1)Update the `apt-get` package manager and upgrade any pre-installed packages:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

2)Install the required developer tools and packages:

```
$ sudo apt-get install build-essential cmake pkg-config
```

it is likely that `pkg-config` is already installed, but just in case it is not, be sure to include it in your `apt-get` command.

3)Install the necessary image I/O packages. These packages allow you to load various image file formats such as JPEG, PNG, TIFF, etc.

```
$ sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev
libpng12-dev
```

`4)` Install the GTK development library. This library is used to build Graphical User Interfaces (GUIs) and is required for the `highgui` library of OpenCV which allows you to view images on your screen:

```
$ sudo apt-get install libgtk2.0-dev
```

`5)` Install the necessary video I/O packages. These packages are used to load video files using OpenCV:

```
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
libv4l-dev
```

`6)` Install libraries that are used to optimize various operations within OpenCV:

```
$ sudo apt-get install libatlas-base-dev gfortran
```

`7)` Install Boost and Boost Python: Boost is a collection of peer-reviewed (i.e. very high quality) C++ libraries that help programmers and developers not get caught up in reinventing the wheel. Boost provides implementations for linear algebra, multithreading, basic image processing, and unit testing, just to name a few. We'll also install boost-python, which provides interoperability between the C++ and Python programming languages.

Both Boost and Boost Python will come in handy when we work with Approximate Nearest Neighbor search to build image search engines and utilize the dlib library:

```
$ sudo apt-get install libboost-all-dev
```

8) Install `pip`, a Python package manager:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

9) Install `virtualenv` and `virtualenvwrapper`, which allow us to create separate Python environments for each project we are working on. This is especially useful if you have projects that require different (or conflicting) versions of a given library:

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
```

Then, append your `~/.bashrc` file to include the following lines:

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

This will ensure that both `virtualenv` and `virtualenvwrapper` are loaded each time you login.

Reload the contents of your `~/.bashrc` file:

```
source ~/.bashrc
```

Create your `gurus` virtual environment:

```
mkvirtualenv gurus
```

10)

Now we can install the Python 2.7 development tools:

```
$ sudo apt-get install python2.7-dev
```

We also need to install NumPy since the OpenCV Python bindings represent images as multi-dimensional NumPy arrays:

```
$ pip install numpy
```

Let’s go ahead and take a second to install some other Python libraries that we’ll be using inside this course:

```
$ pip install scipy matplotlib
$ pip install scikit-learn
$ pip install -U scikit-image
$ pip install mahotas imutils Pillow commentjson
$ pip install pandas
$ pip install dlib
$ pip install face_recognition
$ pip install flask
$ pip install statistics
$ sudo apt-get install python-tk
```

11)

Download OpenCV and unpack it:

```
$ wget -O opencv-2.4.10.zip
http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/
opencv-2.4.10.zip/download
$ unzip opencv-2.4.10.zip
$ cd opencv-2.4.10
```

Setup the build:

```
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_NEW_PYTHON_SUPPORT=ON
-D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON  -D
BUILD_EXAMPLES=ON ..
```

Now we can finally compile OpenCV:

```
$ make -j4
```

Where the `4` can be replaced with the number of cores on your system which can speedup the compilation process.

And assuming that OpenCV compiled without error, you can now install it on your Ubuntu system:

```
$ sudo make install
$ sudo ldconfig
```

12)

If you've gotten this far in the article without error, OpenCV should now be installed in`/usr/local/lib/python2.7/site-packages`

But in order to utilize OpenCV within our `gurus` virtual environment, we first need to sym-link OpenCV into our site-packages directory:

```
$ cd ~/.virtualenvs/gurus/lib/python2.7/site-packages/
$ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
$ ln -s /usr/local/lib/python2.7/site-packages/cv.py cv.py
```

13) installing deep learning libraries

$ pip install pillow

$ pip install h5py

We also need to install Theano. You can certainly use pip to install Theano; however, it's important to note that Keras always uses the latest version of Theano, which is not always the version that is on PyPI. Thus, we should instead install Theano using the following command.Please come to root directory to continue to install the below dependicies:

$ pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git

An alternative method to install Theano is to clone the repository from GitHub and use the setup.py script:

$ git clone https://github.com/Theano/Theano

$ cd Theano

$ python setup.py install

From there, we can use pip to install Keras as well:

$ pip install keras

Step #3: Edit your keras.json configuration file

In order for the ~/.keras/keras.json file to be created you *first need to import keras* into your Python shell. If you *do not* import Keras, then the keras.json

file *will not* be created and you will be unable to find it on your disk. To perform this initial import of Keras simply open up a shell, (optionally) access the Python virtual environment (if you are using virtual environments), and then import Keras:

```
$ workon keras

$ python

>>> import keras

>>> quit()
```

If you see any errors importing Keras, that's okay — they are likely related to your configuration which we will resolve in the paragraphs directly below.

After performing the steps above you should now have a ~/.keras/keras.json file in your home directory. Open this file using your favorite text editor and inspect the contents. The default configuration will look something like this:

```
{
    "image_dim_ordering": "tf",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "tensorflow"
}
```

Looking at the configuration we can see that TensorFlow is being used as the default backend. However, since we are using *Theano* instead of *TensorFlow*, we need to update our keras.json file to reflect this (otherwise, you'll receive an import error regarding TensorFlow not being found when you try to use Keras).

First, we need to change image_dim_ordering from tf to th , indicating that we want to use Theano image dimension ordering rather than TensorFlow.

The second update is to change the backend from tensorflow to theano (I'll be covering how to use TensorFlow as a backend to Keras in a future blog post).

The updated keras.json file should look like this:

```
{
    "image_dim_ordering": "th",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "theano"
}
```

After making the required changes, save the keras.json configuration file and exit your editor.

For those curious about the keras.json file and how it is utilized to determine which backend to use (along with image dimension ordering), please refer to this page in the Keras documentation.

14) Finally, we can give our OpenCV and Python installation a test drive:

```
$ workon gurus
$ python
>>> import cv2
>>> cv2.__version__
'2.4.10'
```

If `cv2` imports without error, then we are all set!

However, let's not stop there. Let's write a Python script to load an image from disk and display it on our screen.

We'll start by downloading the PyImageSearch Gurus logo image:

```
$ cd ~
$ wget
https://gurus.pyimagesearch.com/wp-content/uploads/2015/03/pyimagesearc
h_gurus_logo.png
```

This places the `pyimagesearch_gurus_logo.png` image in our home directory.

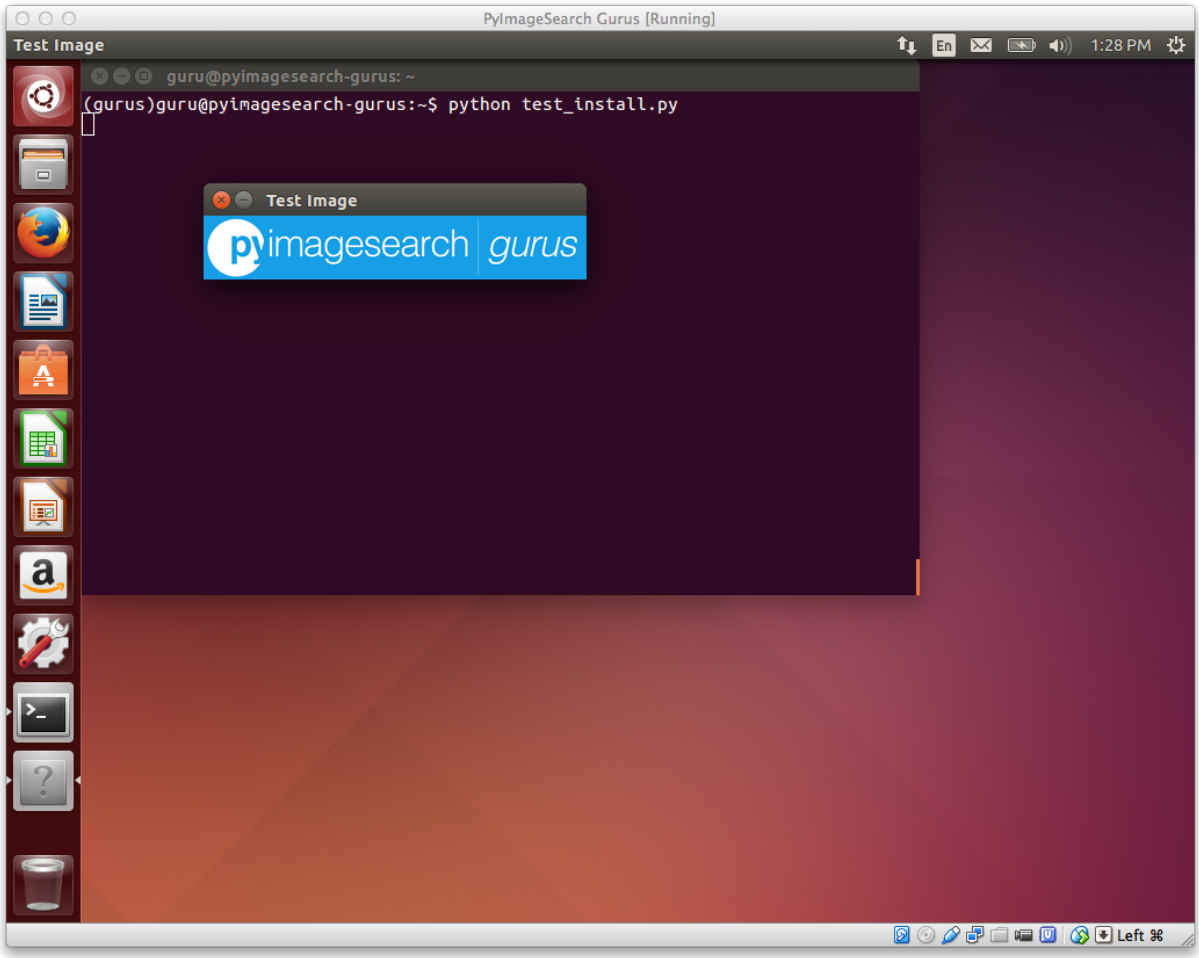
Now let's create a new file, name it `test_install.py`, and insert the following code:

```
import cv2
image = cv2.imread("pyimagesearch_gurus_logo.png")
cv2.imshow("Test Image", image)
cv2.waitKey(0)
```

Save the file, exit, and then execute the following command:

```
$ python test_install.py
```

And if all goes well you should see the PyImageSearch Gurus logo image displayed on your screen:


PyImageSearch Gurus [Running]
Test Image
1:28 PM
guru@pyimagesearch-gurus: ~
(gurus)guru@pyimagesearch-gurus:~$ python test_install.py
Test Image
pyimagesearch gurus
Left ⌘