○ **개요**

본 문서는 2021년도 NIA 학습용 데이터 구축 지원사업 진행을 위해 제공하는 서버 사용법 가이드입니다.

○ **목차**

① 서버 접속 방법

- 원격으로 서버에 접속할 때 주로 SSH(Secure SHell)를 사용합니다. SSH는 물리적으로 멀리 떨어진 다른 PC나 서버에 접속해서 명령을 실행할 수 있도록 해주는 응용 프로그램 또는 프로토콜입니다. 본 문서에서는 SSH를 설정하고 서버에 접속하는데 편리한 'putty'와 'Visual Studio code'를 사용하는 방법을 알려드리겠습니다.

② FTP(File Transfer Protocol) 소프트웨어 사용법

- 서버에서 개발하는게 익숙하지 않은 분들은 서버와 로컬 상호 간에 데이터를 전송하거나 관리할 때 어려움이 있을 수 있습니다. 그래서 이러한 서버와 로컬 상호 간의 데이터를 전송하고 관리하는 기능을 지원하는 'FileZilla' 소프트웨어를 사용하는 방법을 알려드리겠습니다.

③ Anaconda 가상환경 설정 방법

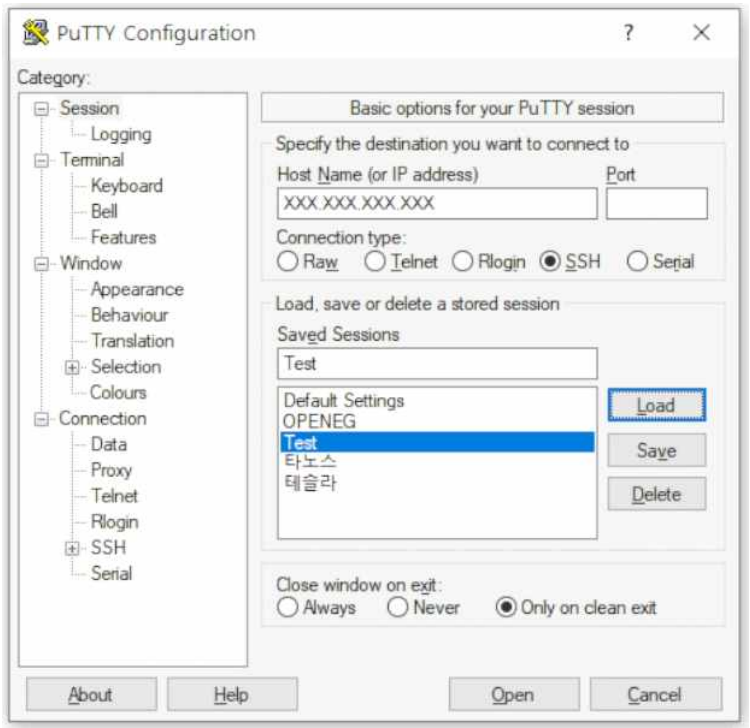
- Anaconda는 서버 내 다중 사용자들에게 독립적인 작업환경에서 패키지와 버전과 같은 디펜던시 관리에 용이한 가상환경을 제공합니다. 본 문서에서는 Anaconda로 가상환경을 생성하고 패키지의 버전을 관리하는 기본적인 사용방법에 대해 알려드리겠습니다.

④ 모델 활용 시 GPU 사용여부 확인 방법

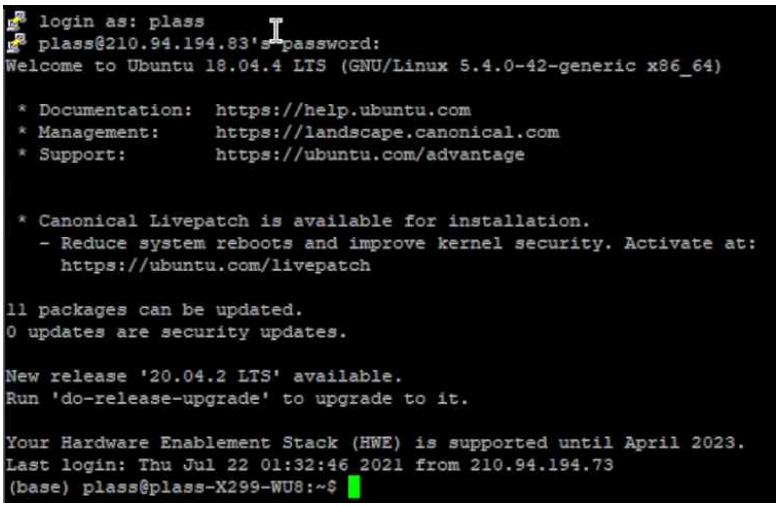
- 예시 소스코드를 통해 모델 학습 및 테스트를 수행할 때 CPU가 아닌 GPU를 사용하는지 확인하는 방법에 대해 알려드리겠습니다.

① 서버 접속 방법

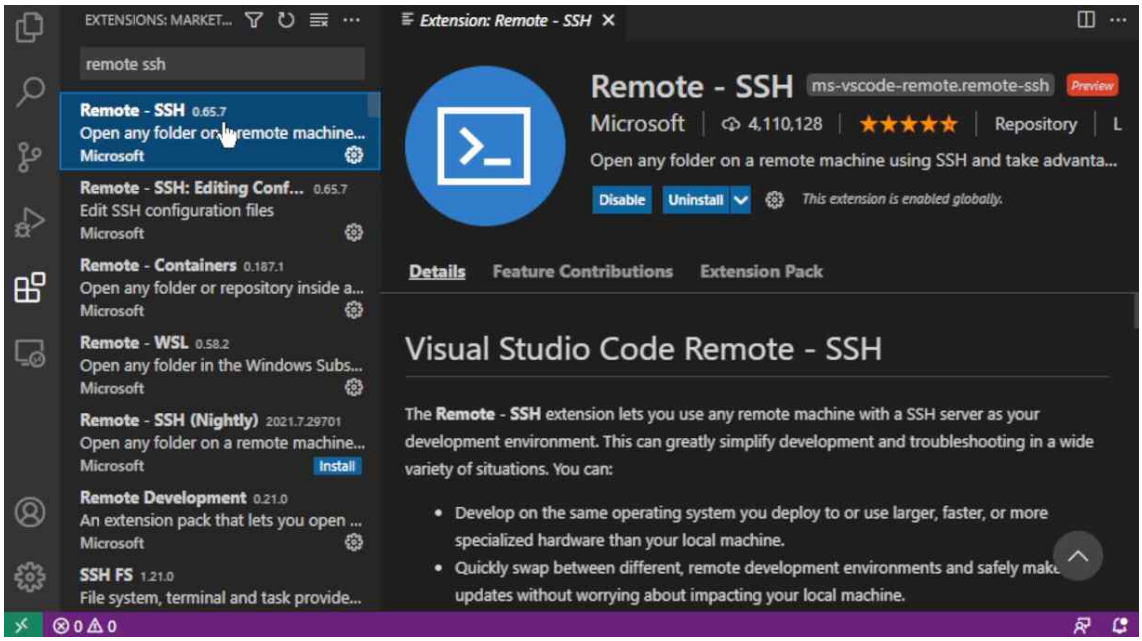
- putty는 실행하면 다음과 같은 화면을 보실 수 있습니다.



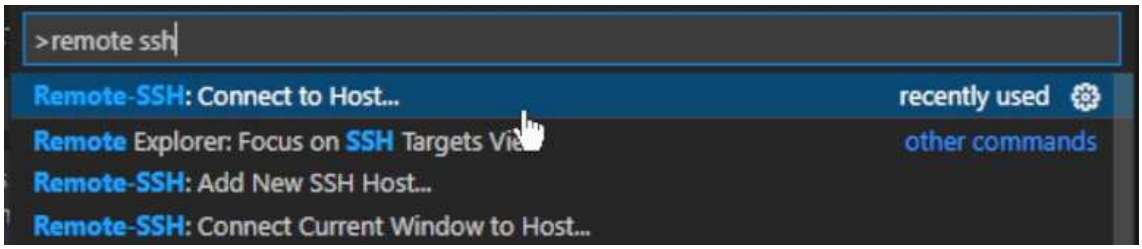
- 사전에 보내드린 서버 접속 정보를 통해 Host Name(or IP address), Port, Connection type(SSH)를 작성해주세요.
- 다음으로 계정 로그인을 위해 ID와 패스워드를 입력해주세요.
- 로그인이 완료되면 다음과 같은 화면이 출력되며, 쉘을 입력하여 서버를 사용하시면 됩니다.



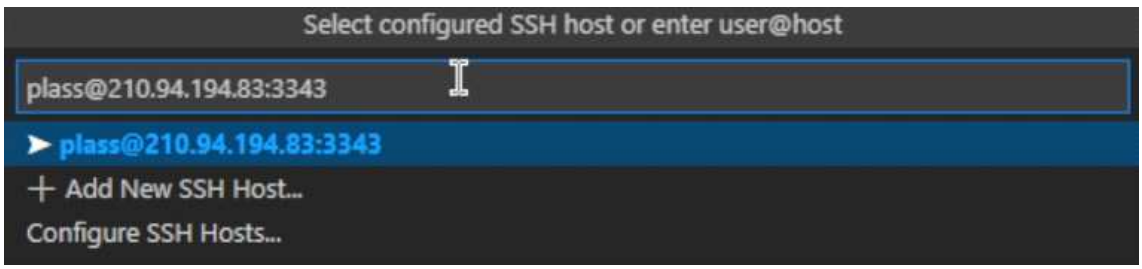
- Visual studio code(이하 VScode)는 좌측 extensions 탭에서 remote ssh를 검색하셔서 설치를 진행해주세요.



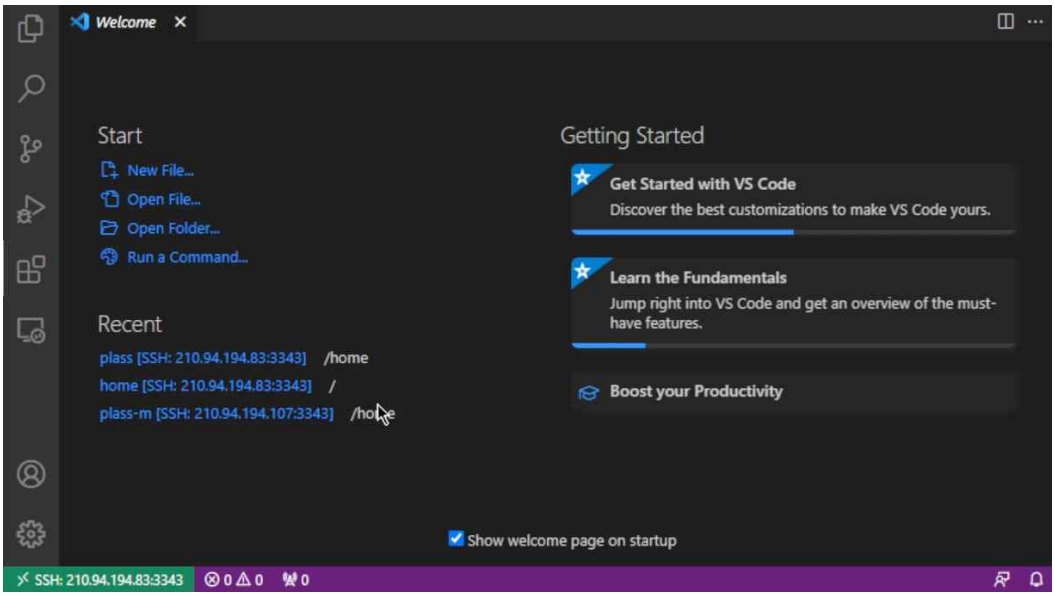
- 설치가 완료되면, F1을 누르셔서 Remote-SSH:Connect to Host...를 선택해주세요.



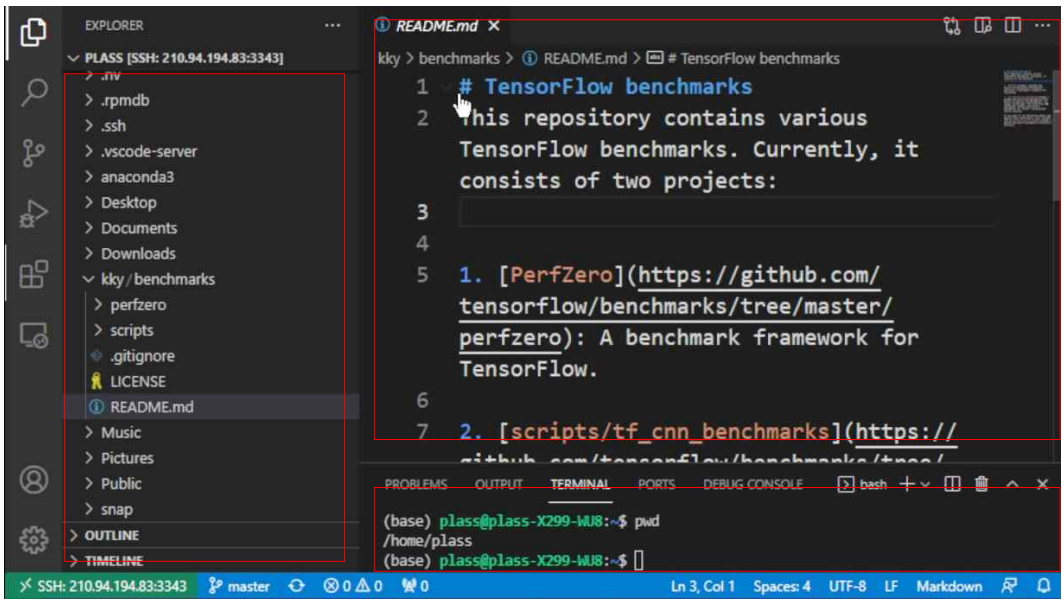
- SSH를 통한 호스트 연결을 선택하시면 다음과 같이 'user@host'를 요청합니다. 이때 다음 그림과 같이 userID@host(IP):port 번호를 입력해주세요.



- user와 host 정보를 입력하면 userID에 대한 비밀번호를 입력할 수 있으며, SSH 연결이 마무리됩니다. 서버에 연결되면 VScode 좌측 하단에 SSH:IP:port 정보를 보실 수 있습니다.

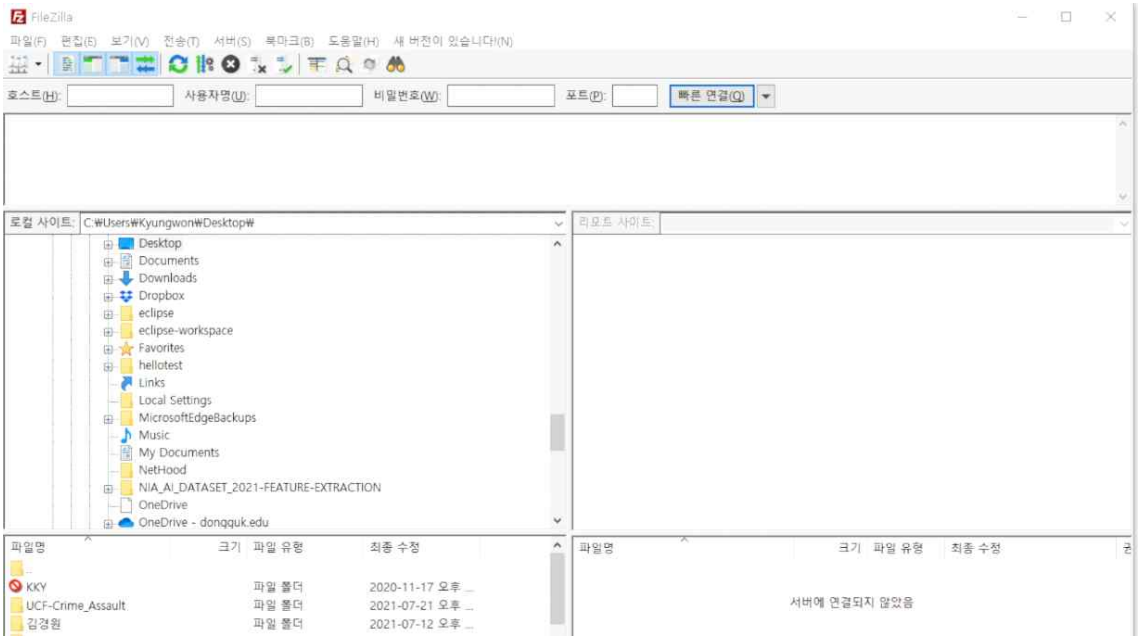


- 다음으로 좌측 explorer 탭에 접속하셔서 'Open Folder'를 클릭하시면 서버 디렉토리를 설정할 수 있습니다.
- 디렉토리 설정이 완료되면, 평소 IDE를 사용하시던 것처럼 프로그래밍하시거나 putty와 같이 터미널 기능도 활용하실 수 있습니다.

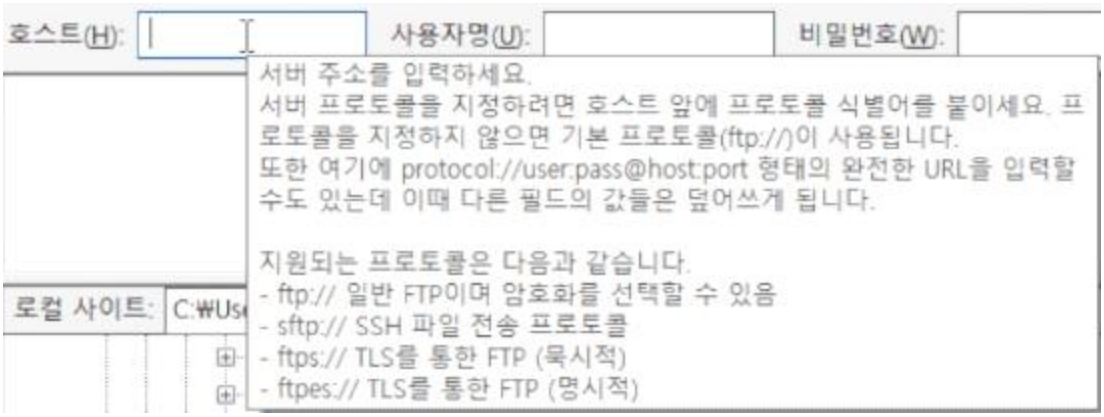


② FTP(File Transfer Protocol) 소프트웨어 사용법

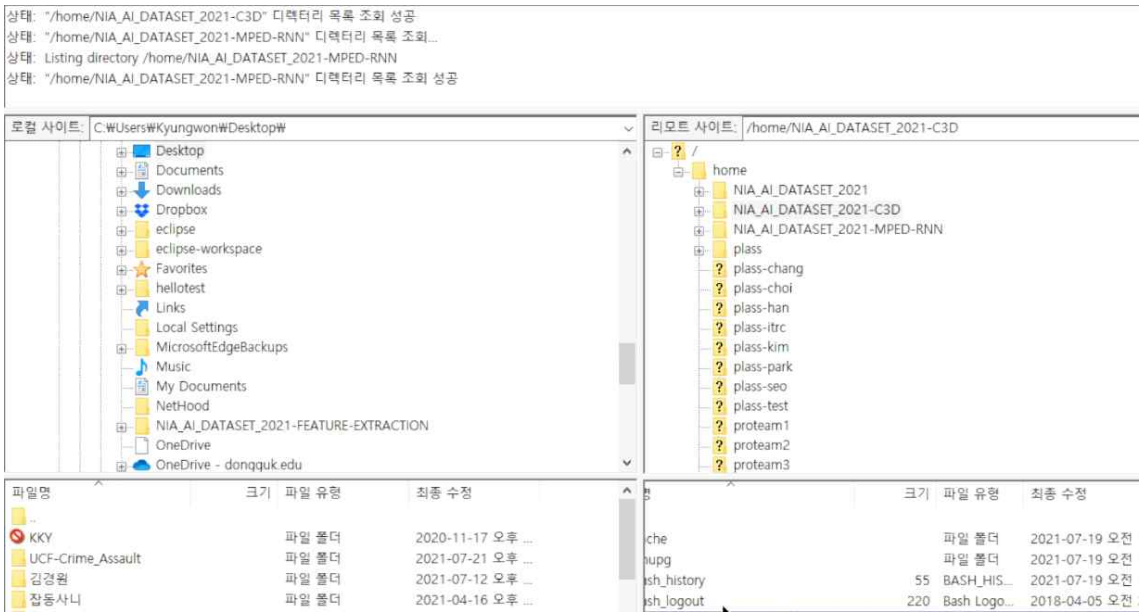
- FileZilla 소프트웨어를 사용하는 방법은 이전의 서버 접속 방법과 동일하게 host, userid, password, port를 입력하시면 됩니다.
- FileZilla를 실행하시면 다음과 같은 초기화면을 보실 수 있습니다.



- 상단에서는 호스트, 사용자명, 비밀번호, 포트를 입력하는 부분이며, 좌측은 로컬 환경, 우측은 서버 환경입니다. 아직 서버에 연결을 하지 않았으므로 우측은 비어있는 상태입니다.
- 호스트를 입력하실 때 다음 그림과 같이 지원되는 프로토콜을 명시해주셔야 합니다. 예시로 sftp://XXX.XXX.XXX.XXX와 같이 입력해주시면 됩니다.



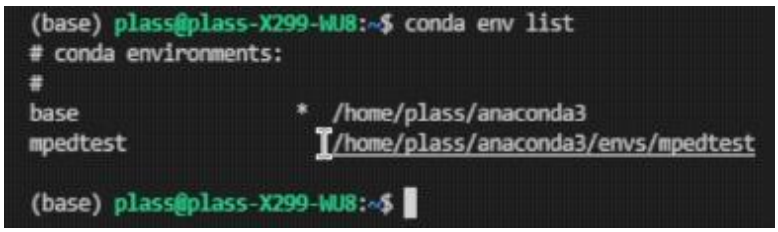
- 서버 연결이 완료되면 우측 서버 환경에 디렉토리 목록을 조회하실 수 있습니다.



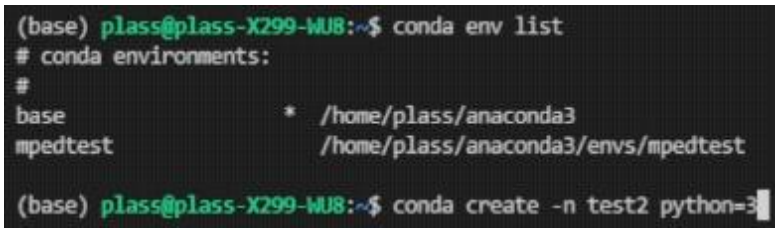
- 이를 통해 로컬과 서버 상호간의 파일을 공유하거나 관리할 수 있으며, 드래그 앤 드롭으로 간편하게 사용하실 수 있습니다.

③ Anaconda 가상환경 설정 방법

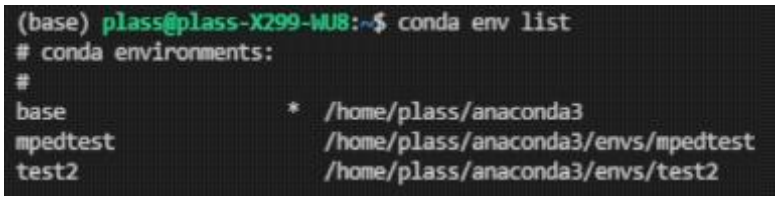
- 현재 모든 서버에 Anaconda 설치를 완료한 상태입니다. 그러므로 다음 그림과 같이 'conda env list' 명령어를 사용해서 현재 생성한 Anaconda 가상환경을 조회할 수 있습니다.



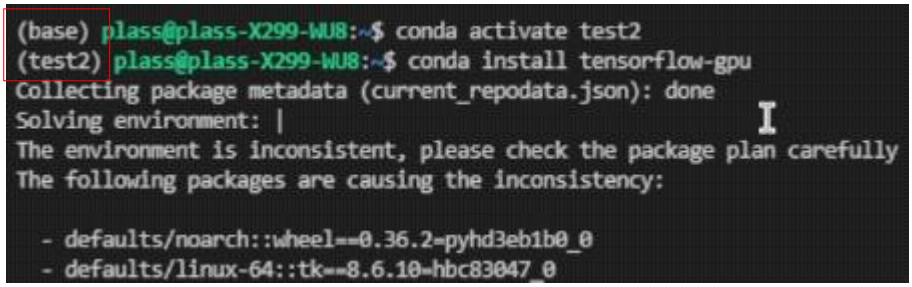
- 위 그림에서는 base라는 기본 가상환경이 있고, 다른 서버 이용자가 추가한 mpedtest 가상환경이 조회되는 것을 알 수 있습니다.



- 새로운 가상환경을 생성하기 위해서는 'conda create -n [가상환경 이름] python=3'와 같이 conda 명령어를 사용할 수 있습니다.



- 가상환경을 생성한 이후 처음과 같이 'conda env list'로 생성된 가상환경 목록을 조회하면 추가된 가상환경 목록을 확인할 수 있습니다.
- 해당 단계까지는 가상환경을 생성하였으나 접속하지는 않은 상태입니다. 즉, 활성화되어 있지 않은 상태이므로 추가한 가상환경에 접속할 수 있는 'conda activate [가상환경이름]' 명령어를 입력해주세요.



- 가상환경을 활성화하면 서버 계정정보 맨 앞부분의 괄호가 활성화된 가상환경 이름으로 변경되며, 해당 가상환경을 활성화하여 접속한 상태를 나타냅니다.
- 이후 가상환경에 독립적인 패키지 혹은 라이브러리 설치는 'conda install [package]' 명령어를 사용할 수 있습니다.

※ Anaconda 명령어를 추가로 검색하셔서 패키지 설치와 관련된 내용을 추가로 살펴보시길 바랍니다.

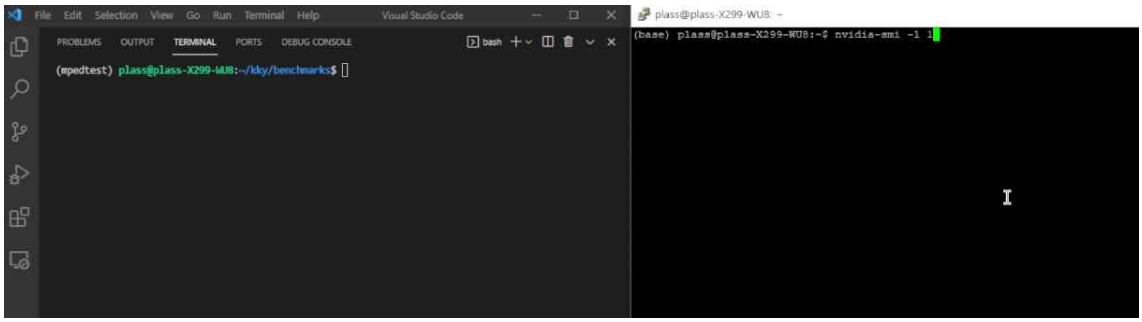
- 활성화된 가상환경을 비활성화 하는 방법은 'conda deactivate' 명령어를 사용하시면 됩니다.
- 가상환경이 비활성화되면 기본설정으로 되어있는 base 환경으로 설정됩니다.

※ Anaconda 가상환경 사용 시 주의사항:

- 팀별 서버 계정이 하나이므로, 팀 내부에서도 가상환경을 관리하시다가 이슈가 생길 수 있습니다. 그러므로 반드시 본인이 만든 가상환경을 활성화하여 사용하시기 바랍니다.
- 또한, 사용자의 실수로 인하여 언제든 가상환경에 문제가 생길 수 있는 경우를 대비하기 위하여 .yaml 확장자 파일로 Anaconda 가상환경을 내보내기(export) 하여 백업용으로 준비해두시는 것도 좋은 방법입니다.

④ 모델 활용 시 GPU 사용여부 확인 방법

- 서버 내에서 모델을 활용할 때 GPU를 사용하는지 확인하기 위해 두 개의 터미널 창을 구성하였습니다.(좌측은 VScode, 우측은 putty)



- 우선 우측의 putty에서 서버 GPU 사용량을 체크하기 위해 'nvidia-smi -l 1' 명령어를 실행합니다. 해당 명령어에서 숫자 1은 1초마다 GPU 사용량을 출력하는 옵션입니다.

```
Thu Jul 22 02:40:52 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.100      Driver Version: 440.100      CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce RTX 208...  Off  | 00000000:19:00.0 Off |                  N/A |
|  0%   34C    P8     7W / 250W |      1MiB / 11019MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce RTX 208...  Off  | 00000000:1A:00.0 Off |                  N/A |
|  0%   33C    P8     6W / 250W |      1MiB / 11019MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce RTX 208...  Off  | 00000000:67:00.0 Off |                  N/A |
|  0%   32C    P8     7W / 250W |      1MiB / 11019MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GeForce RTX 208...  Off  | 00000000:68:00.0 Off |                  N/A |
|  0%   32C    P8     8W / 250W |     80MiB / 11018MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    3      1447      G   /usr/lib/xorg/Xorg                            56MiB |
|    3      1628      G   /usr/bin/gnome-shell                          21MiB |
```

- 위 그림으로 알 수 있는 것은 1초 단위로 갱신되는 서버 내 GPU 구성을 확인할 수 있습니다. 해당 서버는 GeForce RTX 2080TI 4개로 구성되어 있으며, 현재 사용률은 0%입니다.

- GPU 사용량을 테스트하기 위해 텐서플로우를 통한 성능 측정 벤치마크를 활용하였습니다.

  ※ https://hiseon.me/data-analytics/tensorflow/tensorflow-benchmark/

- 벤치마크된 모델들을 스크립트를 통해 실행시키는 명령어는 다음과 같습니다.

```
$ python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --num_gpus=1
--batch_size=32 --model=resnet50 --variable_update=parameter_server
```

- 위 명령어에서 핵심은 –num_gpus=1 옵션입니다. 즉, 해당 벤치마크는 모델을 실행시킬 때 사용하는 GPU의 개수를 옵션으로 전달받아 GPU를 사용하는 것을 알 수 있습니다.

- 테스트를 위해 –num_gpus=4를 옵션으로 벤치마크된 모델을 실행하면 다음과 같은 GPU 사용량을 보실 수 있습니다.

```
| NVIDIA-SMI 440.100      Driver Version: 440.100      CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce RTX 208...  Off  | 00000000:19:00.0 Off |                  N/A |
|  0%   57C    P2   172W / 250W |  10775MiB / 11019MiB |     76%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce RTX 208...  Off  | 00000000:1A:00.0 Off |                  N/A |
|  0%   56C    P2   231W / 250W |  10775MiB / 11019MiB |     76%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce RTX 208...  Off  | 00000000:67:00.0 Off |                  N/A |
|  0%   53C    P2   253W / 250W |  10775MiB / 11019MiB |     78%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GeForce RTX 208...  Off  | 00000000:68:00.0 Off |                  N/A |
|  0%   57C    P2   203W / 250W |  10780MiB / 11018MiB |     77%      Default |
+-------------------------------+----------------------+----------------------+
```

- 이와 같은 방법으로, 팀마다 활용하고자 하는 모델이 학습이나 테스트를 수행할 때 GPU를 사용하는지 확인할 수 있습니다.
- 중요한 점은 모델마다 GPU 개수를 옵션으로 받을지, GPU 사용여부를 소스코드 내부에서 설정하였는지 반드시 확인하셔야 합니다.