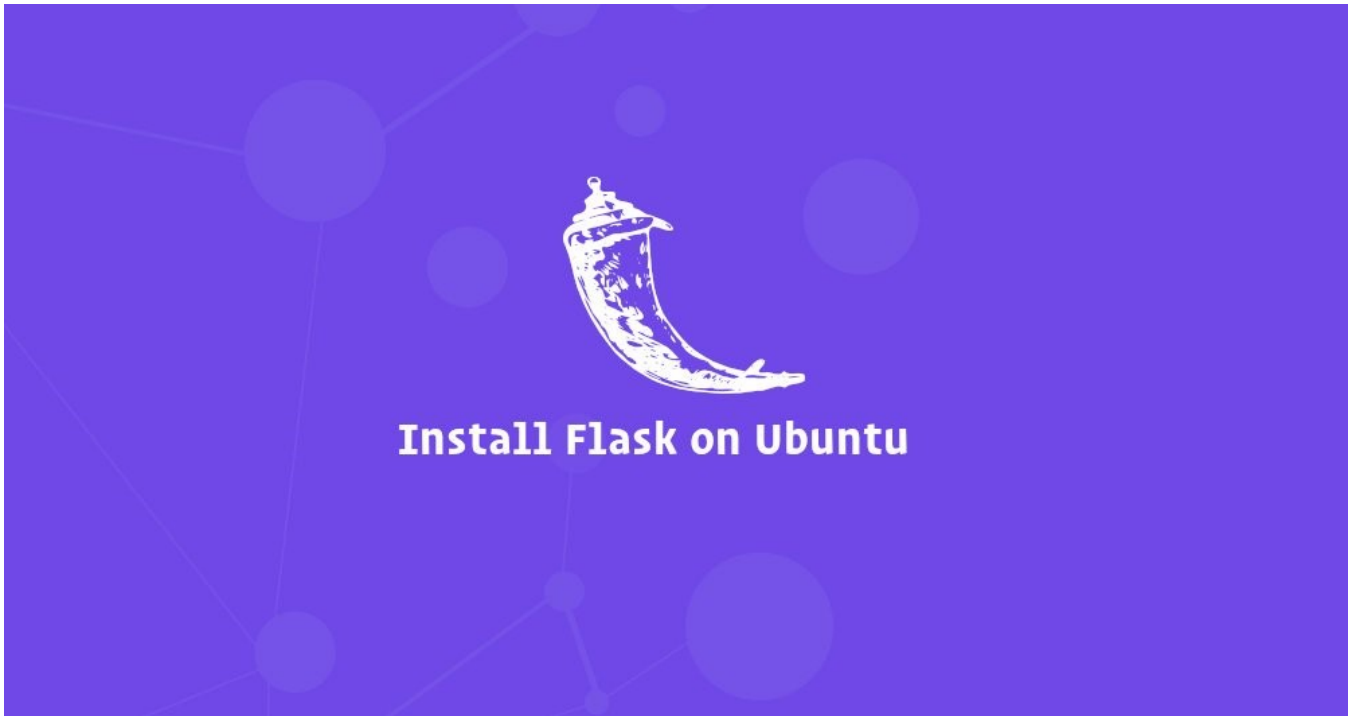
How to Install Flask on Ubuntu 18.04

Updated Apr 29, 2019 • 5 min read



Flask is a free and open-source micro web framework for Python designed to help developers build secure, scalable and maintainable web applications. Flask is based on Werkzeug and uses Jinja2 as a template engine.

Unlike Django, by default Flask doesn't include ORM, form validation or any other functionalities provided by third-party libraries. Flask is built with extensions in mind, which are Python packages that add functionality to a Flask application.

There are different methods to install Flask, depending on your needs. It can be installed system-wide or in a Python virtual environment using pip.

Flask packages are also included in the official Ubuntu repositories and can be installed using the `apt` package manager. This is the easiest method to install

Flask on Ubuntu 18.04, but not as flexible as installing in a virtual environment. Also, the version included in the repositories always lags behind the latest version of Flask.

The main purpose of Python virtual environments is to create an isolated environment for different Python projects. This way you can have multiple different Flask environments on a single computer and install a specific version of a module on a per project basis without worrying that it will affect your other Flask installations. If you install Flask into the global environment then you can install only one Flask version on your computer.

Installing Flask on Ubuntu 18.04

The following sections provide information about how to install Flask in a Python virtual environment on Ubuntu 18.04.

1. Installing Python 3 and venv

Ubuntu 18.04 ships with Python 3.6 by default. You can verify that Python 3 is installed on your system by typing:

```
$ python3 -V
```

The output should look like this:

```
Output
Python 3.6.6
```

Starting from Python 3.6, the recommended way to create a virtual environment is to use the `venv` module. To install the `python3-venv` package that provides the `venv` module run the following command:

```
$ sudo apt install python3-venv
```

Once the module is installed we are ready to create a virtual environment for our Flask application.

2. Creating a Virtual Environment

Start by navigating to the directory where you would like to store your Python 3 virtual environments. It can be your home directory or any other directory where your user has read and write permissions.

Create a new directory for your Flask application and navigate into it:

```
$ mkdir my_flask_app
$ cd my_flask_app
```

Once inside the directory, run the following command to create your new virtual environment:

```
$ python3 -m venv venv
```

The command above creates a directory called `venv`, which contains a copy of the Python binary, the Pip package manager, the standard Python library and other supporting files. You can use any name you want for the virtual environment.

To start using this virtual environment, you need to activate it by running the `activate` script:

```
$ source venv/bin/activate
```

Once activated, the virtual environment's bin directory will be added at the beginning of the `$PATH` variable. Also your shell's prompt will change and it will show the name of the virtual environment you're currently using. In our case that is `venv`:

3. Installing Flask

Now that the virtual environment is activated, you can use the Python package manager pip to install Flask:

```
(venv) $ pip install Flask
```

> Within the virtual environment, you can use the command `pip` instead of `pip3` and `python` instead of `python3`.

Verify the installation with the following command which will print the Flask version:

```
(venv) $ python -m flask --version
```

At the time of writing this article, the latest official Flask version is 1.0.2

```
Output
Flask 1.0.2
Python 3.6.6 (default, Sep 12 2018, 18:26:19)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]]
```

Your Flask version may differ from the version shown here.

4. Creating a Minimal Flask Application

In this guide, we will create a simple hello world application which will just display the text “Hello World!".

Open your text editor or Python IDE and create the following file:

~/my_flask_app/hello.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'
```

Let’s analyze the code line by line.

01. In the first line, we are importing the Flask class.

02. Next, we creating an instance of the Flask class.

03. Then we use the `route()` decorator to register the `hello_world` function for the `/` route. When this route is requested, `hello_world` is called and the message “Hello World!” is returned to the client.

Save the file as `hello.py` and go back to your terminal window.

5. Testing the Development Server

We’ll use the `flask` command to run the application but before that, we need to tell Flask how to load the application by specifying the `FLASK_APP` environment variable:

```
(venv) $ export FLASK_APP=hello
(venv) $ flask run
```

The command above will launch the development builtin server.

The output will look something like the following:

```
Output
 * Serving Flask app "hello"
 * Environment: production
   WARNING: Do not use the development server in a production er
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

> If you installed Flask on a virtual machine and you want to access Flask development server then you can make the server publicly available by appending `--host=0.0.0.0` to the `flask run` command.

Open `http://127.0.0.1:5000` in your web browser and you will be presented with the “Hello World!” message.

To stop the development server type `CTRL-C` in your terminal.

6. Deactivating the Virtual Environment

Once you are done with your work, deactivate the environment, by typing `deactivate` and you will return to your normal shell.

```
(venv) $ deactivate
```

Conclusion

You have learned how to create a Python virtual environment and install Flask on your Ubuntu 18.04 machine. To create additional Flask development environments repeat the steps we outlined in this tutorial.

If you are new to Flask, visit the Flask documentation page and learn how to develop your first Flask app.

python ubuntu

OCT 20, 2018

How to Install Django on Ubuntu 18.04

OCT 9, 2018

How to deploy Odoo 12 on Ubuntu 18.04

SEP 30, 2018

How to Install PyCharm on Ubuntu 18.04

Write a comment