
ARC User Guide

Release 0.1

ARC Team

Sep 22, 2023

CONTENTS

1	Introduction	1
2	ARC Systems	3
2.1	Operating system	3
2.2	Capability cluster (arc)	4
2.3	Throughput cluster (htc)	4
2.4	GPU Resources	6
2.5	Storage	7
2.6	Software	7
3	Connecting to ARC	9
3.1	Connecting from Linux	9
3.2	Connecting from Windows	10
3.3	Connecting using ARC Graphical Nodes	10
4	Copying data to/from ARC	11
4.1	Copying to the ARC systems	11
4.2	Copying from the ARC systems	12
4.3	Copying data from Mac or Linux PC	12
4.4	Copying to/from ARC using Graphical File Transfer utilities	12
4.5	Troubleshooting Graphical File Transfers	15
5	Accessing RFS from the ARC desktop environment	17
6	Accessing RFS from the command line	21
6.1	Using the ARC rfs tool	21
6.2	Using smbclient	25
7	Using the ARC desktop environment	29
7.1	Running applications from the desktop	31
8	Submission Script Basics	33
9	Job Scheduling and Resources	37
9.1	Cluster Node Types	37
9.2	The Job Scheduler	37
9.3	Partitions	38
9.4	Submission Scripts	39
9.5	Interactive Jobs	40
9.6	Interactive MPI Jobs	41
9.7	Memory Resources	41

9.8 GPU Resources	42
10 Submitting Priority Jobs	45
10.1 Submit jobs with priority QoS (Quality of Service)	45
11 ARC Storage	47
11.1 Using ARC \$SCRATCH storage	47
11.2 Quota	48
11.3 Backups	49
11.4 Snapshots	49
12 Accessing Installed Software Applications	53
12.1 Environment Modules	53
13 SLURM Reference Guide	57
13.1 Using the SLURM job scheduler	57
13.2 Introduction	57
13.3 Commands	58
13.4 Preparing a submission script	58
13.5 Example 1: job running on a single node	59
13.6 Example 2: job running on multiple nodes	60
13.7 SLURM partitions	60
13.8 Submitting jobs with the command sbatch	61
13.9 Monitoring jobs with the command squeue	61
13.10 Deleting jobs with the command scancel	62
13.11 Environment variables	62
13.12 Job Dependencies	62
13.13 Job Arrays	63
14 SLURM FAQs	65
14.1 How do I submit, check the status, and/or delete a batch job?	65
14.2 Why does my queued job not start?	65
14.3 Why does my job fail with the error “/bin/bash^M: bad interpreter: No such file or directory”	67
14.4 Why does my job fail/die after running for just a few seconds?	67
14.5 Why does my job fail/die after running for a few hours/days?	67
14.6 How can I increase the walltime of a running job?	67
14.7 How can I get an email notification when a job begins/finishes?	67
14.8 How can I check the availability of free compute nodes?	68
15 General FAQs	69
15.1 How do I become a ARC user?	69
15.2 Where should a new ARC user begin?	69
15.3 What systems can I access as a user?	69
15.4 How much disk space do I have?	69
15.5 How do I log on to the ARC systems?	69
15.6 How do I transfer files to/from the ARC systems?	70
15.7 How do I access the ARC systems from outside the University network?	70
15.8 How do I acknowledge ARC in my publications?	71
15.9 Will application X run on the ARC supercomputers faster than on my workstation?	71
15.10 How many credits do I have left?	71
15.11 How do I change my password?	71
15.12 I have forgotten my password. How do I reset my password?	71
15.13 I accidentally deleted files, how do I get them back?	71
16 Host key fingerprints for ARC login nodes	73

INTRODUCTION

The Advanced Research Computing (ARC) service provides access to High Performance Computing (HPC) resources, support, and advice to researchers within the University of Oxford.

At the core of the ARC service are our HPC compute clusters and high-performance data storage systems. The compute clusters provides a large number of CPU and GPU based compute nodes as well as other specialised resources such as high memory nodes for manipulating and working with large data sets and outputs. These resources support a breadth of research at the University including computational fluid dynamics, bioinformatics, and machine learning/AI to name a few application areas. The majority of ARC's resources are connected via a high bandwidth, low latency interconnect (InfiniBand) to facilitate fast and efficient inter-processor communication across the system, as well as provide high bandwidth connectivity to storage. Data storage systems connect to these clusters, and provide both fast (scratch) storage for the fast, low latency parallel storage demanded by modern HPC applications, and bulk storage for facilitating user home directories and project data storage.

ARC SYSTEMS

At the centre of the ARC service are two high performance compute clusters - **arc** and **htc**.

- **arc** is designed for multi-node parallel computation
- **htc** is designed for high-throughput operation (lower core count jobs).

htc is also a more heterogeneous system offering different types of resources, such as GPGPU computing and high memory systems; nodes on **arc** are uniform. Users get access to both both clusters automatically as part of the process of obtaining an account with ARC, and can use either or both.

For more detailed information on the hardware specifications of these clusters, see the tables below:

Cluster	Description	Login Node	Compute Nodes	Minimum Job Size	Notes:
arc	Our largest compute cluster. Optimised for large parallel jobs spanning multiple nodes. Scheduler prefers large jobs. Offers low-latency interconnect (Mellanox HDR 100).	arc-login	CPU: 48 core Cascade Lake (Intel Xeon Platinum 8268 CPU @ 2.90GHz) Memory: 392GB	1 core	Non-blocking island size is 2212 cores
htc	Optimised for single core jobs, and SMP jobs up to one node in size. Scheduler prefers small jobs. Also catering for jobs requiring resources other than CPU cores (e.g. GPUs).	htc-login	CPUs: mix of Broadwell, Haswell, Cascade Lake GPU: P100, V100, A100, RTX	1 core	Jobs will only be scheduled onto a GPU node if requesting a GPU resource.

2.1 Operating system

The ARC systems use the Linux Operating System (specifically CentOS 8) which is commonly used in HPC. We do not have any HPC systems running Windows (or MacOS). If you are unfamiliar with using Linux, please consider:

- Finding introduction to Linux resources online (through Google/Bing/Yahoo etc).
- Working through our brief Introduction to Linux course.
- Attending our Introduction to ARC training course (this does not teach you how to use Linux but the examples will help you gain a greater understanding).

2.2 Capability cluster (arc)

The capability system - cluster name arc - has a total of 305 48 core worker nodes, some of which are co-investment hardware. These machines are available for general use, but may be subject to job time limits and/or may occasionally be reserved for exclusive use of the entity that purchased them.

The ARC system offers a total of 14,640 CPU cores.

All nodes have the following:

- 2x Intel Platinum 8628 CPU. The Platinum 8628 is a 24 core 2.90GHz Cascade Lake CPU. Thus all nodes have 48 CPU cores per node.
- 384GB memory
- HDR 100 infiniband interconnect. The fabric has a 3:1 blocking factor with non-blocking islands of 44 nodes (2112 cores).
- OS is CentOS Linux 8.1. Scheduler is SLURM.

Login node for the system is 'arc-login.arc.ox.ac.uk', which allows logins from the University network range (including VPN).

The generally available partitions are:

Partition	Nodes / cores	Nodes	Default run time	Maximum run time
short	293 / 14,064	arc-c[001-293]	1 hour	12 hours
medium	242 / 11,616	arc-c[046-287]	12 hours	2 days
long	242 / 11,616	arc-c[046-287]	1 day	unlimited
devel	2 / 96	arc-c[302-303]		10 minutes
interactive	2 / 96	arc-c[304-305]	1 hour	4 hours

2.3 Throughput cluster (htc)

The throughput system - cluster name htc - currently 95 worker nodes, some of which are co-investment hardware. These machines are available for general use, but may be subject to job time limits and/or may occasionally be reserved for exclusive use of the entity that purchased them. The hardware on the HTC system is more heterogeneous than on the ARC system.

49 of the nodes are GPGPU nodes. More information on how to access GPU nodes is available.

2 of the nodes are High Memory nodes with 3TB of RAM.

OS is CentOS Linux 8.1. Scheduler is SLURM.

Login node for the system is 'htc-login.arc.ox.ac.uk', which allows logins from the University network range (including VPN).

Details on the partitions are:

Parti- tion	Nodes / cores, GPUs	Nodes	Default run time	Maximum run time
short	93 / 3,716 <ul style="list-style-type: none"> • 76x V100 • 16x A100 • 24x RTX8000 • 12x RTX A6000 • 20x P100 • 52x Titan RTX 	htc-c[001-046] htc-g[001-006,009-018,020-038,041-052]	1 hour	12 hours
medium	61 / 2,808 <ul style="list-style-type: none"> • 48x V100 • 16x A100 • 24x RTX8000 	htc-c[001-004,006-046] htc-g[009-018,044-049]	12 hours	2 days
long	61 / 2,808 <ul style="list-style-type: none"> • 48x V100 • 16x A100 • 24x RTX8000 	htc-c[001-004,006-046] htc-g[009-018,044-049]	1 day	unlimited
devel	1 / 28 <ul style="list-style-type: none"> • 4x V100 	htc-g039		10 minutes
interac- tive	1 / 28 <ul style="list-style-type: none"> • 4x V100 	htc-g040	1 hour	4 hours

Node CPU details are:

Nodes	CPU	Cores per node	memory per node	inter-connect
htc-c[005-006]	Intel Platinum 8628 (Cascade Lake), 2.90GHz	96	3TB	HDR100
htc-c[007-046]	Intel Platinum 8628 (Cascade Lake), 2.90GHz	48	384GB	
htc-c047	Intel E7-8860v3 (Haswell), 2.60GHz	128	6TB	
htc-g[001-018]	Intel Platinum 8628 (Cascade Lake), 2.90GHz	48	384GB	HDR100
htc-g019	AMD Epyc 7452 (Rome), 2.35GHz	64	1TB	
htc-g[020-029]	Intel Silver 4210 (Cascade Lake), 2.20GHz	20	256GB	
htc-g[030-040]	Intel Gold 5120 (Cascade Lake), 2.20GHz	28	384GB	
htc-g[041-043]	Intel Silver 4112 (Cascade Lake), 2.60GHz	8	192GB	
htc-g[044-049]	Intel E5-2698 v4 (Broadwell), 2.20GHz	40	512GB	
htc-g[050-052]	Intel Silver 4208 (Cascade Lake), 2.10GHz	16	128GB	HDR100

2.4 GPU Resources

ARC has a number of GPU nodes in the “htc” cluster.

Node GPU details are:

Nodes	GPUs	#GPUs	GPU mem-ory	ECC	CUDA cores	CUDA compute capability	nvlink
htc-g[001-008]	V100	2	32GB	yes	5120	7.0	no
htc-g[009-014]	RTX8000	4	40GB	yes	4608	7.5	no
htc-g[015-019]	A100	4	40GB	yes	6912	8.0	no
htc-g[020-029]	Titan RTX	4	24GB	no	4606	7.5	pairwise
htc-g[030-034]	P100	4	16GB	yes	3584	6.0	no
htc-g[035-036]	V100	4	16GB	yes	5120	7.0	no
htc-g[037-038]	V100	4	32GB	yes	5120	7.0	yes
htc-g[039-040]	V100	4	16GB	yes	5120	7.0	yes
htc-g[041-043]	Titan RTX	4	24GB	yes	4606	7.5	pairwise
htc-g044	V100	8	16GB	yes	5120	7.0	yes
htc-g[045-049]	V100-LS	8	32GB	yes	5120	7.0	yes
htc-g[050-052]	RTXA6000	4	48GB	yes	10,752	8.6	yes

2.5 Storage

Our clusters systems share 2PB of high-performance GPFS storage; this holds per-cluster scratch file systems as well as project data storage.

On all nodes with HDR100 interconnect, project data storage is mounted natively; all other nodes access this storage via NFS.

2.6 Software

Users may find the application they are interested in running is already been installed on at least one of the systems. Users are welcome to request the installation of new applications and libraries or updates to already installed applications via our software request form.

CONNECTING TO ARC

Access to ARC systems is via Secure Shell Protocol (SSH).

Fingerprints for the login node host keys can be found [here](#)

3.1 Connecting from Linux

Linux and Mac users should use ssh from a terminal to connect to the ARC systems.

To connect to the ARC cluster:

```
ssh -X username@arc-login.arc.ox.ac.uk
```

To connect to the HTC cluster:

```
ssh -X username@htc-login.arc.ox.ac.uk
```

Access to ARC is available only from within the University of Oxford network. If you are not on the University network, you should use the University VPN service to connect. If you are unable to use the VPN service, you may be able to register your static IP with the ARC team (support@arc.ox.ac.uk) to enable access.

If you are connecting from outside the University network (or from the VPN service) you will need to connect via gateway.arc.ox.ac.uk - ensure you specify your ARC username in the ssh command, as shown below:

```
ssh -X username@gateway.arc.ox.ac.uk
```

You can then SSH to arc-login or htc-login as required.

Note: The above examples include the -X option which allows the tunnelling of X11 graphics from the cluster to your local machine. This is optional. However, if you do require graphics, an X11 server needs to be running on your local machine. For Linux this is typically part of the installation - however on a Mac you may need to install an X11 server such as Xquartz. You can also use the ARC Graphical nodes to render graphics on your local client without using SSH or X11 see the section below on Graphical Nodes for more information.

3.2 Connecting from Windows

SSH is available as a command in PowerShell from Windows 10. Users of Windows 10 or newer should be able to connect to ARC from PowerShell with the commands given above. However, users of older Windows versions, or those requiring X11 forwarding, will need to download and install an application that allows them to connect - a popular example is MobaXterm.

MobaXterm for Windows can be found at <https://mobaxterm.mobatek.net/>

3.3 Connecting using ARC Graphical Nodes

ARC have a number of graphical interactive nodes which you can use to interact with applications which require GUI operation, such as RStudio, Jupyter Notebooks and ANSYS Workbench. This service uses NoMachine NX to provide a remote desktop environment.

In order to use these interactive nodes, you **must** be connected to the university network or be remotely connected via the university VPN service.

3.3.1 Accessing the Graphical nodes via a web browser

You can connect directly via web browser to nx.arc.ox.ac.uk via the web-based client connection (which is lower quality in terms of visual display). See [Configuring NoMachine Web Client](#)

3.3.2 Accessing the Graphical nodes via the client software

You may download the [NoMachine Enterprise Client](#) and install this on your local machine. See [Configuring NoMachine Client](#)

COPYING DATA TO/FROM ARC

MacOS/Linux users (and other Unix like operating systems) can use the command line utilities `rsync`, `sftp`, or `scp` directly to copy data to or from the ARC systems. These commands **must** to be run on your **local** machine as you can only pull files from ARC or push files to it.

Note: \$HOME and \$DATA are shared between the ARC and HTC systems, so the instructions below will work for both systems.

The gateway machine `gateway.arc.ox.ac.uk` has no access to ARC \$HOME, but the path to your ARC \$DATA is available. Do not transfer data into the home directory on `gateway.arc.ox.ac.uk` please use your \$DATA path instead.

4.1 Copying to the ARC systems

- 1) Make sure you know the full path to the destination directory on ARC - The best way to to this is to log in to ARC, change to that directory and run the command `pwd`; this will show you the full path to the directory.

For example if you have created a directory `myscripts` in \$DATA, `pwd` will show output like:

```
[ouit0578@gateway]$ cd $DATA/myscripts
[ouit0578@gateway]$ pwd
/data/system/ouit0578/myscripts
```

- 2) On your local Mac or Linux PC open a terminal window and change directory to the directory where the file(s) you want to copy **to** ARC are located. The command `pwd` will show the full path to this directory, take note of this path. For example if you have a directory named `scripts` in your local home directory, the path may look something like:

```
$ cd scripts
$ pwd
/home/user/scripts
```

- 3) While still working on your MAC or Linux PC: to copy files, use the `scp` command with the format:

```
scp <source> <destination>

scp local/path/filename arcuserid@gateway.arc.ox.ac.uk:/path_to_destination_
↪directory/
```

Note: The **remote** side of the copy requires you to specify your ARC username followed by the @ symbol, then the name of the machine you want to connect to - in this case `gateway.arc.ox.ac.uk` followed by a : (colon) then the remote directory path.

Continuing with the above example - to copy all the files in `/local/scripts` this would be:

```
scp /home/user/scripts/* ouit0578@gateway.arc.ox.ac.uk:/data/system/ouit0578/myscripts/
```

To copy an entire directory hierarchy use the recurse option, `-r`

For example:

```
scp -r /home/user/scripts/* ouit0578@gateway.arc.ox.ac.uk:/data/system/ouit0578/
↪myscripts/
```

In both of the above cases, you will be prompted to authenticate with your ARC password.

4.2 Copying from the ARC systems

The reverse should be used for copying from arc. Using the methods above, take note of the source (ARC) and destination (local PC) paths. In this example we will use:

ARC directory: `/data/system/ouit0578/mydata` Local PC directory: `/home/user/datafromarc`

4.3 Copying data from Mac or Linux PC

Once again, we must run the `scp` command on our local machine as we are going to **pull** the data back from ARC.

To copy files from your local PC:

```
scp ouit0578@gateway.arc.ox.ac.uk:/data/system/ouit0578/mydata/* /home/user/datafromarc/
```

Again a recursive copy can be made using the `-r` option:

```
scp -r ouit0578@gateway.arc.ox.ac.uk:/data/system/ouit0578/mydata/* /home/user/
↪datafromarc/
```

In both the above cases, you will be prompted to authenticate with your ARC password.

4.4 Copying to/from ARC using Graphical File Transfer utilities

Windows users can use tools such as MobaXterm or WinSCP to copy files. In this example we will use the popular **WinSCP** utility, but the method used should translate well to other utilities - consult their documentation for information.

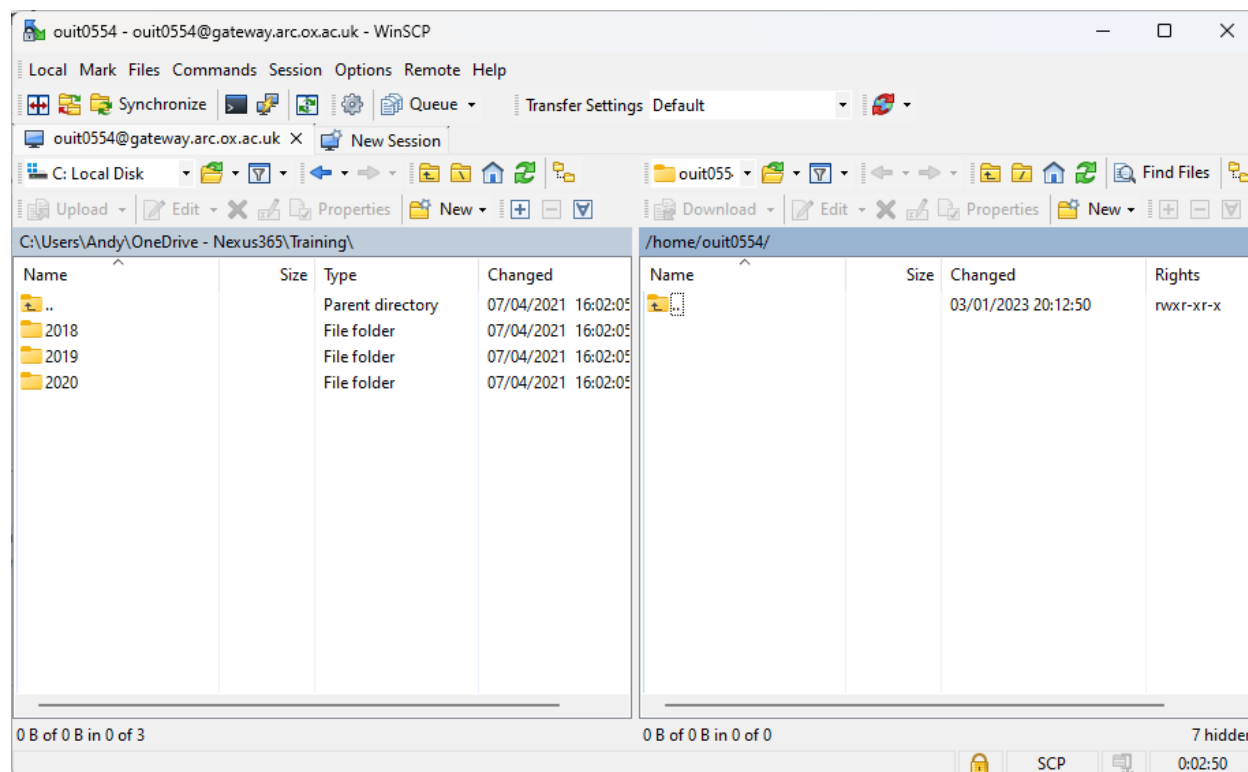
If you have not already, use the discovery method in step 1) at the top of this page above to note the remote ARC path for the transfer - as you will require this information later.

First open **WinSCP** and complete the Session fields as follows:

File Protocol: SCP **Host Name:** gateway.arc.ox.ac.uk **Port Number:** 22 **User Name:** Your ARC username
Password: Your ARC password

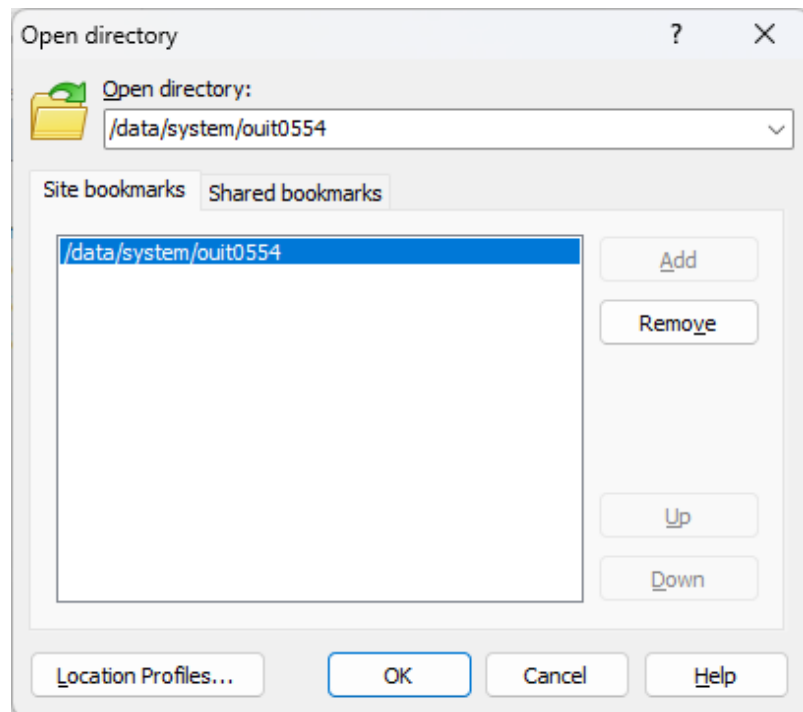
The screenshot shows a 'Login' dialog box with a light gray background. On the left is a large empty rectangular area with a 'New Site' button at the top left. To the right of this area is a 'Session' configuration panel. This panel contains a 'File protocol:' dropdown menu set to 'SCP'. Below it are two input fields: 'Host name:' containing 'gateway.arc.ox.ac.uk' and 'Port number:' containing '22'. Further down are 'User name:' and 'Password:' fields; the user name is 'ouit0554' and the password is masked with dots. At the bottom of the session panel are 'Save' and 'Advanced...' buttons. Below the session panel, there are 'Tools' and 'Manage' dropdown menus. At the very bottom of the dialog are four buttons: 'Login' (with a green plus icon), 'Close', and 'Help'. A checkbox at the bottom left is checked and labeled 'Show Login dialog on startup and when the last session is closed'.

If you wish, you can click Save to save this information for future sessions, otherwise click Login to connect...
You should then be logged in and see the following type of display...



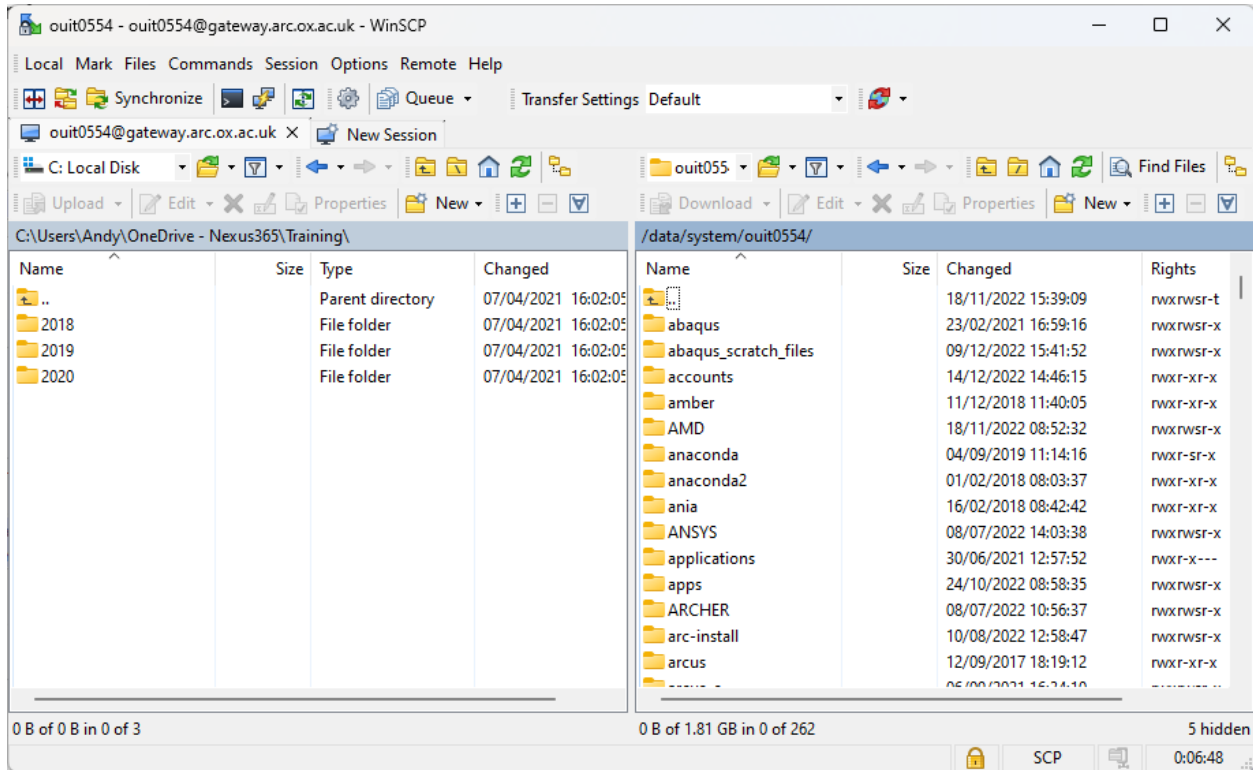
The left “pane” is the local filesystem on your machine, and the right “pane” is the remote ARC filesystem. This defaults to your home directory on `gateway.arc.ox.ac.uk` which is **not** the same place as `$HOME` on ARC - so please do not transfer files here.

Use the WinSCP menu options, **Remote | Go To | Open Directory/Bookmark...** to bring up the following dialogue box:



In the **Open Directory** selector, type in the path to your \$DATA area, as found in Step 1 at the top of this page. In my case this is /data/system/ouit0554

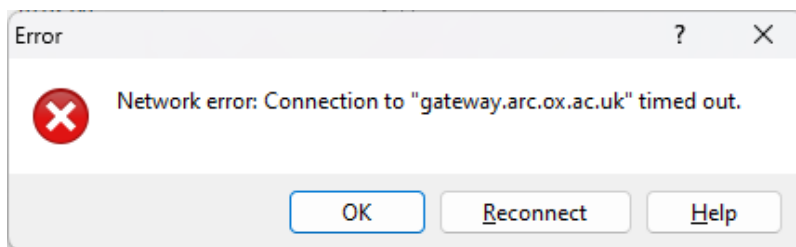
You can click Add to save this as a bookmark for next time, or simply click OK to open this directory on ARC.



You should now see your \$DATA area on the right pane and you can drag/drop files between your local and ARC filesystems.

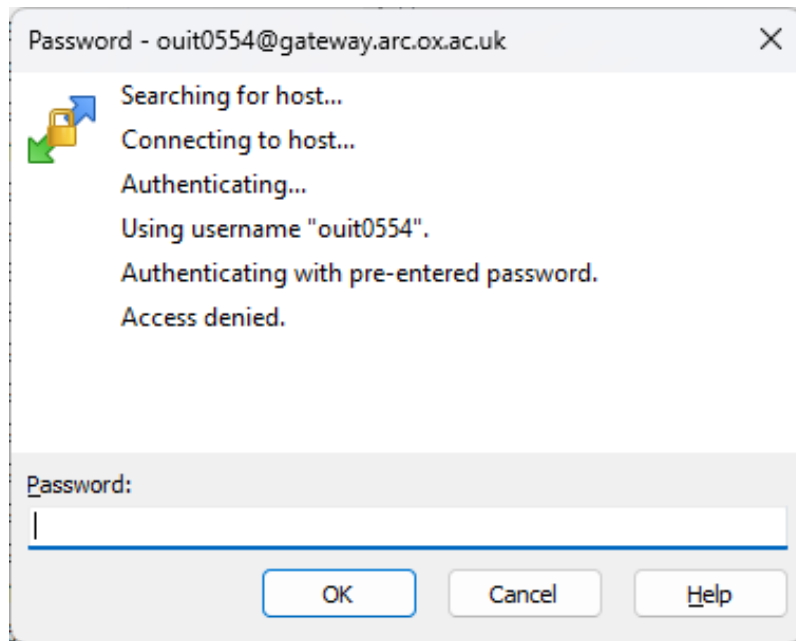
4.5 Troubleshooting Graphical File Transfers

If you see an error of this type:



This indicates a network problem between your local machine and the ARC service. Typically this is caused by being off-campus **without** a working university VPN connection. Try restarting the VPN client.

If you supply the wrong username or password to the file transfer utility, you may see errors such as the following:



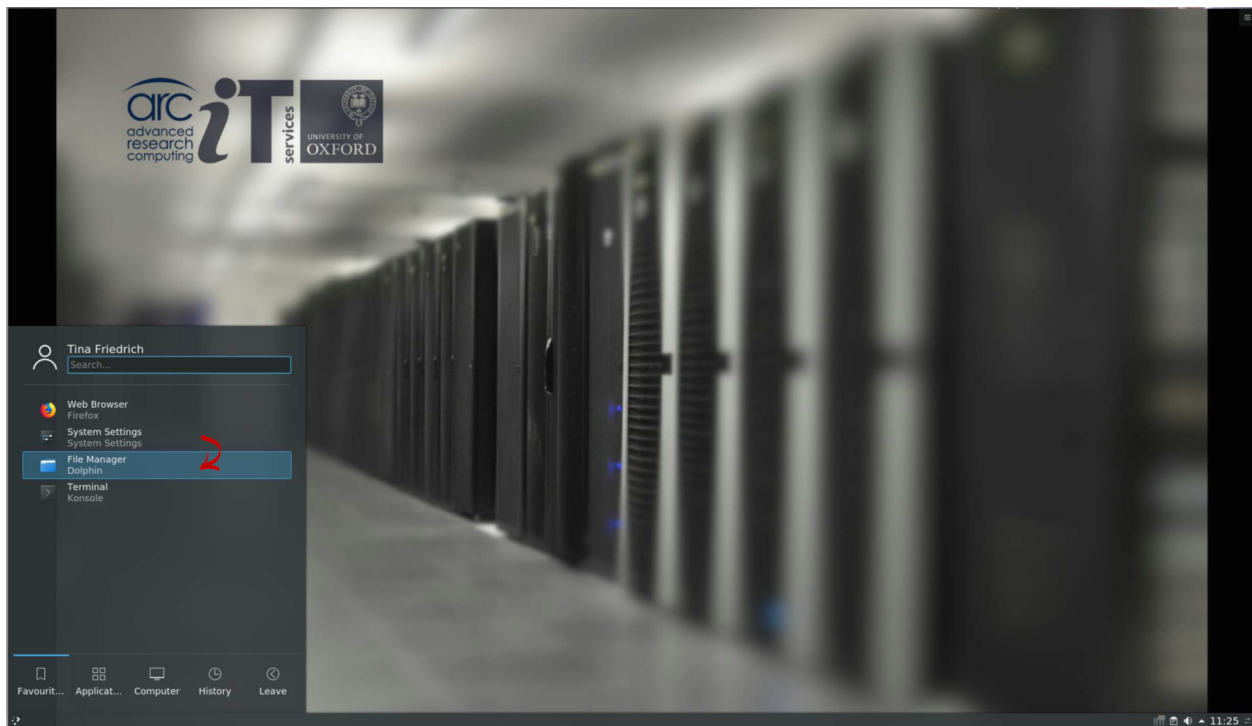
Access Denied commonly means that you have made a connection to ARC, but you have supplied the wrong user-name/password combination. Try checking these using a standard SSH connection, and if the problem persists contact support@arc.ox.ac.uk for assistance.

4.5.1 Copying data from/to the RFS

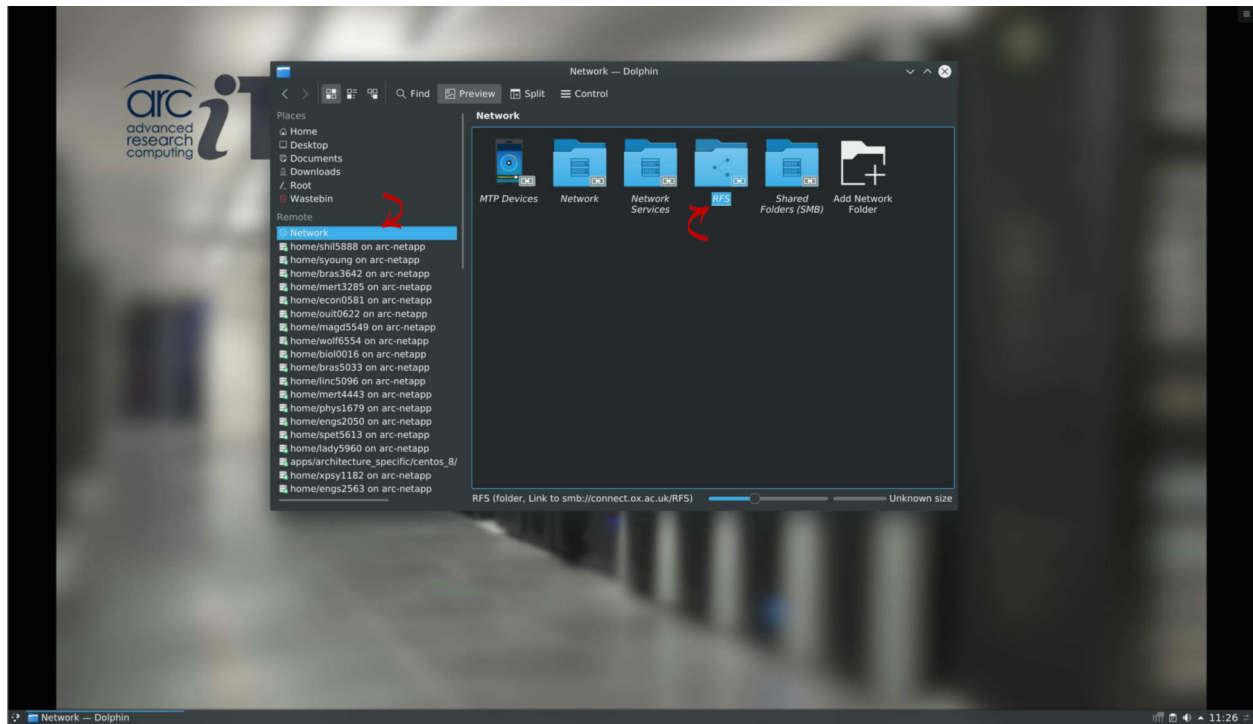
It is possible to access the Research File Service (RFS) from ARC. There are two main ways it can be accessed; via a graphical file browser from the [ARC desktop environment](#), or command line (e.g. from login nodes or cluster nodes).

ACCESSING RFS FROM THE ARC DESKTOP ENVIRONMENT

Log in to the ARC desktop environment. Open the main menu, and start the file manager application (Dolphin).



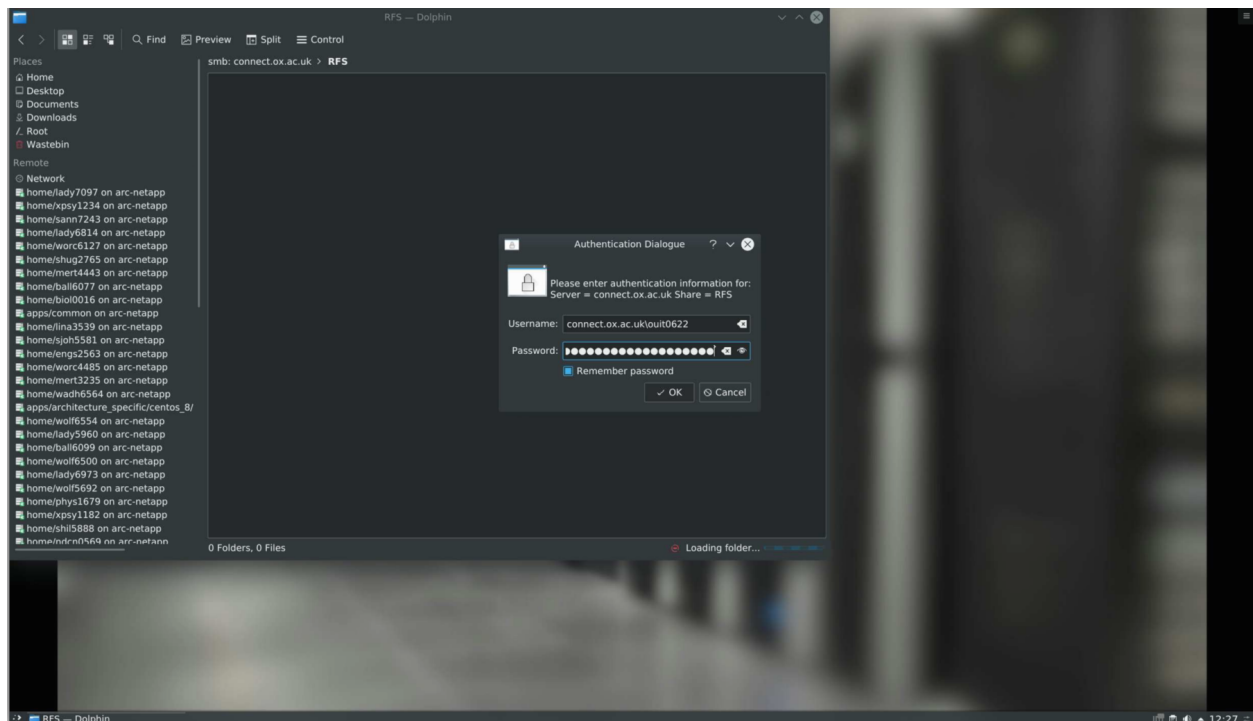
In Dolphin, select 'Network' from the left menu. Click on the folder named 'RFS'.



KDE should ask you for your username and password. This is your **CONNECT** account; you need to enter the username as:

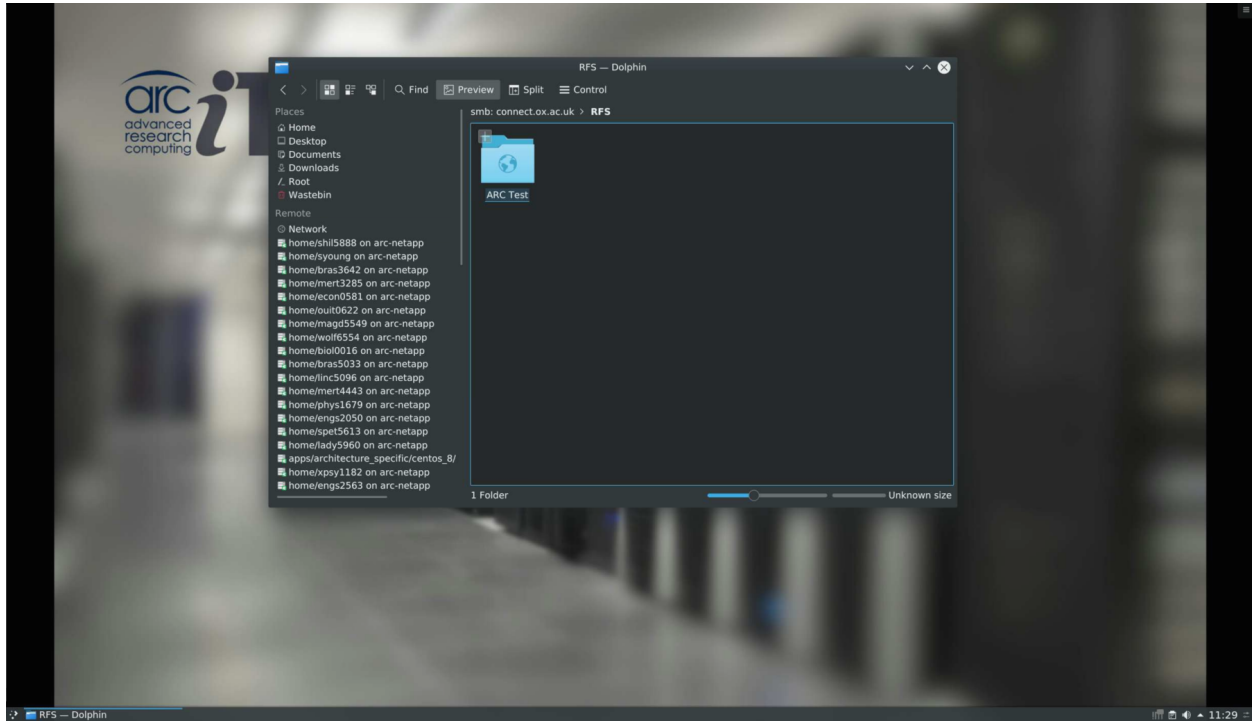
```
connect.ox.ac.uk\USERID
```

your user ID is likely the same as your ARC (and SSO) ID. Enter your **CONNECT** account password.



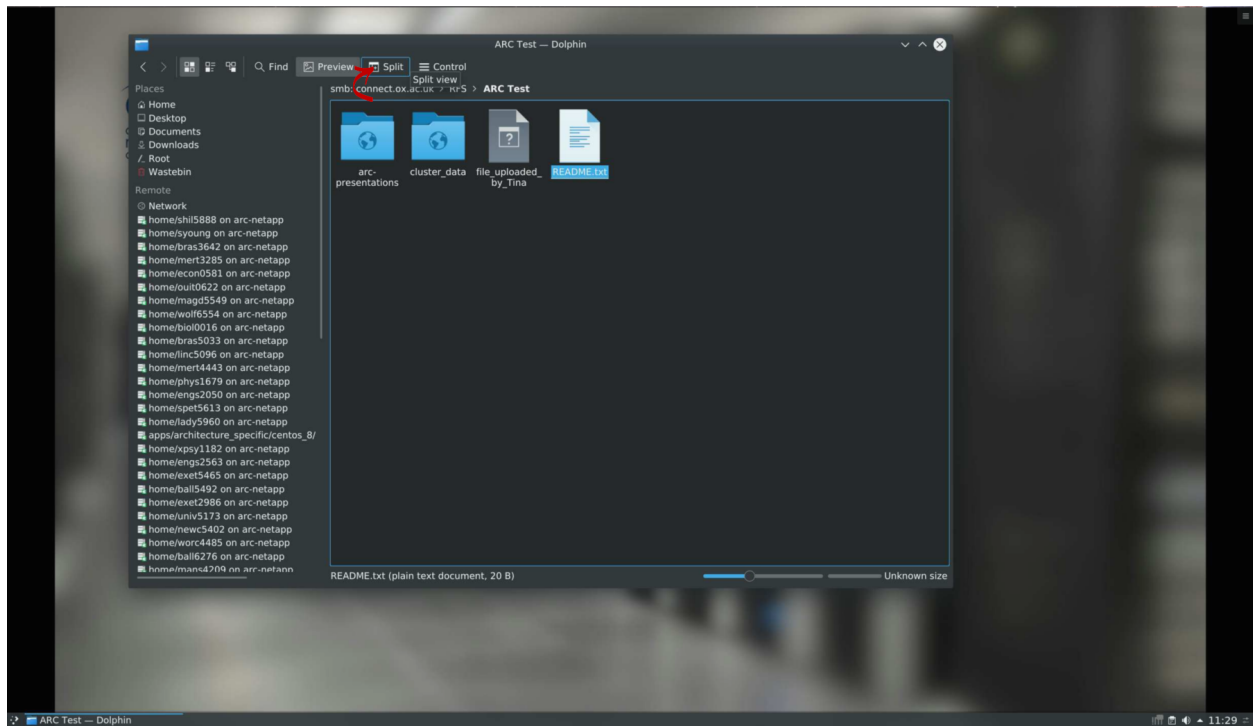
If you select 'Remember password', KDE will save these details for you. It uses its KDE Wallet service to save these

securely. If you have not used that before, it will guide you through the setup and let you set a master password for your wallet. Once you've entered your CONNECT credentials and hit 'enter' (or clicked 'OK'), it should connect you to your RFS share.

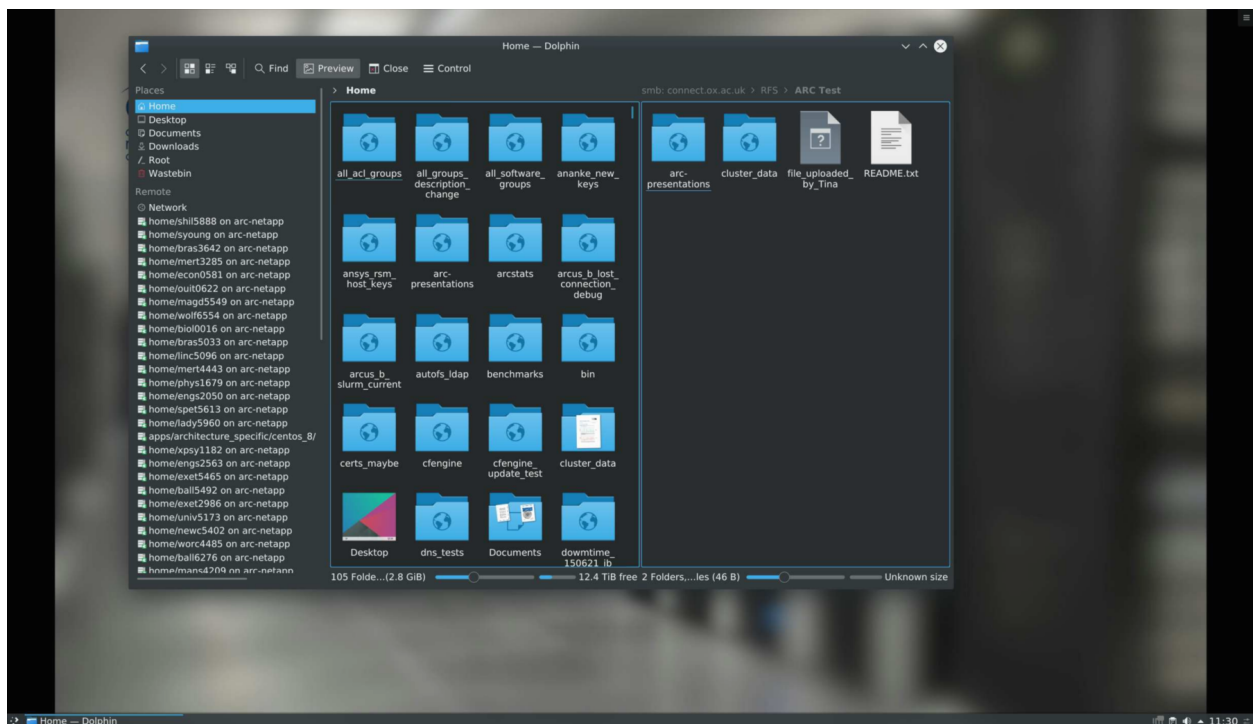


You can interact with this from Dolphin as with a normal folder, and navigate the folder structure on RFS. You can copy files or directories in and out of the service (e.g. via drag-and-drop), and open them in applications.

Dolphin has a 'split' view that allows having multiple places (folders) open at the same time, which is useful for transferring files in and out of RFS. To split the file browser window, click on the 'Split' button in the menu bar:



You can then select one half of the split window and change what is showing to e.g. your Home directory, or navigate to your data folder:



ACCESSING RFS FROM THE COMMAND LINE

6.1 Using the ARC rfs tool

We provide a command-line tool - called 'rfs' - that allows easy interaction with RFS. It can list files on RFS; create directories on RFS; push data to RFS; and fetch data from RFS.

```
[ouit0622@arc-login03 ~]$ rfs -h
Usage: /home/ouit0622/Scripts/tools/rfs [-c credentials file] (list|mkdir|push|fetch) REMOTE_FILE [LOCAL_FILE]...
'list' can take one argument - remote subdirectory to list (e.g. rfs list "MYDIR/MYSUBDIR")
'mkdir' can take two arguments - a directory tree to create, and a remote directory to create it in
      (e.g. rfs mkdir "NEWDIR/NEWSUBDIR" "MYDIR/MYSUBDIR")
      if only one is given, remote directory is assumed to be the RFS root directory
'push' can take two arguments - local file or directory to push, and remote directory to put it into
      (e.g. rfs push "NEWDATA" "MYDIR/MYSUBDIR")
      if only one is given, remote directory is assumed to be the RFS root directory
'fetch' can take two arguments - remote directory to fetch, and local directory to put it into
      (e.g. rfs fetch "MYDIR/MYSUBDIR" "/scratch/")
      if only one is given, local directory is assumed to be the current working directory
Note - pushing (or fetching) a directory will push/fetch the directory and everything in it.
It will create the directory in RFS (or locally) as well.
So e.g. pushing directory 'todays_data' into remote directory MYDATA will create directory
'todays_data' as a subdirectory of MYDATA.
Credential file should be mode '0600' (chmod 0600 FILENAME to set), and need to contain
username = ouit0622
password = your_connect_password
domain = connect.ox.ac.uk
[ouit0622@arc-login03 ~]$
```

It can list directory contents of both the top level directory, or within directories:

```

[ouit0622@arc-login03 ~]$ rfs -c $HOME/connect list
listing files in RFS
.                D            0  Mon Sep 18 17:03:03 2023
..               D            0  Fri Oct  7 10:52:14 2022
ARC Test         Dr            0  Thu May  4 16:18:12 2023

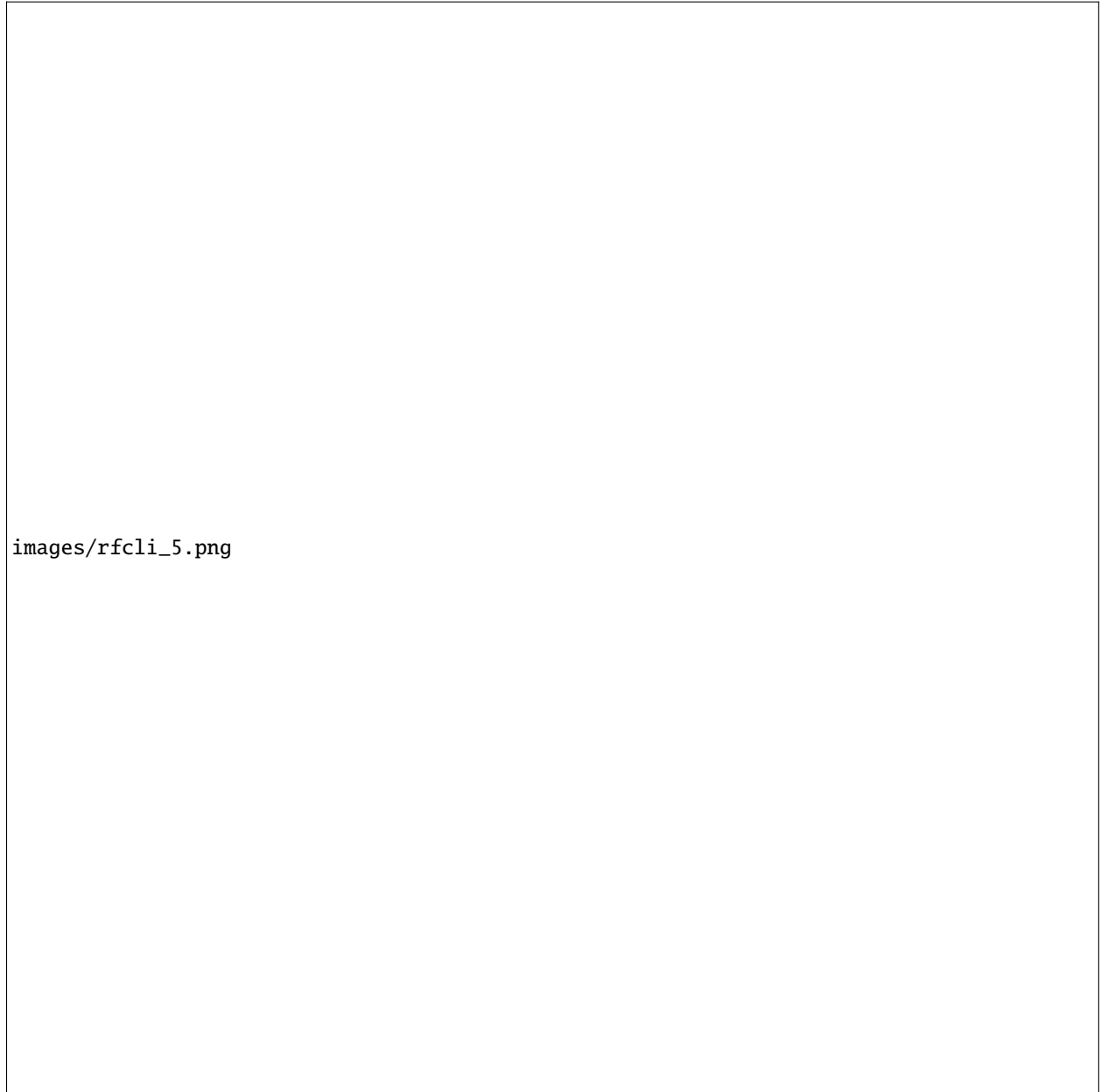
116914431 blocks of size 4096. 105597435 blocks available
[ouit0622@arc-login03 ~]$ rfs -c $HOME/connect list "ARC Test/"
listing files in RFS
.                D            0  Tue Sep 19 16:49:47 2023
..               D            0  Mon Sep 18 16:59:38 2023
arc-presentations D            0  Tue Sep  5 09:57:46 2023
cluster_data      D            0  Wed Sep  6 13:58:17 2023
osu_benchmarks    D            0  Tue Sep 12 11:33:42 2023
README.txt        A           20  Fri May  5 13:16:31 2023

116914431 blocks of size 4096. 105597435 blocks available
[ouit0622@arc-login03 ~]$ rfs -c $HOME/connect list "ARC Test/osu_benchmarks/"
listing files in RFS
.                D            0  Tue Sep 12 11:33:42 2023
..               D            0  Tue Sep 19 16:49:47 2023
run_osu_benchmarks.sh A       1237  Tue Sep 12 11:33:41 2023
slurm-670314.out  A         818  Tue Sep 12 11:33:41 2023
slurm-670315.out  A         818  Tue Sep 12 11:33:41 2023
slurm-670320.out  A       1462  Tue Sep 12 11:33:42 2023
slurm-670322.out  A       1793  Tue Sep 12 11:33:41 2023
slurm-670325.out  A       1793  Tue Sep 12 11:33:41 2023
slurm-670378.out  A       1745  Tue Sep 12 11:33:41 2023
slurm-671355.out  A      4016  Tue Sep 12 11:33:41 2023
slurm-671359.out  A      2356  Tue Sep 12 11:33:42 2023
without_proper_ucx D            0  Tue Sep 12 11:33:42 2023

116914431 blocks of size 4096. 105597428 blocks available
[ouit0622@arc-login03 ~]$ █

```

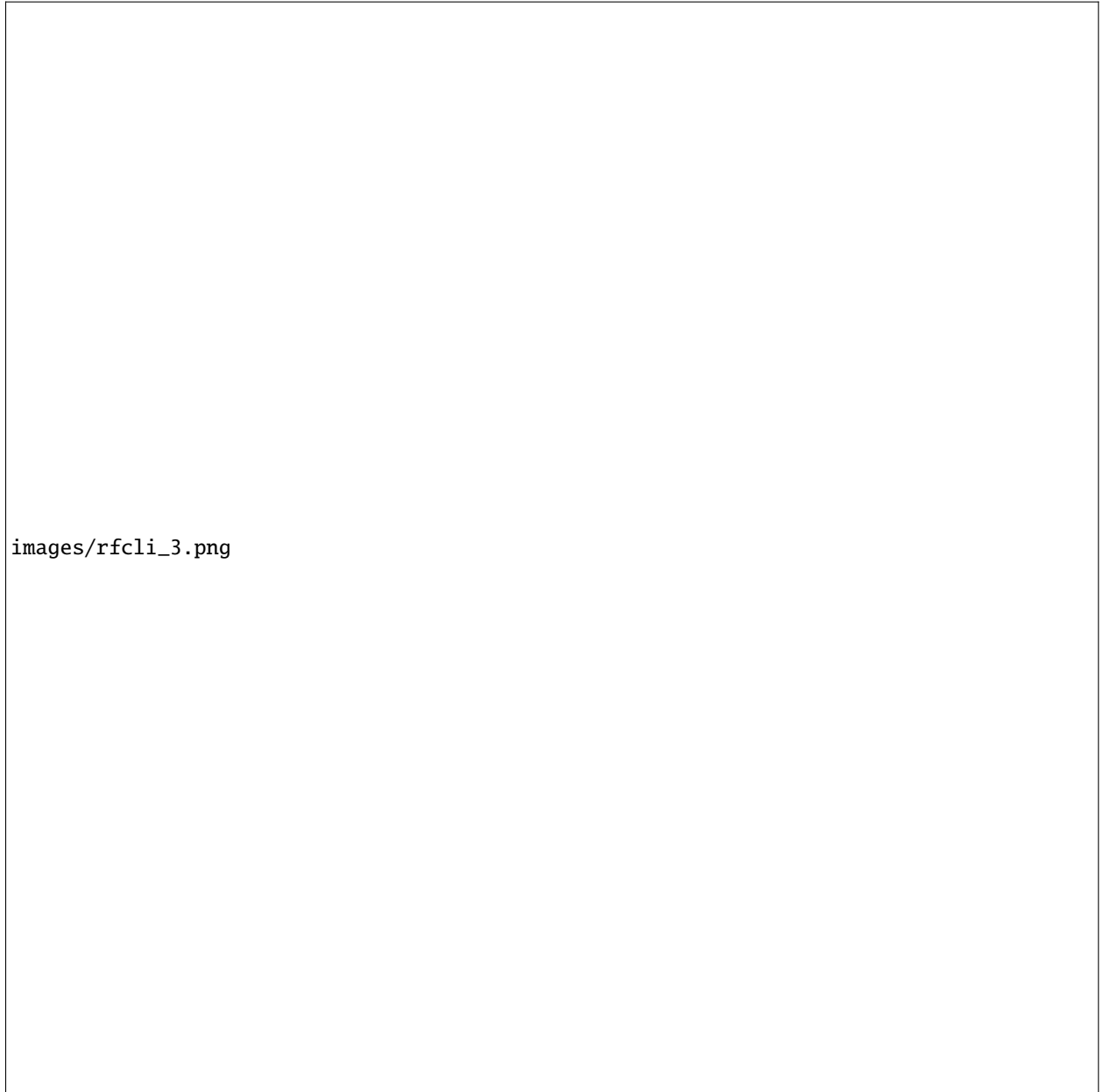
It allows to create directories on RFS, and pushing data up (both files and directories):



images/rfcli_5.png

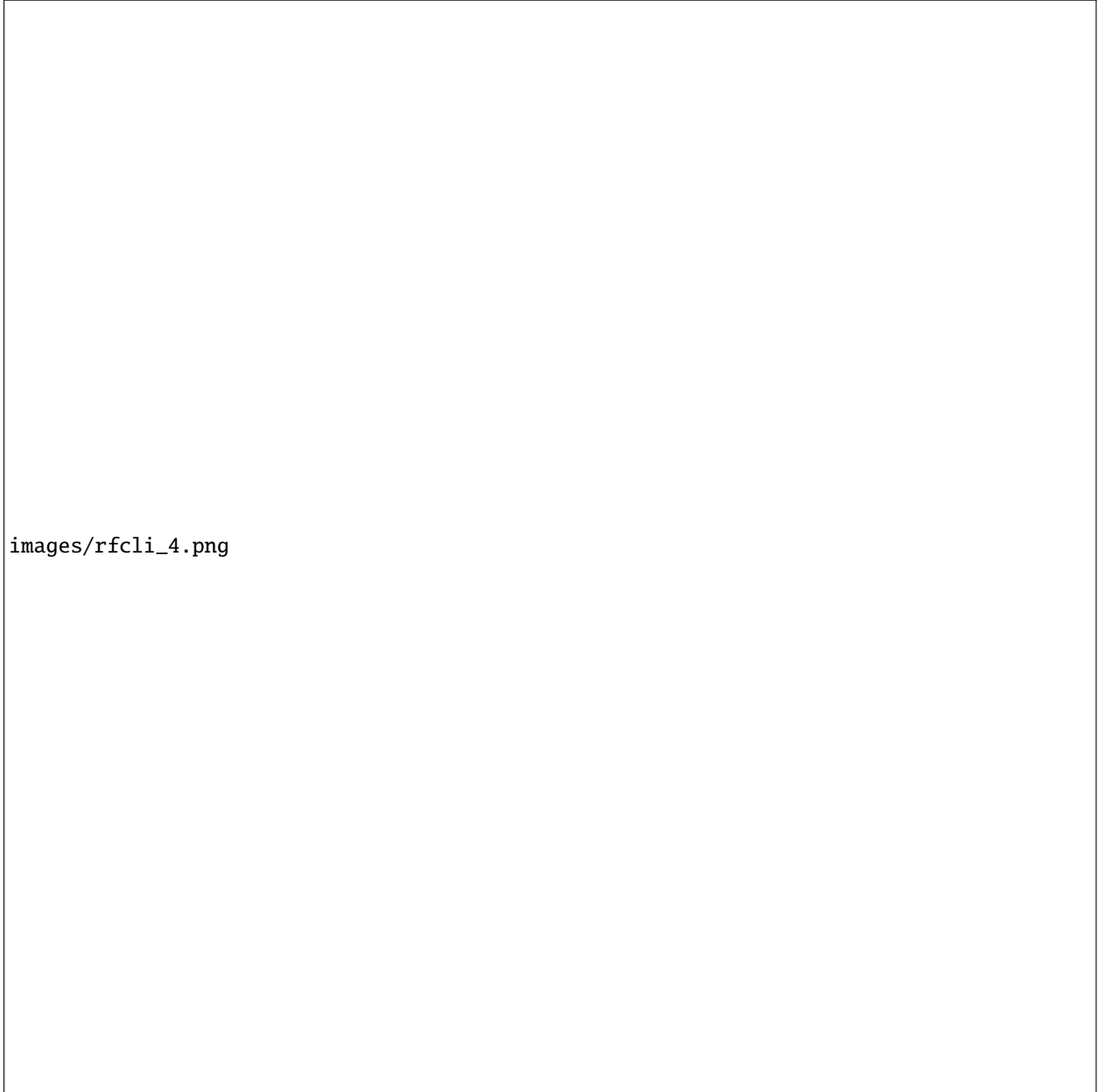
The ‘push’ command will work on both files and directories (directories are pushed recursively).

Files and directories can also be fetched from RFS, into either the current directory:



images/rfcli_3.png

or into a specified directory:



images/rfcli_4.png

Paths entered can be either relative (i.e. relative to the current working directory), or full paths.

6.2 Using smbclient

Ultimately, the tool to interact with RFS from the command line is smbclient. This offers many more ways to interact with RFS than the ARC provided 'rfs' tool (including deleting files).

To start a smbclient session, type

```
smbclient -U connect.ox.ac.uk\\$USER //connect.ox.ac.uk/RFS
```

and enter your connect password. This will start a session. Within smbclient, the 'help' command will give you the list of commands, and help with commands:

```
[ouit0622@arc-login03 ~]$ smbclient -U connect.ox.ac.uk\\$USER //connect.ox.ac.uk/RFS
Password for [CONNECT.OX.AC.UK\ouit0622]:
Try "help" to get a list of possible commands.
smb: \> ls
.                D           0   Mon Sep 18 17:03:03 2023
..               D           0   Fri Oct  7 10:52:14 2022
ARC Test         Dr          0   Thu May  4 16:18:12 2023

116914431 blocks of size 4096. 105701829 blocks available
smb: \> help
?
allinfo          altname          archive          backup
blocksize        cancel           case_sensitive  cd               chmod
chown            close           del             deltree         dir
du               echo            exit            get             getfacl
geteas           hardlink        help            history          iosize
lcd              link            lock            lowercase       ls
l                mask            md              mget            mkdir
more             mput            newer           notify           open
posix            posix_encrypt   posix_open      posix_mkdir     posix_rmdir
posix_unlink     posix_whoami    print           prompt          put
pwd              q               queue           quit            readlink
rd               recurse         reget           rename          reput
rm               rmdir           showacls        setea           setmode
scopy            stat            symlink         tar             tarmode
timeout          translate       unlock          volume          vuid
wdel             logon           listconnect     showconnect     tcon
tdis             tid             utimes          logoff          ..
!
smb: \> help get
HELP get:
    <remote name> [local name] get a file

smb: \> help put
HELP put:
    <local name> [remote name] put a file

smb: \> quit
[ouit0622@arc-login03 ~]$
```

An example session - looking at available files and removing a directory (and it's contents) - would be

```
[ouit0622@arc-login03 ~]$ smbclient -A $HOME/connect //connect.ox.ac.uk/RFS
Try "help" to get a list of possible commands.
smb: \> ls
.
```

.	D	0	Mon Sep 18 17:03:04 2023
..	D	0	Tue Nov 8 15:02:57 2022
ARC Test	Dr	0	Thu May 4 16:18:12 2023

```
15567359 blocks of size 4096. 7115425 blocks available
smb: \> cd "ARC Test"
smb: \ARC Test\> ls
.
```

.	D	0	Wed Sep 20 08:48:19 2023
..	D	0	Mon Sep 18 16:59:38 2023
arc-presentations	D	0	Tue Sep 5 09:57:46 2023
cluster_data	D	0	Wed Sep 6 13:58:17 2023
ior	D	0	Wed Sep 20 08:48:19 2023
kde_config_files	D	0	Wed Sep 20 08:26:33 2023
ldif_backup_slapcat_hecate_20230227.gpg	A	1027569	Tue Sep 19 17:15:12 2023
osu_benchmarks	D	0	Tue Sep 12 11:33:42 2023
README.txt	A	20	Fri May 5 13:16:31 2023
tina_test	D	0	Tue Sep 19 17:40:09 2023
zone_data	D	0	Tue Sep 19 17:03:54 2023

```
268430847 blocks of size 4096. 265231665 blocks available
smb: \ARC Test\> cd kde_config_files
smb: \ARC Test\kde_config_files\> ls
.
```

.	D	0	Wed Sep 20 08:26:33 2023
..	D	0	Wed Sep 20 08:48:19 2023
ARC.directory	A	111	Wed Sep 20 08:26:32 2023
ARC.menu	A	425	Wed Sep 20 08:26:33 2023
kdeglobals	A	380	Wed Sep 20 08:26:32 2023
ksscreenlockerrc	A	150	Wed Sep 20 08:26:32 2023
kssmserverrc	A	81	Wed Sep 20 08:26:33 2023
plasmarc	A	23	Wed Sep 20 08:26:32 2023
rstudio.desktop	A	520	Wed Sep 20 08:26:32 2023

```
268430847 blocks of size 4096. 265231665 blocks available
smb: \ARC Test\kde_config_files\> rm *
smb: \ARC Test\kde_config_files\> ls
.
```

.	D	0	Wed Sep 20 08:57:16 2023
..	D	0	Wed Sep 20 08:48:19 2023

```
268430847 blocks of size 4096. 265231665 blocks available
smb: \ARC Test\kde_config_files\> cd ../
smb: \ARC Test\> rmdir kde_config_files\
smb: \ARC Test\> ls
.
```

.	D	0	Wed Sep 20 08:57:21 2023
..	D	0	Mon Sep 18 16:59:38 2023
arc-presentations	D	0	Tue Sep 5 09:57:46 2023
cluster_data	D	0	Wed Sep 6 13:58:17 2023
ior	D	0	Wed Sep 20 08:48:19 2023
ldif_backup_slapcat_hecate_20230227.gpg	A	1027569	Tue Sep 19 17:15:12 2023
osu_benchmarks	D	0	Tue Sep 12 11:33:42 2023
README.txt	A	20	Fri May 5 13:16:31 2023
tina_test	D	0	Tue Sep 19 17:40:09 2023
zone_data	D	0	Tue Sep 19 17:03:54 2023

```
268430847 blocks of size 4096. 265231666 blocks available
[ouit0622@arc-login03 ~]$
```

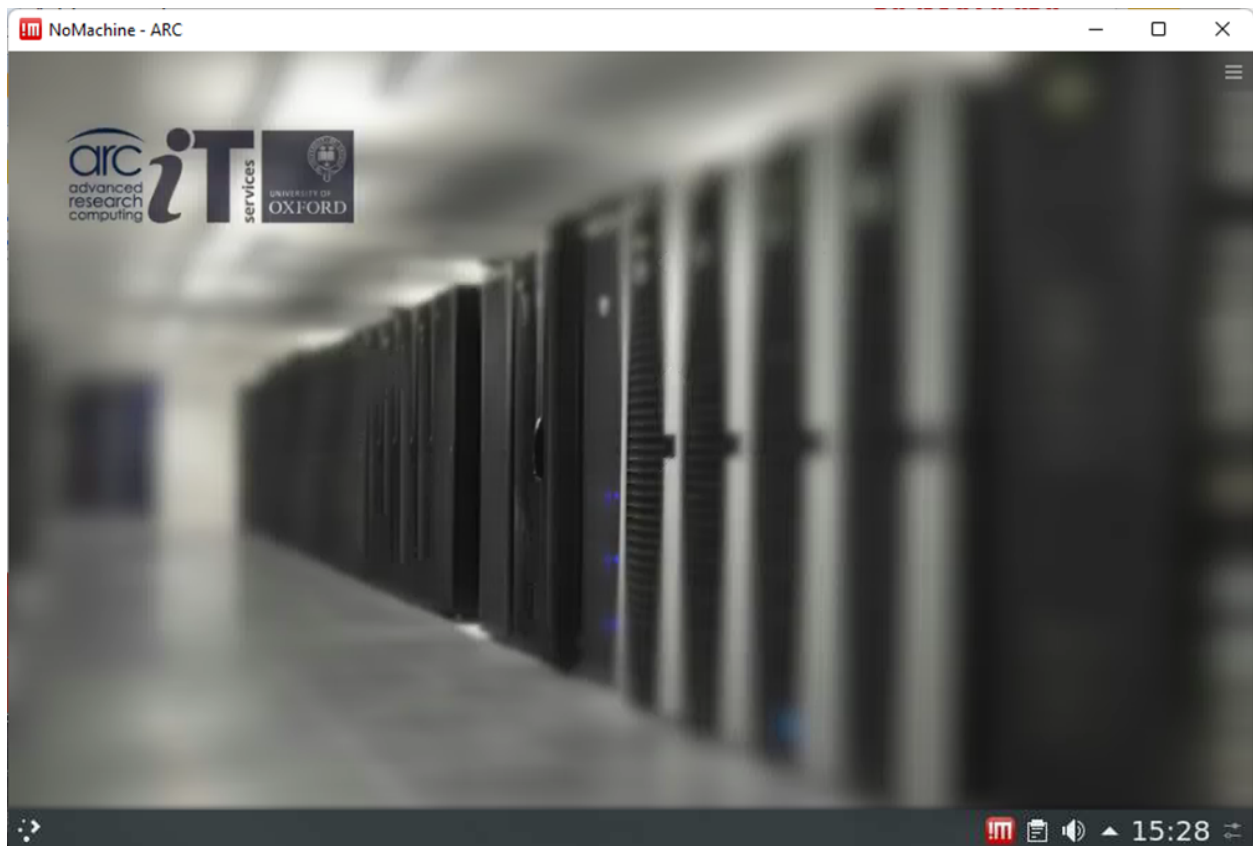

USING THE ARC DESKTOP ENVIRONMENT

ARC provide a graphical login service, to facilitate access to GUI applications such as RStudio, Jupyter Notebooks, and ANSYS Workbench. This service uses NoMachine NX to provide a remote desktop.

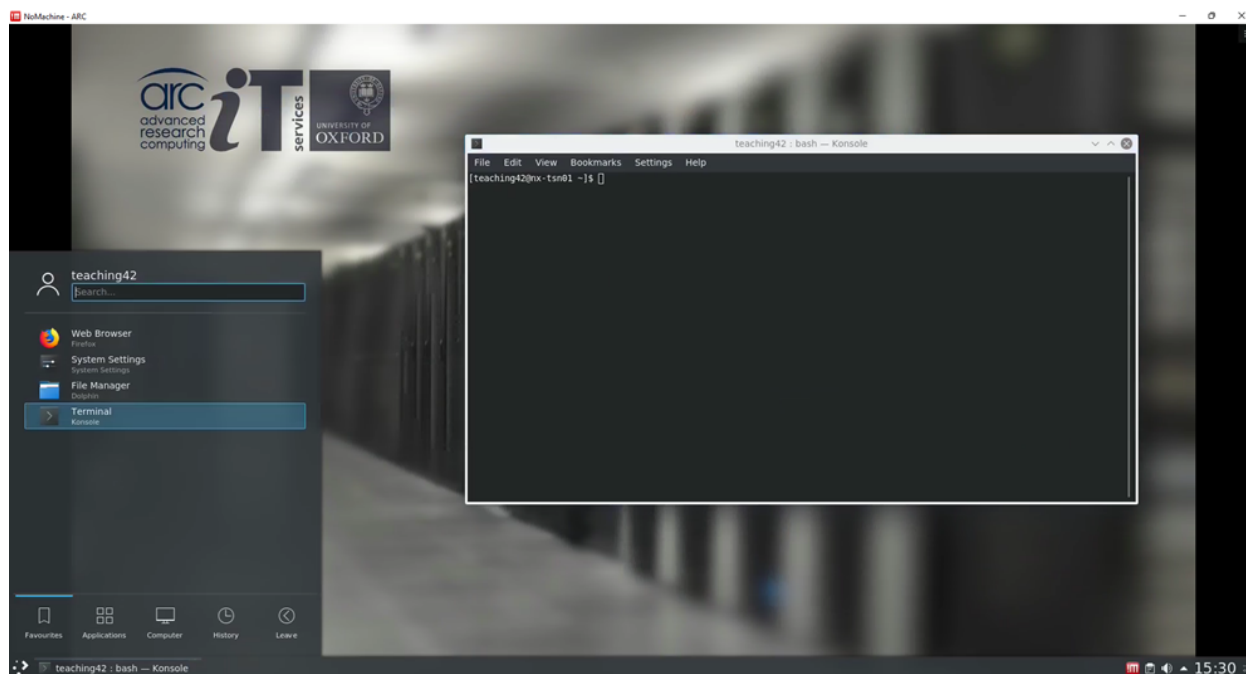
In order to use this service, you must be connected to the university network or be remotely connected via the university VPN service. See [Connecting to ARC](#) for details how to connect to the graphical login nodes.

NoMachine NX allows you to detach from a session, and reconnect to it later - simply close your browser or NX client windows. Processes on the NX nodes will continue to run once you've detached. You can reconnect to a session later; your running sessions will be presented to you when you reconnect. However, exiting a session (or logging out of the desktop environment) will terminate the session, terminating all processes running within it.

Once you have connected to the NX service by one of the described methods, you will be presented with the ARC desktop environment as shown below:



You can use the menu bar at the bottom of the window to access applications or, as in the example below, open the Konsole terminal window:



It is possible to access the clusters from the ARC desktop environment. Open a terminal as described above, and run:

```
module load cluster/arc
```

to access the 'ARC' cluster or:

```
module load cluster/htc
```

for access to the 'HTC' cluster.

The graphical login nodes provide fast access to the data file system; they can (and should) be used for pre- or post-processing.

Note: While it may look like you have your own Linux desktop to work with, the interactive nodes where you are running this desktop are shared with other ARC users, and should not be used to run computationally demanding jobs.

To run applications that are more demanding, please open a Konsole shell window and start an interactive X11 session on a compute node by following the instructions below:

To run an interactive session on the ARC cluster:

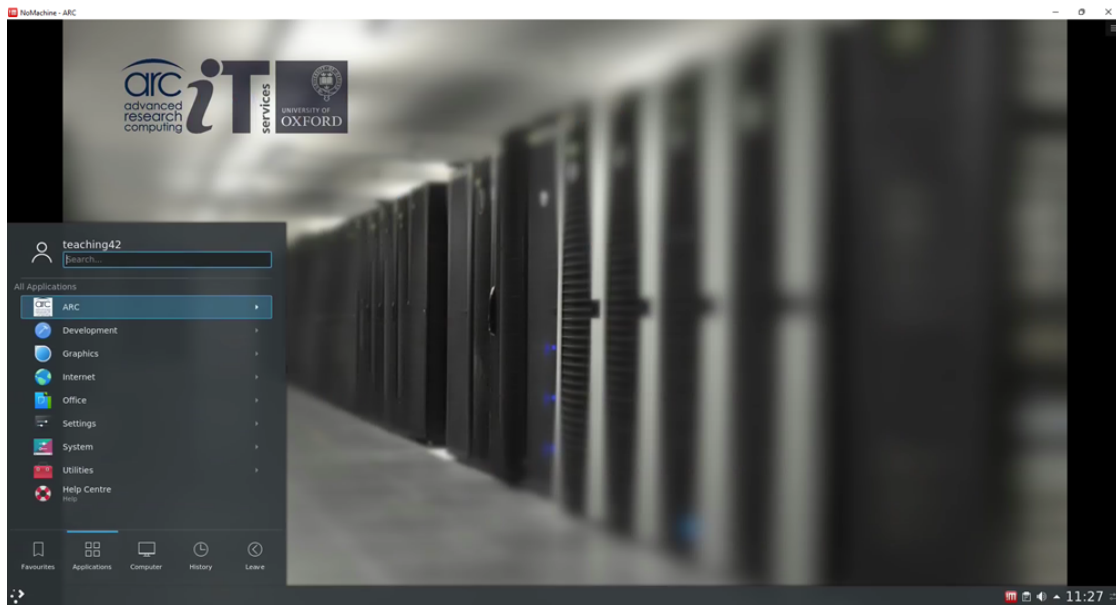
```
module load cluster/arc
srun -p interactive --x11 --pty /bin/bash
```

To run an interactive session on the HTC system:

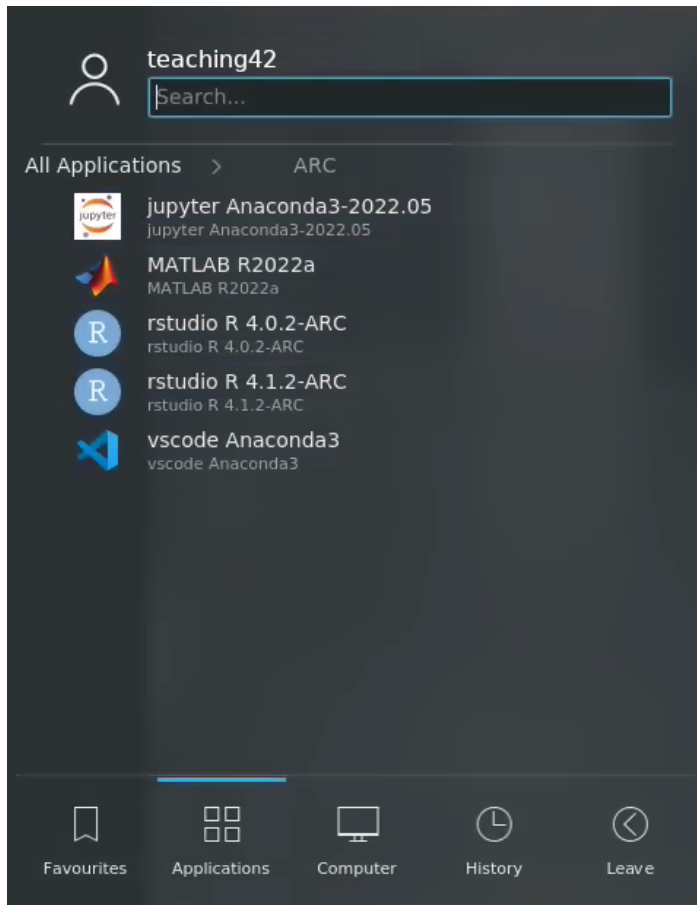
```
module load cluster/htc
srun -p interactive --x11 --pty /bin/bash
```

7.1 Running applications from the desktop

There are a number of predefined applications which you can find under the ARC sub-menu, within the Applications section of the main menu bar, see below for examples:



Applications|ARC Sub-menu



Clicking on these menu items will start up the appropriate version of the specified application, with any other required modules automatically loaded.

Note: If you need customised versions of applications or supporting modules (such as custom Anaconda virtual environments) you should load/activate and run these as appropriate from the Konsole command window, in the same way as the ARC or HTC systems.

SUBMISSION SCRIPT BASICS

What is a submission script

A `submission` script describes the resources you need the resource manager (SLURM) to allocate to your job. It also contains the commands needed to execute the application(s) you wish to run, including any set-up the application(s) may require.

The script needs to be created using a Linux text editor such as `nano` or `vi` - we recommend creating and editing submission scripts on the cluster rather than editing them on a Windows machine, as this can cause problems.

Simple submission script example

In this example we are going to create a submission script to run a test application on the cluster.

Note: If you are not familiar with typing commands into a Linux shell (or terminal window) we suggest you follow an online Linux tutorial such as the Linux tutorial at [Ryans Tutorials](#)

To begin with we change directory to your `$DATA` area and make a new directory named `example` to work in:

```
cd $DATA
mkdir example
cd example
```

The next step is to create the submission script to describe your job to SLURM.

First we use the `nano` editor to create the file:

```
nano submit.sh
```

This command will start the Linux `nano` editor. You can use `nano` to add the following lines:

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --time=00:10:00
#SBATCH --partition=devel

module load mpitest

mpirun mpihello
```

Note: Ensure the `#!/bin/bash` line is the first line of the script and the `#SBATCH` lines start at the beginning of the

line.

To exit nano hold CTRL-X then answer Y to the question Save Modified buffer? then hit enter when asked File Name to Write: submit.sh

The anatomy of the script

The first line `#!/bin/bash` tells Linux that this file is a script which can be run by the BASH shell interpreter.

The following `#SBATCH` lines request specific cluster resources:

`--nodes=2` requests two ARC nodes

`--ntasks-per-node=4` requests 4 cores per node (a total of 8)

`--time=00:10:00` requests a run time of 10 minutes (the maximum for the devel partition)

`--partition=devel` requests that this job runs on the devel partition, which is reserved for testing

`module load mpihello` The **module load** command is used to make an application environment available to use in your job, in this case the **mpitest** application.

`mpirun mpihello` This line runs the **mpihello** command using the special **mpirun** wrapper.

Note: MPI is required for multi-process operation across nodes, and may not be appropriate for all applications.

Submitting the job

Now that you have a submission script, you can submit it to the SLURM resource manager. To do this type the following at the command line:

```
sbatch submit.sh
```

SLURM will respond with:

```
sbatch: CPU resource required, checking settings/requirements...
Submitted batch job nnnnnnn
```

Where `nnnnnnn` is your job ID number.

This job should run very quickly, but you may be able to find it in the job queue by typing:

```
squeue -u $USER
```

If it is running, you will see something like:

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
nnnnnnn	devel	submit.s	ouit0554	R	0:07	2	arc-c[302-303]

If the job is waiting to run (because another user is using the devel nodes) you will see:

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
nnnnnnn	devel	submit.s	ouit0554	PD	0:00	2	(None)

The difference being that in the first case you can see the job state is R for **RUNNING** and in the second it is PD for **PENDING** and it has not been allocated nodes in the NODELIST

Job output

When your job completes, i.e. it is no longer showing in the job queue, you should find the SLURM output in a file named `slurm-nnnnnnn.out` where *nnnnnnn* is the job ID of your completed job.

To view this output you can use the Linux `cat` command, so if our job ID was 2227191, we would use the command:

```
cat slurm-2227191.out
```

This would give the output:

```
Hello world from processor arc-c302, rank 0 out of 8 processors
Hello world from processor arc-c302, rank 1 out of 8 processors
Hello world from processor arc-c302, rank 2 out of 8 processors
Hello world from processor arc-c302, rank 3 out of 8 processors
Hello world from processor arc-c303, rank 4 out of 8 processors
Hello world from processor arc-c303, rank 5 out of 8 processors
Hello world from processor arc-c303, rank 6 out of 8 processors
Hello world from processor arc-c303, rank 7 out of 8 processors
```

The above being the output from running the `mpihello` application on the 8 CPUs that we requested, and you can see it ran with 4 processes on `arc-c302` and 4 on `arc-c303`

JOB SCHEDULING AND RESOURCES

9.1 Cluster Node Types

9.1.1 Login/submit nodes

These are only to be used for accessing the cluster and submitting jobs. Login nodes should not be used for software build or compute tasks. They are not designed for building software or running analysis; they do not have the same CPU architecture as the cluster nodes, and they do not run the same operating system as the cluster nodes. Please use the interactive nodes for any software builds (see below).

We have an explicit policy that user processes can only use a maximum of 1 hour of CPU time on login nodes.

9.1.2 Interactive nodes

These nodes should be used for pre/post processing of data and for building software to be used on the ARC clusters. See the section on interactive jobs for more information.

9.1.3 Compute nodes

Typical **arc** compute nodes have 48 cores. The **htc** compute nodes typically have 16, 20 or 48 cores.

9.2 The Job Scheduler

The **arc/htc** clusters use SLURM as their resource manager (or scheduler). This is the same system used for the ARC's previous clusters (ARCUS-B and ARCUS-HTC) so existing ARC users will be familiar with its commands and submission script syntax.

As a reminder, to work with ARC's clusters, you will need to submit a job to the job scheduler; the login nodes are for preparing and submitting scheduler jobs and should not be used for performing computational work. Most users choose to create a submission script in order to request resources and run their job commands. In this case resources such as the number of CPUs, nodes etc. are specified using **#SBATCH** directives in the script.

Note: You cannot use **#SBATCH** directives directly on the command line, these can only be specified in SLURM submission scripts. There are options to the **sbatch** and **srun** commands which can emulate these resource requests on the command line. For example [sbatch command help](#)

If you need to run interactive computational work such as pre/post processing data or building your own code - this must be performed on interactive nodes - see the section on [Interactive Jobs](#)

Nodes on **arc/htc** are not allocated exclusively to jobs; jobs are allocated the requested number of cores and may share nodes with other jobs. The default number of cores allocated is 1 (as is the default number of nodes). Default amount memory per CPU is 8000MB. You will not be able to use resources you have not requested in your job submission; this includes memory and CPU cores.

Thus if you need more than 1 CPU core, you will need to explicitly ask for them. At its simplest this can be specified by requesting a specific number of tasks, e.g.:

```
#SBATCH --ntasks-per-node=8
```

to request 8 tasks.

For MPI job submissions this would normally be changed to asking for a number of nodes and specifying the number of tasks per node, e.g.:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=48
```

to request two nodes with 48 tasks each.

For a hybrid MPI/OpenMP job, where an MPI tasks spawns multiple CPU threads, the specification needs to also specify how many CPUs per task the job will need. For example, to request 2 nodes with two MPI tasks per node that each start 24 compute threads, you need to request:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=24
```

The default number of CPUs per task is 1.

It is possible to request exclusive access to a node by adding **--exclusive** to your **sbatch** command or the following line to your submit script:

```
#SBATCH --exclusive
```

However, we strongly advise being specific about required resources rather than using exclusive node access; homogeneity of resources (or CPU features) can not be assumed. This is especially true on the HPC system, or when submitting to multiple clusters (see section Job Scheduling).

This is a very short overview; SLURM offers various ways to specify resource requirements; please see `man sbatch` for details.

9.3 Partitions

Both clusters have the following time-based scheduling partitions available:

- short (default run time 1hr, maximum run time 12hrs)
- medium (default run time 12hrs, maximum run time 48hrs)
- long (default run time 24hrs, no run time limit)
- devel (maximum run time 10 minutes - for batch job testing only)
- interactive (maximum run time 24hrs, can oversubscribe, for pre/post-processing and building software)

Jobs in the **short** and **medium** partitions are scheduled with higher priority than those in the **long** partition; however, they will not be able to run for longer than the time allowed on those partitions.

On the previous ARC clusters (ARCUS-B & ARCUS-HTC), users who wanted to submit long running jobs needed to submit the jobs to the scheduler specifying an acceptable timelimit, then once the job had started running, would request that the job's walltime be extended. On the new **arc/htc** clusters, this is no longer required; users can now submit jobs with longer time limits to the **long** partition.

Note: The default time limit on long partition is 1 day; users must specify a time limit if a longer runtime is required. We will no longer extend jobs.

The **htc** cluster has an additional partition available named legacy. This partition contains a number of nodes which have CentOS 7.7 installed in order to maintain compatibility with some legacy commercial applications. Access to the legacy partition is restricted to users with a requirement to use legacy software and will be enabled by the ARC team for specific users when it has been demonstrated that using a more recent version of the software application is not possible.

9.3.1 Cluster selection

By default jobs will be scheduled based upon the login node you using - if you are logged into **arc-login** jobs you submit will be queued to the **arc** cluster. If you are logged into **htc-login** jobs will be queued to the **htc** cluster.

However, The clusters are accessible from either login nodes and can be specified by passing `--clusters=arc` or `--clusters=htc` SLURM options. Additionally, `squeue` can report the status of jobs on either cluster (or both using the option `--clusters=all`).

It is possible for jobs to target either cluster or both clusters using the `-cluster` specification in job scripts, for example:

```
#SBATCH --clusters=arc
```

or:

```
#SBATCH --clusters=htc
```

or:

```
#SBATCH --clusters=all
```

If submitted with `--cluster=all` a job will simply be run on the first available resource, regardless of what cluster this is on.

9.4 Submission Scripts

As an example - to request two compute nodes, running 48 processes per node (using MPI), with one CPU per task (the default) requiring 2GB of memory per CPU, and a two hour wall time, the following submission script could be used:

```
#!/bin/bash

#SBATCH --nodes=2
#SBATCH --ntasks-per-node=48
```

(continues on next page)

(continued from previous page)

```
#SBATCH --mem-per-cpu=2G
#SBATCH --time=02:00:00
#SBATCH --job-name=myjob
#SBATCH --partition=short

module load mpitest/1.0

mpirun mpihello
```

To request a single core for 10 minutes, with one task on the node (and one CPU per task), requiring 8GB memory, a typical submission script would be:

```
#!/bin/bash

#SBATCH --time=00:10:00
#SBATCH --job-name=single_core
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=8G
#SBATCH --partition=short

module purge
module load testapp/1.0

# Calculate number of primes from 2 to 10000
prime 2 10000
```

9.5 Interactive Jobs

An interactive job gives you a login session on a compute node and gives you a shell. This allows users to interact with the node in real time, much like one would interact with a desktop PC, or the login nodes. We now expect users to use interactive jobs in order to run pre/post processing and software build activities - and there are nodes dedicated to these tasks.

To start an interactive session, you need to use the `srun` command, for example:

```
srun -p interactive --pty /bin/bash
```

or for a session that allows graphical interfaces (via X forwarding):

```
srun -p interactive --x11 --pty /bin/bash
```

This would allocate 1 core on one interactive node and log you in to the system (giving you a shell on the system). Multiple cores, memory, or other resources can be requested the same way as for `sbatch`.

Exiting the shell ends the job. It will also be aborted once it exceeds the time limit.

9.6 Interactive MPI Jobs

Extending the concept of interactive jobs to multi-node MPI jobs is reasonably simple.

First start a normal interactive session (as described above):

```
srun -p interactive --pty /bin/bash
```

From this session we now need to request a SLURM allocation for resources we require. In this case we are requesting 8 MPI tasks, with each task being on a different node and each MPI task having access to 2 CPU cores:

```
salloc --ntasks=8 --ntasks-per-node=1 --cpus-per-task=2
salloc: CPU resource required, checking settings/requirements...
salloc: Granted job allocation 3968751
salloc: Waiting for resource configuration
salloc: Nodes arc-c[179,184,191,193-194,201,210,214] are ready for job
```

Now that the job allocation is ready (and this may take a while if your resource requirements are complex) we can load any software environments required, in this example we will load the `mpitest` module:

```
module load mptest/1.0
```

Now, to run the MPI application on our resource allocation all we need to do is use the `srun` command in place of `mpirun`

For example, we can simply run the standard `mpihello` code as follows:

```
srun mpihello
Hello world from processor arc-c194, rank 4 out of 8 processors
Hello world from processor arc-c201, rank 5 out of 8 processors
Hello world from processor arc-c191, rank 2 out of 8 processors
Hello world from processor arc-c179, rank 0 out of 8 processors
Hello world from processor arc-c214, rank 7 out of 8 processors
Hello world from processor arc-c210, rank 6 out of 8 processors
Hello world from processor arc-c184, rank 1 out of 8 processors
Hello world from processor arc-c193, rank 3 out of 8 processors
```

9.7 Memory Resources

It is possible that your job may fail due to an out-of-memory error. These can manifest as explicit “OOM (Out-Of-Memory) killed” messages or errors such as “Segmentation fault” which may also indicate a memory issue.

In these cases it is important to try to understand how much memory the application you are running requires. Some MPI code may need to run on more cores in order to distribute the the problem and use less memory per node.

As shown in the above examples you can use the `--mem` option to request more memory on a node, the maximum per normal compute node on ARC being `--mem=380G`. On HTC there are two high memory nodes, so you can use `--mem=3000G` to use one of these.

Where you are getting persistent memory errors we would advise starting an `srun` session to connect to your job whilst it is running, using the command:

```
srun --jobid <jobid> --pty /bin/bash
```

You can then use the linux `top` command to monitor the memory utilisation (shown in the RES column) over time.

If your job is exceeding the 3TB limit on the HTC nodes, you will have to go back to your application to ascertain how to modify your input data in order to reduce the job size, some options being:

- In the case of large data-sets - splitting these into smaller files with multiple jobs via a job array.
- Reducing the problem/domain size.
- Gain a good understanding of your code by profiling where the large data structures are being created and potentially optimising these - there are many profiling solutions for Python code. ARC has Intel VTune available for use with your own C,C++,Fortran code.

9.8 GPU Resources

GPUs are only available on compute nodes which are part of the **htc** cluster. These resources are requested using the gres SLURM directive in your submission script.

The most basic way you can access a GPU is by requesting a GPU device using the gres option in your submission script:

```
#SBATCH --gres=gpu:1
```

The above will request 1 single GPU device (of any type) - this is the same as the method previously used on ARCUS-B/HTC. Note that - as with CPUs and memory - you will only be able to see the number of GPUs you requested.

You may also request a specific type of GPU device, for example:

```
#SBATCH --gres=gpu:v100:1
```

To request one V100 device, or:

```
#SBATCH --gres=gpu:rtx8000:2
```

To request two RTX8000 devices. Available devices are P100, V100, RTX (Titan RTX), RTX8000, and A100.

Alternatively you can request a GPU (`--gres=gpu:1`) and specify the type via a constraint on the GPU SKU, GPU generation, or GPU compute capability. Each of the following are valid forms of constraint:

```
#SBATCH --gres=gpu:1 --constraint='gpu_sku:V100'

#SBATCH --gres=gpu:1 --constraint='gpu_gen:Ampere'

#SBATCH --gres=gpu:1 --constraint='gpu_cc:10.0'

#SBATCH --gres=gpu:1 --constraint='gpu_mem:32GB'

#SBATCH --gres=gpu:1 --constraint='nvlink:2.0'
```

9.8.1 List of configured GPU related constraints

gpu_gen:	GPU generation (Pascal, Volta, Turing, Ampere)
gpu_sku:	GPU model (P100, V100, RTX, RTX8000, A100)
gpu_cc:	GPU hardware compute capability
gpu_mem:	GPU memory
nvlink:	device has nvlink - constraint exist as simple (-C nvlink) and specifying version (-C 'nvlink:2.0')

Note: Please note that co-investment GPU nodes are limited to short partition, i.e. the maximum job run time is 12 hours. No such restrictions apply to ARC owned GPUs. For details on available options/combinations, and ownership information, see the table of available GPUs.

SUBMITTING PRIORITY JOBS

Jobs submitted to ARC cluster default to the standard service level. To purchase priority credit see the [Requesting Credits](#) page for details. Once your project is set up with priority credits, you can check the balance of your ARC credit using the `mybalance` command. You'll find now that the command reports balance for two accounts. One account is for your standard credit. The account with the `-priority` suffix is for your priority credit.

10.1 Submit jobs with priority QoS (Quality of Service)

To submit jobs with higher priority, you need to add the `--qos=priority` option to your `sbatch` submission. This can be on the command line:

```
sbatch --qos=priority ...
```

where “...” is replaced by whatever other options you might normally include on your `sbatch` commands.

Or you can include an `#SBATCH` directive comment at the beginning of your submission script:

```
#SBATCH --qos=priority
```

A job submitted with the “Quality of Service” (qos) setting “priority” will run with higher priority and consume credits from the `-priority` account. When you run out of priority credit your priority jobs will be held. They can have their QOS set to normal and then they will be able to run as standard priority jobs.

ARC STORAGE

Once your account has been created, you will immediately have access to two persistent storage areas:

\$HOME (/home/username) with a 15GB quota

\$DATA (/data/projectname/username) sharing a 5TB quota with your project colleagues

Additionally, when you run a SLURM job, a per-job **\$SCRATCH** and **\$TMPDIR** for temporary data/workfiles is created.

\$TMPDIR is local to a compute node

\$SCRATCH is on a shared file system and available to all nodes in a job, if a job spans multiple nodes.

11.1 Using ARC **\$SCRATCH** storage

Note: Both **\$SCRATCH** and **\$TMPDIR** are not persistent; they will be automatically removed on job exit. It is important that your job copies all files into your **\$DATA** area before it exits; we will not be able to recover your data if you left it on **\$SCRATCH** or **\$TMPDIR** once a job finished.

As a rule we recommend that you use your **\$DATA** area to store your data, but utilise the per job **\$SCRATCH** or **\$TMPDIR** area - especially for intermediate or temporary files. Generally you would copy all required input data at the start of your job and then copying results back to your **\$DATA** area.

A simple example of how to do this would be:

```
#!/bin/bash
#
# After SBATCH lines in submission script...
#
#
cd $SCRATCH || exit 1
#
# Copy job data to $SCRATCH
#
rsync -av $DATA/myproject/input ./
rsync -av $DATA/myproject/bin ./
#
# Job specific lines...
#
module load foss/2020b

mpirun ./bin/my_software
```

(continues on next page)

(continued from previous page)

```
#  
# Copy data back from $SCRATCH to $DATA/myproject directory  
#  
rsync -av --exclude=input --exclude=bin ./ $DATA/myproject/
```

This example copies the directories `$DATA/myproject/input` and `$DATA/myproject/bin` into **\$SCRATCH** (which will then contain directories `input` and `bin`). The script then runs `./bin/my_software`; and copies all files in the **\$SCRATCH** directory - excluding directories `input` and `bin` - back to `$DATA/myproject/` once the `mpirun` finishes.

The process is more straightforward if you only need to copy single input/output files when the application is centrally hosted, for example:

```
#!/bin/bash  
#  
# After SBATCH lines in submission script...  
#  
#  
# Load appropriate module, in this test case we are using Gaussian  
  
module load Gaussian/16.C.01  
  
# Set input/output filenames  
#  
export INPUT_FILE=test397.com  
export OUTPUT_FILE=test397.log  
  
echo "copying input from $SLURM_SUBMIT_DIR/$INPUT_FILE ..."  
  
cd $SCRATCH || exit 1  
cp $SLURM_SUBMIT_DIR/$INPUT_FILE ./  
  
# Job specific application command line...  
#  
g16 < $INPUT_FILE > $OUTPUT_FILE  
  
# Copy output back from $SCRATCH to job directory  
#  
echo "copying output back to $SLURM_SUBMIT_DIR/ ..."  
cp $OUTPUT_FILE $SLURM_SUBMIT_DIR/
```

If you are unable to access either of these directories, please let us know.

11.2 Quota

By default your **\$HOME** area will have a 15GB quota while the **\$DATA** area will have a 5TB quota that is shared between yourself and the other members of your project.

To check your quota use the command:

```
myquota
```

This command will list both your home quota and the quota of shared project data areas that you are a member of.

We can provide more detailed statements of data area quota usage to project leaders on request.

Larger Data quotas (more than 5TB) are available on request as a chargeable service. Please contact ARC support for further information.

If you are a user of Anaconda virtual environments and find yourself over quota in **\$HOME**, please check your conda package cache size. Information on this can be found here: [Anaconda Package Cache](#)

11.3 Backups

We do NOT currently create backups of the ARC shared file system (although the file system IS resilient to failures). We therefore strongly encourage you to keep copies of your files elsewhere, particularly when that data is critical to your research.

11.4 Snapshots

Snapshots have been configured to be generated on home directories. Snapshots provide easy access to older versions of files. This is useful if files have been accidentally deleted or overwritten. It does not, however, constitute a backup; old snapshots will not be kept indefinitely (max. two weeks for weekly snapshots).

Within your home directory, there is a `.snapshot` directory which contains the hourly, daily and weekly snapshots available. To list/examine the snapshots, simply `'cd'` into `$HOME/.snapshot` and list the available directories:

```
cd $HOME/.snapshot
ls -ltr
```

You will see a listing of all snapshots (reverse order, i.e. newest last):

```
weekly.2020-08-02_0015
weekly.2020-07-26_0015
daily.2020-08-06_0010
hourly.2020-08-07_1105
hourly.2020-08-07_1005
hourly.2020-08-07_0905
daily.2020-08-07_0010
hourly.2020-08-07_1305
hourly.2020-08-07_1205
hourly.2020-08-07_1405
```

To choose a particular snapshot, simply change into the relevant directory:

```
cd hourly.2020-08-07_1205
```

Within those directories you will essentially find a copy of your home directory as it was when the snapshot was taken.

If you've accidentally deleted a file in your home directory which existed earlier than the last snapshot, then you can retrieve the older copy from the snapshot. Simply find the version of the file you are after within the `.snapshot` structure, and copy it back into your home directory.

For example - assuming you have deleted a file `'ARC-Introduction-2018-Hilary.pptx'` from folder `$HOME/Documents` by mistake. To recover it, the steps would be:

```
[$(arcus) Documents]$ pwd
/home/ouit0622/Documents

[$(arcus) Documents]$ ls -l
ARC-Introduction-2018-Hilary.pptx
arc_job_submission_exercises
arc_presentation
MATLAB

[$(arcus) Documents]$ rm ARC-Introduction-2018-Hilary.pptx

[$(arcus) Documents]$ ls -l
arc_job_submission_exercises
arc_presentation
MATLAB

[$(arcus) Documents]$ cd $HOME/.snapshot/
[$(arcus) .snapshot]$ ls -ltr
weekly.2020-08-02_0015
weekly.2020-07-26_0015
daily.2020-08-06_0010
hourly.2020-08-07_1105
hourly.2020-08-07_1005
hourly.2020-08-07_0905
daily.2020-08-07_0010
hourly.2020-08-07_1305
hourly.2020-08-07_1205
hourly.2020-08-07_1405

[$(arcus) .snapshot]$ cd hourly.2020-08-07_1405

[$(arcus) hourly.2020-08-07_1405]$ pwd
/home/ouit0622/.snapshot/hourly.2020-08-07_1405

[$(arcus) hourly.2020-08-07_1405]$ cd Documents

[$(arcus) Documents]$ ls -l
ARC-Introduction-2018-Hilary.pptx
arc_job_submission_exercises
arc_presentation
MATLAB

[$(arcus) Documents]$ cp ARC-Introduction-2018-Hilary.pptx $HOME/Documents

[$(arcus) Documents]$ $HOME/Documents/
[$(arcus) Documents]$ pwd
/home/ouit0622/Documents

[$(arcus) Documents]$ ls -l
ARC-Introduction-2018-Hilary.pptx
arc_job_submission_exercises
arc_presentation
MATLAB
```

Note: Snapshots do not take up space in the file system, i.e. they do not count towards your quota. If you are trying to determine where in your home directory space is used, you must exclude the .snapshot directory from your commands as otherwise the information would be incorrect.

ACCESSING INSTALLED SOFTWARE APPLICATIONS

12.1 Environment Modules

The Linux operating system makes extensive use of the “working environment”, which is a collection of individual environment variables. An environment variable is a named object in the Linux shell that contains information used by one or more applications; two of the most used such variables are **\$HOME**, which defines a user’s home directory name, and **\$PATH**, which represents a list paths to different executables. A large number of environment variables are already defined when a Linux shell is open but the environment can be customised, either by defining new environment variables relevant to certain applications (e.g. software license variables) or by modifying existing ones (e.g. adding a new path to **\$PATH**).

module is a Linux utility, which is used to manage of working environment in preparation for running the applications installed on the ARC systems. By loading the module for a certain installed application, the environment variables that are relevant for that application are automatically defined or modified.

The ARC/HTC software environment comprises a mixture of commercial applications, software built using the Easy-Build framework and software built using our own local build recipes. We use the environment modules system (via the **module** command) to load applications into the environment on ARC/HTC.

However, because the EasyBuild framework adds many new module components into the module list - the best way to search for an application you require is by using the module spider command. For example, to search for the GROMACS application:

```
module spider gromacs

-----
↪ -----
GROMACS:
-----
↪ -----
Description:
  GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the
↪ Newtonian equations of motion for
  systems with hundreds to millions of particles. This is a CPU only build, containing
↪ both MPI and threadMPI builds.

Versions:
  GROMACS/2020-fosscuda-2019b
  GROMACS/2020.4-foss-2020a-PLUMED-2.6.2
  GROMACS/2020.4-foss-2020a
```

Please note, module spider is NOT case-sensitive for searching, so:

```
module spider GROMACS
module spider gromacs
module spider Gromacs
```

... are all equivalent. However, when loading module using module load you must use the correct case, for example:

```
module load GROMACS/2020.4-foss-2020a
```

Please note: It is important to run the module spider command from an interactive compute session because the ARC/HTC login nodes do not have all modules available. See example below:

```
[software@arc-login01 ~]$ srun -p interactive --pty /bin/bash
srun: CPU resource required, checking settings/requirements...

[software@arc-c304 ~]$ module spider GROMACS

-----
↪ -----
GROMACS:
-----
↪ -----
Description:
  GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the
↪ Newtonian equations of motion for systems with hundreds to millions of particles. This
↪ is a CPU only
  build, containing
  both MPI and threadMPI builds.

Versions:
  GROMACS/2020-fosscuda-2019b
  GROMACS/2020.4-foss-2020a-PLUMED-2.6.2
  GROMACS/2020.4-foss-2020a
  GROMACS/2021-foss-2020b
  GROMACS/2021-foss-2021a-PLUMED-2.7.2

-----
↪ -----
  For detailed information about a specific "GROMACS" package (including how to load the
↪ modules) use the module's full name.
  Note that names that have a trailing (E) are extensions provided by other modules.
  For example:

  $ module spider GROMACS/2021-foss-2021a-PLUMED-2.7.2

-----
↪ -----
```

If the software name you are using in “module spider” returns too many options you can use

```
module -r spider '^name'
```

for example:

```
module -r '^Python'
module -r '^R$'
```

You can also build your own software in your home or data directories using one of the compilers provided (which are also available through the environment modules system). Typically the compiler toolchains, including maths libraries and MPI can be loaded using the modules named foss (e.g. foss/2020a) for free open-source software (i.e. GCC) or intel (e.g. intel/2020a) for the Intel compiler suite.

If no version is specified, the default version of the software is loaded (usually the latest version):

```
module load GROMACS
module list GROMACS

Currently Loaded Modules Matching: GROMACS
  1) GROMACS/2020.4-foss-2020a
```

Specific versions, other than the default can be loaded by specifying the version:

```
module load GROMACS/2020.4-foss-2020a-PLUMED-2.6.2
module list GROMACS

Currently Loaded Modules Matching: GROMACS
  1) GROMACS/2020.4-foss-2020a-PLUMED-2.6.2
```

A module can be “unloaded” with the unload option, for example:

```
module unload MATLAB/2020b
```


SLURM REFERENCE GUIDE

13.1 Using the SLURM job scheduler

Important note: This guide is an introduction to the SLURM job scheduler and its use on the ARC clusters. ARC compute nodes typically have two 24 core processors and a range of memory sizes, from 384GB to 3TB; however, there are some nodes with more (or fewer) cores and/or memory. In addition, some nodes in the HTC cluster have NVidia Tesla GPGPU cards.

13.2 Introduction

Jobs are run as batch jobs, i.e. in an unattended manner. Typically a user logs in to a login node (e.g. `arc-login.arc.ox.ac.uk`), prepares a job (which contains details of the work to carry out and the computer resources needed) and submits it to the job queue. The user can then log out (if she/he wishes) until their job has run, to collect the output data.

Jobs are managed by SLURM, which is in charge of:

- allocating the computer resources requested for the job,
- running the job and
- reporting the outcome of the execution back to the user.

Running a job involves, at the minimum, the following steps:

- preparing a submission script and
- submitting the job to execution.

This guide describes basic job submission and monitoring for SLURM. The topics in the guide are:

- the main SLURM commands,
- preparing a submission script,
- SLURM partitions,
- submitting a job to the queue,
- monitoring a job execution,
- deleting a job from the queue and
- environment variables.
- job dependencies
- job arrays

13.3 Commands

The table below gives a short description of the most used SLURM commands.

Com-mand	Description
sbatch	Submit a job script for later execution (the script typically contains one or more srun commands to launch parallel tasks)
sinfo	Reports the state of partitions and nodes managed by SLURM (it has a variety of filtering, sorting, and formatting options)
squeue	Reports the state of jobs (it has a variety of filtering, sorting, and formatting options), by default, reports the running jobs in priority order followed by the pending jobs in priority order
srun	Used to submit a job for execution in real time
scancel	Cancel a pending or running job
sacct	Report job accounting information about active or completed jobs
salloc	Allocate resources for a job in real time (typically used to allocate resources and spawn a shell, in which the srun command is used to launch parallel tasks)

All SLURM commands have extensive help through their man pages, for example:

```
man sbatch
```

Will show you the help pages for the **sbatch** command.

13.4 Preparing a submission script

A submission script is a shell script that:

- describes the processing to carry out (e.g. the application, its input and output, etc.) and
- requests computer resources (number of cpus, amount of memory, etc.) to use for processing.

The simplest case is that of a job that requires a single node (this is the smallest unit we allocate on ARC) with the following requirements:

- the job uses 1 node,
- the application is a single process,
- the job will run for no more than 100 hours,
- the job is given the name “test123” and
- the user should be emailed when the job starts and stops or aborts.

13.5 Example 1: job running on a single node

Supposing the application is called myCode and takes no command line arguments, the following submission script runs the application in a single job:

```
#!/bin/bash
# set the number of nodes
#SBATCH --nodes=1
# set max wallclock time
#SBATCH --time=100:00:00
# set name of job
#SBATCH --job-name=test123
# mail alert at start, end and abortion of execution
#SBATCH --mail-type=ALL
# send mail to this address
#SBATCH --mail-user=john.brown@gmail.com

# run the application
myCode
```

The script starts with `#!/bin/bash` (also called a shebang), which makes the submission script a Linux bash script.

The script continues with a series of lines starting with `#`, which represent bash script comments. For SLURM, the lines starting with `#SBATCH` are directives that request job scheduling resources. (Note: it's very important that you put all the directives at the top of a script, before any other commands; any `#SBATCH` directive coming after a bash script command is ignored!)

The resource request `#SBATCH --nodes=n` determines how many compute nodes a job are allocated by the scheduler; only 1 node is allocated for this job. A note of caution is on threaded single process applications (e.g. Matlab). These applications cannot run on more than a single compute node; allocating more (e.g. `#SBATCH --nodes=2`) will end up with the first node being busy and the rest idle.

The maximum walltime is specified by `#SBATCH --time=T` where T has format hh:mm:ss. Normally, a job is expected to finish before the specified maximum walltime. After the walltime reaches the maximum, the job terminates regardless whether the job processes are still running or not.

The name of the job can be specified too with `#SBATCH --job-name="name"`

Lastly, an email notification is sent if an address is specified with `#SBATCH --mail-user=<email_address>` The notification options can be set with `#SBATCH --mail-type=<type>` where <type> may be BEGIN, END, FAIL, REQUEUE or ALL (for any change of job state).

The final part of a script is normal Linux bash script and describes the set of operations to follow as part of the job. The job starts in the same folder where it was submitted (unless an alternative path is specified), and with the same environment variables (modules, etc.) that the user had at the time of the submission. In this example, this final part only involves invoking the myCode application executable.

13.6 Example 2: job running on multiple nodes

As a second example, suppose we want to run an MPI application called myMPICode with the following requirements:

- the run uses 2 nodes,
- the job will not run for more than 100 hours,
- the job is given the name “test123” and
- the user should be emailed when the job starts and stops or aborts.

Supposing no input needs to be specified, the following submission script runs the application in a single job:

```
#!/bin/bash
# set the number of nodes and processes per node
#SBATCH --nodes=2
# set the number of tasks (processes) per node.
#SBATCH --ntasks-per-node=16
# set max wallclock time
#SBATCH --time=100:00:00
# set name of job
#SBATCH --job-name=test123
# mail alert at start, end and abortion of execution
#SBATCH --mail-type=ALL
# send mail to this address
#SBATCH --mail-user=john.brown@gmail.com

mpirun myMPICode
```

In large part, the script above is similar to the one for a single node job except in this example, #SBATCH --ntasks-per-node=m is used to reserve m cores per node and to prepare the environment for a MPI parallel run with m processes per each compute node.

13.7 SLURM partitions

SLURM partitions are essentially different queues that point to collections of nodes.

You can specify the SLURM partition by adding the #SBATCH --partition= directive to the top of your submission script so adding:

```
#SBATCH --partition=devel
```

will send your job to the **devel** partition. Alternatively, the partition can be supplied with the sbatch command like this:

```
sbatch --partition=devel JOBSRIPT.sh
```

Defining a partition on the sbatch command line takes precedence over the definition in the jobscript.

You can see the current state of the partitions with the sinfo command.

All Slurm commands have extensive help through their man pages; try for example:

```
man sbatch
```


13.8 Submitting jobs with the command sbatch

Once you have a submission script ready (e.g. submit.sh), the job is submitted to the execution queue with the command:

```
sbatch submit.sh
```

The queueing system prints a number (the job id) almost immediately and returns control to the linux prompt. At this point the job is in the submission queue.

Once you have submitted the job, it will sit in a pending state until the resources have been allocated to your job (the length of time your job is in the pending state is dependent upon a number of factors including how busy the system is and what resources you are requesting). You can monitor the progress of the job using the command squeue (see below).

Once the job starts to run you will see files with names such as slurm-1234.out either in the directory you submitted the job from (default behaviour) or in the directory where the script was instructed explicitly to change to.

13.9 Monitoring jobs with the command squeue

squeue is the main command for monitoring the state of systems, groups of jobs or individual jobs.

The command squeue prints the list of current jobs. The list looks something like:

JOBID	PARTITION	NAME	USER	ST	TIME	Nodes	NODELIST(REASON)
2497	short	test1.14	bob	R	0.07	1	arc-c252
2499	long	test1.35	mary	R	0.22	4	arc-c(200-203)
2511	devel	ask.for	steve	PD	0.00	1	(Resources)

The first column gives the job ID, the second the partition (or queue) where the job was submitted, the third the name of the job (specified by the user in the submission script) and the fourth the owner of the job. The fifth is the status of the job (R=running, PD=pending, CA=cancelled, CF=configuring, CG=completing, CD=completed, F=failed). The sixth column gives the elapsed time for each particular job. Finally, there are the number of nodes requested and the nodelist where the job is running (or the cause that it is not running).

Some other useful squeue features include:

```
-u for showing the status of all the jobs of a particular user, e.g. squeue -u bob for
↪user bob;
-l for showing more of the available information;
--start to report the expected start time of pending jobs.
```

Read all the options for squeue on the Linux manual using the command `man squeue` including how to personalize the information to be displayed.

13.10 Deleting jobs with the command scancel

Use the scancel command to delete a job, for example:

```
scancel 1121
```

to delete job with ID 1121. A user can delete his/her own jobs at any time, whether the job is pending (waiting in the queue) or running. A user cannot delete the jobs of another user. Normally, there is a (small) delay between the execution of the scancel command and the time when the job is dequeued and killed. Occasionally a job may not delete properly, in which case, the ARC support team can delete it upon request.

13.11 Environment variables

At the time a job is launched into execution, Slurm defines multiple environment variables, which can be used from within the submission script to define the correct workflow of the job. The most useful of these environment variables are the following:

```
SLURM_SUBMIT_DIR, which points to the directory where the sbatch command is issued;
SLURM_JOB_NODELIST, which returns the list of nodes allocated to the job;
SLURM_JOB_ID, which is a unique number Slurm assigns to a job.
```

In most cases, ``SLURM_SUBMIT_DIR`` does not have to be used, as the job goes by default to the directory where the slurm command was issued. This behaviour of SLURM is in contrast with other schedulers, such as Torque, which goes to the home directory of the user account. SLURM_SUBMIT_DIR can be useful in a submission script when files must be copied to/from a specific directory that is different from the directory where the slurm command was issued.

SLURM_JOB_ID is useful to tag job specific files and directories, typically output files or run directories. For instance, the submission script line:

```
myApp > $SLURM_JOB_ID.out
```

runs the application myApp and redirects the standard output to a file whose name is given by the job ID. The job ID is a number assigned by SLURM and differs from the character string name given to the job in the submission script by the user.

13.12 Job Dependencies

Job dependencies are used to defer the start of a job until the specified dependencies have been satisfied.

They are specified with the `--dependency` option to sbatch in the format:

```
sbatch --dependency=<type:job_id[:job_id][,type:job_id[:job_id]]> ...
```

Dependency types:

```
after:jobid[:jobid...]    job can begin after the specified jobs have started
afterany:jobid[:jobid...]  job can begin after the specified jobs have terminated
afternotok:jobid[:jobid...] job can begin after the specified jobs have failed
afterok:jobid[:jobid...]   job can begin after the specified jobs have run to
↳ completion with an exit code of zero
```

For example:

```

sbatch job1.sh
1802051
sbatch --dependency=afterok:1802051 job2.sh

```

In the above example, job script **job1.sh** is submitted and is given a JobID of 1802051. We then submit **job2.sh** with a dependency that it only run when job 1802051 has completed.

13.13 Job Arrays

Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily. In general, job arrays are useful for applying the same processing routine to a collection of multiple input data files. Job arrays offer a very simple way to submit a large number of independent processing jobs.

By submitting a single job array sbatch script, a specified number of “array-tasks” will be created based on this “master” sbatch script.

For example:

```

#!/bin/bash
#SBATCH --job-name=arrayJob
#SBATCH --output=arrayJob_%A_%a.out
#SBATCH --error=arrayJob_%A_%a.err
#SBATCH --array=1-4
#SBATCH --time=02:00:00

# Print this sub-job's task ID
echo "My SLURM_ARRAY_TASK_ID: " $SLURM_ARRAY_TASK_ID

# Run "application" using input filename modified by SLURM_ARRAY_TASK_ID
./application input_${SLURM_ARRAY_TASK_ID}.txt

```

The above example uses the `--array=1-4` specification to create four array tasks which run the command “application” on different input files, the filename of each being modified by the `SLURM_ARRAY_TASK_ID` variable.

The `%A_%a` construct in the output and error file names is used to generate unique output and error files based on the master job ID (`%A`) and the array-task’s ID (`%a`). In this fashion, each array-task will be able to write to its own output and error file.

For clarity, the input and output files for the above script, if submitted as jobID 1802055 would be:

JobID	--output	--error	Application Input filename
1802055_1	arrayJob_1802055_1.out	arrayJob_1802055_1.err	input_1.txt
1802055_2	arrayJob_1802055_2.out	arrayJob_1802055_2.err	input_2.txt
1802055_3	arrayJob_1802055_3.out	arrayJob_1802055_3.err	input_3.txt
1802055_4	arrayJob_1802055_4.out	arrayJob_1802055_4.err	input_4.txt

Note: You can specify the `--array` option on the `sbatch` command line instead of inside the submission script. For example if the `--array` option was removed from the above script and the script was named **jobArray.sh** the command would be:

```

sbatch --array=1-4 jobArray.sh

```

More information about SLURM job arrays can be found in the [Slurm Job Array Documentation](#)

SLURM FAQs

14.1 How do I submit, check the status, and/or delete a batch job?

Use the SLURM commands : `sbatch`, `squeue` , `scancel`

With a submission script called **submit.sh**, to submit this batch script, use the `sbatch` command:

```
sbatch submit.sh
```

To check the status of a batch job use the command `squeue`. If you know the job ID, ie. 12345 then use the command:

```
squeue -j 12345
```

To delete a batch job, you need to know the job ID and then use the `scancel` command:

```
scancel 12345
```

14.2 Why does my queued job not start?

Jobs may be queued for various reasons. A job may be waiting for resources to become available. Or you might have hit a limit for the maximum number of jobs that can be running on the system. One way to determine why a job is queuing is to use the `scontrol show job` command. For example, if the job ID is 12345:

```
scontrol show job 12345
```

The Reason field, normally near the top of the `scontrol show job` output may be helpful. Some common reason messages include:

JobHeldUser

This means your job is held because your ARC project has run out of compute “credit”. Please contact support@arc.ox.ac.uk for a top up.

You can release jobs held using the command:

```
scontrol release <JobID>
```

Where <JobID> above is the numeric job ID to release.

To check your credit balance at any time you can use the command:

```
mybalance
```

Email us when credit is low and we will add credit so that your jobs are not held.

PartitionTimeLimit

The walltime you have requested is in excess of the time allowed in the partition you have submitted to.

ReqNodeNotAvail

The most common reason for this message is that the nodes are reserved. This may be for systems maintenance or in the case of co-investment nodes, reserved by the stakeholder.

Priority

Your job is waiting because jobs of higher priority are being scheduled first. The main reason for this will be your recent usage of the ARC systems; as more and more of your submitted jobs run on the clusters the successful running of those jobs will gradually erode the level of priority accorded by the scheduler to your next upcoming job. This is to ensure that all users of ARC's systems get a "fair share" of the resource and than no single individual or project can dominate usage of the clusters e.g. by submitting thousands of jobs. Fair-share is decided by the scheduler's fair-share algorithm and your job will run once the higher priority jobs have been scheduled and/or your job's priority increases due to its time in the queue.

Fair-share decay is calculated every 5 minutes, with a half-life of 14 days. It will therefore have no effect if you have not had a running job in the previous 28 days.

Please note fair-share priority is not linked to number of jobs you submitted and are queued, only the number of jobs you have successfully run on the systems.

Resources

Your job is waiting for enough compute resource to become available.

QOSGrpNodeLimit

This happens when you have specified a Quality of Service resource (normally because you are requesting Co-Investment nodes). The message indicates that all the nodes available for your QoS are currently in use. Your job will run once enough resource becomes available on your QoS. If you need to run urgently you could simply run the line:

```
scontrol update jobid=<jobid> qos=standard
```

..to update your job to use **standard** Qos, this will allow your job to use the standard QoS and therefore have access to the whole cluster.

QOSMaxWallDurationPerJobLimit

This issue is typically seen by users that are members of ARC projects which have Basic or "free" quality of service. Basic QoS is typically associated with all projects where a Principal Investigator's home Department is based within an MSD Department. Basic QoS allows free use at the point of access with the following scheduling attributes:

- You have a *maximum* timelimit of 24 hours per job.
- You can submit multiple jobs BUT only run one concurrent job on the clusters.
- Jobs will have a lower prioritisation than jobs submitted with Standard QoS and will therefore be scheduled around Standard and Priority QoS jobs

More details here: <https://www.arc.ox.ac.uk/arc-service-level-agreements>

14.3 Why does my job fail with the error “/bin/bash^M: bad interpreter: No such file or directory”

You have edited your submission script using a Windows editor (such as notepad). Windows has extra characters at the end of a line, which are not needed under Linux and which cause the Torque qsub command to fail.

Eliminate the extra characters by using the following command:

```
dos2unix myScript.sh
```

14.4 Why does my job fail/die after running for just a few seconds?

There is a problem with the job submission script. The error output from the job submission should provide some information as to why the job failed. If you require help with determining what the problem is, please contact the ARC support team and provide relevant details to help with diagnosis. This should include Job ID and batch script details. Time/date of submission and which ARC system the job was submitted is also useful.

14.5 Why does my job fail/die after running for a few hours/days?

Possibly your job has run out of walltime. Every job has a walltime limit that is specified in the submission script or by the sbatch command or picked up from the relevant default value. See next question regarding requesting increase to the walltime of a running job.

14.6 How can I increase the walltime of a running job?

If you submit a job and find that it may not finish within the requested walltime, then to avoid having the job terminated when it reaches its walltime limit, please contact the ARC support team with details of the job (Job ID and ARC system the job is running on) requesting that the job walltime be increased. If you are able to estimate the additional walltime required this is helpful.

14.7 How can I get an email notification when a job begins/finishes?

Include the `--mail-type` and `--mail-user` options in the job submission script. These can be specified at the beginning of the job submission script as a line of the form:

```
#SBATCH --mail-type=BEGIN,END
#SBATCH --mail-user=email.address@unit.ox.ac.uk
```

or included on the sbatch command line as:

```
sbatch --mail-type=BEGIN,END --mail-user=email.address@unit.ox.ac.uk submit.sh
```

More details about sbatch options can be found in the sbatch man page (`man sbatch`)

14.8 How can I check the availability of free compute nodes?

Use the command the SLURM command `sinfo`

GENERAL FAQs

15.1 How do I become a ARC user?

The supercomputing facility is available to all Oxford University researchers. You can become a user by registering with the ARC. Start with the [User registration Page](#)

15.2 Where should a new ARC user begin?

Look at information in [this section of the ARC website](#).

15.3 What systems can I access as a user?

All registered users have access to all the ARC systems.

15.4 How much disk space do I have?

Default quota on \$HOME is 15 gigabytes (GB) per user and on \$DATA is 5 terabytes (TB) per project/group. Larger quotas limits are available on request.

15.5 How do I log on to the ARC systems?

Linux and Mac users should use `ssh` to connect to the ARC systems. For example, issuing the following command:

```
ssh -X bob@system.arc.ox.ac.uk
```

from a Linux or Mac terminal, the user bob log on to **system** (**system** could be `arc-login` or `htc-login`). The user is prompted to provide the password.

We recommend the use of the `-X` flag to `ssh`, which allows users to run graphical applications remotely (such as the emacs editor). This option allows X11 forwarding using subject to security control.

Windows users should install an application called PuTTY. To connect to one of the ARC machines, PuTTY has to operate in the `ssh` mode. To allow remote applications to display graphics through the link, the X11 tunnelling option has to be enabled. (The best way to learn about how to do this is to search the web for “putty x11 tunnelling”.) Additionally, Windows users that wish to use X11 graphics via PuTTY must install a X11 server, and `xming` is arguably the best option.

Windows users can also use a combined SSH/X-Server client such as [MobaXTerm](#) to connect to ARC systems.

15.6 How do I transfer files to/from the ARC systems?

Linux and Mac users should use `scp` for transferring files. For example, the command:

```
scp localfolder/myfile.txt bob@system.arc.ox.ac.uk:/path/to/remote/folder
```

copies the file `myfile.txt` from the local host to system in the directory `/path/to/remote/folder` on the ARC system. Also:

```
scp bob@system.arc.ox.ac.uk:/path/to/remote/folder/myfile.txt localfolder/
```

copies `myfile.txt` from the remote directory to the local one. In both cases, the target file is overwritten if it exists. Study the `scp` Linux manual (`man scp`) to learn more; for example the `-r` (recursive) option is useful for transferring entire directories.

Linux and Mac users may also find `rsync` very useful, for instance for maintaining directory structures on their workstation in sync with copies on the ARC storage.

The command:

```
rsync -avz localfolder bob@system.arc.ox.ac.uk:/path/to/remote/folder
```

copies the directory `localfolder` to the directory `/path/to/remote/folder` on the ARC system. The files are transferred recursively in “archive” mode, which ensures that symbolic links, permissions, ownerships, etc. are preserved. The example above also uses compression `-z` to reduce the size of the transferred data. Unlike `scp`, `rsync` does not simply overwrite existing files but rather updates, transferring only the differences between two files.

Windows users can use `pscp`, usually bundled with PuTTY. Alternatively, WinSCP is a very easy to use, open source `scp` client for Windows.

15.7 How do I access the ARC systems from outside the University network?

There are two main ways to access the ARC systems from outside the University network:

By connecting to the University of Oxford virtual private network (VPN) and accessing the ARC systems as though users were on campus (see [University VPN Information](#) for more details).

By connecting to the ARC external access server, `gateway.arc.ox.ac.uk` gateway has a firewall in place and only permits access from known locations. Thus, users need to provide the ARC (static) IP addresses from the location that they need to connect from. This should be an IP address from an identifiable institution.

The first method is the preferred form for any users who have University SSO passwords. The second method is useful for users who are external collaborators.

15.8 How do I acknowledge ARC in my publications?

Please see the Acknowledgements section on the [ARC Policies](#) page

15.9 Will application X run on the ARC supercomputers faster than on my workstation?

We hope so, but it may not for many reasons. If you wish to investigate whether you can achieve speed up on your application, then please contact us.

15.10 How many credits do I have left?

Use the `mybalance` command on ARC or HTC to find out how many credits you have left.

15.11 How do I change my password?

Use the `passwd` command on the login nodes to change your password:

```
user@arc-login~$ passwd  
  
Enter login(LDAP) password:  
Enter new password:  
Re-enter new password:
```

If you need to change other things in your account (e.g. email address), this is possible - please ask the ARC team by emailing support@arc.ox.ac.uk.

15.12 I have forgotten my password. How do I reset my password?

If you have forgotten your password or your password has expired and you can no longer access ARC to change it yourself, you will need to contact us using the support email address. We will then issue a temporary password via email.

15.13 I accidentally deleted files, how do I get them back?

Unfortunately ARC does not keep dedicated backups of its storage resources. We recommend that users store data at sites other than the ARC, for example, on departmental resources.

HOST KEY FINGERPRINTS FOR ARC LOGIN NODES

Server Name	Hash Algorithm	Signature Type	Fingerprint	
arc-login	MD5	ED25519	3e:d7:e1:20:76:91:af:5c:54:82:9d:15:c6:42:52:85	
		RSA	da:b7:c2:d3:66:f7:0b:35:e5:96:7e:b5:ae:8e:ff:de	
		ECDSA	f2:d7:01:cb:3c:14:ca:1f:c8:6a:34:a9:8c:f2:74:e4	
	SHA256	RSA	47OUV3Jm8crB/j9NkSBa8sjBT6uJ7TVvJ+Hi1XUmyAI	
		ED25519	GA6PFik/IY15ERzqElm3Jts10kg+VwMWKbSIU9CDi6g	
		ECDSA	qRv0Jhq96SQH+g4lHrOtJm3sxtnn0p48h20hWWy1zog	
htc-login	MD5	RSA	b0:6e:1e:1f:d7:be:5f:a4:6f:70:1c:d7:9e:1c:b1:a1	
		ECDSA	68:5e:c4:3d:8d:98:bc:cd:15:12:67:1e:ba:2e:6f:3c	
		ED25519	fe:46:78:54:87:d6:d8:ae:d2:31:df:61:69:e3:50:d4	
	SHA256	ECDSA	+4MBR+UWPBcQl+uomfWQRYaX3H5rRci1ZTNZyaRpjBg	
		ED25519	2Exs0VBQhgVq5ALTA+kYiNlTAuzGpdz0+NaIFDYzWQw	
		RSA	olV+xMGjg4RVxO/PKcPFVrbtfunsAMYW3Qqb5pmxDMQ	
gateway	MD5	RSA	87:84:69:ff:10:4a:01:fa:64:66:28:31:66:1b:3e:7e	
		ED25519	75:a3:c4:d2:38:12:0a:d0:00:d6:d2:ba:15:31:e4:67	
		ECDSA	9e:98:ec:c3:e3:fd:ef:de:99:03:5a:7e:50:d0:17:b5	
	SHA256	RSA	4dZpCZCeLBC+JLRqQBzbXWlX/dVLIVz5DYVFwHPa8	
		ED25519	9le6SKcTU1uvfPSYqvNZ4hHadibLbsb7IZ4yGQn8UJc	
		ECDSA	9Yxtz0/6BXykwap0EgRWKxfQ5sp0Rm9qxQGE+eOk2Y	