Installing & Using Composer Tutorial For Dummies

https://www.codewall.co.uk/

@codewallblog

Just in-case you're not aware of Composer is, it's simply a third-party package managing tool. It brings streamlined efficiency to adding new libraries to your projects by writing simple one-line commands.

Contents

Installation

Prerequisites

Before you follow any of the guides below, you must ensure that PHP is already installed on your computer. Composer will not run properly without it and you will soon run into problems. In addition, you must set PHP as an environmental variable within your system for shortcut command access to the engine. See the following steps to do it on Windows.

1. Go to System Properties
2. Go to the advanced tab
3. Click Environment Variables.
4. If not already present, create a new variable named PATH. If already present, go to the next step.
5. Create a new variable value within the PATH variable, this value needs to be the path to your PHP folder, where the php.exe is located. In my case it's C:\xampp\php
6. After adding this value, press the OK button till all the windows are closed.

Installing Composer Software on Windows

Note: If you don't have sufficient rights on your computer to install any software, please see the next section, installing on Mac & others. As this is a workaround for Windows users with this issue.

The easiest way to install Composer on windows is by downloading the native Composer-Setup.exe. This installs the required files to your local computer that are ready to be utilized within your projects. See the following steps and sub-steps -

Step 1

a. Download the Composer-Setup.exe file from the download page here. (https://getcomposer.org/download/)
b. Install the software.

Step 2

a. Close all file-explorers and command windows.
b. Open a new command window and type `composer`
c. You should see composer should load its initial home output, if not, see the paragraph below.
d. If the prompt complained about PHP not being recognized, then see the prerequisites of this tutorial.

If you got the response of 'composer is not recognized as an internal or external command', then you will need to set the composer bin path in your environmental variables. On my device, this bin folder is located in 'C:\Users\CodeWall\AppData\Roaming\Composer\vendor\bin'. This is the value you will need to add to your PATH variable.

Installing Composer Manually on Mac, Linux & Windows

Composer can be installed via a PHP Archive file, namely composer.phar. This file is available from download page (https://getcomposer.org/download/), underneath the Manual Download sub-heading. Opt for the latest version, at the time of writing it is 1.8.0. After the composer.phar file is downloaded,

Step 1

a. Locate the composer.phar on your device and drag/move it to the root directory of your PHP project folder.
b. Now open your command prompt and navigate to the projects root directory, for example, on mac it would be `cd ~/phpProject` or on Windows `cd C:/phpProject`
c. Once here type the following command `php composer.phar` and composer should load in the terminal. If it doesn't and the terminal complains about PHP not being a recognized command, then see the prerequisite of this tutorial.

That's it with the PHP archived package, it's now installed and ready to be utilised as and when, without physically installing to your system.

Using Composer

Before you can utilize the power of Composer, see the following checklist is complete

- Composer is installed, if not, see Installation section.
- Both PATH variables for PHP and Composer so that they are accessible via shortcut commands in the terminal, if not, see the installation section.

Throughout these remaining examples, the demonstrations will be carried out with a Windows based system using its native command prompt. Some commands may differ on your operating system but will be easy enough to find in the operating systems documentation.

Initializing Composer with your project

Composer needs a kind-of-settings file to run properly, and ultimately for you to instruct it on what it should do. The composer.json file doesn't come fresh-out-the-box so one must be created. Luckily, the composer engine helps with this task, literally walking you through each step of creation. See the following steps to build the .json file.

Step 1

a. Open the command prompt.
b. Go to your PHP project folder, for this example, the following command was used - `cd C:\project`, make sure you replace the path in the example command with your project path.
c. Then run the following command - `composer init`
d. Type in a Package Name, something descriptive of your project and hit enter.
e. Add a small description of your project and hit enter.
f. For the Minimum Stability input, it depends on the project at hand, you can see further details in the docs (https://getcomposer.org/doc/04-schema.md#minimum-stability) but for this example, just press enter, and it defaults to 'stable'.
g. For the Package Type, it can be either of the following, Package, Library, metapackage or composer-plugin. Pick what is relevant, but in this case, project was entered.
h. License is another personal circumstance, if there is going to be a license, then you will need to specify it, if not, simply hit enter.
i. The next line will ask you if you want to define your dependencies, at this moment, type no and press enter, and repeat the same for the dev-dependencies.
j. Finally, type yes to confirm your settings and Composer to generate the file. See Figure 1 for an example of all these steps.

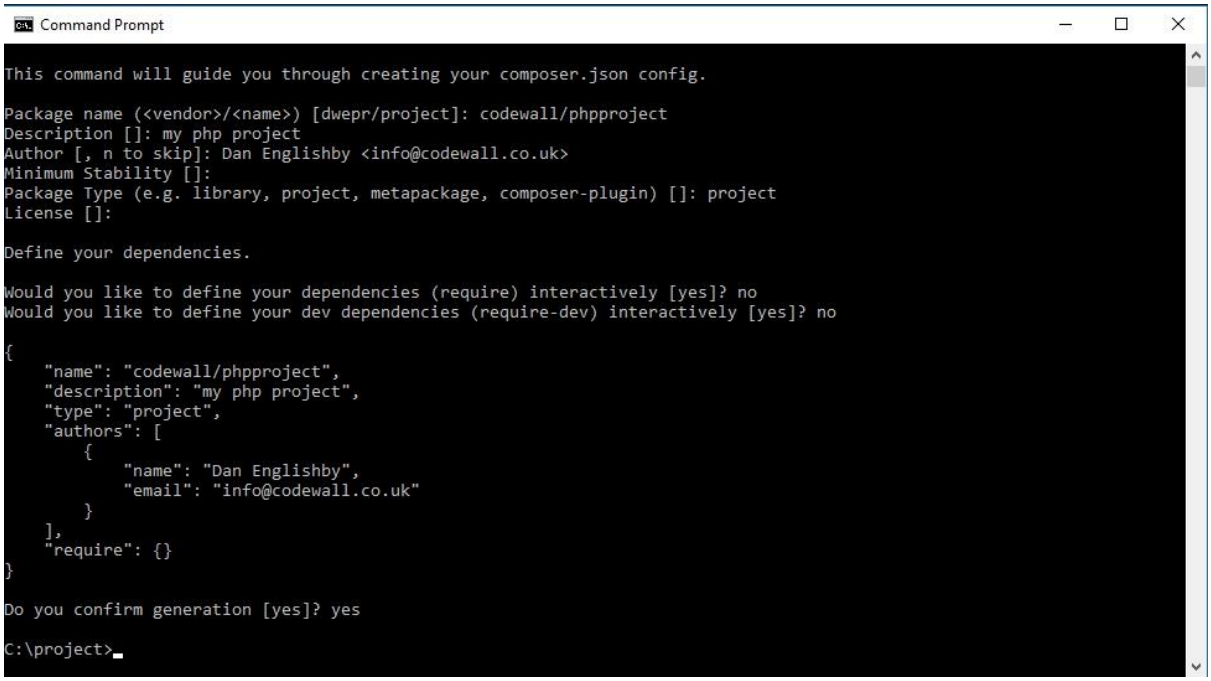

Figure 1.

Step 2

Now let's double check that the file was actually generated.

a. Go to your project folder and locate a file named composer.json.
b. Open the file and check the configuration you entered is present.
c. If it is, then we are good to go! If not, go back to Step 1 and repeat all the sub steps.
d. It should look similar to Figure 2 below.

Figure 2.

Requiring a Package

Now for the fun part, the ability to download and install useful libraries with one line commands! There are actually two ways to do it, one via the command line direct. Or specifying the package within the composer.json file first then running a command. This probably sounds a bit confusing, but in the next couple of examples, you will will come to understand the meaning of two separate ways.

Method 1. Requiring Packages using the Require command

This first way and my personal favourite is to simply use the Require command and let Composer do it's magic. See the following steps to download a package named Guzzle, a excellent HTTP client library, but feel free to choose a library from packagist (https://packagist.org/) of your own, there is currently over 200 thousand to pick from!

1. Open the command prompt and navigate to your projects root directory, for example `cd C:\project`
2. Enter the following command for the packages default version - `composer require guzzlehttp/guzzle` and press enter. **If you need a specific version, skip this step and see step 3**.

3. Requiring a specific version use the following command `composer require guzzlehttp/guzzle ^versionNo`) and press enter.
4. Give it a few seconds and the process should begin.
5. The terminal will output some messages, and you may see that multiple packages have been installed, this is not uncommon, so do not worry. The package probably has dependencies of its own. See Figure 3 for the terminal state after requiring Guzzle.
6. Check your project directory for a new folder named vendor, this will now house some more folders which have just been downloaded and installed for you automatically!
7. Lastly, open your composer.json file again and you should see that the Require section now has a new package present. See Figure 4 for an example.

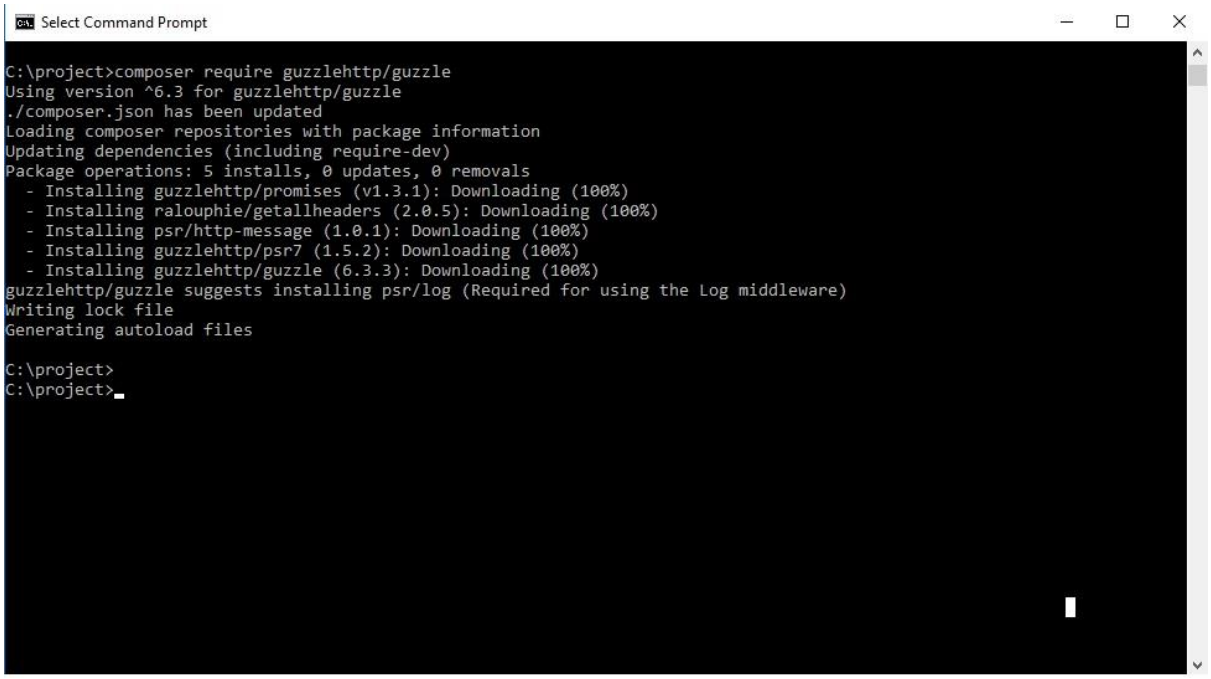

Figure 3.

Figure 4.

Method 2. Requiring Packages using the Install command

So, there is another method of downloading and installing packages without actually specifying the package name in the command line. Rather, you define this in the composer JSON file, and simply run `composer install`. See the following guide on how to utilize this method correctly.

1.
    1. Locate the composer.json file in your projects root directory, for example, C:\project
    2. Open the file with an editor. See Figure 2 above to see what it should be similar too, but maybe you already have packages defined from the last method.
    3. The require object is the point of importance, this is where we want to specify the next package to install.
    4. For this tutorial, the package of Guzzle will be installed again.
    5. Add the following line to your require section - `"guzzlehttp/guzzle": "^6.3"`
    6. Save the file, check Figure 4 for what your file should be similar too.
    7. Now boot up your command prompt and navigate to the root directory of your project, where the composer.json file is located. For example - `cd C:\project`
    8. Now run the following command - `composer install` and hit enter

9. Sit back, relax, and watch the new package install automatically.
10. See Figure 5 for an example of what it looks like after a update command is ran.
11. And that is it, check your vendor folder to see if the packages are now present.

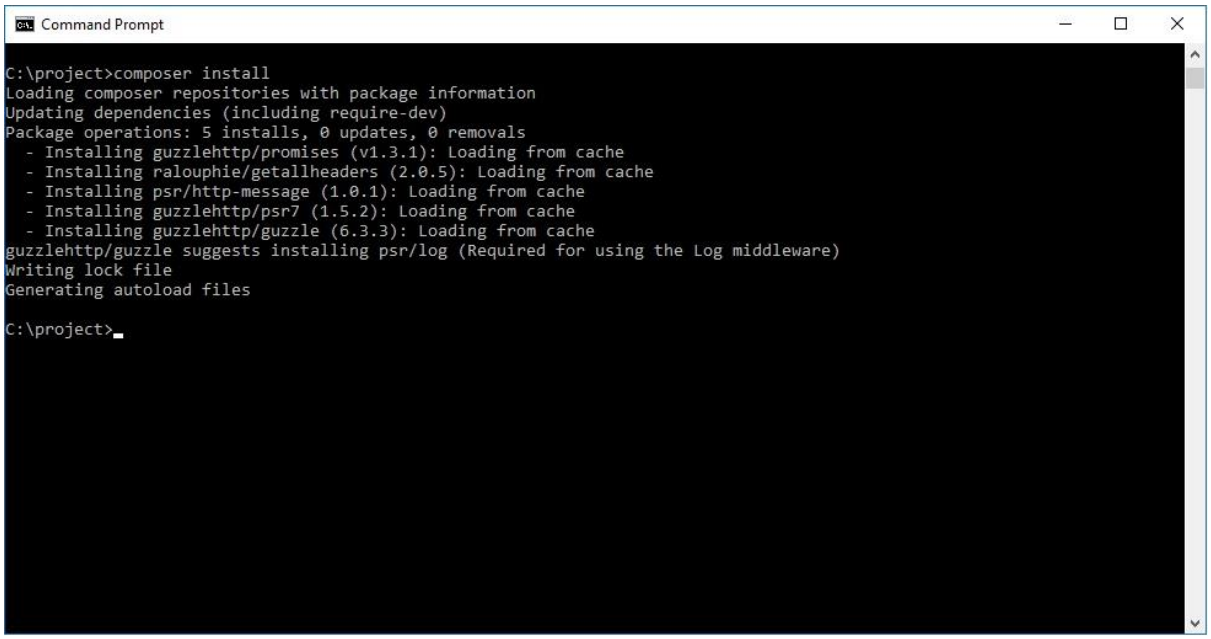

Figure 5.

Method 3. Requiring Packages using the Update command

There is one last way to install packages but it's not the recommended choice. **Before using this method, it must be understood that all packages in the require section of your JSON file may be updated, and could in fact, cause issues with dependencies you're already using.**

For example,

Package 1 is of version 1.0 currently, and your using it throughout your project. When you run composer update, it could update to version 1.2 and some of your code could be using deprecated items or something of a worse case.

Nevertheless, this command can become very handy if you wanted to update all your dependencies in one go!

. See the following guide on how to utilize this method correctly.

1.
    1. Locate the composer.json file in your projects root directory, for example, C:\project
    2. Open the file with an editor. See Figure 2 above to see what it should be similar too, but maybe you already have packages defined from the last method.
    3. The require object is the point of importance, this is where we want to specify the next package to install.

4. For this tutorial, the package of ChartJS (https://packagist.org/packages/nnnick/chartjs) will be installed, a visualization library.
5. Add the following line to your require section - `"nnnick/chartjs": "^2.7.3"`
6. Save the file, check Figure 6 for what it should be similar too.
7. Now boot up your command prompt and navigate to the root directory of your project, where the composer.json file is located. For example - `cd C:\project`
8. Now run the following command - `composer update` and hit enter
9. Sit back, relax, and watch the new package install automatically.
10. See Figure 7 for an example of what it looks like after a update command is ran.
11. And that is it!

Figure 6.

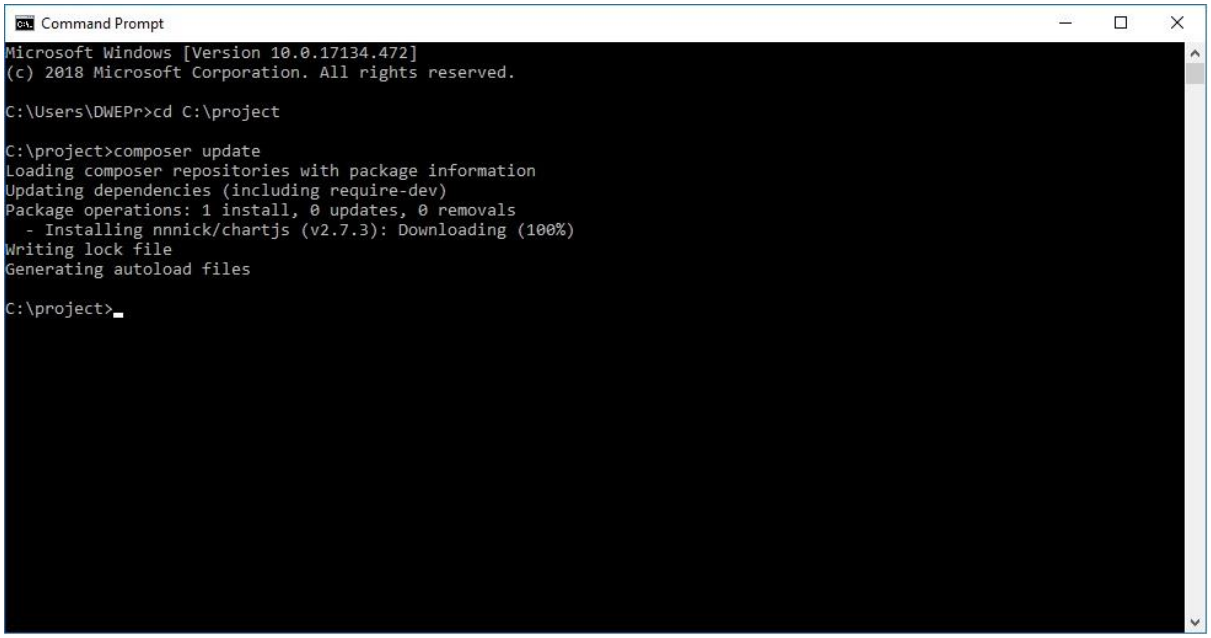

Figure 7.

Deleting Packages from your project

This is one of the real gems of Composer, the ability to delete packages with a few simple actions. The fact that project's package directories can become vast, it can be trivial to start removing unwanted packages. Not with Composer though, it really makes this part a dream for you. See the following steps on how to remove a package.

1. Locate the composer.json file in your projects root directory, for example, C:\project
2. Open the file and find the Require section
3. Remove one of the packages defined here, don't forget the trailing comma.
4. Save the file.
5. Open the command prompt and navigate to the root directory of your project, where the composer.json file is located. For example - `cd C:\project`
6. Now run the following command, referencing the package for removal - `composer remove nnnick/chartjs`and hit enter
7. Once again, allow Composer to do its thing.
8. You will notice that there is a removing message, see Figure 8 for an example.
9. Now the package has been removed, check your projects vendor folder, and you will notice that the package has now vanished!
10. Lastly, check your composer.json file and you will again, notice that this package has been removed from there too.

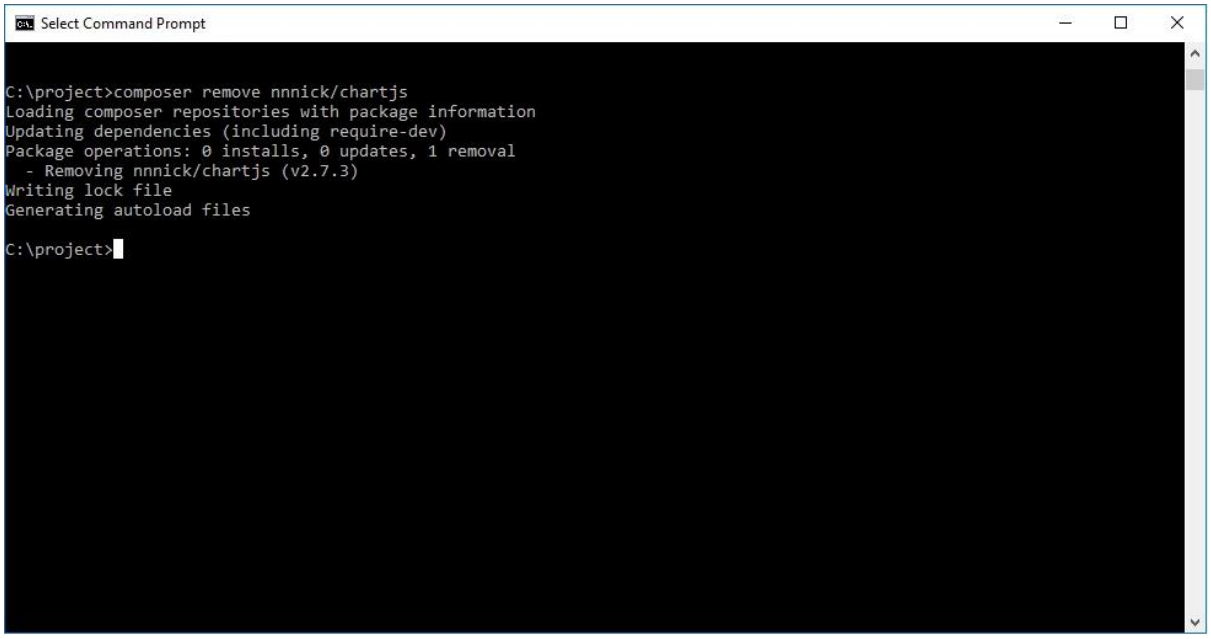

Figure 8.

Using a package in your project

In this very brief example, you will see how to utilize the projects that you have installed. See the following steps to use the GuzzleHttp package that was installed earlier in this tutorial.

1. Open your index.php file.
2. Add the following line to autoload all packages in the vendor directory. `require_once __DIR__ . '/vendor/autoload.php';`
3. Then make sure you define a use statement to expose the library. `use GuzzleHttp\Client;`
4. Finally, call the class like the following - `$guzzle = new Client();`
5. That is it, you can now use the packages full potential.
6. See the full example code below.

```
require_once __DIR__ . '/vendor/autoload.php';

use GuzzleHttp\Client;

$guzzle = new Client();
```

For further understanding of auto-loading packages, see this very detailed article on phpenthusiast. (https://phpenthusiast.com/blog/how-to-autoload-with-composer)

Summary

After this tutorial has been carried out, you will be able to see the power, convenience and great management benefits that are present. Composer is a excellent tool to have and is becoming more and more popular every day.