Bianca Introductory Course
Hands-on Exercises

Your best friend:
http://uppmax.uu.se/support-sv/user-guides/bianca-user-guide

Logging in — Marcus

Command Line

The traditional way to access high-performance compute clusters is through the Linux command-line. The command line is a text-based environment where a program called **bash** interprets and executes the commands you enter. Once you are familiar with Linux and comfortable with bash, this is an extremely powerful and versatile way to work.

1. Follow the instructions in the bianca User Guide to log in to the sens2019503 project cluster on bianca in text or command-line mode.
   a. You can use any ssh-program in all normal platforms (Windows, Mac, Linux). A popular Windows program is **MobaXterm**.
   b. Your username will be <username>-<projid>, e.g. marcusl-sens2019503@bianca.uppmax.uu.se
   c. As password you use your normal UPPMAX password directly followed by the six digits from the UPPMAX second factor. E.g. if your password is "VerySecret" and the second factor code is 123 456 you would type VerySecret123456 as the password in this step.
   d. If the password is correct you will get a message of your projects login node status. It can be "up and running" or "down". If it is down it will automatically spin up, but this takes a few minutes.

```
Marcuss-MacBook-Pro-3:NatureSpirit marcus$ ssh marcusl-sens2018569@bianca.uppmax.uu.se
marcusl-sens2018569@bianca.uppmax.uu.se's password:
Last login: Mon May 14 13:19:13 2018 from emp234-49.eduroam.uu.se
################################################################################
# Login node NOT up! Boot initiated. Just wait for prompt                      #
# or login again later. This can take from 2 to 6 minutes                      #
################################################################################

********************************************************************************
* Notice! No second factor if you use password, but we recommend ssh-keys. *
********************************************************************************
##
```

   e. Then you will be automatically redirected to login at your project's private login node. You will once again have to give your UPPMAX password to log in to the node, but without the two factor authentication code this time.
2. Keep this log-in window open, as you will be using it again soon.

Graphical

Unlike on most systems, there is no way to run a graphical program from the command-line. If you want to use graphical programs, you must log in using the web interface powered by ThinLinc.

1. Using a standard web browser, go to https://bianca.uppmax.uu.se
2. Supply your username (with project) and password with second factor, same as in the command line mode.
3. Supply your username (without project) and password (without second factor) to reach the login node.
4. You should soon see something resembling a desktop.
5. It is important to know the difference between disconnecting a session and logging out.

a. If you choose to log out with “end session”, the system will attempt to close all your open windows and programs.

b. If you disconnect the session, everything will remain as you left it.

In the remainder of the course, you may choose to use either the command-line interface or the graphical interface. We recommend learning to use the command-line if you are not already comfortable with it — it really is worth your time.

Using the Wharf — Jonas

Direct sftp.

If you look into the directory /proj/sens2019503/files_for_transfer, you will see files that you want to get out of bianca (Make sure you should do this before trying). In order to do so, you need to put them onto the wharf. That is the area designated to have access to computers other than bianca. Each wharf is personal and you can only access your own. Now try getting the files to your wharf area. For this project it should be located at

/proj/nobackup/sens2019503/wharf/$USER/$USER-sens2019503,

where $USER is your user name. Copy the files there. Now they are ready for transfer.

Next, you need to log into the computer to which you want the files moved. After that move to a place where you want your files located. Then you use sftp to transfer your files from the wharf by typing (ignore the leading $, this indicates the start of the command line prompt).

$ sftp -r $USER-sens2019503@bianca-sftp.uppmax.uu.se:$USER-sens2019503

You will need to user two factor verification for this. Also, remember the -r after sftp, or it won't take directories.

This will get you to a new prompt

To get the actual files you need to type

$ get <files you want> (or get * for everything)
If you want to reverse the process, i. e. put stuff on bianca, you use the same sftp, but instead type

$ put <files you want> (or get * for everything in your current directory)

Now try to put stuff from rackham onto bianca and see if you can get them to your home there.

Convenience scripts.

Now that you know how it works, you might want to try my scripts for doing the exact same thing. They are located in

/proj/sens2019503/moving_scripts/

Try using them to move stuff to and from the wharf on bianca. Also, try to use the wharf to get the scripts put-bianca.sh and get.bianca.sh to rackham (where they should be).

File management within bianca — Marcus

Move delivered files into a different directory

Many of you will receive sequence data from NGI. NGI puts its data deliveries on the Grus storage system, which can be reached via sftp (according to the Grus User Guide) and via the Deliver function managed in SUPR (according to the Deliver User Guide). The latter will place data directly on bianca for you.

Because only NGI can put data on Grus and create delivery projects, we can't provide an exercise to demonstrate this step, but it can be a good idea to move the contents of a delivery project to a directory with a more meaningful name.

1. Find a data delivery in **/proj/sens2019503** named **delivery12345**.
2. Create a new directory in the project area with a name representing the data (e.g. marcusl_toenail_snpseq)

3. Copy the data in **delivery12345** to the new directory. In the real world, you would **move** the data.
4. Create a file in **delivery12345** indicating where you moved the data. This can be very useful when someone wants to know where a particular delivery ended up.

Intermediate files in nobackup directory

The directory **/proj/sens2019503/nobackup** is ignored by the backup service. This is very important if backup is to function.
The backup service is capable of writing a large amount of information every second, but very large amounts of files and very large numbers of changes in files can cause problems.

It can be difficult to know what should and should not be subject to backup.

The UPPMAX rule of thumb is, when in doubt, to only backup raw sequence data and your scripts. In the worst case, if you delete everything by accident or a landslide causes the computer hall to slide into Fyrisån, this will be enough to eventually recover your results.

Final or semi-final results, which you will not be writing to very often (if at all), can safely be placed under backup as well.

*Intermediate files that are actively being written to must **not** be placed in backup.*

1. Make a script in **/proj/sens2019503/<username>** or **$HOME** that reads input files from the backup area and writes output files in a subdirectory of **/proj/nobackup**. An example (toy) script that you can modify has been placed in **/proj/sens2019503/misc**
2. Tar the output files together and copy/move to **/proj/sens2019503/<username>/finalresults.tar.gz**

Temporary files in $SNIC_TMP

The project file area is a network drive and is visible from all nodes on Bianca. If your job is doing work on temporary files that does not need to be visible to any other jobs, you can (and should) avoid using the cluster's network and the network drive by using the local disk present on every node.

A unique directory on the local disk is created for every job* and is found at **$SNIC_TMP**

1. Convert your script that reads input files in from the backup area and to write its output files to a subdirectory of **$SNIC_TMP**, e.g. **$SNIC_TMP/<username>**

* Note that login nodes have a local disk, but no unique directory is pre-made for you.

Programs, packages, scripts on bianca — Jonas

Module system

On a large scale system like bianca, there will be many different softwares, and they might have need of different versions of the same prerequisite. Many versions of the same software is often impossible to have installed at the same time. Thus, UPPMAX have a system called Lmod, that keeps every installation separate and unavailable until you load the module for it. There are several commands used to load and get information about the modules.

You can find more about the commands on their homepage, but here are a few of the most commonly used ones:

List the modules loaded:
$ module list

To find out what modules are available to be loaded:
$ module avail

To load packages:
$ module load <package1> <package2> ...

To unload packages:
$ module unload <package1> <package2> ...

To get a list of all the commands that module knows about:
$ module help

To search for modules:
$ module spider <text>

Sometimes you will see that the module you are looking for is installed but needs the module bioinfo-tools to work. This is because bioinformatic tools are so many that having them all available would make the list very long. If that happens, in the exact same fashion, just load the module bioinfo-tools before the module you want to load.

Try using the commands above to get software you need for your research to run.

Custom install from web: Download on rackham, then copy to bianca

Sometimes you want to use software that simply isn't installed yet. Also, sInce there is no internet connection, all downloads and installations need to be performed on a different computer. This can in theory be any computer, i. e. your laptop. However, since rackham and bianca have the exact same drivers and modules installed, downloading (or writing) your desired program to rackham and doing everything needed there first will probably work better. After that you can move your files to bianca and install your software there.

Submitting jobs — Marcus

You can get a lot of work done right on the login nodes, but a single node is not much more than an ordinary workstation. For true High Performance Computing, when you have a workload that you need to repeat many times or something that will take a significant amount of time to finish, you must submit jobs that the Slurm job scheduler runs on compute nodes.

There are two types of jobs: batch jobs and interactive jobs.

sbatch

A batch job is a description of a work package, including a resource booking and commands to execute. Typically, a batch job is fully described in a job script. Batch jobs are submitted to the queue with the **sbatch** command and will execute without your immediate oversight, producing results and an output file… eventually.

1. Copy the file /proj/sens2019503/submit_templates/job_template.sh to a place your home directory. It is a good idea to keep this template around as a base for writing new job scripts.
2. Make another copy in your working directory and give it a new name, e.g. "myfirstjob.sh"
3. Open "myfirstjob.sh" for editing.
4. As you can see, this is an ordinary bash script with some special lines near the top. Lines starting with "#SBATCH" are parameters given to Slurm that help define the job.
    - -A sens2019503: sets the project which "pays" for the CPU time.
    - -p core: sets the partition to allow less than 1 full node to be booked. The alternative is "-p node".

- -J Template_script: sets the name of the job to “Template_script”. Changing the name of your jobs to something meaningful can help you manage your jobs.
- -t 01:00:00: sets the maximum runtime of this job. The requested resources will be booked for this amount of time, after which the running processes are killed. There is no penalty for overestimating the run-time, although a conservative estimate will make your job easier (and thus quicker) to schedule. We generally recommend booking 50-100% longer runtimes than you expect to actually need.
- There are many more possible parameters to give to sbatch, e.g. to ask for more cores (-n) or for high-memory nodes (-C mem256GB) or for quick short jobs (-qos=short). Read the Slurm User Guide or the Bianca User Guide for more information.

5. The rest of the script are ordinary bash. Change the job script to do some (pretend) work, e.g. by making it do the work in a previous exercise. Keep the **sleep** command so that the job actually runs for some time.
6. Submit the job with “sbatch myfirstjob.sh”. Make a note of the ID of the job.
7. Check to see the status of the submitted job with “jobinfo”.
8. When the running job tries to print output to the terminal, the text is written to a file named “slurm<jobid>.out”

interactive

Sometimes, you need more power than a single login node, but you can’t write the work as a script and need to control the system manually. Instead of submitting a job with a long **sleep** command, waiting for the job to start, ssh-ing into the job’s node, and working that way, you can just type something like this:

$ interactive -A sens2019503 -t 00:10:00

The **interactive** command submits a job with a high priority and then connects you directly to the node where the job starts. You can then run commands on that node and use the node’s local disk just like a batch job.

For more information, please read the Slurm User Guide.

multicore and multi-node jobs

Thus far, we’ve only directly shown you how to submit single-core jobs. Many programs, codes, and pipelines claim to be multithreaded. Some of them are telling the truth!

In order to benefit from a multithreaded program, you need to book a job with multiple cores.

1. Add the line “#SBATCH -n 4” to the job script, next to the other sbatch parameters.

Now the job will occupy 4 cores on the node. Each node on Bianca has 16 cores. If your program or script can use multiple processes or threads, the job may finish more quickly.

Additionally, a single-core job is allowed only 1/16 of the node’s memory, or about 7GB. Should your job need more that this, it will die. If this happens, you should book additional cores so that your job gets enough memory.

Because Bianca is a resource shared between many very impatient scientists, it is not at all nice to waste resources. Unfortunately, a common occurrence is for people to book multiple cores unnecessarily. They will for example book 16 cores and most of the runtime use just 1 core, and very little memory. This

- wastes their core-hour allocation,
- makes their job slower to schedule, and
- prevents other jobs from using those resources.

UPPMAX provides a few tools to check the quality of your job bookings. The most popular one is **jobstats**.

1. Run “jobstats -p <jobid>”.

If the job has finished and ran for more than 5 minutes, then a PNG image will be created in the current directory.

2. View the image, e.g. by opening with the **eog** program.
3. Compare with real-life examples in **/proj/sens2019503/jobstatsplots**

What now?

Please read our online documentation and give us your opinions on it — we are about to embark on a major revision of our online material.

Ask us questions, share with us your concerns and ideas.