Containerisation

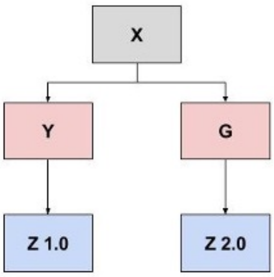
What's the problem and why do we need to use them in data science?

Dependencies

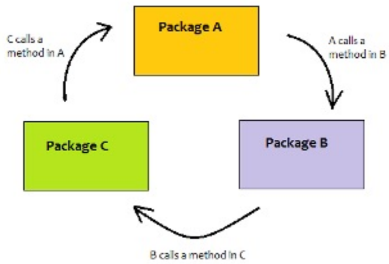
- Library or piece of code that's essential for part of code to run
  - Direct dependencies are libraries your code directly references
  - Transitive dependencies are libraries your dependencies use

Dependency hell!

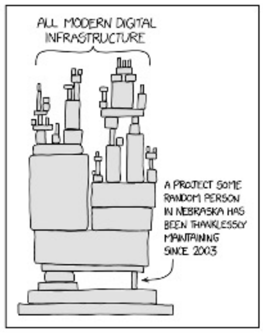
Conflicting dependencies



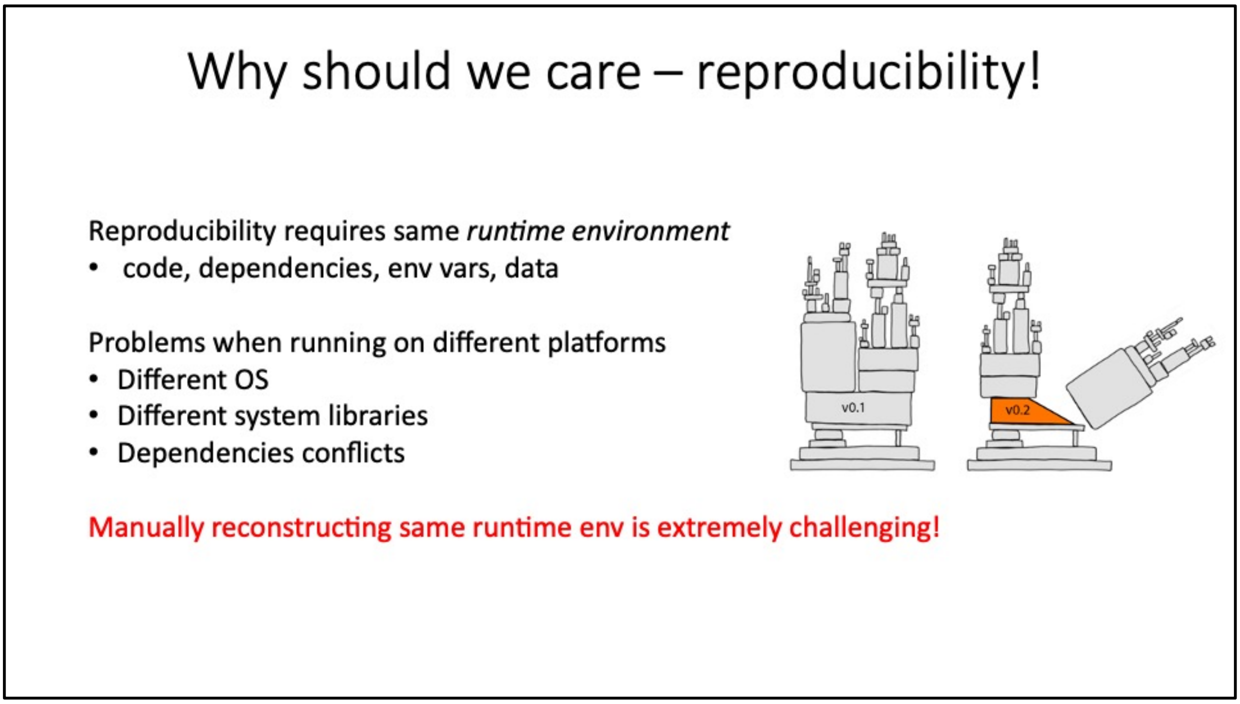
Circular dependencies
(changing any breaks rest)



Library becomes
abandoned over time



Dependency hell is a phrase used to describe problems encountered when dealing with dependencies



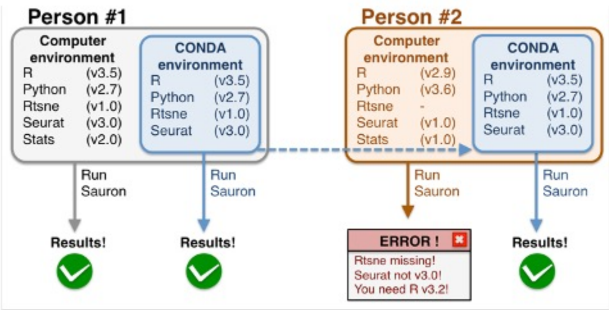
Reproducibility requires recreating the same runtime environment

This includes your code, the required dependencies, the environmental variables and of course your data.

Recreating this runtime env can be challenging
certain packages might be OS specific
system libraries might be different versions or not available
Dependencies might conflict with packages already installed on the system

Package managers- Conda

- Package and environment manager for R and python
- Multiple virtual environments on a single system
- Different package versions for different projects
- Environment file used to share environment



However…

OS distributions do not always share same system libraries

↓

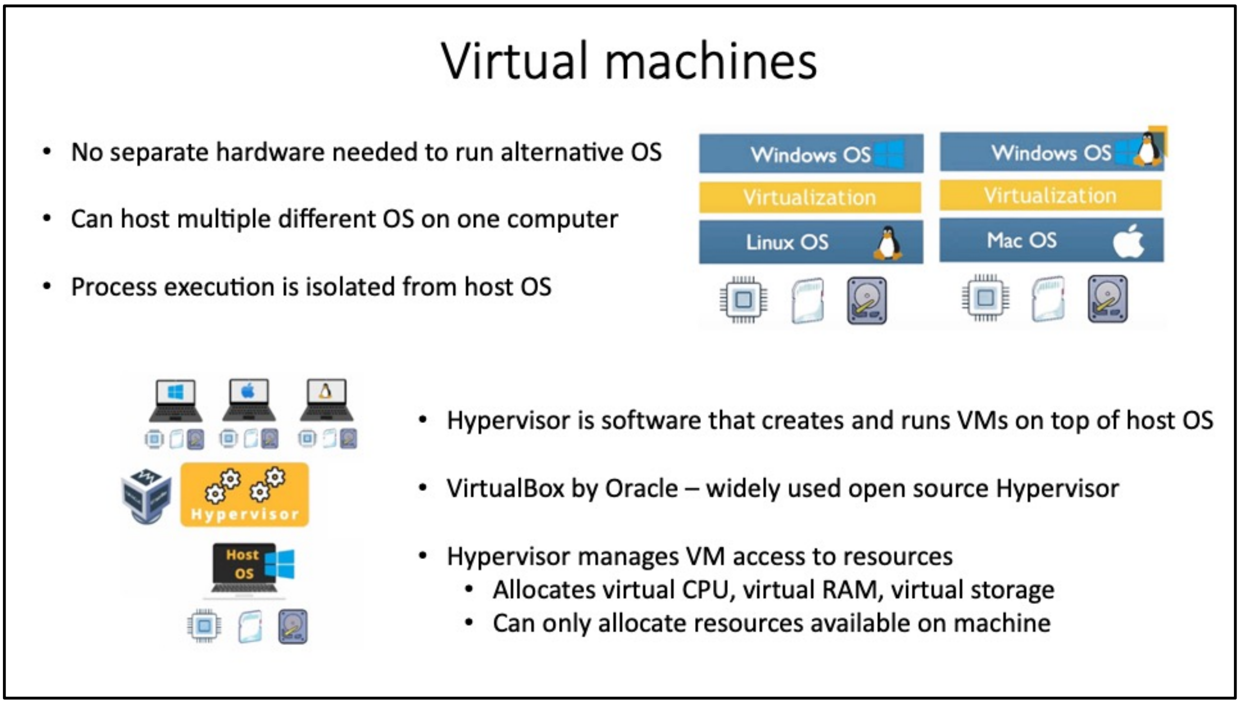
Problems when installing conda envs across platforms

One way of dealing with package installations is through the use of package managers.

The main one I want to talk about is conda, even though there are many different package managers available.

Conda deals with package dependencies
These help deal with dependency intallations and also

OS-level dependencies can cause issues when installing conda envs

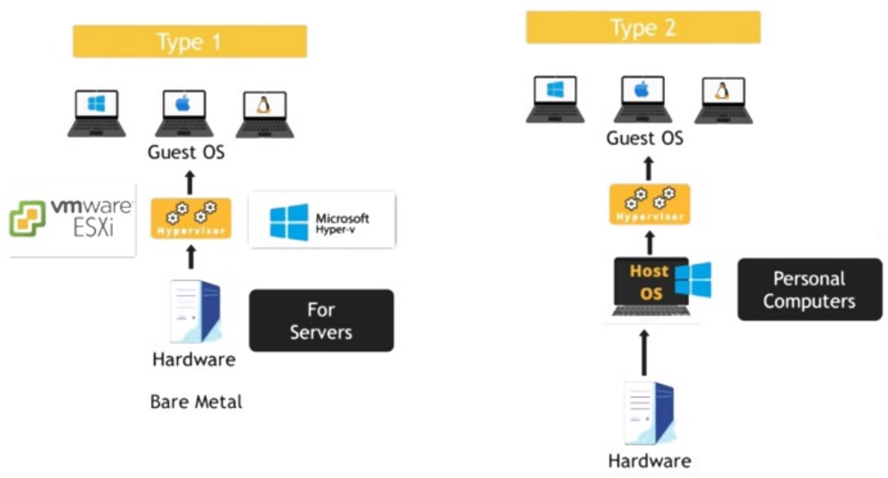


Hypervisor – piece of software that pretends to be a computer

VMs are just files – can be easily distributed

VM has to go through the hypervisor -

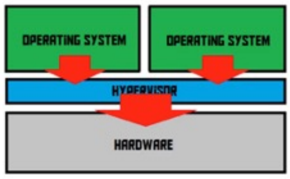
Hypervisors



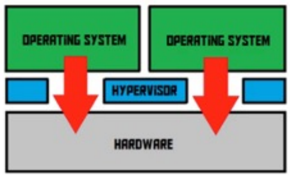
Hardware assisted virtualisation

- First introduced in 1972 by IBM (VM/370)
- CPU natively supports virtualisation
- Hypervisor is still present, but VMs send instructions directly to CPU

Full virtualisation



Hardware assisted virtualisation



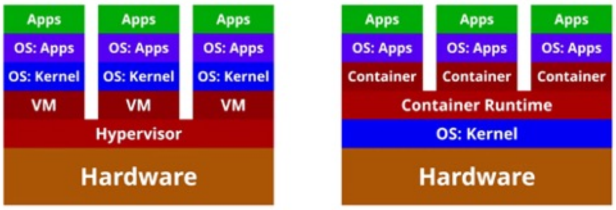
Benefits of VMs

- Efficient usage of hardware resources – partition virtual resources
- If VM breaks doesn't affect host OS
- Abstraction of OS from hardware – VM image is portable
- Reproducibility is independent of host OS
- Run applications which only run on one OS

VMI snapshots

VMs vs Containers

- VMs are virtualisation of the hardware
- Containers are virtualisations of the OS
- Containers are lightweight – utilise host kernel
- Much faster to run a container than a VM – no need for OS to start up
- Resources not used by a container are shared amongst rest

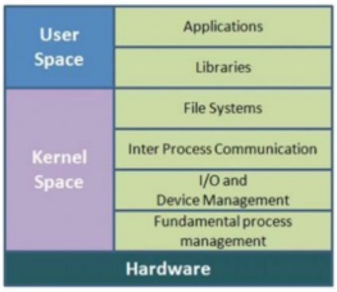


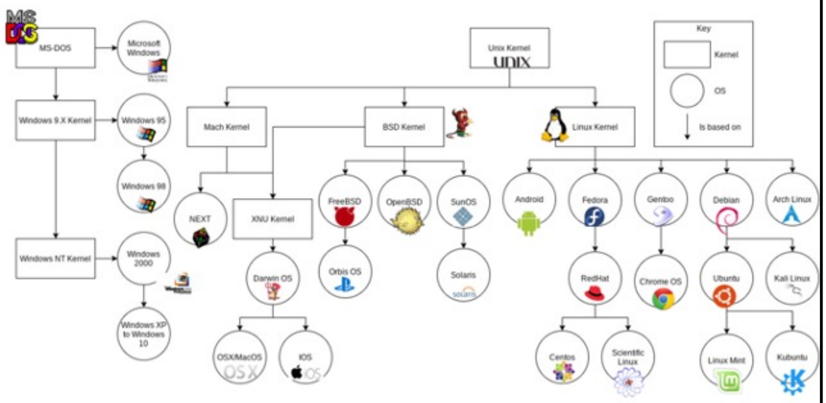
Virtualisation **Containerisation**

i.e. you cant run a windows container on linux or visa versa, but you can mix containers from different linux distributions

Kernel

- Containers need to be compatible with the OS kernel
- Kernel is software at core of OS
- Complete control hardware
  - manages CPU + memory as well as device drivers





How do OS’s control the execution of processes

Containers

Container runtimes

Software that runs and manages containers

Many types: Docker, Podman, Singularity, CharlieCloud, CoreOS

Docker dominates market, but usage is dropping – 99% in 2017; 83% in 2018; ~50% in 2021

Docker daemon requires root access

Docker CLI or API to run and manage containers goes through daemon

HPCs are shared computing environments which do not allow sudo access

Podman + Singularity do not rely on daemon:

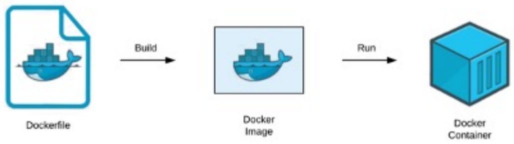
- No single point of failure
- Can run completely rootless

Open container initiative (OCI)

Open industry standards for OS level virtualisation set up in 2015

Allow for increased interoperability between container runtimes – podman and singularity can pull, convert and run docker images

How Docker works



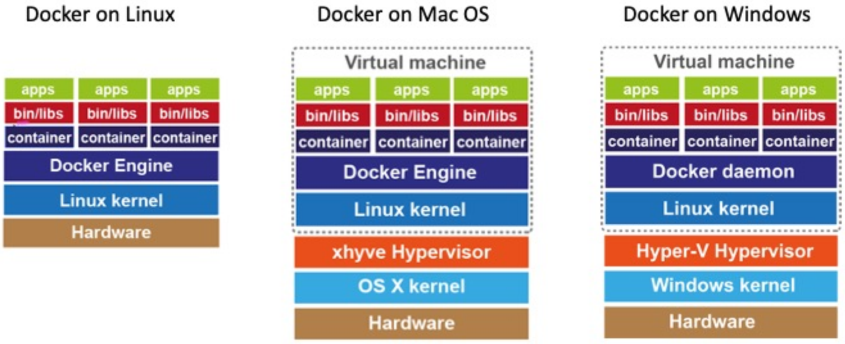
Dockerfile – instruction set for building docker image

Image – collection of files and meta data (blueprint of container)
- Made up of layers
- Layers can add, change and remove files
- Images can share layers to optimise disk usage + memory
- Read only filesystem

Container – read-write instance of an image

Running Docker OS agnostically

Docker runs natively on linux – virtualisation required for windows and OSX



Summary

Package managers

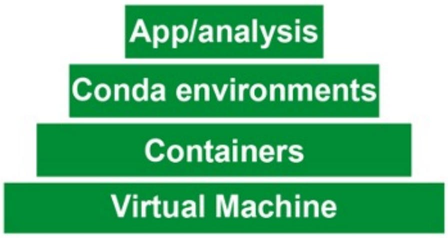
Help resolve package dependencies
Issues can arise with system level libraries

Containers

Virtualisations of the OS
Utilise host kernel but contain their own OS libraries
Lightweight and portable
Host resources can be dynamically shared between active containers
Not fully OS agnostic

Virtual machines

Virtualisations of the hardware
Each VM contains a full copy of the VM
Larger than containers and slower to run
Resource allocation is not dynamic
OS agnostic



ARM – the rise of the M1 chip

ARM PC market share
< 1% in 2020
>8% in Q4 2021

Docker supports multi-arch builds - BUT software binaries need to be available

```
docker buildx build --tag test_image --platform linux/amd64,linux/arm64
```