This document details the procedures to set up the docker containers from NGC.

Prerequisites

- cleIf using a new server with NVIDIA GPU without having NVIDIA drivers, CUDA toolkit, Docker CE software, Nvidia-container runtime and NVIDIA Docker stack installed, the below steps need to be completed, assuming you are using **Ubuntu OS.**
- If you do not have an Nvidia GPU, atleast have docker 19.03.

(DO NOT RUN THE BELOW STEPS IF YOU HAVE THE ABOVE STACK ALREADY INSTALLED)

- **Install NVIDIA Drivers**

  Download and install the relevant drivers (Tesla V100 / Tesla T4) from

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

  OR (use the below steps - Preferred)

  ➢ *sudo apt-get purge nvidia**
  ➢ *sudo add-apt-repository ppa:graphics-drivers/ppa*
  ➢ *sudo apt update*
  ➢ *sudo apt install nvidia-driver-450*

  Note: Any driver starting nvidia-driver-410 or later will work.

- **Install CUDA drivers**

  Step 1: Check the currently installed release

  ➢ apt list --installed cuda-toolkit-*

  Step 2: Update the local database with the latest information from the Ubuntu repository.

  ➢ sudo apt update

  Step 3: Show all available CUDA Toolkit releases.

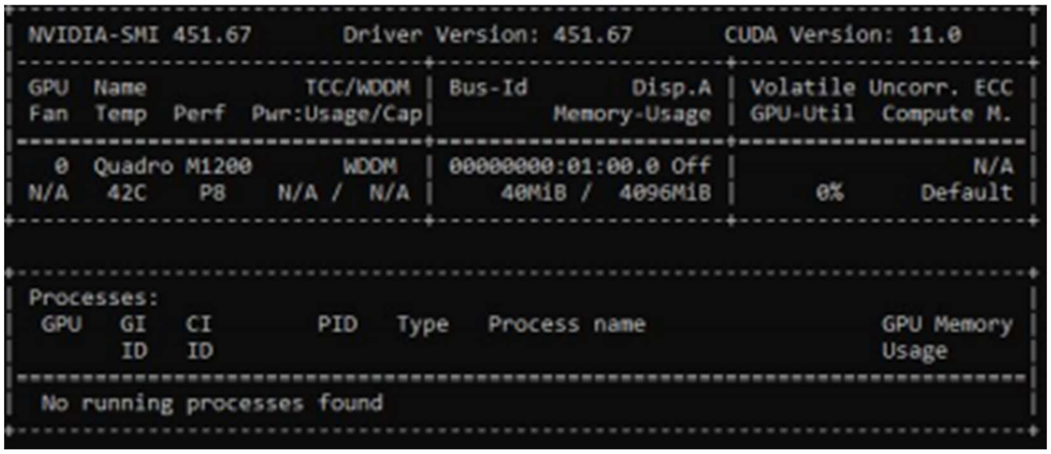
  ➢ apt list cuda-toolkit-*

  Step 4: Install or upgrade the CUDA Toolkit

  ➢ apt install cuda-toolkit-11-2

After performing the above steps (either of the above) **reboot** the instance and check the status of the GPU by using the below:

➢ *nvidia-smi*



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 451.67       Driver Version: 451.67       CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Quadro M1200        WDDM | 00000000:01:00.0 Off |                  N/A |
| N/A   42C    P8    N/A /  N/A |     40MiB /  4096MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

- **Install docker and NVIDIA docker2:**

Check if already installed**: >** ***docker version***

If Not, Install Docker CE using the below steps:

https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

➢ *sudo apt-get update*

➢ *sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common*

➢ *curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - ➢ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"*

➢ *sudo apt-get update*

➢ *sudo apt-get install docker-ce docker-ce-cli containerd.io*

ONLY FOR NVIDIA GPU USER:-

Install nvidia-docker2 using the below steps:
https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html#installing-on ubuntu-and-debian

➢ *curl https://get.docker.com | sh && sudo systemctl start docker && sudo systemctl enable docker*

➢ *distribution=$(. /etc/os-release;echo $ID$VERSION_ID) && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - && curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list*

➢ *curl -s -L https://nvidia.github.io/nvidia-container-*

runtime/experimental/$distribution/nvidia-contain er-runtime.list | sudo tee /etc/apt/sources.list.d/nvidia-container-runtime.list

- ➢ *sudo apt-get update*
- ➢ *sudo apt-get install -y nvidia-docker2*
- ➢ *sudo systemctl restart docker*

Verify the installation of docker and nvidia-docker with following commands:

- ➢ *sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi*
- ➢ *sudo docker run --rm hello-world*

Step wise commands to run jupyter-lab

docker pull frolvlad/alpine-python-machinelearning

1. Show the running container

   $ sudo docker ps

2. Show the docker images
   $ sudo docker images

3. Running the container image
   a. If docker version >= 19.02

General command:

$ docker run --gpus all --rm -v /path/to/data:/workspace/data -it -p 5000:8888 nvcr.io/nvidia/tensorflow:20.03-tf1-py3

Example:

→ Highlighted in red, needed to be change

$ docker run -it --gpus device=0 --rm -v $PWD/data:/workspace/data -p 5000:8888 nvcr.io/nvidia/tensorflow:21.02-tf2-py3

--gpus device = 0

Based on the gpu device you have to use, you can give device as 0, 1, 2, 3(n-1 = device)

-v $PWD/data:/workspace/data

Your present working directory, then add the directory you want to map
e.g. /home/dgx/nvidia → so, docker workspace will be stored in the nvidia directory.

-p 1000:8888

Here, 8888 is the default port and 1000 is the port you want to forward your default port to.

4. Running Jupyter-Notebook

- Execute a JupyterLab-

Install Jupyter lab

```
$ pip install jupyterlab
```

Run jupyter lab in the docker workspace

```
$ jupyter-lab --allow-root --ip=0.0.0.0
```

Note: Copy the token and type IP and port assigned (see docker run) on your browser

a. If on your own system-
 localhost:5000
 Click Enter and paste token

b. If on a different server IP connected with VPN:-
 a.b.c.d:5000
 Click Enter and paste token

***

Additional Commands

If you have container running in the background, then you can follow these steps:
First get the container detail

```
$ docker ps -a
```

Method 1: Stop and remove the running container

```
$ docker stop container_id
$ docker rm container_id
```

Method 2: Start and attach the running container

```
$ docker start container_id
$ docker attach container_id
```