

# IBM1741 – PROJETO FRONT-END

Professor Me. Vanderson Bossi



# APRESENTAÇÃO DO PROFESSOR E ALUNOS

# Apresentação do Professor

## Quem eu sou?

✓ Nome: Prof. Me. Vanderson Bossi

### Perfil

- ✓ Doutorando e Mestre em Ensino de Ciências e Matemática;
- ✓ Especialista em Engenharia de Software.



### Skills & Habilidades

- ✓ Desenvolvedor Front-end e Back-end
- ✓ SQL/Oracle, DevOps e Cloud

### Certificações:



# PLANO DE AULAS

# Ementa

Desenvolvimento de um site ou sistema responsivo para consumir dados de uma API remota ou através de mocks (objetos que simulam o comportamento de objetos reais). Aplicação do projeto usando conceitos de Design Thinking e metodologias ágeis, como Scrum. Git: definições, conceitos, versionamento de código. Utilização de ferramentas de git (GitHub, Gitlab, Bitbucket ou similar). Elaboração de portfólio.

# Objetivos

1. Desenvolver aplicação para consumo/envio de dados de/para APIs web;
2. Implementar ambiente de desenvolvimento e teste de Front-End;
3. Utilizar metodologia e práticas de desenvolvimento ágil;
4. Controlar versões e atualização coletiva do código da aplicação de forma automatizada;

# Plano de Aulas

Aula	Dia	Tópico	Atividades	Preparação Prévia do Aluno
01	08/08/2023	Apresentação da Disciplina; Sistema de versionamento Git (Conceitos iniciais, criação da conta, conceitos de repositório)	Prática em Laboratório	DevOps – <a href="#">Capítulo 3</a>
02	09/08/2023	Sistema de versionamento Git: conceitos de repositório (local e remoto), principais comandos (Clone, <u>add</u> , <u>commit</u> , <u>push</u> ).	Prática em Laboratório	DevOps – <a href="#">Capítulo 4</a>
03	15/08/2023	Sistema de versionamento Git: Trabalhando em grupo principais comandos (Clone, <u>add</u> , <u>commit</u> , <u>pull</u> , <u>push</u> ).	Prática em Laboratório	DevOps – <a href="#">Capítulo 4</a> página 68
04	16/08/2023	Conceito de DevOps (CI/CD e PipeLine), <u>Git Actions</u> ; Hospedando a 1ª página com Git Pages (individual e em Grupo)	Prática em Laboratório	DevOps – <a href="#">Capítulo 1</a>
05	22/08/2023	Iniciando as definições do Projeto (Cliente/Grupos e entregas), definindo repositórios do projeto e Trello.	Prática em Laboratório	
06	23/08/2023	Definição de requisitos do projeto: requisitos funcionais e não funcionais, histórias de usuário, construção de personas e documentações a serem entregues, reuniões com cliente;	Prática em Laboratório	<a href="#">Análise e Projeto de Sistemas</a> , <a href="#">Design Thinking</a>
07	29/08/2023	<b>IBMEC DAY</b>		
08	30/08/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> ),	Prática em Laboratório	<a href="#">Análise e Projeto de Sistemas</a> , <a href="#">Design Thinking</a>

# Plano de Aulas

09	05/09/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Análise e Projeto de Sistemas, Design Thinking</a>
10	06/09/2023	<b>Entrega parcial 1:</b> Relatório da reunião com cliente e requisitos funcionais e requisitos não funcionais.	Prática em Laboratório	
11	12/09/2023	Definição de protótipos com base nos requisitos.	Prática em Laboratório	<a href="#">Análise e Projeto de Sistemas, Design Thinking</a>
12	13/09/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Análise e Projeto de Sistemas, Design Thinking</a>
13	19/09/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Análise e Projeto de Sistemas, Design Thinking</a>
14	20/09/2023	<b>Entrega parcial 2:</b> Apresentação dos primeiros protótipos + Correções da entrega 1 + relatório parcial da evolução do grupo.	Prática em Laboratório	
15	26/09/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Projetos de Sistemas Web</a>
16	27/09/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Projetos de Sistemas Web</a>



# Plano de Aulas

17	03/10/2023	Orientação e Acompanhamento do projeto e dos artefatos ( <b>Período de Avaliações Parciais AP1</b> )	Prática em Laboratório	<a href="#">Projetos de Sistemas Web</a>
18	04/10/2023	Orientação e Acompanhamento do projeto e dos artefatos ( <b>Período de Avaliações Parciais AP1</b> )	Prática em Laboratório	<a href="#">Projetos de Sistemas Web</a>
19	10/10/2023	Início do desenvolvimento funcional do projeto.	Prática em Laboratório	<a href="#">Projetos de Sistemas Web</a>
20	11/10/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Projetos de Sistemas Web</a>
21	17/10/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Projetos de Sistemas Web</a>
22	18/10/2023	<b>Eventos Tech/Engenharia</b>	Prática em Laboratório	
23	24/10/2023	<b>Entrega Parcial 3:</b> Entrega dos primeiros códigos via Git conforme requisitos priorizados + relatório	Prática em Laboratório	
24	25/10/2023	Definição de <u>Mocks</u> , plano de testes e ferramentas.	Prática em Laboratório	<a href="#">Teste de Software</a>
25	31/10/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Teste de Software</a>
26	01/11/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	<a href="#">Teste de Software</a>
27	07/11/2023	<b>Entrega Parcial 4:</b> Entrega do plano de testes e primeiros testes já funcionais + relatório	Prática em Laboratório	

# Plano de Aulas

28	08/11/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	
29	14/11/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	
30	15/11/2023	Orientação e Acompanhamento do projeto e dos artefatos (os grupos apresentaram o andamento e impedimentos no padrão <u>daily</u> )	Prática em Laboratório	
31	21/11/2023	Última Validação do Projeto (Período de AP2)	Prática em Laboratório	
32	22/11/2023	Última Validação do Projeto (Período de AP2)	Prática em Laboratório	
33	28/11/2023	Apresentação do Projeto Finalizado	Prática em Laboratório	
34	29/11/2023	Acompanhamento e definição para os próximos passos.	Prática em Laboratório	

# Controle de versão

- Gerenciamento do histórico de todas as modificações realizadas nos artefatos de um projeto.

# Controle de versão

Isto inclui:

- ✓ Documentação
- ✓ Script de testes
- ✓ Casos de testes automatizados
- ✓ Scripts de configuração e rede
- ✓ Scripts de implantação
- ✓ Criação do banco de dados
- ✓ Atualizações e inicialização
- ✓ Configuração de plataforma tecnologia
- ✓ Bibliotecas
- ✓ Ferramental
- ✓ Documentação técnica

# Como Instalar o Git

É possível instalar uma versão local do Git

Acesse para:

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

Windows <https://gitforwindows.org/>

Mac <https://sourceforge.net/projects/git-osx-installer/>

Linux

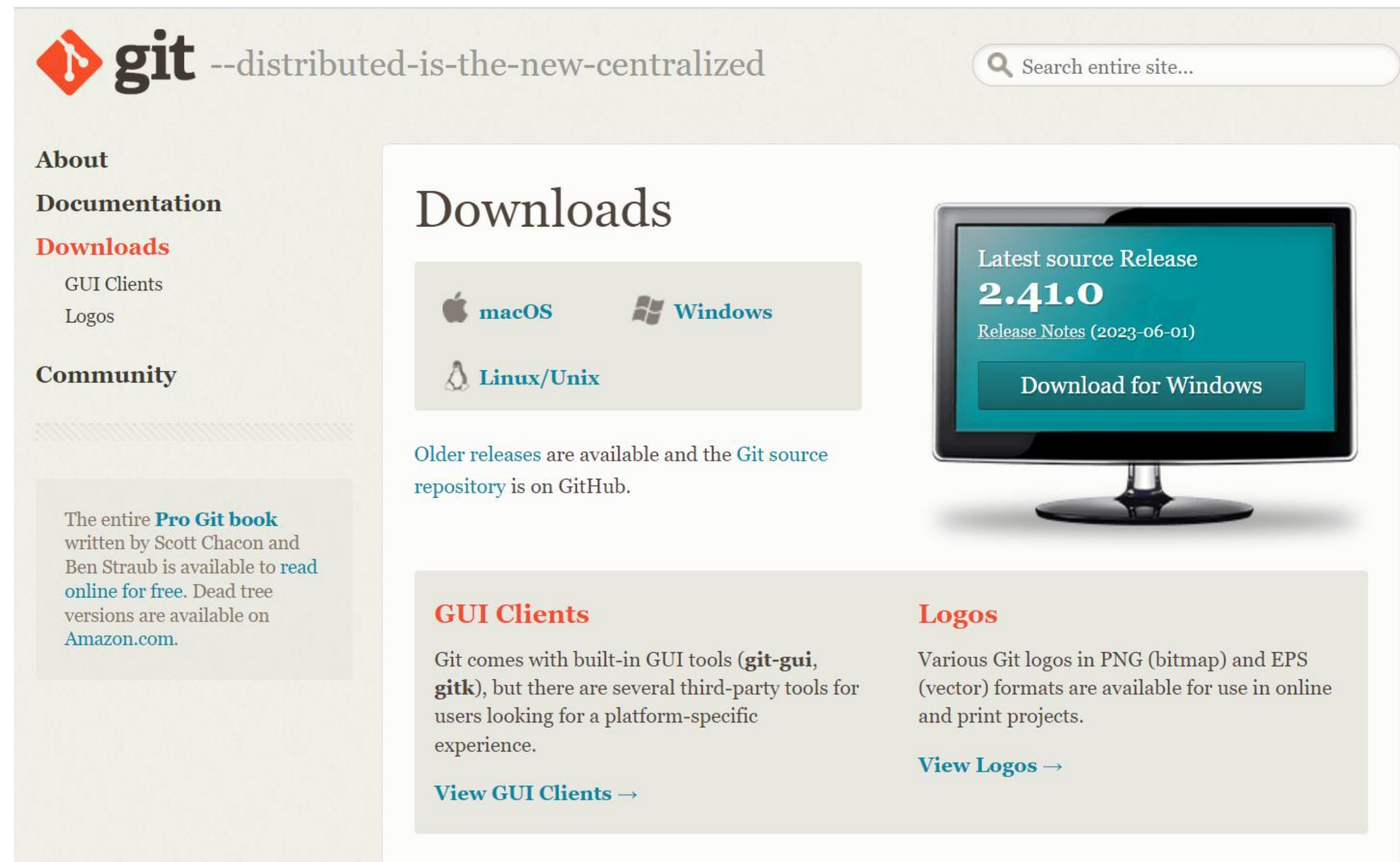
Para o Fedora `yum install git-core`

Para o Debian `apt-get install git`

Fonte: <http://comandosgit.github.io/>

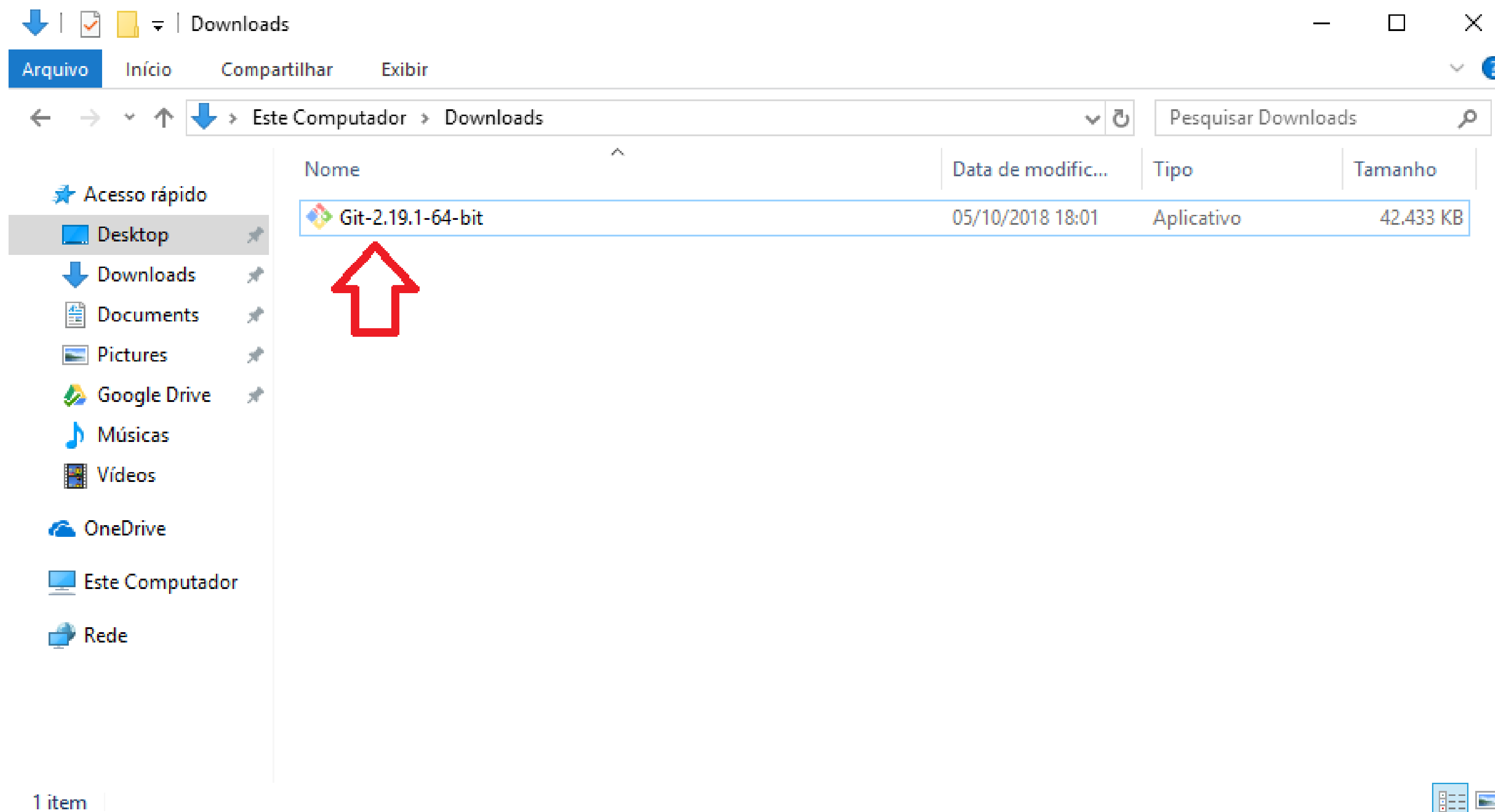
# Exemplo de instalação no Windows

Faça o download do Instalador



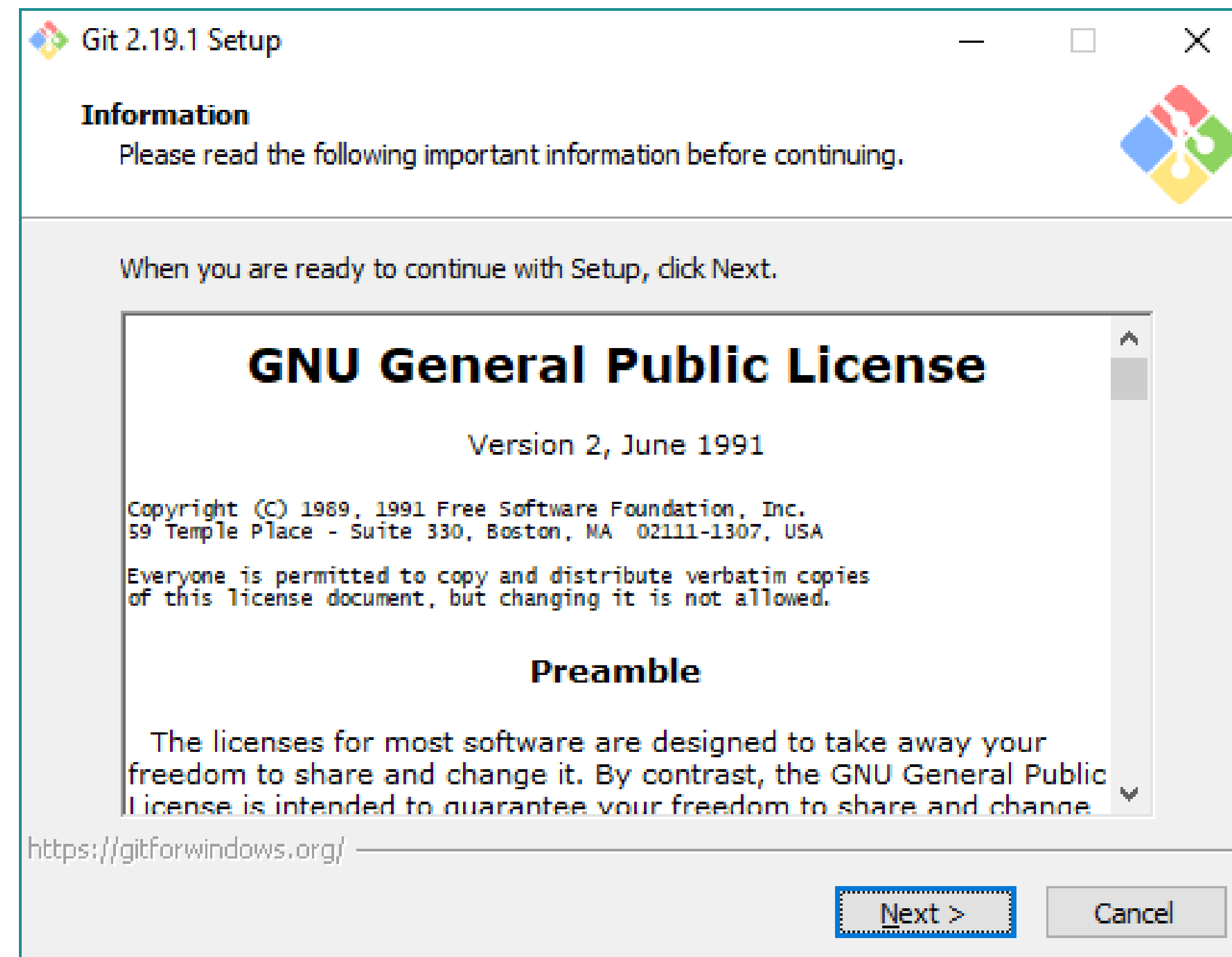
# Exemplo de instalação no Windows

Execute o Instalador após o download



# Exemplo de instalação no Windows

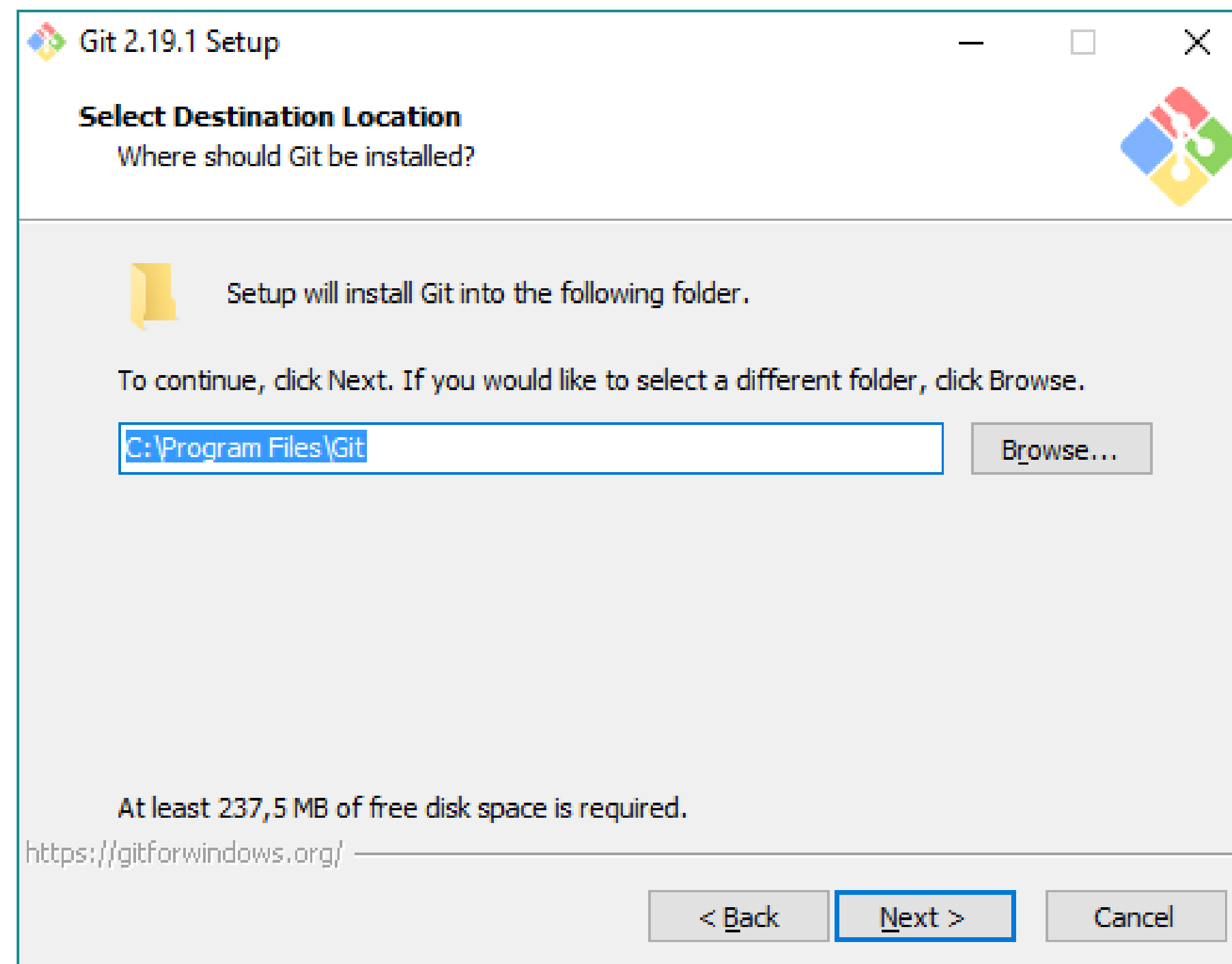
A primeira tela apresenta os termos da licença clique em next:





# Exemplo de instalação no Windows

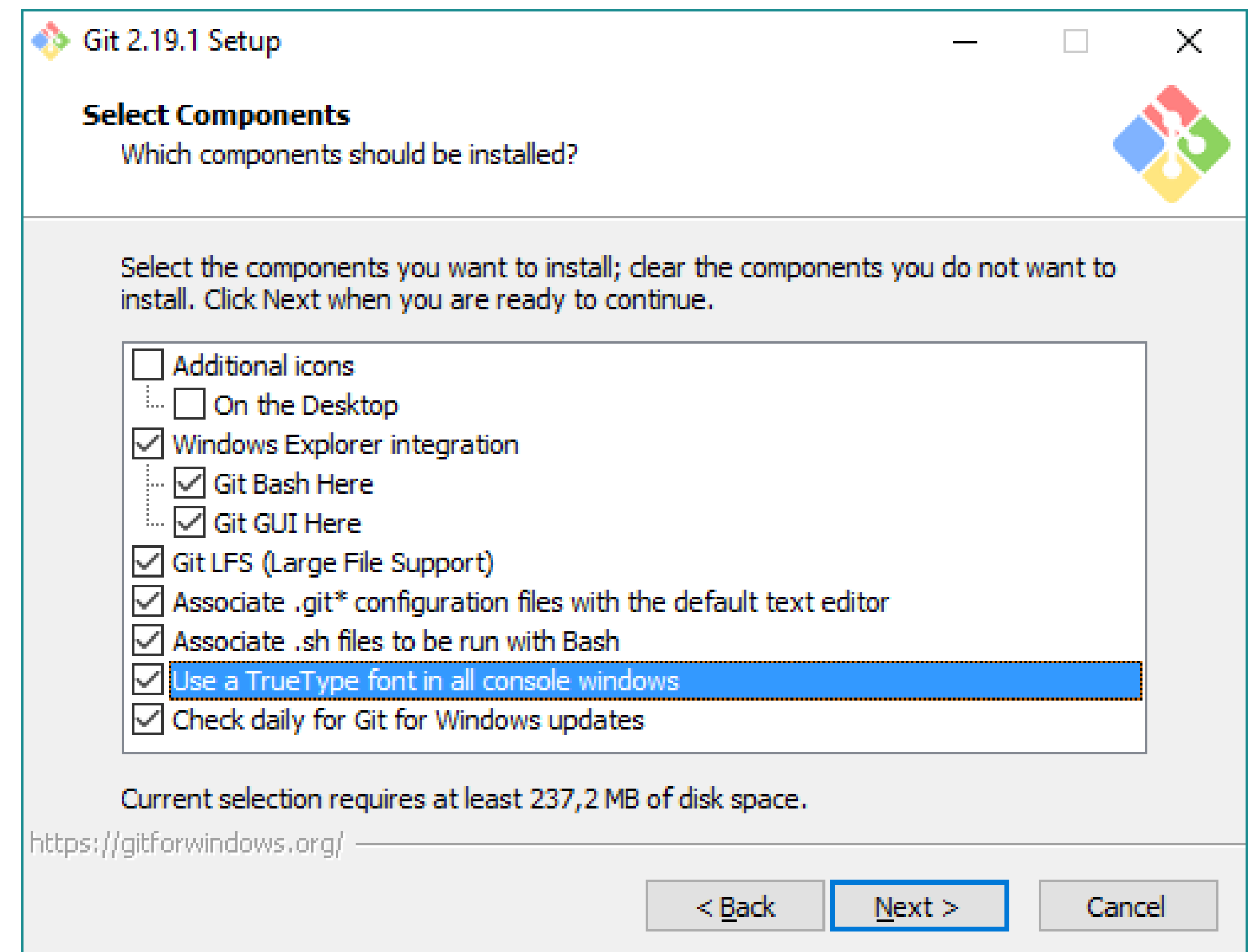
A próxima tela apresenta o local de Instalação, pode ser alterado caso deseje.



# Exemplo de instalação no Windows

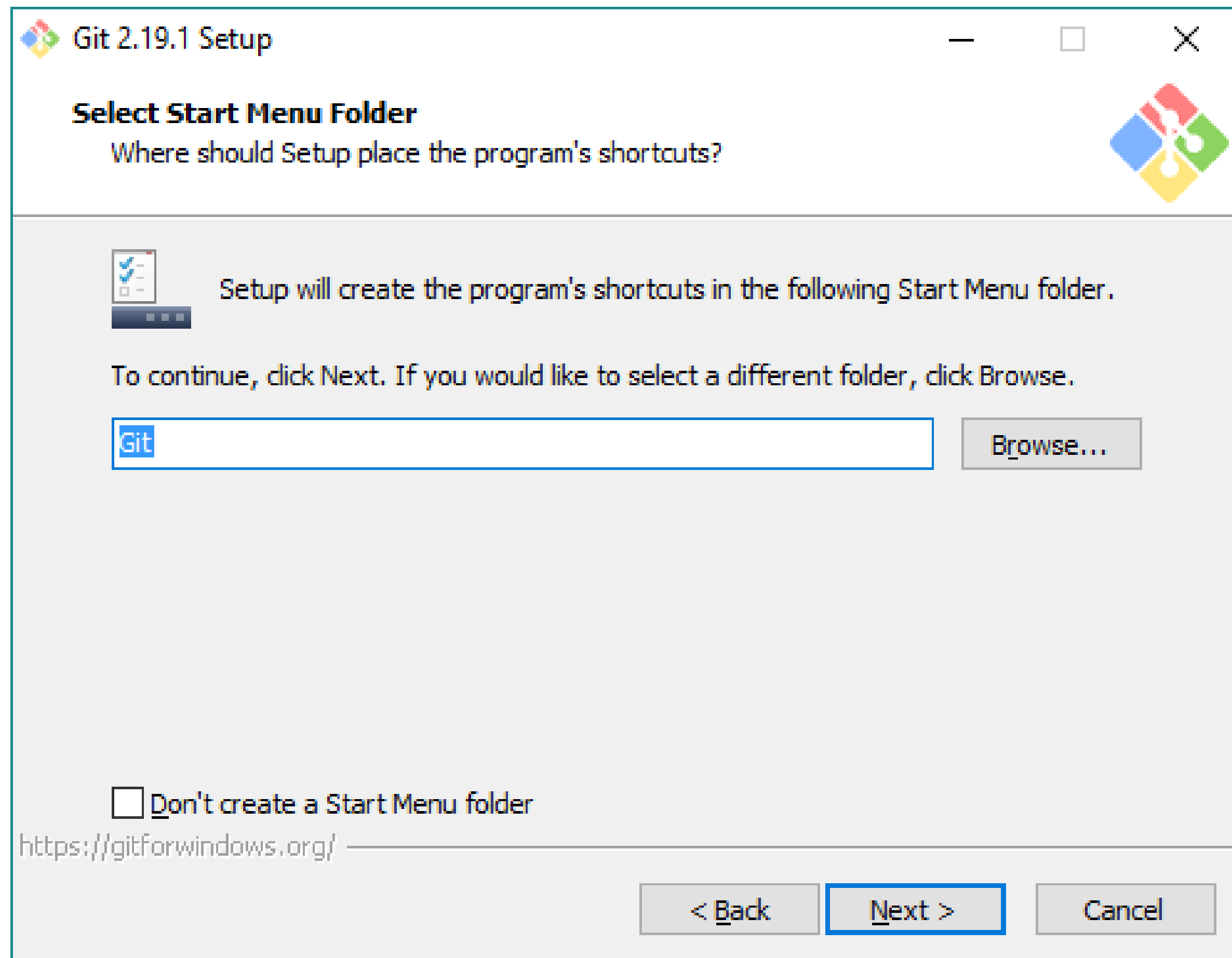
Desmarquei apenas adicionar ícones no Desktop.  
Os demais componentes selecionamos todos:

- ✓ Integração com Windows Explorer
- ✓ Suporte a arquivos grandes
- ✓ Associação de arquivos, e editor de texto padrão
- ✓ Permitir que o prompt de comandos seja colorido para as opções do git
- ✓ Verificação diária de atualização.
- ✓ Escolha os componentes de sua preferência e clique em Next.



# Exemplo de instalação no Windows

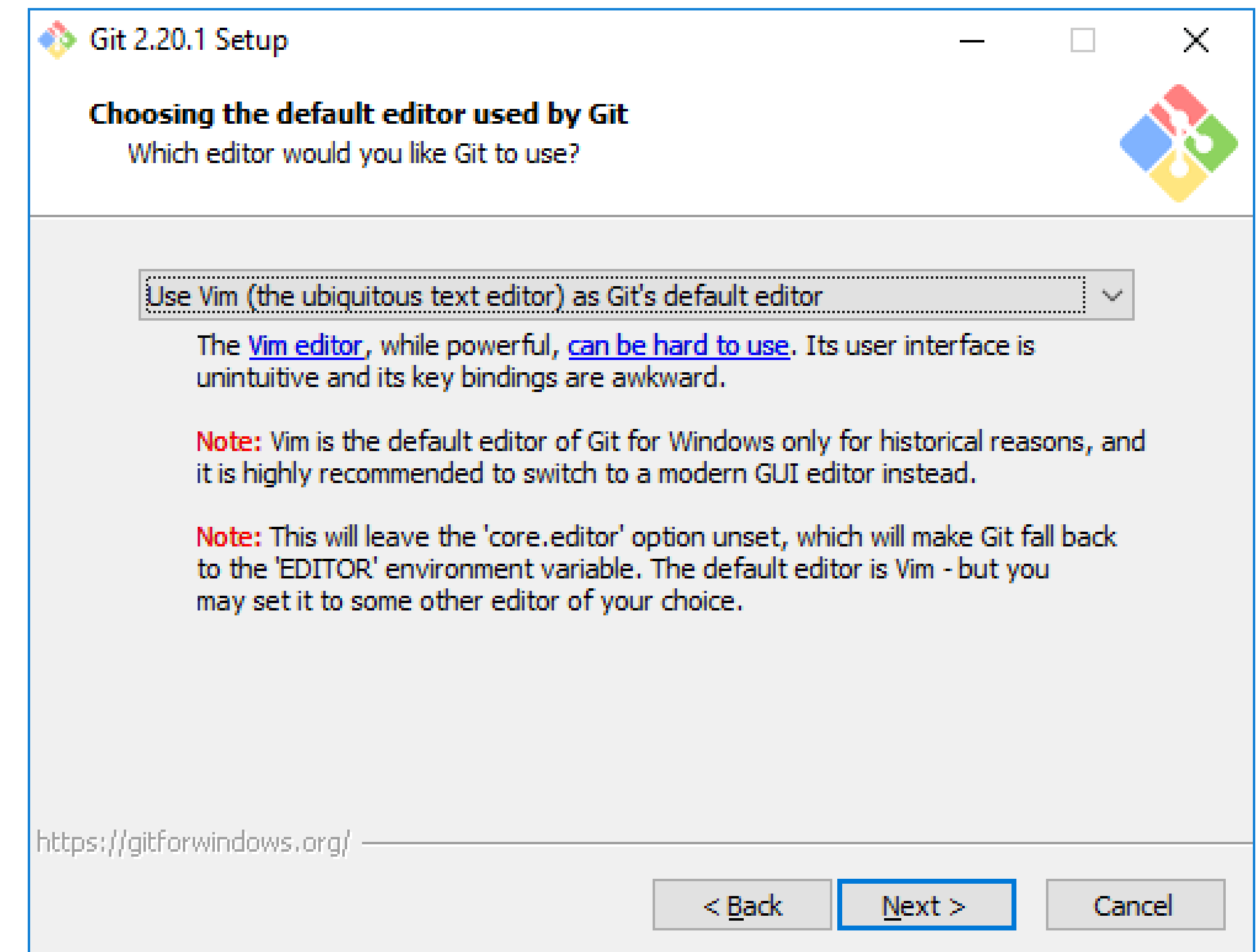
Clique em Next para continuar a instalação.



# Exemplo de instalação no Windows

Na sequência o instalador pede para selecionarmos o editor de texto que deve ser utilizado para editar os conflitos que por ventura acontecerem.

Você pode alterar para Notepad++, Sublime, Atom, VS Code, ou outro editor de sua preferência..



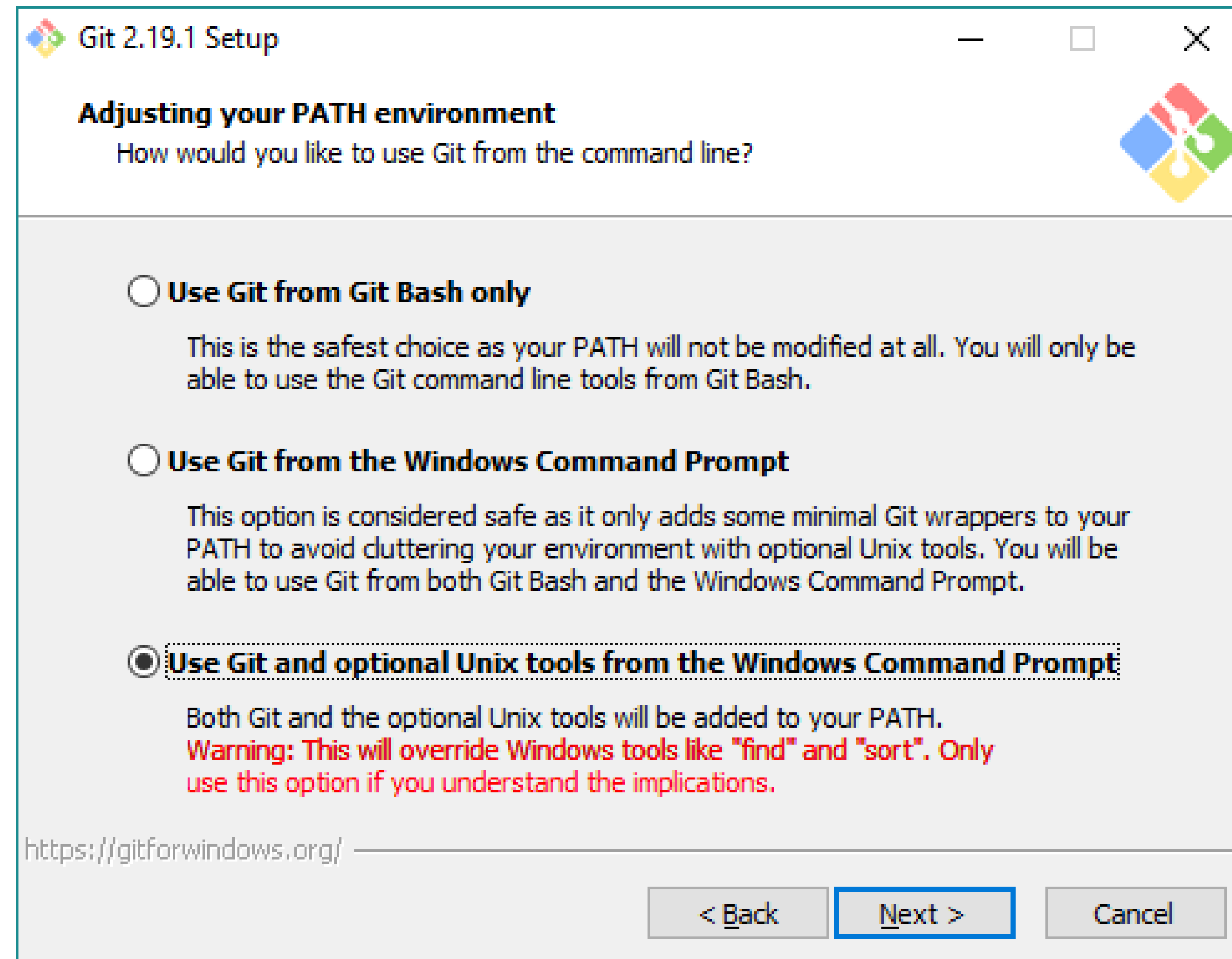
# Exemplo de instalação no Windows

Na próxima tela o instalador do GIT nos pergunta se queremos usar os comandos do git:

- somente no prompt de comandos do próprio git (chamado de Git Bash), neste caso ele não vai alterar a variável de ambiente PATH.
- no prompt do Windows (Windows Command Prompt), neste caso a variável de ambiente PATH será alterada para incluir o caminho de onde está o executável git.exe.
- no prompt do Windows + comandos utilitários do linux. **Vamos selecionar esta opção**, porque o instalador traz para o Windows alguns comandos do Linux, como cat, ls, find, etc. Neste caso a variável de ambiente PATH será alterada para incluir o caminho do executável git.exe e dos executáveis de cada comando utilitário do linux.

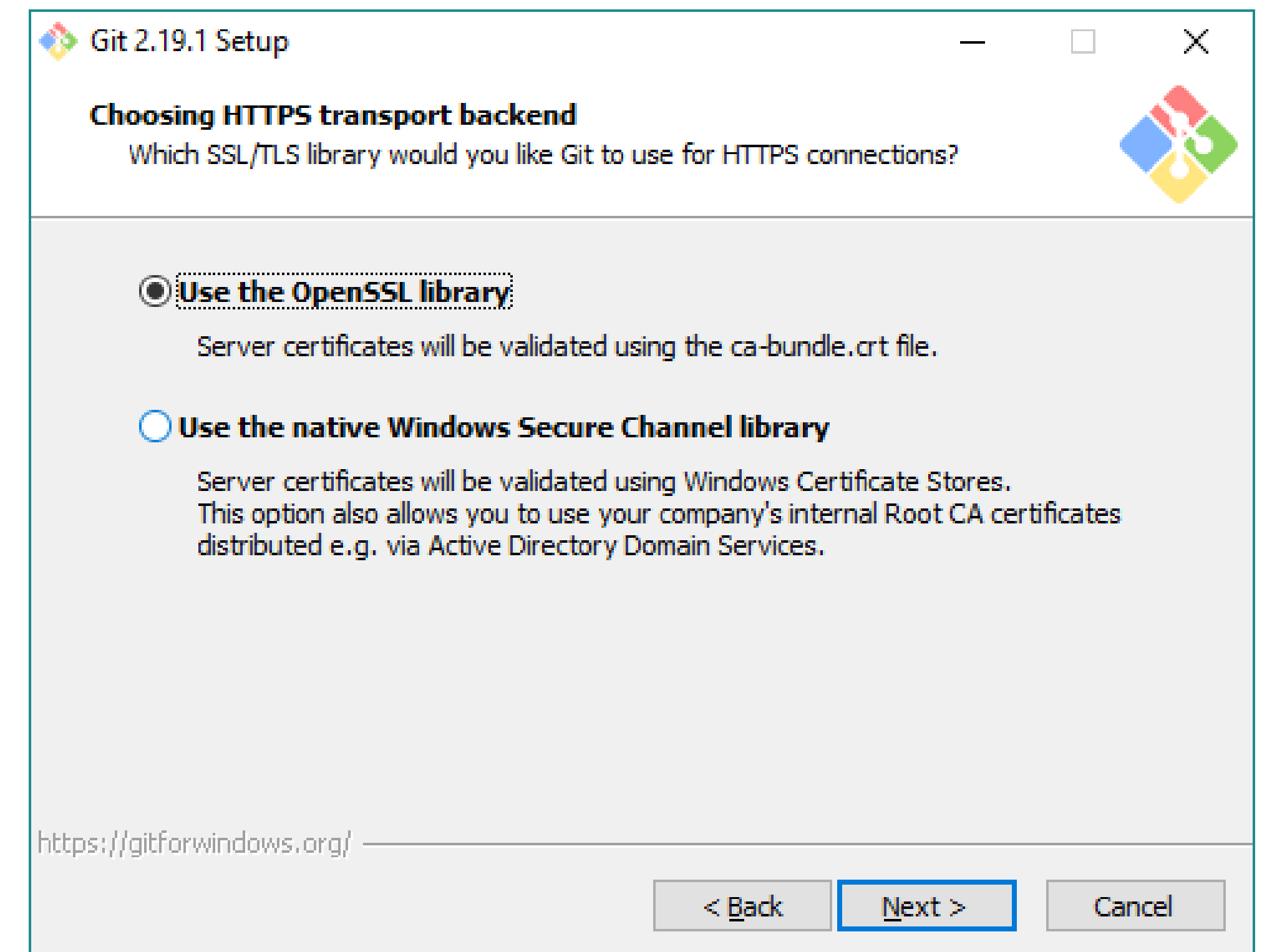
# Exemplo de instalação no Windows

Veja a seleção abaixo:



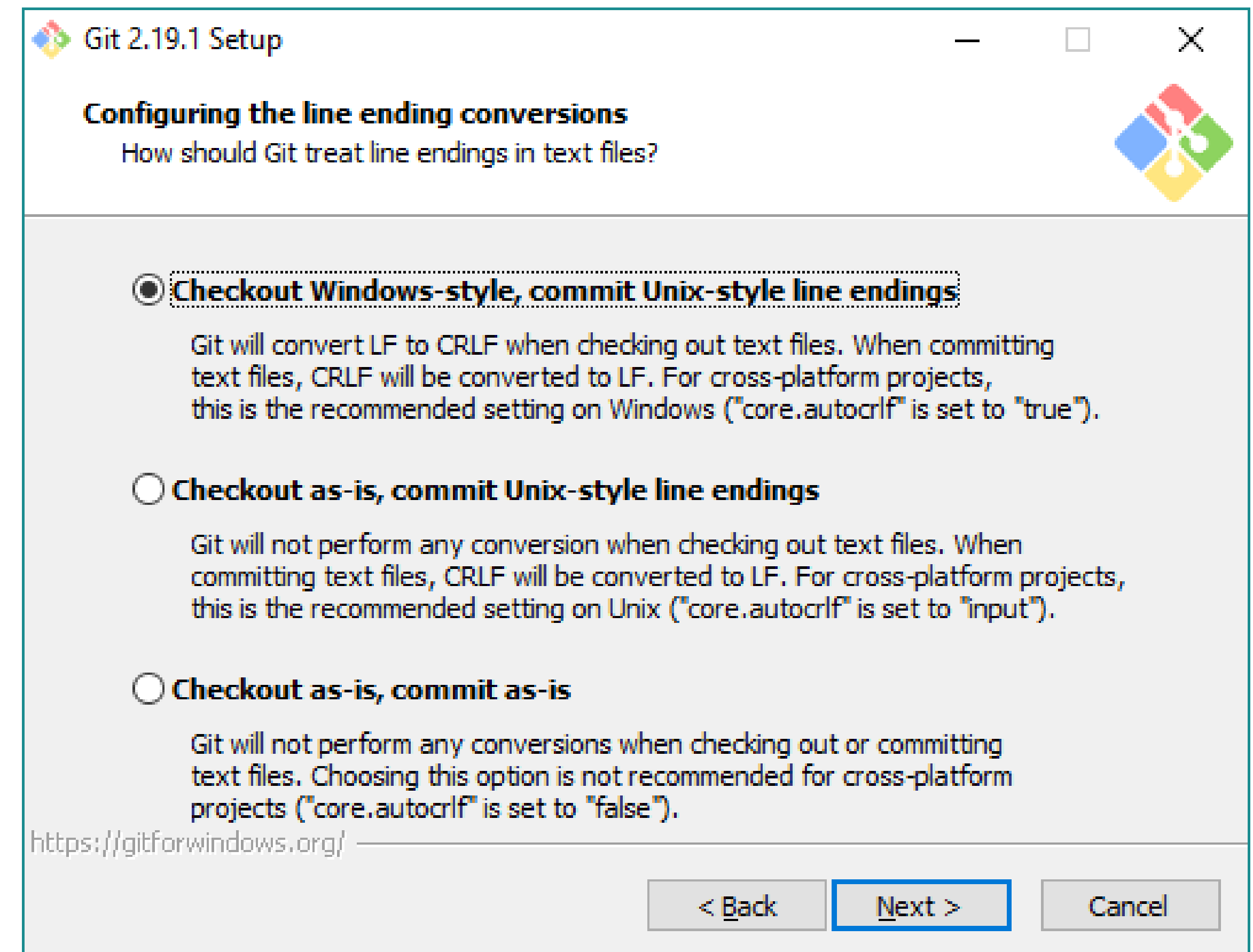
# Exemplo de instalação no Windows

Nesta tela o instalador oferece a opção de escolher a biblioteca de validação de chaves de segurança SSL. Vamos selecionar a OpenSSL, que é compatível com outras plataformas. Depois clique em Next.



# Exemplo de instalação no windows

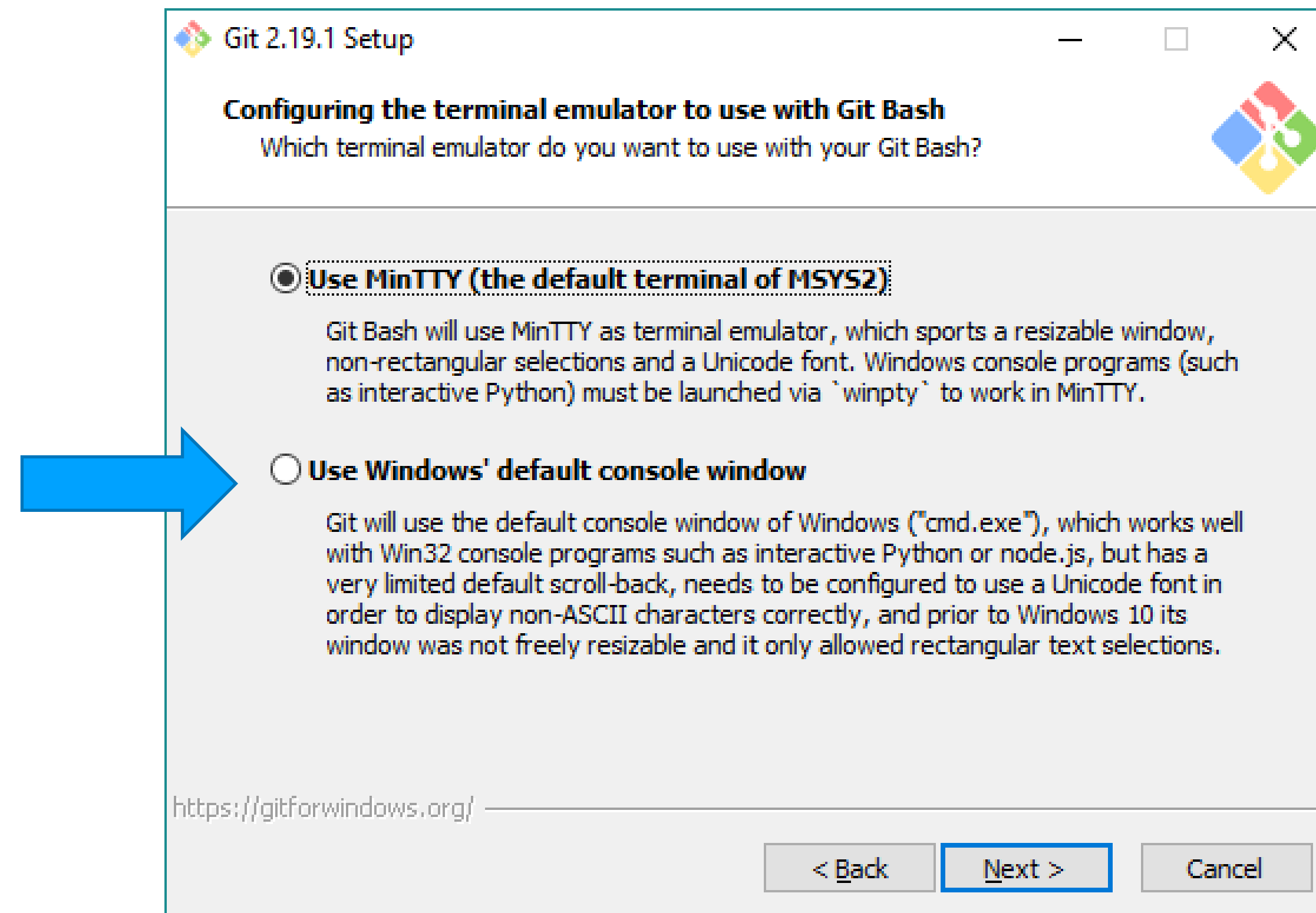
Nesta tela selecionamos a primeira opção assim evitando incompatibilidade com outras plataformas.





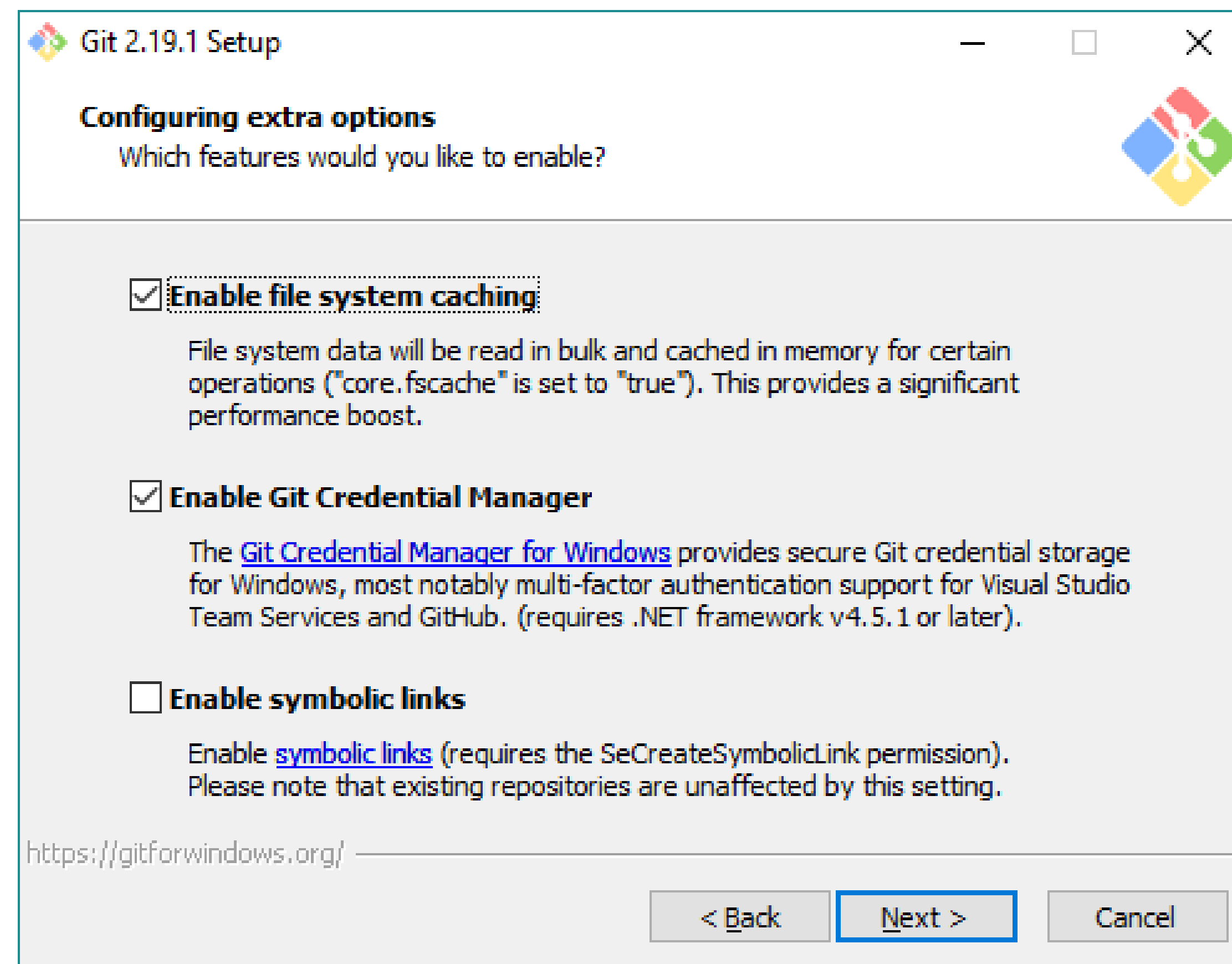
# Exemplo de instalação no windows

Selecione a segunda opção: Use Windows default console.



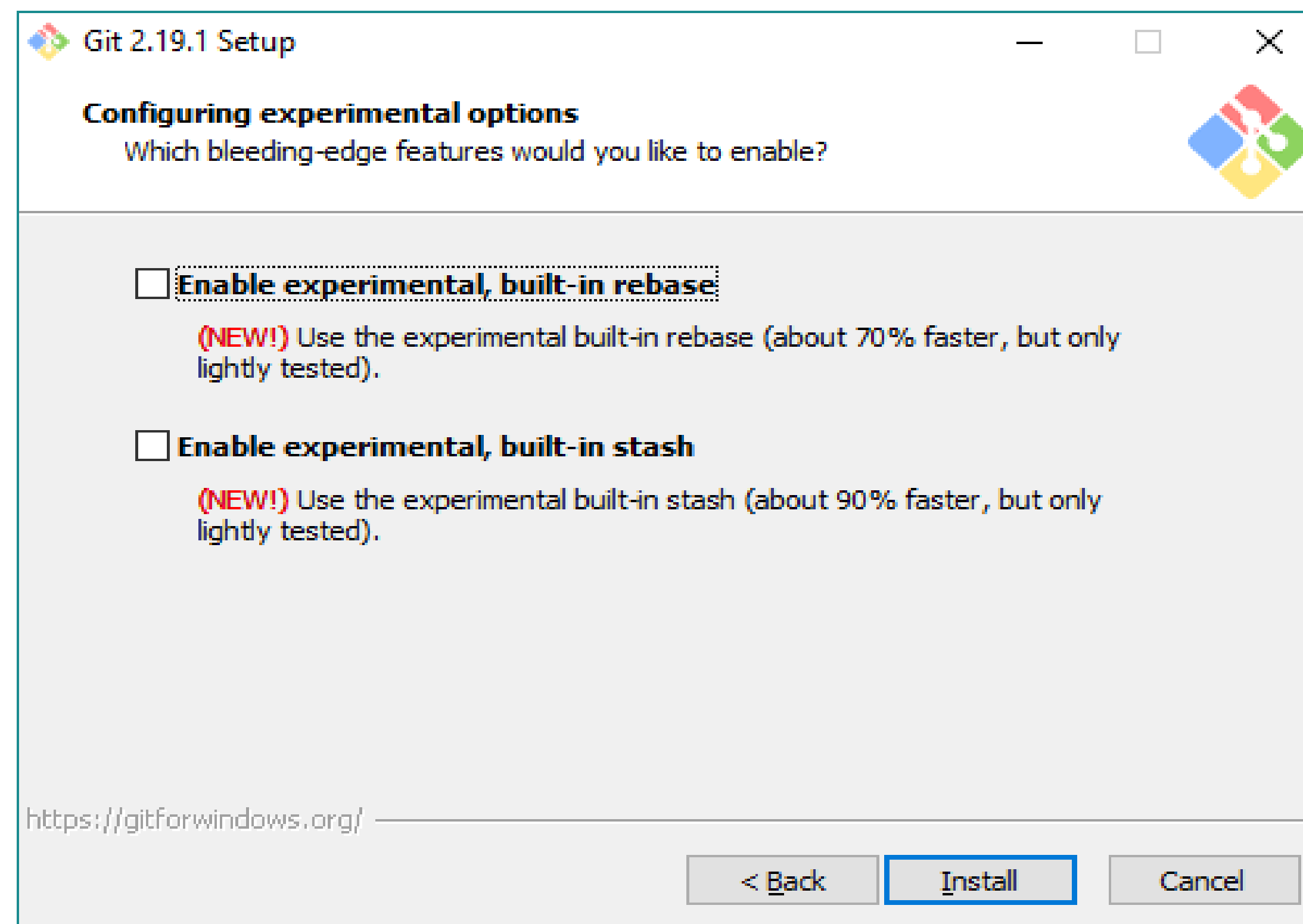
# Exemplo de instalação no Windows

Algumas opções extras para gerenciamento de memória, clique em next



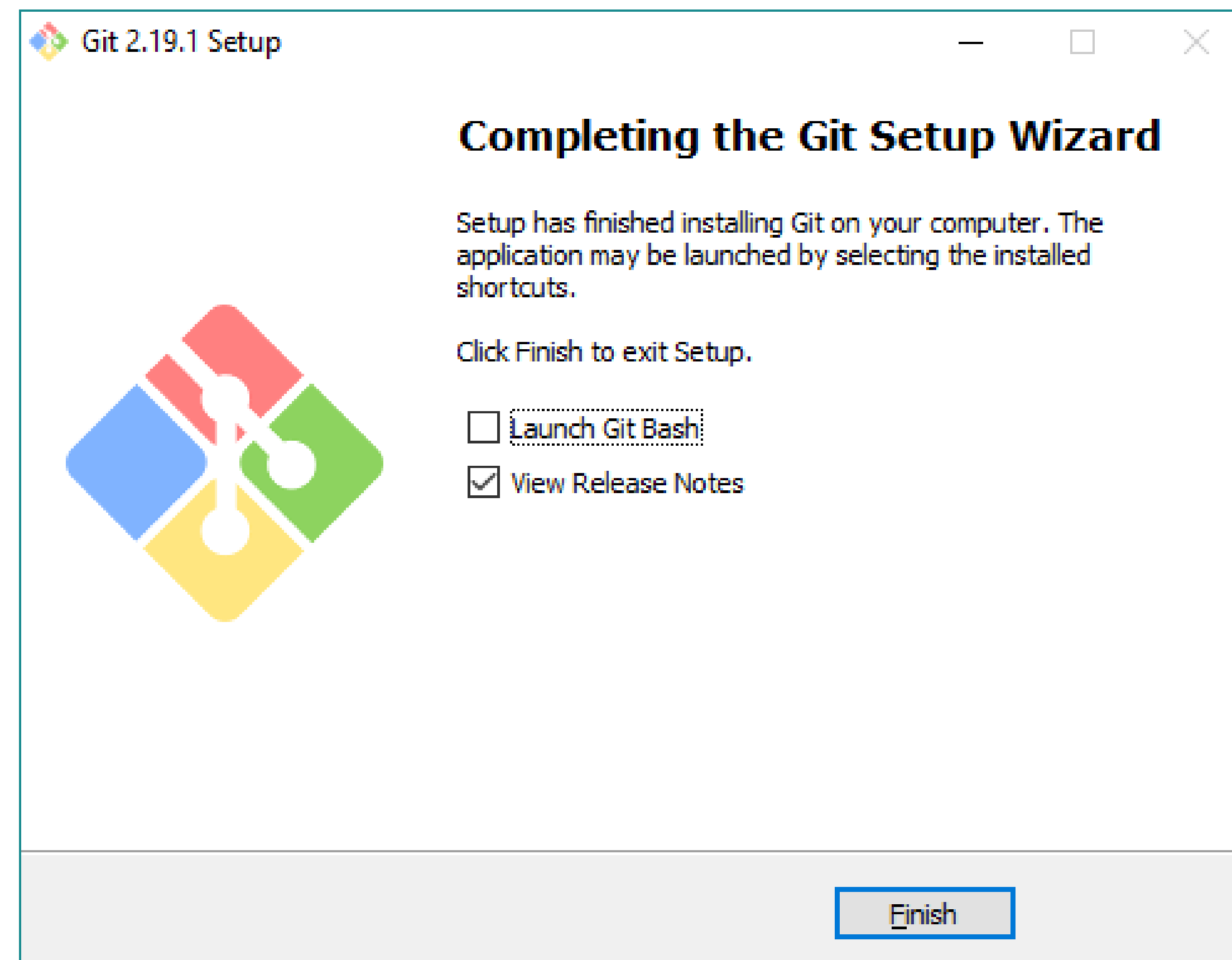
# Exemplo de instalação no Windows

Última tela de configuração não selecione nenhum item ainda são versões Beta, clique em Install.



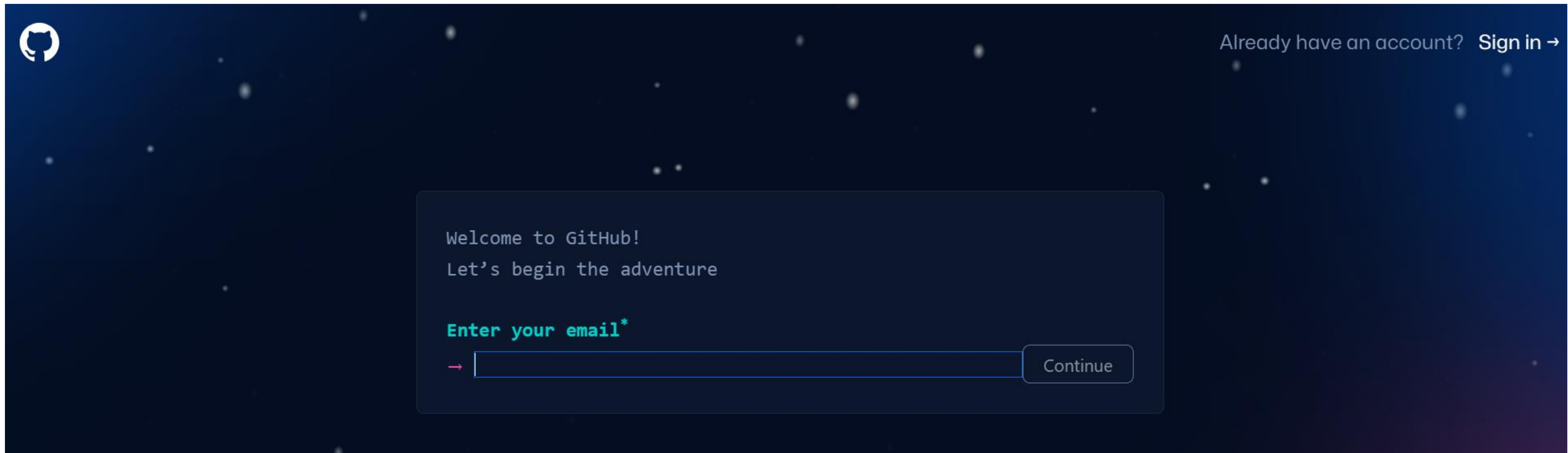
# Exemplo de instalação no Windows

Aguarde finalizar a instalação e clique em finish.



# Crie sua conta no GitHub

O **Git SCM** não funciona sozinho então crie sua conta no **GitHub**, para isso acesse: <https://github.com/> .  
Preencha os campos conforme as instruções do site.  
Fique atento pois são informações pessoais, quando finalizar você será direcionado a tela principal do GitHub.

A screenshot of the GitHub sign-up page. The background is dark blue with a starry pattern. In the top left corner is the GitHub logo. In the top right corner, it says "Already have an account? Sign in →". In the center, there is a white box containing the text "Welcome to GitHub!" and "Let's begin the adventure". Below this, it says "Enter your email\*" in green. There is a text input field with a red arrow pointing to it, and a "Continue" button to its right.

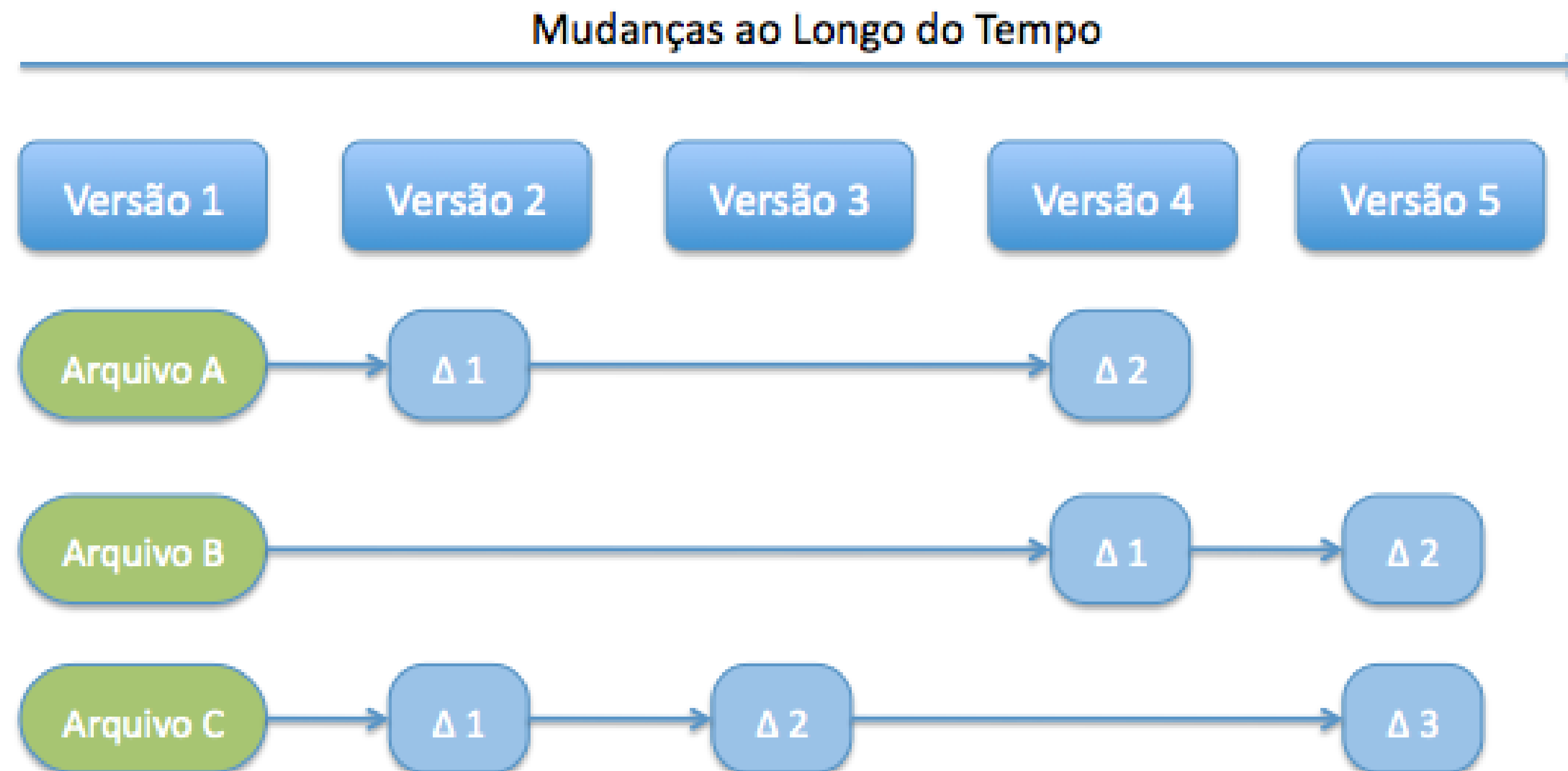
Caso já tenha uma conta não será necessário criar outra.

# Controle de versão

- ✓ Significa que o conjunto deve ter algum identificador único, como o número da compilação ou o número do item de mudança no controle de versão.

# Controle de versão

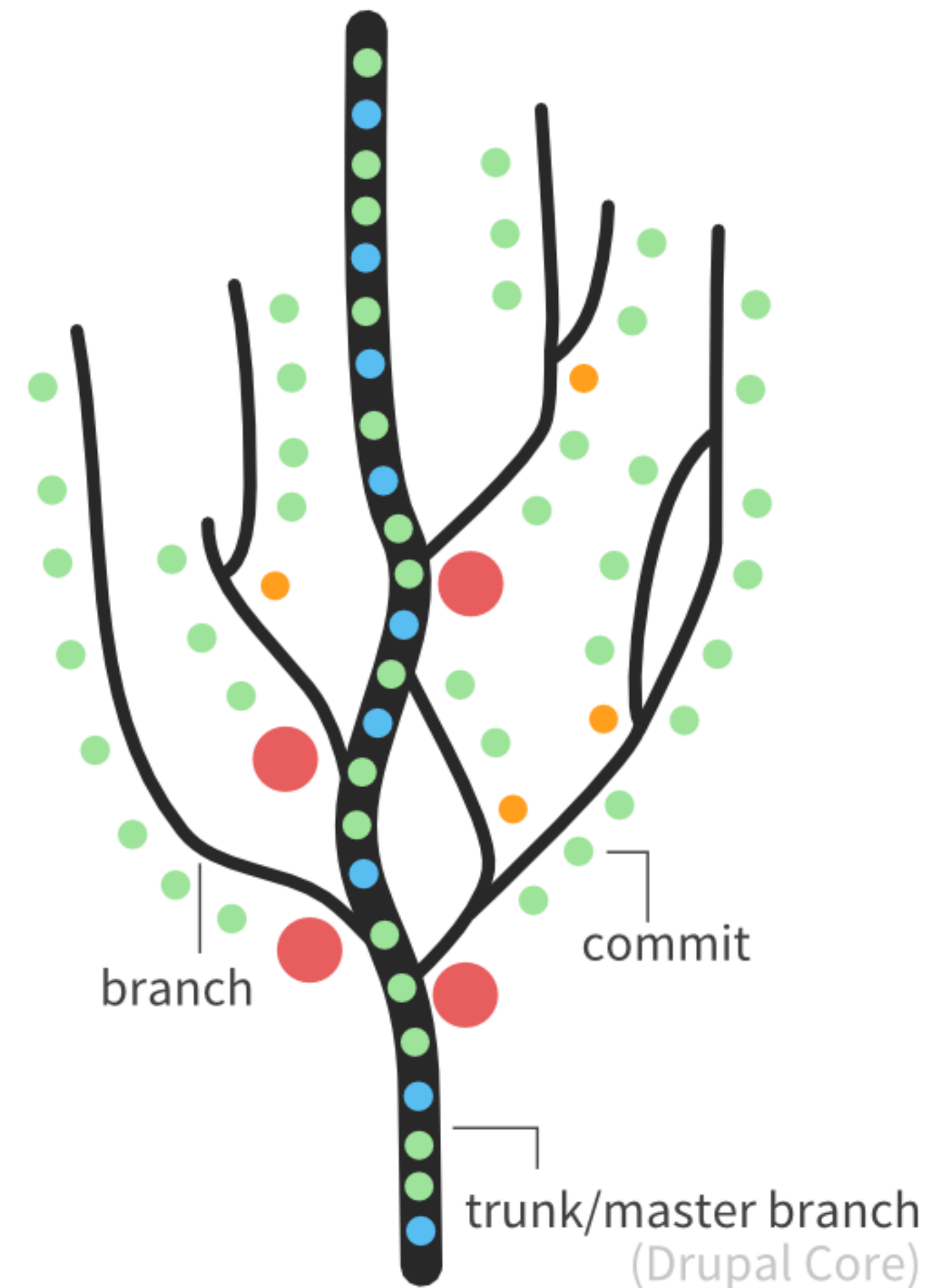
- Sistema de versionamento que armazena as diferenças de versão



<https://rafaelsotoblog.wordpress.com/2010/03/28/tutorial-controle-de-versao-distribuido-git-parte-1/>

# Controle de versão

- Árvore de versões



• Fonte: <http://www.drupal.org/node/991716>

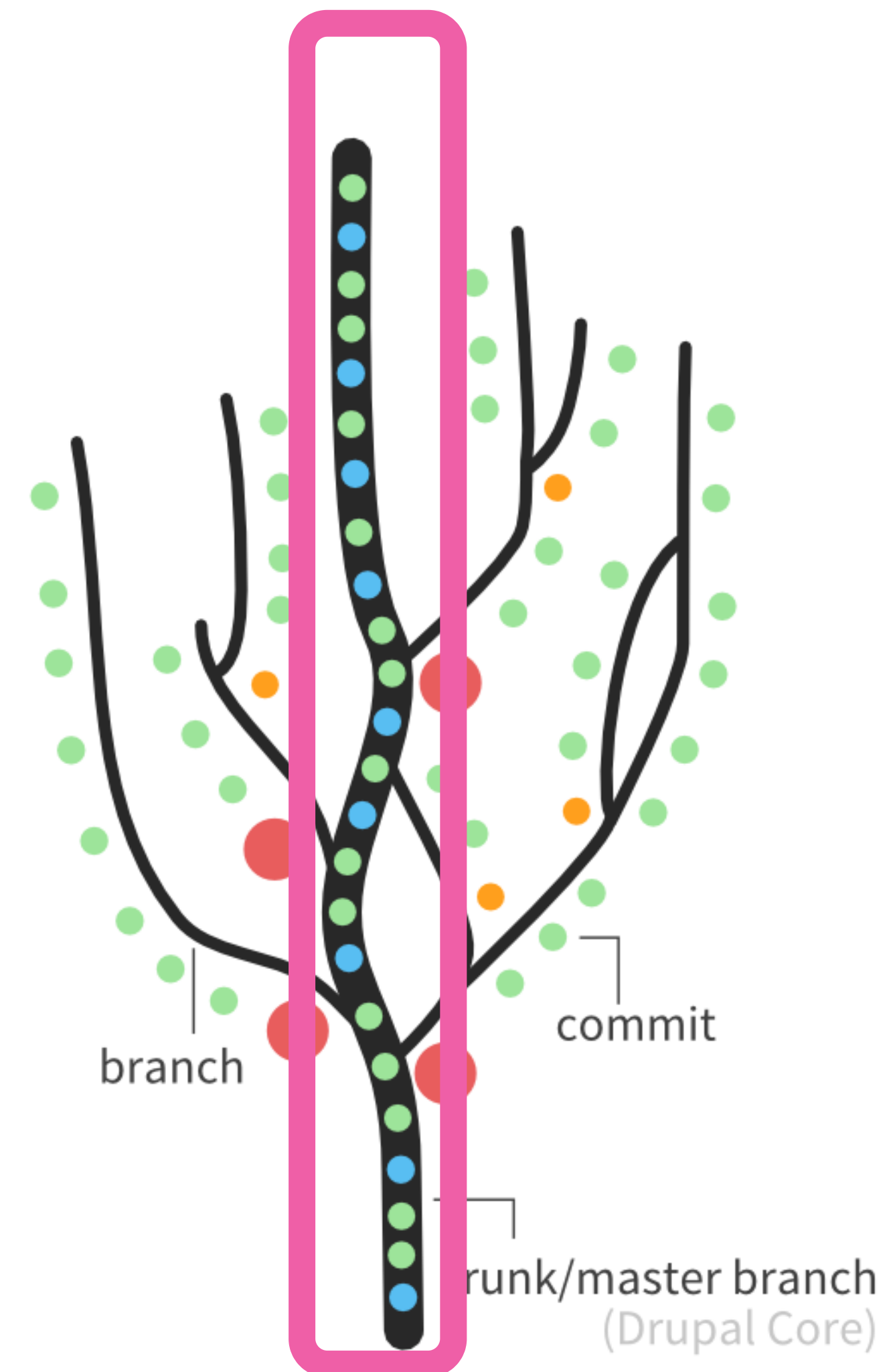


# Controle de versão

- Tronco (trunk) ou ramo master (branch master)

Histórico principal de modificações.

- Fonte

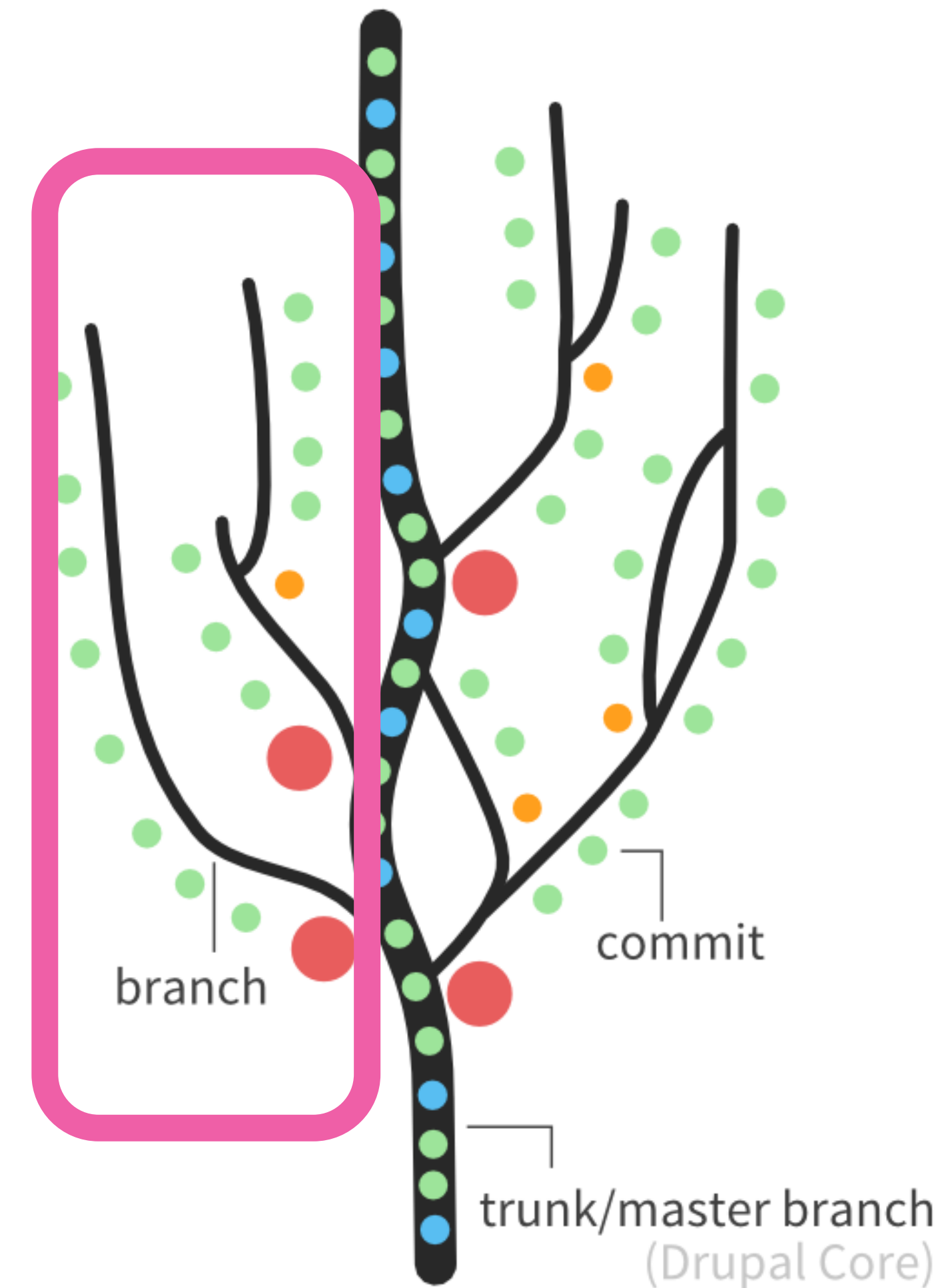


# Controle de versão

- Ramo (branch)

Histórico de modificações que se originam de outro histórico.

- Fonte: <http://www.drupal.org/node/991716>

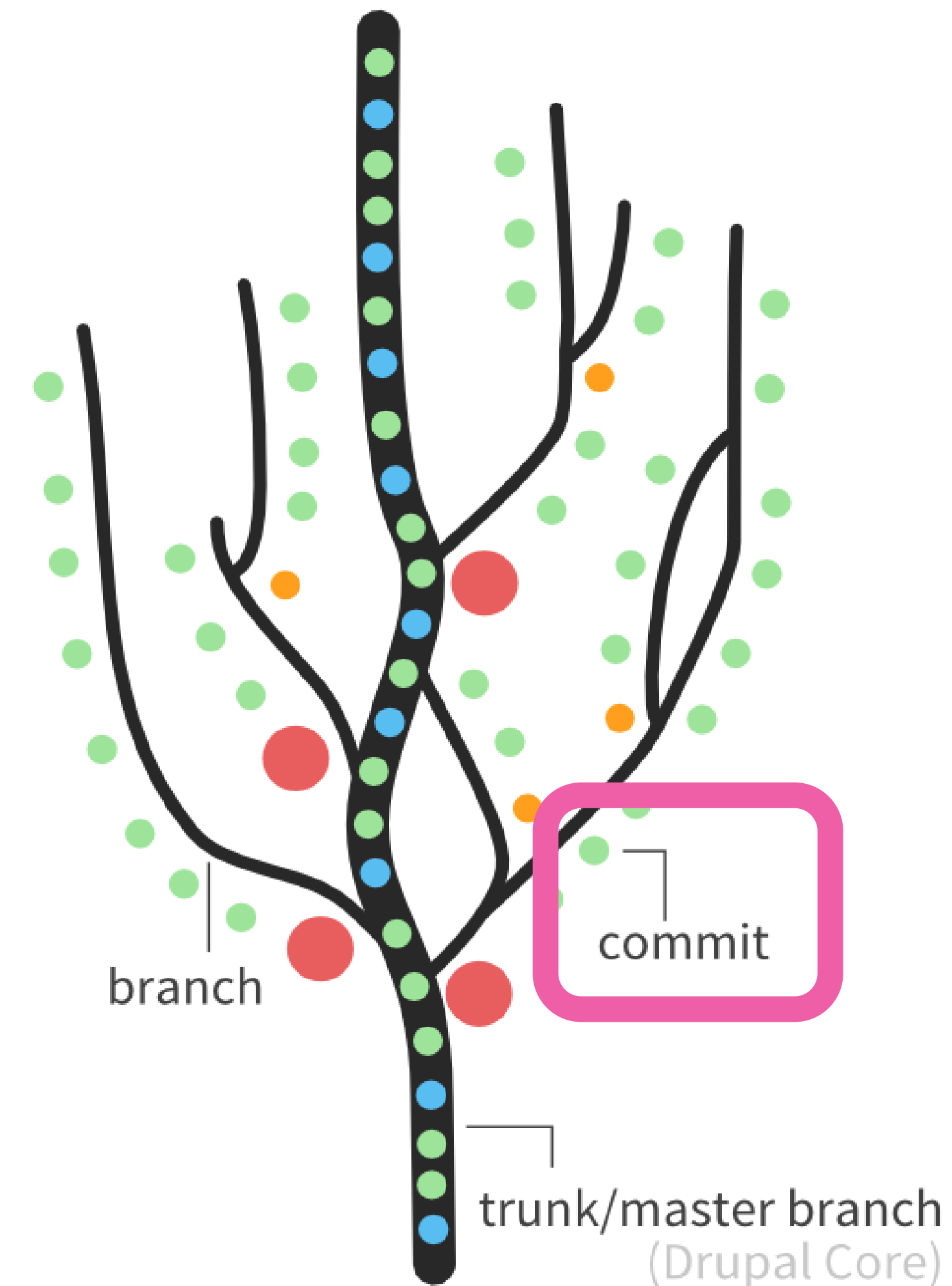


# Controle de versão

- Commit

Registro de modificações no histórico (entregar ou enviar uma modificação).

- Fonte: <http://www.drupal.org/node/991716>

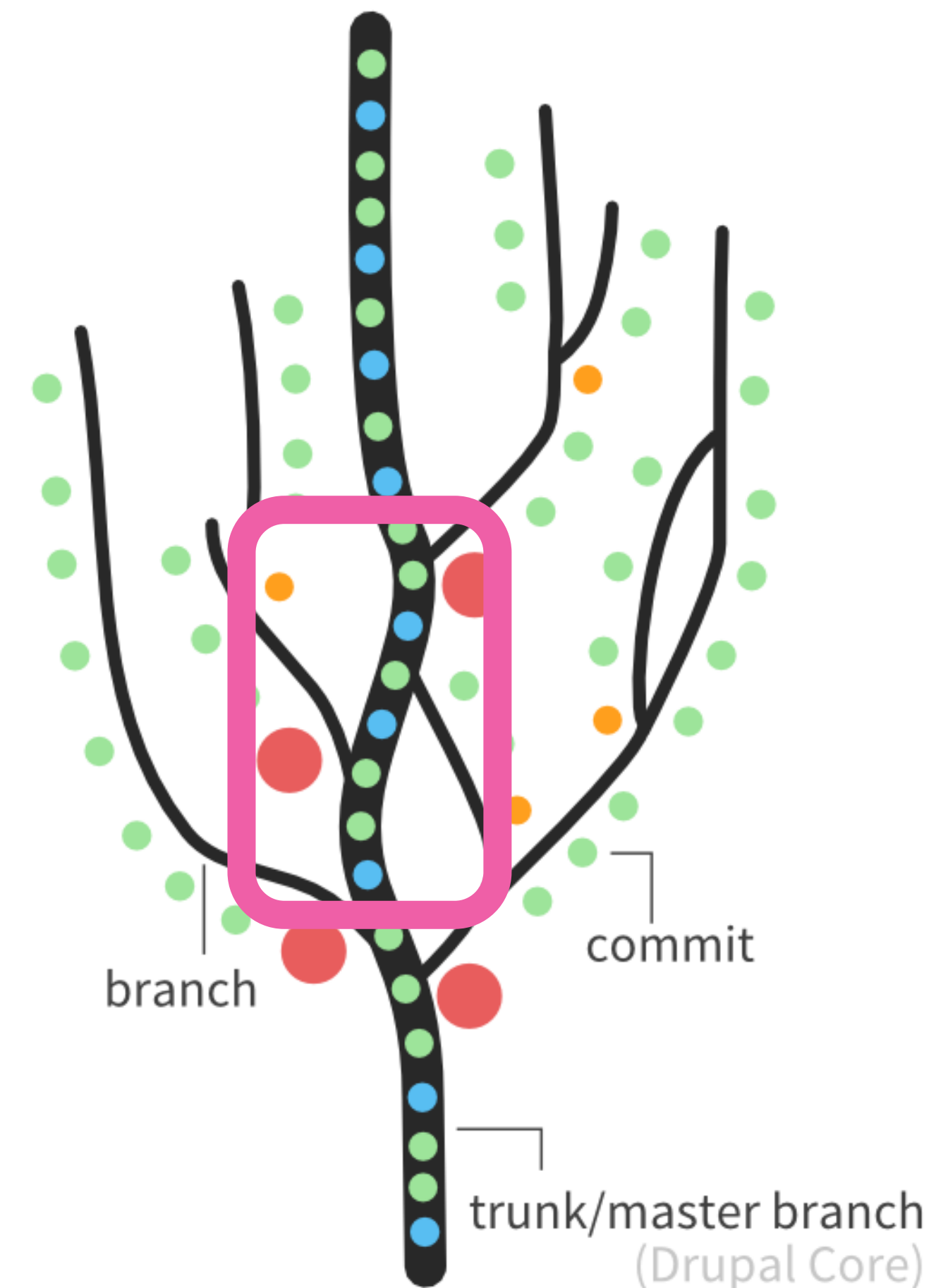


# Controle de versão

- Merge (fundir, misturar)

Incorporação das modificações de um branch em outro.

- Fonte: <http://www.drupal.org/node/991716>



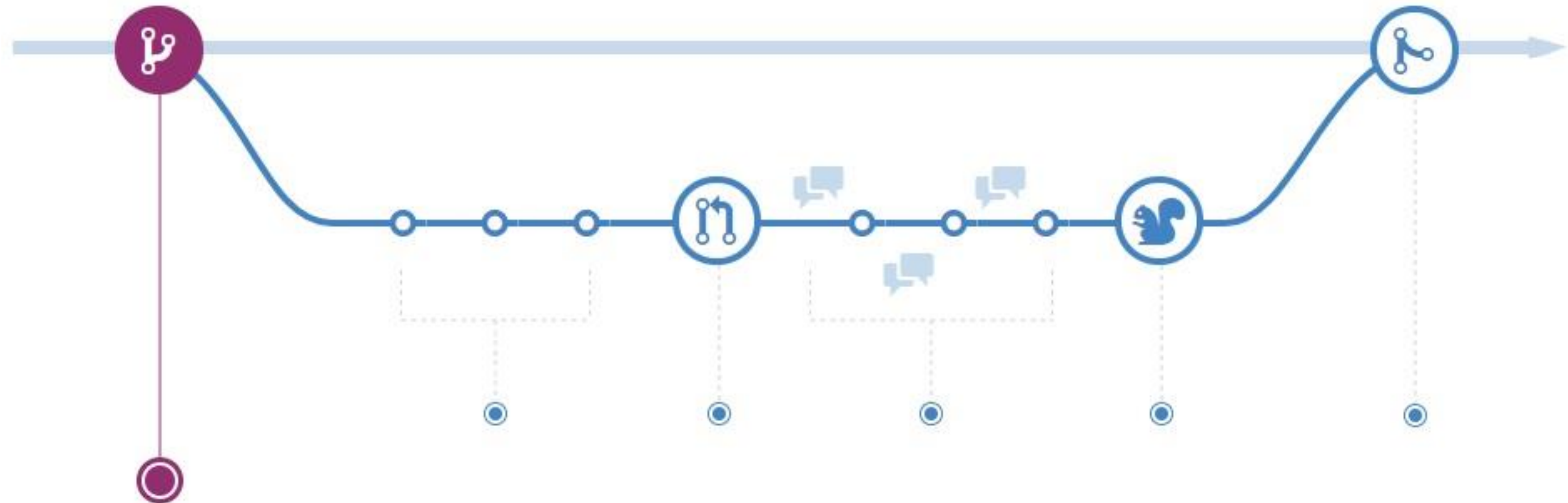
# Workflow recomendado pelo GitHub

1. Criar um branch.
2. Realizar os commits desejados no branch criado.
3. Criar um pull request (solicitação de atualização).
4. Discutir e revisar suas alterações com outros colaboradores.
5. Deploy (implantar)
6. Realizar o merge no branch master.

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

# Workflow recomendado pelo Github

## 1. Criar um branch.

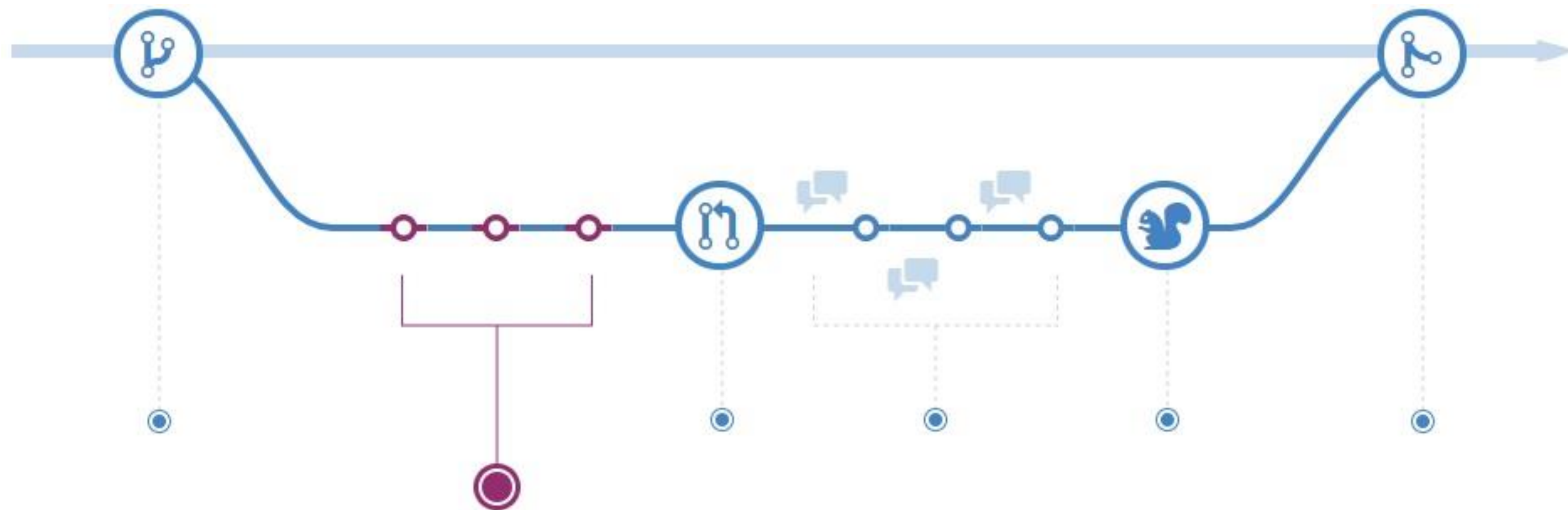


Fonte: <https://guides.github.com/introduction/flow>



# Workflow recomendado pelo GitHub

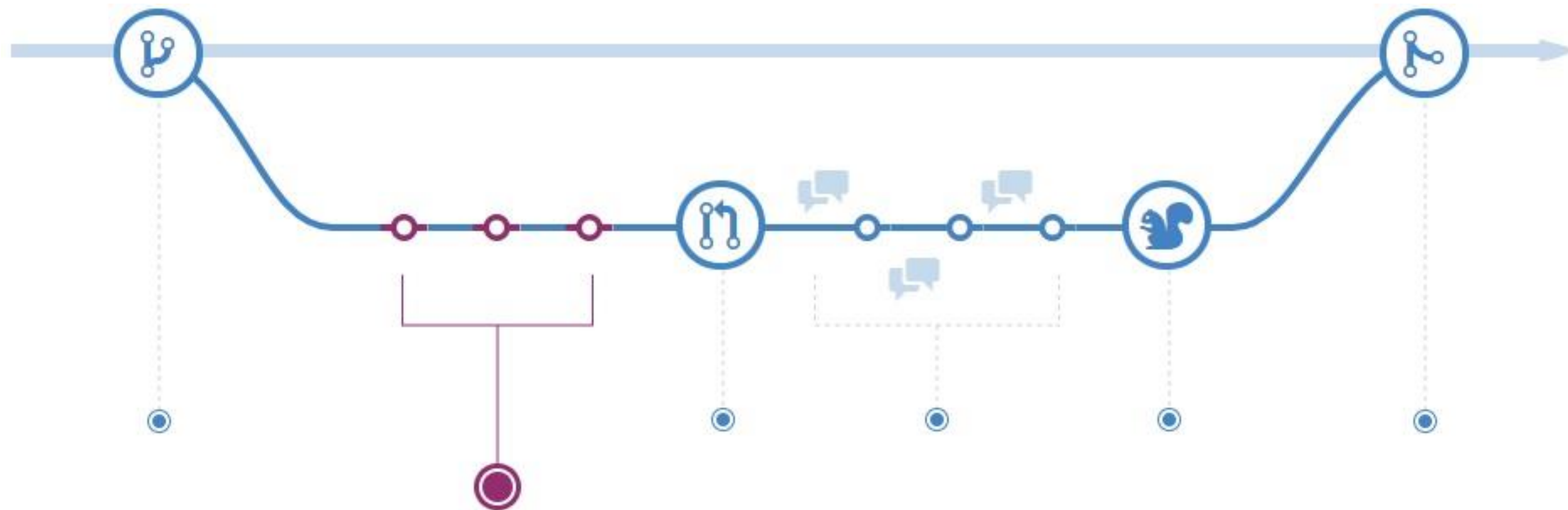
2. Realizar os commits desejados no branch criado.



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

# Workflow recomendado pelo GitHub

3. Criar um **pull request** (solicitação de atualização).

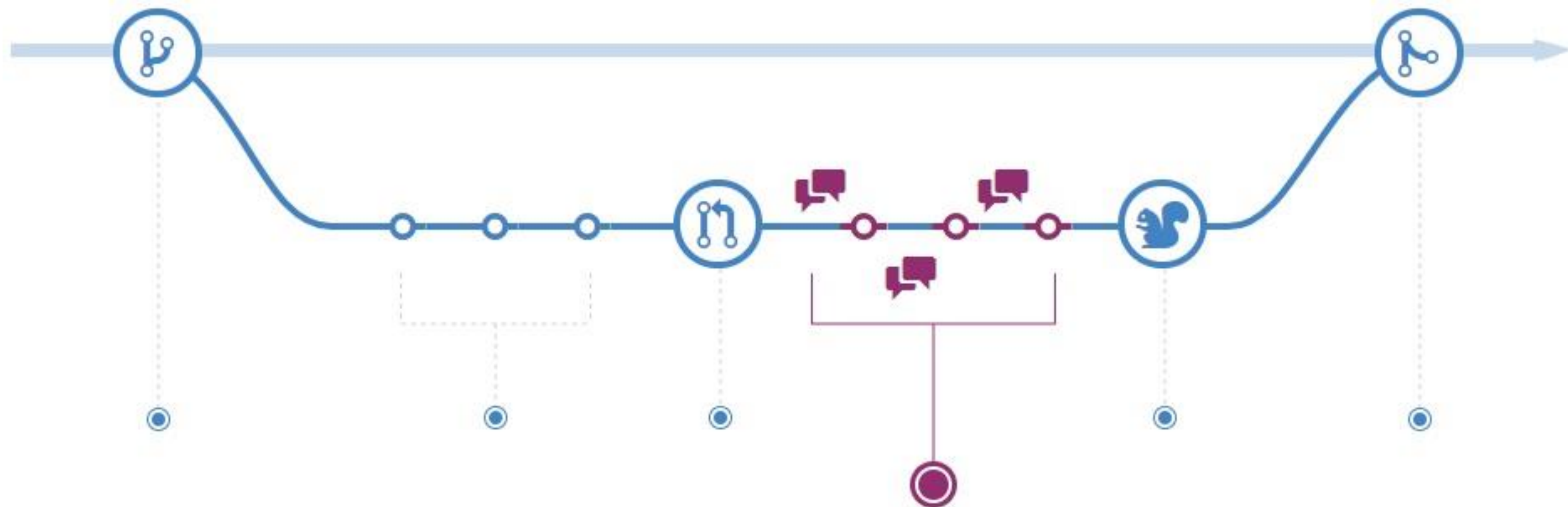


Fonte: <https://guides.github.com/introduction/flow>



# Workflow recomendado pelo GitHub

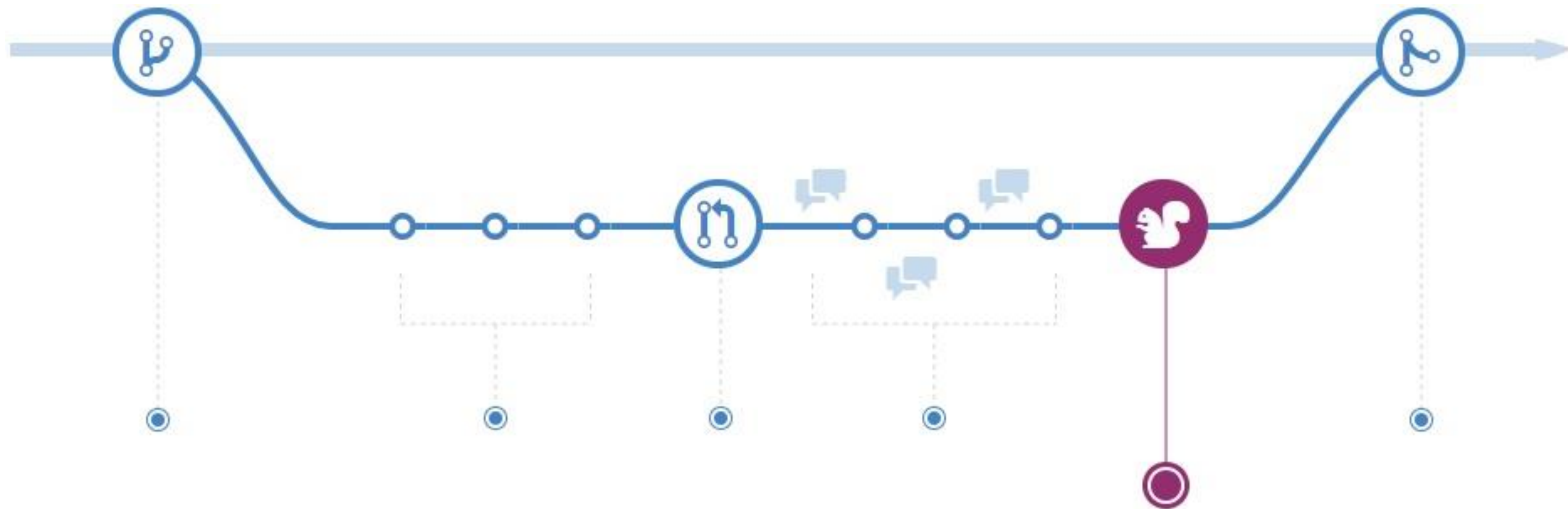
4. Discutir e revisar suas alterações com outros colaboradores.



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

# Workflow recomendado pelo GitHub

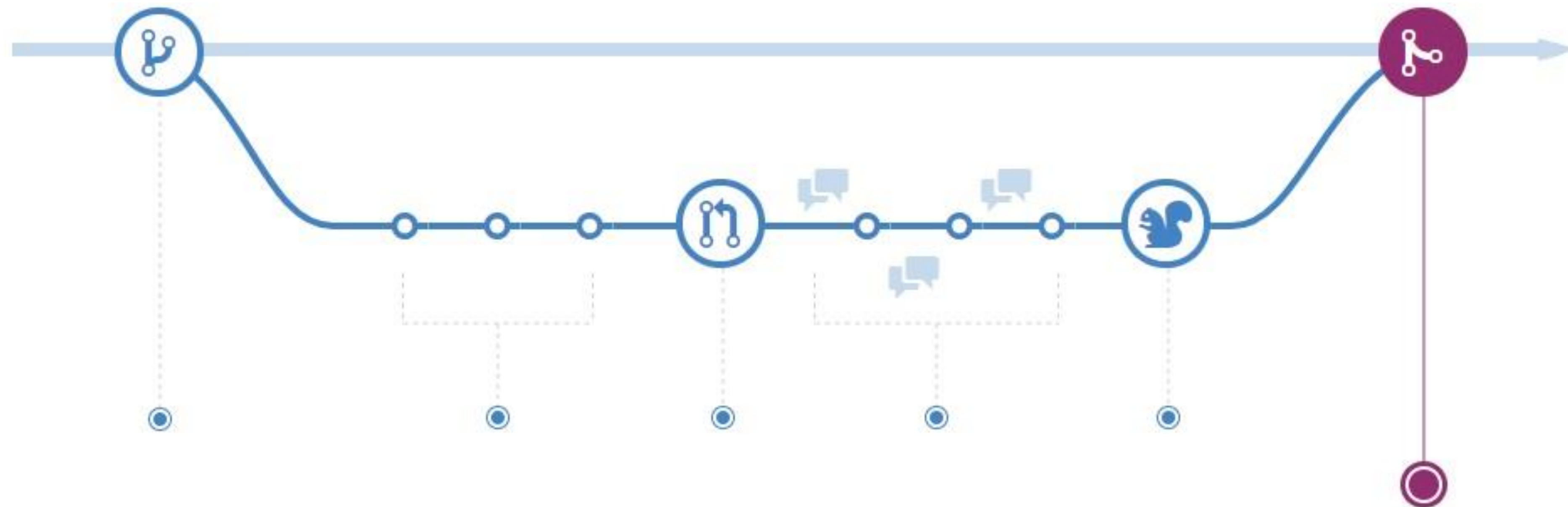
## 5. Deploy (implantar)



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

# Workflow recomendado pelo GitHub

6. Realizar o merge no branch master.

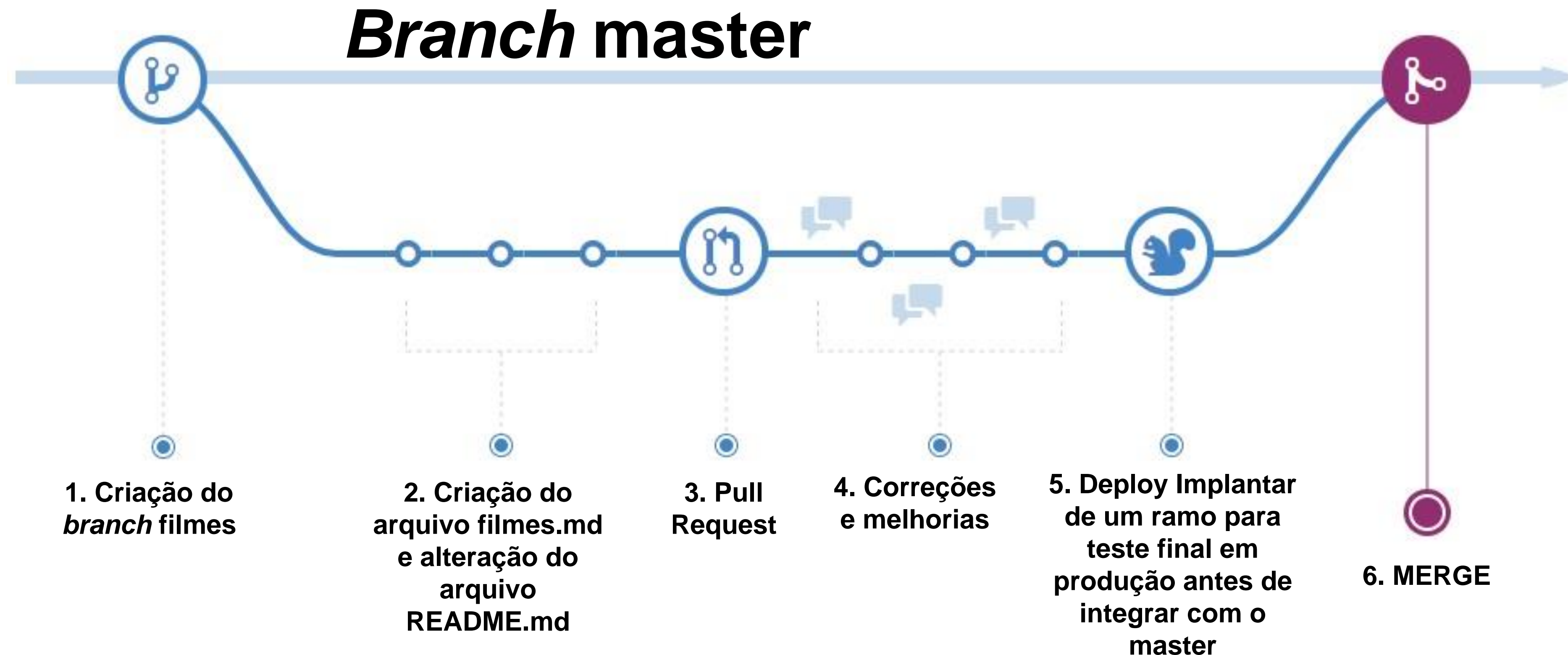


Fonte: <https://guides.github.com/introduction/flow>

# Merge

- ✓ **Significado:** Fundir, Misturar, Integrar
- ✓ Integração das modificações de um *branch* (*ramo*) em outro.

# Exemplo de merge



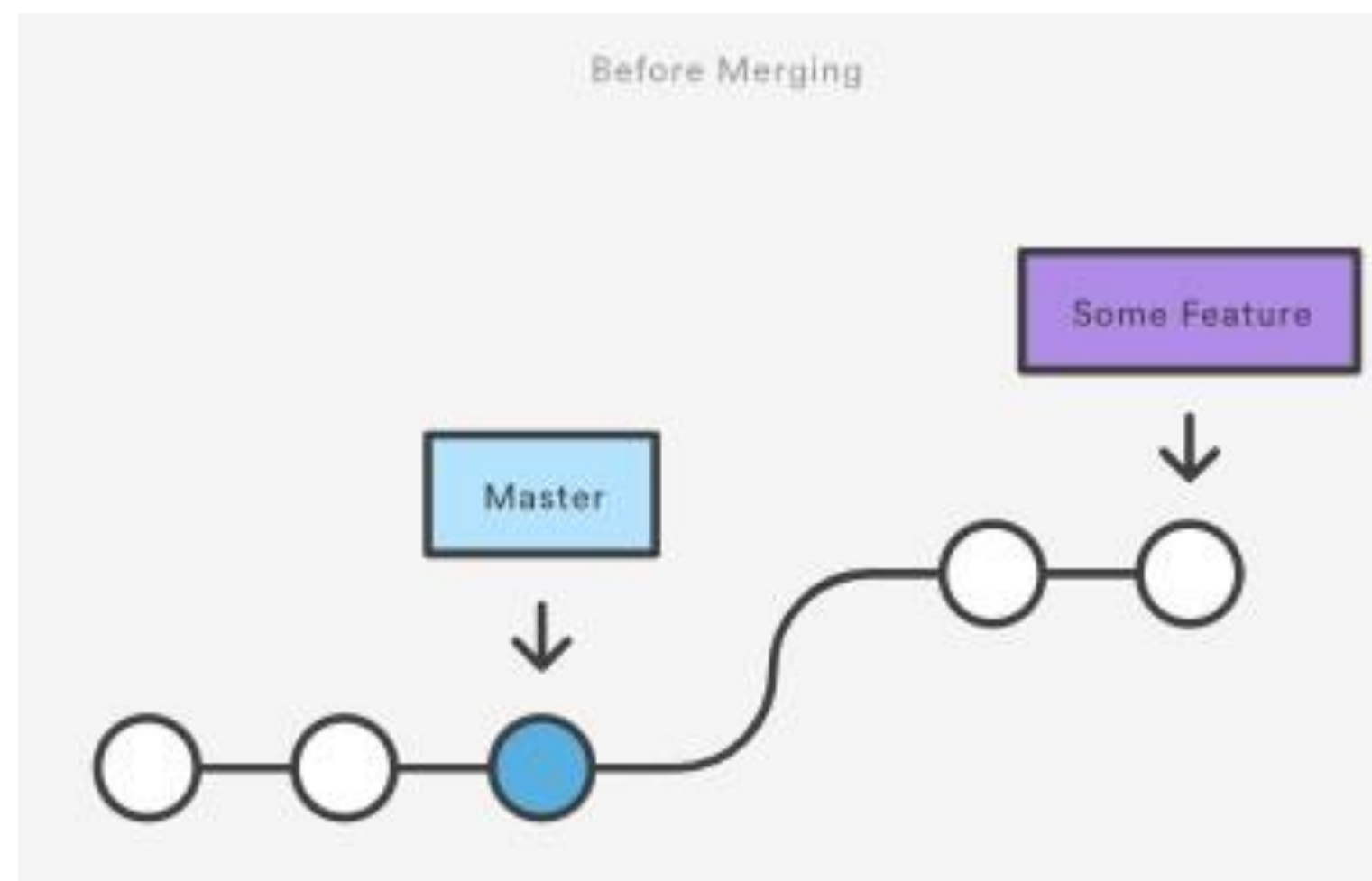
Fonte: <https://guides.github.com/introduction/flow>



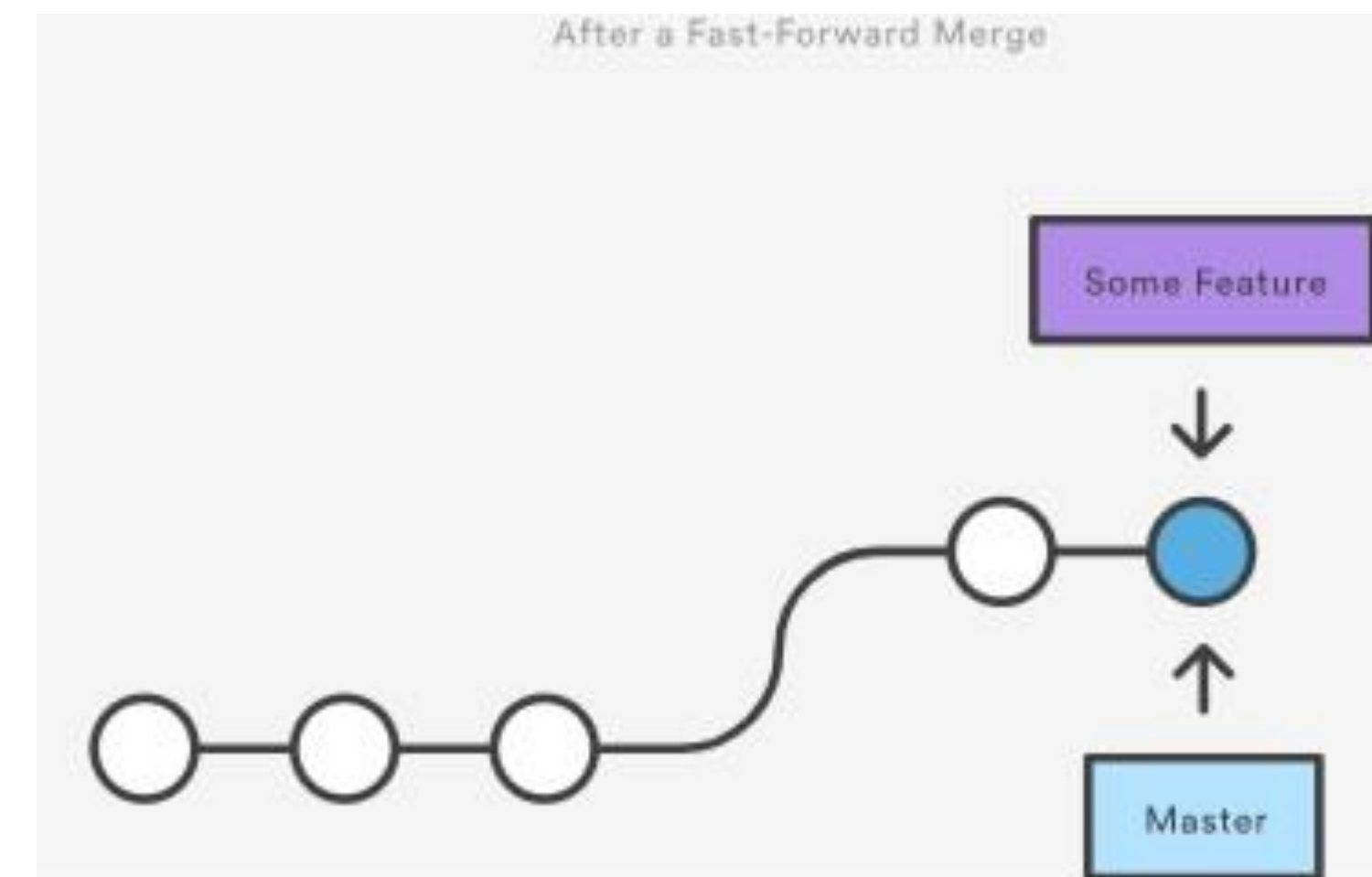
# Fast-forward merge

- Integração realizada quando o branch master não sofreu nenhuma alteração desde a criação do branch que será integrado.

antes do merge



depois do merge

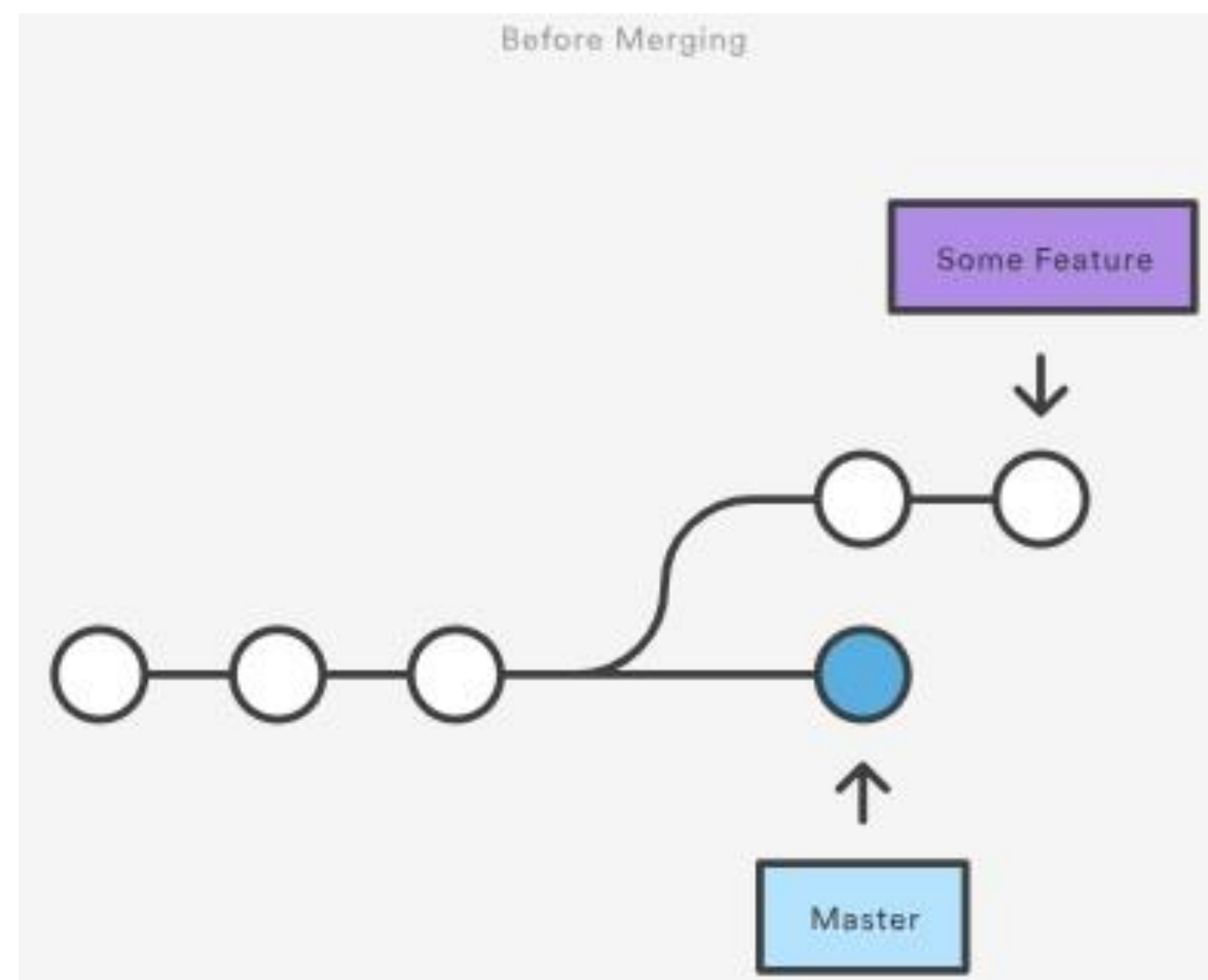


Fonte: <https://www.atlassian.com/git/tutorials/git-merge>

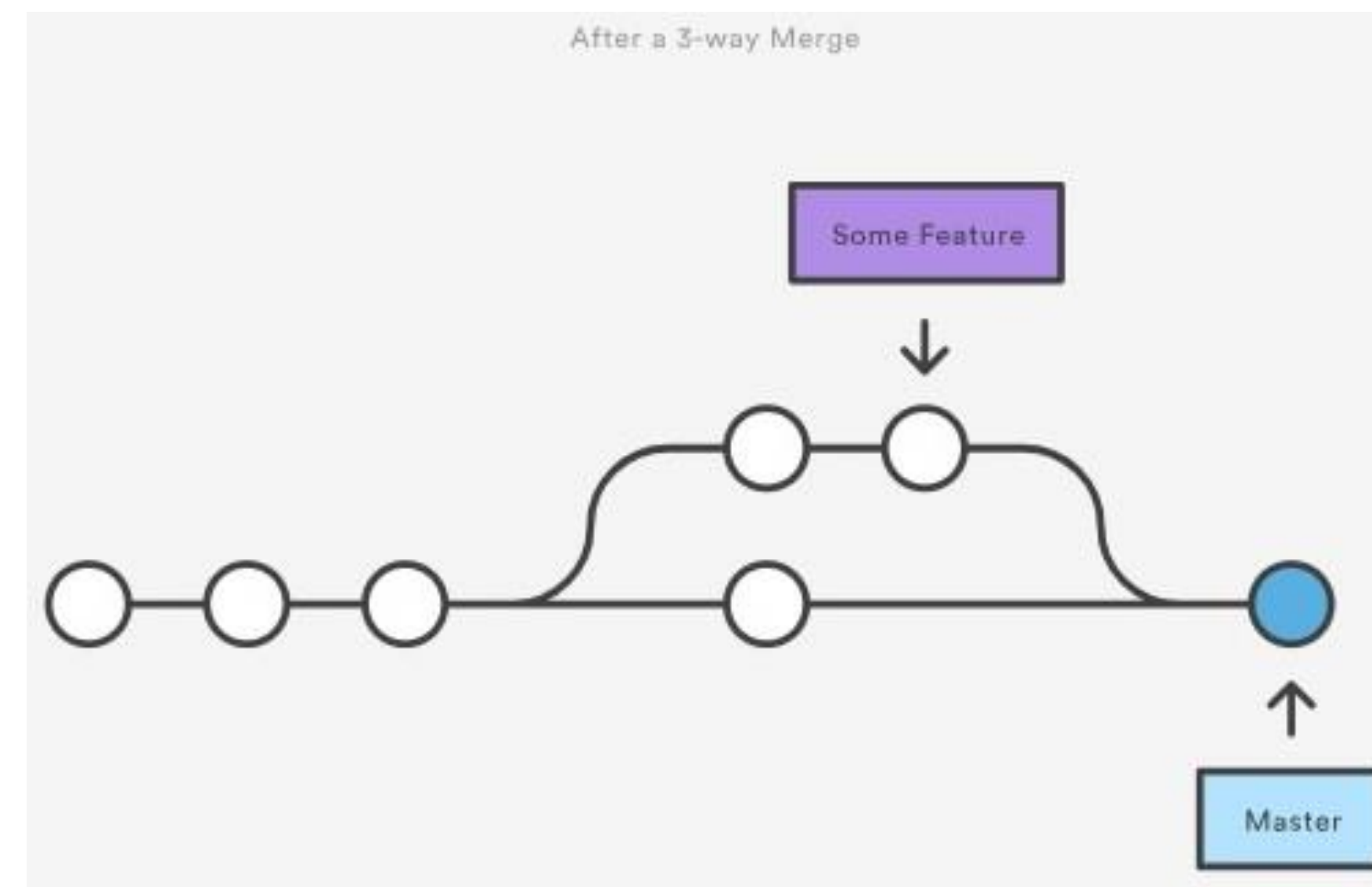
# way merge

- Integração realizada quando o branch master **sofreu** alguma alteração desde a criação do branch que será integrado.

antes do merge



depois do merge

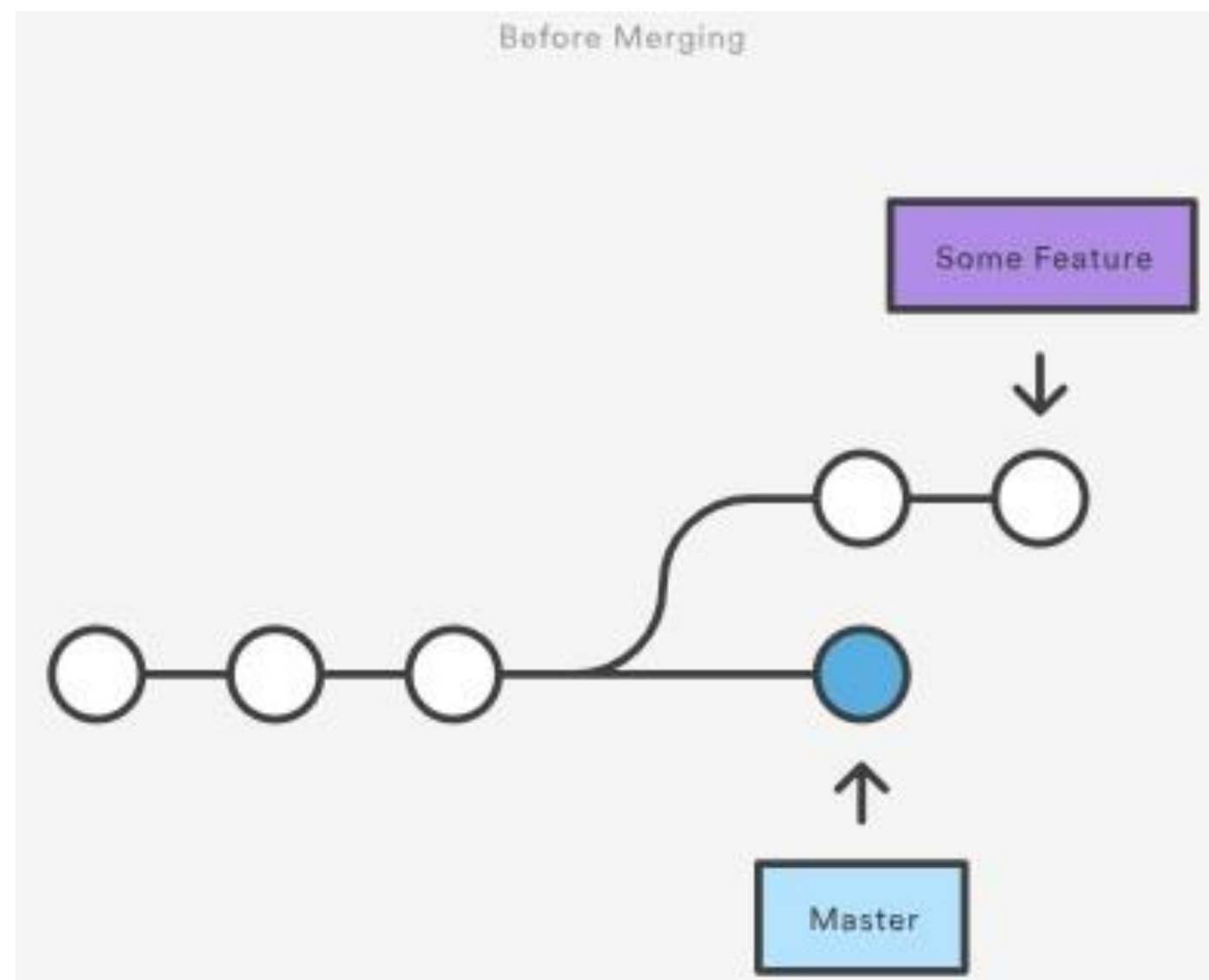


Fonte: <https://www.atlassian.com/git/tutorials/git-merge>

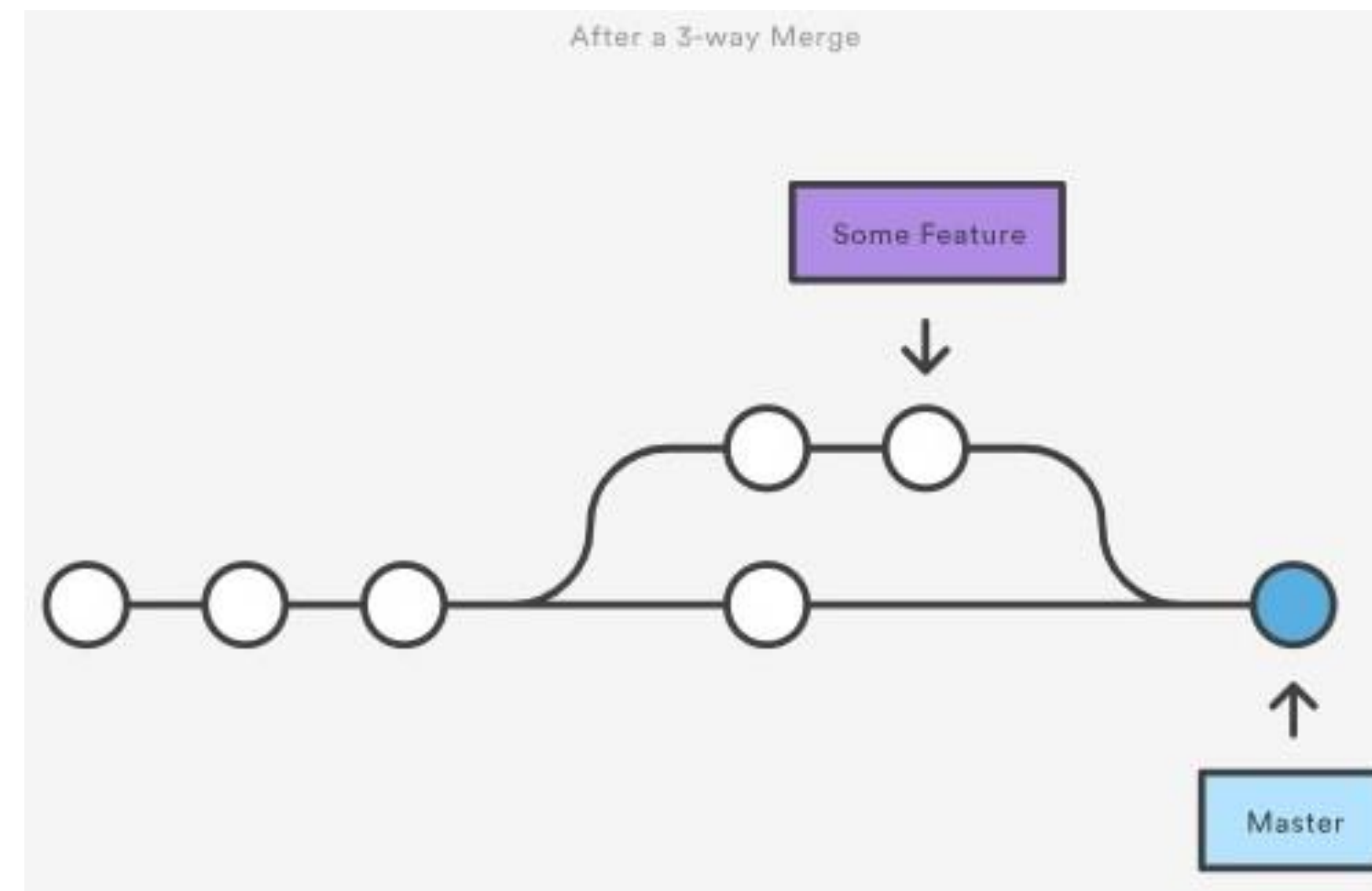
# way merge

- A nomenclatura vem do fato de que Git usa três commits para gerar o merge commit: os dois commits do ramo e o commit anterior comum.

antes do merge



depois do merge



Fonte: <https://www.atlassian.com/git/tutorials/git-merge>



# Conflitos

- Se dois branches que você está tentando integrar alteraram a mesma parte de um mesmo arquivo, **é necessário resolver o conflito manualmente.**
- O colaborador responsável por resolver os conflitos deverá visualizar as diferenças e decidir qual será o conteúdo final.

# Resolução de Conflitos

#5: Update README.md > Conflicts

1 conflicting file	README.md	1 conflict	Prev ^	Next v	⚙	Mark as resolved
 README.md README.md	<pre>1  # Exemplos de ferramentas de controle de versão 2 3  &lt;&lt;&lt;&lt;&lt;&lt;&lt; update1 4  * Subversion (SVN) 5  ===== 6  * SVN 7  * Microsoft Source Safe 8  &gt;&gt;&gt;&gt;&gt;&gt;&gt; master 9  * Clearcase 10 * Git 11 * Mercurial 12 * CVS 13 14</pre>					

# Comandos Git

Uma vez instalado, você deve configurar sua conta git local, através do comando:

## **Git config**

```
git config --global user.name "nome do seu usuário"
```

```
git config --global user.email seu_email@xxxx.com
```

## **Git clone (Faz download do projeto pra um diretório local)**

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

## **Git add (Para adicionar um arquivo)**

```
git add nome_arquivo.extensão
```

Fonte: <http://comandosgit.github.io/>

# Comandos Git

## Git status

O comando lhe mostra em qual branch você se encontra. Vamos dizer que você adicione um novo arquivo em seu projeto, um simples arquivo README. Caso o arquivo não exista e você execute git status, você verá o arquivo não monitorado dessa forma:

```
# On branch master
# Untracked files:
# (use "git add {file}..." to include in what will be committed)
#
# README
nothing added to commit but untracked files present (use "git add" to track)
```

## Git Commit

git commit -m "Mensagem pra informar o commit criado"

Fonte: <http://comandosgit.github.io/>

# Comandos Git

**Git push** (empurra/envia os commit para um repositório remoto)

git push

Fonte: <http://comandosgit.github.io/>

# Comandos Git

## Git merge (merge básico, unido ao master)

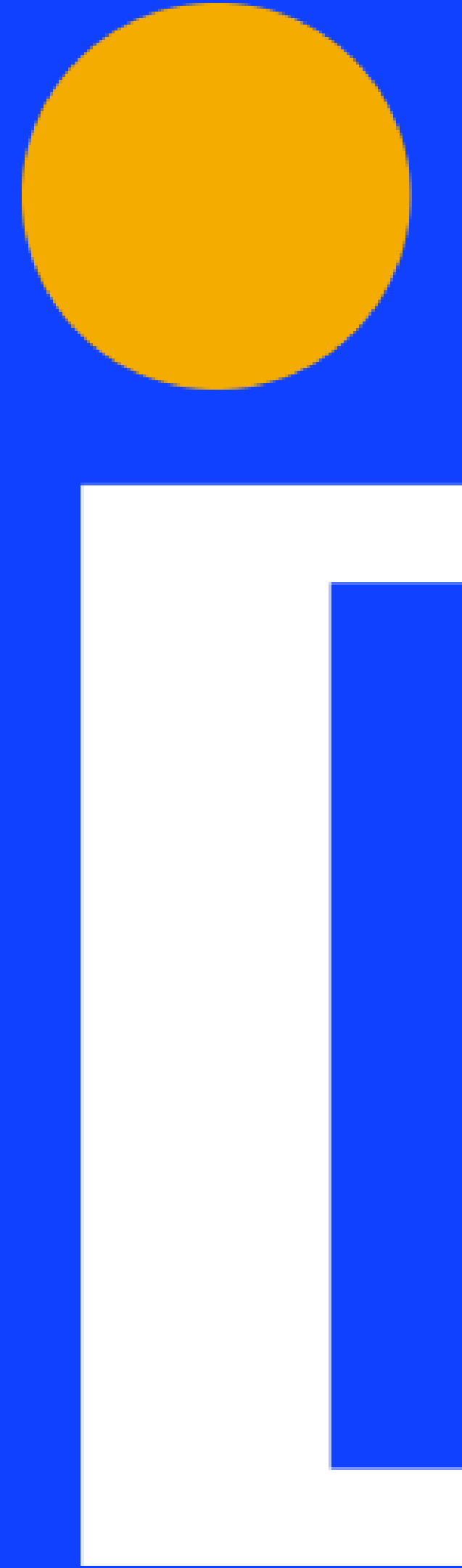
Suponha que você decidiu que o trabalho na tarefa XPTO está completo e pronto para ser feito o merge no branch **master**. Para fazer isso, você fará o merge do seu branch **novabranh**, bem como o merge do branch **hotfix** de antes. Tudo que você tem a fazer é executar o checkout do branch para onde deseja fazer o merge e então rodar o comando git merge:

```
git checkout master  
git merge novabranh
```

Fonte: <http://comandosgit.github.io/>

# Referência desta aula

- GitHub. GitHub Guides.
- Disponível em <https://guides.github.com>



IBMEC.BR

 /IBMEC

 IBMEC

 @IBMEC\_OFICIAL

 @IBMEC

 **ibmec**