

한양생명과학기술원

BI/컴퓨팅 서버 사용자 교육

목차

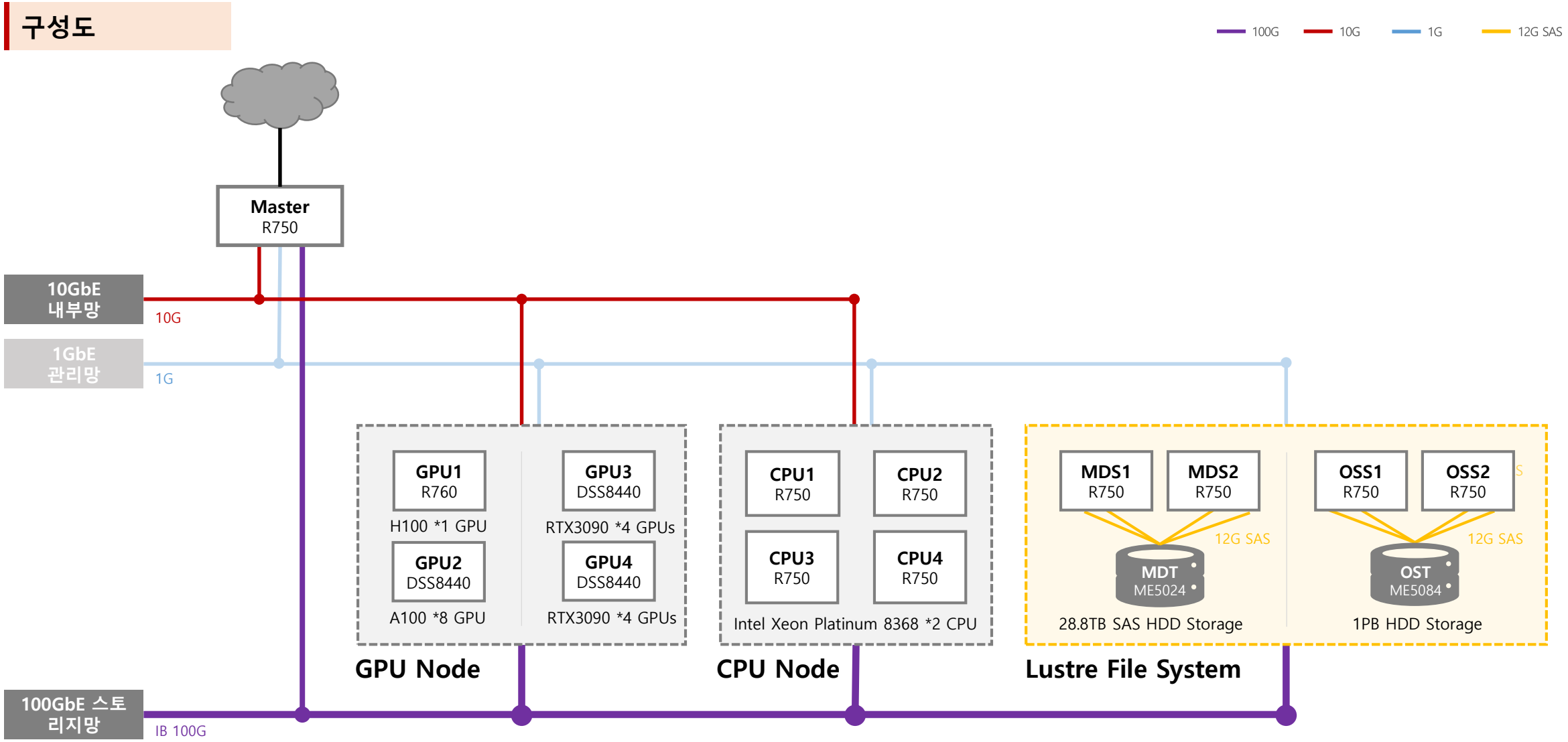
Index

1. BI/컴퓨팅 서버 클러스터 구성
2. 접속 방법
3. 비밀번호 설정
4. 한양 VPN
5. 자원관리자 (Resource Manager)
6. Environment Module
7. Anaconda
8. Jupyter Notebook 과 R studio
9. 실습

1. 클러스터 구성

1. BI/컴퓨팅 서버 클러스터 구성

구성도



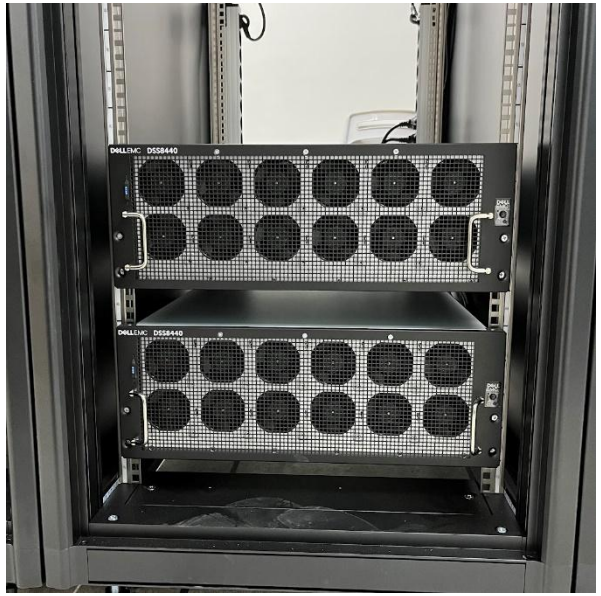
한양생명과학기술원

1. BI/컴퓨팅 서버 클러스터 구성









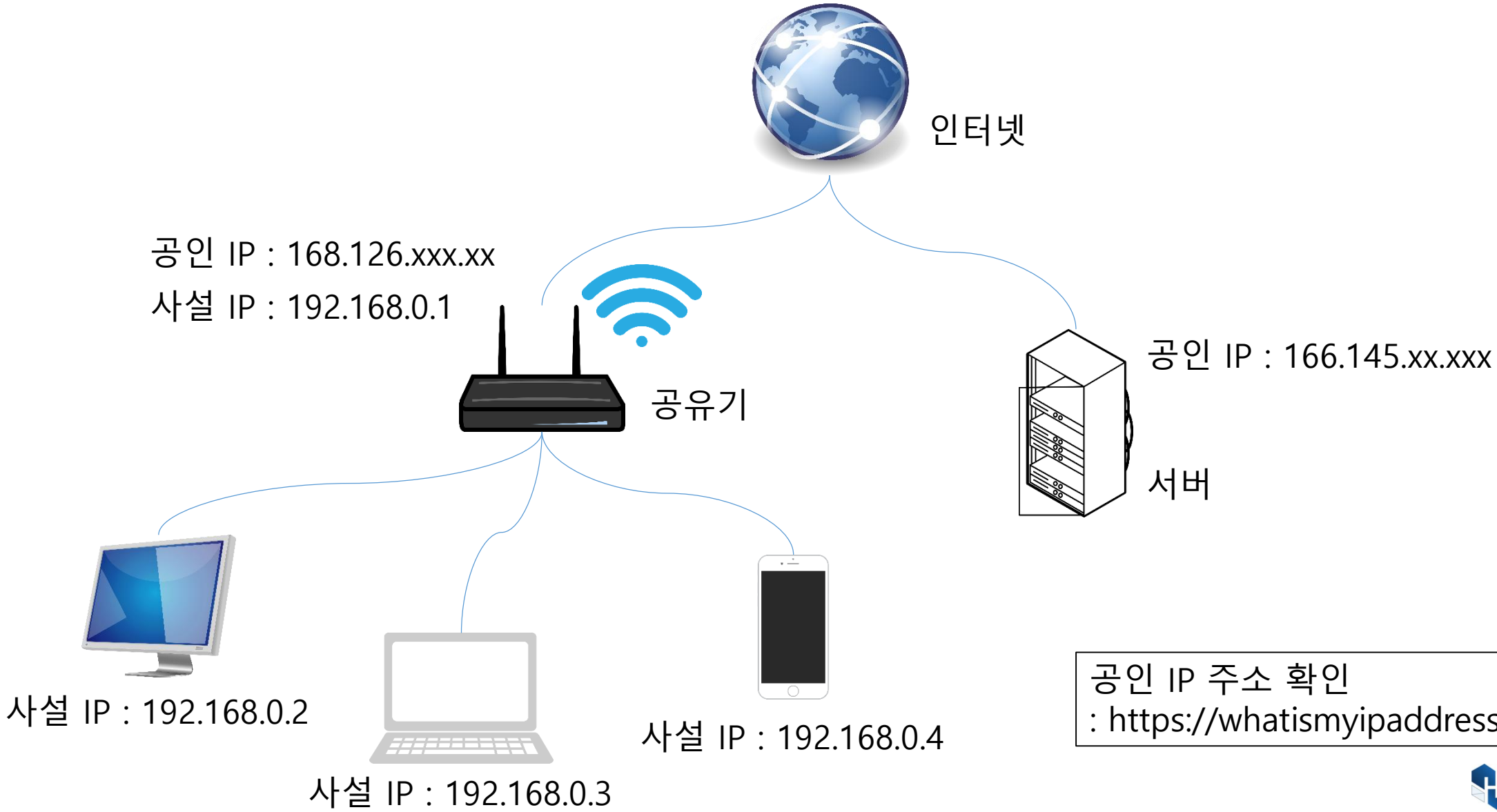




한양생명과학기술원
HY Institute of Bioscience and Biotechnology

2. 접속 방법

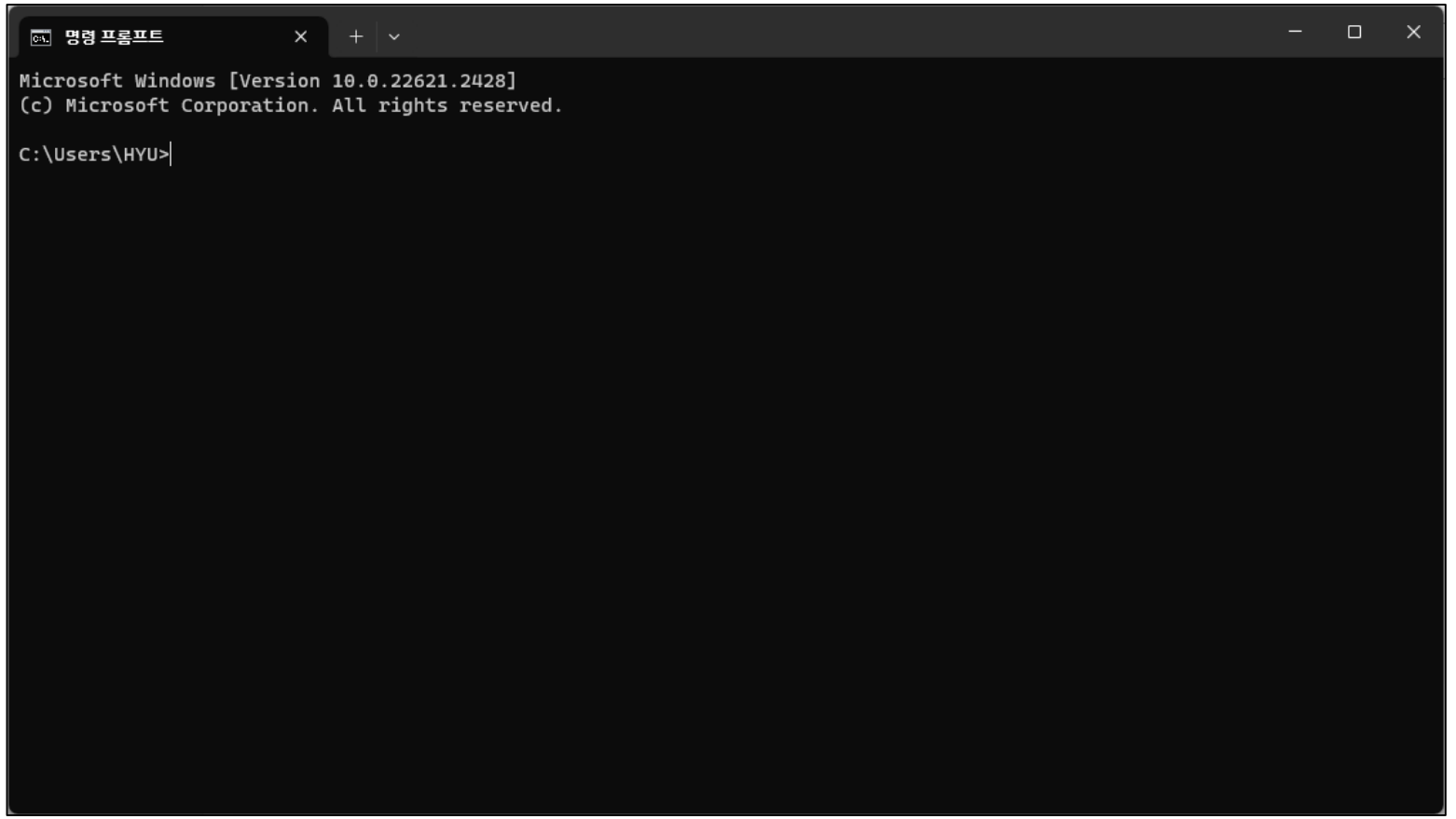
2. 접속 방법



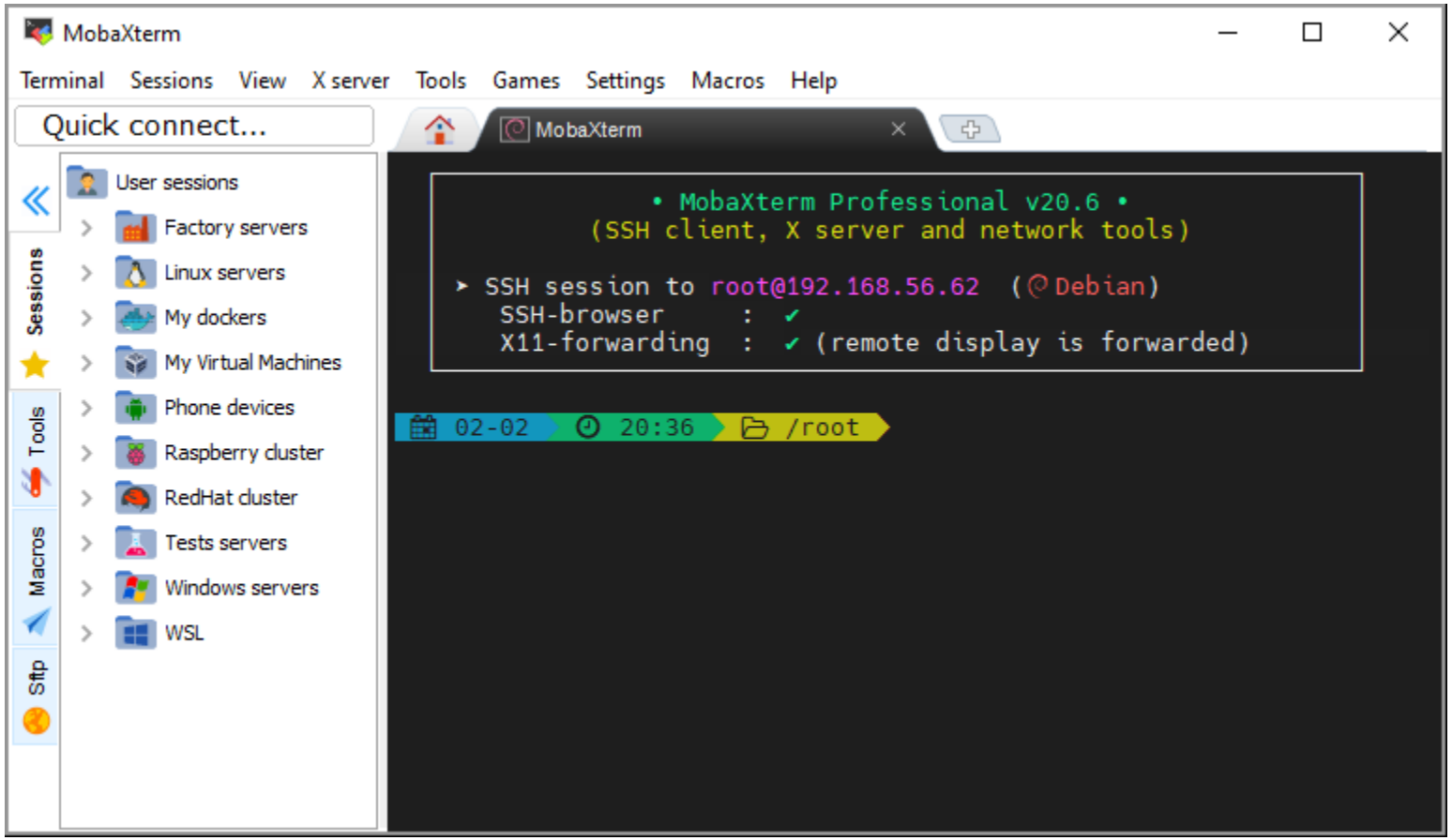
공인 IP 주소 확인
: https://whatismyipaddress.com/

한양생명과학기술원

2. 접속 방법 (1. 기본 터미널)

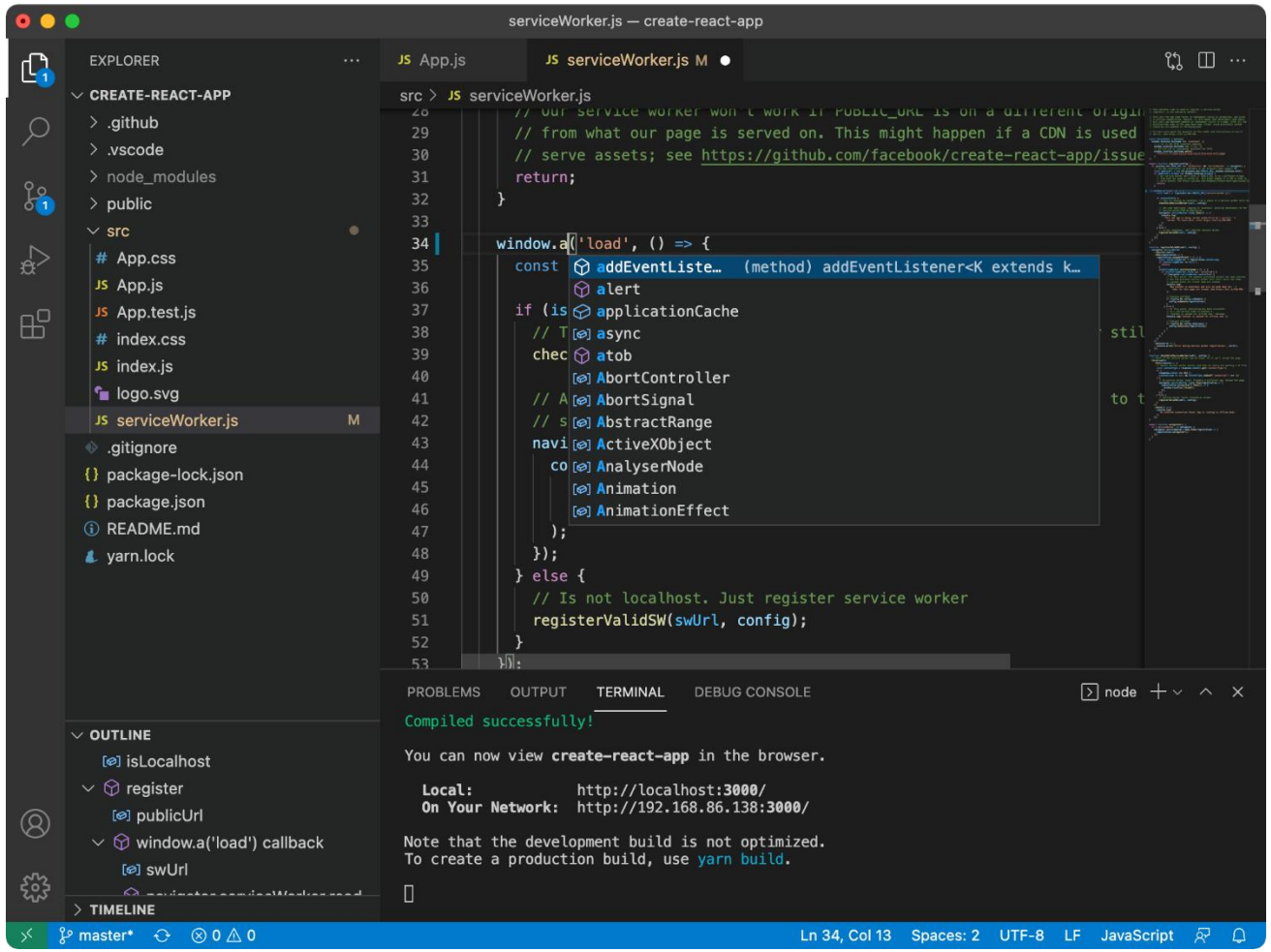


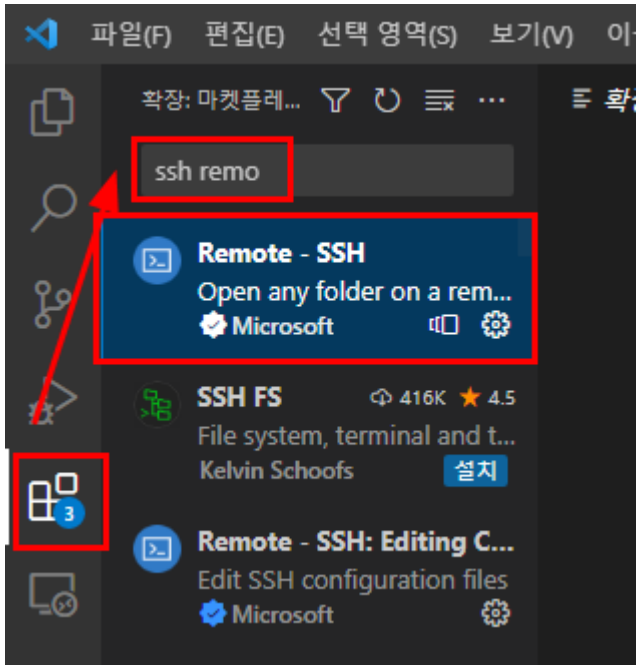
2. 접속 방법 (2. mobaxterm)



https://mobaxterm.mobatek.net

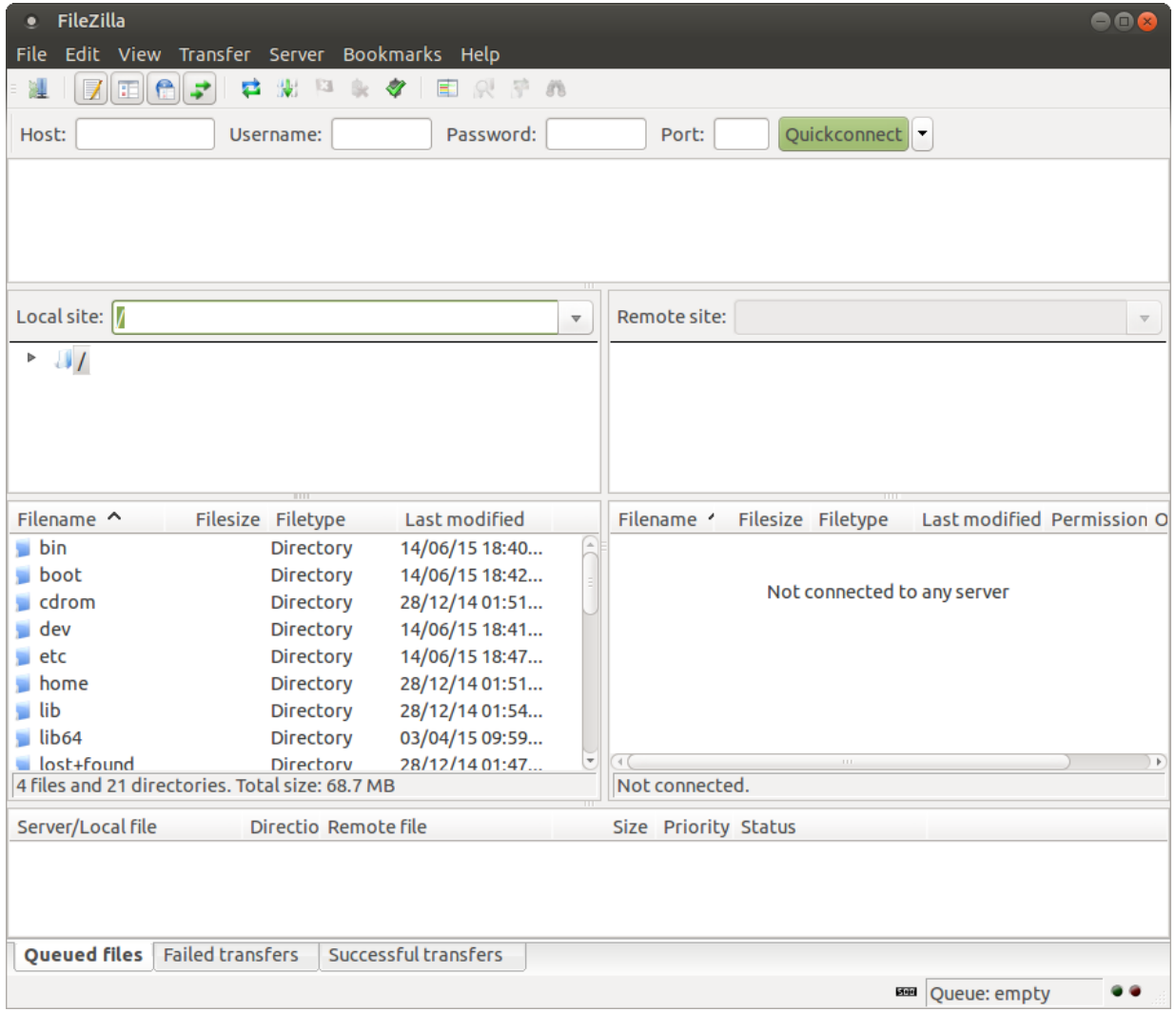
2. 접속 방법 (3. vscode)





https://code.visualstudio.com/

2. 접속 방법 (4. FileZilla)



https://filezilla-project.org/

한양생명과학기술원

3. 비밀번호 설정

3. 비밀번호 설정

※ 비밀번호는 1개 이상의 숫자, 영문, 특수문자가 포함되고 최소 9자리 이상이어야 하며
동일한 유형의 문자는 연속적으로 3회 까지만 입력 가능합니다. (가능 : asd , 135 / 불가 : asdf , 1357)

```
(base) [testuser2@ibb-master ~]$ passwd
Changing password for user testuser2.
Current password:
New password:
BAD PASSWORD: The password contains more than 3 characters of the same class consecutively
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

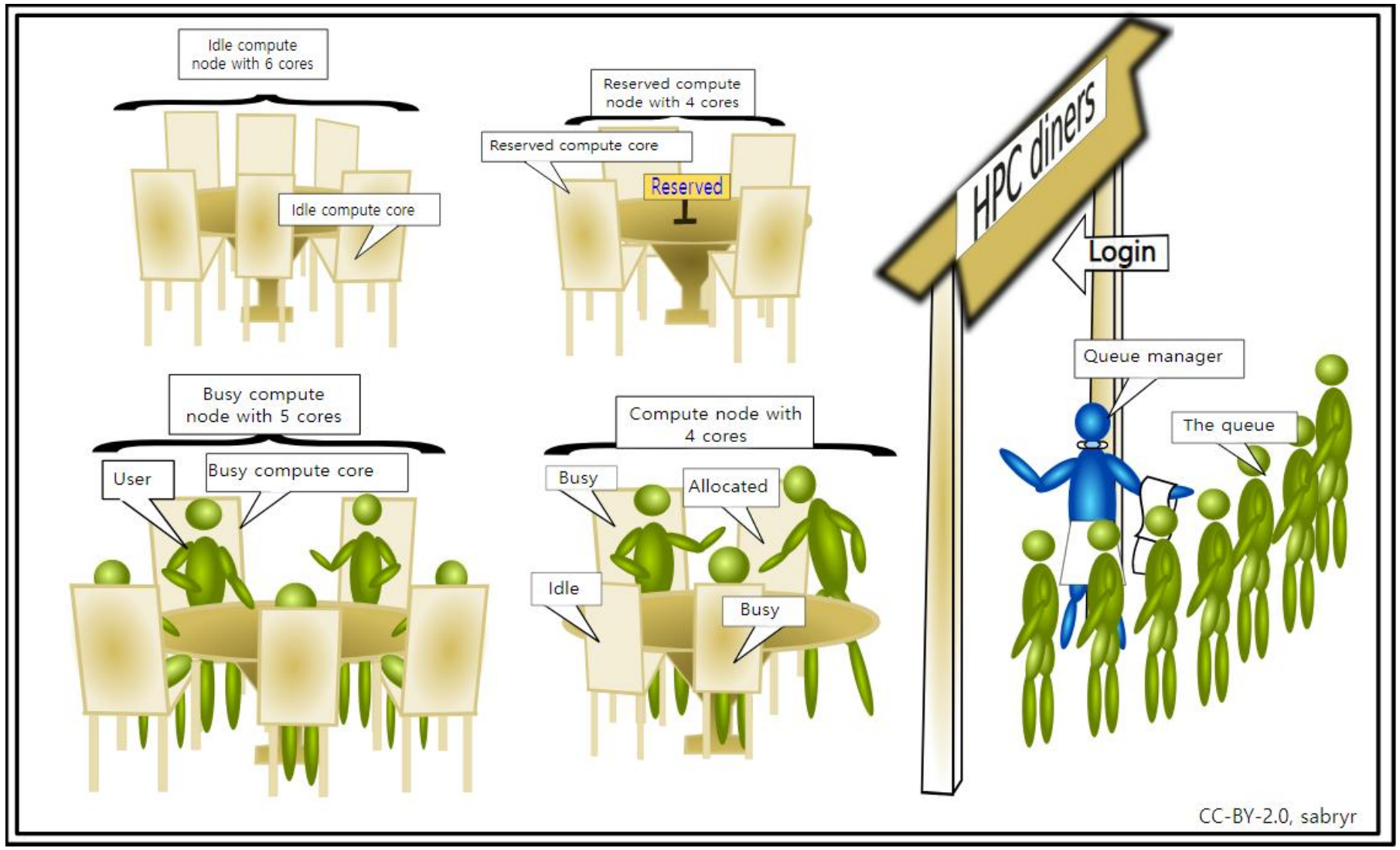
4. 한양 VPN

4. 한양 VPN

- 외부에서의 접근 혹은 교내에서 장소 이동이 잦아 특정 IP 내에서만 서버 접속 하는 것이 어려울 경우 한양 VPN 서비스 사용 권장합니다.
- 한양대학교 포털에서 신청 후 학교 정보인프라팀의 승인을 거쳐 사용 가능합니다.
  (당일 승인 불가, 한학기 단위로 신청 및 연장)
- 보안 및 라이센스 문제로 최대 연결 시간이 4시간으로 고정되어 있습니다.
- 한양 VPN 서비스 안내 페이지 : http://vpninfo.hanyang.ac.kr/

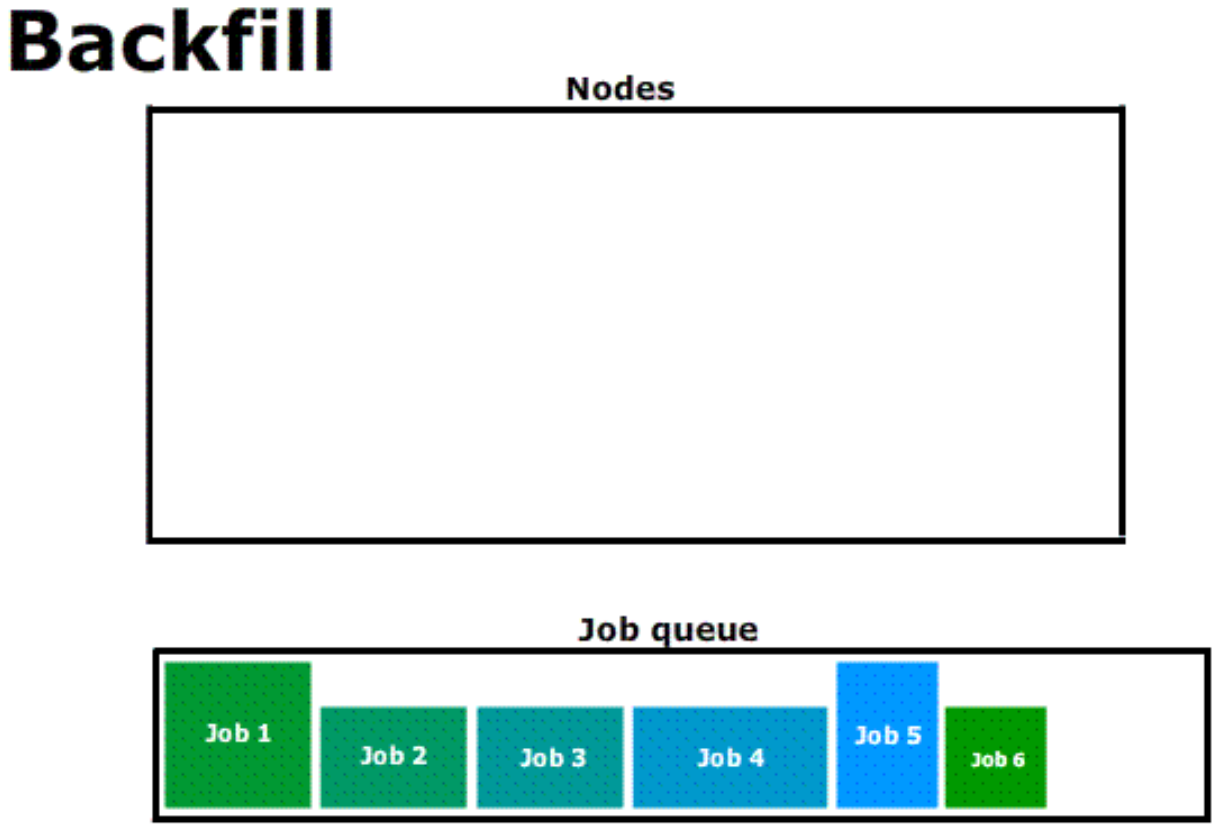
5. 자원관리자 (Resource Manager)

5. 자원관리자 (Resource Manager)



https://sabryr.github.io/hpc-intro/13-scheduler/index.html

5. 자원관리자 (Resource Manager)



http://docs.adaptivecomputing.com/torque/5-0-1/Content/topics/moabWorkloadManager/topics/optimization/backfill.html#overview



SLURM

Simple LinUx Resource Manager

5. SLURM 주요 명령어

```
sinfo                  # 노드의 상태, 사용 가능한 자원 상태
sinfolong

squeue                 # 대기열 (queue) 의 상태, 작업 현황
squeuelong

srun                   # 대화형 (Interactive) 작업 시작
sbatch  batch.sh       # 일괄형 (batch) 작업 제출
scancel <JOB ID>       # 작업 취소

scontrol  show  job  <JOB ID>     # 작업의 자세한 상태 보기
scontrol  show  node <NODE NAME> # 노드의 자세한 상태 보기
```

5. SLURM 주요 명령어

```
(base) [testuser2@ibb-master ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu*         up 7-00:00:00      4   idle node[01-04]
a100         up 7-00:00:00      1   idle gpu01
rtx3090      up 7-00:00:00      1    mix gpu02
rtx3090      up 7-00:00:00      1   idle gpu03
(base) [testuser2@ibb-master ~]$ sinfolong
PARTITION        CPUS(A/I/O/T)  STATE  NODELIST        NODES GRES            TIMELIMIT   AVAIL_FEATURES
cpu*             0/608/0/608    idle   node[01-04]     4     (null)          7-00:00:00  (null)
a100             0/80/0/80      idle   gpu01           1     gpu:a100:8      7-00:00:00  (null)
rtx3090          20/60/0/80     mix    gpu02           1     gpu:rtx3090:4   7-00:00:00  (null)
rtx3090          0/80/0/80      idle   gpu03           1     gpu:rtx3090:4   7-00:00:00  (null)
(base) [testuser2@ibb-master ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1035   rtx3090 sleep.sh testuser  R       0:08      1 gpu02
(base) [testuser2@ibb-master ~]$ squeuelong
JOBID   NAME         STATE     USER          GROUP    PARTITION  QOS      NODE NODELIST(REASON)     CPUS TRES_PER_NODE
1035    sleep.sh     RUNNING   testuser2     testuser rtx3090    rtx3090- 1    gpu02                  20   gres:gpu:rtx3090:1
(base) [testuser2@ibb-master ~]$
```

한양생명과학기술원
HY Institute of Bioscience and Biotechnology

5. SLURM 활용 (1. srun 통해 자원 할당 받기 - CPU자원)

```
(base) [testuser2@ibb-master ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu*         up 7-00:00:00      4   idle node[01-04]
a100         up 7-00:00:00      1   idle gpu01
rtx3090      up 7-00:00:00      2   idle gpu[02-03]
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ srun --qos=cpu-40 --cpus-per-task=5 --pty bash
(base) [testuser2@node01 ~]$
(base) [testuser2@node01 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu*         up 7-00:00:00      1    mix node01
cpu*         up 7-00:00:00      3   idle node[02-04]
a100         up 7-00:00:00      1   idle gpu01
rtx3090      up 7-00:00:00      2   idle gpu[02-03]
(base) [testuser2@node01 ~]$
(base) [testuser2@node01 ~]$ sinfolong
PARTITION        CPUS(A/I/O/T)  STATE  NODELIST        NODES GRES            TIMELIMIT   AVAIL_FEATURES
cpu*             5/147/0/152    mix    node01          1     (null)          7-00:00:00  (null)
cpu*             0/456/0/456    idle   node[02-04]     3     (null)          7-00:00:00  (null)
a100             0/80/0/80      idle   gpu01           1     gpu:a100:8      7-00:00:00  (null)
rtx3090          0/160/0/160    idle   gpu[02-03]      2     gpu:rtx3090:4   7-00:00:00  (null)
(base) [testuser2@node01 ~]$
(base) [testuser2@node01 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1042       cpu     bash testuser  R       0:13      1 node01
(base) [testuser2@node01 ~]$
(base) [testuser2@node01 ~]$ squeuelong
JOBID    NAME         STATE     USER           GROUP    PARTITION  QOS      NODE NODELIST(REASON)     CPUS TRES_PER_NODE
1042     bash         RUNNING   testuser2      testuser cpu        cpu-40   1    node01               5    N/A
(base) [testuser2@node01 ~]$
(base) [testuser2@node01 ~]$ exit
exit
(base) [testuser2@ibb-master ~]$
```

5. SLURM 활용 (1. srun 통해 자원 할당 받기 - GPU자원)

```
(base) [testuser2@ibb-master ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu*         up 7-00:00:00      4   idle node[01-04]
a100         up 7-00:00:00      1   idle gpu01
rtx3090      up 7-00:00:00      2   idle gpu[02-03]
(base) [testuser2@ibb-master ~]$ srun --qos=rtx3090-8 --partition=rtx3090 --gres=gpu:rtx3090:1 --pty bash
(base) [testuser2@gpu02 ~]$
(base) [testuser2@gpu02 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu*         up 7-00:00:00      4   idle node[01-04]
a100         up 7-00:00:00      1   idle gpu01
rtx3090      up 7-00:00:00      1    mix gpu02
rtx3090      up 7-00:00:00      1   idle gpu03
(base) [testuser2@gpu02 ~]$
(base) [testuser2@gpu02 ~]$ sinfolong
PARTITION        CPUS(A/I/O/T)  STATE  NODELIST        NODES GRES            TIMELIMIT   AVAIL_FEATURES
cpu*             0/608/0/608    idle   node[01-04]     4     (null)          7-00:00:00  (null)
a100             0/80/0/80      idle   gpu01           1     gpu:a100:8      7-00:00:00  (null)
rtx3090          20/60/0/80     mix    gpu02           1     gpu:rtx3090:4   7-00:00:00  (null)
rtx3090          0/80/0/80      idle   gpu03           1     gpu:rtx3090:4   7-00:00:00  (null)
(base) [testuser2@gpu02 ~]$
(base) [testuser2@gpu02 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1043   rtx3090     bash testuser  R       0:34      1 gpu02
(base) [testuser2@gpu02 ~]$
(base) [testuser2@gpu02 ~]$ squeuelong
JOBID   NAME         STATE     USER          GROUP    PARTITION  QOS      NODE NODELIST(REASON)     CPUS TRES_PER_NODE
1043    bash         RUNNING   testuser2     testuser rtx3090    rtx3090- 1    gpu02                  20   gres:gpu:rtx3090:1
(base) [testuser2@gpu02 ~]$
(base) [testuser2@gpu02 ~]$ exit
exit
(base) [testuser2@ibb-master ~]$
```

5. SLURM 활용 (2. sbatch 통해 작업 제출하기 - CPU자원)

```
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ sbatch --qos=cpu-80 --time=0-00:10:00 sleep60.sh
Submitted batch job 1452
(base) [testuser2@ibb-master sbatch]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1452       cpu sleep60. testuser  R       0:03      1 node01
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ sinfolong
PARTITION        CPUS(A/I/O/T)  STATE  NODELIST        NODES GRES            TIMELIMIT   AVAIL_FEATURES
cpu*             1/151/0/152    mix    node01          1     (null)          7-00:00:00  (null)
cpu*             0/456/0/456    idle   node[02-04]     3     (null)          7-00:00:00  (null)
a100             0/80/0/80      idle   gpu01           1     gpu:a100:8      7-00:00:00  (null)
rtx3090          0/160/0/160    idle   gpu[02-03]      2     gpu:rtx3090:4   7-00:00:00  (null)
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ sbatch --qos=cpu-80 --cpus-per-task=20 --time=0-00:10:00 sleep60.sh
Submitted batch job 1453
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1453       cpu sleep60. testuser  R       0:02      1 node02
              1452       cpu sleep60. testuser  R       0:32      1 node01
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ sinfolong
PARTITION        CPUS(A/I/O/T)  STATE  NODELIST        NODES GRES            TIMELIMIT   AVAIL_FEATURES
cpu*             21/283/0/304   mix    node[01-02]     2     (null)          7-00:00:00  (null)
cpu*             0/304/0/304    idle   node[03-04]     2     (null)          7-00:00:00  (null)
a100             0/80/0/80      idle   gpu01           1     gpu:a100:8      7-00:00:00  (null)
rtx3090          0/160/0/160    idle   gpu[02-03]      2     gpu:rtx3090:4   7-00:00:00  (null)
(base) [testuser2@ibb-master sbatch]$
```

한양생명과학기술원

5. SLURM 활용 (2. sbatch 통해 작업 제출하기 - GPU자원)

```
(base) [testuser2@ibb-master sbatch]$ sinfolong
PARTITION        CPUS(A/I/O/T)  STATE  NODELIST        NODES GRES            TIMELIMIT   AVAIL_FEATURES
cpu*             0/608/0/608    idle   node[01-04]     4     (null)          7-00:00:00  (null)
a100             0/80/0/80      idle   gpu01           1     gpu:a100:8      7-00:00:00  (null)
rtx3090          0/160/0/160    idle   gpu[02-03]      2     gpu:rtx3090:4   7-00:00:00  (null)
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ sbatch --qos=a100-8 --partition=a100 --gres=gpu:a100:1 --time=10 sleep60.sh
Submitted batch job 1457
(base) [testuser2@ibb-master sbatch]$ sbatch --qos=rtx3090-8 --partition=rtx3090 --gres=gpu:rtx3090:1 --time=10 sleep60.sh
Submitted batch job 1458
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ squeuelong
JOBID   NAME         STATE     USER           GROUP    PARTITION  QOS      NODE NODELIST(REASON)     CPUS TRES_PER_NODE        TIME_LIMIT  TIME_LEFT
1457    sleep60.sh   RUNNING   testuser2      testuser a100       a100-8   1    gpu01                10   gres:gpu:a100:1      10:00       9:37
1458    sleep60.sh   RUNNING   testuser2      testuser rtx3090    rtx3090- 1    gpu02                20   gres:gpu:rtx3090:1   10:00       9:52
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ sinfolong
PARTITION        CPUS(A/I/O/T)  STATE  NODELIST        NODES GRES            TIMELIMIT   AVAIL_FEATURES
cpu*             0/608/0/608    idle   node[01-04]     4     (null)          7-00:00:00  (null)
a100             10/70/0/80     mix    gpu01           1     gpu:a100:8      7-00:00:00  (null)
rtx3090          20/60/0/80     mix    gpu02           1     gpu:rtx3090:4   7-00:00:00  (null)
rtx3090          0/80/0/80      idle   gpu03           1     gpu:rtx3090:4   7-00:00:00  (null)
(base) [testuser2@ibb-master sbatch]$
(base) [testuser2@ibb-master sbatch]$ ls
batch.sh  sleep10.sh  sleep60.sh  slurm-1457.out  slurm-1458.out
(base) [testuser2@ibb-master sbatch]$
```

5. SLURM 활용 (3. 주의사항)

- SLURM을 이용하지 않고 마스터 노드에서 작업을 수행하는 것은 전체 서버에 영향을 미치므로 피하도록 합니다.
- srun 은 작업이 완료되어도 exit 로 나가지 않으면 계속 노드 접근 상태가 유지되므로 대기시간을 줄이기 위해 간단한 디버깅 용도로만 짧게 사용하고 가능하면 sbatch 를 사용하는 것을 권장합니다.
- `sbatch` 작업 제출시 예상되는 최대 시간을 지정하도록 합니다.
- 활성화된 `slurm` 세션을 이용하여 `ssh` 등으로 노드에 직접 접근하는 것은 다른 사용자와의 충돌을 야기할 수 있으므로 금하도록 합니다.
- qos 값은 사용자가 신청한 노드 단위에 따라 다음과 같은 형식으로 설정되어 있습니다.

- CPU 10 threads : cpu-10 / CPU 20 threads : cpu-20 / CPU 30 threads : cpu-30 ...
- GPU A100 1장 : a100-1 / GPU A100 2장 : a100-2 / GPU A100 3장 : a100-3 ...
- GPU RTX3090 1장 : rtx3090-1 / GPU RTX3090 2장 : rtx3090-2 / GPU RTX3090 3장 : rtx3090-3 ...

- .sh 파일의 경우 첫 줄에 ' #!/bin/bash ' 를 입력해주어야 정상적으로 파일내 구문 실행이 가능합니다.

5. SLURM Batch 스크립트

※ sbatch 의 경우 자주 사용하는 옵션들을 쉘 스크립트 파일 내에 넣어주면 매번 반복하여 입력하지 않고 실행이 가능합니다.

```
#!/bin/bash

#SBATCH --job-name=sleep                # Job name
#SBATCH --output=sleep.%j               # Name of stdout output file (%j expands to %jobId)
#SBATCH --qos=cpu-40                    # QoS
#SBATCH --partition=cpu                 # queue or partiton name
#SBATCH --time=0-00:10:00               # Max Run time (day-hh:mm:ss)
#SBATCH --cpus-per-task=10              # number of cpus required per task


date
sleep 60
date
```

```
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ sbatch sleep.sh
sbatch: error: Batch job submission failed: Invalid qos specification
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ sbatch batch.sh
Submitted batch job 1068
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1068       cpu    sleep testuser  R       0:04      1 node01
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ scancel 1068
(base) [testuser2@ibb-master ~]$
```

6. Environment-module

6. Environment-module

```
module  list                    # 현재 load 되어 있는 모듈 목록

module  avail                   # 현재 load 할 수 있는 모듈 목록

module  purge                   # 현재 load 되어 있는 모듈을 모두 unload

module  unload  cuda/11.8               # 특정 모듈만 unload

module  load    cuda/11.8               # 특정 모듈만 load

module  swap    cuda/11.8  cuda/12.0    # 모듈 교체
```

- ~/.bashrc 에 명령어를 넣어서 로그인시 기본 실행 환경 구성.
- Batch 스크립트 파일에 넣어서 실행 환경 구성.

6. Environment-module

```
(base) [testuser2@ibb-master ~]$ module avail

------------------------------------------ /opt/ohpc/pub/modulefiles ------------------------------------------
   AmpliconNoise/1.29       breakdancer/1.4.5            infernal/1.1.4          randfold
   CAP3                     bwa/0.7.15                   intel/2022.2.1          rmblast/2.13.0
   Cytoscape/3.9.1          bwa/0.7.17          (D)      java/11.0.17            rsem/1.3.3
   ETCHING/1.4.2            cdhit/4.8.1                  java/17.0.6      (D)    saintexpress/3.6.3
   EasyBuild/4.6.2          cdk/5.0                      jellyfish/2.2.7         salmon/0.9.1          (D)
   HiC-Pro/3.1.0            charliecloud/0.15            libfabric/1.13.0        salmon/1.4.0
   IGVTools/2.16.0          circos/0.69-9                lobSTR/4.0.6            salmon/1.8.0
   MetaVelvet/1.2.02        clustalw/2.1                 lordec/0.9              sambamba/0.7.1
   R/4.2.2                  cmake/3.24.2                 lumpy-sv/0.2.13         samblaster/0.1.24
   RNAcode/0.3              conda-py39                   manta/1.5.0      (D)    samtools/1.17
   RNAz/2.1.1               cuda/11.8           (L,D)    manta/1.6.0             sickle/1.2
   RepeatMasker/4.1.4       cuda/12.0                    meme/5.3.0              sickle/1.33           (D)
   SOAPdenovo2/2.04         cufflinks/2.2.1              mfold_util/4.7          singularity/3.7.1     (L)
   STAR/2.5.3a              delly/1.1.6                  mirdeep2                snvmix2
   STAR/2.5.4          (D)  exonerate/2.2.0              mothur/1.48.0           somatic-sniper/1.0.5.0
   STAR/2.7.10a             fastqc/0.11.8                muscle/3.8.31           sparsehash/2.0.4
   ViennaRNA/2.5.1          fastx-toolkit/0.0.13         nanocorr/0.1            speedseq/0.1.2
   augustus/2.5.5           gatb/1.4.1                   novobreak/1.1.3rc       sratoolkit/2.9.0      (D)
   augustus/3.5.0      (D)  gatk/4.1.6.0                 ohpc                (L) sratoolkit/3.0.0
   autotools                gatk/4.1.7.0        (D)      os                      stringtie/1.3.4
   bamtools/2.5.1           gnu12/12.2.0                 papi/6.0.0              svaba/1.1.3
   bedtools2/2.29.0    (D)  gnu9/9.4.0                   phast/1.5               sylamer
   bedtools2/2.30.0         gridss/2.10.2                picard/2.27.5           taco/0.7.3
   bismark/0.22.3           gsl/2.7.1                    picard/3.0.0     (D)    tcoffee/13.45
   blasr                    hdf5/1.14.0                  piranha/1.2.1           tophat/2.1.1
   blast/2.13.0+            hisat2/2.2.1                 primer3/2.6.1           trinity/2.6.5
   blat/37                  hmmer/2.3.2                  provean/1.1.5           ucx/1.11.2
   boost/1.81.0             hmmer/3.3           (D)      prun/2.2                valgrind/3.19.0
   bowtie/1.3.1             homer/4.11                   python/3.8.16           vcftools/0.1.16
   bowtie/2.3.4.1      (D)  hwloc/2.7.0                  python/3.11.3    (D)    velvet/1.2.10

  Where:
   D:  Default Module
   L:  Module is loaded

If the avail list is too long consider trying:

"module --default avail" or "ml -d av" to just list the default modules.
"module overview" or "ml ov" to display the number of modules for each name.

Use "module spider" to find all possible modules and extensions.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".


(base) [testuser2@ibb-master ~]$
```

한양생명과학기술원
HY Institute of Bioscience and Biotechnology

6. Environment-module

```
(base) [testuser2@ibb-master ~]$ module list

Currently Loaded Modules:
  1) cuda/11.8   2) singularity/3.7.1   3) ohpc



(base) [testuser2@ibb-master ~]$ module load bowtie
(base) [testuser2@ibb-master ~]$ module list

Currently Loaded Modules:
  1) cuda/11.8   2) singularity/3.7.1   3) ohpc   4) bowtie/2.3.4.1



(base) [testuser2@ibb-master ~]$ module swap cuda/11.8 cuda/12.0

The following have been reloaded with a version change:
  1) cuda/11.8 => cuda/12.0

(base) [testuser2@ibb-master ~]$ module list

Currently Loaded Modules:
  1) singularity/3.7.1   2) ohpc   3) bowtie/2.3.4.1   4) cuda/12.0



(base) [testuser2@ibb-master ~]$ module unload bowtie
(base) [testuser2@ibb-master ~]$ module list

Currently Loaded Modules:
  1) singularity/3.7.1   2) ohpc   3) cuda/12.0



(base) [testuser2@ibb-master ~]$ module purge
(base) [testuser2@ibb-master ~]$ module list
No modules loaded
(base) [testuser2@ibb-master ~]$
```

7. Anaconda

```
conda create -n bio python=3.7      # 새로운 가상환경 생성

conda remove -n bio --all           # 가상환경 삭제

conda env list                      # 가상환경 목록 출력

conda activate bio                  # 가상환경 활성화

conda deactivate                    # 사용중인 가상환경 비활성화

conda list          # 활성화된 가상환경에 설치된 패키지 목록 출력
```

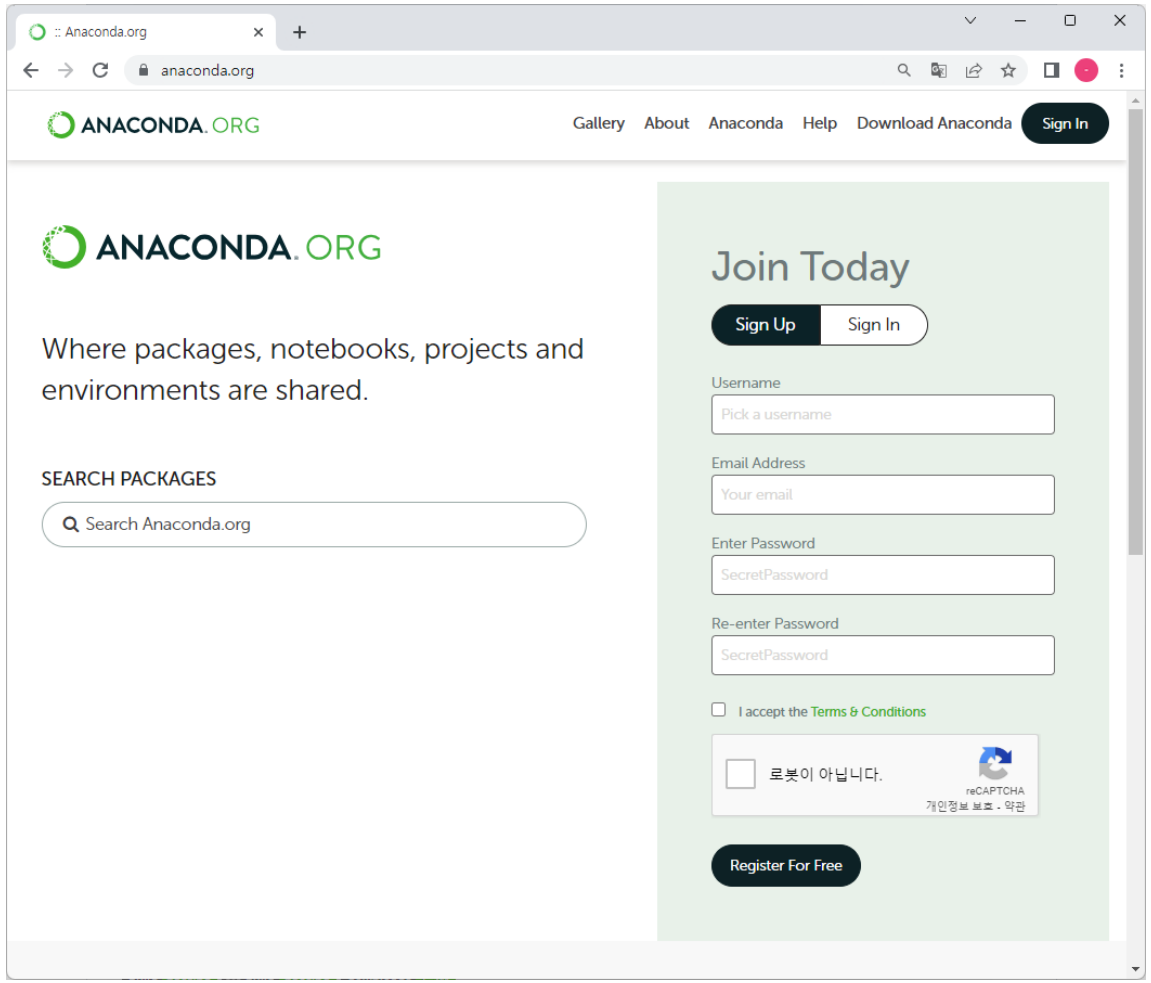
- ~/.bashrc 에 명령어를 넣어서 로그인시 기본 실행 환경 구성.
- Batch 스크립트 파일에 넣어서 실행 환경 구성. (source .bashrc)

7. Anaconda

```
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ conda create -n bio python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/testuser2/.conda/envs/bio

  added / updated specs:
    - python=3.7


The following packages will be downloaded:
```

```
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ conda env list
# conda environments:
#
bio                      /home/testuser2/.conda/envs/bio
base                  *  /opt/ohpc/pub/apps/anaconda3

(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ conda activate bio
(bio) [testuser2@ibb-master ~]$
(bio) [testuser2@ibb-master ~]$ conda deactivate
(base) [testuser2@ibb-master ~]$
(base) [testuser2@ibb-master ~]$ conda deactivate
[testuser2@ibb-master ~]$
[testuser2@ibb-master ~]$
```

7. Anaconda



https://anaconda.org

8. Jupyter Notebook 과 R studio

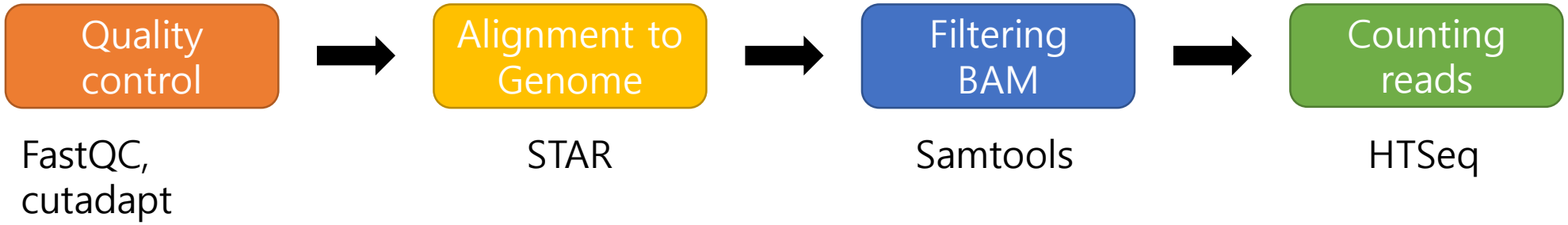
9. 실습

*** RNA-seq_analysis**

GSM7152249(SRR24086621) : 폐동맥 고혈압 환자 혈소판 세포의 RNA-seq 데이터

GEO DataSets - NCBI : https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM7152249
EBI Search - EMBL-EBI : https://www.ebi.ac.uk/ena/browser/view/SRR24086621

Module : fastqc, samtools, STAR/2.7.10a
conda package : cutadapt, HTSeq



※ 한양생명과학기술원 홈페이지 > 바이오공동기기센터 > 공지사항 > BI/컴퓨팅 서버 사용자 교육자료
(BI/컴퓨팅 서버 자율사용자 이용가이드)

9. 실습

1) 실습 계정으로 서버에 접속합니다. (인터넷은 HY-IBB 로 시작하는 Wi-Fi 에 연결)

```
$ ssh -p [포트번호] testuser2@[서버IP주소]
```

2) 분석에 사용할 패키지 중 Environment-module에 설치되어 있는 패키지를 불러옵니다.

```
$ module load  fastqc  samtools  STAR/2.7.10a
$ module list
```

(현재 활성화된 모듈 목록을 출력하여 3가지 패키지가 load 되었는지 확인)

3) srun 명령어를 통해 CPU Node에 접속합니다.

```
$ srun --qos=cpu-100 --cpus-per-task=10 --pty bash
```

(testuser2@ 뒷부분이 ibb-master 에서 node01 과 같이 변경됩니다.)

9. 실습

4) 본인이 사용할 Anaconda 가상환경을 생성합니다.

$ conda create -n [가상환경명] python=3.7
(python 버전 옵션을 생략하면 최신버전으로 설치됩니다.)

5) 생성한 가상환경을 활성화합니다.

$ conda activate [가상환경명]

6) 필요한 패키지를 설치합니다. ('conda install [패키지명]' 으로 검색)

$ conda install bioconda::cutadapt bioconda::htseq
(설치 완료 후 cutadapt 와 htseq-count 를 각각 입력했을 때,
'command not found...' 가 아닌 다른 에러 메시지가 뜨면 설치가 잘 완료된 것입니다.)

9. 실습

7) 개별 분석을 위해 샘플 폴더를 복사하여 2개의 폴더를 생성합니다.

① 샘플 폴더가 있는 경로로 이동합니다.

```
$ cd /home/testuser2/user_training_sample/training
```

② 샘플 폴더를 두 번 복사하고 이름을 구분하여 표시합니다.

```
$ cp -r SRR24086621 [name]_srun
$ cp -r SRR24086621 [name]_sbatch
```

(구분 가능하도록 [name] 부분에 임의의 폴더명을 입력합니다.)

8) 복사한 폴더 중 하나로 이동한 후 sbatch 명령어를 통해 분석 스크립트를 실행합니다.

```
$ cd /home/testuser2/user_training_sample/training/[name]_sbatch
$ sbatch --qos=cpu-100 --cpus-per-task=5 --time=60 RNA-seq_pipieline.sh
```

9) 복사해둔 다른 폴더로 이동합니다.

```
$ cd /home/testuser2/user_training_sample/training/[name]_srun
```

10) 이후 명령어를 차례로 실행하여 분석을 진행합니다.

Quality control

```
$ TrimGalore-0.6.6/trim_galore --illumina -o fastqc_trimmed_results SRR24086621.fastq.gz
```

Alignment to Genome

```
$ STAR --readFilesCommand zcat ₩
--genomeDir /home/shared/Reference_genome/GENCODE/STAR_Homo_sapiens_hg38 ₩
--readFilesIn fastqc_trimmed_results/SRR24086621_trimmed.fq.gz --runThreadN 10 ₩
--outFileNamePrefix output/SRR24086621_
```

Quality control

```
@SRR24086621.1 1/1
NCCCTATTTTCTGTTCAGTTTTTTAAAAGTGTTGTAAGAGACTCCTGAGGAAGACACACAGTTTATTTGTAAAGTAGCTCTGTGAGAGCCACCATGCCCTC
+
#≤BBGGGGGCDGGGGEGGGGGGGGGGGFGGGGGGGGGG>FGGGGGGGGGGGG>GGGGGGGGGGGGGGGGGGGGGGGGGGGGEGGGGGGGGGGGECG
@SRR24086621.2 2/1
NGAGACAGCTGAACCCTCGTGGAGCCATTCATACAGGTCCCTAATTAAGGAACAAGTGATTATGCTACCTTTGCACGGTTAGGGTACCGCGGCCGTTAAAC
+
#=::A@1;=FDGGGDGDFBGG>FBGGCGGGGGGGGG>FFGGGGGGGG:FC@FGGD@GGFGGGGGCGGFFGGGGGGGGG@FGGGGGEGFDGGGGGGGG<@GC
@SRR24086621.3 3/1
NAGCCAGATTCGGGCGCTCCCATCTTGCGCAAGTTGGTCACGTGGTCACCCAATTCTTTGATGGCTTTCACCTGCTCATTCAGGTAATGTGTCTCAATGAA
+
#=ABBGGGGFGGGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
@SRR24086621.4 4/1
NTTGGTAATAGTGTAGAAGTGATCTGGTTCTTACAATGGGAGATGAAGAACATTTGTTATTGGGTTACTACTAACCCTGTCCCAAGAATAGTAATATCACC
+
#=ABBGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGFGGGGBGBG=FGGGFGGGGGG
@SRR24086621.5 5/1
NAGTATGGCCTTTTCTGGGAAGGTGACCCTCCTTGGCCACCCCTTGCATATACTTTGATGCCCTAGGGCACACCCCCTTTATTTCCCTCATAGAAACAGCC
+
#<ABBGGGGGGGGGGGGGGEGGGGGEGGGGGGGGGGGGGGGGGGGGGGGGGEGGGEGGGGGGFGGGGGGBGEGGGGGGGCGEGGEGCGGBF>FGGGGBCB
@SRR24086621.6 6/1
NAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+
#=ABBGGGGGGGGGGGGGGGGGGGGGGGGGGDGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
```

SRR24086621.fastq.gz

Quality control

```
=== Summary ===

Total reads processed:                8,569,377
Reads with adapters:                  2,901,441 (33.9%)
Reads written (passing filters):      8,569,377 (100.0%)

Total basepairs processed:   865,507,077 bp
Quality-trimmed:              20,467,627 bp (2.4%)
Total written (filtered):    841,084,566 bp (97.2%)

=== Adapter 1 ===

Sequence: AGATCGGAAGAGC; Type: regular 3'; Length: 13; Trimmed: 2901441 times.

No. of allowed errors:
0-9 bp: 0; 10-13 bp: 1

Bases preceding removed adapters:
  A: 41.6%
  C: 22.7%
  G: 14.6%
  T: 21.0%
  none/other: 0.1%

Overview of removed sequences
length  count    expect  max.err error counts
1       2208957 2142344.2       0       2208957
2       511433  535586.1        0       511433
3       135461  133896.5        0       135461
4       33019   33474.1 0       33019
5       6242    8368.5  0       6242
6       1391    2092.1  0       1391
7       526     523.0   0       526
8       394     130.8   0       394
```

SRR24086621.fastq.gz_trimming_report.txt

Alignment to Genome

```
                                 Started job on |       Aug 23 18:04:52
                             Started mapping on |       Aug 23 18:05:16
                                    Finished on |       Aug 23 18:06:07
           Mapping speed, Million of reads per hour |       604.31

                               Number of input reads |       8561039
                        Average input read length |       98
                                      UNIQUE READS:
                     Uniquely mapped reads number |       7284633
                          Uniquely mapped reads % |       85.09%
                            Average mapped length |       99.16
                         Number of splices: Total |       589250
              Number of splices: Annotated (sjdb) |       431821
                         Number of splices: GT/AG |       582050
                         Number of splices: GC/AG |       2445
                         Number of splices: AT/AC |       331
                 Number of splices: Non-canonical |       4424
                        Mismatch rate per base, % |       0.43%
                           Deletion rate per base |       0.02%
                          Deletion average length |       1.21
                          Insertion rate per base |       0.01%
                         Insertion average length |       1.32
                               MULTI-MAPPING READS:
            Number of reads mapped to multiple loci |       688789
                 % of reads mapped to multiple loci |       8.05%
            Number of reads mapped to too many loci |       26031
                 % of reads mapped to too many loci |       0.30%
                                   UNMAPPED READS:
       Number of reads unmapped: too many mismatches |       0
            % of reads unmapped: too many mismatches |       0.00%
                 Number of reads unmapped: too short |       547588
                      % of reads unmapped: too short |       6.40%
                     Number of reads unmapped: other |       13998
                          % of reads unmapped: other |       0.16%
                                         CHIMERIC READS:
                              Number of chimeric reads |       0
                                   % of chimeric reads |       0.00%
```

SRR24086621_Log.final.out

Filtering BAM

1. Changing file format from SAM to BAM
$ samtools view -bS output/SRR24086621_Aligned.out.sam ₩
> output/SRR24086621_Aligned.out.sam.bam

2. Sorting BAM
$ samtools sort -o output/SRR24086621_Aligned.out.sam_sort.bam ₩
output/SRR24086621_Aligned.out.sam.bam

3. Indexing BAM
$ samtools index output/SRR24086621_Aligned.out.sam_sort.bam

Counting reads

```
$ htseq-count -s no -t exon -f bam output/SRR24086621_Aligned.out.sam_sort.bam ₩
/home/shared/Reference_genome/GENCODE/GENCODE_Homo_sapiens_hg38/GRCh38.gencode.v43.basic.annotation.gtf ₩
> htseq/SRR24086621_htseq_count.out 2> htseq/SRR24086621_htseq.err
```

```
ENSG00000000003.15      2
ENSG00000000005.6       1
ENSG00000000419.14      4
ENSG00000000457.14      28
ENSG00000000460.17      13
ENSG00000000938.13      16
ENSG00000000971.17      0
ENSG00000001036.14      56
ENSG00000001084.13      9
ENSG00000001167.15      0
ENSG00000001460.18      8
ENSG00000001461.17      11
```

SRR24086621_htseq_count.out

Q&A

감사합니다.