CONTENTS

RELATED

How To Setup uWSGI On Ubuntu 12.10

View

How To Create Nagios Plugins With Python On CentOS 6

View

// Tutorial //

How To Manage Python with Pyenv and Direnv

Published on December 11, 2019

P

By Johnny M

While we believe that this content benefits our community, we have not yet thoroughly reviewed it. If you have any suggestions for improvements, please let us know by clicking the “report an issue“ button at the bottom of the tutorial.

# Introduction

Whether you’re just getting started or you’re a seasoned Python developer, you may have found managing your Python environments to be tedious and painful. Managing Python versions, libraries, and various dependencies is like playing shuffleboard where the object is to not hit any other puck; if you do, the probability of a cascading effect of pucks flying everywhere you don’t want them to be will soon follow.

In this tutorial, you’ll install pyenv for managing python environments, install direnv to auto configure and source the virtualenv for projects, set a global Python version and a local Python version for projects, and configure a virtualenv and auto source the venv [illegible] moving into your project directories. By the end of this tutorial, you will be able to install any valid version of python, set up and configure virtual environments for each project, and bring sanity from chaos.

# Prerequisites

We're going to install both `pyenv` and `direnv` via `homebrew` to ease the install process. Before getting started, install homebrew if you don't have it already.

# Working with pyenv

To start, you'll see how to work with pyenv.

# Installing pyenv

To get things started, we're going to install `pyenv` with homebrew and add the required `pyenv` init to our `~/.bashrc` file.

```
$ brew install pyenv
$ echo 'eval "$(pyenv init -)"' >> ~/.bashrc # initialize pyenv on
$ source ~/.bashrc # reinitialize bashrc to reflect changes in you
```

Once it's installed, take a moment to examine what our environment looks like. On a fresh system, you'll see something similar to this:

```
$ which pyenv
```

```
Output
/usr/local/bin/pyenv
```

```
$ pyenv versions
```

```
Output
* system
```

```
$ which pip
```

```
Output
/usr/local/bin/pip
```

```
$ which python
```

Copy

```
Output
/usr/local/bin/python
```

This is a pretty standard snapshot. When you execute `python version`, you'll notice `* system`, as the reference implies, this is your system's python version. The asterisk denotes the current python binary sourced in your `${PATH}`. A good rule of thumb is to leave the system binary alone. If you want to see the list of python versions that `pyenv` can install for you, use `pyenv install --list`.

```
$ pyenv install 2.7.15
$ pyenv install 3.7.0
$ pyenv versions
```

Copy

```
Output
* system (set by /Users/iamjohnnym/.pyenv/version)
  2.7.15
  3.7.0
```

# Configuring the Base Requirements

Now, let's upgrade `pip` since chances are, it installed an older version. The following command will loop through your installed versions and update `pip` to the latest.

```
for VERSION in $(pyenv versions --bare) ; do
  pyenv shell ${VERSION} ;
  pip install --upgrade pip ;
done
```

Copy

For the sake of our desired workflow, we want to install `py2venv` for our `Python 2.x` versions so we can mimic how `python 3.x` installs virtualenvs. `python -m venv env`.

```
for VERSION in $(pyenv versions --bare | egrep '^2.') ; do
  pyenv shell ${VERSION} ;
  pip install py2venv ;
done
```

Copy

# Setting the Global Python Version

Even though we have `pyenv` installed, it will still default to `system`. To change this, set `python 3.7.0` as our global version:

```
$ pyenv global 3.7.0
$ pyenv versions
```

Copy

```
Output
  system
  2.7.15
```

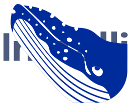
We're hiring Blog Docs Get Support Contact Sales

Tutorials Questions Learning Paths For Businesses Product Docs Social Impact

We’ve got `pyenv` setup and functional. Let’s move on to `direnv`.

# Working with direnv

`direnv` is a handy utility that allows you to create a file that’s placed in any directory that you want that functions like `.bashrc`. Whenever you enter the directory with this file, your shell will automatically execute it. The capabilities are endless but for the purpose of this post, we’re going to use it to configure a Python virtualenv based on the file `.python-version` and then activate it for us. The purpose is to create a seamless development flow. No need to worry about manually configuring or activating virtualenvs; let your computer do the work for you.



## Installing direnv

Installation is straightforward with homebrew:

```
$ brew install direnv
```

Copy

`direnv` is now installed, and ready for exploitation. Let’s try a sample project.

# Creating a New Project

Let’s start up a project. We’re going to create a new directory, set a local python version, set up our `.envrc` file, and activate it.

```
$ mkdir -p ~/python-projects/pyenv-tutorial
$ cd $_ # if you're unaware, $_ will execute the last argument of
$ pwd
```

Copy

```
Output
/Users/iamjohnnym/ python-projects/pyenv-tutorial
```

```
$ pyenv local 3.7.0
$ cat .python-version
```

Copy

```
Output
3.7.0
```

## Configuring the Environment File

At this point, we’re ready to create our `.envrc` file. With your favorite editor, create that file and add the following contents:

Copy

```
# check if python version is set in current dir
if [ -f ".python-version" ] ; then
    if [ ! -d ".venv" ] ; then
        echo "Installing virtualenv for $(python -V)"
        # if we didn't install `py2venv` for python 2.x, we would r
        # `virtualenv`, which you would have to install separately.
        python -m venv .venv
    fi
    echo "Activating $(python -V) virtualenv"
    source .venv/bin/activate
fi
```

```
# announce python version and show the path of the current python i
echo "Virtualenv has been activated for $(python -V)"
echo "$(which python)"
```

Save the file. If you did this via a shell editor, such as vim. You'll see the following message, `direnv: error .envrc is blocked. Run direnv allow to approve its content`. Don't be alarmed as this is a security feature to prevent auto-execution of the file. Whenever this file is changed, it requires manual approval before it will auto-execute again. To activate it, simply type `direnv allow` from the project dir.

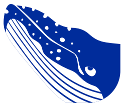
```
$ direnv allow
```

Copy

```
Output
direnv: loading .envrc
Installing virtualenv for Python 3.7.0
Activating Python 3.7.0 virtualenv
Virtualenv has been activated for Python 3.7.0
/Users/iamjohnnym/.personal/tutorials/pyenv-direnv/.venv/bin/python
direnv: export +VIRTUAL_ENV ~PATH
```

# Conclusion

You now have the tools you need to manage different python versions and project dependencies!



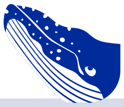
About the authors

Johnny M Author

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No



Comments

Leave a comment

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

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

MySQL

Docker

Kubernetes

All tutorials →

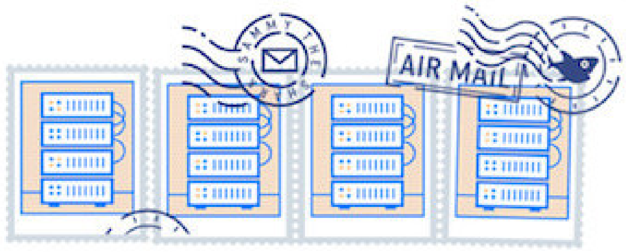
Free Managed Hosting →

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

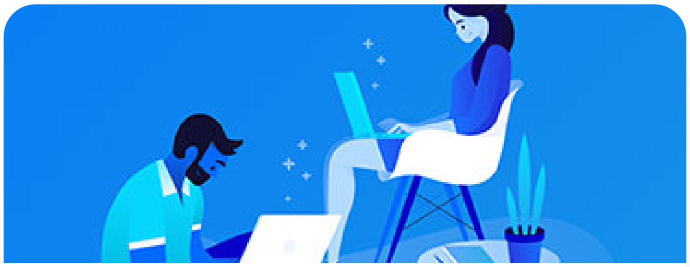
Sign up for Infrastructure as a Newsletter.

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 →

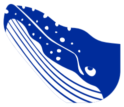
Featured on Community

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

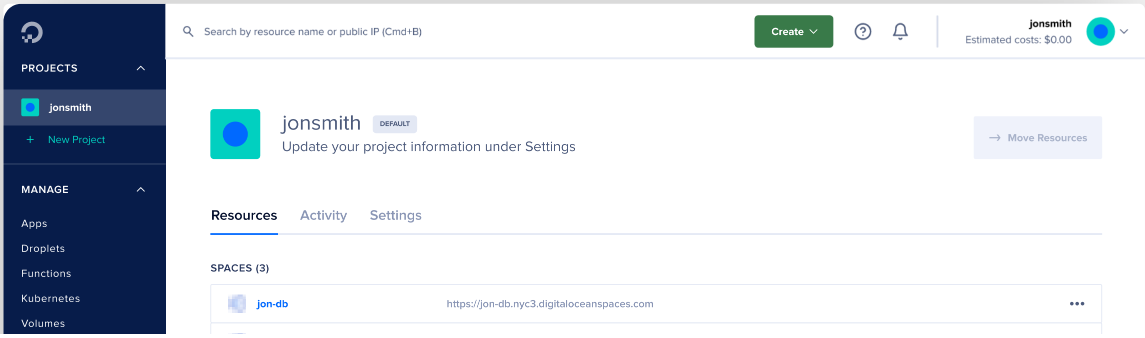
Welcome to the developer cloud



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.

COOKIE PREFERENCES

Learn more →



Company

About

Leadership

Blog

Careers

Customers

Partners

Channel Partners

Referral Program

Affiliate Program

Press

Legal

Security

Investor Relations

DO Impact

Products

Products Overview

Droplets

Kubernetes

App Platform

Functions

Cloudways

Managed Databases

Spaces

Marketplace

Load Balancers

Block Storage

Tools & Integrations

API

Pricing

Community

Tutorials

Q&A

CSS-Tricks

Write for DOnations

Currents Research

Hatch Startup Program

deploy by DigitalOcean

Shop Swag

Research Program

Open Source

Code of Conduct

Newsletter Signup

Solutions

Website Hosting

VPS Hosting

Web & Mobile Apps

Game Development

Streaming

VPN

SaaS Platforms

Cloud Hosting for Blockchain

Startup Resources

Contact

Support

Sales

Report Abuse

System Status

Share your ideas

COOKIE PREFERENCES

Documentation Meetups

Release Notes

Uptime