# Introduction to Software Engineering

Bachelor's Degree in Computer Engineering

Academic Year 2024/2025

## Basics of Git & GitHub

Andres.gorostidi@universidadeuropea.es

Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

## Development Tools: Git

Git is a distributed version control system created by Linus Torvalds in 2005. It is a tool that allows developers to manage and record changes made to their source code over time.

## Development Tools: Git

What is it for?

- Version Control: Git allows you to keep a detailed history of all changes to your project, making it easy to retrieve previous versions and compare them.
- Team Collaboration: With Git, multiple developers can work on the same project simultaneously, creating separate branches for each feature or fix, and then merging those changes into the main branch.
- Security and Backups: Because it is distributed, each developer has a complete copy of the project history, which adds an extra layer of security against data loss.

## Development Tools: GitHub

GitHub is a web-based platform that uses Git as a version control system. It provides a centralised space where developers can store their code repositories, collaborate on projects, and share code with the global community.

## Development Tools: GitHub

What is it for?

- Repository storage: GitHub allows you to host Git repositories publicly or privately, facilitating project management and collaboration.
- Collaboration and Contribution: Provides tools for developers to contribute to existing projects through pull requests, code reviews, and issue management.
- Documentation and Tracking: GitHub includes functionality for creating wikis, managing projects with project boards and keeping track of issues and feature requests.

## Development Tools: GitHub

### Correct use of GitHub

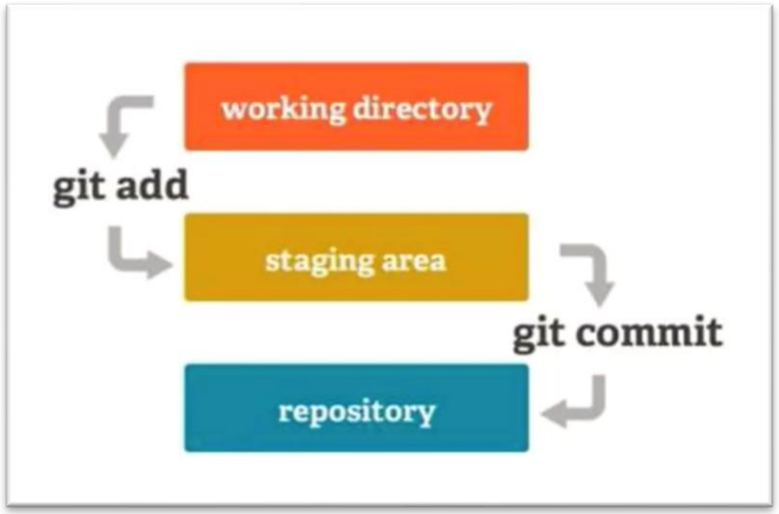
- Create a Repository: Start a new project or clone an existing one on your local machine. You can create a repository from the GitHub interface or via the command line with git init and git remote add origin <URL-of-repository>.
- Create Branches: Branches allow you to work on new features or fixes without affecting the main code. Create a new branch with git branch <branch-name> and switch to it with git checkout <branch-name>.
- Contributing to a Project: When making changes to your branch, use git add to add the changes, git commit -m "Commit message" to commit them, and git push to push those changes to the remote repository. Send a pull request from GitHub for the project maintainers to review and, if appropriate, integrate your changes into the main branch.

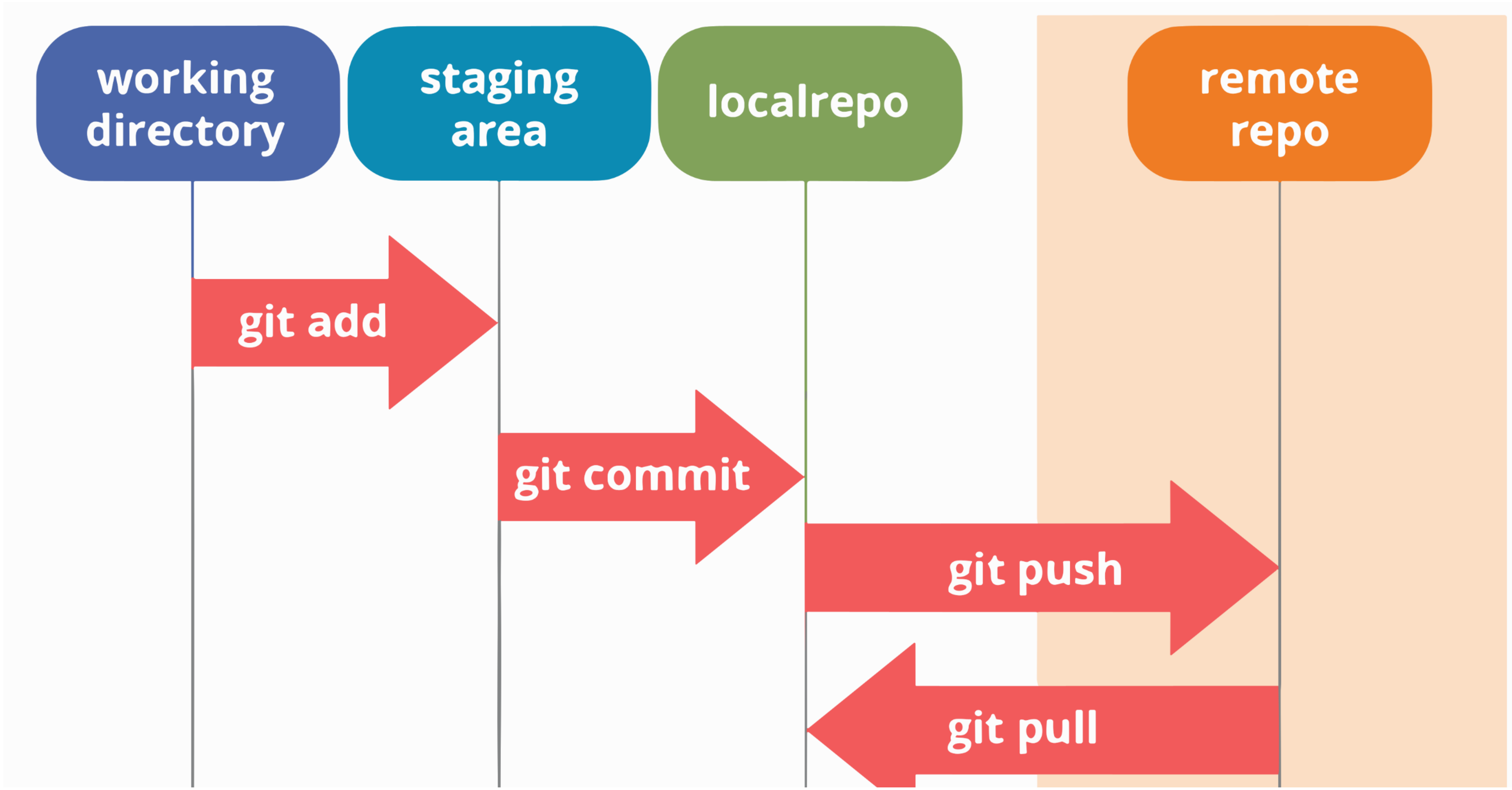
## Development Tools: GitHub

### Basic commands

Git commands allow you to interact with your local and remote repository, manage your source code, and collaborate with other developers.

Learning these basic commands will give you the control you need to manage your projects effectively on GitHub.




working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull

## Development Tools: GitHub

### Basic commands

- git clone: Clone a remote repository on your local machine (git clone <URL-of-repository>).
- git init: Initialises a new Git repository in your current directory. This is necesary if you are not going to clone a directory, so you can start to use git localy, and later on, add the remote repository using the command git remote add
- git status: Displays the status of files in your working directory and staging area.
- git add: Add file changes to the staging area - (git add <filename> or git add).
- git commit: Saves changes from the staging area to the project history (git commit -m "Commit description message"). Note: if you use the flag –a on git commit, you do automatically a git add.

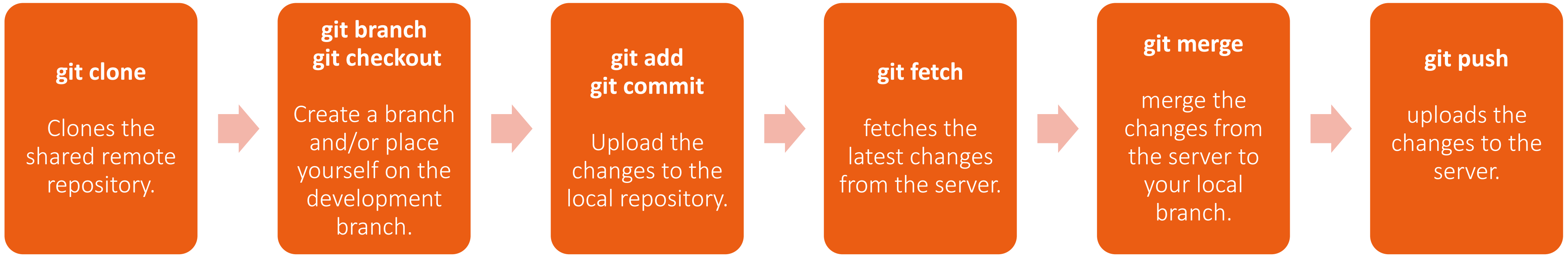
## Development Tools: GitHub

### Basic commands

- git push: Upload local commits to a remote repository (git push origin <branchname>).
- git pull: Update your local repository with changes from the remote repository (git pull origin <branchname>).
- git branch: List, create or delete branches (git branch to list, git branch <branch-name> to create a new branch).
- git checkout: Switch to a specific branch or check out a specific file in the history (git checkout <branch-name> or git checkout <commit-id> <file>).
- git merge: Merge changes from a branch into the current branch (git merge <branch-name>).

## Development Tools: GitHub

How to collaborate on a Git project?



**git clone**

Clones the shared remote repository.

**git branch**
**git checkout**

Create a branch and/or place yourself on the development branch.

**git add**
**git commit**

Upload the changes to the local repository.

**git fetch**

fetches the latest changes from the server.

**git merge**

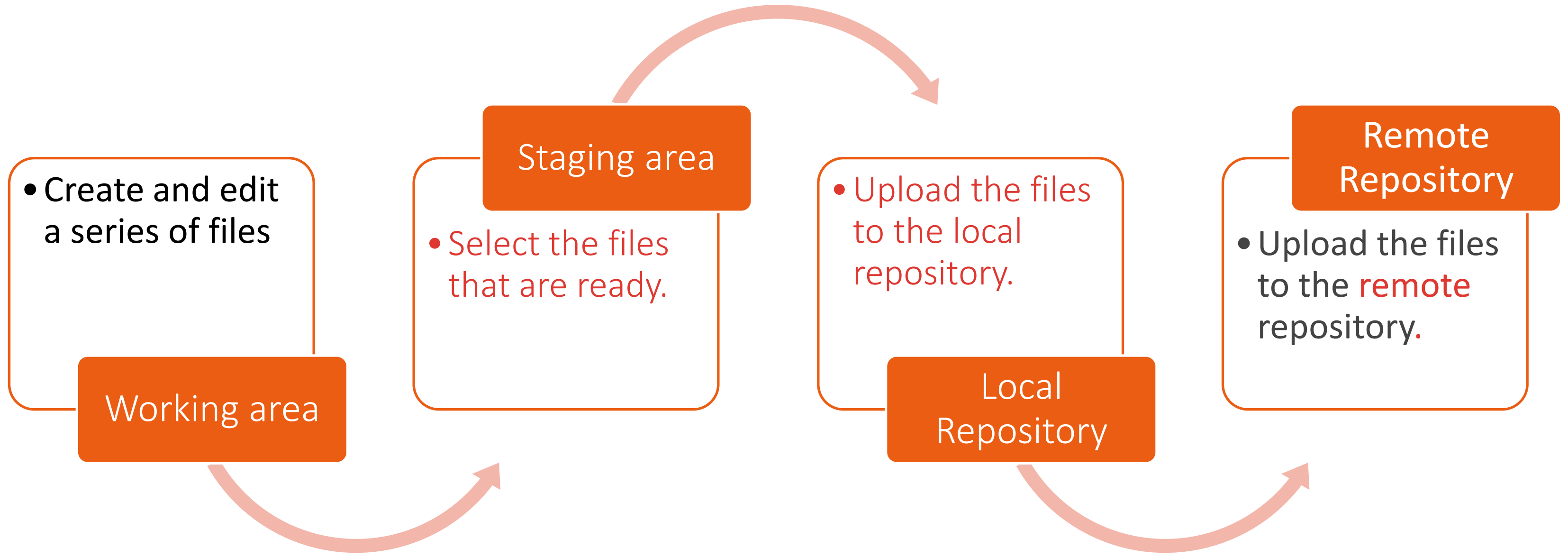
merge the changes from the server to your local branch.

**git push**

uploads the changes to the server.

## Development Tools: GitHub

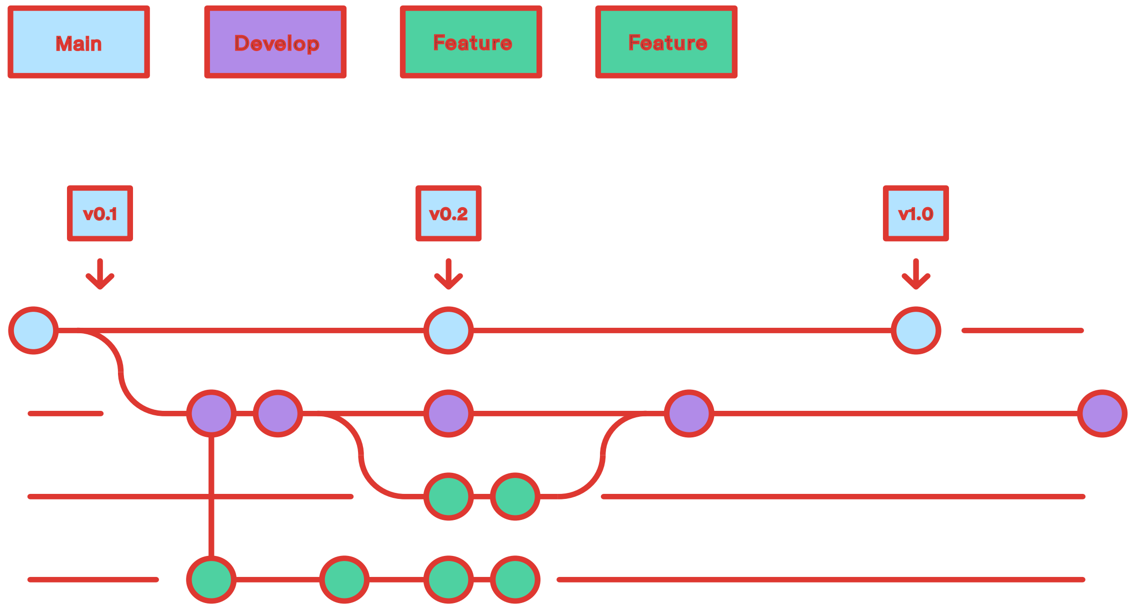
### Workflow


• Create and edit a series of files
Working area
Staging area
• Select the files that are ready.
• Upload the files to the local repository.
Local Repository
Remote Repository
• Upload the files to the remote repository.

## Development Tools: GitHub

### Workflow (Git Flow)

The Git Flow workflow developed by Vincent Driessen is a structured model of Git usage that organises the development process into different branches to manage development, release versions and bug fixes efficiently.



## Development Tools: GitHub

### Workflow structure (Git Flow)

Main Branches:

- master (or main): This is the main branch that always contains the stable production code. Any final version that is deployed to production must be in this branch.
- develop: This is the main development branch where all new features are integrated before a release. It is the basis for the creation of feature branches and release branches.

## Development Tools: GitHub

### Workflow structure (Git Flow)

Support Branches:

- feature/: Branches for developing new features. They are created from develop and merged back into develop when the functionality is complete.
- release/: Branches created when the code in develop is ready for a new version. Here final details are polished and minor bugs are fixed. They are then merged into master (for release) and develop.
- hotfix/: Branches to fix critical bugs in production. They are created from master, the problem is fixed, and merged into both master and develop so that the fix is in future releases.

## Development Tools: GitHub

### Work process (Git Flow)

Support Branches:

- Feature development: A developer starts by creating a feature branch from develop. He works on the new functionality in this branch. Upon completion, he performs a merge of the feature branch into develop.
- Preparing a Release: When it is decided that the code in develop is ready to be released, a release branch is created. Final tests and minor fixes are performed on this branch. Once ready, the release branch is merged into master and develop. The master branch is tagged with the corresponding version.
- Critical Bug Fixes: If a critical bug arises in production, a hotfix branch is created from master. The bug is corrected and the hotfix branch is merged into both master and develop.

## Other important GitHub Issues

- Take care of your profile
- MarkDown Language: You should have a basic understanding of the format of the MD files (ie, README.MD) and the formating of such a files
- Clone vs Init localy: You may start to work on GitHub from an existing local directory or from scratch (creating new one). On the first case, you should always do a git init and add the remote directory. On the second case, you start on github, and clone it locally, so git init is not necesary neither define remote directories.

.

## Different ways to create a repository

| Create a GitHub Repository from Your Local Directory | Create a GitHub Repository on GitHub and Clone it Locally |
|---|---|
| cd path/to/your/local-directory<br>git init<br>git add .<br>git commit -m "Initial commit"<br>git remote add origin [GitHub repository URL]<br>git push -u origin main | git clone [GitHub repository URL] cd repository-name<br>git add .<br>git commit -m "Initial commit“<br>git push |

## PRE-ELIMINARY EXERCISE ( BASIC USAGE OF GIT)

- Installing Git Bash Tools (Go to https://git-scm.com/)
- Create a GitHub Account
- Create a Repository on GitHub
- Clone a Repository localy (see https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository)
- Access to your local working directoty, where you repository was cloned
- Create 1 addtional new file
- Add that file to the stage área (git add file)
- Commit the changes (git commit)
- Upload the changes (git push)

## EXERCISE B - MERGING AND BRANCHES

Clone Exercise &  Additional Doc from  -> https://github.com/agorosti/GitHub-Training/

Universidad Europea

# Introduction to Software Engineering

Academic Year 2024/2025

**Andrés Gorostidi Pulgar**

Andres.gorostidi@universidadeuropea.es

Go further!

© Copyright Universidad Europea. Todos los derechos reservados

Questions

Universidad Europea Madrid

LAUREATE INTERNATIONAL UNIVERSITIES