CONTENTS

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 Odoo on Ubuntu 20.04 with Docker

Published on March 11, 2022

Docker Ubuntu

By Jamon Camisso
Developer and author at DigitalOcean.

Introduction

Odoo is an open-source enterprise resource planning (ERP) tool written in Python. It supports a number of plugins for different kinds of business needs like accounting, payroll, inventory management, and more.

In this tutorial you will install Odoo and a PostgreSQL database using Docker Compose, then install Nginx to act as a reverse proxy for your Odoo site. Finally, you will enable secure HTTPS connections by using Certbot to download and configure a TLS certificate from the Let's Encrypt Certificate Authority.

Prerequisites

To complete this tutorial, you will need:

- An Ubuntu 20.04 server with 2 or more CPUs, a non-root user with sudo access, and a firewall. To set this up, follow our Initial Server Setup with Ubuntu 20.04 tutorial.
- Docker installed. Follow **Step 1** and **Step 2** of How To Install and Use Docker on Ubuntu 20.04 to install it and configure your non-**root** user to be able to run `docker` commands.

Note: You can skip these prerequisites if you are using DigitalOcean's One-Click Docker Image. This image is pre-configured with Docker, Docker Compose, and UFW.

Launch a new Docker image in the region of your choice, then log in as the **root** user and proceed with the tutorial.

Finally, to enable TLS you'll need a domain name pointed at your server's public IP address. This should be something like `example.com` or `odoo.example.com`. If you are using DigitalOcean, please see our DNS Quickstart for information on creating domain resources in your control panel.

Once you have all the prerequisites in place, proceed to **Step 1**, where you'll install the `docker-compose` package.

Step 1 – Installing Docker Compose

To install the `docker-compose` command line tool, refresh your package list, then install the package using `apt`:

```
$ sudo apt update
$ sudo apt install docker-compose
```

Copy

Note: You can also install a more recent Docker Compose package than the one that is included with Ubuntu 20.04. To do so, follow **Step 1** of How To Install and Use Docker Compose on Ubuntu 20.04.

If you opt to use this version of Docker Compose, you will need to substitute `docker compose` as the command in place of `docker-compose`.

You can confirm that the package is installed by running the following command:

```
$ docker-compose –version
```

Copy

You should receive output like the following:

```
Output
docker-compose version 1.25.0, build unknown
docker-py version: 4.1.0
CPython version: 3.8.10
```

Once you have confirmed that Docker Compose is installed on your server, you will configure and launch Odoo and PostgreSQL using Docker Compose in the next step of this tutorial.

Step 2 – Running Odoo and PostgreSQL with Docker Compose

To get started creating your Odoo and PostgreSQL containers, create a directory called `odoo` in your home directory to store the files that you will create in this tutorial. You'll use this directory to store all the files that you need to run Odoo.

files that you need to run Odoo.

Run the following commands to create the directory and then `cd` into it:

```
$ mkdir ~/odoo
$ cd ~/odoo
```

Copy

Now open a new blank YAML file called `docker-compose.yml` using `nano` or your preferred editor:

```
$ nano docker-compose.yml
```

Copy

You will use this file with the `docker-compose` command to start your Odoo and PostgreSQL containers and link them together. Add the following lines to the file:

docker-compose.yml

```
version: '3'
services:
  odoo:
    image: odoo:15.0
    env_file: .env
    depends_on:
      - postgres
    ports:
      - "127.0.0.1:8069:8069"
    volumes:
      - data:/var/lib/odoo
  postgres:
    image: postgres:13
    env_file: .env
    volumes:
      - db:/var/lib/postgresql/data/pgdata

volumes:
  data:
  db:
```

Copy

The file defines two `services`. The first is called `odoo`, which runs the Odoo application. The second is called `postgres`, which is the PostgreSQL database container. Both services reference named volumes that they use to store data outside of the running container instances. Finally, the `odoo` service exposes port `8069` on your server to the Odoo container that is running on the same port `8069`.

Save and exit the file when you are done editing it. If you are using `nano`, press `CTRL+O` then `RETURN` to save, then `CTRL+X` to exit.

The Odoo and PostgreSQL containers use environment variables to configure themselves. The `docker-compose.yml` file specifies the `env_file` directive for both services. That directive then includes the referenced file that contains the variables that each service needs to run.

This approach is generally recommended instead of adding environment variables to the `docker-compose.yml` file directly, sinceit is a good practice to keep passwords out of your `docker-compose.yml` file. This approach is especially applicable if you'll be committing your files to a Git repository or another source control system.

Open a new `.env` file with `nano`:

```
$ nano .env
```

Copy

Add the following lines into the file, substituting in a `POSTGRES_USER` and `POSTGRES_PASSWORD` of your choice in place of the highlighted values:

.env

```
# postgresql environment variables
POSTGRES_DB=postgres
POSTGRES_PASSWORD=a_strong_password_for_user
POSTGRES_USER=odoo
PGDATA=/var/lib/postgresql/data/pgdata

# odoo environment variables
```

```
HOST=postgres
USER=odoo
PASSWORD=a_strong_password_for_user
```

To generate a password for Odoo and PostgreSQL, use the `openssl` command, which should be available on most Linux systems. Run the following command on your server to generate a random set of bytes and print a base64 encoded version that you can use as a password:

```
$ openssl rand -base64 30
```
Copy

Paste the resulting string into your `.env` file in place of the `a_strong_password_for_user` placeholder passwords.

When you're done editing your `.env` file, save and exit your text editor.

You're now ready to start the `odoo` and `postgres` containers with the `docker-compose` command:

```
$ docker-compose up -d
```
Copy

The `up` sub-command tells `docker-compose` to start the containers and the associated volumes and networks that are defined in the `docker-compose.yml` file. The `-d` flag (which stands for "daemonize") tells `docker-compose` to run the containers in the background so the command doesn't take over your terminal. `docker-compose` will print some brief output as it downloads the required Docker images and then starts the containers:

```
Output
Creating network "odoo_default" with the default driver
Creating volume "odoo_odoo_data" with default driver
Creating volume "odoo_postgres_data" with default driver
Pulling odoo (odoo:14.0)...
15.0: Pulling from library/odoo
. . .
```

If you would like to stop your Odoo and PostgreSQL containers at any time, run the following command in your `~/odoo` directory:

```
$ docker-compose stop
```
Copy

The containers will be stopped. The configuration and data in their volumes will be preserved so that you can start the containers again with the `docker-compose up -d` command.

When that's done, Odoo should be running. You can test that a webserver is running at `127.0.0.1:8069` by fetching the homepage using the `curl` command:

```
$ curl --head http://localhost:8069
```
Copy

This will print out only the HTTP headers from the response:

```
Output
HTTP/1.0 303 SEE OTHER
Content-Type: text/html; charset=utf-8
Content-Length: 215
Location: http://localhost:8069/web
Set-Cookie: session_id=142fa5c02742d0f5f16c73bc14ec8144b8230f8a; Expires=Mon, 06-Jun-2022 20:45:34 GMT
Server: Werkzeug/0.14.1 Python/3.7.3
Date: Tue, 08 Mar 2022 20:45:34 GMT
```

The `303 SEE OTHER` response means the Odoo server is up and running, but that you should visit another page to complete the installation. The highlighted `http://localhost:8069/web` Location header indicates where to visit the Odoo installer page in your browser.

Next we'll set up Nginx to proxy public traffic to the Odoo container.

Step 3 – Installing and Configuring Nginx

Putting a web server such as Nginx in front of your Odoo server can improve performance by offloading caching, compression, and static file serving to a more efficient process. We're going to install Nginx and configure it to *reverse proxy* requests to Odoo, meaning it will take care of handing requests from your users to Odoo and back again. Using a non-containerized Nginx process will also make it easier to add Let's Encrypt TLS certificates in the next step.

First, refresh your package list, then install Nginx using `apt`:

```
$ sudo apt update
$ sudo apt install nginx
```

Copy

Allow public traffic to ports `80` and `443` (HTTP and HTTPS) using the **Nginx Full** UFW application profile:

```
$ sudo ufw allow "Nginx Full"
```

Copy

```
Output
Rule added
Rule added (v6)
```

Next, open up a new Nginx configuration file in the `/etc/nginx/sites-available` directory. We'll call ours `odoo.conf` but you could use a different name:

```
$ sudo nano /etc/nginx/sites-available/odoo.conf
```

Copy

Paste the following into the new configuration file, being sure to replace `your_domain_here` with the domain that you've configured to point to your Odoo server. This should be something like `odoo.example.com`, for instance:

/etc/nginx/sites-available/odoo.conf

```
server {
    listen       80;
    listen       [::]:80;
    server_name  your_domain_here;

    access_log  /var/log/nginx/odoo.access.log;
    error_log   /var/log/nginx/odoo.error.log;

    location / {
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Proto https;
      proxy_pass http://localhost:8069;
  }
}
```

This configuration is HTTP-only for now, as we'll let Certbot take care of configuring TLS in the next step. The rest of the configuration file sets up logging locations and then passes all traffic, as well as some important proxy headers, along to `http://localhost:8069`, the Odoo container that we started up in the previous step.

Save and close the file, then enable the configuration by linking it into `/etc/nginx/sites-enabled/`:

```
$ sudo ln -s /etc/nginx/sites-available/odoo.conf /etc/nginx/sites-enabled/
```

Copy

Use `nginx -t` to verify that the configuration file syntax is correct:

```
$ sudo nginx -t
```

Copy

```
Output
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
```

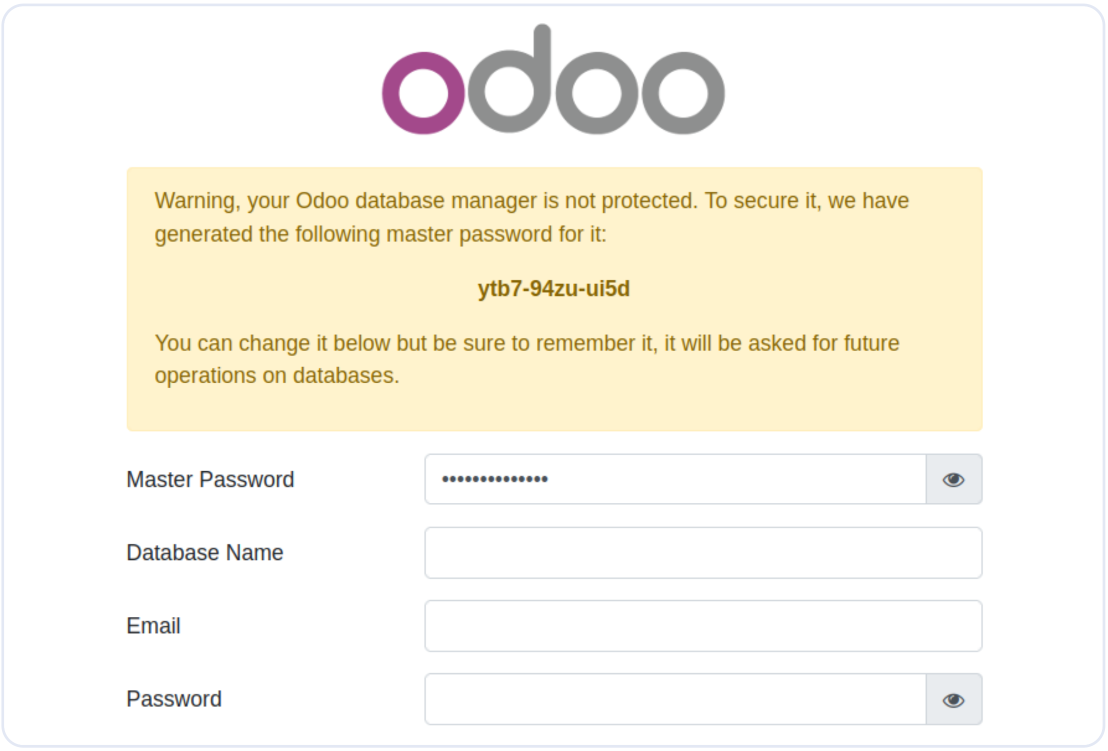
```
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

And finally, reload the `nginx` service with the new configuration:

```
$ sudo systemctl reload nginx.service
```

Copy

Your Odoo site should now be available on plain HTTP. Load `http://your_domain_here` (you may have to click through a security warning) and it will look like this:



Warning, your Odoo database manager is not protected. To secure it, we have generated the following master password for it:

ytb7-94zu-ui5d

You can change it below but be sure to remember it, it will be asked for future operations on databases.

Master Password	••••••••••••••
Database Name	
Email	
Password	

Now that you have your site up and running over HTTP, it's time to secure the connection with Certbot and Let's Encrypt certificates. You should do this *before* going through Odoo's web-based setup procedure.

Step 4 – Installing Certbot and Setting Up TLS Certificates

Thanks to Certbot and the Let's Encrypt free certificate authority, adding TLS encryption to your Odoo app will take only two commands.

First, install Certbot and its Nginx plugin:

```
$ sudo apt install certbot python3-certbot-nginx
```

Copy

Next, run `certbot` in `--nginx` mode, and specify the same domain that you used in the Nginx `server_name` configuration directive:

```
$ sudo certbot --nginx -d your_domain_here
```

Copy

You'll be prompted to agree to the Let's Encrypt terms of service, and to enter an email address.

Afterwards, you'll be asked if you want to redirect all HTTP traffic to HTTPS. It's up to you, but this is generally recommended and safe to do.

After that, Let's Encrypt will confirm your request and Certbot will download your certificate:

```
Output
Congratulations! You have successfully enabled https://odoo.example.com

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=odoo.example.com
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/odoo.example.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/odoo.example.com/privkey.pem
   Your cert will expire on 2022-05-09. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Certbot will automatically reload Nginx with the new configuration and certificates. Reload your site in your browser and it should switch you over to HTTPS automatically if you chose the redirect option.

Your site is now secure and it's safe to continue with the web-based setup steps.

Step 5 – Setting Up Odoo

Back in your web browser, reload the page. You should now have Odoo's database configuration page open via a secure `https://` connection. Now you can enter usernames and passwords safely to complete the installation process.

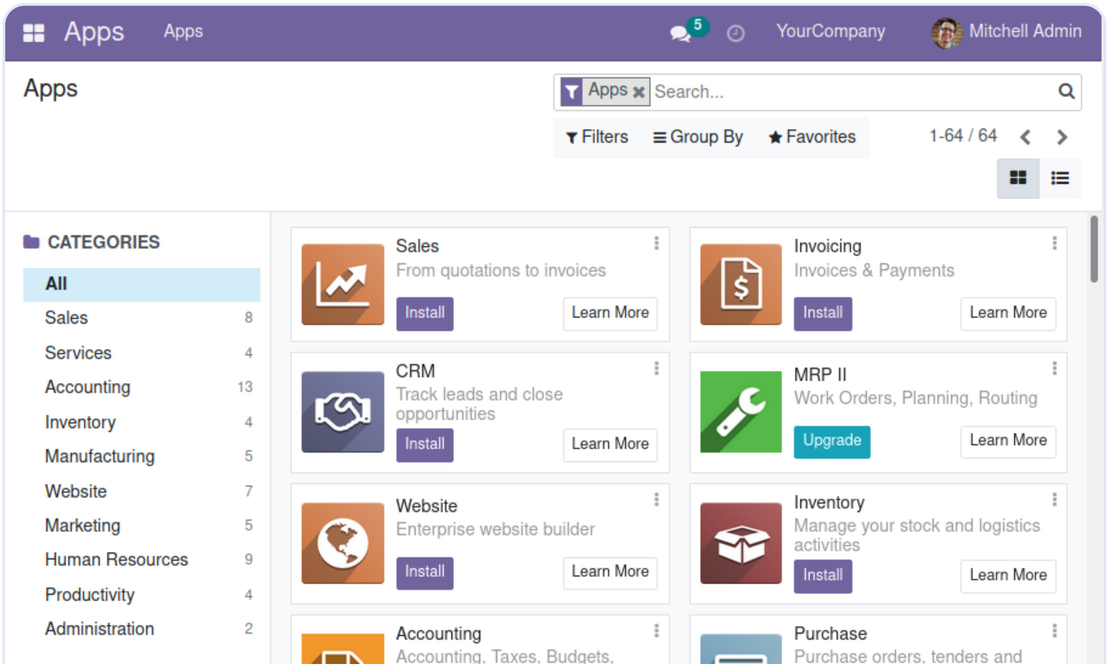
The information that you fill in on this page will tell the Odoo application how to create its PostgreSQL database and details about the default administrative user.

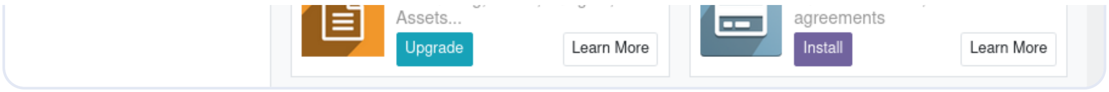
Fill out the following fields:

- **Database Name:** odoo
- **Email:** your email address
- **Password:** a strong and unique password for your administrator login
- **Demo data:** ensure that this option is checked if this is the first time that you are installing odoo

The defaults are fine for the remaining fields. Be sure to record the email and password values that you choose since you will use them to login to Odoo in the future.

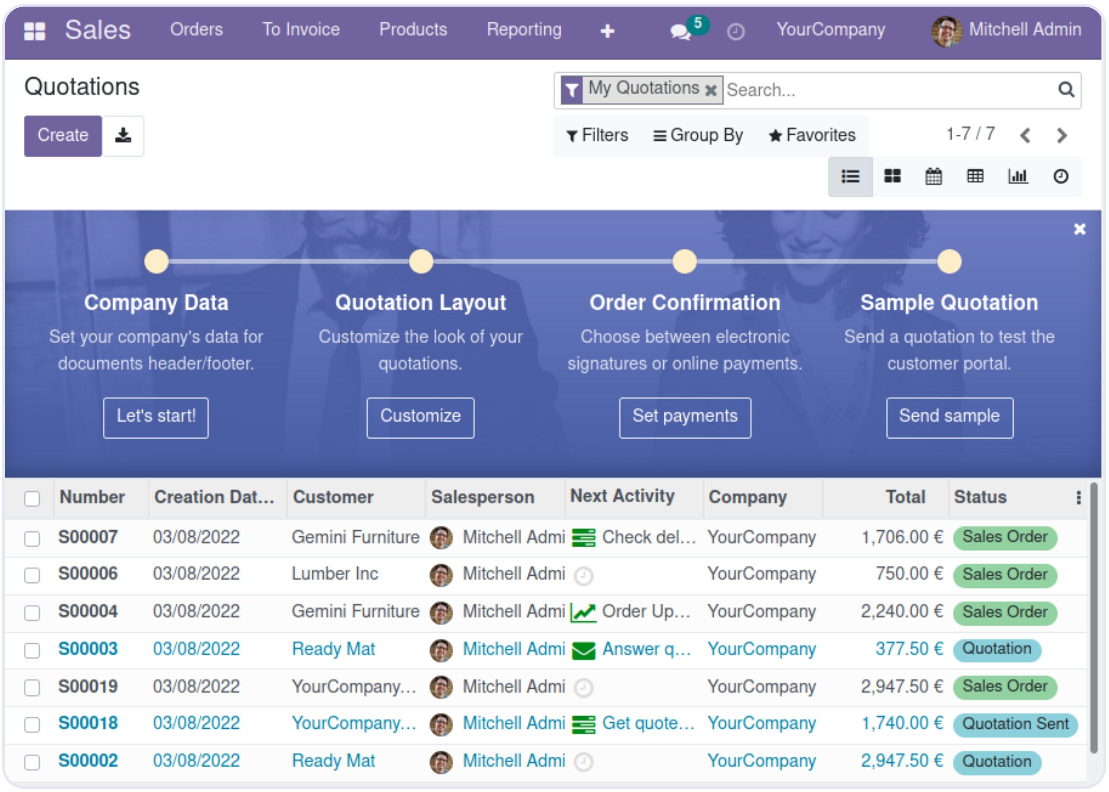
Now click the **Create database** button at the bottom left of the page. It may take a minute or two for Odoo to create its database tables. When the process is complete you will be redirected to the Odoo **Apps** administrative page.





From here you can choose which Odoo modules you would like to install and use for your ERP needs. If you would like to test an app, click the **Install** button on the **Sales** tile. Odoo will install the module and then redirect you to your personal Discuss app page.

Click the segmented square icon at the top left of your screen and then select the **Sales** link in the list of dropdown options.



Number	Creation Dat...	Customer	Salesperson	Next Activity	Company	Total	Status
S00007	03/08/2022	Gemini Furniture	Mitchell Admi	Check del...	YourCompany	1,706.00 €	Sales Order
S00006	03/08/2022	Lumber Inc	Mitchell Admi		YourCompany	750.00 €	Sales Order
S00004	03/08/2022	Gemini Furniture	Mitchell Admi	Order Up...	YourCompany	2,240.00 €	Sales Order
S00003	03/08/2022	Ready Mat	Mitchell Admi	Answer q...	YourCompany	377.50 €	Quotation
S00019	03/08/2022	YourCompany...	Mitchell Admi		YourCompany	2,947.50 €	Sales Order
S00018	03/08/2022	YourCompany...	Mitchell Admi	Get quote...	YourCompany	1,740.00 €	Quotation Sent
S00002	03/08/2022	Ready Mat	Mitchell Admi		YourCompany	2,947.50 €	Quotation

You will be on a page that will guide you through customizing data, quotes, orders, and a list of example sales that you can experiment with.

Conclusion

In this tutorial, you launched the Odoo ERP app and a PostgreSQL database using Docker Compose, then set up an Nginx reverse proxy and secured it using Let's Encrypt TLS certificates.

You're now ready to start building your ERP website using the supplied modules. For more information about using Odoo please see the official Odoo documentation.

If you would like to write your own custom Odoo modules or customize existing modules, the Developer documentation is a good place to start.

About the authors

Jamon Camisso Author
Developer and author at DigitalOcean.

Still looking for an answer?

Ask a question

Search for more help

Was this helpful? Yes No

Comments

1 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

SleepingElectricBlueWhale • July 13, 2022

hello, does it work for version 14? If I wanted to add third party modules, what would be the path to upload the files?

Popular Topics

Ubuntu
Linux Basics
JavaScript
React
Python
Security
Apache
MySQL
Databases
Docker
Kubernetes
Ebooks
Browse all topic tags

All tutorials →

Questions

Q&A
Ask a question
DigitalOcean Product Docs
DigitalOcean Support

Events

Tech Talks
Hacktoberfest
Deploy

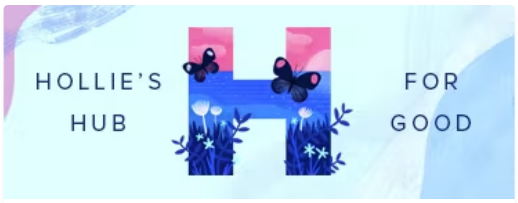
Get involved

Community Newsletter
Hollie's Hub for Good
Write for DOnations
Community tools and integrations
Hatch Startup program

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

GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.



HOLLIE'S HUB FOR GOOD

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

BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

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

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

C ny Products Community Solutions Contact