

# 서버 연결 및 쥬피터노트북 작업

👤 생성자	📧 버리 문
🕒 생성 일시	@2024년 9월 23일 오후 3:19
🏷 태그	

이 페이지에서는 학교 서버에 접속하여, 작업을 수행하기 위한 설정 및 쥬피터 노트북 파일을 여는 방법과, 작업과정을 정리해 놓는다.

## 학교 서버 접속

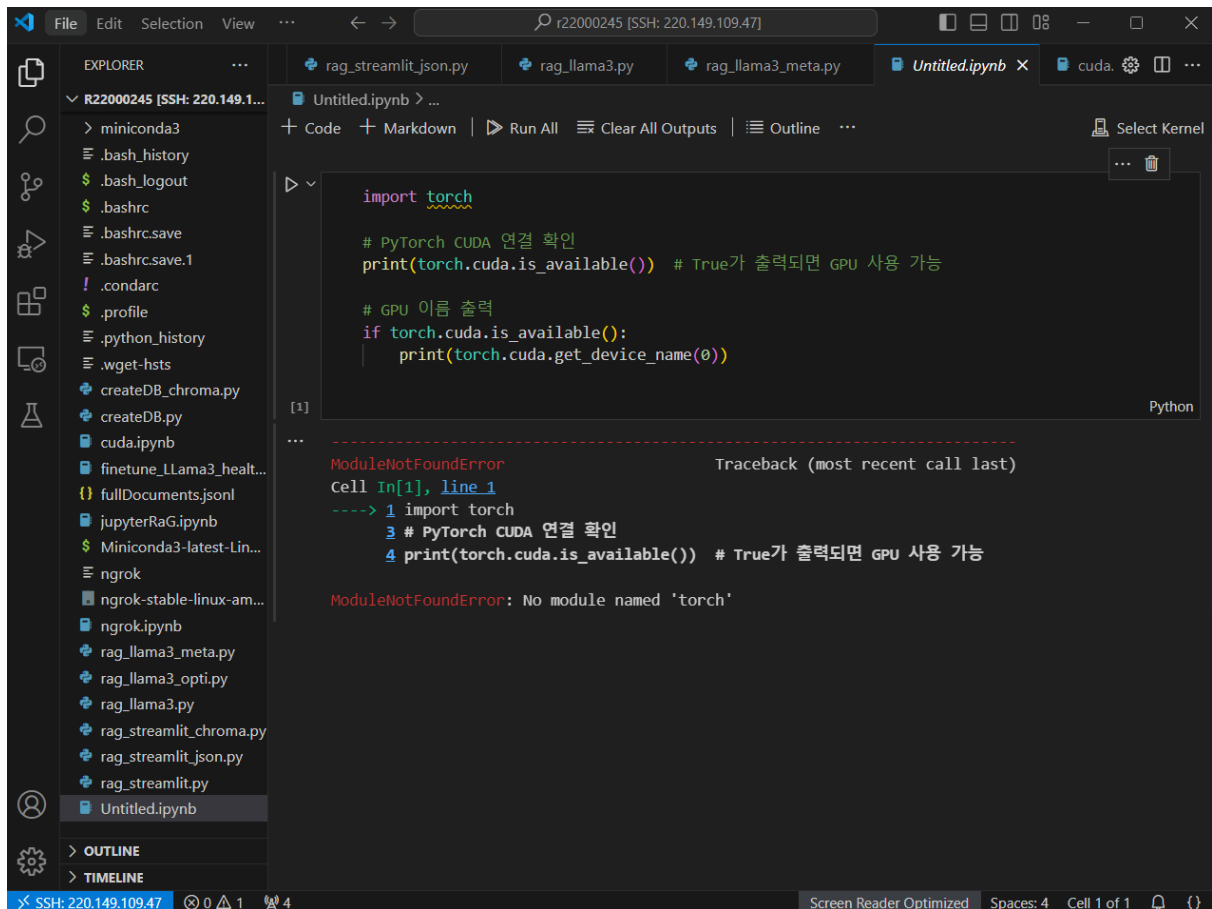
VS Code에서 `ctrl+shift+P` → Remote-SSH : Add New SSH Host를 선택한 뒤

내가 할당받은 ID와 주소를 입력한다

```
ssh r22000245@220.149.109.47
```

이후 동일하게 `ctrl+shift+P` → Remote-SSH : Connect to SSH를 통해 저장한 주소, 220.149.109.47로 접속한다

이후 서버에 대한 password를 입력하고, `open file` → `home/r2200245` 작업공간으로 들어간다.



해당 이미지와 같이 되면 성공

## 로그인 서버에서 GPU를 할당받은 후 서버에 접속

지금 접속한 장소는 로그인서버이며, 해당 디렉토리에 여러 작업물들이 올라가 있기는 하지만 해당 내용들을 곧바로 사용할수는 없다. 이제 터미널을 열고 서버 및 환경을 만들어야 한다.

```
bsub -Is -q jupyter_gpu -W 4:00 -gpu "num=1:mode=exclusive
_process:mig=4" bash
```

쥬피터 노트북의 gpu버전을 사용하고, 사용시간은 4시간, gpu 1개, mig 4개를 요청한다. 이후에

(dev) r22000245@hgucoss:~\$ 이 상태에서

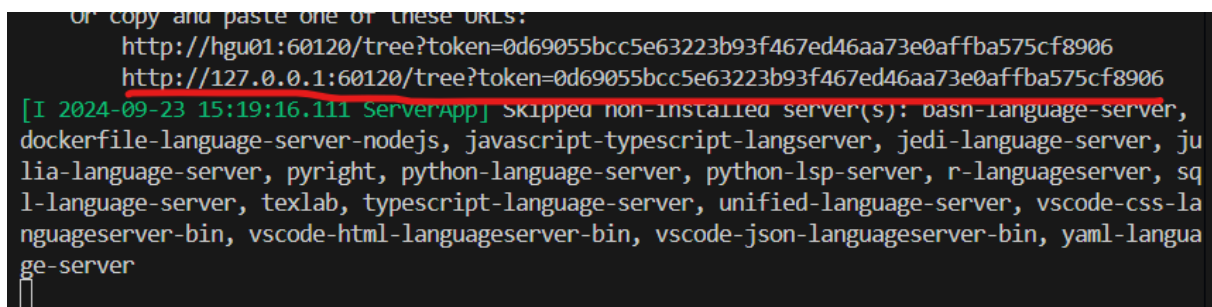
r22000245@hgu01:~\$ 이렇게 서버가 자동으로 할당되어 @뒷부분이 바뀔 것이다. 이때 내가 속한 서버는 hgu01서버이며, 서버의 상황에 따라 hgu02서버로 할당받을 수도 있다.

이후

```
conda activate dev
jupyter notebook --no-browser --ip=0.0.0.0 --port=60120
```

을 통해 conda 환경을 실행시키고 주피터 노트북을 실행시킨다. 이때 사용되는 포트는 60120으로 내게 배정된 포트에 연결한다.

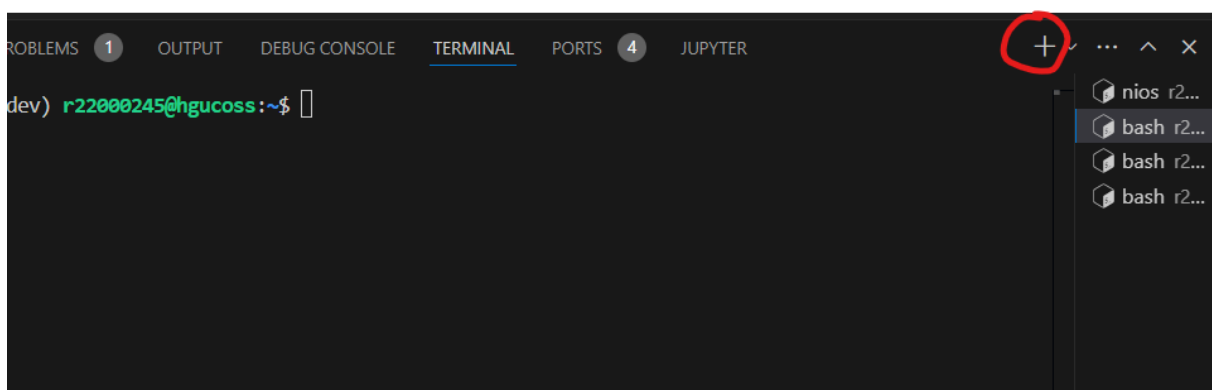
이후 터미널에서



```
or copy and paste one of these URLs:
http://hgu01:60120/tree?token=0d69055bcc5e63223b93f467ed46aa73e0affba575cf8906
http://127.0.0.1:60120/tree?token=0d69055bcc5e63223b93f467ed46aa73e0affba575cf8906
[I 2024-09-23 15:19:16.111 ServerApp] Skipped non-installed server(s): basn-language-server,
dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, ju
lia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sq
l-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-la
nguageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yamllangua
ge-server
```

다음 밑줄 친 부분과 같이 url을 제공해주는데 이를 복사하여 브라우저로 실행시키자  
그러면 당연히 아직은 아무페이지도 열리지 않는다.

다음으로는 두 개의 새로운 터미널을 열어준다



지금 표시된 + 버튼을 클릭하면 새로운 터미널이 실행되고, 해당 위치에서는 다시 로그인 서  
버에서 접속한 위치로 표시된다. 이제 이 터미널에

```
ssh -L :60120:localhost:60120 -N r22000245@hgu01
```

다음과 같은 코드를 입력하여 60120포트에 대한 접속을 시도한다.

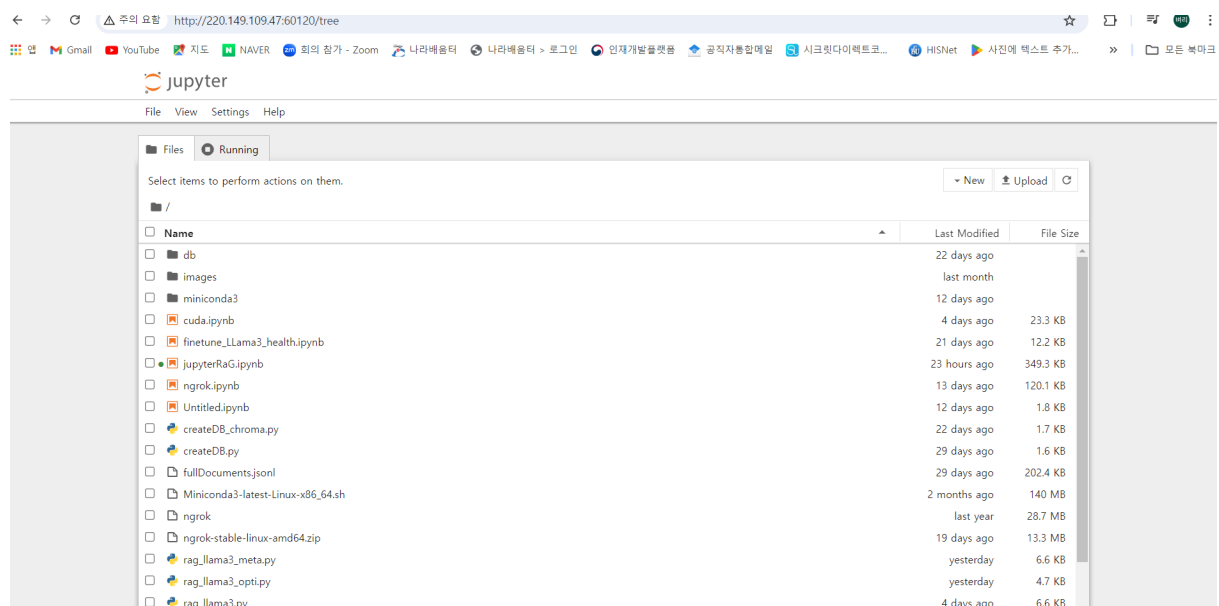
이때 코드는 내가 쥬피터 노트북을 요청하고 할당받은 서버와 동일해야 한다.

ex) 만약 할당받은 서버가 hgu02라면,

```
ssh -L :60120:localhost:60120 -N r22000245@hgu02
```

다음과 같은 코드를 통해 접속해야 함

이제 아까 복사한 웹 브라우저에서 ip주소 부분을 220.149.109.47로 변경한 뒤 접속을 시도하면 쥬피터 노트북이 실행된다.



다음 이미지처럼 웹 브라우저에서 쥬피터 노트북이 실행되면 성공이다.

## streamlit 코드 실행

쥬피터 노트북에서 만들어놓은 다양한 파일들 중, streamlit코드를 실행시키기 위해서 ipynb파일 중 jupyterRaG 파일에 들어간다.

해당 위치에서 여러 코드 셀이 존재하지만 우선 streamlit파일 실행을 위해 필요한 패키지 다운을 먼저 시도한다

```
!pip install chromadb tiktoken transformers sentence_transformers jq
!pip install openai langchain
!pip install -U langchain-openai langchain-community
```

```
pip install faiss-cpu
```

```
pip install streamlit loguru langchain openai faiss-cpu sentence-transformers langchain-huggingface langchain-community
```

이러한 패키지 다운을 먼저 시도한 후, 다음으로 DB를 생성한다

DB생성은 실행하고자 하는 Rag streamlit코드가 DB를 사용하는 것을 전제로 만들어졌기 때문에 필요하다.

```
!python createDB_chroma.py
```

동일한 디렉토리에 위치한 createDB\_chroma.py 라는 코드를 실행시켜 DB를 생성하고, streamlit 코드를 실행한다

```
!streamlit run rag_llama3.py --server.port=60121
```

rag\_llama3.py라는 코드를 streamlit run으로 실행하며 사용하는 포트는 60121이다. 이 때, 서버 터미널에서 60121에 대한 권한을 받지 못하면 열리지 않는다

다시 VS Code 터미널을 새로 열어준 뒤,

```
ssh -L :60121:localhost:60121 -N r22000245@hgu01
```

서버에 대한 60121 포트를 열고 브라우저에서 코드를 실행하면 정상적으로 rag\_llama3.py 코드가 작동할 것이다.

```
[ ]: !streamlit run rag_llama3.py --server.port=60121
```

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

Local URL: <http://localhost:60121>  
Network URL: <http://192.168.1.48:60121>  
External URL: <http://220.149.109.55:60121>

2024-09-17 21:40:45.141 Uncaught exception GET /\_stcore/stream (127.0.0.1)  
HTTPServerRequest(protocol='http', host='localhost:60121', method='GET', uri='/\_stcore/stream', version='HTTP/1.1', remote\_ip='127.0.0.1')  
Traceback (most recent call last):  
File "/home/r22000245/miniconda3/envs/dev/lib/python3.11/site-packages/tornado/websocket.py", line 938, in \_accept\_connection  
open\_result = handler.open(\*handler.open\_args, \*\*handler.open\_kwargs)