Help yourself on a remote server

bioinformatics@age.mpg.de

https://mpg-age-bioinformatics.github.io

Outline

- Connecting to a remote server
- Copying files to/from a remote server
- The modules system
- Installing software without `su` access
- Installing R packages
- Installing python packages
- Installing perl packages
- SLURM

Connecting to a remote server

Connecting to a remote server over a Secure Shell (ssh) – ssh username@remote.adress :

```
ssh JBoucas@amalia.age.mpg.de
```

With X forwarding:

```
ssh –X JBoucas@amalia.age.mpg.de
```

Copying files to/from a remote server

Copying files over ssh to your home folder on a remote server:

```
scp file.txt UName@ServerAddress:~/
```

Copying files over ssh from your home folder on a remote server:

```
scp UName@ServerAddress:~/file.txt .
```

Both `scp` will only allow you to copy files (not directories) unless you use the `-r` argument for `recursively`. For speed use `-o Cipher=arcfour`.

rsync to a remote server

```
rsync -rtvh -e "ssh -c arcfour" source_folder \
UName@ServerAddress:destination
```

Not all servers will have the less costly cipher arcfour encryption algorithm and therefore you can remove the whole `-e "ssh -c arcfour"` block.

The modules system

A centralized software system.

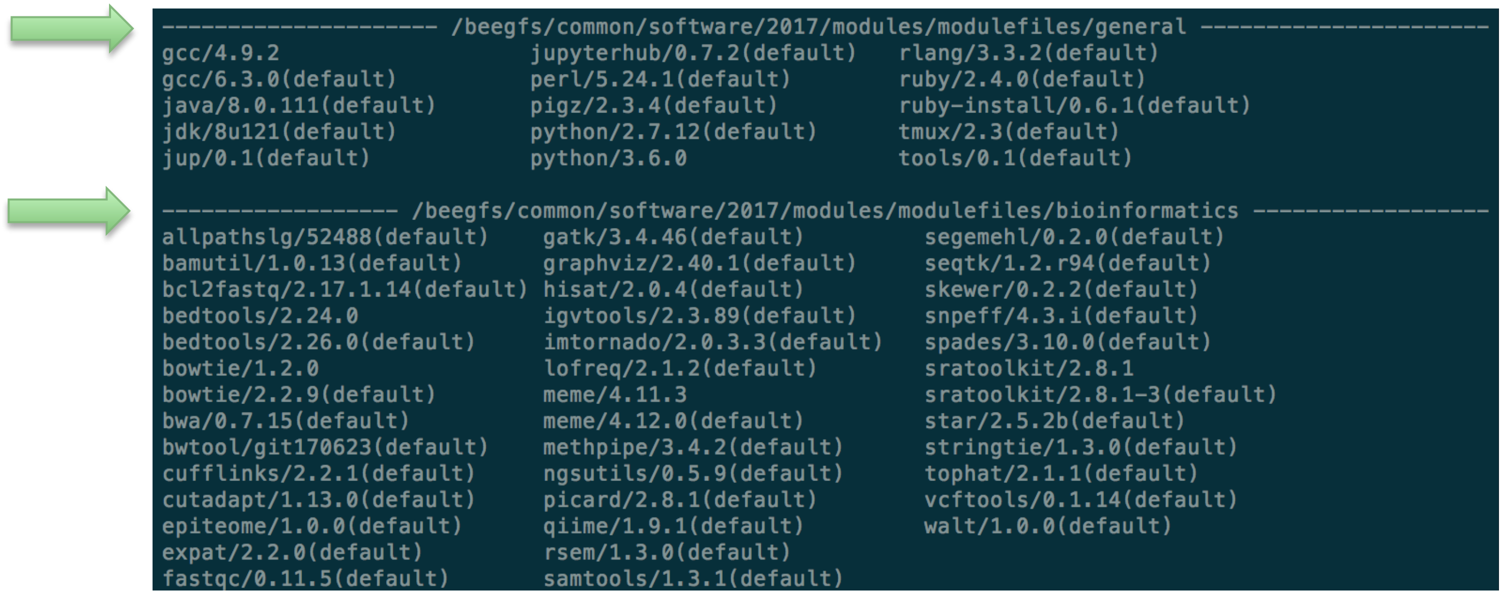
The modules system **loads software** (version of choice) and changes **environment variables** (eg. LD_LIBRARY_PATH)

```
module avail             # shows available modules
module whatis SAMtools   # shows a description of the SAMtools module
module show SAMtools     # shows environment changes for SAMtools
module load SAMtools     # loads SAMtools
module list              # lists all loaded modules
module unload SAMtools   # unloads the SAMtools module
module purge             # unloads all loaded modules
```

more on http://modules.sourceforge.net

module avail

Example `module avail` output.



```
---------------------- /beegfs/common/software/2017/modules/modulefiles/general ----------------------
gcc/4.9.2                   jupyterhub/0.7.2(default)   rlang/3.3.2(default)
gcc/6.3.0(default)          perl/5.24.1(default)        ruby/2.4.0(default)
java/8.0.111(default)       pigz/2.3.4(default)         ruby-install/0.6.1(default)
jdk/8u121(default)          python/2.7.12(default)      tmux/2.3(default)
jup/0.1(default)            python/3.6.0                tools/0.1(default)

------------------ /beegfs/common/software/2017/modules/modulefiles/bioinformatics ------------------
allpathslg/52488(default)   gatk/3.4.46(default)        segemehl/0.2.0(default)
bamutil/1.0.13(default)     graphviz/2.40.1(default)    seqtk/1.2.r94(default)
bcl2fastq/2.17.1.14(default) hisat/2.0.4(default)       skewer/0.2.2(default)
bedtools/2.24.0             igvtools/2.3.89(default)    snpeff/4.3.i(default)
bedtools/2.26.0(default)    imtornado/2.0.3.3(default)  spades/3.10.0(default)
bowtie/1.2.0                lofreq/2.1.2(default)       sratoolkit/2.8.1
bowtie/2.2.9(default)       meme/4.11.3                 sratoolkit/2.8.1-3(default)
bwa/0.7.15(default)         meme/4.12.0(default)        star/2.5.2b(default)
bwtool/git170623(default)   methpipe/3.4.2(default)     stringtie/1.3.0(default)
cufflinks/2.2.1(default)    ngsutils/0.5.9(default)     tophat/2.1.1(default)
cutadapt/1.13.0(default)    picard/2.8.1(default)       vcftools/0.1.14(default)
epiteome/1.0.0(default)     qiime/1.9.1(default)        walt/1.0.0(default)
expat/2.2.0(default)        rsem/1.3.0(default)
fastqc/0.11.5(default)      samtools/1.3.1(default)
```

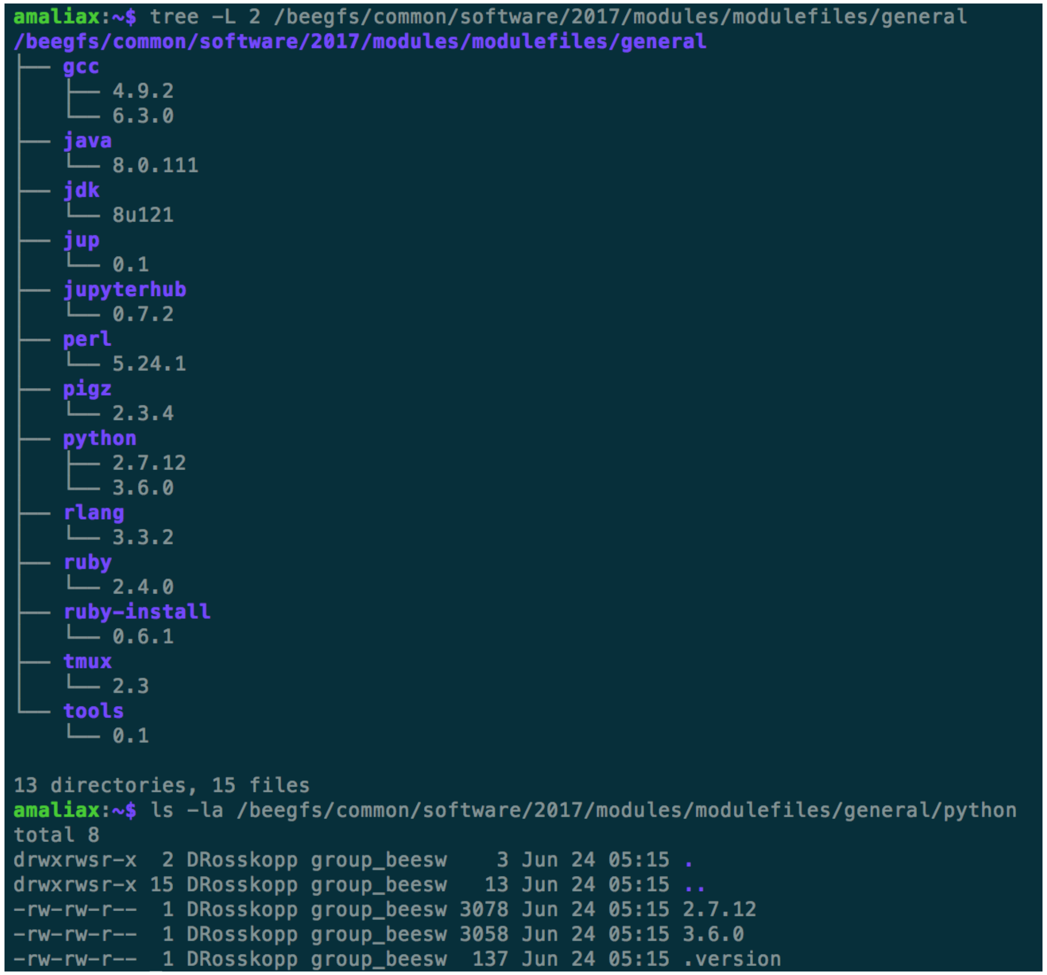
Notice the 2 arrows and the two distinct blocks of software being shown.

The distinct blocks are generated by the different paths kept on the `MODULEPATH` variable set like this:

```
export MODULEPATH=/beegfs/common/software/2017/modules/modulefiles\
/general:/beegfs/common/software/2017/modules/modulefiles/bioinformatics
```

The modules system

`module avail` simply lists the contents of the `MODULEPATH` variable.

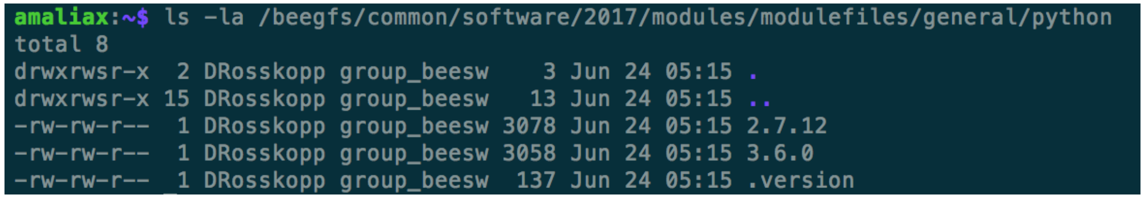


```
amaliax:~$ tree -L 2 /beegfs/common/software/2017/modules/modulefiles/general
/beegfs/common/software/2017/modules/modulefiles/general
├── gcc
│   ├── 4.9.2
│   └── 6.3.0
├── java
│   └── 8.0.111
├── jdk
│   └── 8u121
├── jup
│   └── 0.1
├── jupyterhub
│   └── 0.7.2
├── perl
│   └── 5.24.1
├── pigz
│   └── 2.3.4
├── python
│   ├── 2.7.12
│   └── 3.6.0
├── rlang
│   └── 3.3.2
├── ruby
│   └── 2.4.0
├── ruby-install
│   └── 0.6.1
├── tmux
│   └── 2.3
└── tools
    └── 0.1

13 directories, 15 files
amaliax:~$ ls -la /beegfs/common/software/2017/modules/modulefiles/general/python
total 8
drwxrwsr-x  2 DRosskopp group_beesw    3 Jun 24 05:15 .
drwxrwsr-x 15 DRosskopp group_beesw   13 Jun 24 05:15 ..
-rw-rw-r--  1 DRosskopp group_beesw 3078 Jun 24 05:15 2.7.12
-rw-rw-r--  1 DRosskopp group_beesw 3058 Jun 24 05:15 3.6.0
-rw-rw-r--  1 DRosskopp group_beesw  137 Jun 24 05:15 .version
```

Example content of a folder in the `MODULEPATH`.

Example content for Python.

The modules system



```
amaliax:~$ ls -la /beegfs/common/software/2017/modules/modulefiles/general/python
total 8
drwxrwsr-x  2 DRosskopp group_beesw    3 Jun 24 05:15 .
drwxrwsr-x 15 DRosskopp group_beesw   13 Jun 24 05:15 ..
-rw-rw-r--  1 DRosskopp group_beesw 3078 Jun 24 05:15 2.7.12
-rw-rw-r--  1 DRosskopp group_beesw 3058 Jun 24 05:15 3.6.0
-rw-rw-r--  1 DRosskopp group_beesw  137 Jun 24 05:15 .version
```

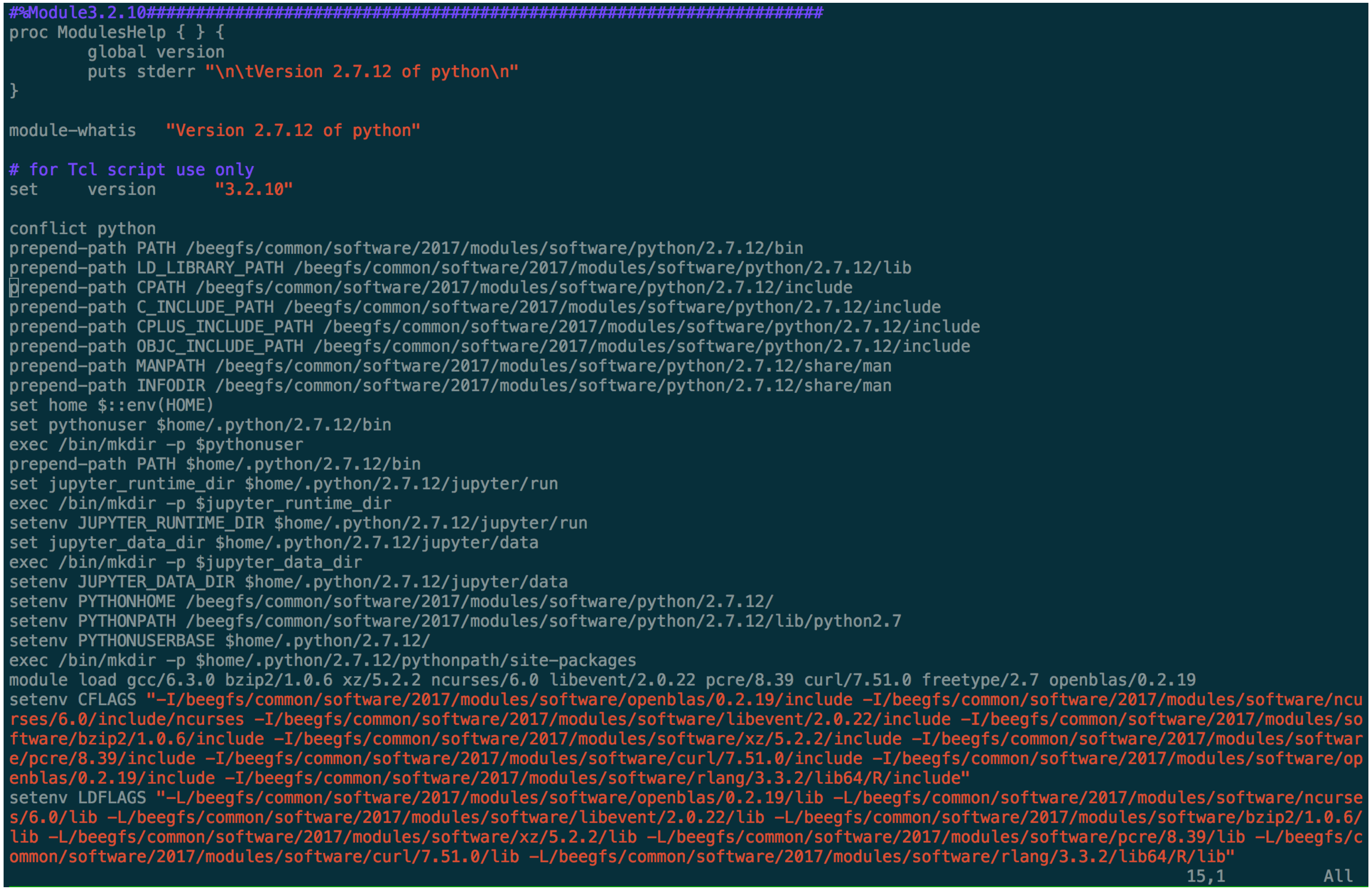
`2.7.12` module file for python 2.7.12

`3.6.0` module file for python 3.6.0

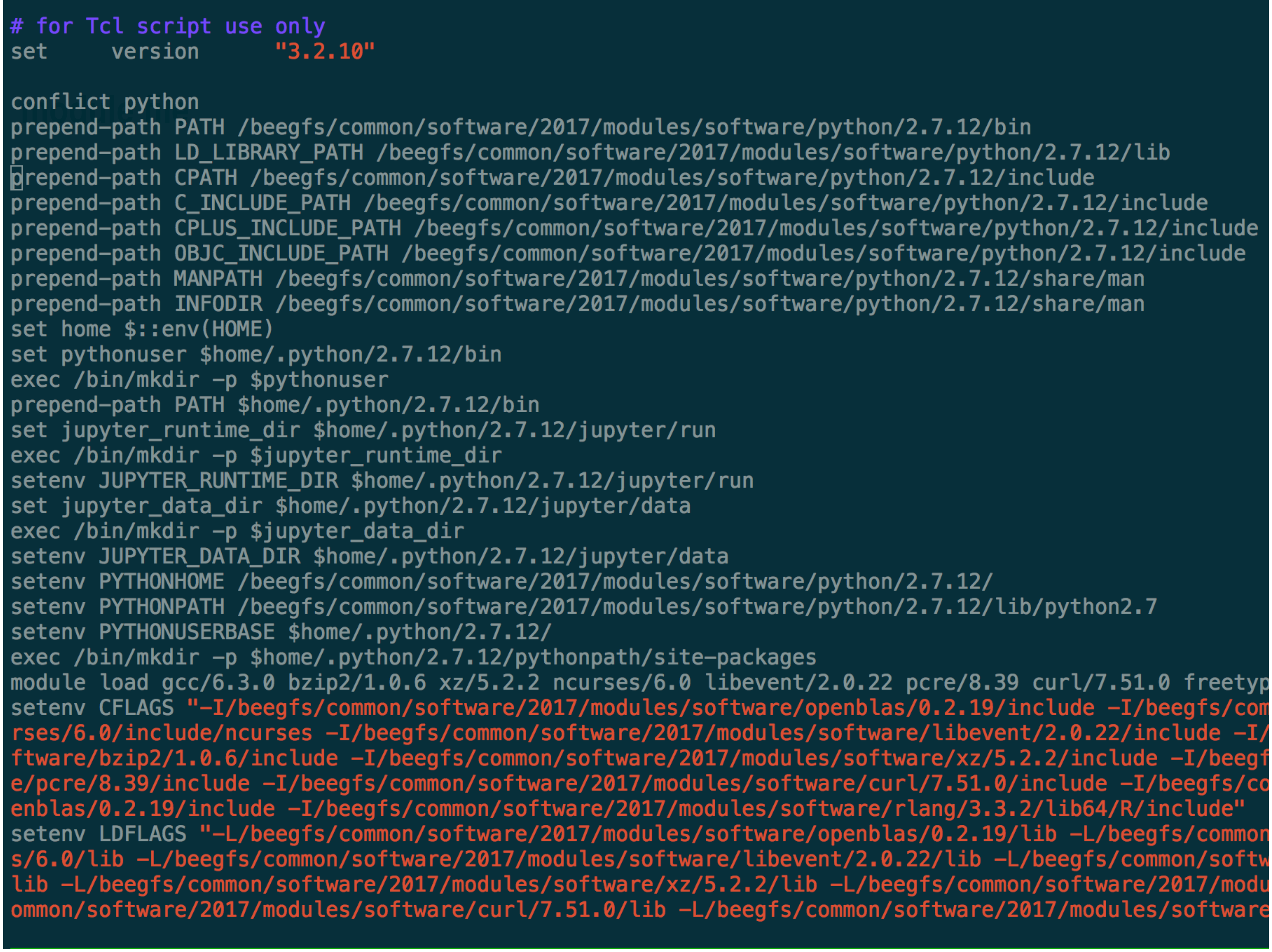
`.version` file defining the default version:

```
#%Module1.0##########################################################
##
## version file for python
##
set ModulesVersion      "2.7.12"
```

module file

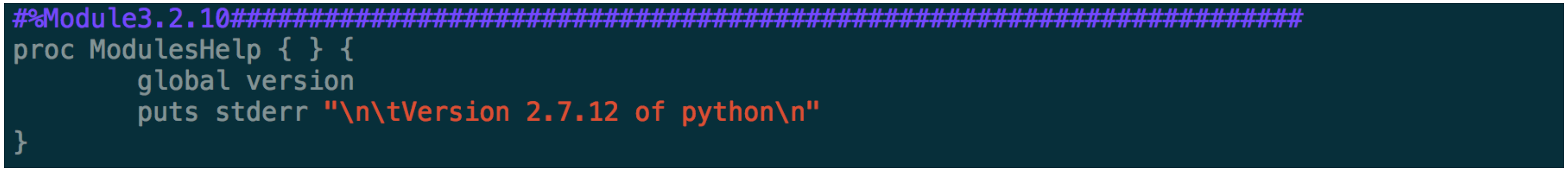


```
#%Module3.2.10#####################################################################
proc ModulesHelp { } {
        global version
        puts stderr "\n\tVersion 2.7.12 of python\n"
}

module-whatis   "Version 2.7.12 of python"

# for Tcl script use only
set     version      "3.2.10"

conflict python
prepend-path PATH /beegfs/common/software/2017/modules/software/python/2.7.12/bin
prepend-path LD_LIBRARY_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/lib
prepend-path CPATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path C_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path CPLUS_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path OBJC_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path MANPATH /beegfs/common/software/2017/modules/software/python/2.7.12/share/man
prepend-path INFODIR /beegfs/common/software/2017/modules/software/python/2.7.12/share/man
set home $::env(HOME)
set pythonuser $home/.python/2.7.12/bin
exec /bin/mkdir -p $pythonuser
prepend-path PATH $home/.python/2.7.12/bin
set jupyter_runtime_dir $home/.python/2.7.12/jupyter/run
exec /bin/mkdir -p $jupyter_runtime_dir
setenv JUPYTER_RUNTIME_DIR $home/.python/2.7.12/jupyter/run
set jupyter_data_dir $home/.python/2.7.12/jupyter/data
exec /bin/mkdir -p $jupyter_data_dir
setenv JUPYTER_DATA_DIR $home/.python/2.7.12/jupyter/data
setenv PYTHONHOME /beegfs/common/software/2017/modules/software/python/2.7.12/
setenv PYTHONPATH /beegfs/common/software/2017/modules/software/python/2.7.12/lib/python2.7
setenv PYTHONUSERBASE $home/.python/2.7.12/
exec /bin/mkdir -p $home/.python/2.7.12/pythonpath/site-packages
module load gcc/6.3.0 bzip2/1.0.6 xz/5.2.2 ncurses/6.0 libevent/2.0.22 pcre/8.39 curl/7.51.0 freetype/2.7 openblas/0.2.19
setenv CFLAGS "-I/beegfs/common/software/2017/modules/software/openblas/0.2.19/include -I/beegfs/common/software/2017/modules/software/ncu
rses/6.0/include/ncurses -I/beegfs/common/software/2017/modules/software/libevent/2.0.22/include -I/beegfs/common/software/2017/modules/so
ftware/bzip2/1.0.6/include -I/beegfs/common/software/2017/modules/software/xz/5.2.2/include -I/beegfs/common/software/2017/modules/softwar
e/pcre/8.39/include -I/beegfs/common/software/2017/modules/software/curl/7.51.0/include -I/beegfs/common/software/2017/modules/software/op
enblas/0.2.19/include -I/beegfs/common/software/2017/modules/software/rlang/3.3.2/lib64/R/include"
setenv LDFLAGS "-L/beegfs/common/software/2017/modules/software/openblas/0.2.19/lib -L/beegfs/common/software/2017/modules/software/ncurse
s/6.0/lib -L/beegfs/common/software/2017/modules/software/libevent/2.0.22/lib -L/beegfs/common/software/2017/modules/software/bzip2/1.0.6/
lib -L/beegfs/common/software/2017/modules/software/xz/5.2.2/lib -L/beegfs/common/software/2017/modules/software/pcre/8.39/lib -L/beegfs/c
ommon/software/2017/modules/software/curl/7.51.0/lib -L/beegfs/common/software/2017/modules/software/rlang/3.3.2/lib64/R/lib"
                                                                                                                    15,1          All
```



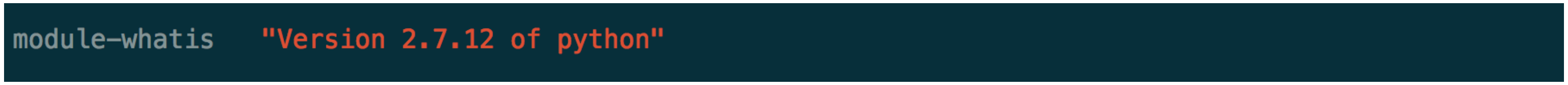
```
# for Tcl script use only
set     version      "3.2.10"

conflict python
prepend-path PATH /beegfs/common/software/2017/modules/software/python/2.7.12/bin
prepend-path LD_LIBRARY_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/lib
prepend-path CPATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path C_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path CPLUS_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path OBJC_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path MANPATH /beegfs/common/software/2017/modules/software/python/2.7.12/share/man
prepend-path INFODIR /beegfs/common/software/2017/modules/software/python/2.7.12/share/man
set home $::env(HOME)
set pythonuser $home/.python/2.7.12/bin
exec /bin/mkdir -p $pythonuser
prepend-path PATH $home/.python/2.7.12/bin
set jupyter_runtime_dir $home/.python/2.7.12/jupyter/run
exec /bin/mkdir -p $jupyter_runtime_dir
setenv JUPYTER_RUNTIME_DIR $home/.python/2.7.12/jupyter/run
set jupyter_data_dir $home/.python/2.7.12/jupyter/data
exec /bin/mkdir -p $jupyter_data_dir
setenv JUPYTER_DATA_DIR $home/.python/2.7.12/jupyter/data
setenv PYTHONHOME /beegfs/common/software/2017/modules/software/python/2.7.12/
setenv PYTHONPATH /beegfs/common/software/2017/modules/software/python/2.7.12/lib/python2.7
setenv PYTHONUSERBASE $home/.python/2.7.12/
exec /bin/mkdir -p $home/.python/2.7.12/pythonpath/site-packages
module load gcc/6.3.0 bzip2/1.0.6 xz/5.2.2 ncurses/6.0 libevent/2.0.22 pcre/8.39 curl/7.51.0 freetyp
setenv CFLAGS "-I/beegfs/common/software/2017/modules/software/openblas/0.2.19/include -I/beegfs/com
rses/6.0/include/ncurses -I/beegfs/common/software/2017/modules/software/libevent/2.0.22/include -I/
ftware/bzip2/1.0.6/include -I/beegfs/common/software/2017/modules/software/xz/5.2.2/include -I/beegf
e/pcre/8.39/include -I/beegfs/common/software/2017/modules/software/curl/7.51.0/include -I/beegfs/co
enblas/0.2.19/include -I/beegfs/common/software/2017/modules/software/rlang/3.3.2/lib64/R/include"
setenv LDFLAGS "-L/beegfs/common/software/2017/modules/software/openblas/0.2.19/lib -L/beegfs/common
s/6.0/lib -L/beegfs/common/software/2017/modules/software/libevent/2.0.22/lib -L/beegfs/common/softw
lib -L/beegfs/common/software/2017/modules/software/xz/5.2.2/lib -L/beegfs/common/software/2017/modu
ommon/software/2017/modules/software/curl/7.51.0/lib -L/beegfs/common/software/2017/modules/software
```

module file



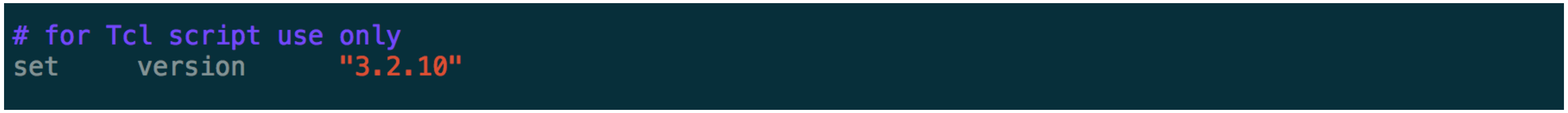
```
#%Module3.2.10######################################################################
proc ModulesHelp { } {
        global version
        puts stderr "\n\tVersion 2.7.12 of python\n"
}
```

Help message shown for this modules with `module help python/2.7.12`



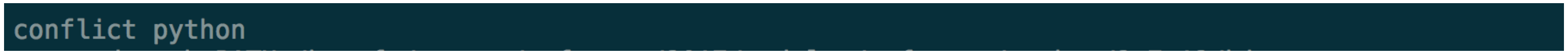
```
module-whatis   "Version 2.7.12 of python"
```

Message to be shown by `module whatis python/2.7.12`.



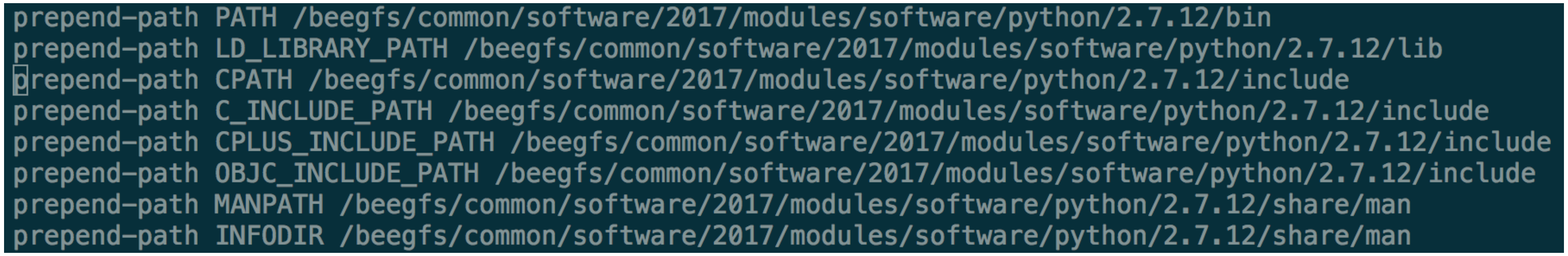
```
# for Tcl script use only
set     version      "3.2.10"
```

Modules system version.



```
conflict python
```

Conflicts. ie. Do not load if a different python version is already loaded.



```
prepend-path PATH /beegfs/common/software/2017/modules/software/python/2.7.12/bin
prepend-path LD_LIBRARY_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/lib
prepend-path CPATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path C_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path CPLUS_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path OBJC_INCLUDE_PATH /beegfs/common/software/2017/modules/software/python/2.7.12/include
prepend-path MANPATH /beegfs/common/software/2017/modules/software/python/2.7.12/share/man
prepend-path INFODIR /beegfs/common/software/2017/modules/software/python/2.7.12/share/man
```

Paths prepended when the module is loaded and removed when the module is unloaded.

module file

```
set home $::env(HOME)
```

`$::env(HOME)` captures the variable HOME from the environment of the user loading the module.

`set home $::env(HOME)` sets a variable `home` to be used within the module with the result of `$::env(HOME)`.

```
set pythonuser $home/.python/2.7.12/bin
```

Sets the python user variable to be used within the module.

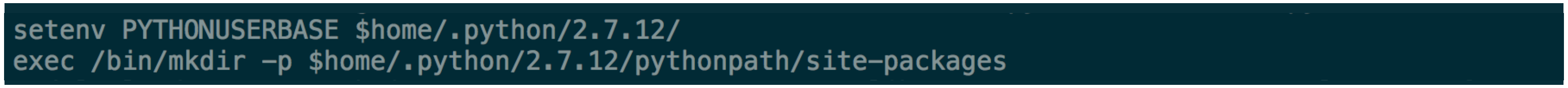
```
exec /bin/mkdir -p $pythonuser
```

Executes the command `/bin/mkdir -p $pythonuser` every time the module is loaded.

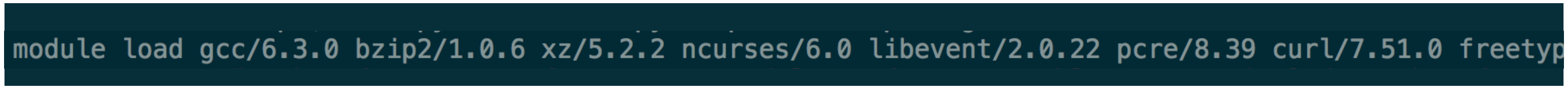
```
prepend-path PATH $home/.python/2.7.12/bin
```

Prepends `$home/.python/2.7.12/bin` to the user environment PATH variable.

module file


```
setenv PYTHONUSERBASE $home/.python/2.7.12/
exec /bin/mkdir -p $home/.python/2.7.12/pythonpath/site-packages
```

Please notice `PYTHONUSERBASE`


```
module load gcc/6.3.0 bzip2/1.0.6 xz/5.2.2 ncurses/6.0 libevent/2.0.22 pcre/8.39 curl/7.51.0 freetyp
```

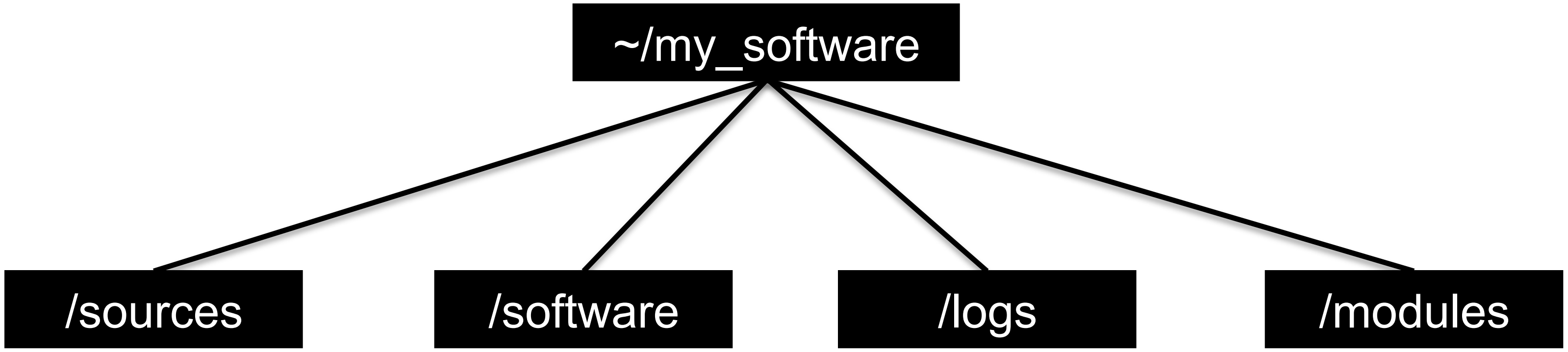
Loads other modules as dependencies when loading the python module.

Installing software without `su` access

Several examples can be obtained from:

https://github.com/mpg-age-bioinformatics/draco_pipelines/blob/master/software/software.sh

Structure:



```
mkdir -p ~/my_software/sources ~/my_software/software \
~/my_software/logs  ~/my_software/modules
```

Installing software without `su` access

Download and edit the script for automatic generation of module files:

```
cd ~/
wget https://raw.githubusercontent.com/mpg-age-bioinformatics/
draco_pipelines/master/software/newmod.sh

sed -i 's/\/u\/jboucas\/modules/~\/my_software/g newmod.sh
chmod +x newmod.sh
```

Add the `~/my_software/modules`to the MODULEPATH:

```
export MODULEPATH=$MODULEPATH:~/my_software/modules
```

You might want to add this line to your `~/.bashrc`

Installing software without `su` access

```
cd ~/my_software/sources
wget http://zlib.net/pigz/pigz-2.3.4.tar.gz
tar -zxvf pigz-2.3.4.tar.gz
cd pigz-2.3.4
make
mkdir -p ~/my_software/software/pigz/2.3.4/bin
cp pigz unpigz ~/my_software/software/pigz/2.3.4/bin
~/newmod.sh -s pigz -p ~/my_software/modules -v 2.3.4 -d 2.3.4


# TEST #
module avail
module show pigz/2.3.4
module load pigz/2.3.4
which pigz
```

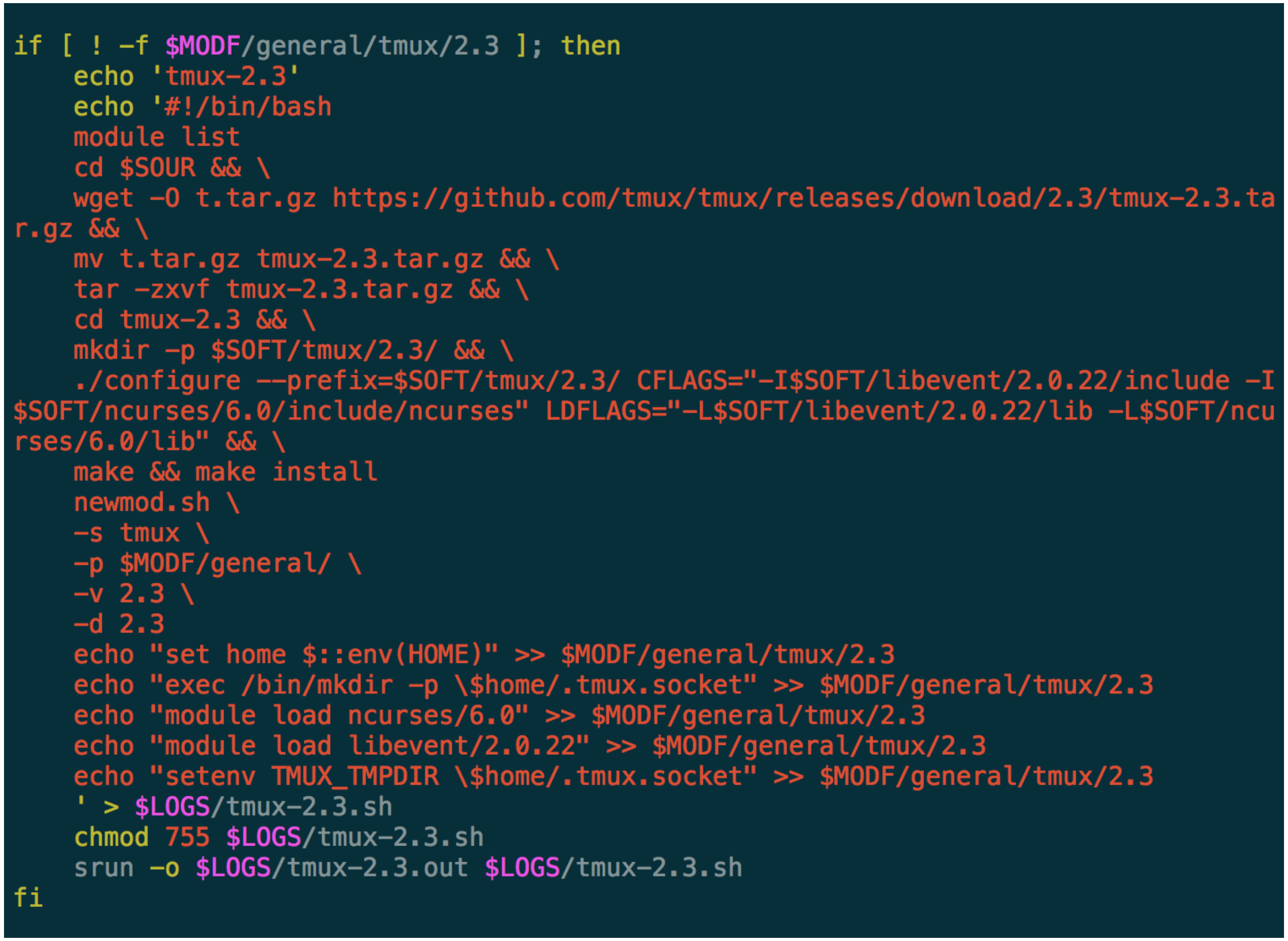
Installing software without `su` access

```
cd ~/my_software/sources
wget http://zlib.net/zlib-1.2.11.tar.gz
tar -zxvf zlib-1.2.11.tar.gz
cd zlib-1.2.11
make
mkdir -p ~/my_software/software/libz/1.2.11
./configure -prefix=~/my_software/software/libz/1.2.11
make
make install
~/newmod.sh -s libz -p ~/my_software/modules -v 1.2.11 -d 1.2.11
```

Installing software without `su` access

```
echo "#!/bin/bash
cd ~/my_software/sources
wget http://zlib.net/zlib-1.2.11.tar.gz
tar -zxvf zlib-1.2.11.tar.gz
cd zlib-1.2.11
make
mkdir -p ~/my_software/software/libz/1.2.11
./configure -prefix=~/my_software/software/libz/1.2.11
make
make install
~/newmod.sh -s libz -p ~/my_software/modules -v 1.2.11 -d 1.2.11
" > ~/my_software/logs/libz-1.2.11.sh
chmod +x ~/my_software/logs/libz-1.2.11.sh
~/my_software/logs/libz-1.2.11.sh 2>&1 | tee \
~/my_software/logs/libz-1.2.11.log
```

Installing software without `su` access



```
if [ ! -f $MODF/general/tmux/2.3 ]; then
    echo 'tmux-2.3'
    echo '#!/bin/bash
    module list
    cd $SOUR && \
    wget -O t.tar.gz https://github.com/tmux/tmux/releases/download/2.3/tmux-2.3.ta
r.gz && \
    mv t.tar.gz tmux-2.3.tar.gz && \
    tar -zxvf tmux-2.3.tar.gz && \
    cd tmux-2.3 && \
    mkdir -p $SOFT/tmux/2.3/ && \
    ./configure --prefix=$SOFT/tmux/2.3/ CFLAGS="-I$SOFT/libevent/2.0.22/include -I
$SOFT/ncurses/6.0/include/ncurses" LDFLAGS="-L$SOFT/libevent/2.0.22/lib -L$SOFT/ncu
rses/6.0/lib" && \
    make && make install
    newmod.sh \
    -s tmux \
    -p $MODF/general/ \
    -v 2.3 \
    -d 2.3
    echo "set home $::env(HOME)" >> $MODF/general/tmux/2.3
    echo "exec /bin/mkdir -p \$home/.tmux.socket" >> $MODF/general/tmux/2.3
    echo "module load ncurses/6.0" >> $MODF/general/tmux/2.3
    echo "module load libevent/2.0.22" >> $MODF/general/tmux/2.3
    echo "setenv TMUX_TMPDIR \$home/.tmux.socket" >> $MODF/general/tmux/2.3
    ' > $LOGS/tmux-2.3.sh
    chmod 755 $LOGS/tmux-2.3.sh
    srun -o $LOGS/tmux-2.3.out $LOGS/tmux-2.3.sh
fi
```

Installing R packages

R standard packages:

```
base  datasets  graphics  grDevices  methods  parallel  stats  stats4
tcltk  tools  utils
```

This are the only packages that should be inside `` `lib64/R/library` ``. The `` `library` `` folder should have permissions `` `chmod –R 555 library` ``. This will enforce the use of the `` `R_LIBS_USER` `` variable for installation of new packages.

Alternative (eg. install biomaRt):

```
source("https://bioconductor.org/biocLite.R")
biocLite("biomaRt”, lib=Sys.getenv("R_LIBS_USER"))
```

The `` `lib` `` argument can also be used in (eg. install packrat):

```
install.packages(”packrat”, lib=Sys.getenv("R_LIBS_USER"))
```

Installing python packages

Installing the AGEpy package:

```
module load python
pip install AGEpy --user
```

Alternative 1:

```
 module load python
git clone https://github.com/mpg-age-bioinformatics/AGEpy
cd AGEpy
pip install ../AGEpy --user
```

Alternative 1b (no package control):

```
 cd AGEpy
python setup.py install --user
```

The `` `--user` `` argument will make use of the `` `PYTHONUSERBASE` `` variable.

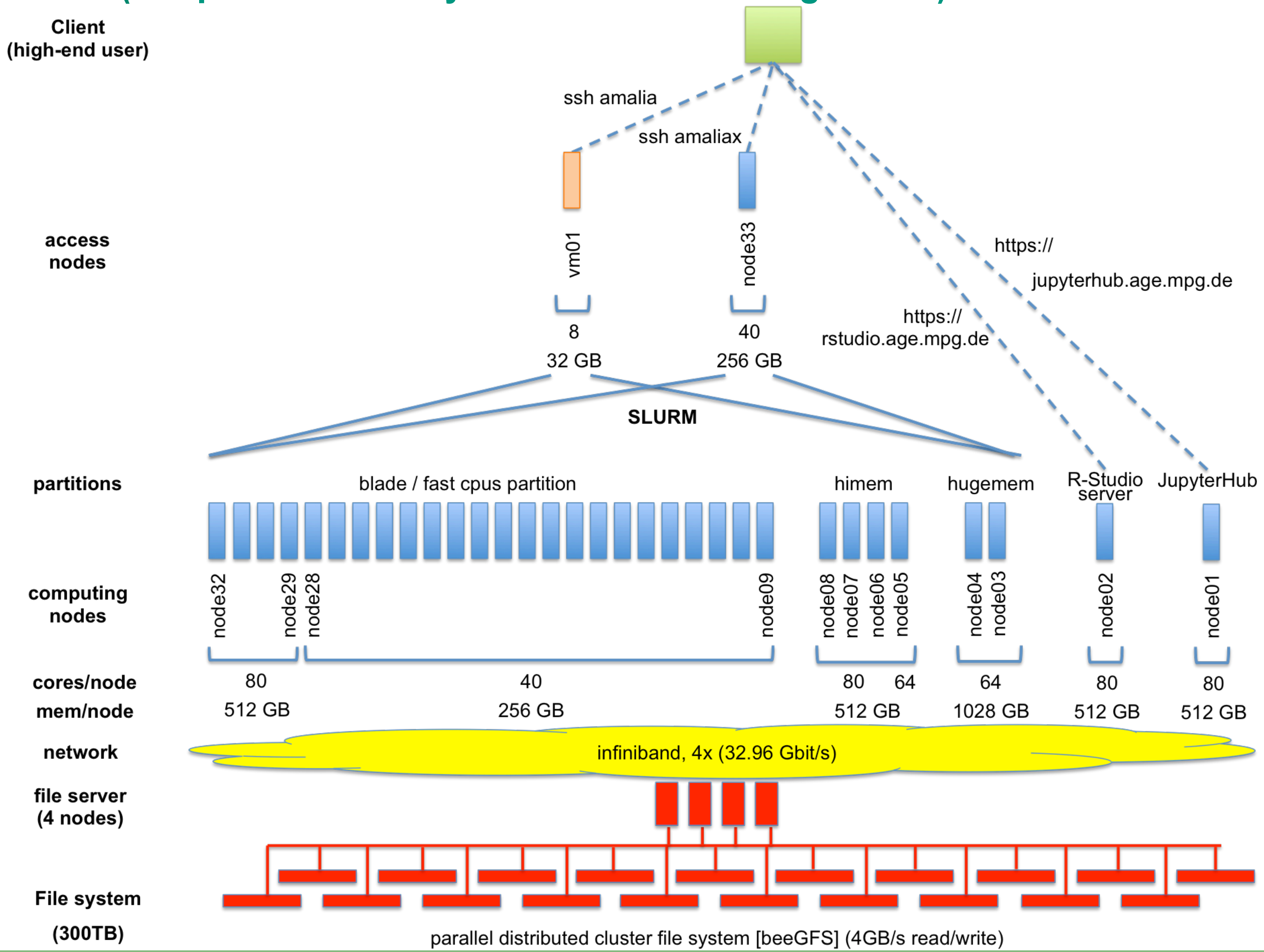
Installing perl packages

Perl libraries are loaded from the value of `` `PERL5LIB` ``. We have added a variable to module perl to ease local installations - `` `PERLUSER` `` - check out the content of these two variables.

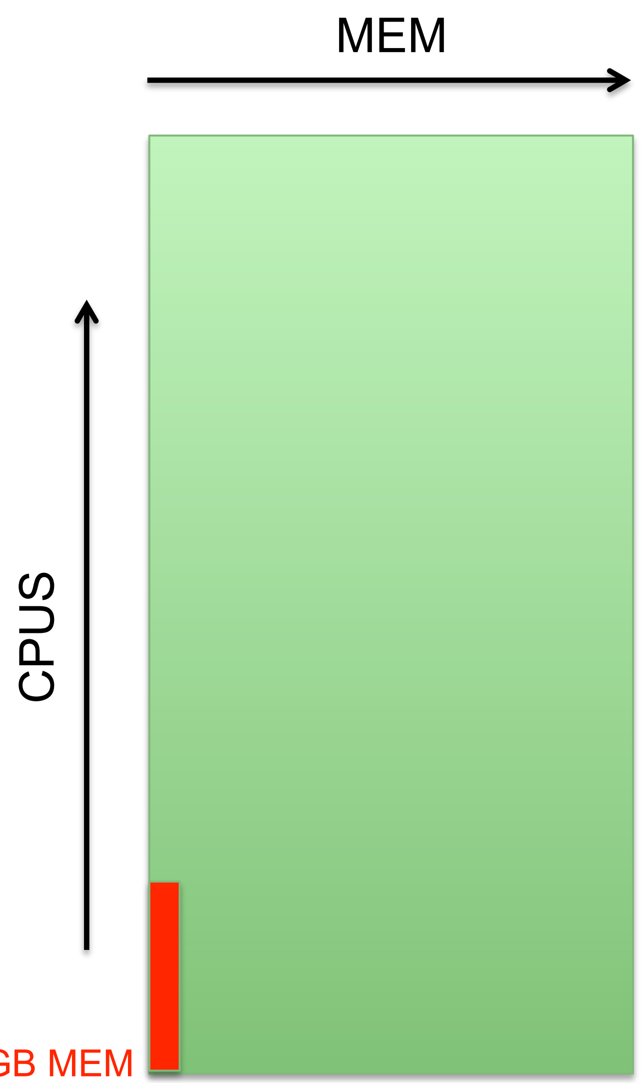
Example instalation of `Log::Log4perl`:

```
module load perl/5.24.1
cpanm Log::Log4perl -l $PERLUSER

# check where it got installed
perldoc -l Log::Log4perl
```

SLURM (Simple Linux Utility for Resource Management)



SLURM: Why an HPC cluster?



One HPC node:
40 CPUS
256 GB MEM

MAC mini: 8 cpus, 16 GB MEM

In-house/MPI-AGE

~13 TFlops
Queuing times:
0-20' : 80% of jobs
0-180' : 98% of jobs

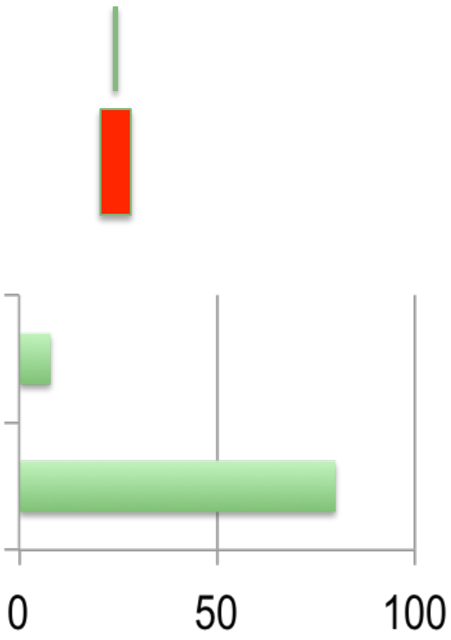
SLURM: Why an HPC cluster?

MAC mini

1 user

4 raw data files

20 cpus / raw file

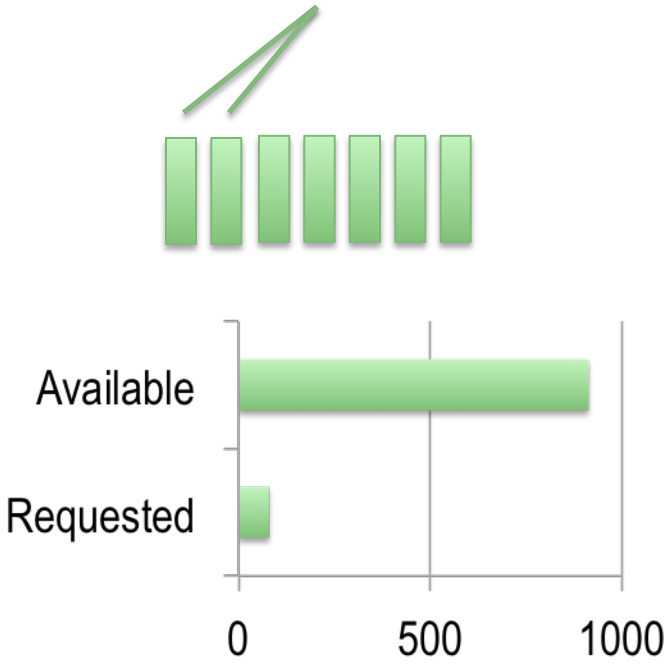


HPC CLUSTER

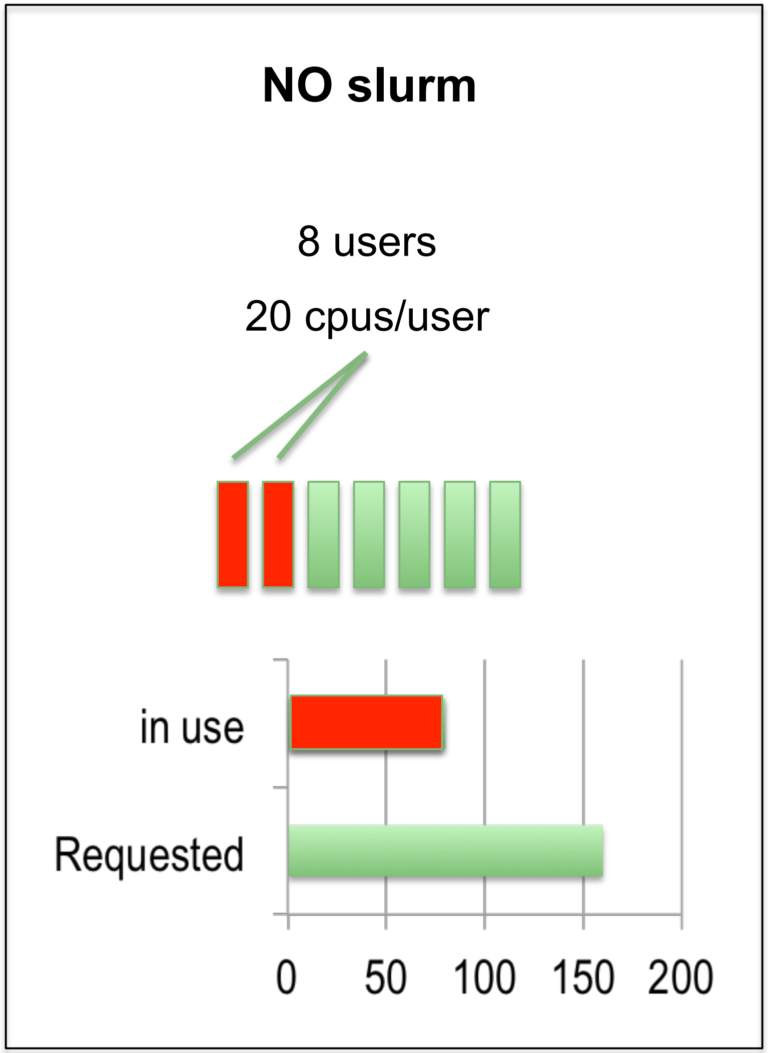
1 user

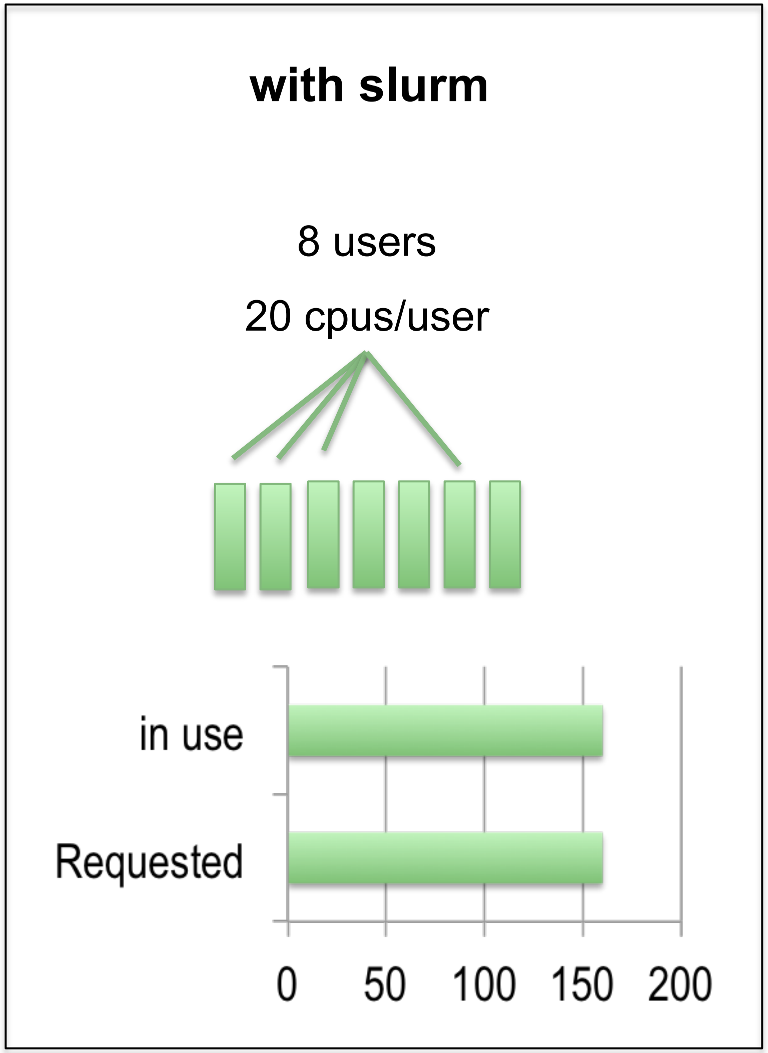
4 raw data files

20 cpus / raw file



SLURM: Why SLURM?





SLURM: Why SLURM?

NO slurm

```
bwa mem sample1.fastq
```

with slurm

```
srun bwa mem sample1.fastq
```

SLURM: How to?

NO slurm

```
bwa mem –T 18 sample1.fastq
```

with slurm

```
srun --cpus-per-task=18 \
bwa mem –T 18 sample1.fastq
```

SLURM: How to?

NO slurm

```
bwa mem –T 18 sample1.fastq
```

with slurm

```
srun --cpus-per-task=18 \
--mem=64gb \
bwa mem –T 18 sample1.fastq
```

SLURM: How to?

NO slurm

```
bwa mem –T 18 sample1.fastq
```

with slurm

```
srun --cpus-per-task=18 \
--mem=15gb \
--time=5-24 \
bwa mem –T 18 sample1.fastq
```

(5 days and 24 hours = 6 days

alternative: 144:00:00;

maximum = 14 days)

SLURM: How to?

NO slurm

```
bwa mem –T 18 sample1.fastq
```

with slurm

```
srun --cpus-per-task=18 \
--mem=15gb \
--time=5-24 \
-p blade \
bwa mem –T 18 sample1.fastq
```

SLURM: How to?

NO slurm

```
bwa mem –T 18 sample1.fastq
```

with slurm

```
srun --cpus-per-task=18 \
--mem=15gb \
--time=5-24 \
-p blade \
-o slurm_logs/bwa_1.out \
bwa mem –T 18 sample1.fastq
```

SLURM: How to?

NO slurm

```
./align_1.sh
```

```
#!/bin/bash
bwa mem –T 18 sample1.fastq
exit
```

with slurm

```
sbatch --cpus-per-task=18 \
--mem=15gb \
--time=5-24 \
-p blade \
-o slurm_logs/bwa_1.out \
align_1.sh
```

SLURM: scripting

NO slurm

```
#!/bin/bash
cd ~/project/raw_data
bwa mem –T 18 sample1.fastq
exit



./align_1.sh
```

with slurm

```
#!/bin/bash
#SBATCH --cpus-per-task=18
#SBATCH --mem=15gb
#SBATCH --time=5-24
#SBATCH -p blade
#SBATCH -o slurm_logs/bwa_1.out
cd ~/project/raw_data
bwa mem –T 18 sample1.fastq
exit

sbatch align_1.sh
```

SLURM: iterating jobs over files

```
#!/bin/bash
cd ~/project/raw_data                                # go to folder containing files

for f in $(ls *.fastq); do echo “#!/bin/bash         # for each file, echo a script
cd ~/project/raw_data                                # that goes to folder
bwa mem –T 18 ${f}                                   # executes job on file and
rm ~/project/tmp/${f}.sh                             # once completed removes
“ > ~/project/tmp/${f}.sh                            # temporary script

chmod 755 ~/project/tmp/${f}.sh
rm ~/project/slurm_logs/${f}.*.out                   # removes pre-existing logs

sbatch --cpus-per-task=18 --mem=15gb \               # start batch job
--time=5-24 -p blade \
-o ~/project/slurm_logs/${f}.%j.out \                # keeps log with job number
~/project/tmp/${f}.sh

done; exit
```

SLURM: iterating jobs over files

```
#!/bin/bash
cd ~/project/raw_data

for f in $(ls *.fastq);
    do rm ~/project/slurm_logs/${f}.*.out

    sbatch --cpus-per-task=18 --mem=15gb --time=5-24 \
    -p blade -o ~/project/slurm_logs/${f}.%j.out << EOF
#!/bin/bash
cd ~/project/raw_data
bwa mem -T 18 ${f}
EOF

done
exit
```

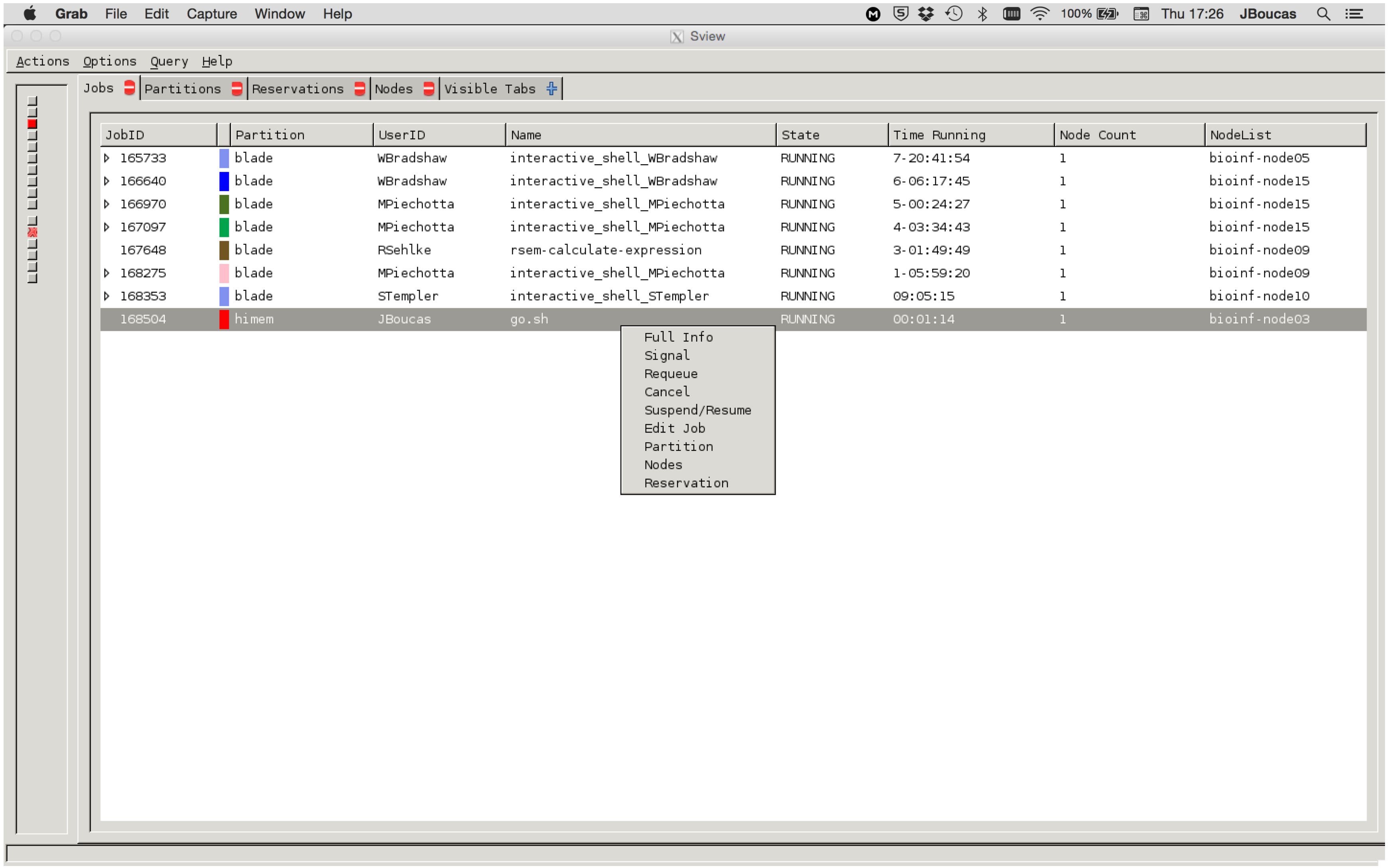
SLURM: iterating without crashing

```
#!/bin/bash
cd ~/project/raw_data
for f in $(ls *.fastq);
   do  rm ~/project/slurm_logs/${f}.*.out

   while [  `squeue -u Jboucas | wc -l` -gt “500” ];
      do echo “sleeping”; sleep 300
   done

   sbatch --cpus-per-task=18 --mem=15gb --time=5-24 \
   -p blade -o ~/project/slurm_logs/${f}.%j.out ~/project/tmp/${f}.sh <<EOF
#!/bin/bash
cd ~/project/raw_data
bwa mem -T 18 ${f}
EOF

done
exit
```

SLURM: other options

```
--mail-type=BEGIN, END,FAIL,REQUEUE,ALL

# Specifies when email is sent to the job owner. The option argument may consist of a
combination of the allowed mail types

--mail-user=username@age.mpg.de
```

SLURM: sview



Grab File Edit Capture Window Help
100% Thu 17:26 JBoucas
Sview
Actions Options Query Help
Jobs Partitions Reservations Nodes Visible Tabs

JobID		Partition	UserID	Name	State	Time Running	Node Count	NodeList
165733	🟦	blade	WBradshaw	interactive_shell_WBradshaw	RUNNING	7-20:41:54	1	bioinf-node05
166640	🟦	blade	WBradshaw	interactive_shell_WBradshaw	RUNNING	6-06:17:45	1	bioinf-node15
166970	🟩	blade	MPiechotta	interactive_shell_MPiechotta	RUNNING	5-00:24:27	1	bioinf-node15
167097	🟩	blade	MPiechotta	interactive_shell_MPiechotta	RUNNING	4-03:34:43	1	bioinf-node15
167648	🟫	blade	RSehlke	rsem-calculate-expression	RUNNING	3-01:49:49	1	bioinf-node09
168275	🟪	blade	MPiechotta	interactive_shell_MPiechotta	RUNNING	1-05:59:20	1	bioinf-node09
168353	🟦	blade	STempler	interactive_shell_STempler	RUNNING	09:05:15	1	bioinf-node10
168504	🟥	himem	JBoucas	go.sh	RUNNING	00:01:14	1	bioinf-node03

SLURM: other useful commands

```
# show the partitions

sinfo


# show information on nodes

sinfo -N -O partitionname,nodehost,cpus,cpusload,freemem,memory


# requires X forwarding and allows viewing and manipulation of submitted jobs

sview


# lists running jobs

squeue


# show the queue for a user

squeue -u <user name>
```

SLURM: other useful commands

```
# cancels job 65673

scancel 65673


# cancels all jobs of user JBoucas

scancel –u JBoucas


# shows detailed resource information on job 43433

scontrol show job 43433


# show information on a partition

scontrol show partition <parition name>


# show information on a node

scontrol show node <node name>
```

SLURM: other useful commands

```
# starts an interactive terminal window on node15

srun –w node15 ––pty bash


# submit a job3 after job1 and job2 are successfully ready

job1=$(sbatch ––parsable <script1>)

job2=$(sbatch ––parsable <script2>)

sbatch –d afterok:${job1}:${job2} <script3>


# attach to a running job and run a command

srun ––jodib <JOBID> ––pty <command>


# change the partitions of a pending job

scontrol update job <job id> partition=<partition1>,<partition2>,<partition3>
```

Notes:

For current information on RStudio server and JupyterHub please check our cluster first steps page:

https://github.com/mpg-age-bioinformatics/cluster_first_steps

SLURM manual:

http://slurm.schedmd.com/man_index.html

Coming soon:

docker on HPC

END

bioinformatics@age.mpg.de

https://mpg-age-bioinformatics.github.io