Genome Assembly Tutorial: Command Line (CLI)

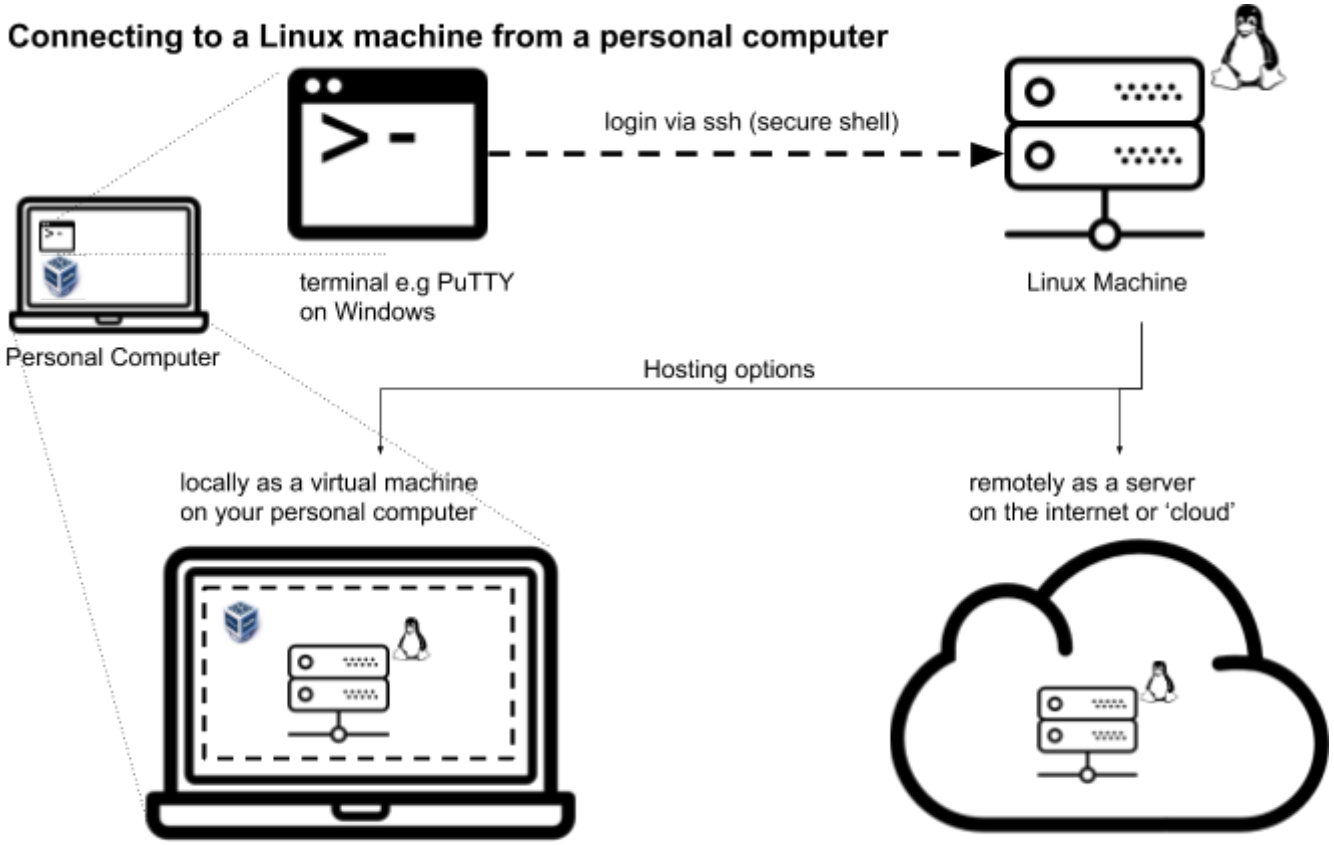
In the last tutorial you explored assembling a bacterial genome using the web-based Galaxy tool. In this tutorial you will perform the same operation but on the linux command line. As a prerequisite for this training you should have completed the command line training on this website (a free account will be sufficient) if you are unfamiliar with the Linux environment.
An video to accompany this tutorial can be found here

For the purposes of this tutorial we will login into a Linux machine for two reasons:

1. Most personal computers do not have sufficient power to run large scale analyses and so for bioinformatics analyses users often login to a machine that has better processing power than a personal computer
2. On a Windows machine it is not possible to access a Linux environment easily and so it is best to login to a machine that is natively Linux

To make this straightforward for the tutorial you will use a pre-built virtual machine (VM) image and use the software VirtualBox to run this virtual machine. This hosts a Linux machine within your local personal computer, whereas often the Linux machine would be hosted in the internet or 'on the cloud'

This setup is represented in the diagram below



The VM has all the software needed to perform the assembly pre-installed. In order to run the VM you will need a recent laptop/desktop with 8Gb RAM. To check if your personal computer is virtualisation ready please follow the steps in this web post.

Learning Objectives

In this tutorial you will learn how to

1. Set up a virtual machine
2. Connect to a Linux server using ssh via the PuTTY program
3. Transfer files to a Linux server using the WinSCP program
4. Run commands on a Linux server
5. Perform a genome assembly on the command line

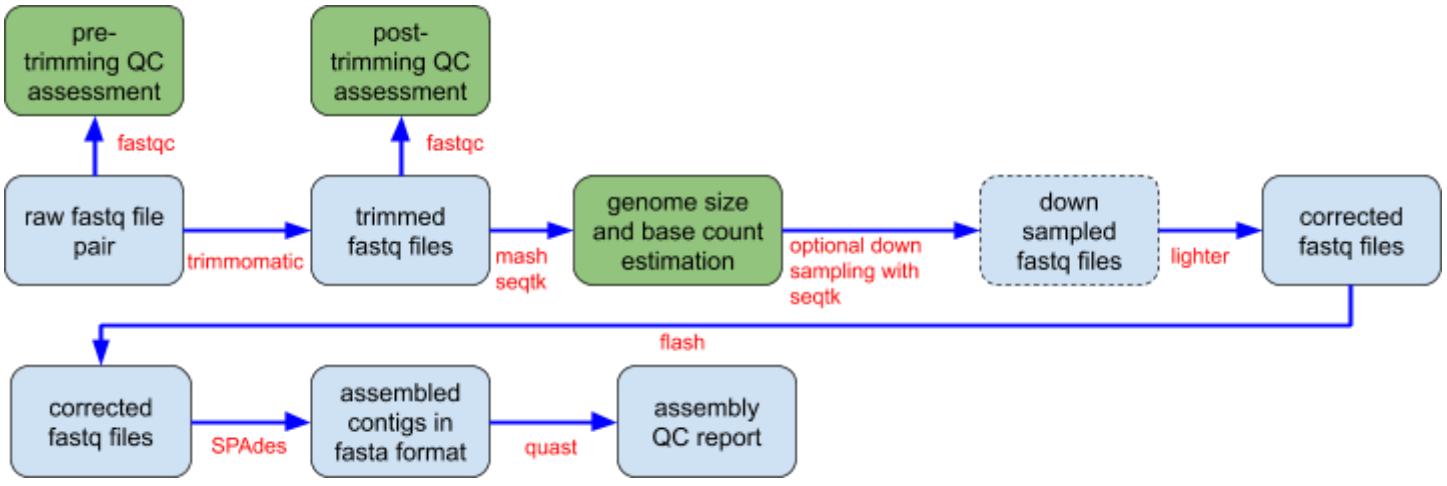
By the end of the tutorial you should be familiar with some of the software used to assemble genomes from the fastq files that come directly from a sequencing machine and how to run through the individual steps necessary to produce a genome assembly and assess it using basic quality metrics.

Tutorial

This tutorial will be structured as follows

- Setting up a Linux virtual machine
- Connecting to a Linux machine
- Transferring files to a Linux Machine
- Assembling genomes on the command line

A schematic the bioinformatics assembly process is as follows

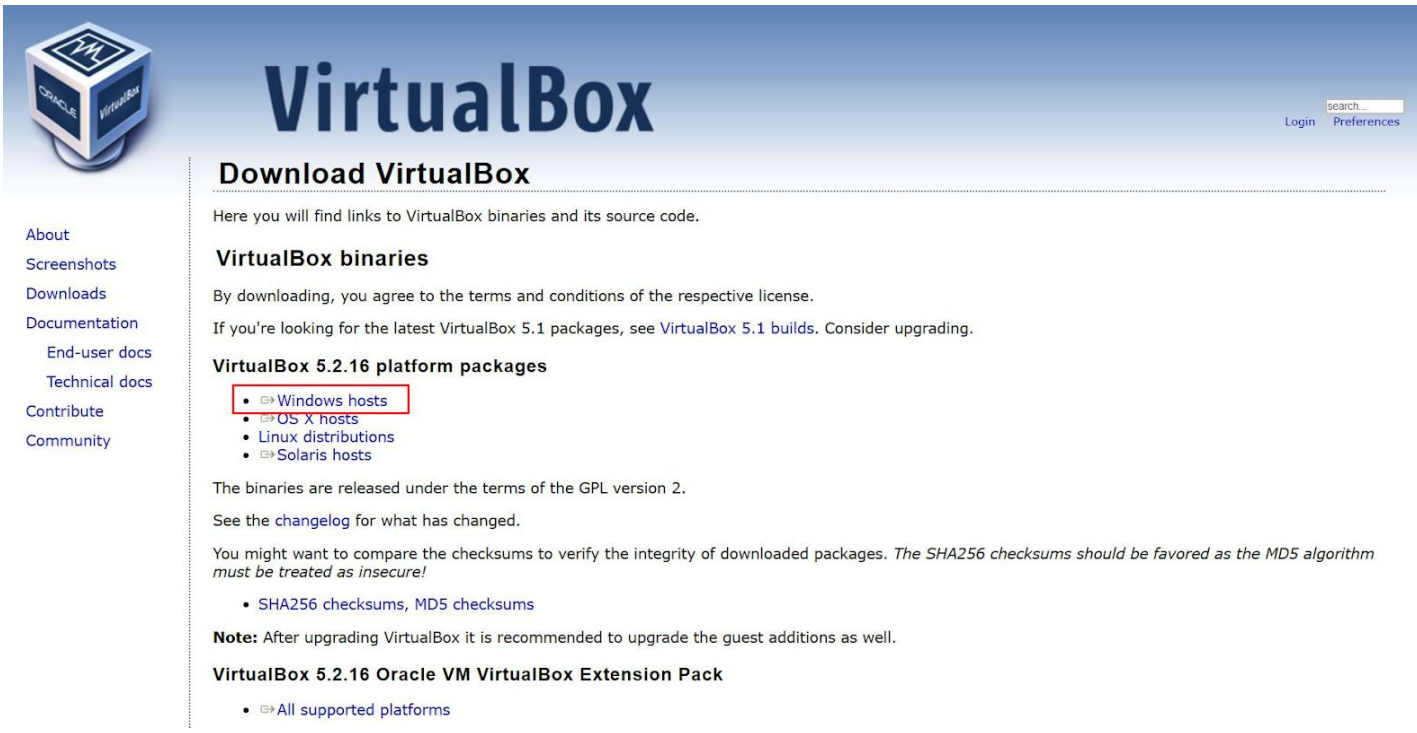


An video to accompany this tutorial can be found here

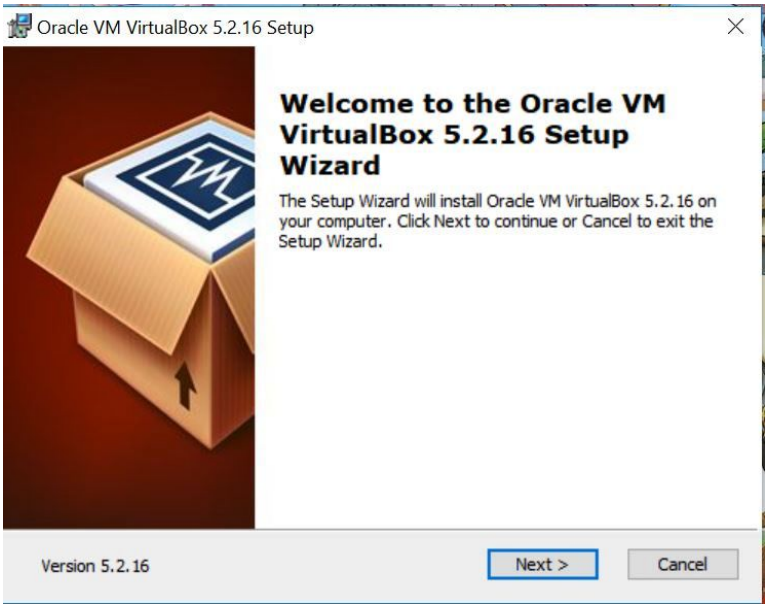
Setting up the virtual machine

1. First download the VM image (this is large 2.4Gb)

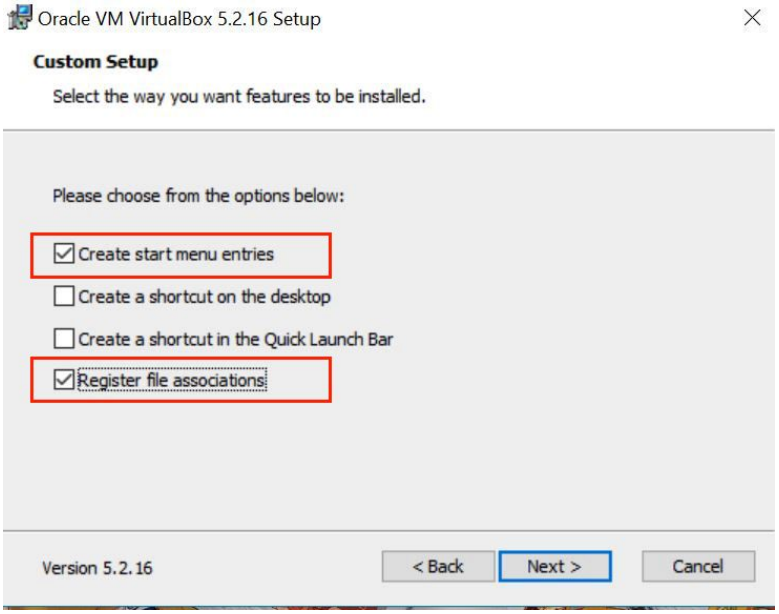
2. Download VirtualBox from https://www.virtualbox.org/wiki/Downloads
   Select Windows Hosts link to download the file.



3. Once downloaded, install by double clicking the exe file. Now walk through the steps

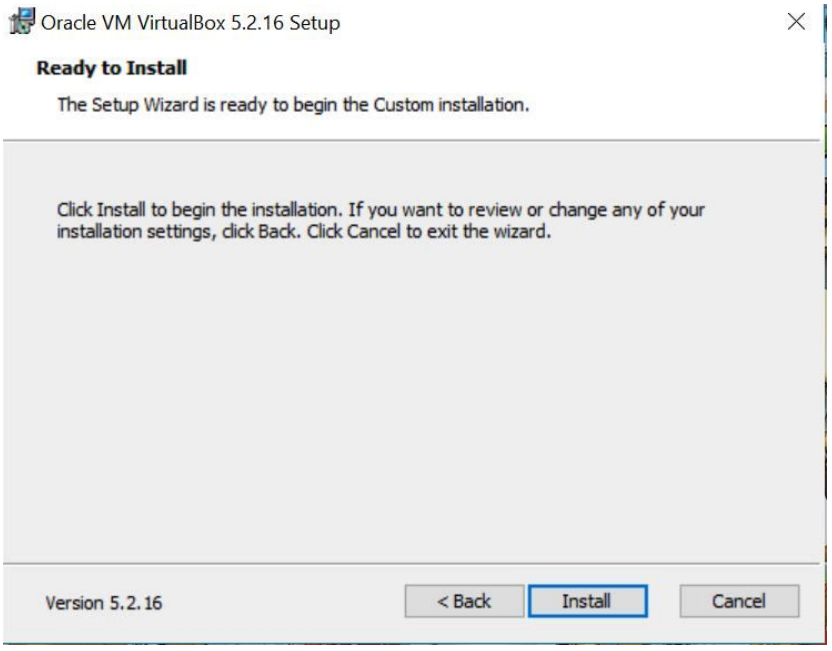


   At the custom setup stage make sure the 2 options below are selected

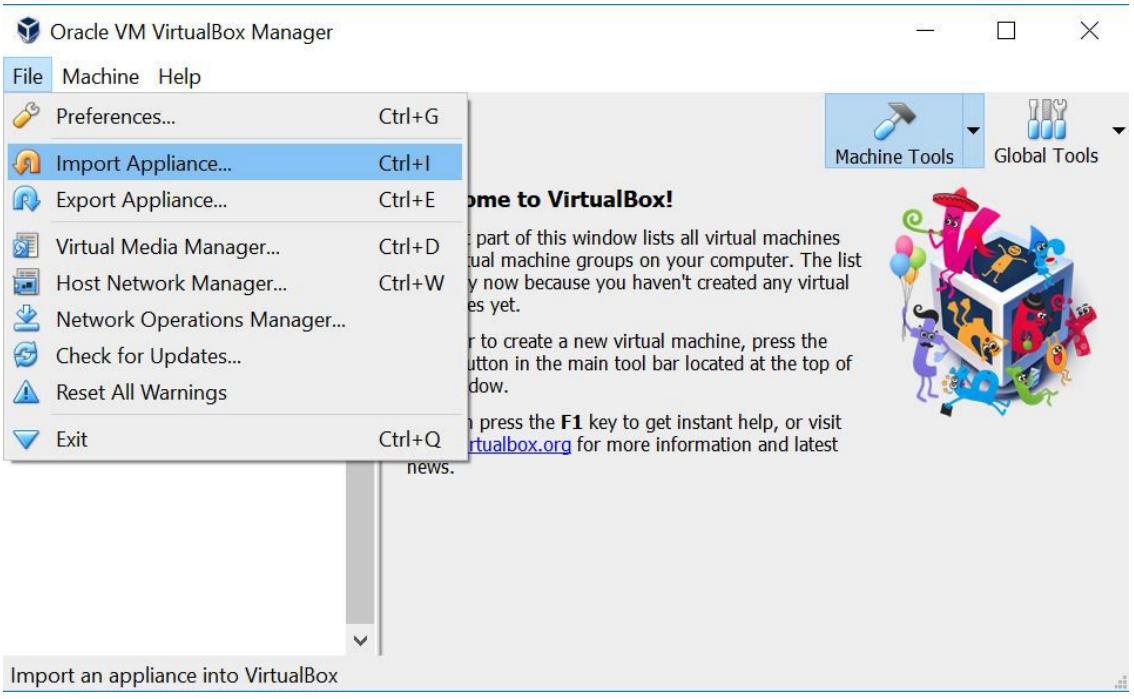


Ignore the warning and click on 'Yes'

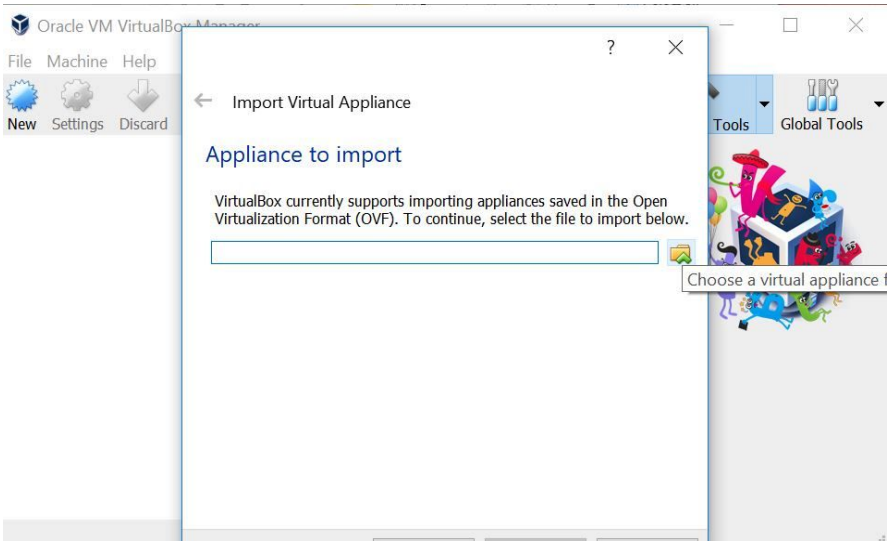
Finally click on Install



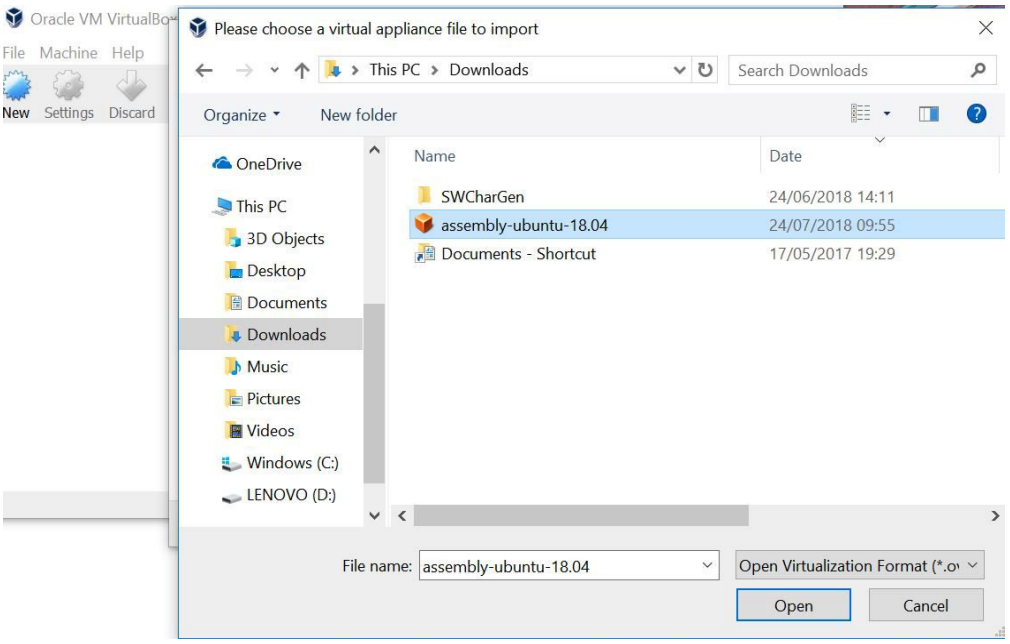
4. Now open VirtualBox from the Windows menu and click on 'Import Appliance...'



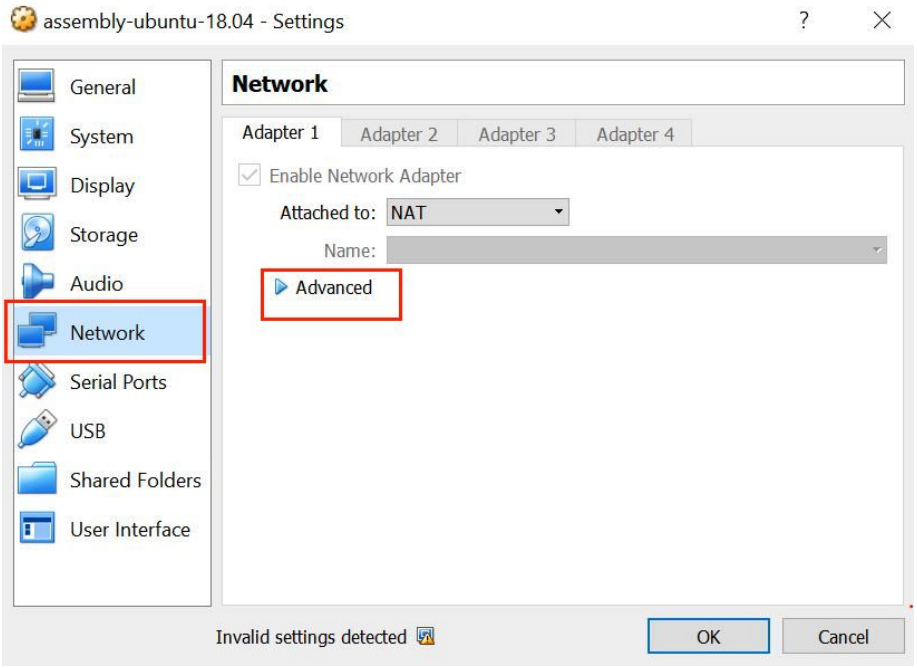
In the dialog box click on the yellow folder icon to be able to select the image you downloaded in step 1



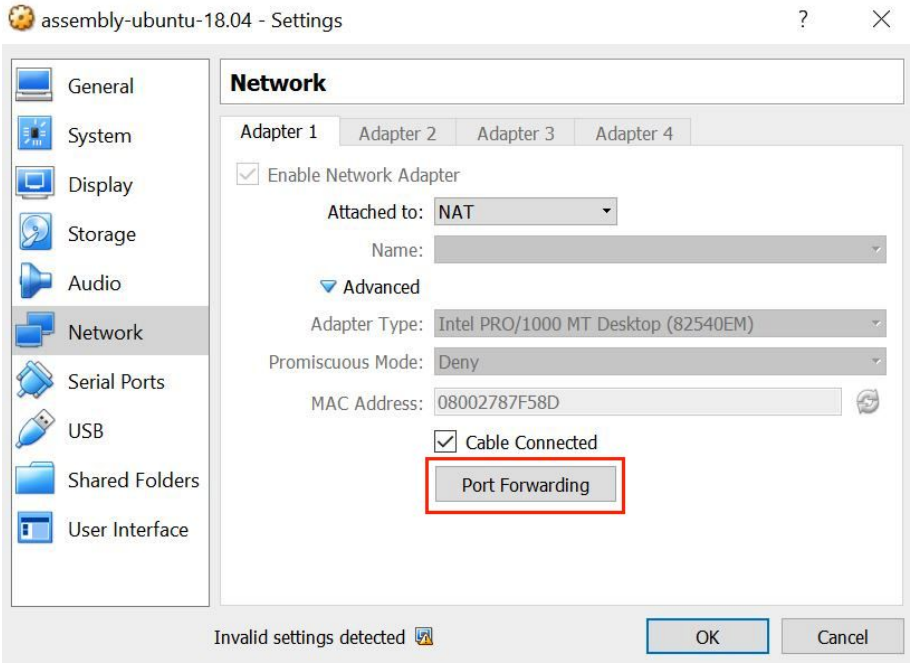
Select the image file (with the suffix ova) you downloaded and click open to import the VM.



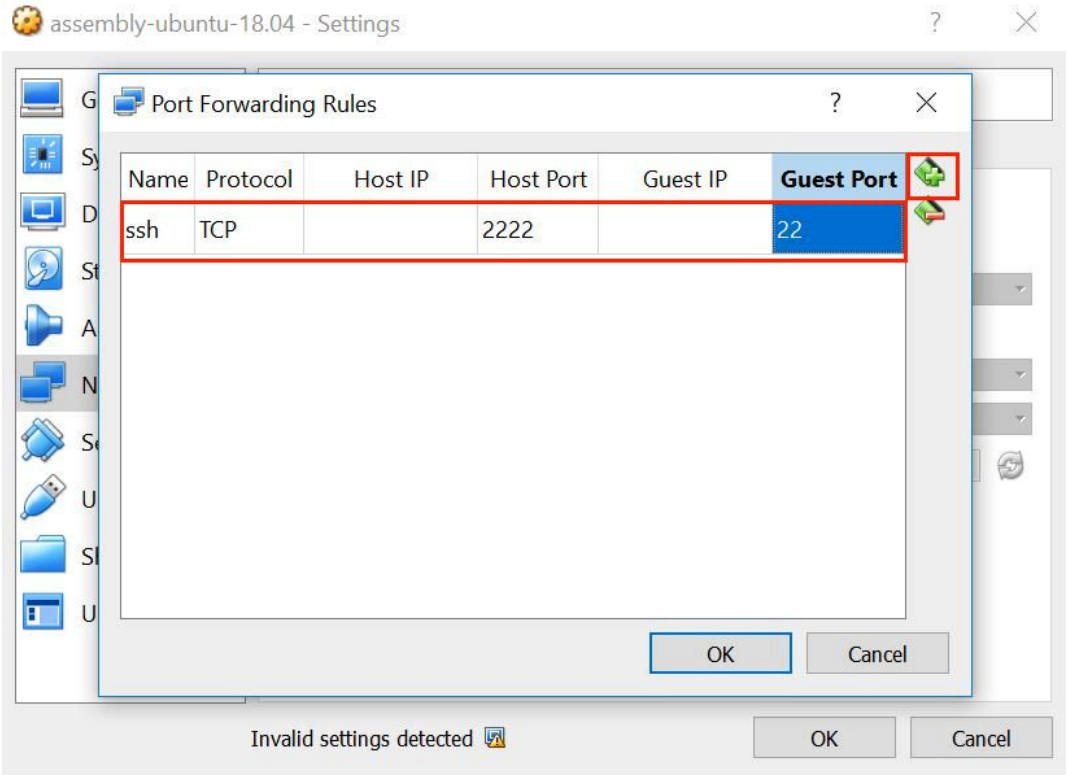
5. Before starting the VM you will need to find if virtualisation is enabled on your laptop/PC. Follow this article to determine if it is enabled. This is easiest on Windows 8/10 machines. If it is not, try enabling Virtualisation capability in the BIOS following the appropriate steps in this article.
6. Now configure the network so that we can connect to the machine later. Select the machine you have just imported (it will be called assembly-ubuntu-18.04) and click on settings in VirtualBox Manager. Select Network (it should say attached to NAT) and then click on the Advanced drop down icon



Then click on 'Port Forwarding'.



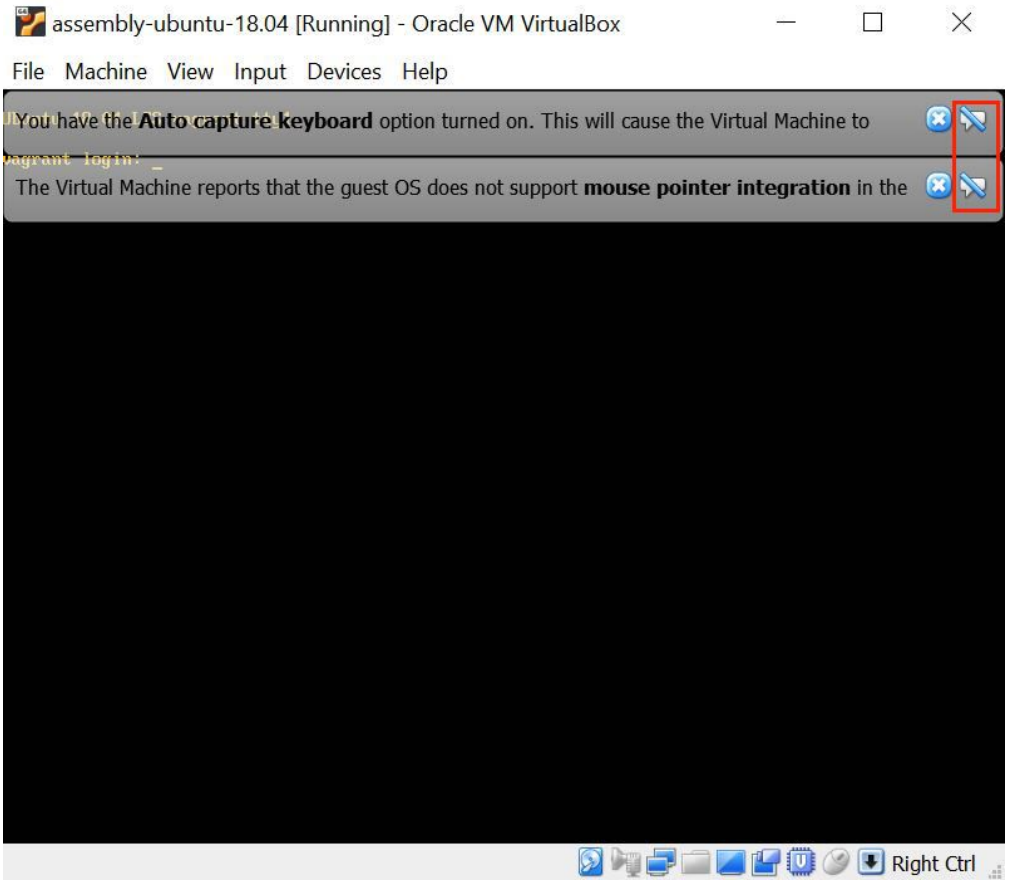
And finally click on the icon with a plus and enter the settings as below



Once you have done that press 'OK' until the setting has been saved.

Connecting to the a Linux machine (in this case a VM) using ssh/PuTTY

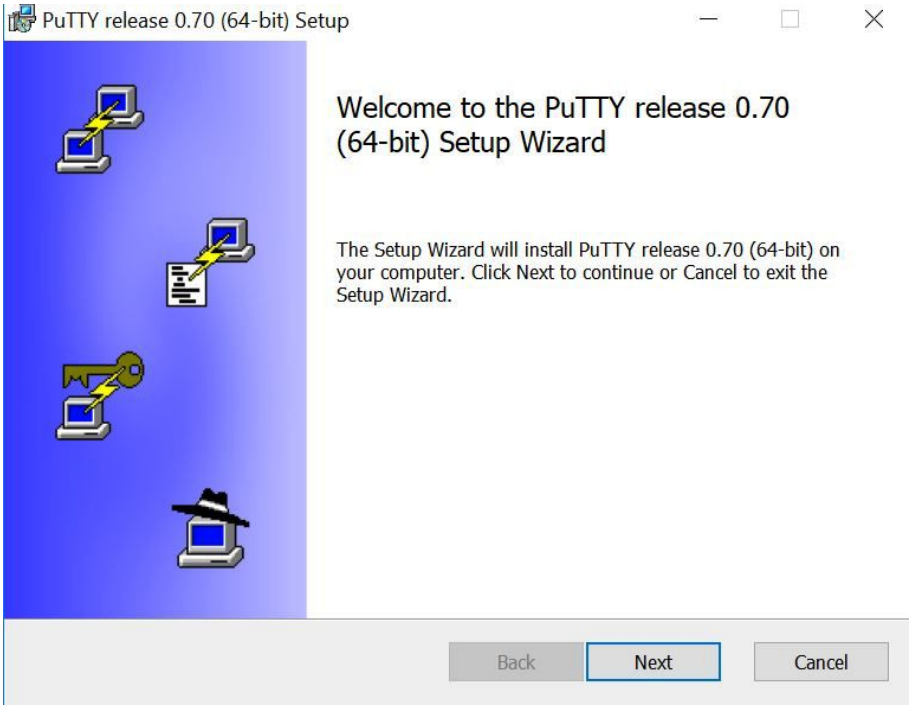
1. You can now start your VM. Select the assembly-ubuntu-18.04 machine in VirtualBox Manager and press the green start arrow. If all proceeds well. You should see a screen as follows:



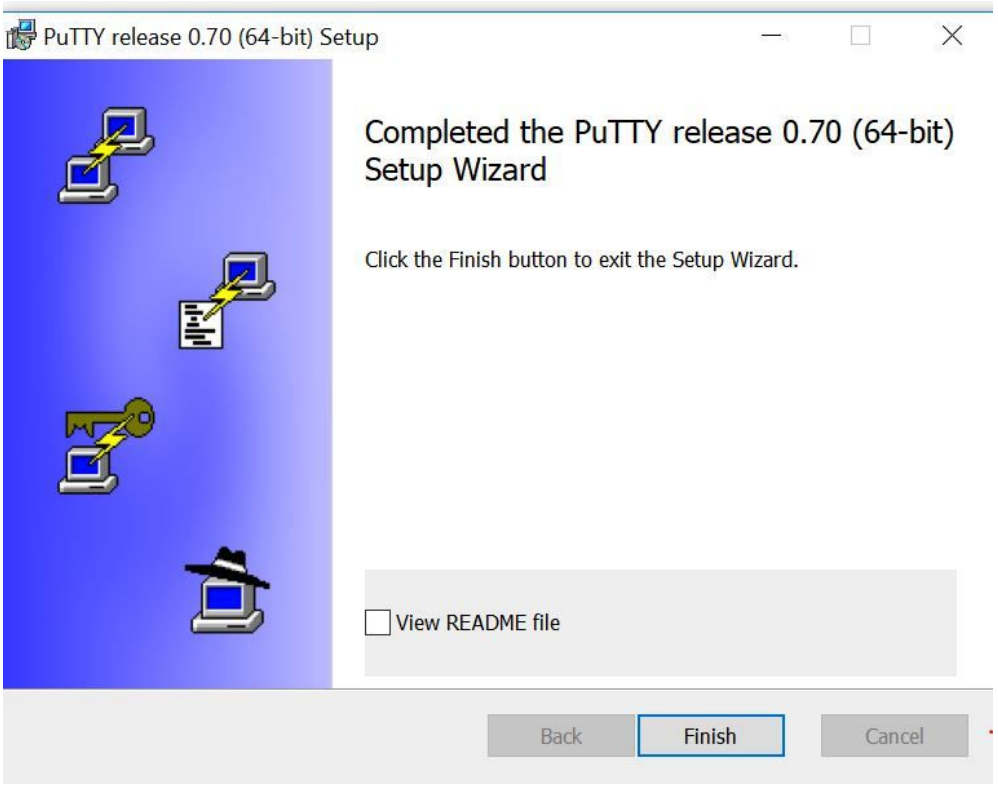
Click on the two dismiss notification buttons so you won't see them again.

2. In order to connect to the VM we will use a tool called PuTTY for 2 reasons
    a. It is the same tool that would be used to connect to a 'real' Linux server
    b. It provides a secure connection

Go to the PuTTY download page and download the relevant MSI installer. It should be a 64-bit installer. Once it has downloaded double click the MSI installer and walk through the installation steps.

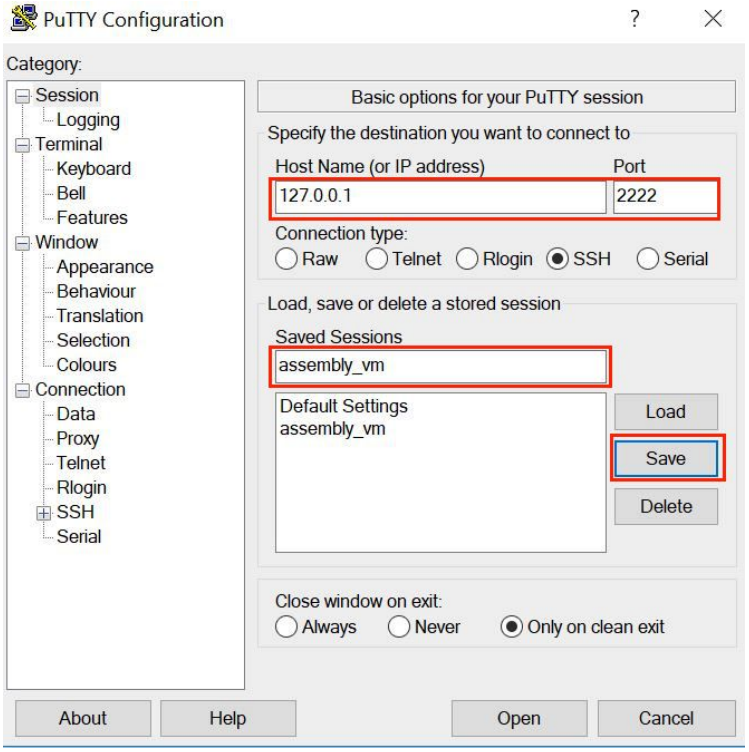


Click through Next until you reach the Finish page (uncheck the View README file). Click on Finish

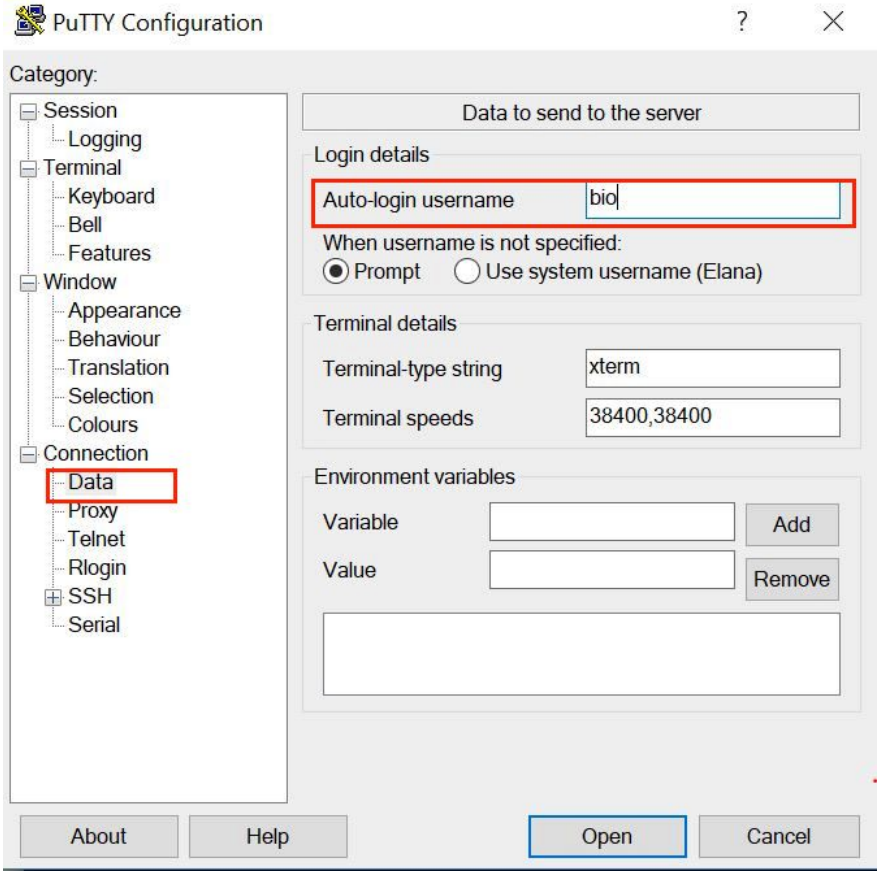


9. Open PuTTY from the Windows menu and configure it as follows:
    a. Enter an IP address of 127.0.0.1 and a Port of 2222

b. Enter a Saved Session name of assembly_vm and click save

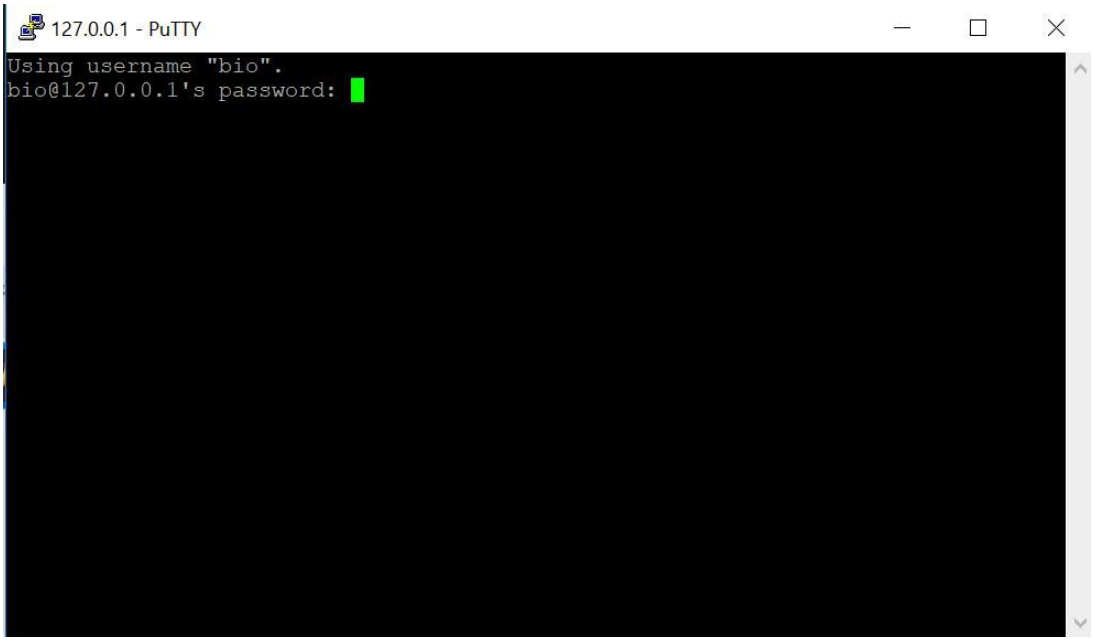


c. Click on ‘Data’ under ‘Connection’, enter bio in the ‘Auto-login username’ field and then back to ‘Session’ and Save again

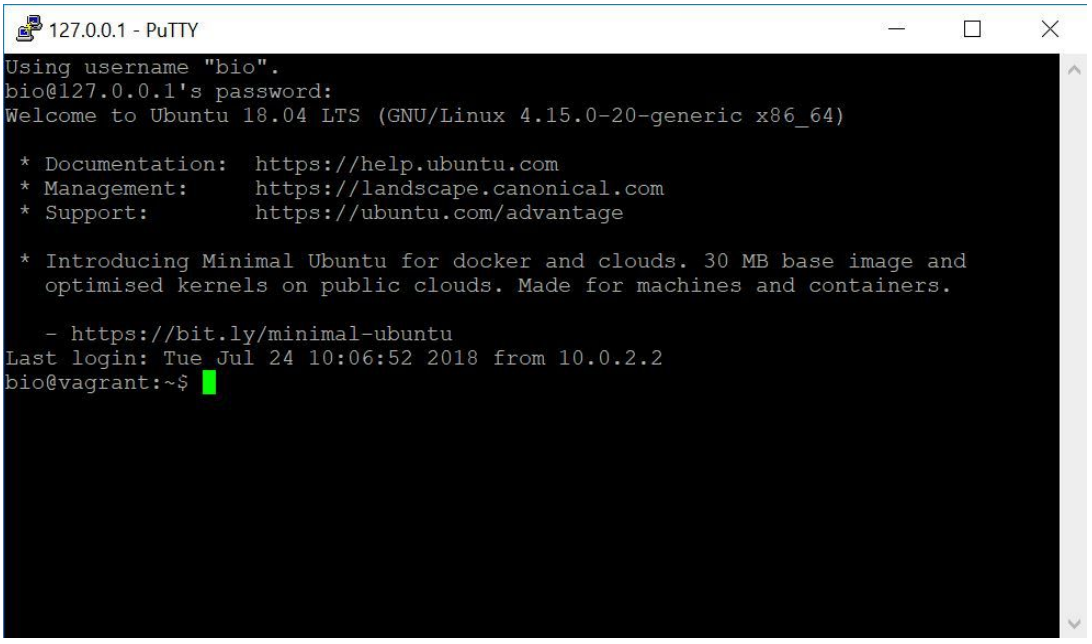


d. Now click on ‘Open’. (For future sessions just select assembly_vm, click ‘Load’ and then ‘Open’)

e. A login window will appear where you should type the password 'bio101'



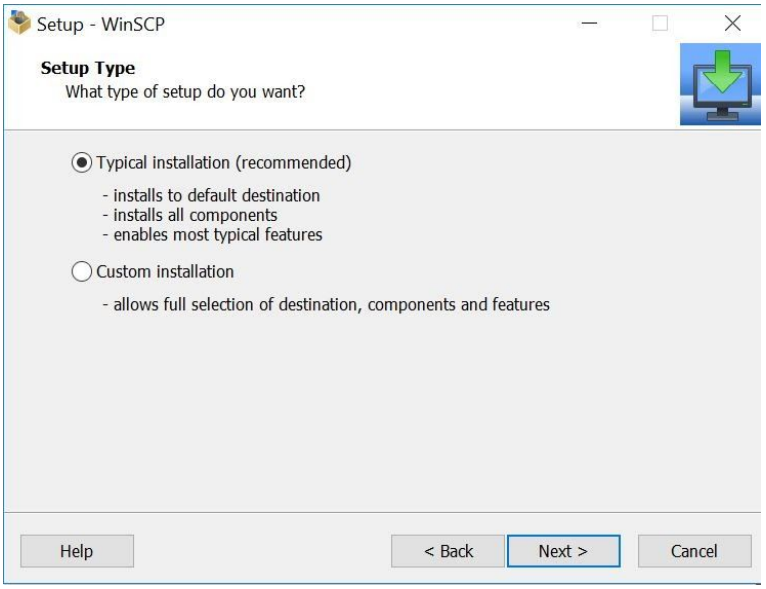
f. If login is successful you should see a screen such as that below and you are good to proceed with the rest of the tutorial. Congratulations, you have a working Linux box!!



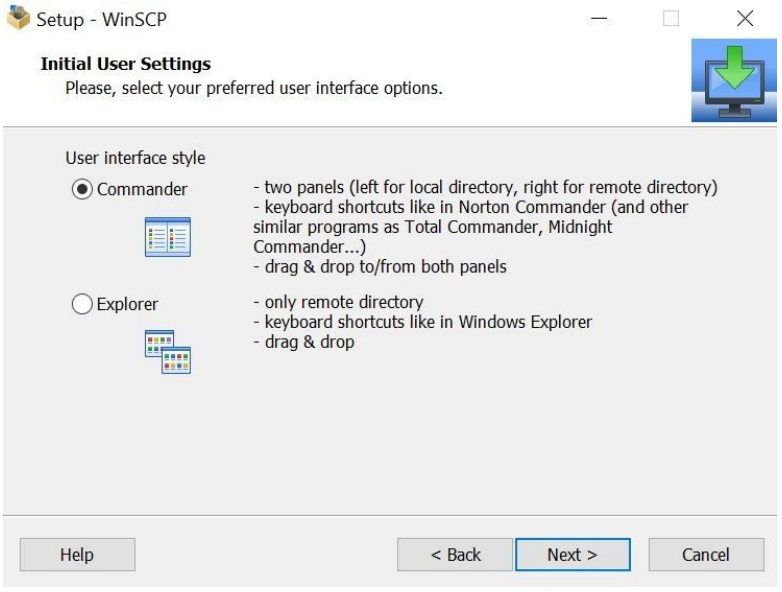
Transferring files to a Linux machine (in this case a VM) using SCP

10. In order to transfer files onto the linux machine you will use WinSCP. Install and configure this as follows:
    a. Download from this link https://winscp.net/download/WinSCP-5.13.3-Setup.exe

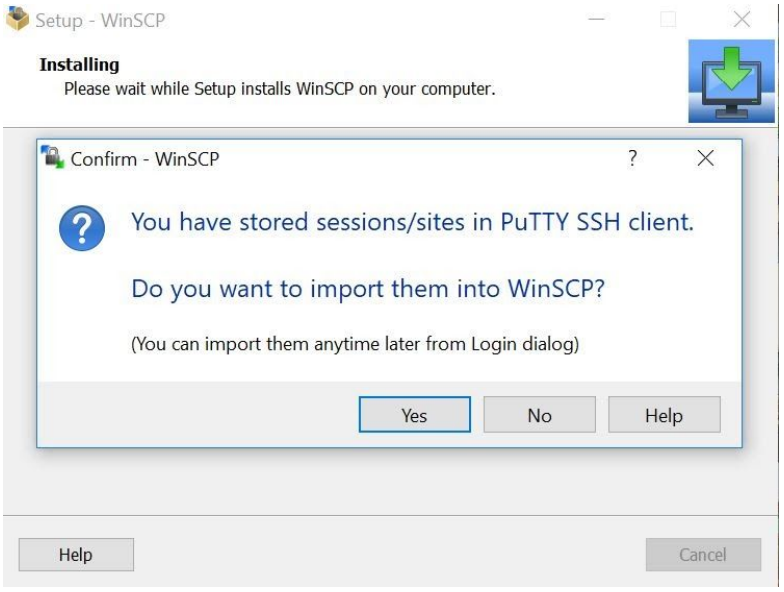
b. Double click the installer and during the install process select Typical

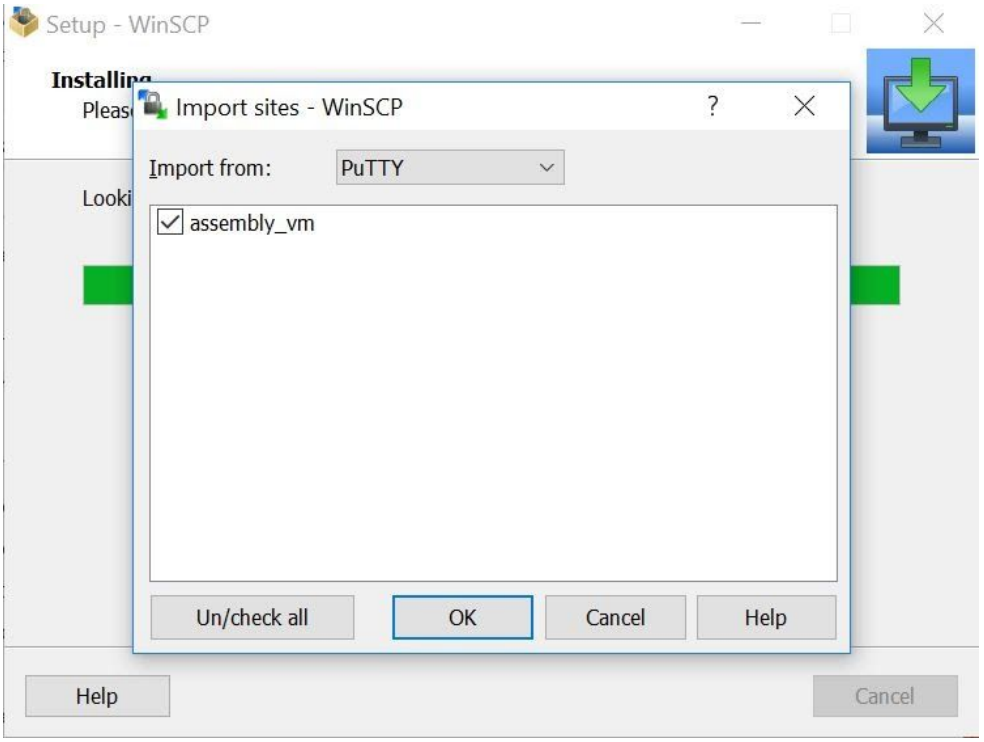


And then Commander for the user interface style



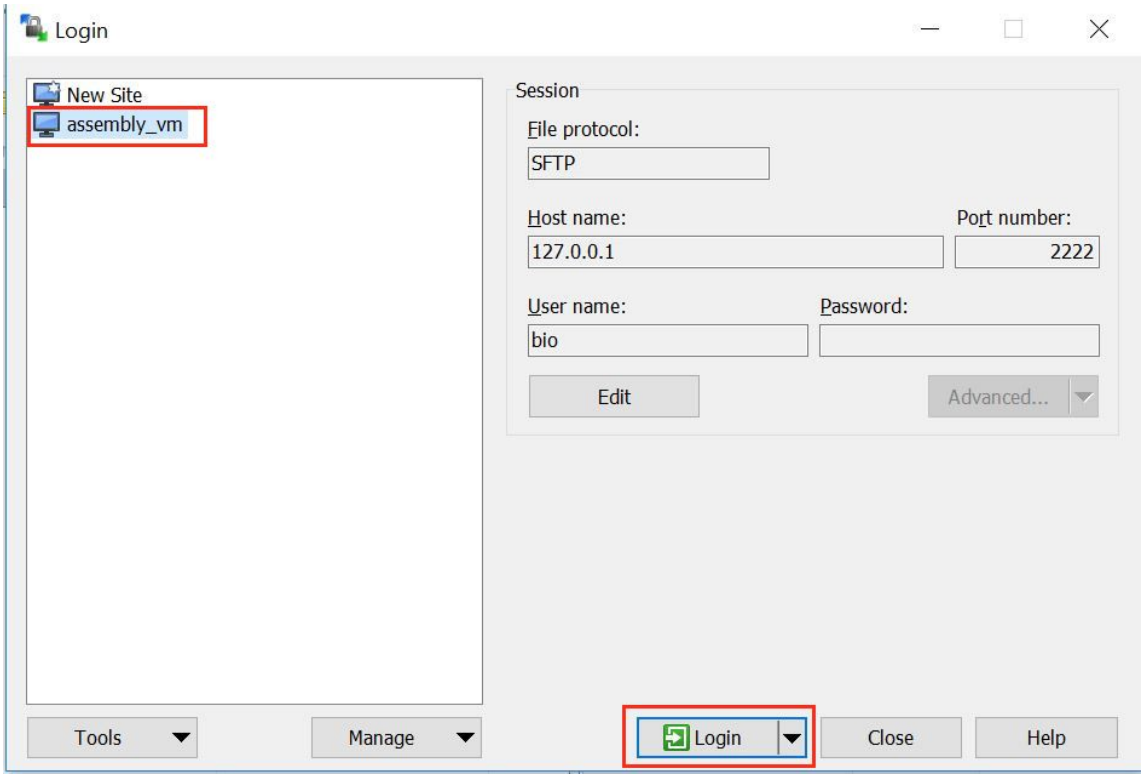
When asked if you want to import settings stored sessions/sites from PuTTY click Yes.



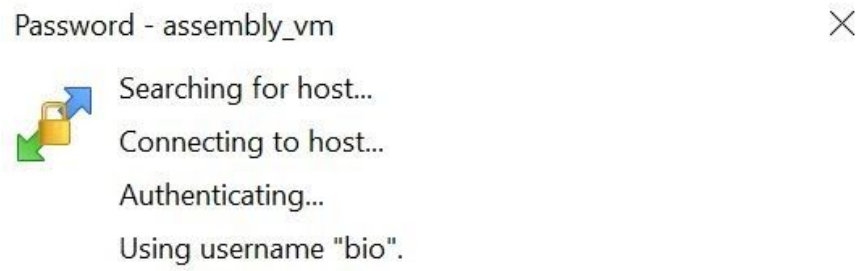


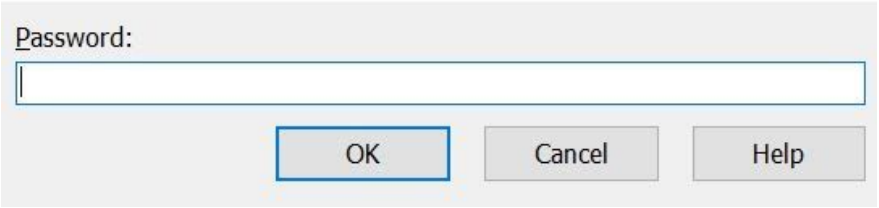
Finally uncheck the Open Getting started page and click Finish

c. Now open WinSCP by selecting it from the Windows menu, select the imported assembly_vm and click 'Login'

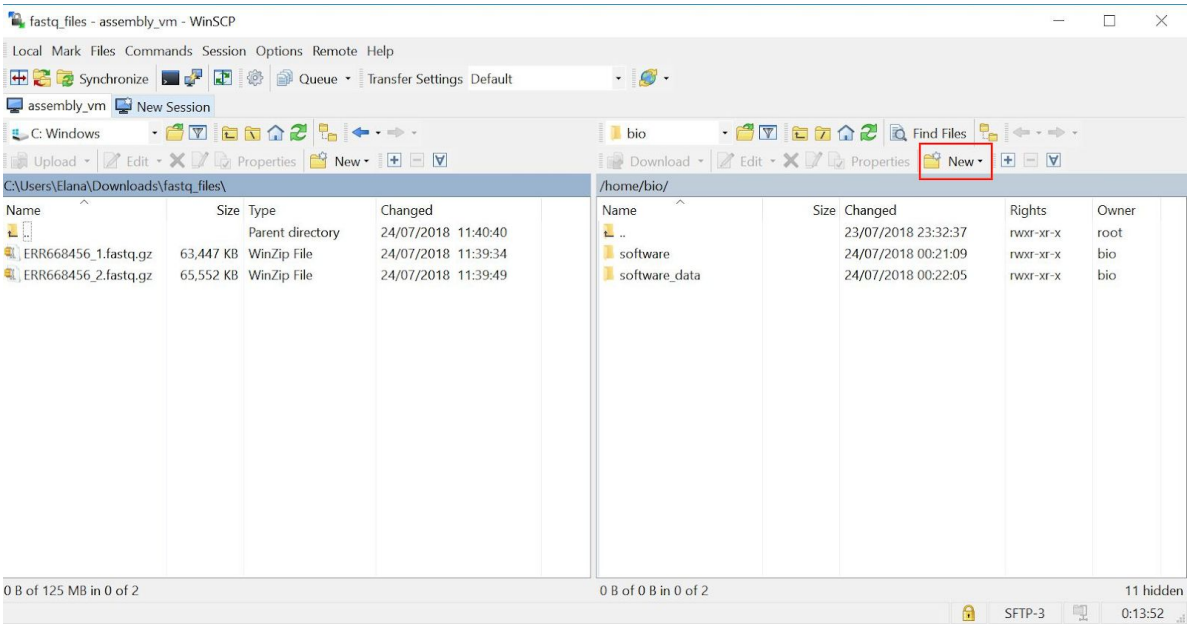


The program will now login to the virtual image. The password is the same as for the ssh connection using PuTTY (bio101)

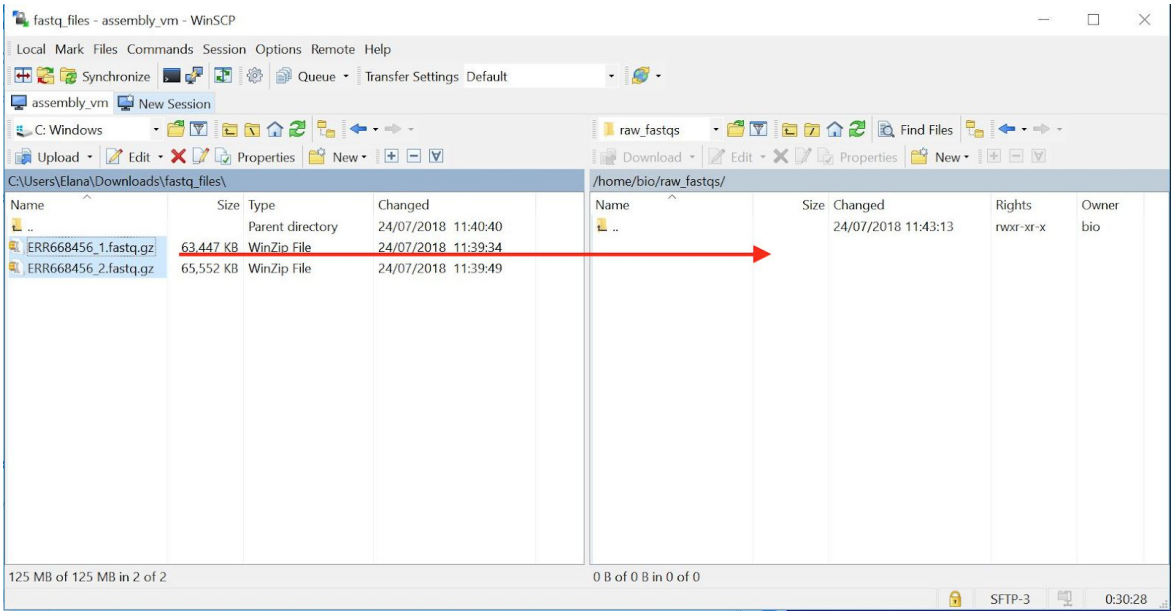




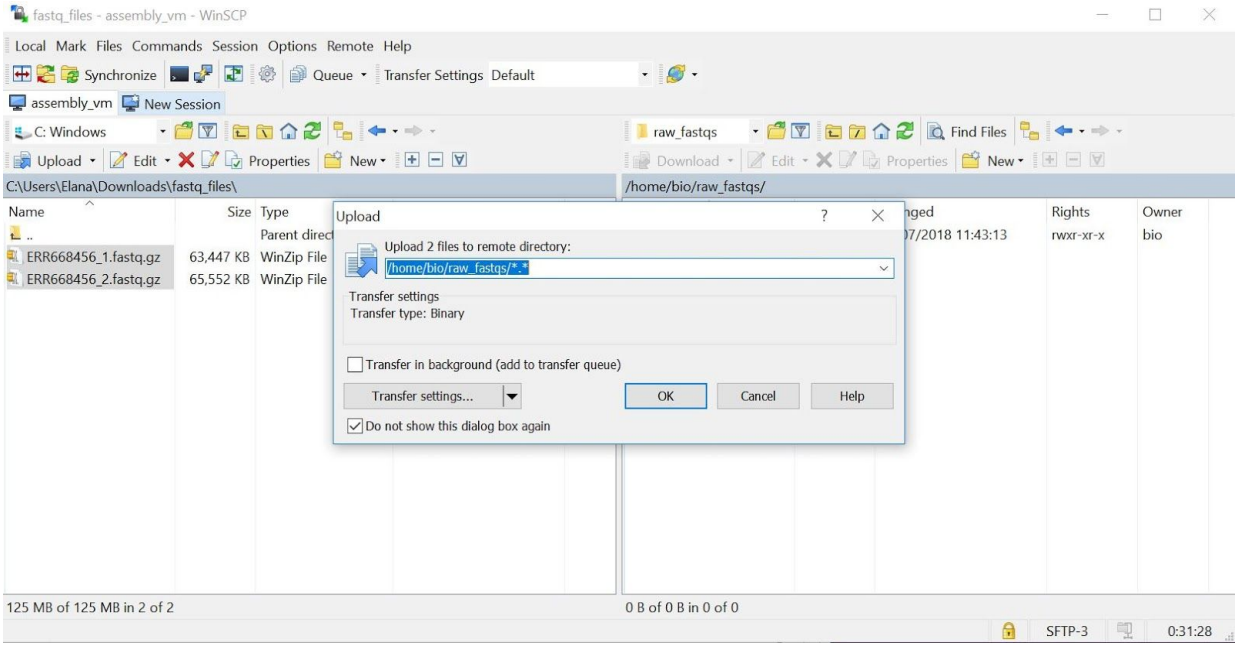
You will see a list of the files on the remote server (in this case your virtual box) and a list of local files. Make a new directory called raw_fastqs by clicking on the 'New Folder' button



Navigate to the local directory where you downloaded the fastqs from the Galaxy assembly tutorial and drag them across to the raw_fastq directory you have created



Click OK when asked if you want to transfer the files



Once this has completed you are now able to start assembling the fastq files on the command line. Are you ready?...

Assembling genomes on the command line

1. First of all connect to the Linux VM using PuTTY as described above using the 'Load' and 'Open' buttons. Type in the password (bio101)
2. Change directory to the raw_fastqs directory and check that the two fastq files you uploaded are present.

```
cd raw_fastqs/
ls -l
```

```
-rw-rw-r-- 1 bio bio 64969275 Jul 27 17:18 ERR668456_1.fastq.gz
-rw-rw-r-- 1 bio bio 67124517 Jul 27 17:18 ERR668456_2.fastq.gz
```

3. **Pre-trimming QC**
   You will now use fastqc to perform a quality assessment of the raw fastq files. Return to the top level directory, make a directory to receive the output of fastqc and then run fastqc. This is a simple command in the format:

   fastqc <FASTQ FILE 1> <FASTQ FILE 2> ….. <FASTQ FILE n> -o <OUTPUT DIRECTORY>

   The -o parameter specifies the path to the output directory

   In our case the commands are:

   cd ..
   mkdir qc_pre_trimming

   ```
   fastqc raw_fastqs/ERR668456_1.fastq.gz raw_fastqs/ERR668456_2.fastq.gz -o
   qc_pre_trimming
   ```

   Look in the output directory to see the outputs

   ```
   ls -l qc_pre_trimming
   ```

   ```
   -rw-rw-r-- 1 bio bio 776345 Jul 30 14:51 ERR668456_1_fastqc.html
   -rw-rw-r-- 1 bio bio 524237 Jul 30 14:51 ERR668456_1_fastqc.zip
   -rw-rw-r-- 1 bio bio 779120 Jul 30 14:51 ERR668456_2_fastqc.html
   -rw-rw-r-- 1 bio bio 530225 Jul 30 14:51 ERR668456_2_fastqc.zip
   ```

   Use WinSCP to download these files and view them by double-clicking them. These should look very similar to those you saw in Galaxy. You may have to click on the refresh button  on the right hand side to view the files.

4. **Trimming reads to remove low quality data and adapters**
   As in the Galaxy tutorial you will now trim the reads using trimmomatic. The general format for the command is:

   trimmomatic PE <FASTQ FILE 1> <FASTQ FILE 2> <TRIMMED PAIRED FASTQ 1> <TRIMMED ORPHAN FASTQ 1> <TRIMMED PAIRED FASTQ 2> <TRIMMED ORPHAN FASTQ 2> <TRIMMING OPTIONS>

   PE refers to paired end read input. The TRIMMED files are the output. In this case we will direct the orphan reads to the 'bin' (specified using /dev/null) and the options will be exactly the same as those used in the galaxy tutorial. Note that for the ILLUMINACLIP option the first item after the : is the path to the file containing the Illumina adapter sequences.

   The commands to type to run the trimmomatic analysis including creation of a directory are:

```
mkdir trimmed_fastqs
trimmomatic PE raw_fastqs/ERR668456_1.fastq.gz
raw_fastqs/ERR668456_2.fastq.gz trimmed_fastqs/ERR668456_1.fastq.gz
/dev/null trimmed_fastqs/ERR668456_2.fastq.gz /dev/null
ILLUMINACLIP:/home/bio/software_data/adapters.fas:2:30:10
SLIDINGWINDOW:4:20 LEADING:25 TRAILING:25 MINLEN:30
```

5. **Post-Trimming QC**
   Now that you have trimmed the fastq files you can re-assess the quality of the reads using the same command as previously but specifying the trimmed fastq files as inputs and a new output directory

```
mkdir qc_post_trimming

fastqc trimmed_fastqs/ERR668456_1.fastq.gz
trimmed_fastqs/ERR668456_2.fastq.gz -o qc_post_trimming
```

6. **Genome size and read coverage estimation**
   We will perform an additional step now (not performed in Galaxy) to assess if the fastq files contain too much data. Any more than 50x coverage of the genome will be unlikely to improve the quality of the assembly so if we have greater read depth it is best to 'downsample' the files in order to optimise the speed of the process.

   a. First estimate the genome size using mash. Type the following command

```
mash sketch -o /tmp/sketch  -k 32 -m 3 -r
trimmed_fastqs/ERR668456_1.fastq.gz  2> mash.stats
```

      The output from mash is not required so we send the output to a temporary directory (/tmp/sketch). The parameters -k 32 and -m 3 refer to the kmer size used when estimating read depths and the number of times a kmer must be seen for it to be included in the analysis. By

specifying 3 this means that infrequent kmers produced by sequencing errors will be discarded. The -r parameter specifies the path to the read file (only 1 file is needed since it will contain random reads across the whole genome). The 2> mash.stats redirects the output that would normally be seen on the screen to a file, here called 'mash.stats'. Examine the contents of this file by typing:

```
more mash.stats
```

You will see output that looks like this

```
Sketching trimmed_fastqs/ERR668456_1.fastq.gz…
Estimated genome size: 4.92043e+06
Estimated coverage:    15.397
Writing to /tmp/sketch.msh..
```

The estimated size is 4.92 Mb, about right for a *Salmonella* genome.

b. Second we will look at the number of bases contained in the reads using the seqtk command. Type the command:

```
seqtk fqchk -q 25 trimmed_fastqs/ERR668456_1.fastq.gz
```

This will look at the number of bases above a quality score of 25. If you scroll to the top of the output you will see output as follows:

```
min_len: 30; max_len: 100; avg_len: 97.58; 8 distinct quality values
POS   #bases     %A    %C    %G    %T    %N    avgQ  errQ  %low  %high
ALL   109899427  24.7  25.3  25.2  24.8  0.0   37.3  35.7  0.3   99.7
1     1126305    22.0  23.1  39.9  15.0  0.0   33.0  33.0  0.0   100.0
.....
```

c. It is now possible to calculate the mean depth of coverage by multiplying the number of bases in the R1 file by 2 (the assumption being that approximately the same number of bases will be seen in both fastq files in a pair) and dividing this by the genome size. The calculation is:

109899427 x 2 / 4920000 = 44.7 (depth of coverage)

Since anything below 50x coverage will likely yield useful data, in this case it is not necessary to downsample the files.

d. If downsampling was required. The required downsampling factor would be calculated. For example if the depth of coverage was 100x the downsampling factor would be 0.5 and the commands to downsample each fastq from the read pair would be as follows:

```
mkdir downsampled_fastqs
seqtk sample trimmed_fastqs/ERR668456_1.fastq.gz 0.5 | gzip >
downsampled_fastqs/ERR668456_1.fastq.gz
```

```
seqtk sample trimmed_fastqs/ERR668456_2.fastq.gz 0.5 | gzip >
downsampled_fastqs/ERR668456_2.fastq.gz
```

7. **Read Correction**

   You should now perform an additional step not seen in the Galaxy tutorial to correct sequencing errors in the reads using a tool called lighter - see here for a link to the paper that describes the software. The command takes the format:

   lighter -od <OUTPUT DIRECTORY> -r <FASTQ FILE 1> -r <FASTQ FILE 2> -K <KMER LENGTH> <GENOME SIZE (bp)> -maxcor <MAX NUM CORRECTIONS>

   Type the command below to correct the read pair:

   ```
   lighter -od corrected_fastqs -r trimmed_fastqs/ERR668456_1.fastq.gz -r
   trimmed_fastqs/ERR668456_2.fastq.gz -K 32 4920000 -maxcor 1
   ```

   The -od parameter specifies the output directory for the corrected fastqs the -r parameter used twice specifies the paths to the trimmed fastq files which will be corrected. If you had downsampled the fastq files you would have used paths that start with downsampled_fastqs. The -K parameter specifies a kmer to use when read correcting and the genome size. You should use 32 and the genome size estimated above of 4920000. The -maxcore parameter specifies the maximum number of corrections in a 20bp window. Given that sequencing errors on the Illumina platform are usually at a rate of less than 1%, a value of 1 is appropriate here.

8. **Read Merging**

Finally before starting read assembly you should merge the reads. This is because with short inserts the forward (R1) and reverse (R2) reads can overlap and assemblers do not always handle overlapping read pairs well. So to overcome this you should merge the reads if possible. The software used to perform this is called flash. This commands takes the following format:

flash -m <MIN OVERLAP BETWEEN READ PAIRS> -M <MAX OVERLAP EXPECTED> -d <OUTPUT DIRECTORY> -z <FASTQ FILE 1> <FASTQ FILE 2>

You should set -m to 20 and -M to 100. The -z parameter will gzip the files produced to compress them and use the same format as we have been using throughout the tutorial. So in this case the command you should type is:

```
flash -m 20 -M 100 -d merged_fastqs -o ERR668456 -z corrected_fastqs/
ERR668456_1.cor.fq.gz corrected_fastqs/ERR668456_2.cor.fq.gz
```

After this has finished, look in the merged_fastqs directory using ls and you will see the 3 files that will be used in the assembly step

```
ls -l merged_fastqs/

-rw-r--r-- 1 bio bio 24022582 Jul 30 16:35
ERR668456.extendedFrags.fastq.gz
-rw-rw-r-- 1 bio bio     1042 Jul 30 16:35 ERR668456.hist
-rw-rw-r-- 1 bio bio     5708 Jul 30 16:35 ERR668456.histogram
-rw-r--r-- 1 bio bio 42312670 Jul 30 16:35
ERR668456.notCombined_1.fastq.gz
-rw-r--r-- 1 bio bio 43817398 Jul 30 16:35
ERR668456.notCombined_2.fastq.gz
```

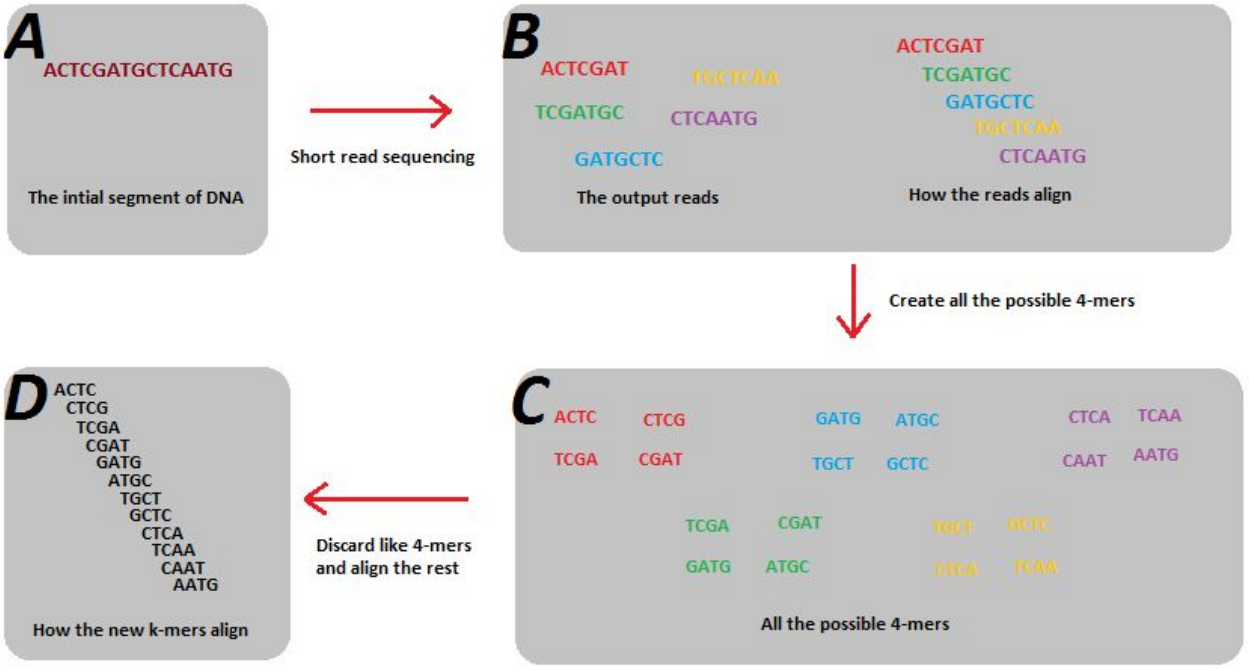
The ERR668456.extendedFrags.fastq.gz file represents the reads that have been merged and the files ERR668456.notCombined_1.fastq.gz and ERR668456.notCombined_2.fastq.gz represent the reads that have a good insert size and the reads do not overlap.

9. Assembly

Finally we have reached the stage where the reads have been processed so they are in the best state to be assembled. We will use the SPAdes assembler. The format of the command to run this is:

spades.py --pe1-1 <NOT MERGED FASTQ FILE 1> --pe1-2 <NOT MERGED FASTQ FILE 2> --pe1-m < MERGED FASTQ FILE> --only-assembler -o <OUTPUT DIRECTORY> --tmp-dir <TEMP DIRECTORY> -k <COMMA SEPARATED KMER LIST> --memory <MAX AMOUNT OF RAM TO USE>

The fastq files are those from the previous merging step. Since you have already performed read correction we will only perform assembly and therefore use the --only assembler parameter. The assembly algorithm breaks the reads into kmers (short DNA sequences of length k). These are used to construct De Bruijn graphs from which contiguous stretches of DNA (contigs) are pieced together. This figure from wikipedia summarises the process:



If you would like to read more then look at this article and follow the links contained within it. For the purposes of assembly, several kmers are applied and then the results combined. It is usual to start fairly small (21) and then use several more kmer sizes. These should be odd numbers and spaced evenly and never be more than 90% of the read length. You will use a range of kmers that satisfy these requirements. Since your VM has limited memory we can restrict the amount of memory that SPAdes use using the --memory parameter. The exact command you should type is:

```
spades.py --pe1-1 merged_fastqs/ERR668456.notCombined_1.fastq.gz --pe1-2
merged_fastqs/ERR668456.notCombined_2.fastq.gz --pe1-m
merged_fastqs/ERR668456.extendedFrags.fastq.gz --only-assembler  -o
assembly --tmp-dir /tmp -k 21,33,43,53,63,75 --memory 2
```

This will take a while so now may be a good time to take a cup of coffee or tea and read up on assembly or kmers :)

10. **Quality Assessment**

To assess the quality of the assembly you will use quast. First you should filter out small and low coverage contigs. Use a small Python script to perform this:

```
python3 software/filter_contigs.py -f assembly/contigs.fasta -l 500 -c 3
```

where -f is the path to the assembly -l is the minimum contig size to keep and -c is the minimum coverage depth to keep (this will exclude spurious contigs). This will produce a file in the assembly directory called `contigs.filter_gt_500bp_gt_3.0cov.fasta`

The format of the command is:
quast.py <CONTIGS FASTA FILE> -o <OUTPUT DIRECTORY>
Therefore type this command:

```
quast.py assembly/contigs.filter_gt_500bp_gt_3.0cov.fasta -o assembly
```

Have a look at the report file produced using the more command:

```
more assembly/report.tsv
```

```
Assembly                    contigs.filter_gt_500bp_gt_3.0cov
# contigs (>= 0 bp)         91
# contigs (>= 1000 bp)      81
# contigs (>= 5000 bp)      66
# contigs (>= 10000 bp)     55
# contigs (>= 25000 bp)     46
# contigs (>= 50000 bp)     31
Total length (>= 0 bp)      4893332
Total length (>= 1000 bp)       4885805
Total length (>= 5000 bp)       4847762
Total length (>= 10000 bp)      4765732
Total length (>= 25000 bp)      4615562
Total length (>= 50000 bp)      4025781
# contigs           91
Largest contig      488881
Total length        4893332
GC (%)      52.01
N50         143655
N75         65748
L50         13
L75         25
# N's per 100 kbp           0.00
```

This is almost identical to the results obtained on galaxy.

Congratulations

If you have got this far you have successfully performed a *de novo* assembly on the command line. Now it's your turn to try the same process with a second pair of fastqs

Exercise: Assembly of a new sample

Create a new directory within your home directory (hot tip: to return to your home directory just type `cd↵` on its own). Call it something like `ex2`.
Now download the following pair of fastq files

- https://drive.google.com/file/d/165fwtRm8BfAkeYVYP-Pr6HG_febCVx6e/view?usp=sharing
- https://drive.google.com/file/d/16-iHD15kVWpa7z47NsAN3jK_mS_z2zb6/view?usp=sharing

Proceed with the assembly as before, download the following output files using WinSCP and send them as a zip file.

- qc_pre_trimming directory
- qc_post_trimming directory
- contigs.fasta and scaffold.fasta from the SPAdes assembly
- report.tsv from quast

Version Control Table

Title	Command Line Genome Assembly Tutorial			
Description	A document describing how to perform *de novo* genome assembly for a single bacterial genome starting from a pair of Fastq files generated with short read sequencing using command line tools and mirroring the process used in the complementary document demonstrating assembly on the Galaxy web platform.			
Created By	Anthony Underwood			
Date Created	1st August 2018			
Maintained By	Anthony Underwood			
Version Number	**Modified By**	**Modifications Made**	**Date Modified**	**Status**
1.0	Anthony Underwood	First version	1st August 2018	First Live Version