



# Urban Bigdata and AI Institute of University of Seoul

---

슈퍼컴퓨터 사용법 교육

# 01 교육프로그램 및 UBAI 소개

## 교육 목차

시작 시간	종료 시간	소요 시간	교육 내용
17:30	17:40	10min	교육 프로그램 및 UBAI 소개
17:40	18:10	30min	UBAI 서버 접속
18:10	18:30	20min	Python 환경 구축 (Miniconda)
18:30	18:40	10min	쉬는 시간
18:40	19:30	50min	슈퍼컴퓨팅 환경 이해 및 Job 제출

- WIFI: **UBAI\_CO\_SPACE\_01\_5G/6G** 또는 **UBAI\_CO\_SPACE\_02\_5G/6G** 로 접속
  - PW: *bigdata2025*

# 01 교육프로그램 및 UBAI 소개

## 교육 자료 다운로드

- [https://github.com/uos-ubai/practice\\_november](https://github.com/uos-ubai/practice_november)
- 슈퍼컴퓨터 사용법 교육.pdf, edu.txt, predict.py, requirements.txt, run.sh를 다운 받아주세요.

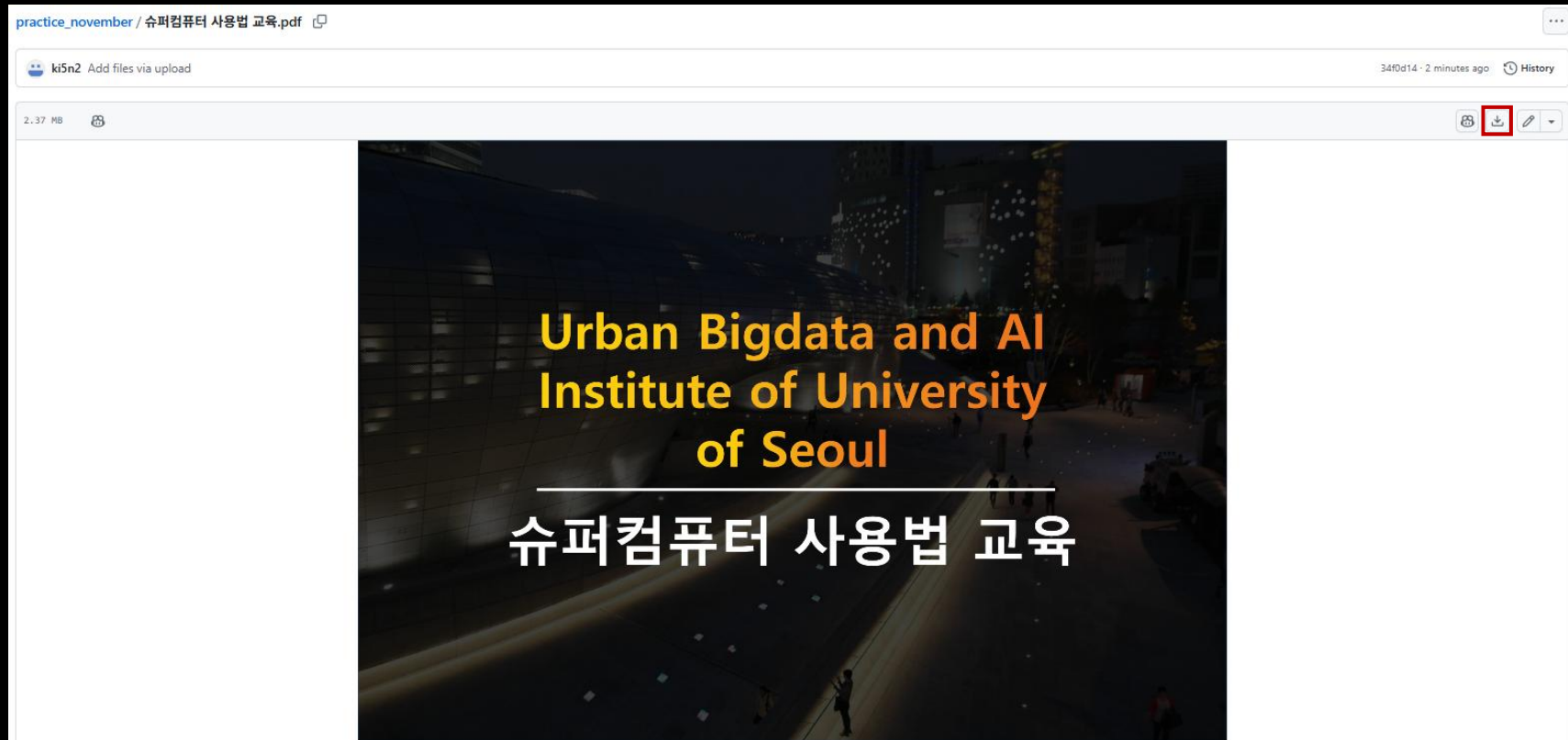
The screenshot shows the GitHub repository page for 'practice\_november'. The repository is private and has 1 branch and 0 tags. The commit history shows a commit by 'ki5n2' titled 'Delete jupyter.sh' with 32 commits. The file list includes:

File	Commit Message	Time
README.md	Update README.md	last week
edu.txt	Update edu.txt	20 hours ago
multi_run_files.zip	Add files via upload	17 hours ago
predict.py	Rename test.py to predict.py	3 weeks ago
requirements.txt	Create requirements.txt	last month
run.sh	Update run.sh	last week
슈퍼컴퓨터 사용법 교육.pdf	Add files via upload	16 hours ago

# 01 교육프로그램 및 UBAI 소개

## 교육 자료 다운로드

- [https://github.com/uos-ubai/practice\\_november](https://github.com/uos-ubai/practice_november)
- 슈퍼컴퓨터 사용법 교육.pdf, edu.txt, predict.py, requirements.txt, run.sh를 다운 받아주세요.



## 슈퍼컴퓨터란 무엇인가?

- 슈퍼컴퓨터(Supercomputer)는 일반 컴퓨터보다 훨씬 빠른 속도로 방대한 계산을 수행할 수 있는 고성능 컴퓨터(HPC, High Performance Computing)입니다.
- 일반 PC가 1개의 CPU로 일처리를 한다면, 슈퍼컴퓨터는 수많은 CPU, 또는 GPU를 동시에 사용하여 병렬로 작업을 처리합니다.
- 주로 기후 및 분자 시뮬레이션, 유전체 분석, 인공지능 모델 학습 등과 같은 대규모 과학 및 공학 문제 해결에 사용됩니다.

## 도시과학빅데이터·AI연구원(UBAI) 운영방향

### 01 도시과학 연구

---

도시 경쟁력을 갖춘 선도도시의 기반이 되는  
도시과학 연구

### 02 인재양성

---

빅데이터와 인공지능 분야의  
역량을 갖춘 인재 양성

### 03 시스템 구축

---

창업 및 신규 첨단산업 생태계 구축  
최첨단 빅데이터·인공지능 기술 개발 및 인프라  
제공

### 04 대·내외 협력

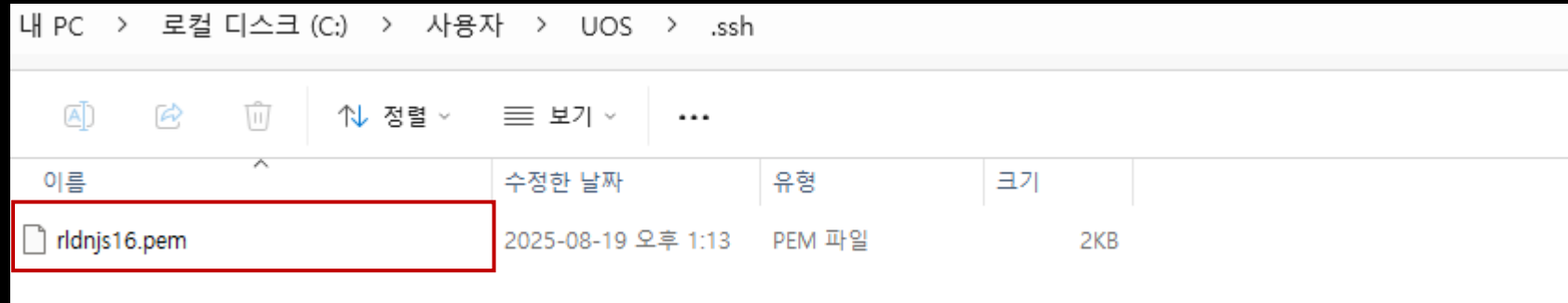
---

서울 및 국내외 기관과의 대내외 협력

## 02 UBAI 서버 접속

### 계정 생성

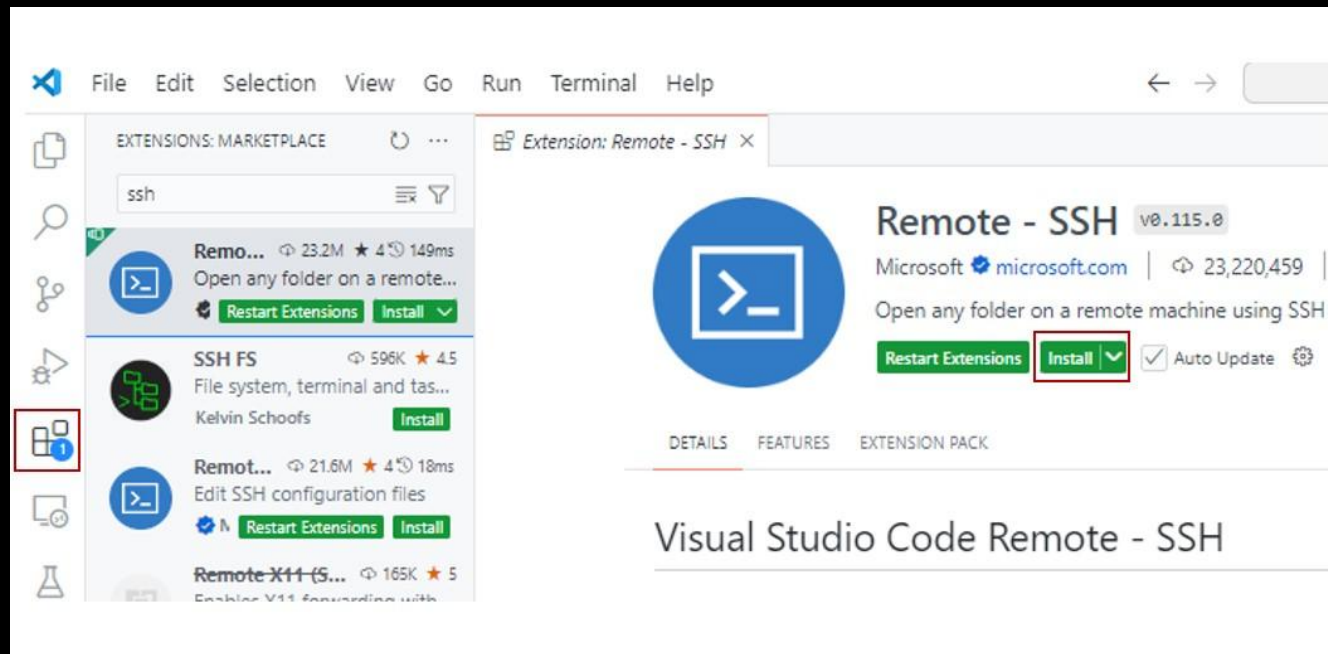
- UBAI Cluster 사용을 위해서는 사용자 계정을 발급받아야 합니다.
- 키 파일을 받은 상태에서 실습에 참가해야 합니다.
- 키 파일을 다음과 같은 경로에 저장합니다.
  - Windows : C:\Users\사용자 이름\.ssh\사용자 계정.pem
  - MacOS : /Users/사용자 이름/.ssh/사용자 계정.pem



## 02 UBAI 서버 접속

### 접속 방법

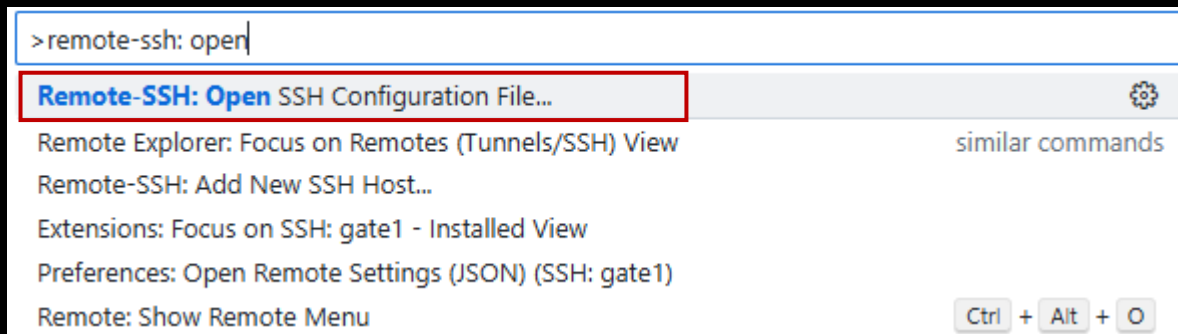
- Visual Studio Code 다운로드 및 접속
  - <https://code.visualstudio.com/download>
- Remote-SSH Extension 설치





### 접속 방법

- Config 파일 수정
  - Ctrl + Shift + P
  - Remote-SSH: Open SSH Configuration File...

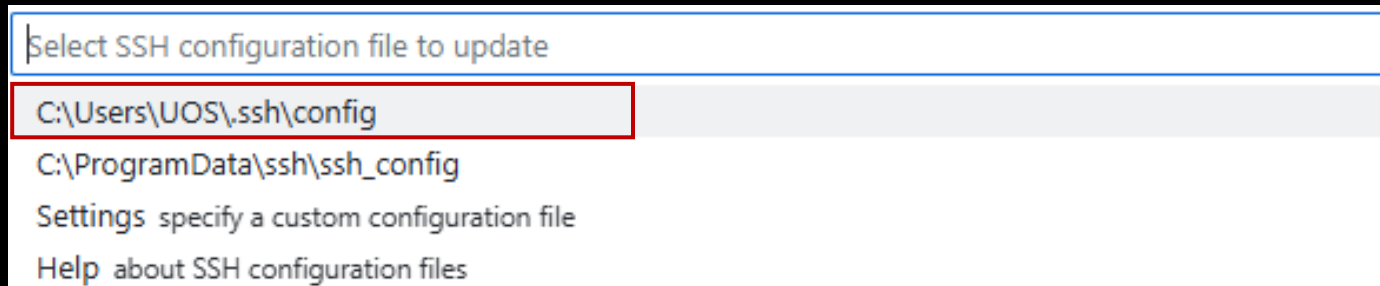


- .ssh폴더가 없는 경우에는 .ssh폴더와 config파일을 같이 생성
- .ssh폴더가 있는 경우에는 config파일만 생성

## 02 UBAI 서버 접속

### 접속 방법

- Config 파일 수정
  - 이후, 입력창에 C:\Users\사용자 이름\.ssh\config 선택



### 접속 방법

- Config 파일 수정

- Host:

- gate1
    - gate2

- HostName:

- 172.16.10.36(gate1인 경우)
    - 172.16.10.37(gate2인 경우)

- Port: 22

- User: 사용자ID

- IdentityFile: 키파일 경로

- Windows : C:\Users\사용자 이름\.ssh\사용자 계정.pem
    - MacOS : /Users/사용자 이름/.ssh/사용자 계정.pem

사진은 예시이므로, 사용자 환경에 맞게 반드시 수정합니다.

```
config
C: > Users > UOS > .ssh > config
1 Host gate1
2     HostName 172.16.10.36
3     Port 22
4     User rldnjs16
5     IdentityFile C:\Users\UOS\.ssh\rldnjs16.pem
6
7 Host gate2
8     HostName 172.16.10.37
9     Port 22
10    User rldnjs16
11    IdentityFile C:\Users\UOS\.ssh\rldnjs16.pem
12
```

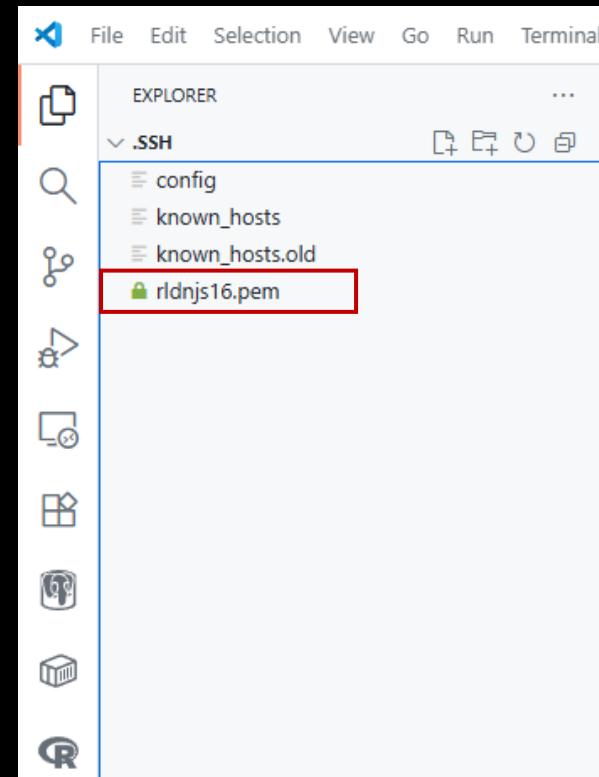
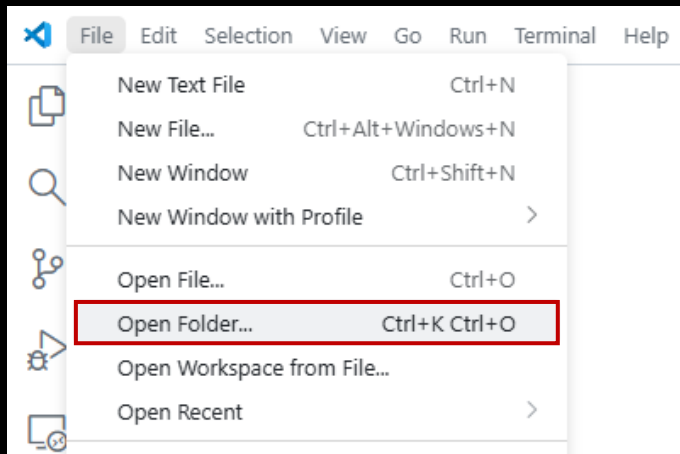
### 접속 방법(Only Mac User)

- MacOS 사용자는 다음과 같은 두 가지 추가 작업이 요구됨
- 키 권한 작업
  1. Terminal – New Terminal 열기
  2. **chmod 400 <비밀키 위치 경로 및 비밀키>**  
**Ex) chmod 400 /Users/UOS/.ssh/rldnjs16.pem**
    1. UOS를 사용자 이름으로 변경
    2. rldnjs16을 사용자 계정으로 변경
- 키 포맷 변경 작업
  1. VS code 내에서 키 파일(사용자 계정.pem) 열기
    - 키 파일 위치: /Users/사용자 이름/.ssh/사용자 계정.pem
  2. 하단 메뉴에서 파일 포맷 확인 후 클릭
  3. **파일 포맷 LF로 변경**
  4. 변경 사항 저장(또는 덮어쓰기)

## 02 UBAI 서버 접속

### 접속 방법(Only Mac User)

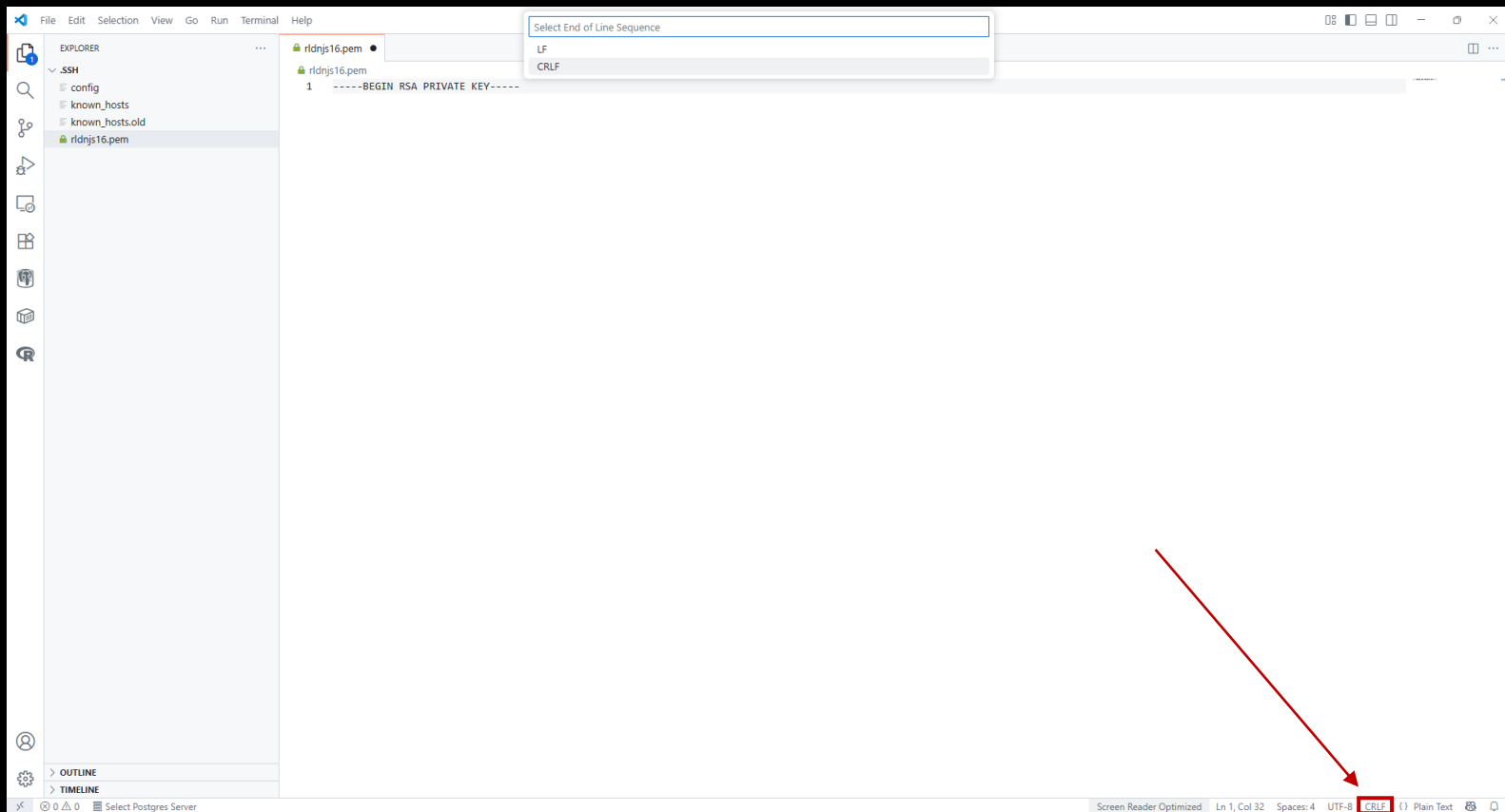
- 파일 열기
  - File – Open Floder 선택
  - 키 파일 저장한 “/Users/사용자 이름/.ssh”폴더 열기



## 02 UBAI 서버 접속

### 접속 방법(Only Mac User)

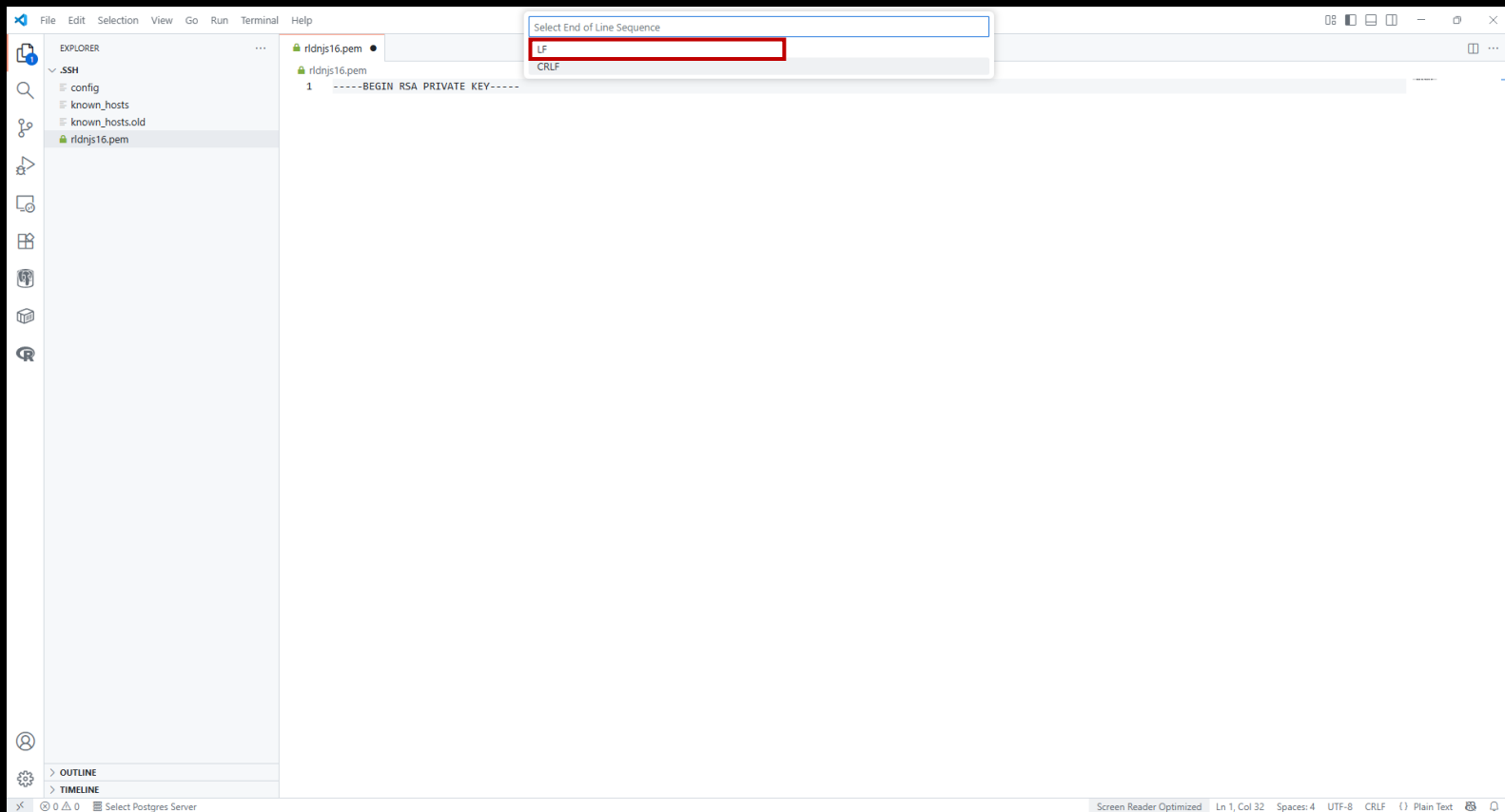
- 파일 포맷 확인
  - 파일 포맷 확인 후 클릭



## 02 UBAI 서버 접속

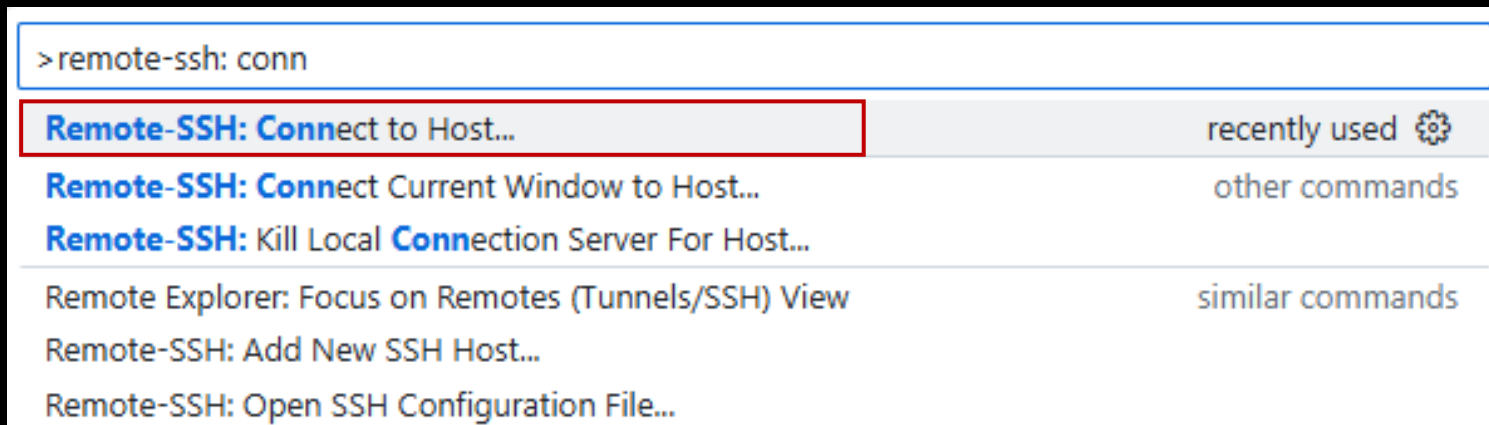
### 접속 방법(Only Mac User)

- 파일 포맷 변경
  - 파일 포맷 LF로 변경 후 저장



### 접속 방법

- SSH 접속
  - Ctrl + Shift + P
  - > Remote-SSH: Connect to Host...

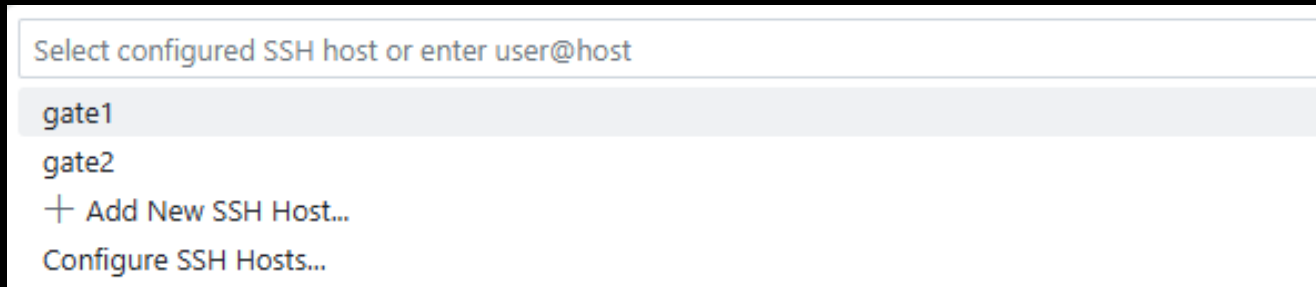




## 02 UBAI 서버 접속

### 접속 방법

- SSH 접속
  - gate1 또는 gate2로 접속



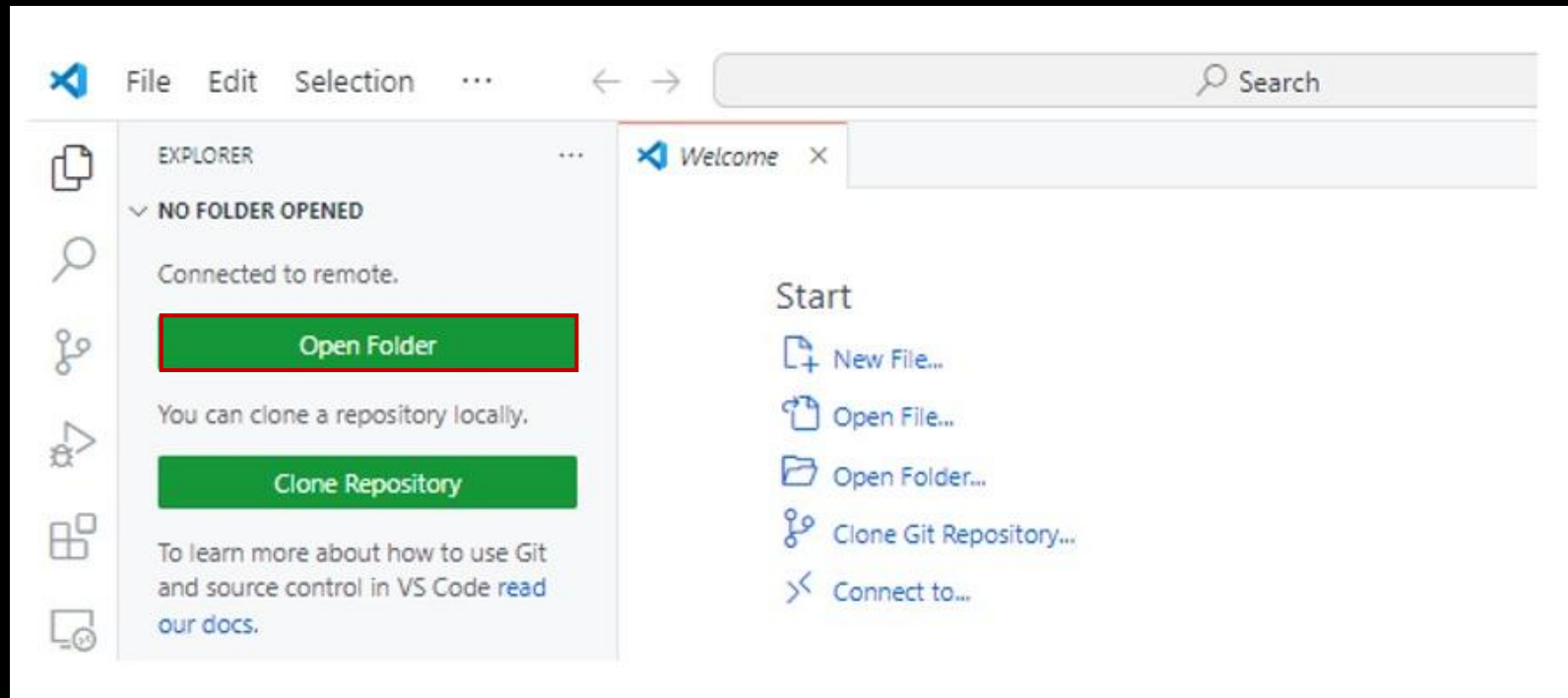
- 운영체제 반드시 Linux로 선택(★ ★ ★)
- Continue 선택
- Gate 접속 확인 (왼쪽 하단)



## 02 UBAI 서버 접속

### 접속 방법

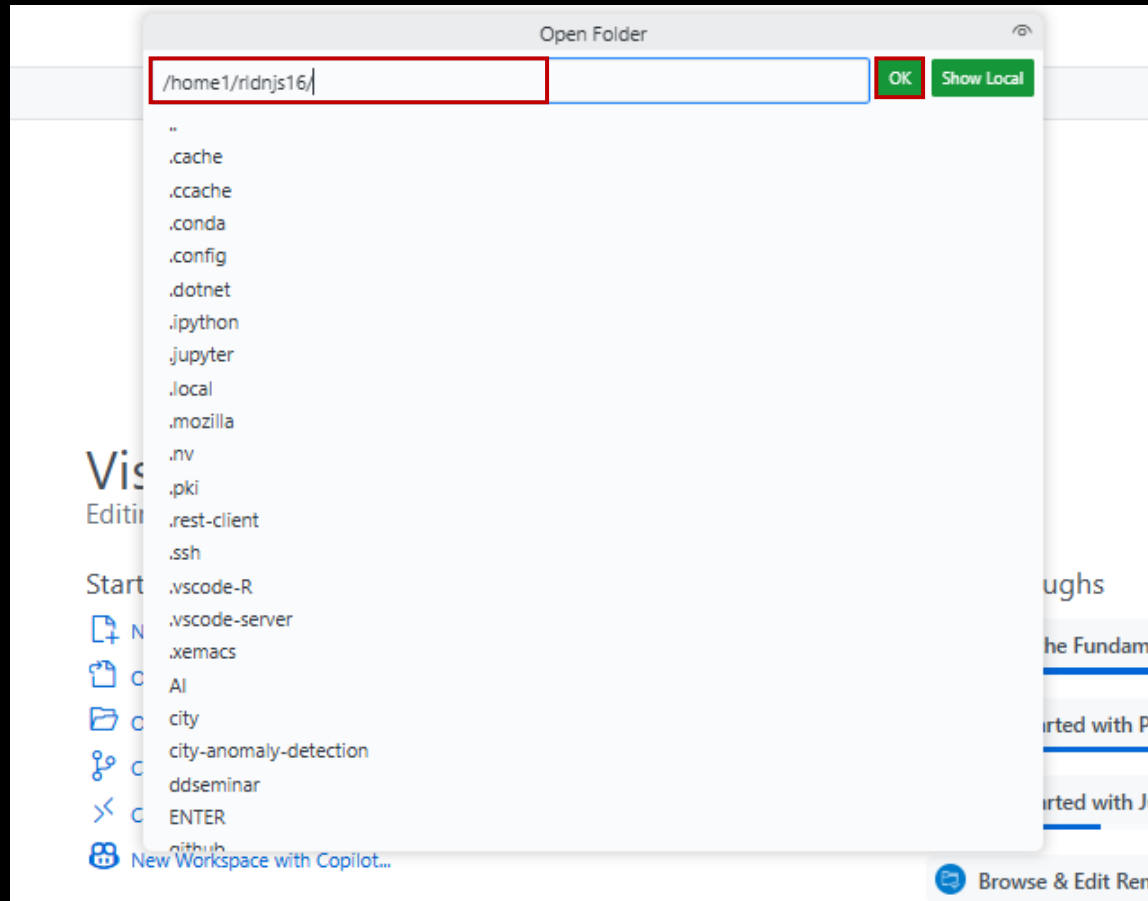
- 디렉토리 접속



## 02 UBAI 서버 접속

### 접속 방법

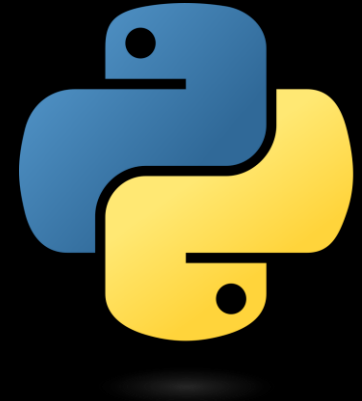
- 디렉토리 접속
  - 사용자(계정명) 폴더로 접속합니다.



## 03 Python 환경 구축

### Python

- Python은 웹 어플리케이션, 소프트웨어 개발, 데이터 사이언스, 머신러닝, 딥러닝에 널리 사용되는 프로그래밍 언어입니다.
- 오픈소스 환경을 가지고 있으며, 많은 사람들이 이용하는 언어입니다.
- Python을 활용하기 위해서는 보통 Anaconda를 활용하지만 리눅스 컴퓨팅 환경에서는 Miniconda를 활용합니다.
- UBAI 슈퍼컴퓨터에서도 Python을 이용하기 위해서는 Miniconda를 사용해야 합니다.



## 03 Python 환경 구축

### Miniconda

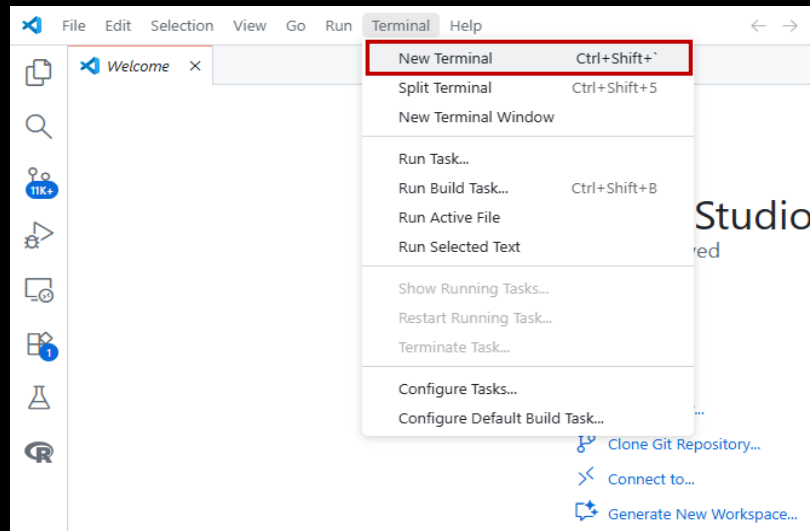
- Anaconda는 머신러닝이나 데이터 분석 등에 사용하는 여러가지 패키지가 기본적으로 포함되어있는 파이썬 배포판입니다.
- Python의 가상환경을 구축하는데 매우 유용하게 사용됩니다.
- Miniconda는 Anaconda의 경량 버전이라고 볼 수 있습니다.



# 03 Python 환경 구축

## Miniconda 설치

- Terminal 접속



## 03 Python 환경 구축

### Miniconda 설치

- Miniconda 설치 파일 다운로드
  - 앞서 github에서 다운로드한 **edu.txt**에 저장된 명령어를 복사 후 붙여 넣어줍니다.
  - **wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
○ (base) [rldnjs16@gate1 ~]$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

- Bash 명령어를 통해 Miniconda 설치
  - **bash Miniconda3-latest-Linux-x86\_64.sh**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
○ (base) [rldnjs16@gate1 ~]$ bash Miniconda3-latest-Linux-x86_64.sh
```

- 라이선스 동의에 대한 내용이 모두 나타날 때까지 **Enter**를 꼭 눌러줍니다.
- 그 후 라이선스 동의 확인에 대한 질문에 **yes**라고 입력해줍니다.
- Enter를 너무 오래 눌러 동의 **화면이 바로 넘어가지 않게** 주의해주세요.

### Miniconda 설치

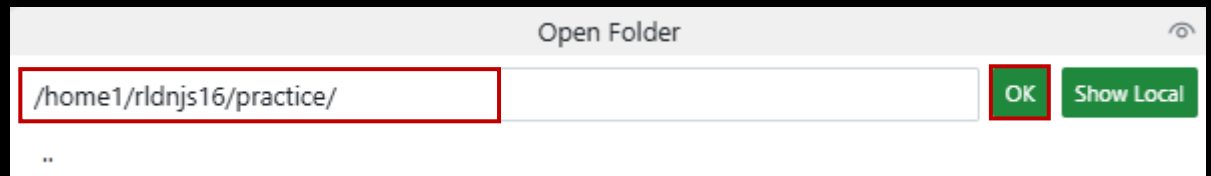
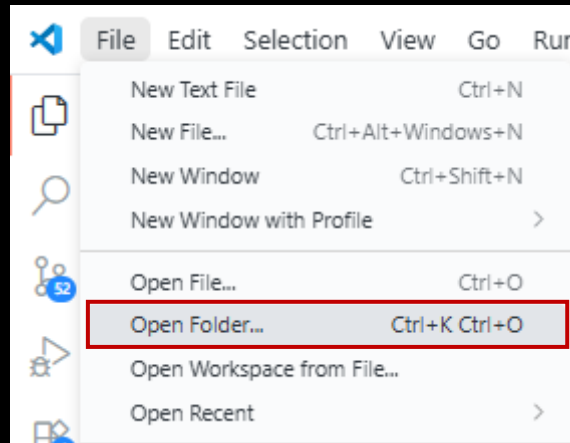
- Bash 명령어를 통해 Miniconda 설치
  1. 화면에서 해당 경로가 자신의 서버에 있는 **경로**와 맞는지 확인 후 **Enter**를 눌러줍니다.
  2. 처음 접속 시, **conda init** 진행 선택에 대한 질문이 나타납니다.  
**Yes**를 입력하신 후 enter를 눌러 주시면 됩니다.
  3. 변경사항 반영을 위해 지금 작업한 터미널을 닫고 **새로운 터미널로 접속**해주세요.
  4. 재접속 시, termina에 **(base)[사용자ID@gate1/2]**가 보인다면  
성공적으로 설치가 완료된 것입니다.
  5. 설치 후 왼쪽의 탐색기(Explorer) 목록에 **miniconda 폴더**가 있는지 꼭 확인해주세요.



## 03 Python 환경 구축

### Miniconda 실행

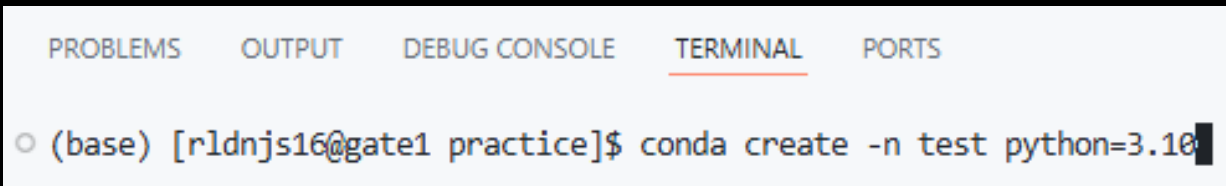
- 사용자 계정 내 임의의 폴더를 생성합니다
  - Ex) practice
- 생성한 폴더로 이동합니다.
  - /home1/사용자 계정/폴더 이름/



## 03 Python 환경 구축

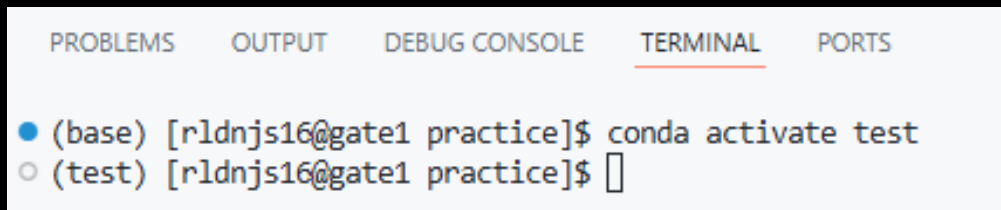
### Miniconda 실행

- 가상 환경 생성
  - `conda create -n {가상환경_이름} python={설치할_python_version}`
  - Ex) 가상환경 이름을 `test`로 설정하겠습니다.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ (base) [rldnjs16@gate1 practice]$ conda create -n test python=3.10
```

- Proceed ([y]/n)? y
- 가상 환경 실행
  - `conda activate {가상환경 이름}`

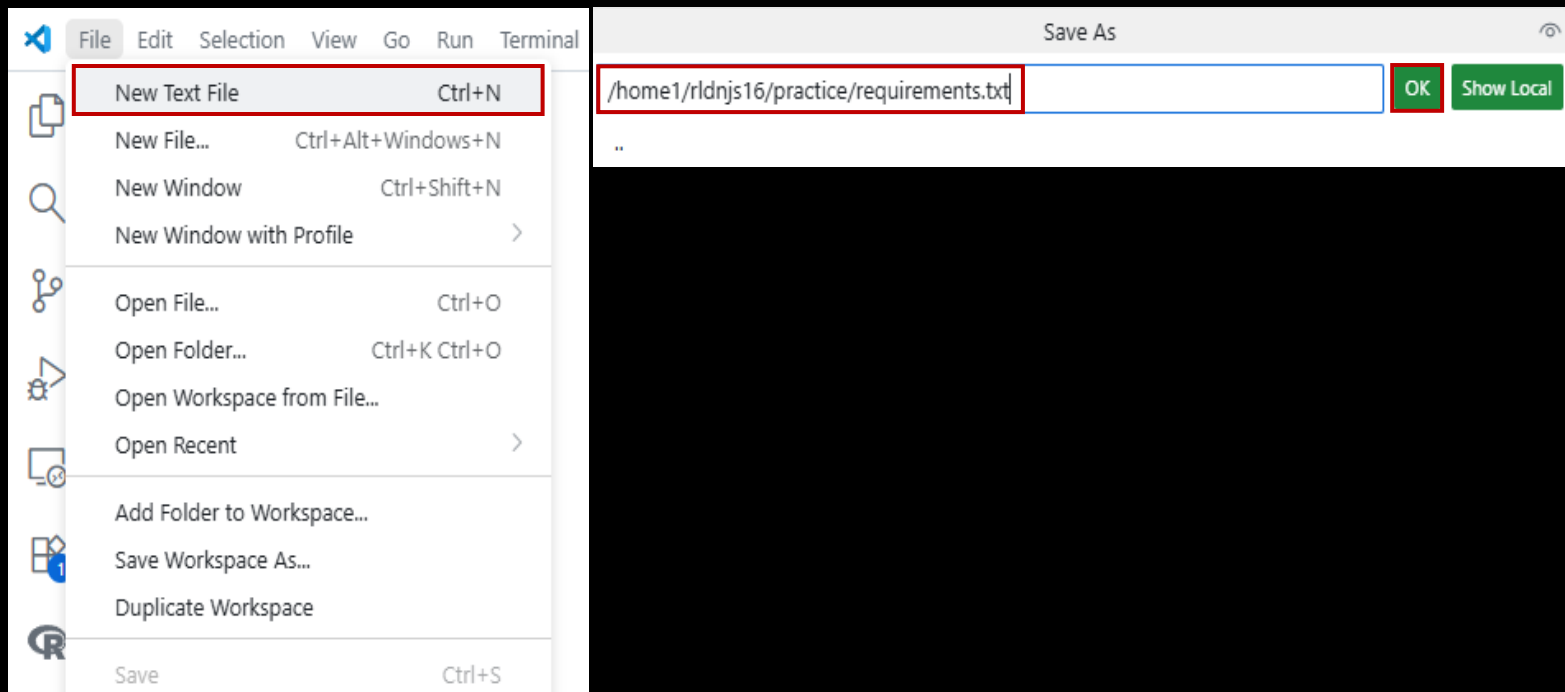


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● (base) [rldnjs16@gate1 practice]$ conda activate test
○ (test) [rldnjs16@gate1 practice]$
```

## 03 Python 환경 구축

### Miniconda 실행

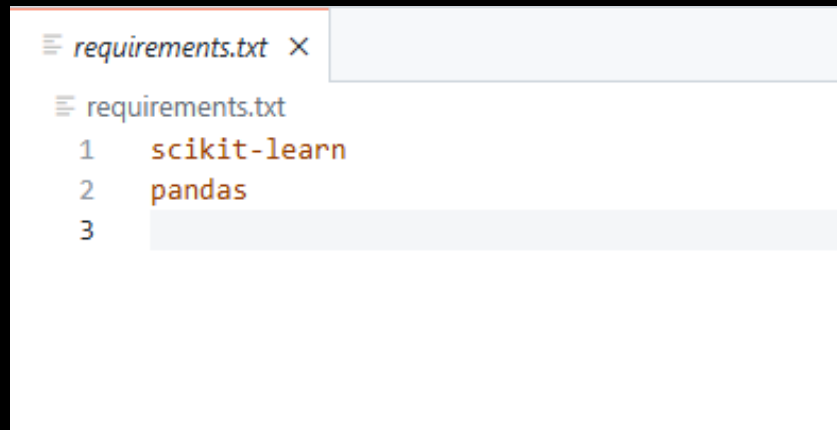
- requirements.txt
  - 앞서 github에서 다운로드한 requirements.txt의 내용을 복사(또는 다운로드)한 후, 새 파일에 붙여넣고 **requirements.txt**라는 이름으로 저장해 주세요.



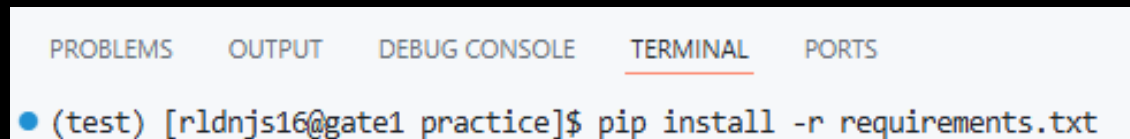
## 03 Python 환경 구축

### Miniconda 실행

- requirements.txt
  - terminal에서 가상환경 접속 후 `pip install -r requirements.txt`를 입력합니다.
  - 간단한 실습을 위해, 필요한 패키지들을 한 번에 설치합니다.
  - 이후 가상환경에서 별도 설치 없이 패키지를 사용할 수 있습니다.

A screenshot of a code editor window. The title bar shows 'requirements.txt' with a close button. The editor content shows a file named 'requirements.txt' with three lines of code: '1 scikit-learn', '2 pandas', and '3' followed by a blank line.

```
requirements.txt
1  scikit-learn
2  pandas
3
```

A screenshot of a terminal window. The tabs at the top are 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal shows a command prompt for a virtual environment named '(test)' and the command to install packages from requirements.txt.

```
(test) [rldnjs16@gate1 practice]$ pip install -r requirements.txt
```



**Break Time**

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### Slurm

- 도시과학빅데이터·AI연구원은 사용자에게 고성능컴퓨팅(HPC) 자원을 제공합니다.
- 사용자는 Slurm을 통해 독점적인 자원을 할당받고  
AI학습 및 추론, 연산, 시뮬레이션 등의 작업(Job)을 수행할 수 있습니다.
- Slurm은 다양한 사용자들의 다양한 요구를 수용하고(Job Submit),  
각 사용자들의 작업을 스케줄링하며(Task Scheduling),  
자원을 관리(Resource Management)하는 Linux 유틸리티 입니다.
- 여러 명의 사용자가 UBAI 클러스터를 이용하는데 있어서  
원활한 실험 및 계산을 위한 중재자 역할을 수행합니다.
- UBAI는 여러 명의 사용자에게 UBAI Cluster를 제공하기 위해 Slurm을 사용합니다.

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### Cluster

- 여러 대의 컴퓨터들이 연결되어 하나의 시스템처럼 동작하는 컴퓨터들의 집합을 말합니다.

### Partition

- Partition은 특정 자원 그룹을 정의하는 논리적 단위입니다.
- 사용자들이 작업을 제출할 때 특정 Partition을 지정하여 자원을 할당받을 수 있습니다.

Partition	# of Nodes	# of Cores /node	CPU	GPU/node	Memory/node
gpu1	14(n001-n014)	48	Intel Xeon Gold 6240R	RTX3090(4EA)	768GB
gpu6	25(n015-n039)	48	Intel Xeon Gold 6240R	A10(4EA)	768GB
cpu1	10(n040-n049)	48	Intel Xeon Gold 6240R	-	768GB
gpu2	11(n051-n061)	56	Intel Xeon Gold 6348R	A10(4EA)	1024GB
gpu3	10(n062-n071)	56	Intel Xeon Gold 6348R	A6000ada(4EA)	1024GB
gpu4	29(n072-n100)	56	Intel Xeon Gold 6348R	A6000(4EA)	1024GB
Gpu5	6(n101-n106)	64	Intel Xeon Platinum-8358	A6000(4EA)	1024GB

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### Node

- 클러스터를 구성하는 개별 컴퓨터(서버)를 의미합니다.
- 각 Node는 CPU, 메모리, GPU, 디스크 등의 자원을 갖습니다.

### Job

- 컴퓨터 클러스터와 같은 경우  
스케줄러(이하 Slurm)를 사용하여 컴퓨터 자원에 접근하기 위한 요청을 합니다.
- 이를 작업(Job)이라고 부르며,  
이는 사용자가 원하는 프로그램이나 스크립트를 실행하고 싶을 때의 행동을 지칭하기도 합니다.



## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

The screenshot shows the GitHub interface for a repository named 'practice\_november'. The repository is private and has 1 branch (main) and 0 tags. The commit history shows a series of updates to various files. The file 'run.sh' is highlighted with a red box, indicating it is the focus of the 'Batch File Creation' section.

File	Commit Message	Time
README.md	Update README.md	last week
edu.txt	Update edu.txt	20 hours ago
multi_run_files.zip	Add files via upload	17 hours ago
predict.py	Rename test.py to predict.py	3 weeks ago
requirements.txt	Create requirements.txt	last month
run.sh	Update run.sh	last week
슈퍼컴퓨터 사용법 교육.pdf	Add files via upload	16 hours ago

# 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

## 배치 파일 생성

- 배치 파일을 만들어주기 위해, 파일을 전부 복사(또는 다운로드)해줍니다.

practice\_november / run.sh

ki5n2 Update run.sh

Code Blame 26 lines (21 loc) · 1.35 KB

```

1  #!/bin/bash
2  #SBATCH --job-name=MY_JOB          # 사용자의 작업 이름으로 변경
3  #SBATCH --output=./output/%j.out
4  #SBATCH --error=./output/%j.err
5  #SBATCH --partition=gpu0          # 사용할 파티션 이름으로 변경 필요 -> gpu number(gpu1 ~ gpu6)
6  #SBATCH --nodelist=n107          # 사용할 노드 이름으로 변경 -> node number(n001 ~ n106)
7  #SBATCH --gres=gpu:1             # 사용할 gpu 수
8  #SBATCH --cpus-per-task=1        # 하나의 태스크가 사용할 CPU 코어 수
9  ##SBATCH --mem=128G              # 메모리 할당량 (##이므로 해당 명령어 비활성화)
10 ##SBATCH --time=48:00:00         # 최대 실행 시간 (##이므로 해당 명령어 비활성화)
11
12 echo "start at: $(date)"          # 접속한 날짜 표기
13 echo "node: $HOSTNAME"           # 접속한 노드 번호 표기
14 echo "jobid: $SLURM_JOB_ID"      # jobid 표기
15
16 # Load modules (cuda 환경)
17 module load cuda/11.8.0
18
19 # Load env (python 환경)
20 source ~/miniconda3/etc/profile.d/conda.sh
21
22 # 가상환경 활성화 (설치한 가상환경 이름으로 변경 필요, test -> 가상환경 이름)
23 conda activate test              # test라는 conda 환경에서 슈퍼컴퓨팅 을 준비 완료
24
25 # python 스크립트 실행
26 python predict.py
  
```

그림 이모티콘

잘라내기

**복사**

붙여넣기

일반 텍스트로 붙여넣기

전체 선택

"#!/bin/bash #SBATCH --job-name=MY\_JOB..." 검색

인쇄 Ctrl+P

맞춤법 검사

쓰기 방향

음성으로 듣기

쿼리치

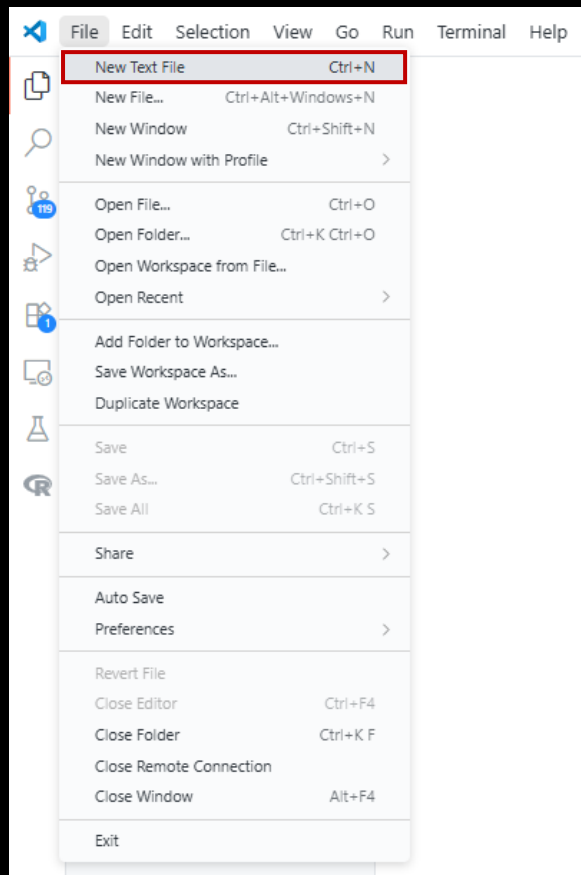
스크립트에 추가

검사

# 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

## 배치 파일 생성

- 배치 파일 생성을 위해 새로운 텍스트 파일을 만들어줍니다.



# 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

## 배치 파일 생성

- 파일명.sh로 저장해줍니다.(예시, run.sh)

```

1  #!/bin/bash
2  #SBATCH --job-name=MY_JOB          # 사용자의 작업 이름으로 변경
3  #SBATCH --output=./output/%j.out
4  #SBATCH --error=./output/%j.err
5  #SBATCH --partition=gpu0          # 사용할 파티션 이름으로 변경 필요 -> gpu number(gpu1 ~ gpu6)
6  #SBATCH --odelist=n107           # 사용할 노드 이름으로 변경-> node number(n001 ~ n106)
7  #SBATCH --gres=gpu:1             # 사용할 gpu 수
8  #SBATCH --cpus-per-task=1        # 하나의 태스크가 사용할 CPU 코어 수
9  ##SBATCH --mem=128G              # 메모리 할당량 (##이므로 해당 명령어 비활성화)
10 ##SBATCH --time=48:00:00         # 최대 실행 시간 (##이므로 해당 명령어 비활성화)
11
12 echo "start at: $(date)"          # 접속한 날짜 표기
13 echo "node: $HOSTNAME"           # 접속한 노드 번호 표기
14 echo "jobid: $SLURM_JOB_ID"      # jobid 표기
15
16 # Load modules (cuda 환경)
17 module load cuda/11.8.0
18
19 # Load env (python 환경)
20 source ~/miniconda3/etc/profile.d/conda.sh
21
22 # 가상환경 활성화 (설치한 가상환경 이름으로 변경 필요, test -> 가상환경 이름)
23 conda activate test              # test라는 conda 환경에서 슈퍼컴퓨팅 쓸 준비 완료
24
25 # python 스크립트 실행
26 python predict.py
  
```

Save As dialog box content:

```

/home1/rldnjs16/practice/run.sh
..
requirements.txt
  
```

# 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

## 배치 파일 생성

- 프로젝트 수행을 위한 배치 파일을 생성합니다.
- 다음의 사항을 본인의 작업(job)에 맞게 입력한 후, 본인이 원하는 파일의 이름을 지정하여 filename.sh 형식으로 파일을 저장합니다.
- 예를 들어, run.sh로 저장해보겠습니다.
- 파일명이 .sh 형식인지 반드시 확인해주세요.

```
$ run.sh
$ run.sh
1  #!/bin/bash
2  #SBATCH --job-name=MY_JOB           # 사용자의 작업 이름으로 변경
3  #SBATCH --output=./output/%j.out
4  #SBATCH --error=./output/%j.err
5  #SBATCH --partition=gpu0            # 사용할 파티션 이름으로 변경 필요 -> gpu number(gpu1 ~ gpu6)
6  #SBATCH --nodelist=n107             # 사용할 노드 이름으로 변경-> node number(n001 ~ n106)
7  #SBATCH --gres=gpu:1                # 사용할 gpu 수
8  #SBATCH --cpus-per-task=1           # 하나의 태스크가 사용할 CPU 코어 수
9  ##SBATCH --mem=128G                 # 메모리 할당량 (##이므로 해당 명령어 비활성화)
10 ##SBATCH --time=48:00:00           # 최대 실행 시간 (##이므로 해당 명령어 비활성화)
11
12 echo "start at: $(date)"            # 접속한 날짜 표기
13 echo "node: $HOSTNAME"              # 접속한 노드 번호 표기
14 echo "jobid: $SLURM_JOB_ID"         # jobid 표기
15
16 # Load modules (cuda 환경)
17 module load cuda/11.8.0
18
19 # Load env (python 환경)
20 source ~/miniconda3/etc/profile.d/conda.sh
21
22 # 가상환경 활성화 (설치한 가상환경 이름으로 변경 필요, test -> 가상환경 이름)
23 conda activate test                 # test라는 conda 환경에서 슈퍼컴퓨팅 쓸 준비 완료
24
25 # python 스크립트 실행
26 python predict.py
```

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

**#SBATCH --job-name=JOB\_NAME**

- 작업 이름을 지정하는 명령어입니다.
- Ex) #SBATCH --job-name=MY\_JOB

**#SBATCH --output=./output/%j.out**

- 작업 결과 파일의 저장 위치를 지정하는 명령어입니다.
- Ex) #SBATCH --output=./output/%j.out

**#SBATCH --error=./output/%j.err**

- 작업 중 발생하는 오류 메시지의 파일 저장 위치를 지정하는 명령어입니다.
- Ex) #SBATCH --error=./output/%j.err

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

#### #SBATCH --partition=gpu\_n

- 사용할 Partition을 지정하는 명령어입니다.
- 지정된 노드가 속한 파티션을 입력해줍니다.
- 사용할 노드 대응되는 Partition 번호로 변경해줍니다. (gpu1 ~ gpu6, 27page 참고)

#### #SBATCH --nodelist=n107

- 사용할 노드를 지정하는 명령어입니다.
- 생략 시 해당 파티션 내 자동 배정
- 사용하고자 하는 노드 번호로 변경해줍니다. (n001 ~ n106, 27page 참고)
- 지정한 노드 번호에 맞추어 Partition 번호도 변경해줍니다.
- 노드 번호와 Partition 번호가 올바르게 대응되지 않은 경우, 다음과 같은 에러가 발생합니다.

```
⊗ (test) [rldnjs16@gate1 practice]$ sbatch run.sh
sbatch: error: Batch job submission failed: Invalid node name specified
```

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

#### #SBATCH --gres=gpu:1

- 몇 개의 GPU를 사용할 것인지 지정하는 명령어입니다.
- 각 노드는 4개의 GPU를 가지고 있으며, 해당 옵션을 사용하면 해당 노드의 GPU 중 1개를 현재 작업에 할당함을 의미합니다.

#### #SBATCH --cpus-per-task=1

- 총 필요한 CPU 코어의 개수를 지정하는 명령어입니다.
- 각 파티션마다 CPU 코어의 개수가 다릅니다.
- 사용자의 작업에 따라 요구하는 CPU 자원이 다르므로, 작업에 맞게 코어 개수를 설정합니다.



## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

#### #SBATCH --mem=128G

- 작업을 실행할 때 필요한 메모리를 요청합니다.
- 요청한 메모리 이상을 초과해서 사용하려고 한다면, 해당 작업은 자동으로 중단됩니다.
- 따라서 메모리를 충분히 설정하거나, “##SBATCH --mem=128G”을 통해 기본값(메모리 제한 없음)으로 실행합니다.
- 그럼에도, 각 노드가 갖는 메모리(1024G)를 초과하여 사용할 경우, 해당 작업은 중단됩니다.

#### #SBATCH --time=24:00:00

- 최대 실행 시간을 설정할 수 있습니다.
- 작업에 필요한 시간을 알기 어려우므로, “##SBATCH --time=24:00:00”을 통해 기본값(48시간, 최대)으로 실행합니다.

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

```
echo "start at: $(date)"
```

- 접속 날짜가 표기됩니다.

```
echo "node: $HOSTNAME"
```

- 접속한 노드 번호가 표기됩니다.

```
echo "jobid: $SLURM_JOB_ID"
```

- jobid가 표기됩니다.

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

#### `module ~`

- 원하는 Linux 환경을 구축할 수 있고, 기본적으로 CUDA/11.2.2 실행으로 셋팅되어 있습니다.
- 지금과 같이 다른 GPU 환경을 원할 경우, 원하는 모듈을 load 합니다.
- GPU 환경을 사용하고 싶은 경우에만 해당하며, GPU 환경을 사용하지 않을 경우(CPU Partition 사용) 지우셔도 무관합니다.

#### `source ~/miniconda3/etc/profile.d/conda.sh`

- conda 환경을 사용하기 위한 초기 설정을 불러오는 작업입니다.
- 해당 명령을 통해 `conda activate` 등 conda 관련 명령어들을 사용할 수 있습니다.
- Slurm 같은 비인터랙티브 셸은 보통 `~/.bashrc`를 안 읽어 conda가 인식되지 않으므로, 스크립트(.sh 파일)에 `source ~/miniconda3/etc/profile.d/conda.sh`로 conda 함수를 로드해야 합니다.

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 생성

#### `conda activate test`

- 본인이 생성한 가상환경을 활성화시켜,  
해당 환경에서 설치된 패키지들을 사용하여 작업을 진행할 수 있습니다.

#### `python predict.py`

- 원하는 python 파일을 실행합니다.
- 실행하려는 파일은 .py 파일의 형태로 존재해야 합니다.
- Ex) `predict.py`

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 실행

- Terminal에 **sbatch** 명령어를 이용하여 **배치 파일명**을 입력 및 실행하세요.
- 다음과 같이 배치 파일을 실행합니다.
  - 작업 제출 전 **ls -al** 명령어를 통해 현재 작업 디렉토리에 **run.sh** 파일이 존재하는지 확인합니다.
  - 존재하면, **sbatch filename.sh**를 통해 작업을 제출합니다.
  - Ex) **sbatch run.sh**
- 이는 할당받은 노드에 작업(job)을 제출한다는 의미입니다.
  - 작업(job) 제출이 정상적으로 진행되었다면, **output 폴더** 안에 해당 작업에 대한 **err/out 파일**이 각각 생성됩니다.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● (test) [rldnjs16@gate1 practice]$ sbatch run.sh
Submitted batch job 502245
```

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 배치 파일 실행

- 작업 상태 확인
  - 만일 해당 파일이 생성되지 않았다면, 해당 노드가 이미 제출한 다른 작업(job)으로 모두 할당되어 작업 대기 중일 가능성이 큽니다.
    - terminal에 **qstat** 명령어를 입력하여, 본인의 ID를 확인해 작업 현황을 점검하시기 바랍니다.
  - 배정이 되어 실행되고 있다면, Use 란에 **R**이라고 표시되어 있지만, 작업을 대기하고 있는 경우 **Q**라고 표시되어 있습니다.
  - 작업이 대기(**Q**)에 걸린 경우, 앞선 작업들이 끝나는 것을 기다려야 합니다.

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 제출 작업 확인

- qstat: 모든 제출된 작업 상태를 확인
  - 사용시 특정 사용자가 제출한 작업만을 확인하기 위해서는 -u 옵션이 필요합니다.
  - `qstat -u <user_name>`
  - Ex) `qstat -u rldnjs16`
- Use - Q: 작업 대기 중

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• (test) [rldnjs16@gate1 practice]$ qstat -u rldnjs16

gate1.hpc:

Job id                Username Queue    Name                SessID NDS   TSK      Req'd  Req'd   Elap
-----
502245                rldnjs16 gpu3     MY_JOB              --      1    1         --  48:00 Q 00:00

```

# 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

## 제출 작업 확인

- Use – R: 작업 진행 중

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• (test) [rldnjs16@gate1 practice]$ qstat -u rldnjs16

gate1.hpc:

Job id                Username Queue    Name                SessID NDS    TSK      Req'd  Req'd   Elap
-----
502245                rldnjs16 gpu3    MY_JOB              --      1      1       --  48:00 R  00:00
  
```

- Use – C: 작업 종료

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• (test) [rldnjs16@gate1 practice]$ qstat -u rldnjs16

gate1.hpc:

Job id                Username Queue    Name                SessID NDS    TSK      Req'd  Req'd   Elap
-----
502245                rldnjs16 gpu3    MY_JOB              --      1      1       --  48:00 C  00:00
  
```



## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 제출 작업 취소

- 제출한 작업을 취소하기 위해서는 `scancel` 명령어를 사용하세요.
  - `scancel <job_id>` : 특정 작업만 취소합니다.
  - Ex) `scancel 502304`

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• (test) [rldnjs16@gate1 practice]$ qstat -u rldnjs16

gate1.hpc:

Job id          Username Queue   Name              SessID NDS    TSK    Req'd  Req'd  Elap
-----
502304          rldnjs16 gpu1    MY_JOB            --      1      1      --  48:00 C 00:00
  
```

- `scancel -u <user_name>` : 자신이 제출한 모든 작업을 취소합니다.
  - Ex) `scancel -u rldnjs16`
- Use에 **C**가 나타났다면, 정상적으로 작업이 취소된 것입니다.
- (Tip) Use에 **C**가 나타난 경우 1. 작업 수행 종료 2. (사용자 요청) 작업 취소로 인한 종료

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 기타 명령어

- `pestat`: 각 노드의 점유 상태 확인
- `pestat -G`: 각 노드의 점유 상태를 GPU 중심으로 확인
- Node State
  - `idle`: 현재 어떤 작업도 수행하지 않는 상태로, 해당 노드를 자유롭게 활용할 수 있습니다.
  - `Mix`: 일부 자원만 사용 중인 상태로, CPU 또는 GPU 자원이 남아 있습니다.
    - Case 1: CPU는 모두 점유되어 있고 GPU만 남아 있는 경우 → 추가 작업 불가
    - Case 2: GPU는 모두 점유되어 있고 CPU만 남아 있는 경우 → CPU만 사용하는 작업 가능 (GPU 사용 불가)
    - Case 3: CPU와 GPU가 모두 남아 있는 경우 → CPU 및 GPU 모두 활용 가능
    - 즉, 자신의 작업 특성에 맞는 자원이 남아 있는 노드를 선택하여 작업을 제출해야 합니다.
  - `alloc`: CPU와 GPU 자원이 모두 할당된 상태로, 이 노드에 작업을 제출하면 대기열(Q)에 등록됩니다.
  - `resv / drain`: 예약 또는 오류 상태로, 해당 노드는 작업 제출 대상에서 제외해야 합니다.

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 기타 명령어

- Num Use/CPU Tot
  - Num Use: 현재 사용 중인 CPU 코어 수
  - CPU Tot: 해당 노드의 전체 CPU 코어 수
  - Ex) 0 / 48 → 전체 48개 코어 중 사용 중인 코어가 없음 (모두 여유 상태)
  - Ex) 24 / 48 → 전체 48개 코어 중 24개 코어가 사용 중 (24 코어 이하 작업 제출 시 작업 수행 가능)
  - Ex) 48 / 48 → 전체 48개 코어가 모두 사용 중 (작업 제출 시 대기)
- Joblist (JobID User GRES/job ...)
  - JobID: 작업 고유 번호
  - User: 작업 실행 사용자
  - GRES/job ...: 사용 중인 GPU 개수
  - Ex) 500000 rldnjs16 gpu:rtx3090=1
  - Joblist 내 사용 중인 GPU 개수가 몇 개인지 확인 필요
    - 4개 미만이면, 남은 GPU 개수만큼 GPU 사용 가능

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### 기타 명령어

- CPU / GPU 설정 안내
  - 작업을 처음 수행할 경우, CPU 코어 개수와 GPU 개수를 각각 1로 설정하여 시작
  - 이후 CPU 연산량이 많거나 GPU 연산이 집중적으로 필요한 경우, 자원 사용량을 점진적으로 늘려가며 실험을 진행
  - 매 실행 시 현재 사용 중인 CPU 코어 수 및 GPU 개수를 확인하여, 자원 사용이 적절한지 점검
  - 과도한 자원 점유를 피하고, 다른 사용자의 작업에 영향을 주지 않도록 주의
  - 자원을 과도하게 점유하거나 비효율적으로 사용하는 경우, 관리자가 모니터링 후 제재 조치가 이루어질 수 있음

## 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

### Python 프로젝트 생성

- 캘리포니아 집값 예측 테스트
  - [https://github.com/uos-ubai/practice\\_november/blob/main/predict.py](https://github.com/uos-ubai/practice_november/blob/main/predict.py)

The screenshot shows the GitHub interface for a repository named 'practice\_november' (Private). The repository has 1 branch (main) and 0 tags. A search bar 'Go to file' and buttons 'Add file' and 'Code' are visible. The commit history table lists several files and their updates:

File	Commit Message	Time
ki5n2 Delete jupyter.sh	ad753c1 · now	32 Commits
README.md	Update README.md	last week
edu.txt	Update edu.txt	20 hours ago
multi_run_files.zip	Add files via upload	17 hours ago
predict.py	Rename test.py to predict.py	3 weeks ago
requirements.txt	Create requirements.txt	last month
run.sh	Update run.sh	last week
슈퍼컴퓨터 사용법 교육.pdf	Add files via upload	16 hours ago

# 04 슈퍼컴퓨팅 환경 이해 및 Job 제출

## Python 프로젝트 생성

- 캘리포니아 집값 예측 테스트
  - pestat -G로 빈 노드를 확인한 뒤, 해당 노드·파티션에 맞게 run.sh를 수정하고 작업을 제출

practice\_november / predict.py

ki5n2 Rename test.py to predict.py 806816b · last week History

Code Blame 32 lines (26 loc) · 1.01 KB

```

1  from sklearn.datasets import fetch_california_housing
2  from sklearn.ensemble import RandomForestRegressor
3  from sklearn.model_selection import train_test_split
4  from sklearn.metrics import r2_score
5  import pandas as pd
6
7  # 1. 데이터 로딩 (sklearn 내장)
8  housing = fetch_california_housing()
9  X, y = housing.data, housing.target
10 print(f"데이터 로딩: {len(X):,}개 읽음")
11
12 # 2. 훈련/테스트 분할
13 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
14
15 # 3. 모델 학습
16 model = RandomForestRegressor(n_estimators=100, random_state=42)
17 model.fit(X_train, y_train)
18 print("모델 학습 완료")
19
20 # 4. 예측 및 평가
21 predictions = model.predict(X_test)
22 score = r2_score(y_test, predictions)
23 print(f"예측 정확도: {score:.3f} ({score*100:.1f}%)")
24
25 # 5. 예측 예시
26 print("\n예측 예시:")
27 for i in range(3):
28     actual = y_test[i]
29     pred = predictions[i]
30     print(f"실제: ${actual:.1f}만, 예측: ${pred:.1f}만 (오차: ${abs(actual-pred):.1f}만)")
31
32 print("분석 완료")

```

## 사사의 글 안내

도시과학빅데이터AI연구원의 슈퍼컴퓨터 자원을 이용하신 분들은 논문, 프로젝트, 통계 등 실적에 다음과 같이 연구원 사사를 필수적으로 적어주시기 바랍니다.

(국문) 본 논문은 서울시립대학교 도시과학빅데이터AI연구원의 슈퍼컴퓨팅 자원을 지원 받아 수행되었습니다.

(영문) The authors acknowledge the Urban Big data and AI Institute of the University of Seoul supercomputing resources (<http://ubai.uos.ac.kr>) made available for conducting the research reported in this paper.

**UBAI**





# Thank You

UBAI  
[UBAI@uos.ac.kr](mailto:UBAI@uos.ac.kr)