Logging in to beagle

1. Make sure you are connected to the pitt VPN through the pulse client.
2. Open a terminal.
3. Type `ssh [pitt username]@beagle.mmg.pitt.edu` to login to beagle.

Note that all terminal commands should be typed out in a single line. The commands below were spaced out on multiple lines to help with readability.

Some basic linux commands

Folders can renamed or moved to a different folderlater by using the `mv` command.

```
mv [original folder] [new folder name/path]
```

Files saved on Beagle/Cepacia can be moved onto a local computer using

```
scp -r [pitt username]@beagle.mmg.pitt.edu:[full path to folder to move]
[destination folder on local machine]
```

Tmux session

Tmux sessions allow a long-running command to continue even after you have logged out (or if the pulse client disconnects). Each session should be named so that they are easy to log back into.

To make a new tmux session:

```
tmux new -s [name]
```

To log back into a tmux session:

```
tmux attach -t [name]
```

To kill any tmux session that is no longer running any programs, run

```
tmux kill-session -t [name]
```

If you forgot what you named the tmux session, `tmux ls` will list all the running sessions. Multiple tmux sessions can be active at a time to run several programs at once.

Set up a project folder

All sequenced files are saved in the /home/data/dmux folder and are organized by date as YYMMDD. If you are looking for samples sequenced on March 16, they will be in the 190316 folder with the name 'CooperLab' followed by your initials. Each sample folder will have 4 files:

- two files with either 'I1' or 'I2' in their filename. These can be ignored.
- forward read: a file with 'R1' in its filename.
- reverse read: a file with 'R2' in its filename.

Since we don't want to change any of the files in the dmux folder, you should copy (use the command `cp`)the files from the dmux folder into a project folder in your home directory.

Example: If the samples were sequenced on April 7, they will be in the `/home/data/dmux/20190407/CooperLabABC` folder. To move them to a personal folder run (using the `-r` option for folders)

```
cp -r /home/data/dmux/20190407/CooperLabABC /home/ABC123/project1
```

where ABC123 is your personal folder.

Module system

Beagle uses a module system to sandbox programs into a specific environment. This helps to prevent bugs from interfering with analyses. The user has to explicitly load the programs/module they want to use during a tmux session using the command `module load [program name]`. The command `srun` is used to manage resources on Beagle which only means you should start any program commands with `srun`.

Trimming

There are two steps to finding mutations in samples: trimming and variant calling. The trimming step removes adapters used during sequencing that may be present in your sample and removes low-quality.

First, load the module:

```
module load trimmomatic
```

Then trim your sample reads using `trimmomatic`. This can take ~30 minutes to complete. The output files should be named with a `.fastq` extension and will be uncompressed.

```
srun trimmomatic PE -phred33
    [forward read] [reverse read]
    [forward paired output] [forward unpaired output]
    [reverse paired output] [reverse unpaired output]
    ILLUMINACLIP:/opt/trimmomatic/Trimmomatic-0.36/adapters/NexteraPE-
PE.fa:2:30:10
```

Some Useful Options:

- `ILLUMINACLIP:[fastafile]:[mismatches]:[palindrome clip threshold] [simple clip threshold]`: cut adapter and other illumina-specific sequences from each read. The illumina-specific sequences should be provided by a fasta file located at `fastafile`. The other three arguments specify how well a sequence in the sample must match the illumina-specific sequences to be considered a match.
- `LEADING:[quality]`: Cut low-quality bases off the start of a read if they fall below the `quality` value.
- `TRAILING:[quality]`: Cut low-quality bases off the end of a read if their quality is below `quality`.
- `MINLEN:[length]`: Only keep reads which have at least `length` bases.
- `SLIDINGWINDOW:[size]:[quality]`: Calculate the average base quality within a sliding window of length `size` and cut the read if the average quality falls below `quality`.

The above command allows you to specify what you want the resulting files to be called. To simplify the command omit the [output] file names and let `trimmomatic` automatically generate names for you. It's worth noting that recent samples (since ~ May 2019) are stripped of adapters during the demultiplexing step so trimming is not necessary anymore. It is still worthwhile to trim your reads to account for read quality.

Only the paired output files and forward unpaired sequences are used by other programs, so the reverse unpaired output files can be ignored unless your analysis uses them in some way.

Assembly

The assembly of each sample takes 3 steps:

1. Trim reads and adapters (described above)
2. Assemble using spades
3. Annotate with prokka

Assemble Genomes

The Assembly should be generated using the forward/reverse paired reads from `trimmomatic` as well as the forward unpaired reads, is available. Each assembly is constructed using `shovill`, which relies on `spades` to generate an initial assembly which is then polished using `pilon`.

```
shovill
    --minlen 500
    --assembler spades
    --outdir [output_folder]
    --R1 [forward trimmed reads]
    --R2 [reverse trimmed reads]
```

Options:

- `minlen`: Any contigs shorter than this threshold will be dropped from the final assembly.
- `assembler`: Shovill can use other assemblers to create the original assembly.

Annotate Genomes

Prokka can use the reference genome assembly files available from the ncbi to add the proper locus tags to the assembled genome. The downloaded genome from the ncbi website includes a `[ID]_translated_cds.faa` file with better annotations than the default databases that prokka uses. Each annotation looks something like this:

```
[gene=madL] [locus_tag=BCEN2424_RS06295] [protein=malonate transporter
subunit MadL][protein_id=WP_006485307.1] [location=1398707..1399102]
[gbkey=CDS]
```

Since we want to include the locus tags in the annotation, use the cds file mentioned above as a source of annotations for `prokka`.

```
prokka
    --genus [genus] --species [species]
    --proteins [[ID]_translated_cds.faa] # NOT cds_from_genomic.fna
    --outdir annotations
    --prefix [sample name]
    --rawproduct
    --usegenus
    [contigs.fna from shovill]
```

Option Descriptions:

- `genus`, `species`: Helps `prokka` choose the database to annotate from.
- `--proteins`: Either a fasta or genbank file with protien sequences to draw annotations from. It's recommended to use the `[ID]_translated_cds.faa` file since it has the most annotations. fasta file where each contig is a protein sequence with annotations as the contig name.
- `--addgenes`: "Add 'gene' features for each 'CDS' feature." This is used alongside the `--rawproduct` option to make sure the final annotated files contain the locus tags and other information, which would be removed under prokka's default options.
- `--rawproduct`: prokka removes extra information from annotations in order to standardize the annotation format. Since this also removes the locus tags that we want to keep, use this flag to disable the standardization step.

- `--usegenus`: If prokka cannot find an annotation from the cds file it falls back to generic databases and attempts to find a match there. This flag tells prokka to select a database related to the genus specified with the `--genus` option.

Variant Calling

Variant calling is done via `breseq`.

```
module load breseq
```

`Breseq` can take up to a couple hours to finish. The `-j` option reduces the runtime by allowingthe use of more threads/cores (typically 8).

```
srun --cpus [threads] breseq -j [threads] -r [reference file] -o [output folder] [trimmed forward paired file] [trimmed reverse paired file]
```

Note that breseq can read both compressed and uncompressed files, as long as the file extension is accurate. The output files are uncompressed, so should only be named with a `.fastq` extension. This process can take up to a couple hours to run, so if it is still running after, say, 3-4 hours, there is usually something wrong with the input files.

The index.html file can be converted to an excel file using the `breseq_parser.py` available at `https://github.com/cdeitrick/LabScripts`

```
python breseq_parser.py [path to index.html file]
```