

# WINDOW10에서 WSL2를 활용한 Ubuntu18.04 가상 AI 학습 모듈 세팅

작성자 : CS 김태현 사원

## 1. wsl이란?

▶ Windows Subsystem for Linux(WSL). 버전은 wsl1 과 wsl2가 존재

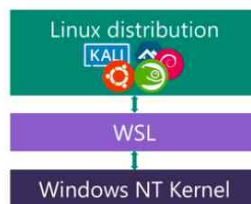
참고 블로그 :

<https://xeppetto.github.io/%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4/WSL-and-Docker/03-What-is-WSL/>

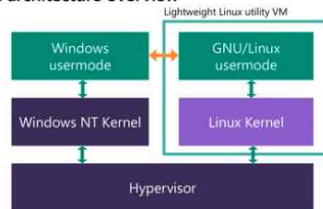
▶ 「WSL은 윈도우 10에서 Linux의 ELF 64를 실행할 수 있는 덕분에 속도는 Virtual Machine 보다 훨씬 빠르고, 재부팅을 해야 할 필요가 없다. 또한 WSL을 사용할 Windows 컴퓨터의 CPU와 Memory 속도만 확보하면 되며, 별도의 서버를 구성하기 위해 비싼 HW 가격을 고려하거나, 속도가 느린 시스템을 사용하느라 고통 받을 필요가 없다..」

▶ WSL 아키텍처

WSL architecture



WSL 2 architecture overview



WSL1과 WSL2의 아키텍처 비교

(Hypervisor 위에서 실행된다는점이 중요!)

**\*\* Hypervisor 위에서 동작하기 때문에, WSL를 사용하면 VM이 동작하지 않음.(테스트 버전 VM에서는 동작된다고 함..(카더라))**

▶ WSL1 과 WSL2의 차이점

개발자 입장에서는 WSL1에서는 docker를 사용할수 없었지만, WSL2에서는 docker를 사용할 수 있다는 점이 매우 중요

## 2. WSL + CUDA Toolkit 설치(2021.06.23.일 기준)

참고:

<https://docs.nvidia.com/cuda/wsl-user-guide/index.html#installing-wip>

### ▶ 윈도우 설정

#### ㄱ. Docker(3.3.0) 버전 설치

- 1) <https://docs.docker.com/docker-for-windows/release-notes/>

#### ㄴ. Microsoft Windows Insider 설정

- 1) Microsoft Windows Insider 프로그램에 등록  
<https://insider.windows.com/en-us/getting-started#register>
- 2) Dev Channel 에서 최신 빌드를 설치  
<https://aka.ms/WIPSettings>

#### ㄷ. 윈도우 버전 확인

- 1) ctrl+R - winver(입력) 해서 OS 빌드 버전이 21332 이상인지 확인

*(We recommend being on WIP OS 21332 and higher with Linux Kernel 5.4.91+ for the best performance.)*

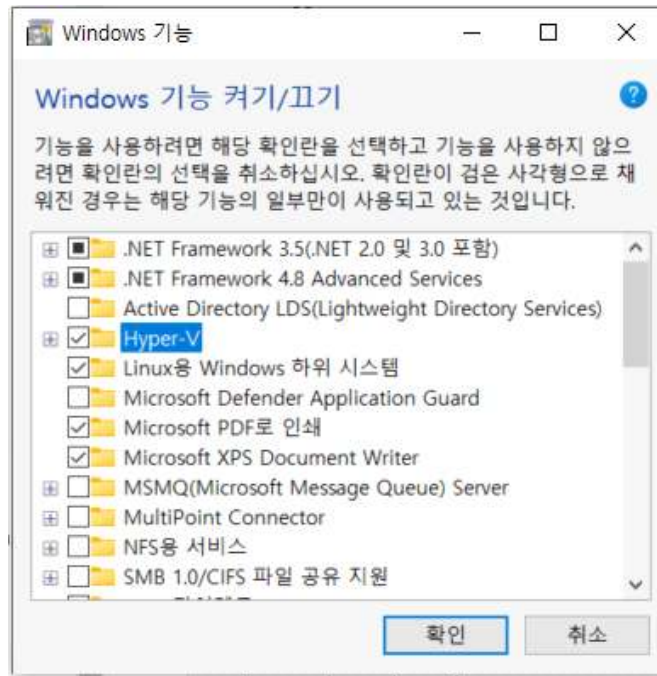


#### ㄹ. WSL용 NVIDIA 드라이버 설치

- 1) <https://developer.nvidia.com/cuda/wsl/download>

## ㄱ. Hyper-V 활성화

1) 설정 - 앱 - 프로그램 및 기능 - Windows 기능 켜기/끄기



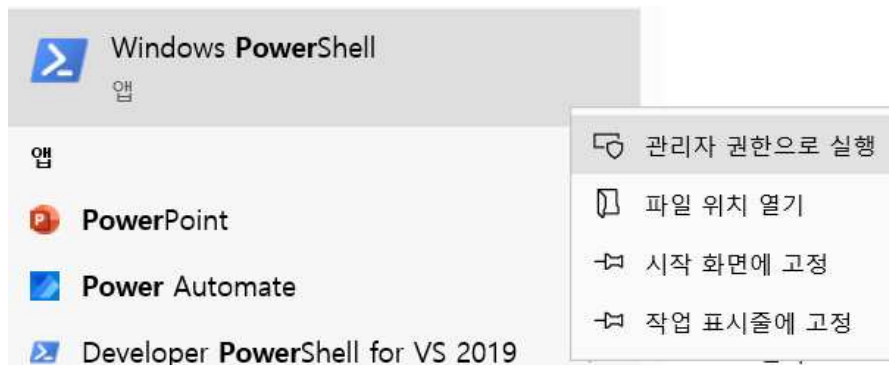
Hyper-V 활성화

## ㄴ. x64 머신용 최신 WSL2 Linux 커널 업데이트 패키지 설치

1) [https://wslstorestorage.blob.core.windows.net/wslblob/wsl\\_update\\_x64.msi](https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi)

## ㄷ. Linux용 Windows 하위 시스템사용, Virtual Machine 플랫폼 옵션 기능 활성화

1) PowerShell 관리자 권한 실행



PowerShell 관리자 권한 실행

2) 명령어 입력

```
> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart  
> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

```
관리자: Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\nacyot> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

배포 이미지 서비스 및 관리 도구
버전: 10.0.19041.1

이미지 버전: 10.0.19041.264

기능을 사용하도록 설정하는 중
[=====100.0%=====]
작업을 완료했습니다.
PS C:\Users\nacyot> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

배포 이미지 서비스 및 관리 도구
버전: 10.0.19041.1

이미지 버전: 10.0.19041.264

기능을 사용하도록 설정하는 중
[=====100.0%=====]
작업을 완료했습니다.
PS C:\Users\nacyot>
```

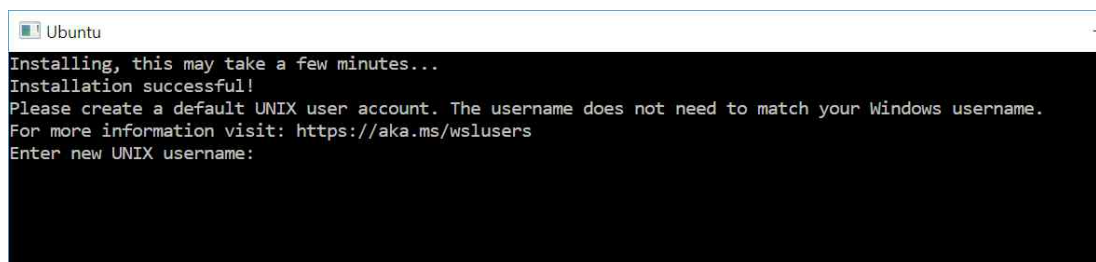
이미지 출처 : <https://www.44bits.io/ko/post/wsl2-install-and-basic-usage>

## □. Microsoft Store에서 WSL용 리눅스 배포판(18.04) 설치

1) <https://www.microsoft.com/store/productId/9N9TNGVNDL3Q> 에서 배포판 설치



스토어 화면(18.04 말고 20.04 버전도 존재)



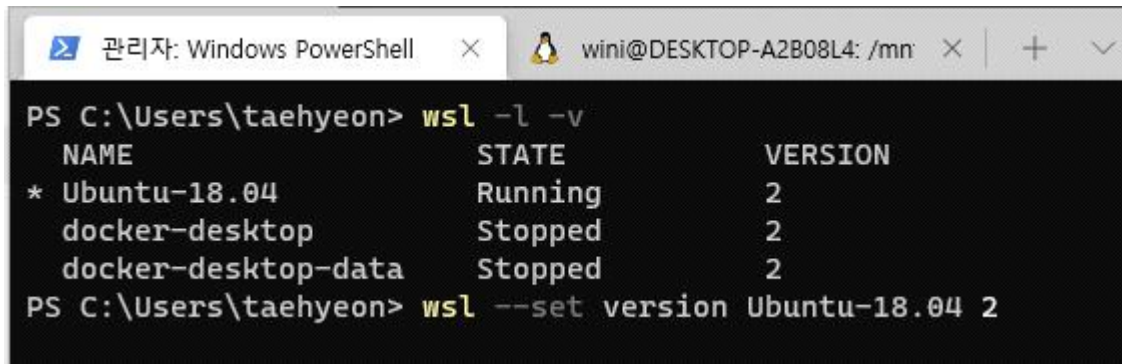
설치시 username 입력 화면

## ㄴ. WSL 2 설정(PowerShell에서)

> wsl --set-version <distribution name> <versionNumber>

\*\* <distribution name> : 설치한 WSL 패키지명

\*\* <versionNumber> : 변경하고자 하는 WSL 버전



```
PS C:\Users\taehyeon> wsl -l -v
NAME                STATE      VERSION
* Ubuntu-18.04      Running    2
  docker-desktop     Stopped    2
  docker-desktop-data Stopped    2
PS C:\Users\taehyeon> wsl --set version Ubuntu-18.04 2
```

> wsl --set-default-version 2      /\*\* wsl 이후 이미지도 WSL2로 설정 \*\*/

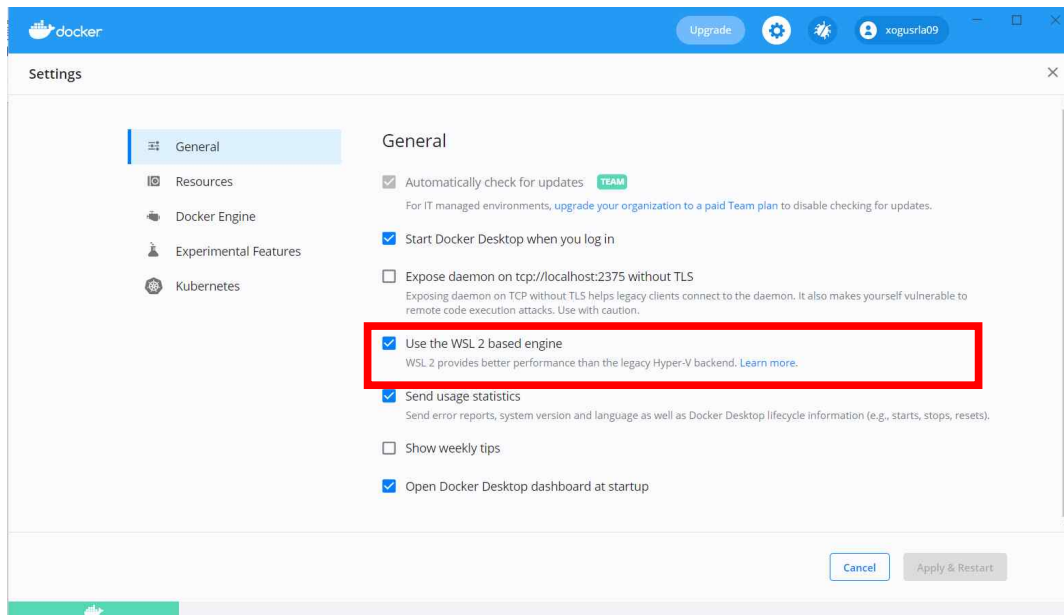
> wsl --update                    /\*\* wsl 업그레이드 \*\*/

참고: Windows 10에 Linux용 Windows 하위 시스템 설치 가이드

<https://docs.microsoft.com/ko-kr/windows/wsl/install-win10>

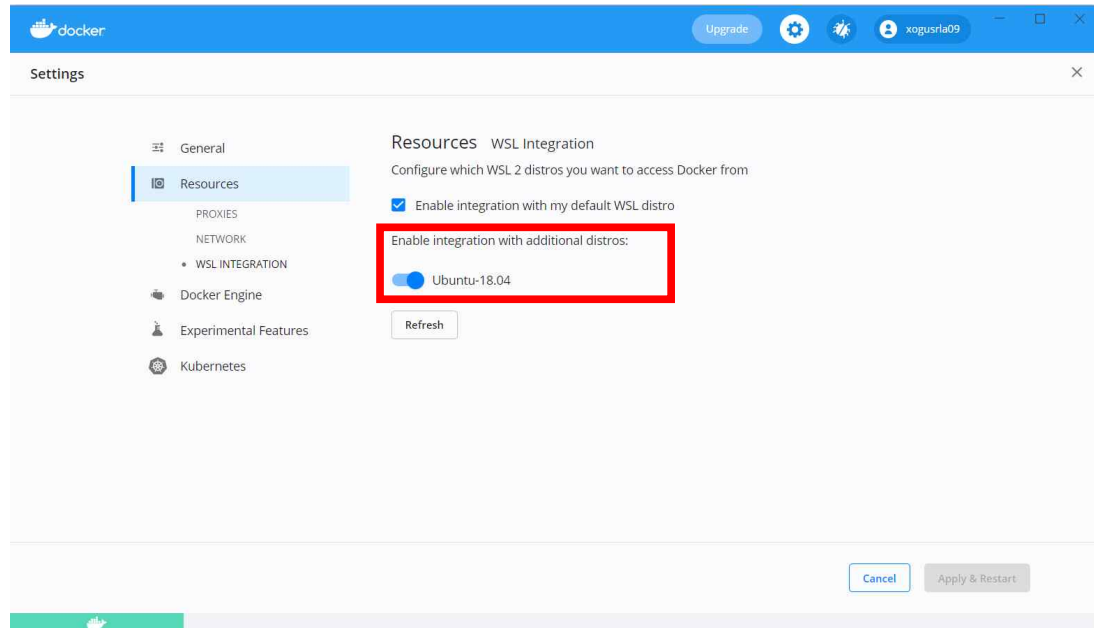
## ㄷ. window docker desktop 설정

> docker setting - General에서 Use the WSL 2 based engine 콤보박스 체크



WSL2 설정

> docker setting - Resources - WSL INTERGRATION에서 해당 이미지 체크



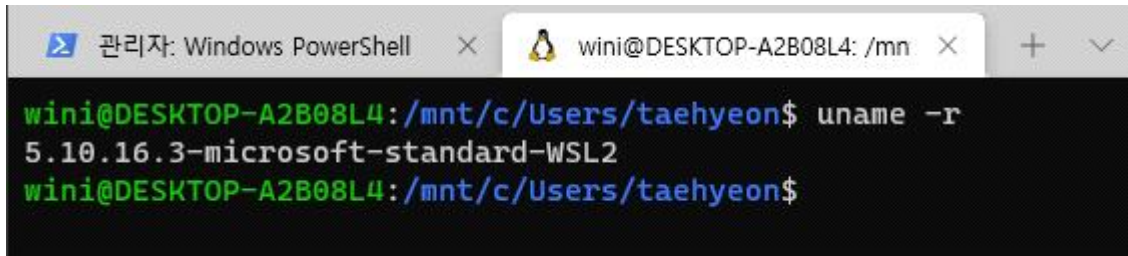
## ▶ WSL2 Ubuntu 설정

\*\* Optional : window Terminal 설치 추천

<https://www.microsoft.com/store/productId/9N0DX20HK701>

## ㄱ. Ubuntu WSL 버전 확인

1) \$ `uname -r` 이후 출력되는 버전이 4.19.121 보다 높은지 확인

A screenshot of a Windows PowerShell terminal window. The title bar shows '관리자: Windows PowerShell' and a tab for 'wini@DESKTOP-A2B08L4: /mnt'. The terminal content shows the command 'uname -r' being executed, with the output '5.10.16.3-microsoft-standard-WSL2'. The prompt is 'wini@DESKTOP-A2B08L4:/mnt/c/Users/taehyeon\$'.

버전 출력 확인

## ㄴ. CUDA Toolkit 설치(설치버전은 10.2로 설치)

```
$ apt-key adv --fetch-keys
```

```
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub
```

```
$ sh -c 'echo "deb http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 /"
> /etc/apt/sources.list.d/cuda.list'
```

```
$ apt-get update
```

```
$ apt-get install -y cuda-toolkit-10-2
```

## ㄷ. CUDA Toolkit 테스트

/usr/local/cuda/samples/4\_Finance/BlackScholes 경로에서

`sudo make` 이후

`./BlackScholes` 실행

## ㄹ. docker-ce 설치

```
>> curl https://get.docker.com | sh
```

## ㅁ. NVIDIA Container Toolkit 설치

1) 환경설정 저장소

```
$ distribution = $(cat /etc/os-release; echo $ID $VERSION_ID)
```

```
$ curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key 추가-
```

```
$ 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/libnvidia-container/experimental/$distribution/libnvidia-container-experimental.list | sudo tee /etc/apt/sources.list.d/libnvidia-container-experimental.list
```

## 2) 패키지 목록 업데이트

```
$ sudo apt-get update
```

## 3) NVIDIA Runtime Package 설치

```
$ sudo apt-get install -y nvidia-docker2
```

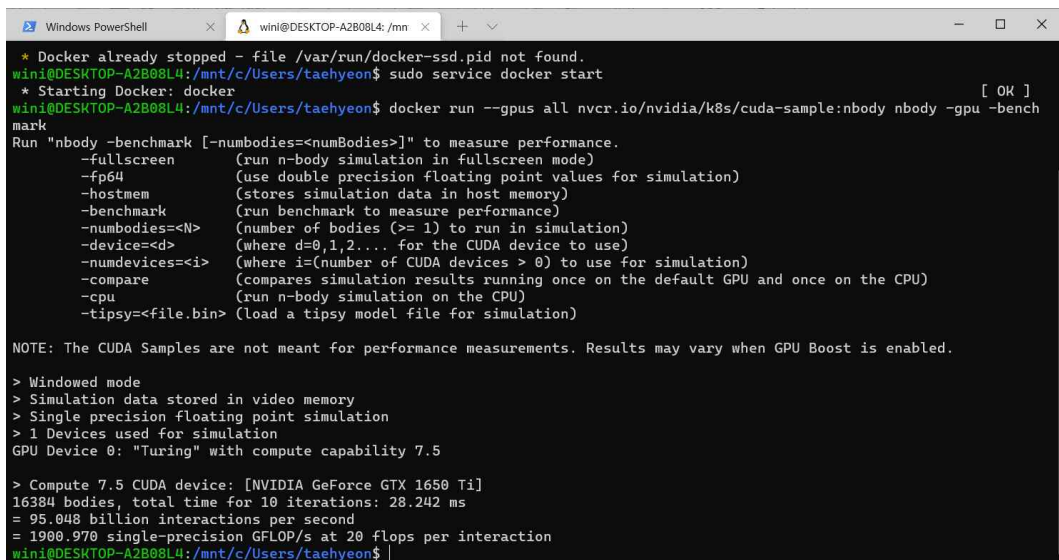
## 4) WSL Docker 데몬 재시작

```
$ sudo service docker stop
```

```
$ sudo service docker start
```

## 5) CUDA 컨테이너 테스트

```
$ docker run --gpus all nvcr.io/nvidia/k8s/cuda-sample:nbody nbody -gpu -benchmark
```



```
Windows PowerShell
wini@DESKTOP-A2B08L4: /mnt/c/Users/taehyeon$ sudo service docker start
* Docker already stopped - file /var/run/docker-ssd.pid not found.
wini@DESKTOP-A2B08L4: /mnt/c/Users/taehyeon$ sudo service docker start
* Starting Docker: docker
wini@DESKTOP-A2B08L4: /mnt/c/Users/taehyeon$ docker run --gpus all nvcr.io/nvidia/k8s/cuda-sample:nbody nbody -gpu -benchmark
mark
Run "nbody -benchmark [-numbodies=<numBodies>]" to measure performance.
  -fullscreen      (run n-body simulation in fullscreen mode)
  -fp64            (use double precision floating point values for simulation)
  -hostmem         (stores simulation data in host memory)
  -benchmark      (run benchmark to measure performance)
  -numbodies=<N>   (number of bodies (>= 1) to run in simulation)
  -device=<d>      (where d=0,1,2,... for the CUDA device to use)
  -numdevices=<i>  (where i=(number of CUDA devices > 0) to use for simulation)
  -compare         (compares simulation results running once on the default GPU and once on the CPU)
  -cpu            (run n-body simulation on the CPU)
  -tipsy=<file.bin> (load a tipsy model file for simulation)

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.

> Windowed mode
> Simulation data stored in video memory
> Single precision floating point simulation
> 1 Devices used for simulation
GPU Device 0: "Turing" with compute capability 7.5

> Compute 7.5 CUDA device: [NVIDIA GeForce GTX 1650 Ti]
16384 bodies, total time for 10 iterations: 28.242 ms
= 95.048 billion interactions per second
= 1900.970 single-precision GFLOP/s at 20 flops per interaction
wini@DESKTOP-A2B08L4: /mnt/c/Users/taehyeon$
```

화면 표출 내용

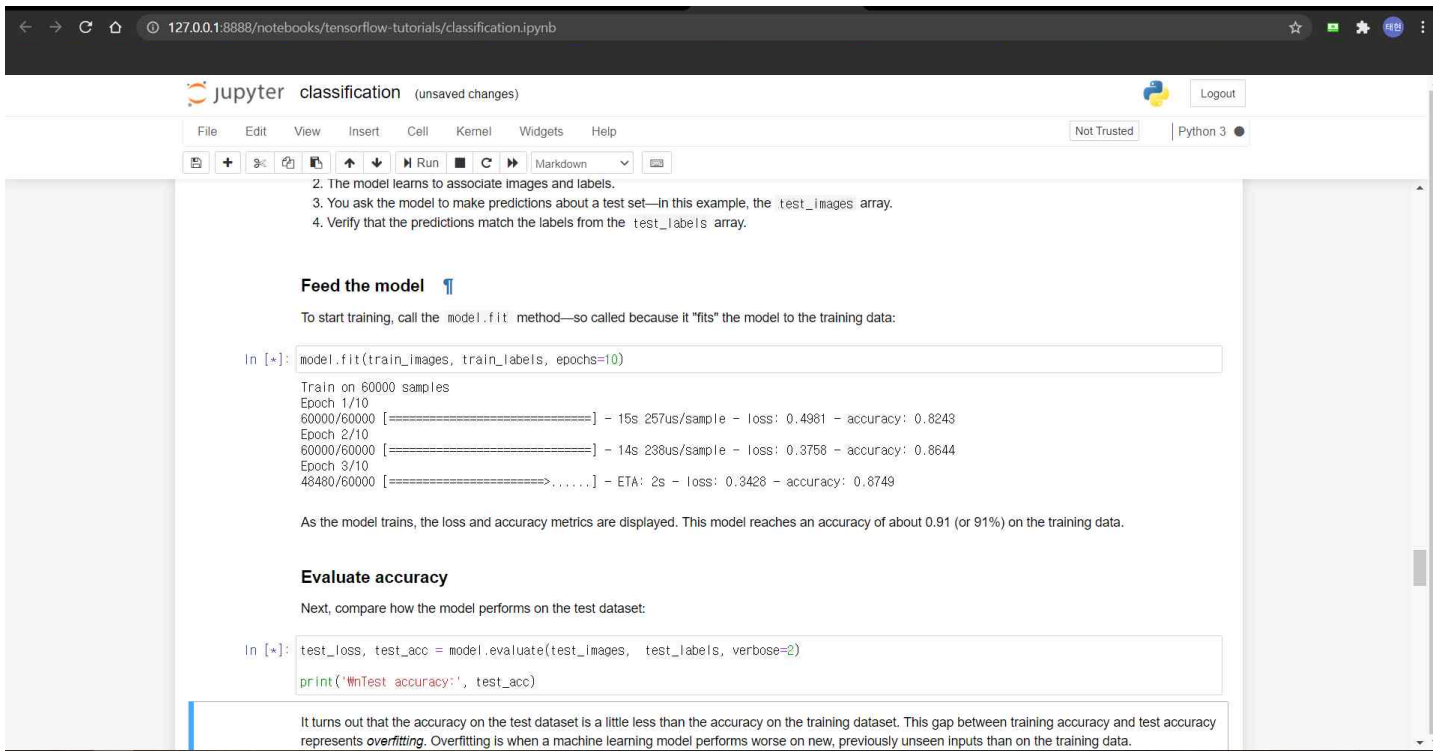
설치 참고 : <https://docs.nvidia.com/cuda/wsl-user-guide/index.html>

## ▶ 추가) GUI Application을 위한 설정

설치 참고: <https://docs.microsoft.com/ko-kr/windows/wsl/tutorials/gui-apps>



## ▶ WSL2 + CUDA를 통한 학습 GPU 로드율 100% 확인



The screenshot shows a Jupyter Notebook interface with the following content:

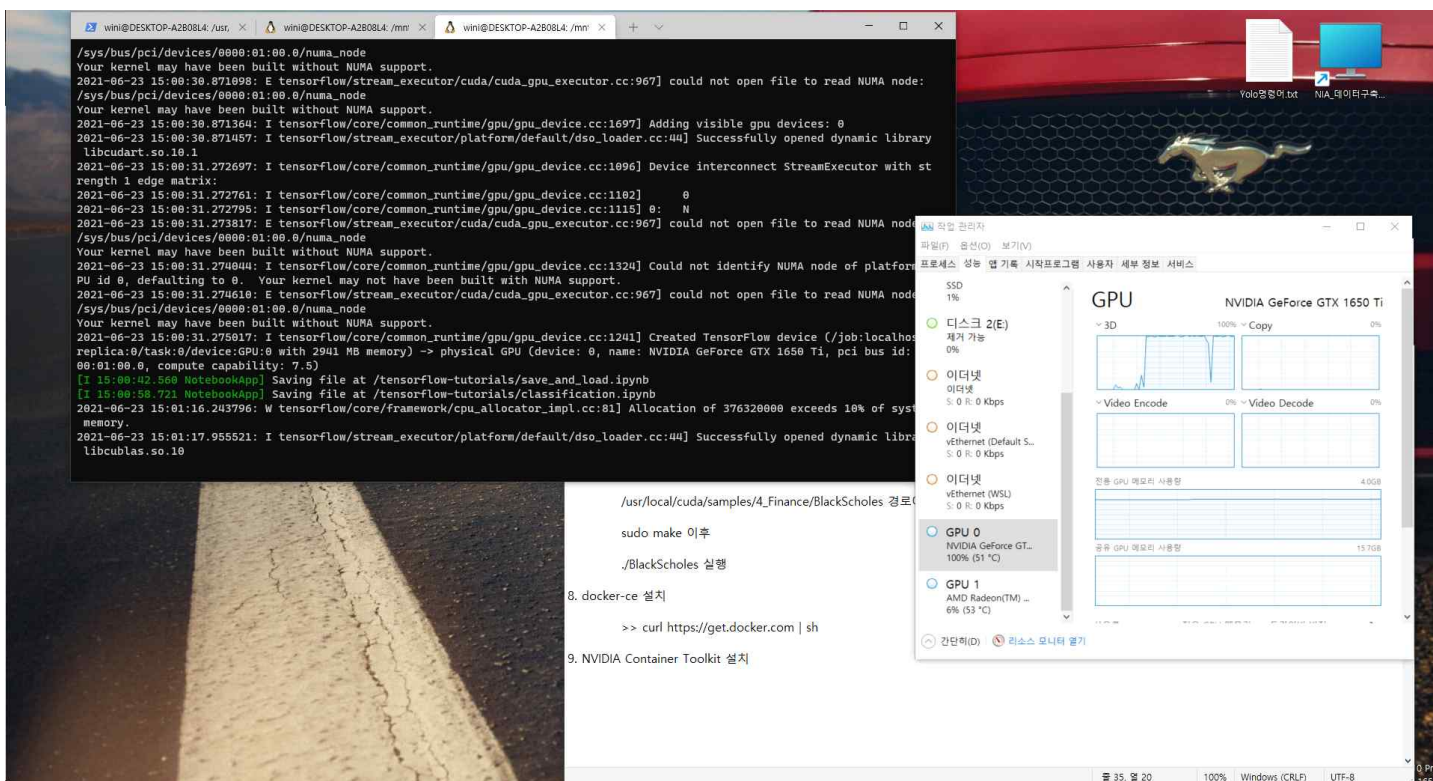
- Instructions: 2. The model learns to associate images and labels. 3. You ask the model to make predictions about a test set—in this example, the `test_images` array. 4. Verify that the predictions match the labels from the `test_labels` array.
- Section: **Feed the model**
- Text: To start training, call the `model.fit` method—so called because it “fits” the model to the training data:
- Code cell:

```
model.fit(train_images, train_labels, epochs=10)
```
- Output:

```
Train on 60000 samples
Epoch 1/10
60000/60000 [=====] - 15s 257us/sample - loss: 0.4981 - accuracy: 0.8243
Epoch 2/10
60000/60000 [=====] - 14s 238us/sample - loss: 0.3758 - accuracy: 0.8644
Epoch 3/10
48480/60000 [=====>.....] - ETA: 2s - loss: 0.3428 - accuracy: 0.8749
```
- Text: As the model trains, the loss and accuracy metrics are displayed. This model reaches an accuracy of about 0.91 (or 91%) on the training data.
- Section: **Evaluate accuracy**
- Text: Next, compare how the model performs on the test dataset:
- Code cell:

```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print('Test accuracy:', test_acc)
```
- Text: It turns out that the accuracy on the test dataset is a little less than the accuracy on the training dataset. This gap between training accuracy and test accuracy represents *overfitting*. Overfitting is when a machine learning model performs worse on new, previously unseen inputs than on the training data.

## Jupyter를 통한 간단한 학습 실행



The screenshot shows a Windows desktop environment with the following elements:

- A terminal window running WSL2 commands to set up the environment:

```
/sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-06-23 15:00:30.871098: E tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:967] could not open file to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-06-23 15:00:30.871364: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1697] Adding visible gpu devices: 0
2021-06-23 15:00:30.871457: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudart.so.10.1
2021-06-23 15:00:31.272697: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1096] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-06-23 15:00:31.272761: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] 0
2021-06-23 15:00:31.272795: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] 0: N
2021-06-23 15:00:31.273817: E tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:967] could not open file to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-06-23 15:00:31.274004: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1324] Could not identify NUMA node of platform PU id 0, defaulting to 0. Your kernel may not have been built with NUMA support.
2021-06-23 15:00:31.274610: E tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:967] could not open file to read NUMA node: /sys/bus/pci/devices/0000:01:00.0/numa_node
Your kernel may have been built without NUMA support.
2021-06-23 15:00:31.275017: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1241] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 2941 MB memory) -> physical GPU (device: 0, name: NVIDIA GeForce GTX 1650 Ti, pci bus id: 00:01:00.0, compute capability: 7.5)
[15:00:42.350 NotebookApp] Saving file at /tensorflow-tutorials/save_and_load.ipynb
[15:00:45.721 NotebookApp] Saving file at /tensorflow-tutorials/classification.ipynb
2021-06-23 15:01:16.243796: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 376320000 exceeds 18% of system memory.
2021-06-23 15:01:17.955521: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10
```
- A command prompt window showing the execution of the following commands:

```
/usr/local/cuda/samples/4_Finance/BlackScholes 경론
sudo make 이후
./BlackScholes 실행
8. docker-ce 설치
>> curl https://get.docker.com | sh
9. NVIDIA Container Toolkit 설치
```
- A GPU monitoring tool window showing the status of the NVIDIA GeForce GTX 1650 Ti GPU. The GPU is at 100% utilization (51°C) and is running at 6% (53°C). The tool also shows the status of the SSD, disk, and network.

GPU 로드율 100%

## ▶ wsl\_docker 오류

<https://forums.developer.nvidia.com/t/failure-to-install-cuda-on-wsl-2-ubuntu/128592/64> 참고

- libnvidia-container1을 버전 1.3.3으로 다운 그레이드하거나 DockerDesktop을 3.3.0로 다운 그레이드하거나 nvidia GPU 드라이버 업데이트를 기다려야합니다.(원문 해석)

- 패키지 재설치 명령어

```
sudo apt-get install \  
libnvidia-container1=1.3.3~rc.2-1 \  
libnvidia-container-tools=1.3.3~rc.2-1 \  
nvidia-container-toolkit=1.4.1-1 \  
nvidia-container-runtime=3.4.1-1 \  
nvidia-docker2=2.5.0-1
```

## ▶ docker 명령어 sudo 권한 부여

```
>> sudo adduser $USER docker
```

## ▶ WSL2 문제점

- ✓ Hypervisor 위에 동작하므로 다른 VM이 동작 하지 않음
- ✓ 윈도우를 개발버전으로 올려야 하므로 다른 프로그램에 영향