

00000111

CHAPTER 7

Part II Preliminaries

Ch-7: Part II Preliminaries

Learning Objectives

- *Verify the installation of Git*
- *Update to the latest version of Git*
- *Test Git from the command line*
- *Sign up for a GitHub account*
- *Install Pipenv*
- *Verify Pipenv command*
- *Set Bash as the default shell on macOS*
- *Update Bash on macOS*

0
0
0
0
0
0
1
1
1

INTRODUCTION

This is a short but important chapter as it prepares you for what lies ahead in the remaining chapters of Part II. I named Part II “Foundations” because you will use what you learn here throughout the remainder of this book and throughout your software development career. These skills include the use of *Git* for source code configuration management, the use of *GitHub* as the remote repository, the use of the *Bash* shell to write scripts to orchestrate complex build tasks, and the use of *Pipenv* to create virtual environments that enable development with different versions of Python on the same machine.

With the exception of *Pipenv*, you can use *Git*, *GitHub*, and *Bash* to aid your software development efforts regardless of programming language. These skills also serve as the foundation for a set of software development, deployment, and operations practices referred to as *DevOps*.

Follow the steps in this chapter to ensure your development environment is properly configured before proceeding with chapters 8, 9, 10, & 11. In that regard, consider the tasks you carry out here an extension to the baseline development environment you previously configured in Chapter 1: Part I Preliminaries: Baseline Development Environment. Let’s start by verifying the installation of *Git*.

1 VERIFY INSTALLATION OF GIT

This section prepares you for *Chapter 8: Configuration Management with Git & GitHub* by showing you how to verify the installation of *Git* and how to update *Git* to the latest version. While running the latest version of *Git* is not strictly required, it’s always a good idea to use the latest versions of software development tools as they plug security holes and generally offer improved performance. I’ll proceed by operating system, starting with Microsoft Windows.

1.1 MICROSOFT WINDOWS

Recall from chapter 1, in order to get a *bash* terminal on a Windows machine, you installed *Git For Windows*, which gave you the *Git Bash* (*mintty*) terminal. *Git For Windows* installs *Git*, so launch a *Git Bash* terminal and type `git` at the command prompt as shown in figure 7-1.

Referring to figure 7-1 — Running `git` from the command line without any arguments results in help being printed to the console. If you see this, you’re good to go.

1.2 MACOS

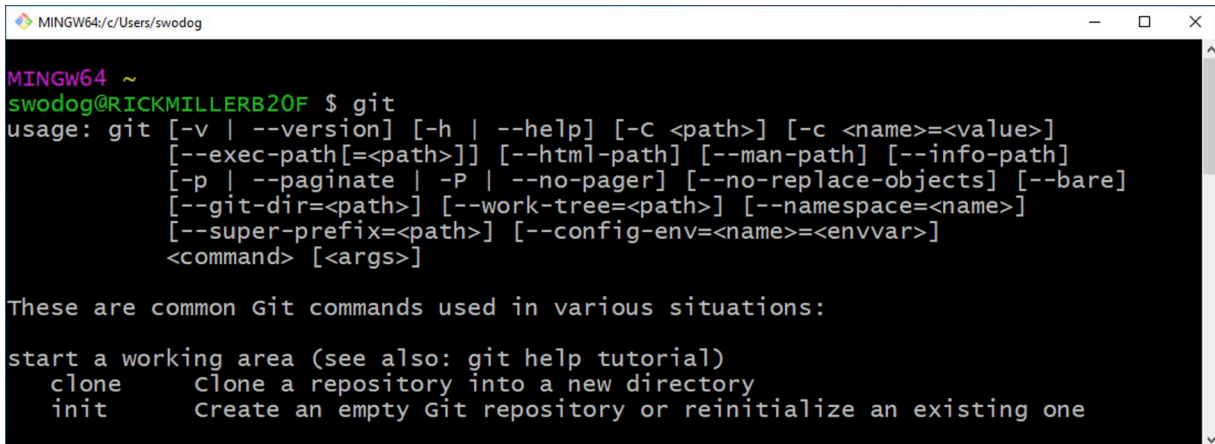
Git comes with macOS. Launch a terminal and type `git` at the command prompt as shown in figure 7-2.

Referring to figure 7-2 — OK, so far so good. Next, let’s check which version of `git` is running. At the command prompt type the following command:

```
git --version
```

Figure 7-3 shows the results on my machine.

Referring to figure 7-3 — It shows I’m running `git version 2.21.1 (Apple Git-122.3)`. The version number you’ll see depends on the version of macOS you’re running.



```

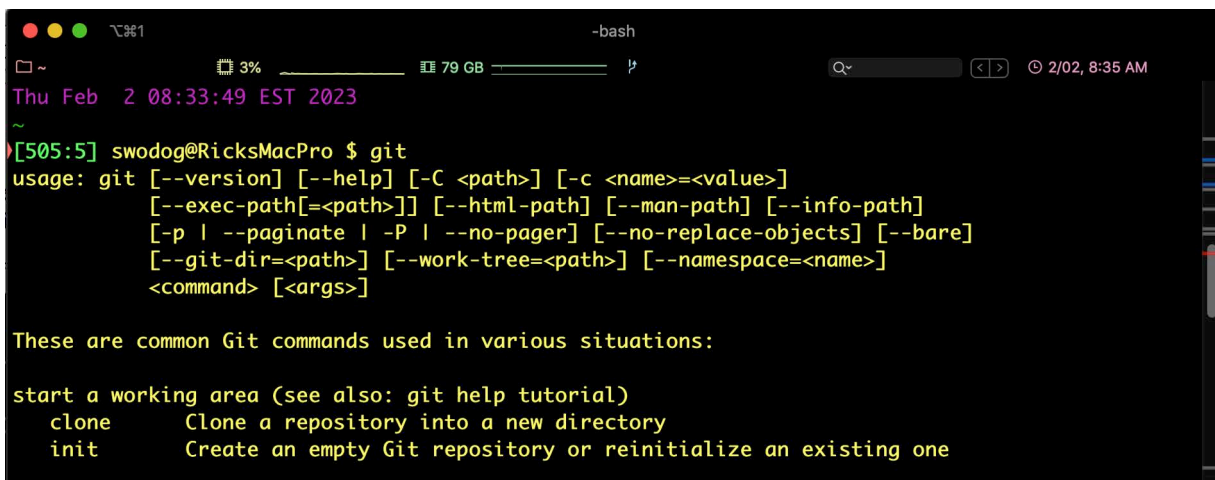
MINGW64 ~
swodog@RICKMILLERB20F $ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          [--super-prefix=<path>] [--config-env=<name>=<envvar>]
          <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

```

Figure 7-1: Running git Command in Git Bash Terminal



```

~
Thu Feb  2 08:33:49 EST 2023
~
[505:5] swodog@RicksMacPro $ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

```

Figure 7-2: Running git Command in iTerm2 on macOS



```

~
Thu Feb  2 08:39:11 EST 2023
~
[508:8] swodog@RicksMacPro $ git --version
git version 2.21.1 (Apple Git-122.3)

```

Figure 7-3: Checking Git Version macOS

1.2.1 UPDATE GIT WITH BREW

It's always a good idea to run the latest edition of development tools as newer versions plug security holes and generally offer improved performance. I said that already...anyway, a check on the [Git-SCM](#) site lists the latest version as 2.39.1, so I'm going to update Git via the brew package manager. At the command prompt type the following command:

```
brew info git
```

Figure 7-4 show the results.

```

~
[509:9] swodog@RicksMacPro $ brew info git
==> git: stable 2.39.0, HEAD
Distributed revision control system
https://git-scm.com
Not installed
From: https://github.com/Homebrew/homebrew-core/blob/HEAD/Formula/git.rb
License: GPL-2.0-only
==> Dependencies
Required: gettext ✓, pcre2 ✓, curl ✗
==> Options
--HEAD
    Install HEAD version
==> Caveats
The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the `git-gui` formula.
Subversion interoperability (git-svn) is now in the `git-svn` formula.
==> Analytics
install: 433,679 (30 days), 1,040,851 (90 days), 3,731,673 (365 days)
install-on-request: 426,676 (30 days), 1,023,727 (90 days), 3,659,895 (365 days)
build-error: 29 (30 days)

```

Figure 7-4: Checking git Package Info with Brew

Referring to figure 7-4 — Looks like brew has the latest version, (*The version may be different after this book is published, for sure.*) so I’m going to install it. It looks like it’ll install the curl utility as well. To install the newer version of git with brew, type the following command:

```
brew install git
```

When installation completes, close and relaunch the terminal, then verify the updated version is working as shown in figure 7-5.

```

~
[502:2] swodog@RicksMacPro $ git --version
git version 2.39.1

```

Figure 7-5: Updated Git

1.3 LINUX MINT

Git comes with Linux as well. Verify it runs by launching a terminal and running the git command. The stable release version will depend on your distribution. For Linux Mint the installed version of git is 2.34.1. To update to the latest version of git, launch a terminal and enter the following command:

```
sudo add-apt-repository ppa:git-core/ppa
```

When prompted, enter your password. When this command completes enter the following commands:

```
apt update
```

```
apt install git
```

When the install command completes, verify the latest version of `git` is working as expected. Note that these commands are listed on the Git-SCM Linux download page: <https://git-scm.com/download/linux>

QUICK REVIEW

In this book I'll be using Git for source code configuration management. Git comes with Git For Windows and with macOS and Linux. The version of Git installed with Git For Windows is fairly recent. Use `brew` to update Git to the latest version on macOS. Use `apt` to update Git on Linux.

2 CREATE A GITHUB ACCOUNT

If you haven't yet done so, procrastinate no longer. It's time for you to create your GitHub account. You'll need a GitHub account to create remote repositories. (*Technically, and in the interest of full disclosure, you can host remote Git repositories in a number of ways, but GitHub is the most popular site if you don't want the hassle of hosting your own Git remote repository on a dedicated server. Other Git remote repository sites include GitLab and SourceForge.*) I assume, going forward, you are using GitHub.

Navigate to <https://github.com> and in the upper right corner of the page click on the **Sign up** button as shown in figure 7-6.

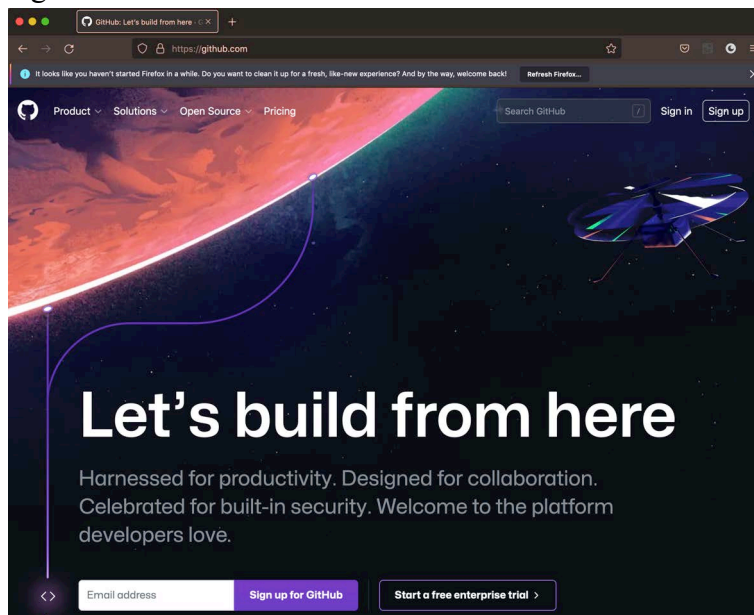


Figure 7-6: GitHub Home Page

Follow the prompts on the Sign Up page. I'm not going to hold your hand during this process, but when you've finished creating your account, activate multi-factor authentication for added security. GitHub has excellent guidance on how to do this. Also, be sure to save your account recovery keys in a safe, secure location. Individual accounts get free public and private repositories. I will go into detail on how to create repositories and securely connect to them using SSL in chapter 8.

QUICK REVIEW

If you haven't already done so, create your GitHub account. You'll need a GitHub account when you move on to *Chapter 8: Configuration Management with Git & GitHub*.

3 INSTALL PIPENV

Pipenv, <https://pipenv.pypa.io>, combines the power of Python package management and virtual environments into one convenient, easy-to-use tool. How I recommend you install Pipenv depends on whether you're running Windows or macOS/Linux.

3.1 WINDOWS

On Windows, launch a Git Bash terminal and run the following command:

```
pip install --user pipenv
```

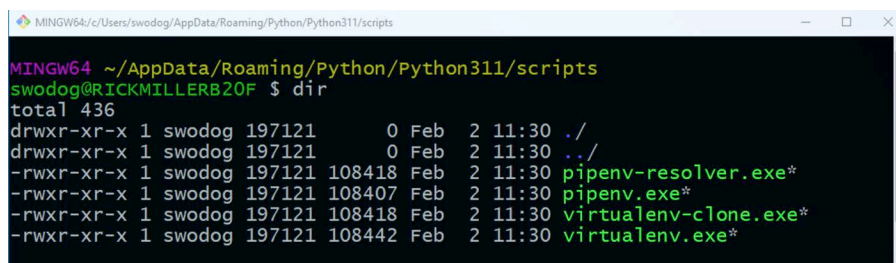
When installation completes, you may be prompted to upgrade pip as well. To do so, run the following command

```
python.exe -m pip install --upgrade pip
```

When this command completes, type pipenv at the command prompt. If you get a message stating the pipenv command was not found, you'll need to add its location to your system PATH.

3.2 LOCATING PIPENV INSTALLATION DIRECTORY

Pipenv's installation directory will depend on which version of Python you have installed. If you installed Python 3.11, you'll need to navigate to the `~/AppData/Roaming/Python/Python311/scripts` directory as shown in figure 7-7.



```

MINGW64~/Users/swodog/AppData/Roaming/Python/Python311/scripts
swodog@RICKMILLERB20F $ dir
total 436
drwxr-xr-x 1 swodog 197121      0 Feb  2 11:30 ./
drwxr-xr-x 1 swodog 197121      0 Feb  2 11:30 ../
-rwxr-xr-x 1 swodog 197121 108418 Feb  2 11:30 pipenv-resolver.exe*
-rwxr-xr-x 1 swodog 197121 108407 Feb  2 11:30 pipenv.exe*
-rwxr-xr-x 1 swodog 197121 108418 Feb  2 11:30 virtualenv-clone.exe*
-rwxr-xr-x 1 swodog 197121 108442 Feb  2 11:30 virtualenv.exe*

```

Figure 7-7: Pipenv Command Installation Location for Python 3.11 on Windows

Referring to figure 7-7 — Note the path to the scripts directory. I'm assuming you're using the Git Bash terminal so the path will have forward slashes and look like this: `~/AppData/Roaming/`

`Python/Python311/scripts`. You'll need to convert that path into a valid, absolute Windows path like so: `C:\Users\swodog\AppData\Roaming\Python\Python311\scripts`. Add this path to your User's Path environment variable as shown in figure 7-8 (See "What Is An Environment Variable?" on page 49. or watch this video: [Creating Environment Variables in Windows 10](#)).

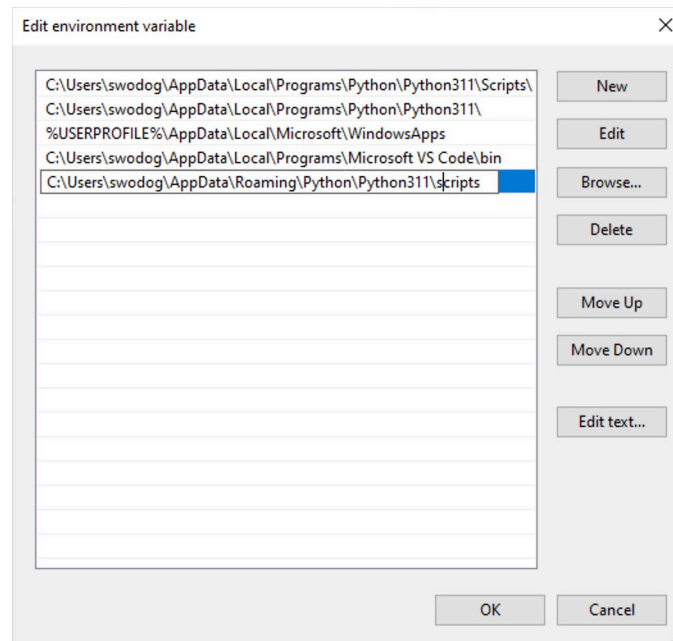


Figure 7-8: Add Path to pipenv Command to User's Path Environment Variable

Referring to figure 7-8 — Click the New button and paste the path to the `pipenv` command into the text box. Click OK, and OK again to close the Environment Variables dialog. Close and relaunch the Git Bash terminal. Type `pipenv` at the command prompt and you should now see a help listing as shown in figure 7-9.

```

MINGW64 ~/
swodog@RICKMILLERB20F $ pipenv
Usage: pipenv [OPTIONS] COMMAND [ARGS]...

Options:
  --where          Output project home information.
  --venv           Output virtualenv information.
  --py             Output Python interpreter information.
  --envs          Output Environment Variable options.
  --rm            Remove the virtualenv.
  --bare          Minimal output.
  --man           Display manpage.
  --support       Output diagnostic information for use in
                  GitHub issues.
  --site-packages / --no-site-packages
                  Enable site-packages for the virtualenv.
                  [env var: PIPENV_SITE_PACKAGES]
  --python TEXT   Specify which version of Python virtualenv
                  should use.
  --three         Use Python 3 when creating virtualenv.
  
```

Figure 7-9: The pipenv Command Is Now Working

3.3 MACOS & LINUX

There are a couple ways to install Pipenv on macOS and Linux. I recommend using the package managers (brew for macOS and apt for Linux) as they provide the latest stable release for the targeted distribution, however, the version of Pipenv they install may run behind the latest official version. For example, on the Python Package Index site, <https://pypi.org>, the latest version of Pipenv, as I write this, is version 2023.2.4. The Pipenv version on Linux Mint shows 2022.10.12 and on macOS, after installing with brew, shows 2022.12.19. So, all over the calendar so to speak.

3.3.1 MACOS

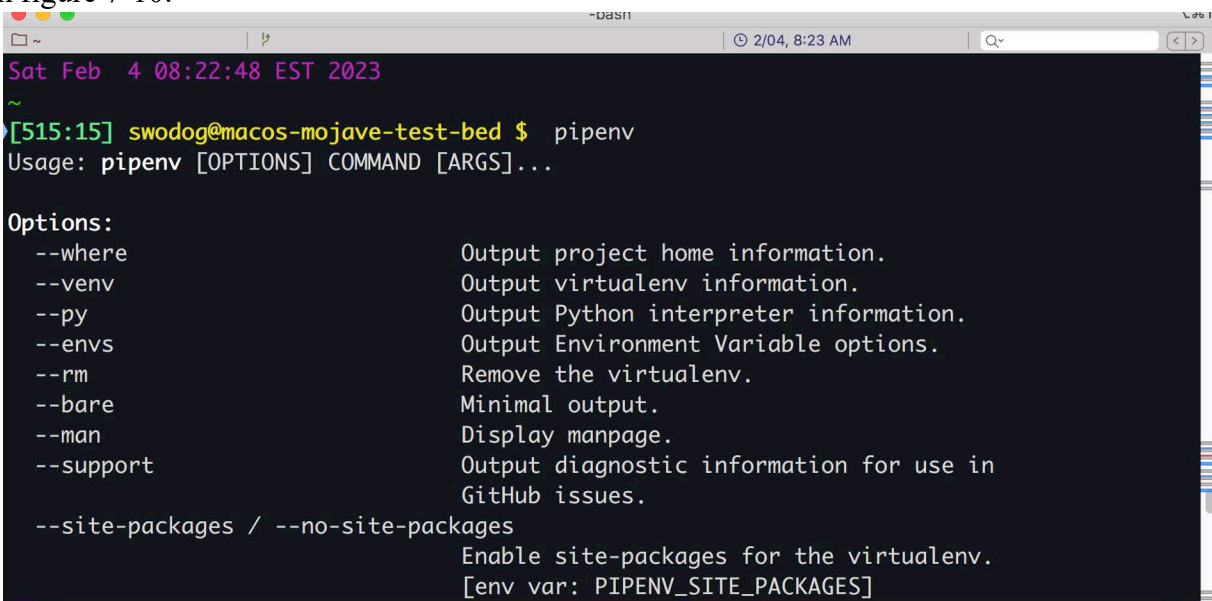
You'll need to install Pipenv on macOS. Like I said above, I recommend installing Pipenv using the brew package manager. Launch a terminal and type the following command:

```
brew install pipenv
```

When installation completes, verify installation by executing pipenv at the command prompt like so:

```
pipenv
```

Running pipenv without command-line arguments displays help text on the console as shown in figure 7-10:



```

Sat Feb  4 08:22:48 EST 2023
~
[515:15] swodog@macos-mojave-test-bed $ pipenv
Usage: pipenv [OPTIONS] COMMAND [ARGS]...

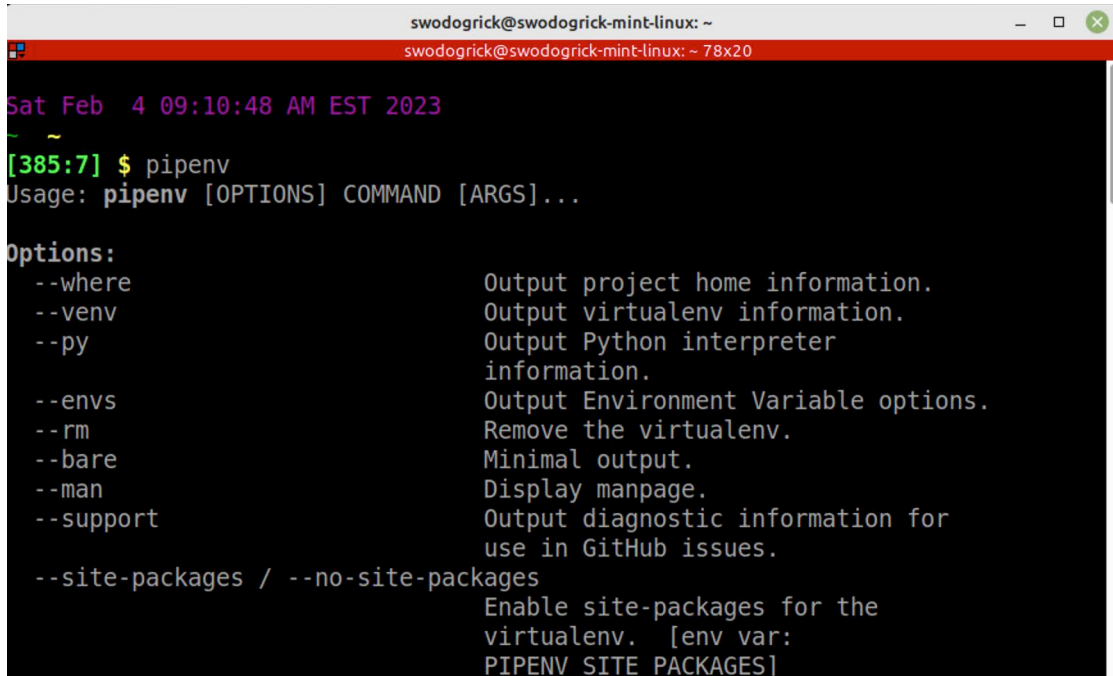
Options:
  --where                Output project home information.
  --venv                 Output virtualenv information.
  --py                   Output Python interpreter information.
  --envs                 Output Environment Variable options.
  --rm                   Remove the virtualenv.
  --bare                 Minimal output.
  --man                  Display manpage.
  --support              Output diagnostic information for use in
                        GitHub issues.
  --site-packages / --no-site-packages
                        Enable site-packages for the virtualenv.
                        [env var: PIPENV_SITE_PACKAGES]

```

Figure 7-10: Verifying pipenv Command macOS

3.3.2 LINUX

Pipenv comes installed on Linux Mint, so if that's what you're running you should be good to go. You can always verify its installation by launching a terminal and typing pipenv at the command prompt as shown in figure 7-11.



```

swodogrick@swodogrick-mint-linux: ~
swodogrick@swodogrick-mint-linux: ~ 78x20

Sat Feb  4 09:10:48 AM EST 2023

[385:7] $ pipenv
Usage: pipenv [OPTIONS] COMMAND [ARGS]...

Options:
  --where          Output project home information.
  --venv           Output virtualenv information.
  --py            Output Python interpreter
                  information.
  --envs          Output Environment Variable options.
  --rm            Remove the virtualenv.
  --bare          Minimal output.
  --man           Display manpage.
  --support       Output diagnostic information for
                  use in GitHub issues.
  --site-packages / --no-site-packages
                  Enable site-packages for the
                  virtualenv. [env var:
                  PIPENV_SITE_PACKAGES]

```

Figure 7-11: Verifying pipenv Command Linux Mint

3.4 SET PIPENV_VENV_IN_PROJECT ENVIRONMENT VARIABLE

Once you have Pipenv installed you'll need to configure an environment variable that instructs Pipenv to create the virtual environment, which includes the `.venv` directory and a *Pipfile*, in the project directory. You do this by creating an environment variable named `PIPENV_VENV_IN_PROJECT` and setting it to `"true"`. I'll start with Windows.

3.4.1 WINDOWS

Open the Environment Variables dialog and create a new environment variable as shown in figure 7-12.

Referring to figure 7-12 — Click OK to save the variable. You should now have a new environment variable as shown in figure 7-13.

Referring to figure 7-13 — Verify the new environment variable and its value are spelled correctly. If you did happen to make a mistake, you'll find out soon enough!

3.4.2 MACOS & LINUX

Linux and macOS users will need to export a shell environment variable by adding the following line to the `~/.bash_profile` file:

```
export PIPENV_VENV_IN_PROJECT="true"
```

Save the file and run the following command from the home directory to reload the `~/.bash_profile` file:

```
source ~/.bash_profile
```

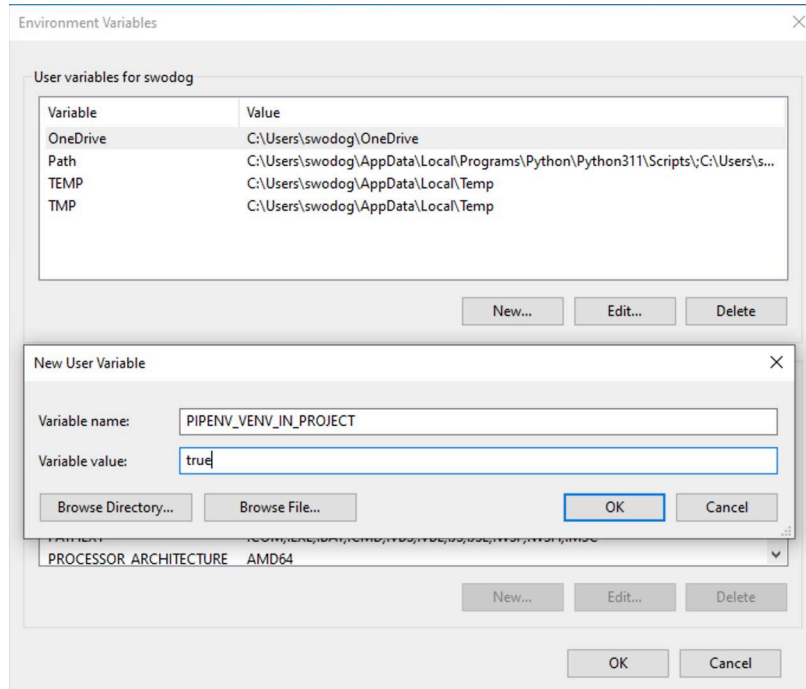


Figure 7-12: Create New User Variable Named PIPENV_VENV_IN_PROJECT and Set Value to true

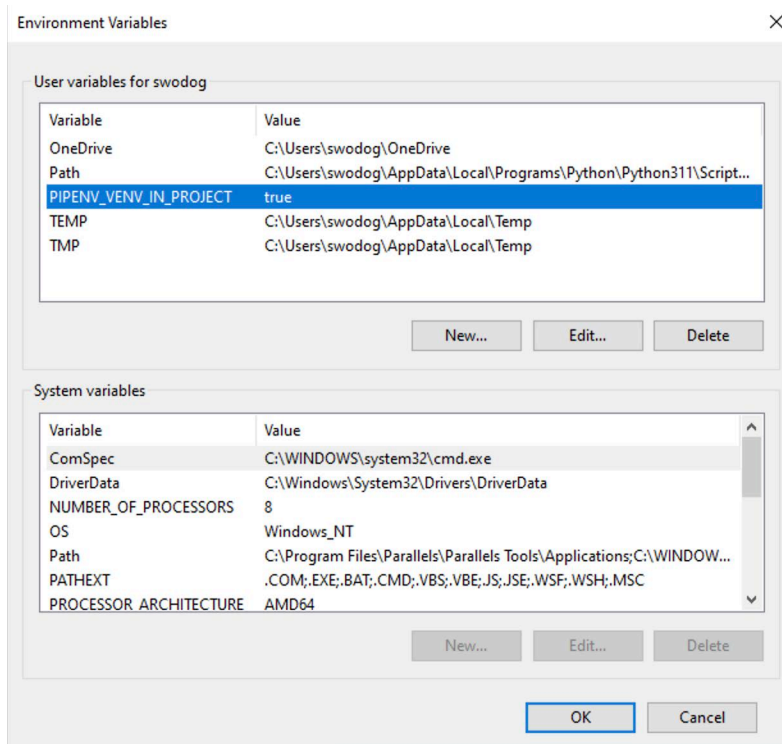


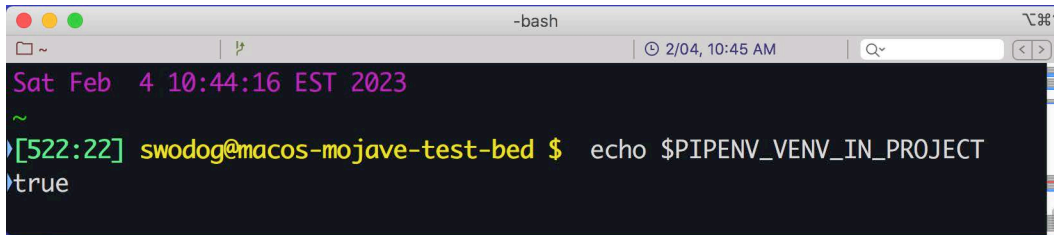
Figure 7-13: Verify New Environment Variable and its Value are Spelled Correctly

0
0
0
0
0
1
1
1

After you run this command, verify the environment variable is being exported by typing the following command:

```
echo $PIPENV_VENV_IN_PROJECT
```

It should print true on the next line as shown in figure 7-14.



```
-bash
Sat Feb  4 10:44:16 EST 2023
~
[522:22] swodog@macos-mojave-test-bed $ echo $PIPENV_VENV_IN_PROJECT
true
```

Figure 7-14: Verifying PIPENV_VENV_IN_PROJECT is Exported

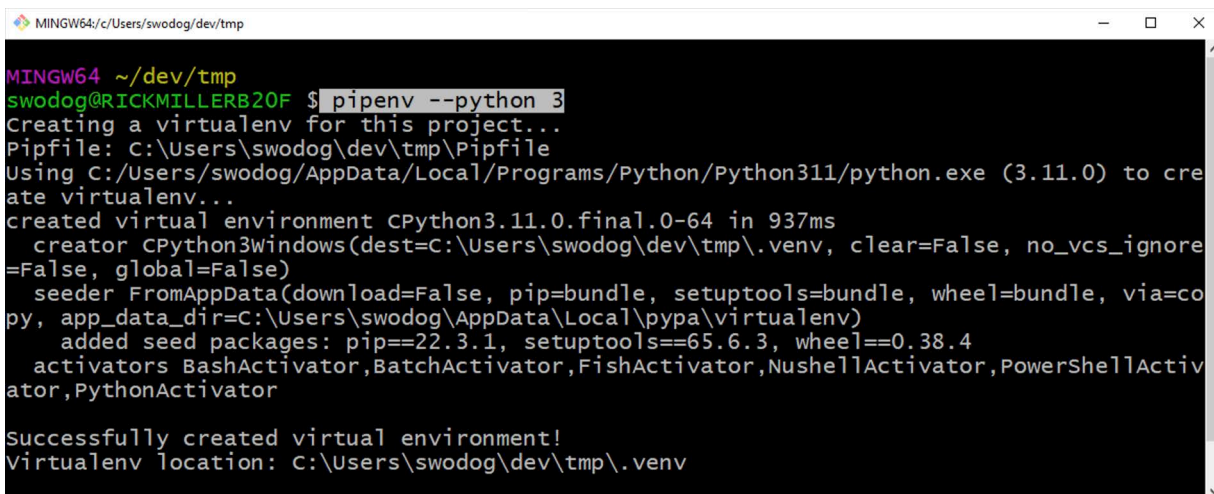
Referring to figure 7-14 — Note that the echo command will work with the Git Bash terminal in Windows as well. If you are running Windows, launch a Git Bash terminal and use the echo command to verify the environment variable is property set.

3.4.3 TESTING THE PIPENV_VENV_IN_PROJECT VARIABLE

OK, now for the real test. Launch a terminal, navigate to your `~/dev` folder and create a temporary (`tmp`) directory. Navigate to your newly-created `~/dev/tmp` directory and run the following command:

```
pipenv --python 3
```

Figure 7-15 shows the results of running this command:



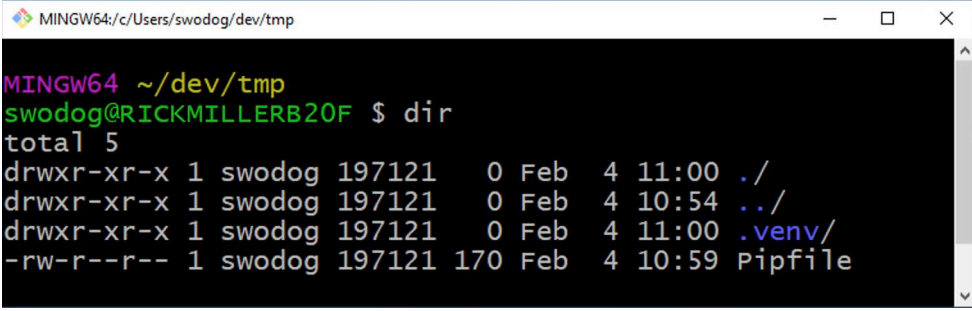
```
MINGW64~/dev/tmp
swodog@RICKMILLERB20F $ pipenv --python 3
Creating a virtualenv for this project...
Pipfile: C:\Users\swodog\dev\tmp\Pipfile
Using C:\Users\swodog\AppData\Local\Programs\Python\Python311\python.exe (3.11.0) to create virtualenv...
created virtual environment CPython3.11.0.final.0-64 in 937ms
  creator CPython3Windows(dest=C:\Users\swodog\dev\tmp\.venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\swodog\AppData\Local\pypa\virtualenv)
  added seed packages: pip==22.3.1, setuptools==65.6.3, wheel==0.38.4
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

Successfully created virtual environment!
Virtualenv location: C:\Users\swodog\dev\tmp\.venv
```

Figure 7-15: Creating New Python 3 Virtual Environment with pipenv Command

Referring to figure 7-15 — This will create a new virtual environment in the current (working) directory, which, if you're in the `~/dev/tmp` directory will be the `tmp` directory. Verify this by listing the contents of the directory. You should see a new directory named `.venv`, and a new file named `Pipfile` as is shown in figure 7-16

Referring to figure 7-17 — If you don't see the `.venv` or `Pipfile` then recheck the `PIPENV_VENV_IN_PROJECT` environment variable. Examine the output from running the `pipenv --`



```

MINGW64: c:/Users/swodog/dev/tmp
MINGW64 ~/dev/tmp
swodog@RICKMILLERB20F $ dir
total 5
drwxr-xr-x 1 swodog 197121  0 Feb  4 11:00 ./
drwxr-xr-x 1 swodog 197121  0 Feb  4 10:54 ../
drwxr-xr-x 1 swodog 197121  0 Feb  4 11:00 .venv/
-rw-r--r-- 1 swodog 197121 170 Feb  4 10:59 Pipfile

```

Figure 7-16: Verify pipenv Command Created Virtual Environment in Working (Project) Directory

python command (see the last line of figure 7-16) to see where it created the virtual environment. Regardless, to remove the virtual environment (.venv directory) just type the following command:

```
pipenv --rm
```

This will leave the Pipfile in place. You can delete it as well but you may need to track it down if the PIPENV_VENV_IN_PROJECT variable was improperly set. If everything works as expected then you're good to go.

QUICK REVIEW

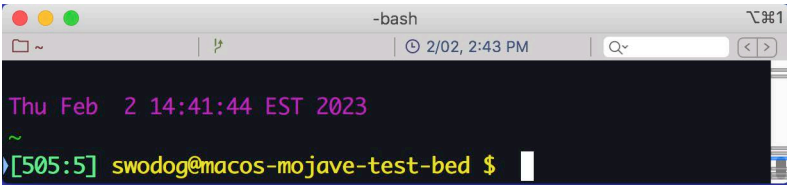
Pipenv is used to create Python virtual environments that enable development with different versions of Python on the same machine. It also enables deterministic builds by keeping track of installed Python package versions.

4 BASH ON MACOS

The latest version of the bash shell is 5.2.9 according to the GNU Bash page: <https://www.gnu.org/software/bash/>. The version of Bash that ships with macOS is a little dated because the default shell on newer versions of macOS is set to the Z Shell (zsh).

4.1 CHANGE DEFAULT SHELL TO BASH

First, launch a terminal window and check the title bar. It should show “-bash” as shown in figure 7-10.



```

-bash
Thu Feb  2 14:41:44 EST 2023
~
[505:5] swodog@macos-mojave-test-bed $

```

Figure 7-17: iTerm Terminal Running Bash Shell as Indicated by -bash in Window Title Bar

Referring to figure 7-10 — If instead you see “-zsh” then you need to change your default shell to Bash by typing the following command at the command prompt:

```
chsh -s /bin/bash
```

Close the terminal window and relaunch. It should read “-bash”. If not, take the following approach:

Step 1: Open System Preferences

Step 2: Select Users & Groups

Step 3: Unlock the settings by clicking the padlock and entering your password

*Step 4: Right-click your user name and select **Advanced Options...*** (See figure 7-11)

*Step 5: From the **Login shell:** dropdown select /bin/bash.* (See figure 7-12)

Step 6: Click OK, lock the settings, and close System Preferences

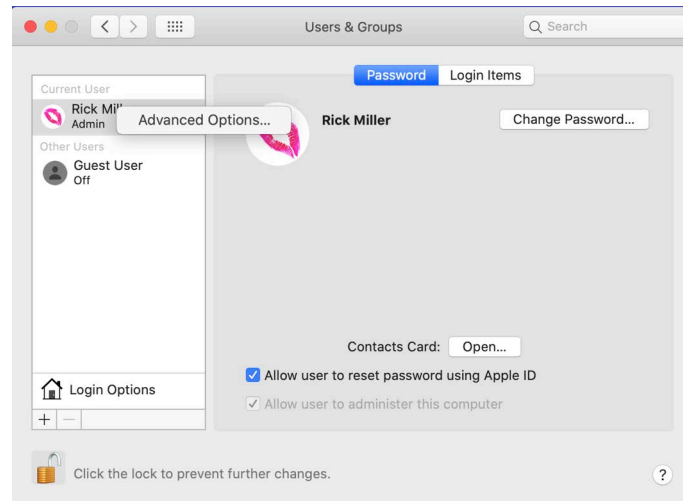


Figure 7-18: Right-Click Your User Name and Select Advanced Options...

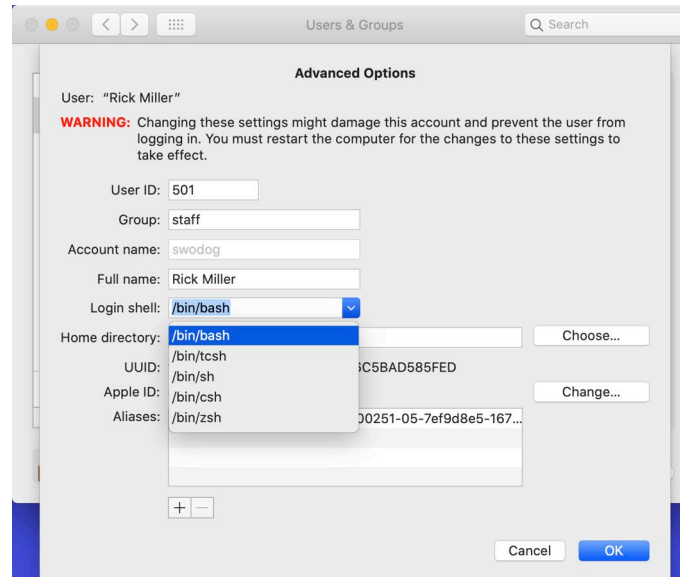


Figure 7-19: From Login Shell: Dropdown Select /bin/bash

Launch a fresh terminal window. You should now see “-bash” in the title bar.

4.2 UPDATE BASH

Now that you have set Bash as your default shell, update to the latest version using brew by typing the following command:

```
brew install bash
```

When this command completes, close and relaunch the terminal and check the Bash version by typing: `bash --version` — It should report version 5.2.9.

QUICK REVIEW

In macOS, you'll need to ensure your default shell is set to Bash to run the examples in this book. While not strictly necessary, it's a good idea to update to the latest version of Bash as the version that ships with macOS is somewhat dated.

SUMMARY

In this book I'll be using Git for source code configuration management. Git comes with Git For Windows and with macOS and Linux. The version of Git installed with Git For Windows is fairly recent. Use `brew` to update Git to the latest version on macOS. Use `apt` to update Git on Linux.

If you haven't already done so, create your GitHub account. You'll need a GitHub account when you move on to Chapter 8: Configuration Management with Git and GitHub.

`Pipenv` is used to create Python virtual environments that enable development with different versions of Python on the same machine. It also enables deterministic builds by keeping track of installed Python package versions.

In macOS, you'll need to ensure your default shell is set to Bash to run the examples in this book. While not strictly necessary, it's a good idea to update to the latest version of Bash as the version that ships with macOS is somewhat dated.

SKILL-BUILDING EXERCISES

None

SUGGESTED PROJECTS

None

SELF-TEST QUESTIONS

None

REFERENCES

GNU Bash Site, <https://www.gnu.org/software/bash/>

Pipenv Site, <https://pipenv.pypa.io/>

Git-SCM Site, <https://git-scm.com/>

GitHub Site, <https://github.com/>

NOTES

0
0
0
0
0
1
1
1