

Search

Introduction to OpenShift Applications

🚩 Tell us what you think!

Congratulations on completing at least 25% of the course. We would appreciate feedback on your learning experience. Click below to fill out a brief survey.

TAKE ME TO THE SURVEY ([HTTPS://SURVEY.OLE.REDHAT.COM/SURVEY?COURSE=DO101&MODALITY=COURSE&OFFERING=101](https://survey.ole.redhat.com/survey?course=DO101&modality=course&offering=101))

NOT RIGHT NOW

☐ Don't ask again

TRANSLATIONS ▾



P	(/rol/app/courses/do101-4.2/pages/pr01)	(/rol/app/courses/do101-4.2/pages/pr01s02)	1	(/rol/app/courses/do101-4.2/pages/ch01)
(/rol/app/courses/do101-4.2/pages/pr01)	(/rol/app/courses/do101-4.2/pages/ch01s02)	(/rol/app/courses/do101-4.2/pages/ch01s03)	(/rol/app/courses/do101-4.2/pages/ch01)	(/rol/app/courses/do101-4.2/pages/ch01s04)
4.2/pages/pr01	(/rol/app/courses/do101-4.2/pages/ch01s05)	(/rol/app/courses/do101-4.2/pages/ch01s06)	(/rol/app/courses/do101-4.2/pages/ch01)	(/rol/app/courses/do101-4.2/pages/ch01s07)
(/rol/app/courses/do101-4.2/pages/ch02)	(/rol/app/courses/do101-4.2/pages/ch02s02)	(/rol/app/courses/do101-4.2/pages/ch02s03)	(/rol/app/courses/do101-4.2/pages/ch02s03)	(/rol/app/courses/do101-4.2/pages/ch02s04)
(/rol/app/courses/do101-4.2/pages/ch03)	(/rol/app/courses/do101-4.2/pages/ch03s02)	(/rol/app/courses/do101-4.2/pages/ch03s03)	(/rol/app/courses/do101-4.2/pages/ch03s03)	(/rol/app/courses/do101-4.2/pages/ch03s04)
(/rol/app/courses/do101-4.2/pages/ch03s04)	(/rol/app/courses/do101-4.2/pages/ch03s05)	(/rol/app/courses/do101-4.2/pages/ch03s06)	(/rol/app/courses/do101-4.2/pages/ch03s06)	(/rol/app/courses/do101-4.2/pages/ch03s07)
(/rol/app/courses/do101-4.2/pages/ch03s07)	4 (/rol/app/courses/do101-4.2/pages/ch04)	(/rol/app/courses/do101-4.2/pages/ch04s02)	(/rol/app/courses/do101-4.2/pages/ch04s02)	(/rol/app/courses/do101-4.2/pages/ch04s03)
(/rol/app/courses/do101-4.2/pages/ch04s03)	(/rol/app/courses/do101-4.2/pages/ch05)	(/rol/app/courses/do101-4.2/pages/ch05s02)	(/rol/app/courses/do101-4.2/pages/ch05s02)	(/rol/app/courses/do101-4.2/pages/ch05s03)
	4.2/pages/ch05			

← PREVIOUS (/ROL/APP/COURSES/DO101-4.2/PAGES/CH01S02)

→ NEXT (/ROL/APP/COURSES/DO101-4.2/PAGES/CH01S04)

Initializing a Git Repository



Objectives

After completing this section, you should be able to create a Git repository.

Software Version Control

A Version Control System (VCS) enables you to efficiently manage and collaborate on code changes with others. Version control systems provide many benefits, including:

- The ability to review and restore old versions of files.
- The ability to compare two versions of the same file to identify changes.
- A record or log of who made changes at a particular time.
- Mechanisms for multiple users to collaboratively modify files, resolve conflicting changes, and merge the changes together.

There are several open source version control systems available including:

- CVS
- SVN
- Git
- Mercurial

Introducing Git

Git is the most popularly used version control system. For this reason, you use Git as the version control system for all the exercises in this course.

Git can convert any local system folder into a Git repository. Although you have many of the benefits of version control, your Git repository only exists on your local system.

To share your repository with another collaborator, you must host the repository on a code repository platform.

There are many free code repository platforms, including:

- GitHub
- GitLab
- BitBucket
- SourceForge

In this course, you use GitHub to host and share your Git repositories.

Git Workflow Overview

To retrieve the project files for an existing software project with Git, you **clone** the Git repository for the project.

When you clone a project, a complete copy of the original remote repository is created locally on your system. Your local copy of the repository contains the entire history of the project files, not just the latest version of project files. You can switch to different versions of the application, or compare two different versions of a file, without connecting to the remote repository. This allows you to continue implementing code changes when the remote repository is not available.

Git does not automatically synchronize local repository changes to the remote repository, nor does it automatically download new remote changes to your local copy of the repository. You control when Git downloads commits from the remote repository, and when local commits are uploaded to the remote repository. Git provides several mechanisms to safely synchronize your

local repository with the remote project repository.

To track file changes in Git, you create a series of project snapshots as you make changes to your project. Each snapshot is called a **commit**. When you commit code changes to your repository, you create a new code snapshot in your Git repository.

Each commit contains metadata to help you find and load this snapshot at a later time:

- **commit message** - A high level summary of the file changes in the commit.
- **timestamp** - The date and time that the commit was created.
- **author** - A field that describes who created the commit.
- **commit hash** - A unique identifier for the commit. A commit hash consists of 40 hexadecimal numbers. If a Git command requires a commit hash to perform an operation, then you can abbreviate the commit to seven characters.

After you create commits in a local repository on your system, you must **push** your changes to the remote repository. When you push changes to a remote Git repository, you upload local commits to the remote repository. After a push, your commits are available for others to download.

When other contributors push commits to a remote repository, those commits are not present in your local repository. To download new commits from other contributors, you **pull** changes from the remote Git repository.

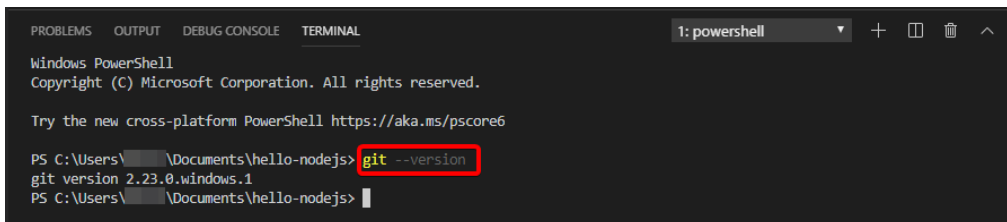
Installing Git

Git is an open source version control system that is available for Linux, MacOS, and Windows systems. Before you can use Git, you must install it.

In a browser, navigate to <https://git-scm.com/downloads> (<https://git-scm.com/downloads>) and follow the directions for your operating system.

After installing Git on your system, you can use VS Code to manage your Git source code repositories.

To test your Git installation, open VS Code and access the integrated terminal (**View** → **Terminal**). At the terminal prompt, execute `git --version`. If Git is correctly installed, then a version number prints in the terminal:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\...\Documents\hello-nodejs> git --version
git version 2.23.0.windows.1
PS C:\Users\...\Documents\hello-nodejs>
```

Figure 1.9: Testing the installation of Git in VS Code

Configure the Source Control view in VS Code

Use the VS Code Command Palette (**View** → **Command Palette...**) and the Source Control view (**View** → **SCM**) to execute Git operations, such as cloning a repository or committing code changes.

By default, the VS Code Source Control view is different when you have one Git repository in your workspace, compared to when you have multiple Git repositories in your workspace.

When you have multiple Git repositories in your workspace, then the Source Control view displays a **SOURCE CONTROL PROVIDERS** list:

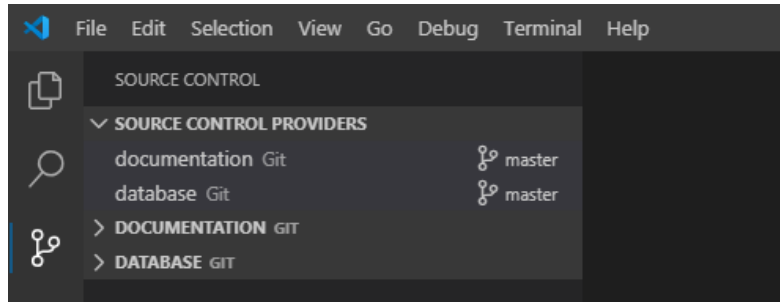


Figure 1.10: The Source Control Providers list in the Source Control view of VS Code.

By default, when there is only a single Git repository in your workspace, then the Source Control view does not display the SOURCE CONTROL PROVIDERS list.

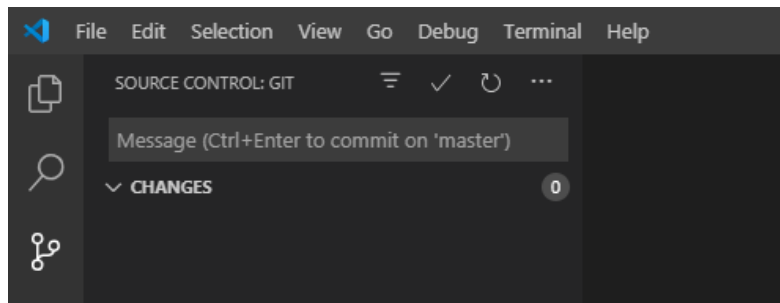


Figure 1.11: Source Control view for a single Git repository.

For a consistent user interface, independent of the number of Git repositories in your workspace, you must enable the Always Show Providers source control management option.

To enable this option, access the Command Palette (**View** → **Command Palette...**) and type settings. Select Preferences: Open Settings (UI) from the list of options. VS Code displays a settings window:

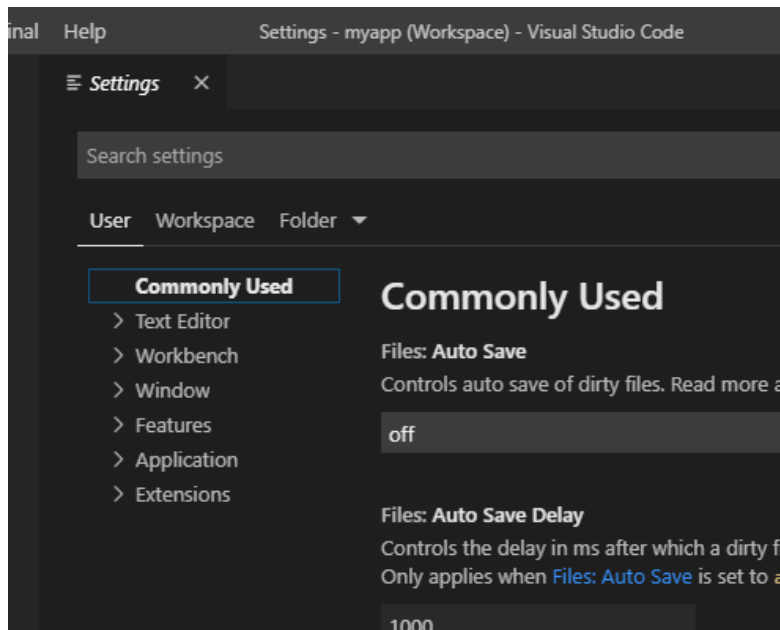


Figure 1.12: The Settings window for VS Code.

Click **User**, and then click **Features** → **SCM**. VS Code displays Source Control Management (SCM) options for VS Code. Select **Always Show Providers**

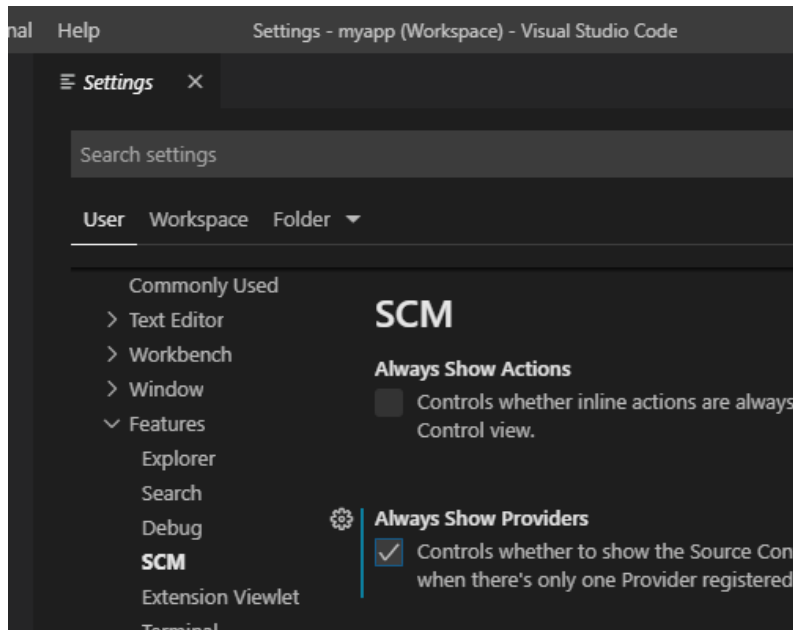


Figure 1.13: Option to always show the Source Control Providers list in the Source Control view of VS Code.

When you enable this option, then VS Code displays the `SOURCE CONTROL PROVIDERS` list in the Source Control view for any number of workspace Git repositories, including only one repository.

Cloning a Git Repository

Use the VS Code Command Palette (**View** → **Command Palette...**) and the Source Control view (**View** → **SCM**) to execute Git operations, such as cloning a repository or committing code changes.

To clone a remote Git repository in VS Code, access the Command Palette (**View** → **Command Palette...**). Type `clone` at the prompt, and then select `Git: Clone` from the list of options.

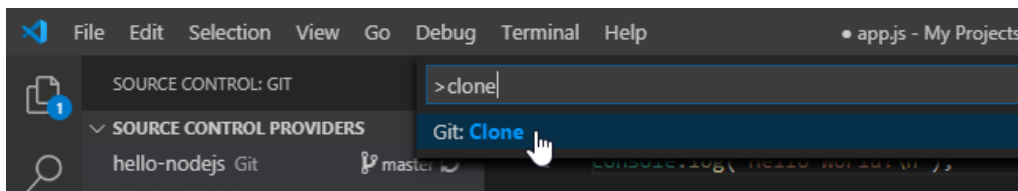


Figure 1.14: Using the command palette to clone a repository.

VS Code prompts you for the URL of the remote repository, and then prompts for the desired location of the local repository on your file system.

After VS Code clones the repository, add the cloned repository folder to your VS Code workspace.

Committing Code Changes

After adding a cloned repository to your VS Code workspace, you can edit project files like any other workspace files. As you edit project files, Git assigns a status to each file:

- **modified** - The file contains saved differences from the most recent version. Modified files are not automatically added or committed to your Git repository.
- **staged** - A staged file is a modified file that you flag to be included as part of your next code commit to the repository.

When you commit code to the repository, only those files with a **staged** status are included in the commit. If your project contains modified files that are not staged, then those files are not included in the commit. This feature enables you to control the file changes that are included in each commit.

In VS Code, the Source Control view (**View** → **SCM**) displays all modified and staged repository files. After you save edits to a file, the file name displays in the `CHANGES` list.

To add a modified file to your next code commit, click the modified file in the CHANGES list, from the Source Control view. VS Code displays a new tab to highlight the changes to the file:

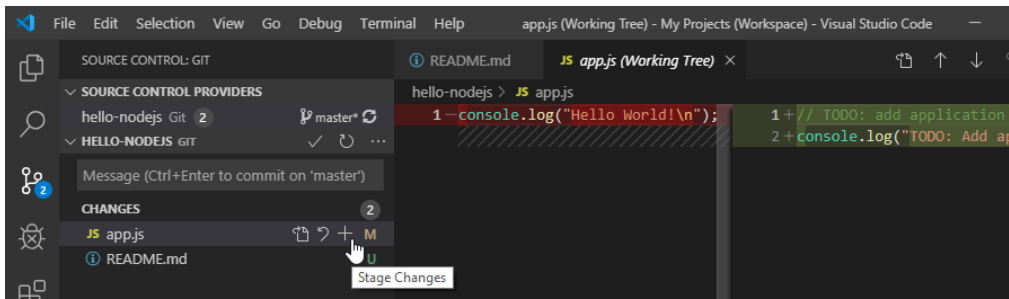


Figure 1.15: Review file changes before staging a file.

If the file changes are accurate and complete, click **Stage Changes** to add the file changes to your next code commit.

After all of your desired file changes are staged, provide a commit message in the Source Control view. Then, select the check box to commit all of the staged file changes to your local repository.

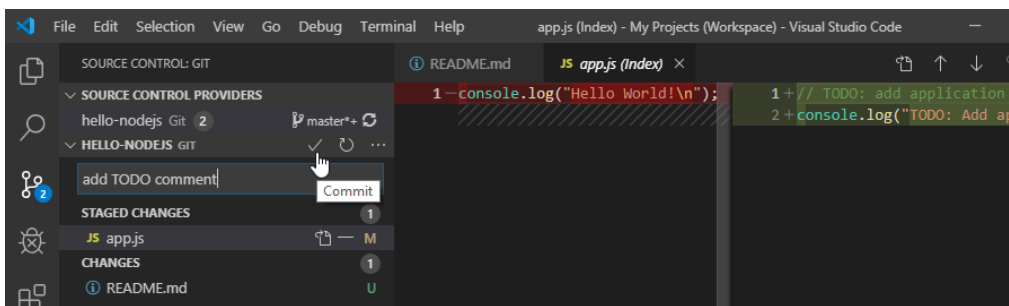


Figure 1.16: Commit staged files in VS Code.

Using a Remote Repository

When you commit code changes, you only commit code to your local repository. No changes are made to the remote repository.

When you are ready to share your work, synchronize your local repository to the remote repository. To retrieve commits from the remote repository, Git performs a pull operation. To publish local commits to the remote repository, Git performs a push operation. VS Code handles the pull and push Git operations when you synchronize your local repository to the remote repository.

The Source Control view compares your local repository with the corresponding remote repository. If there are commits to download from the remote repository, then the number of commits displays with a download arrow icon. If there are local commits that you have not published to the remote repository, then the number of commits appears next to an upload arrow icon.

In the figure that follows, there are zero commits to download and one commit to upload to the remote repository:

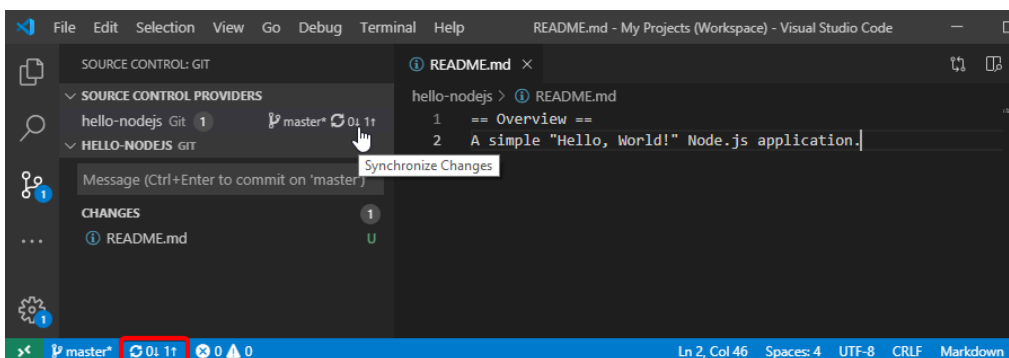


Figure 1.17: Git repository status in the Source Control view and status bar.

Click **Synchronize Changes** to publish your local code commits to the remote repository. Alternatively, you can click the same icon in the status bar to publish your code commits.

IMPORTANT

If the Source Control view (**View** → **SCM**) does not contain a **SOURCE CONTROL PROVIDERS** heading, then you will not see a **Synchronize Changes** icon.

To enable it, right-click **SOURCE CONTROL: GIT**. If a check mark does not display to the left of **Source Control Providers**, then click **Source Control Providers**.

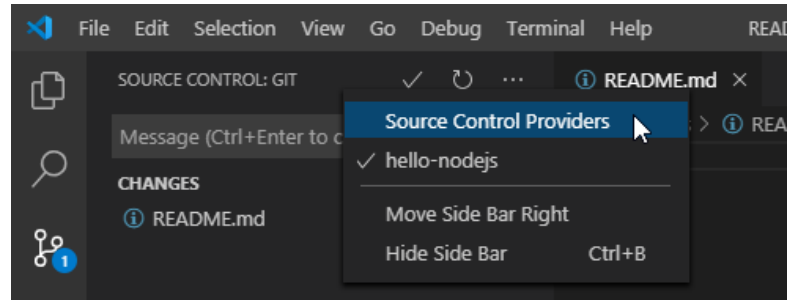


Figure 1.18: Enable the source control providers listing.

Initializing a New Git Repository

If you have an existing software project that needs version control, then you can convert it to a Git repository.

To convert any file system folder into a Git repository, access the VS Code Command Palette (**View** → **Command Palette...**). Type `intialize` at the prompt, and then select `Git: Initialize Repository` from the list of options.

VS Code displays a list of project folders in the workspace.

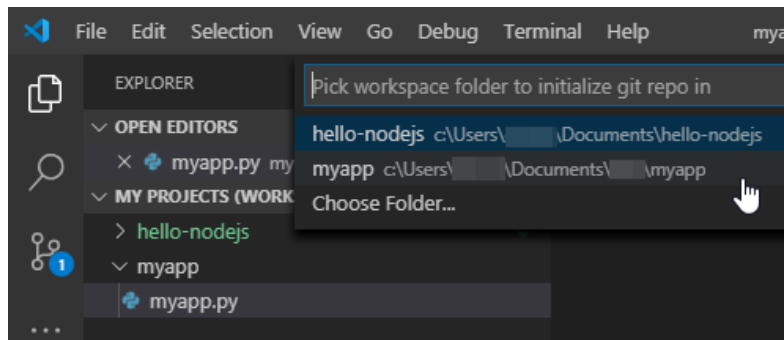


Figure 1.19: Using VS Code to initialize a Git repository.

After you select a project folder, Git initializes the folder as an empty Git repository. The Source Control view displays an entry for the new repository. Because the folder is initialized as an empty repository, every file in the project folder is marked as an untracked file.

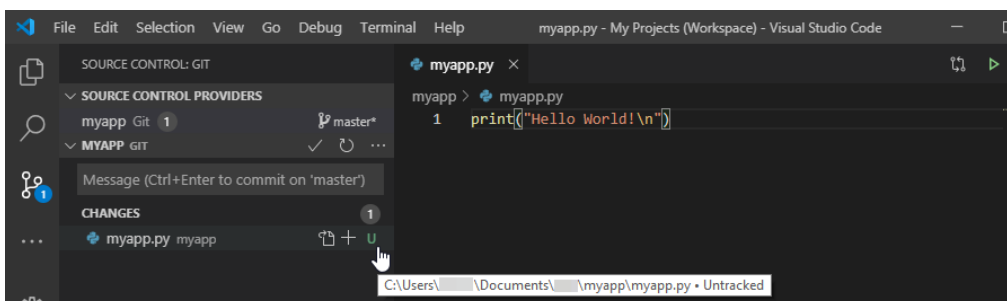


Figure 1.20: Untracked files in a Git repository.

For any project file that requires version control, click the plus icon to add the file to the local repository. Each added file displays in the **STAGED CHANGES** list in the Source Control view. After you stage all of the project files, provide a commit message, and then click the check mark icon to create the first commit in the repository.

When you create a new repository from a local file system folder, the new repository is not associated with a remote repository. If you need to share your repository:

1. Create a new Git repository on a code hosting platform, such as GitHub.
2. Associate your local repository to the new remote repository, and then synchronize changes.

Adding a Remote Repository to a Local Repository

After you create a new repository on a code hosting platform, the platform provides you with a HTTPS and SSH URL to access the repository. Use this URL to add this hosted repository as a remote repository for your local repository.

NOTE

In this course, you only use HTTPS URLs to access remote code repositories. HTTPS access to a Git repository requires very little additional configuration, but does require that you provide credentials for the code hosting platform.

You can configure your Git installation to cache your credentials. This helps minimize re-entering credentials each time you connect to the remote repository.

SSH access to Git repository requires the configuration of your SSH keys with the code hosting platform.

SSH key configuration is beyond the scope of this course.

Access the VS Code Command Palette to add a remote repository to your local repository. Type `add remote` at the prompt, and then select `Git: Add Remote` from the list of options. If you are prompted to select a local repository, then make an appropriate selection.

At the prompt, enter `origin` for the remote name; `origin` is the conventional name given to the remote repository that is designated as the central code repository.

At the prompt, enter the HTTPS URL for your remote repository. If you have commits in your local repository, a `Publish Changes` icon displays in the **SOURCE CONTROL PROVIDERS** list.

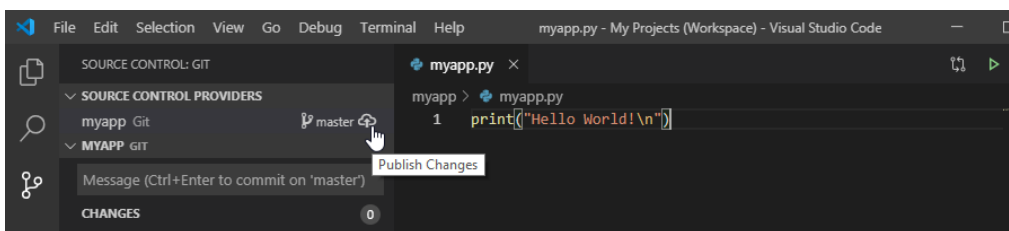


Figure 1.21: Publish a local Git repository to a remote repository.

Click the `Publish Changes` icon to push your local commits to the remote repository. If you are prompted, then provide the necessary remote repository credentials.

REFERENCES

For more information about installing Git, refer to the Git documentation at <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git> (<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>)

Git Basics (<https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>)

For more information about using Git and version control in VS Code, refer to the VS Code documentation at <https://code.visualstudio.com/docs/editor/versioncontrol> (<https://code.visualstudio.com/docs/editor/versioncontrol>)

← [PREVIOUS \(/ROL/APP/COURSES/DO101-4.2/PAGES/CH01S02\)](/ROL/APP/COURSES/DO101-4.2/PAGES/CH01S02)

[NEXT \(/ROL/APP/COURSES/DO101-4.2/PAGES/CH01S04\)](/ROL/APP/COURSES/DO101-4.2/PAGES/CH01S04) →