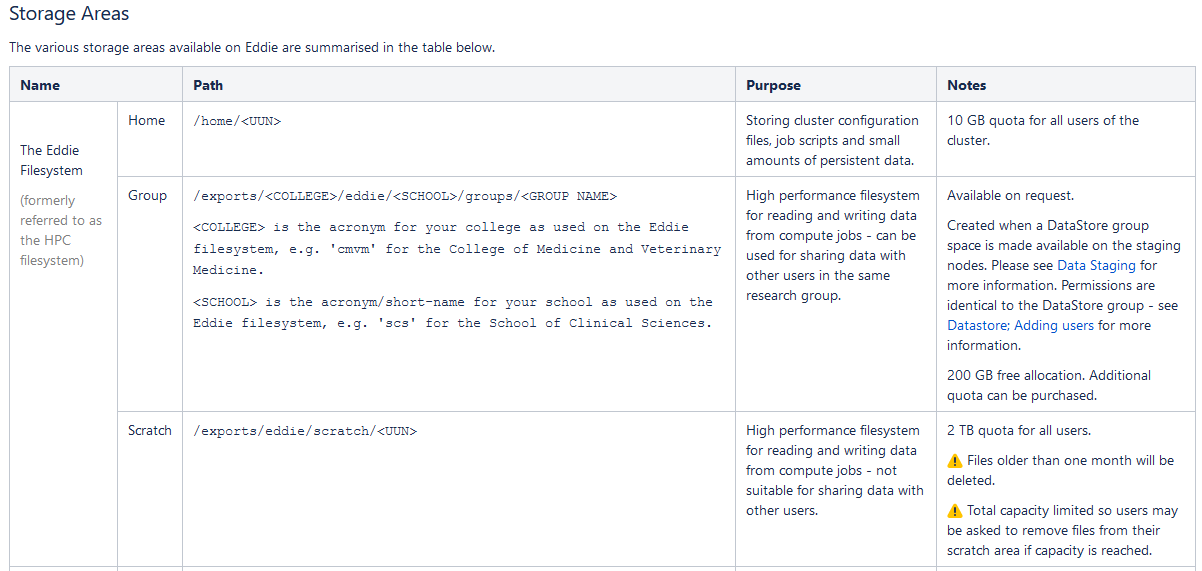
# Quick help notes

**Basic UNIX commands:**

|  |  |
| --- | --- |
| cat | Shows the contents of a file (like the text inside) |
| cd | Current directory (shows the directory you are in) |
| cd /path to directory you want to get to | Moves you to that specific directory |
| chmod u+x | Change file type to readable/executable file |
| chmod u+x insertnamehere.sh | Makes an already created file executable (green) |
| clear | Clear the screen |
| cp | Copy |
| cp ~/filename.sh . | This tells it to copy the filename.sh from the home directory (~) to the current directory (.) |
| dos2unix <<filename.sh>> | To make sure scripts are formatted for Unix |
| echo “text inside the file” > filetexteditorname.sh | Creates a file with the text “text inside the file” inside it. Then we need to change the file type. |
| echo > inserttextfileeditorname.sh | Make and empty MobaTextEditor file with a certain name and .sh file extension |
| grep “insert text” filename | Finds a string of text inside a file (aka search) |
| gunzip and bunzip2 <<folder/file.gz>> | To uncompress archive |
| gzip or bzip2 <<folder/file>> | To compress an archive. Creates .gz archive |
| id -Gn --zero | xargs -0 -n1 | List all the groups that the user is part of |
| less inserteddiejobnameerrorfile (example= less stagein.e24428506) | Check the error file of a job-if empty it worked (stagein=job name; e=error file; 24428506=job number) |
| less -lh |  |
| less -S | Check contents of a file |
| ls | Listing (list the contents of the current directory) |
| ls -l | List the permissions for the file. Who has access to the files and if they have read, write permissions or both. |
| ls -lh | List the current contents, in order and also lists the size |
| man | Manual of commands |
| mkdir | Make directory (make new folder) |
| mv (example mv  Shrimp\_stagein2.sh ~/.) | Move (here we move the text editor file named “Shrimp\_stagein2.sh”. You need to write ‘mv@ then the path of the folder/file then add a space then add the path to the directory where you wanna move the file |
| nano insertnamehere.sh | Open the created file to work in it |
| Press ‘TAB’ | Autofill code |
| Press Arrow ‘up’, ‘down’, ‘left’, ‘right’ on the keyboard | Use to move around the code and also if you press up or down arrow it will cycle through the lines of code that you wrote |
| pwd | check the directory you are in -> shows the file path |
| q | After the ‘man’ command and an insanely long list of documentation, the only way to minimise the info and start typing again is to press ‘q’  ‘q’ also works to exit other stuff like infinite rows with ‘END’ and the end |
| qdel xxxxxxxx | Deletes/removes job no xxxxxxxx |
| qlogin -q staging | Log in to staging node |
| qstat | Check the status of the job in the queue |
| qstat -j xxxxxxxx | Gives job error description on job no xxxxxxxx |
| qsub inserttexteditorfilenamehere.sh | Submit a job to eddie that will run from a .sh text editor file |
| quota | Gives disk usage quota on persona; home directory (/home/s2117972), on personal scratch on eddie (/exports/eddie/scratch/s2117972) and on the datastore on the group you have access to (/exports/cmvm/eddie/eb/groups/drobledo\_grp..or bean\_grp) |
| quota | Shows the directories you have access to and how much available space there is in all of them |
| readlink –f filename | Shows the entire path of a file |
| rm | Remove (deletes a file) |
| rm –r | Remove a directory with files (incl the files inside) |
| rmdir | Remove empty directory (deletes an empty ls -l directory) |
| Rsync <<source directory path>> <<destination directory path>> | Synchronizing files and directories between two locations. Like copy, but it will only copy the files or directories that are new or changed, saveing you some time. |
| tar | Create an uncompressed archive |
| tree | To see the folder architecture of the current directory |
| unzip | To uncompress .zip archive |
| zip | To compress a file/folder into .zip archive |



**File types:**

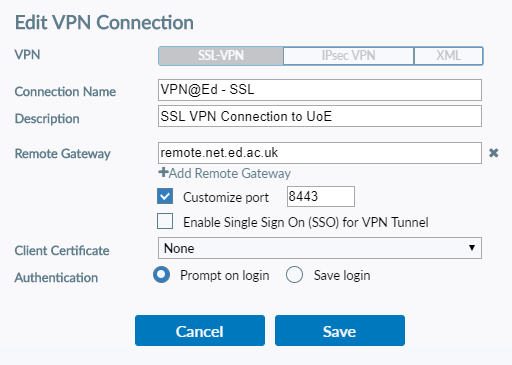
* **Blue**: Directory
* **Green**: Executable or recognized data file
* **Cyan** (Sky Blue): Symbolic link file
* **Yellow** with black background: Device
* **Magenta** (Pink): Graphic image file
* **Red**: Archive file
* **Red** with black background: Broken link

# Pre-Parse preparations

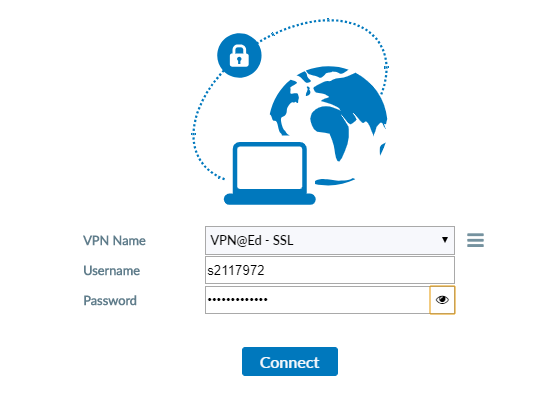
## Install VPN+MobaXterm

Install the FortiClient VPN so you can remotely use Eddie: <https://www.ed.ac.uk/information-services/computing/desktop-personal/vpn/forticlient-vpn>

Enter the following settings:



Click save and enter credentials:

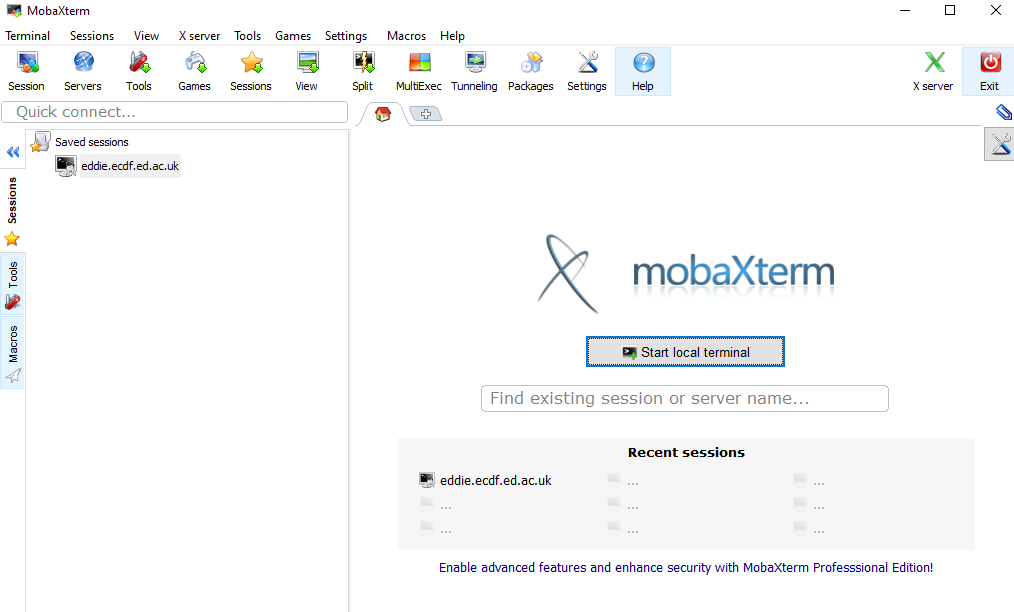


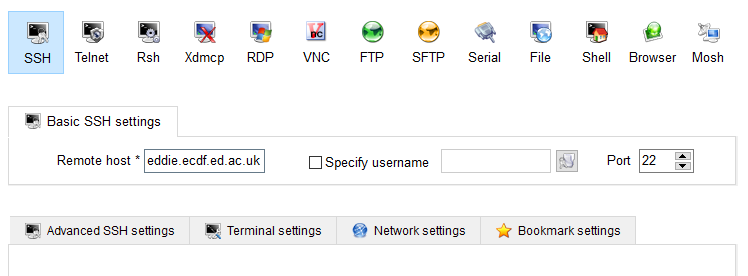
Type Username (I will use my old UNN as example through the document) & password

Click connect

## 2. Download and install MobaXterm app from UOE Software Centre

MobaXterm -> Session -> SSH-> Remote host = eddie.ecdf.ed.ac.uk -> OK -> login as: s2117972 -> you get to Eddie





**OR you can get to Eddie by starting the MobaXterm and logging in to Eddie as follows:**

$ ssh s2117972@eddie.ecdf.ed.ac.uk

#Eddie will then ask for the password. Enter your UOE pw: Note, the pw will be invisible, just type and enter

#Check current working directory (which should be home directory at first)

$ pwd

#Home directory will be /home/s2117972

#NOTE 1: Files on the scratch folder get deleted every 30 days, but can still recover data up to 60 days after deletion. Files on the eddie/groups/ do not get deleted after 30 days, but can still lose data if Eddie crashes. So always move output files and data to the datastore (it is backed-up by the university)

#NOTE 2: Click on “Fallow terminal folder” in eddie to see the current folder you are in and its contents. Can drag-and-drop files in the folders here if we wish to. Click “refresh” if things do not appear.

## 3. Install Parse App on EDDIE

Documentation for Parse App is on the website: <https://support.parsebiosciences.com/hc/en-us/articles/23060102930580-Pipeline-Installation-Current-Version> (Needs verified login – username & password so make an account)

GitHub for Pooran’s analysis as reference: <https://github.com/Pooran-Dewari/parse_single_cell_analysis/tree/main>

### 3.1 Create a conda environment for installation

Miniconda (the minimal version of "Anaconda") is a software distribution that contains a package manager (referred to as "conda") which allows one to install the software in their home directory without administrative (i.e. sudo) privileges. This step is also necessary to ensure that the appropriate versions of pipeline dependencies are installed and do not conflict with the user's current software environment.

Note 1: Anaconda documentation on Eddie wiki: <https://www.wiki.ed.ac.uk/display/ResearchServices/Anaconda>

Note 2: First choose a location to store your conda environments. DO NOT make it on the home directory (/home/s2117972) as it will fill up over the 10GB quota. This needs to go to the group space: (/exports/cmvm/eddie/eb/groups/..)

#requesting a node to work on

$ qlogin -l h\_vmem=20G

#make an a folder for Anaconda on the Eddie group space

$ cd /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/

$ mkdir anaconda

#now go to the anaconda folder and create 2 folders for the environment (envs) and packages (pkgs)

$ cd /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/anaconda

$ mkdir envs

$ mkdir pkgs

# Now load the anaconda module

$ module load anaconda/2024.02

#create a new conda environment 'spipe', and install all required dependencies in it

$ conda create -n spipe conda-forge::python==3.10

#the command above will take a while. Press yes ($ y) when prompted to install the packages.

### 3.2 create directory structure, download parse pipeline, and install

# Activate the conda environment

$ conda activate spipe

#Go to my scratch space

$ cd /exports/eddie/scratch/s2117972

#create directory structure for download. We will put the Parse stuff here

$ mkdir Parse\_app

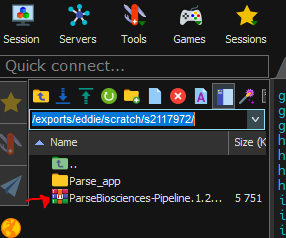
#Go within this new directory and create 3 folders for the pipeline outputs (/analysis), experimental data like fastq files (/expdata) and reference genome files

$ cd /exports/eddie/scratch/s2117972/Parse\_app/

$ mkdir analysis expdata genomes

Download the latest Parse pipeline in zip format: <https://support.parsebiosciences.com/hc/en-us/articles/17200056667924-Pipeline-Download-Current-Version>

#Now go to the scratch/s2117972/ directory on the top right (where the terminal is supposed to follow your directory)and drag and drop the parse pipeline zip in there



#Unzip the files we just moved

$ unzip ParseBiosciences-Pipeline.1.2.1.zip

#Go to the Pipeline folder and install the pipeline

$ cd ParseBiosciences-Pipeline.1.2.1/

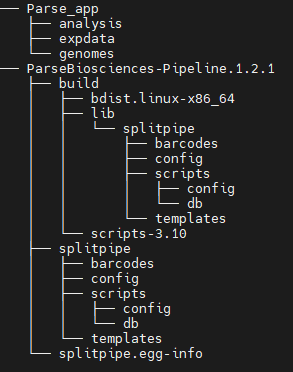
$ bash ./install\_dependencies\_conda.sh -i -y

$ “

### 3.3 Check directory structure after the install

$ tree -d

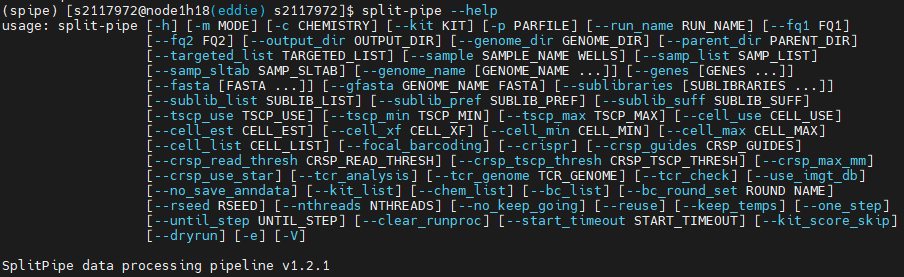
#The output in your scratch directory should look like this:



### 3.4 Check if installed correctly

$ split-pipe --help

#If installed correctly you get the below output:



## 4. Deal with Split Fastq samples (need to merge the files)

If you have multi-species reads (oyster and shrimp data) in the same lane, there are two ways to go forward with the analysis:

* **split the data first for each spcecies, and then analyse them separately**, or
* align the data first against a mixed-speices reference genome and split data later

I am going ahead with the **option 1** above, this will make sure that we do not lose reads that would have otherwise aligned to multiple loci with a mixed-species reference genome.

Need to move the sequencing data from Novogene from the datastore where we saved it to Eddie scratch.

### 3.2 Run the stagein script to move the library files in our scratch directory

#More info on stagein script here: <https://www.wiki.ed.ac.uk/display/ResearchServices/Data+Staging>

<https://www.wiki.ed.ac.uk/display/ResearchServices/Example+Staging+Scripts>

#### 3.2.1 Create a moba text editor file

#NOTE: I will make all the .sh executable files containing pieces of script in the /exports/eddie3\_homes\_local/s2117972"

**#**First select the directory where we are making .sh executable files

$ cd /exports/eddie3\_homes\_local/s2117972

#Then we create an empty file by right-click on terminal->New empty file-> -> name the file (incl the .sh)

#Now we have a ParseData\_stagein.sh file. But we need to make it an executable file

$ chmod u+x ParseData\_stagein.sh

#Now if we list the files in the directory, the Shrimp\_stagein.sh should be green

#Open the file from the terminal and start writing the script for Eddie

#### 3.2.2 Write the script inside the file

**\\…**

#!/bin/bash

#name the job”stagein”

#$ -N stagein

#Choose the staging environment

#$ -cwd

#$ -q staging

#Hard runtime limit. Priority list is <30min is high priority, >30min and <47h 59min is medium priority, >48h is least priority.

#$ -l h\_rt=12:00:00

# Make the job resubmit itself if it runs out of time: rsync will start where it left off

#$ -r yes

#$ -notify

# Make eddie send an e-mail to you when the job is done or has an error

#$ -m beas

trap 'exit 99' sigusr1 sigusr2 sigterm

#Add the source and destination directories

#Note: these paths are only available on the staging nodes

#It should start with one of /exports/csce/datastore, /exports/chss/datastore, /exports/cmvm/datastore or /exports/igmm/datastore

SOURCE=/exports/cmvm/datastore/eb/groups/drobledo\_grp/Alex\_F/Parse\_Data/X204SC24021433/RawData

# Destination path on Eddie. It should be on the fast Eddie HPC file system, starting with one of: /exports/csce/eddie, /exports/chss/eddie, /exports/cmvm/eddie, /exports/igmm/eddie or /exports/eddie/scratch,

DESTINATION=/exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/

# Perform copy with rsync

# Note: do not use -p or -a (implies -p) as this can break file ACLs at the destination

rsync -rl ${SOURCE} ${DESTINATION}

**…//**

#### 3.2.3 Run the script

#Now we need to save the script (File->Save) and submit the job to Eddie

$ qsub ParseData\_stagein.sh

#Now our job is submitted to Eddie

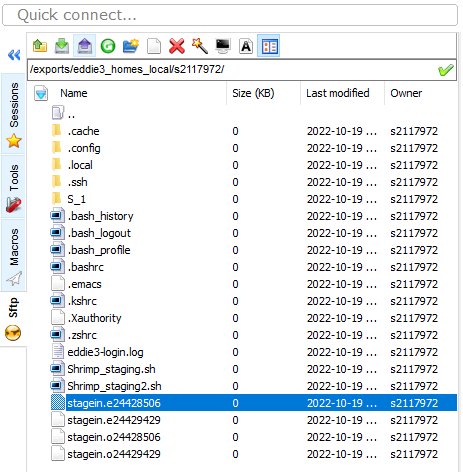
#We can check the state of the job with ‘qstat’. Eddie will give us a job name (eg: 24428422) and tell us if it is in the queue ‘qw’ or if it is running ‘r’ or “blank” if the job is ready.

$ qstat

#### 3.2.4. Repeat with all 8 sub-libraries until everything is moved over (~15-20min)

#Upon completion of a job. Eddie will make 2 files type “o2 and type “e”. They are going to be nameofeddiejob.ejobcodenumber (ex:stagein.e2442850). The “e” files are error files.

#Now we need to check if the error files created are empty:



$ less -S stagein.e24428506

#We want to make sure the file is empty. It should give an output like this



#now press ‘q’ on the keyboard to exit the file contents

$ q

#Now we want to make sure that the files we copied are in the directory they are supposed to. In our case, each of the 8 sub-libraries (A1 -> A8) into their respective folders

$ ls /exports/eddie/scratch/s2117972/rawdata

#Now make sure you remove the .e and .o files so we don’t get confused.

$ rm stagein.o2442850

$ rm stagein.e2442850

#### 3.2.4 Check the files with MD5 sum check

I did not have permissions to read the file. But Pooran made a .sh script file for it. See below as reference:

//….

RES="results\_md5sum.txt"

touch $RES

ls -d \*/ | while read DIR;

do

cd $DIR

echo "working on $DIR" >> ../$RES 2>&1

md5sum -c MD5.txt >> ../$RES 2>&1

echo "done $DIR !" >> ../$RES 2>&1

cd ..

done

…//

You need both the MD5.txt and the checkSize.xls for this to run. Otherwise you cannot do it.

As an alternative, this is the code I used for the checks for the 10X data as reference:

//….

#!/bin/bash

#What we tell the program to do is take the second (-f2) column of the checkSize.xls file, then for each value f in that column (the column has the file paths) calculate the size of the file (wc -c), then export to a file called "sizelist" in the tmp directory

path=/exports/eddie/scratch/s2117972/analysis/data/shrimp1/S\_1

for f in `cut -f2 checkSize.xls`; do wc -c ${f}; done > sizelist

awk ' ##Starting awk program from here.

FNR==NR{ ##Checking FNR==NR condition which will be TRUE when file1.txt is being read.

arr[$1,$2] ##Creating array with 1st and 2nd field here.

next ##next will skip all further statements from here.

}

(($1,$2) in arr){ ##Checking condition if 1st and 2nd field of file2.txt is present in arr then do following.

print $0, "OK" ##Print the current line here - $0, then print "OK"

arr1[$1,$2] ##Creating array arr1 with index of 1st and 2nd fields here, these are for the second file = sizelist

}

END{ ##Starting END block of this program from here.

for(i in arr){ ##Traversing through arr all elements from here (checkSize.xls)

if(!(i in arr1)){ ##Checking if an element/key is NOT present in arr1 (sizelist) then do following.

print i, "Error" ##Printing index and "Error" here.

}

}

}

' checkSize.xls sizelist ##Mentioning Input\_file names here.

…//

#### 3.2.5 Concatenate fastq files sequenced on different lanes (1-2min per cat file)

* Needs to be done for each sublibrary separately.
* It's a good idea to combine reads if a sample was sequenced on multiple lanes, and then do the splitting by groups/species.
* "If you have fastq files from multiple lanes they must be concatenated, but sublibraries should always remain separate"

# for sublibrary A1

$ cat A1\_EKDL240002473-1A\_223M7CLT4\_L7\_1.fq.gz A1\_EKDL240002473-1A\_223M7CLT4\_L8\_1.fq.gz > A1\_EKDL240002473-1A\_223M7CLT4\_cat\_1.fq.gz

$ cat A1\_EKDL240002473-1A\_223M7CLT4\_L7\_2.fq.gz A1\_EKDL240002473-1A\_223M7CLT4\_L8\_2.fq.gz > A1\_EKDL240002473-1A\_223M7CLT4\_cat\_2.fq.gz

#For sublibrary A2

$ cat A2\_EKDL240002474-1A\_223M7CLT4\_L7\_1.fq.gz A2\_EKDL240002474-1A\_223M7CLT4\_L8\_1.fq.gz > A2\_EKDL240002474-1A\_223M7CLT4\_cat\_1.fq.gz

$ cat A2\_EKDL240002474-1A\_223M7CLT4\_L7\_2.fq.gz A2\_EKDL240002474-1A\_223M7CLT4\_L8\_2.fq.gz > A2\_EKDL240002474-1A\_223M7CLT4\_cat\_2.fq.gz

#For sublibrary A3

$ cat A3\_EKDL240002475-1A\_223M7CLT4\_L7\_1.fq.gz A3\_EKDL240002475-1A\_223M7CLT4\_L8\_1.fq.gz > A3\_EKDL240002475-1A\_223M7CLT4\_cat\_1.fq.gz

$ cat A3\_EKDL240002475-1A\_223M7CLT4\_L7\_2.fq.gz A3\_EKDL240002475-1A\_223M7CLT4\_L8\_2.fq.gz > A3\_EKDL240002475-1A\_223M7CLT4\_cat\_2.fq.gz

#For sublibrary A4

$ cat A4\_EKDL240002476-1A\_223M7CLT4\_L7\_1.fq.gz A4\_EKDL240002476-1A\_223M7CLT4\_L8\_1.fq.gz > A4\_EKDL240002476-1A\_223M7CLT4\_cat\_1.fq.gz

$ cat A4\_EKDL240002476-1A\_223M7CLT4\_L7\_2.fq.gz A4\_EKDL240002476-1A\_223M7CLT4\_L8\_2.fq.gz > A4\_EKDL240002476-1A\_223M7CLT4\_cat\_2.fq.gz

#For sublibrary A5

$ cat A5\_EKDL240002477-1A\_223M7CLT4\_L7\_1.fq.gz A5\_EKDL240002477-1A\_223M7CLT4\_L8\_1.fq.gz > A5\_EKDL240002477-1A\_223M7CLT4\_cat\_1.fq.gz

$ cat A5\_EKDL240002477-1A\_223M7CLT4\_L7\_2.fq.gz A5\_EKDL240002477-1A\_223M7CLT4\_L8\_2.fq.gz > A5\_EKDL240002477-1A\_223M7CLT4\_cat\_2.fq.gz

#For sublibrary A2

$ cat A6\_EKDL240002478-1A\_223M7CLT4\_L7\_1.fq.gz A6\_EKDL240002478-1A\_223M7CLT4\_L8\_1.fq.gz > A6\_EKDL240002478-1A\_223M7CLT4\_cat\_1.fq.gz

$ cat A6\_EKDL240002478-1A\_223M7CLT4\_L7\_2.fq.gz A6\_EKDL240002478-1A\_223M7CLT4\_L8\_2.fq.gz > A6\_EKDL240002478-1A\_223M7CLT4\_cat\_2.fq.gz

#For sublibrary A7

$ cat A7\_EKDL240002479-1A\_223M7CLT4\_L7\_1.fq.gz A7\_EKDL240002479-1A\_223M7CLT4\_L8\_1.fq.gz > A7\_EKDL240002479-1A\_223M7CLT4\_cat\_1.fq.gz

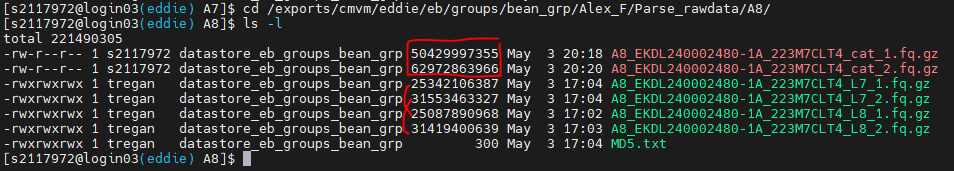
$ cat A7\_EKDL240002479-1A\_223M7CLT4\_L7\_2.fq.gz A7\_EKDL240002479-1A\_223M7CLT4\_L8\_2.fq.gz > A7\_EKDL240002479-1A\_223M7CLT4\_cat\_2.fq.gz

#For sublibrary A8

$ cat A8\_EKDL240002480-1A\_223M7CLT4\_L7\_1.fq.gz A8\_EKDL240002480-1A\_223M7CLT4\_L8\_1.fq.gz > A8\_EKDL240002480-1A\_223M7CLT4\_cat\_1.fq.gz

$ cat A8\_EKDL240002480-1A\_223M7CLT4\_L7\_2.fq.gz A8\_EKDL240002480-1A\_223M7CLT4\_L8\_2.fq.gz > A8\_EKDL240002480-1A\_223M7CLT4\_cat\_2.fq.gz

Check the \_cat files are in the folder and that they have the combined size of the 2 concatenated files:



#### 3.2.5 Split fastq files by species/groups (7-12h per sub-library)

Use the “fastq\_sep\_groups\_old” python script I downloaded.

Can also be found here: <https://www.dropbox.com/scl/fi/5d8zx5xjnumk2q2zvau0g/fastq_sep_groups.py?rlkey=wc58rop4yl5lmzudic3djagkw&e=1&dl=0>

**This is where the Python script is:** /exports/eddie/scratch/s2117972/Parse\_app/py\_scripts/fastq\_sep\_groups.py/

Note: Make sure the script is an executable (green) file (use chmod u+x if needed)

**This is where the sub-library fast q files are:**

/exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1 to A8

--group in the example below, shrimp samples are in three rows A1-A12, B1-B12, C1-C8  
--kit\_score\_skip I am using this option because the script didn't guess WT\_mega kit with a high score and wouldn't run!

screen -S split\_fastq

qlogin -l h\_vmem=20G #request 20GB of memory

cd /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/ #go to directory that has the A1 to A8 sub-libraries

SCRIPTPATH="/exports/eddie/scratch/s2117972/Parse\_app/py\_scripts/fastq\_sep\_groups.py" #script path -> the path that leads to the python script

FQ\_DIR=”/exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1/" #path to the directory containing fastq files

python $SCRIPTPATH \

--kit WT\_mega \ #Specify we used Parse WT\_mega kit

--kit\_score\_skip \

--fq1 ${FQ\_DIR}A1\_EKDL240002473-1A\_223M7CLT4\_cat\_1.fq.gz \ #Sublibrary A1 FW sequences (concatenated from 2 lanes)

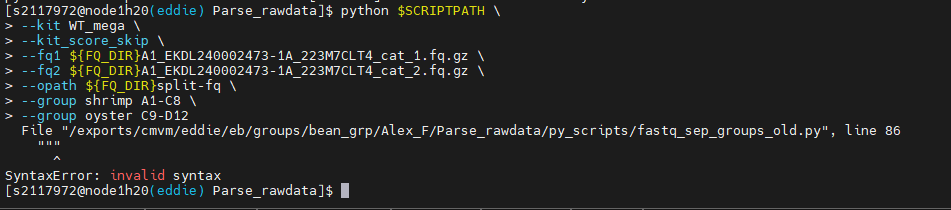
--fq2 ${FQ\_DIR}A1\_EKDL240002473-1A\_223M7CLT4\_cat\_2.fq.gz \ #Sublibrary A1 R sequences (concatenated from 2 lanes)

--opath ${FQ\_DIR}split-fq \

--group shrimp A1-C8 \ #Split shrimp samples from rows A1-A12, B1-B12, C1-C8…

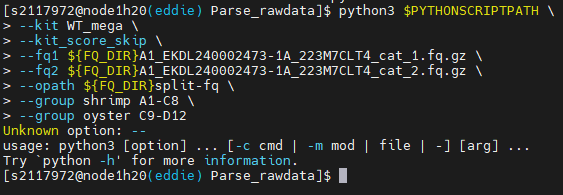
--group oyster C9-D12 #...from the oyster ones on rows C9-C12, D1-D12

This kept giving me the below error:



Then Pooran suggested I force it to run with python 3 just in case it had 2 by default:

So the code was with python3 $PYTHONSCRIPTPATH \ instead of python $SCRIPTPATH \. But then it gave me the below error:



I got in touch with Parse Bioscience (Ashwini Girish Kumar - [ashwini@parsebiosciences.com](mailto:ashwini@parsebiosciences.com)) and she sent me a new code to try alongside the new version of the python script. E-mail below:

*“Here is an updated version of the script where you can specify the exact chemistry and kit without any changes.*

*Here's an example of how to run the script. Make sure to update the file paths, sample information and fastq file names before you run the script:*

*python <path>/fastq\_sep\_groups.py \  
--chemistry v2 \  
--fq1 <path>/S1\_R1.fastq.gz \  
--opath <path>/split-fq \  
--group shrimp A1-C8 \  
--group oyster C9-D12*

*Looks like a problem with the quotes in your example, you can try removing them altogether. Or specify the full path (<path>) in the command instead of using the partial paths and variables.”*

New code:

screen -S split\_fastq

qlogin -l h\_vmem=20G

cd /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/

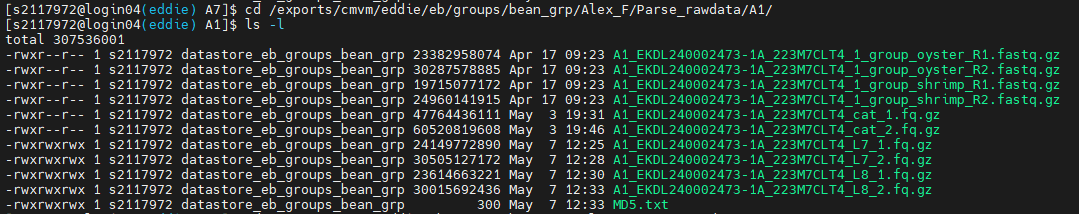
SCRIPTPATH=/exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/py\_scripts/fastq\_sep\_groups\_old.py

FQ\_DIR=/exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1/

python /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/py\_scripts/fastq\_sep\_groups\_old.py \  
--chemistry v2 \  
--fq1 /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1/A1\_EKDL240002473-1A\_223M7CLT4\_cat\_1.fq.gz \

--fq2 /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1/A1\_EKDL240002473-1A\_223M7CLT4\_cat\_2.fq.gz \  
--opath /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1/split-fq \  
--group shrimp A1-C8 \  
--group oyster C9-D12

Example from A1 below:



This has 2 files for shrimp and 2 for oyster. I will just need the shrimp data though.

Example output for A1:

number\_of\_reads 706205225

reads\_too\_short 0

reads\_valid\_bc 655887939

reads\_ambig\_bc1 348178

bc1\_Q30 0.749

bc2\_Q30 0.742

bc3\_Q30 0.951

bc\_edit\_dist\_NA 50317286

bc\_edit\_dist\_0 522694991

bc\_edit\_dist\_1 49123934

bc\_edit\_dist\_2 84069014

index\_with\_outs 633651030

index\_mult\_outs 26215441

index\_no\_outs 22236909

total\_outputs 659866471

New files: /exports/eddie/scratch/pdewari/newvolume/A1/split-fq/A1\_EKDL240002473-1A\_223M7CLT4\_cat\_1\_group\_shrimp\_R1.fastq.gz (and R2) 295197926

New files: /exports/eddie/scratch/pdewari/newvolume/A1/split-fq/A1\_EKDL240002473-1A\_223M7CLT4\_cat\_1\_group\_human\_R1.fastq.gz (and R2) 364668545

Total time 7:29:17.00

## 5. Prepare genome files (.fna & .gtf)

#### L. vannamei genome files

**2 possible genomes:**

I will work with the reference one (2019) due to mtRNA and rRNA presence in the annotation.

The **Mexican one from 2024** (no mitochondrial genome (mtRNA) or ribosomal (rRNA) in there).

I got the .fna (FASTA genome assembly) and the .gtf (genome annotation file) from the Mexican group by e-mail. Latest genome from 2024. The genome files can be downloaded from:

<https://www.ncbi.nlm.nih.gov/Traces/wgs/DAWKWD01?display=download>

Reference publication: [10.1093/jhered/esae015](https://doi.org/10.1093/jhered/esae015)

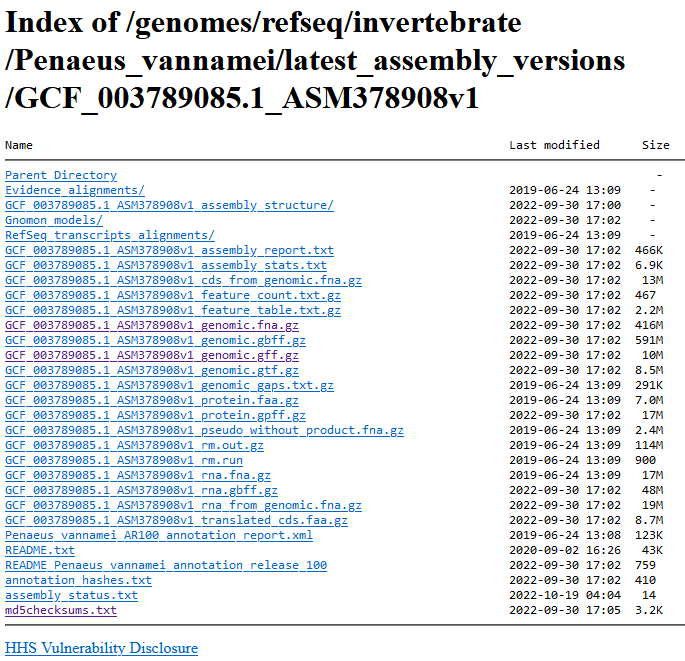
The **Reference genome from 2019** that I used before:

Can be downloaded from here: <https://www.ncbi.nlm.nih.gov/datasets/genome/GCA_003789085.1/>

Can download the gtf and the fasta files from the website or you can **download them from the FTP server here:** [**https://ftp.ncbi.nlm.nih.gov/genomes/refseq/invertebrate/Penaeus\_vannamei/latest\_assembly\_versions/GCF\_003789085.1\_ASM378908v1/**](https://ftp.ncbi.nlm.nih.gov/genomes/refseq/invertebrate/Penaeus_vannamei/latest_assembly_versions/GCF_003789085.1_ASM378908v1/) **(Get files from here please)**

Download:

* GCF\_003789085.1\_ASM378908v1\_genomic.gtf.gz
* GCF\_003789085.1\_ASM378908v1\_genomic.fna.gz
* md5checksums.txt (can't download, just open in browser and copy paste contents into a new md5checksums.txt file manually)



I copied everything on my desktop and then dragged and dropped inside the left side of Eddie terminal. I put the files here:

/exports/eddie/scratch/s2117972/Parse\_app/genomes/2019\_v2/

Then I asked for the md5sum numbers to be displayed on the terminal for each file:

$ cd /exports/eddie/scratch/s2117972/Parse\_app/genomes/2019\_v2/

$ ls

$ md5sum GCF\_003789085.1\_ASM378908v1\_genomic.fna.gz

$ md5sum GCF\_003789085.1\_ASM378908v1\_genomic.gtf.gz

OUTPUTS:

712ddcef5ff9795c4e273625b02a0b8f GCF\_003789085.1\_ASM378908v1\_genomic.fna.gz

6a3dbb015105c88a17080a96f3be3459 GCF\_003789085.1\_ASM378908v1\_genomic.gtf.gz

Check these numbers match with the numer inside the md5checksums.txt. They do so move on!

Unzip both files. They are .gz so use $ gunzip command

$ gunzip GCF\_003789085.1\_ASM378908v1\_genomic.gtf.gz

$ gunzip GCF\_003789085.1\_ASM378908v1\_genomic.fna.gz

Parse pipeline strictly requires a gtf gene annotation file, and would not work with gff3 format. Fortunately our file is already .gtf so no need to do anything else.

BUT, IF the file was .gff3 we would have had to:

1. Get AGAT (best tool for this) <https://agat.readthedocs.io/en/latest/gff_to_gtf.html>
2. Install agat in the same spipe conda environment as created previously.

# requesting node to run conda

qlogin -l h\_vmem=20G

module load anaconda/2024.02

conda activate spipe

conda install -c bioconda agat

1. Download gff3 file
2. Convert to gtf using agat

agat\_convert\_sp\_gff2gtf.pl --gff Crassostrea\_gigas.cgigas\_uk\_roslin\_v1.58.chr.gff3 -o Crassostrea\_gigas\_uk\_roslin\_v1.gtf

**Before using the gtf file further, must check that it complies with the** [**requirements**](https://support.parsebiosciences.com/hc/en-us/articles/11606689895828-GTF-Formatting-Guidelines)**, login required to access the page!**

<https://support.parsebiosciences.com/hc/en-us/articles/11606689895828-GTF-Formatting-Guidelines>

### Formatting

A GTF file is a tab-delimited file that consists of 9 separate columns. The contents of each column are described below; Definitions were taken directly from the [Ensemble website.](https://uswest.ensembl.org/info/website/upload/gff.html)

1. Seqname - name of the chromosome or scaffold; chromosome names can be given with or without the 'chr' prefix. Important note: the seqname must be one used within Ensembl, i.e. a standard chromosome name or an Ensembl identifier such as a scaffold ID, without any additional content such as species or assembly. See the example GFF output below.
2. Source - name of the program that generated this feature, or the data source (database or project name)
3. Feature - feature type name, e.g. Gene, Variation, Similarity
4. Start - Start position\* of the feature, with sequence numbering starting at 1.
5. End - End position\* of the feature, with sequence numbering starting at 1.
6. Score - A floating point value.
7. Strand - defined as + (forward) or - (reverse).
8. Frame - One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on..
9. Attribute - A semicolon-separated list of tag-value pairs, providing additional information about each feature.

The Parse pipeline mkref command requires the standard 9-column GTF file. In addition, there are minimal requirements for data elements in columns 3 and 9. For each annotated "feature" (gene or non-coding region), there must be the following:

* Column 3 - a "**gene**" and "**exon**" label. This means that you will have at least **two entries** for every feature (see the example below).
* Column 9 - a "**gene\_id**" and "**gene\_biotype**" or "**gene\_type**". Note that "biotype" is typically used in Ensembl GTFs, while "type" is more commonly used in Gencode GTFs.

Here is an example of a single feature that has the minimal required inputs. Note that there is one line each with **"gene" and "exon"** as features in column 3. Both lines include "gene\_biotype" keys in the attributes in column 9.



How can you check to see if your GTF file meets these requirements?

* For column 3 features, run the command cut -f3 gene\_model.gtf | sort | uniq and you should see both "gene" and "exon" show up in the results.
* For column 9 attributes, simply run grep -c "gene\_type" gene\_model.gtf and grep -c "gene\_biotype" gene\_model.gtf. If a match for either entry shows up (i.e. the number reported is greater than zero), then you have the required input for this field.

#### WSSV genome files – this is a step added on the second round of the analysis.

**NOTE: I re-did the analysis a second time by merging the WSSV and LV genomes and annotation files to look for WSSV integration into LV cells. This step did not exist in the initial analysis I did. So what I will do, is put these WSSV files into a new folder together with a new copy of the LV genome files and re-do everything from here on. I will change all the paths and file names to match.**

Ok. So in addition to the analysis between infected and control in terms of gene expression, I walso want to look if the WSSV has integrated transposable elements or any parts of the genome inside the host cell genome. So I need to look for WSSV genome too and merge the files together.

WSSV genome from 2023: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5824010/>

Use this assembly: <https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_003972525.1/>

Just like WSSV, download .fna and .gtf files for the genome and annotation.

Now put the files inside a filder together with a new copy of LV genome and annotation using WinSCP.

The genomes for LV and WSSV are here:

/exports/eddie/scratch/s2117972/Parse\_app/genomes/LV\_WSSV\_comb/

Now I need to merge the genome files together and then the annotation files together. Use concatenate command:

$ cd /exports/eddie/scratch/s2117972/Parse\_app/genomes/LV\_WSSV\_comb/

$ cat LV\_annotation.gtf WSSV\_annotation.gtf > LV\_WSSV\_comb\_annotation.gtf

$ cat GCF\_003789085.1\_ASM378908v1\_LV\_genomic.fna GCF\_003972525.1\_ASM397252v1\_WSSV\_genomic.fna > LV\_WSSV\_comb\_genome.fna

Check if the files were made

$ ls

So from here I will continue as I did on my first round of analysis, with just changed names and paths:

#### make ref genome using spipe ( ~ 8min)

Make a Ref\_genome.sh script file using $ nano command and format with $ dos2unix. More explanations on these 2 commands in the Parse pipeline section.

$ cd /exports/eddie3\_homes\_local/s2117972/

$ nano LV\_WSSV\_ref\_genome.sh

-> copy the below code and paste it with right click insode the script file

-> Press ctr+x -> Press Y to save-> Press enter to exit.

//…

#$ -V -cwd

#$ -l h\_rt=12:00:00 ###HH:MM:SS #Hard runtime limit

#$ -l h\_vmem=24G #Memeory per core. That means 16cores x24Gb, 384GB memory

#$ -pe sharedmem 8 #request 16 threads (aka cores)

#$ -P roslin\_bean\_grp #run this on the superdome cluster. Need to specify the “project with -P followed by the personal cluster name, aka roslin\_bean\_grp

module load anaconda/2024.02

conda activate spipe

split-pipe \

--mode mkref \

--genome\_name L\_vannamei\_WSSV \

--fasta /exports/eddie/scratch/s2117972/Parse\_app/genomes/LV\_WSSV\_comb/LV\_WSSV\_comb\_genome.fna \

--genes /exports/eddie/scratch/s2117972/Parse\_app/genomes/LV\_WSSV\_comb/LV\_WSSV\_comb\_annotation.gtf \

--output\_dir /exports/eddie/scratch/s2117972/Parse\_app/genomes/LV\_WSSV\_ref\_genome

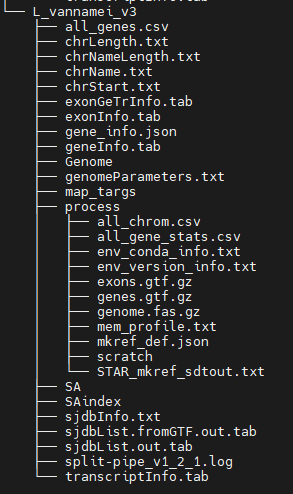
…//

$ dos2unix LV\_WSSV\_ref\_genome.sh

$ qsub LV\_WSSV\_ref\_genome.sh

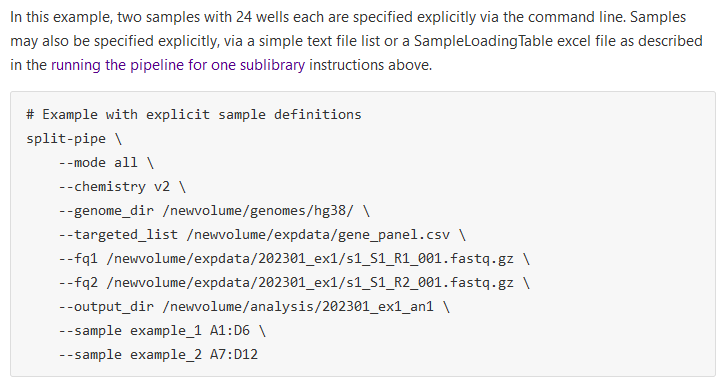
This will create the directories below. Have a look at the “split-pipe\_v1\_2\_1.log and check if there are any errors or problems at the end of the file. Things may have not worked properly even if they look like so.

Also check the “split-pipe\_v1\_2\_1.log” inside the ref\_genome folder to see if the task finished successfully without any problems or erros.



# Run Parse Pipeline (3-5h/sublibrary)

Example from Parse website:



### Memory and CPU considerations

As per Parse [guidelines](https://support.parsebiosciences.com/hc/en-us/articles/23060102930580-Pipeline-Installation-Current-Version), below is the recommended minimum memory & CPU requirements.

For processing a single sublibrary:

Memory:

* 100M reads or less = 64GB
* 100M-500M reads = 128GB
* 500M-1B reads = 256GB

CPU/threads:

* 100M reads or less = 8 threads
* 100M-500M reads = 16 threads
* 500M-1B reads = 24-32 threads

### Run the pipeline for sub-library A1

After fastq splitting, we have approx 400M paired-reads in each sublibrary; would need aprox 300GB memory & 18 threads

Files in: /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1 to A8

**SAMPLE LOADING:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |  |
| **A** | S1 | S1 | S1 | S2 | S2 | S2 | S3 | S3 | S3 | S4 | S4 | S4 |  |
| **B** | S5a | S5a | S5a | S5b | S5b | S5b | S7 | S7 | S7 | S8 | S8 | S8 |  |
| **C** | S9 | S9 | S10 | S10 | S11 | S11 | S12 | S12 | Oys | Oys | Oys | Oys |  |
| **D** | Oys | Oys | Oys | Oys | Oys | Oys | Oys | Oys | Oys | Oys | Oys | Oys |  |
| **E** | Bottom 48 wells are empty | | | | | | | | | | | | |
| **F** |
| **G** |
| **H** |

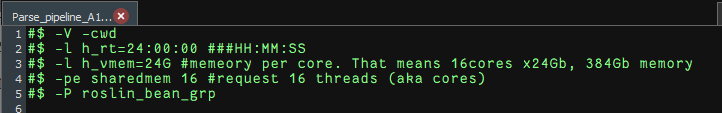
Oys=Ambre’s oyster samples (ignore these); S1 -> S12 =My shrimp samples

HP=S1, S2, S5a, S5b, S7, S8; LO=S3, S4, S9, S10, S11, S12

**NEW WAY OF SUBMITTING JOBS. ALSO, THESE WILL BE SUBMITTED ON THE SUPERDOME (aka Eddie’s priority cluster)!**

Ideally you want these submitted as priority because otherwise they will never run. First time I tried the normal way the job was stuck in queue 7 days without starting yet. And that was job 1 out of 8. So if you have time constraints, like I did, this is the way:

Ok- so in order ro run stuff on the superdome, you need to contact IT and request them to make you a log-in name and ID for this. Ours is “roslin\_bean\_grp” . You need to add the following command in the script file so it goes to those nodes: #$ -P roslin\_bean\_grp



OK. Now memeory wise, DO NOT request 18 cores as Pooran did in his example. Turns out that will lock me out of majority of Eddie’s core space. That is because the cores go 16, 32, 40 and 80.

Below e-mail from Andy Law

*“If you’d requested 16 cores, with 24GB per core, then your jobs would have started much quicker (and hence completed sooner). The reason for that is that we have many (many) times more nodes with 16 cores than we do with 32, 40 or 80 (the other sizes available). Obviously an 18-core job can’t run on a 16-core node so by asking for that, you are restricting yourself to a fraction of what Eddie has available and you have to wait for a slot to become available on that small fraction.*

*Having 18 cores might make the job run a small amount quicker once it’s started, but if you have to wait a week for the job to actually start then it will be much slower overall.”*

So we will go with 16 cores, 24Gb each.

**CREATE A FILE WITH PROPER UNIX FORMATTING:**

$ cd /exports/eddie3\_homes\_local/s2117972/

$ nano Parse\_pipeline\_A1.sh

This will open a new tab (not a text editor) which is still inside the terminal. So can’t be edited like a text file. Just paste (right-click) the code below in there:

Script file:

…/

#$ -V -cwd

#$ -l h\_rt=24:00:00 ###HH:MM:SS #Hard runtime limit

#$ -l h\_vmem=24G #Memeory per core. That means 16cores x24Gb, 384GB memory

#$ -pe sharedmem 16 #request 16 threads (aka cores)

#$ -P roslin\_bean\_grp #run this on the superdome cluster. Need to specify the “project with -P followed by the personal cluster name, aka roslin\_bean\_grp

module load anaconda/2024.02 #Load the Anaconda module

conda activate spipe #Activate the conda module (aka min version of Anaconda)

# Can see 360 million paired reads so anything > 128GB and 16 threads should do

# We have already split by groups/species, no point in defining wells for Ambre’s samples (oyster). Just do the shrimp ones.

#Mention our kit chemistry (v2) and the kit we used (WT\_mega)

#Add the parameter --kit\_score\_skip to ignore warning about low-score detection of the kit.

# don't need --parfile parfile.txt as we have already split the fastq files by species.

split-pipe \

--mode all \

--chemistry v2 \

--kit WT\_mega \

--kit\_score\_skip \

--genome\_dir /exports/eddie/scratch/s2117972/Parse\_app/genomes/LV\_WSSV\_ref\_genome/ \

--fq1 /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1/A1\_EKDL240002473-1A\_223M7CLT4\_1\_group\_shrimp\_R1.fastq.gz \

--fq2 /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A8/A8\_EKDL240002473-1A\_223M7CLT4\_1\_group\_shrimp\_R2.fastq.gz \

--output\_dir /exports/eddie/scratch/s2117972/Parse\_app/analysis/A1\_results\_v2 \

--sample S1\_HP A1:A3 \

--sample S2\_HP A4:A6 \

--sample S3\_LO A7:A9 \

--sample S4\_LO A10:A12 \

--sample S5a\_HP B1:B3 \

--sample S5b\_HP B4:B6 \

--sample S7\_HP B7:B9 \

--sample S8\_HP B10:B12 \

--sample S9\_LO C1:C2 \

--sample S10\_LO C3:C4 \

--sample S11\_LO C5:C6 \

--sample S12\_LO C7:C8

…/

Now press “ctr+x “ -> Press “Y” to save -> Press “Enter” to exit

**NO NEED For chmod u+x name.sh anymore!**

**BUT WE WANT TO FORMAT THE FILE FOR UNIX:**

$ dos2unix Parse\_pipeline\_A1.sh

$ qsub Parse\_pipeline\_A1.sh

Reference genome directory:

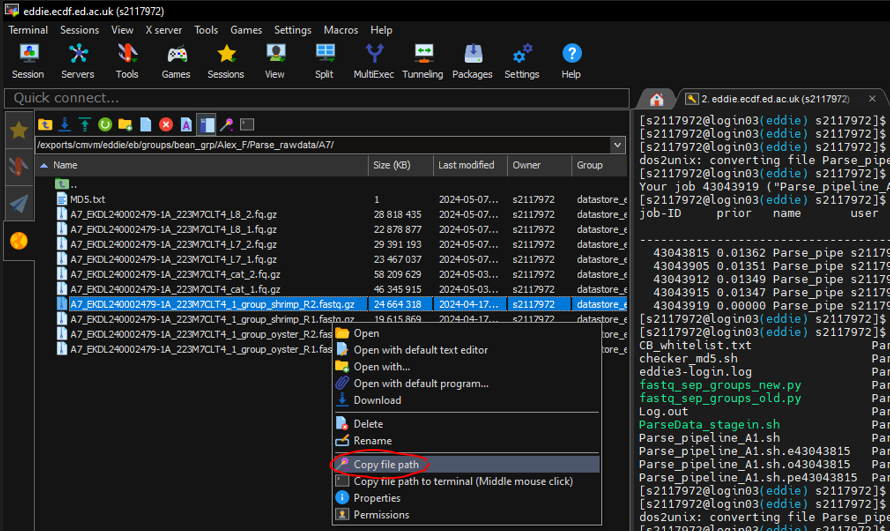
/exports/eddie/scratch/s2117972/Parse\_app/genomes/LV\_WSSV\_ref\_genome/

Shrimp library files:

/exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_rawdata/A1 /

**NOTE: WHOLE FILE HAS A DIFFERENT NAME< NOT JUST A1/A2..etc. Change the whole path !!!!**

Can get the path and file name easily by navigating to the file directory in the terminal window to the right, right-click on the file and “Copy file path”. Just careful to delete the quotation marks.



All parse split-pipe finished successfully. I checked all the logs. 0 problems. They all ran in 3-5hrs each.

### Combine all files from the results (~5min)

$ nano sublibs\_v2.lis -> paste the below paths inside -> ctr+x -> Y -> enter

//…

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A1\_results\_v2

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A2\_results\_v2

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A3\_results\_v3

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A4\_results\_v2

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A5\_results\_v2

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A6\_results\_v2

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A7\_results\_v2

/exports/eddie/scratch/s2117972/Parse\_app/analysis/A8\_results\_v2

…//

Now create the script file and let it run on the cluster.

$ nano Parse\_comb\_sublibs\_v2.sh -> paste the below code inside -> xtr+x -> Y -> enter

//…

#$ -V -cwd  
#$ -l h\_rt=02:00:00 ###HH:MM:SS  
#$ -l h\_vmem=24G  
#$ -pe sharedmem 8

module load anaconda/2024.02  
conda activate spipe

split-pipe \  
    --mode comb \  
    --sublib\_list /exports/eddie3\_homes\_local/s2117972/sublibs\_v2.lis \  
    --output\_dir /exports/eddie/scratch/s2117972/Parse\_app/analysis/combined\_shrimp\_WSSV

…//

$ dos2unix Parse\_comb\_sublibs\_v2.sh

$ qsub Parse\_comb\_sublibs\_v2.sh

### Move the end files from the combined\_shrimp directory onto datastore & bean eddie grp:

From: /exports/eddie/scratch/s2117972/Parse\_app/analysis/combined\_shrimp\_WSSV/

To-backup 1: /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_Eddie\_out/

To-backup 2: /exports/cmvm/datastore/eb/groups/bean\_grp/Alex\_F/shrimp\_WSSV\_parse/

First log into the staging nodes and then go to the destination locations and make the “shrimp\_WSSV\_parse\_v2” folders using $ mkdir command

$ qlogin -q staging

$ cd /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/

$ mkdir Parse\_Eddie\_out\_v2

$ cd /exports/cmvm/datastore/eb/groups/bean\_grp/Alex\_F/

$ mkdir Parse\_Eddie\_out\_v2

$ cp -r /exports/eddie/scratch/s2117972/Parse\_app/analysis/combined\_shrimp\_WSSV/ /exports/cmvm/eddie/eb/groups/bean\_grp/Alex\_F/Parse\_Eddie\_out\_v2/

$ cp -r /exports/eddie/scratch/s2117972/Parse\_app/analysis/combined\_shrimp\_WSSV/ /exports/cmvm/datastore/eb/groups/bean\_grp/Alex\_F/shrimp\_WSSV\_parse\_v2/