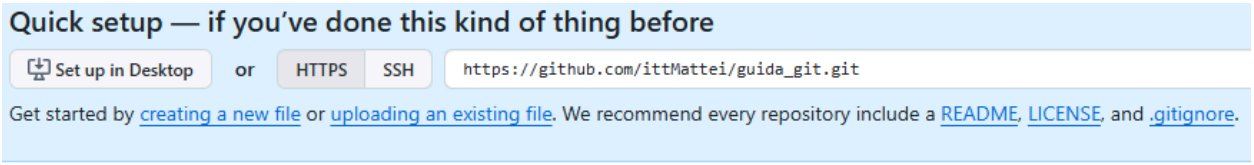
MINI GUIDA GIT

In questa prima immagine vediamo come è possibile collegare un repository on-line con uno in locale. Dopo essere andati su **github** e aver creato un repository vuoto, ci apparirà questa schermata:



...or create a new repository on the command line

```
echo "# guida_git" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ittMattei/guida_git.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/ittMattei/guida_git.git
git branch -M main
git push -u origin main
```

Le possibilità sono 2, se non ho ancora materiale seguo il primo suggerimento, altrimenti se voglio collegare un repository locale già esistente non devo far altro che comunicare al mio PC che questa cartella farà capo a uno spazio remoto**(git remote add**). Inoltre si deve effettuare un **push** per sincronizzare il remoto con ciò che ho già in locale.

A questo punto posso clonare in qualsiasi luogo questo mio spazio di lavoro attraverso il comando **git clone**:

```
federico.piccinali@PB111 MINGW64 ~/Desktop/prova
$ git clone https://github.com/ittMattei/guida_git.git
Cloning into 'guida_git'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 0), reused 10 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (10/10), 279.96 KiB | 2.86 MiB/s, done.
```

Sarebbe possibile, dal browser, navigare sul sito sopra citato e semplicemente scaricare i file. Questo però ci preclude la possibilità di collaborare.

LAVORO IN LOCALE

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari
$ ls -la
total 12
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:26 ./
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:26 ../

federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari
$ git init
Initialized empty Git repository in C:/Users/federico.piccinali/Desktop/alimentari/.git/

federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ ls -la
total 16
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:27 ./
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:26 ../
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:27 .git/
```

Inizializzo un repository locale git nella cartella “alimentari”. Cosa è successo? viene creata una cartella nascosta .git che conterrà tutte le informazioni per il **versionamento**.

Adesso creiamo un paio di file e cominciamo a tracciarli:

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ echo "Repo per la mia lista della spesa" > README.md

federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ echo "banana" > lista_spesa.txt
```

Per cominciare bisogna aggiungerli all’**area di stage:**

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git add .
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'lista_spesa.txt', LF will be replaced by CRLF the next time Git touches it
```

Per il momento ignoriamo i warning e proseguiamo per il commit:

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git commit -m "Aggiunta una banana alla lista!"
[master (root-commit) 6e566e8] Aggiunta una banana alla lista!
 Committer: PICCINALI FEDERICO <federico.piccinali>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 2 files changed, 2 insertions(+)
 create mode 100644 README.md
 create mode 100644 lista_spesa.txt
```

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git log
commit 6e566e88ef18c33df757c9b30da67efab64a0ce3 (HEAD -> master)
Author: PICCINALI FEDERICO <federico.piccinali>
Date:   Wed May 7 15:37:03 2025 +0200

    Aggiunta una banana alla lista!
```

Diamo uno sguardo al commit con git log: tutti gli oggetti usati da Git sono identificati con una stringa di hash SHA-1 a 40 caratteri hex.

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git log --format=raw
commit 6e566e88ef18c33df757c9b30da67efab64a0ce3
tree 0d9ffbc93a54b3dd82ff15a482449e80beb56d3e
author PICCINALI FEDERICO <federico.piccinali> 1746625023 +0200
committer PICCINALI FEDERICO <federico.piccinali> 1746625023 +0200

    Aggiunta una banana alla lista!
```

Ora, per approfondire un attimo diamo l'opzione raw al log, vediamo la comparsa di un nuovo elemento: un albero.

L'albero è l'equivalente di una directory, ovvero un contenitore di file o altre directory.
Ma quali altri oggetti usa GIT? In tutto sono 4:

1. commit
2. alberi
3. blob
4. tag annotati

Vediamo cos'è un blob:

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git cat-file -p 6e56
tree 0d9ffbc93a54b3dd82ff15a482449e80beb56d3e
author PICCINALI FEDERICO <federico.piccinali> 1746625023 +0200
committer PICCINALI FEDERICO <federico.piccinali> 1746625023 +0200

Aggiunta una banana alla lista!

federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git cat-file -p 0d9ff
100644 blob c15cbb7ce364bac840d5572826e4efbd18b042de    README.md
100644 blob 637a09b86af61897fb72f26bfb874f2ae726db82    lista_spesa.txt
```

Il criptico comando git cat mi permette di visualizzare il contenuto di un file (è un plumbing command). Ciò che accade qui è che visualizzando il contenuto dell'albero 0d9ff mi compare il suo contenuto, ovvero i due file blob.
Ora cosa succede se visualizzo il contenuto dei due oggetti?

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git cat-file -p c15cbb
Repo per la mia lista della spesa

federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ git cat-file -p 637a
banana
```

Ottengo proprio il contenuto dei file.
Notare che l'hash di "banana" non cambia…

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ echo "banana" | git hash-object --stdin
637a09b86af61897fb72f26bfb874f2ae726db82
```

quindi l'hash del file spesa è l'hash della stringa banana in pratica.
Da ciò dobbiamo capire un importante modo per risparmiare di git: se due file sono diversi ma con lo stesso contenuto avranno UN SOLO BLOB.

Vediamo dove sono salvate queste informazioni:

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ ll .git/
total 13
-rw-r--r-- 1 federico.piccinali 1049089  32 May  7 15:37 COMMIT_EDITMSG
-rw-r--r-- 1 federico.piccinali 1049089  23 May  7 15:27 HEAD
-rw-r--r-- 1 federico.piccinali 1049089 130 May  7 15:27 config
-rw-r--r-- 1 federico.piccinali 1049089  73 May  7 15:27 description
drwxr-xr-x 1 federico.piccinali 1049089   0 May  7 15:27 hooks/
-rw-r--r-- 1 federico.piccinali 1049089 217 May  7 15:37 index
drwxr-xr-x 1 federico.piccinali 1049089   0 May  7 15:27 info/
drwxr-xr-x 1 federico.piccinali 1049089   0 May  7 15:37 logs/
drwxr-xr-x 1 federico.piccinali 1049089   0 May  7 15:37 objects/
drwxr-xr-x 1 federico.piccinali 1049089   0 May  7 15:27 refs/
```

Nella cartella obj:

```
federico.piccinali@LABSIS61 MINGW64 ~/Desktop/alimentari (master)
$ ll .git/objects/
total 0
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:37 0d/
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:35 63/
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:37 6e/
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:35 c1/
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:27 info/
drwxr-xr-x 1 federico.piccinali 1049089 0 May  7 15:27 pack/
```

Ora facciamo qualche altra operazione e visualizziamo lo stato..

```
ProFP@LAPTOP-DNED4V7H MINGW64 ~/Desktop/guida_git (main)
$ git log --oneline --decorate --graph
* 3d2ae84 (HEAD -> main, origin/main, origin/HEAD) aggiunta un'arancia
* 77a69f8 aggiunta una mela
* 621c018 Aggiunta banana alla lista della spesa
* b6b9c48 Update README.md
* c8665d6 Aggiornata guida pdf con immagine clone
* 07f2b5c Aggiunta guida pdf
* eef4d82 Aggiunto moto
* 124b678 Aggiunto auto alla lista
```

Ho fatto alcuni commit sul ramo main. Vediamo che l'etichetta HEAD punta sempre all'ultimo commit (che saranno anche le modifiche visibili nella cartella reale).

```
ProFP@LAPTOP-DNED4V7H MINGW64 ~/Desktop/guida_git (main)
$ cat .git/refs/heads/main
3d2ae84f854d7fa5d1ec7b1312210658dfa1fed9
```

infatti il contenuto di HEAD è proprio l'ultimo commit.

Ora facciamo una biforcazione aggiungendo un nuovo ramo:

```
ProFP@LAPTOP-DNED4V7H MINGW64 ~/Desktop/guida_git (main)
$ git branch piccoli_frutti

ProFP@LAPTOP-DNED4V7H MINGW64 ~/Desktop/guida_git (main)
$ gl
* 3d2ae84 (HEAD -> main, origin/main, origin/HEAD, piccoli_frutti) aggiunta un'arancia
* 77a69f8 aggiunta una mela
* 621c018 Aggiunta banana alla lista della spesa
* b6b9c48 Update README.md
* c8665d6 Aggiornata guida pdf con immagine clone
* 07f2b5c Aggiunta guida pdf
* eef4d82 Aggiunto moto
* 124b678 Aggiunto auto alla lista
```

Come si può vedere ora nell'ultimo commit c'è una biforcazione..
Lasciamo ora il ramo main per quello nuovo e facciamo un'aggiunta: