

Week 3

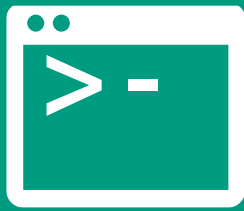
# Operating Systems and Containers



# Agenda



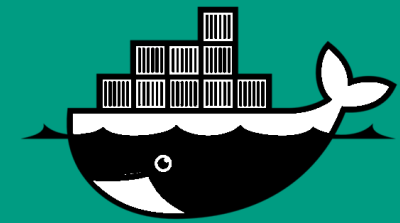
**What is an  
Operating System?**



**Commands**



**Scripts**



**Containers**

# What is an Operating System? (operating system/OS)



# What is an operating system?

System software on the computer

Sample tasks:

- Taking care of the user interface
- Enable multiple applications to run simultaneously
- Allocation of CPU and memory to applications
- Taking care of data storage
- Taking care of network communication
- Taking care of access control
- ...



# What is an operating system?

- Runs directly on the hardware of the computer
- Makes the computer user-friendly.
- All applications run on the operating system
- Ensures that programs work properly and don't crash.



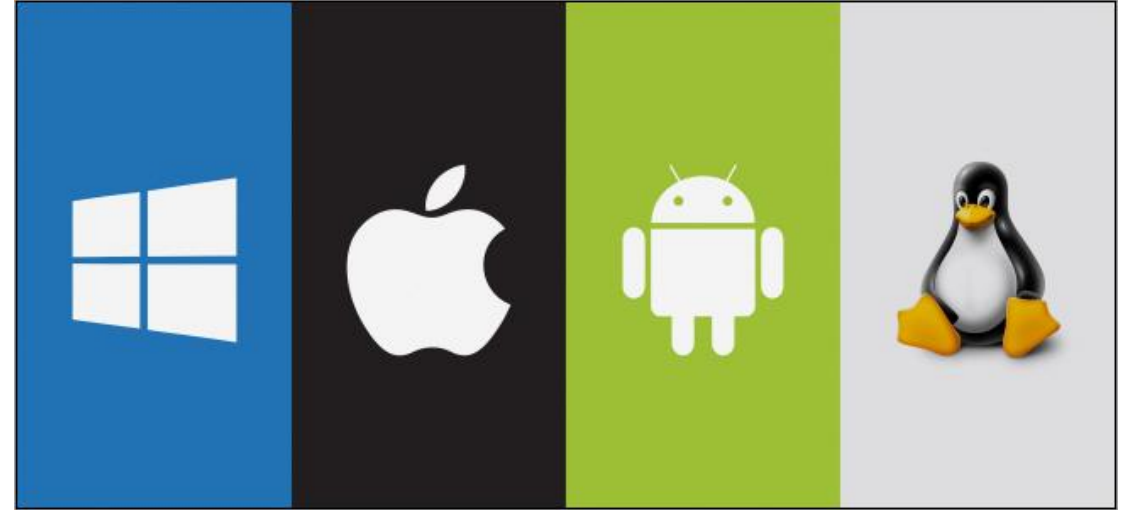
# Most common OSes

## Computers

- Windows 10/11
- Windows Server (2019, 2022 ..)
- MacOS
- Ubuntu ( Linux)
- Debian ( Linux)
- Centos ( Linux)
- Redhat ( Linux )
- Etc.

## Phone

- Android OS
- iOS ( Apple )
- Tizen (Linux)
- Etc.





# Desktop vs Server



# Desktop vs Server

Desktop	Server
Client computers (desktop or laptop)	Server computers
Runs applications	Provides services that can be accessed by clients through the network such as: <ul style="list-style-type: none"><li>• Data storage (file server)</li><li>• Access control and management of client computers (LDAP/Active Directory)</li><li>• Mail server</li><li>• ...</li></ul>
<b>Example:</b> Windows 10/11	<b>Example:</b> Windows Server 2019, Windows Server 2022





# Roles of a server

- **LDAP/Active Directory:** The central database in a network.
- **DNS:** Conversion of names to IP addresses in the network.
- **DHCP:** Distributes IP addresses.
- **File sharing:** Allows users to share files together.
- **Print Services:** Centralizes control of all network printers and places all print jobs in a queue.



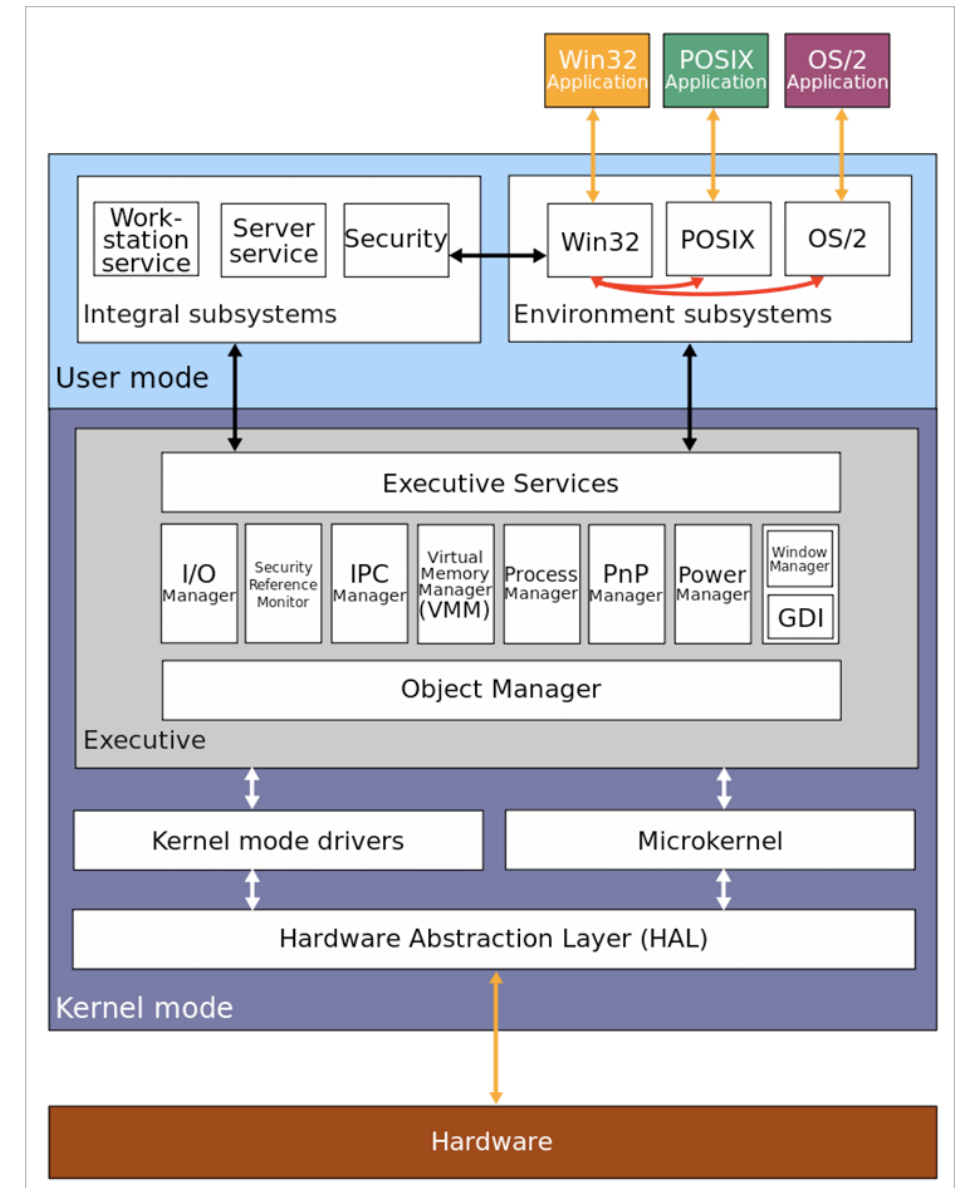
# Windows architecture

## User Mode

- Shielded mode where the user runs programs
- **Executive environment** is a link between the applications in user mode and the kernel functions.

## Kernel mode

- The kernel mode has full access to hardware and computer system resources. It executes code in a protected memory area.
- takes care of the main functions of the operating system



# Linux

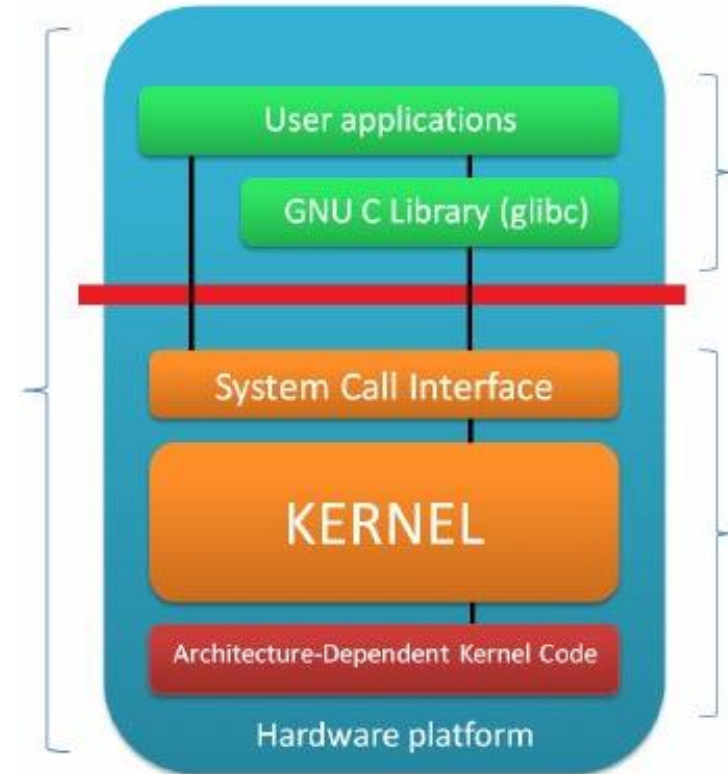
- Multiple distributions use the same Linux kernel.
- The kernel manages the hardware
- Linux is open source. So, it is free to download and adapt to your own taste and environment.
- Ubuntu, Debian, CentOS and RedHat are all unix-like operating systems. (distributions)



# Linux Architecture

- **Kernel:**  
part of the OS that has direct access to the hardware
- **Shell:**  
Program to issue commands to the OS.
- **Applications:**  
Other applications that may be present (such as browser, editor, compilers etc)

Shell and applications run in user space.



# Ubuntu Desktop vs Ubuntu Server (Linux OS)

Ubuntu Desktop	Ubuntu Server
Has Graphical User Interface (GUI)	<b>Does not have a Graphical User Interface (GUI)</b>
Applications like editors, browser, openoffice etc.	Software for e.g. email server, file server, web server etc.

Both use the same kernel: Linux 6.8.0 on Ubuntu 24.04

CANONICAL

ubuntu<sup>®</sup>

Enterprise ▾ Developer ▾ Community ▾

Ubuntu Desktop ›

Download Ubuntu desktop and replace your current operating system whether it's Windows or Mac OS, or, run Ubuntu alongside it.

Ubuntu Server ›

The most popular server Linux in the cloud and data centre, you can rely on Ubuntu Server and its five years of guaranteed free upgrades.



# Applications

Most operating systems like Windows and MacOS also come with built-in programs called applications.

Applications are also called **apps**, programs or software.

For example, programs for:

- Word Processing
- Photo Management
- E-mail
- Video playback
- Web surfing
- etc.



# Types of applications

- Some applications are only developed for certain operating systems
- **Cross-platform applications** can be used on multiple operating systems.

Important requirements running applications:

- The operating system must be up to date.
- Drivers must be installed correctly and in the correct version in the operating system.



Windows applications

Cross-platform applications

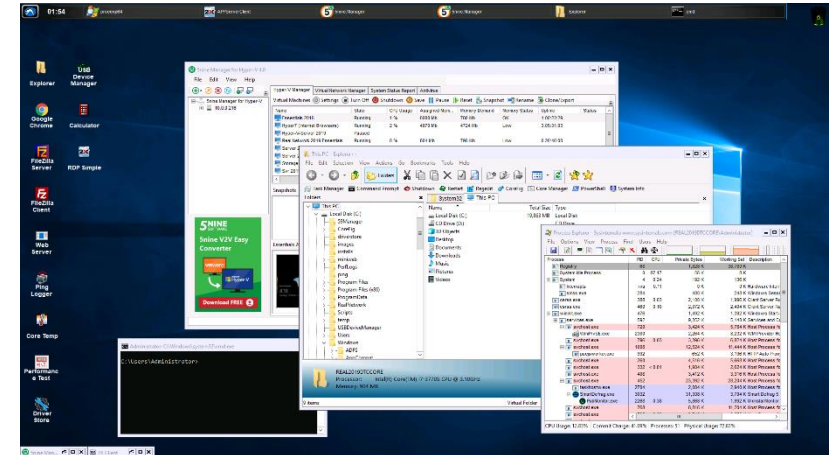


Mac application

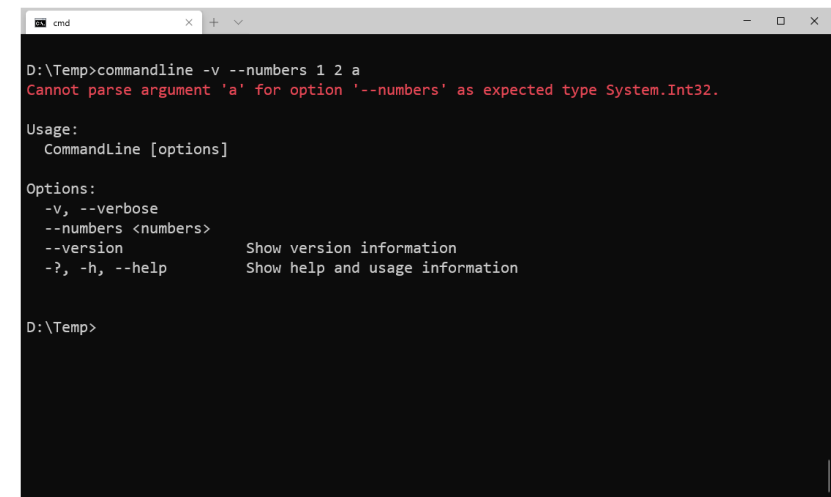
# Configure OS

Setting up and configuring the OS via:

1. Graphical User Interface (GUI)
  2. Commandline
- GUI : user-friendly; however, often many mouse clicks are required.
  - The command line: often more possibilities. Often used by system administrators and IT specialists.

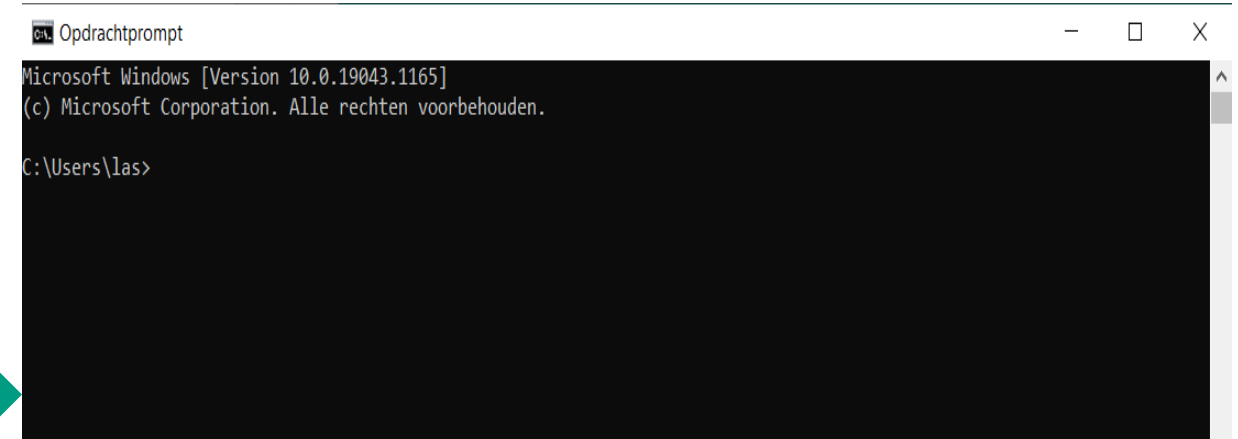
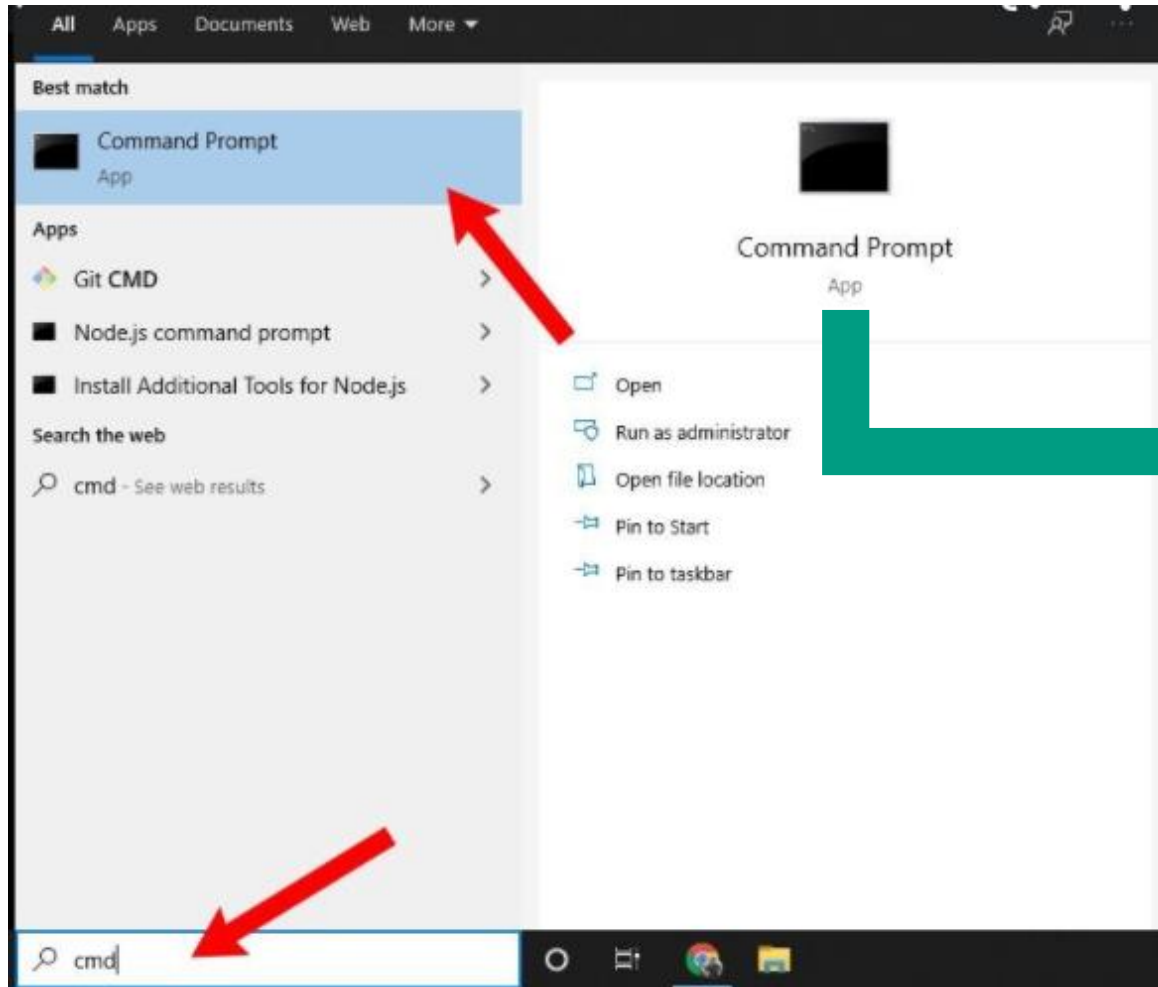


1. Graphical User Interface (GUI)

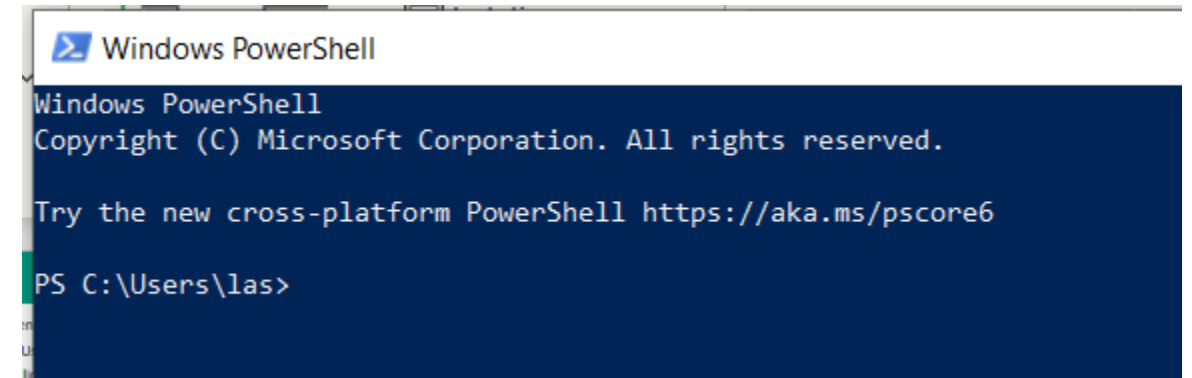


2. Commandline

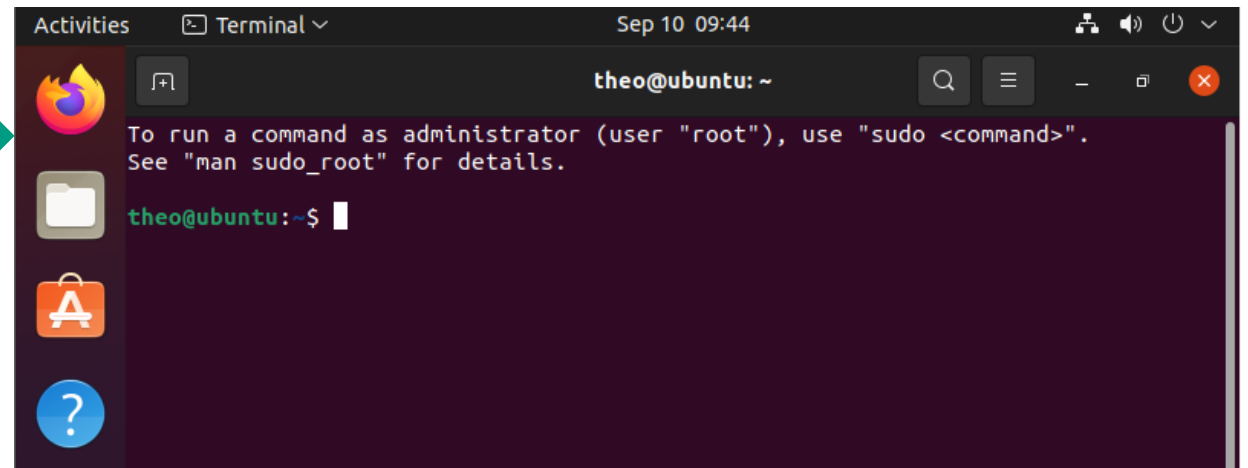
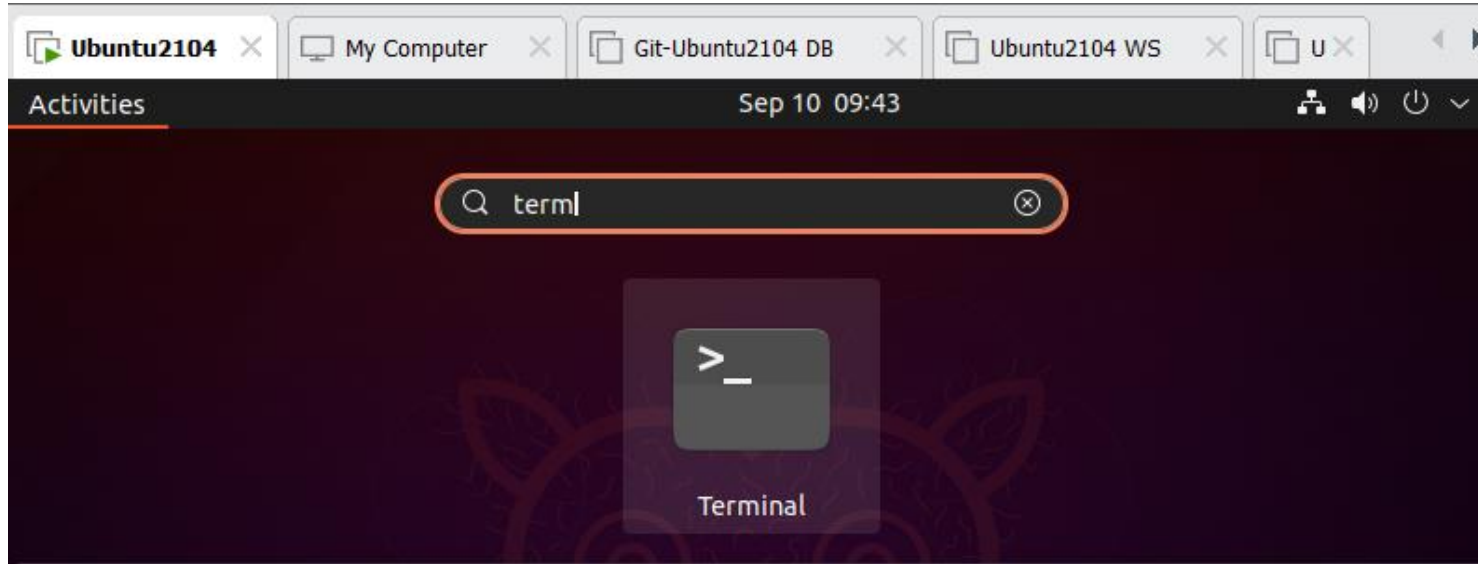
# Windows Commandline



Windows PowerShell: more powerful and advanced.



# Linux Commandline





# Installing Windows software

- Frequent license fees
- Install file manually (MSI file or exe file)
- Possibly downloaded from a specific location on the internet.

# Installing Linux software

- Is done using **Package Manager**
- Differs per **Linux version**

Linux version	Packagemanager
Ubuntu	apt (aptitude package manager)
Debian	dpkg (Debian Package Manager)
Centos, Redhat	yum (Yellow Dog Updater Modified)

- Example commands apt:

**apt update** Updating the package database

**apt upgrade** Upgrading the installed packages

**apt install < packagename>** Install new package (e.g. Firefox)

- Open Source so free!
- Execute centrally via the command line

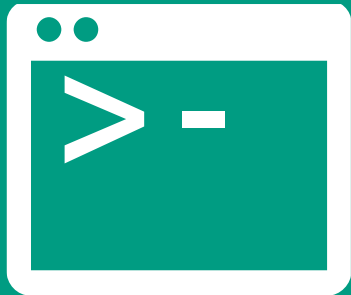
**Note:**

apt was originally apt-get, however, builds since 2014/16 allow apt

# Installing Linux software

- Often '**root rights**' are needed. Someone who is an administrator user can do that:
  - run command as root: put sudo in front of the command (super user do; )
  - first switch to root: command su (switch user)

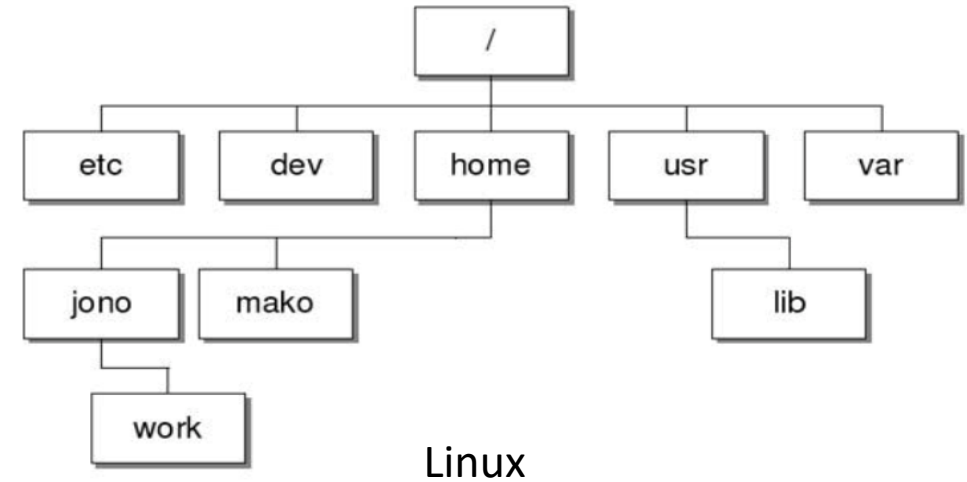
# Windows and Linux Commands



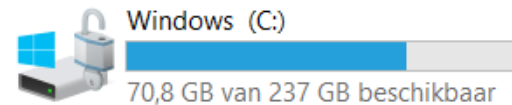
```
...close", "click .collapse"
...function() { var b, d = this, e = this,
...$.ajax({ url: c.router.themePath +
...ready"), a(document.body),
...ready"), c.router.selected,
...this.undelegateEvents(),
...$.ajax({ url: "collapsed").toggleClass(
...togglePreviewDeviceButtons
...keyEvent: function(a) {
...maybeRequestFileSystem,
...$.backbone.View.extend({
...$.listenTo(c.collection, "change",
...length), c.announceSearchResults,
...function() { c.overlay,
...collapse(b)) } } }, render: function() {
...this.renderTheme
```

# Difference file structure Windows and Linux

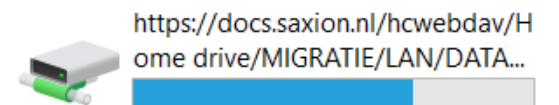
- Windows knows C: and/or D: disk.
- Linux not: file system is 1 "tree" (there can be multiple disks but they are merged)
- Pathname in **Windows**:  
c:\Users\Pieter\myfile
- Pathname in **Linux**:  
/home/jono/work/myfile



## Apparaten en stations (2)



## Netwerklocaties (2)



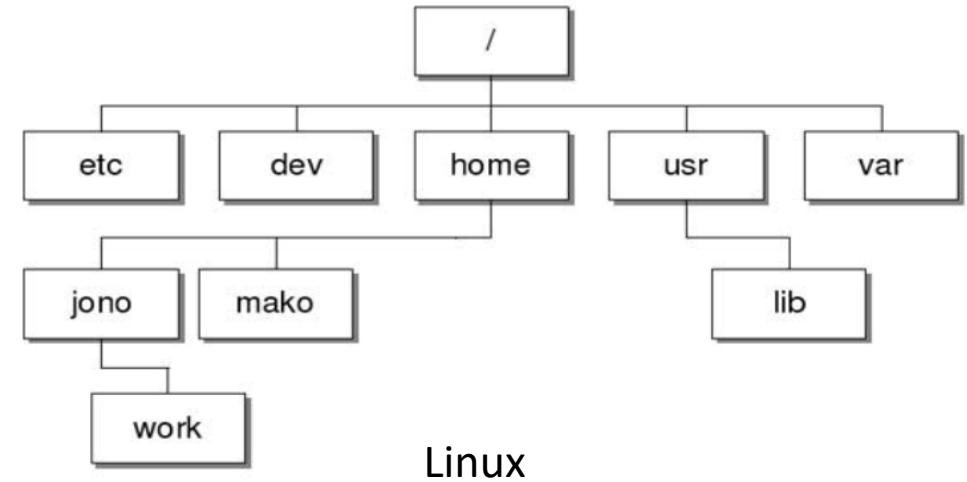
Windows



# Difference file structure Windows and Linux

## Important!

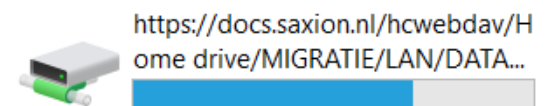
- **Linux:** difference between uppercase and lowercase letters
- **Windows:** no difference



### Apparaten en stations (2)



### Netwerklocaties (2)



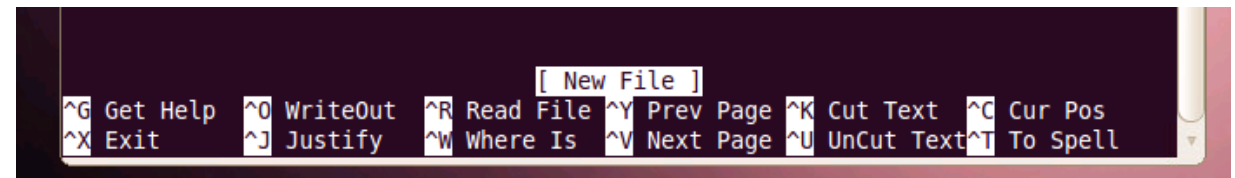
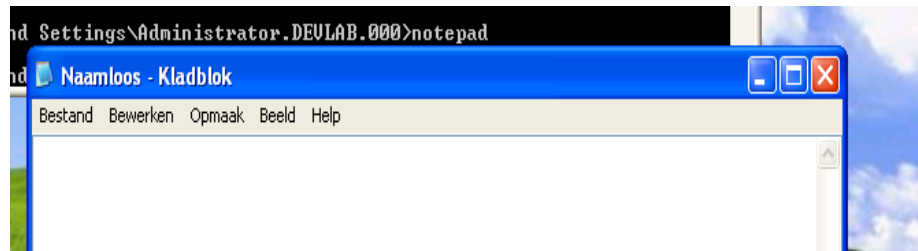
Windows

# Commands Windows - Linux

Commands	Windows	Linux
List all files and directories in the current directory	dir	ls (ls -l gives extra information (- l = long))
Create directory	mkdir <dirname>	mkdir < dirname>
Query current directory name	cd	pwd

# Create and view text file

Commands	Windows	Linux
Create file with editor	notepad <filename>	nano <filename>
View file content	type <filename>	cat <filename>



# Directory

Commands	Windows	Linux
Create directory	<b>mkdir</b> < dirname>	
Remove directory		
Change directory		
	<b>cd ..</b> (go up 1) <b>cd &lt; path&gt;</b> (go to < path>)	

# Copy files

Command	Windows	Linux
File Copy	<ul style="list-style-type: none"><li>• <code>copy &lt;file1&gt; &lt;file2&gt;</code></li><li>• Example with full pathname:  <code>copy c:\etc\test c:\etc\test2</code></li></ul> <p><b>Windows is not case sensitive (!!)</b></p>	<ul style="list-style-type: none"><li>• <code>cp &lt;file1&gt; &lt;file2&gt;</code></li><li>• Example with full pathname:  <code>cp /etc/test /etc/test2</code></li></ul> <p><b>Linux is case sensitive (!!)</b></p>



# Change file or directory name

Command	Windows	Linux
Change the name of the file or directory	<b>rename</b> < oldname> <newname>	<b>mv</b> < oldname> <newname>  <i>(mv stands for move)</i>

# delete files

Command	Windows	Linux
Remove	<b>del &lt;filename&gt;</b>	<b>rm &lt;filename&gt;</b>  <i>rm stands for remove</i> <ul style="list-style-type: none"><li>• Powerful: <b>rm -r &lt;dirname&gt;</b> Removes directory with all contents and underlying directories.</li></ul>

# Time retrieval

- Windows:  
time

- Linux:  
date

Command time is  
for performance  
measurement on Linux:

- `time ls` (measures how long ls lasts).

```
C:\>time
Huidige tijd: 12:13:58,63
Voer de nieuwe tijd in:
C:\>
```

```
Documents  examples.desktop  music
lansink@ubuntu:~$ date
Mon Sep  9 12:13:31 CEST 2013
lansink@ubuntu:~$
```

```
lansink@ubuntu:~$ time ls
Desktop    Downloads      homes  Pictures  Templates  tt
Documents  examples.desktop  Music  Public    test       Videos

real    0m0.004s
user    0m0.008s
sys     0m0.000s
lansink@ubuntu:~$
```

# Wildcards Windows

- Wildcards are characters to complete names.
- Windows:
  - ? Replaces 1 character
  - \* replaces any number of characters but does not replace .
- Suppose we have three files:
  - file1, file2 and file3
- `del file?` Deletes all three files.
- `del ?ile?` likewise.

# Windows wildcards

- Suppose we have the files fileaap.txt, filenoot.txt and filemies.txt
- `del file*.txt` deletes all 3 files
- `del *.txt` likewise
- The \* works to the next point or to the end if there is no point.

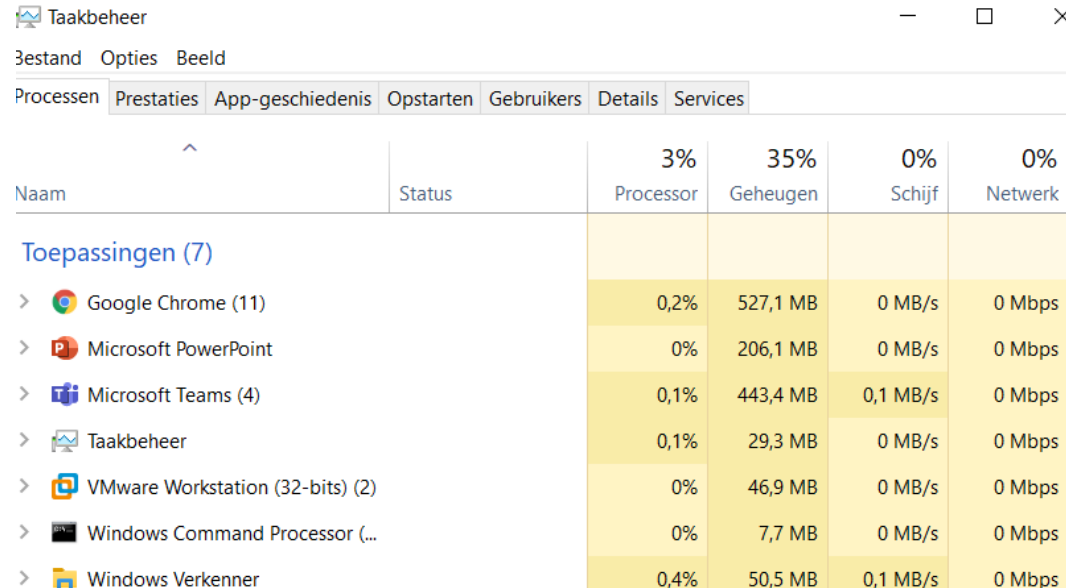
# Linux wildcard

- \* and ?
- ? Replaces 1 character (as in windows)
- \* **is different now:**  
Replaces arbitrarily many characters including .
- Suppose we have the files fileaap.txt.v1,  
filenoot.txt.v1 and filemies.txt.v1
- rm file\* removes all 3 files
- rm \*.v1 likewise



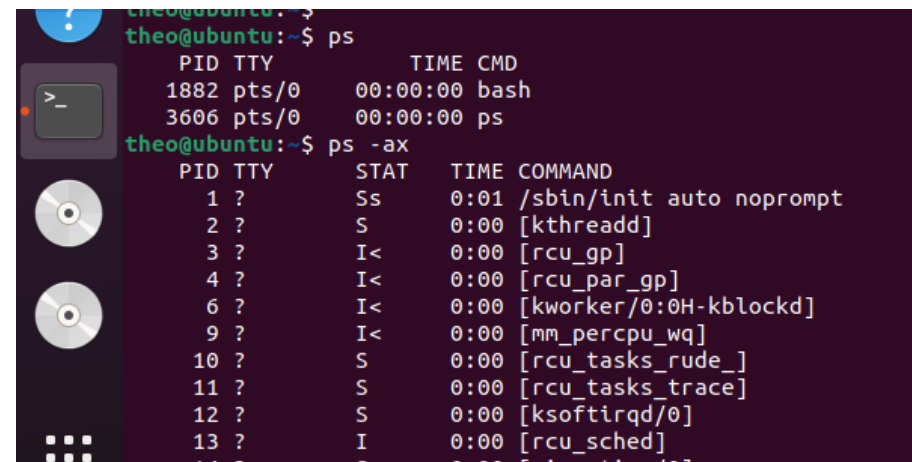
# What processes are running?

- Sometimes it is useful to see what processes are running.
- Windows  
<CTRL><ALT><DEL>  
task manager
- Linux:  
ps or ps -ax  
top or htop



The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It lists several running applications with their status, CPU usage, memory usage, disk usage, and network usage.

Naam	Status	3% Processor	35% Geheugen	0% Schijf	0% Netwerk
<b>Toepassingen (7)</b>					
> Google Chrome (11)		0,2%	527,1 MB	0 MB/s	0 Mbps
> Microsoft PowerPoint		0%	206,1 MB	0 MB/s	0 Mbps
> Microsoft Teams (4)		0,1%	443,4 MB	0,1 MB/s	0 Mbps
> Taakbeheer		0,1%	29,3 MB	0 MB/s	0 Mbps
> VMware Workstation (32-bits) (2)		0%	46,9 MB	0 MB/s	0 Mbps
> Windows Command Processor (...)		0%	7,7 MB	0 MB/s	0 Mbps
> Windows Verkenner		0,4%	50,5 MB	0,1 MB/s	0 Mbps



```
theo@ubuntu:~$ ps
  PID TTY          TIME CMD
 1882 pts/0    00:00:00 bash
 3606 pts/0    00:00:00 ps
theo@ubuntu:~$ ps -ax
  PID TTY          STAT TIME COMMAND
    1 ?           Ss    0:01 /sbin/init auto noprompt
    2 ?           S      0:00 [kthreadd]
    3 ?           I<    0:00 [rcu_gp]
    4 ?           I<    0:00 [rcu_par_gp]
    6 ?           I<    0:00 [kworker/0:0H-kblockd]
    9 ?           I<    0:00 [mm_percpu_wq]
   10 ?           S      0:00 [rcu_tasks_rude_]
   11 ?           S      0:00 [rcu_tasks_trace]
   12 ?           S      0:00 [ksoftirqd/0]
   13 ?           I      0:00 [rcu_sched]
   14 ?           S      0:00 [migration/0]
```



# Linux vs Windows commands in a row

Linux cmd	Windows cmd	Meaning
cat < filename>	type < filename>	Show file contents
cd < path> cd ..	cd < path> cd ..	Go to directory ; to parent directory
cp <file1> <file2>	copy <file1> <file2>	Copy <file1> to <file2>
ls, ls -l	dir	List files and directories (-l = additional info)
mkdir < dirname>	mkdir < dirname>	Make directory < dirname>
mv <file1> <file2>	rename <file1> <file2>	rename <file1> to <file2>
nano <file1> pwd ps -ax, top, htop rm <file1> rmdir <dir1>	notepad <file1> cd <CTL><ALT><DEL> task list del <file1> rmdir <dir1>	Edit <file1> Show path of current directory Show current processes Delete <file1>. Delete dir1 if empty



# Activity

1. Open the Linux Terminal
2. Show the pathname of the directory you are currently in.
3. Create a directory "myfiles"
4. Go to this directory
5. Make a file "myfile" with the content "This is the first line".
6. Make 2 copies of the file "myfile" and name them "myfile.cp1" and "myfile.cp2".
7. View the contents of the file "myfile.cp2".
8. Change the name of the file "myfile" to "myfile.org".
9. Discard both copies with one command.
10. Throw away the file "myfile.org".
11. Go up a directory.
12. Discard the directory "myfiles".

Scripting (combine  
commands in file)



```
query( );  
have_posts() ) : $popularposts  
posts">  
function_exists("has_post_thumbnail")  
$col-md-6 col-sm-6 col-xs-12 sidebar-  
the_post_thumbnail( '', array( 'class'  
endif; ?>  
div class="col-md-6 col-sm-6 col-xs-12 sidebar-  
<h3><a href="<?php the_permalink()>  
<p> <?php echo digital_news($post->post_title);
```

# Why scripting?

Suppose we have to issue many commands in succession (and again several times on e.g. different virtual machines).

Disadvantage of doing that all the time:

- Frequent typos
- Takes a lot of effort

## Solution:

- Put all the commands together in a file (we call it a script or batch script).
- Run that script with just 1 command.

### Example:

```
sudo apt update
```

```
sudo apt upgrade
```

```
sudo apt install <package1>
```

```
sudo apt install <package2>
```

# Windows (BAT or CMD file)

- In windows this is a file with name ending in .bat or .cmd
  - (cmd is the updated version and is used for Windows NT systems)
- Example: `myfirstbatfile.cmd`
  - `cd c:\users\pieter`
  - `mkdir myfiles`
  - `cd myfiles`
  - ....
- Can be started immediately:  
`.\myfirstbatfile.cmd`

# BASH (LINUX) SHELL

## File createfiles.sh

```
#!/bin/bash
mkdir myfiles
cd myfiles
echo "This is the first line" > file1.txt
cp file1.txt file2.txt
cp file2.txt file3.txt
cp file3.txt file4.txt
cd ..
ls
```

- Make it executable:  
`chmod +x createfiles.sh`
- Then start it:  
`./createfiles.sh`





# Activity

1. In the Ubuntu VM, create the bash file shown on the previous slide using the nano-editor.
2. Make the file executable and run it as instructed.
3. Check that the output is what you expect.
4. Clean up the created files and directory afterwards.



# Containers



docker



# Containers



## Containers:

**Containers** are lightweight, portable units that **package** an **application** and its dependencies together, allowing it to run consistently across different computing environments.

Unlike traditional virtualization methods that use virtual machines (VMs) with their own operating systems, containers **share the host operating system's kernel** while maintaining isolation between applications. This makes containers more efficient in terms of **resource utilization** and **startup time**.

**Docker** is a popular container runtime that facilitates the creation and management of containers.

# VMs vs Containers



VS



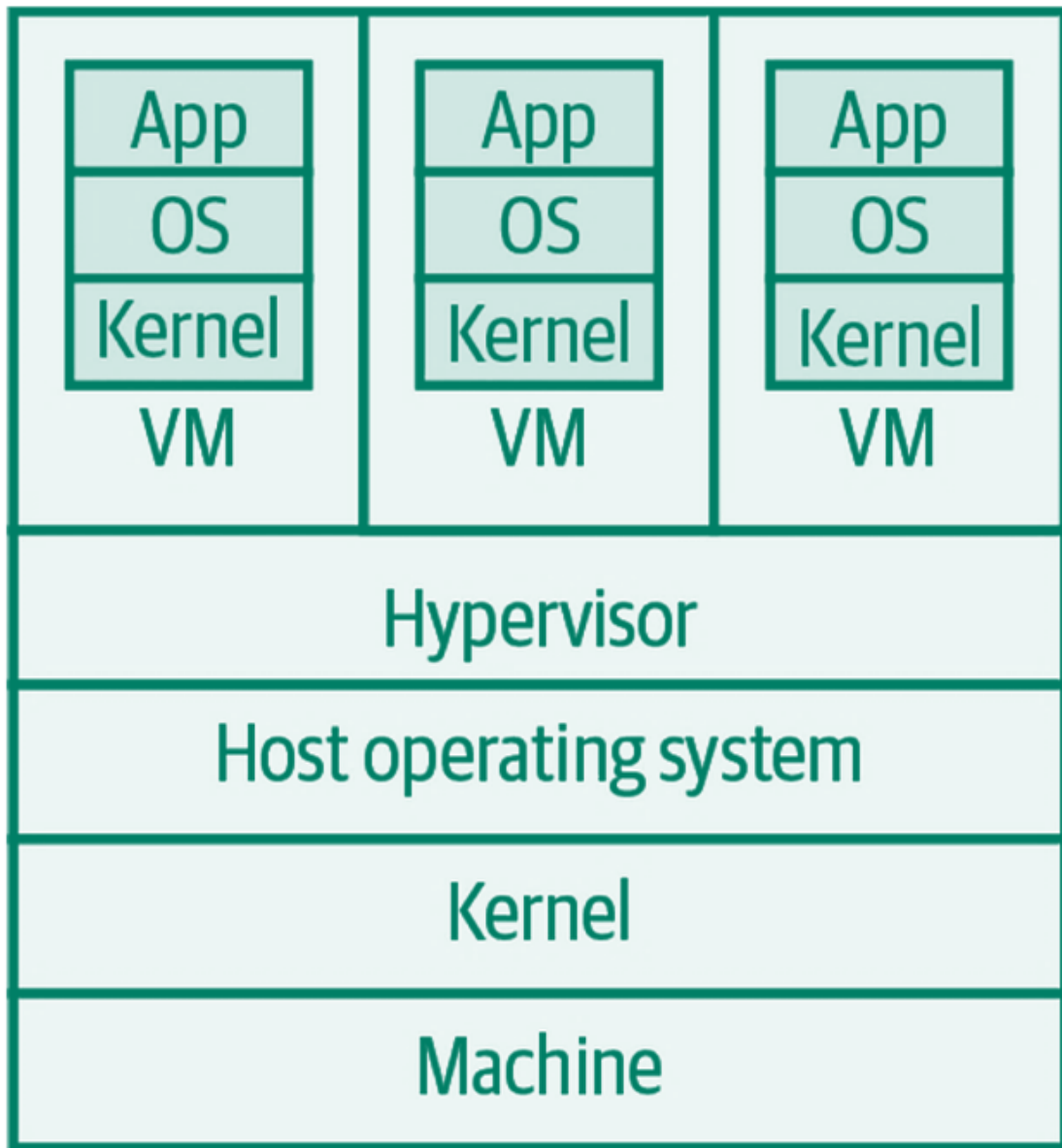
## VMs or Containers:

When choosing between virtual machines (VMs) and containers, it's essential to understand their strengths and ideal use cases.

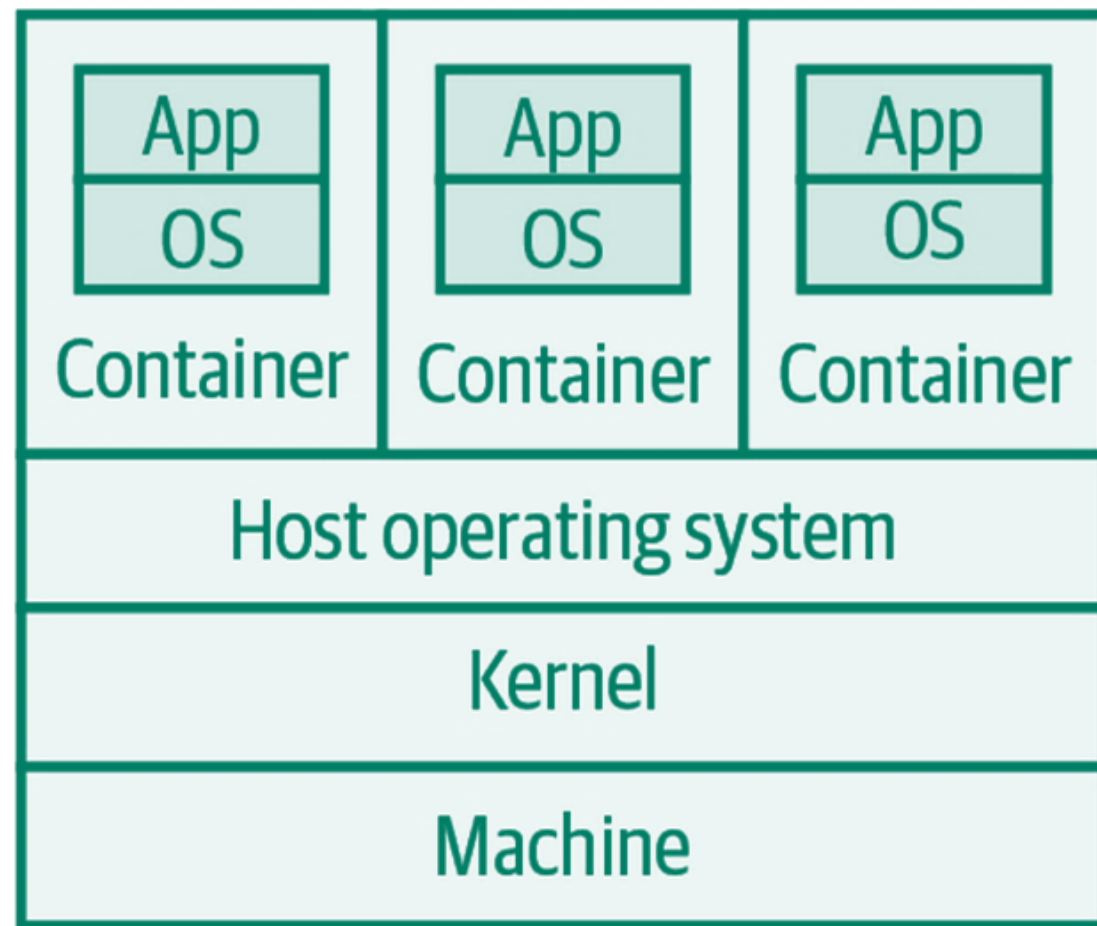
VMs provide a robust solution for scenarios requiring **strong isolation** and **support for multiple operating systems**, making them suitable for legacy applications with specific configurations.

Containers excel in environments focused on microservices architecture and rapid development, as they offer lightweight, fast startup times and efficient resource utilization by **sharing the host OS kernel**. Containers ensure a consistent environment across development and production, making them perfect for situations where scalability and resource optimization are key.

# VMs vs Containers



Type 2 virtualization



Container-based virtualization



# Working on the case





# Case

- Do the assignments of week 3

Please consult the assignments document and the template report for more details.

Any questions?

