



Reskilling4Employment Software Developer

Programação para a WEB - cliente (client-side)

Bruno Santos

bruno.santos@cesae.pt

Git

- O git é um sistema de controlo de versões.
- Possibilita o rastreio de alterações, adicionar novos ficheiros, modificar ficheiros e caso essa alteração esteja errada, voltar ao estado inicial.
- Tem a funcionalidade de inspecionar alterações (ex.: descobrir quando um erro foi gerado e como foi gerado).

Git

- A distribuição de ficheiros é uma das funcionalidades mais importantes do git.
- Quando um colaborador quer usar o projeto não vai usar o código diretamente no servidor, vai sim fazer o clone de todo o repositório e todas as alterações para a máquina local.

Git

- Se cada colaborador clonar todo o repositório para sua máquina ao usar o projeto, significa que todas as pessoas no seu projeto tem uma cópia local do trabalho, assim, caso o seu servidor/máquina tenha algum problema (ex.: avaria), todos os colaboradores possuem uma cópia do repositório e cada um deles pode para distribuir novamente para os outros colaboradores.

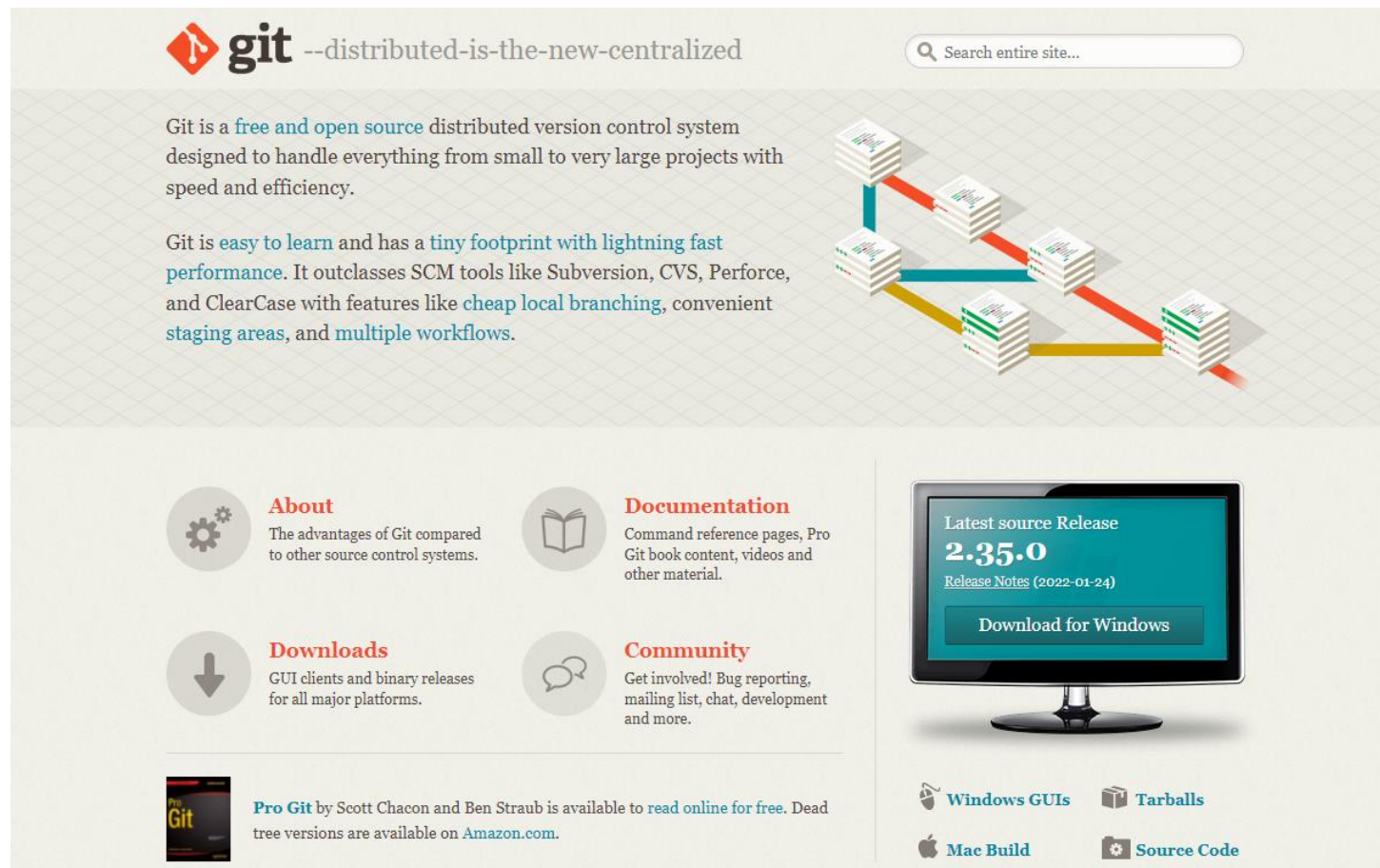
Git

- Para que um projeto seja totalmente perdido utilizando o Git só se houver apenas uma cópia do repositório, ou seja, tendo várias instâncias conseguimos ter backups recuperáveis.

Git

- O conceito de distribuição trás o conceito de branch e merge.
- Branch é uma ramificação do seu projeto num certo ponto, uma cópia exata do projeto. Ou seja, podem ser feitas todas as alterações que pretender que não irá afetar o trabalho de outros colaboradores.
- Por exemplo, um conjunto de colaboradores está responsável cada um por uma parte de um projeto, cada um desenvolve o seu trabalho sem interferir com os outros.
- No final é possível combinar essa funcionalidade (merge) na versão final do projeto e partir desse ponto distribuir para os outros colaboradores envolvidos no projeto.

Instalação do Git



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 on the right. The main text describes Git as a free and open source distributed version control system. To the right is a diagram showing a network of repositories. Below this are four sections: About, Documentation, Downloads, and Community, each with an icon and a brief description. On the right, a monitor displays the latest source release 2.35.0 with a download button. At the bottom, there are links for Windows GUIs, Tarballs, Mac Build, 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.

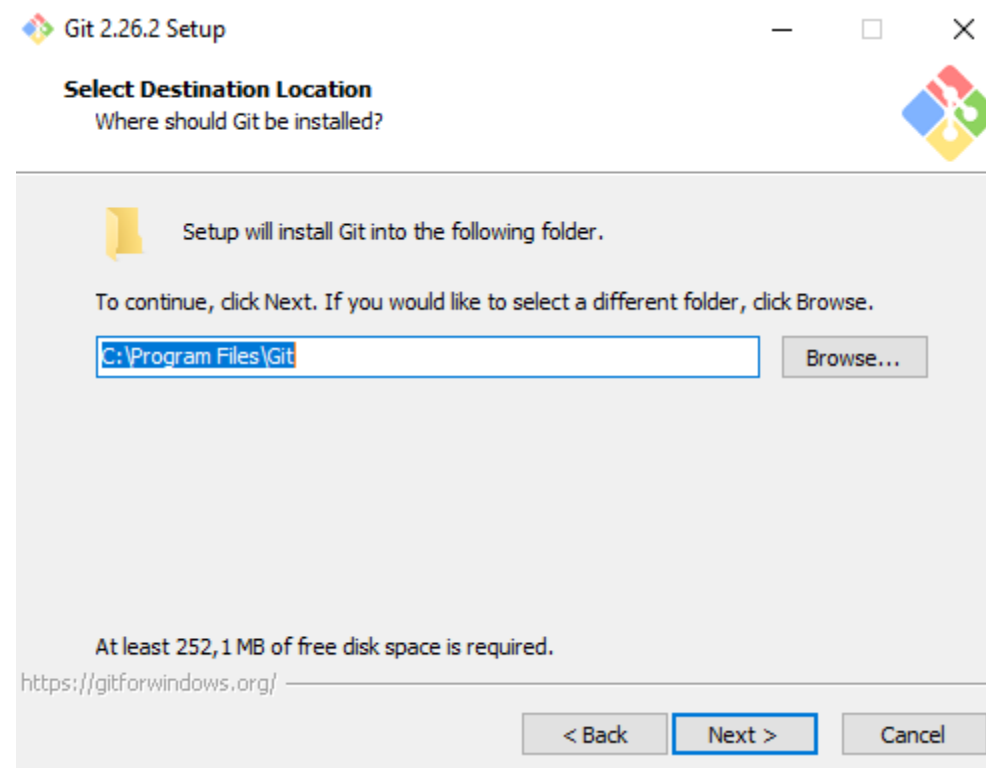
Latest source Release
2.35.0
Release Notes (2022-01-24)
Download for Windows

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

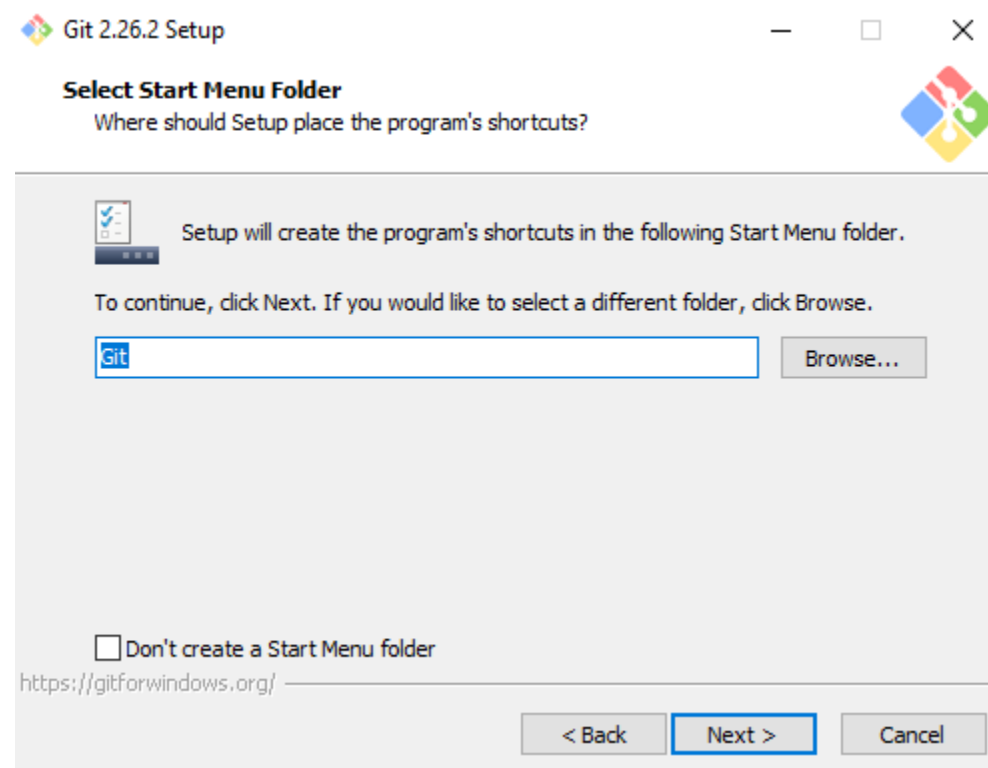
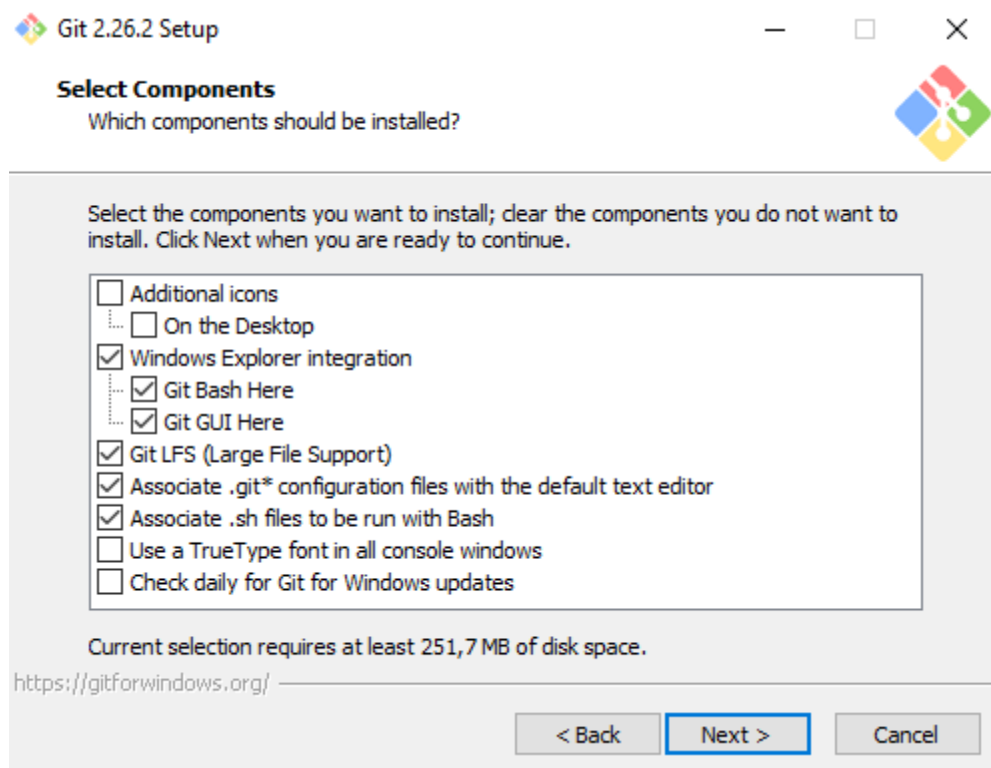
Windows GUIs Tarballs
Mac Build Source Code

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

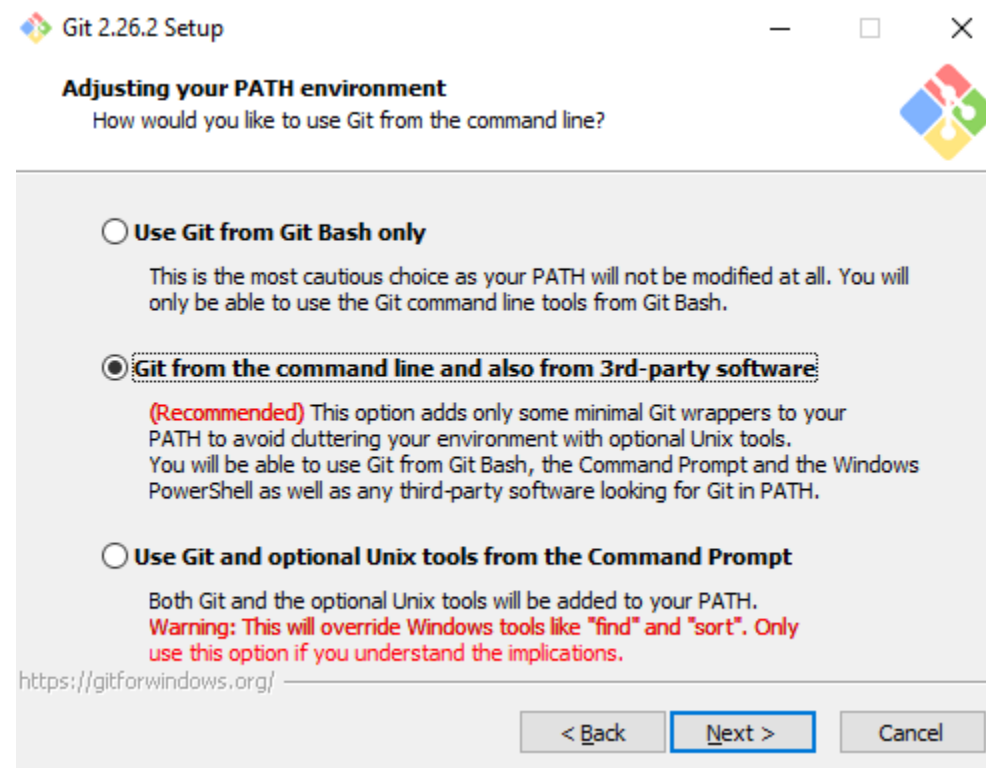
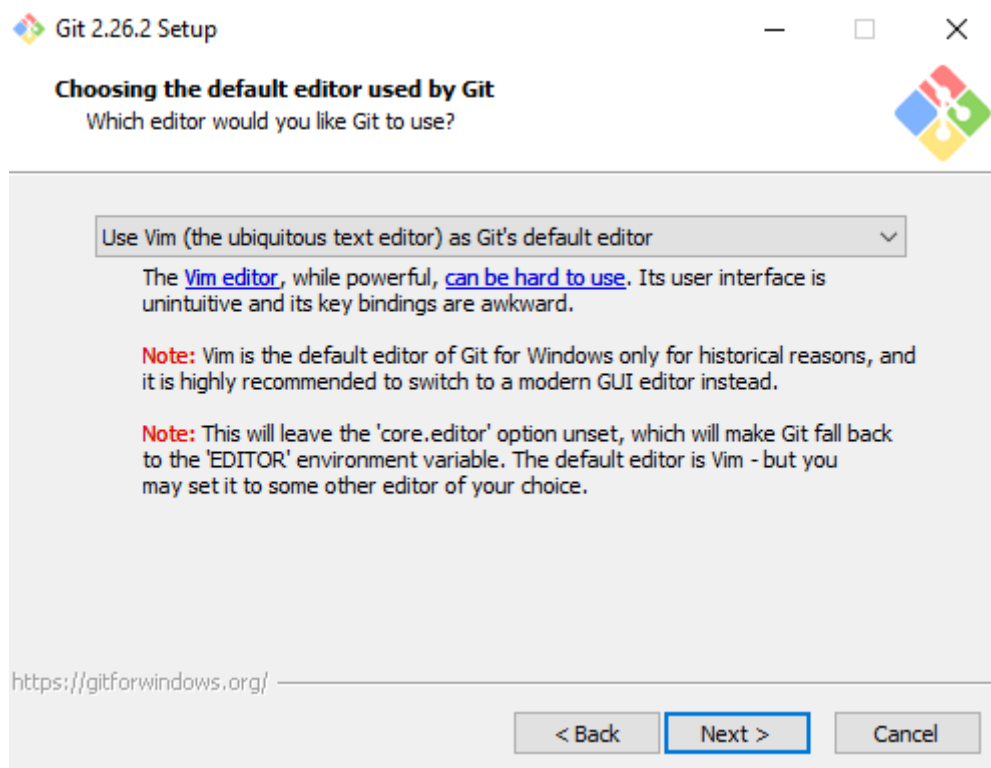
Instalação do Git



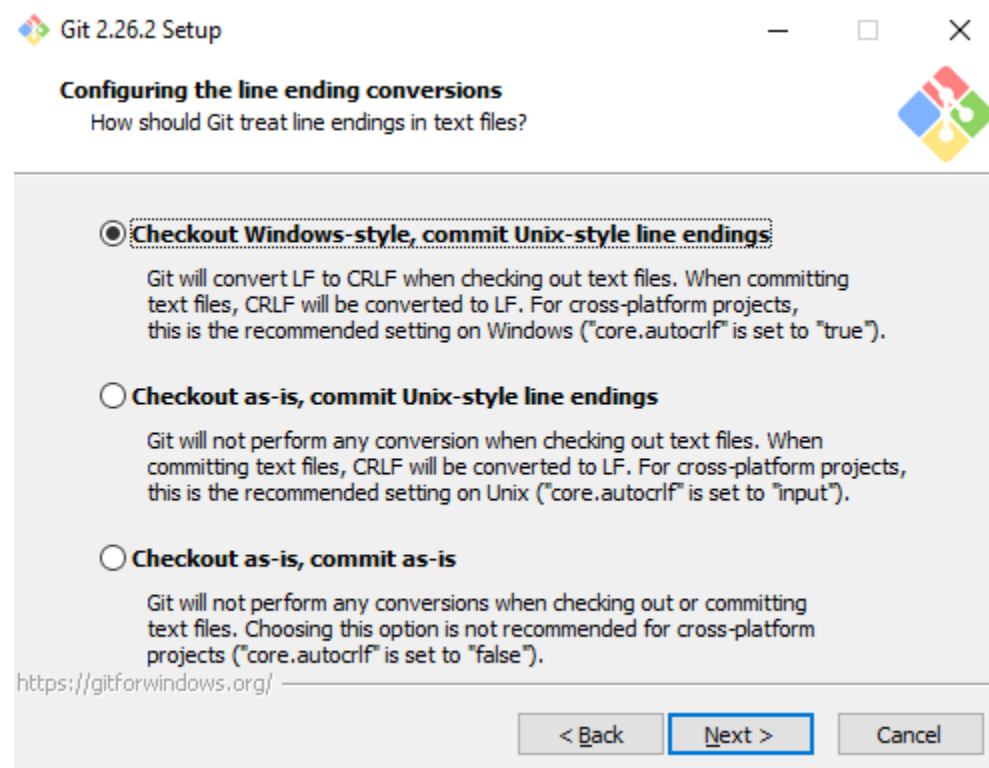
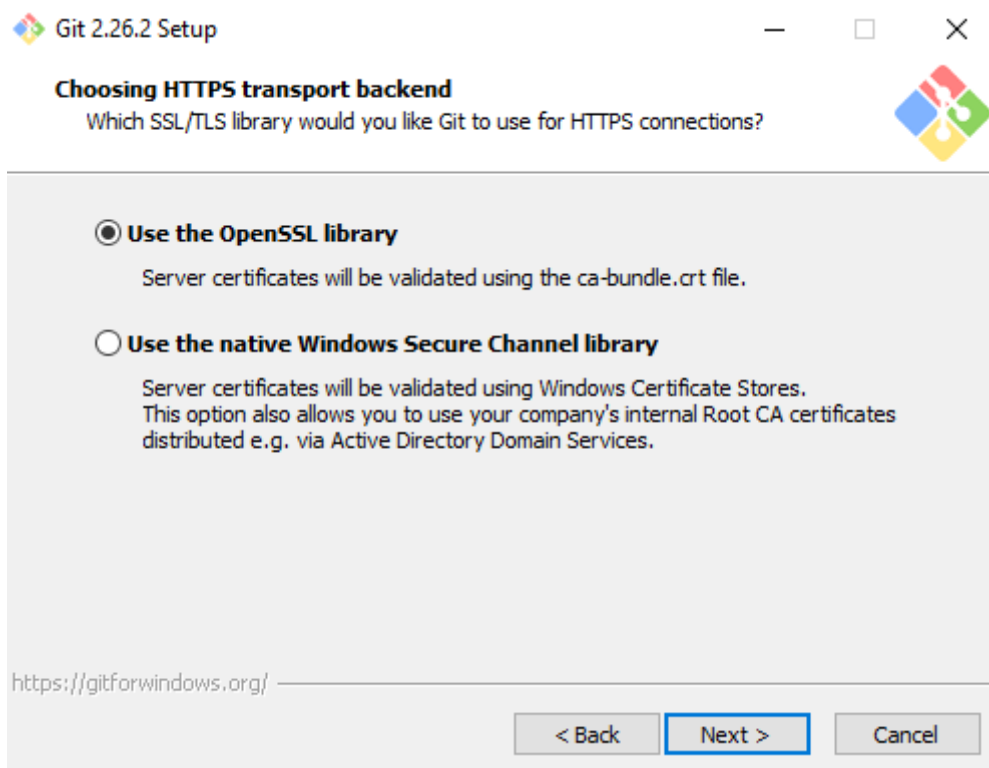
Instalação do Git



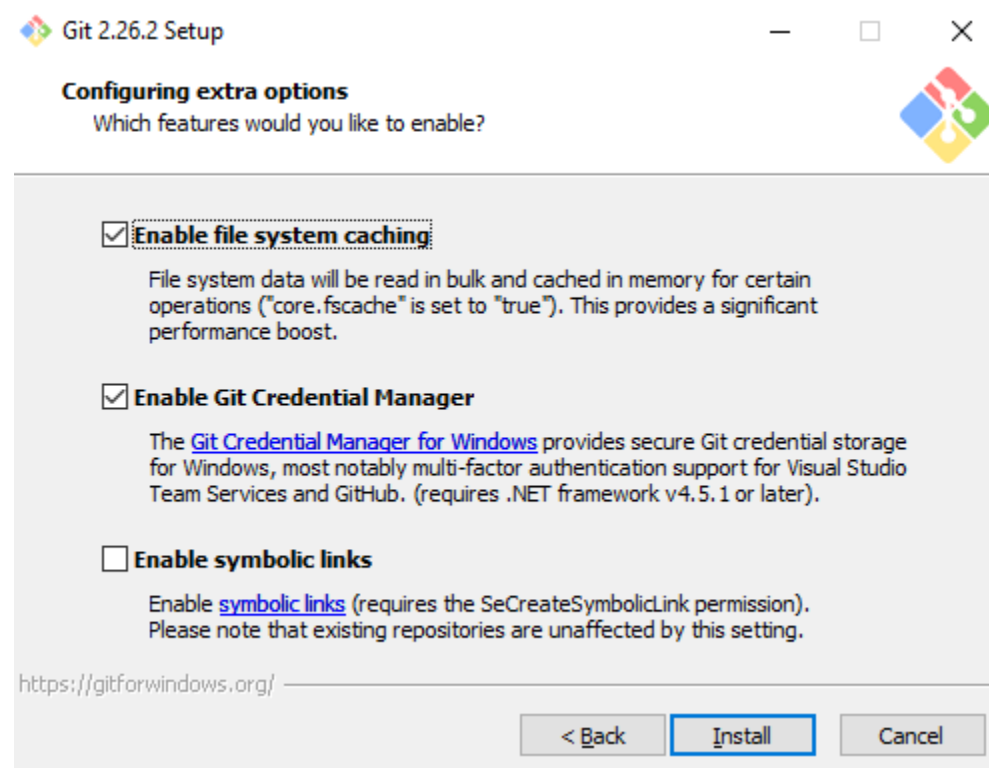
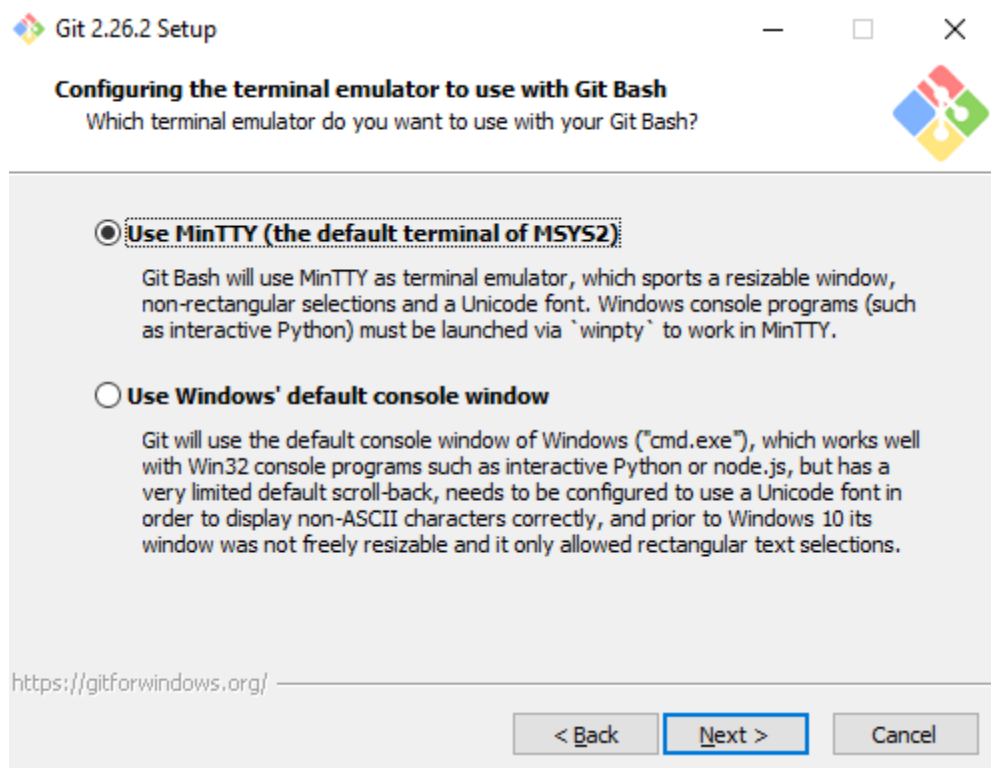
Instalação do Git



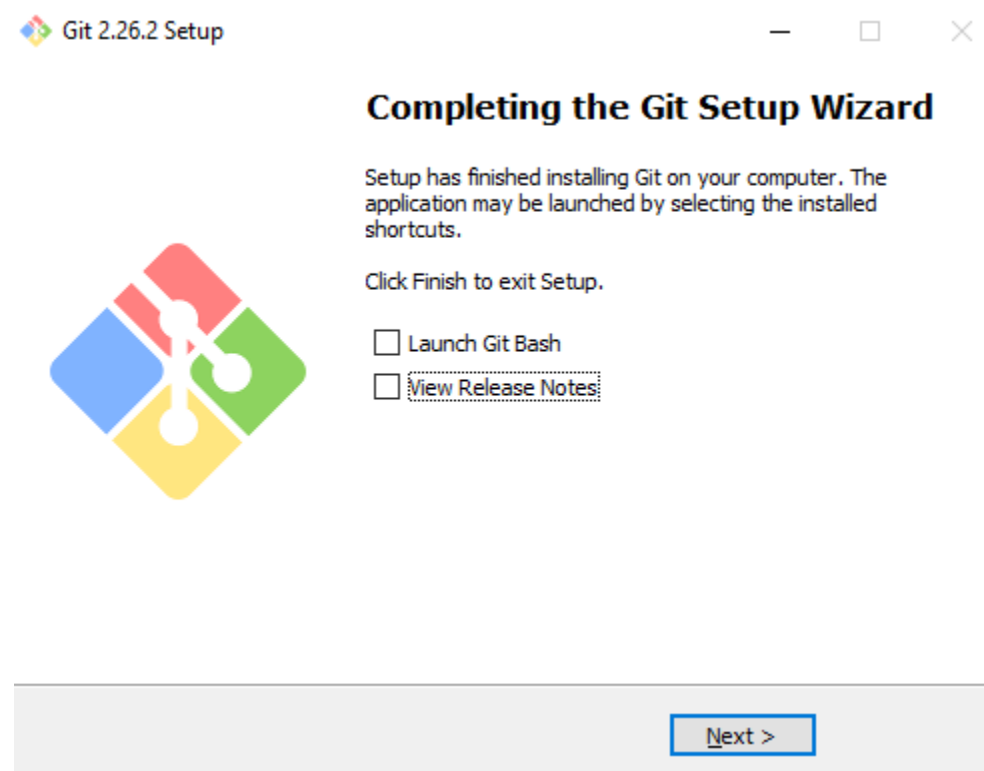
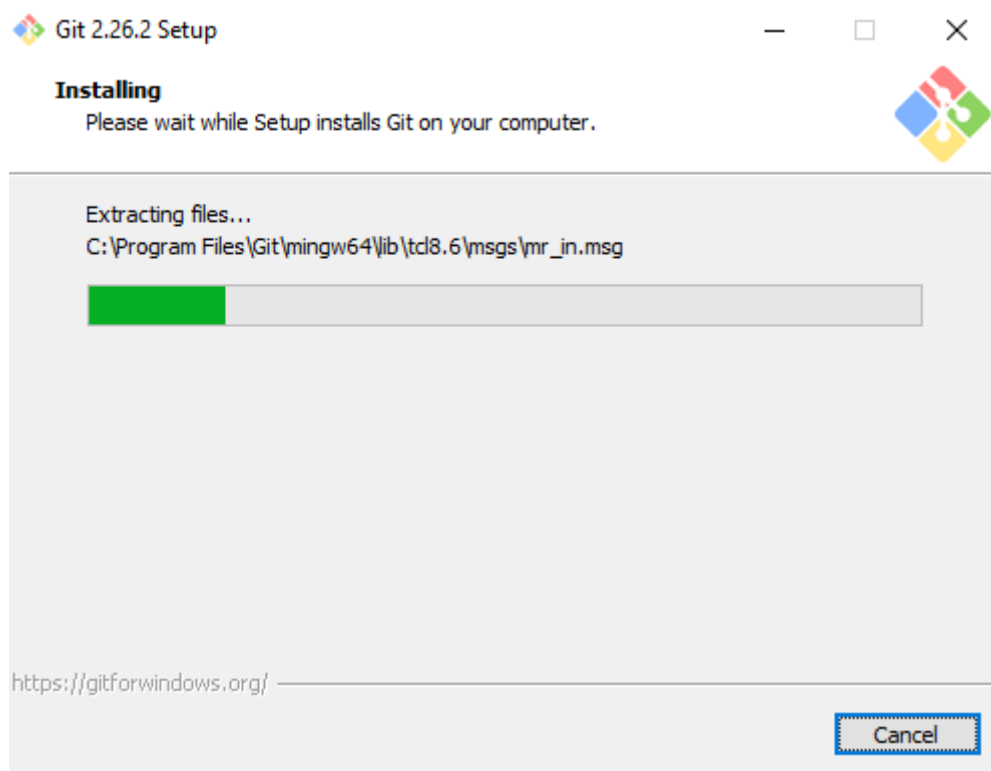
Instalação do Git



Instalação do Git



Instalação do Git

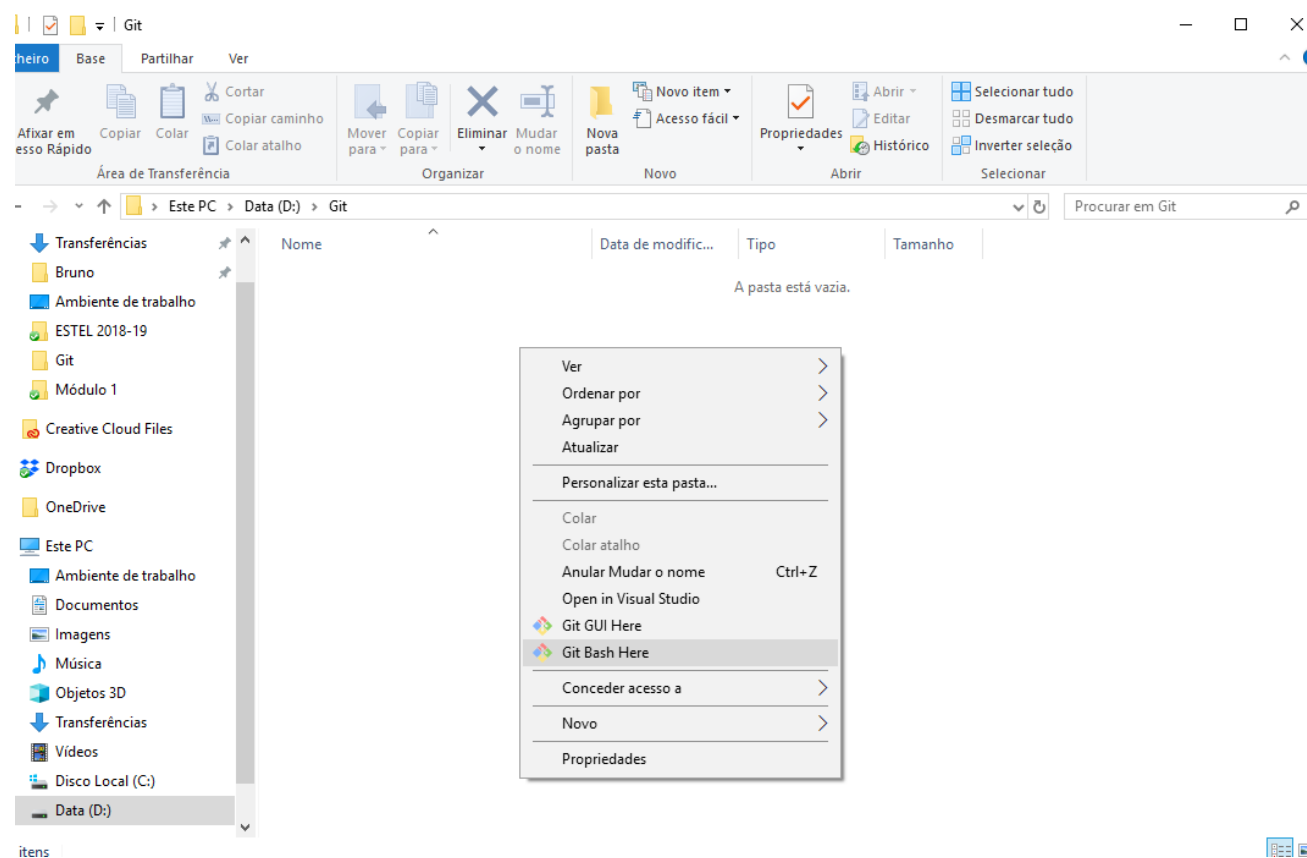


Exemplos de utilização

- Nos slides seguintes iremos utilizar a bash do git, ainda assim podemos utilizar outros terminais como por exemplo o inserido diretamente no Visual Studio Code.

Criar pasta como repositório git

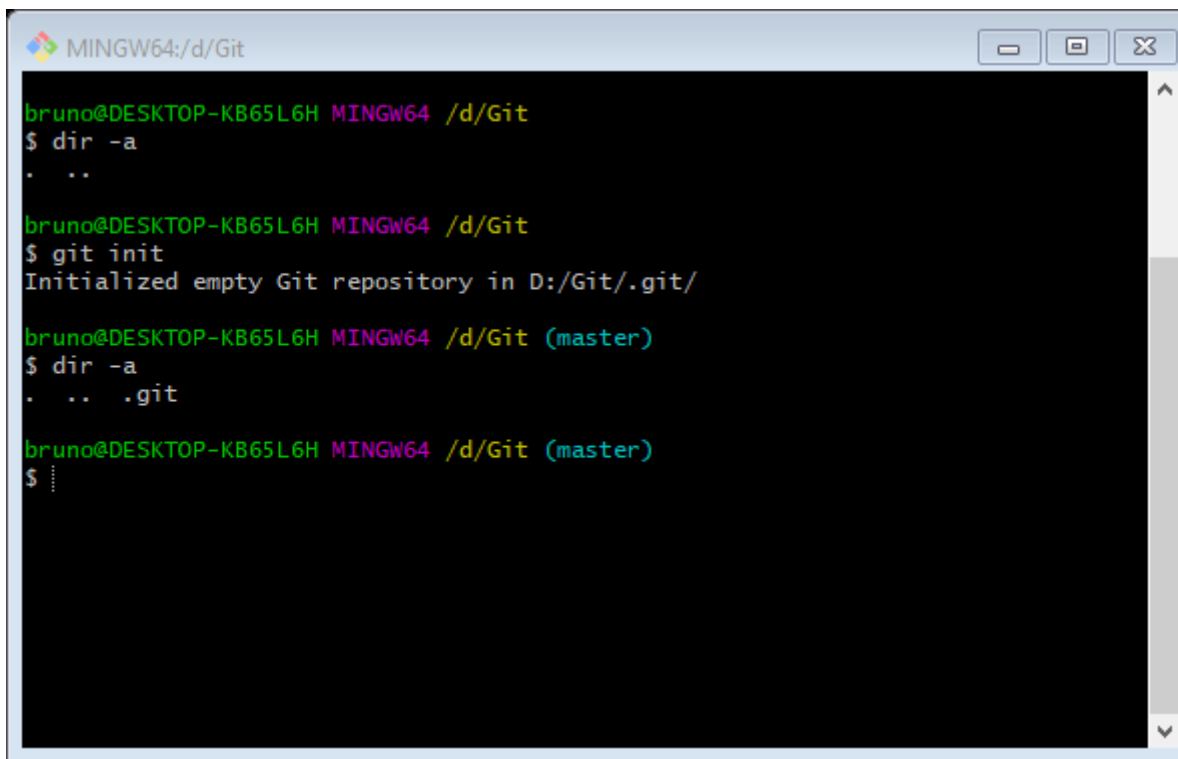
- Criar uma pasta, e com o botão direito do rato abrir Git Bash Here



Criar pasta como repositório git

- Para iniciarmos a pasta como um diretório git devemos usar o comando `git init`
- Usando o comando `dir -a` conseguimos ver os ficheiros presentes na pasta. De notar que depois do comando `git init` passa a existir um ficheiro `.git` na pasta que será o responsável pela gestão do repositório. Também é visível a anotação (master) associada à pasta.

Criar pasta como repositório git

A screenshot of a Windows terminal window titled 'MINGW64:/d/Git'. The window has a black background with green and white text. It shows the steps to initialize a new Git repository in the directory D:/Git. The user 'bruno' is at the prompt. The commands and their outputs are: 'dir -a' showing '.' and '..'; 'git init' showing 'Initialized empty Git repository in D:/Git/.git/'; and a second 'dir -a' showing '.', '..', and '.git'. The terminal is currently on the '(master)' branch.

```
bruno@DESKTOP-KB65L6H MINGW64 /d/Git
$ dir -a
.  ..

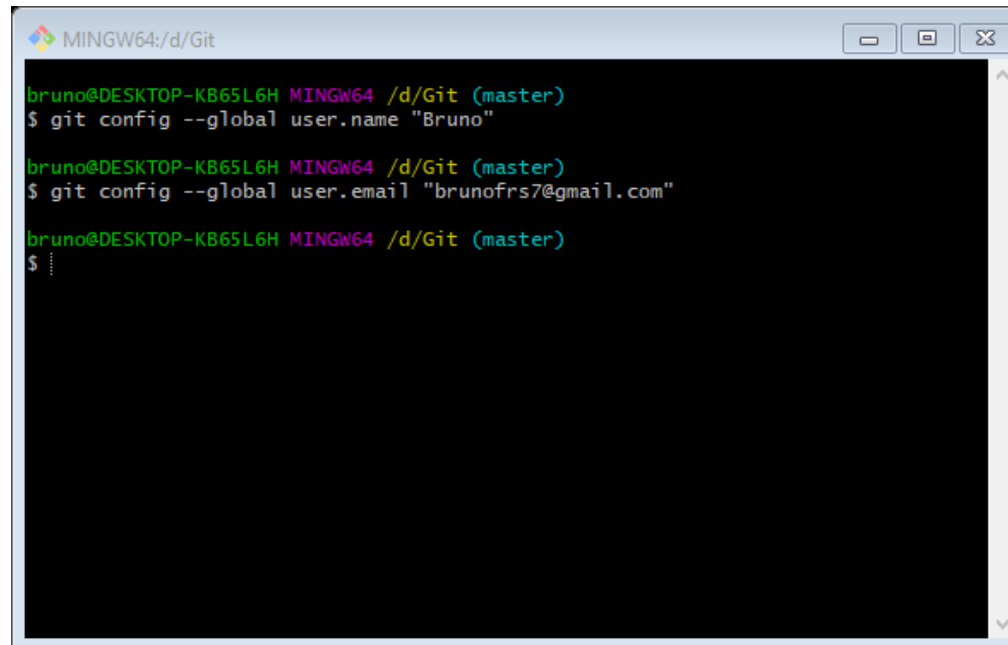
bruno@DESKTOP-KB65L6H MINGW64 /d/Git
$ git init
Initialized empty Git repository in D:/Git/.git/

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ dir -a
.  ..  .git

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Primeiros passos

- Em primeiro lugar deve ser definido o nome e email do utilizador que está a usar o repositório de forma a ser possível identificar unicamente cada colaborador.



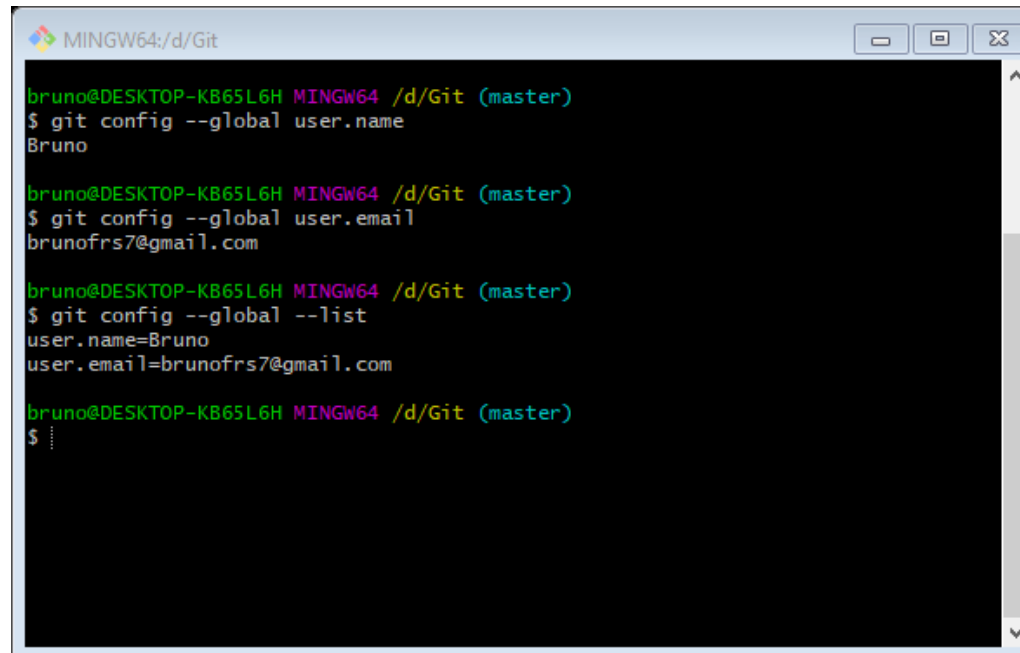
```
MINGW64:/d/Git
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git config --global user.name "Bruno"

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git config --global user.email "brunofrs7@gmail.com"

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Primeiros passos

- Para consultar os dados inseridos podemos inserir os mesmos comandos sem o valor entre aspas ou usando a flag --list



```
MINGW64:/d/Git
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git config --global user.name
Bruno

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git config --global user.email
brunofrs7@gmail.com

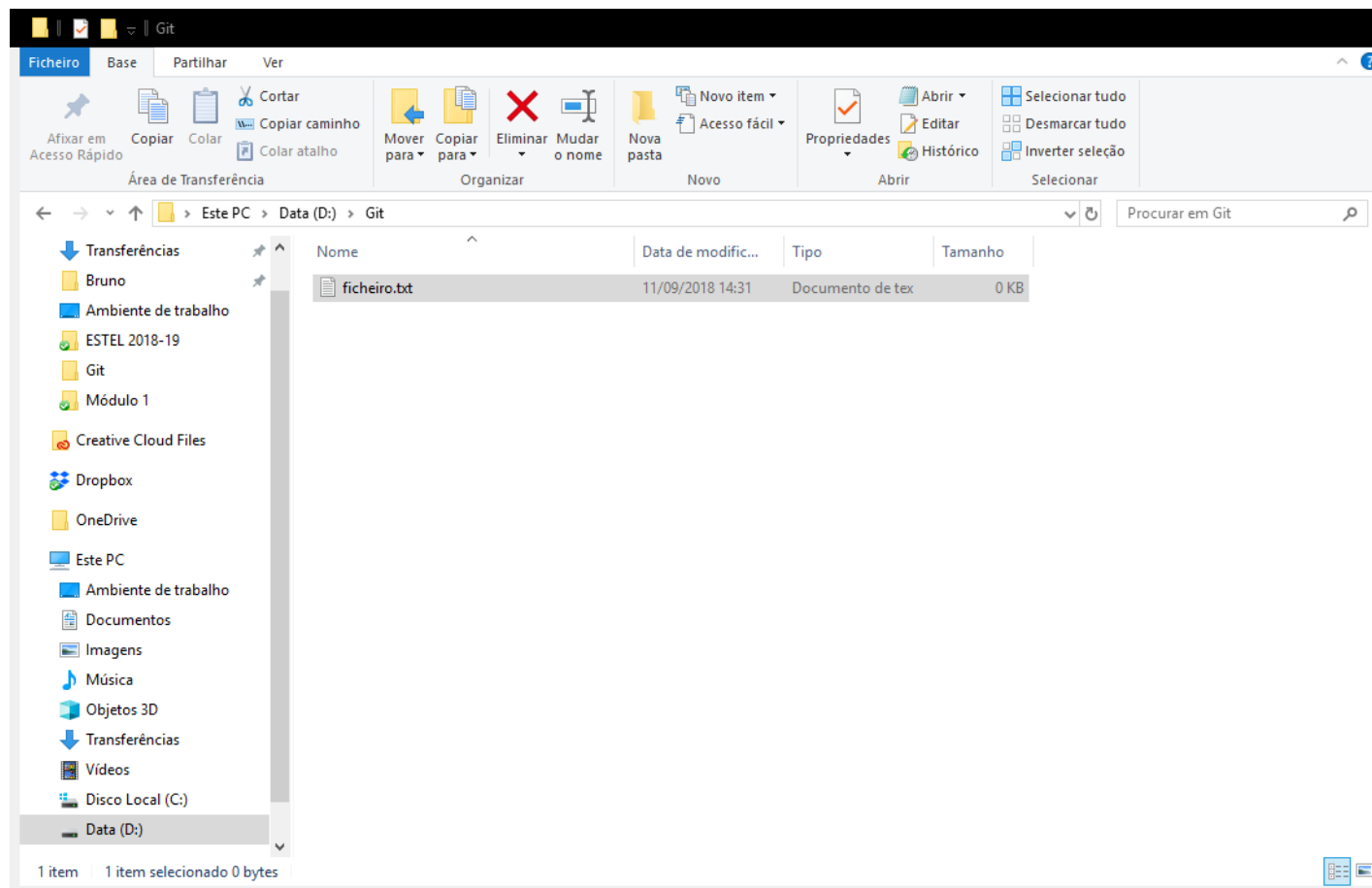
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git config --global --list
user.name=Bruno
user.email=brunofrs7@gmail.com

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

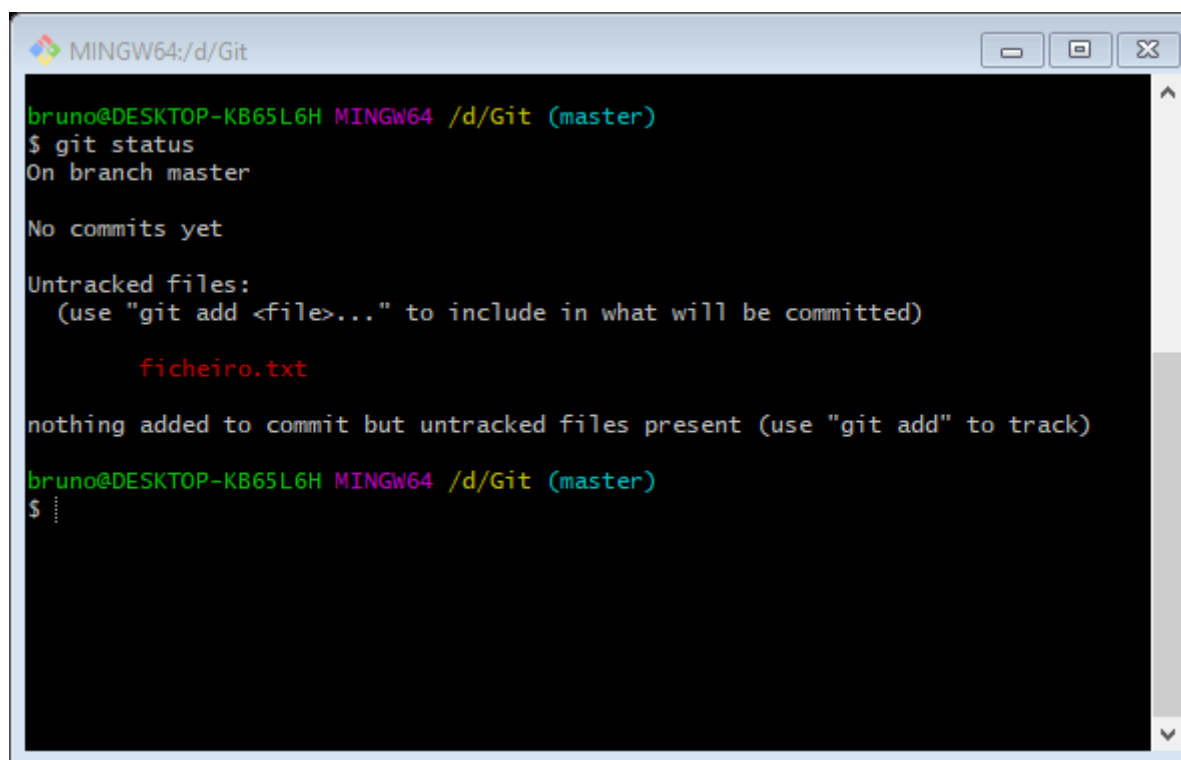
Verificar estado do repositório

- Após a configuração do nome e email do utilizador e inicialização do repositório podemos verificar quais os ficheiros que estão a ser adicionados ao controlo de versões usando o comando `git status`

Verificar estado do repositório



Verificar estado do repositório

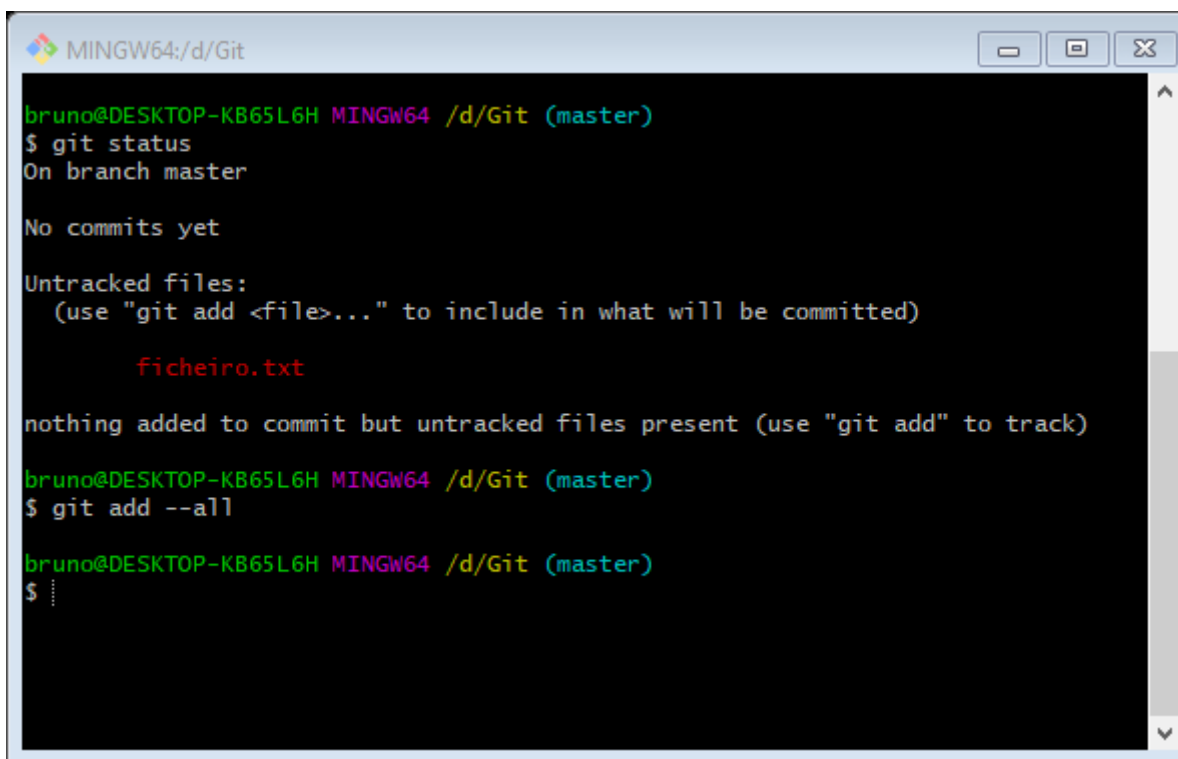
A screenshot of a Windows terminal window titled 'MINGW64:/d/Git'. The window has a black background with white and green text. The user 'bruno@DESKTOP-KB65L6H' is in the 'MINGW64 /d/Git (master)' directory. They have entered the command '\$ git status'. The output shows they are on the 'master' branch with no commits yet. There is one untracked file named 'ficheiro.txt' (displayed in red). The terminal concludes with 'nothing added to commit but untracked files present (use "git add" to track)'. The user then enters a second '\$' prompt.

```
MINGW64:/d/Git  
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)  
$ git status  
On branch master  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
      ficheiro.txt  
  
nothing added to commit but untracked files present (use "git add" to track)  
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)  
$
```

Adicionar ficheiros ao controlo de versão

- Nesta fase conseguimos saber os ficheiros que não fazem parte do controlo de versão e adicioná-los usando um dos comandos:
- `git add nomeficheiro` (adiciona apenas o ficheiro)
- `git add .` (adiciona todos os ficheiros)

Adicionar ficheiros ao controlo de versão

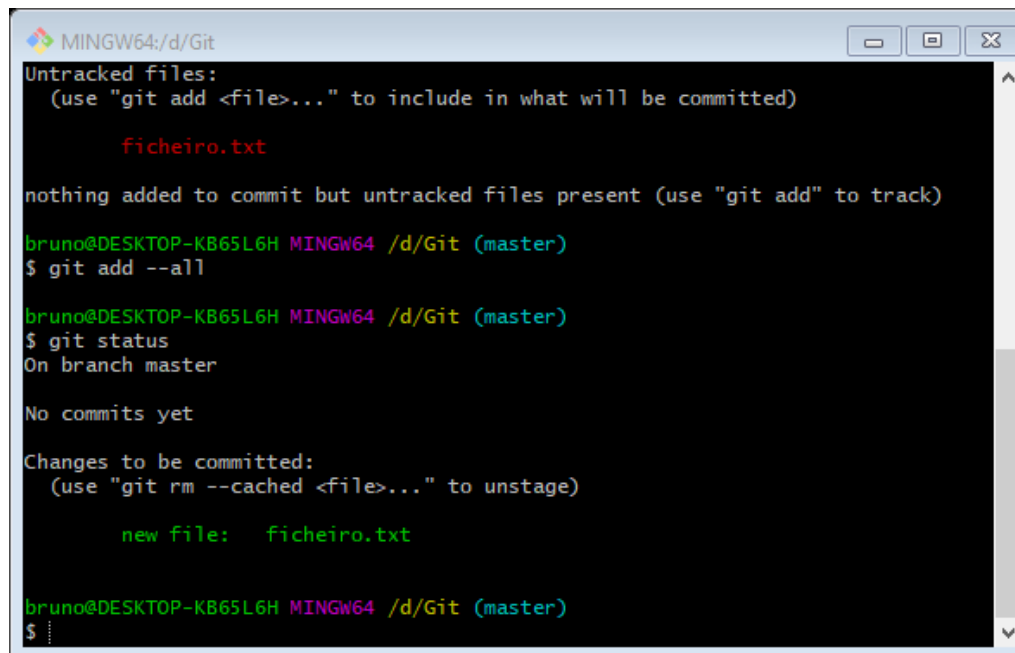


```
MINGW64:/d/Git  
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)  
$ git status  
On branch master  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
      ficheiro.txt  
  
nothing added to commit but untracked files present (use "git add" to track)  
  
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)  
$ git add --all  
  
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)  
$
```


Adicionar ficheiros ao controlo de versão

- Após adicionar o ficheiro, fazendo novamente git status aparecem os ficheiros adicionados e que não estão no commit

Adicionar ficheiros ao controlo de versão



```
MINGW64:/d/Git
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        ficheiro.txt

nothing added to commit but untracked files present (use "git add" to track)

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git add --all

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git status
On branch master

No commits yet

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

        new file:   ficheiro.txt

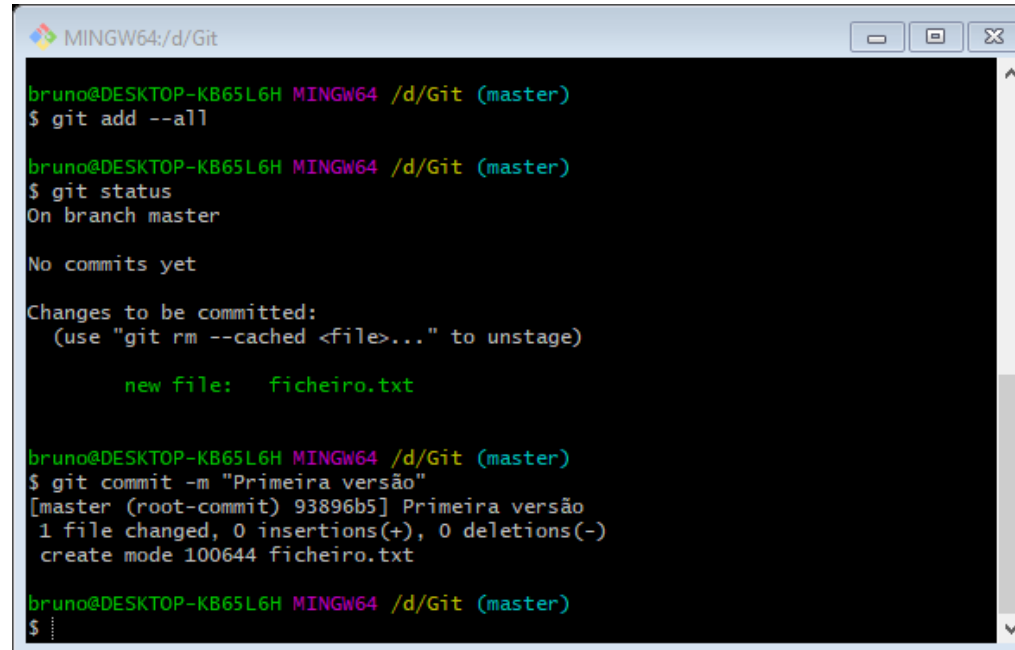
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Commit

- Depois de adicionados os ficheiros vamos fazer um commit, associando-lhe um comentário, usando o comando:

```
git commit -m "Comentário"
```

Commit



```
MINGW64:/d/Git

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git add --all

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git status
On branch master

No commits yet

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

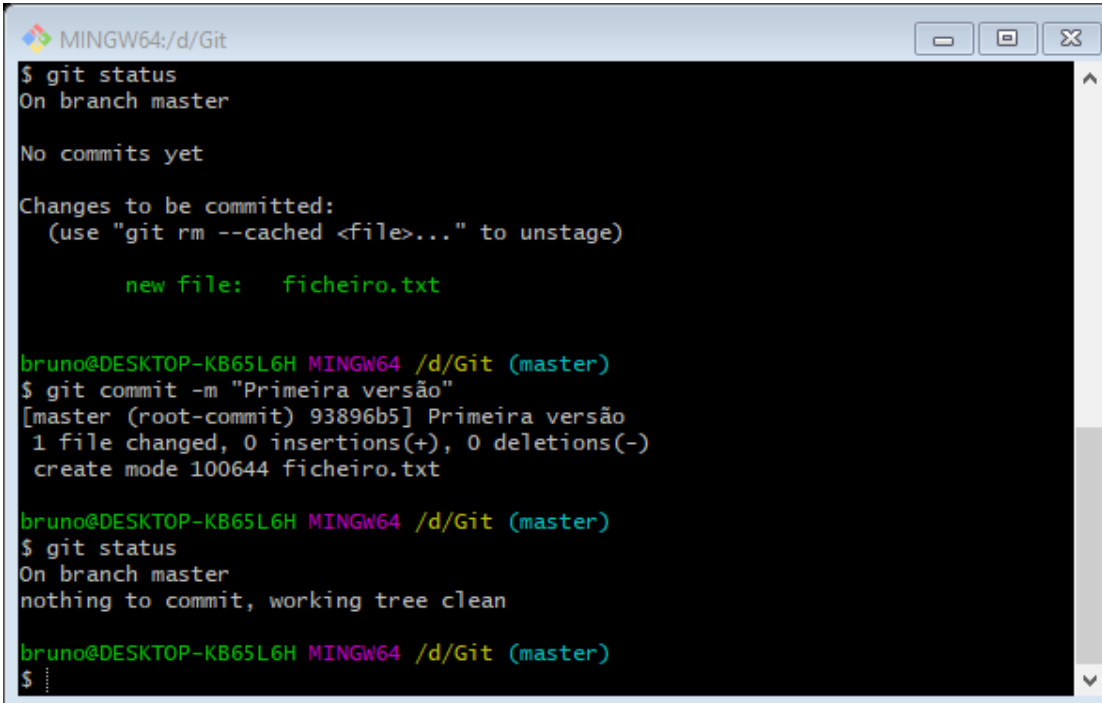
        new file:   ficheiro.txt

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git commit -m "Primeira versão"
[master (root-commit) 93896b5] Primeira versão
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ficheiro.txt

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Verificar commit

- Podemos agora fazendo `git status` verificar que não existe nenhuma alteração desde o commit.



```
MINGW64:/d/Git
$ git status
On branch master

No commits yet

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

        new file:   ficheiro.txt

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git commit -m "Primeira versão"
[master (root-commit) 93896b5] Primeira versão
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ficheiro.txt

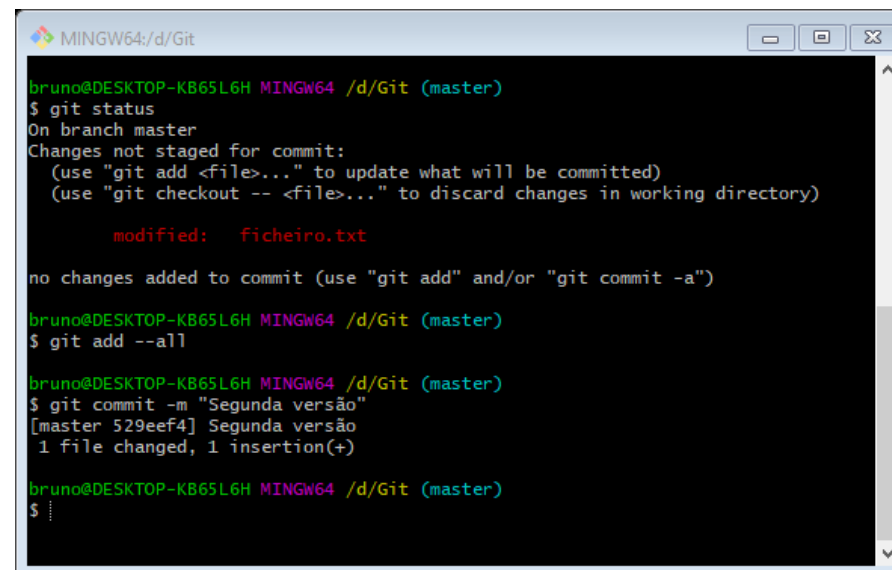
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git status
On branch master
nothing to commit, working tree clean

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Histórico de alterações

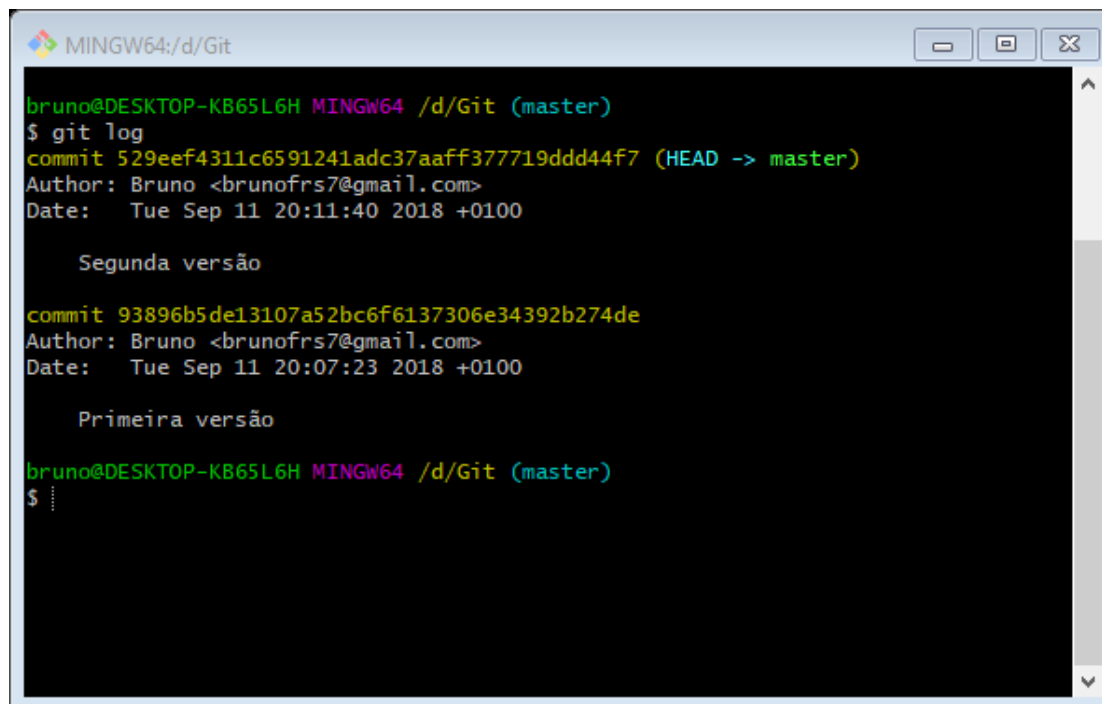
- Vamos alterar o conteúdo do ficheiro.txt, adicionar o ficheiro ao próximo controlo de versão e fazer o commit.

Histórico de alterações

A screenshot of a terminal window titled 'MINGW64: d/Git'. It shows a series of Git commands and their outputs. The user is on the 'master' branch. The first command is 'git status', which shows that 'ficheiro.txt' is modified but not staged. The second command is 'git add --all', which stages the changes. The third command is 'git commit -m "Segunda versão"', which creates a new commit with the message 'Segunda versão'. The output shows the commit hash '529eef4' and indicates that 1 file changed with 1 insertion. The prompt returns to '\$'.

Histórico de alterações

- Com o comando “git log” podemos verificar as várias versões que estão guardadas



```
MINGW64:/d/Git
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git log
commit 529eef4311c6591241adc37aaff377719ddd44f7 (HEAD -> master)
Author: Bruno <brunofrs7@gmail.com>
Date: Tue Sep 11 20:11:40 2018 +0100

    Segunda versão

commit 93896b5de13107a52bc6f6137306e34392b274de
Author: Bruno <brunofrs7@gmail.com>
Date: Tue Sep 11 20:07:23 2018 +0100

    Primeira versão

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ .....
```

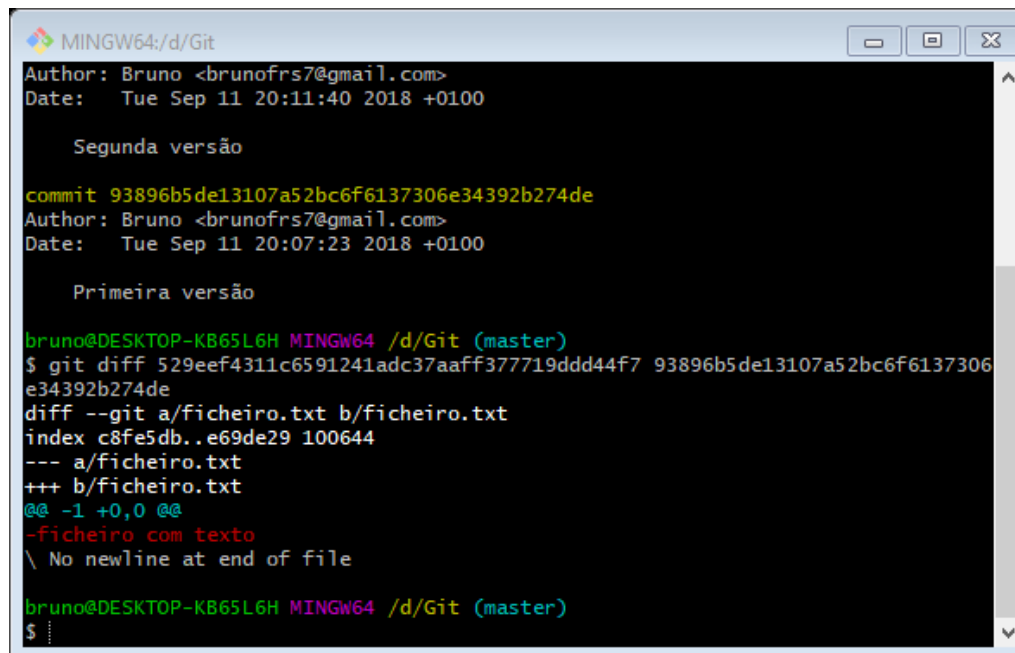

Histórico de alterações

- Para verificar as alterações que foram feitas entre dois commit podemos utilizar o comando:

`git diff HASH1 HASH2`

- Substituindo hash1 e hash2 pelos códigos dos commit apresentados

Histórico de alterações

A screenshot of a Windows terminal window titled 'MINGW64: d/Git'. The window shows the output of a 'git diff' command. It displays the commit history for a file named 'ficheiro.txt', showing the first and second versions. The second version is highlighted in yellow. The diff output shows a change from a single line to a multi-line structure, with a new line added at the end of the file. The terminal text is as follows:

```
MINGW64: d/Git
Author: Bruno <brunofrs7@gmail.com>
Date: Tue Sep 11 20:11:40 2018 +0100

Segunda versão

commit 93896b5de13107a52bc6f6137306e34392b274de
Author: Bruno <brunofrs7@gmail.com>
Date: Tue Sep 11 20:07:23 2018 +0100

Primeira versão

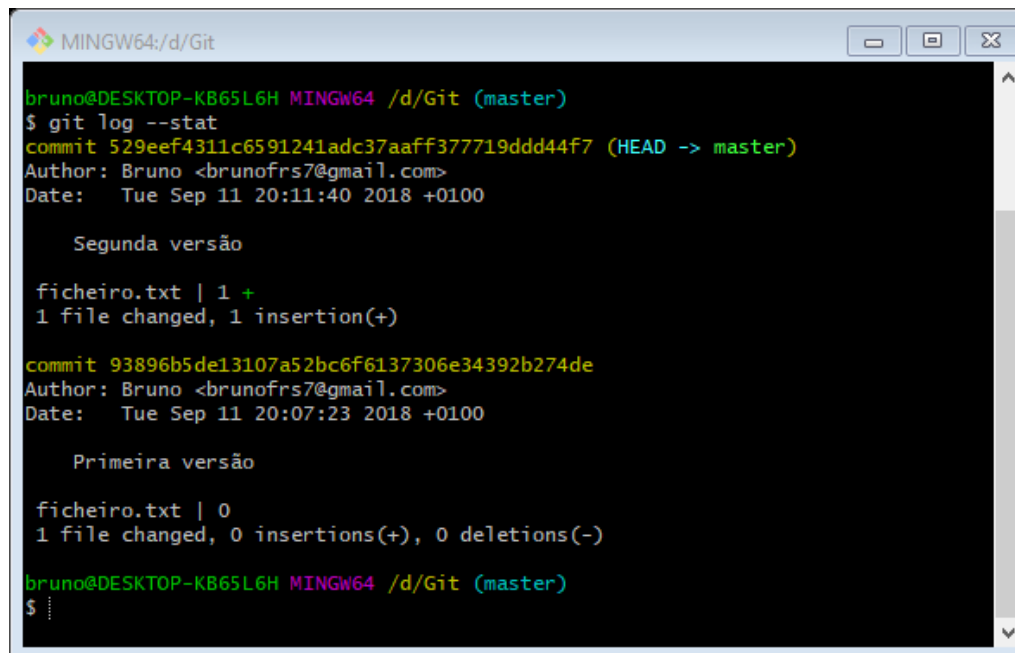
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git diff 529eef4311c6591241adc37aaff377719ddd44f7 93896b5de13107a52bc6f6137306e34392b274de
diff --git a/ficheiro.txt b/ficheiro.txt
index c8fe5db..e69de29 100644
--- a/ficheiro.txt
+++ b/ficheiro.txt
@@ -1,0 @@
-ficheiro com texto
\ No newline at end of file

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Histórico de alterações

- Utilizando o comando `git log --stat` conseguimos fazer uma análise mais aprofundada do log de versões

Histórico de alterações



```
MINGW64: d/Git

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git log --stat
commit 529eef4311c6591241adc37aaff377719ddd44f7 (HEAD -> master)
Author: Bruno <brunofrs7@gmail.com>
Date: Tue Sep 11 20:11:40 2018 +0100

    Segunda versão

    ficheiro.txt | 1 +
    1 file changed, 1 insertion(+)

commit 93896b5de13107a52bc6f6137306e34392b274de
Author: Bruno <brunofrs7@gmail.com>
Date: Tue Sep 11 20:07:23 2018 +0100

    Primeira versão

    ficheiro.txt | 0
    1 file changed, 0 insertions(+), 0 deletions(-)

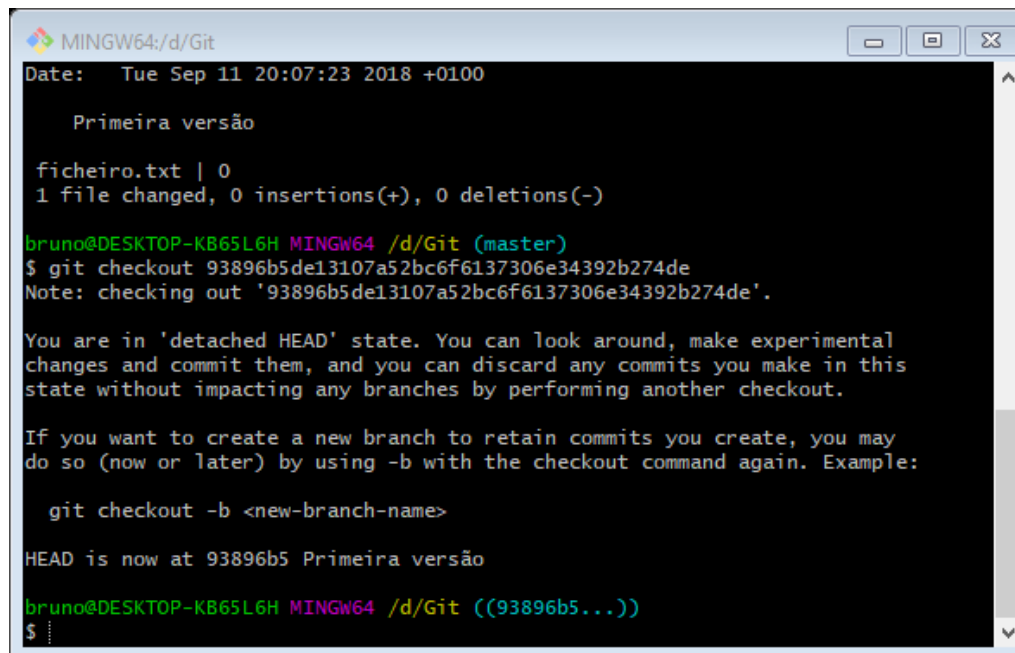
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Voltar a um commit anterior

- É possível voltar a um commit anterior descartando as alterações efetuadas desde esse momento com o comando:

`git checkout HASH`

Voltar a um commit anterior



```
MINGW64:/d/Git
Date: Tue Sep 11 20:07:23 2018 +0100

Primeira versão

ficheiro.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git checkout 93896b5de13107a52bc6f6137306e34392b274de
Note: checking out '93896b5de13107a52bc6f6137306e34392b274de'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at 93896b5 Primeira versão
bruno@DESKTOP-KB65L6H MINGW64 /d/Git ((93896b5...))
$
```

Voltar a um commit anterior

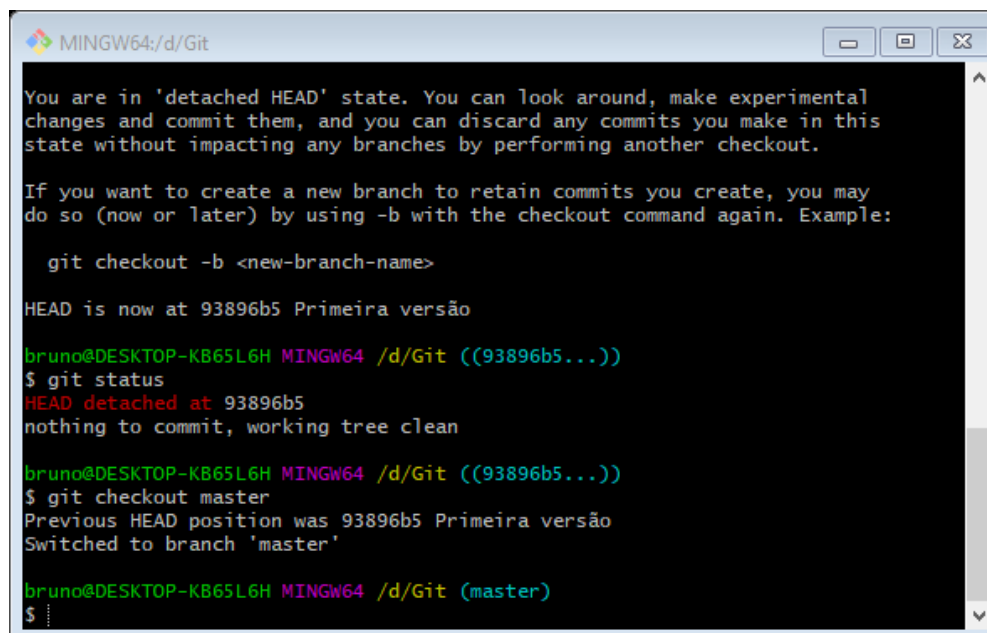
- É imediatamente perceptível que não estamos no último estado pois o valor a azul claro que indica a versão que estamos a usar deixou de ter a indicação de master para passar ao valor do commit.

```
bruno@DESKTOP-KB65L6H MINGW64 /d/Git ((93896b5...))
```

- Para além desta alteração, o ficheiro.txt voltou ao estado anterior ao último commit.
- Nota: fazendo um git log apenas aparecerão os commit até ao atual.

Voltar a um commit anterior

- Para voltar ao último commit realizado basta utilizar o mesmo comando (git checkout) indicando que a HASH é a master



```
MINGW64:/d/Git

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at 93896b5 Primeira versão
bruno@DESKTOP-KB65L6H MINGW64 /d/Git ((93896b5...))
$ git status
HEAD detached at 93896b5
nothing to commit, working tree clean

bruno@DESKTOP-KB65L6H MINGW64 /d/Git ((93896b5...))
$ git checkout master
Previous HEAD position was 93896b5 Primeira versão
Switched to branch 'master'

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```


Desfazer alterações antes de commit

- Colocando o cenário de que um ficheiro que foi alterado não o devia ter sido ou que a alteração efetuada está errada, é possível, apenas naquele ficheiro, ou conjunto de ficheiros, voltar ao estado anterior com o comando:

`git checkout nomeficheiro`

- Caso sejam muitos ficheiros e quisermos todos podemos fazer:

`git reset --hard`

Desfazer alterações antes de commit

- Considerando agora que a alteração realizada foi a criação de um ou mais ficheiros e que a mesma não devia ter sido realizada.
- Como os ficheiros não foram adicionados ao commit o comando anterior não funciona, para isso podemos fazer o comando `git clean -n` para serem apresentados quais os ficheiros nessas condições e `git clean -f` para os eliminar.
- Utilizando `git clean -i` podemos especificar quais os ficheiros a eliminar.
- Nota: foram criados novos ficheiros na pasta de trabalho para realizar este exemplo.

Desfazer alterações antes de commit

```
MINGW64:/d/Git
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ dir -a
.      ficheiro\ -\ Cópia\ (2).txt  ficheiro\ -\ Cópia\ (5).txt  ficheiro.txt
..     ficheiro\ -\ Cópia\ (3).txt  ficheiro\ -\ Cópia\ (6).txt
.git   ficheiro\ -\ Cópia\ (4).txt  ficheiro\ -\ Cópia.txt

bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git clean -n
Would remove "ficheiro - C:\303\263pia (2).txt"
Would remove "ficheiro - C:\303\263pia (3).txt"
Would remove "ficheiro - C:\303\263pia (4).txt"
Would remove "ficheiro - C:\303\263pia (5).txt"
Would remove "ficheiro - C:\303\263pia (6).txt"
Would remove "ficheiro - C:\303\263pia.txt"




bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$ git clean -f
Removing "ficheiro - C:\303\263pia (2).txt"
Removing "ficheiro - C:\303\263pia (3).txt"
Removing "ficheiro - C:\303\263pia (4).txt"
Removing "ficheiro - C:\303\263pia (5).txt"
Removing "ficheiro - C:\303\263pia (6).txt"
Removing "ficheiro - C:\303\263pia.txt"

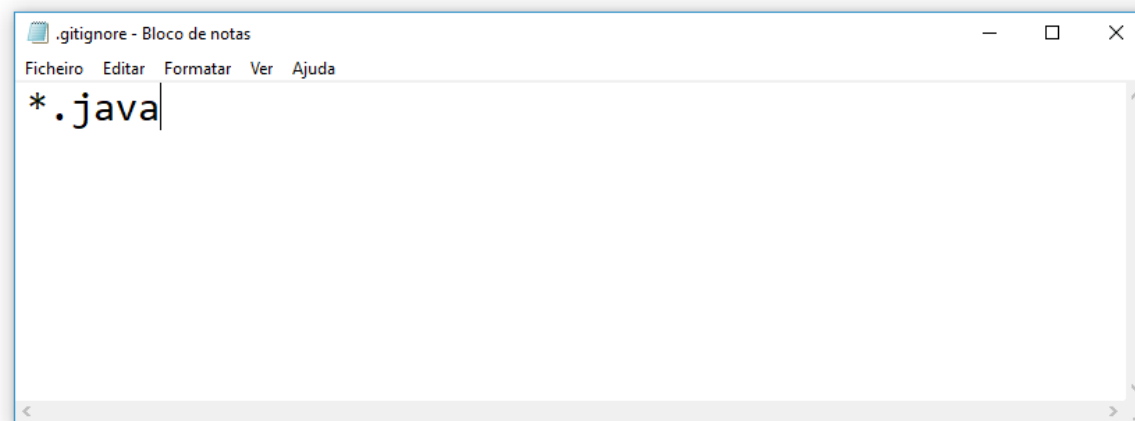
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)
$
```

Ignorar ficheiros num commit

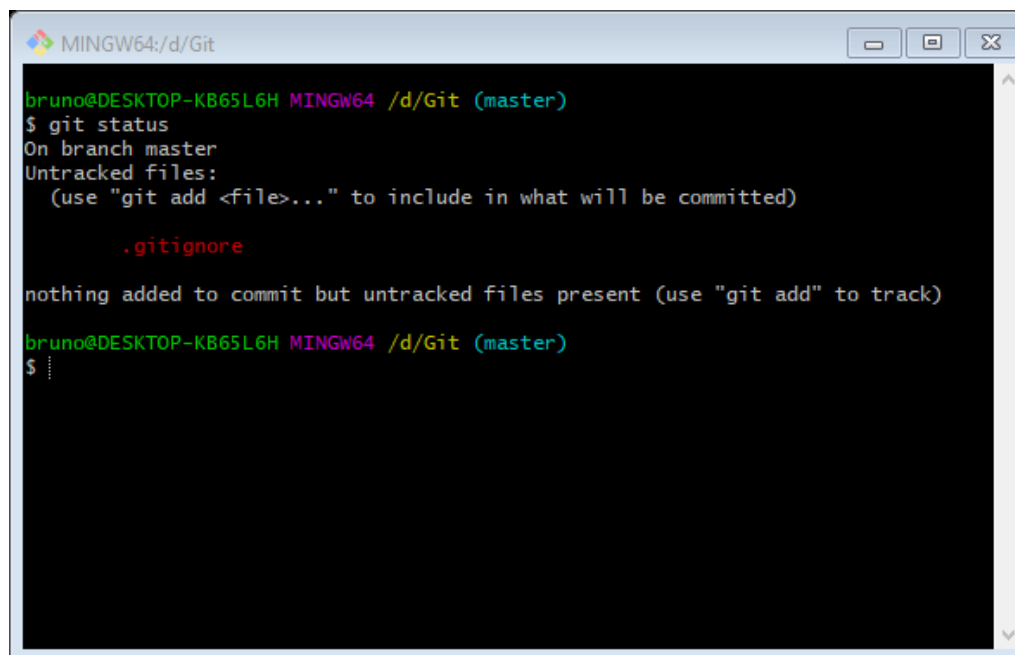
- Para ignorar ficheiros basta acrescentar à pasta onde está configurado o repositório um ficheiro sem nome e com extensão .gitignore
- Dentro deste ficheiro deve ser colocados os ficheiros ou tipo de ficheiros a excluir.
- NOTA: em Windows não é possível criar um ficheiro sem nome, assim o ficheiro deve ter como nome .gitignore.

Ignorar ficheiros num commit

 .gitignore	17/09/2018 22:21	Documento de tex	1 KB
 ficheiro.java	11/09/2018 20:25	Ficheiro JAVA	1 KB
 ficheiro.txt	11/09/2018 20:25	Documento de tex	1 KB



Ignorar ficheiros num commit

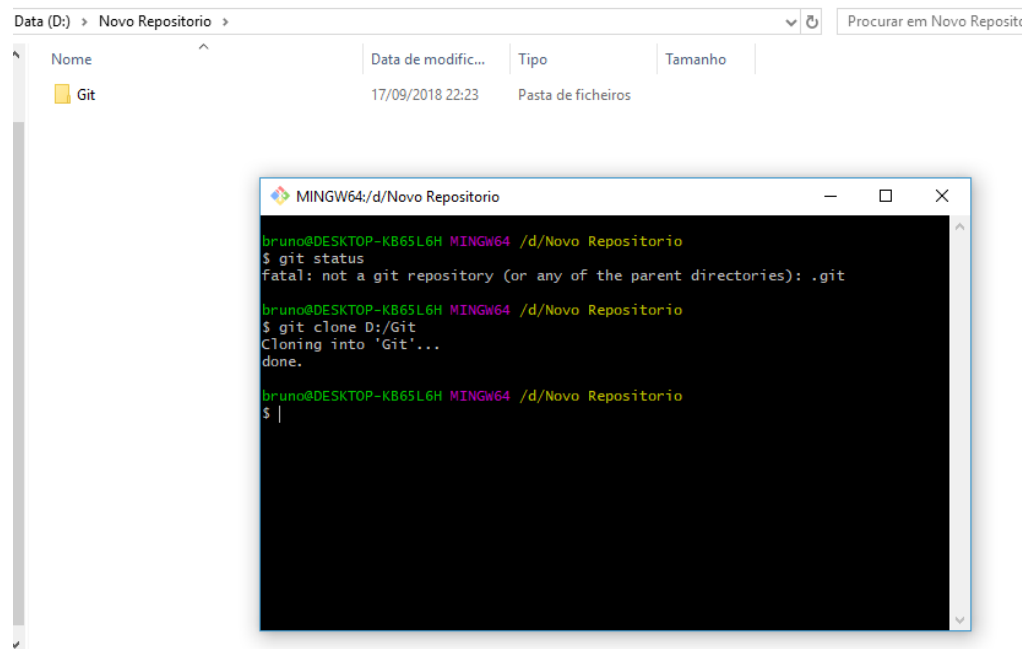


```
MINGW64:/d/Git  
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)  
$ git status  
On branch master  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
    .gitignore  
  
nothing added to commit but untracked files present (use "git add" to track)  
bruno@DESKTOP-KB65L6H MINGW64 /d/Git (master)  
$
```

Clonar Repositório

- Para clonar um repositório para uma nova pasta basta usar o comando `git clone <origem>` e especificar a origem de um repositório.
- Nota: para além da última versão são clonados todos os commit anteriores.

Clonar Repositório



Data (D:) > Novo Repositorio > Procurar em Novo Reposit

Nome	Data de modific...	Tipo	Tamanho
Git	17/09/2018 22:23	Pasta de ficheiros	

```
bruno@DESKTOP-KB65L6H MINGW64 /d/Novo Repositorio
$ git status
fatal: not a git repository (or any of the parent directories): .git

bruno@DESKTOP-KB65L6H MINGW64 /d/Novo Repositorio
$ git clone D:/Git
Cloning into 'Git'...
done.

bruno@DESKTOP-KB65L6H MINGW64 /d/Novo Repositorio
$ |
```


GitHub

- O GitHub é uma plataforma de armazenamento de projetos com controlo de versão em Git.
- Vamos começar por criar um repositório no GitHub


GitHub

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

 brunofrs7 /

Great repository names are short and memorable. Need inspiration? How about [literate-octo-giggle](#).

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

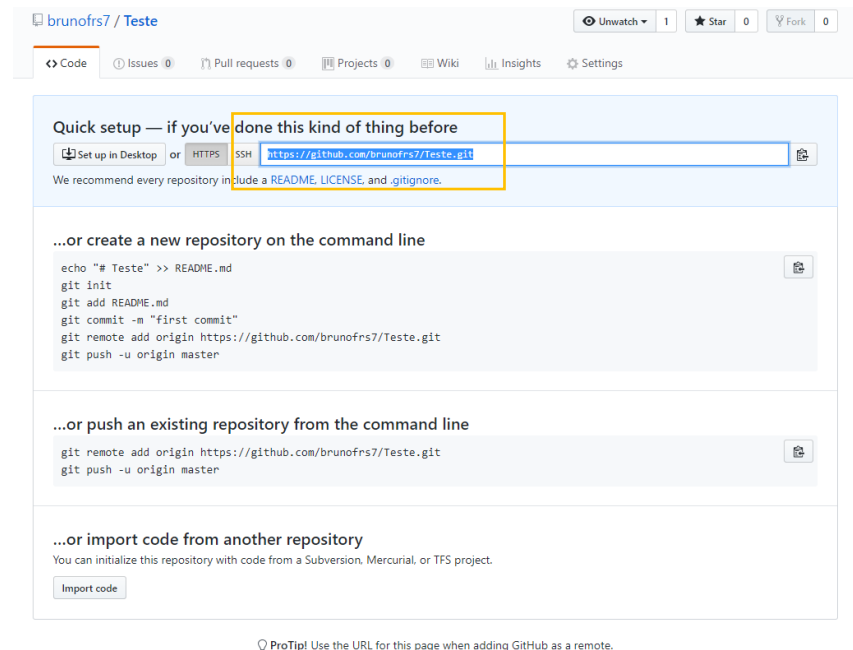
Add .gitignore: **None**

Add a license: **None**



Create repository

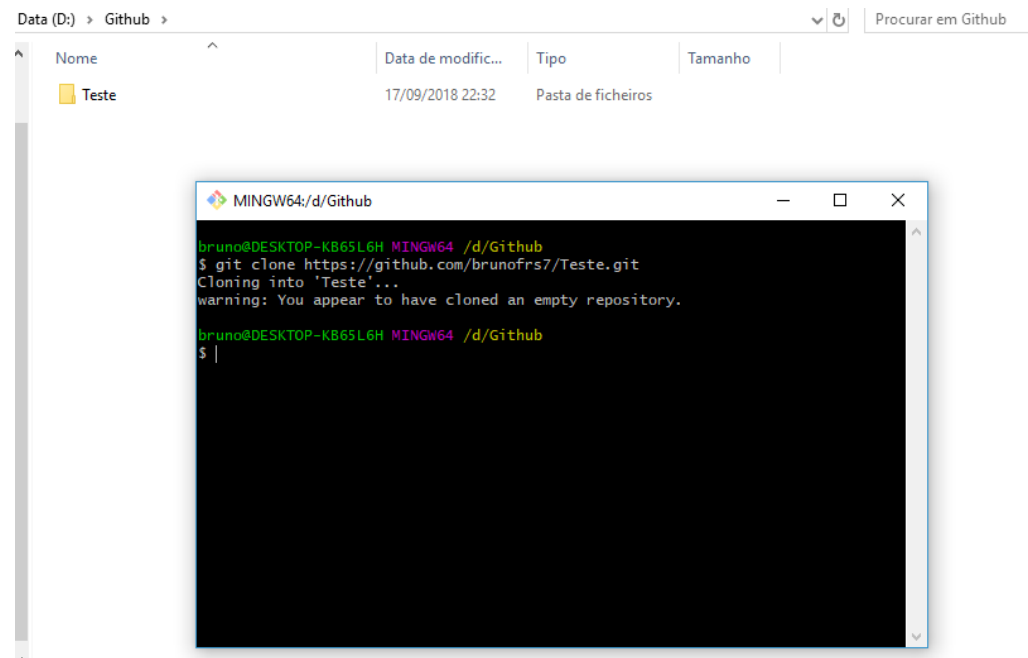
GitHub



GitHub

- Utilizando o link fornecido como origem do comando `git clone` conseguimos descarregar o repositório para a máquina local.

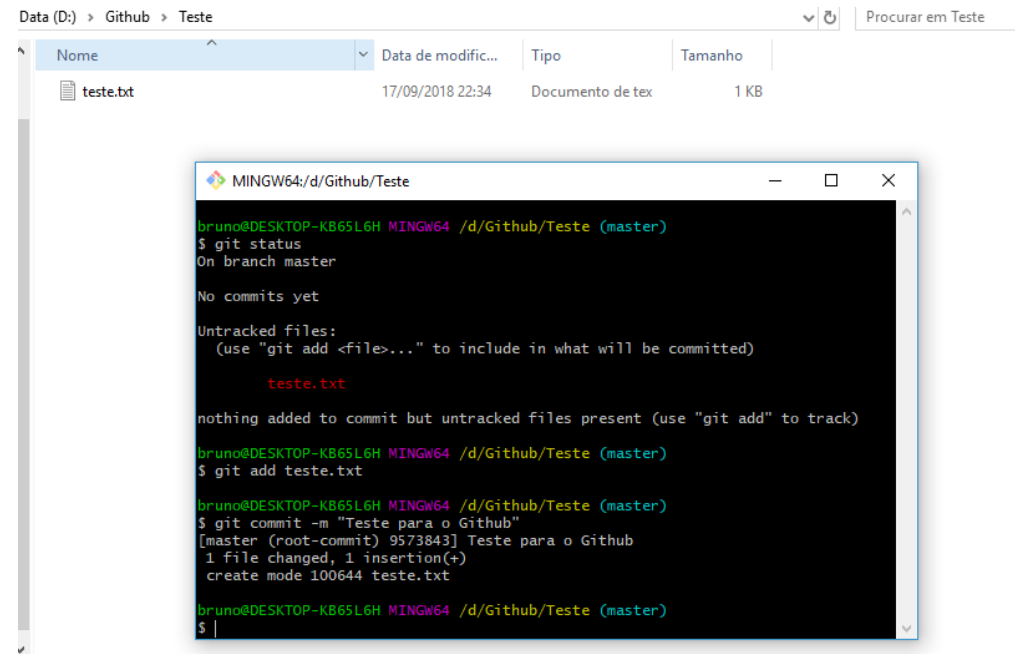
GitHub



GitHub

- Após realizar algumas alterações queremos submeter o novo commit para o GitHub, para isso, após a criação do commit na máquina local deve ser utilizado o comando `git push` para enviar as alterações para o servidor.

GitHub



The image shows a Windows File Explorer window and a terminal window. The File Explorer window is titled 'Data (D:) > Github > Teste' and shows a file named 'teste.txt' with a size of 1 KB, modified on 17/09/2018 at 22:34. The terminal window is titled 'MINGW64:/d/Github/Teste' and shows the following commands and output:

```
bruno@DESKTOP-KB65L6H MINGW64 /d/Github/Teste (master)
$ git status
On branch master

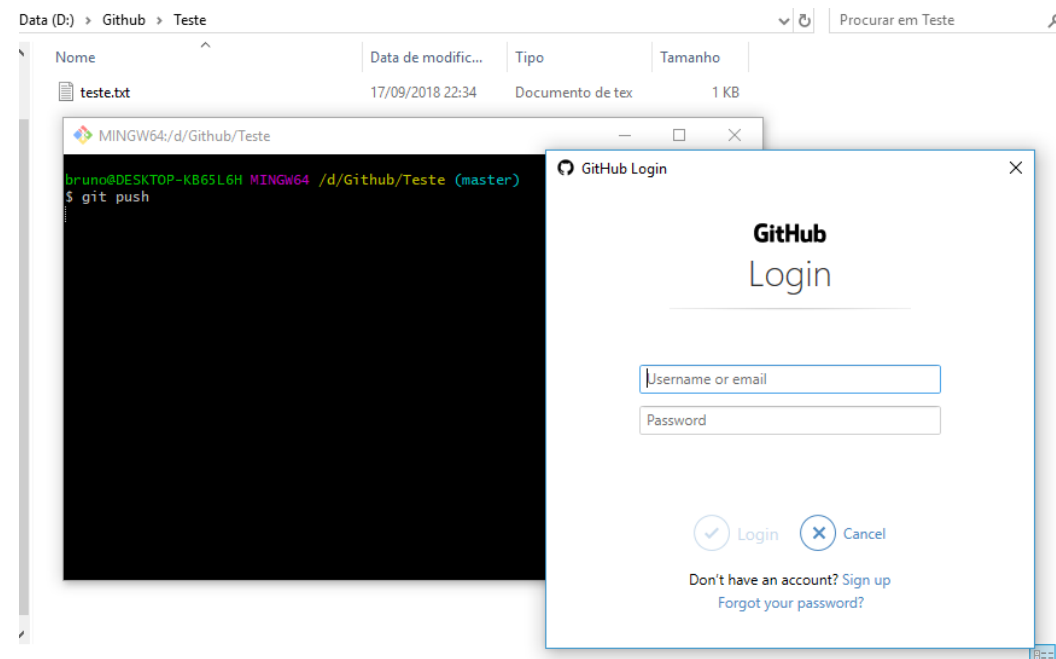
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

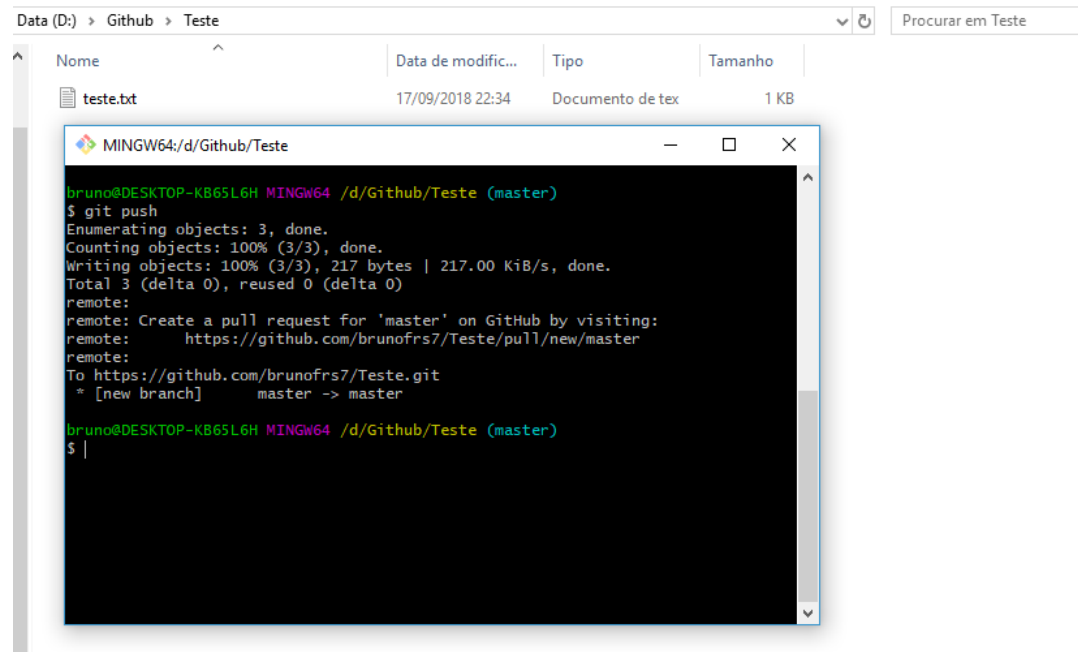
        teste.txt

nothing added to commit but untracked files present (use "git add" to track)
bruno@DESKTOP-KB65L6H MINGW64 /d/Github/Teste (master)
$ git add teste.txt
bruno@DESKTOP-KB65L6H MINGW64 /d/Github/Teste (master)
$ git commit -m "Teste para o Github"
[master (root-commit) 9573843] Teste para o Github
1 file changed, 1 insertion(+)
 create mode 100644 teste.txt
bruno@DESKTOP-KB65L6H MINGW64 /d/Github/Teste (master)
$ |
```

GitHub



GitHub



```
bruno@DESKTOP-KB65L6H MINGW64 /d/Github/Teste (master)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 217 bytes | 217.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/brunofrs7/Teste/pull/new/master
remote:
To https://github.com/brunofrs7/Teste.git
 * [new branch]      master -> master

bruno@DESKTOP-KB65L6H MINGW64 /d/Github/Teste (master)
$ |
```

GitHub

