

## Manual de Instalação do Git no computador

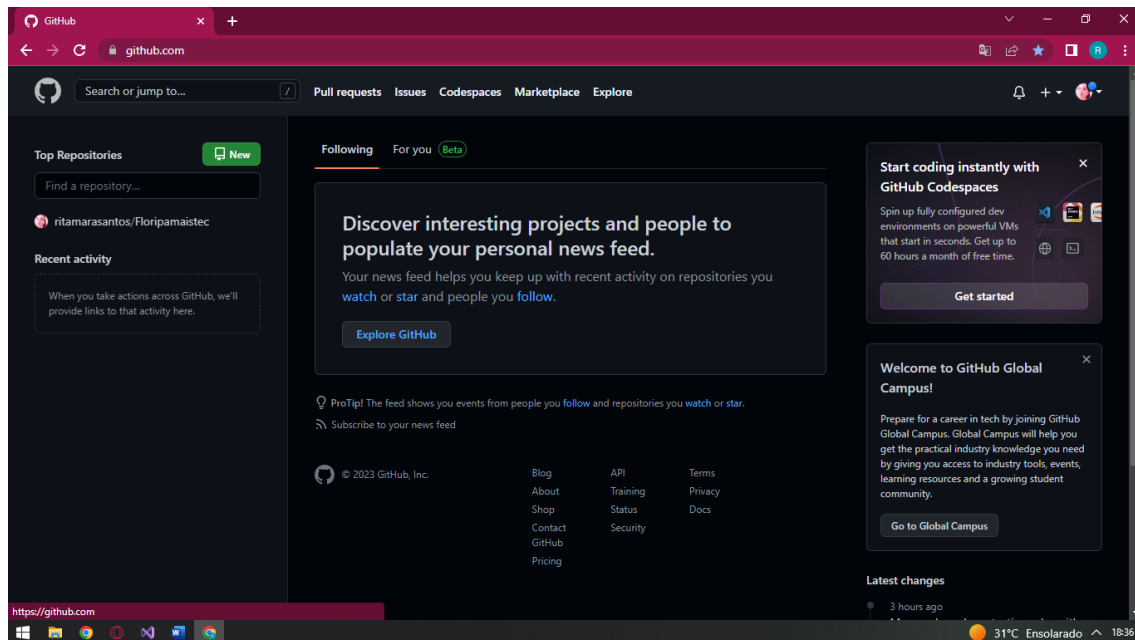
### 1-Instalação GitHub

<https://www.github.com/>

- Clique em inscrever-se (Sign up)
- Crie uma senha forte
- Crie um nome de usuário(que não esteja em uso. Ex.: Nome\_nome ou nomenome)
- Será enviado um e-mail para confirmação de conta.
- Fará algumas perguntas sobre a criação da conta (estudante)
- Continuar free
- Digite seu e-mail pessoal

Conta criada!

- Entrar em perfil, configurações, e-mails e vincular a conta de estudante para ter pacotes free.
- Acessar <https://education.github.com/pack>
- clicar em pacote, escolher estudante
- (Enviar print da tela para confirmar matrícula no curso. (Deve conter data)



## 2-Instalação do GitBash

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

- selecionar 64bits
- clicar em next, next, marcar todos next, install e finish.
- Abrir e digitar o comando:
- Git – version, enter
- Git –help, enter

### Comandos:

```
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>] [--config-env=<name>=<envvar>]
      <command> [<args>]
```

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

```
MINGW64/c/Users/computer/Desktop/Exercicios FullStack/Floripamaitec
$ git --help
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path<path>] [--html<path>] [--man<path>] [--info<path>]
          [-p | --paginate] [-P | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir<path>] [--work-tree<path>] [--namespace<name>]
          [--super-prefix<path>] [--config-env<name>=<envvar>]
          <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflow)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

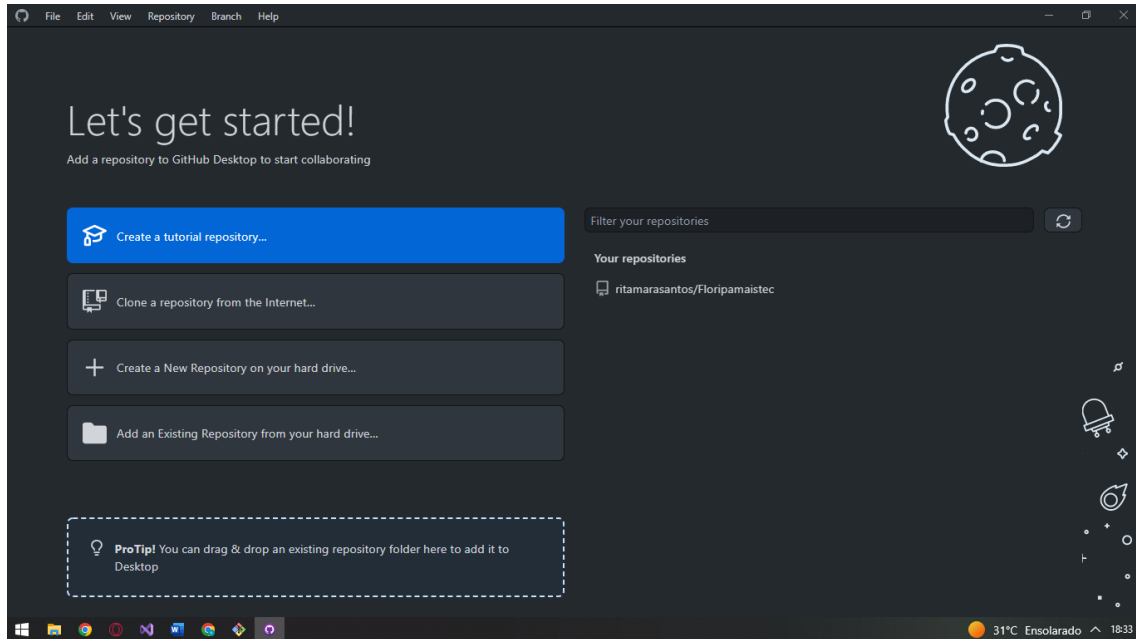
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

computer@DESKTOP-B00GIMS MINGW64 ~/Desktop/Exercicios FullStack/Floripamaitec (
main)
$ |
```

### 3-Instalação do GitHub Desktop

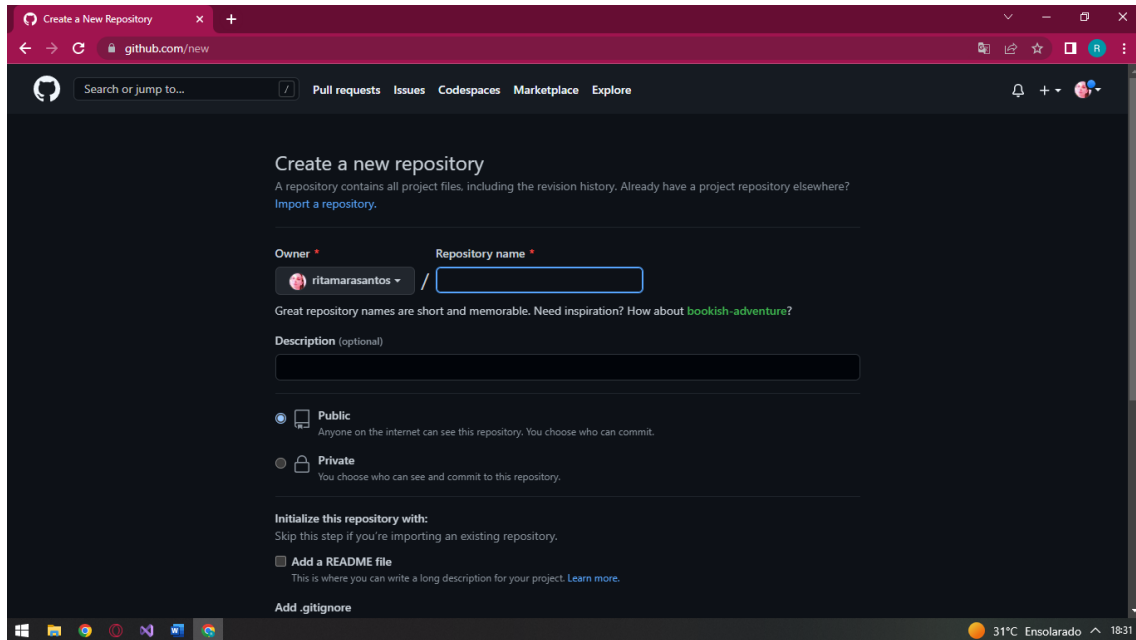
<https://desktop.github.com>

- **baixar versão 64 bits**
- **Clicar em vincular conta, autorizar área de trabalho, acertar o e-mail como pessoal e finish.**



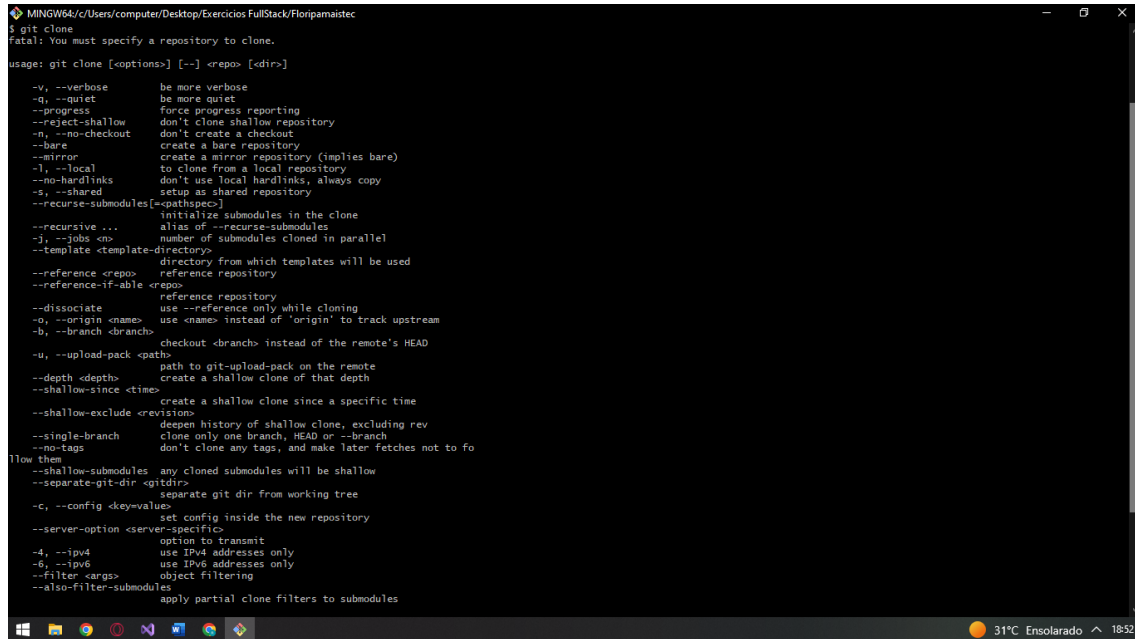
#### 4-Instalar e configurar

- **Página github, perfil, seus repositórios, novo, nome repositório “Floripamaistec”, criar uma descrição, público, add README, criar repositório.**



## 5-<>CODE

- **HTTPS – Open withtth GitHub Desktop ou copiar o código e baixar esta pasta no documento onde irá usar o arquivo de exercícios.**
- **No Windows escolher a pastinha e clicar com direito do mouse em Git Bash Here.**



```
MINGW64/C:/Users/compute/Desktop/Exercicios FullStack/Floripamaistec
$ git clone
fatal: You must specify a repository to clone.

usage: git clone [<options>] [--] <repo> [<dir>]

    -v, --verbose            be more verbose
    -q, --quiet             be more quiet
    --progress              force progress reporting
    --reject-shallow        don't clone shallow repository
    -n, --no-checkout        don't create a checkout
    --bare                  create a bare repository
    --mirror                create a mirror repository (implies bare)
    -l, --local              to clone from a local repository
    --no-hardlinks           don't use local hardlinks, always copy
    -s, --shared             setup as shared repository
    --recurse-submodules[=<pathspec>]
                            initialize submodules in the clone
    --recursive ...         alias of --recurse-submodules
    -j, --jobs <n>          number of submodules cloned in parallel
    --template <template-directory>
                            directory from which templates will be used
    --reference <repo>      reference repository
    --reference-if-able <repo>
                            reference repository
    --dissociate             use --reference only while cloning
    -o, --origin <name>     use <name> instead of 'origin' to track upstream
    -b, --branch <branch>   checkout <branch> instead of the remote's HEAD
    -u, --upload-pack <path>
                            path to git-upload-pack on the remote
    --depth <depth>         create a shallow clone of that depth
    --shallow-since <time>   create a shallow clone since a specific time
    --shallow-exclude <revision>
                            deepen history of shallow clone, excluding rev
    --single-branch          clone only one branch, HEAD or --branch
    --no-tags                don't clone any tags, and make later fetches not to fo
clone them
    --shallow-submodules    any cloned submodules will be shallow
    --separate-git-dir <gitdir>
                            separate git dir from working tree
    -c, --config <key=value>
                            set config inside the new repository
    --server-option <server-specific>
                            option to transmit
    -4, --ipv4               use IPv4 addresses only
    -6, --ipv6               use IPv6 addresses only
    --filter <args>          object filtering
    --also-filter-submodules
                            apply partial clone filters to submodules
```