Introduction to High Performance Computing (HPC) – Session 1

using the "**C**omputational **S**hared **F**acility" (CSF)

Research Platforms, Research IT, IT Services

Course materials / Slides available from:
https://ri.itservices.manchester.ac.uk/course/rcsf/

CSF online documentation
https://ri.itservices.manchester.ac.uk/csf3/

Contact Research Platforms via the Connect Portal
https://ri.itservices.manchester.ac.uk/csf3/help/

Housekeeping

- Please let me know if you're leaving
  - 10am - 12:30pm (practical sessions 1, 2, & 3)
  - Lunch *approx.* 12:30pm - 1:30pm
  - 1:30pm - 4pm (practical sessions 4 & 5)
- 1-to-1 help is available if needed during exercises.
- Power adapters are available for the purposes of charging laptops, please be considerate of other users and *be careful of any trailing leads*.
- Got a question at any point? PLEASE ASK!!

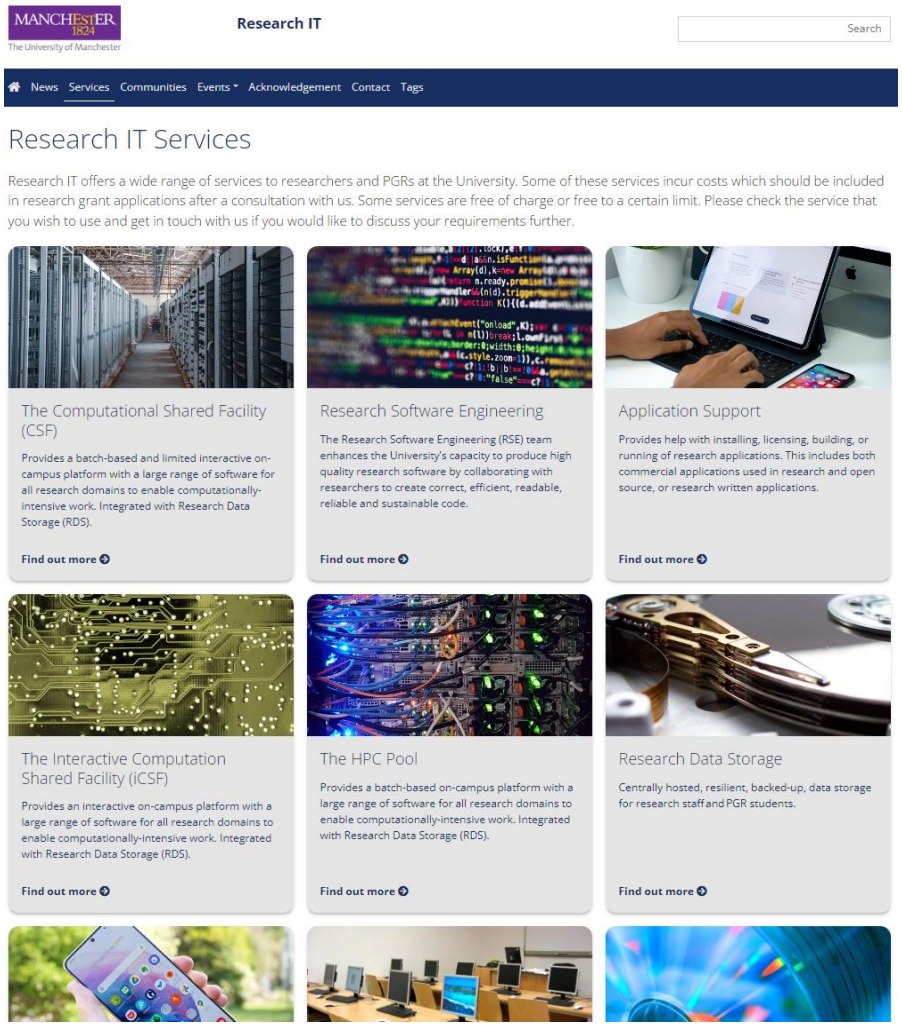
Feedback

- Your feedback is important to us!
- Please give feedback on this course
  - Quick form at https://goo.gl/forms/zfZyTLw4DDaySnCF3
    (choose "*Introduction to HPC (Using CSF)*")
  - Feedback is important to help us improve our courses
  - Records your attendance on the course

Course materials at https://ri.itservices.manchester.ac.uk/course/rcsf/

Who we are - Research IT

https://research-it.manchester.ac.uk/services/



What we'll cover today

- Brief intro to High Performance Computing (HPC) as motivation
- *Using* the University's HPC system - The "Computational Shared Facility" (CSF)
  - What the CSF is and what it can do for you
  - Logging in
  - Running work ("jobs") on the system
  - Different types of jobs (simple to advanced)
  - Using real applications
  - Doing the above in the practical sessions today

Who the course is for - everyone

- People new to HPC / research computing, or who just want to try the CSF
  - We'll introduce you to these topics and you'll try it out today
- Maybe your supervisor asked you to get a CSF account
  - We'll teach you how to use it
- Those who have used CSF already, but want to know more
  - Parallel jobs, job arrays, the batch system, ...

- ***Using*** the CSF is today's focus, so that you can use it effectively in your work
  - *not* theoretical aspects of HPC
    - but we'll explain some of the basics to help you make good use of the resources
  - *not* parallel software development or version control
    - But we'll show you how to run high-end parallel applications
  - *not* Linux installation / administration
    - but we'll cover the basic Linux commands needed to use the CSF
  - *not* the specifics of the software you plan on using
    - ask your PI/supervisor for help with that

Invest a little time now, get results much faster!

WHY & WHAT …

High Performance Computing: why use it & what is it

Motivation: Why use HPC (and the CSF)?

- Some (most?) research computation not suitable for your desktop/laptop
  - Takes *too long* to run
  - Needs more memory
  - Uses *too much* disk/storage space
- Use advanced, centrally-managed, UoM hardware
- Eventually, use regional / national supercomputers

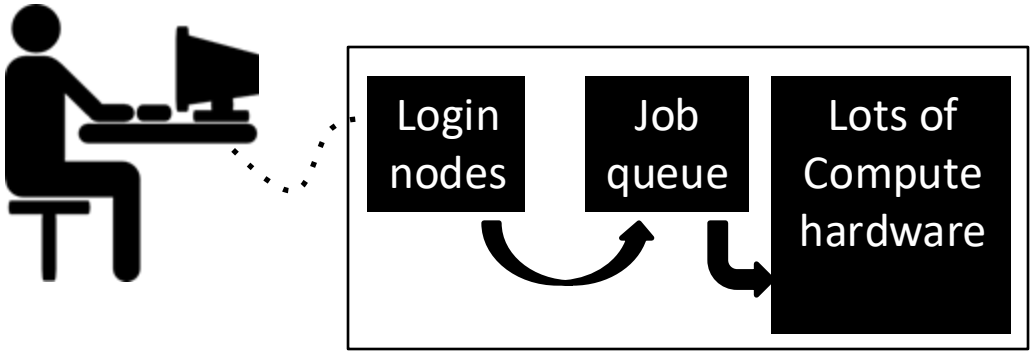
Do not let the size/capacity/power of your computer dictate the size and complexity of the models/simulations/systems/problems you are solving!

What is HPC?

- insideHPC.com
  - *High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.*
- HPC systems are usually a cluster of *compute nodes* (with some extra items such as *login nodes*, storage, networking)
  - The CSF fits this description

A new way of working!





Running on a desktop

- You can fire up a GUI, run an app immediately. BUT:
- Got enough memory, cores, storage?
- Need to keep the PC to yourself (public cluster PC?)
- For several days?!
- Only one "job" (simulation, analysis) at a time?

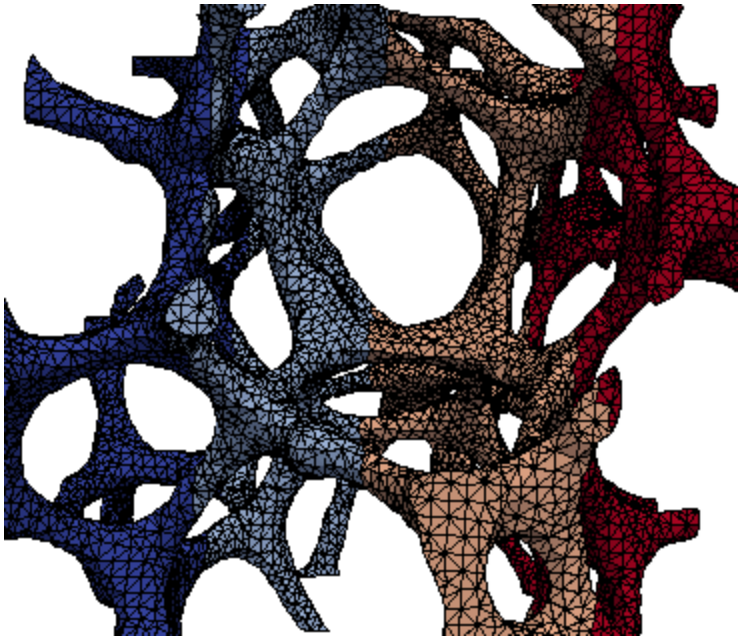
vs

Logging in to the HPC system

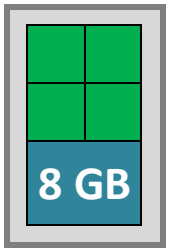
- Submit "**jobs**" to the queue
- Jobs wait until selected to run
- Jobs run on *high-end* hardware (lots of cores, memory, disk, GPU)
- Jobs run safely for days
- Many jobs can run at once
- Can log out any time (jobs still run)
- Log in to check on progress, get results

HPC Example: Finite Element Analysis

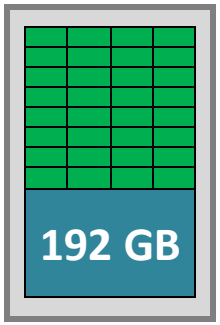
- Perform stress analysis on 3D mesh
  - The app splits the *input* into chunks
  - It performs calculations on chunks, in *parallel*
    - Faster and/or larger problem size



Source: Professor Paul Mummery (MACE) using ParaFEM software http://parafem.org.uk

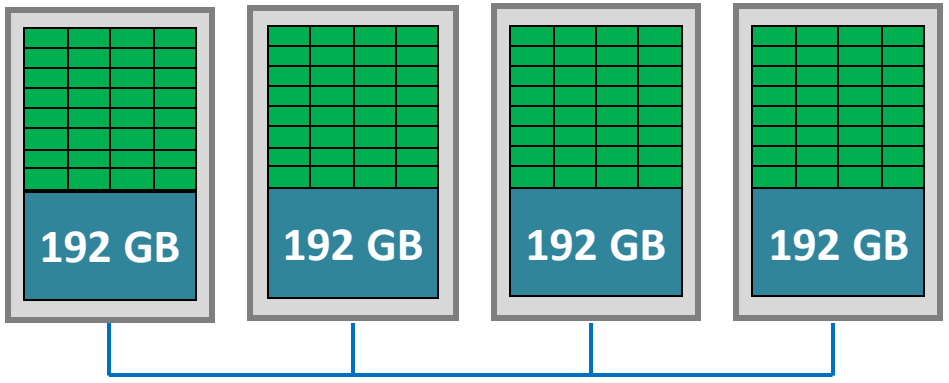


Desktop: 4 cores, 8GB RAM.
8 days to complete



Single HPC *compute node*: 32 cores, 192GB RAM
~2 days to complete

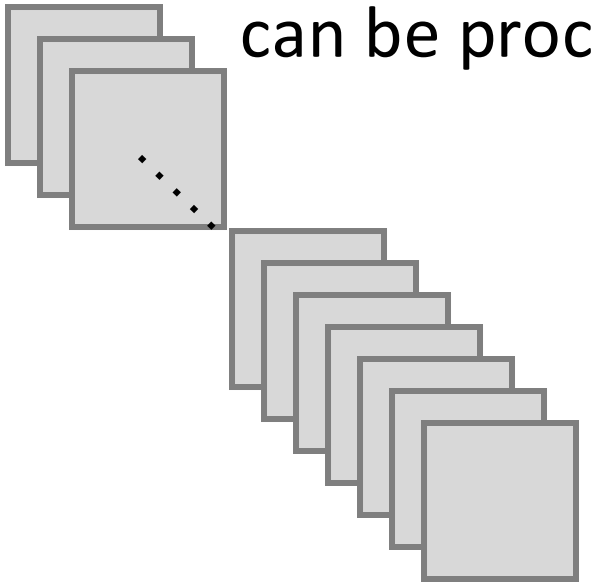
Some apps can use more than one *compute node*



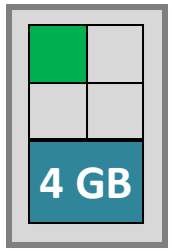
Multiple HPC *compute nodes*: 128 cores, 768GB RAM
~0.5 days to complete

"HTC" Example: Image Analysis

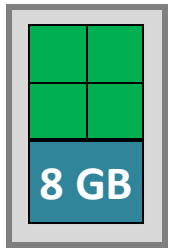
- High *Throughput* Computing
  - Not all s/w is "HPC" / parallel
  - But you might have *lots* of data
  - EG: Each image takes 1hr to process (and are independent - can be processed in any order)



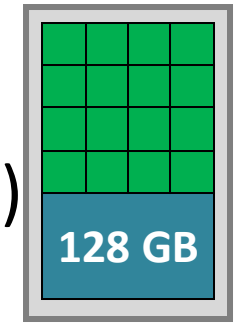
Example: 10,000 image scans to be analysed by an image processing application. Each image takes *1 hour* to process.



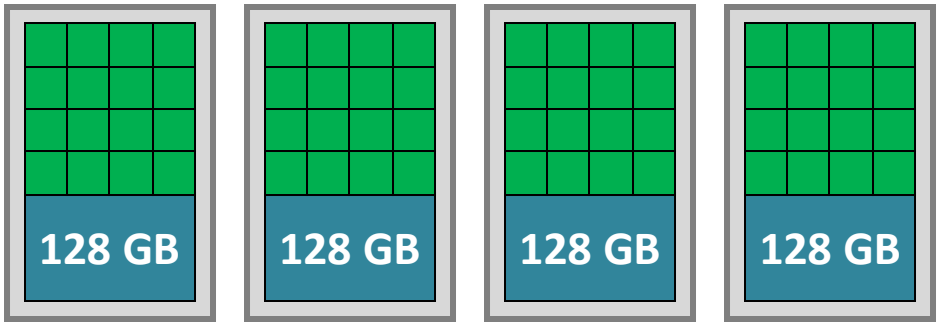
Laptop: 1 copy of software running on 4-core CPU.
Over 1 year to complete!!



Desktop: 4 cores, 4 copies of software running.
~100 days to complete!



Single HPC compute node: 16 copies of software running.
~26 days to complete



Multiple HPC compute nodes: 64 copies of software running independently.
~6 days to complete

What we'll be using today - the "CSF"

- Q: who has used the Computational Shared Facility (CSF) before?
- CSF3 current config:
  - A large Linux cluster system
  - 28,856 CPU cores (AMD Genoa and Intel Xeon CPUs)
  - 204 Nvidia GPUs (76 x A100, 88 x L40S, 32 x H200)
  - Got big datasets to process? Can run large-memory jobs (up to 4TB RAM)
  - (we'll cover these details throughout the course)

BUT, you *do not* need to be running huge parallel jobs, or be a Linux / HPC expert, to use our systems and to benefit from the CSF

Who can use the CSF?

The following info is mainly for people who may want to "buy in" to the CSF. Your PI/supervisor or School may well have already done this! **If interested, ask us at the end of the course**.

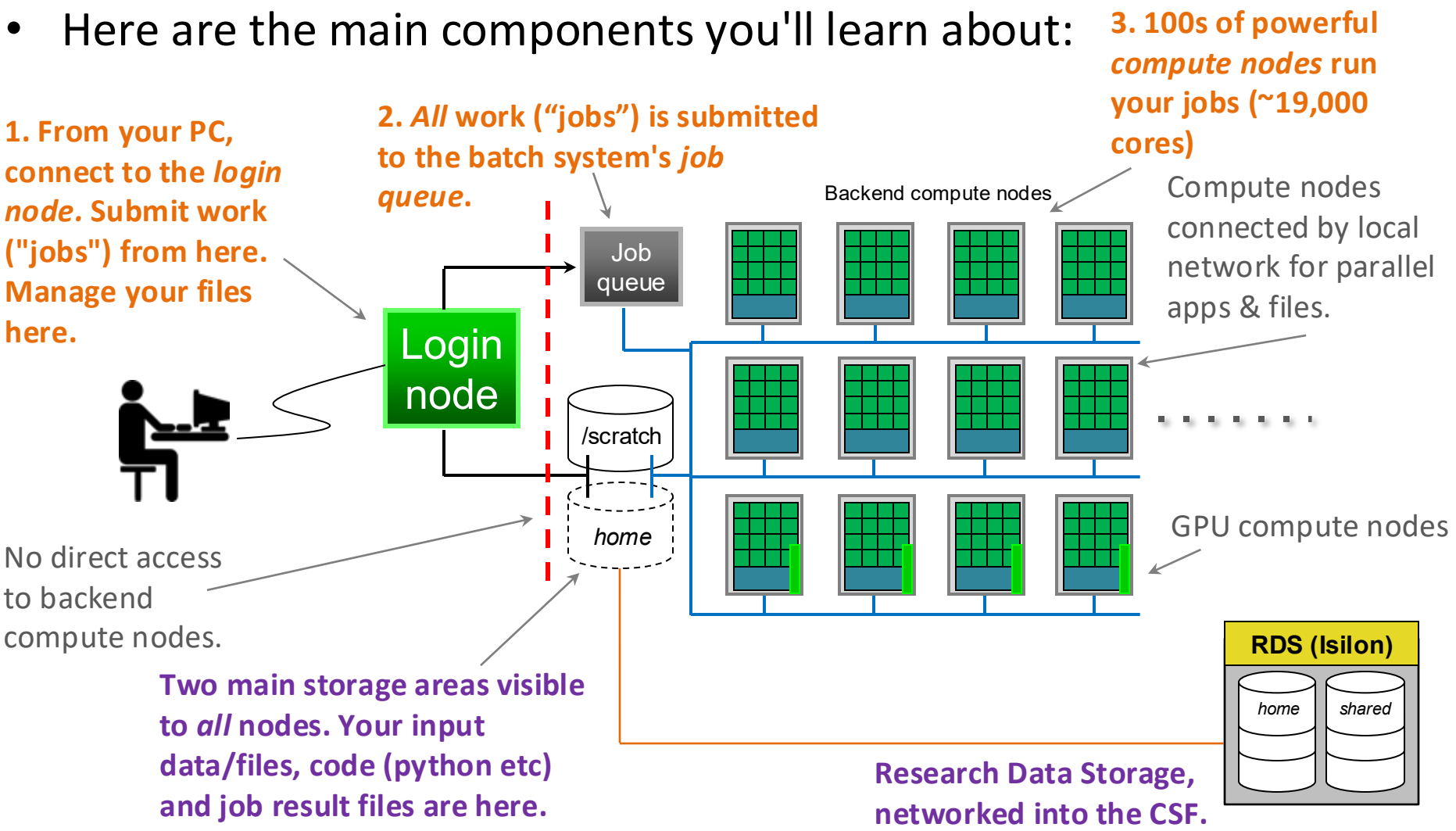
- CSF uses a *shared* funding model
  - Researchers/academics/schools *contribute financially* to buy compute hardware
  - All h/w pooled so that all users can access the h/w
  - H/w *not* associated with individuals so it can always be in use as long as there are jobs to run!
- The time it takes to run all of your jobs depends on the size of the contribution with which you are associated
  - A research group that contributes more will be able to get more jobs done *sooner – they have more "throughput"*
  - Managed automatically by the *batch system* – you just submit jobs!
- Some limited 'free at the point of use' access for non-contributors

CSF: THE BASICS…

Hardware, OS, logging in, security, home filesystem, copying files, Linux, GUIs

What is the CSF? (more details)

- **C**omputational **S**hared **F**acility
- A *batch* compute *cluster* to run your "jobs" (simulations, analysis,...)
- Here are the main components you'll learn about:



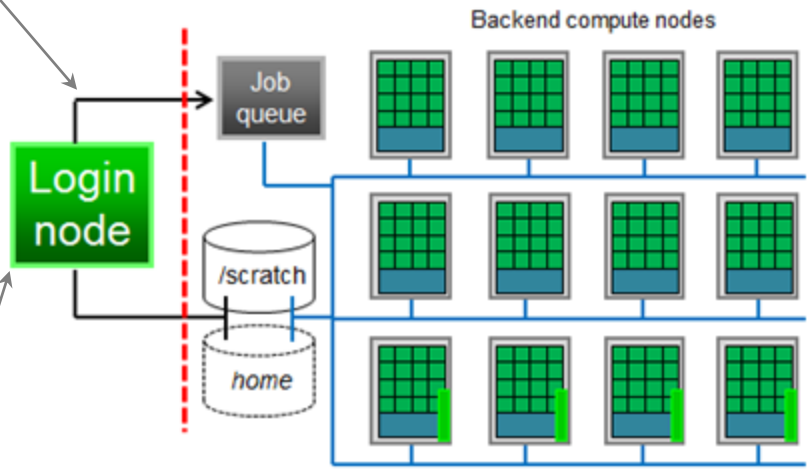
Login Nodes

- We mostly only interact directly with the (three) CSF login nodes
  - Approximately 500 compute nodes
  - Far too many for *you* to find a free compute node to use
  - Instead: connect to the login node, let the *job queue* find a node

Key concept!

- Do: Submit work ("*jobs*") to be run
  - No direct access to compute nodes
  - Submit jobs to the *job queue*.
  - The system will run your work on available *compute nodes* meeting your requirements.

- Do not: run applications on the login node
  - Shared by all users, not much memory
  - For *lightweight* tasks (job submission, file transfer, …)

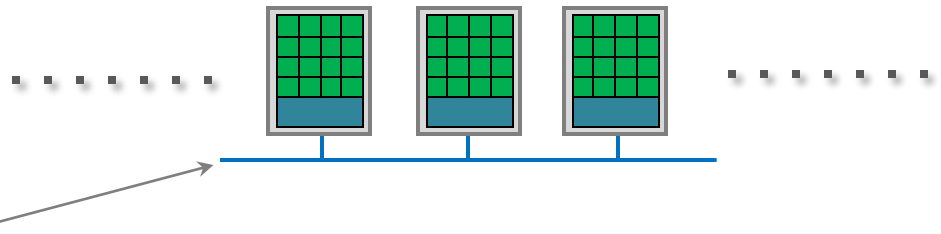


Compute Nodes

- The CSF is a cluster of powerful "*compute nodes*"
  - Can think of the compute nodes as ***very* high-end** PCs where your simulations / data analysis / ML training … will run
  - Different types of compute nodes available to suit your requirements (high mem, GPU, or a standard / default node)

- Multi-core CPUs (e.g., 32, 168 cores)
- Lots of RAM (e.g., 512GB, …, 2TB, 4TB!)
- Network (possibly fast *InfiniBand* n/w)
- O/S (Linux)
- Local disk (for temp files)
- Maybe a GPU

- Not all nodes have the same hardware.
- You can specify certain requirements for your job – e.g., "I need a node with a GPU." "I need extra RAM."



Local network connecting compute nodes. This allows:

- some applications to use more than one node (e.g., for big data / large simulations)
- all nodes to see all of your files - so you don't have to copy files to the compute nodes

Some pictures of the CSF



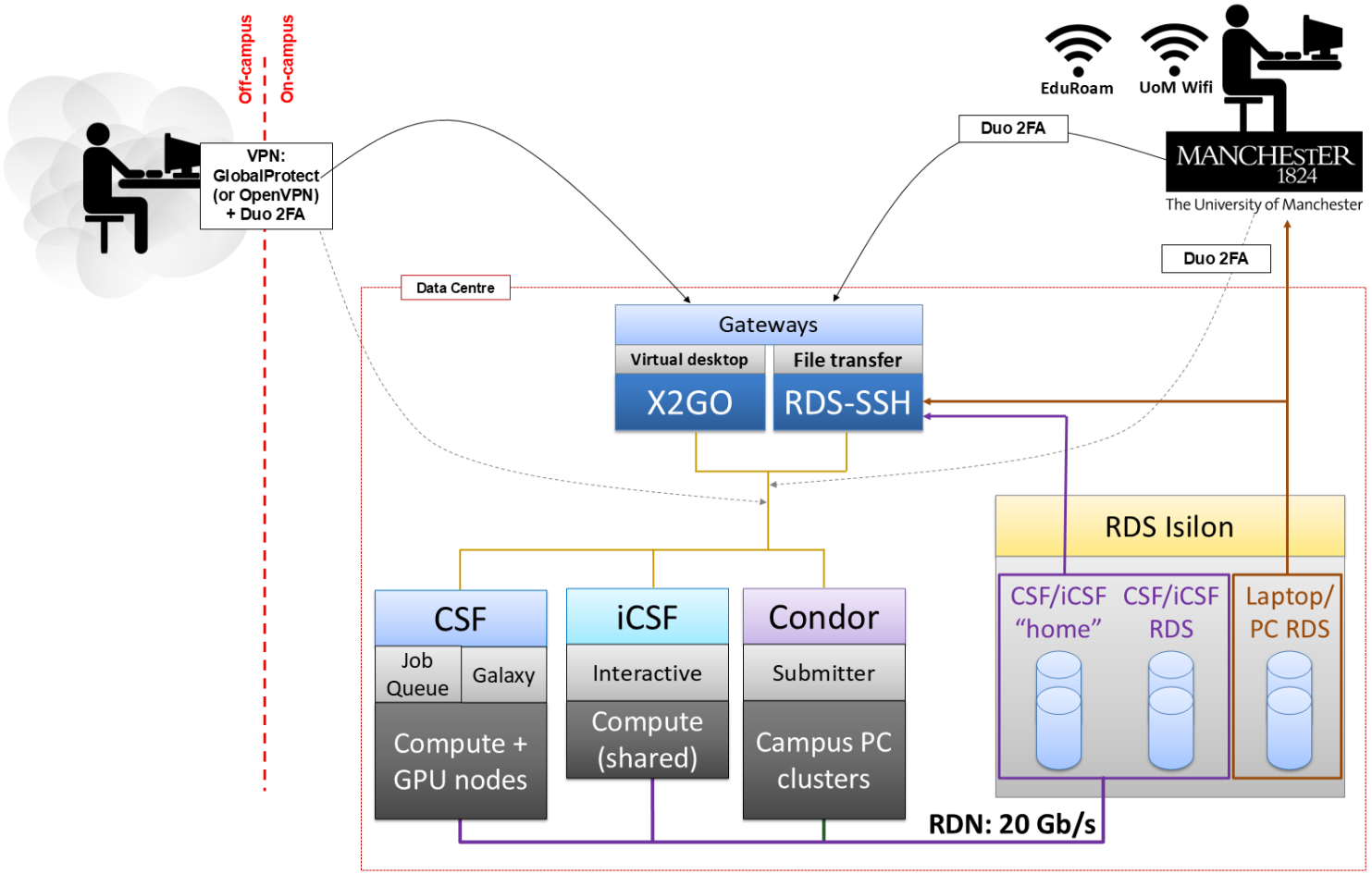




Other Benefits of the CSF

- Over 150 software applications already installed, (and compilers and libraries if writing your own s/w)
  - Abaqus, Ansys, Gromacs, Bowtie2, Gaussian, LAMMPS, MATLAB, OpenFOAM, PyTorch, StarCCM, TensorFlow, VASP + many more
- Backed-up file storage (no more risky USB disks!!!!)
- Leave computational work running for days without needing to be logged in
- 28,500+ cores & 200+ GPUs currently in the system
  - If a compute node fails, we can remove it for maintenance without you noticing. What if your laptop / workstation fails?
- Dedicated support team
  - Connect Portal tickets come directly to us

Part of the *Computationally Intensive Research* Ecosystem

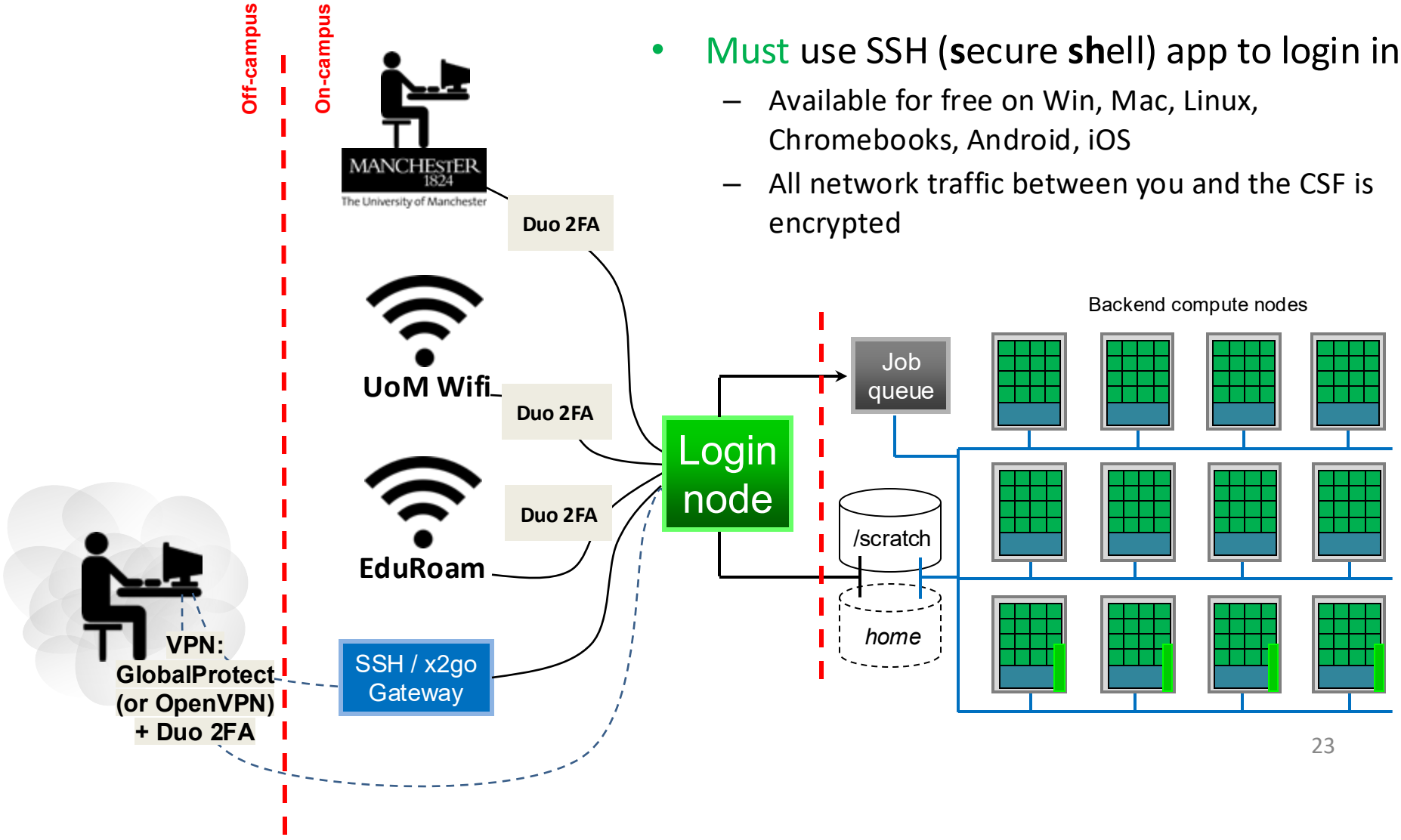


Logging in …

Let's get access to the system

Where can I log in from?

- Must use SSH (**s**ecure **sh**ell) app to login in
  - Available for free on Win, Mac, Linux, Chromebooks, Android, iOS
  - All network traffic between you and the CSF is encrypted



Security (1)

- The CSF has a private (campus-only) IP address
  - Firewall also controls connections to and from the system

- **When you are on-campus**
  - Connect from any PC/laptop with a wired connection, or UoM WiFi, or EduRoam WiFi
  - Does not matter if using GlobalProtect or not, but you will *always* be asked to authenticate using your 2FA (DUO) device (see later)

- **When you are off-campus**
  - **First**, sign-in to University GlobalProtect VPN + DUO 2FA
  - Then can login as normal to CSF (won't have to DUO 2FA again)

- Further documentation: https://ri.itservices.manchester.ac.uk/csf3/getting-started/connecting/

Login Overview

- We'll soon do exercise 1 to login to the CSF
  - The exercise sheet provides all of the steps, including app installs
  - There are also reference slides here giving all of the details
- Access the CSF from a PC / laptop using a Secure Shell (SSH) app
  - Sometimes called a "terminal"
  - There's no web-based or other fancy GUI on the CSF – use the Linux "command line"
- On **your** PC/laptop:
  - Windows users will install a free terminal+ssh app called MobaXterm
  - Mac users will install a free app named XQuartz, then use the built-in Terminal app and ssh command
  - Linux users will use the built-in terminal and ssh command

- In all cases you will need:
  - Your UoM IT username (like *mabcxyz1*, NOT your email address!)
  - Your UoM IT password (same as used for email, blackboard etc.)
  - Your DUO 2FA device (most likely your mobile phone.)

- Everyone will use the address of the CSF:
  `csf3.itservices.manchester.ac.uk`

DUO 2FA (when on-campus)

- Note: When on-campus, after you enter your password during CSF login, a*ll* login methods will then ask you to do DUO 2FA:

```
Duo two-factor login for mabcxyz1

Enter a passcode or select one of the
following options:

 1. Duo Push to +XX XXXX XX7890

Passcode or option (1-1): 1
Success. Logging you in...

(the Message of the Day is now displayed)

[mabcxyz1@login1[csf3] ~]$
```

Type **1** (and press Enter ↵) in your ssh app to generate a DUO *push* to your device.

Then accept the push on your device.

You are now logged in :-)

What you see once you've log in

- The CSF uses Rocky Linux (c.f. Red Hat EL)
  - Command line – requires the input of commands, can be a little scary at first to new users
  - A welcome *Message of the Day* - announcements
  - The system awaits input/commands from you at a *prompt* (after you've *logged in*):

```
[username@login1[csf3] ~]$
[username@login2[csf3] ~]$
[username@login3[csf3] ~]$
```

Type Linux commands at "the prompt"

- Learning Linux commands (more later):
  - https://www.chm.bris.ac.uk/unix/

Security (2)

- It is NOT permitted to share your CSF account
- CSF uses your **IT password** – i.e. same as needed to access UoM email, Blackboard and so on ...
  - NEVER share it with ANYONE, including IT staff and your supervisor
  - Forgotten it? You can reset it via the IT Account Manager. Will affect *all* systems that require it.
    - https://iam.manchester.ac.uk/
- Reminder: Other general safety measures
  - Install a virus scanner https://www.itservices.manchester.ac.uk/cybersecurity/advice/virusprotection/
  - By aware of phishing emails https://www.itservices.manchester.ac.uk/cybersecurity/advice/phishing/

PRACTICAL SESSION 1

Logging in

Practical Session 1 – Logging in

- Exercise 1 sheet (pdf) available at: https://ri.itservices.manchester.ac.uk/course/rcsf/
- **See also the reference slides after this one for Windows, Mac and Linux users.**

- Tip 1: During login, you'll be asked for your password. **Type it carefully!**
  - The cursor will *not* be displayed.
  - You won't see any characters or ***s as you press the keys.
  - **But it IS noticing what you type!**

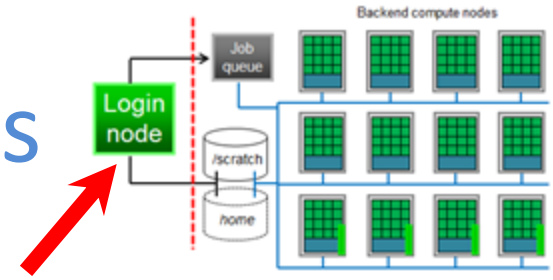
- Tip 2: Once logged in to the CSF, you can run Linux commands "at the prompt".
  - Many Linux commands do not display anything after you've run them.
  - This is usually a good sign – it often means your command worked.
  - If you've got something wrong, it will *usually* tell you via an error message.
  - You can't do any harm. Typos / incorrect commands won't run, but also won't do any damage! Just try again.

- By the end of this exercise, everyone must have successfully logged-in to the CSF!
- PLEASE ASK FOR HELP IF YOU RUN IN TO ANY PROBLEMS – WE ARE HERE TO HELP!

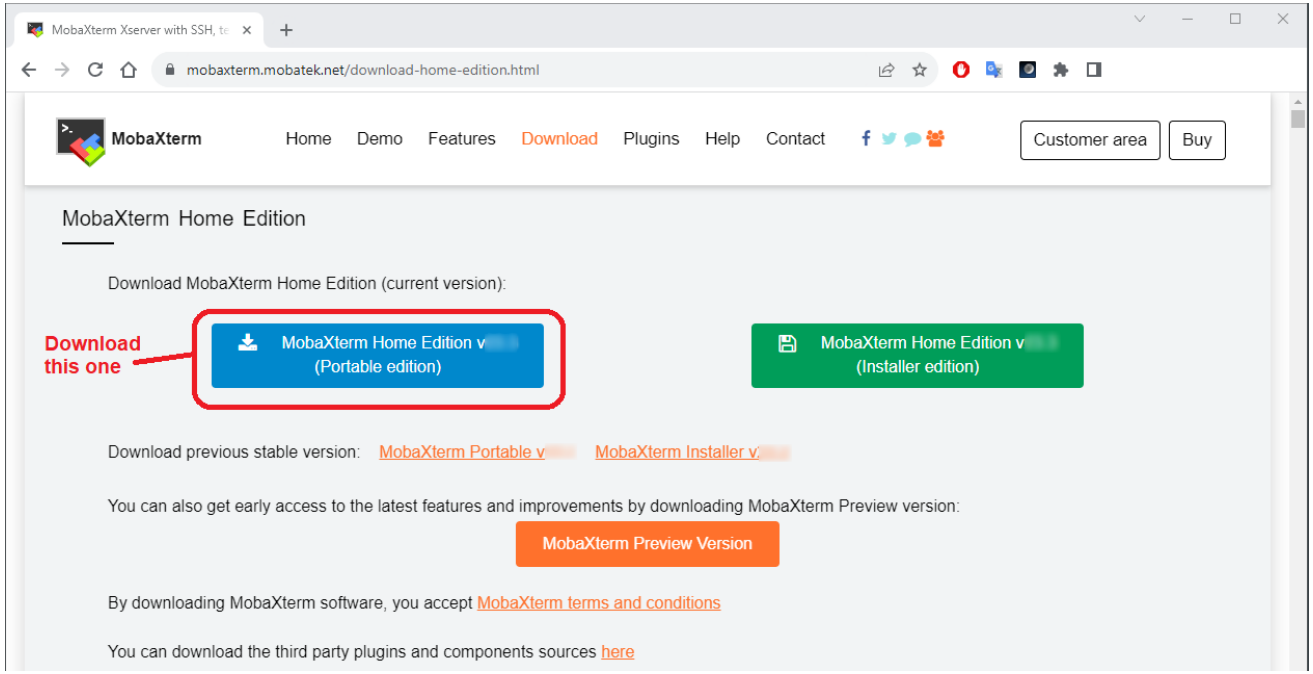
Exercise 1 – Logging-in Reference Slides

See also the exercise sheet for necessary steps

Connect to CSF from Windows



- **Windows users** need to install a free *terminal* app called **MobaXterm**
- https://mobaxterm.mobatek.net/download-home-edition.html
  the Home edition (portable edition) does *not require* Administrator rights - just *extract* the small .zip file in your P-Drive or Downloads or USB stick for example.



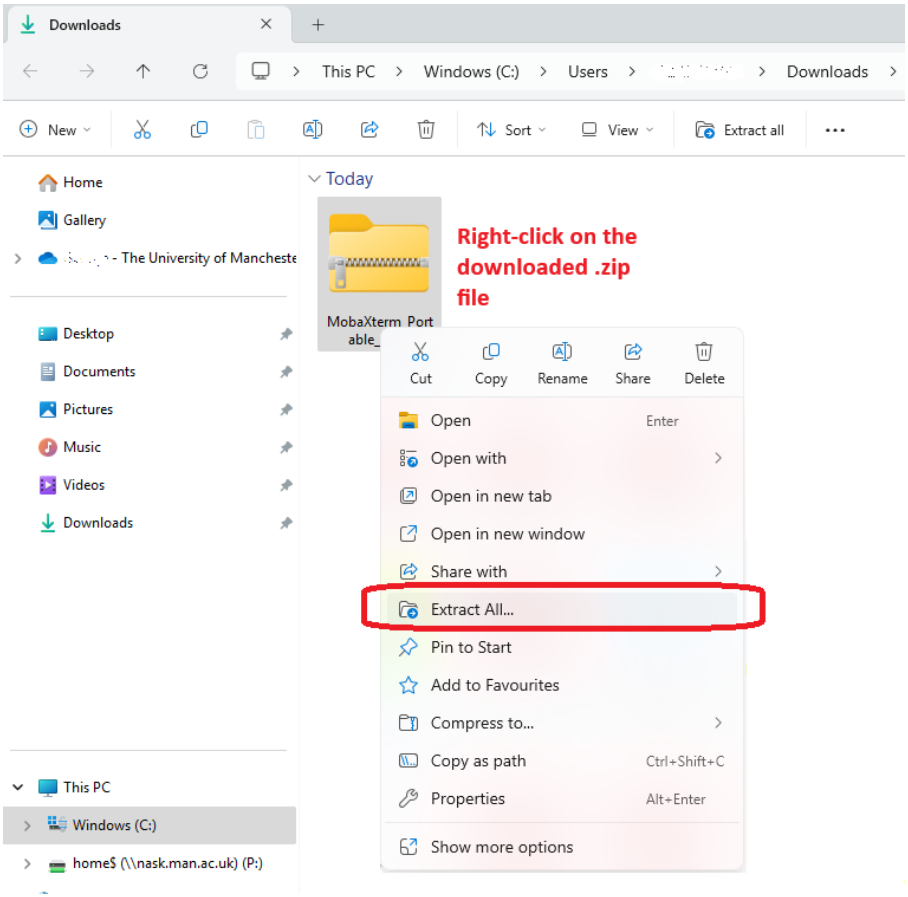
1. Download using the blue box.
2. Once downloaded, *right-click* on the .zip file and select:

   "**Extract all** …"

   This will *unpack* the .zip file to a folder.

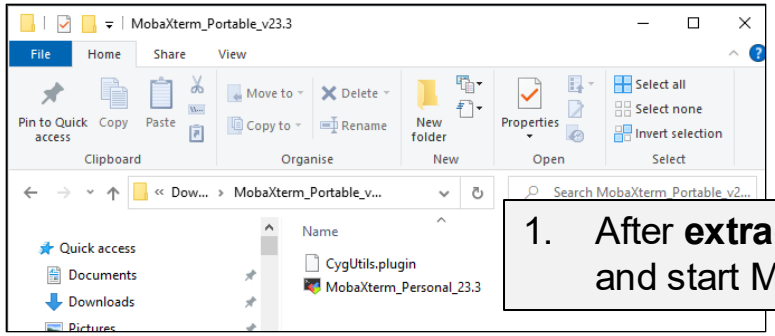
Install the MobaXterm app on your laptop/PC

- You **must** right-click on the downloaded .zip file, then "Extract all..."
- Simple double-clicking on the .zip file to browse the contents does NOT install it correctly.

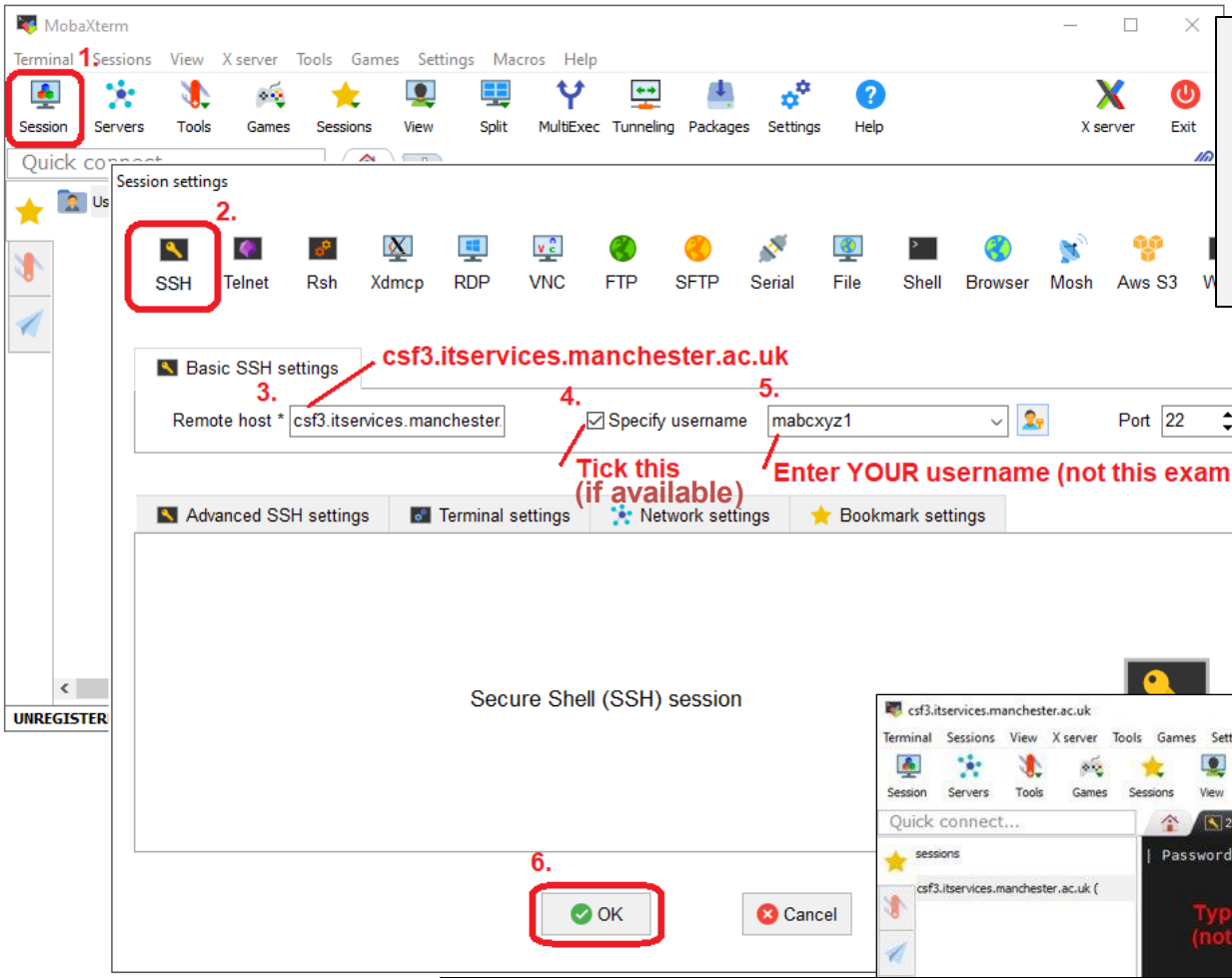


MobaXterm "Session"

(username saved in the session setup)



1. After **extracting** the .zip file, go into the MobaXterm_Personal_v*xy.z* folder and start MobaXterm_Personal_*xy.z* (double-click on the icon)



2 (1-6). Create a "Session" which saves the CSF's details along with *your* **username**.

This is needed to make file drag-n-drop work (see later.)



3. This will then start to log you into the CSF – it will ask for your password. Type carefully!

4. See slide about 2FA – you may be asked for DUO after your password

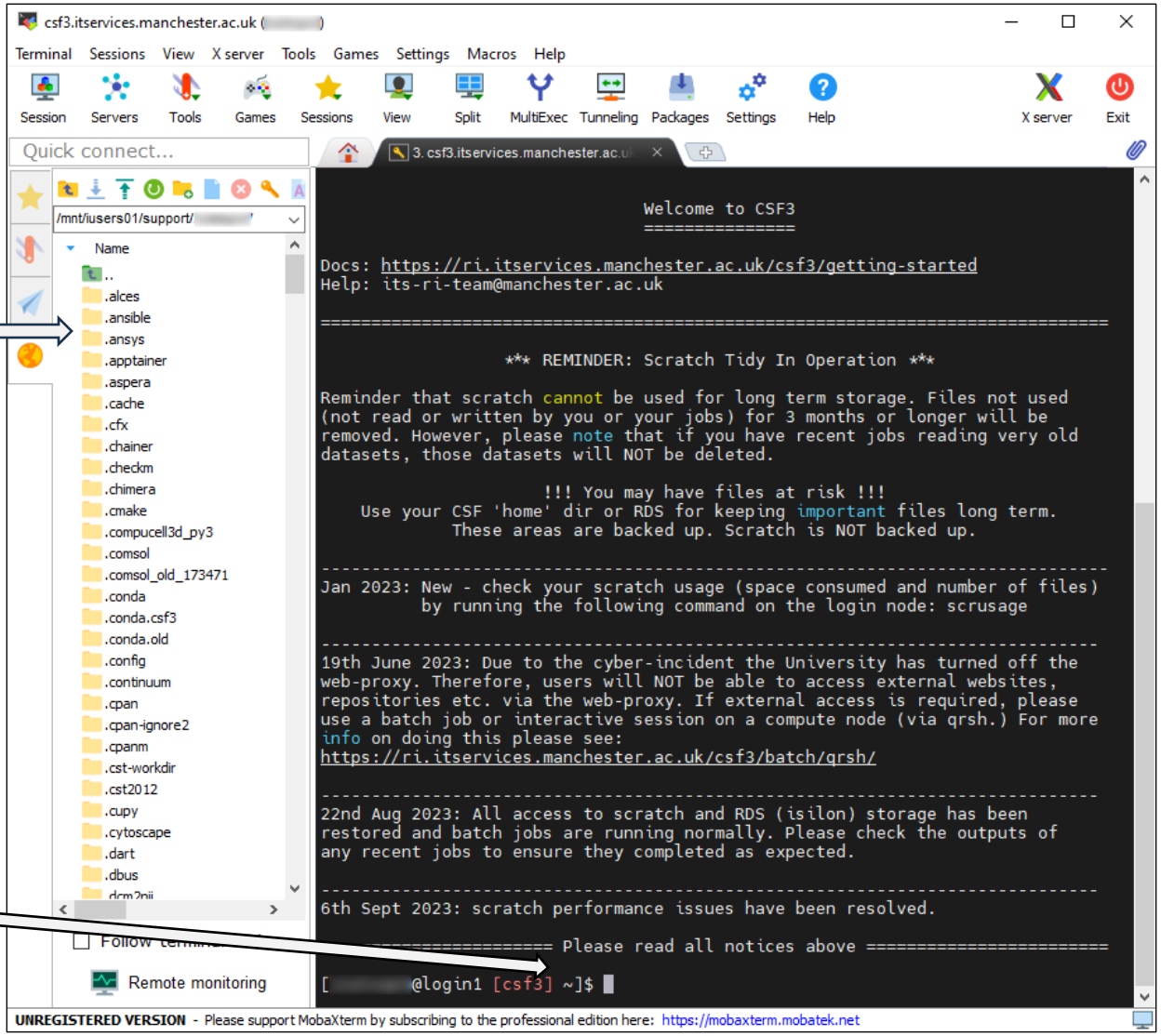


If asked to save your password, we recommend you say "No", for security.

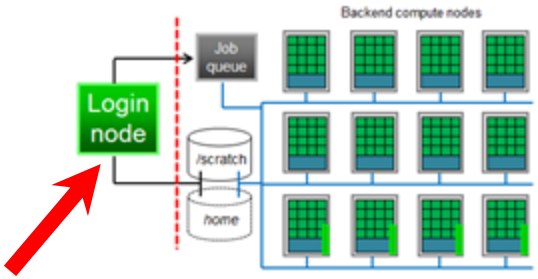
Drag-n-drop file browser for upload / download

(new users won't have as many items in the list!)

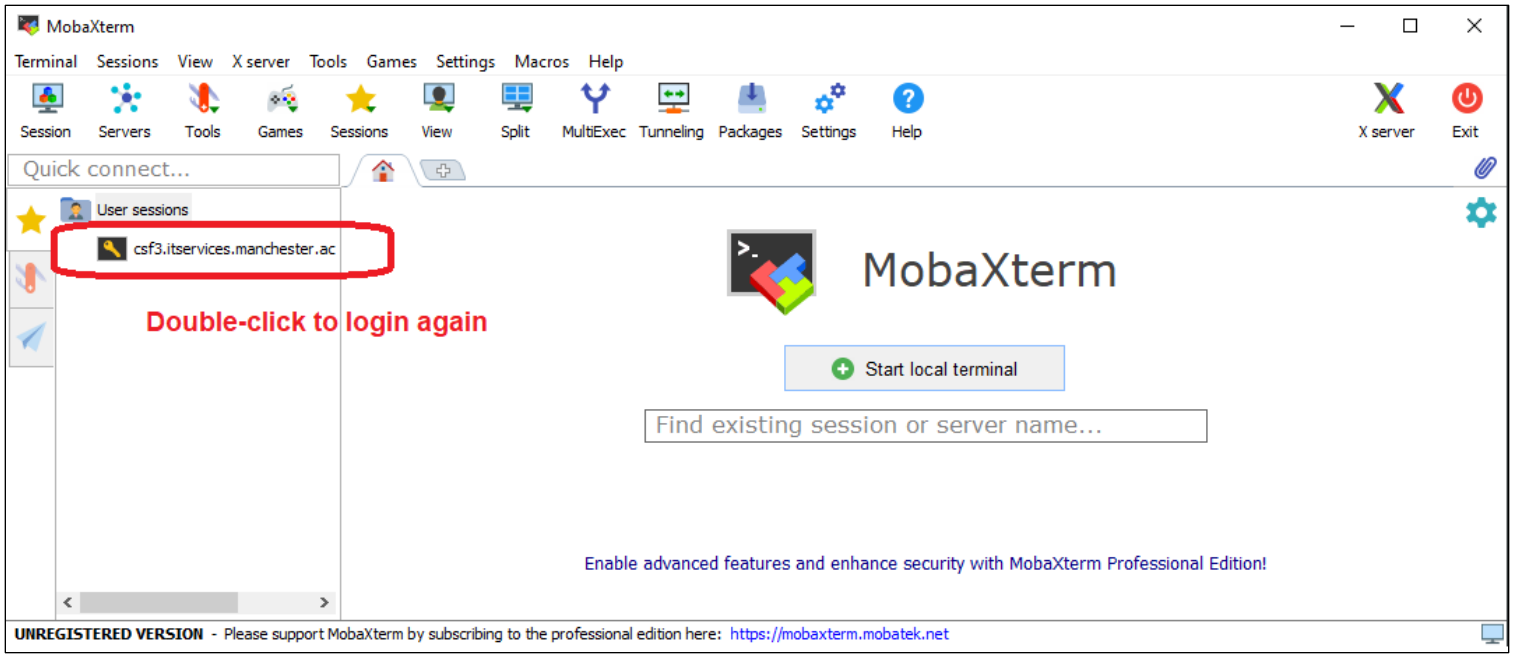
We're on (one of) the CSF login nodes. Any commands you use will be typed "at the prompt", which shows your username and current directory (folder.)



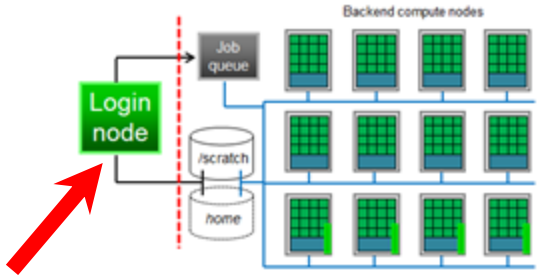
Next time you want to login to CSF from Windows



- Just double-click the csf3 "session" in the list of "User sessions"
- The CSF details are saved in the "session"
- (this also makes the file browser work, for drag-n-drop file transfers.)



Connect to CSF from a Mac



- **Mac users** - have a *terminal* application by default
  - **You will first need to install X-Quartz first**

    https://www.xquartz.org/ (install, then you should **reboot your Mac**)
  - Start a *Terminal* app (possibly from Go > Utilities > Terminal) and then type the following command:

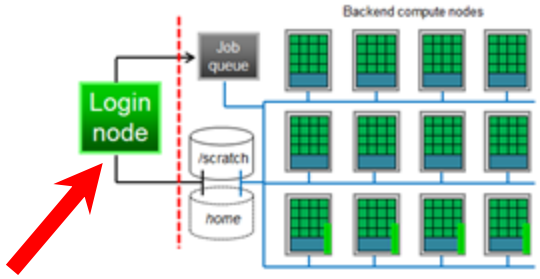
```
ssh -Y username@csf3.itservices.manchester.ac.uk
```

UPPERcase Y

Central IT Services username.
Answer 'Yes' to continue *if* asked.
Enter central IT password when asked (same as for email)

- Finished using CSF? Log out with: `logout` or `exit`

Connect to CSF from Linux



- **Linux** users - have a *terminal* application available by default
  - Start a Terminal (e.g., MATE terminal) and type the following command:

```
ssh -X username@csf3.itservices.manchester.ac.uk
```

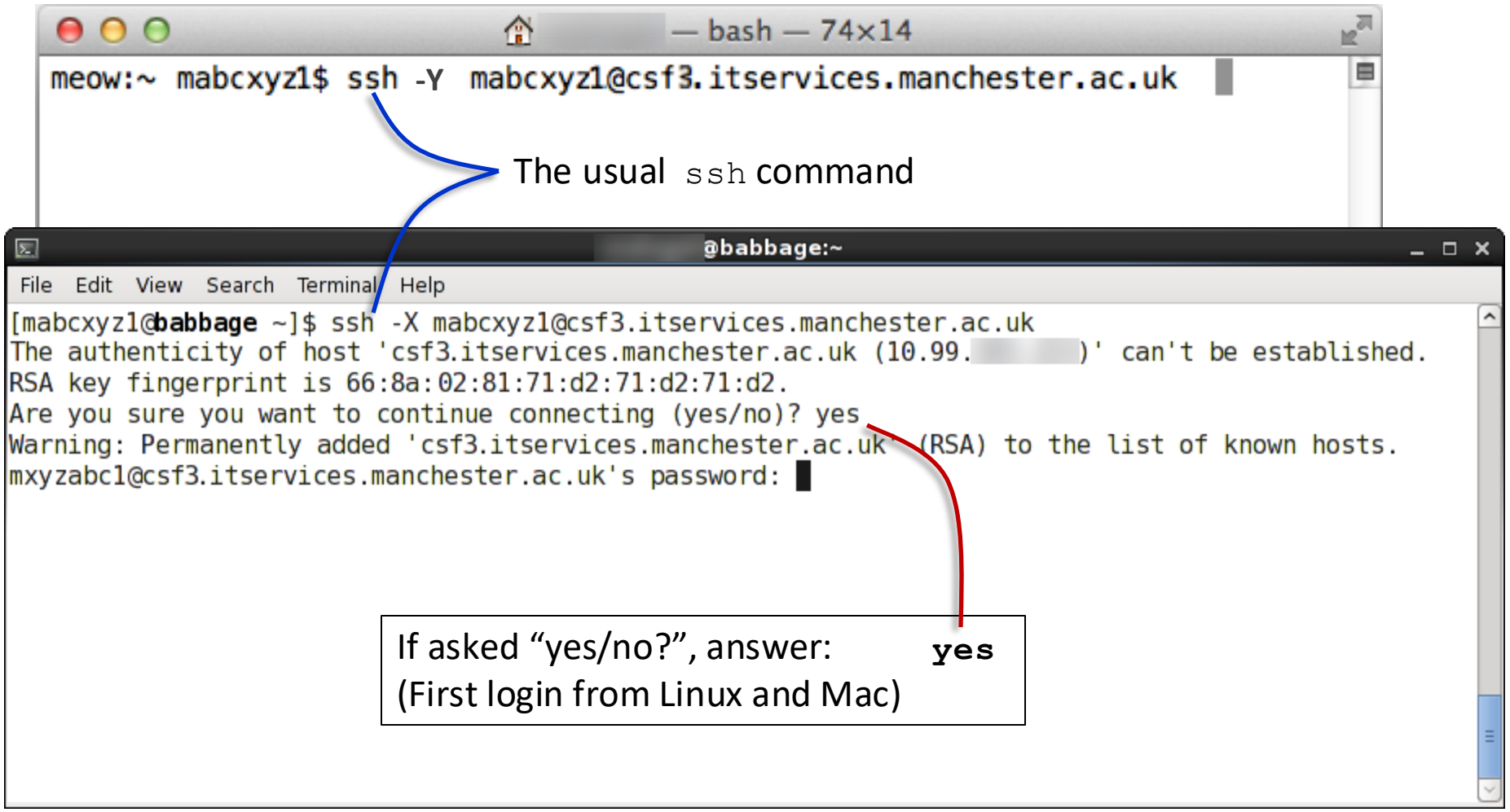
UPPERcase X

Central IT Services username.
Answer 'Yes' to continue *if* asked.
Enter central IT password when asked (same as for email)

- Finished using CSF? Log out with: `logout` or `exit`

Linux / Mac Terminals

- You might be asked a question upon first login, before it asks for your password. It is safe to answer "yes".

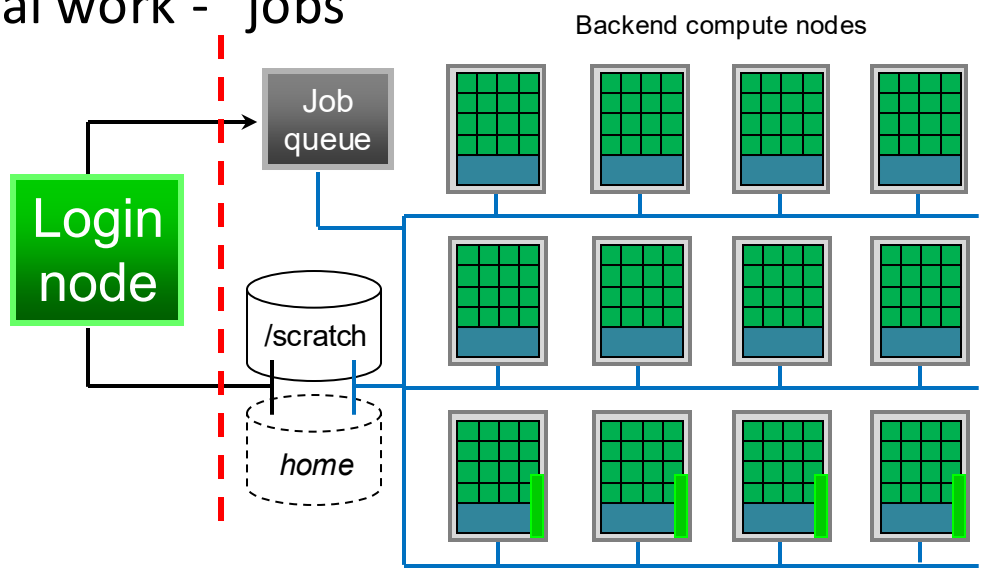


RUNNING JOBS

Doing real work on the CSF

Jobs, Jobscripts and the Batch System

- We want to do computational work - “jobs”



- You decide:
  - Which program(s) to run
  - Which resources it needs (#cores, CPU type, memory, GPU?)
  - How much time the job will need to complete its work
  - Which of your folders ("directory") to run the job in
- You'll put these requirements into a *jobscript* file
- Then submit your *jobscript* to the batch system ("Slurm")
- Slurm decides *when* the job runs and on which compute node(s). It ensures you get all of your requested resources.

Reminder: The login nodes

- Do *not* run computational work here:
  - Not enough cores
  - Not enough memory
  - 100+ users connected, so running work causes serious problems
- You *can* do the following:
  - Transfer files on and off the CSF
  - Set up and submit your jobs (covered in next few slides)
  - Basic data processing/viewing
- Computational work running on the login node will be killed without warning!

Creating a Jobscript file (1)

- You need to be able to create a small text file to describe your job
- Run `gedit` on the CSF login node - a simple text editor
  - Creates and saves the file directly *on the CSF*
  - Is similar to notepad (other Linux editors: xnedit, nano, emacs, vi – only use these last two if you know how to use them.)
- Once logged in to the CSF, run one of these:

```
gedit &
gedit filename &
```

'&' allows you to carry on using the command-line. Try it without to see.

  - (start a new file or edit an existing one. Then save it.)

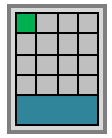
Can I Write Jobscripts on Windows?

- A warning about Windows text files (EG: in *notepad*)
  - There's an inconsistency over the (hidden) *end-of-line* characters in text files:
    - Windows: CR (carriage return \r) + LF (line feed \n)
    - Linux/Unix: LF (line feed \n)
  - The extra CR from Windows is a problem in jobscripts.
  - It causes `sbatch` to reject your job immediately.

```
sbatch: error: Batch script contains DOS line breaks (\r\n)
sbatch: error: instead of expected UNIX line breaks (\n).
```

- Solutions
  - Use `gedit` on the CSF login node (writes Linux text files)
  - Or use *notepad*, upload, then run `dos2unix myfile.txt`
    - Use `dos2unix` only on jobscripts – will break other files.
    - Do not come to rely on this – it is too easy to forget to do it – use gedit directly on the CSF!

A simple Jobscript – *Serial* (1 core)



`#!` on first line only (a special line)

First line indicates we use the *bash* scripting language to write our jobscript.

`myjob.txt`

```
#!/bin/bash --login

#SBATCH -p serial
#SBATCH -n 1
#SBATCH -t 5

# Let's do some work
date
hostname
sleep 120
date
```

`#SBATCH` indicates a **batch system parameter** to specify our job requirements. We'll use various combinations of these.

`-p` (`--partition=`) think of this as the queue, for serial (1-core) jobs in this case.

`-n` (`--ntasks=`) number of cores, which must be 1 for serial jobs (**optional line!**).

`-t` (`--time=`) maximum "wallclock" time the job is allowed to run for. Various formats. 5 is 5 minutes. 4-0 would be 4 days (0 hours).

`#` lines are just comments - anything on the line after it will be ignored.

Actual Linux commands we run in our job. They will execute on a compute node.

Submit Jobscript to Job Queue

- Submit the jobscript from the login node with:

```
sbatch jobscript        # EG: sbatch myjob.txt
```

- You will be given a unique *JobID* (currently a 7-digit number). Can use this in other commands.

```
Submitted batch job 5980521
```

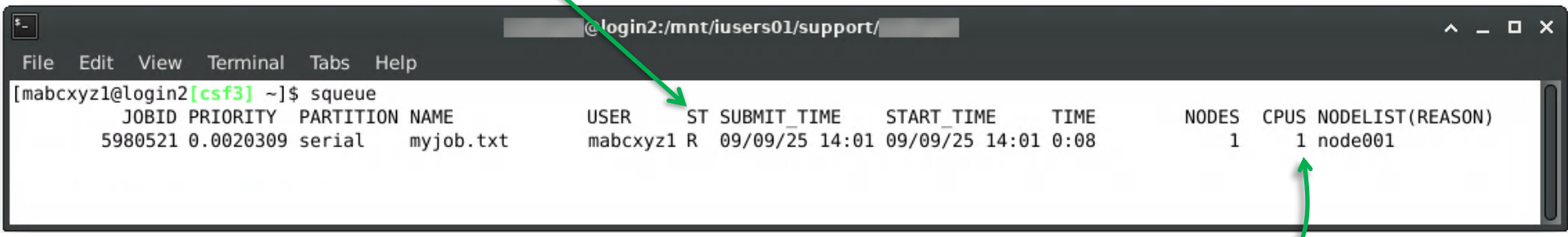
- You can then:
  - carry on with other work
  - check on your job queue and the jobs
  - submit more jobs, without disturbing previous jobs
  - log out of the CSF and your jobs will still run

Check status of your jobs

- To see your job(s) in the batch system, run:

  `squeue`

  `ST` (state) is either pending (`PD`), running (`R`) or failed (`F`). These are the most common states.



  `CPUS` is the number of cores (1 by default - a *serial* job)

Serial (1-core) Job Properties

- Our simple example job:
  - Serial (only **1** core is used)
  - Standard memory (*no* `#SBATCH` line asking for *more*)
    - Our job gets ~5GB RAM to work with
    - The serial partition contains node with Intel CPUs, 5GB/core
  - We said the job would need no more than 5 minutes runtime (max permitted is 7-0 i.e., 7 days)
- Standard serial jobs will be placed on: Intel nodes
  - Other nodes exist for other job types
  - The batch system looks for a free core (in a list of compute nodes in the "serial" partition).
  - That core is assigned to your job, for the requested wallclock time.

Optional Serial Job Resources

https://ri.itservices.manchester.ac.uk/csf3/batch-slurm/serial-jobs

Jobscript flag(s)	Description
`#SBATCH -C haswell`	5GB/core Intel "haswell" CPU. Otherwise Slurm chooses.
`#SBATCH -C skylake`	6GB/core Intel "skylake" CPU. Otherwise Slurm chooses.
How to run serial jobs on the AMD 168-core nodes	
`#SBATCH -p interactive`	8GB/core AMD "Genoa" CPU. Despite the name, **can be used for batch jobs**. **Max permitted wallclock is 1 hour**. Adjust your "`#SBATCH -t`" line. We'll cover interactive jobs later.

- **Max permitted wallclock limit is 7-days runtime limit. You must say how long.**
- Our simple jobscript did *not* use any of the above. Not needed in most cases.
- If you limit a job by *node-type* it may wait longer in the queue.
- You will see that the example jobscripts in the exercises have:
  `#SBATCH --reservation=course`
  - **Only for use today** (we have reserved nodes on a teaching day.)
  - **Remove** if practicing after today (jobs will fail to submit otherwise.)

So where did my results go?

- If `squeue` returns no output - means job has finished!
- Where are my results? Three possibilities:
  1. If your app usually prints to screen: output captured to a text file called ***slurm-JOBID*.out** where
     - *JOBID* is the number of your job
     - Previous example: `slurm-5980521.out`
  2. Output file(s) specific to your application
     - EG Abaqus: *`casename`*`.dat`, *`casename`*`.prt`, …
  3. Your job had a problem or failed: it may be reported in one of the above files – check the ***slurm-JOBID*.out** file.
     - (Technically: "the std error stream is redirected to the file")
- Various ways to view the files (they are plain text):

```
cat filename          # EG: cat slurm-5980521.out
less filename   (allows you to page through with spacebar)
gedit filename  (not recommended if it is large)
```

Why is my job still waiting?

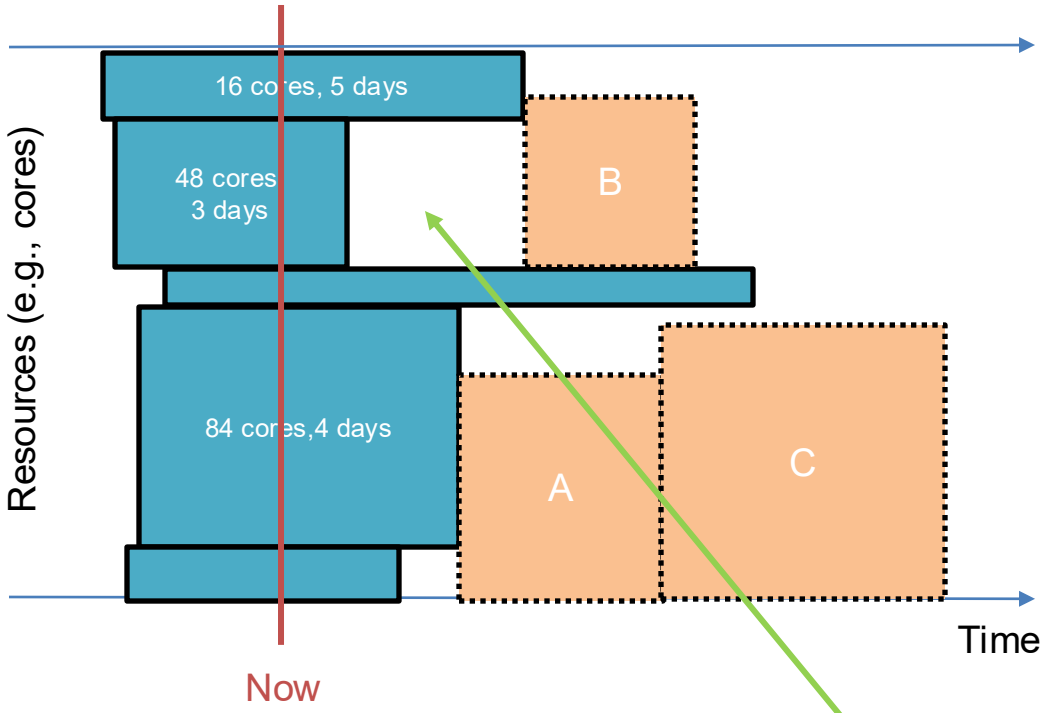
- Your job will wait until there are resources available (meeting your jobscript's requirements).
- Slurm keeps track of when resources will become free, based on the wallclock specified in all jobs.
  - Jobs are starting and finishing all the time
  - A more realistic wallclock time helps with scheduling
- The queue can be frustrating, but has advantages:
  - You can log off, switch off your laptop and your job will stay on the CSF. Log in later to check on job / collect the results.
  - You can submit many (100s, 1000s) jobs.
    - They might not *all* run at once - Slurm will decide this…
    - But many jobs might run at the same time – so more of your work is completed sooner
      - (you may need to use different files/folders for each job).

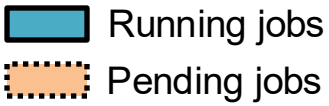
Why do I now need to specify a Wallclock time?

- If you've used the CSF in the past, you probably just relied on the default 7-day wallclock limit.
- This made it more difficult for the batch system to schedule jobs – it had to assume that all jobs would run for the full 7 days.
- We now require that you add a wallclock time to your jobscripts:
  - `#SBATCH -t 2-0` 2 days (0 hours). 7 days is the max.
  - `#SBATCH -t d-hh:mm:ss` Various formats accepted
- By specifying an accurate(ish) wallclock, Slurm can better plan when resources will become free.
  - Always err on side of caution – too much time is better than not enough time!
  - **Slurm will kill a job if still running, once the job's wallclock limit has been reached!**
  - You might have to run a few jobs to get a feel for how long they take.
  - Or run the first job with 7-days then check the actual runtime.

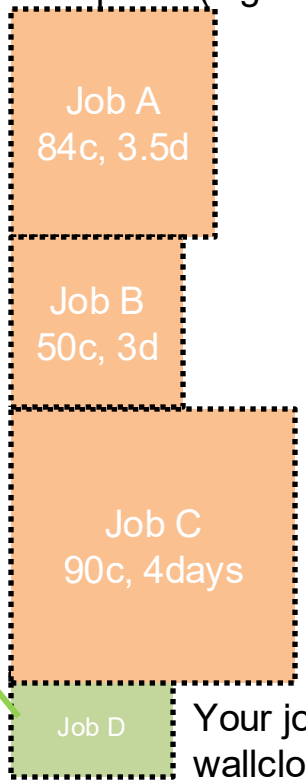
- Can't I just request 7-days for all of my jobs (or 4-days for GPU jobs)?
  - Yes, you can. But …
  - It might be possible to fit a shorter, small job in before larger jobs are able to use the resources.
  - Ultimately, everyone will wait longer

Backfill Scheduling





Job queue (highest priority at top)



Your job with a realistic wallclock limit.

A resource gap can be filled by a lower priority job, *provided* doing so will not delay higher priority jobs.

Without backfill, those resources would be idle.

Many Users Sharing the CSF

- 100s of users running 1000s of jobs
- Slurm gives each job a priority (number), which depends on
  - Size of research group's / school's CSF contribution
  - Amount of work already put through by that group and by you as an individual (this month)
- The time for your job to start depends on
  - Priority
  - Availability of requested resource (is CSF busy?)
- Jobs submitted *after* yours may start *before* yours!
- A few Jobs may never start
  - Slurm tries to spot invalid resource requests in jobscripts
  - Some may still get through then never run
- We try to ensure that if you submit some jobs, some of them will start within **24 hours**.
  - We make a check every morning of the waiting jobs

How busy is it?

- The CSF is always very busy!
- However, jobs frequently finish, allowing waiting ones to start
- To see all the jobs for everyone (put a blackslash at the start)

```
\squeue
```

- Note: *all* jobs shown as one long list - misleading
  - It displays running and waiting jobs
  - Your job is not necessarily stuck behind all others above yours in the `squeue` output.
  - CSF is split into a few partitions – the very big jobs do not compete with the smaller jobs for cores
- Do not try to guess when is a good time to submit your jobs.
  - If you have work ready to go, just **submit** it!
  - If your jobs are not in the queue, the scheduler cannot consider them
  - You will waste time, not gain it, by *not* submitting

Checking your jobs and deleting jobs

- `squeue` reports your job as 'F' (failed)
- Or your job has finished but you suspect it failed to complete correctly
  - The `slurm-`*`JOBID`*`.out` contains errors / incomplete results
  - Ask Slurm for info about your job once it has finished
    `seff` *`JOBID`* - Resource usage efficiency and exit code
    `sacct -j` *`JOBID`* - LOTS of stats about job (`MaxRSS` is peak mem used.)
- Most common causes of errors:
  - Job ran out of memory ("OOM" error in `slurm-`*`JOBID`*`.out`)
  - Missing directory (did you rename the directory before job ran?)
  - Unusual characters or **spaces** in file and directory names
  - No disk space on the filesystem – run in the scratch area to avoid this
- Detailed advice:
  https://ri.itservices.manchester.ac.uk/csf3/batch-slurm/monitoring/
- To delete a job from the queue (e.g., a failed job, or you just no longer want it – pending or running):

  `scancel` *`JOBID`*

PRACTICAL SESSION 2

Serial job

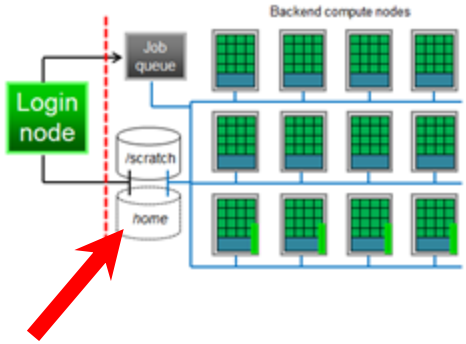
Practical Session 2 - Submitting jobs

- Follow the handout ‘Practical Session 2’
  - Use **`sbatch`** to submit a simple serial job on the CSF
  - Use **`squeue`** to look at the queues
  - Use **`scancel`** to kill jobs
  - Use **`sacct`** and **`seff`** to look at finished jobs
- Exercise sheet (pdf) available at: https://ri.itservices.manchester.ac.uk/course/rcsf/

CSF STORAGE (FILESYSTEMS)

Where to store your files…

Storage – *Home* filesystem



- Upon login, automatically placed in your *home* directory (folder)

```
/mnt/iusers01/group01/username
```

- Limited space, quota shared by everyone in the group
- Uses the Research Data Service (networked storage)
  - Large files can be slow-ish to read/write (implications for jobs)
- Which directory (folder) am I currently in?

```
pwd
```

- How much space am I using? (Linux commands!)

```
du -sh dirname          # Can take a while
```

- How big is that file?

```
ls -lh filename         # Letter el not number 1
```

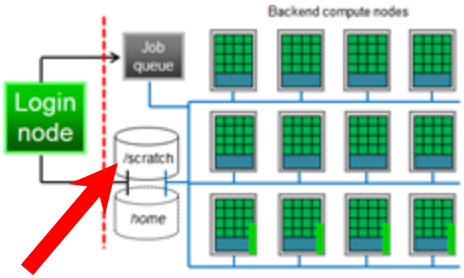
- How much space is used/free overall?

```
df -h .                  # The . is important!
```

Storage - *Home* filesystem

- *Home* is backed up and mirrored to another datacentre.
  - Provides a *small amount* of storage to enable login
  - Keep important files here (jobscripts, source code, …)
  - Deleted a file by mistake? We can tell *you* how to retrieve it
- Only you can access your home directory
  - File permissions can be used to give others access
  - Contact us if you want advice on this as they can be complex
- Do not run jobs from your *home* area (see later)
  - Can generate a lot of files, some of them large
  - Filling up the shared space will make your colleagues unhappy!
  - Consider compressing large (text) files with `gzip`

Storage – *Scratch* filesystem



- Filesystem local to CSF for:
  - Temporary files - can be huge
  - Running jobs from (it is faster!). Recommended!
- Shared by all CSF users, but we have 1.9PB
- Tidy up after each job finishes
- **Clean-up policy applies: files that have not been accessed (read, created or updated) for the past 90 days may be deleted automatically**
- Not backed up!
  - Move/copy important results to *home* area
  - Not considered safe for long term storage - hardware failure could cause data loss

Storage – *Scratch* filesystem

- Using scratch is easy: after log in, change to it:

  `cd ~/scratch`

  – Uses a 'symlink' (short cut) in your home dir to `/scratch/`*`username`*

- Create a directory (now we're in scratch):

  `mkdir myjobdir`

- Put all files relevant to your job in that directory and *run your jobs* from there - we'll try this out soon...
- All compute nodes see the same scratch area

Extra Storage Space (Optional)

- Some research groups have extra space, called "Research Data Storage" (RDS)
  - Ideal for storing larger results (which can fill your *home* area)
  - Check with your PI / supervisor whether they have any
- Example path:

  `/mnt/eps01-rds/group/username`
- There maybe a shortcut in your home area. But can always use:

  `cd /mnt/eps01-rds/group/username`
- To create a shortcut (named `data`) in your home area:

```
cd ~
ln -s /mnt/eps01-rds/group/username  data
```
- This storage is also backed up (like your *home* area)
- Often many TB, but again shared by everyone else from your group
  - Be fair in your usage

File Transfer

- Transfer files to-and-from your **home**, **scratch** or additional **Research Data Storage (RDS)** filesystems, to your local desktop/laptop.
- Transfers will go via one of the 3x CSF3 login nodes.
- Various methods available to transfer data to-and-from the CSF3.
  - **File Transfer Clients (GUIs)**
  - **Command- Line Tools**
- Covered in next practical (Practical 3)
- https://ri.itservices.manchester.ac.uk/csf3/filesystems/file-transfer/

File Transfer Command-Line Tools

- Fast and lightweight, ideal for large or automated transfers.
- Easily scriptable for batch operations or scheduled jobs.
- Uses secure protocols like SSH for encrypted transfers.
- Cross-platform: Works on Linux, macOS, and Windows (via terminals like MobaXterm, PuTTY).
- Examples include **scp** & **rsync**

```
scp myfile.txt username@csf3.itservices.manchester.ac.uk:
```

```
rsync -avz myfile.txt username@csf3.itservices.manchester.ac.uk:
```

- https://ri.itservices.manchester.ac.uk/csf3/filesystems/file-transfer/linux-mac/

File Transfer Clients (GUIs)

- Offer a user-friendly intuitive graphical interface for browsing and transferring files.
- Easier to explore remote directories and allow users to drag-and-drop files.
- Provides visual progress indicators and error messages.
- Clients typically use SSH in the background in order to transfer files securely
- Various clients available
  - MobaXterm (Windows)
  - WinSCP (Windows)
  - FileZilla (Windows, macOS, Linux)
  - CyberDuck (macOS)

Additional filesystem/file transfer info

- We have additional info about how to manage your files and your disk usage:
  https://ri.itservices.manchester.ac.uk/userdocs/file-management/
- Docs about file transfer:
  https://ri.itservices.manchester.ac.uk/userdocs/file-transfer/
- If you need to transfer a lot of files or big files to and from the CSF please do not do it on the login node
  - Ask for an account on the RDS-SSH service
- Many file management tasks can be included in your batch jobs – see the FAQ.

Basic Linux File Commands

A good Linux tutorial is available at: https://www.chm.bris.ac.uk/unix/

Command	Description
`less file1` `zless file2.gz`	Display the content of `file1` (text file) a page at a time on screen. If you've compressed `file2` with `gzip`, no need to uncompress first. Press `space` to page down through a long file Press `return` to scroll down a line at a time Press `b` to scroll back up a page Press `G` to go to end of file Press `q` to quit/exit
`cat file1` `zcat file2.gz`	Dump entire file to screen (a quick way to look at text files). If you've compress `file2` with `gzip`, no need to uncompress first.
`gedit file1`	Edit `file1` using a simple graphical text editor (similar to notepad on Windows). See later for more on opening graphical programs on the CSF so that they display a window on your computer.
`file filenameA`	Try to tell us what type of data is in `filenameA`. Useful to determine the output of some program where you are not sure what type of output it has generated. For example: `file output.dat` Might be `ASCII text` (so we can look at it with `less` or `gedit`) or might be `data` (you'll need some other program to read it)
`du -sh .`	How much disk space is current directory (all files and subdirs) using?
`df -h .`	How much free space is there in the current area?

Basic Linux File Commands

A good Linux tutorial is available at: https://www.chm.bris.ac.uk/unix/

Command	Description

`cd dir1`
`cd ~/dir1/dir2`
`cd ..`
`cd`
	Change directory (go in to `dir1` which is located in the current dir)
	Go in to `dir2` in `dir1` in home (~ is shorthand for *home*)
	Go up to parent directory (e.g., from `~/dir1/dir2` to `~/dir1`)
	Go back to *home* (useful if you become lost)

`pwd`	Lost? Print Working Directory (display current location)

`ls`
`ls -lh`
`ls -lh file1 dirA`
`ls -lh dirA/*.dat`
	List content (names of files and directories) of current directory
	List in long form (dates, file sizes, names) current directory
	List in long form (dates, file sizes, names) specified files, directories …
	List in long form all files ending in `.dat` in directory `dirA`

`mkdir dirA`	Make directory named `dirA` (in the current directory)
`cp fileA fileB`	Copy (duplicate) a file (copy `fileA` to a new file `fileB`)

`mv fileC fileD`
`mv fileE dirA`
`mv fileF dirA/fileG`
	Rename a file (from `fileC` to `fileD`). Works for directories too.
	Move `fileE` in to sub-directory `dirA` (`dirA` must exist)
	Move `fileF` AND rename it all in one go (`dirA` must exist)

`rm fileH`	Delete (remove) a file (caution!!)
`rm -rf dir1`	Delete directory and all files (and other sub-dirs) in there (caution!!!!!)

`gzip bigfile`
 	
`gunzip bigfile.gz`
	Compress a file (becomes `bigfile.gz`) to make better use of disk-space. Text files usually compress well.
	Uncompress previously compressed file (becomes `bigfile`).

PRACTICAL SESSION 3

File Transfer

Practical Session 3 – File Transfer

- Follow the hand-out ‘Practical Session 3’
  - Transfer a file: from the CSF to your PC
  - Transfer a file: from your PC to the CSF
  - Windows: use MobaXterm, Mac/Linux: use "scp"
    - Or, if time permits, Windows users can try option 2 - https://winscp.net/eng/download.php
- This is not a ‘real’ world example, but:
  - You may need to generate files on your PC for processing on the CSF (e.g. an "abaqus" input file)
  - Your supervisor may give you files that you then need to transfer to CSF
- Exercise sheet (pdf) available at: https://ri.itservices.manchester.ac.uk/course/rcsf/

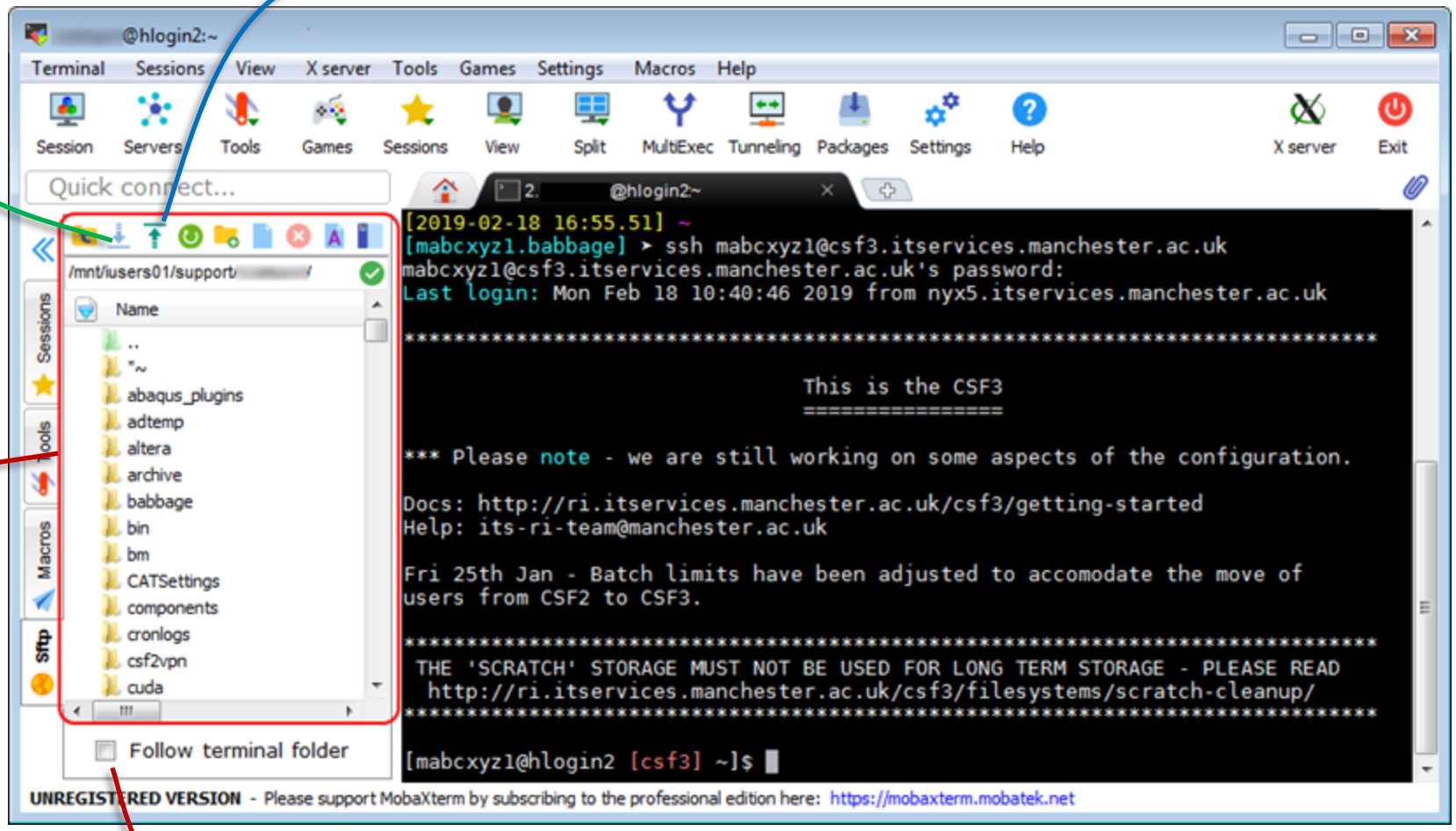
Exercise 3 – File Transfer Reference Slides

See also the exercise sheet for necessary steps

File transfer with MobaXterm

2. First select files in the MobaXterm browser. Then the ***Download*** button opens a file-browser to select a destination folder on your PC.

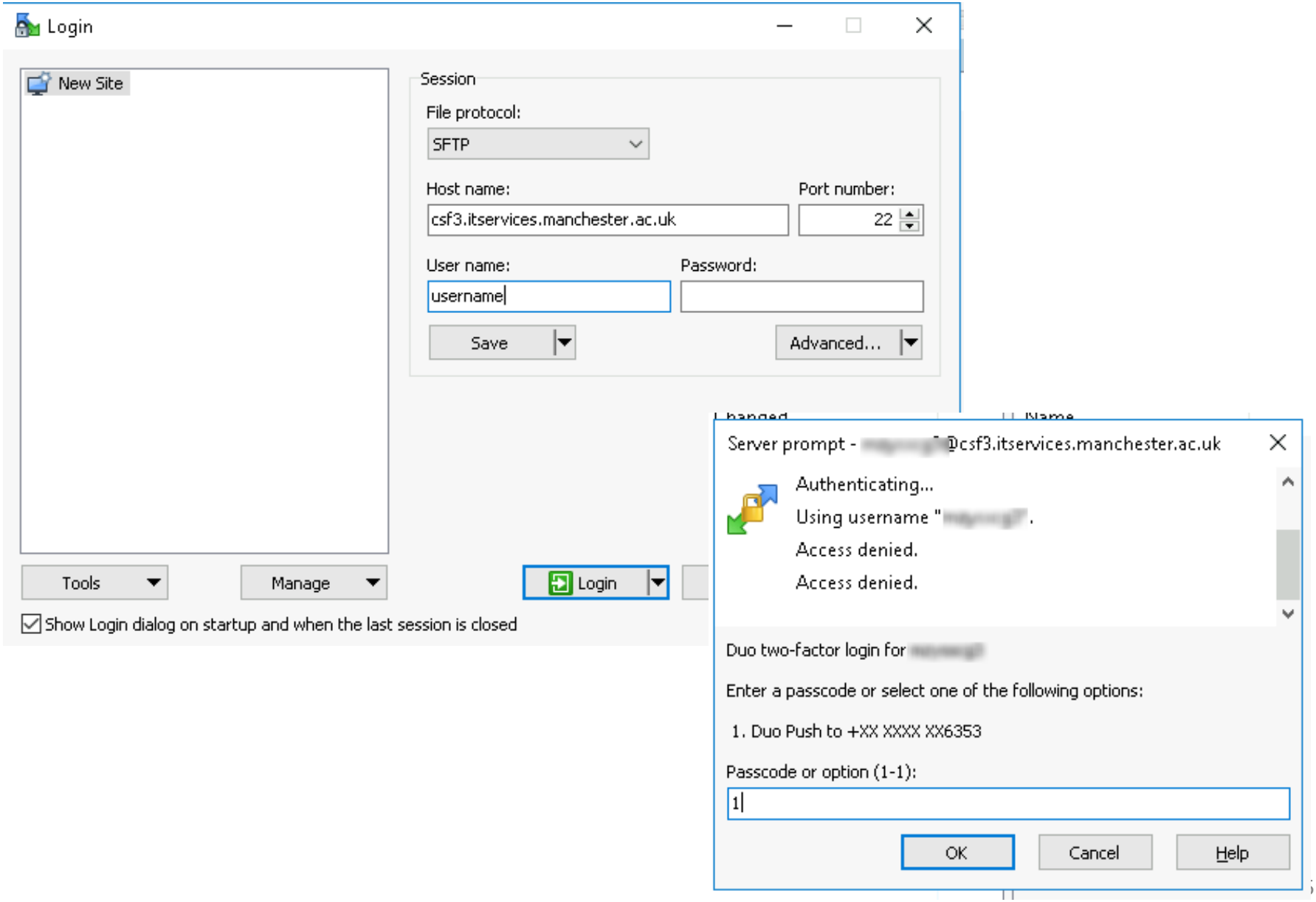
3. The ***Upload*** button opens a file-browser to select files on your computer to **upload to CSF** (current directory).



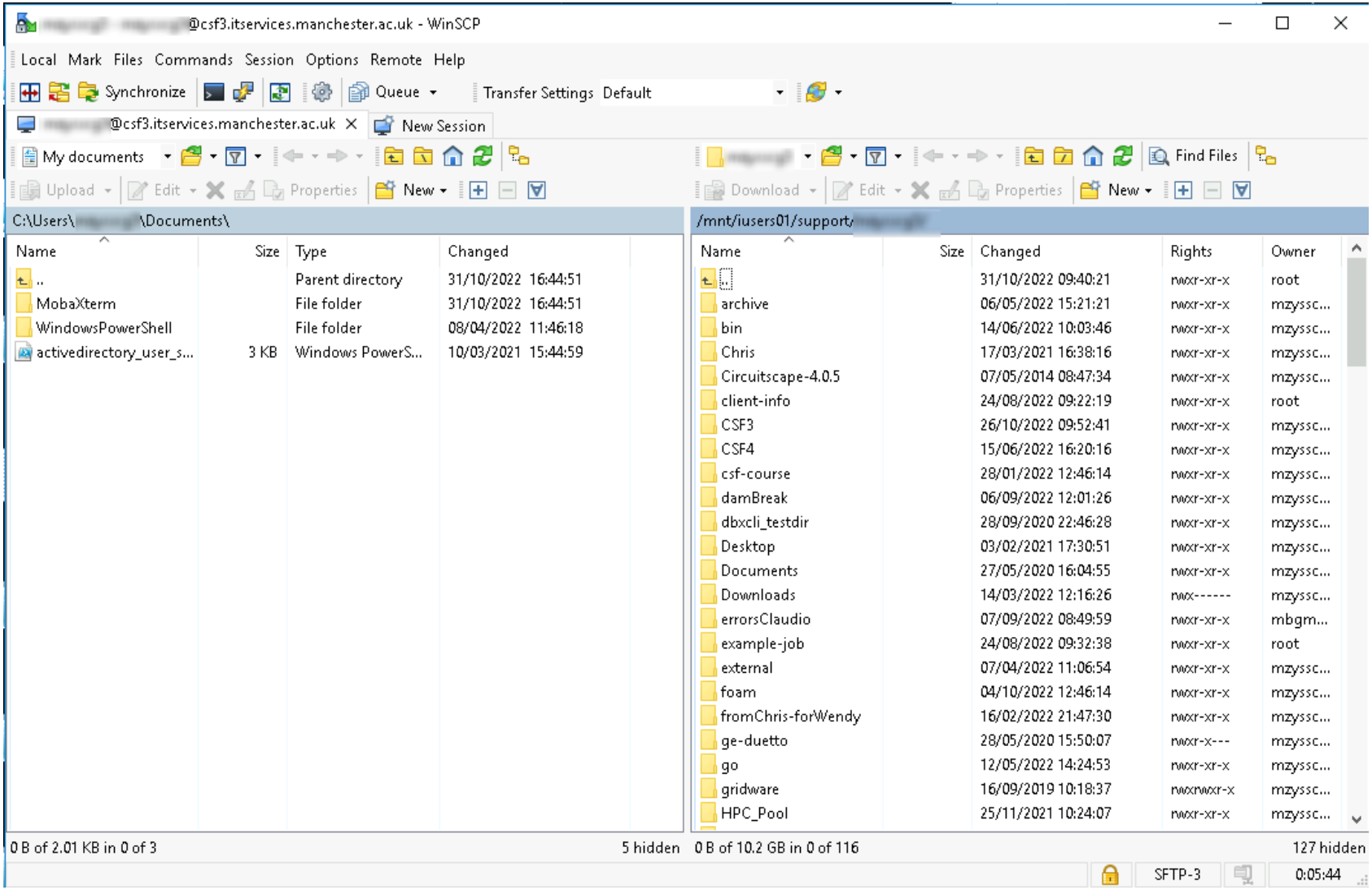
1. Drag-n-drop files from Windows Explorer or Desktop to/from here.

This can be flaky - do not use

File transfer with WinSCP



File transfer with WinSCP



home file transfer with Linux / Mac
(can also type in a local MobaXterm window)

The **:** is important!

- Transfer a file from your computer to your CSF home dir

```
scp myfile.txt username@csf3.itservices.manchester.ac.uk:~/training/
```

Exercise: Create a file on your PC named myfile.txt containing some text then transfer it to the CSF.

Destination directory on the **CSF**
~ is shorthand meaning "your home directory"
If no destination after the : then uses "your home directory"

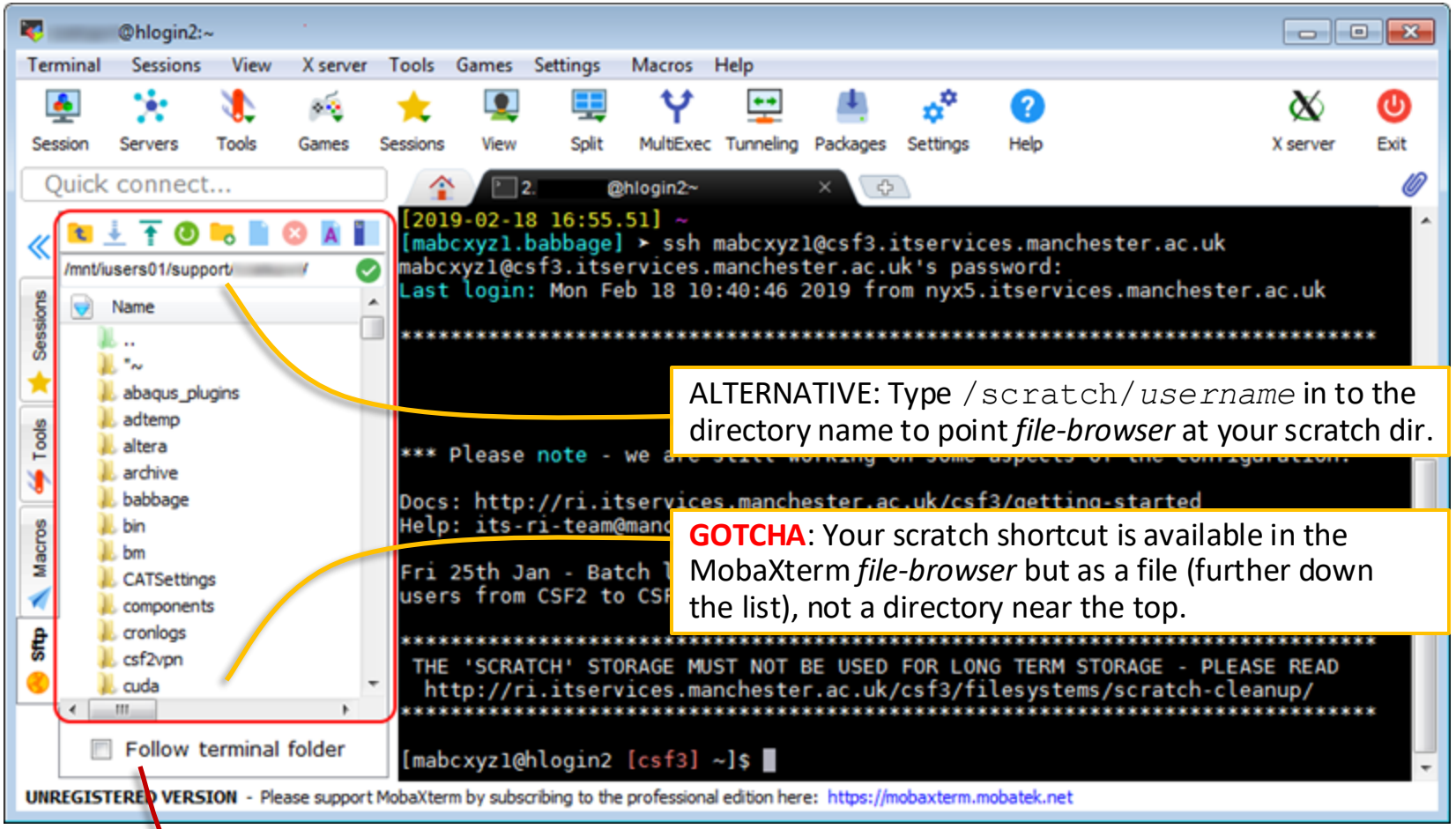
- Transfer a file from your CSF home dir to your computer

```
scp username@csf3.itservices.manchester.ac.uk:results.out .
```

The **.** is shorthand meaning
"the current directory" on **your** computer

- Change directory & filenames…

Accessing *scratch* with MobaXterm

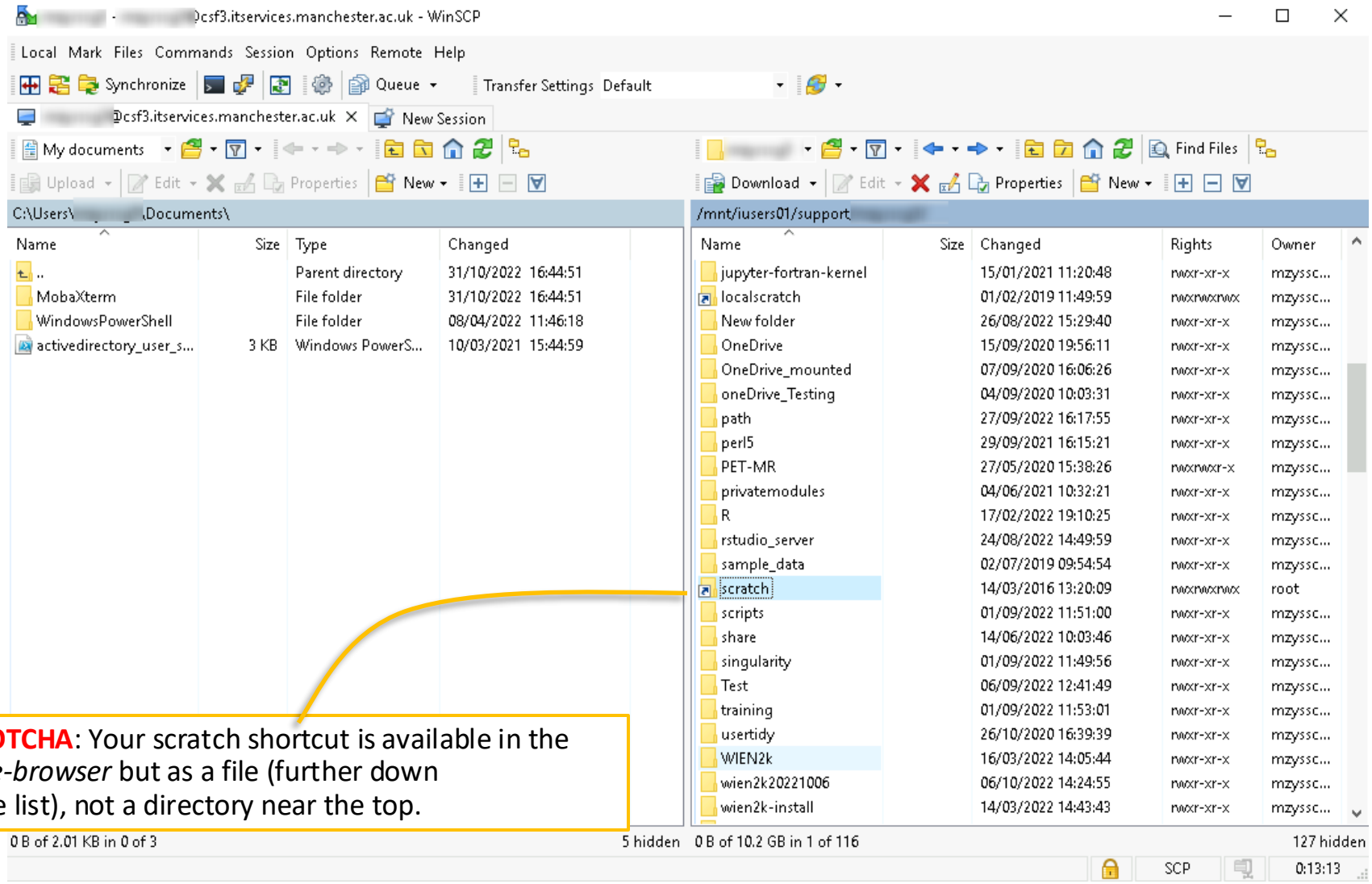


ALTERNATIVE: Type `/scratch/username` in to the directory name to point *file-browser* at your scratch dir.

GOTCHA: Your scratch shortcut is available in the MobaXterm *file-browser* but as a file (further down the list), not a directory near the top.

This can be flaky - do not use

Accessing *scratch* with WinSCP



GOTCHA: Your scratch shortcut is available in the *file-browser* but as a file (further down the list), not a directory near the top.

scratch file transfer with Linux / Mac
(can also type in a local MobaXterm window)

- Very similar to commands used earlier for our *home* directory

- Transfer a file from your computer to your CSF scratch dir

```
scp file2.txt username@csf3.itservices.manchester.ac.uk:~/scratch/
```

We give a destination after
the : meaning "use my scratch shortcut"
Omit the destination to transfer to *home* dir

- Transfer a file from your CSF scratch to your computer

```
scp username@csf3.itservices.manchester.ac.uk:scratch/results2.out results2.copy
```

Now you get a copy with a
different name on your computer.
Use . to keep the same name (results2.out)

- Change directory & filenames as required…

Need more help with the CSF?

- Extensive documentation about all aspects of the service:

  https://ri.itservices.manchester.ac.uk/csf3/
- Contact the Research Infrastructure Team via your Connect Portal

  https://ri.itservices.manchester.ac.uk/csf3/help/
- See you:
  - After lunch (in-person courses)

Thank you!