Elyadata

GIT DOCKER & FASTAPI

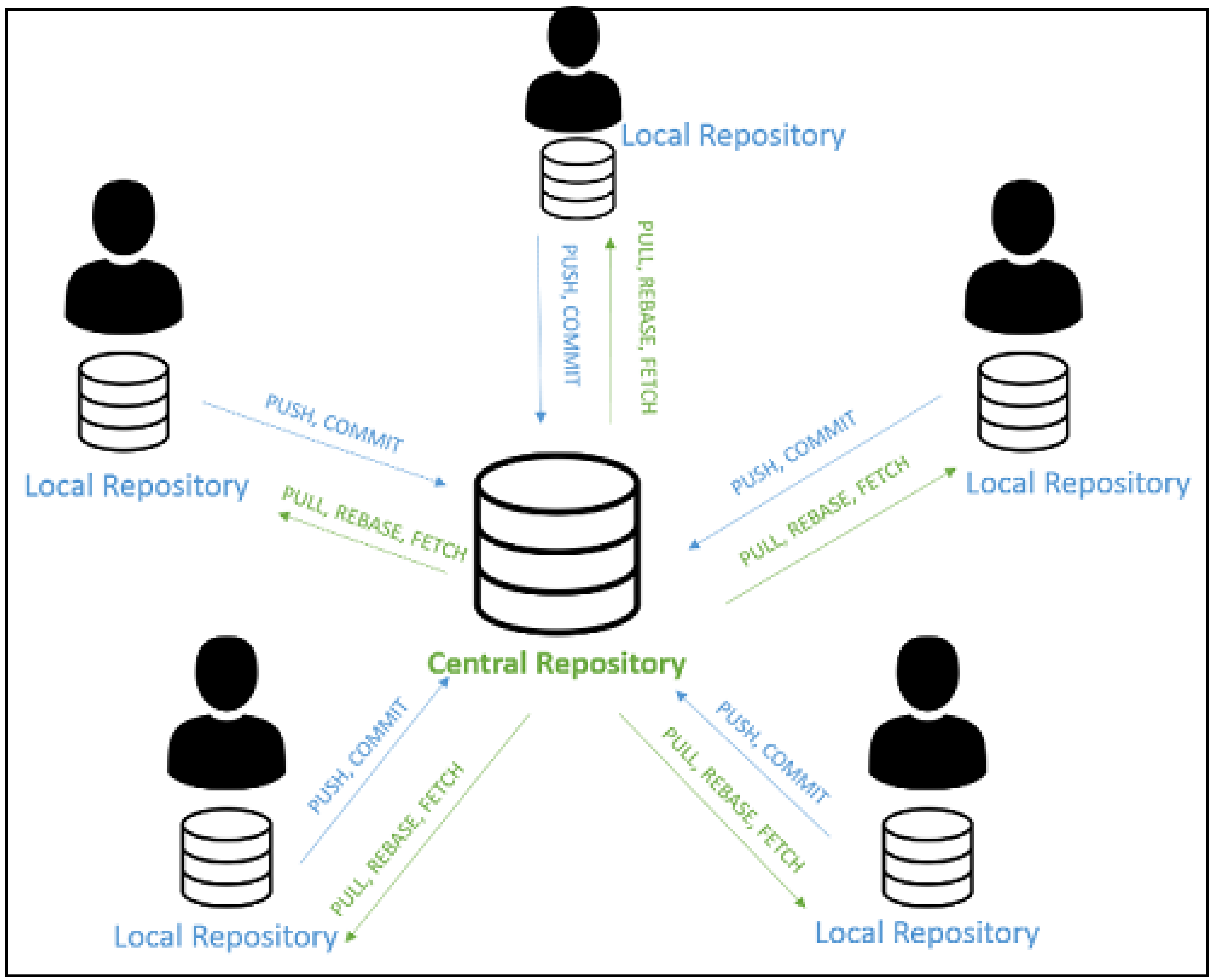
Presented by : Tasnim Charaa
Guided by : Imen Laouirine & Sirine Hammami

Agenda

Let’s start

USE CASE

Imagine a team of software developers working on a web application. Each developer need to has a copy of the project stored locally on their computer to make changes on the code and programming their own solution about the tasks. Using Git, they can create branches to work on specific tasks or features.



What is Git?

Git is a distributed version control which control system that allows you to track changes to your code, it is widely used in software development to manage and collaborate on code.

Version control : A system for managing changes to files over time, helps keep track of different versions of code and facilitates collaboration with others.

With Git, you can create branches to work on different features or bug fixes.

Git allows you to easily merge changes from different branches and track the history of you code.

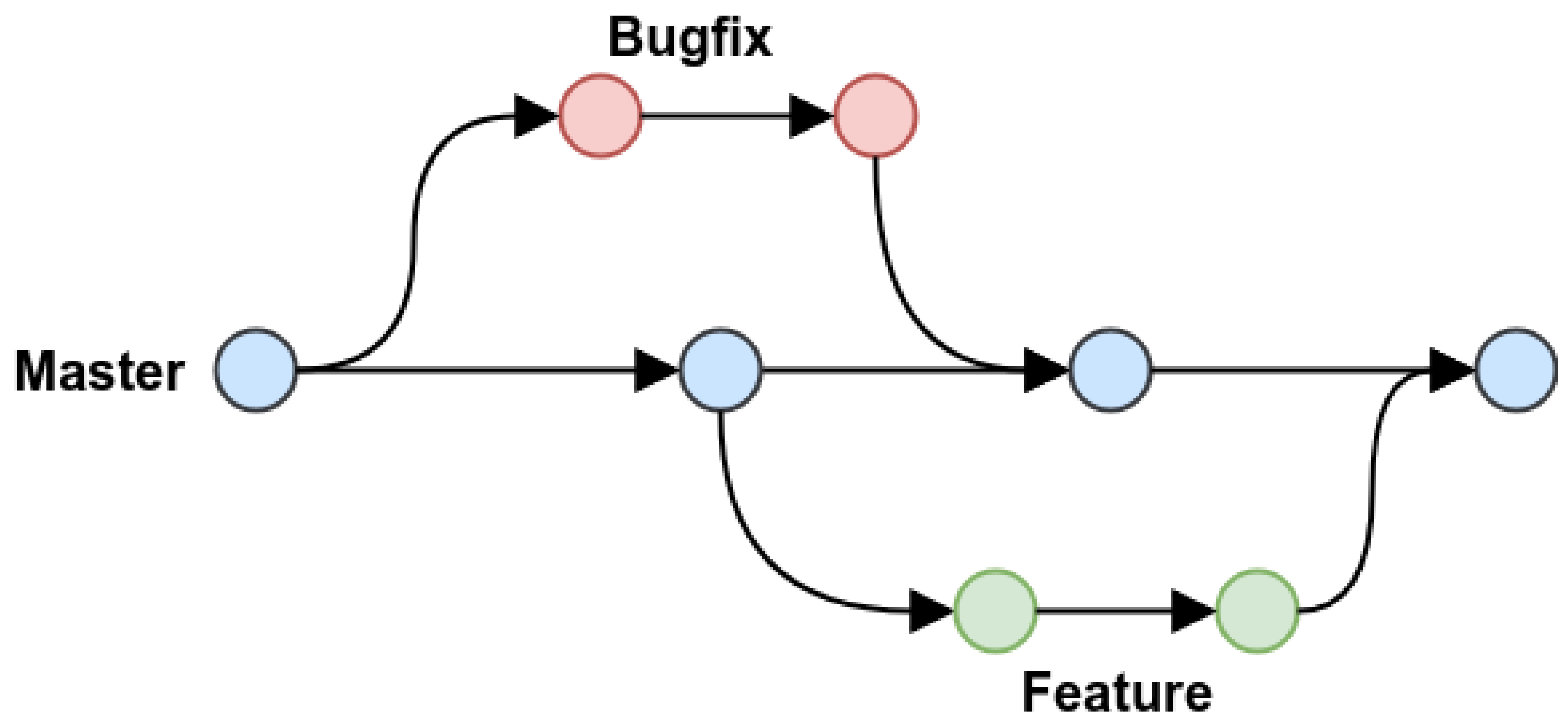
Git Basics

Key Concepts

Repository	A repository, or repo, is a storage location where all project files are stored.
Clone	Used to create a copy of a remote repository on your local machine.
Remote	Remotes are references to repositories on other servers When you clone a repository, you create a connection to its remote.
Staging Area (Index)	Before committing changes, Git allows you to selectively choose which modifications should be included in the next commit.
Branch	Is a parallel line of development within a repository. It allows you to work on different features

Branching

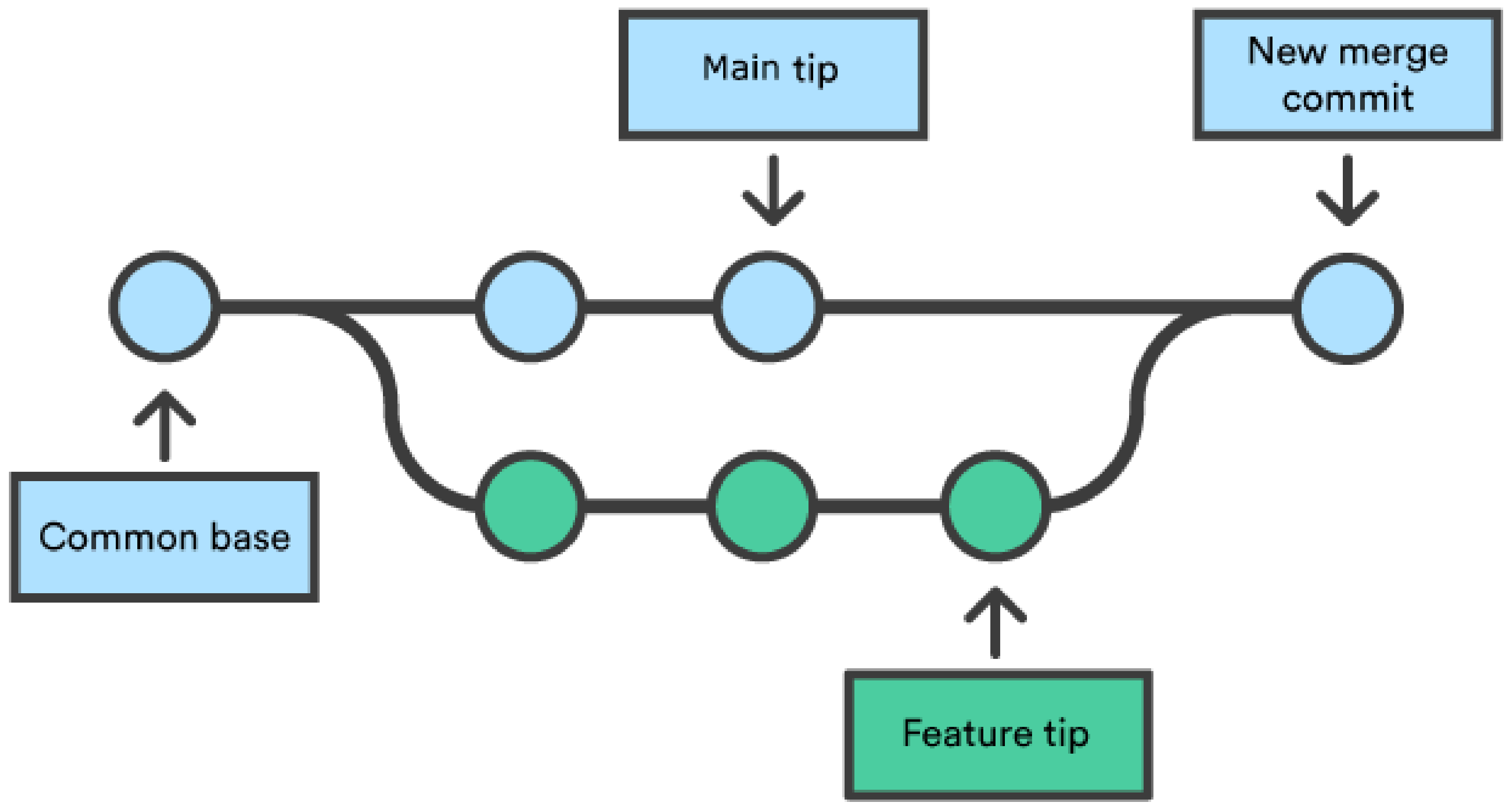
Branching in Git is a powerful feature that allows developers to work on different features or fixes simultaneously



Merging and Rebasing

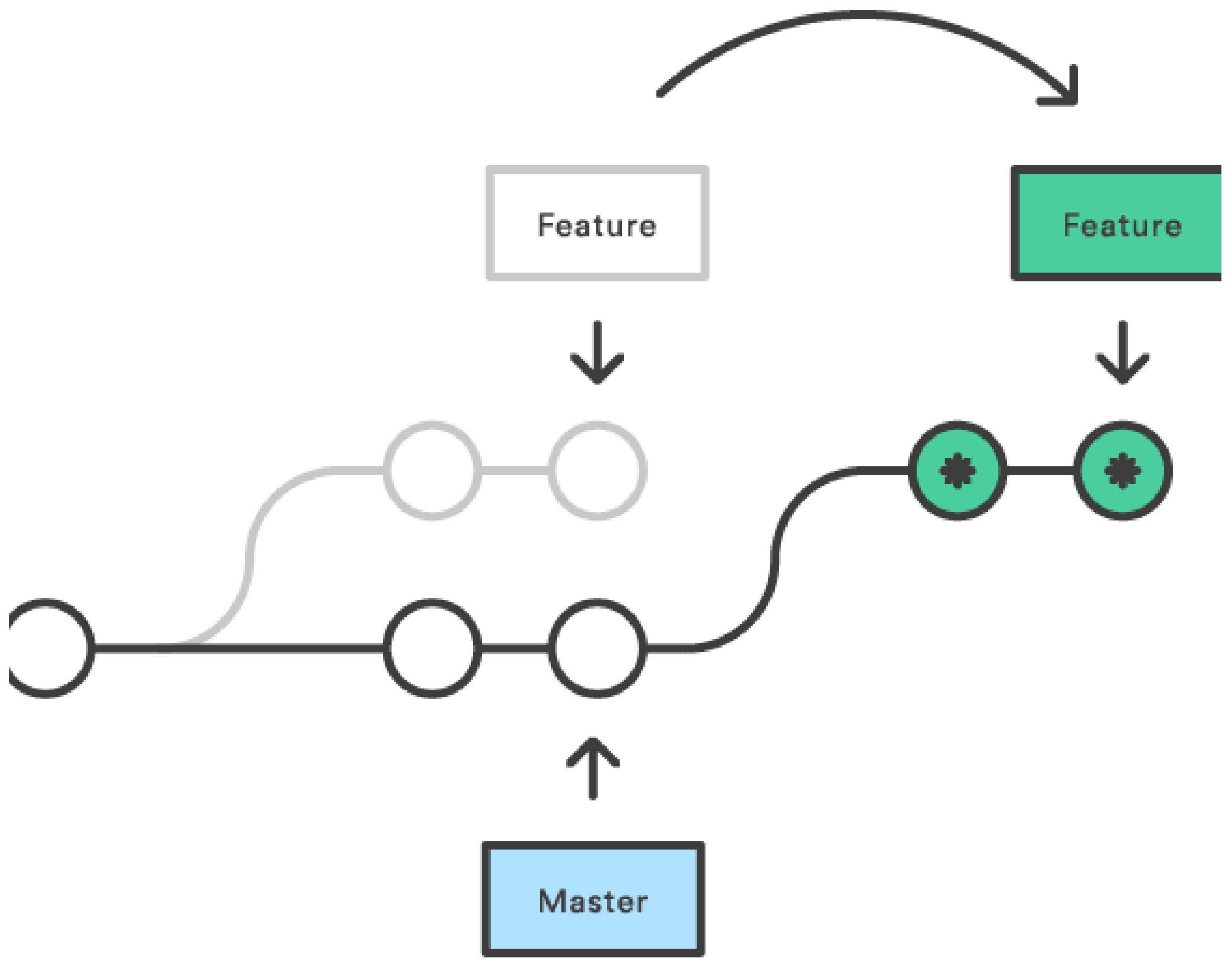
Merging

Using the git merge command is like bringing different storylines, created with git branch, and putting them together into one single path. It helps us combine the different parts of our code into a united and organized story.

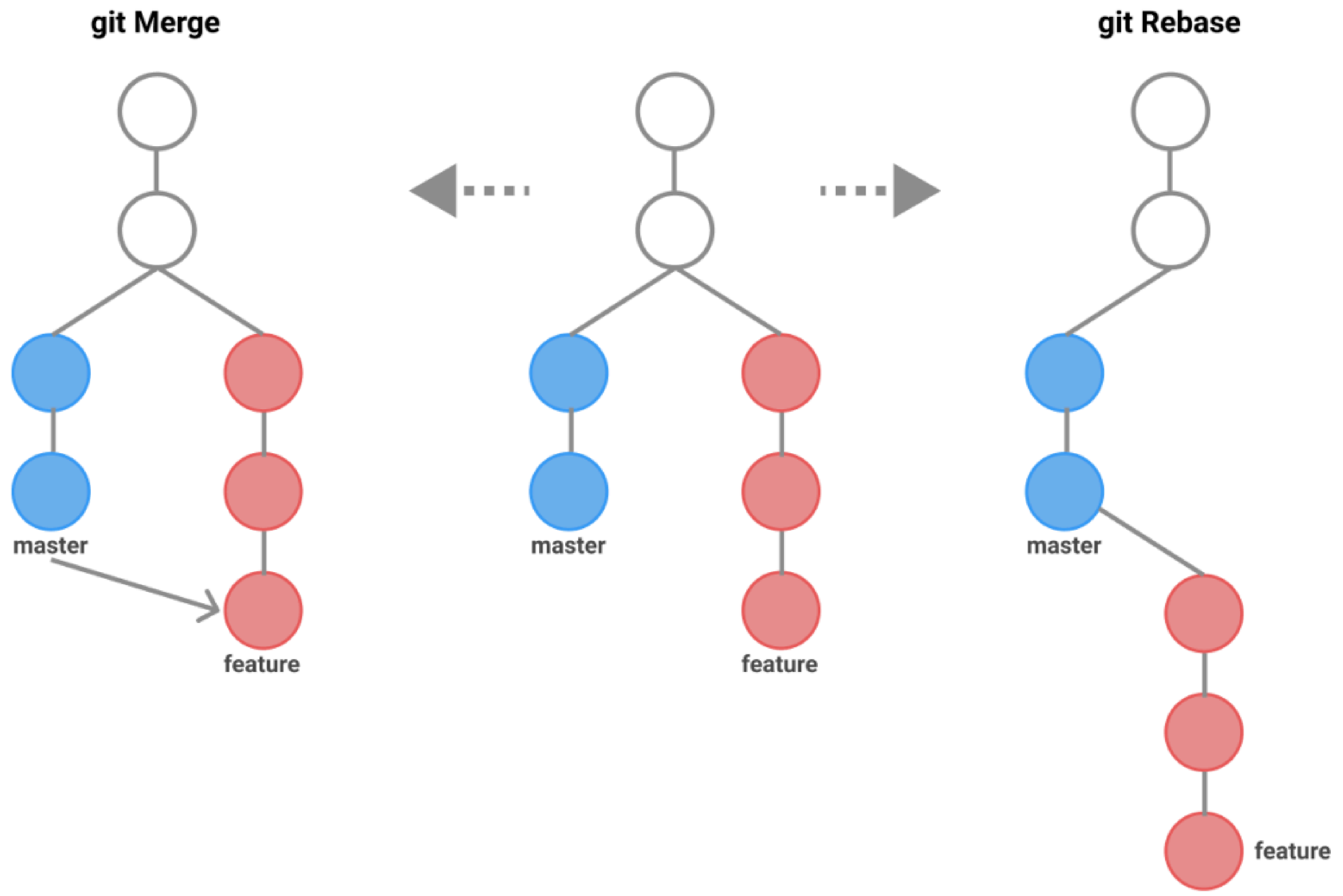


Rebasing

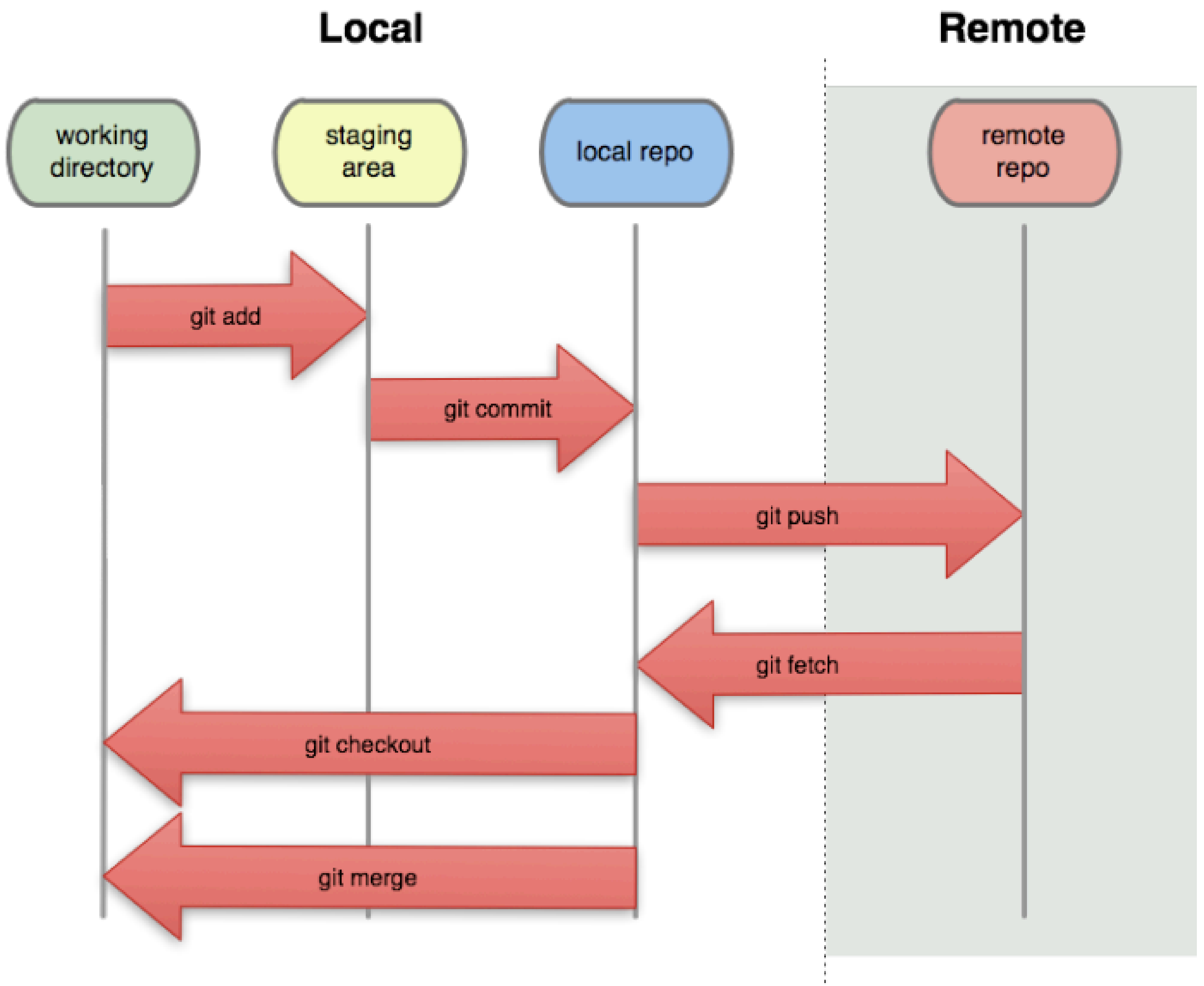
Moving or combining commits to a new starting point.
This process changes the original order of our commits by moving them to a new starting point.



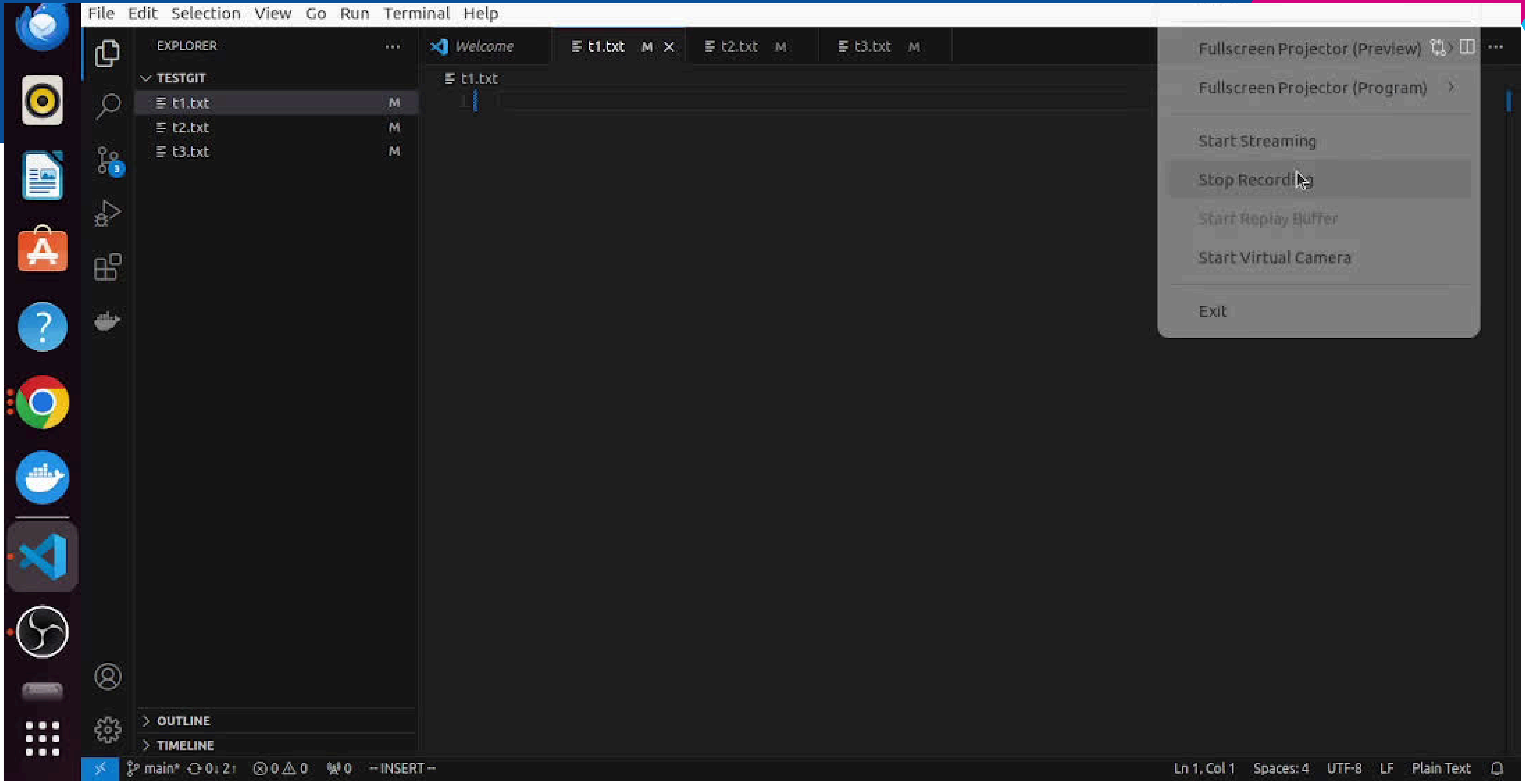
Rebasing VS Merging

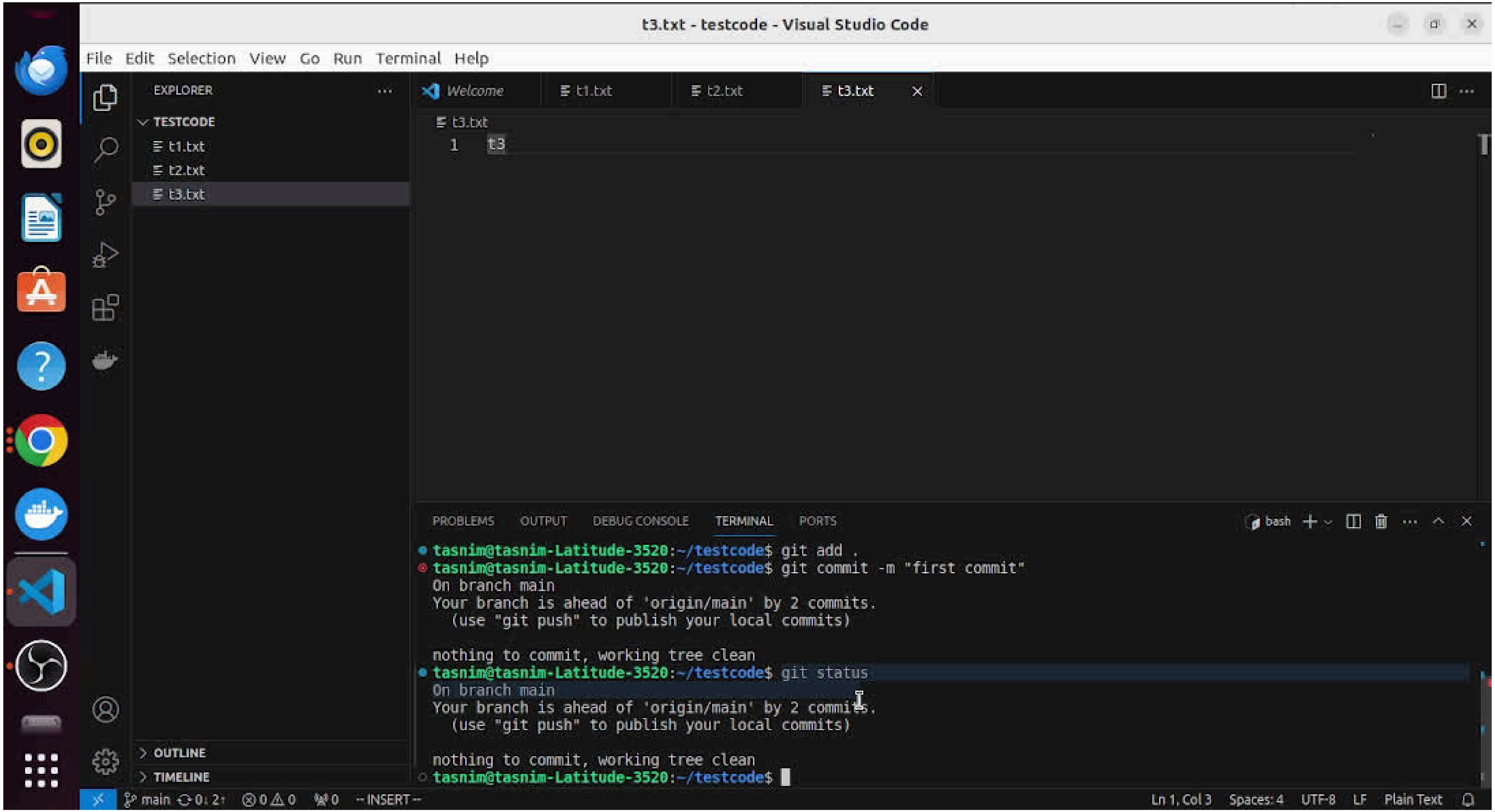


FROM LOCAL TO REMOTE -WORKFLOW



Let's participate together


File Edit Selection View Go Run Terminal Help
EXPLORER
TESTGIT
t1.txt M
t2.txt M
t3.txt M
Welcome
t1.txt M
t2.txt M
t3.txt M
Fullscreen Projector (Preview)
Fullscreen Projector (Program)
Start Streaming
Stop Recording
Start Replay Buffer
Start Virtual Camera
Exit
OUTLINE
TIMELINE
main* 0↓ 2↑ ⊗0 ⚠0 0 -- INSERT --
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text

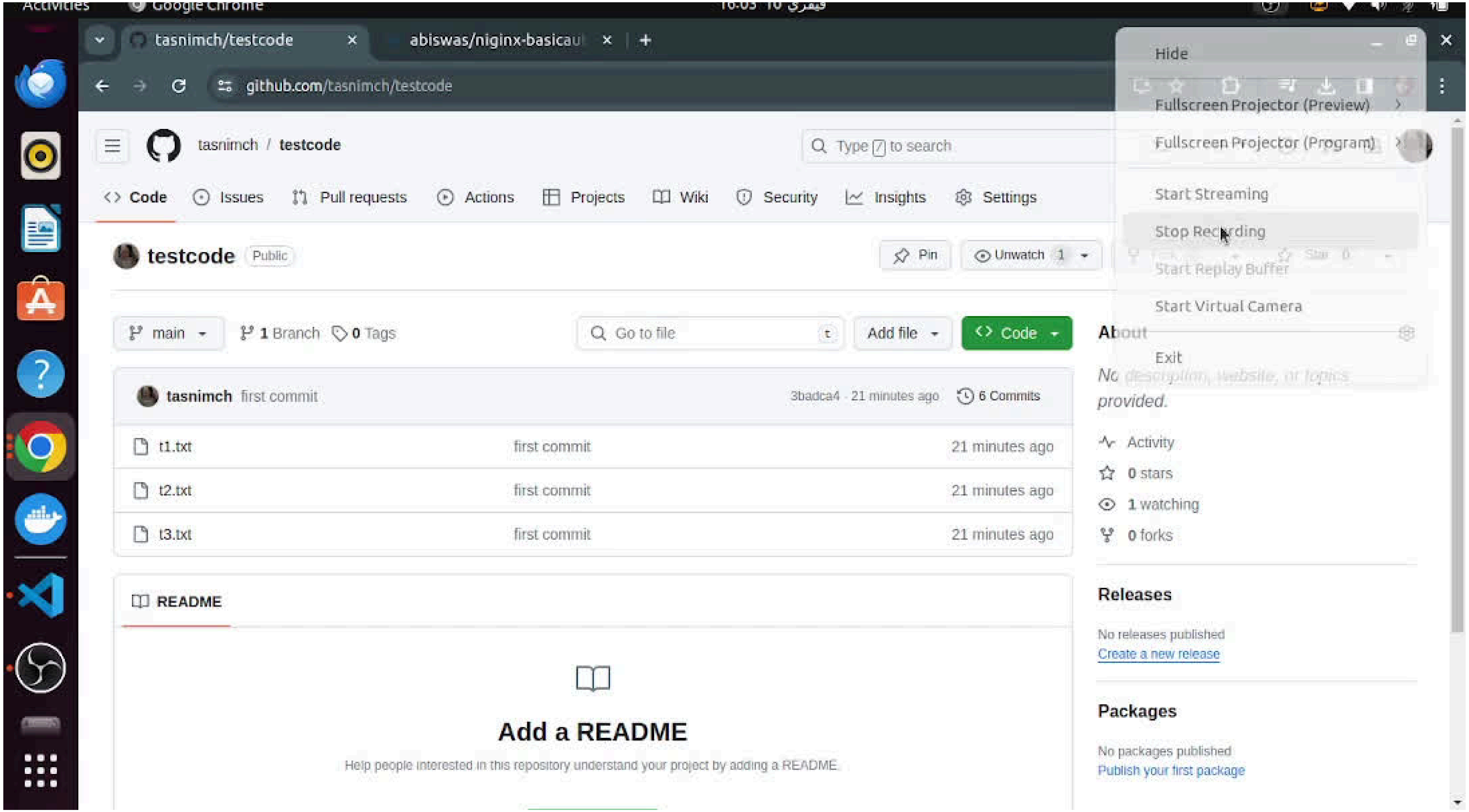

t3.txt - testcode - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
TESTCODE
t1.txt
t2.txt
t3.txt
Welcome
t3.txt
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
main 0↓ 2↑ 0 0 0 -- INSERT --
Ln 1, Col 3 Spaces: 4 UTF-8 LF Plain Text

```
1  t3
```

```
tasnim@tasnim-Latitude-3520:~/testcode$ git add .
tasnim@tasnim-Latitude-3520:~/testcode$ git commit -m "first commit"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
tasnim@tasnim-Latitude-3520:~/testcode$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
tasnim@tasnim-Latitude-3520:~/testcode$ 
```



testcode Public

tasnimch first commit 3badca4 · 21 minutes ago 6 Commits

t1.txt	first commit	21 minutes ago
t2.txt	first commit	21 minutes ago
t3.txt	first commit	21 minutes ago

README

Add a README

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

About

No description, website, or topics provided.

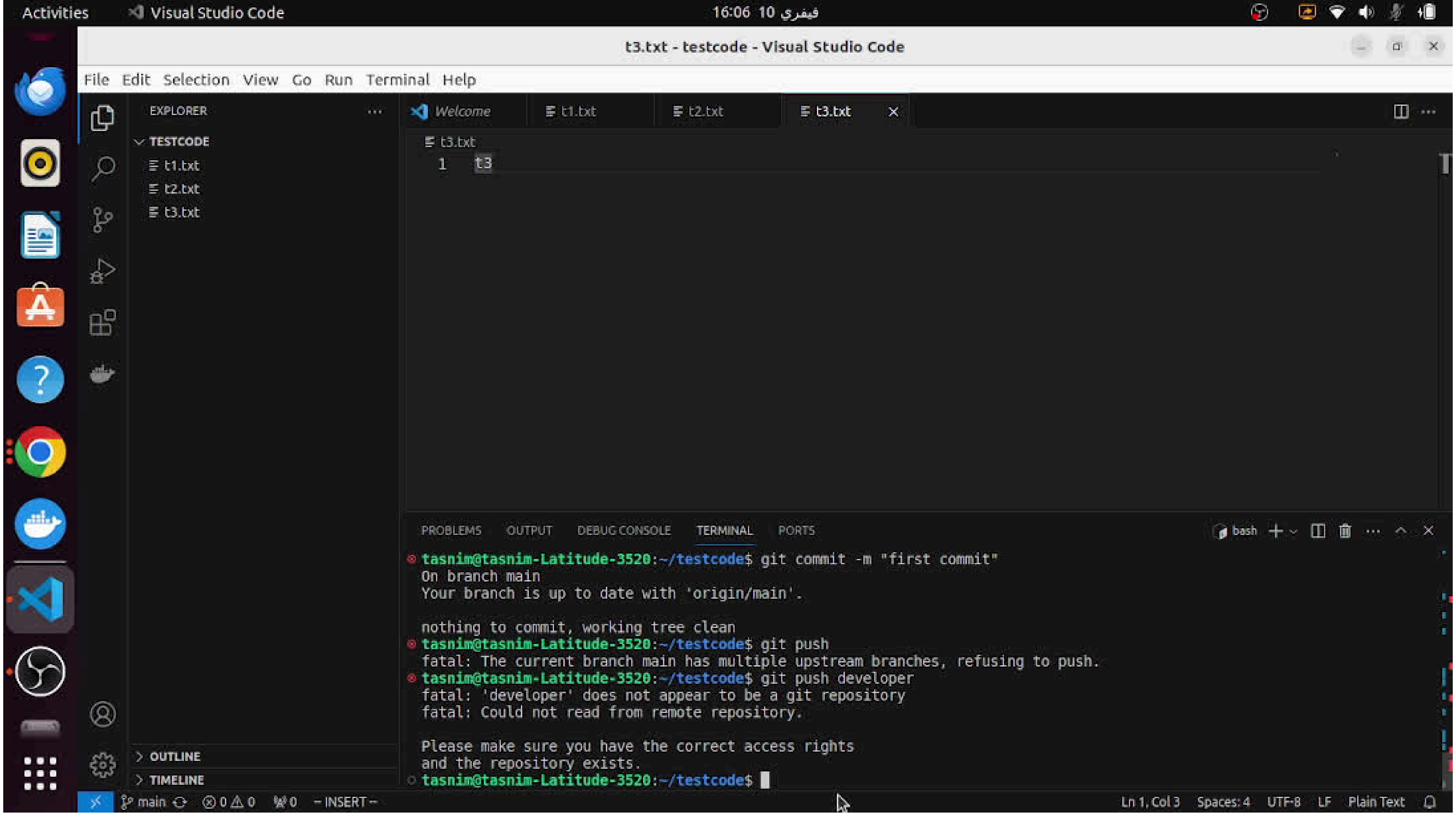
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package


Activities
Visual Studio Code
16:06 10 فيفري
t3.txt - testcode - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
TESTCODE
t1.txt
t2.txt
t3.txt
Welcome
t1.txt
t2.txt
t3.txt
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
main
-- INSERT --
Ln 1, Col 3 Spaces: 4 UTF-8 LF Plain Text

```
1  t3
```

```
tasnim@tasnim-Latitude-3520:~/testcode$ git commit -m "first commit"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
tasnim@tasnim-Latitude-3520:~/testcode$ git push
fatal: The current branch main has multiple upstream branches, refusing to push.
tasnim@tasnim-Latitude-3520:~/testcode$ git push developer
fatal: 'developer' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
tasnim@tasnim-Latitude-3520:~/testcode$ 
```

Docker

What is Docker

Was a platform and a set of tools designed to facilitate the creation, deployment, and execution of applications in lightweight, portable containers.

Elyadata

Images

An executable packages that include application code, runtime, libraries, and dependencies.

Containers

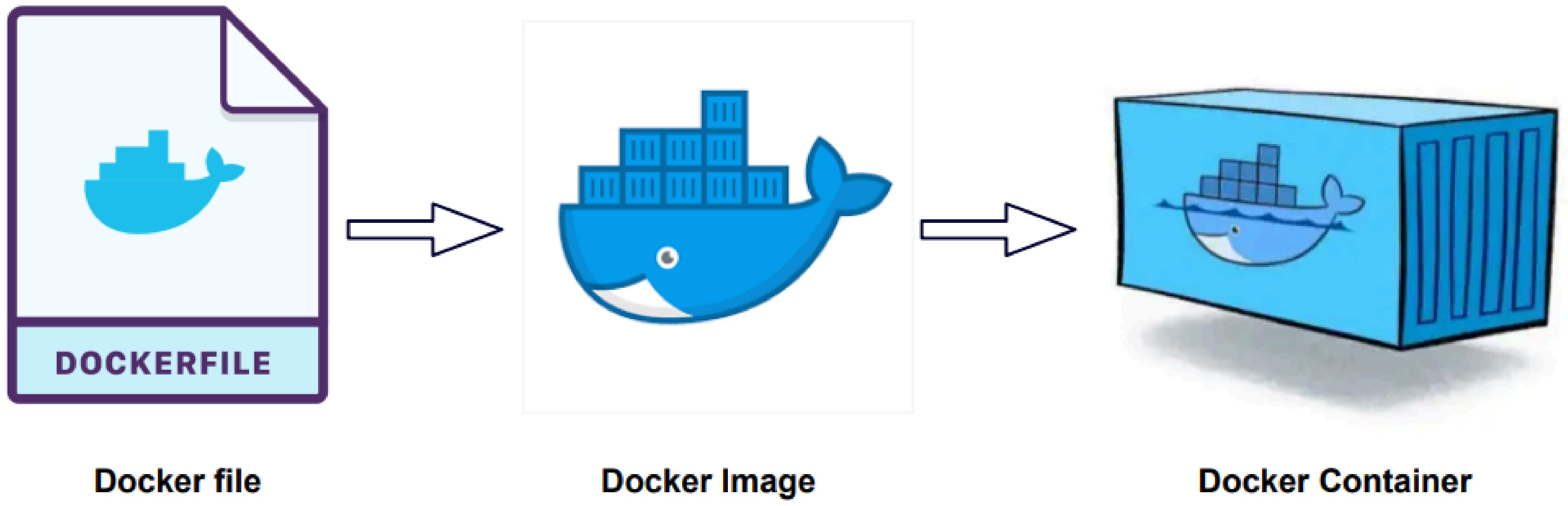
Running instances of Docker images, isolated from each other and the host system.

Dockerfile

A Dockerfile is a script used to build a Docker image.
Starting point for your application.

Concept of containerization

Containerization is a lightweight and portable solution for packaging, distributing, and running applications. Imagine a container as a self-contained unit that includes everything an application needs to run: code, runtime, system tools, libraries, and settings.It has become a fundamental technology in modern software development.



What Are Docker Volumes?

Docker volumes are a feature of Docker that provide a way to manage data in containers.It allows data to be shared and retained even when containers are stopped, started, or removed.

- **Anonymous Volumes**: are created with no specific source or name. They are typically used to store temporary or transient data generated by a container during its lifecycle.

```
docker run -d -v /app/data some_image
```

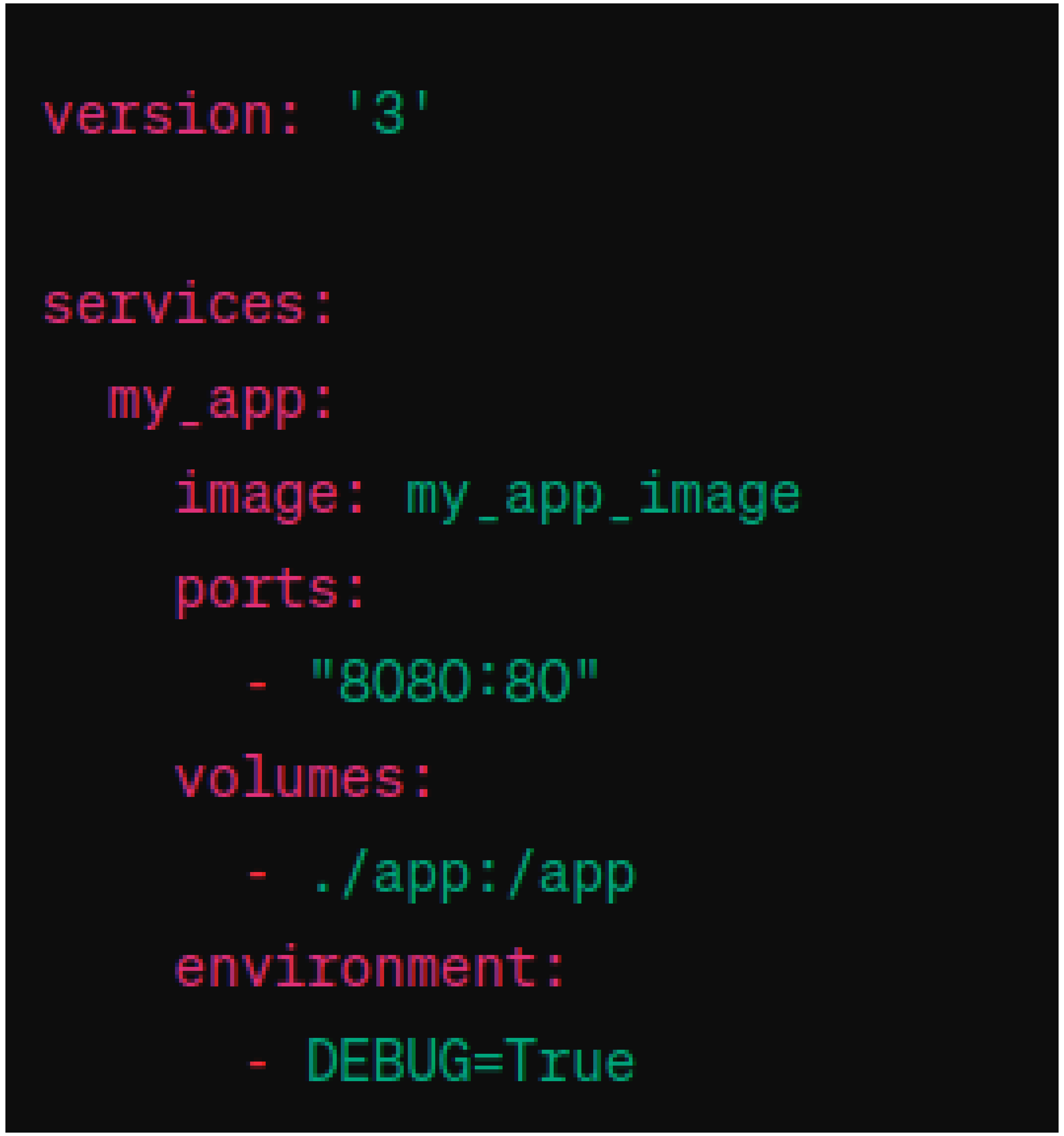
- **Named Volumes**: are created and managed with a user-defined name and specific source, allowing containers to independently share data across. NOTE: Named volumes are generally recommended for the production environment.

```
docker volume create my_volume
```

```
docker run -d -v my_volume:/app/data some_image
```

Docker Compose

Was a tool provided by Docker that allows you to define and run multi-container Docker applications. It uses a YAML file to configure the services, networks, and volumes of your application, making it easier to manage and deploy complex applications with multiple interconnected containers.



```yaml
version: '3'

services:
  my_app:
    image: my_app_image
    ports:
      - "8080:80"
    volumes:
      - ./app:/app
    environment:
      - DEBUG=True
```

Once you have your docker-compose.yml file ready, you can run your multi-container application with:

```
docker-compose up
```

Benefits of Docker

01 **PORTABILITY**

Run anywhere

02 **ISOLATION**

Avoiding conflicts between dependencies

03 **SCALABILITY**

Easy replication and scaling

04 **EFFICIENCY**

Resource optimization

FastAPI

What is FastAPI?

- FastAPI is a modern, fast (high-performance), web framework for building APIs with Python.
- It is based on standard Python's type hints, making it easy to use for data scientists familiar with Python.

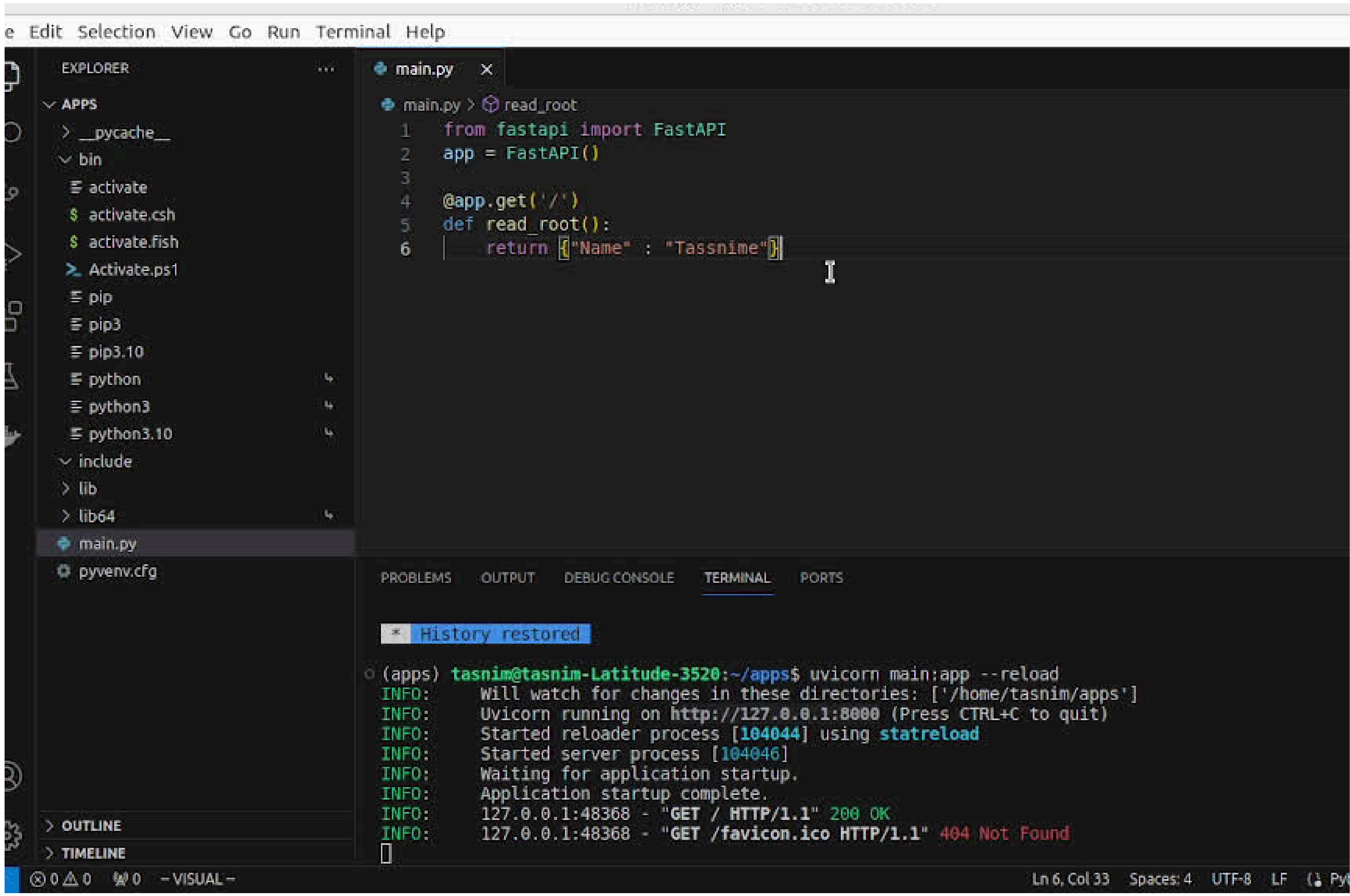
Why FastAPI for Data Science?

- Performance: FastAPI is built for high performance, making it suitable for data-intensive tasks.
- Automatic Docs: Generates interactive API documentation automatically based on Python type hints.

Installing and Create a Simple FastAPI

- It can be installed using pip. You will need to install FastAPI and the ASGI server `uvicorn`.

- Let’s directly get into creating a very simple toy API. I am using VS Code to implement this, but you can use any editor you like.



```python
from fastapi import FastAPI
app = FastAPI()

@app.get('/')
def read_root():
    return {"Name" : "Tassnime"}
```

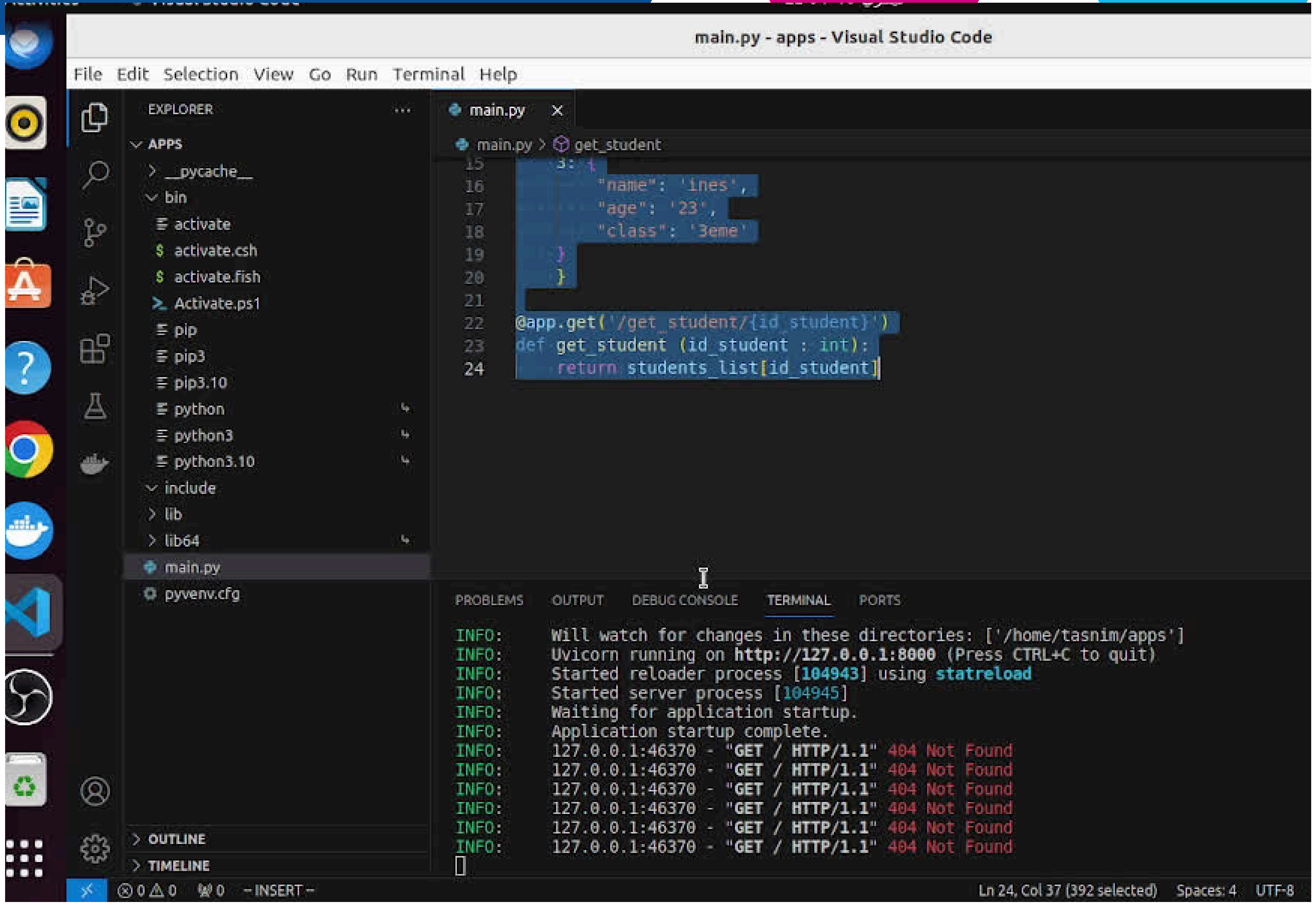
```
History restored

(apps) tasnim@tasnim-Latitude-3520:~/apps$ uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['/home/tasnim/apps']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [104044] using statreload
INFO:     Started server process [104046]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:48368 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:48368 - "GET /favicon.ico HTTP/1.1" 404 Not Found
```

Interactive API Docs

- FastAPI generates a "schema" with all your APIs using the OpenAPI standard for defining APIs. A "schema" is a definition or description of something. Not the code that implements it, but just an abstract description.

  - To see the documentation, just add `/docs` to the url (`http://127.0.0.1:8000/docs`). This link will show automatic interactive API documentation.



```python
        3: {
            "name": 'ines',
            "age": '23',
            "class": '3eme'
        }
    }

@app.get('/get_student/{id_student}')
def get_student (id_student : int):
    return students_list[id_student]
```

```
INFO:     Will watch for changes in these directories: ['/home/tasnim/apps']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [104943] using statreload
INFO:     Started server process [104945]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:46370 - "GET / HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:46370 - "GET / HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:46370 - "GET / HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:46370 - "GET / HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:46370 - "GET / HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:46370 - "GET / HTTP/1.1" 404 Not Found
```

More advanced examples

01 **GET**

To Read a Data

02 **POST**

To Create a Data

03 **PUT**

To Update Data

04 **DELETE**

To Delete Data

Elyadata

Thank you!