Getting Started on Wynton

Getting an account

1. **Submit a ticket to IT** to get a UID/GID assignment.
2. Once you have a UID/GID from IT, then **fill out the account request form**, checking the box for Gladstone and providing you UID/GID.
3. Wynton admins will then setup your account and work with you to make sure you can access the cluster.

Gladstone docs

- How to Move Files
- How to Submit Jobs
- Interactive Sessions on Wynton
- Running Analyses on Wynton

General docs

- Getting Started: https://ucsf-hpc.github.io/wynton/get-started/access-cluster.html
- Transferring Files: https://ucsf-hpc.github.io/wynton/transfers/files-and-directories.html
- Submittine Jobs: https://wynton.ucsf.edu/hpc/scheduler/submit-jobs.html
- FAQ: https://wynton.ucsf.edu/hpc/support/faq.html

Getting help

- Email wynton_admin@ucsf.edu for quick responses to Wynton questions.
- Join the Wynton mailing list to find solutions and ask questions: https://listsrv.ucsf.edu/cgi-bin/wa?A0=wynton-help
    - Need UCSF credentials to create an account, then respond to email confirmation.
    - Log in to subscribe and customize your preferences.
- Join the Wynton Slack channel to reach active community and ask questions: https://app.slack.com/client/T6ACBEFPE/learning-slack
    - Need to request to join, then respond to email confirmation.
    - Add to your Slack app or view in browser.
- Email bioinformatics@gladstone.ucsf.edu for questions about our hosted conainters, resources and workflows.
- Initiate an iLabs request to the Bioinformatics Core to get one-on-one or lab-wide consulting and training.

How to Move Files

FTP CLIENT

If you prefer a GUI tool, you can use any number of FTP client apps to transfer files from your machine to Wynton:

- Fetch
- CyberDuck
- FileZilla

The FTP connection hostname would be a Wynton transfer node: `dt1.wynton.ucsf.edu` or `dt2.wynton.ucsf.edu`.

SCP

By command line, you can transfer files from any machine to your Wynton home directory:

```
scp filename.tsv username@dt2.wynton.ucsf.edu:~/
```

See Wynton documentation for more examples.

SCRATCH SPACE

If you are copying files that will only be needed temporarily, for example as input to a job, then you have the option of copying them directly to a global scratch space at `/wynton/scratch/`. There is 492TiB of space available for this purpose. It is ***automatically cleared after 2 weeks***, but you should go ahead and delete the files when you no longer need them.

Note: it is good practice to first create your own subdirectory here and copy to that

```
mkdir /wynton/scracth/my_own_space
```

```
scp filename.tsv username@dt2.wynton.ucsf.edu:/wynton/scratch/my_own_space
```

See Wynton documentation for more about local and global scratch spaces.

UNDER CONSTRUCTION

- [ ] AD credential login
- [ ] Bionas and hub mounted

Accessing data transfer node

- Login to data transfer node with ssh

```
{localmachine}$ ssh username@gidt01.gladstone.org
```

- Start tmux with `tmux new -s <session name>`
- To detach: Use keys Ctrl+B, release, then press 'D'
- Reconnect to pervious tmux with `tmux attach`
- To kill the session: User Ctrl+D

```
{gidt01}$ tmux new -s first
[detached]
{gidt01}$ tmux attach
[exited]
```

Transfers Gladstone > Wynton

- Determine source and destination paths
- Run rsync

```
{gidt01}$ rsync -avrt /local/project/folder username@dt1.wynton.ucsf.edu:/gladstone/path/that/exists
```

Transfer Wynton > Gladstone

- Logged into Wynton or from data transfer node
- Determine source and destination paths
- Run rsync

```
{gidt01}$ cd /local/project/folder
{gidt01}$ rsync -avrt username@dt1.wynton.ucsf.edu:/gladstone/path/that/exists .
```

Details of /wynton/scratch and local node scratch

How to Submit Jobs

For any non-trivial computational tasks, you will want to submit them to the queueing system as *jobs*. The job scheduler on Wynton will automatically find and allocate compute nodes for your job. *Note: whether you submit them from a login node or a development node, jobs are always run on compute nodes.* There area a number of parameters that you should specify when submitting a job, including the **maximum memory** and **maximum runtime**. Sometimes it is necessary to specify the **current working directory** as well. Additionally, *Gladstone has a dedicated queue* that all Gladstone employees should automatically be associated with, called the ***member.q***, which will ensure that your jobs are run core that we have exclusive access to.

Example job submission

`qsub -l h_rt=00:01:00 -l mem_free=1G my_job.sge` (replace time, memory and file name)

- `-l h_rt` = maximum runtime (hh:mm:ss or seconds)
- `-l mem_free` = maximum memory (K|M|G)

Checking on a job

- Current status: `qstat` or `qstat -j 191442` (replace job id)
- After successful job: `grep "usage" my_job.sge.o2854740` (replace output file name)
- After failed job: `tail -100000 /opt/sge/wynton/common/accounting | qacct -f - -j 191442` (replace job id)

More about ...

- Job submission commands
- Types of nodes and their specifications
- Determining time and memory usage, and failed jobs

Additional tips...

- For intensive jobs during busy times, you can reserve resources for your job as soon as they become available by including this parameter `-R y`
- **Compute nodes** DO NOT have access to the internet, i.e., you can not run jobs that include steps like downloading files from online resources. **D ev nodes** DO have access to the internet. If your script or pipeline requires access to the internet, consider splitting up the work *(this is how nextflow works, by the way)*: **run a script on a dev node that retrieves online files and then submits jobs to be run on compute nodes.** *Viola.*
- Similiar to the previous point, you can run **cron jobs** on a dev node to periodically download files separate from compute-heavy jobs that can be submitted to compute nodes.

Resource Availability

To check Gladstone-specific resources...

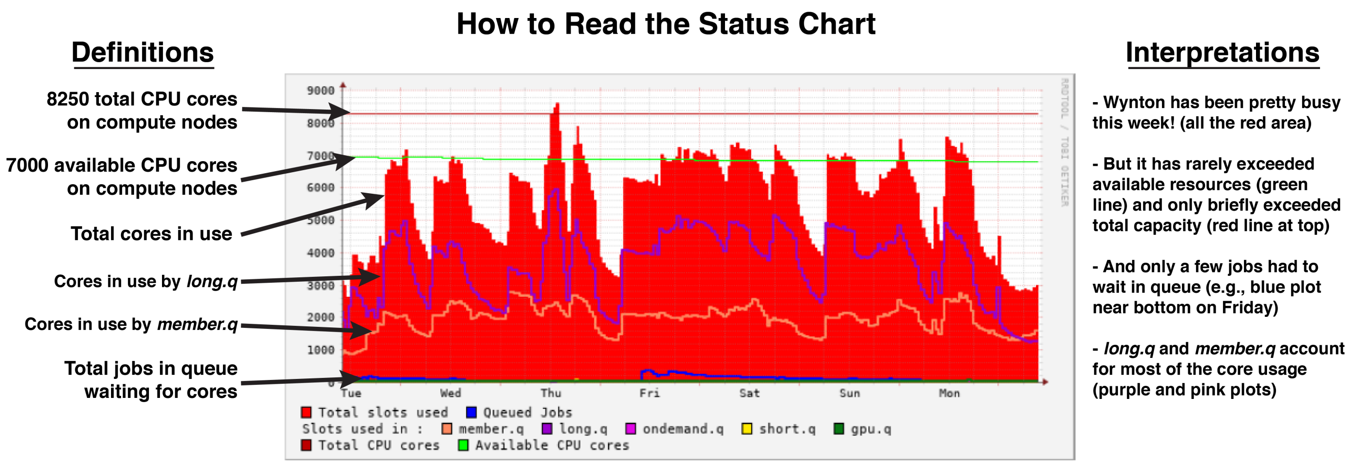
- Availability of cores for our ***member.q***

```
qquota -u "*" | grep gladstone
```

- Availability of disk space for our group members

```
beegfs-ctl --getquota --gid gladstone
```

Check the status of Wynton HPC and the availability of compute nodes: Wynton Status Charts



How to Read the Status Chart

Definitions

8250 total CPU cores on compute nodes

7000 available CPU cores on compute nodes

Total cores in use

Cores in use by *long.q*

Cores in use by *member.q*

Total jobs in queue waiting for cores

Interpretations

- Wynton has been pretty busy this week! (all the red area)

- But it has rarely exceeded available resources (green line) and only briefly exceeded total capacity (red line at top)

- And only a few jobs had to wait in queue (e.g., blue plot near bottom on Friday)

- *long.q* and *member.q* account for most of the core usage (purple and pink plots)

9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Tue
Wed
Thu
Fri
Sat
Sun
Mon
RRDTOOL / TOBI OETIKER
🟥 Total slots used 🟦 Queued Jobs
Slots used in : 🟧 member.q 🟪 long.q 🟪 ondemand.q 🟨 short.q 🟩 gpu.q
🟥 Total CPU cores 🟩 Available CPU cores

Scheduling Jobs on Gladstone Clusters

There are 3 job schedulers you might come across at Gladstone:

- **PBS/Torque** used on Finkbeiner cluster and Rigel
- **SGE/UGE** used on Wynton (until Q3 2020)   ***current option on Wynton***
- **Slurm** will be used on Wynton (starting Q3 2020)

Basic scheduler commands:

Action	PBS/Torque	SGE/UGE	Slurm
Job submission	qsub [script_file]	**qsub [script_file]**	sbatch [script_file]
Job deletion	qdel [job_id]	qdel [job_id]	scancel [job_id]
Job status by job	qstat -f [-j job_id]	**qstat -j job_id**	squeue [job_id]
Job status by user	qstat [-u user_name]	**qstat** [-u user_name]	squeue -u [user_name]
Job hold	qhold [job_id]	qhold [job_id]	scontrol hold [job_id]
Job release (from job on hold)	qrls [job_id]	qrls [job_id]	scontrol release [job_id]

Job specific commands:

Action	PBS/Torque	SGE/UGE	Slurm
Script directive		#$	#SBATCH
queue	-q [queue]	-q [queue]	-p [queue]
count of nodes	nodes=[count]*	N/A	-N [min[-max]]
CPU count	ppn=[count]*	-pe [PE] [count]	-n [count]
Wall clock limit	walltime=[hh:mm:ss]*	**-l h_rt=[seconds]**	-t [min] OR -t [days-hh:mm:ss]
Standard out file	-o [file_name]	-o [file_name]	-o [file_name]
Standard error file	-e [file_name]	-e [file_name]	e [file_name]
Combine STDOUT & STDERR files	-j oe	-j yes	(use -o without -e)
Copy environment	-V	-V	--export=[ALL | NONE | variables]
Event notification	-m abe	-m abe	--mail-type=[events]
send notification email	-M [address]	-M [address]	--mail-user=[address]
Job name	-N [name]	-N [name]	--job-name=[name]
Restart job	-r [yes|no]	-r [yes|no]	--requeue OR --no-requeue (NOTE: configurable default)
Set working directory	-d	-wd [directory] Alternatively set current working directory as directory: -cwd	--workdir=[dir_name]
Resource sharing	N/A	-l exclusive	--exclusive OR--shared
Memory size	--mem=[mem][M|G]	**-l mem_free=[memory][K|M|G]**	--mem=[mem][M|G|T] OR --mem-per-cpu=[mem][M|G|T]
Job dependancy	-W depend= [before|after:job_id]	-hold_jid [job_id | job_name]	--depend=[state:job_id]
Generic Resources	-l [resource]=[value]	-l [resource]=[value]	--gres=[resource_spec]
	* = requires -l and can be put in one string separated by commas		

Examples

I want to submit a script, "my_script.sh", to the queue "my_lab", requesting 4GB of memory and 10 cores. I expect my job to run for no more than 20 minutes, and I need 5GB of scratch space.

Rigel:

```
login_server $ qsub my_script.sh
login_server $ qsub -q my_lab -l mem=4GB,nodes=1,ppn=10 walltime=00:20:00 my_script.sh
```

Breaking that down:

qsub = submit jobs

-q my_lab = selects the specified queue: 'my_lab'

-l [resources] = specific resources requested, 4GB of memory, 1 node, 10 processors per node

walltime [hh:mm:ss] = expected run time

my_script.sh = your job

Wynton:

```
login_server $ qsub my_script.sh
login_server $ qsub -q my_lab -l mem_free=4g -l h_rt=00:20:00 -l scratch=5g -pe smp 10 my_script.sh
```

Breaking that down:

qsub = submit jobs

-q my_lab = selects the specified queue: 'my_lab'

-l [resources] = specific resources requested, 4G of memory available, 20 minute run time, 5G of /scratch available on node

-pe [Parallel Environment] [Core count] = requesting 10 cores

- Wynton has multiple parallel environments, read more about them here: https://ucsf-hpc.github.io/wynton/scheduler/submit-jobs.html#parallel-processing-on-a-single-machine

my_script.sh = your job

How to troubleshoot

- Check the **error log**: unless otherwise specified, this will be in your directory you launched your job from and will be formatted as job script name followed by .e<jobid>
- Check the **output log**: unless otherwise specified, this will be in your directory you launched your job from and will be formatted as job script name followed by .o<jobid>
- Send an email to wynton_admin@ucsf.edu for quick responses to Wynton questions.

Interactive Sessions on Wynton

It is currently *not* possible to request *interactive* jobs (aka `qlogin`). Instead, there are dedicated development nodes that can be used for *short-term interactive development needs* such building software and prototyping scripts before submitting them to the scheduler.

Interactive R session

1. ssh into login node
2. ssh to dev node
3. type `R` to enter session
4. type `q()` to quit session

Interactive python session

1. ssh into login node
2. ssh to dev node
3. type `python3` to enter session
4. type `exit()` to quit session

Running Analyses on Wynton

Here are protocols for common analyses previously run on Rigel. Some of these rely on popular tools and pipelines maintained externally. Others rely on containers maintained by the bioinformatics community at Gladstone. In each case, the protocols have been tested by Bioinformatics Core staff and are in regular use by the core and collaborators. There are many alternatives available. These are our the tools we are currently recommending and supporting.

Nextflow Pipelines

Bulk RNA-seq

We are recommending the popular nextflow pipeline for "RNA sequencing analysis pipeline using STAR, HISAT2 and Salmon with gene counts and quality control."

Useful links

- Nextflow RNA-seq main: https://nf-co.re/rnaseq
- Nextflow RNA-seq usage: https://nf-co.re/rnaseq/docs/usage#running-the-pipeline
- Nextflow configuration: https://www.nextflow.io/docs/latest/config.html
- Nextflow pipelines: https://nf-co.re/pipelines
- Solution to SGE using the wrong shell: https://github.com/nextflow-io/nextflow/issues/21

Installation and test run

Steps	Commands
1. Log into wynton	`ssh user@log2.wynton.ucsf.edu`
2. ssh into a dev node	`ssh dev2`
3. From your home directory, download nextflow	`curl -fsSL get.nextflow.io | bash`

4. Make a `bin` directory (if you haven't already) and move nextflow there	`mkdir bin`
	`mv nextflow ~/bin/`

5. Create a nextflow configuration file to specify SGE settings	`printf 'process.executor = "sge"\nprocess.penv = "smp"\nprocess.clusterOptions = "-S /bin/bash"' > .nextflow/config`
6. Run the nextflow test pipeline specifying the singularity profile. The console will display the progress in realtime. A warning message will appear during the first run regarding the automatic creation of a singularity cache directory.	`nextflow run nf-core/rnaseq -profile test,singularity`

7. The output be in the `results` directory. Pipeline reports are in `results/pipeline_info/`. *Note: if you get an error, try running it a second time.*	`ls results/`
	`ls results/pipeline_info`

Custom runs

Now you can setup and run the pipeline on your own data with step like the following:

1. Copy your fastq files over to wynton (see How to move data)
2. Specify `max_memory`, `genome`, `reads` and optional `skip*` arguments in the command (see docs on reads, genome and many others args that considered carefully)

```
   nextflow run nf-core/rnaseq --max_memory '8.GB' --skipBiotypeQC --skipFastQC --skipTrimming --genome GRCh38 --
reads '*_R{1,2}.fastq.gz' -profile singularity
```

Pro-tips:

- Review the `execution_report.html` to determine the necessary `max_memory` value for your analysis.

- You may want to use screen or tmux to manage longer runs.

Others

- Other nextflow pipelines: https://nf-co.re/pipelines

Bioinformatics Core Pipelines

Single Cell RNA-seq with Seurat

We maintain our own singularity container for the Seurat workflow at `/wynton/group/gladstone/biocore/containers/samples/scRNAseq/Seurat`. We provide a sample script and sample data for testing.

Useful links

- Seurat home page: https://satijalab.org/seurat/
- Seurat vignettes: https://satijalab.org/seurat/vignettes.html
- Seurat PCMB vignette: https://satijalab.org/seurat/v3.0/pbmc3k_tutorial.html

Installation and test run

Steps	Commands
1. Log into wynton	`ssh user@log2.wynton.ucsf.edu`

2. Copy `Seurat` folder to your home directory and cd into it	`cp -R /wynton/group/gladstone/biocore/containers/samples/scRNAseq/Seurat .`
	`cd Seurat`

3. Open `scripts/Load10XdataSet.R` and set the `user.dir` variable. *Pro-tip: To edit using vi, type* "**i**" *to enter edit mode, hit* **esc** *to exit edit mode, and type* "**:wq**" *to save and quit.*	`vi scripts/Load10XdataSet.R`
4. Open `scripts/JobSubmit.sh` and set the `userDir` variable	`vi scripts/JobSubmit.sh`
5. Submit job to the queue. You will see a message including your Job ID.	`qsub -cwd scripts/JobSubmit.sh`
6. The output will be a pdf of a violin plot	`ls`

Custom runs

Now you can adapt the script to run on your own data with steps like the following:

1. Copy your barcodes, genes and matrix files over to wynton into a new data subdirectory (see How to move data)
2. Edit a copy of the sample R script `Load10XdataSet.R`
    a. Point to your new data.dir
    b. Adjust args for `CreateSeuratObject` (see docs and links above)
    c. Adjust plot code (see visualization vignette)
3. Edit a copy of the sample submission script `JobSubmit.sh`
    a. Point to your custom R script
    b. Note: you can continue to use the same conainter or swap it out for your own

---