1
Developer Workflow

To fully take advantage of Drupal 8 it helps to have an established developer workflow. This can range from having a local web server to using a fully integrated virtualized AMP (Apache, MySQL, PHP) stack that mimics the development, staging, and production servers that one may need to deploy Drupal. It also helps to establish a set of best practices to follow when it comes to installing, managing, and working with Drupal 8 on a project to project basis. While there are various tools to choose from, we will look at some of the most common ones available to us, including **Composer**, **Drush**, **Drupal Console**, and **Git**.

The deeper we dive into mastering Drupal 8 the more vital it becomes to learn command line interfaces such as Composer, Drush, Git and Drupal Console. These require the use of a terminal window and a comfort level working with the command line. While we will not be covering the basics of Windows or Unix shell, we will explain what each command does and how each tool speeds up our developer workflow.

In this chapter, we will be covering the basics of how to use each of these tools to install, extend, and manage a typical Drupal 8 developer workflow, including:

- Deciding on a local AMP stack
- The role of Composer
- Speeding up tasks using Drush
- A quick look at Drupal Console
- Using Git to manage source code
- Virtualizing an environment

Deciding on a local AMP stack

Any developer workflow begins with having an AMP (Apache, MySQL, PHP) stack installed and configured on a Windows, OS X, or *nix based machine. Depending on the operating system, there are a lot of different methods that one can take to setup an ideal environment. However, when it comes down to choices there are only three:

- **Native AMP stack**: This option refers to systems that generally either come preconfigured with Apache, MySQL, and PHP or have a generally easy installation path to download and configure these three requirements. There are plenty of great tutorials on how to achieve this workflow but this does require familiarity with the operating system.
- **Packaged AMP stacks**: This option refers to third-party solutions such as MAMP--`https://www.mamp.info/en/`, WAMP--`http://www.wampserver.com/en/`, or Acquia Dev Desktop--`https://dev.acquia.com/downloads`. These solutions come with an installer that generally works on Windows and OS X and is a self-contained AMP stack allowing for general web server development. Out of these three only Acquia Dev Desktop is Drupal specific.
- **Virtual machine**: This option is often the best solution as it closely represents the actual development, staging, and production web servers. However, this can also be the most complex to initially setup and requires some knowledge of how to configure specific parts of the AMP stack. There are a few well documented VM's available that can help reduce the experience needed. Two great virtual machines to look at are Drupal VM--`https://www.drupalvm.com/` and **Vagrant Drupal Development** (**VDD**)--`https://www.drupal.org/project/vdd`.

In the end, my recommendation is to choose an environment that is flexible enough to quickly install, setup, and configure Drupal instances. The preceding choices are all good to start with, and by no means is any single solution a bad choice.

If you are a single person developer, then a packaged AMP stack such as MAMP may be the perfect choice. However, if you are in a team environment I would strongly recommend one of the previously mentioned VM options or consider creating your own VM environment that can be distributed to your team.

We will discuss virtualized environments in more detail, but before we do, we need to have a basic understanding of how to work with three very important command line interfaces. Composer, Drush, and Drupal Console.

The role of Composer

Drupal 8 and each minor version introduces new features and functionality. Everything from moving the most commonly used 3rd party modules into its core to the introduction of an object-oriented PHP framework. These improvements also introduced the **Symfony** framework which brings along the ability to use a dependency management tool called Composer.

Composer (`https://getcomposer.org/`) is a dependency manager for PHP that allows us to perform a multitude of tasks. Everything from creating a Drupal project to declaring libraries and even installing contributed modules just to name a few. The advantage to using Composer is that it allows us to quickly install and update dependencies by simply running a few commands. These configurations are then stored within a `composer.json` file that can be shared with other developers to quickly setup identical Drupal instances.

If you are new to Composer, then let's take a moment to discuss how to go about installing Composer for the first time within a local environment.

Installing Composer locally

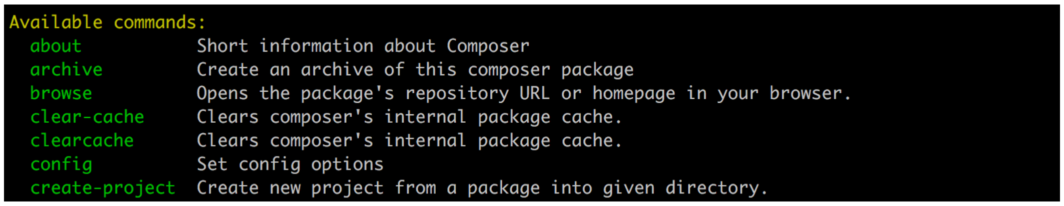
Composer can be installed on Windows, Linux, Unix, and OS X. For this example, we will be following the install found at `https://getcomposer.org/download/`. Make sure to look at the *Getting Started* documentation that corresponds with your operating system.

Begin by opening a new terminal window. By default, our terminal window should place us in the user directory. We can then continue by executing the following four commands:

1. Download Composer installer to local directory:
   ```
   php -r "copy('https://getcomposer.org/installer',
   'composer-setup.php');"
   ```
2. Verify the installer:
   ```
   php -r "if (hash_file('SHA384', 'composer-setup.php') ===
   '55d6ead61b29c7bdee5cccfb50076874187bd9f21f65d8991d46ec5cc9
   0518f447387fb9f76ebae1fbbacf329e583e30') { echo 'Installer
   verified'; } else { echo 'Installer corrupt';
   unlink('composer-setup.php'); } echo PHP_EOL;"
   ```

   Since Composer versions are often updated it is important to refer to these directions to ensure the hash file mentioned above is the most current one.

3. Run the installer:
   ```
   php composer-setup.php
   ```
4. Remove the installer:
   ```
   php -r "unlink('composer-setup.php');"
   ```
5. Composer is now installed locally and we can verify this by executing the following command within a terminal window:
   ```
   php composer.phar
   ```
6. Composer should now present us with a list of available commands:



```
Available commands:
  about           Short information about Composer
  archive         Create an archive of this composer package
  browse          Opens the package's repository URL or homepage in your browser.
  clear-cache     Clears composer's internal package cache.
  clearcache      Clears composer's internal package cache.
  config          Set config options
  create-project  Create new project from a package into given directory.
```

The challenge with having Composer installed locally is that it restricts us from using it outside the current user directory. In most cases, we will be creating projects outside of our user directory, so having the ability to globally use Composer quickly becomes a necessity.

Installing Composer globally

Moving the `composer.phar` file from its current location to a global directory can be achieved by executing the following command within a terminal window:

```
mv composer.phar /usr/local/bin/composer
```

We can now execute Composer commands globally by typing *composer* in the terminal window.

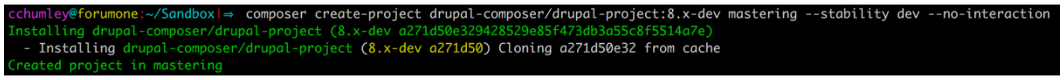
Using Composer to create a Drupal project

One of the most common uses for Composer is the ability to create a PHP project. The `create-project` command takes several arguments including the type of PHP project we want to build, the location of where we want to install the project, and optionally, the package version. Using this command, we no longer need to manually download Drupal and extract the contents into an install directory. We can speed up the entire process by using one simple command.

Begin by opening a terminal window and navigating to a folder where we want to install Drupal. Next we can use Composer to execute the following command:

```
composer create-project drupal-composer/drupal-project:8.x-dev
mastering --stability dev --no-interaction
```

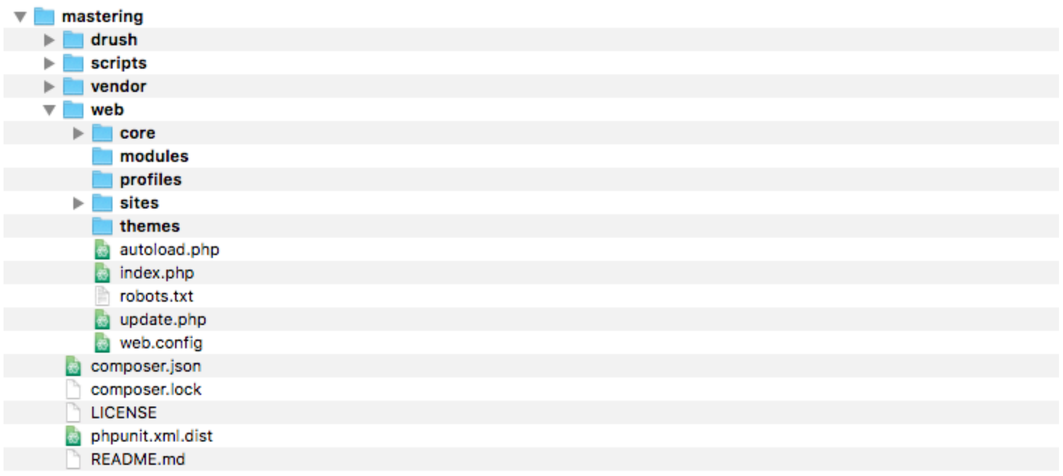
The `create-project` command tells Composer that we want to create a new Drupal project within a folder called `mastering`. We also tell Composer that we want the most stable development version. Once the command is executed, Composer locates the current version of Drupal and installs the project along with any additional dependencies that it needs:



```
cchumley@forumone:~/Sandbox|⇒  composer create-project drupal-composer/drupal-project:8.x-dev mastering --stability dev --no-interaction
Installing drupal-composer/drupal-project (8.x-dev a271d50e329428529e85f473db3a55c8f5514a7e)
  - Installing drupal-composer/drupal-project (8.x-dev a271d50) Cloning a271d50e32 from cache
Created project in mastering
```

The Composer project template provides a kick start for managing Drupal projects following best practice implementation. This includes installing both Drush and Drupal Console which are command line tools we can use to work with Drupal outside of the typical user interface. The reason Drush and Drupal console are packaged with the Composer project is to both avoid dependency issues and to allow for different versions of these tools per project. We will explore Drush and Drupal Console in greater detail a little later.

Composer also scaffolds a new directory structure that warrants taking a moment to review.



```
▼ mastering
    ▶ drush
    ▶ scripts
    ▶ vendor
    ▼ web
        ▶ core
          modules
          profiles
        ▶ sites
          themes
          autoload.php
          index.php
          robots.txt
          update.php
          web.config
      composer.json
      composer.lock
      LICENSE
      phpunit.xml.dist
      README.md
```

The new directory structure places everything related to Drupal within the `/web` folder, including **core**, **modules**, **profiles**, **sites** and **themes**. Drush and Drupal Console along with any dependencies that Drupal needs get installed within the `/vendor` folder. The remaining two folders `/drush` and `/scripts` are utilized by Drush and Drupal 8 to help configure our project.

All the installation, configuration and scaffolding that takes place is a result of the `composer.json` file that Composer uses to create our project. Often referred as a **package**, the `composer.json` file allows us to distribute it to other computers, web servers or team members to generate an identical Drupal 8 code base by simply executing `composer install`.

We will be using Composer to manage every aspect of a Drupal project. This will include the ability to update Drupal core when new versions are available, install and update Modules that we may want to use to extend Drupal and to add any additional configuration to manager installer paths and possibly patch modules. We will review these additional commands throughout the book.

For now, let’s switch our focus to some of the command line tools that were installed with our Drupal project beginning with Drush.

Speeding up tasks using Drush

Drush (`http://www.drush.org/en/master/`) is a command line shell and Unix scripting interface that allows us to interact with Drupal. Drush gives us the ability to use the command line to accomplish tasks quicker without the need of relying on the Drupal admin UI. As part of the composer install our project has the latest version of Drush installed automatically.

Executing a Drush command is typically as easy as typing the word `drush` within a terminal window.

However, the challenge of having a per project instance of Drush is in the way we are forced to currently execute Drush commands. Since the drush executable is located within the projects `/vendor/bin/drush` folder, if we are within the root of our project, we execute drush by entering the following within the terminal window:

```
./vendor/bin/drush
```

The problem is the path can easily change if for instance we are in the `/web` root the same command would be:

```
../vendor/bin/drush
```

Notice the two dots indicating I must traverse up a level to locate the `/vendor` folder.

This is not ideal when we will be using Drush quite frequently to perform various tasks. We can resolve this a couple different ways.

Using Drush wrapper

The first is to use `drush.wrapper` located within the `/vendor/drush/drush/examples` folder. This file is a wrapper script which launches Drush within a project. If we open the file within an editor, we will see that it states we need to copy the file to our `/web` folder and rename it to *drush*.

Choosing to follow this method would then allow us from within the `/web` folder to execute drush commands by entering the following within our terminal window:

```
./drush
```

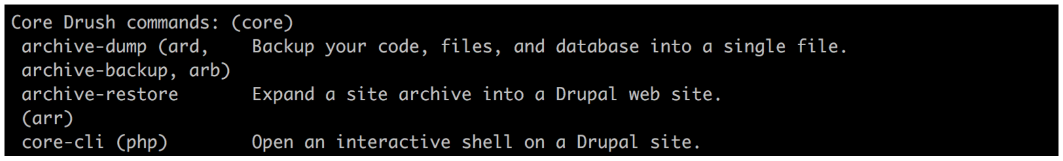
A little better however this is not quite as nice as simply typing the word `drush` without the need to know how to run a script. We can accomplish this by globally installing Drush using Composer.

Installing Drush globally

Installing Drush globally varies based on the operating system or AMP stack as there is a dependency of PHP 5.5.9 or higher. In most cases this dependency will be satisfied but make sure to verify the version of PHP that is available.

Begin by opening the terminal window, changing into the user directory, and executing the following commands:

1. Verify Composer is installed:
   ```
   composer
   ```
2. Add Composer's bin directory to the system path:
   ```
   export PATH="$HOME/.composer/vendor/bin:$PATH"
   ```
3. Install latest stable release:
   ```
   composer global require drush/drush
   ```
4. Verify that Drush works:
   ```
   drush status
   ```
5. Now that Drush has been installed globally we can easily make sure that we always have the latest version by running:
   ```
   composer global update
   ```
6. To get our first glance at the available commands that Drush provides we can execute the following:
   ```
   drush
   ```



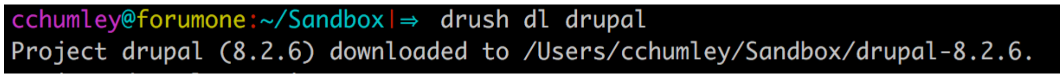
```
Core Drush commands: (core)
 archive-dump (ard,    Backup your code, files, and database into a single file.
 archive-backup, arb)
 archive-restore       Expand a site archive into a Drupal web site.
 (arr)
 core-cli (php)        Open an interactive shell on a Drupal site.
```

The list of Drush commands is quite long but it does provide us with the ability to perform almost any action we may need when working on a Drupal project. Some simple commands that we will commonly use throughout the book are clearing cache, managing configurations, and even installing Drupal. For a list of all the various commands we can browse to Drush commands `https://drushcommands.com/`.

Using Drush to create a Drupal project

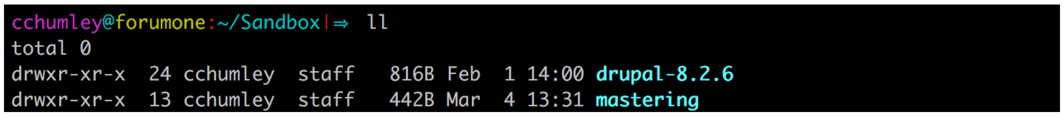
Some common uses of Drush is to download modules, themes and even Drupal itself. The command to execute this task is `drush dl`. Since we previously installed Drush globally we can change into a brand-new directory using the terminal window and download another copy of Drupal by executing the following command:

```
drush dl drupal
```



```
cchumley@forumone:~/Sandbox|⇒  drush dl drupal
Project drupal (8.2.6) downloaded to /Users/cchumley/Sandbox/drupal-8.2.6.
```

As we can see from the preceding screenshot, executing the command downloaded the current version of Drupal. We can verify this by listing the contents of the current directory:



```
cchumley@forumone:~/Sandbox|⇒  ll
total 0
drwxr-xr-x  24 cchumley  staff   816B Feb  1 14:00 drupal-8.2.6
drwxr-xr-x  13 cchumley  staff   442B Mar  4 13:31 mastering
```

Now that we have a second copy of Drupal, we can use Drush to perform a quick install:

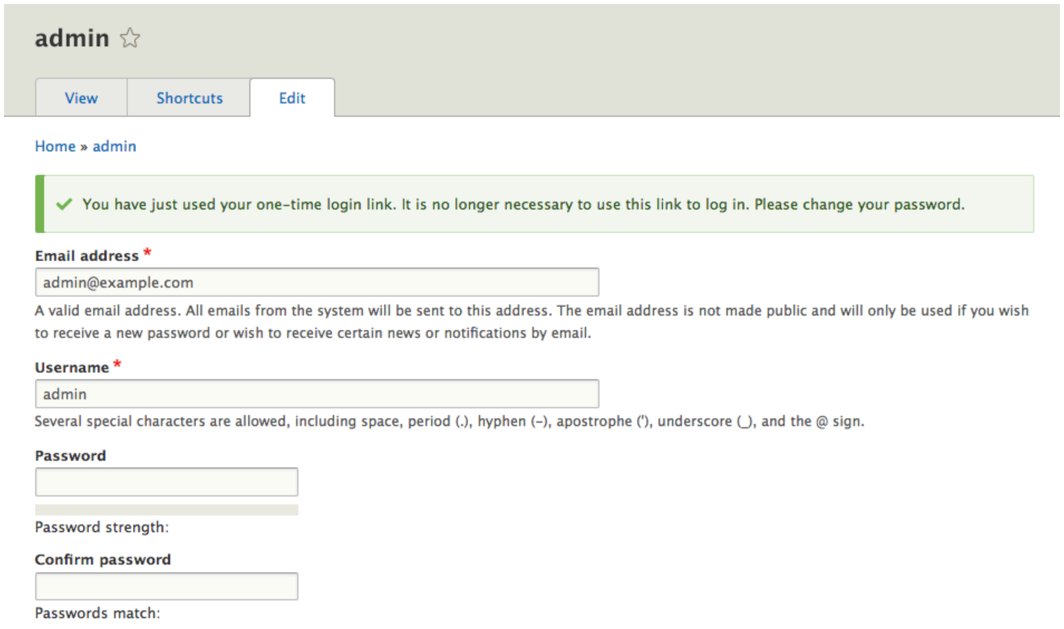
> Note that to use Drush to install Drupal without setting up or configuring an instance of an *AMP stack, we will need to at least have PHP 5.5.9 or higher installed.

Within a terminal window, change into the `drupal-8.x` directory that Drush downloaded and execute the following command:

```
drush qd --use-existing --uri=http://localhost:8383 --
profile=standard
```

This command tells Drush to perform a quick Drupal installation using the existing source files. Drupal will use the standard profile and once the installation has completed a PHP server will be started on localhost port `8383`.

Make sure to specify that we want to continue with the installation when prompted. Once the Drupal installation has finished, a browser window will open to the admin user page with the one-time login where we can then create a new password.


admin

You have just used your one-time login link. It is no longer necessary to use this link to log in. Please change your password.

Email address *
admin@example.com
A valid email address. All emails from the system will be sent to this address. The email address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by email.

Username *
admin
Several special characters are allowed, including space, period (.), hyphen (-), apostrophe ('), underscore (_), and the @ sign.

Password

Password strength:

Confirm password

Passwords match:

We will not be using this instance of Drupal so we can terminate the PHP server that is currently running in the terminal window by entering *Ctrl* + *C* on the keyboard.

Hopefully we can begin to see how using Drush can speed up common tasks. Throughout each lesson we will explore Drush in more detail and utilize additional commands. Now that we have a better understanding of Drush, it's time we look at another command line tool that we can benefit from using when developing a Drupal website.

A quick look at Drupal Console

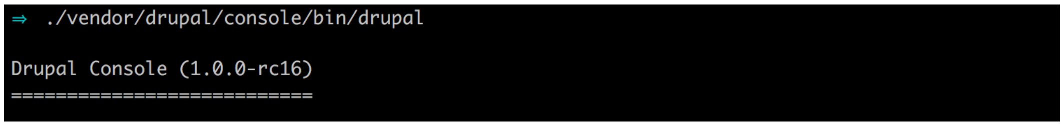
Drupal Console (`https://drupalconsole.com/`) is a new command line tool that has been welcomed by the Drupal community. Like Drush, but in my opinion much more powerful, Drupal Console allows us to perform site installs, manage configurations, create content, generate boilerplate code, and much more.

Accessing Drupal Console locally

As part of the original composer install of our Drupal project, Drupal console was installed. However just like accessing Drush locally we are faced with the same complexities of knowing the exact location of the Drupal console executable.

If we look within the `/vendor/drupal/console/bin` folder, we will see the executable that allows us to use Drupal console from the command line. We can enter the following command within the terminal to run the executable:

```
./vendor/drupal/console/bin/drupal
```



```
⇒  ./vendor/drupal/console/bin/drupal

Drupal Console (1.0.0-rc16)
===========================
```

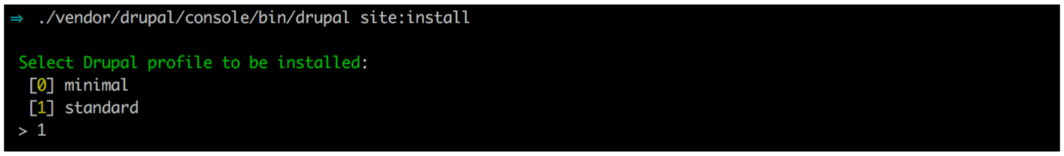
Installing Drupal using Drupal Console

We should all be familiar with the typical install process of Drupal. Download files, create database, setup a localhost, open a browser, and finish the installation. As we all know, this is a necessary evil but also a time-consuming task. Since we now have Drupal Console installed, we can achieve all of this by executing one single command.

Begin by opening a terminal window, changing into the `mastering` folder, and executing the following command:

```
./vendor/drupal/console/bin/drupal site:install
```

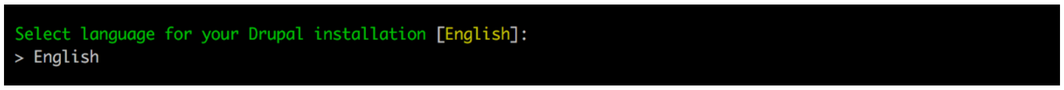
This command will begin a series of prompts that will walk us through the remaining install process beginning with choosing an install profile:



```
⇒  ./vendor/drupal/console/bin/drupal site:install

 Select Drupal profile to be installed:
  [0] minimal
  [1] standard
 > 1
```

Select the **Standard** install which is option **1** and press *Enter*.

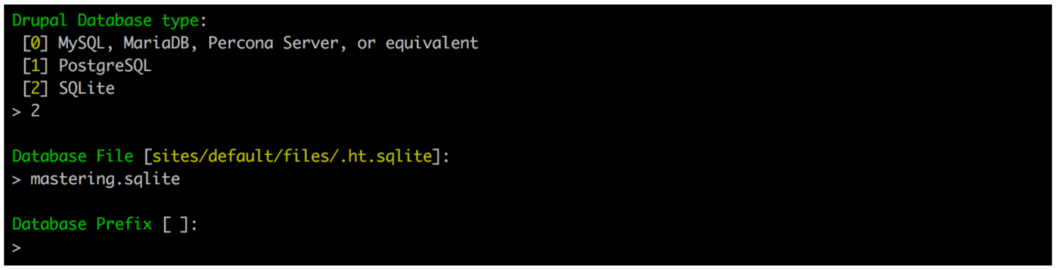
We will then be prompted to select a language that we want Drupal installed in:



```
Select language for your Drupal installation [English]:
> English
```

Input *English* and then press *Enter*.

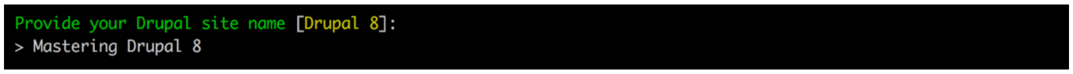
Next we will be prompted to choose the **Drupal Database type**, **Database File**, and **Database Prefix** that Drupal will use for the necessary database and tables. For sake of demonstration, we will let Drupal Console create a **SQLite** database:



```
Drupal Database type:
 [0] MySQL, MariaDB, Percona Server, or equivalent
 [1] PostgreSQL
 [2] SQLite
> 2

Database File [sites/default/files/.ht.sqlite]:
> mastering.sqlite

Database Prefix [ ]:
>
```

Select option *2* and then press *Enter*. Next we will enter a value of *mastering.sqlite* as the default name for the **Database File** and leave the default for the **Database Prefix**.

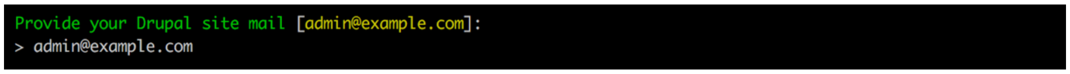
At this point we will be prompted to provide the site name for our Drupal instance:



```
Provide your Drupal site name [Drupal 8]:
> Mastering Drupal 8
```

Input the site name as *Mastering Drupal 8* and then press *Enter*.

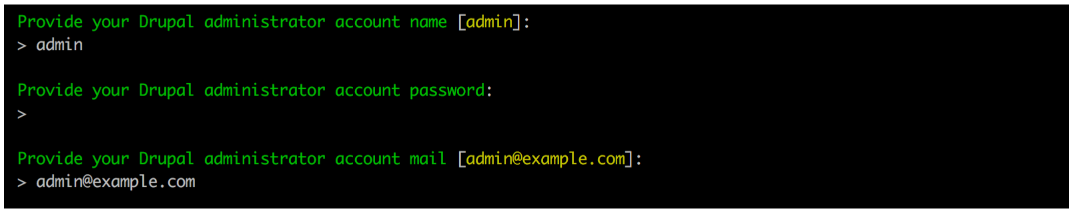
Drupal Console now requires us to provide a site e-mail that will be used to notify us of any updates, users that request an account, and various other administrative notifications:



```
Provide your Drupal site mail [admin@example.com]:
> admin@example.com
```

Input the e-mail as *admin@example.com* and then press *Enter*.

The next three values which we will need to provide will be for our administrators account and consist of the admin account name, e-mail, and password:



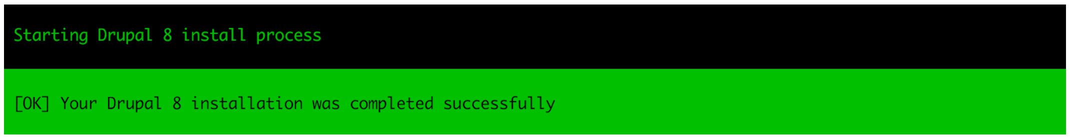
```
Provide your Drupal administrator account name [admin]:
> admin

Provide your Drupal administrator account password:
>

Provide your Drupal administrator account mail [admin@example.com]:
> admin@example.com
```

We will input *admin* for our administrator account name and then press *Enter*.

Next we will add a generic administrator account e-mail of *admin@example.com* and then press *Enter*.

Finally, we will input an administrator account password of *admin* and then press *Enter*.

At this point, Drupal Console will begin the install process and configure our new Drupal 8 instance. If everything is successful, we will be prompted with a notification that the Drupal 8 installation was completed successfully:



```
Starting Drupal 8 install process

[OK] Your Drupal 8 installation was completed successfully
```

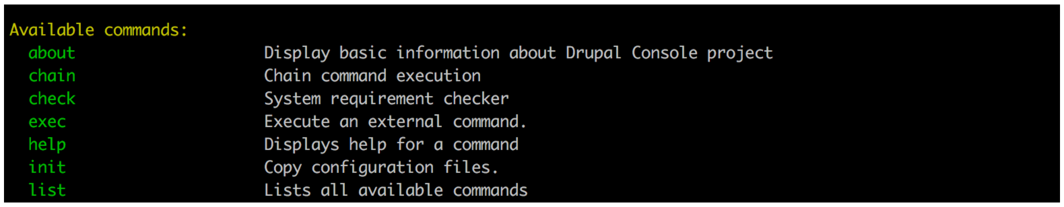
Now that Drupal 8 is installed and configured it would be nice to not have to always type the full path to Drupal Console the next time we want to use it. We can shorten this up to just entering `drupal` by installing Drupal console globally like we did for Drush.

Installing Drupal Console globally

Having global access to Drupal Console will allow us to execute commands regardless of our location within a project by simply typing `drupal`.

Begin by opening the terminal window, changing into our user directory, and executing the following commands:

1. Install Drupal Console Launcher:
   ```
   curl https://drupalconsole.com/installer -L -o drupal.phar
   mv drupal.phar /usr/local/bin/drupal
   chmod +x /usr/local/bin/drupal
   ```
2. Update Drupal Console Launcher:
   ```
   drupal self-update
   ```
3. Run Drupal Console to list all commands:
   ```
   drupal list
   ```



```
Available commands:
  about                      Display basic information about Drupal Console project
  chain                      Chain command execution
  check                      System requirement checker
  exec                       Execute an external command.
  help                       Displays help for a command
  init                       Copy configuration files.
  list                       Lists all available commands
```

One thing to note is that to see a full list of the available commands we will need to run `drupal list` inside the root directory of a Drupal project.

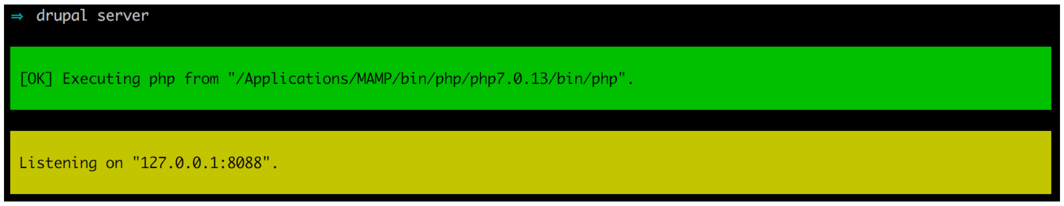
Running a built-in PHP web server

Another advantage of using Drupal Console within our project is that we can utilize the built-in PHP web server to display our new Drupal 8 site. If we look at the available commands listed by Drupal Console, we will notice a command called `server`.

Open a terminal window and enter the following command:

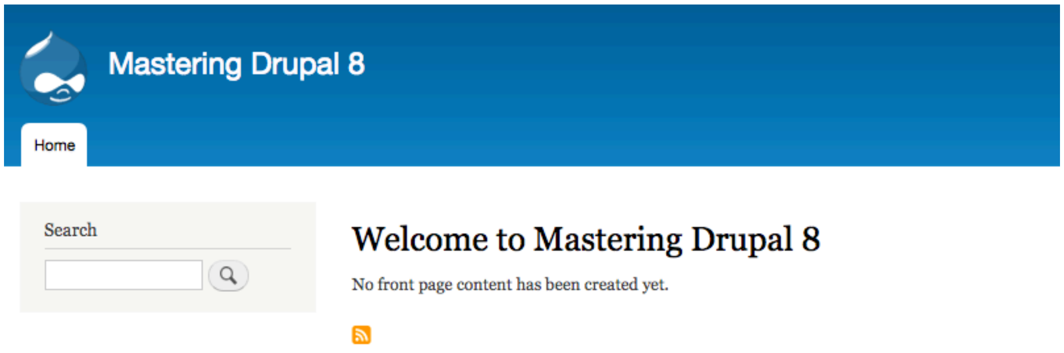
```
drupal server
```

Drupal Console can utilize the current version of PHP installed on our system. It identifies the document root of our Drupal installation and allows us to preview our site within the browser by navigating to `http://127.0.0.1:8088`.



```
⇒ drupal server

[OK] Executing php from "/Applications/MAMP/bin/php/php7.0.13/bin/php".

Listening on "127.0.0.1:8088".
```

If we open a browser and enter the url of `http://127.0.0.1:8088` we will be taken to our new Drupal 8 instance.



Welcome to Mastering Drupal 8

No front page content has been created yet.

The advantages of using Drupal Console to execute a wide range of commands, including installing Drupal, is a huge time saver. As we dig deeper into Mastering Drupal 8 we will discover additional commands that will allow us to manage users, roles and content.

So far we have looked at Composer, Drush, and Drupal Console. However, all of this is of no benefit to us if we have no way to ensure our work is protected and can be shared with other developers. In fact, managing source code is the most important tool any development workflow should embrace.

Using Git to manage source code

Git (https://git-scm.com) is probably the most popular open-source software available to manage source code. Git allows us to distribute code to ourselves or other developers and provides a robust mechanism for tracking changes, creating branches, and staging changes to software, or in our case, web-projects.

While we will not be diving deep into all the great flexibility that this tool provides, it is important that we touch on the basics of how to use Git within a development workflow.

Generally, there are a handful of tasks that we will perform with Git:

1. Creating a repository to store our code.
2. Adding code to our repository.
3. Tracking changes to our code.
4. Pulling and pushing changes.

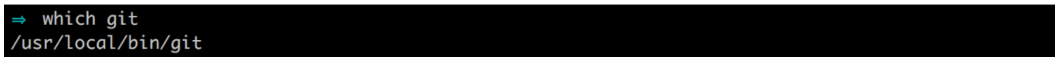
Installing Git

Git can be installed using a variety of methods including browsing to the Git website at https://git-scm.com/downloads and downloading the latest release suitable for your operating system.

For sake of demonstration, we will be installing Git on Mac OS X. Once we click on the appropriate link our download will start and the binary files will be copied to our designated downloads folder. All that is left to do is to extract the files and then double-click on the installer to complete the installation process.

We can validate that Git has been installed correctly by opening a terminal window and executing the following command:

```
which git
```



```
⇒  which git
/usr/local/bin/git
```

If at any point there is a need to refer to the Git documentation, we can browse to https://git-scm.com/doc. The documentation covers everything from the basics to advanced topics.

Assuming we have Git installed properly, we will need to configure it for use.

Configuring Git

Git can be configured locally per project or globally. In most cases, we will want to globally configure Git for use with all our projects. We are only concerned with a few configurations to begin with. Mainly our `user.name` and `user.email` which is used for associating our user with commit messages when tracking code.

Begin by opening a terminal window and executing the following commands:

```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

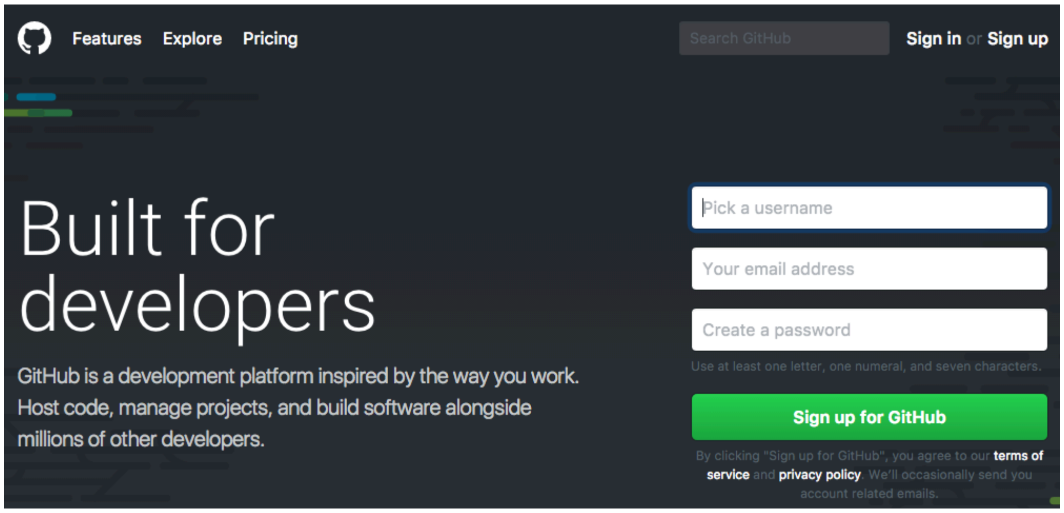
If we ever need to view what our configuration contains we can execute the following command:

```
git config --list
```

Now that we have Git installed and configured we will need to decide where we want to store our code.

Creating a remote repository

While we can create a local repository, it would make more sense to create a remote repository. When someone mentions Git it is generally synonymous with GitHub `https://github.com/`. To use GitHub, we will need to sign up for a free account or login to an existing account.

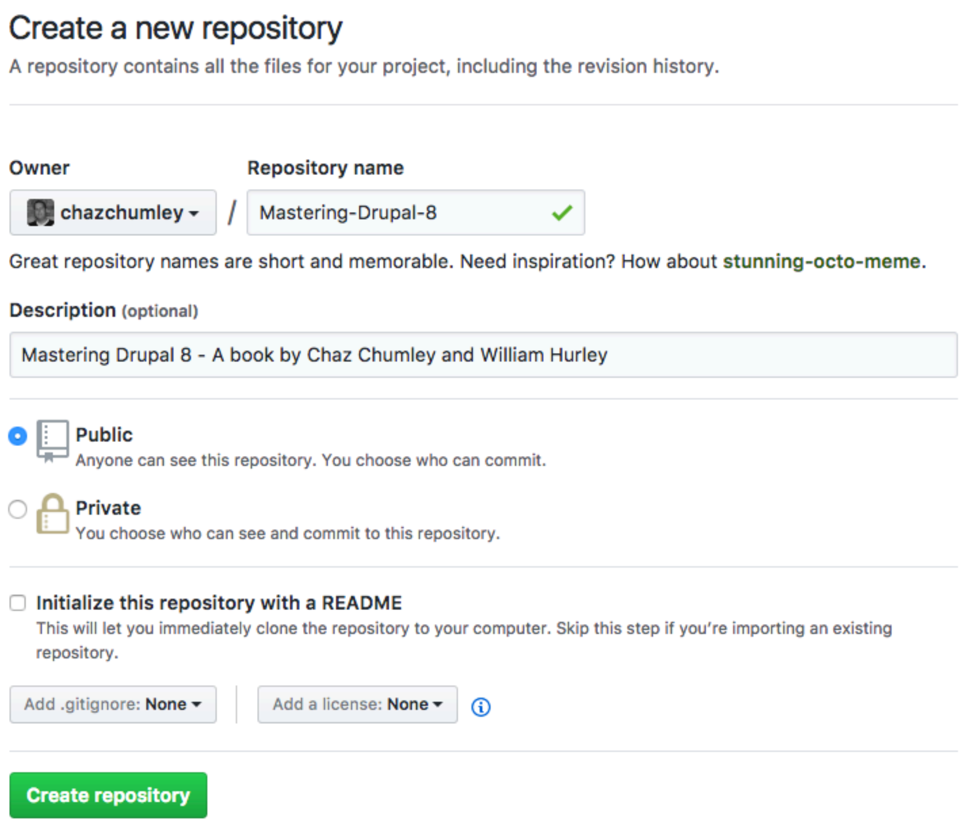


GitHub is a development platform inspired by the way you work. Host code, manage projects, and build software alongside millions of other developers.

Use at least one letter, one numeral, and seven characters.

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We'll occasionally send you account related emails.

Once logged into GitHub we will create a new empty repository. For the sake of demonstration, we will call our repository *Mastering-Drupal-8*.


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: chazchumley / Repository name: Mastering-Drupal-8

Great repository names are short and memorable. Need inspiration? How about stunning-octo-meme.

Description (optional): Mastering Drupal 8 - A book by Chaz Chumley and William Hurley

[✓ Public] Anyone can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

☐ Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None

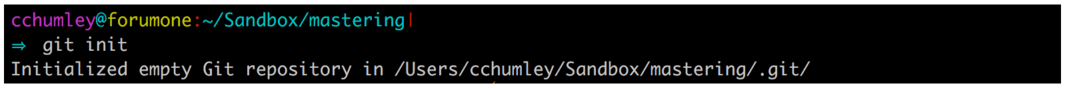
In the example above the Owner field would be replaced with your account name and Repository name based on your preferences. At this point we can click on the **Create repository** button to finish the creation of our Remote repository. Next we will create a local repository and push our local file up to GitHub.

Setting up a local repository

To start a local repository, we need to make sure we are within the folder that contains the files we want to begin tracking. Instantiating a local repository allows us to add files, commit them and push them up to the remote repository that others can clone and work from. For our example, we will add the Drupal 8 instance we just created.

Begin by opening a terminal window and entering the following command:

```
git init
```



```
cchumley@forumone:~/Sandbox/mastering|
⇒ git init
Initialized empty Git repository in /Users/cchumley/Sandbox/mastering/.git/
```

Tracking and committing files with Git

Now that we have initialized our `mastering` folder to be a local repository, we can add the contents of the folder to Git for tracking any changes. Adding and Committing files requires two steps:

First is adding the entire contents of the folder or specific files. In our example, we can add the entire Drupal instance by typing the following command in the terminal window:

```
git add .
```

Second we need to tell Git what we have added by committing the files and including a message describing what the addition contains. This can be accomplished by entering the following command in the terminal window:

```
git commit -m 'Initial Drupal instance added to repo'
```

Adding a remote origin

With our files added and committed locally we now need to add a remote origin that our local repository can push to. We can execute the following command in a terminal window remembering to replace the origin url with your own repo path:

```
git remote add origin https://github.com/chazchumley/Mastering-
Drupal-8.git
```

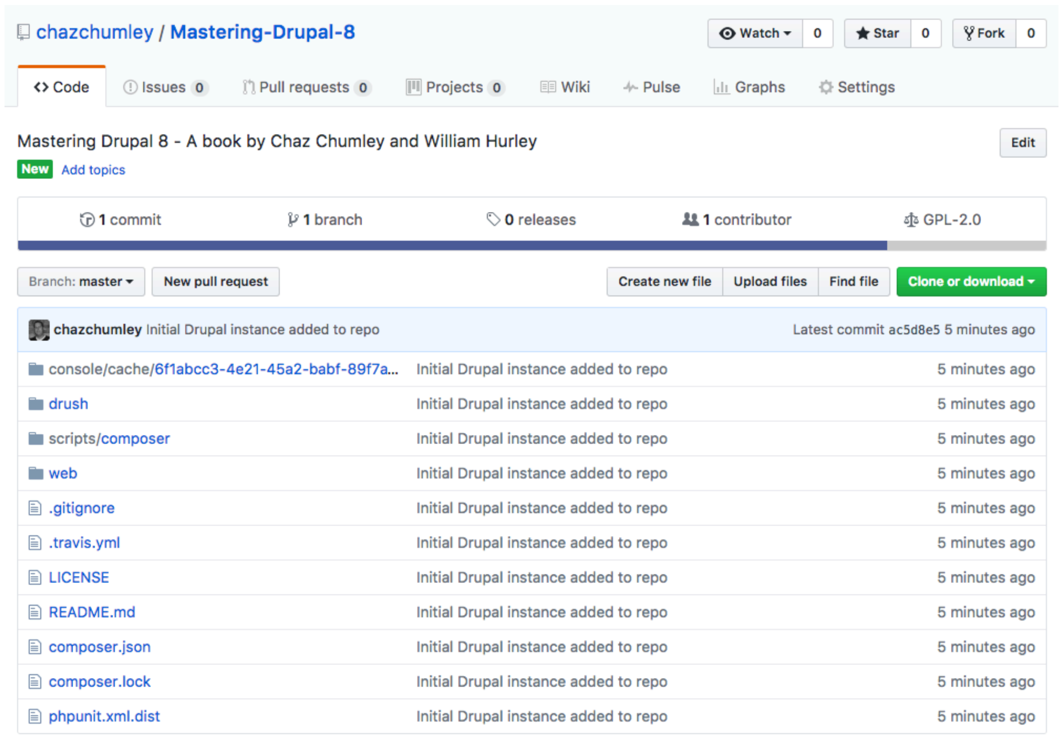
To find the correct origin url simply look at the url within the browser after the remote repo was created.

Pushing files to remote repository

Now that our local repository knows that we have a remote repository we can simply push the committed files to GitHub by executing the following command in a terminal window:

```
git push -u origin master
```

If we navigate to GitHub we will now see that our once empty repo contains the Drupal 8 instance that we locally added.


chazchumley / Mastering-Drupal-8

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

Mastering Drupal 8 - A book by Chaz Chumley and William Hurley

Edit

New Add topics

1 commit | 1 branch | 0 releases | 1 contributor | GPL-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

chazchumley Initial Drupal instance added to repo — Latest commit ac5d8e5 5 minutes ago

console/cache/6f1abcc3-4e21-45a2-babf-89f7a...	Initial Drupal instance added to repo	5 minutes ago
drush	Initial Drupal instance added to repo	5 minutes ago
scripts/composer	Initial Drupal instance added to repo	5 minutes ago
web	Initial Drupal instance added to repo	5 minutes ago
.gitignore	Initial Drupal instance added to repo	5 minutes ago
.travis.yml	Initial Drupal instance added to repo	5 minutes ago
LICENSE	Initial Drupal instance added to repo	5 minutes ago
README.md	Initial Drupal instance added to repo	5 minutes ago
composer.json	Initial Drupal instance added to repo	5 minutes ago
composer.lock	Initial Drupal instance added to repo	5 minutes ago
phpunit.xml.dist	Initial Drupal instance added to repo	5 minutes ago

With our files now safely being tracked both locally and remotely, we can ensure that any change we make can be safely retrieved and reverted. Think of this as a snapshot of our code. If we are working in a team environment, we can share the repo with others to clone the repo to their local machines. There is a lot of great documentation on how to manage Drupal workflows using Git at `https://www.drupal.org/node/803746`.

Realize that these are the very basics of using Git and depending on the size of your development team there are additional strategies that may need to be implemented.

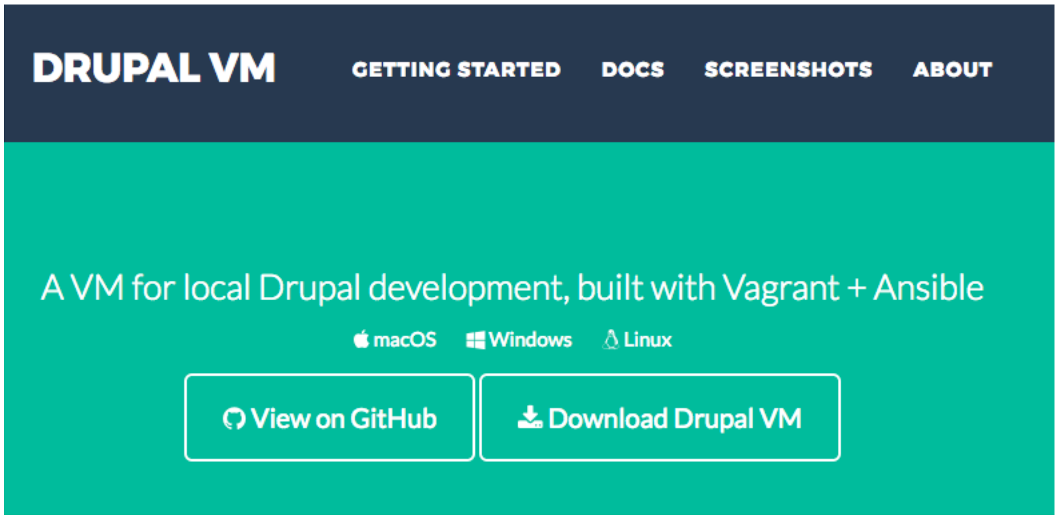
At this point you may be thinking that there is a lot of information to remember when installing and configuring a Drupal project. While you may be right, it is also the reason why **Virtualizing** a development environment makes perfect sense.

Virtualizing an environment

When we first began with this lesson, we mentioned the various ways to setup a local development environment. Depending on the size of your organization or team, having a repeatable and configured method to starting each Drupal project cuts down on having to manually install tools and dependencies. A virtual machine also eliminates issues with something working locally but not working on a remote server.

Using Drupal VM as a web starter

Depending on your skill level, it may make sense to create your own web starter by packaging and configuring Vagrant, VirtualBox, PHP, MySQL, and the list goes on. But if some of those terms seem foreign to you then I would recommend the well documented and easy to use Drupal VM--`http://www.drupalvm.com`.



Drupal VM is a virtual machine for local Drupal development built with Vagrant and Ansible that can be installed and ran on Mac, Windows, or Linux. This package allows for a consistent development experience with a robust set of tools already installed including Composer, Drush, and Drupal Console.

Drupal VM was created and is maintained by Jeff Geerling. It is by far the best option to work with a Drupal based web project and while the steps involved to install it are clearly spelled out in the documentation, I would recommend starting with the *Quick Start Guide* `https://github.com/geerlingguy/drupal-vm#quick-start-guide`.

It is worth taking the time to learn how to work with a virtualized environment that can be configured and customized to work with any requirements your next Drupal project may require.

If at any point you experience any issues, the Drupal VM repository issue queue--`https://github.com/geerlingguy/drupal-vm/issues` is available to post questions to for assistance.

Summary

As we progress through each lesson of Mastering Drupal 8 it is important that a local development environment has been setup with the basic tools we covered in this lesson. That includes an instance of Drupal 8 with Composer, Drush, and Drupal Console. It should also be clear by now how each tool can expedite tasks that would manually take a lot longer to perform. These tools should be part of every Drupal project as they will help you master your skills as a developer.

In the next chapter, we will begin to walk through Drupal 8 site configuration including changes to the administrative interface, how to manage regional settings, site information, and performance while developing.