UFV

UFV.ES

Introduction to Software Development

MODULE 4 / UNIT 1 / 0.7

MOISES M. MARTINEZ

FUNDAMENTALS OF COMPUTER ENGINEERING

2024/25

What is Software Development?

Software Development

Software development refers to the process of designing, creating, testing, and maintaining software systems or applications.

This involves a combination of programming, problem-solving, and creativity to develop software that meets specific requirements or addresses particular challenges.

Software Development

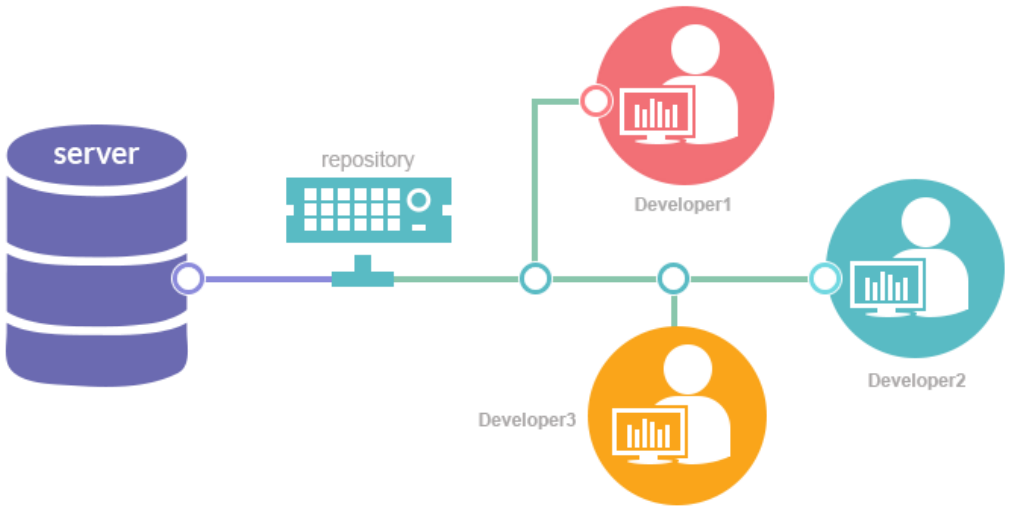
A significant portion of the software development life cycle revolves around source code:

- Coding and Implementation.
- Testing.
- Deployment.
- Maintenance and Updates.

Software Development

A significant portion of the software development life cycle revolves around source code:

- Coding and Implementation.
- Testing.
- Deployment.
- Maintenance and Updates.



Source code is handled using repositories

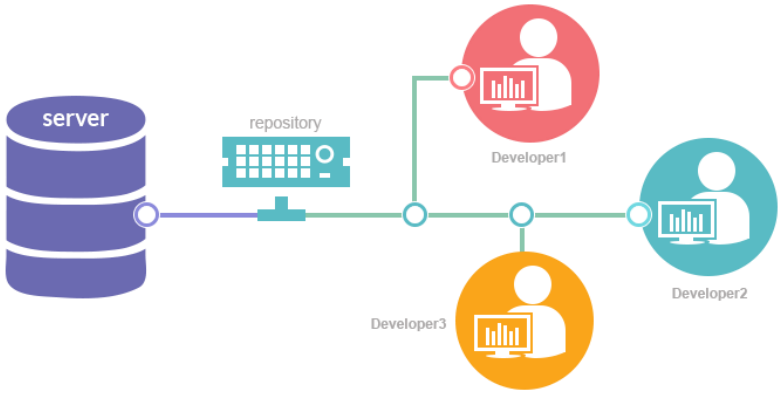
Repositories

01

Source code repositories

A source code repository, also referred to as a version control repository or code repository, is a centralized hub where software developers store, manage, and organize their source code files.

- It keeps a detailed history of code changes, making it easy to revert to previous versions, compare differences, and manage different development branches.
- Multiple developers can work on the same project simultaneously, merging their changes without overwriting each other's work.
- Ensures that the source code remains intact, and any issues can be traced back to specific changes or contributors.
- It acts as a backup, ensuring that the source code is stored securely and can be accessed when needed, even in case of system failures.



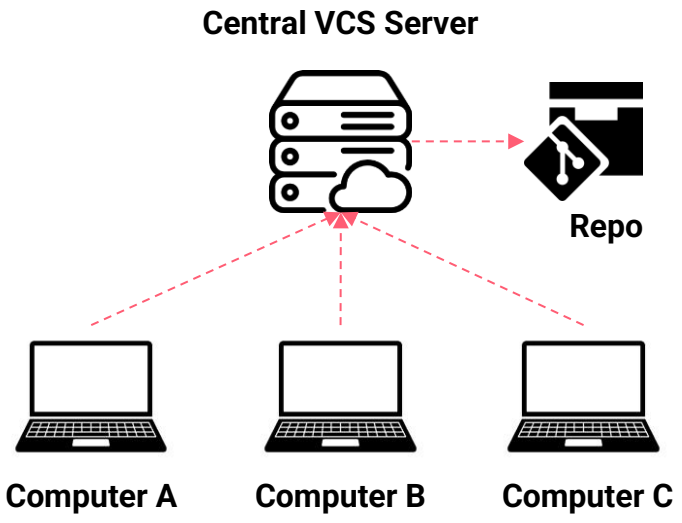
Source code repositories – Types of repositories

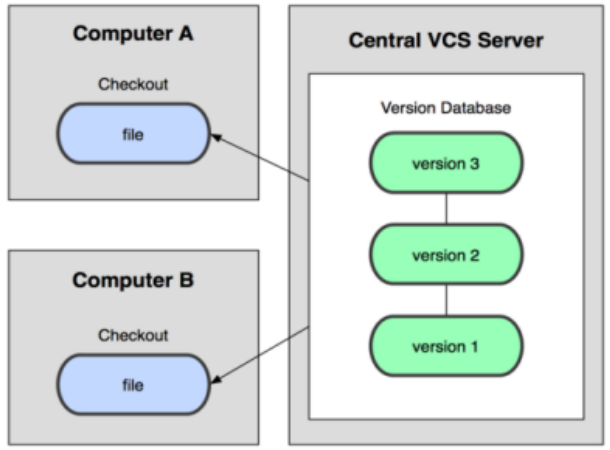
In a centralized version control system (VCS), a central server holds the "official copy" of the source code along with its complete version history.

1. Developers **check out** a copy of the code from the central server to their local machine.
2. Developers make changes to the code on their local machine.
3. Once the local modifications are complete, developers **check in** their changes to the central server. The server then updates the official copy of the code, recording these changes in the version history.

The centralized approach ensures that the server maintains a single, authoritative version history, and all code updates are coordinated through it.

Concurrent Version System (CVS) **Subversion (SVN)**

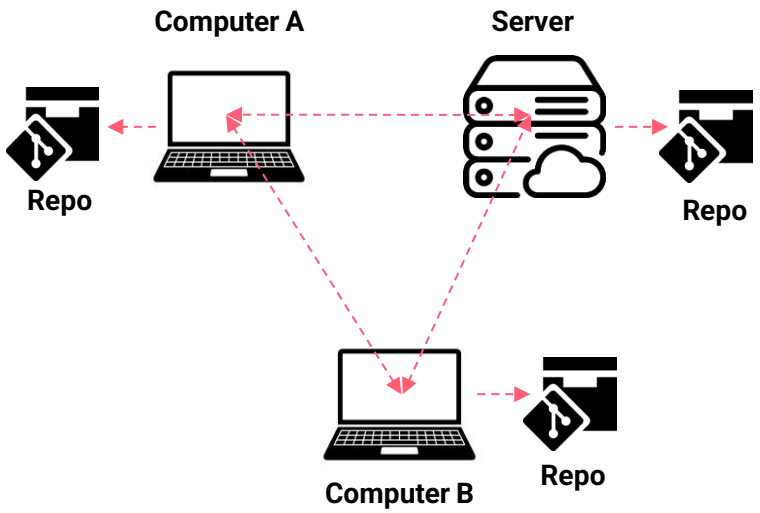


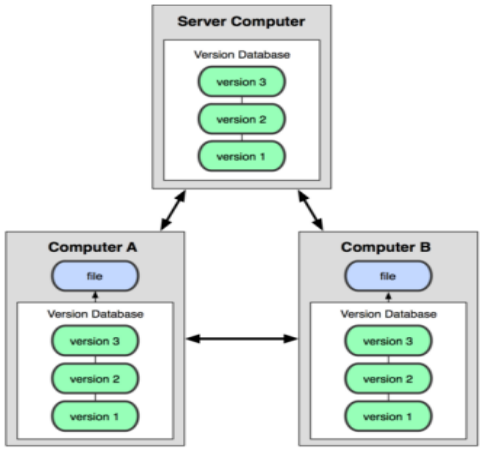


Source code repositories – Types of repositories

In a distributed version control system (DVCS), there is no central VCS server that holds the only authoritative copy of the code. Every developer's local repository contains a complete copy of the entire project, including the full version history.

1. Each developer's local machine contains a full clone of the repository.
2. Developers can execute most version control operations (e.g., commit, branch, merge) without contacting a remote server.
3. When updates are available on the remote server, developers can "pull" changes from it to synchronize their local repositories.
4. After making local changes, developers can "push" their modifications back to the remote server, which then integrates those changes into the shared repository for other developers to access.





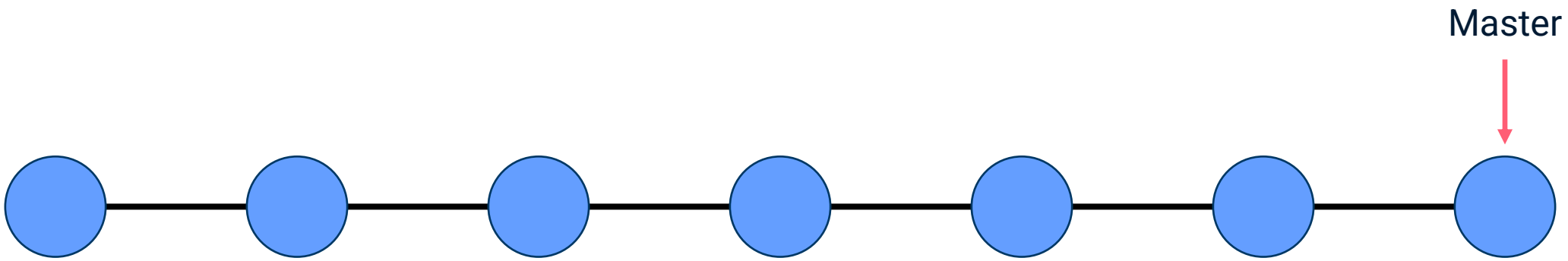
Mercurial **GIT**

GIT

02

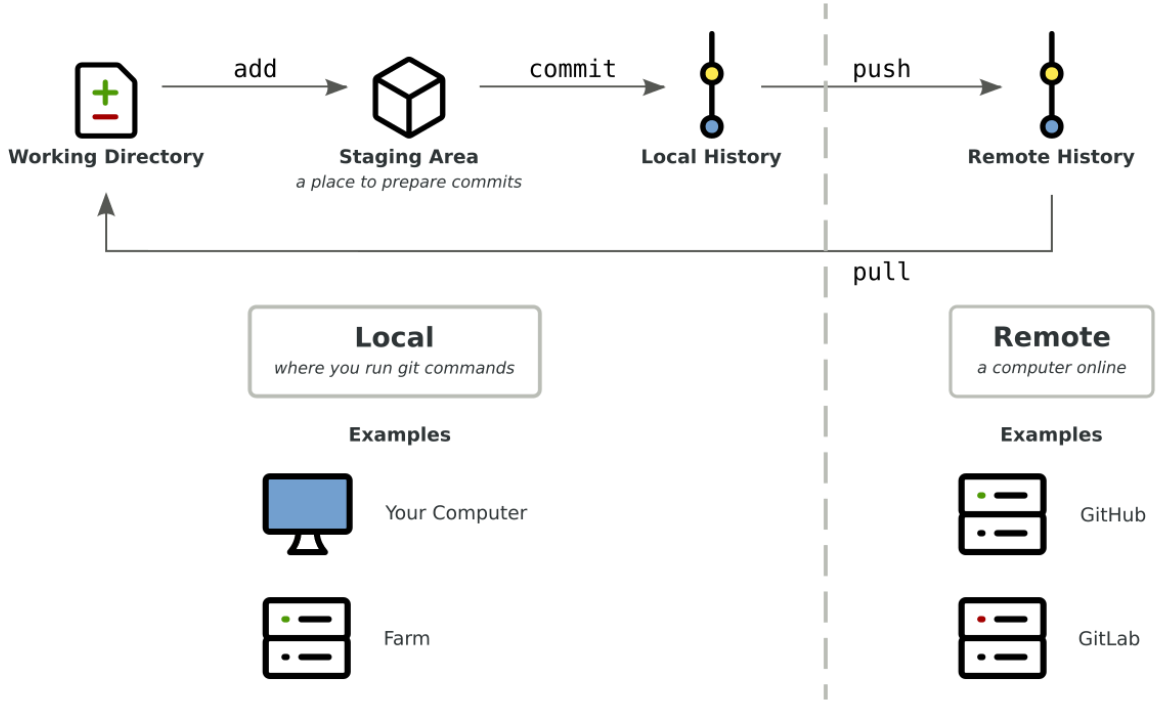
Software Development

Git is a distributed, open-source version control system that allows developers and data scientists to efficiently manage and track changes in their codebase enabling team collaboration.



Git has become an industry standard for version control due to its versatility and wide adoption across various development environments. It is compatible with nearly all development tools, command-line interfaces, and operating systems, making it accessible to a broad range of users.

Git environment: Local and remote

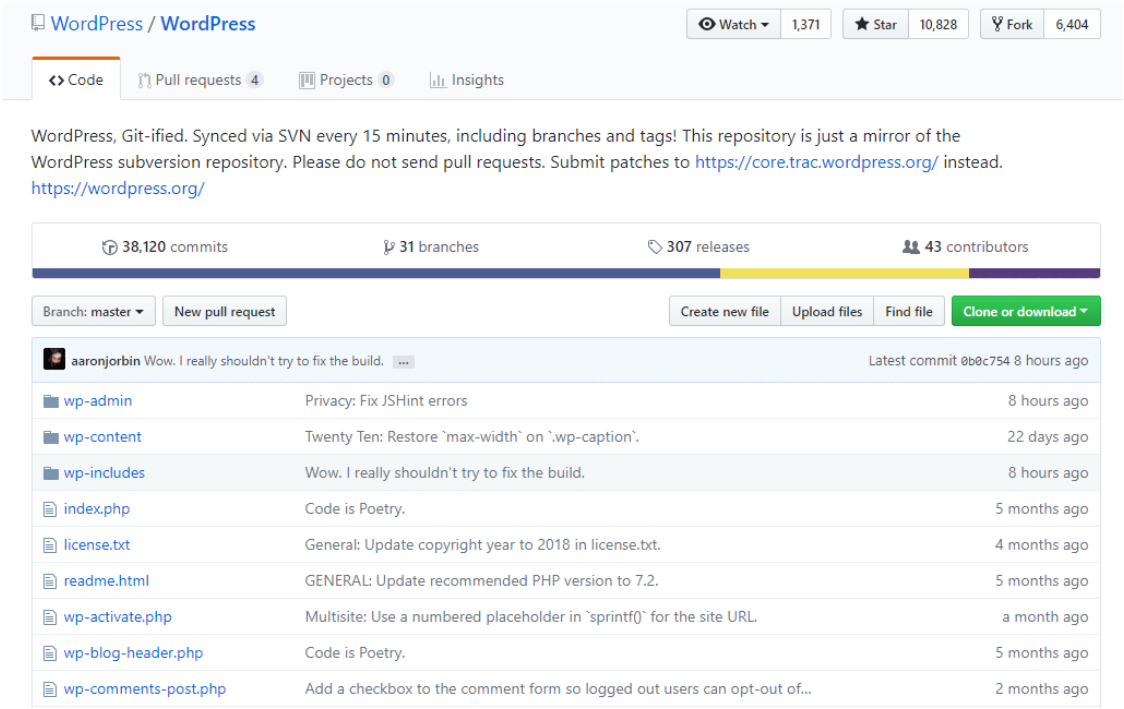


- The working directory is the area where developers modify files representing the current state of the files on the developer's machine.
- The staging area (index) is an intermediate zone between the working directory and the repository where developers add files before committing changes.
- A local repository is a complete copy of the project, stored on each developer's machine, containing the full history of the project and all its branches.
- A shared central repository serves as the remote repository where developers synchronize their local copies by pulling updates from it and pushing their changes to keep the project up-to-date across all contributors.

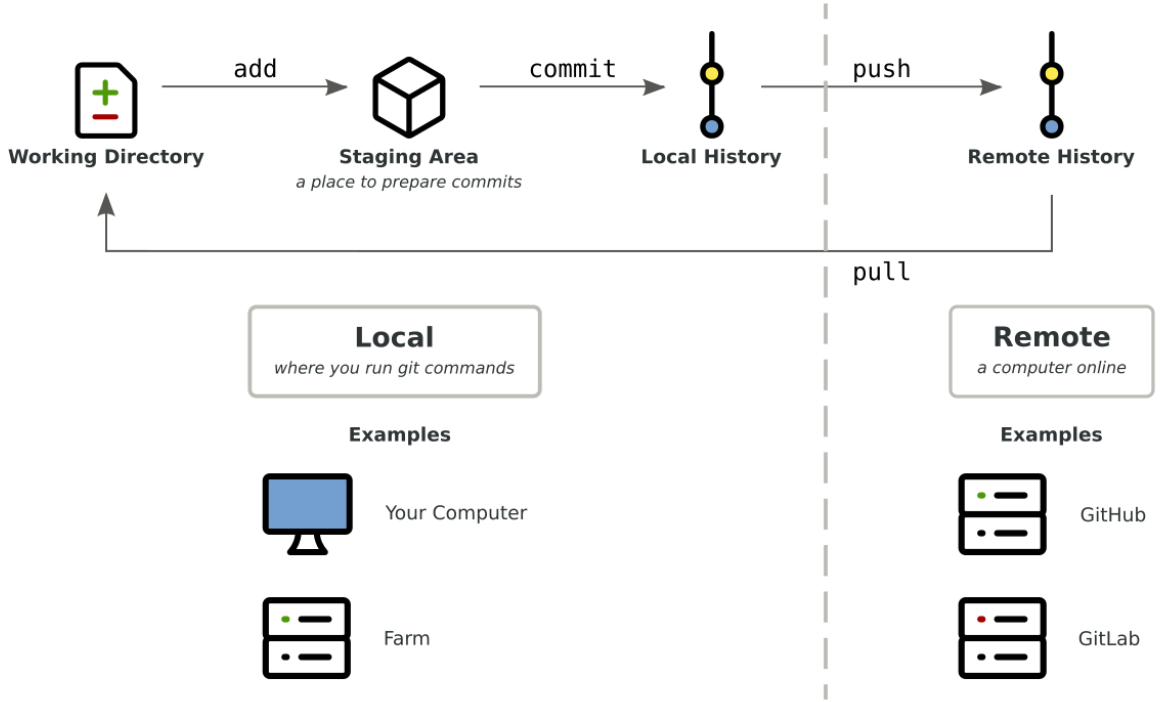
Git environment: Repository (repo)

A Git repository, commonly called a "repo," is a data structure that organizes and tracks a project's files, directories, and their complete revision history. It functions as a central hub for managing changes to source code or any collection of files, allowing developers to efficiently monitor, collaborate, and revert to previous versions when needed.

- Files and Directories
- Commit History
- Branches
- Remote Repository Connection



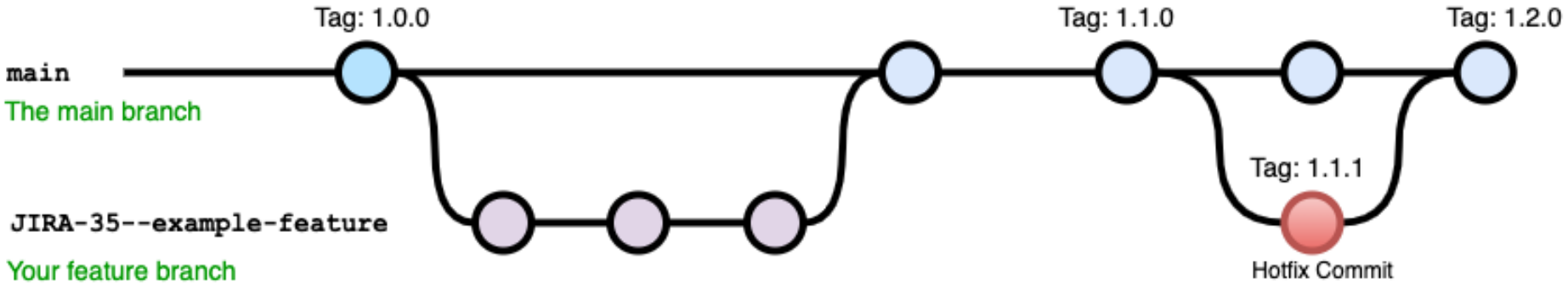
Git environment: Main operations



- Developers can **clone** repositories, which includes all files, branches, and the complete version history, storing it locally on their machine.
- Developers can make local changes, **commit** them to their repository, create new branches, and manage the code without an Internet connection.
- Developers can **pull** updates from the remote server to synchronize their local repository with the latest changes made by others.
- Developers can **push** their local changes to the remote repository, allowing others to access and integrate their contributions.

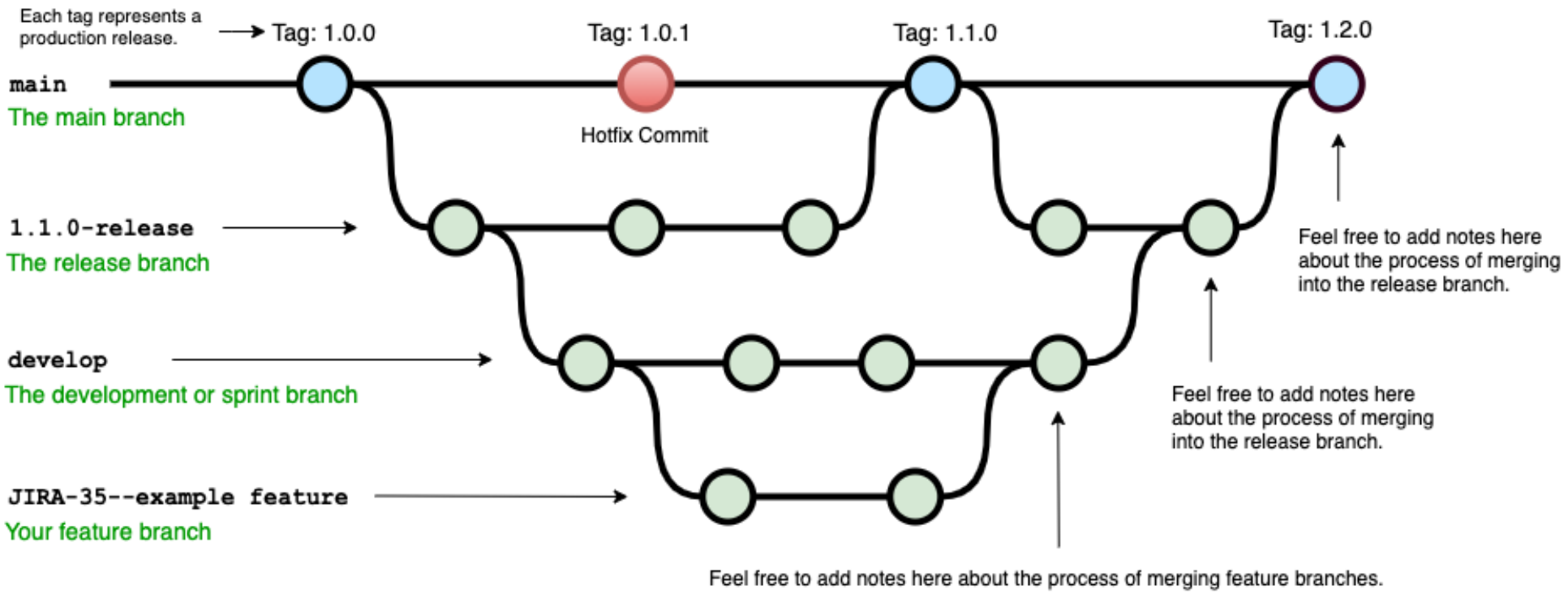
Git environment: Branches

In Git, a branch is a lightweight, movable pointer that represents an independent line of development within the repository which enables multiple developers to work on separate features, bug fixes, or experimental changes concurrently, keeping their work isolated from the main codebase and from each other.



This isolation ensures that developers can make changes without interfering with others' progress, and once the work is complete, branches can be safely merged back into the main branch.

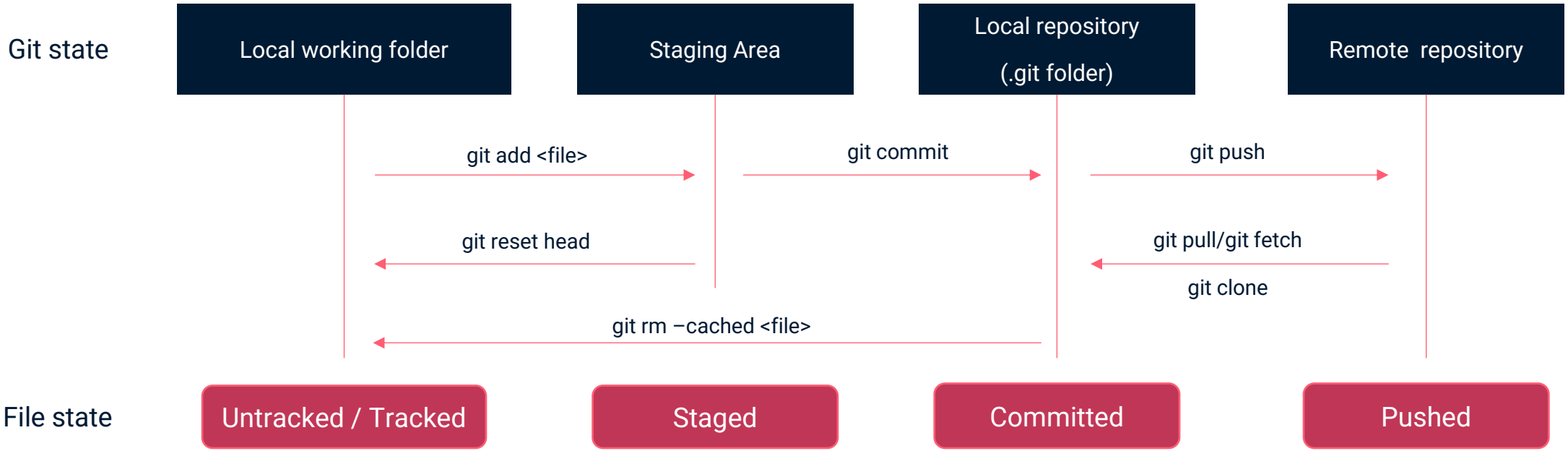
Git environment: Branches



The primary branch in a Git repository is typically called the **main** or **master** branch. It serves as the default branch where the stable, production-ready code is maintained.

Git environment: file livecycle

In Git, files can exist in one of several **states** that represent their position in the Git workflow and their relationship to the repository.



These states are part of the file's lifecycle as it moves between the **working directory**, the **staging area**, and the **repository**.

Git environment: file livecycle

Git file states represent the lifecycle of files as they move through different states within the Git version control system:

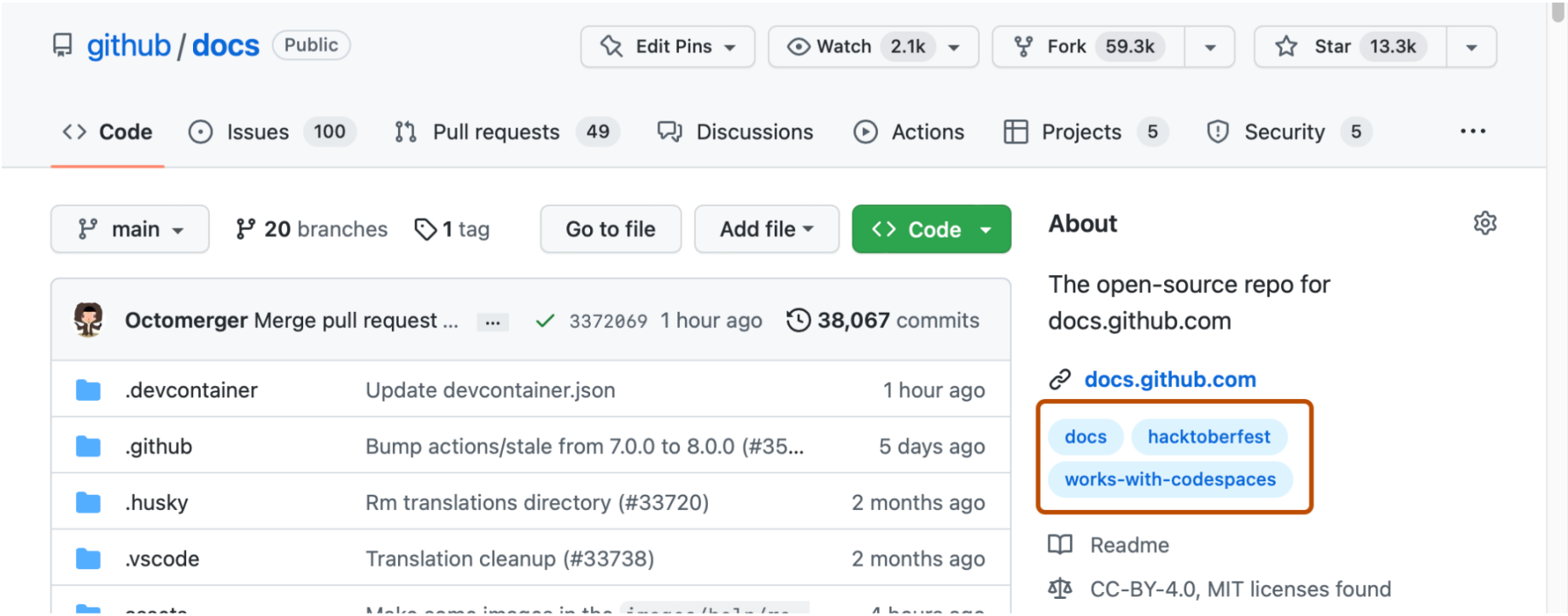
- Untracked: Files that exist in your working directory but are not being tracked by Git. These files are new and have not yet been added to the staging area.
- Tracked: Files that Git is actively monitoring. Tracked files can exist in one of two sub-states:
  - Unmodified: Tracked files that have not been changed since the last commit.
  - Modified: Tracked files that have been changed in the working directory but have not yet been staged for a commit.
- Staged (or Indexed): Files that have been added to the staging area and are ready to be included in the next commit.
- Committed: Files that have been saved to the Git repository. These files are stored in Git's database and are part of the repository's history.
- Pushed: Files that have been committed locally and then uploaded to a remote Git repository. These files are now stored on the remote server and are accessible to other collaborators.

GitHub account

03

OFF-EXAM CONTENT

Create a GitHub account

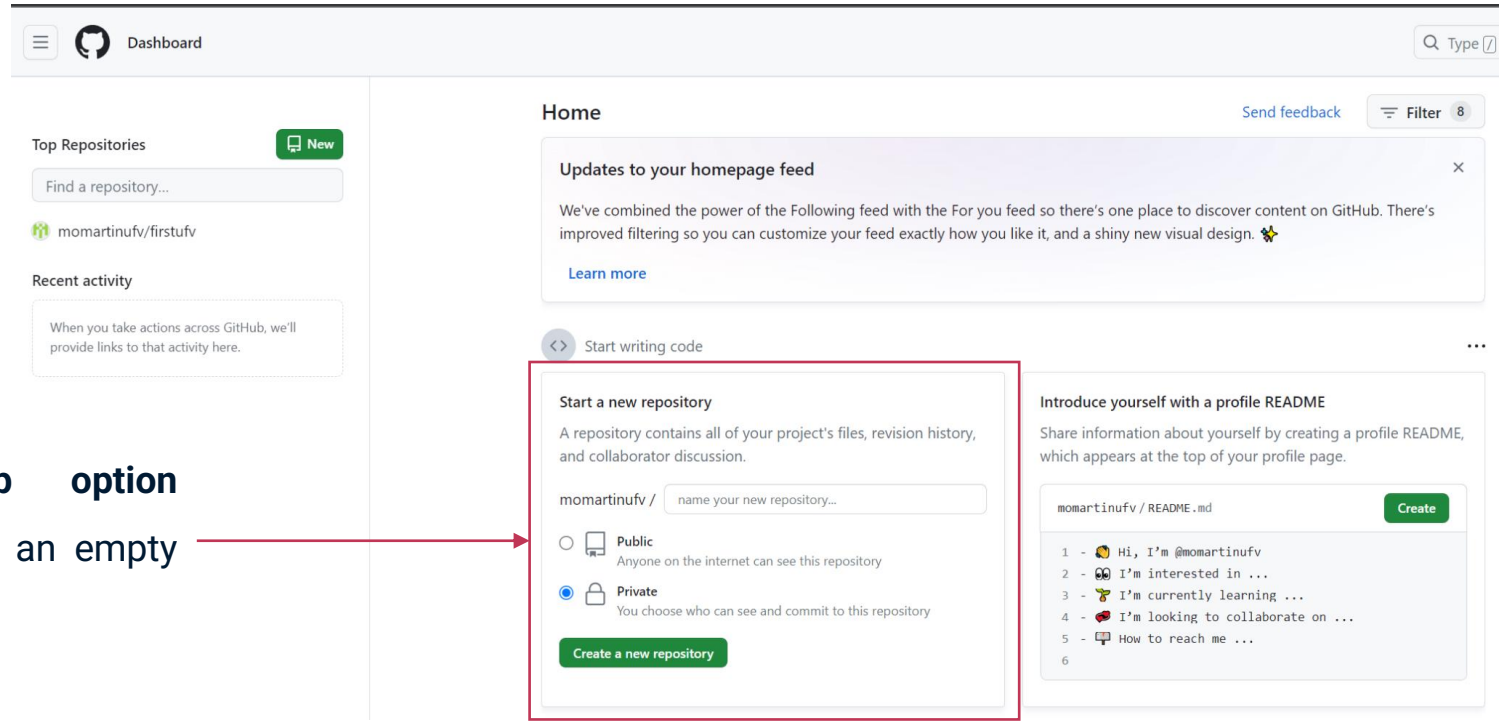


Join to GH

GitHub is a web-based platform that offers repository hosting services for version control, utilizing Git as the underlying system.

OFF-EXAM CONTENT

Create a GitHub repo

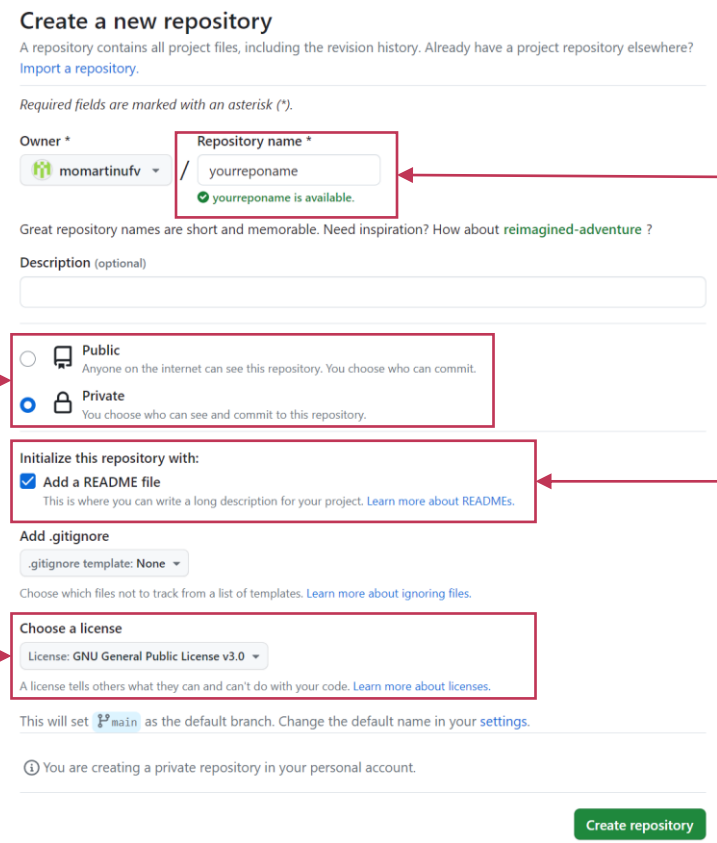


The **Quick setup option** consists on creating an empty repository quickly.

You can create repositories using various options on your GitHub dashboard. The quick setup option allows you to create an empty repository without including files like README.md or a license file.

OFF-EXAM CONTENT

Create a GitHub repo (Standard version)



The repository name must be unique and available within your GitHub account or organization.

The visibility of a GitHub repository can be set to either public or private. For most projects, it is recommended to choose private visibility.

The README file includes key information about the repository.

You can define a license for your source code to specify how others can use, modify, and distribute it.

It is recommended to include a license for the code stored in a repository.

GitHub tokens

04

OFF-EXAM CONTENT

Create a GitHub token

GitHub tokens are secure authentication credentials used to interact with GitHub's API and perform actions programmatically without using your regular password.

abcdefghijklmnopqrstuvwxyz1234567890

This token is your new password to access to your repository

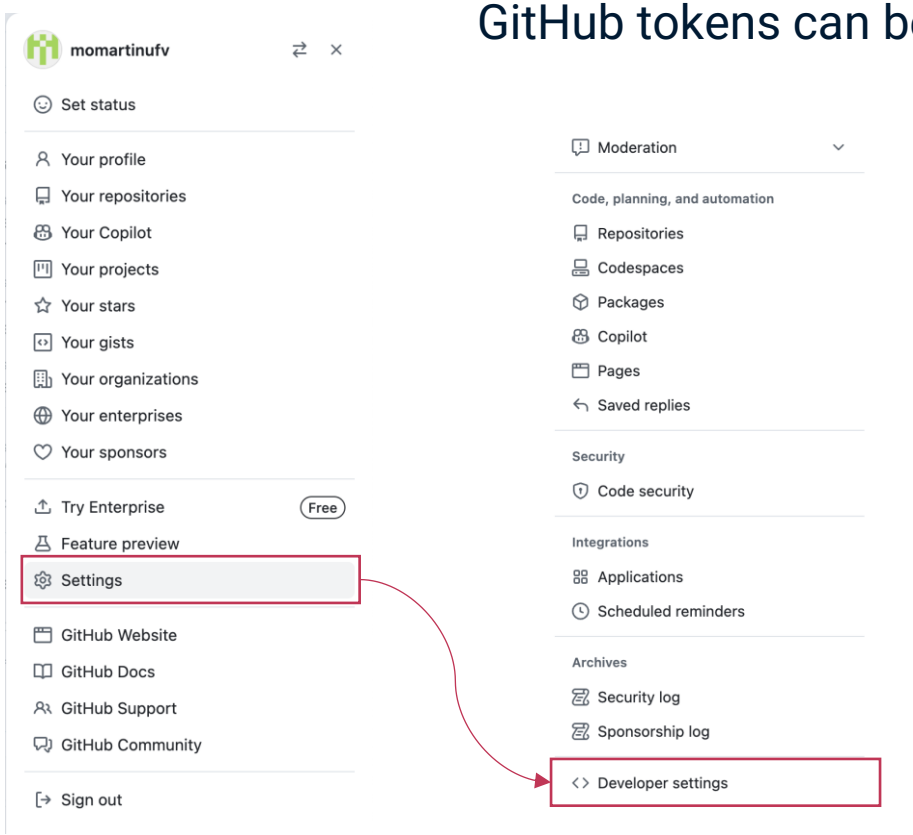
These tokens provide a way to access GitHub resources without using a username and password, enhancing security and enabling automation.

In macOS (and other operating systems), GitHub no longer allows using a password for Git operations (like cloning, pulling, or pushing) over HTTPS because of a security enhancement introduced by GitHub.

OFF-EXAM CONTENT

Create a GitHub token

GitHub tokens can be created in your GitHub settings.



Log in to GitHub:

- Go to GitHub and log in with your account.

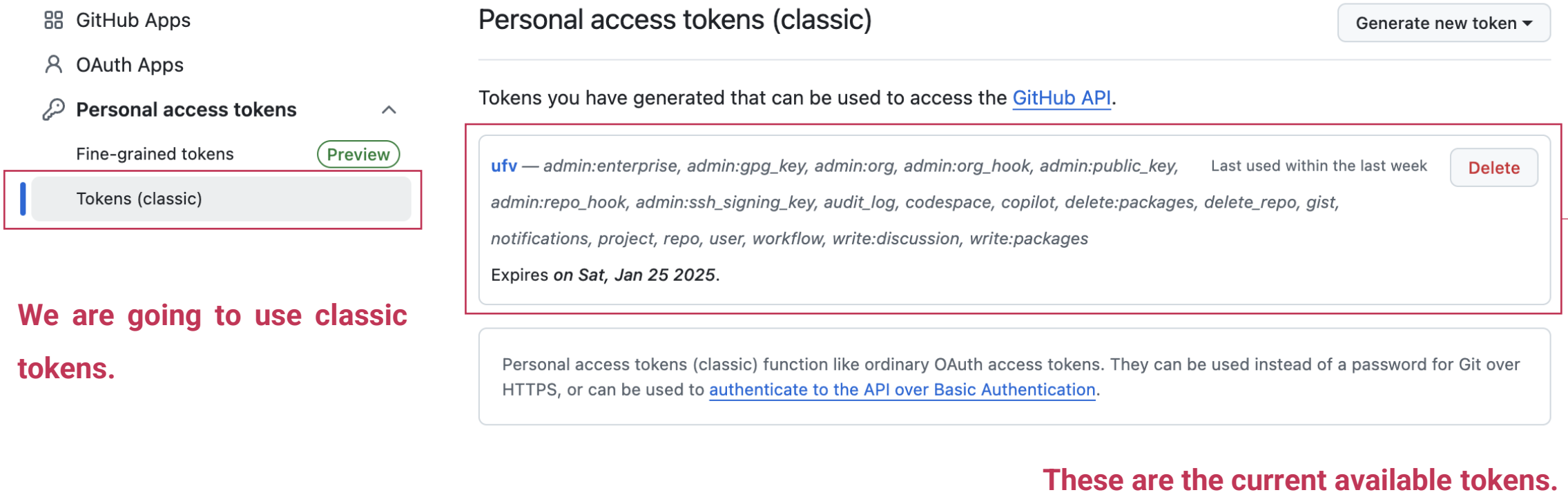
Navigate to Developer Settings:

- In the upper-right corner, click your profile picture, and then click **Settings**.
- Scroll down and find **Developer Settings** in the left-hand menu.

OFF-EXAM CONTENT

Create a GitHub token

There are different type of tokens: (1) **Tokens (classic)** for the traditional token type, or (2) **Fine-grained Tokens** for more scoped and precise control.



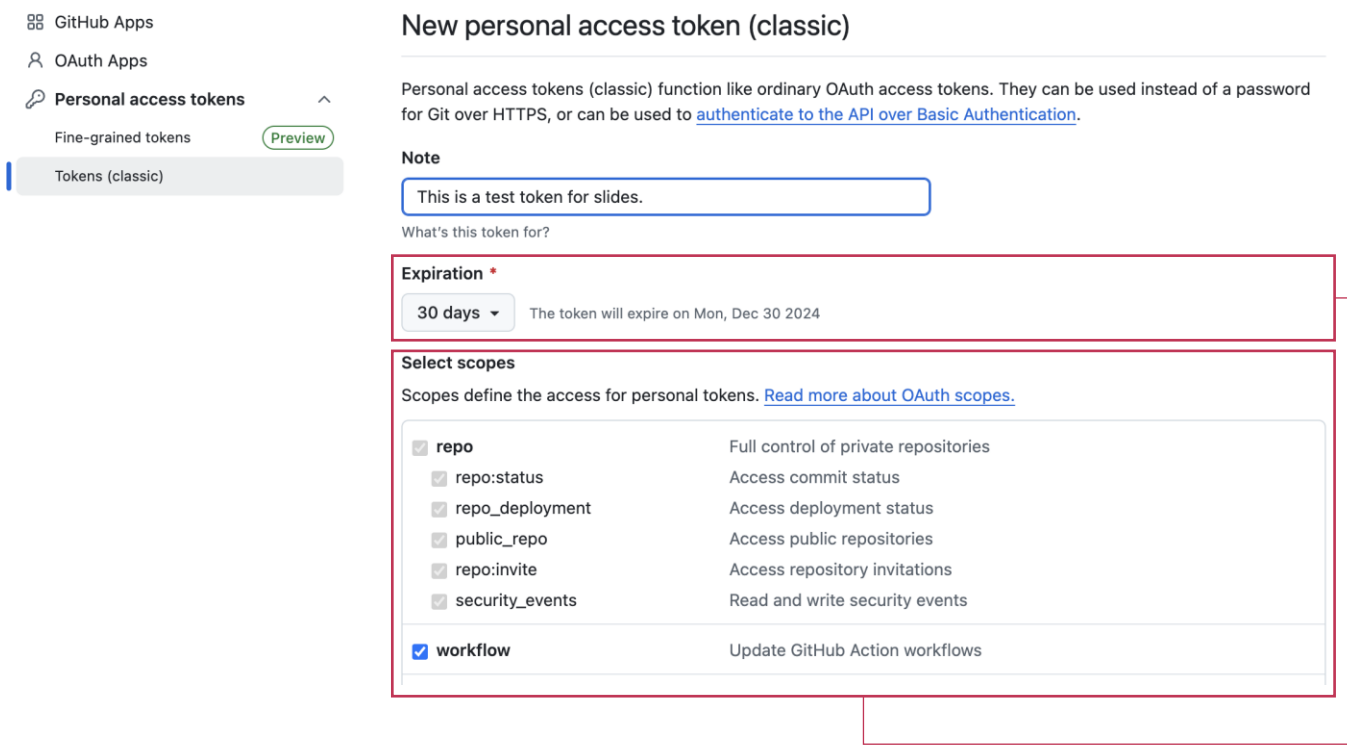
We are going to use classic tokens.

These are the current available tokens.

OFF-EXAM CONTENT

Create a GitHub token

A classic token in GitHub is a personal access token (PAT) used to authenticate API requests or Git operations, configured with specific scopes to control permissions.



We need to include a description for the token to identify its purpose (e.g., "This is a test token for slides").

Set the **expiration date** (e.g., 30 days, 60 days, etc., or no expiration for permanent tokens).

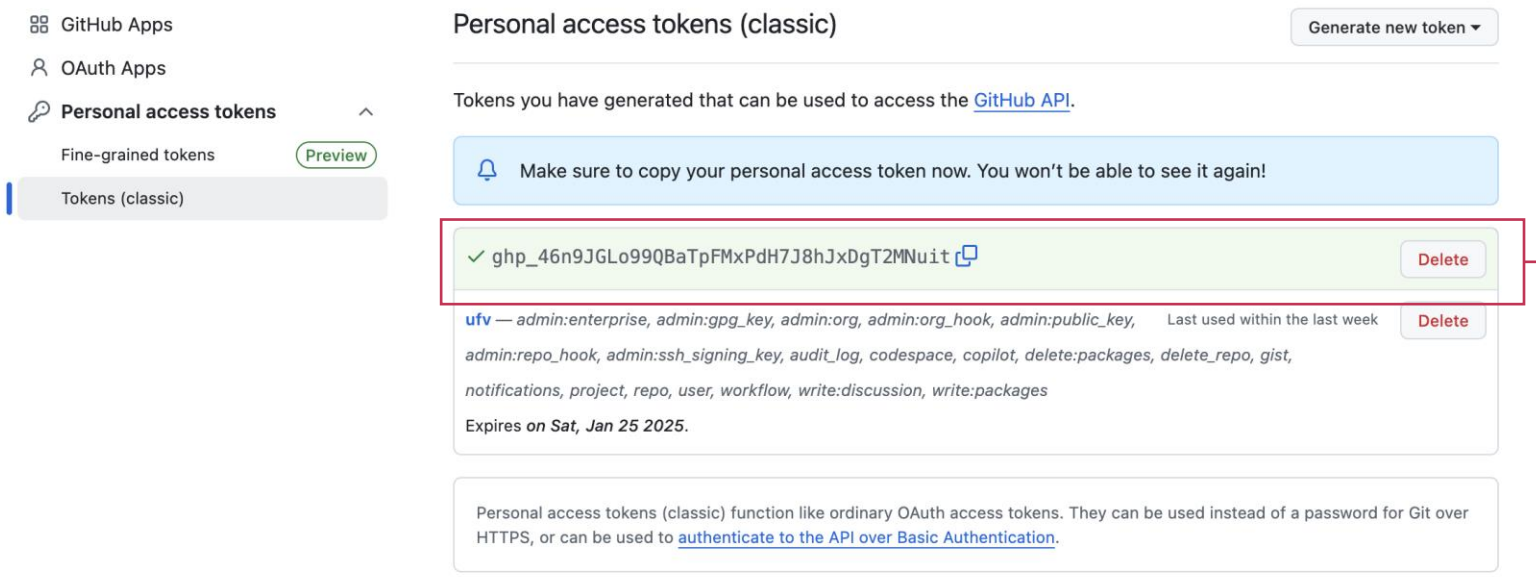
Select the necessary **scopes** (permissions over the functionalities). For example:

- repo: Access to repositories.
- read:org: Read access to organization data.
- admin:org: Manage organization settings.

OFF-EXAM CONTENT

Create a GitHub token

Once the GitHub token is generated, it will be displayed through the application's web interface.



Tokens in GitHub can only be viewed during their creation, so it is crucial to copy or securely store the token immediately after generating it.

This token will serve as your password from now on for executing Git commands with your GitHub repository.

Git – Basic commands

05

OFF-EXAM CONTENT

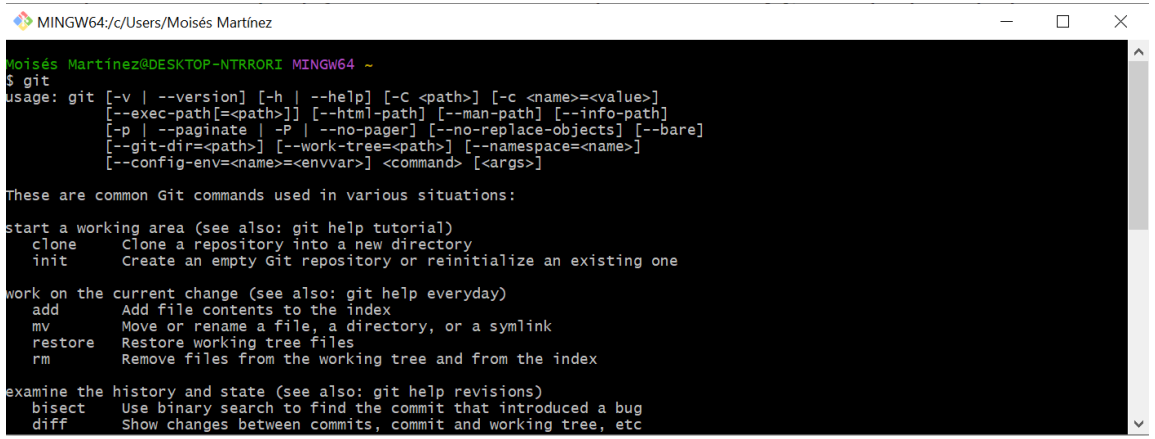
Install Git in your laptop

- Linux (Ubuntu) → Command sudo apt-get install git
- Windows → http://git-scm.com/download/win
- Mac → http://git-scm.com/download/mac

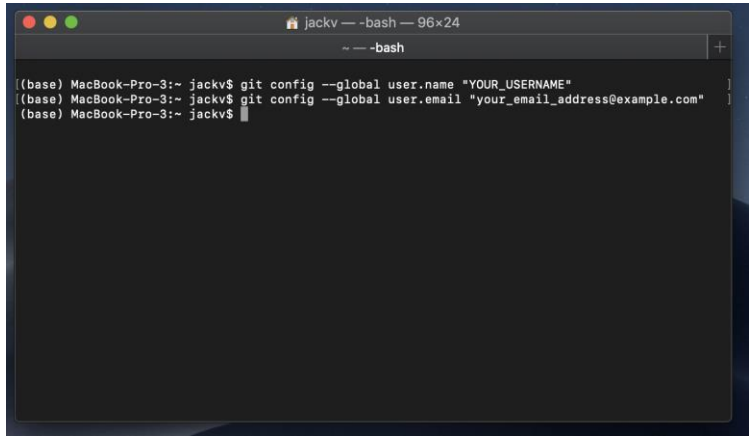
Fundamental concepts: Running Git in your command line

The primary method for using Git is through the command line, which provides direct access to all Git functions and features.

- For Windows users, Git Bash is commonly used, offering a Unix-like shell to run Git commands.
- For macOS and Linux, the native terminal can be used for running Git commands.



Git Bash on Windows



Terminal on Mac

Fundamental concepts: Running Git in your command line

There are some basic commands for using the command line (Bash/Unix) that will be useful for you.

Command syntax	Description
cd **folder_name**	The cd command, short for "change directory," allows users to switch from their current working directory to a specified directory
ls -la	The ls command allows users to list files and directories in the current working directory. It provides a concise view of the contents, displaying names of files and folders. Adding options like -l provides detailed information, while -a shows hidden files.
mkdir **folder_name**	The mkdir command allows users to create a new directory or folder. By specifying a “folder name” as an argument, it instantly generates the specified directory in the current working directory
rmdir **folder_name**	The rmdir command allows users to remove an empty directory or folder. It deletes the specified directory if it contains no subdirectories or files.
cat **file_name**	The cat command allows users to concatenate and display the contents of one or more files. It can be employed to view the entire content of a file.
rm **file_name new_file_name**	The mv command allows users to moves files and directories from one directory to another or renames a file or directory. If you move a file or directory to a new directory, it retains the base file name.

Fundamental concepts: Running Git in your command line

There are some basic commands for using the command line (Windows) that will be useful for you.

Command syntax	Description
cd **folder_name**	The cd command, short for "change directory," allows users to switch from their current working directory to a specified directory
dir /p	The dir command allows users to list files and directories in the current working directory.
mkdir **folder_name**	The mkdir command allows users to create a new directory or folder. By specifying a “folder name” as an argument, it instantly generates the specified directory in the current working directory
rmdir **folder_name**	The rmdir command allows users to remove an empty directory or folder. It deletes the specified directory if it contains no subdirectories or files.
cat file_name	The cat command allows users to concatenate and display the contents of one or more files. It can be employed to view the entire content of a file.
rm **file_name new_file_name**	The mv command allows users to moves files and directories from one directory to another or renames a file or directory. If you move a file or directory to a new directory, it retains the base file name.

Fundamental concepts: Create a local repo

The git init command is used in Git to initialize a new Git repository in a directory on your local machine. This command sets up the necessary Git infrastructure, creating an empty repository with an initial commit.

```
git init
```

After executing git init, you will see a new subdirectory named **.git** created within the current directory. This directory is where Git stores all the necessary files and metadata to manage version control for your project.

Fundamental concepts: Clone a repo

The git clone command is used in Git to create a local copy of a remote repository. When you clone a repository, you retrieve all of its files, commit history, and branches onto your local machine.

This is the account or organization that owns the repository.

```
git clone https://github.com/your-username/your-repository.git
```

This is the specific repository within that account.

Once you have created a repository on GitHub, you can proceed to clone it, thereby creating a local copy on your machine. To initiate this process, please follow the subsequent instructions for creating your repository.

Fundamental concepts: Add changes/files to the repo

The git add command is used in Git to add changes or new files to the **staging area**. The staging area is a critical part of Git's workflow, where you select and prepare specific changes or files to be included in the next commit.

We can add individual files (txt, docx, c, etc).

```
git add <file_name_1> <file_name_2> <folder_name_1> ...
```

We can add folders

After running git add, the specified changes or files are added to the staging area, ready to be committed.

Git add is used to adding change to existing files and add new files.

Fundamental concepts: Commit changes to the local repo.

The git commit command is used in Git to create a new commit in a Git repository. A commit represents a snapshot of the current state of your project's files and serves as a permanent record of changes made to the repository.

The -m option is a shortcut to include commit messages directly in the command

```
git commit -m "Added new features and fixed bugs"
```

A commit message is a concise description of the changes made in a particular commit.

It is possible to execute a git commit command without using the -m option; however, it is considered best practice to always include a clear and descriptive commit message.

A well-crafted message helps provide context and clarity to developers.

Fundamental concepts: Push to the remote repo.

The git push command is used in Git to upload local commits from your repository to a remote repository. It allows you to synchronize your changes with the remote repository, ensuring that others can access your updates.

The <remote> in the git push command specifies the name of the remote repository where you want to upload your local commits. The <remote> allows Git to know which repository to interact with, making it an essential part of synchronizing your local work with a remote source.

```
git push <remote> <branch>
```

The <branch> in the git push command specifies the branch you want to upload to the remote repository.

You can omit the <branch> in the git push command if your current branch is tracking a remote branch.

Fundamental concepts: Push to the remote repo.

During the initial push, there is a chance that Git credentials might not be configured correctly. In such cases, it's important to set up your username and email address to properly associate commits with your identity before proceeding.

user.name specifies the name of the user associated with the commits being pushed to the repository.

```
git config --global user.name "My super name"
git config --global user.email "username@alumnos.ufv.es"
```

user.email specifies the email address associated with the user making commits and pushing changes to the repository.

For GitHub, the user.email must match the one registered with your GitHub account to properly link your commits to your profile.

Fundamental concepts: Pull from the remote repo.

The git pull command is used in Git to fetch and integrate changes from a remote repository into your current branch.

The <remote> specifies the name of the remote repository from which you want to pull changes, with origin being the default remote in most cases.

This option is optional if your branch is already tracking a branch on a remote.

```
git pull <remote> <branch>
```

The <branch> specifies the branch in the remote repository from which you want to pull changes into your current branch.

This option is optional if your local branch is already tracking a specific remote branch.

Fundamental concepts: Fetch from the remote repo.

The git fetch command is used in Git to retrieve updates from a remote repository without automatically merging them into your local branch, giving you the opportunity to review and integrate changes at your discretion.

The <remote> specifies the name of the remote repository from which you want to pull changes, with origin being the default remote in most cases.
This option is optional if your branch is already tracking a branch on a remote.

```
git fetch <remote> <branch>
```

The <branch> specifies the branch in the remote repository from which you want to pull changes into your current branch.
This option is optional if your local branch is already tracking a specific remote branch.

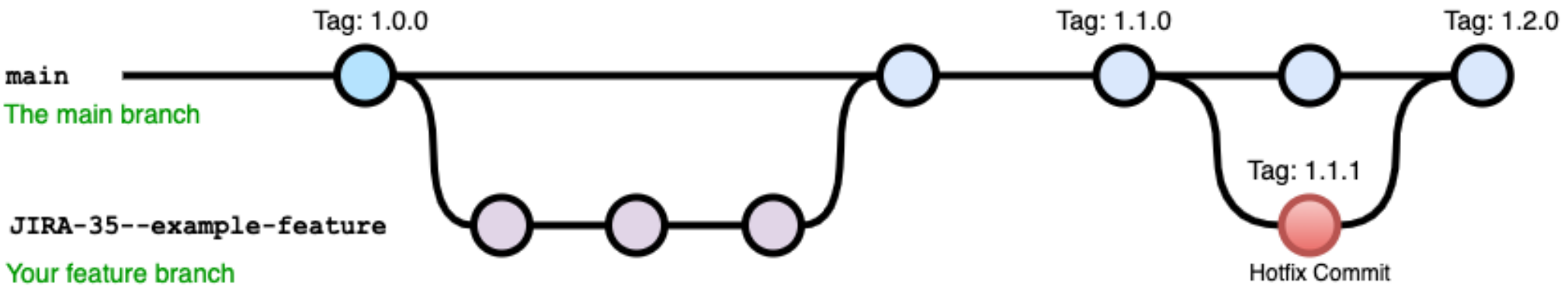
Git – Branches

06

Branches of code

A **branch** in Git is a lightweight pointer to a specific commit in the repository's history. These branches exist in two forms:

- **local branches** (stored on your machine)
- **remote branches** (stored on a remote repository, such as GitHub or GitLab).



Branches allow developers to isolate their work, enabling parallel development, experimentation, and collaboration without affecting the main codebase.

Branches of code: Check for branches

The git branch command is used in Git to manage and interact with branches in a Git repository.

```
git branch
```

```
* main
  feature-branch
  bugfix
  remotes/origin/main
  remotes/origin/feature-branch
  remotes/origin/bugfix
```

The currently active branch is highlighted with an asterisk (*) and displayed in a different color.

The plain git branch command show all the branches that exist in your local repository.

Branches of code: Check for branches

The git branch command is used in Git to manage and interact with branches in a Git repository.

Command syntax	Description
git branch	List local branches.
git branch -r	List remote branches.
git branch -a	List all branches (local + remote).
git branch -M <branch_name>	Rename the current branch to branch_name.
git rev-parse --abbrev-ref HEAD	Show the current branch.
git remote show origin	Show details about remote branches.
git log --oneline --graph --decorate --all	Show a graphical view of all branches.

Branches of code: Change to another branch

The git checkout command is used in Git to manage branches by switching branches, restoring files, and detaching the HEAD to inspect specific commits or states in a repository.

```
git checkout branch_name
```

The branch_name must exist. This command switches the working directory to the branch_name branch and updates the files to match the state of that branch.

After switching branches, it is necessary to run a git pull to download any changes made by other developers.

Branches of code: Change to another branch

The git checkout command is used in Git to manage branches by switching branches, restoring files, and detaching the HEAD to inspect specific commits or states in a repository.

```
git checkout -b branch_name
```

The -b flag in the git checkout command is used to create a new branch and immediately switch to it. It combines two basic operations into one:

1. **Creating a new branch (equivalent to git branch branch_name).**
2. **Switching to the new branch (equivalent to git checkout branch_name).**

Branches of code: Change to another branch

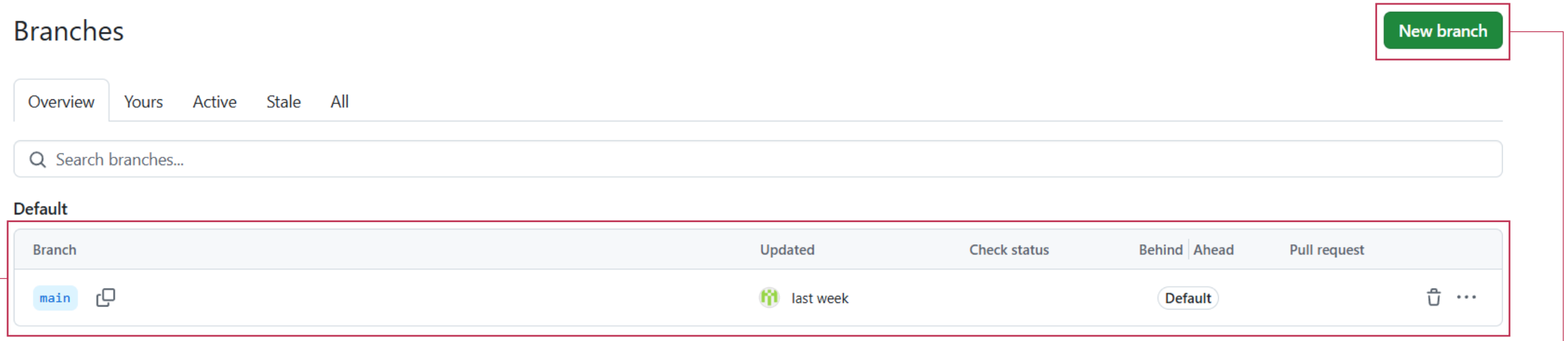
The git checkout command is used in Git to manage branches by switching branches, restoring files, and detaching the HEAD to inspect specific commits or states in a repository.

Command syntax	Description
git checkout branch_name	It switch to a branch. The branch must be existed.
git checkout -b new_branch_name	It create and switch to a branch.
git checkout branch_name -- file_name	It restore a file in the branch.
git checkout commit_hash	It inspect a commit.

The git checkout command has been partially replaced by the git switch and git restore commands.

Branches of code: GitHub

Branches can be created using the GitHub web interface pressing in the branch list button

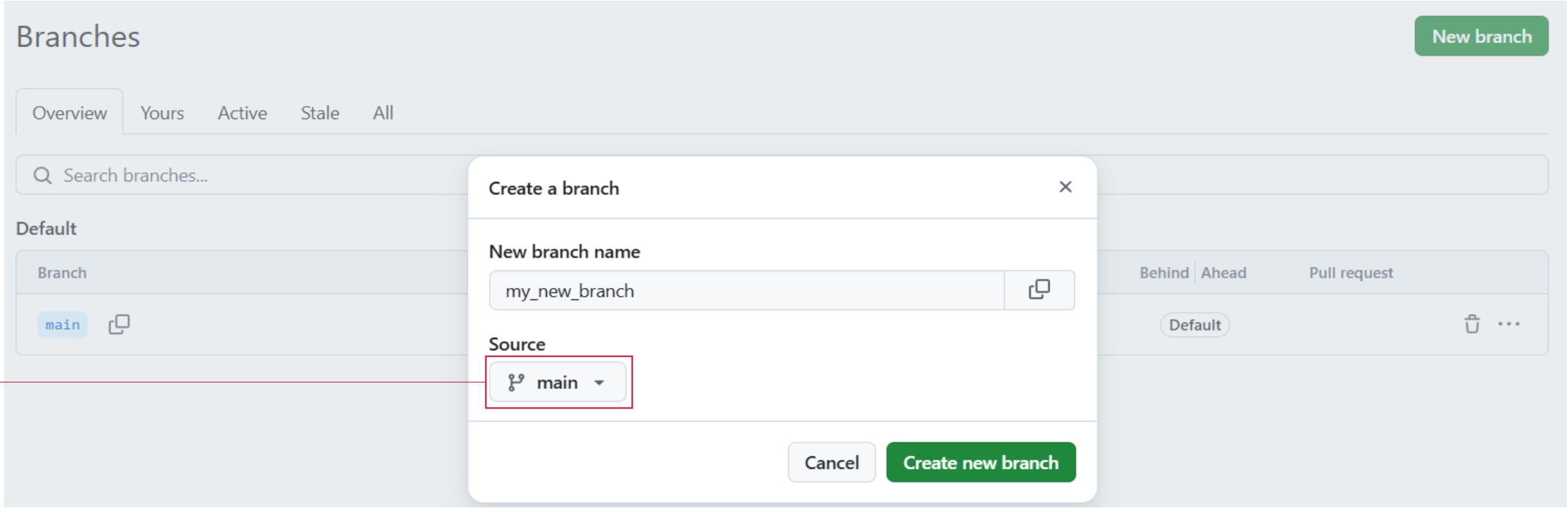


The page provides an overview of the status and activity of all branches in the repository.

New branches can be created by clicking the New Branch button.

Branches of code: GitHub

Branches can be created using the GitHub web interface pressing in the branch list button 



Branches are created from an existing branch, duplicating its content into the new branch as a starting point.

Git – Files

07

Git files

In a Git repository, certain **special files** serve specific purposes to help manage the project, provide documentation, or configure Git's behavior.

- Git ignore file: It specifies files and directories that Git should ignore and not track.
- Readme file: It provides a project overview, including its purpose, installation instructions, usage examples, and other documentation.
- ChangeLog file: It serves as a historical reference and is especially useful for developers and users to understand what has changed between versions of the project.
- License file: It specifies the terms and conditions for using, modifying, and distributing the project.

Git files: README.md

A README.md file is a widely used document in software development projects, especially in open-source projects hosted on platforms like GitHub.

Markdown Input (editable)

```
# Foobar

Foobar is a Python library for dealing with word pluralization.

## Installation

Use the package manager [pip](https://pip.pypa.io/en/stable/) to install foobar.

```bash
pip install foobar
```
```

Rendered

Foobar

Foobar is a Python library for dealing with word pluralization.

Installation

Use the package manager pip to install foobar.

```
pip install foobar
```

It is written in Markdown, a lightweight and flexible markup language that allows for clear, structured text formatting, making it simple to create organized and visually appealing documentation.

Git files: .gitignore

A .gitignore file is a configuration file in a Git repository that specifies intentionally untracked files or directories that Git should ignore.

```
# Prerequisites
*.d

# Object files
*.o
*.ko
*.obj
*.elf
```

The files with extensions like *.o, *.ko, *.obj, and *.elf are typically not included in a Git repository because they are generated files (build artifacts) that are produced during the build or compilation process.

There are general gitignore templates in the official github repository.

https://github.com/github/gitignore/blob/main/C.gitignore

These files are excluded from version control, ensuring they are not accidentally added to the repository.

Git files: .gitignore

The .gitignore file helps keep the repository clean by excluding unnecessary files or sensitive data file such as:

- Temporary files created by editors or IDEs.
- Build artifacts, such as .o files or node_modules/.
- Logs and caches.
- API keys, passwords, credentials or environment variables.

By ignoring these files, the repository focuses only on the important files (e.g., source code and configuration files), making it secure and easier to navigate and manage for developing teams.

Git files: licenses

In Git, a license file is a document that outlines the legal terms and conditions for sharing and using a software project. It defines the permissions, restrictions, and obligations related to the use, modification, distribution, and licensing of the project's source code. Commonly used licenses include popular open-source options such as:

- **MIT License**
- **Apache License 2.0**
- **GNU General Public License (GPL)**
- **Creative Commons Licenses**

The license file is a vital component of open-source projects. It clarifies how others can utilize, modify, and share the project's code while ensuring adherence to legal requirements.