TECNICHE DI RAPPRESENTAZIONE E MODELLIZZAZIONE DEI DATI

— Part 1 —

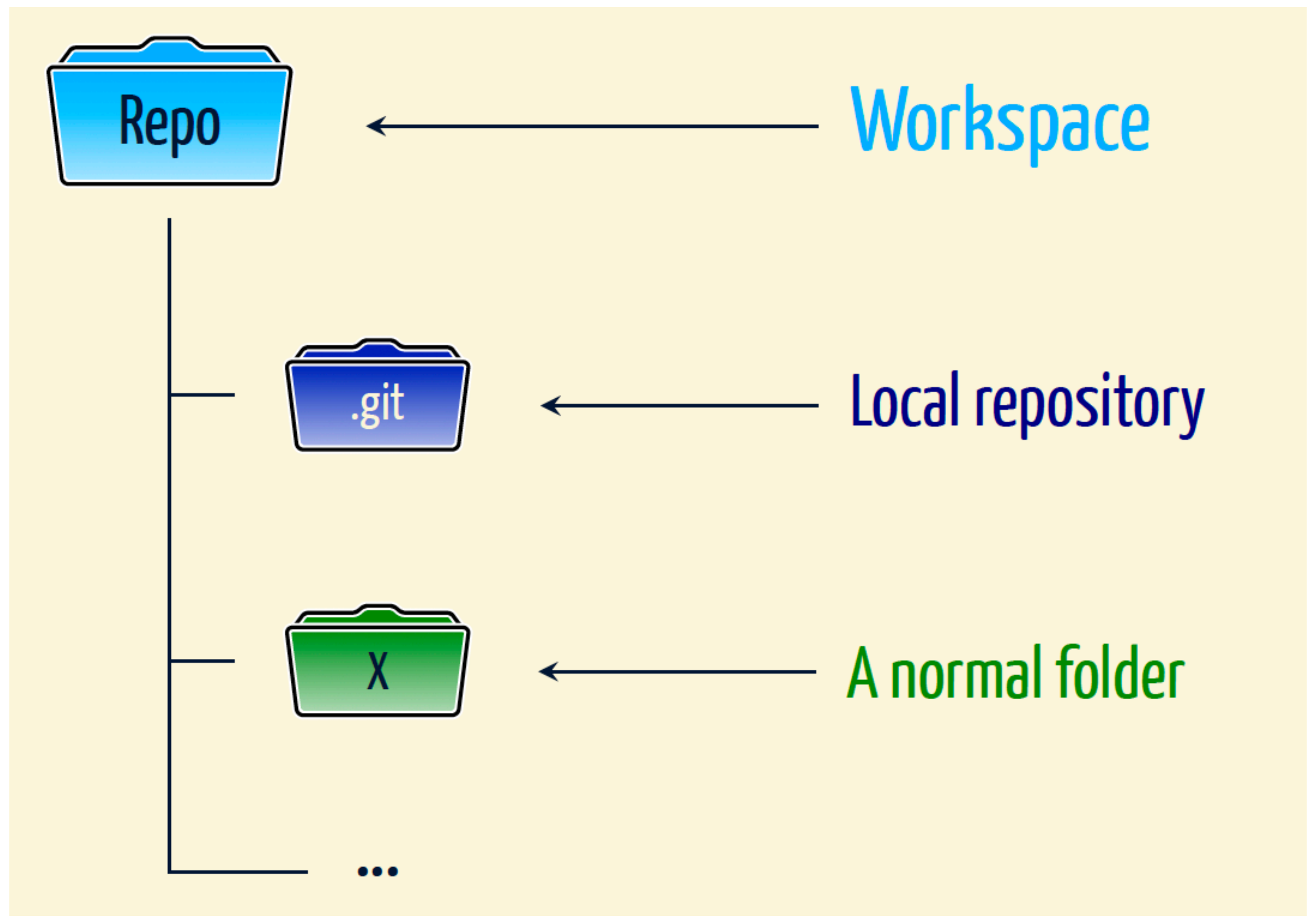
(2 CFU out of 6 total CFU)

Link moodle: https://moodle2.units.it/course/view.php?id=14486

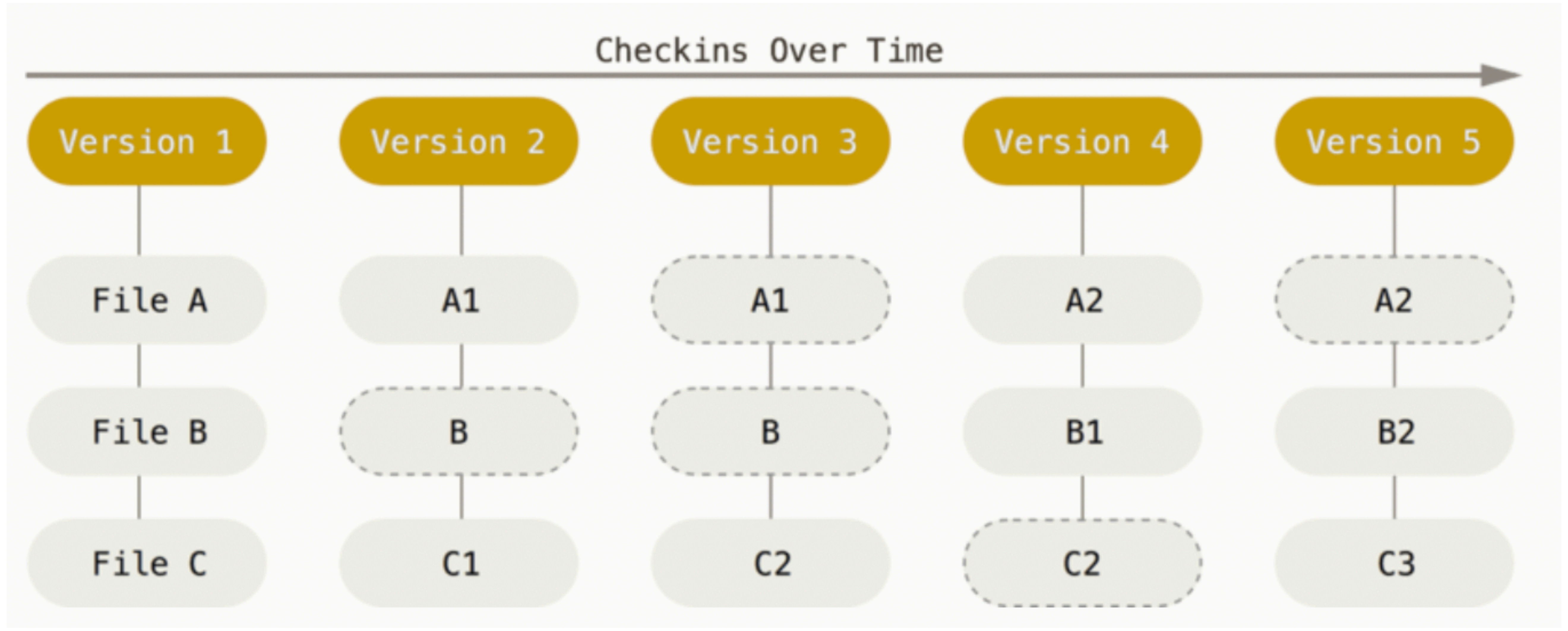
Codice Teams del corso: d2cmkh8

Git: Intro

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



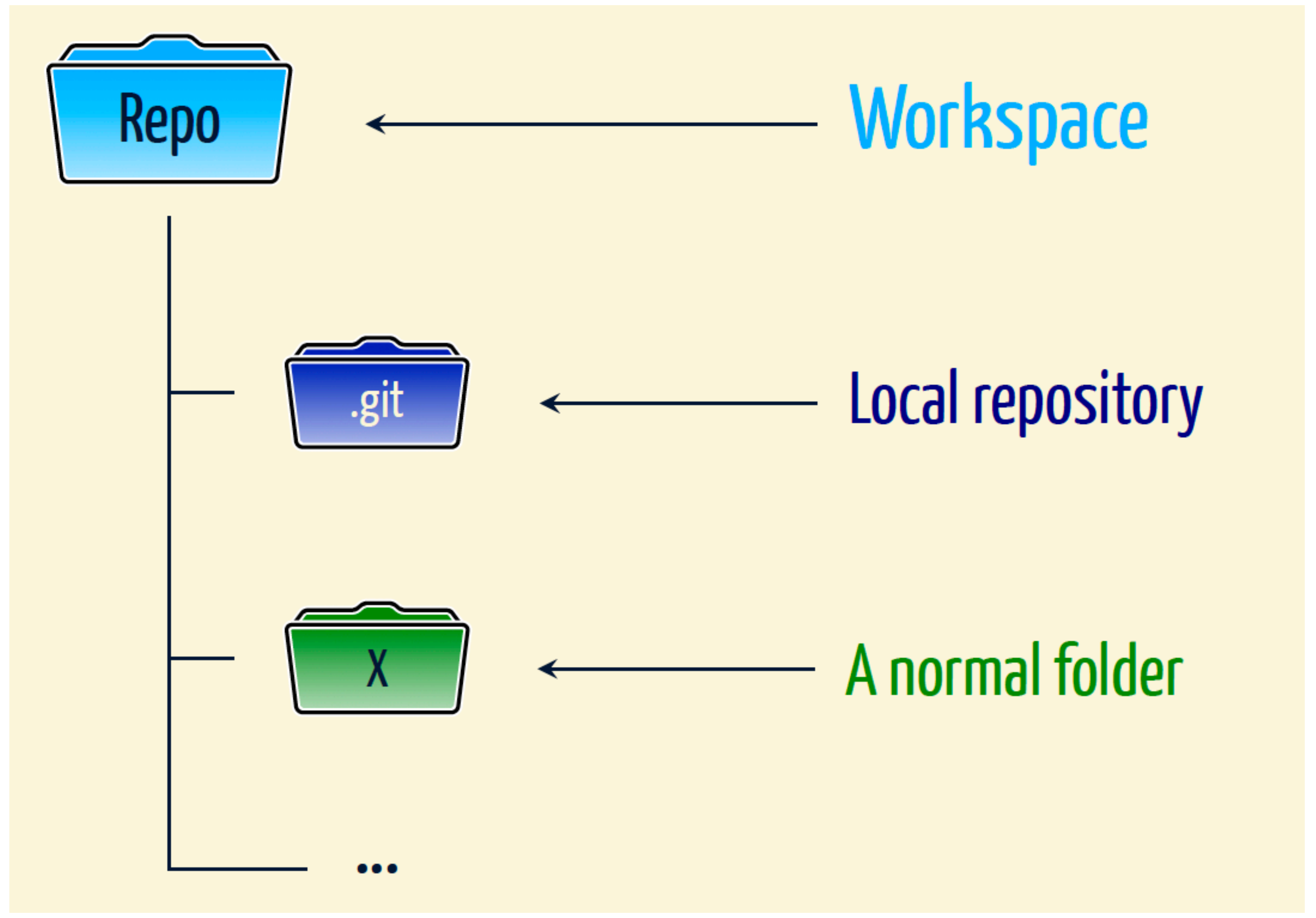
Records are based on snapshots taken over time

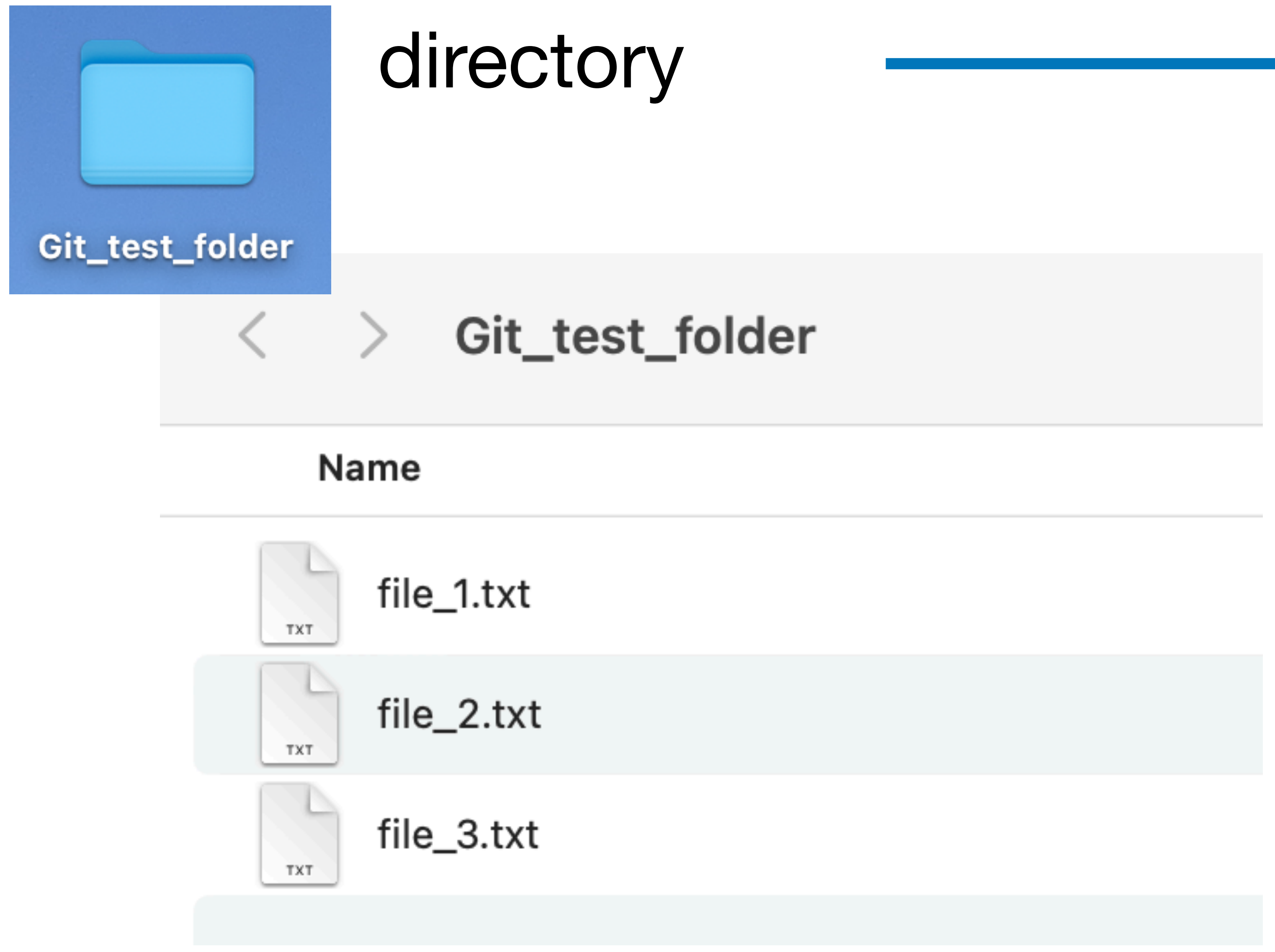


Snapshots are called commits and are referenced by a checksum

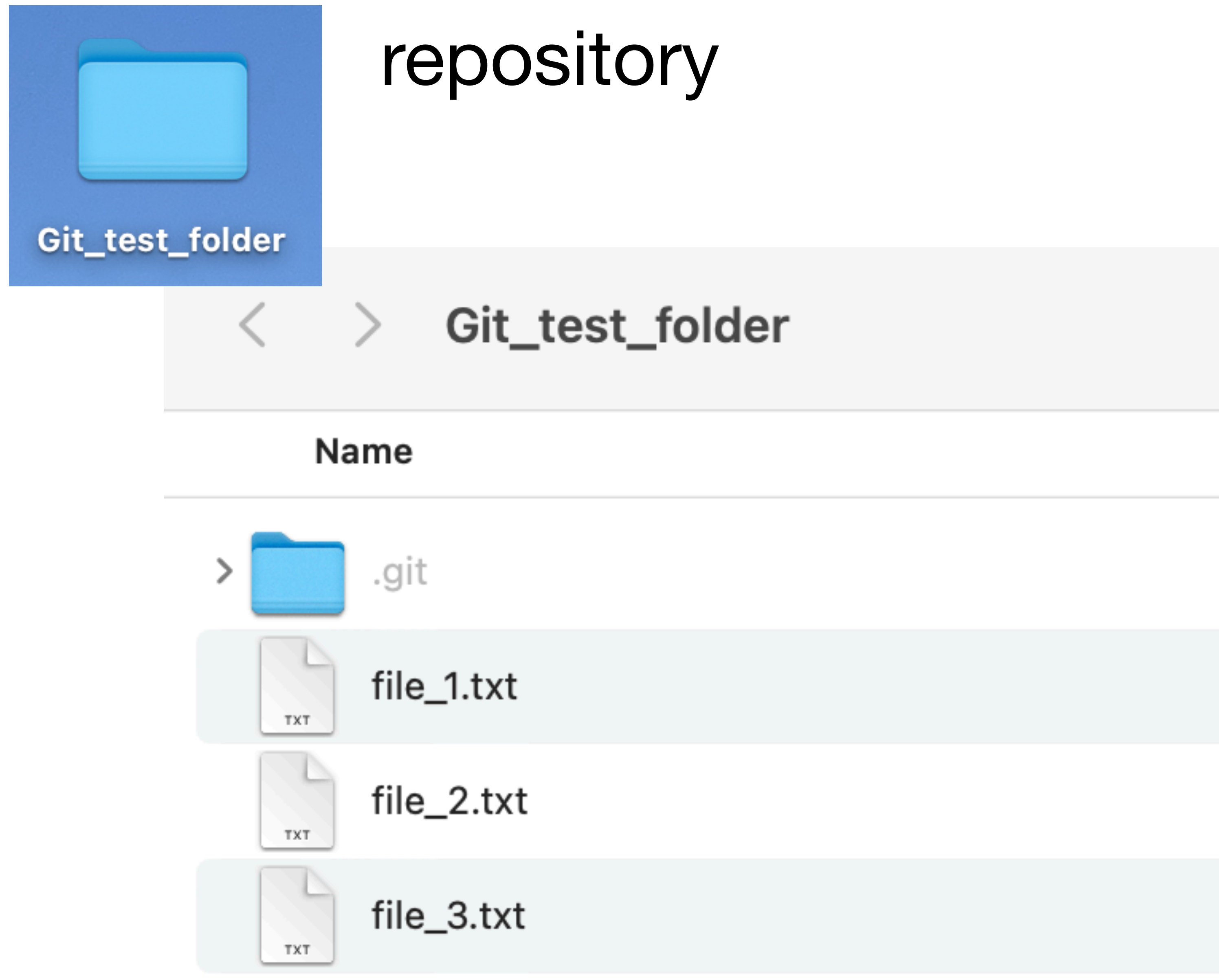
Git: Intro

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

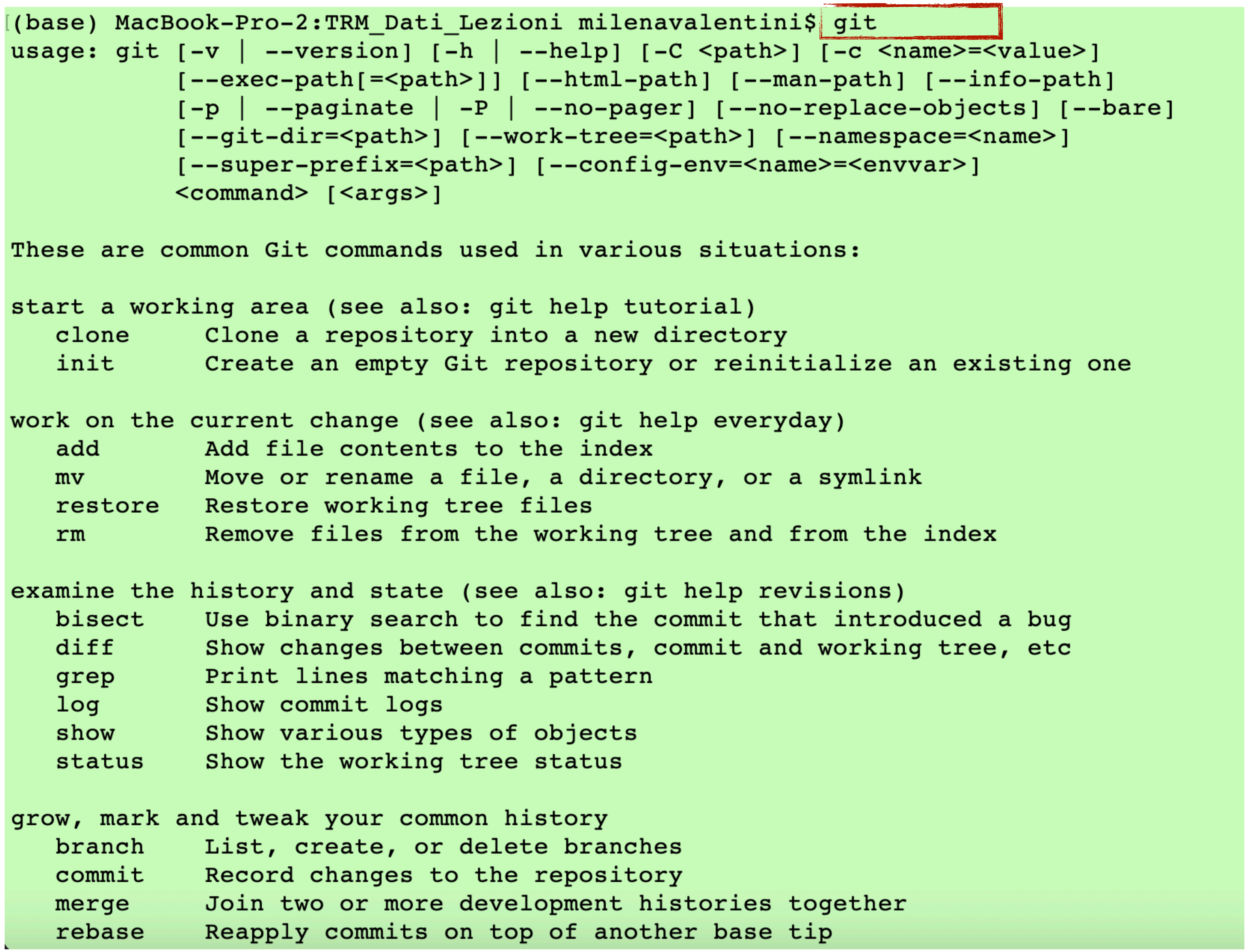




git init →



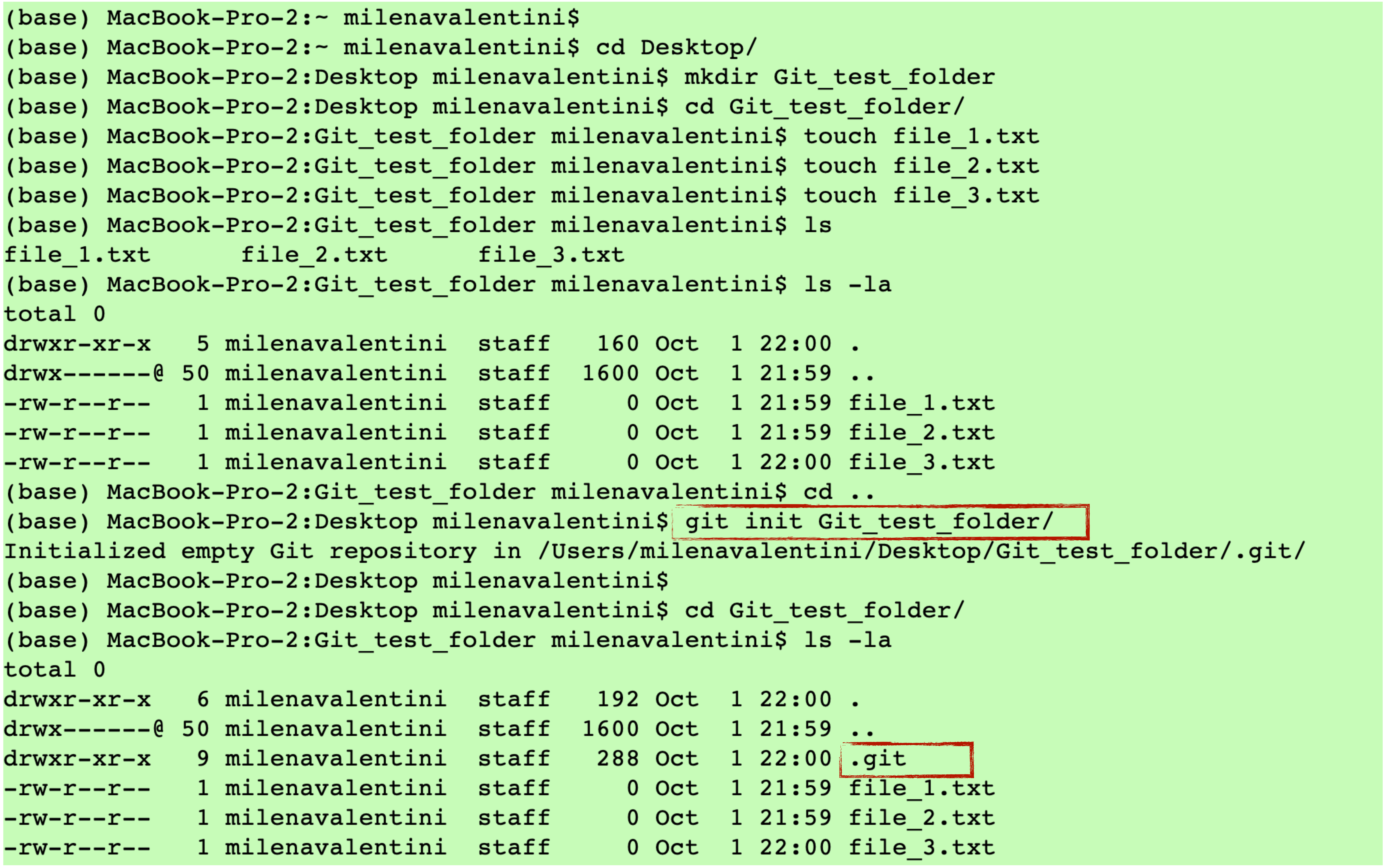
Git: essentials and how-to



```
(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)
   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
```

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

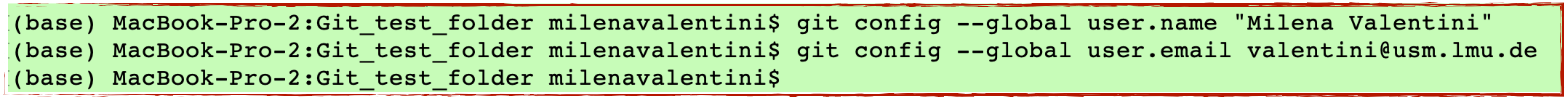
Git: essentials and how-to



```
(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
```

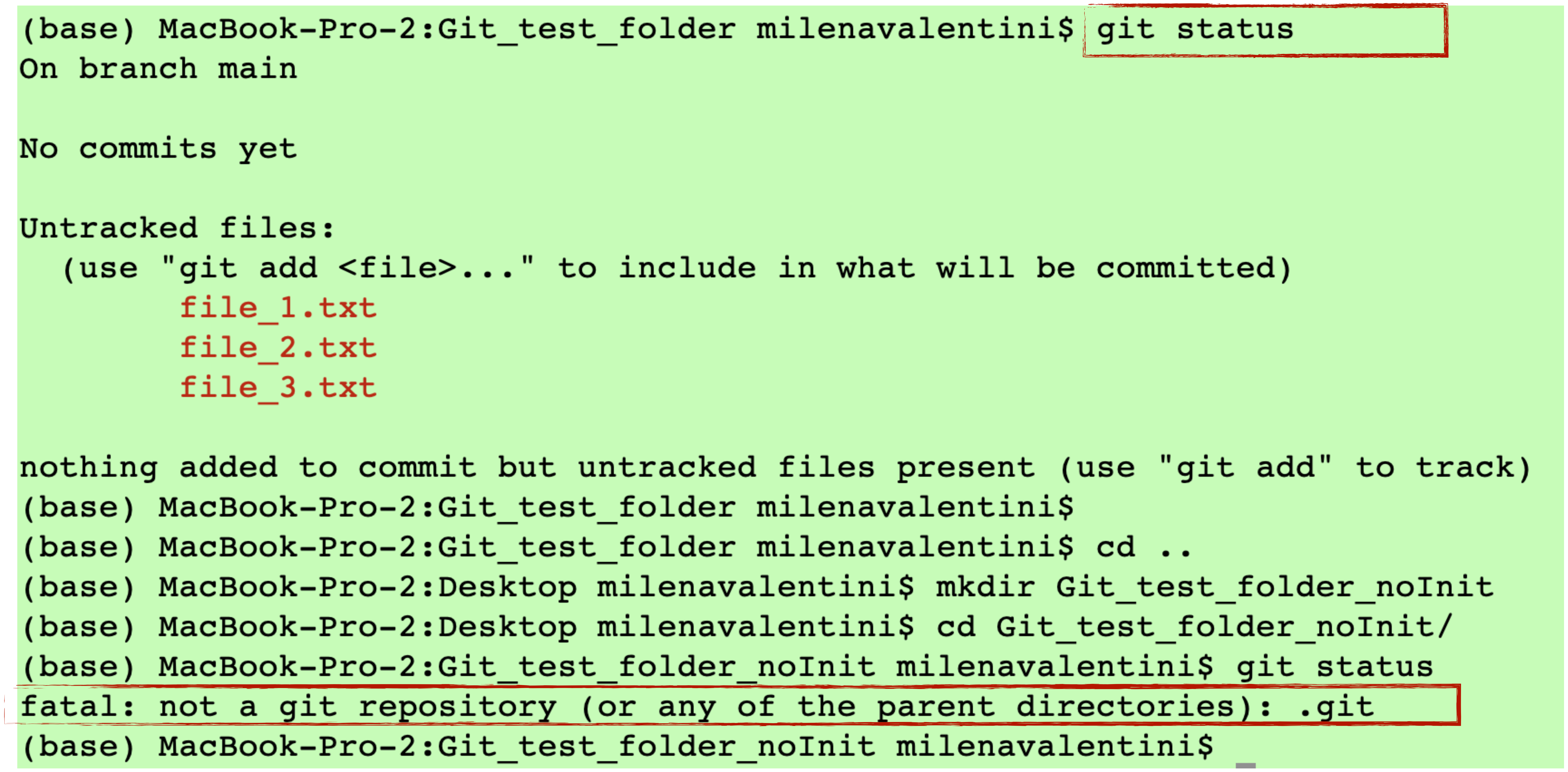
Git: essentials and how-to

Initial configuration



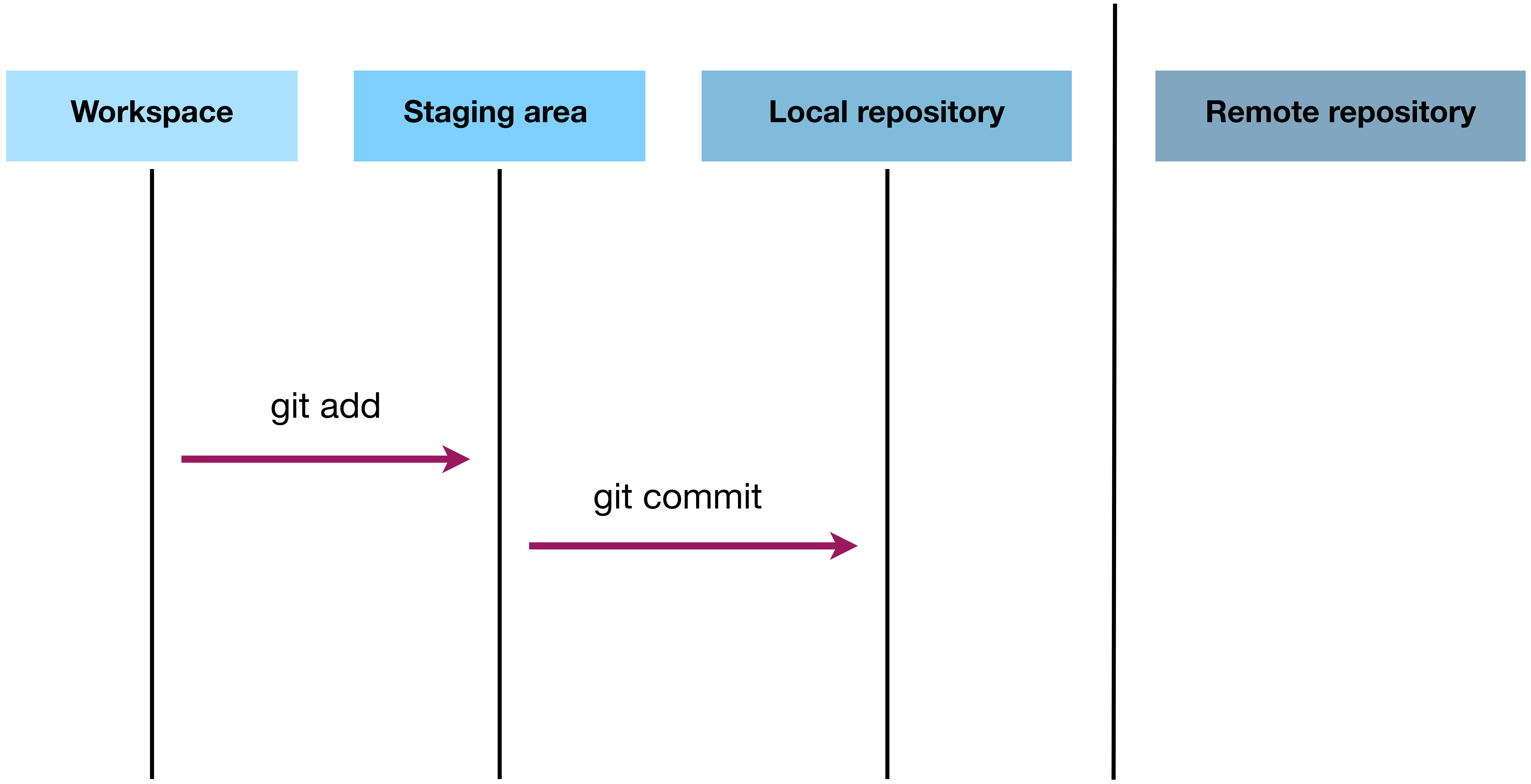
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git config --global user.name "Milena Valentini"
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git config --global user.email valentini@usm.lmu.de
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

Git: essentials and how-to



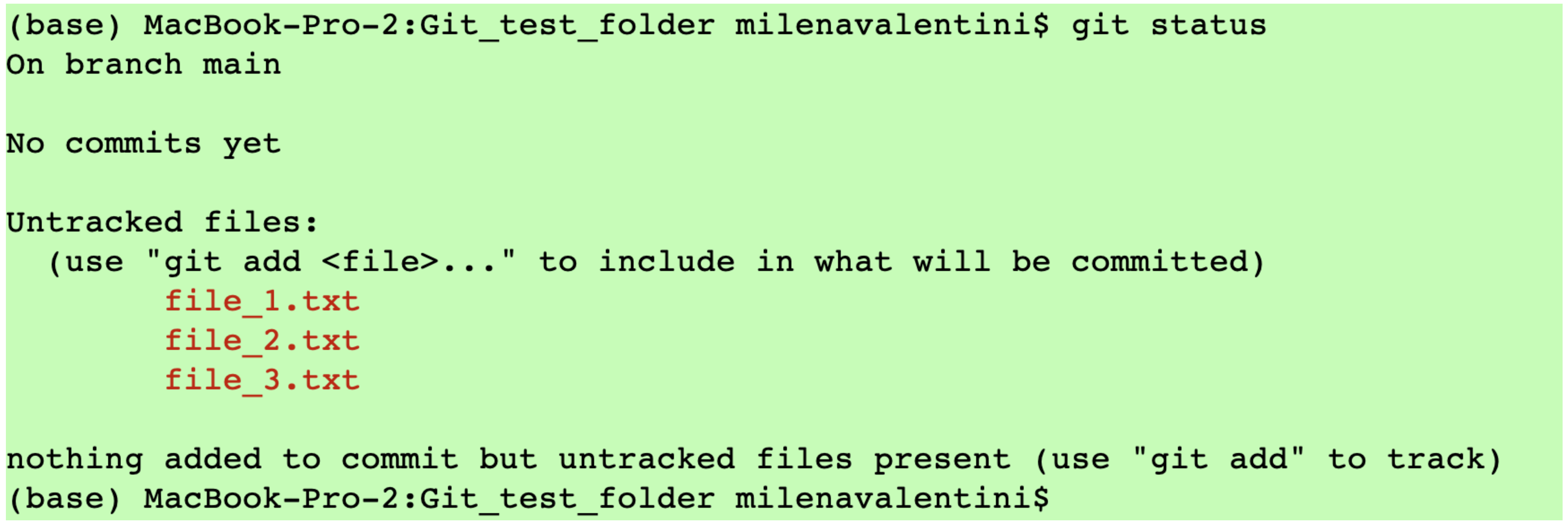
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git status
On branch main

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



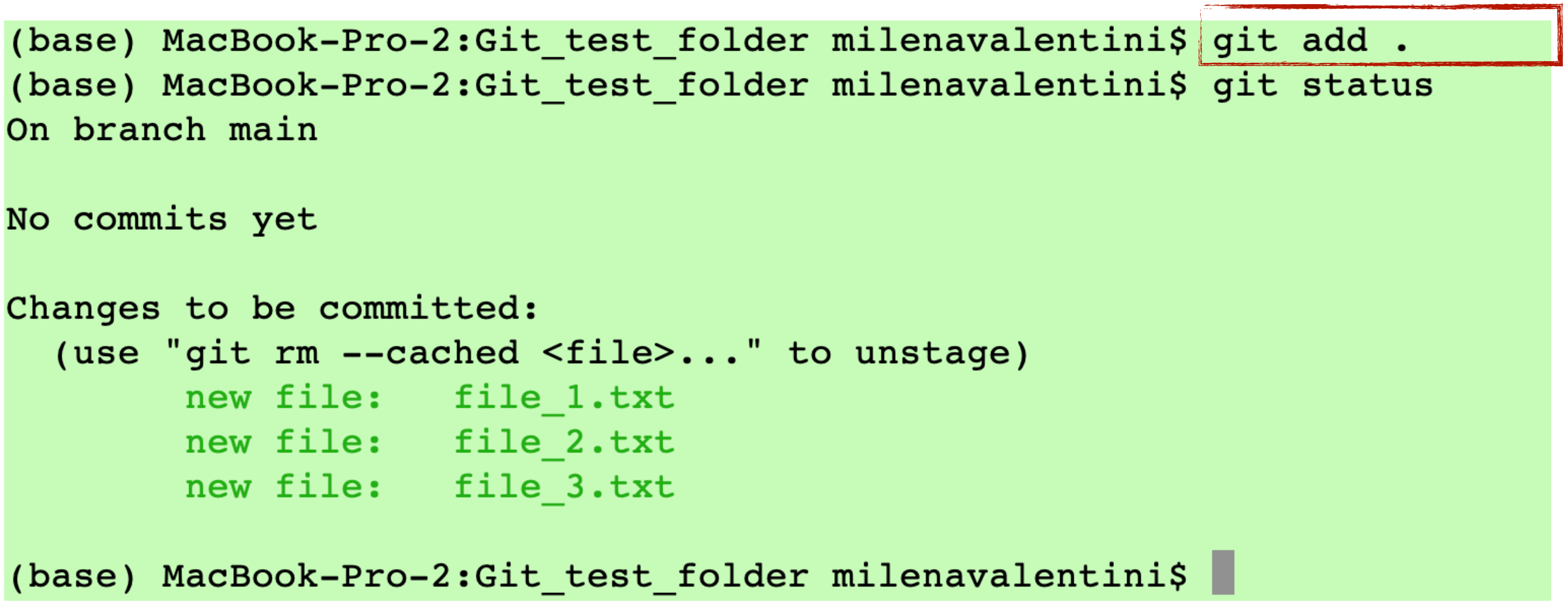
List of Git commands: https://git-scm.com/docs

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$
```

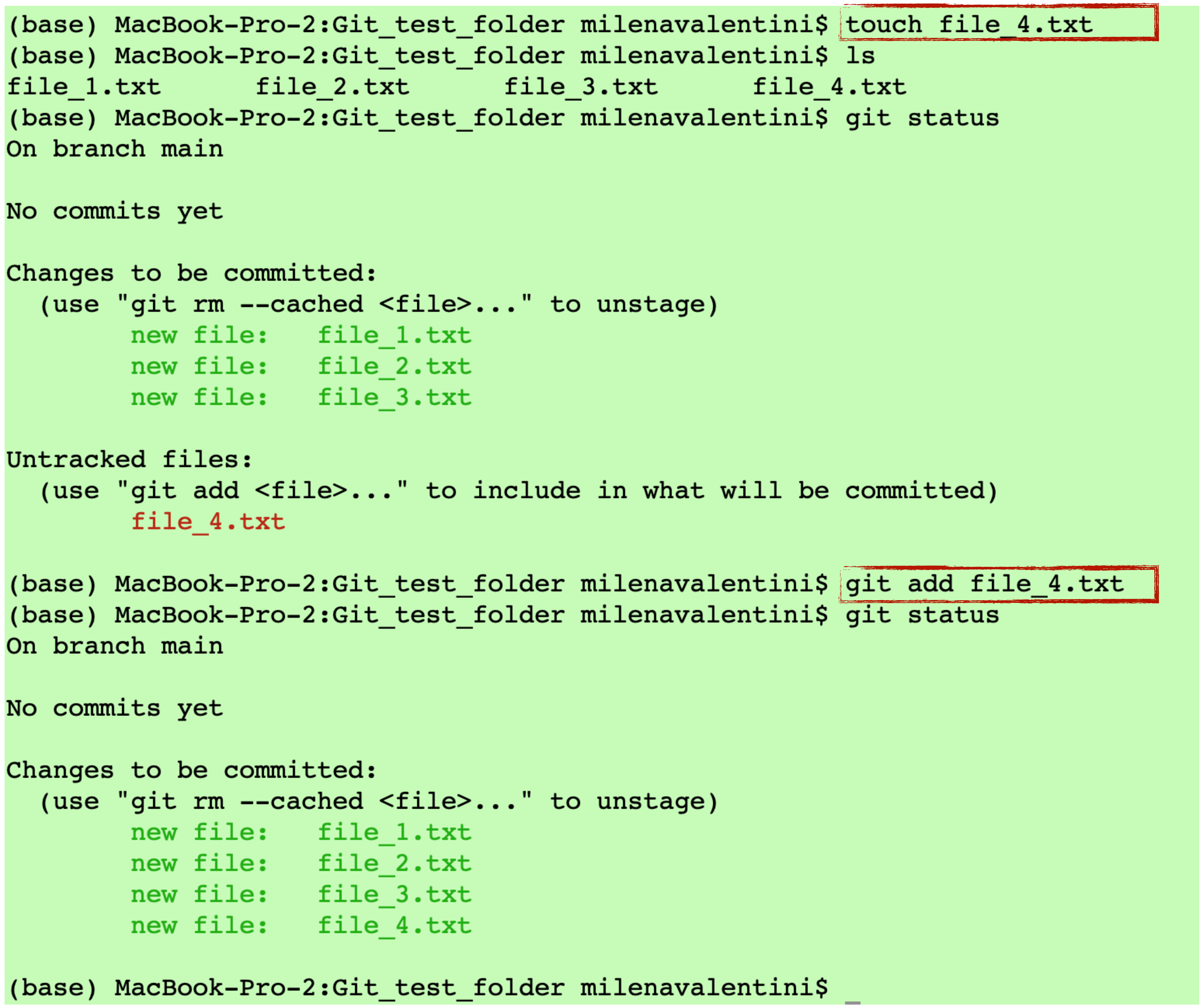
git status displays
an overview of the workspace
and
the current snapshot (only)
of the staging area



```
(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

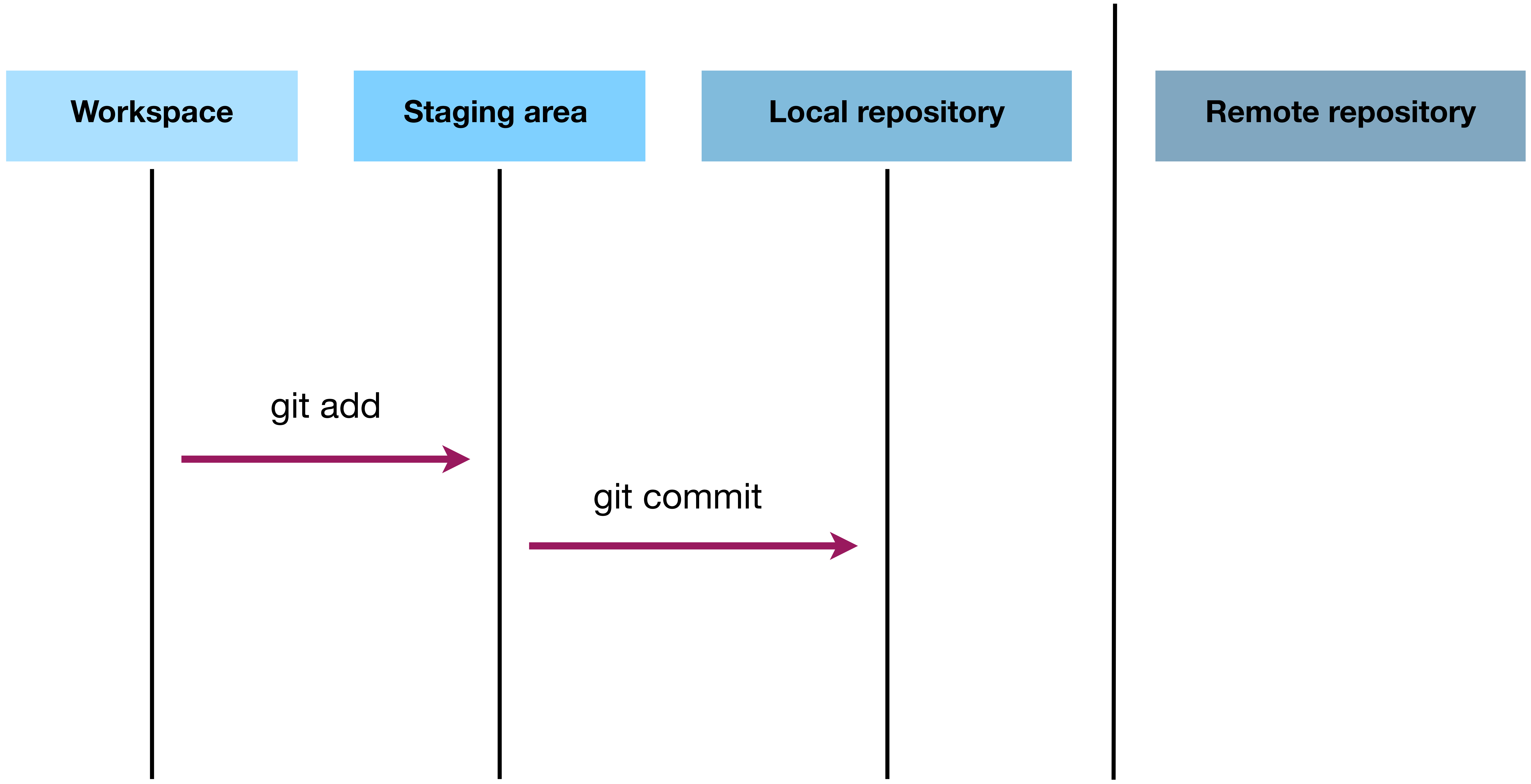


```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ touch file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt      file_4.txt
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git status
On branch main

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

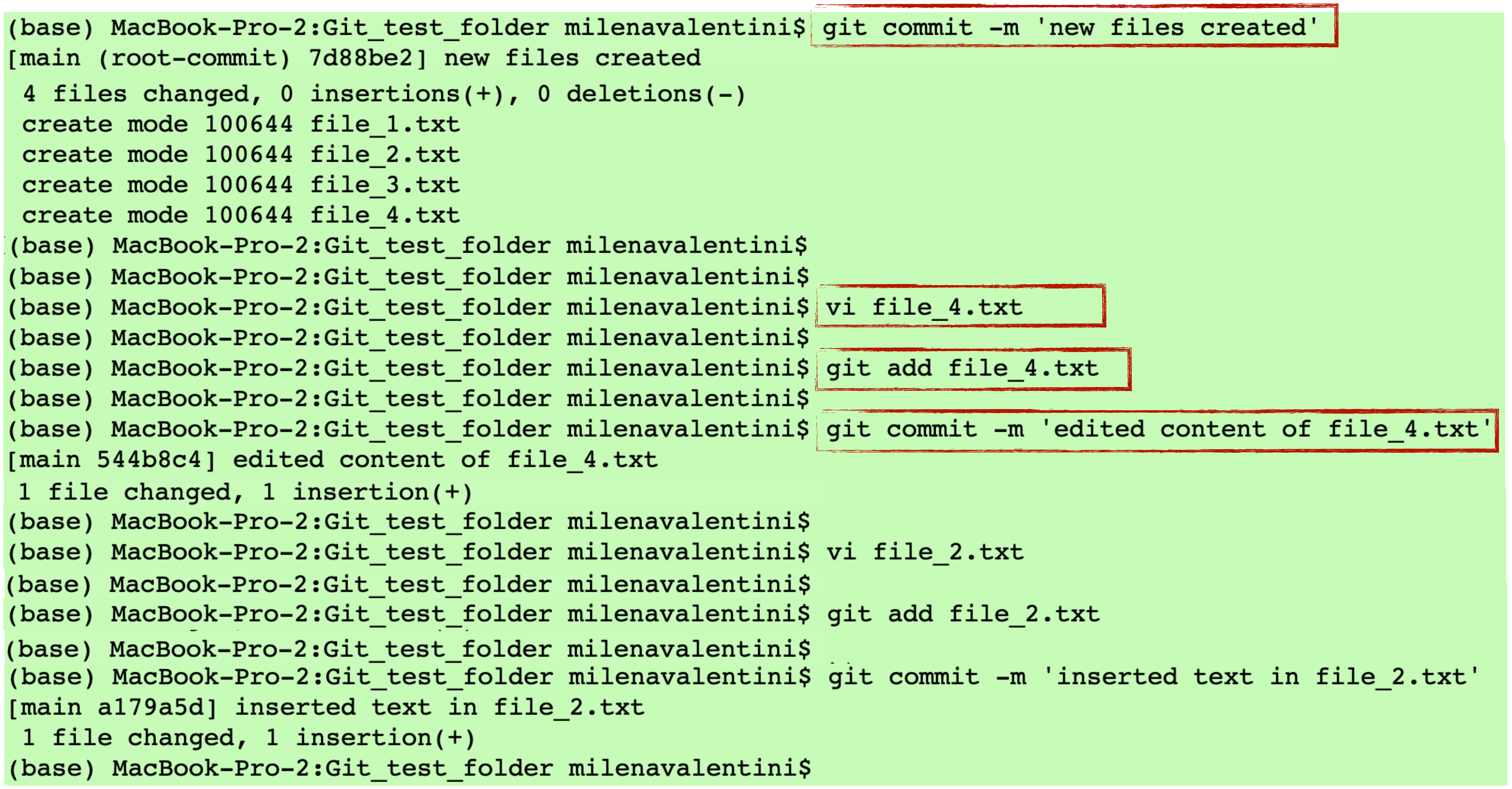
git add to “copy”
all the files in the folder
from the workspace
to the staging area
(otherwise, specify
the filename to be added)

Git: essentials and how-to



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

Git: essentials and how-to

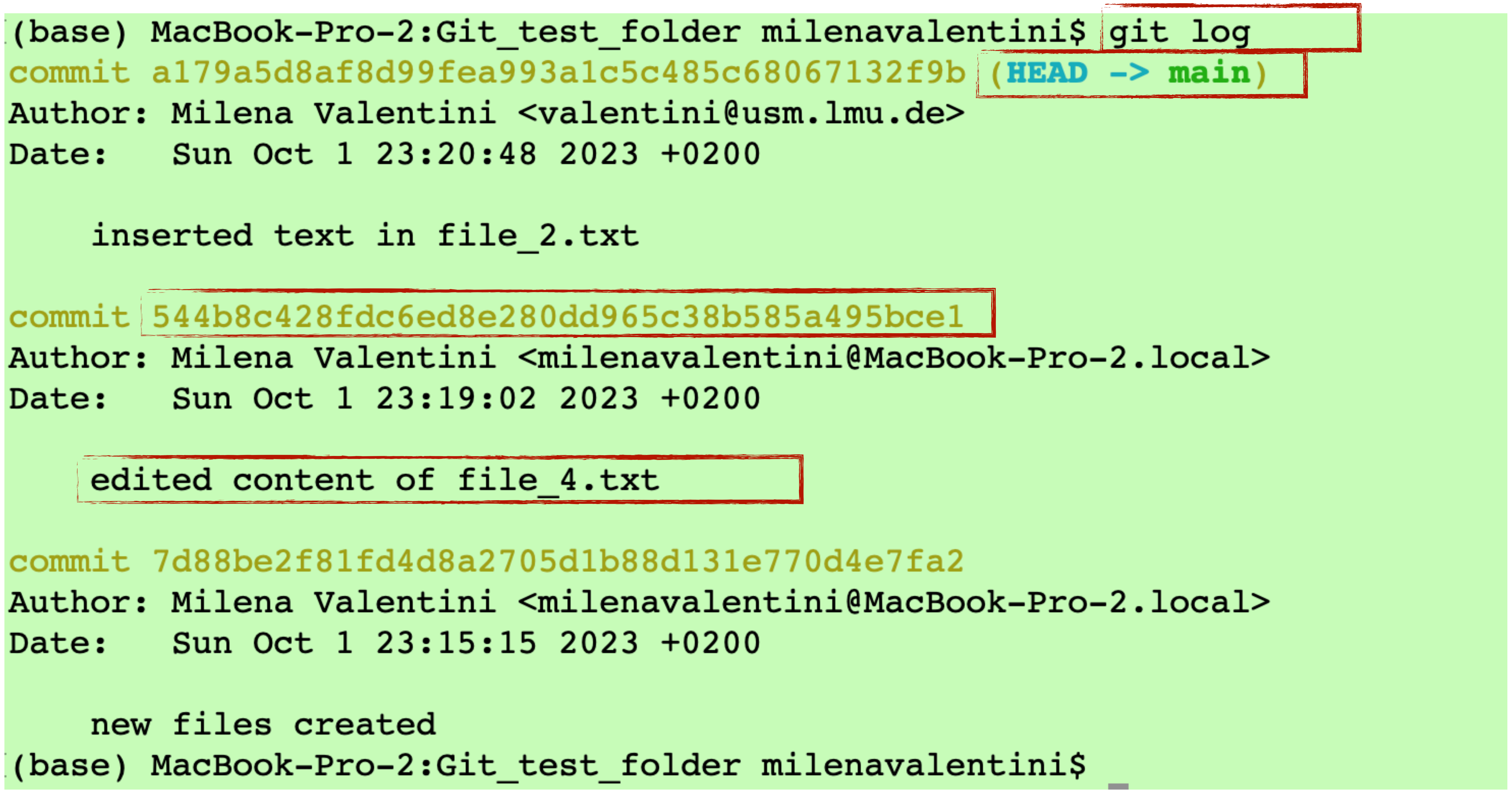


```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git commit -m 'new files created'
[main (root-commit) 7d88be2] new files created
 4 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file_1.txt
 create mode 100644 file_2.txt
 create mode 100644 file_3.txt
 create mode 100644 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)

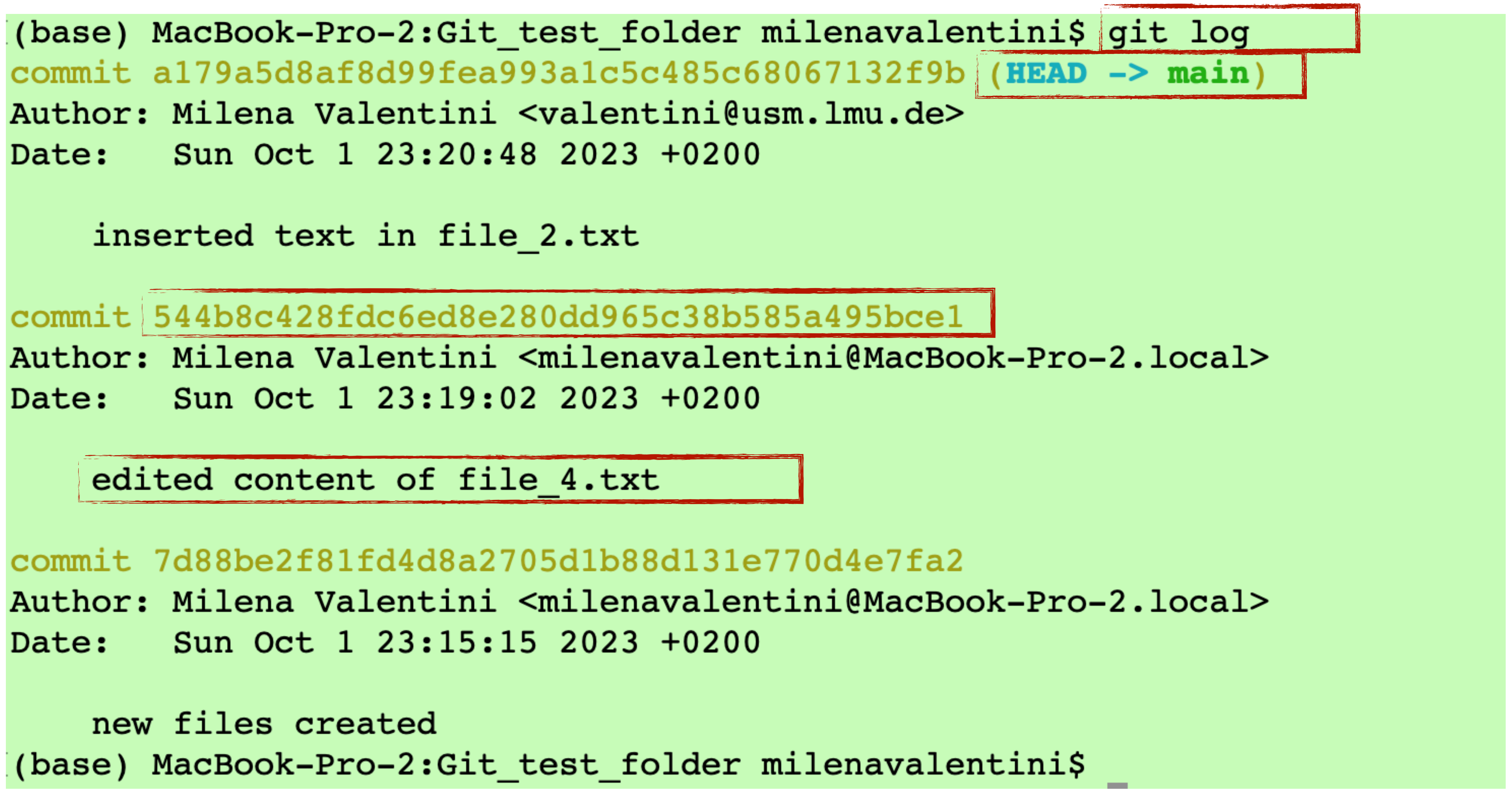
Git: essentials and how-to



```
(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$ 
```

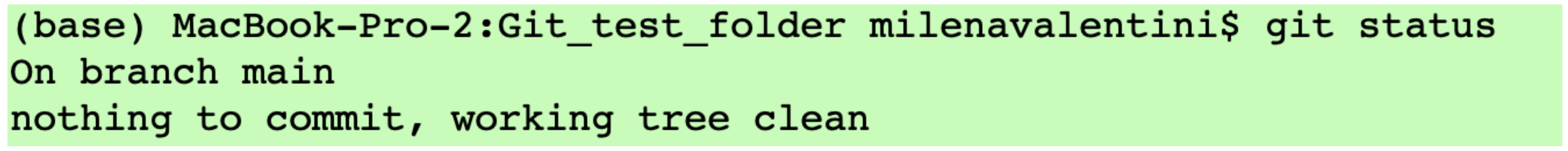
git log
to output
changes with time
(most recent files on top)

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: hands-on

Exercise 1

Create a new directory. Create a file (e.g., my_script_1.py) in the directory and modify it (just write ‘import numpy’ in it).
Verify that the directory is NOT a git repository.
Initialize a git repository in the directory. Verify that it is a git repository now.

Exercise 2

Verify the status of the staging area. Create the first commit.
Verify the commit history of the repository.

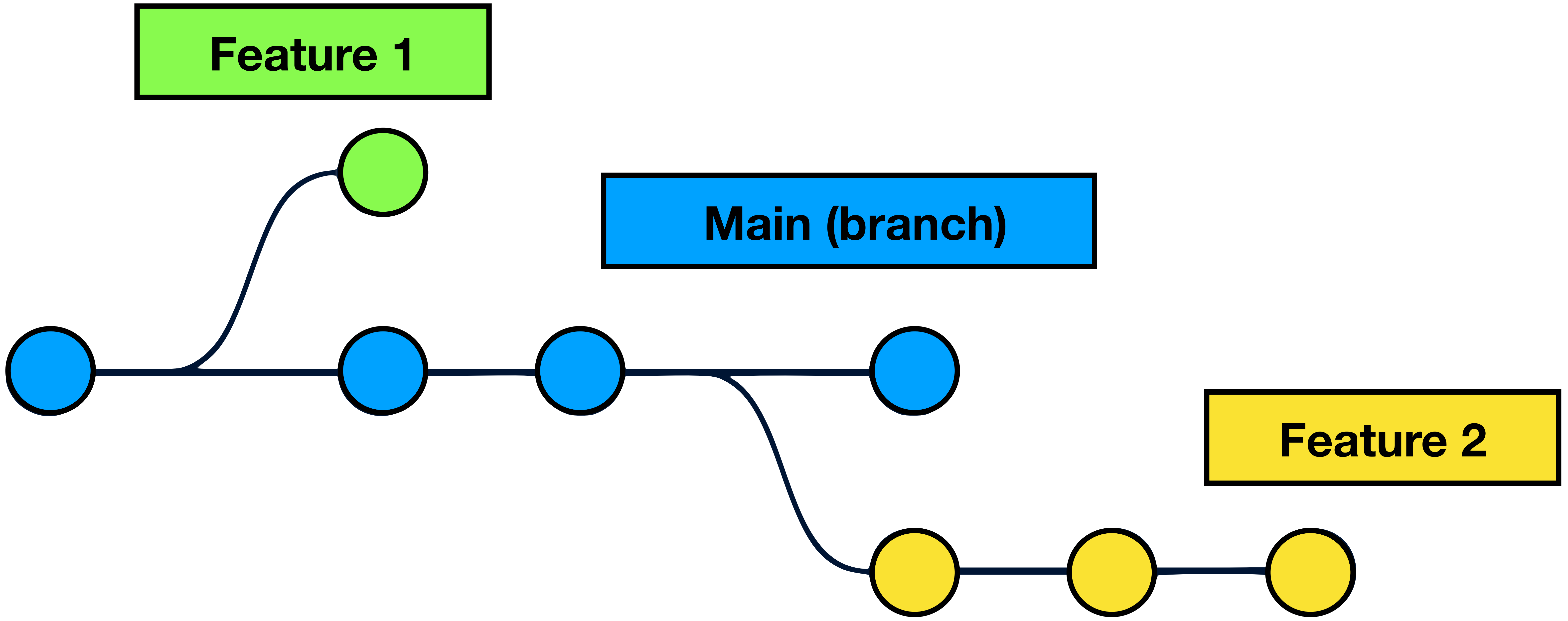
Exercise 3

Modify my_script_1.py (to import scipy) and create a new commit.
Check the commit history.

Useful commands:

git init
git status
git add
git commit
git log
git branch
git switch
git merge
git remote add (origin)
git push (-u origin)
git pull

Git: branches



Branches store different versions of your project

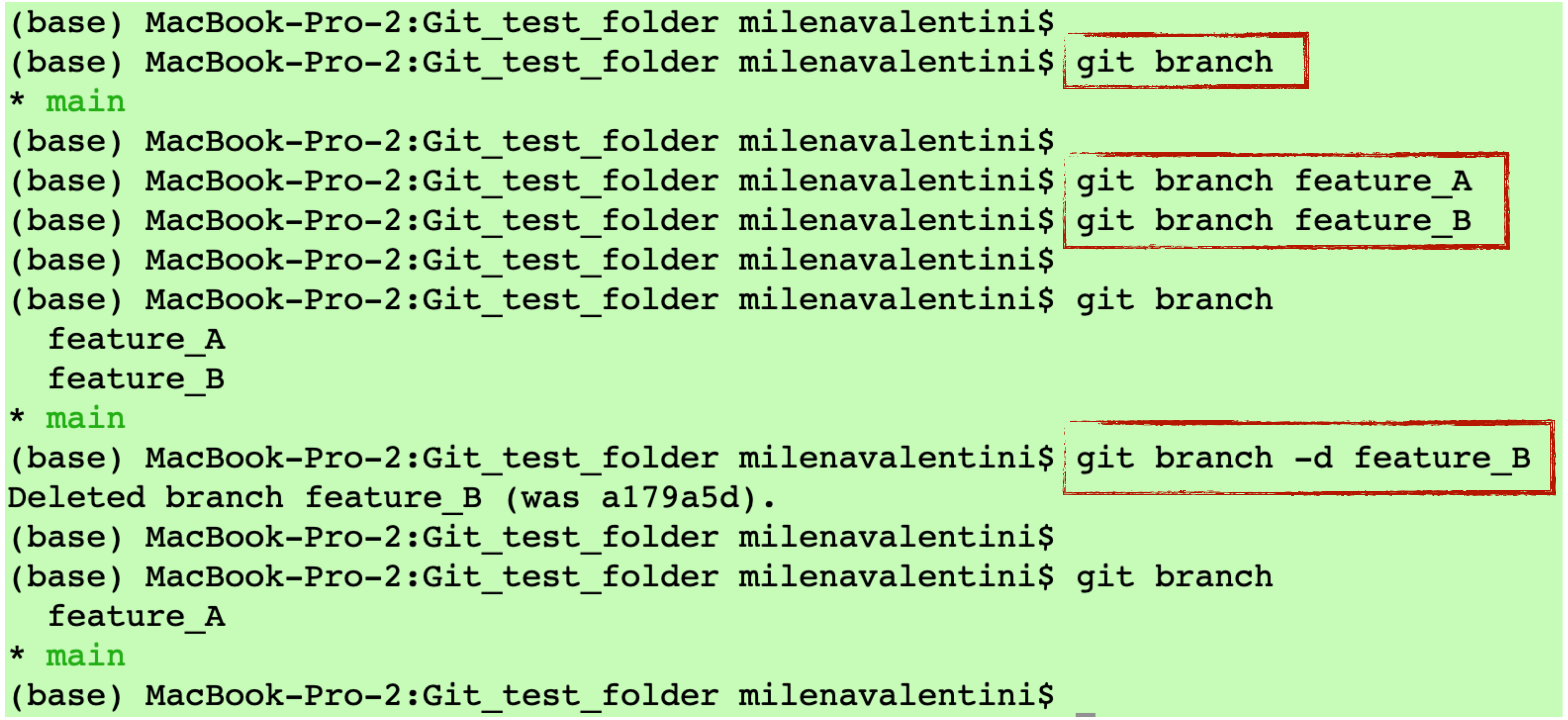
They are pointers to a commit

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

Key for parallel development

Git: branches

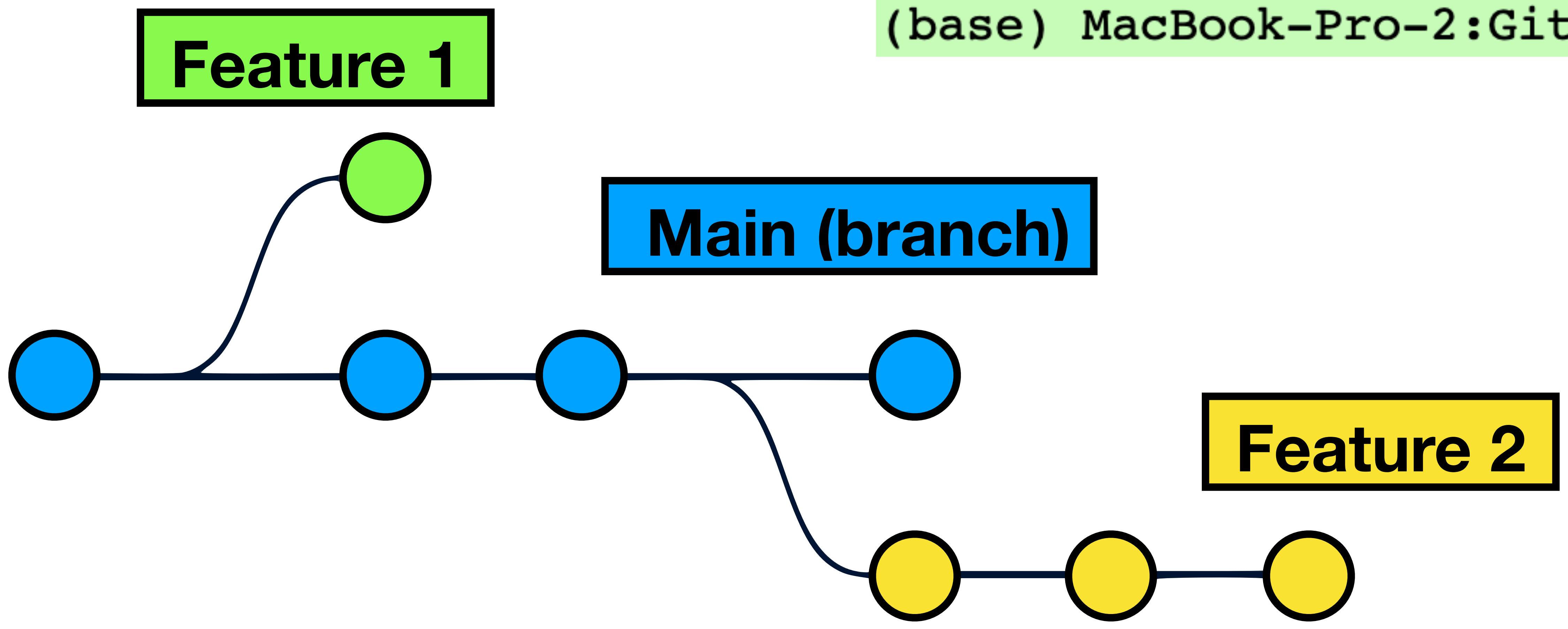
List all existing local branches



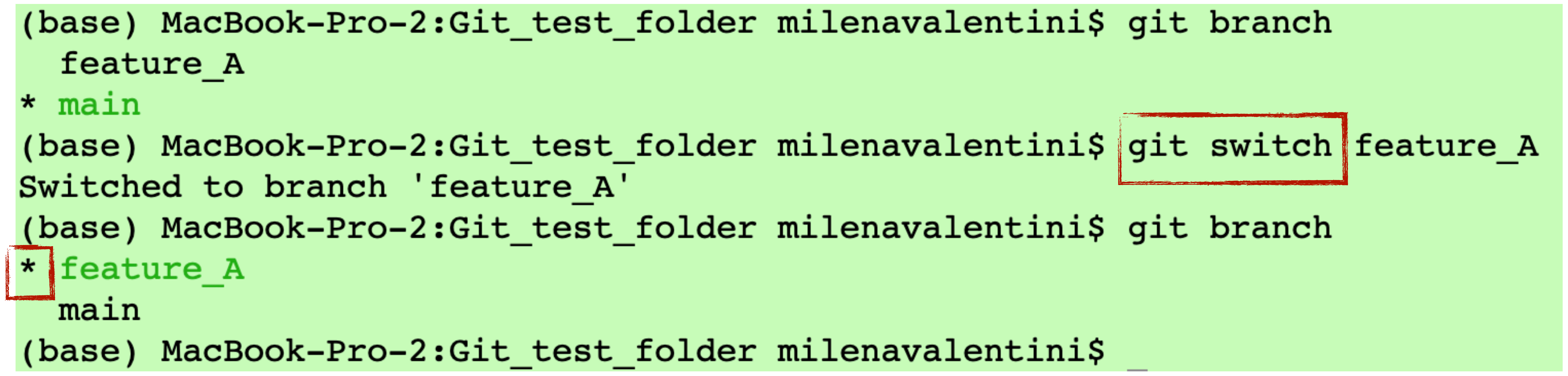
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch feature_A
(base) MacBook-Pro-2:Git_test_folder milenavalentini$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

Create a new branch

Delete a branch



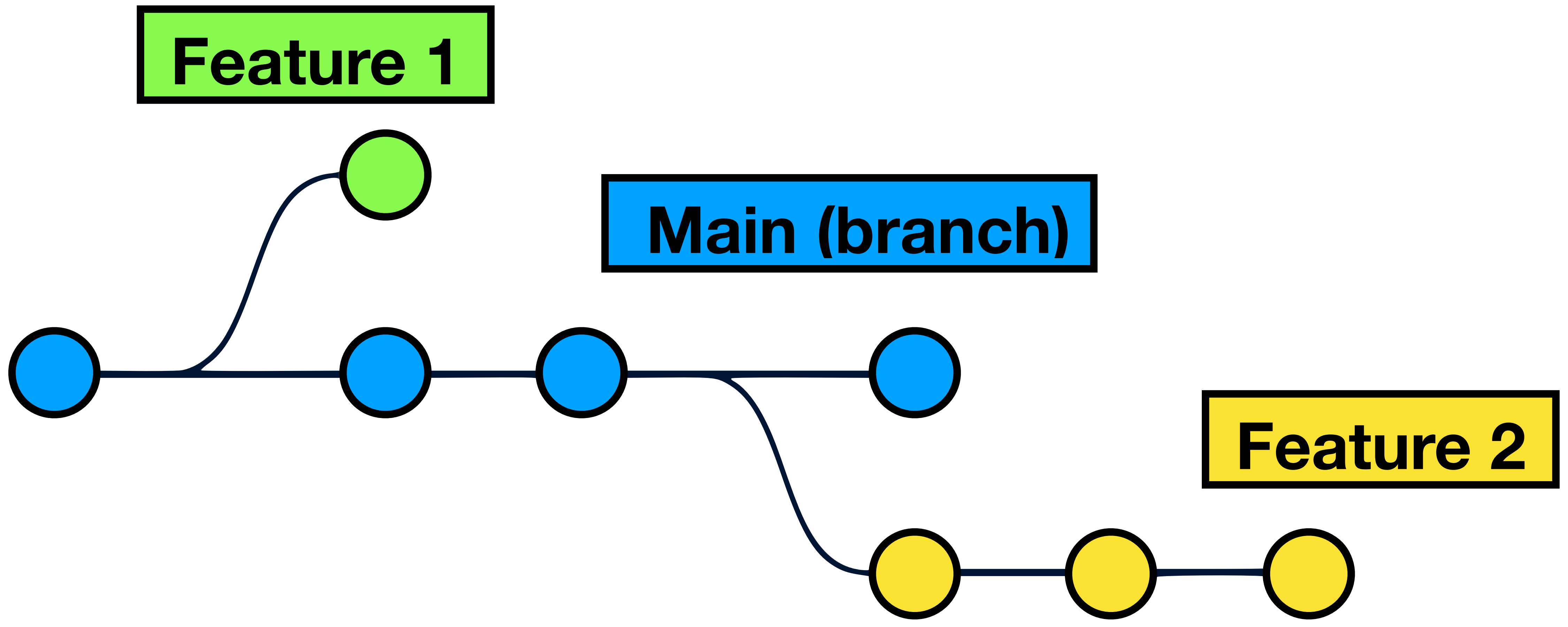
Git: branches



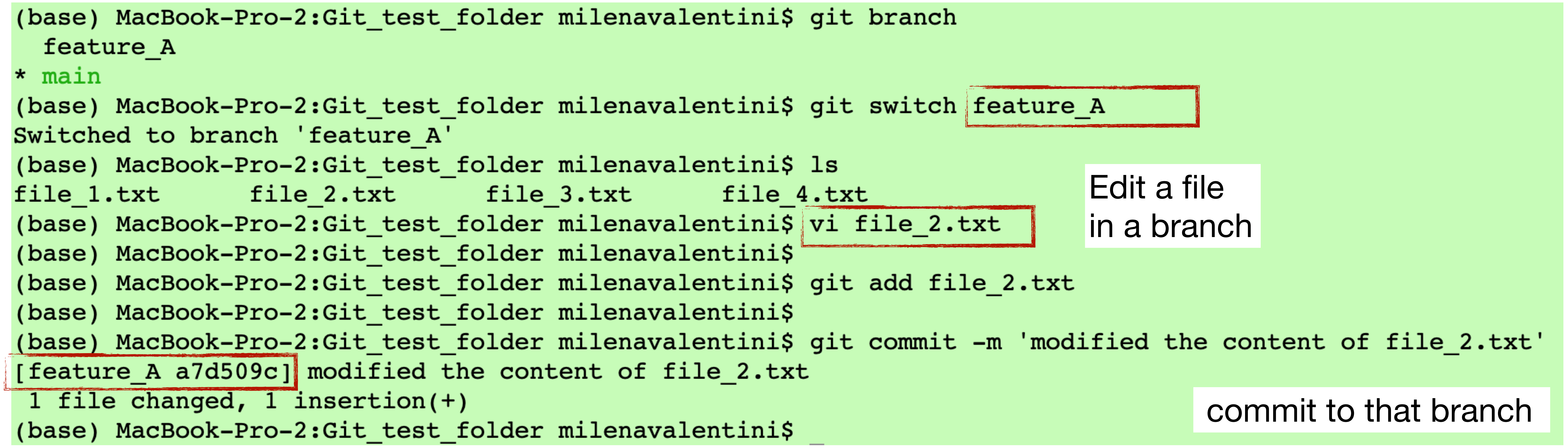
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_A
Switched to branch 'feature_A'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
* feature_A
  main
(base) MacBook-Pro-2:Git_test_folder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

Switch to another branch

git branch operates on the local repository



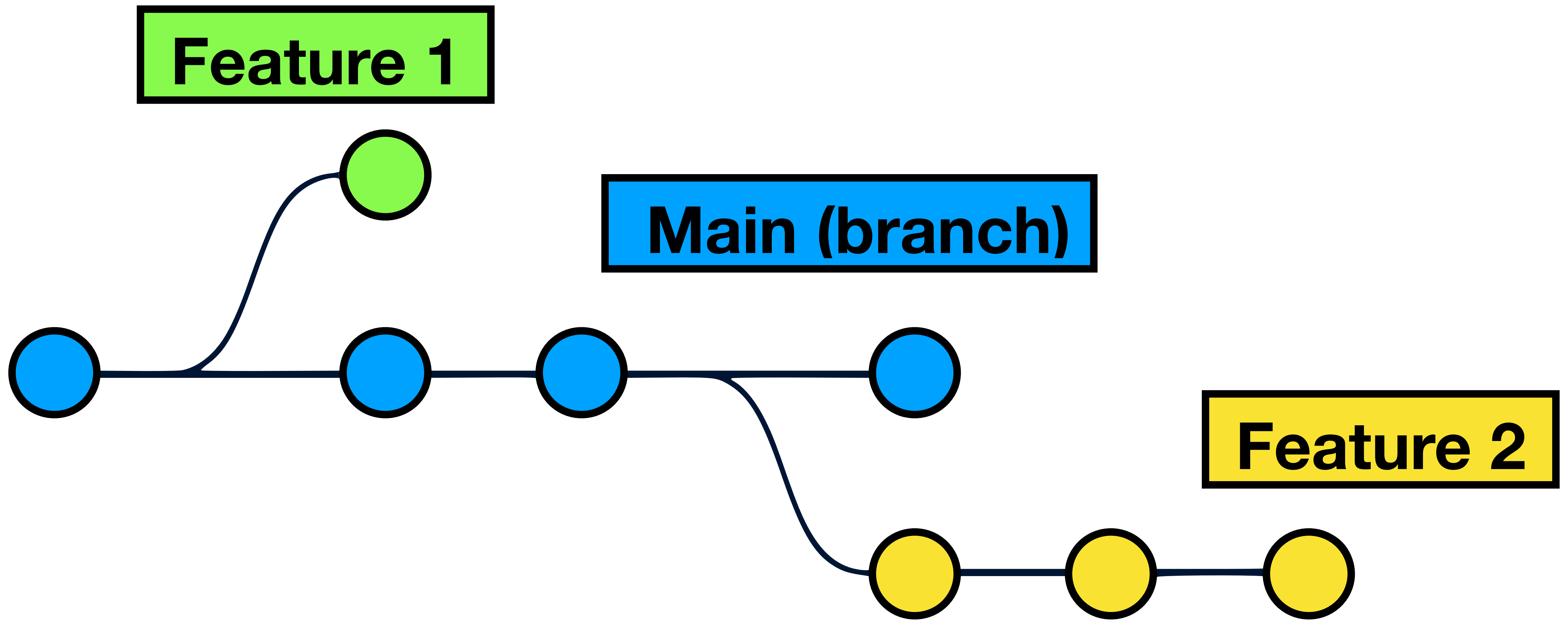
Git: branches



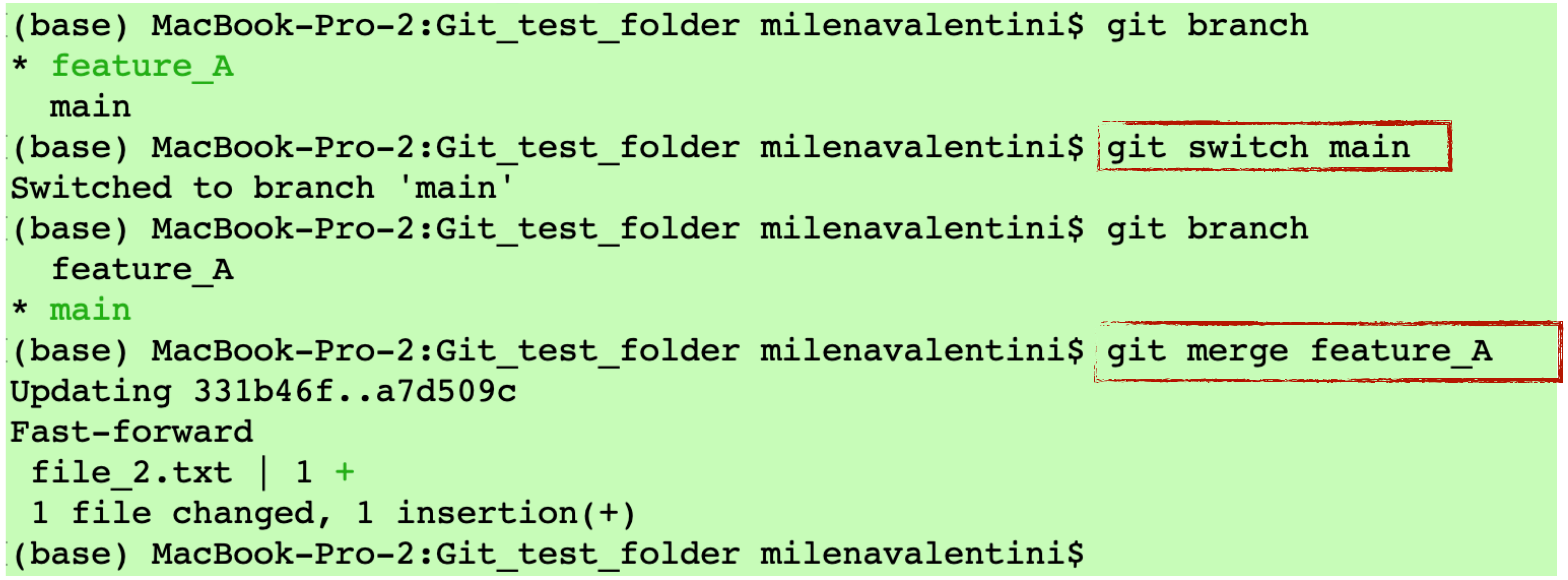
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_A
Switched to branch 'feature_A'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ ls
file_1.txt      file_2.txt      file_3.txt      file_4.txt
(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

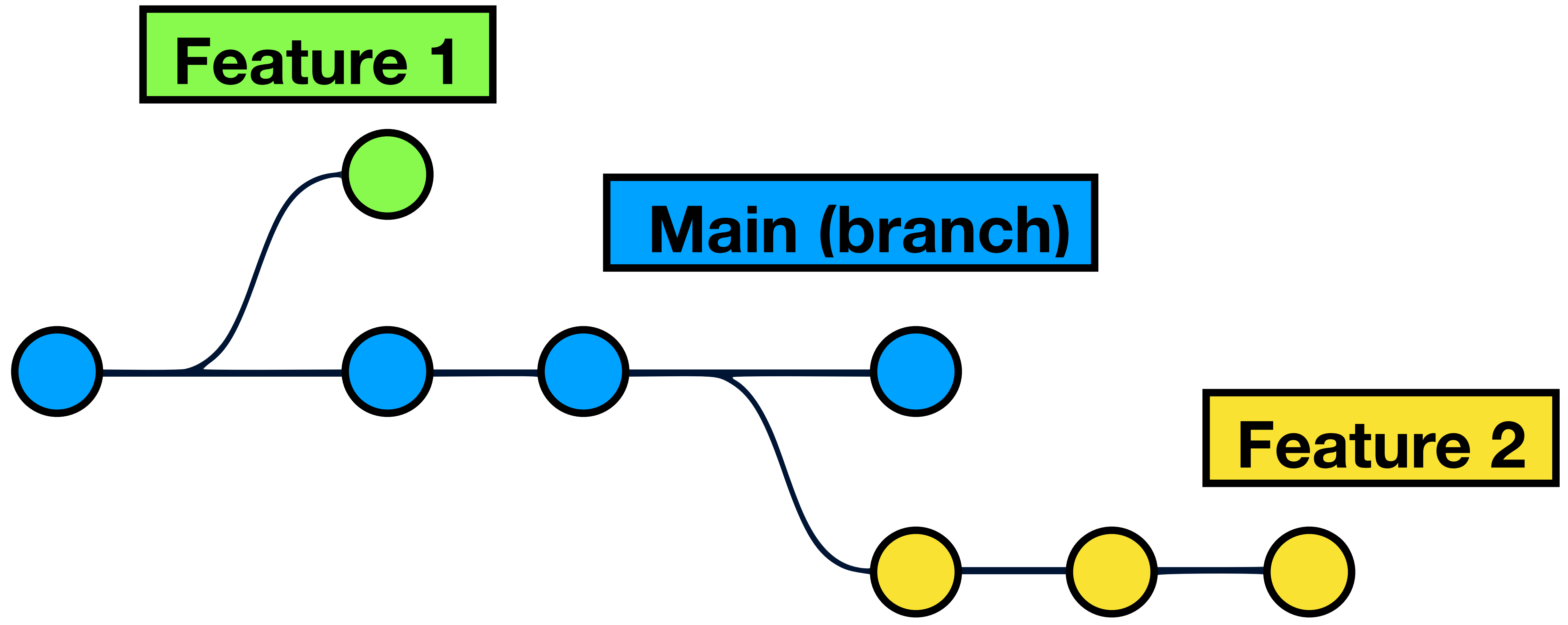
commit to that branch



Git: branches



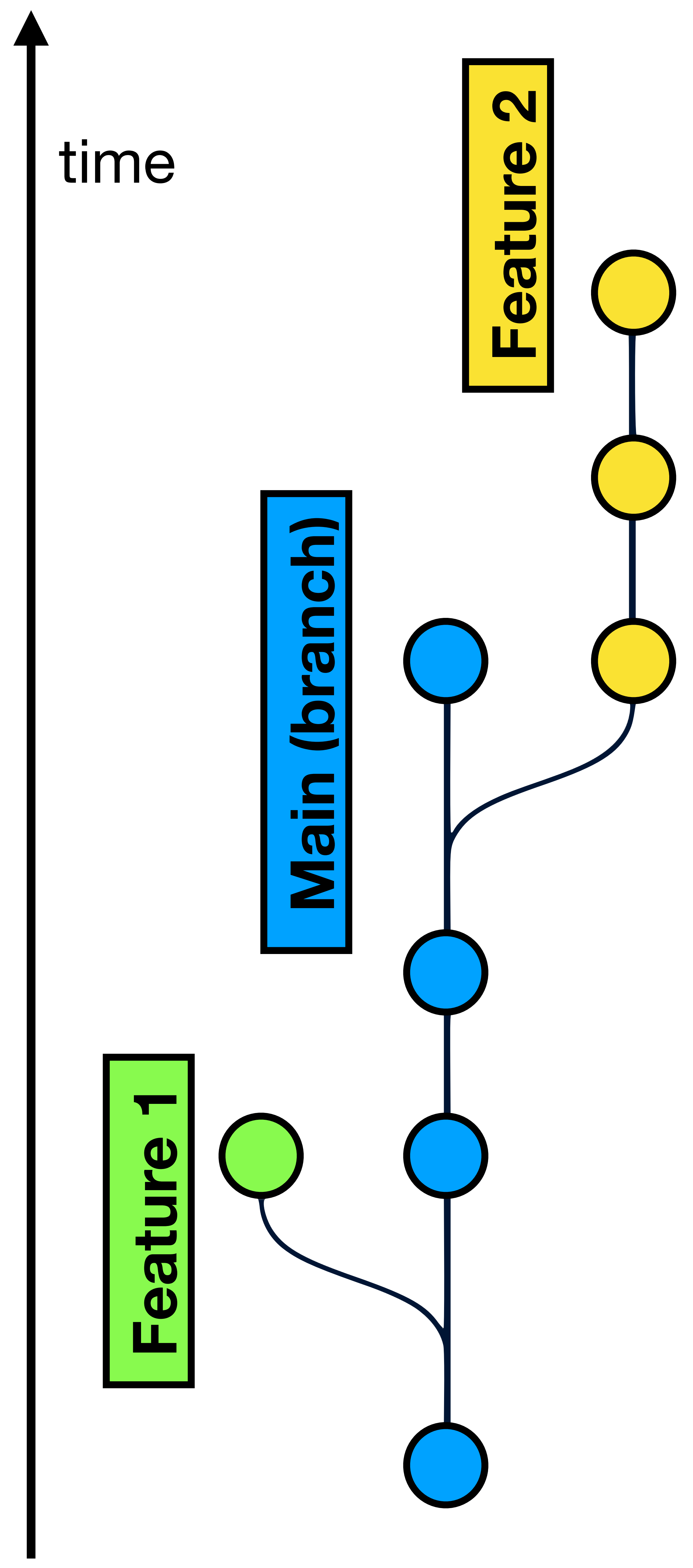
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
* feature_A
  main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch main
Switched to branch 'main'
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
* main
(base) MacBook-Pro-2:Git_test_folder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

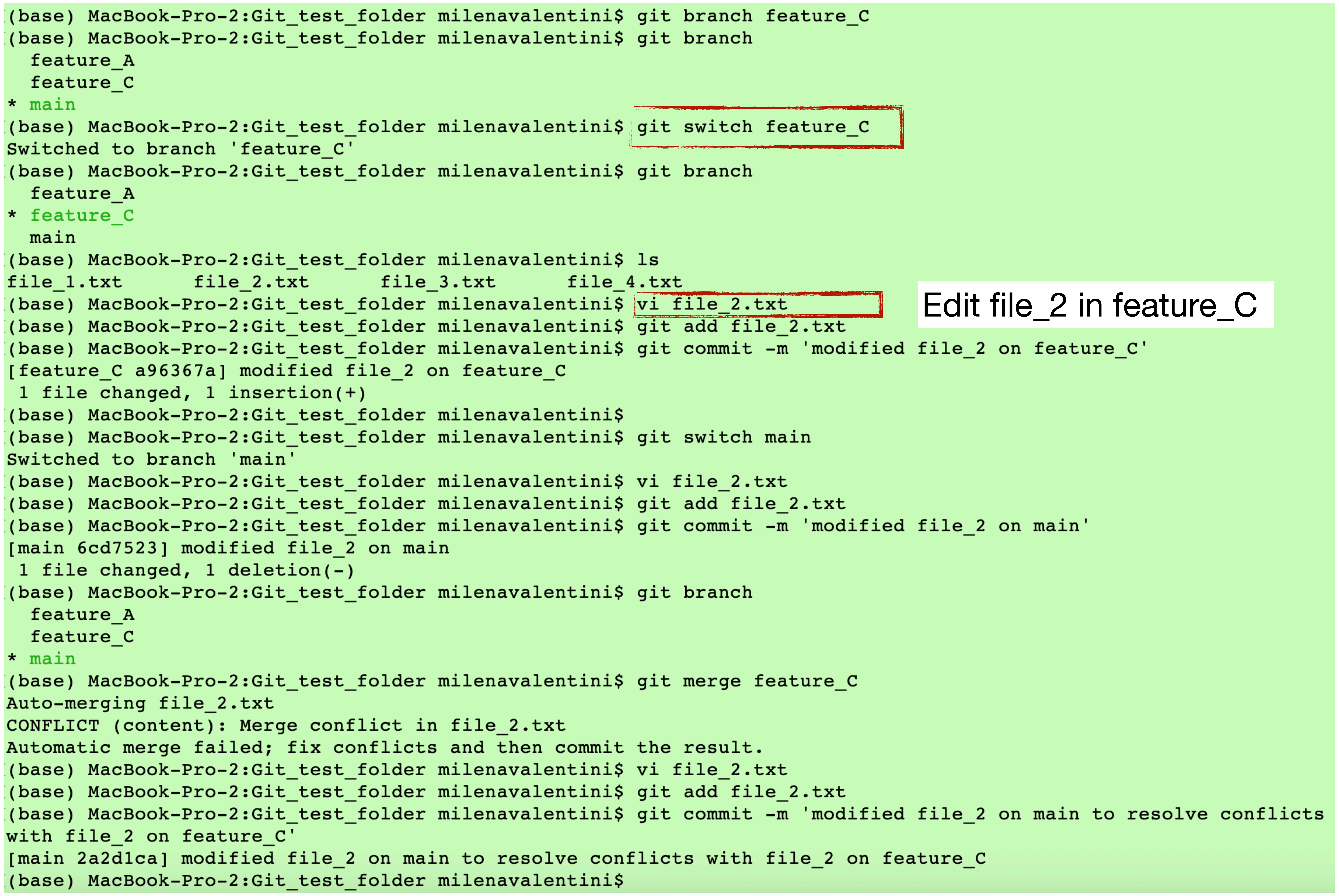


Copies the content
of the specified branch (i.e. feature_A)
in the current branch (i.e. main)

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

Git: branches

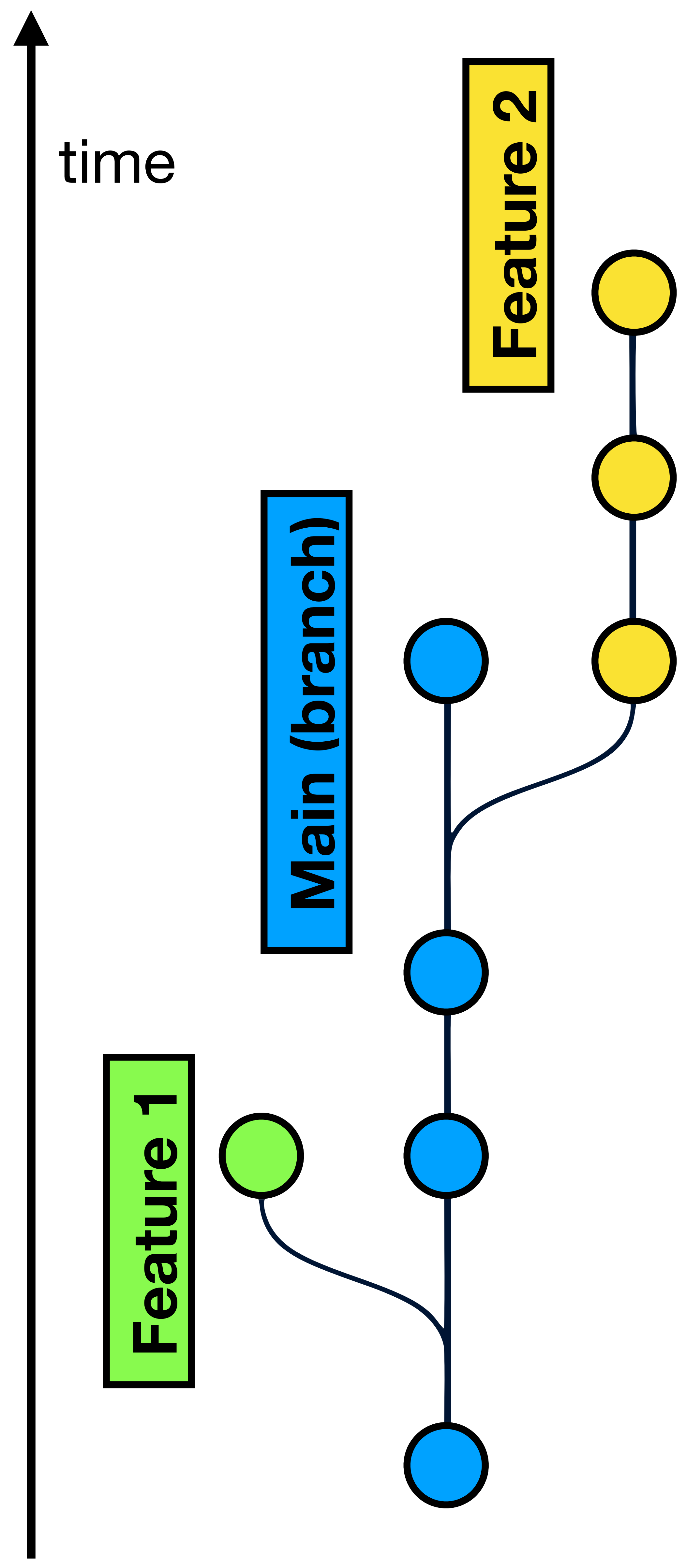


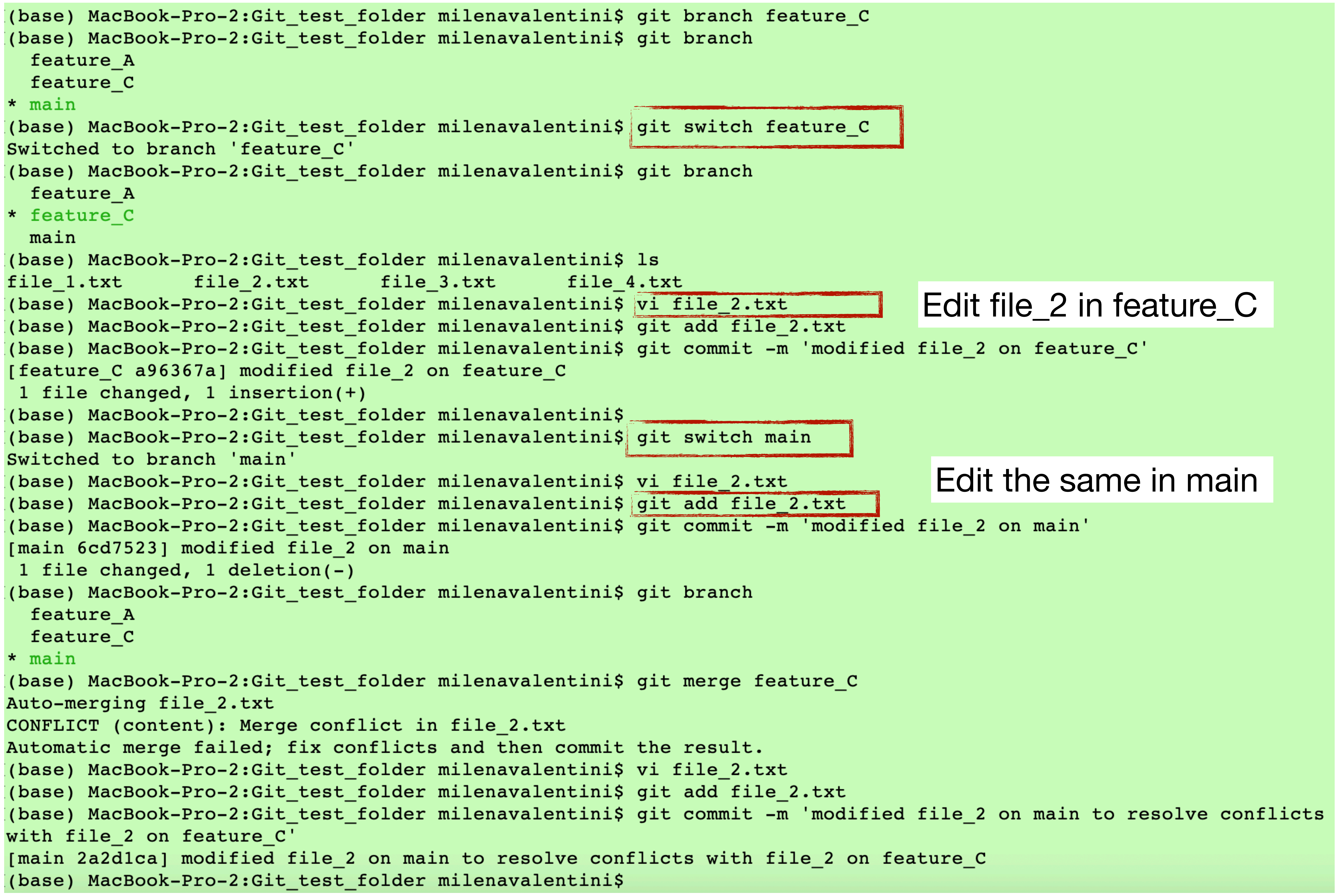


```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_C
Switched to branch 'feature_C'
(base) MacBook-Pro-2:Git_test_folder 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 2a2d1ca] 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



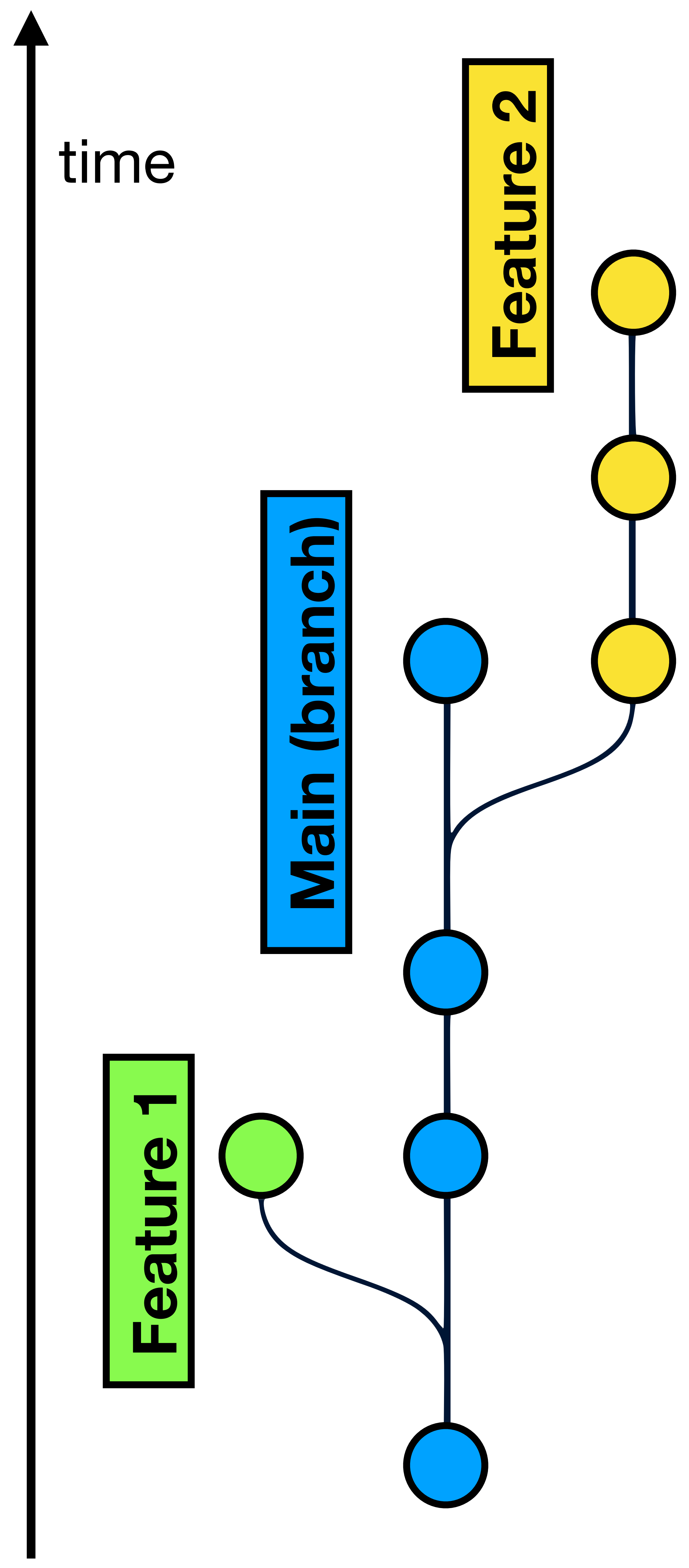


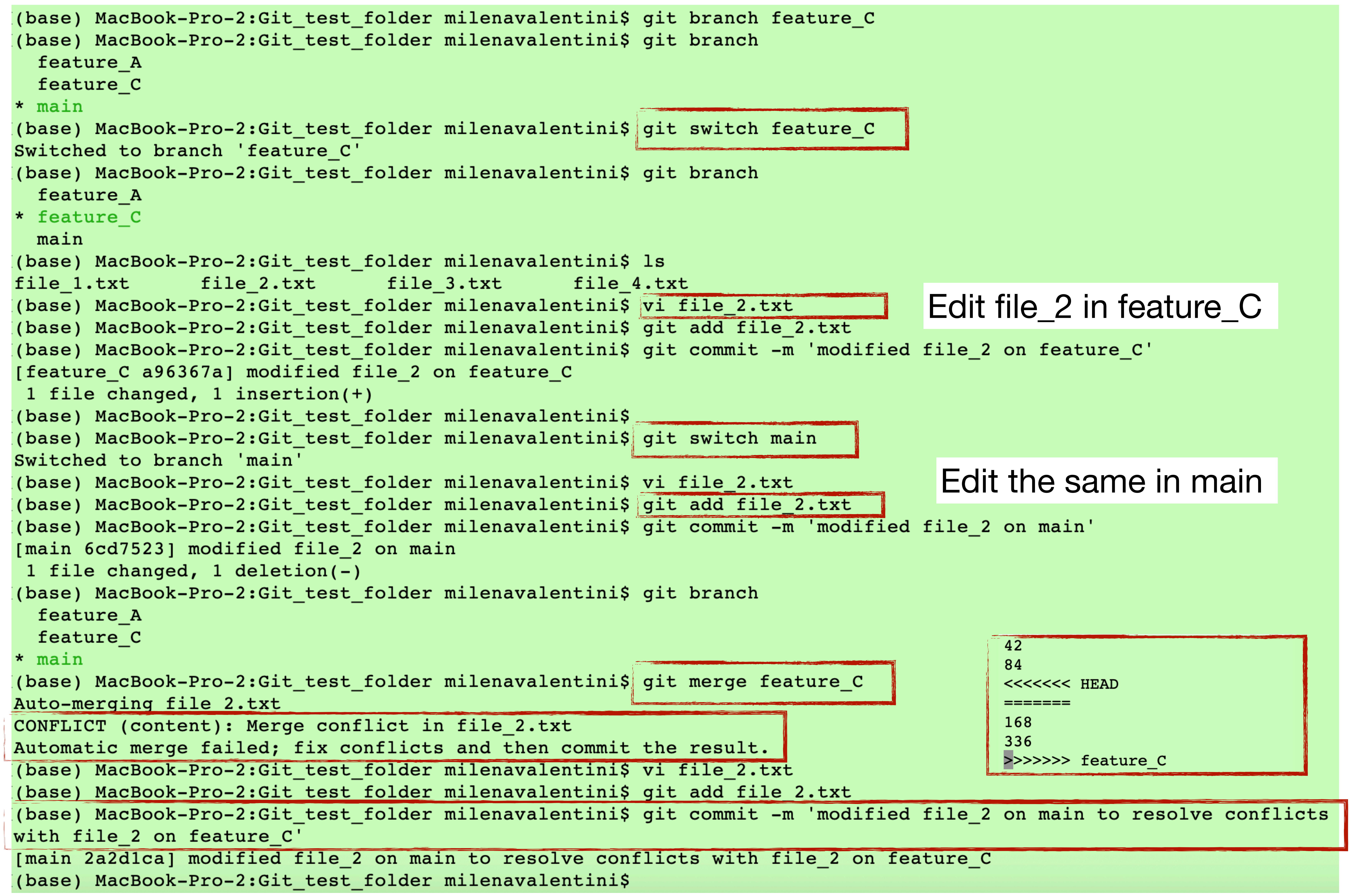
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_C
Switched to branch 'feature_C'
(base) MacBook-Pro-2:Git_test_folder 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 2a2d1ca] 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





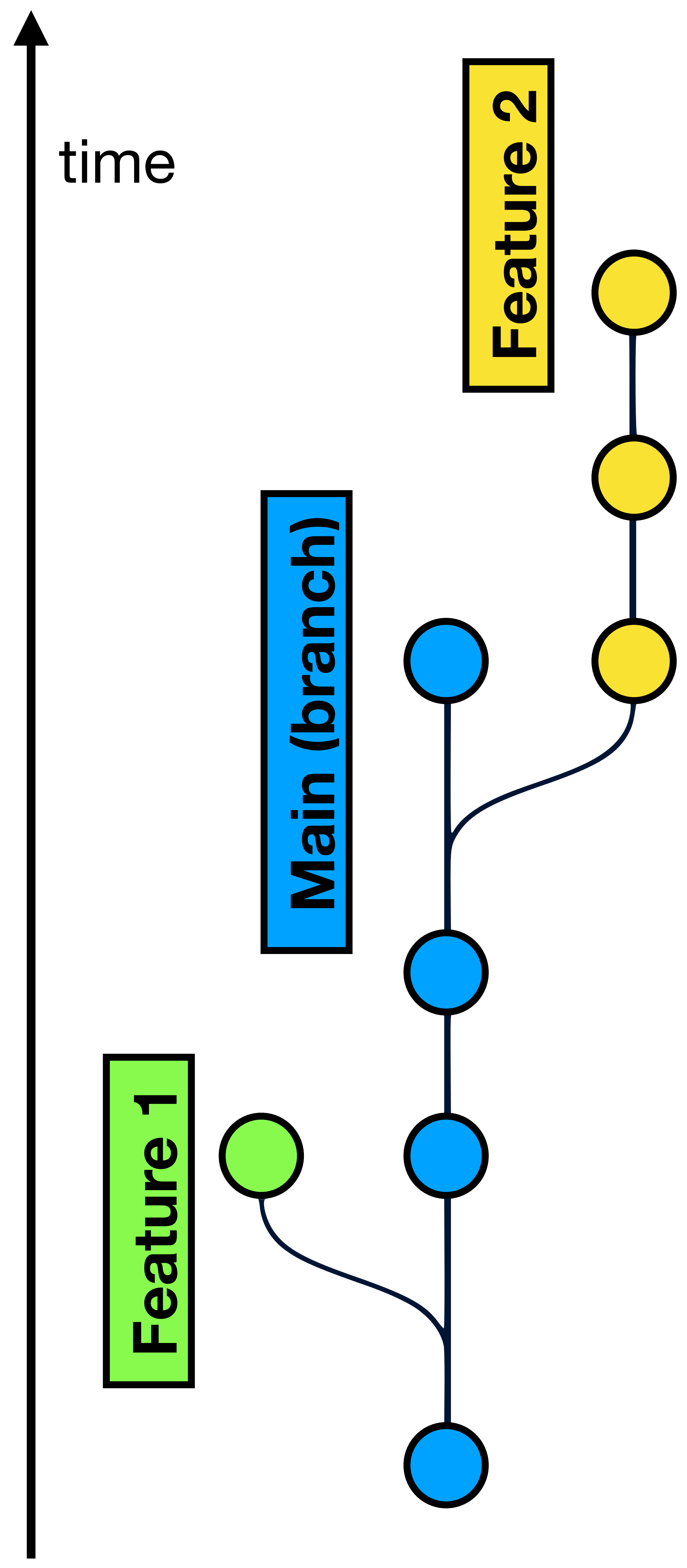
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch
  feature_A
  feature_C
* main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git switch feature_C
Switched to branch 'feature_C'
(base) MacBook-Pro-2:Git_test_folder 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 2a2d1ca] modified file_2 on main to resolve conflicts with file_2 on feature_C
(base) MacBook-Pro-2:Git_test_folder milenavalentini$
```

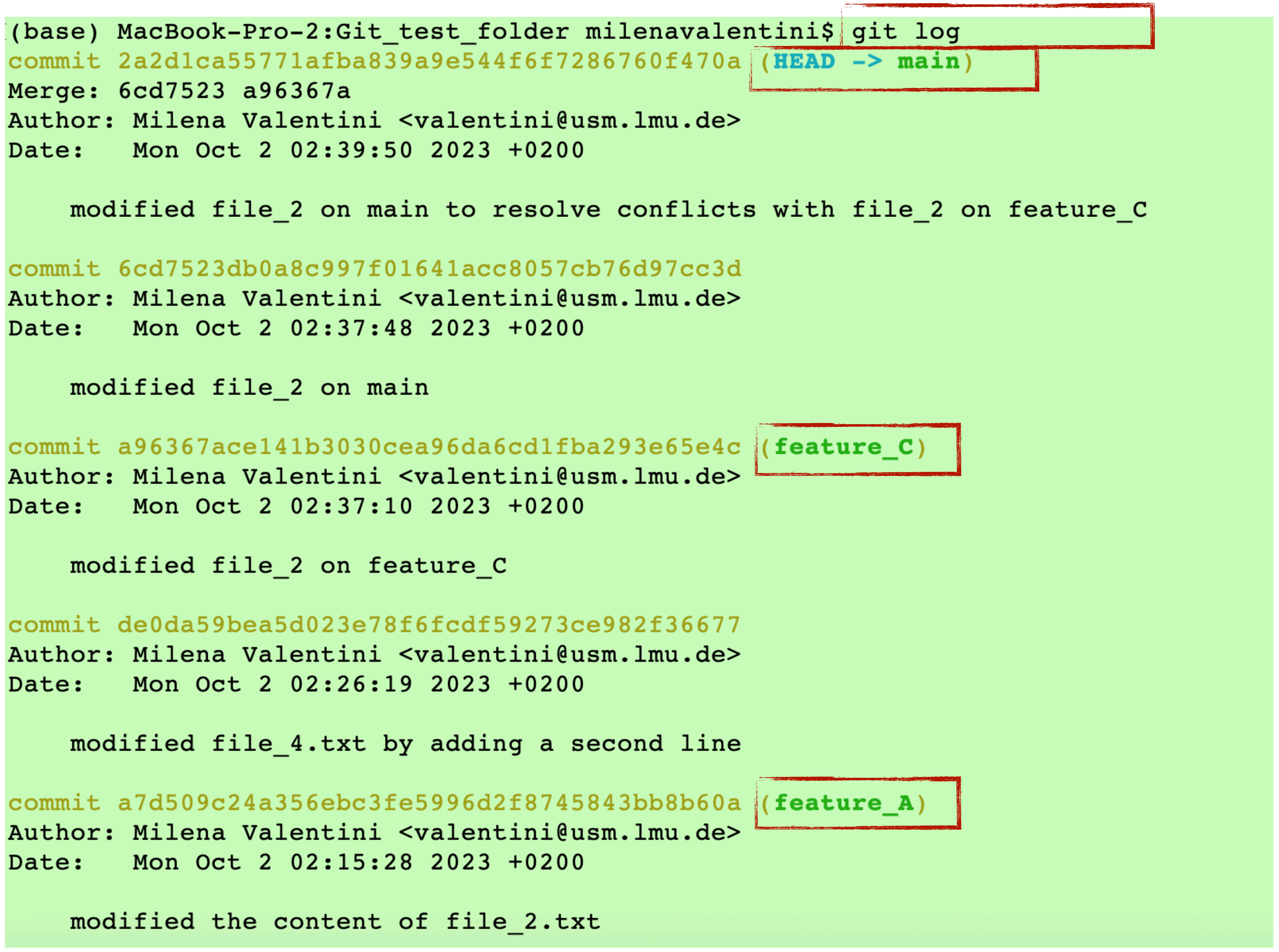
Edit file_2 in feature_C

Edit the same in main

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

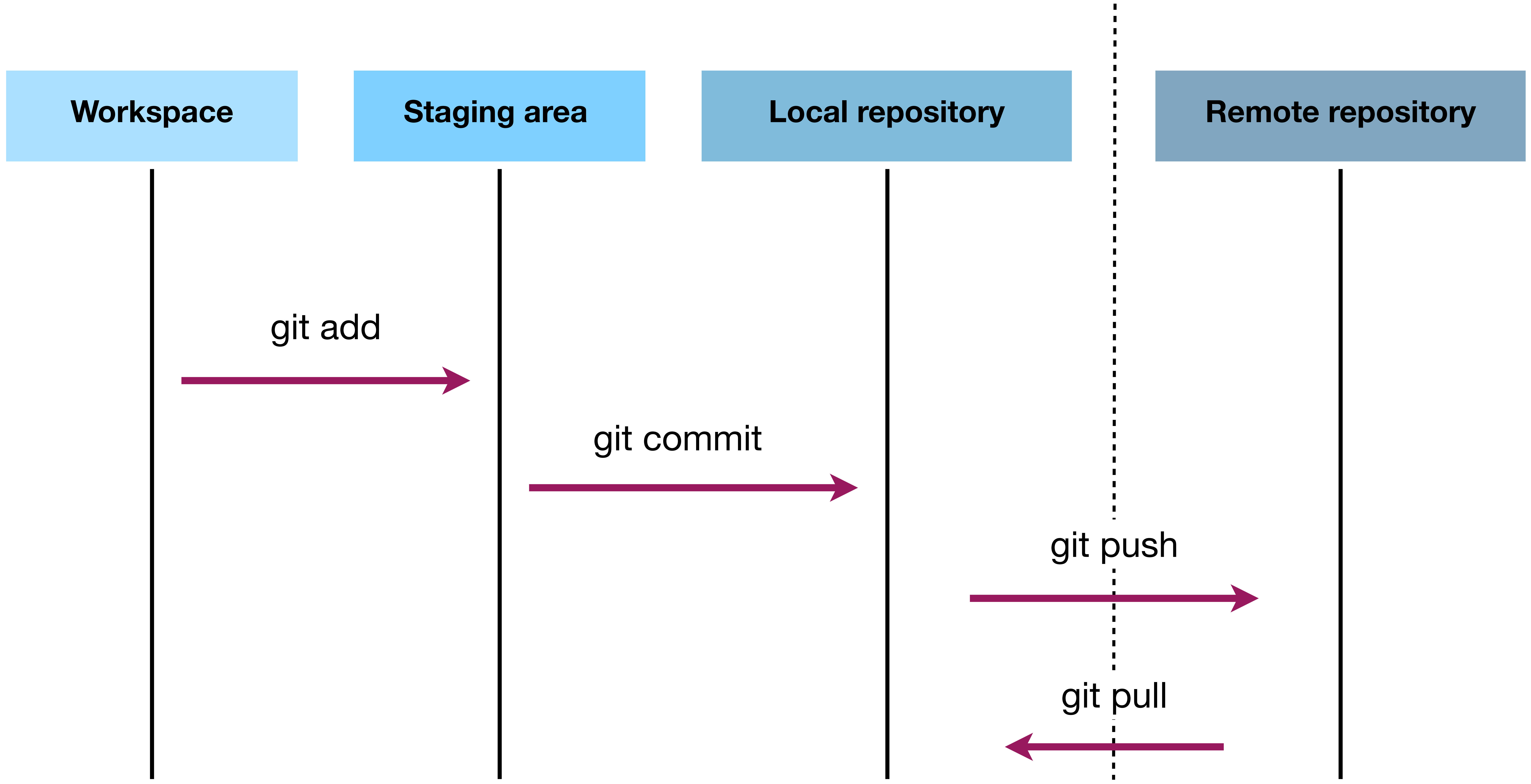
Git: branches



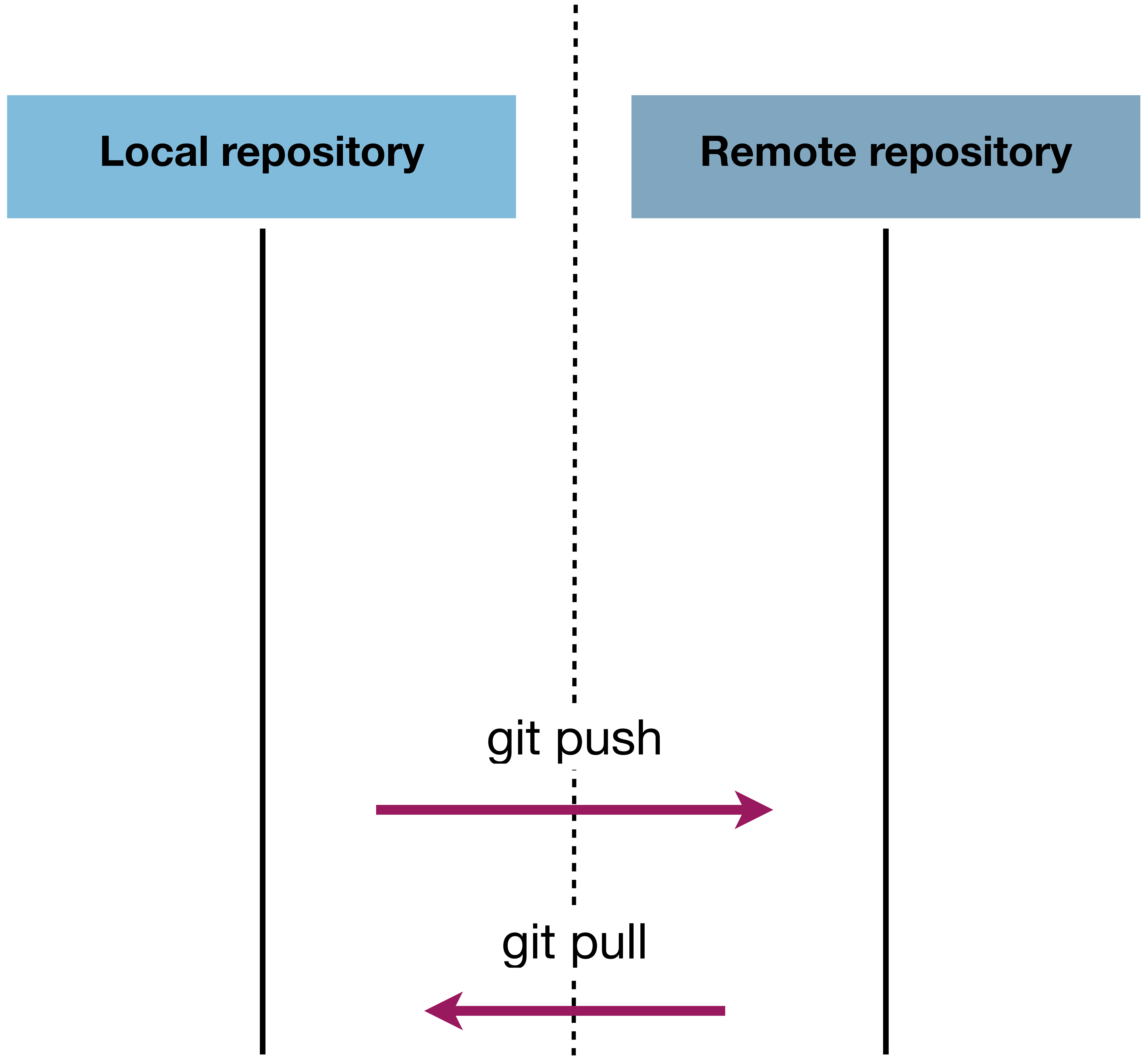


```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git log
commit 2a2d1ca55771afba839a9e544f6f7286760f470a (HEAD -> main)
Merge: 6cd7523 a96367a
Author: Milena Valentini <valentini@usm.lmu.de>
Date:   Mon Oct 2 02:39:50 2023 +0200

    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 de0da59bea5d023e78f6fcdf59273ce982f36677
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

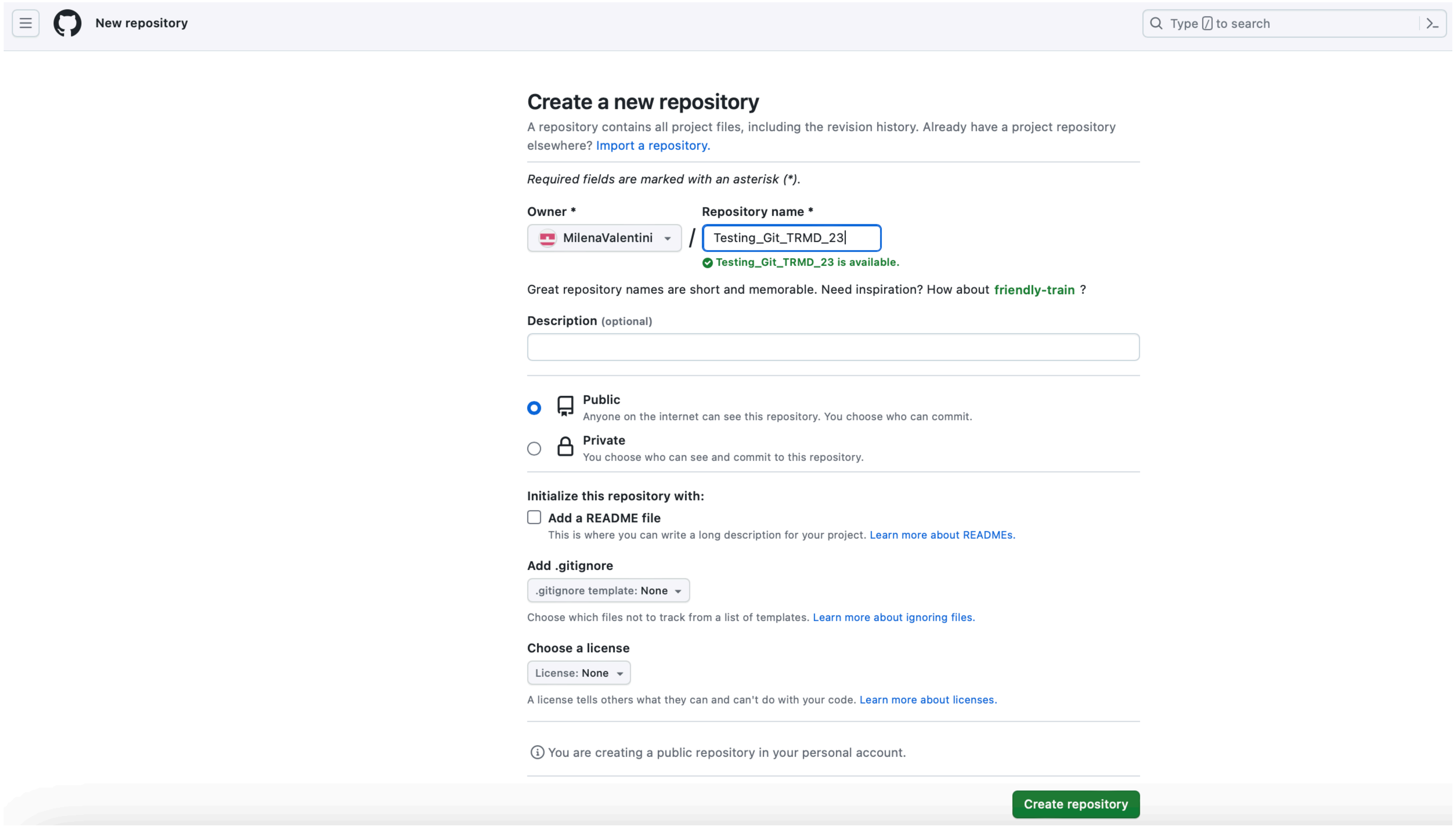


Git: from local to remote repository



Create a new, empty repository on Git

Git: from local to remote 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 * MilenaValentini / **Repository name *** Testing_Git_TRMD_23

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

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

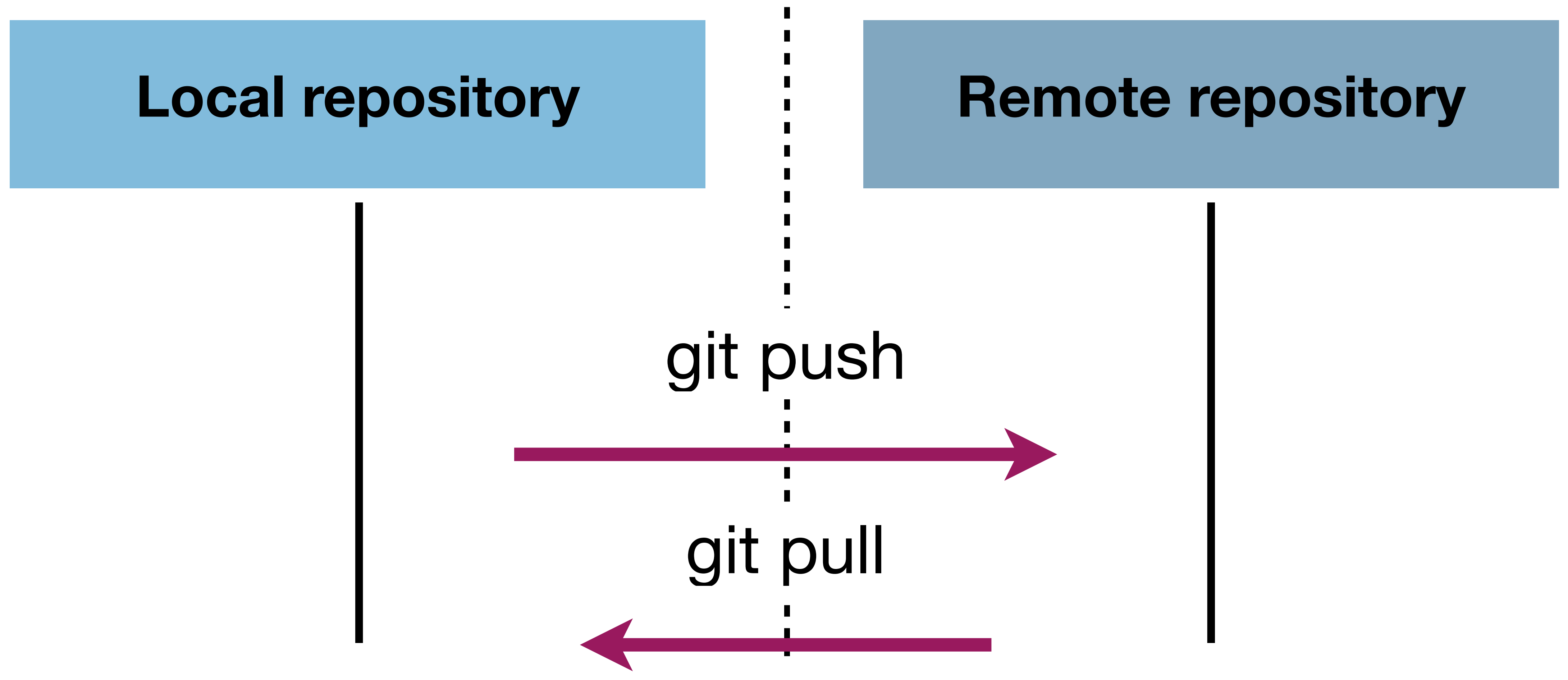
Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

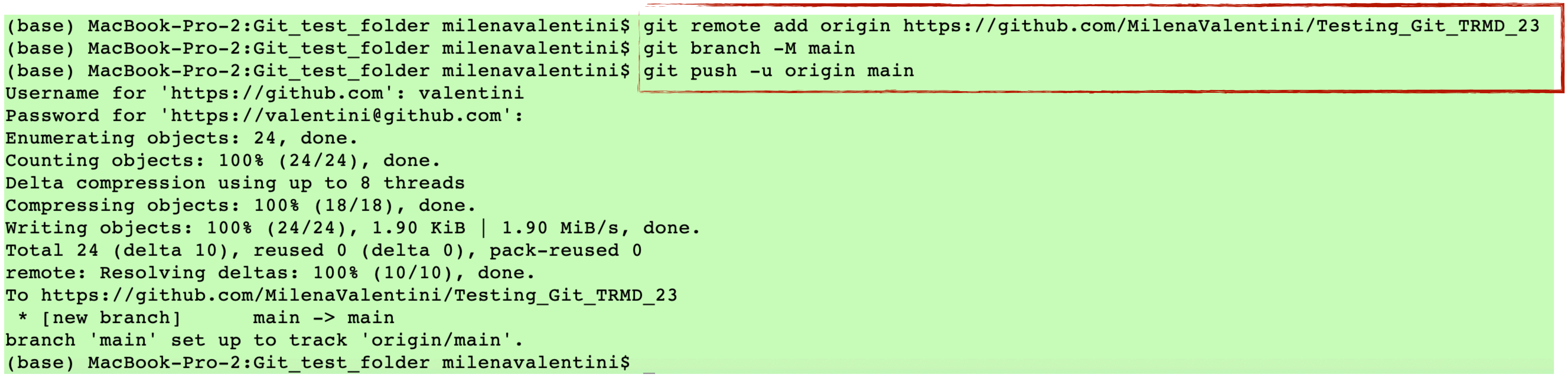
You are creating a public repository in your personal account.

Git: from local to remote repository



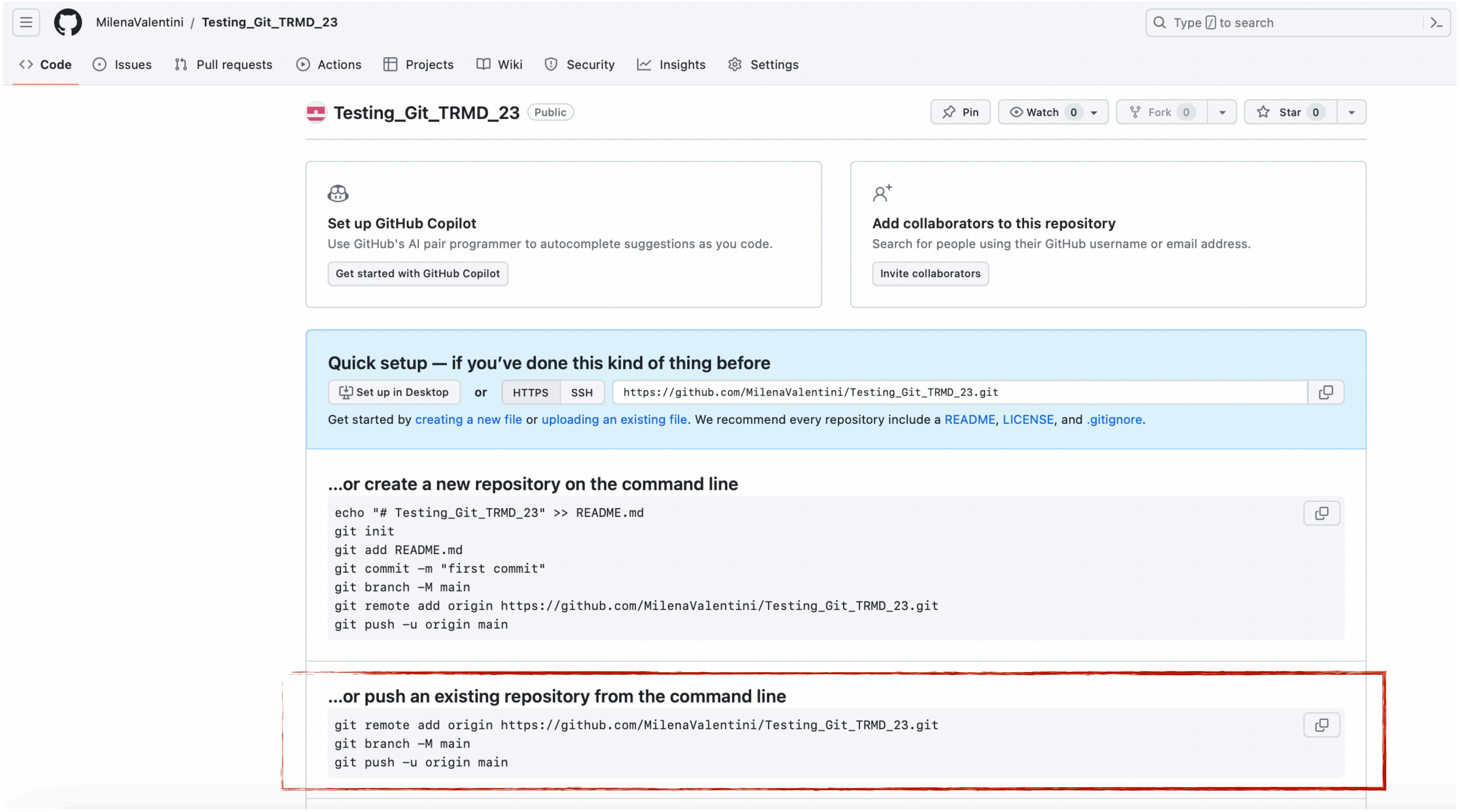
Create a new, empty repository on Git

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



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.

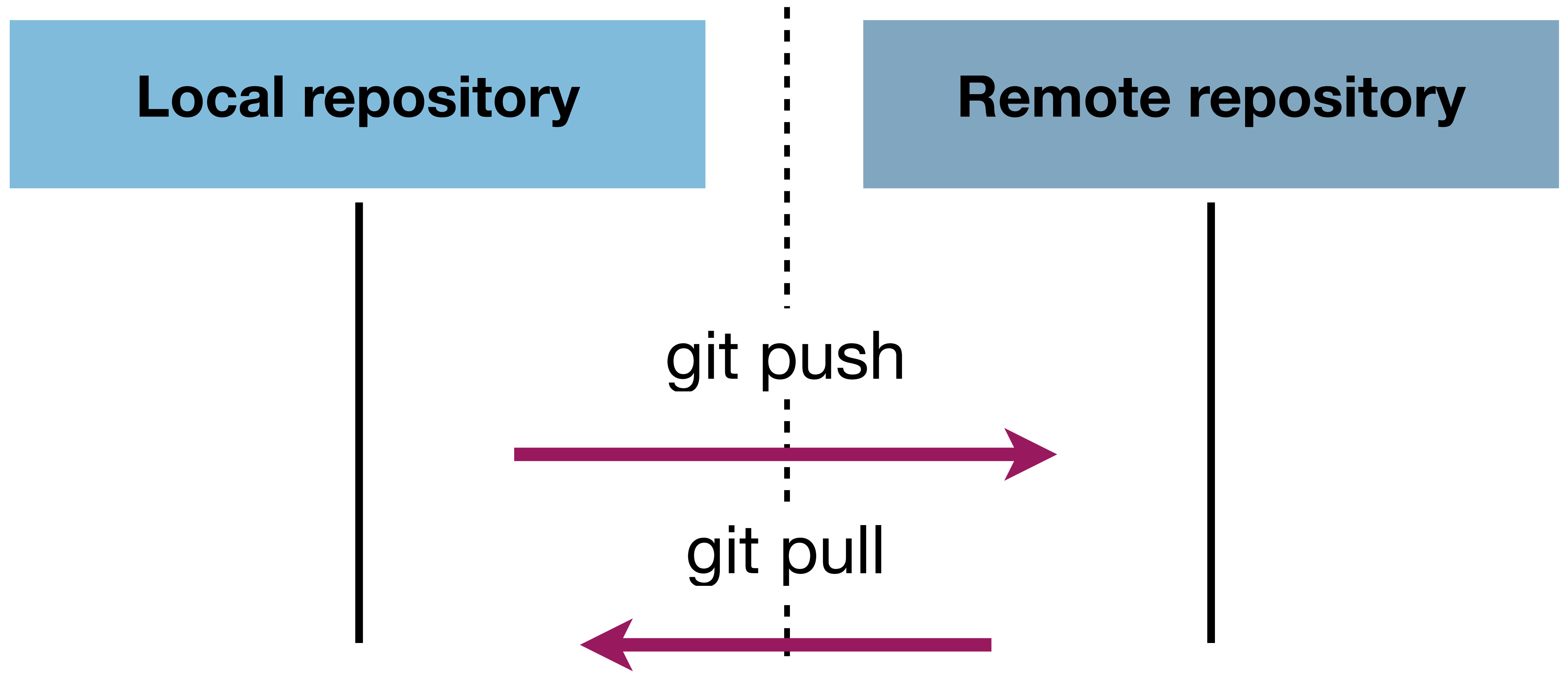
...or create a new repository on the command line

```
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

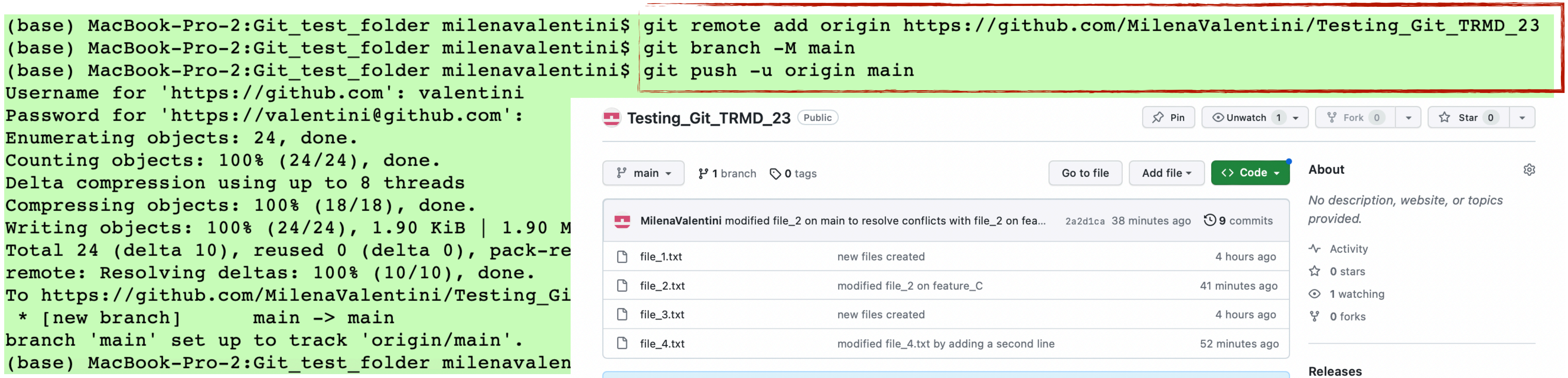
```
git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23.git
git branch -M main
git push -u origin main
```

Git: from local to remote repository

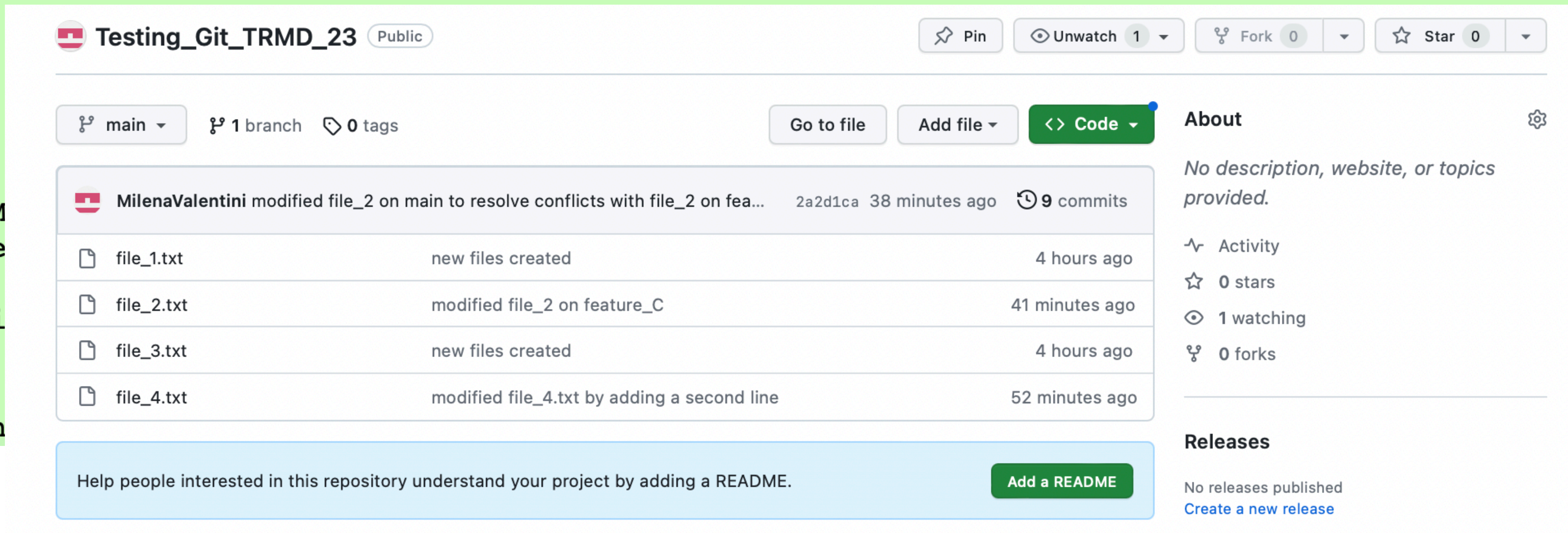


Create a new, empty repository on Git

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
```



No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

MilenaValentini modified file_2 on main to resolve conflicts with file_2 on fea... 2a2d1ca 38 minutes ago 9 commits

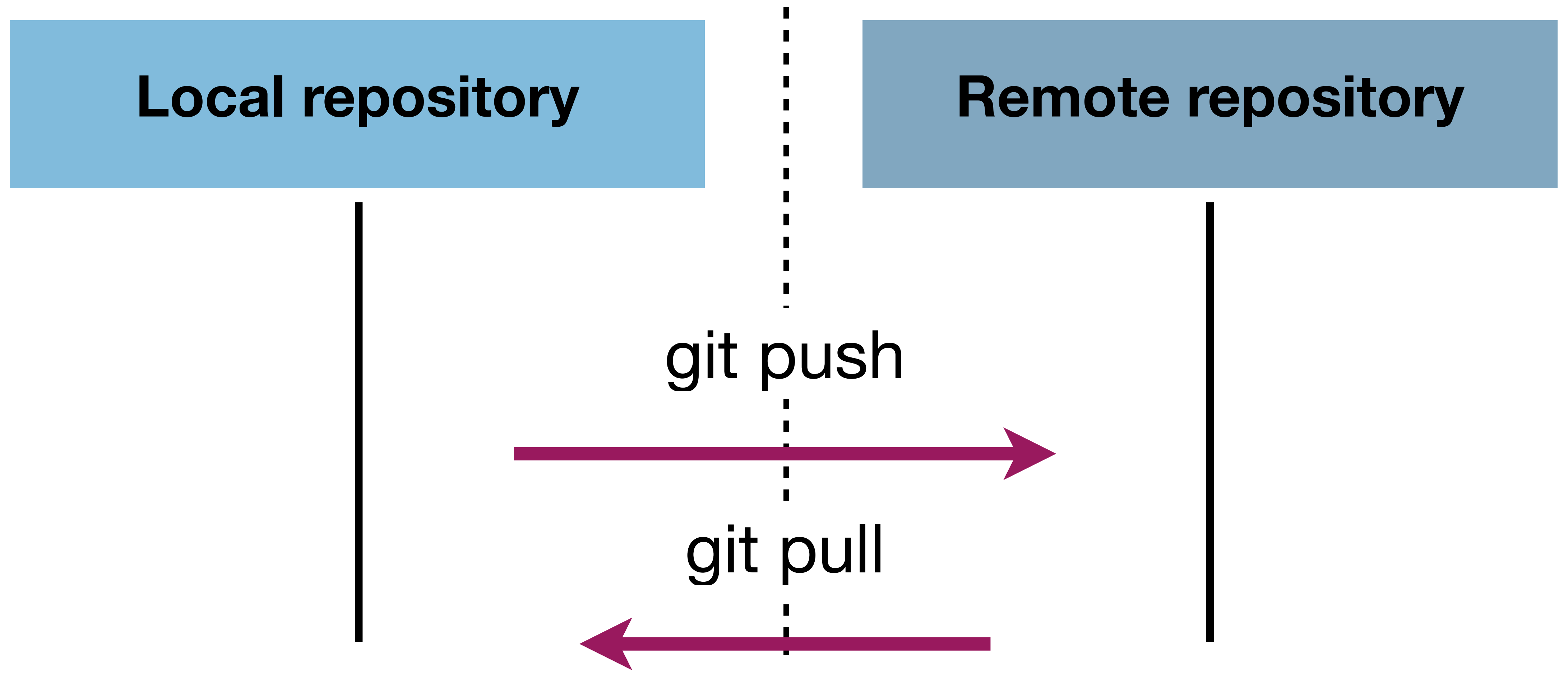
file_1.txt	new files created	4 hours ago
file_2.txt	modified file_2 on feature_C	41 minutes ago
file_3.txt	new files created	4 hours ago
file_4.txt	modified file_4.txt by adding a second line	52 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

Releases

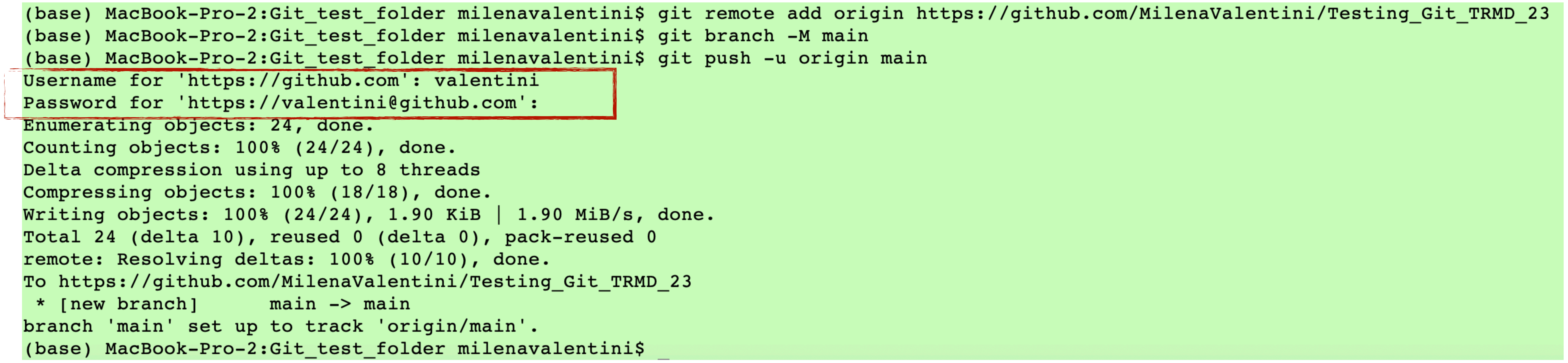
No releases published
Create a new release

Git: from local to remote repository



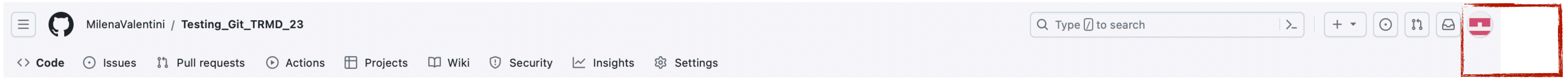
Create a new, empty repository on Git

In the main branch in the local repository:



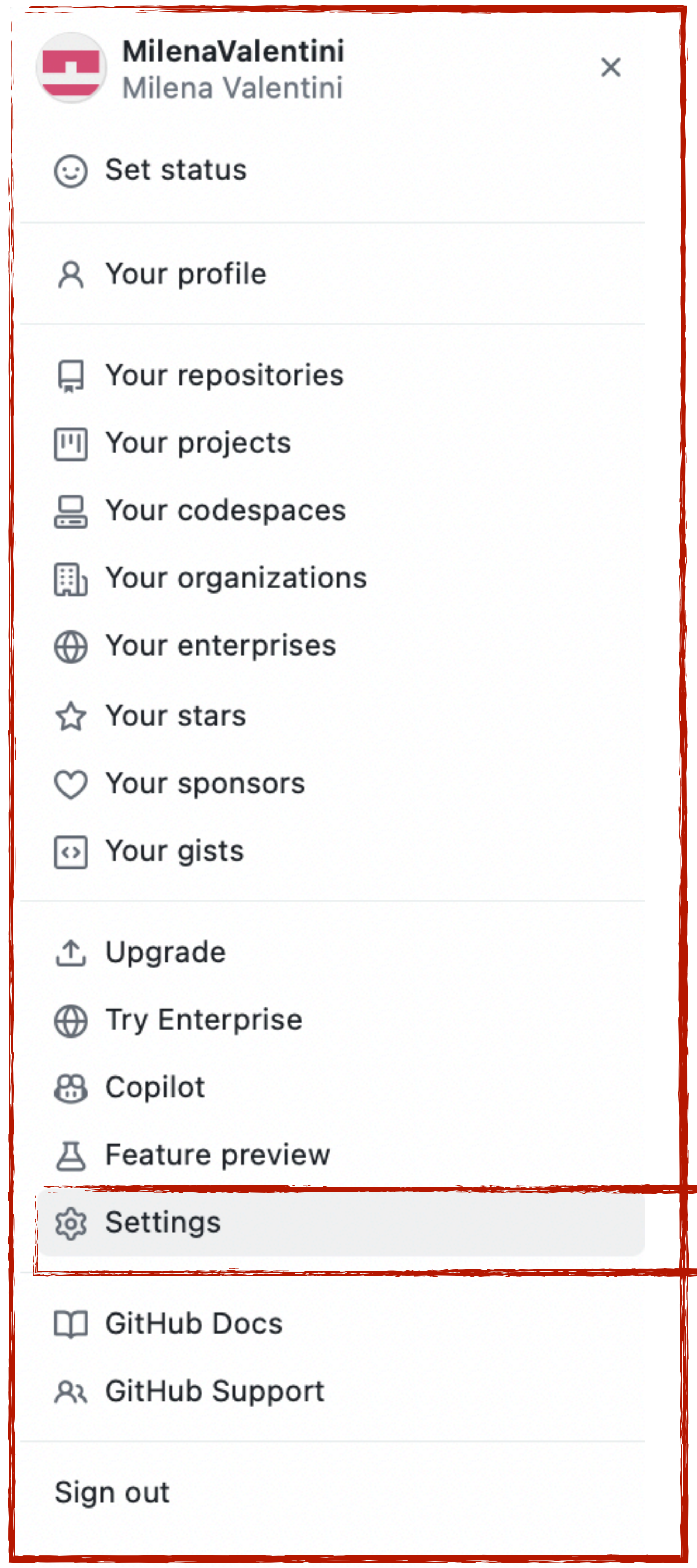
```
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git remote add origin https://github.com/MilenaValentini/Testing_Git_TRMD_23
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git branch -M main
(base) MacBook-Pro-2:Git_test_folder milenavalentini$ git push -u origin main
Username for 'https://github.com': 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


MilenaValentini / Testing_Git_TRMD_23

Type / to search

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights · Settings


MilenaValentini
Milena Valentini

Set status

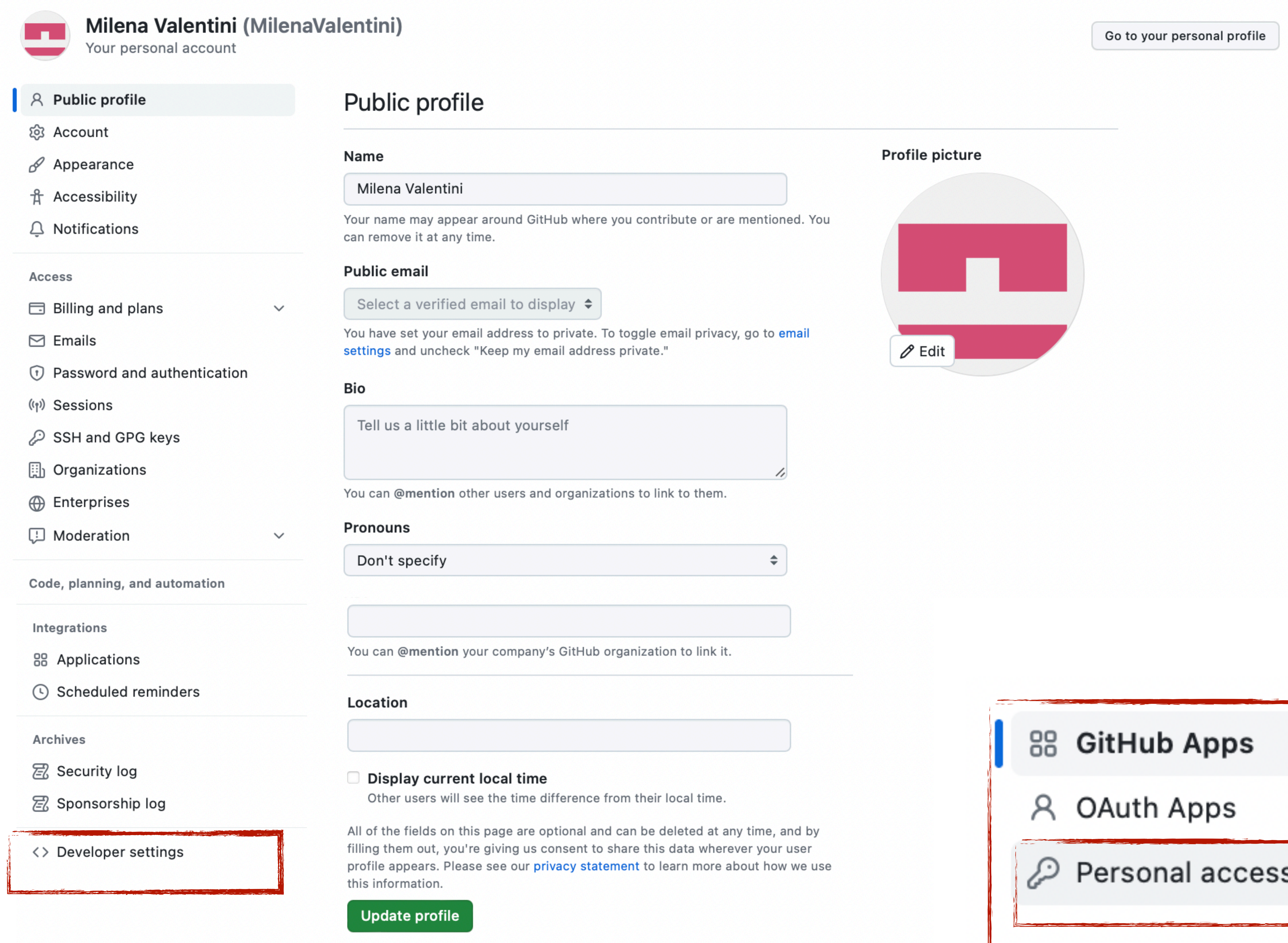
Your profile

Your repositories
Your projects
Your codespaces
Your organizations
Your enterprises
Your stars
Your sponsors
Your gists

Upgrade
Try Enterprise
Copilot
Feature preview
Settings

GitHub Docs
GitHub Support

Sign out


Milena Valentini (MilenaValentini)
Your personal account

Go to your personal profile

Public profile
Account
Appearance
Accessibility
Notifications

Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation

Code, planning, and automation

Integrations
Applications
Scheduled reminders

Archives
Security log
Sponsorship log

Developer settings

Public profile

Name
Milena Valentini
Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email
Select a verified email to display
You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."

Bio
Tell us a little bit about yourself
You can @mention other users and organizations to link to them.

Pronouns
Don't specify

You can @mention your company's GitHub organization to link it.

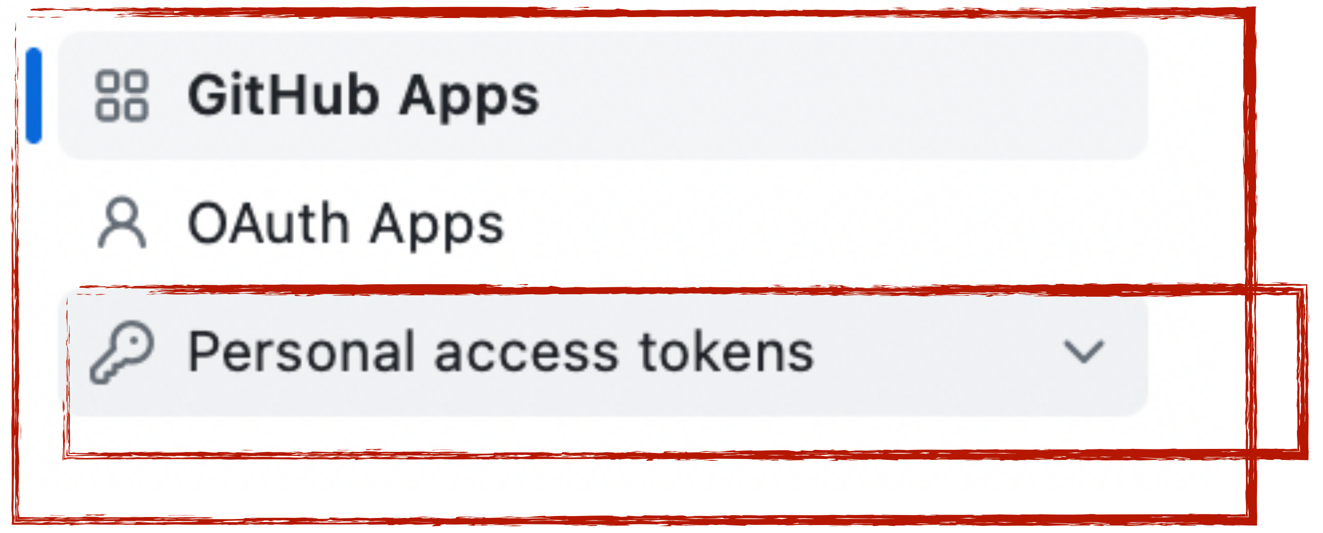
Location

Display current local time
Other users will see the time difference from their local time.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our privacy statement to learn more about how we use this information.

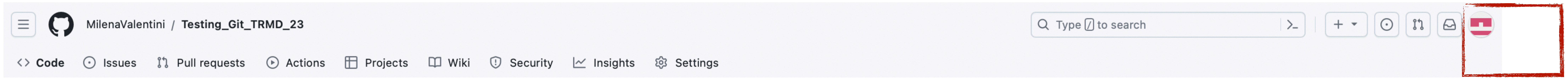
Update profile

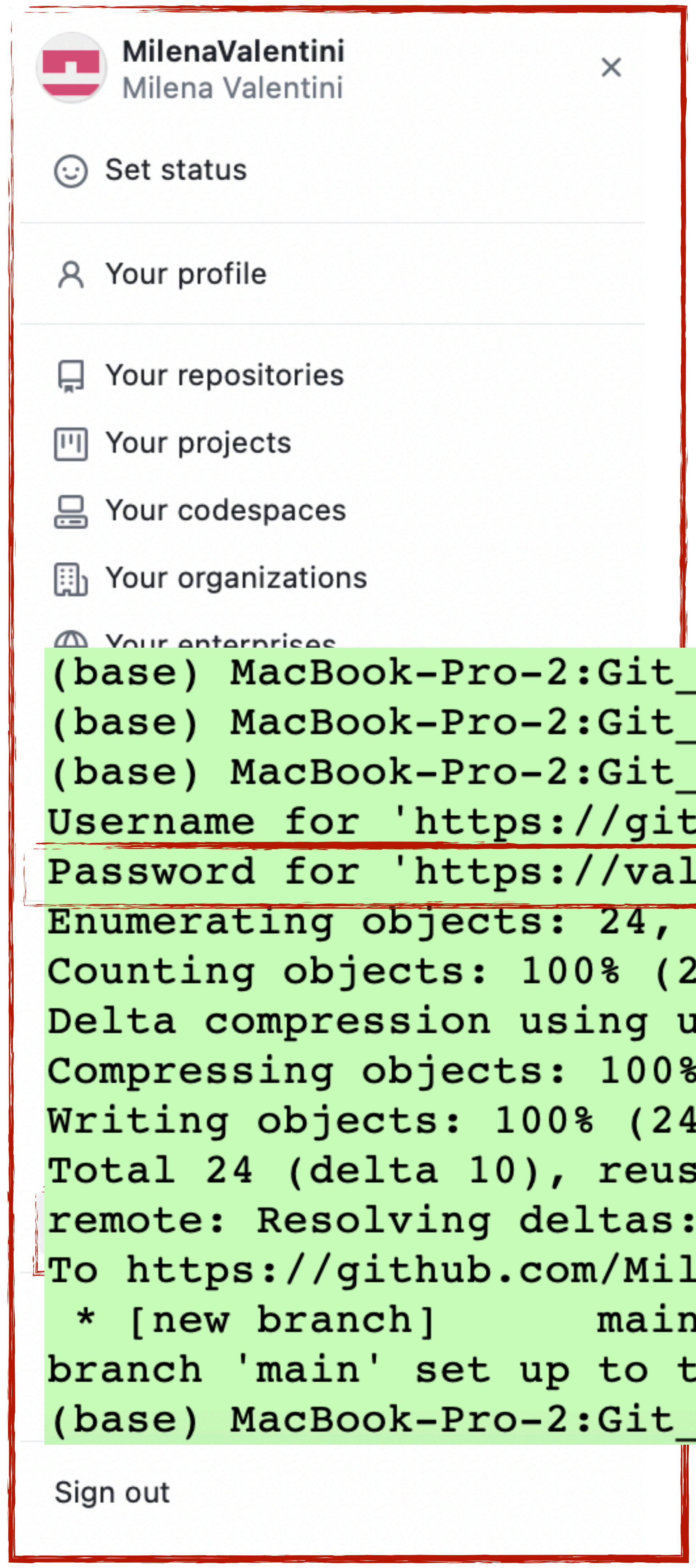
Profile picture
Edit

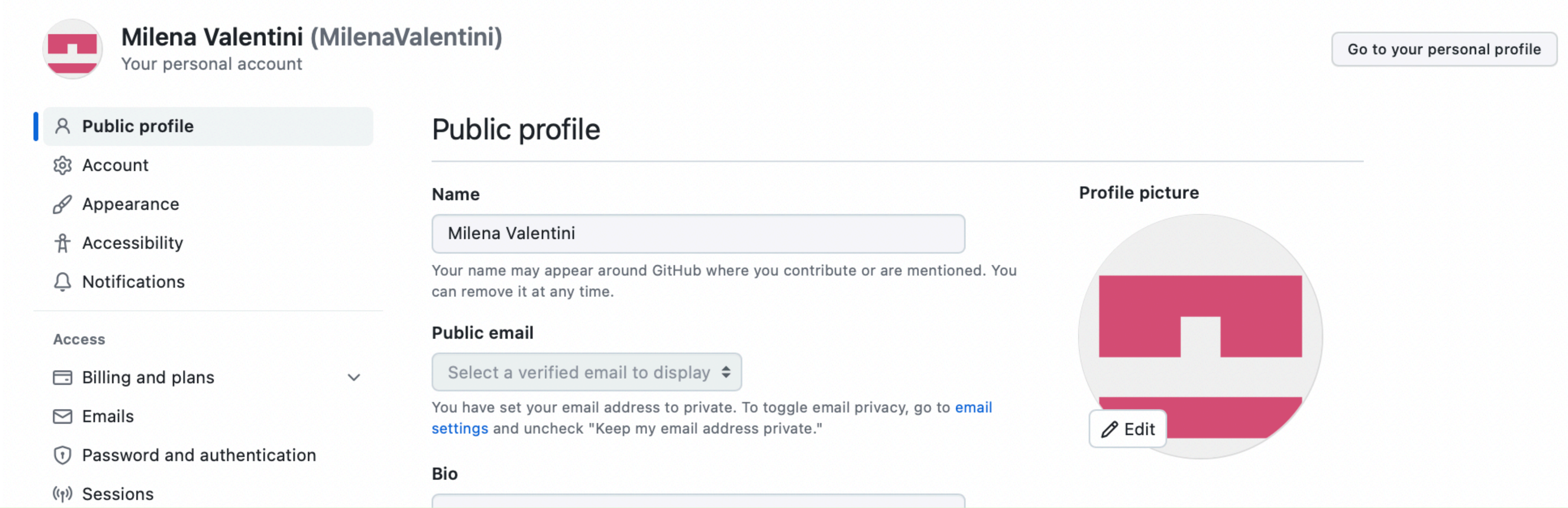

GitHub Apps
OAuth Apps
Personal access tokens

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

Git: from local to remote repository







Public profile

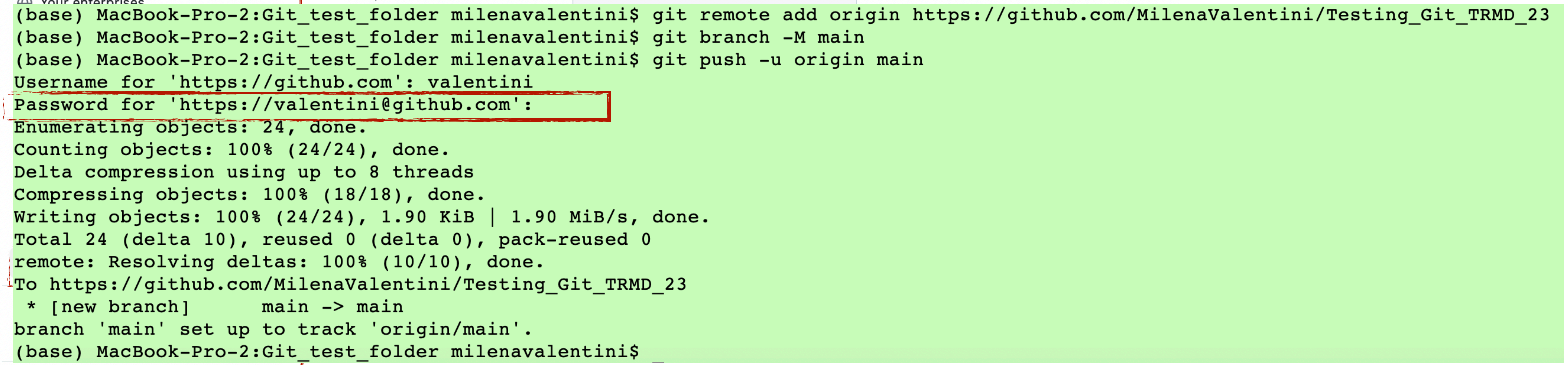
Name: Milena Valentini

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email: Select a verified email to display

You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."

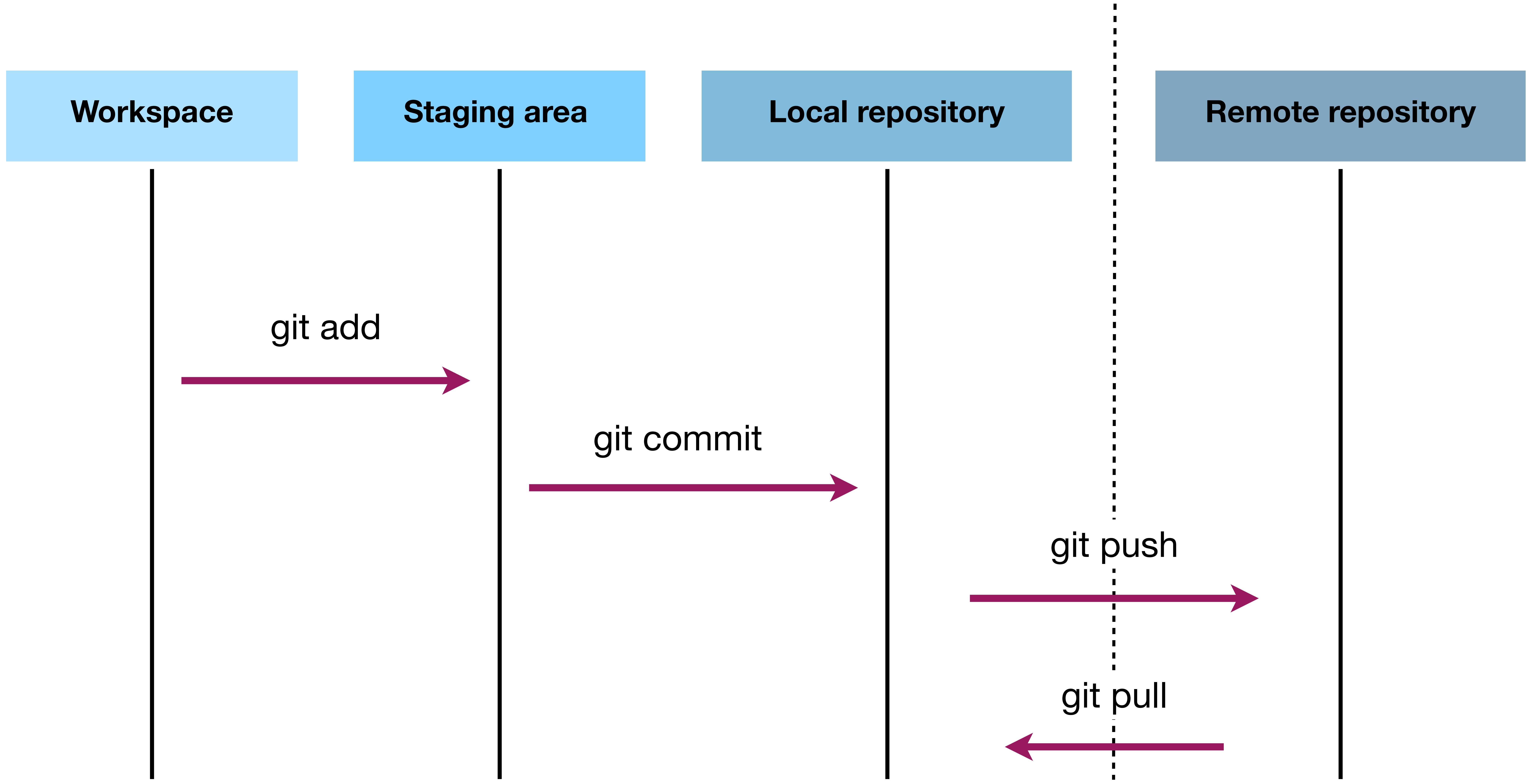
Bio



```
(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$ _
```

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

Git: from local to remote repository



Git: hands-on

Exercise 4

Create a new branch called ‘feature’. In the new branch, create a new file called my_script_2.py. Modify my_script_1.py to import matplotlib instead of scipy. Commit the changes.

Exercise 5

Verify that on branch master there is not my_script_2.py. Modify my_script_1.py on master, so that the import is now for healpy. Merge branch feature into master. Resolve the ensuing conflict by importing both healpy and matplotlib.

Exercise 6

Create an empty remote repository and set it to track the master branch of the local one (hint: follow instructions on GitHub website, the splash page when you create a new repository). Create the file my_script_3.py in your local repository and push the changes to the remote one. Then create my_script_4.py in the remote repository and pull the changes in your local one.

Useful commands:

git init
git status
git add
git commit
git log
git branch
git switch
git merge
git remote add (origin)
git push (-u origin)
git pull

Bash: intro

Bash is a scripting/command language very close to the Unix operative system

Useful resources: https://www.gnu.org/software/bash/manual/

http://mywiki.wooledge.org/BashGuide

https://www.tldp.org/LDP/abs/html/

Bash: intro

Many shells have scripting abilities: put multiple commands in a script and the shell executes them as if they were typed from the keyboard.

A command or a script that you can execute consists of one or more processes. The processes can be categorized into different broad groups. The main two are:

— interactive processes, and
— batch processed

Most shells offer additional programming constructs that extend the scripting feature into a programming language.

Bash: intro

Bash interprets a line of code as follows:

- Bash divides each line into words separated by a tab/space character
- The first word of the line is the name of the **command** to be execute
- All the remaining words are arguments of that command (options, filenames, etc.).

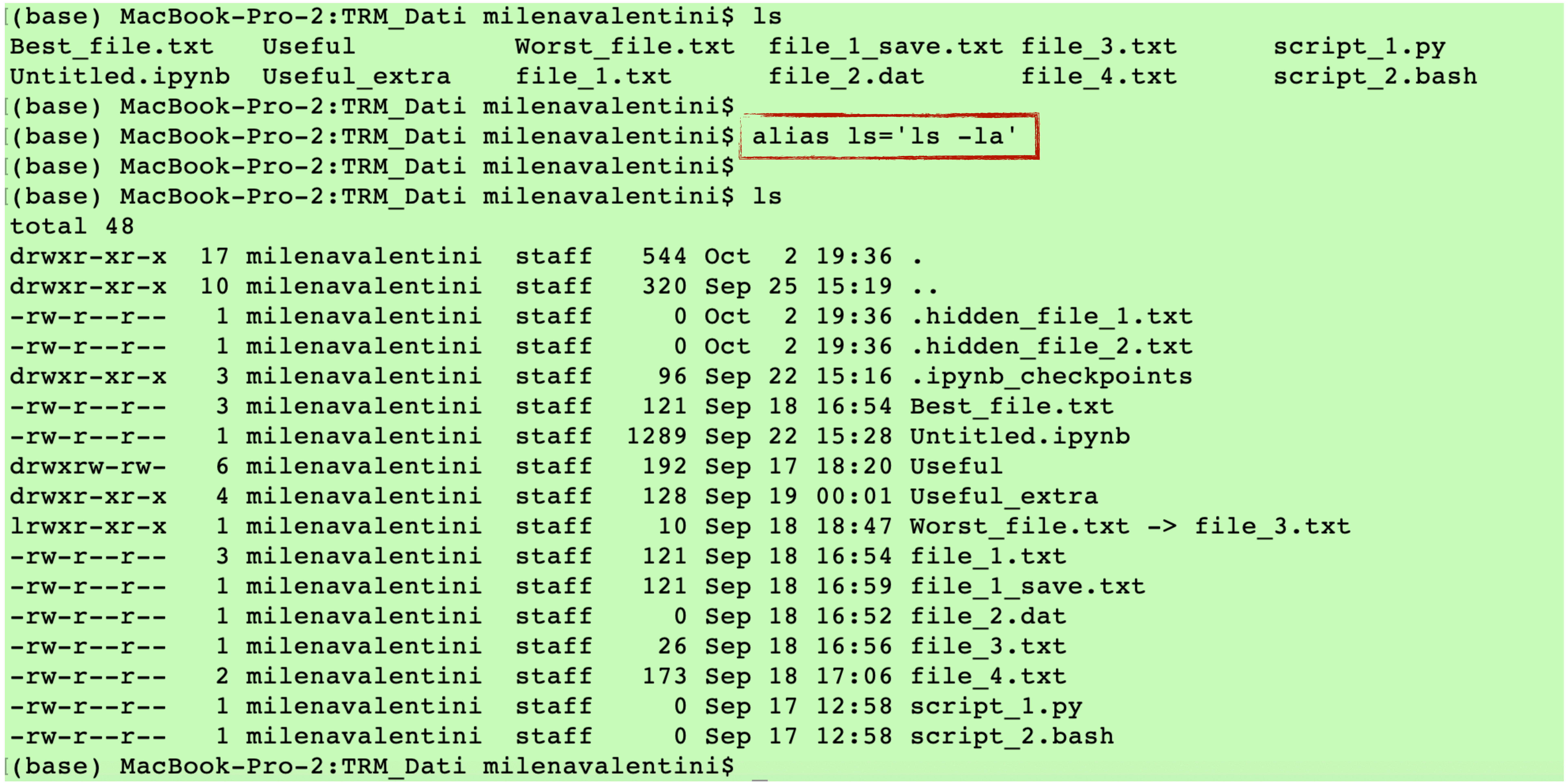
Different types of commands:

- aliases
- functions
- builtins
- keywords
- executables

Bash essentials

An **alias** is a way of shortening a command

example:



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls
Best_file.txt   Useful          Worst_file.txt  file_1_save.txt file_3.txt      script_1.py
Untitled.ipynb  Useful_extra    file_1.txt      file_2.dat      file_4.txt      script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ alias ls='ls -la'
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ ls
total 48
drwxr-xr-x  17 milenavalentini  staff   544 Oct  2 19:36 .
drwxr-xr-x  10 milenavalentini  staff   320 Sep 25 15:19 ..
-rw-r--r--   1 milenavalentini  staff     0 Oct  2 19:36 .hidden_file_1.txt
-rw-r--r--   1 milenavalentini  staff     0 Oct  2 19:36 .hidden_file_2.txt
drwxr-xr-x   3 milenavalentini  staff    96 Sep 22 15:16 .ipynb_checkpoints
-rw-r--r--   3 milenavalentini  staff   121 Sep 18 16:54 Best_file.txt
-rw-r--r--   1 milenavalentini  staff  1289 Sep 22 15:28 Untitled.ipynb
drwxrw-rw-   6 milenavalentini  staff   192 Sep 17 18:20 Useful
drwxr-xr-x   4 milenavalentini  staff   128 Sep 19 00:01 Useful_extra
lrwxr-xr-x   1 milenavalentini  staff    10 Sep 18 18:47 Worst_file.txt -> file_3.txt
-rw-r--r--   3 milenavalentini  staff   121 Sep 18 16:54 file_1.txt
-rw-r--r--   1 milenavalentini  staff   121 Sep 18 16:59 file_1_save.txt
-rw-r--r--   1 milenavalentini  staff     0 Sep 18 16:52 file_2.dat
-rw-r--r--   1 milenavalentini  staff    26 Sep 18 16:56 file_3.txt
-rw-r--r--   2 milenavalentini  staff   173 Sep 18 17:06 file_4.txt
-rw-r--r--   1 milenavalentini  staff     0 Sep 17 12:58 script_1.py
-rw-r--r--   1 milenavalentini  staff     0 Sep 17 12:58 script_2.bash
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ _
```

Bash essentials

An **alias** is a way of shortening a command

example:

```
alias ls='ls -la'
```

- Word mapped into a string
- What is useful for: to change the default behaviour of a command
- What is not aimed at: complex tasks (functions should be preferred)
- Not used in scripts, only in the interactive mode

Bash essentials

A **function** is a set of shell commands

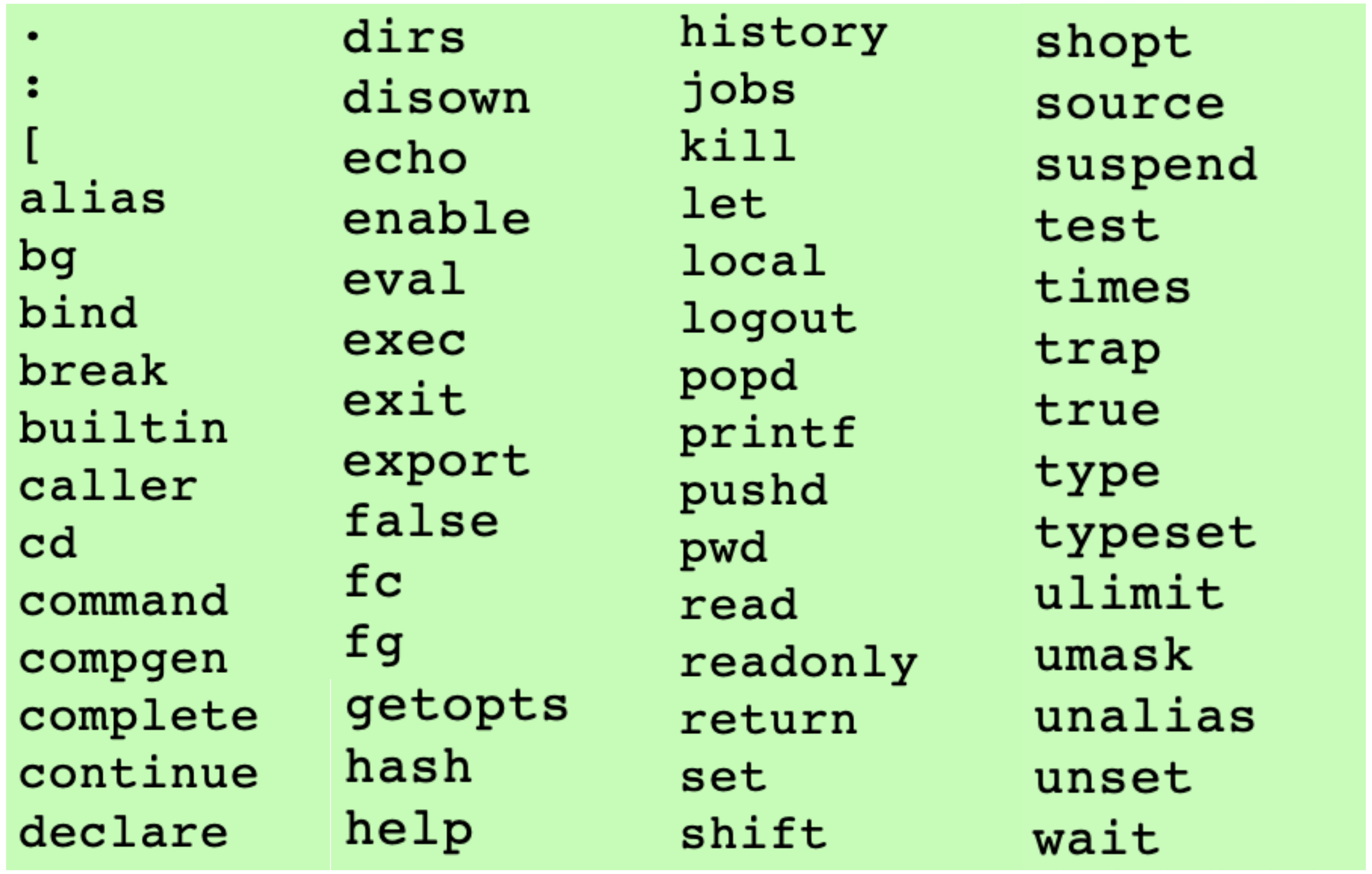
example:

```
JUST_A_SECOND=1

fun ()
{ # A somewhat more complex function.
        local i=0
        REPEATS=30
        sleep $JUST_A_SECOND # Hey, wait a second!
        while [ $i -lt $REPEATS ]
        do
                echo '<----------FUN---------->'
                let 'i+=1'
        done
}

fun
exit $?
```

- When a function is called, commands in it are executed
- A function behaves like a (short) script
- It can be used in scripts
- Functions can take arguments and create local variables

Bash essentials

Builtins are basic commands that bash has built in it



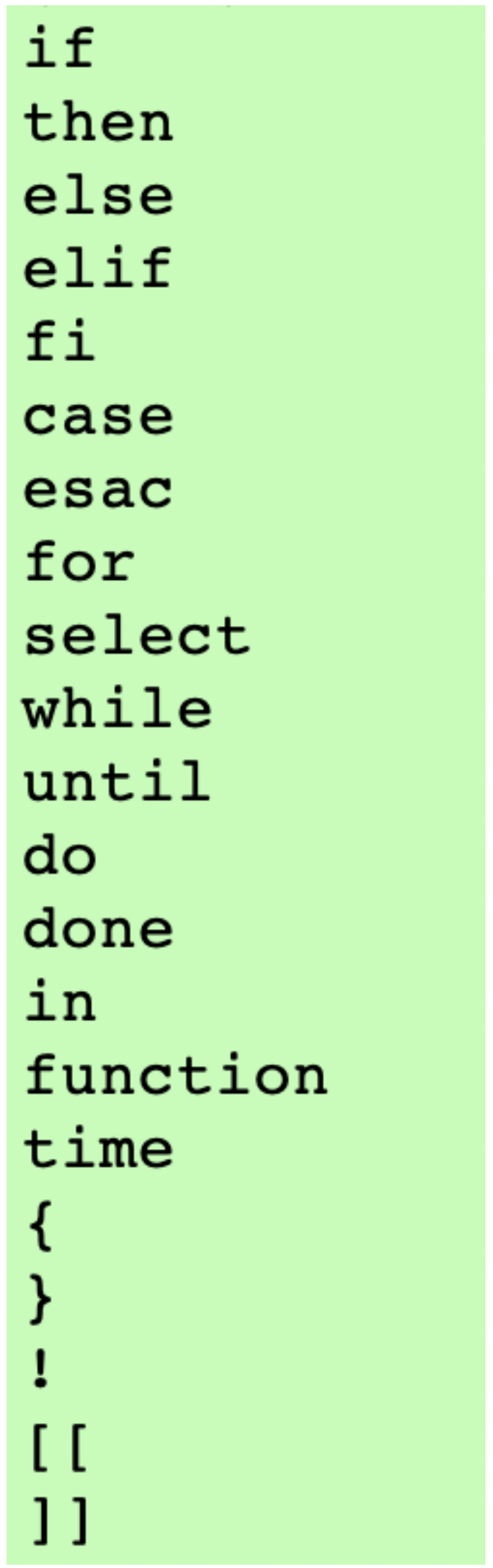
.	dirs	history	shopt
:	disown	jobs	source
[	echo	kill	suspend
alias	enable	let	test
bg	eval	local	times
bind	exec	logout	trap
break	exit	popd	true
builtin	export	printf	type
caller	false	pushd	typeset
cd	fc	pwd	ulimit
command	fg	read	umask
compgen	getopts	readonly	unalias
complete	hash	return	unset
continue	help	set	wait
declare		shift	

- They are simple functions already provided to the users
- Include builtin keywords

Bash essentials

Keywords are reserved words that have a special meaning to the shell.

- They are used to begin and end the shell command
- Similar to builtins
- Keywords are recognised when unquoted
- They are interpreted in a special way



```
if
then
else
elif
fi
case
esac
for
select
while
until
do
done
in
function
time
{
}
!
[[
]]
```

Bash essentials

Executables are external commands or applications

- They are usually invoked by their name
- Bash has to be able to find them

- If a command is not an alias, a function, a builtin nor a keyword and it is specified without a file path, bash searches through the directories listed in the environmental variable PATH

- Directories are scanned from left to right and the first executable that matches is run

Bash scripting

Bash scripts are sequences of bash commands in a file

The first line of a script should be reserved for an interpreter directive also called *hashbang* or *shebang*. This is used when the kernel executes a non-binary file. Use one of the two following alternatives

- `#!/bin/bash`
- `#!/usr/bin/env bash`

Bash scripting

- Script execution requires the script has “execute” permissions

- The script can be executed in either of the following ways:

```
bash scriptname                    # the shebang is considered a comment
./scriptname                       # the shebang is used
```

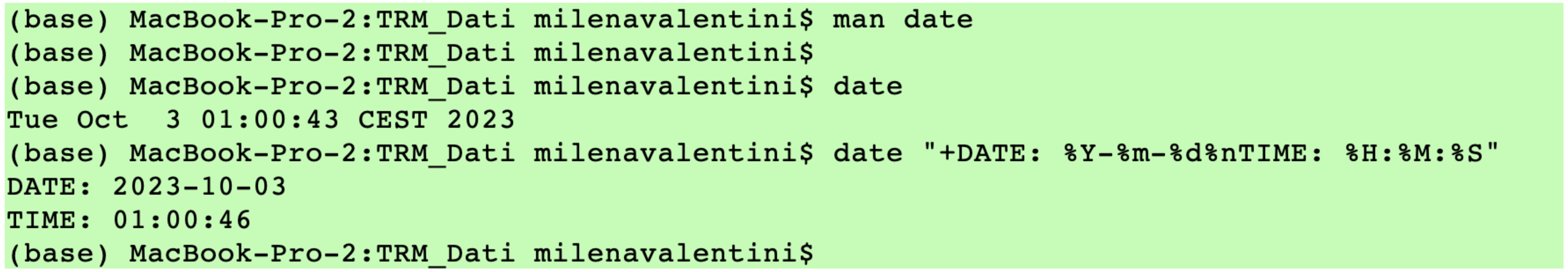
- The script can be made available as a command:
  - move the script to /usr/local/bin , making it available to all users as a system wide executable. The script could then be invoked by simply typing “scriptname” [ENTER] from the command-line.
  - Or include the directory containing the script in the user's $PATH

Bash scripting

- Write a bash script that:
  - Says “Hello!”
  - Displays date and time
  - Saves this information in a file
- Make the script executable and execute it
- Make the script available as a command

Bash scripting

- ✦ Write a bash script that:
  - Says “Hello!”
  - Displays date and time
  - Saves this information in a file
- ✦ Make the script executable and execute it
- ✦ Make the script available as a command



```
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ man date
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ date
Tue Oct  3 01:00:43 CEST 2023
(base) MacBook-Pro-2:TRM_Dati milenavalentini$ date "+DATE: %Y-%m-%d%nTIME: %H:%M:%S"
DATE: 2023-10-03
TIME: 01:00:46
(base) MacBook-Pro-2:TRM_Dati milenavalentini$
```

Shell scripting

Quoting

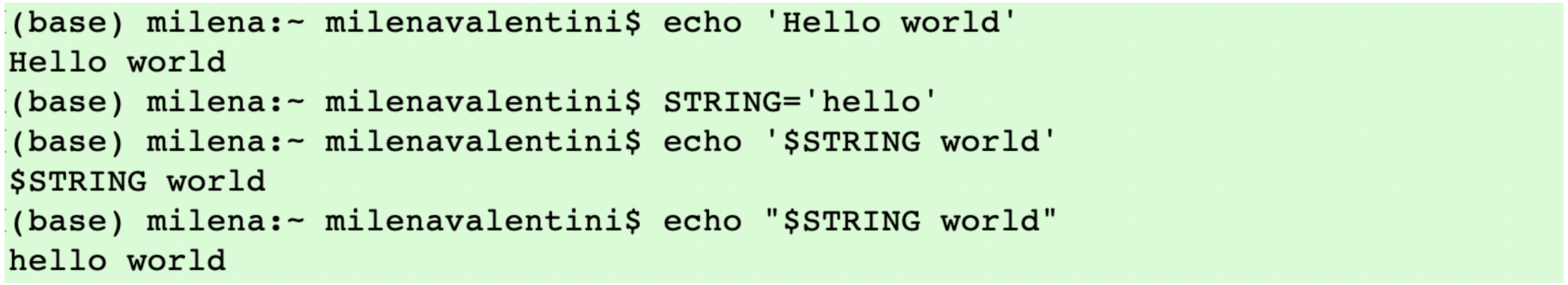
Single quotes: ‘string’ — what’s inside becomes a string

Shell scripting

Quoting

Single quotes: ‘string’	— what’s inside becomes a string
Double quotes: “ … ”	— needed to perform actions, e.g. substitutions that begin with $
	— they allow the shell to interpret dollar sign ($), backtick (`), backslash (\ ) and exclamation mark (!). These characters have special meaning when used with double quotes, and before display, they are evaluated

Shell scripting



```
(base) milena:~ milenavalentini$ echo 'Hello world'
Hello world
(base) milena:~ milenavalentini$ STRING='hello'
(base) milena:~ milenavalentini$ echo '$STRING world'
$STRING world
(base) milena:~ milenavalentini$ echo "$STRING world"
hello world
```

Shell scripting

Special characters

Single quotes: ‘string’

Double quotes: “ … ”

Whitespace: used to determine where words begin and end. The first word is the command name.

$: introduces different types of **expansion**

\ : put in front of a metacharacter removes its special meaning

: introduces a comment till the end of line

Shell scripting

Special characters

Single quotes: ‘string’

Double quotes: “ … ”

Whitespace: used to determine where words begin and end. The first word is the command name.

$: introduces different types of **expansion**

\ : put in front of a metacharacter removes its special meaning

: introduces a comment till the end of line

! : negate keyword, reverses a test or an exit status

= : assignment (white space not allowed on either side)

[[ ]] : testing keywords, allows to evaluate a conditional expression to determine if “true” or “false”

Shell scripting

Special characters

Single quotes: **‘**string**’**

Double quotes: **“** … **”**

Whitespace: used to determine where words begin and end. The first word is the command name.

$: introduces different types of **expansion**

\ : put in front of a metacharacter removes its special meaning

: introduces a comment till the end of line

! : negate keyword, reverses a test or an exit status

= : assignment (white space not allowed on either side)

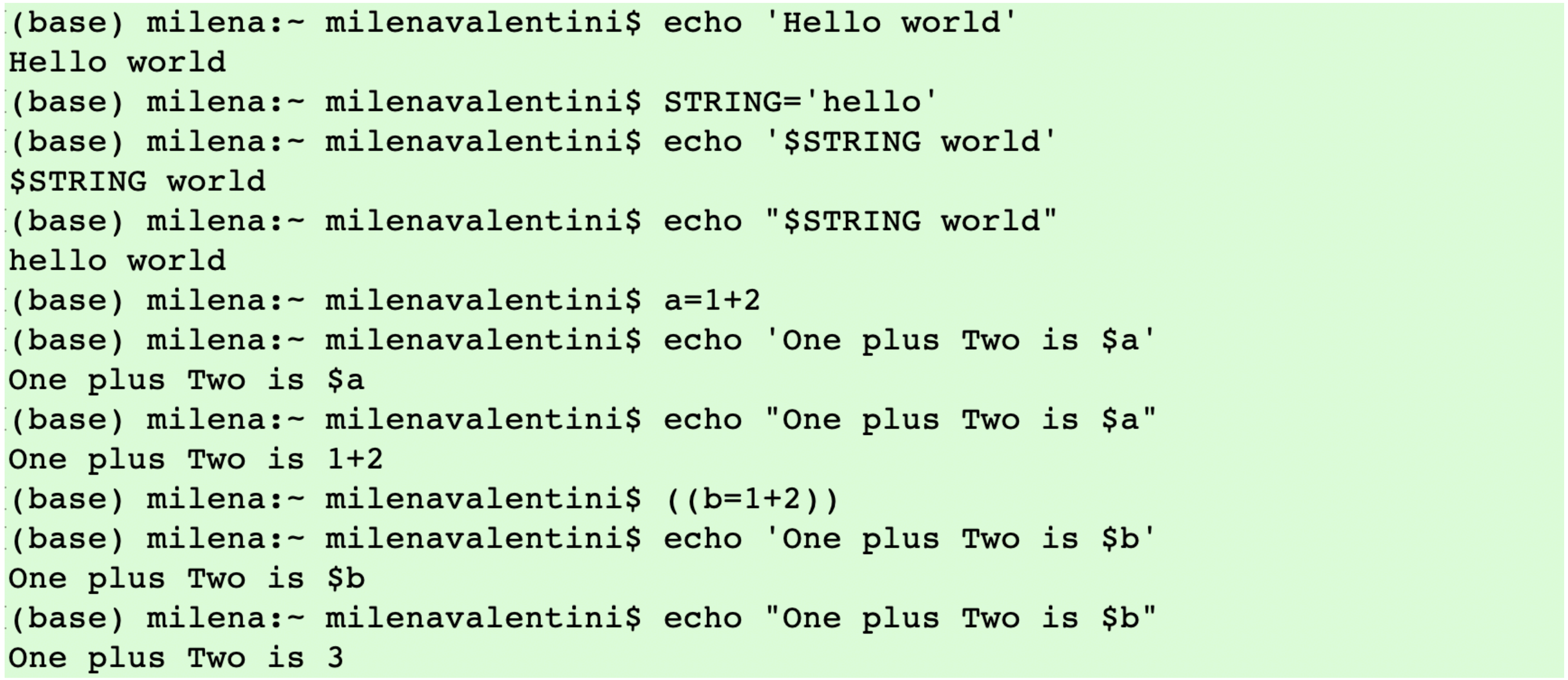
[[ ]] : testing keywords, allows to evaluate a conditional expression to determine if “true” or “false”

> >> < : redirection of a command’s output or input to a file

| : the pipeline sends the output from one command to the input of another command

; : command separator of multiple commands that are on the same line

Shell scripting

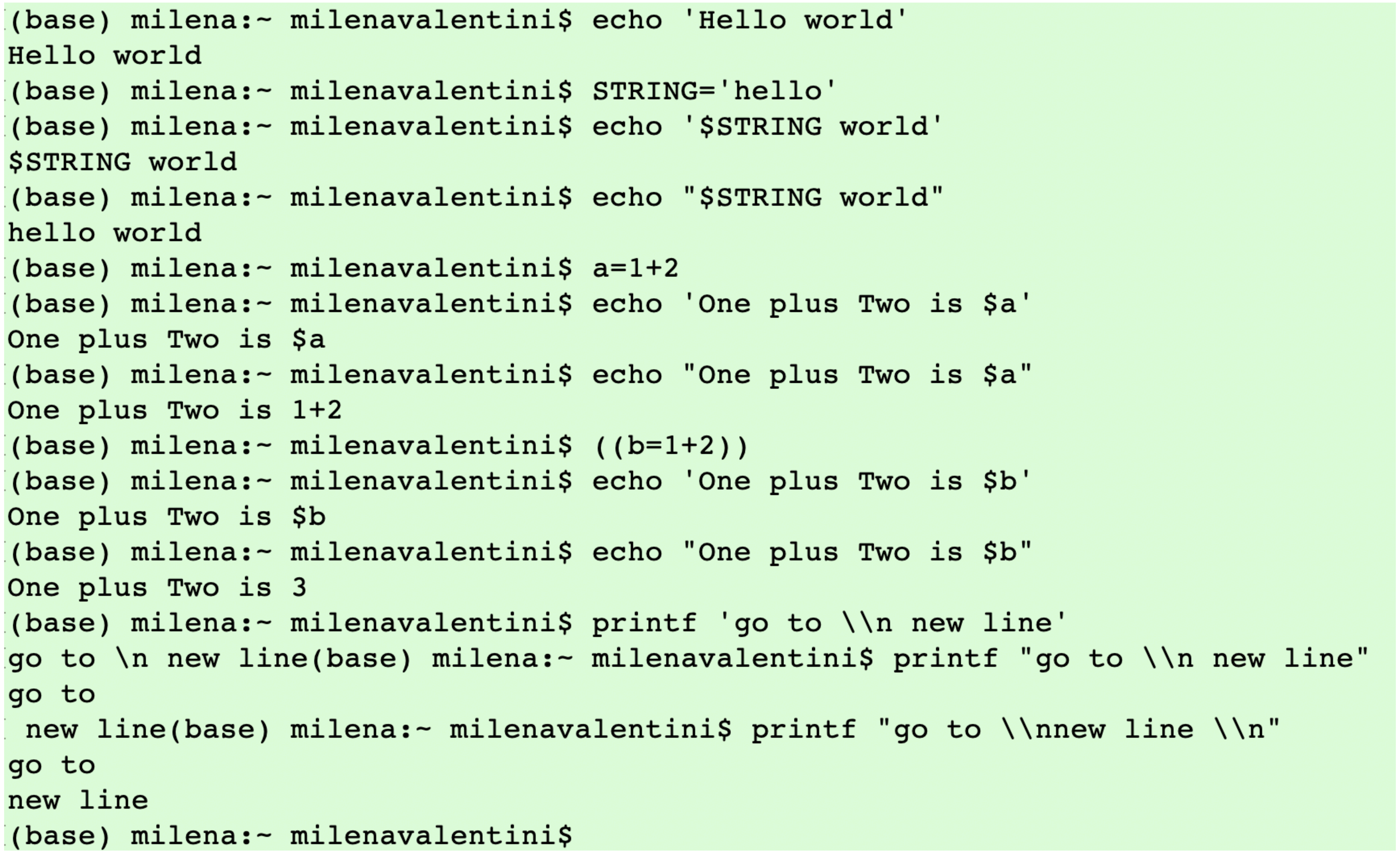


```
(base) milena:~ milenavalentini$ echo 'Hello world'
Hello world
(base) milena:~ milenavalentini$ STRING='hello'
(base) milena:~ milenavalentini$ echo '$STRING world'
$STRING world
(base) milena:~ milenavalentini$ echo "$STRING world"
hello world
(base) milena:~ milenavalentini$ a=1+2
(base) milena:~ milenavalentini$ echo 'One plus Two is $a'
One plus Two is $a
(base) milena:~ milenavalentini$ echo "One plus Two is $a"
One plus Two is 1+2
(base) milena:~ milenavalentini$ ((b=1+2))
(base) milena:~ milenavalentini$ echo 'One plus Two is $b'
One plus Two is $b
(base) milena:~ milenavalentini$ echo "One plus Two is $b"
One plus Two is 3
```

(( )) : within an arithmetic expression, **mathematical operators (+ - * /)** are used for calculations. Round brackets used for:

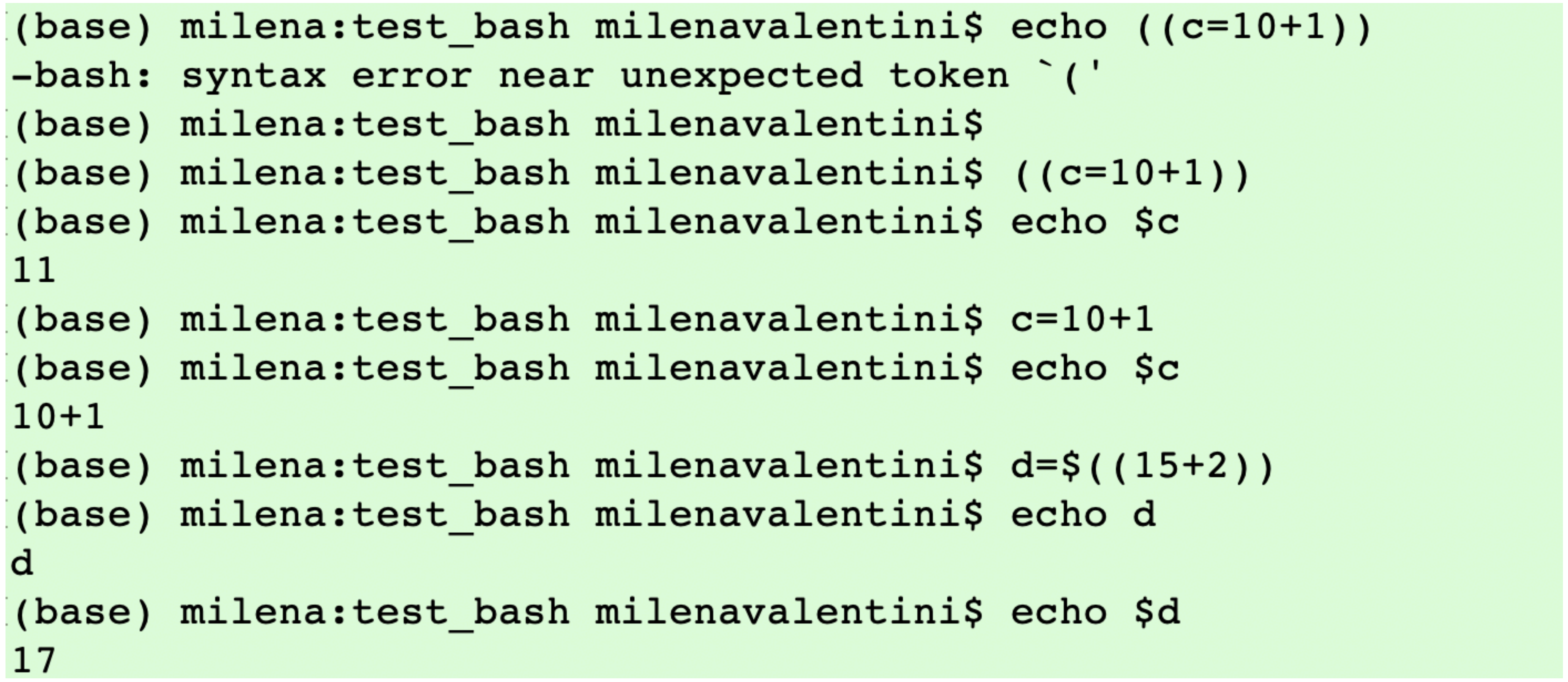
— variable assignments ((a=1+4))
— tests evaluation ((a<b))

Shell scripting



```
(base) milena:~ milenavalentini$ echo 'Hello world'
Hello world
(base) milena:~ milenavalentini$ STRING='hello'
(base) milena:~ milenavalentini$ echo '$STRING world'
$STRING world
(base) milena:~ milenavalentini$ echo "$STRING world"
hello world
(base) milena:~ milenavalentini$ a=1+2
(base) milena:~ milenavalentini$ echo 'One plus Two is $a'
One plus Two is $a
(base) milena:~ milenavalentini$ echo "One plus Two is $a"
One plus Two is 1+2
(base) milena:~ milenavalentini$ ((b=1+2))
(base) milena:~ milenavalentini$ echo 'One plus Two is $b'
One plus Two is $b
(base) milena:~ milenavalentini$ echo "One plus Two is $b"
One plus Two is 3
(base) milena:~ milenavalentini$ printf 'go to \\n new line'
go to \n new line(base) milena:~ milenavalentini$ printf "go to \\n new line"
go to
 new line(base) milena:~ milenavalentini$ printf "go to \\nnew line \\n"
go to
new line
(base) milena:~ milenavalentini$
```

Shell scripting



```
(base) milena:test_bash milenavalentini$ echo ((c=10+1))
-bash: syntax error near unexpected token `('
(base) milena:test_bash milenavalentini$
(base) milena:test_bash milenavalentini$ ((c=10+1))
(base) milena:test_bash milenavalentini$ echo $c
11
(base) milena:test_bash milenavalentini$ c=10+1
(base) milena:test_bash milenavalentini$ echo $c
10+1
(base) milena:test_bash milenavalentini$ d=$((15+2))
(base) milena:test_bash milenavalentini$ echo d
d
(base) milena:test_bash milenavalentini$ echo $d
17
```

$(( )) : comparable to (( )), but the expression is replaced with the result of its evaluation

$(...) : command substitution

Shell scripting

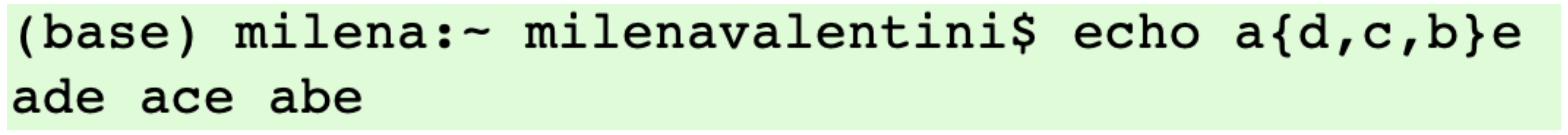
Special characters

$: introduces different types of **expansion**

Shell scripting

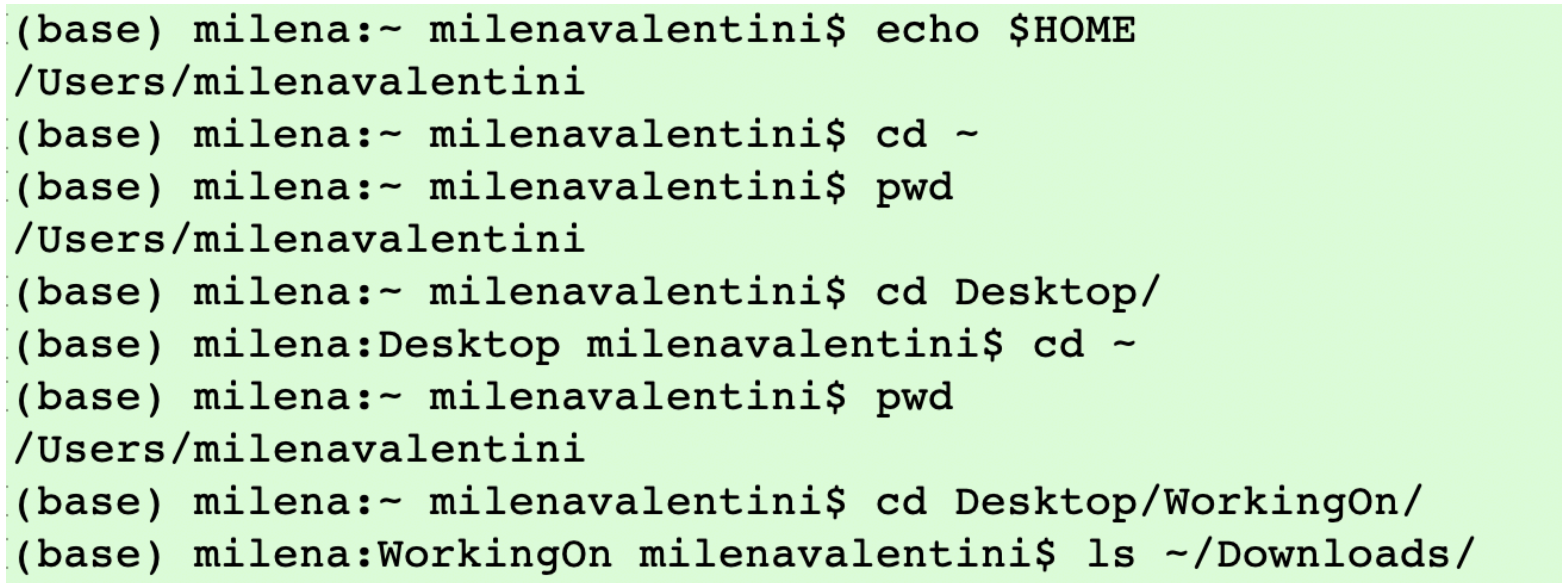
Expansions. They come in different flavours:

Brace expansion: mechanism by which arbitrary strings may be generated



```
(base) milena:~ milenavalentini$ echo a{d,c,b}e
ade ace abe
```

Tilde expansion: the unquoted ~ character is usually replaced with the value of the home shell variable.



```
(base) milena:~ milenavalentini$ echo $HOME
/Users/milenavalentini
(base) milena:~ milenavalentini$ cd ~
(base) milena:~ milenavalentini$ pwd
/Users/milenavalentini
(base) milena:~ milenavalentini$ cd Desktop/
(base) milena:Desktop milenavalentini$ cd ~
(base) milena:~ milenavalentini$ pwd
/Users/milenavalentini
(base) milena:~ milenavalentini$ cd Desktop/WorkingOn/
(base) milena:WorkingOn milenavalentini$ ls ~/Downloads/
```