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

- https://www.xquartz.org/

- (warning: needs a reboot)

This is not a SKILLPILLS

Mini Course:
TERMINAL

Lecture III:

Interacting with Deigo

OIST

February 2022

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

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

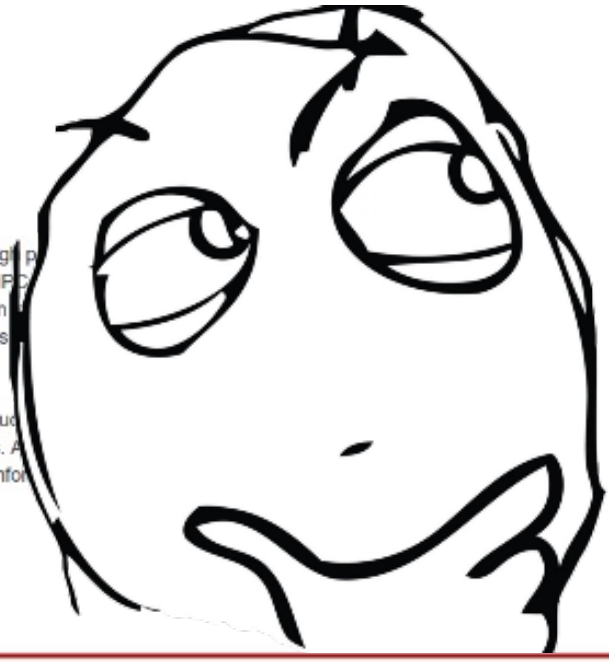
Scientific Computing & Data Analysis Section

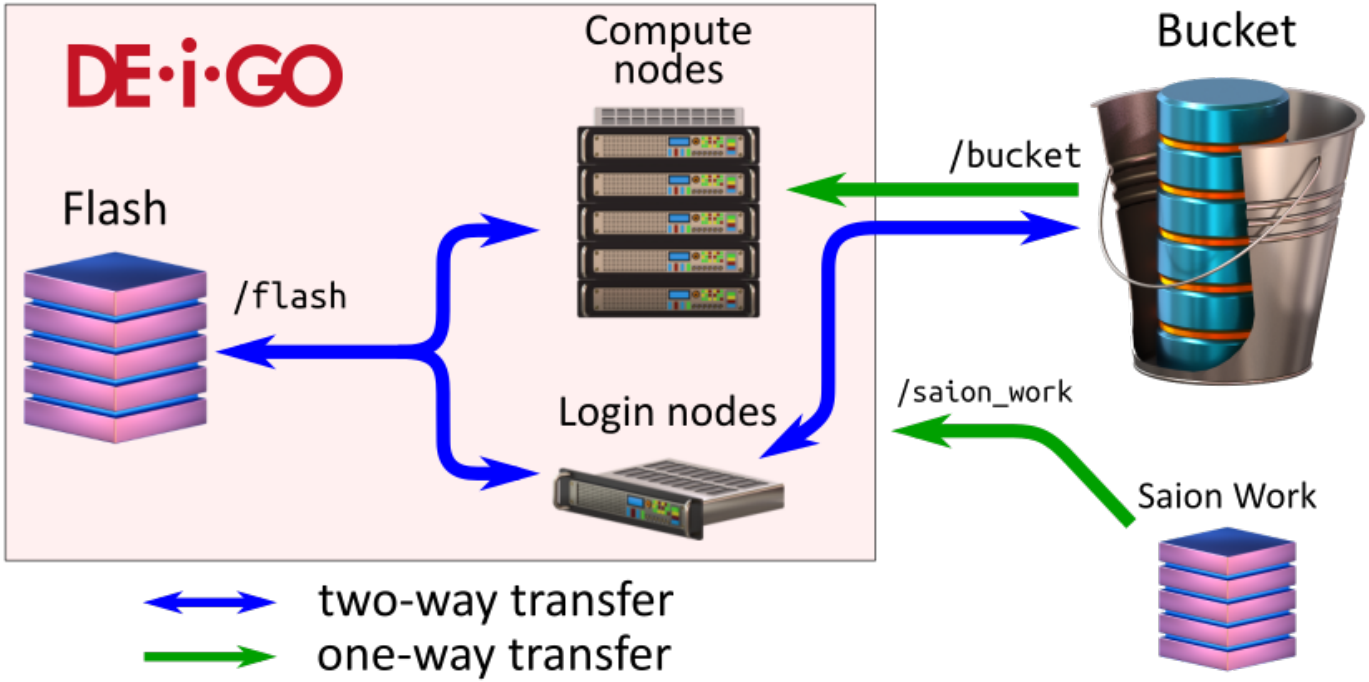
Getting started

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.

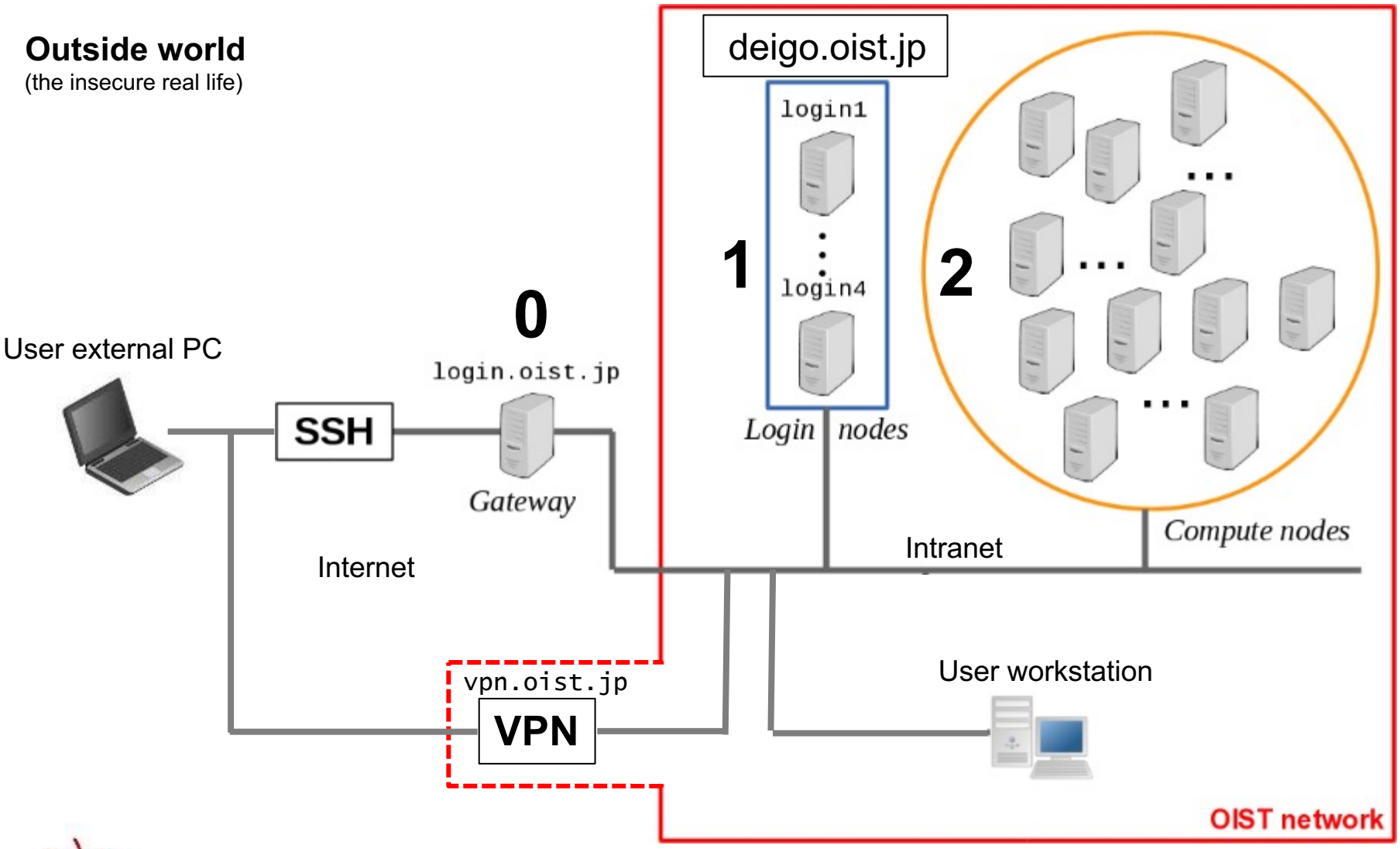
There are two HPC clusters at OIST, Sango and Tombo. Sango is OIST main produc
cluster used for testing, seminar, workshop, demo, and on-demand specific events. A
without restriction. All OIST HPC clusters use Linux Operating System. For more infor
computing resources page.





https://groups.oist.jp/scs/deigo

The OIST network

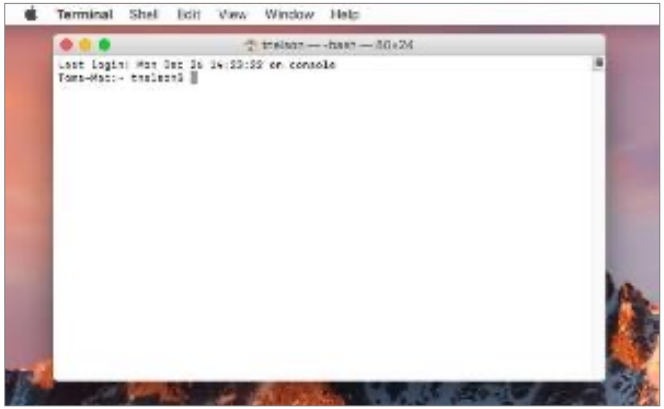


Popular terminal clients on popular OSes

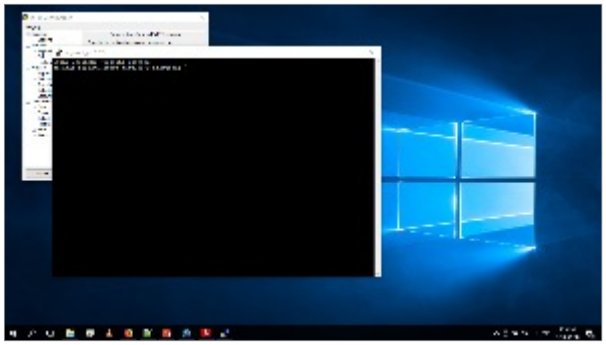
SKILLPILLS

Terminal

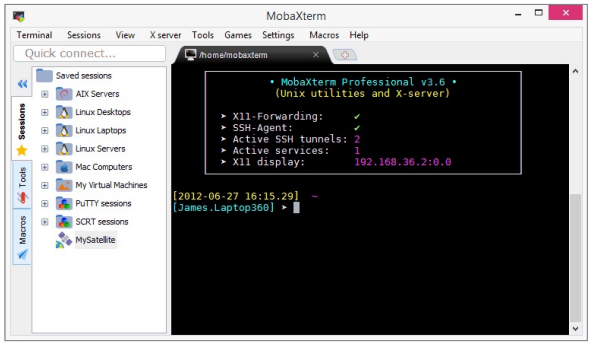
Terminal



Putty



MobaXterm



Secure Shell (SSH) is a tool for using network services securely over an unsecured network using encrypted communications and a public/private key system for identifying oneself. We use it to **connect to remote systems** such as our computing cluster and to **copy files over the network**.

Syntax

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

e.g.

```
$ ssh tida-name@login.oist.jp
$ ssh deigo squeue
```

Exercise #1: Login

1) Try to login into login.oist.jp and deigo.oist.jp

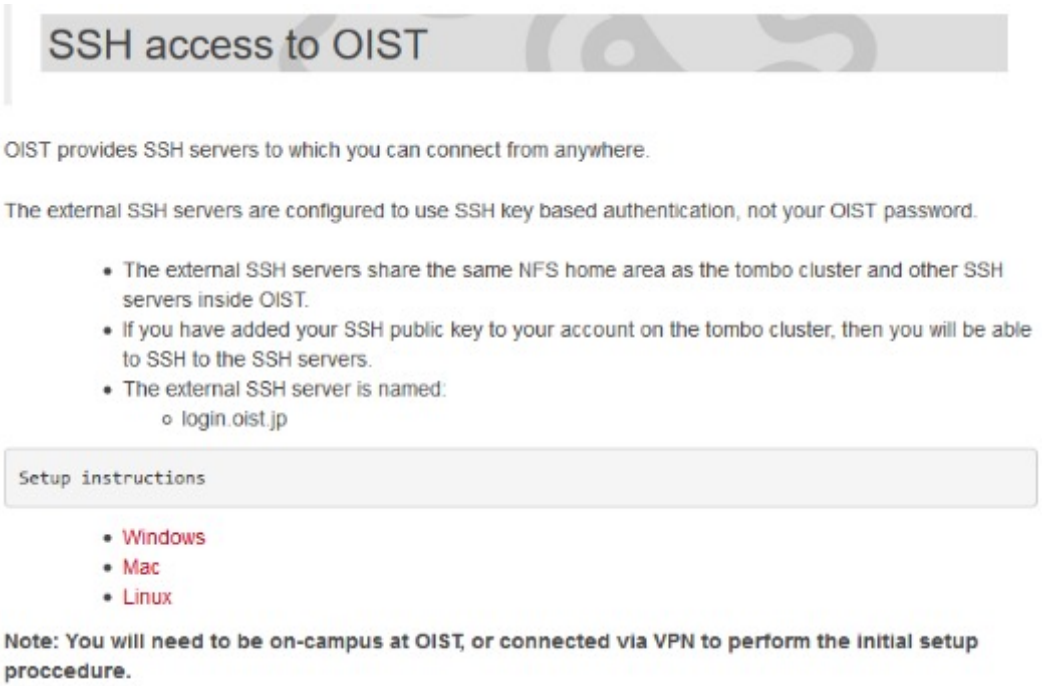
```
ssh tida-name@login.oist.jp
```

```
ssh tida-name@deigo.oist.jp
```

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

The external OIST SSH servers (such as `login.oist.jp`) are configured to only allow SSH key based authentication: you can not use your 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.

Imagine you have a treasure protected by many guards. You need to delegate access to a trusted person. How can the guard recognise your delegate?

Public key given to all guards

Instruction to the guards:

"Give access to somebody who will show you where the public key came from"

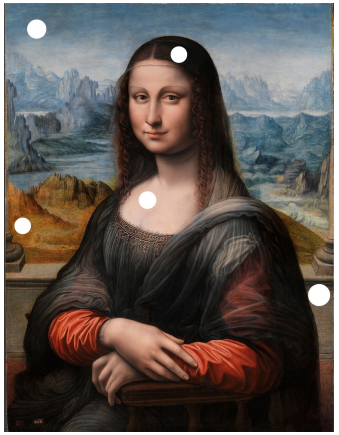
Imagine you have a treasure protected by many guards. You need to delegate access to a trusted person. How can the guard recognise your delegate?

Public key given to all guards

Instruction to the guards:

"Give access to somebody who will show you where the public key came from"

Private key (secret)



In SSH, the agent does not need to see the private key to know it matches the public key.

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:

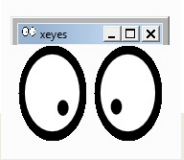
```
$ ssh -A tida-name@login.oist.jp
```

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

```
$ ssh -Y tida-name@deigo.oist.jp
```

Only do this on a trusted network

Run `xeyes` to test your session




```
$ xeyes
```

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 deigo*
    User tida-name
    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/
- Want to log in `deigo` directly from outside? Check the `ProxyJump` configuration directive in `.ssh/config`.

Exercise #2: Deeper inception login

1) Try to login and login and login...

```
ssh-add
```

```
ssh –A tida-name@deigo.oist.jp
```

```
ssh deigo-login1
```

```
ssh deigo-login2
```

2) How many times do you need to exit ?

Transfer data to/from Deigo

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
    tida-login@deigo.oist.jp:~/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 #3: Copy files from/to server

1) Copy `AliceInWonderland.txt` to your personal directory

```
scp \
[user]@deigo.oist.jp:/apps/unit/GradschoolD/terminal
  /material/less_test/AliceInWonderland.txt \
.
```

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

```
scp -r \
[user]@deigo-
  login1.oist.jp:/apps/unit/GradschoolD/terminal \
[user]@deigo-login2.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` 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/
    tida-name@deigo.oist.jp:~/myfolder/
```

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

If you are desperate...

- https://magic-wormhole.readthedocs.io/

Module

```
ml bioinfo-ugrp-modules DebianMed magic-wormhole
```

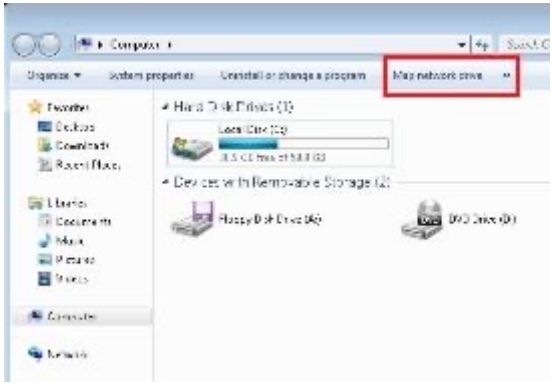
Syntax

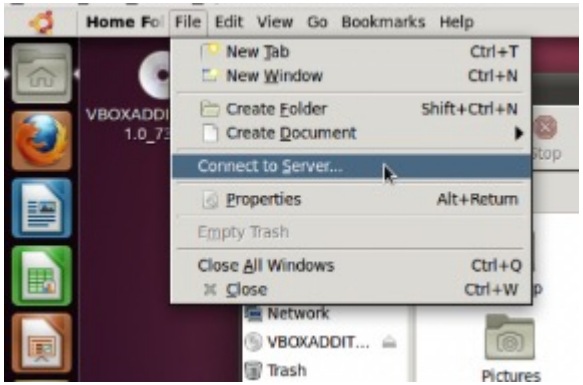
```
wormhole send yourfile
```

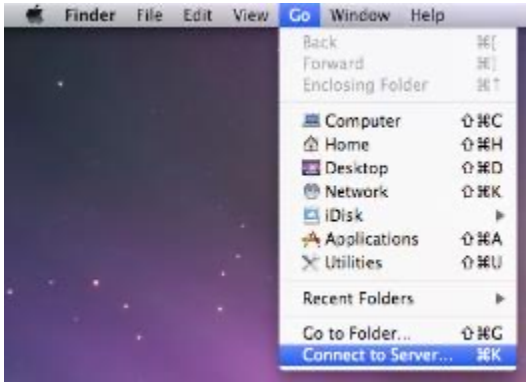
```
wormhole receive
```

Try to send a file to your neighbour !

Using network drives







https://groups.oist.jp/scs/copy-and-move-data

“Map network drive”

Username: OIST\tida-name
Password: *****

“Connect to Server...”

Domain: OIST
Username: tida-name
Password: *****

Network drives

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
HPacquire drive:	smb://hpacquire-ui.oist.jp/	\\hpacquire-ui.oist.jp

Software available

Software modules

SKILLPILLS

To see the list of all available software on Sango:

Syntax

```
ml avail
ml spider software
```

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

Syntax

```
ml software/version
```

e.g.

```
$ ml ncbi-blast/2.10.0+
```

To unload a software:

Syntax

```
ml unload software/name
```

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

Exercise #4: Loading modules (software)

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

2) Now type `ml ncbi-blast/2.10.0+` and 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
```

The Bioinformatics user group (https://github.com/oist/BioinfoUgrp) has prepared a couple of general-use software modules:

- Better FTP clients: `ncftp`, `lftp`
- A ZIP implementation that better handles large borken ZIP files: `p7zip`
- Perl's `rename` command (example: `rename 's/OSIT/OIST/' *txt`)
- And of course, the `magic-wormhole`.

```
ml bioInfoUgrp DebianMed
ml <the name of the tool>
```

- In principle, we can easily add anything that exists as a small Debian package.

Runing computation jobs on Deigo

Inside Deigo

SKILLPILLS

https://groups.oist.jp/scs/deigo

Partition	Total cores	Cores per node	Memory per node	Memory per user	Max cores per user	Max time per job
short	66552	36-128	500GB	6500G	4000	2h
compute	45568	128	500GB	7500G	2000	4 days
largejob	12800	128	500GB	—[1]	—[1]	—[1]
largemem	2424	36-40	500-750GB	—[2]	—[2]	—[2]
bigmem	32	32	3T	—	—	—

1: largejob allocations are decided by the SCC.

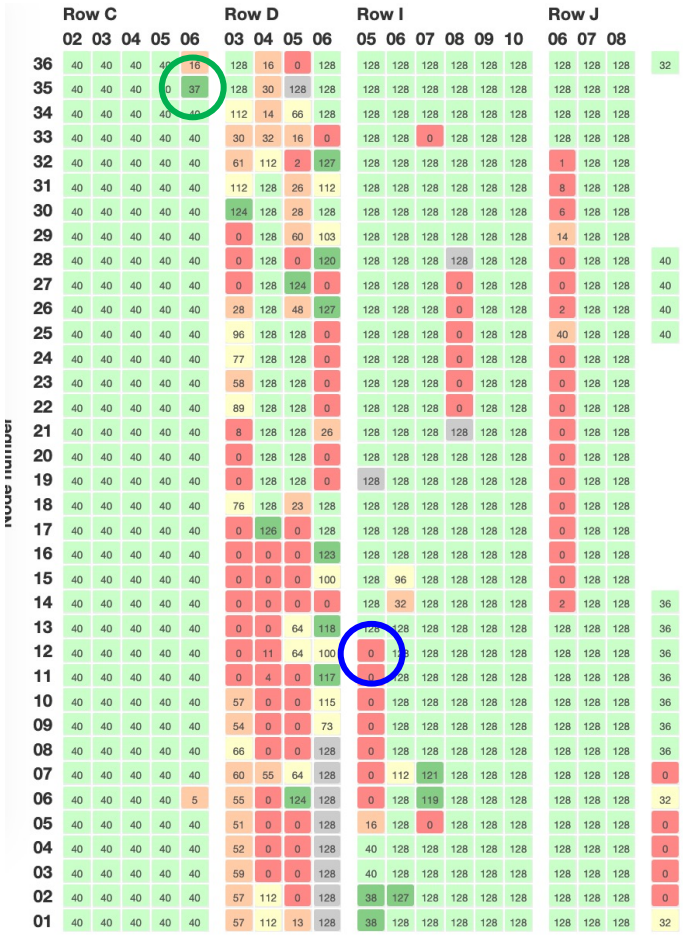
2: largemem limits you to ~5 nodes worth of resources in total.

“Compute” partition – nodes heatmap

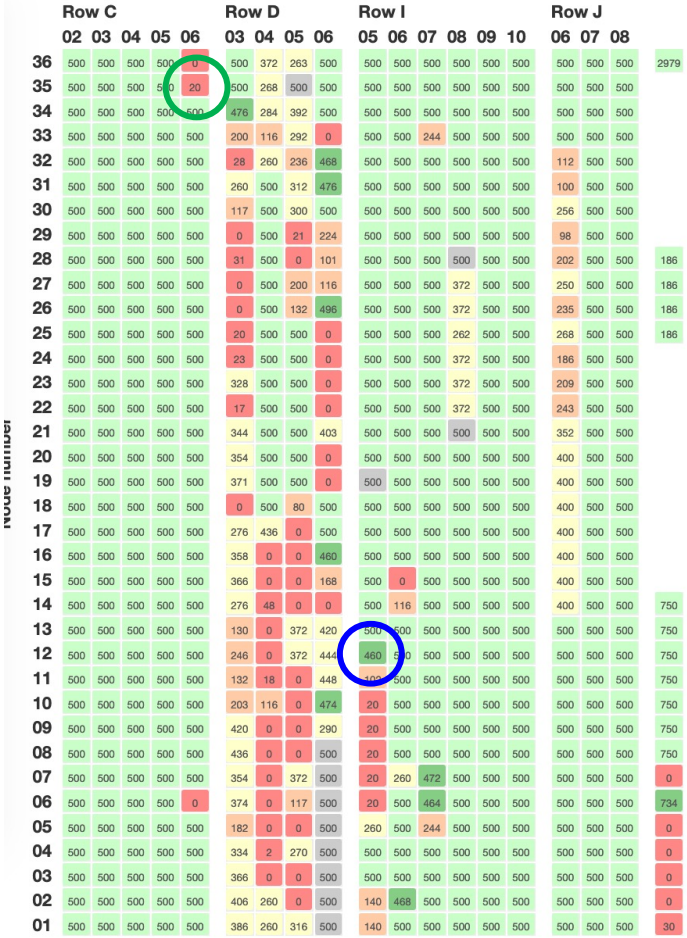
SKILLPILLS

https:/highsci.oist.jp

Cores available



Memory available



Best practices and user responsibilities

- Remember Deigo 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.

Run computation jobs on the clusters

SKILLPILLS

Run your Jobs with

Run computation jobs directly on login node	**DON'T DO THAT!**
Run jobs with the SLURM scheduler.	`sbatch`
Run interactive commands with SLURM.	`srun`

SKILLPILLS

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

To return, use: `exit`

srun sends a requests for resource allocation that you can customise:

e.g.

```
$ ml ncbi-blast/2.10.0+
$ 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 -Y tida-login@deigo.oist.jp
$ ml matlab
$ srun -p compute -c 4 --mem=8g --x11 --pty matlab
```

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

Tip: use `eog` to visualise pictures and `evince` to visualise PDFs.

BASH scripting for SLURM

(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.

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.

$$a \neq A$$

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/
echo hello I am in $(pwd)

```

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
```

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
```

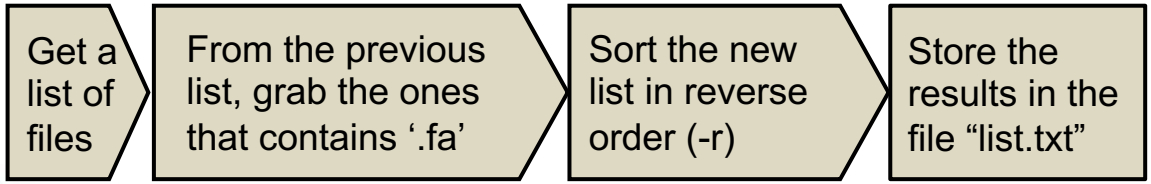
Greater-than signs (**>** or **>>**) allows the output (stdout) of the previous command to be stored into a file, replacing the file's content (**>**) or just appending the data to the end (**>>**).

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 '.txt' | sort -r > list.txt
```

3) Check your output:

```
cat list.txt
```

SBATCH header in shell scripts

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 Deigo 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 partition Deigo 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

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).

SLURM 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}
```

Tip: you can log in a compute node where you have jobs running.

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:

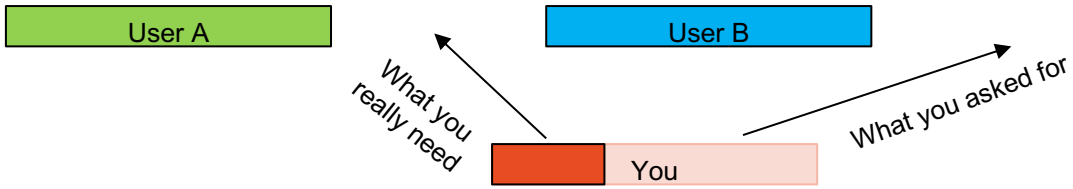
```
scancel [job_ID]
```

6) Check the output file and the error log:

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

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

- 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 sbatch arrays to submit several jobs

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 -1 ]}
```

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
```