Unix/Linux helping material

- **df** (disk filesystem): Display information about total space and available space.
  - -h => human readable, show info in GB
  - -a => displays all info along with dummy files as well
  - -k => show info in bytes(1024)
  - -m => display info in MB
  - -i => display info of inode(index of location)
  - -l => locally mounted file system
- **ps aux | grep nginx:** used to get information about the running process and the user details.
- **grep processor /proc/cpuinfo | wc -l:** used to get cpuinfo, i-e number of cpu
- **df -h --total -T / | grep -vE dev :** Used to monitor real time hard disk size
- **wget:** Utility used for download files from the web
  - -m => (mirroring) used for downloading all files of website
  - -i => read urls from local or external file
  - -b => download files in the background
  - -c => used to continue/resume interrupted download

  –limit-rate= => to limit the download speed. E.g –limit-rate= 128k
  - -r => full site and all its pages
  - -l => number of levels you wanna go

- **nslookup**(name server lookup): Used to get information from DNS server, also used to obtain ip address or domain name or any other DNS record.
  - -type => used to get info about specific type, e.g any,soa(start of authority), ns(for ns record), mx(mail exchange), a (for A record),
  - -query => usage same as above
  - -timeout => you can give time to respond to the server
  - -debug => used to enable debug mode
- **nsupdate:** used to submit a dynamic DNS update requests as per

- **chmod(**change mod): used to change the permission of file
  - **-R** => if we want to include files in subdirectories as well
  - **+rwx** => it will add permissions to file or directory
  - **-rwx =>** used to remove permission from file or directory
  - **+x =>** it will give executable permissions

u+x **=>** used to give executable permission to user

- **chown(**change ownership**):** used to change the ownership of file directory.

    -methods => chown user_name file_name

    chown user:group file_name (will change ownership to user and group)

    chown :group_name(will only group)

    -R => Used to change permissions recursively

    -h => Used to change permission for a symbolic link

    --reference => Used to change permission of file same as referenced file

- **dd**(Data Duplicator): used to copy files or convert the file. It can be used to get a specific/fixed amount of data, or backup. Can convert to/from ASCII, EBCDIC text encoding.

-if	=> input file or STDIN
-of	=> output file or STDOUT
-conv=noerror	=> continue if no error
sync	=> to copy each and everything

- **uname:** used to get info about OS and System Hardware.

-r	=> for kernel release
-m	=> machine hardware
-v	=> for OS version

- **ls:** Used to display list of files and directories

-l	=> display directory size, update time/date, permission etc
-a	=> used to display hidden files
-h	=> make human readable
-r	=> display files/directories in reverse order
-s	=> display size of file as well
-i	=> display inode number as well
-n	=> display user id along with group id

- **whoami:** used to get username of current logged-in user

- **ping:** used for testing, diagnosing, troubleshooting the network connectivity.

-c	=> send only specific number of packets, e.g c 10 and exit
-I	=> to specify the resource, from where you need to send req
-4	=> for ipv4
-6	=> for ipv6

- **pwd:** used to display path of working directory

-L	=> display symbolic link
-P	=> actual path

- **cd:** change directory to

../ ../	=> 2 level up
cd -	=> navigate to previous directory
cd ~	=> to home directory
cd ~username	=> switch to that user
‘ ’	=> if directory has spaces

- **touch**: used to create a file with no content

-a	=> used to change the access time
-c	=> used to check whether file is created or not
-m	=> change modification time only

- **mkdir:** make directories

-p	=> create parent directory if needed
-m	=> for mode/permission
mkdir -p test/{test1,test2}	=>{create a parent and then }

- **cat:** used to create single or multiple files, view content of file

-n	=> display number of lines
-e	=> used to show spaces in paragraph(on multiple lines)

- **apt-get**: package manager

-i	=> install
-d	=> download
-f	=> fix broken file/dependencies
-m	=> ignore/fix missing
-y	=> say yes to all

- **echo:** used to display text or string passed in argument

-e	=> enable interception, like you can remove spaces now

- **cp:** used for copying files.

-i	=> interactive i-e warn before overwriting
-b	=> used for backup
-f	=> force

-r	=> used for recursive copying directories

- **rm:** utility used to delete files and directories

-f	=> never prompt to the user, and ignore nonexistent files
-v	=> verbose, give info what is being removed
-d	=> used to delete empty directories
-r	=> used to delete directories recursively
-i	=> prompt before deletion of each file
-rf	=> used when file inside directory is protected from deletion

- **man:** used to display used manual

-f	=> give section in which command is present
-a	=> display all available intro
-k	=> used to search for command in given regular expression
-w	=> gives location where command is present

- **mv:** used for moving or renaming file or directory

-i	=> interactive, i-e it prompts before moving
-f	=> force, overwrite by force, deny permission
-m	=> prevent file from being overwritten
-b	=> it will backup and then move the file

- **sudo:** super user do, allow user to run a command by privileges of another user

-v	=> validate, used for giving validation for 5 min
-k	=> used to kill the given validation
-K	=> sure kill
-b	=> run sudo process in background
-p	=> used for custom prompt instead of pre-built prompt
-n	=> non-interactive, so that run sudo process in background, used where silently work to be done is required
-u	=> used to run a command other then root
-a	=> authentication type, used for adding new authentication type in etc/login.config
-	=> stop processing command line arguments

- **du**: dick usage

-h	=> human readable
-a	=> print all files including directories
-c	=> prints total size
-d	=> -d 2, depth levels, how deep in directory you need to go

-s => summary of files

- **date:** used for displaying date or update date
  - -time => for getting timestamp
  - -u => Display time is GMT/UTC(coordinated universal time)

- **id:** used to display to username and group name and ids of current/all users on server
  - -u => for user id when user name is given in front of -u
  - -g => for group id when group name is given
  - -G => all group to whom a user belongs
  - -r -g/-u=> give real id

- **passwd:** user to change the password
  - -d => delete password, make account password less
  - -e => expire, immediately expires the account password and ask user to generate new password
  - -i => inactive_day, which of the number of day, when password will expire
  - -l => used to lock the account, append with this !, and you can login with ssh.
  - -m => user can't change password for min days
  - -q => change password in quiet mode, i-e don't display msg
  - -S => show status of password in detail
  - -u => unlock password of an account
  - -w => display the message of warning for expiring password
  - -x => set the max days after that user is forced to change password

- **top:** dynamic real-time view of the running system
  - -u => display special user process
  - -b => Batch,send product to another task
  - -s => use top in secure mode

- **htop:** allow users to interactively see resource usage
  - -d => used to display delay between updates in tenth of second
  - -c => no color or non chromemode
  - -u => display processes of a given user
  - -p => used to display process ids
  - -s => sort by column

- **which:** locate executable file path in environment variable e.g cpp, python
    - -a => display all matching parameter of argument

- **dpkg:** manage package on debian based linux
    - -i => install package
    - -R => install packages recursively
    - -r => remove packages and config files
    - -l => list of all installed packages
    - -p => Display details of packages

- **kill:** kill process manually.
    - -l => display all available signals
    - -s => to show how to send signals to processes

- **pkill:** process kill, used to kill a process by its name

- **whereis:** used to give location of source-file of command
    - -m => show manual page location of command
    - -s => search for the sources

- **history:** used to display history of previous commands
    - -d => to delete history of specific event
    - -c => clean all history

- **tar:** tap archive, used to archive, extract archive files, compress/uncompress
    - -c => create archive
    - -x => extract archive
    - -f => create archive with given filename
    - -z => create tar archive using gzip
    - -t => display list of files in archived files
    - -j => filter tar archive using tbzip
- **sort:** used to sort a files, or record in a particular order
    - -o => output of one file to a new file
    - -r => sort in reverse order
    - -n => to sort a file numerically
    - -k => sort by a specific column
    - -c => checks whether the file is sorted or not, if not then sort
    - -u => sort and remove duplicate

- **head:** print first 10 lines of specified file
  - -n => number of lines
  - -c => number of bytes from specified fields
  - -q => show output from multiple files

- **gzip:** command used for compressing, and each file is compressed in single file
  - -f => skip permissions by force
  - -k => it compress file but keep original as well
  - -r => can compress files in folders i-e recursively
  - -v => display difference between compressed and uncompressed
  - -d => allow decompression of file

- **nohup:** no hangup, allow process to run even shell/terminal is closed
- **lsb_release:** Linux Standard Base, give info about your specific linux distribution
  - -a => give all information
  - -d => verbose description about release version

- **curl:** used to transfer data in different protocols
  - -o => save downloaded files on local machine with name provided in parameter
  - -O => save with same name, provided in url
  - -C => continue file which was stopped by some reason
  - --limit-rate => limit rate to a specific point
  - -u => download files from user_authenticated FTP server
  - -T => helps to upload file to FTP server
  - -x => let us to use proxy to access URL
  - -X => used for custom req to network. i-e curl -X POST url

- **grep:** global regular expression suggestion string grep command used for searching a file for particular file pattern of characters
  - -i => search and don't care about character sensitivity
  - -c => count number of files matches the pattern
  - -l => display the file names that matches the pattern
  - -w => only match the whole word. E.g unix is alive. unix here.
  - -o => display only matched pattern
  - -n => show number of line which hold pattern
  - -v => display line which are not matched with a pattern, i-e left
  - -B => display lines before match, B 4

-A => display lines after match, A 4
-C => display lines before and after match, C 4
-r => will search in subdirectories as well
-q => quiet, do not write anything on standard output

- **cron:** utility used for automating pre-scheduled tasks on predetermined time.
  * * * * => minutes(0-59), hours(0-23), day of month(1-31), day of week(0-6)https://www.tecmint.com/wp-content/uploads/2017/08/Run-PHP-Script-as-User-via-Cron.png
- **ifconfig:** interface configuration, used to assign ip addresses and netmasks to interface, can be used to configure kernel-resident network interface.
  -a => display all interfaces available
  -s => display short list instead of details
  interface up => to active driver for interface
  interface down => deactivate the driver for interface
  del => remove ipv6 address
  add => add ipv6 address
  curl ifconfig.me => to get ip exact
- **sed:** stream editor, work line by line and its non-interactive
  -e => to provide multiple commands on command line
  -f => get command from file
  -n => print (only) line if command is applied to that line
  -i => it will made change to the original source file
  Sample commands
  sed ‘s/Linux/Unix/’ filename
  sed ‘s/Linux/Unix/g’ filename
  sed ‘s/Linux/Unix/3g’ filename
  sed ‘3 s/Linux/Unix/’ filename
  sed -n ‘s/Linux/Unix/p’ filename
  sed ‘__d’ filename => {used for deletion}
  sed ‘/^$/d’ filename => {it will delete empty lines}
  sed G filename => {insert one blank space after each line}
  sed ‘G;G’ filename => {insert 2 blank lines after each line}
  sed "s/usman/${variable}/" testing.txt
- **set:** used to set or unset certain settings/flags in the shell environment. These flags and settings determine the behaviour of a defined script and help in executing a script without any issue.
  -x **=>** used to turn on debugging information
  +x **=>** used to turn off debugging information
  -C **=>** it will disable the default behaviour of script

-e **=>** to stop a script immediately when error occurs
+e **=>** will continue the script till end of file even after the error encountered

- **scp root@ip:/path/of/file /path/of_local_machine:** used for download and storing a remote file locally. E.g
  scp usman@192.168.56.102:/tmp/info.txt /root/
  -C **=>** faster the process of transferring file
  -p => provides modification time, access time and transfer rate
  -v **=>** used to print debug information about, as without flags scp copy files in background i-e it will only display info either process is successful or not
  -l => used to limit the bandwidth to use

- **ssh with home document =>** used for running commands remotely.
- **change ssh default port** **=>** nano **/etc/ssh/sshd_config**
  Then change the 22 to 2233 or any new port

- **xargs:** used to accept inputs from another command just like grep n perform operation. Just **grep "usman" fileName | xargs** will get all usman words and put in line without line break.
  echo "one two three" | xargs mkdir => will make directories
  | xargs rm => will perform operation same as above

- **tar -czf /destination/path /directory/to/be/backup =>** how to backup a folder
- **user = $(whoami) =>** here output of command whoami is directly assigned to user

  - **cut:** used for cutting specific sections of file/text. It works byte by byte, character by character, or by field.

    -b => cut by bytes, i-e 1,2 or 1-3c
    -d => set a delimiter e.g “ ” space
    -f => set field number, e.g -d “ ” -f 3
    -c => cut by character, e.g -c 2
  - **find:** used to find file, folder, creation date, permission, owner and modification date

    -type => used for specifying type, i-e d(directory) f(file)
    -name => search file by name, and even by extension(.txt)
    -i => used for quitting case sensitivity
    -empty => used to search for empty directories
    -exec => to execute another command in front of find command
    -perm => used for checking permissions
    -not => display other then specified type
    -o => or, i-e -name ‘.txt’ -o -name ‘.pdf’
    -user => find all files owned by the user specified
    -group => find all files owned by group
    -size => find files of specific file, i-e -size 2M
    -mtime=> find files modified days ago, i-e -mtime 8(modified 8 days
    find / -type f -name 'test.*' -exec rm -f {} \;
  - **tail**: tail command used to print last number of input data

    -n => print the specified number instead of default i-e 10
    -f => it will show not already built connections but newly edited as well. Can be used for logs
    -q => used for two lines files and merge them
    -c => specify the no of bytes
  - **awk:** is a programming language used for usually tabular data like csv, file of data. More general solution is **awk {‘script‘} fileName**

    awk ‘{print “$1”}’ filename => print first field in column
    NF(no of fields) => awk ‘{print NF}’ filename, i-e 4 or 2 etc
    => awk ‘{print $NF}’ filename, prints whole col.
    NR(no of row) => awk ‘{print NR}’ filename, print no of row
    -f => read from file where you have write the all pattern, i-e awk filename.awk fileName_whereApply_changes
    -F => is used as delimiter in awk

$NF => can be used for getting last entry/column
OFS => output field separator, allow you to format your output field
ORS => output record separator

> **=>** is used to redirect the output**(stdout)** of command to file
2> **=>** used to redirect redirect **stderr**
&> **=>** is used to redirect both **stderr** and **stdout**
find $1 -type f | wc -l => total number of files in directory
find $1 -type d | wc -l => total number of directories in directory
$# **=>** Gives number of arguments supplied e.g usman here => 2 arguments
$* **=>** prints all arguments

__Nano Text Editor__

Ctrl + _	=> enter line number where you want to jump
Ctrl + w	=> used for search
Ctrl +\	=> used for replacing the searched keyword
Ctrl + o	=> used to write on disk
Ctrl + w	=> where is, To find a word
Ctrl + J	=> used for justifying
Ctrl + \	=> search and replace
alt + R	=> move/forward line to the right
Ctrl + i	=> give tab space

To install specific version of psql on ubuntu 20/18

```
RUN sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg
main" > /etc/apt/sources.list.d/pgdg.list'
RUN wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key
add -
RUN apt-get update
RUN apt-get install -y postgresql-10
```

Docker

Docker is a containerization technology. **D**ocker is a set of PAAS(Platform as a service) products which use OS-level virtualization to deliver software in packages called containers.
Docker containers are lightweight standard unit software, which contain all its dependencies and enable applications to run quickly in one go.

Dockerfile

Dockerfile is a text file which contains all instructions and commands to build a docker image. To build an image we use **docker build** command so that several commands and instructions are in succession.

Docker-Compose

Docker-compose is a YAML file, used to write configurations to run multiple containers in one go. In Docker-Compose a single command is used to run multiple services, i-e to start multiple services and can run the whole app in one go. Most famous command to start and run services,
docker-compose up

Docker Commands

docker ps => show all docker running container instances
docker ps -a => will show all containers available on machine
docker pull => used to pull docker images
docker images => will show all images downloaded on machine
docker rm => is used for removing images
docker rm -f => used for removing running docker containers
docker rmi => used to remove docker image
docker restart => used to restart docker container
docker system prune => is used for removing stopped containers, all networks not used, all dangling images
docker image prune => used for removing dangling images
docker stop => used for stopping docker container
docker start => used to start a docker container
docker logs => to see logs of running container
docker logs -f => used to see logs continuously
docker volume create => used to create volume

docker volume ls	=> to see volume used by containers
docker volume inspect vol_Name	=> to see details of volume
docker ps --size	=> to check the size of docker containers
docker ps -aq	**=>** list all containers id
docker rm -v	=> used to remove container volume
e.g	=> docker run -v volName:/var/lib
-p	=> used for assigning port **-p 8080:80**
-d	=> usd to run a container in detached mode
-exec	=> used to attach with a running container
--rm	=> if you use this while run while running container then it will delete/remove container right after when it got executed
ctrl+p+q	=> used to detach from a running container cli
docker cp db.sql 40e7e1697192:/	to copy from local to docker container

Docker-Compose Commands

docker-compose up	=> used for starting services from scratch
docker-compose up -d	=> used for starting services in detached mode
docker-compose start	=> used for starting composer services(existing services)
docker-compose down	=> used to stop services and removing containers
docker-compose stop	=> used for stopping services without removing them(containers, volume, networks)
docker-compose ps	=> used to see running processes of docker compose
docker-compose config	=> used for checking the validity of docker-compose
docker-compose pause	=> pausing running services
docker-compose unpause	=> used for unpausing paused services
docker-compose up -d --scale serviceName=2,3,4	=> for scaling a service
docker-compose top	=> used for giving docker services pid, user, and time
docker-compose restart	=> used to restart services
build	=> build or rebuild services

bundle	=> generate docker bundle from docker compose
create	=> used to create services
event	=> receive real time events from docker containers
exec	=> execute a command in running container
images	=> give list of images

BASH Scripting

A Bash script is a file that contains a series of commands that can be executed when you will run bash script.
Bash script has extension **.sh** and to execute you must have to give permissions i-e
chmod +x fileName.sh

Bash script begins with following line
#!/bin/bash
To execute the bash script, ./fileName.sh

- **VARIABLES:**
  A variable is a character string which is assigned to a value and value can be anything like text, filename, number etc.
  e.g Name="Muhammad Usman"
  ```
      echo $Name
      echo “My name is $Name”
      echo “My name is ${Name}and can write here as well”
  ```
  $$ => used to represent the PID of the current shell

- **Input/output printing:**
  Input and output of data can be done using the **read** command. If you don't specify the variable name then it will by default be stored in $REPLY variable.
  read => can read input from the user
  -p => used to prompt the message before input
  -n => used for the number of character to be allowed for input
  -s => used for getting sensitive information

- **Arithmetic Operations:**
  ```
  num1=10
  num2=20
  echo $(( num1 + num2  ))
  echo $((  num1 * num2 ))
            Or
  echo $(expr $num1 / $num2 ))
  echo $(expr $num1 - $num2 ))
  echo $(expr $num1 % $num2 ))
  ```

Or

```
expr 10 + 20
num=`expr 10 + 20`
echo $num
```

if-else statements

```
read -p “Please enter your number: ” var
if ((   var < 100 ))
  then
    echo “value is less than hundred”
elif [ var -ge 100 ]
  then
    echo “value is greater or equal to hundred”
else
    echo “something strange happened”
fi
```

- **Pipes**
  Pipe “|” is file that allows you to connect the output of one process to another process.
  history | grep “cat”
  In this scenario we are getting output of history from the left side and passing/catching the output of that file to the next command.

- **String Manipulation:**
  In bash we can get length of string, can update string and we can update the string as well.
  To get variable length in bash we can use =>**echo ${#var}**
  To Update a string in file => echo ${variableName/string/toNewString }
  To replace multiple characters => echo ${variable//stringtobeRep/TobeRepWith}
  Replacing starting and ending of string => ${variable/%.*/TobeReplaceWith}
  Replace start of String => ${variable/#\/wordTobeReplace/tobeReplaceWith}

- **Exit status codes**
  A code that is returned to the parent process by the executable. According to POSIX standard 0 is for success and 1-255 for anything else. Mostly 1 is used for general error.
  To get exit code => **echo $?**

- **Reading from files**
  We can read files using while loop in different ways. One of the methods is using while loop.

```
while read var
do
 echo $var
done < fileName.sh
```

 Or

```
cat filename | while read var
do
echo $var
done < filename.sh
```

- **Writing to file**

  Saving output of one file into another file
  To edit in a file on go, **cat > fileName.txt**

  Appending output in a file
  To append text in same file, **cat >> fileName.txt**

- **Comments**

  Single line comment: **=>** # line is commented

  Multi-line comment
  : ‘
  Multiline commenting goes here
  And ends here **‘**

- **Functions:**

  Functions are used when you have to reuse any code.
  functionName() {

# body of function
echo “my function out”
}
# to call the function
functionName
We can also return status code that either our function is successfully executed or not.
statusCode () {
echo “My name is usman”
return 1
}
#calling function
statusCode
echo “status code of function is $?”

- **Arrays:**
  Array is a Data Structure which contains a collection of elements, identified by the one index key.
  To declare array in Bash
  array_name=(usman,nouman,adnan)
  echo ${array_name[0]}
  To print all elements => echo ${array_name[*]})
  => echo ${array_name[@]})
  To delete an element of array => unset array_name[2]

- ## SSH
  SSH (secure shell) is an encryption method used for secure communication over an unsecure network between a client and server.
  Ssh uses Asymmetric Encryption for encrypting and decrypting data. In this way we send our public key to the server and we save that in the server, the server encrypts messages with our public key and then sends via ssh tunnel to client(my machine) and that is decrypted using client private key, then we acknowledge the server and server verifies us to communicate over network.

- **SSH logs and Analyzing SSH logs**
  SSH logs are stored at **/var/log/auth.logs** by default.
  We can analyze ssh log by going in this directory and using different command line utilities like cut, awk, head, tail, sed and find etc in more better way and can print easily in human readable form.

IP Tables

- ## iptables

  IPtables(modern variant netfilter) is an extremely flexible command line utility that uses the policy chain that allows or blocks the traffic. When a connection tries to connect/establish with a system, iptables looks for rules in the list to match with it. If it doesn't find any rule then it will redirect to default actions to be taken.
  It is actually frontend to kernel-level netfilters hook that can manipulate Linux network stack. It filters each and every packet which is crossing over the network through a set of rules and decides what to do.
- **Actions**

  When a defined pattern matches, an action takes place called Target. A target can be anything like a final policy decision so that either keep the connection or drop the connection, or move to a different chain for processing, or simply log the encounter.
  Accept, Reject, Drop, Queue, log.

- ## Tables in iptables and their purpose and working

  Filter table: Used for packet filtering. It is used to make a decision whether a packet should go to its intended location or not.

  ```
  iptables -t filter -L
  ```

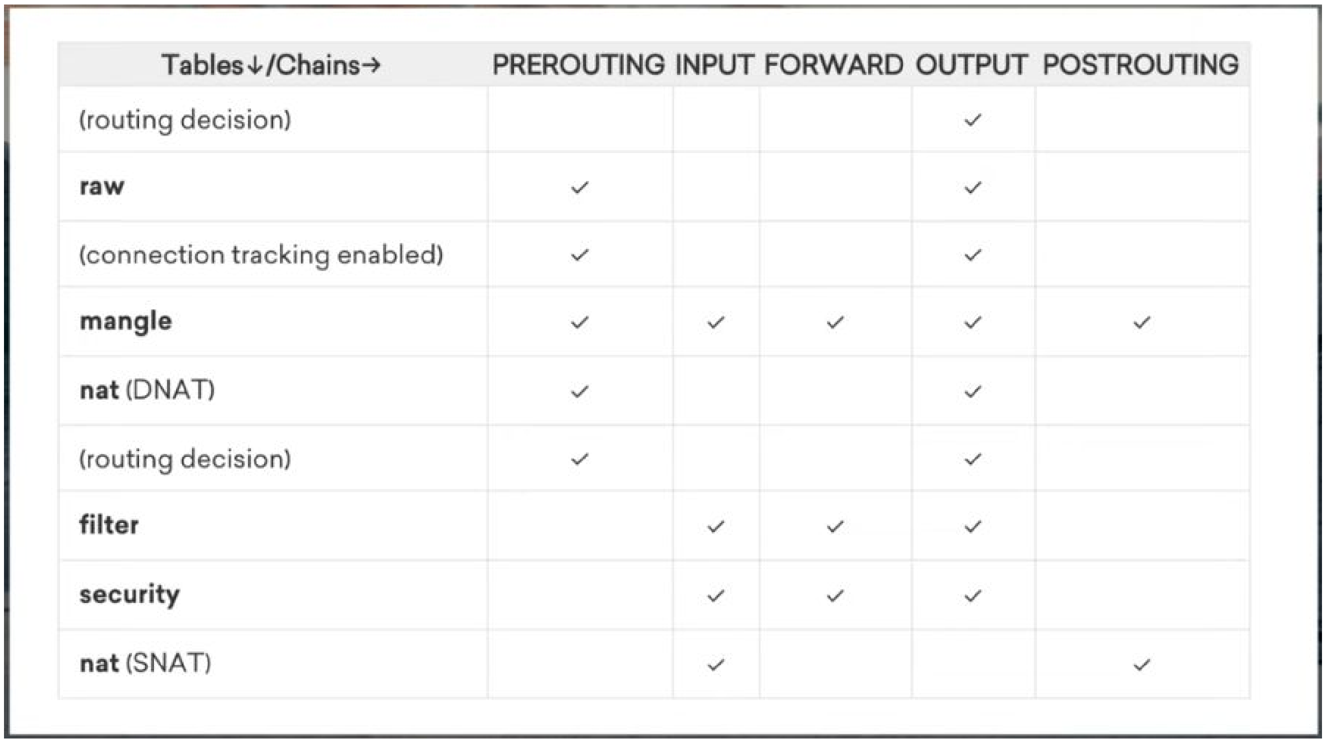
  Nat table: Nat table is used for address translation rules. It is used to determine when a packet enters in a network stack, nat rules will determine weather and how to modify the packet source and destination addresses so that to impact the packets and any response is routed. DNAT(destination nat) and SNAT(source nat)

  ```
  iptables -t nat -L
  ```

  Mangle table: The mangle table is used to alter the ip headers of packets in various ways. e.g you can adjust the TTL of any packet

  ```
  iptable -t mangle -L
  ```

  Security Table: Security table is used to set the internal SELinux security mark on the packet. It is used to enhance the SELinux and other systems that can intercept SELinux security context handle packets.
  Raw Table: Packets in the network come in a sequence instead of random stream. So that packets are evaluated according to previous packets in the stack. Raw tables are used for marking packets in order.



Tables↓/Chains→	PREROUTING	INPUT	FORWARD	OUTPUT	POSTROUTING
(routing decision)				✓	
raw	✓			✓	
(connection tracking enabled)	✓			✓	
mangle	✓	✓	✓	✓	✓
nat (DNAT)	✓			✓	
(routing decision)	✓			✓	
filter		✓	✓	✓	
security		✓	✓	✓	
nat (SNAT)		✓			✓

- **Chains**

  The rules are organized into groups called chains. Chain is set of rules against which packet is supposed to be checked, if packet matches one of the rules specified then it will take action accordingly and do not go to the next rule for checking.
  Chains can be created by the user and by default there are three chains:
  1) INPUT
  2) OUTPUT
  3) FORWARD
  - **Input:** This chain will handle the incoming traffic to your server. E.g if a user tries to get logged in your server, it will look for rules in iptable so that this ip and port is allowed by the rule or not.
  - **Forward**: This chain is used for those connections or traffic which is not for your local server but for other servers. In this way we can configure our server for other traffic.

- **Output:** This chain rule is used for configuring rules for traffic created by your server. For example if you want to ping from your server, i-e ping google.com, you have to check for those rules that what to do over ping command from iptables.
- **Pre-routing:** It is used to add rules which define actions that need to be taken before a routing decision is made by the kernel. Modify Packets as they arrive.
- **Post-routing:** It is used to add rules which will define actions that need to be taken after a routing decision taken by the kernel. Modify Packets as they are leaving.

- ## Useful commands used in iptables

To see all iptables rules => **iptables -L**

-A	=> used to append rule in the chain
-L	=> List of all current list of rules
-p	=> connection protocol used
--dport	=> used for destination port, it could be single or multiple ports(22-27)
-n	=> do not resolve the dns, and quickly show output.
-j	=> jump to specified target, i-e -j ACCEPT/REJECT/DROP/LOG
-v	=> verbose, used to display more information in output.
-s	=> source address(source) specification
-S	**=>** it will list rules by specification
-d	=> destination, address(destination) specification
-I	=> insert rule, take two args, -I INPUT 5(insert rule in input on 5 index)
-t	**=>** used for checking tables, like iptable -t mangle => it will give me a list of all mangle table rules.
-i	=>interface, only matches pattern coming on specific interface e.g eth, lo

--line-numbers => to get line number as well

-m -comment --comment “write here comment” => to write comment

Allow all incoming traffic on specific port
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

Delete rule in iptables
iptables -D INPUT rule_no
iptables -D INPUT 1 // it will remove first entry

Drop all the incoming traffic
iptables -P INPUT -j DROP

Accept all incoming traffic
iptables -P INPUT -j ACCEPT

DROP all outgoing traffic
iptables -P OUTPUT -j DROP

BLOCK outgoing traffic to specific ip
iptable -A OUTPUT -d 1.1.2.2 -j DROP

CUSTOM chain as action/target
iptables -A INPUT -s 1.1.1.1 -j custom_chain

- **Allow a specific ip to all tcp traffic**
  iptables -A INPUT -p tcp -s/d 1.1.1.1 -j ACCEPT

- **Block a specific port for a specific network**
  iptables -A INPUT -p tcp -s 1.1.1.0/24 --dport 22 -j REJECT

- **Allow a specific network for specific ports**
  iptables -A INPUT -p tcp -s 1.1.1.0/24 --dport 22-25 -j ACCEPT

- **Drop incoming to port 80 if more then 5 requests hitted per min**
  iptables -A INPUT -p tcp --dport 80 -i eth0 -m state --state NEW -m recent --update --seconds 60 --hitcount 5 -j DROP

- **Create/Delete Custom chain rule**

  <u>To add custom chain rule</u>
  iptables -N Custom_chain_rule_name

e.g => iptables -N my-new-rule
iptables -A my-new-rule -s 1.1.1.1 -j ACCEPT

<u>**To delete custom chain rule**</u>

iptables -X => it will delete all chains(non-default)

e.g iptables -X

iptables -F => it will flush all chains

e.g iptables -F INPUT // it will flush all input chain rules
iptables -t nat -F

- **To delete rule securely**
  To delete rules securely you must need to backup the all the rules and then delete them, for back the following command is used
  <u>iptables-save > fileName</u>
  Then we will delete all rules using following command
  iptables -X
  To **restore** the rules
  <u>iptables-restore < /path/fileName</u>

- **iptable persistent**
  By default iptables are not persistent so that they will be reset when system reboots, to make them persistent install a package named **iptable-persistent**
  apt-get install iptable-persistent
  By default all the rule will be stored in folders ipv4/ipv6 accordingly
  /etc/iptables/rules.v4
  /etc/iptables/rules.v6
  To backup
  iptables-save > /etc/iptables/rules.v4
  ip6tables-save > /etc/iptables/rules.v6
  To remove these, you just simply remove the required particular file.

- **Port Forwarding:**
  In Linux kernel port forwarding is achieved by packet filter rules in iptables. It is basically a NAT gateway changing the destination address or port to reach a host within a public or private network. It can be used to connect a remote system within a private network.
  sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 1.1.1.1:8080

- **log packets**
  First of all allow insert rule, iptables -I INPUT -p tcp --dport http -j ACCEPT
  Then insert rule for log **iptables -I INPUT 1 -p tcp --dport http -j LOG**

- **analyze iptables logs**
  To analyze all dropped(incoming) INPUT Packets
  To insert new rule => iptables -N LOGGING
  Append that rule => iptables -A INPUT -j LOGGING
  Log incoming packets to syslogs
  iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables-Dropped" --log-level 4
  And for really dropping logs => iptables -A LOGGING -j DROP
  Flag used
  -m limit => used for limiting any matching module
  --limit 2/min => maximum average rate for logging, it could be2/sec, 2/min 2/hour , 2/day. Used to avoid messages of same logs cluttering
  -j => here -j indicate the target of packet is log
  --log-prefix => write anything that will be displayed/appended along with log message in /var/log/message
  --log-level => it is standard syslog level, 4 is warning and we can use from 0 to 7, 0 is for emergency and 7 for debug.

  Log all dropped outgoing packets
  To insert new rule => **iptables -N LOGGING**
  Append that rule => **iptables -A OUTPUT -j LOGGING**
  Log incoming packets to syslogs
  iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "IPTables-Dropped" --log-level 4
  And for really dropping logs => **iptables -A LOGGING -j DROP**

  To Drop both incoming and outgoing packets
  ```
   iptables -N LOGGING
   iptables -A INPUT -j LOGGING
   iptables -A OUTPUT -j LOGGING
   iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix  "IPTables-Dropped"
  --log-level 4
   iptables -A LOGGING -j DROP
  ```

- **States in iptables**
  - **New:**
    When a new connection packet arrives and it is not associated with an existing connection, and also the packet is not invalid, then a new connection will be added to the system with New label.

  - **ESTABLISHED**
    When a connection is changed from New to Established state when it recieves response from the opposite side.
  - **RELATED**
    A packet that is not part of an existing connection but part of an existing connection then that is labeled as RELATED.
  - **INVALID**
    A packet that is not related with existing and not appropriate for a new connection then it is labeled as Invalid.
  - **UNTRACKED**
    Packets that have been targeted to raw table chains to bypass tracking.
  - **SNAT**
    A virtual state when a source address has been altered with NAT operations
  - **DNAT**
    A virtual state when a destination address has been altered with NAT operations.
- **Multiport:**
  Multiport match module is used to match the set of source or destination ports and upto 15 ports can be specified.
  Command
  Following command is for destination ports
  iptables -A INPUT -p tcp --match multiport --dports 80,22,33 -j ACCEPT
  Following command is for source ports
  iptables -A INPUT -p tcp --match multiport --sports 80,22 -j ACCEPT
  To specify the range
  iptables -A INPUT -p tcp --match multiport --dports 80:90 -j ACCEPT

- **Ping flood attack:**
  Ping flood attack, also known as ICMP flood, is a common DOS attack in which a user takes down the victim's computer by overwhelming it with ICMP echo requests, also known as ping. The attack involves flooding victims' networks with request packets. Other tools like hping and scapy are also used for this attacking purpose.
  working:
  The attacker sends many ICMP echo request packets to the targeted server using multiple devices.
  The targeted server then sends an ICMP echo reply packet to each requesting device's IP address as a response.

How to avoid
Disabling a ping flood is most easily accomplished by disabling the ICMP functionality of the targeted router, computer or other device. As the filter table does not support pre-routing so we will use the mangle table and prerouting chain.
Drop ICMP

```
iptables -t mangle -A PREROUTING -p icmp -j DROP
```

- **Rules to avoid ssh brute force attack**
  To block a SSH brute force attack, we need to slow down the flow of requests. We can do this by **rate-limiting requests** to SSH with iptables. We will create smaller pipes for the ssh session and this point can make ssh attack ineffective.
  - One of the best ways is to allow only your trusted list of ip so that other traffic does not come to your server.
  - If we change the default port(22) to another port for ssh then we can protect our system in a better way from automated attacks.
  - Hash-limits: iptables -I INPUT -m hashlimit -p tcp --dport ssh --hashlimit 1/min
  - The rule for blocking requests which tried to attempt 3 times per minute. This rule will be applicable for only new connections not established connections.

```
iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent --set
iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent  --update
--seconds 60 --hitcount 4 -j DROP
```

Rules to fight against DDOS + DOS
- First of all we will apply filtration on the prerouting chain so that we can filter bad packets from the start, and for this purpose we will use a mangle table because the filter table does not support the pre-routing.
- Optimise kernel settings
- **Block invalid packets**
  ```
  iptables -t mangle -A PREROUTING -m conntrack --ctstate INVALID -j DROP
  ```
- **Block New Packets That Are Not SYN**
  ```
  iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate NEW
  -j DROP
  ```
- **Block Uncommon MSS Values**
  ```
  iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW
  -m tcpmss ! --mss 536:65535 -j DROP
  ```
- **Block Packets With Bogus TCP Flags**
- **Block Packets From Private Subnets (Spoofing)**
- **Useful link where all rules are listed**
  https://javapipe.com/blog/iptables-ddos-protection/

UDP invalid packet attack
These attacks involve the transmission of fraudulent UDP or ICMP packets that are larger than the network's MTU, (usually ~1500 bytes). As these packets are fake, and are unable to be reassembled, the target server's resources are quickly consumed, resulting in **server unavailability**.
Rule
```
iptables -t mangle -A PREROUTING -p udp -m conntrack --ctstate INVALID -j DROP
```

Nginx

Nginx:

Nginx is an open-source web server, which can be used for reverse-proxy, chaining and load balancing.
It has HTTP server capabilities, and designed for maximum capabilities and stability. It can function as a proxy-server for email.
Nginx uses master slave architecture by supporting event driven, asynchronous and non-blocking model.

- **Installing nginx and php in linux:**

```
sudo apt-get update
sudo apt-get install nginx
```

Step 2 (optional): Install Nginx from Official Repository

Add Security Key In a terminal window, enter the following:

```
sudo wget https://nginx.org/keys/nginx_signing.key
sudo apt-key add nginx_signing.key
```

Open sources.list File for Editing In the terminal, enter the following:

```
sudo vi /etc/apt/sources.list
```

Add Nginx Sources to Repository List
Enter the following lines in the /etc/apt/sources.list file you just opened:

```
deb https://nginx.org/packages/mainline/debian/ <CODENAME> nginx
deb-src https://nginx.org/packages/mainline/debian/ <CODENAME> nginx
```

Install Latest Release of Nginx
To install the latest release of Nginx, use the commands:

```
sudo apt-get remove nginx-common
sudo apt-get update
sudo apt-get install nginx
```

Step 3: Start Nginx and Configure to Launch on Reboot To start Nginx:

```
sudo systemctl start nginx
```

To enable Nginx:
sudo systemctl enable nginx

To check Nginx is running:
sudo systemctl status nginx

To install php
sudo apt-get install -y php7.4-cli php7.4-curl php7.4-mysql php7.4-fpm php7.4-gd php7.4-xml php7.4-mbstring php7.4-zip php7.4-soap php7.4-dev

- **Difference between nginx and apache**

Nginx	Apache
Nginx runs on all unix systems and has limited support for the windows.	Apache runs on all unix systems as well as has complete support for the windows as well
Nginx is designed to be web server and proxy browser as well	Apache is designed to be a web server
Nginx does not process dynamic content natively	Apache process dynamic content within web server itself
Nginx uses event driven approach to serve client requests	Apache uses multithreading approach to process clients requests
Single thread can handle multiple connections.	Single thread can only one process a single connection.
Nginx can simultaneously run thousands of connections of static content two times faster than Apache and uses little less memory.	The performance of Apache for static content is lower than Nginx.

- **Caching in Nginx:**
When caching is enabled, NGINX Plus saves responses in a disk cache and uses them to respond to clients without having to proxy requests for the same content every time.
To enable caching, include the proxy_cache_path directive in the top‑level http {} context. keys_zone parameter defines the name and size of the shared memory zone that is used to store metadata about cached items

```
http {
   ...
   proxy_cache_path /data/nginx/cache keys_zone=one:10m;
}
```

Then include the proxy_cache directive in the context

```
http {
   …
 proxy_cache_path /data/nginx/cache keys_zone=one:10m;
server {
        proxy_cache mycache;
        location / {
            proxy_pass http://localhost:8000;
        }
    }
}
```

Allowing/Dropping Specific host:

```
stream {
   #...
      server {
                  listen 80;
                  allow 0.0.0.0;
                  deny 1.1.1.1;
            }
}
```

Configuring reverse proxy

```
            sudo nano custom_server.conf
            In the file enter the following
            server {

            listen 80;

            location / {

            proxy_pass http://localhost;

            }
}
```

The proxy_pass command directs all traffic on port 80 to http://localhost.

Link to the activate configurations
```
ln -s /etc/nginx/sites-available/custom_server.conf
/etc/nginx/sites-enabled/custom_server.conf
```

Test the configurations
```
nginx -t
```

Restart server
```
systemctl restart nginx
```

- **For static website**

```
server {
   listen 443 ssl;
   include snippets/self-signed.conf;
   include snippets/ssl-params.conf;

        index index.html;
        root /var/www/site1;

        server_name a.example.com;
        location / {
                try_files $uri $uri/ /index.html;
         }
}

server {
        listen 80;
        index index.html;
        server_name a.example.com;
        return 302 https://$server_name$request_uri;
}
```

- **Php configuration**

```
server {
        listen 443 ssl;
   include snippets/self-signed.conf;
   include snippets/ssl-params.conf;

        index index.php;
        root /var/www/php;

        server_name b.example.com;
   location ~ \.php$ {
        include snippets/fastcgi-php.conf;
   fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
```

```
    }
}

server {
        listen 80;
        index index.php;
        server_name b.example.com;
        return 302 https://$server_name$request_uri;
}
```

- **laravel configuration**

```
server {
    listen 443 ssl;
    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;

    index index.html index.htm index.php;
    root /var/www/demo/public;

        server_name c.example.com;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    error_page 404 /index.php;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
        deny all;
    }
}

server {
        listen 80;
        index index.html;
        server_name c.example.com;
        return 302 https://$server_name$request_uri;
}
```

- **wordpress configuration**

```
server {
      listen 443 ssl;
      include snippets/self-signed.conf;
      include snippets/ssl-params.conf;

      root /var/www/wordpress/wordpress;
      index  index.php index.html index.htm;
      server_name d.example.com;

      client_max_body_size 100M;
      location / {
            try_files $uri $uri/ /index.php?$args;
      }
      location ~ \.php$ {
            include snippets/fastcgi-php.conf;
            fastcgi_pass unix:/run/php/php7.4-fpm.sock;
            fastcgi_param   SCRIPT_FILENAME $document_root$fastcgi_script_name;
      }
}
server {
        listen 80;
        index index.php index.html index.htm;
        server_name d.example.com;
        return 302 https://$server_name$request_uri;
}
```

- **Creating self signed key and signature with openssh**

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

Creating strong Diffie-Hellman group

```
sudo openssl dhparam -out /etc/nginx/dhparam.pem 4096
```

Creating a Configuration Snippet Pointing to the SSL Key and Certificate

```
sudo nano /etc/nginx/snippets/self-signed.conf
```

Paste the following lines into that:

```
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
```

Creating a Configuration Snippet with Strong Encryption Settings

```
sudo nano /etc/nginx/snippets/ssl-params.conf
```

Paste the following

```
ssl_protocols TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_dhparam /etc/nginx/dhparam.pem;
ssl_ciphers
ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-
AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA
384;
ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0
ssl_session_timeout  10m;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off; # Requires nginx >= 1.5.9
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
# Disable strict transport security for now. You can uncomment the following
# line if you understand the implications.
# add_header Strict-Transport-Security "max-age=63072000; includeSubDomains;
preload";
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
```

Your new configuration file should look like this

```
server {
   listen 443 ssl;
   include snippets/self-signed.conf;
   include snippets/ssl-params.conf;

   server_name example.com www.example.com;

   root /var/www/example.com/html;
   index index.html index.htm index.nginx-debian.html;
```

```
    . . .
}
```

Adjust firewall rules

nginx -t

sudo systemctl restart nginx

- **Rule for redirecting traffic from http to https**

  return 302 https://$server_name$request_uri;
  Changing to permanent redirect
  return 301 https://$server_name$request_uri;

- **Cloudflare:**
  Cloudflare is a web-infrastructure and website security company which provides Content delivery network security, DDoS mitigation, internet security and distributed domain name server services.

  Nginx configuration for cloudflare:
  The Cloudflare Origin CA lets you generate a free TLS certificate signed by Cloudflare to install on your Nginx server.
  To generate a certificate for CA, navigate to the Crypto part in dashboard, and click on create certificate, and click on next and do not change the default settings.
  First, copy the contents of the Origin Certificate displayed in the dialog box in your browser.Then, on your server, open /etc/ssl/certs/cert.pem for editing:

  ```
  sudo nano /etc/ssl/certs/cert.pem
  ```

  Paste the content of the file in this.

  Installing the Origin CA certificate in Nginx:

  Now you have to update the Nginx configuration for your site to use the origin certificate and private key to secure the connection between Cloudflare's servers and your server.
  Modifying the configurations

  example.com

  ```
  server {
  ```

```
    # SSL configuration

    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ssl        on;
    ssl_certificate         /etc/ssl/certs/cert.pem;
    ssl_certificate_key     /etc/ssl/private/key.pem;

    server_name example.com www.example.com;

    root /var/www/example.com/html;
    index index.html index.htm index.nginx-debian.html;



    location / {
            try_files $uri $uri/ =404;
    }
}
```

Setting up Authenticated Origin pulls:

sudo nano /etc/ssl/certs/cloudflare.crt
Go in the above directory and paste the following certificate into that.

```
-----BEGIN CERTIFICATE-----
MIIGBjCCA/CgAwIBAgIIV5G6lVbCLmEwCwYJKoZIhvcNAQENMIGQMQswCQYDVQQG
EwJVUzEZMBcGA1UEChMQQ2xvdWRGbGFyZSwgSW5jLjEUMBIGA1UECxMLT3JpZ2lu
IFB1bGwxFjAUBgNVBAcTDVNhbiBGcmFuY2lzY28xEzARBgNVBAgTCkNhbGlmb3Ju
aWExIzAhBgNVBAMTGm9yaWdpbi1wdWxsLmNsb3VkZmxhcmUubmV0MB4XDTE1MDEx
MzAyNDc1M1oXDTIwMDExMjAyNTI1M1owgZAxCzAJBgNVBAYTAlVTMRkwFwYDVQQK
ExBDbG91ZEZsYXJlLCBJbmMuMRQwEgYDVQQLEwtPcmlnaW4gUHVsbDEWMBQGA1UE
BxMNU2FuIEZyYW5jaXNjbzETMBEGA1UECBMKQ2FsaWZvcm5pYTEjMCEGA1UEAxMa
b3JpZ2luLXB1bGwuY2xvdWRmbGFyZS5uZXQwggIiMA0GCSqGSIb3DQEBAQUAA4IC
DwAwggIKAoICAQDdsts6I2H5dGyn4adACQRXlfo0KmwsN7B5rxD8C5qgy6spyONr
WV0ecvdeGQfWa8Gy/yuTuOnsXfy7oyZ1dm93c3Mea7YkM7KNMc5Y6m520E9tHooc
f1qxeDpGSsnWc7HWibFgD7qZQx+T+yfNqt63vPI0HYBOYao6hWd3JQhu5caAcIS2
ms5tzSSZVH83ZPe6Lkb5xRgLl3eXEFcfI2DjnlOtLFqpjHuEB3Tr6agfdWyaGEEi
lRY1IB3k6TfLTaSiX2/SyJ96bp92wvTSjR7USjDV9ypf7AD6u6vwJZ3bwNisNw5L
ptph0FBnc1R6nDoHmvQRoyytoe0rl/d801i9Nru/fXa+l5K2nf1koR3IX440Z2i9
+Z4iVA69NmCbT4MVjm7K3zlOtwfI7i1KYVv+ATo4ycgBuZfY9f/2lBhIv7BHuZal
b9D+/EK8aMUfjDF4icEGm+RQfExv2nOpkR4BfQppF/dLmkYfjgtO1403X0ihkT6T
PYQdmYS6Jf53/KpqC3aA+R7zg2birtvprinlR14MNvwOsDOzsK4p8WYsgZOR4Qr2
gAx+z2aVOs/87+TVOR0r14irQsxbg7uP2X4t+EXx13glHxwG+CnzUVycDLMVGvuG
aUgF9hukZxlOZnrl6VOf1fg0Caf3uvV8smOkVw6DMsGhBZSJVwao0UQNqQIDAQAB
```

```
o2YwZDAOBgNVHQ8BAf8EBAMCAAYwEgYDVR0TAQH/BAgwBgEB/wIBAjAdBgNVHQ4E
FgQUQ1lLK2mLgOERM2pXzVc42p59xeswHwYDVR0jBBgwFoAUQ1lLK2mLgOERM2pX
zVc42p59xeswCwYJKoZIhvcNAQENA4ICAQDKDQM1qPRVP/4Gltz0D6OU6xezFBKr
LWtDoA1qW2F7pkiYawCP9MrDPDJsHy7dx+xw3bBZxOsK5PA/T7p1dqpEl6i8F692
g//EuYOifLYw3ySPe3LRNhvPl/1f6Sn862VhPvLa8aQAAwR9e/CZvlY3fj+6G5ik
3it7fikmKUsVnugNOkjmwI3hZqXfJNc7AtHDFw0mEOV0dSeAPTo95N9cxBbm9PKv
qAEmTEXp2trQ/RjJ/AomJyfA1BQjsD0j++DI3a9/BbDwWmr1lJciKxiNKaa0BRLB
dKMrYQD+PkPNCgEuojT+paLKRrMyFUzHSG1doYm46NE9/WARTh3sFUp1B7HZSBqA
kHleoB/vQ/mDuW9C3/8Jk2uRUdZxR+LoNZItuOjU8oTy6zpN1+GgSj7bHjiy9rfA
F+ehdrz+IOh80WIiqs763PGoaYUyzxLvVowLWNoxVVoc9G+PqFKqD988XlipHVB6
Bz+1CD4D/bWrs3cC9+kk/jFmrrAymZlkFX8tDb5aXASSLJjUjcptci9SKqtI2h0J
wUGkD7+bQAr+7vr8/R+CBmNMe7csE8NeEX6lVMF7Dh0a1YKQa6hUN18bBuYgTMuT
QzMmZpRpIBB321ZBlcnlxiTJvWxvbCPHKHj20VwwAz7LONF59s84ZsOqfoBv8gKM
s0s5dsq5zpLeaw==
-----END CERTIFICATE-----
```

Now update your Nginx configuration to use TLS Authenticated Origin Pulls. Open the configur file for your domain:

```
server {
   # SSL configuration
   listen 443 ssl http2;
   ssl         on;
   ssl_certificate         /etc/ssl/certs/cert.pem;
   ssl_certificate_key     /etc/ssl/private/key.pem;
   ssl_client_certificate /etc/ssl/certs/cloudflare.crt;
   ssl_verify_client on;
……..

}
```

Checking for configuration syntax

```
nginx -t
```

Restart your nginx service

```
systemctl restart nginx
```

- **Nginx logs**

  Access log:

  The NGINX logs the activities of all the visitors to your site in the access logs. Here you can find which files are accessed, how NGINX responded to a request, what browser a client is using, IP address of clients and more. These logs in http context are by default enabled.
  To add in the file syntax is

  ```
  access_log  /var/log/nginx/access.log;
  ```

  Error log:
  If NGINX faces any glitches then it will record the event to the error log.
  To add error log in the conf file, syntax is

  ```
  error_log  /var/log/nginx/error_log  crit;
  ```

  There are other messages available in nginx as well:
  emerg: emergency message when your system is unstable
  alert: alert message of serious issue
  error: an error has occurred
  Warn: a warning message, look into that
  notice: simple log message that can be ignored
  info: an info message you want know
  debug: info that is required to pinpoint the error.

- **Nginx multi threading:**
  Nginx is a multi-threaded application which means that it does not start a single thread for each connection. Take a scenario, The real data received from a network card to a buffer is done in the kernel when the network card issues an interrupt. Then nginx gets a request in a buffer and processes it. It has no meaning to start processing another request till the current request processing is done or till the current request processing requires an action that might block (for example disk read).
- **Load balancer**
  Nginx can work as load balancer, and there are three types of load balancers, Simple(round-robin) method, weighted, least-connected.
- **listen on specific ip and default direction to a specific api/webpage**

  ```
  server{
          listen 192.1.1.1:80; # it will listen on specific ip
          # if you want to make a request to a default ip/api or website, make that default
          i-e
          listen 80 default_server      #will make default requests to this.
  }
  ```

https://serverfault.com/questions/382914/nginx-rewrite-with-php-fpm