

# Introduction to Software Development

MODULE 4 / UNIT 2 / 0.4

MOISES M. MARTINEZ

FUNDAMENTALS OF COMPUTER ENGINEERING

Which development tools  
do you know?

# Software development

# 01

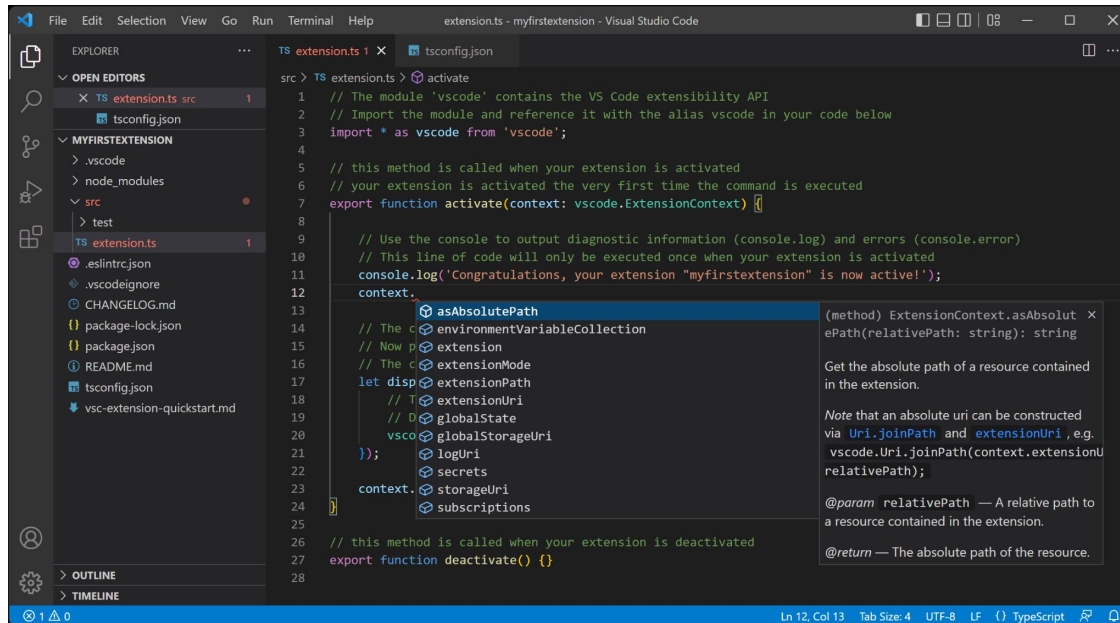
A **development tool**, also known as a software development tool or programming tool, refers to any computer program or application that developers use to create, debug, maintain, or support other programs and applications. Some common types of development tools are:

- Integrated Development Environments (IDEs).
- Text Editors.
- Version Control Systems (VCS) like Git.
- Debuggers.

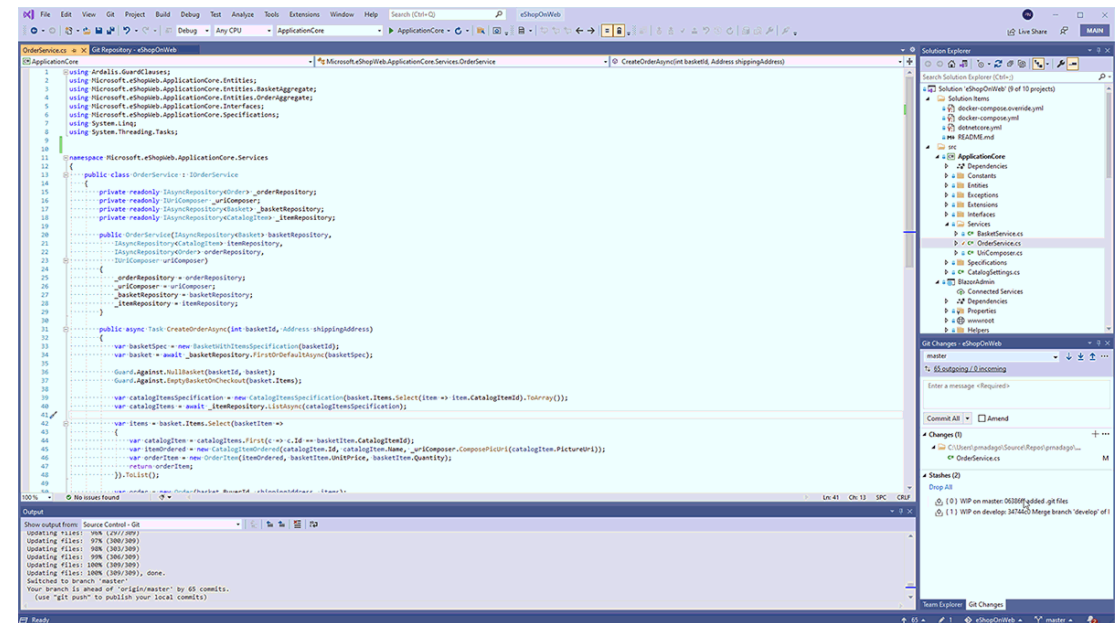


# What is an IDE?

An Integrated Development Environment (IDE) is a type of application software that provides a set of tools and features to facilitate and streamline the process of software development.



```
src > TS extensions > activate
1 // The module 'vscode' contains the VS Code extensibility API
2 // Import the module and reference it with the alias vscode in your code below
3 import * as vscode from 'vscode';
4
5 // this method is called when your extension is activated
6 // your extension is activated the very first time the command is executed
7 export function activate(context: vscode.ExtensionContext) {
8
9     // Use the console to output diagnostic information (console.log) and errors (console.error)
10    // This line of code will only be executed once when your extension is activated
11    console.log('Congratulations, your extension "myfirstextension" is now active!');
12    context.subscriptions
13
14    // The vscode.env variableCollection
15    // Now p extension
16    // The vscode.ExtensionContext
17    let disposable = vscode.commands.registerCommand('extension.helloWorld', () => {
18        // This function is called when the command is executed
19        // Use the console to output diagnostic information (console.log) and errors (console.error)
20        // This line of code will only be executed once when your extension is activated
21        console.log('Congratulations, your extension "myfirstextension" is now active!');
22    });
23    context.subscriptions.push(disposable);
24
25    // this method is called when your extension is deactivated
26    export function deactivate() {}
27
28 }
```

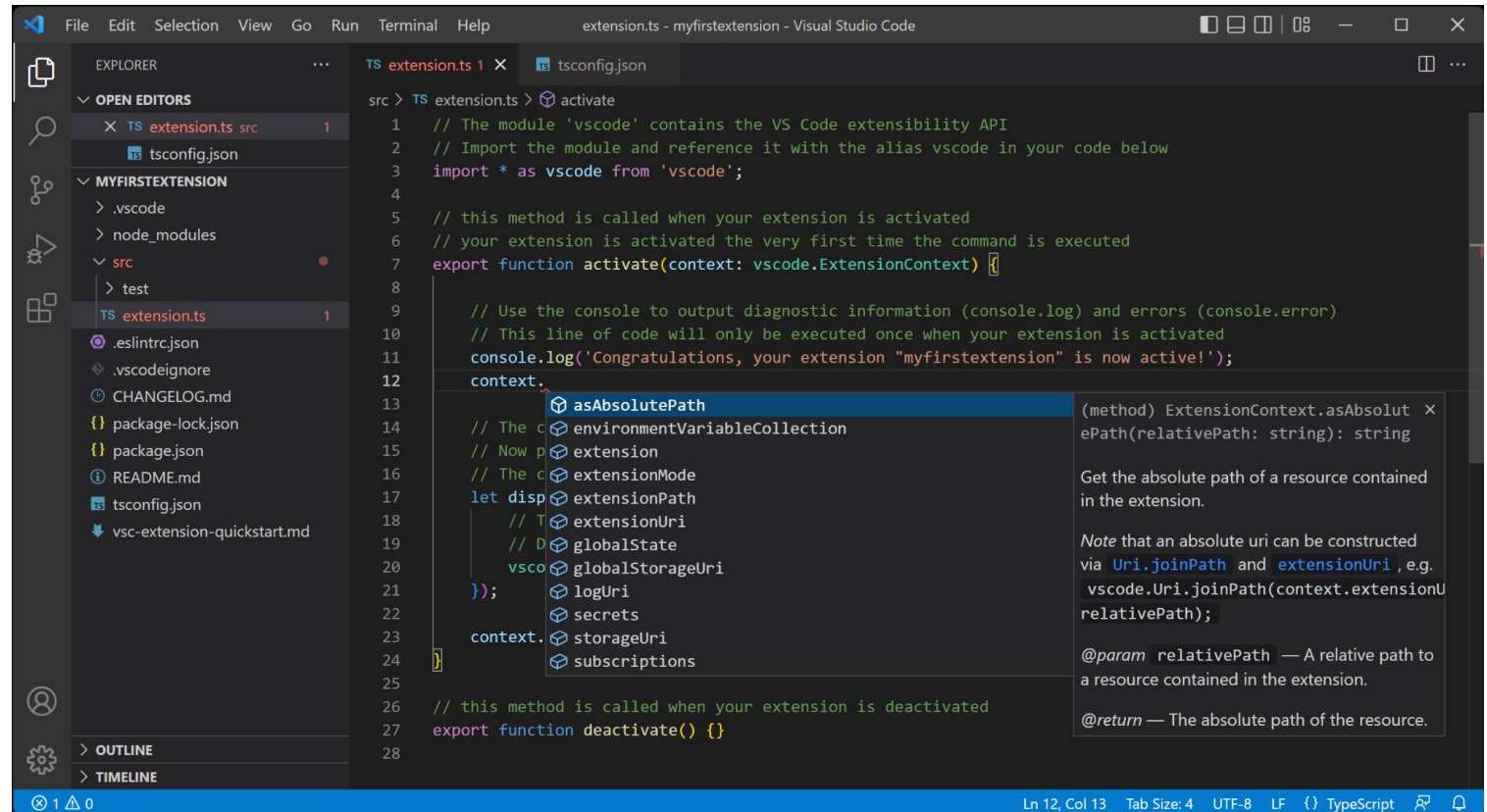


```
src > TS extensions > activate
1 // The module 'vscode' contains the VS Code extensibility API
2 // Import the module and reference it with the alias vscode in your code below
3 import * as vscode from 'vscode';
4
5 // this method is called when your extension is activated
6 // your extension is activated the very first time the command is executed
7 export function activate(context: vscode.ExtensionContext) {
8
9     // Use the console to output diagnostic information (console.log) and errors (console.error)
10    // This line of code will only be executed once when your extension is activated
11    console.log('Congratulations, your extension "myfirstextension" is now active!');
12    context.subscriptions
13
14    // The vscode.env variableCollection
15    // Now p extension
16    // The vscode.ExtensionContext
17    let disposable = vscode.commands.registerCommand('extension.helloWorld', () => {
18        // This function is called when the command is executed
19        // Use the console to output diagnostic information (console.log) and errors (console.error)
20        // This line of code will only be executed once when your extension is activated
21        console.log('Congratulations, your extension "myfirstextension" is now active!');
22    });
23    context.subscriptions.push(disposable);
24
25    // this method is called when your extension is deactivated
26    export function deactivate() {}
27
28 }
```

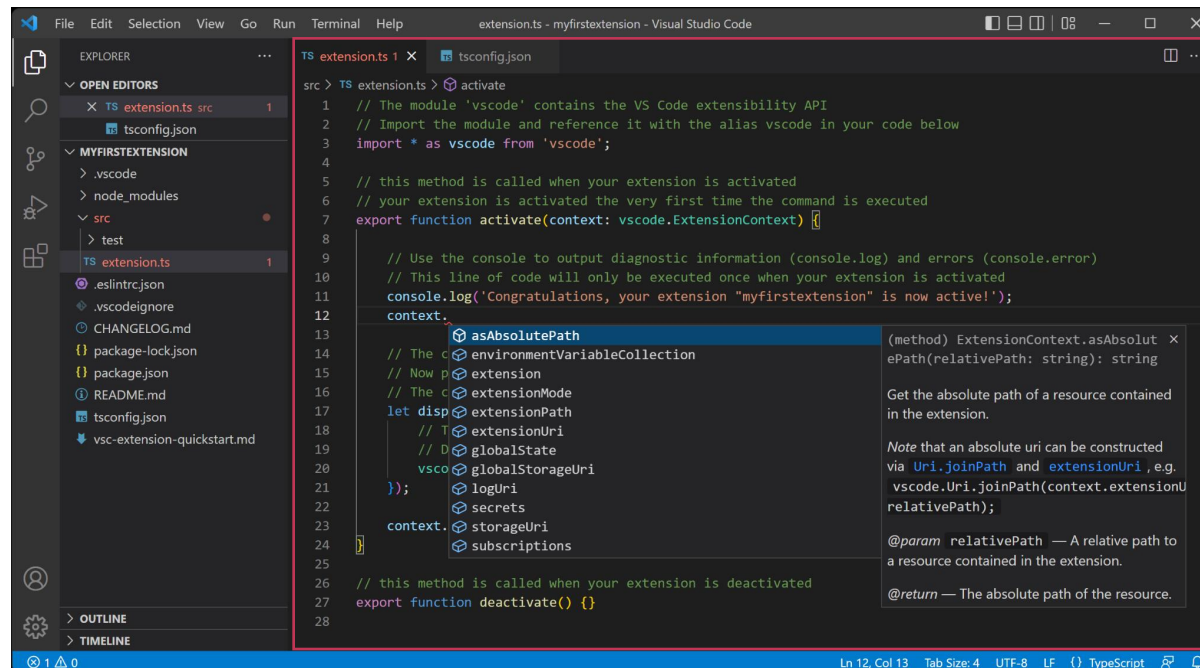
IDEs are designed to be a central hub where developers can write, edit, compile, debug, and manage their code in a unified environment.

The main features commonly found in an IDE are:

- Source Code Editor.
- Compiler/Interpreter.
- Debugger.
- Build Automation tools.
- Version Control Integration.
- Project Management tools.



The source code editor is a text editor with advanced functionalities specifically tailored for programming. It often includes features like:

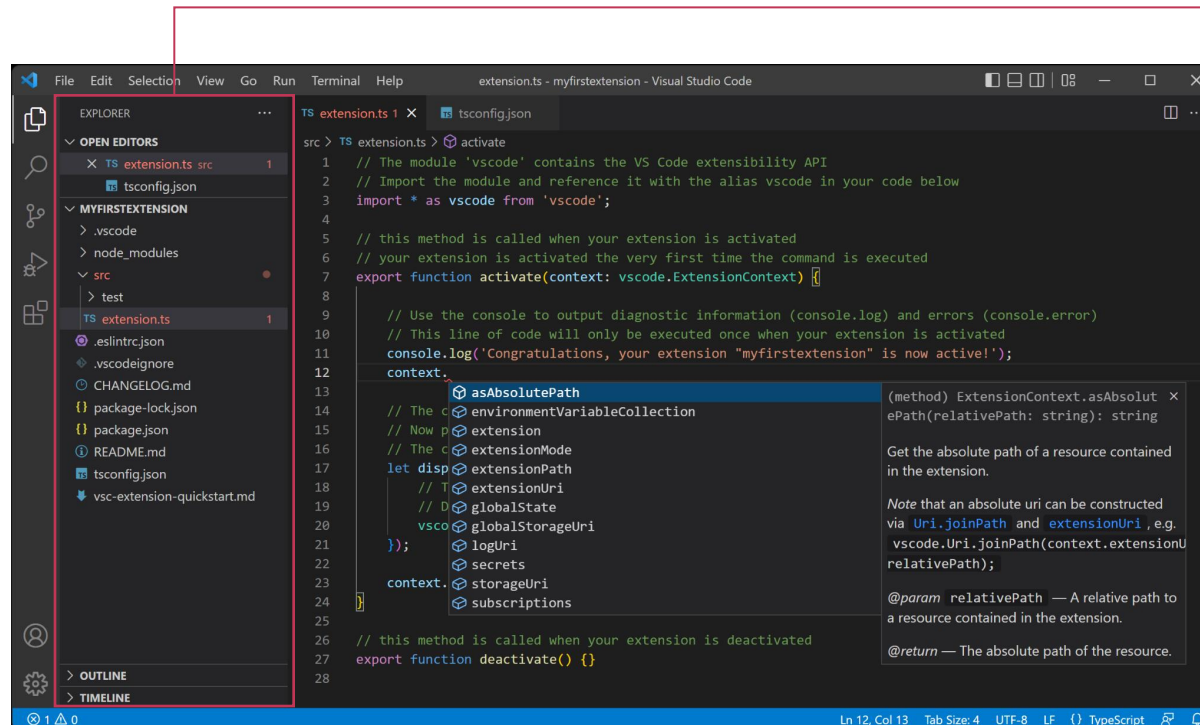


- Syntax highlighting
- Code completion
- Code folding

**These features enhance code readability, streamline development, and improve productivity.**



The Project Management tools include features for managing projects, organizing files, and handling dependencies.



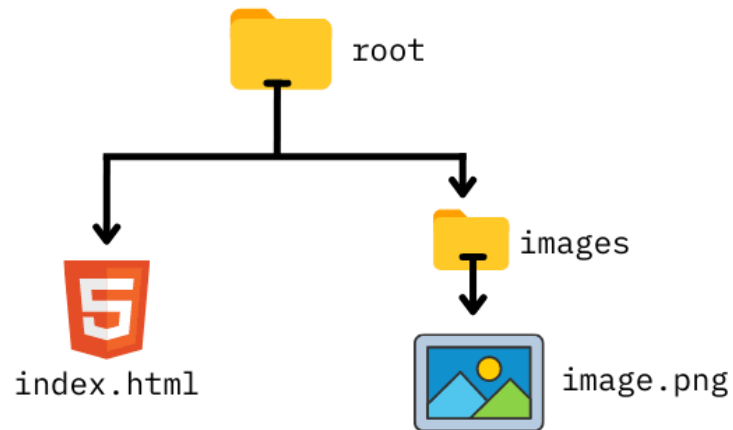
extension.ts is a typescript file.

The IDE links directories stored in secondary memory (such as HDD) as projects, encompassing various file types.

The Project Management tools include features for managing projects, organizing **files**, and handling dependencies. It is essential to define clear and structured paths to locate and manage these files effectively.

- An absolute path specifies the exact location of a file or folder starting from the root directory of the file system.
- A relative path defines the location of a file or folder in relation to the current working directory. There are a few key conventions to keep in mind when defining them:
  - / represents the root of the current drive. For example, on Windows operating systems, C:/ denotes the root of the C drive.
  - ./ refers to the current directory.
  - ../ indicates the parent directory of the current directory.

The Project Management tools include features for managing projects, organizing **files**, and handling dependencies.



This example shows a basic HTML web page (index.html) that includes an image (image.png).

The image is referenced using a relative path, showcasing how to link resources within a project directory.

```

```

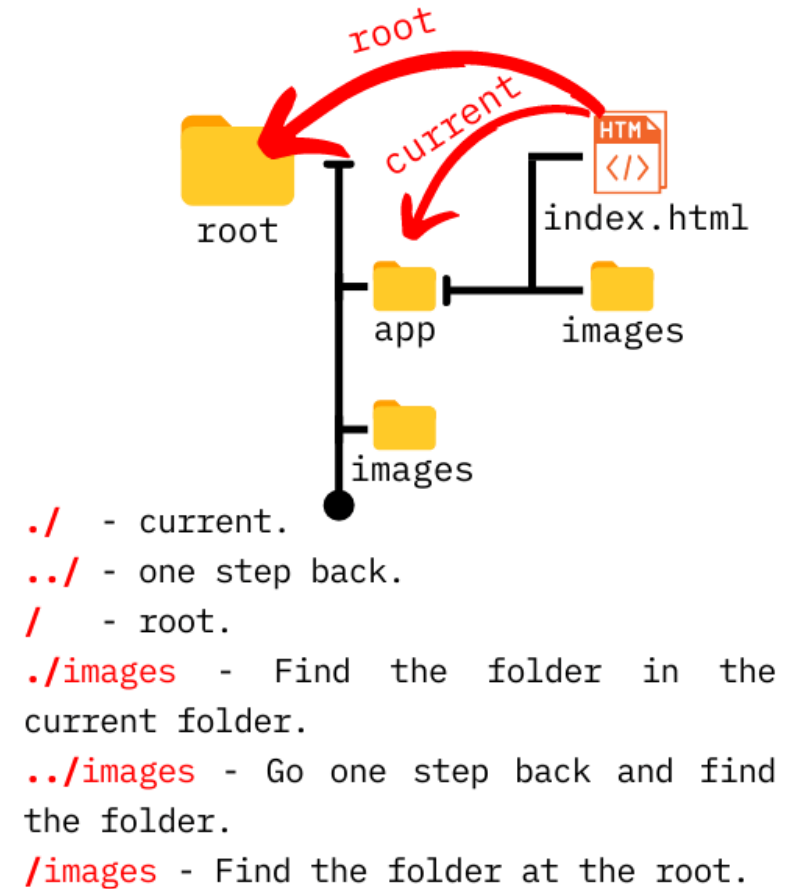
**OR**

```

```

This example outlines the configuration of an HDD with its home directory set as the root.

- A web application is being hosted inside the root folder.
- The HDD contains different image folders:
  - one folder for general images.
  - another file specifically designated for web application. The one inside app folder.

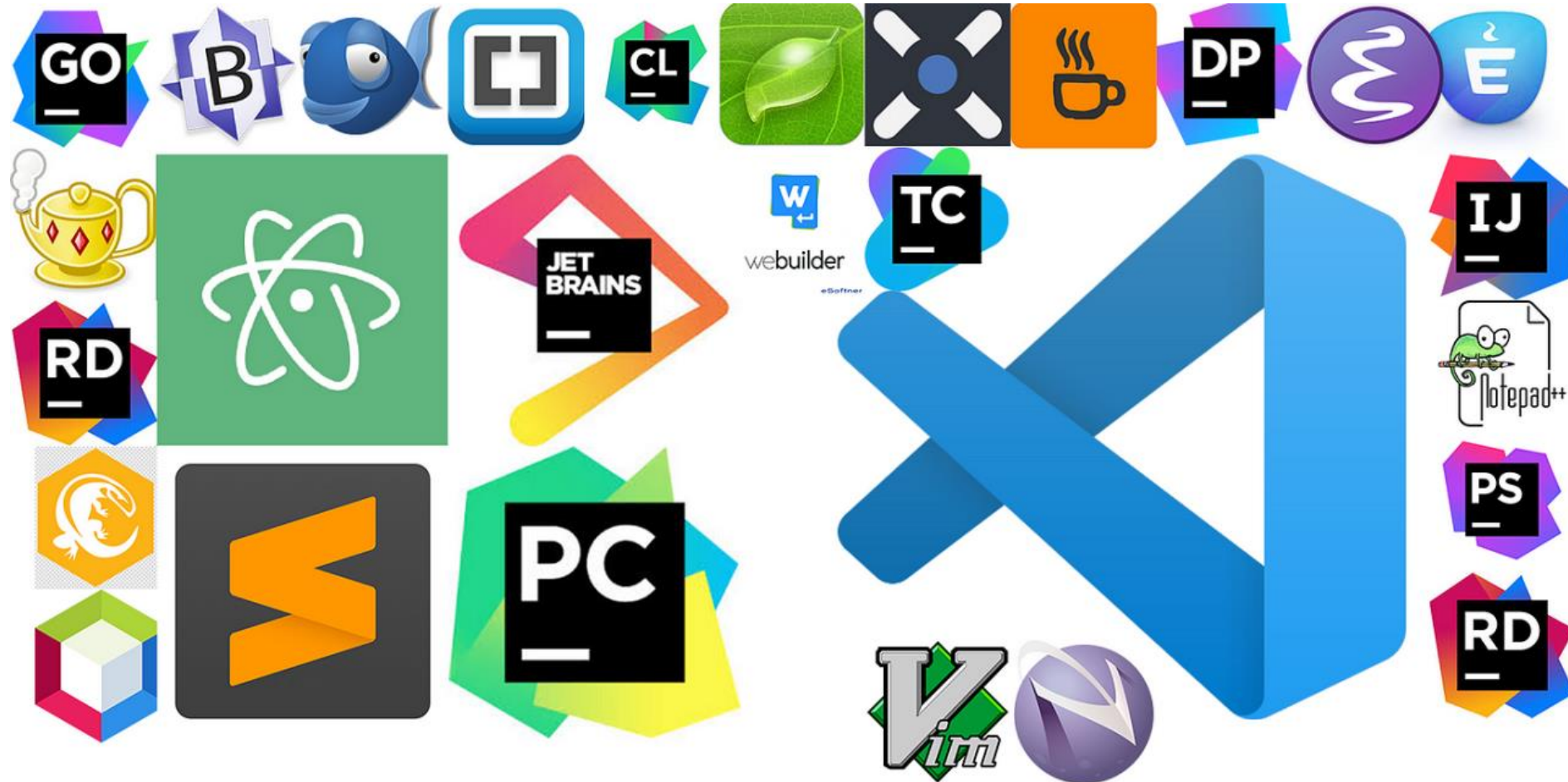


The Compiler/Interpreter is a component responsible for translating the source code written by a programmer into a format that the computer can execute. The specific role depends on the programming language and the nature of the code.

- A compiler translates the entire source code of a program into machine code or an intermediate code all at once. Compilation is typically a separate step before running the program.
- An interpreter translates the source code into machine code or an intermediate code line by line as the program is executed. The interpretation happens in real-time during program execution, and there is no separate compilation step.

**In some cases, a language may use a combination of compilation and interpretation. For example, Java programs are initially compiled into an intermediate bytecode, and then the Java Virtual Machine (JVM) interprets or compiles this bytecode at runtime.**

Some of the most popular programming IDEs:



Some of the most popular programming IDEs:

- Eclipse is a widely-used IDE for Java development, but it supports many other languages through plugins.
- Jupyter notebook is an open-source IDE that is used to create Jupyter documents that can be created and shared with live codes.
- PyCharm is an IDE specifically designed for Python development, with features like code completion, debugging, and support for web development frameworks.
- Visual Studio Code, often abbreviated as VS Code, is a free open-source source IDE developed by Microsoft. This one of the most common IDEs used by web apps.

# Team development using Git

# 02



In the realm of modern software development, **collaboration** is key to building robust, scalable, and high-quality applications.



In the realm of modern software development, **collaboration** is key to building robust, scalable, and high-quality applications. There are some good practices for Git collaboration that help ensure a smooth and efficient workflow:

- Collaborative branching model.
- Pull Requests (PRs) or Merge Requests (MRs).
- Code reviews from your team members.
- Commit guidelines to clarify the changes in the code.
- Conflict resolution strategy.

## Collaborative branching model

The collaborative branching model is a structured approach to managing code changes in a Git repository. It allows teams to work on multiple features, fixes, or experiments simultaneously without interfering with the stability of the main codebase. These model define the naming of the branches:

- Main Branch, often named main or master, serves as the stable and deployable version of the codebase.
- Develop Branch, often named dev or develop, acts as an integration branch where completed features and fixes are merged before they are finalized for release.
- Feature Branches, are dedicated branches created for implementing new features or enhancements. These branches usually follows a naming conventions like feature/<feature-name>.
- Bug-Fix Branches, used for resolving specific issues in the codebase. These branches usually follows a naming conventions like fix/<bug-description> or hotfix/<description>.

## Collaborative branching model

The collaborative branching model is a structured approach to managing code changes in a Git repository. It allows teams to work on multiple features, fixes, or experiments simultaneously without interfering with the stability of the main codebase. These model define the naming of the branches:

- Main Branch, often named main or master, serves as the stable and deployable version of the codebase.
- Develop Branch, often named dev or develop, acts as an integration branch where completed features and fixes are merged before they are finalized for release.
- Feature Branches, are dedicated branches created for implementing new features or enhancements. These branches usually follows a naming conventions like feature/<feature-name>.
- Bug-Fix Branches, used for resolving specific issues in the codebase. These branches usually follows a naming conventions like fix/<bug-description> or hotfix/<description>.

## Pull requests

In Git, a pull request (PR) is an important feature that facilitates collaboration by allowing developers to propose changes to a codebase.

1. A developer creates a new branch to work on changes.
2. After completing the changes, they push the branch to the remote repository.
3. The developer then opens a pull request to merge the branch into another (e.g., main).
4. Team members can review the pull request, comment on specific lines of code, suggest improvements, or request changes.
5. Once the pull request is approved and all tests pass, it can be merged into the target branch.
6. The merging process integrates the changes into the main codebase.

It serves as a way to review, discuss, and merge changes from one branch to another, typically into a main or production branch. Here's an overview:

## Pull requests

In Git, a pull request (PR) is an important feature that facilitates collaboration by allowing developers to propose changes to a codebase.

1. A developer creates a new branch to work on changes.
2. After completing the changes, they push the branch to the remote repository.
3. The developer then opens a pull request to merge the branch into another (e.g., main).
4. Team members can review the pull request, comment on specific lines of code, suggest improvements, or request changes.
5. Once the pull request is approved and all tests pass, it can be merged into the target branch.
6. The merging process integrates the changes into the main codebase.

It serves as a way to review, discuss, and merge changes from one branch to another, typically into a main or production branch. Here's an overview:

## Conflict resolution strategy.

A conflict resolution strategy refers to the systematic approach developers use to handle merge conflicts in Git.

**Merge conflicts occur when changes made in one branch conflict with changes in another branch, preventing Git from automatically merging the branches.**

These conflicts usually happen when:

- Multiple team members edit the same line of a file.
- A file is modified in one branch and deleted in another.
- Structural changes, such as file renaming, clash across branches.

## Conflict resolution strategy – Merge in Git

A merge in Git is the process of combining changes from two or more branches into a single branch. It is a key operation for collaborative development, as it integrates work from multiple developers or branches into a unified codebase.

The git merge command is used in Git to integrate changes from one branch into another.



```
git merge <branch>
```



**The name of the branch you want to merge into the current branch.**



## Conflict resolution strategy – how to solve them?

1. Identify the conflict. Git always notifies you of a conflict during a merge or rebase.



```
git status
```

The git status command is used in Git to provides an overview of the current state of the working directory and the staging area. This command is essential for understanding what actions are needed before committing changes to the repository.

## Conflict resolution strategy – how to solve them?

1. Identify the conflicts. Git always notifies you of a conflict during a merge or rebase.




```
git status
```

The git status command is used in Git to provides an overview of the current state of the working directory and the staging area. This command is essential for understanding what actions are needed before committing changes to the repository.

## Conflict resolution strategy – how to solve them?

1. Identify the conflicts. Git always notifies you of a conflict during a merge or rebase.
2. Understand the conflicts. Files with conflicts contain special markers to help identify the conflicting sections.



```
<<<<<< HEAD
Your changes in the current branch
=====
Changes from the branch you are merging
>>>>>> branch_name
```

Markers indicate portions of a file where Git is unable to resolve differences between changes made in two branches.

## Conflict resolution strategy – how to solve them?

1. Identify the conflicts. Git always notifies you of a conflict during a merge or rebase.
2. Understand the conflicts. Files with conflicts contain special markers to help identify the conflicting sections.
3. Resolve the conflicts. Manually edit the file to merge the conflicting changes; we can choose one version, combine both changes, or rewrite the section to satisfy both requirements.

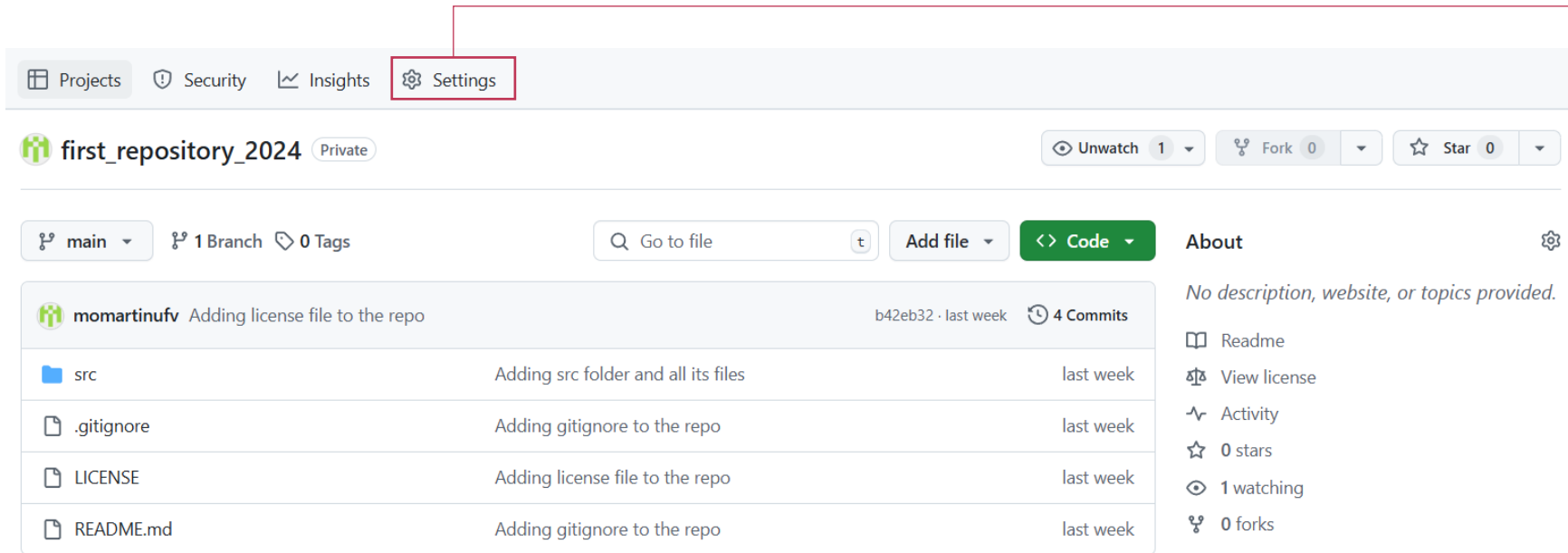
## Conflict resolution strategy – how to solve them?

1. Identify the conflicts. Git always notifies you of a conflict during a merge or rebase.
2. Understand the conflicts. Files with conflicts contain special markers to help identify the conflicting sections.
3. Resolve the conflicts. Manually edit the file to merge the conflicting changes; we can choose one version, combine both changes, or rewrite the section to satisfy both requirements.
4. Mark conflict as resolved pushing the files into the repository.

```
git add file1 file2
git commit -m "Conflicts solved in files ...."
git push
```

## Collaborators

In GitHub, collaborators are individuals who have been granted direct access to a repository by its owner. They play a key role in contributing to and managing the repository's content.



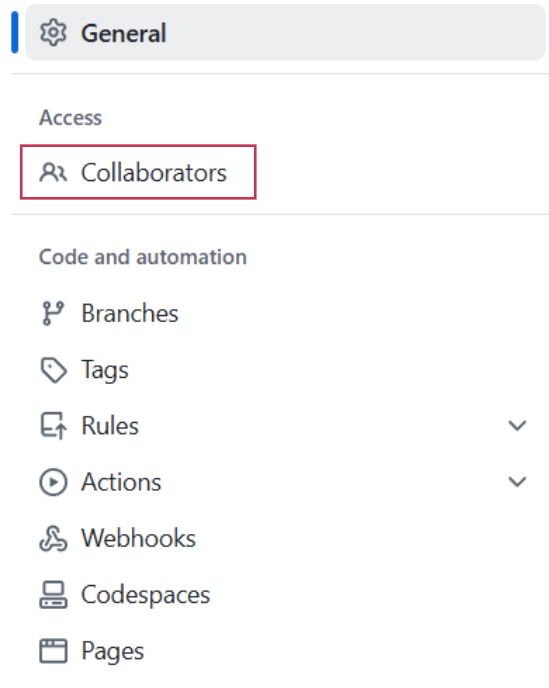
The screenshot shows the GitHub interface for a repository named 'first\_repository\_2024'. The 'Settings' tab is highlighted with a red box. A red arrow points from this box to the text on the right. The repository page displays a commit history table with the following data:

Commit Message	Author	Date
Adding license file to the repo	momartinufv	b42eb32 · last week
Adding src folder and all its files		last week
Adding gitignore to the repo		last week
Adding license file to the repo		last week
Adding gitignore to the repo		last week

We can manage collaborators in a repository in the settings section.

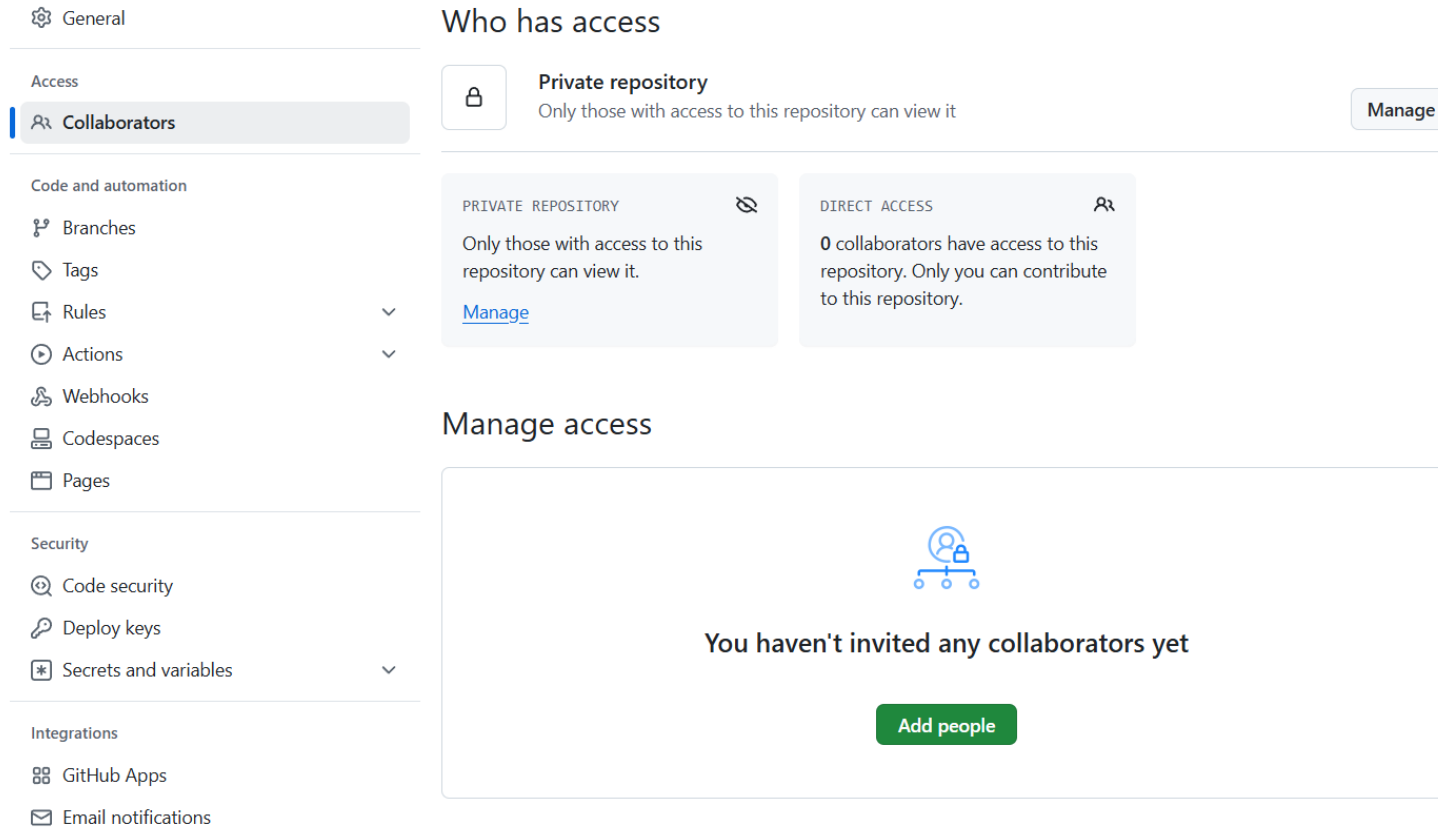
## Collaborators

In GitHub, collaborators are individuals who have been granted direct access to a repository by its owner. They play a key role in contributing to and managing the repository's content.



- Repository owners can invite collaborators by navigating to the repository settings.
- The repository owner can assign one of several permission levels to the collaborators:
  - Read permissions to allow them to view and clone the repository but not making any changes.
  - Write permissions to allow them to push changes to the repository.
  - Maintain permissions to allow them to moderate issues and manage repository settings.
  - Admin permissions providing full access to manage the repository, including settings, collaborators, and branches.

## Collaborators



The screenshot shows the GitHub repository settings page for a repository. The left sidebar contains a navigation menu with sections: General, Access, Code and automation, Security, and Integrations. The 'Access' section is expanded, showing 'Collaborators' as the selected option. The main content area is titled 'Who has access' and shows the repository is 'Private repository'. It indicates that only those with access can view it. Below this, there are two cards: 'PRIVATE REPOSITORY' and 'DIRECT ACCESS'. The 'PRIVATE REPOSITORY' card states 'Only those with access to this repository can view it.' and has a 'Manage' link. The 'DIRECT ACCESS' card states '0 collaborators have access to this repository. Only you can contribute to this repository.' and has a collaborator icon. Below these cards is a section titled 'Manage access' which contains a message: 'You haven't invited any collaborators yet' with a green 'Add people' button.

General

Access

**Collaborators**

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Codespaces

Pages

Security

Code security

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Who has access

Private repository

Only those with access to this repository can view it

Manage

PRIVATE REPOSITORY

Only those with access to this repository can view it.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

Add people

## How to add a collaborator

1. Press on **Add people** button in the repository settings.
2. Enter the username or email address of the person you want to invite.
3. Click Add collaborator and wait for the invitee to accept the invitation.
4. The invited collaborator will receive an email with some instructions to join the repository.
5. Once the collaborators accept the invitation, they will be gain access to contribute to the repository.

Owners can revoke access or adjust permissions at any time from the settings page.



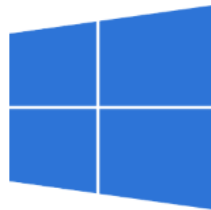
# Software development using VS code

# 03

## Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

<https://code.visualstudio.com/download>



↓ Windows

Windows 10, 11

User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE

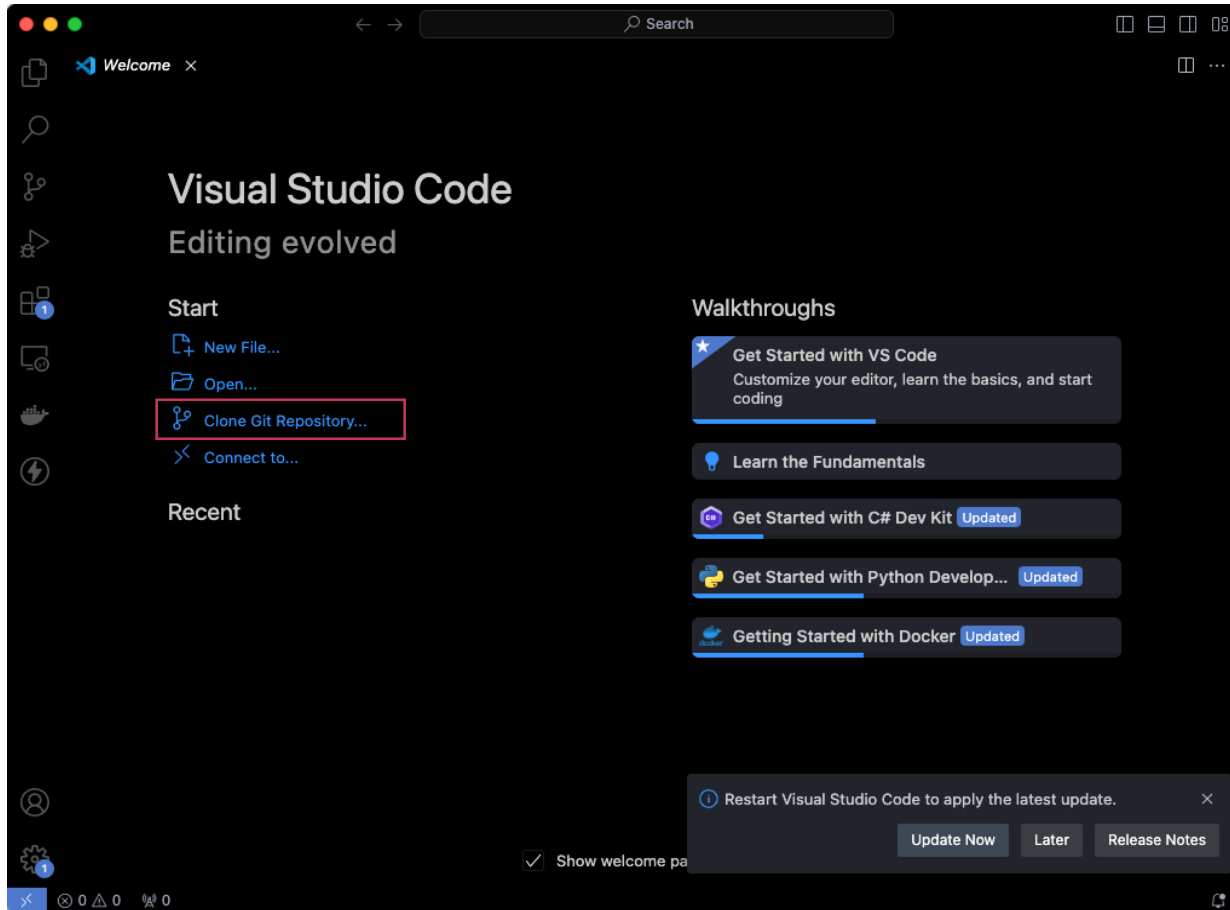
.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64



↓ Mac

macOS 10.15+

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	



We can directly clone a repository from Visual Studio, making it easy to set up and start working with the code immediately.

Remember, you must configure your Git username and email, If you have not done before:

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

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

## Visual studio code – Cloning a repo

The easiest way to connect your GitHub account with visual studio code is to clone a repository from the terminal.


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

This action will open a window in your browser that syncs your account with the IDE. From this time you will be able to pull and push on the repositories associated with your user.

## Visual studio code – Cloning a repo

Certain issues may arise depending on your Visual Studio Code configuration:

- Error 1 - user errors



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


→ **This is your GitHub username**

If you have different GitHub users, you can try to clone a repository using a wrong user. To solve this problem you can force the user in the git clone command.

## Visual studio code – Cloning a repo

Certain issues may arise depending on your Visual Studio Code configuration:

- Error 2 – username and email identification



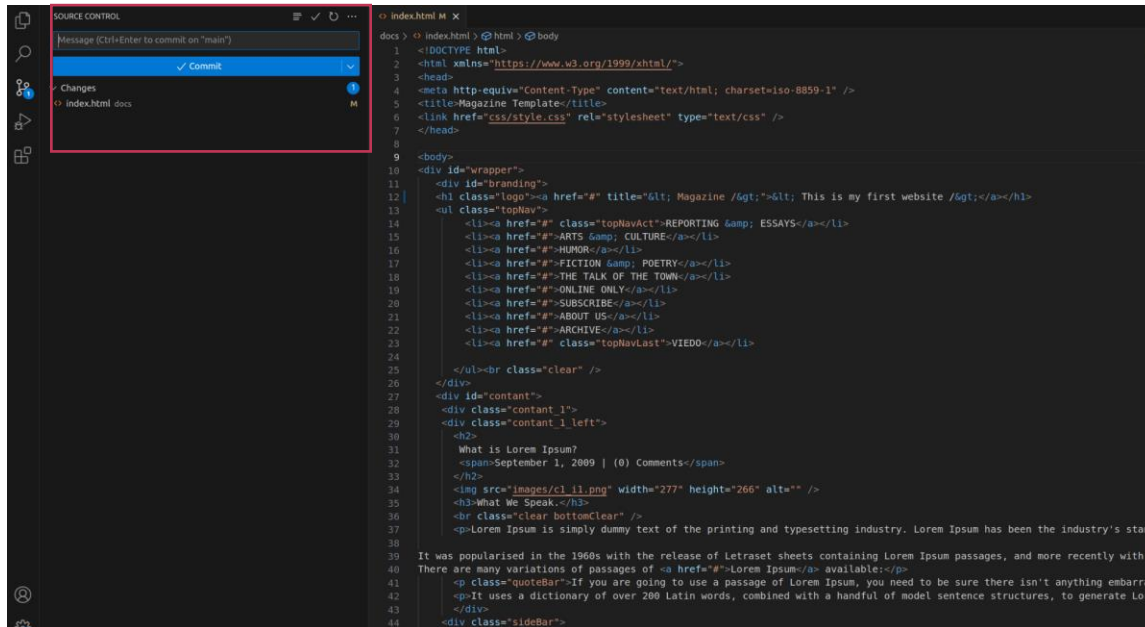
```
git config --global user.name "My super name"
git config --global user.email "username@alumnos.ufv.es"
```

Ensure the email matches the one linked to your GitHub or remote repository account to avoid authentication errors.

If your email and username are not correctly configured, you may encounter issues when attempting to push to the repository.

## Visual studio code – Using a repo

IDEs typically provide visual tools for managing source code with Git. Visual Studio Code includes a dedicated **Source Control** section in the left sidebar, allowing you to track changes, stage files, and execute operations like commits and pushes seamlessly.

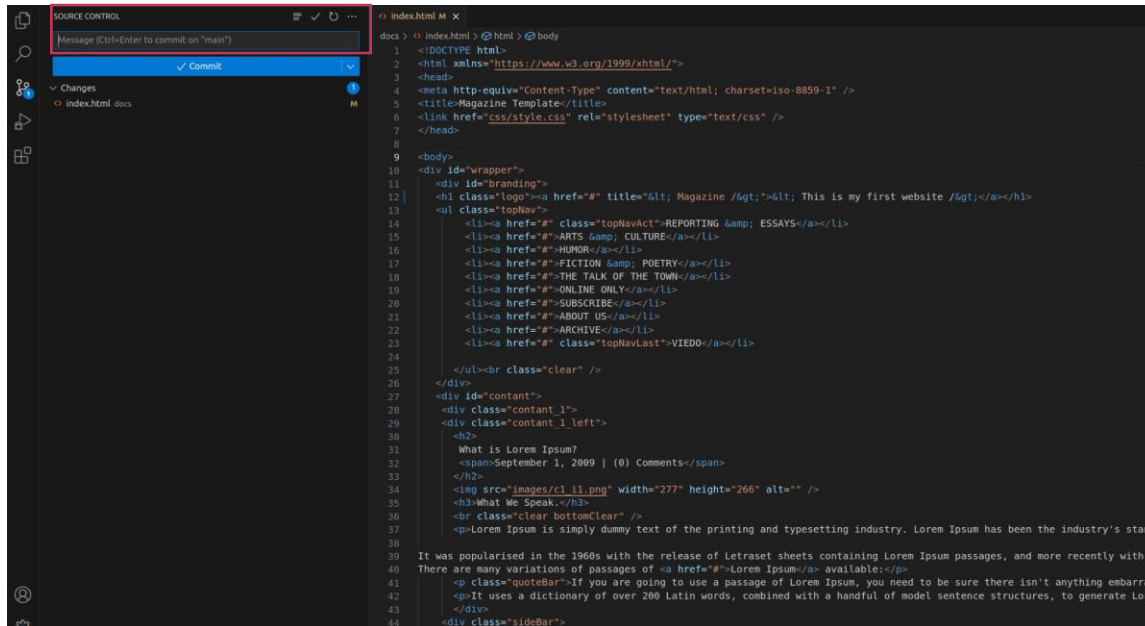


You can easily select the files you want to include in a commit and push them to the repository with just a few clicks.

- Add file changes to the staging area by clicking the + icon next to the file name in the Source Control panel.
- Revert file changes by pressing the circle arrow icon next to the file.

## Visual studio code – Using a repo

IDEs typically provide visual tools for managing source code with Git. Visual Studio Code includes a dedicated **Source Control** section in the left sidebar, allowing you to track changes, stage files, and execute operations like commits and pushes seamlessly.



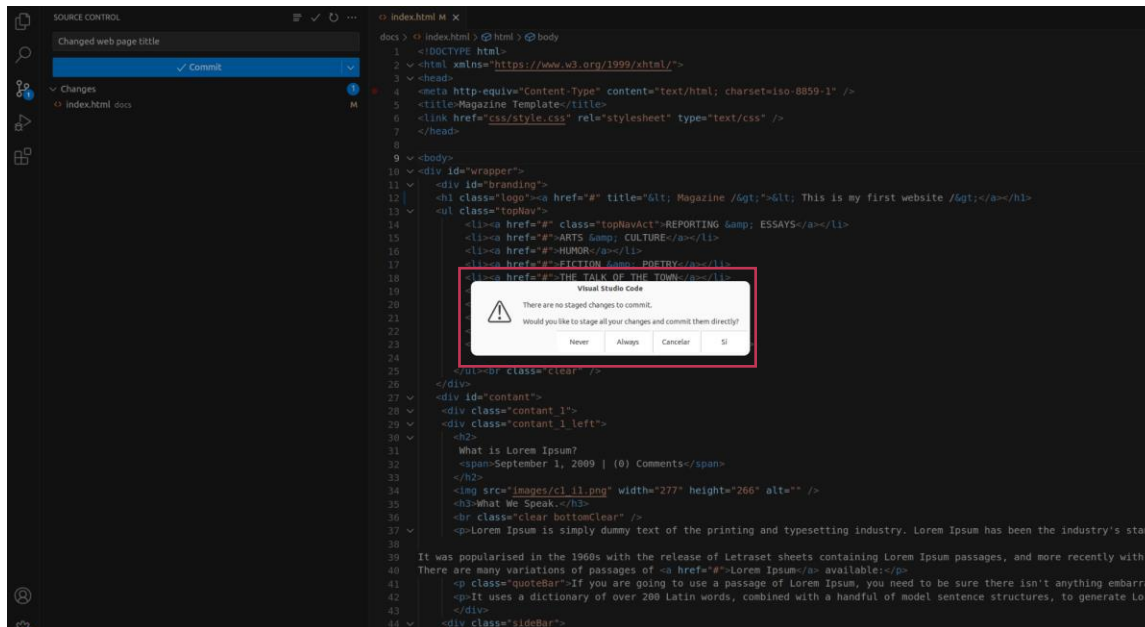
Do not forget to put a comment in your commits

Commit messages are important because they provide a clear and concise record of changes made to the codebase.



## Visual studio code – Using a repo

IDEs typically provide visual tools for managing source code with Git. Visual Studio Code includes a dedicated **Source Control** section in the left sidebar, allowing you to track changes, stage files, and execute operations like commits and pushes seamlessly.

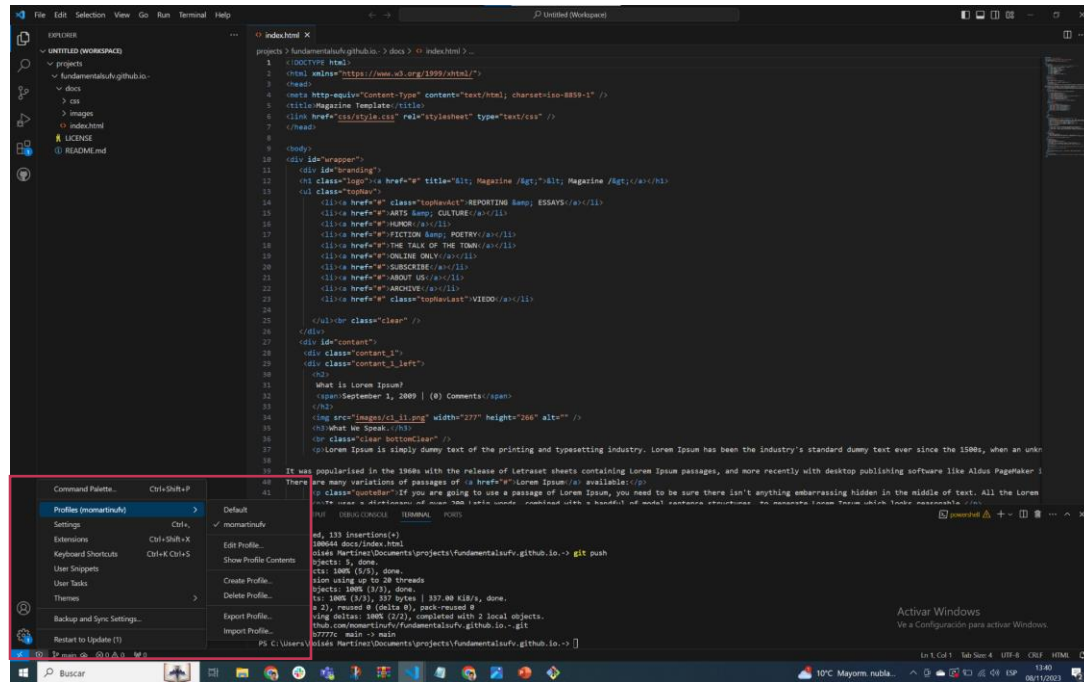


## Commit Assistance

Visual Studio Code provides alerts when no changes have been added to the staging area, reminding you to stage your changes before committing.

## Visual studio code – Configure your account

IDEs often include features for managing Git credentials to streamline authentication with remote repositories.



Visual Studio Code simplifies this process with built-in Git integration, allowing you to configure, manage, and update credentials directly within the interface for seamless interaction with platforms like GitHub or GitLab.

