

# **Guia Básico Controle de Versão com Git e Github no Power BI**



**git**



**GitHub**

# Sumário

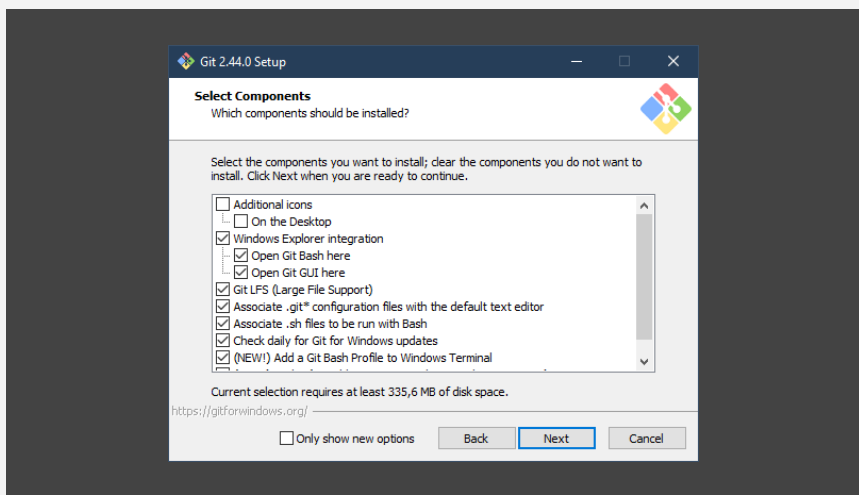
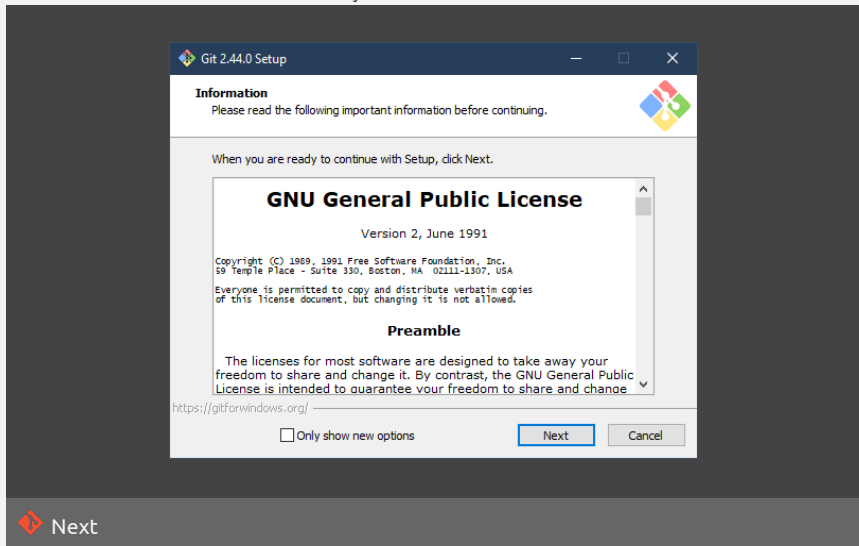
Sumário.....	2
Instalação do Git.....	3
Criação de um repositório Git.....	4
Adicionar arquivos ao repositório.....	5
Confirmar as alterações.....	6
Conectar-se a um repositório remoto.....	7
Enviar as alterações para o repositório remoto.....	8
Sincronização com o repositório remoto.....	9



# Instalação do Git

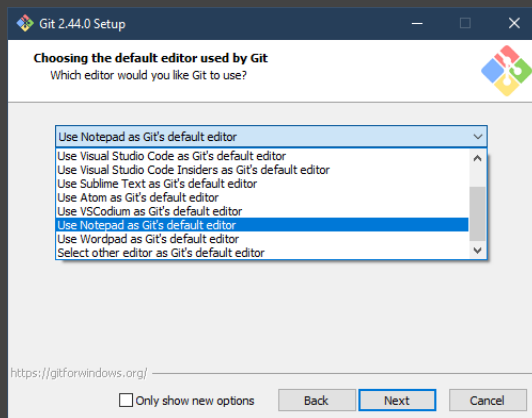
Se ainda não tiver o Git instalado, faça o download e instale a partir do site oficial do Git (<https://git-scm.com/>).

Acesse o instalador e inicie a instalação conforme as telas abaixo.

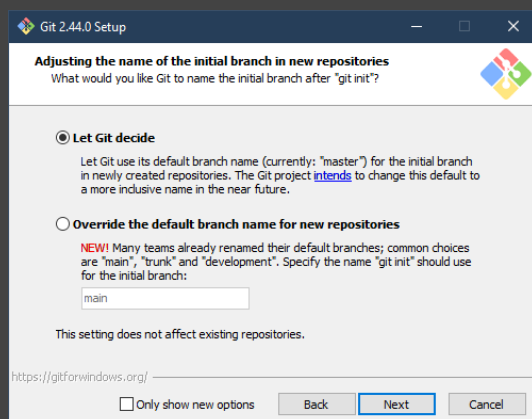




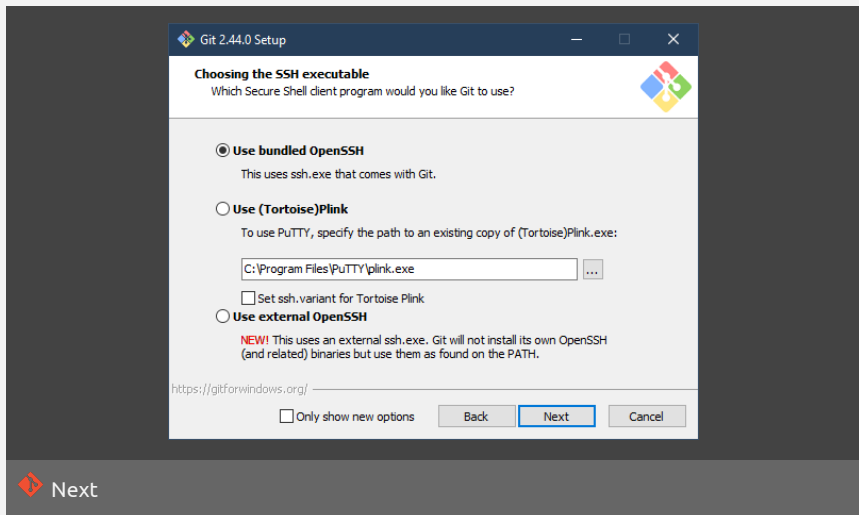
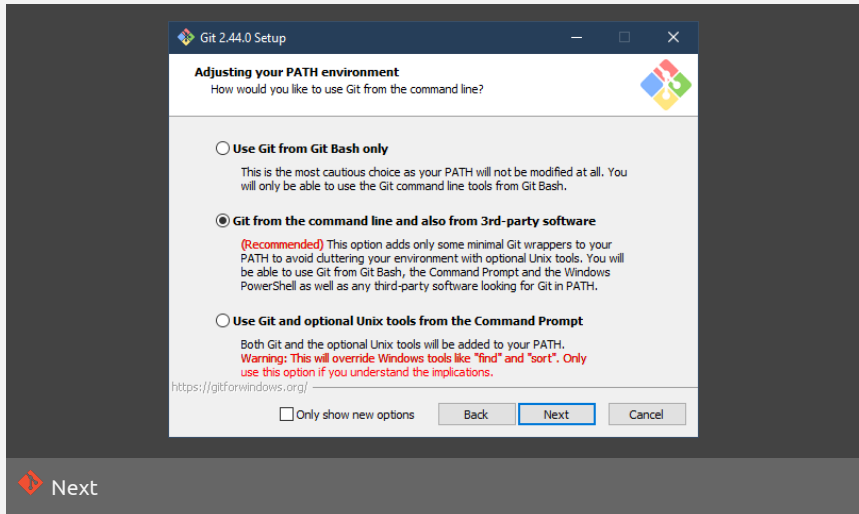
Next

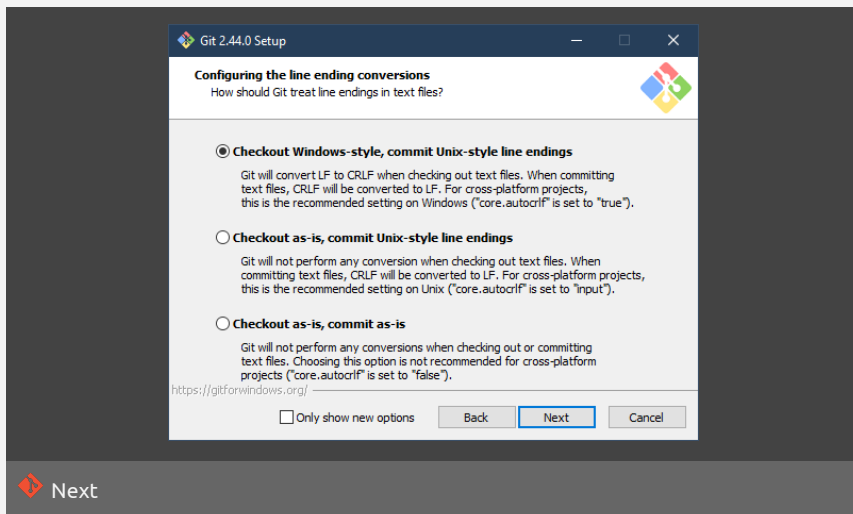
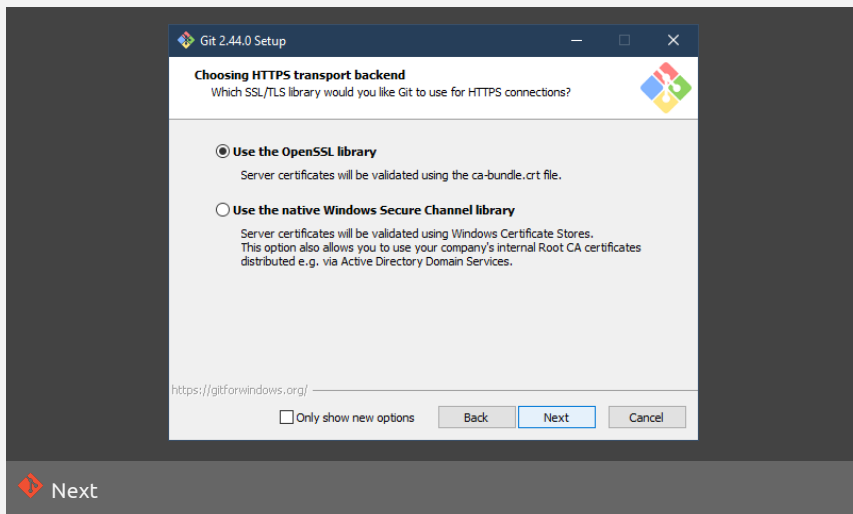


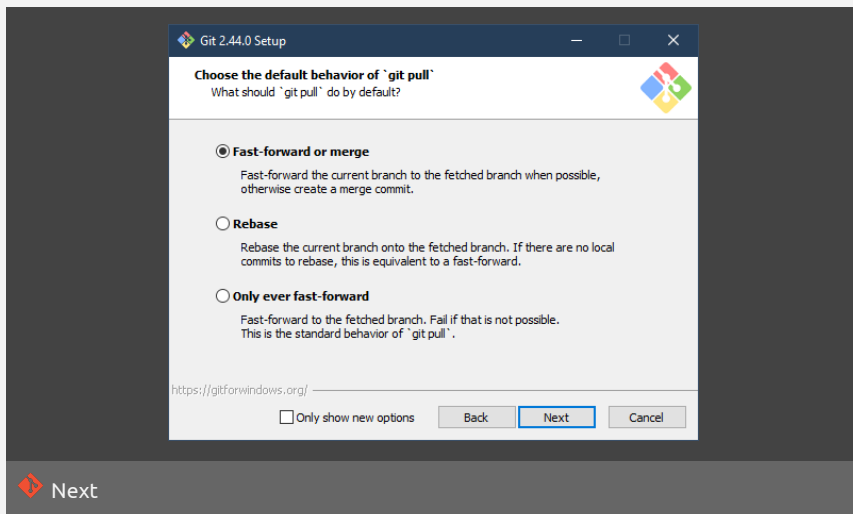
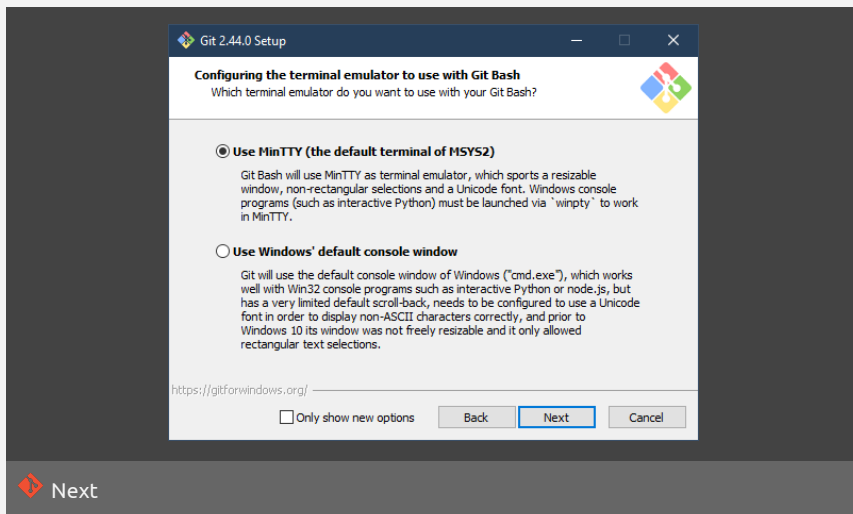
Aqui é possível selecionar o editor de preferência para editar os arquivos Git.

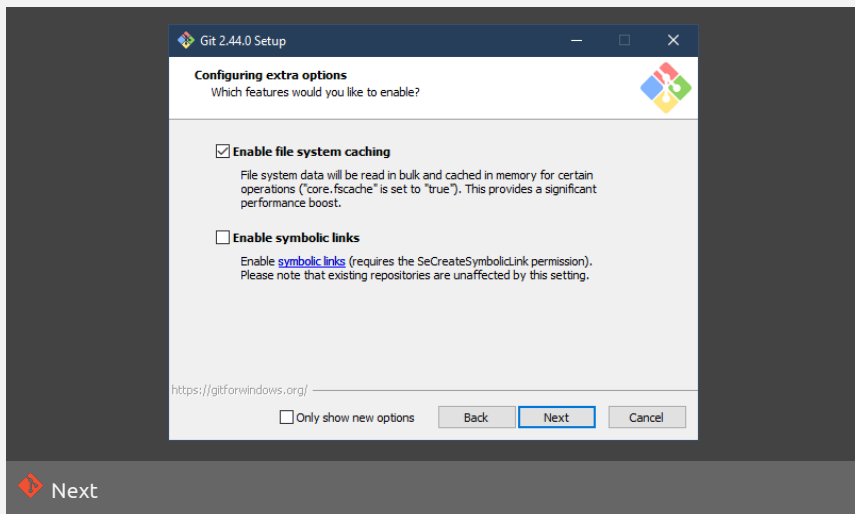
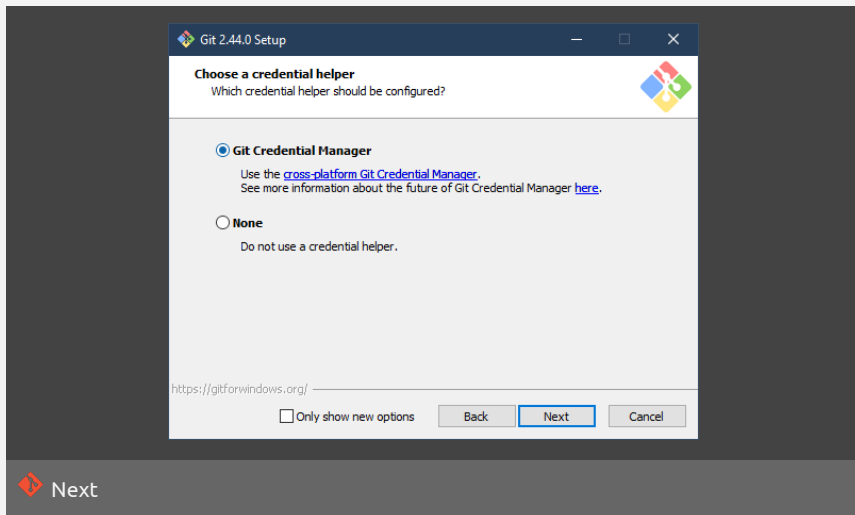


Aqui é possível definir a branch principal, se não for definido nada será usado "master".

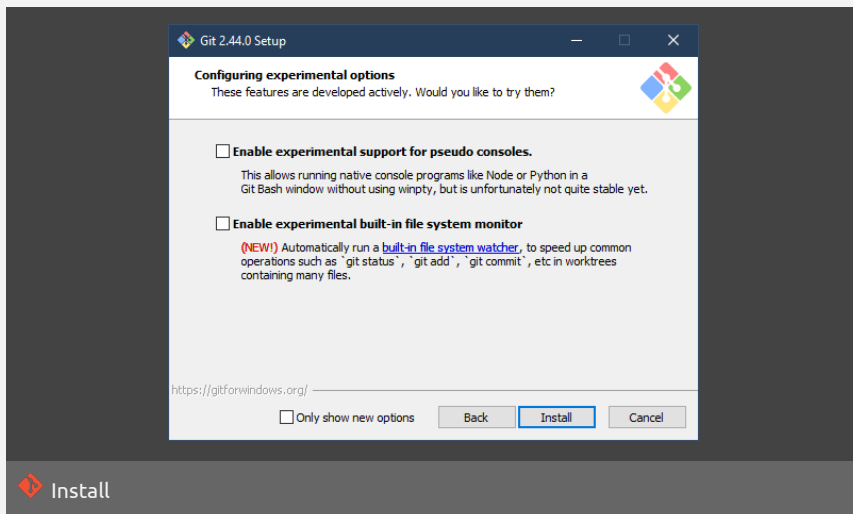












Com a instalação finalizada é possível seguir com a interface gráfica ou através do prompt de comando.

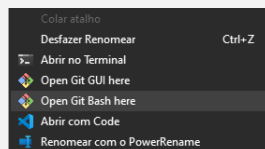


# Criação de um repositório Git

## Usando Prompt de Comando

Abra o Git Bash ou o terminal de sua preferência e navegue até o diretório onde seus arquivos do Power BI estão localizados. Use o comando “git init” para inicializar um repositório Git nesse diretório.

Caso não tenha familiaridade com prompt de comando basta abrir o “Explorer” no Windows e navegar até a pasta onde tenha os arquivos que queira versionar, clicar com o botão direito do mouse em uma área qualquer da pasta e selecionar “Open Git Bash here”.



Com o prompt aberto digite o comando “git init”.

```
MINGW64:/c:/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi
$ git init
Initialized empty Git repository in C:/GitPowerBi/.git/
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ _
```

Será criada uma pasta oculta com o nome “.git”.



# Adicionar arquivos ao repositório

O comando “git add” é usado para dizer ao Git quais arquivos você quer que ele comece a acompanhar. É como dizer ao Git: “Por favor, preste atenção nestes arquivos porque eu quero salvar suas alterações”.

Por exemplo, se você fez algumas alterações em um arquivo do Power BI e quer que o Git comece a rastreá-lo para que você possa salvar essas alterações mais tarde, você usaria “git add <nome-do-arquivo>” para adicionar esse arquivo à lista de arquivos que o Git está monitorando.

O comando “git add” é o primeiro passo antes de salvar suas alterações com o comando “git commit”, que é como você **salva** as alterações no Git.

## Usando Prompt de Comando

Use o comando git add <nome-do-arquivo> para adicionar os arquivos do Power BI que você deseja rastrear.

```
MINGW64:~/c:/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ git add PowerBi_Git_GitHub.pbix
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   PowerBi_Git_GitHub.pbix

(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$
```

Aqui usamos o comando “git add <nome\_arquivo>” para adicionar o arquivo específico ao “Staging Area” a partir deste momento este arquivo passa a ser rastreado estando pronto para o “commit” e qualquer modificação será observado pelo Git o que nos possibilitará voltar a um ponto específico no tempo.

Na sequência foi usado o comando “git status”, este comando nos mostra o estado em que os arquivos se encontram no repositório. Neste caso mostra que foi incluído um novo arquivo na área de preparo, “PowerBi\_Git\_GitHub.pbix”.



```
MINGW64:/c/GitPowerBi
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   PowerBi_Git_GitHub.pbix

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   PowerBi_Git_GitHub.pbix

(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (master)
$ _
/dev/pty0 {}@16:002 U+0020
```

Após fazermos uma alteração no arquivo e salvarmos, retornamos ao prompt e repetimos o comando "git status" podemos ver que o Git já observou que há modificações no arquivo. Para incluir estas novas modificações usamos o comando "git add <nome\_arquivo>" novamente deixando-o pronto para o "commit".

Se quiser submeter todos os arquivos da pasta ao rastreamento do Git podemos usar os seguintes comandos: "git add --all" ou "git add .".



# Confirmar as alterações

O comando "git commit" é usado para salvar as alterações feitas nos arquivos que você adicionou ao Git usando "git add". É como tirar uma foto das suas alterações e salvar essa foto em um álbum, para que você possa voltar a ela mais tarde se precisar.

Quando você usa "git commit", você também precisa **adicionar uma mensagem** que descreve as alterações que você fez. Esta mensagem é importante porque ajuda a lembrar o que foi alterado em cada commit e também pode ser útil para outras pessoas que estão trabalhando no mesmo projeto.

Por exemplo, se você adicionou um novo gráfico ao seu relatório do Power BI, você usaria "git add" para adicionar o arquivo do relatório modificado ao Git e depois usaria "git commit -m 'Adicionado novo gráfico ao relatório'" para salvar essas alterações com uma mensagem descritiva.

## Usando Prompt de Comando

Use o comando 'git commit -m "Mensagem de commit"' para confirmar as alterações e adicionar uma mensagem descritiva.

```
MINGW64/c/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (master)
$ git commit -m "Primeiro commit"
[master (root-commit) 50dffa2] Primeiro commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 PowerBi_Git_GitHub.pbix
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (master)
$ git status
On branch master
nothing to commit, working tree clean
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (master)
$
```

/dev/pty0 {}@13:002 U+0020



Aqui podemos ver o primeiro commit concluído com sucesso.



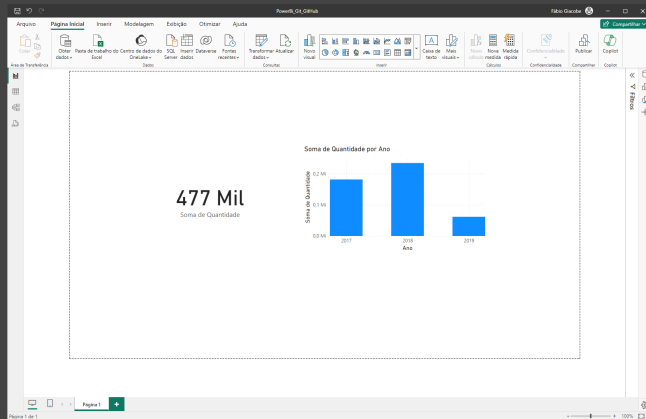
```
MINGW64:/c:/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ git add PowerBi_Git_GitHub.pbix
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   PowerBi_Git_GitHub.pbix
```

Aqui adicionamos as novas alterações ao "staging area".

```
MINGW64:/c:/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   PowerBi_Git_GitHub.pbix

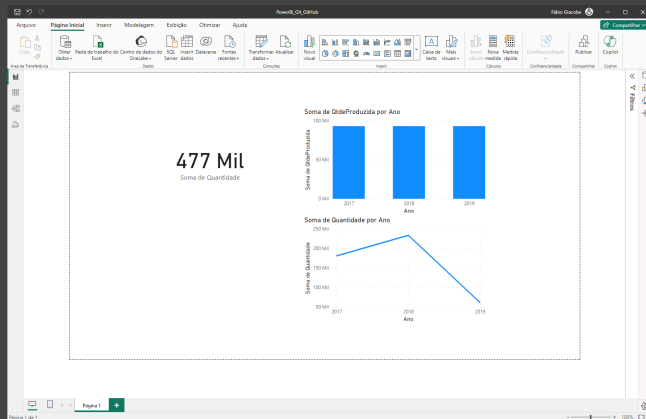
no changes added to commit (use "git add" and/or "git commit -a")
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ git add PowerBi_Git_GitHub.pbix
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ git commit -m "Inclusão de gráfico quantidade"
[master 2ab92b5] Inclusão de gráfico quantidade
 1 file changed, 0 insertions(+), 0 deletions(-)
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (master)
$ _
/dev/pty0 [!@20:002 U+0020]
```

Após o commit podemos ver que as alterações foram salvas.



Exemplo do arquivo PBIX criado e adicionado ao Git.

Agora vamos supor que adicionamos um gráfico de linhas e alteramos o gráfico de barras quantidade produzida ao invés de quantidade vendida.



Alterações feitas, arquivo salvo, update feito no Power BI Serviço.



```
MINGW64: c:/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (master)
$ git add PowerBi_Git_GitHub.pbix
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (master)
$ git commit -m "Alterado gráfico de barras e incluído gráfico de linha"
On branch master
nothing to commit, working tree clean
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (master)
$ _
/dev/pty0 {}@10:002 U+0020
```



Commit feito, alterações salvas no repositório local.

Agora, vamos supor que o cliente não aprovou as últimas alterações e precisamos voltar para a última versão do arquivo, podemos usar os seguintes comandos:

"git checkout -- <nome-arquivo>"

Este comando irá retornar para o último commit efetuado

"git reset <commit>"

Este comando permite retornarmos para qualquer commit já efetuado bastando pegar o seu id com o comando "git log" que irá listar todos os commits feitos.





```

MINGW64 ~/c:/GitPowerBi
(base)
fabio@Del15547 MINGW64 /c:/GitPowerBi (master)
$ git log
commit 5105c2932404fb80aa879bc8a5d9dc2f5ac5256 (HEAD -> master)
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 13:27:18 2024 -0300

    Gráfico de barras e gráfico de linhas

commit fa7fc9f58b7342840c25fbec1dd787fafd3aad2b
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 10:23:09 2024 -0300

    Inclusão de gráfico quantidade

commit f6e64d45469b8de0a8e303f4fecb1078089e0e9a
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 10:21:05 2024 -0300

    Inclusão de card custo unitário

commit 50dffa2d70a4515db46ea147fefd109d76a80977
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 10:13:56 2024 -0300

    Primeiro commit
(base)
fabio@Del15547 MINGW64 /c:/GitPowerBi (master)
$ -
/dev/pty0 [j]@28:002 U=0020
    
```

🔑 Aqui podemos ver os commits feitos e selecionar a qual queremos voltar, voltaremos ao commit “Inclusão de gráfico quantidade”.

```

MINGW64 ~/c:/GitPowerBi
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 13:27:18 2024 -0300

    Gráfico de barras e gráfico de linhas

commit fa7fc9f58b7342840c25fbec1dd787fafd3aad2b
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 10:23:09 2024 -0300

    Inclusão de gráfico quantidade

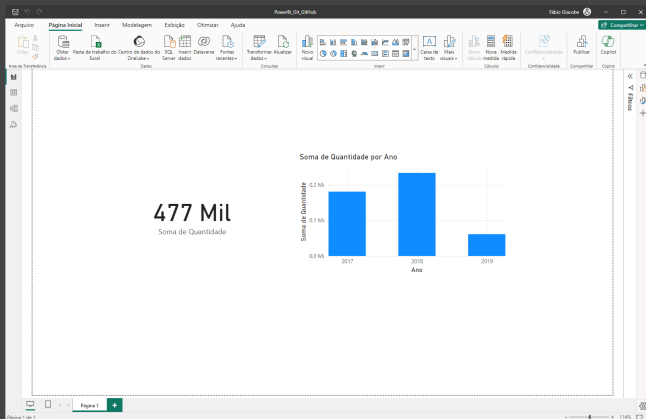
commit f6e64d45469b8de0a8e303f4fecb1078089e0e9a
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 10:21:05 2024 -0300

    Inclusão de card custo unitário

commit 50dffa2d70a4515db46ea147fefd109d76a80977
Author: FGiacobe <fabiogiaccobe@gmail.com>
Date: Thu Apr 25 10:13:56 2024 -0300

    Primeiro commit
(base)
fabio@Del15547 MINGW64 /c:/GitPowerBi (master)
$ git reset fa7fc9f58b7342840c25fbec1dd787fafd3aad2b
Unstaged changes after reset:
M   PowerBi_Git_Github.pbix
(base)
fabio@Del15547 MINGW64 /c:/GitPowerBi (master)
$ -
/dev/pty0 [j]@29:002 U=0020
    
```

🔑 Com isto voltamos a versão anterior do arquivo, ao abrí-lo novamente vemos que retornamos ao ponto anterior.



Aqui estão as alterações desfeitas.

*Obs: Este procedimento de restaurar versões deve ser feito com o arquivo fechado, fechamos o arquivo e restauramos a versão desejada e então o abrimos novamente.*



# Conectar-se a um repositório remoto

Se você não tiver um repositório remoto configurado, pode conectá-lo a um repositório remoto existente usando o comando "git remote add origin <url-do-repositorio-remoto>".

```
MINGW64/c/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (main)
$ git remote add origin git@github.com:FGiacobe/ebook_powerbi_git_github.git_

/dev/pty0 {}@02:076 U+0020
```



Desta forma configuramos o repositório remoto.



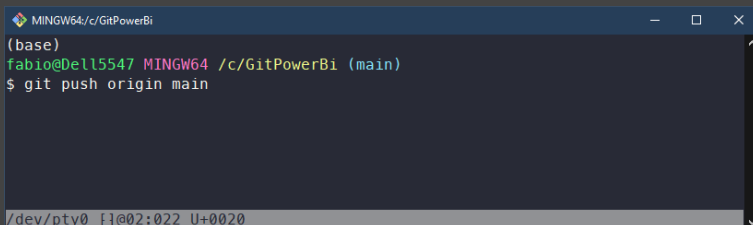
```
MINGW64/c:/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (main)
$ git remote -v
origin  git@github.com:FGiacobe/ebook_powerbi_git_github.git (fetch)
origin  git@github.com:FGiacobe/ebook_powerbi_git_github.git (push)
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (main)
$
/dev/pty0 {}@07:002 U+0020
```

Com o comando “git remote -v” listamos os repositórios remotos ligados ao projeto.



# Enviar as alterações para o repositório remoto

Use o comando “git push -u origin master” para enviar as alterações para o repositório remoto. Se for a primeira vez que você está enviando para o repositório remoto, pode ser necessário fornecer suas credenciais.



```
MINGW64/c/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c/GitPowerBi (main)
$ git push origin main

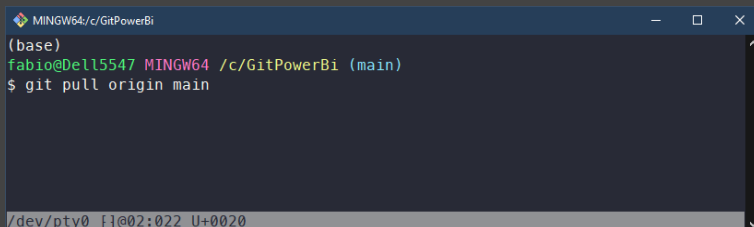
/dev/pty0 {}@02:022 U+0020
```

Com o comando “git push origin main” atualizamos o repositório remoto através do repositório local.



# Sincronização com o repositório remoto

Para sincronizar suas alterações locais com o repositório remoto, use o comando `git pull origin master`.



```
MINGW64/c:/GitPowerBi
(base)
fabio@Dell5547 MINGW64 /c:/GitPowerBi (main)
$ git pull origin main

/dev/pty0 [{}@02:022 U+0020
```

Com este comando “`git pull origin main`” atualizamos o repositório local através do repositório remoto.

Lembre-se de que este é um guia básico e existem muitos outros recursos avançados do Git que podem ser úteis, dependendo das suas necessidades específicas.

Para um guia mais avançado acesso o [Pro Git Book Oficial do Git](#).