CONTENTS

Tutorial Series: How To Code in PHP

3/9 How To Install PHP 8.1 and S...
4/9 How To Write Your First PHP...
5/9 How To W

// Tutorial //

How To Install PHP 8.1 and Set Up a Local Development Environment on Ubuntu 22.04

Published on May 4, 2022

Development PHP Ubuntu 22.04

By Jamon Camisso

A previous version of this tutorial was written by alenaholligan.

Introduction

PHP is a popular server scripting language known for creating dynamic and interactive web pages. Getting up and running with your language of choice is the first step in learning to program.

This tutorial will guide you through installing PHP 8.1 on Ubuntu and setting up a local programming environment via the command line. You will also install a dependency manager, Composer, and test your installation by running a script.

Prerequisites

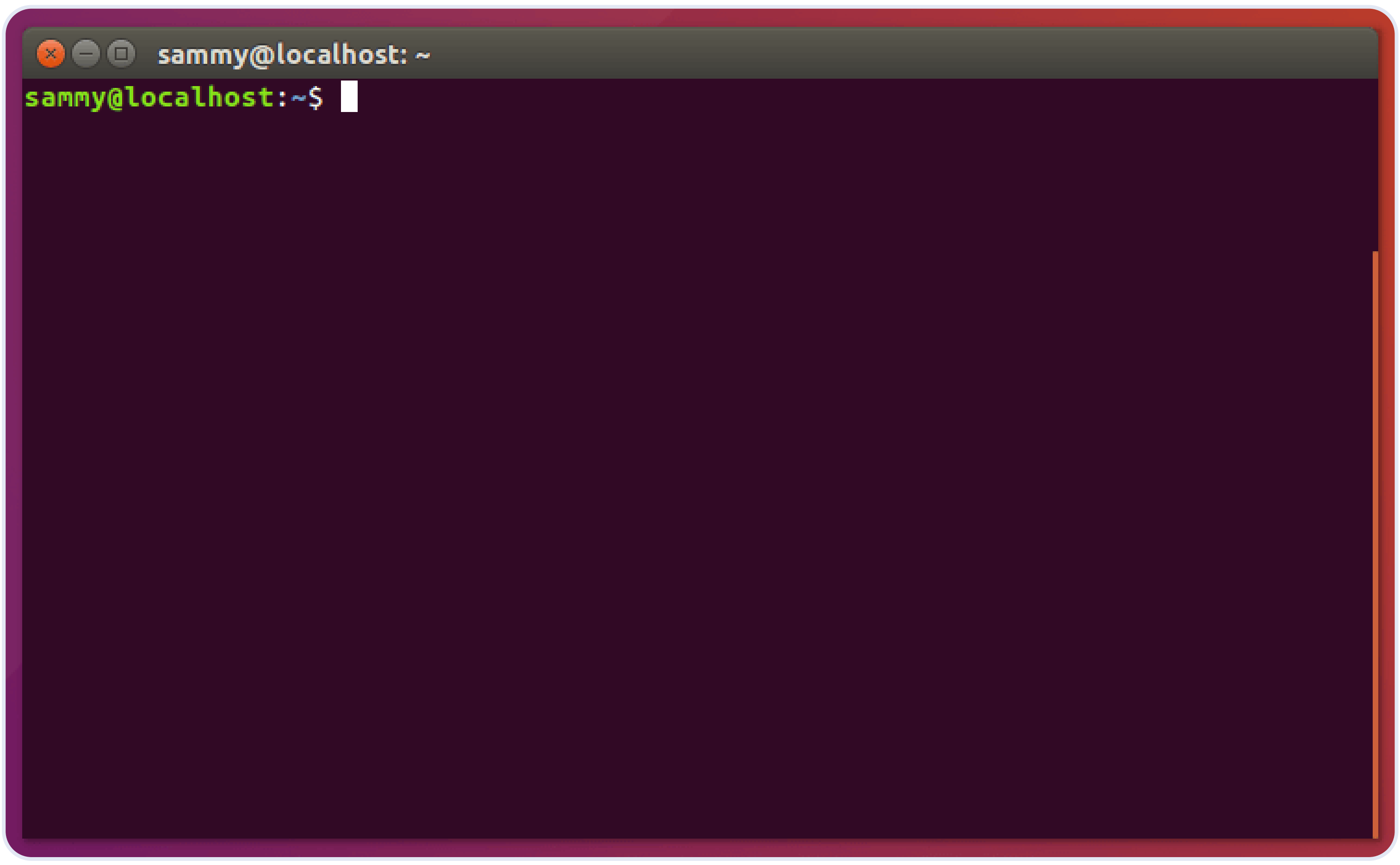
To complete this tutorial, you will need a local or virtual machine with Ubuntu 22.04 installed and have administrative access and an internet connection to that machine. You can download this operating system via the Ubuntu releases page.

Step 1 – Setting Up PHP 8.1

You'll be completing your installation and setup on the command line, which is a non-graphical way to interact with your computer. That is, instead of clicking on buttons, you'll be typing in text and receiving feedback from your computer through text as well.

The command line, also known as a shell or terminal, can help you modify and automate many of the tasks you do on a computer every day and is an essential tool for software developers. There are many terminal commands to learn that can enable you to do more powerful things. The article An Introduction to the Linux Terminal can get you better oriented with the terminal.

On Ubuntu, you can find the Terminal application by clicking on the Ubuntu icon in the upper-left-hand corner of your screen and typing `terminal` into the search bar. Click on the Terminal application icon to open it. Alternatively, you can hit the `CTRL`, `ALT`, and `T` keys on your keyboard at the same time to open the Terminal application automatically.



```
sammy@localhost:~$
```

Note: Ubuntu 22.04 ships with PHP 8.1 in its repositories. This means that if you attempt to install PHP without a specified version, it will use 8.1.

If you would like to use a different version of PHP on your Ubuntu 22.04 server, you can use the phpenv project to install and manage different versions.

Run the following commands to update your list of available packages, then then install PHP 8.1:

```
$ sudo apt update
```

```
$ sudo apt install --no-install-recommends php8.1
```

You will receive output like this:

```
Output
PHP 8.1.2 (cli) (built: Apr  7 2022 17:46:26) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.2, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.2, Copyright (c), by Zend Technologies
```

Besides PHP itself, you will likely want to install some additional PHP modules. You can use this command to install additional modules, replacing `PACKAGE_NAME` with the package you wish to install:

```
$ sudo apt-get install php8.1-PACKAGE_NAME
```

You can also install more than one package at a time. Here are a few suggestions of the most common modules you will most likely want to install:

```
$ sudo apt-get install -y php8.1-cli php8.1-common php8.1-mysql php8.1-zip php8.1-gd
```

This command will install the following modules:

- `php8.1-cli` - command interpreter, useful for testing PHP scripts from a shell or performing general shell scripting tasks
- `php8.1-common` - documentation, examples, and common modules for PHP
- `php8.1-mysql` - for working with MySQL databases
- `php8.1-zip` - for working with compressed files
- `php8.1-gd` - for working with images
- `php8.1-mbstring` - used to manage non-ASCII strings
- `php8.1-curl` - lets you make HTTP requests in PHP
- `php8.1-xml` - for working with XML data
- `php8.1-bcmath` - used when working with precision floats

PHP configurations related to Apache are stored in `/etc/php/8.1/apache2/php.ini`. You can list all loaded PHP modules with the following command:

```
$ php -m
```

You have installed PHP and verified the version you have running. You also installed any required PHP modules and were able to list the modules that you have loaded.

You could start using PHP right now, but you will likely want to use various libraries to build PHP applications quickly. Before you test your PHP environment, first set up a dependency manager for your projects.

Step 2 – Setting Up Composer for Dependency Management (Optional)

Libraries are a collection of code that can help you solve common problems without needing to write everything yourself. Since there are many libraries available, using a dependency manager will help you manage multiple libraries as you become more experienced in writing PHP.

Composer is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and will manage installing and updating these packages.

Although similar, Composer is not a package manager in the same sense as `yum` or `apt`. It deals with "packages" or libraries, but it manages them on a per-project basis, installing them in a directory (e.g. `vendor`) inside your project. By default, it does not install anything globally. Thus, it is a *dependency manager*. It does, however, support a global project for convenience via the `global` command.

This idea is not new, and Composer is strongly inspired by Node's `npm` and Ruby's `bundler`.

Suppose:

- You have a project that depends on several libraries.
- Some of those libraries depend on other libraries.

Composer:

- Enables you to declare the libraries you depend on.
- Finds out which versions of which packages can and need to be installed and installs them by downloading them into your project.
- Enables you to update all your dependencies in one command.
- Enables you to see the Basic Usage chapter for more details on declaring dependencies.

To install Composer, download the installer first with the following `curl` command:

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

Copy

Next, 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

To verify the obtained value, you can run:

```
$ echo $HASH
```

Copy

```
Output
55ce33d7678c5a611085589f1f3ddf8b3c52d662cd01d4ba75c0ee0459970c2200a51f492d557530c71c15d8dba01
```

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 'Ins
```

Copy

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 --filename=composer
```

Copy

You'll receive output similar to this:

```
Output
```

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

Composer (version 2.3.5) successfully installed to: /usr/local/bin/composer
Use it: php /usr/local/bin/composer
```

To test your installation, run:

```
$ composer
```

You will receive output like the following:

```
Output
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 2.3.5 2022-04-13 16:43:00

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display help for the given command. When no command is given
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi|--no-ansi           Force (or disable --no-ansi) ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
      --no-scripts               Skips the execution of all scripts defined in composer.json
  -d, --working-dir=WORKING-DIR  If specified, use the given directory as working directory.
      --no-cache                 Prevent use of the cache
  -v|vv|vvv, --verbose           Increase the verbosity of messages: 1 for normal output, 2 f
. . .
```

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 your 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` without any arguments. This command will generate a `composer.phar` file in your current directory, which you can run using `php composer.phar`.

Step 3 – Using Composer in a PHP Project

As a final step, you may optionally initialize your project with `composer init`. This will create the `composer.json` file that will manage your project dependencies. Initializing the project will also let you define project details such as Author and License, and use Composer's autoload functionality. You can define dependencies now or add them later.

First, make a directory and change into it to contain your project files:

```
$ cd ~
$ mkdir example-project
$ cd example-project
```

Now initialize your project:

```
$ composer init
```

Running this command will start the setup wizard. Details that you enter in the wizard can be updated later, so feel free to leave the defaults and just press `ENTER`. If you aren't ready to install any dependencies, you can choose `no`. Enter in your details at each prompt:

```
Output


  Welcome to the Composer config generator



This command will guide you through creating your composer.json config.

Package name (<vendor>/<name>) [sammy/example-project]:  sammy/project1
Description []:
Author [n to skip]: Sammy <sammy@digitalocean.com>
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []:

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? no
Would you like to define your dev dependencies (require-dev) interactively [yes]? no

Add PSR-4 autoload mapping? Maps namespace "Sammy\Project1" to the entered relative path. [sr

{
    "name": "sammy/project1",
    "type": "project",
    "authors": [
        {
            "name": "Sammy",
            "email": "sammy@digitalocean.com"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]? yes
```

Before you confirm the generation, you will see a sample of the `composer.json` file that the wizard will create. If it all looks good, you can confirm the default of `yes`. If you need to start over, choose `no`.

The first time you define any dependency, Composer will create a `vendor` folder. All dependencies install into this `vendor` folder. Composer also creates a `composer.lock` file. This file specifies the **exact** version of each dependency and sub-dependency used in your project. This assures that any machine on which your program is run, will be using the exact same version of each packages.

Note: The `vendor` folder should never be committed to your version control system (VCS). The `vendor` folder only contains packages you have installed from other vendors. Those individual vendors will maintain their own code in their own version control systems. You should only be tracking the code you write. Instead of committing the `vendor` folder, you only need to commit your `composer.json` and `composer.lock` files. You can learn more about ignoring specific files in How To Use Git: A Reference Guide.

Now that you have PHP installed and a way to manage your project dependencies using Composer, you're ready to test your environment.

Step 3 – Testing the PHP Environment

To test that your system is configured correctly for PHP, you can create and run a basic PHP script. Call this script `hello.php`:

```
$ nano hello.php
```

Copy

This will open a blank file. Put the following text, which is valid PHP code, inside the file:

hello.php

```
<?php
echo 'Hello World!';
?>
```

Once you've added the text, save and close the file. You can do this by holding down the `CTRL` key and pressing the `x` key. Then choose `y` and press `ENTER`.

Now you can test to make sure that PHP processes your script correctly. Type `php` to tell PHP to process the file, followed by the name of the file:

```
$ php hello.php
```

If the PHP is processed properly, you will see only the characters within the quotes:

```
Output
Hello World!
```

PHP has successfully processed the script, meaning that your PHP environment is successfully installed and you're ready to continue your programming journey.

Conclusion

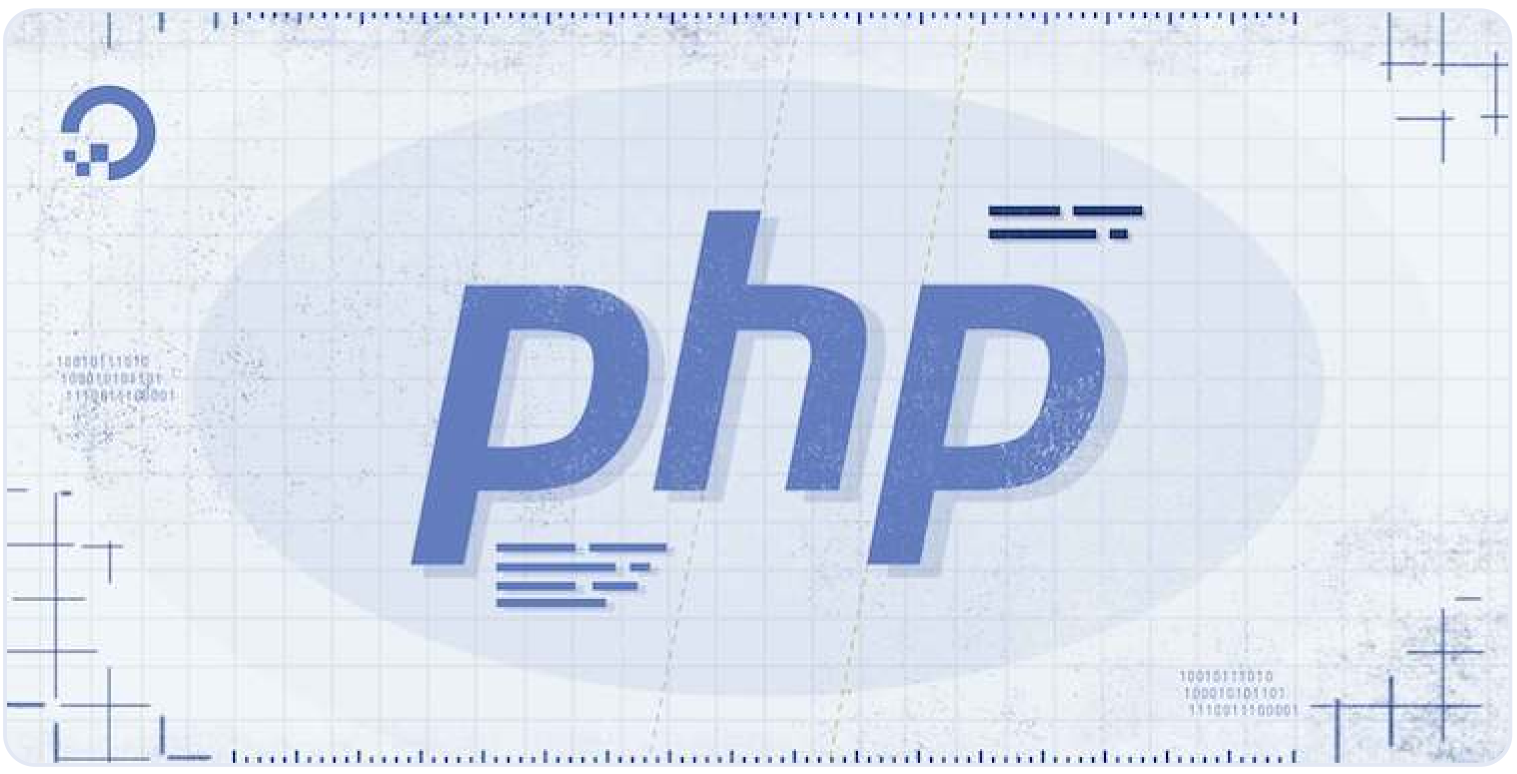
At this point, you have a PHP 8.1 programming environment set up on your Ubuntu system and can begin a coding project.

Before you start coding, you may want to set up an Integrated Development Environment (IDE). While there are many IDEs to choose from, VS Code is a popular choice as it offers many powerful features such as a graphical interface, syntax highlighting, and debugging.

With your local machine ready for software development, you can continue to learn more about coding in PHP by following How To Work With Strings in PHP.

Next in series: How To Write Your First PHP Program →

Tutorial Series: How To Code in PHP



PHP is a popular server scripting language known for creating dynamic and interactive web pages.

Subscribe

Development PHP Ubuntu 22.04

Browse Series: 9 articles

Expand to view all

About the authors

Jamon Camisso Author

Still looking for an answer? Ask a question Search for more help

Was this helpful? Yes No

Comments

4 Comments

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

tarikwaleed • May 9, 2023

to be able to folllow this guide, you have to add this PPA . run the following command `sudo add-apt-repository ppa:ondrej/php`

Reply

Geoff Maddock • March 8, 2023

Running into a confusing issue. I'm Ubuntu 18.04.6, but still attempted to install PHP 8.1 using the command in this, and it stated that it worked, however if I run php -v, I get:

```
gmaddock@Wrecked:/var/www/dev-events$ php -v
PHP 8.0.14 (cli) (built: Dec 20 2021 21:22:38) ( NTS )
Copyright (c) The PHP Group
Zend Engine v4.0.14, Copyright (c) Zend Technologies
    with Zend OPcache v8.0.14, Copyright (c), by Zend Technologies
```

If I run the install again, it says php8.1 is already the newest version:

```
gmaddock@Wrecked:/var/www/dev-events$ sudo apt install --no-install-recommends php8.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
php8.1 is already the newest version (8.1.16+repack-1+ubuntu18.04.1+deb.sury.org+1).
0 upgraded, 0 newly installed, 0 to remove and 168 not upgraded.
```

Reply

Enigma • January 10, 2023

You rock.

Reply

Jane Doe • May 13, 2022

This comment has been deleted

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

Python

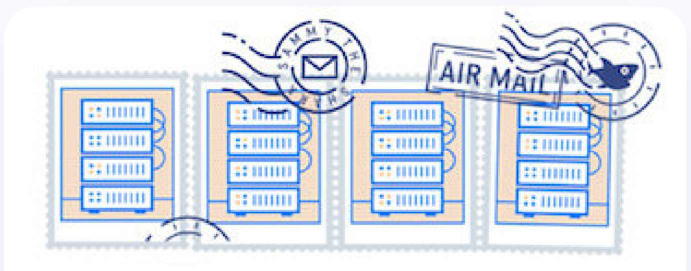
MySQL

Docker

Kubernetes

All tutorials →

Free Managed Hosting →



Get our biweekly newsletter

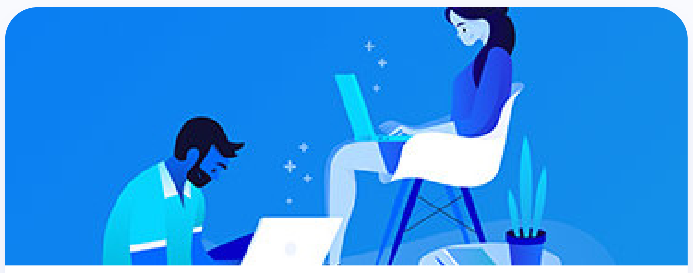
Sign up for Infrastructure as a Newsletter.

Sign up →

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

Learn more →



Become a contributor

You get paid; we donate to tech nonprofits.

Learn more →

Featured on Community

Kubernetes Course Learn Python 3 Machine Learning in Python Getting started with Go Intro to Kubernetes

DigitalOcean Products

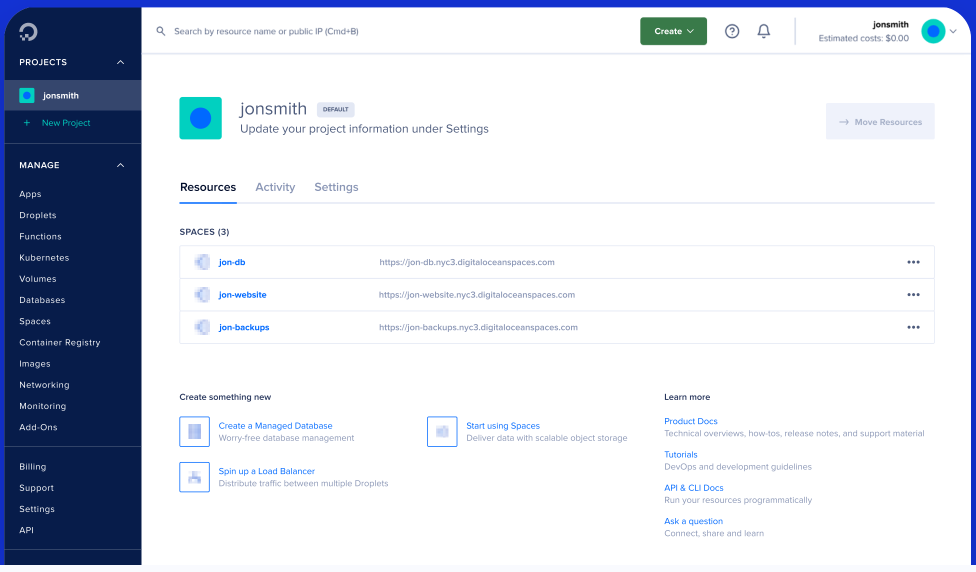
Cloudways Virtual Machines Managed Databases Managed Kubernetes Block Storage Object Storage

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 →


PROJECTS
jonsmith
New Project
MANAGE
Apps
Droplets
Functions
Kubernetes
Volumes
Databases
Spaces
Container Registry
Images
Networking
Monitoring
Add-Ons
Billing
Support
Settings
API
Search by resource name or public IP (Cmd+B)
Create
jonsmith
Estimated costs: $0.00
jonsmith DEFAULT
Update your project information under Settings
Move Resources
Resources
Activity
Settings
SPACES (3)
jon-db https://jon-db.nyc3.digitaloceanspaces.com
jon-website https://jon-website.nyc3.digitaloceanspaces.com
jon-backups https://jon-backups.nyc3.digitaloceanspaces.com
Create something new
Create a Managed Database
Worry-free database management
Start using Spaces
Deliver data with scalable object storage
Spin up a Load Balancer
Distribute traffic between multiple Droplets
Learn more
Product Docs
Technical overviews, how-tos, release notes, and support material
Tutorials
DevOps and development guidelines
API & CLI Docs
Run your resources programmatically
Ask a question
Connect, share and learn

Get started for free

Enter your email to get $200 in credit for your first 60 days with DigitalOcean.

Email address

Send My Promo

New accounts only. By submitting your email you agree to our Privacy Policy.

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

Uptime

Contact

Support

Sales

Report Abuse

System Status

Share your ideas