

# GPU Computing

## Preparing for CUDA Programming

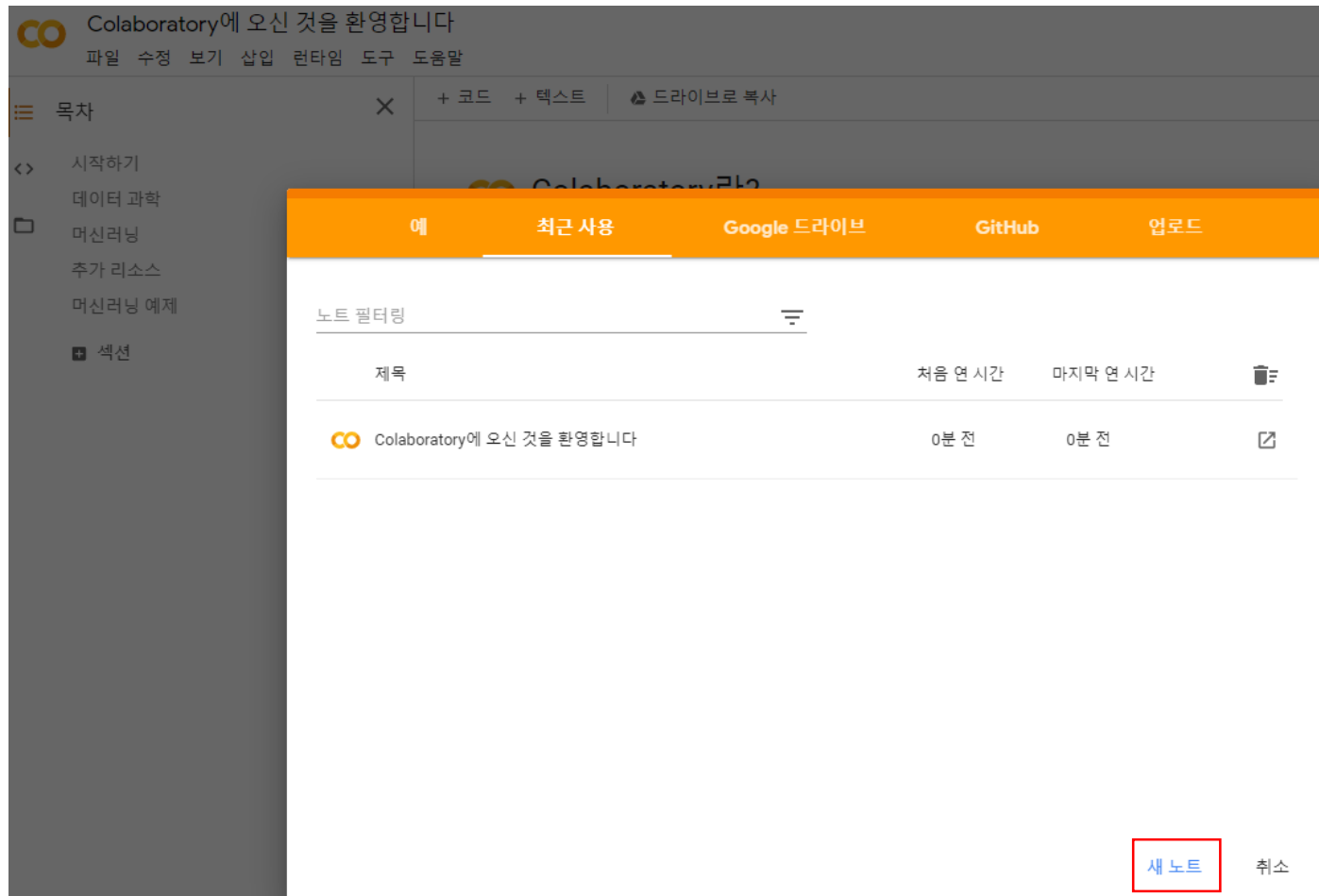
Young-Ho Gong



광운대학교  
KwangWoon University

# CUDA Install without Nvidia GPU

- Nvidia GPU가 없을 경우, Google Colab 을 사용
  - <https://colab.research.google.com/> 사이트로 접속하고 로그인하면 아래 창에서 '새 노트' 를 클릭



# CUDA Install without Nvidia GPU

- Nvidia GPU가 없을 경우, Google Colab 을 사용
  - 런타임 → 런타임 유형 변경
  - 노트설정에서 하드웨어 가속기를 GPU로 변경

The screenshot displays the Google Colab interface for a notebook titled 'Untitled0.ipynb'. On the left, the 'Runtime' menu is open, showing various execution options. The option '런타임 유형 변경' (Change runtime type) is highlighted with a red box. On the right, the '노트 설정' (Notebook Settings) panel is visible. In this panel, the '런타임 유형' (Runtime type) is set to 'Python 3'. The '하드웨어 가속기' (Hardware accelerator) dropdown menu is also highlighted with a red box, and it is currently set to 'GPU'. A blue question mark icon is visible next to the 'GPU' selection. Below the dropdown, there is a checkbox labeled '이 노트를 저장할 때 코드 셀 출력 생략' (Omit code cell output when saving this notebook), which is currently unchecked. At the bottom right of the settings panel, there are buttons for '취소' (Cancel) and '저장' (Save).

Google Colab interface showing the 'Runtime' menu and the 'Notebook Settings' panel.

The 'Runtime' menu options include:

- 모두 실행 (Ctrl+F9)
- 이전 셀 실행 (Ctrl+F8)
- 초점이 맞춰진 셀 실행 (Ctrl+Enter)
- 선택항목 실행 (Ctrl+Shift+Enter)
- 이후 셀 실행 (Ctrl+F10)
- 실행 중단 (Ctrl+M I)
- 런타임 다시 시작... (Ctrl+M .)
- 다시 시작 및 모두 실행...
- 런타임 초기화
- 런타임 유형 변경** (highlighted)
- 세션 관리
- 런타임 로그 보기

The 'Notebook Settings' panel shows:

- 런타임 유형: Python 3
- 하드웨어 가속기: **GPU** (highlighted)
- ☐ 이 노트를 저장할 때 코드 셀 출력 생략

Buttons at the bottom right: 취소 (Cancel), 저장 (Save)

# CUDA Install without Nvidia GPU

- NVCC (CUDA Compiler 설치 확인)

- `!/usr/local/cuda/bin/nvcc --version`

- 위 명령어를 타이핑 후, CTRL+Enter 혹은 왼쪽의 Play 모양 버튼 click

```
[ ] 1 !/usr/local/cuda/bin/nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Wed_Jul_22_19:09:09_PDT_2020
Cuda compilation tools, release 11.0, V11.0.221
Build cuda_11.0_bu.TC445_37.28845127_0
```

- 위와 같이 출력되면 OK  
(NVCC Compiler 가 정상적으로 동작하는 환경)

# CUDA Install without Nvidia GPU

## ■ Hello CUDA 코드 작성

– %%writefile cudahello.cu

- cudahello.cu 라는 파일에 아래의 내용을 작성하도록 지시

– Line 3~Line 16 까지의 코드가 cudahello.cu 파일에 저장됨

```
1 %%writefile cudahello.cu
2
3 #include <iostream>
4 #include <cuda.h>
5 using namespace std;
6
7 __global__ void cuda_hello(){
8     printf("Hello World from GPU!\n");
9 }
10
11 int main(){
12     printf("Hello World from CPU!\n");
13     cuda_hello<<<1,1>>> ();
14     cudaDeviceSynchronize();
15     return 0;
16 }
```

# CUDA Install without Nvidia GPU

## ■ Hello CUDA 코드 작성

– %%writefile cudahello.cu

- cudahello.cu 라는 파일에 아래의 내용을 작성하도록 지시

– Line 3~Line 16 까지의 코드가 cudahello.cu 파일에 저장됨

```
1 %%writefile cudahello.cu
2
3 #include <iostream>
4 #include <cuda.h>
5 using namespace std;
6
7 __global__ void cuda_hello(){
8     printf("Hello World from GPU!\n");
9 }
10
11 int main(){
12     printf("Hello World from CPU!\n");
13     cuda_hello<<<1,1>>> ();
14     cudaDeviceSynchronize();
15     return 0;
16 }
```

# CUDA Install without Nvidia GPU

## ■ Hello CUDA 코드 컴파일

– NVCC 를 이용하여, cudahello.cu 파일을 compile

- 아래 명령어 줄에서, -arch=sm\_35 의 경우, cuda capability 를 sm\_35 버전으로 컴파일 하는 명령어 줄에 해당함

```
1 !nvcc -o cudahello -arch=sm_35 cudahello.cu
```

- 최신 버전에서는 GPU 커널 함수 내에서의 printf 구문을 지원하지 않기 때문에, -arch=sm\_35 없이 컴파일 시, printf 출력 확인 불가

– 컴파일된 binary 실행

```
1 !./cudahello
```

- 결과

```
Hello World from CPU!  
Hello World from GPU!
```

← Main function 의 printf 수행 결과  
← cuda\_hello function 의 printf 수행 결과

# Thank you

Any questions?



**광운대학교**  
KwangWoon University



# Backup slides

CUDA Install in NVIDIA GPU-enabled Systems



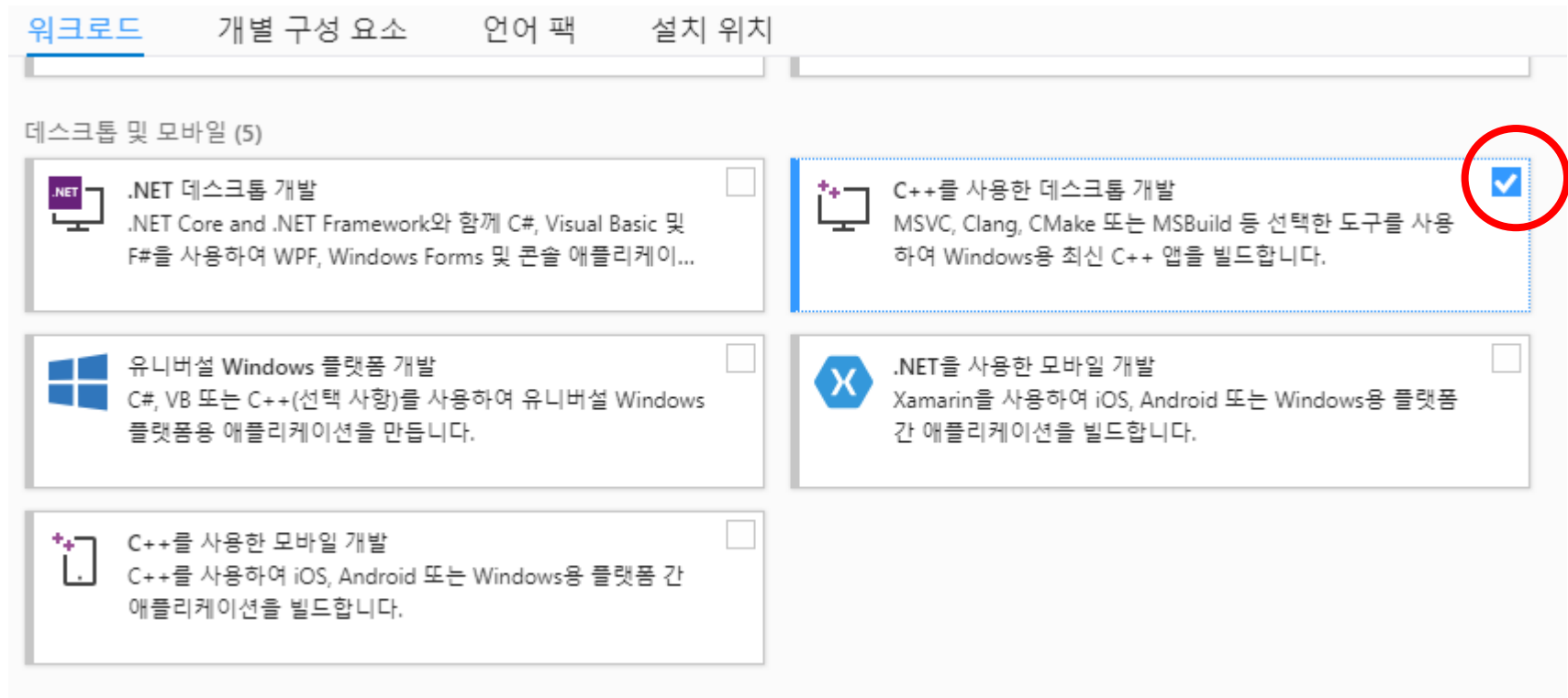
**광운대학교**  
KwangWoon University

King-Ho Gong - GPU

Computing

# CUDA Install with Nvidia GPU

- Install Visual Studio 2019
  - <https://visualstudio.microsoft.com/ko/vs/community/>
  - Reboot before installing CUDA SDK.
- Visual studio 설치 후, 워크로드 설치



# CUDA Install with Nvidia GPU

- Install CUDA SDK

- [https://developer.nvidia.com/cuda-downloads?target\\_os=Windows&target\\_arch=x86\\_64&target\\_version=10](https://developer.nvidia.com/cuda-downloads?target_os=Windows&target_arch=x86_64&target_version=10)

## Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System

Windows

Linux

Mac OSX

Architecture

x86\_64

Version

10

8.1

7

Server 2019

Server 2016

Server 2012 R2

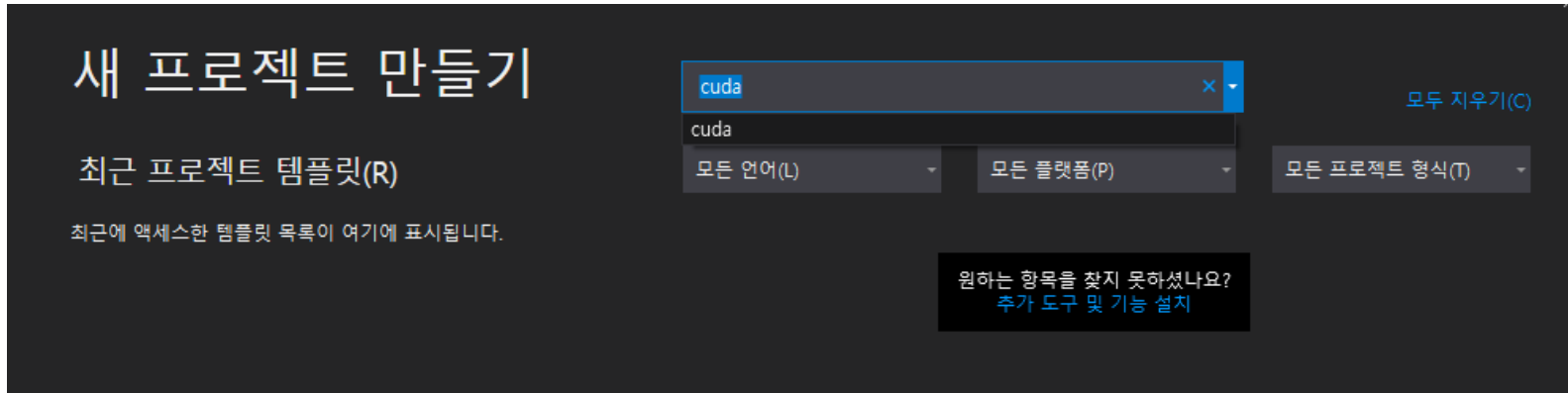
Installer Type

exe (network)

exe (local)

# CUDA Install with Nvidia GPU

- 아래와 같이 Visual Studio 에서 CUDA project 만들기가 보이지 않을 경우



- <https://bigcode.tistory.com/47>
- 위 사이트의 guide 참조
- 주의사항: 설치한 CUDA SDK 버전에 맞게 적용 (10.2 ver 일 경우, 10.1 → 10.2 로 바꾸어서 위 guide 적용)

# CUDA Install with Nvidia GPU

## ■ 1.Build Extension 옮기기

The image shows two Windows File Explorer windows. The top window, titled 'MSBuildExtensions', displays the contents of the 'visual\_studio\_integration' folder. It contains four files: 'CUDA 10.2.props' (14KB), 'CUDA 10.2.targets' (48KB), 'CUDA 10.2' (29KB), and 'Nvda.Build.CudaTasks.v10.2.dll' (260KB). A large black arrow points from this window to the bottom window. The bottom window, titled 'BuildCustomizations', shows the 'Microsoft > VC > v160' folder, which contains 'ImageContentTask.props' (1KB) and 'ImageContentTask.targets' (4KB).

**MSBuildExtensions**

이름	수정한 날짜	유형	크기
CUDA 10.2.props	2019-10-24 오후 5:28	Project Property F...	14KB
CUDA 10.2.targets	2019-10-24 오후 5:28	Project Targets File	48KB
CUDA 10.2	2019-10-24 오후 5:28	XML 문서	29KB
Nvda.Build.CudaTasks.v10.2.dll	2019-10-24 오후 5:28	응용 프로그램 확장	260KB

**BuildCustomizations**

이름	수정한 날짜	유형	크기
ImageContentTask.props	2020-03-16 오후 3:21	Project Property F...	1KB
ImageContentTask.targets	2020-03-16 오후 3:21	Project Targets File	4KB

# CUDA Install with Nvidia GPU

## ■ 2. CUDA Wizards 디렉토리 생성

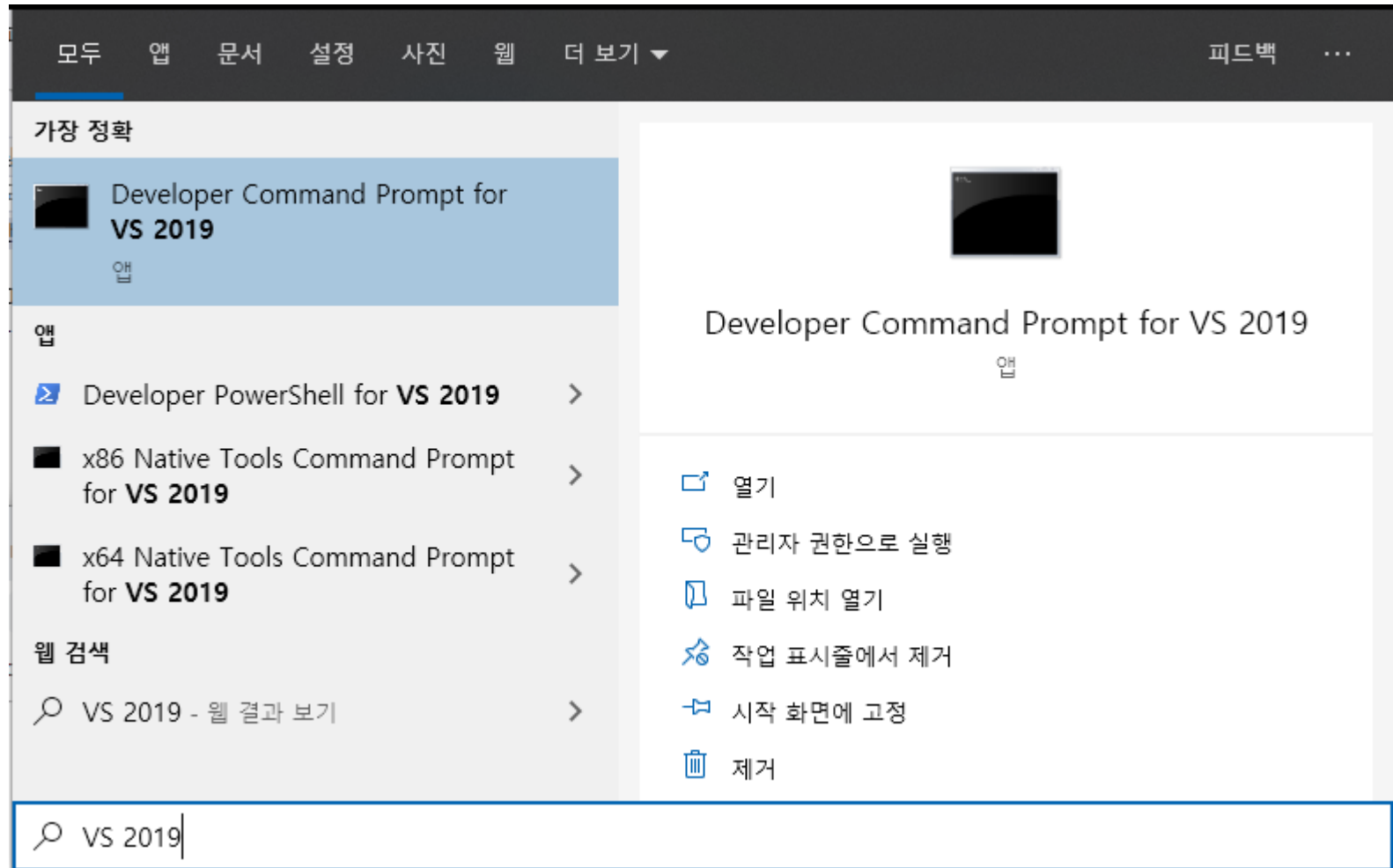
- C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\Extensions\NVIDIA
- 위 경로로 들어가서,
- "CUDA 10.2 Wizards\10.2" 디렉토리를 만듭니다.
  - CUDA 10.2 Wizards 폴더 아래에 10.2 폴더를 만드세요.

## ■ 3. Wizard 관련 바로가기 생성

- 아래 경로 파일의 바로가기를 우선 바탕화면에 만들어둡니다.
- C:\ProgramData\NVIDIA GPU Computing Toolkit\v10.2\extras\visual\_studio\_integration\CudaProjectVsWizards\2019\extension.vsixmanifest
- C:\ProgramData\NVIDIA GPU Computing Toolkit\v10.2\extras\visual\_studio\_integration\CudaProjectVsWizards\Nvda.Vsip.CudaWizards.dll.pkgdef
- 이후, 해당 바로가기들을 2번에서 생성한 10.2 폴더에 복사합니다.

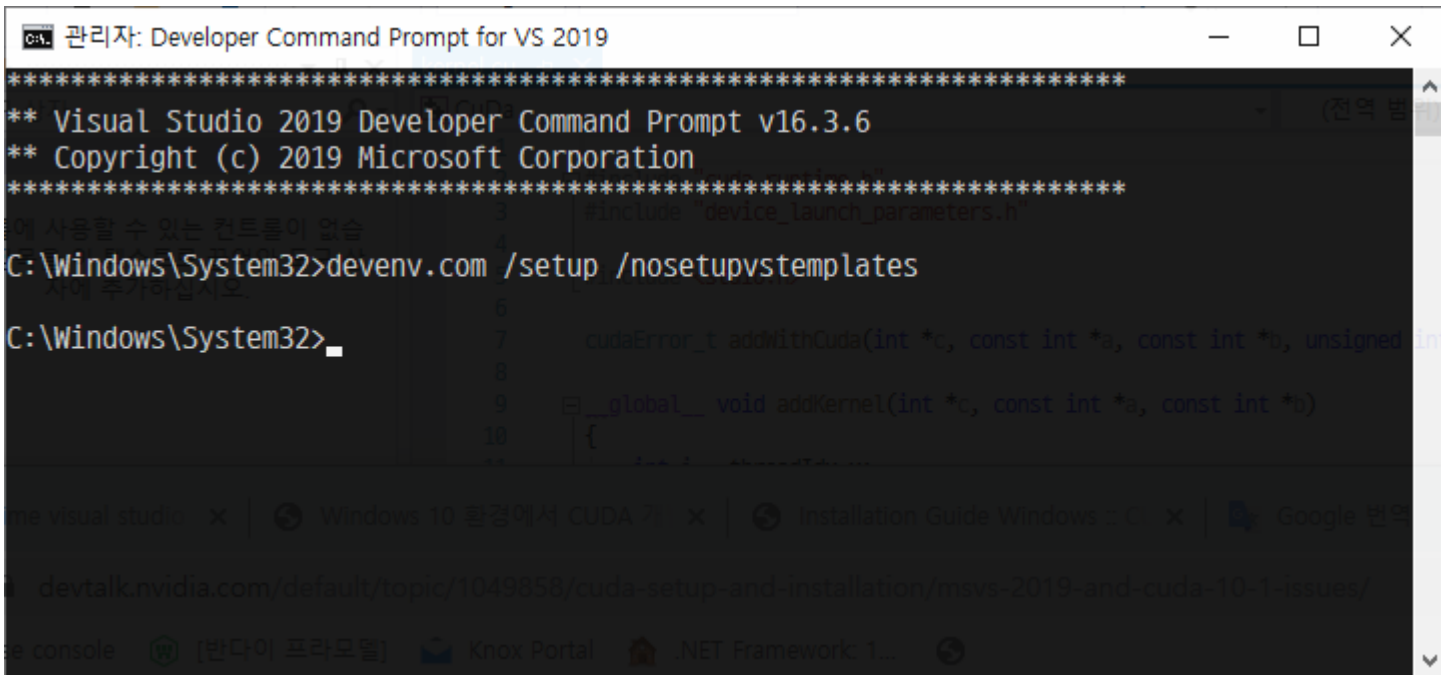
# CUDA Install with Nvidia GPU

## ■ 4. VS 2019 프롬프트 관리자 권한 실행



# CUDA Install with Nvidia GPU

- 5. 프롬프트에서 아래 명령어 입력
  - devenv.com /setup /nosetupvstemplates

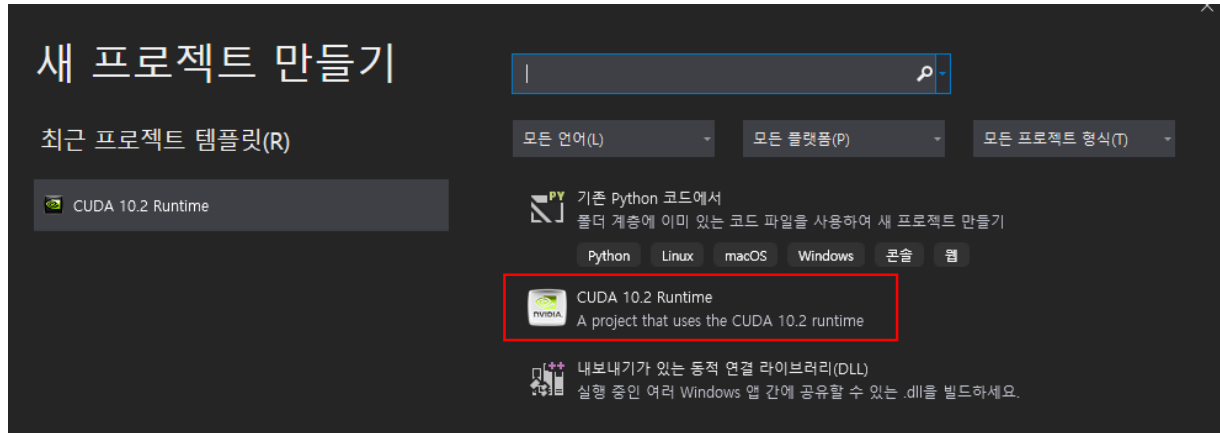


```
관리자: Developer Command Prompt for VS 2019
*****
** Visual Studio 2019 Developer Command Prompt v16.3.6
** Copyright (c) 2019 Microsoft Corporation
*****
C:\Windows\System32>devenv.com /setup /nosetupvstemplates
C:\Windows\System32>
```



# CUDA Install with Nvidia GPU

- 6. CUDA 10.2 runtime 프로젝트 생성
  - VS 2019 실행 → 새 프로젝트 만들기 → CUDA 10.2 Runtime



- 7. 기존 코드를 삭제하고, Hello world code 수행으로 테스트

```
#include <stdio.h>

__global__ void cuda_hello() {
    printf("Hello World from GPU!\n");
}

int main() {
    cuda_hello <<<1,1>>> ();
    return 0;
}
```



선택 Microsoft Visual Studio 디버그 콘솔

Hello World from GPU!