CUDA 9.0 installation guide for Ubuntu 16.04

This tutorial has been put together in order to assist the students of the Autonomous Drones Lab to set up CUDA quickly and easily. If you come across problems with these instructions, need assistance or want to suggest improvements, please address your requests to maayantamari@mail.tau.ac.il.

Environment

- Ubuntu 16.04
- NVIDIA GPU

Pre-installation steps

Step 1: Verification of prerequisites

We want to verify the followings before starting the installation:

- **CUDA-capable GPU**
- **Supported version of Linux**
- **GCC**
- **Correct kernel headers**
- **No other versions of CUDA or NVIDIA drivers are installed**

1. **Verify the system has a CUDA-capable GPU.**

   To verify that your GPU is CUDA-capable, go to your distribution's equivalent of System Properties, or, from the command line, enter:

```
lspci | grep –i nvidia
```

2. **Verify the system is running a supported version of Linux.**
   The CUDA Development Tools are only supported on some specific distributions of Linux.
   To determine which distribution and release number you are running, type the following at the command line:

```
uname -m && cat /etc/*release
```

You should see output similar to the following, modified for your particular system:

```
yoni6@yoni6-ThinkPad-E470:~$ uname -m && cat /etc/*release
x86_64
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.6 LTS"
NAME="Ubuntu"
VERSION="16.04.6 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.6 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
```

The x86_64 line indicates you are running on a 64-bit system. The remainder gives information about your distribution.

Supported distributions of Linux from the official guide for CUDA 9.0:

Table 1 Native Linux Distribution Support in CUDA 9.0

Distribution	Kernel	GCC	GLIBC	ICC	PGI	XLC	CLANG
x86_64							
RHEL 7.x	3.10	4.8.5	2.17	17.0	17.1	NO	3.9
RHEL 6.x	2.6.32	4.4.7	2.12				
CentOS 7.x	3.10	4.8.5	2.17				
CentOS 6.x	2.6.32	4.4.7	2.12				
Fedora 25	4.8.8	6.2.1	2.24-3				
OpenSUSE Leap 42.2	4.4.27	4.8	2.22				
SLES 12 SP2	4.4.21	4.8.5	2.22				
Ubuntu 17.04	4.9.0	6.3.0	2.24-3				
Ubuntu 16.04	4.4	5.3.1	2.23				
POWER8(*)							
RHEL 7.x	3.10	4.8.5	2.17	NO	17.1	13.1.5	NO
Ubuntu 16.04	4.4	5.3.1	2.23	NO	17.1	13.1.5	NO

(*) Only the Tesla P100 GPU is supported for CUDA 9.0 on POWER8.

3. **Verify the system has GCC installed.**
   To verify the version of GCC installed on your system, type the following on the command line:

```
gcc --version
```

   If an error message displays, you need to install the development tools from your Linux distribution or obtain a version of GCC and its accompanying toolchain from the Web.

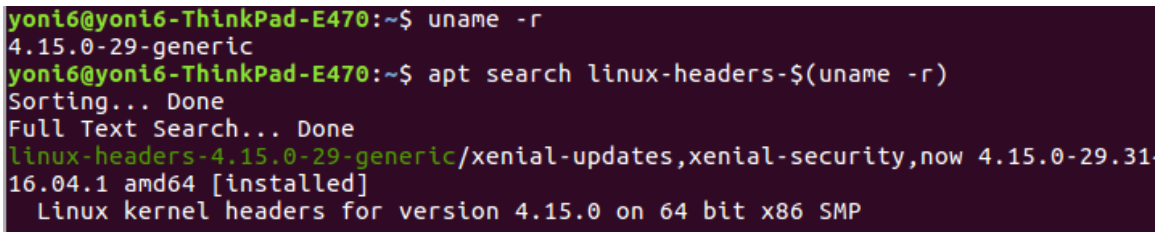
4. **Verify the system has the correct kernel headers and development packages installed.**
   The version of the kernel your system is running can be found by running the following command:

```
uname -r
```

   To check if the kernel headers are installed:

```
apt search linux-headers-$(uname -r)
```

   The headers should be in **usr/src**/



   `linux-headers-$(uname -r)` is the version of the kernel headers and development packages that must be installed prior to installing the CUDA Drivers. If the kernel headers are missing, you should install them.

   To install the kernel headers (if needed):

```
sudo apt-get install linux-headers-$(uname -r)
```

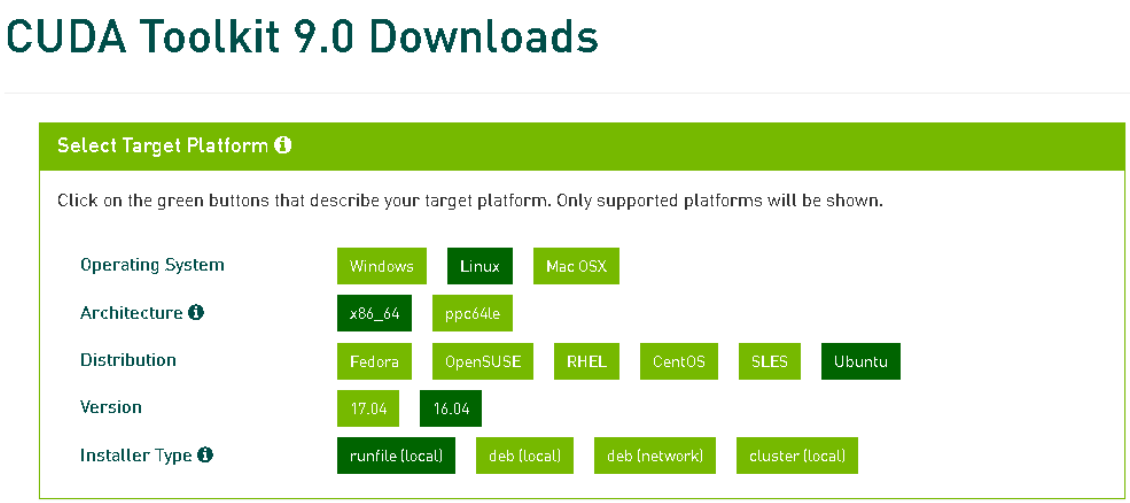
5. **Handle conflicting installation methods.**
   Before installing CUDA, any previously installations that could conflict should be uninstalled.

```
sudo apt-get purge nvidia-cuda*
sudo apt-get purge nvidia-*
```

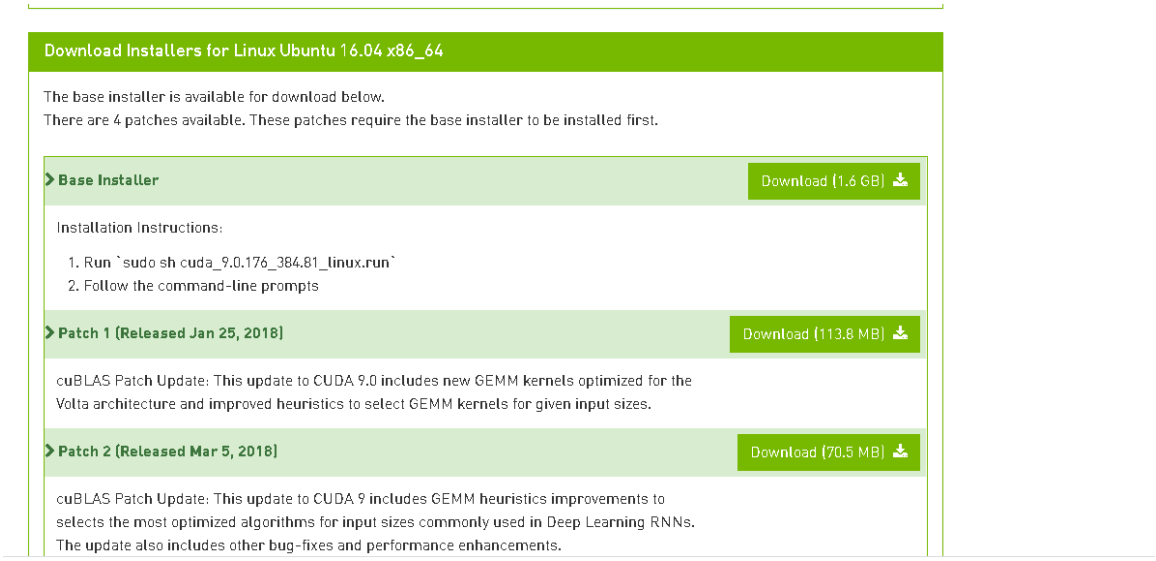
Step 2: Download the NVIDIA CUDA Toolkit

Go to https://developer.nvidia.com/cuda-90-download-archive?target_os=Linux

Choose the followings



After selecting the target platform, you will be able to download the runfile. Scroll down and press download for the base installer. Ignore the installation instructions written there for now.



Another method of getting the runfile:

```
cd
wget
https://developer.nvidia.com/compute/cuda/9.0/Prod/local_installers/cuda_9.0.176_384.81_linux-run
```

The runfile you downloaded is a package containing the following three components:

1. NVIDIA driver installer;
2. CUDA installer;
3. CUDA samples installer;

Installation steps

Step 1: Logout of your GUI

In order to install the display drivers, logout from your GUI. To log out press (ctrl + alt + F1). That will lead you to a terminal session.

Step 2: Disable LightDM and Nuoveau drivers

LightDM is the display manager running in Ubuntu up to version 16.04 LTS.
Nouveau is a free and open-source graphics device driver for NVIDIA video cards. Both causes issues with the installation, so we what to disable them before starting.

Stop LightDM:

```
sudo service lightdm stop
```

Stop Nouveau:
Each distribution of Linux has a different method for disabling Nouveau. The Nouveau drivers are loaded if the following command prints anything:

```
lsmod | grep nouveau
```

To disable, create a file at **/etc/modprobe.d/blacklist-nouveau.conf** with the following contents:

```
blacklist nouveau
options nouveau modeset=0
```

Then execute:

```
sudo update-initramfs -u
```

Step 3: Extract the components of the runfile

To extract the three components of the runfile, execute the runfile with –extract

```
chmod +x cuda_9.0.176_384.81_linux-run
./cuda_9.0.176_384.81_linux-run --extract=$HOME
```

You should have unpacked three components:

- NVIDIA-Linux-x86_64-384.81.run (1. NVIDIA driver),
- cuda-linux.9.0.176-22781540.run (2. CUDA 9.0 installer),
- cuda-samples.9.0.176-22781540-linux.run (3. CUDA 9.0 Samples).

Step 4: Install the NVIDIA driver

First, you should check the NVIDIA driver, it is usually a stale version and you might want to replace it with a newer version (You can also upgrade in the future if you do not want to do it now). If you look at the following table, taken from the link bellow, you can see that the driver version you downloaded in the package is the oldest version compatible to CUDA 9. Newer drivers can run older version of CUDA.

Table 1. CUDA Toolkit and Compatible Driver Versions

CUDA Toolkit	Linux x86_64 Driver Version
CUDA 10.2 (10.2.89)	>= 440.33
CUDA 10.1 (10.1.105)	>= 418.39
CUDA 10.0 (10.0.130)	>= 410.48
CUDA 9.2 (9.2.88)	>= 396.26
CUDA 9.1 (9.1.85)	>= 390.46
CUDA 9.0 (9.0.76)	>= 384.81
CUDA 8.0 (8.0.61 GA2)	>= 375.26
CUDA 8.0 (8.0.44)	>= 367.48
CUDA 7.5 (7.5.16)	>= 352.31
CUDA 7.0 (7.0.28)	>= 346.46

https://docs.nvidia.com/deploy/cuda-compatibility/index.html#binary-compatibility__table-toolkit-driver

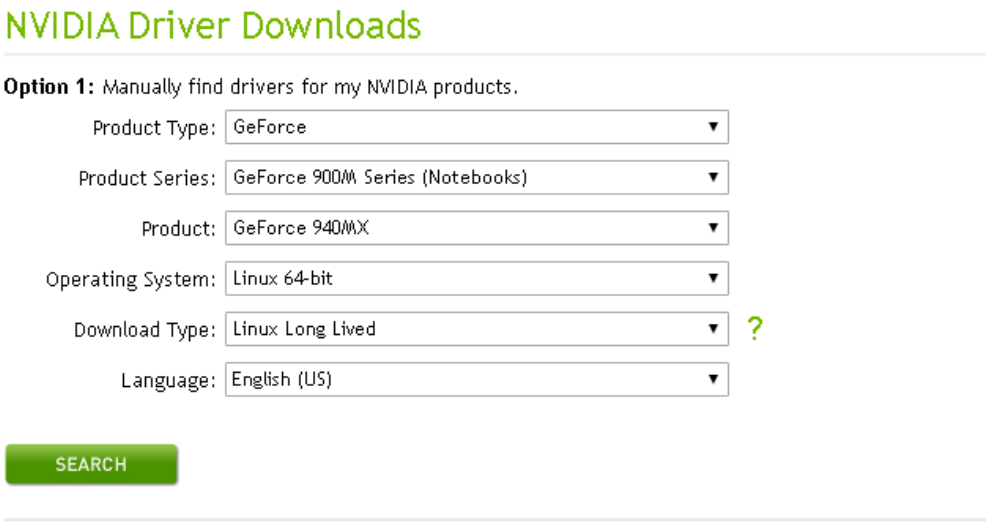
Two ways to download the driver-

Downloading a run file from the official NVIDIA website:

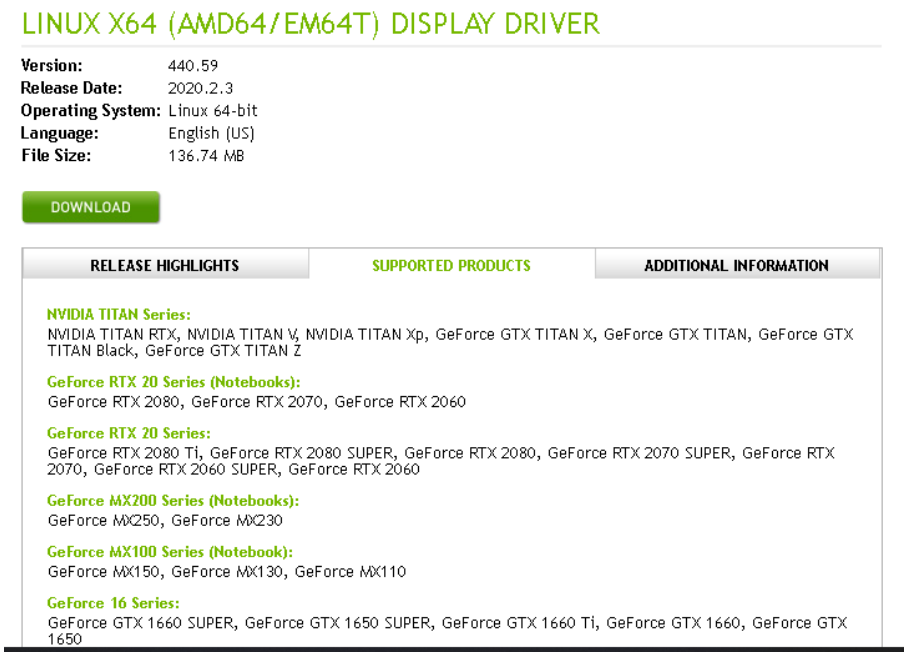
Go to:

https://www.nvidia.com/Download/index.aspx

Provide your system and GPU information and press search.



You will be provided with the latest version compatible to your GPU. You can check out the supported products bellow to make sure it is compatible.



After downloading the runfile, go to its location and execute:

```
sudo su
chmod +x NVIDIA-Linux-x86_64-285.05.09.run
sudo sh NVIDIA-Linux-x86_64-285.05.09.run
sudo apt-get install nvidia-modprobe
```

Verify the NVIDIA driver installation

```
nvidia-smi
```

Downloading from PPA:

You should check the versions available in the following link. Not all versions are available in PPA. Make sure it is compatible to your version of Ubuntu.

https://launchpad.net/~graphics-drivers/+archive/ubuntu/ppa

Add the PPA and install

```
sudo add-apt-repository ppa:graphocs-drivers
sudo apt-get update
sudo apt-get install nvidia-<version> nvidia-modprobe
```

Verify the NVIDIA driver installation

```
nvidia-smi
```

Step 5: Install the CUDA Toolkit 9.0

Execute

```
sudo ./cuda-linux.9.0.176-22781540.run
```

You now have to accept the license by scrolling down to the bottom and enter "accept". Next, accept the defaults.

Step 6: Install the CUDA sample tests

To verify the CUDA installation, install the sample tests by

```
sudo ./cuda-samples.9.0.176-22781540-linux.run
```

Step 7: Set up paths

After the installation finishes, configure the runtime library.

```
sudo bash -c "echo /usr/local/cuda/lib64/ >
/etc/ld.so.conf.d/cuda.conf"
sudo ldconfig
```

it is also recommended to append string **/usr/local/cuda/bin** to system file **/etc/environments** so that nvcc will be included in $PATH. This will take effect after reboot.
Open **/etc/environments,**

```
sudo vim /etc/environments
```

and add
:/usr/local/cuda/bin (including the ":") at the end of the PATH string.

Step 8: Reboot and test installation

After a reboot, you can test the installation by making and invoking tests:

```
cd /usr/local/cuda-9.0/samples
sudo make
```

It is a long process with many irrelevant warnings.

After it completes, run **deviceQuery**:

```
cd /usr/local/cuda/samples/bin/x86_64/linux/release
./deviceQuery
```

The result of running **deviceQuery** should look something like this:

```
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "GeForce 940MX"
  CUDA Driver Version / Runtime Version          9.0 / 9.0
  CUDA Capability Major/Minor version number:    5.0
  Total amount of global memory:                 2003 MBytes (2100232192
bytes)
  ( 3) Multiprocessors, (128) CUDA Cores/MP:     384 CUDA Cores
  GPU Max Clock rate:                            1242 MHz (1.24 GHz)
  Memory Clock rate:                             1001 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 1048576 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536,
65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048
layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
```

```
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Supports Cooperative Kernel Launch:            No
  Supports MultiDevice Co-op Kernel Launch:      No
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device
simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 9.0, CUDA Runtime
Version = 9.0, NumDevs = 1
Result = PASS
```

Notice that you get a **Result = PASS** at the end.

Step 9: Enable LightDM and Nuoveau drivers

If the installation was successful, you want to enable Lightdm:

```
sudo service lightdm start
```

In addition, enable Nouveau drivers:

Remove the blacklist file created in the Disabling Nouveau section, and regenerate the kernel

```
rm /etc/modprobe.d/blacklist-nouveau.conf
sudo update-initramfs -u
```

Sources:

This guide is based on the official CUDA installation guide:

http://developer.download.nvidia.com/compute/cuda/9.0/Prod/docs/sidebar/CUDA_Installation_Guide_Linux.pdf

In addition, these two posts:

1. https://askubuntu.com/questions/799184/how-can-i-install-cuda-on-ubuntu-16-04
2. https://gist.github.com/zhanwenchen/e520767a409325d9961072f666815bb8