- Mac users, if you arrived in advance, how about installing Xquartz ?

- https://www.xquartz.org/

- (warning: needs a reboot)

SKILL PILLS

SKILL PILL:
TERMINAL

Lecture III:

Interacting with Sango

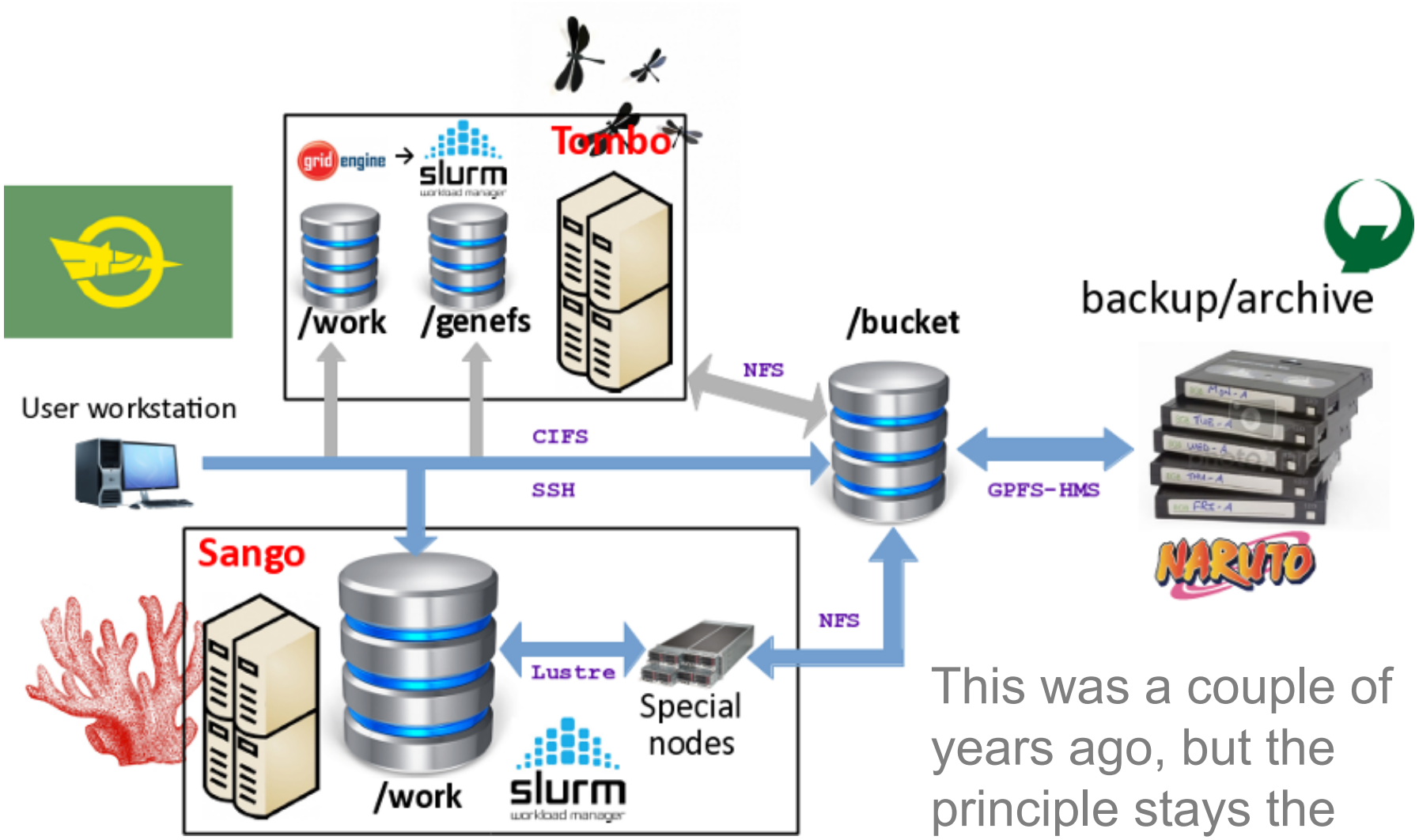
OIST

May 2019

Presenter: Charles Plessy, using slides from previous years (Takeshi R. Tabuchi Y.)

OIST Network

OIST Network



This was a couple of years ago, but the principle stays the same as of today

Access Sango

The OIST Linux super cluster

SKILLPILLS

https://groups.oist.jp/scs/getting-started

Go up to RSD site
SCDA Top
About
Equipment Gallery
Getting started
Computing resources
Documentation
Systems status
Registration & Forms
Policies & legal notice
Calendar
Events
Posts

Contact Scientific Computing Section
Email: it-help@oist.jp

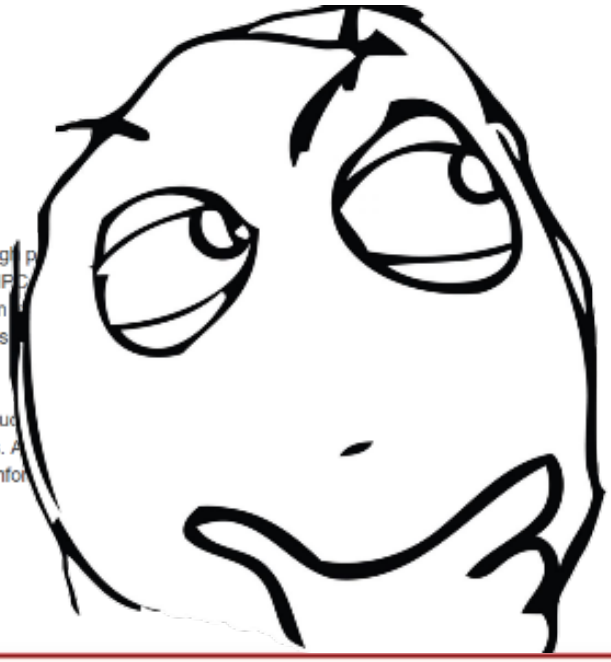
Scientific Computing & Data Analysis Section

Getting started

HPC clusters at OIST

OIST provides clusters, storage and software to support and promote the use of high p
activities. All university faculty, research staff and students are eligible to use the HP
HPC resources are shared between all users, a per unit and per user quota system
for the licensed software. Computations on the clusters are done by submitting jobs
processing resources.

There are two HPC clusters at OIST, Sango and Tombo. Sango is OIST main produc
cluster used for testing, seminar, workshop, demo, and on-demand specific events. A
without restriction. All OIST HPC clusters use Linux Operating System. For more infor
computing resources page.

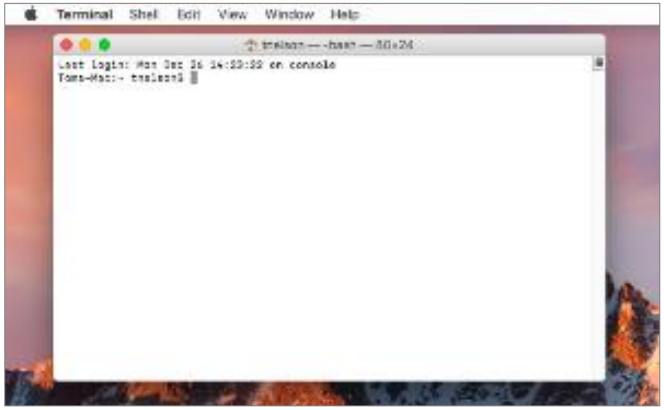


Terminal

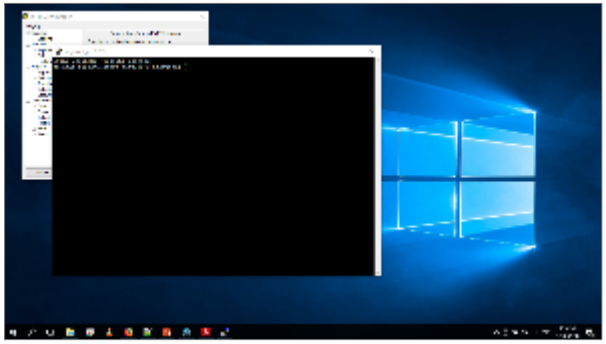
SKILLPILLS

Terminal

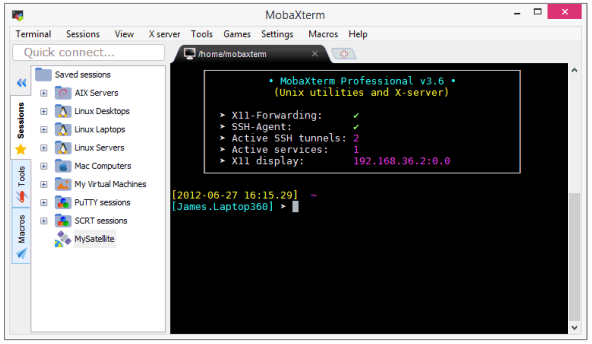
Terminal



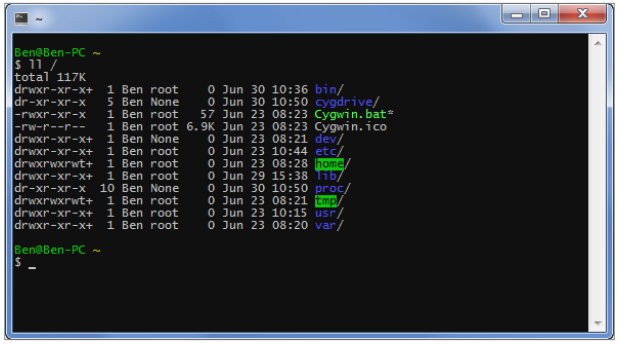
Putty



MobaXterm



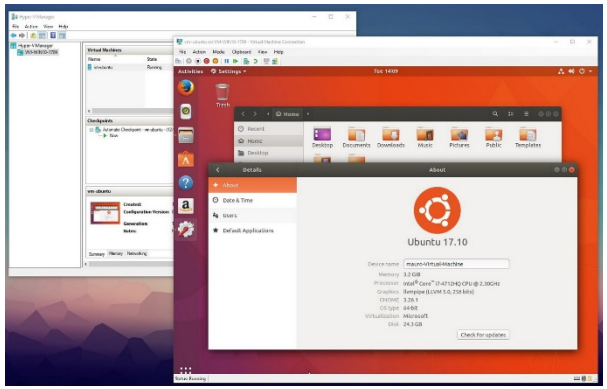
Cygwin



More options for Windows users

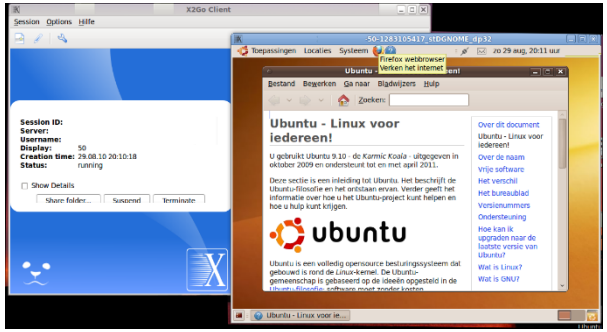
SKILLPILLS

Virtual machine



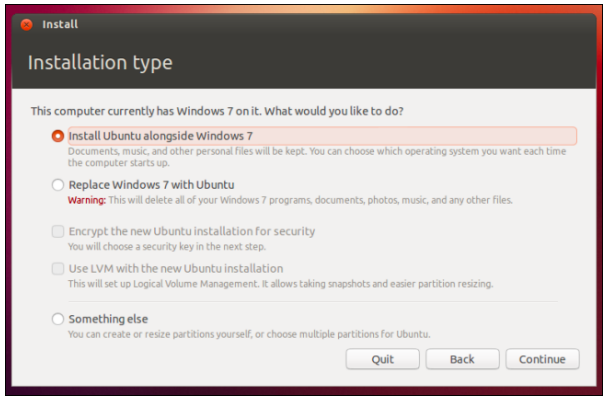
Hyper-V

Remote machine/
Cloud computing



X2Go

Dual-boot



Ubuntu

SSH (Secure Shell)

SKILLPILLS

Secure Shell (SSH) is a protocol for operating network services securely over an unsecured network. Normally used for **remote login** by users to a network system.

Syntax

```
ssh [-options] [user@]server[:port] [commands]
```

e.g.

```
$ ssh derpy-derp@login.oist.jp
$ ssh derpy-derp@sango.oist.jp
$ ssh derpy-derp@tombo.oist.jp
```

Graphical SSH session

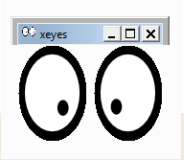
SKILLPILLS

If you want to run programs with **graphical interface**:

```
$ ssh -Y derpy-derp@sango.oist.jp
```

Only do this on a trusted network

Run xeyes to test your session




```
$ ssh -Y derpy-derp@sango.oist.jp
```

If it does not work, you need a “X server”

-> MobaXterm https://mobaxterm.mobatek.net/

-> XQartz https://www.xquartz.org/

Place a configuration file in `~/.ssh/config` to set a default user name and default options

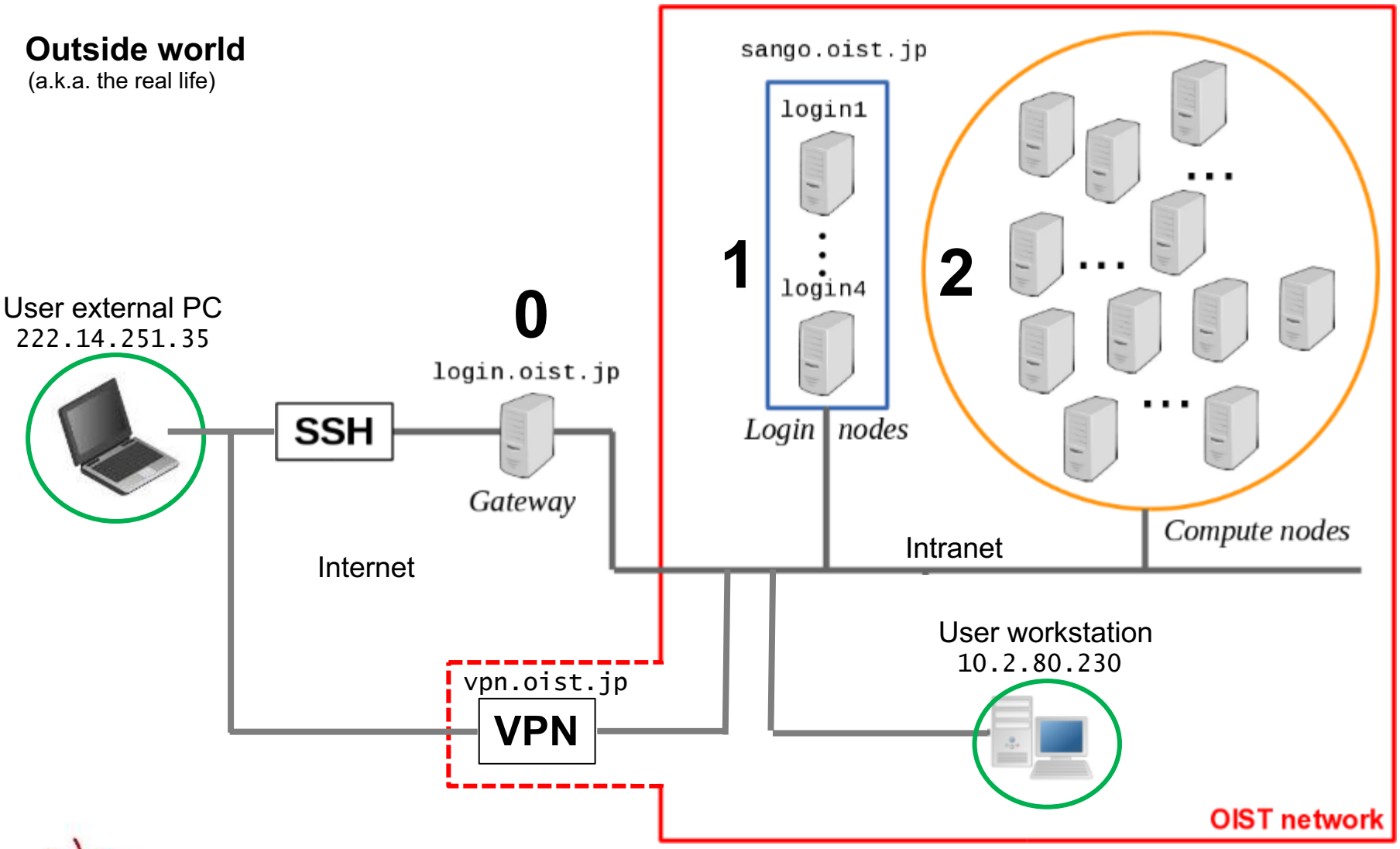
e.g.

```
Host sango*
    User charles-plessy
    ForwardX11 Yes
    Forward Agent yes
```

Only export X11 windows and SSH keys like this on a trusted network

Tired of SSH lagging? Try Mosh! https://mosh.org/

Sango access



Exercise #1: Login

1) Try to login into tombo.oist.jp and sango.oist.jp

```
ssh derpy-derp@tombo.oist.jp
```

```
ssh derpy-derp@sango.oist.jp
```

From inside OIST network

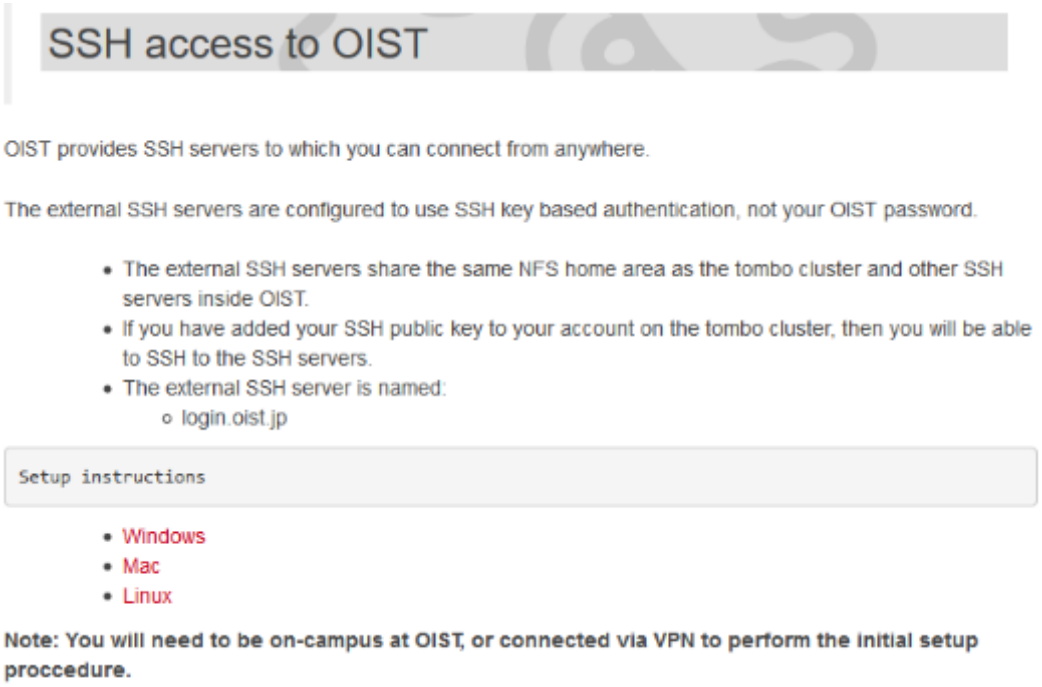
```
[takeshi-tabuchi@local-host ~]$ ssh takeshi-tabuchi@sango.oist.jp

takeshi-tabuchi@sango.oist.jp's password:


Last login: Wed Nov 29 14:29:33 2017 from 10.2.80.94
 ************************************************************************
 *                                                                      *
 *  Unauthorized access to this resource is prohibited.                 *
 *  Okinawa Institute of Science and Technology.                        *
 *                                                                      *
 ************************************************************************

[takeshi-tabuchi@sango-login2 ~]$
```

SKILLPILLS

SSH key (public-private encryption) is used to uniquely identify you.

The external OIST SSH servers are configured to use SSH key based authentication **instead of OIST password**.

https://groups.oist.jp/it/ssh-0



SSH access to OIST

OIST provides SSH servers to which you can connect from anywhere.

The external SSH servers are configured to use SSH key based authentication, not your OIST password.

- The external SSH servers share the same NFS home area as the tombo cluster and other SSH servers inside OIST.
- If you have added your SSH public key to your account on the tombo cluster, then you will be able to SSH to the SSH servers.
- The external SSH server is named:
  - login.oist.jp

```
Setup instructions
```

- Windows
- Mac
- Linux

Note: You will need to be on-campus at OIST, or connected via VPN to perform the initial setup proccedure.

Always protect your SSH key with a passphrase

SKILLPILLS

At anytime, you can change your passphrase with the command:

```
$ ssh-keygen -p
```

Tip: when using your key, ssh ask for your pass***phrase***.* If it asks for your pass***word***, something is going wrong. *(It is SSH jargon, but yes, a passphrase can be a single word…)*

To save typing, you can ask your computer to memorize your passphrase during one session.

```
$ ssh-add
```

You can also forward your key to the (trusted!) server you are connecting to:

```
$ ssh -A yourusername@sango.oist.jp
```

From the outside world

```
[takeshi-tabuchi@local-host ~]$ ssh takeshi-tabuchi@login.oist.jp
Using username "takeshi-tabuchi".
Authenticating with public key "rsa-key-20171002"
Last login: Fri Apr 20 16:12:51 2018 from 10.3.160.162

.........................ZZ.....................................................
.........................Z...Unauthorized access to this resource is prohibited.
...........:ZZZZZ........Z...... Okinawa Institute of Science and Technology ...
........ ..ZZ....ZZO.....Z...7.....................................

[takeshi-tabuchi@loginc01 ~]$ ssh takeshi-tabuchi@sango.oist.jp
takeshi-tabuchi@sango.oist.jp's password:

Last login: Wed Nov 29 14:29:33 2017 from 10.2.80.94
 ************************************************************************
 *                                                                      *
 *  Unauthorized access to this resource is prohibited.                 *
 *  Okinawa Institute of Science and Technology.                        *
 *                                                                      *
 ************************************************************************

[takeshi-tabuchi@sango-login2 ~]$
```

SKILLPILLS

https://groups.oist.jp/scs/getting-started

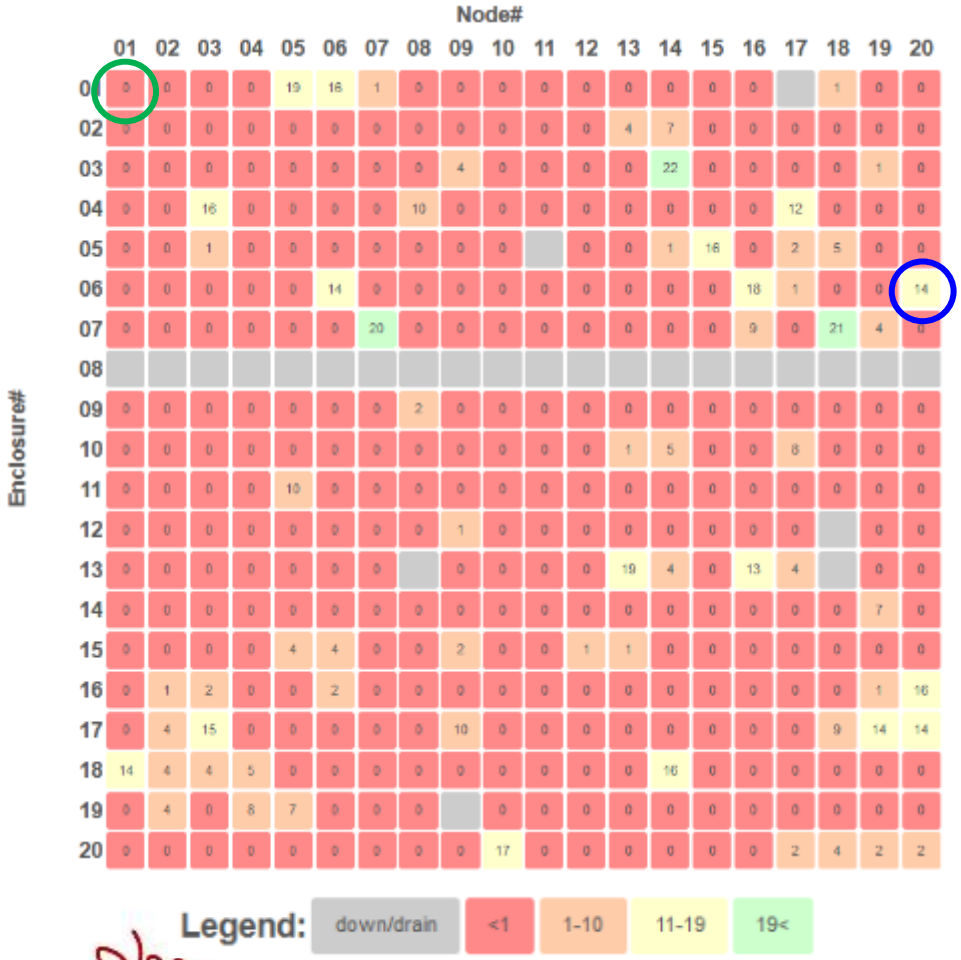
partition name	maximum memory	number of nodes	number of cores	Job runtime (*)		Availability
				default	maximum	
`compute`	128 GiB	400	24	**8 hours**	**7 days**	unrestricted
`largemem`	512 GiB	27	24	**7 weeks**	∞	restricted (**)
`bigmem`	3072 GiB	1	32	**7 weeks**	∞	restricted (**)
`phi`	128 GiB (16GiB/cd.)	3 (4 cards/nd.)	24 (4 x 61)	2 days	2 days	restricted
`datacp`	64 GiB	2	24	8 hours	8 hours	unrestricted
`gpu`	128 GiB (2 x 12GiB/cd.)	3 (4 cards/nd.)	24 (4 x 2 x 2496)	8 hours	2 days	restricted

(*) for `compute` and `largemem` partitions, jobs that do not specify their walltime using the `--time=` option will have the **default** value as their walltime. The walltime value can be set up to the **maximum** value.
(**) priority should be given to genomics computation with high memory and very long computation time, and to jobs requiring more than 128 GiB of memory.
For restricted resources, please consult with us during the open hours to obtain access.

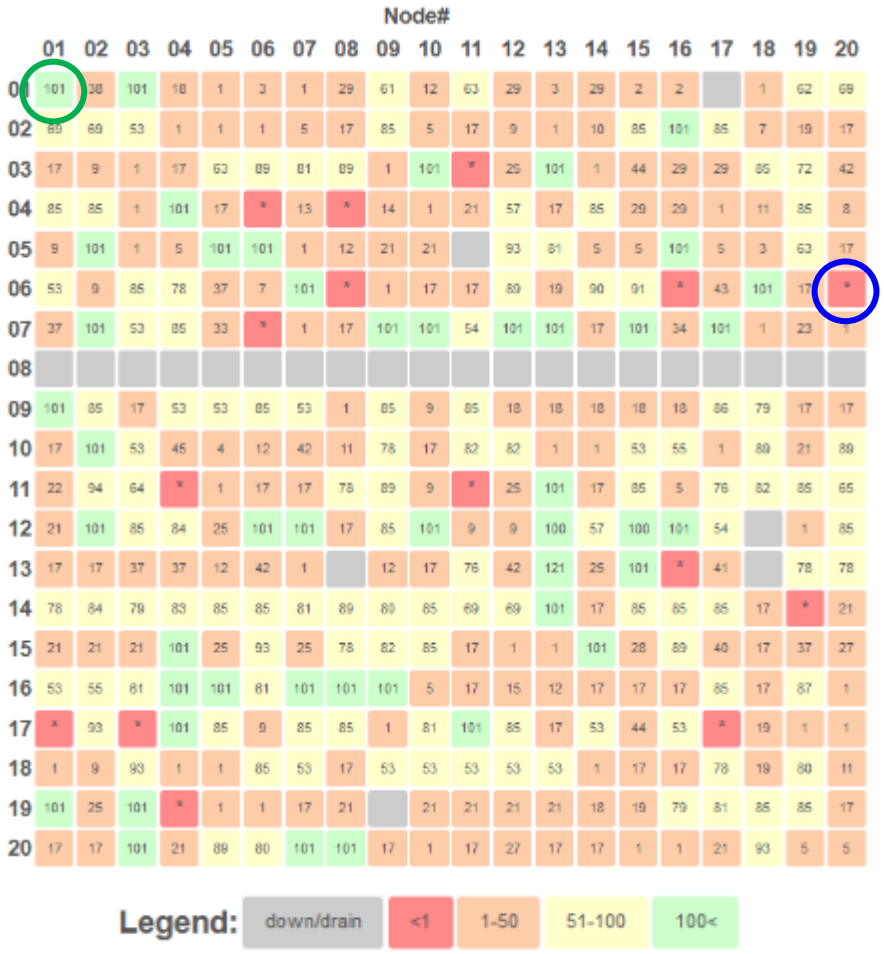
“Compute” partition – nodes heatmap

SKILLPILLS

Cores available (24 cores/node)



Memory available (128 GiB/node)



Best practices and user responsibilities

SKILLPILLS

- Remember Sango is a shared resource: think about others!
- Don't run computational intense programs on the login nodes!
- Limit the resources you request (cores, memory and time) and the number of simultaneous jobs (no more than 50 at a time).
- Remove your temporary files.
- Reach for SCS (it-help@oist.jp or open hours 15h30-17h30, B648) if you have any question or problem, they will gladly help you.

Transfer data to/from Sango

SKILLPILLS

scp stands for secure cp (copy), which means you can copy files across ssh connection.

Copy from Local machine to Server

Syntax

```
scp [-options] localfile [user@server:]file
```

e.g.

```
$ scp /local/path/data.tar.gz
    derpy-derp@sango.oist.jp:/work/data.tar.gz
```

Copy from Server to Local machine

Syntax

```
scp [user@server:]remotefile file
```

Copy from Server 1 to Server 2

Syntax

```
scp [user1@server1:]file [user2@server2:]file
```

Exercise #2: Copy files from/to server

1) Copy `file.fa` to your personal directory

```
scp
  [user]@sango.oist.jp:/work/scratch/skillpill
  /terminal/file.fa
  [user]@tombo.oist.jp:/your/directory/file.fa
```

2) Now, also copy the file `db.fa`

3) Finally copy the folder (creatively) named `/folder`

```
scp -r
  [user]@sango.oist.jp:/work/scratch/skillpill
  /terminal/folder/
  [user]@tombo.oist.jp:/your/directory/folder/
```

-r recursive mode (use it if you want to copy folders)

rsync is faster, more flexible and versatile than cp, rcp and scp. Minimize the work by comparing the files that are already at the destination and only coping only the ones that have changed.

Syntax

```
rsync [-options] /source/path/ /destination/path/
```

It also works over remote servers (ssh):

Syntax

```
rsync [-options] user@server:/source/ /destination/
```

```
rsync [-options] /source/ user@server:/destination/
```

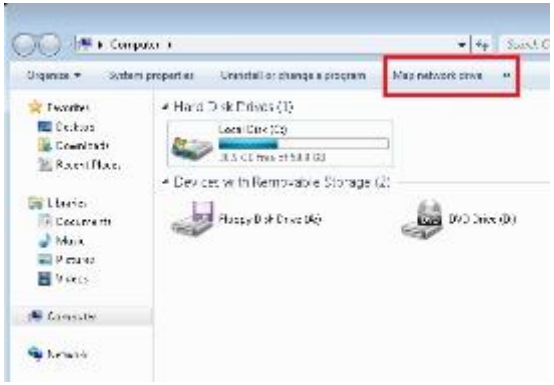
e.g.

```
$ rsync -a -V /local/path/myfolder/
    derpy-derp@sango.oist.jp:/work/myfolder/
```

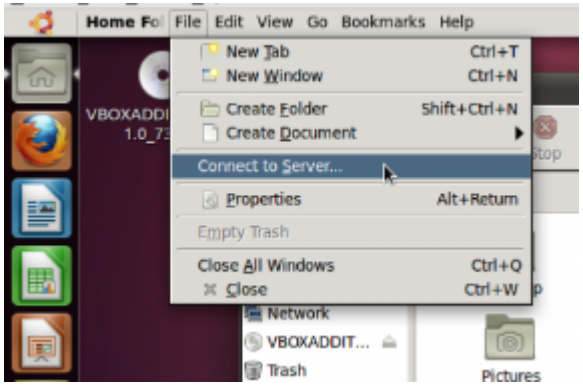
* Options: (-a) Archive mode, (-V) Verbose

Using network drives

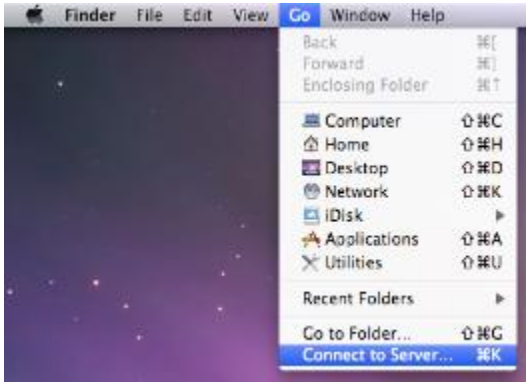
SKILLPILLS



https://groups.oist.jp/it/connecting-software-file-server-windows



https://groups.oist.jp/it/connecting-software-file-server-linux



https://groups.oist.jp/it/connecting-software-file-server-mac

“Map network drive”

Username: OIST\derpy-derp
Password: *****

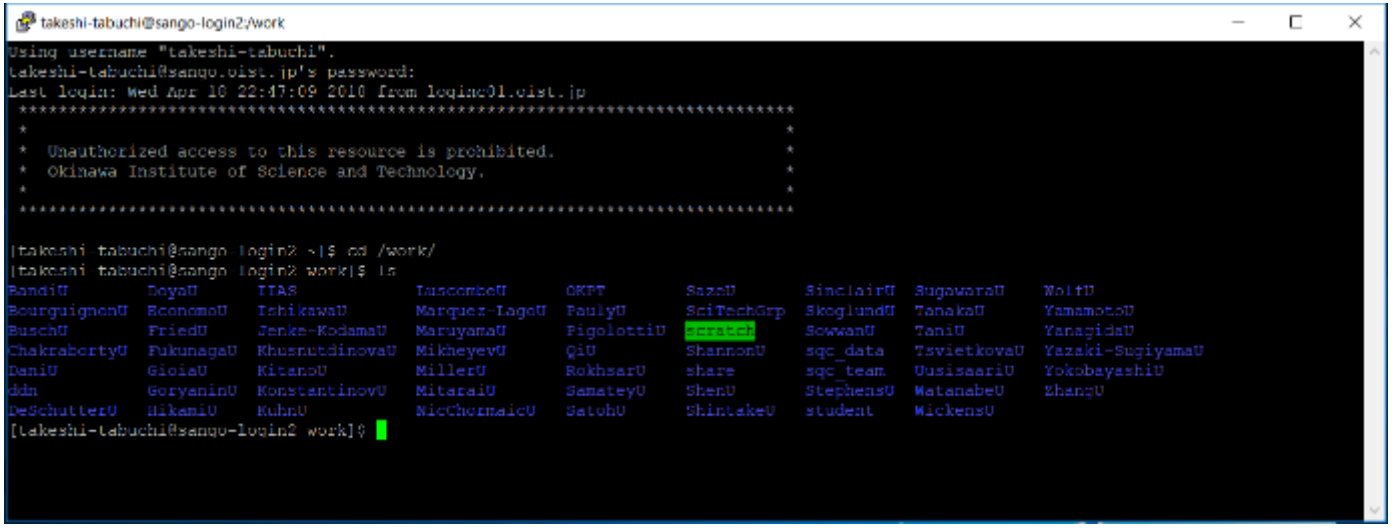
“Connect to Server...”

Domain: OIST
Username: derpy-derp
Password: *****

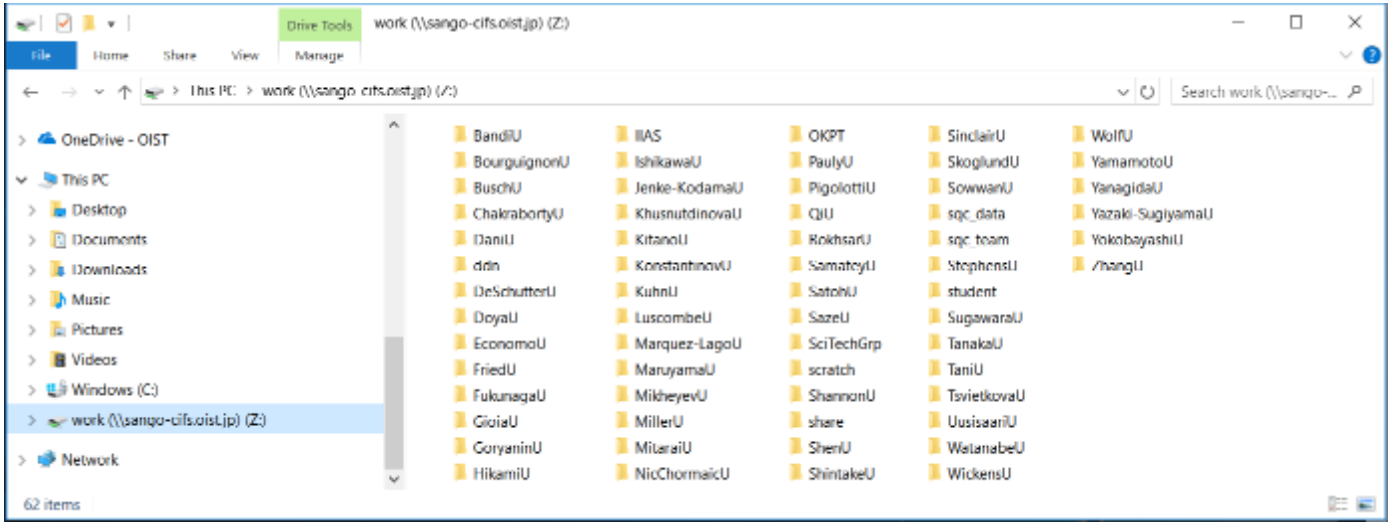
Accessing //sango-cifs.oist.jp:/work

SKILLPILLS

Accessing "/work" using terminal



Mapping "\work" as a network drive



SKILLPILLS

Only work **inside OIST network** (or using VPN).

Note: Sometimes it could lag and not update in real-time.

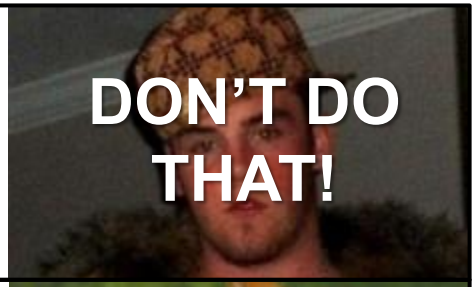


Licensed software:	smb://software.oist.jp/licenced	\\software.oist.jp\licensed
OIST /bucket drive:	smb://bucket.oist.jp/bucket	\\bucket.oist.jp\bucket
Sango /work drive:	smb://sango-cifs.oist.jp/work	\\sango-cifs.oist.jp\work
HPacquire drive:	smb://hpacquire-ui.oist.jp/	\\hpacquire-ui.oist.jp

Using the Sango to run computation jobs

Run computation jobs on the clusters

SKILLPILLS

Run computation jobs directly on login node	
Request resource allocation for the job.	
Allocate resources with the scheduler.	
Run a parallel task across the allocated resources.	

salloc

SKILLPILLS

Syntax `salloc [options] [script] [arguments]`

To return, use: `exit`

To only allocate resources:

e.g.
```
$ salloc -p compute -n 1 -c 1 --mem-per-cpu 1g -t 30
```

* Options: (-p) partition, (-n) number of tasks, (-c) number of cpu (cores) per task, (--mem-per-cpu) memory per cpu, (-t) time.

To allocate resources and run a command (the allocation will be revoked after finishing):

e.g.
```
$ module load ncbi-blast/2.2.30+
$ salloc -p compute -n 1 -t 30 “blastn -query file.fa
   -remote -db nr -out result.out -evalue 1e-30"
```

Exercise #3: Using sango/tombo resources

```
salloc -p compute -n 1 -c 1 --mem-per-cpu 1g -t 30
```

```
squeue -u ${USER}
```

```
[takeshi-tabuchi@sango-login2 ~]$ salloc -p compute -n 1 -c 1 --mem-per-cpu 1g -t 30
salloc: Pending job allocation 2327848
salloc: job 2327848 queued and waiting for resources
salloc: job 2327848 has been allocated resources
salloc: Granted job allocation 2327848

[takeshi-tabuchi@sango-login2 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2327848   compute     bash takeshi-  R       0:03      1 sango10118


[takeshi-tabuchi@sango-login2 ~]$ exit
exit
salloc: Relinquishing job allocation 2327848
salloc: Job allocation 2327848 has been revoked.

[takeshi-tabuchi@sango-login2 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)

[takeshi-tabuchi@sango-login2 ~]$
```

SKILLPILLS

Syntax `srun [options] [script] [arguments]`

To return, use: `exit`

To run a parallel task:

e.g.

```
$ salloc -p compute -n 1 -c 1 --mem-per-cpu=1g -t 30
$ module load ncbi-blast/2.2.30+
$ srun "blastn -query file.fa -remote -db nr -out
     result.out -evalue 1e-30"
```

srun will use the resources allocated with *salloc*.

```
salloc -p compute -n 1 -c 1 --mem-per-cpu 1g -t 30
```

```
squeue -u ${USER}
```

```
srun --pty bash
```

```
[takeshi-tabuchi@sango-login2 ~]$ salloc -p compute -n 1 -c 1 --mem-per-cpu 1g -t 30
salloc: Pending job allocation 2327848
salloc: job 2327848 queued and waiting for resources
salloc: job 2327848 has been allocated resources
salloc: Granted job allocation 2327848

[takeshi-tabuchi@sango-login2 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2327848   compute     bash takeshi-  R       0:03      1 sango10118

[takeshi-tabuchi@sango-login2 ~]$ srun --pty bash

[takeshi-tabuchi@sango10118 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2327848   compute     bash takeshi-  R       0:08      1 sango10118

[takeshi-tabuchi@sango10118 ~]$ exit
exit

[takeshi-tabuchi@sango-login2 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2327848   compute     bash takeshi-  R       0:11      1 sango10118

[takeshi-tabuchi@sango-login2 ~]$ exit
exit
salloc: Relinquishing job allocation 2327848
salloc: Job allocation 2327848 has been revoked.

[takeshi-tabuchi@sango-login2 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)

[takeshi-tabuchi@sango-login2 ~]$
```

If *srun* is invoked when no resources are allocated, it will send a requests for resource allocation:

e.g.

```
$ module load ncbi-blast/2.2.30+
$ srun -p compute -n 1 --mem=1g -o "%j.out"
   -e "%j.err" "blastn -query file.fa -remote
   -db nr -out result.out -evalue 1e-30"
```

* Options: (-p) partition, (-n) number of tasks, (--mem=) total memory, (-o) output file, (-e) error log file.

SKILLPILLS

To open an interactive session in a node:

e.g.

```
$ srun -p compute -c 1 --mem=1g --pty bash
```

* Options: (--pty) connects your tty (terminal) session to a node.

To open an interactive session with a graphical interface (GUI):

e.g.

```
$ ssh -X derp@sango.oist.jp
$ module load matlab
$ srun -p compute -c 4 --mem=8g --x11 --pty matlab
```

* Options: (--x11) forwards to the X11 server (graphic interface).

```
srun -p compute -c 1 --mem=1g --pty bash
```

```
squeue -u ${USER}
```

```
[takeshi-tabuchi@sango-login2 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)

[takeshi-tabuchi@sango-login2 ~]$ srun -p compute -c 1 --mem=1g --pty bash


[takeshi-tabuchi@sango10118 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2327962   compute     bash takeshi-  R       0:02      1 sango10118


[takeshi-tabuchi@sango10118 ~]$ exit
exit

[takeshi-tabuchi@sango-login2 ~]$ squeue -u ${USER}
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)

[takeshi-tabuchi@sango-login2 ~]$
```

Software available

SKILLPILLS

To see the list of all available software on Sango:

Syntax
```
module avail
```

To load a software (and all its dependences/environment):

Syntax
```
module load software/name
```

e.g.
```
$ module load ncbi-blast/2.2.30+
```

To unload a software:

Syntax
```
module unload software/name
```

If you require a specific software that is not already installed in Sango, contact it-help@oist.jp

module avail

SKILLPILLS

```
[takashi-tabuchi@sango-login2 ~]$ module avail

------------------------------------------------------------------------------ /apps/.modulefiles72 -------------------------------------------------------------------------------
abinit/8.2.3                  cde/2011-08-15                Gaussian/09RB01               julia/0.6.0                   nest/2.12.0                   raxml/8.2.11
allines/5.1                   cd-hit/2007-0131              gcc/4.7.4                     jupyter.py/2.7                netcdf/4.3.2                  raxml/8.2.4
allines/6.0                   cd-hit/2009-0427              gcc/4.9.2                     jupyter.py/3.5                netcdf.icc/4.3.2              rdp_classifier/2.2
allines/7.0.5                 cd-hit/2016-0304              gcc/6.2.0                     jupyter.R/3.3.2               neuron/7.3                    relion/2.1
amos/3.1.0                    chimera/1.12                  gdata/2.0.17                  KAT/2.3.2                     NextGenMap/0.4.12             rnaetmer/1.2
applicenoise/v1.27            clearcut/1.0.9                gdb/7.11.1                    kmergenie/1.6950              ngsplot/1.0.2                 rsem/1.3.0
argtable/2.13                 clustalw2.gcc/2.0.12          ghc/6.8.3                     lammps/15May15                NgsP HMM/0.0.1b               RStudio/1.0.143.x332
arpack/3.1.5                  clustalw2.gcc/2.1             git/2.3.2                     lammps/6Mar15                 nwchem/6.6                    rtax/0.984
arpack/3.4.0                  cmake/3.2.1                   glib/2.31.20                  last/548                      opencl.intel/1.2-4.5.0.8      ruse/1.0
ATLAS/3.10.2                  cmake/3.3.1                   glog/0.3.3                    libsv/11.7                    opencv/2.4.11                 samtools/0.1.19
augustus/3.0.3                cmake/3.6.0                   glpk/4.54                     mafft/7.164                   opencv/3.2.0                  samtools/1.0
augustus/3.3                  cmake/3.6.2                   gmsp-gsnap/2014-10-16         mafft/7.220                   opencv.gpu/2.4.11             samtools/1.2
autotools/1.15                comsol/43a                    gnuplot/4.4.4                 mafft/7.266                   OpenFOAM/4.1                  samtools/1.3.1
bamtools/2.3.0                comsol/43b                    gnuplot/4.6.5                 raffl/7.305                   openmpi.gcc/1.10.1(default)   scilab/5.5.2
bamtools/2.4.1                comsol/44                     gnuplot/5.0.2                 VASP/5.3.5                    openmpi.gcc/1.10.3a1          sengrok/1.38.1
bcftools/1.3.1                comsol/50                     gromacs.gpu/5.1               MaterialsStudio/2016          openmpi.gcc/1.6.5             sicsta/4.0
bcraqlc/1261                  comsol/51                     gromacs.icc/5.1               mathematica/10.0              openmpi.gcc/1.7.5             snpeff/4.3q
beagle/1262                   comsol/52                     gsl/1.16                      mathematica/11.0              openmpi.gcc/1.8.6             SOAPdenovo2/r241
beagle/2.1.2                  cpptest/1.1.2                 gsl/2.0                       mathematica/8.0               openmpi.gcc/1.8.7             sourcetracker/0.9.5
beagle.gpu/2.1.2              crystalwave/4.9               hdf5.gcc/1.8.14               mathematica/9.0               openmpi.gcc/1.8.8             SPAdes/3.11.1
beast2/v2.1.3                 csds/2016.0                   hdf5.gcc/1.8.15               matlab/MCR                    openmpi.icc/1.7.5             SPAdes/3.6.2
beast2/v2.4.4                 csds/2017.0                   hdf5.gcc/1.8.16               matlab/R2009b                 openmpi.icc/1.8.6             sra-tools/2.8.2-1
beast2/v2.4.8                 cuda/5.5.22                   hdf5.gcc/1.8.17               matlab/R2011b                 openmpi.icc/1.8.7             stacks/1.34
badtools/v2.25.0              cuda/6.0.37                   hdf5.icc/1.8.14               matlab/R2012a                 openmpi.icc/1.8.8             stacks/1.42
BerkeleyGW/1.2.0              cuda/6.5.14                   hdf5.icc/1.8.15               matlab/R2012b                 paqan/20140814                stacks/1.47
bertini/1.5.1                 cuda/7.0.28                   hdf5.icc/1.8.16               matlab/R2013a                 paqan/20150723                SuiteSparse/4.4.2
binutils/2.24                 cuda/7.5.18                   hdf5.icc/1.8.17               matlab/R2013a_tah             parsnr/1.1.5                  SuiteSparse/4.4.4
binutils/2.25                 cuda/8.0.27                   hmmer/3.1b2                   matlab/R2013b                 parsinsert/1.04               SuiteSparse/4.5.5
binutils/2.27                 cudasw/3.1.1                  HTSeq/0.9.1                   matlab/R2014a                 pasa/r20140417                SuiteSparse.gpu/4.4.5
blat.gcc/35                   cuDNN/2.0_CUDA_6.5            hwloc/1.10.1                  matlab/R2014b                 phenix/1.12-2829              tax2tree/v1.0
blat.gcc/36                   cuDNN/4.0_CUDA_7.0.50         hwloc/1.6.1                   matlab/R2015a                 phenix/1.9-1692               telseq/0.0.1.20160317
blat.icc/35                   cuDNN/5.0_CUDA_7.5.18         igv/2.3.82                    matlab/R2015b                 phenix/dev-1951               tensorflow/1.1.0
blender/2.76                  cuDNN/5.1_CUDA_7.5.18         infernal/1.0.2                matlab/R2016b                 phyldog/2.0-git               tensorflow.gpu/1.1.0
boost/1.55.0                  cuDNN/5.1_CUDA_8.0.27         intel/2013 sp1 update3        mc/4.8.14                     phylobayes/1.6j               tmhmm/2.0c
boost/1.57.0                  cuDNN/6.0a_CUDA_7.5.18        intel/2015 update2            mc/4.8.17                     phylobayes/4.1c               tophat/2.1.1
boost/1.59.0                  cuDNN/6.0_CUDA_7.5.18         intel/2015_update3            MEGA/7.0.20-1                 picard/2.7.0                  tpp/5.0.0
boost/1.61.0                  cuDNN/6.0_CUDA_8.0.27         intel/2015_update6            mercurious/2.0.5              platanus/1.2.1                Trimmomatic/0.33
boost.intel/1.57.0            cufflinks/2.2.1               intel/2016_update2            mercurious/2.2.2.5            platanus/1.2.4                Trinity/2.1.1
boost.intel/1.61.0            cusfaw3.gcc/3.0.3             intel/2016_update4            metabat/2.12.1                pplacer/v1.1                  Trinity/2.3.2
bowtie/1.1.0                  cusfaw3.icc/3.0.3             intel/2017                    MetaSeq/2.1.1                 prank/140603                  TrinoLate/3.1.1
bowtie/1.1.2                  deps_glibc/1.0.0              intel/2017_update1            microbiomutil/201
bowtie2/2.2.3                 dials/1.3.5                   intel.mpi/2015 update3        molgen/50
bowtie2/2.2.6                 dials/452                     intel.mpi/2015 update6        Mothur/1.25.0
BUSCO/1.2                     eigen/3.3.1                   intel.mpi/2016_update2        Mothur/1.39.5
BUSCO/3.0.2                   emboss/6.6.0                  intel.mpi/2016_update4        motioncorr/2.1
bwa.gcc/0.5.9                 exabayes/1.5                  intel.mpi/2017                mpe2.gcc/1.3.0
bwa.gcc/0.6.1                 examl/3.0.17                  intel.mpi/2017_update1        mpe2.gcc/2.4.9b
bwa.gcc/0.7.10                exonerate/2.2.0               intel.mpi/4.1.1.036           mpiBLAST/1.6.0
bwa.icc/0.5.9                 FASTX/1.06                    intel.mpi/5.0.3.048           MrBayes.mpi/3.2.3
bwa.icc/0.6.1                 fasta/35.4.12                 intel.mpirt/2013 sp1 update3  mummer/3.23
bwa.icc/0.7.10                fasta/36.3.7a                 intel.mpirt/2015 update2      mummergpu/2.0
cafte/1.0                     fastqc/0.11.4                 IS_mapper/0.1.4               muscle/3.8.31
catte/rc3                     fastqc/0.11.5                 java-jdk/1.6.0_45             nemd/2.12
catte2/0.7                    fastx_toolkit/0.0.14          java-jdk/1.7.0_67             ncbi-blast/2.2.22
caffe2.gpu/0.7                ffbw.gcc/3.3.4                java-jdk/1.8.0_20             ncbi-blast/2.2.29+
caffe.gpu/1.0                 ffbw.gcc/3.3.5                jellyfish/2.2.7               ncbi-blast/2.2.30+
caffe.gpu/rc3                 ffbw.icc/3.3.4                julia/0.3.5                   ncbi-blast/2.6.0+
cars/5.1.0                    ffbw.icc/3.3.5                julia/0.4.0                   ncbi blast/2.7.1+
ccp4/6.5                      freebayes/0.9.10              julia/0.4.6                   nccl/2.0.5 3 CUDA
cdbtasta/0.99-20100722        freebayes/1.0.2               julia/0.5.0                   necn/0.30.0
```

Exercise #4: Loading modules (software)

1) Try typing `blast` and then press “tab”.

2) Now type `module load ncbi-blast/2.2.30+` , press “enter”.

3) Try again `blast` and then press “tab”.

This should appear now:

```
blastdb_aliastool          blast_formatter            blast_services_unit_test
blastdbcheck               blast_format_unit_test     blast_unit_test
blastdbcmd                 blastinput_unit_test       blastx
blastdbcp                  blastn
```

BASH script

(Unix Shell)

- Bash is a Unix shell and command language.
- It is the default login shell and command processor (interpreter) for most Linux distributions and macOS.
- All commands you can use in Linux terminal can be used in a BASH script and vice versa.

SKILLPILLS

The first line of the file (header) indicates how the file is going to be interpreted an run, in this case by BASH shell:

```
1 #!/bin/bash
```

Bellow that, you can write your script in BASH language:

```
2 cd /your/working/directory/
3 echo “Hello World“
4 #etc, etc, etc
```

Save it as: `myscript.sh`

SKILLPILLS

Important:

OS's represent differently the EOL (End of Line) instruction, set when pressing the "enter ⏎" key.

 **LF** (Left Feed) or \n

 **CR** (Carriage Return) or \r

 **CR** **LF** or \r\n

The Linux interpreter will return an error if the BASH script contains EOL different than **LF**.

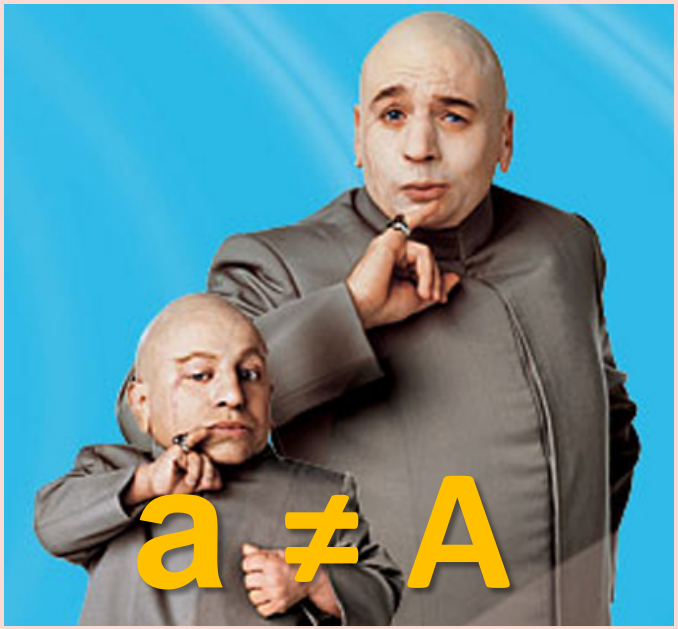
If you write your script outside a Linux text editor be sure to convert all EOL into **LF**.

SKILLPILLS

Important:

Linus is case sensitive.

In Bash script and Linux terminal lowercase and uppercase letters are different.



SKILLPILLS

In linux, file extensions (like .sh) mean nothing, so to be able to execute the script you need to change its properties:

Syntax

```
$ chmod [options] permissions file
```

e.g.

```
$ chmod u=rwx,g=rx,o=r myscript.sh
```

or similarly:

```
$ chmod 754 myscript.sh
```

This means:

- "**u**ser" can "**r**ead"(4)+"**w**rite"(2)+"e**x**ecute"(1)=**7**
- "**g**roup" can "**r**ead"(4)+"not write"(0)+"e**x**ecute"(1)=**5**
- "**o**thers" can "**r**ead"(4)+"not write"(0)+"not execute"(0)=**4**

To execute the script simply type:

```
$ ./myscript.sh
```

chmod advanced tips: group permissions

SKILLPILLS

Some people (rotation students, …) belong to more than one group.
The ‘setgid’ bit tells them to use the parent group ID for new directories.

Syntax

```
$ chmod g+s directory
```

By default, we do not give write permissions to our group members.
This can be changed by setting a different ‘umask’.

```
$ umask 002
```

Put it in your startup script to make it automatic!

Exercise #5: Bash script

1) Write your script as plain text:

```
#!/bin/bash
cd /your/working/directory/
module load ncbi-blast/2.2.30+
blastn -query file.fa -subject db.fa -out
    result.out -evalue 1e-30
```

2) Save it as: `myscript.sh`

3) Make it executable:

```
chmod 777 myscript.sh
```

4) Be a good user, open an interactive session:

```
srun -p compute -c 1 --mem=1g --pty bash
```

5) Finally run your script:

```
./myscript.sh
```

SKILLPILLS

http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Some useful commands:

./	head	let
../	less	read
|	more	for; do; done
>	seq	while; do; done
>>	bc	if; then; else; fi
ls	cat	a=5
wc	grep	i=$(($a + 2))
echo	sed	i++
sort	tr	array=("A" "B" "C")
tail	awk	${array[@]}

Pipelines

SKILLPILLS

Pipelines (**|**) allows you to stream your work. It transfers the output of one command to the input of the next one:

Syntax

```
command1 | command2 | command3
```

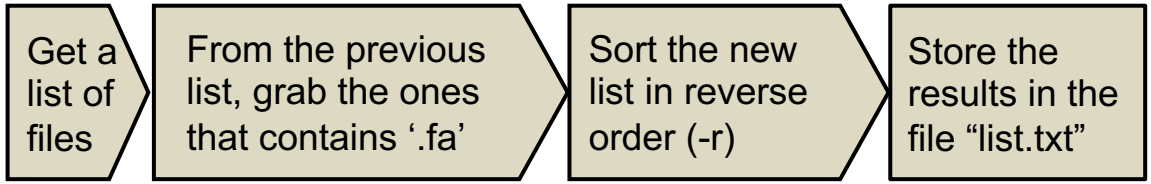
Greater-than signs (**>** or **>>**) allows the output (stdout) of the previous command to be stored into a file, replacing the file's content (**>**) or just appending the data to the end (**>>**).

Syntax

```
command > file
```

e.g.

```
$ ls | grep '.fa' | sort -r > list.txt
```



Exercise #6: Pipelines

1) Open the folder `/folder`

```
cd your/directory/folder
```

2) Type your pipeline:

```
ls | grep '.fa' | sort -r > list.txt
```

3) Check your output:

```
cat list.txt
```

Submit jobs to slurm

workload manager

SKILLPILLS

To use the scheduler first you need to create a BASH script:

Scheduler instructions

```
1  #!/bin/bash
2
3  #SBATCH --job-name=derp_script
4  #SBATCH --partition=compute
5  #SBATCH --time=00:30:00
6  #SBATCH --mem-per-cpu=1G
7  #SBATCH --cpus-per-task=1
8  #SBATCH --ntasks=1
9  #SBATCH --mail-user=%u@oist.jp
10 #SBATCH --mail-type=BEGIN,FAIL,END
11 #SBATCH --output=job_%j.out
12 #SBATCH --error=job_%j.err
13
14 your | super | awesome | bash > script
```

Save the script as: `job_script_name.slurm`

https://groups.oist.jp/scs/getting-started

Parameter	Value in the example	Explanation
`-J, --job-name=`	`test_ucsc`	name of the job as it will appear in the output of the `squeue` command
`-p, --partition=`	`compute`	name of the partition where the job will be executed. See "SLURM partitions in Sango" for more information about Sango partitions.

`-t, --time=`	`00:30:00`	limit on the total run time of the job. If not set the default job runtime value will be applied (see "SLURM partitions in Sango" for the list of default runtimes)
		Acceptable formats are:
		`days-hours:minutes:seconds`
		`days-hours:minutes`
		`days-hours`
		`hours:minutes:seconds`
		`minutes:seconds`
		`minutes`

`--mem-per-cpu=`	`1G`	minimum memory required per allocated core (the default value is 4g). This is a mandatory parameter when `--mem=` is not specified.
`--mem=`	`8g`	total real memory required per allocated node (the default value is 4g). This is a mandatory parameter when `--mem-per-cpu=` is not specified.
`-n, --ntasks=`	`1`	maximum number of tasks that the SLURM controller will launch on the allocated resources.
`-c, --cpus-per-task=`	`1`	number of CPUs (cores) per task that SLURM should allocate.
`--mail-user=`	`john-doe@oist.jp`	user to receive email notification of job state changes

SKILLPILLS

To submit it to the scheduler use:

Syntax `sbatch [options] script [arguments]`

e.g.
```
$ sbatch job_script.slurm
```

The scheduler will allocate resources according to your request, depending on the availability and priority (first-come-first-served & less demanding jobs = higher priority).

Scheduler will normally create 2 files (XXXX is the job ID):

job_XXXX.out

Stores every printed output (e.g. *echo*) of the script.

job_XXXX.err

Contains all the error messages.

Exercise #7: Submit jobs to the scheduler

1) Create the script of your job as plain text:

```
#!/bin/bash

#SBATCH --job-name=derp_script
#SBATCH --partition=compute
#SBATCH --time=00:10:00
#SBATCH --mem-per-cpu=10M
#SBATCH --cpus-per-task=1
#SBATCH --ntasks=1
#SBATCH --mail-user=%u@oist.jp
#SBATCH --mail-type=BEGIN,FAIL,END
#SBATCH --output=job_%j.out
#SBATCH --error=job_%j.err

a=1
echo "The value of a is "$a
```

2) Save the script as: `job.slurm`

3) Submit the job to the scheduler:

```
sbatch job.slurm
```

4) Check the output file and the error log:

```
cat job_[ID].out
```

```
cat job_[ID].err
```

SKILLPILLS

To check the status of your jobs in the queue:

Syntax
```
squeue [options]
```

e.g.
```
$ squeue -u ${USER}
```

To cancel one (or more) job(s):

Syntax
```
scancel [options] [jobID[_arrayID][.stepID]...]
```

e.g.
```
$ scancel 1337_1 1337_2 1340 1345
```

```
$ scancel -t PENDING -u ${USER} -p compute
```

```
$ for i in {1337..1345}; do scancel $i; done
```

Exercise #8: Cancel jobs

1) Create a new script:

```
#!/bin/bash

#SBATCH --job-name=5min
#SBATCH --partition=compute
#SBATCH --time=00:10:00
#SBATCH --mem-per-cpu=10M
#SBATCH --cpus-per-task=1
#SBATCH --ntasks=1
#SBATCH --mail-user=%u@oist.jp
#SBATCH --mail-type=BEGIN,FAIL,END
#SBATCH --output=job_%j.out
#SBATCH --error=job_%j.err

echo "Wait for 5 minutes."
sleep 300
echo "We just wasted 5 minutes."
```

2) Save the script as: `job2.slurm`

3) Submit the job to the scheduler:

```
sbatch job2.slurm
```

4) Let’s see the status of your job:

```
squeue -u ${USER}
```

5) Now cancel your job:

```
scancel [job_ID]
```

6) Check the output file and the error log:

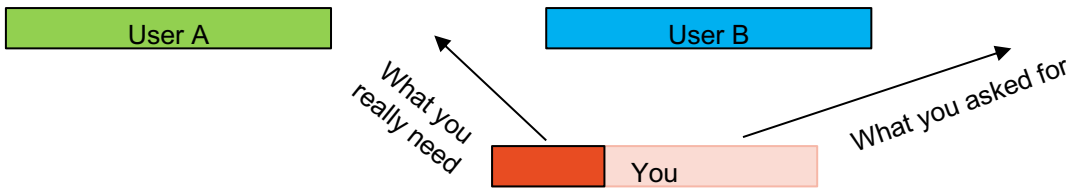
```
cat job_[ID].out
```

```
cat job_[ID].err
```

Resource usage

SKILLPILLS

- How much time can we use ?
  - The compute partition has a limit of time. Talk to the SCDA admins if you need to run a program for a long time ?
  - `--time` too short, and your job is cut, but
  - `--time` too long, and you wait.



- How much resources can we use ?
  - If many nodes are idle, we are wasting electricity, but,
  - if one person uses all the nodes, everybody is waiting.

Use script to submit several jobs

Method 1: One by one

SKILLPILLS

First create your .slurm jobs normally:

```
job_script_1.slurm
job_script_2.slurm
job_script_3.slurm
```

Now write your .sh script and then execute it:

Simple way:

```
1 #!/bin/bash
2
3 sbatch job_1.slurm
4 sbatch job_2.slurm
5 sbatch job_3.slurm
```

Slightly better way:

```
1 #!/bin/bash
2
3 for i in job_*.slurm ; do
4   sbatch $i
5 done
```

Method 2: sbatch array

SKILLPILLS

```
1  #!/bin/bash
2
3  #SBATCH --job-name=ArrayJob
4  #SBATCH --output=job_%j_%A-%a.out
5  #SBATCH --error=job_%j_%A-%a.err
6  #SBATCH --partition=compute
7  #SBATCH --mem=1g
8  #SBATCH --array=1-6%3
9
10 Arr=(0.05 0.075 0.1 0.25 0.5 1.0)
11
12 echo ${Arr[$(( $SLURM_ARRAY_TASK_ID
    % ${#Arr[@]} ))]}
```

This will run an array of 6 jobs (indexes 1 to 6), but limited to 3 simultaneous jobs.

```
$SLURM_JOBID=1337
$SLURM_ARRAY_JOB_ID=1337
$SLURM_ARRAY_TASK_ID=1

$SLURM_JOBID=1338
$SLURM_ARRAY_JOB_ID=1337
$SLURM_ARRAY_TASK_ID=2
…
```

* If you include *srun* in the script, a new job step will be created to run it within the task.

This command will simply print one of the elements of the array “Arr” depending on the ID of the current task.

```
%j ≈ $SLURM_JOBID
%A ≈ $SLURM_ARRAY_JOB_ID
%a ≈ $SLURM_ARRAY_TASK_ID
%s ≈ $SLURM_ARRAY_TASK_STEP
```

Method 3: auto-create the jobs from template

SKILLPILLS

Create your template:

```
#!/bin/bash

#SBATCH --job-name=Job_VAR
#SBATCH --output=job_%j.out
#SBATCH --error=job_%j.err
#SBATCH --partition=compute
#SBATCH --mem=4g

a=VAR

grep ">" $a.fa > $a.out
```

Save it: **template.slurm**

Create your script:

```
#!/bin/bash

arr=("f_1" "f_3" "f_4")

for i in ${arr[@]}; do
  cat template.slurm |
     sed s/'VAR'/$i/g >
     job_$i.slurm
  sbatch job_$i.slurm
done
```

Save and execute: **script.sh**

Method 4: auto-create the jobs from scratch

SKILLPILLS

Create your script:

```
#!/bin/bash

arr=("f_1" "f_3" "f_6")
for i in ${arr[@]}; do
  echo '#!/bin/bash' > job_$i.slurm
  echo '#SBATCH --job-name=Job_'$i >> job_$i.slurm
  echo '#SBATCH --output=job_%j.out' >> job_$i.slurm
  echo '#SBATCH --error=job_%j.err' >> job_$i.slurm
  echo '#SBATCH --partition=compute' >> job_$i.slurm
  echo '#SBATCH --mem=4g' >> job_$i.slurm
  echo 'grep ">" '$i'.fa > '$i'.out' >> job_$i.slurm
  echo '' >> job_$i.slurm
  sbatch job_$i.slurm
done
```

Save and execute: **script.sh**