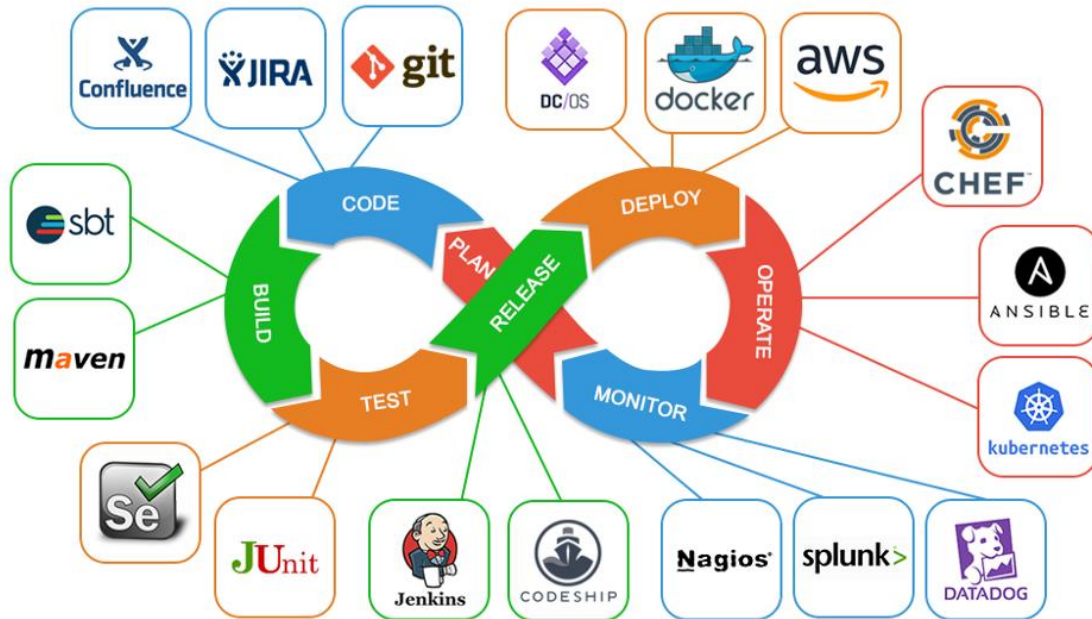
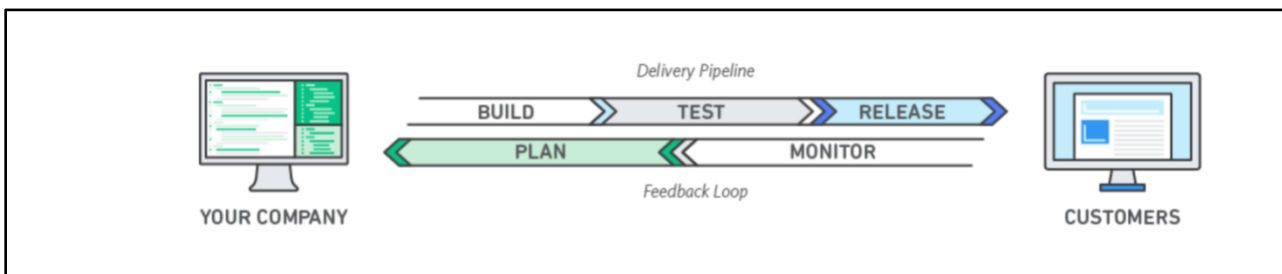


# DevOps



<https://aws.amazon.com/devops/what-is-devops/>

**DevOps:** DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



## How DevOps Works

Under a DevOps model, development and operations teams are no longer "siloesd." Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

## Benefits of DevOps

- 1) Speed
- 2) Rapid Delivery
- 3) Reliability
- 4) Scale
- 5) Improved Collaboration
- 6) Security

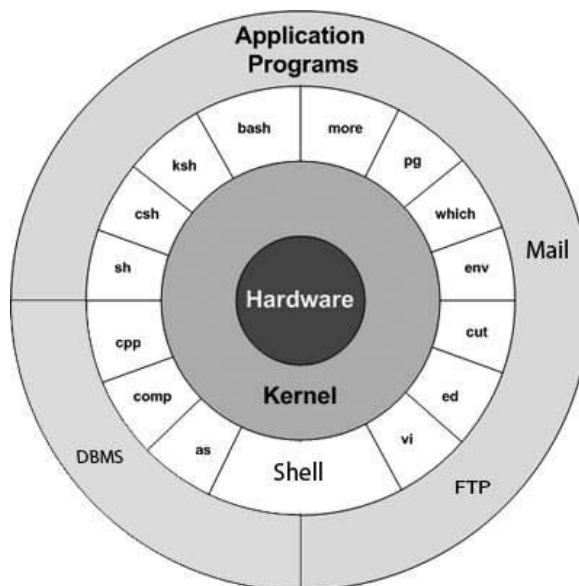
## Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

## Amazon Machine Image

An **Amazon Machine Image (AMI)** is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud ("EC2"). It serves as the basic unit of deployment for services delivered using EC2.

## Unix Architecture:



The main concept that unites all the versions of Unix is the following four basics –

- **Kernel** – The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.
- **Shell** – The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.
- **Commands and Utilities** – There are various commands and utilities which you can make use of in your day to day activities. **cp**, **mv**, **cat** and **grep**, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3<sup>rd</sup> party software. All the commands come along with various options.

## Different types of Linux

Debian, Gentoo, Ubuntu, Linux Mint, Red Hat Enterprise Linux, CentOS, Fedora and Kali Linux.

# COMMANDS

**Clear** : To clear the screen

**pwd** : present working directory

**mkdir** : command is used to create a directory/folder

Example: **mkdir directoryname**

**mkdir -p** : command is used to create a folder/directory if it doesn't exist

**mkdir -p /home/ubuntu/directory name** – used to create directory in specified path

**mkdir dir/dir1/dir2/dir3**

**cd** : command is used to change directory

Example: **cd folder name**

**"cd .."** – command is used to come out of the directory/ back to previous folder

**touch** - command is used to create a file

Example: **touch filename**

## To create multiple files/directories

Folder: **mkdir foldername1 foldername2 ....**

File: **touch filename1 filename2 ....**

**ls** - command is used to check directory and file/ used to display in long list

if it starts with "d" - directory

if it starts with "-" – file

if it starts with "l" link

**vi** - editor which is used to edit the file

Example: **vi filename**

**(esc)l** – to insert character – contents can be added only in the insert mode

**(escape):set nu** - used to set the line numbers in a side

**(escape)/patternname** - search pattern, press n to move the cursor to next pattern

**(esc) dd** – delete the line in file

**(esc)u** – to undo deleted line in file

**(esc):3s/pattern/newpattern/g** – to replace the pattern – it will replace all the same pattern in 3<sup>rd</sup> line of file,

**(esc):3,5s/pattern/newpattern/g** – to replace the pattern from line 3 to line 5

**(esc):3,\$s/pattern/newpattern/g** – it will replace the pattern from line 3 to end of the file

**(esc):2g/pattern/newpattern/** -

**(esc):\$s/pattern/newpattern/g** –

**:line number** – move the cursor to specific line

**(esc):%s/pattern/newpattern/g** – will replace the pattern with new pattern in all the lines of a file

**(esc):s/pattern/newpattern/g** – it replace the pattern in current line of a file

**:5,7d** – delete lines from 5 to

**:3,5d** – delete the line from 3 to 5

**:5d** – delete the line 5

**(escape)dd** - deleting the line

**:wq!** - write and quit the program

**:q!** - quit without saving

**:set nu** – to set line number

**:set nonu** – to remove line numbers

**cat** - to display the content of the file

**rm -rf:** Command is used to delete the folder. Example: **rm -rf folder name**

**rm:** command is used to delete the file. Example: **rm filename**

### To delete multiple files/directories

Folder: **rm -rf foldername1 foldername2 ....**

File: rm filename1 filename2 ....

### Deleting files through cases

Case 1 : by names – rm filename

Case 2 : by extensions – rm \*.ext (.html , .xls etc)

r – recursive

f - forcefully

### Command to set warning before deleting file/directory

rm -i filenames

rm -rf -i foldernames

**rm -rf \*** : to delete the complete directory

ls – to list the files in current directory

**ls -lt** : used to display recently created files at the top order

**ls -lrt** : used to display recently created files at the bottom

l – long list

r- reverse

t- time

**ls -a** : used to display hidden files (hidden files starts with ".filename")

ls -l filename : to check username/ownership of file

ls -l foldername : to check username/ownership of directory

du : used to display the size of the file

**du -sh folder name** : used to display size of the file/ folder(current file)

**du -sh \*** -used to display size of all the directories/ folders

**du** - disk usage **s**- size **h**- human readable format

df – disk free/size -used to check the size of the disk drive

**df -h** : used to check the size/memory of the system

**df -h .** : used to check size /memory of the current drive

df - disk free

**ls foldername** - to display the content of folder

### Copy files/folder:

**cp** : used to copy a file/folder from source to destination r - recursive

**cp pathofsource pathofdetination** - used to copy the file from source to destination

**cp -r pathfoldername pathofdestination**- used to copy the folder from source to destination

**cp pathofsource samepathassourcewithnewname** – it will create duplicate file with new name,

**note**-since we are copying a file to same path if file already exist then it will over write, if file doesnot exists it will create duplicate files

### Move/Rename files or folders

**mv** - cmd used to rename file or directory and also used to move the file or directories from source to destination path.

**mv oldfilenmae newfilename** - renaming a file

**mv olldirectoryname newdirectoryname** - renaming a directory

**mv directory pathofdestination/** : to move the files or directories to destination

### Change ownership

**ex** : **chown -R root:root foldername**

**sudo chown -R root:root foldername** - chnage username with root permission

**chown** - used to chnage the usernmae and root name of a file and directory

**chown -R usernmae:groupname dircetoryname**- used to chnage ownerhsip of folder name

**chown usernmae:groupname filename** - used to chnage ownerhsip of file name

**"/" -> current directory**

**"/" ->previous directory**

**"/" ->previous to previous directory**

Change permissions

**User : Group : others**

R-4 W-2 X-1

r-read; w-write; x-executable; a-all

**chmod : used to change the permission of a file or directory**

**chmod permission filename** ex: **chmod 764 filename**

**chmod -R permission directoryname** ex: **chmod -R 761 directoryname**

example:

1)change the permission for the directory as read write for user, read execute for group and write for other

`chmod -R 654 directory`

2) read permission to the user for a file

`chmod u+r file name`

3) Add execute for user, write for group and remove write for others

`chmod u+x, g+w, o-w filename`

4) Add execute for all for a file and directory

`chmod a+x filename`

`chmod -R a+x directory`

**history >file**

**head:** command used to display first n lines of a file

ex: `head filename` (default n-10)

`head -5 filename`: first 5 lines

`head -3 filename`: first 3 lines

**tail:** to display last n lines of a file

ex: `tail filename`

`tail -2 filename`: last 2 lines

count the number of words in 35<sup>th</sup> line of a file and store in a new file

count the number of characters in 42<sup>nd</sup> line of a file and upend to the existing file

**wc :word count;** used to count no. of lines, no of words, no. of characters in a file

l- lines

w - words

c - characters

`wc filename`: no of lines, words, characters

`wc -l filename`: no of lines

`wc -w filename`: no of words

`wc -c filename`: no of characters

**>: redirect operator (>)** - used to copy output of one command to a new file.

If the file already exists it will over write.

ex: `wc file > filename ; wc file > file1 ; wc -l file > file1`

**>>: append symbol Append (>>)** - used to append the output of command to the end of a file.

ex: `wc file >> filename ; wc file >>file1`

not only "wc" any other command like "ls" etc can be used along with above operators

**pipe (|)** - used to give or pass output of one command as an input for next command.

**xargs:** the output of the previous command is passed as a sequence (line by line) to the input for next command

`xargs rm -rf`

ex: `file = filename`

to display 7th line of a file

`head -7 file | tail -1`

to display 5th line of a file

`head -5 file | tail -1`

to display 5th and 6th line of a file

`head -6 file | tail -2`

to count no. of characters in a 7th line of a file

`head -7 file | tail -1 | wc -c`

to count no. of words in 5th & 6th line of a file

`head -6 file | tail -2 | wc -w`

to count no. of characters in a 8th line of a file and copy to a new file

`head -8 file | tail -1 | wc -c >file1`

If an empty line occurs in between file, what would be the output? Answer: `=> 1`

**hostname :** used to display server name

`uname -v` : check linux version

`uname -a` : to display all the details of linux

**To print the statements:**

`echo *` : this command does the same job as "ls" command do

`echo` : used to print the statements :syntax : `echo "welcome"` ; output = welcome

`echo -e` : used to print the multiple statements. Example `echo -e "good morning \n welcome"`

`\n` : will display the statements in two lines: output will be displayed as

good morning

welcome



ex: echo -e "good morning \t welcome"

\t : t => tab - will display the statements in with space(tab) output: good morning    welcome

### grep : is used to search a pattern in a file

syntax: grep pattern filename or grep "pattern" filename

recommended to use pattern within "" ie, 2nd syntax

#### ***It is used to search specific pattern in a file (case sensitive)***

ex: grep "pattern" filename

- grep -e "patternname1" -e "patternname2" filename - used to search multiple patterns, we can this as e grep which is used to search multiple patterns

example for case insensitive =>

- grep -i -e "patternname1" -e "patternname2" filename

- grep -i "patternname" filename

example to search specific word in a file=>

- grep -i -w "pattern" filename

the pattern in above syntax has to be a word from file (case insensitive)

- grep -i -c "pattern" filename => used to count no. of lines which contain the patterns  
  
grep -w -c "pattern" filename – used to count no. of lines which contain the patterns
- grep -i "patternname" \* => used to search pattern in all the files which is in current directory (directory will not be considered)
- grep -i -R "patternname" \* => used to display all the files in directory and in its sub directory (case insensitive). It will display the file name along with that also displays the line which contains pattern
- grep -i -rl "patternname" \* => same as ( "grep -R -l "patternname" \* ) but case insensitive. It will list all the files in directory and in sub directory, without the contents of the file
- grep -l "patternname" \* => it will list all the files which contain the pattern in current directory.
- grep -R -l "patternname" \* => it will list file name in the present directory as well as sub directory only if the file contains the pattern
- grep -i "^patternname" \* => it will print the line which starts with the pattern
- grep -i "patternname\$" \* => to print the lines which ends with letters/patterns

- `grep -i "^$" *` => to print the empty lines in a file
- `grep -i -c "^$" filename` => to count the empty lines of a file

`grep -i "^hi" filename` => display the line which start with hi

`grep -i "remove$" filename` => display the line which ends with remove

`grep -i "^varun" filename` => display the line which starts with Varun

- `grep -v "pattern" filename` => to print all the lines which does not contain a pattern
- `grep -n "pattern" filename` => to print lines along with line number which contain a pattern
- `grep -v -n "pattern" filename` => it will display line which doesnot contain a pattern along with line numbers

### SED command (stream editor)

It is used to replace a string by another string in a file, delete the specific lines from a file & to display the specific line.

Sed 's/pattern/pattern2/g' filename : replace a string only in a console o/p not in the file

S indicates substitute -

Sed -i 's/pattern/pattern2/g' filename : it will replace a pattern in a string in a file

i indicates insert-

Sed -i 's/pattern/pattern2/lg' filename: it will replace the string irrespective of cases

Capital I indicates case insensitive

Sed -i '\$s/pattern/pattern2/lg' filename – to change the pattern at the last line of the file

Sed -i '2s/pattern/pattern2/lg' filename - 2S : it will replace only in a second line

Sed -i '2, 5s/pattern/pattern2/lg' filename - 2, 5S means it will replace in 2 thru 5 lines

Sed -i '1s/pattern/pattern2/2g' filename: It will replace from second occurrence of a file in the 1<sup>st</sup> line

Sed -i '1s/pattern/pattern2/2' filename: It will replace only the second occurrence of a file in the 1<sup>st</sup> line

Sed -i 's/pattern//g' filename: to delete the pattern

### How to delete a line using sed commands

Sed -i '4d' file name : it will delete 4<sup>th</sup> line of a file

sed -i '2,5d' file name : it will delete lines from 2 to 5 of a file

sed -i '2d;5d' filename : it will delete 2<sup>nd</sup> and 5<sup>th</sup> line of a file

sed -i '/^\$/d' file name : it will delete a empty line in a file

sed -i '/patternname/d' file name : it will delete a line which contains a pattern

### How to print the lines using sed command

sed -n '4p' filename : to print the 4<sup>th</sup> line of a file

sed -n '2,5p' file name : to print the 2 to 5<sup>th</sup> line of a file

sed -n '3p;5p' filename : to print 3<sup>rd</sup> and 5<sup>th</sup> line of a file

sed -n '\$p' file name to print the last line of a file

### Cut is used to cut the files (display) in column wise

cut -d " " -f1 filename : d : display the first column of a file

delimiter f :column 1: first column

cut -d " " -f1-3 filename : display from 1<sup>st</sup> through 3<sup>rd</sup> column of a file

cut -d " " -f1,3 filename ; display 1<sup>st</sup> and 3<sup>rd</sup> column of a file

### Disadvantages,

1. we can't display in row wise
2. If we leave more space between the columns, then we will not get desired output.

### awk : this command is used to cut (display) the files in both column and row wise

awk -F " " '{print \$2}' filename : To cut & display the 2<sup>nd</sup> column of a file

-F : field separator

awk -F " " '{print \$NF}' filename : : To cut & display the last column of a file

awk '{print \$NF}' filename : To cut & display the last column of a file

awk '{print \$(NF-1)}' filename : To cut & display the last but one column of a file

awk '{print \$2, \$4}' filename: To cut & display the 2<sup>nd</sup> and 4<sup>th</sup> column of a file

awk ' (NR>1) {print \$2, \$5}' filename : display from 2<sup>nd</sup> row and print 2<sup>nd</sup> & 5<sup>th</sup> column in a file

awk ' NR==2, NR==3 {print \$NF}' filename : it will display rows 2 to 3 of last column

awk ' NR==2; NR==4 {print \$0}' filename: it will display only 2<sup>nd</sup> and 4<sup>th</sup> row of a file

awk -F " " 'NR==3 | | NR==5 {print \$2}' filename : it will print column 2 from line 3 and 5 of a file

### Free: it is used to check the system memory (RAM)

Free -m mega bites free -g giga bites

### Find : it is used to find the location of a file or directory

Find / -iname filename / indicates searches in root directory -l case insensitive

Find . . (dot) indicates current directory

Find only the files with the file name Devops in the current directory

Find . -type f -iname Devops

Find only the directories with day1 in root directory

Find / -type d -iname day1 : use sudo

How to list the files which are created or modified 3 months ago

Find . -type f -mtime +90

How to list the files which are created or modified within 3 months

Find . -type f -mtime -90

How to list the files which are created or modified within 60days and 30 days ago

Find . -type f -mtime -60 ! -mtime -30

List the files which are modified within 10 min

Find . -type f -mmin -10

List the files which are modified 10 min ago

Find . -type f -mmin +10

List the files which have permission 777

Find . -type f -perm 777

List the files which are empty

Find . -type f -empty

List the files which have size is greater than 4k

Find . -type f -size 4k

To restrict the automatic recursive, we use maxdepth

Max depth (number) indicates it will search the number of level you gave.

Find . -maxdepth 2 -type f -iname filename

### **Sudo apt install tree**

Su (super user or switch user) : it is used to switch (login) as another user

Example: su -username :

Sudo su - : to switch into root userr

Sudo (super user does): it is used to run the command with root permissions

Sudo command

Sudo chown username:group name file name

Example: to switch to root user sudo su –

Exit : to logout

Add user: it is used to create the user

syntax: adduser username

passwd: it is used to set the password for the user

Syntax: passwd username

userdel username : to remove the user

.bashrc file: it is an auto execute file automatically when user logs into system

[Link : it is a shortcut to the file.](#)

Meanwhile if we make any changes in link also it will reflect in file

**There will be two types of links**

1. **Soft link (symbolic link)** : it is short cut to a file if we make any changes to original file it will reflect in link, if the file is deleted then link doesn't work  
Syntax : **ln -s filename linkname**
2. **Hard link**: it is short cut to a file if we make any changes to original file it will reflect in link, if the file is deleted then link will work  
Syntax: **ln filename linkname**

**Diff b/n Soft link and hard link**

1. If I delete the actual file, soft link will not work
2. If I delete the actual file, hard link will work, because it points to the inode of a file. Inode is the memory address (unique identification number for file)

**umask** : it is used to set the default permission for the files and directories

umask 000 – gives full permission for the newly created file

ex 1 : rw user, rw group and rw other

6 6 6 - SUB FROM 777 = 111

umask 111 : it will set the permission rw for user, group and others

ex 2 : need to set rw:r:r 644 -SUB FROM 777 = 133

umask 133: it will set rw:r:r permission to newly created files

**List the files which are started with "t"**

1. `ls t*`
2. `ls |grep -i "^t"`

**port number :**

telnet 23,

http 80,

https 443,

dns 53,

ssh 22

scp 22

tcp

### How do you check from how long the server was up and running?

From how long the system/server is up and running : **uptime** it is used to check load average and also to check the server is up & running

Using uptime we can display load average, for the past 1, 5 and 15 min respectively

**Load average:** it is the average number of process that are either in runnable (using CPU, waiting to use CPU) or uninterruptible (waiting for IO access)

IO : Input output

**Zero (0) : is no load and 1 : is fully loaded**

Load average displays average for the past 1, 5 and 15 min respectively

### How to check other /remote servers/systems are up & running

Ping **ip address of server**

### htop/ top: how you're going to check which process is taking more memory

it is used to check which process is using more cpu usage, whichever the process that displayed on the first row, it will be having more cpu usage

**Ps (process status):** ps is a command used to check current process running on a system in a background

How do you check whether a specific process is running or not.

Ps -ef : it will print all running process with more detailed output

Ps -e: print all the process with in the system

Ps : print all the process for current user and terminal

How do you check/list whether specific process running or not

**Ps -ef | grep -l "processname"** or ps -C processname

How do you list process started by a specific user

Ps -u username

### Kill: kill is used to stop the process forcefully

Kill -9 pid

Pid – process id

### Gracefull stop

Sudo service servicename stop

Sudo systemctl stop servicename

Sudo systemctl start servicename

Sudo systemctl status servicename

### How your going to stop the process gracefully

Sudo service service/processname stop

Sudo service service/processname start

Sudo service service/processname restart

Sudo service service/processname status

Sudo apt(ubuntu)/yum(redhat) install packagename

How to install the packages for ubuntu/debian, apt install package name

For redhat/sentos, yum install packagename  
Ubuntu/Debian – apt or apt-get

Redhat/centos – package manager is yum

### Installing Packages

Syntax - Package manager install package name

For ubuntu = apt install packagename

For centos/redhat = yum m install packagename

To install packages we require admin access or sudo permissions

Ex: Sudo apt install netstat

Sudo apt list –install packagename

How do you check whether a specific package is installed or not

Sudo apt list –install packagename | net

how do you run the script at the background: use and symbol (&) at the end of the command

**example:** ls &, cat &

**Nohup (Nohang up)** ; while running a script if execution is stopped due to some network issue once network is back it resumes the execution from where it was stopped.

nohup command name &

how to list all the users in a server

cat /etc/passwd | awk -F ":" '{print \$1}'

or awk -F ":" '{print \$1}' /etc/passwd

**Who :** it will display how many users logged in to the system

**whoami:** to check who is the user

**Netstat :** it is used to see the available ports on system and listening(used) ports

net -tools => netsat -a : listening(used) and open ports

netstat -at : display only tcp ports

netsat -au : to check only udp ports

netstat -l : only listening ports

netstat -lt : listening tcp ports

How to check which are all the ports used

netstat -atulp

How do you check whether a specific port is open or not/

Netstat -atulp | grep "portnumber"

-n indicates active connections, address and port numbers are expressed numerically & no attempt is made to determine names

-p stands to check protocols (tcp, udp)

**Date:** it is used to display the current system date and

date '+%D': it will display date in mm/dd/yy

date '+%A': it will display day

date '+%d': it will display only date

date '+%a': it will display the first 3 letters of specific day

date '+%M': it will display minutes

date '+%m': it will display minutes

date '+%h': it will display month name

date '+%H': it will display hours

date '+%S': it will display Seconds

date '+%T': it will display time in the 24h format



```

ubuntu@ip-172-31-36-17:~$ date
Wed Aug 25 02:17:18 UTC 2021
ubuntu@ip-172-31-36-17:~$ date '+%D'
08/25/21
ubuntu@ip-172-31-36-17:~$ date '+%A'
Wednesday
ubuntu@ip-172-31-36-17:~$ date '+%d'
25
ubuntu@ip-172-31-36-17:~$ date '+%a'
Wed
ubuntu@ip-172-31-36-17:~$ date '+%m'
08
ubuntu@ip-172-31-36-17:~$ date '+%H'
02
ubuntu@ip-172-31-36-17:~$ date
Wed Aug 25 02:18:53 UTC 2021
ubuntu@ip-172-31-36-17:~$ date '+%T'
02:19:08
ubuntu@ip-172-31-36-17:~$ date '+%t'

ubuntu@ip-172-31-36-17:~$ date '+%S'
31
ubuntu@ip-172-31-36-17:~$ date '+%M'
19
ubuntu@ip-172-31-36-17:~$ █

```

**History:** it will display all the previously executed commands

**Whereis :** it will display the executable files and source files of the command where it is present

**telnet :** command is used to check if the remote server port is open or not

it is used to break the firewall

Syntax: **telnet servername port value**

how do you list open files?

**lsof:**

find the process running on specific port

Syntax: **lsof -l portnumber**

Example: **lsof -i tcp:22**

List the files used by a specific process

**lsof -p pid(process id)**

List the open files for a specific user

**lsof -i -u usernames**

## How do you set up password less connection between server

**In Server 1:** generate the ssh keys (create a public key and private key) using **ssh-keygen** command

Copy the public key and go to server 2

**Server 2:** go to server 2 copy the public key to the `authorized_keys` file under `.ssh` folder

Next time we login it will login without password

How do you login to the remote server

Ssh `username@privateipaddress/servername`

We can also login using a pem file

Ssh `-i pemfile username@servername`

## SCP: used to copy file or directory from one server to another

Scp `pathoffile username@server:/home/ubuntu`

Ex: Scp `/home/ubuntu/filename root@privateipaddress:/home/ubuntu`

Scp `-r pathofthedirectory username@servername:pathofthedestination`

## How to you login (ssh) or login to remote server

Ssh secure shell

ssh `username (root/ubuntu/any)@server`

**scp** : it is used to copy a file or directory from one server to another server

**scp (secure copy)** : scp `filename username@servername:/home/ubuntu/`

to copy a directory scp `-r directory name user@servername:/home/ubuntu`

**rsync**: it is also used to copy a file/directory from one server to another server

## Diff b/w rsync and scp

While copying a data from one server to another server if copying stopped in between due to some network issue once system is back if I use scp command it will start copying from beginning

If I use rsync command it will start copying from where it was stopped [Uptime:](#)

## How to sort lines and numbers:

Sort,

Sort -r

Sort -n

Sort -nr

Ubuntu user in server 1 and root user in server2

Ssh root@ipaddress

Curl :

Sudo Apt-get curl

wget : used to download the resources into system ex :

unzip: unzip filename

zip:

tar:

untaz:

## SHELL SCRIPT

Write a script to read two or multiple integer values and display them:

```
#!/bin/bash
echo " the cost of the apple is a"
read a
echo " the cost of the pineapple is b"
read b
echo " the cost of the mango is c"
read c
echo -e " the cost of the apple is $a \n the cost of the pineapple is $b \n the cost of the mango is $c"
```

```
ubuntu@ip-172-31-42-125:~/linuxscripts$ ./integer.sh
 the cost of the apple is a
10
 the cost of the pineapple is b
20
 the cost of the mango is c
30
 the cost of the apple is 10
 the cost of the pineapple is 20
 the cost of the mango is 30
ubuntu@ip-172-31-42-125:~/linuxscripts$
```

Write a script store the command to a variable and execute in a shell script

```
#!/bin/bash
ls
pwd
var1=`pwd`
echo "present current directory is : $var1"
```

Write a script to find the sum / multiply / division of two numbers

```
#!/bin/bash
echo "enter the value of a"
read a
echo "enter the value of b"
read b
c=`expr $a + $b`
d=`expr $a \* $b`
e=`expr $a / $b`
echo " the value of a and b is $c"
echo " the value of a ann b is $d"
echo "the value of a and b is $e"
```

```

#l/bin/bash
echo -e "welcome to the devops class \n devops is good at market now \n devops can be easy to understand"

*****
***hi *****
*****

#l/bin/bash
name="sandesh"
echo "the name is $name"

the name is sandesh

name: sandesh
empid: 1234
designation: devops engineer
companyname: capgemini

$# : it indicates the total number of arguments(count) passed to the script
$a : all argument passed to the string
@a : all argument passed to the string stored in array format. ex: a=$@, $a{1}, $a{2}
$$ : process id of the current running process
$_ : process id of last command went into the background
 $? : used to print the status of last executed command it display 0, if command is successful and display non Zero if the command is failure

```

### 3types of conditional statements:

```

type1
if [ condition ]
then
    statement
fi

type2
ifelse :
if [ condition ]
then
    statement
else
    statement1
fi

type3
if [ condition1 ]
then
    statement1
elif [ condition 2 ]
then
    statement2
elif [ condition 3 ]
then
    statement3
else
    statement4
fi

```

### Note:

#### 1. Strings

#### Numbers

String	Numbers	Meaning
==	-eq	Equal
!=	-ne	not equal
>	-gt	greater than
>=	-ge	greater than or equal
<	-lt	less than
<=	-le	less than or equal

Write a script to check whether the given number is equal to 5 or not

```
#!/bin/bash
echo "enter the number"
read num
if [ $num -eq 5 ]
then
    echo "the given number $num is equal to 5"
else
    echo " the given number $num is not equal to 5"
fi
```

write a script to find the biggest of two numbers

```
#!/bin/bash
echo "enter the value of a"
read a
echo "enter the value of b"
read b
if [ $a -gt $b ]
then
    echo "the value of a is greater than b"
else
    echo "the value of b is greater than a"
fi
```

Assn: need to write a script to find the biggest of 3 numbers

```
#!/bin/bash
echo "enter the value of a"
read a
echo "enter the value of b"
read b
echo "enter the value of c"
read c
if [ $a -gt $b ] | [ $a -gt $c ]
then
    echo " the value of a is greater than b and c"
elif [ $b -gt $a ] | [ $b -gt $c ]
then
    echo " the value of b is greater than a and c"
else
    echo " the value of c is greater than a and b"
fi
```

write a script to check the given string is a file or a directory or a link or it doesn't exist

```
#!/bin/bash
echo "enter the string name"
read string
if [ -f $string ]
then
    echo "the given string is file"
elif [ -d $string ]
then
    echo "the given string is directory"
elif [ -L $string ]
then
    echo "the given string is link"
else
    echo "the given string doesnt exist"
fi
```

Note: if the file/directory/link exist in sub directories we need to give path before string name

```
ubuntu@ip-172-31-42-125:~$ ./search.sh
enter the string name
release/1.txt
the given string is file
```

### Factorial:

$4=4*3*2*1$  and  $5=5*4*3*2*1$

$4 = \text{`expr 4 \* 3 \* 2 \* 1`}$

$5 = \text{`expr 1 \* 5 \* 4 \* 3 \* 2 \* 1`}$

looping statement:

type 1 :

while [ condition ]

do

echo "any statement"

done

### Write a script to find the factorial of a given number:

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

```
echo "enter the number to find the factorial"
```

```
read num
```

```
result=1
```

```
while [ $num -gt 0 ]
```

```
do
```

```
result=`expr $num \* $result`
```

```
num=`expr $num - 1`
```

```
done
```

```
echo " the factorial of given number is $result"
```

### Write a scrip to find the sum of first n numbers

```
#!/bin/bash
echo " Enter the number "
read number
result=0
while [ $number -gt 0 ]
do
    result=`expr $number + $result`
    number=`expr $number - 1`
done
echo "sum of first n numbers is $result"
```

Write a script to find the number of characters in each line of a file

```
#!/bin/bash
echo "enter the name of a file"
read name
num=1
while read line
do
    n=`echo "$line" | wc -c`
    echo "The number of characters in line number $num is $n"
    num=`expr $num + 1`
done < $name
```

Write a script to display the person having age greater than 60

```
#!/bin/bash
echo "enter the name of a file"
read name
while read line
do
    age=`echo $line | awk -F " " '{print $3}'`
    if [ $age -gt 60 ]
    then
        echo "$line" | awk -F " " '{print $1}'
    fi
done < $name
```

```
#!/bin/bash
echo "enter the file name"
read file
line=`cat $file | wc -l`
while [ $line -gt 0 ]
do
    head -$line $file | tail -1
    line=`expr $line - 1`
done
```

Use while read line

Write a script to display the content of file in reverse

Write a script to display a file names if they contain pattern and display respective message whether the file contain pattern or not

```
#!/bin/bash
echo "enter the pattern which you need to find: "
read pattern
grep -R -i -l $pattern * > output
if [ $? -eq 0 ] ; then
    echo "the below files contain a $pattern "
    cat output
else
    echo "the files doesn't contain the pattern"
fi
```



Write a script to rename all text files into html files

```
#!/bin/bash
ls | grep -i txt$ > output
while read line
do
name=`echo $line | awk -F "." '{print $1}'`
mv $name.txt $name.html
done < output
ls *.html
```

## For loop

### Syntax

For i in value1 value2, .....

Do

Statement

Done

Write a script to display the values

Ex1: Was

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

```
for i in 5, 10
do
echo "$i"
done
```

PS: for array related refer for loop

```
#!/bin/bash
num="23 46 99 98"
for i in $num
do
    echo $i
done
```

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

```
For i in { 1..100}
```

Do

Echo \$i

done

Ass: Mail -s "welcome" -c mailid

Write a script to find the sum of elements in an array

```
#!/bin/bash
num="5 8 10 13"
sum=0
for i in $num
#!/bin/bash
num="5 8 10 13"
sum=0
for i in $num
do
    sum=`expr $i + $sum`
done
echo "the sum of array is $sum"
~
```

```
write a script to find the sum of elements in an array.
#!/bin/bash
num="1 2 3 4"
sum=0
for i in $num
do
    sum=`expr $sum + $i`
done
echo "The sum of elements is $sum"

write a script to find factorial for a given set of numbers.
#!/bin/bash
num="4 2 6 7"

write a script to monitor the usage of servers if it reached 70% (size of the current drive).
It will send a mail notification to the concerned person.
#!/bin/bash
size=$(df -h | awk -F " " 'NR==2 {print $(NF-1)}' | sed 's/K/ /g')
if [ $size -ge 70 ]
then
echo "Server utilized maximum memory" | mail -s "memory usage" -c girign@gmail.com
fi
Note:
Have you written any script?
1. Yes I have written a script to monitor the usage of server memory.If the usage of memory reached 70% it will send a mail notification to the concerned pers

write a script to monitor the services if the services are stopped automatically it as to send a mail notification
to the concerned team.
[
```

Ass: Write a script to find factorial for a given set of numbers

```
#!/bin/bash
num="5 6 8 10"
for i in $num
do
    temp=$i
    result=1
    while [ $i -gt 0 ]
    do
        result=`expr $i \* $result`
        i=`expr $i - 1`
    done
    echo "the factorial of $temp is $result"
done
```

Write a script to monitor the usage of the server memory, if the server memory reaches its threshold value (70%). It will send an email to the concerned person

```
#!/bin/bash
size=`df -h . | awk -F " " 'NR==2 {print $(NF - 1)}' | sed 's/%/ /g`
if [ $size -gt 24 ]
then
    echo " the directory is used max trushold value"
fi
```

have you written any scripts?

Yes, I have written many scrips and some of them are

1. Written a script to monitor the usage of the server memory, if the server memory reaches its threshold value (70%). It will send an email to the concerned person.
2. Written a script to monitor the services, if the service stop automatically it has to send an email notification to the concerned person.
3. written a script to clean up the old builds (we need to retain new (latest 10) builds and delete old builds)

Write a script to monitor the services, if the services stop automatically it has to send a mail notification to the concerned team

```
#!/bin/bash
services="sshd jenkins"
for i in $services
do
    ps -C $i
    if [ $? -ne 0 ]
    then
        echo "$i Service is not running" | mail -s "monitoring services" -c girigmn@gmail.com
    fi
done
```

Write a script to clean up the old builds (files), we want to retain 10 builds and delete all old builds

```
#!/bin/bash
var=$1 #path
n=$2
echo "$n"
ls -lrt $1 | awk -F " " 'NR>1 {print $NF}' >output
total=`cat output | wc -l`
echo "$total"
while read line
do
    if [ $total -gt $n ]
    then
        rm -rf $var/$line
        total=`expr $total - 1`
    fi
done<output
```

## Switch case

```
#!/bin/bash
echo "enter the input value"
read var
case $var in
1) echo "the value is 1"
;;
2) echo "the value is 2"
;;
'abc') echo "the value is abc"
;;
*) echo "the value is other than 1,2,abc"
;;
esac
```

Write a script to clean up the old builds.(files) (We want to retain 10 builds and delete all old builds.  
1. n=10 ; total no. files =40 ; retain =40-10 =>30  
2. to clean up the old builds(we need to retain 'n' builds and delete old builds.

Switch case syntax:

```
case $var in
value1) statement1 statement2
;;
value2) statement1 statement2
;;
*) statement
;;
esac
```

Write a script to check whether the value is 1 or 2 or string "abc" or other than this.

```
;;
'abc' | 'bca') echo "the value is abc"
;;
*) echo "the value is other than 1,2,abc"
```

## Ass: Write a script to do the following things

If the day is mon, we need to create two files

If the day is tue add content to above file and create one folder "temp" : echo >> tues | mkdir -p folder

Wed move this files to the temp folder : mv

Thu take the backup : cp

Fri remove the files which we have created on Monday

Sat and Sunday display as holiday

```
#!/bin/bash
day=`date +%a`
case $day in
'Mon') touch monday1.txt monday2.txt
;;
'Tue') echo "welcome to devops" >>monday1.txt | cat monday1.txt >>monday2.txt | mkdir -p temp
;;
'Wed') mv monday1.txt temp/ | mv monday2.txt temp/
;;
'Thu') touch backup | cp monday1.txt backup/ | cp monday2.txt backup/
;;
'Fri') rm monday1.txt monday2.txt
;;
'Sat'|'Sun') echo "saturday and sunday is holiday"
;;
esac
```

## Script Automation:

**Crontab** – it is a scheduler used to schedule scripts in Linux

**Crontab -e** : it is used to edit the schedule the scripts

**Crontab -l** : it is used to list the scheduled scripts/jobs

How to schedule the scripts:

\* \* \* \* \* path of the file

1<sup>st</sup> indicates minutes

2<sup>nd</sup> indicates hours

3<sup>rd</sup> indicates date

4<sup>th</sup> indicates month

5<sup>th</sup> indicates day of the week

Ex: I want to run the scripts on 4<sup>th</sup> sept 2021 at 9am

9.05

05 : it will execute exact 5min

\*/05 : it will execute for every 5min

\* : it will execute for every min

```
05 --> It will execute if minutes is exactly 5 minutes of every hour
* --> Will execute for each minute.

Hour--> */09 --> executes for every 9 hours
        09 --> executes exactly at 09th hour
        * --> executes for each hour
DAY --> by default value of Sunday ==> 00
        Monday ==> 01
        .....
        Saturday ==> 06

Last value as 01-03 --> Monday to Wednesday
as 01,03 --> Monday and Wednesday
```

**Crontab \_** it doesn't require input from the user

I want to run the script on 24<sup>th</sup> aug 11.30 pm

Last value as 01-03 --> Monday to Wednesday  
as 01,03 --> Monday and Wednesday

Crontab --> doesn't require any inputs from the user.

Usually used to initiate the disk usage, service stop/start notifications or any other intimations regarding

2) I want to run the script on 24th Aug 2021 11.30PM ?

```
30 23 24 08 05 filepath
```

3) I need to run every 1 hour of Tuesday ?

```
MM HH DD MM DAY  
* * * * 02 filepath
```

4) I need to run for every 30mins on Friday ?

```
*/30 * * * 05 filepath
```

```
*/30 * * * 05 filepath
```

5) I need to run on Monday and Thursday for every 5 and 10mins ?

```
*/05 * * * 01,04 filepath
```

For every 5mins also means every 10mins, it is executed.

Assn: Write a script if we

press 1 it will create 2 files

Press 2 to change the permission of a file

Press 3 to search a pattern in a file

Press 4 to find / list the files in current directory recursively

```
#!/bin/bash
echo "press any number from 1 to 4 and 5 to exit"
read var
case $var in
  1)
    echo "enter the file names which you want to create"
    read file
    touch $file
    ;;
  2)
    echo "enter the file name to change the permission you want"
    read name
    echo "enter the permission"
    read perm
    chmod $perm $name
    ;;
  3)
    echo "enter the pattern name"
    read pattern
    grep -i $pattern *
    ;;
  4)
    ls -R
    ;;
  5)
    exit 0
    ;;
esac
-- INSERT --
```

# GIT : Version controlled tool

git is a version control tool used to keep track of the version of files and directories.

Some of the version control tools are GIT, SVN, CVS, GITLAB, BITBUCKET

Library/dependency/package – packagename –version

Git –version – to check version

Currently am using git 2.25.1 version

[How to initialize the git repo \(how to convert normal folder into git repo\)](#)

Switch in to the folder and execute the command git init

[How to configure the user](#)

git config --global user.name "username"

git config --global user.email "mail id"

Should created git token

**git architecture there are 3phases**

1. workspace: it is the place where we edit the project related files
2. staging area: it is a intermediate area where we save the changes by using the command git add filename, we can add filename to the staging area
3. Git repo: here versions of files will be tracked. We can use Git commit command to move to the git repo

[Git clone url address](#)

[Git clone -b branchname](#)

[git status](#): it is used to check whether the files are in workspace or in staging area or in get repo

[git add --all](#) : used to move all the files from workspace to staging area/ it will also move deleted files/changes

[git add filename](#): used to move the specific file from workspace to staging area

[git add .](#) : used to move the all file from workspace to staging area.

[git commit -m "message"](#) : used to move the files from staging area to git repo

[git push origin master](#): used to move the files from local to central repo

[git log](#): it will display the history of a repo

[git log filename](#) : it will display the history of a specific file

[git log -2](#) : it will display the history of the last 2 commits

[git log commit id](#) : it will display the history(username, servername, date, time and commit id) of the specific commit / it will move the version of that commit ID or it will display from that commit ID

**tag**: tag is a name given to set of versions of files and directories. tag easy to remember in future, it indicates milestone of a project

[git tag](#): it will list all the tags created

[git tag tagname](#): used to create a tag with updated version

`git tag -d tagname` : used to delete the tag locally

`git push origin --delete tagname`: to delete the tag in central repo through terminal

`git checkout tagname`: it will switch (copy) to a specific tag

`git push origin tagname`: it will push from a specific tag

`git push --tags`: it will push all the tags to central repo

#### **Ass:**

1. create 3 files and create a tag with tagname release1.0 and push this tag to the central repo.
2. modify above 3 files which are created
3. create a tag with release1.1 and push this tag to the central repo
4. create 2 more files and create a tag with release1.2 and push this tag to the central repo
5. 4 delete the tag release1.1 remotely (central repo)

**Branches:** branch is for parallel development, if two team/people will work on the same piece of code by creating different branches and we integrate by merging.

`git branch` : used to list all the branches

`git branch branchname`: used to create a branch

`git branch -d branchname` : used to delete the branch local repo

`git push origin --delete branchname` : used to delete the branch in central repo

`git branch branchname tagname/oldbranchname` : used to create new branch from tag or other branch

`git checkout branchname` : used to switch to the another branch

`git checkout -b branchname` : used to create new branch and switch to that branch

`git merge branchname` : before executing this command, switch to the branch where we need to merge and type `git merge branchname` (give branch name from where we need to merge) command

`git push origin --all` : it will push all the branches to the central repo

`git stash` : when am working on current branch if I get any critical bug which needs to be fixed in other branch, before switching to other branch I need to stash incomplete(half done) work on current branch. Once I fixed the bug I switch back to the current branch I will pop the stash entry and continue with the work. This will avoid committing the changes in wrong Branches.

#### **git stash**

Use: This will avoid committing the code on wrong **other** branches.

`git stash`: it will stash the incomplete work/ it will save the changes in stash entry. Before executing this need to execute `git add filename` or `git add` .

`git stash pop` : it will retrieve the last saved changes in the stash entries

`git stash --list` : it will list the stash entries that you have saved

`git stash --drop`: it will remove the last stash entry from the list.

`git stash --clear`: it will remove the all the stash entries



**Merge conflicts:** it will occur when same line of code is modified on two different branches or from 2 teams on same file & When we try to merge these changes of two branches, we get merge conflicts.

I don't know whose changes should consider and which changes should consider to merge so I will contact developer who modified the codes on two different branches, and they will discuss and give the new change. I will put this new change in the file and continue with the merge. Using git log filename we can get developer who have modified the file

### **Difference between rebase and merge:**

**Merge** just integrate the changes from one branch to another branch

**Rebase** is nothing but a merge, one branch will get added to the tip of another branch in rebase **we can squash multiple commits as a single commit and then we can merge.**

While merging it doesn't allow any git function until unless merge conflicts resolved

Git rebase branchname

Git rebase -i HEAD~3:

Difference between git merge and rebase

Git merge just integrate the changes from one branch to another

How do you merge specific change: **git cherry-pick commit id**  
**git cherry-pick commit id 1 commit id 2 : merge changes from 2 commits**

How do I know how many files are modified under one commit: **git show commit id**

How your going to check whether it is a good commit or bad commit: **git bisect**

How your going to check who modified each line of a code: **git blame**

**Git clone:** it is used to bring the central repo or remote repo to the local repo for the first time.

Syntax: git clone url

**Git pull:** it will bring the changes from the central/remote repo and merges to the local workspace automatically

Or It will bring the changes from all the branches of central/remote repo and merge to the respective branch in local workspace

Syntax: git pull url

**Git fetch :** it will bring the changes from central/remote repo and stores it in separate branch in local repo. We can review the changes and merge if it is required.

Syntax: git fetch url

### **Difference between git and svn, CVS or (any other version control tools)**

Git is a distributed version control system the whole repo will be there in local workspace/local repo. If I want to go to the previous version of a code I can go in local workspace itself

Svn/cvs is a centralized version control system only latest version of code will be there in local workspace. If I want to go to the previous version of a code I need to checkout from central repo

Git has many advanced features like **stash, rebase, reset and fetch** where as in svn we don't have direct commands of these.

Example for distributed version control tool: git, bitbucket

For centralized version control: svn, cvs

**Repository:** where your Git projects are stored. All changes in a project and versions of saved files are in its repository.

Types of repo:

1. **Bare repo** : it acts as a remote repo, where we can only do push and pull the code to this repo. We cannot run any git function  
Syntax: git init --bare
2. **Non bare repo**: it is a local repo where we edit our project related files, we can run all git commands.

**Git revert:** it is used to undo the committed changes. but History will be tracked.

**Git revert commit id** : it will revert the changes which is we have done in specific commit

**Git revert HEAD:** it will revert the changes made in last commit

**Git revert HEAD~1:** it will revert the changes made in last but one commit

**Git reset:** it is used to undo the committed changes but history will not be tracked. The name itself indicates reset pointer HEAD to the previous commit.

**Types of reset:**

1. **Mixed** : it is used to move the files from staging area back to the workspace  
Syntax: git reset --mixed commit id
2. **Soft** : it will remove history and changes will be moved from git repo to the staging area, there w  
Syntax: git reset --soft commit id
3. **Hard**: it will remove the history and changes from everywhere (git repo, staging area and workspace)  
Syntax: git reset --hard commit id

.gitignore:

**How you're going to give the access for the repo:**

We need click on settings of repo, in that click on collaboration and teams. Here we are going to add the users By adding username. Only we will give read access.

**Branching strategy:**

Branches can be created for multiple reason here we create branches for releases.

We followed GitFlow as the branching strategy. Our Branching strategy has branches like , Feature branch , Dev branch , Master branch and Hotfix branches.

Feature branch: developers will create their own feature branch and merge it to the dev branch.

Release branch: used to maintain the releases.

Hot fix branch: we used these branches to deliver critical fix to the production environment.

Master branch: Would be the replica our production environment, Its always a clean branch.

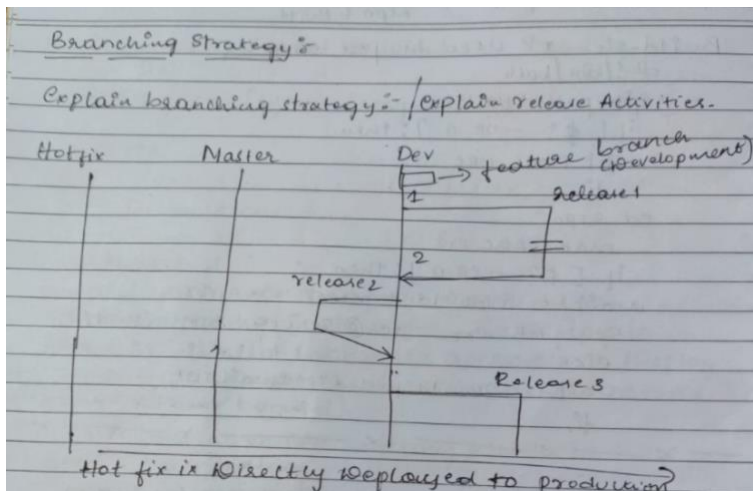
Development will be going on dev branch once the code is ready for first release on dev branch, we create separate branch for first release, and we make release from release branch.

Whatever the issues related to first release will be fixed on first release branch.

Parallel development will be going on dev branch for second release.

Once the code is ready for second release on dev branch, before we create release2 branch we merge the first release to the dev branch and then we create release2 branch for second release.

Whatever the issues we have seen in first release will not be seen in second release.



**Feature branch:** developers will create their own feature branch and merge it to the dev branch.

**Release branch:** used to maintain the releases.

**Hot fix branch:** we used these branches to deliver critical fix to the production environment.

**Master branch:** developer directly can't merge the code to master branch, it is used to keep clean working code.

**Build:** binary or executable before testing.

**Release:** it is a tested build ready to release to a customer.

**Sanity or bvt (build verification test):** it is a basic functionality of a build that should never break.

**Hot fix or patch build:** it is a critical fix which needs to be delivered to a customer within few hours. If I modified 5 files among thousands of files, only 5 files will be compiled and regenerate build. It takes less time.

**Load build (full build):** we compile source code from the scratch. It takes more time.

**Release note:** it has two things tagname and known issues. Devops engineer will prepare release notes.

### What kind of issues you faced while building? (build issues)

Once the build is broken, we need to debug and identify why the build is broken it may be two reasons,

- a. Compilation issue: If it is compilation issue, we need to work with developers.
- b. Build environment issue: If it is build environment issue, we need to debug and fix it. This issue may be the reason of following.
  1. Memory is full on the server
  2. CPU usage is high on server
  3. Slave machine will not available in Jenkins job
  4. Compatibility issues (version mismatch)
  5. Dependency not installed

### **Build Tools:**

C and C++ : make => .exe , it will generate filename.exe

Java: maven, gradle, nodejs, sbt => it will generate .jar, .war and .ear

Dot net: npm, nuget

### **Compiler**

C: Gcc

C++: g++

Java: javac

Note: we support deployment to the QA, dev, stable, UAT and production. We deploy build to the respective server(environment) using Jenkins job. To deploy to the QA and Dev environment we need manager approval.

### **How are we deploying in prod environment?**

Before deploying to production environment, we need to test in lower environments (QA and Dev) if its success then we will raise a ticket to SRE (site reliability engineer) for deploying in production environment. After approval with the help of SRE team squad we are going to deploy to production environment.

### **Do you write pom.xml**

It is given by dev team. Usually we will update for local dependencies and also there is a file called settings.xml in this file, we will update the path for dependencies. Settings.xml should be on build server where compilation will be going on.

### **Maven repository:**

Maven search dependencies in local repo, if it doesn't find then it searches in central repo (artifactory tools) if it doesn't find then its search in remote repo.

Have you written pom.xml: pom.xml is given by dev team. We mention path for local dependency.

<https://github.com/vikas0105>

clone parcel service

Mvn clean install

### Maven goal life cycle:

It is a build tool used to run the java projects

1. **Validate** : validate the project is correct and all necessary information is available
2. **Compile** : compile the source code of the project
3. **Test** : test the compiled source code using a suitable unit testing framework.
4. **Package** : take the compiled code and package it in its distributable format, such as a JAR/WAR
5. **Integration\_test**: used to do the integration test for the compiled source code
6. **Verify** : verify the integration test to ensure the quality criteria are met
7. **Install** : install the package into the local repository
8. **Deploy** : we will deploy the final package into the artifactory tools.

### Tomcat server:

Sudo apt update

Install java: sudo apt install java

Install tomcat: apt install tomcat

Public ip address:8080 (new tab)

C++ pearl java

Nexus

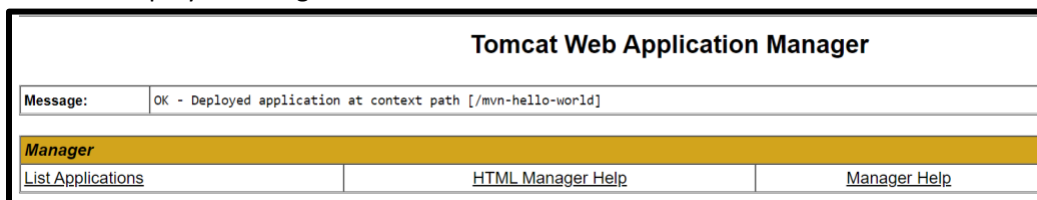
**For Tomcat installation please follow below link:**

### First we need to install java

1. sudo apt-get update or sudo apt update
2. sudo apt-get install default-jdk

[https://github.com/sandy2498/Simple-DevOps-Project/blob/master/Tomcat/tomcat\\_installation.MD](https://github.com/sandy2498/Simple-DevOps-Project/blob/master/Tomcat/tomcat_installation.MD)

when we deploy we will get below result in tomcat.



### Where to add rules for a branch?

We need to go for a setting in repo, click on branches and click on add rule. Here we will add the rules whichever we required

# JENKINS

Jenkins is an open-source automation tool, which is used to create CI/CD pipelines

**Continuous integration:** integrate the changes from development team continuously without manual intervention. We can give quick feedback to developers for their change.

**Continuous deployment:** tested build should be deployed to production environment as early as possible.

**Continuous delivery:** each change from dev team should be build and deploy to the test environment to test the changes.

## Installing Jenkins

first check if java is installed or not.

if not installed install

go to google and search Jenkin installation ubuntu - use digital ocean!

<https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-22-04>

note: pkg.jenkins.io - we get all the Jenkin related files.

To install Jenkins please follow the steps in below link:

<https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04>

after installation go to Jenkins – add user – in manage Jenkins and then go security under authorization give matrix based security

**Which type of jobs you have created.**

1. Freestyle : lets you create general-purpose build jobs with maximum flexibility.
2. Pipeline
3. Multibranch pipeline
4. Github organization

## 1. How to install the plugins?

We will download the plugin from Jenkin update server/centre (updates.jenkins.io)

We will add the plugin from manage Jenkins, manage plugins in advance tab we are going to upload a file, and install without restart.

## 2. How we are going to secure Jenkins or how do you secure Jenkins?

We used matrix-based security it provides check boxes, whichever boxes we check users will have those permissions. We can provide permissions by selecting check boxes.

Manage Jenkins ->Configure global security->Select matrix-based security ->add users ->select the check boxes

### Note: 3 types of security

1. Matrix based
2. Project based
3. Role based

### Master Slave:

#### Why slave machines are required?

Master slaves are used to distribute the load to different servers. And also, we can run specific jobs on specific servers.

Eg: If I want to compile a "c source code" on server which provide "c environment" only.

### Build triggers:

1. **Build Periodically**  
It will trigger the build based on the time, whether commit or no commit
2. **Poll SCM**  
It will trigger jenkins job based on the commit, if there are any commit within specific time, it will trigger Jenkins Job.
3. **Webhooks**  
It will trigger the job based on the commit and push in the git repo.

### Plugin Installation:

CVS, upstream, downstream, quality gate, artifactory, blueocean, selenium, gearman, publish over CIFS, parameterized plugin, docker pipeline, sonar qube, docker many more

### How to manage plugins?

manage jenkins manage plugins

- available plugins : shows default plugins available

type docker pipeline n select it

click install

- installed plugins : shows all the installed plugins

- advance setting : used to add plugins from local, once uploaded

need to search it in available the plugins

<https://plugins.jenkins.io/sonar/#releases>

the downloaded plugin files will be in .hpi extension

### \*Parameterized plugin:

It is used to pass parameters to Jenkins job. I configured parameterized plugins for deployment job. Deployment job has 2 parameters

1. Server name
2. Build number

These values will be passed to deployment script inside Jenkins job.

### **\*Gearman Plugin (high availability plugin)**

It is a high availability plugin if Jenkin master goes down, Jenkins will be up and running automatically on other servers. Jenkins will be highly available.

Gearman plugin will allow us to add other server details (2<sup>nd</sup> master). When Jenkins is running on first master both servers will be in sync. As soon as Jenkin master goes down Jenkins will be up and running on other server.

### **\*Publish over CIFS:**

Used to copy files from linux to windows

Ex cvn, github, upstream, downstream, artifactory tools

### **\*Sonar qube**

Sonar cube is used to check the quality of source code. If quality of source code is more than 70% then we consider quality of source code is good.

We integrated sonar qube with a Jenkins sonar qube works as a vulnerability tool in terms of scanning the source code.

The unit test case results would be uploaded into sonar qube, upon the unit test cases results we will get Metrix from quality profile and quality gates.

### **How do you trigger jobs based on commit?**

We use Webhooks

Script will push Jenkins's job, Jenkins configuration files to the git repository.

### **\*\*\*How do you take Jenkins's backup? \*\*\***

We need to take back up from the path /var/lib/Jenkins. We take Jenkins's backup and store it in a separate git repository. We have written a script to do this automatically and schedule in crontab.

### **How do you create Jenkins Job?**

In dashboard click new item, select the type of job and job name click okay

In configure page under source code management, select git and type the url of the gitrepo and select the credentials to clone

Under build triggers select required trigger method and in build section select execute shell and type the commands to build



Under post build action select appropriate actions which require, it may be deployment, sending email, triggering other jobs or

### \*\*\*CICD Pipeline (end to end process) \*\*\*

Once developer completes his work, he will raise a PR (pull request) as soon as developer raise PR local build job will get triggered. Once build is success, reviewer or approver will merge the code to central repo by clicking on merge button (merge button will not be enabled in pull request if local build fails). As soon as merges to central repo Jenkins's pipeline job will get triggered.

It has 3 stages.

1. **Build stage:** It will checkout the source code from GitHub to Jenkins local workspace. It compiles the source code and generates the build. Once build stage is success it will automatically trigger deploy stage.
2. **Deploy stage:** It will deploy the build to the QA server and runs sanity test(bvt). Once deploy stage is success it will trigger test stage automatically.
3. **Test stage:** Here we run the test cases which are given by the testing team. We configure this test cases and trigger them automatically. If there are any failure it will be taken care by testers.

How to add sudo permission to Jenkins

- in root vi /etc/sudoers next search in google

**Jenkins pipeline (Jenkins file):** Jenkins file is where we put all our job configuration to achieve multiple instance execution in a single pipeline job (by default it will take one executor until unless we configure more executor)

**Declarative pipeline:** [Pipeline Syntax \(jenkins.io\)](#)[Pipeline Syntax \(jenkins.io\)](#)

It will start with the key word pipeline

2 type of Jenkins pipeline:

1. **Declarative pipeline:**
  - a. **Agent:** It is a block to define which jobs to run in which machine. We can mention agent in pipeline level or in stages level,  
We can define 4 types of agents: a
    - 1) **Any-** It is defined at the pipeline level. It can be run in any slaves based on the load.  
Syntax: agent any
    - 2) **None-** If I define agent as none at pipeline level, then it is mandatory to define agent on each and every stage.
    - 3) **Docker-** Here we are giving image name  
Syntax: agent {image name}
    - 4) **Label-** If I want to run jobs on particular slave then we will use label.  
Syntax: agent { label 'slavename' }
2. **Scripted pipeline**  
It will start with node

22 lines (21 sloc) | 473 Bytes

```
1 pipeline {
2   agent { label 'slave_n1' }
3   stages {
4     stage('Directory') {
5       steps {
6         // sh 'cd /home/slave_n1/workspace/Pipelinejob/java-mvn-hello-world-web-app/'
7         sh 'cd ${WORKSPACE}'
8       }
9     }
10    stage('Compile') {
11      steps {
12        sh 'mvn compile'
13      }
14    }
15    stage('Package') {
16      steps {
17        sh 'mvn package'
18      }
19    }
20  }
21 }
22
```

### What Is the path to change the port in Jenkins?

Etc/default/Jenkins

### What Is groovy? Types of groovy

Groovy is used to define a new pipeline as script and there are two types of groovy.

Declarative pipeline

Scripted pipeline

### What is the difference between declarative & scripted pipeline?

#### Declarative pipeline:

It is blocked label pipeline & it is simple & more optimized groovy syntax

It is relatively new feature to support pipeline code.

The code is written in Jenkin file & which can be checked in source code management.

#### Scripted pipeline:

It is traditional way of writing code.

The code is written in Jenkin file on the Jenkins UI interface.

Scripted pipeline was not typically desirable.

### How you are going to manage Jenkins in master? OR what will do if Jenkin master goes down?

We are using primary & secondary master and whenever primary master goes down then the secondary master will be up & running,

Both the primary & secondary master controlled by JENKIN OPERAION CENTER.

### What is artifact?

It is code ready to conduct deployment.

Sample Jenkins pipeline [Comprehensive Guide To Jenkins Declarative Pipeline \[With Examples\] \(lambdatest.com\)](#)

## Master slave why slave is required

It is used to distribute the load to different server and also we can run specific job on specific server

ex: if I want to compile, Java source code on server which provides Java environment only

note 1: Normally we will never run jobs on master

2: Normally 10 - 15 slaves will be there

3: we will use labels to run on available slave

Write steps for master slave connect.

Q1 - I am having 3 server S1 S2 S3 S1 -master S2 -slave S3 -tomcat, now I need to build project on slave and deploy war file/artifact on tomcat server.

war will be available in S2

Need to copy that file to tomcat webhooks

Need to use scp

What is shared libraries/modules in Jenkins ?

A: Shared libraries/modules in Jenkins refer to a collection of reusable code and resources that can be shared across multiple Jenkins jobs. This allows for easier maintenance, reduced duplication, and improved consistency across multiple build processes.

For example, shared libraries/modules can be used in cases like: - Libraries: Custom Java libraries, groovy functions, and other resources that can be reused across multiple jobs.

Libraries: Custom Java libraries, groovy functions, and other resources that can be reused across multiple jobs.

- Jenkinsfile/Template:

A shared Jenkinsfile can be used to define the build process for multiple jobs, reducing duplication and making it easier to manage the build process for multiple projects.

## diff bw Normal pipeline job & Multibranch job

normal pipeline job is meant for building a specific branch from scm & it deploy respective environment (UAT prod/dev environment).

Pipeline jobs supports both steps to be added in Jenkins config and scm

we can use pipeline for adobe jobs, parameterized jobs execution and to debugging has a code

## multi

multibranch pipeline is meant for building a multiple branches from repo and deploy to respective envs

Note-Do not use multibranch pipeline, if u donot have a std branching strategy and cicd stages.

## Jenkins Shared Libraries

A shared library is a way to store commonly used code(reusable code), such as scripts or functions, that can be used by different Jenkins pipelines.

Extending Jenkins with a Shared Library Instead of writing the same code again and again in multiple pipelines, you can create a shared library and use it in all the pipelines that need it.

This can make your code more organized and easier to maintain.

Think of it like a library of books, Instead of buying the same book over and over again, you can borrow it from the library whenever you need it.

### Why do we need Jenkins shared library?

To deploy the same applications in multiple environments, we may need to go through different stages, but here each stage requires a similar kind of logic to be placed in the pipeline for each environment. Since it doesn't satisfy the DRY(Don't Repeat yourself) principle of Jenkins, there is a need for the concept of Jenkins shared library.

### Global Shared Libraries

There are several places where Shared Libraries can be defined, depending on the use-case. Manage Jenkins » Configure System » Global Pipeline Libraries as many libraries as necessary can be configured.

Since these libraries will be globally usable, any Pipeline in the system can utilize functionality implemented in these libraries.

### Using libraries

Pipelines can access shared libraries marked Load implicitly, They may immediately use classes or global variables defined by any such libraries (details below).

To access other shared libraries, a script needs to use the `@Library` annotation, specifying the library's name:

```
@Library('somelib') Untitled-1 ●  
1  @Library('somelib')  
2  /* Using a version specifier, such as branch, tag, etc */  
3  @Library('somelib@1.0')  
4  /* Accessing multiple libraries with one statement */  
5  @Library(['somelib', 'otherlib@abc1234'])  
6
```

## What goes inside a Shared Library?

### Steps:

These are called Global Variables in Jenkins terminology, but these are the custom steps that you want to be available to all your Jenkins pipelines.

eg -A standard step to deploy an application, or perform a code review. To do this, add your code into **vars/YourStepName.groovy** and then implement a def call function, like this:

```
def call() {  
    1  def call() {  
    2      // Do something here...  
    3  }
```

### Other common code:

This might include helper classes, or common code that you might want to include inside pipeline steps themselves. You could also use it as a place to store static constants which you want to use throughout your pipelines.

Code like this needs to go in the **src/your/package/name** directory, and then you can use normal Groovy syntax.

```
jenkins-shared-lib  
src\com\samplebuilder
```

eg- Here's a common file, xyz/tomd/GlobalVars.groovy (note the package xyz.tomd at the top):

```
#!/usr/bin/env groovy  
1  #!/usr/bin/env groovy  
2  package xyz.tomd  
3  
4  class GlobalVars {  
5      static String foo = "bar"  
6  }
```

## Setting up the library in Jenkins

To add your shared library, In Jenkins, go to Manage Jenkins → Configure System. Under Global Pipeline Libraries, add a library with the following settings:

Name: pipeline-library-demo

Default version: Specify a Git reference (branch or commit SHA), e.g. master

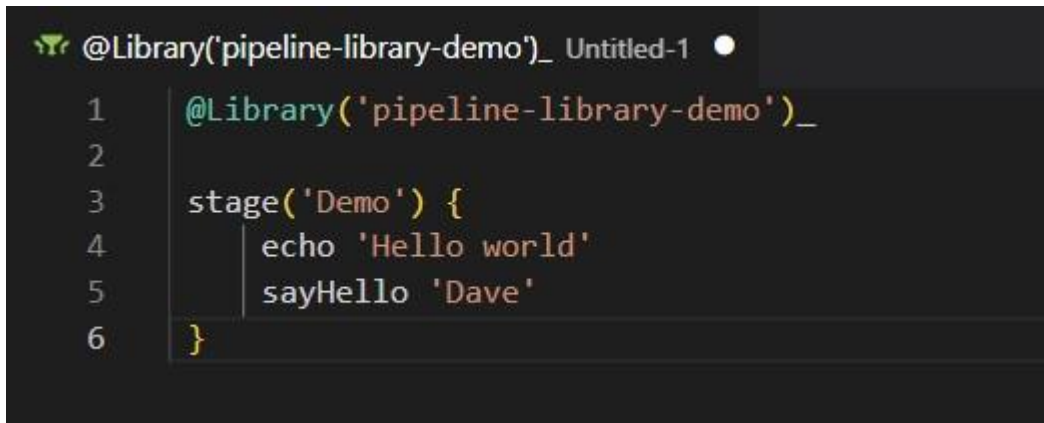
Retrieval method: Modern SCM

Select the Git type

Project repository: <github repo url>

## Using the library in a pipeline

To use the shared library in a pipeline, you add `@Library('your-library-name')` to the top of your pipeline definition, or Jenkinsfile. Then call your step by name, e.g. `sayHello`:

A screenshot of a code editor showing a Jenkinsfile snippet. The editor has a dark background with light-colored text. The title bar of the editor shows a green icon, the text '@Library('pipeline-library-demo')\_ Untitled-1', and a circular icon. The code is as follows:

```
1  @Library('pipeline-library-demo')_
2
3  stage('Demo') {
4      echo 'Hello world'
5      sayHello 'Dave'
6  }
```

## Commonly asked interview questions on shared libraries

What is a Jenkins Shared Library, and why would you use it in Jenkins pipelines?

How do you define and use a function in a Jenkins Shared Library?

What is the purpose of the vars directory in a Jenkins Shared Library, and how is it used?

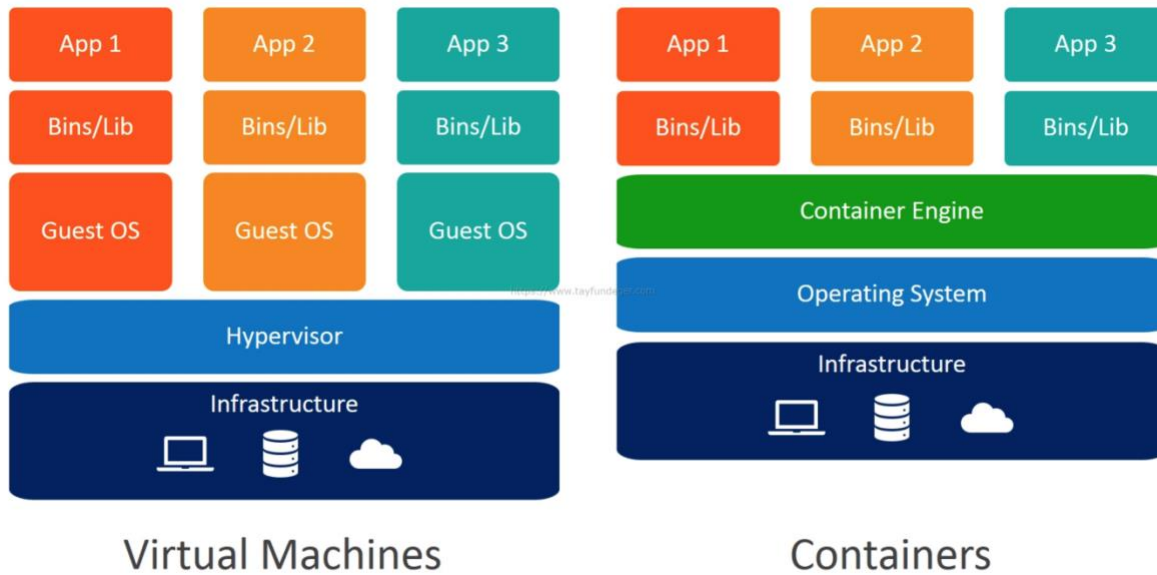
Explain the concept of dynamic loading in Jenkins Shared Libraries.

What are the benefits of using Jenkins Shared Libraries in a CI/CD pipeline?

How can you handle external dependencies in a Jenkins Shared Library?

# DOCKER

Nitro hypervisor – the hypervisor we are using in aws



**Docker Image/image:** Application , code configurations and dependencies are packaged into a single file is called docker image.

Docker images are created by Dockerfile

**Dockerfile:** set of instructions used to create image.

**Docker:** Docker is a PaaS that allows you to build, test, and deploy applications quickly

**Can we install docker on windows?**

No, we cannot install docker as a core software or a programme, but docker has provided a way to install on windows called docker desktop.

When we installed docker desktop, it will automatically spin a Linux virtual machine on windows (oracle virtual machines)

**Container:** It is a group of isolated process. It is a instance or server for that application

**How docker or container work internally?**

**What are cgroups and namespaces how it is related to docker?**

**What is Linux container?**

In a normal virtualized environment one or more VM run on top of physical machine using a hypervisor like xen hypervisor

Container or docker used Linux kernel features like namespace control group (Cgroup), chroot to create container on top of a OS and automate application deployment on container.

Container are isolated in a host using three Linux kernel features

Namespace, cgroup and chroot.

1. **namespace** : process isolation requires separation from other processes which is achieved from default Linux feature called as namespace

namespace helps us to separate and group some process from other process

Some namespaces are PID, UFS (isolated kernel & version identifier), GID , cgroup , net(network interfaces), IPC, Mount, User

2. **cgroup ( resource limiter)** : Cgroup will helps us to control the access, creation and deletion of files.

The resources used by container is managed by Linux cgroup

We can decide on how much CPU and memory resources a container should use using cgroup ex: CPU, Memory (RAM), sockets(Port), devices (usb).

3. **chroot**: it will create root directory

### **Have you ever created or changed cgroup in a container?**

In our company we have not changed any default characteristic but I have worked with cgroup which are created default by docker.

### **How docker will limit resources to container using cgroup?**

CPU, Memory (RAM), sockets(Port), devices (usb).

To install docker, follow link [How To Install and Use Docker on Ubuntu 20.04 | DigitalOcean](#)

Docker commands link [The Ultimate Docker Cheat Sheet | dockerlabs \(collabnix.com\)](#)

<https://github.com/collabnix/dockerlabs/blob/master/docker/cheatsheet/README.md>

### **Docker file instructions:**

1. **FROM**: to pull the image

2. **WORKDIR**: it is similar to CD, to define working directory we use WORKDIR ex: WORKDIR /home, WORKDIR /lib

3. **RUN**: it is used to update the image and commit and also used to run commands and commit it.

4. **CMD**: it is the default command runs when new container starts.

5. **ENTRY POINT**: it is a default command runs when new container starts.

6. **EXPOSE**: it is used to make port available for all container in same cluster.

7. **PUBLISH**: publish is used to map port inside container to the port of host machine, where the same container is in run.

8. **MAINTAINER**: use the name of the user who maintains the image. Usually given after **FROM**

**COPY**: Instructions will copy new file from source path to container file system path.

**ADD**: ADD will do the same thing as copy, but add allows us to do URL's , tar or zip file as a source file, while executing it extract the data and

Arg: `docker build -t imagename --build -arg TEST=/test`



Env: `docker run -d -e TEST="5"`

Difference between ENV & ARG?

**ARG-**

Values are available in docker file during building the docker image

values are not available after image is built or running container won't have access.

**ENV-**

It is mainly meant to provide default values for your future environment variables.

Running docker applications can access environment variables.

It is a great way to pass configuration values to your project.

`-e` – environmental variable for container which can be accessed anywhere inside container.

### Difference between CMD and ENTRY POINT

When both commands are present CMD becomes argument to ENTRY POINT. We can override CMD at run time, but we cannot override ENTRY POINT.

**Can we have two or more CMD or ENTRY POINT commands?**

Yes, but it will consider the last CMD or ENTRY POINT only, remaining will be ignored.

Docker engine is called as docker daemon.

**Docker images:** it is a command used to list all the images exist in local machine

**Docker ps:** it is used to list running containers in local machine

**docker ps -a :** it is used to list all the containers in local machine

**Docker build:** it is a command used to build the images

Syntax: `docker build -t <image name> .` or `docker build -t <image name>:<tag> .`

Or `docker build -t <image name> -f Dockerfile`

**Docker run:** it is a command used to create the container from the image

Syntax: `docker run -it <image name>:<tag>`

`-it` : interactive mode

`-d`: detached mode

**Docker rmi -f imageid:** to delete the docker images

**Docker rm -f containerid:** to delete the containers

**How do I know image uses which port?**

By using docker hub documentation I will come to know image is using which port.

## Port mapping:

It will inform the docker daemon to associated port of container to the host machine.

### There are 2 ways:

1. **-p:** It maps individual port of container to the port of host machine.  
Example: `-p <hostport>:<containerport>`  
`-p 8888:8080`
2. **-P:** All ports of container is used to same port of the host machine.  
Example: `docker run -it <imageid> -P`

**Docker exec:** it is used to get inside the running container. Syntax: `docker .exec -it <container id> /bin/bash`  
(without /bin/bash I can give just "bash" as well)

Docker exec -it containerid bash

### Without base image how to create an image? Or how to create own image or base image?

FROM scratch, it doesn't pull any images the images which is empty

## Volumes and binds/ type of layers/Types of volumes:

Docker containers doesn't store persistent data, any data return to persistent layer will no longer be available once the container stops running.

To solve the problem of persistent data from a container docker has two options

1. Volumes
2. bind mount

**Bind mount:** It is a file or a folder stored anywhere on container host file system, mounted into a running file system.

It exist in host machine so process outsider of docker can also modify it.

Note: (Disadvantage)

Non docker process on docker host or docker container can modify data at anytime

Syn: `-v hostmachinefile/folderpath:containerfile/folderpath`

**Volumes:** single volume can be attached to multiple containers. Volumes are maintained by docker hub no one can access the volume outside the container, this is mountable entity can be used to store persistent data

Volumes are stored in host file system but which is managed by docker, non-docker process should not modify this part of file system.

Create volume syntax: `docker volume create volumename`

**Docker inspect:** if you want to get more information or details of container or volume we use docker inspect.

Syn: `docker inspect container id or volume id`

**List the volumes:** docker volume ls

**Inspect the volume:** docker inspect volume ls

**Mount the volume:** -v volume:containerpath (container path : /usr/local/tomact/webapps)

**Docker tag:** it is used to tag the image with different name or it is used to rename the image

Syntax: docker tag <name of the existing image>:<tag> <new or existing name of the image>:new tag

**Docker registry:** place where docker images are saved.

Default docker registry is docker hub other registries are amazon ECR, nexus registry etc...

**Docker login:** command to communicate with docker hub/docker registry

We need to login to docker registry before pushing or pulling the images

Syntax: docker login -u <username> -p <password> -h <docker registry name>

**How do you list all containers for a particular image?**

Docker ps -a --filter ancestor=<image name>

**How to check whatever execution is going on in a container or how to check logs of the container?**

Sudo docker logs <container id>

We can check only running container

**How to list id of running container?**

Docker ps -q

Docker ps -a -q: to list all the stopped container

**How do you delete all running container**

Docker stop \$( sudo docker ps -q )

Docker rm \$( sudo docker ps -a -q )

**How to check command execution is successful or not**

After execution, if it is success it will display container id otherwise it will display errors

**How docker pull images internally or stores image internally?**

It just stores difference between the images

**Dangling images:** images without name and tag and which are not used by any container are called dangling images

**How to list dangling images only?**

Docker images --filter dangling=true

If `dangling=false` it will list all images other than dangling images

To delete only dangling images

`Docker rmi $(sudo docker images -q --filter dangling=true)`

**Difference between kill and stop?**

**Stop:** it will have grace period to stop process inside the container (10sec)

**Kill:** it will forcefully stop the container without thinking of any process running.

**Pause:** it will pause all process within one or more container.

**Docker objects:** image, container, volume, networks & bridges.

**How do you check which are all ports mapped on to container?**

`Docker ps port containerid`

**Prune:** remove all stopped container, image and volume.

Syntax: `docker image prune`

Remove all dangling images or unused images

**Docker system prune:** it removes all unused docker objects

**Data only container:**

It is used for persistent data storage example : databases in docker. The data only container does nothing else except exposing a data volume. It is used to add volume to other container.

**Postgres-** used to save and pull the data from database. Postgres is one of the databases, we can use it as container.

**Docker Networks/networking:**

For docker containers to communicate with each other and the outside world via the host machine, there has to be a layer of networking involved that is docker networking.

**Different types of docker networks:**

1. Bridge/host networks
2. Overlay networks
3. Macvlan network

**Docker default networking (docker0):**

When docker is installed a default bridge network named `docker0` is created, each new docker container is automatically attached to this network, unless a custom network is specified.

Besides `docker0` two other networks get created automatically by docker.

1. Host (no isolation between host and containers on this network, to the outside world when they are on the same network)

2. None (attached containers run on container specific network stack) or disabling all networks for a container.

### Bridge/host networks:

Bridge networking is the most common network type. It is limited to container within a single host running on the docker engine. Bridge networks are easy to create, manage and troubleshoot.

Bridge networks are usually used when your application run in standalone containers that needs to communicate.

**Host:** for standalone containers, remove the network isolation between the container and the docker host, and uses the host networking directly.

### Overlay networks:

Overlay networks connect multiple docker demons together and enable warm services to communicate with each other. You can also use overlay networks to facilitate communication between a swarm service and a standalone container, or between two standalone containers on different docker demon.

This strategy removes the need to do OS level routing between this container.

### Macvlan networks:

Macvlan networks allow you to assign a MAC address to a container, making it appear as a physical device on your network. docker demon routes the traffic to containers by their MAC addresses.

**Docker compose / docker compose file:** it is a tool for defining and running multiple container docker applications, with compose we use a YAML file to configure your application services. Then with a single command you create and start all services from your configuration.

It is a YAML file defining service network and volumes, it is used to run multiple containers as a single service.

We can start and stop all services using single command.

Docker-compose up or docker compose up -d (detached mode)

Docker-compose down

**Docker swarm:** I have not used docker swarm, but I have idea on this, and I use k8(Kubernetes) for container deployment.

It is a container orchestration tool, meaning that it allows user to manage multiple containers deployed across multiple host machine.

One of the key benefits associated with the operation of a docker swarm is the high level of availability offered for application.

### Install docker compose:

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04>

Ass:Write a jenkins file it has 3 stages

1. Need to checkout source from github
2. Build stage : docker build to build a new image
3. Deploy stage: docker run

And

Deploy to different server

Publish stage: push the new image to docker hub

### **Advantages of Multistage build:**

Multistage build is used to harden the docker images

Multistage build helps in reducing docker image size

In Multistage build we use light weight images such as alpine and Dabian.net images

Each FROM instructions can use different base image.

**Note:** Alpine is light weight images and is about 5mb.

In order build the docker images we need small size image (advantages).

### **\*\*\*CI/CD Pipeline (end to end process) \*\*\***

Once the developer completes his work, he will raise a PR (pull request) as soon as the developer raise the PR local build job will get triggered. Once the build is success, the reviewer or approver will merge the code to the central repo by clicking on merge button (merge button will not be enabled in pull request if local build fails). As soon as merges to the central repo Jenkins's pipeline job will get triggered.

1. **Build stage:** we build respective docker images once the docker images is completes build stage is complete.
2. **Publish and Test:** here we run some basic test cases written by dev and QA team, once the test cases are completed, we send reports to QA team and push the successful artifacts to docker registry.
3. **Deliver:** we do the delivery to the lower environments once the QA test is successful, we do deployment to production environments

### **Docker security:**

There four major areas to considered when reviewing docker security

1. Intrinsic security of the kernel and its support for namespaces and Cgroup

2. The attach surface of the docker daemon itself
3. Loopholes in the container configuration profile in either by default or by customized by customer (we should use always official docker images instead of customized)
4. The hardening security feature of kernel and how they interact with containers

## K8s (Kubernetes)

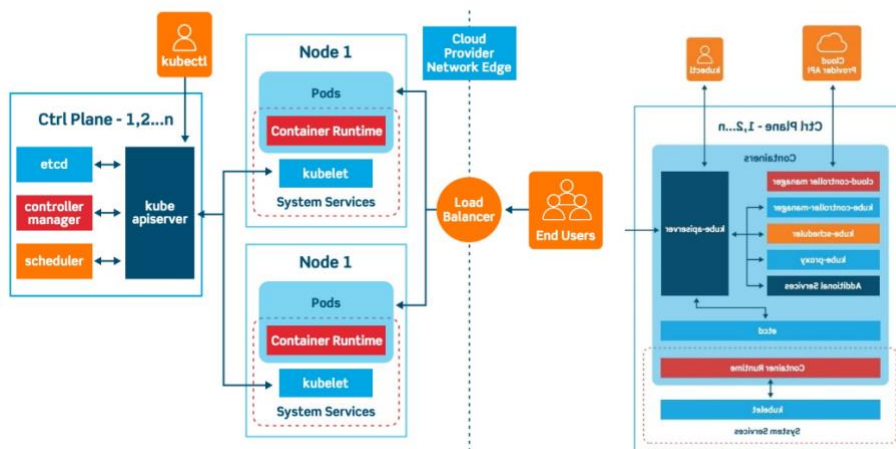
Kubernetes is an open source container orchestration tool developed by google to help you manage the containerised / dockerised application supporting multiple deployment environments like on premise, cloud, or virtual machines.

What are the main features of K8s offers?

1. Assures high availability with zero down time
2. High performance and scalable
3. Reliable infrastructure to support data recovery

Disadvantages: it is very complex or takes lot of time for setting up the k8s infra

**K8s architecture or cluster:**



**K8s cluster has two major components**

1. Master node
2. Worker node

**Node:** physical machine or virtual worker machine where containers will be deployed by k8s.

**Cluster:** set of nodes grouped together.

Master or control panel manages the cluster

**Components of K8s:**

API server, scheduler, control manager etcd, kubelet, container runtime and kube-proxy.

## Kubernetes Basics

we will learn some important Basics of Kubernetes:

### Cluster:

It is a collection of hosts(servers) that helps you to aggregate their available resources. That includes ram, CPU, disk, and their devices into a usable pool.

### Master:

The master is a collection of components which make up the control panel of Kubernetes. These components are used for all cluster decisions. It includes both scheduling and responding to cluster events.

### Node:

It is a single host which is capable of running on a physical or virtual machine. A node should run both kube-proxy, minikube, and kubelet which are considered as a part of the cluster.

### Namespace:

It is a logical cluster or environment. It is a widely used method which is used for scoping access or dividing a cluster.

## Kubernetes Architecture:

Kubernetes has two nodes – master node and worker/slave node.

### Master node:

The master node is the first and most vital component which is responsible for the management of Kubernetes cluster. It is the entry point of all kind of administrative tasks. There might be more than one master node in the cluster to check for fault tolerance.

The master node has various components like API server, Controller manager, kubectl, scheduler and ETCD.

### ETCD:

- This component stores the configuration details and essential values in **key value form**

```
{  
  "name": "xyz",  
  "part": "ght"  
}
```

- It communicates with all other components to receive the commands to perform an action.
- It is also called as meta data storage.

### Controller Manager:

- A daemon (server) that runs in continuous loop and is responsible for gathering information and sending it to the API server.
- Works to get the shared set of clusters and change them to the desired state of the server.



- The key controllers are the replication controllers, endpoint controllers, namespace controllers and service account controllers.
- The controller manager runs controllers to administer nodes and endpoints.

### Scheduler

- The Scheduler assigns the tasks to the slave nodes.
- It is responsible for distributing the workload and stores resource usage information on every node.

### API Server

- Kubernetes uses the API server to perform all operations on the cluster.
- It is central management entity that receives all REST requests for modifications, serving as frontend to the cluster.

### Kubectl

- Kubectl controls the Kubernetes cluster manager.

### Worker/Slave nodes

Worker nodes are another essential component which contains all the required services to manage the networking between the containers, communicate with the master node, which allows you to assign resources to the scheduled containers.

- **Kubelet**: gets the configuration of a Pod from the API server and ensures that the described containers are up and running.
- **Docker Container**: Docker container runs on each of the worker nodes, which runs the configured pods.
- **Kube-proxy**: Kube-proxy acts as a load balancer and network proxy to perform service on a single worker node.
- **Pods**: A pod is a combination of single or multiple containers that logically runs together on nodes.

### Pods:

Pods are smallest objects which can be created in Kubernetes cluster.

In terms of docker concepts, a pod is similar to a group of docker containers with shared namespace and shared file system volumes.

**YAML (yet another mark-up language) File: YAML** is a human readable data serialization language. It is commonly used for configuration files and in applications where data is being stored or

Kubectl create -f filename

### Life cycle of k8s:

**Waiting**: if the container is not neither in running or terminated state it is waiting.

**Running**: the running status indicates the container is running without issues

Terminated: the container in the terminated began execution and then either run to completion or fail for some reason.

### Publishing service:

K8s service type allows you to specify what kind of service you want, the default is cluster IP

Type values and their behaviours are:

1. ClusterIP: Exposes the service on a cluster internal IP. Using this value makes the service only reachable from within the cluster. This is the default service type.
2. NodePort: exposes the service on each nodes IP at a static port.  
A cluster IP service, to which the nodes port service routes are automatically created.
3. LoadBalancer: exposes the service externally using a cloud providers load balancer.  
Node port and cluster IP services, to which the external load balancer routes or automatically created.
4. ExternalName/ExternalIP: maps the service to the content of external fields by written a C name records within a value.

```
apiVersion: v1
kind: Service
metadata:
  name: myjenkins
spec:
  selector:
    app: myapp
  ports:
    - name: port1
      protocol: TCP
      port: 30001
      targetPort: 8080
    - name: port2
      protocol: TCP
      port: 33000
      targetPort: 50000
  externalIPs:
    - 172.31.20.120
```

### Namespaces:

Namespace helps in creating a virtual cluster inside your Kubernetes cluster

Namespace helps you isolate your resources between different teams. Allocate resources or apply quota of resources

Advantages of namespace:

Helps in easy management of applications of each environment

Control multiple app with single name in a single cluster

Cost effective

### Default k8s namespaces are.

**Default:** adding an object to a cluster without providing a namespace will place it within the default namespace

**Kube-system:** kube system namespace is used for k8s components managed by k8s

**Kube-public:** kube public it is readable by everyone, but the namespace is reserved for system usage

Daemonset: it ensures that all nodes are running exactly one copy of a pod  
Daemonset will even create the pod on new nodes that are added to the cluster  
Daemonset tells Kubernetes to make sure there is one instance of the pod on nodes in your cluster

### Default limits:

We can limit the memory and CPU which container will be using.

Priority	Class Name	Description
1 (highest)	Guaranteed	If limits and optionally requests are set (not equal to 0) for all resources and they are equal.
2	Burstable	If requests and optionally limits are set (not equal to 0) for all resources, and they are not equal
3 (lowest)	BestEffort	If requests and limits are not set for any of the resources

**Taint and toleration:** these are work together to ensure that pods are not scheduled onto inappropriate nodes.

One or more taints are applied to a node, this marks that the node should not accept any pods that do not tolerate the taints.

Tolerations are applied to pods and allow the pods to schedule onto nodes with matching taints.

Taint and toleration are only meant to restrict the nodes to accept certain pods.

Taint and toleration don't tell a pod to go to a particular node, instead it tell the node to accept pods with certain tolerations.

**kubecttl taint nodes node1 key1=value1:NoSchedule**

### Node Selector:

NodeSelector is the simplest recommended form of node selection constraint. NodeSelector is a field of PodSpec. It specifies a map of key-value pairs. For the pod to be eligible to run on a node, the node must have each of the indicated key-value pairs as labels (it can have additional labels as well).

Step #1 - First label the node using kubecttl command

#kubecttl label nodes = #kubecttl label nodes node2.example.com size=large

Step #2: Now create a pod definition, specifying the node selector for the pod.

There are limitations. You cannot use the condition that "place the pod on node either "large" or "medium" but not "small".

You cannot use multiple checks. For that we use "affinity" and "anti-affinity"

### Affinity:

For a variety of reasons a service or container should only run on a specific type of hardware. Maybe

one machine has faster disk speeds, another is running a conflicting application, and yet another is part of your bare-metal cluster.

A “Hard rule” – Node Affinity Required During Scheduling Ignored during Execution, which means the rule is “required during scheduling” but has no effect on an already-running Pod.

A “Soft rule” – NodeAffinity Preferred during Scheduling Ignored during Execution, which means the rule is “preferred during scheduling” but likewise has no effect on an already-running pod.

Together, these rules are called NodeAffinity because they indicate a Pod’s “attraction” to certain Nodes

requiredDuringSchedulingIgnoredDuringExecution		
preferredDuringSchedulingIgnoredDuringExecution		
	DuringScheduling	DuringExecution
Type 1	Required	Ignored
Type 2	Preferred	Ignored

## ReplicaSet:

A replicaset is a set of pod templates that describes a set of pod replicas. It uses a template that describes what each pod must contain.

The replicaset ensures that a specified number of pod replicas are running at any time.

We want to replicate containers (and thereby applications) for several reasons, including:

**#1 - Reliability:** By having multiple versions of an application, you prevent problems if one or more fails. This is particularly true if the system replaces any containers that fail.

**#2 - Load balancing:** Having multiple versions of a container enables you to easily send traffic to different instances to prevent overloading of a single instance or node. This is something that Kubernetes does out of the box, making it extremely convenient.

**#3 - Scaling:** When load does become too much for the number of existing instances, Kubernetes enables you to easily scale up your application, adding additional instances as needed.

However, a Deployment is a higher-level concept that manages ReplicaSets and provides declarative updates to Pods along with a lot of other useful features.

Therefore, its recommended using Deployments instead of directly using ReplicaSets, unless you require custom update orchestration or don’t require updates at all.

### Deployment:

A deployment is an object in Kubernetes that lets you manage a set of identical pods. Without a deployment, you'd need to create, update, and delete a bunch of pods manually.

With a deployment, you declare a single object in a YAML file.

This object is responsible for creating the pods, making sure they stay up to date, and ensuring there are enough of them running.

### Rollingupdate:

Kubernetes deployments rollout pod version updates with a rolling update strategy.

This strategy aims to prevent application downtime by keeping atleast some instance up and running at any point intime while performing the updates.

Old pods are only shutdown after new pods of the new deployment version have started up and became ready to handle the traffic.

MaxSurge: it indicates maximum number of pods need to be created before terminating the old pods.

maxUnavailable: this indicates number of unavailable pods during rolling updates

### Persistent volume:

Piece of storage in the cluster that has been provisioned by the administrator or dynamically provisioned using storage classes. It is a resource in the cluster just like a node in the cluster resource.

PVs are volumes plugins like volumes, but have a life cycle independent of any individual pod that uses the PV.

### K8s persistent volume has the following attributes:

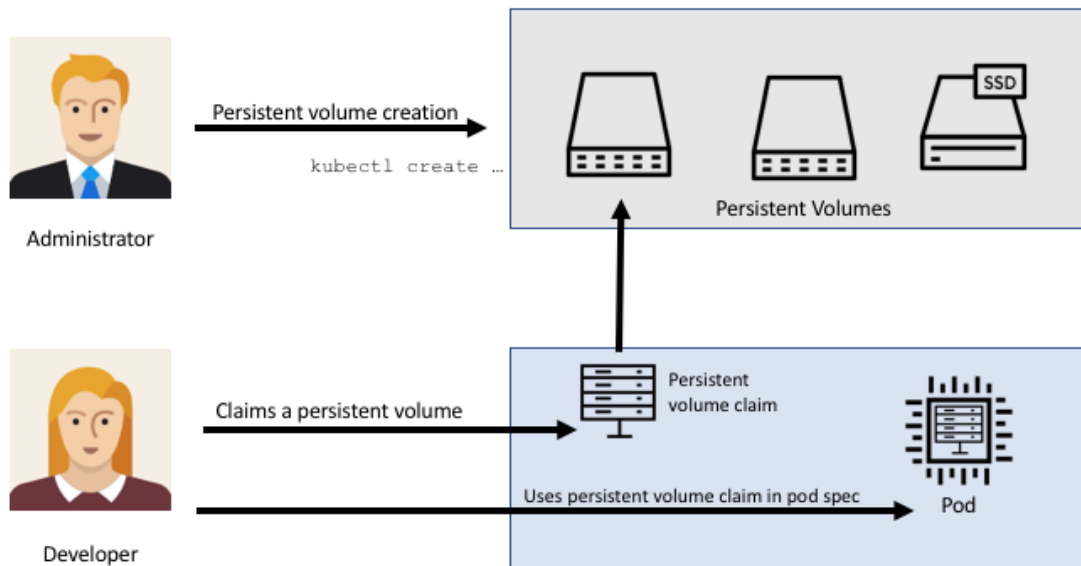
1. It is provisioned either dynamically or by an administrator.
2. Created with a particular filesystem.
3. Has a particular size.
4. Has identifying characteristics such as volume ids and a name.

### Persistent volume claim (PVC):

PVC is a request for storage by a user it is similar to Pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific level of requests (CPU & Memory). Claims can request specific size and access modes (example: they can be mounted once read/write or many times read/write).

Persistent volume claim describes the amount and characteristics of the storage required by the pod, finds any matching persistent volumes and claims this.

K8s persistent volume remains available outside of the pod life cycle this means that the volume will remain even after the pod is deleted. It is available to claim by another pod if required and the data is retained.



### Kubernetes Secrets:

A secret is an object that contains a small amount of sensitive data such as password, a token or a key.

To use a Secret, a Pod needs to reference the Secret. A Secret can be used with a Pod in three ways:

1. As files in a volume mounted on one or more of its containers.
2. As container environment variable.
3. By the kubelet when pulling images for the Pod.

ConfigMaps: Configmaps are similar to secret.

They can be created and shared in the containers in the same way. The only big difference between them is base64 encoding obfuscation.

Configmaps are intended for non-sensitive data, configuration data like config files and environment variables and are a great way to create customized running service from generic container images.

### StatefulSet:

It is introduced from 1.19 version of k8s

Statefulset is the workload api object used to manage stateful applications

Manages the deployment and scaling of a set of pods and provides guarantees about the ordering and uniqueness of these pods.

Like deployment, a statefulset manages pods that are based on an identical container spec. unlike deployment, a statefulset maintains a sticky identity for each of their pods. These pods are created from the same spec but are not interchangeable. Each has a persistent identifier that it maintains across any rescheduling.

StatefulSets are valuable for applications that require one or more of the following.

Stable, unique network identifiers.

Stable, persistent storage.

Ordered, graceful deployment and scaling.

Ordered, automated rolling updates.

Let's see the main differences between *Deployment* vs. *StatefulSet*.

Deployment	StatefulSet
Deployment is used to deploy stateless applications	<i>StatefulSets</i> is used to deploy stateful applications
Pods are interchangeable	Pods are not interchangeable. Each pod has a persistent identifier that it maintains across any rescheduling
Pod names are unique	Pod names are in sequential order
Service is required to interact with pods in a deployment	A Headless Service is responsible for the network identity of the pods
The specified <code>PersistentVolumeClaim</code> is shared by all pod replicas. In other words, shared volume	The specified <code>volumeClaimTemplates</code> so that each replica pod gets a unique <code>PersistentVolumeClaim</code> associated with it. In other words, no shared volume

### Kubernetes networking:

There are 5 essential things to understand about networking in Kubernetes

1. Communication between containers on same pod
2. Communication between pods on same node
3. Communication between pods on different nodes
4. Communication between pods and services
5. How does DNS work and how do we discover IP addresses?

#### What is network namespace:

It is a collection of network interfaces (connection between two pieces of equipment on a network) and routing tables (instruction for where to send a network traffic)

**There is a secret container that runs on every pod in k8s.**

These containers first job is to keep the namespace open in case all the other containers or pods die. Its called pause containers.

Every pod has a unique IP.

This pod IP is shared by all containers in this pod, and its route table from all the other pods.

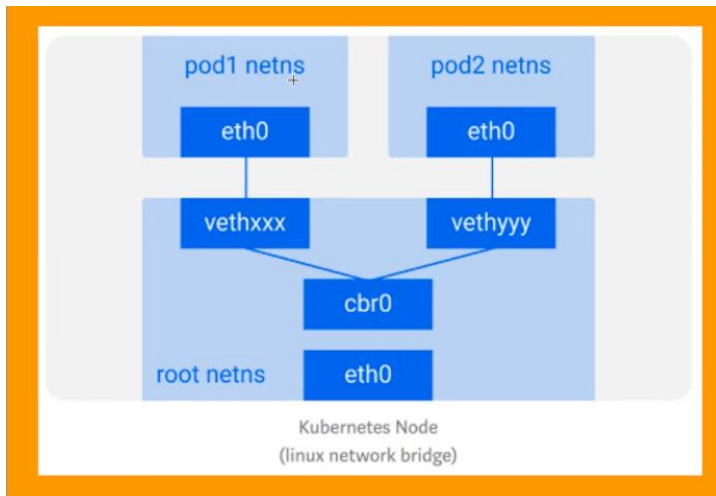
Pause containers are called sandbox containers, whose only job is to reserve and hold a network namespace which is shared by all the containers in a pod. This way, a pod IP doesn't change a container dies and a new container is created in a space.

A huge benefits of the IP per pod is there are no IP of port collision with the underlying ports and we don't have to worry about what port the application use.

#### Communication between containers on same pod (intranode communication)

On every k8s node, which is a Linux machine, there is a root network namespace- root netns. The main network interfaces eth0 is root netns.

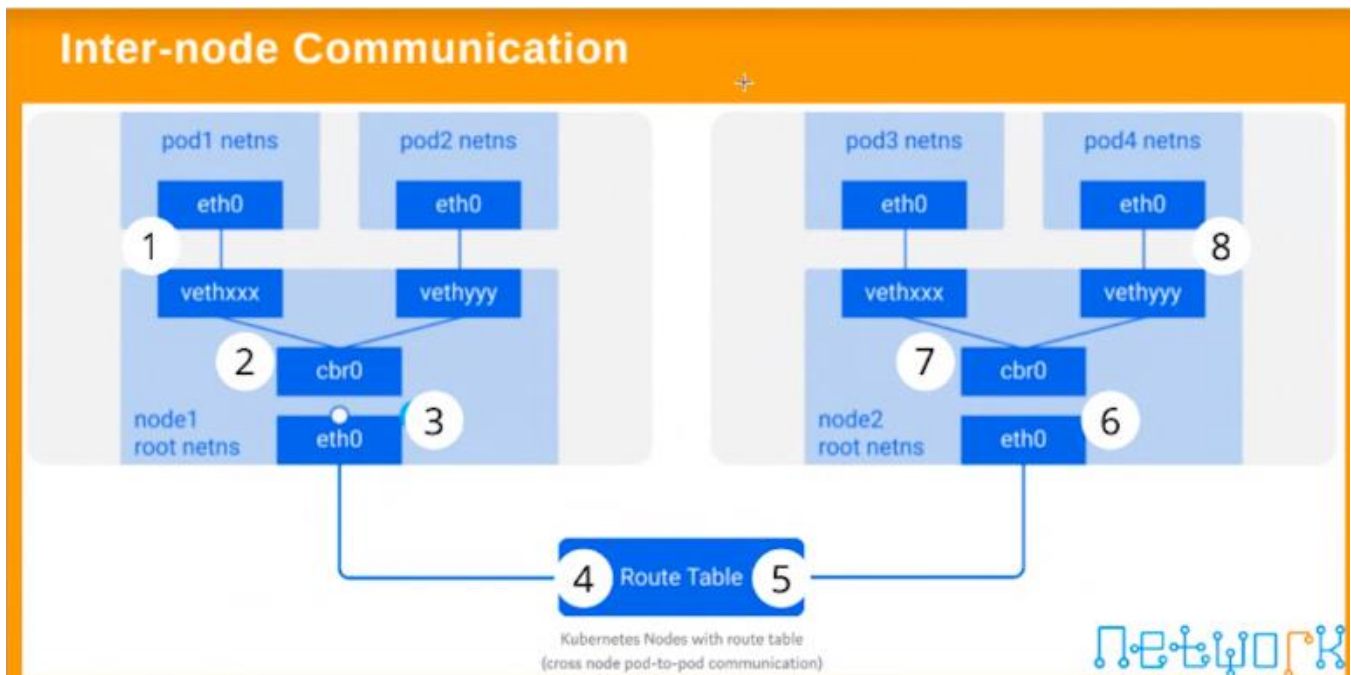
Similarly, each pod has its own netns with a virtual ethernet pair connecting it to the root netns.



Assume a packet is going from pod1 to pod2.

1. It leaves pod1's netns at eth0 and enters the root netns at vethxxx.
2. It's passed on to cbr0, which discovers the destination using an ARP request, saying "who has this IP?"
3. vethyyy says it has that IP, so the bridge knows where to forward the packet.
4. The packet reaches vethyyy, crosses the pipe-pair and reaches pod2's netns.

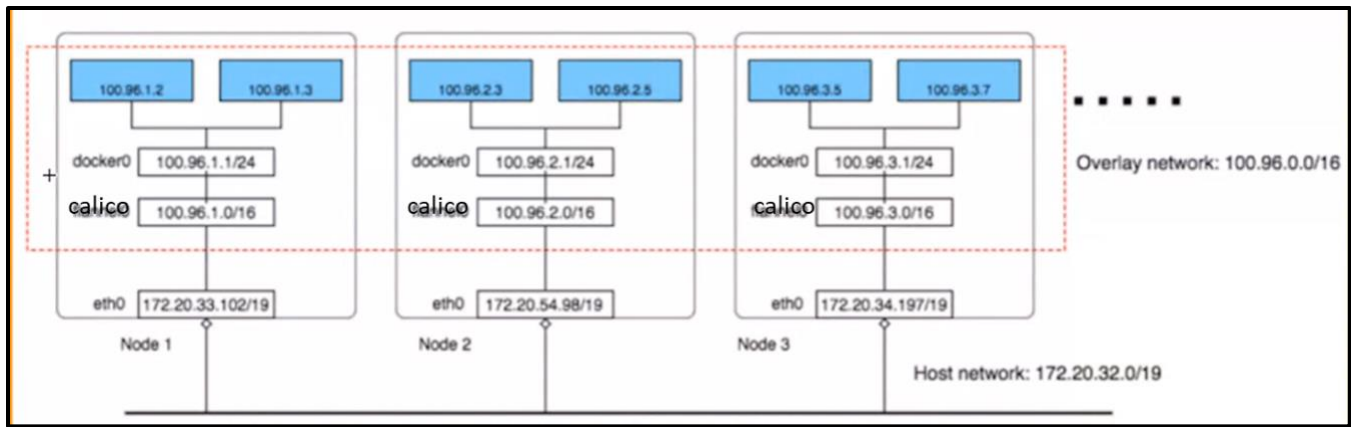
**Inter-node communication:**



**Communication between pods and services:**

Overlay network:





Kubernetes does not provide any default network implementation, rather it only defines the model and leaves to other tools to implement it.

There are many implementations nowadays, calico is one of them and one of the simplest.

Overlay networks are not required by default, however, they help in specific situations. Like when we don't have enough IP space, or network can't handle the extra routes or when we want some extra management features the overlays provide.

There are three networks in this cluster:

**Host network:** all instances are in one VPC subnet 172.20.32.0/19. They have been assigned ipaddresses in this range, all hosts can connect to each other because they are in same LAN.

**Flannel/calico overlay network:** flannel has created another network 100.96.0.0/16, it's a bigger network which can hold 216(65536) addresses, and it's across all Kubernetes nodes, each pod will be assigned one address in this range, later we will see how flannel achieves this.

**In-Host docker network:** inside each host, flannel assigned a 100.96.x.0/24 network to all pods in this host, it can hold 28(256) addresses. The docker bridge interface docker0 will use this network to create new containers.

## Liveliness

Liveness and Readiness probes are used to control the health of an application running inside a Pod's container. Both are very similar in functionality, and usage.

### Readiness probe:

In some cases, we would like our applications to be alive but not serve the traffic unless some conditions are met. In such cases we use readiness probe if the condition inside readiness probe passes, then only our applications can serve the traffic.

Readiness probe runs on the container during its whole life cycle.

Note: liveness probe doesn't wait for readiness probe to succeed. If you want to wait before executing a liveness probe you should use a initial delay seconds.

initialDelaySeconds: 3

This is the delay which tells kubelet to wait for 3 seconds before performing the first probe

periodSeconds: 5

This field specifies that kubelet should perform a probe every 5 seconds.

```
livenessProbe:
  httpGet:
    path: /healthz
    port: 8080
  initialDelaySeconds: 3
  periodSeconds: 3
```

```
readinessProbe:
  exec:
    command:
      - cat
      - /tmp/healthy
  initialDelaySeconds: 5
  periodSeconds: 5
```

Both liveness and readiness probe are used to control the health of the application. Failing liveness probe will restart the container, whereas failing readiness probe will stop our application from serving the traffic.

**ECR (Elastic container registry)- it's a AWS Service**

**What is Ingress?**

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.



An Ingress does not expose arbitrary ports or protocols. Exposing services other than HTTP and HTTPS to the internet typically uses a service of type `Service.Type=NodePort` or `Service.Type=LoadBalancer`.

## Helm:

In simple terms, helm is a package manager for Kubernetes. Helm is the k8s equivalent of yum or apt.

Helm deploy charts, which you can think of as a packaged application. It is a collection of all your versioned, preconfigured application resources which can be deployed as one unit.

Helm helps in three keyways:

1. Improves productivity
2. Reduces the complexity of deployment of microservices.
3. Enables the adoption of cloud native application.

Helm charts are simply k8s yaml manifest combined into a single package that can be advertised to your k8s cluster.

Helm -There are 3 concepts we need to get familiar with

1. Chart: a package of preconfigured k8s resources.
2. Release: a specific instance of a chart which has been deployed to the cluster using helm.
3. Repository: a group of published charts which can be made available to others.

**How to create helm:** helm create (applicationname)

Chart.yaml: this is where you put all the information about the chart your packaging. So for example your version number, name of the application etc.. this is where you put all the details.

Charts: this is where you store other charts that your charts depend on. You might be calling another chart that your chart needs to functions properly.

Templates: this folder is where you put the actual manifest (yam file) you are deploying with the chart. For example, you might be deploying nginx deployment that needs a service, config map and secrets. You will have your deployment. yaml, service.yaml and secret.yaml all in the template directory this will all get their values from values.yaml file.

Values.yaml: this is where you define all the values you want to inject into your templates.

**ANSIBLE:** Ansible is a configuration tool and also, agent less such that it requires nothing to be installed on target host machine except SSH and Python. It is push based methodology.

To check the ansible master is connected to the server: ansible group-name or ip address -m ping

## Difference between chef and ansible

Chef	Ansible
Chef is client server architecture	pushes based architecture
chef works with ruby	it works with python
we need to register each and every host individually	it supports dynamic inventory
chef is more secure than ansible	it is not secured by default, but we can use external module called vault
chef is not easy to setup	Ansible is easy to set up

Note: Raw Module:

Raw module is also called as dirty module. If there is no python in target machine, we use raw module to install python or run some shell commands

### **Ansible inventory:**

**Static inventory:** it is just a file containing list of ip address of target host machine and group it together based on the user requirements in “ini” format. The default location of the inventory file is /etc/ansible/host

**Dynamic inventory:** cloud itself will provide the files script and ini file. If we run script it will automatically fetch ip address of target machine and store in a ini file

Ansible Modules:

Ping module: it is used to check connection of host machine on inventory it will get reply with pong.

Ansible ipaddress -m ping or ansible ipaddress groupname -m ping.

**Package module:** it is same as yum in centos or redhat & apt in ubuntu (Debian)

Repository path in Linux: ubuntu /etc/apt/sources.list.d for yum: /etc/yum/repos.d

State: Ansible behave as a hydopotential behaviour

1. state: present: it will check for the package installed or not. If it is not installed, then package module is executed
2. State: latest: it will check whether the package is installed or not if it is installed it will ensure upto date or it will be installed with the latest version.
3. State: remove or State: absent: it will uninstall the installed package.

### **Privilege access (become module):**

If I want to run with root permission or user with root permission, then we are going to use become module

become: yes – to run with root permission

become\_usr: username (ex: become\_usr: ubuntu) – to run the commands as a specific user

```

---
- hosts: all
  tasks:
    - name: update repository
      apt:
        update_cache: yes
    - name: install packages
      apt:
        name: git
        state: latest
        become: yes
    - name: install packages
      apt:
        name: wget
        state: present
        become: yes

```

```

1 ---
2 - hosts: all
3   tasks:
4     - name: install multiple packages
5       apt:
6         name: "{{ item }}"
7         state: latest
8       with_items:
9         - wget
10        - git

```

**Copy module:** it is used to copy a file from the host machine to the multiple target machines.

Syntax:

copy:

src: path of a file

dest: path of a destination

File Module: how to change the permission of a file.

File:

Path: path of a file

Mode: 0655

Group: username

Owner: username

Ansible vault: allows us to keep sensitive data such as password or keys in encrypted files rather than plain text in playbooks or rules.

Commands:

Create: it is used to create ansible vault file in the encrypted format “Ansible-vault create filename (yaml file)”

View: it is used to view the data of encrypted files “ansible-vault view filename”

Encrypt: used to encrypt the unencrypted files “ansible-vault encrypt filename”

Decrypt: used to decrypt an encrypted file “ansible-vault decrypt filename”

--ask-vault-pass: while running playbooks if I need to provide passwords then we will use this.

Ansible-playbook filename --ask-vault-pass

--vault-password-file: used to pass a password through a file

Ansible-playbook filename --vault-password-file filename (file which contains password).

Ansible-galaxy:

It is a command used to create and manage the roles.

It is a public repository of ansible roles.

Ansible-galaxy list: it displays a list of installed roles with version number

Ansible galaxy remove role\_name: it will remove the installed roles

Ansible galaxy info: it will provide information about the ansible galaxy

Ansible galaxy init rolename: it will create a role

**Roles:** it will split the single playbook into multiple files. Ansible roles is defined with 8 directories and 8 files

## Terraform

IAC (Infrastructure as a code): IAC is a widespread terminology among Devops professionals.

It is the process of managing and provisioning the complete IT infrastructure (comprises both physical and virtual machines) using machine readable definition files.

It is a software engineering approach towards operation.

It helps in automating the complete data entry by using programming scripts.

Infrastructure as a code has multiple challenges

1. Need to learn to code.
2. Don't know the change impact

3. Need to revert the change.
4. Cannot track changes.
5. Cannot automate a resource
6. Multiple environments for infrastructure.

Terraform has been created to solve these challenges.

What is terraform?

Terraform is an open-source infrastructure as code software tool developed by HashiCorp. It is used to define and provision the complete infrastructure using easy to learn declarative languages.

It is an infrastructure provisioning tool where you can store your cloud infrastructure set up as code. It is very similar to tools such as cloud formation, which would use to automate your aws infrastructure, but you can only use that on AWS. With terraform you can use it on other cloud platform as well.

Benefits of using terraform:

1. Does orchestration, not just configuration management.
2. Supports multiple providers such as AWS, Azure, GCP, digital motions and etc.
3. Provides immutable infrastructure where configuration changes smoothly
4. Uses easy to understand language HCL (HashiCorp configuration language)
5. Easily portable to any other provider.
6. Supports client only architecture. So, no need of additional configuration management on a server.

#### Terraform core concepts:

Below are the core concepts/terminologies used in Terraform:

**Variables:** Also used as input-variables, it is key-value pair used by Terraform modules to allow customization.

**Provider:** It is a plugin to interact with APIs of service and access its related resources.

**Module:** It is a folder with Terraform templates where all the configurations are defined

**State:** It consists of cached information about the infrastructure managed by Terraform and the related configurations.

**Resources:** It refers to a block of one or more infrastructure objects (compute instances, virtual networks, etc.), which are used in configuring and managing the infrastructure.

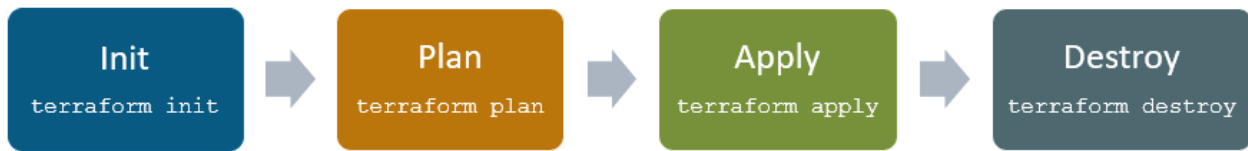
**Data Source:** It is implemented by providers to return information on external objects to terraform.

**Output Values:** These are return values of a terraform module that can be used by other configurations.

**Plan:** It is one of the stages where it determines what needs to be created, updated, or destroyed to move from real/current state of the infrastructure to the desired state.

**Apply:** It is one of the stages where it applies the changes real/current state of the infrastructure in order to move to the desired state.

#### Terraform Life Cycle:



**Terraform init** which initializes the working directory which consists of all the configuration files.

**Terraform Plan** is used to create and execute the plan to reach desired state of the infrastructure. changes in the configuration files are done in order to achieve the desired state.

**Terraform Apply** then makes the changes in the infrastructure as defined in the plan, and the infrastructure comes to the desired state.

**Terraform destroy** is used to delete all the old infrastructure resources which are marked tainted after the apply phase.

How terraform works:



# Amazon Web Service

**Aws:** AWS services is amazon's cloud web hosting platform that offers flexible, reliable, scalable, easy to use and cost effective solutions.

**Cloud computing:** is a term refer to storing and accessing the data over the internet

**There are 3 types of cloud services:**

1. Private cloud: it is dedicated to single **tenant**. It is dedicated in terms of hardware and security.
2. Public cloud: shared with multiple tenants and cost is lesser. Ex: aws, GCP (google), Oracle, Microsoft Azure, Alibaba cloud etc...
3. Hybrid cloud: hybrid is a combination of both public and private cloud. It is the most successful cloud practice. Example: Openstack & VMware

What is the reason behind cloud? Answer: Virtualization

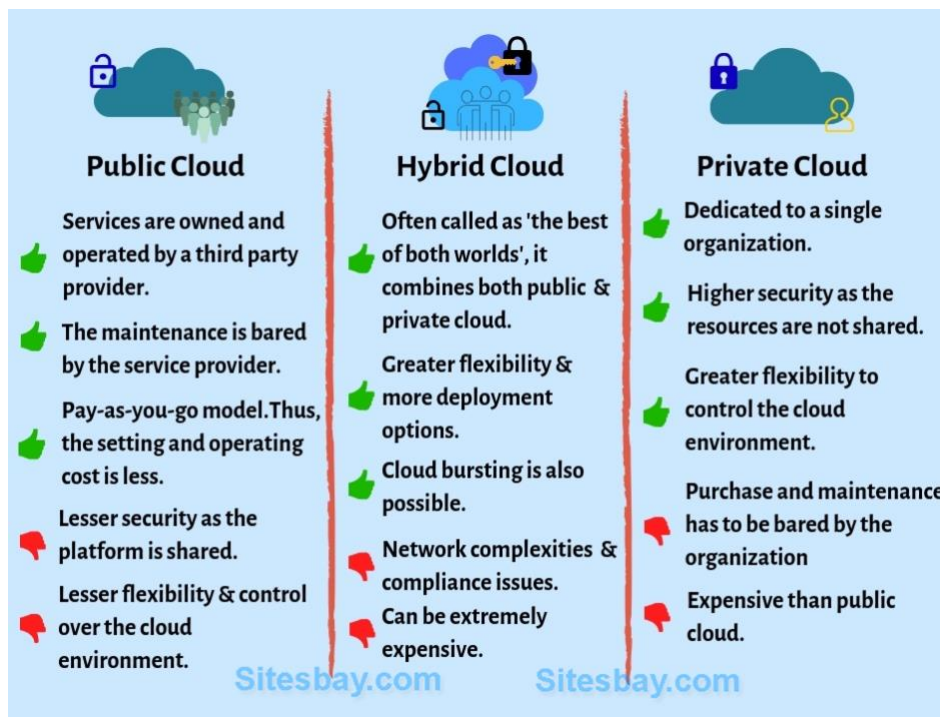
**Virtualization** will transfer hardware into software example: VMware

**Private cloud:** is having dedicated to a single organization, highly secured and Greater flexibility.

**Public cloud:** Third party provider makes resources and services available to the customer via  
Internet

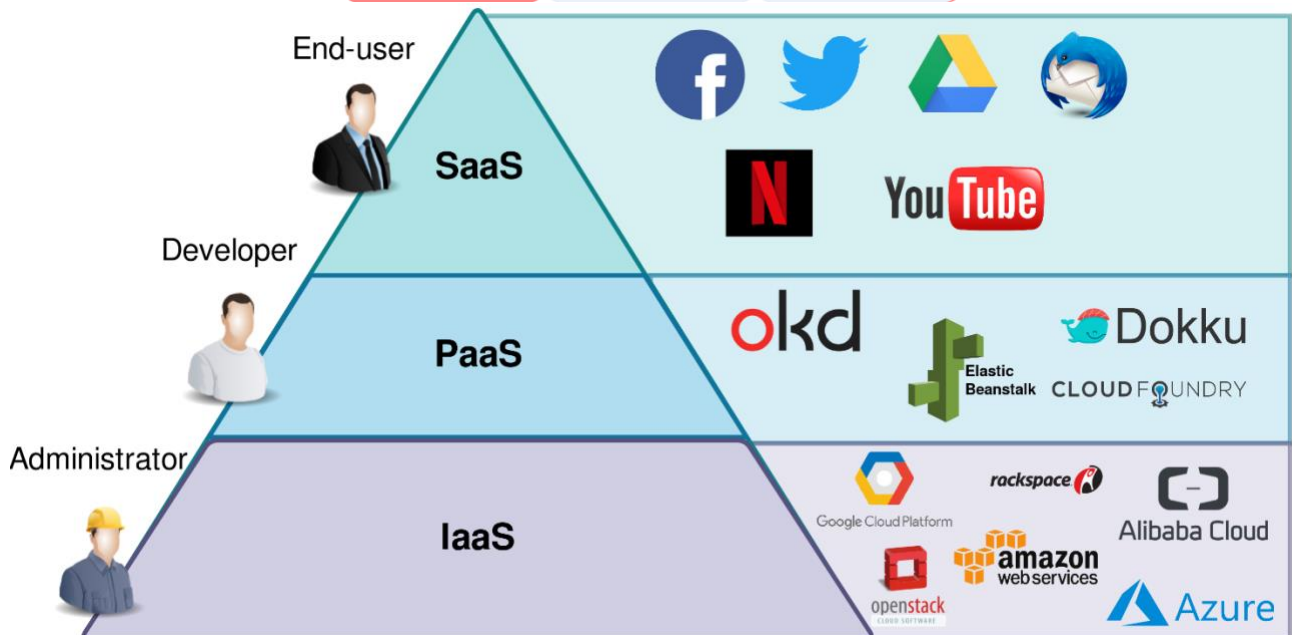
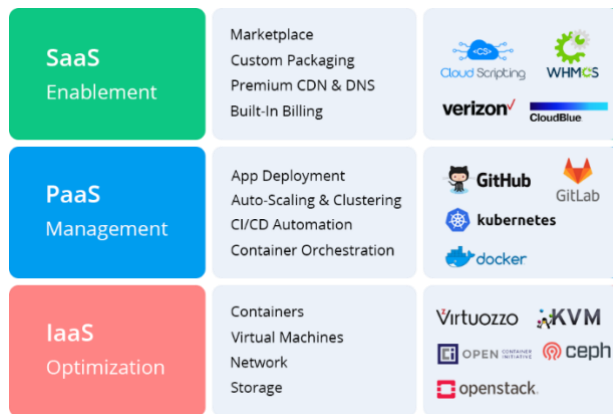
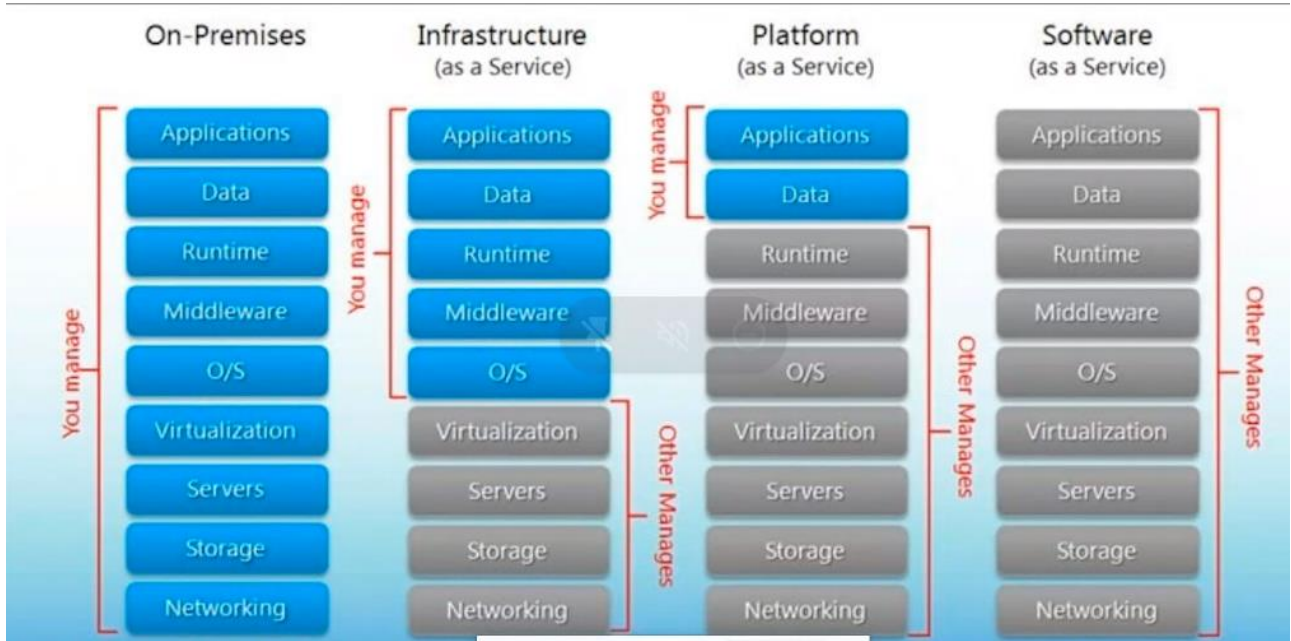
Advantages: cost effective, reliability, unlimited storage, backup & recovery

Third party providers: aws, GCP (google), Oracle, Microsoft Azure etc...



## 3 types of services

1. IaaS=Infra structure as a service
2. PaaS = Platform as a service
3. SaaS= Software as a service



### **Advantages of IAS:**

1. Shared infrastructure
2. Pay as per you use model
3. Focus on core business
4. On demand scalability

### **Disadvantage of IAS:**

1. Security
2. Maintenance & upgrade

### **PAS**

Ex: Google app engine, salesforce, windows azure etc

### **Advantages of PAS:**

1. Simplified Development
2. Lower risk
3. Scalability

### **Disadvantages of PAS:**

1. Vendor locking /flexibility
2. Integrating with rest of the applications

### **SAS**

Example: Google, Microsoft office 365

### **Advantages of SAS:**

1. Reduced time to benefit
2. Lower costs
3. Scalability and integration
4. Trouble-free Upgradation
5. Easy to use and perform proof-of-concepts

### **Disadvantages**

1. Insufficient Data Security
2. Difficulty with Regulations Compliance
3. Cumbersome Data Mobility
4. Low Performance

### **Ip address:**

Class A : 1.X.X.X TO 126.X.X.X

Class B : 128.X.X.X TO 191.X.X.X

Class C: 192.X.X.X TO 223.X.X.X

Class D: 224.X.X.X TO 239.X.X.X

Class E: 240.X.X.X TO 254.X.X.X

127.0.0.0 is called loopback ip (it is reserved)

### IP Header Classes:

Class	Address Range	Subnet masking	Example IP	Leading bits	Max number of networks	Application
IP Class A	1 to 126	255.0.0.0	1.1.1.1	8	128	Used for large number of hosts.
IP Class B	128 to 191	255.255.0.0	128.1.1.1	16	16384	Used for medium size network.
IP Class C	192 to 223	255.255.255.0	192.1.1.1	24	2097152	Used for local area network.
IP Class D	224 to 239	NA	NA	NA	NA	Reserve for multi-tasking.
IP Class E	240 to 254	NA	NA	NA	NA	This class is reserved for research and Development Purposes.

## Subnet Mask Hierarchy

Subnet Mask	CIDR	Binary Notation	Available Addresses Per Subnet
255.255.255.255	/32	11111111.11111111.11111111.11111111	1
255.255.255.254	/31	11111111.11111111.11111111.11111110	2
255.255.255.252	/30	11111111.11111111.11111111.11111100	4
255.255.255.248	/29	11111111.11111111.11111111.11111000	8
255.255.255.240	/28	11111111.11111111.11111111.11110000	16
255.255.255.224	/27	11111111.11111111.11111111.11100000	32
255.255.255.192	/26	11111111.11111111.11111111.11000000	64
255.255.255.128	/25	11111111.11111111.11111111.10000000	128
255.255.255.0	/24	11111111.11111111.11111111.00000000	256
255.255.254.0	/23	11111111.11111111.11111110.00000000	512
255.255.252.0	/22	11111111.11111111.11111100.00000000	1024
255.255.248.0	/21	11111111.11111111.11111000.00000000	2048
255.255.240.0	/20	11111111.11111111.11110000.00000000	4096
255.255.224.0	/19	11111111.11111111.11100000.00000000	8192
255.255.192.0	/18	11111111.11111111.11000000.00000000	16384
255.255.128.0	/17	11111111.11111111.10000000.00000000	32768
255.255.0.0	/16	11111111.11111111.00000000.00000000	65536
255.254.0.0	/15	11111111.11111110.00000000.00000000	131072
255.252.0.0	/14	11111111.11111100.00000000.00000000	262144
255.248.0.0	/13	11111111.11111000.00000000.00000000	524288
255.240.0.0	/12	11111111.11110000.00000000.00000000	1048576
255.224.0.0	/11	11111111.11100000.00000000.00000000	2097152
255.192.0.0	/10	11111111.11000000.00000000.00000000	4194304
255.128.0.0	/9	11111111.10000000.00000000.00000000	8388608
255.0.0.0	/8	11111111.00000000.00000000.00000000	16777216

## Subnet Blocks

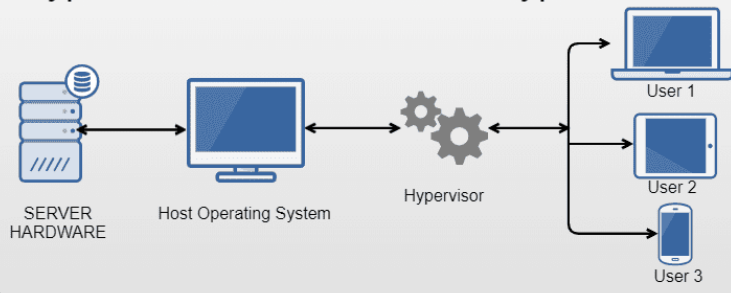
Binary	Decimal
$2^8-2^0$	255
$2^8-2^1$	254
$2^8-2^2$	252
$2^8-2^3$	248
$2^8-2^4$	240
$2^8-2^5$	224
$2^8-2^6$	192
$2^8-2^7$	128

Number of valid hosts is always two less than the subnet block

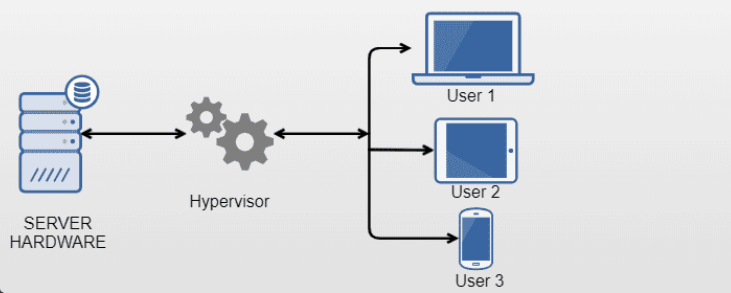
CIDR Notation	Host Formula	Available Hosts
/8	$2^{32-8} - 2$	16,777,214
/9	$2^{32-9} - 2$	8,388,606
/10	$2^{32-10} - 2$	4,194,302
/11	$2^{32-11} - 2$	2,097,150
/12	$2^{32-12} - 2$	1,048,574
/13	$2^{32-13} - 2$	524,286
/14	$2^{32-14} - 2$	262,142
/15	$2^{32-15} - 2$	131,070
/16	$2^{32-16} - 2$	65,534
/17	$2^{32-17} - 2$	32,766
/18	$2^{32-18} - 2$	16,382
/19	$2^{32-19} - 2$	8,190
/20	$2^{32-20} - 2$	4,094
/21	$2^{32-21} - 2$	2,046
/22	$2^{32-22} - 2$	1,022
/23	$2^{32-23} - 2$	510
/24	$2^{32-24} - 2$	254
/25	$2^{32-25} - 2$	126
/26	$2^{32-26} - 2$	62
/27	$2^{32-27} - 2$	30
/28	$2^{32-28} - 2$	14
/29	$2^{32-29} - 2$	6
/30	$2^{32-30} - 2$	2

Number of 1's in Octet	Subnet Mask
<b>1</b>	<b>128</b>
<b>2</b>	<b>192</b>
<b>3</b>	<b>224</b>
<b>4</b>	<b>240</b>
<b>5</b>	<b>248</b>
<b>6</b>	<b>252</b>
<b>7</b>	<b>254</b>
<b>8</b>	<b>255</b>

### Type I / Bare-metal / Native Hypervisor

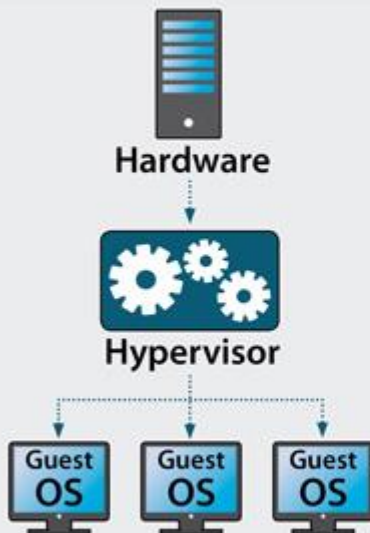


### Type II / Embedded / Hosted Hypervisor

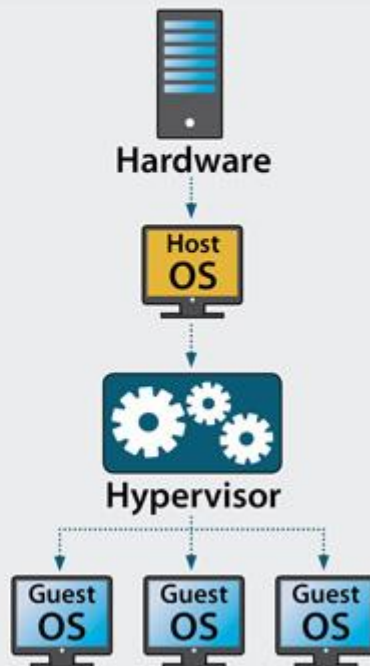


## Hypervisor or Virtual Machine Monitor (VMM)

### Type 1 Native (bare metal)



### Type 2 Hosted





1) **What is VPC (virtual private cloud)?**

It is a virtual network dedicated to your AWS account, it logically isolates from other virtual networks in the AWS cloud, and where you can launch your AWS instance.

VPC consist of subnets, network gateway & Routing table.

**What is a subnet?**

Subnet is a logical subdivision of IP network. The practice of dividing a network into 2 or more networks is called sub netting.

2) **What is Route table?**

A set of rules called routes that are used to determine where network traffic is directed.

3) **What is Internet gateway?**

A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet.

4) **What is NAT gateway?**

NAT Gateway is used to connect to the Internet from instances within a private subnet in the VPC.

Nat gateway will be created from Public subnet and attach to the private subnet.

5) **What is VPC endpoint?**

It enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by private link without requiring an internet gateway, NAT device, VPN connection or AWS direct connection.

6) **What is VPC peering connection?**

A VPC peering connection is a network connection between 2 VPC'S that enables you to route traffic between them using private IPV4 or IPV6 address.

Instances in either VPC can communicate with each other as if they are within the same network.

You can create a VPC peering connection between your own VPC or with a VPC in another AWS account.

7) **Limitations of VPC peering.**

- I. You cannot create a VPC peering connection between VPC'S that have matching or overlapping IPV4 CIDR blocks. Amazon always assigns a unique IPV6 CIDR block.
- II. You have a quota on the number of active and pending VPC peering connections that you can have per VPC.
- III. VPC peering does not support transitive peering relationships. In VPC peering connection, one VPC does not have access to any other VPC with which the peer VPC may be peered.
- IV. Cannot have more than 1 VPC peering connection between the same 2 VPC at the same time.

8) **What is NACL (Network ACL)? Network Access Control Lists**

NACL are firewall at the subnet level.

You can use a network ACL to control the traffic to and from the subnet in which NAT gateway is located.

9) **What is a security group?**

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign up to 5 security groups to the instance. Security groups act at instance level and not at subnet level.

10) **Difference between public subnet and private subnet**

Public Subnet – Users can access resources from the internet. Internet traffic is routed via internet gateway. Applications are stored in public subnet.

Private Subnet - Users cannot access resources from the internet. Internet traffic is routed via NAT gateway. Data is stored in private subnet (database, API calls, passwords)

AWS has 25 regions

80 availability zones(az)

230+ edge locations (network boosters/content delivery networks)

12 edge caches => regional cache

AMI => Amazon Machine Image : Master image for the creation of virtual servers called EC2.

I



## AWS Instance classes :

- 1) General Purpose => It is used for all kind of basic testing and lower environments. Ex: T2, T3, M6
- 2) Compute Optimized=> It is used for all kind of gaming servers and most of the machine learning servers. Ex:C6 C5.
- 3) Memory Optimized => It is used in terms of faster perf and ex: R6 series.
- 4) Accelerated Computing => high graphic processors.  
Basically used in handling gaming s/w. Ex: P3, P2 series.
- 5) Storage Optimized=> used in terms of handling heavy workloads and also for archiving huge data. Ex: L series

Security groups : firewall at a instances level. Are statefull. we can define inbound and outbound rules.

---

NACL: firewall at a subnet level. Stateless. Define only inbound rules.

## Configuration of EC2 Models:

### ▼ Instances

**Instances** New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances New

Dedicated Hosts

Capacity Reservations

## Types of EC2 payment models:

1. On demand or capacity reservation:
  - a. These instances work as pay as you go model.
  - b. Long time commitment is not required for this instance.
  - c. These are bit costlier
2. Spot instances:
  - a. These are called as bidding instances.
  - b. We can bid these instances as per the requirement.
3. Dedicated hosts:
  - a. Basically, dedicated instances are provided with the hardware configurations.
4. Reserved instances:
  - a. These are utilized for the longer time frame.
  - b. These would be having discounts in terms of pricing because of longer utilization time frame.
  - c. Will get to know platform, tenancy, instance type and payment options for reserved instances.

## AWS Storage:

1. Elastic Block storage (EBS)
2. Elastic file storage (EFS)
3. Simple Storage Service (S3)

## 2. **EFS (Elastic File System):**

It is similar to shared disk, EFS is more used in case of sharing the disk space. EFS storage is used in cluster management in order to have availability.

Storage space sharing is possible in EFS and not possible in EBS

Eg – Cassandra cluster, Kubernetes, Machine learning and AWS lambda.

### **Advantages of EFS:**

cost effective, Speed, Disk share.

## 3. **EBS (Elastic Block Storage):**

EBS provides simple, scalable, high available block storage. EBS can only attach to one EC2 system. It is a block storage.

Storage space sharing is not possible in EBS

### **Important 3 models of EBS:**

1. Provision IOPS (64000)
2. General purpose (16000)
3. Magnetic (5000)

**Note: EBS can be attached to only one instance, EBS and Volume should be there in same region to avail the EBS**

### **Benefits of EBS:**

2. SSD storage technology (solid-state drive)
3. Highly available, fast and scalable

## 4. **S3 (Simple Storage Service) :**

It is basically a object storage. S3 cannot hold data, but it can store & hold data which are in form of objects. S3 is not region specific. We can host static website on S3.

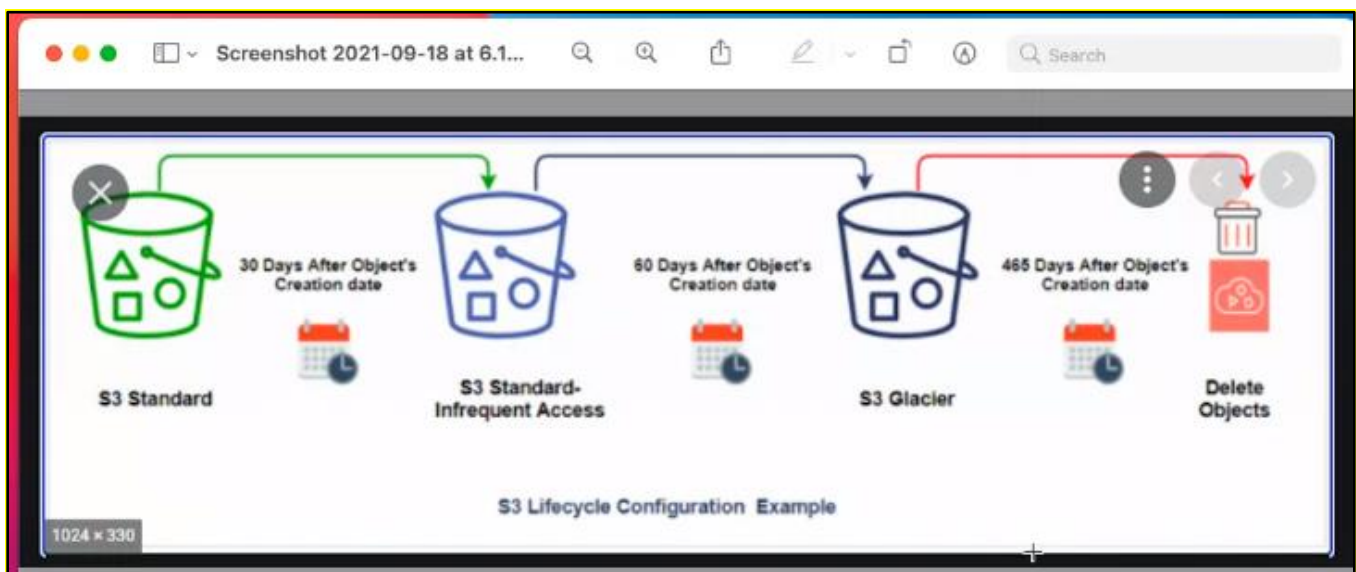
Types of S3 bucket -

- a) S3 Standard\_ia
- b) S3 Standard
- c) S3 intelligent tiering
- d) S3 Glacier
- e) S3 Deep archive

## 5. **Key Features of S3:**

- a. Versioning – AWS S3 is a means of keeping multiple variants of an object in same bucket, you can use the S3 versioning feature to preserve, retrieve and restore every

- version of every object stored in your bucket. Versioning enabled buckets can help you recover objects from accidental deletion or overwrite.
- b. Life cycle management.
  - c. Encryption
  - d. Multifactor authentication for object deletion.
6. S3 bucket Security:
- There are two types of bucket security:
- a. Bucket Policies – Json based scripts which are embedded in IAM policies of AWS which can be utilized for S3 bucket security.
  - b. Access control List.



## S3 Classes

- ▶ **S3 Standard**—The default storage class. If you don't specify the storage class when you upload an object, Amazon S3 assigns the S3 Standard storage class
- ▶ **S3 Intelligent-Tiering** is an Amazon S3 storage class designed to optimize storage costs by automatically moving data to the most cost-effective access tier, without operational overhead. It is the only cloud storage that delivers automatic cost savings by moving data on a granular object level between access tiers when access patterns change. S3 Intelligent-Tiering is the perfect storage class when you want to optimize storage costs for data that has unknown or changing access patterns. There are no retrieval fees for S3 Intelligent-Tiering.

## S3 Standard-IA

- ▶ The **S3 Standard-IA** and **S3 One Zone-IA** storage classes are designed for long-lived and infrequently accessed data. (IA stands for *infrequent access*.) S3 Standard-IA and S3 One Zone-IA objects are available for millisecond access (same as the S3 Standard storage class). Amazon S3 charges a retrieval fee for these objects, so they are most suitable for infrequently accessed data

## S3 glacier

- ▶ The **S3 Glacier** and **S3 Glacier Deep Archive** storage classes are designed for low-cost data archiving. These storage classes offer the same durability and resiliency as the S3 Standard storage class

### **What is partitions? How many AWS partitions are there? \*\***

A Partition is a group of AWS Region and Service objects.

You can use a partition to determine what services are available in a region, or what regions a service is available in.

AWS accounts are scoped to a single partition. You can get a partition by name. Valid partition names include:

1. "aws" - Public AWS partition
2. "aws-cn" - AWS China
3. "aws-us-gov" - AWS GovCloud
4. "AWS-ISO,
5. "AWS-ISO-b"

The last two are only for Secret and Top-Secret US Government data.

### **Define Auto-scaling?**

Auto-scaling is an activity that lets you dispatch advanced instances on demand.

Moreover, auto-scaling helps you to increase or decrease resource capacity according to the application.

### **Can you illustrate the relationship between an instance and AMI?**

With the help of just a single AMI, you can launch multiple instances and that to even different types.

At the same time, an instance type is characterized by the host

### **What is a default storage class in S3?**

The standard frequency accessed is the default storage class in S3.

### **What is the standard size of an S3 bucket?**

The maximum size of an S3 bucket is five terabytes

### **Is Amazon S3 an international/Global service?**

Yes. Amazon S3 is an international service.

Its main objective is to provide an object storage facility through the web interface, and it utilizes the Amazon scalable storage infrastructure to function its global network.

### **Can you name some AWS services that are not region-specific?**

- IAM
- Route 53
- S3
- Web application firewall
- CloudFront

### **Can you define EIP?**

EIP stands for Elastic IP address.

It is a static Ipv4 address that is provided by AWS to administer dynamic cloud computing services.

### **IAM (Identity Access management) :**

IAM allows you to manage users and their level of access to AWS console.

#### **Key components of IAM**

1) **Users** – Users are end users within an organization.

eg – developers, tester and infrastructure ppl.

2) **User group** – User group are collection of user, each user in the group will inherit the permission of the group.

3) **Policies** – Policies are made up of documents called policy document, these documents are in the format of json and they give permission as to what a user group or a role is able to do.

4) **IAM role** – It is an IAM entity that defines a set of permission making AWS service request. It is not associated with specific user or groups.

#### **Advantages of IAM**

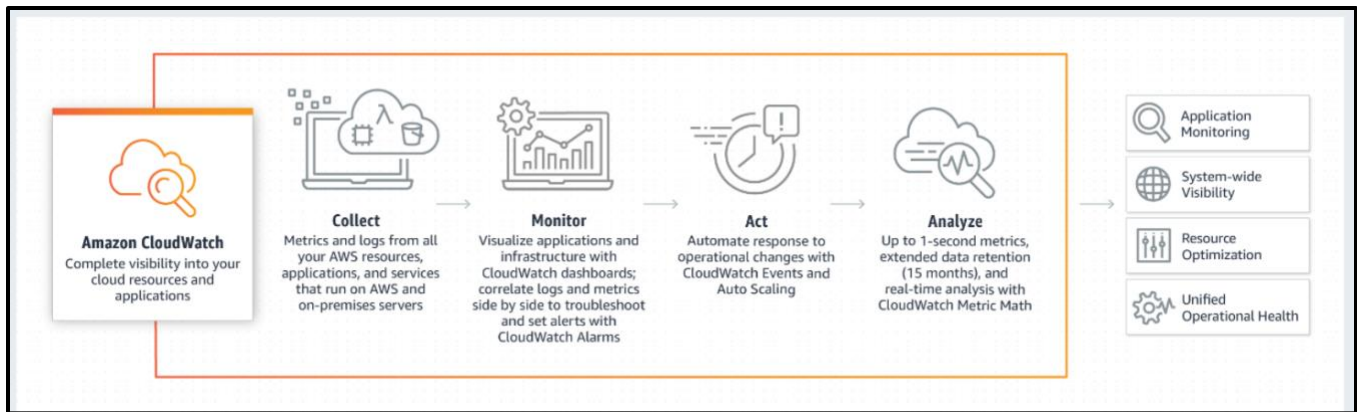
It provides centralized control of your AWS account.

Shared access to your AWS account.

Multi factor authentication.

Identity federation

#### **Cloud watch:**



Amazon cloud watch is monitoring and observability service build for all the application team members.

Cloud watch collects monitoring and operational data in the form of log, matrix and events.

Cloud watch is useful in setting up alarms, visualized log and matrix side by side.

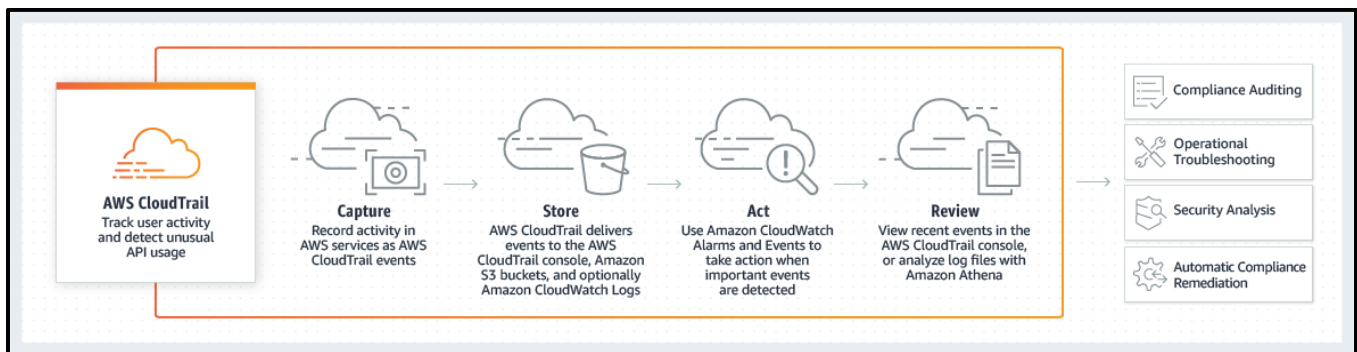
### SNS (simple notification service):

Amazon SNS is fully managed messaging service for both application to person and application to application communication.

Use cases:

1. Send messages directly to millions of users
2. Reliably deliver messages
3. Automatically scaling workload.

### Clod Trail:



AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account

Cloud trail will store in S3 bucket

### Benefits:

2. Simplified compliance
3. Visibility into user and resource activity
4. Security analysis and troubleshooting
5. Security automation

### Amazon RDS -

It is a relational database, RDS is fully managed fast and creditable performance.



RDS is simple and scalable.

RDS is low cost and pay for what we use.

Eg – MySQL, PostgreSQL, MariaDB, Oracle, Amazon Aurora

### Amazon Aurora -

Is RDS reinvented for cloud, Aurora is 5 times better performance than MySQL.

Aurora is available at 1/10 cost of commercial db.

### RDS -

It is easy to administer, RDS is highly scalable.

RDS is available & durable.

RDS provides a feature called ready replica.

Ready Replica – Amazon RDS synchronously replicates the data to a standby instance in a different availability zone.

Amazon RDS supports most demanding applications and is fast.

It is made easy to control n/w access to your db.

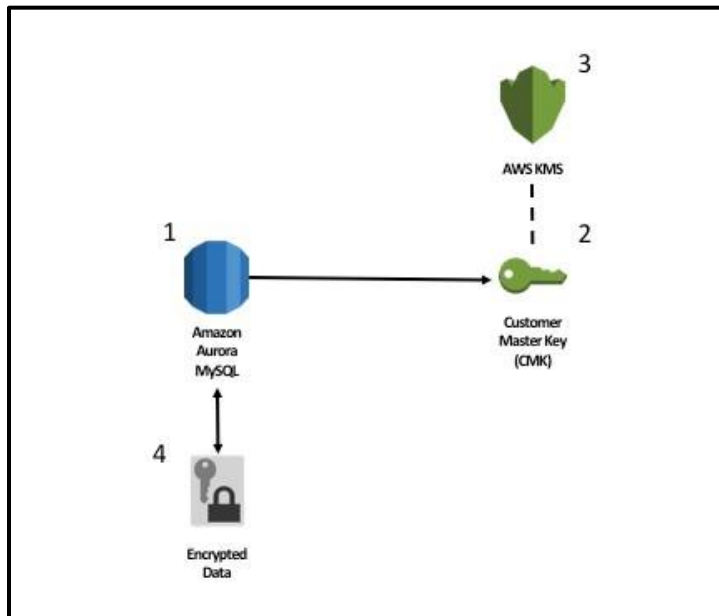
Amazon RDS let you to run your database in your instance in VPC.

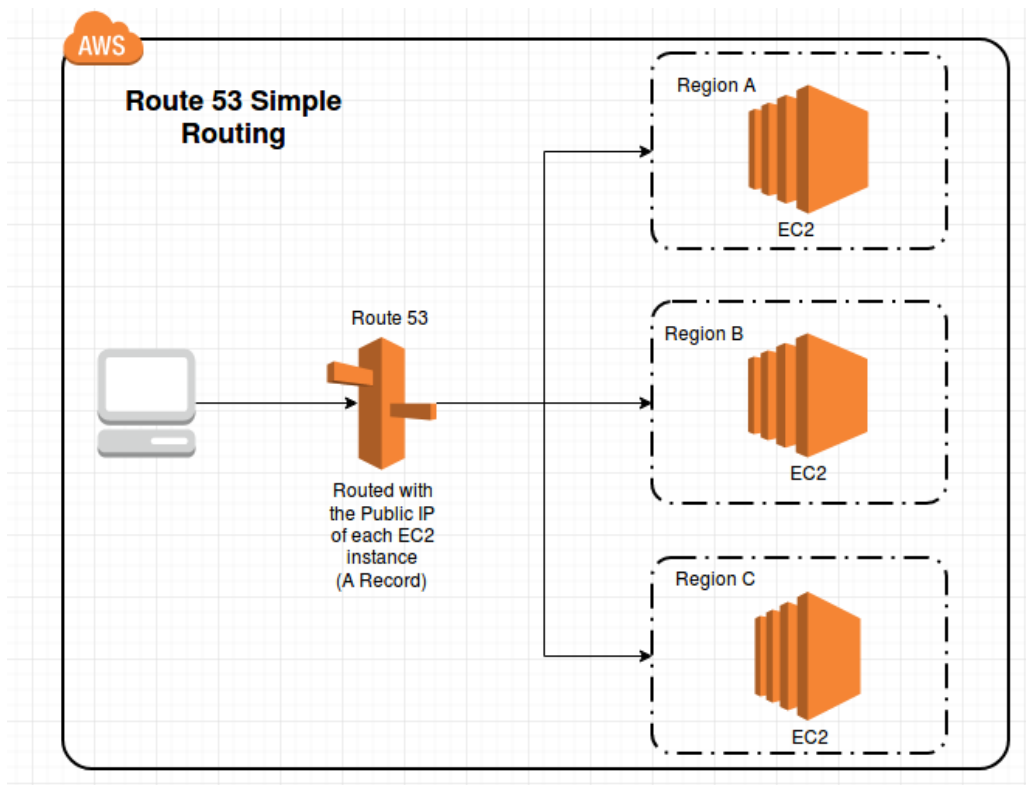
It isolates db and make it secure.

**AWS KMS (Key Management service):** AWS KMS makes it easy for us to create and manage cryptographic keys and control their use across the wide range of use and their applications

Benefits:

1. Fully managed: you control access to your encrypted data by defining permission to use keys while aws
2. Centralised Key management:
3. Manage encryption for AWS services
4. Low cost





### Route 53:

**What is DNS (domain name server) :** it is a technique used by computers to translate human readable domain names into IP address.

How it works? Lets take an example, we want to access [www.example.com](http://www.example.com) from web or a browser whenever we hit [www.example.com](http://www.example.com) on the web/browser/address bar the request for [www.example.com](http://www.example.com) is routed to a DNS resolver which is typically managed by user internet service provider.

The DNS resolver for

## Route 53

Camlin Page  
Date / /

\* What is DNS?

Ans - DNS (Domain name server)  
- DNS is a technique used by computers to translate human readable domain names into ip address

\* How DNS works?

Ans - If we want to access www.example.com from web or browser, where ever we hit www.example.com on web, the request for www.example.com is routed to a DNS resolver which typically managed by users internet service providers (isp)

The DNS resolver for isp forwards request for www.example.com to DNS root name server

List of DNS record types

(i) A record: It is record to route a traffic to resource such as web server, using ~~ip~~ ipv4 address in dotted decimal notation

Ex:-

(ii) AAAA record

ex:-

(iii) CNAME record

ex:-

(iv) SOA (Start of authority)

You use an A record to route traffic to a resource, such as a web server, using an IPv4 address in dotted decimal notation.

192.0.2.1

AAAA record type You use an AAAA record to route traffic to a resource, such as a web server, using an IPv6 address in colon-separated hexadecimal format.

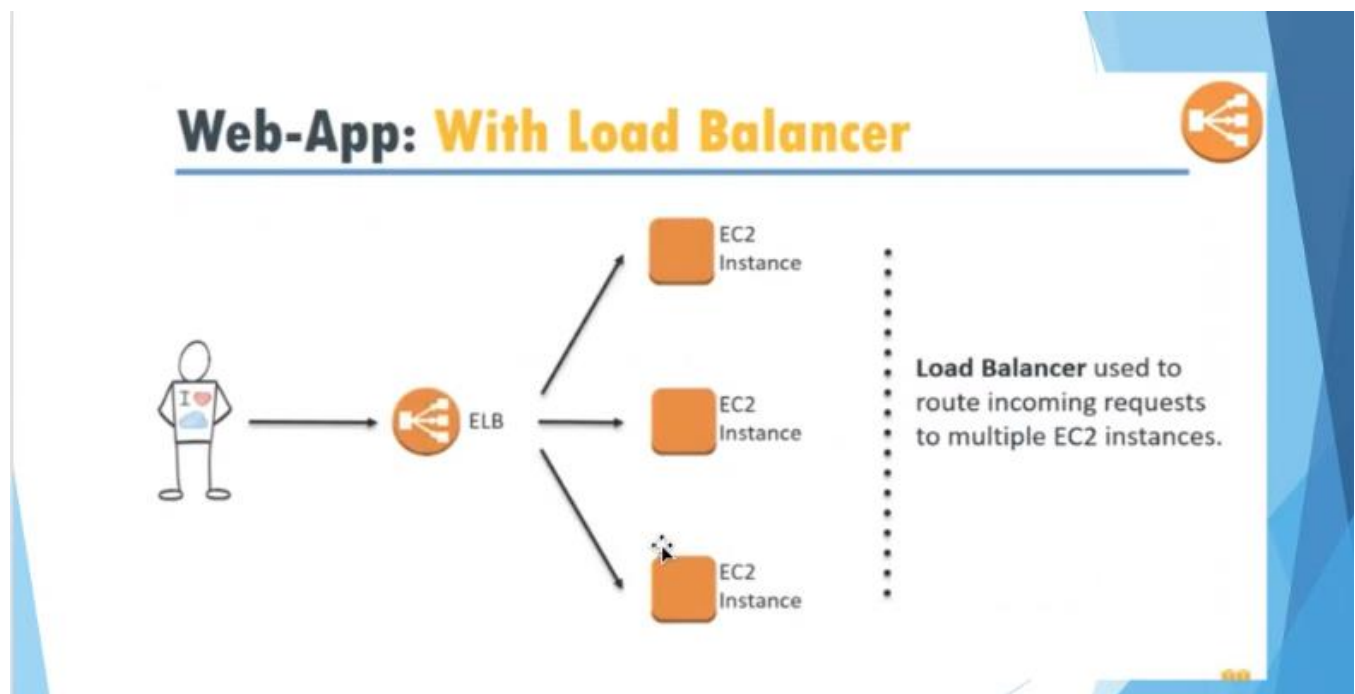
2001:0db8:85a3:0:0:8a2e: 0370:7334

A CNAME record maps DNS queries for the name of the current record, such as acme.example.com, to another domain (example.com or example.net) or subdomain (acme.example.com or zenith.example.org).

hostname.example.com

SOA record type

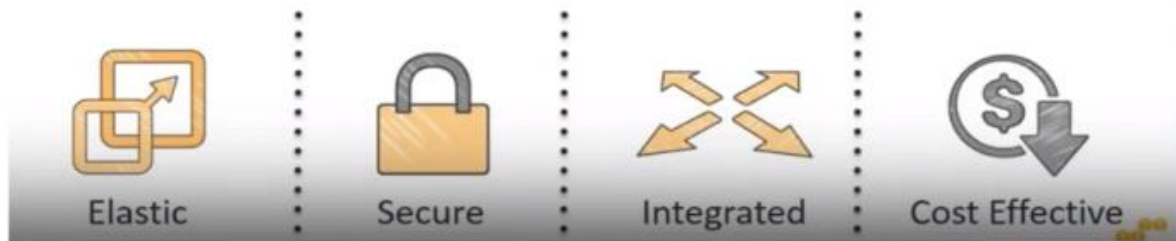
A start of authority (SOA) record provides information about a domain and the corresponding Amazon Route 53 hosted zone. For information about the fields in an SOA record, example: ns-2048.awsdns-64.net hostmaster.awsdns.com 1 1 1 1 60



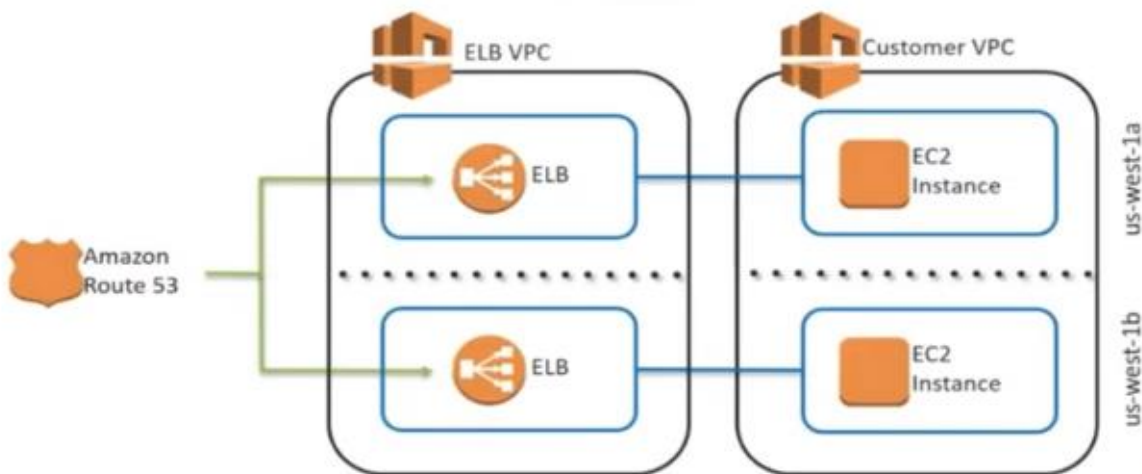
## AWS: Elastic Load Balancer



Elastic Load Balancing automatically distributes incoming application traffic across multiple applications, microservices and containers hosted on Amazon EC2 instances.



## AWS: ELB Architecture



Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses.

It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones.

Elastic Load Balancing offers three types of load balancers that all feature the high availability, automatic scaling, and robust security necessary to make your applications fault tolerant.

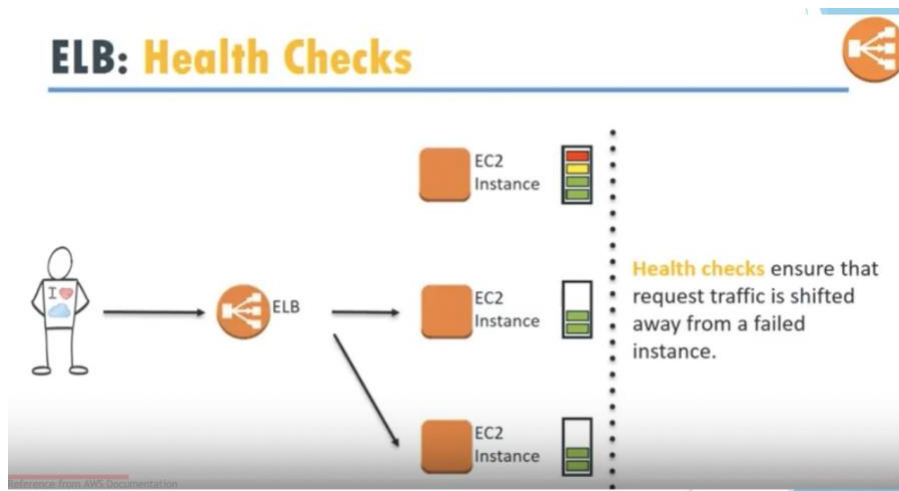
## ELB Features:

ELB is used to load balance over EC2 instance within a VPC. Support both public and private IPS.

Full control over load balancer and security group.

Tightly integrated into associated VPC and subnet.

How load balancer will do health check



## ELB: Health Checks

- Support for **TCP, HTTP & HTTPS** health checks.
- Customize the frequency, failure thresholds, and **list of successful response codes**
- Detailed **reasons for health check failures are now returned via the API**
- Consider the depth and accuracy of your health checks





## Application layer Load Balancing:

You can load balance HTTP/HTTPS applications and use layer 7-specific features, such as X-Forwarded-For headers.

### HTTPS Support:

An Application Load Balancer supports HTTPS termination between the clients and the load balancer. Application Load Balancers also offer management of SSL certificates through AWS Identity and Access Management (IAM) and AWS Certificate Manager for pre-defined security policies.

### Server Name Indication (SNI):

Server Name Indication (SNI) is an extension to the TLS protocol by which a client indicates the hostname to connect to at the start of the TLS handshake. The load balancer can present multiple certificates through the same secure listener, which enables it to support multiple secure websites using a single secure listener.

### IP addresses as Targets:

You can load balance any application hosted in AWS or on-premises using IP addresses of the application backends as targets. This allows load balancing to an application backend hosted on any IP address and any interface on an instance. Each application hosted on the same instance can have an associated security group and use the same port. You can also use IP addresses as targets to load balance applications hosted in on-premises locations (over a Direct Connect or VPN connection), peered VPCs and EC2-Classic (using Classic Link). The ability to load balance across AWS and on-prem resources helps you migrate-to-cloud, burst-to-cloud or failover-to-cloud.

## Network load balancer:

Network Load Balancer operates at the connection level (Layer 4), routing connections to targets - Amazon EC2 instances, microservices, and containers – within Amazon Virtual Private Cloud (Amazon VPC) based on IP protocol data. Ideal for load balancing of both TCP and UDP traffic, Network Load Balancer is capable of handling millions of requests per second while maintaining ultra-low latencies. Network Load Balancer is optimized to handle sudden and volatile traffic patterns while using a single static IP address per Availability Zone. It is integrated with other popular AWS services such as Auto Scaling, Amazon EC2 Container Service (ECS), Amazon CloudFormation and AWS Certificate Manager (ACM).

### Network load balance key features:

#### 1. Connection-based Load Balancing

You can load balance both TCP and UDP traffic, routing connections to targets - Amazon EC2 instances, microservices, and containers.

#### 2. High Availability

Network Load Balancer is highly available. It accepts incoming traffic from clients and distributes this traffic across the targets within the same Availability Zone. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets.

#### 3. high Throughput

Network Load Balancer is designed to handle traffic as it grows and can load balance millions of requests/sec. It can also handle sudden volatile traffic patterns.

## Gateway load balancer:

Gateway Load Balancer makes it easy to deploy, scale, and manage your third-party virtual appliances. It gives you one gateway for distributing traffic across multiple virtual appliances, while scaling them up, or down, based on demand. This eliminates potential points of failure in your network and increases availability.

What is OSI mode?

SDLC life cycle: Agile methodology

APPLICATION LOAD BALANCER	NETWORK LOAD BALANCER	CLASSIC LOAD BALANCER
It operates on Layer 7	It operates on Layer 4	It operates on Layer 7 and Layer 4
Its supports HTTP/ HTTPS (Internet)	Its supports TCP/UDP/TLS	It supports on HTTP/ HTTPS/ TCP/ TLS
Supports path-based routing, host-based routing, query string parameters-based routing and source IP-address based routing.	It offers ultra-high performance, Low latency a TSL offloading at scale.	Old generations not recommended for new applications.
Operates on request level.	Operates on the connection level.	It operates on both the request level and the connection level.
Supports IP addresses, Lambda Functions and containers as targets.	Supports UDP and static IP addresses as targets.	Use for existing applications running on EC2-Classic.
Provide load balancing to multiple ports on an instance	Provide load balancing to multiple ports on an instance	NA

## Autoscaling:

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost

## Autoscaling group:

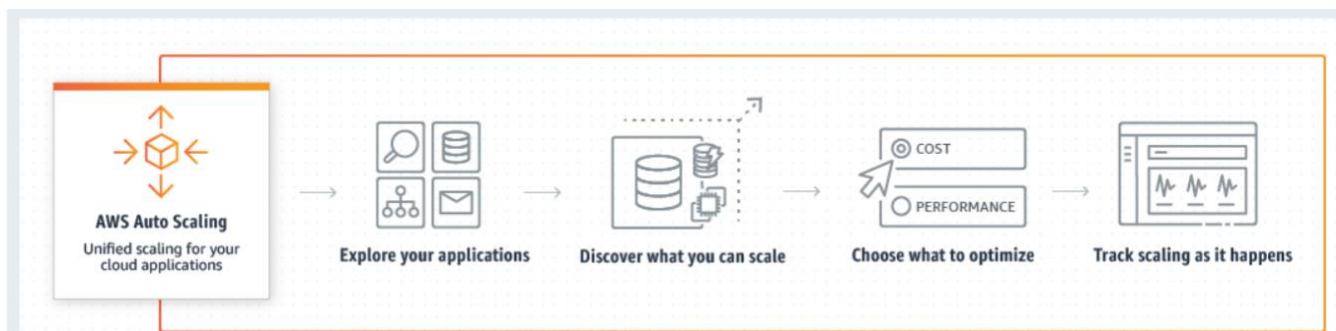
SETUP SCALING QUICKLY

AUTOMATICALLY MAINTAIN PERFORMANCE

MAKE SMART SCALING DECISIONS

PAY ONLY FOR WHAT YOU NEED

AWS Auto Scaling Features





# Autoscaling

