

## **Roles in software development :**

1. programming
2. Software testing
3. Release code and compress it and pass it to the servers .
4. Upgraded

**Two main :** developers and operation team .

**Challenges for operational team :** application should not have a down time when upgrading in the servers , if servers die application should still be running , able to handle huge traffic ,

**Developers knowledge :** programming language , databases, version control , frameworks etc.

**Operational :** OS (mostly linux), scripting, command line , monitoring tools

**Problem here is :** there is no interface btw these two roles , usually development team will give document to the operational team which are not clear or something is missing there . So this takes months to upgrade or introduce new features

## **Solution : DevOps culture >**

DevOps was a concept way of working btw dev and ops .

Common language to communication.

Become its own roles and jobs many new concepts and tools were introduced .

## **DevOps tasks and responsibilities :**

1. Need to know some know how from dev and ops team have (subset of skills from both sides).
2. additional devOps skills
3. Main thing is to build a CI/CD pipeline .  
Commit code ->test->build->push->deploy

## **Traditional waterfall vs Agile :**

Agile is a heart of CI/CD .

## **Operating System and Linux basics :**

OS : how we interact with cpu , memory , storage , io devices .

No application directly talk to the hardware directly they interact with OS (translator).

OS will manage resources among application and isolate content of application .

What are the tasks of OS :

1. managing cpu resources
2. 1cpu can only process 1 test at a time cpu is switching so fast that u don't notice it .
3. Multiple cpu : dual-core or Quad-core more cpus more faster application
4. Memory management : Ram is limited on the computer when memory is out OS swaps memory between application one app become inactive, new one gets resources.
5. This swapping slows down the device
6. Storage management loaded into the memory secondary management .
7. Storing in Structured way folders and directory .
8. Managing the IO devices
9. Security and networking : different users in the same device , assigning ports and ip addresses.

How OS is constructed ?

OS components :

1: kernel (this loads first when turn on the computer ,heart of the Os ,it hold device drivers to interact IO devices to the devices)

Kernel starts the process for app ,allocates resources to the app , clears up the resources

Kernel is a program , consists of device drivers , dispatcher , scheduler , file system etc.

2:Application layer : entOS, debian are different linux distributions ,

Different application layers , but based as same linux kernel.

Android is also based on linux kernel : which is for phone and running in completely different hardware .

Linus kernel is most widely used .

OS : -> ONOtherHand : macOS , IOS is based on different kernel call Darwin.

**OS for servers : mostly based on Linux , more light weight and performant as No GUI or other user application Just have a CLI to interact with a machine .**

Eg: mobile OS : bottom most hardware , then linux kernel , then Android application layer , user interface .

3 Main OS : linux , windows , macOS ,

Kernel remain the same but the application layer is developed and improved

**Server OS are more lightweight as noGUI or Io devices now they can use hardware more .**

75 % of the websites uses linux OS

**MacOs vs Linux :**

**Command line , file structure etc. similar how?->** back in 1970 UNiX was a codebase for many different OS unix wis develop independent of linux (1991) , many Os build on top of Unix and one of them was maxOs kernel Darwin in based on unix. To make all the OS that comes by unix to keep them compatible standards were introduced for the application which run of these OS  
Standards : posix(portable Operating system interface) .

Linux was developed in parallel to unix

Created by linus torvalds

No source code of unix

Linux was build with same ideas or philosophy of unix

Clone of unix of also called 'unix like'

There for linux and macOS both POSIX compliant

LINUX is Mostly used OS for servers!

Knowing linux is a must for devOps engineer as you need to work with servers.

Installing and configuring servers , linux native technologies (docker , kubernetes ,ansible)

Where as windows is completely different

**What is Virtualisation >**

No separate hardware needed to install OS .

To can uses a windows OS and using virtualisation u can used Linux OS on top of it or vice versa

By using hypervisor is a technology that allows hosting multiple virtual computer in a physical computer on top of OS u have already installed .

So virtual box is a hypervisor by oracle I, open source ,works on all OS .

VirtualBox takes hardware resources from HOST OS

To create virtual CPU, ram , virtual storage for each virtual machine

You can only give resources you actually have

U are sharing hardware resources of one machine to run virtual machine

Virtual machine are completely isolated it does not even know it is a virtual machine

If something breaks inside VM, it doesn't affect the host machine

### **Benefits of virtual machine :**

Learning and experiment :

Test your app on different OS :

### **Different type of hypervisor:**

**Type2 hypervisor (personal computers)->** on host OS install hypervisor and using virtual box install guest OS

Guest os borrows hardware form the host OS.

**Type1 hypervisor(install hypervisor directly on hardware): bare metal**

**Eg : of type1 hypervisor : vmware ESXI , Microsoft hyper-v (for servers)**

Type1 hypervisor is used by cloud platform eg :AWS ,google cloud etc.

U are creating a physical machine in physical server so these vm are completely isolated so if something happen in one vm it will not effect the other one . (They don't share network , resources share nothing).

What are the benefits for companies using virtualisation >

Efficient usage of hardware resources

Use all the resources of a performant big server

Users can choose any resource combinations.

Now these cloud providers can divide these storage servers resources and rent them for money .

### **Why are companies adopting virtualisation ?**

Abstraction of oS form the hardware

Eg: without virtualisation OS is tightly coupled to the hardware means if the hardware of that os exploded all the application os and all data is gone

One point of failure .

With virtualisation :

Os as a portable file -> virtual machine images

That portable file will contain os. ,all application on it

Since it is a file u can make copy of back up of it (backup of entire os )

snapshot(backup)

Secure very easily , portable , no dependent of any physical server .

### **Install virtualBox hypervisor**

#### **Setup linux ubuntu virtual machine**

Steps: setup environment go to new setup name \_ linux-ubuntu.  
Set type linux version ubuntu 64 bit .set ram memory , hard disk space .  
And now install linux os so to ubuntu download here we are download ing the os image .

Setting advance -> shared clipboard bidirectional to share copy paste this from host to guest vice versa.

Or Drag and drop didirectional

For these to be active we need VM virtualBox extension pack .

Virtual machine has its own network which is completely isolated with host network  
Sharing things is not good for isolation .

### **Linux VS Windows File System :**

Linux file system is hierarchical tree structure .

Windows has multiple root folders .-> a: and B (removable Disks) c : (local disk) , d(data) .

### **Linux file system:**

Home directory of all other users is home and root is the home directory of root user  
Multiple users on compute each user has its own space meaning its own ->Home directory

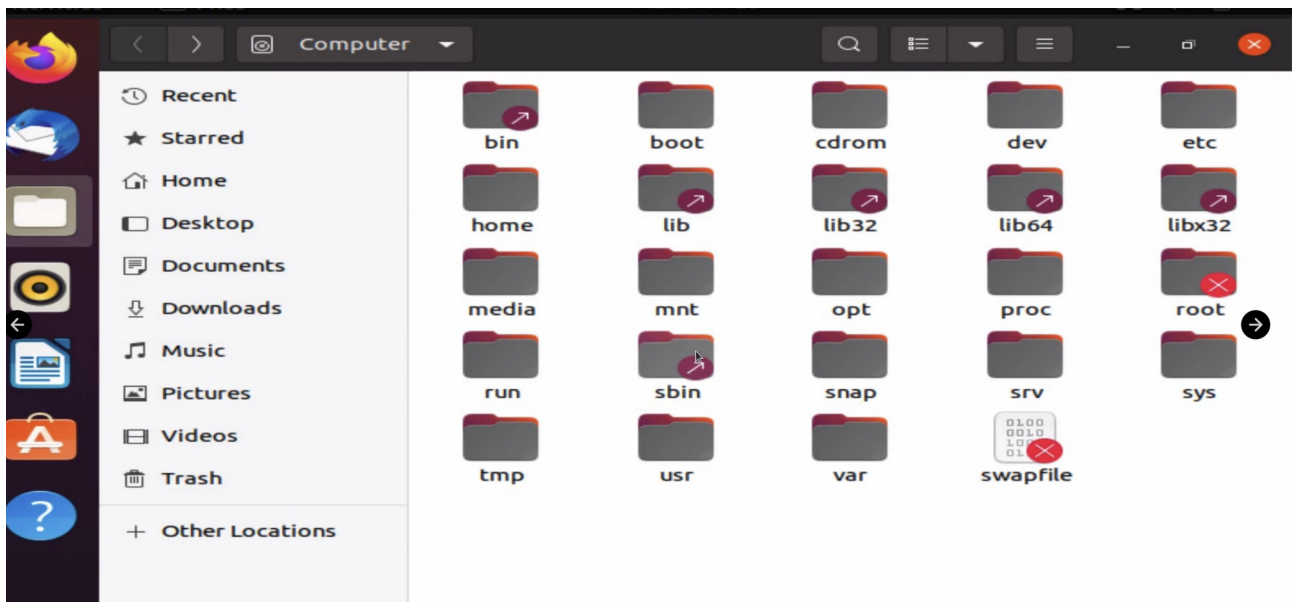
PROGRAM INSTALLED SYSTEM wide, are available for all users on that computer .

1./bin : bin folder inside the root directory contains the most basic command u have available for your OS . Eg: cat(shows u the content of the file) , cp(copy) ,echo etc. to all the user too the system itself.

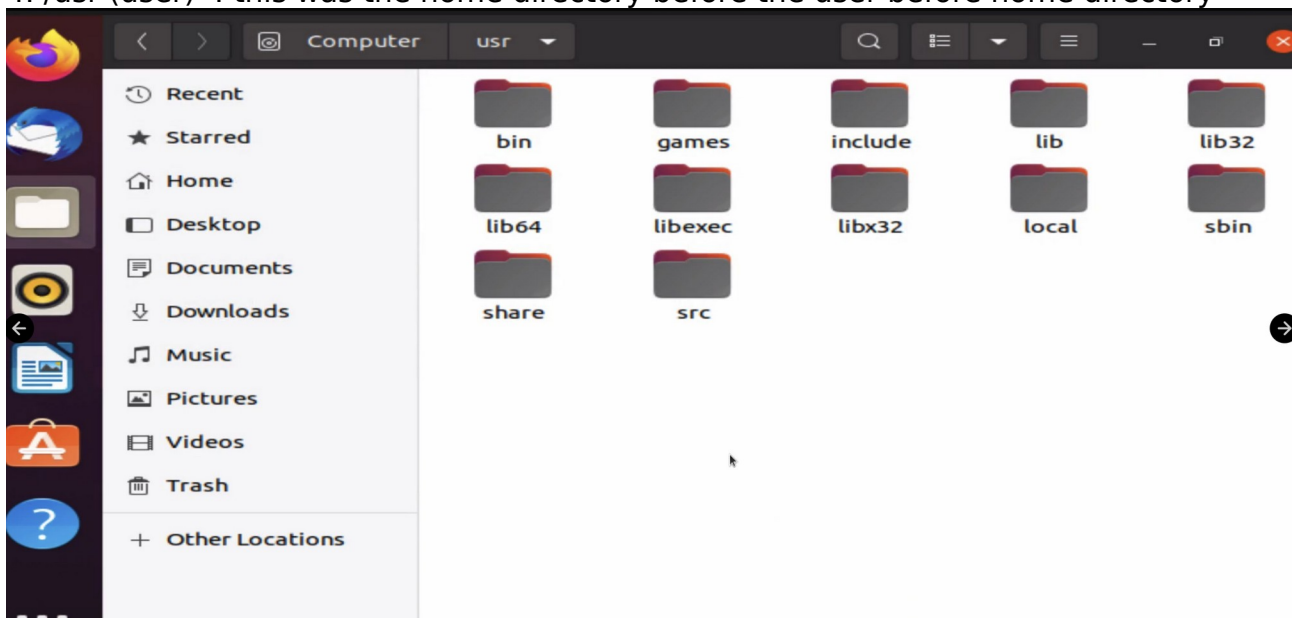
What is binary ? Computer readable format there for all these commands are in bin file.

2./sbin : system binary these commands use super users to operate them . Eg : adducer, change password, u can't use them need superuser privilege.

3./lib : essential shared libraries that executables from /bin or / sbin use .



4. /usr (user) : this was the home directory before the user before home directory



.Insider user folder->

Why this split : because of storage limitation it was split into root binary and user binary folder .

And as there is no storage limitation there is the copy of these bin lib sbin on usr and home

When we use these command commands they are usually executed from the user bin folder.

usr folder also has a local folder which also has bin ,sbin ,lib folder

/usr/local : program that you install in our computer.

: that's where third party application like docker , Minikube ,java .... Will go after installation .

I can't want only application for your self install in home directory .

Eg: if installing java the binary file will go in bin , lib file in lib and so on ....

As user folder is inside the root so anything you install will be available for all the users.

Two location : home and in computer (root)

5. /opt : third party application install .

Diff btw opt and user / local directory => usr/local (programs which split its components .

Opt (program which not split its component .

Eg: IDEs , web browsers are in opt .

Again system wise application available for all the user .

6. /boot : contains files required for booting .

**All folders till now are READONLY folder**

7. /etc : system configuration , network interface user data password we can write them change them . Place where system wide application are stored.  
Emerged for main configuration location .

8. /dev : app and drives will access these not the user .  
Files that system needs to interact with devices.

9. /var /log: contains files in which system writes the data during the course of operation .

10. /var/cache : contains cached data from the application program .

11. /tmp : stores temp files required for some processes are store here temporary .

12. /media : contains subdirectories. When removable media devices inserted in the computer are mounted . Eg: while inserting cd a directory is automatically be created and u can access the content of that cd inside that directory . /div folder also .

13. /mnt : historically ,sys admins mounted temporary file system here .mount file system to your operating system .

### **IMPORTANT POINTS :**

// usually u are not interacting with these folders.

// u will do installing apps with package manager.

// interacting with os installer/package manger is handling this.

// OS is handling this

Eg: plugin USB (automatically mounted in your system /media.

Change configurations /etc.

Install app /bin ,/lib , ..

### **HIDDEN FILES :**

Automatically generated by different application or OS .

File name start with dot.

In unix also called dot files .

Is primary used to help prevent important data form begin accidentally deleted.

Creating hidden folder just add . In start of file name.

Basic linux commands :

ubuntu@ubuntu ~\$ :

Username,computername,~=homedirectory , \$=u are working as a regular user in OS.

If log in as root user # .

1.pwd : print working directory.

2. Ls : list folders and files.  
3. Cd : change directory .  
4. Mkdir : make directory  
5. Touch : to create a file .  
6. Rm : to remove a file .  
7. Cd .. one level up the file system tree.  
8. Rm -r python-project : recursively remove the python directory of folder .  
9. Cd / : will bring u to the root folder .  
10. ~(tilt home directory) we have / (forward slash root user)  
11. /\$ pwd : / (we are in root folder) .  
Every thing in linux is a file .  
Text documents , pictures etc.  
Directories , commands like pwd ls etc

12. Cd usr/local/bin  
13. Cd ../.. or cd /usr ( go back to usr)  
14.cd /etc/network : absolute path more from any where .  
15. Cd ~ : go back to home directory .  
16: from home directory to see the root directory etc/network :  
Ls /etc/network  
17. Cd Doc + tab to auto complet directory name  
18. Mv old\_name new\_name .  
19. Cp -r old\_project new\_project : make a new folder newProject and copy all the content of old project to new project .  
20 cp Read.md readtest.md : copy the file to new file .  
21. Ls -R Documets/ : see the content of that file .  
22 . History to see the command history in current terminal session .  
23. Ctrl + r ( to search the command ) :  
24 . Ctrl + c (to stop the command execution )  
25. Ctrl + shift +v : to copy paste in terminal .  
26. Ls -a : to see the hidden files .  
27. Cat : display file content  
28. ubuntu@ubuntu:~\$ cat Documents/newproject/Readmu.md  
hello

Why to use cli ? Work more quickly , easier for bulk operation , can more than GUI ,

Displaying the OS information :  
Uname -a : show the system and kernel  
Lscpu : to check hardware info  
ubuntu@ubuntu:~\$ cat /etc/os-release

### **Execute commands as superuser > :**

ubuntu@ubuntu:~\$ sudo adduser admin  
[sudo] password for ubuntu:  
Sudo : this tells the system that allow regular user to run the command with security privileges of the superuser or root .

ubuntu@ubuntu:~\$ ls /home (new admin added)  
admin ubuntu

LOGIN TO other users :  
ubuntu@ubuntu:~\$ su -admin  
Password:  
To run a command as administrator (user "root"), use "sudo <command>".

See "man sudo\_root" for details.

```
admin@ubuntu:~$
```

```
admin@ubuntu:~$ pwd
/home/admin
```

```
29.ls .bash_history.
30 cat .bash_history.
```

## **INTRODUCTION TO PACKAGE MANAGER:**

### **How to install software on linux :**

Software package is a compressed archive file , containing all req files.  
Apps usually have dependencies.  
Installing software in linux is more complex as files are spilt in different directories.  
Uninstalling completely is difficult .  
Package manager : downloads , install or update a software from a repository  
Also insure integrity and authenticity of the package.  
Manages all the required dependencies.  
It will know where to put the different files from the packages.  
U can just easily update the software .

Package manager already included in every linux distribution  
In ubuntu we have apt PM .

### **Managing software with PM (apt).**

Apt on terminal to see it .

```
ubuntu@ubuntu:~$ apt search openjdk*(to search the package)
```

```
ubuntu@ubuntu:~$ sudo apt install openjdk-21-jdk (to install)
```

```
ubuntu@ubuntu:~$ sudo apt remove default-jdk ( to delete a package)
```

### **Difference btw APT and APT\_GET :**

Apt is more user friendly it give more cleaner output , apt get don't have search command ,

### **Where does that installed packages come from > Repositories :**

```
ubuntu@ubuntu:~$ ls /etc/apt
apt.conf.d  keyrings    sources.list  sources.list.d
auth.conf.d preferences.d sources.list.curtin.old trusted.gpg.d
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ cat /etc/apt/source.list( this is the repository used by apt PM to
fetch the repositories and install the packages.
```

### **Alternative way to install software :**

**There are package , which are not present in these official repositories.**

**Availiable , but not the latest version**

**// because ubuntu ; verify the software packages, before addition to repo.**

**// verification process may take time**

### **1.Ubuntu software center :**



2. **Snap package manager** : release in 2014,  
Snap is a software packaging and deployment system for OS uses Linux kernel .

### **DIFFERENCE BTW SNAP AND APT ;**

Snap contained dependencies contained in the package.

Apt dependencies are share eg bin lib

Snap has automatic update , apt manual update

Support universal linux packages (package type .snap), (package type .deb) only for specific linux distribution .

Large installation file snap , smaller installation files apt.

Dependencies are shared among the application so it will not download it again (apt)

In snap shared dependencies are downloaded in each application .

### **3.add repository to official apt source :**

Sudo add-apt-repository

When add repository it will automatically be added into /etc/apt/source.list

Added repository are mostly PPA(personal package archive ) which are not official verified by ubuntu so careful add them .

### **LINUX DISTROS THERE ARE HUNDRED OF LINUX DISTRIBUTION BASED ON SAME SOURCE THEY ARE DIVIDED INTO TO CATEGORIES :**

Debian based : ubuntu , debian , mint which all use apt ,or apt get

Red hat based : RHEL , CentOS, fedora , which all use YUM .

Similar concept :

PM uses official repo , download packages, resolve dependencies etc.

Difference in rep : more or newer version of packages .

### **VI AND VIM TEXT EDITOR;**

Vim is improved versions of vim it enable use to edit the file in cli .

Based on linux distro

It supports multiple file format.

When no gui like in remote server when want to edit some files in server.

This is to insert or edit = I

:wq = write quit to save the edited file .

Esc to finish editing

:q! : without making the changes .

Delete a line : enter esc mode then type dd entire line deleted

Delete a multiple line : dd10 or 1 ,2 3, no . Represent the no of lines u want to delete

Type u to undo the changes , uu ()

A = switch to the end of the line and also insert mode on

O = to go to start of the line .

\$ = without going to insert mode go to end of line .

12G = jump to that line

/ search in vim file

:%s/this/that : replacing that word in file with new word .

ctrl+r : to redo the changes.

U : to undo the changes.

### **Users & Permissions :**

**3 type of user :**

**1 . Root or super user ;**

**That has unrestricted permission to all the system .**

**For admin task need to log in as root user or execute sudo command**

**2.user account : user we create in time or log in .**

**Eg : /home/vishal**

**3. Service user : relevant for linux server distros .**

**Each service of application will get its own user eg: mysql , apache etc.**

**Eg: mysql user will start mysql application .**

**Best practice for security.**

**Dont run services with root user. Because those service may take privileges of that user or some one can hack into our system.**

**Always have one root user per computer.**

**Can have multiple regular user or service user.**

**Why multiple standard user?**

**Windows is able to centrally manage the user . Where admin add user to the system.**

**Login to any hardware that is connected to that system .**

**But in linux don't have the central managing system.**

**One of the reason ,why windows is preferred in many companies and university.**

**How ever in linux user account are managed in that specific hardware.**

**Servers are managed by multiple peoples but why not use a shared user>>?**

**Why giving each team member there own user is important in servers.**

**>**

**?**

**They need non root user**

**Permission per team member .**

**Junior senior less more privileges etc.**

**Traceability - who did what on the system .**

**So admin creates users with permissions .**

**HOW TO MANAGE PERMISSION :**

**Now way User lvl : give permission to user directly**

**Best way : group users into linux groups**

**Give permission to group**

**The way to go if u are managing multiple users**

**Eg : devops group , admin group , developers group and users are added to the group.**

**Permission defines for the group .**

**WHERE ARE all the user system stored : >?**

**Cat /etc/passwd (etc is system configuration location)(/etc/passwd stores user account information )(every one can read it but only root user can change the file )**

*Example:* `nana:x:1000:1000:Nana,,,:/home/nana:/bin/bash`

`USERNAME : PASSWORD : UID : GID : GECOS : HOMEDIR : SHELL`

**Username : used when user log in**

**X : encrypted password is stored in /etc/shadow file**

**User id: each user has a unique id,uid 0 is reserved for root .**

**GID : is a primary group id stored in /etc/group file**

**User description**

**HomeDIR : absolute path of user home directory.**

**SHELL : this is user default shell which is bash .**

**~\$ sudo adduser <username> // create a new user .**

**Sudo Passwd <username> // to change the password of a user.**

**Su - <username> // to login to new user.**

**Su - // login to root user .**

**Sudo groupadd <groupname> // add new group**

**Cat /etc/group**

**Difference two add user , add group ,delgroup ,defuser VS user add , groupadd ,user del,groupdel. ?**

**First one is more friendly and interactive( use them when executing them manually),create home directory with automatic config automatically ,**

**Second one is low level utilities. , u need to provide infos your self(in scripts when using in automated way**

**How to change the group of a user :**

**Sudo usermod -g devops<groupname> vishal <username>**

**Sudo delgroup vishal**

**To multiple group one user :**

**Sudo usermod -G <groups names > vishal .**

**Groups // command which will list the groups the current login user belongs to**

**Group vishal // vishal user belongs to which groups.**

**Exit// this will log out form the current user and go back to the previous user .**

**Sudo useradd -G devops vishal // at same this create a group and user**

**Sudo gpasswd -d vishal devops // remove user vishal from devops group.**

Every thing in linux is a file .

User permission are related to reading , writing and executing file .

Ls to display all files in a directory

Ls -a to display hidden files also

Ls -l that shows the permission of file and folders in that directory .

ubuntu@ubuntu:~/Documents/newproject\$ ls -l

total 12

-rw-rw-r-- 1 ubuntu ubuntu 79 May 3 15:33 Readmu.md

```
-rw-rw-r-- 1 ubuntu ubuntu 89 May  3 15:59 config.yaml
-rw-rw-r-- 1 ubuntu ubuntu  6 May  3 11:45 testReadmu.md
```

First Ubuntu is owner of the folder or a file who created it  
Second ubuntu is the group owner of folder or a file.primary group of the user.

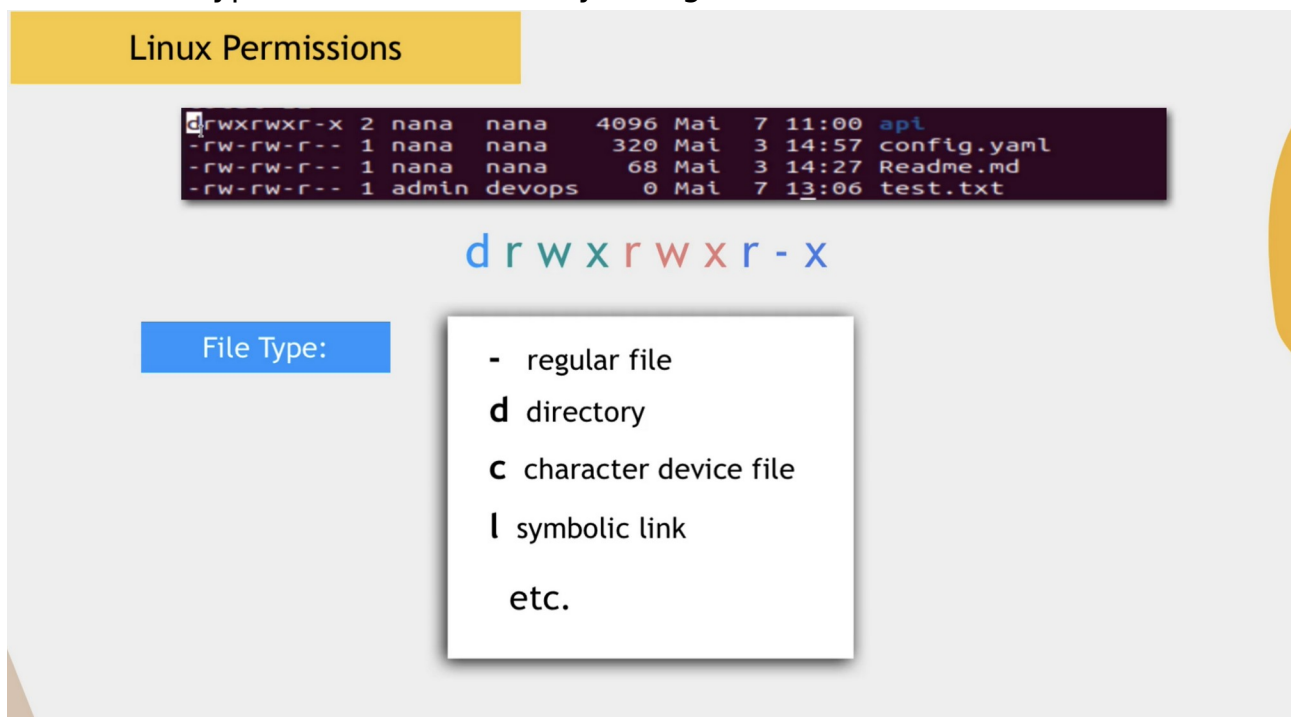
Changing the owner ship of the file to different user in home directory :

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chown admin:ubuntu test.txt
ubuntu@ubuntu:~/Documents/newproject$ ls -l
total 12
-rw-rw-r-- 1 ubuntu ubuntu 79 May  3 15:33 Readmu.md
-rw-rw-r-- 1 ubuntu ubuntu 89 May  3 15:59 config.yaml
-rw-rw-r-- 1 admin  ubuntu  0 May  3 16:04 test.txt
-rw-rw-r-- 1 ubuntu ubuntu  6 May  3 11:45 testReadmu.md
```

### **Most of the root group and root user own these system files.**

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l /etc/
total 1140
drwxr-xr-x  4 root root  4096 Oct 21  2023 ModemManager
drwxr-xr-x  7 root root  4096 Oct 21  2023 NetworkManager
drwxr-xr-x  2 root root  4096 Aug 10  2023 PackageKit
```

D is the file type . D means directory , - regular file



The infographic titled "Linux Permissions" illustrates the structure of file permissions. At the top, a terminal window shows the output of the 'ls -l' command for a directory named 'api', listing files like 'config.yaml', 'Readme.md', and 'test.txt' with their respective permissions, owners, groups, sizes, dates, and names. Below this, the permissions 'drwxrwxr-x' are broken down into their individual components: 'd' (directory), 'r' (read), 'w' (write), 'x' (execute), and so on. A 'File Type:' box highlights the first character, and a list explains the meaning of each character: '-' for regular file, 'd' for directory, 'c' for character device file, 'l' for symbolic link, and 'etc.' for other special files.

**Linux Permissions**

```
drwxrwxr-x 2 nana nana 4096 Mai  7 11:00 api
-rw-rw-r-- 1 nana nana 320 Mai  3 14:57 config.yaml
-rw-rw-r-- 1 nana nana  68 Mai  3 14:27 Readme.md
-rw-rw-r-- 1 admin devops  0 Mai  7 13:06 test.txt
```

**d r w x r w x r - x**

**File Type:**

- regular file
- d** directory
- c** character device file
- l** symbolic link
- etc.

## Linux Permissions

```
drwxrwxr-x 2 nana nana 4096 Mai 7 11:00 apt
-rw-rw-r-- 1 nana nana 320 Mai 3 14:57 config.yaml
-rw-rw-r-- 1 nana nana 68 Mai 3 14:27 Readme.md
-rw-rw-r-- 1 admin devops 0 Mai 7 13:06 test.txt
```

d r w x r w x r - x

Owner

Group

**r** Read

**w** Write

**x** Execute

- No permission

► Dash instead of the letter means, this permission is absent

## MODIFY THE PERMISSIONS :

### Remove permissions

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l
```

```
total 12
```

```
-rw-rw-r-- 1 ubuntu ubuntu 79 May 3 15:33 Readmu.md
```

```
-rw-rw-r-- 1 ubuntu ubuntu 89 May 3 15:59 config.yaml
```

```
-rw-rw-r-- 1 admin ubuntu 0 May 3 16:04 test.txt
```

```
-rw-rw-r-- 1 ubuntu ubuntu 6 May 3 11:45 testReadmu.md
```

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chmod -r Readmu.md
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l
```

```
total 12
```

```
--w--w---- 1 ubuntu ubuntu 79 May 3 15:33 Readmu.md
```

```
-rw-rw-r-- 1 ubuntu ubuntu 89 May 3 15:59 config.yaml
```

```
-rw-rw-r-- 1 admin ubuntu 0 May 3 16:04 test.txt
```

```
-rw-rw-r-- 1 ubuntu ubuntu 6 May 3 11:45 testReadmu.md
```

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chmod g-w test.txt
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l
```

```
total 12
```

```
--w--w---- 1 ubuntu ubuntu 79 May 3 15:33 Readmu.md
```

```
-rw-rw-r-- 1 ubuntu ubuntu 89 May 3 15:59 config.yaml
```

```
-rw-r--r-- 1 admin ubuntu 0 May 3 16:04 test.txt
```

```
-rw-rw-r-- 1 ubuntu ubuntu 6 May 3 11:45 testReadmu.md
```

```
ubuntu@ubuntu:~/Documents/newproject$
```

### Adding the permissions

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chmod g+x test.txt
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l
```

total 12

```
--w--w---- 1 ubuntu ubuntu 79 May  3 15:33 Readmu.md
-rw-rw-r-- 1 ubuntu ubuntu 89 May  3 15:59 config.yaml
-rw-r-xr-- 1 admin  ubuntu  0 May  3 16:04 test.txt
-rw-rw-r-- 1 ubuntu ubuntu  6 May  3 11:45 testReadmu.md
ubuntu@ubuntu:~/Documents/newproject$
```

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chmod u+x script.sh
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -ls
```

total 12

```
4 --w--w---- 1 ubuntu ubuntu 79 May  3 15:33 Readmu.md
4 -rw-rw-r-- 1 ubuntu ubuntu 89 May  3 15:59 config.yaml
0 -rwxr-xr-- 1 admin  ubuntu  0 May  3 16:04 script.sh
4 -rw-rw-r-- 1 ubuntu ubuntu  6 May  3 11:45 testReadmu.md
ubuntu@ubuntu:~/Documents/newproject$
```

---

Owner	Group	Other			
u	g	o	+	add	r Read
			-	remove	W Write
					X Execute

### Apply changes to all the users :

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chmod a-r testReadmu.md
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l
```

total 12

```
--w--w---- 1 ubuntu ubuntu 79 May  3 15:33 Readmu.md
-rw-rw-r-- 1 ubuntu ubuntu 89 May  3 15:59 config.yaml
-rwxr-xr-- 1 admin  ubuntu  0 May  3 16:04 script.sh
--w--w---- 1 ubuntu ubuntu  6 May  3 11:45 testReadmu.md
ubuntu@ubuntu:~/Documents/newproject$
```

### To modify whole at once :

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chmod g=rwx script.sh
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l
```

total 12

```
--w--w---- 1 ubuntu ubuntu 79 May  3 15:33 Readmu.md
-rw-rw-r-- 1 ubuntu ubuntu 89 May  3 15:59 config.yaml
-rwxrwxr-- 1 admin  ubuntu  0 May  3 16:04 script.sh
--w--w---- 1 ubuntu ubuntu  6 May  3 11:45 testReadmu.md
```

# Absolute(Numeric) Mode

Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--X
2	Write	-W-
3	Execute + Write	-WX
4	Read	r--
5	Read + Execute	r-X
6	Read +Write	rw-
7	Read + Write + Execute	rwX

```
ubuntu@ubuntu:~/Documents/newproject$ sudo chmod g=rwx script.sh
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -l
```

```
total 12
```

```
--w--w---- 1 ubuntu ubuntu 79 May  3 15:33 Readmu.md
-rw-rw-r-- 1 ubuntu ubuntu 89 May  3 15:59 config.yaml
-rwxrwxr-- 1 admin  ubuntu  0 May  3 16:04 script.sh
--w--w---- 1 ubuntu ubuntu  6 May  3 11:45 testReadmu.md
```

```
ubuntu@ubuntu:~/Documents/newproject$ ls -la
```

```
total 20
```

```
drwxrwxr-x 2 ubuntu ubuntu 4096 May  3 16:44 .
drwxr-xr-x 4 ubuntu ubuntu 4096 May  3 11:44 ..
--w--w---- 1 ubuntu ubuntu  79 May  3 15:33 Readmu.md
-rw-rw-r-- 1 ubuntu ubuntu  89 May  3 15:59 config.yaml
-rwxrwxrwx 1 admin  ubuntu   0 May  3 16:04 script.sh
--w--w---- 1 ubuntu ubuntu   6 May  3 11:45 testReadmu.md
```

**To see the hidden files also ls -la**

## PIPES AND REDIRECTS :

**The output of one command can become the input of other command .**

If u want to see the output of the command cat if it is a big log file u can use pipe



U can take the output to that and take that as input for other program and get output

| : pipes the output of the previous command as an input to the next command.

Less : displays the content of a file or command output one page at a time . It allows u to navigate forward or backward through the file .

Mostly used to open large file , faster load time.

```
ubuntu@ubuntu:~$ ls /usr/bin | less
```

And just press space .

This will give us page by page via of all this directory context.

Press b to go to back page

Q to quit

**ubuntu@ubuntu:~\$ history | grep sudo // include “this” if more then one word to search .**

```
17 sudo apt update
18 sudo apt install spice-vdagent spice -webdavd
19 sudo apt install spice-vdagent spice-webdavd
20 sudo reboot
102 sudo adduser admin
111 sudo addgroup devops
118 sudo apt search openjdk
120 sudo apt update
122 sudo apt install default-jdk
126 sudo apt install openjdk-21-jdk
128 sudo apt remove default-jdk
130 sudo apt update
147 sudo chmod 777 Readmu.md
161 sudo apt update
162 sudo apt install less
167 history | grep sudo
```

Or history | grep sudo | less

```
ubuntu@ubuntu:~$ ls /usr/bin | grep java
```

java

javac

javadoc

javap

## REDIRECTION IN LINUX:

```
ubuntu@ubuntu:~$ history | grep sudo > sudo.txt
```

```
ubuntu@ubuntu:~$ ls
```

```
Desktop Downloads Pictures Templates snap
```

```
Documents Music Public Videos sudo.txt
```

```
ubuntu@ubuntu:~$
```

```
ubuntu@ubuntu:~$ cat sudo.txt > sudo-rm-commands.txt
```

```
ubuntu@ubuntu:~$ ls
```

```
Desktop Downloads Pictures Templates snap
```

```
sudo.txt
```

```
Documents Music Public Videos sudo-rm-commands.txt
```

```
ubuntu@ubuntu:~$
```

**(Copying the content of this file to other file .**

>> = this will make sure there is no overriding

```
255 history | grep sudo > sudo.txt
```

```
6 rm .bash_history
```

```
32 rm main.py
```

```
36 cd rm -r python-project
```

```
37 cd rm -r python-project
```

```
38 rm -r python-project
```

```
67 rm Readme.md
```

```
257 cat sudo.txt > sudo-rm-commands.txt
```

```
260 cat sudo-rm-commands.txt
```

```
261 history | grep rm >> sudo-rm-commands.txt
```

```
ubuntu@ubuntu:~$ c
```

**this is one line commands**

**ubuntu@ubuntu:~\$ clear ; sleep 3 ; echo "this is one line commands"**

## **INTRODUCTION TO SHELL SCRIPTING :**

Useradd vis , groupadd devops , mkdir project , touch file.txt , chmod 750 /path , sudo apt docker , docker run .

If I need to run these same commands on other server do I need to again write all these commands NO.

Eg : we have 10 servers where we need to do exactly same things .

Shell scripting : avoid repetitive work, keep history of configuration ,share the instruction , logic and build operation .

If there are 100 of users to be added etc bulk task s

-> write command on file

-> Execute the file .

File is movable .

Such file is called shell script

Shell script have .sh extension .

### **Why we call it shell script :**

In unix like system : like linux and macOS.

**Shell** =a program that interpret and execute the various commands that we type on terminal .

And Translate the commands that OS kernel can understand .

### **Different shell implementation .**

**1.sh(Bourne shell) : /bin/sh : it was used to be the default shell of many =unix type system .**

**2.Bash (Bourne again shell ) : /bin/bash : improved version of sh. Bash is the default shell program for most unix like system .**

**Shell and bash term often used interchangeably .**

**Bash is a shell program .**

**Bash is a programming language . With which we can write shell script.**

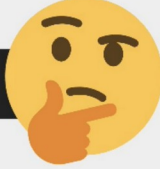
**Write shell script using bash syntax .**

**As all shell script file have same extension how does the OS knows which shall to execute ?**

**We may we write shell script in sh , bash or zsh .**

## Shebang

► All shell script files have the same .sh file extension

How does OS know which shell to use? 

SH

BASH

ZSH

We need to tell the OS!

`#!/bin/sh`

`#!/bin/bash`

`#!/bin/zsh`

**#=sharp**

**!=bang**

**Therefore shebang**

**Best to use bash ,one advantage of sh is used in every unix system .**

By default a file create is without execute command .

First `sudo chmod u+x setup.sh`

**Then `./setup.sh` ( this will execute the setup script ) .**

### **Concept of Functions in bash scripting :**

Function name(){ list of commands ; }

This code block can be referenced anywhere in the script multiple times .

Functions is a way to clean up your script to group things together .

And give them name who that your script is easy to read and understandable by other users.

```
#!/bin/bash
```

```
echo "setup and conf. server"
```

```
function score_sum {
```

```
    sum = 0 ;
```

```
while true
do
    read -p "enter a score " score

    if ["$score" == "q"]
    then
        break
    fi
    sum = $((sum + $score))
    echo "total score: $sum"

done
}
```

score\_sum

~

~

-- INSERT --

**With parameter:**

**ubuntu@ubuntu:~\$ cat setup.sh**

**#!/bin/bash**

**echo "setup and conf. server"**

```
function score_sum {
    sum=0
    while true
    do
        read -p "enter a score: " score

        if [ "$score" == "q" ]
        then
            break
        fi
        sum=$((sum + score))
        echo "total score: $sum"
    done
}
```

```
function create_file() {
```

```
file_name=$1
touch "$file_name"
echo "file $file_name created"
}
```

```
create_file test.txt
create_file myfile.yaml
create_file myscript.sh
```

```
ubuntu@ubuntu:~$ ./setup.sh
setup and conf. server
file test.txt created
file myfile.yaml created
file myscript.sh created
```

### **Adding multiple para meter :**

```
function create_file() {
    file_name=$1
    is_shell_script=$2
    touch "$file_name"
    echo "file $file_name created"

    if[ "$is_shell_script" = true ]
    then
        chmod u+x $file_name
        echo "added execute permission"
    fi
}
```

```
create_file test.txt
create_file myfile.yaml
create_file myscript.sh true
```

```
ubuntu@ubuntu:~$ ./setup.sh
setup and conf. server
./setup.sh: line 27: syntax error near unexpected token `then'
./setup.sh: line 27: ` then '
ubuntu@ubuntu:~$ vim setup.sh
ubuntu@ubuntu:~$ ./setup.sh
setup and conf. server
file test.txt created
```

file myfile.yaml created  
file myscript.sh created  
added execute permission

### **Returning value from the function :**

```
function sum(){  
total=$(( $1+$2 ))  
return $total  
}  
sum 2 10  
result=$?  
echo "sum of 2 and 10 is $result"
```

**// we can have scripts for backup**  
**// scripts that configure the servers**  
**// scripts that monitor servers.**  
**//automate your work**  
**//portable ,shared files**

### **ENVIRONMENT VARIABLES :**

Multiple user environment is very common on many OS.  
Each user has its own environment  
Each user can config its own environment  
Eg : different font , wallpaper , different shell(bash , sh , zsh) ,editor  
(vim,nano) , chrome, Firefox.  
These OS configuration should be isolated from other user environment .

### **Where does OS store these configuration :**

In Environment variables , which are pices of information defined by key values pair.

Variable store information .

Eg: SHELL =/bin/bash(default shell program location of the user)(user can config it to zsh)

HOME = /home/vis(current user home directory )

USER = vis ( has value of currently logged in user)

Environment variables are available for whole environment .  
Keys of all the environment variable are defines in all capital letters.

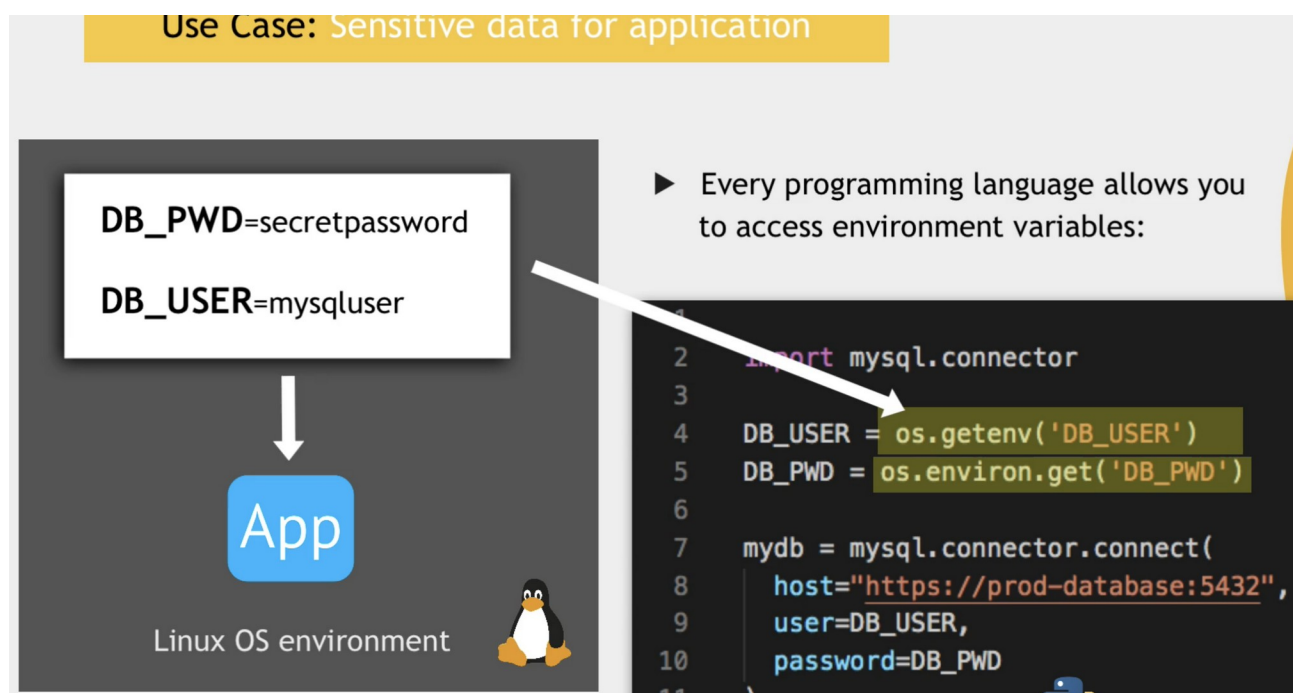
**Printenv : this will show all the EV set for current user .**

**ubuntu@ubuntu:~\$ printenv | less**

**Printenv USER : this will be specific to that EV**  
**Or Printenv | grep USER :**

User cases of EV :

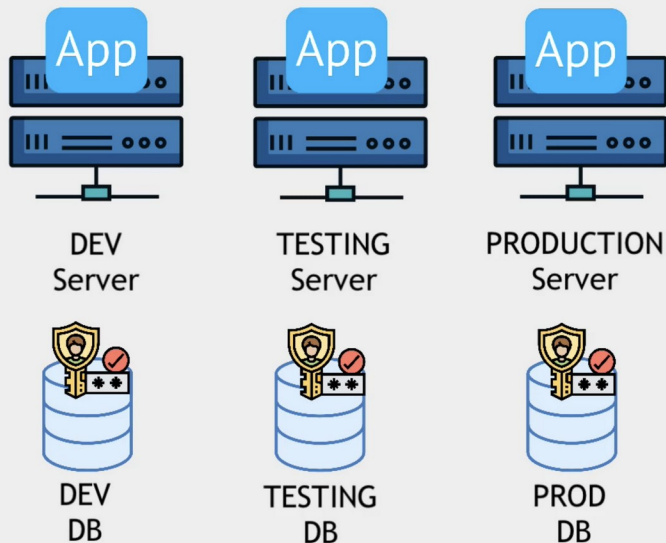
We can also create our own EV .



**Making application more flexible .**



### Use Case: Make application more flexibel

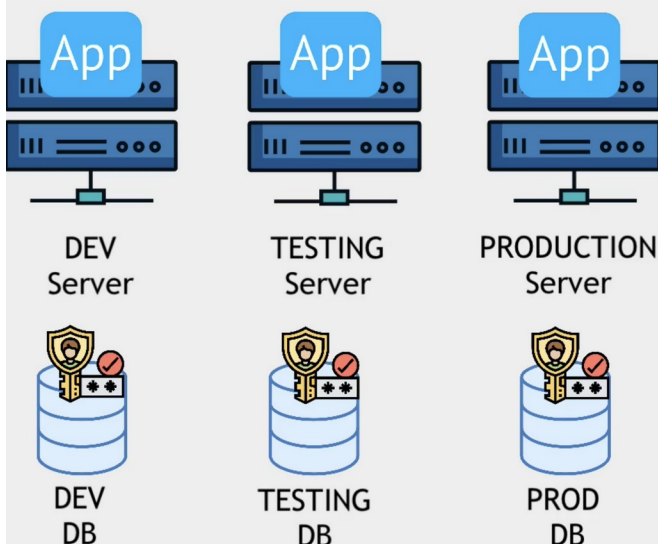


```
import mysql.connector

mydb = mysql.connector.connect(
    host="https://mydb-database:5432",
    user="mysqluser@me",
    password="secretpassword"
)
```

- Do not hardcode these connection values (Db URL, Db User, Db Pwd)

### Use Case: Make application more flexibel



```
DB_URL = os.environ.get('DB_URL')
DB_USER = os.getenv('DB_USER')
DB_PWD = os.environ.get('DB_PWD')
```

```
mydb = mysql.connector.connect(
    host=DB_URL,
    user=DB_USER,
    password=DB_PWD
)
```

 Python example

**Creating environment variable :**

**Export DB\_USERNAME = vishal**

**Remove delete EV : unset DB\_USERNAME**

**The export command will set EV for that terminal session only new terminal that EV is gone which we created .**

**For permanent save EV :**

**Vim .bashrc : store all the EV store in that environment**

**Here add : Export DB\_USERNAME = vishal**

**How reload system to notice the changes : source .bashrc**

**IF I WANT TO CHANGE SYSTEM WISE ENVIRONMENT VARIABLE :**

## **What is a PATH variable : (available to all the users in the system )**

Vim /etc/environment we will see path (which is globally set ENV): which is list of directories which is separated by : with binary files in them

Tell the shell which directories to search for the executable in response to our executed command.

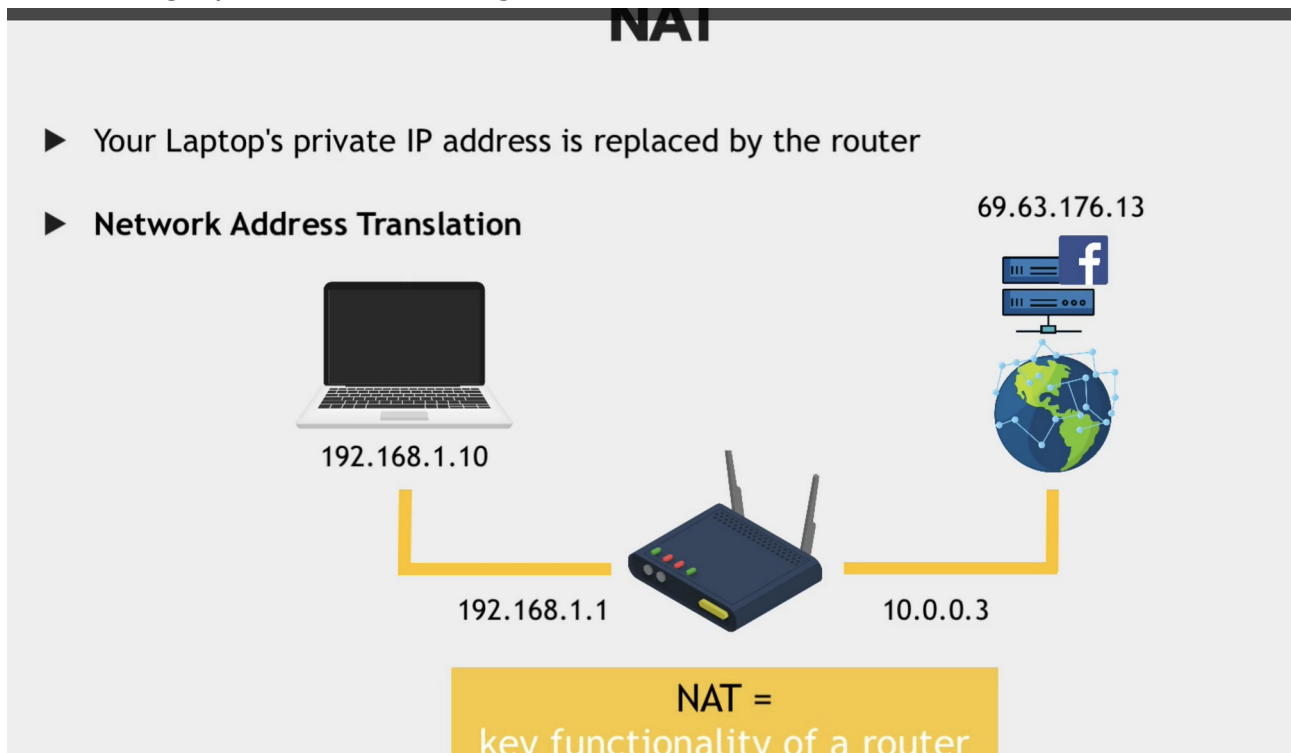
## **NETWORKING :**

Router sits btw the local and wAn will allow to access internet .

Phone send request to router and router send req over Facebook server over a Internet.

Gate way is the ip address of a router .

Subnetting : process of dividing a network into two or more network .



**Nat : security and protection of devices with in the lan**

**Reuse ip addresses**

**Two huge companies can have the same Ip address range with in there LAN ;**

**They will not know about each other so no conflict .**

## **FireWall:**

A system that prevent unauthorised access from entering private network .

What is a port :

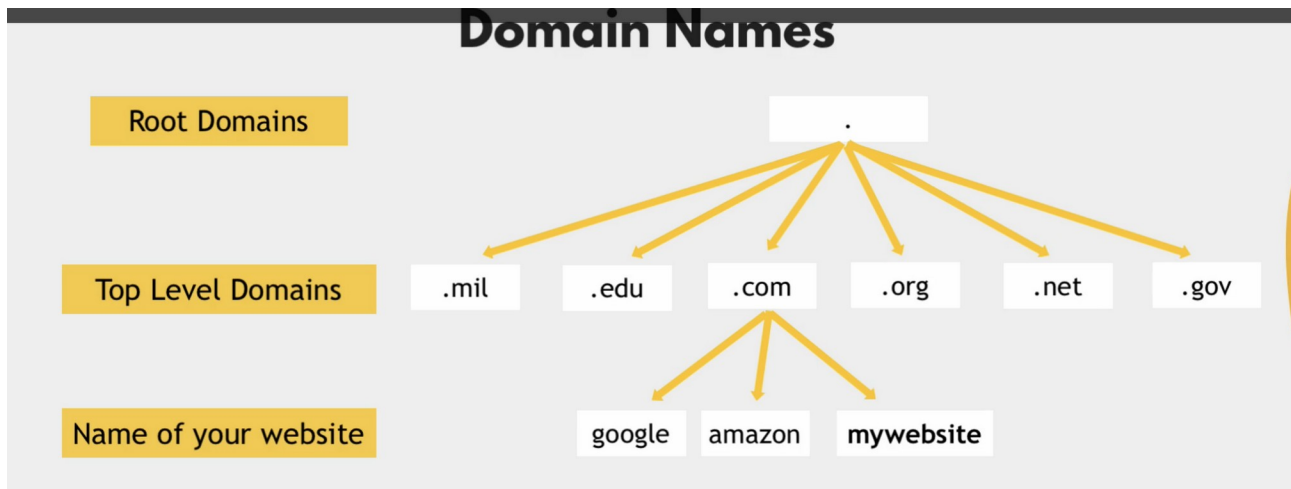
Every device has a set of ports : standards port for many application

Eg: port 80 : web servers .

Mysql port is 3306

Each port is unique on a device .

For every application u need a port.



ICANN manages all these .

Networking commands : ifconfig , netstat (active connection to your machine ).

Ps aux (current running process and programs which port they are running and how much resources using )

nslookup(check the ip addresss of any domain name),ping [google.com](http://google.com) (**weather that address is accessible or not** )\

## SSH: important concept in devops .

Secure shell :

Eg : copy shell script to another server , how to u access that remote server .

SSH is a network protocol that gives users a secure way to access computer over a network

Ensure encrypted data communication .

We need to authenticate our self with the remote servers .

2 ways to authenticate to remote server over ssh :

A : user name and password create on that remote server .

Admin creates a user on remote server .

User can then connect with username and password .

Vishal --> ssh -->remote server (admin create vishal user).

2.SSH key pair : more secure than first one .:

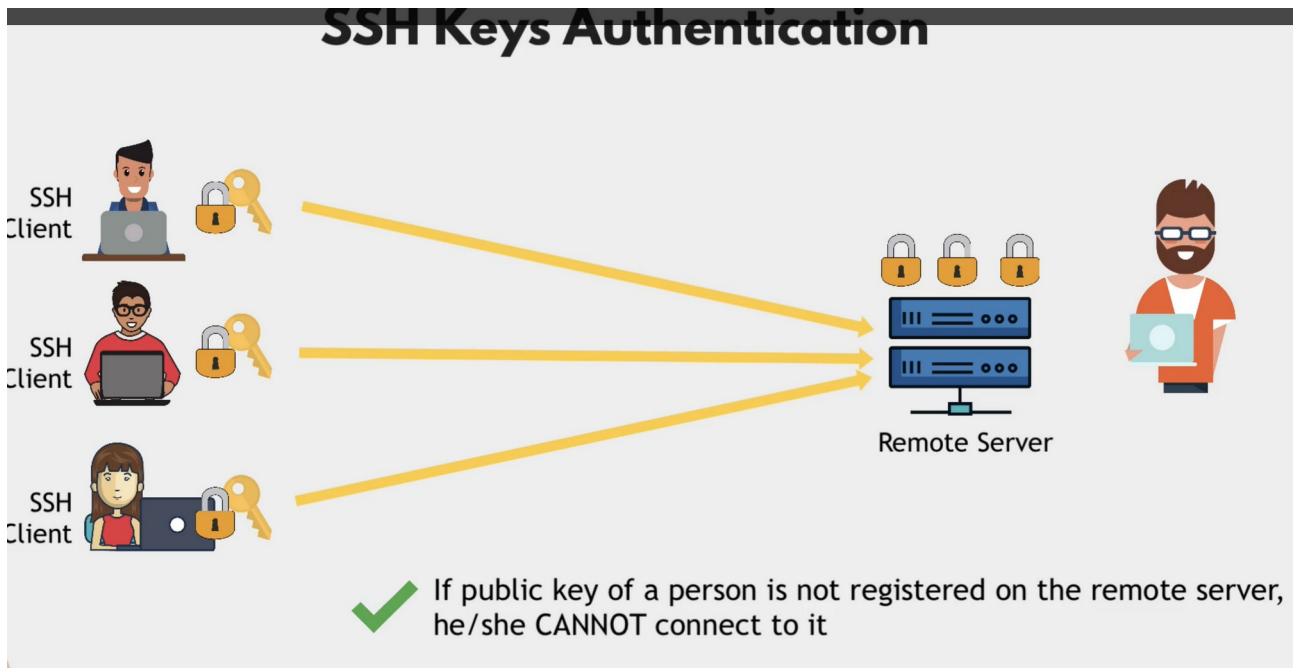
Client creates a SSH key pair which has a public and a private key

Keys are encrypted values of random hashes .

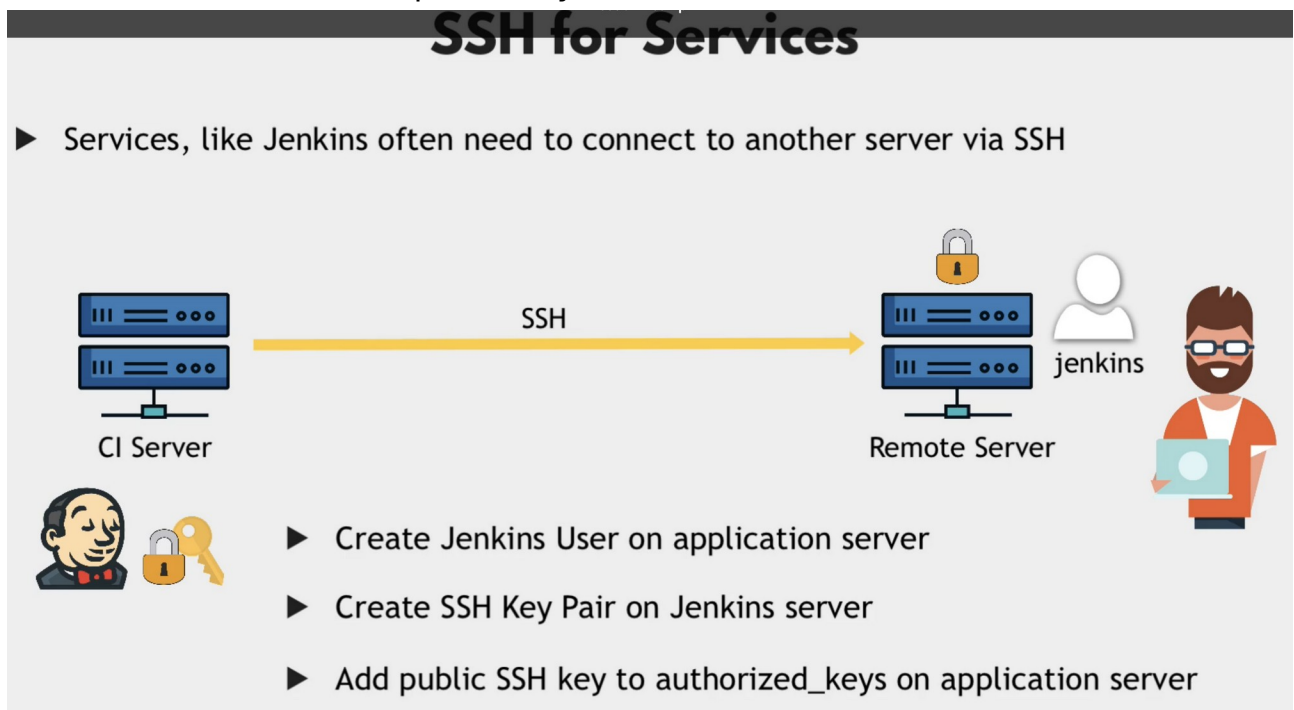
Private key : is a secret key that only client machine who created it has access.

Public key : can be shared with other eg with remote server . Saying client machine for that public key can safely connect .

Client can unlock the public key with its private key .



Admin will add the client public key on remote servers first.



//

Communication must be explicitly allowed by firewall rule by default it is blocked .

SSH authentication is a step after connection

Ssh service runs by default on machine

By default ,ssh services listen to port 22 .

In firewall rule we can specify the source : who can access server on this port. ?  
(they cant access other services in the machine )

SSH is more powerful if u are connect with ssh have access to the whole system . Therefore it need to be restricted to specific ip addresses.

## SSH IN ACTION :

Create remote server on cloud platform : digital ocean

Generate ssh key pair in your system .

Copy a bash script in the remote server .

Execute script on remote server .

How to connect via ssh with remote user :

1.Ssh <username by default root >@<ip address of the remote server>

2. Provide a password . Password that we configured when creating a remote server.

3. Now u can start using the terminal of remote server : u will be in the home directory of the root user .

4.we have connected to the remote server via a username and password not SSH key till now .

B: connection via SSH key pair :

1. Ls .ssh/ : check if there is a ssh folder in user home directory when we Confirm the prompt to host authenticity . :: showing - > knowhost

2. In user terminal : Ssh-keygen -t rsa(rsa is a cryptographic algo to encrypt those keys),

```
nana@nana-VirtualBox:~$ pwd
/home/nana
nana@nana-VirtualBox:~$ ls .ssh/
known_hosts
nana@nana-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nana/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nana/.ssh/id_rsa
Your public key has been saved in /home/nana/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:QbMeoF/mvw3HE2kQkobyEbmZ8Ty8QZBoFShOBPKVFhw nana@nana-VirtualBox
The key's randomart image is:
+---[RSA 3072]---+
|oo..EB*B+..|
|..o.O.*o++ .|
| o.+..o %* .|
| . .==*O . .|
| . S+ +|
| . . o .|
| o +|
| = .|
| . .|
+-----[SHA256]-----+
```

3. Ls .ssh/ : id\_rsa (private key ) , id\_rsa.pub(public key )



4. Private stays in the computer we don't share it with any one and public key can be shared with the remote server we want to connect to .
5. Now how to put that public key to the remote server so that we can access that via ssh key .

```
nana@nana-VirtualBox:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDvHAPX7fhJn5eonF/TXwkOsVtx8Q0AuvU9aF6Yf3r
Tcxu8i85yTzp+pQ2m8ZjG0eMjsSN5BcpCEDWUY1FSCTrsMmlc+GYL07LoWmdPgLPPrQfr3d5jozH49DB
zPw2EqHkGwRYKp8zxpDDgbc5L0XyJRjkkQxgV56SyVIGrNpsTq2FKmSGz9H50gzyJFgyAvV1onNZh35
OXRvfscsRl1JDAL/XGILx3VtqKUBPwNIAZce17Lqa/8kzxtSfzZINfilI92YSdKcp9ep3Jy1R0v2Zws
lm2/D90Q+EU6f6CZtpke/+Gd07gBXwRYOLPOUzGgugAWU9EFhfXBeQ1lxhnYqxroBWaWvdhZa0eVST
eZQnCifG6Xc5D0HGse05TnnwWqQ8cksB//91q5oCwLYdD5gKEwCBzYxt50f33P7RbFp6eSk35H5yxy
4yYNAAsP0qQfgItUR98Y8NLJeyuaVfB8/OcNbNqxy/rGEUuPAHsbkIBXaFPCmorIchZU2j55rQfdE=
nana@nana-VirtualBox:~$
```

```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# ls .ssh
authorized_keys
root@ubuntu-s-1vcpu-1gb-fra1-01:~# cat .ssh/authorized_keys
root@ubuntu-s-1vcpu-1gb-fra1-01:~# vim
root@ubuntu-s-1vcpu-1gb-fra1-01:~# vim .ssh/authorized_keys
root@ubuntu-s-1vcpu-1gb-fra1-01:~# exit
logout
Connection to 159.89.14.94 closed.
nana@nana-VirtualBox:~$
```

We can add multiple client public key on authorised keys file  
NOW FINNALLY CONNECT TO REMOTE SERVER VIA SSH KEY;

```
nana@nana-VirtualBox:~$ ssh root@159.89.14.94
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

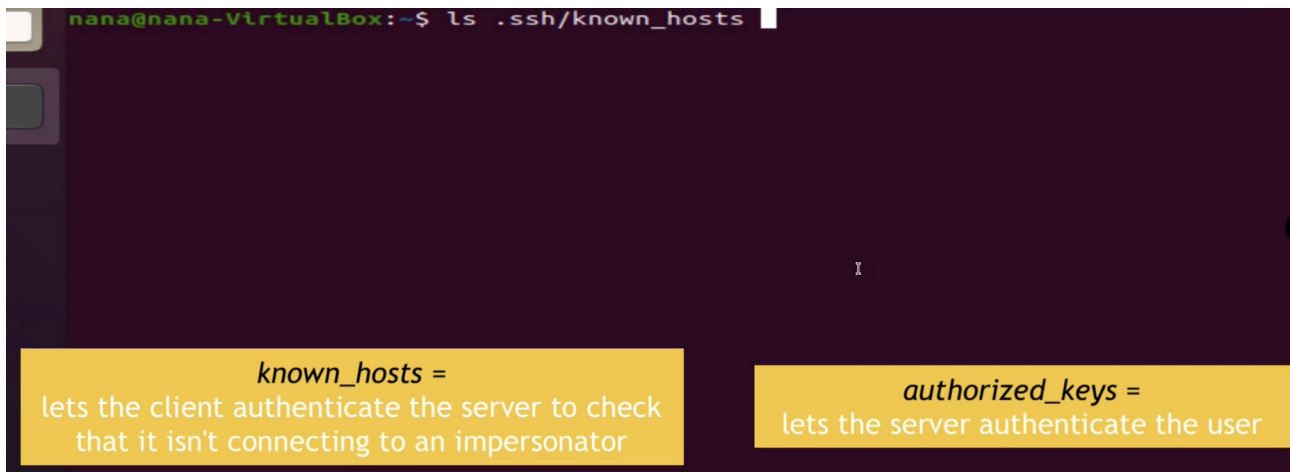
System information as of Sat May 15 10:59:38 UTC 2021

System load:  0.08               Users logged in:  0
Usage of /:   5.1% of 24.06GB    IPv4 address for eth0: 159.89.14.94
Memory usage: 18%               IPv4 address for eth0: 10.19.0.5
Swap usage:   0%                IPv4 address for eth1: 10.114.0.2
Processes:   100

1 update can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Sat May 15 10:50:51 2021 from 213.185.161.134
root@ubuntu-s-1vcpu-1gb-fra1-01:~#
```



Ssh -I .ssh/id\_rsa [root@159.99.44.3](#) (u want to provide which private key is used)  
By default as we have only one private key we can use this : ssh  
[root@156.42.21.21](#)

:: creating a small script file in our machine and copy in the remote and  
execute it there . ??

Vim test.sh : #!/bin/bash : echo "I am executed on the remote server" :

Copy this test.sh to the remote server : scp(secure copy) test.sh(source)

[root@198.22.3.3](#)(target ie.remote server/root is the user name of the remote  
server ) .

Continue => root@12.3..4.5:/root (mention path where to copy )

A terminal window with a dark purple background. The prompt is 'nana@nana-VirtualBox:~\$' and the command is 'vim test.sh'. The next line shows the prompt 'nana@nana-VirtualBox:~\$' and the command 'scp test.sh root@ test.sh'.

**This command takes private key as parameter :**

**Eg: scp -I .ssh/id\_rsa test.sh root@123.4.5.6:/root**

**Execut in remote server : ./test.sh**