

TECNICHE DI RAPPRESENTAZIONE E MODELLIZZAZIONE DEI DATI

– Part 1 –

(2 CFU out of 6 total CFU)

Link moodle: <https://moodle2.units.it/course/view.php?id=11703>

Teams code: 0ftoqj8

The Integrated Development Environment

Setup a working environment:

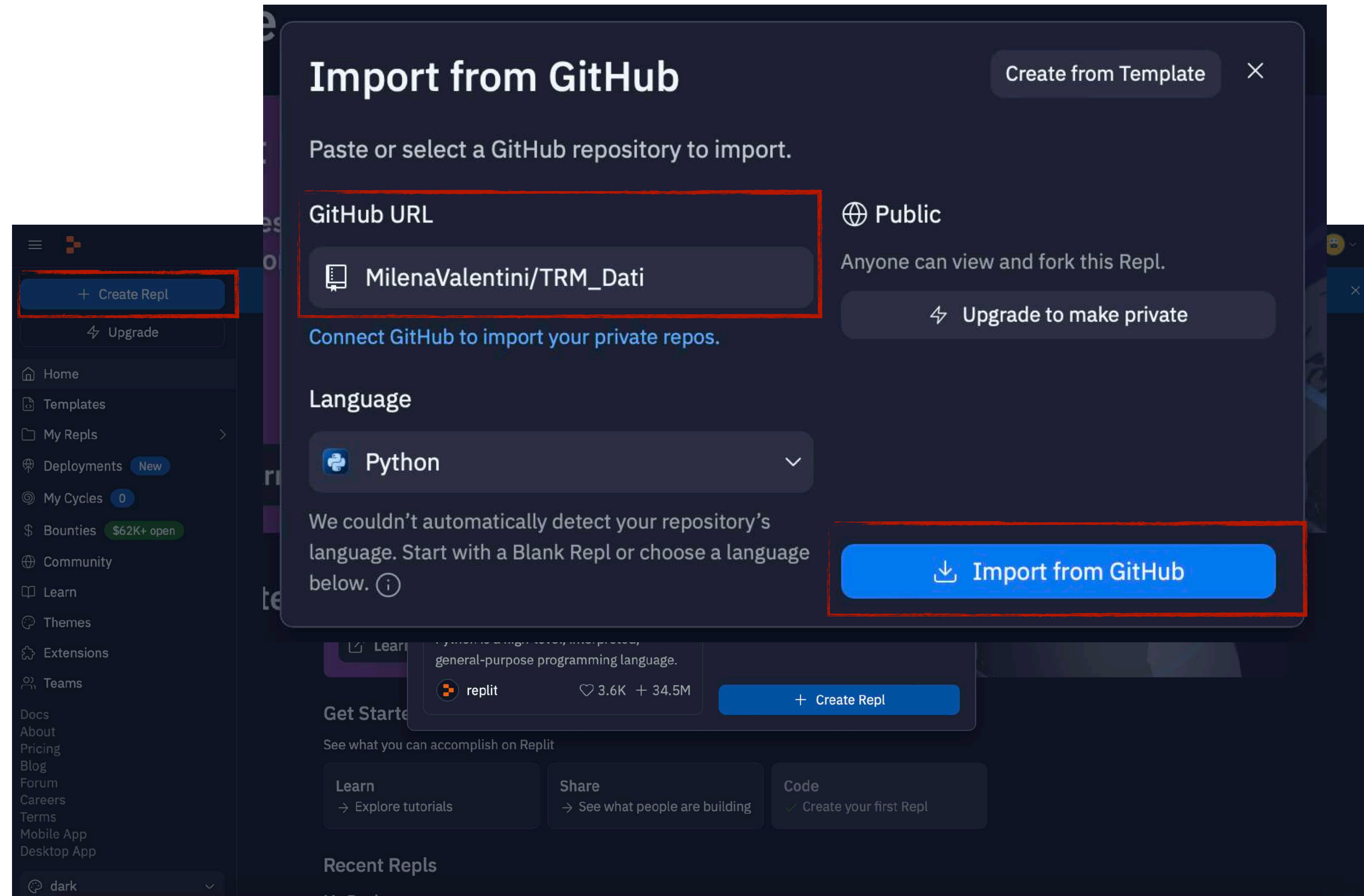
Register on Git

Create a public repository on Git

Register on repl.it (replit.com)

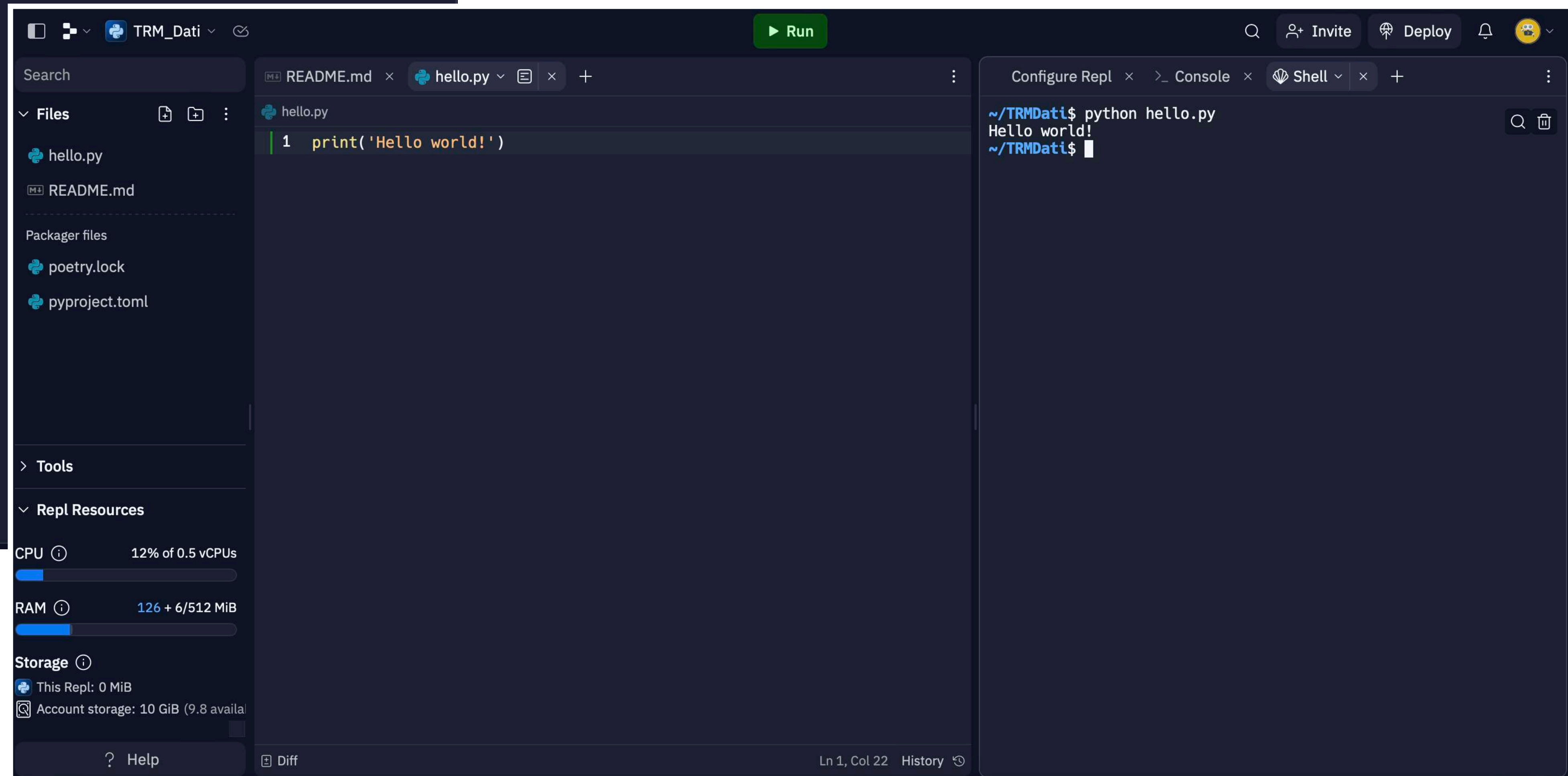
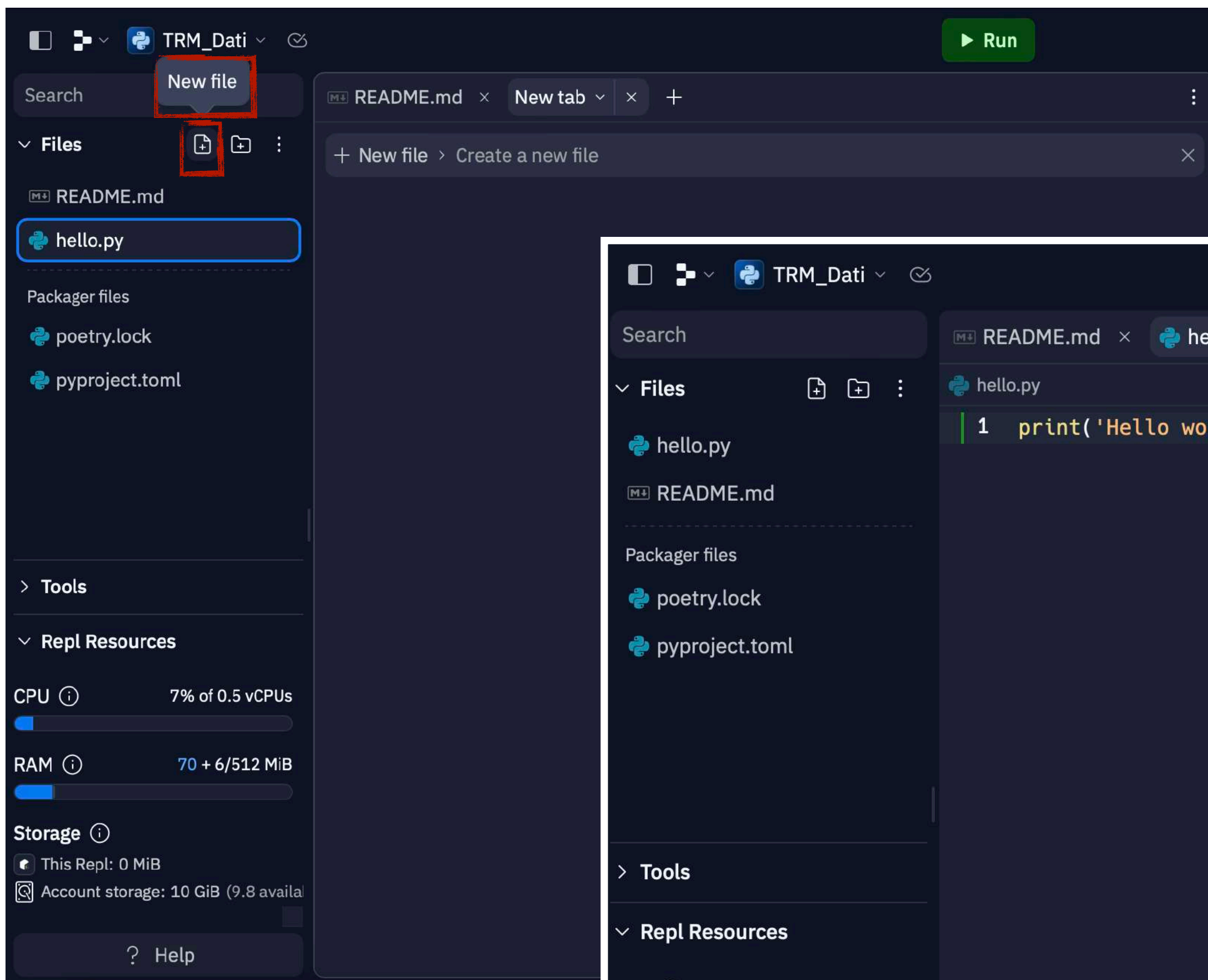
Create a new repl project by importing from Git the repository you have just created there

Set Python as default language



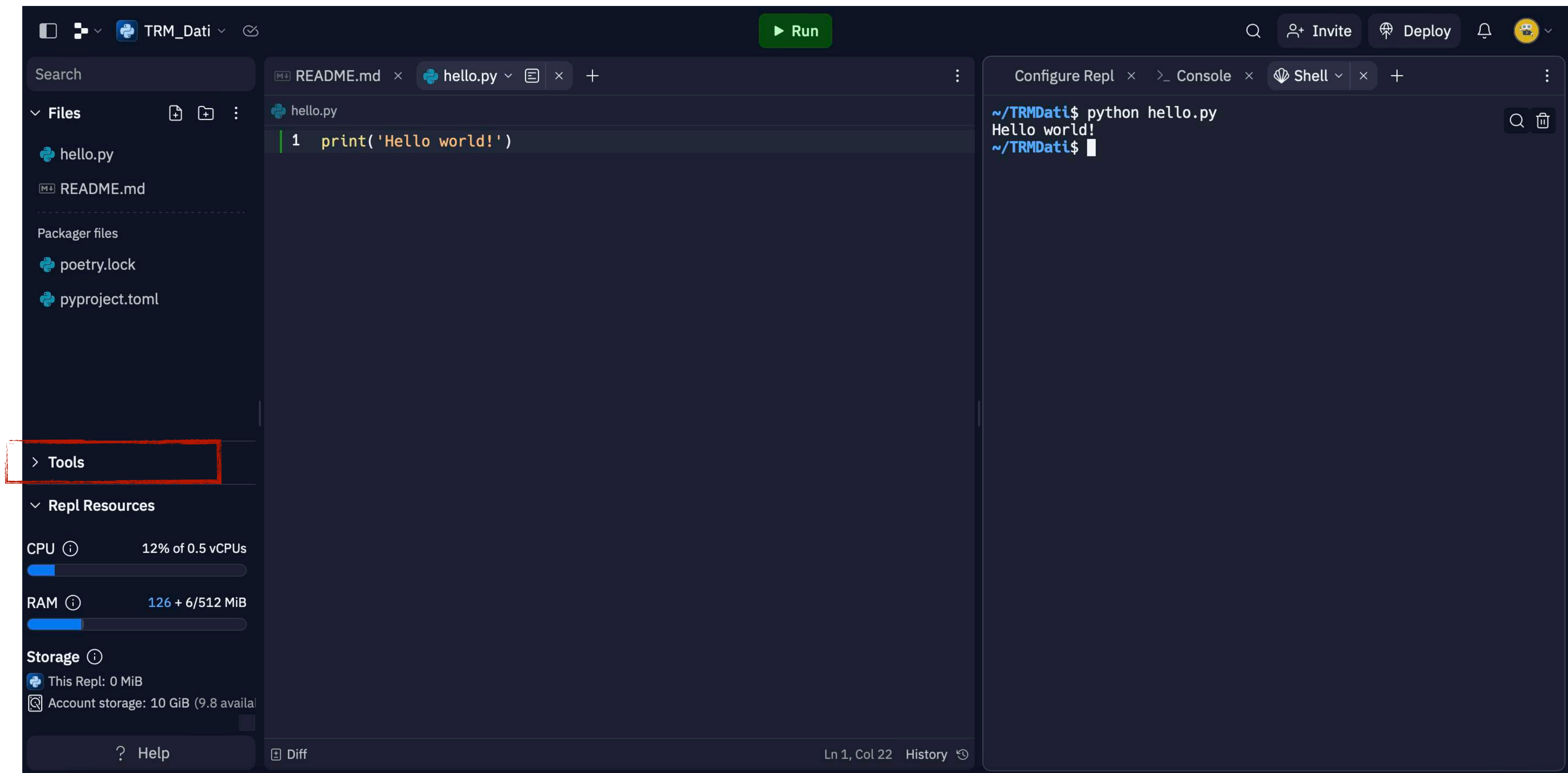
The Integrated Development Environment

Create a new file to produce the first working script



The Integrated Development Environment

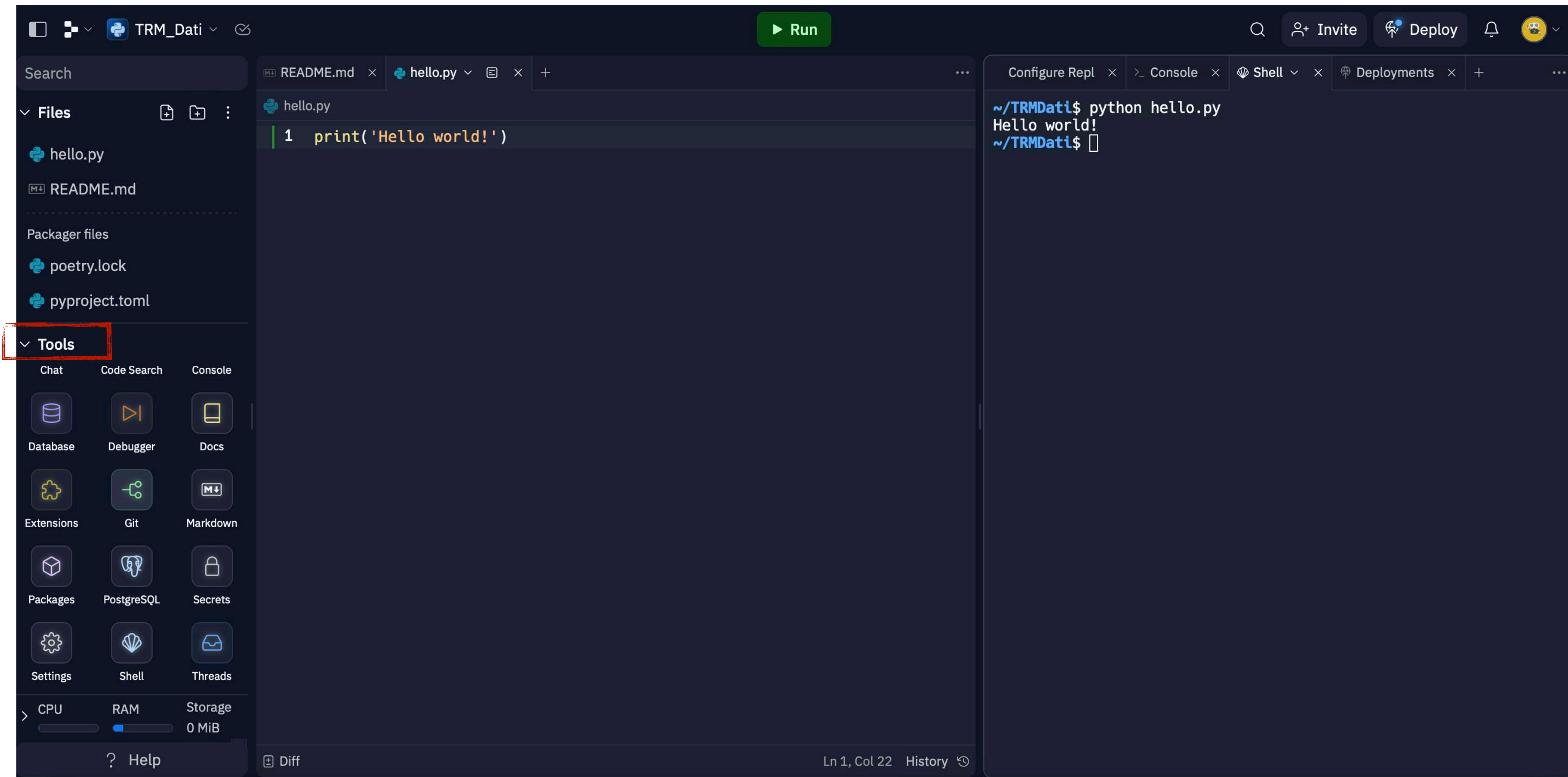
Connecting replit to Git



Other useful material: <https://replit.com/talk/learn/Replit-Git-Tutorial/23331>

The Integrated Development Environment

Connecting replit to Git




The Integrated Development Environment

Connecting replit to Git

The screenshot displays the Replit IDE interface. On the left sidebar, the 'Tools' section is expanded, and the 'Git' tool icon is highlighted with a red box. The main editor area shows a Python file named 'hello.py' with the code `print('Hello world!')`. On the right-hand side, the 'Remote Updates' panel shows a notification: 'Can't push or pull from GitHub. This Repl has a GitHub repository as a remote, but you are not connected to GitHub.' Below this notification is a blue button labeled 'Connect to GitHub'. The bottom right panel shows the 'Commit' section with a list of 5 changed files: `.replit`, `hello.py`, `poetry.lock`, `pyproject.toml`, and `replit.nix`, all marked as 'Added'.

The Integrated Development Environment



Install & Authorize Replit

Install & Authorize on your personal account Milena Valentini

All repositories
This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

Only select repositories
Select at least one repository.
Also includes public repositories (read-only).

with these permissions:

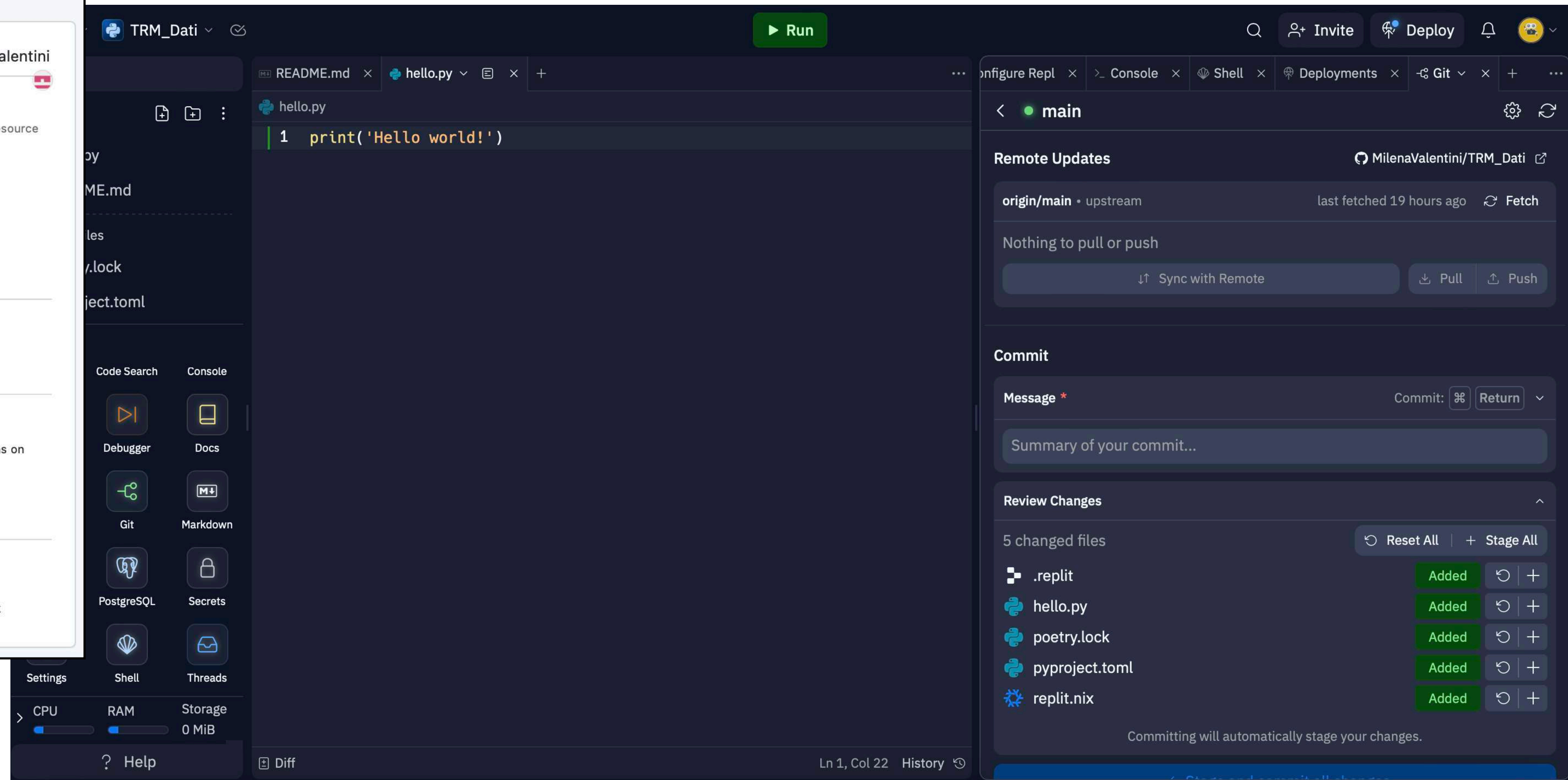
- ✓ Read access to metadata
- ✓ Read and write access to administration and code

User permissions
Installing and authorizing Replit immediately grants these permissions on your account: MilenaValentini.

- ✓ Read access to email addresses

Install & Authorize Cancel

Next: you'll be redirected to <https://replit.com/auth/github/callback>

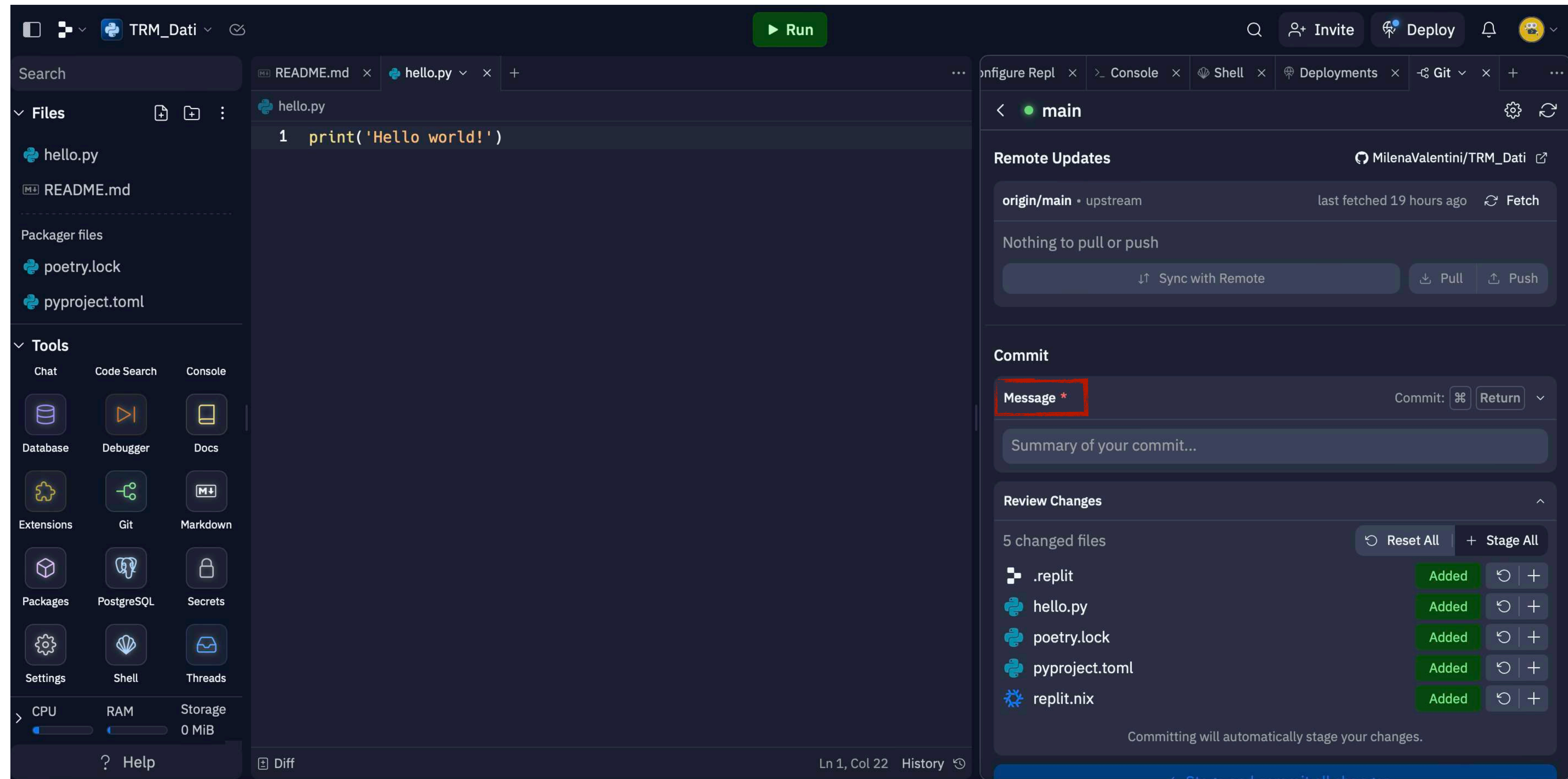


The screenshot shows the Replit IDE interface. At the top, there's a 'Run' button and a search bar. The main area is a code editor with a file named 'hello.py' containing the code: `1 print('Hello world!')`. To the right of the code editor is a 'Commit' panel showing a list of 5 changed files: `.replit`, `hello.py`, `poetry.lock`, `pyproject.toml`, and `replit.nix`, all marked as 'Added'. Below the file list, there's a 'Commit' message field and a 'Commit' button. The bottom of the interface shows system resources like CPU, RAM, and Storage, and a 'Help' button.

Authorize Replit on Git

The Integrated Development Environment

Add message to describe your commit

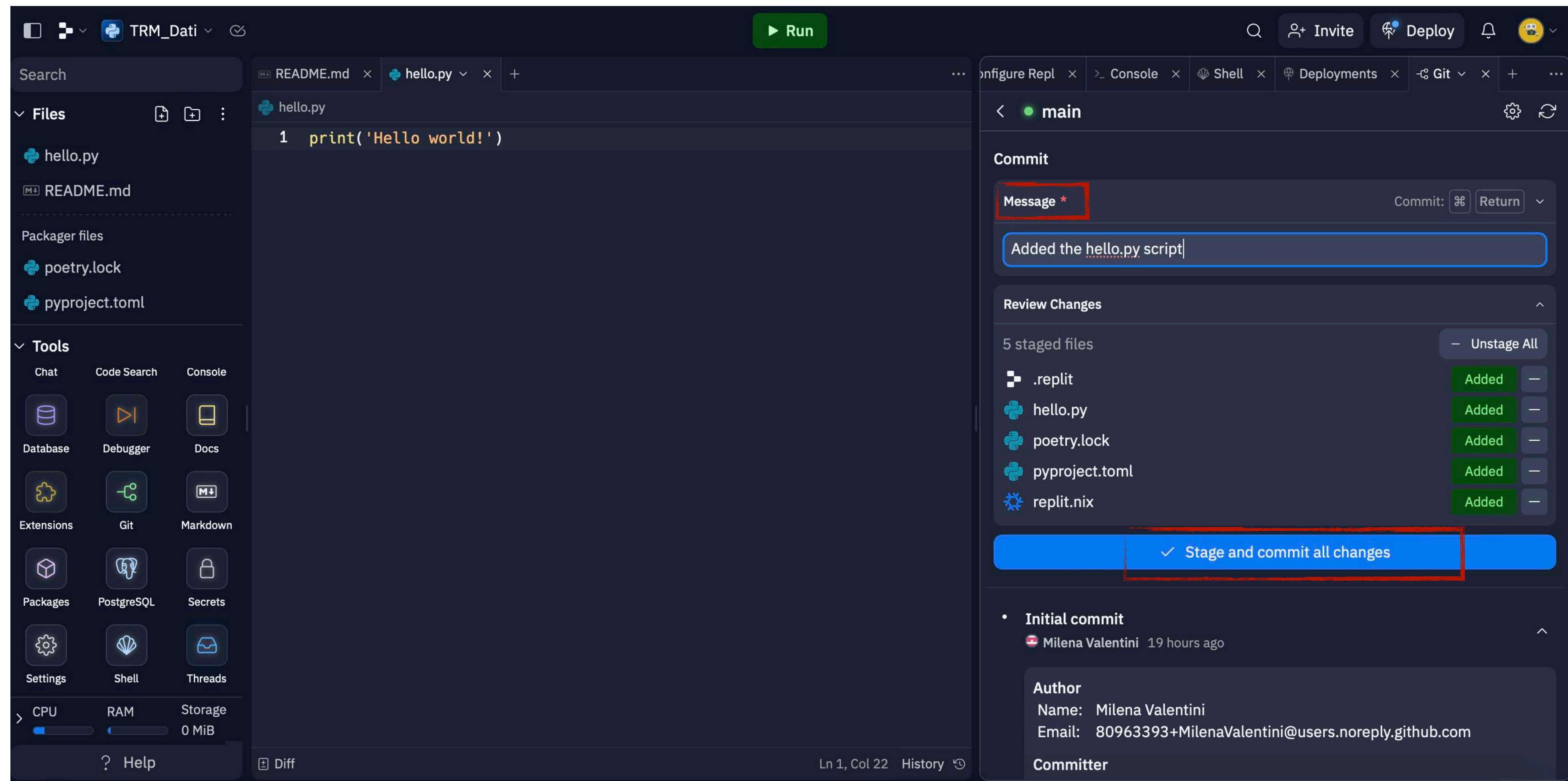


The Integrated Development Environment

Add message to describe your commit

Stage files to commit (the hello.py script, and additional files internal to replit plus related to libraries)

Commit



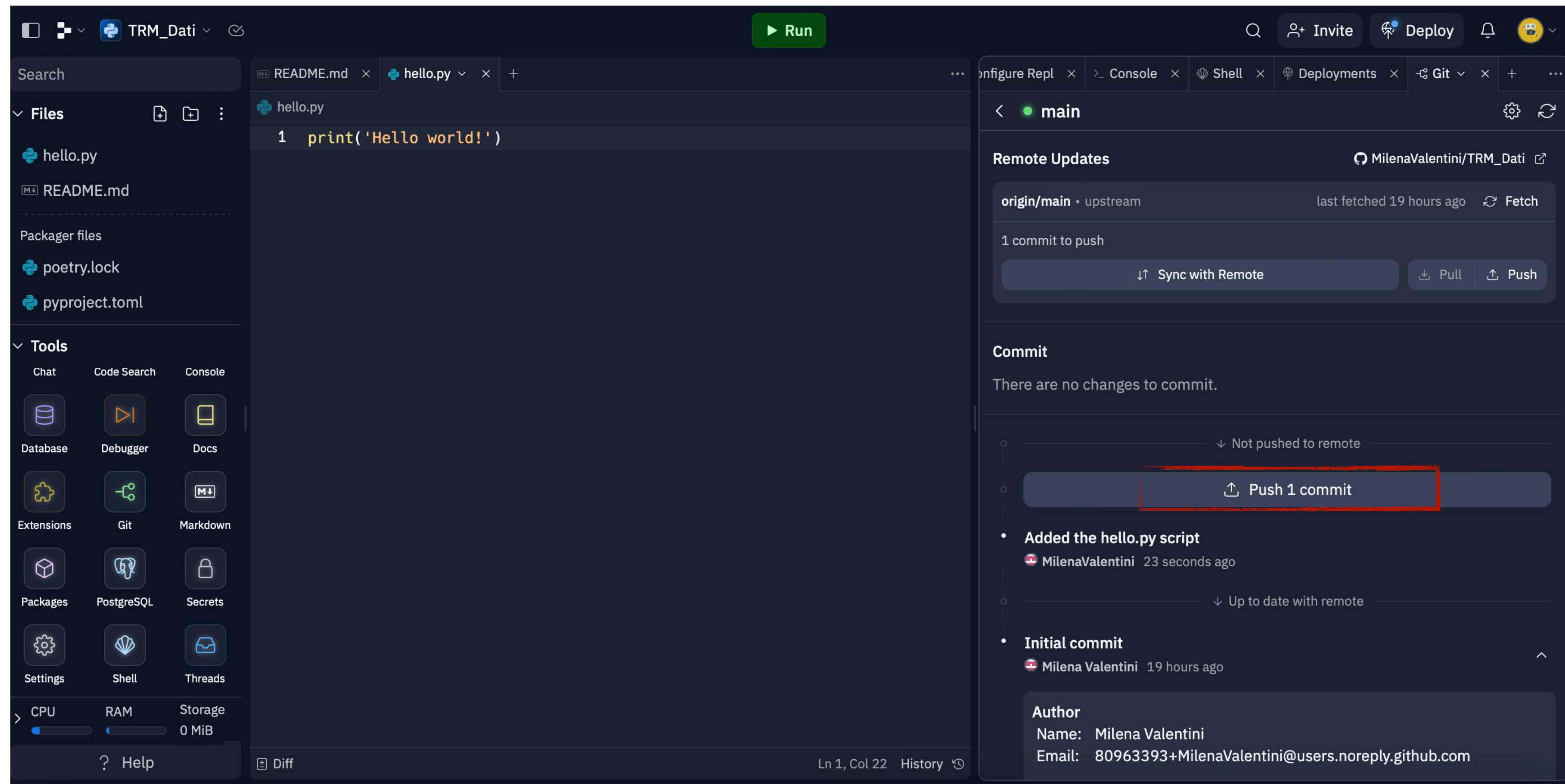
The Integrated Development Environment

Add message to describe your commit

Stage files to commit (the hello.py script, and additional files internal to replit plus related to libraries)

Commit

Push to Git



The Integrated Development Environment

This screenshot shows the GitHub repository page for MilenaValentini / TRM_Dati. The repository is public and has 1 branch (main) and 0 tags. The commit history shows an initial commit by MilenaValentini 19 hours ago. The README.md file content is displayed below the commit history.

TRM_Dati Public

main 1 branch 0 tags

MilenaValentini Initial commit f26615d 19 hours ago 1 commit

README.md Initial commit 19 hours ago

README.md

TRM_Dati

Public repository of the course: Tecniche di Rappresentazione e Modellizzazione Dati

This screenshot shows the GitHub repository page for MilenaValentini / TRM_Dati, updated to show the latest commit. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a new commit by MilenaValentini 1 minute ago. The file list shows the addition of several files, including .replit, README.md, hello.py, poetry.lock, pyproject.toml, and replit.nix. The README.md file content is displayed below the file list.

TRM_Dati Public

main 1 branch 0 tags

MilenaValentini Added the hello.py script 4e32227 1 minute ago 2 commits

.replit Added the hello.py script 1 minute ago

README.md Initial commit 19 hours ago

hello.py Added the hello.py script 1 minute ago

poetry.lock Added the hello.py script 1 minute ago

pyproject.toml Added the hello.py script 1 minute ago

replit.nix Added the hello.py script 1 minute ago

README.md

TRM_Dati

Public repository of the course: Tecniche di Rappresentazione e Modellizzazione Dati

About

Public repository of the course: Tecniche di Rappresentazione e Modellizzazione Dati

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

Nix 58.0% Python 42.0%

Suggested Workflows

Based on your tech stack

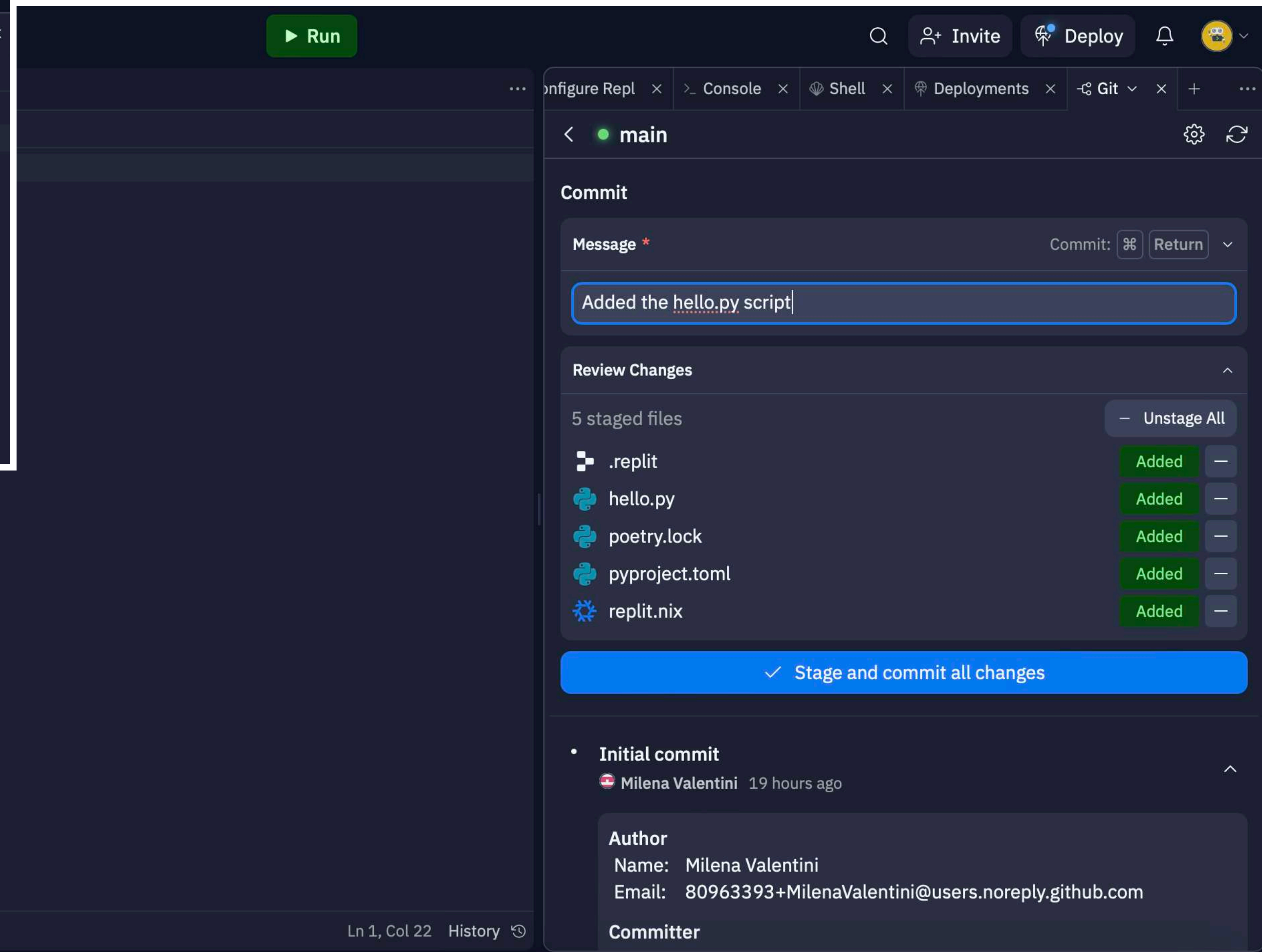
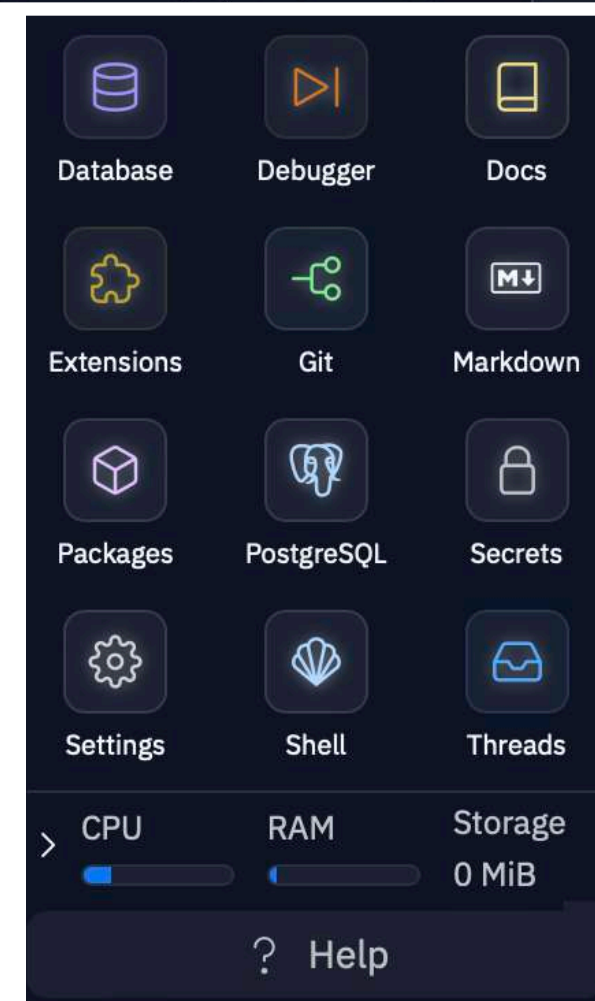
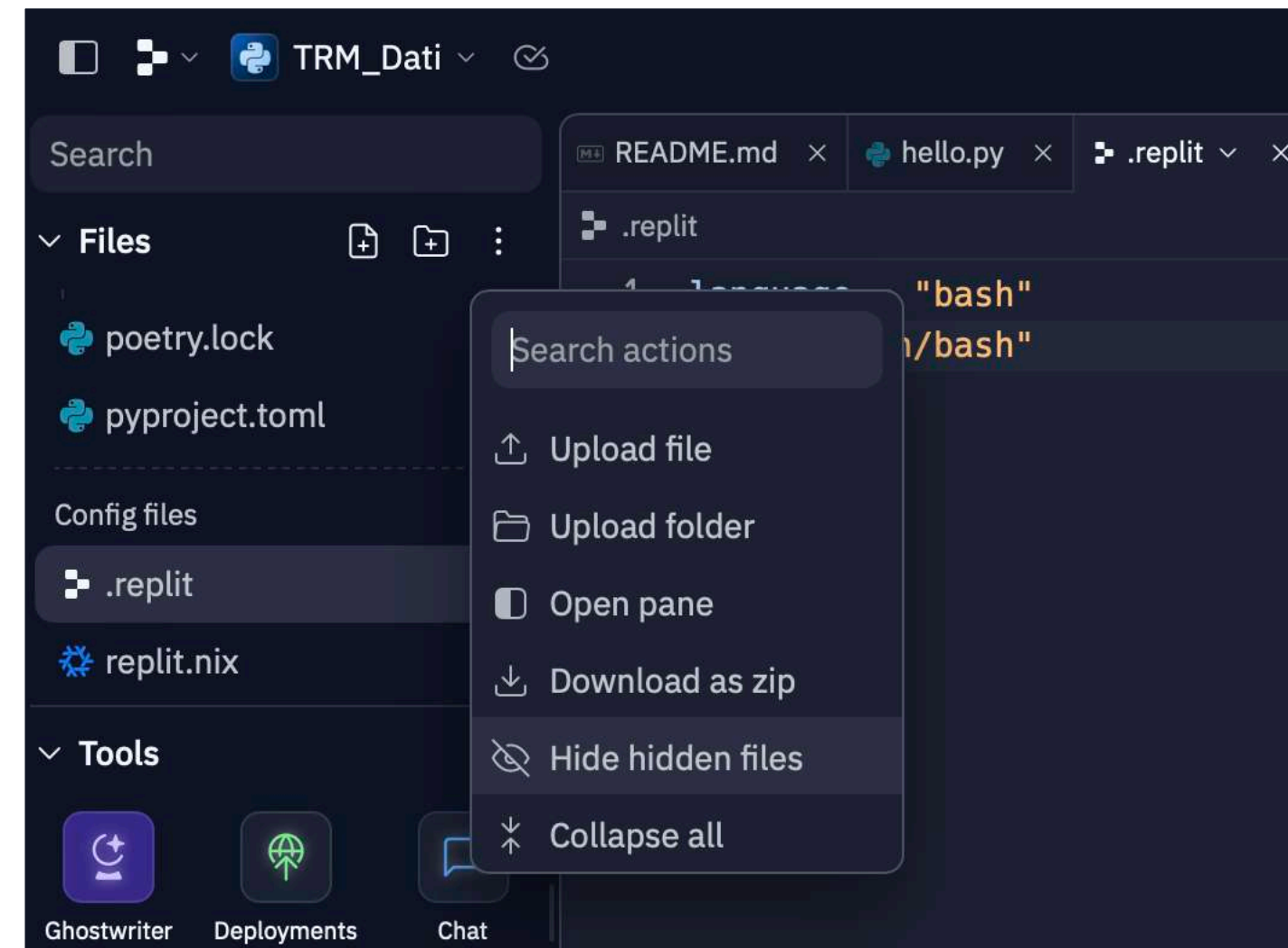
The Git repository has been successfully updated

The Integrated Development Environment

Make sure to configure the hidden file `.replit` as follows:

language = "bash"
run = "/bin/bash"

Stage, commit and push it to Git



Git: Intro

Git is the most used version control system and it's under continuous development

A screenshot of the Git website homepage. At the top left is the Git logo (a red diamond with a white branching diagram) followed by the word "git" in a bold, lowercase font. To the right of the logo is the tagline "--distributed-even-if-your-workflow-isnt". In the top right corner, there is a search bar with a magnifying glass icon and the text "Search entire site...". Below the header, the main content area has a light gray background with a subtle grid pattern. On the left side, there are two paragraphs of text. The first paragraph describes Git as a free and open source distributed version control system. The second paragraph describes Git as easy to learn and having a tiny footprint with lightning fast performance. On the right side, there is a 3D diagram showing several stacks of white papers representing code repositories. These stacks are connected by colored lines (red, blue, yellow) that represent branching and merging operations. The diagram shows a central stack with branches leading to other stacks, illustrating the distributed nature of the system.

git --distributed-even-if-your-workflow-isnt

Search entire site...

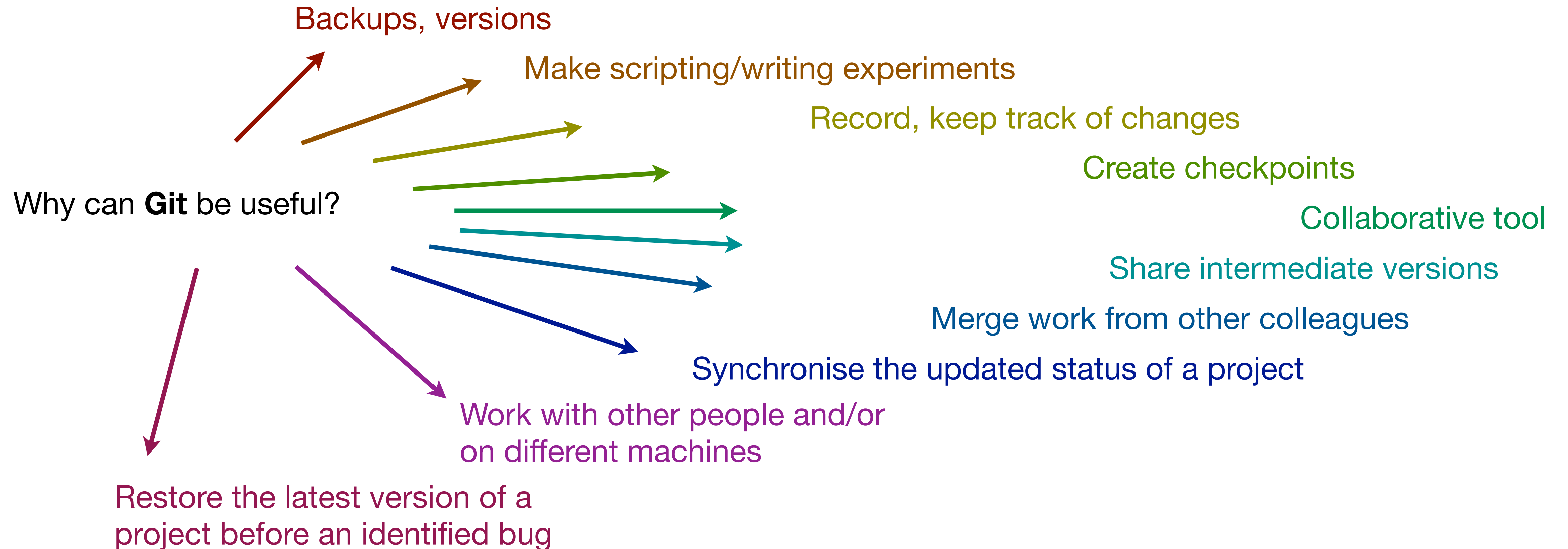
Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

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**.

Git clients: <https://git-scm.com/downloads/guis/>

You can work from the terminal

Git: essentials and how-to

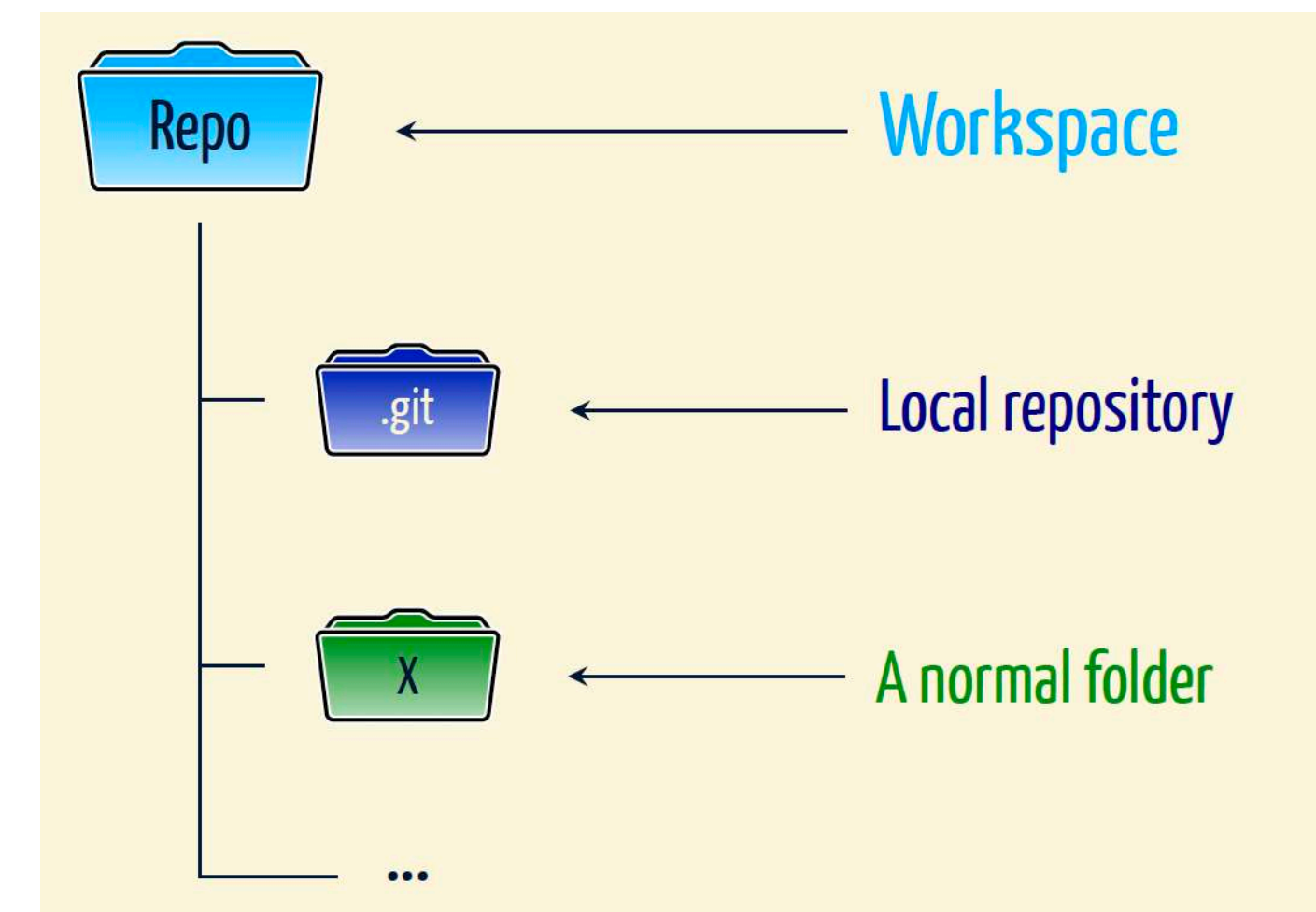


How would that be without Git?

Useful resources:
<https://git-scm.com/book/en/v2>
<https://github.com/AxelKrypton/Git-crash-course>

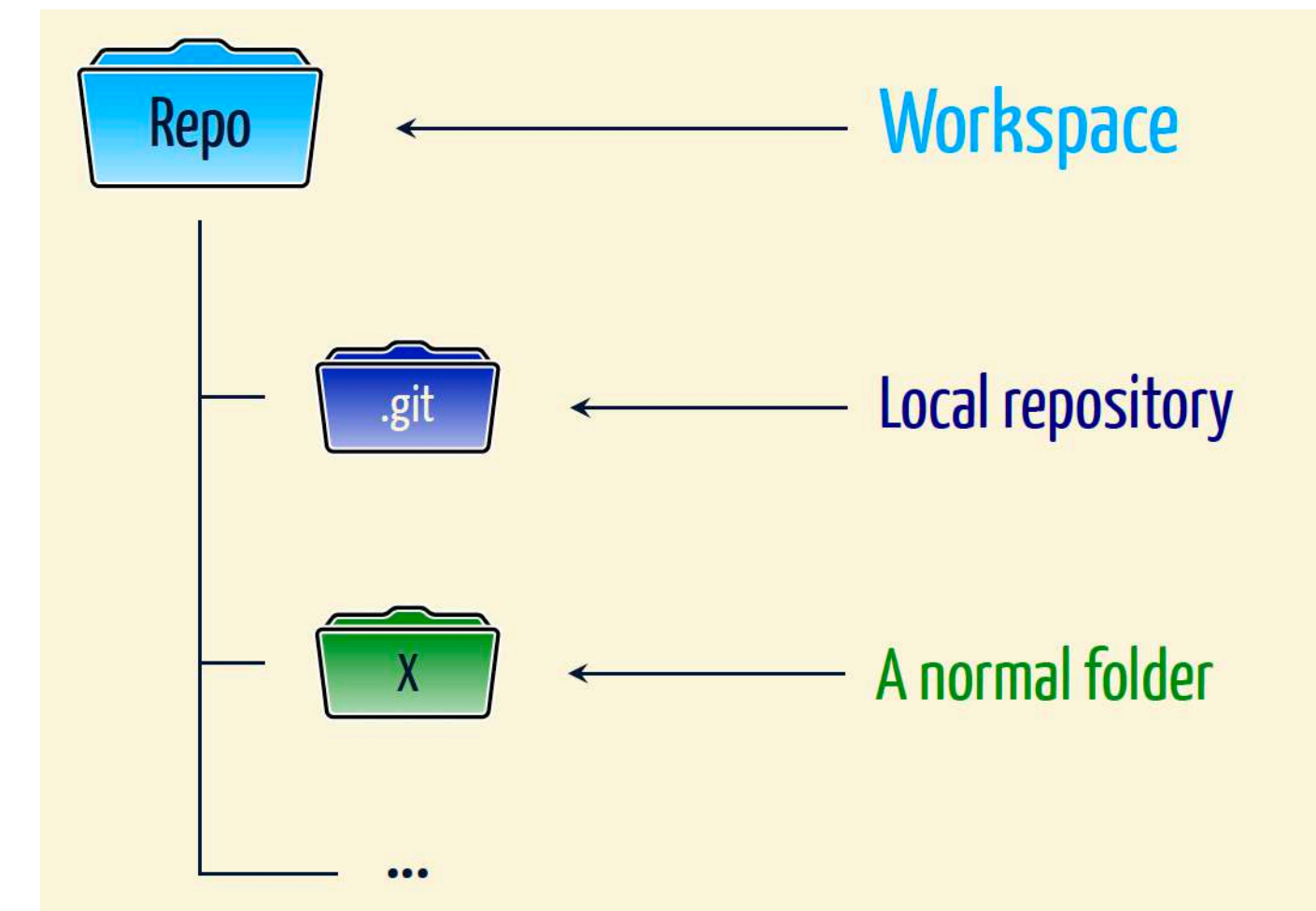
Git: Intro

A repository is a directory containing all the versions of the files

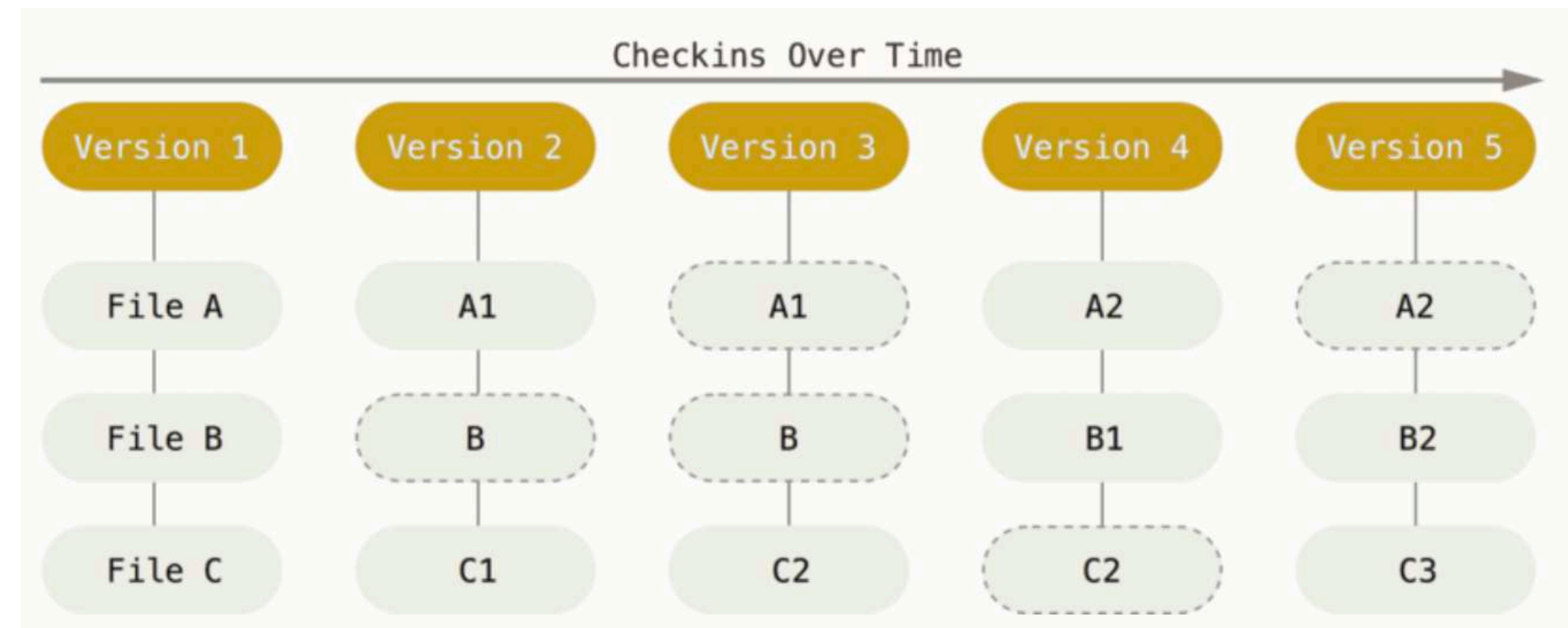


Git: Intro

A repository is a directory containing all the versions of the files



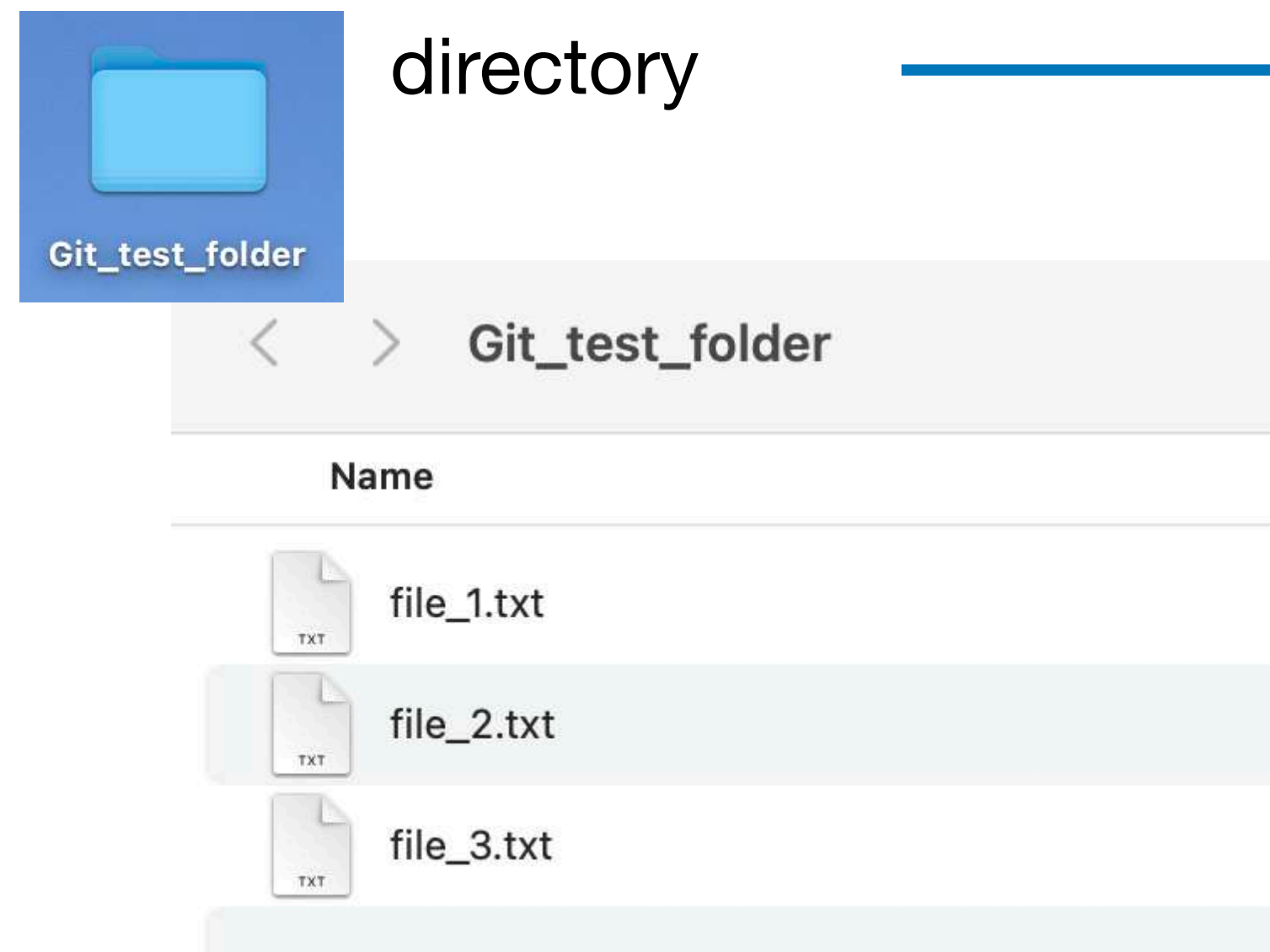
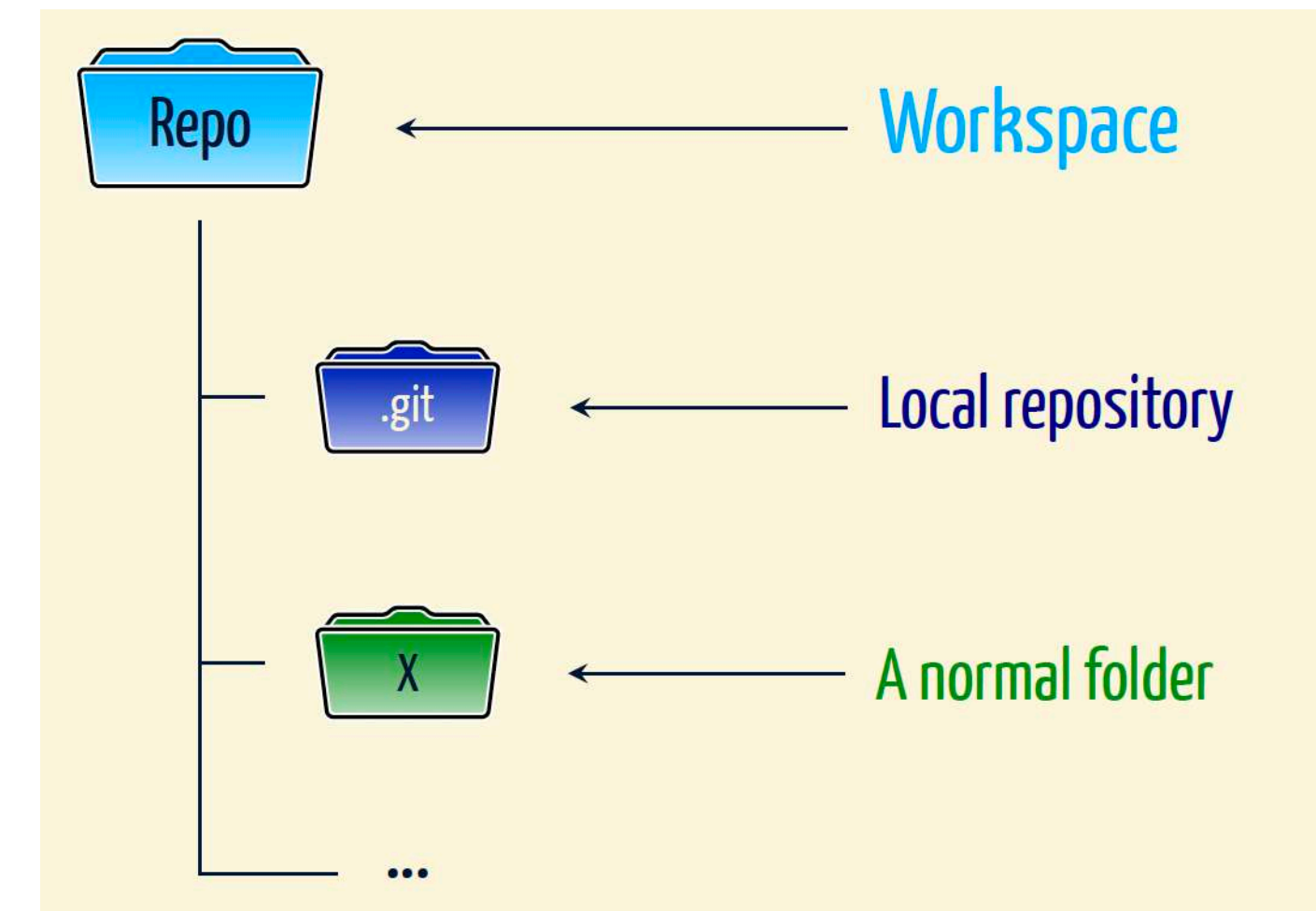
Records are based on snapshots taken over time



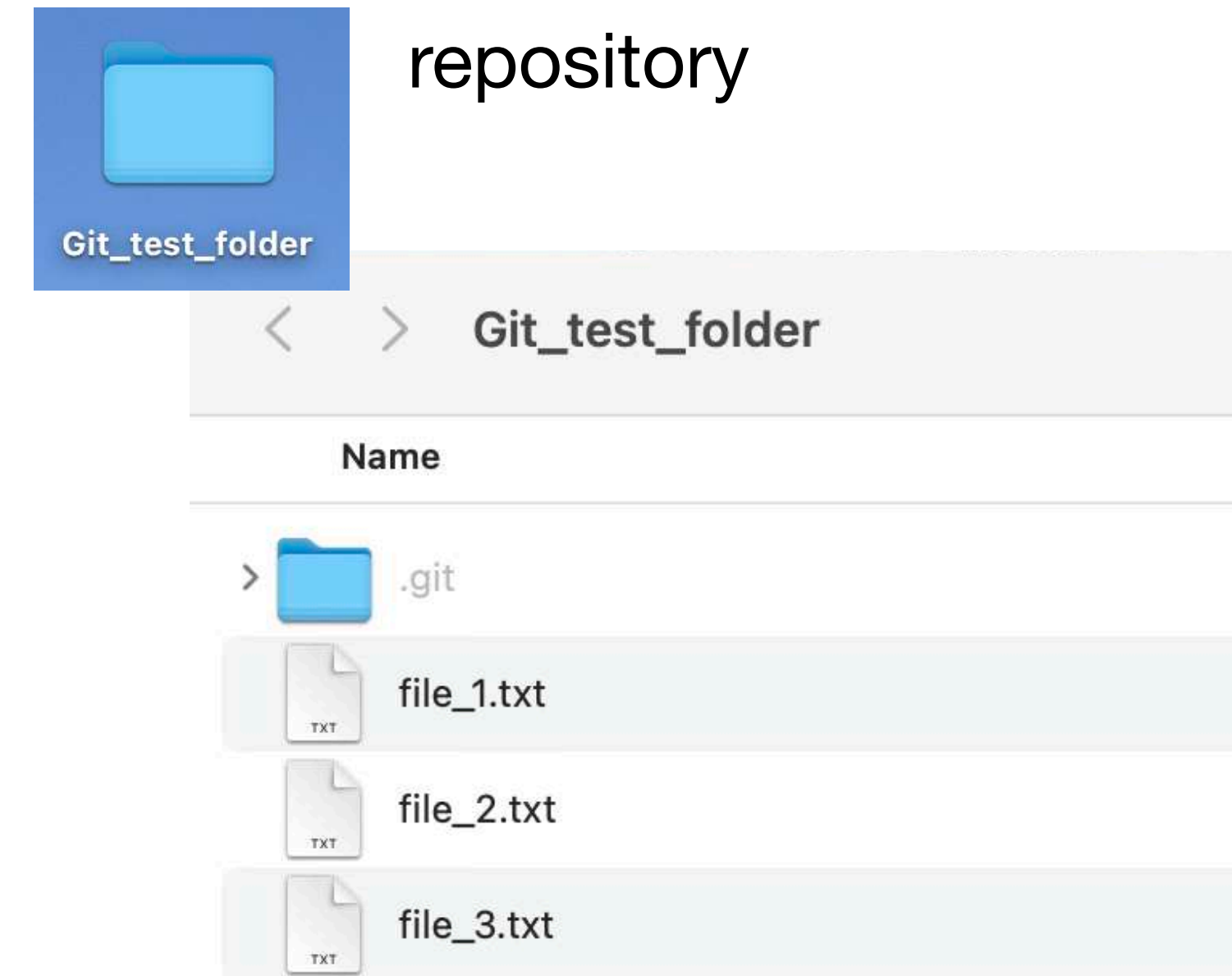
Snapshots are called commit and are referenced by a checksum

Git: Intro

A repository is a directory containing all the versions of the files



git init →



Git: essentials and how-to

```
[(base) MacBook-Pro-2:TRM_Dati_Lezioni milenavalentini$ git
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`)

<code>clone</code>	Clone a repository into a new directory
<code>init</code>	Create an empty Git repository or reinitialize an existing one

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

<code>add</code>	Add file contents to the index
<code>mv</code>	Move or rename a file, a directory, or a symlink
<code>restore</code>	Restore working tree files
<code>rm</code>	Remove files from the working tree and from the index

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

<code>bisect</code>	Use binary search to find the commit that introduced a bug
<code>diff</code>	Show changes between commits, commit and working tree, etc
<code>grep</code>	Print lines matching a pattern
<code>log</code>	Show commit logs
<code>show</code>	Show various types of objects
<code>status</code>	Show the working tree status

grow, mark and tweak your common history

<code>branch</code>	List, create, or delete branches
<code>commit</code>	Record changes to the repository
<code>merge</code>	Join two or more development histories together
<code>rebase</code>	Reapply commits on top of another base tip

```
apt-get update
apt-get install (--user) git
```


Git: essentials and how-to

```
(base) MacBook-Pro-2:~ milenavalentini$
(base) MacBook-Pro-2:~ milenavalentini$ cd Desktop/
(base) MacBook-Pro-2:Desktop milenavalentini$ mkdir Git_test_folder
(base) MacBook-Pro-2:Desktop milenavalentini$ cd Git_test_folder/
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ touch file_1.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ touch file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ touch file_3.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls -la
total 0
drwxr-xr-x   5 milenavalentini  staff   160 Oct  1 22:00 .
drwx-----@ 50 milenavalentini  staff  1600 Oct  1 21:59 ..
-rw-r--r--   1 milenavalentini  staff    0 Oct  1 21:59 file_1.txt
-rw-r--r--   1 milenavalentini  staff    0 Oct  1 21:59 file_2.txt
-rw-r--r--   1 milenavalentini  staff    0 Oct  1 22:00 file_3.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ cd ..
(base) MacBook-Pro-2:Desktop milenavalentini$ git init Git_test_folder/
Initialized empty Git repository in /Users/milenavalentini/Desktop/Git_test_folder/.git/
(base) MacBook-Pro-2:Desktop milenavalentini$
(base) MacBook-Pro-2:Desktop milenavalentini$ cd Git_test_folder/
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls -la
total 0
drwxr-xr-x   6 milenavalentini  staff   192 Oct  1 22:00 .
drwx-----@ 50 milenavalentini  staff  1600 Oct  1 21:59 ..
drwxr-xr-x   9 milenavalentini  staff   288 Oct  1 22:00 .git
-rw-r--r--   1 milenavalentini  staff    0 Oct  1 21:59 file_1.txt
-rw-r--r--   1 milenavalentini  staff    0 Oct  1 21:59 file_2.txt
-rw-r--r--   1 milenavalentini  staff    0 Oct  1 22:00 file_3.txt
```


Git: essentials and how-to

Initial configuration

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git config --global user.name "Milena Valentini"  
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git config --global user.email valentini@usm.lmu.de  
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```


Git: essentials and how-to

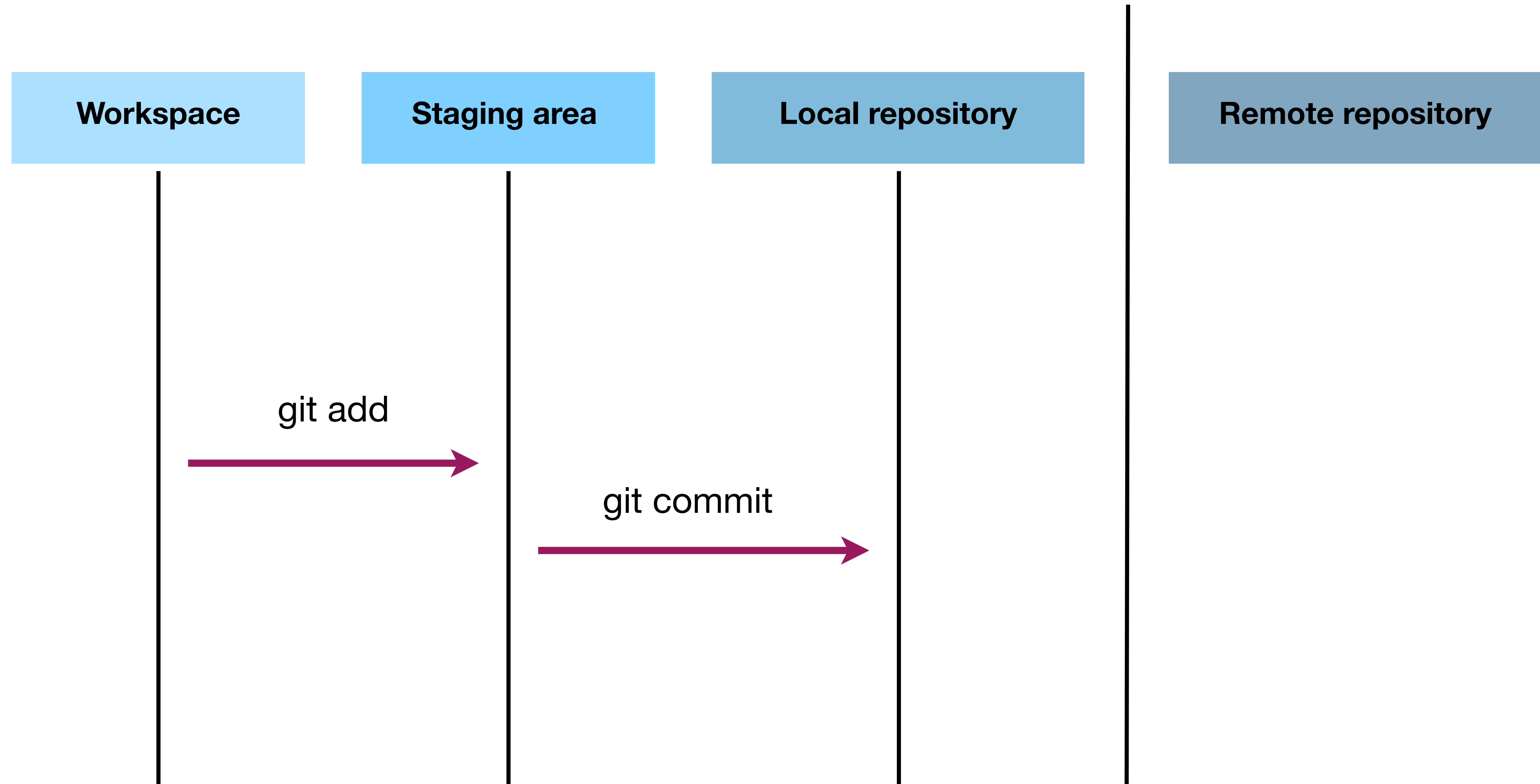
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file_1.txt
    file_2.txt
    file_3.txt

nothing added to commit but untracked files present (use "git add" to track)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ cd ..
(base) MacBook-Pro-2:Desktop milenavalentini$ mkdir Git_test_folder_noInit
(base) MacBook-Pro-2:Desktop milenavalentini$ cd Git_test_folder_noInit/
(base) MacBook-Pro-2:Git_test_folder_noInit milenavalentini$ git status
fatal: not a git repository (or any of the parent directories): .git
(base) MacBook-Pro-2:Git_test_folder_noInit milenavalentini$
```

Git: essentials and how-to



List of Git commands: <https://git-scm.com/docs>

Git: essentials and how-to

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
   file_1.txt
   file_2.txt
   file_3.txt

nothing added to commit but untracked files present (use "git add" to track)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

git status displays an overview of the workspace and the current snapshot (only) of the staging area

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add .
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
   new file:   file_1.txt
   new file:   file_2.txt
   new file:   file_3.txt

(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

git add to “copy” all the files in the folder from the workspace to the staging area (otherwise, specify the filename to be added)

Git: essentials and how-to

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ touch file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt      file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file_1.txt
        new file:   file_2.txt
        new file:   file_3.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file_4.txt

(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git status
On branch main

No commits yet

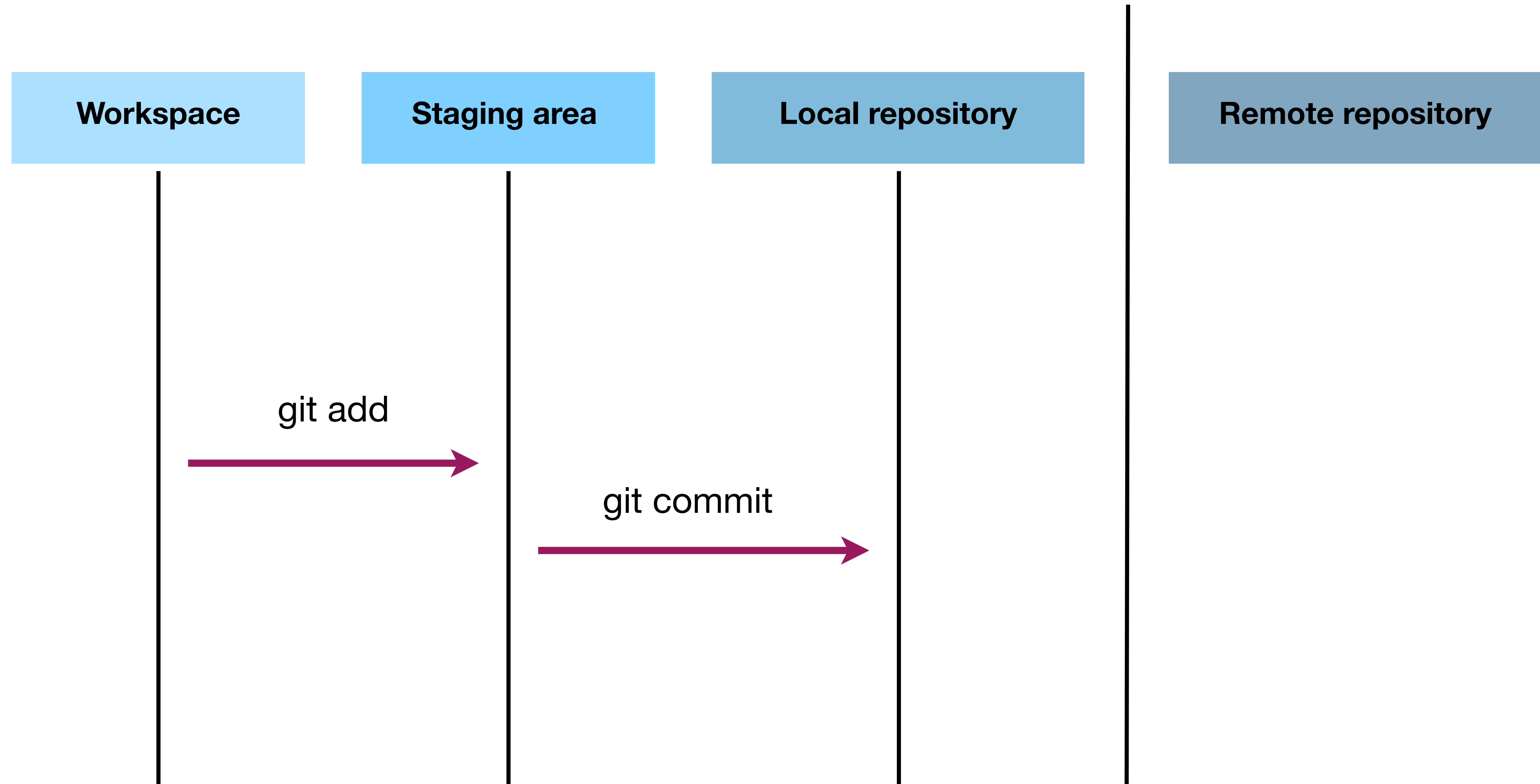
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file_1.txt
        new file:   file_2.txt
        new file:   file_3.txt
        new file:   file_4.txt

(base) MacBook-Pro-2:Git_test_folder milenavalentini$ _
```

git status displays an overview of the workspace and the current snapshot (only) of the staging area

git add to “copy” all the files in the folder from the workspace to the staging area (otherwise, specify the filename to be added)

Git: essentials and how-to



List of Git commands: <https://git-scm.com/docs>

Git: essentials and how-to

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'new files created'
[main (root-commit) 7d88be2] new files created
4 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file_1.txt
create mode 100644 file_2.txt
create mode 100644 file_3.txt
create mode 100644 file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'edited content of file_4.txt'
[main 544b8c4] edited content of file_4.txt
1 file changed, 1 insertion(+)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'inserted text in file_2.txt'
[main a179a5d] inserted text in file_2.txt
1 file changed, 1 insertion(+)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

git commit
copies files
from
the staging area
to
the local repository

Workflow:
edit
add
commit
(and repeat)

Git: essentials and how-to

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git log
commit a179a5d8af8d99fea993a1c5c485c68067132f9b (HEAD -> main)
Author: Milena Valentini <valentini@usm.lmu.de>
Date: Sun Oct 1 23:20:48 2023 +0200

    inserted text in file_2.txt

commit 544b8c428fdc6ed8e280dd965c38b585a495bce1
Author: Milena Valentini <milenavalentini@MacBook-Pro-2.local>
Date: Sun Oct 1 23:19:02 2023 +0200

    edited content of file_4.txt

commit 7d88be2f81fd4d8a2705d1b88d131e770d4e7fa2
Author: Milena Valentini <milenavalentini@MacBook-Pro-2.local>
Date: Sun Oct 1 23:15:15 2023 +0200

    new files created
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

git log
to output
changes with time
(most recent files on top)

Git: essentials and how-to

```
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git log
commit a179a5d8af8d99fea993a1c5c485c68067132f9b (HEAD -> main)
Author: Milena Valentini <valentini@usm.lmu.de>
Date: Sun Oct 1 23:20:48 2023 +0200

    inserted text in file_2.txt

commit 544b8c428fdc6ed8e280dd965c38b585a495bce1
Author: Milena Valentini <milenaValentini@MacBook-Pro-2.local>
Date: Sun Oct 1 23:19:02 2023 +0200

    edited content of file_4.txt

commit 7d88be2f81fd4d8a2705d1b88d131e770d4e7fa2
Author: Milena Valentini <milenaValentini@MacBook-Pro-2.local>
Date: Sun Oct 1 23:15:15 2023 +0200

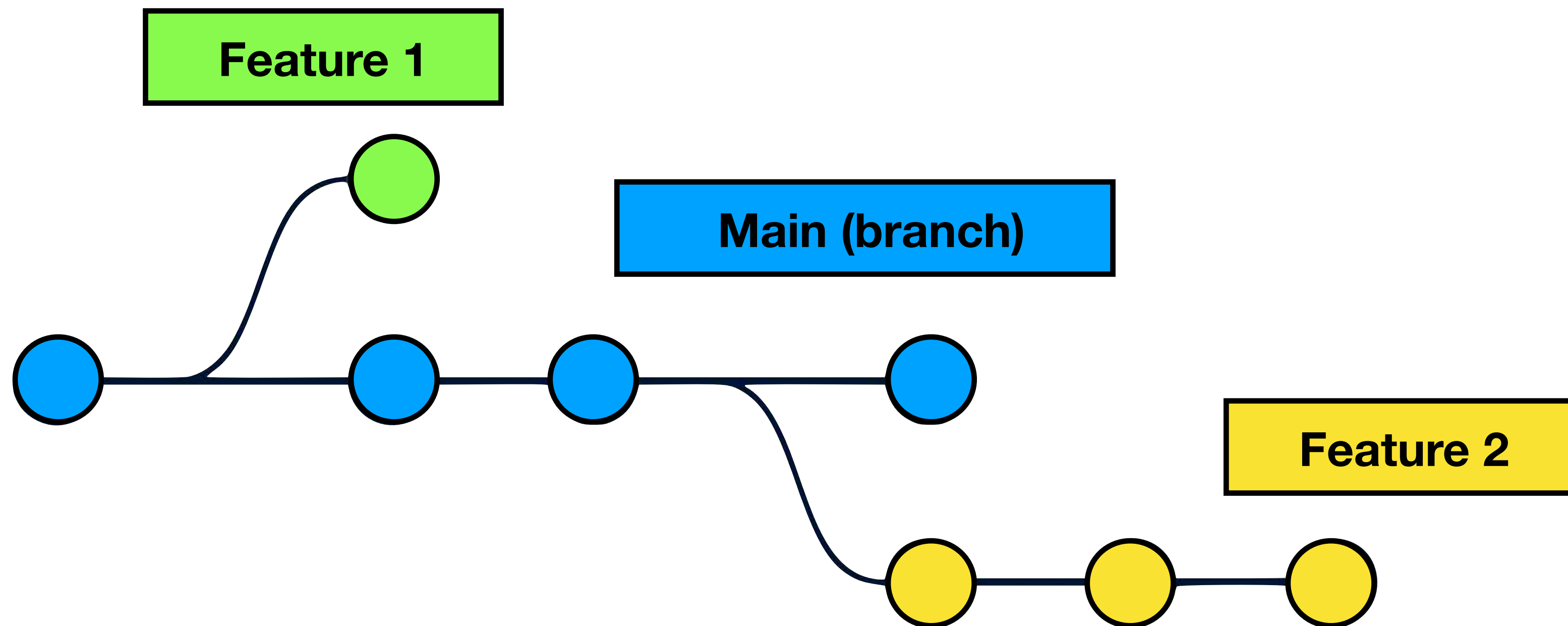
    new files created
(base) MacBook-Pro-2:Git_test_folder milenaValentini$
```

git log
to output
changes with time
(most recent files on top)

```
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git status
On branch main
nothing to commit, working tree clean
```

git status

Git: branches



Branches store different versions of your project

They are pointers to a commit

The main (or master) branch always exists and it is created during the initialization

Key for parallel development

Git: branches

List all existing local branches

```
(base) MacBook-Pro-2:Git_test_folder milenaValentini$  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git branch  
* main  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git branch feature_A  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git branch feature_B  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git branch  
feature_A  
feature_B  
* main  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git branch -d feature_B  
Deleted branch feature_B (was a179a5d).  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ git branch  
feature_A  
* main  
(base) MacBook-Pro-2:Git_test_folder milenaValentini$ _
```

git branch

git branch feature_A

git branch feature_B

git branch

git branch -d feature_B

git branch

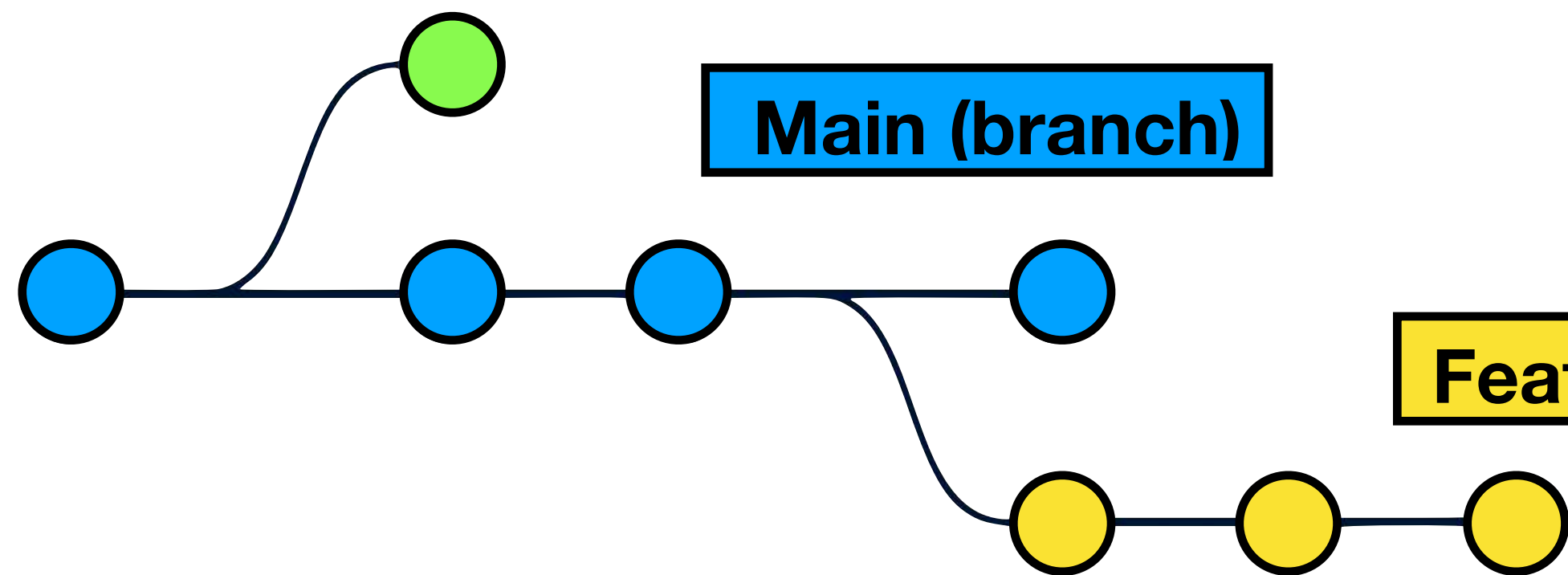
Create a new branch

Delete a branch

Feature 1

Main (branch)

Feature 2

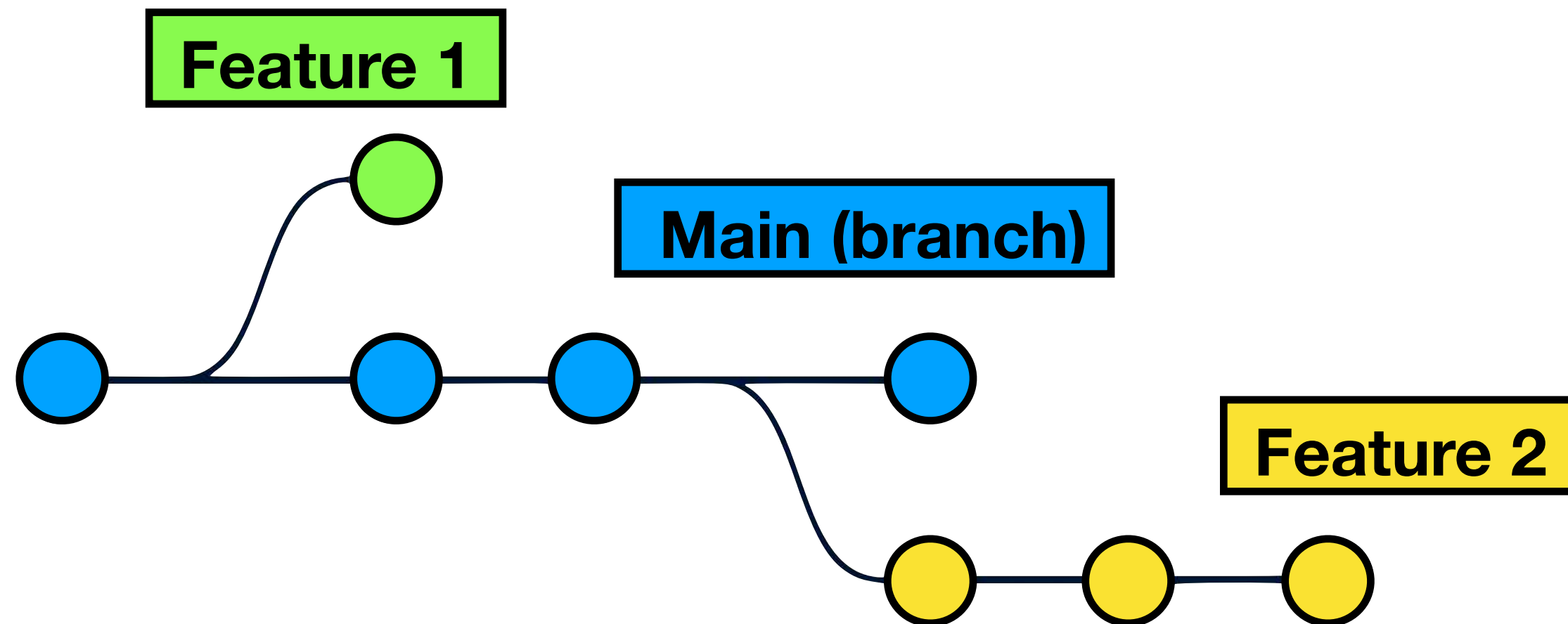


Git: branches

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
feature_A
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_A
Switched to branch 'feature_A'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
* feature_A
main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ _
```

Switch
to
another
branch

git branch operates on the local repository

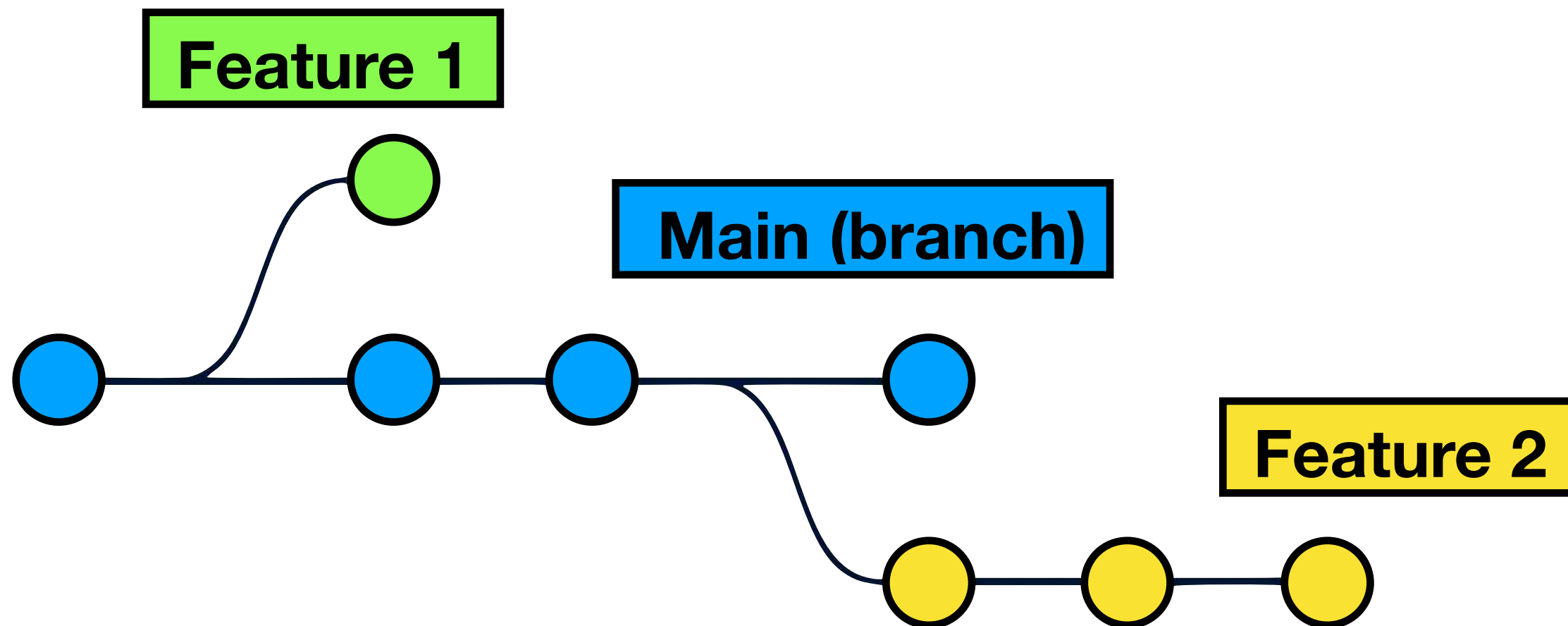


Git: branches

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
feature_A
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_A
Switched to branch 'feature_A'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt      file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified the content of file_2.txt'
[feature_A a7d509c] modified the content of file_2.txt
1 file changed, 1 insertion(+)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

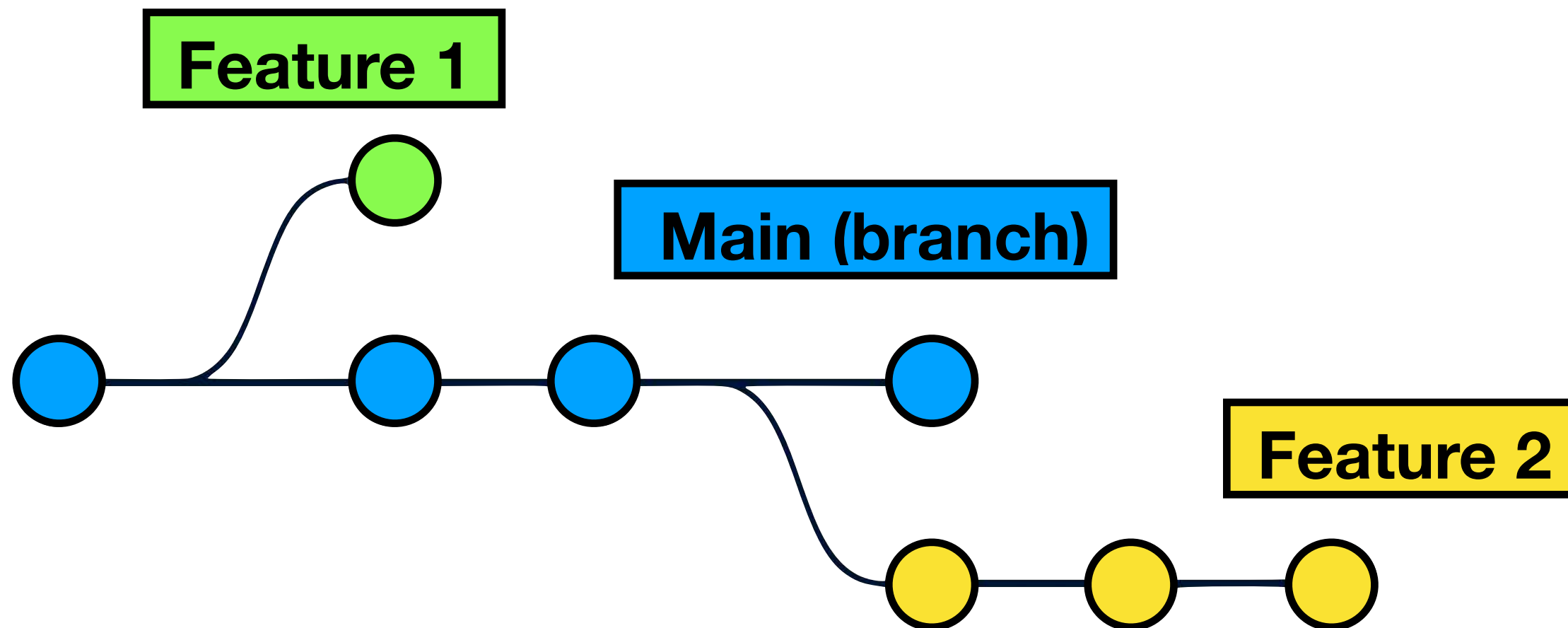
Edit a file
in a branch

commit to that branch



Git: branches

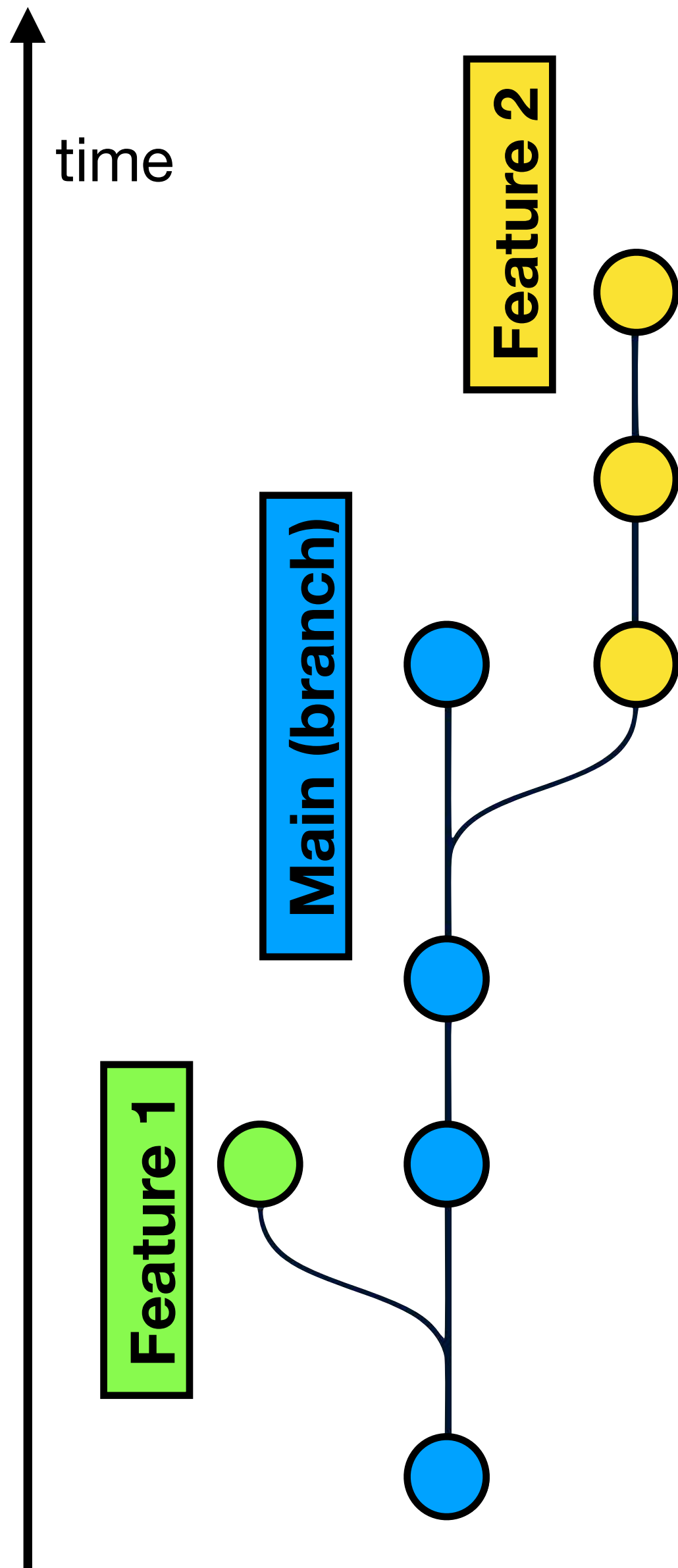
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
* feature_A
  main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch main
Switched to branch 'main'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git merge feature_A
Updating 331b46f..a7d509c
Fast-forward
 file_2.txt | 1 +
 1 file changed, 1 insertion(+)
```



Copies the content of the specified branch (i.e. feature_A) in the current branch (i.e. main)

Crucial to understand in which branch I am before making a merge

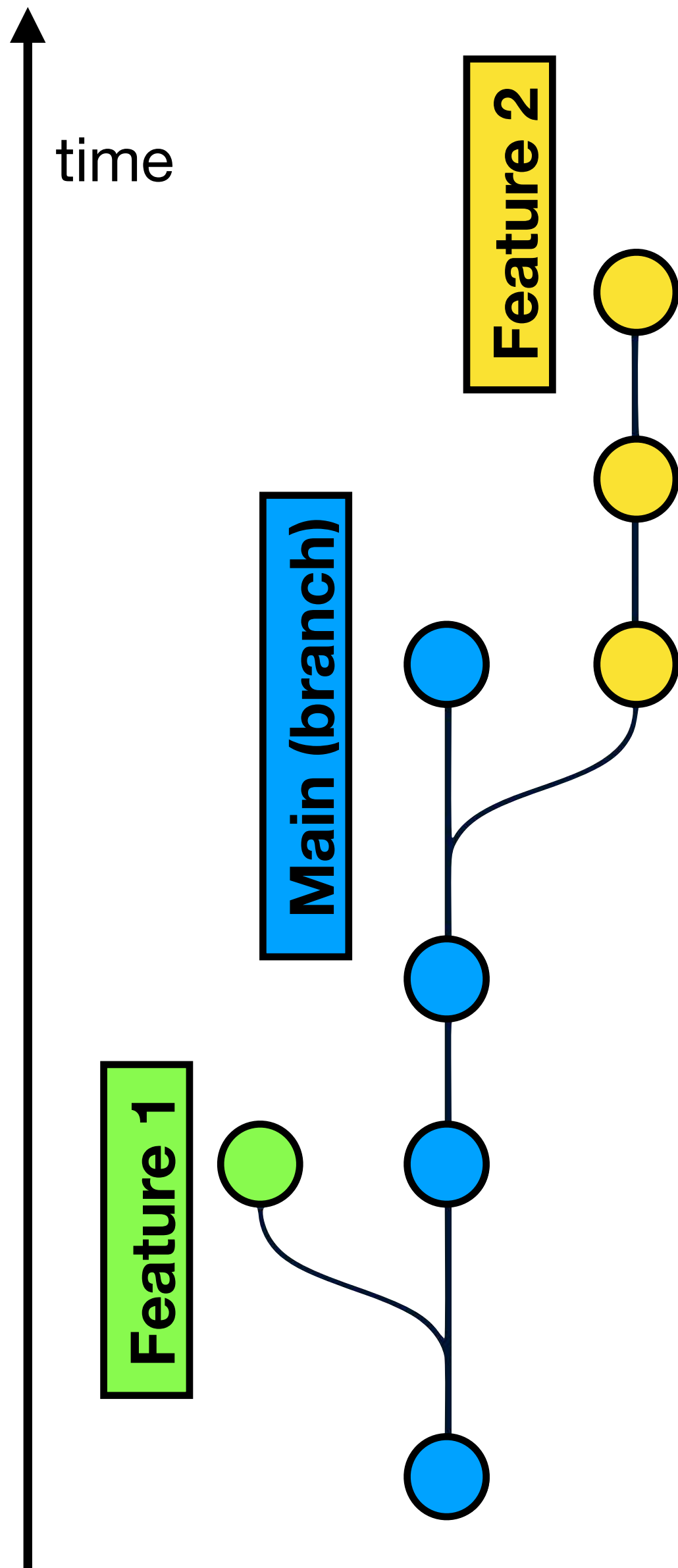
Git: branches



```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_C
Switched to branch 'feature_C'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
* feature_C
  main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt      file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on feature_C'
[feature_C a96367a] modified file_2 on feature_C
 1 file changed, 1 insertion(+)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch main
Switched to branch 'main'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on main'
[main 6cd7523] modified file_2 on main
 1 file changed, 1 deletion(-)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git merge feature_C
Auto-merging file_2.txt
CONFLICT (content): Merge conflict in file_2.txt
Automatic merge failed; fix conflicts and then commit the result.
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on main to resolve conflicts
with file_2 on feature_C'
[main 2a2dlca] modified file_2 on main to resolve conflicts with file_2 on feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

Edit file_2 in feature_C

Git: branches

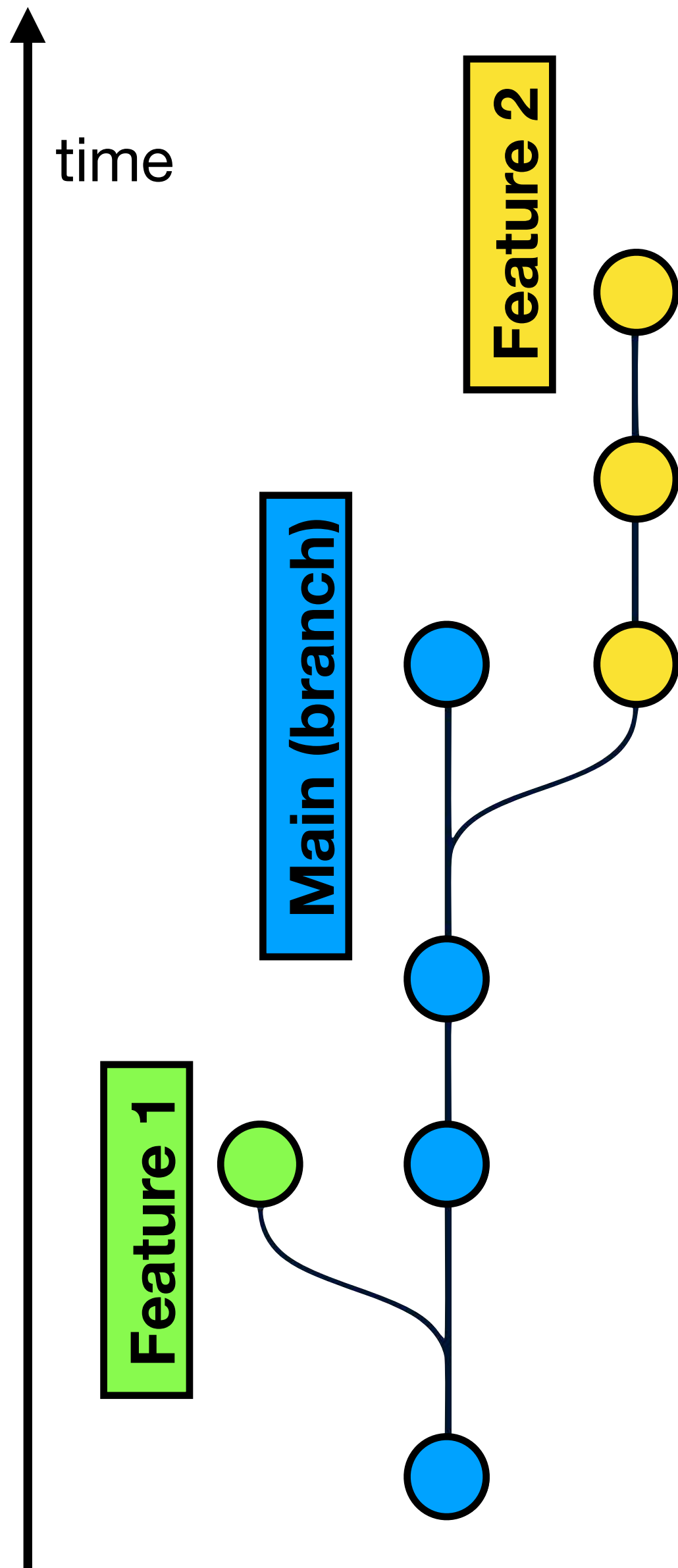


```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_C
Switched to branch 'feature_C'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
* feature_C
  main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt      file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on feature_C'
[feature_C a96367a] modified file_2 on feature_C
 1 file changed, 1 insertion(+)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch main
Switched to branch 'main'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on main'
[main 6cd7523] modified file_2 on main
 1 file changed, 1 deletion(-)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git merge feature_C
Auto-merging file_2.txt
CONFLICT (content): Merge conflict in file_2.txt
Automatic merge failed; fix conflicts and then commit the result.
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on main to resolve conflicts
with file_2 on feature_C'
[main 2a2dlca] modified file_2 on main to resolve conflicts with file_2 on feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

Edit file_2 in feature_C

Edit the same in main

Git: branches



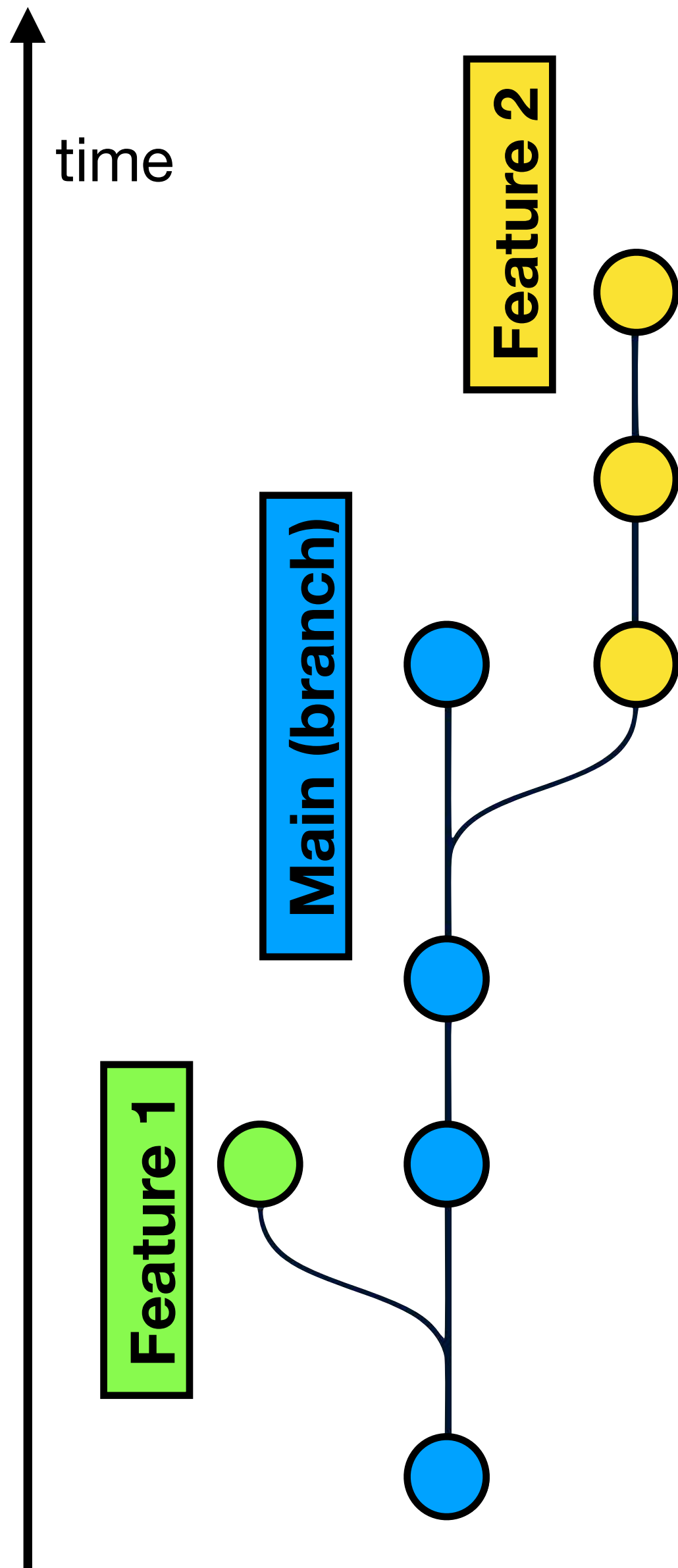
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_C
Switched to branch 'feature_C'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
* feature_C
  main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt      file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on feature_C'
[feature_C a96367a] modified file_2 on feature_C
 1 file changed, 1 insertion(+)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch main
Switched to branch 'main'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on main'
[main 6cd7523] modified file_2 on main
 1 file changed, 1 deletion(-)
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git merge feature_C
Auto-merging file_2.txt
CONFLICT (content): Merge conflict in file_2.txt
Automatic merge failed; fix conflicts and then commit the result.
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ vi file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git add file_2.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'modified file_2 on main to resolve conflicts
with file_2 on feature_C'
[main 2a2dlca] modified file_2 on main to resolve conflicts with file_2 on feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

Edit file_2 in feature_C

Edit the same in main

```
42
84
<<<<<<< HEAD
=====
168
336
>>>>>> feature_C
```


Git: branches



```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git log
commit 2a2dlca55771afba839a9e544f6f7286760f470a (HEAD -> main)
Merge: 6cd7523 a96367a
Author: Milena Valentini <valentini@usm.lmu.de>
Date: Mon Oct 2 02:39:50 2023 +0200

    modified file_2 on main to resolve conflicts with file_2 on feature_C

commit 6cd7523db0a8c997f01641acc8057cb76d97cc3d
Author: Milena Valentini <valentini@usm.lmu.de>
Date: Mon Oct 2 02:37:48 2023 +0200

    modified file_2 on main

commit a96367ace141b3030cea96da6cd1fba293e65e4c (feature_C)
Author: Milena Valentini <valentini@usm.lmu.de>
Date: Mon Oct 2 02:37:10 2023 +0200

    modified file_2 on feature_C

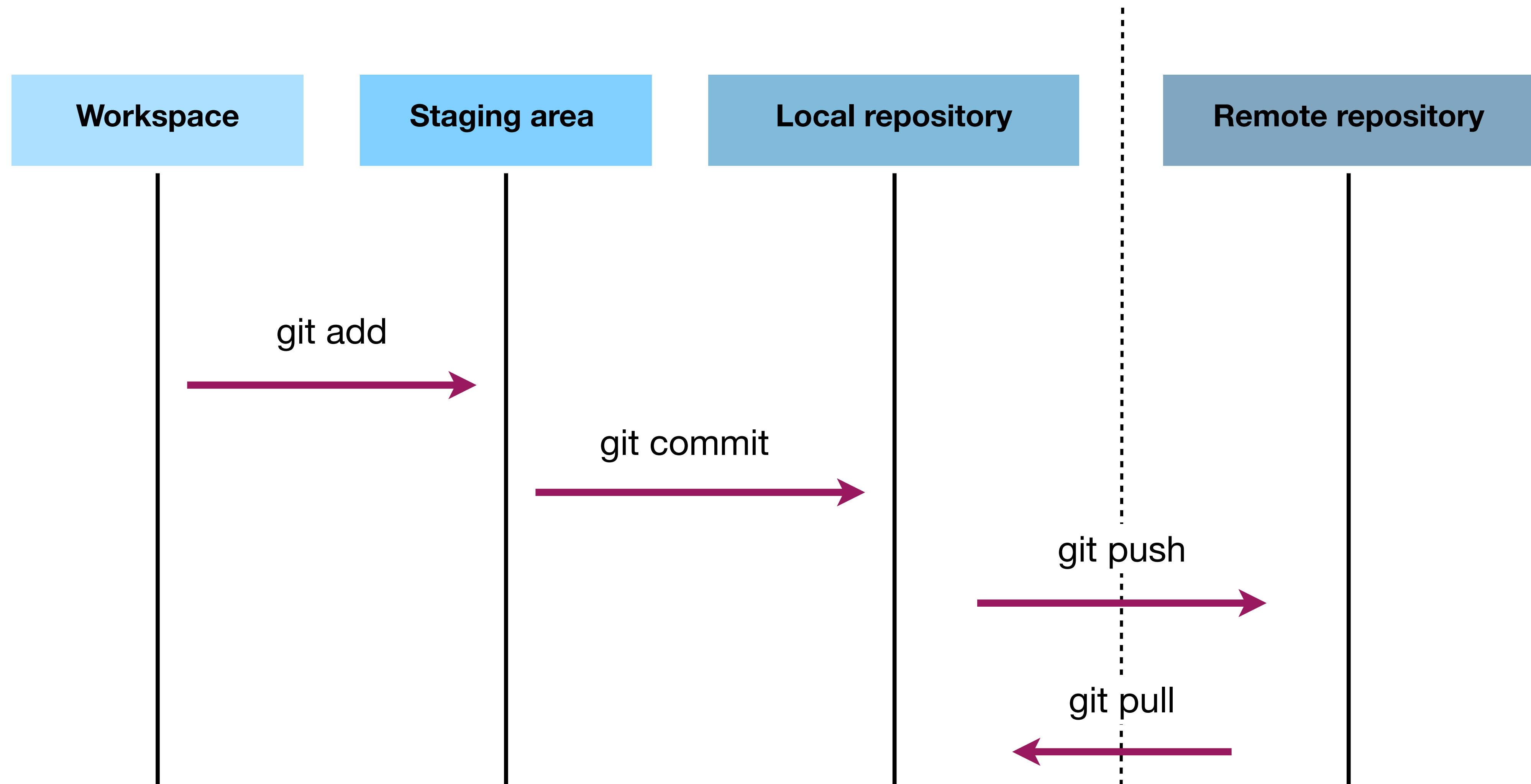
commit de0da59bea5d023e78f6fcdcf59273ce982f36677
Author: Milena Valentini <valentini@usm.lmu.de>
Date: Mon Oct 2 02:26:19 2023 +0200

    modified file_4.txt by adding a second line

commit a7d509c24a356ebc3fe5996d2f8745843bb8b60a (feature_A)
Author: Milena Valentini <valentini@usm.lmu.de>
Date: Mon Oct 2 02:15:28 2023 +0200

    modified the content of file_2.txt
```


Git: from local to remote repository



Git: from local to remote repository

Local repository

Remote repository

Create a new, empty repository on Git

git push

git pull

Git: from local to remote repository

New repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * / Repository name *

Testing_Git_TRMD_23 is available.

Great repository names are short and memorable. Need inspiration? How about [friendly-train](#) ?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

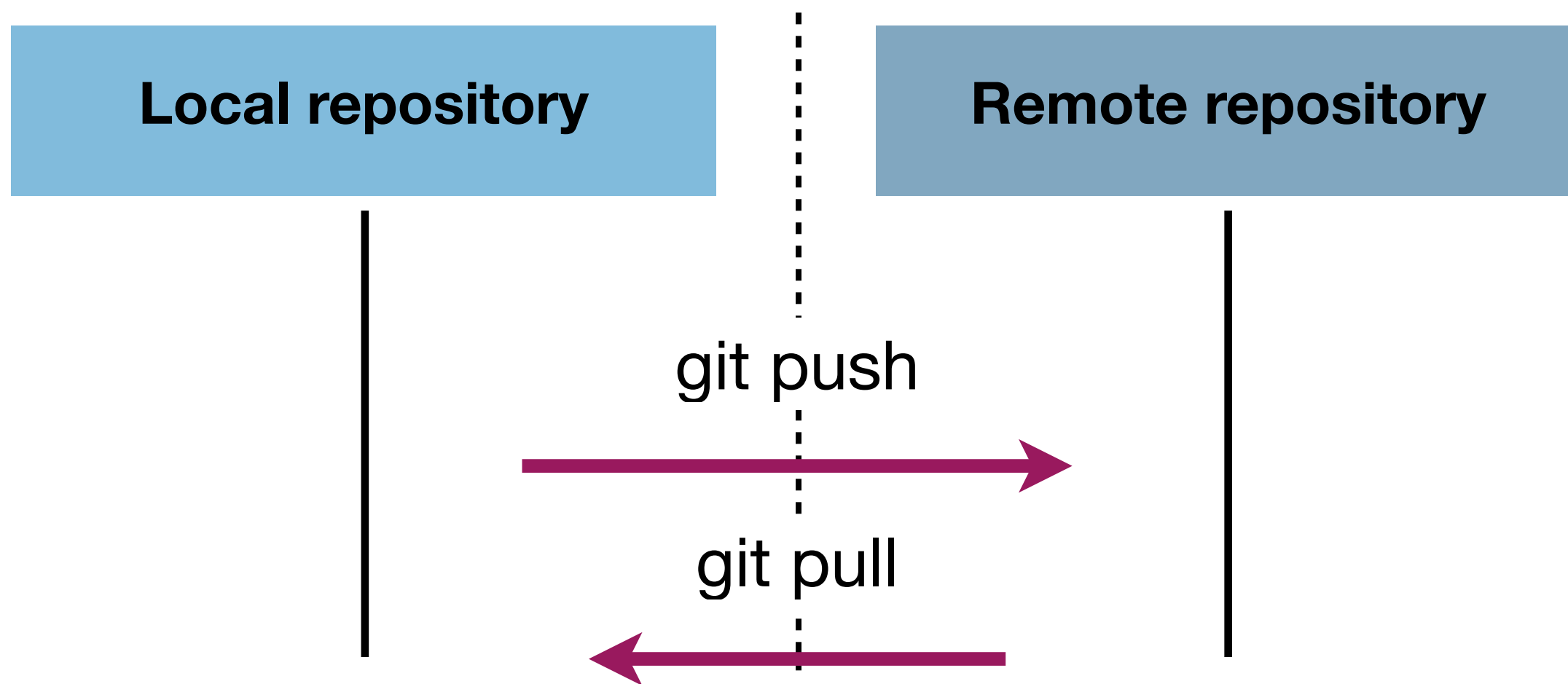
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

You are creating a public repository in your personal account.

Git: from local to remote repository



Create a new, empty repository on Git

In the main branch in the local repository:

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch -M main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git push -u origin main
Username for 'https://github.com': valentini
Password for 'https://valentini@github.com':
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 8 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (24/24), 1.90 KiB | 1.90 MiB/s, done.
Total 24 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/MilenaValentini/Testing_Git_TRMD_23
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```


Git: from local to remote repository

The screenshot shows the GitHub interface for a repository named 'Testing_Git_TRMD_23' by user 'MilenaValentini'. The repository is public. The page includes navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the repository name, there are buttons for Pin, Watch (0), Fork (0), and Star (0). Two main action boxes are present: 'Set up GitHub Copilot' and 'Add collaborators to this repository'. A 'Quick setup' section offers options for Desktop, HTTPS, or SSH, with the URL 'https://github.com/MilenaValentini/Testing_Git_TRMD_23.git'. Below this, two sections provide command-line instructions for creating a new repository or pushing an existing one. The second section is highlighted with a red border.

MilenaValentini / Testing_Git_TRMD_23

Testing_Git_TRMD_23 Public

Pin Watch 0 Fork 0 Star 0

Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot

Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/MilenaValentini/Testing_Git_TRMD_23.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

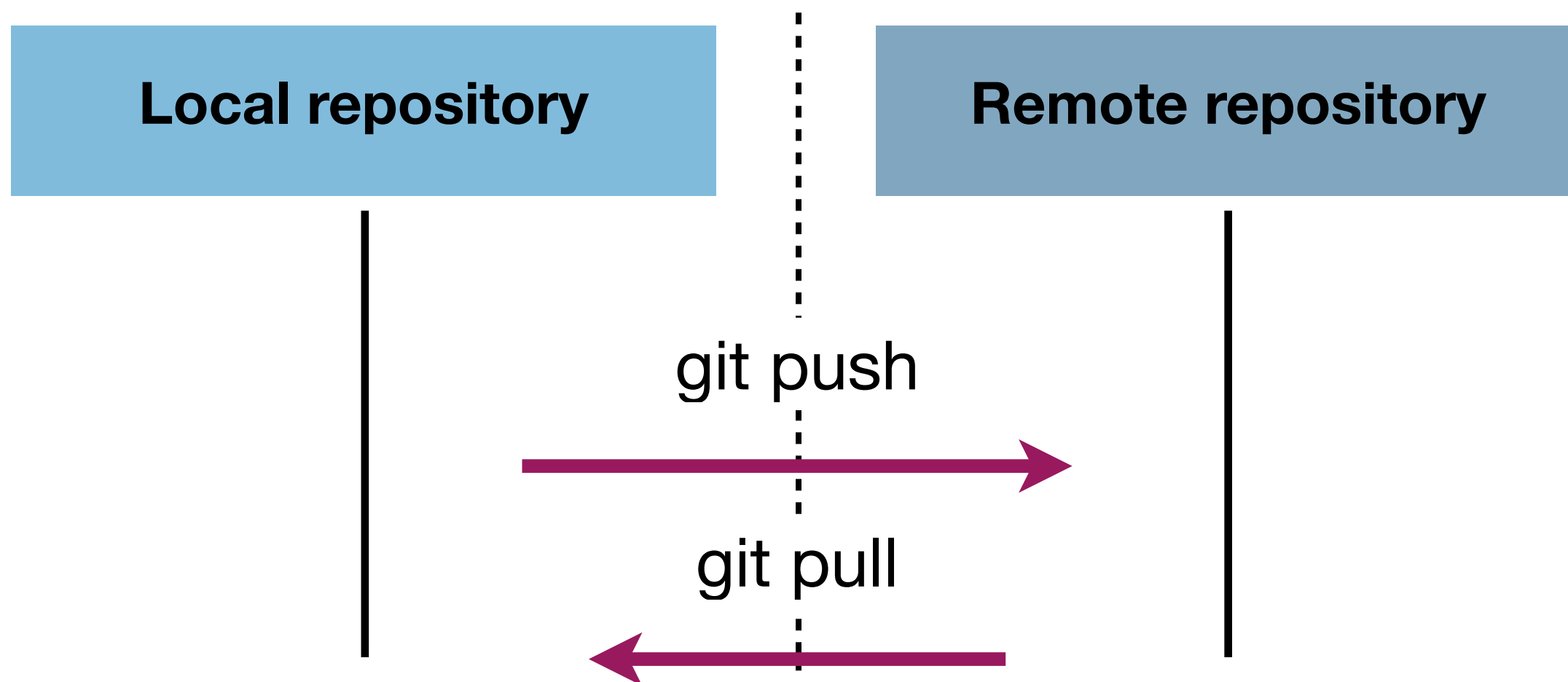
...or create a new repository on the command line

```
echo "# Testing_Git_TRMD_23" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23.git
git push -u origin main
```

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

```
git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23.git
git branch -M main
git push -u origin main
```


Git: from local to remote repository



Create a new, empty repository on Git

In the main branch in the local repository:

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch -M main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git push -u origin main
Username for 'https://github.com': valentini
Password for 'https://valentini@github.com':
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 8 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (24/24), 1.90 KiB | 1.90 M
Total 24 (delta 10), reused 0 (delta 0), pack-re
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/MilenaValentini/Testing_Gi
* [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(base) MacBook-Pro-2:Git_test_folder milenavalen
```

Testing_Git_TRMD_23 Public

Pin Unwatch 1 Fork 0 Star 0

main 1 branch 0 tags

Go to file Add file Code

About

No description, website, or topics provided.

Activity

0 stars

1 watching

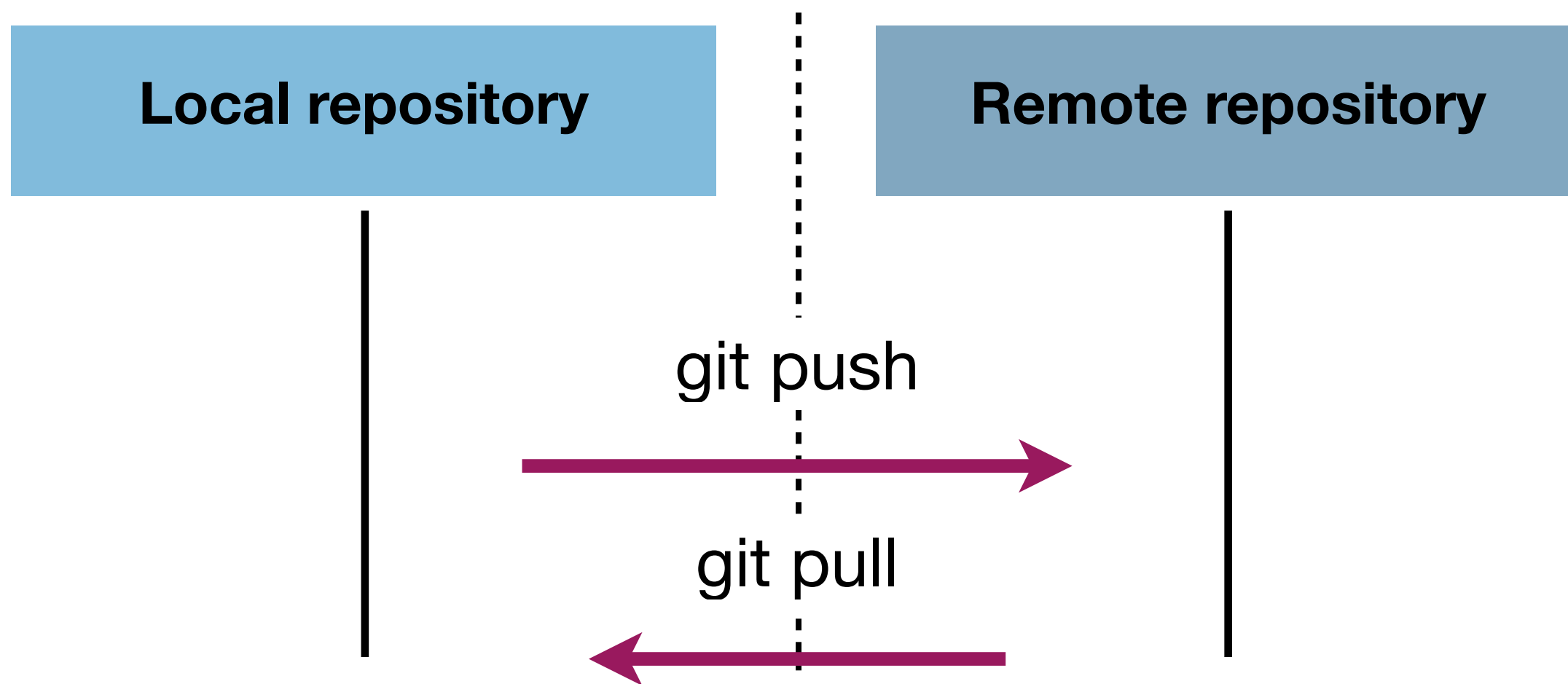
0 forks

Releases

No releases published

Create a new release

Git: from local to remote repository



Create a new, empty repository on Git

In the main branch in the local repository:

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch -M main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git push -u origin main
Username for 'https://github.com': valentini
Password for 'https://valentini@github.com':
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 8 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (24/24), 1.90 KiB | 1.90 MiB/s, done.
Total 24 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/MilenaValentini/Testing_Git_TRMD_23
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

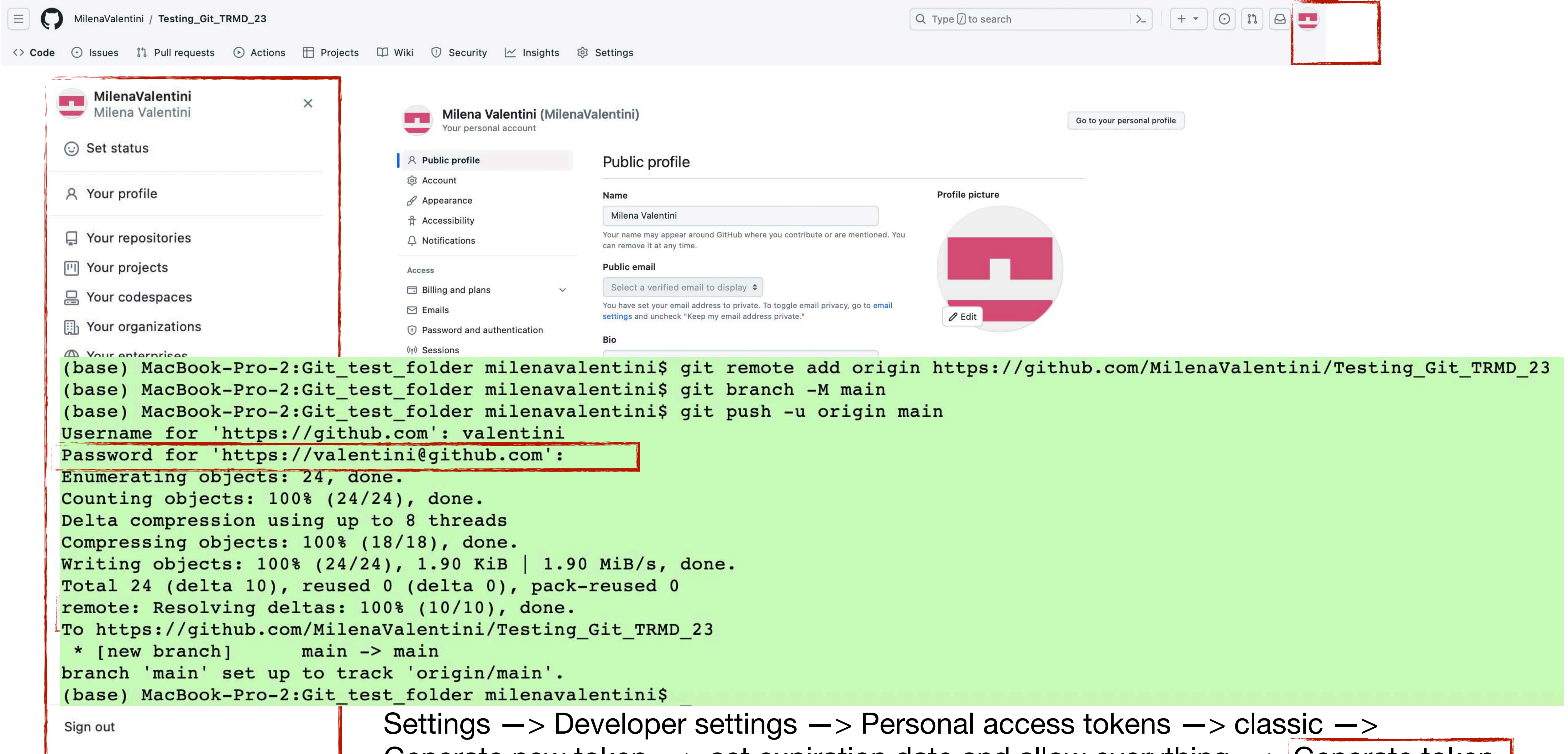

Git: from local to remote repository

The screenshot shows the GitHub user profile settings page for Milena Valentini. The page is divided into several sections:

- Left Sidebar:** A navigation menu with items like 'Set status', 'Your profile', 'Your repositories', 'Your projects', 'Your codespaces', 'Your organizations', 'Your enterprises', 'Your stars', 'Your sponsors', 'Your gists', 'Upgrade', 'Try Enterprise', 'Copilot', 'Feature preview', 'Settings' (highlighted with a red box), 'GitHub Docs', 'GitHub Support', and 'Sign out'.
- Top Bar:** The user's name 'Milena Valentini (MilenaValentini)' and a 'Go to your personal profile' button.
- Public profile:** A section for editing the public profile, including fields for Name, Public email, Bio, Pronouns, and Location. There is also a 'Display current local time' checkbox and an 'Update profile' button.
- Right Sidebar:** A section for 'GitHub Apps' and 'OAuth Apps', with a 'Personal access tokens' dropdown menu (highlighted with a red box).
- Developer settings:** A link to 'Developer settings' (highlighted with a red box) located below the main settings list.

Settings —> Developer settings —> Personal access tokens —> classic —> Generate new token —> set expiration date and allow everything —> Generate token

Git: from local to remote repository



The image shows a GitHub profile page for Milena Valentini. The left sidebar contains navigation options: Set status, Your profile, Your repositories, Your projects, Your codespaces, Your organizations, and Your enterprises. The main content area displays the 'Public profile' settings, including Name (Milena Valentini), Profile picture, Public email (set to private), and Bio. A terminal window is overlaid on the bottom half of the page, showing the following commands and output:

```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch -M main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git push -u origin main
Username for 'https://github.com': valentini
Password for 'https://valentini@github.com':
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 8 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (24/24), 1.90 KiB | 1.90 MiB/s, done.
Total 24 (delta 10), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (10/10), done.
To https://github.com/MilenaValentini/Testing_Git_TRMD_23
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

Below the terminal output, the navigation path for generating a personal access token is shown: Settings —> Developer settings —> Personal access tokens —> classic —> Generate new token —> set expiration date and allow everything —> Generate token.

Git: from local to remote repository

