

CONTENTS

Prerequisites

Step 1 — Installing PHP and Additional Dependencies

Step 2 — Downloading and Installing Composer

Step 3 — Using Composer in a PHP Project

Step 4 — Including the Autoload Script

Step 5 — Updating Project Dependencies

Conclusion

RELATED

Initial Server Setup with Ubuntu 12.04

[View](#) 

How To Install Ruby on Rails on Ubuntu 12.04 LTS (Precise Pangolin) with RVM

[View](#) 

// Tutorial //

How To Install and Use Composer on Ubuntu 20.04

Published on May 4, 2020 · Updated on March 18, 2022

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!





Not using Ubuntu 20.04?

Choose a different version or distribution.

Ubuntu 20.04 ▼

Introduction

[Composer](#) is a popular dependency management tool for PHP, created mainly to facilitate installation and updates for project dependencies. It will check which other packages a specific project depends on and install them for you, using the appropriate versions according to the project requirements. Composer is also commonly used to bootstrap new projects based on popular PHP frameworks, such as [Symfony](#) and [Laravel](#).

In this tutorial, you'll install and get started with Composer on an Ubuntu 20.04 system.

Prerequisites

In order to follow this guide, you will need access to an Ubuntu 20.04 server as a non-root `sudo` user, and a firewall enabled on your server. To set this up, you can follow our

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!



In addition to dependencies that should be already included within your Ubuntu 20.04 system, such as `git` and `curl`, Composer requires `php-cli` in order to execute PHP scripts in the command line, and `unzip` to extract zipped archives. We'll install these dependencies now.

First, update the package manager cache by running:

```
$ sudo apt update
```

[Copy](#)

Next, run the following command to install the required packages:

```
$ sudo apt install php-cli unzip
```

[Copy](#)

You will be prompted to confirm installation by typing `Y` and then `ENTER`.

Once the prerequisites are installed, you can proceed to installing Composer.

Step 2 – Downloading and Installing Composer

Composer provides an [installer](#) script written in PHP. We'll download it, verify that it's not corrupted, and then use it to install Composer.

Make sure you're in your home directory, then retrieve the installer using `curl`:

```
$ cd ~  
$ curl -sS https://getcomposer.org/installer -o /tmp/composer-setup.php
```

[Copy](#)

Next, we'll verify that the downloaded installer matches the SHA-384 hash for the latest installer found on the [Composer Public Keys / Signatures](#) page. To facilitate the verification step, you can use the following command to programmatically obtain the latest hash from the Composer page and store it in a shell variable:

```
$ HASH=`curl -sS https://composer.github.io/installer.sig`
```

[Copy](#)

If you want to verify the obtained value, you can run:

```
$ echo $HASH
```

[Copy](#)

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

Now execute the following PHP code, as provided in the Composer [download page](#), to verify that the installation script is safe to run:

```
$ php -r "if (hash_file('SHA384', '/tmp/composer-setup.php') === '$HASH') { echo 'Copy e
```

You'll see the following output:

Output

```
Installer verified
```

If the output says `Installer corrupt`, you'll need to download the installation script again and double check that you're using the correct hash. Then, repeat the verification process. When you have a verified installer, you can continue.

To install `composer` globally, use the following command which will download and install Composer as a system-wide command named `composer`, under `/usr/local/bin`:

```
$ sudo php /tmp/composer-setup.php --install-dir=/usr/local/bin --file Copy c
```

You'll see output similar to this:

Output

```
All settings correct for using Composer
Downloading...
```

```
Composer (version 2.2.9) successfully installed to: /usr/local/bin/composer
Use it: php /usr/local/bin/composer
```

To test your installation, run:

```
$ composer
```

Copy

Output

```
/usr/local/bin/composer
```

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

Options:

-h, --help	Display this help message
-q, --quiet	Do not output any message
-V, --version	Display this application version
--ansi	Force ANSI output
--no-ansi	Disable ANSI output
-n, --no-interaction	Do not ask any interactive question
--profile	Display timing and memory usage information
--no-plugins	Whether to disable plugins.
-d, --working-dir=WORKING-DIR	If specified, use the given directory as work
--no-cache	Prevent use of the cache
-v vv vvv, --verbose	Increase the verbosity of messages: 1 for nor
...	

This verifies that Composer was successfully installed on your system and is available system-wide.

Note: If you prefer to have separate Composer executables for each project you host on this server, you can install it locally, on a per-project basis. This method is also useful when your system user doesn't have permission to install software system-wide.

To do this, use the command `php /tmp/composer-setup.php`. This will generate a `composer.phar` file in your current directory, which can be executed with `php composer.phar`.

Now let's look at using Composer to manage dependencies.

Step 3 – Using Composer in a PHP Project

PHP projects often depend on external libraries, and managing those dependencies and their versions can be tricky. Composer solves that problem by keeping track of project versions and dependencies, while also facilitating the process of finding, installing, and updating packages that are required by a project.

In order to use Composer in your project, you'll need a `composer.json` file. The `composer.json` file tells Composer which dependencies it needs to download for your project, and which versions of each package are allowed to be installed. This is extremely important to keep your project consistent and avoid installing unstable

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

`composer.json` file when you run a `composer require` command to include a dependency in a newly created project.

The process of using Composer to install a package as dependency in a project involves the following steps:

- Identify what kind of library the application needs.
- Research a suitable open source library on [Packagist.org](https://packagist.org), the official package repository for Composer.
- Choose the package you want to depend on.
- Run `composer require` to include the dependency in the `composer.json` file and install the package.

Let's try this out with a demo application.

The goal of this application is to transform a given sentence into a URL-friendly string - a *slug*. This is commonly used to convert page titles to URL paths (like the final portion of the URL for this tutorial).

Let's start by creating a directory for our project. We'll call it **slugify**:

```
$ cd ~  
$ mkdir slugify  
$ cd slugify
```

Copy

Although not required, you could now run a `composer init` command to create a detailed `composer.json` file for your project. Because our project's only objective is to demonstrate how to install dependencies with Composer, we'll use a simpler `composer.json` file that will be auto-generated when we require our first package.

Now it's time to search [Packagist.org](https://packagist.org) for a package that can help us generate *slugs*. If you search for the term "slug" on Packagist, you'll get a result similar to this:

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

The screenshot shows the Packagist website with the search term 'slug' entered. The page displays a list of packages and a sidebar with package types and tags.

Package Name	Description	PHP	Installs	Stars
cocur/slugify	Converts a string into a slug.	PHP	17 079 056	2 477
behat/transliterator	String transliterator	PHP	42 061 730	1 546
stof/doctrine-extensions-bundle	Integration of the gedmo/doctrine-extensions with Symfony2	PHP	20 851 302	1 609
nette/utils	Nette Utils: lightweight utilities for string & array manipulation, image handling, safe JSON encoding/decoding, validation, slug or strong password generating etc.	PHP	15 983 306	1 006
gedmo/doctrine-extensions	Doctrine2 behavioral extensions	PHP	29 390 595	3 286
cviebrock/eloquent-sluggable	Easy creation of slugs for your Eloquent models in Laravel	PHP	2 901 288	2 885
ausi/slug-generator	Slug Generator	PHP	242 535	714

Package type

amila-laravel-cms-plugin	5
anime-db-plugin	8
application	4
authserver-plugin	4
barberry-plugin	5
cakephp-plugin	1,533
canvashack-plugin	24
cdev-plugin	6
cetera-cms-plugin	12
claroline-plugin	74
command	4
component	63
composer-installer	19
composer-plugin	621
contao-bundle	27

Tags

Search for other...

- ☐ plugin 2,804
- ☐ cms 1,130
- ☐ wordpress 1,057
- ☐ craftcms 1,003
- ☐ craft 967
- ☐ craft-plugin 931
- ☐ cakephp 845
- ☐ elgg 493
- ☐ yii2 483

You'll see two numbers on the right side of each package in the list. The number on the top represents how many times the package was installed via Composer, and the number on the bottom shows how many times a package was starred on GitHub. Generally speaking, packages with more installations and more stars tend to be more stable, since so many people are using them. It's also important to check the package description for relevance to make sure it's what you need.

We need a *string-to-slug* converter. From the search results, the package `cocur/slugify`, which appears as the first result in that page, seems to be a good match, with a reasonable amount of installations and stars.

Packages on Packagist have a **vendor** name and a **package** name. Each package has a unique identifier (a namespace) in the same format GitHub uses for its repositories: `vendor / package`. The library we want to install uses the namespace `cocur/slugify`. You need a package's namespace in order to require it in your project.

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

modules a package relies on. In the case of the `cocur/slugify` package, it requires a PHP module that we haven't installed yet.

When a required package relies on a system library that is currently not installed on your server, you will get an error telling which requirement is missing:

```
$ composer require cocur/slugify
```

[Copy](#)

Output

```
Using version ^4.0 for cocur/slugify
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Your requirements could not be resolved to an installable set of packages.
```

Problem 1

- Installation request for cocur/slugify ^4.0 -> satisfiable by cocur/slugify v4.0.0
- cocur/slugify v4.0.0 requires ext-mbstring * -> the requested PHP extension mbstring is missing from your system.

...

To solve the system dependency problem, we can search for the missing package using `apt search`:

```
$ apt search mbstring
```

[Copy](#)

Output

```
Sorting... Done
Full Text Search... Done
php-mbstring/focal 2:7.4+75 all
  MBSTRING module for PHP [default]

php-patchwork-utf8/focal 1.3.1-1 all
  UTF-8 strings handling for PHP

php7.4-mbstring/focal 7.4.3-4ubuntu1 amd64
  MBSTRING module for PHP
```

After locating the correct package name, you can use `apt` once again to install the

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!


```
$ composer require cocur/slugify
```

Copy

Need live support within 30 minutes for mission-critical emergencies? Sign up for Premium Support! →

We're
hiring

[Blog](#)[Docs](#)

Get
Support

[Sales](#)

Questions Learning Paths For Businesses For Developers Social Impact



```
Writing lock file
Generating autoload files
```

As you can see from the output, Composer automatically decided which version of the package to use. If you check your project's directory now, it will contain two new files: `composer.json` and `composer.lock`, and a `vendor` directory:

```
$ ls -l
```

Copy

Output

```
total 12
-rw-rw-r-- 1 sammy sammy  59 May  4 13:56 composer.json
-rw-rw-r-- 1 sammy sammy 3229 May  4 13:56 composer.lock
drwxrwxr-x 4 sammy sammy 4096 May  4 13:56 vendor
```

The `composer.lock` file is used to store information about which versions of each package are installed, and ensure the same versions are used if someone else clones your project and installs its dependencies. The `vendor` directory is where the project dependencies are located. The `vendor` folder shouldn't be committed into version control - you only need to include the **`composer.json`** and **`composer.lock`** files.

When installing a project that already contains a `composer.json` file, run `composer install` in order to download the project's dependencies.

Let's take a quick look at version constraints. If you check the contents of your `composer.json` file, you'll see something like this:

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

```
"cocur/slugify": "^4.0"
```



```
}  
}
```

You might notice the special character `^` before the version number in `composer.json`. Composer supports several different constraints and formats for defining the required package version, in order to provide flexibility while also keeping your project stable. The caret (`^`) operator used by the auto-generated `composer.json` file is the recommended operator for maximum interoperability, following [semantic versioning](#). In this case, it defines **4.0** as the minimum compatible version, and allows updates to any future version below **5.0**.

Generally speaking, you won't need to tamper with version constraints in your `composer.json` file. However, some situations might require that you manually edit the constraints—for instance, when a major new version of your required library is released and you want to upgrade, or when the library you want to use doesn't follow semantic versioning.

Here are some examples to give you a better understanding of how Composer version constraints work:

Constraint	Meaning	Example Versions Allowed
<code>^1.0</code>	<code>>= 1.0 < 2.0</code>	1.0, 1.2.3, 1.9.9
<code>^1.1.0</code>	<code>>= 1.1.0 < 2.0</code>	1.1.0, 1.5.6, 1.9.9
<code>~1.0</code>	<code>>= 1.0 < 2.0.0</code>	1.0, 1.4.1, 1.9.9
<code>~1.0.0</code>	<code>>= 1.0.0 < 1.1</code>	1.0.0, 1.0.4, 1.0.9
<code>1.2.1</code>	<code>1.2.1</code>	1.2.1

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

For a more in-depth view of Composer version constraints, see [the official documentation](#).

Next, let's look at how to load dependencies automatically with Composer.

Step 4 – Including the Autoload Script

Since PHP itself doesn't automatically load classes, Composer provides an autoload script that you can include in your project to get autoloading working for your project. This file is automatically generated by Composer when you add your first dependency.

The only thing you need to do is include the `vendor/autoload.php` file in your PHP scripts before any class instantiation.

Let's try it out in our demo application. Open a new file called `test.php` in your text editor:

```
$ nano test.php
```

[Copy](#)

Add the following code which brings in the `vendor/autoload.php` file, loads the `cocur/slugify` dependency, and uses it to create a slug:

test.php

```
<?php
require __DIR__ . '/vendor/autoload.php';

use Cocur\Slugify\Slugify;

$slugify = new Slugify();

echo $slugify->slugify('Hello World, this is a long sentence and I need to mak
```

[Copy](#)

Save the file and exit your editor.

Now run the script:

```
$ php test.php
```

[Copy](#)

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

Step 5 – Updating Project Dependencies

Whenever you want to update your project dependencies to more recent versions, run the update command:

```
$ composer update
```

[Copy](#)

This will check for newer versions of the libraries you required in your project. If a newer version is found and it's compatible with the version constraint defined in the `composer.json` file, Composer will replace the previous version installed. The `composer.lock` file will be updated to reflect these changes.

You can also update one or more specific libraries by specifying them like this:

```
$ composer update vendor/package vendor2/package2
```

[Copy](#)

Be sure to check in your `composer.json` and `composer.lock` files within your version control system after you update your dependencies so that others can install these newer versions too.

Conclusion

Composer is a powerful tool that can greatly facilitate the work of managing dependencies in PHP projects. It provides a reliable way of discovering, installing, and updating PHP packages that a project depends on. In this guide, we saw how to install Composer, how to include new dependencies in a project, and how to update these dependencies once new versions are available.

Get Ubuntu on a hosted virtual machine in seconds with DigitalOcean Droplets! Simple enough for any user, powerful enough for fast-growing applications or businesses.

[Learn more here →](#)

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!



Want to learn more? Join the DigitalOcean Community!

Join our DigitalOcean community of over a million developers for free! Get help and share knowledge in our Questions & Answers section, find tutorials and tools that will help you grow as a developer and scale your project or business, and subscribe to topics of interest.

Sign up now →

About the authors



[Erika Heidi](#) Author

Developer Advocate

Dev/Ops passionate about open source, PHP, and Linux.

Still looking for an answer?

Ask a question

Search for more help

Was this helpful?

Yes

No













Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!



6 Comments

B *I* U ~~S~~    H₁ H₂ H₃   “,”     

Leave a comment...

This textbox defaults to using **Markdown** to format your answer.

You can type **!ref** in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

Chris Rawley • February 25, 2022

I'm having trouble getting Composer installed with a USER_DATA script through the API. I have run many scripts with no issues. As far as I can tell, a permissions issue is preventing this line from running the Composer Installer script:

```
php /var/www/myapp/app/composer-setup.php --install-dir=/usr/local/bin --
filename=composer
```

USER_DATA Script below.

```
USER_DATA="#!/bin/sh
```

```
#UPDATE UBUNTU apt -y install php-cli unzip
```

```
curl -sS https://getcomposer.org/installer -o /var/www/myapp/app/composer-  
setup.php chmod 755 /var/www/myapp/app/composer-setup.php php  
/var/www/myapp/app/composer-setup.php --install-dir=/usr/local/bin --  
filename=composer
```

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

```
/root/composer.ph /root/composer.phar install --working-dir=/var/www/myapp/app -n "
```

[Reply](#)

David Rodríguez • November 18, 2021 

Used to use containers and tools like Lando or DDEV, I had to upgrade my main system and did not remember some steps. This tutorial has been very useful for me to reinstall Composer! Thank you so much.

[Reply](#)

SimonBrown • September 2, 2021 

Ubuntu comes with apt so why don't we use it to install composer?

[Show replies](#)  [Reply](#)

cloudnine • May 18, 2021 

This comment has been deleted

[Reply](#)

Rakesh kumar • October 16, 2020 

thanks for this informative blog...ma'am ,I was so confused before...

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

Really helpful and clear coverage on composer!

[Reply](#)



This work is licensed under a Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International License.

Try DigitalOcean for free

Click below to sign up and get **\$200 of credit** to try our products over 60 days!

Sign up →

Popular Topics

Ubuntu

Linux Basics

JavaScript

React

Python

Security

MySQL

Docker

Kubernetes

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

Questions

Q&A Forum

Ask a question

DigitalOcean Support



Congratulations on unlocking the whale ambience easter egg! Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

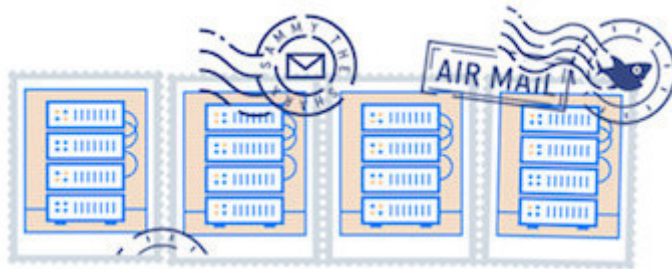


Thank you to the [Glacier Bay National Park & Preserve](#) and [Merrick079](#) for the sounds behind this easter egg.



Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the [Whale and Dolphin Conservation](#).

Reset easter egg to be discovered again / Permanently dismiss and hide easter egg



GET OUR BIWEEKLY NEWSLETTER

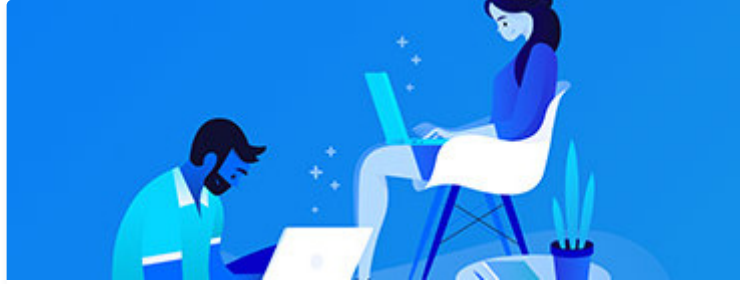
Sign up for Infrastructure as a
Newsletter.



Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

and spurring economic growth?
We'd like to help.



BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

Featured on Community [Intro to Kubernetes](#) [Learn Python 3](#) [Machine Learning in Python](#)
[Getting started with Go](#)

DigitalOcean Products [Virtual Machines](#) [Managed Databases](#) [Managed Kubernetes](#) [Block Storage](#)
[Object Storage](#) [Marketplace](#) [VPC](#) [Load Balancers](#)

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

[Learn More](#)





Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

Company	Products	Community	Solutions	Contact
About	Products	Tutorials	Website Hosting	Support
Leadership	Overview	Q&A	VPS Hosting	Sales
Blog	Droplets	CSS-Tricks	Web & Mobile Apps	Report Abuse
Careers	Kubernetes	Write for	Game Development	System Status
Customers	App Platform	DOnations	Streaming	Share your ideas
Partners	Functions	Currents	VPN	
Channel Partners	Cloudways	Research	SaaS Platforms	
Referral Program	Managed Databases	Hatch Startup Program	Cloud Hosting for Blockchain	
Affiliate Program	Spaces	deploy by DigitalOcean	Startup Resources	
Press	Marketplace	Shop Swag		
Legal	Load Balancers	Research Program		
Security	Block Storage	Open Source		
Investor Relations	Tools & Integrations	Code of Conduct		
DO Impact	API	Newsletter Signup		
	Pricing	Meetups		
	Documentation			
	Release Notes			
	Uptime			


© 2023 DigitalOcean, LLC. All rights reserved.



×

Try DigitalOcean for free

Click here to [Sign up](#) and get \$200 of credit to try our products over 60 days!

 COOKIE PREFERENCES