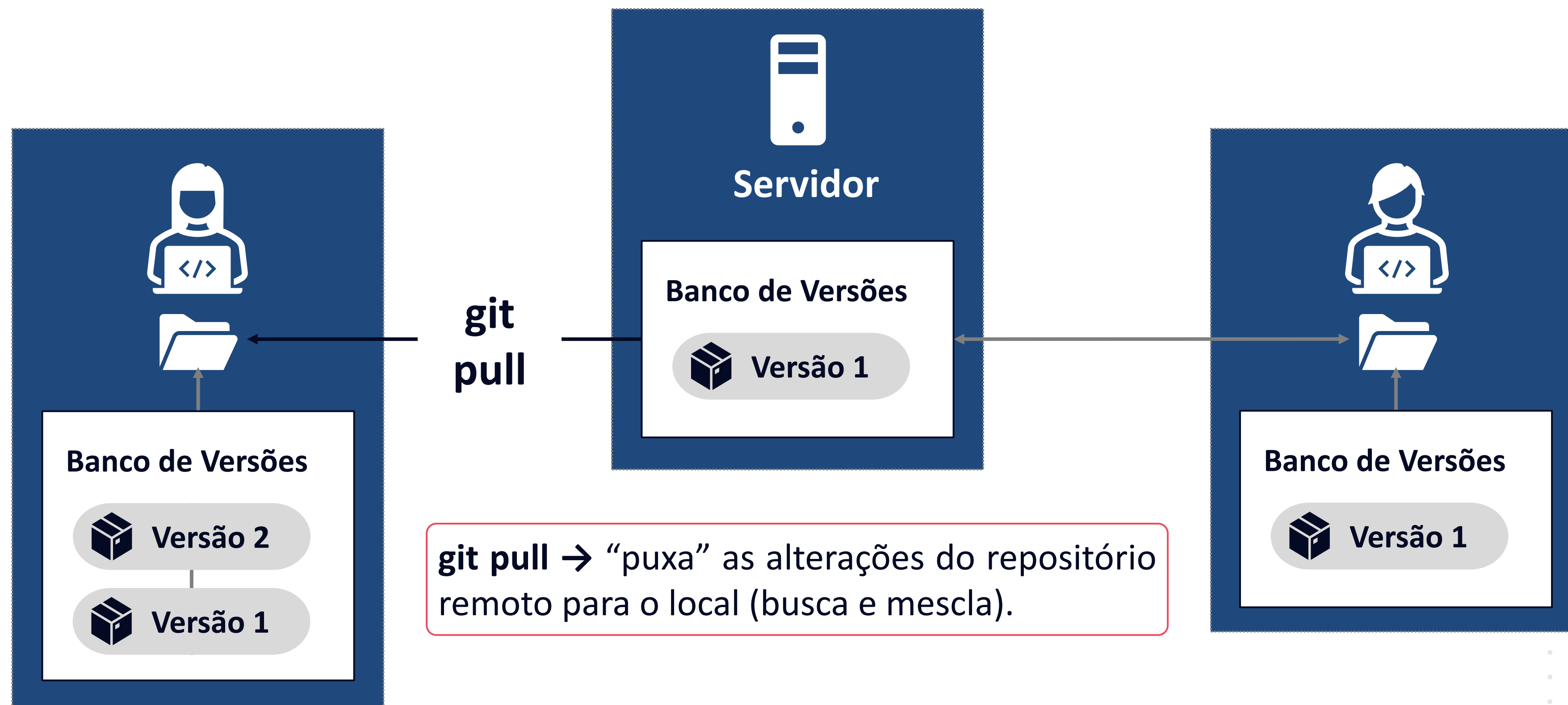
VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



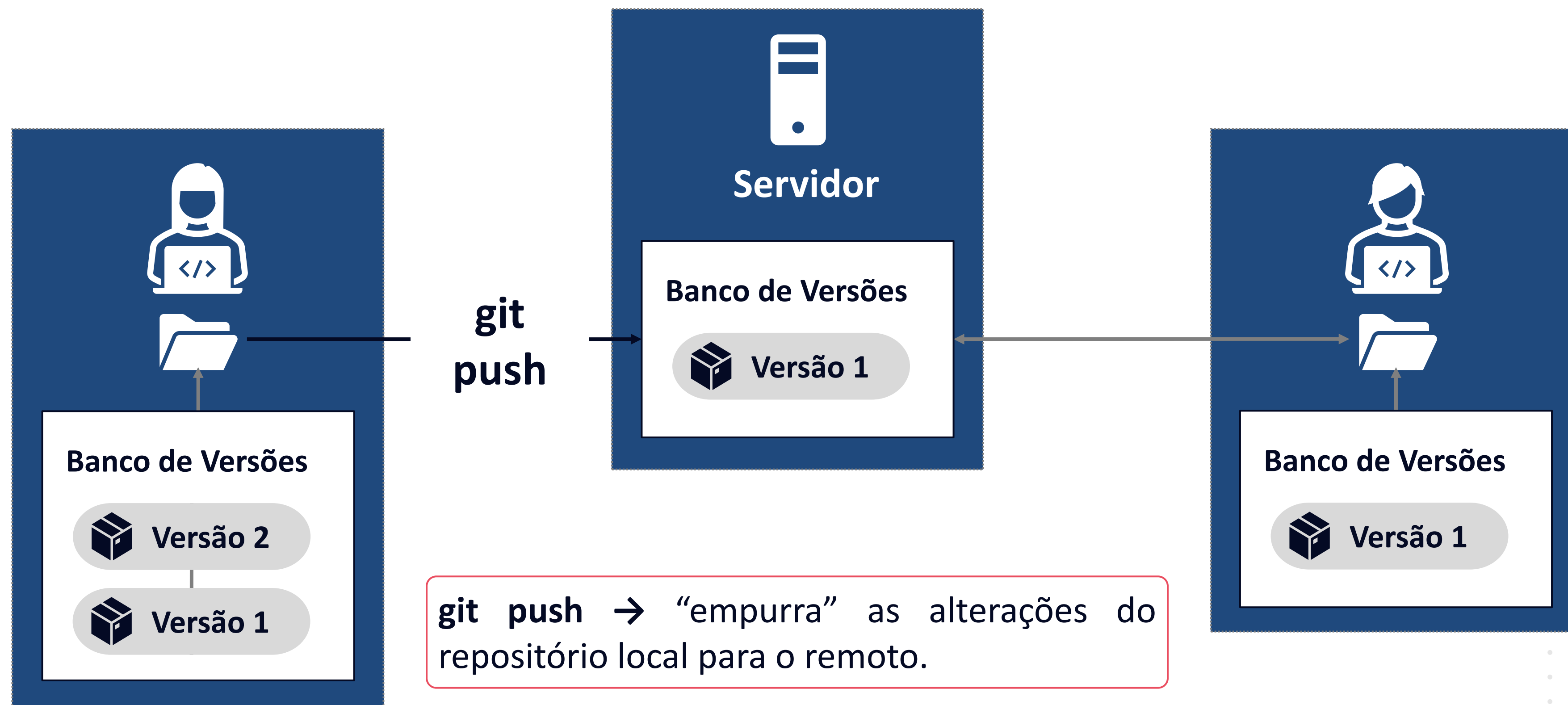
VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



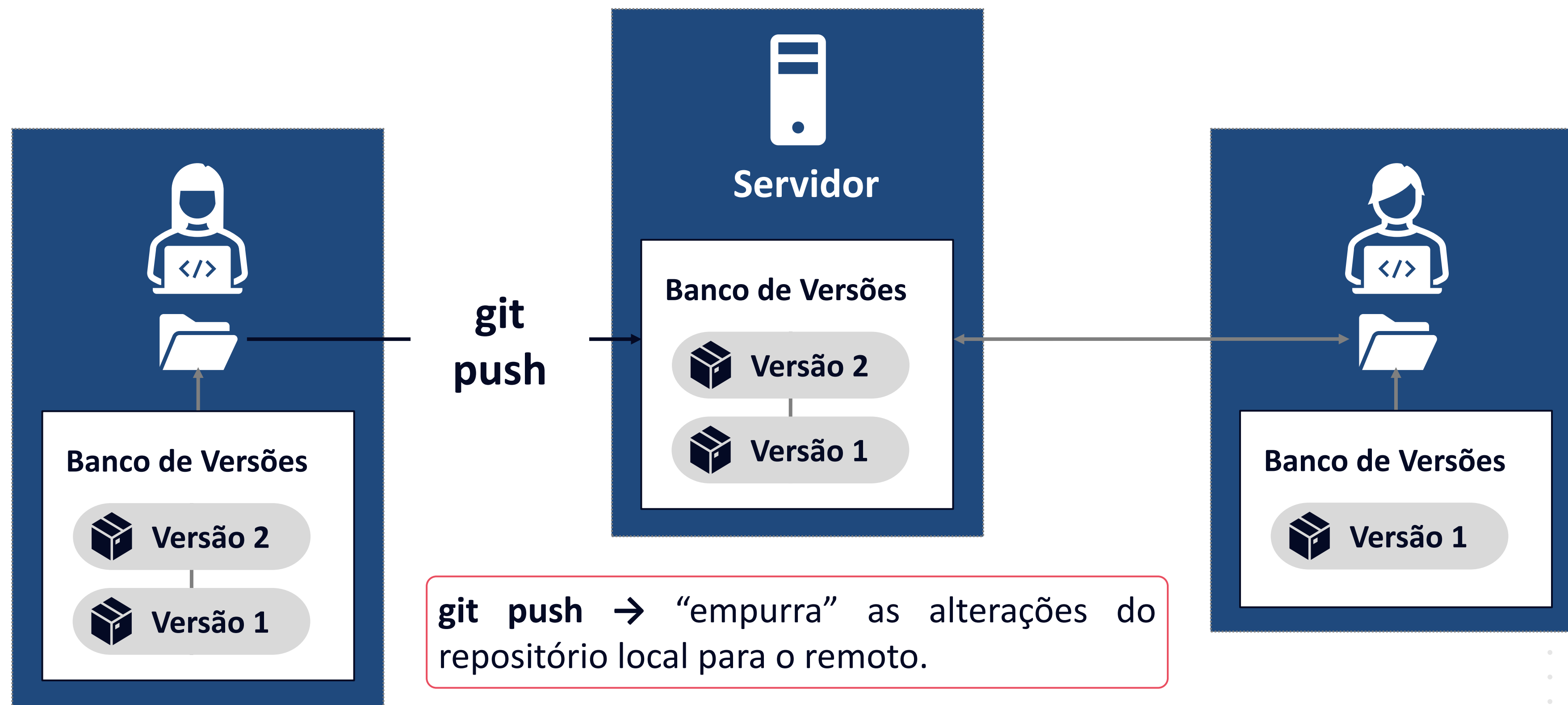
VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



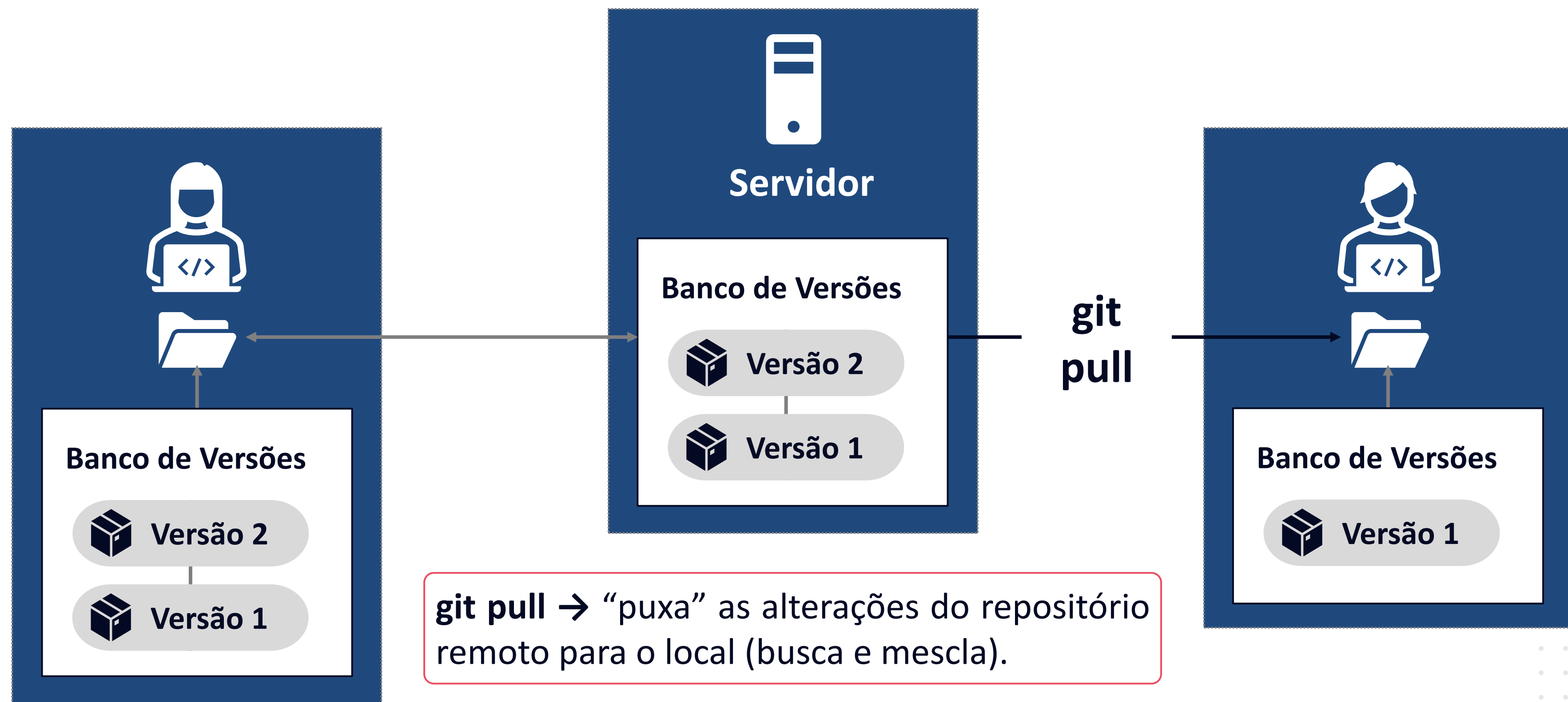
VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



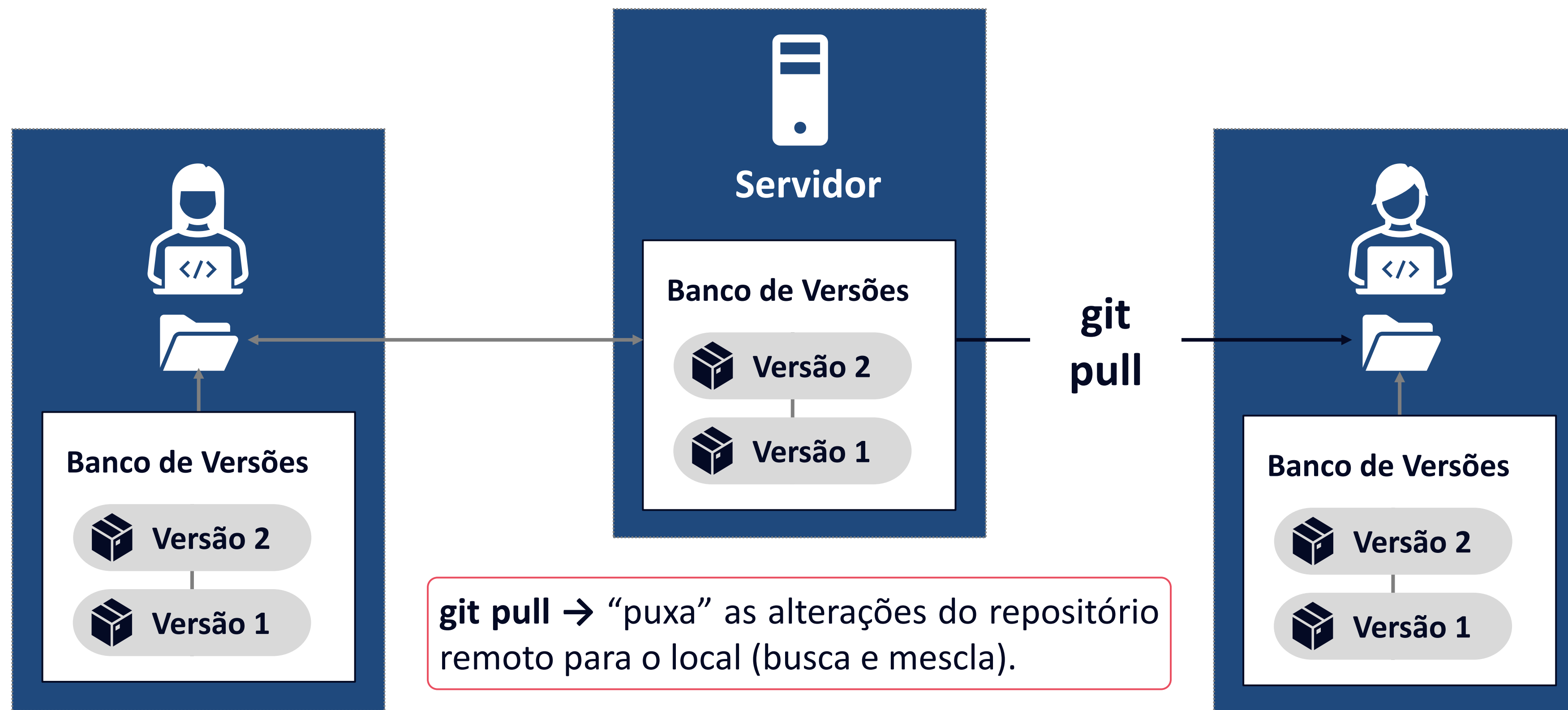
VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



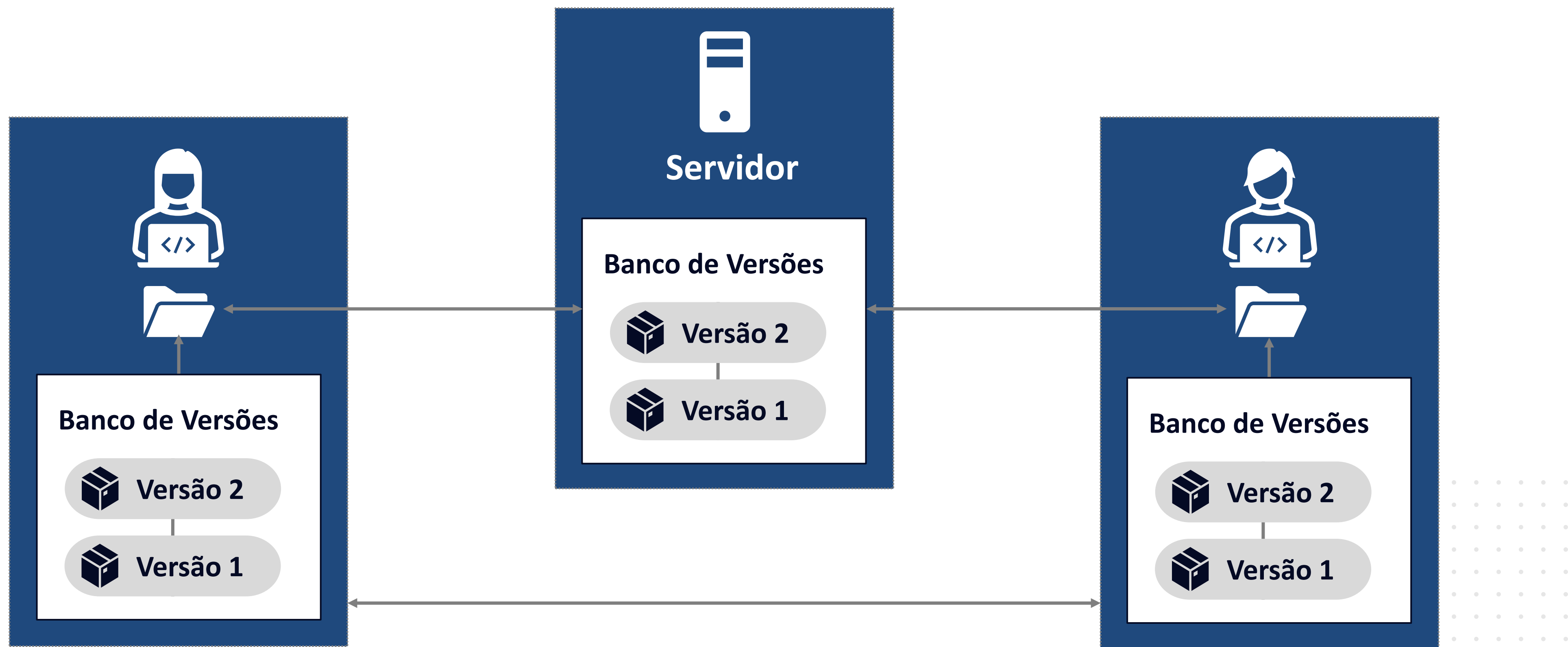
VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO



VERSIONAMENTO

GITHUB – O QUE É?

Plataforma de hospedagem de código para controle de versão com Git, e colaboração.



Comunidade ativa;



Utilizado mundialmente;

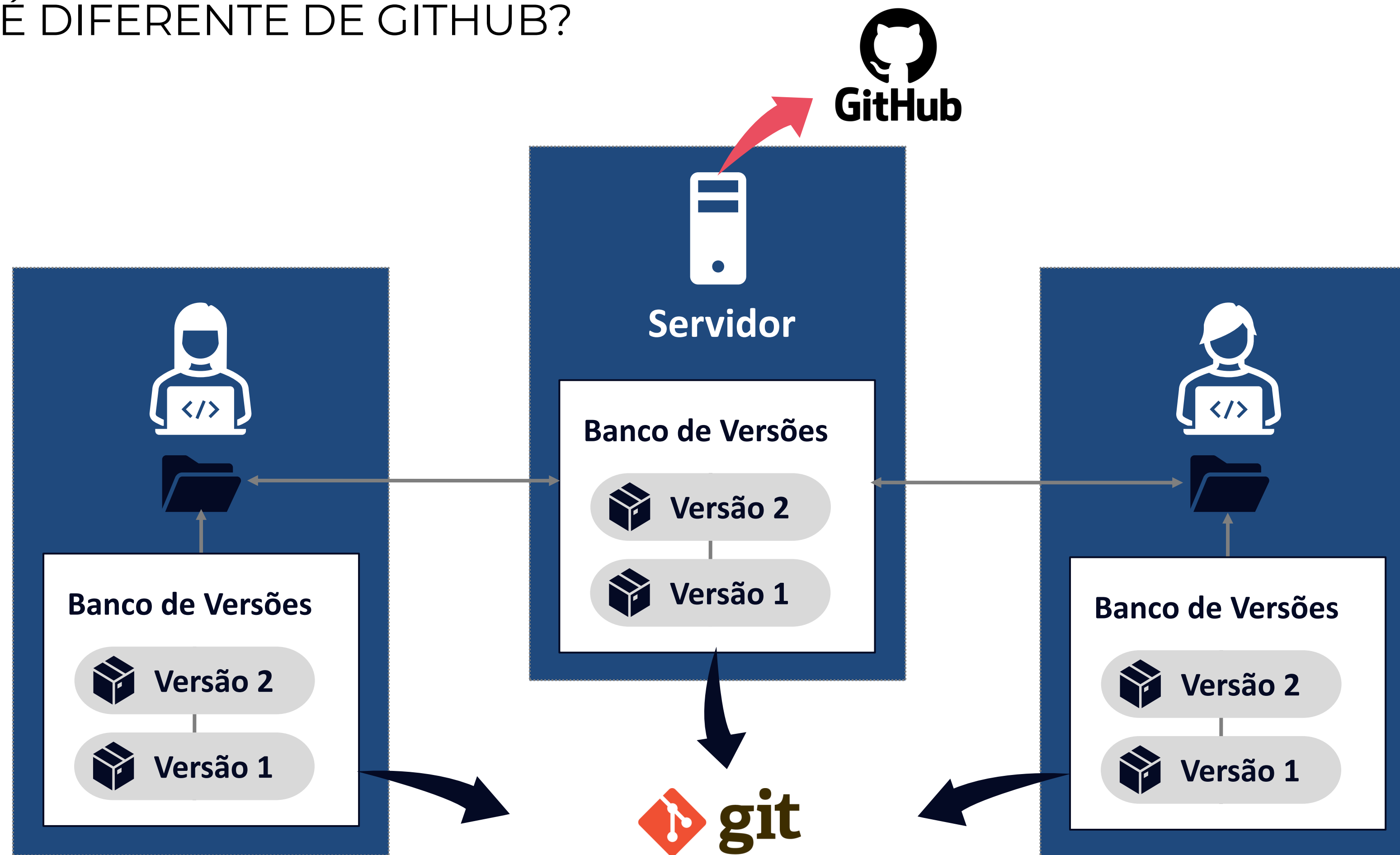


Mascote “Octocat”.



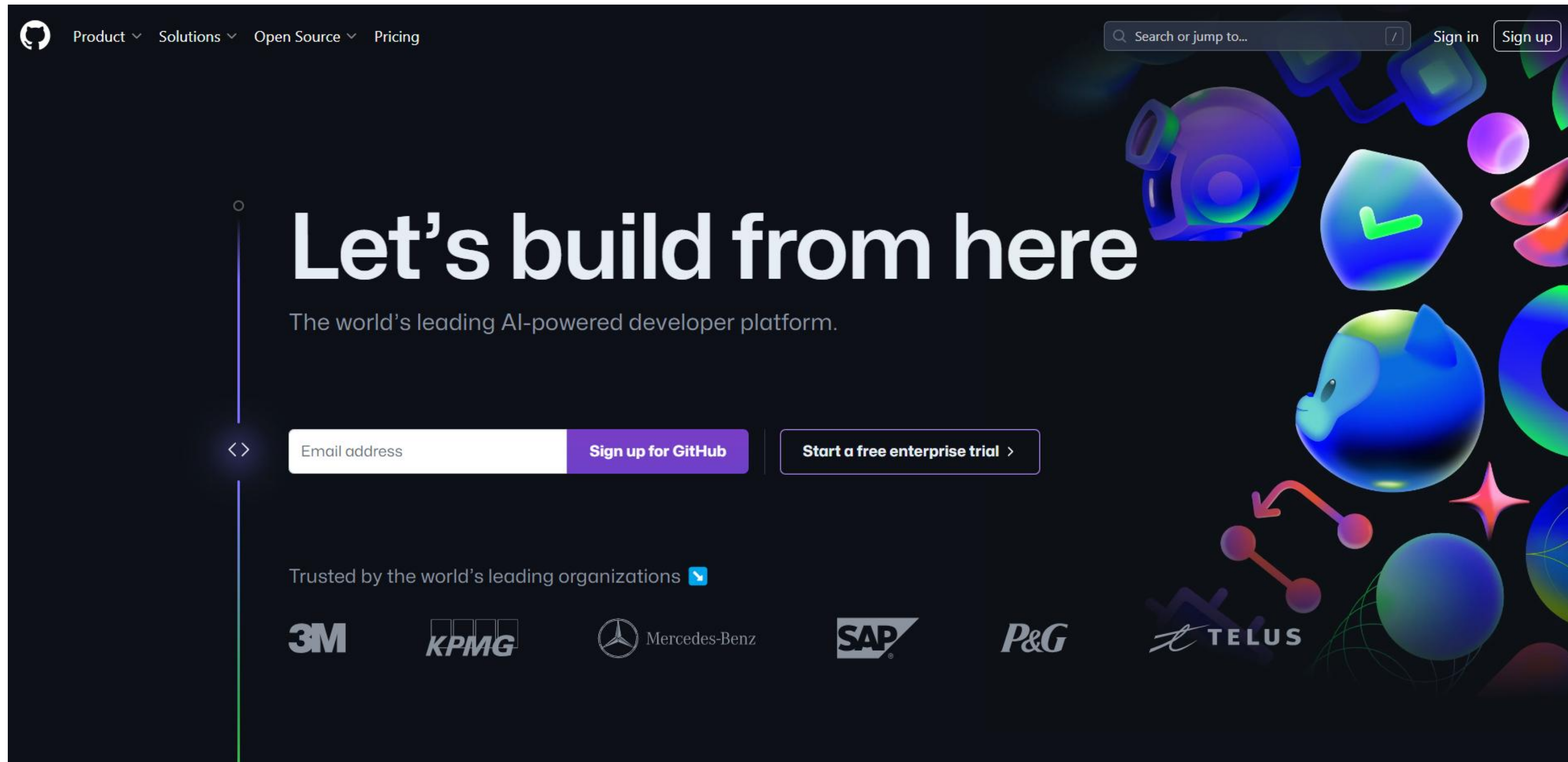
VERSIONAMENTO

GIT É DIFERENTE DE GITHUB?



VERSIONAMENTO

GITHUB – VAMOS CRIAR UMA CONTA



VERSIONAMENTO

GITHUB – AUTENTICAÇÃO DE 2 FATORES

Recomendado: acesse sua conta do GitHub e vá em Settings > Password and authentication > Two-factor authentication > Authenticator app

1

Leia o QR Code através do aplicativo autenticador (ex.: Microsoft Authenticator) e insira o código no GitHub;

2

Salve os códigos de recuperação;

3

Autenticação ativada! ;D





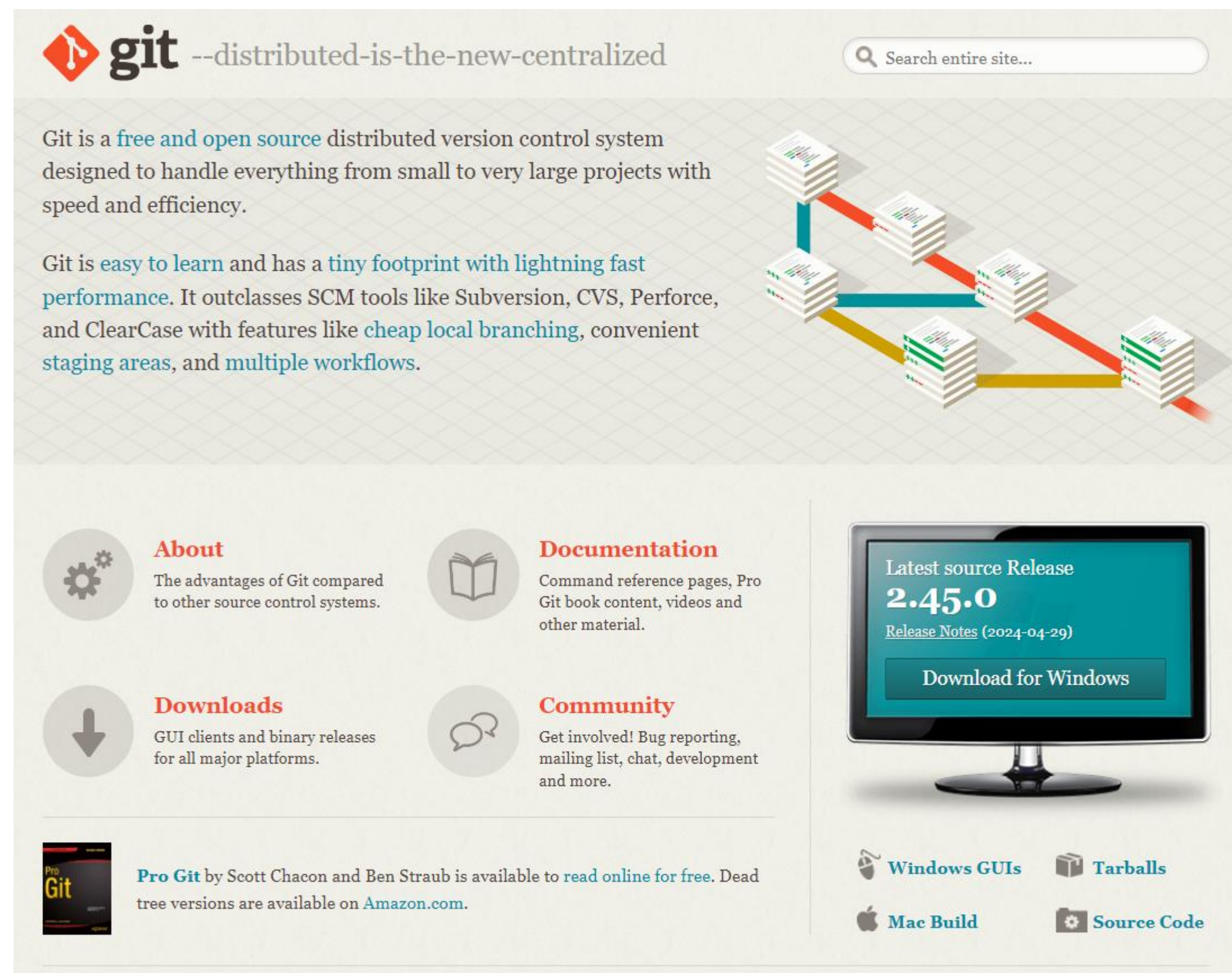
GIT – INSTALAÇÃO E CONFIGURAÇÃO

VERSIONAMENTO

GIT – INSTALAÇÃO E CONFIGURAÇÃO

<https://git-scm.com/>

Faça download da versão
compatível com o seu SO,
no nosso caso Windows



The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--distributed-is-the-new-centralized". A search bar is located in the top right corner. The main content area features two paragraphs: "Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency." and "Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**." To the right of the text is a diagram illustrating a distributed version control system with multiple repositories connected by a network. Below the main text are four sections: "About" (The advantages of Git compared to other source control systems.), "Documentation" (Command reference pages, Pro Git book content, videos and other material.), "Downloads" (GUI clients and binary releases for all major platforms.), and "Community" (Get involved! Bug reporting, mailing list, chat, development and more.). On the right side, there is a large monitor displaying the latest source release "2.45.0" and a button "Download for Windows". At the bottom, there are links for "Windows GUIs", "Mac Build", "Tarballs", and "Source Code".

git --distributed-is-the-new-centralized

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.45.0
Release Notes (2024-04-29)
Download for Windows

Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Windows GUIs
Mac Build
Tarballs
Source Code

VERSIONAMENTO

GIT – FLUXO DE TRABALHO BÁSICO

- 1 Acesse < <https://git-scm.com/download/win> >;
- 2 Faça o download do instalador e execute;
- 3 Aceite a licença e clique em “Next”, e siga configurando como desejar¹ e clicando em “Next”;
- 4 Finalize clicando em “Install”, e “Finish”.

¹Em "Select Components", deixe as opções “Git Bash Here” e “Git GUI Here” marcadas.

VERSIONAMENTO

GIT – INSTALAÇÃO E CONFIGURAÇÃO

```
$ git config --list
```

1 Configurando seu nome de usuário e e-mail (globalmente):

```
$ git config --global user.name "Nome Sobrenome"  
$ git config --global user.email seuemail@email.com
```

2 Configurando o nome da Branch Padrão:

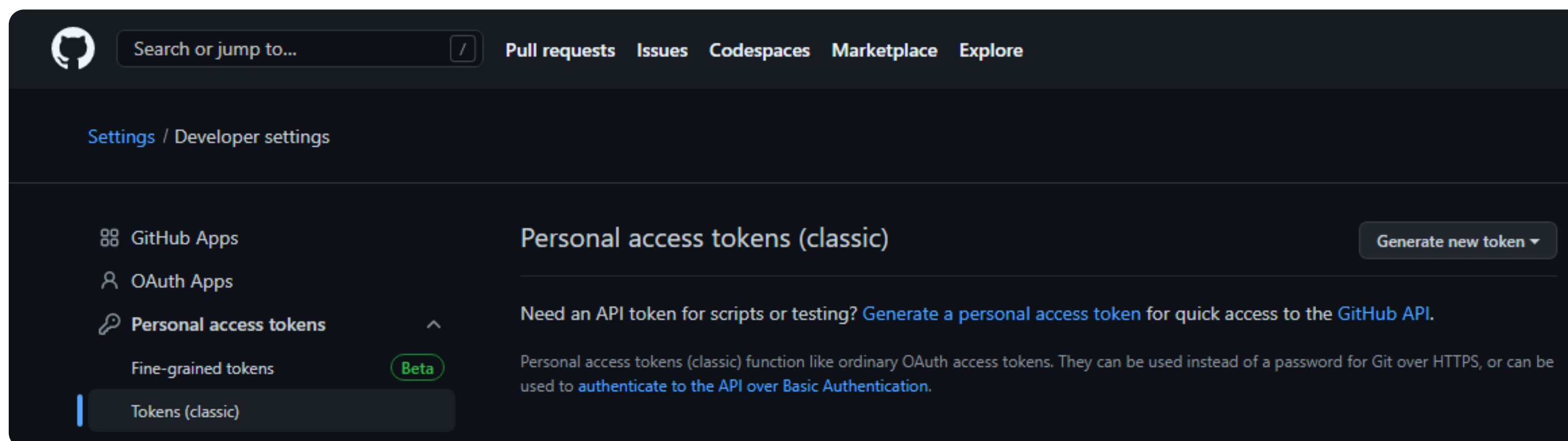
```
$ git config --global init.defaultBranch main
```

VERSIONAMENTO

GIT – AUTENTICAÇÃO VIA TOKEN



Para gerar um Token, acesse sua conta no GitHub, e no menu superior direito clique em ***Settings > Developer settings > Tokens (classic) > Generate new token.***



VERSIONAMENTO

GIT – ARMAZENANDO CREDENCIAIS

Você pode armazenar suas credenciais para reduzir o número de vezes que você deve digitar seu nome de usuário ou senha:



Salvando no cache:

```
$ git config --global credential.helper cache
```



Ou permanentemente:

```
$ git config --global credential.helper store
```

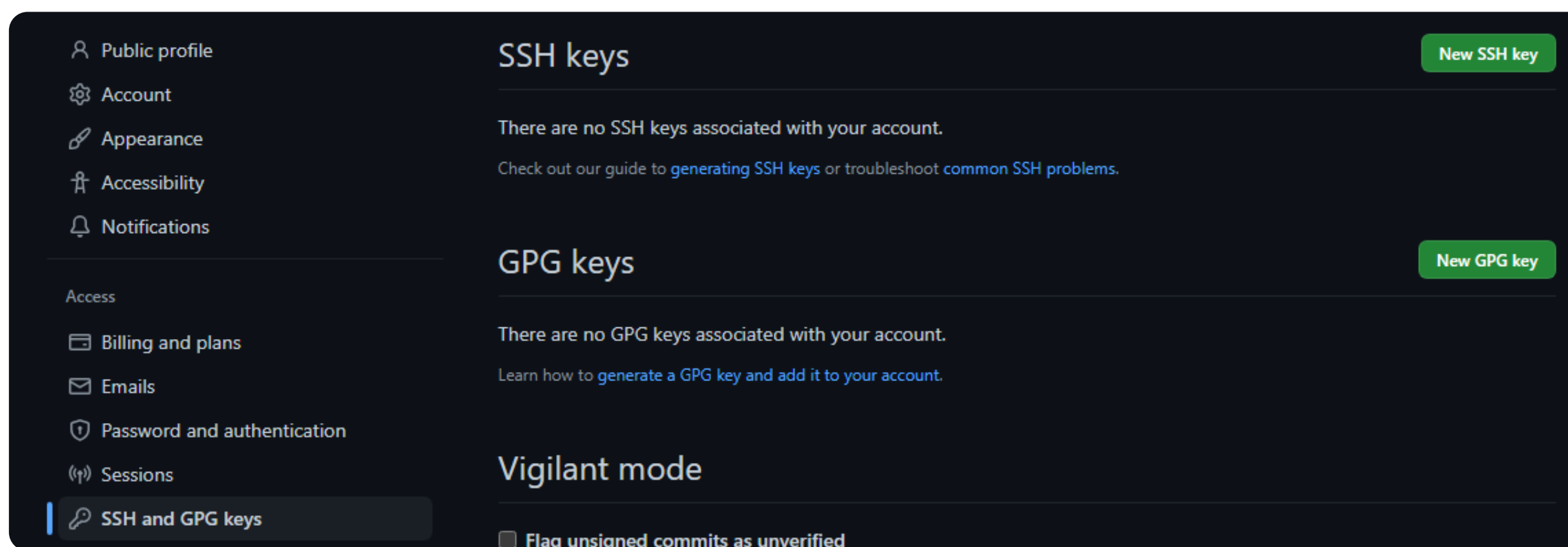
Veja mais na doc.: <https://git-scm.com/book/pt-br/v2/Git-Tools-Credential-Storage>

VERSIONAMENTO

GIT – AUTENTICAÇÃO VIA CHAVE SSH



Para adicionar uma Chave SSH, acesse sua conta no GitHub, e no menu superior direito clique em ***Settings*** > ***SSH and GPG keys*** > ***New SSH key***.





PRIMEIROS PASSOS COM GIT E GITHUB

VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Criando e clonando repositórios

Existem duas formas de obter um repositório Git na sua máquina:

- 1** Transformando um diretório local que não está sob controle de versão, num repositório Git;
- 2** Clonando um repositório Git existente.



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Criando um repositório local

Acesse a pasta que deseja transformar em um repositório Git pelo terminal ou clique no atalho em “Git Bash Here”:

- 1 Inicialize um repositório Git no diretório escolhido:

```
$ git init
```

- 2 Conecte o repositório local com o repositório remoto:

```
$ git remote add origin  
https://github.com/username/nome-do-repositorio.git
```

VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Clonando um repositório

Para clonar um repositório no Git, acesse seu repositório no GitHub e siga os próximos passos:

- 1 Em “Code”, copie o código HTTPS ou SSH (a depender de como autenticou sua conta) do repositório no GitHub;
- 2 Abra o GitBash, insira o comando `git clone` e cole o conteúdo copiado para cloná-lo:

```
$ git clone https://github.com/username/nome-do-repositorio
```


VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Criando um repositório remoto

Acesse a sua conta do GitHub, clique no “+” no canto superior direito, e em “New repository”:

- 1 Insira um nome (obrigatório), e a descrição (opcional);
- 2 Coloque uma breve descrição sobre o projeto, essa etapa é opcional;
- 3 Defina se o acesso será público ou privado;



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Criando um repositório remoto

Acesse a sua conta do GitHub, clique no “+” no canto superior direito, e em “New repository”:

4

Escolha como deseja inicializar seu repositório (se quiser vazio, deixe as opções desmarcadas)

5

Clique em “Create repository”, e pronto!



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Salvando alterações no repositório local

1) Como criar um commit:

- 1 Adicione o conteúdo que deseja inserir no commit:

```
$ git add
```

- 2 Crie um commit e adicione uma mensagem descritiva:

```
$ git commit -m "message"
```

VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Desfazendo alterações no repositório local

1) Como alterar a mensagem do último commit:

```
$ git commit --amend
```

Alterando a mensagem sem abrir o editor:

```
$ git commit --amend -m "nova mensagem"
```

VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Desfazendo alterações no repositório local

2) Como desfazer um commit:

```
$ git reset
```

```
$ git reset --soft (mantém as alterações no modo stage)
```

```
$ git reset --mixed (retira as alterações do modo stage)
```

```
$ git reset --hard (descarta mudanças e volta ao anterior)
```

VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Enviando alterações para o repositório remoto

Como enviar as alterações do repositório local para o remoto:

```
$ git push
```

“Puxar” as alterações do repositório remoto para o local (busca e mescla).

```
$ git pull
```

VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Trabalhando com branches

De maneira simplista, uma Branch (em tradução, “Ramo”), é uma ramificação do seu projeto.



É um ponteiro móvel para um commit no histórico do repositório;



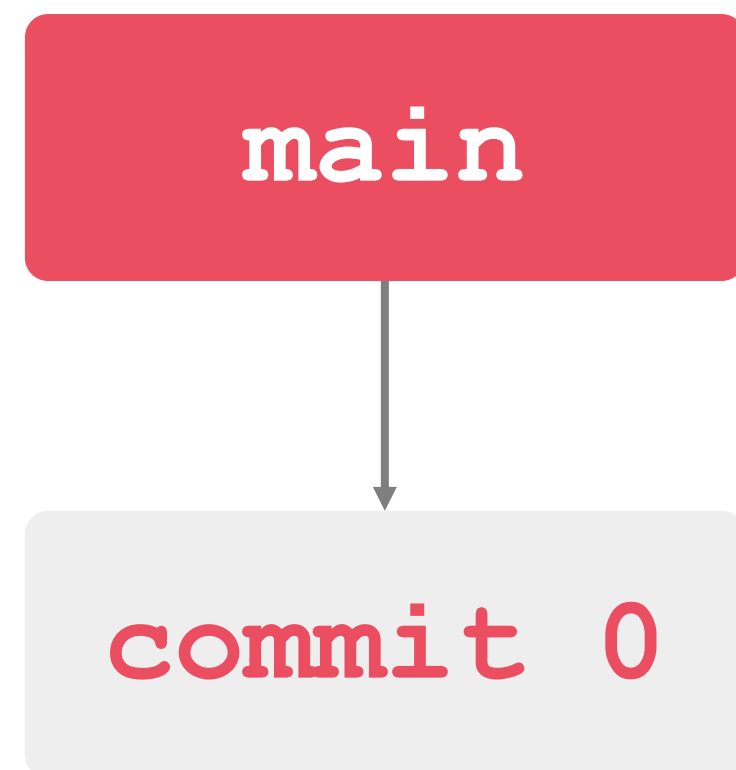
Quando você cria uma nova Branch a partir de outra existente, a nova se inicia apontando para o mesmo commit da Branch que estava quando foi criada.



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

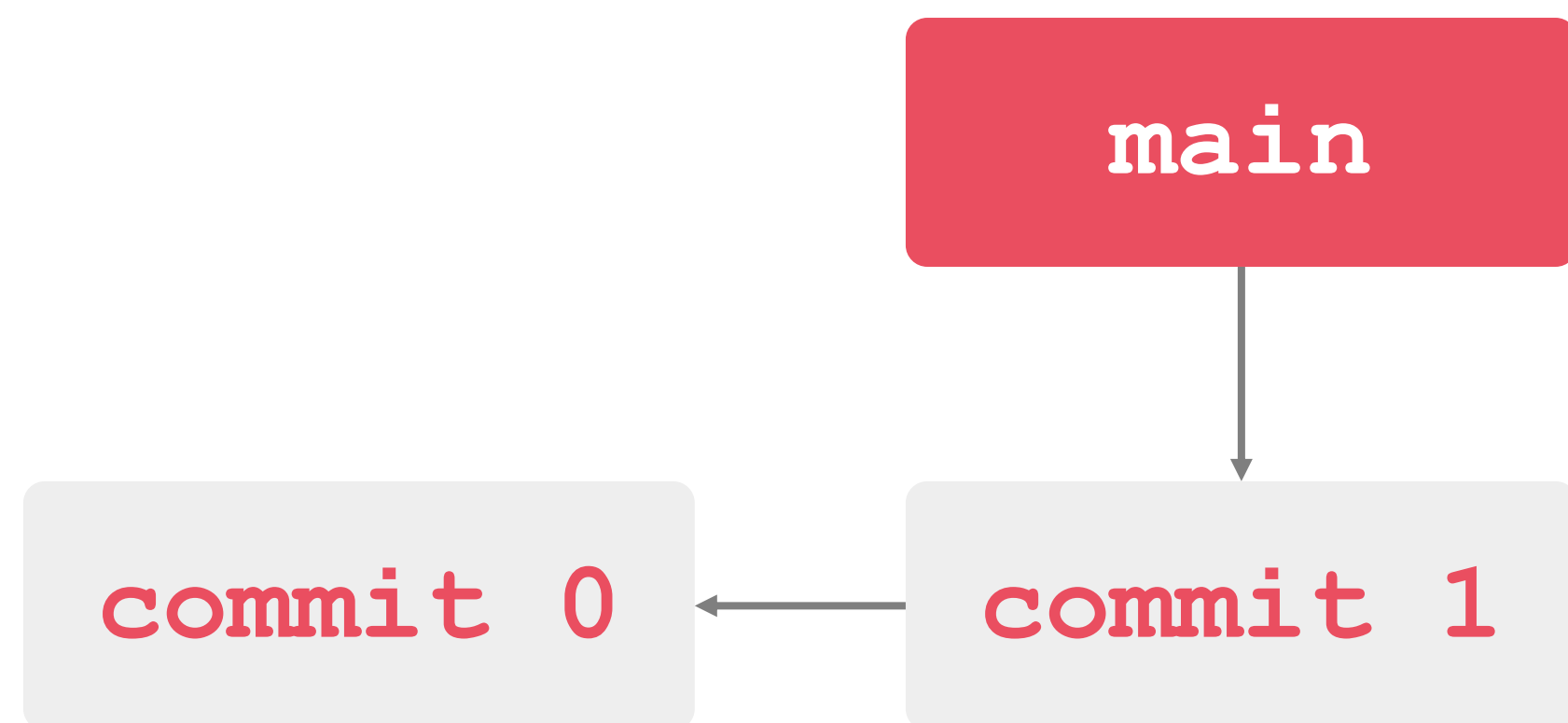
Trabalhando com branches



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

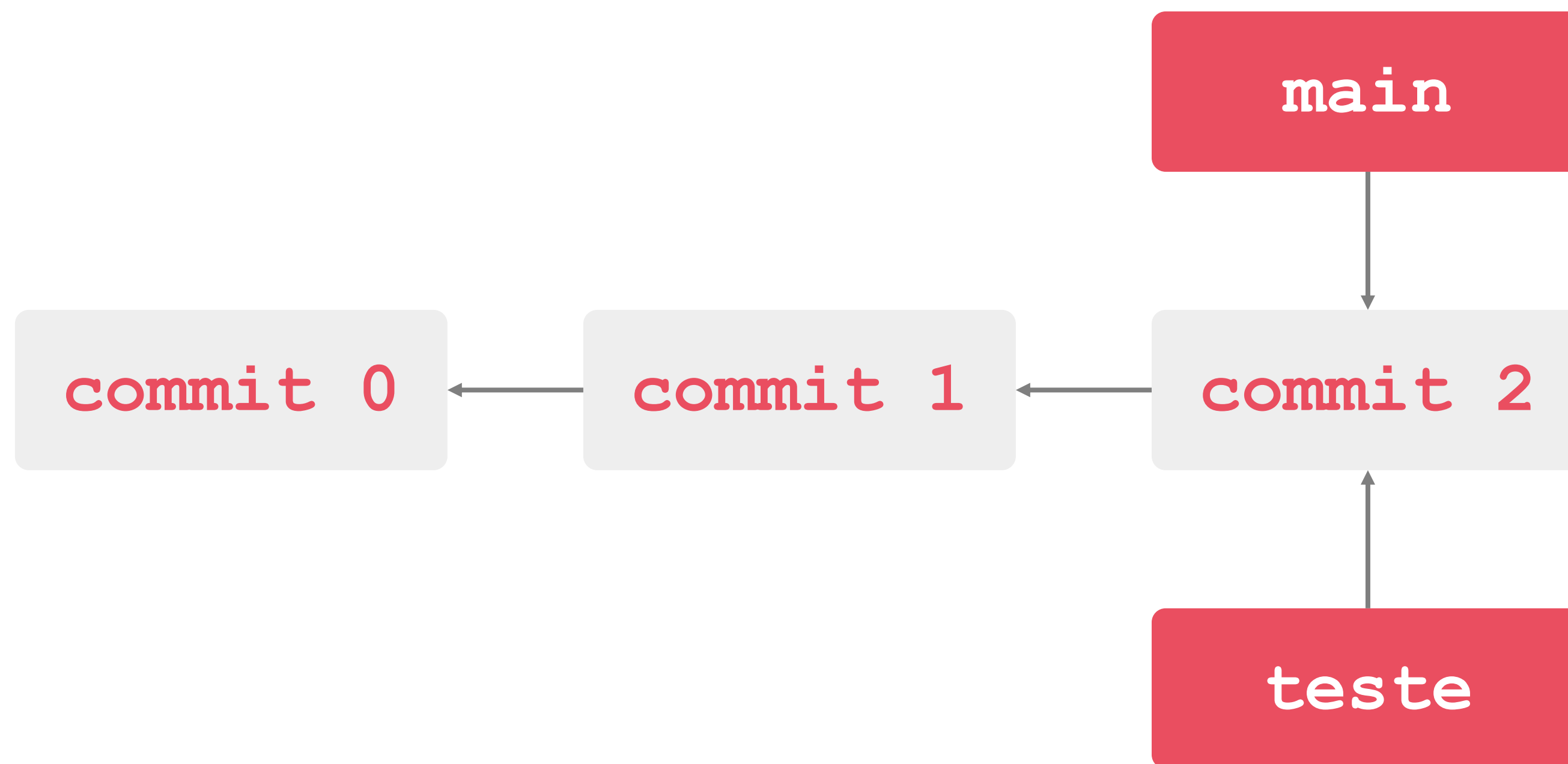
Trabalhando com branches



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

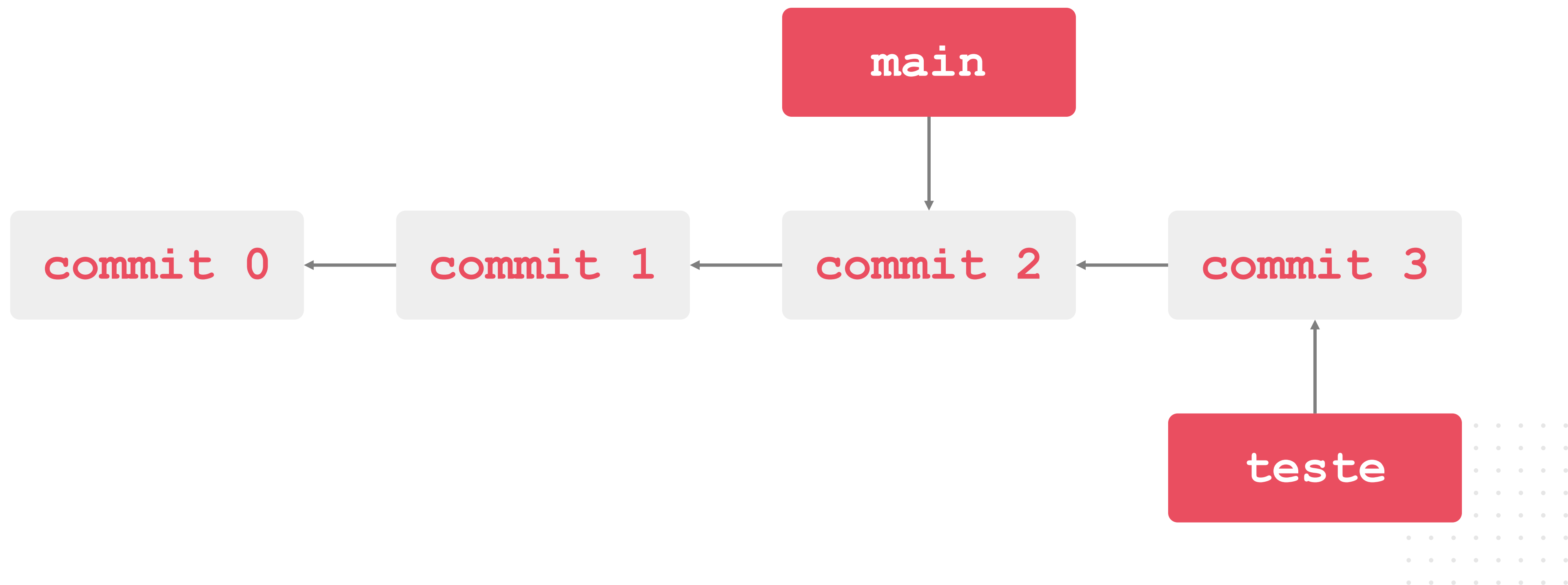
Trabalhando com branches



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Trabalhando com branches



VERSIONAMENTO

PRIMEIROS PASSOS COM GIT E GITHUB

Trabalhando com branches

```
$ git branch
```

- Trocar de Branch e criar uma nova:

```
$ git checkout -b nova-branch
```

- Deletar uma Branch

```
$ git branch -d nome-da-branch
```

- Ver o último commit de cada Branch:

```
$ git branch -v
```

VERSIONAMENTO

GIT – CONFIGURAÇÃO BÁSICA

Git Config -- User.Name

Comando para adicionar uma identificação ao repositório.

```
$ git config --global user.name "Fulano da Silva"
```

Git Config -- User.Email

Comando para adicionar uma identificação ao repositório.

```
$ git config --global user.email fulanodasilva.git@gmail.com
```



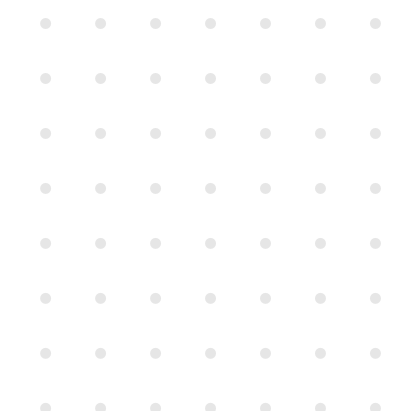
VERSIONAMENTO

GIT – CONFIGURAÇÃO BÁSICA

Git Config -- List

Comando para listar as configurações.

```
$ git config --list
```



VERSIONAMENTO

GIT – PRINCIPAIS COMANDOS

Git Init

Comando para inicializar um repositório git local

```
$ git init
```

Git Remote

Comando para adicionar uma conexão remota ao repositório local.

```
$ git remote add origin https://github.com/gittower/example.git
```



VERSIONAMENTO

GIT – PRINCIPAIS COMANDOS

Git Clone

Comando para baixar o código-fonte do repositório remoto.

```
$ git clone <https://link-com-o-nome-do-repositório>
```

Git Branch

Comando para ramificar o projeto. Possibilita trabalho simultâneo no mesmo projeto.

```
$ git branch <nome-da-branch>
```



VERSIONAMENTO

GIT – PRINCIPAIS COMANDOS

Git Checkout

Comando para navegar entre as branches

```
$ git checkout <nome-da-ramificação>
```

Git Status

Comando para visualizar as alterações em relação a master.

```
$ git status
```



VERSIONAMENTO

GIT – PRINCIPAIS COMANDOS

Git Diff

Comando para visualizar as alterações realizadas.

```
$ git diff
```

Git Add

Comando para adicionar alteração de um arquivo no commit.

```
$ git add <arquivo>
```



VERSIONAMENTO

GIT – PRINCIPAIS COMANDOS

Git Commit

Comando para criar um ponto de verificação das alterações.

```
$ git commit -m "mensagem explicando a mudança no código"
```

Git Push

Comando para enviar as alterações ao servidor remoto.

```
$ git push <remote> <nome-do-branch>
```



VERSIONAMENTO

GIT – PRINCIPAIS COMANDOS

Git Pull

O git pull é usado para obter atualizações do repositório remoto.

```
$ git pull <remote>
```

Git Fetch

Comando para obter atualização, diferente do Pull ele não altera o repositório local (Merge).

```
$ git fetch <remote>
```

