How to transfer data from IRDS at UWA to Zeus and Magnus

In many cases, you will be able to transfer from IRDS to your Nimbus instance by using a program such as FileZilla or CyberDuck. However, this will use your local internet bandwidth, which may be an issue if you are not at a University/Institute site and you need to transfer large files. Additionally, if you are on a laptop, moving away from an internet connection may disrupt your upload. An alternative is setting up the transfer directly on Nimbus via 'screen' (https://linuxize.com/post/how-to-use-linux-screen/). Then, you can close your laptop and not worry about your download, or bandwidth.

However, it is important to note that IRDS will not allow transfer of files with unusual extensions, such as .bam, .bai, .md5, and others. One way around this is to zip/tar your files before transfer. It is also possible to change the extension to an acceptable format for transfer (i.e. change .bam to .bam.txt).

Rclone is a command line transfer client with webdav capabilities. By setting up rclone you can improve the ease and speed of your transfer from IRDS.

With rclone (recommended)

Part 1. Download rclone

You will need to download rclone. This will need to be placed either in your home or group directory.

Download rclone

```
curl -O https://downloads.rclone.org/rclone-current-linux-amd64.zip
unzip rclone-current-linux-amd64.zip
rm rclone-current-linux-amd64.zip
cd rclone-*-linux-amd64
ls [this will reveal the rclone binary file called 'rclone']
```

Then, you will need to decide if you would prefer to copy the rclone binary into an directory that is already listed in your PATH, or add the rclone directory to your PATH. If you are not sure, it's most likely easier to do the second option.

Part 2. Adding rclone to your PATH

Double check you are in the rclone directory! Then add the current directory to your path with the following command, and then either close and reopen your terminal window

```
echo "export PATH=$PWD:\$PATH" >> ~/.bashrc
```

You should then be able to access rclone from any directory by typing rclone.

Part 3. Congifuring rclone

You will enter the rclone settings by typing `rclone config`. The program then guides you through the configuration process as shown below. Enter the answers as per this guide to add your IRDS setup to rclone. This looks complicated but you will see it's quite simple as you move through it. More info can be found here: https://rclone.org/webdav/

Rclone configuration

```
rclone config

**********************************
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
name> irds [or whatever you want]
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Webdav
   \ "webdav"
[snip]
Storage> webdav [or 34 if there are number options]
URL of http host to connect to
Choose a number from below, or type in your own value
1 / Connect to example.com
   \ "https://example.com"
url> https://unidrive.uwa.edu.au/staff/irds/PERKINS-AA-001/ [swap out to your IRDS name]

Name of the Webdav site/service/software you are using
Choose a number from below, or type in your own value
1 / Nextcloud
   \ "nextcloud"
2 / Owncloud
   \ "owncloud"
3 / Sharepoint
   \ "sharepoint"
4 / Other site/service or software
   \ "other"
vendor> 1
User name
user> phemeID#
Password.
y) Yes type in my own password
g) Generate random password
n) No leave this optional password blank
y/g/n> y
Enter the password:
password: [current pheme password, doesn’t matter about special characters]
Confirm the password:
password:
Bearer token instead of user/pass (eg a Macaroon)
bearer_token>
[Just leave this blank by hitting enter, and then say do then same when asked about advanced options]
--------------------
[remote]
type = webdav
url = https://example.com/remote.php/webdav/
vendor = nextcloud
user = user
pass = *** ENCRYPTED ***
bearer_token =
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

Once the config is done, you can test that your setup has worked by typing

```
rclone lsd irds:
```

This should show you all the files in your IRDS directory.

Part 4. Copy files to your instance

The copy process is quite straightforward. In the following example, we will change into the directory where we want to place the data. Then we give the rclone command, and tell it to transfer for the current working directory through use of the dot.

```
cd /dir/for/data/
rclone copy irds:path/to/my/irds/file .
```

More information on the full usage of rclone can be found here: https://rclone.org/commands/

Copying data back to IRDS

If you would like to use rclone to transfer data back to IRDS, you might find there is an issue where large files are not transferrable. Rclone provides a function called chunker, which transparently breaks large files into chunks for transfer, then reassmebles them at the other end. The overall documentation can be found here.

```
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
name> overlay
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Transparently chunk/split large files
   \ "chunker"
[snip]
Storage> chunker
Remote to chunk/unchunk.
Normally should contain a ':' and a path, eg "myremote:path/to/dir",
"myremote:bucket" or maybe "myremote:" (not recommended).
Enter a string value. Press Enter for the default ("").
remote> remote:path [e.g. irds]
Files larger than chunk size will be split in chunks.
Enter a size with suffix k,M,G,T. Press Enter for the default ("2G").
chunk_size> 100M
Choose how chunker handles hash sums. All modes but "none" require metadata.
Enter a string value. Press Enter for the default ("md5").
Choose a number from below, or type in your own value
 1 / Pass any hash supported by wrapped remote for non-chunked files, return nothing otherwise
   \ "none"
 2 / MD5 for composite files
   \ "md5"
 3 / SHA1 for composite files
   \ "sha1"
 4 / MD5 for all files
   \ "md5all"
 5 / SHA1 for all files
   \ "sha1all"
 6 / Copying a file to chunker will request MD5 from the source falling back to SHA1 if unsupported
   \ "md5quick"
 7 / Similar to "md5quick" but prefers SHA1 over MD5
   \ "sha1quick"
hash_type> md5
Edit advanced config? (y/n)
y) Yes
n) No
y/n> n
Remote config
--------------------
[overlay]
type = chunker
remote = remote:bucket
chunk_size = 100M
hash_type = md5
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

Then you should be able to transfer data back to IRDS in small chunks using something like the following command:

```
rclone copy sample.fastq.gz chunker:
```

Wget guide (archived)

This guide will help you transfer data from the IRDS to Zeus/Magnus without the need to mount IRDS. This can be be used for automated batch transfers of your data, parallelising using up to 16 cores. However, it is important to note that IRDS will not allow transfer of files with unusual extensions, such as .bam, .bai, .md5, and others. One way around this is to zip/tar your files before transfer. It is also possible to change the extension to an acceptable format for transfer (i.e. change .bam to .bam.txt). IRDS may close the connection after several hours, so it is important to check your log files to see if you transfer completed successfully. If you use the -c flag as specified below, you will be able to resume failed transfer by simply repeating your transfer script.

Part 1. Set up wget

You will need to make a ~/.wgetrc file for your user name and password. These should not be put on the wget command line as other users can see them by using "top" or "ps".

```
##Change into your home directory
cd

##make ~/.wgetrc file
touch .wgetrc
##change the permissions on the file so only you can view and edit
chmod 600 .wgetrc
##open .wgetrc for editing nano .wgetrc
#Add the following two lines with your credentials, then save the file
--http-user=UWA staff number
--http-passwd=Pheme password
```

A note on passwords

This does not apply to passwords in your ~./wgetrc file:

UWA enforces use of 'special characters' in passwords for security reasons. Special characters generally refer to the following **`[ \ ^ $ . | ? * + ( )`**. These characters can cause issues in the commandline environment, making your password invalid because it's not been read correctly. Let's say your password is **Ph3m3Password!** To make this be evaluated correctly, you would need to 'escape' the special character (in this case an exclamation mark) by placing a backslash before the exclamation mark e.g. **Ph3m3Password\!**

If your password has multiple special characters, be sure to place the backslash before each special character.

Part 2. Set up your transfer scripts

1. In the directory where you would like to download the files at Pawsey, make a .txt file with the full paths of the files you would like to transfer (eg. https://unidrive.uwa.edu.au/staff//irds/path/to/your/files), with one file path per line. Alternatively, compress the files you would like to transfer into one zip/tar.gz file and use that path instead.
    a. **Note: the double forward slashes between staff and irds (i.e. staff//irds) are deliberate.**
2. Make an sbatch script to perform the transfer. We have generated a semi-automated script for you here. It requires your input file list to be in the same working directory. There are editable variables at the top for you to change to suit your needs.

```
#!/bin/bash

#editable variables. Change these to suit your needs and pawsey account
my_account="pawsey"
job_prefix="data_trf"
run_time="00:10:00"
input_file_list="test_input.txt"

###Add slurm parameters to the job script
echo "#!/bin/sh
#SBATCH --account=${my_account}
#SBATCH --partition=copyq
#SBATCH --nodes=1
#SBATCH --cpus-per-task=16
#SBATCH --output=${job_prefix}-%j.log
#SBATCH --error=${job_prefix}-%J.log
#SBATCH --time=${run_time}" >> sbatch_irds_trf.sh.tmp

#split the input sample list into 16, without breaking up any text lines.
#Gives prefix of "input-" to the output split files
split -n l/16 ${input_file_list} input-

#add the input files to the job script
for file in $(ls input-*)
do
echo "wget -r -np -N -c -i $file &" >> sbatch_irds_trf.sh.tmp
done

#remove the "&" character from the very last line of the job script
sed '$s/&$//' sbatch_irds_trf.sh.tmp >> sbatch_irds_trf.sh

#remove temp file
rm sbatch_irds_trf.sh.tmp

#submit job script to slurm
sbatch sbatch_irds_trf.sh
```

The -c flag allows easy resuming of failed transfers if they fail. The -i flag reads the list of files you made in Part 2, Step 1. Your user name and password should be read automatically from the .wgetrc file you made earlier.

Note that IRDS tends to disconnect users after several hours, so don't bother setting a long wall time as you will almost certainly get disconnected before that. You will need to check your log files when your job completes to know if the transfer failed at any point. If it did fail, simply restart the script and it will continue where it left off. Make sure to delete the sbatch_irds_trf.sh file before restarting your job.

The log files tend to be very long, so you are best to use tail rather than cat or zless.

FTP troubleshooting:

If you are adapting this process for non-IRDS files, you might find that wget will not download via FTP using the above method. One workaround is changing the line

```
echo "wget -r -np -N -c -i $file &"
```

to

```
echo "wget -r -np -N -c $(cat $file) &"
```

This reads the list of files to wget and bypasses the 'no URL' error.

A note on file extensions

IRDS doesn't allow unusual file extensions to be transferred (e.g. bam, bai, md5). If you need to transfer non 'standard' file types, it is recommended to zip/tar them instead. Adding .txt to the file extension may also work, but do that at your own risk.