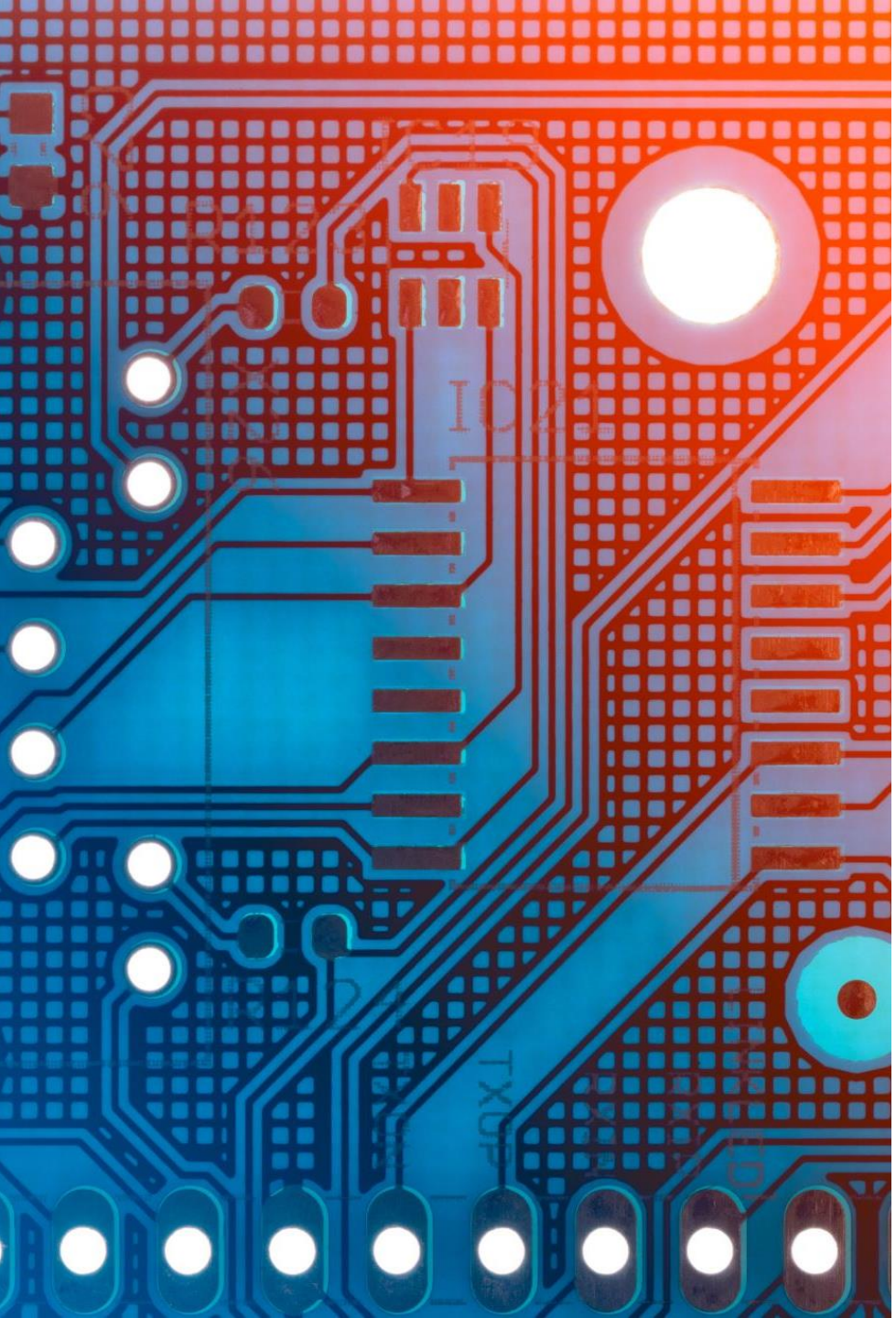
CPU-GPU CLUSTER 기초 사용법

도시과학빅데이터 AI연구원

2022.09



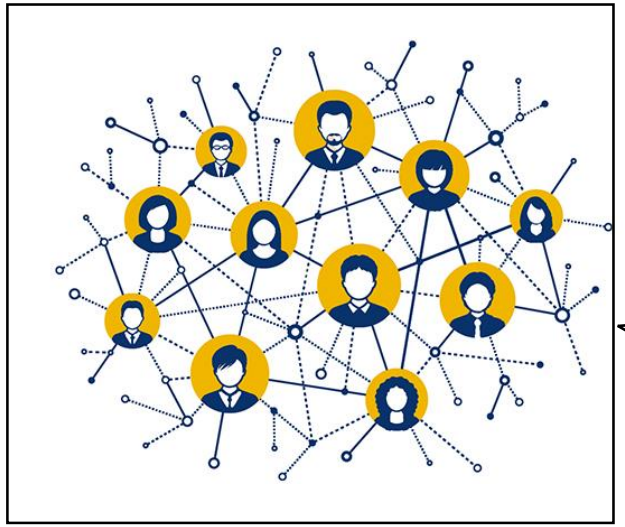
Agenda

1. CPU GPU Cluster 구조 및 사양
2. 외부 접속 방법
3. 기초 환경 호출 방법 (Linux Environment Modules)
4. SLURM Job Sscheduler 사용법
5. SLURM 예제
6. SLRUM job으로 jupyter lab 생성

1. CPU GPU Cluster 구조 및 사양

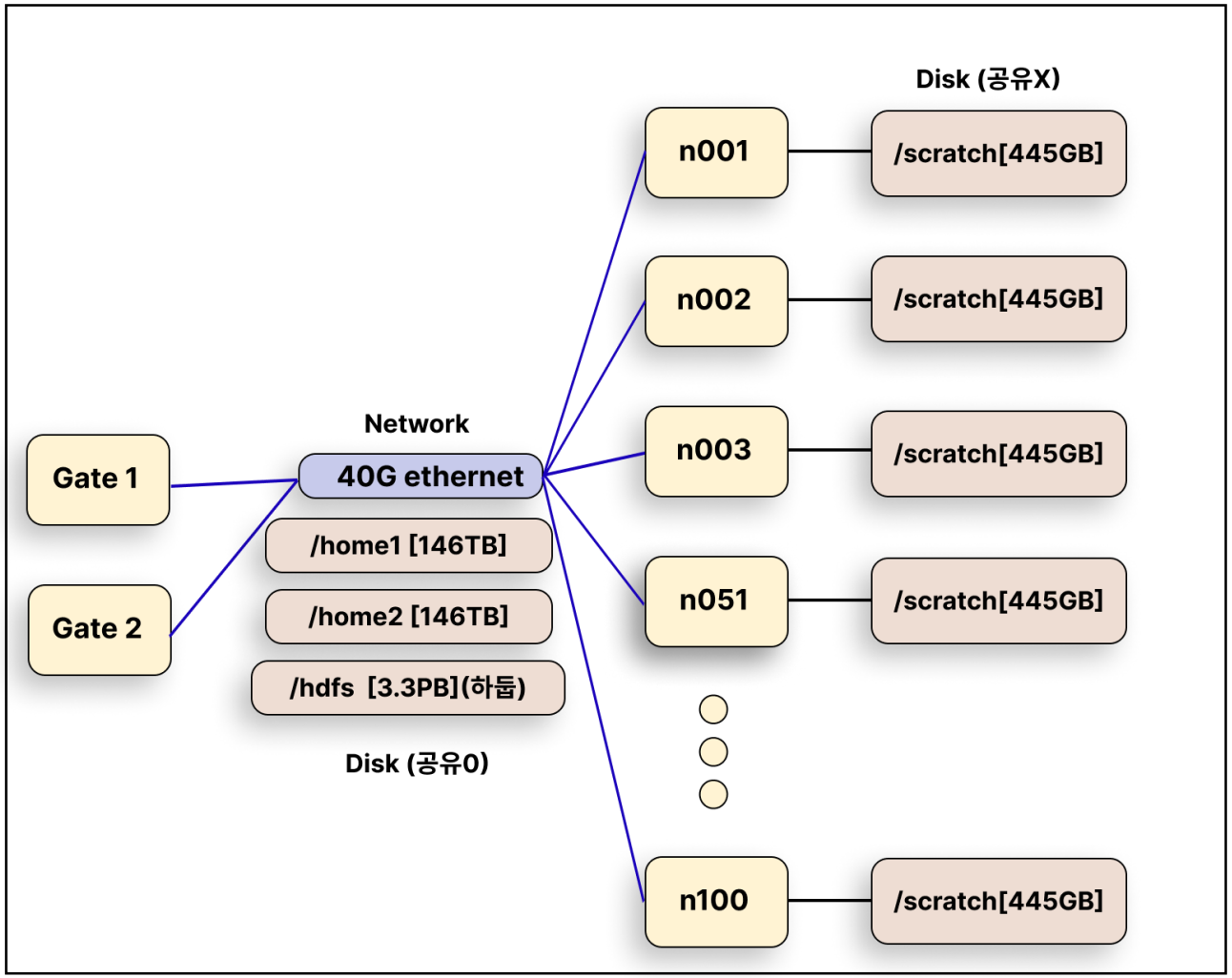
UBAI 네트워크 172.16.10.0/24

서울 시립대학교 내부 네트워크



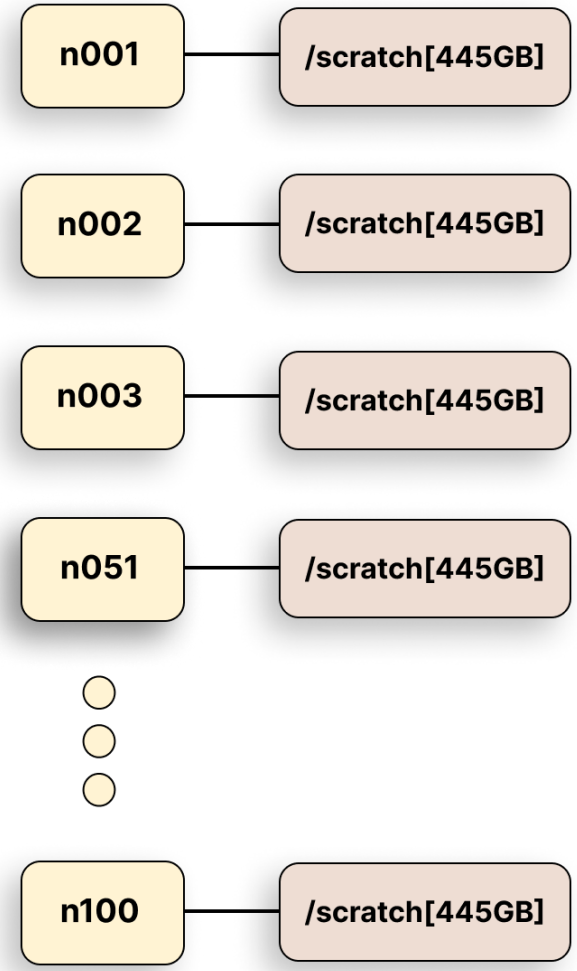
172.16.10.36:22

172.16.10.37:22



※ Member@UOS wifi 로는 접속 불가

1. CPU GPU Cluster 구조 및 사양



- 2022년 3월 31일 기준
- gpu1 partition (n001 ~ n013, 13대)
  - Intel Xeon Gold 6240R 2.4GHz 48 cores / 768GB MEMORY / SSD 480GB / Nvidia RTX3090 4EA
- cpu1 그룹 (n014 ~ n048, 35대)
  - Intel Xeon Gold 6240R 2.4GHz 48 cores / 768GB MEMORY / SSD 480GB
- n049 : 윈도우 서버 설치 / 독립 운영
  - Intel Xeon Gold 6240R 2.4GHz 48 cores / 768GB MEMORY / SSD 480GB / Nvidia RTX3090 1EA / 14TB HDD 8EA
- n001~n048중 n024, n048 노드 제외한 46노드 14TB HDD 8EA씩 장착 (모두 HADOOP에 동원됨)
- hgx 그룹 (n050)
  - Intel Xeon Gold 6248R 3.0GHz 48 cores / 1536GB MEMORY / SSD 2TB
- gpu2 그룹 (n051 ~ n086, 36대)
  - Intel Xeon Gold 6348R 2.6GHz 56 cores / 1024GB MEMORY / SSD 480GB / Nvidia A10 4EA
- cpu2 그룹 (n087 ~ n100, 14대)
  - Intel Xeon Gold 6348R 2.6GHz 56 cores / 1024GB MEMORY / SSD 480GB

2. 외부 접속방법

1. 기본 터미널 접속

- **Windows 10**
  - **Powershell**
- **Linux/MacOS**
  - **Bash, zsh**

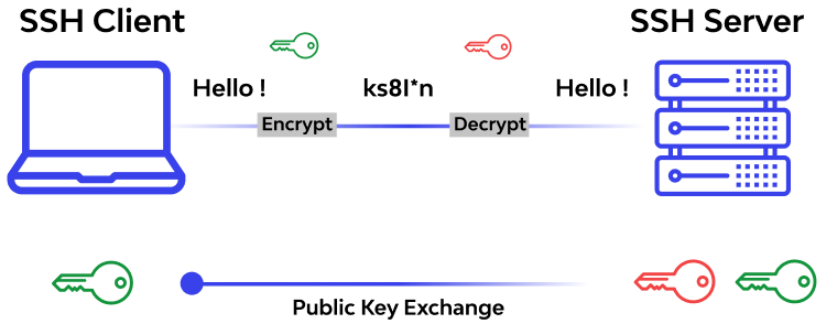
2. 상용 터미널 프로그램을 통한 접속

- **MobaXterm**
- **VScode**

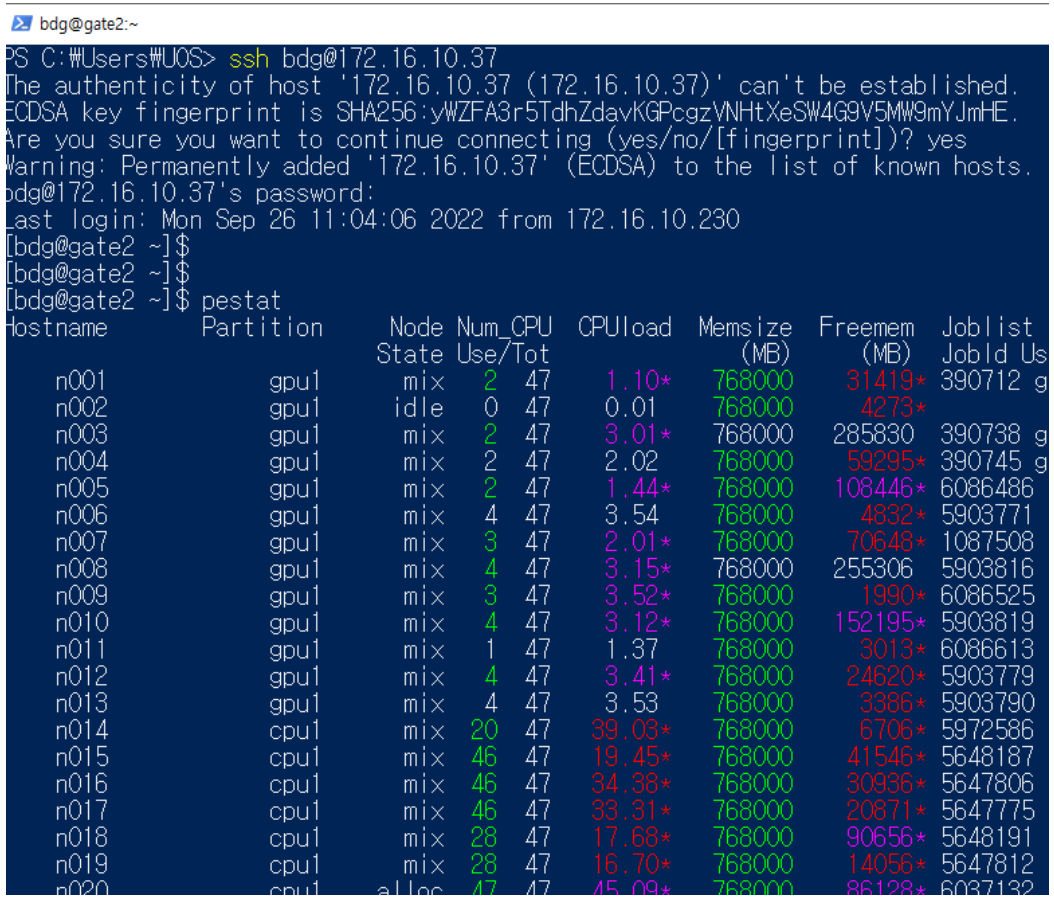
SSH Secure Shell Protocol

SSH 는 원격으로 다른 컴퓨터에 접속하여 command에 기반한 작업을 수행하도록 돕는 도구이다.

- Windows10, macOS, Ubuntu 등 다양한 운영체제들은 기본적인 ssh client 서버를 내장하고 있다.
- 상용 터미널 프로그램을 활용하면, 터미널 보다 편리한 작업이 가능하다.



2. 외부 접속방법



 # 기본 터미널

 # Windows Powershell 실행

1. ssh 접속

PS C:\Users\UOS> **ssh 사용자id@172.16.10.37 혹은 172.16.10.36**

... to continue connecting (yes/no/[fingerprint])? **yes**

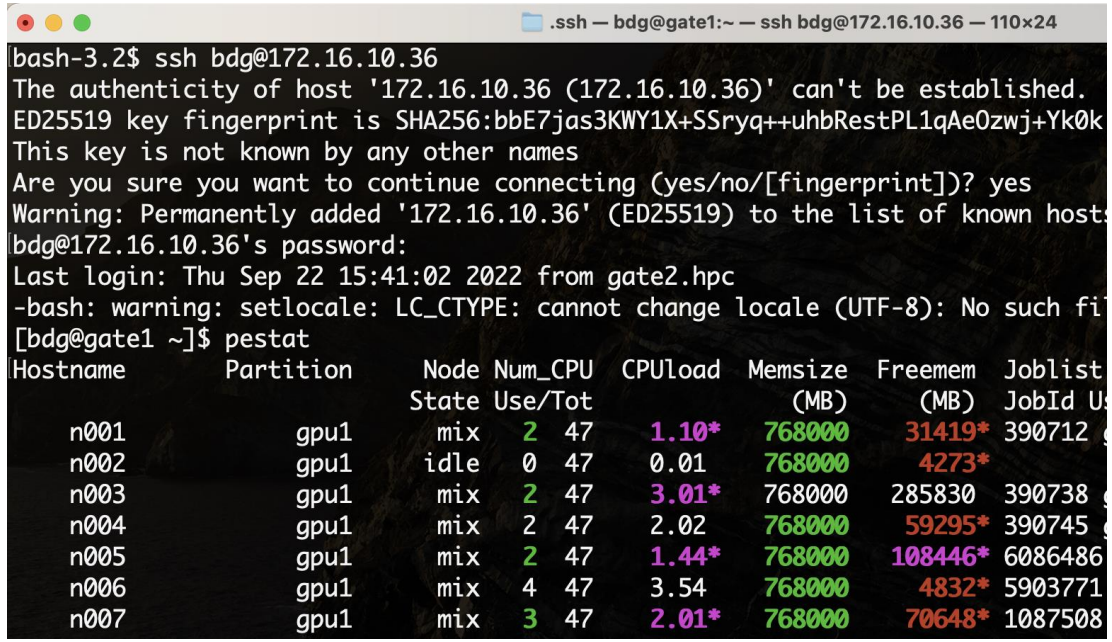
사용자id@172.16.10.37's password: **사용자password**

3. Slurm 클러스터 확인

[사용자id@gate2 ~]$ **# 접속 성공**

```
Hostname       Partition     Node
Num_CPU  CPUload  Memsize  Freemem  Joblist
                   State Use/Tot          (MB)     (MB)  JobId User ...
   n001          gpu1      mix   2  47    1.10*
   n002          gpu1     idle   0  47    0.01
   n003          gpu1      mix   2  47    3.01*
   n004          gpu1      mix   2  47    2.02
   n005          gpu1      mix   2  47    1.44*
   n006          gpu1      mix   4  47    3.54
   n007          gpu1      mix   3  47    2.01*
```

2. 외부 접속방법



 # 기본 터미널

 # Linux/macOS Bashshell 실행

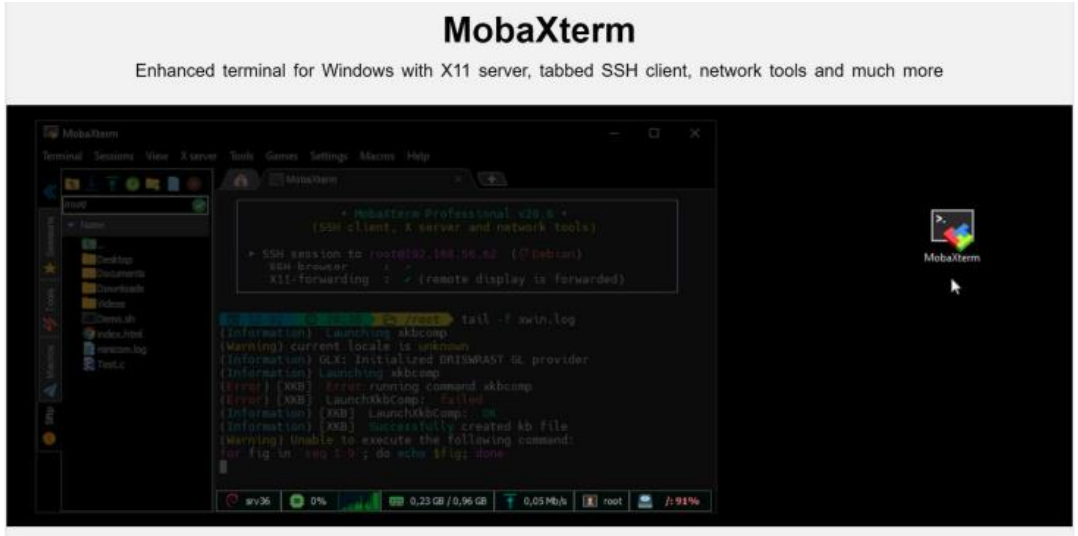
 # 1. ssh 접속

bash-3.2$ **ssh 사용자id@172.16.10.36 혹은 172.16.10.37**
... continue connecting (yes/no/[fingerprint])? **yes 입력**
사용자id@172.16.10.36's password: **사용자password 입력**
...
[사용자id@gate1 ~]$ **# 접속 성공**

 # 2. Slurm 클러스터 확인

[사용자id@gate1 ~]$ **pestat**

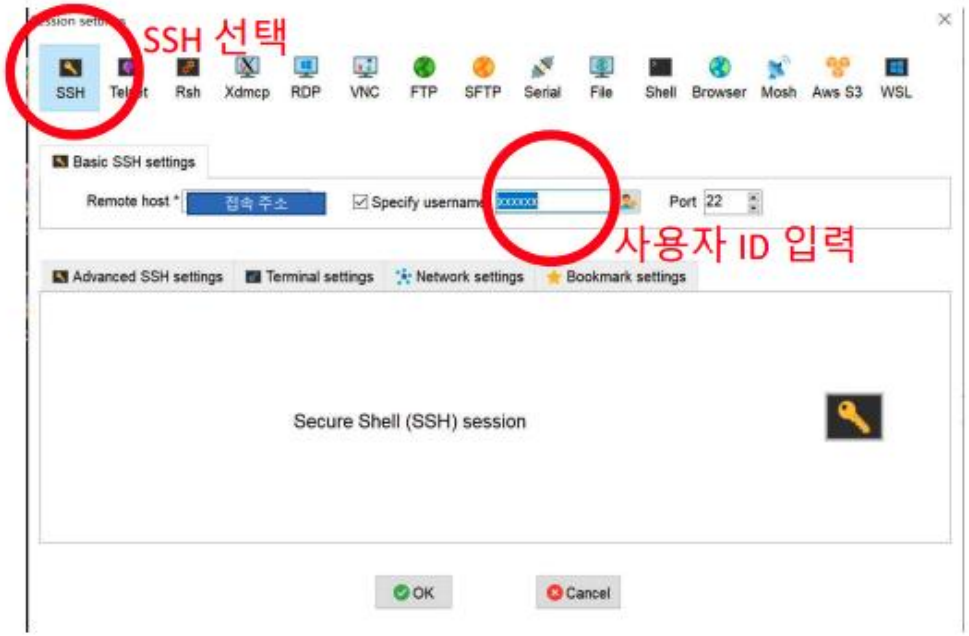
2. 외부 접속방법



 # 상용 터미널 프로그램 mobaXterm

1. mobaXterm (https://mobaxterm.mobatek.net) 설치
2. ssh 선택
3. 접속 주소 172.16.10.36 혹은 172.16.10.37 입력
4. 사용자id 입력
5. password 입력
6. 접속 성공

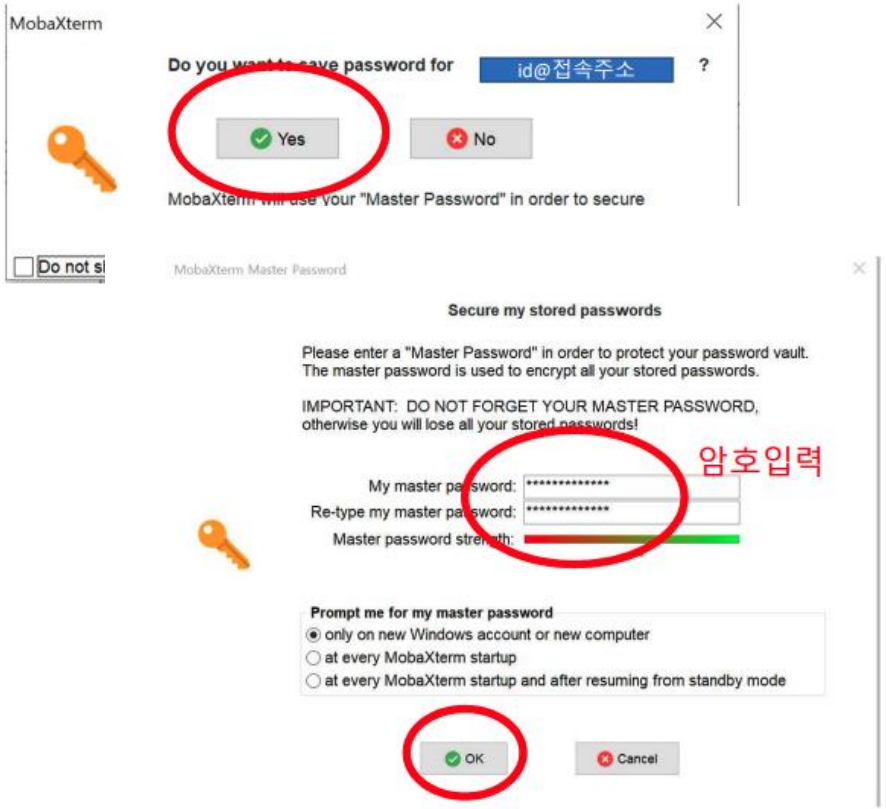
2. 외부 접속방법



 # 상용 터미널 프로그램 mobaXterm

1. mobaXterm (https://mobaxterm.mobatek.net) 설치
2. ssh 선택
3. 접속 주소 172.16.10.36 혹은 172.16.10.37 입력
4. 사용자id 입력
5. password 입력
6. 접속 성공

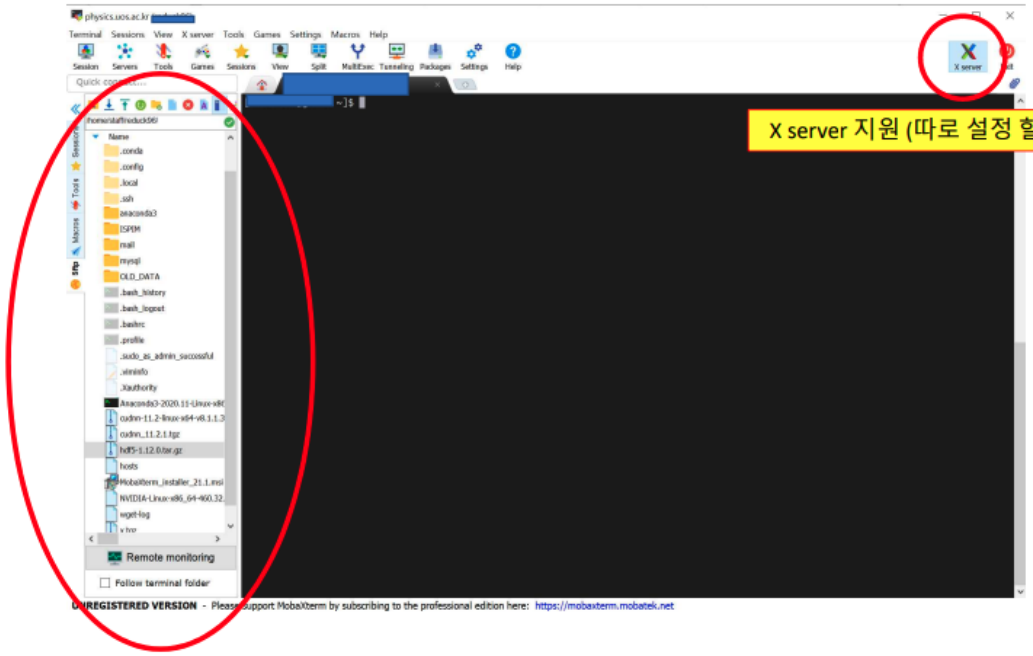
2. 외부 접속방법



 # 상용 터미널 프로그램 mobaXterm

1. mobaXterm (https://mobaxterm.mobatek.net) 설치
2. ssh 선택
3. 접속 주소 172.16.10.36 혹은 172.16.10.37 입력
4. 사용자id 입력
5. **password 입력**
6. 접속 성공

2. 외부 접속방법

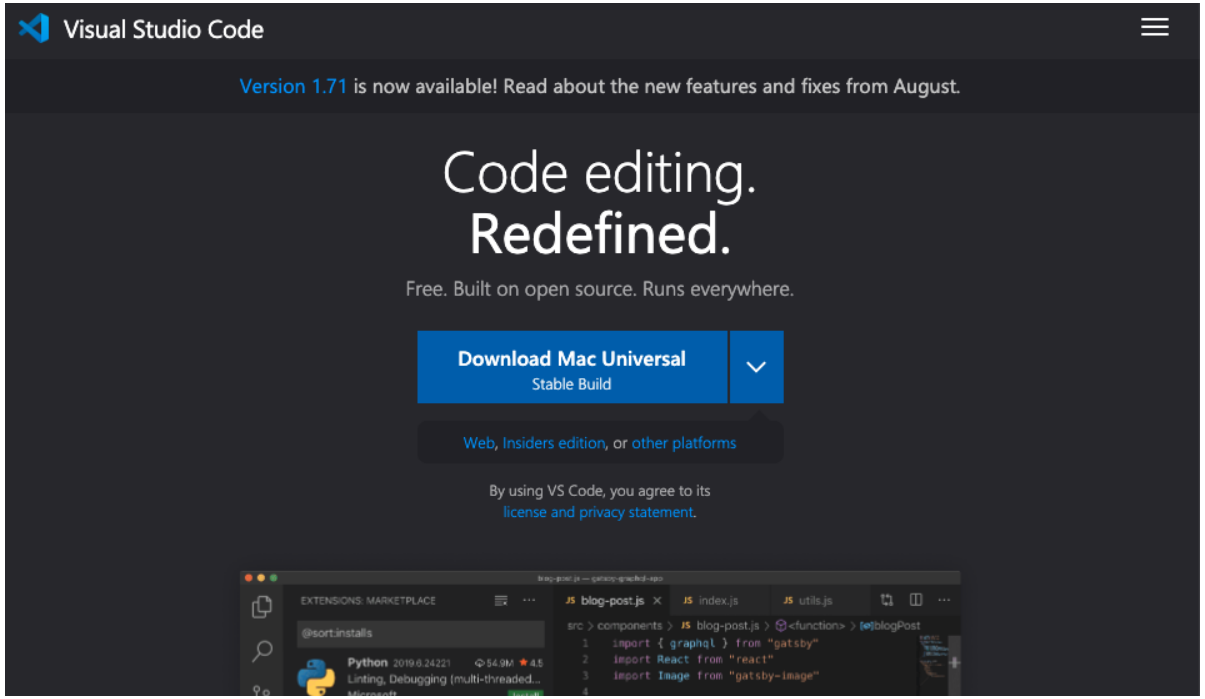


자유로운 파일 업로드/다운로드(마우스로 드래그 앤 드롭 지원)

 # 상용 터미널 프로그램 mobaXterm

1. mobaXterm (https://mobaxterm.mobatek.net) 설치
2. ssh 선택
3. 접속 주소 172.16.10.36 혹은 172.16.10.37 입력
4. 사용자id 입력
5. password 입력
6. 접속 성공

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

 # 프로그램 설치 및 초기 설정

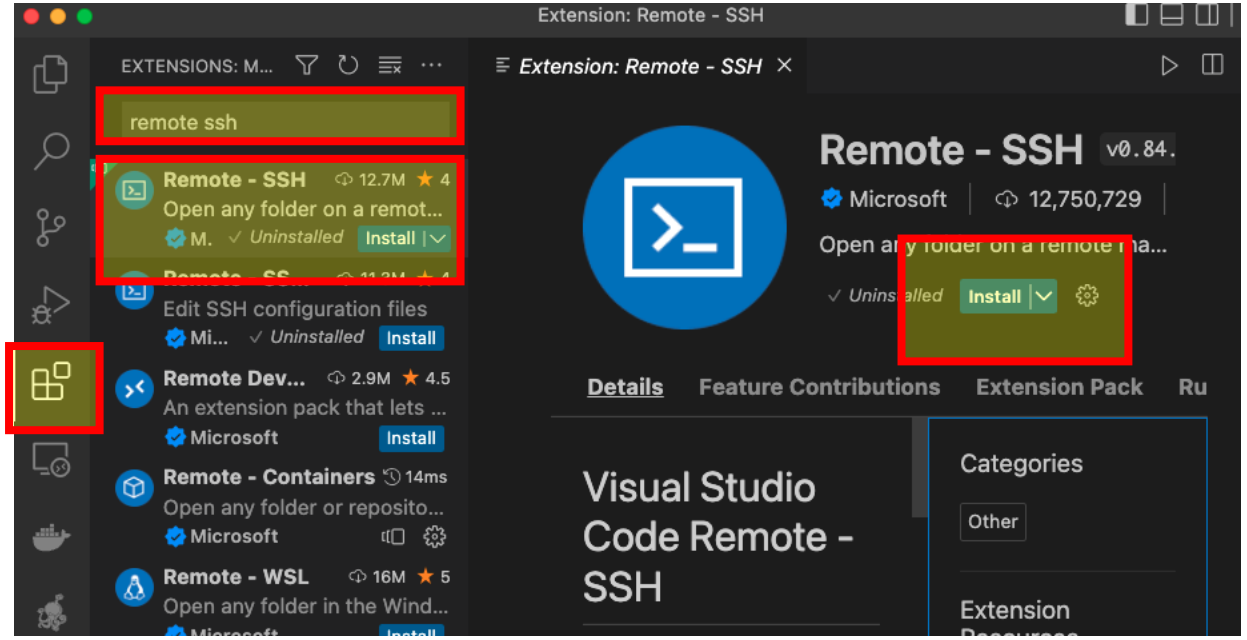
1. VScode (https://code.visualstudio.com/) 설치

2. Plugin에서 remote ssh 검색하여 설치
3. Ctrl + Shift + P (혹은 Command + Shift + P)
4. > Remote-SSH: Open SSH Confi... 검색하여 클릭
5. /Users/사용자/.ssh/config 클릭
6. config 파일 작성(User는 발급받은 사용자 id 입력)
7. ctrl + S (저장)

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭
10. kwon host 등록(continue 클릭), password 입력
11. 왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)
12. OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

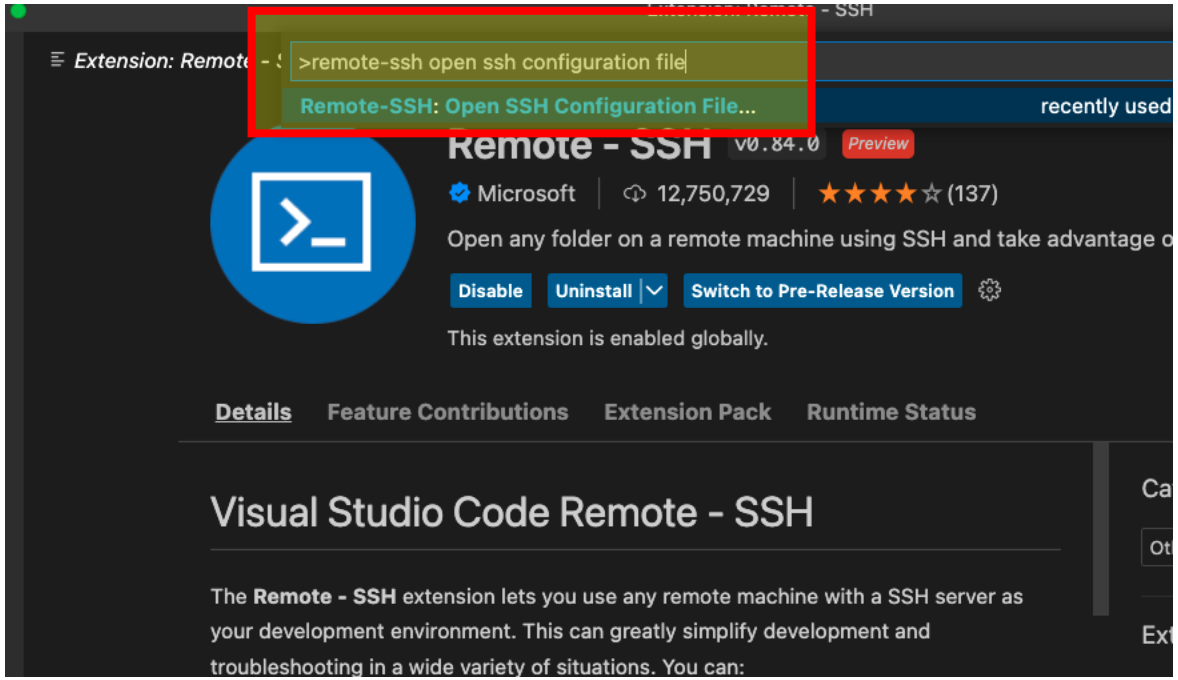
 # 프로그램 설치 및 초기 설정

1. VScode (https://code.visualstudio.com/) 설치
2. **Plugin에서 remote ssh 검색하여 설치**
3. Ctrl + Shift + P (혹은 Command + Shift + P)
4. > Remote-SSH: Open SSH Confi... 검색하여 클릭
5. /Users/사용자/.ssh/config 클릭
6. config 파일 작성(User는 발급받은 사용자 id 입력)
7. ctrl + S (저장)

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭
10. kwon host 등록(continue 클릭), password 입력
11. 왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)
12. OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

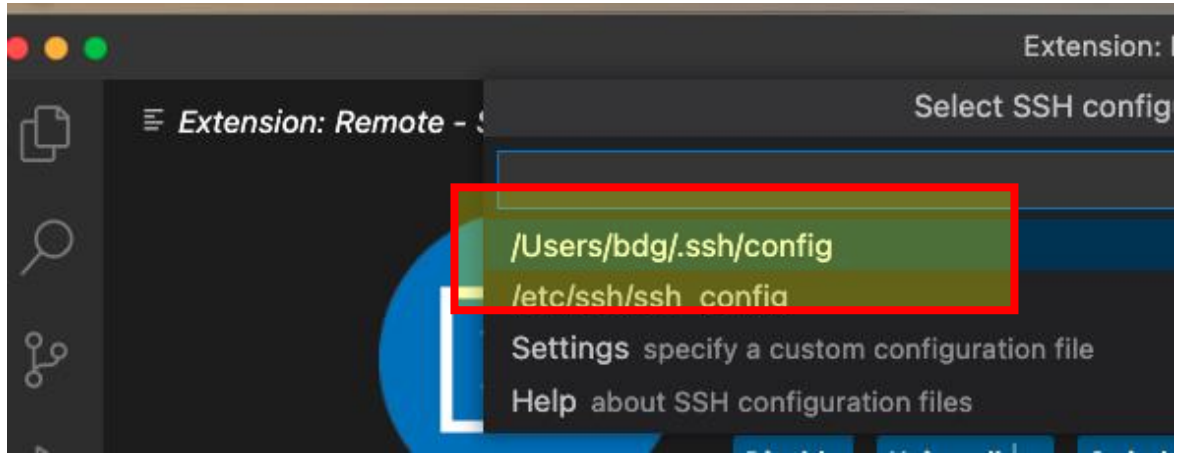
 # 프로그램 설치 및 초기 설정

1. VScode (https://code.visualstudio.com/) 설치
2. Plugin에서 remote ssh 검색하여 설치
3. **Ctrl + Shift + P (혹은 Command + Shift + P)**
4. **> Remote-SSH: Open SSH Confi... 검색하여 클릭**
5. /Users/사용자/.ssh/config 클릭
6. config 파일 작성(User는 발급받은 사용자 id 입력)
7. ctrl + S (저장)

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭
10. kwon host 등록(continue 클릭), password 입력
11. 왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)
12. OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

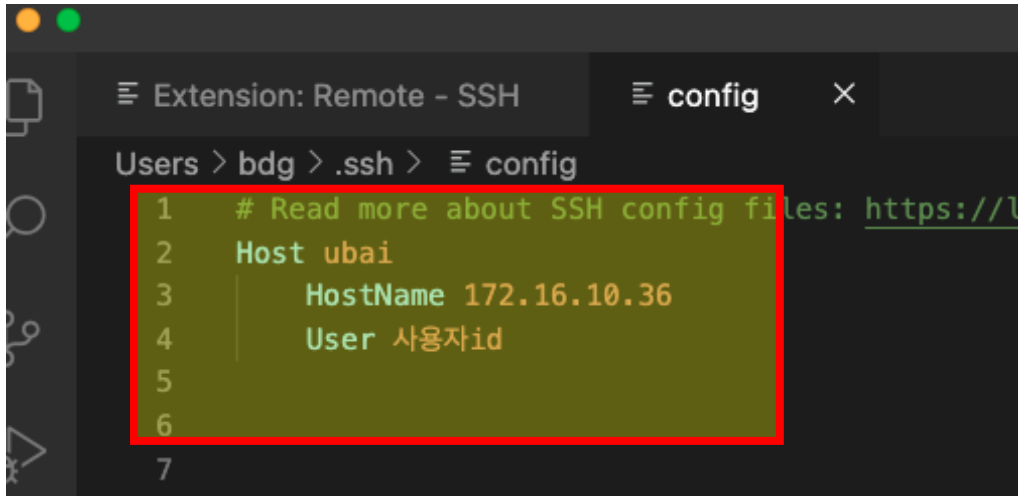
 # 프로그램 설치 및 초기 설정

1. VScode (https://code.visualstudio.com/) 설치
2. Plugin에서 remote ssh 검색하여 설치
3. Ctrl + Shift + P (혹은 Command + Shift + P)
4. > Remote-SSH: Open SSH Confi... 검색하여 클릭
5. **/Users/사용자/.ssh/config 클릭**
6. config 파일 작성(User는 발급받은 사용자 id 입력)
7. ctrl + S (저장)

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭
10. kwon host 등록(continue 클릭), password 입력
11. 왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)
12. OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

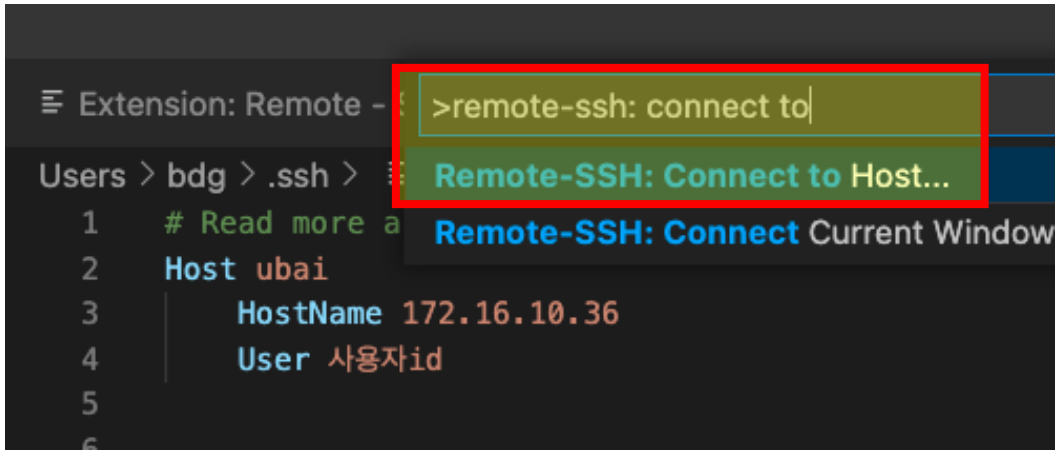
 # 프로그램 설치 및 초기 설정

1. VScode (https://code.visualstudio.com/) 설치
2. Plugin에서 remote ssh 검색하여 설치
3. Ctrl + Shift + P (혹은 Command + Shift + P)
4. > Remote-SSH: Open SSH Confi... 검색하여 클릭
5. /Users/사용자/.ssh/config 클릭
6. **config 파일 작성(User는 발급받은 사용자 id 입력)**
7. **ctrl + S (저장)**

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭
10. kwon host 등록(continue 클릭), password 입력
11. 왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)
12. OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

 # 프로그램 설치 및 초기 설정

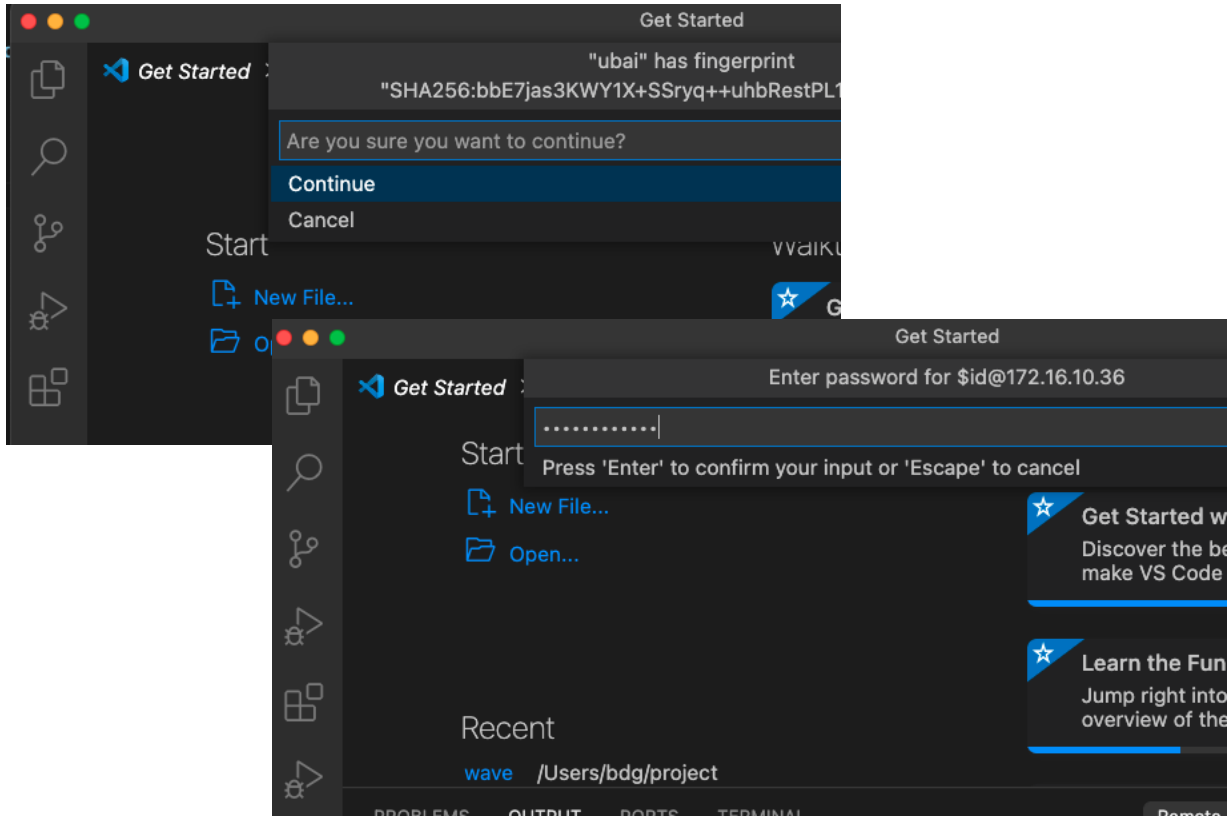
1. VScode (https://code.visualstudio.com/) 설치
2. Plugin에서 remote ssh 검색하여 설치
3. Ctrl + Shift + P (혹은 Command + Shift + P)
4. > Remote-SSH: Open SSH Confi... 검색하여 클릭
5. /Users/사용자/.ssh/config 클릭
6. config 파일 작성(User는 발급받은 사용자 id 입력)
7. ctrl + S (저장)

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭

10. kwon host 등록(continue 클릭), password 입력
11. 왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)
12. OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

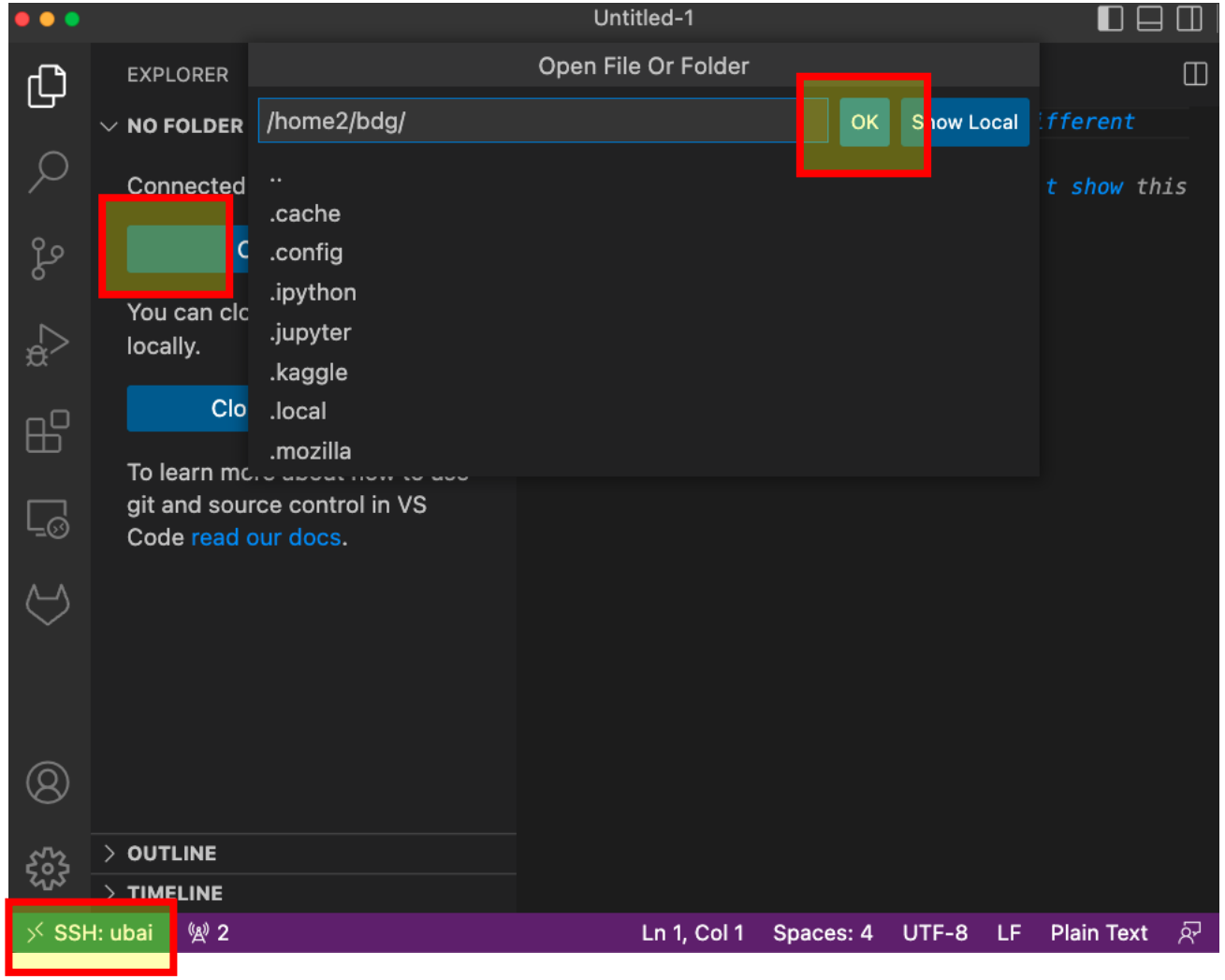
 # 프로그램 설치 및 초기 설정

1. VScode (https://code.visualstudio.com/) 설치
2. Plugin에서 remote ssh 검색하여 설치
3. Ctrl + Shift + P (혹은 Command + Shift + P)
4. > Remote-SSH: Open SSH Confi... 검색하여 클릭
5. /Users/사용자/.ssh/config 클릭
6. config 파일 작성(User는 발급받은 사용자 id 입력)
7. ctrl + S (저장)

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭
10. **kwon host 등록(continue 클릭), password 입력**
11. 왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)
12. OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭

2. 외부 접속방법



 # 상용 터미널 프로그램 VScode 설치

 # 프로그램 설치 및 초기 설정

1. VScode (https://code.visualstudio.com/) 설치
2. Plugin에서 remote ssh 검색하여 설치
3. Ctrl + Shift + P (혹은 Command + Shift + P)
4. > Remote-SSH: Open SSH Confi... 검색하여 클릭
5. /Users/사용자/.ssh/config 클릭
6. config 파일 작성(User는 발급받은 사용자 id 입력)
7. ctrl + S (저장)

 # ssh 접속

8. Ctrl + Shift + P (혹은 Command + Shift + P)
9. > Remote-SSH: Connect to Host... 검색하여 클릭
10. kwon host 등록(continue 클릭), password 입력
11. **왼쪽 아래 SSH:ubai 확인 (<- 접속 성공)**
12. **OpenFolder 클릭하여 /home1,2/사용자id/ OK클릭**

3. 기초 환경 호출 방법 **(Linux Environment Modules)**

1. module avil
2. module load
3. module list
4. module rm
5. module purge

```
------------------------- /TGM/local/etc/modulefiles -------
ANACONDA/2019.10            CUDA/11.2.2
ANACONDA/2020.02_Python3.7  CUDA/11.3.0
ANACONDA/2020.07_Python3.8  GNUcompiler/10.2.0
ANACONDA/2020.11            GNUcompiler/9.3.0
ANACONDA/2021.05            GROMACS/2021.2
Compiler/Intel/17.0.7       MKL/2018.5.274
Compiler/Intel/18.0.5       MKL/2020.4.304
Compiler/Intel/19.1.3.304   MPI/intel/2018.4.274
CUDA/10.0.130               MPI/intel/2019.8.254
CUDA/10.1.105               MPI/intel/2019.9.304
CUDA/10.2.89                NV_HPC_SDK/21.5
CUDA/11.0.3                 QE/6.8_CPU
CUDA/11.1.1                 QE/6.8_GPU
```

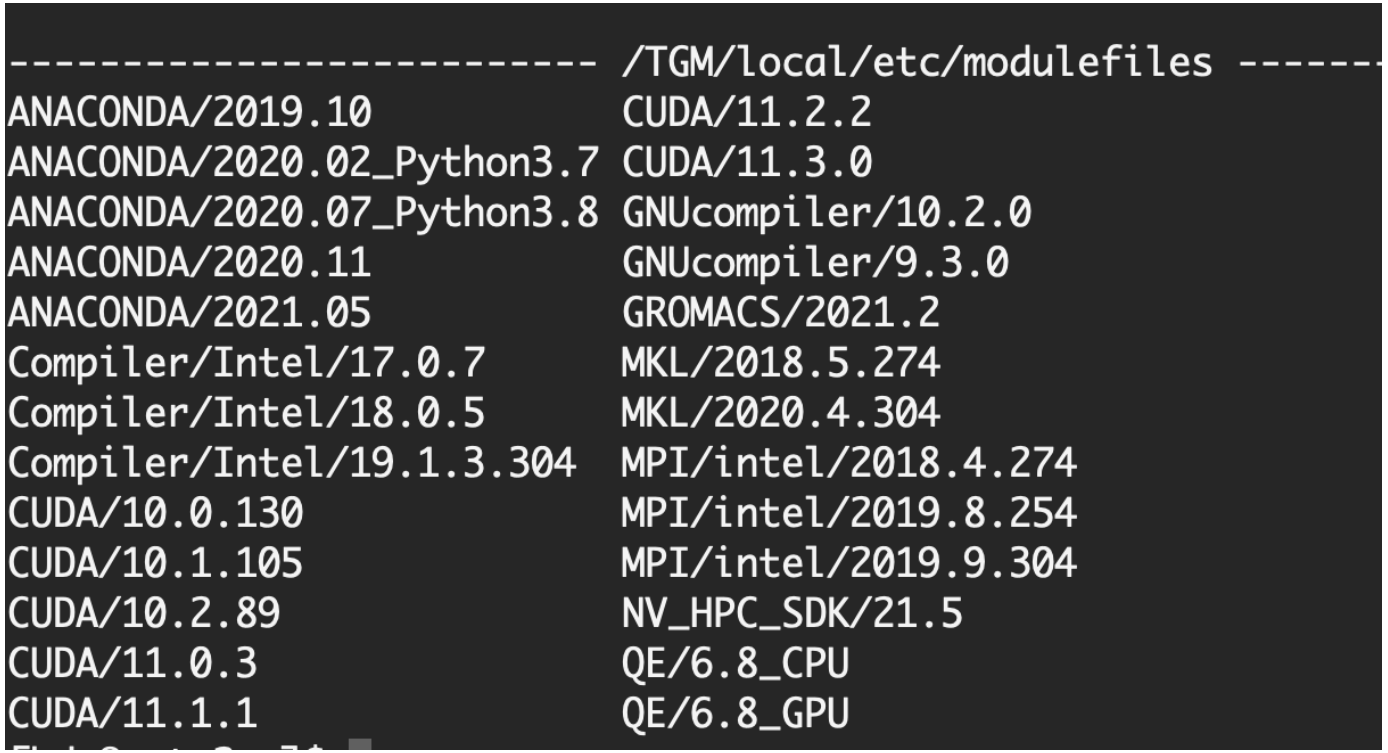
손쉽게 사용자의 shell을 초기화하고

사용환경을 변경할 수 있도록 도와주는 도구

3. 기초 환경 호출 방법 **(Linux Environment Modules)**

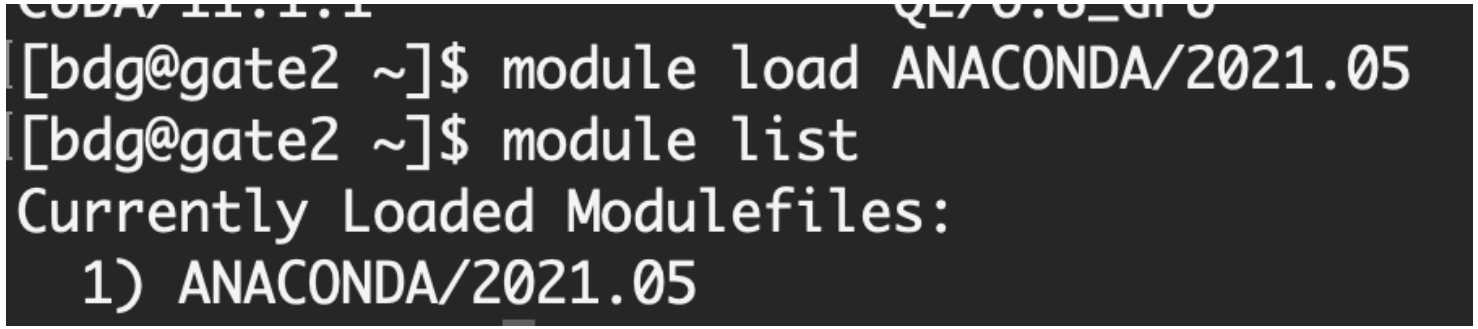
1. **module avil**
   UBAI cluster 내에서 사용가능한 모듈 확인

2. module load
3. module list
4. module rm
5. module purge



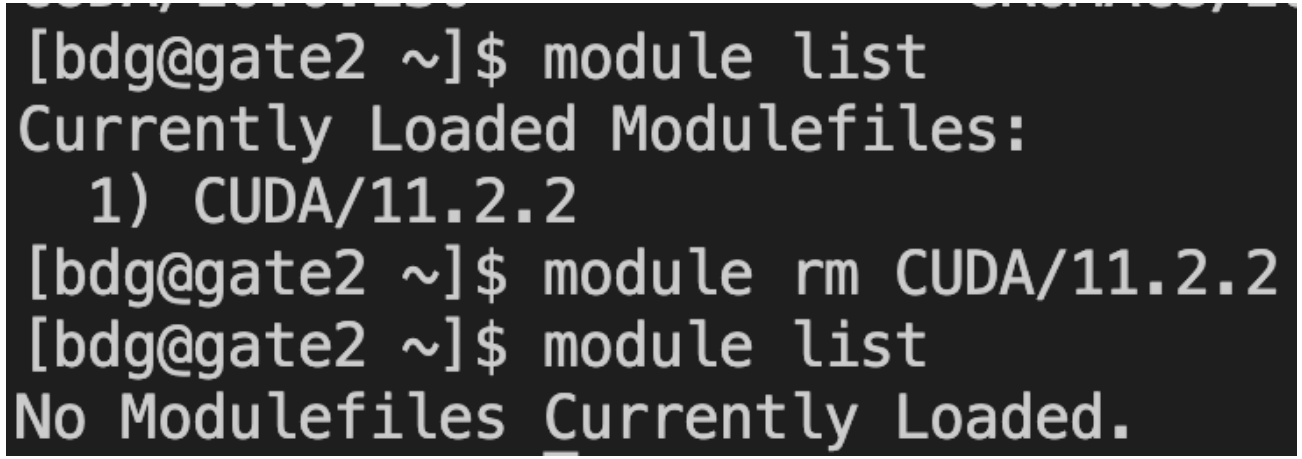
3. 기초 환경 호출 방법 **(Linux Environment Modules)**

1. module avil
2. **module load**
   특정 모듈 불러오기
3. **module list**
   현재 사용중인 모듈 목록 확인하기
4. module rm
5. module purge



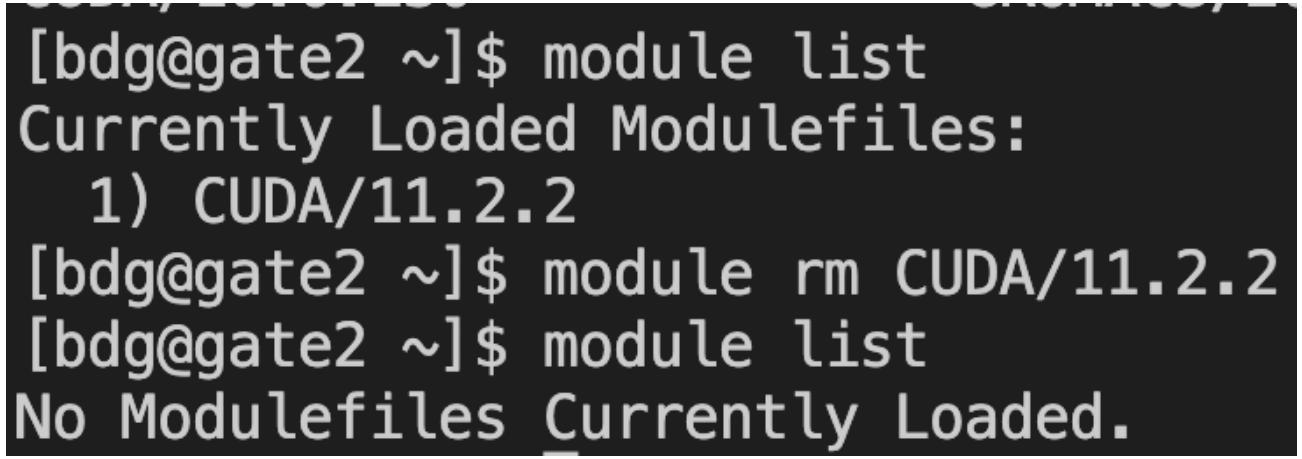
3. 기초 환경 호출 방법 **(Linux Environment Modules)**

1. module avil
2. module load
3. module list
4. **module rm**
   사용목록에서 특정 모듈 제거
5. **module purge**
   사용목록 초기화



3. 기초 환경 호출 방법 **(Anaconda Python Virtualenv)**

1. module avil
2. module load
3. module list
4. **module rm**
   사용목록에서 특정 모듈 제거
5. **module purge**
   사용목록 초기화

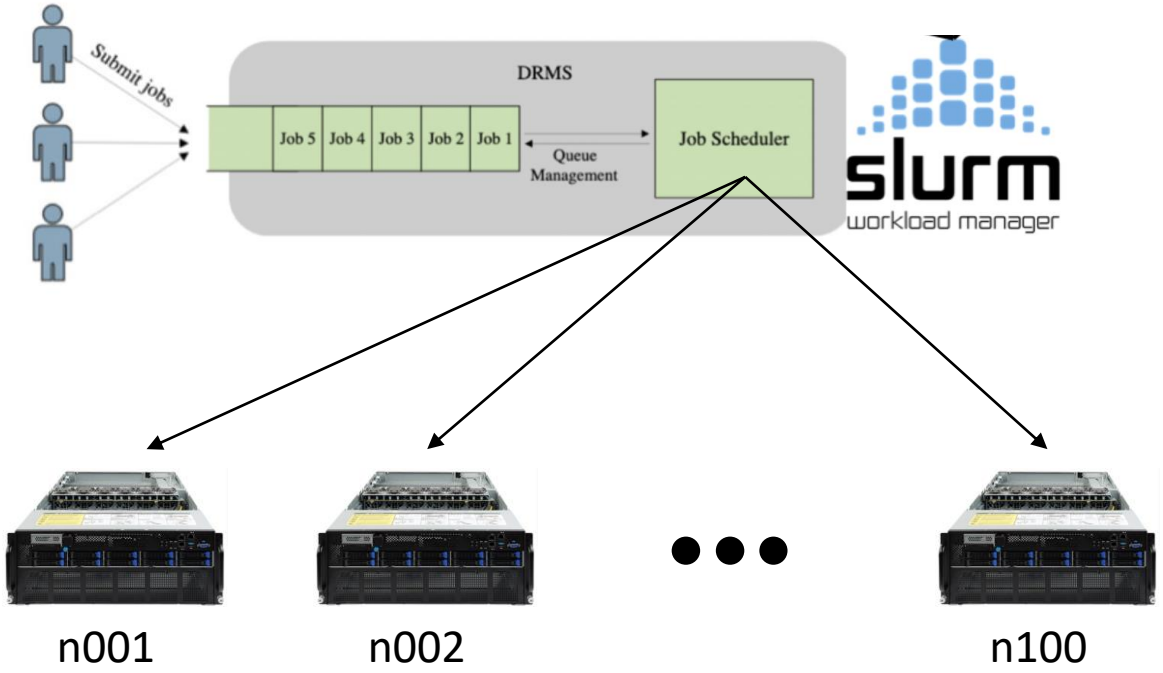


4. SLURM Job Scheduler 사용법

1. sinfo
2. pestat
3. Sbatch
4. squeue
5. qstat
6. scontrol show job JOBID

```
[bdg@gate2 ~]$ pestat
Hostname       Partition     Node Num_CPU  CPUload  Memsize  Freemem
                           State Use/Tot              (MB)     (MB)
    n001           gpu1      mix   1  47    0.36*   768000    12086*
    n002           gpu1      mix   3  47    1.57*   768000     3224*
    n003           gpu1      mix   2  47    3.04*   768000   285677
    n004           gpu1      mix   2  47    2.01    768000    59421*
    n005           gpu1      mix   1  47    1.03    768000   111504*
    n006           gpu1      mix   3  47    2.85    768000     3789*
    n007           gpu1      mix   3  47    1.96*   768000    67653*
    n008           gpu1      mix   4  47    3.17*   768000   254853
    n009           gpu1      mix   2  47    2.94*   768000     2273*
    n010           gpu1      mix   4  47    3.18*   768000   151312*
    n011           gpu1      mix   3  47    2.63    768000     5125*
    n012           gpu1      mix   3  47    2.34*   768000    25564*
    n013           gpu1     idle   0  47    1.26*   768000     5499*
    n014           cpu1      mix  40  47   42.13*   768000     8028*
    n015           cpu1    alloc  47  47   43.65*   768000    71254*
    n016           cpu1      mix  28  47   33.15*   768000    10326*
    n017           cpu1      mix  46  47   33.13*   768000    19174*
    n018           cpu1      mix  28  47   31.87*   768000    62437*
    n019           cpu1      mix  46  47   31.67*   768000    42982*
    n020           cpu1      mix  40  47   40.77*   768000    90361*
    n021           cpu1      mix  46  47   32.72*   768000    45863*
    n022           cpu1      mix  46  47   34.30*   768000     7745*
    n023           cpu1    alloc  47  47   47.90*   768000   210109
    n024           cpu1      mix  28  47   20.14*   768000   204385
    n025           cpu1      mix  46  47   32.33*   768000    64315*
    n026           cpu1      mix  46  47   31.00*   768000    21246*
    n027           cpu1      mix  40  47   41.38*   768000    25774*
    n028           cpu1    alloc  47  47   46.91    768000    15734*
```

slurm은 cluster 로 들어오는 다양한 작업 요청을 적절한 계산 노드에 할당해주는 관리 도구이다.



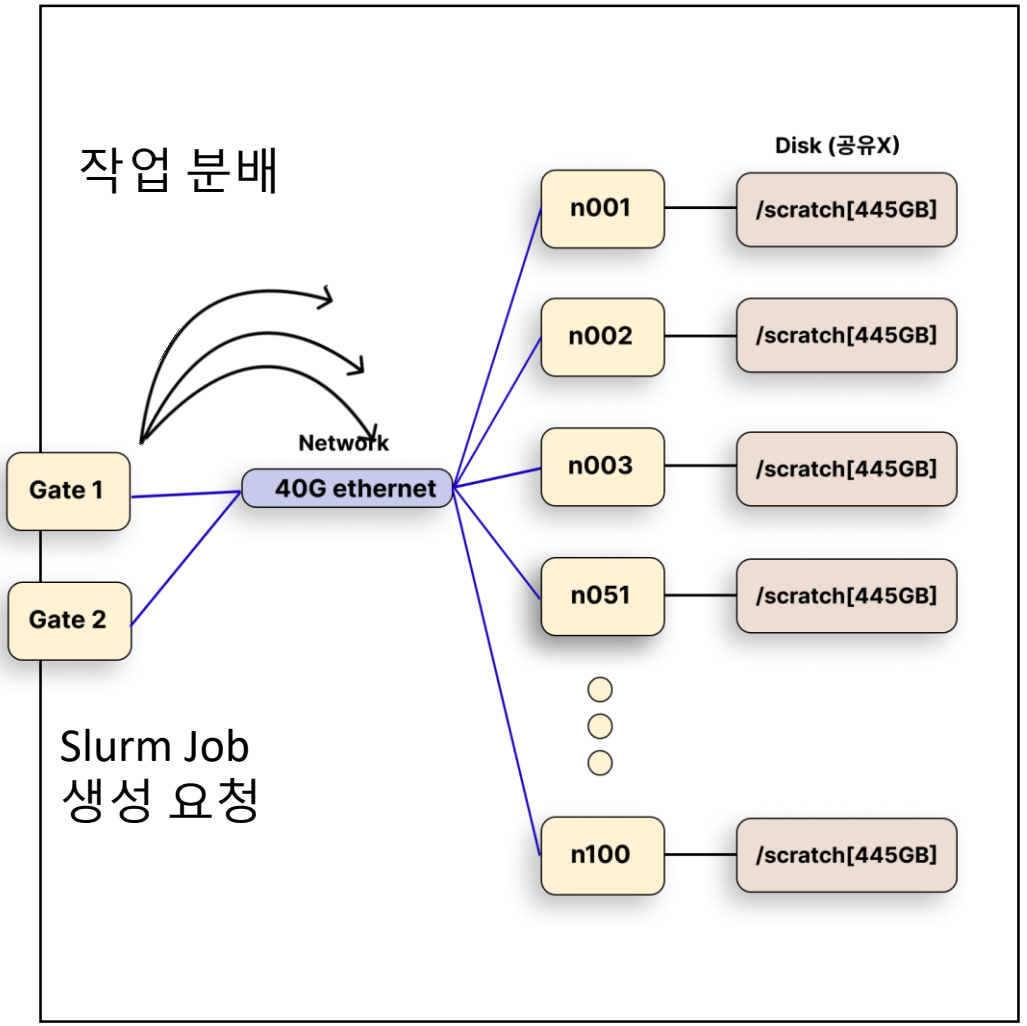
4. SLURM Job Scheduler 사용법

서울 시립대학교 내부 네트워크

172.16.10.36:22

172.16.10.37:22

SSH 접속



4. SLURM Job Scheduler 사용법

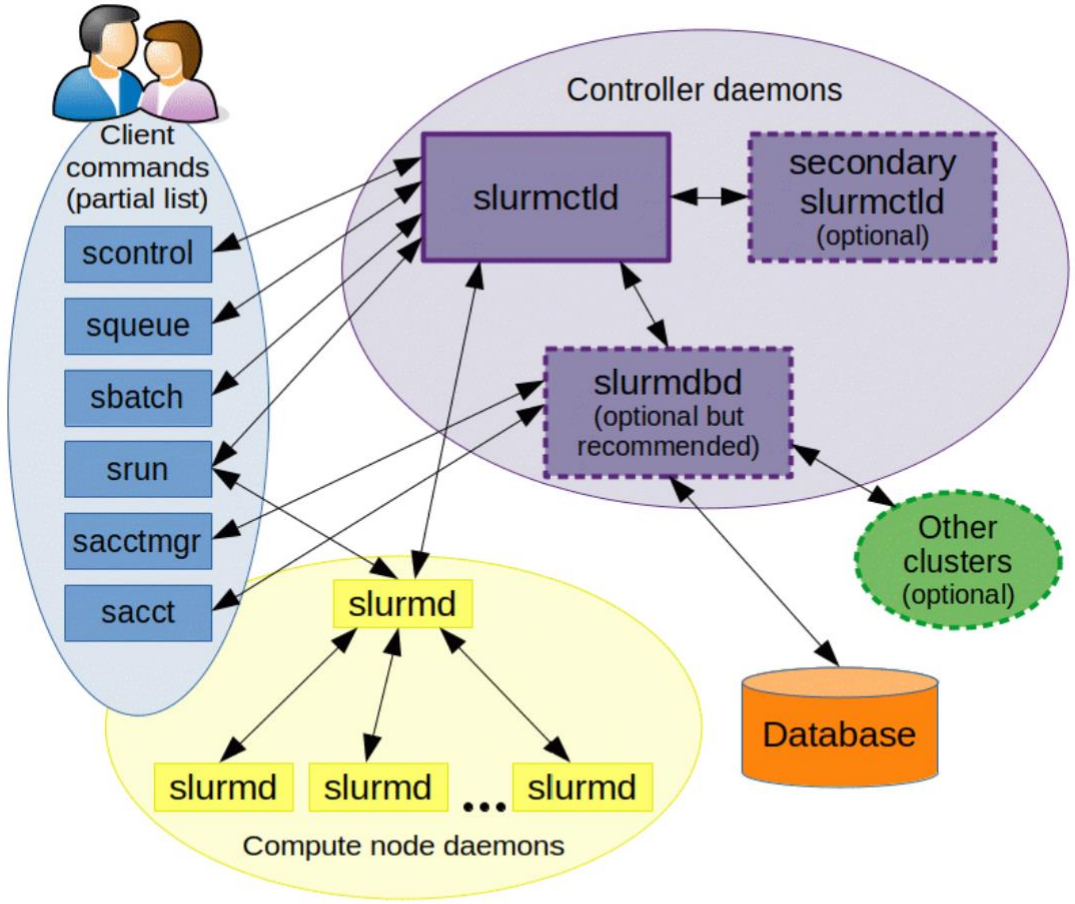


Figure 1. Slurm components

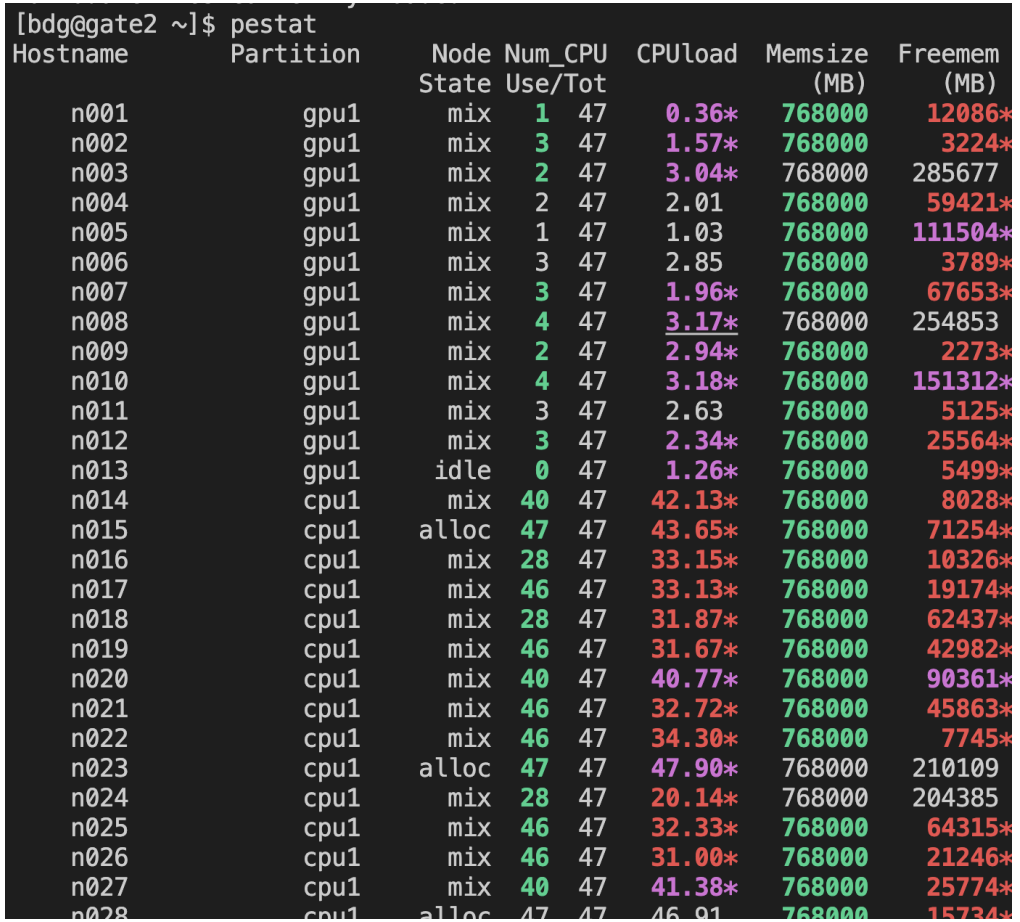
4. SLURM Job Scheduler 사용법

1. **sinfo**
   전체 시스템 상태 확인
2. pestat
3. sbatch
4. squeue
5. qstat
6. scontrol show job JOBID

```
[ubai01@tgm-master ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
gpu1         up   infinite     12    mix n[001-005,007-013]
gpu1         up   infinite      1   idle n006
cpu1         up   infinite     14    mix n[014,016-017,019-021,023-026,029-032]
cpu1         up   infinite      7  alloc n[015,018,022,027-028,033-034]
cpu1         up   infinite     14   idle n[035-048]
hgx          up   infinite      1    mix n050
gpu2         up   infinite      1  down* n069
gpu2         up   infinite      1   comp n055
gpu2         up   infinite      3    mix n[051,053-054]
gpu2         up   infinite      3  alloc n[052,063,066]
gpu2         up   infinite     28   idle n[056-062,064-065,067-068,070-086]
cpu2*        up   infinite      1    mix n089
cpu2*        up   infinite      8  alloc n[087-088,091-096]
cpu2*        up   infinite      5   idle n[090,097-100]
```

4. SLURM Job Scheduler 사용법

1. sinfo
2. **pestat**
   계산 노드에 할당된 JobId 확인
   내가 요청한 job이 실행되는지 확인하는 용도
3. sbatch
4. squeue
5. qstat
6. scontrol show job JOBID



```
[bdg@gate2 ~]$ pestat
Hostname       Partition     Node Num_CPU  CPUload  Memsize  Freemem
                         State Use/Tot              (MB)     (MB)
    n001           gpu1      mix   1  47    0.36*   768000   12086*
    n002           gpu1      mix   3  47    1.57*   768000    3224*
    n003           gpu1      mix   2  47    3.04*   768000  285677
    n004           gpu1      mix   2  47     2.01   768000   59421*
    n005           gpu1      mix   1  47     1.03   768000  111504*
    n006           gpu1      mix   3  47     2.85   768000    3789*
    n007           gpu1      mix   3  47    1.96*   768000   67653*
    n008           gpu1      mix   4  47    3.17*   768000  254853
    n009           gpu1      mix   2  47    2.94*   768000    2273*
    n010           gpu1      mix   4  47    3.18*   768000  151312*
    n011           gpu1      mix   3  47     2.63   768000    5125*
    n012           gpu1      mix   3  47    2.34*   768000   25564*
    n013           gpu1     idle   0  47    1.26*   768000    5499*
    n014           cpu1      mix  40  47   42.13*   768000    8028*
    n015           cpu1    alloc  47  47   43.65*   768000   71254*
    n016           cpu1      mix  28  47   33.15*   768000   10326*
    n017           cpu1      mix  46  47   33.13*   768000   19174*
    n018           cpu1      mix  28  47   31.87*   768000   62437*
    n019           cpu1      mix  46  47   31.67*   768000   42982*
    n020           cpu1      mix  40  47   40.77*   768000   90361*
    n021           cpu1      mix  46  47   32.72*   768000   45863*
    n022           cpu1      mix  46  47   34.30*   768000    7745*
    n023           cpu1    alloc  47  47   47.90*   768000  210109
    n024           cpu1      mix  28  47   20.14*   768000  204385
    n025           cpu1      mix  46  47   32.33*   768000   64315*
    n026           cpu1      mix  46  47   31.00*   768000   21246*
    n027           cpu1      mix  40  47   41.38*   768000   25774*
    n028           cpu1    alloc  47  47   46.91   768000   15734*
```

4. SLURM Job Scheduler 사용법

1. sinfo
2. pestat
3. **sbatch**
   작업 요청 (세부 내용은 예제에서 설명)
4. squeue
5. qstat
6. scontrol show job JOBID

test.sh 작성

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --nodelist=n003
#SBATCH --cpus-per-task=2
#SBATCH --ntasks-per-node=1    # Cores per node
#SBATCH --partition=gpu1        # Partition Name (gpu, hgx)
##
#SBATCH --job-name=bdg-test
#SBATCH -o ./log/SLURM.%N.%j.out         # STDOUT
#SBATCH -e ./log/SLURM.%N.%j.err         # STDERR

hostname
date

/bin/sleep 10

echo 'start test job'

echo 'done!'
```

4. SLURM Job Scheduler 사용법

1. sinfo
2. pestat
3. **sbatch /scancel jobid**
   작업 요청 (세부 내용은 예제에서 설명)
4. squeue
5. qstat
6. scontrol show job JOBID

test.sh 실행

```
[bdg@gate2 test-job]$ sbatch test.sh
Submitted batch job 6107675
[bdg@gate2 test-job]$ pestat
Hostname       Partition     Node Num_CPU  CPUload  Memsize  Freemem  Joblist
                             State Use/Tot              (MB)     (MB)  JobId User ...
    n001           gpu1      mix   1  47    0.08*   768000    15806* 390712
    n002           gpu1      mix   3  47    1.35*   768000     3217* 5904101
    n003           gpu1      mix   4  47    3.04*   768000   285690  6107675 bdg
    n004           gpu1      mix   2  47    2.03    768000    58856* 390745
    n005           gpu1      mix   1  47    1.06    768000   111597* 2866492
```

4. SLURM Job Scheduler 사용법

1. sinfo
2. pestat
3. sbatch
4. **squeue**
   제출된 작업 상태 확인
5. qstat
6. scontrol show job JOBID

```
[bdg@gate2 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           6077373      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077370      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077368      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077365      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077363      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077360      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077358      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077355      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
           6077353      cpu1 1.08Ener [redacted] PD       0:00      1 (AssocMaxJobsLimit)
```

JOB STATE CODES

CA CANCELLED
CD COMPLETED
CF CONFIGURING
CG COMPLETING
DL DEADLINE
F FAILED
NF NODE_FAIL
OOM OUT_OF_MEMORY
PD PENDING
PR PREEMPTED
R RUNNING
RD RESV_DEL_HOLD
RF REQUEUE_FED
RH REQUEUE_HOLD
RQ REQUEUED
RS RESIZING
RV REVOKED
SI SIGNALING
SE SPECIAL_EXIT
SO STAGE_OUT
ST STOPPED
S SUSPENDED
TO TIMEOUT

https://slurm.schedmd.com/squeue.html

4. SLURM Job Scheduler 사용법

1. sinfo
2. pestat
3. sbatch
4. squeue
5. **qstat (-r)**
   제출된 작업 확인
6. scontrol show job JOBID

Queue -> Partition 으로 해석

```
[bdg@gate2 ~]$ qstat
Job id                    Name             Username        Time Use S Queue
------------------------- ---------------- --------------- -------- - ---------------
390712                    Total.py                         202:06:3 R gpu1
390738                    Total.py                         202:05:1 R gpu1
390744                    Total.py                         202:04:4 R gpu1
390745                    Total.py                         202:04:4 R gpu1
1087508                   title_preprocess                 56:10:23 R gpu1
2866492                   fakenews                         32:04:58 R gpu1
4167419                   dasan_test                       20:04:53 R gpu1
4843130                   dasan_data                       13:06:08 R gpu1
```

4. SLURM Job Scheduler 사용법

1. sinfo
2. pestat
3. sbatch
4. squeue
5. qstat
6. **scontrol show job JOBID**
   종료된 job에 대한 상세한 정보 조회

```
[bdg@gate2 test-job]$ scontrol show job 6134741
JobId=6134741 JobName=bdg-test
   UserId=bdg(1132) GroupId=users(100) MCS_label=N/A
   Priority=4168 Nice=0 Account=physics QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:10 TimeLimit=2-00:00:00 TimeMin=N/A
   SubmitTime=2022-09-26T23:39:38 EligibleTime=2022-09-26T23:39:38
   AccrueTime=2022-09-26T23:39:38
   StartTime=2022-09-26T23:39:38 EndTime=2022-09-28T23:39:38 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2022-09-26T23:39:38
   Partition=gpu1 AllocNode:Sid=gate2:103804
   ReqNodeList=n003 ExcNodeList=(null)
   NodeList=n003
   BatchHost=n003
   NumNodes=1 NumCPUs=2 NumTasks=1 CPUs/Task=2 ReqB:S:C:T=0:0:*:*
   TRES=cpu=2,node=1,billing=2
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
   MinCPUsNode=2 MinMemoryCPU=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
```

5. SLURM 예제

1. CPU만 사용한 작업 요청

```
[        @gate1 ~]$ cp -a /home1/SLURM_EXAMPLE/JustCPU_CORE .
[        @gate1 ~]$ cd JustCPU_CORE/
[        @gate1 JustCPU_CORE]$ ls
example.py  run_slurm.sh
[        @gate1 JustCPU_CORE]$ cat run_slurm.sh
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1    # Cores per node
#SBATCH --partition=gpu        # Partition Name (gpu, hgx)
##
#SBATCH --job-name=SlurmExample
#SBATCH -o SLURM.%N.%j.out         # STDOUT
#SBATCH -e SLURM.%N.%j.err         # STDERR
##

hostname
date

##
module add  ANACONDA/2020.11

python example.py
[        @gate1 JustCPU_CORE]$
```

#!/bin/bash	
#SBATCH --nodes=1	총 필요한 노드 수
#SBATCH --ntasks-per-node=1	노드 당 사용할 core 수
#SBATCH --partition=gpu	사용할 partition 이름 (gpu, hgx 선택 가능)
##	##
#SBATCH --job-name=SlurmExample	작업 이름
#SBATCH -o SLURM.%N.%j.out	STDOUT 파일명 (Standard output)
#SBATCH -e SLURM.%N.%j.err	STDERR 파일명 (Standard error)
##	##
hostname	계산이 작동되는 HOST 이름
date	계산 시작 시 시간
##	##
module add ANACONDA/2020.11	Module로 ANACONDA/2020.11 호출
python example.py	python example.py 실행

5. SLURM 예제

1. CPU만 사용한 작업 요청

```
[          @gate1 JustCPU_CORE]$ ls
example.py  run_slurm.sh
[          @gate1 JustCPU_CORE]$ sbatch run_slurm.sh
Submitted batch job 1555
[          @gate1 JustCPU_CORE]$ ls
example.py  run_slurm.sh  SLURM.n002.1555.err  SLURM.n002.1555.out
[          @gate1 JustCPU_CORE]$
```

```
[          @gate1 JustCPU_CORE]$ qstat
Job id                    Name             Username        Time Use S Queue
------------------------- ---------------- --------------- -------- - ---------------
525                       Q3L512near_criti                 04:16:09 R gpu
526                       Q3L512near_criti                 04:16:09 R gpu
527                       Q3L512near_criti                 04:16:09 R gpu
528                       Q3L512near_criti                 04:16:09 R gpu
529                       Q3L512near_criti                 04:16:09 R gpu
1423                      LaNiO                            01:01:26 R gpu
1424                      LaNiO                            01:01:25 R gpu
1555                      SlurmExample                     00:00:00 C gpu
[          @gate1 JustCPU_CORE]$
```

- sbatch로 job 올린 뒤 qstat으로 job 상태 확인
- Job id 가 작업 ID임, 작업별로 고유 번호 생성. 이 번호로 JOB을 cancel할 수 있고 추후 뒷 추적을 할 수 있으며 사용자 별로 자원을 어떻게 사용하였는지 확인 가능함

5. SLURM 예제

2. GPU를 사용한 작업 요청

```
[        @gate1 ~]$ rm -rf Use_GPU/
[        @gate1 ~]$ cp -a /home1/SLURM_EXAMPLE/Use_GPU .
[        @gate1 ~]$ cd Use_GPU/
[        @gate1 Use_GPU]$ ls
GPUTest.x  run_slurm.sh
        @gate1 Use_GPU]$ cat run_slurm.sh
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1    # Cores per node
#SBATCH --partition=gpu        # Partition Name (gpu, hgx)
##
#SBATCH --job-name=UseGPUExample
#SBATCH -o SLURM.%N.%j.out         # STDOUT
#SBATCH -e SLURM.%N.%j.err         # STDERR
##
#SBATCH --gres=gpu:rtx3090:1

hostname
date

module add CUDA/11.2.2

./GPUTest.x 1024 1.0 2.0 3.0 0 1000 10 0

##
```

#!/bin/bash	
#SBATCH --nodes=1	총 필요한 노드 수
#SBATCH --ntasks-per-node=1	노드 당 사용할 core 수
#SBATCH --partition=gpu	사용할 partition 이름 (gpu, hgx 선택 가능)
##	##
#SBATCH --job-name=UseGPUExample	작업 이름
#SBATCH -o SLURM.%N.%j.out	STDOUT 파일명 (Standard output)
#SBATCH -e SLURM.%N.%j.err	STDERR 파일명 (Standard error)
##	##
#SBATCH –gres=gpu:rtx3090:1	GPU를 사용하기 위한 옵션
hostname	계산이 작동되는 HOST 이름
date	계산 시작 시 시간
module add CUDA/11.2.2	Module로 CUDA 11.2.2 환경 호출
./GPUTest.x 1024 1.0 2.0 3.0 0 1000 10 0	GPU를 이용하는 프로그램 실행

5. SLURM 예제

2. GPU를 사용하는 작업 요청

```
[redacted]@gate1 Use_GPU]$ sbatch run_slurm.sh
Submitted batch job 1557
[redacted]@gate1 Use_GPU]$ scontrol show job 1557
JobId=1557 JobName=UseGPUExample
   UserId=reduck96(1000) GroupId=users(100) MCS_label=N/A
   Priority=3283 Nice=0 Account=physics QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:07 TimeLimit=UNLIMITED TimeMin=N/A
   SubmitTime=2021-05-11T02:13:09 EligibleTime=2021-05-11T02:13:09
   AccrueTime=2021-05-11T02:13:09
   StartTime=2021-05-11T02:13:09 EndTime=Unknown Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2021-05-11T02:13:09
   Partition=gpu AllocNode:Sid=gate1:102537
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=n003
   BatchHost=n003
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,node=1,billing=1
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryCPU=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home1/staff[redacted]/Use_GPU/run_slurm.sh
   WorkDir=/home1/staff[redacted]/Use_GPU
   StdErr=/home1/staff/[redacted]Use_GPU/SLURM.%N.1557.err
   StdIn=/dev/null
   StdOut=/home1/staff/[redacted]Use_GPU/SLURM.%N.1557.out
   Power=
   TresPerNode=gpu:rtx3090:1
   NtasksPerTRES:0

[redacted]@gate1 Use_GPU]$
```

GPU사용 시 –gres 옵션 꼭 추가해야 함

--gres 옵션 없이 job 예약 하고 프로그램에서 GPU 호출 시 잡 죽음

Partition name이 gpu인 그룹은 현재(2021년 5월 기준) RTX3090 카드를 1개 밖에 쓸 수 없음

Partition name이 hgx인 경우에는 A100 GPU를 최대 8개까지 쓸 수 있음.
GRES 옵션 규칙 : #SBATCH –gres=gpu:hgx:8

만약 A100을 2개만 사용한다면 다음과 같이 입력함
#SBATCH –gres=gpu:hgx:2

5. SLURM 예제

3. 프로그램별 사용 예제

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24      # Cores per node
#SBATCH --partition=gpu          # Partition Name
##
#SBATCH --job-name=test36
#SBATCH -o SJrecal.%N.%j.out          # STDOUT
#SBATCH -e SJrecal.%N.%j.err          # STDERR
##

hostname
date

cd $SLURM_SUBMIT_DIR
mpirun -np $SLURM_NTASKS /home1/staff/reduck96/VASP/bin/6.2.0/O3/NORMAL/vasp.6.2.0.NORMAL.wannier90v3.ncl.x > stdout.log
```

```
[redacted]@gate1 R_Example]$ cat run_slurm.sh
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1     # Cores per node
#SBATCH --partition=gpu         # Partition Name (gpu, hgx)
##
#SBATCH --job-name=SlurmExample
#SBATCH -o SLURM.%N.%j.out          # STDOUT
#SBATCH -e SLURM.%N.%j.err          # STDERR
##

hostname
date

##
R --save < R_input.inc > STDOUT.txt
```

```
# take input from the user
#nterms = as.integer(readline(prompt="How many terms? "))
nterms = 7
# first two terms
n1 = 0
n2 = 1
count = 2
# check if the number of terms is valid
if(nterms <= 0) {
print("Plese enter a positive integer")
} else {
if(nterms == 1) {
print("Fibonacci sequence:")
print(n1)
} else {
print("Fibonacci sequence:")
print(n1)
print(n2)
while(count < nterms) {
nth = n1 + n2
print(nth)
# update values
n1 = n2
n2 = nth
count = count + 1
}
}
}
```

R_input.inc

5. SLURM 예제

4. MNIST 사용 예제

```python
import torch
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
import torch.nn as nn
import random

USE_CUDA = torch.cuda.is_available() # GPU를 사용가능하면 True, 아니라면 False를 리턴
device = torch.device("cuda" if USE_CUDA else "cpu") # GPU 사용 가능하면 사용하고 아니면 CPU 사용
print("다음 기기로 학습합니다:", device)

training_epochs = 15
batch_size = 100

# MNIST dataset
mnist_train = dsets.MNIST(root='data/',
                          train=True,
                          transform=transforms.ToTensor(),
                          download=True)

mnist_test = dsets.MNIST(root='data/',
                         train=False,
                         transform=transforms.ToTensor(),
                         download=True)

# dataset loader
data_loader = DataLoader(dataset=mnist_train,
                         batch_size=batch_size, # 배치 크기는 100
                         shuffle=True,
                         drop_last=True)

model = nn.Linear(784, 10, bias=True).to(device)
```

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1    # Cores per node
#SBATCH --partition=gpu2        # Partition Name (gpu1, cpu1, hgx, gpu2, cpu2)
##
#SBATCH --job-name=MnistExample
#SBATCH -o SLURM.%N.%j.out          # STDOUT
#SBATCH -e SLURM.%N.%j.err          # STDERR
##

hostname
date

module add ANACONDA/2021.05
module add CUDA/11.3.0

python3 mnist_exam.py
```

6. SLURM JOB 으로 jupyter lab 생성

1. ssh config 설정

```
# ~/.ssh/config
Host ubai-gate
  HostName <gate-ip>
  User <user>
  LocalForward <jupyterport> 0.0.0.0:<jupyterport>
```

```
>> ssh ubai-gate
```

<jupyterport>는 사용할 jupyter lab의 port를 의미합니다. 사용자별로 다른 port를 설정해야 충돌을 막을 수 있습니다. 만약 다른 사용자가 사용하는 port를 사용하게 될 경우, slurm job으로 생성된 jupyter server에 정상적으로 접근할 수 없습니다. gate서버의 <jupyterport>를 사용자의 데스크탑에서 접근할 수 있도록 LocalForward를 추가합니다.

6. SLURM JOB 으로 jupyter lab 생성

2. mini conda 설치

```
>> cd ~/.
>> sh Miniconda3-latest-Linux-x86_64.sh
```

LocalForward가 된 상태에서 ubai gate에 접근했다면 이제 minconda 를 설치합니다. 최신 miniconda 설치 파일은 https://docs.conda.io/en/latest/miniconda.html 에서 다운받을 수 있습니다.

6. SLURM JOB 으로 jupyter lab 생성

3. Jupyter 환경 구성

```
>> source activate conda
>> conda create -n jupyter python=3.9
>> conda activate jupyter
>> conda install -c conda-forge jupyterlab
>> pip install ipykernel
>> python -m ipykernel install --user --name jupyter --dis
play-name "jupyter"
>> jupyter lab password
  Enter password:
  Verify password:
```

설치된 miniconda에서 jupyter라는 이름의 가상환경을 생성합니다.

6. SLURM JOB 으로 jupyter lab 생성

3. Jupyter 환경 구성

```
>># pytorch라는 이름의 가상환경 생성
>> conda create -n pytorch python=3.9
>> conda activate pytorch
>> pip install ipykernel
>> python -m ipykernel install --user --name pytorch --dis
play-name "pytorch"
>>
>># tensorflow라는 이름의 가상환경 생성
>> conda create -n tensorflow python=3.9
>> conda activate tensorflow
>> pip install ipykernel
>> python -m ipykernel install --user --name tensorflow --
display-name "tensorflow"
```

만약 여러개의 가상환경을 사용하려면 다음과 같이 추가 환경을 생성할 수 있습니다. 이 가상환경을 ipykernel에 등록하면 jupyer 환경에서 실행된 jupyer server가 이 환경을 사용할 수 있습니다.(환경별로 jupyter를 설치할 필요는 없습니다. ipykernel만 설치합니다.)

6. SLURM JOB 으로 jupyter lab 생성

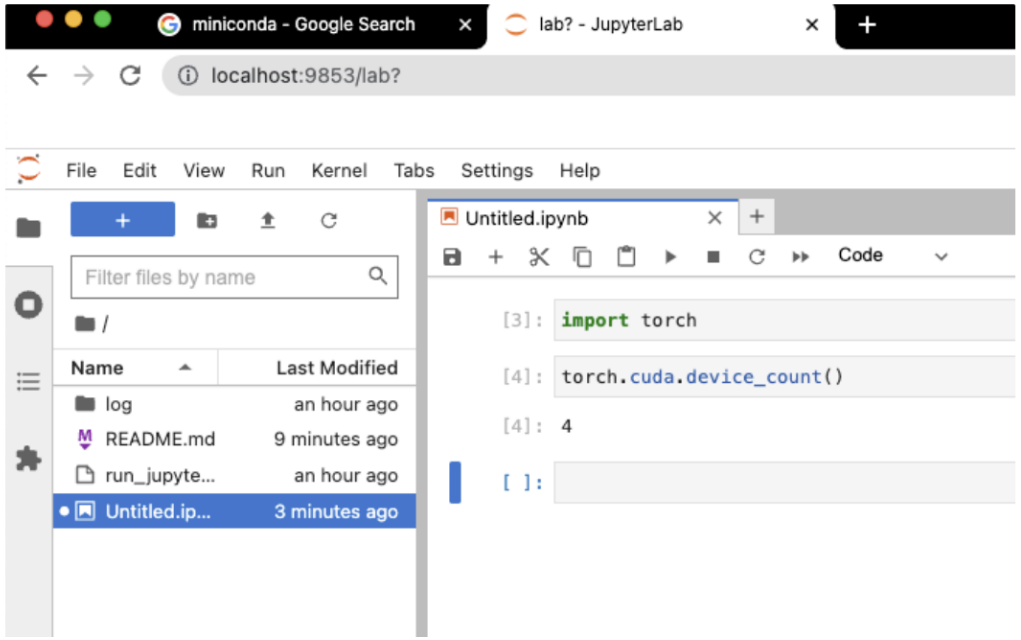
4. slurm job script(run_jupyter_job.sh) 생성

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --cpus-per-task=1
#SBATCH --ntasks-per-node=1    # Cores per node
#SBATCH --partition=gpu1        # Partition Name (gpu, hgx)
##
#SBATCH --job-name=bdg-jupyter
#SBATCH -o ./log/SLURM.%N.%j.out         # STDOUT
#SBATCH -e ./log/SLURM.%N.%j.err         # STDERR
##
#SBATCH --gres=gpu:rtx3090:1

hostname
date

echo 'Remote forwarding from slurm node to gate server in background'
ssh <user-name>@<gateip> -R <jupyterport>:localhost:<jupyterport> -fN "while sle
ep 100; do; done"&

echo 'Start jupyter lab'
/home2/<user-name>/miniconda3/envs/jupyter/bin/python3 -m jupyter lab --ip=0.0.
0.0 --port <jupyterport>

echo 'done'
```

6. SLURM JOB 으로 jupyter lab 생성

5. slurm job 실행

```
>> sbatch run_jupyter_job.sh
```



scancel 을 통해서 job을 삭제하지 않으면 job이 계속 자원을 잡고있기 때문에 사용하지 않을 때는 job을 삭제해야 합니다.

```
>> scancel <jobid>
```

slurm®
workload manager

Job Submission

salloc - Obtain a job allocation.
sbatch - Submit a batch script for later execution.
srun - Obtain a job allocation (as needed) and execute an application.

--array=<indexes> (e.g. "--array=1-10")	Job array specification. (sbatch command only)
--account=<name>	Account to be charged for resources used.
--begin=<time> (e.g. "--begin=18:00:00")	Initiate job after specified time.
--clusters=<name>	Cluster(s) to run the job. (sbatch command only)
--constraint=<features>	Required node features.
--cpu-per-task=<count>	Number of CPUs required per task.
--dependency=<state:jobid>	Defer job until specified jobs reach specified state.
--error=<filename>	File in which to store job error messages.
--exclude=<names>	Specific host names to exclude from job allocation.
--exclusive[=user]	Allocated nodes can not be shared with other jobs/users.
--export=<name[=value]>	Export identified environment variables.
--gres=<name[:count]>	Generic resources required per node.
--input=<name>	File from which to read job input data.
--job-name=<name>	Job name.
--label	Prepend task ID to output. (srun command only)
--licenses=<name[:count]>	License resources required for entire job.
--mem=<MB>	Memory required per node.
--mem-per-cpu=<MB>	Memory required per allocated CPU.
-N<minnodes[-maxnodes]>	Node count required for the job.
-n<count>	Number of tasks to be launched.
--nodelist=<names>	Specific host names to include in job allocation.
--output=<name>	File in which to store job output.
--partition=<names>	Partition/queue in which to run the job.
--qos=<name>	Quality Of Service.
--signal=[B:]<num>[@time]	Signal job when approaching time limit.
--time=<time>	Wall clock time limit.
--wrap=<command_string>	Wrap specified command in a simple "sh" shell. (sbatch command only)

Accounting

sacct - Display accounting data.

--allusers	Displays all users jobs.
--accounts=<name>	Displays jobs with specified accounts.
--endtime=<time>	End of reporting period.
--format=<spec>	Format output.
--name=<jobname>	Display jobs that have any of these name(s).
--partition=<names>	Comma separated list of partitions to select jobs and job steps from.
--state=<state_list>	Display jobs with specified states.
--starttime=<time>	Start of reporting period.

SchedMD
Slurm Support and Development

sacctmgr - View and modify account information.
Options:

--immediate	Commit changes immediately.
--parseable	Output delimited by '|'

Commands:

add <ENTITY> <SPECS> create <ENTITY> <SPECS>	Add an entity. Identical to the **create** command.
delete <*ENTITY*> where <*SPECS*>	Delete the specified entities.
list <ENTITY> [<SPECS>]	Display information about the specific entity.
modify <*ENTITY*> where <*SPECS*> set <*SPECS*>	Modify an entity.

Entities:

account	Account associated with job.
cluster	*ClusterName* parameter in the *slurm.conf*.
qos	Quality of Service.
user	User name in system.

Job Management

sbcast - Transfer file to a job's compute nodes.

sbcast [options] SOURCE DESTINATION

--force	Replace previously existing file.
--preserve	Preserve modification times, access times, and access permissions.

scancel - Signal jobs, job arrays, and/or job steps.

--account=<name>	Operate only on jobs charging the specified account.
--name=<name>	Operate only on jobs with specified name.
--partition=<names>	Operate only on jobs in the specified partition/queue.
--qos=<name>	Operate only on jobs using the specified quality of service.

--reservation=<name>	Operate only on jobs using the specified reservation.
--state=<names>	Operate only on jobs in the specified state.
--user=<name>	Operate only on jobs from the specified user.
--nodelist=<names>	Operate only on jobs using the specified compute nodes.

squeue - View information about jobs.

--account=<name>	View only jobs with specified accounts.
--clusters=<name>	View jobs on specified clusters.
--format=<spec> (e.g. "--format=%i %j")	Output format to display. Specify fields, size, order, etc.
--jobs<job_id_list>	Comma separated list of job IDs to display.
--name=<name>	View only jobs with specified names.
--partition=<names>	View only jobs in specified partitions.
--priority	Sort jobs by priority.
--qos=<name>	View only jobs with specified Qualities Of Service.
--start	Report the expected start time and resources to be allocated for pending jobs in order of increasing start time.
--state=<names>	View only jobs with specified states.
--users=<names>	View only jobs for specified users.

sinfo - View information about nodes and partitions.

--all	Display information about all partitions.
--dead	If set, only report state information for non-responding (dead) nodes.
--format=<spec>	Output format to display.
--iterate=<seconds>	Print the state at specified interval.
--long	Print more detailed information.
--Node	Print information in a node-oriented format.
--partition=<names>	View only specified partitions.
--reservation	Display information about advanced reservations.
-R	Display reasons nodes are in the down, drained, fail or failing state.
--state=<names>	View only nodes specified states.

scontrol - Used view and modify configuration and state. Also see the **sview** graphical user interface version.

--details	Make show command print more details.
--oneliner	Print information on one line.

Commands:

create *SPECIFICATION*	Create a new partition or .
delete *SPECIFICATION*	Delete the entry with the specified SPECIFICATION
reconfigure	All Slurm daemons will re-read the configuration file.
requeue JOB_LIST	Requeue a running, suspended or completed batch job.
show ENTITY ID	Display the state of the specified entity with the specified identification
update *SPECIFICATION*	Update job, step, node, partition, or reservation configuration per the supplied specification.

Environment Variables

SLURM_ARRAY_JOB_ID	Set to the job ID if part of a job array.
SLURM_ARRAY_TASK_ID	Set to the task ID if part of a job array.
SLURM_CLUSTER_NAME	Name of the cluster executing the job.
SLURM_CPUS_PER_TASK	Number of CPUs requested per task.
SLURM_JOB_ACCOUNT	Account name.
SLURM_JOB_ID	Job ID.
SLURM_JOB_NAME	Job Name.
SLURM_JOB_NODELIST	Names of nodes allocated to job.
SLURM_JOB_NUM_NODES	Number of nodes allocated to job.
SLURM_JOB_PARTITION	Partition/queue running the job.
SLURM_JOB_UID	User ID of the job's owner.
SLURM_JOB_USER	User name of the job's owner.
SLURM_RESTART_COUNT	Number of times job has restarted.
SLURM_PROCID	Task ID (MPI rank).
SLURM_STEP_ID	Job step ID.
SLURM_STEP_NUM_TASKS	Task count (number of MPI ranks).

Daemons

slurmctld	Executes on cluster's "head" node to manage workload.
slurmd	Executes on each compute node to locally manage resources.
slurmdbd	Manages database of resources limits, licenses, and archives accounting records.

SchedMD Slurm Support and Development

slurm workload manager

http://www.schedmd.com