ASTR4004/ASTR8004
Astronomical Computing
Lecture 04

Christoph Federrath

05 August 2019

Bash, Remote Computing

1 Advanced methods for remote computing

1.1 Setting up an ssh tunnel

1. If you are connected to the VPN (see notes from last lecture), you can directly connect to `malice.anu.edu.au`. However, if you are outside the ANU network and you cannot use the VPN client for some reason, there is still another possibility to connect to the MSO servers. This requires you to first connect to a specific server at MSO that is accessible from the outside world. This server is called '`msossh1.anu.edu.au`'.

2. In order to connect to `malice.anu.edu.au`, we simply connect to `msossh1.anu.edu.au` first and then `ssh` from `msossh1.anu.edu.au` to `malice.anu.edu.au`. But in order to make our life easier, especially when copying files between remote computers, we can set up an `ssh` tunnel. Do this by adding/modifying the following lines in your `.ssh/config`:
```
Host malice
Hostname malice.anu.edu.au
ProxyCommand ssh -q -a -Y [your_mso_username]@msossh1.anu.edu.au nc %h %p
User [your_mso_username]
```
Save and close `.ssh/config`.

3. Now you should be able to connect to `malice.anu.edu.au` by using `> ssh -Y malice` directly from your local computer. It will probably ask for your password twice; the first time when it connects to `msossh1.anu.edu.au` and the second time when it connects from there to `malice.anu.edu.au`. So what this effectively does is that we tunnel through `msossh1` to `malice`.

4. As earlier, add a 'malice' alias to your `.alias` file as a shortcut for `ssh -Y malice`.

1.2 Copying files/data across remote computers

1. With the changes to `.ssh/config` that we made earlier, it is now also very simple to copy files between your computer and the remote host. Lets copy the density PDF data file from assignment 1 into your home directory on malice. First, download the data file to your local computer (if you don't have it already) from http://www.mso.anu.edu.au/

~chfeder/teaching/astr_4004_8004/material/mM4_10048_pdfs/EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data

2. Now copy the data file to `malice` using `scp` ('secure copy'):
   `> scp EXTREME_hdf5_plt_cnt_0050_dens.pdf_ln_data malice:`
   Note that you can use the tab key to auto-complete the rest of the awkward filename. If you just provided the initial few letters 'EX' and hit the tab key, the shell should automatically complete the rest of the filename. Note also that when you start `> scp` and then hit the tab key twice, you will be given a list of possible options for files in this directory. This is a very useful auto-completion function of the shell. Once you executed the `scp` command, you should see the file being transferred to malice. Note that the colon behind 'malice' refers to your home directory on malice. If you want to copy the file somewhere else, you just have to expand the path to the destination folder on malice after the colon.

2 Mounting remote file systems

1. If you are frequently accessing files on a remote host and would like to use them on your computer, you can use `sshfs` to mount a remote file system on your computer. This means you don't have to explicitly copy files every time, but they will simply be present in on of your local directories on your laptop and only if needed, will the files in there be copied via the network

2. For example, if we want to mount your $HOME from the MSO network on your laptop, we can first create a directory in your local (on your laptop) computer, called 'mso_home':
   `> mkdir ~/mso_home`

3. Now we mount your home dir at MSO into this folder: `> sshfs malice: ~/mso_home/`

4. Be very careful when move files in or out of this directory `~/mso_home`, because any modification of it locally on your laptop will be exactly reflected on your home dir in the MSO system. Moreover, the transfer of data goes via the network, so if you do this at your home it can take bandwidth from your internet connection, depending on the operations that you are doing in this directory; e.g., copying a 1 GB file in or out of `~/mso_home` will take time depending on your internet connection and will of course transfer that amount of data.

5. You can un-mount the directory (on Mac OS) via: `> diskutil unmount ~/mso_home/'`

2.1 Using rsync to create backups and to copy large data sets; use of tarballs

1. Download the tarball (a set of files bundled together in one compressed file) from http://www.mso.anu.edu.au/~chfeder/teaching/astr_4004_8004/material/mM4_10048_pdfs/EXTREME_pdfs.tar.gz

2. Create a new directory on your local computer called 'astro_comp' in your home directory:
   `> mkdir astro_comp`

3. Now move `EXTREME_pdfs.tar.gz` into this new directory:
   `> mv EXTREME_pdfs.tar.gz astro_comp/`
   ...and change into `astro_comp/`

4. Decompress and un-tar the tarball, using:
   `> tar -zxvf EXTREME_pdfs.tar.gz`
   This should generate 71 files. Do you remember how to check the total number of files in `astro_comp/`?

5. Now lets make a complete backup of the directory `astro_comp/` and all the files in it by sending a copy to `malice`. First change back into your home directory on your local machine. Then do:
   `> rsync -av --stats astro_comp/ malice:astro_comp/`
   This generates a complete 1:1 copy of your local directory `astro_comp/` in your `malice`'s home directory with the same directory name there. Please have a look at the status summary output when `rsync` returns from doing its job. Actually, `rsync` can be used to backup data without having to transfer/copy all the files every time, but instead it only transfers changes. For instance, if you now run the same rsync command from above again, you will see that no files will be transferred (because nothing changed in your local copy). If, however, you were to modify one or more of the files in `astro_comp/` and you do the `rsync` again, only the modified files will be transferred. This is really useful also if you have many many big files to transfer to/from a computer or between different hosts, because if some of the files can't be transferred within a day or so and/or the ssh connection closes for some reason, at least rsync will continue from the same point where it stopped to synchronise the folders and files, instead of starting the copy process from scratch. BTW: an important part of astronomical computing is to make regular backups of your files :)

2.2 Use of `nohup` and `nice`

1. Make a Bash script called `nohup_script.sh` that prints the current time every second for the next hour. Use a bash loop and `date` to print the current date and time. Let the code pause for 1 second within each loop increment, by using `sleep 1`.

2. Start the script to see if it works. It should print the date and time to the screen every second. Note that you can stop the script by pressing Ctrl-C.

3. Now start the script, but redirect the stdout and stderr to a file `shell.out`. While the script is running in one shell, open another shell and look into the file. You should see that every second a new line is appended with the current date and time to the end of the file.

4. Start the same script, but this time with `nohup`. Simply place `nohup` in front of the command that starts the script and `&` after the command. `nohup` is a wrapper for any command that you want to start such that it does not hang up (`nohup` means 'no hang up') when you log out. The final `&` at the end of a command line is used to start that process in the background, which means that you get the shell back when you hit enter, but the program keeps running in the background.

5. Use `tail -f shell.out` to show the end of the output file. It also follows any changes to the file (the `-f` option). You should see how the file grows and how every second, the date and time is appended as a new line to the file.

6. Now log out of the MSO server. Since we have started the script with `nohup`, it should not hang up after we logged out, but instead will keep running on the server even though

we are logged out. So, lets login again and see if the script kept on writing the time to the file while we were away.

7. Caution: if you start something with `nohup`, it will keep running unless it's killed by the user or an admin. We made the script so it would stop after one hour, but lets simply kill the script by hand right now. To achieve that, we use the `ps` command, which shows all running processes on the host. Lets first add a customised version of `ps` to our `.alias` file to make it an easy task to show all our own processes (there is a whole bunch of other processes that are also running simultaneously on the server, but that we don't want to be listed). Add the following alias to your aliases:
   `alias myps='ps -elf |grep $USER'`
   and log out and log back in for this change to take effect or simply `source .alias` in your home directory.

8. Now do `> myps` in the shell. This should list your current active processes (see the pipe to `grep $USER`). Find the process ID that belongs to the running script that we started with `nohup` and kill the job using:
   `> kill -9 [PID]`
   where you have to replace '[PID]' with the process ID of the job to be killed. Alternatively, you can use
   `> killall 'bash'`
   which kills jobs based on their name. This will kill all your running bash scripts.

9. Note: You'd normally want to use `nohup` for jobs/scripts that take very long to run, for example over night or even longer and you don't want to keep an active shell open to the remote host while the script is running. This is when `nohup` is really useful. However, we should consider that other users might be running jobs or that a user could be physically sitting at that computer/host at the same time and trying to do some work. So a *nice* thing in this case is to use `nice`, which means that your job will only use CPU time (or compute power) when it is available. So in case the machine is under heavy load and running many processes at the same time, if you started your script with `nohup nice ./script.sh &`, then your job won't block the jobs of other users so much, because it waits for times when the machine is not under heavy work load.