

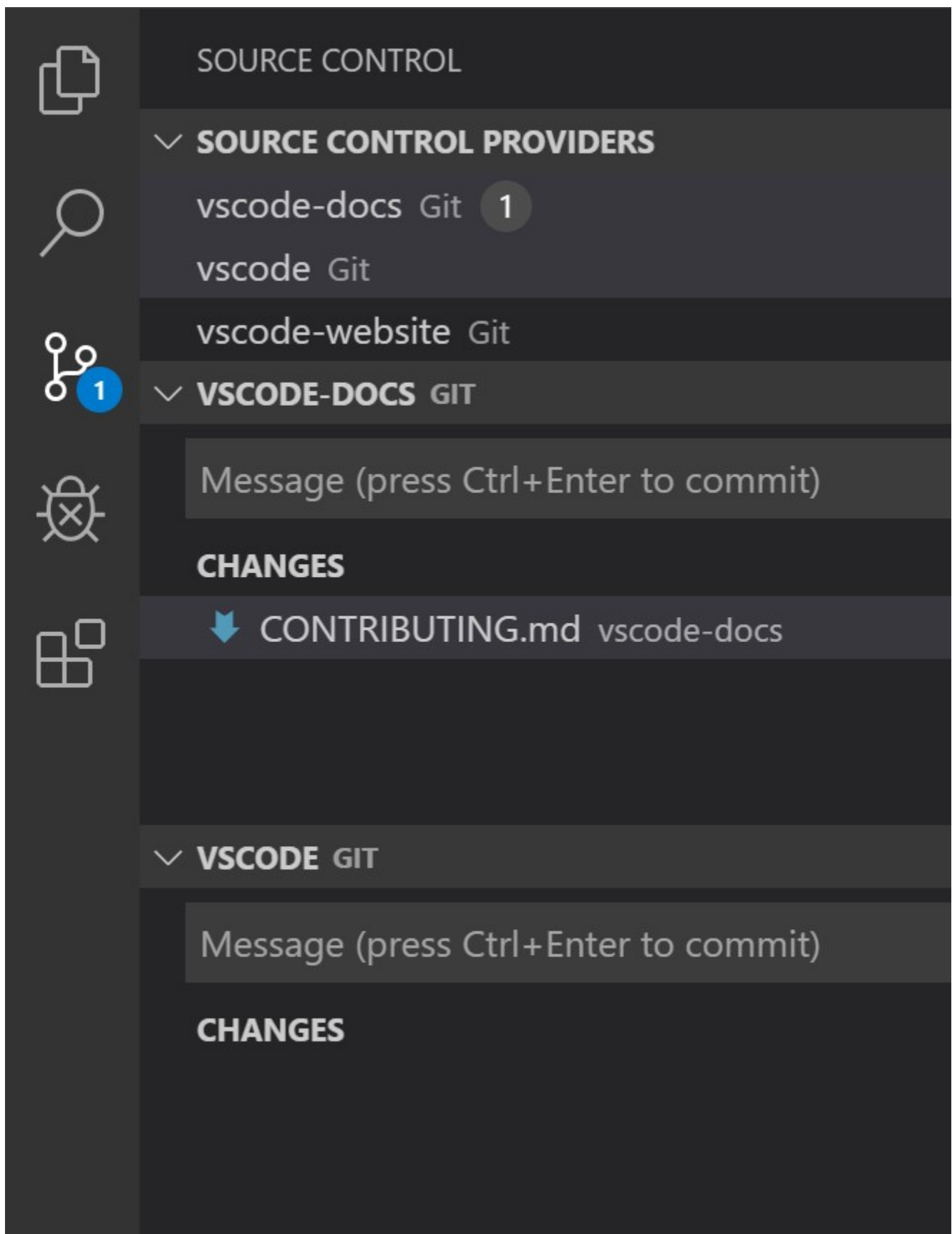
# Version Control in Visual Studio Code

Visual Studio Code has integrated source control and includes [Git](#) support in-the-box. Many other source control providers are available through [extensions](#) on the VS Code Marketplace.

**Tip:** Click on an extension tile to read the description and reviews in the Marketplace.

## SCM Providers

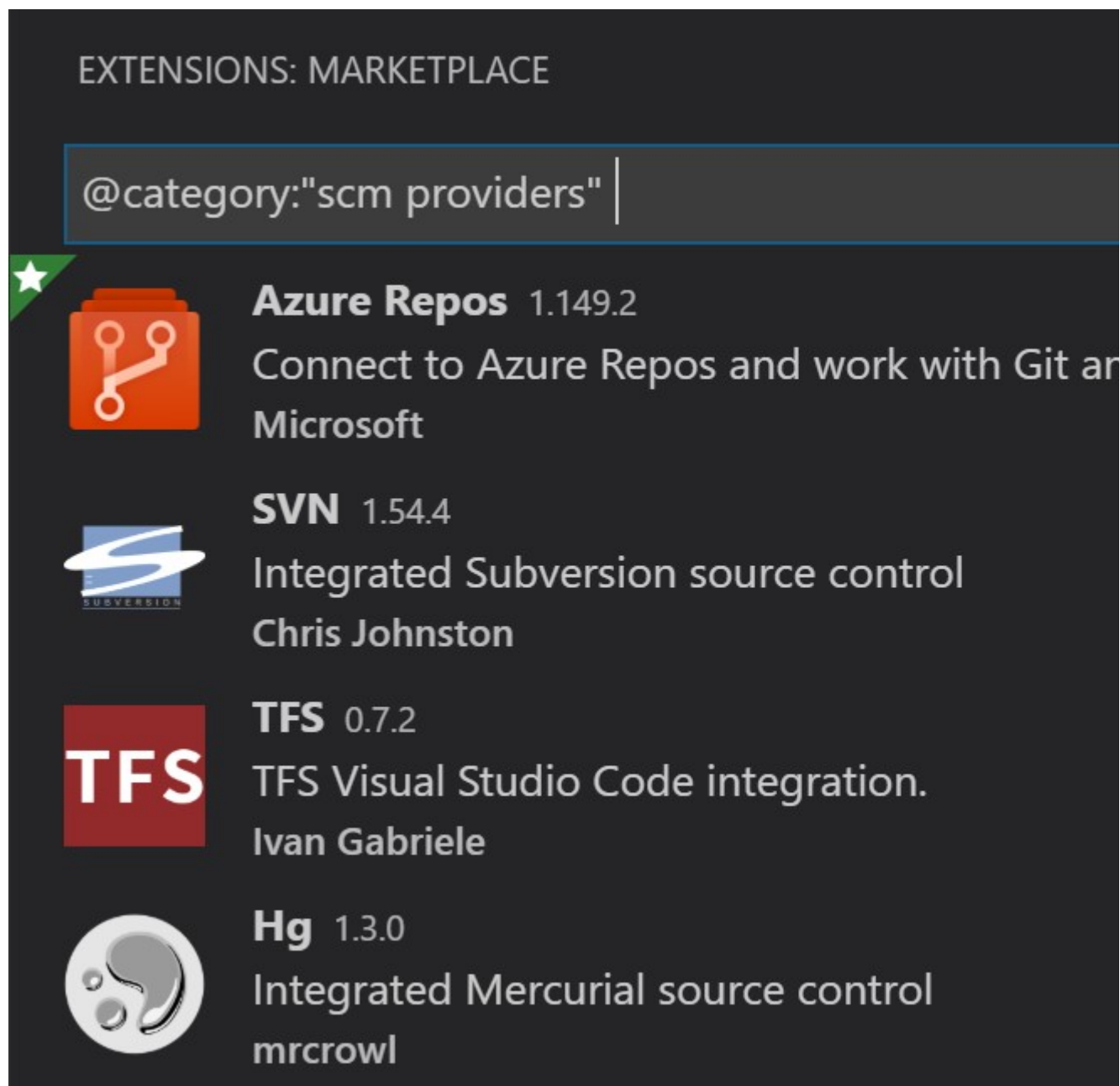
VS Code has support for handling multiple Source Control providers simultaneously. For example, you can open multiple Git repositories alongside your TFS local workspace and seamlessly work across your projects. The **SOURCE CONTROL PROVIDERS** list of the **Source Control** view (Ctrl+Shift+G) shows the detected providers and repositories and you can scope the display of your changes by selecting a specific provider.



## SCM Provider extensions

If you would like to install an additional SCM provider, you can search on the **scm providers** extension category in the Extensions view (Ctrl+Shift+X). Start typing

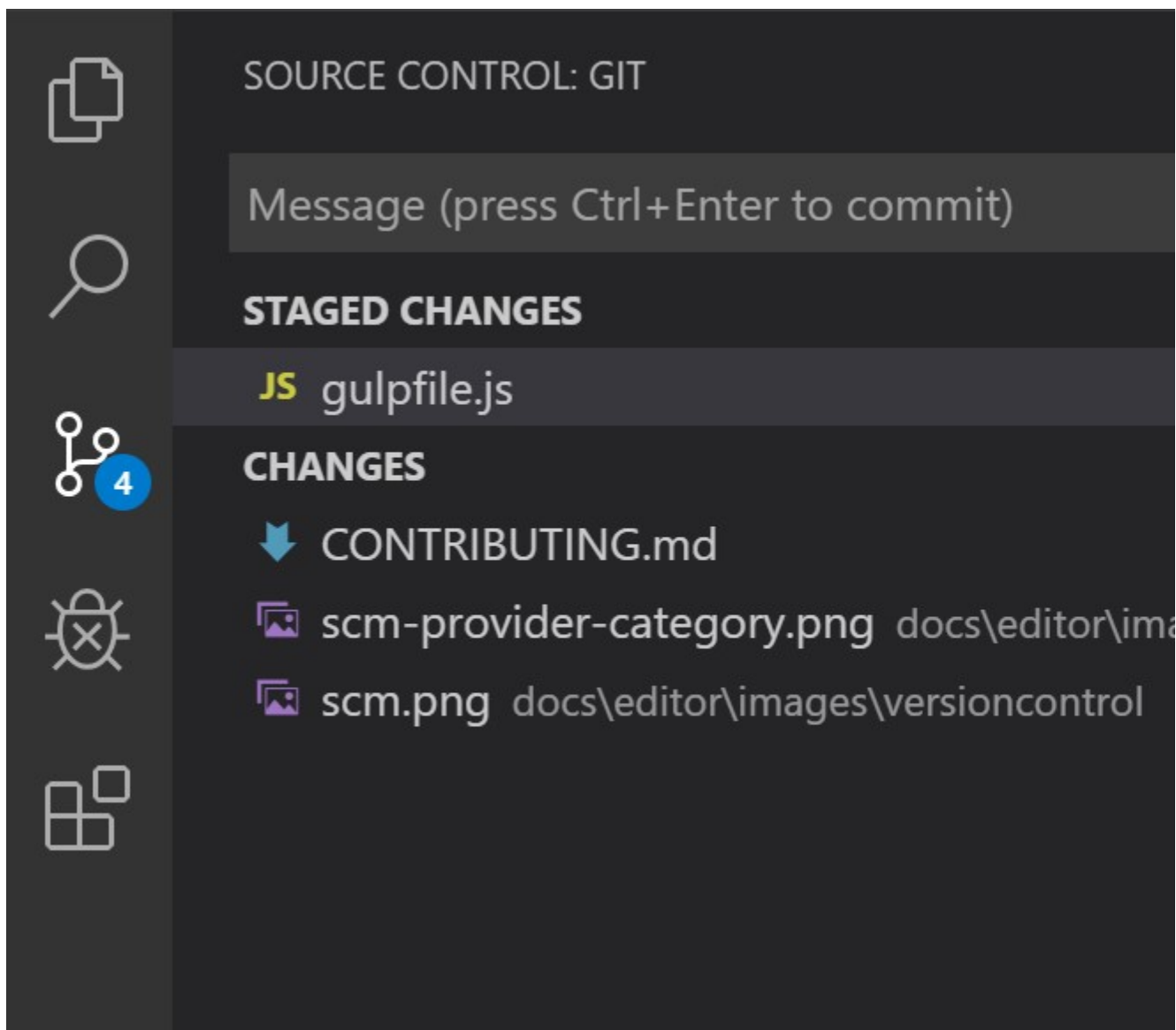
'@ca' and you will see suggestions for extension categories like debuggers and linters. Select `@category:"scm providers"` to see available SCM providers.



## Git support

VS Code ships with a Git source control manager (SCM) extension. Most of the source control UI and work flows are common across other SCM extensions, so reading about the Git support will help you understand how to use another provider.

**Note:** If you are new to Git, the [git-scm](#) website is a good place to start with a popular online [book](#), Getting Started [videos](#) and [cheat sheets](#). The VS Code documentation assumes you are already familiar with Git.



**Note:** VS Code will leverage your machine's Git installation, so you need to [install Git](#) first before you get these features. Make sure you install at least version 2.0.0.

**Tip:** VS Code will work with any Git repository. If you don't already have a private hosted Git provider, [Azure DevOps Services](#) is a great free option. You can sign up at [Get started with Azure DevOps](#).

The Source Control icon on the left will always indicate an **overview of how many changes** you currently have in your repository. Clicking it will show you the details of your current repository changes: **CHANGES**, **STAGED CHANGES** and **MERGE CHANGES**.

Clicking each item will show you in detail **the textual changes within each file**. Note that for unstaged changes, the editor on the right still lets you edit the file: feel free to use it!

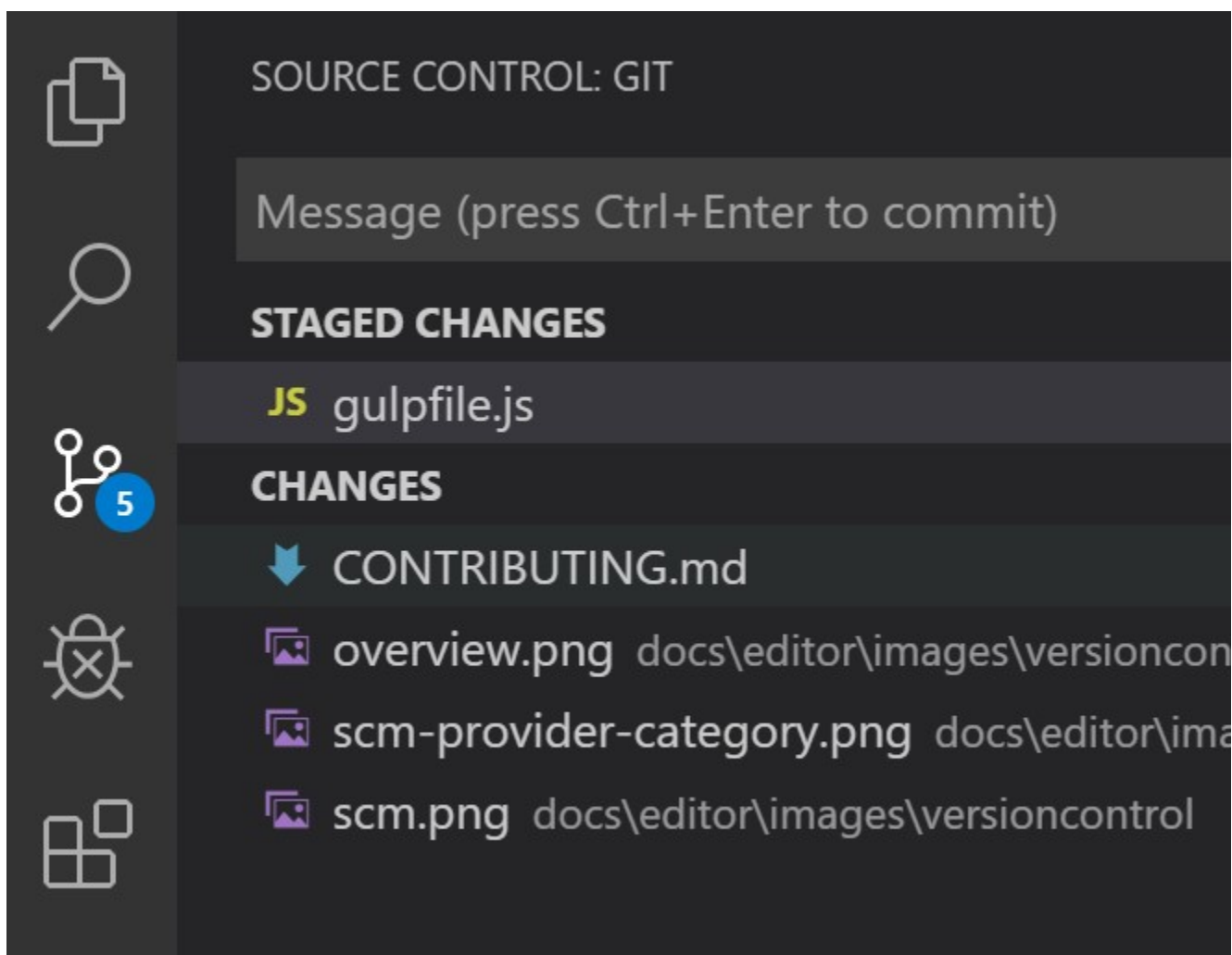
You can also find indicators of the **status of your repository** in the bottom left corner of VS Code: the **current branch**, **dirty indicators** and the number of **incoming and outgoing commits** of the current branch. You can **checkout** any

branch in your repository by clicking that status indicator and selecting the Git reference from the list.

**Tip:** You can open VS Code in a sub-directory of a Git repository. VS Code's Git services will still work as usual, showing all changes within the repository, but file changes outside of the scoped directory are shaded with a tool tip indicating they are located outside the current workspace.

## Commit

**Staging** (git add) and **unstaging** (git reset) can be done via contextual actions in the files or by drag-and-drop.

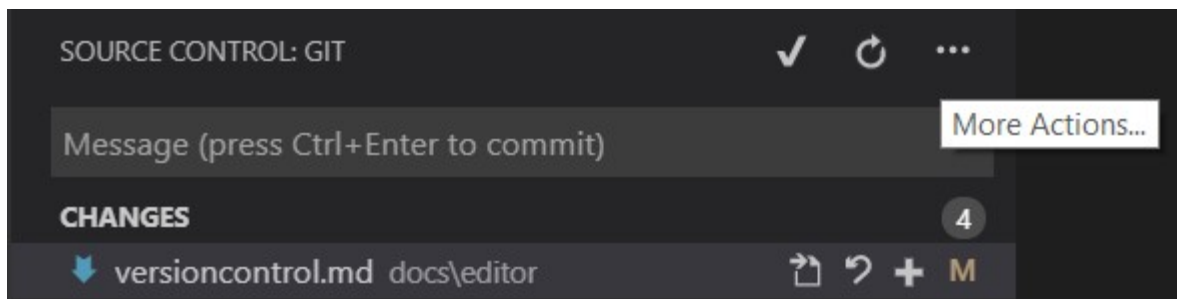


You can type a commit message above the changes and press Ctrl+Enter (macOS: ⌘+Enter) to commit them. If there are any staged changes, only those will be committed, otherwise all changes will be committed.

We've found this to be a great workflow. For example, in the earlier screenshot, only the staged changes to `gulpfile.js` will be included in the commit. A consecutive commit action could commit later changes to `gulpfile.js`, the deletion of `yarn.lock`, and changes to `tests.js` in a separate commit.

More specific **Commit** actions can be found in the **More Actions** . . . menu on the

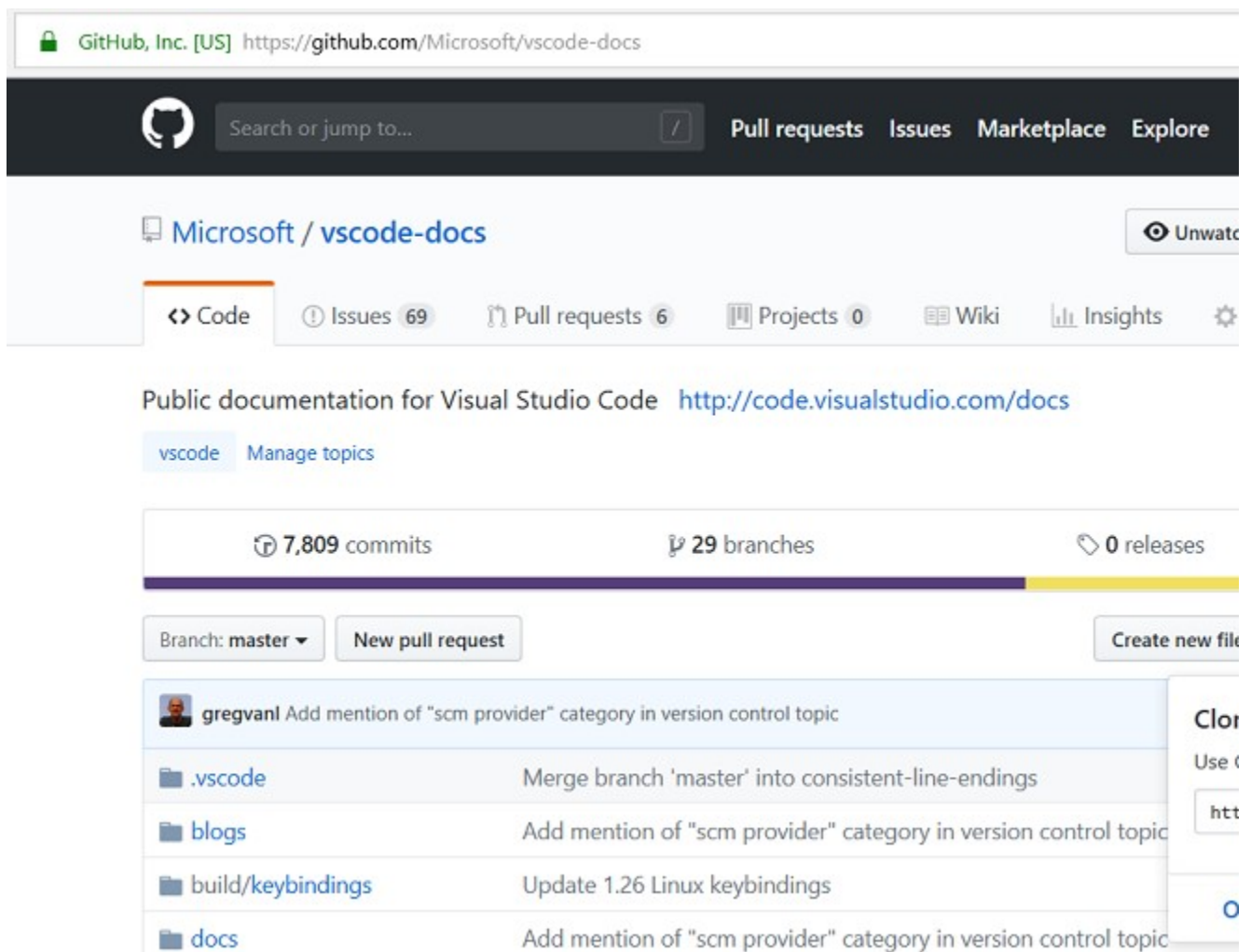
top of the Git view.



## Cloning a repository

You can clone a Git repository with the **Git: Clone** command in the **Command Palette** (Ctrl+Shift+P). You will be asked for the URL of the remote repository (for example on [GitHub](https://github.com)) and the parent directory under which to put the local repository.

For a GitHub repository, you would find the URL from the GitHub **Clone or download** dialog.



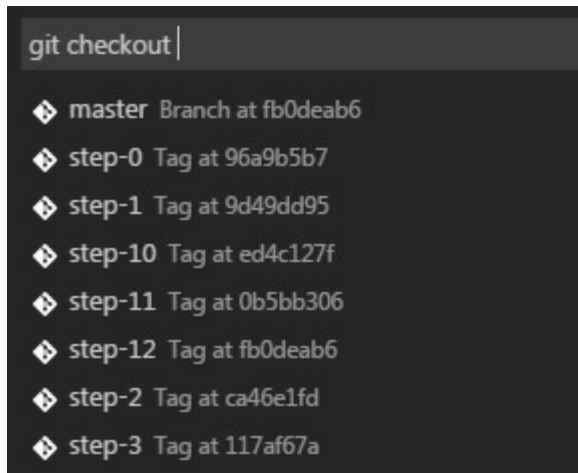
You would then paste that URL into the **Git: Clone** prompt.

A screenshot of the 'Repository URL' input field in Visual Studio Code. The text 'https://github.com/Microsoft/vscode-docs.git' is entered into the field.

Repository URL (Press 'Enter' to confirm or 'Escape' to cancel)

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).

If you run **Git: Checkout to**, you will see a drop-down list containing all of the branches or tags in the current repository.



The **Git: Create Branch** command lets you quickly create a new branch. Just provide the name of your new branch and VS Code will create the branch and switch to it.

## Remotes

Given that your repository is connected to some remote and that your checked out branch has an [upstream link](#) to a branch in that remote, VS Code offers you useful actions to **push**, **pull** and **sync** that branch (the latter will run a **pull** command followed by a **push** command). You can find these actions in the **More Actions** ... menu.

VS Code is able to periodically fetch changes from your remotes. This enables VS Code to show how many changes your local repository is ahead or behind the remote. Starting with VS Code 1.19, this feature is disabled by default and you can use the `git.autofetch` [setting](#) to enable it.

**Tip:** You should [set up a credential helper](#) to avoid getting asked for credentials every time VS Code talks to your Git remotes. If you don't do this, you may want to consider disabling automatic fetching via the `git.autofetch` [setting](#) to reduce the number of prompts you get.

## Git Status Bar actions

There is a **Synchronize Changes** action in the Status Bar, next to the branch



indicator, when the current checked out branch has an upstream branch configured. **Synchronize Changes** will pull remote changes down to your local repository and then push local commits to the upstream branch.



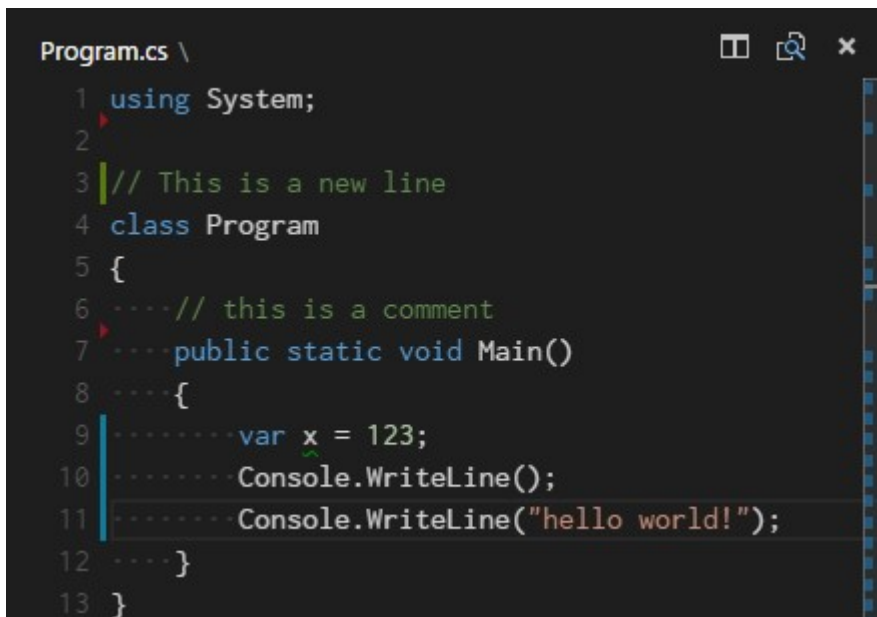
If there is no upstream branch configured and the Git repository has remotes set up, the **Publish** action is enabled. This will let you publish the current branch to a remote.



## Gutter indicators

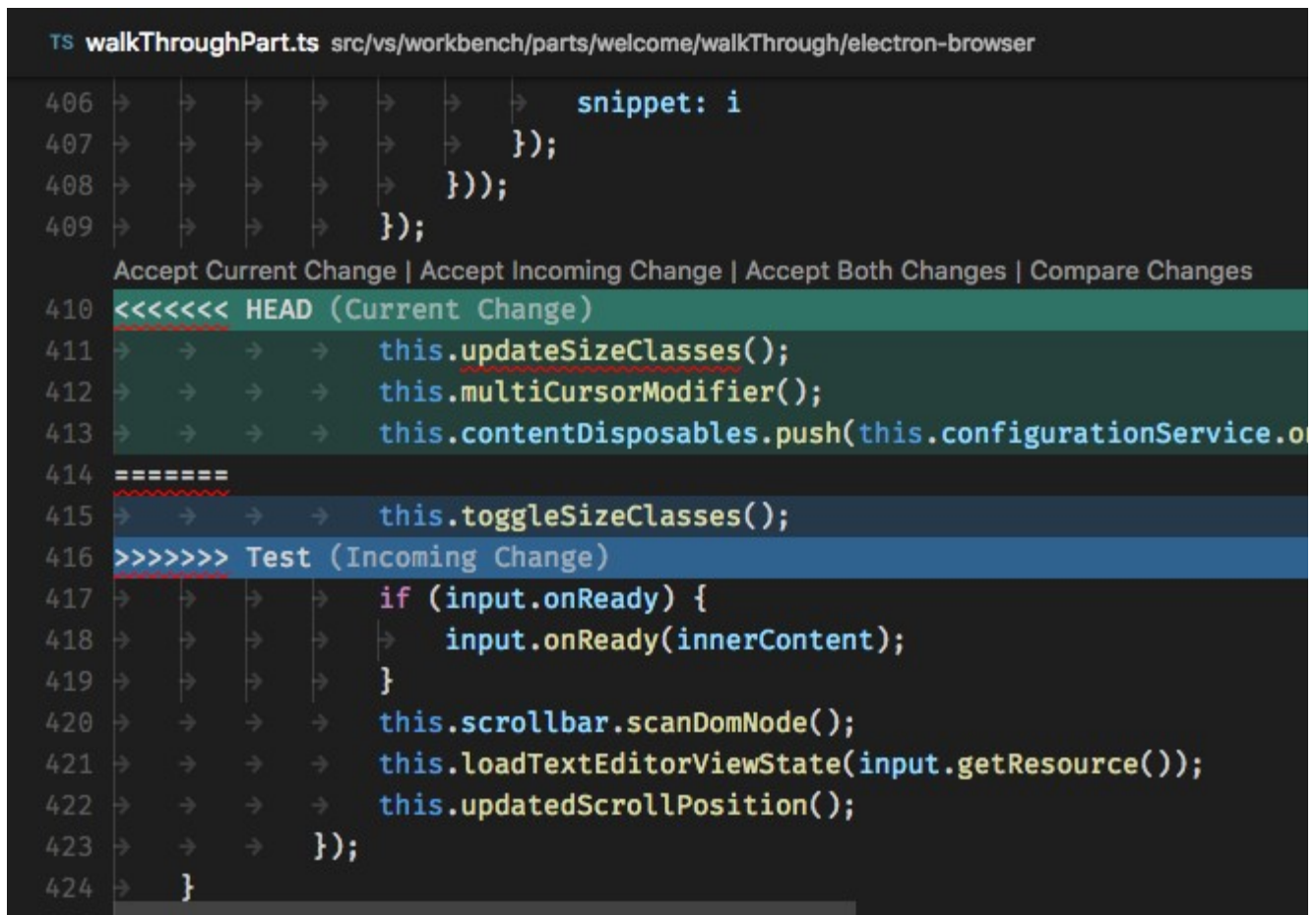
If you open a folder that is a Git repository and begin making changes, VS Code will add useful annotations to the gutter and to the overview ruler.

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



## Merge conflicts





```
TS walkThroughPart.ts src/vs/workbench/parts/welcome/walkThrough/electron-browser

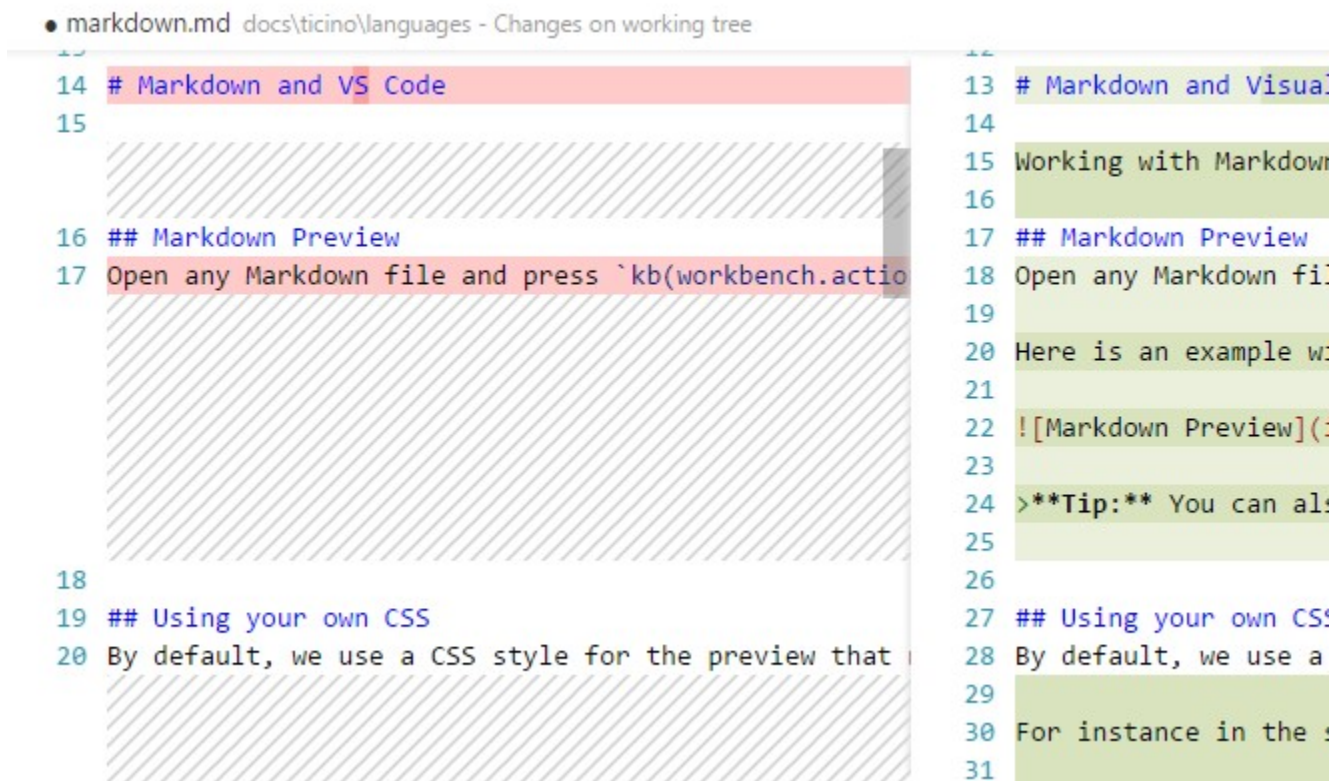
406 → → → → → snippet: i
407 → → → → → });
408 → → → → → }));
409 → → → → → });

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
410 <<<<<< HEAD (Current Change)
411 → → → → → this.updateSizeClasses();
412 → → → → → this.multiCursorModifier();
413 → → → → → this.contentDisposables.push(this.configurationService.o
414 =====
415 → → → → → this.toggleSizeClasses();
416 >>>>>> Test (Incoming Change)
417 → → → → → if (input.onReady) {
418 → → → → → | input.onReady(innerContent);
419 → → → → → }
420 → → → → → this.scrollbar.scanDomNode();
421 → → → → → this.loadTextEditorViewState(input.getResource());
422 → → → → → this.updatedScrollPosition();
423 → → → → → });
424 → }
```

Merge conflicts are recognized by VS Code. Differences are highlighted and there are inline actions to accept either one or both changes. Once the conflicts are resolved, stage the conflicting file so you can commit those changes.

## Viewing diffs

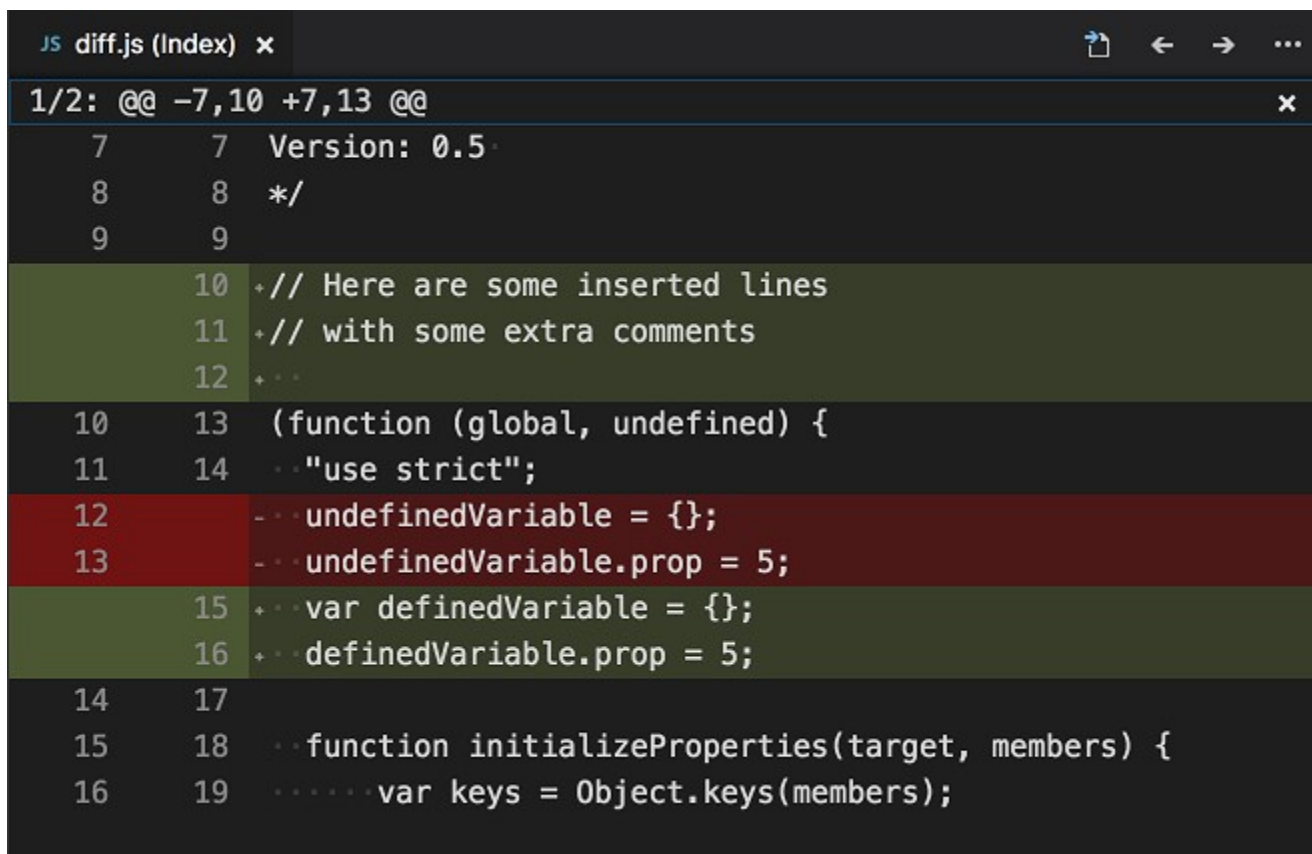
Our Git tooling supports viewing of diffs within VS Code.



**Tip:** You can diff any two files by first right clicking on a file in the Explorer or **OPEN EDITORS** list and selecting **Select for Compare** and then right-click on the second file to compare with and select **Compare with 'file\_name\_you\_chose'**. Alternatively from the keyboard hit Ctrl+Shift+P and select **File: Compare Active File With** and you will be presented with a list of recent files.

## Diff editor review pane

There is a review pane in the Diff editor which presents changes in a unified patch format. You can navigate between changes with **Go to Next Difference** (F7) and **Go to Previous Difference** (Shift+F7). Lines can be navigated with arrow keys and pressing Enter will jump back in the Diff editor and the selected line.



```
JS diff.js (Index) x
1/2: @@ -7,10 +7,13 @@
7      7  Version: 0.5
8      8  */
9      9
10     10  +// Here are some inserted lines
11     11  +// with some extra comments
12     12  +...
10     13  (function (global, undefined) {
11     14    .."use strict";
12     15    -..undefinedVariable = {};
13     16    -..undefinedVariable.prop = 5;
14     17    +..var definedVariable = {};
15     18    +..definedVariable.prop = 5;
16     19
14     17
15     18    ..function initializeProperties(target, members) {
16     19    .....var keys = Object.keys(members);
```

**Note:** This experience is especially helpful for screen reader users.

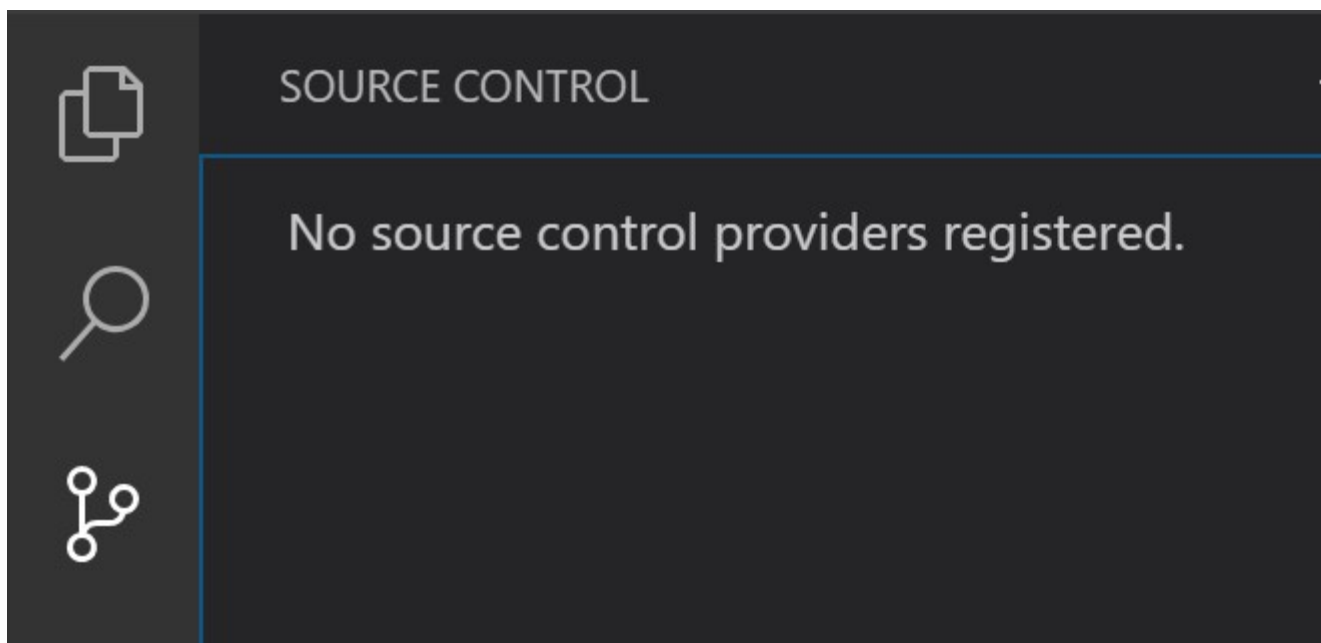
## Git output window

You can always peek under the hood to see the Git commands we are using. This is helpful if something strange is happening or if you are just curious. :)

To open the Git output window, run **View > Output** and select **Git** from the dropdown list.

## Initialize a repository

If your workspace isn't under Git source control, you can easily create a Git repository with the **Initialize Repository** command. When VS Code doesn't detect an existing Git repository, you will see a **No source control providers registered.** message in the Source Control view and the **Initialize Repository** command will be available on the title bar. You can also run the **Git: Initialize Repository** command from the **Command Palette** (Ctrl+Shift+P).



Running **Initialize Repository** will create the necessary Git repository metadata files and show your workspace files as untracked changes ready to be staged.

## VS Code as Git editor

When you launch VS Code from the command line, you can pass the `--wait` argument to make the launch command wait until you have closed the new VS Code instance. This can be useful when you configure VS Code as your Git external editor so Git will wait until you close the launched VS Code instance.

Here are the steps to do so:

1. Make sure you can run `code --help` from the command line and you get help.
  - if you do not see help, please follow these steps:
    - macOS: Select **Shell Command: Install 'Code' command in path** from the **Command Palette**.
    - Windows: Make sure you selected **Add to PATH** during the installation.
    - Linux: Make sure you installed Code via our new `.deb` or `.rpm` packages.
2. From the command line, run `git config --global core.editor "code --wait"`

Now you can run `git config --global -e` and use VS Code as editor for configuring Git.

## VS Code as Git diff tool

Add the following to your Git configurations to use VS Code as the diff tool:

```
[diff]
  tool = default-difftool
[difftool "default-difftool"]
  cmd = code --wait --diff $LOCAL $REMOTE
```

This leverages the `--diff` option you can pass to VS Code to compare 2 files side by side.

To summarize, here are some examples of where you can use VS Code as the editor:

- `git rebase HEAD~3 -i` do interactive rebase using VS Code
- `git commit` use VS Code for the commit message
- `git add -p` followed by `e` for interactive add
- `git difftool <commit>^ <commit>` use VS Code as the diff editor for changes

## Working with pull requests

Visual Studio Code also supports pull request workflows through [extensions](#) available on the VS Code Marketplace. Pull request extensions let you review, comment, and verify source code contributions directly within VS Code.

**Tip:** Click on an extension tile to read the description and reviews in the Marketplace.

## Next steps

- [Intro Video - Git Version Control](#) - An introductory video providing an overview of VS Code Git support.
- [Basic Editing](#) - Learn about the powerful VS Code editor.
- [Code Navigation](#) - Move quickly through your source code.
- [Debugging](#) - This is where VS Code really shines
- [Tasks](#) - Running tasks with Gulp, Grunt and Jake. Showing Errors and Warnings
- [Source Control API](#) - If you want to integrate another Source Control provider into VS Code, see our Source Control API.

## Common questions

To **push**, **pull**, and **sync** you need to have a Git origin set up. You can get the required URL from the repository host. Once you have that URL, you need to add it to the Git settings by running a couple of command-line actions. For example:

```
> git remote add origin https://github.com/<repo owner>/<repo name>.git
> git push -u origin master
```

### My team is using Team Foundation Version Control (TFVC) instead of Git. What should I do?

Use the [Azure Repos](#) extension and this will light up TFVC support.

### Why do the Pull, Push and Sync actions never finish?

This usually means there is no credential management configured in Git and you're not getting credential prompts for some reason.

You can always set up a [credential helper](#) in order to pull and push from a remote

server without having VS Code prompt for your credentials each time.

## **How can I sign in to Git with my Azure DevOps organization which requires multi-factor authentication?**

There are now [Git credential helpers](#) that assist with multi-factor authentication. You can download these from [Git Credential Manager for Mac and Linux](#) and [Git Credential Manager for Windows](#).

## **I have GitHub Desktop installed on my computer but VS Code ignores it**

VS Code only supports the [official Git distribution](#) for its Git integration.

## **I keep getting Git authentication dialogs whenever VS Code is running**

VS Code automatically fetches changes from the server in order to present you with a summary of incoming changes. The Git authentication dialog is independent from VS Code itself and is a part of your current Git credential helper.

One way to avoid these prompts is to set up a [credential helper](#) which remembers your credentials.

Another option is to disable the auto fetch feature by changing the following setting:

```
"git.autofetch": false.
```

## **Can I use SSH Git authentication with VS Code?**

Yes, though VS Code works most easily with SSH keys without a passphrase. If you have an SSH key with a passphrase, you'll need to launch VS Code from a Git Bash prompt to inherit its SSH environment.

4/8/2020