



Git



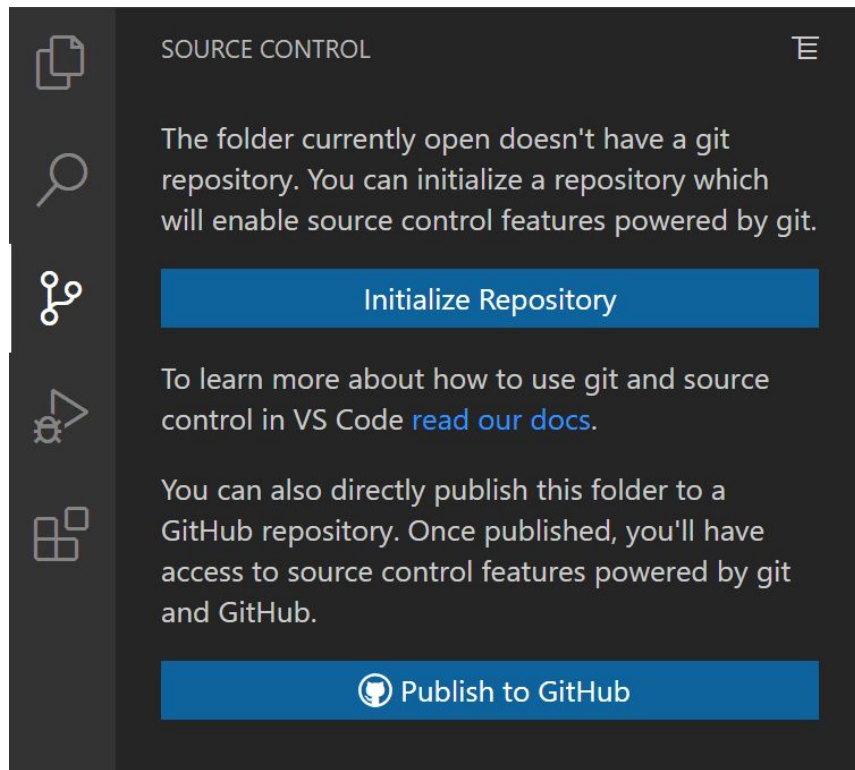
VS Code

Visual Studio Code has integrated source control management (SCM) and includes Git support in-the-box.



VS Code

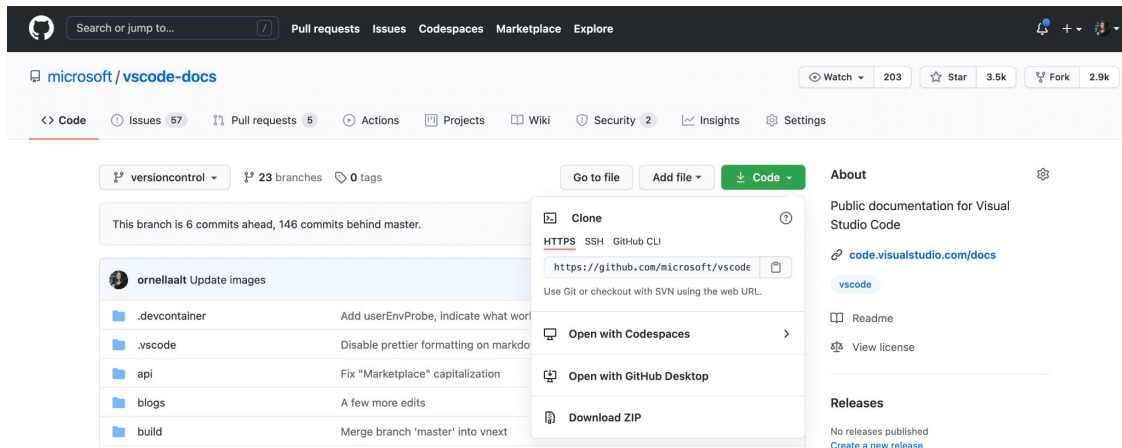
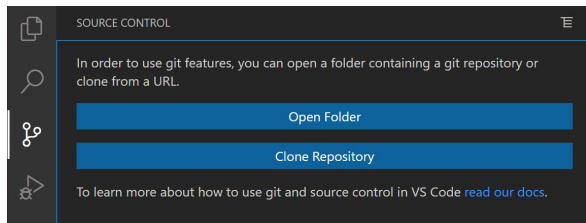
Initialize a repository



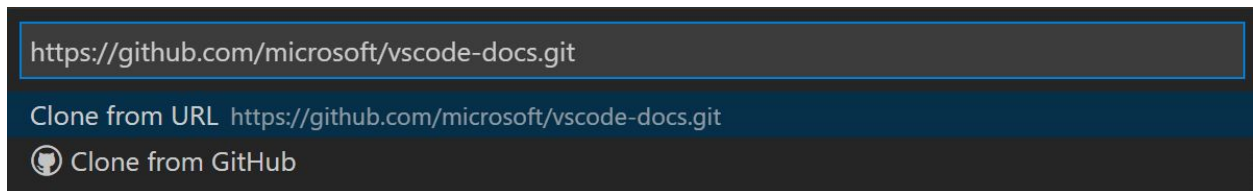
If your workspace is on your local machine, you can enable Git source control by creating a Git repository with the Initialize Repository command.

VS Code

Cloning a Repository

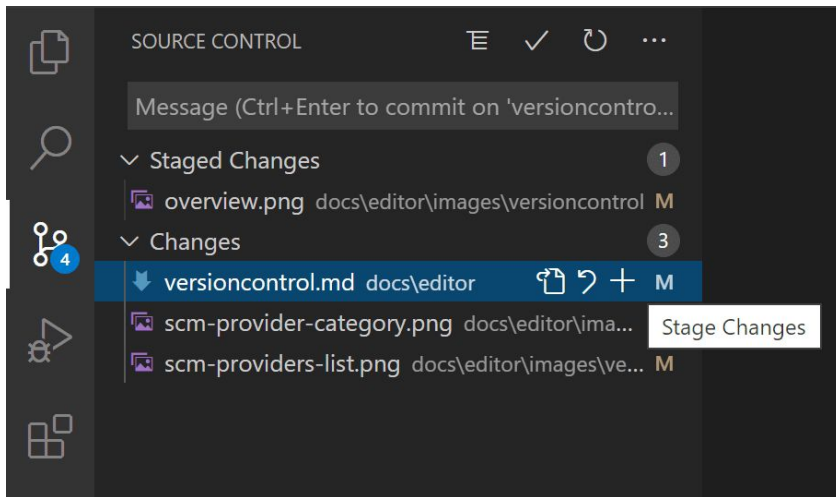


Ctrl+Shift+P



VS Code

Commit



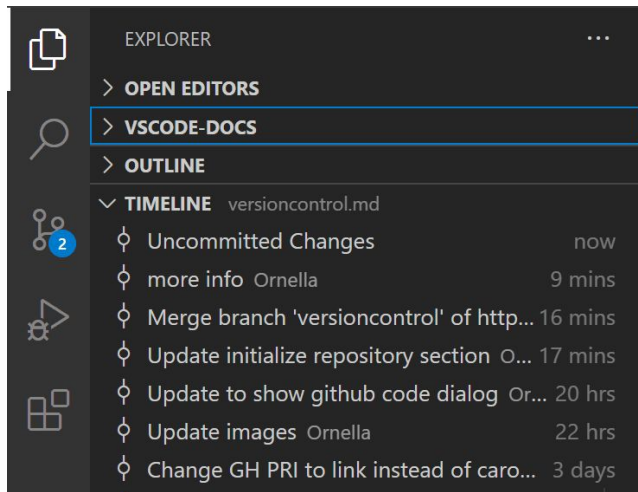
type a commit message above the changes and press Ctrl+Enter (macOS: ⌘+Enter) to commit

Tip: If you commit your change to the wrong branch, undo your commit using the Git: Undo Last Commit command in the Command Palette (Ctrl+Shift+P).

VS Code

Timeline View

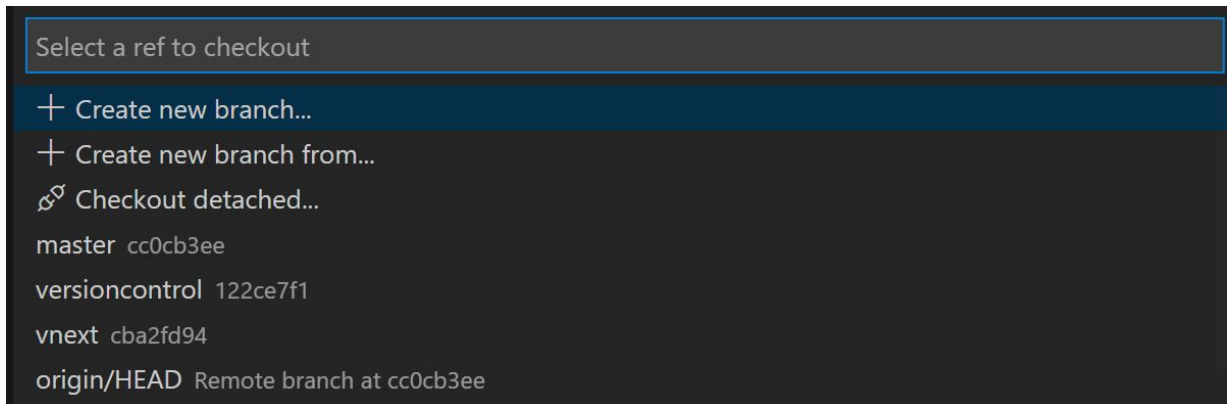
The Timeline view, accessible at the bottom of the File Explorer by default, is a unified view for visualizing time-series events (for example, Git commits) for a file.



Selecting a commit will open a diff view of the changes introduced by that commit. When you right-click on a commit, you'll get options to Copy Commit ID and Copy Commit Message.

VS Code

Branch & Tags



You can create and checkout branches directly within VS code through the Git: Create Branch and Git: Checkout to commands in the Command Palette (`Ctrl+Shift+P`) .

VS Code

Git Status Bar actions

Synchronize Changes will pull remote changes down to your local repository and then push local commits to the upstream branch.

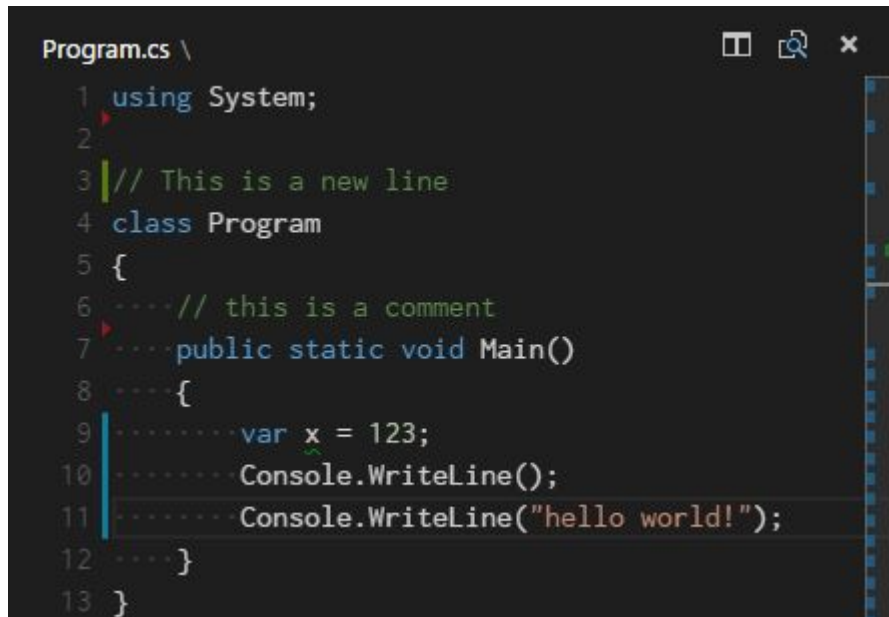


If there is no upstream branch configured and the Git repository has remotes set up, the Publish action is enabled. This will let you publish the current branch to a remote.



VS Code

Gutter Indicators

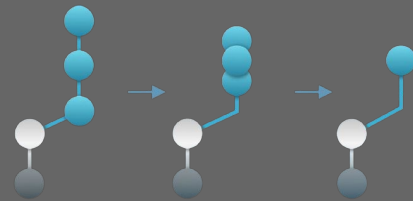


```
Program.cs \
1 using System;
2
3 // This is a new line
4 class Program
5 {
6 // this is a comment
7 public static void Main()
8 {
9 var x = 123;
10 Console.WriteLine();
11 Console.WriteLine("hello world!");
12 }
13 }
```

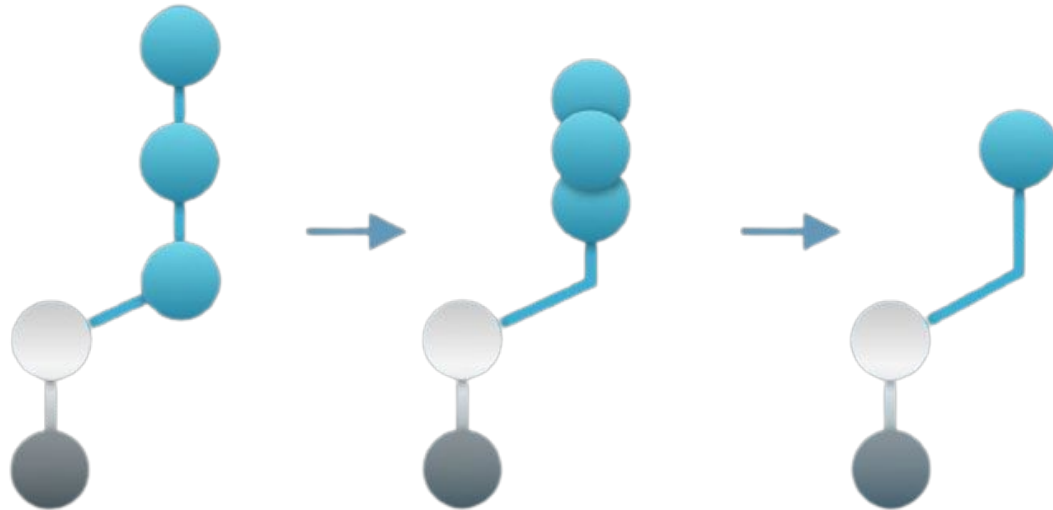
If you open a folder that is a Git repository and begin making changes, VS Code will add useful annotations to the gutter and to the overview ruler.

- A red triangle indicates where lines have been deleted
- A green bar indicates new added lines
- A blue bar indicates modified lines

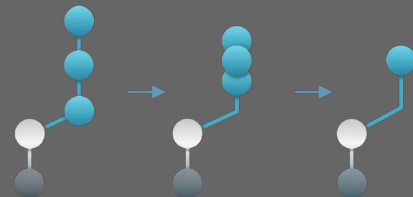
Git Squash



Git Squash is used for merging commits.



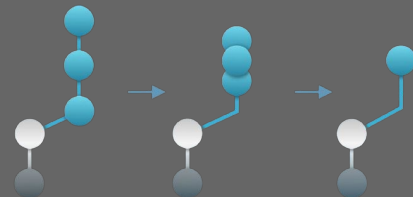
Git Squash





`git rebase -i HEAD~<n>`

- p, pick = use commit
- r, reword = use commit, but edit the commit message
- e, edit = use commit, but stop for amending
- s, squash = use commit, but meld into previous commit
- f, fixup = like "squash", but discard this commit's log message
- x, exec = run command (the rest of the line) using shell
- d, drop = remove commit

Git Squash









This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

command line instructions.







Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

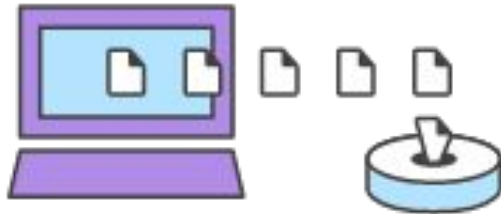
Squash and merge
The 2 commits from this branch will be combined into one commit in the base branch.

Rebase and merge
The 2 commits from this branch will be rebased and added to the base branch.



Git Stash

Git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.



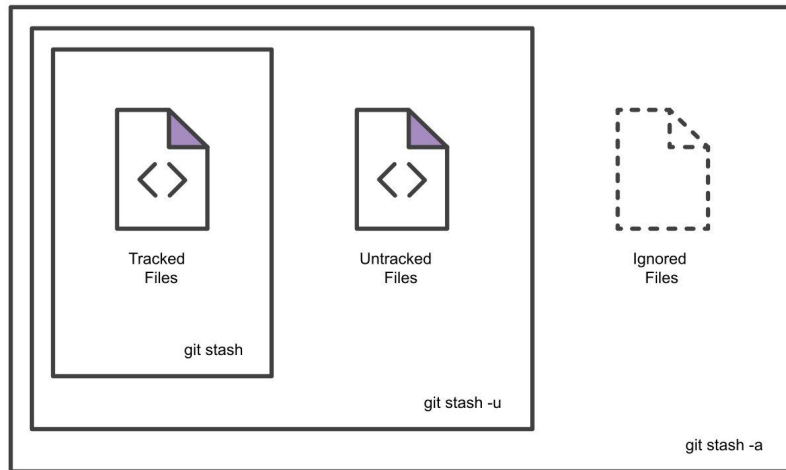
Git Stash

Stashing your Work

```
git stash #stashes your work
```

```
git stash save "user provided text" # stashes you work
```

git stash options



Git Stash

Listing Stashes

```
git stash list
```

```
#list your stashes
```

```
$ git stash list  
stash@{0}: WIP on main: 5002d47 our new homepage  
stash@{1}: WIP on main: 5002d47 our new homepage  
stash@{2}: WIP on main: 5002d47 our new homepage
```

Git Stash

Cleaning Up Stash

`git stash drop` `#drop stashes`

```
$ git stash drop stash@{1}  
Dropped stash@{1} (17e2697fd8251df6163117cb3d58c1f62a)
```

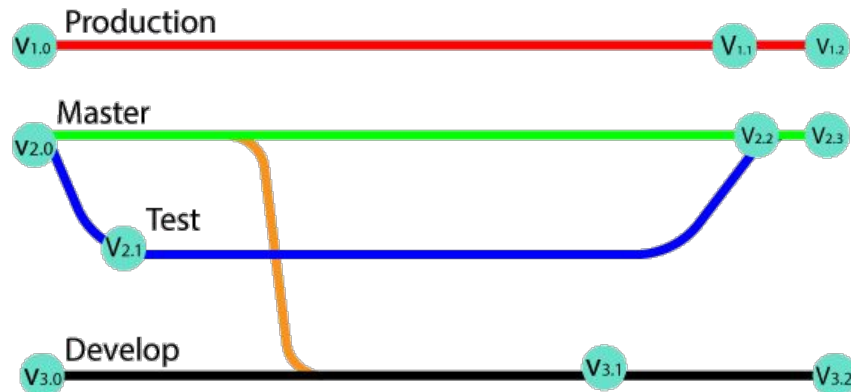
To delete all your stashes

```
$ git stash clear
```


Git Tags

Git tags is used for marking a commit.

- Tags are refs that point to specific points in Git history.
- Tagging is generally used to capture a point in history that is used for a marked version release (i.e. v1.0.1).



Git Tags

Creating a Tag

```
git tag <tagname>
```

Annotated Tag

```
git tag -a v1.4 -m "my version 1.4"
```

Lightweight Tag

```
git tag v1.4-lw
```

Git Tags

Listing Tags

```
git tag
```

```
v0.10.0  
  v0.10.0-rc1  
  v0.11.0  
  v0.11.0-rc1  
  v0.11.1  
  v0.11.2  
  v0.12.0  
  v0.12.0-rc1  
  v0.12.1  
  v0.12.2  
  v0.13.0  
  v0.13.0-rc1  
  v0.13.0-rc2
```

Key Takepoints

- ❑ Git GUI
 - ❑ Initialising Repository
 - ❑ Push Pull Workflow
 - ❑ Branches
- ❑ Squashing
- ❑ Stashing
- ❑ Git Tags

