



# GIT, GITHUB & VS CODE

## CHEATSHEET

JEFF ODHIAMBO

VSTech Limited



# Topics

---

1. Getting started with vscode	3
2. VSCode Tips and Tricks	6
3. Setting Up Git	11
4. Git Basics & Commands	13
5. Using Git in VSCode	19



# 1. Getting started with vscode

## What is Visual Studio Code?

VS Code is a free open-source code editor for the development and debugging of modern cloud and web applications which is available for free on Linux, OS X and Windows.

## Versions

Version	Date Released
1.74.3	
1.74.2	Nov 2022
1.74.1	

Version	Date Released
1.73.1	Oct 2022
172.2	
172.1	Sept 2022
1.71.2	
1.71.1	Aug 2022
1.70.3	
1.70.2	July 2022
1.70.1	

To learn of other versions visit: [https://code.visualstudio.com/updates/?wt.mc\\_id=studentamb\\_219615](https://code.visualstudio.com/updates/?wt.mc_id=studentamb_219615)

# Installation or Setup

## On Windows

- [Download the Visual Studio Code](#) installer for Windows.
- Once it is downloaded, run the installer (VSCodeSetup-version.exe). This will only take a minute.

By default, VS Code is installed under `C:\Program Files (x86)\Microsoft VS Code` for a 64-bit machine.

Tip: Setup will optionally add Visual Studio Code to your %PATH%, so from the console you can type 'code .' to open VS Code on that folder.

You will need to restart your console after the installation for the change to the %PATH% environmental variable to take effect.

## On Mac

- [Download Visual Studio Code](#) for Mac.
- Double-click on the downloaded archive to expand the contents.
- Drag Visual Studio Code.app to the Applications folder, making it available in the Launchpad. •
- Add VS Code to your Dock by right-clicking on the icon and choosing Options, Keep in Dock.

You can also run VS Code from the terminal by typing 'code' after adding it to the path

# Installation or Setup in Linux

---

## Debian and Ubuntu based distributions



The easiest way to install for Debian/Ubuntu based distributions is to download and install the .deb package (64-bit) either through the graphical software center if it's available or through the command line with:

`sudo dpkg -i <file>.deb`

`sudo apt-get install -f # Install dependencies`

[https://code.visualstudio.com/docs/setup/linux/?wt.mc\\_id=studentamb\\_219615](https://code.visualstudio.com/docs/setup/linux/?wt.mc_id=studentamb_219615)

# 2. VSCode Tips and Tricks

## 1. Multiple Cursors

Visual Studio lets you work on different sections of a file at the same time. This can come in handy when you need to perform many identical changes. To create a new cursor, simply click on the respective area in the text editor while holding down the **alt** key. From that point onwards any keyboard input will be applied to all cursor positions. It is also possible to select multiple sections in your code

```
1  <!DOCTYPE html>
2  <html lang="en">
3  >   <head>...
4  </head>
5  <body>
6
7      <ul class="nav justify-content-center">
8          <li class="nav-item">
9              <a class="nav-link active" href="#">Active link</a>
10         </li>
11         <li class="nav-item">
12             <a class="nav-link" href="#">Link</a>
13         </li>
14         <li class="nav-item">
15             <a class="nav-link disabled" href="#">Disabled link</a>
16         </li>
17     </ul>
18
19
20
21
22     <div id="carouselId" class="carousel slide" data-ride="carousel">
23         <ol class="carousel-indicators">
24             <li data-target="#carouselId" data-slide-to="0" class="active"></li>
25             <li data-target="#carouselId" data-slide-to="1"></li>
26             <li data-target="#carouselId" data-slide-to="2"></li>
27         </ol>
28         <div class="carousel-inner" role="listbox">
29             <div class="carousel-item active">
30                 
34         </div>
35     </div>
36
37
38
39
40
41
42
43
```

[https://code.visualstudio.com/docs/getstarted/tips-and-tricks/?wt.mc\\_id=studentamb\\_219615](https://code.visualstudio.com/docs/getstarted/tips-and-tricks/?wt.mc_id=studentamb_219615)

## 2. Emmet Snippets

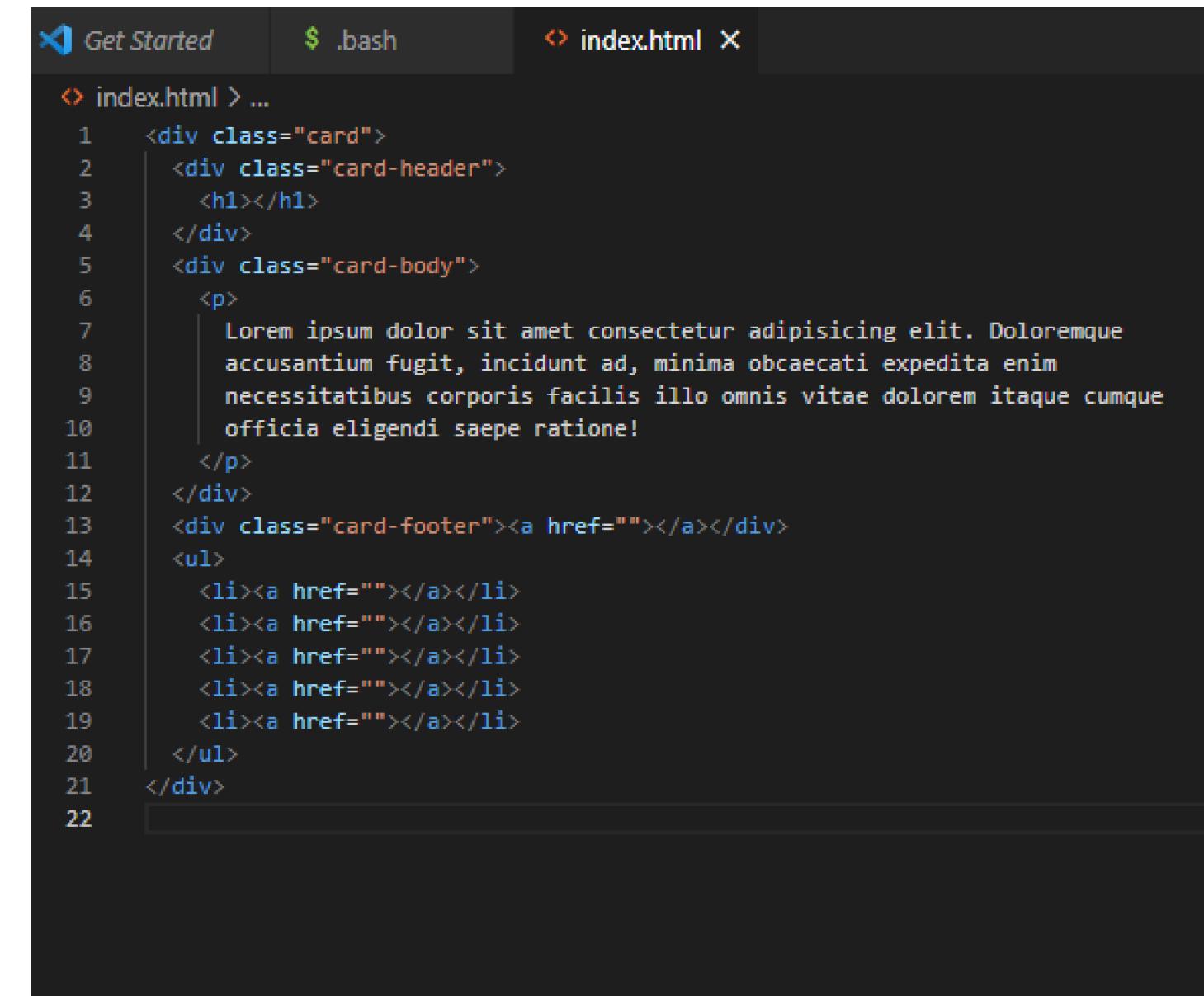
VS Code supports Emmet snippets. Emmet snippets are expressions that are converted to e.g. HTML or XML after pressing the tab key.

Find all sorts of Emmet constructs at  
<http://docs.emmet.io/cheat-sheet/>



The screenshot shows the VS Code interface with three tabs: "Get Started", ".bash", and "index.html". The "index.html" tab is active and displays the Emmet snippet expansion process. The input in the editor is ".card>(.card-header>h1)+(.card-body>p>lorem)+(.card-footer>a))+ul>li\*5>a". The output in the preview pane shows the expanded HTML code:

```
<div class="card">
  <div class="card-header">
    <h1></h1>
  </div>
  <div class="card-body">
    <p>
      Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque accusantium fugit, incidunt ad, minima obcaecati expedita enim necessitatibus corporis facilis illo omnis vitae dolorem itaque cumque officia eligendi saepe ratione!
    </p>
  </div>
  <div class="card-footer"><a href=""></a></div>
<ul>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
</ul>
</div>
```



The screenshot shows the VS Code interface with three tabs: "Get Started", ".bash", and "index.html". The "index.html" tab is active and displays the expanded HTML code. The input in the editor is ".card>(.card-header>h1)+(.card-body>p>lorem)+(.card-footer>a))+ul>li\*5>a". The output in the preview pane shows the expanded HTML code:

```
<div class="card">
  <div class="card-header">
    <h1></h1>
  </div>
  <div class="card-body">
    <p>
      Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloremque accusantium fugit, incidunt ad, minima obcaecati expedita enim necessitatibus corporis facilis illo omnis vitae dolorem itaque cumque officia eligendi saepe ratione!
    </p>
  </div>
  <div class="card-footer"><a href=""></a></div>
<ul>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
  <li><a href=""></a></li>
</ul>
</div>
```

### 3. Keyboard Shortcut

VSCode supports various keyboard shortcuts, which allow developers to quickly and efficiently perform common actions and tasks within the code editor. These shortcuts can be used to navigate the interface, write and edit code, debug, and more. Some examples of VSCode keyboard shortcuts include:

- Ctrl + P (or Command + P on Mac) to quickly open a file by name
- Ctrl + Shift + P (or Command + Shift + P on Mac) to open the Command Palette, which allows developers to access various commands and features
- Ctrl + Shift + Tab to switch between recently used files
- Ctrl + Shift + N to open a new window
- Ctrl + D to select the next occurrence of a word
- Ctrl + Shift + L to select all occurrences of a word
- Ctrl + F (or Command + F on Mac) to find text within a file
- Ctrl + H (or Command + Option + F on Mac) to find and replace text within a file
- Ctrl + Shift + I (or Control + Shift + I on Mac) to open the integrated developer console
- Ctrl + / (or Command + / on Mac) to comment or uncomment selected lines of code

## 4. Markdown Preview

Visual Studio offers a markdown preview function that can be opened beneath the text. VS Code is using the GitHub Flavoured Markdown which offers additional functions over the original markdown format. This comes in handy especially for GitHub “read me” documents

Using **Ctrl+Shift+V**, you can switch between Markdown and Preview. **Alt+Z** toggles word wrapping, which can also be done using the View menu.

The screenshot shows the Visual Studio Code interface with two panes. The left pane displays the file 'test.md' containing GitHub Flavored Markdown. The right pane, titled 'Preview test.md', shows the rendered HTML output of the same content. The preview includes syntax highlighting and correctly handles code blocks and links. A status bar at the bottom indicates the preview is active.

```
public static void main(String[] args) {
    System.out.println(Math.sqrt(144));
}

Output: 12

Example using static import
```
import static java.lang.Math.*;
public class Test{
    public static void main(String[] args) {
        System.out.println(sqrt(144));
    }
}

Handling name conflicts
The only time we need to pay attention to packages is when we have a name conflict . For example both, java.util Date . So if we import both packages in program as follows:
```
import java.util.*;
import java.sql.*;
```
And then use Date class, then we will get a compile-time error :
`Date today ; //ERROR-- java.util.Date or java.sql.Date`
The compiler will not be able to figure out which Date class do we want. This problem can be solved by using a specific import statement:
```
import java.util.Date;
import java.sql.*;
```
If we need both Date classes then, we need to use a full package name every time we declare a new object of that class. For Example:
```
java.util.Date deadline = new java.util.Date();
java.sql.Date today = new java.sql.Date();
```
How to compile the Package (if not using IDE)
If you are not using any IDE, you need to follow the syntax given below:
`javac -d <directory or dot operator> <javafilename>`
For example
`javac -d . Simple.java`
The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home etc. For more visit [Java Installation](https://dev.to/killallnano/java-for-beginners-installation-of-java-nhc).
**How to run the Package (if not using IDE)**
You need to use fully qualified name e.g. mypack.Simple etc to run the class.
**To Compile:** `javac - d . Simple.java`
The '-d' is a switch that tells the compiler where to put the class file i.e. it represents destination. The '.' This marks the end of the Java for beginners.
You can learn more about Java through the urls below:
[https://beginnersbook.com/2013/03/packages-in-java/](https://beginnersbook.com/2013/03/packages-in-java/)  

[https://www.geeksforgeeks.org/packages-in-java/](https://www.geeksforgeeks.org/packages-in-java/)

public static void main(String[] args) {
    System.out.println(Math.sqrt(144));
}

Output: 12 Example using static import

import static java.lang.Math.*;
public class Test{
    public static void main(String[] args) {
        System.out.println(sqrt(144));
    }
}

Handling name conflicts The only time we need to pay attention to packages is when we have a name conflict . For example both, java.util and java.sql packages have a class named Date . So if we import both packages in program as follows:

import java.util.*;
import java.sql.*;

And then use Date class, then we will get a compile-time error : Date today ; //ERROR-- java.util.Date or java.sql.Date? The compiler will not be able to figure out which Date class do we want. This problem can be solved by using a specific import statement:

import java.util.Date;
import java.sql.*;

If we need both Date classes then, we need to use a full package name every time we declare a new object of that class. For Example:

java.util.Date deadline = new java.util.Date();
java.sql.Date today = new java.sql.Date();

How to compile the Package (if not using IDE) If you are not using any IDE, you need to follow the syntax given below: javac -d <directory or dot operator> <javafilename> For example javac -d . Simple.java The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot). For more visit Java Installation. How to run the Package (if not using IDE) You need to use fully qualified name e.g. mypack.Simple etc to run the class. To Compile: javac - d . Simple.java The -d is a switch that tells the compiler where to put the class file i.e. it represents destination. The . represents the current folder.

This marks the end of the Java for beginners. You can learn more about Java through the urls below:
https://beginnersbook.com/2013/03/packages-in-java/ https://www.geeksforgeeks.org/packages-in-java/
```

## 5. Visualizing CSS Selectors

CSS selectors define what HTML elements to style with the subsequent properties. Selectors can become pretty complicated – and be it because sometimes you just might not be too familiar with one of them that you are using less frequently.

Visual Studio Code solves this problem by presenting a preview for the current selector that shows what HTML elements would have to be like to be covered by the selector.

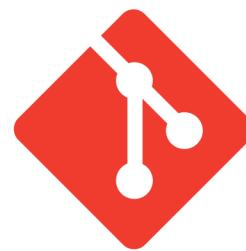
```
1 <section class="article">
2   <p>
3
4 section.article > p {
5   color: green;
6 }
```

```
1 <h1>
2 <p>
3   <color important="true">
4     ...
5       <element>
6 h1 + p > color[important=true] * {
7   color: red;
8 }
```

```
1 <p id="MainText">
2
3 p#MainText {
4   color: green;
5 }
```

```
1 <section class="footer">
2 ...
3   <a :hover>
4
5 section.footer a:hover {
6   text-decoration: underline;
7 }
```



# 3. Git

## Learning Objectives

1. Learn what version control is
2. Understand distributed version control systems, like Git
3. Create a new Git project and configure it
4. Make and track changes to code by using Git
5. Use Git to recover from simple mistakes



## What is version control?

A version control system (VCS) is a program or set of programs that track changes to a collection of files. One goal of a VCS is to easily recall earlier versions of individual files or of the entire project. Another goal is to allow several team members to work on a project, even on the same files, at the same time without affecting each other's work.

Another name for a VCS is a software configuration management (SCM) system.

## Features of VCS

- VCS allows you to track and manage changes made to a project
- You can view changes made to your project, when they were made, and who made them
- Includes the ability to include a message with each change to explain the reasoning behind it
- Retrieve past versions of the entire project or individual files
- Create branches to experimentally make changes without affecting the main branch
- Merge desired changes back into the main branch
- Ability to attach tags to versions, making it easy to keep track of different versions of software
- Facilitate collaboration among team members.

To learn of other versions visit: <https://git-scm.com/downloads>

## What is Git?

Git is a distributed version control system that allows developers to track and manage changes in their code

# 4. Git Basics

---

## Git terminology

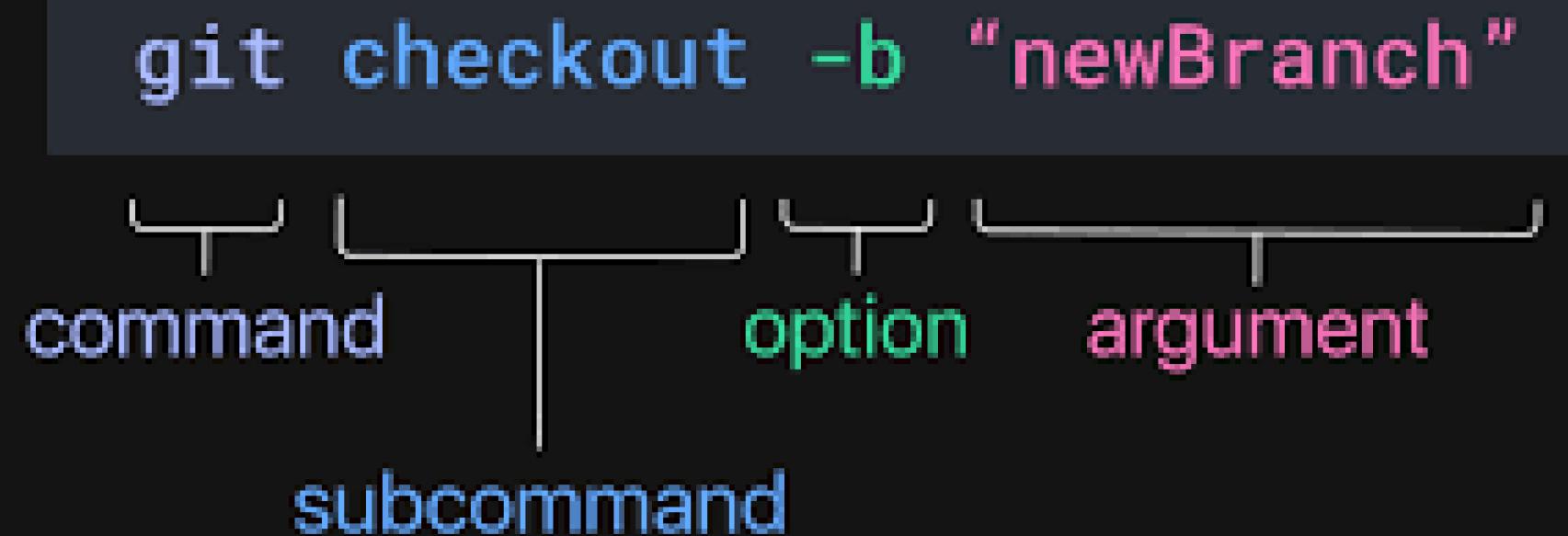
To understand Git, you have to understand the terminology. Here's a short list of terms that Git users frequently use.

| Terminology       | Definition                                                                                                                                                                                                                                                                     |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Working tree      | The set of nested directories and files that contain the project that's being worked on.                                                                                                                                                                                       |
| Repository (repo) | The directory, located at the top level of a working tree, where Git keeps all the history and metadata for a project.                                                                                                                                                         |
| Hash              | A number produced by a hash function that represents the contents of a file or another object as a fixed number of digits.                                                                                                                                                     |
| Object            | A Git repo contains four types of objects. A blob object contains an ordinary file. A tree object represents a directory; it contains names, hashes, and permissions. A commit object represents a specific version of the working tree. A tag is a name attached to a commit. |
| Commit            | Commit is an operation which sends the latest changes of the source code to the repository                                                                                                                                                                                     |

## Terminology    Definition

---

|                                    |                                                                                                                                                                                                                                                                                                                                 |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Branch                             | A branch is a named series of linked commits. The most recent commit on a branch is called the head. The default branch, which is created when you initialize a repository, is called main, often named master in Git.                                                                                                          |
| Remote                             | A remote is a named reference to another Git repository.                                                                                                                                                                                                                                                                        |
| Commands, subcommands, and options | Git operations are performed by using commands like git push and git pull. git is the command, and push or pull is the subcommand. The subcommand specifies the operation you want Git to perform. Commands frequently are accompanied by options, which use hyphens (-) or double hyphens (--). For example, git reset --hard. |



# The Git command line

The Git command line is a tool that allows users to interact with a Git repository using a command-line interface.

## Sample Git Commands

| Command                                     | Operation                                                                                    |
|---------------------------------------------|----------------------------------------------------------------------------------------------|
| git --version                               | Checks the version of Git currently installed on your system.                                |
| git config --global user.name <USER_NAME>   | Sets user's name                                                                             |
| git config --global user.email <USER_EMAIL> | Sets user's email                                                                            |
| git config –list                            | Lists global configurations such as name and email                                           |
| git init/git init –initial-branch='main'    | Initializes a new Git repository in the current directory.                                   |
| git clone <repository>                      | Create a copy of an existing Git repository on your local machine.                           |
| git add                                     | Add a file to the staging area/tells Git to start keeping track of changes in certain files. |
| git commit -m <message>                     | Saves changes to the repository with a brief message describing the changes.                 |
| git status                                  | Checks the status of the working tree.                                                       |

| Command                                   | Operation                                                                                              |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------|
| git diff                                  | Views the differences between the working directory and the last committed version of the code.        |
| git log                                   | Views the commit history of the repository                                                             |
| git branch                                | Views the branches in the repository                                                                   |
| git checkout -b <branch>                  | Used to switch to a different branch                                                                   |
| git init/git init --initial-branch='main' | Initializes a new Git repository in the current directory.                                             |
| git merge <branch>                        | Used to merge changes from one branch into another                                                     |
| git pull                                  | Used to download the latest changes from a remote repository and merge them into the local repository. |
| git push                                  | Used to upload local changes to a remote repository                                                    |
| git fork                                  | Used to create a personal copy of a repository on GitHub.                                              |
| git rm <file>                             | Used to remove a file from the repository and delete it from the file system.                          |
| git revert                                | Used to undo the previous commit.                                                                      |
| git reset                                 | Used to undo commits by discarding commits from the current branch.                                    |
| git clean                                 | Used to remove untracked files from the working tree.                                                  |
| git stash                                 | This command temporarily saves changes that are not committed yet                                      |



# Git and GitHub

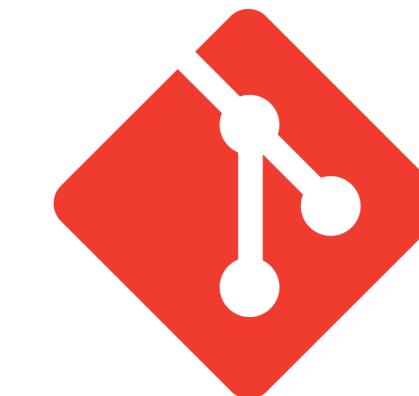
**Git** is a distributed version control system (DVCS) that multiple developers and other contributors can use to work on a project. It provides a way to work with one or more local branches and then push them to a remote repository.



**GitHub** is a cloud platform that uses Git as its core technology. GitHub simplifies the process of collaborating on projects and provides a website, more command-line tools, and overall flow that developers and users can use to work together. GitHub acts as the remote repository mentioned earlier.

**Key features provided by GitHub include:**

- Issues.
- Discussions.
- Pull requests.
- Notifications.
- Labels.
- Actions.
- Forks.
- Projects



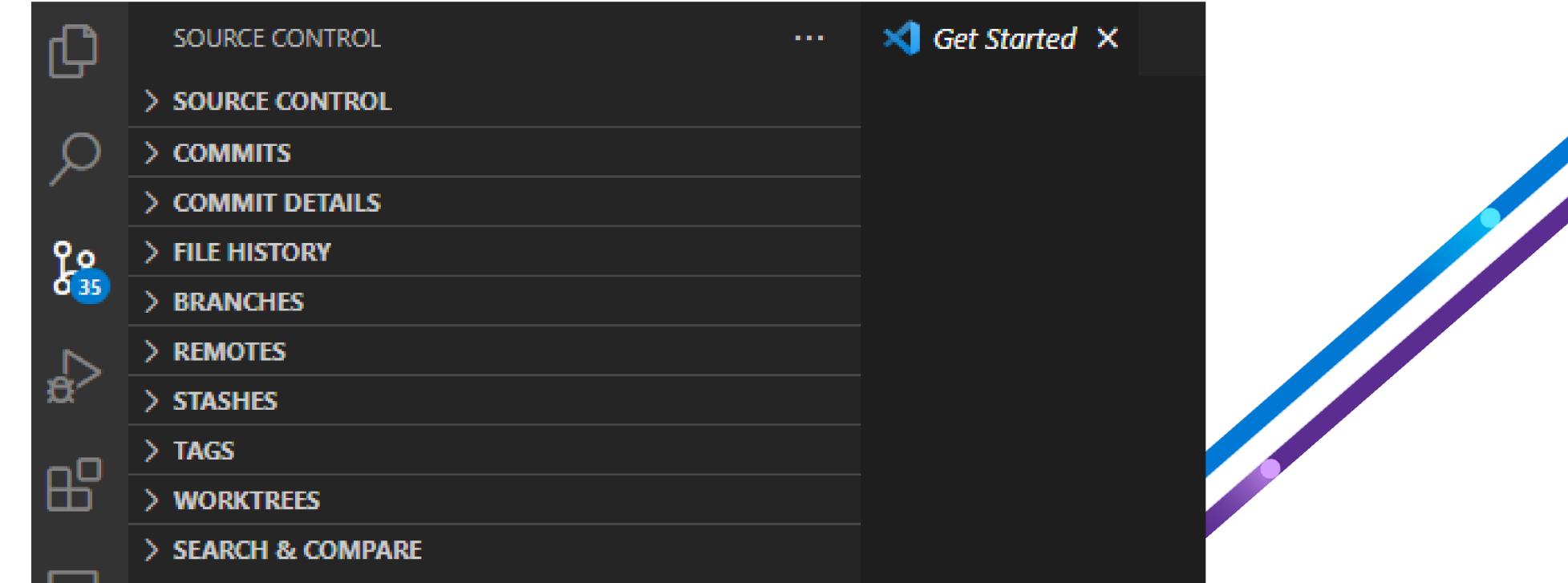
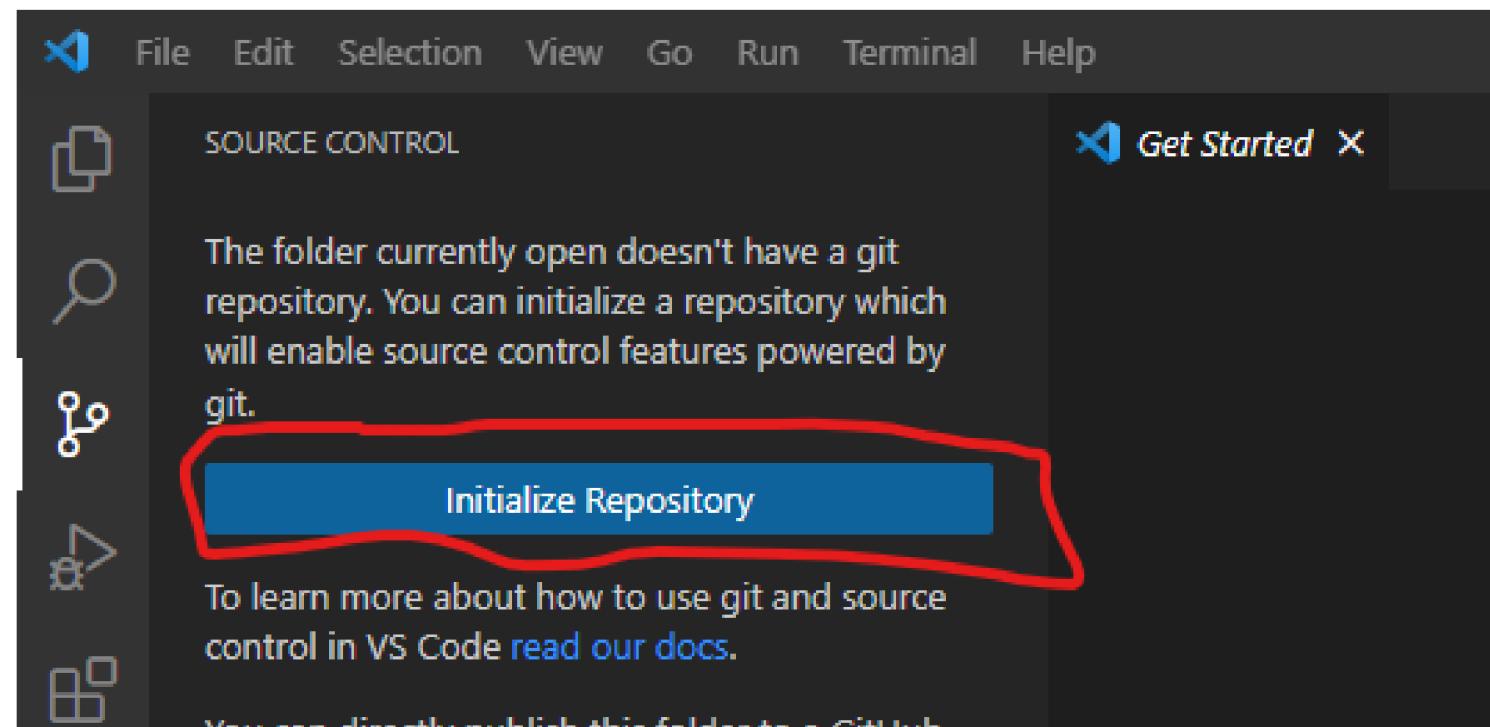
To learn more about GitHub, see the [Introduction to GitHub](#) Microsoft Learn module or the [Getting started with GitHub](#) help documentation.

# 5. Using Git in VSCode

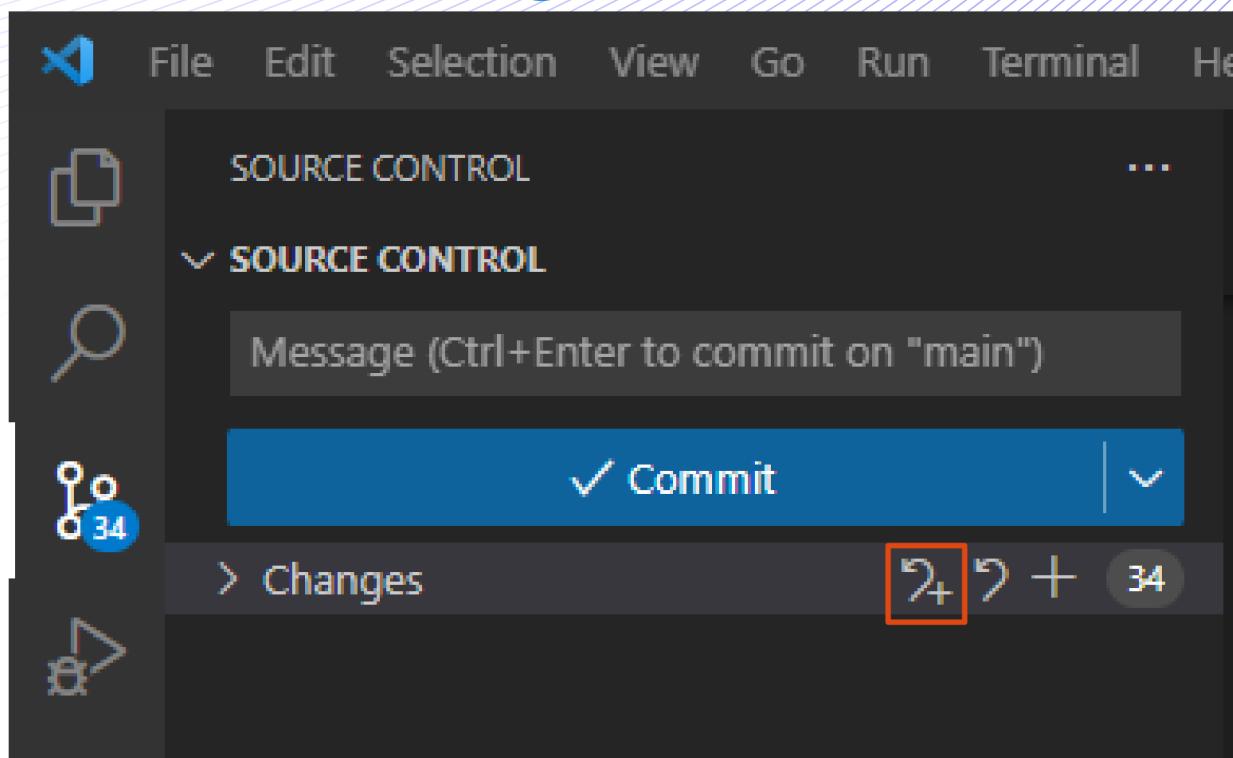
In this guide, we will cover how to use the Source Control inbuilt feature for Visual Studio Code (VSCode), including how to use Git commands, view changes in your code, and undo changes via GUI.

By the end of this guide, you will have a solid understanding of how to use Git within VSCode and be able to confidently manage your code revisions.

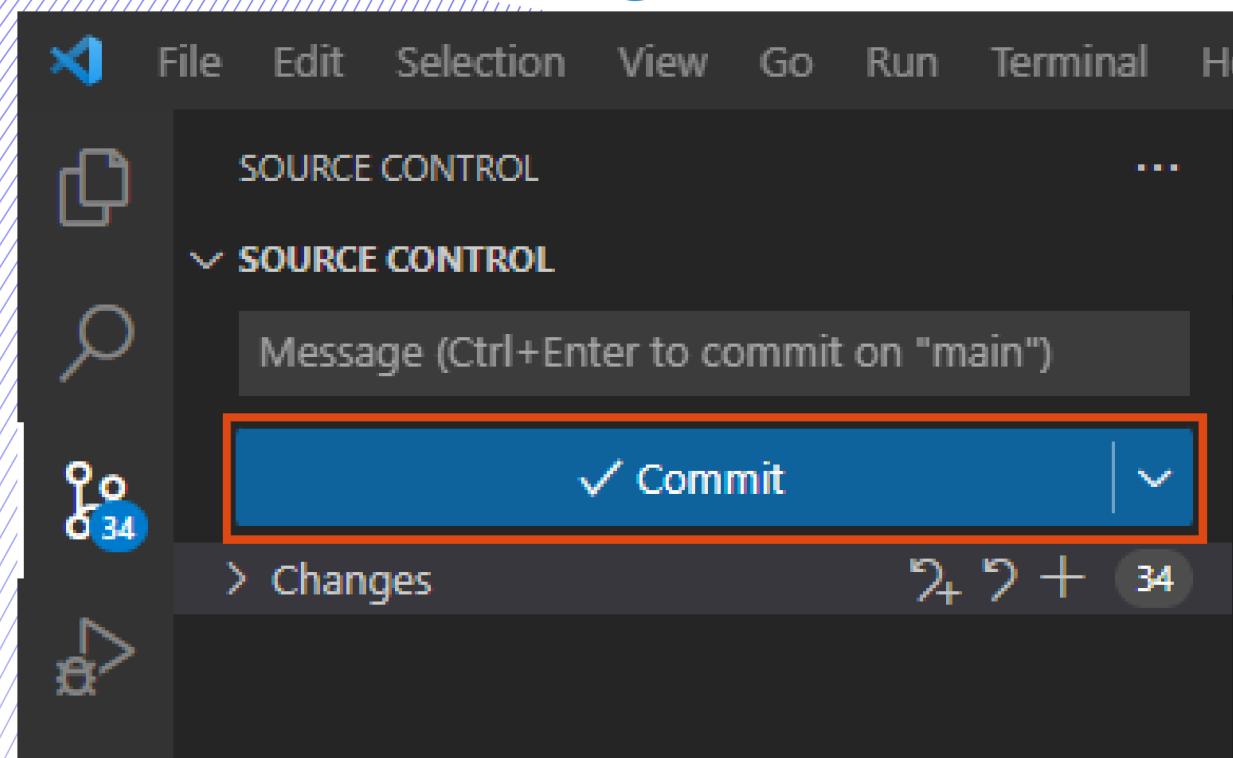
## 1. Initialize Repository



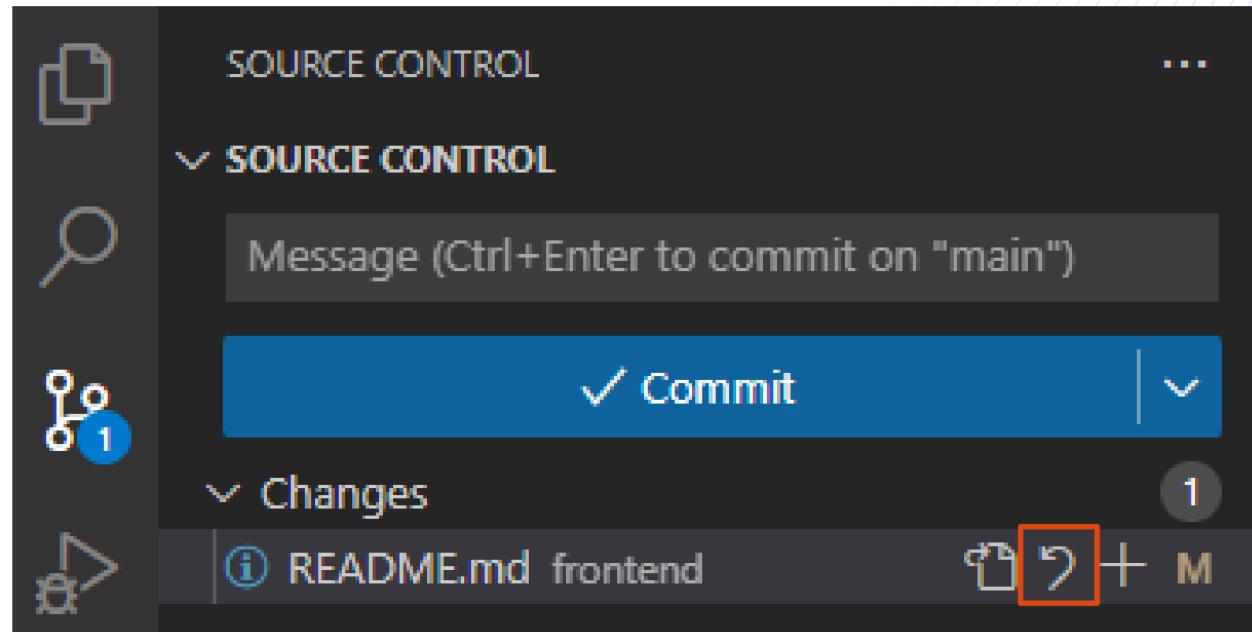
## 2. Stash Changes



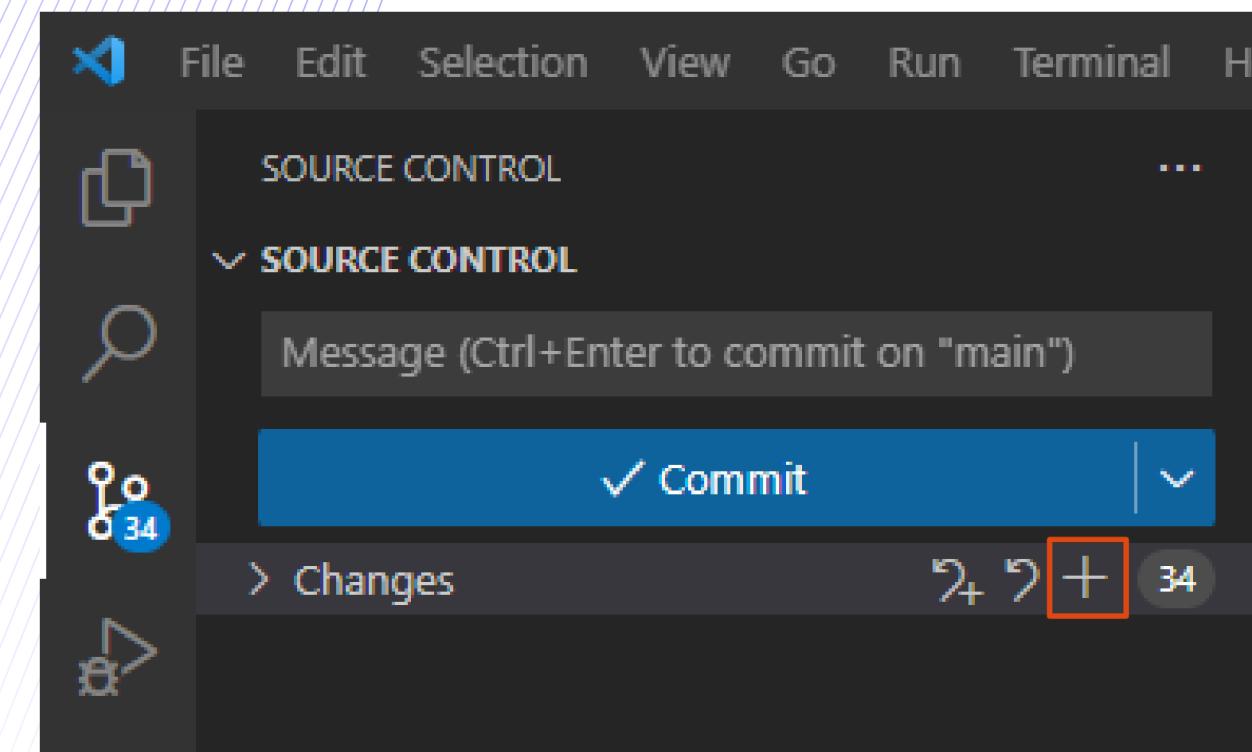
## 3. Commit Changes



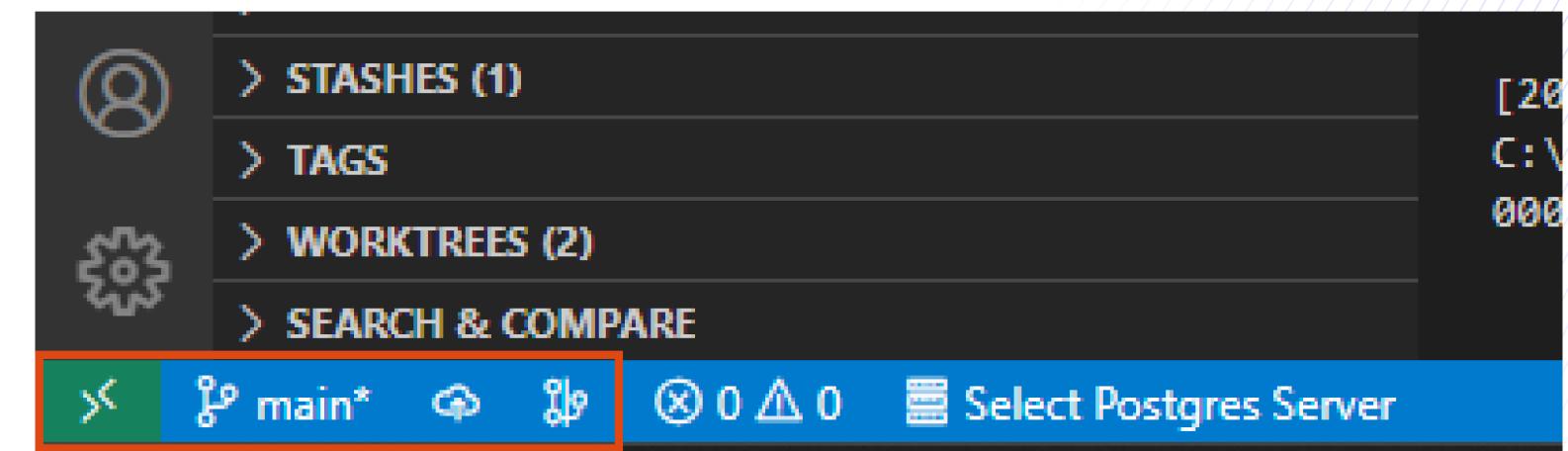
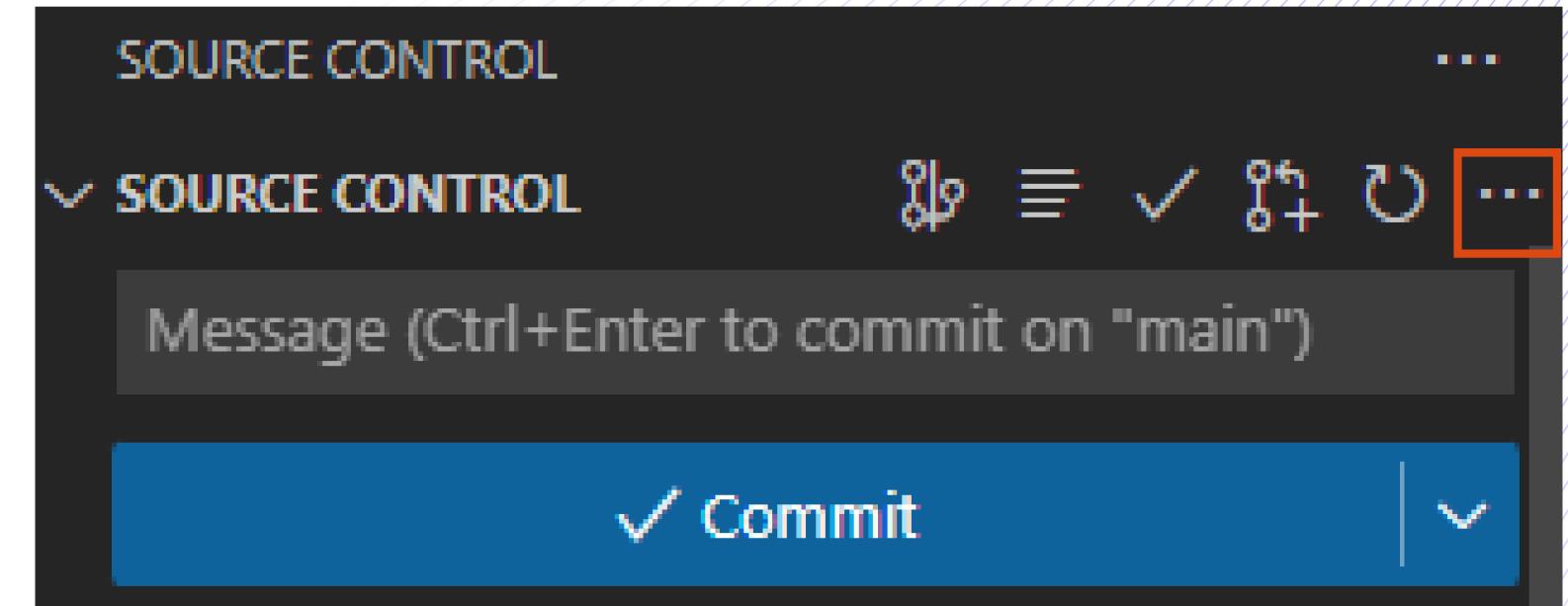
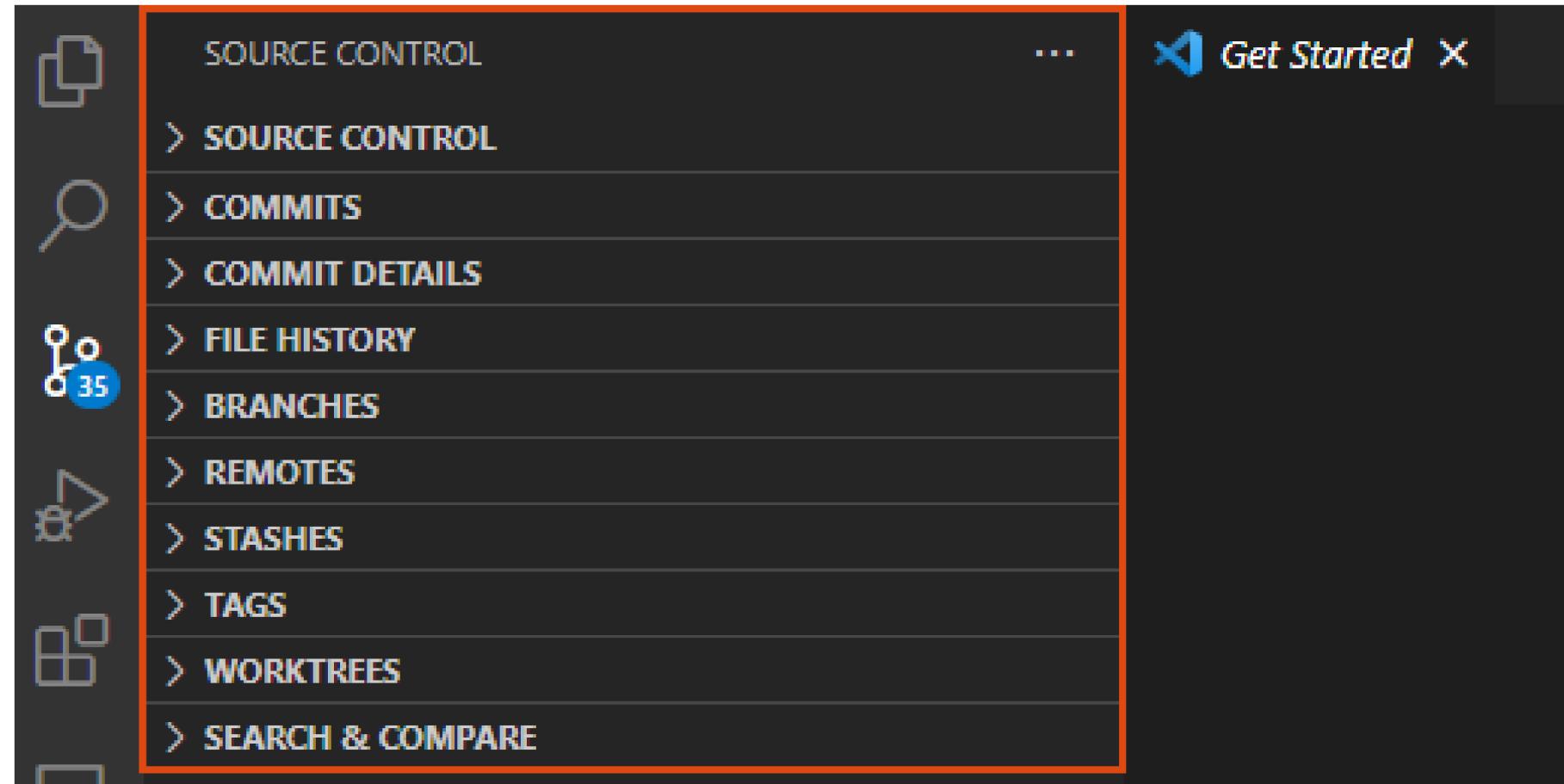
## 4. Discard Changes



## 5. Add Files

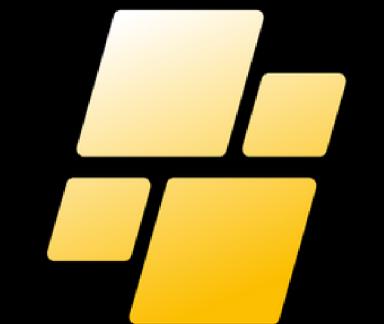


# Other Git Shortcuts





[ END ]



VSTech  
Limited Company