

# Introdução ao Git e GitHub

---

# Eucurso.com.br

Este é um ebook produzido para o portal EuCurso.com.br.

Você pode utilizá-lo livremente para qualquer fim, inclusive comercial. Porém não está autorizada a distribuição do ebook em qualquer portal. Divulgações podem ser feitas com links para o portal eucurso.com.br.

Todos os direitos reservados para Alfamídia

---

# E-book sobre Git e GitHub

Este e-book apresenta a instalação do Git e o uso básico do Git e GitHub.

O e-book **Apostila de AWS - Configurando Servidor Linux 2** apresenta a instalação de um ambiente Linux, que será utilizado para a instalação do Git em ambiente Linux.

O e-book **Transferindo uma Aplicação PHP Windows para um servidor Linux na Nuvem Amazon** apresenta parte do conteúdo deste e-book, pois utiliza o Git para a transferência de arquivos entre dois ambientes. No presente e-book veremos alguns recursos adicionais do Git, de forma a mostrar o uso básico do Git de forma mais completa.

O Git é um sistema de controle de versões distribuído. Ele nos permite, por exemplo, manter o controle das diferentes versões do código-fonte de um software, a medida que o mesmo vai sendo desenvolvido, identificando quem fez determinadas modificações, retornando para versões anteriores, etc.

O GitHub é uma plataforma que hospeda os arquivos gerenciados pelo Git, de forma que diferentes programadores possam compartilhar arquivos na internet.

---

# Instalando o Git no Linux

Neste e-book utilizaremos como ambiente Linux um servidor Amazon AWS Linux 2, cuja instalação é mostrada no e-book books **Apostila de AWS - Configurando Servidor Linux 2**.

Como todo o processo de criação de uma instância linux na AWS (Amazon Web Service), acesso remoto ao shell do Linux, e configurações básicas são etapas mostrados no e-book, citado, a configuração do Git exige apenas o comando a seguir:

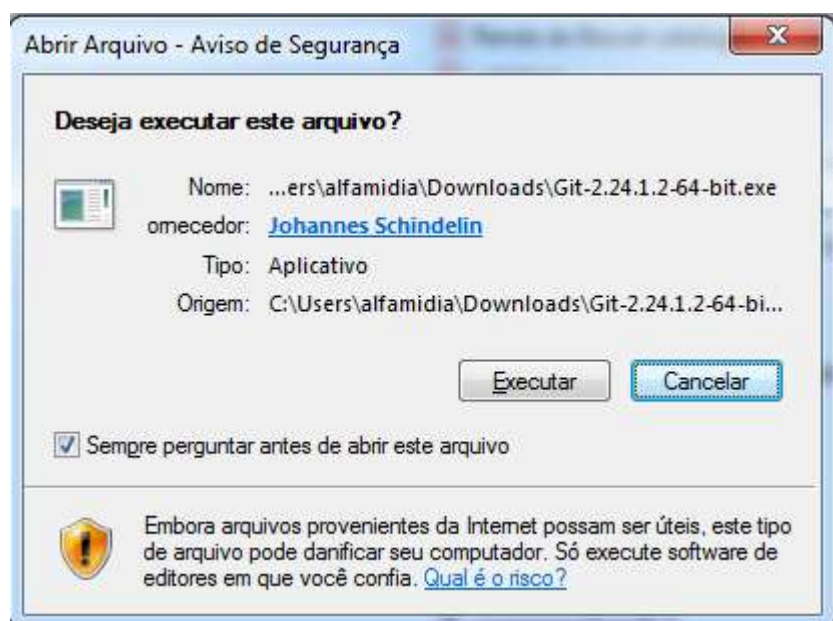
```
sudo yum install git -y
```

O comando `sudo` define que estamos executando a ferramenta *yum* como superusuário. Os parâmetros *install git* basicamente informam que estamos fazendo a instalação do Git, enquanto o parâmetro *-y* define que todas as perguntas serão respondidas com *y* (*yes*, ou seja, resposta *sim* para todas as perguntas na instalação).

# Instalando o Git em Windows

Apenas digitando “git” no google, tanto a página do Git quanto do GitHub são mostradas entre os primeiros links. Você pode também ir direto para o site <https://git-scm.com/>. No site entre na seção “downloads”, e selecione o botão de download para Windows. Uma vez concluído o download, inicie a instalação.

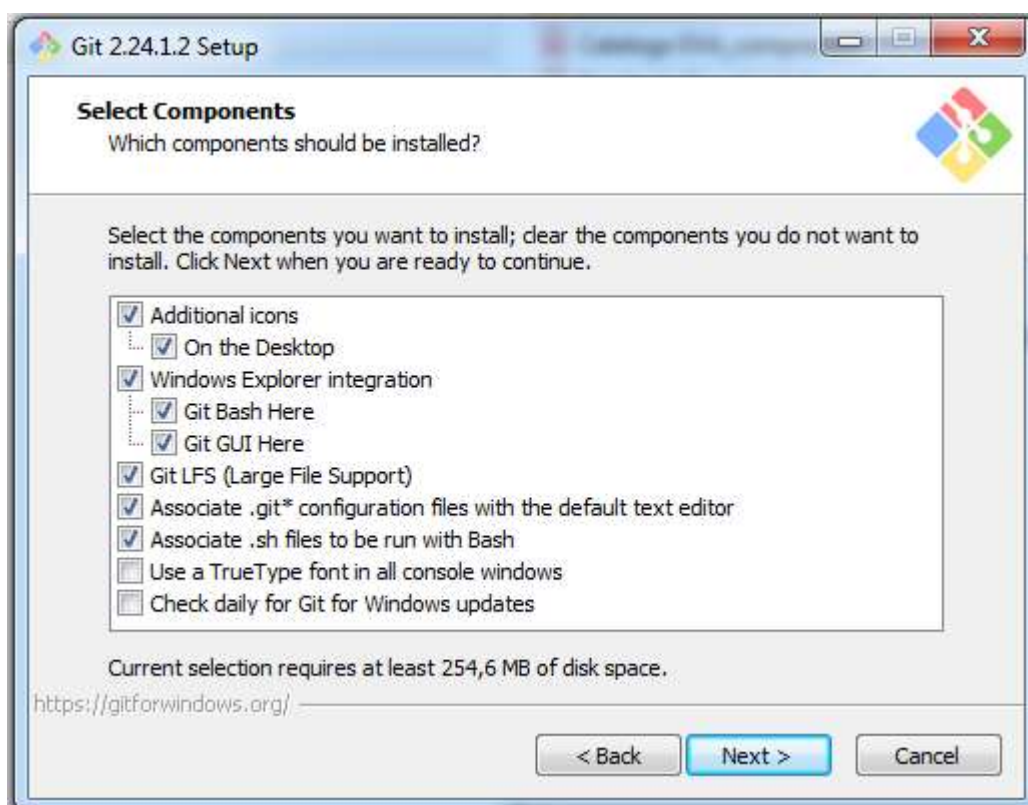
Ao executar o instalador, a janela a seguir deve ser exibida:



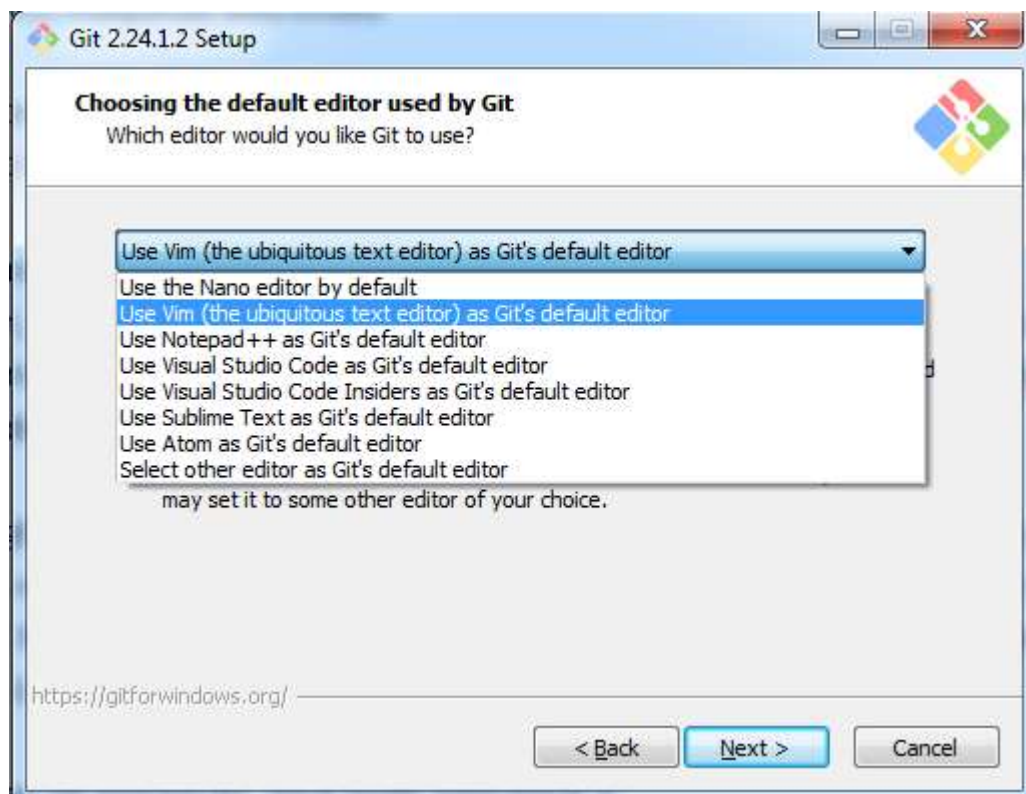
A tela seguinte mostra a licença de uso GNU GPL, que basicamente informa seu direito de utilizar livremente o Git.



A tela seguinte abre as opções de configuração dos componentes a serem instalados. Em nosso tutorial não vamos fazer nenhuma alteração. Você pode apenas clicar *Next*.



No processo de instalação, você pode selecionar o editor que será utilizado nas edições de configurações do Git, conforme mostrado na figura a seguir. Utilize o editor que julgar mais conveniente.

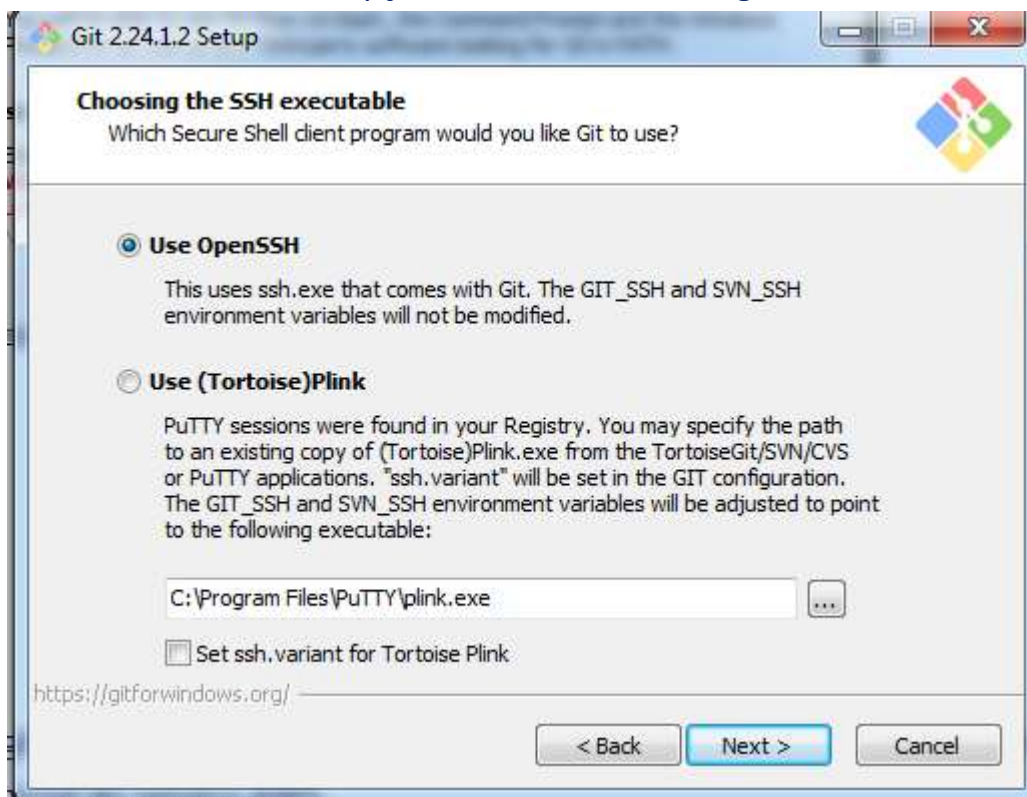


A janela seguinte oferece algumas opções e configuração para uso do Git. Em nosso exemplo, vamos manter na opção recomendada.

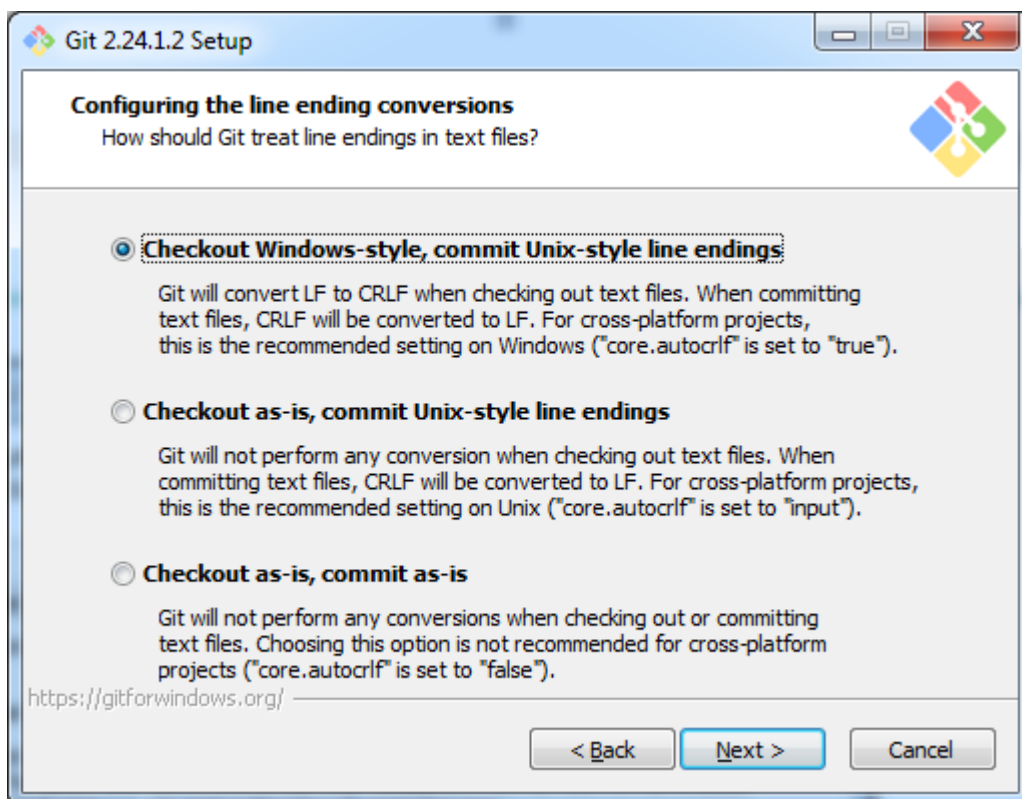
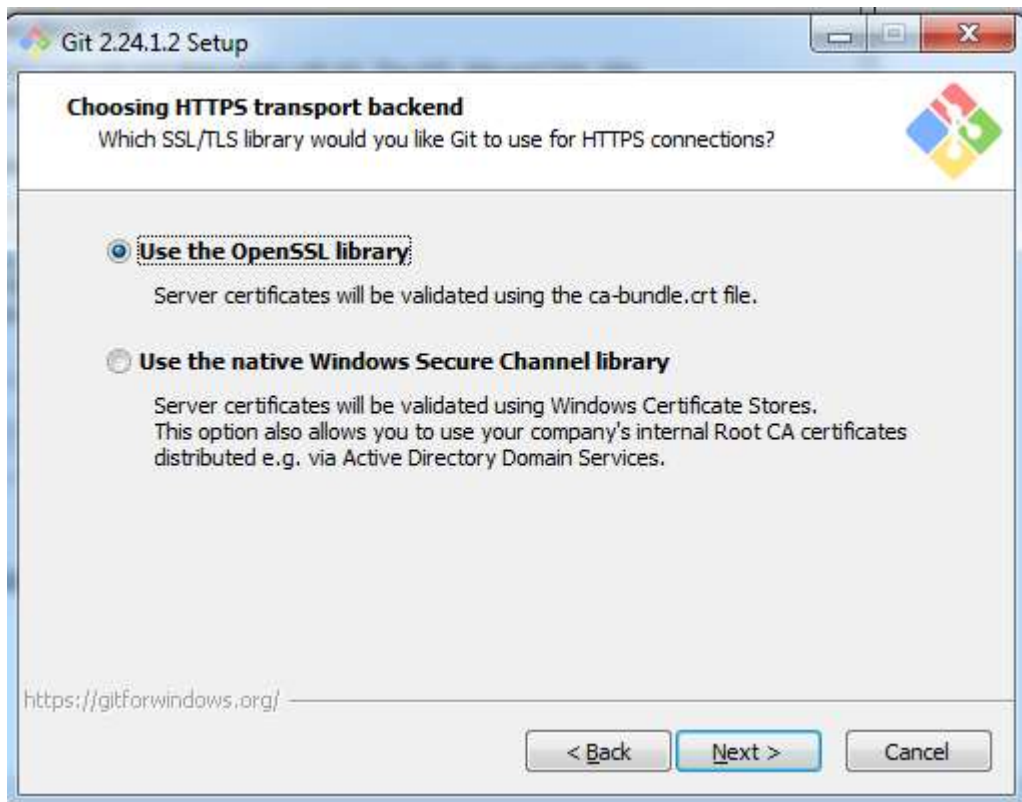


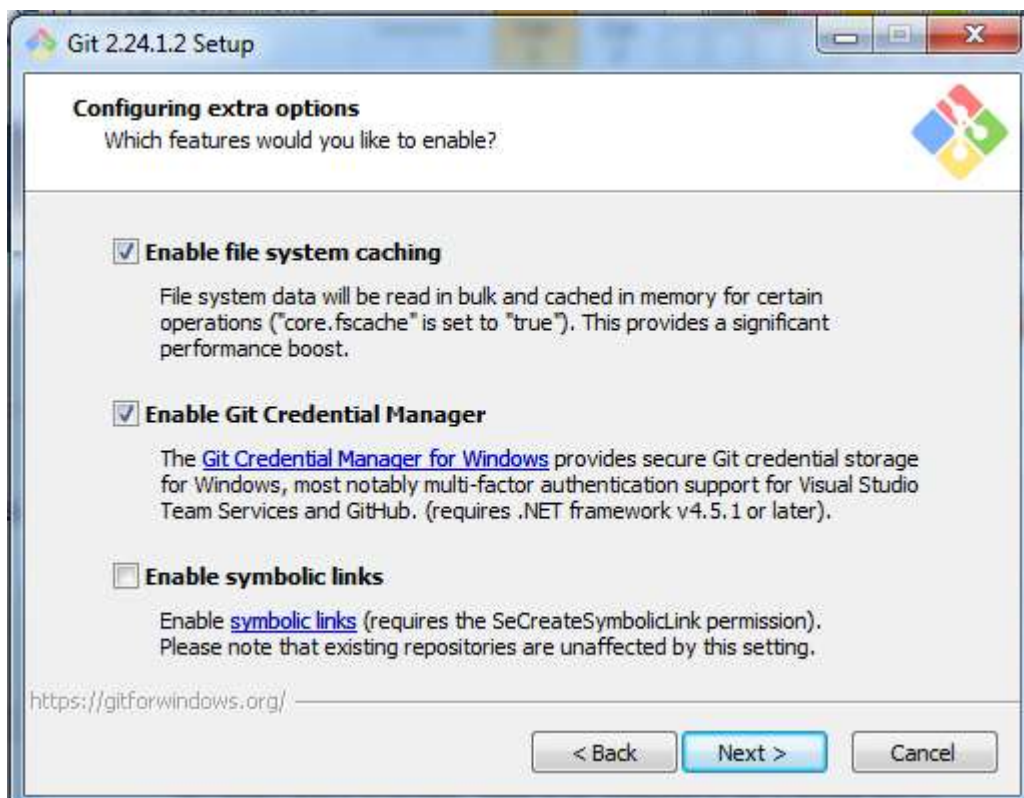
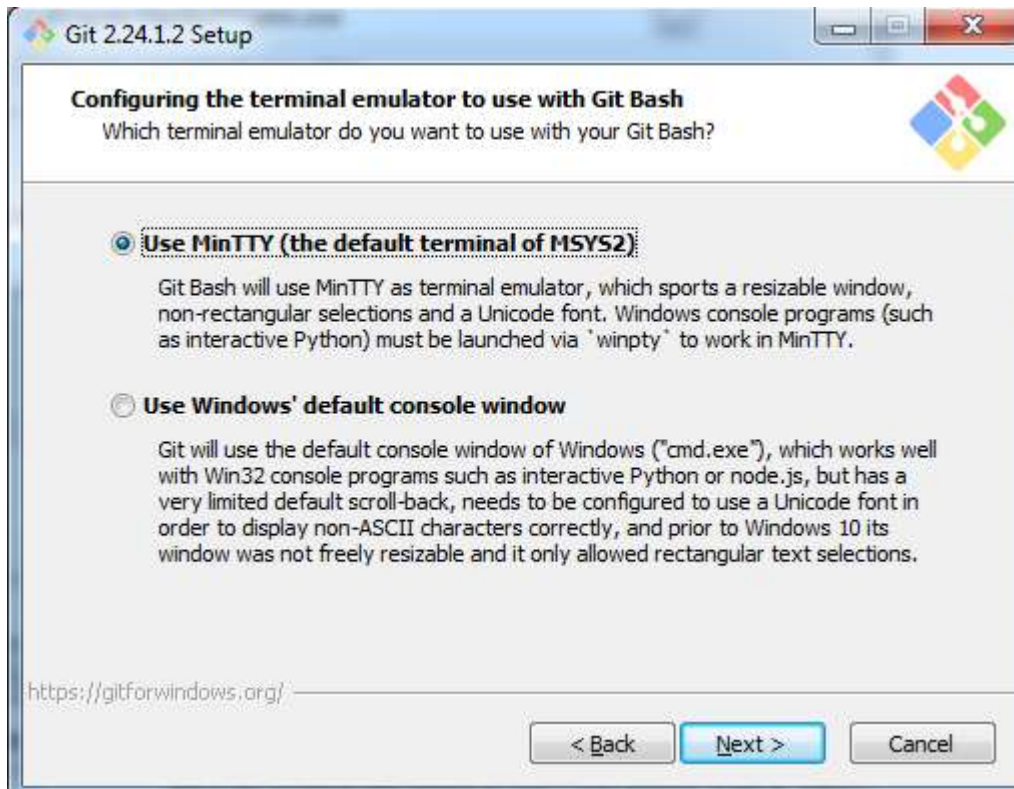


Mantemos também a opção default nas telas seguintes:

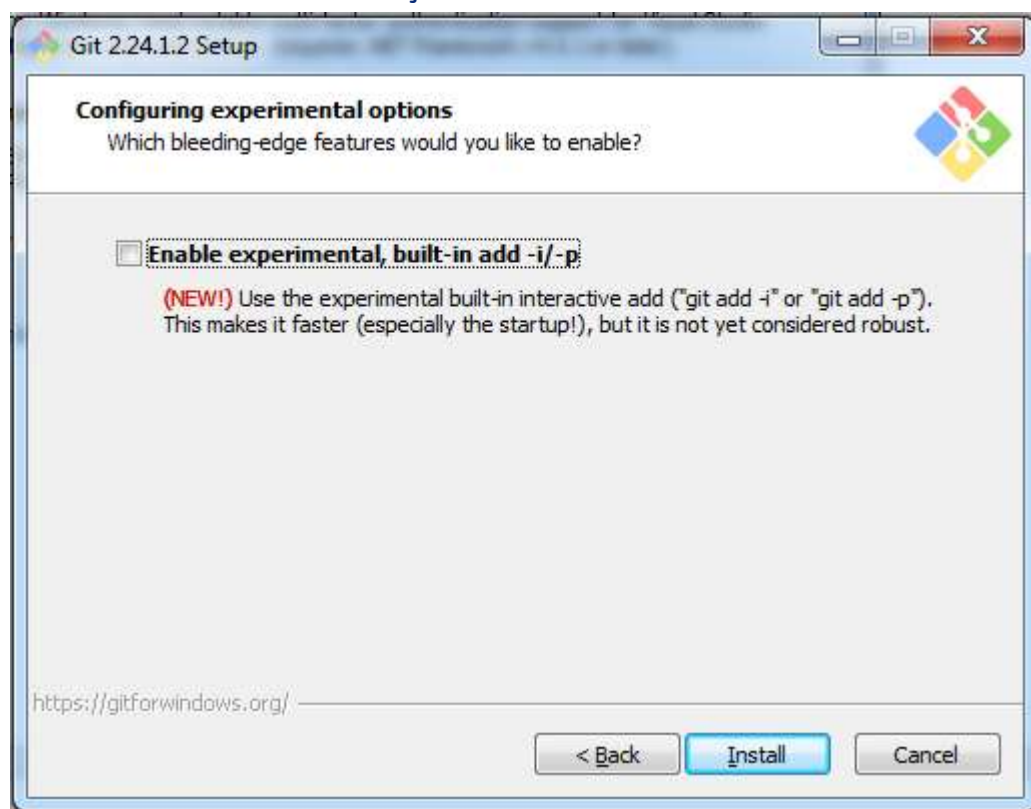




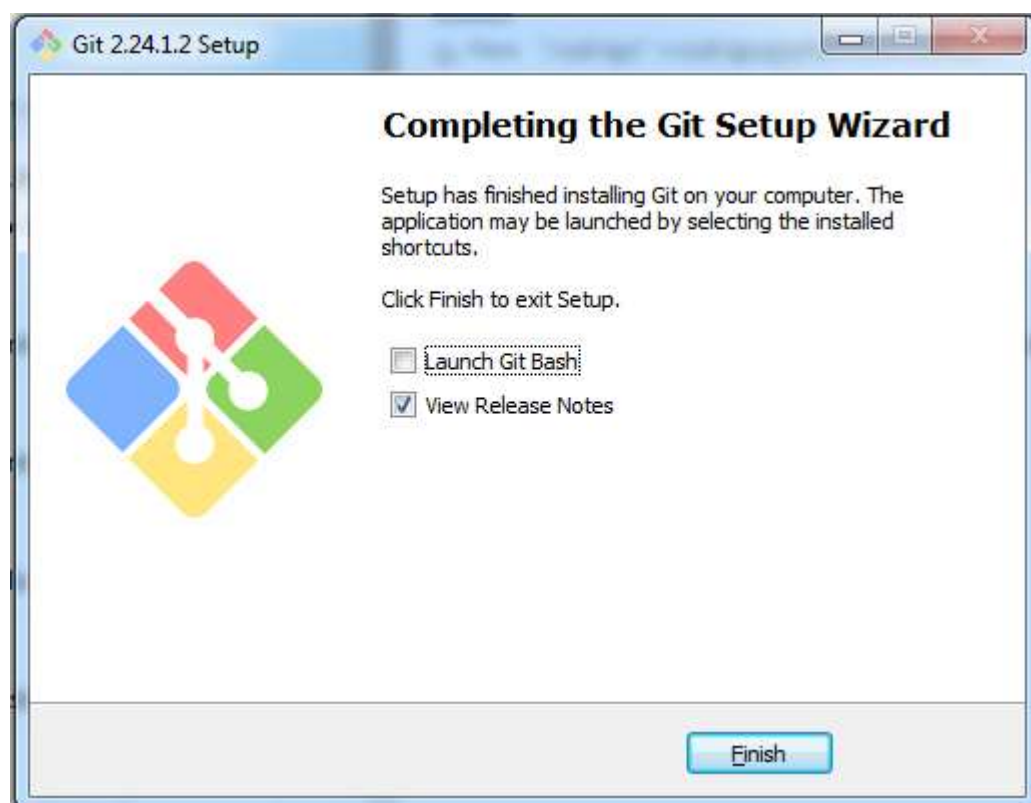




Ao final confirme a instalação:



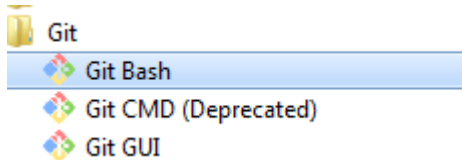
Após temos a janela de instalação concluída. Na unidade seguinte veremos a configuração do Git tanto em Windows quanto Linux:



---

# Configurando o Git

Faremos a configuração do Git por linha de comando. Os comandos são exatamente os mesmos no Windows, quando chamamos o Git Bash, como na figura a seguir, e no Linux, que acessaremos pelo Putty como mostrado nos ebooks de configuração do Linux.

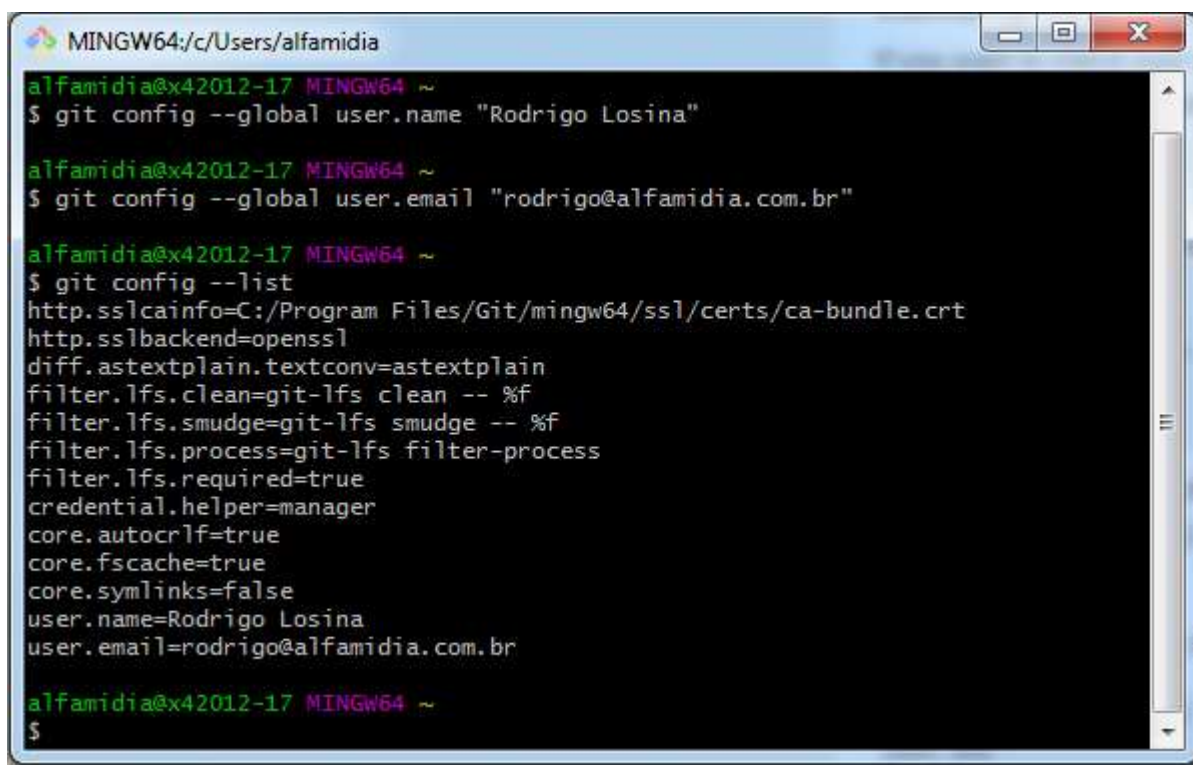


Em ambos vamos definir nosso nome e e-mail. Estas informações serão úteis quando trabalharmos com equipes para definir quem realizou qual modificação em determinada versão dos arquivos.

Utilize os seguintes comandos:

```
$ git config --global user.name "Seu nome"
$ git config --global user.email seu-email@dominido.com.br
```

Utilizando o comando “git config –list” você obtém a lista de configurações do Git, incluindo seu nome e e-mail recém definidos.



```
MINGW64:/c/Users/alfamidia
alfamidia@x42012-17 MINGW64 ~
$ git config --global user.name "Rodrigo Losina"

alfamidia@x42012-17 MINGW64 ~
$ git config --global user.email "rodrigo@alfamidia.com.br"

alfamidia@x42012-17 MINGW64 ~
$ git config --list
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
core.autocrlf=true
core.fscache=true
core.symlinks=false
user.name=Rodrigo Losina
user.email=rodrigo@alfamidia.com.br

alfamidia@x42012-17 MINGW64 ~
$
```

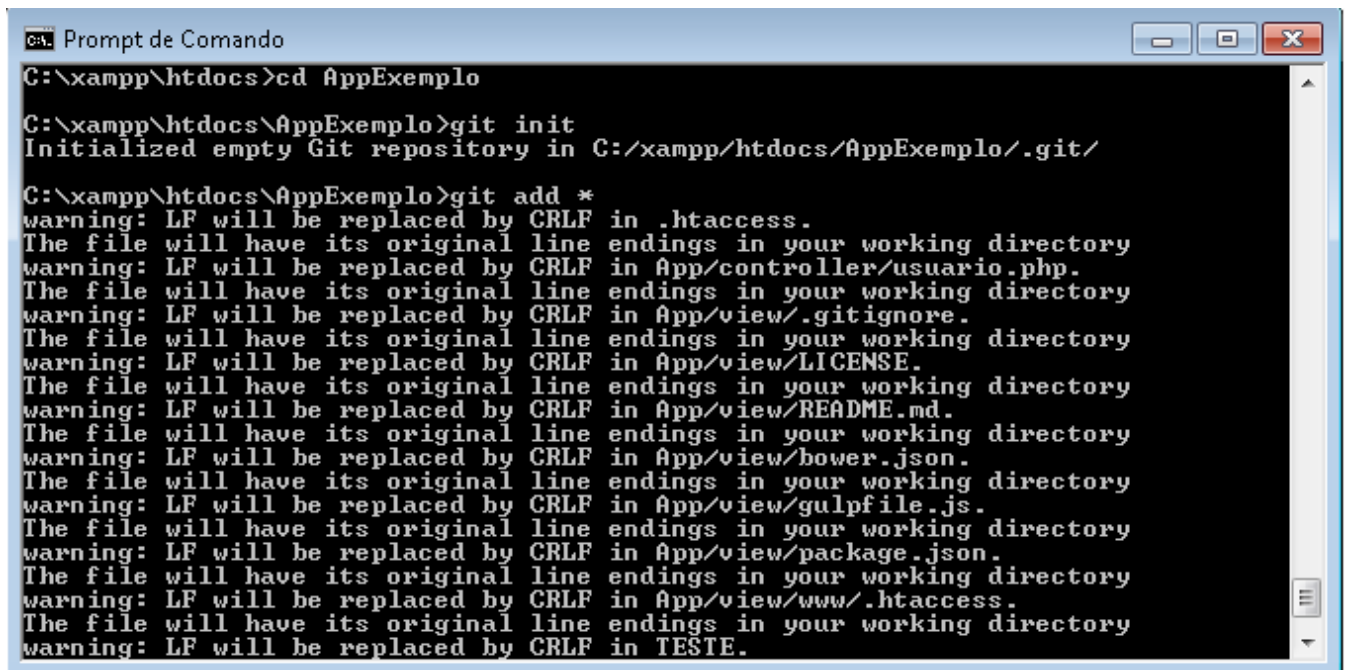
# Criando um Repositório

Neste exemplo, mostraremos a criação de um repositório Git em uma máquina Windows. Os comandos serão equivalentes em uma máquina Linux, alterando apenas o diretório em que o repositório será criado.

Em nosso exemplo, o código-fonte de nossa aplicação encontra-se em `c:\xampp\htdocs\AppExemplo`. O que nós faremos é chamar nosso console, ir para a pasta da aplicação e executar os comandos de inicialização do git (`git init`) e de inclusão de todos os arquivos nos arquivos rastreados pelo git.

Observe e reproduza a sequência de comandos a seguir, observe a imagem, e a descrição a seguir, do que realizamos:

```
cd c:\xampp\htdocs\AppExemplo
git init
git add *
git commit -m "primeiro commit"
```

A screenshot of a Windows Command Prompt window titled "Prompt de Comando". The window shows the following commands and their output:  
1. `C:\xampp\htdocs>cd AppExemplo`  
2. `C:\xampp\htdocs\AppExemplo>git init`  
Output: `Initialized empty Git repository in C:/xampp/htdocs/AppExemplo/.git/`  
3. `C:\xampp\htdocs\AppExemplo>git add *`  
Output: A series of warnings for each file being added, indicating that LF line endings will be replaced by CRLF in the repository. The files listed are: `.htaccess`, `App/controller/usuario.php`, `App/view/.gitignore`, `App/view/LICENSE`, `App/view/README.md`, `App/view/bower.json`, `App/view/gulpfile.js`, `App/view/package.json`, and `App/view/www/.htaccess`. The final line of output is `warning: LF will be replaced by CRLF in TESTE.`

O primeiro comando nos leva a pasta correspondente a nossa aplicação.



---

Com o comando *git init* o que temos é a criação de uma pasta chamada `.git`, em que serão armazenadas as cópias com controle de versão dos arquivos a serem rastreados.

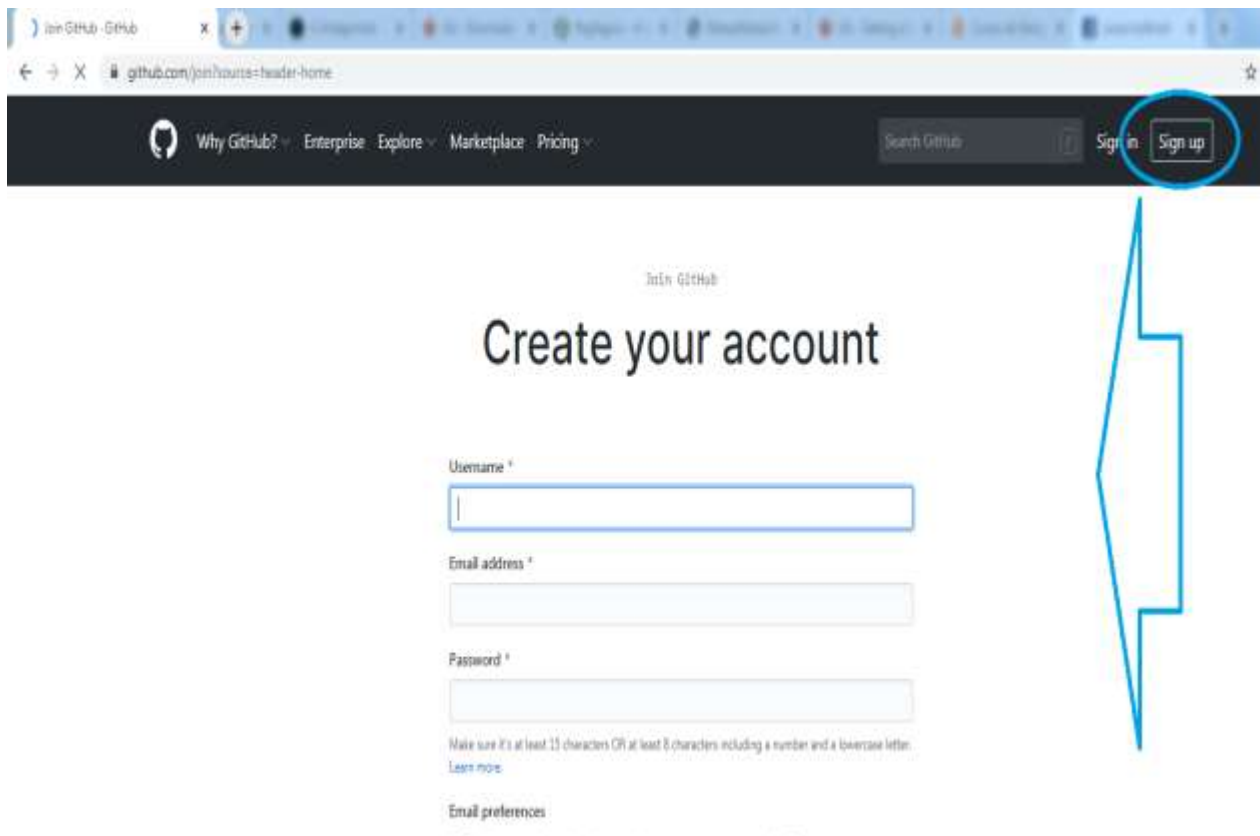
O comando *git add \** basicamente define que todos os arquivos da aplicação estão sendo rastreáveis. Esta é uma prática que requer uma certa atenção, pois eventuais arquivos específicos da configuração da máquina Windows, por exemplo, não deveriam ser rastreados junto, mas não é objetivo deste ebook entrar neste nível de detalhe.

Por fim, o comando *git commit* encerra nossas modificações, permitindo que sejam transferidas para um repositório geral.

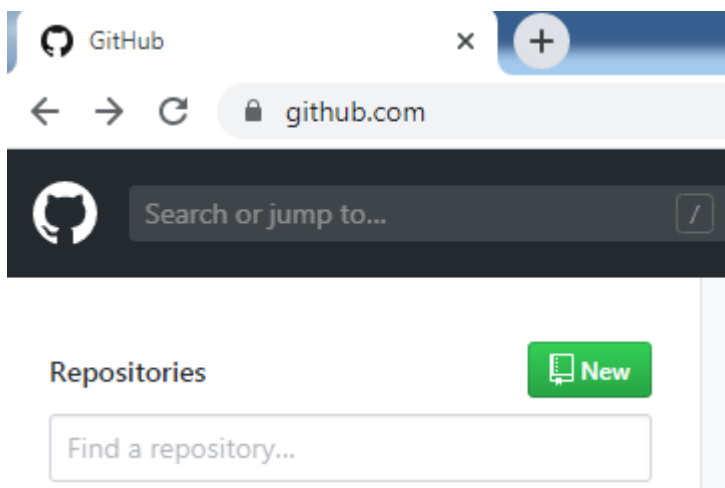


# Criando um Repositório no GitHub

O primeiro passo para criar um repositório no GitHub é criar uma conta. Este é um passo simples e bastante semelhante a criação de contas em redes sociais e outras plataformas de serviços online. Entre em GitHub e crie uma conta gratuita clicando em *Sign up* - <https://github.com/>

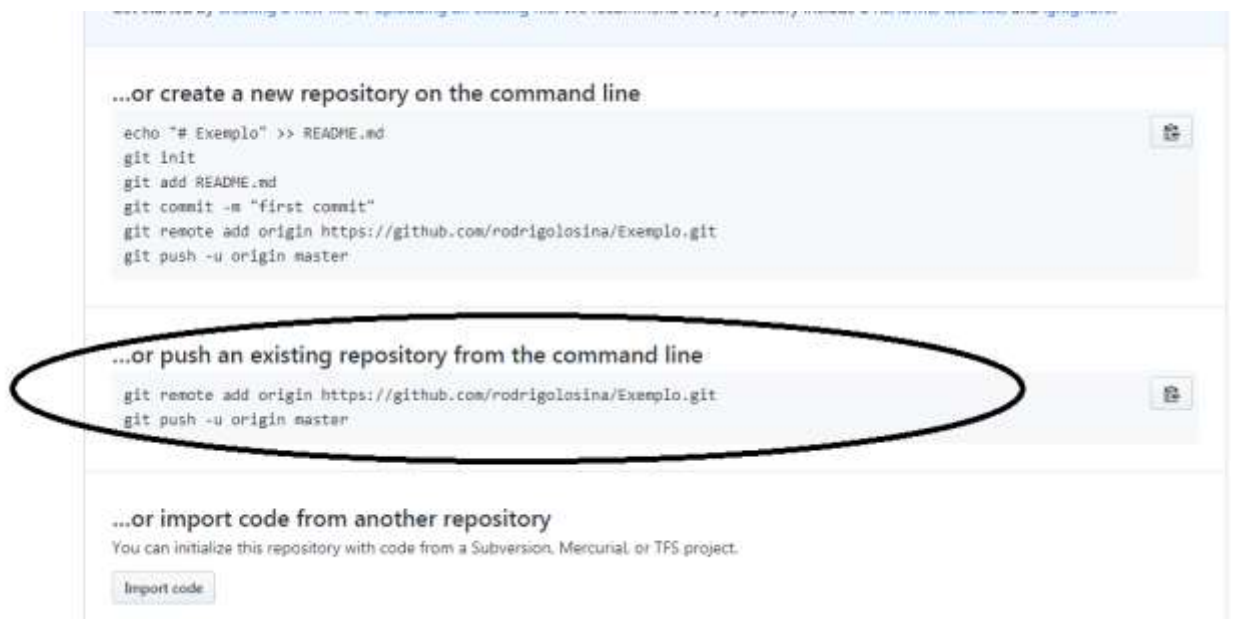


Ao se logar, você poderá criar um novo repositório Git diretamente pela interface:



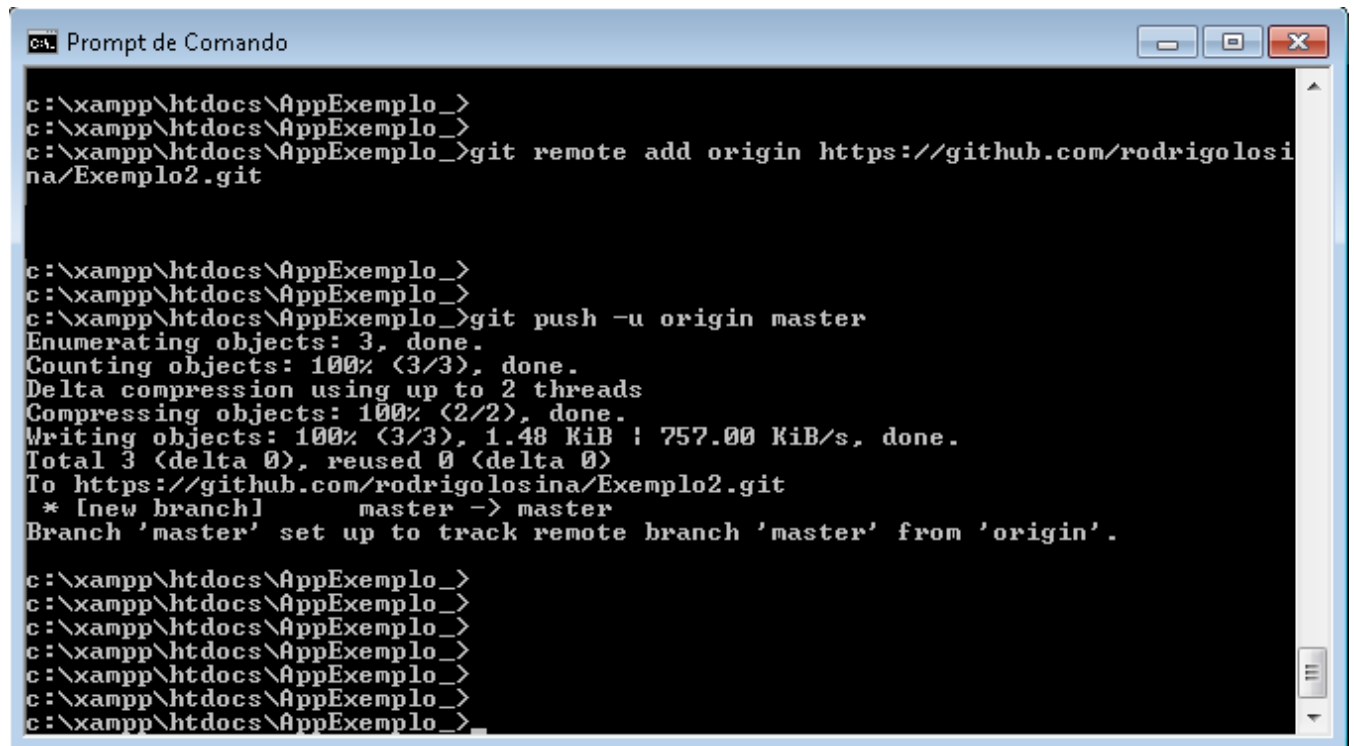
Para este ebook, utilizaremos o nome Exemplo para o repositório criado. Observe que se ele for público, outras pessoas terão acesso ao código que você vai transferir.

Após a criação do repositório, a própria interface nos mostra o comando necessário para transferir para ele nosso repositório criado em nossa máquina local.



Retornando ao nosso shell da máquina Windows, podemos executar estes dois comandos para transferir nossos arquivos para o GitHub:

```
git remote add origin https://github.com/rodrigolosina/Exemplo.git
git push -u origin master
```



The screenshot shows a Windows Command Prompt window titled "Prompt de Comando". The window has a standard Windows title bar with minimize, maximize, and close buttons. The command history is as follows:

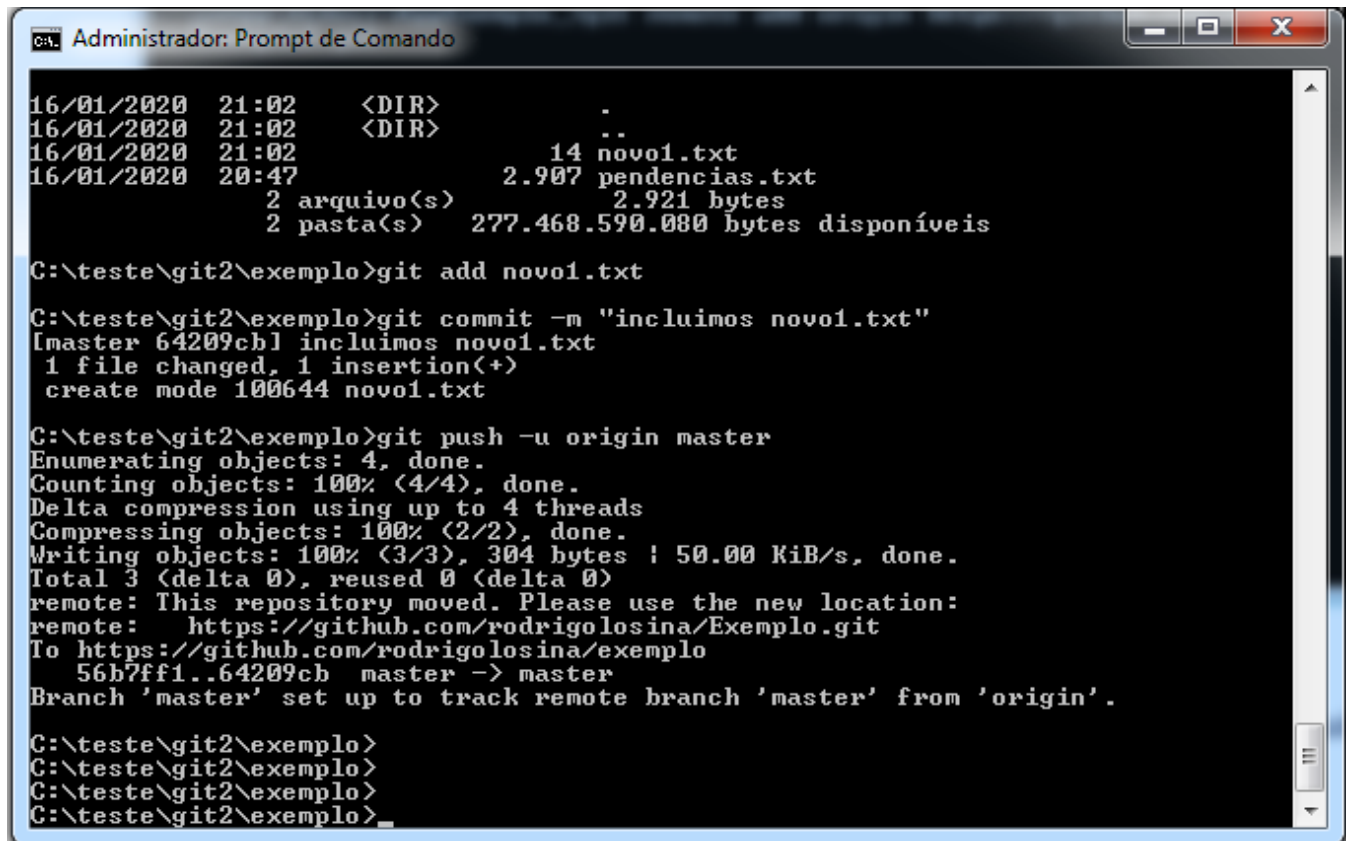
```
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>git remote add origin https://github.com/rodrigolosina/Exemplo2.git

c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 1.48 KiB | 757.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/rodrigolosina/Exemplo2.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
c:\xampp\htdocs\AppExemplo_>
```

Por fim, repetimos o comando visto anteriormente para enviar o repositório atualizado para o GitHub:

*git push -u origin master*



```
C:\> Administrador: Prompt de Comando

16/01/2020 21:02 <DIR> .
16/01/2020 21:02 <DIR> ..
16/01/2020 21:02          14 novo1.txt
16/01/2020 20:47        2.907 pendencias.txt
                2 arquivo(s)        2.921 bytes
                2 pasta(s) 277.468.590.080 bytes disponíveis

C:\teste\git2\exemplo>git add novo1.txt

C:\teste\git2\exemplo>git commit -m "incluimos novo1.txt"
[master 64209cb] incluimos novo1.txt
 1 file changed, 1 insertion(+)
 create mode 100644 novo1.txt

C:\teste\git2\exemplo>git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 50.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: This repository moved. Please use the new location:
remote:   https://github.com/rodrigolosina/Exemplo.git
To https://github.com/rodrigolosina/exemplo
 56b7ff1..64209cb master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

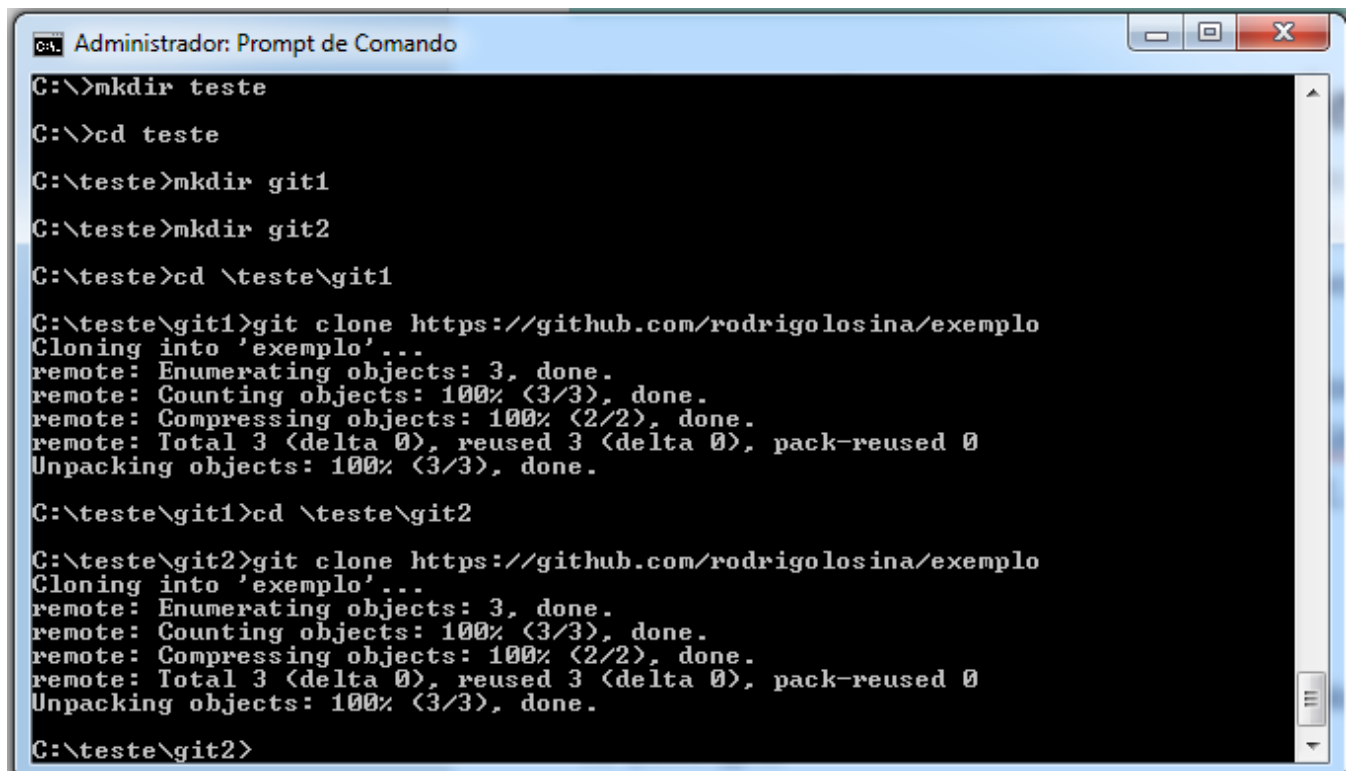
C:\teste\git2\exemplo>
C:\teste\git2\exemplo>
C:\teste\git2\exemplo>
C:\teste\git2\exemplo>
```

# Clonando um Repositório Git

Após criar um repositório Git em uma máquina e transferi-lo para o GitHub, o que devemos fazer é criar um clone deste repositório nas demais máquinas em que estaremos trabalhando com versões destes arquivos.

Para praticar os recursos do Git, você pode criar dois diretórios em sua máquina local, e testar nestes diretórios o controle de versões do Git. Faremos isto no exemplo a seguir criando dois diretórios `c:\teste\git1` e `c:\teste\git2`, e clonando nosso repositório para ambos os locais, com os comandos a seguir:

```
cd \teste\git1
git clone https://github.com/rodrigolosina/exemplo
cd \teste\git2
git clone https://github.com/rodrigolosina/exemplo
```



```
C:\>mkdir teste
C:\>cd teste
C:\teste>mkdir git1
C:\teste>mkdir git2
C:\teste>cd \teste\git1
C:\teste\git1>git clone https://github.com/rodrigolosina/exemplo
Cloning into 'exemplo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
C:\teste\git1>cd \teste\git2
C:\teste\git2>git clone https://github.com/rodrigolosina/exemplo
Cloning into 'exemplo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
C:\teste\git2>
```

Desta forma, neste momento temos dois repositórios git, em `git1\exemplo` e `git2\exemplo`, ambos sincronizados com nosso repositório exemplo no GitHub.

# Criando um Novo Arquivo e Incluindo no Repositório

Nós iremos, agora, criar um novo arquivo em nosso diretório git2 e iremos incluí-lo no repositório Git.

Naturalmente, em uma situação real, iríamos abrir um editor de texto ou uma IDE para criar este arquivo, mas, como é apenas um teste, utilizaremos o seguinte comando no próprio shell ou prompt de comando para criar um arquivo novo1.txt, que contém apenas o texto “ola mundo”, conforme a figura a seguir.

*echo “ola mundo” > novo1.txt*



```
Administrador: Prompt de Comando

C:\teste\git2\exemplo>dir
O volume na unidade C não tem nome.
O Número de Série do Volume é DC6D-BA18

Pasta de C:\teste\git2\exemplo
16/01/2020  20:47    <DIR>          .
16/01/2020  20:47    <DIR>          ..
16/01/2020  20:47                2.907 pendencias.txt
                1 arquivo(s)          2.907 bytes
                2 pasta(s)      277.468.590.080 bytes disponíveis

C:\teste\git2\exemplo>echo "ola mundo" > novo1.txt

C:\teste\git2\exemplo>dir
O volume na unidade C não tem nome.
O Número de Série do Volume é DC6D-BA18

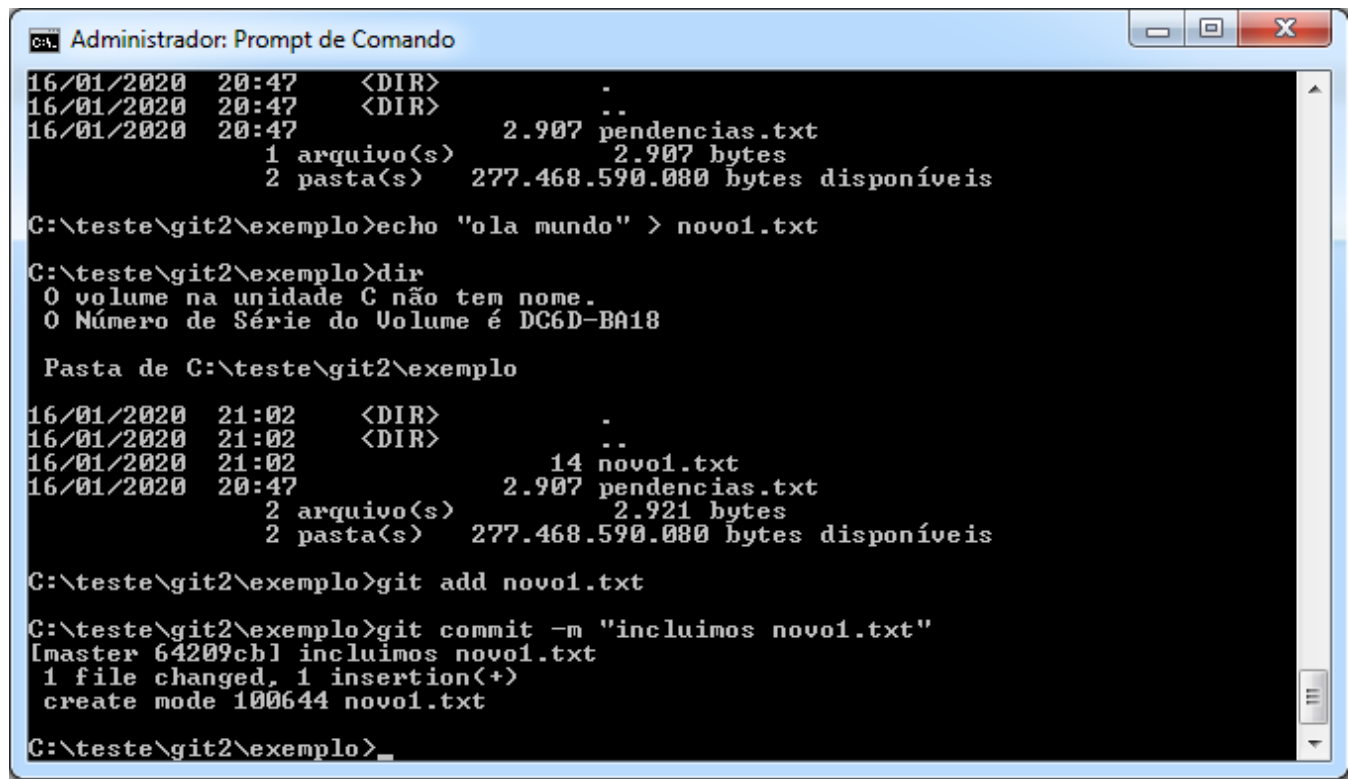
Pasta de C:\teste\git2\exemplo
16/01/2020  21:02    <DIR>          .
16/01/2020  21:02    <DIR>          ..
16/01/2020  21:02                14 novo1.txt
16/01/2020  20:47                2.907 pendencias.txt
                2 arquivo(s)          2.921 bytes
                2 pasta(s)      277.468.590.080 bytes disponíveis

C:\teste\git2\exemplo>
```

Agora, com os dois comandos seguintes, nós iremos incluir o arquivo novo1.txt no controle de versões, e realizar um commit, de forma a estarmos prontos para fazer o envio do repositório atualizado para o GitHub.

`git add novo1.txt`

`git commit -m "incluímos novo1.txt"`



```
C:\> Administrador: Prompt de Comando

16/01/2020 20:47 <DIR> .
16/01/2020 20:47 <DIR> ..
16/01/2020 20:47          2.907 pendencias.txt
                1 arquivo(s)          2.907 bytes
                2 pasta(s) 277.468.590.080 bytes disponíveis

C:\teste\git2\exemplo>echo "ola mundo" > novo1.txt

C:\teste\git2\exemplo>dir
0 volume na unidade C não tem nome.
0 Número de Série do Volume é DC6D-BA18

Pasta de C:\teste\git2\exemplo
16/01/2020 21:02 <DIR> .
16/01/2020 21:02 <DIR> ..
16/01/2020 21:02          14 novo1.txt
16/01/2020 20:47          2.907 pendencias.txt
                2 arquivo(s)          2.921 bytes
                2 pasta(s) 277.468.590.080 bytes disponíveis

C:\teste\git2\exemplo>git add novo1.txt

C:\teste\git2\exemplo>git commit -m "incluimos novo1.txt"
[master 64209cbl incluimos novo1.txt
 1 file changed, 1 insertion(+)
 create mode 100644 novo1.txt

C:\teste\git2\exemplo>
```

O parâmetro -m permite definirmos na própria linha de comando a mensagem associada a esta versão do repositório.

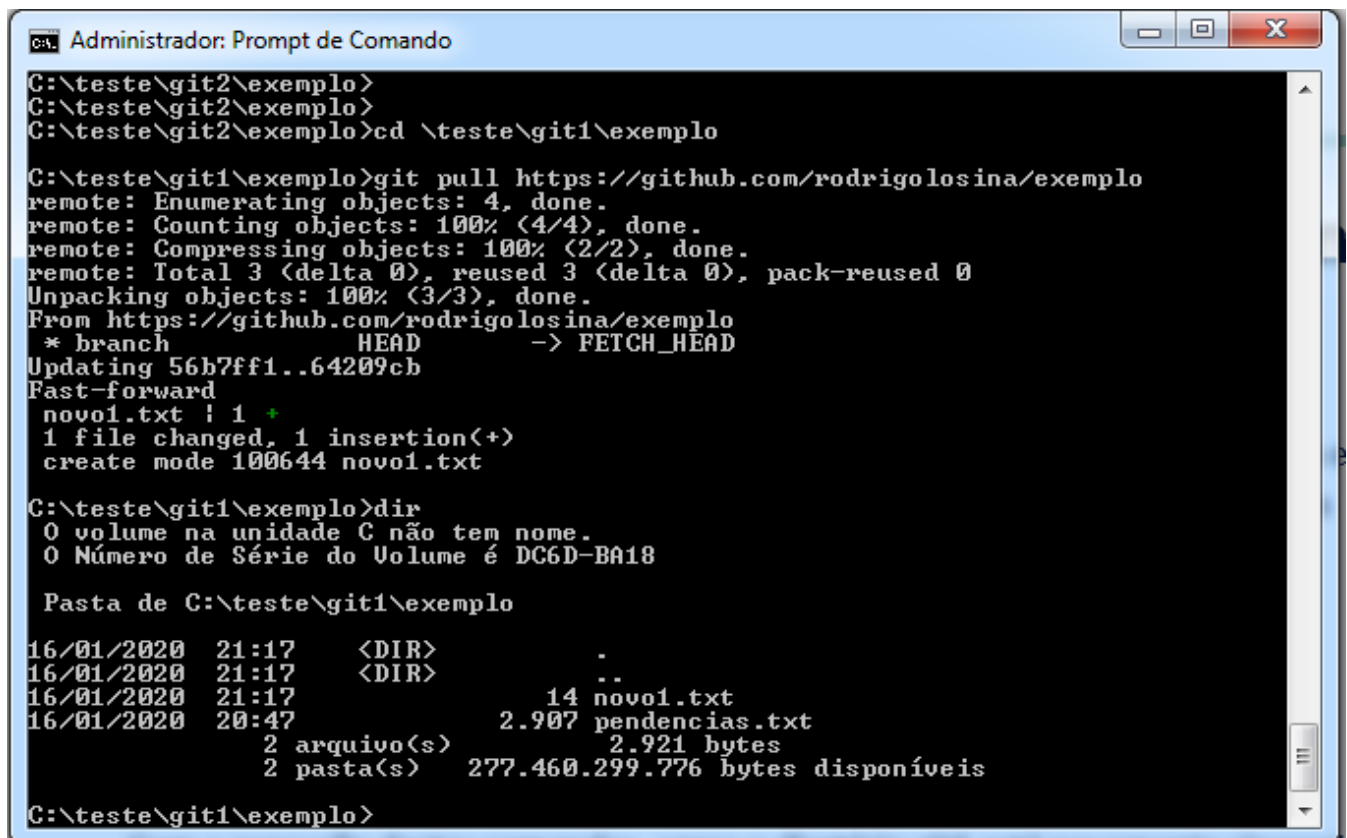


# Sincronizando Repositório Local com GitHub

Seguindo em nosso exemplo, iremos agora para o `git1\exemplo` e atualizaremos este diretório com o GitHub (portanto obtendo o arquivo `novo1.txt` que foi criado em `git2\exemplo`).

```
cd \teste\git1\exemplo
git pull https://github.com/rodrigolosina/exemplo
```

Com a execução destes comandos, agora o diretório `git1` está novamente sincronizado com o GitHub.



```
Administrador: Prompt de Comando
C:\teste\git2\exemplo>
C:\teste\git2\exemplo>
C:\teste\git2\exemplo>cd \teste\git1\exemplo

C:\teste\git1\exemplo>git pull https://github.com/rodrigolosina/exemplo
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/rodrigolosina/exemplo
 * branch            HEAD       -> FETCH_HEAD
Updating 56b7ff1..64209cb
Fast-forward
 novo1.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 novo1.txt

C:\teste\git1\exemplo>dir
O volume na unidade C não tem nome.
O Número de Série do Volume é DC6D-BA18

Pasta de C:\teste\git1\exemplo

16/01/2020  21:17    <DIR>          .
16/01/2020  21:17    <DIR>          ..
16/01/2020  21:17                  14 novo1.txt
16/01/2020  20:47             2.907 pendencias.txt
                2 arquivo(s)          2.921 bytes
                2 pasta(s)       277.460.299.776 bytes disponíveis

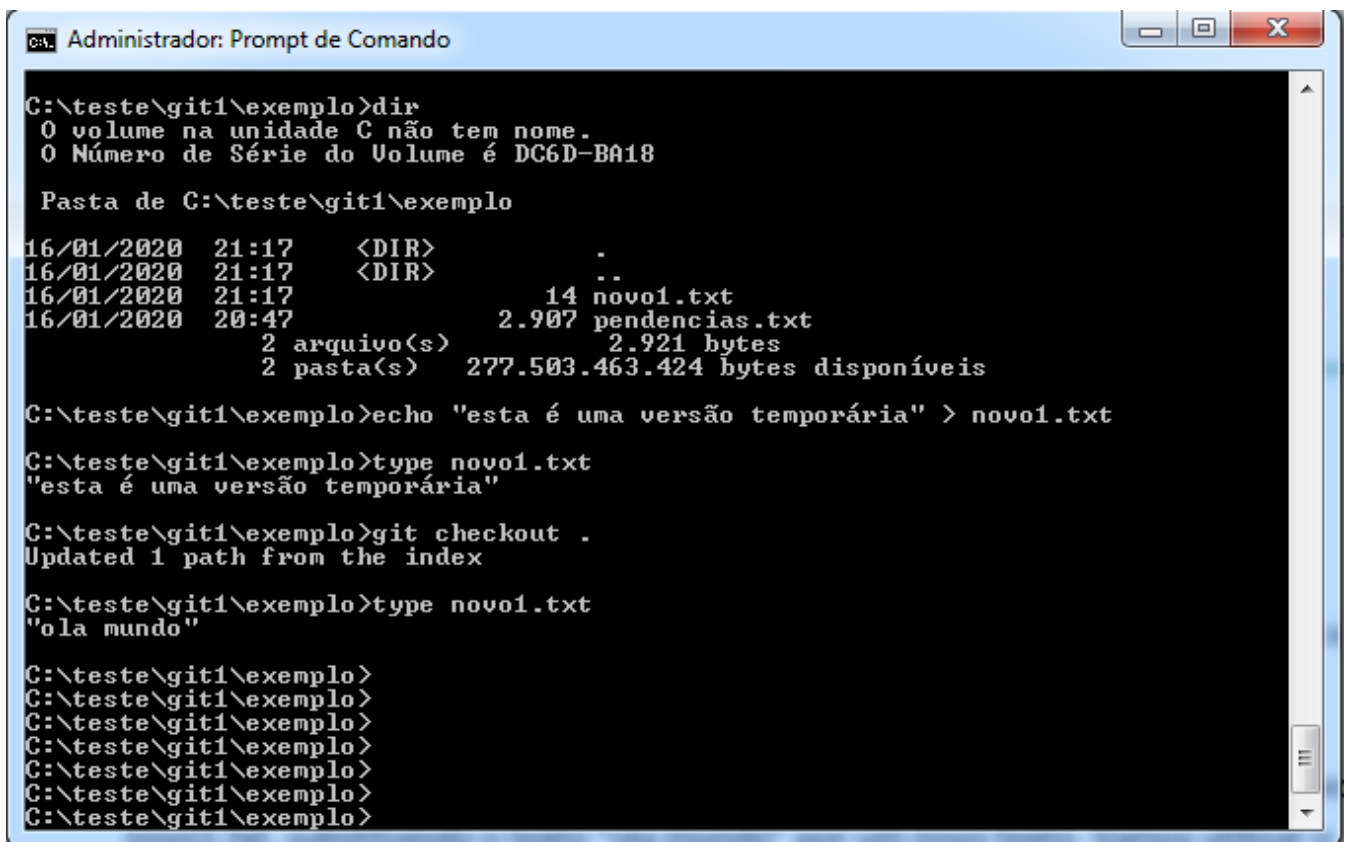
C:\teste\git1\exemplo>
```

# Retornando para Versão Anterior

Uma das principais vantagens do controle de versões é podermos retornar para uma versão anterior, desfazendo modificações que tenhamos feito.

Vamos exemplificar esta ação com os comandos a seguir. Vamos alterar o arquivo novo1.txt, substituindo o texto “ola mundo” por um novo texto. Vamos utilizar o comando *type* (no Windows) para mostrar o conteúdo. Após vamos utilizar o comando *git* para retornar para o último commit:

```
echo "esta é uma versão temporária" > novo1.txt
type novo1.txt
git checkout .
type novo1.txt
```



```
Administrador: Prompt de Comando

C:\teste\git1\exemplo>dir
O volume na unidade C não tem nome.
O Número de Série do Volume é DC6D-BA18

Pasta de C:\teste\git1\exemplo

16/01/2020  21:17    <DIR>          .
16/01/2020  21:17    <DIR>          ..
16/01/2020  21:17                14 novo1.txt
16/01/2020  20:47            2.907 pendencias.txt
                2 arquivo(s)        2.921 bytes
                2 pasta(s)    277.503.463.424 bytes disponíveis

C:\teste\git1\exemplo>echo "esta é uma versão temporária" > novo1.txt

C:\teste\git1\exemplo>type novo1.txt
"esta é uma versão temporária"

C:\teste\git1\exemplo>git checkout .
Updated 1 path from the index

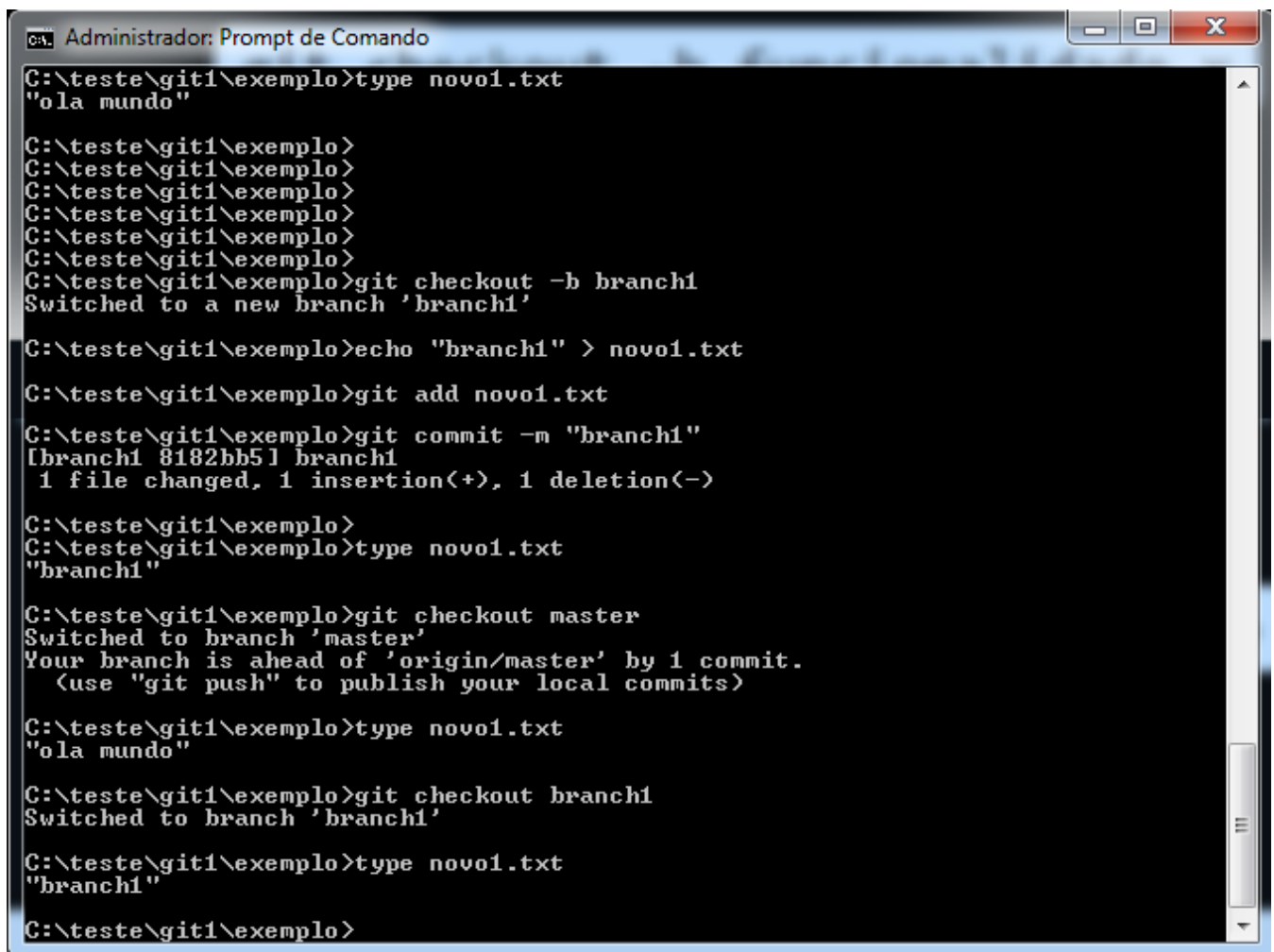
C:\teste\git1\exemplo>type novo1.txt
"ola mundo"

C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
```

# Criando um Branch

Um *Branch* ou *Ramo* é um desvio em nosso controle de versões, o que nos permite trabalhar com versões em separado, que, no futuro, poderemos mesclar novamente. Observe no exemplo a seguir, que criamos uma branch chamada `branch1`.

```
git checkout -b branch1
echo "branch1" > novo1.txt
git add novo1.txt
git commit -m "branch1"
type novo1.txt
git checkout master
type novo1.txt
```



```
C:\teste\git1\exemplo>type novo1.txt
"ola mundo"

C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>
C:\teste\git1\exemplo>git checkout -b branch1
Switched to a new branch 'branch1'

C:\teste\git1\exemplo>echo "branch1" > novo1.txt

C:\teste\git1\exemplo>git add novo1.txt

C:\teste\git1\exemplo>git commit -m "branch1"
[branch1 8182bb5] branch1
1 file changed, 1 insertion(+), 1 deletion(-)

C:\teste\git1\exemplo>
C:\teste\git1\exemplo>type novo1.txt
"branch1"

C:\teste\git1\exemplo>git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

C:\teste\git1\exemplo>type novo1.txt
"ola mundo"

C:\teste\git1\exemplo>git checkout branch1
Switched to branch 'branch1'

C:\teste\git1\exemplo>type novo1.txt
"branch1"

C:\teste\git1\exemplo>
```

---

O comando *git checkout -b branch1* cria uma nova “branch” chamada “branch1”. Após, comandos *git checkout master* e *git checkout branch1* permitem alterar entre a branch principal e a recém criada. O exemplo acima ilustra que se tratam de versões diferentes mostrando conteúdos diferentes do arquivo novo1.txt.

Por fim, o comando a seguir envia a “branch1” para nosso repositório no GitHub.

*Git push origin branch1*