How To Set Up Python 2.7.6 and 3.3.3 on CentOS 6.4

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: http://faq.asphosthelpdesk.com/article.php?id=268

More often than not, as a developer you will be responsible for managing the server(s) your application runs on to a certain degree. When it comes to choosing the operating system, especially for production, going with a sound choice such as CentOS can be a tempting (safe) bet for the future.

When you get started with CentOS, however, you will be surprised to see that Python still is at version 2.6 (or even 2.4.3) and it should not be used by deployed applications regardless as it is reserved for system's use.

In this article, we are going to talk about downloading and setting up Python (2.7.6 and 3.3.3) without breaking the system's default 2.6 (or 2.4). It is rather important to not to get involved with that as critical system tools such as YUM depend on it. Furthermore, we will install two popular must-have Python companions pip and virtualenv.

After we are done, you will be able to simultaneously use either versions of Python on your CentOS 6.4 or 5.8, create and use virtual environments and finally, download and manage Python packages for each version.

CentOS and Its Design Choices

Before we get on with the installation, let's talk about CentOS.

Why does CentOS ship with older versions of applications?

CentOS is derived from RHEL (Red Hat Enterprise Linux). The target clientele for these two distributions consist of businesses, which require the systems to be up and running the most stable way for many years.

Therefore, the main reason here is the desire for *stability* for the system, achieved by supplying tested and more stable versions of applications. The philosophy behind "**if it ain't broke, don't fix it**" is heavily applied here.

Why do deployment libraries / applications need to be installed separately?

CentOS by default does not come with many tools, and the ones that are supplied are used by the system applications (e.g.***YUM***). Extreme care must be paid before changing or modifying them or their dependencies if you wish to keep your system running smoothly without breaking anything neither now or in future.

Do not assume the tools shipped with the operating system are for your use, and start making the habit of setting up all you need on your own.

Using this simple-to-follow tutorial, you will be able to use **any** version of Python and it will also teach you how to install almost any other application - from the source - as well.

Preparing The System and Installing Python

Like many other applications you will encounter, installation of Python on CentOS consists of a few (simple) stages, starting with updating the system and followed by actually getting Python for any desired version and proceeding with the set up process.

Remember: You can see all available releases of Python by checking out the *Releases* page. Using the instructions here, you should be able to install any or all of them.

Note: This guide should be valid for CentOS version 6.5 as well as 5.8 and 6.4.

Updating The Default CentOS Applications

Before we begin with the installation, let's make sure to update the default system applications to have the latest versions available.

Run the following command to *update* the system applications:

```
yum -y update
```

Preparing The System for Development Installations

CentOS distributions are lean - perhaps, a little *too* lean - meaning they do not come with many of the popular applications and tools that you are likely to need.

This is an intentional design choice. For our installations, however, we are going to need some libraries and tools (i.e.***development [related] tools***) not shipped by default. Therefore, we need to get them downloaded and installed before we continue.

There are two ways of getting the development tools on your system using the package manager yum:

Option #1 (not recommended) consists of downloading these tools (e.g. make, gcc etc.)*one-by-one*. It is followed by trying to develop something and highly-likely running into errors midway through - because you will have forgotten another package so you will switch back to downloading.

The recommended and sane way of doing this is following **Option #2**: simply downloading a bunch of tools by a single command with yum software groups.

YUM Software Groups

YUM Software Groups consist of bunch of commonly used tools (applications) bundled together, ready for download all at the same time via execution of a single command and stating a ***group name***. Using YUM, you can even download multiple groups together.

The group in question for us is the ***Development Tools***.

How to Install Development Tools using YUM on CentOS

—

In order to get the necessary development tools, run the following:

```
yum groupinstall -y development
```

or;

```
yum groupinstall -y 'development tools'
```

Note: The former (shorter) version might not work on older distributions of CentOS.

To download some additional packages which are handy:

```
yum install -y zlib-dev openssl-devel sqlite-devel bzip2-devel
```

Remember: Albeit optional, these "handy" tools are very much required for most of the tasks that you will come across in future. Unless they are installed in advance, Python, during compilation, will not be able to link to them.

Python Installation Procedure From Source

Setting up Python on our system will consist of 3 stages and 4 tools:

1. **Downloading** the compressed source code package (wget),
2. **Extracting** the files from this package (tar),
3. **Configuring** and **building** the application (autoconf (configure) / make).

GNU wget

GNU's "wgetâ€• is an application used to download files over various protocols (such as HTTP, FTP). Despite being missing from the older versions of CentOS, it now comes by default.

Example Usage for **wget**: `wget [URL]`

GNU Tar

GNU's Tar is basically a file archive creation and manipulation tool. Using various options available, it is possible to create compressed packages as well as extracting them at a later time.

Example Usage for **tar**: `tar [options] [arguments]`

GNU autoconf and GNU make

GNU *autoconf* and *make* are two different tools, (mostly) used together to configure the source code before building and installing applications.

We will:

- **Use ./configure**to configure everything before installation
- **Use make**to connect libraries and the source before
- **Using make install**- altinstall in our case - to build (compile) source code to create binary files and install the application on our system as configured using `./configure`.

To learn more about **autoconf**, consider reading its manual.

To learn more about **make**, consider reading its manual.

Downloading, Building (Compiling) and Installing Python

In this section, all the instructions given can be used to download any version of Python. You will just need to replace the version stated (which is "2.7.6â€• in the example below) with the version you require (e.g. "3.3.3â€•). You can install and use multiple versions at the same time. Although, you will need to specify their version during the execution (i.e. instead of python, you will need to use python2.7 or python3.3).

Downloading the Source Archive

Let's begin with retrieving the (compressed) archive containing Python source code. We will target `--version 2.7.6`.

```
wget http://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
```

Example for version 3.3.3:

```
wget http://www.python.org/ftp/python/3.3.3/Python-3.3.3.tar.xz
```

[Optional Step]XZtools:

This file is compressed using XZ library. Your system, depending on its version, might not have it. If that is the case, run the following to install XZ library:

```
yum install xz-libs
```

Extracting the Compressed Source Archive

This process consists of two steps: first decoding the XZ archive, followed by extracting the tar.

```
# Let's decode (-d) the XZ encoded tar archive:
xz -d Python-2.7.6.tar.xz
```

```
# Now we can perform the extraction:
tar -xvf Python-2.7.6.tar
```

Example for version 3.3.3:

```
xz -d Python-3.3.3.tar.xz
tar -xvf Python-3.3.3.tar
```

Configuring and Installation

—

Before building the source, we need to make sure that all the dependencies are there and prepare the environment. This is achieved automatically by using `./configure` to handle the task for us.

```
# Enter the file directory:
cd Python-2.7.6
# Start the configuration (setting the installation directory)
# By default files are installed in /usr/local.
# You can modify the --prefix to modify it (e.g. for $HOME).
./configure --prefix=/usr/local
```

Example for version 3.3.3:

```
cd Python-3.3.3
./configure
```

This procedure should execute without any hiccups - as we have downloaded all the necessary tools and applications. When it is complete, we will be ready to move on to the next step: **building and installing**.

Building and Installing

—

After configuring everything for the system we are working on, we can continue with **building** (compiling) the source and **installing** the application. Normally, one would use "make installâ€•; however, in order not to override system defaults - replacing the Python already used by the system - we will use`make altinstall`.

```
# Let's build (compile) the source
# This procedure can take awhile (~a few minutes)
make
# After building everything:
make altinstall
```

Example for version 3.3.3:

```
make && make altinstall   # <--- Two commands joint together
```

[Optional Step] Adding New Python Installation Location to PATH

Note: If you have followed the instructions using the default settings, you should not have the need to go through this section. However, if you have chosen a different *path* than **/usr/local** to install Python, you will need to perform the following to be able to run it without explicitly stating its full [installation] path *each time*.

Once the installation is complete, we can access the generated binaries (i.e. the Python interpreter for the version we have chosen) only by specifying its full location (path) (e.g. /usr/local/bin/python2.7) - unless of course the path exists already in the PATH variable (i.e. the variable which tells contains information on where to look for files stated).

If you would like to be able to access the newly installed Python interpreter without explicitly telling each and every time where to look for it, its*path*needs to be appended to`PATH`variable:

```
# Example: export PATH="[/path/to/installation]:$PATH"
export PATH="/usr/local/bin:$PATH"
```

To learn more about PATH, consider reading PATH definition at The Linux Information Project.

Setting Up Common Python Tools pip and virtualenv

Having installed Python, we can now finalize completing the basics for application production and deployment. For this, we will set up two of the most commonly used tools: **pip** package manager and **virtualenv** environment manager.

Installing pip on CentOS Using a New Python Installation

Before installing pip, we need to get its only external dependency -**setuptools**.

From the article on virtualenv and pip:

It [setuptools] builds on the (standard) functionality of Python's distribution utilities toolset called distutils. Given that distils is provided by default, all we need left is setuptools.

Execute the following commands to install setuptools:

This will install it for version 2.7.6

```
# Let's download the installation file using wget:
```

```
wget --no-check-certificate
https://pypi.python.org/packages/source/s/setuptools/setuptools-1.4.2.tar.gz
# Extract the files from the archive:
tar -xvf setuptools-1.4.2.tar.gz
# Enter the extracted directory:
cd setuptools-1.4.2
# Install setuptools using the Python we've installed (2.7.6)
python2.7 setup.py install
```

Installing pip itself is a very simple process afterwards. We will make use of the instructions from the article mentioned above to have it downloaded and installed automatically and securely using ***cURL*** library.

Let's download the setup files for*pip*and have Python (2.7) install it:

This will install it for version 2.7.6

```
curl https://raw.github.com/pypa/pip/master/contrib/get-pip.py | python2.7 -
```

Installing virtualenv on CentOS Using a New Python Installation

---

Now that we have pip the package manager ready, getting virtualenv on the system is a breeze.

Run the following command to download and install virtualenv:

```
pip install virtualenv
```