Tutorial: How to Configure the Development Environment on Google Cloud Platform

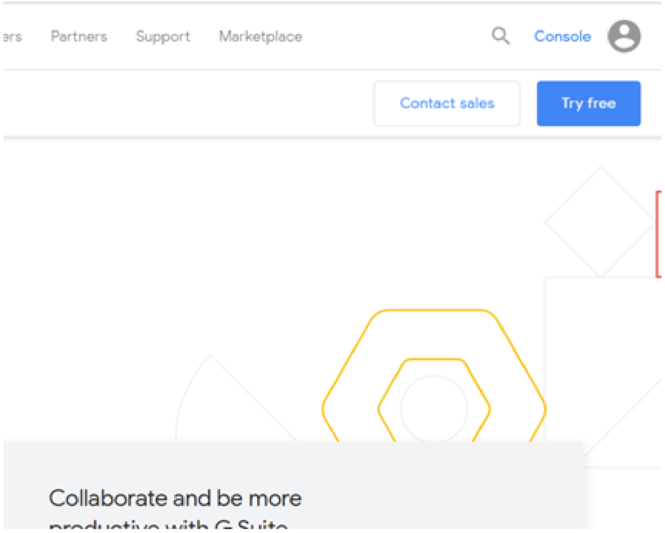
This tutorial is to teach how to configure a development environment for the GPU training of Keras neural networks with MXNet as backend on Google Cloud Platform.

Contents

Start an Instance on Google Cloud Platform

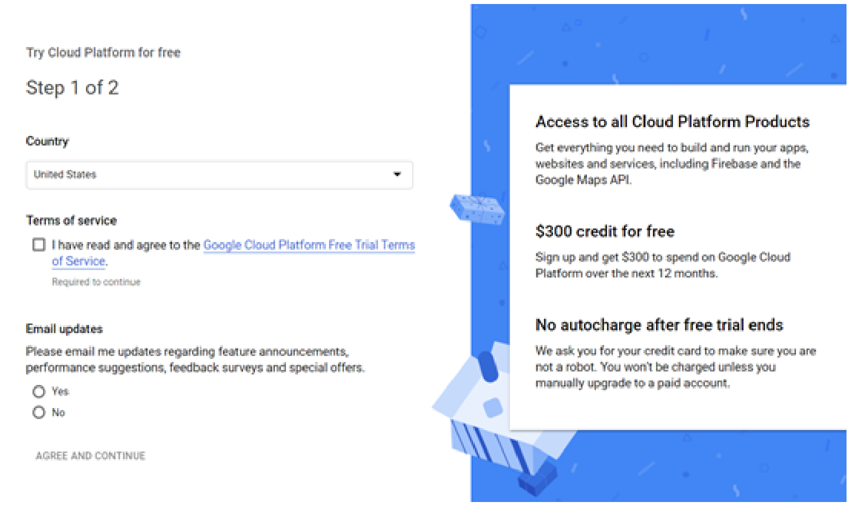
Create a Google Cloud account

Enter https://cloud.google.com/ Click "Try free"



Collaborate and be more productive with G Suite

Fill in billing information, Google Cloud will provide $300 credit for free



Try Cloud Platform for free

Step 1 of 2

Country

United States

Terms of service

□ I have read and agree to the Google Cloud Platform Free Trial Terms of Service.

Required to continue

Email updates

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

○ 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.

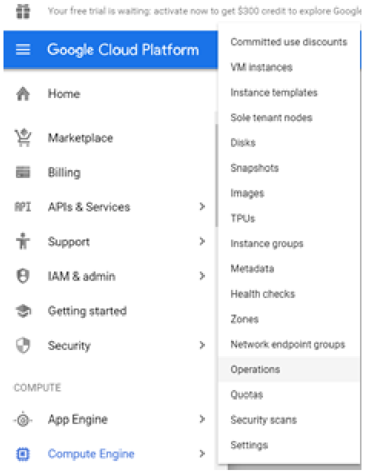
No autocharge after free trial ends

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.

Then we can enter console

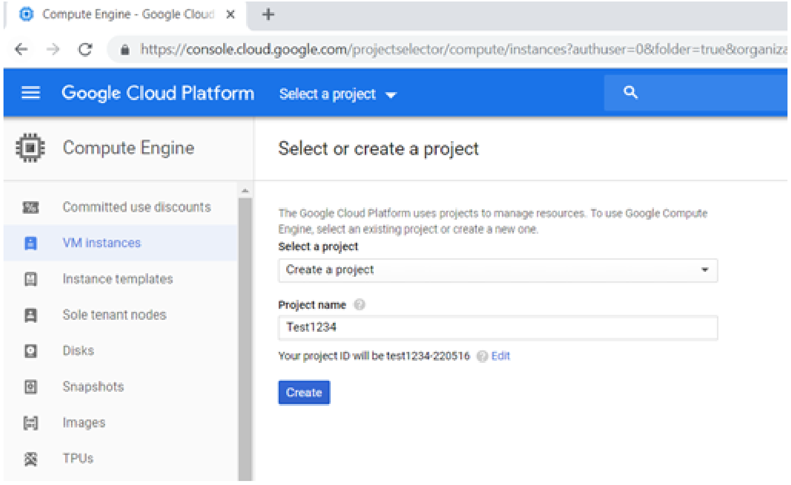
Create a new VM instance

Click Compute Engine on the left tab



Your free trial is waiting: activate now to get $300 credit to explore Googl

Create a new project, Google Cloud will automatically generate a project ID for us



Select or create a project

The Google Cloud Platform uses projects to manage resources. To use Google Compute Engine, select an existing project or create a new one.

Select a project

Create a project

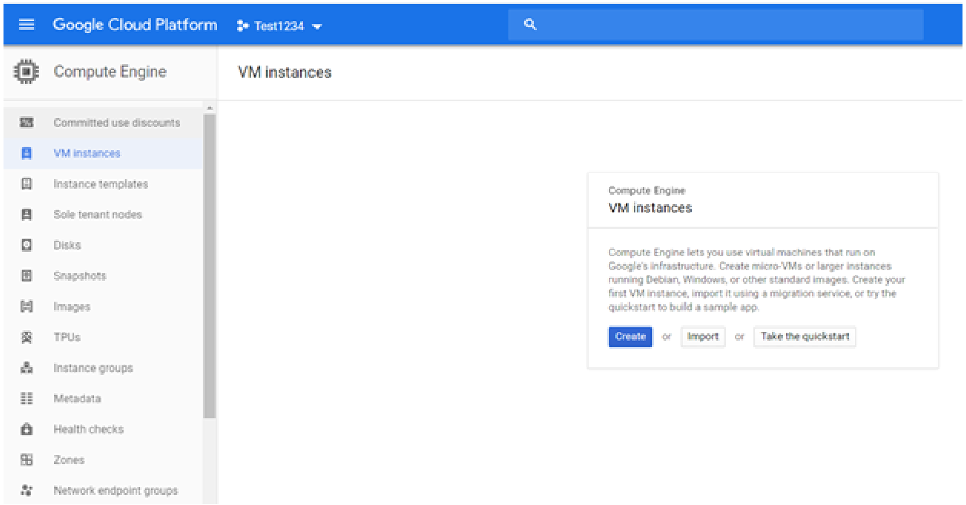
Project name

Test1234

Your project ID will be test1234-220516 Edit

Create

Then create a new VM instance

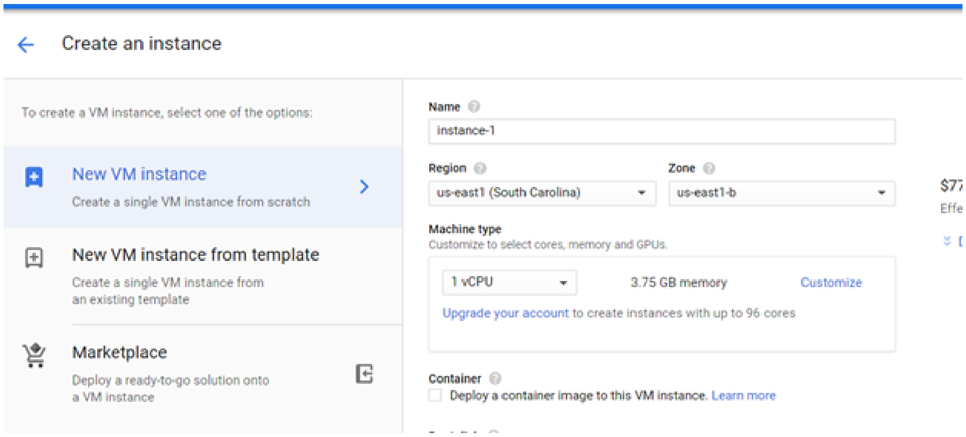


Compute Engine

VM instances

Compute Engine lets you use virtual machines that run on Google's infrastructure. Create micro-VMs or larger instances running Debian, Windows, or other standard images. Create your first VM instance, import it using a migration service, or try the quickstart to build a sample app.

We need change some options here. First, select a proper Region and Zone



Create an instance

To create a VM instance, select one of the options:

New VM instance
Create a single VM instance from scratch

New VM instance from template
Create a single VM instance from an existing template

Marketplace
Deploy a ready-to-go solution onto a VM instance

Name: instance-1

Region: us-east1 (South Carolina)

Zone: us-east1-b

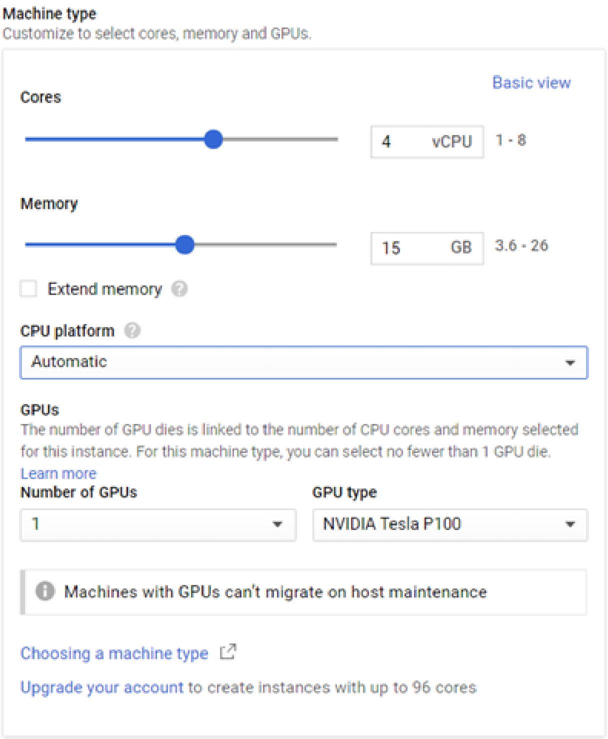
Machine type
Customize to select cores, memory and GPUs.

1 vCPU 3.75 GB memory

Upgrade your account to create instances with up to 96 cores

Container
□ Deploy a container image to this VM instance. Learn more

We can customize the machine type, here we choose 4 vCPU and 1 NVIDA Tesla P100 GPU



Machine type
Customize to select cores, memory and GPUs.

Cores: 4 vCPU

Memory: 15 GB

□ 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.

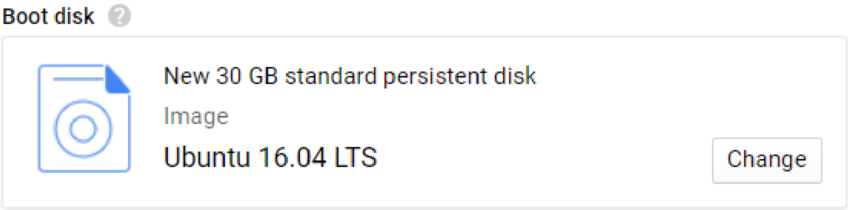
Number of GPUs: 1

GPU type: NVIDIA Tesla P100

Machines with GPUs can't migrate on host maintenance

Upgrade your account to create instances with up to 96 cores

We can select boot disk. Since we need install several large package later, here we'd like to have a large disk. Here we choose Ubuntu 16.04 LTS, 30GB

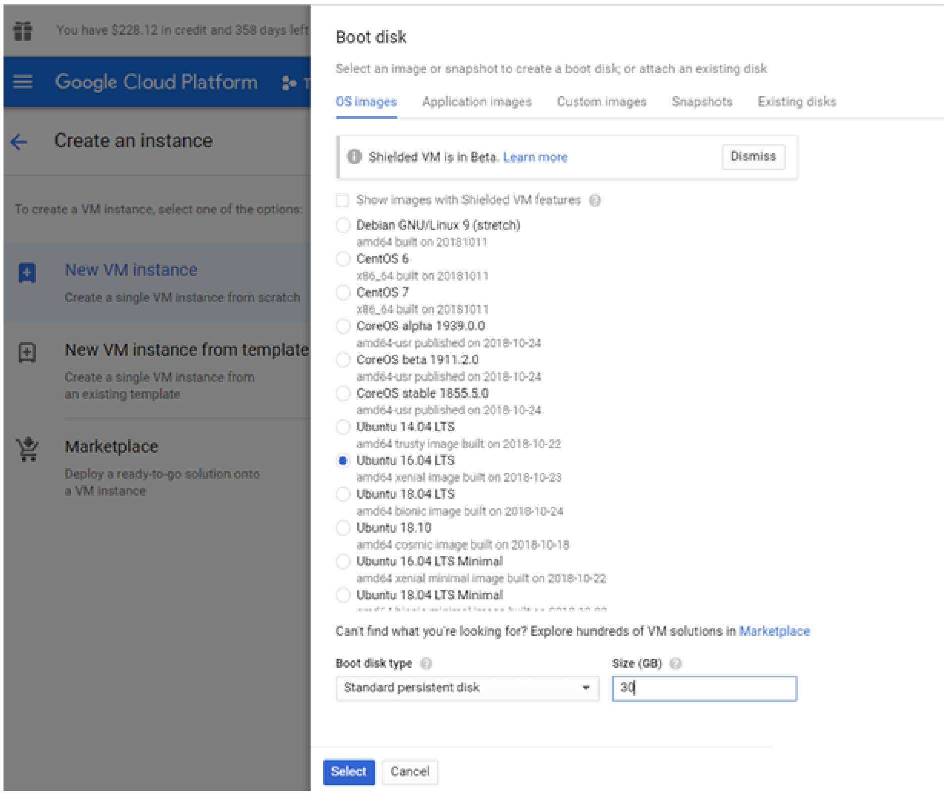


Boot disk

New 30 GB standard persistent disk

Image

Ubuntu 16.04 LTS



You have $228.12 in credit and 358 days left

Create an instance

To create a VM instance, select one of the options:

New VM instance
Create a single VM instance from scratch

New VM instance from template
Create a single VM instance from an existing template

Marketplace
Deploy a ready-to-go solution onto a VM instance

Boot disk

Select an image or snapshot to create a boot disk; or attach an existing disk

Shielded VM is in Beta. Learn more

□ Show images with Shielded VM features

○ Debian GNU/Linux 9 (stretch)
amd64 built on 20181011

○ CentOS 6
x86_64 built on 20181011

○ CentOS 7
x86_64 built on 20181011

○ CoreOS alpha 1939.0.0
amd64-usr published on 2018-10-24

○ CoreOS beta 1911.2.0
amd64-usr published on 2018-10-24

○ CoreOS stable 1855.5.0
amd64-usr published on 2018-10-24

○ Ubuntu 14.04 LTS
amd64 trusty image built on 2018-10-22

● Ubuntu 16.04 LTS
amd64 xenial image built on 2018-10-23

○ Ubuntu 18.04 LTS
amd64 bionic image built on 2018-10-24

○ Ubuntu 18.10
amd64 cosmic image built on 2018-10-18

○ Ubuntu 16.04 LTS Minimal
amd64 xenial minimal image built on 2018-10-22

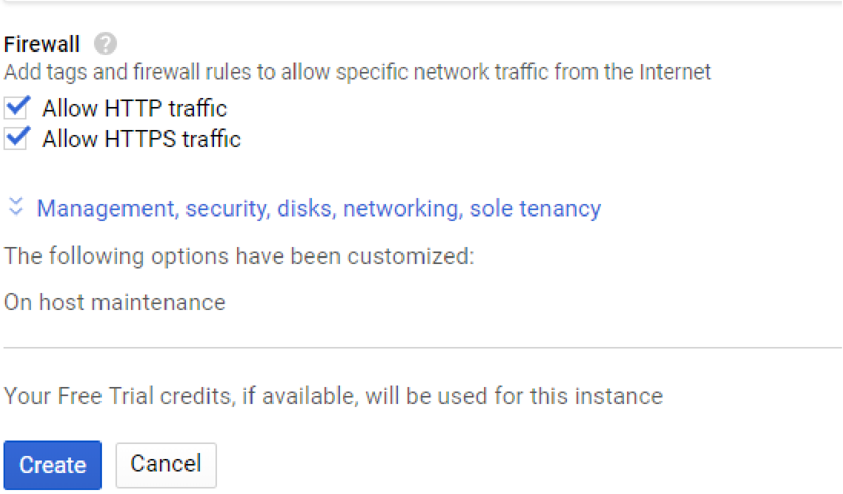
○ Ubuntu 18.04 LTS Minimal

Can't find what you're looking for? Explore hundreds of VM solutions in Marketplace

Boot disk type: Standard persistent disk

Size (GB): 30

Select "Allow HTTP traffic" and "Allow HTTPS traffic" check box in Firewall session, click "Create", wait a few minute for create



Firewall

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

☑ Allow HTTP traffic

☑ Allow HTTPS traffic

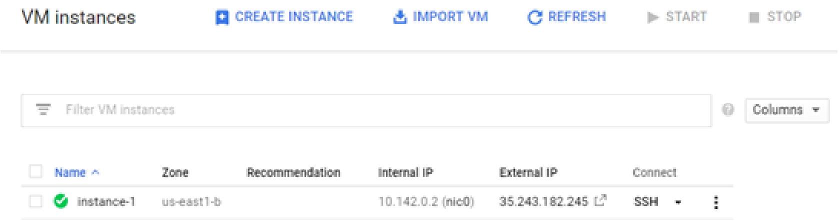
Management, security, disks, networking, sole tenancy

The following options have been customized:

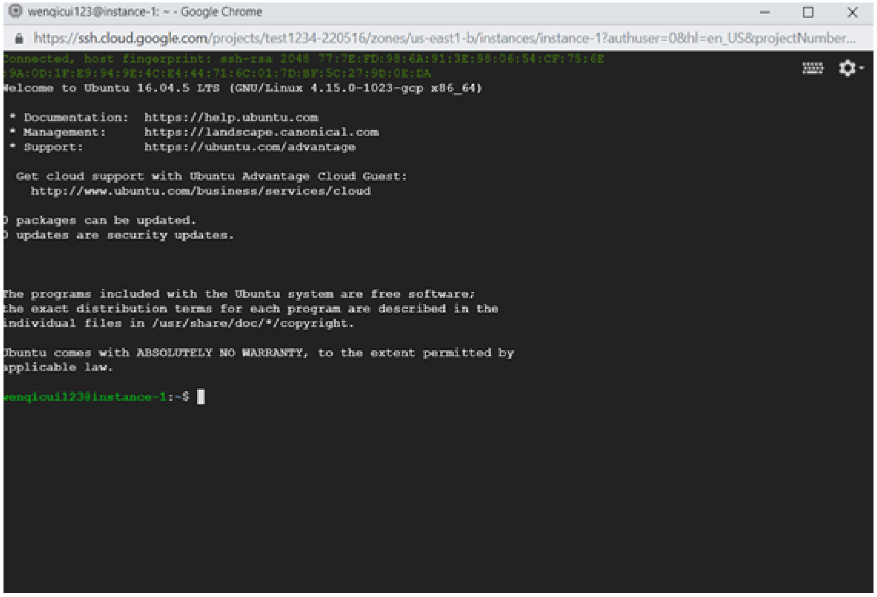
On host maintenance

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

Here we got our new VM instance, we can directly click the "SSH" to connect to the instance. Don't forget to stop the instance after finish the task, or Google Cloud will charge you continuously



Name	Zone	Recommendation	Internal IP	External IP	Connect
instance-1	us-east1-b		10.142.0.2 (nic0)	35.243.182.245	SSH


wenqicui123@instance-1: ~ - Google Chrome
https://ssh.cloud.google.com/projects/test1234-220516/zones/us-east1-b/instances/instance-1?authuser=0&hl=en_US&projectNumber...

```
Connected, host fingerprint: ssh-rsa 2048 77:7E:FD:98:6A:91:3E:98:06:54:CF:75:6E
:9A:0D:1F:E9:94:9E:4C:E4:44:71:6C:01:7D:BF:5C:27:9D:0E:DA
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-1023-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.

wenqicui123@instance-1:~$
```

Configure Python-related Environment and File Transfer

Install Anaconda

> Why do we need Anaconda? Anaconda contains python core, jupyter notebook(as IDE), and several useful packages. It is easy for us not only to install but also to manage the development environment.

Go to the Anaconda Downloads page(here), choose the linux version and **copy** the download link address, for now(Oct 2018) the link is `https://repo.anaconda.com/archive/Anaconda3-5.3.0-Linux-x86_64.sh`

Open the SSH of your VM instance, find a path where you want anaconda installed to, use `curl` to download the link we just copied.

```
$ curl -O https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh
```

Run what the `.sh` file when it's downloaded.

```
$ bash Anaconda3-5.2.0-Linux-x86_64.sh
```

Follow the instructions to install anaconda.

Once installed, you can activate the installation with the following command:

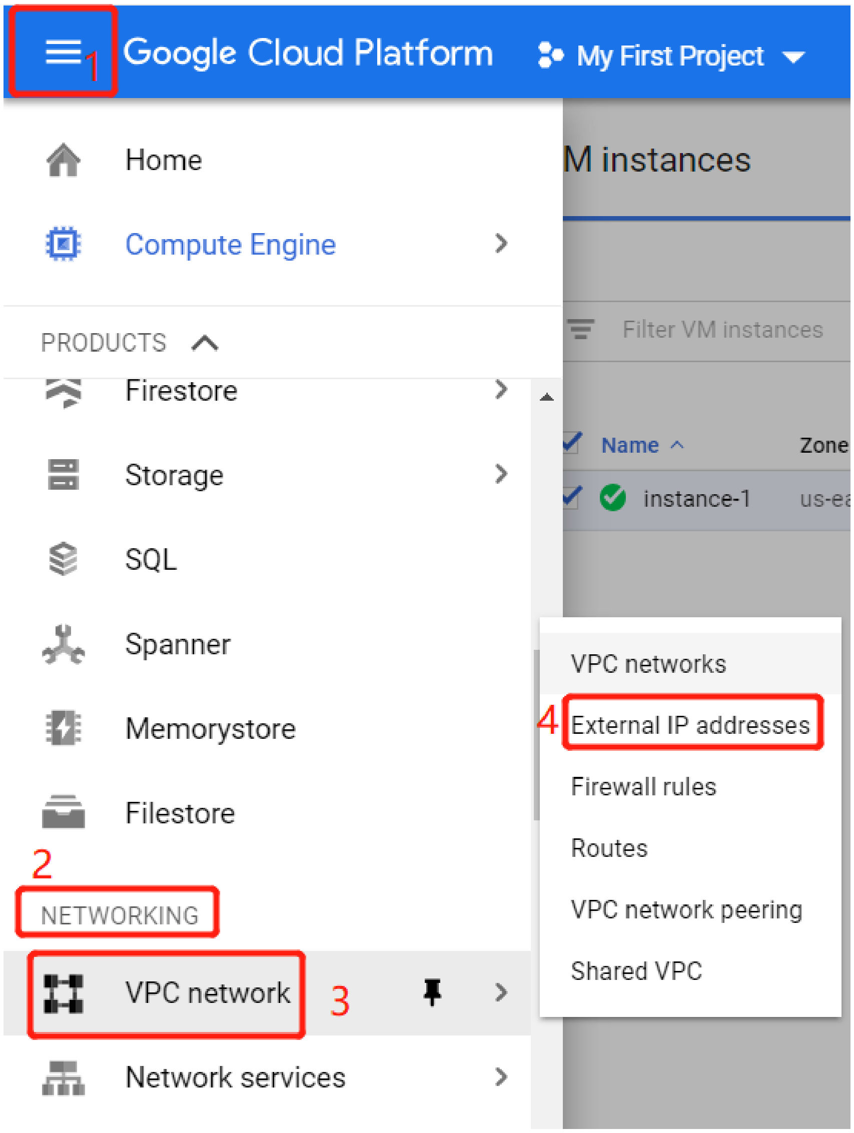
```
$ source ~/.bashrc
```

Set IP address to static

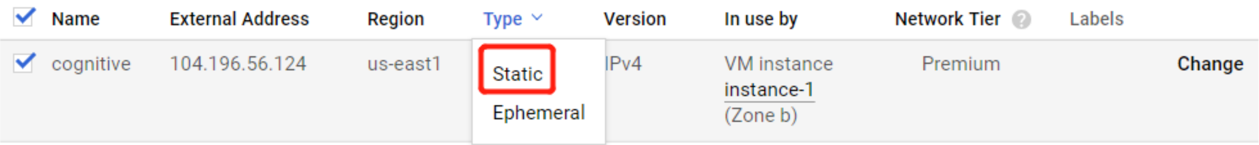
To open jupyter notebook from your local web browser, we need to set the external IP address of our Google Cloud instance from the default--dynamic to static.

To find where to change this setting, follow the step below:

```
MENU->NETWORKING->VPC network->External IP Addresses
```



In the list of your addresses, find the one you would like to change and change the type.

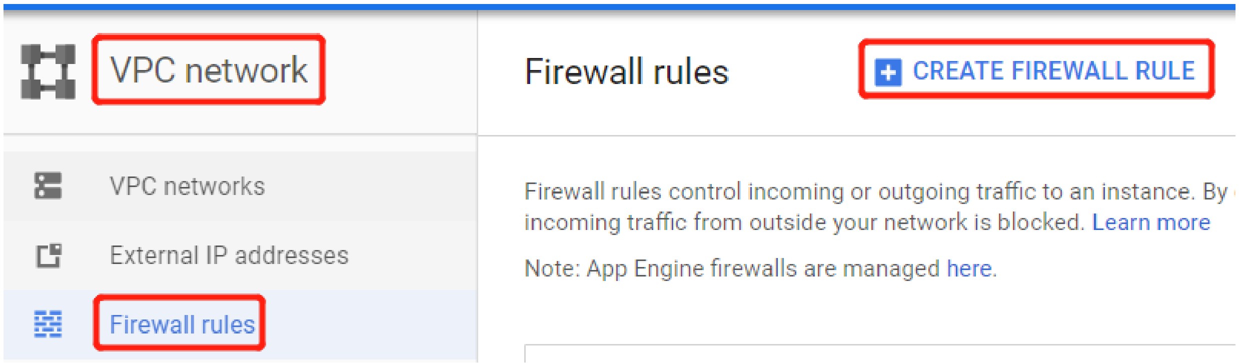


☑	Name	External Address	Region	Type	Version	In use by	Network Tier	Labels	
☑	cognitive	104.196.56.124	us-east1	Static Ephemeral	IPv4	VM instance instance-1 (Zone b)	Premium		Change

Then you are done. The external IP address of your instance now is static.

Open Jupyter Notebook in a web browser

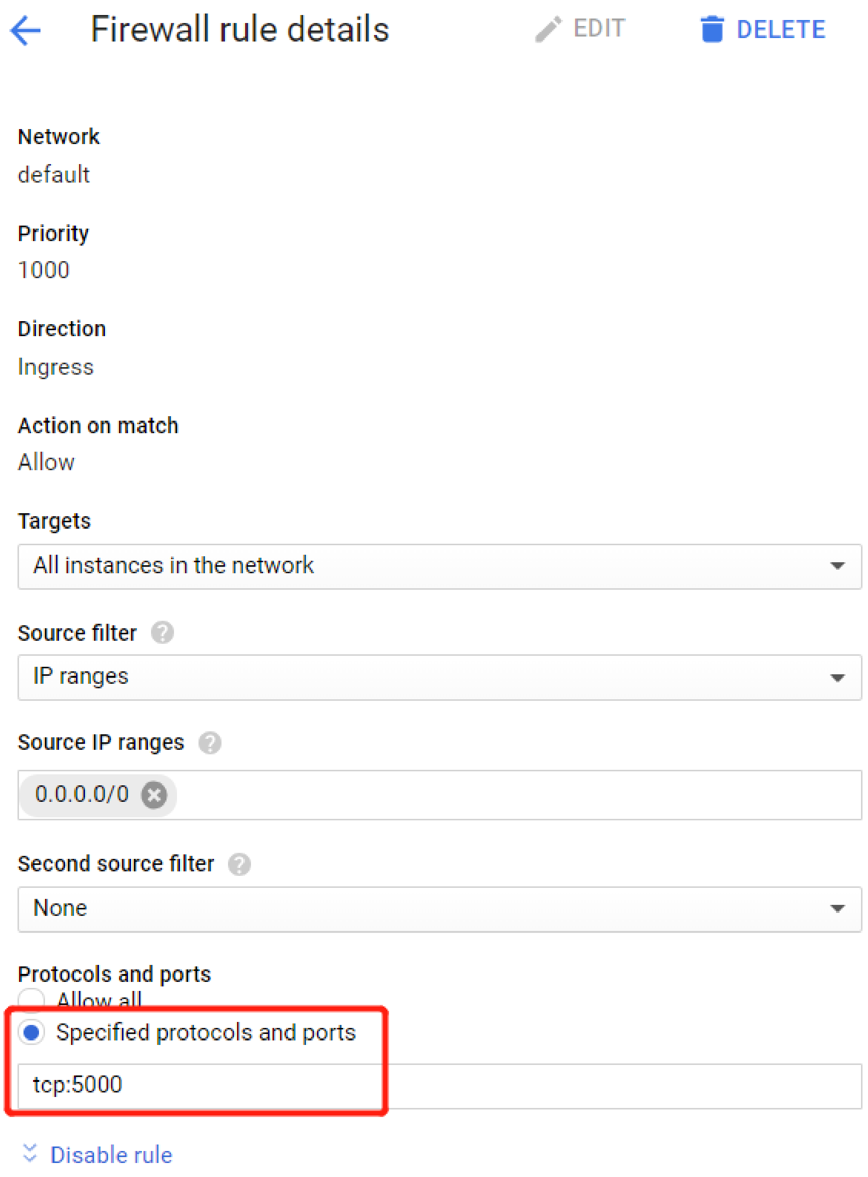
First we need to create a new firewall rule.



Firewall rules control incoming or outgoing traffic to an instance. By incoming traffic from outside your network is blocked. Learn more

Note: App Engine firewalls are managed here.

For 'Protocols and ports', choose Specified protocols and ports and set a tcp number you like. I've chosen tcp:5000 as my port number.



Network
default

Priority
1000

Direction
Ingress

Action on match
Allow

Targets
All instances in the network

Source filter
IP ranges

Source IP ranges
0.0.0.0/0

Second source filter
None

Protocols and ports
○ Allow all
● Specified protocols and ports
tcp:5000

Now click on the save button.

Then, in the SSH of your VM, check if you have a Jupyter configuration file by typing this commands:

```
$ ls ~/.jupyter/jupyter_notebook_config.py
```

If it doesn’t exist, create one:

```
$ jupyter notebook --generate-config
```

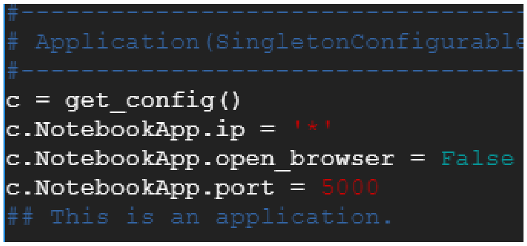
Now open the config file with vi:

```
$ vi jupyter_notebook_config.py
```

Then we need to add several lines in it, with the `<Port Number>` you set before. For me, it's `5000` .

```
c = get_config()
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.port = <Port Number>
```

You can add them anywhere inside this file since most of them are comments. It should look like this:

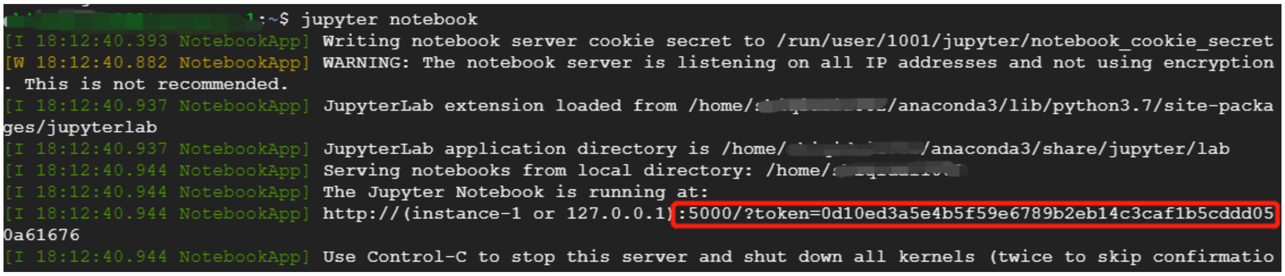


```
#------------------------------------
# Application(SingletonConfigurable
#------------------------------------
c = get_config()
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.port = 5000
## This is an application.
```

Now you can run jupyter notebook on your server.

```
$ jupyter notebook
```

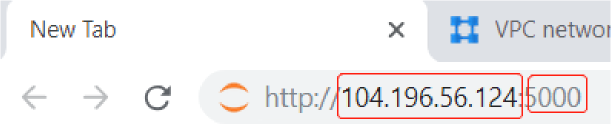
You will get something like this:



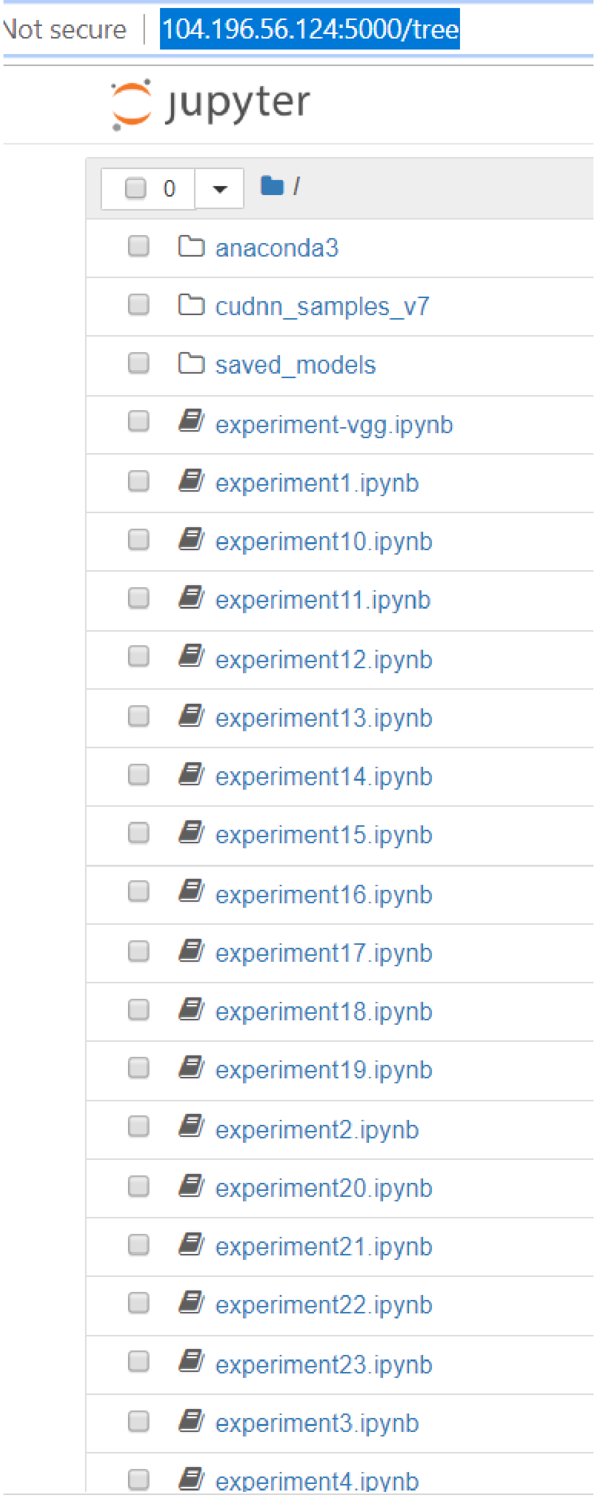
```
[redacted]:~$ jupyter notebook
[I 18:12:40.393 NotebookApp] Writing notebook server cookie secret to /run/user/1001/jupyter/notebook_cookie_secret
[W 18:12:40.882 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption
. This is not recommended.
[I 18:12:40.937 NotebookApp] JupyterLab extension loaded from /home/[redacted]/anaconda3/lib/python3.7/site-packa
ges/jupyterlab
[I 18:12:40.937 NotebookApp] JupyterLab application directory is /home/[redacted]/anaconda3/share/jupyter/lab
[I 18:12:40.944 NotebookApp] Serving notebooks from local directory: /home/[redacted]
[I 18:12:40.944 NotebookApp] The Jupyter Notebook is running at:
[I 18:12:40.944 NotebookApp] http://(instance-1 or 127.0.0.1):5000/?token=0d10ed3a5e4b5f59e6789b2eb14c3caf1b5cddd05
0a61676
[I 18:12:40.944 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmatio
```

Copy the token for later login.

Open your web browser and type your `<static external IP address>:<port number>` like this:



Then you can use your token to login jupyter notebook!



Not secure | 104.196.56.124:5000/tree

jupyter

□ 0 ▾ /

- anaconda3
- cudnn_samples_v7
- saved_models
- experiment-vgg.ipynb
- experiment1.ipynb
- experiment10.ipynb
- experiment11.ipynb
- experiment12.ipynb
- experiment13.ipynb
- experiment14.ipynb
- experiment15.ipynb
- experiment16.ipynb
- experiment17.ipynb
- experiment18.ipynb
- experiment19.ipynb
- experiment2.ipynb
- experiment20.ipynb
- experiment21.ipynb
- experiment22.ipynb
- experiment23.ipynb
- experiment3.ipynb
- experiment4.ipynb

File Transfer via WinSCP

> Why use WinSCP? WinSCP can help you easily upload and download files between your local and the instance. It is used in the next part of our tutorial. More importantly, we will need it to download notebooks and models.

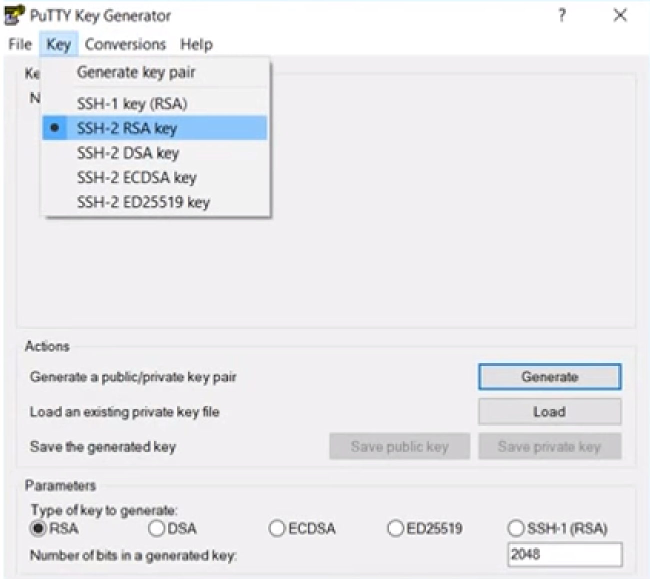
Before starting you should:

- Have WinSCP installed

Now first, we need to use PuTTYgen tool to generate new key.

> PuTTYgen installs by default with WinSCP.

Open PuTTYgen, choose `SSH-2 RSA key` under the `Key` tab.

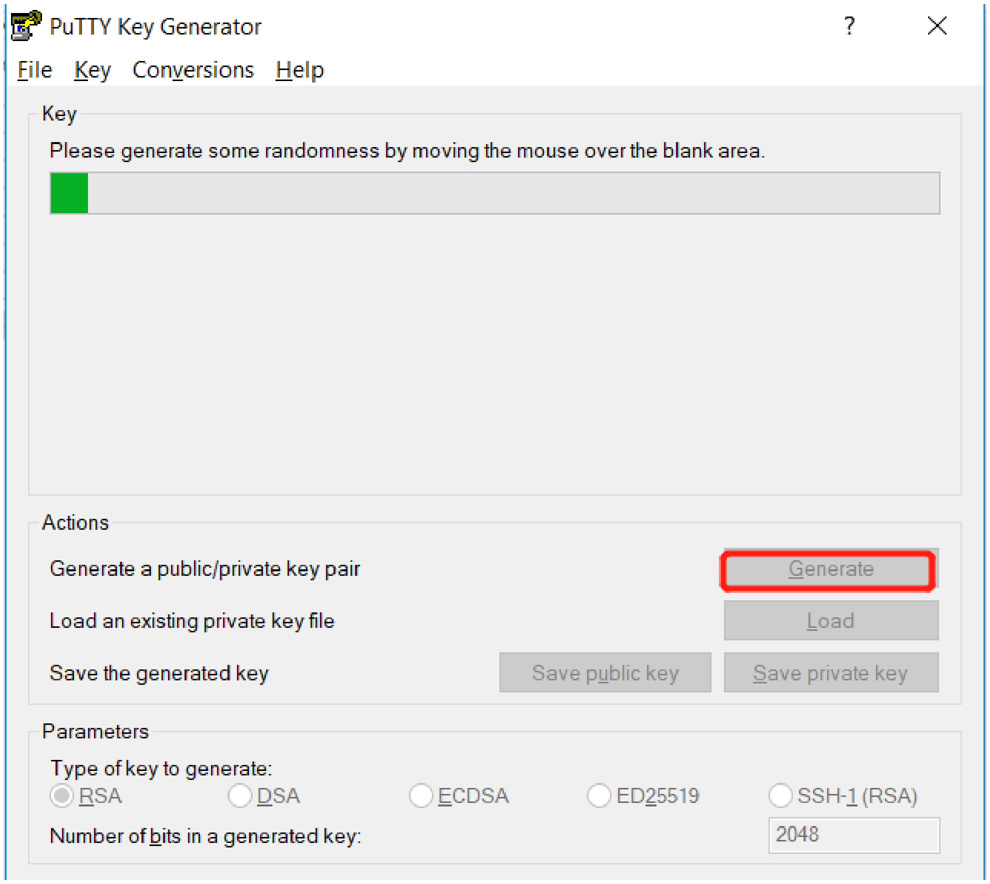


Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

Click `Generate`

> while generating, remember move the mouse over the blank area.

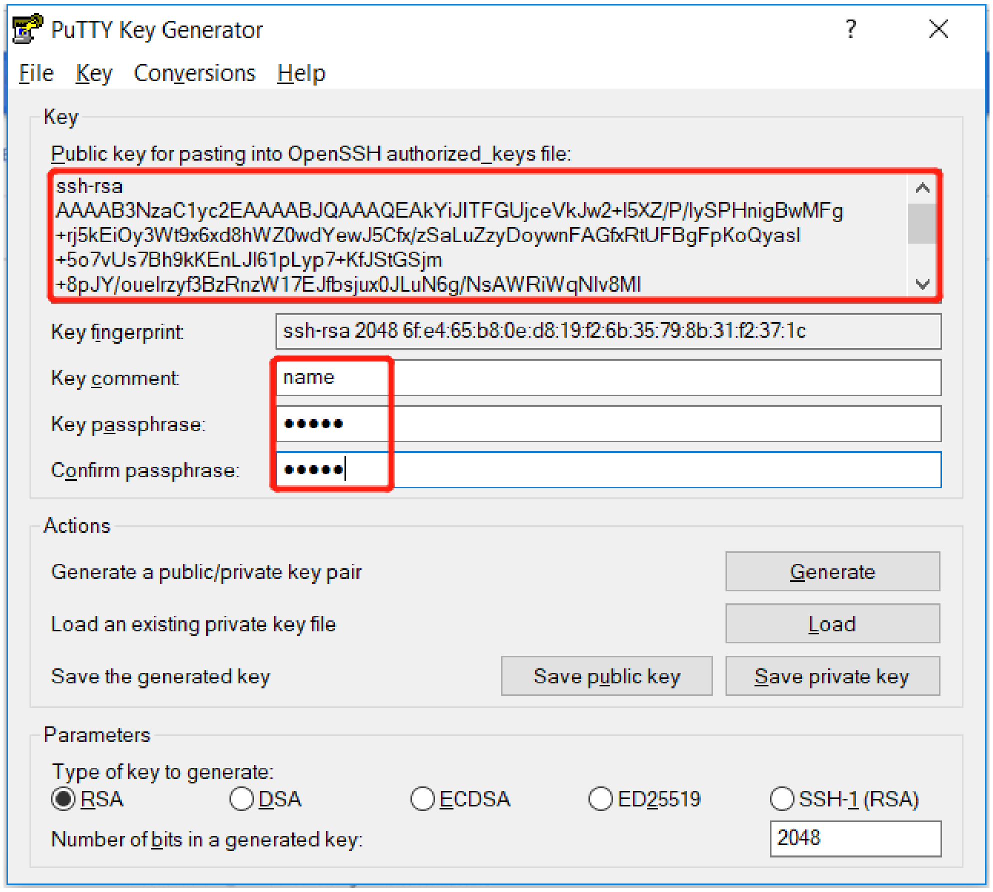


Please generate some randomness by moving the mouse over the blank area.

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

When the generating is done, set the key comment and passphrase(red box below) and you will have a long text string(red box above), copy it.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAkYiJITFGUjceVkJw2+l5XZ/P/lySPHnigBwMFg
+rj5kEiOy3Wt9x6xd8hWZ0wdYewJ5Cfx/zSaLuZzyDoywnFAGfxRtUFBgFpKoQyasl
+5o7vUs7Bh9kKEnLJl61pLyp7+KfJStGSjm
+8pJY/ouelrzyf3BzRnzW17EJfbsjux0JLuN6g/NsAWRiWqNlv8MI
```

Key fingerprint: ssh-rsa 2048 6f:e4:65:b8:0e:d8:19:f2:6b:35:79:8b:31:f2:37:1c

Key comment: name

Key passphrase: •••••

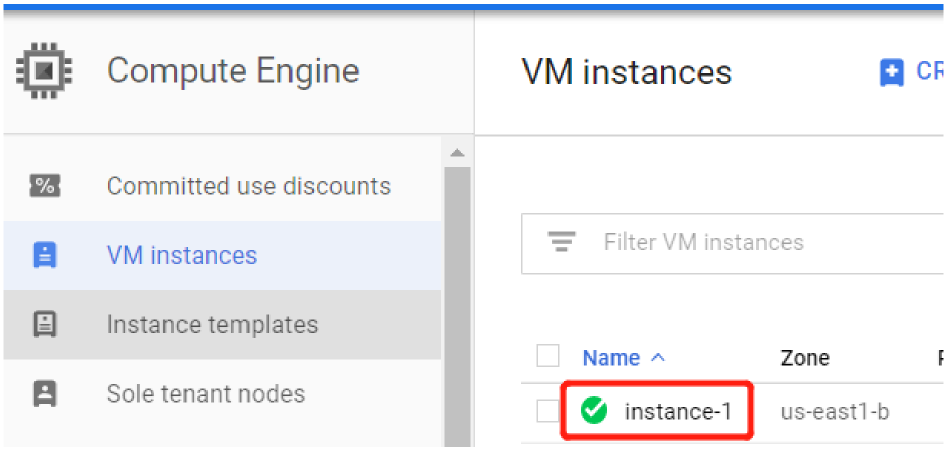
Confirm passphrase: •••••

Type of key to generate:
● RSA ○ DSA ○ ECDSA ○ ED25519 ○ SSH-1 (RSA)

Number of bits in a generated key: 2048

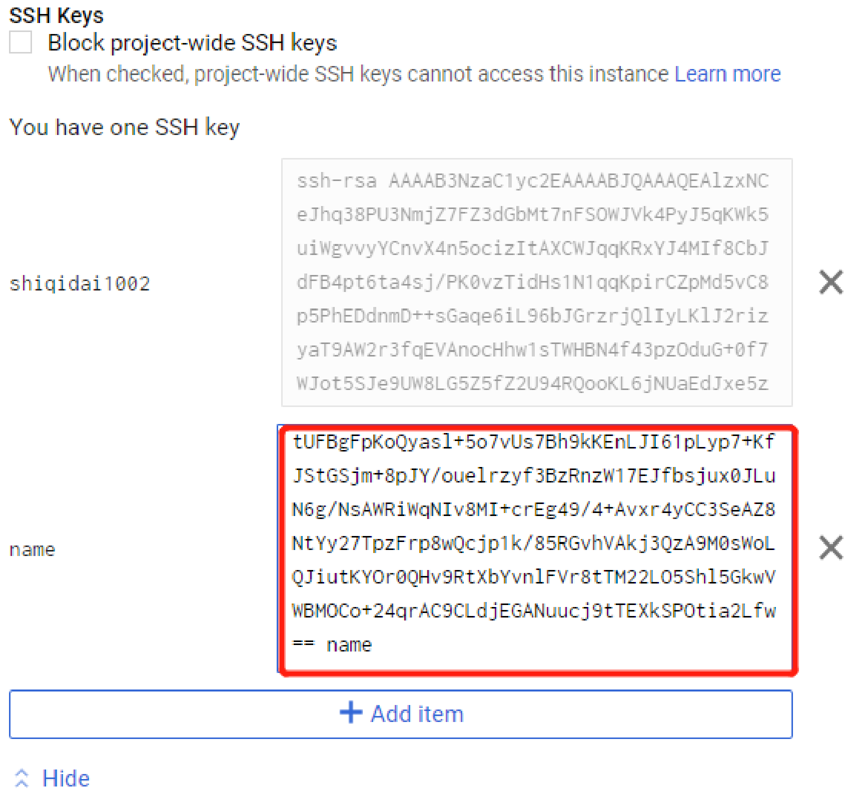
Then click the save button to save your private key to your local.

Go to Google Cloud Platform, click on your instance to enter the detail page.



Name	Zone
instance-1	us-east1-b

Then click `edit`, scroll down, under `SSH Keys` click `show and edit`, then click `Add Item` and paste the long text of previous generated key here.



SSH Keys

□ Block project-wide SSH keys

When checked, project-wide SSH keys cannot access this instance

You have one SSH key

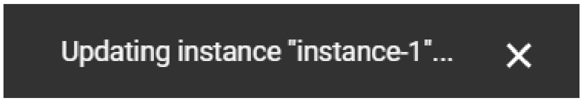
shiqidai1002

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAlzxNC
eJhq38PU3NmjZ7FZ3dGbMt7nFSOWJVk4PyJ5qKWk5
uiWgvvyYCnvX4n5ocizItAXCWJqqKRxYJ4MIf8CbJ
dFB4pt6ta4sj/PK0vzTidHs1N1qqKpirCZpMd5vC8
p5PhEDdnmD++sGaqe6iL96bJGrzrjQlIyLKlJ2riz
yaT9AW2r3fqEVAnocHhw1sTWHBN4f43pzOduG+0f7
WJot5SJe9UW8LG5Z5fZ2U94RQooKL6jNUaEdJxe5z
```

name

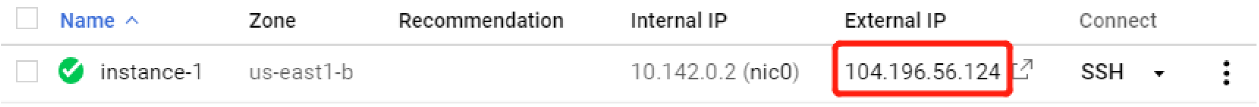
```
tUFBgFpKoQyasl+5o7vUs7Bh9kKEnLJI61pLyp7+Kf
JStGSjm+8pJY/ouelrzyf3BzRnzW17EJfbsjux0JLu
N6g/NsAWRiWqNIv8MI+crEg49/4+Avxr4yCC3SeAZ8
NtYy27TpzFrp8wQcjp1k/85RGvhVAkj3QzA9M0sWoL
QJiutKYOr0QHv9RtXbYvnlFVr8tTM22LO5Shl5GkwV
WBMOCo+24qrAC9CLdjEGANuucj9tTEXkSPOtia2Lfw
== name
```

Scroll down to the bottom, `save` it. You will see a pop-up like this:



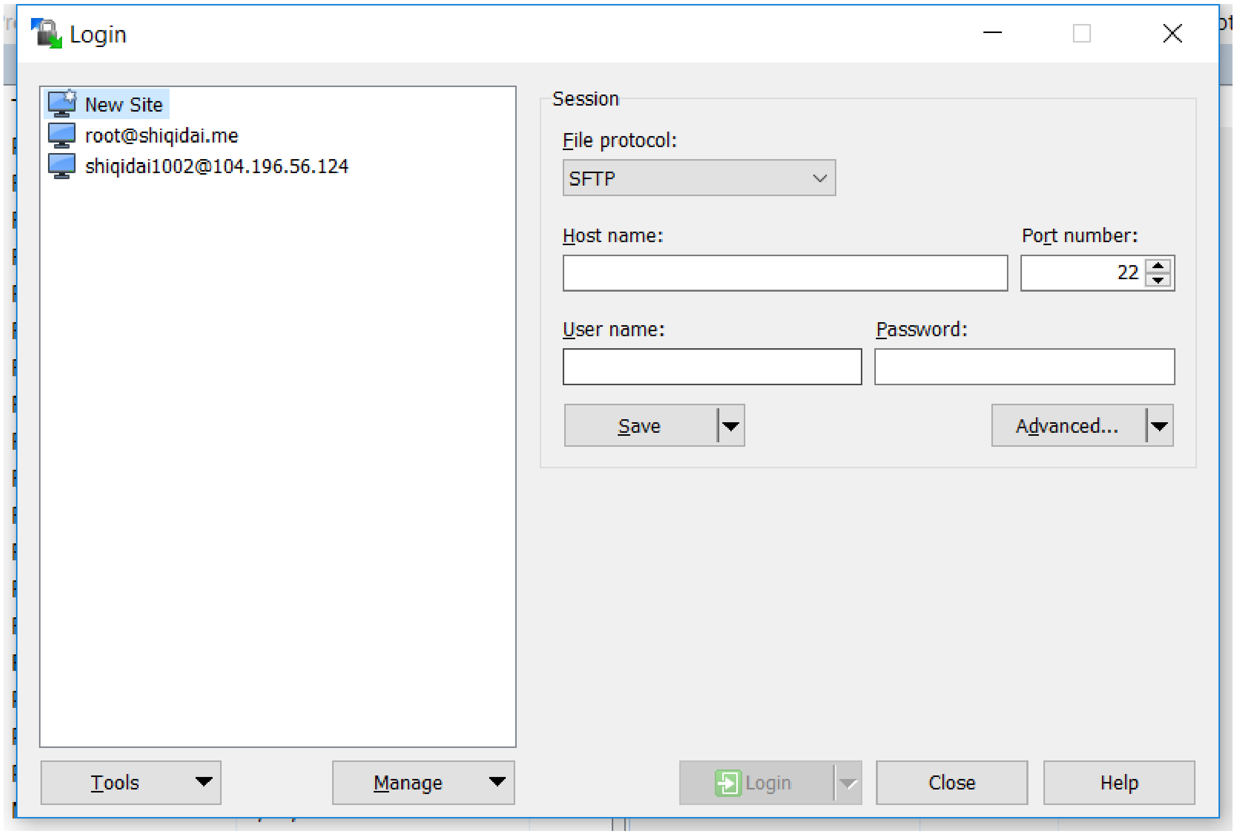
Updating instance "instance-1"...

Now go back to your instance, copy the external IP address.

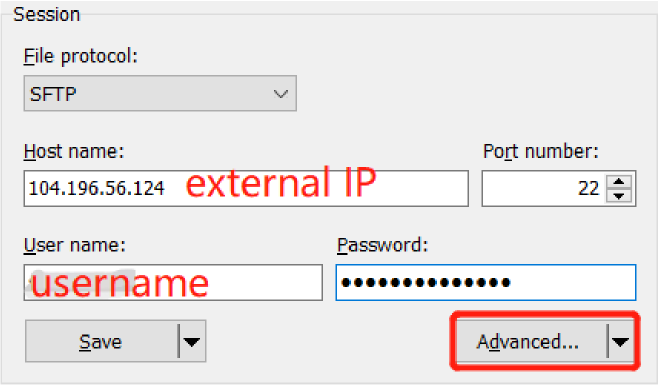


□	Name ^	Zone	Recommendation	Internal IP	External IP	Connect
□	✓ instance-1	us-east1-b		10.142.0.2 (nic0)	104.196.56.124	SSH ▾

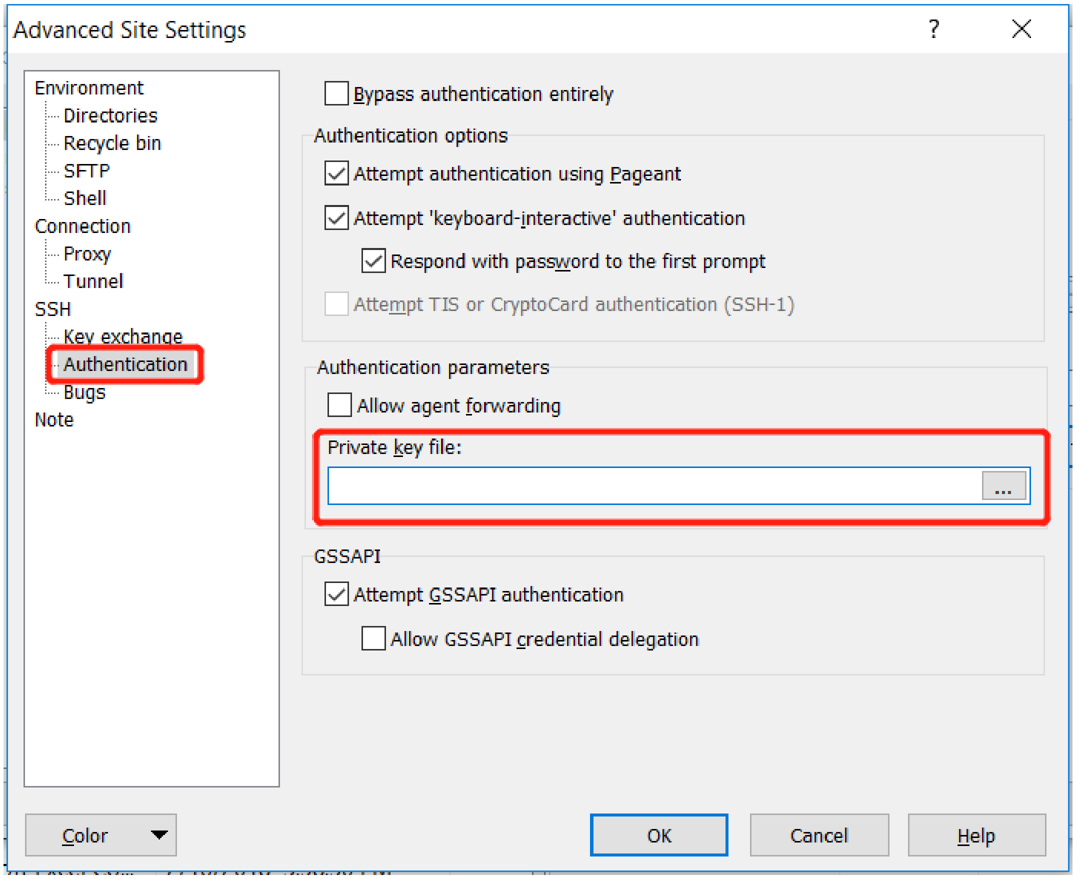
Open WinSCP from your local, from the left panel, choose `New Site` :



Paste your external IP and type you Google user name and password, then click `Advanced` :



Under advanced setting, choose SSH->Authentication->chose your private key file saved in your local before. Then click `ok`



□ Bypass authentication entirely

Authentication options

☑ Attempt authentication using Pageant

☑ Attempt 'keyboard-interactive' authentication

☑ Respond with password to the first prompt

□ Attempt TIS or CryptoCard authentication (SSH-1)

Authentication parameters

□ Allow agent forwarding

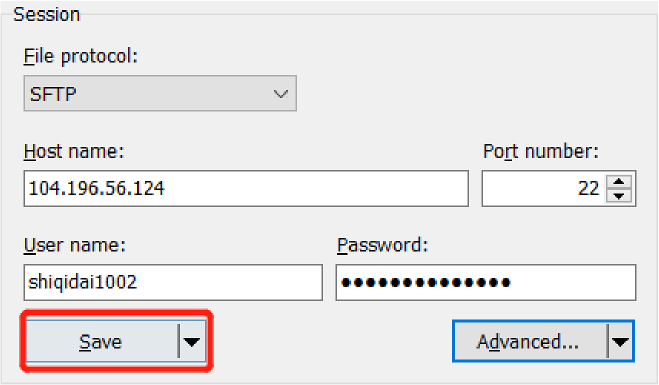
Private key file: ...

GSSAPI

☑ Attempt GSSAPI authentication

□ Allow GSSAPI credential delegation

Back to the login window, click `save` to make it easier for you to log in the next time.



File protocol: SFTP

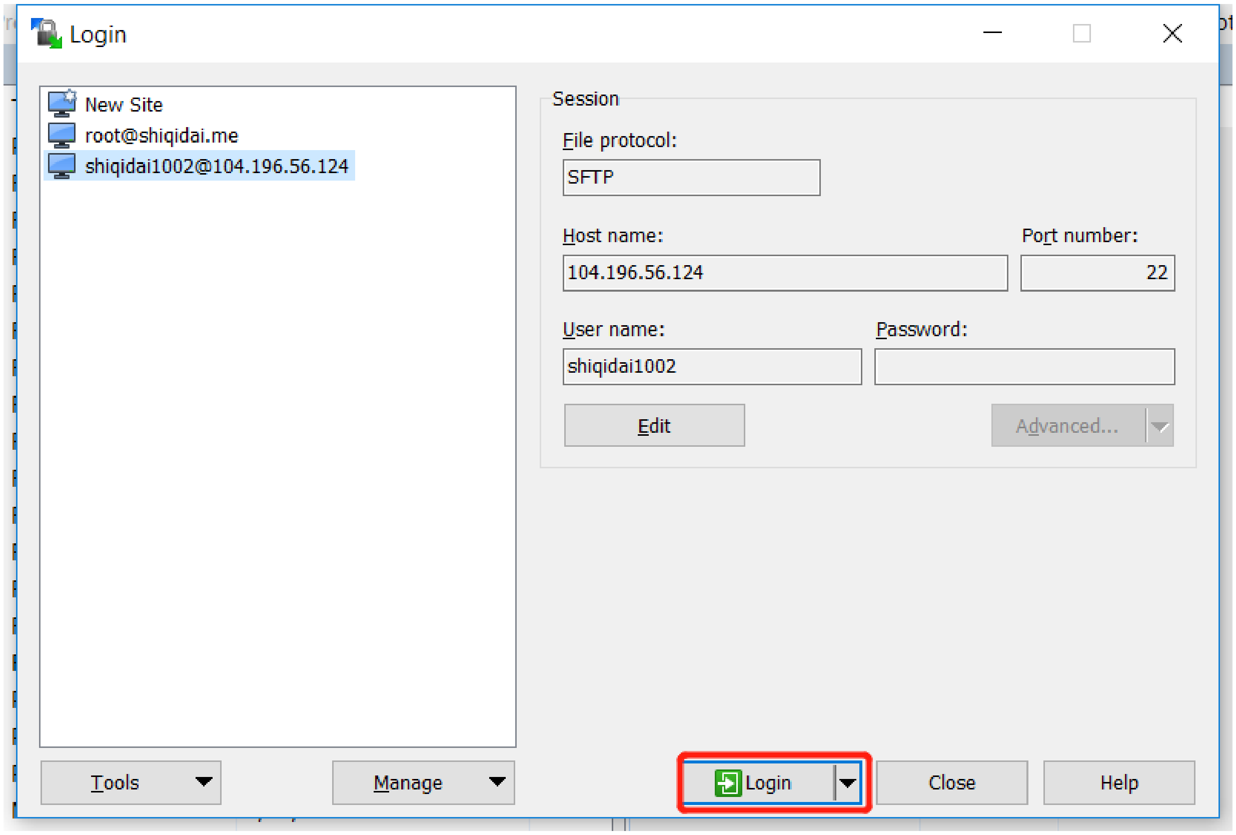
Host name: 104.196.56.124

Port number: 22

User name: shiqidai1002

Password: ●●●●●●●●●●●●●●

Now we can `login`



File protocol: SFTP

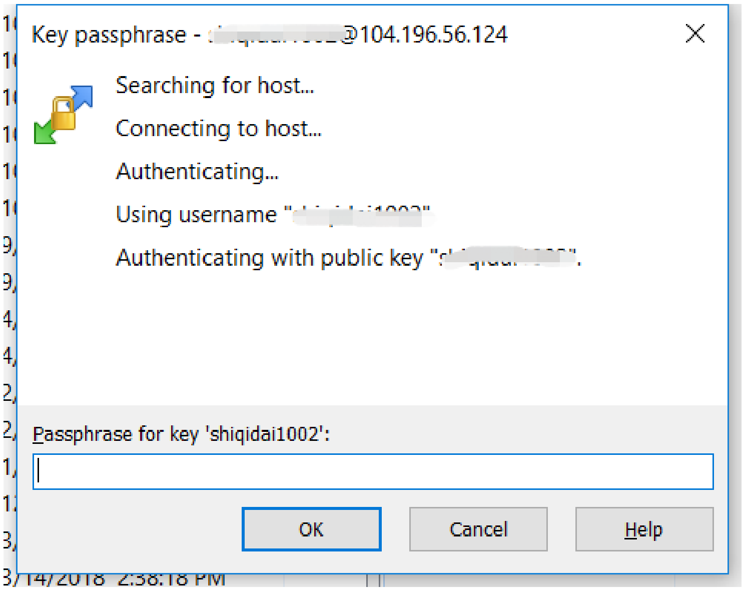
Host name: 104.196.56.124

Port number: 22

User name: shiqidai1002

Password:

It will ask you to enter you passphrase, the one you set when you created the key.



Key passphrase - [redacted]@104.196.56.124

Searching for host...

Connecting to host...

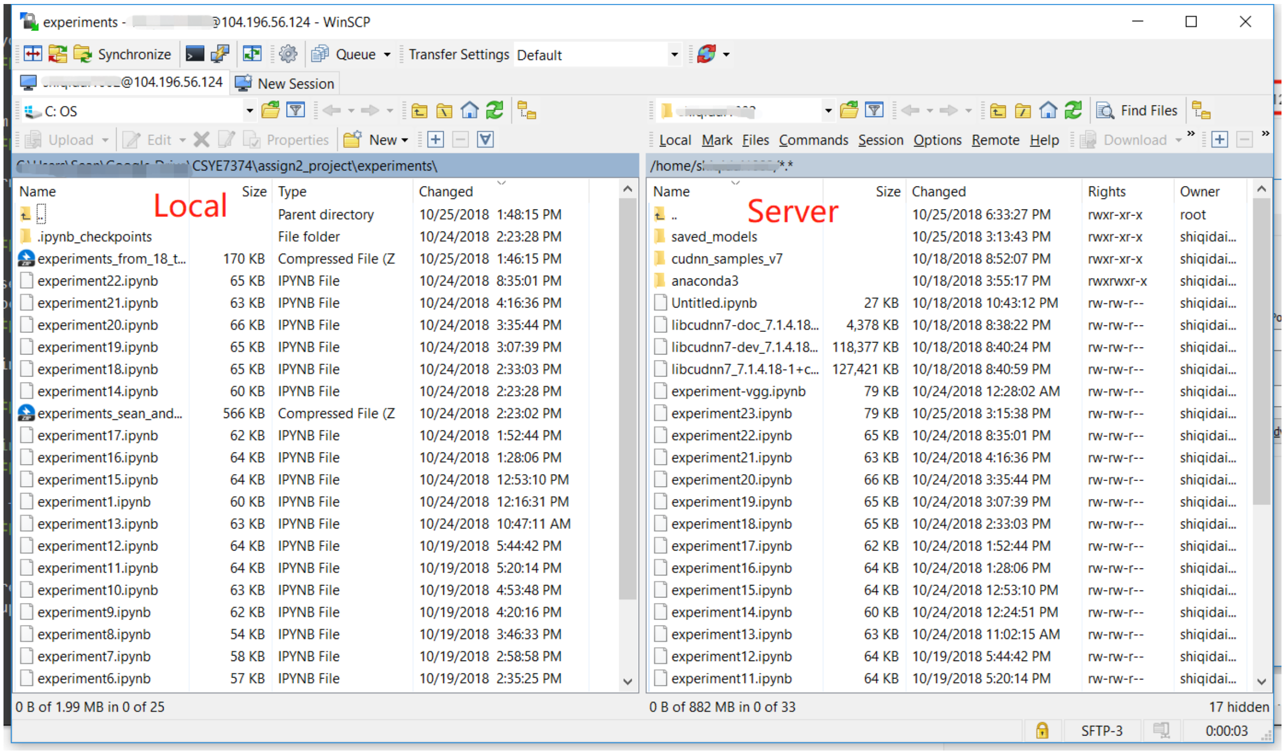
Authenticating...

Using username "[redacted]"

Authenticating with public key "[redacted]".

Passphrase for key 'shiqidai1002':

By entering correct passphrase, you are able to connect to the server. Now you are able to easily upload or download files!


experiments - [redacted]@104.196.56.124 - WinSCP
Synchronize
Queue
Transfer Settings Default
[redacted]@104.196.56.124
New Session
C: OS
Upload
Edit
Properties
New
Find Files
Local Mark Files Commands Session Options Remote Help
Download
Local
Server

[redacted]\CSYE7374\assign2_project\experiments\

Name	Size	Type	Changed
..		Parent directory	10/25/2018 1:48:15 PM
.ipynb_checkpoints		File folder	10/24/2018 2:23:28 PM
experiments_from_18_t...	170 KB	Compressed File (Z	10/25/2018 1:46:15 PM
experiment22.ipynb	65 KB	IPYNB File	10/24/2018 8:35:01 PM
experiment21.ipynb	63 KB	IPYNB File	10/24/2018 4:16:36 PM
experiment20.ipynb	66 KB	IPYNB File	10/24/2018 3:35:44 PM
experiment19.ipynb	65 KB	IPYNB File	10/24/2018 3:07:39 PM
experiment18.ipynb	65 KB	IPYNB File	10/24/2018 2:33:03 PM
experiment14.ipynb	60 KB	IPYNB File	10/24/2018 2:23:28 PM
experiments_sean_and...	566 KB	Compressed File (Z	10/24/2018 2:23:02 PM
experiment17.ipynb	62 KB	IPYNB File	10/24/2018 1:52:44 PM
experiment16.ipynb	64 KB	IPYNB File	10/24/2018 1:28:06 PM
experiment15.ipynb	64 KB	IPYNB File	10/24/2018 12:53:10 PM
experiment1.ipynb	60 KB	IPYNB File	10/24/2018 12:16:31 PM
experiment13.ipynb	63 KB	IPYNB File	10/24/2018 10:47:11 AM
experiment12.ipynb	64 KB	IPYNB File	10/19/2018 5:44:42 PM
experiment11.ipynb	64 KB	IPYNB File	10/19/2018 5:20:14 PM
experiment10.ipynb	63 KB	IPYNB File	10/19/2018 4:53:48 PM
experiment9.ipynb	62 KB	IPYNB File	10/19/2018 4:20:16 PM
experiment8.ipynb	54 KB	IPYNB File	10/19/2018 3:46:33 PM
experiment7.ipynb	58 KB	IPYNB File	10/19/2018 2:58:58 PM
experiment6.ipynb	57 KB	IPYNB File	10/19/2018 2:35:25 PM

0 B of 1.99 MB in 0 of 25

/home/s[redacted]/*.*

Name	Size	Changed	Rights	Owner
..		10/25/2018 6:33:27 PM	rwxr-xr-x	root
saved_models		10/25/2018 3:13:43 PM	rwxr-xr-x	shiqidai...
cudnn_samples_v7		10/18/2018 8:52:07 PM	rwxr-xr-x	shiqidai...
anaconda3		10/18/2018 3:55:17 PM	rwxrwxr-x	shiqidai...
Untitled.ipynb	27 KB	10/18/2018 10:43:12 PM	rw-rw-r--	shiqidai...
libcudnn7-doc_7.1.4.18...	4,378 KB	10/18/2018 8:38:22 PM	rw-rw-r--	shiqidai...
libcudnn7-dev_7.1.4.18...	118,377 KB	10/18/2018 8:40:24 PM	rw-rw-r--	shiqidai...
libcudnn7_7.1.4.18-1+c...	127,421 KB	10/18/2018 8:40:59 PM	rw-rw-r--	shiqidai...
experiment-vgg.ipynb	79 KB	10/24/2018 12:28:02 AM	rw-rw-r--	shiqidai...
experiment23.ipynb	79 KB	10/25/2018 3:15:38 PM	rw-rw-r--	shiqidai...
experiment22.ipynb	65 KB	10/24/2018 8:35:01 PM	rw-rw-r--	shiqidai...
experiment21.ipynb	63 KB	10/24/2018 4:16:36 PM	rw-rw-r--	shiqidai...
experiment20.ipynb	66 KB	10/24/2018 3:35:44 PM	rw-rw-r--	shiqidai...
experiment19.ipynb	65 KB	10/24/2018 3:07:39 PM	rw-rw-r--	shiqidai...
experiment18.ipynb	65 KB	10/24/2018 2:33:03 PM	rw-rw-r--	shiqidai...
experiment17.ipynb	62 KB	10/24/2018 1:52:44 PM	rw-rw-r--	shiqidai...
experiment16.ipynb	64 KB	10/24/2018 1:28:06 PM	rw-rw-r--	shiqidai...
experiment15.ipynb	64 KB	10/24/2018 12:53:10 PM	rw-rw-r--	shiqidai...
experiment14.ipynb	60 KB	10/24/2018 12:24:51 PM	rw-rw-r--	shiqidai...
experiment13.ipynb	63 KB	10/24/2018 11:02:15 AM	rw-rw-r--	shiqidai...
experiment12.ipynb	64 KB	10/19/2018 5:44:42 PM	rw-rw-r--	shiqidai...
experiment11.ipynb	64 KB	10/19/2018 5:20:14 PM	rw-rw-r--	shiqidai...

0 B of 882 MB in 0 of 33 17 hidden

SFTP-3 0:00:03

Keras using MXNet as backend

After previous actions, we've already got a Python environment installed in our server. Now we are going to configure MXNet based Keras using GPU training.

Install CUDA

To utilize GPU(NVIDIA branded) for training, we have to install two NVIDIA libraries to setup with GPU support. One of them is CUDA, another is cuDNN.

First let's install CUDA.

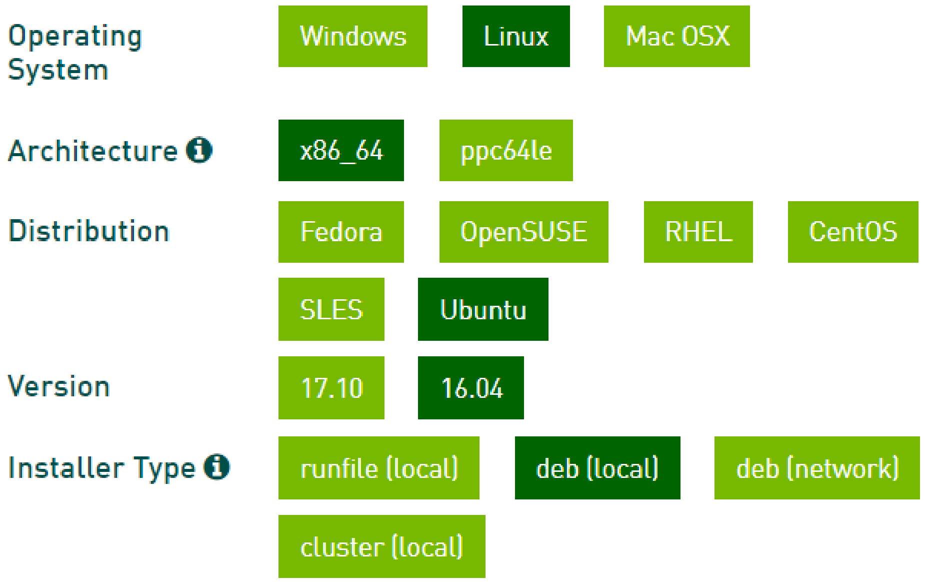
The updated version of CUDA is CUDA10, but normally we use previous version like CUDA8 and CUDA9.0+. In our experiments, we used CUDA9.2, so we will use CUDA9.2 as an example.

> You can also install CUDA 9.2 following the NVIDIA's installation guide.

1. **Download CUDA from NVIDIA**

Since we are using an Ubuntu machine, you have to choose the correct CUDA debian package when trying to download from NVIDIA(link).

Make sure you're choosing this:



Operating System: Windows [✓ Linux] Mac OSX

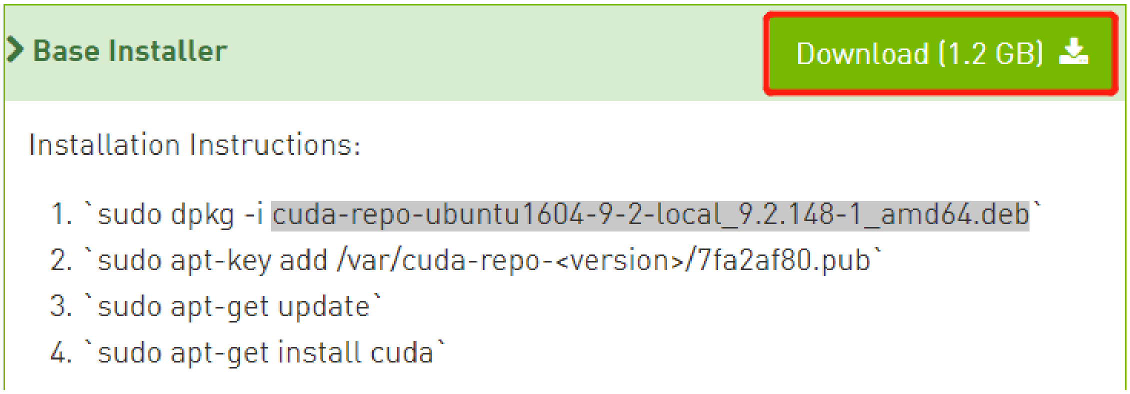
Architecture: [✓ x86_64] ppc64le

Distribution: Fedora OpenSUSE RHEL CentOS SLES [✓ Ubuntu]

Version: 17.10 [✓ 16.04]

Installer Type: runfile (local) [✓ deb (local)] deb (network) cluster (local)

Then you will see the 'base installer'



> Base Installer

Download (1.2 GB)

Installation Instructions:

1. `sudo dpkg -i cuda-repo-ubuntu1604-9-2-local_9.2.148-1_amd64.deb`
2. `sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub`
3. `sudo apt-get update`
4. `sudo apt-get install cuda`

For the choices we made, you should get a deb file named as ' `cuda-repo-ubuntu1604-9-2-local_9.2.148-1_amd64.deb` '

Then put the deb file into your sever folder and run the following commands to install it.

> You can upload and download files between local and sever by using WinSCP
>
> ```
> $ sudo dpkg -i cuda-repo-ubuntu1604-9-2-local_9.2.148-1_amd64.deb
> $ sudo apt-key add /var/cuda-repo-ubuntu1604-9-2-local_9.2.148-1_amd64/7fa2af80.pub
> $ sudo apt-get update
> $ sudo apt-get install cuda
> ```
>
> Note: the `<version>` of `cuda-repo-<version>` should be replaced by the version info corresponding to the name of the downloaded deb file.

Make sure to add the CUDA install path to `LD_LIBRARY_PATH` . Using the following commands:

```
$ export CUDA_HOME=/usr/local/cuda
$ export PATH=${CUDA_HOME}/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=${CUDA_HOME}/lib64/:$LD_LIBRARY_PATH
```

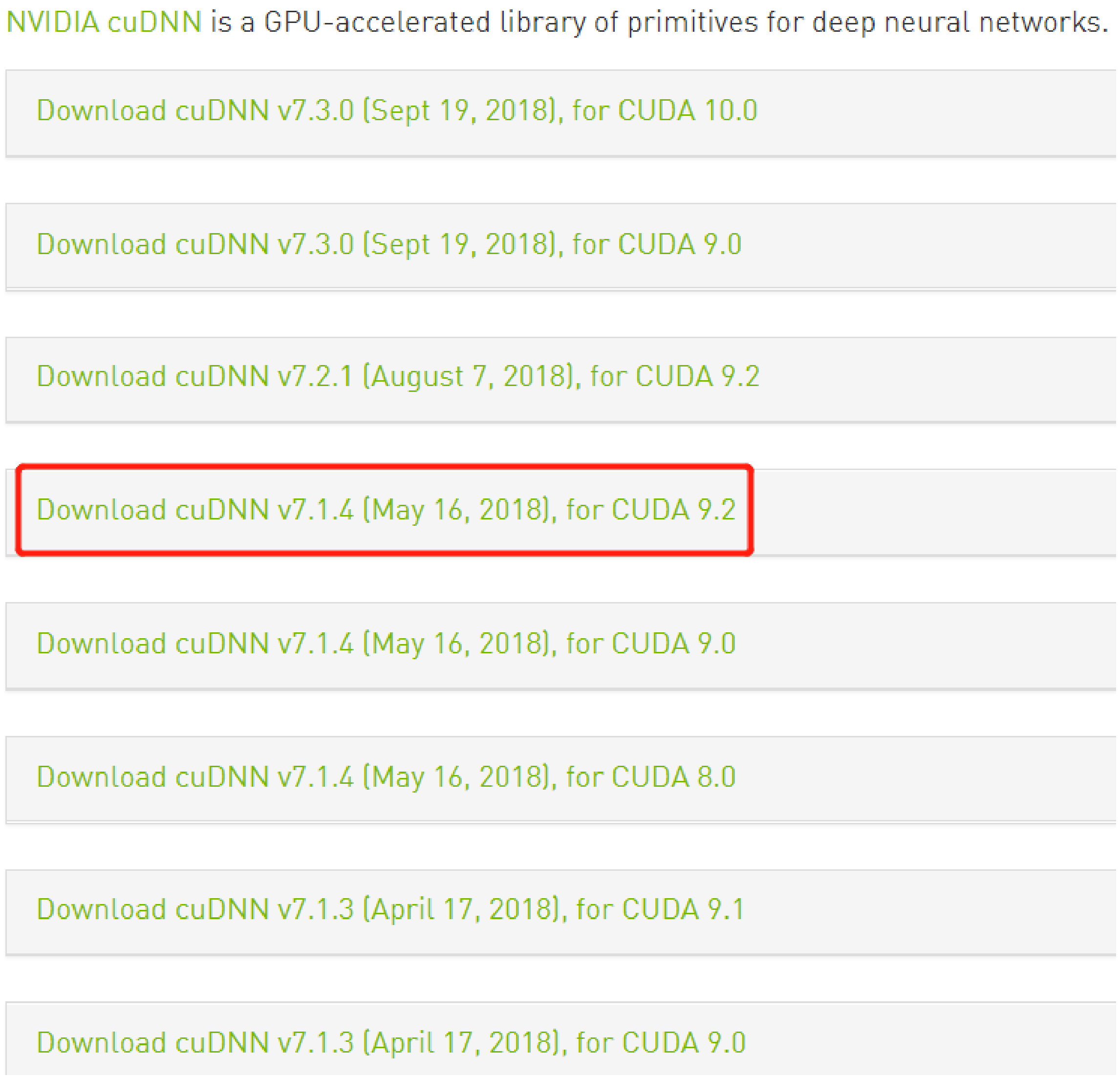
Install cuDNN

Now let's install cuDNN.

First, to download cuDNN, you have to own an NVIDIA developer account. If you don't, you can register it here.

Once you got a developer account, you can download cuDNN from here.

Choose the correct version of cuDNN for your CUDA. For us, we installed CUDA9.2, so we should install cuDNN 7.1.4 or 7.2.1.



NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

Download cuDNN v7.3.0 (Sept 19, 2018), for CUDA 10.0

Download cuDNN v7.3.0 (Sept 19, 2018), for CUDA 9.0

Download cuDNN v7.2.1 (August 7, 2018), for CUDA 9.2

Download cuDNN v7.1.4 (May 16, 2018), for CUDA 9.2

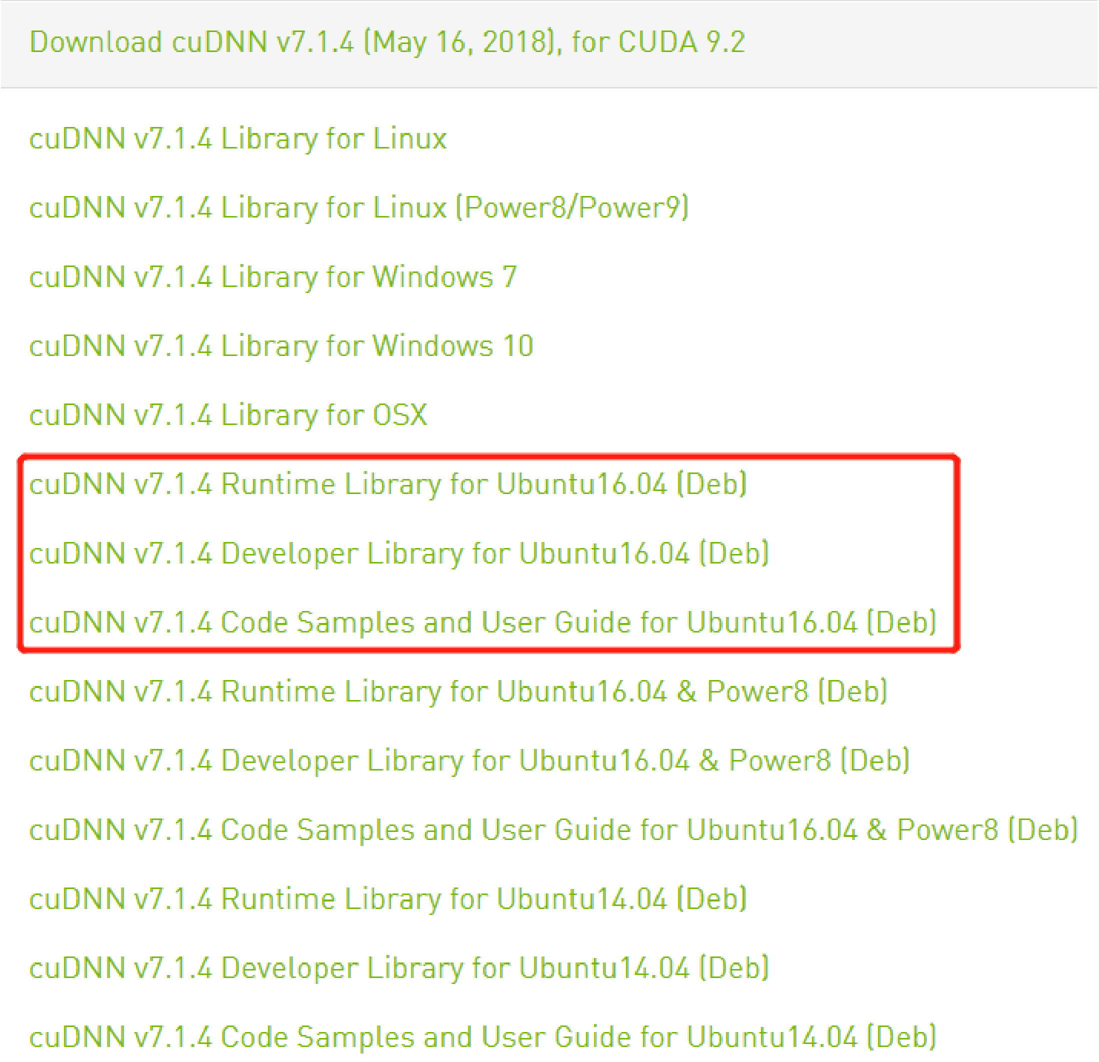
Download cuDNN v7.1.4 (May 16, 2018), for CUDA 9.0

Download cuDNN v7.1.4 (May 16, 2018), for CUDA 8.0

Download cuDNN v7.1.3 (April 17, 2018), for CUDA 9.1

Download cuDNN v7.1.3 (April 17, 2018), for CUDA 9.0

Download all 3 '.deb' files: the runtime library, the developer library, and the code samples library for Ubuntu 16.04.



Download cuDNN v7.1.4 (May 16, 2018), for CUDA 9.2

cuDNN v7.1.4 Library for Linux

cuDNN v7.1.4 Library for Linux (Power8/Power9)

cuDNN v7.1.4 Library for Windows 7

cuDNN v7.1.4 Library for Windows 10

cuDNN v7.1.4 Library for OSX

cuDNN v7.1.4 Runtime Library for Ubuntu16.04 (Deb)

cuDNN v7.1.4 Developer Library for Ubuntu16.04 (Deb)

cuDNN v7.1.4 Code Samples and User Guide for Ubuntu16.04 (Deb)

cuDNN v7.1.4 Runtime Library for Ubuntu16.04 & Power8 (Deb)

cuDNN v7.1.4 Developer Library for Ubuntu16.04 & Power8 (Deb)

cuDNN v7.1.4 Code Samples and User Guide for Ubuntu16.04 & Power8 (Deb)

cuDNN v7.1.4 Runtime Library for Ubuntu14.04 (Deb)

cuDNN v7.1.4 Developer Library for Ubuntu14.04 (Deb)

cuDNN v7.1.4 Code Samples and User Guide for Ubuntu14.04 (Deb)

In your download folder, install them in the same order:

```
# the runtime library
$ sudo dpkg -i libcudnn7_7.1.4.18-1+cuda9.2_amd64.deb

# the developer library
$ sudo dpkg -i libcudnn7-dev_7.1.4.18-1+cuda9.2_amd64.deb

# the code samples
$ sudo dpkg -i libcudnn7-doc_7.1.4.18-1+cuda9.2_amd64.deb
```

Now we can verify the cuDNN installation:

1. Copy the code samples somewhere you have write access:
   `cp -r /usr/src/cudnn_samples_v7/ ~`
2. Go to the MNIST example code:
   `cd ~/cudnn_samples_v7/mnistCUDNN`
3. Compile the MNIST example:
   `make clean && make`
4. Run the MNIST example:
   `./mnistCUDNN`

If your installation is successful, you should see `Test passed!` at the end of the output.

Now we are exporting environment variables LD_LIBRARY_PATH in your .bashrc fileby putting the following line in the end or your `.bashrc` file.

```
$ export LD_LIBRARY_PATH="LD_LIBRARY_PATH=${LD_LIBRARY_PATH:+${LD_LIBRARY_PATH}:}/usr/local/cuda/extras/CUPTI/lib64"
```

Last, source it:

```
$ source ~/.bashrc
```

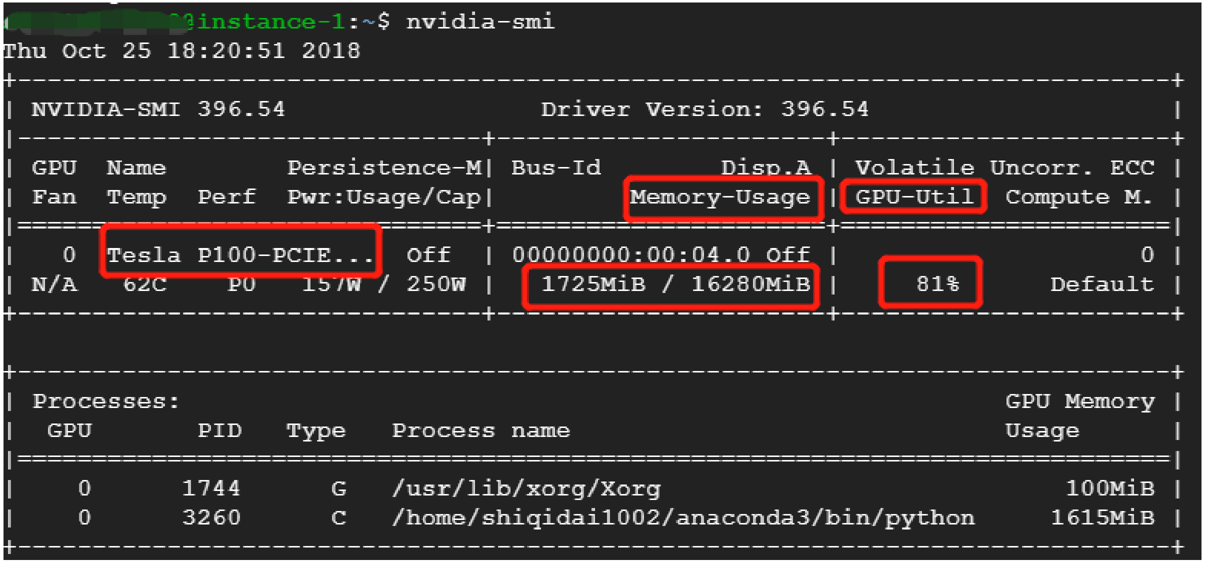
Check if everything is fine

You can verify your CUDA setup with the following commands:

```
$ nvcc --version
$ nvidia-smi
```

> Note: this command also can be used to monitor the usage of your GPU(s) whiling training.

It will look like this when training with GPU:



```
[redacted]@instance-1:~$ nvidia-smi
Thu Oct 25 18:20:51 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.54                 Driver Version: 396.54                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   62C    P0   157W / 250W |   1725MiB / 16280MiB |     81%      Default |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1744      G   /usr/lib/xorg/Xorg                            100MiB |
|    0      3260      C   /home/shiqidai1002/anaconda3/bin/python      1615MiB |
+-----------------------------------------------------------------------------+
```

Install MXNet

You have to install the right mxnet corresponding to your installed CUDA version. For example, if you use CUDA9, the mxnet for you is 'mxnet-cu90'. If CUDA8, then use 'mxnet-cu80'.

Use the following code to install.

```
$ pip install mxnet-cu92
```

> Since our CUDA version is 9.2, we installed mxnet-cu92 correspondingly.

Install Keras with MXNet backend

We need to install a special version of Keras called 'keras-mxnet', since we are going to use MXNet as backend for Keras.

Use the following code:

```
  $ pip install keras-mxnet --user
```

Configure Keras backend and image_data_format

In the previous step, we installed the `keras-mxnet` , by default, the following values are set in the keras config file `keras.json` .

```
backend: mxnet
image_data_format: channels_last
```

Accordong to Keras official documentation:

> We strongly recommend changing the image_data_format to channels_first. MXNet is significantly faster on 'channels_first' data. Default is set to 'channels_last' with an objective to be compatible with majority of existing users of Keras. See performance tips guide for more details.

Thus, we are gonna change the default setting. Usually, you can find `keras.json` file in this path: `.keras/keras.json`

You can use vim to open and edit it when in the same folder.

`sudo vi keras.json`

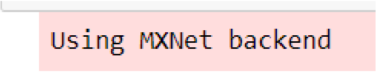
To edit it, press 'i' to enter edit mode. After changing `image_data_format` to `channels_first` , type ':wq' to save and quit.

Validate the Installation

You can validate the installation by trying to import Keras in Python terminal and verifying that Keras is using mxnet backend.

```
$ python
  >>> import keras as k
    Using mxnet backend>>>
```

Or something like this if you are using jupyter notebook



```
Using MXNet backend
```

Related Links

- Keras with Apache MXNet Documentation
- Install CUDA 9.2 and cuDNN 7.1 for PyTorch (GPU) on Ubuntu 16.04
- How To Install Anaconda on Ubuntu 18.04
- Running Jupyter Notebook on Google Cloud Platform in 15 min
- Connecting Securely to Google Compute Engine Server with SFTP
- connect to google compute engine from WinSCP and PuttY