Bioinformatics Lab: Introduction of High Performance Computing

Ivan Gesteira Costa & Zhijian Li
Institute for Computational Genomics

Institute for Computational Genomics
RWTH AACHEN UNIVERSITY

Login

You have to use the *secure shell protocol* (**ssh**) to log in

```
$ ssh <username>@cluster.rz.rwth-aachen.de
```

Login

You have to use the *secure shell protocol* (**ssh**) to log in

```
$ ssh <username>@cluster.rz.rwth-aachen.de
```

Front end name	*OS*
cluster.rz.rwth-aachen.de ***cluster-linux.rz.rwth-aachen.de***	***Linux***
cluster-x.rz.rwth-aachen.de ***cluster-x2.rz.rwth-aachen.de***	***Linux for graphical login***
cluster-copy.rz.rwth-aachen.de ***cluster-copy2.rz.rwth-aachen.de***	***Linux for data transfers***

Login

You have to use the *secure shell protocol* (**ssh**) to log in

```
$ ssh <username>@cluster.rz.rwth-aachen.de
```

Front end name	*OS*
cluster.rz.rwth-aachen.de ***cluster-linux.rz.rwth-aachen.de***	***Linux***
cluster-x.rz.rwth-aachen.de ***cluster-x2.rz.rwth-aachen.de***	***Linux for graphical login***
cluster-copy.rz.rwth-aachen.de ***cluster-copy2.rz.rwth-aachen.de***	***Linux for data transfers***

Basic commands to move around and look at things

Run *pwd* to find out where we are:

```
$ pwd
```

Basic commands to move around and look at things

Run *pwd* to find out where we are:

```
$ pwd
```

See what’s in our current directory:

```
$ ls
```

Basic commands to move around and look at things

Run *pwd* to find out where we are:

```
$ pwd
```

See what’s in our current directory:

```
$ ls
```

Make a new directory for us:

```
$ mkdir documents
```

Basic commands to move around and look at things

Run *pwd* to find out where we are:

```
$ pwd
```

See what’s in our current directory:

```
$ ls
```

Make a new directory for us:

```
$ mkdir documents
```

Go inside it:

```
$ cd documents
```

Available file systems

File System	PATH	Backup	Quota (space)	Quota (#files)
$HOME	*/home/<username>*	*yes*	*150 GB*	*unlimited*
$WORK	*/work/<username>*	*no*	*250 GB*	*1.000.000*
$HPCWORK	*/hpcwork/<username>*	*no*	*1 TB*	*50.000*

Available file systems

File System	PATH	Backup	Quota (space)	Quota (#files)
$HOME	*/home/<username>*	*yes*	*150 GB*	*unlimited*
$WORK	*/work/<username>*	*no*	*250 GB*	*1.000.000*
$HPCWORK	*/hpcwork/<username>*	*no*	*1 TB*	*50.000*

Source code, configuration files

Available file systems

File System	PATH	Backup	Quota (space)	Quota (#files)
$HOME	*/home/<username>*	*yes*	*150 GB*	*unlimited*
$WORK	*/work/<username>*	*no*	*250 GB*	*1.000.000*
$HPCWORK	*/hpcwork/<username>*	*no*	*1 TB*	*50.000*

Source code, configuration files

Output files, working data

Available file systems

File System	PATH	Backup	Quota (space)	Quota (#files)
$HOME	/home/<username>	yes	150 GB	unlimited
$WORK	/work/<username>	no	250 GB	1.000.000
$HPCWORK	/hpcwork/<username>	no	1 TB	50.000

Source code, configuration files

Output files, working data

IO intensive jobs, large files

Data Access

Using *sshfs* command to mount remote directory to you local machine:

```
$ sshfs -o idmap=user <your tim>@cluster-linux.rz.rwth-aachen.de:PATH
<your local folder>
```

Data Access

Using *sshfs* command to mount remote directory to you local machine:

```
$ sshfs -o idmap=user <your tim>@cluster-linux.rz.rwth-aachen.de:PATH
<your local folder>
```

Example:

```
sshfs -o idmap=user rs619065@cluster-linux.rz.rwth-
aachen.de:/home/rs619065 ~/Cluster/home
```

Using the batch system

Using *bsub* command to submit jobs to the batch system:

```
$ bsub [options] command [arguments]
```

Using the batch system

Using below parameters to specify output information:

Parameter	***Function***
-J <name>	***Job name***
-o <path>	***Standard output (and error if option -e <path> is not used)***
-e <path>	***Standard error***

Using the batch system

Parameters for requiring job resources:

Parameter	*Function*
-W <runlimit>	***Set the runtime limit in format [hours:]minutes***
-M <memlimit>	***Set the per-process memory limit in MB***

Using the batch system

Using a bath script to specify job requirements:

```
$ bsub < script.zsh
```

Using the batch system

Using a bath script to specify job requirements:

```
$ bsub < script.zsh
```

```
################################################################
export PATH=~/local/bin:$PATH
export PATH=~/.local/bin:$PATH

export PYTHONPATH=$PYTHONPATH:~/.local/lib/python2.7/site-packages
export PYTHONPATH=$PYTHONPATH:~/local/lib/python2.7/site-packages
```

Using the batch system

```
## Job name
#BSUB -J PeakCalling

### File / path where STDOUT & STDERR will be written
#BSUB -o PeakCalling

### Request the time you need for execution in minutes
### The format for the parameter is: [hour:]minute,
### that means for 80 minutes you could also use this: 1:20
#BSUB -W 5:00

### Request memory you need for your job in TOTAL in MB
#BSUB -M 10240

### Make sure the jobs can use the HPCWORK file system
#BSUB -R "select[hpcwork]"

### Assign the job to the specified project
#BSUB -P lect0023
```

Using the batch system

```
mkdir -p ~/Peaks

target=/hpcwork/lect0023/bioinfolab_2018/Data/ATAC-seq/H1hesc/ATAC.bam

macs2 callpeak -t $target -n H1hesc --outdir ~/Peaks -g hs
```

Using the batch system

Using *bjobs* to monitor jobs:

```
$ bjobs
```

Using the batch system

Using *bjobs* to monitor jobs:

```
$ bjobs
```

Two useful options:

Option	*Description*
-r	***Displays running jobs***
-p	***Displays pending job and the pending reason***

Using the batch system

Using *bkill* to delete jobs:

```
$ bkill [job_id]
```

Using the batch system

Using *bkill* to delete jobs:

```
$ bkill [job_id]
```

If you would like to kill all submitted jobs:

```
$ bkill 0
```

More information

https://doc.itc.rwth-aachen.de/display/CC/Using+the+batch+system

https://doc.itc.rwth-aachen.de/pages/viewpage.action?pageId=3473705