

LINUX FOR DEVOPS

Day 1

What is Operating System?

A operating system is system software that manages computer hardware and software resources and provides common services for computer programs. Operating system is a bridge between hardware of the computer and the users.

What is Linux?

Linux is an open source operating system that is made up of the kernel, the base component of the OS and the tools, apps, and the services bundled along with it.

- Linux is free and do not have to worry about paying anything at all.
- Allowing you to make changes to its code and adding new functionality to be used by other users.
- It is one of the best secure and stable OS, and it is very low chance of getting attacked by any hackers or virus
- Linux does not require much memory and space effectively working. It is very low system requirements.
- We can run along with linux and windows in one system and within windows using virtual machines.

Why choose Linux?

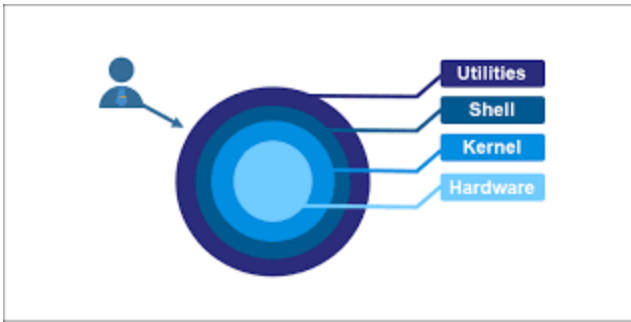
- Linux is widely used for troubleshooting purpose for other computer.
- We want to build or host a website.
- It is one of the secure and robust operating system and most stable one.

Linux for DevOps

Linux is important for DevOps because it provides an environment needed to manage infrastructure, automate tasks and troubleshoot issues. Linux is backbone of the devops.

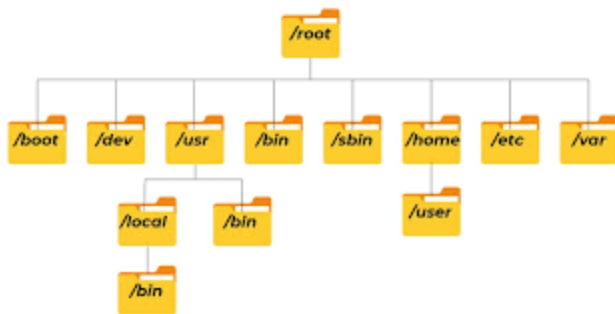
- A core goal of DevOps is fast software delivery and building on existing infrastructure, linux is big part of that.
- Linux and any other OS is an essential component of any IT operation, to know how to configure Linux for DevOps is essential to a continual and speedy software delivery process.
- It enables the creation of design and security applications to a specific development environment or set of development objectives. Compared to Windows, there is a lot more control over how the operating system operates.
- Most of the DevOps tools are Linux based tools and supports linux. DevOps engineer may easily and internally do all testing using a Linux-based operating system
- Scalability enables fast delivery without forcing developers to compromise the quality of their codebase.

Linux Architecture



- **Hardware:** It consists of all peripheral devices such as RAM, CPU, HDD etc...
- **Kernal:** It is the central component of an operating system. It means harware resuorces, system calls and provide essential services to applications.
- **Shell:** Shell takes commands from the users and execute kernal functions. It is a commandline-interpreter that lets linux users control their operating systems with CLI (commandline interface).
- **Utilities:** Provides a set of functions and routines that application use to interact with the kernal and perform various tasks.

Linux Filesystem Hierarchy Structure



- 1) **/root** : The top most directory of the filesystem. All files and directories start from the root directory.
- 2) **/bin(Binaries:)** It contains user commands binaries or executables like ls, mv, cp, rm etc. These commands are available for all users.
- 3) **/sbin(System Binaries):** It contains system commands binaries used for administrative tasks like ifconfig, fdisk, htop, dh - f, etc.
- 4) **/boot:** This directory contains booting files for the system and bootloader configuration files.
- 5) **/dev(Devices):** Contains device files representing hardware devices such as disks(/dev/sda), USB devices, terminals(/dev/tty)
- 6) **/etc(Configuration files):** This directory contains system-wide configuration files and shell scripts for system initialization. Filesystem tables(/etc/fstab), network hosts(/etc/hosts), user information(/etc/hostname).
- 7) **/home:** This is the home directory for all user. Every user personal files and documents stored here.
- 8) **/lib(Libraries):** It contains shared library files for required by the binaries in /bin and /sbin. Libraries are essential for the

basic functionality of the system.

9) `/media`: A mount point for removable media like USB drives, CDs and DVD. When we inserted these media then automatically mounted under `/media`.

10) `/mnt(Mount)`: It is a temporary mount filesystem. Users use it to mount filesystem temporarily.

11) `/opt(Optional)`: It contains third party application installations.

12) `/var(Variable files)`: It contains variable data files such as logs(`/var/log`), databases(`/var/lib`) and cached files(`/var/cache`). This is a read and writable directory used for files change frequently.

13) `/run`: Stores runtime data for processes since the system was booted. This data is typically volatile and not persistent across reboots.

14) `/srv(Service)`: Contains data for services provided by the system, such as web servers, FTP servers, or databases (e.g., `/srv/www` for web server files).

15) `/sys(System)`: A virtual filesystem providing information about devices, kernel modules, and other kernel-related data structures.

16) `/tmp(Temporary Files)`: A directory for temporary files created by applications and the system itself. Files in `/tmp` are typically deleted on reboot or after a certain period.

17) `/usr(User Binaries & Read-Only Data)`: Contains user utilities and applications. It is divided into subdirectories such as `/usr/bin` (non-essential command binaries), `/usr/sbin` (non-essential system binaries), `/usr/lib` (libraries), and `/usr/share` (shared data like icons, fonts, and documentation).

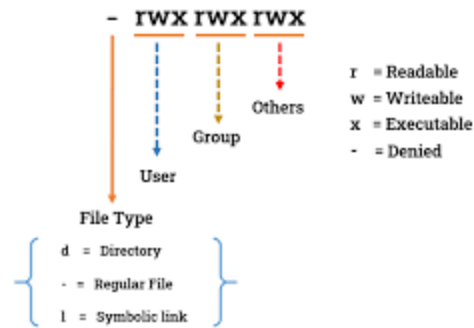
18) `/proc(proces)`: This directory contains information about the system process. (`/proc/cpuinfo`), (`/proc/meminfo`).

Basic Linux Commands

- `ls` : This command is used to list the contents of the directory
- `cd` : This command used to change the directories
- `pwd` : This command will show present working directory
- `cp` : This command is used to copy the files
- `mv` : This command is used to rename the files and directories and move files or directories one directory to another
- `rm` : This command is used to remove file or directories
- `mkdir` : This command is used to create directories

Linux File Permissions

- **File Access Modes** :- The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the read, write, and execute permissions, which have been explained below
- **Read** :- Grants the capability to read, i.e., view the contents of the file.
- **Write** :- Grants the capability to modify, or remove the content of the file.
- **Execute** :- User with execute permissions can run a file as a program.
- **Directory Access Modes** :- Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned
- **Read** :- Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.
- **Write** :- Access means that the user can add or delete files from the directory.
- **Execute** :- Executing a directory doesn't really make sense, so think of this as a traverse permission.



NUMBER	PERMISSION REPRESENTATION	REFERENCE
0	No permission	--
1	Execution permission	--X
2	Write permission	-W-
3	Execute & Write permission	-WX
4	Read permission	r--
5	Read & Execute permission	r-X
6	Read & Execute permission	rw-
7	Read, Write & Execute permission	rwX

Changing File permissions and Ownership and groups

Each file and directory in Linux has three types of permissions (read, write, execute) for three types of users (owner, group, others).

- **Changing Ownership:**
- To change the owner and group of a file or directory:

```
$ sudo chown saikrishna:voziq task.txt
```

```
chmod +rwx filename: Adds permissions
chmod -rwx directoryname: Removes permissions
chmod +x filename: Allows executable permissions
chmod -wx filename: Takes out write and executable permissions
chmod +rwx filename: Allows Read and write execute permissions
chmod g+w filename: Changes directory permissions for group owners
```

User and Group management

In Linux, user and group management is essential for controlling access to files, directories, and system resources. Overview of the key concepts and commands related to user and group management. User and group management in Linux allows administrators to control access to system resources efficiently. By assigning users to groups, administrators can manage permissions on files, directories, and processes, ensuring that the system is secure and users have appropriate levels of

access.

1. Users in Linux

Users types:

- **Root User:** This user is superuser. This user have complete control over the system. This user can perform any action, including administrative tasks.
- **Regular User:** A non root users or non-priviledged user. This user have limited permissions. This user can have only permissions for modify files and directories they own.
- **System User:** This users created by systems. This users don't have login access and used run background process.
- A user have Username, UID(user id), Home directory of a user and Shell.

2. Groups in Linux

Groups are collections of users, which allow for easier management of file permissions and system access.

- **Primary Group:** Every user is a member of a primary group, which is specified when the user is created. Files created by the user are usually associated with this group.
- **Supplementary (Secondary) Groups:** Users can also belong to additional groups, which provide extra permissions.
- A group have Group Name and Group ID.

3. User and Group Files

- **/etc/passwd:** Contains user account information.
- **/etc/group:** Contains group information.
- **/etc/shadow:** Contains encrypted password information.
- **Create user and Password changing and changing one user to another**
 - Creating user in linux using `useradd` and `adduser` and check the users list in linux using `/etc/passwd` command

```
ubuntu@ip-172-31-30-122:~$ sudo adduser saikrishna
info: Adding user 'saikrishna' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group 'saikrishna' (1001) ...
info: Adding new user 'saikrishna' (1001) with group 'saikrishna (1001)' ...
info: Creating home directory '/home/saikrishna' ...
info: Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for saikrishna
Enter the new value, or press ENTER for the default
    Full Name []: saikrishna
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
info: Adding new user 'saikrishna' to supplemental / extra groups 'users' ...
info: Adding user 'saikrishna' to group 'users' ...
ubuntu@ip-172-31-30-122:~$ sudo useradd sai
```

```
ubuntu@ip-172-31-30-122:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534:/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:996:996:systemd Time Synchronization:/:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
messagebus:x:101:101:/nonexistent:/usr/sbin/nologin
syslog:x:102:102:/nonexistent:/usr/sbin/nologin
systemd-resolve:x:991:991:systemd Resolver:/:/usr/sbin/nologin
uidd:x:103:103:/run/uidd:/usr/sbin/nologin
tss:x:104:104:TPM software stack,,,:/var/lib/tpm:/bin/false
sshd:x:105:65534:/run/sshd:/usr/sbin/nologin
pollinate:x:106:1:/var/cache/pollinate:/bin/false
tcpdump:x:107:108:/nonexistent:/usr/sbin/nologin
landscape:x:108:109:/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:990:990:Firmware update daemon:/var/lib/fwupd:/usr/sbin/nologin
polkitd:x:989:989:User for polkitd:/:/usr/sbin/nologin
ec2-instance-connect:x:109:65534:/nonexistent:/usr/sbin/nologin
_chrony:x:110:112:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
saikrishna:x:1001:1001:saikrishna,,,:/home/saikrishna:/bin/bash
sai:x:1002:1002:/home/sai:/bin/sh
ubuntu@ip-172-31-30-122:~$
```

- Change the password of the user using `passwd` command and change one user to another user
how to change the password

```
ubuntu@ip-172-31-30-122:~$ ls -l /home
total 8
drwxr-x--- 2 saikrishna saikrishna 4096 Sep 11 07:45 saikrishna
drwxr-x--- 4 ubuntu      ubuntu      4096 Sep 11 07:43 ubuntu
ubuntu@ip-172-31-30-122:~$ sudo passwd sai
New password:
Retype new password:
passwd: password updated successfully
ubuntu@ip-172-31-30-122:~$
```

changing one user to another user in linux

```
ubuntu@ip-172-31-30-122:~$ su saikrishna
Password:
saikrishna@ip-172-31-30-122:/home/ubuntu$ ls
```

- Create group and add that group to user

Create group using `groupadd` command and add user to that group using `usermod` command and check the group is created or not

```
ubuntu@ip-172-31-30-122:~$ sudo groupadd voziq
ubuntu@ip-172-31-30-122:~$ sudo usermod -aG voziq saikrishna
```

check the user is created or not using `cat /etc/group`

```
ubuntu@ip-172-31-30-122:~$ cat /etc/group
ubuntu:x:1000:
admin:x:110:
netdev:x:111:
_chrony:x:112:
ubuntu:x:1000:
saikrishna:x:1001:
sai:x:1002:
voziq:x:1003:saikrishna
```

o

EXERCISES FOR DAY-1

1. Create, move and delete files and directories

- Created directories and files using `mkdir` and `touch` commands

```
ubuntu@ip-172-31-36-66:~$ mkdir spet
ubuntu@ip-172-31-36-66:~$ cd spet/
ubuntu@ip-172-31-36-66:~/spet$ ls
ubuntu@ip-172-31-36-66:~/spet$ mkdir abc1 abc2 abc3
ubuntu@ip-172-31-36-66:~/spet$ ll
total 20
drwxrwxr-x 5 ubuntu ubuntu 4096 Sep  5 18:51 ./
drwxr-x--- 16 ubuntu ubuntu 4096 Sep  5 18:50 ../
drwxrwxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc1/
drwxrwxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc2/
drwxrwxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc3/
```

Created directories `spet`, `abc1`, `abc2` and `abc3`

```
ubuntu@ip-172-31-36-66:~/spet$ ls -l
total 12
dr---wxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc1
drwxrwxrwx 2 ubuntu ubuntu 4096 Sep  5 18:51 abc2
drw-r--r-- 2 ubuntu ubuntu 4096 Sep  5 18:51 abc3
ubuntu@ip-172-31-36-66:~/spet$ touch file1.txt file2.txt file3.txt
ubuntu@ip-172-31-36-66:~/spet$ ll
total 20
drwxrwxr-x 5 ubuntu ubuntu 4096 Sep  5 18:58 ./
drwxr-x--- 16 ubuntu ubuntu 4096 Sep  5 18:50 ../
dr---wxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc1/
drwxrwxrwx 2 ubuntu ubuntu 4096 Sep  5 18:51 abc2/
drw-r--r-- 2 ubuntu ubuntu 4096 Sep  5 18:51 abc3/
-rw-rw-r-- 1 ubuntu ubuntu  0 Sep  5 18:58 file1.txt
-rw-rw-r-- 1 ubuntu ubuntu  0 Sep  5 18:58 file2.txt
-rw-rw-r-- 1 ubuntu ubuntu  0 Sep  5 18:58 file3.txt
```

Created files `file1.txt`, `file2.txt` and `file3.txt`

- Changing one directory to another using `cd` command

```
ubuntu@ip-172-31-36-66:~/spet$ cd abc2
ubuntu@ip-172-31-36-66:~/spet/abc2$ ls
```

- Moving a file to directory using `mv` command

```
ubuntu@ip-172-31-36-66:~/spet$ sudo mv file1.txt abc1
ubuntu@ip-172-31-36-66:~/spet$ ls
abc1 abc2 abc3 file2.txt file3.txt
```

- List files in a directory using `ls` command

```
ubuntu@ip-172-31-36-66:~/spet$ sudo ls abc1/
file1.txt
ubuntu@ip-172-31-36-66:~/spet$ |
```

- Removing or deleting a directory or file using `rm` command

```
ubuntu@ip-172-31-36-66:~/spet$ sudo rm -rf abc1
ubuntu@ip-172-31-36-66:~/spet$ ls
abc2 abc3 file2.txt file3.txt
ubuntu@ip-172-31-36-66:~/spet$ |
```

- Using `pwd` command for current working directory

```
ubuntu@ip-172-31-36-66:~/spet/abc2$ pwd
/home/ubuntu/spet/abc2
ubuntu@ip-172-31-36-66:~/spet/abc2$ |
```

2. Practice using basic shell commands to interact with the file system

- Finding and Searching

`find`: Finds files and directories based on conditions.

```
$ find /abc2 -name "file1.txt"
```

```
$ grep "test" file2.txt
```

```
$ grep -r "test" /abc1/file1.txt
```

- Viewing file contents

`less` This command allow you to view the content of file page by page

```
$ less file3.txt
```

`head` & `tail` These commands will display the first and last lines of a file. `head` is first lines and `tail` is last lines

```
$ head file1.txt
```

```
$ tail file2.txt
```

- Copy the files or directories

`cp` This command is used to copy the files or directories

```
$ cp file3.txt abc3
```

3. Create files with different permission settings and modify them using `chmod`

- I am giving a directory to some permissions like 644 users=read+write, group=read, others=read

```
ubuntu@ip-172-31-36-66:~/spet$ ls -l
total 12
dr---wxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc1
drwxrwxrwx 2 ubuntu ubuntu 4096 Sep  5 18:51 abc2
drwxrwxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc3
ubuntu@ip-172-31-36-66:~/spet$ chmod 644 abc3
ubuntu@ip-172-31-36-66:~/spet$ ls -l
total 12
dr---wxr-x 2 ubuntu ubuntu 4096 Sep  5 18:51 abc1
drwxrwxrwx 2 ubuntu ubuntu 4096 Sep  5 18:51 abc2
drw-r--r-- 2 ubuntu ubuntu 4096 Sep  5 18:51 abc3
ubuntu@ip-172-31-36-66:~/spet$
```

- Giving permissions to a file for 755 users=read+write+execute, group=read+execute, others=read+execute

```
$ chmod 755 file2.txt
```

Day-3

Process Management, System Monitoring and Networking

Process Management

- **Process** A process is basically a program that is executing. Executing the program happens in sequential. To run anything on OS, a process has to be created
 - A program is loaded into memory and it becomes process. Process has 4 sections
 - Stack: Stack contains temporary data such as what method/function parameters, return addresses & local
 - Heap: This is dynamically allocated to process
 - Data: Global & static variables
 - Text: Program counter, process register
 - **Process Life cycle**
 - Start
 - Ready
 - Running
 - Waiting
 - Terminated or exited
- **What is Process Management**

The fundamental functions of process management include starting, stopping, and arranging processes within an operating system. Consider a process as an instance of a program currently running, complete with its execution environment, memory, and CPU state. Due to its ability to manage several processes concurrently, Linux is a multitasking operating system that enables users to run multiple programs simultaneously.

 - **There are two types of processes :**
 - **1.Foreground** : These are the processes which are to be executed or initiated by the user or the programmer, they can not be initialized by system services. Such processes take input from the user and return the output.
 - **2.Background** : These are the processes that are to be executed or initiated by the system itself or by users, though they can even be managed by users. These processes have a unique PID or process id assigned to them

and we can initiate other processes within the same terminal from which they are initiated.

- **3.Stopped**

- **Commands and examples**

- `jobs`: List the jobs
- `fg %n`: Bring the current or specified job in the foreground
- `bg %n`: Place the current or specified job in the background
- `Conyto1-z` Stops the foreground job & places it in the background as stopped job

- **Examples**

- `sleep 1d` & job placed in the background
- `fg %1` job placed in the foreground (1 is job id number)

ubuntu@ip-172-31-12-49: ~

```
ubuntu@ip-172-31-12-49:~$ sleep 1d &
[1] 1713
ubuntu@ip-172-31-12-49:~$ jobs
[1]+  Running                  sleep 1d &
ubuntu@ip-172-31-12-49:~$ fg %1
sleep 1d
^Z
[1]+  Stopped                  sleep 1d
ubuntu@ip-172-31-12-49:~$
```

ec2-user@ip-172-31-3-82: ~

```
[ec2-user@ip-172-31-3-82 ~]$ sleep 1d &
[1] 4534
[ec2-user@ip-172-31-3-82 ~]$ jobs
[1]+  Running                  sleep 1d &
[ec2-user@ip-172-31-3-82 ~]$ fg %1
sleep 1d
^Z
[1]+  Stopped                  sleep 1d
[ec2-user@ip-172-31-3-82 ~]$
```

- **Commands for Process Management in Linux**

- Linux provides several commands for managing processes
- `ps` Report the snapshot of the current process. This command will show static info
 - `ps x` This command will show all of the our process regardless of what terminal they are controlled by.
 - `ps aux`
 - a**= All board: Every process, regardless of its owner, is displayed, democratizing process management.
 - u**= Know they user: It adds a human touch, showing the user behind each process, along with vital stats like CPU and memory usage.
 - x**= The hidden: Even processes lurking in the shadows, without a terminal to call home, are brought into the light.

```
ubuntu@ip-172-31-30-122:~$ ps
  PID TTY          TIME CMD
 26334 pts/0        00:00:00 bash
 26343 pts/0        00:00:00 ps
ubuntu@ip-172-31-30-122:~$
```

```
ubuntu@ip-172-31-30-122:~$ ps x
  PID TTY          STAT TIME COMMAND
 26190 ?            Ss    0:00 /usr/lib/systemd/systemd --user
 26191 ?            S      0:00 (sd-pam)
 26300 ?            S      0:00 sshd: ubuntu@pts/0
 26334 pts/0        Ss    0:00 -bash
 26367 pts/0        R+    0:00 ps x
ubuntu@ip-172-31-30-122:~$
```

- Explanation of the above proces State

R : Running

S : Sleeping

D : Uninterruptible sleep

T : Stopped

< : A priority process

N : A low priority process

```
ubuntu@ip-172-31-30-122:~$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  1.3 22532 13696 ?        Ss   Sep11    0:09 /usr/lib/systemd/systemd --system --deserialize=118
root         2  0.0  0.0      0     0 ?        S    Sep11    0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Sep11    0:00 [pool_workqueue_release]
root         4  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-rcu_g]
root         5  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-rcu_p]
root         6  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-slab_]
root         7  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-netns]
root        10  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/0:0H-events_highpri]
root        12  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-mm_pe]
root        13  0.0  0.0      0     0 ?        I    Sep11    0:00 [rcu_tasks_rude_kthread]
root        14  0.0  0.0      0     0 ?        I    Sep11    0:00 [rcu_tasks_trace_kthread]
root        15  0.0  0.0      0     0 ?        S    Sep11    0:00 [ksoftirqd/0]
root        16  0.0  0.0      0     0 ?        I    Sep11    0:00 [rcu_sched]
root        17  0.0  0.0      0     0 ?        S    Sep11    0:01 [migration/0]
root        18  0.0  0.0      0     0 ?        S    Sep11    0:00 [idle_inject/0]
root        19  0.0  0.0      0     0 ?        S    Sep11    0:00 [cpuhp/0]
root        20  0.0  0.0      0     0 ?        S    Sep11    0:00 [kdevtmpfs]
root        21  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-inet_]
root        23  0.0  0.0      0     0 ?        S    Sep11    0:00 [kauditd]
root        24  0.0  0.0      0     0 ?        S    Sep11    0:00 [khungtaskd]
root        25  0.0  0.0      0     0 ?        S    Sep11    0:00 [oom_reaper]
root        27  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-write]
root        28  0.0  0.0      0     0 ?        S    Sep11    0:06 [kcompactd0]
root        29  0.0  0.0      0     0 ?        SN   Sep11    0:00 [ksmd]
root        30  0.0  0.0      0     0 ?        SN   Sep11    0:00 [khugepaged]
root        31  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-kinte]
root        32  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-kbloc]
root        33  0.0  0.0      0     0 ?        I<   Sep11    0:00 [kworker/R-blkcg]
```

- Explanation of the above process

USER : User ID this is owner of the process

%CPU : Cpu usage in percent

%MEM : Memory usage in percent

VSZ : Virtual memory size

RSS : Resident set size. This is amount of physical memory the process is using in kilobytes

START: Time when the process is started

- top** This will provides all running Linux process. This command will show dynamic info.

It helps in monitoring system usage and performances It is mainly used to detect load on the server by system administration. I

The top command stands for table of processes. It is a task manager program, detected in several Unix-like operating systems, that shows information about memory and CPU utilization.

```

20343 ps/0 00:00:00 ps
ubuntu@ip-172-31-30-122:~$ top
top - 06:15:32 up 1 day, 22:39, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 106 total, 1 running, 105 sleeping, 0 stopped, 0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  957.4 total,  101.0 free,  389.6 used,  638.2 buff/cache
MiB Swap:   0.0 total,   0.0 free,   0.0 used.  567.8 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
    1 root        20   0   22532  13696  9600 S   0.0   1.4   0:09.63 systemd
    2 root        20   0     0     0     0 S   0.0   0.0   0:00.01 kthreadd
    3 root        20   0     0     0     0 S   0.0   0.0   0:00.00 pool_workqueue_release
    4 root         0 -20     0     0     0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
    5 root         0 -20     0     0     0 I   0.0   0.0   0:00.00 kworker/R-rcu_p
    6 root         0 -20     0     0     0 I   0.0   0.0   0:00.00 kworker/R-slub_
    7 root         0 -20     0     0     0 I   0.0   0.0   0:00.00 kworker/R-netns
   10 root         0 -20     0     0     0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
   12 root         0 -20     0     0     0 I   0.0   0.0   0:00.00 kworker/R-mm_pe
   13 root        20   0     0     0     0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
   14 root        20   0     0     0     0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
   15 root        20   0     0     0     0 S   0.0   0.0   0:00.59 ksoftirqd/0
   16 root        20   0     0     0     0 I   0.0   0.0   0:00.86 rcu_sched
   17 root        rt   0     0     0     0 S   0.0   0.0   0:01.03 migration/0
   18 root       -51   0     0     0     0 S   0.0   0.0   0:00.00 idle_inject/0
   19 root        20   0     0     0     0 S   0.0   0.0   0:00.00 cpuhp/0
   20 root        20   0     0     0     0 S   0.0   0.0   0:00.00 kdevtmpfs
   21 root         0 -20     0     0     0 I   0.0   0.0   0:00.00 kworker/R-inet_
   23 root        20   0     0     0     0 S   0.0   0.0   0:00.00 kauditd

```

Day-4

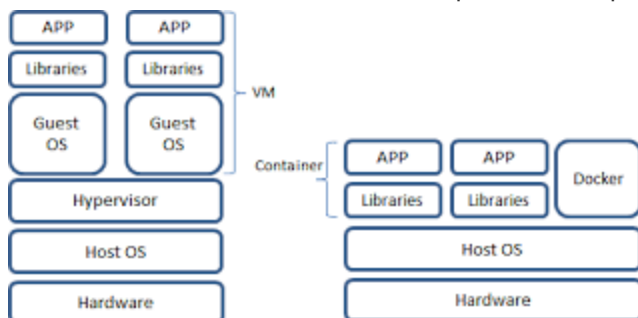
INTRODUCTION TO DOCKER

What is Docker?

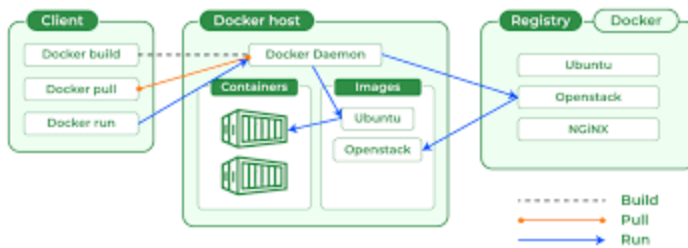
- Docker is a open-source containerization tool that allow you to build, test, deploy and manage virtualized application in containers on a common operating system.

How Docker works

- Docker works by providing a standard way to run your code. Docker is an operating system for containers.
- Similar to how a virtual machine virtualizes server hardware, containers virtualize the operating system of a server.
- Docker is installed on each server and providers simple commands you can use to build, start, or stop containers.



Docker Architecture



- **What is Dockerfile**

A Dockerfile is a script that contains instructions for building a customized docker image. Each instruction in a Dockerfile creates a new layer in the image, and the final image is composed of all the layers stacked on top of each other.

- **What is Docker Image**

It is a file, comprised of multiple layers, used to execute code in a Docker container. They are a set of instructions used to create docker containers. Docker Image is an executable package of software that includes everything needed to run an application.

- **What is Docker Container**

Docker container is a runtime instance of an image. Allows developers to package applications with all parts needed such as libraries and other dependencies. Docker Containers are runtime instances of Docker images.

- **What is Docker Hub**

Docker Hub is a repository service and it is a cloud-based service where people push their Docker Container Images and also pull the Docker Container Images from the Docker Hub anytime or anywhere via the internet.

- **Docker Engine**

The software that hosts the containers is named Docker Engine. Docker Engine is a client-server based application. The docker engine has 3 main components:

Server: It is responsible for creating and managing Docker images, containers, networks, and volumes on the Docker. It is referred to as a daemon process.

REST API: It specifies how the applications can interact with the Server and instructs it what to do.

Client: The Client is a docker command-line interface (CLI), that allows us to interact with Docker using the docker commands.

Docker daemon The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

Docker client The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

EXERCISES FOR DAY-4

1. Docker Installation on Linux Ubuntu

- Follow the official documentation [refer here](#)

```
# Add Docker's official GPG key: for import and verify the authenticity of docker repository package.
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ub
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# List the available versions:
$ apt-cache madison docker-ce | awk '{ print $3 }'

# VERSION_STRING= You want specific version then enter the above command the docker versions list will appear in that
$ sudo apt-get install docker-ce=$VERSION_STRING docker-ce-cli=$VERSION_STRING containerd.io docker-buildx-plugin dock

# Now check the Docker version
$ docker --version
```

2. Pull image and Run container

- Pull the nginx image

```
PS C:\Users\DELL> docker image pull nginx
Using default tag: latest
latest: Pulling from library/nginx
a2318d6c47ec: Pull complete
095d327c79ae: Pull complete
bbfaa25db775: Pull complete
7bb6fb0cfb2b: Pull complete
0723edc10c17: Pull complete
24b3fdc4d1e3: Pull complete
3122471704d5: Pull complete
Digest: sha256:04ba374043ccd2fc5c593885c0eacddebabd5ca375f9323666f28dfd5a9710e3
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
PS C:\Users\DELL> docker image ls
REPOSITORY      TAG           IMAGE ID       CREATED        SIZE
nginx           latest       39286ab8a5e1   3 weeks ago   188MB
PS C:\Users\DELL>
```

- Run the container

```
PS C:\Users\DELL> docker run -d -p nginx
7d7f455b61aa2f397c95ed4aeecf0eec4ba30c4fca523de35bd92e65f7f58e30
PS C:\Users\DELL> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                    NAMES
7d7f455b61aa   nginx    "/docker-entrypoint..." 6 seconds ago  Up 6 seconds  0.0.0.0:32768->80/tcp    pensive_feynman
PS C:\Users\DELL>
```

3. Docker basic commands

- basic docker commands

```
PS C:\Users\DELL> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
7d7f455b61aa   nginx    "/docker-entrypoint...." 2 minutes ago  Up 2 minutes  0
PS C:\Users\DELL> docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx         latest   39286ab8a5e1   3 weeks ago   188MB
PS C:\Users\DELL> docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
7d7f455b61aa   nginx    "/docker-entrypoint...." 3 minutes ago  Up 3 minutes  0
PS C:\Users\DELL> docker container stop 7d7f455b61aa
```

- Stoppin and removing container

```
PS C:\Users\DELL> docker container stop 7d7f455b61aa
7d7f455b61aa
PS C:\Users\DELL> docker container rm 7d7f455b61aa
7d7f455b61aa
PS C:\Users\DELL> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
PS C:\Users\DELL> docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
PS C:\Users\DELL> |
```

4. Simple Dockerfile for nignx image taking base image ubuntu deploy it at access in your web browser,and push to docker registry.

- Dockerfile

```
FROM ubuntu:latest
RUN apt update && apt install nginx -y
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- Build docker image

```
PS C:\Users\DELL\voziq> docker build -t test .
[+] Building 41.5s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 142B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/2] FROM docker.io/library/ubuntu:latest@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee
=> => resolve docker.io/library/ubuntu:latest@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee
=> => sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee 1.34kB / 1.34kB
=> => sha256:d35dfc2fe3ef66bcc085ca00d3152b482e6cafb23cdda1864154caf3b19094ba 424B / 424B
=> => sha256:edbf74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a 2.30kB / 2.30kB
=> => sha256:31e907dcc94a592a57796786399eb004dcbb714389fa615f5efa05a91316356 29.71MB / 29.71MB
=> => extracting sha256:31e907dcc94a592a57796786399eb004dcbb714389fa615f5efa05a91316356
=> [2/2] RUN apt update && apt install nginx -y
=> exporting to image
=> => exporting layers
=> => writing image sha256:456035e0cdb72340d412991fc50691fcb9a655038c99dc0f36c92561f2e64f8
=> => naming to docker.io/library/test
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/26e315jtte1io9gopyr1p723g
```

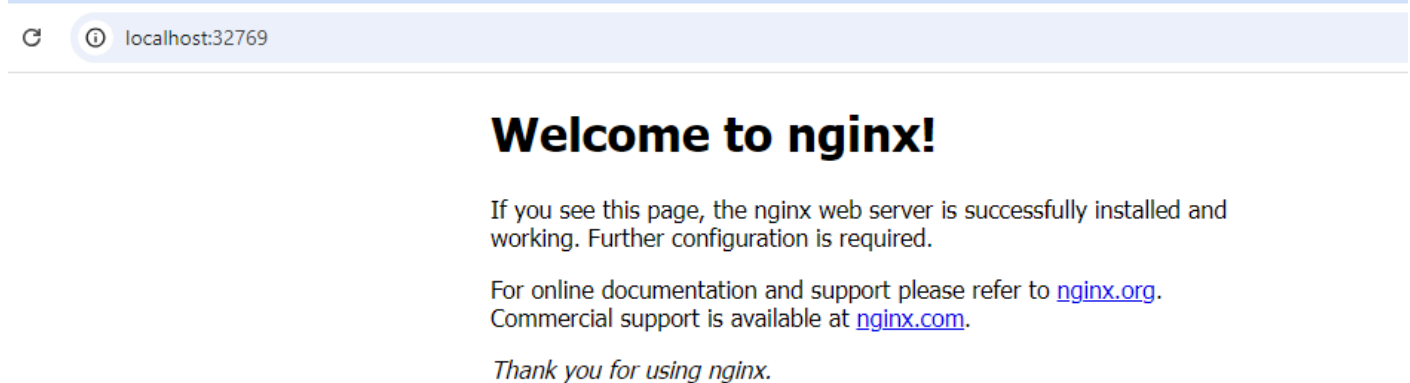
- Run as container


```

PS C:\Users\DELL\voziq> docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
test latest 456035e0cdb7 7 seconds ago 126MB
nginx latest 39286ab8a5e1 3 weeks ago 188MB
PS C:\Users\DELL\voziq> cat .\Dockerfile
FROM ubuntu:latest
RUN apt update && apt install nginx -y
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
PS C:\Users\DELL\voziq> docker run -d -p test
d6a534e3af7f884b785fd89747d1eee0ce0fb8045ef9d3f91ace18c88c8a94fe
PS C:\Users\DELL\voziq> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d6a534e3af7f test "nginx -g 'daemon of..." 4 seconds ago Up 3 seconds 0.0.0.0:32769->80/tcp nice_ganguly
PS C:\Users\DELL\voziq>

```

- Accessing via browser port number 32769



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

- Tag docker image

```

PS C:\Users\DELL\voziq> docker tag test:latest saikrishna38721/test:latest

```

- Login your dockerhub credentials and push tagged image to dockerhub

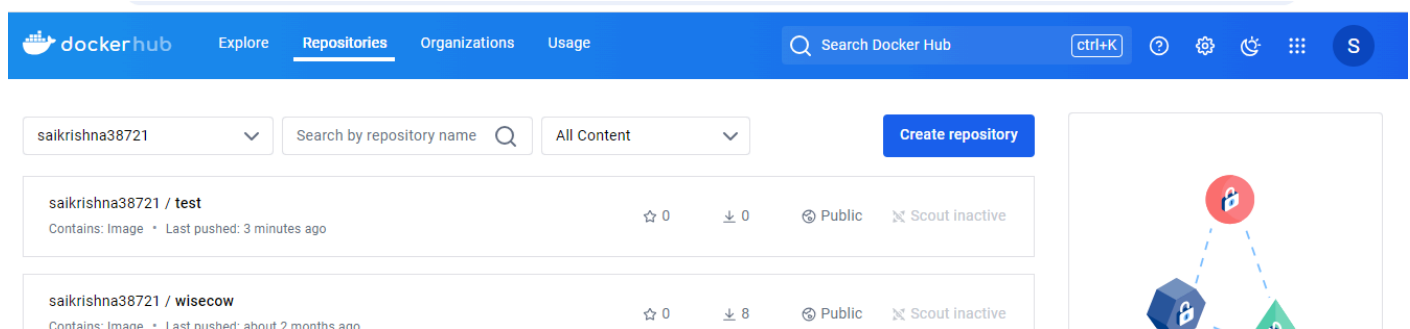
```

PS C:\Users\DELL\voziq> docker login -u saikrishna38721
Password:

Login Succeeded
PS C:\Users\DELL\voziq> docker push saikrishna38721/test:latest
The push refers to repository [docker.io/saikrishna38721/test]
43088ec9df2a: Pushed
f36fd4bb7334: Pushed
latest: digest: sha256:d6a8ff4bde35aae206c17d3bf5e0d777ae5cb5db29ea60218a69804ca4355e7f size: 741

```

- In dockerhub new image test is present



DAY-5,6

Working with Docker Volumes and Docker Networks

Docker Volumes:-

Docker volumes are a mechanism for persistently storing data generated or used by Docker containers. They provide a means to separate data from the container's lifecycle, allowing containers to be stopped, started, or even destroyed without affecting the data stored within the volumes.

Benifits of Docker Volumes

- **Data Persistence:** Volumes ensure that data remains intact even when containers are updated or replaced. This allows for seamless data management and avoids data loss or disruption.
- **Sharing Data between Containers:** Volumes provide a way to share data between containers running on the same host. Multiple containers can access and modify the data within a volume, facilitating collaboration and decoupling applications.
- **Easy Backup and Restore:** Docker volumes simplify the process of backing up and restoring data. Since volumes exist independently of containers, you can easily create backups of the data stored within volumes and restore them as needed.

Types of Docker Volumes

- **Named Volumes:** Named volumes are the most commonly used type of volume in Docker. They are created and managed by Docker itself. You can give a volume a specific name, and Docker takes care of creating and maintaining the volume for you. Named volumes are easy to use and provide better readability in your Docker commands.
- **Host Bind Mounts:** Host bind mounts allow you to mount a directory from your host machine into a Docker container. With this type of volume, the directory's contents on the host are directly accessible to the container. Changes made in the container will be reflected on the host and vice versa. Host bind mounts provide a way to share data between the host and the container.
- **Tmpfs Mounts:** Tmpfs mounts are stored in the host's memory and not written to the host filesystem. They are useful when you need a lightweight and temporary storage option. Tmpfs mounts are created and managed by Docker, and their data resides in the host's memory. Once the container is stopped or removed, the data in a tmpfs mount is lost.

Docker Networking:-

Docker networking enables containers to communicate with each other and with external systems. It provides isolation, security, and flexibility by creating virtual networks that connect containers, allowing them to share data, services, and resources while maintaining separation.

Benefits of Docker Networking

- **Container Isolation:** Docker networking allows containers to operate in isolation while providing controlled communication channels between them. Each container can have its own network namespace, IP address, and port space.
- **Scalability and Load Balancing:** Docker networking simplifies the process of scaling containerized applications. By leveraging load balancing techniques and network overlay features, traffic can be distributed across multiple containers, improving performance and reliability.
- **Service Discovery:** Docker provides built-in mechanisms for service discovery within a network. Containers can refer to each other using service names rather than hard-coded IP addresses, allowing for dynamic configuration and easy updates.
- **Connectivity with External Networks:** Docker containers can connect to external networks, allowing interaction with other systems, databases, or services running outside the Docker environment. This enables seamless integration with existing infrastructure.

Types of Docker Networks

- **There are 3 types of Networks are available in Docker**
- **bridge:-** If you build a container without specifying the kind of driver, the container will only be created in the bridge network, which is the default network.
- **host:-** Containers will not have any IP address they will be directly created in the system network which will remove isolation between the docker host and containers.
- **null:-** These containments are not accessible to us from the outside or from any other container.