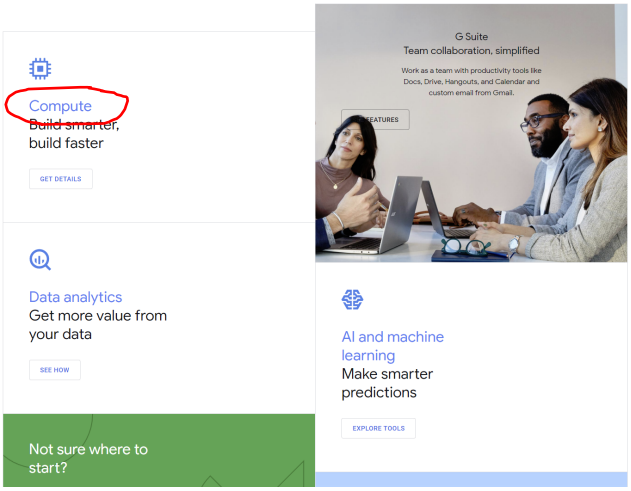
For the project and assignments with huge size input data, we offer an option to use Google Cloud Platform (GCP) for developing and testing your implementations. This tutorial lists the necessary steps of working on the assignments using Google Cloud. This tutorial also helps you install GUI on GCP to make your development more convenient. It also includes the installation of some necessary libraries such as OpenCV, TensorFlow and Keras. If you don't need them, just ignore that part.

This tutorial goes through how to set up your own Google Compute Engine (GCE) instance. When you sign up for the first time, you will receive $300 credits from Google by default.

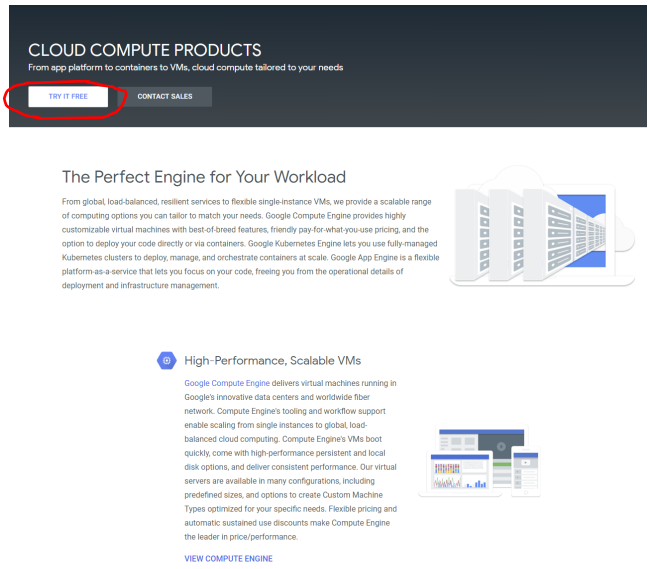
Part I: Setup your computer on the cloud

1. First, if you don't have a Google Cloud account already, create one by going to the Google Cloud homepage and clicking on **Compute**.
   https://cloud.google.com/products/compute/

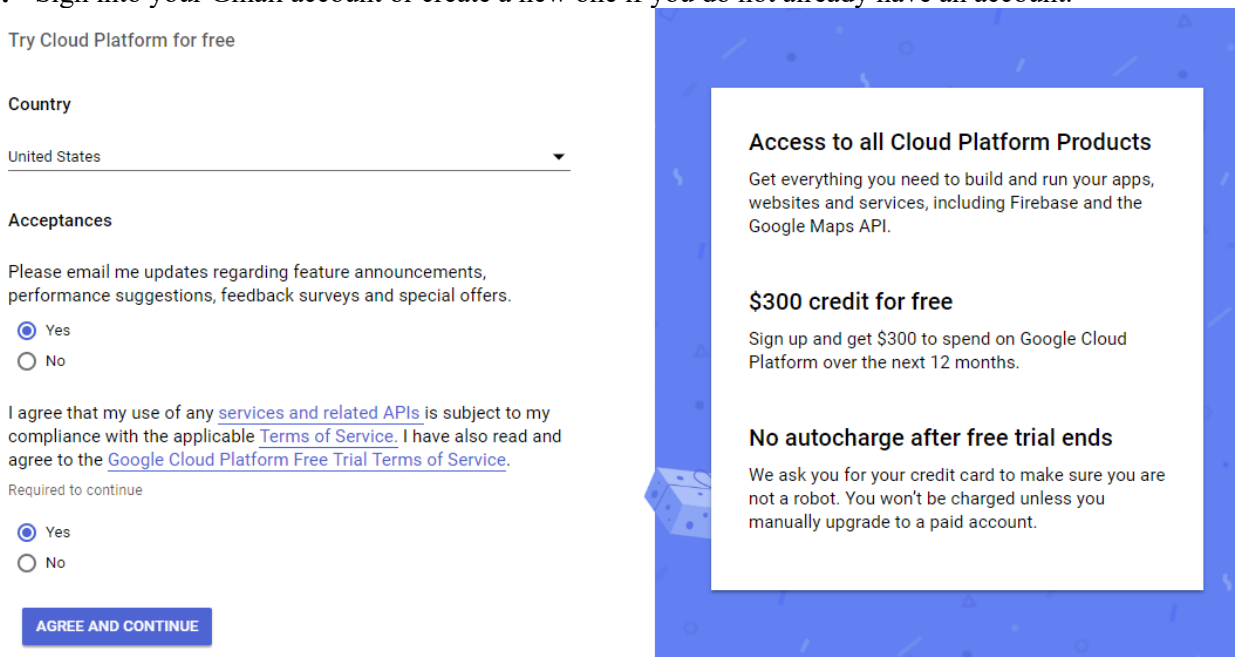
GCP Free Tier



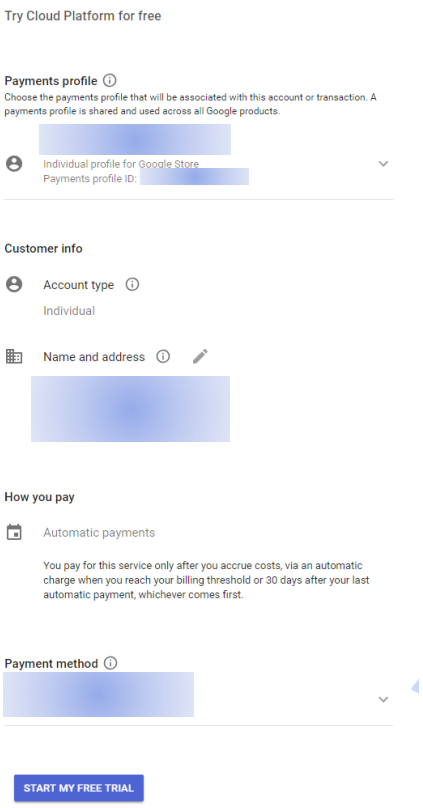
2. When you get to the next page, click on the blue **TRY IT FREE** button.



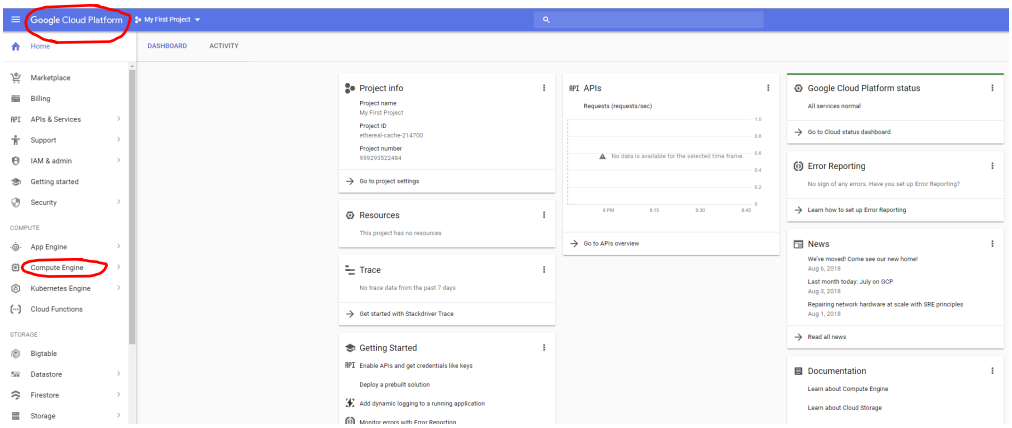
3. Sign into your Gmail account or create a new one if you do not already have an account.



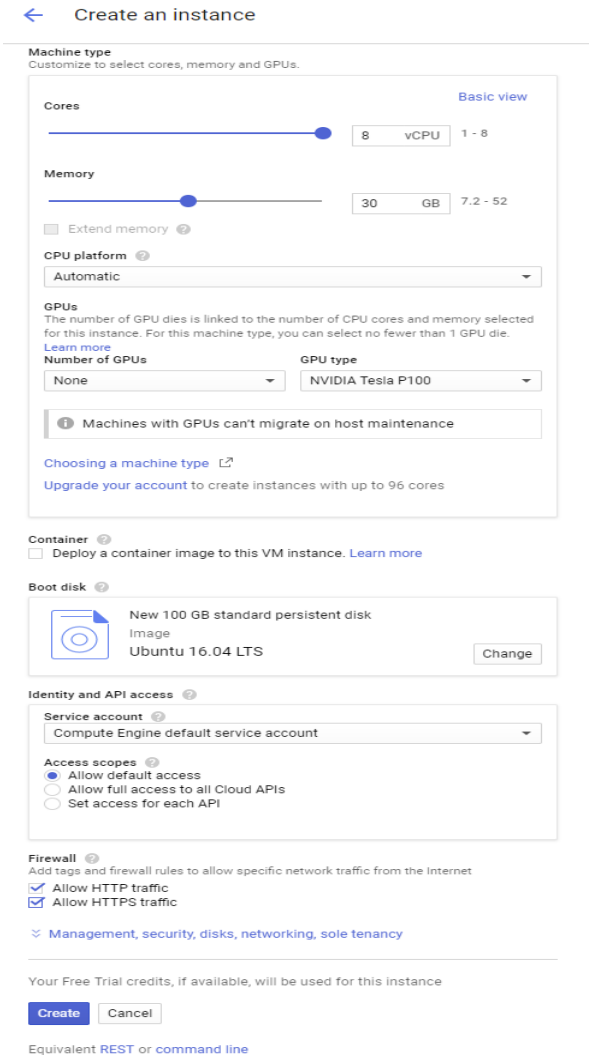
Click the appropriate **yes** or **no** button for the first option and check **yes** for the second option after you have read the required agreements. Press the blue **Agree and continue** button to continue to the next page to enter the requested information (your name, billing address and credit card information). Remember to select "**Individual**" as "Account Type":



4. Once you have entered the required information, press the blue **Start my free trial** button. Press the "Google Cloud Platform" and it will take you to the main dashboard:



5. To launch a virtual instance, go to the **Compute Engine** menu on the left column of your dashboard and click on **VM instances**. Then click on the blue **Create** button on the next page. This will take you to a page that looks like the screenshot below. **(NOTE: Please carefully read the instructions in addition to looking at the screenshots. The instructions tell you exactly what values to fill in).**



Give your instance a cool name! (I use "nn4cv".)

zone: choose us-west1-b (this zone has good GPUs. You will need one for your final project.)

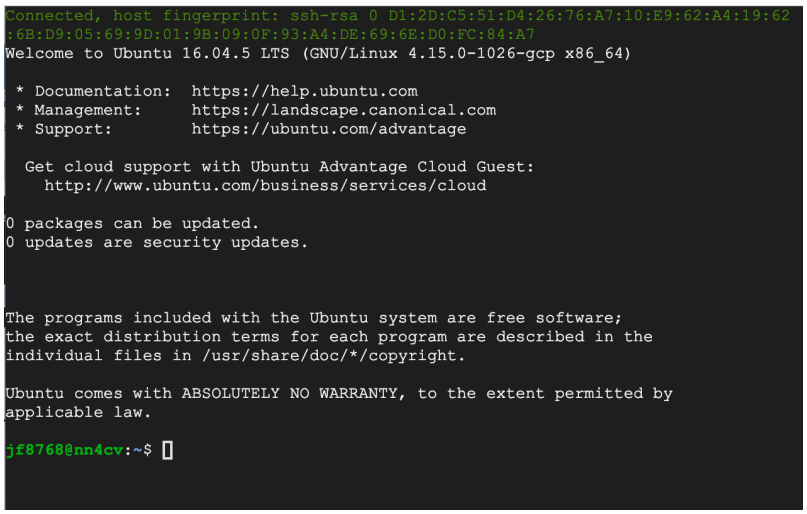
Machine Type: click "customize". I select a 8-core CPU and 30GB memory.

Boot Disk: Ubuntu 16.04 LTS (this is the base image your instance will run on). If you prefer 17.04 or 18.04, you can try. Choose standard persistent disk and choose the size of memory you need. Specify 100GB.

Firewall: allow HTTP/HTTPS traffic. Click "create".

6. Click on "Open in browser window"

Now you can use your cloud computer!



Part II: Graphical user interface (GUI) for Google Compute Engine instance

If you prefer to use command line only. Please skip this part.

1. In the terminal, type and execute the following commands one by one.

   sudo apt-get update
   sudo apt-get install ubuntu-desktop
   sudo apt-get install gnome-panel gnome-settings-daemon metacity nautilus gnome-terminal
   vnc4server
   sudo ufw allow 5901:5910/tcp
   vncserver

   Start the vnc server, you'll then be prompted to create and verify a new password.
   Note: *this password will grant access to your instance*

   If everything went fine your VNC server is now running and listening on port 5901. You can verify this with netcat from the Google Compute Engine instance:
   Type command: nc localhost 5901
   then you will see: *RFB 003.008*

*Ctrl+D to quit

2. We now need to kill the session we just created and make a tweak to the startup script for VNCServer to make it work properly. If we don't perform this step then all we will see is a grey cross-hatched screen with an "X" cursor and/or a grey screen with a Terminal Session, depending on the Ubuntu version. Not very useful!

So, type the following command to kill the session:

vncserver -kill :1

Now open the file we need to edit:

vim .vnc/xstartup

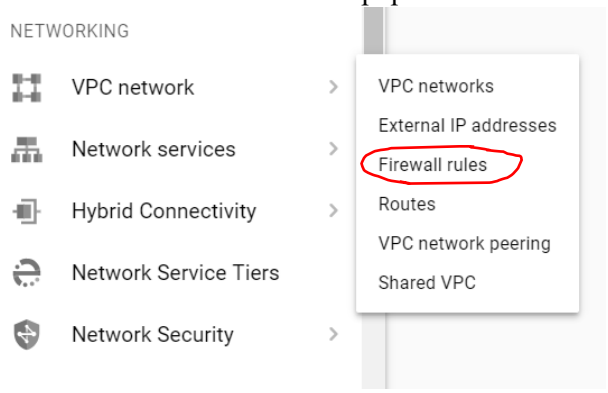
Press the [Insert] key (“i” in Ubuntu) once (this will switch us into “edit” mode) and then edit the script so it ends up looking like this:

```
#!/bin/sh

# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
x-window-manager &

gnome-panel &
gnome-settings-daemon &
metacity &
nautilus &
```

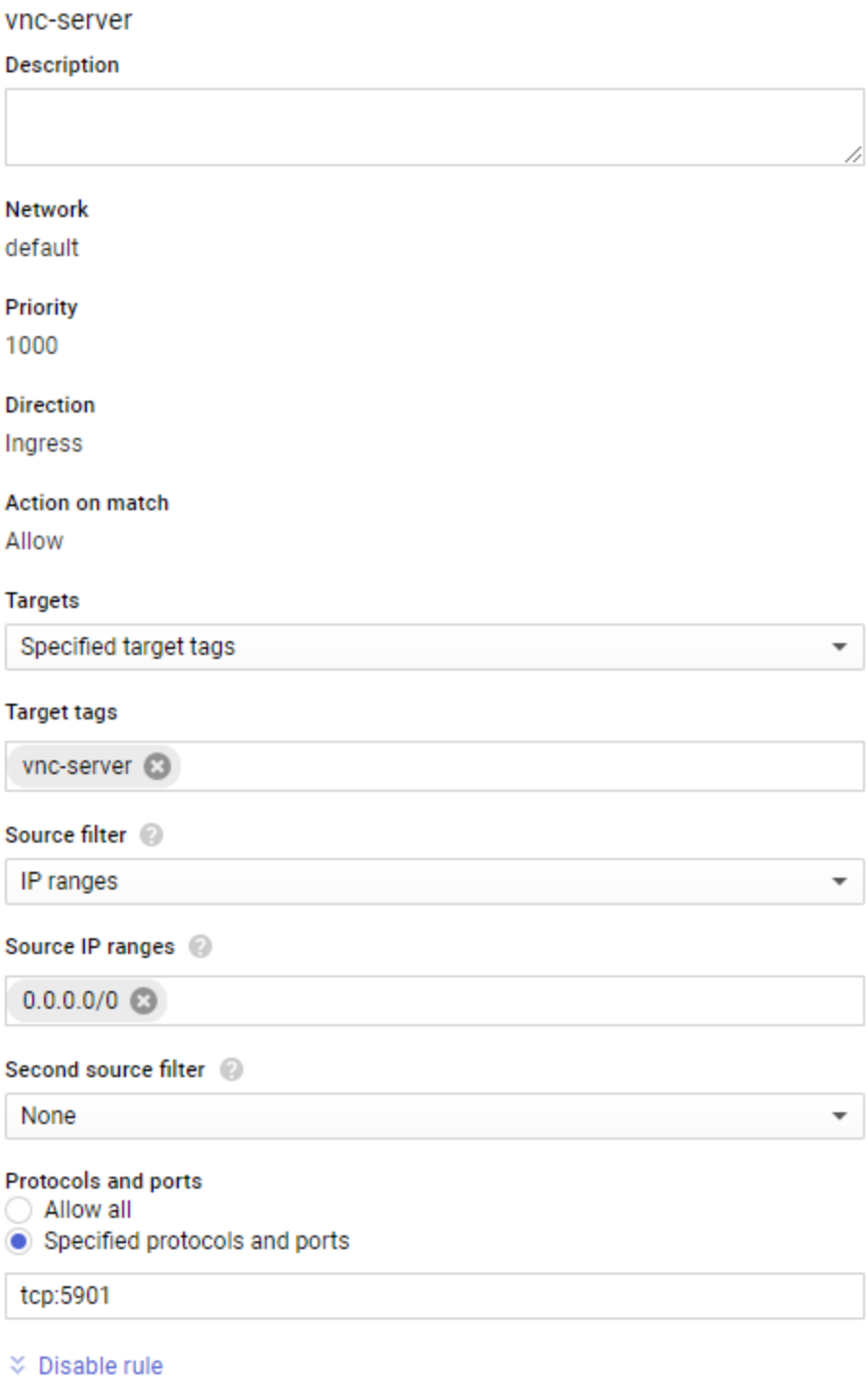
When you’re done editing the .vnc/xstartup file, press the [Esc] key once and type the following to save the changes and bring you back to the command line:

:wq

3. From the Google Cloud web console, go to the top left menu, scroll down to VPC Networks, and select Firewall Rules from the pop-out menu.



4. From the Firewall rules menu, click CREATE FIREWALL RULE from the top. For our scenario, we want to allow access for TCP port 5901 only to instances with the tag ‘vnc-server’ without exposing the rest of our network to the same port. We want to be able to remote to our desktop from any public computer. Therefore our settings are in the below screenshot, and also as follows:



5. Once this is done, click **Create**. Our firewall rule is created, but we still cannot remote to our Linux desktop because our "nn4cv" instance does not have the network tag we specified as our firewall rule target. Let's fix that now.

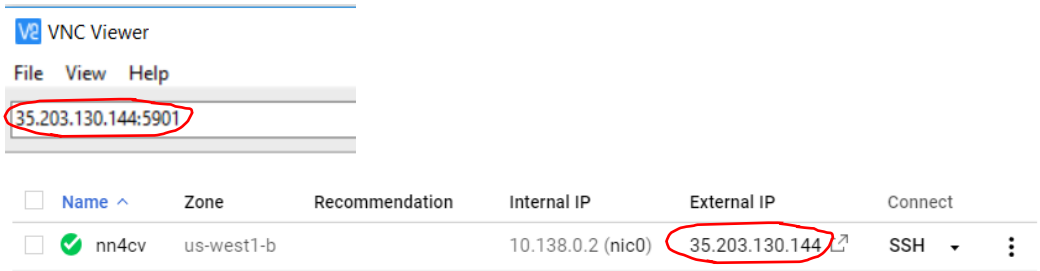
Go to Compute Engine and click on our instance 'nn4cv', then click Edit.

Under Network tags, type the name of our network tag to match the firewall target. In this case, ‘vnc-server’. Then scroll to the bottom to and click Save to save changes.

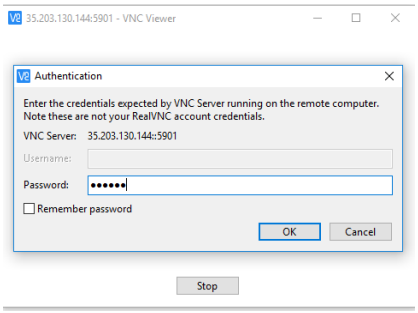
6. Type vncserver to start a new session

7. Now install a VNC client on your local machine. There are many options available (TightVNC, RealVNC etc.). Install any one. Here I use RealVNC.

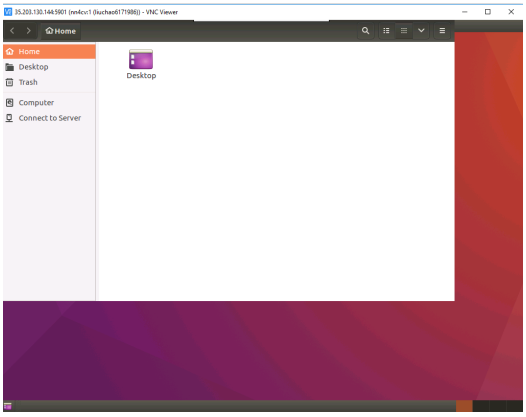
8. Launch your VNC Viewer and type in your own external IP (it should be different every time).



Type in your password



Now you see your desktop!



Part III: Configuring Ubuntu for deep learning with Python

Python3, OpenCV 3.3,Tensorflow, Keras, etc

Step #1: Install Ubuntu system dependencies

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

sudo apt-get install build-essential cmake git unzip pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk-3-dev
sudo apt-get install libhdf5-serial-dev graphviz
sudo apt-get install libopenblas-dev libatlas-base-dev gfortran
sudo apt-get install python-tk python3-tk python-imaging-tk

sudo apt-get install python2.7-dev python3-dev
```

Step #2: Create your Python virtual environment

In this section we will setup a Python virtual environment on your system.

Installing pip

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
sudo python3 get-pip.py
```

Installing virtualenv and virtualenvwrapper

These Python packages allow you to create ***separate, independent*** Python environments for ***each*** project that you are working on.

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

Once we have virtualenv and virtualenvwrapper installed, we need to update our~/.bashrc file to include the following lines at the *bottom* of the file:

```
export WORKON_HOME=$HOME/.virtualenvs
```

```
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh


echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.bashrc
echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.bashrc
echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.bashrc
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc
```

After editing our ~/.bashrc file, we need to reload the changes:

```
source ~/.bashrc
```

Note: *Calling source on ~/.bashrc only has to be done* ***once*** *for our current shell session. Anytime we open up a new terminal, the contents of ~/.bashrc will be* ***automatically*** *executed (including our updates).*

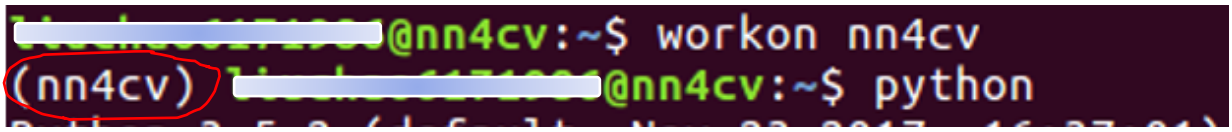
Creating a virtual environment for deep learning and computer vision

```
mkvirtualenv nn4cv -p python3
```

Verifying that you are in the "nn4cv" virtual environment

If you ever reboot your Ubuntu system; log out and log back in; or open up a new terminal, you'll need to use the workon command to re-access your "nn4cv" virtual environment.

```
workon nn4cv
```



Installing NumPy

To install NumPy, ensure you are in the nn4cv virtual environment (otherwise NumPy will be installed into the *system* version of Python rather than the nn4cv environment).
From there execute the following command:

```
pip install numpy
```

Step #3: Compile and Install OpenCV

```
cd ~
```

```
wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip
unzip opencv.zip
unzip opencv_contrib.zip
cd ~/opencv-3.3.0/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D WITH_CUDA=OFF \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
    -D BUILD_EXAMPLES=ON ..
```

Compiling OpenCV

```
make -j4
sudo make install
sudo ldconfig
cd ~
rm -rf opencv-3.3.0 opencv.zip
rm -rf opencv_contrib-3.3.0 opencv_contrib.zip
```

Symbolic linking OpenCV to your virtual environment

```
cd ~/.virtualenvs/nn4cv/lib/python3.5/site-packages/
ln -s /usr/local/lib/python3.5/site-packages/cv2.cpython-35m-x86_64-linux-gnu.so cv2.so
cd ~
```

Testing your OpenCV 3.3 install

```
python
>>> import cv2
>>> cv2.__version__
'3.3.0'
```

Step #4: Install Keras

For this step, be sure that you are in the nn4cv environment by issuing the workon nn4cv command. Then install our various Python computer vision, image processing, and machine learning libraries:

```
pip install scipy matplotlib pillow
pip install imutils h5py requests progressbar2
pip install scikit-learn scikit-image
```

Next, install Tensorflow (CPU version) and Keras:

```
pip install tensorflow
pip install keras
```

You can test our Keras install from a Python shell: