

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.

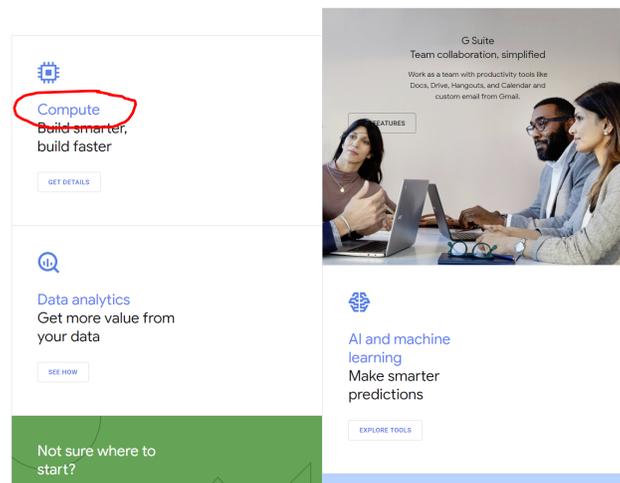
This tutorial goes through how to set up your own Google Compute Engine (GCE) instance. When you sign up for the first time, you will receive \$300 credits from Google by default.

## Part I: Setup your computer on the cloud

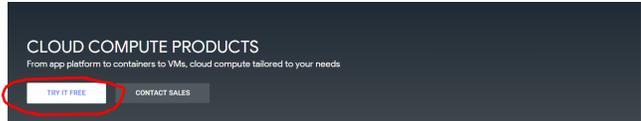
1. First, if you don't have a Google Cloud account already, create one by going to the [Google Cloud homepage](https://cloud.google.com/products/compute/) and clicking on **Compute**.

<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.



### The Perfect Engine for Your Workload

From global, load-balanced, resilient services to flexible single-instance VMs, we provide a scalable range of computing options you can tailor to match your needs. Google Compute Engine provides highly customizable virtual machines with best-of-breed features, friendly pay-for-what-you-use pricing, and the option to deploy your code directly or via containers. Google Kubernetes Engine lets you use fully-managed Kubernetes clusters to deploy, manage, and orchestrate containers at scale. Google App Engine is a flexible platform-as-a-service that lets you focus on your code, freeing you from the operational details of deployment and infrastructure management.



#### High-Performance, Scalable VMs

Google Compute Engine delivers virtual machines running in Google's innovative data centers and worldwide fiber network. Compute Engine's tooling and workflow support enable scaling from single instances to global, load-balanced cloud computing. Compute Engine's VMs boot quickly, come with high-performance persistent and local disk options, and deliver consistent performance. Our virtual servers are available in many configurations, including predefined sizes, and options to create Custom Machine Types optimized for your specific needs. Flexible pricing and automatic sustained use discounts make Compute Engine the leader in price/performance.



[VIEW COMPUTE ENGINE](#)

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

Try Cloud Platform for free

Country

United States

Acceptances

Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.

- Yes
- No

I agree that my use of any [services and related APIs](#) is subject to my compliance with the applicable [Terms of Service](#). I have also read and agree to the [Google Cloud Platform Free Trial Terms of Service](#).

Required to continue

- Yes
- No

**AGREE AND CONTINUE**

### Access to all Cloud Platform Products

Get everything you need to build and run your apps, websites and services, including Firebase and the Google Maps API.

### \$300 credit for free

Sign up and get \$300 to spend on Google Cloud Platform over the next 12 months.

### No autocharge after free trial ends

We ask you for your credit card to make sure you are not a robot. You won't be charged unless you manually upgrade to a paid 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”:

## Try Cloud Platform for free

### Payments profile ⓘ

Choose the payments profile that will be associated with this account or transaction. A payments profile is shared and used across all Google products.

Individual profile for Google Store  
Payments profile ID: [REDACTED]

### Customer info

#### Account type ⓘ

Individual

#### Name and address ⓘ ✎

[REDACTED]

### How you pay

#### Automatic payments

You pay for this service only after you accrue costs, via an automatic charge when you reach your billing threshold or 30 days after your last automatic payment, whichever comes first.

### Payment method ⓘ

[REDACTED]

[START MY FREE TRIAL](#)

- 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:

The screenshot shows the Google Cloud Platform dashboard for a new project. The top navigation bar includes the Google Cloud Platform logo (circled in red) and the project name 'My First Project'. The left-hand navigation menu lists various services, with 'Compute Engine' (circled in red) highlighted. The main content area is divided into several sections: Project info (Project name: My First Project, Project ID: ef4eaa0-cd4bc-214700, Project number: 95923522484), Resources (This project has no resources), Trace (No trace data from the past 7 days), API APIs (Requests (requests/sec) chart showing 0.0, 0.0, 0.0, 0.0, 0.0), Google Cloud Platform status (All services normal), Error Reporting (No sign of any errors), News (We've moved! Come see our new home!), and Documentation (Learn about Compute Engine, Learn about Cloud Storage).

- 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.**)

← Create an instance

**Machine type**  
Customize to select cores, memory and GPUs.

**Cores** Basic view

8 vCPU 1 - 8

**Memory**

30 GB 7.2 - 52

Extend memory

**CPU platform**

Automatic

**GPUs**  
The number of GPU dies is linked to the number of CPU cores and memory selected for this instance. For this machine type, you can select no fewer than 1 GPU die. [Learn more](#)

**Number of GPUs** **GPU type**

None NVIDIA Tesla P100

**Machines with GPUs can't migrate on host maintenance**

[Choosing a machine type](#)

[Upgrade your account](#) to create instances with up to 96 cores

---

**Container**

Deploy a container image to this VM instance. [Learn more](#)

**Boot disk**

New 100 GB standard persistent disk  
Image  
Ubuntu 16.04 LTS Change

**Identity and API access**

**Service account**

Compute Engine default service account

**Access scopes**

Allow default access  
 Allow full access to all Cloud APIs  
 Set access for each API

**Firewall**

Add tags and firewall rules to allow specific network traffic from the Internet

Allow HTTP traffic  
 Allow HTTPS traffic

[Management, security, disks, networking, sole tenancy](#)

---

Your Free Trial credits, if available, will be used for this instance

Create Cancel

Equivalent [REST](#) or [command line](#)

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”.

- Click on “Open in browser window”

<input type="checkbox"/>	Name ^	Zone	Recommendation	Internal IP	External IP	Connect
<input checked="" type="checkbox"/>	nn4cv	us-west1-b		10.138.0.2 (nic0)	104.198.15.103 ↗	SSH ▾ ⋮

- Open in browser window
- Open in browser window on custom port
- View gcloud command
- Use another SSH client

Now you can use your cloud computer!

```
Connected, host fingerprint: ssh-rsa 0 D1:2D:C5:51:D4:26:76:A7:10:E9:62:A4:19:62:6B:D9:05:69:9D:01:9B:09:0F:93:A4:DE:69:6E:D0:FC:84:A7
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-1026-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

j#8768@nn4cv:~$ █
```

## 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

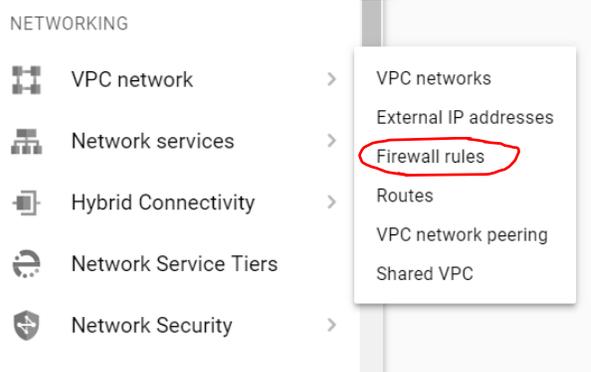
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
x-window-manager &

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:

vnc-server

Description

Network

default

Priority

1000

Direction

Ingress

Action on match

Allow

Targets

Target tags

Source filter ?

Source IP ranges ?

Second source filter ?

Protocols and ports

Allow all

Specified protocols and ports

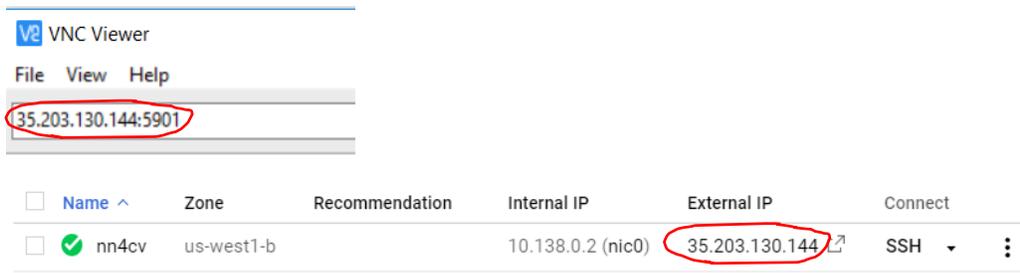
[Disable rule](#)

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.

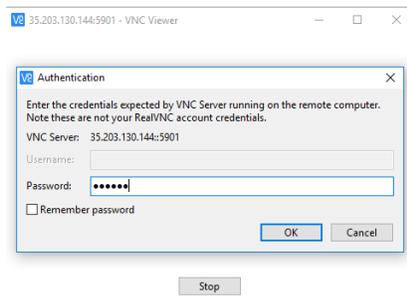
Go to Compute Engine and click on our instance ‘nn4cv’, then click Edit.

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.

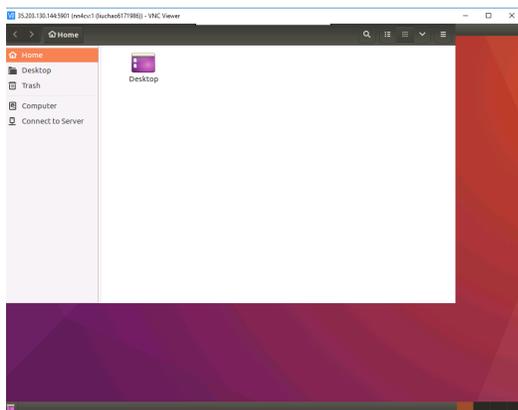
6. Type `vncserver` to start a new session
7. Now install a VNC client on your local machine. There are many options available ( [TightVNC](#), [RealVNC](#) etc. ). Install any one. Here I use RealVNC.
8. Launch your VNC Viewer and type in your own external IP (it should be different every time).



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).*

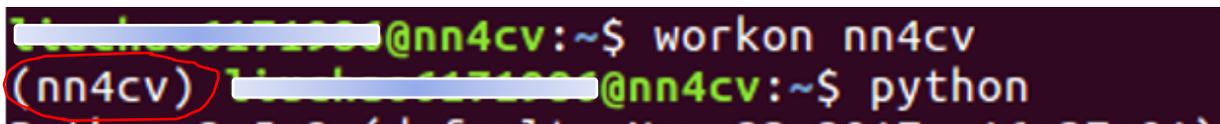
## Creating a virtual environment for deep learning and computer vision

```
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:

```
35.203.130.144:5901 (nn4cv:1 (liuchao6171986)) - VNC Viewer
File Edit View Search Terminal Help
[redacted]@nn4cv:~$ python
Python 2.7.12 (default, Dec 4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
[redacted]@nn4cv:~$ workon nn4cv
(nn4cv) [redacted]@nn4cv:~$ python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>> █
```