

Introduction to Linux and Anaconda

Getting familiar with the terminal

Espen Mikal Robertsen
01.11.2021, Lisbon, Portugal

Outline

- How to work with VMs, setup and access
- **Checkpoint 1: Getting everyone up and running!**
- Getting around in Linux **practical / live demo**
- **Checkpoint 2: Copying your practical files!**
- Intro to Anaconda package manager
- Managing environments in Anaconda **practical**
- **Checkpoint 3: Playing around with Anaconda**

Introduction to Hands-on

Most bioinformatic analyses tools require Linux to work.
We will be working on virtual Ubuntu machines hosted in a cloud

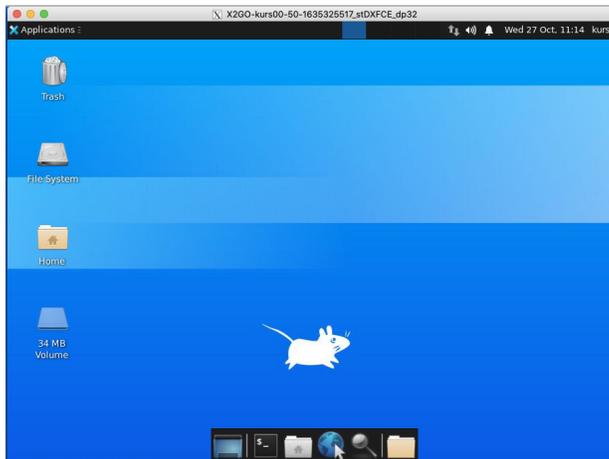


Virtual machines via remote desktop

To use your designated virtual machine you need to connect to it using a remote desktop tool on your client machine
We will use a tool called X2Go to achieve this



Client (Windows)



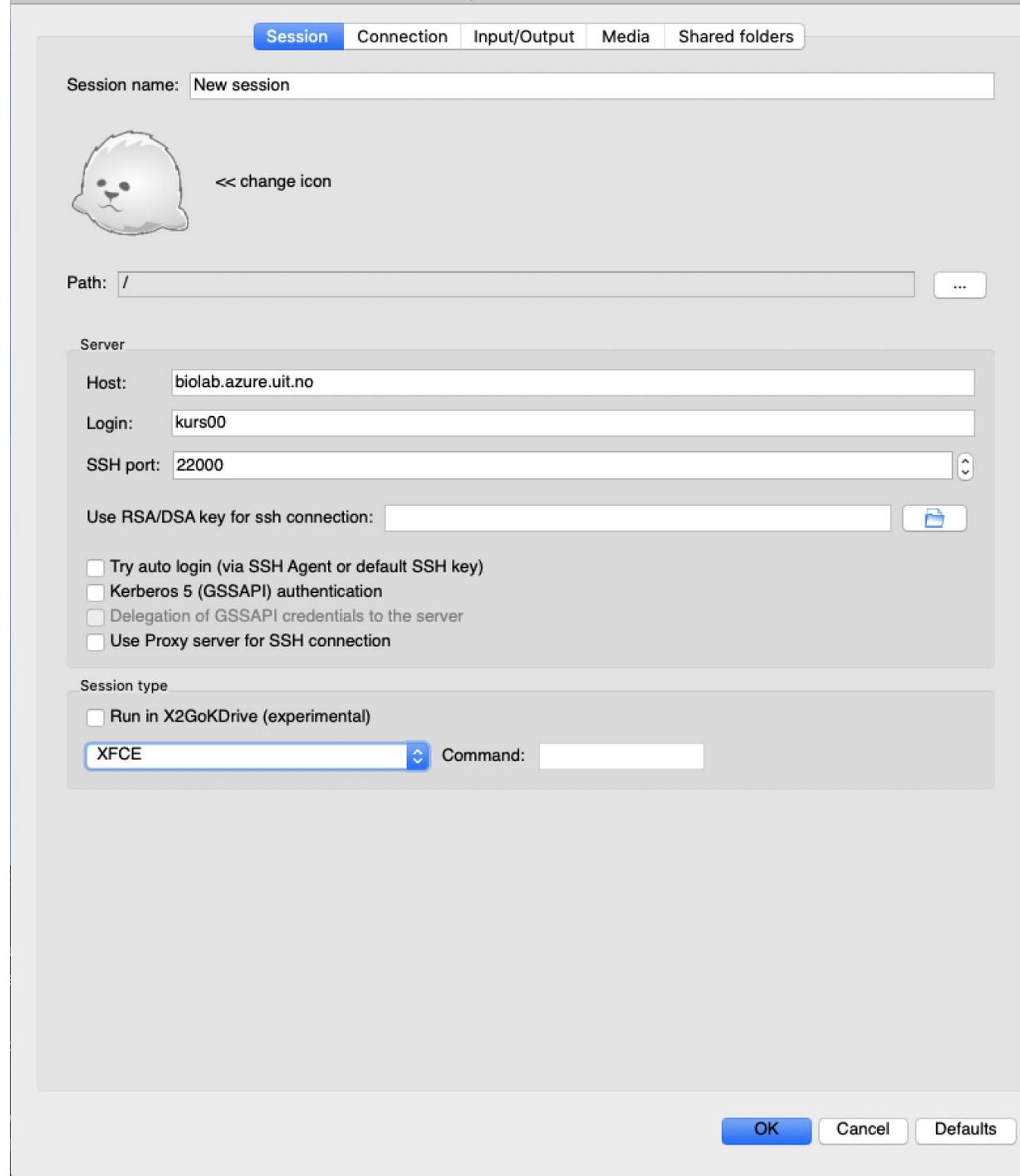
VM (Ubuntu)



Someone else's
computers
(Cloud)

Connecting with x2go

To use your VM you need a host (address), a username and an SSH port specific to your user



The screenshot shows the X2Go session configuration dialog box. At the top, there are tabs for "Session", "Connection", "Input/Output", "Media", and "Shared folders". The "Session" tab is active. The "Session name" field contains "New session". Below it is a default icon of a white seal with the text "<< change icon". The "Path" field contains "/". The "Server" section includes fields for "Host" (biolab.azure.uit.no), "Login" (kurs00), and "SSH port" (22000). There is a field for "Use RSA/DSA key for ssh connection" with a file selection icon. Below this are four unchecked checkboxes: "Try auto login (via SSH Agent or default SSH key)", "Kerberos 5 (GSSAPI) authentication", "Delegation of GSSAPI credentials to the server", and "Use Proxy server for SSH connection". The "Session type" section has an unchecked checkbox for "Run in X2GoKDrive (experimental)" and a dropdown menu set to "XFCE" next to a "Command:" field. At the bottom right, there are "OK", "Cancel", and "Defaults" buttons.

Session name: New session

<< change icon

Path: /

Server

Host: biolab.azure.uit.no

Login: kurs00

SSH port: 22000

Use RSA/DSA key for ssh connection:

Try auto login (via SSH Agent or default SSH key)

Kerberos 5 (GSSAPI) authentication

Delegation of GSSAPI credentials to the server

Use Proxy server for SSH connection

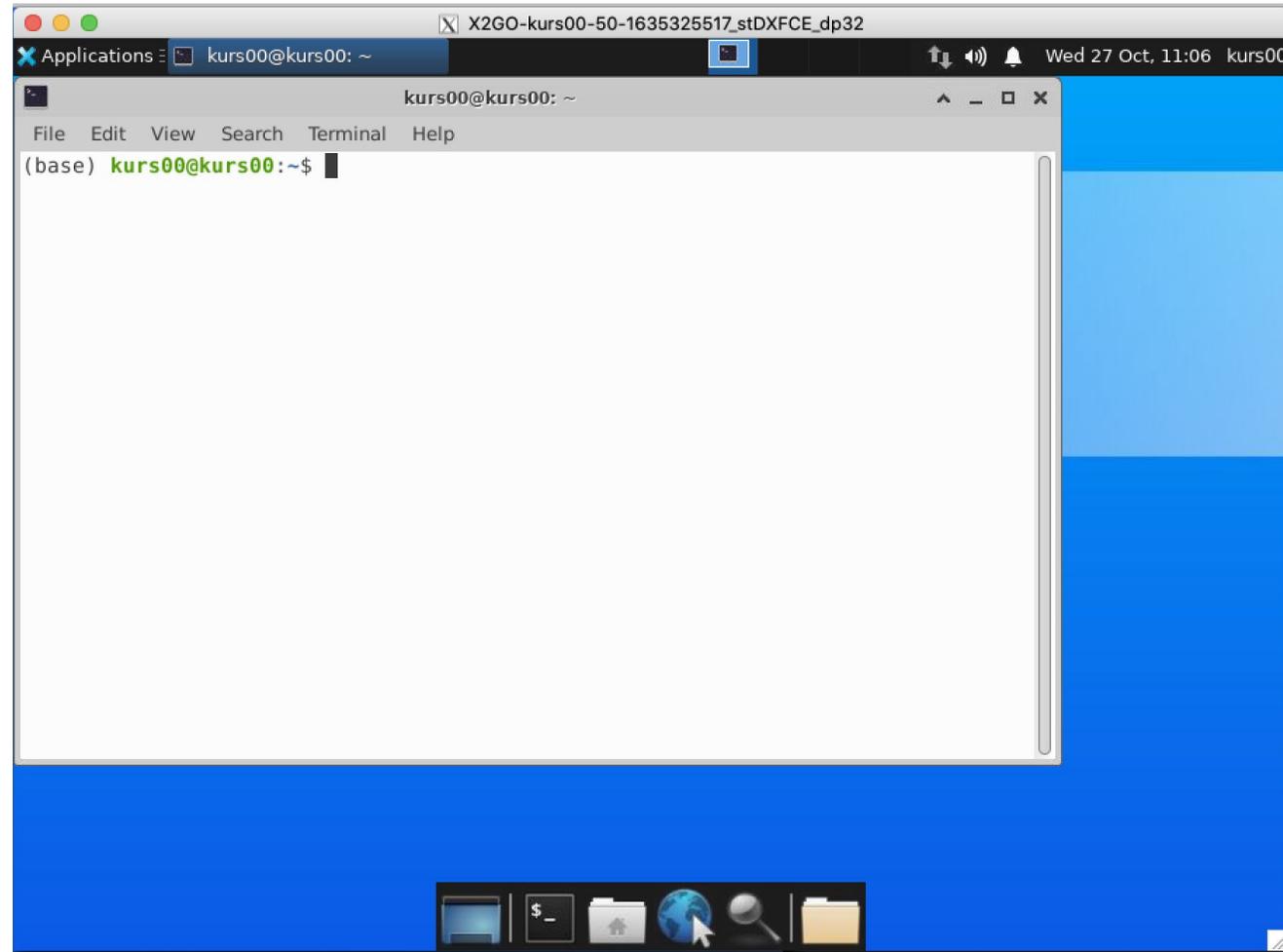
Session type

Run in X2GoKDrive (experimental)

XFCE Command:

OK Cancel Defaults

Once connected, your personal VM should look like this



Once inside your virtual VM, you need to start a terminal window to start working on the exercises. Data (Practical) will be copied from shared disk.

Vms cost money and are on a time schedule

- Automatic shutdown at 18:00
- You can control your VM from a website:
<https://biolab.azurewebsites.net/>
Remember to run **stop-server** in the terminal when done!

UiT Biolab host control

Biolab host control

Press the button to start the corresponding machine. The machine will start within a few minutes. Please, only start the machine you have been assigned.

When you are done run the command `stop-server` from the command prompt and the machine will shut down itself. This helps us keep the cost down as the machines are billed by the minute.

Start kurs00	Start kurs01	Start kurs02	Start kurs03	Start kurs04
Start kurs05	Start kurs06	Start kurs07	Start kurs08	Start kurs09
Start kurs10	Start kurs11	Start kurs12	Start kurs13	Start kurs14
Start kurs15	Start kurs16	Start kurs17	Start kurs18	Start kurs19

Checkpoint 1: Let's get everyone up and running!

1. Start Remote desktop client (X2Go)
2. Input address / credentials (details in link at the bottom)
3. Log in to VM from login screen

<https://bit.ly/3EgXMUt>

Checkpoint 1: Let's get everyone up and running!

The screenshot shows the X2Go configuration window with the following details:

- Session name: New session
- Path: /
- Server: biolab.azure.uit.no
- Host: biolab.azure.uit.no
- Login: kurs00
- SSH port: 22000
- Use RSA/DSA key for ssh connection: (empty)
- Try auto login (via SSH Agent or default SSH key):
- Kerberos 5 (GSSAPI) authentication:
- Delegation of GSSAPI credentials to the server:
- Use Proxy server for SSH connection:
- Session type: Run in X2GoKDrive (experimental)
- XFCE (selected in dropdown)
- Command: (empty)

At the bottom, there are buttons for OK, Cancel, and Defaults.

The screenshot shows a Google Sheet titled "User_credentials" with the following content:

Host: biolab.azure.uit.no

Example SSH: `ssh kurs02@biolab.azure.uit.no -p 22002`

Example X2Go: Host: [kurs02@biolab.azure.uit.no](https://biolab.azure.uit.no)

Login: [kurs02](https://biolab.azure.uit.no)

SSH Port: [22002](https://biolab.azure.uit.no)

Startup: <https://biolab.azurewebsites.net/>

Needed by X2Go				Input your credentials		Success?	
Machine name	User	UID	SSH Port	Password	Student Name	Student Email	I'm in!
kurs00	kurs00	22000	22000	uuw7jieH	(Teachers)	---	X
kurs01	kurs01	22001	22001	NaiHei7r			
kurs02	kurs02	22002	22002	bu4haiVa			
kurs03	kurs03	22003	22003	agaiWuc4			
kurs04	kurs04	22004	22004	aphaeM4v			
kurs05	kurs05	22005	22005	guiM9Aik			
kurs06	kurs06	22006	22006	Yec4thoh			
kurs07	kurs07	22007	22007	aa4Vied3			
kurs08	kurs08	22008	22008	aev3xae9			
kurs09	kurs09	22009	22009	Eefeiya3			
kurs10	kurs10	22010	22010	uPook3ou			
kurs11	kurs11	22011	22011	Keish9Ah			
kurs12	kurs12	22012	22012	em7asie3			
kurs13	kurs13	22013	22013	idieJ4ie			
kurs14	kurs14	22014	22014	HaecaV4u			
kurs15	kurs15	22015	22015	uen9Heju			
kurs16	kurs16	22016	22016	RahWoh4z			
kurs17	kurs17	22017	22017	eey9Epai			
kurs18	kurs18	22018	22018	jeTh7gug			
kurs19	kurs19	22019	22019	Paesh9be			
kurs20	kurs20	22020	22020	Jaeng3so			
kurs21	kurs21	22021	22021	ioch3Chu			

The screenshot shows a login dialog box with the following details:

- Title: Biolab XFCE on biolab.azure.uit.no
- Login: (input field)
- Password: (input field)
- Buttons: Ok, Cancel

Understanding the filesystem is apparently not trivial knowledge anymore

GEN Z KIDS APPARENTLY DON'T UNDERSTAND HOW FILE SYSTEMS WORK

"THEY SEE IT LIKE ONE BUCKET, AND EVERYTHING'S IN THE BUCKET."

Giant Bucket

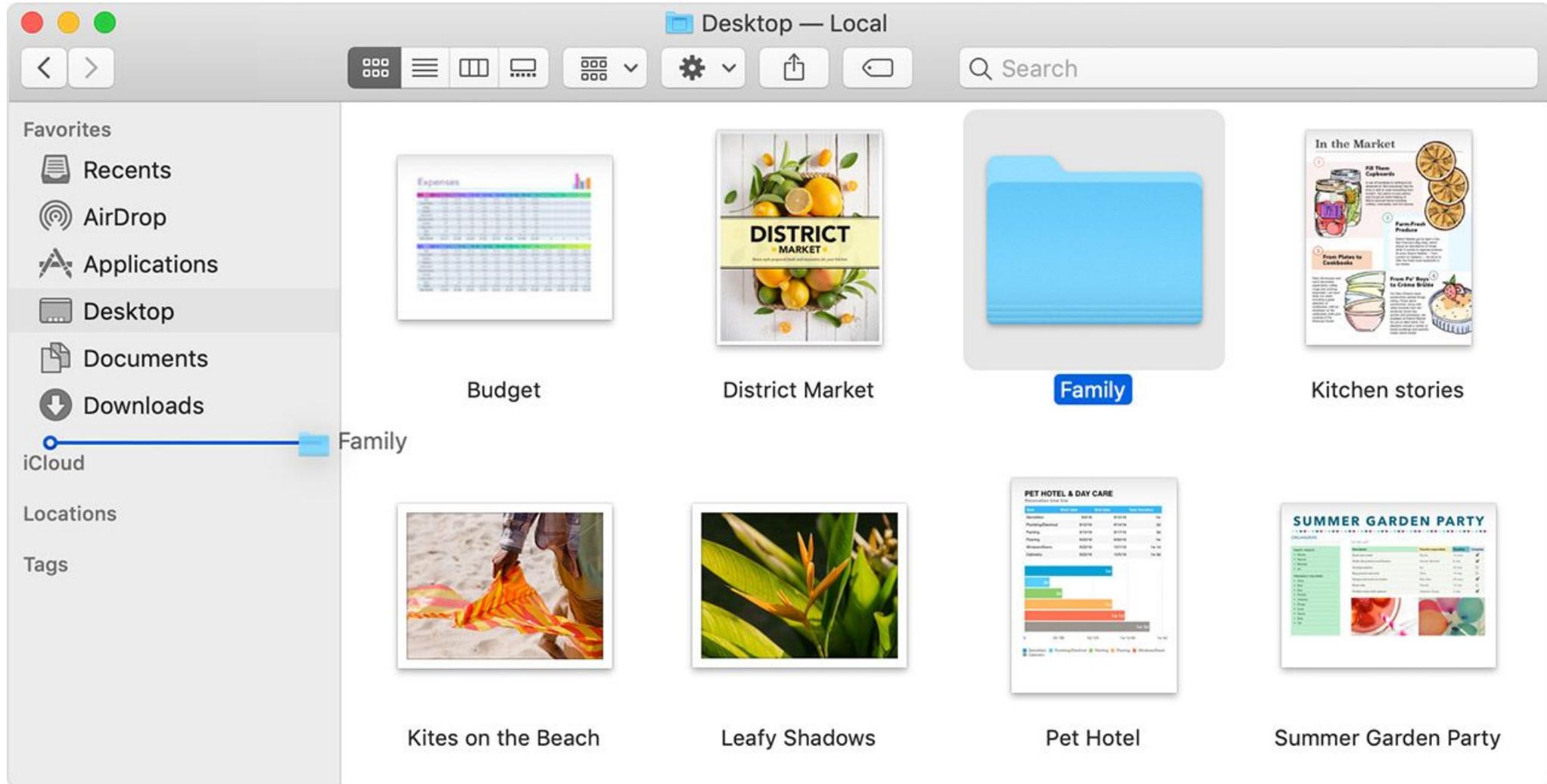
Over the past few years, many professors have noticed an alarming trend among their students. Overall, members of Gen Z, even those studying [technical scientific fields](#), seem to have a total misunderstanding of computer storage, [The Verge reports](#), and many fail to conceptualize the concept of directories and folders filled with digital files.

"The most intuitive thing would be the laundry basket where you have everything kind of together, and you're just kind of pulling out what you need at any given time," Princeton University senior Joshua Drossman told [The Verge](#).

(<https://futurism.com/the-byte/gen-z-kids-file-systems>)

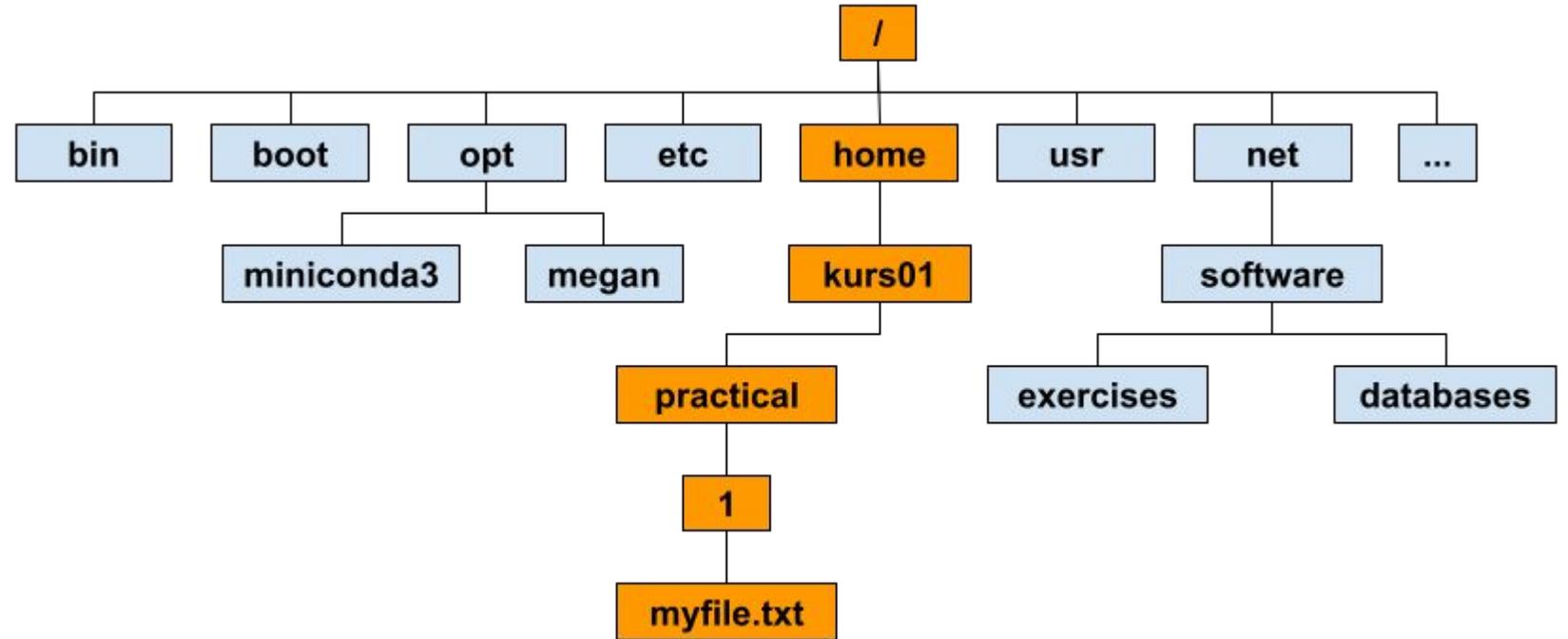
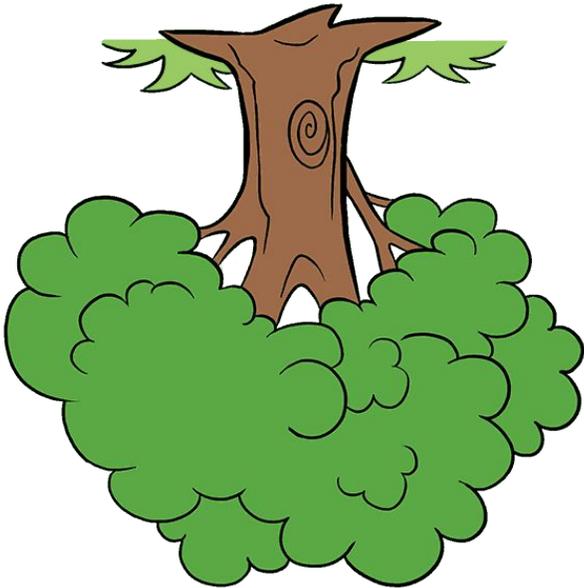
A quick overview of the filesystem

The shiny user friendly file system browser on mac is an abstraction.



The filesystem using terminal is not abstracted

... which can be a bit confusing at first. Think up-side-down tree!



`/home/kurso1/practical/1/myfile.txt`

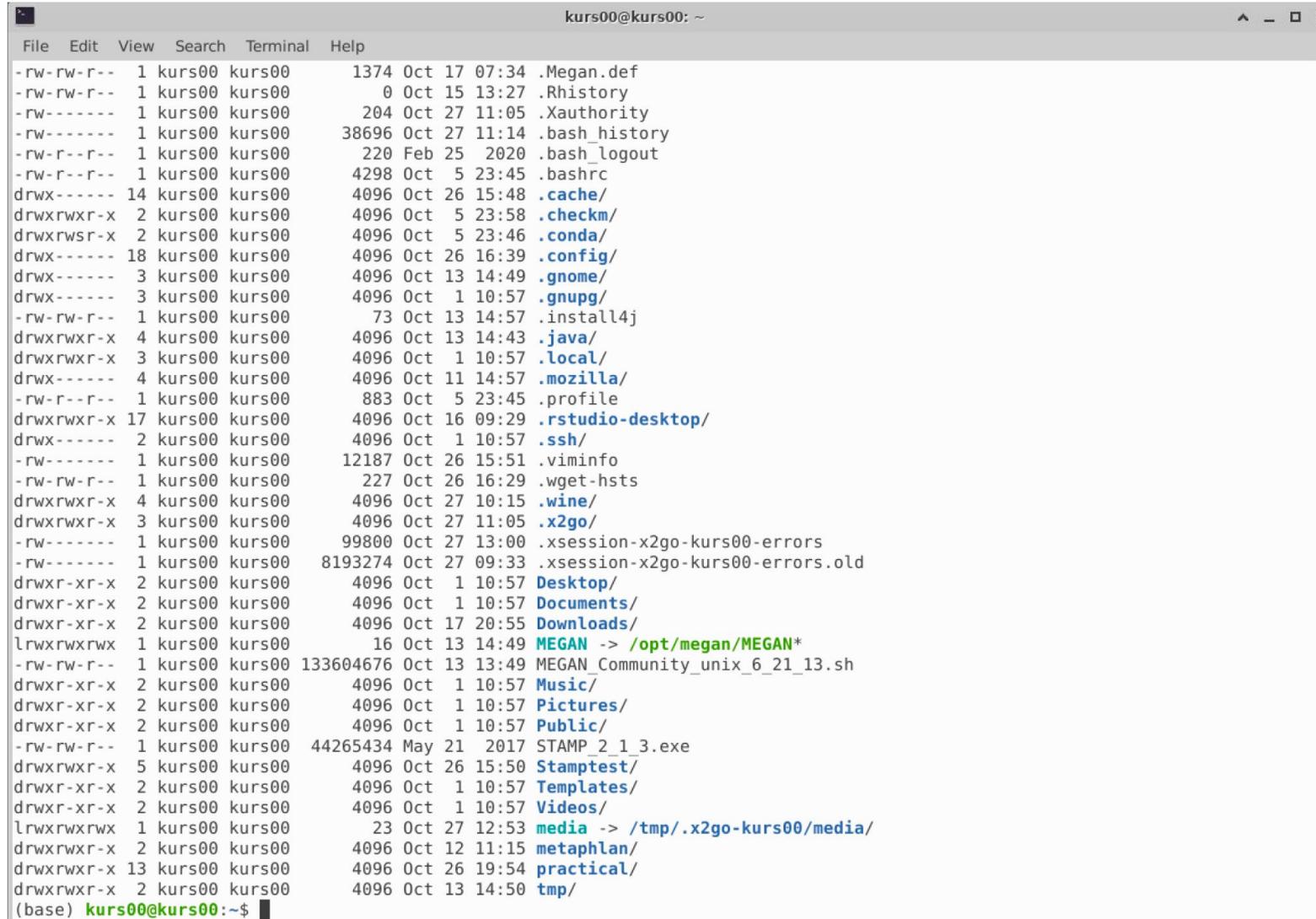
An quick overview of the filesystem (from root, /)

The complete linux filesystem looks quite different from Windows and Apple abstractions.

Path	Description
/bin	Common executables available for everyone, “cp”, “rm”, “ls” etc
/boot	Kernel and boot configuration
/etc	System and program configuration files
/home	Non-root user home directories (/home/kurs00)
/lost+found	Saved files due to some failure
/media - /mnt - /net	Directories for mounting external devices such as disks
/opt	Various software
/proc	Virtual filesystem for resources, processes and more
/tmp	Temporary files. These will disappear on reboot!
/usr	“Mini filesystem” on a user level instead of system level
/var	Variable files, logs etc.

The filesystem using terminal is not abstracted

... and looks quite different in the terminal on linux, but is actually the same thing.



```
kurs00@kurs00: ~  
File Edit View Search Terminal Help  
-rw-rw-r-- 1 kurs00 kurs00 1374 Oct 17 07:34 .Megan.def  
-rw-rw-r-- 1 kurs00 kurs00 0 Oct 15 13:27 .Rhistory  
-rw----- 1 kurs00 kurs00 204 Oct 27 11:05 .Xauthority  
-rw----- 1 kurs00 kurs00 38696 Oct 27 11:14 .bash_history  
-rw-r--r-- 1 kurs00 kurs00 220 Feb 25 2020 .bash_logout  
-rw-r--r-- 1 kurs00 kurs00 4298 Oct 5 23:45 .bashrc  
drwx----- 14 kurs00 kurs00 4096 Oct 26 15:48 .cache/  
drwxrwxr-x 2 kurs00 kurs00 4096 Oct 5 23:58 .checkm/  
drwxrwsr-x 2 kurs00 kurs00 4096 Oct 5 23:46 .conda/  
drwx----- 18 kurs00 kurs00 4096 Oct 26 16:39 .config/  
drwx----- 3 kurs00 kurs00 4096 Oct 13 14:49 .gnome/  
drwx----- 3 kurs00 kurs00 4096 Oct 1 10:57 .gnupg/  
-rw-rw-r-- 1 kurs00 kurs00 73 Oct 13 14:57 .install4j  
drwxrwxr-x 4 kurs00 kurs00 4096 Oct 13 14:43 .java/  
drwxrwxr-x 3 kurs00 kurs00 4096 Oct 1 10:57 .local/  
drwx----- 4 kurs00 kurs00 4096 Oct 11 14:57 .mozilla/  
-rw-r--r-- 1 kurs00 kurs00 883 Oct 5 23:45 .profile  
drwxrwxr-x 17 kurs00 kurs00 4096 Oct 16 09:29 .rstudio-desktop/  
drwx----- 2 kurs00 kurs00 4096 Oct 1 10:57 .ssh/  
-rw----- 1 kurs00 kurs00 12187 Oct 26 15:51 .viminfo  
-rw-rw-r-- 1 kurs00 kurs00 227 Oct 26 16:29 .wget-hsts  
drwxrwxr-x 4 kurs00 kurs00 4096 Oct 27 10:15 .wine/  
drwxrwxr-x 3 kurs00 kurs00 4096 Oct 27 11:05 .x2go/  
-rw----- 1 kurs00 kurs00 99800 Oct 27 13:00 .xsession-x2go-kurs00-errors  
-rw----- 1 kurs00 kurs00 8193274 Oct 27 09:33 .xsession-x2go-kurs00-errors.old  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 1 10:57 Desktop/  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 1 10:57 Documents/  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 17 20:55 Downloads/  
lrwxrwxrwx 1 kurs00 kurs00 16 Oct 13 14:49 MEGAN -> /opt/megan/MEGAN*  
-rw-rw-r-- 1 kurs00 kurs00 133604676 Oct 13 13:49 MEGAN_Community_unix_6_21_13.sh  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 1 10:57 Music/  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 1 10:57 Pictures/  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 1 10:57 Public/  
-rw-rw-r-- 1 kurs00 kurs00 44265434 May 21 2017 STAMP_2_1_3.exe  
drwxrwxr-x 5 kurs00 kurs00 4096 Oct 26 15:50 Stamptest/  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 1 10:57 Templates/  
drwxr-xr-x 2 kurs00 kurs00 4096 Oct 1 10:57 Videos/  
lrwxrwxrwx 1 kurs00 kurs00 23 Oct 27 12:53 media -> /tmp/.x2go-kurs00/media/  
drwxrwxr-x 2 kurs00 kurs00 4096 Oct 12 11:15 metaphlan/  
drwxrwxr-x 13 kurs00 kurs00 4096 Oct 26 19:54 practical/  
drwxrwxr-x 2 kurs00 kurs00 4096 Oct 13 14:50 tmp/  
(base) kurs00@kurs00:~$
```

When using the terminal on Unix systems

... remember these 4 things:

Three characters with special meaning

- ~ The tilde character implies your home directory (/home/kursXX)
- . The dot implies this directory (current)
- .. The double dot implies the parent directory ("one up")

Use tab auto-completion!

Practical / Live demo: Getting around in Linux

Green boxes and blue boxes

Green - Essential to complete the course

Blue - Convenience / Power user

- Filesystem navigation and manipulation (create, remove, move files/folders)
- Generic program execution
- Inspect / manipulate files
- Characters with special meaning and other tricks
- Build confidence!

The “prompt”:

```
(base) kursxx@kursxx:~$
```

```
$conda init bash
```

Navigating the filesystem

The same way as you would click files and folders in a file browser, you can navigate through your filesystem in the terminal with some simple commands.

The terminal offers **TAB autocompletion** and a history, which makes things a lot easier

Dots and double dots - This folder, parent folder

Tilde (~) - Refers to your home folder

Command	Function	Relevant Examples
cd	Change directory	“cd”, “cd ..”, “cd practical/”
ls	List the contents of a directory	“ll”, “ls -h”
pwd	Shows current location	“pwd”

Creating folders, copying and renaming

We can also create, remove and copy files and folders in the terminal the same way as we would in the file manager

Rule of thumb: What file? Where?

Command	Function	Relevant Examples
mv	Move and / or rename files / directories	"mv text.txt ~/text.txt"
cp	Copy files / directories	"cp text.txt ~/text.txt"
mkdir	Create a directory	"mkdir mydir"
rm	Remove files / directories	"rm text.txt", "rm -rf my_dir/"

Executing programs

Programs are usually executed in a generic way in linux terminal in the form of **programname --parameter --parameter2 --parameter3** etc...

Programs are attached to the terminal (tty) and will terminate if you close the terminal

Command	Function	Relevant Examples
programname --parameter -p	Run a program	"ls -alh"
programname --help -h	Generic help function	"ls "
which programname	Shows absolute path for shortcut	"which ls"
Ctrl + C / (D)	Terminate process	"sleep 50" then Ctrl+C

Checkpoint 2: Let's copy practicals and exercises!

Your practicals (files needed to complete assignments) are located on a shared disk you only have read access to

This way, if you screw something up, you can always get a fresh copy of your material from this disk

Let's copy it over to your home folder:

1. "Activate" the shared folder (**cd /net/software**)
2. Navigate to your home folder (**cd**)
3. Make a practicals folder (**mkdir practicals**)
4. Change directory (**cd practicals**)
5. Copy practicals/ : **cp -r /net/software/practical/1 .**

We will most likely do this for every module (1,2,3,4...) for terminal-practise (and lower wait times)

Inspecting / Editing files

More often than not, you want to inspect your files to see if your result is what it is supposed to be. Use "less" to inspect files

Use less on big files as it reads file incrementally (needs way less memory)

Use vi (Vim) on smaller files if you want to edit something (or gedit if you prefer)

Handy vim commands:

Quit and write: **:q :w :qw**

Insert-mode: **press <i>, esc to exit insert mode**

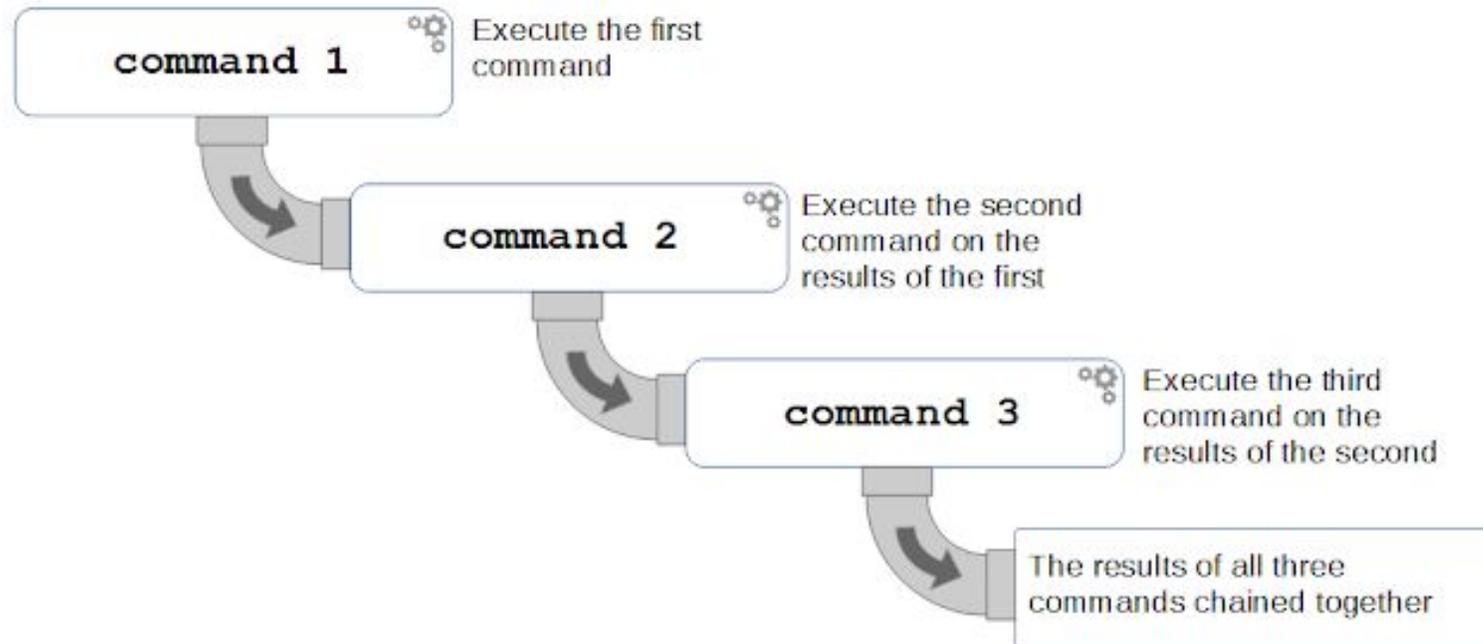
Search: **/<string>**

Command	Function	Relevant Examples
less	View contents of file	"less text.txt"
vi	View / edit contents of file	

Piping and redirects

In the terminal you can chain together programs using the pipe-symbol to make them work together (the output of the first is given to the second and so fourth).

command₁ | command₂ | command₃



(<https://www.d3noob.org/>)

Command	Function	Relevant Examples
	Chain stdout to stdin	"ll head grep root"
< >	Redirect stdout / stdin	"ll > contents.txt"

Chaining programs to manipulate text in the terminal

Let's take it up a notch and start using this one some real data using inbuilt linux text manipulation programs

cd ~/practical/1/interpro

<https://interproscan-docs.readthedocs.io/en/latest/OutputFormats.html#example-output>

Command	Function	Relevant Examples
cat	Print contents of file	"cat file.txt"
cut	Cut out column based on delimiter	"cat file.txt cut -f 1"
grep	Search for lines containing pattern	"cat file.txt grep PATTERN"
sort	Sort lines	"cat file.txt grep PATTERN sort -c"
uniq	Find unique lines	"cat file.txt sort uniq"
head	Print first X lines of lines	"cat file.txt head"
tail	Print last X lines of lines	"cat file.txt tail"
wc	Word count	"Ls -l wc -l"

Maintaining control of your resources

Obviously, your VM does not have an infinite amount of resources. Sometimes it can be handy to know how much disk space you have left, or how much memory you are using.

If you surpass f.ex memory available, memory will get swapped to disk and your VM will slow down immensely.

Command	Function	Relevant Examples
df	Overview of disk space usage	“df”, “df -h .”
free	Overview of memory usage	“free”, “free -m”
top	Overview of processes running	“top”

Controlling your processes

It can be confusing and tedious to have many terminal windows activate at once. You can run processes in the background with appending “&”, detaching them from your terminal.

You can administer / select different jobs with + / - and %

Command	Function	Relevant Examples
Ctrl + Z	Put process to sleep in background	“sleep 50” then “Ctrl + Z”
jobs	List all user submitted processes	“jobs”
ps	List all user processes	“ps” “ps u”
kill	Terminate a process	“kill 12345”
bg	Run last process in background	“bg” “bg % 3”
fg	Return last process to foreground	“fg” “fg %2”
Append &	Run process in background	“sleep 10 &”
disown	Detach process from terminal	“disown -a”

Getting everything to work in Linux can be frustrating

“I don’t understand, it worked last week?!”

Day 1: Installing and running Program 1	Day 2: Installing and running Program 2	Day 3: Running Program 1 again...
Program 1 Numpy 1.10.1 Scipy 1.1.0 Zlib 1.0	Program 2 Numpy 1.15.4 Scipy 1.1.0 Zlib 1.0	Program 1 Numpy 1.15.4 Scipy 1.1.0 Zlib 1.0
Works fine!	Works fine!	“!#%!#”
Mood: 	Mood: 	Mood: 

Introduction to Anaconda

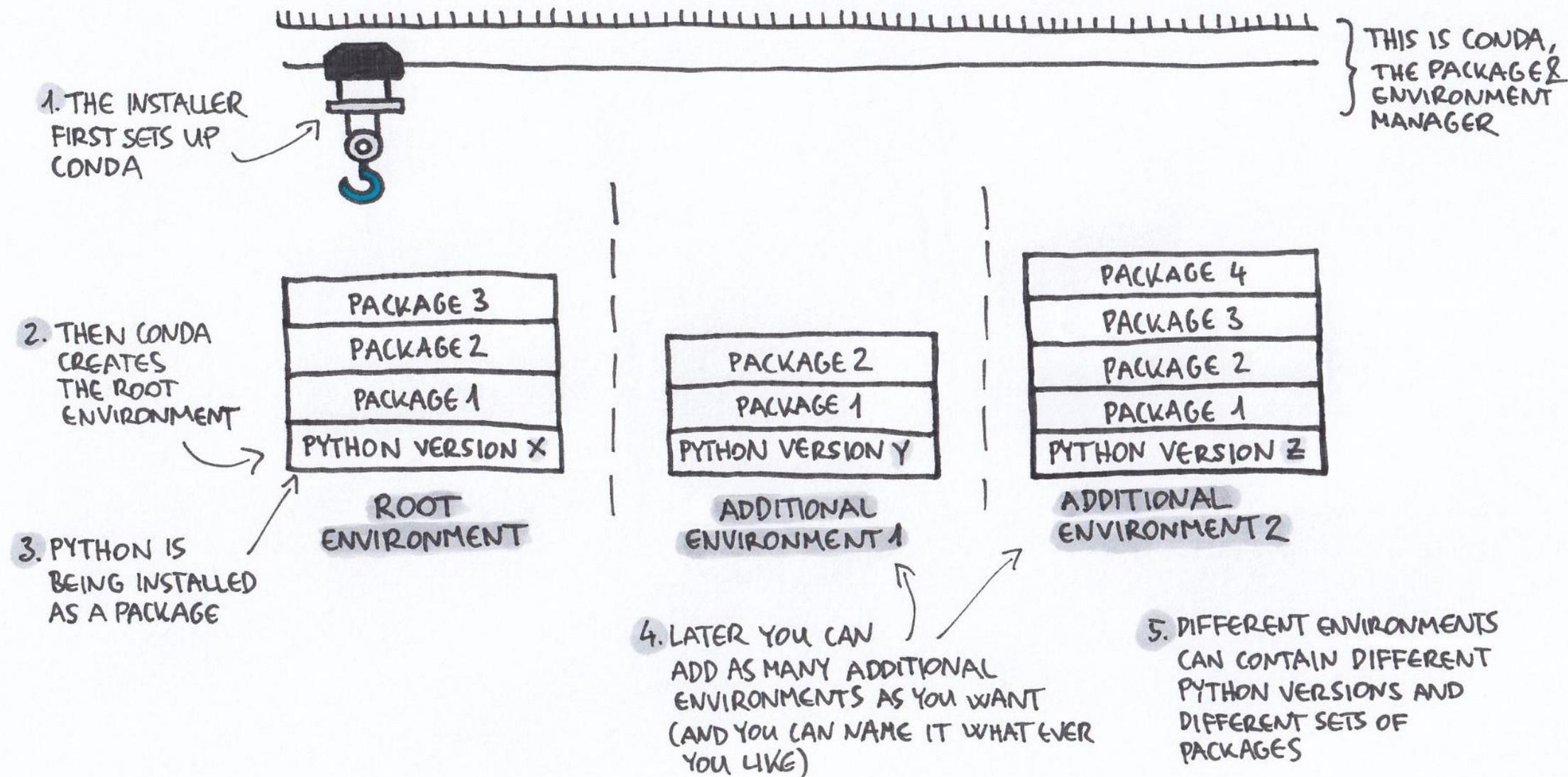


Anaconda is an environment and package manager which makes it easier to install and manage software

Most software require certain dependencies to run. These are also installed automatically with anaconda

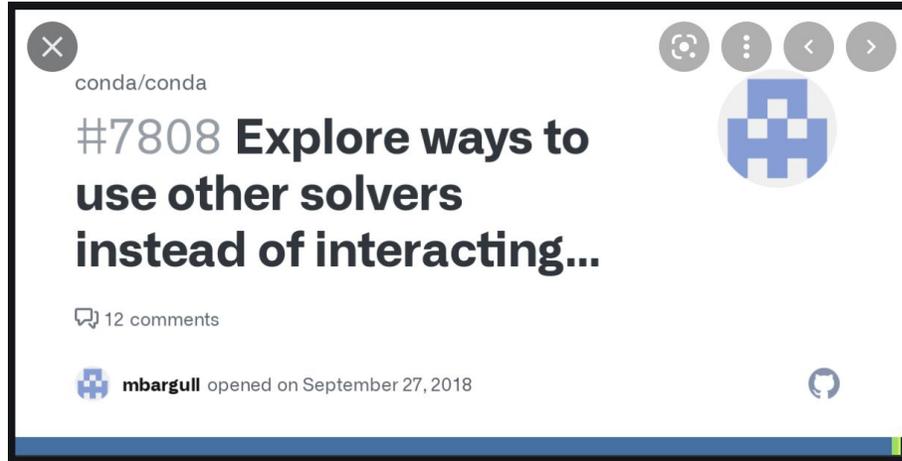
You will be using Anaconda to switch between different environments which contain different tools during this workshop

How Anaconda works



But Anaconda has a problem

... the infamous SAT-solver



... which is why sometimes, we need to call in its quicker little sibling - Mamba



(<https://www.renovablesverdes.com/en/black-mamba/>)

Introduction to Anaconda

- **Checkpoint 3:** Let's play around a bit with Anaconda
 - `conda activate <env>`
 - `conda deactivate`
 - `conda env list`
 - `conda list`
 - `conda search -c <channel> <package>`
 - `conda create -n <name> -c <channel> <package1> <package2> ...`
 - `mamba create -n <name> -c <channel> <package1> <package2> ...`