Session-6: (Cover till 1hour 36mints)

- 3 Tier Architecture
- Project Started
- MySQL configured(Data Base)

3 tier architecture

Desktop applications

Web applications

Disadvantages of Desktop applications

1. Storage
2. Installation
3. Upgrade
4. Compatibility
5. If system crash we will lose data
6. More system resources required or consumes

Web applications (simply available on online with cloud storage option)

Road side cart	Hotel
1 person → 10 persons	50 people → He will hire extra resources
Cooking	2 persons → 50 people
Bill collection	1 cook → cooking and serving
Serving	1 owner → tokens issue, bill collection
Queuing	

500 people → **restaurant**

1 captain → welcome and show the table

Waiter → take the order, plating

Chef → cook the order

Raw Items → Cook (customers can eat) ⇨ Plating (with onion and Keera)

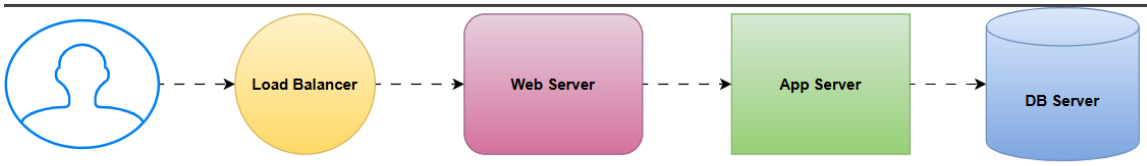
- **Responsibilities are shared to everyone, they can focus only on their work.**
- **Security**
- **Queuing**

Previously only one server → DB, Java Application, HTML application

The main job of an application server is to perform CRUD operations → create, read, update and delete.

Email, name, pan card, card details

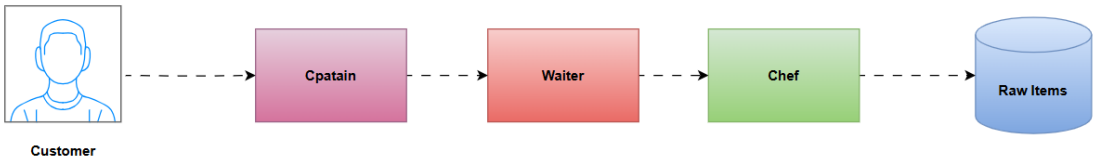
Users --> table --> RDBMS (Relational Database Management System).

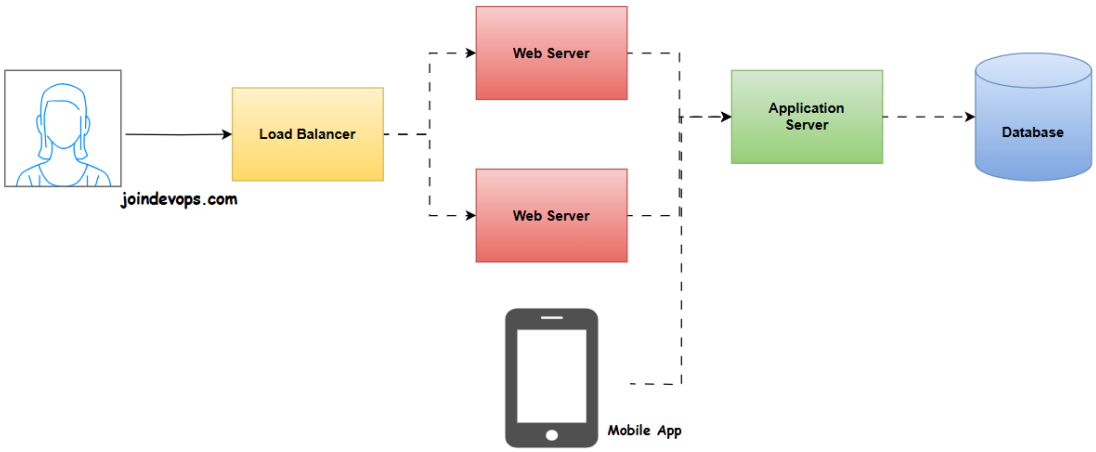


DB server→ *raw data*

Application Server → *Connects to DB and do CRUD operations*

Web Server → take the requests, queue the request, forward the request to application server, and format the data





DB Tier → RDBMS (MySQL, Oracle, Postgres, etc.), NoSQL (MongoDB), Redis (Cache), RabbitMQ (Queue based)

NoSQL stands for **"Not Only SQL."**

It refers to a category of databases that provide a mechanism for storage and retrieval of data that is modelled in ways other than the traditional tabular relations used in **relational databases (SQL databases)**.

Key points about NoSQL:

- Designed for **scalability and flexibility**
- Supports **unstructured**, **semi-structured**, or **structured** data
- Common types: **Document**, **Key-Value**, **Column-Family**, and **Graph** databases
- Popular examples: **MongoDB**, **Cassandra**, **Redis**, **Neo4j**

Application/API (Application programming interface) Tier → Backend/middleware applications --> Java, .NET, Python, Go, NodeJS, etc.

```
{
        "username": "sivakumar",
        "dob": "01-JAN-2000",
        "address": "Sanath nagar, HYD, 543234"
}
```

What is JSON Format?

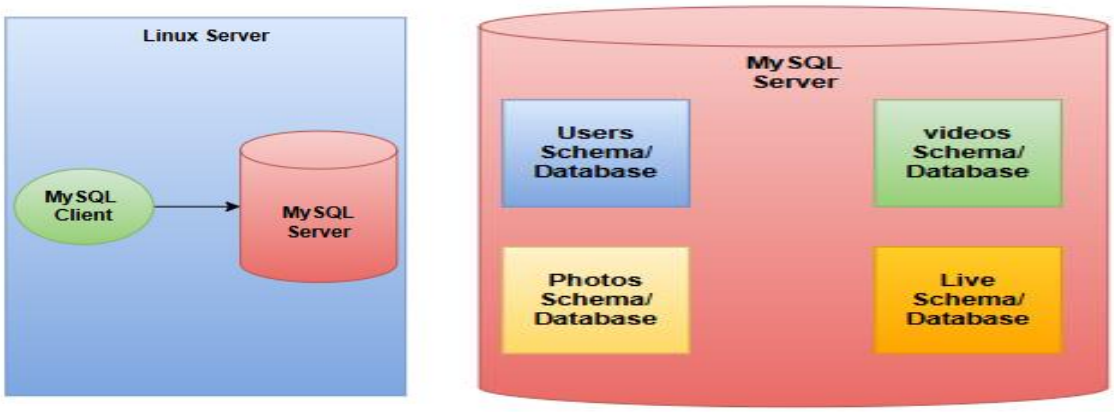
JSON stands for **JavaScript Object Notation**.
It is a **lightweight, human-readable data format** used to store and exchange data between systems, especially between a server and a web application.

Web (Frontend tier) tier → Load Balancer, Frontend Servers -> HTML, CSS and JS, React JS, AngularJS.

MERN --> MongoDB, ExpressJS, ReactJS, NodeJS

devops-practice --> joindevops (RHEL9 based) --> ec2-user, DevOps321 (RHEL-9-DevOps-Practice)

Linux Server --> Physical Server



show databases; → displays the schema/database available

use <database-name>; → you are using that schema

show tables; → display all the tables in the schema

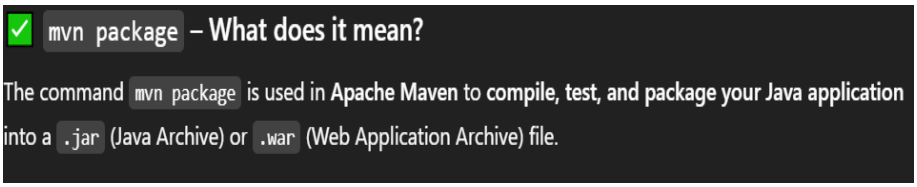
select * from table-name; → display the data inside table

Application Server:-

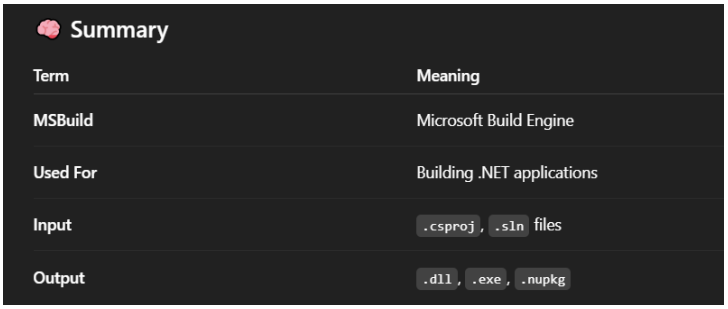
#include <stdio.h> → our C programming depends on this... so these are called as dependencies/libraries

NodeJS → package.json (build file) (contains dependencies/libraries required by NodeJS)

Java → pom.xml --> project version, description, dependencies/libraries mvn package



.NET → msbuild --> project version, description, dependencies/libraries



Python → requirements.txt → project version, description, dependencies/libraries pip install

C language → Makefile → make

npm install → here npm means **Node Package Manager (NPM)** → **we should run this command only where package.json folder is available**

systemD --> here D means Daemon

`**simply systemctl is used only for services.**

Service → actually service means it should run continuously

--> /etc/systemd/system

--> create a .service file

Practice:-

****** DATBASE (MySQL) ********private IP address: 172.31.18.102 ******

ssh ec2-user@54.224.177.26

password is DevOps321

sudo su -

dnf install mysql-server -y

systemctl start mysqld

systemctl enable mysqld

systemctl status mysqld → to check the status

ps -ef | grep mysql → to check any processes (mysql) is working or not.

netstat -lntp → to check ports status ----mysqld default port number is 3306

sudo su -

mysql_secure_installation --set-root-pass ExpenseApp@1

mysql -h <host-address> -u root -p<password> → to login to the database but client and server both are located in the same server so we are not using this command line.

mysql

show databases; → dispays the schema/database available

use <database-name>; → you are using that schema

use mysql

show tables; → display all the tables in the schema

select * from <table-name>; → display the data inside table

exit;

mysqld default port number is 3306

****

****** BACKEND **********private IP address: 172.31.92.96 **********

ssh ec2-user@ 52.90.79.129

password is DevOps321

sudo su -

dnf list available | grep nodejs → To check which version is available in our server

dnf module disable nodejs -y

dnf module enable nodejs:20 -y

dnf list available | grep nodejs

dnf install nodejs –y

useradd expense

mkdir /app

curl -o /tmp/backend.zip https://expense-builds.s3.us-east-1.amazonaws.com/expense-backend-v2.zip

cd /app

unzip /tmp/backend.zip

ls -l

npm install → this command looks for package.json folder

ls -l

cd node_modules/ → Here all dependencies are listed which are downloaded for nodejs using npm package.json

vim /etc/systemd/system/backend.service

```
[Unit]
Description = Backend Service

[Service]
User=expense
Environment=DB_HOST="172.31.92.96"
ExecStart=/bin/node /app/index.js
SyslogIdentifier=backend

[Install]
WantedBy=multi-user.target
```

```
systemctl daemon-reload
systemctl start backend
systemctl enable backend
systemctl status backend
    dnf install mysql -y
    mysql -h 172.31.92.96 -u root -pExpenseApp@1
mysql -h 172.31.92.96 -u root -pExpenseApp@1 < /app/schema/backend.sql
systemctl daemon-reload
systemctl restart backend
systemctl status backend
```

Node default port number is 8080

😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊

Session-7: (Cover till 1hour 20 mints)

- PublicIP vs PrivateIP
- Putty, MobaXterm
- **Reverse proxy vs Forward proxy**
- Backend Configuration
- Frontend Configuration
- Quiz

systemctl service?

if you want your applications to run as a service, create a file with extension .service in /etc/systemd/system

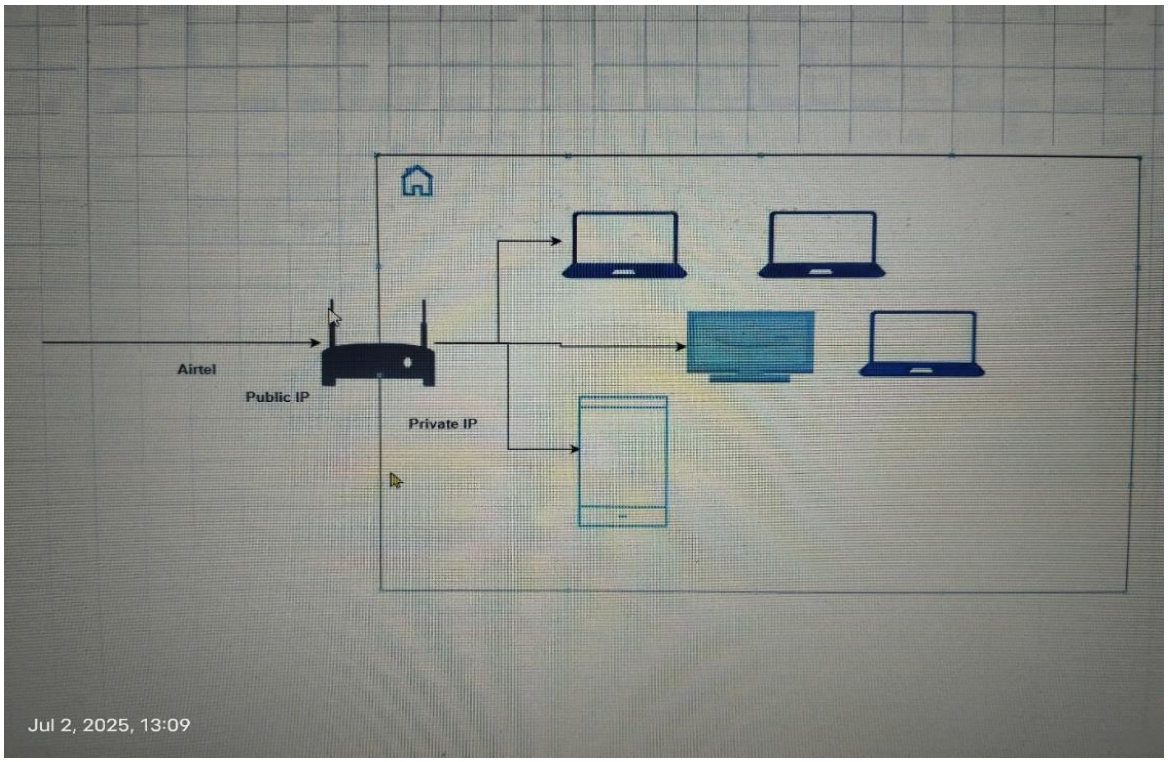
vim /etc/systemd/system/backend.service

```
[Unit]
Description = Backend Service

[Service]
User=expense
Environment=DB_HOST="172.31.92.96"
ExecStart=/bin/node /app/index.js
SyslogIdentifier=backend

[Install]
WantedBy=multi-user.target
```

Mostly every device which has internet connection has two IP address one is private IP address and second one is public IP address.

browser: 49.204.161.202 --> public IP (my IP address on browser)

cmd: 192.168.0.107 --> private IP (Ipconfig in command prompt)



🌐 Total Number of Public IP Addresses in Existence

Public IP addresses are primarily governed by **IPv4** and **IPv6**.

📦 IPv4 (Internet Protocol version 4)

- IPv4 addresses are **32-bit**.
- Total possible unique IP addresses = `2^32` = **4,294,967,296** (~4.3 billion)

But not all are usable:

- Some are **reserved for private use** (like 192.168.x.x, 10.x.x.x, etc.)
- Some are **reserved for special purposes** (like 127.0.0.1 for localhost)

✅ **Usable Public IPv4 addresses**: Around **3.7 billion**

📦 IPv6 (Internet Protocol version 6)

- IPv6 addresses are **128-bit**.
- Total possible unique addresses = `2^128` = **340,282,366,920,938,463,463,374,607,431,768,211,456**
  (That's **340 undecillion** addresses!)

✅ **Effectively unlimited** for current and future needs.

📌 Summary Table

Protocol	Address Size	Total Addresses	Usable Public IPs
IPv4	32-bit	~4.3 billion (`2^32`)	~3.7 billion
IPv6	128-bit	~340 undecillion (`2^128`)	Practically unlimited

✅ Final Answer:

- **Total Public IPv4**: ~3.7 billion
- **Total Public IPv6**: ~340 undecillion (enough to assign trillions of addresses per device)

Let me know if you want to see reserved IPv4 ranges or how IP allocation works globally!

Proxy --> forward proxy and reverse proxy

A VPN (Virtual Private Network) is *not technically* a forward proxy, but it performs similar functions with more security and wider coverage.

Server is not aware that client is using VPN. Client is aware of VPN

- Traffic restrict, traffic monitoring
- Geolocation hiding
- Anonymous client identity
- Access private network/files

Reverse proxy

Client is not aware of proxy. Server is aware of proxy.

- Backend applications are behind reverse proxy servers for security and queueing.
- Cache servers.

Nginx --> popular webserver and reverse proxy server

nginx home directory: /etc/nginx

nginx configuration: /etc/nginx/nginx.conf

html directory: /usr/share/nginx/html

Total number of ports (0 to 65,535) = 65,536 ports

JoinDevOps AMI

https://github.com/learndevopsonline/aws-image-devops-session.

Frontend server (or) web serverprivate IP address: 172.31.93.152

ssh ec2-user@ 44.202.123.183

sudo Su -

dnf install nginx -y

systemctl enable nginx

systemctl start nginx

systemctl status nginx

netstat -lntp

cd /usr/share/nginx/html

rm -rf /usr/share/nginx/html/*

cd

curl -o /tmp/frontend.zip https://expense-builds.s3.us-east-1.amazonaws.com/expense-frontend-v2.zip

cd /usr/share/nginx/html

unzip /tmp/frontend.zip

systemctl restart nginx

vim /etc/nginx/default.d/expense.conf

```
proxy_http_version 1.1;

location /api/ { proxy_pass http:// 172.31.88.121:8080/; }

location /health {
  stub_status on;
  access_log off;
}
```

systemctl restart nginx

Ngnix default port number is 80

If 44.202.123.183 address is accessible, project is sucessful.

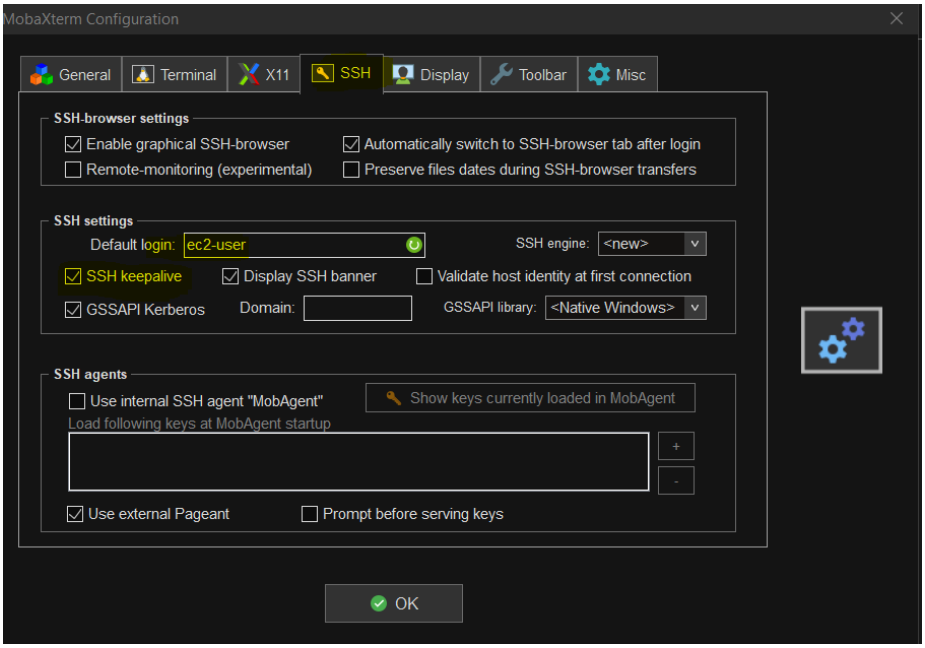
☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-8: (Cover till 1hour 28 mints)

- How does DNS work?
- Domain booking
- Project using DNS
- Inode, symlink and hardlink

What is DNS?
How DNS works?

Default settings for MobaXterm app for easy use



Public IP --> stop and start then we can see change in IP
Private IP --> but when terminate and recreate private, IP changes

Human names, computers numbers

Whenever backend IP changes, I should edit systemctl file. deamon reload and restart the service

Word = meaning → Dictonary
Name = number →mobile contact savings process
Facebook = IP

ISP → internet service provider (example BSNL, Airtel).

ICANN --> Internet Corporation for assigned names and numbers --> countries, reputed organisations

There are 13 root servers in the world

Top Level Domain(TLD)

.telugu
.com
.in
.uk
.net
.edu
.gov, .us, .au, .org, .ai, .online

.gov.in,
.co.in → sub level domain

ICANN --> I am going to start .telugu domain. I need to complete all the process

joindevops.telugu
tfc.telugu

domain registers(mediators) --> godaddy, hostinger, aws, gcp, azure

joindevops --> joindevops.com(not available), try joindevops.telugu
someone registered joindevops.telugu

Hostinger updates **Radix Registry** about daws82s**.online** --> who bought this domain and nameservers

nameservers = who managed this domain = records to the DNS

A record = IP address

change in NS --> Hostinger updates the change of Nameservers to .online TLD

now aws manages my domain

mysql.daws82s.online --> DNS resolver --> .online TLD --> provides nameservers to daws82s.online --> mysql.daws82s.online A record

Record types

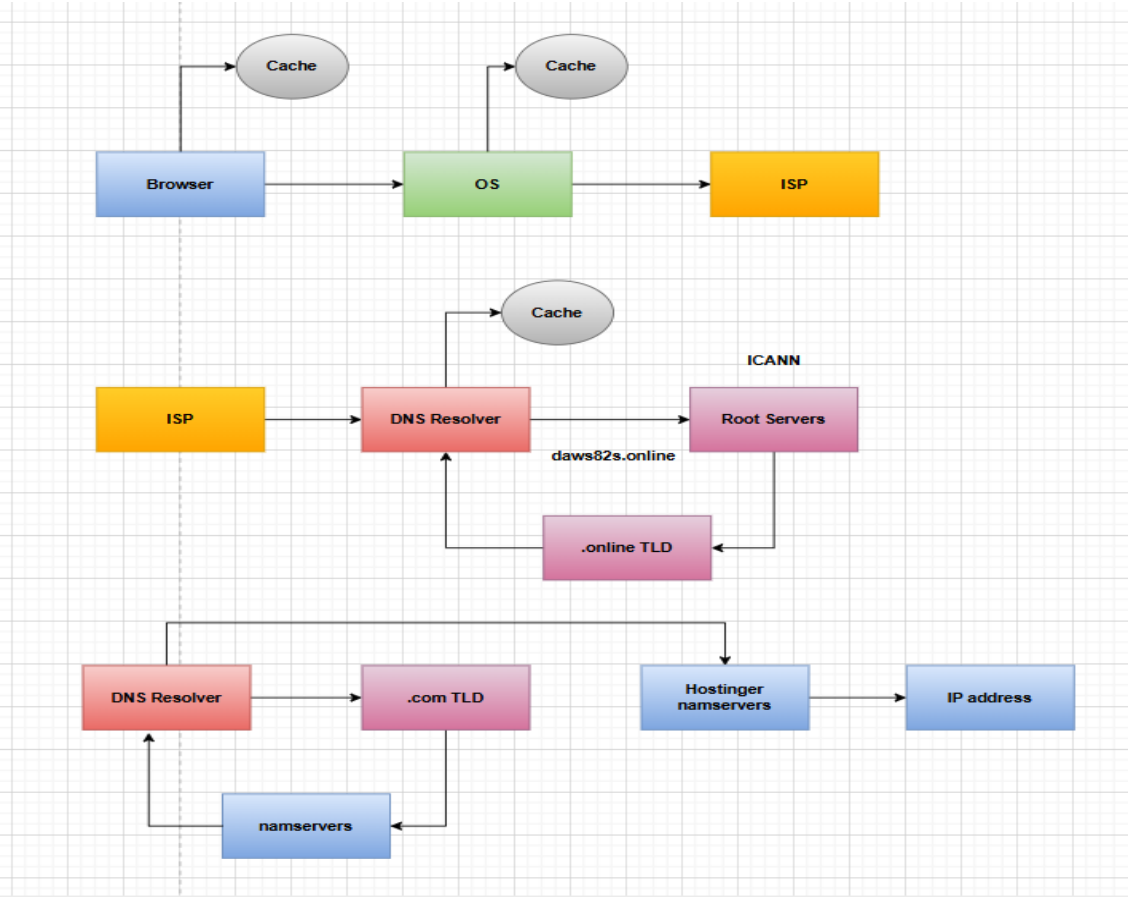
A --> points to IP address
CNAME --> points to another domain
MX --> mail records (info@joindevops.com)
TXT --> Domain ownership validaton purpose
NS --> nameservers
SOA --> who is the authority of this domain

What happens when we book domain?
What happens when someone enter our domain in browser?
How to become TLD?

http://daws81s.online/api/transaction

http://backend.daws81s.online:8080/transaction

http://daws81s.online/api/transaction --> send request to backend --> backend responds with data



Vim /etc/systemd/system/backend.service

```
[Unit]
Description = Backend Service

[Service]
User=expense
Environment=DB_HOST="mysql.psk135.tech"
ExecStart=/bin/node /app/index.js
SyslogIdentifier=backend

[Install]
WantedBy=multi-user.target
```

Vim /etc/nginx/default.d/expense.conf

```
proxy_http_version 1.1;

location /api/ { proxy_pass http://backend.psk135.tech:8080/; }

location /health {
  stub_status on;
  access_log off;
}
```

Inode, symlink/softlink and hardlink

what is inode?(explained with backend server or app server)

inode stores the file type(file or folder), permissions, ownership, file size, timestamp, disk location(memory location)

cd /app/

ls -li → gives us Inode number in 1st column

stat < filename> or <folder_name > → gives us full details

stat DbConfig.js

symlink/softlink

ln -s DbConfig.js DbConfig1.js → Here DbConfig1.js is shortcut for DbConfig.js but Inode number is not same.

ls -l

lrwxrwxrwx 1 root root   11 Dec 26 03:10 DbConfig1.js -> DbConfig.js

l represents link file

cat DbConfig1.js

stat DbConfig.js

stat DbConfig1.js

echo "Hello world" > hello.txt

ls -li

ln -s hello.txt hi.txt

rm -rf hello.txt

ls -li

rm -rf hi.txt

symlink is like shortcut it points to the original file. Symlink file's inode and actual file inode is different.

- symlink breaks when actual file is deleted.
- symlink can be created to folders/directories

hardlink

echo "Hello world" > hello.txt

ln hello.txt hi.txt →here we created hardlink

ls –li

Hardlink inode is same as actual file. hardlink is useful for backup of the file.

- If original file is deleted hardlink remains same.
- we can't create hardlinks to folders/directories

how do you findout hardlinks for a particular file?

find / -inum "<inode-number>

find / -inum xxxxx

why we have to use hardlink if it behaves as a copy file?

Because to keep updated of all copies of original file. Through the link

==========My Practice part==========

https://howdns.works/ --- to understand DNS

Go to AWS account

Route 53 --> Hosted zones --> psk135.tech

Below are the name serves of psk135.tech domain which are now organised by AWS.

ns-1492.awsdns-58.org

ns-1926.awsdns-48.co.uk

ns-393.awsdns-49.com

ns-594.awsdns-10.net

Now create new records in psk15.tech hosted zone.

mysql.psk135.tech → 172.31.88.121 → TTL=1 second.

backend.psk135.tech → 172.31.92.96 → TTL=1 second.

psk135.tech → 44.202.123.183 → TTL=1 second.

To check status of above records on hosted zones, run below commands on local git bash

nslookup mysql.psk135.tech

nslookup backend.psk135.tech

nslookup psk135.tech

Trouble shooting techniques.

First check NETWOKS Tab

Second check Nginx logs (cat /var/log/nginx/access.log) on front end server

Third check logs on backend server (cat /var/logs/messages)

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-9: (Cover till 1hour 22 mints)

- HTTP Methods and status codes
- Troubleshooting commands
- Linux Folder structure

ping ip

ping mysql.psk135.tech

to exit from ping clik CTRL + C

telnet <db-IP> 3306 → DB running but backend not able to connect DB == check DB security group ingress rules

telnet mysql.psk135.tech 3306

Ping sometimes not able to connect but telnet should always connect to that URl or IP-address.

Same server == localhost == 127.0.0.1

curl http://localhost:8080/health

HTTP Methods and status codes

HTTP Methods:-

CRUD

GET --> getting/read from server

POST --> posting/create the information

```
{
    amount: "200",
    desc: "travel"
}
```

PUT --> Update the information

DELETE --> Delete the information

100 == 1XX == Informational codes

200 == 2XX == Success status codes

300 == 3XX == Redirectional

400 == 4XX == Client side error

500 == 5XX == Server side error

backend.daws82s.onine --> 404 --> NOTFOUND --> Client side error

403 --> Forbidden --> You dont have access to that

401 --> Unauthorized --> you should login

405 --> HTTP POST, If you use GET --> Method not allowed

400 --> bad request --> check the payload data once again

500 --> Internal Server Error --> Server side error

502 --> Bad Gateway --> Frontend not able to connect backend

503 --> Service temporarily unavailable

In front end server to check live logs we use below command at

cd /var/log/nginx

tail –f access.log

ctrl+c → for exit

How to check memory of linux server? memory == RAM

free → shows data in human bytes

free –h → human readable format of free command

RAM vs ROM

HD --> RAM --> User

Swap (Reserved RAM from HD)

top

press q to exit.

dnf install htop –y → we are installed it to see top command data with human readable formant.

htop

press F10 to exit

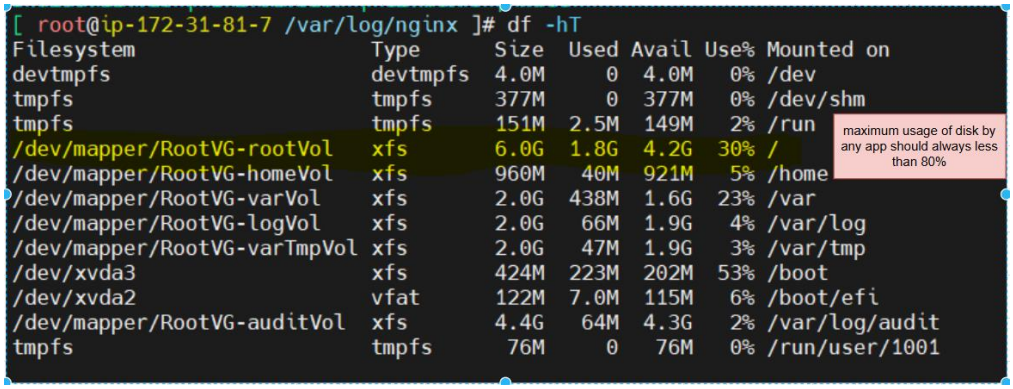
cat /proc/meminfo

How do you list top 10 high memory process?

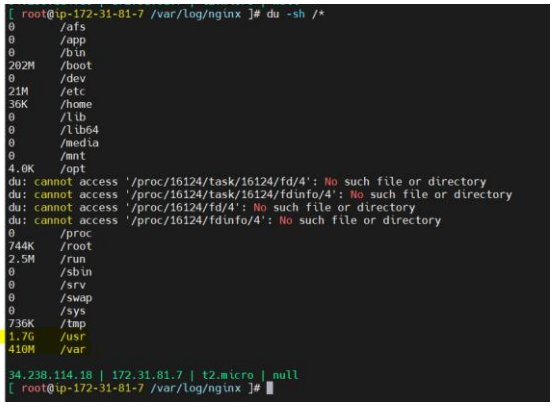
ps aux --sort -%mem | head -n 10

Disk usage?

df -hT



du -sh /* → gives us the disk usage of files and folders in root directory



du -sh /user/* → gives us the disk usage of files and folders in user directory

du -sh * → gives us the disk usage of files and folders in current directory

cat /proc/cpuinfo → for CPU information

linux-filesystem

https://github.com/DAWS-82S/concepts/blob/main/linux-filesystem.MD

Linux FileSystem Hierarchy

```
#SUB-DIRECTORY PURPOSE
/bin      common binary executables used by all users
/boot     files associated with boot loader
/dev      attached devices (usb, cdrom, mouse, keyboard)
/etc      configuration files
/home     personal directories for each user account
/lib      shared system libraries
/media    directory for mounting removable devices (floppy drive, cdrom)
/mnt      directory for mounting filesystems (nfs, smb)
/opt      optional vendor add-on software
/proc     virtual filesystem for system processes/resources information
/root     home directory for administrator account
/run      storage for runtime information
/sbin     binary executables used by administrator
/srv      data for server services
/sys      virtual filesystem for hardware/driver information
/tmp      temporary files purged on reboot
/usr      utilities and read-only user data/programs
/var      variable and log files
```

- The Linux filesystem hierarchy is organized into a tree-like structure starting at the root directory (/).
- Each directory serves a specific purpose, and understanding it is essential for navigating and managing Linux systems. Here's an overview of the key directories:

Root(/)

- The top-level directory of the filesystem.
- All other directories branch out from here.
- Only accessible by the root user for administrative tasks.

Bin(/bin)

- Contains essential user command binaries (programs) needed for system operation.
- Examples: `ls, cp, mv, cat, grep.`

Boot(/boot)

- Contains files required to boot the system, such as: The Linux kernel (vmlinuz).
- Bootloader files (e.g., GRUB configurations).
- Example: `/boot/grub/grub.cfg.`

Dev(/dev)

- Device files: Special files representing hardware devices (e.g., disks, USBs) and virtual devices.
- Examples: `/dev/sda (hard drive).`

Etc(/etc)

- Configuration files: System-wide configuration files and scripts.
- Examples:

```
/etc/passwd (user accounts).
/etc/hosts (hostname mappings).
```

Etc(/etc)

- Configuration files: System-wide configuration files and scripts.
- Examples:

```
/etc/passwd (user accounts).
/etc/hosts (hostname mappings).
/etc/systemd/system (service files)
/etc/yum.repos.d (repos)
/etc/nginx (Nginx config directory)
```

Home(/home)

- Contains personal directories for each user.
- Example: `/home/siva (Siva's home directory).`

Lib(/lib & /lib64)

- Essential shared libraries needed for binaries in /bin and /sbin.
- Examples: `/lib/libc.so.6 (C library).`

Media(/media)

- Removable media: Mount point for external devices (e.g., USB drives, DVDs).
- Example: `/media/usb.`

Mnt(/mnt)

- Temporary mount points: Used for mounting filesystems temporarily during maintenance or installation.

Mnt(/mnt)

- Temporary mount points: Used for mounting filesystems temporarily during maintenance or installation.
- Example: `/mnt/temp`

Opt(/opt)

- Optional software: Third-party software packages like tomcat server, prometheus, etc.
- Example: `/opt/prometheus`

Proc(/proc)

- Virtual filesystem: Provides system information and kernel data as files.
- Examples:

```
/proc/cpuinfo (CPU details).
/proc/meminfo (memory usage).
```

Root(/root)

- Root user's home directory: Personal directory for the root user.
- Example: `/root/.bashrc`

Run(/run)

- Runtime files: Stores temporary system information since the last boot.
- Examples: `/run/utmp (active user sessions)`

Sbin(/sbin)

- System binaries: Contains administrative commands and utilities used by the root user.
- Examples: `iptables, reboot.`

Srv(/srv)

- Service data: Contains files related to specific services (e.g., web servers).
- Example: `/srv/www (website data)`

Sys(/sys)

- System information: Interface for accessing and managing kernel-related hardware information.
- Example: `/sys/class/net (network devices)`

Swap(/swap)

Used as swap space for virtual memory when RAM is full.

Tmp(/tmp)

- Temporary files: Stores temporary data; cleared on reboot.
- Examples: Session files, temporary cache.

Usr (/usr)

- User programs: Contains user applications and utilities.
- Example

Which ls → we use this command to check where this command is listed→/bin

Which cat → data is available in bin

🧠 For DevOps, special focus on:

- **Logs:** `/var/log`
- **Configs:** `/etc`
- **System services:** `/etc/systemd/system/`
- **User scripts:** `/usr/local/bin` or `/opt`
- **Docker/Kubernetes:** `/var/lib/docker`, `/etc/docker`, `/etc/kubernetes`

Explained Linux booting process

https://www.youtube.com/watch?v=XpFsMB6FoOs

If possible go to devops and cloud with Siva YouTube channel and cover below play lists

AWS IAM—4 videos

NETWORKING BASICS—2 videos

1-10 commands

human errors

time taking

Shell Scripting

If you keep all your commands in a single file and execute that file --> Shell Scripting

native linux scripting --> Linux/Shell commands

Linux Server --> I need to fetch some info from AWS Cloud --> Python

For Blog writting below topics can be considered

1. Linux commands we use on daily basis
2. Forward proxy vs Reverse proxy
3. HTTP Methods and Status codes
4. Inode, symlink vs hardlink

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺