LLM / vLLM 로컬 GPU 환경 세팅 가이드 (Windows + WSL2)

Windows 환경에서 WSL2 + NVIDIA GPU를 활용하여 vLLM 기반 LLM 서빙 서버를 구성

---

0. 사전 요구사항

- Windows 11
- NVIDIA GPU (GeForce RTX 5090)
- NVIDIA GPU Driver

---

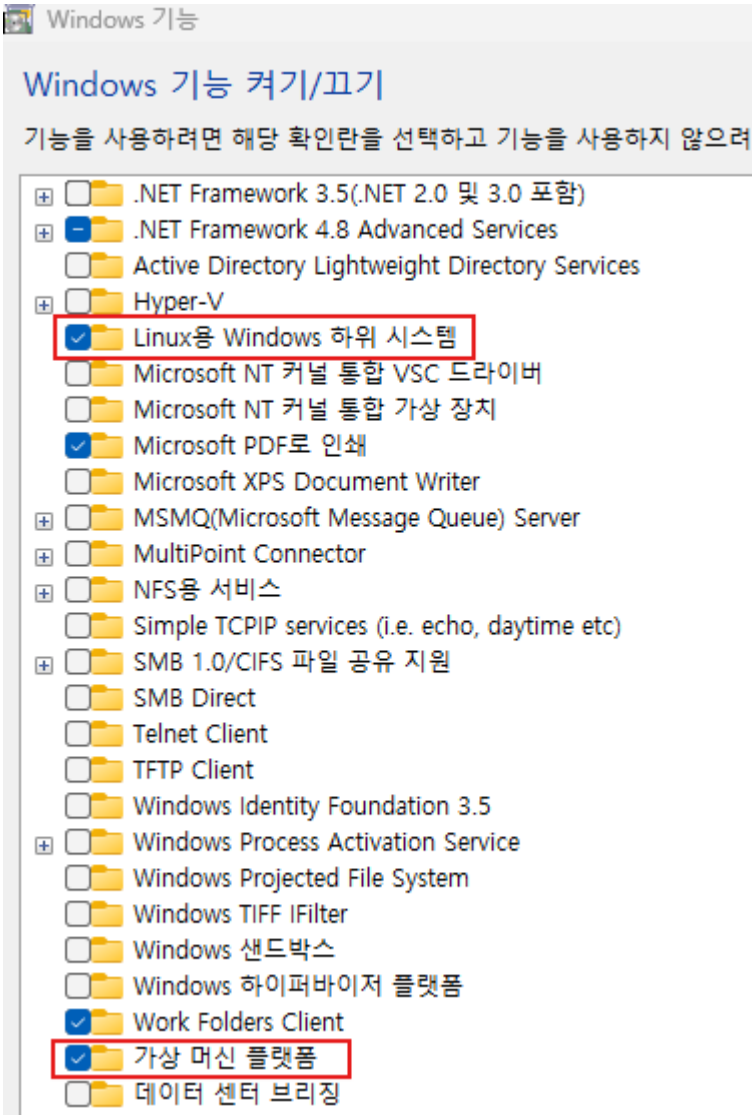
1. WSL2 설치 (Windows Subsystem for Linux)

1-1. Windows 기능 설정

제어판 → 프로그램 → 프로그램 및 기능 → Windows 기능 켜기/끄기

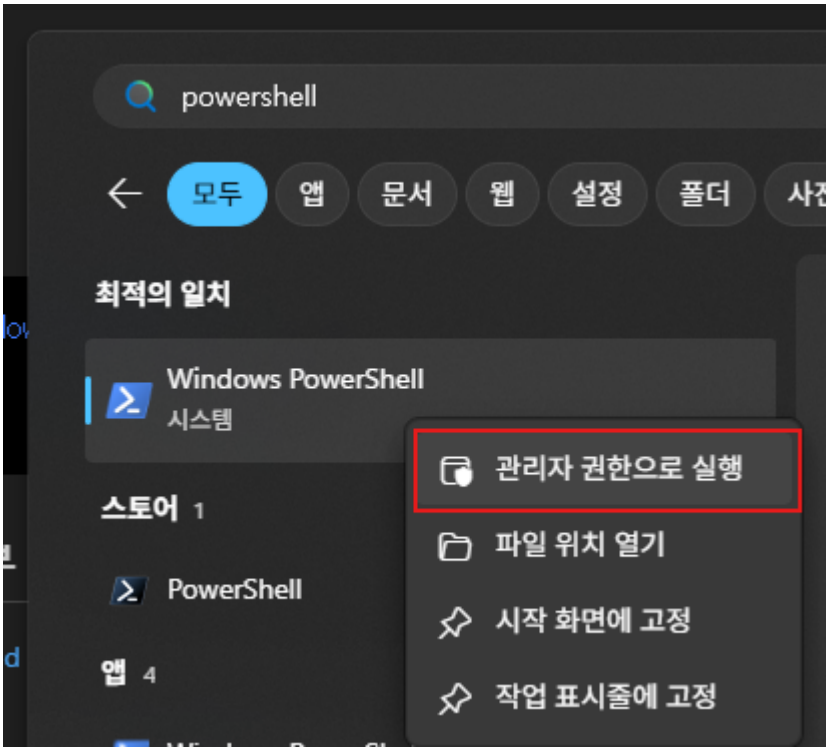
아래 항목 체크 후 **재부팅**:

- Linux용 Windows 하위 시스템
- 가상 머신 플랫폼



1-2. WSL2 설정

- PowerShell 관리자 권한으로 실행



```
# WSL 기본 버전을 2로 설정
wsl --set-default-version 2

# 설치 가능한 배포판 목록 확인
wsl -l -o

# Ubuntu 24.04 설치 ( 설치 도중 Unix Account 생성 )
wsl --install -d Ubuntu-24.04
```

1-3. 기본 사용 명령어

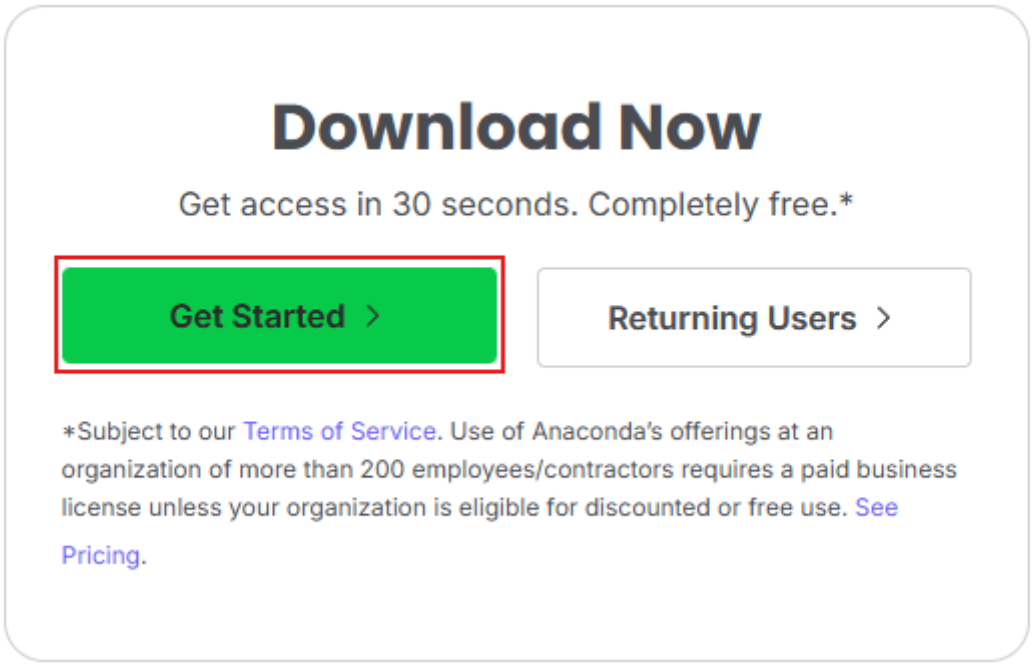
- Linux 실행 : wsl
- Linux 종료 : exit
- 설치 확인 : wsl -l -v

```
PS C:₩Windows₩system32> wsl
wjhjs@LAPTOP-SRMOU6EH:/mnt/c/Windows/system32$ exit
logout
PS C:₩Windows₩system32> wsl -l -v
  NAME                   STATE           VERSION
* Ubuntu-24.04           Running         2
```

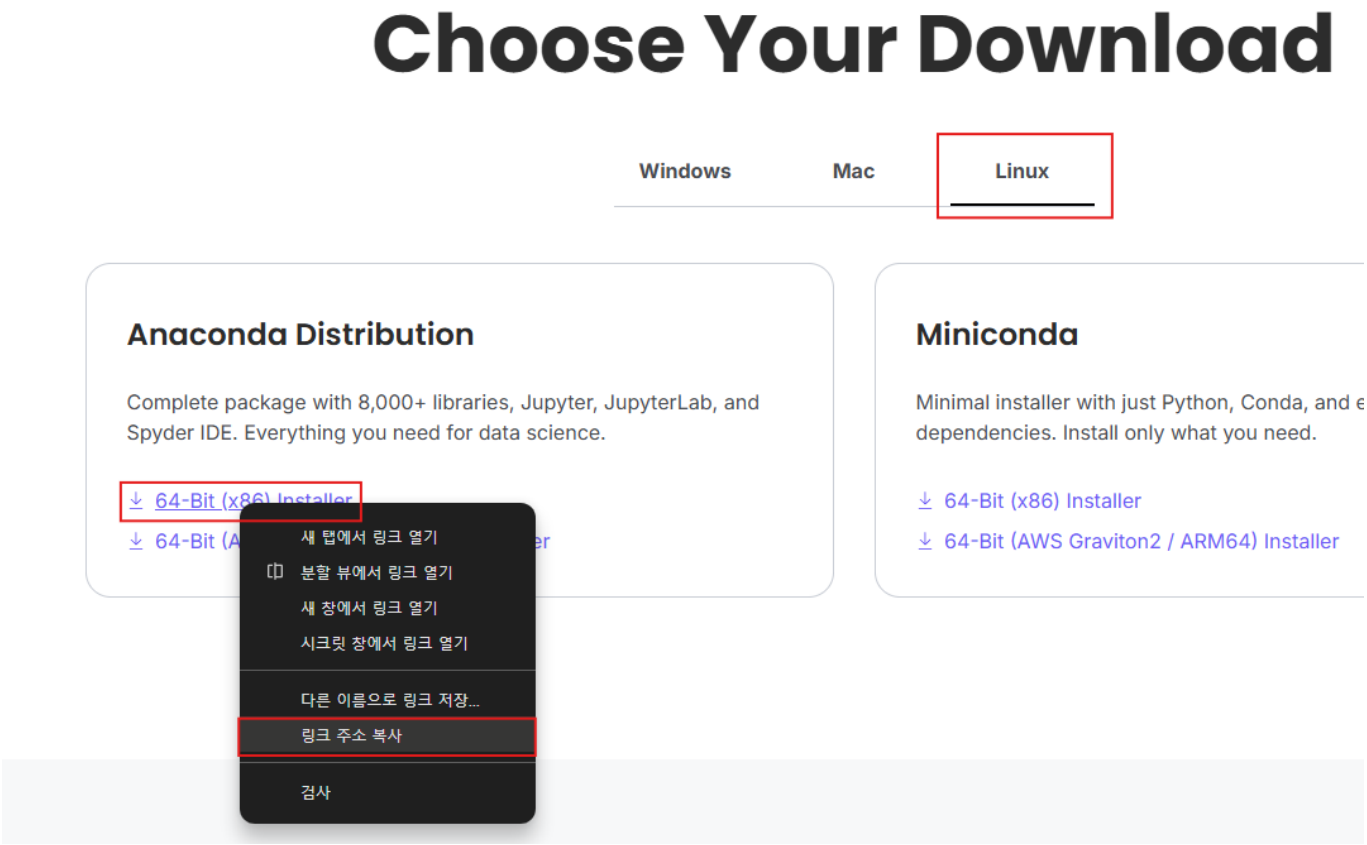
2. Anaconda 설치 (WSL 내부)

2-1. 다운로드 링크 복사

- https://www.anaconda.com/download 접속



- 링크 복사



2-1. 다운로드 및 설치

- Powershell WSL Linux 실행

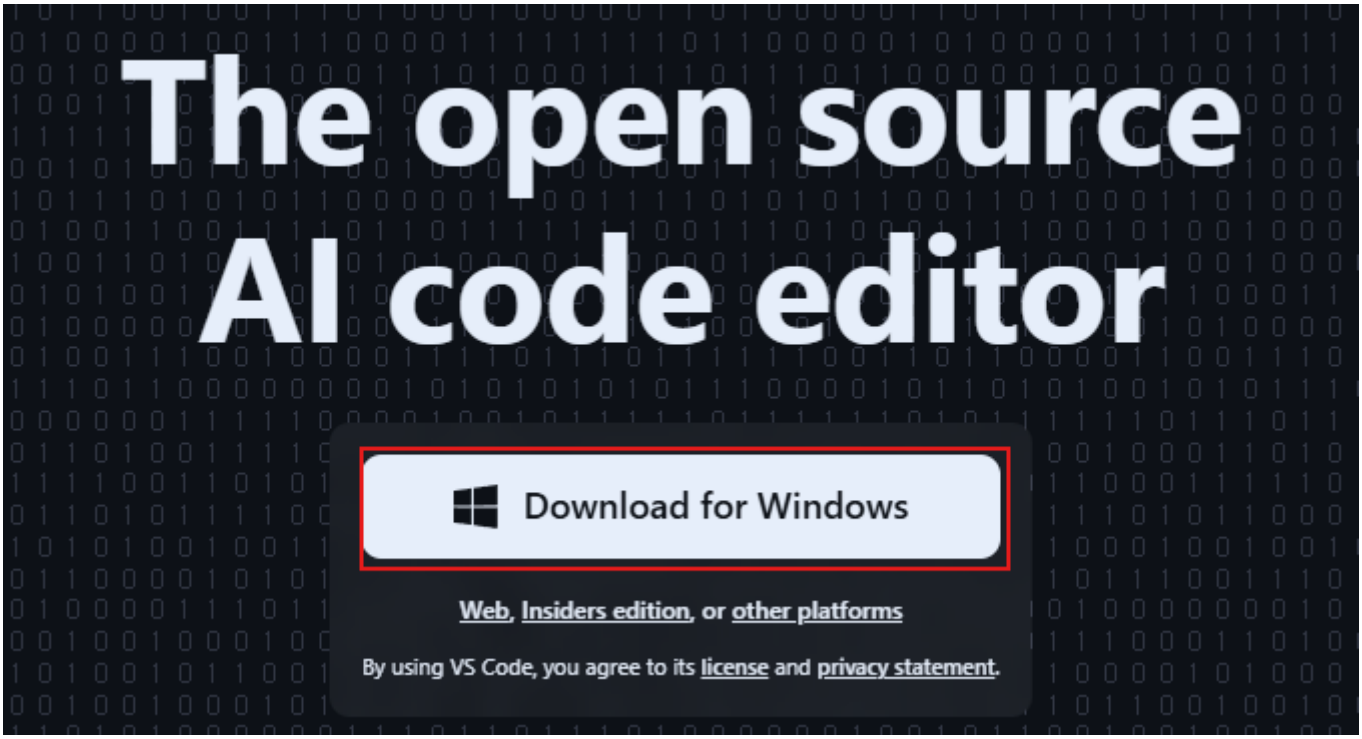
```
# 경로 이동
cd /home/{1-2. 에서 생성한 Unix Account Name}

# 설치 파일을 저장할 폴더 생성
mkdir installers

# 설치 파일 폴더로 이동
cd installers

# Anaconda Linux 설피 파일 다운로드 ( wget { download link } )
wget https://repo.anaconda.com/archive/Anaconda3-2025.12-1-Linux-x86_64.sh

# Anaconda 설치 실행 ( 설치 과정에서 나오는 질문은 yes 입력 )
bash Anaconda3-2025.12-1-Linux-x86_64.sh
```

- PowerShell 에서 WSL 재실행 (base 표시 확인)

3. Visual Studio Code 설치

3-1. 다운로드 및 설치

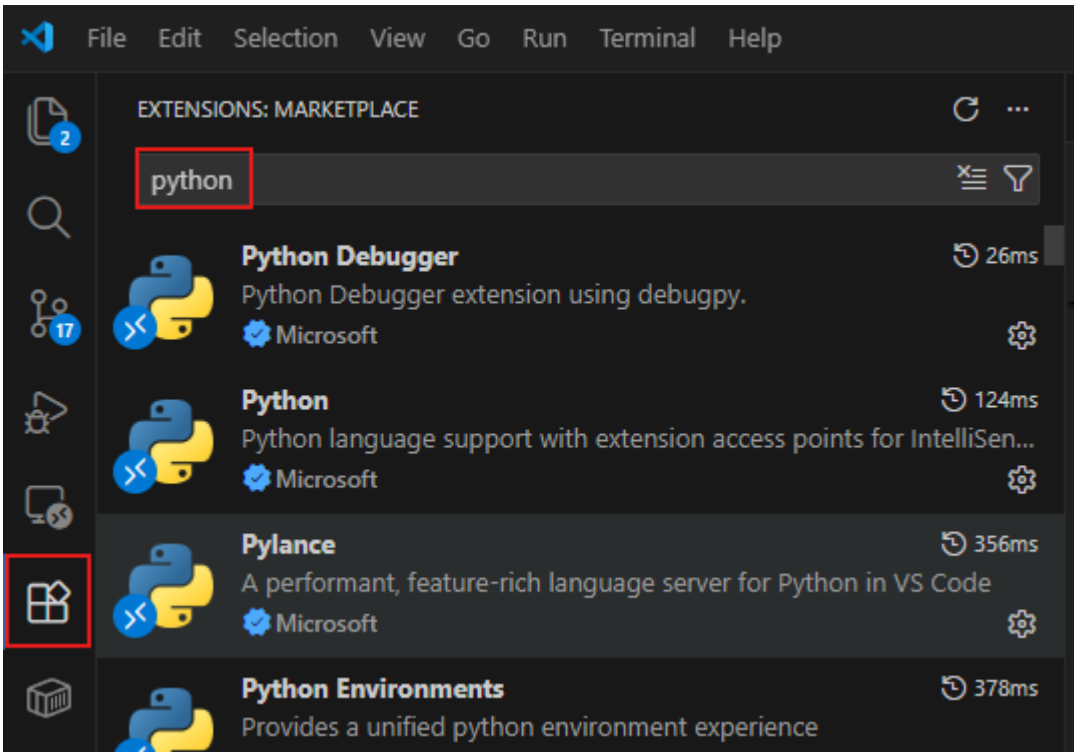
- https://code.visualstudio.com/ 접속, 설치파일 다운로드



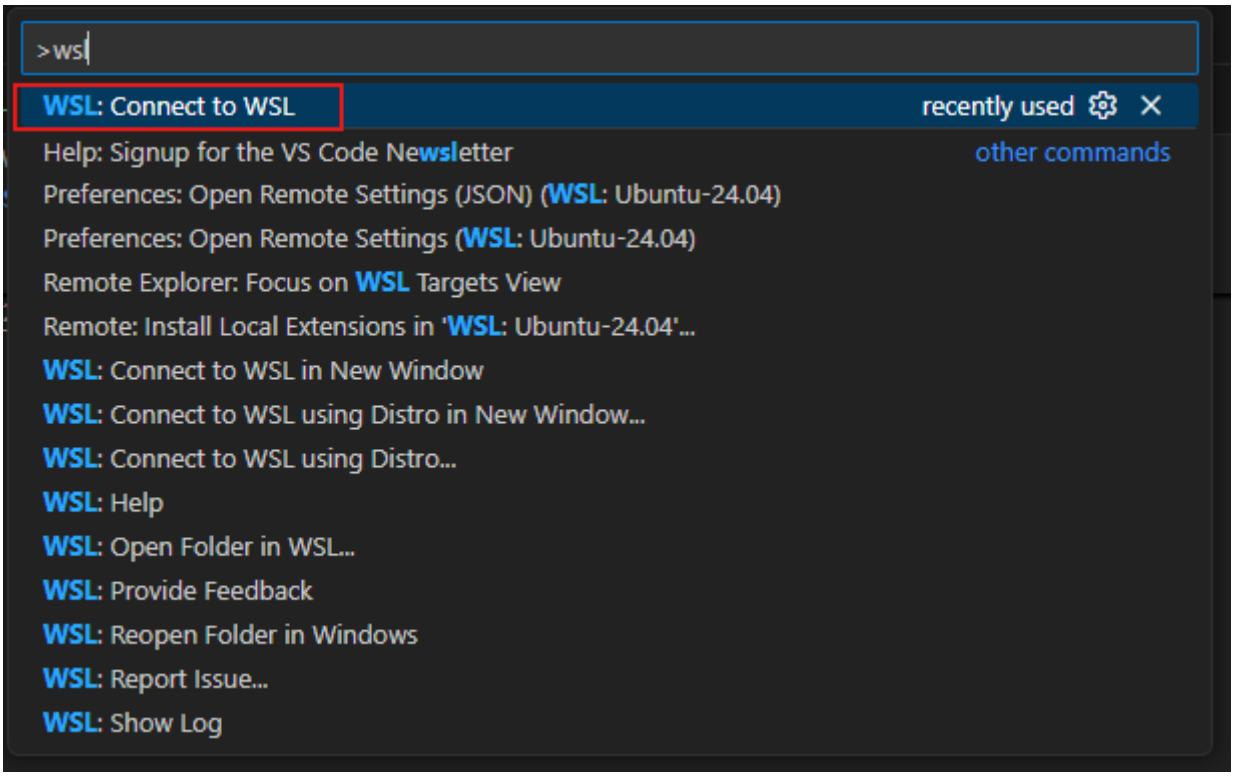
- 설치파일 실행

3-2. Extensions 설치 및 WSL 접속

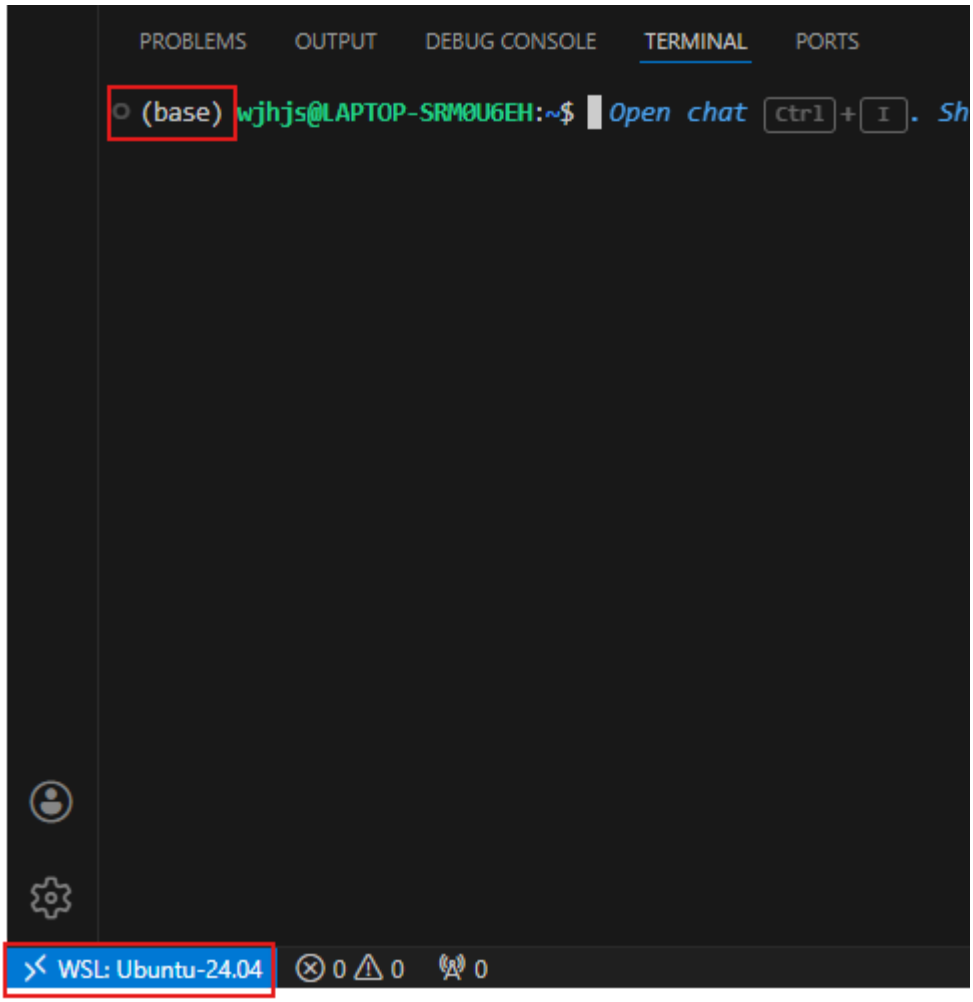
- Extensions 탭에서 python, jupyter, wsl 검색 후 설치



- ctrl + shift + p -> Command Palette (명령 팔레트) -> WSL:Connect to WSL



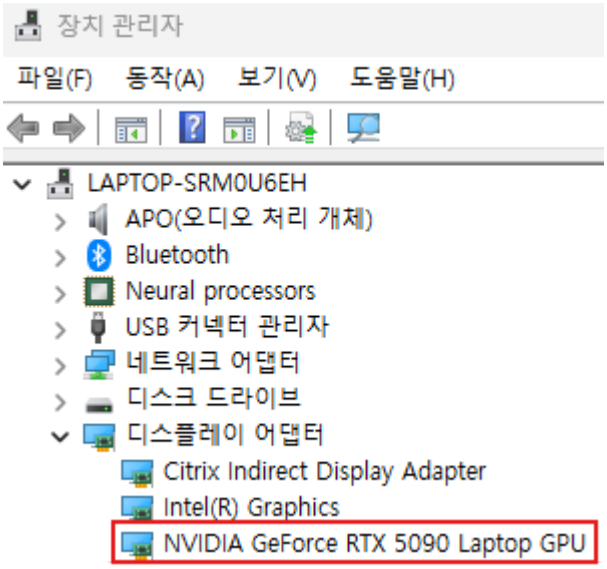
- ctrl + shift + ` -> terminal 실행



4. GPU 세팅

4-1. GPU 확인

- 장치관리자 > 디스플레이 어댑터 > GPU 확인 (GeForce RTX 5090)



- https://ko.wikipedia.org/wiki/CUDA 접속 후 컴퓨팅 기능과 CUDA SDK 버전 확인

1. 컴퓨팅 기능 (12.0)

목차 숨기기
처음 위치
배경
온톨로지
프로그래밍 기능
장점
한계
예시
지원 GPU
버전 기능 및 사양
데이터 유형
부동소수점 유형
버전 지원

컴퓨팅 기능 (버전)	마이크로-아키텍처	GPU	지포스	쿼드로, NVS	테슬라/데이터센터	테그라, 젯슨, DRIVE
	블랙웰					AGX 토르
10.3		GB200, G10			B300, GB10	
12.0		GB202, GB203, GB205, GB206, GB207	GeForce RTX 5090, RTX 5080, RTX 5070 Ti, RTX 5070, RTX 5060 Ti, RTX 5060, RTX 5050	RTX PRO 6000 Blackwell, RTX PRO 5000 Blackwell, RTX PRO 4500 Blackwell, RTX PRO 4000 Blackwell	B40	
12.1						
컴퓨팅 기능 (버전)	마이크로-아키텍처	GPU	지포스	쿼드로, NVS	테슬라/데이터센터	테그라, 젯슨, DRIVE

2. CUDA SDK 버전 (12.8 ~)

목차 숨기기
처음 위치
배경
온톨로지
프로그래밍 기능
장점
한계
예시
지원 GPU
버전 기능 및 사양
데이터 유형
부동소수점 유형
버전 지원

CUDA SDK 버전	테슬라	페르미	케플러 (초기)	케플러 (후기)	맥스웰	파스칼	볼타	튜링	암페어	에이다 러브레이스	호퍼	블랙웰
6.5	1.1			3.7	5.x							
7.0 – 7.5		2.0			5.x							
8.0		2.0				6.x						
9.0 – 9.2			3.0				7.0 – 7.2					
10.0 – 10.2			3.0					7.5				
11.0[47]				3.5					8.0			
11.1 – 11.4[48]				3.5					8.6			
11.5 – 11.7.1[49]				3.5					8.7			
11.8[50]				3.5						8.9	9.0	
12.0 – 12.6					5.0						9.0	
12.8					5.0							12.0
12.9					5.0							12.1

4-2. CUDA 설치

- https://developer.nvidia.com/cuda-toolkit-archive 접속

CUDA Toolkit Archive

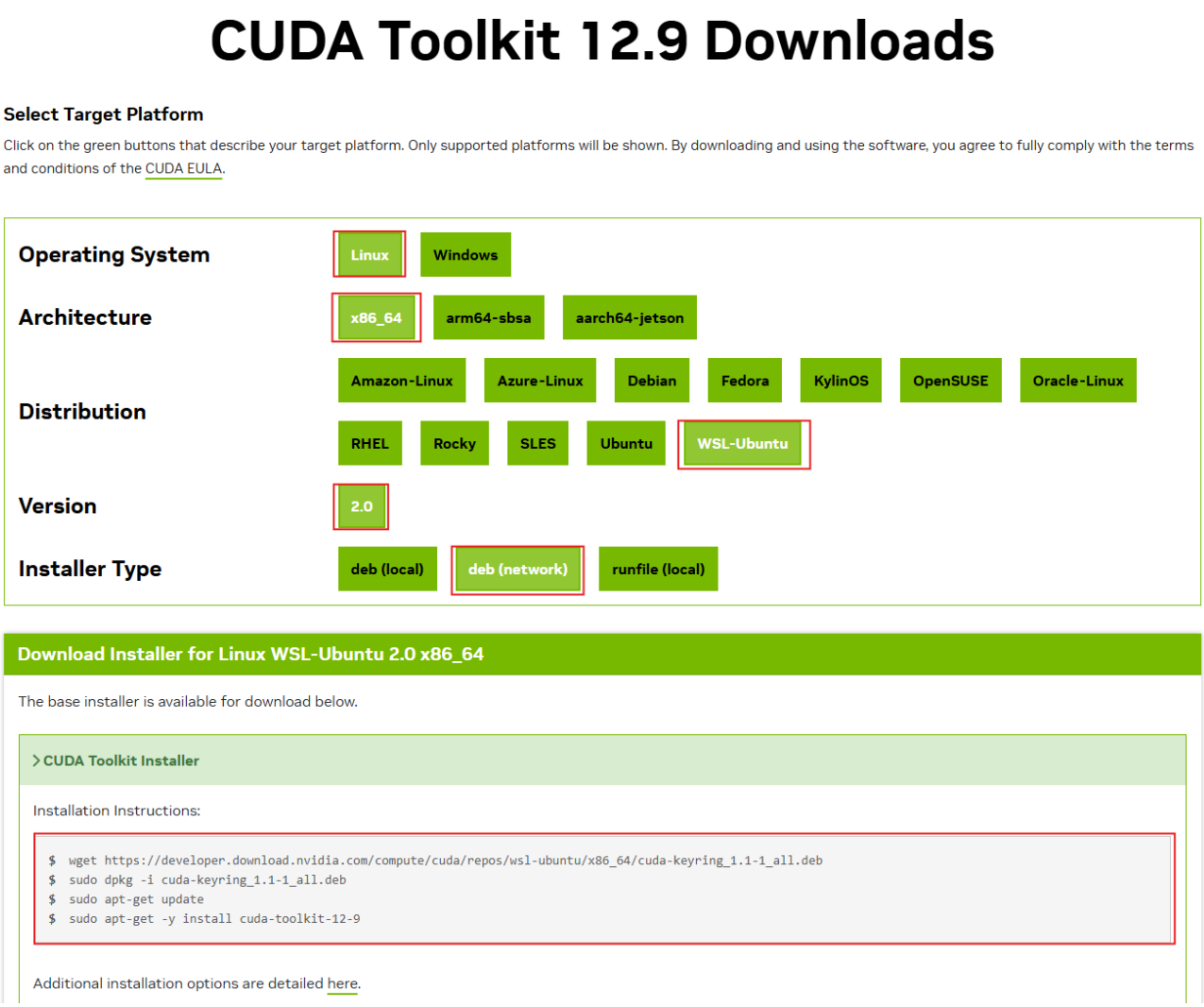
Previous releases of the CUDA Toolkit, GPU Computing SDK, documentati

Download Latest CUDA Toolkit
Learn More about CUDA Toolkit

Latest Release
CUDA Toolkit 13.1.1 (January 2026), Versioned Online Documentation

Archived Releases

CUDA Toolkit 13.1.0 (December 2025), Versioned Online Documentation
CUDA Toolkit 13.0.2 (October 2025), Versioned Online Documentation
CUDA Toolkit 13.0.1 (September 2025), Versioned Online Documentation
CUDA Toolkit 13.0.0 (August 2025), Versioned Online Documentation
CUDA Toolkit 12.9.1 (June 2025), Versioned Online Documentation
CUDA Toolkit 12.9.0 (May 2025), Versioned Online Documentation
CUDA Toolkit 12.8.1 (March 2025), Versioned Online Documentation
CUDA Toolkit 12.8.0 (January 2025), Versioned Online Documentation



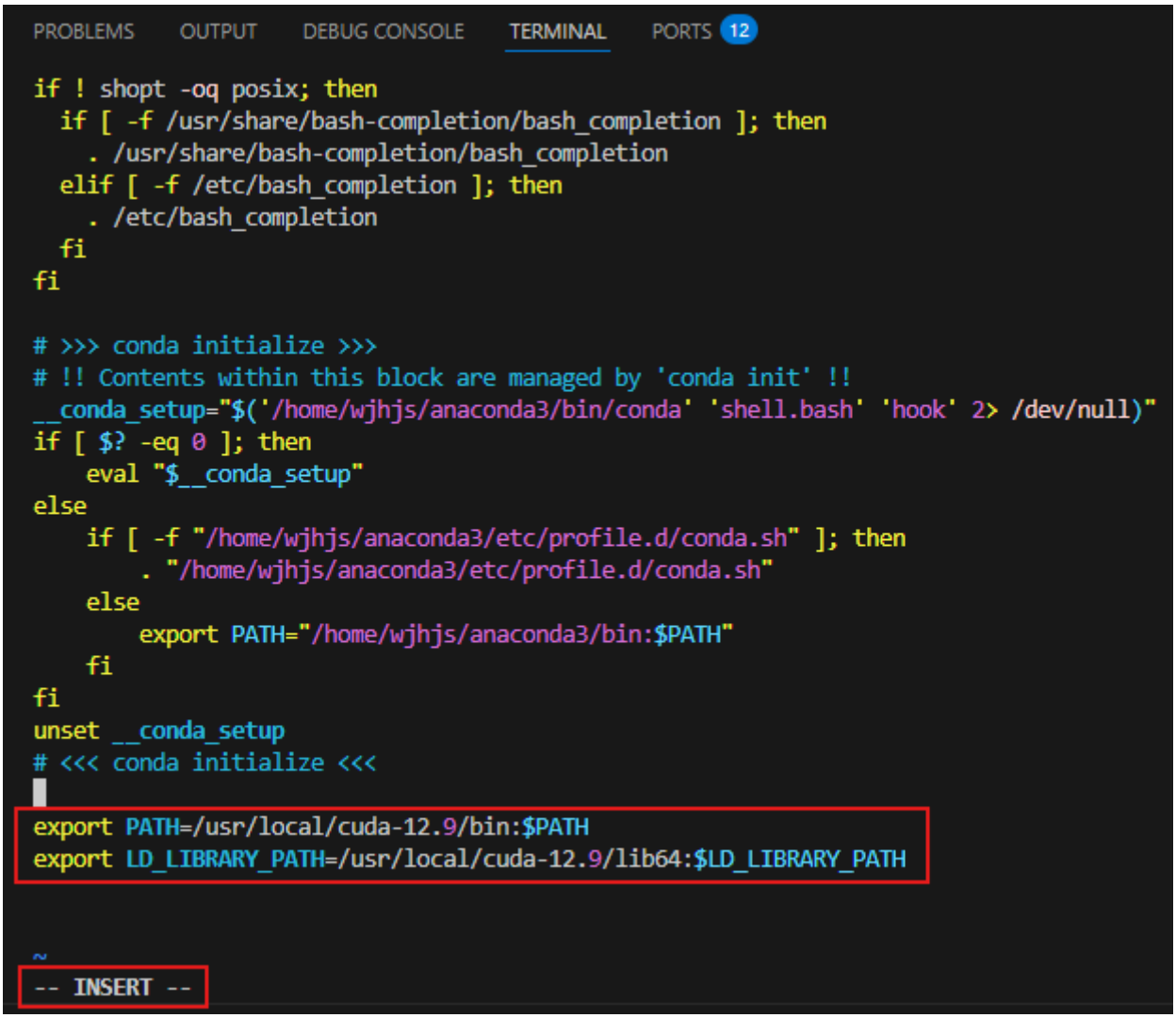
- VS Code의 WSL접속 모드 터미널 or PowerShell WSL Linux 에서 아래 명령어 실행

```
# 경로이동
cd /home/{1-2. 에서 생성한 Unix Account Name}

# 설치 파일 폴더로 이동
cd installers

# CUDA 홈페이지에서 복사한 코드 실행
wget
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/x86_64
/cuda-keyring_1.1-1_all.deb
sudo dpkg -i cuda-keyring_1.1-1_all.deb
sudo apt-get update
sudo apt-get -y install cuda-toolkit-12-9

# bashrc 파일 편집
vim ~/.bashrc
```

- 환경변수 설정

1. 터미널에서 'a' (편집 모드) 키 입력 -> 환경변수 추가

```
export PATH=/usr/local/cuda-12.9/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-12.9/lib64:$LD_LIBRARY_PATH
```



2. Esc (편집 모드 종료) -> : (종료 옵션) 키 입력 -> wq (저장 후 종료) 입력 -> Enter

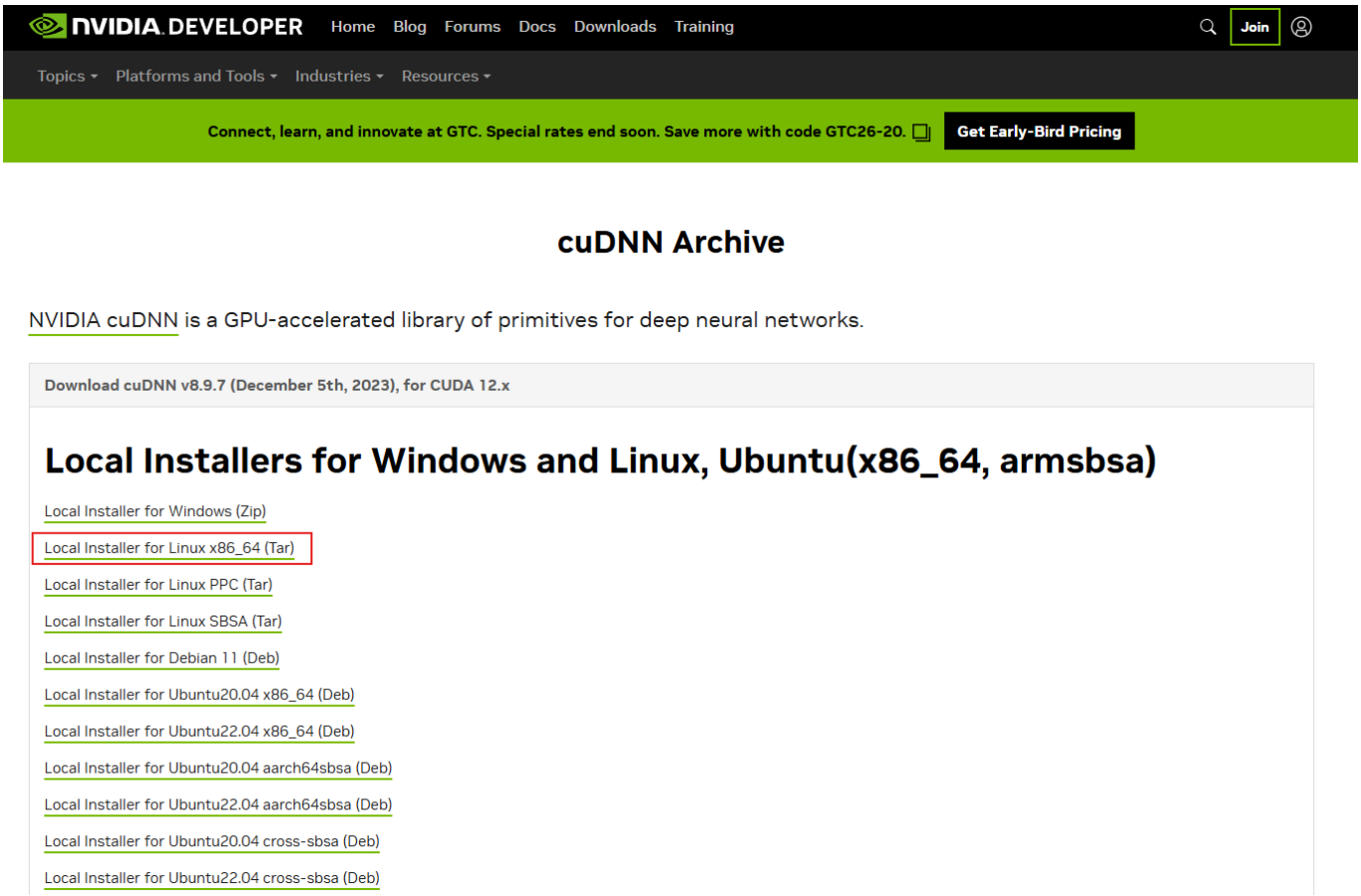
```
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/home/wjhjs/anaconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/wjhjs/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/home/wjhjs/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/wjhjs/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<

export PATH=/usr/local/cuda-12.9/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-12.9/lib64:$LD_LIBRARY_PATH


~
:wq
```

```
# 환경변수 편집 내용 적용
source ~/.bashrc

# CUDA 설치 확인
nvcc --version
```

```
(base) wjhjs@LAPTOP-SRM0U6EH:~/installers$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2025 NVIDIA Corporation
Built on Tue May 27 02:21:03 PDT 2025
Cuda compilation tools, release 12.9, V12.9.86
Build cuda_12.9.r12.9/compiler.36037853_0
```

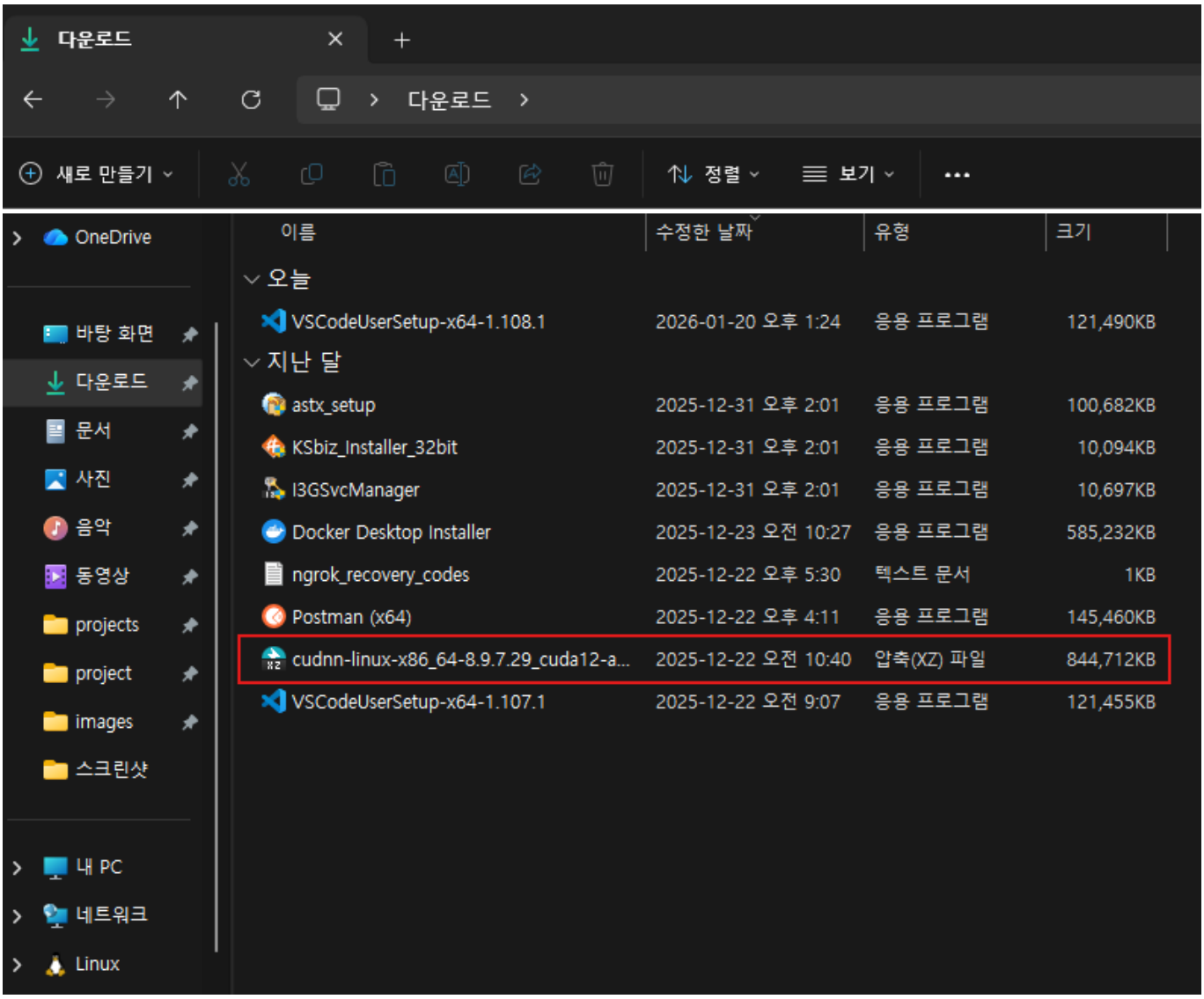
4-3. cuDNN 설치

- https://developer.nvidia.com/rdp/cudnn-archive 접속, CUDA 버전(12.9)과 맞는 최신 cuDNN 다운로드 (Linux용 tar 파일) - **회원가입** 필요

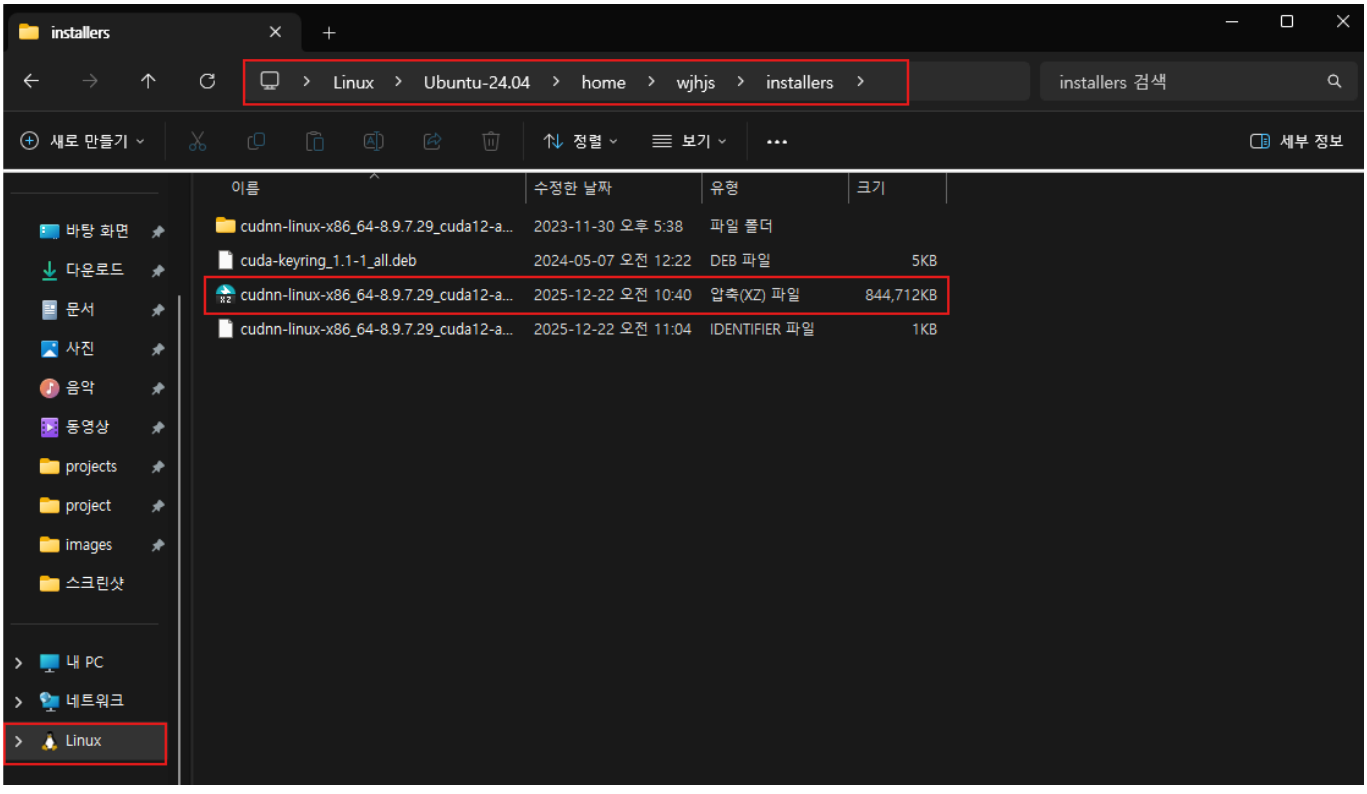


- 파일 탐색기에서 windows로 다운받은 설치파일을 linux (/home/wjhjs/installers/) 로 이동

1. windows download



2. linux (/home/wjhjs/installers/)



- 압축 풀고 설치

```
# 설치 파일이 있는 경로로 이동
cd /home/wjhjs/installers/

# cuDNN 설치 파일이 있는 디렉터리로 이동
cd /home/wjhjs/installers/

# 다운로드한 cuDNN 압축 파일(.tar.xz) 해제
tar -xvf cudnn-linux-x86_64-8.9.7.29_cuda12-archive.tar.xz

# cuDNN 헤더 파일(cudnn*.h)을 CUDA include 디렉터리로 복사
sudo cp cudnn-linux-x86_64-8.9.7.29_cuda12-archive/include/cudnn*.h \
  /usr/local/cuda-12.9/include

# cuDNN 라이브러리 파일(libcudnn*)을 CUDA lib64 디렉터리로 복사
sudo cp -P cudnn-linux-x86_64-8.9.7.29_cuda12-archive/lib/libcudnn* \
  /usr/local/cuda-12.9/lib64

# 모든 사용자에게 cuDNN 라이브러리 읽기 권한 부여
sudo chmod a+r /usr/local/cuda-12.9/lib64/libcudnn*

# cuDNN 설치 확인
cat /usr/local/cuda/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
```

- cuDNN 8.9.7 확인

```
(base) wjhjs@LAPTOP-SRM0U6EH:~/installers$ cat /usr/local/cuda/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 8
#define CUDNN_MINOR 9
#define CUDNN_PATCHLEVEL 7
--
#define CUDNN_VERSION (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)

/* cannot use constexpr here since this is a C-only file */
```

5. vLLM 세팅

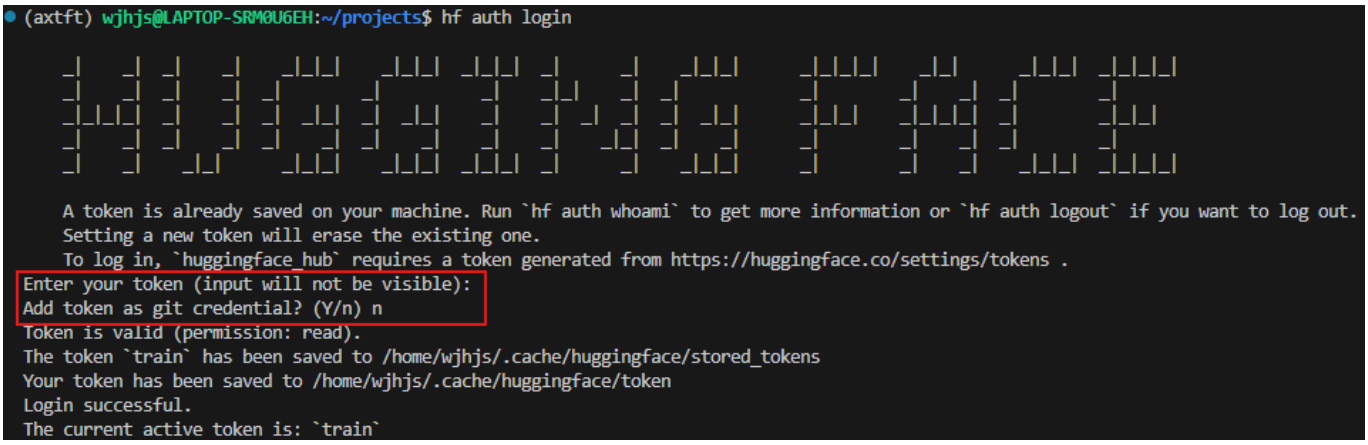
5-1. conda 가상환경

```
# Python 3.12 가상환경 생성 ( conda create -n { 가상환경 이름 } python=3.12 )
conda create -n axtft python=3.12

# 가상환경 실행 ( conda activate { 가상환경 이름 } )
conda activate axtft

# 필요 패키지 설치
pip install vllm==0.13.0
pip install timm
pip install fastapi
```

5-2. 모델 다운로드

```
# 가상환경 실행
conda activate axtft

# hugging face 로그인
hf auth login
```

- hugging face token 입력 (hugging face 회원가입 후 토큰 생성 가능), Add token as git credential? (Y/n) -> n 입력



- 모델 다운로드 (hf download { hugging face 모델명 } --local-dir { 저장할 경로 })

```
# gpt-oss-20b 모델 다운로드
hf download openai/gpt-oss-20b --local-dir ./models/gpt-oss-20b

# gemma-3n-E4B-it 모델 다운로드
hf download google/gemma-3n-E4B-it --local-dir ./models/gemma-3n-E4B-it
```

5-3. vLLM 서빙

```
# 가상환경 실행
conda activate axtft

# vllm 서빙 시작
vllm serve ./models/gpt-oss-20b --served-model-name gpt-oss-20b --max-
model-len 8192 --gpu-memory-utilization 0.8 --port 8000 --host 0.0.0.0
# serve 명령어는 vLLM을 OpenAI 호환 API 서버 형태로 실행
# ./models/gpt-oss-20b : 로컬에 다운로드된 모델 디렉터리 경로
# --served-model-name gpt-oss-20b : API에서 호출할 때 사용하는 모델 이름
# --max-model-len 8192 : 모델이 처리할 수 있는 최대 토큰 길이 ( 가ㅅ이 클수록 KV
Cache 사용량 증가 → GPU 메모리 소모 증가 -> 호출 응답시가ㄴ 길어짐 ), 8192 토큰도 충분
# --gpu-memory-utilization 0.8 : GPU 전체 메모리 중 vLLM이 사용할 최대 비율 ( 기본
적으로 노트북이 GPU를 어느정도 사용하기 때문에 80퍼센트로 할당 )
```

```
# --port 8000 : vLLM API 서버가 바인딩할 포트 번호
# --host 0.0.0.0 : 모든 네트워크 인터페이스에서 접근 허용
```