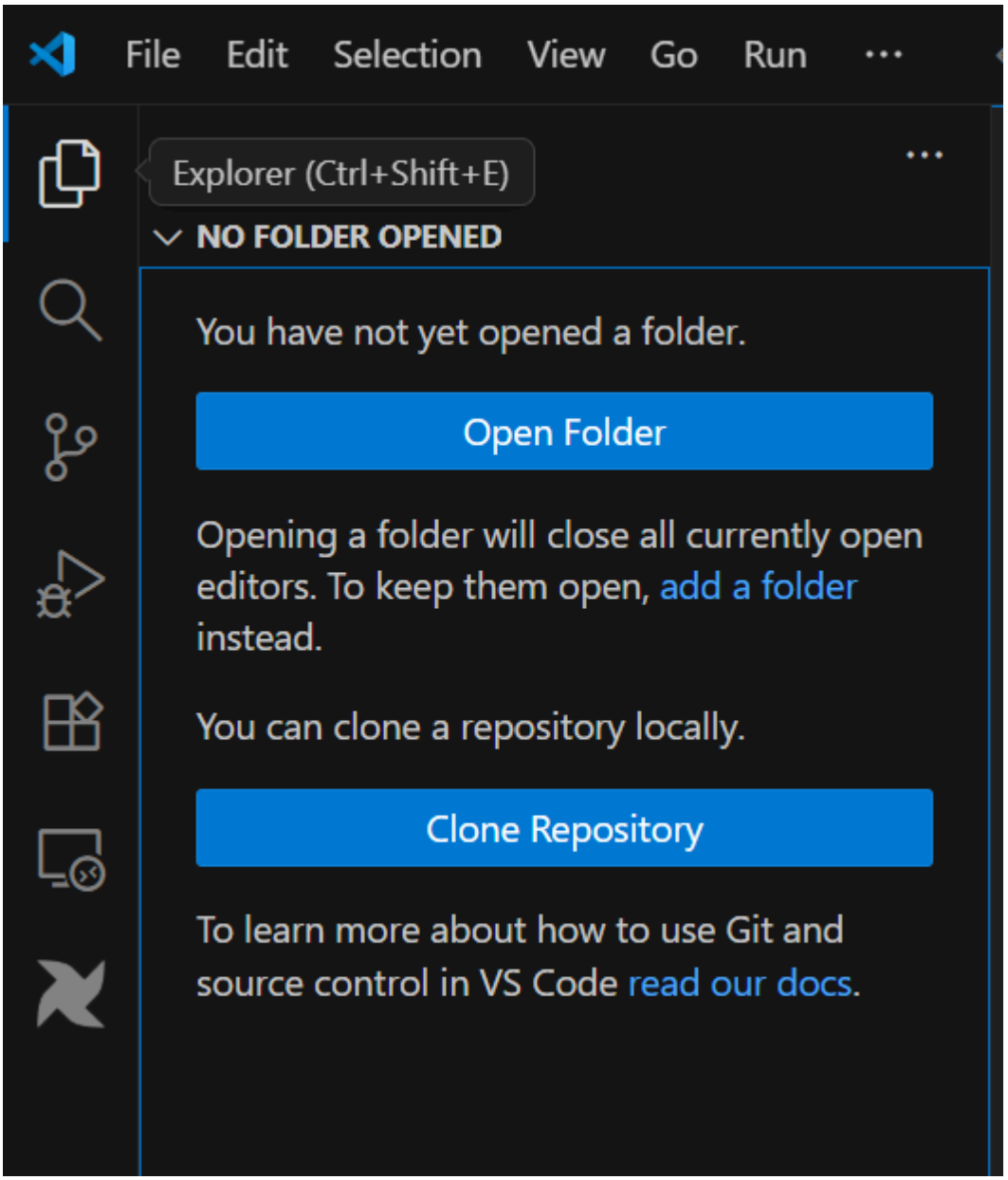
클라우드 튜토리얼

로컬에서 VScode 설치하기

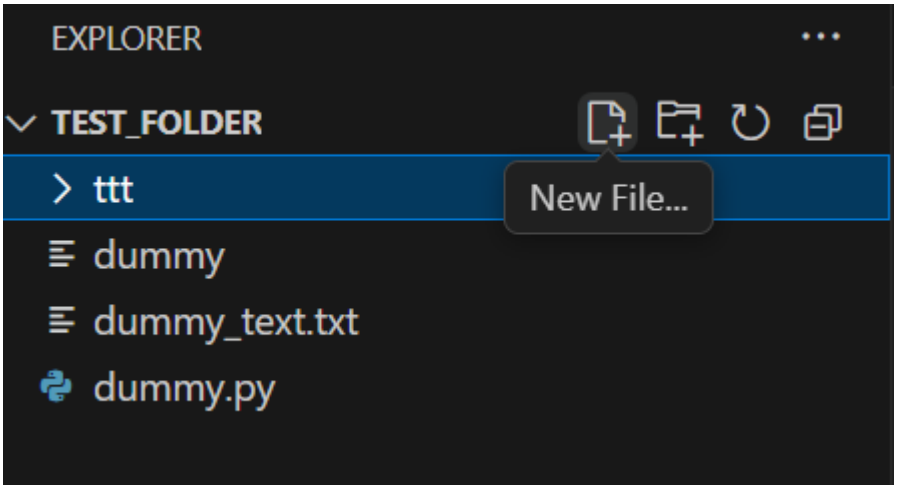
링크에 접속하여 본인 운영체제에 맞는 VScode를 다운로드합니다.

다운로드 받은 VScode를 설치 후 실행합니다.(설치 중 컴퓨터 재시작이 필요할 수 있습니다)



좌측 상단의 폴더 모양 Explorer를 선택하거나 Ctrl+Shift+E를 누릅니다.

원하는 폴더를 선택하면 다음과 같이 파일 탐색기가 등장합니다.



만일 파일 탐색기의 루트를 바꾸고 싶으면 상단 메뉴 File-Open Folder를 선택하여 루트 폴더를 선택합니다.

Explorer의 상단 메뉴는 다음과 같습니다.

- New File...
  현재 선택된 폴더의 하위 항목에 파일을 생성합니다(미선택/전체 선택시 루트 폴더)
- New Folder...

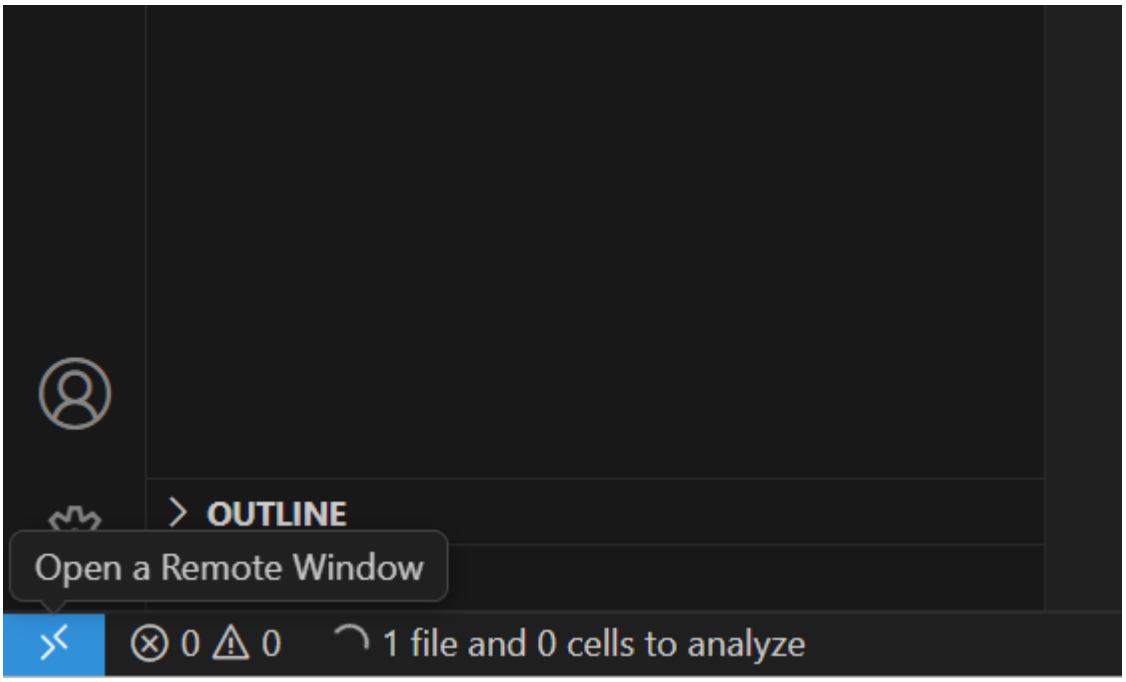
  현재 선택된 폴더의 하위 항목에 폴더를 생성합니다(미선택/전체 선택시 루트 폴더)
- Refresh Folder

  파일 탐색기의 변경 사항을 갱신합니다
- Collapse Folders in Explorer

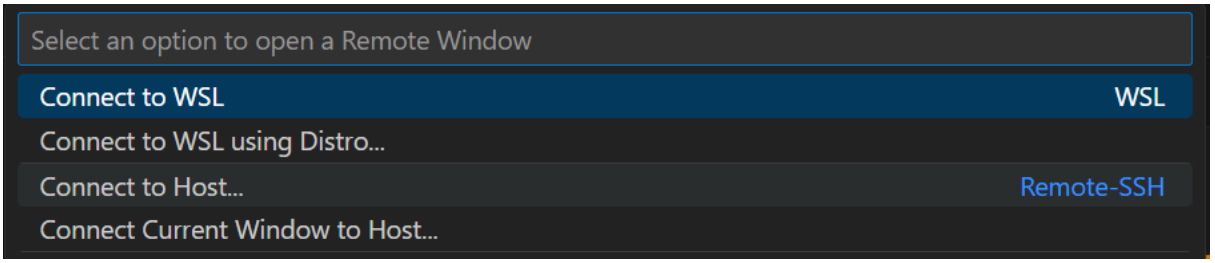
  파일 탐색기 내의 펼쳐져 있는 모든 폴더를 접습니다.

VScode로 SSH 접속하기

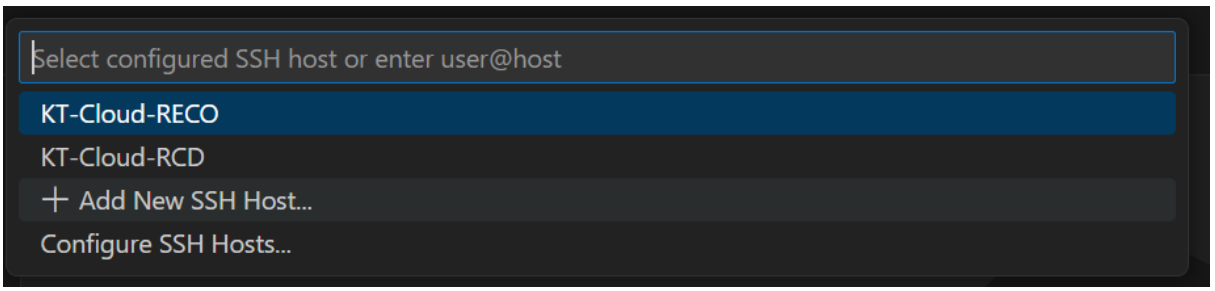
- VScode 실행
- 좌측 Extension에서 Remote - SSH 검색
- Remote -SSH extension install



- 우측 하단 Remote Explorer 클릭(>< 아이콘)

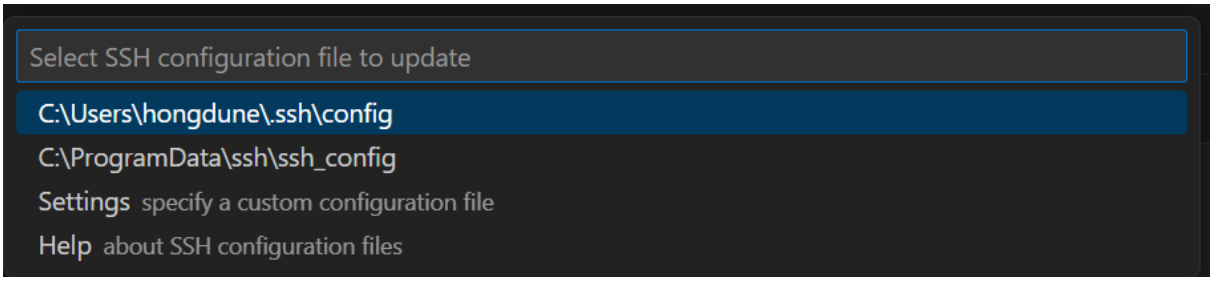


- Connect to Host... 선택

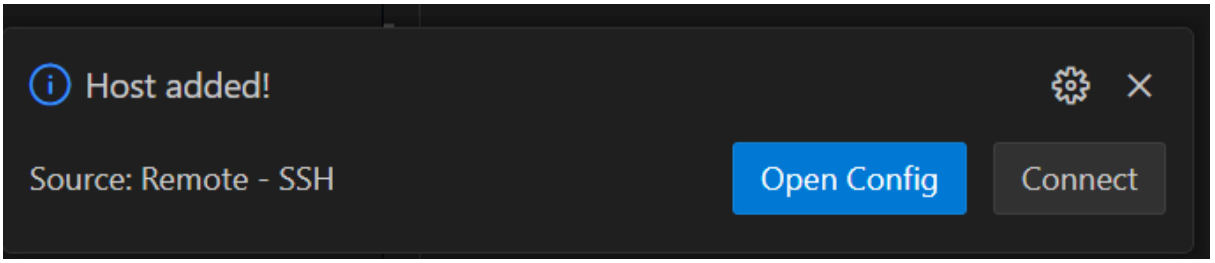


- + Add New SSH Host... 선택 후 아래와 같이 입력

ssh -i "key 위치" <유저명>@<instance public 주소>
예) ssh -i "~/Downloads/test_key.pem" ec2-user@ec2-18-116-241-215.us-east-2.compute.amazonaws.com

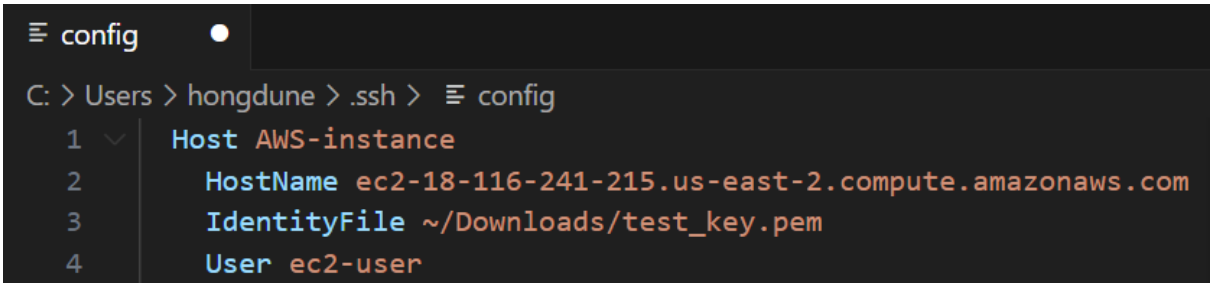


설정이 저장될 위치를 선택합니다.

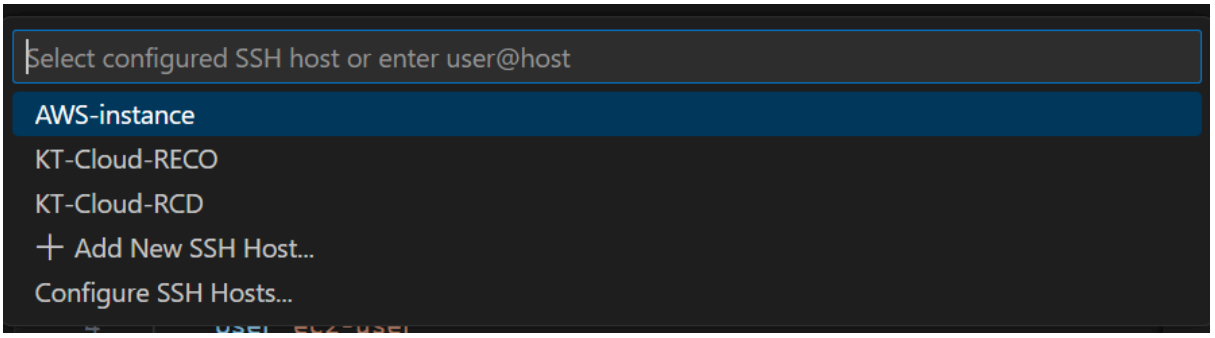


설정이 저장되면 우측 하단에 팝업창이 등장합니다.

(선택사항)Open Config를 클릭하여 Host 이름을 아래와 같이 수정합니다.

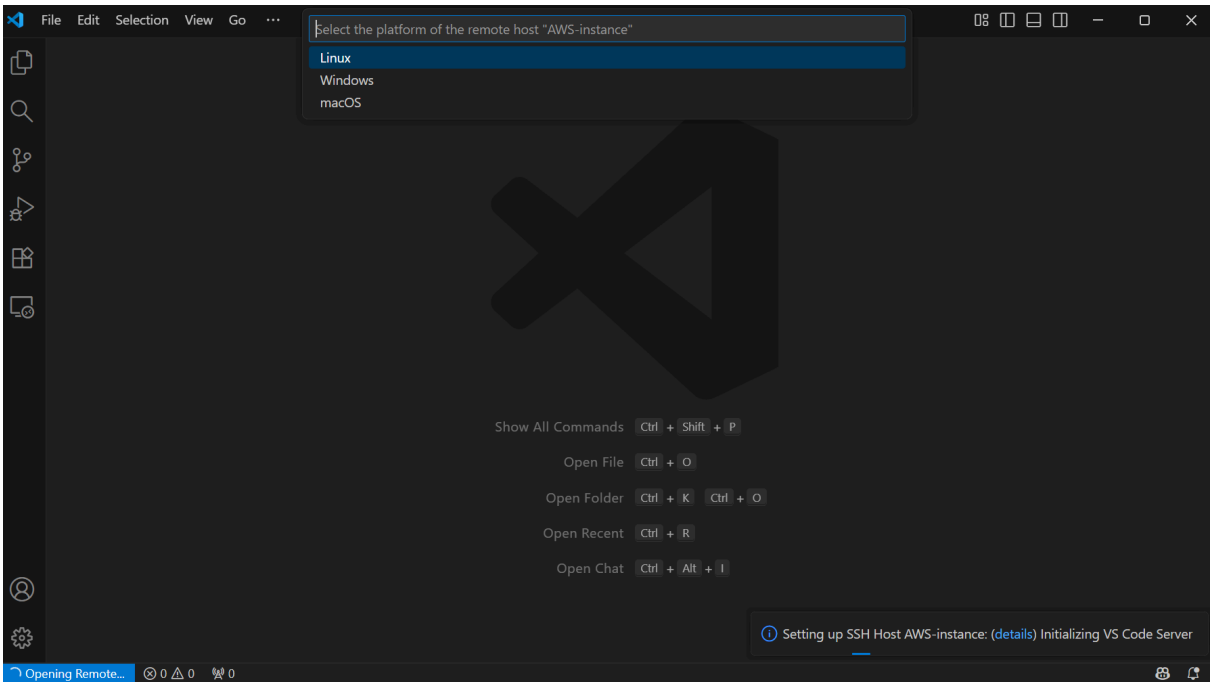


이후 Remote Explorer 클릭 - Connect to Host... 선택 을 반복하면 아래와 같이 인스턴스가 등록됩니다.



등록한 인스턴스를 클릭합니다.

아래와 같이 새 창이 열리면서 운영체제를 선택할 수 있게 등장하면 성공입니다.



여기에서는 Linux를 선택합니다.

인스턴스 살펴보기

```
df -h
```

```
sudo chown azureuser:azureuser /mnt
```

그래픽 드라이버 설치하기

주의 사항 : 실습 예시에서는 NVIDIA TESLA T4 GPU를 상정하고 작성되었습니다. 실제 제공 받을 환경과 다를 수 있습니다.

`ctrl+shift+`` 혹은 상단 메뉴에서 Terminal - New Terminal을 선택합니다.

다음 코드를 입력합니다.

```
nvidia-smi
```

아래와 같은 에러 메시지 혹은 유사한 메시지가 등장하면 그래픽 드라이버가 설치되지 않았다는 의미입니다.

```
azureuser@azureuser:~$ nvidia-smi
Command 'nvidia-smi' not found, but can be installed with:
sudo apt install nvidia-utils-470         # version 470.256.02-0ubuntu0.24.04.1, or
sudo apt install nvidia-utils-470-server  # version 470.256.02-0ubuntu0.24.04.1
sudo apt install nvidia-utils-535-server  # version 535.261.03-0ubuntu0.24.04.1
sudo apt install nvidia-utils-570         # version 570.172.08-0ubuntu0.24.04.1
sudo apt install nvidia-utils-570-server  # version 570.172.08-0ubuntu0.24.04.1
sudo apt install nvidia-utils-580         # version 580.65.06-0ubuntu0.24.04.4
sudo apt install nvidia-utils-580-server  # version 580.65.06-0ubuntu0.24.04.1
sudo apt install nvidia-utils-535         # version 535.247.01-0ubuntu0.24.04.1
sudo apt install nvidia-utils-550         # version 550.163.01-0ubuntu0.24.04.1
sudo apt install nvidia-utils-565-server  # version 565.57.01-0ubuntu0.24.04.3
sudo apt install nvidia-utils-525         # version 525.147.05-0ubuntu1
sudo apt install nvidia-utils-525-server  # version 525.147.05-0ubuntu1
sudo apt install nvidia-utils-550-server  # version 550.163.01-0ubuntu0.24.04.1
```

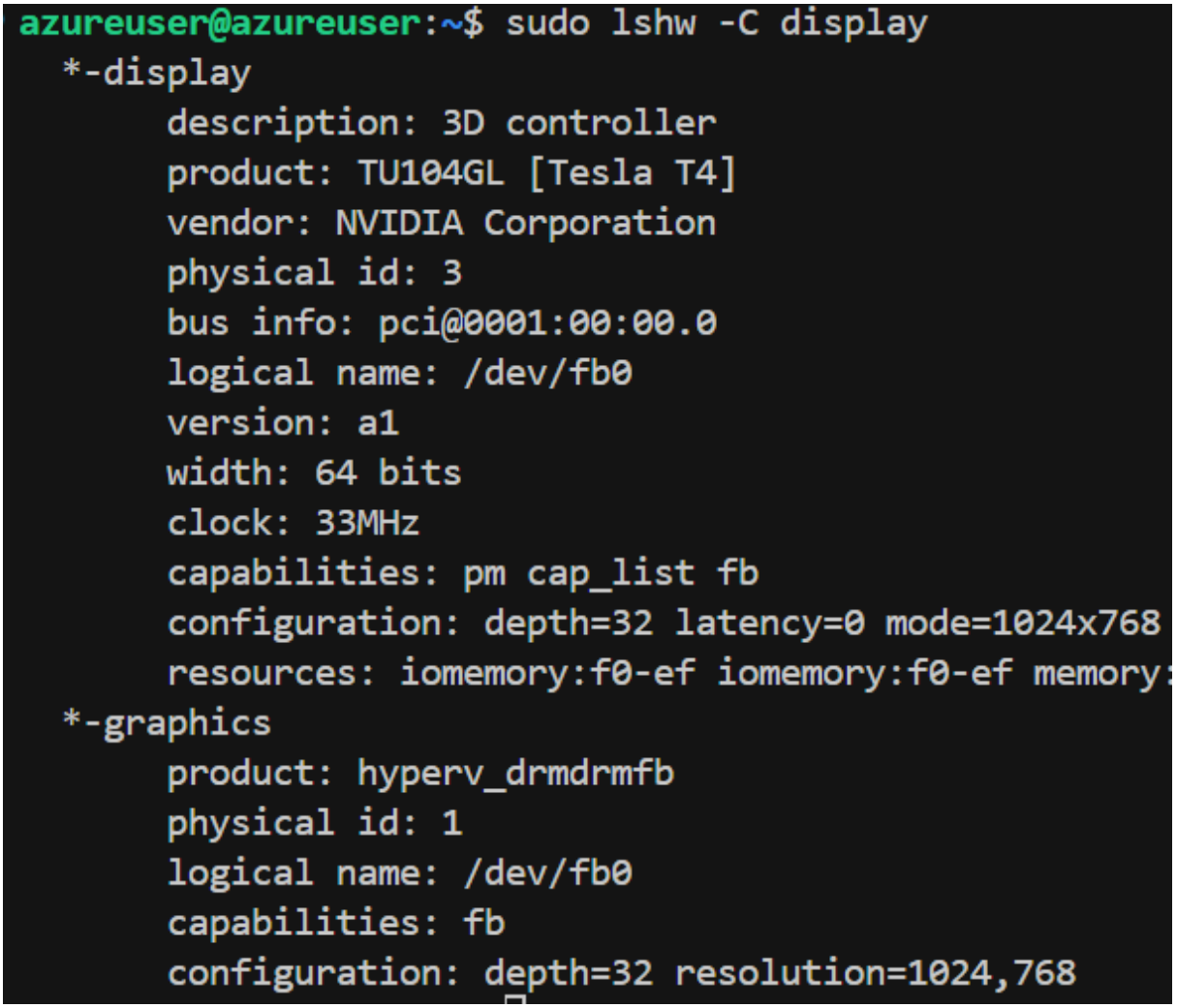
현재 인스턴스에 GPU 장치가 있는지 확인하기 위해선 다음 명령어를 실행합니다.

```
# lspci 설치 필요

lspci | grep -i nvidia
```

```
azureuser@azureuser:~$ lspci | grep -i nvidia
0001:00:00.0 3D controller: NVIDIA Corporation TU104GL [Tesla T4] (rev a1)
```

혹은 다음과 같이 입력합니다.

```
sudo lshw -C display
```



만일 위와 상이한 화면이 나오는 경우 강사, 보조강사 혹은 매니저에게 상황을 알려주시기 바랍니다.

정상적으로 GPU를 사용하기 위해선 그래픽 드라이버를 설치하여야합니다. 그래픽 드라이버는 하드웨어와 운영체제 양쪽의 호환성을 맞추어야합니다.

간단하게는 아래 링크를 통해 적절한 드라이버를 찾을 수 있습니다.

https://www.nvidia.com/en-us/drivers/

Tesla T4 GPU, Ubuntu 24.04를 기준으로 적절한 그래픽 드라이버 버전을 찾아봅시다.

▼ Driver Version & CUDA Toolkit

> Driver Version: 580.105.08
> CUDA Toolkit: 13.0

위 내용은 현재 T4는 그래픽 드라이버 버전은 어디까지 쓸 수 있고 이 경우 CUDA Toolkit은 13.0까지 쓸 수 있음을 알려줍니다. 만일 그래픽 드라이버 버전을 다운그레이

드하려면 CUDA Toolkit 버전도 호환성을 맞춰 함께 다운그레이드해야합니다.

다음 수순을 따라 그래픽 카드 드라이버를 설치합니다. 아래 설치 순서는 NVIDIA Document를 참고하고 있습니다.[링크]

1. 필수 의존성 설치

```
sudo apt update
sudo apt install -y build-essential dkms linux-headers-$(uname -r)
```

2. NVIDIA 레포지토리 등록

```
distribution=$(. /etc/os-release; echo ${ID}${VERSION_ID} | tr -d .)
curl -fsSL https://developer.download.nvidia.com/compute/cuda/repos/${distribution}/x86_64/cuda-keyring_1.1-1_all.deb -o cuda-keyring.deb
sudo dpkg -i cuda-keyring.deb
sudo apt update
```

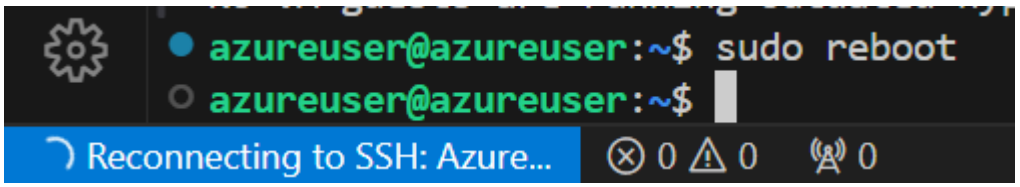
3. 그래픽 드라이버 설치

```
sudo apt install -y nvidia-headless-580 nvidia-utils-580
```

4. 재부팅

```
sudo reboot
```

재부팅 명령 후 실제 부팅까진 상당한 시간이 소요될 수 있습니다. 잠시 후에 다시 시도해주시고 5분이 넘었는데도 문제가 계속된다면 강사나 보조 강사에게 말씀해주시기 바랍니다.

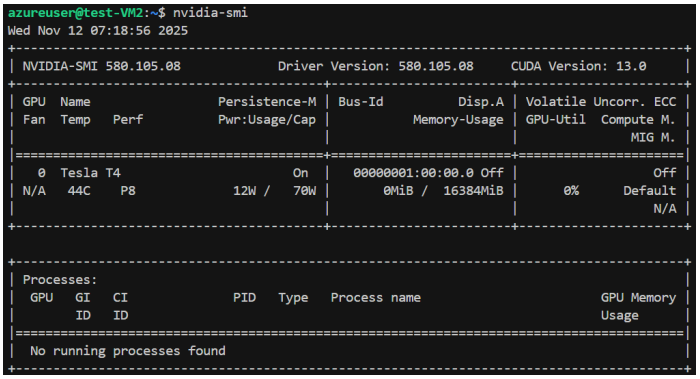


5. 설치 확인

재부팅이 완료되면 다음 명령어를 수행합니다.

```
nvidia-smi
```

아래와 같은 화면이 등장하면 성공입니다.



설치된 Driver Version에 맞는 CUDA와 cuDNN을 설치해주어야합니다. 화면에 나오는 CUDA Version은 추천 CUDA 버전으로 실제와 다를 수 있습니다.

CUDA toolkit과 cuDNN 설치하기

주의사항

PyTorch는 시스템 CUDA 버전과 무관하게, 자체 CUDA 런타임을 포함한 패키지를 배포 및 사용합니다.

때문에 PyTorch를 통해 딥러닝을 개발하거나 이후 과정인 **가상환경 별 CUDA / cuDNN 설정하기**를 진행하는 경우 이 스텝을 생략하고 이후 배포시에 OS 단의 CUDA / cuDNN이 필요할때 진행하여도 무방합니다.

CUDA toolkit 설치하기

CUDA toolkit 버전은 그래픽 드라이버가 지원하는 가장 높은 버전 이하라면 비교적 자유롭게 설정 가능합니다.

일반적으로 cuDNN의 경우 CUDA 버전에 맞춰서 설치합니다.

설치 가능한 CUDA 버전은 다음과 같이 조회할 수 있습니다.

```
apt-cache search '^cuda-toolkit' | sort
```

앞선 **그래픽 드라이버 설치하기** 실습에서는 `580.105.08` 버전을 설치했습니다. 이 경우 CUDA는 13.0까지 설치할 수 있습니다.

여기에서는 Ubuntu 24.04에서 안정적인 것으로 알려진 12.5 버전으로 설치해보겠습니다.

각 버전별 문서는 링크를 통해서 확인할 수 있습니다.

1. CUDA toolkit 설치하기

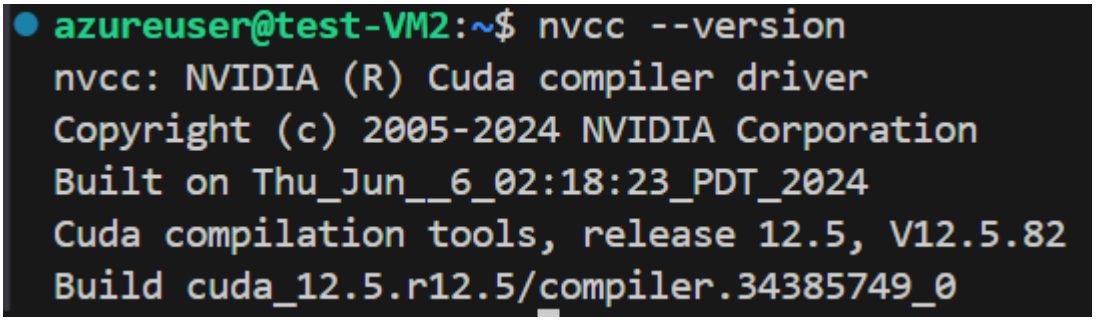
다음 코드를 수행해 설치를 진행합니다.

```
sudo apt update
sudo apt install -y cuda-toolkit-12-5
```

2. 설치 버전 확인하기

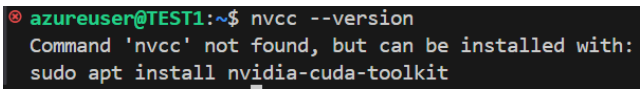
   설치가 완료되면 다음 코드를 실행시켜 설치된 CUDA 버전을 확인합니다.

   ```
   nvcc --version
   ```



▼ 만일 위 화면이 등장하지 않은 경우

예시 이미지 대신 다음과 같은 화면이 등장할 경우 먼저 `sudo reboot` 후 다시 시도합니다.



상황이 반복되면 다음 코드를 실행시켜 설치가 되었는지 확인합니다.

```
sudo find /usr -name nvcc -type f 2>/dev/null
```

`/usr/local/cuda-12.5/bin/nvcc` 와 같은 메시지가 등장하면 정상적으로 설치된 것입니다.

이 경우 다음 명령어를 수행해 PATH를 등록 후 이어서 진행합니다.

```
echo 'export PATH=/usr/local/cuda-12.5/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
```

cuDNN 설치하기

설치한 Toolkit 버전에 맞춰서 cuDNN도 설치해주어야합니다. 설치해야할 cuDNN 패키지 버전은 링크를 통해 확인할 수 있습니다.

해당 내용에 따르면 cuDNN 9.15.1 for CUDA 12.x을 설치하면 됩니다.

설치 과정은 다음과 같습니다.

1. 다운로드 링크를 통해 필요한 버전을 선택합니다.

   다음과 같은 옵션을 선택해 진행하겠습니다.

   - Operating System : Linux
   - Architecture : x86_64
   - Distribution : Ubuntu
   - Version : 24.04
   - Installer Type : deb (network)
   - Configuration : FULL

2. Install 가이드를 확인합니다.

   주의. 기본 값으로 설치하게 되면 CUDA 13 버전용으로 다운로드되게 됩니다. 아래 코드를 참고해 진행해주세요.

   ```
   wget https://developer.download.nvidia.com/compute/cuda/repos/ubu
   ntu2404/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 cudnn9-cuda-12
   ```

3. 설치 버전을 확인합니다.

   ```
   dpkg -l | grep cudnn
   ```

   아래와 같이 CUDA 12.x용 cuDNN이 설치되어야합니다.

▼ 만일 CUDA 13 버전 용으로 설치되었을 경우

아래 수순을 통해 cuDNN 삭제 후 재설치해주세요.

먼저 cuDNN을 삭제합니다.

```
sudo apt remove -y cudnn9-cuda-13 cudnn9-cuda-13-0 libcudnn9-cuda-13 \
  libcudnn9-dev-cuda-13 libcudnn9-headers-cuda-13 libcudnn9-static-cuda-13
```

메타 패키지 역시 삭제합니다.

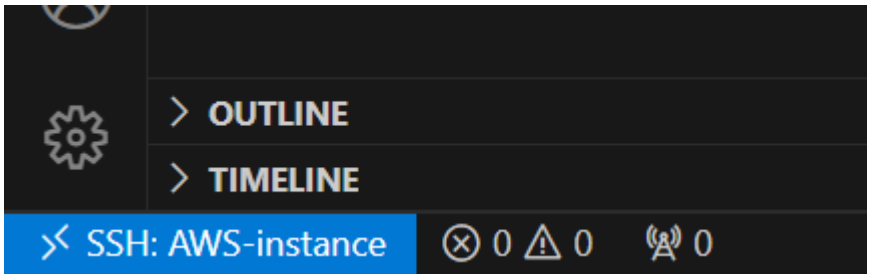
```
sudo apt remove -y cudnn
sudo apt autoremove -y
```

이후 위 설치 과정을 진행합니다.

아나콘다 설치하기

아나콘다 설치하기 부터는 클라우드 인스턴스에 접속되었다는 전제 하에 설명이 진행됩니다.

만일 **VScode로 SSH 접속하기** 과정을 진행하였다면 화면 우측 하단에 아래와 같이 SSH 접속이 되어야합니다.



아나콘다를 검색하거나 링크를 통해 Anconda 설치 페이지로 접속합니다.(혹은 Miniconda로 진행해도 됩니다)

여기에서는 리눅스 환경을 기준으로 설치하겠습니다.

1. 다음 코드를 실행하여 설치 파일을 다운로드 받습니다. 만일 특정 버전의 아나콘다를 다운로드 받을 때는 운영체제, 장치의 아키텍처 타입(x86, ARM64 등)을 맞추어 설치를 진행합니다.

```
curl -O https://repo.anaconda.com/archive/Anaconda3-2025.06-0-Linux-x86_64.sh
```

다음과 같이 파일이 다운로드 받아져야하며 약 1GB 가량의 파일이 다운로드 받아져야 합니다.

```
[ec2-user@ip-172-31-39-30 ~]$ curl -O https://repo.anaconda.com/archive/Anaconda3-2025.06-0-Linux-x86_64.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 84 1059M   84  895M    0     0   152M      0  0:00:06  0:00:05  0:00:01  146M
```

2. 다운로드 받은 파일을 실행하여 설치를 진행합니다.

```
bash ~/Anaconda3-2025.06-0-Linux-x86_64.sh
```

3. 다음 화면이 등장하면 Enter 키를 눌러 라이센스를 확인합니다.

```
[ec2-user@ip-172-31-39-30 ~]$ bash ~/Anaconda3-2025.06-0-Linux-x86_64.sh

Welcome to Anaconda3 2025.06-0

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

4. 라이센스를 확인하였으면 yes를 입력하여 라이센스에 동의합니다.

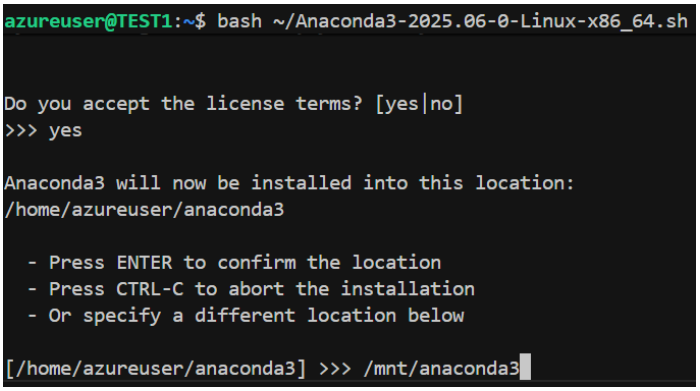
```
[ec2-user@ip-172-31-39-30 ~]$ bash ~/Anaconda3-2025.06-0-Linux-x86_64.sh

Welcome to Anaconda3 2025.06-0

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
By continuing installation, you hereby consent to the Anaconda Terms of Service available at https://anaconda.com/legal.


Do you accept the license terms? [yes|no]
>>> yes
```

5. 아나콘다 설치 위치를 지정합니다.

   설치 위치를 변경할 의도가 없다면 Enter를 입력합니다.

   단, 이번 실습에서는 이후 실습을 위해 다음 경로를 선택합니다.

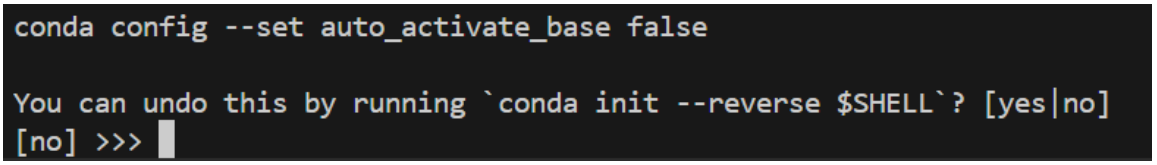
   `/mnt/anaconda3`



위와 같이 진행할 경우 해당 경로에 `anaconda3` 폴더가 생성되고 해당 폴더 안에 라이브러리 및 패키지가 설치되게 됩니다. 이후 Anconda 자체에 심각한 오류가 발생하여 개별 환경을 삭제하는 것으로 해결되지 않는 경우 해당 디렉토리를 삭제 후 Anaconda를 재설치 할 수 있습니다.

설치 완료까지는 일정 시간이 소요됩니다.

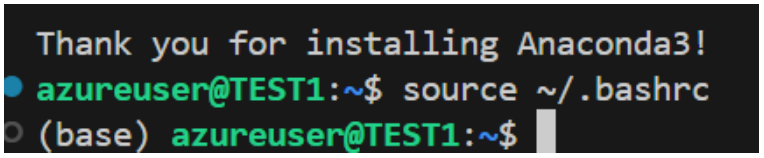
6. Anconda가 터미널 쉘에 기본적으로 적용 되기를 원하면 yes를 입력하십시오.



7. 아래 명령어를 입력하여 Anaconda를 활성화합니다.

```
source ~/.bashrc
```

아래처럼 앞에 (base) 혹은 (가상환경명)이 떠있어야 정상적으로 진행이 된 것입니다.



가상환경 준비하기

Anaconda가 설치되었다면 다음과 같이 가상 환경을 만들 수 있습니다.

```
conda create -n <환경이름> python=<버전>
```

```
# 예시
conda create -n main python=3.8
```

생성한 가상환경은 다음과 같이 켜고 끌 수 있습니다.

가상환경 켜기

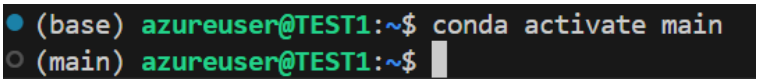
```
conda activate <환경이름>
```

```
# 예시
conda activate main
```

가상환경 끄기

```
conda deactivate
```

현재의 가상환경은 다음과 같이 앞 부분에 가상환경명을 통해 확인할 수 있습니다.



설치하자마자 적용된 base는 conda 자체가 설치된 환경이며 이 자체로도 가상환경이기 때문에 사용하여도 무방하지만 권장 사항은 각 프로젝트별로 별도의 가상환경을 만들고 진행하는 것입니다.

생성한 가상환경은 각 환경별로 독립적으로 동작하기 때문에 다른 곳에서 설치한 라이브러리를 사용할 수 없어 필요한 라이브러리로 재구성해주어야합니다. 설치는 다음과 같이 설치할 수 있습니다.

```
conda install <라이브러리명>
```

만일 Conda로 유지하고 싶지만 최신 라이브러리를 사용하고 싶으신 경우에는 다음과 같이 커뮤니티에 등록된 라이브러리를 쓸 수 있습니다.

```
conda install -c conda-forge <라이브러리명>
```

혹은 PYPI에서 가져오는 방법도 적용 가능합니다.

```
pip install <라이브러리명>
```

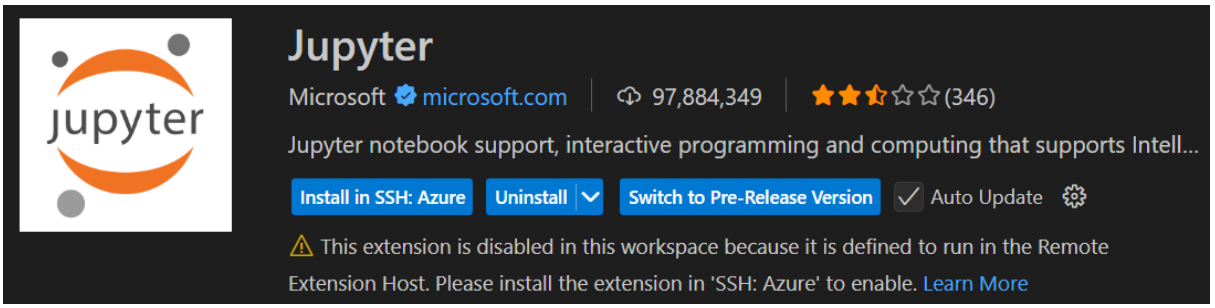
단, conda와 pip의 혼용은 추천하지 않습니다.

VScode에서 Jupyter Notebook 사용하기

다음 명령어를 통해 Jupyter Notebook을 설치합니다.

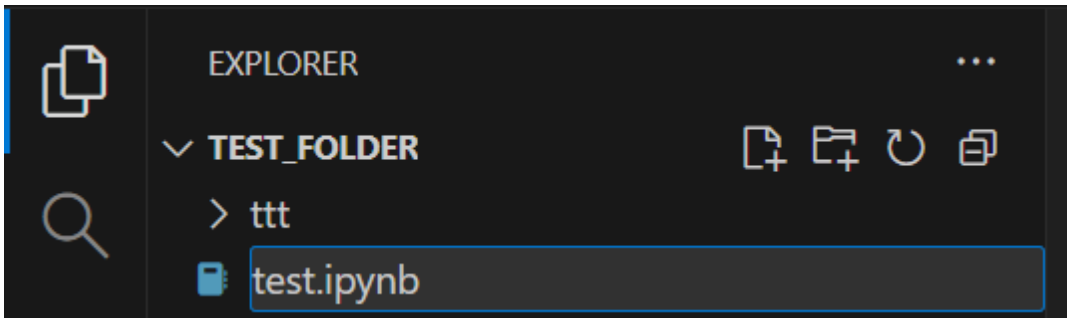
```
conda activate <가상환경명>
conda install jupyter
```

설치가 완료되었으면 우측 Extension을 열거나 ctrl+shift+x를 누르고 상단에 jupyter를 검색합니다.

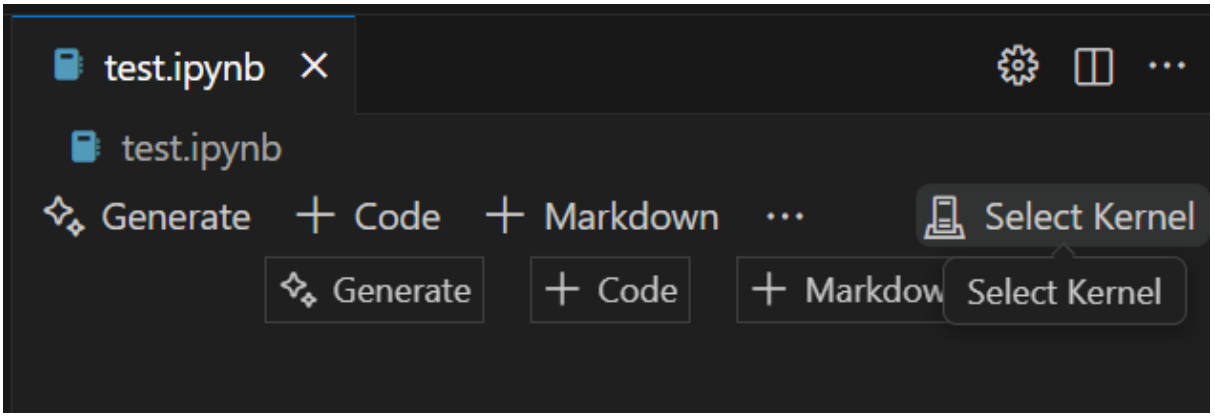


만일 로컬에도 설치가 되어있는 경우에는 `Install in SSH: <인스턴스명>`을 클릭하여 인스턴스 내에 설치가 진행되어야합니다.

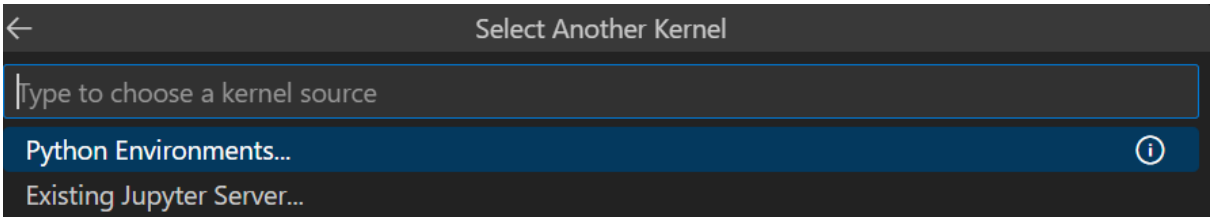
Extention 설치까지 완료되었으면 파일 매니저로 돌아가 ipynb 파일 하나를 생성합니다.



아래와 같이 생성한 ipynb 파일의 문서 우측 상단의 Select Kernel을 누릅니다.

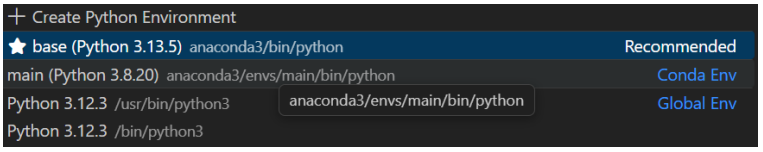


Python Environments...를 선택합니다.



노트북의 코드를 동작시킬 가상환경을 선택합니다.

만일 base만 떠있다면 **가상환경 준비하기**를 참조하여 가상환경을 먼저 만든 다음 진행하고 만일 base도 없이 Python 3.xx만 등장한다면 인스턴스에서 **아나콘다 설치하기**를 먼저 진행 후 이어서 해야 합니다.



가상환경 별 CUDA / cuDNN 설정하기

PyTorch는 OS의 CUDA, cuDNN으로 동작하지 않고 자체적인 런타임으로 동작합니다.

conda 패키지에 등록된 버전은 링크를 통해 확인할 수 있습니다.

25년 11월 기준 CUDA 12.4 버전까지 등록되어있습니다.

다음을 각각 진행하여 가상환경을 생성해보겠습니다.

```
conda create -n test38 python=3.8 -y
conda activate test38
conda install ipykernel pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
```

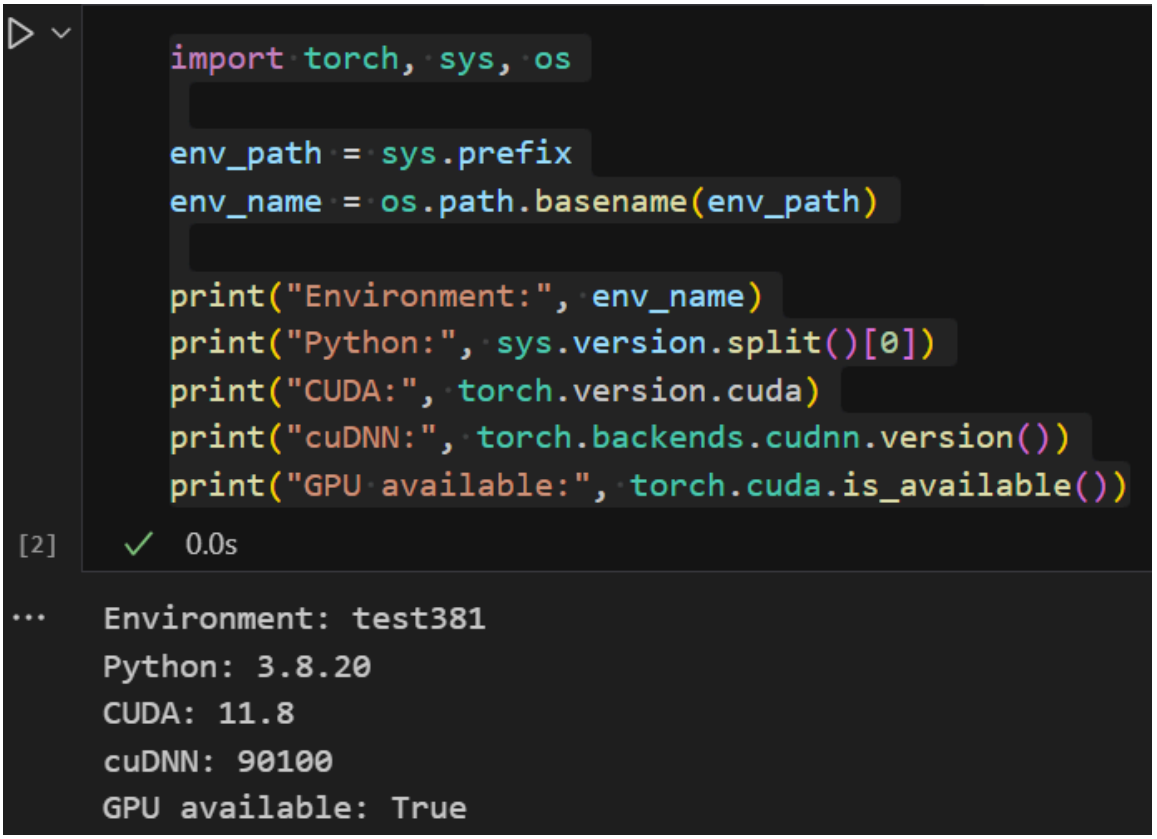
```
conda create -n test310 python=3.10 -y
conda activate test310

pip install ipykernel
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu124
```

이후 ipynb 파일에 다음 코드를 붙여넣고 각각의 가상환경에 따라 어떻게 결과가 나오는지 확인해보겠습니다.

```
import torch
print("CUDA:", torch.version.cuda)
print("cuDNN:", torch.backends.cudnn.version())
print("GPU available:", torch.cuda.is_available())
```

각각이 다음과 같은 결과가 나오면 성공입니다.

▼ test38



▼ test310

```
import torch, sys, os

env_path = sys.prefix
env_name = os.path.basename(env_path)

print("Environment:", env_name)
print("Python:", sys.version.split()[0])
print("CUDA:", torch.version.cuda)
print("cuDNN:", torch.backends.cudnn.version())
print("GPU available:", torch.cuda.is_available())
```

[1] ✓ 1.6s

```
Environment: test310
Python: 3.10.19
CUDA: 12.4
cuDNN: 90100
GPU available: True
```

실습하기

다음 레포지토리를 참고하여 각 프로젝트에 맞는 가상환경을 생성하고 모델 인퍼런스를 진행해보세요.

https://github.com/microsoft/BitNet/tree/main/gpu

https://github.com/facebookresearch/segment-anything

https://huggingface.co/deepseek-ai/DeepSeek-OCR