-: DEVOPS-AWS-NOTES :-

****

###############################
DevOps with AWS Master Program
###############################

Pre-Requisites : No - Prerequisites

Note: Programming Knowledge is not required to learn DevOps

##############
Course Content
##############

Module-1 : DevOps Introduction

Module-2 : Linux OS & Shell Scripting

Module-3 : AWS Cloud (20+ AWS Services)

Module-4 : DevOps Tools (15+ Tools) (Docker & K8S)

Module-5 : Realtime Projects

Module-6 : Interview Guide (Interview Questions, Resume Prep, Tips & Tricks)

##############
What is DevOps
##############

-> DevOps = Development + Operations

-> DevOps is a culture / process

-> DevOps means set of practises

-> DevOps will help us in simplifying application delivery process

-> DevOps will help us in automating application build & deployment process

-> Using DevOps practises we can deliver project to client quickly and easily

-> To automate application build & deployment we will use several tools (DevOps tools)

Build Process : Compile & Package (Converting project into Server Understandble Format)

Server : Server is a program which is used to run our application (every body can access)

Deployment : The process of keeping application in server

-> Dev Team will take care of project development activities

-> Operations Team will take care of project deployment & deliveriy activities

-> To automate project build & deployment we will use several tools

###############
DevOps Tools
###############

SVN / Github / Bitbucket : For source code management

Maven / Gradle : Build Tool (Compile & Package)

Sonarqube / Sonarlint : Code Quality Checking (Code Review)

Nexus / JFrog : For build artifact storage

Jenkins / Bamboos : To automate build & deployment using CI CD Pipeline

Tomcat / Jboss / IIS : It is a server to run our web applications

Docker : It is a containerization platform (Containers)

Kubernetes / Docker Swarm / Open Shift : It is an Orchestration Platform (Managing the container)

Ansible / Chef / Puppet : Configuration Management

Terraform : To create infrastructure in cloud (AWS / Azure / GCP)

Promethues / App Dynamics : Monitoring & Alerting tool

Grafana : Visulation tool (It will give dashboard to monitor our servers)

ELK Stack / Splunk : For application log monitoring

JIRA : Project Management Tool (Work assignment & tracking)

##
Roles & Responsibilities of DevOps Engineer
##

1) Creating Infrastructure (Creating Machines) in Cloud using Terraform

2) Manage Configurations in the Machines using Ansible

3) Setup Database required for the Project

4) Setup Storage Required for the Project

5) Setup Servers which are required to run our application

6) Create Git Hub Repositories required for Project

7) Manage Permissions for Repositories (Read & Write)

8) Create CI CD Pipelines using Jenkins to build & deploy our application

Note: CI CD means continuous integration & Continuous Deployment

9) Manage & Monitor CI CD Pipelines

10) Monitor and Manage our servers

11) Monitor and Manage our infrastructure

12) Monitor & manage our applications

13) Secure our infrastructure, servers & applications

##############################
AWS Introduction
##############################

-> For Every Project we need to setup infrastructure (IT Infrastructure)

-> Infrastructure means the softwares & hardwares which are required to develop & run our applications

1) We need machines (computers)

2) We need application servers to run our application

3) We need database to store our application data

4) We need to setup High Speed Network for our machines

5) We need to setup Power & Power Backup

6) We need a room to keep our machines

7) Setup Air Conditioner for server room

8) We need people to manage these servers, network, power and server room

##############################
Challenges with our Infrastructure
##############################

1) Cost
2) Maintenence
3) Scalability
4) Availability
5) Security
6) Privacy

Note: If business is not running successfully, our infrastructure will be wasted (Time waste & Money waste)

-> To over come the problems of ou Infrastructure setup , we can go for "Cloud Computing"

################
Cloud Computing
###############

-> The process of delivering IT resources over the internet on demand basis is called as Cloud Computing.

-> Instead of purchasing, owning and maintaining resources we can take the resources from Cloud Provider on Pay As You Go Pricing.

-> "Pay as you go" means pay for use (like post paid bill, credit card bill etc...)

#############################
Cloud Computing Advantages
#############################

1) Low Cost
2) Pay For Use
3) Scalability
4) Availability (24/7)
5) Reliability
6) Security
7) Unlimited Storage
8) Backup

################
Cloud Providers
###############

-> The companies which are providing IT resources over the internet are called as Cloud Providers

1) Amazon   (AWS)

2) Microsoft (AZURE)

3) Google   (GCP)

4) Salesforce (Salesforce CRM)

5) Oracle (Oracle Cloud)

6) IBM (IBM Cloud)

7) VMWare (VMWare)

###############
Cloud Services
###############

*************** IAAS   : Infrastructure as a service *****************

-> Cloud Provider will provide below components

Network
Storage
Virtualization
Servers

-> We have to manage below components

OS
Middleware
Runtime
Application

*******************PAAS : Platform as a service*************

-> Cloud Provider will provide the platform to run our application
-> We just need to run our application on the platform given by provider
-> Infrastructure & Runtime will be taken care by Provider

Ex: AWS Elastic Beanstalk

******************SAAS : Software as a service*******************

-> We will use provider application to run our business
-> Application development, app infrastructure, maintenence will be taken caren by Provider only

Ex: google drive, zoom, Salesforce CRM, drop box

=========================
What is Operating System ?
=========================

-> Operating System is a software

-> It acts as a mediator between end user and computer hardware components

-> OS is mandatory is mandatory to use a computer

-> There are several Operating Systems available in the market

Ex: Windows, Linux, Unix, Solaris, Mac, Android, IOS etc...

========
Windows
========

1) Windows OS developed by Microsoft

2) Widows is Licensed OS

3) Windows is single user based OS (only one person can use at a time)

4) Security features are very less in windows OS

5) Windows OS is recommended for personal use

=======
Linux
=======

1) Linux OS is developed by Linus Torvalds

2) Linux is free & open source OS

3) Linux is Multi User based OS (multiple users can use Linux Machine at a time)

4) Security Features are very high in Linux OS

5) Linux OS is higly recommended for applications deployments & servers setup

=============
History of Linux
=============

-> Linus Torvalds identified few issues in Unix and told to Unix team to change the Unix OS

-> Unix team didnt accept Linus Torvalds changes

-> Linus Torvalds started developing his own Operating System by using 'Minux' OS

(Li)nus Toravalds   + Mi(nux) ==> Linux OS

-> Linus Torvalds released Linux OS for free of cost

-> Linux Torvalds provided Linux OS source code also in the internet

-> People and companies downloaded source code of Linux OS and they modified according to their requirement and released into market with their brand names

-> As companies released their own Linux OS we can see several distributions for Linux OS

Ex:

Ubuntu
CentOS
RedHat
Fedora
SUSE
KALI
Debian etc....

Note: There are 200+ Linux Distributions available in the market

================
Environment Setup
================

1) Create Account in AWS (Free Tier)

2) Create Linux Virtual Machine in AWS using EC2 service

3) Connect to Linux Virtual Machine using MobaXterm / Putty

================
Linux File System
================

Everything will be represented as file in Linux

3 Types of Files

1) Normal Files

2) Directory Files (Folders)

3) Link Files

-> Normal file will start with '-'

-> Directory file will start with 'd'

-> Link file will start with 'l'

==============
Files creation
=============

touch : It is used to create empty files

Syntax:

```
touch <filename>
```

```
                touch f1.txt

                touch a1.txt a2.txt a3.txt

cat :  It is used to create files with content & it is used to print file content

Syntax:

        cat > filename   : Create file with data   (ctrl + d to close the file)

        cat >> filename  : Append data to existing file

        cat  filename   : Print file content

        cat filename1  >  filename2  : Copy data from one file to another file

        cat  f1.txt f2.txt > f3.txt  : Copy more than one file data into another
file

mv : It is used to move/rename the files

Syntax:

                mv present-name  new-name

rm : It is used to remove the files

syntax:
                rm filename

=====================
Working with Directories
=====================

mkdir : it is used to create/make directory

rmdir : it is used to remove only empty directory

rm -r  <dirname> : It is used to delete non-empty directories

cd <dirname>  : Change directory

cd ..  : Come out from the directory

ls -l <dirname> : list the content of given directory


cat <filename>   : It will display all the data available in the file

head  : It will display first 10 lines of the file  from top  (10 is the default
count)

                        head  <filename>  (it will give first 10 lines of data)
                        head  -n  15 <filename>     (it will give first 15 lines of
data)
                        head  -n  25 <filename>   (it will give first 25 lines of
data)

tail   :  It will display last 10 lines of the file from bottom (10 is the default
count)
```

```
                        tail  <filename>   (it will give last 10 lines data)
                        tail -n 15 <filename>   (it will give last 15 lines data)
                        tail -n 25 <filename> (it will give last 25 lines data)

                        tail -n 50 <filename> (it will give last 50 lines data)
                        tail -n +50 <filename> (it will give data from 50th line to
till last line)
```

Note: To get latest data from file we need to use 'tail' command because lastest data will be appended at bottom

grep : Global Regular Expression Print

-> Grep is used for searching

```
                grep -i 'Linux' *    (It will search for linux word in  all the
files)

                grep -i 'Linux' <filename>   (it will search for linux word in
given filename)
```

wc : (word count) it is used to count no.of lines, words and no.of characters of given file

```
                wc  <filename>
```

cp : It is used for copy and paste

```
                cp  <filename>  <filename>
```

Note: If we want to copy more than one file data then we should go for 'cat' command

```
                cat f1.txt f2.txt > f3.txt
```

mv : it is used for renaming & moving

```
                mv  <existing-file-name>  <new-file-name>

                mv  <source>  <destination>
```

==================
Text Editor
====================
vi : Visual Editor

vi <filename> : It will open the file

press 'i' to enter into 'insert' mode

-> in insert mode we can change file data

press 'esc' to come out from insert mode

press :wq to save and quit the file

press :q! to close the file without saving

=================

SED : Stream Editor
=================

-> It is used to perform search, find, replace, insert and delete

-> This command is famous for substitution (replacement)

-> Using SED command we can perform operations without opening the file

```
sed 's/Python/JAVA/' python.txt     (it will print output on console but original data will not modified in the file)

sed -i 's/Python/JAVA/' python.txt  (it will make changes in original file)

sed -i '5d' <filename>  (it will delete 5th line)
```

=============
Man command
=============

-> It will provide information about given command (documentation command)

```
man ls

man cat

man tac
```

======================
Find & Locate Commands
======================

-> find & locate commands are used for file search in linux machine

-> locate command will perform search operation in locate database (internal database in linux)

-> find command will perform search operation in entire linux file system (It will give accurate result)

```
locate apache (it will give all the files path which are having apache in the file name)

locate -c apache (it will give file count)

find /home -name python.txt   (it will search for the file with name python.txt inside /home directory)

find /home -type f -empty (It will print all empty files available inside /home directory)

find /home -type d -empty (it will print all empty directories)
```

===========
history
===========

-> It will give commands execution history

```
===============================

ls -ltr
touch
cat
tac
rm
mv
mkdir
rmdir
rm -r <dirname>
head <filename>
head -n 100 <filename>
tail <filename>
tail -n 50 <filename>
tail -n +25 <filename>
grep
wc
vi
sed
man
find
locate
history
```

```
#################################
Working with User Accounts in Linux
################################
```

-> Linux is a multi-user operating system

-> With in one linux machine we can create multiple user accounts

-> Multiple users can access one linux vm at a time and they can perform Multi-tasking

Note: Every linux machine will have 'root' account (super user --> sudo)

-> When we launch a linux vm using 'Amazon Linux AMI' we will get 'ec2-user' account by default

whoami -> it will give currently logged in username

id <username> : It will give information about user account

pwd : It will display present working directory

Note: For every user account, one home directory will be created.

For ec2-user account the home directory is : /home/ec2-user

```
# Switch to root user

$ sudo su
```

```
Note: To perform admin activities we will use 'sudo' permission


#################
Create user in linux
#################

# create user
$ sudo useradd  <username>

# set password for user
$ sudo passwd <username>

# verify user account details
$ id <username>

# List all users in Linux
$ cat /etc/passwd

# switch user account
$ su <username>

Note: After swithcing username, change home directory also

# delete user
$ sudo userdel <username>

# Display all groups
$ cat /etc/group

# Create a group
$ sudo groupadd <groupname>

# Add user to a group
$ sudo usermod -aG <group-name> <user-name>

# Remove user from the group
$ sudo gpasswd -d <username> <group-name>

# Delete group
$ sudo groupdel <group-name>

# print users who are belongs to particular group
$ sudo lid -g <group-name>


===============
File Permissions
===============

-> File Permissions will decide who can do what on that file

-> Every File will have 3 types of permissions

                1) Read  (r)
                2) Write (w)
                3) Execute (x)

-> In Linux, File Permissions are divided into 3 sections

                1) User Permissions  (first 3 characters)
                2) Group Permissions (charters 4, 5, 6)
                3) Others Permissions (charaters 7,8 & 9)
```

```
        Ex:   rwxrwxrwx


-> To change file permissions we have 'chmod' command


// add execute permission for user
$ chmod u+x <filename>

// add execute permission for group
$ chmod g+x <filename>

// remove write permission for group
$ chmod g-w <filename>

// add write & execute permissions for others
$  chmod o+wx <filename>

// add write permissions for others (2 files at a time)
$ chmod o+w <file1> <file2>


************** We can represent file permissions with Numeric Numbers also
****************

0 - No Permission

1 - Execute

2 - Write

3 - Execute & Write

4 - Read

5 - Read & Execute

6 - Read & Write

7 - Read, Write & Execute


$ chmod 777 <filename>

$ chmod 765 <filename>


#########
chown
########

-> 'chown' command is used to change file ownership

-> We can see owner of the file using 'ls -l' command

# changing owner of a file

$ sudo chown <uname> <filename>

-> We can change file owner using userID also

$ sudo chown <UID> <filename>

Note: We can get UID for username using 'id uname' command
```

```
$ id ramesh

# change group of a file
$ sudo chown :groupName <fileName>
```

Q) What is the difference between 'chmod' and 'chown' commands ?

chmod ---> it is used for managing file permissions

chown ---> it is used for managing file ownership

```
====================
Networking Commands
====================
```

ifconfig : This command is used to get ip address of our machine

ping : It is used to check connectivity

```
                $ ping <ip>
```

-> 0% packet loss means our ping is succesful (we got response from server)

-> 100% packet loss means our ping is failure (no response from server)

wget : It is used to download a resource from internet using URL

```
                $ wget <url>

                $ wget
https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.zip
```

Note: We can extract that zip file using 'unzip' command

```
                $ unzip <zip-file-name>
```

curl : It is used to send http request to a server

```
                $ curl <url>
```

```
============================
Softwares Installation In Linux
============================
```

-> To install softwares in Linux we will use Package Manager

-> There are several package managers available for Linux

1) yum
2) apt
3) deb
4) RHEL

-> Based on Linux distribution we need to choose package manager

Amazon Linux / Cent OS / Red Hat -------> yum

Ubuntu / Debian ---> apt

```
# update existing packages
$ sudo yum update -y

# install git client software in linux
$ sudo yum install git

# check git version
$ git --version

# install java
$ sudo yum install java

# install java 1.8 version
$ sudo yum install java-1.8.0-openjdk

# check java version
$ java -version

# install maven
$ sudo yum install maven

# check maven version
$ mvn -version

# Un-install software
$ sudo yum remove maven
```

==============================
Hosting Static website in Webserver
==============================

-> Website nothing but collection of web pages

-> Websites are 2 types

    1) static website
    2) dynamic website

-> static website will give same response for everybody (fixed content)

-> dynamic website means content will display based on user action

-> To run the website we need to deploy that website inside a server

######## install webserver #######

```
$ sudo yum install httpd
```

Note: httpd is a web server package and it is used for static websites hosting

########## start webserver #############

```
$ sudo service httpd start
```

Note: HTTPD webserver runs on HTTP protocol with 80 port number (enable that in security group inbound rules)

########## modify webpage content ############

```
$ cd /var/www/html
```

$ sudo vi index.html

Note: write content in index.html file then save and close

############ Access website using EC2 VM public ip in your browser ###############

URL : http://13.233.129.51/

URL : http://13.233.129.51:80/

************* Note: To deploy & run dynamic websites we will use below servers ***************

1) Apache Tomcat
2) JBOSS
3) Weblogic
4) WebSphere
5) Glassfish etc....

==================
Linux Architecture
==================

-> Linux Architecture works based on 4 layers

1) Applications

2) Shell

3) Kernel

4) Hardware Components

Hardware âˆ' Hardware consists of all physical devices attached to the System. For example: Hard disk drive, RAM, Motherboard, CPU etc.

Kernel âˆ' Kernel is the core component for any (Linux) operating system which directly interacts with the hardware.

Shell âˆ' Shell is the interface which takes input from Users and sends instructions to the Kernel, Also takes the output from Kernel and send the result back to output shell.

Applications âˆ' These are the utility programs which runs on Shell. This can be any application like Your web browser, media player, text editor etc.

=======================================

Q) How to find machine running time ?

$ uptime : It is used to find from how many hours our machine is running

Q) How to find free space available in linux machine

$ df -h : It is used to find out how much free space available in our machine

Q) how to change password of user ?

$ sudo passwd username

Q) How to display list of users available in linux ?

```
$  cat /etc/passwd : It will print all users available in linux
```

Q) How to display all groups available in Linux ?

```
$ cat /etc/group : It will print all groups available in linux
```

Q) How to assign an user for a group ?

```
$ sudo usermod -aG <groupName> <uname>
```

Q) What is sudo in linux ?

-> Super user (administrator permissions)

###############
Shell Scripting
###############

++++++++++++++
What is shell ?
++++++++++++++

-> Shell is responsible for reading commands given by user

-> Shell will verify command and will give instructions to kernel to process that command

-> If command is invalid shell will give error

-> Kernel will execute our command with System Hardware Components

-> Shell acts as mediator between User and Kernel

+++++++++++++++++++
What is Scripting ?
+++++++++++++++++++

-> Scripting means set of commands mentioned in a file for execution

-> Scripting is used to automate our routine work

-> For example i want to execute below commands every day as a linux user

```
$ date
$ cal
$ whoami
$ pwd
$ ls -l
```

-> Instead of executing all these commands manually we can keep them in a file and we can execute that file.

-> The file which contains set of commands for execution is called as 'Script file'

-> Script file will have .sh extension

        Ex: task.sh

######################
What is Shell Scripting ?
######################

-> Shell Scripting is used to execute set of commands using a script file

-> When we execute script file then shell will read those commands and will verify commands syntax

-> Shell will give instructions to 'Kernel'

-> Kernel will give instructions to hardware components to perform actual operations

##############
Types of Shells
###############

-> There are several shells available in linux OS

1) Bourne Shell
2) Bash Shell
3) Korn Shell
4) CShell
5) TShell
6) ZShell

```
# Display all the shells of our linux machines
$ cat /etc/shells

# Display the default shell of our linux machine
$ echo $SHELL
```

Note: The most commonly used shell in linux is 'BASH SHELL'.

Note: Shell Script files will have .sh extension

###################################
Working with First Shell Script Program
###################################

```
# Create a script file
$ vi task.sh

whoami
pwd
date
```

-> Save the file and close it (ESC + :wq)

```
# Run the shell script (Option-1)
$ sh task.sh
```

Note: If we get permission denied then we should provide 'execute' permission using 'chmod' command

```
# Run the shell script (Option-2)
$ ./task.sh
```

##############################
What is sha-bang in shell script ?
##############################

-> Sha-bang is used to specify which shell should be used to process our script

Syntax :

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

***************Shell Script - 2 (print output to console) *************

```
#! /bin/bash

echo "Welcome to Scripting"
echo "Scripting is used to automate regular work"
echo "Scripting requires lot of practise"
```

*****************Shell Script - 3 (take input from user) ***********************

```
#! /bin/bash

echo "Enter your name:"
read name
echo "Good Morning $name"
```

********************* Shell Script - 4 ******************************

```
#! /bin/bash

a=10
b=20

c=$(($a + $b))

echo "Sum of $a and $b is =  $c"
```

*************************** Shell Script - 5 *******************************

```
#! /bin/bash

echo "Enter First Number"
read a
echo "Enter Second Number"
read b

c=$(($a + $b))

echo "Sum of $a and $b is =  $c"
```

```
*********************************
Variables
Control Statements
Case Statements
Loops
Functions
*********************************
```

-> Variables are place-holders to store the value

-> Varibles are key-value pairs

-> In Shell Scripting there is no concept of Data Type.

-> Every value will be treated as text/string

Ex:

```
name=ashok
age=30
email=ashokitschool@gmail.com
phno=1234
```

-> Variables are divided into 2 types

1) Environment Variables or System variables
2) User Defined Variables

-> The variables which are already defined and using by our system are called as Environment/System variables

Ex:

```
$ echo $USER
$ echo $SHELL
```

-> Based on our requirement we can define our own variables those are called as user defined variables

Ex:

```
name=ashok
age=30

$echo $name $ age
```

################
Variable Rules
#################

-> We should not use special symbols like -, @, # etc....
-> Variable name should not start with digit

Note: It is recommended to use uppercase characters for variable name

-> we can use 'readonly' for variable so that variable value modification will not be allowed

########################
Command Line Arguments
########################

-> The arguments which will pass to script file at the time of execution

-> Cmd args are used to supply the values dynamically to the script file

Ex:

```
$ sh demo.sh ashokit 30
```

-> We can access cmd args in script file like below

$# - no.of args
$0 - script file name

```
$1 - First Cmd Arg
$2 - Second Cmd Arg
$3 - Third Cmd Arg
$* -  All Cmd Args
```

-> To comment any single line we can use '#'

```
echo 'hi'
#echo 'hello'
```

-> We can comment multiple lines also in script file like below

```
<<COMMENT
        ....................

COMMENT
```

-> We can hold script execution for some time using 'sleep' command

```
#! /bin/bash

echo $#
echo $0
echo $1
sleep 30s
echo $2
#echo $*
```

######################
Conditional Statements
#######################

-> Conditional statements are used to execute commands based on condition

Syntax:

```
if [ conition ]
then
        stmts
else
        stmts
fi
```

-> If given condition satisfied then if statments will be executed otherwise else statements will be executed

```
if [ condition ]
then
        stmts
elif [ condition ]
then
        stmts
else
        stmts
fi
```

Ex:

```
#!/bin/bash

echo "Enter Your Favorite Color"

read COLOR

if [ $COLOR == 'red' ]
then
        echo "Your are cheerful"
elif [ $COLOR == 'blue' ]
then
        echo "You are joyful"
else
         echo "You are lucky"
fi
```

```
###################
Working with loops
###################
```

-> Loops are used to execute stmts multiple times

-> We can use 2 types of loops

```
                1) Range based loop  ( ex: for loop )
                2) Conditional based loop ( ex: while loop )
```

```
################
For Loop Example
################

#! /bin/bash

for ((i=1; i<=10; i++))
do
echo "$i"
done

###################
While loop Example
####################

#! /bin/bash
i=10
while [ $i -ge 0 ]
do
echo "$i"
let i--;
done


#################
Infinite Loop
################

#! /bin/bash
while true
do
echo "This is my loop stmt"
done
```

Note: To stop infinite loop we will use 'ctrl + c'

##############
Functions
##############

-> The big task can be divided into smaller tasks using functions

-> Function is used to perform an action / task

-> Using functions we can divide our tasks logically

-> Functions are re-usable

Syntax:

```
function functionaName( ) {

     // commands to execute
}
```

######################
Writing welcome function
#######################

```
#! /bin/bash

function welcome(){
  echo "Welcome to functions...";
  echo "I am learning Shell Scripting";
  echo "Shell Scripting is used to automate our regular work";
}

# call the function
welcome
```

#######################
Function with Parameters
#######################

```
#! /bin/bash

function welcome ( ) {
     echo "$1";
}

# call function
welcome Linux
welcome AWS
welcome DevOps
```

================
Cloud Computing
================

-> Cloud Computing means the process of delivering on-demanded resources over the internet

-> Instead of buying, owning & maintaing infrastructure we can take it from Cloud Provider

-> There are several advantages with Cloud Computing

1) Low cost
2) Pay As You Go (Ex: Post paid mobile bill, credit card bill etc..)
3) Scalability (increasning & decreasing)
4) Availability (zero downtime)
5) Reliability (Strong)
6) Security
7) Unlimited Storage
8) Backup

On-Premise Infrastructure (in-house infrastructure)

Cloud Infrastructure (provider managed infrastructure)

=============
Cloud Provider
=============

-> The company which is providing cloud infrastructure is called as Cloud Provider

Amazon ----------> AWS (80 %)

Microsoft -------> Azure

Google ---------> GCP

Salesforce ------> Salesforce CRM

Oracle -----------> Oracle Cloud

IBM --------------> IBM Cloud

=============
Cloud Services
=============

IAAS : Infrastructure as a service

PAAS : Platform as a service

SAAS : Software as a service

===========
AWS Cloud
===========

-> AWS stands for Amazon Webservices

-> AWS providing cloud infrastructure since 2006

-> AWS providing cloud infrastructure based on 'Pay As You Go' model

-> AWS offering 200+ services

-> AWS cloud services are using in 190+ countries

-> AWS having Global Infrastructure

=======================
AWS Global Infrastructure
=======================

-> AWS providing global infrastructure using Regions & Availability Zone

Region ---------> Geographical Location

Avaliability Zone -----------> Data Center

Note: One Region will have multiple Availability Zones

======================
AWS Introduction
======================

-> AWS stands for Amazon Webservices

-> AWS cloud managing by Amazon company

-> AWS is one of the leading Cloud Provider in the market

-> AWS started providing IT resources over internet from 2006 onwards

-> 190+ countries using AWS Cloud

-> AWS providing 200+ Services

-> AWS providing Cloud Services based on 'Pay As You Go' model

======================
AWS Services Names
======================

EC2 : Elastic Compute Cloud : To create virtual machines

EBS : Elastic Block Store (External HD)

EFS : Elastic File System

S3 : Simple Storage Service : Unlimited Storage

RDS : Relational Database System : To create SQL Databases (Oracle, MySQL, Postgres, MS SQL etc..)

VPC : Virtual Private Cloud : Isolated Network

Route 53 : Domain Name Mapping (URL Mapping)

BeanStalk : For Paas Model

IAM : Identity & Access Management (who can access which service in AWS)

ECS : Elastic Container service (To run containers)

ELB : Elastic Load Balancer (Load Balancing)

Lambda : Serverless Computing (run the code without thinking about servers)

++

-> To use AWS provided cloud services we need to create one account in AWS

Note: It will ask debit / credit card for account creation

-> AWS will charge 2 rs for account creation and they will send 2 rs back to our account after account verified

-> In AWS few services are free and few services are paid

-> As part of our training we will use both free and paid services

Note: When we use paid services, after practise completion we need to delete those service to avoid billing

-> If bill got generated we can request AWS Support team to waveoff our bill because we are AWS learners and we are exploring AWS Cloud services.

-> AWS will not deduct bill amount from our card directley. We need to pay that bill manually.

-> If we don't pay AWS bill amount then our AWS account will be terminated.

++

-> AWS providing global infrastructure

-> 190+ contries are using AWS Cloud through internet

-> To provide Global Infrastructure AWS using Regions & Availability Zones concept

-> Region means one geograhical location

-> Availability Zone means data center

-> Data Center means a big building which contains servers with network

-> One Region can have multiple Availability Zones (AZ)

-> AWS Having 26 Regions and 84 Availability Zones in the world

-> In india AWS having 1 region (Mumbai) ---- ap-south-1

-> Mumbai region having 3 availability zones

```
                                - ap-south-1a
                                - ap-south-1b
                                - ap-south-1c
```

Note: Hyd Region Coming Soon

Note: It is recommended to use Nearest Region in AWS to setup our infrastructure

Note: In AWS few services are global (S3, Route 53 etc...) and few services regional (EC2, VPC, RDS etc...)

========
EC2
========

-> EC2 stands for " Elastic Compute Cloud "

-> EC2 is the most demanded service in AWS

-> EC2 is used to create virtual machines

-> EC2 instances are re-sizable

-> EC2 is regional service

-> EC2 instance minimum billing period is 1 hour

        Ex: If you run for 5 minutes also it will be considered as 1 hour

-> The Virtual Machine which we create using EC2 service is called as EC2 instance

        Alias Names : EC2 Instance / Virtual Machine / VM / Server

-> EC2 is a paid service

Note: EC2 providing t2.micro as free tier eligible for 1 year with monthly 750 hours

-> When we create EC2 instance, AWS will provide default storage and default network

-> Storage will be provided using EBS (Elastic Block Store)

-> Network will be provided using VPC (Virtual Private Cloud)

===================
EC2 instances types
===================

1) On-Demanded Instances

2) Reserved Instances

3) Spot Instances

4) Dedicated Host

====================
OnDemanded Instance
====================

-> Whenever we want then we can create it

-> Fixed Price (Hourly)

-> Pay For Use

-> No Prior Payments

-> No Committment

====================
Reserved Instances
===================

-> It is like advanced booking

-> Long Term Commitment

-> Prior Payment option available (paritial payment / full payment)

-> Commitment for 1 year or 3 years

-> AWS will provide discount on price

=================
Spot Instances
==================

-> It is like bidding or Auction

-> AWS will offer high capacity systems for low price

-> 72% savings on price

=====================
Dedicated Host Instance
=====================

-> This is a physical machine

-> Licensed Softwares

-> It is very costly when compared with other instance types

=====================
EC2 instance states
=====================

Running --- Billing

Stopped -- No Bill

Terminated -- No Bill

=====================
EC2 instance families
=====================

t2. t3, t4, c2, c3, i2, i3, m2, m3, m4 etc....

-> We are using t2.micro instance for practise

Note : t2.micro instances are Free for 1 year (from aws account creation date)

Note: Monthly 750 hours usage is free for 1 year when we go for t2.micro

Note: We can find EC2 instance types service rates in dashboard

===========================
Amazon Machine Image (AMI)
===========================

-> AMI is a template to launch our instances

-> Image represents pre-configured system with softwares installed

-> To create EC2 instance with Windows OS we can use Windows OS AMI

-> To create EC2 instance with Linux OS we can use Linux OS AMI

-> In AWS, we can create our own AMI also

=======================
What is Key Pair in EC2 ?
=======================

-> Key Pair is the combination of Public Key and Private key

-> AWS will store public key and it will provide private key for us (We have to save that)

-> Private Key file extension will be .pem

-> Public key & Private Key is used to connect with EC2 instance securley

-> If we want to connect with EC2 VM we need to provide private key for AWS then AWS will compare our private key with its public key. If both keys are matched then only AWS will allow to connect with EC2.

Note: One key pair we can use to create multiple instances

Note: Key-Pairs are free of cost (No Bill)

Note: Key pair we can use for any OS Based VM

==================
Security Groups
==================

-> Security Group is like a virtual firewall for our EC2 instance

-> Security Group will control Incoming and Outgoing traffic of our EC2 instance

-> To allow the incoming traffic we will configure Inbound Rules

-> To allow the outgoing traffic we will configure Outbound Rules

Windows => RDP : 3389

Linux => SSH : 22

Webserver => HTTP : 80

HTTPS => HTTPS : 443

Note: Security Groups are free

Note: One security group we can use for Multiple Instances

=============================
Creating Windows Virtual Machine
=============================

1) Goto EC2 service

2) Launch instance

3) Select AMI (Windows - Free tier eligible)

4) Select Instance Type (t2.micro - Free tier eligible)

5) Storage (Default 30 GB - Free tier)

6) Network (Default VPC)

7) Create New Pair

8) Create New security group with RDP protocol

=> Once EC2 VM created then click on 'Connect' button and get below details

```
DNS : ec2-15-207-89-254.ap-south-1.compute.amazonaws.com
Username : Administrator
Password : rcAF2=D3s%g&%O98o?)*xUYd&!vdw?dp
```

-> Connect to Windows VM using RDP

====================================
Launching Linux Virtual Machine in AWS
==================================

1) Goto EC2 service

2) Launch instance

3) Select AMI (Amazon Linux - Free tier eligible)

4) Select Instance Type (t2.micro - Free tier eligible)

5) Storage (Default 8 GB - Free tier)

6) Network (Default VPC)

7) Create / Use Key Pair

8) Create New security group with SSH protocol

=> Once EC2 VM created then click on 'Connect' button and get details

=> Connec to Linux VM using MobaXterm

================
Types of IP's in AWS
=================

-> IP stands for Internet Protocol

-> Every Machine should have one IP address

-> IP is like an address for computer

-> AWS providing 3 types of IPs

    1) private ip

    2) public ip

        3) elastic ip (static ip)

-> When we launch EC2 instance then AWS will provide one private ip and one public ip for our instance

-> Private IP is a fixed IP and it is used by AWS for internal purpose. It will not change when we re-start our EC2 instance.

-> Public IP is a dynamic IP. When we re-start our EC2 instance new Public IP will be generated.

Note: To connect with EC2 instance from outside we will use Public IP.

-> Elastic IP means fixed public IP address.

-> We can create Elastic IP and we can associate that elastic ip for our EC2 instance

-> Elastic IP address will not change when we re-start our ec2 instance

Note: Elastic IPs are commercial (paid)

Working process
+++++++++++++++
1) Create Elastic IP
2) Associate Elastic IP for EC2 instance
3) If we don't want to use elastic ip then De-Associate Elastic IP from EC2 instance
4) Release Elastic IP to AWS (Its mandatory)

***** Note: If we create Elastic IP then we have to associate that Elastic IP otherwise bill will be generated for that******
(We shouldn't keep Elastic IP as un-used ip)

================
Load Balancing
=================

-> If we deploy our application in one server then burden will increase on that server

-> If burden increased on server then below are the problems we are going face

        1) Request processing will become slow

        2) Responses will be delayed for customers

        3) Server might get crash

        4) Brand / Trust issues on our business

        5) Revenue Loss

        6) Single point of failure

Note: Good Business needs Good website

-> To overcome all these problems we will run our application in Multiple Servers

-> The process of running our application in Multiple Servers is called as Load Balancing.

-> To implement Load Balacing we will use Load Balancer (ELB) in AWS

-> LBR will recieve the request and it will distribute the requests to servers in round robbin fashion

==============================
Types of Load Balancers in AWS
=============================

1) Application Load Balancer (ALB)

2) Network Load Balancer (NLB)

3) Gateway Load Balancer (GLB)

4) Classic Load Balancer (old, getting retried on 15-Aug-2022)

-> To implement load balancing for HTTP & HTTPS requests we will go for Application Load Balancer (ALB)
-> By using Application Load Balancer we can implement Path Based Routing

==========================
Implemenging Load Balancer
===========================

1) Create 1st EC2 intance with below user data script

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 1</h1></html>" > index.html
service httpd start
```

2) Create 2nd EC2 intance with below user data script

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 2</h1></html>" > index.html
service httpd start
```

3) Create 3rd EC2 intance with below user data script

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT :: Server 3</h1></html>" > index.html
service httpd start
```

4) Create Target Group with above 3 EC2 instances
(Target Group means group of servers which are running our aplication)

5) Create Application Load Balancer using Target Group

6) Access the application Load Balancer DNS URL

Note: When request comes to Load Balancer it will distribute the requests to servers which are part of given target group.

========================
Monolith Vs Microservices
=========================

-> Application can be in 2 ways

1) Monolith Architecture

2) Microservices Architecture

-> Monolith Architecture means all the functionalities will be developed in single project
-> A big war file will be created
-> Monolith Architecture based project is difficult to maintain
-> For any small change in the code then we have to re-deploy entire application
-> Single Point Of failure

Note: To overcome the problems of Monolith Architecture we are using Microservices Architecture

-> Microservices is an architectural design pattern
-> Micservices architecture means project functionalities will be developed in multiple apis
-> Every API is called as one project
-> Every API contains limited functionality
-> Making changes to the functionality is easy in microservices
-> Maintenence of the project will become easy
-> For any code changes we no need re-deploy all the apis (deploy only changed api)

Note: For Monolith app load balancing one target group will be created and application will be deployed in all the servers who are belong that target group.

===================================
Microservices Based Load Balancing
===================================

-> Multiple APIs will be available In Microservices Architecure

-> Every API is called as one microservice

-> For every microservice one target group will be created

Hotels API ===> Hotels Target Group with 3 servers

Flights API ===> Flights Target Group with 3 servers

Trains API ===> Trains Target Group with 3 servers

===
How to implement LBR for Microservices based application
===

1) Create EC2 Instance with below user-data (Name it as HotelServer-1)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Hotel Server - 1</h1></html>" > index.html
service httpd start
```

2) Create EC2 instance with below user-data (Name it as HotelServer-2)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Hotels Server - 2</h1></html>" > index.html
service httpd start
```

3) Create HotelServers Target group with above 2 instances

4) Create EC2 instance with below user-data (Name it as FlightServer-1)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Flights Server - 1</h1></html>" > index.html
service httpd start
```

4) Create EC2 instance with below user-data (Name it as FlightServer-2)

```
#! /bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Flights Server - 2</h1></html>" > index.html
service httpd start
```

5) Create FlightsServers Target group with above 2 instances

6) Create Load Balancer by select HotelServers Target Group

7) Goto LBR Listeners and configure Route Based Routing for Flights Target Group

8) Test it with DNS name

Note: Once practise completed, delete LBR, Target Groups and EC2 instances

================
Auto Scaling
=================

=> AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

=> Using AWS Auto Scaling, itâ€™s easy to setup application scaling for multiple resources across multiple services in minutes.

=> Amazon EC2 Auto Scaling helps you ensure that you have the correct number of

Amazon EC2 instances available to handle the load for your application.

=============================
Advantages with Auto Scaling
===========================

1) Fault Tolerenece

2) Availability

3) Cost Management

============================
How to setup Auto Scaling Group
===========================

1) Create Launch Template

2) Create AutoScaling Group with Launch Template

3) Configure Desired, Min and Max Capacity

4) Attach AutoScaling Group to particular Target Group

5) Configure Scaling Policy

+++++++++++++++++
Amazon EBS Volume
++++++++++++++++++

Elastic Block Store volume is a block level storage device that can be associated with an EC2 instance

EBS Volumes can be used as both primary storage and secondary storage

The primary EBS volume acts as root volume and should be created and attached to the instance at the time of instance launch.

Storage can be increased in the future if needed. This primary EBS volume cannot be detached from the instance.

The secondary volume can be attached, detached and modified at any time (Additional Volume)

An instance can have one primary EBS volume and multiple secondary volumes

One Primary EBS Volume can be only associated with one instance and we can attach a single secondary EBS Volume to multiple Instances, provided they are in same Availability Zone.

++++++++++++++++++++++++++++++
There are 5 types of EBS volumes
++++++++++++++++++++++++++++++

General Purpose SSD (gp2) -- Provides balance of both price and performance and is generally chosen by default.

Provisioned IOPS SSD (io1) -- Most expensive of the volume types with highest performance and well-suited for tasks with heavy workloads.

Throughput Optimized HDD (st1) -- A low-cost volume that focuses on optimizing throughput and is generally used for large sequential workloads dealing with big data warehouses. These volumes cannot be used as root volumes for EC2 instances.

Cold HDD (sc1) -- least expensive of the volume types and specifically designed for workloads which are accessed less frequently. These volumes also cannot be used as root volumes for EC2 instances.

Magnetic (Standard) -- Previous generation magnetic volumes which cannot be used as root volumes for EC2 instances

Resizing EBS Volumes

In this digital world of ever enlarging data itâ€™s not enough to build solutions with hard-coded amount of storage that cannot scale.

If the volume associated with the EC2 instance fills up, we have to increase the side of the volume.

Advantages of using EBS Volumes
High availability and flexibility

Data can be kept persistently on a file system even after shut downing the instance

Enables snapshots, which capture the data stored at a point in time and can be restored at any time.

The snapshots enables us to create a volume and attach it to another instance if needed.

Can be resized at any time as and when required

Comes equipped with encryption (and encryption-at-rest).

EBS Volumes can be attached, detached and associated with other instances at any point in time (exception the primary volume)

+++++++++++++++++++++++++++++++++++
Today's Lab Task
+++++++++++++++++++++++++++++++++++
1) Create an EC2 Linux instance with Amazon Linux 2 AMI (it will have EBS root volume with 8 GB)
2) Create Additional EBS volume with 10 GB
3) Attach EBS Additional volume of 10 GB to EC2 for storing data
4) Connect to EC2 instance and verify volumes (EBS both volumes shud display)
5) Create a directory and mount EBS Additional volume to created directory
6) Store the data in EBS Additional volume mounted directory (create few files)
7) Detach EBS Additional volume from EC2 and stop/terminate EC2 instance
8) Create new EC2 instance and attach previously created EBS Additional volume
9) Connect to new EC2 instance and verify volumes (EBS volume shud display)
10) Create a directory and mount EBS Additional volume to created directory
11) Verify data which stored previously in EBS available or not (it should present)
12) Stop Instances and delete volumes

+++

List the attached EBS volumes: â€“

```
$ lsblk

Check if the volume has any data using the â€œsudo file -s /dev/xvdfâ€ command.
$ sudo file -s /dev/xvdf

Format partition using the command â€œsudo mke2fs -j /dev/xvdfâ€.
$ sudo mke2fs -j /dev/xvdf

Mounting this disk partition as â€œnewdiskâ€

b. create directory with a name newdisk
$ sudo mkdir newdisk

c. Mount EBS volume to newdisk using the command
$ sudo mount /dev/xvdf newdisk

Navigate to newdisk directory (it is available under root (/)
$ cd newdisk

Creat new file in newdisk directory and store the data
$ touch f1.txt

--------------------------------------------------------------------------------
1) Detach Volume from EC2 instance and Stop EC2 instance

2) Create New EC2 instance

3) Attach EBS Additional Volume

4) Connect to newly created EC2 instance

5) Check volumes available in EC2 using (lslbk command)

6) Create directory

        $ mkdir newdrive

6) Mount EBS volume to directory

        $ sudo su
        $ sudo mount /dev/xvdf newdrive

5) Verify data available in EBS volume came to our directory or not

        $ ls -l newdrive

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++

=> Root volumes contains operating system

=> We will use EBS Additional volumes to store our application & application
configuration files






################
EBS Snapshots
################

-> Snapshots are used to create Volume backups
```

-> Snapshots are regional specific (volumes are zone specific)

-> Snapshot can't be attached with EC2 instance (volume can be attached with EC2)

-> From Volume we can create a snapshot and from snapshot we can create volume

snapshot --------------> volume ---------------> snapshot

#################
Snapshot Lab Task
#################

1) Create EC2 instance in 1a availability zone (It will have default root volume in 1a zone)

2) Connect to EC2 instance and create few files in Ec2 instance (files will store in root volume)

3) Create Snapshot from the root volume of our EC2 instance (volume is in 1a zone)

4) From Created Snapshot create EBS volume in 1b zone

5) Create EC2 instance in 1b availability zone (it will create default root volume in 1b zone)

6) Attach the volume (created using snapshot) to EC2 instance which we have created in 1b AZ

7) Mount the volume to a directory and see the data

========
DevOps
========

-> DevOps is a process to automate build & deployment of our application

-> DevOps process is used to simplify application delivery process

-> DevOps culture will help us to speed up application delivery process

-> DevOps nothing but set of best practises

DevOps = Development + Operations

-> DevOps is used to colloborate Development team activites & Operations Team activities

-> As part of DevOps process we will use set of tools those are called as 'DevOps Tools' or 'DevOps Tool Chain'

1) Build Tools   -> Maven & Gradle   => Code Compilation & Packaging

2) Version Control Tools -> Git Hub & BitBucket => To store project source code & monitor changes

3) Code Review Tools -> SonarQube => For Code Quality Checking & identify dev team mistakes in the code

4) Artifact Repo Tools -> Nexus / JFrog => For storing project artifacts (jars & wars) & shared libs

5) WebServers -> Tomcat / JBoss => To run our applications (we need to deploy war file in server)

6) CI CD Tools -> Jenkins / Bamboos / UDeploy => To automate build & deployment process

7) Infrastructure Tools -> Terraform (IAAC) => To create infrastructure in cloud

8) Configuration Tools -> Ansible (IAAC) => To manage configurations in our machines

9) Containerization -> Docker => To package, build & run our application in any machine

10) Orchestration -> Kubernetes / Openshift => To manage docker containers

11) App Monitoring Tools -> Grafana / Promethues => Alerting & Monitoring Tools for our applications

12) Log Monitoring Tools -> ELK Stack / Splunk => To monitor application logs

```
                                        E - Elastic Search
                                        L - Logstash
                                        K - Kibana
```

13) Project Management Tool -> JIRA => Project work & Team work will be managed using this jira

14) Ticketing Tool -> Service Now -> To create tickets for request access / permissions

---===========

===========
Introduction
===========

-> Java is a programming language

-> Java language developed by sun microsystem company in 1991

-> Oracle company acquired Sun Microsystem in 2010

-> Java is under license of Oracle company

-> Java is a high level programming language

-> Java is simple programming language

-> Java is free & open source programming language

-> Java is platform independent

-> Java program files will have .java as extension

        Ex: Demo.java, Hello.java, Driver.java, Calculator.java etc..

-> Java Programs (.java file) contains source code

-> We can't execute .java files directley

-> Java Programs should be converted into Machine understandable format to execute

-> We need to compile java source code into byte code using java compiler (javac)

```
Ex: javac Demo.java
```

-> When we compile java code it will create .class file

-> We need to execute .class file to run the java program

```
Ex: java Demo
```

-> When we run java program using java command, JVM will start and it will execute java program

Note: JVM stands for Java Virtual Machine

-> JVM will convert byte code into machine understandable code

-> Java project contains several java programs (.java files)

-> We need to compile project source code into byte code

-> When we compile project source code we will get .class files

-> To deploy java project, we will package all .class files as JAR file or WAR file

JAR : Java Archieve
WAR : Web Archieve

-> Standalone java projects will be packaged as a JAR file

Note: Stadalone applications means the project which runs only in one computer

Ex: notepad, calculator....

-> Web Applications will be packaged as WAR file

Note: Multiple users can access web applications at a time

Ex: gmail, facebook, flipkart, naukri etc....

===========
Maven
===========

-> Maven is a free and open source software given by Apache Organization

-> Maven s/w is developed using Java programming language

-> Maven is used to perform Build Automation for java projects

-> Maven is called as Java Build Tool

Note: Maven is used as a build tool for only java projects.

========================
What we can do using maven
========================

1) We can create initial project folder structure using maven

2) We can download "project dependencies" using maven
(ex : springboot, hibernate, kafka, redis, email, log4j, junit, security...)

-> To develop one java project we will use several frameworks like spring, hibernate etc along with Java

-> We need to download those frameworks and we should add to our java project

-> These frameworks we are using in our project are called as our project dependencies

-> Instead of we are downloading dependencies, we can tell to maven s/w to download dependencies

Note: Required dependencies we will add in "pom.xml" file then maven s/w will download them

-> pom stands for project object model

-> When we create maven project then pom.xml file will be created automatically

-> pom.xml will act as input file for maven software

3) We can compile project source code using maven

4) We can package java project as jar or war file using maven

======================
Maven Installation
======================

1) Download and install Java software

-> When we install java we will below 2 things

    a) JDK (Java Development Kit)

    b) JRE (Java Runtime Environment)

-> JDK contains set of tools to develop java programs

-> JRE contains platform/environment which is used to run java programs

Link To Download Java : https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html

2) Set JAVA_HOME in Environment Variables (System Env Variables)

User Environment Variables: Specific to particular account which logged in our PC

System Envrionment Variables : For All Accounts

Note: Environment Variables will be used by operating system

```
JAVA_HOME = C:\Program Files\Java\jdk1.8.0_202
```

3) Set Path for JAVA (Go to System Env Variables -> Env Variables -> System Variables -> Select Path and Click on Edit then add JDK path like below)

```
Path = C:\Program Files\Java\jdk1.8.0_202\bin
```

4) Verify Java installation by executing below command in "Command Prompt"

```
> java -version
```

Note: It should dipslay java version which we have installed

5) Download Maven software from Apache website

```
                Link To download Maven : https://maven.apache.org/download.cgi
                File Name :     apache-maven-3.8.5-bin.zip (Binary Archive)
```

6) Extract Maven Zip file -> Copy extracted maven folder and paste it in "C" drive

7) Set MAVEN_HOME in System Environment Variables

```
                MAVEN_HOME = C:\apache-maven-3.8.5
```

8) Set Path for Maven in System Environment Variables

```
                Path : C:\apache-maven-3.8.5\bin
```

9) Open Command Prompt and verify Maven Installaton using below command

```
                > mvn -version
```

======================
Maven Terminology
======================

archetype
groupId
artifactId
packaging

-> Archteype represents what type of project we want to create

                => maven-archetype-quickstart : It represents java standalone application

                => maven-archetype-webapp : It represents java web application

Note: Maven providing 1500+ archetypes

-> groupId represents company name or project name

-> artifactId represents project name or project module name

-> packaging represents how we want to package our java application (jar or war)

==
Creating standalone application using maven
==

1) Create one folder for maven practise

2) Open Command prompt from that folder

3) Execute below command to create maven project

```
>> mvn archetype:generate -DgroupId=in.ashokit -DartifactId=01-Maven-App
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

4) Once project created verify project folder structure

```
                        01-Maven-App
                                - src
```

```
                                        - main
                                                -java

                                        - test
                                                -java

                                - pom.xml
```

src/main/java : Application source code (.java files)

src/test/java : Application Unit Test code (.java files)

pom.xml : Project Object Model (Maven configuration file)

5) We can add dependencies in pom.xml file

6) We can find maven dependencies in www.mvnrepository.com website

7) Add below dependency in pom.xml file

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.2.22.RELEASE</version>
</dependency>
```

==
How maven will downoad dependencies
==
-> Maven will download dependencies using repository

-> In Maven we have 3 types of repositories

1) Central Repository

2) Remote Repository

3) Local Repository

-> central repository is maintaining by apache organization

-> every company will maintain their own remote repository

-> Local repository will be created in our system (Location :
C://users/<uname>/.m2)

-> When we add dependency in pom.xml, maven will search for that dependency in
local repository. If it is available it will add to project build path.

-> If dependency not available in local repository then maven will connect to
Central Repository or Remote Repository based on our configuration.

Note: By default maven will connect with central repository. If we want to use
remote reposiotry then we need to configure remote repository details.

Note: Every software company will maintain their own remote repository (Ex: JFrog)

=================================
Configuring Remote Repository
=================================
```
<repositories>
```

```
        <repository>
            <id>id</id>
            <url>jfrong-repo-url/</url>
        </repository>
</repositories>
```

===========
Maven Goals
===========

-> To perform project build activities maven prorvided several goals for us

```
clean
compile
test
package
install
```

-> clean goal is used to delete target folder

-> compile goal is used to compile project source code. Compiled code will be stored in target folder

```
                                        compile
                                .java ------------> .class
```

-> test goal is used to execute unit test code of our application (junit code)

-> package goal is used to generate jar or war file for our application based on packaging type available in pom.xml file.

Note: jar or war file will be created in target folder.

-> install goal is used to install our project as a dependency in maven local repository.

Note: Every maven goal is associated with maven plugin. When we execute maven goal then respective maven plugin will execute to perform the operation.

```
Syntax : mvn  <goal-name>
```

Note: We need to execute maven goals from project folder

=================================
creating web application using maven
=================================

```
>> mvn archetype:generate -DarchetypeArtifactId=maven-archetype-webapp
-DgroupId=in.ashokit -DartifactId=01-maven-web-app -DinteractiveMode=false
```

---##################
Sonatype Nexus
##################

-> Nexus is an Open Source Software & It is free

-> It is an Artifact Repository Server

-> It is used to store and retrieve build artifacts

-> Nexus software developed using Java

Note: To install Nexus s/w we need to install java first

-> Currently people are using Nexus 3.x

Java : jar, war and ear

Docker : Docker images

Node JS : NPM package

Q) What is difference between Nexus and GitHub ?

-> Github is a SCM software which is used to store source code of the project

-> Nexus is Artifact Repository which is used to store build artifacts

##################
Nexus Setup
##################

-> Take t2.medium instance

-> Java s/w is required to install Nexus

-> Connect to t2.medium instance using mobaxterm

```
#  Nexus S/w Installation Process in Amazon Linux OS

$ sudo su -

$ cd /opt

# install java 1.8v
$ sudo yum install java-1.8.0-openjdk
```

Links to download :
https://help.sonatype.com/repomanager3/product-information/download

```
# latest version
$ wget https://download.sonatype.com/nexus/3/nexus-3.40.1-01-unix.tar.gz

$ tar -zxvf nexus-3.40.1-01-unix.tar.gz

$ mv /opt/nexus-3.40.1-01 /opt/nexus
```

#As a good security practice, Nexus is not advised to run nexus service as a root user, so create a new user called nexus and grant sudo access to manage nexus services as follows.

```
$ useradd nexus
```

#Give the sudo access to nexus user

```
$ visudo
nexus ALL=(ALL) NOPASSWD: ALL
```

#Change the owner and group permissions to /opt/nexus and /opt/sonatype-work directories.

```
$ chown -R nexus:nexus /opt/nexus
$ chown -R nexus:nexus /opt/sonatype-work
$ chmod -R 775 /opt/nexus
$ chmod -R 775 /opt/sonatype-work

# Open /opt/nexus/bin/nexus.rc file and  uncomment run_as_user parameter and set as
nexus user.

$ vi /opt/nexus/bin/nexus.rc
run_as_user="nexus"

# Create nexus as a service

$ ln -s /opt/nexus/bin/nexus /etc/init.d/nexus

# Switch as a nexus user and start the nexus service as follows.

$ su - nexus

# Enable the nexus services
$ sudo systemctl enable nexus

# Start the nexus service
$ sudo systemctl start nexus

#Access the Nexus server from Laptop/Desktop browser.

URL : http://IPAddess:8081/

Note: Enable this 8081 port number in Security Group

# Default Username
User Name: admin

# we can copy nexus password using below command
$ sudo cat /opt/sonatype-work/nexus3/admin.password


-> We can change nexus default properties

        /opt/nexus/etc/nexus-default.properties


####################################
Integrate Maven with Nexus
####################################

-> Create Repositories in Nexus to store build artifacts

-> We will create 2 types of repositories in Nexus

        1) snapshot

        2) release

-> If project is under development then that project build artifacts will be stored
into snapshot repository

-> If project development completed and released to production then that project
build artifacts will be stored to release repository

Snanpshot Repo URL : http://65.0.92.87:8081/repository/ashokit-snapshot-repository/

Release Repo URL : http://65.0.92.87:8081/repository/ashokit-release-repository/

Note: Based on <version/> name available in project pom.xml file it will decide
artifacts should be stored to which repository
```

-> Nexus Repository details we will configure in project pom.xml file like below

```
  <distributionManagement>
        <repository>
                <id>nexus</id>
                <name>Ashok IT Releases Nexus Repo</name>

<url>http://43.205.146.33:8081/repository/ashokit-release-repository/</url>
        </repository>

        <snapshotRepository>
                <id>nexus</id>
                <name>Ashok IT Snapshots Nexus Repo</name>

<url>http://43.205.146.33:8081/repository/ashokit-snapshot-repository/</url>
        </snapshotRepository>
</distributionManagement>
```

-> Nexus Server Credentials will be configured in Maven "settings.xml" file

-> Maven Location : C:\apache-maven-3.8.5\conf

-> In settings.xml file, under <servers> tag add below <server> tag

```
        <server>
                <id>nexus</id>
                <username>admin</username>
                <password>admin</password>
        </server>
```

-> Once these details are configured then we can run below maven goal to upload build artifacts to Nexus Server

```
        $ mvn clean deploy
```

Note: When we execute maven deploy goal, internally it will execute 'compile + test + package + install + deploy' goals.

##################
Remote Repository
##################

-> Remote repository used for shared libraries (common jars for projects)

-> If we want to use few jar files in multiple projects in the company then we will use Remote Repository

-> Remote repository is specific to our company projects

-> Create remote repo in nexus and upload a jar file

-> Go to Repositories
-> Create New Repository
-> Choose Maven (Hosted) Repository
-> Give a name for Repository (Ex: ashokit-remote-repository) & Complete the process

Note: With above steps Remote Repository got created.

-> Go to BrowseSection
-> Select Remote Repository (By default it is

```
empty)
                                -> Click on Upload Component
                                -> Upload Jar file and give groupId, artifactId and
Version

                                                groupId : in.ashokit
                                                artifactId : pwd-utils
                                                version : 1.0


-> Take dependency details of uploaded jar file and add in project pom.xml as a
dependency like below

<dependency>
  <groupId>in.ashokit</groupId>
  <artifactId>pwd-utils</artifactId>
  <version>1.0</version>
</dependency>

-> We need to add Remote Repository Details in pom.xml above <dependencies/> tag

                <repositories>
                        <repository>
                                <id>nexus</id>

<url>http://43.205.146.33:8081/repository/ashokit-remote-repository/</url>
                        </repository>
                </repositories>

-> After adding the remote repository details in pom.xml then execute maven package
goal and see dependency is downloading from nexus repo or not.

                $ mvn clean package

-> We will create users and will give access for users for our repositories

-----------------------------------------------------------------------------------
---------------------
```

```
========================================================
Source Code Repository Tools (or) Version Control Softwares
========================================================
```

-> Multiple developers will work for project development

-> Developers will be working from multiple locations

-> All developers code should be stored at one place (Code Integration Should Happen)

-> To integrate all the developers source code at one place we will use Source Code Repository Softwares

```
============================================
Advantages with Source code repository sofwares
============================================
```

1) All the developers can connect to repository server and can integrate the code

2) Code Integration will become easy

3) Repository server will provide monitored access

```
                                          - Who
                                          - When
                                          - Why
                                          - What
```

```
==============
Repository Tools
==============
```

SVN (outdated)

Git Hub

BitBucket

```
=================================
Environment Setup to work with Git Hub
=================================
```

1) Create Github account (www.github.com)

2) Download and install Git Client software (https://git-scm.com/downloads)

3) Once installation completed, right click on the mouse and verify git options display (If git options displaying our git client installation completed successfully)

```
=====================
Working with GitHub
=====================
```

-> Login into github account with your credentials

-> Create Repository in github

Note: Repository is used to store project source code. Every Project will have one repository

-> When we create a repository, unique URL will be generated with Repository Name (i.e Repo URL)

                Ex: https://github.com/ashokitschool/hdfc_bank_app.git

-> All the developers will connect to repository using Repository URL

-> We can create 2 types of Repositories in Git Hub

                        1) public repository

                        2) private repository

-> Public Repository means everybody can access but we can choose who can modify our repository

-> Private Repository means we will choose who can access and who can modify

                Repo URL : https://github.com/ashokitschool/01-devops-app.git

```
sudo yum install git
```

```
====================
Working with Git Bash
====================

-> Git Bash we can use as Git Client software to perform Git Operations

-> Download and install git client (https://git-scm.com/downloads)

-> Right Click on Mouse and choose "Open Git Bash Here"

git help : It will display frequently used git commands

git help <cmd-name> : It will open documentation for given command

==================================================
Configure Your Email and Name in GitBash with Commands
==================================================

$ git config --global user.email "youremail@yourdomain.com"

$ git config --global user.name "name"

Note: email and name we will configure only for first time.

$ git init : To initialize our folder as git working tree folder

$ git clone : To clone git repository to our machine from github.com

                Syntax : $ git clone <project-repo-url>

$ git status : It will display staged , un-staged and un-tracked files

                Syntax : $ git status

                Staged Files : The files which are added for commit

                Un-Staged Files : The files which are modified but not added for
commit

                Un-tracked files : Newly created files

Note: To commit a file(s), we should add to staging area first

$ git add : It is used to add file(s) to staging area

                Syntax : $ git add <file-name>

                                $ git add .

$ git commit : It is used to commit staged files to git local repository

                Syntax : $ git commit -m 'reason for commit'

$ git push : To push changes from git local repository to git central repository

                Syntax  : $ git push

$ git rm : To remove file(s) from repository
```

==
Steps to push code to github central repository
==

1) Create one public repository in git hub (take github repo url)

2) Clone github repository using 'git clone' command

```
                $ git clone 'repo-url'
```

3) Navigate to repository folder

4) Create one file in repository folder

```
                $ touch Demo.java
```

5) Check status of the file using 'git status' command

```
                $ git status (It will display as untracked file)
```

6) Add file to staging area using 'git add' command

```
                $ git add .
```

7) Commit file to git local repository

```
                $ git commit -m 'commit-msg'
```

8) Push file from git local repository to git central repository using 'git push' command

```
                $ git push
```

Note: If you are doing 'git push' for first time it will ask to enter your github account password.

Note: Git bash will ask our password only for first time. It will save our git credentials in Credential Manager in Windows machine.

-> Go to Credential Manager -> Windows Credentials -> Select Github -> We can modify and delete saved credentials from here
----

-> When we do git commit then it will geneate a commit-id with 40 characters length

-> From this commit-id it will display first 7 characters in git hub central repository

-> We can check commit history using 'git log' command

==
Steps to commit Maven Project to Github Repository
==

1) Create Maven Project

2) Create GitHub Repository

Note: After creating git repository, it will display set of commands to execute

3) Open gitbash from project folder and execute below commands

```
$ git init
$ git add .
$ git commit -m 'commit-msg'
$ git branch -M main
$ git remote add origin <repo-url>
$ git push -u origin master
```

When we are working on one task suddenly we may get some other priority task.

Usecase
++++++++
-> ager assigned task id : 101
-> I am working on that task (i am in middle of the task)
-> Managed told that stop the work for 101 and complete 102 on priority.
-> Once 102 is completed then resume your work on 101

-> When manager asked me to start 102 task, i have already done few changes for 101

(partially completed)

-> We can't push partial changes to repository because with our partial changes existing functionality may break.

-> We can't delete our changes because we have spent few hours of time to implement those changes

*************** In this scenario we will go for 'git stash' option *********************

=> Git stash is used to save working tree changes to temporary location and make working tree clean.

-> After priority work completed we can get stashed changes back using 'gitstash apply'

============
Git Branches
============

-> Branches are used to maintain seperate code bases for our project

-> In GiT repository we can create multiple branches

main
develop
qa
release
research

-> development team will integrate the code in 'develop' branch

-> bug-fixing team will integrate the code in 'qa' branch

-> R & D team will integrate the code in 'research' branch

-> In github we can create branches

-> To clone particular branch in git repo we will use below command

```
                $ git clone -b <branch-name> <repo-url>
```

What is Git branch ?

Why we need git branches ?

How to create git branches ?

How to clone particular branch

How to switch from one branch to another branch

==============
Branch Merging
==============

=> The process of merging changes from one branch to another branch is called as Branch merging

=> We will use Pull Request for Branch Merging

============================
Steps to do branch merging
============================

1) Create feature branch from main branch

2) clone feature branch

3) create new file in feature branch then commit and push to central repo

4) Go to central repository then create pull request to merge feature branch changes to main branch

Note: Once feature branch changes are merged to main branch then we can delete feature branch (if required)

================
What is .gitignore ?
================

-> This .gitignore is used to configure the files or folders which we want to ignore from our commits

-> The files and folders which are not required to commit to central repository those things we can configure in .gitignore file

Ex: In maven project, 'target' folder will be available which is not required to commit to central repository. This we can configure in .gitignore file.

---------------------------------------.gitignore--

```
HELP.md
target/
!.mvn/wrapper/maven-wrapper.jar
!**/src/main/**/target/
!**/src/test/**/target/
```

```
### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/
build/
!**/src/main/**/build/
!**/src/test/**/build/

### VS Code ###
.vscode/
```

===================

git merge vs git rebase

===================

=> These commands are used to merge changes from one branch to another branch

-> git merge will maintain commit history

-> git rebase will not maintain that rebase history

-> When we are working on particular sprint and we want to merge changes from one branch to another branch then we will use 'git merge' command

-> Once sprint-1 is delivered then we want to take latest code of sprint-1 to start sprint-2 development. In this scenario we don't need commit history so we will use 'git rebase' command.

======================

What is git pull command

======================

-> pull command is used to take latest changes from repository to local

-> When we want to make some changes to code, it is always recommended to take git pull

Note: When we execute 'git pull' there is a chance of getting conflicts. We need to resolve the conflict and we should push the code without conflicts.

commit vs push

push vs pull

clone vs pull

checkout vs pull

fetch vs pull

What is Source Code repository
Why we need source code repository
What are the source code repository servers available
What is Code Integration
What is Monitored Access
What is git hub
What is git
What is version control
What is Repository
Public Repository vs Private Repository
Cloning Repository
Staged vs Unstaged vs Untracked File
Adding Files to Stating Area
Unstaging the files from staging
Discarding local changes
What is working tree
What is Local Repostiory
What is Central Repository
Commit from working tree to local repo
push from local repo to central repo
Taking latest code changes
push vs pull
What is conflict
How to resolve conflicts
What is branch in git hub
How to create branches
How to clone particular branch
how to switch to particular branch
How to merge branches
What is pull request
git merge vs rebase
what is .gitignore

```
git init
git help
git config
git clone
git status
git add .
git add <file-name>
git restore
git commit
git push
git pull
git log
git rm
git branch
git checkout
git merge
git rebase
```

================
Git Hub Lab Task
================
1) Create Maven Web Application
2) Add 'Spring-Core' dependency in project pom.xml file (www.mvnrepository.com)
3) Package maven project as war file using maven goal
4) Create Git repository in github.com (public repo)
5) Push maven project into github repo using gitbash
   (target folder shouldn't be commited, add this in .gitignore file)
6) Make changes in pom.xml and push changes to github repo using git bash
7) Create 'feature' branch in git repo from main branch
8) Make changes in 'feature' branch pom.xml file
9) Switch to feature branch from git bash and push changes to central repo
10) Create pull request and merge 'feature' branch changes to 'main' branch

----
------------------------------============
Sonar Qube
============

-> Sonar Qube is Continuous Code Quality Checking Tool

-> We can do Code Review using Sonar Qube tool

====================================
What is Code Coverage & Code Review
====================================

Code Coverage : How many lines of source code is tested by unit test cases

Note: Industry standard Code Coverage is 80 %

Code Review : Checking Coding Conventions / Standards

-> Sonar Qube is an open source, software quality management tool

-> It will continuosly analyze and measures quality of the source code

-> It will generate code review report in html format / pdf format

-> It is a web based tool and it supports 29 Programming Languages

-> It will support multi OS platform

-> It will support multiple databases (MySQL, Oracle, SQL Server, PostGres SQL...)

-> It supports multiple browsers

-> Sonar Qube will identify below category of issues in project source code

1) Duplicate Code
2) Coding Standards
3) Unit Tests
4) Code Coverage
4) Complex Code
5) Commented Code
6) Potential Bugs

=> Initially Sonar Qube was developed only for Java Projects

=> Today Sonar Qube is supporting for 29 Languages

================
Environment Setup
================

-> Java is the pre-requisite software

7.6 --> Java 1.8v

7.8 - 8.x --> Java 11v

Note: We can check this compatability in official sonar website

=====================
Hardware Requirements
=====================

Minimum RAM : 2 GB

t2.medium ---> 4 GB RAM

-> Create EC2 instance with 4 GB RAM (t2.medium)   (Amazon Linux AMI)

-> Connect with EC2 instance using MobaXterm

-> check space (free -h)

```
$ sudo su
$ cd /opt
$ sudo yum install java-1.8.0-openjdk
$ java -version
$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.8.zip
$ unzip sonarqube-7.8.zip
```

*********************************** Note: SonarQube server will not run with root user ***********************************

-> Create new user in ec2 instance
```
$ useradd sonar
$ visudo
```

-> Configure sonar user without pwd in suderos file
```
sonar ALL=(ALL) NOPASSWD: ALL
```

```
# Change ownership for sonar folder/directory
$ chown -R sonar:sonar /opt/sonarqube-7.8/
$ chmod -R 775 /opt/sonarqube-7.8
$ su - sonar
```

-> Goto bin directory then goto linux directory and run sonar server

```
$ cd /opt/sonarqube-7.8/bin/linux-x86-64

$ sh sonar.sh start
```

-> Check sonar server status

```
$ sh sonar.sh status
```

Note: Sonar Server runs on 9000 port number by default

Note: We can change default port of sonar server (conf/sonar.properties)

```
Ex:   sonar.web.port=6000
```

-> Enable port number in EC2 VM - Security Group

-> Access Sonar Server in Browser

```
                URL : http://EC2-VM-IP:9000/
```

-> Default Credentials of Sonar User is admin & admin

-> After login, we can go to Security and we can enable Force Authentication.

Note: Once your work got completed then stop your EC2 instance because we have t2.medium so bill be generated.

```
$ sh sonar.sh status
```

Note: If sonar not started, then go to log file and see

```
$ sudo rum -rf /opt/sonar-folder/temp/

$ cd ../bin/

$ sh sonar.sh start

$ sh sonar.sh status
```

-> Access sonar server in browser and login into that

=====================================
Integrate Sonar server with Java Maven App
======================================

-> Clone git repository : https://github.com/ashokitschool/SB-REST-H2-DB-APP

-> Configure Sonar Properties under <properties/> tag in "pom.xml"

```
  <properties>
        <sonar.host.url>http://ec2-public-ip:9000/</sonar.host.url>
        <sonar.login>admin</sonar.login>
        <sonar.password>admin</sonar.password>
  </properties>
```

-> Go to project pom.xml file location and execute below goal

```
                        $ mvn sonar:sonar
```

-> After build success, goto sonar dashboard and verify that

Note: Instead of username and pwd we can configure sonar token in pom.xml

=======================
Working with Sonar Token
=======================
-> Goto Sonar -> Login -> Click on profile -> My Account -> Security -> Generate Token

-> Copy the token and configure that token in pom.xml file like below

```
        <sonar.host.url>http://3.110.114.186:9000/</sonar.host.url>
        <sonar.login>ff4d464eda3eccdea05d77b742767c777545863e</sonar.login>
```

-> Then build the project using "mvn sonar:sonar" goal

==============
Quality Profile
==============

-> For each programming language sonar qube provided one quality profile with set of rules

-> Quality Profile means set of rules to perform code review

-> We can create our own quality profile based on project requirement

-> Create One Quality Profile

```
                        - Name : SBI_Project
                        - Language: Java
                        - Parent : None
```

Note: We can make our quality profile as default one then it will be applicable for all the projects which gets reviewed under this sonar server.

Note: If we have any common ruleset for all projects then we can create one quality profile and we can use that as parent quality profile for other projects.

-> We can configure quality profile to specific project

```
                        - click on project name
                        - Go to administration
                        - Click on quality profile
                        - Select profile required
```

==============
Quality Gate
==============

-> Quality Gate represents set of metrics to identify project quality is Passed or Failed

-> Every Project Quality Gate should be passed

-> In Sonar We have default Quality Gate

-> If required, we can create our own Quality Gate also

Note: If project quality gate is failed then we should not accept that code for deployment.

-> If project is having Sonar issues then development team is responsible to fix those issues

-> As a DevOps engineer, we will perform Code Review and we will send Code Review report to Development team (we will send sonar server URL to development team)

=================================
Sonar Server with Jenkins Integration
=================================

```
Pre-Requisites
================
1) Sonar Qube Server
2) Jenkins Server

=> On SonarQube Server Generate a Token

=> On Jenkins Server
                - install apache maven
                - Install Sonar Plugin
                - Configure SonarServer with Token
                - Install Sonar Scanner
                - Run Jenkins Pipeline Job




-> Connect to Jenkins server VM using mobaxterm and start sonar server

-> Access Sonar Server in browser & generate token

-> Connect to  Jenkins Server using MobaXterm

-> Execute below commands in Jenkins Server VM

$ sudo su
$ cd /opt
$ wget
https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.
6.3-bin.tar.gz
$ tar -xvf apache-maven-3.6.3-bin.tar.gz


-> Manage Jenkins -> Plugins -> Available -> Sonar Qube Scanner Plugin -> Install
it

-> Manage Jenkins -> Configure System -> Sonar Qube Servers -> Add Sonar Qube
Server

                                - Name : Sonar-Server-7.8
                                - Server URL : http://52.66.247.11:9000/   (Give
your sonar server url here)
                                - Add Sonar Server Token

                                (Token we should add as secret text)

                                (Save it)

-> Goto jenkins dashboard -> click on 'New Item -> Enter Item name -> Select
Pipeline ->  Click on Next

-> Enter below pipeline script in given text box

pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
       stage('GetCode'){
            steps{
                git ''https://github.com/ashokitschool/maven-web-app.git"
            }
         }
       stage('Build'){
            steps{
                sh 'mvn clean package'
            }
         }
        stage('SonarQube analysis') {
        steps{
                withSonarQubeEnv('Sonar-Server-7.8') {
                sh "mvn sonar:sonar"
                }
        }
        }
    }
}
```


=================
What is Jenkins?
=================

=> Jenkins is an open source automation tool for CI and CD

=> Jenkins tool developed using Java

=> Jenkins is part of Hudson Project

=> Intially it is called as Hudson then later it renamed to Jenkins

=================
About CI CD
=================

=> CI and CD are two most frequently used terms in modern development practises and DevOps practises.

=> CI stands for Continuous Integration. It is fundamental DevOps best practise where developers frequently merge code changes to central repository where automated builds and tests runs.

=> CD means Continuous Delivery or Continous Deployment.

=> Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

==================================
Build & Deployment Process
==================================

1) Take latest source code from repository

2) Compile source code

3) Execute Unit tests (Junits)

4) Perform Code Review

5) Package code as war file

6) Deploy the war file into server

Note: All the above build and deployment tasks can be automated using Jenkins tool.

=================
Jenkins Installation:
=================

1) Create an EC2 instance with Ubuntu AMI (t2.micro instance)

2) Connect to your EC2 instance using MobaXterm

3) Install Java In Ubuntu VM with below commands

```
$ sudo apt-get update

$ sudo apt-get install default-jre
```

4) Install Jenkins in Ubuntu VM with below commands

```
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key
add -

$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'

$ sudo apt-get update

$ sudo apt-get install jenkins

$ sudo systemctl status jenkins
```

5) Access Jenkins Server in browser using below URL

URL : http://ec2-public-ip:8080/

Note: Enable 8080 port in security group

6) Get the initial administrative password

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

pwd : 5fe6ddcc9db244cab6aca5ccf2d6a83a
```

-> Provide pwd which we have copied to unlock jenkins

-> Select "Install Suggested Plugins" card (it will install those plugins)

-> Create Admin account

=========================
Creating First Job in Jenkins
=========================

1) Goto Jenkins Dashboard

2) Click on New Item

-> Enter Item Name (Job Name)
-> Select Free Style Project & Click OK
-> Enter some description
-> Click on 'Build' tab
-> Click on 'Add Build Step' and select 'Execute Shell'

3) Enter below shellscript

```
echo "Hello Guys,"
```

```
touch ashokit.txt
echo "Hello Guys, Welcome to Jenkins Classes" >> ashokit.txt
echo "Done..!!"
```

4) Apply and Save

Note: With above steps we have created JENKINS Job

5) Click on 'Build Now' to start Job execution

6) Click on 'Build Number' and then click on 'Console Ouput' to see job execution details.

=> Jenkins Home Directory in EC2 : /var/lib/jenkins/workspace/

```
                $ cd /var/lib/jenkins/workspace/
```

7) Go to Jenkins home directory and check for the job name --> check the file created inside the job

===
Jenkins Job with with GIT Hub Repo + Maven - Integeration
===

Pre-Requisites : Java, Maven and Git client

# Git installation In EC2 VM

```
$ sudo apt install git -y
```

==================================
Maven Installation In Jenkins:
==================================

Jenkins Dashboard -> Manage Jenkins --> Global Tools Configuration -> Add maven

==================================
Sample Git Repo URLS For Practise
==================================

```
Git Hub Repo URL-1  : https://github.com/ashokitschool/JAVA-MAVEN-WEB-APP.git
Git Hub Repo URL-2  : https://github.com/ashokitschool/maven-web-app.git
```

===
JOB-2 :: Steps To Create Jenkins Job with Git Repo + Maven
===

1) Connect to EC2 instance in which jenkins server got installed

2) Start Jenkins Server

3) Access Jenkins Server Dashboard and Login with your jenkins credentials

4) Create Jenkins Job with Git Hub Repo

```
                -> New Item
                -> Enter Item Name (Job Name)
                -> Select 'Free Style Project' & Click OK
                -> Enter some description
                -> Go to "Source Code Management" Tab and Select "Git"
                -> Enter Project "Git Repo URL"
                -> Add Your Github account credentials
                -> Go to "Build tab"
                -> Click on Add Build Step and Select 'Inovke Top Level Maven
```

```
Targets'
                -> Select Maven and enter goals 'clean package'
                -> Click on Apply and Save

Note: With above steps we have created JENKINS Job

5) Click on 'Build Now' to start Job execution

6) Click on 'Build Number' and then click on 'Console Ouput' to see job execution
details.

=> Jenkins Home Directory in EC2 : /var/lib/jenkins/workspace/

=> Go to jenkins workspace and then go to job folder then goto target folder there
we see war file created.

-----------------------------------------------------------------------------------
--------------

=> Access below URL in browser to stop Jenkins Server

                URL : http://EC2-VM-IP:8080/exit/

                (Click on Retry using Post button)

==============================================================
Job-3 :: Steps To Create Jenkins Job with Git Repo + Maven + Tomcat Server
==============================================================

1) Go to Jenkins Dashboard -> Manage Jenkins --> Manage Plugins -> Goto Available
Tab -> Search For
   "Deploy To Container" Plugin -> Install without restart.

2) Create Jenkins Job

                -> New Item
                -> Enter Item Name (Job Name)
                -> Select Free Style Project & Click OK
                -> Enter some description
                -> Go to "Source Code Management" Tab and Select "Git"
                -> Enter Project "Git Repo URL"
                -> Add Your Github account credentials
                -> Go to "Build tab"
                -> Click on Add Build Step and Select 'Inovke Top Level Maven
Targets'
                -> Select Maven and enter goals 'clean package'
                -> Click on 'Post Build Action' and Select 'Deploy war/ear to
container' option
                -> Give path of war file (You can give like this also : **/*.war )
                -> Enter Context Path (give project name Ex: java_web_app)
                -> Click on 'Add Container' and select Tomcat version 9.x
                -> Add Tomcat server credentials (give the username & pwd which is
having manager-script role)
                -> Enter Tomact Server URL (http://ec2-vm-ip:tomcat-server-port)
                -> Click on Apply and Save

4) Run the job now using 'Build Now' option and see see 'Console Output' of job

5) Once Job Executed successfully, go to tomcat server dashboard and see
application should be displayed.

6) Click on the applicaton name (it should display our application)

-----------------------------------------------------------------------------------
---------------------
```

```
**** If we forgot Jenkins Password, then how to recover it ? *****

-> Go to /var/lib/jenkins/

        Open  : config.xml file

-> Set Value for useSecurity as false

        ex: <useSecurity>false</useSecurity>

-> Re-start jenkins and try to access


******* How to change Jenkins Port Number ***********

-> Go to root directory

-> Go to /etc/sysconfig

-> Open jenkins file and change Jenkins port number

-> Restart Jenkins server

===================================================
How to Create Jenkins Jobs with Build Parameters
===================================================

=> Build Parameters are used to supply input to run the Job. Using Build Parameters
we can avoid hard coding.

                Ex : We can pass branch name as build parameter.

-> Create New Item
-> Enter Item Name & Select Free Style Project
-> Select "This Project is parameterized" in General Section
-> Select Choice Parameter
-> Name : BranchName
-> Choices : Enter every branch name in nextline
-> Branches to Build : */${BranchName}


==================================
Creating Users in Jenkins
==================================

-> Manage Jenkins -> Manage Users

-> Create Users

-> Configure Global Security For Users

-> Manage Roles & Assign Roles

Note: By default admin role will be available and we can create custom role based
on requirement

-> In Role we can configure what that Role assigned user can do in jenkins

-> In Assign Roles we can add users to particular role


=========
Excercise
========
```

-> create 1 account for DevOps team member

-> Create 1 account for Development team member

-> Configure Roles for DevOps team member and Development team member

==================================
Jenkins Master and Slave Configuration
==================================
-> When we build the Jenkins job in a single Jenkins master node then Jenkins uses the resource of the base machine and If no executor is available then the jobs are queued in the Jenkins server.

-> Sometimes you might need several different environments to test your builds. This cannot be done by a single Jenkins server.

-> It is recommended not to run different jobs in the same system that required a different environment. In such scenarios where we need a different machine with a different environment that takes the specific job from the master to build.

-> On the same Jenkins setup, multiple teams are working with their jobs. All jobs are running on the same base operating system and the base operating system has limited resources.

-> To overcome this problem, Jenkins provided Distributed Architecture i.e Jenkins Master Slave Architecture

-> Jenkins uses A Master-Slave architecture to manage distributed builds. The machine where we install Jenkins software will be Jenkins master and that runâ€™s on port 8080 by default. On the slave machine, we install a program called Agent. This agent requires JVM. This agent executes the tasks provided by Jenkins master. We can launch n numbers of agents and we can configure which task will be run on which agent server from Jenkins master by assigning the agent to the task.

==============
Jenkins Master
===============

-> Your main Jenkins server is the Master. The Masterâ€™s job is to handle:

a) Scheduling build jobs.

b) Dispatching builds to the slaves for the actual execution.

c) Monitor the slaves (possibly taking them online and offline as required).

d) Recording and presenting the build results.

e) A Master instance of Jenkins can also execute build jobs directly.

==============
Jenkins Slave
==============
A Slave is a Java executable that runs on a remote machine. Following are the characteristics of Jenkins Slaves:

1) It hears requests from the Jenkins Master instance.

2) Slaves can run on a variety of operating systems.

3) The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.

4) You can configure a project to always run on a particular Slave machine or a particular type of Slave machine, or simply let Jenkins pick the next available Slave.

===
Working with Jenkins Master Slave Architecture
===

==========================
Step-1 : Create Jenkins Master
==========================

1) Create EC2 instance
2) Connect EC2 using Mobaxterm
3) Install Git client
4) Install Java Software
5) Install jenkins server
6) Add Git, JDK and Maven Plugins
7) Enable Jenkins Port Number in Security Group
8) Access Jenkins Server in Browser and Login

=========================
Step-2 : Create Jenkins Slave
=========================

1) Create EC2 instance
2) Connect to EC2 using Mobaxterm
3) Install Git client
4) Install Java Software
5) Create one directory in /home/ec2-user (ex: slavenode)

===
Step-3: Configure Slave Node in Jenkins Master Node
==

1) Go to Jenkins Dashboard
2) Go to Manage Jenkins
3) Go to Manage Nodes & Clouds
4) Click on 'New Node' -> Enter Node Name -> Select Permanent Agent
5) Enter Remote Root Directory (/home/ec2-user/slavenode)
6) Enter Label name as Slave-1
7) Select Launch Method as 'Launch Agent Via SSH'
8) Give Host as 'Slave VM DNS URL'
9) Add Credentials (Select Kind as : SSH Username with private key)
10) Enter Username as : ec2-user
11) Select Private Key as Enter Directley and add private key

Note: Open gitbash from your .pem file location and execute below command to get private key from pem file

```
$ cat <key-pair-file-name>.pem
```

Note: It will display private key on Git Bash terminal (Just copy and paste in jenkins)

12) Select Host Key Strategy as 'Manually Trusted Key Verification Strategy'

13) Click on Apply and Save (We can see configure slave)

************************With above steps Master and Slave Configuration Completed***************************

-> Go to Jenkins Server and Create Jenkins Job

Note: Under Generation Section of Job creation process, Select "Restrict Where This Project Can Run" and enter Slave Nodel Label name and finish job creation.

-> Execute the Job using 'Build Now' option

Note: Job will be executed on Slave Node (Go to Job Console Ouput and verify execution details)

=================
JENKINS - Pipeline
=================

-> Jenkins Pipeline is an automation solution that lets you create simple or complex pipelines.

-> Jenkins Pipeline is a combination of Plugins which automates number of tasks and makes the CI/CD pipeline efficient, high in quality and reliable.

-> Jenkins provides two ways of developing a pipeline

1) Scripted
2) Declarative

-> Traditionally, Jenkins jobs were created using Jenkins UI called FreeStyle jobs.

-> In Jenkins 2.0, Jenkins introduced a new way to create jobs using the technique called pipeline as code.

-> In pipeline as code technique, jobs are created using a script file that contains the steps to be executed by the job.

-> In Jenkins, that scripted file is called Jenkinsfile.

=================
What is Jenkinsfile?
==================

-> Jenkinsfile is nothing but a simple text file which is used to write the Jenkins Pipeline and to automate the Continuous Integration process.

-> Jenkinsfile usually checked in along with the projectâ€™s source code in Git repo. Ideally, every application will have its own Jenkinsfile.

-> Jenkinsfile can be written in two ways â€“

1) Scripted pipeline syntax
2) Declarative pipeline syntax

=============================
What is Jenkins Scripted Pipeline?
=============================

-> Jenkins pipelines are traditionally written as scripted pipelines. Ideally, the scripted pipeline is stored in Jenkins webUI as a Jenkins file. The end-to-end scripted pipeline script is written in Groovy.

-> It requires knowledge of Groovy programming as a prerequisite. Jenkinsfile starts with the word node. Can contain standard programming constructs like if-else block, try-catch block, etc.

```
+++++++++++++++++++++++++
Sample Scripted Pipeline
+++++++++++++++++++++++++
node {
    stage('Stage 1') {
        echo 'hello'
    }
}
```

===============================
What is Jenkins Declarative Pipeline?
===============================

-> The Declarative Pipeline subsystem in Jenkins Pipeline is relatively new, and provides a simplified, opinionated syntax on top of the Pipeline subsystems.

-> The latest addition in Jenkins pipeline job creation technique.

-> Jenkins declarative pipeline needs to use the predefined constructs to create pipelines. Hence, it is not flexible as a scripted pipeline.

-> Jenkinsfile starts with the word pipeline.

-> Jenkins declarative pipeline should be the preferred way to create a Jenkins job as they offer a rich set of features, come with less learning curve & no prerequisite to learn a programming language like Groovy just for the sake of writing pipeline code.

-> We can also validate the syntax of the Declarative pipeline code before running the job. It helps to avoid a lot of runtime issues with the build script.

=========================
Our First Declarative Pipeline
=========================

```
pipeline {
    agent any
    stages {
        stage('Welcome Step') {
            steps {
                echo 'Welcome to Jenkins Scripting'
            }
        }
    }
}
```

pipeline : Entire Declarative pipeline script should be written inside the pipeline block. Itâ€™s a mandatory block.

agent : Specify where the Jenkins build job should run. agent can be at pipeline level or stage level. Itâ€™s mandatory to define an agent.

stages : stages block constitutes different executable stage blocks. At least one stage block is mandatory inside stages block.

stage : stage block contains the actual execution steps. Stage block has to be defined within stages block. Itâ€™s mandatory to have at least one stage block inside the stage block. Also its mandatory to name each stage block & this name will be shown in the Stage View after we run the job.

steps : steps block contains the actual build step. Itâ€™s mandatory to have at least one step block inside a stage block.

Depending on the Agentâ€™s operating system (where Jenkins job runs), we can use shell, bat, etc., inside the steps command.

```
==================
Build Pipeline Script
==================
pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
        stage('GetCode'){
            steps{
                git branch: 'main',
                url:
'https://github.com/ashokitschool/maven_web_app_jenkins_pipeline.git'
            }
         }
        stage('Build'){
            steps {
                sh 'mvn clean package'
            }
        }
    }
}
====================================
Build Pipeline + Sonar Qube Server - Script
====================================

pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
       stage('GetCode'){
            steps{
                git 'https://github.com/ashokitschool/maven-web-app.git'
            }
         }
       stage('Build'){
            steps{
                sh 'mvn clean package'
            }
         }
        stage('SonarQube analysis') {
            steps{
                withSonarQubeEnv('Sonar-Server-7.8') {
                    sh "mvn sonar:sonar"
                }
            }
        }

    }
}

============================================================
JENKINS PIPELINE ( JENKINS + MAVEN + GIT HUB + SONAR + TOMCAT )
=============================================================

Note: Install 'ssh-agent' plugin and generate code using pipeline syntax

pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
       stage('GetCode'){
            steps{
                git 'https://github.com/ashokitschool/maven-web-app.git'
            }
         }
       stage('Build'){
            steps{
                sh 'mvn clean package'
            }
         }
        stage('SonarQube Analysis') {
            steps{
                withSonarQubeEnv('Sonar-Server-7.8') {
                    sh "mvn sonar:sonar"
                }
            }
        }
                stage('Code deploy') {
            steps{
                                sshagent(['Tomcat-Server-Agent']) {
                                  sh 'scp -o StrictHostKeyChecking=no
target/01-maven-web-app.war
ec2-user@13.235.68.29:/home/ec2-user/apache-tomcat-9.0.63/webapps'
                                }
            }
        }
    }
}
```

=========================
Email Notifications In Jenkins
=========================

-> We can configure Email notifications in Jenkins

-> With this option we can send email notification to team members after jenkins job execution completed

-> We need to configure SMTP properties to send emails

-> Go To Manage Jenkins
-> Add Email Extension Server
-> We will add company provided SMTP server details

Note: For practise we can use GMAIL SMTP Properties

-> Once SMTP properties added then we can configure email notification as 'Post Build Action' in Jenkins job

----

```
pipeline {
    agent any
    stages {
        stage('Hello') {
            steps {
                echo "Hello world"
                    }
            }
        }
    post{
        always{
            mail to: "ashokitschool@gmail.com",
            subject: "Test Email",
            body: "Test"
        }
    }
}
```

####################################
DevOps Project Setup with CI CD Pipeline
####################################

1) Maven

2) Git Hub (Repo URL : https://github.com/ashokitschool/maven-web-app.git)

3) Tomcat (URL : http://3.109.4.210:8080)

4) SonarQube (URL : http://65.2.172.44:9000/)

5) Nexus Repo (URL : http://65.0.203.145:8081/)

6) Jenkins

###################
JENKINS VM Setup
##################

1) Create Ubuntu VM using AWS EC2

2) Install Java & Jenkins using below commands

```
$ sudo apt-get update

$ sudo apt-get install default-jre

$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

$ sudo apt-get update

$ sudo apt-get install jenkins

$ sudo systemctl status jenkins

# Copy jenkins admin pwd
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

# Open jenkins server in browser using VM public ip

URL : http://public-ip:8080/

-> Finish Jenkins setup

##################
Pipeline creation
##################

```
=> Create CI CD Pipeline with below stages

===================
1) Create github stage
====================

stage('clone repo') {

        git credentialsId: 'GIT-Credentials', url:
'https://github.com/ashokitschool/maven-web-app.git'

}

==============================================
2) Create Maven Build Stage (Add maven in global tools)
================================================

stage ('Maven Clean Build'){
        def mavenHome = tool name: "Maven-3.8.6", type: "maven"
        def mavenCMD = "${mavenHome}/bin/mvn"
        sh "${mavenCMD} clean package"
}

============================
3) Create SonarQube stage
===========================
-> Connect to sonar vm and run sonar server

$ cd /opt/sonarqube-7.8/bin/linux-x86-64

$ sh sonar.sh start


************************ Sonar Token : ce6f1e2cc07e41f2779354ccd65672ceeb6afba9
************************

-> Manage Jenkins -> Plugins -> Available -> Sonar Qube Scanner Plugin -> Install
it

->  Generate Sonar Token and Add it as Secret text in Global Credentials

Ex: dfeedf267e77b3e1ae0ff9d07bceefed3d02b73a

-> Manage Jenkins -> Configure System -> Sonar Qube Servers -> Add Sonar Qube
Server

                                - Name : Sonar-Server-7.8
                                - Server URL : http://52.66.247.11:9000/   (Give
your sonar server url here)
                                - Add Sonar Server Token

                                (Token we should add as secret text)

                                (Save it)

-> Once above steps are completed, then add below stage in the pipeline

stage('SonarQube analysis') {
                        withSonarQubeEnv('Sonar-Server-7.8') {
                        def mavenHome = tool name: "Maven-3.8.6", type: "maven"
                        def mavenCMD = "${mavenHome}/bin/mvn"
                        sh "${mavenCMD} sonar:sonar"
        }
}
```

```
=======================
4) Create Nexus Stage
======================
-> Run nexus VM and create nexus repository

-> Create Nexus Repository

-> Install Nexus Repository Plugin using Manage Plugins   ( Plugin Name : Nexus
Artifact Uploader)

-> Generate Nexus Pipeline Syntax

stage ('Nexus Upload'){
nexusArtifactUploader artifacts: [[artifactId: '01-Maven-Web-App', classifier: '',
file: 'target/01-maven-web-app.war', type: 'war']], credentialsId:
'Nexus-Credentials', groupId: 'in.ashokit', nexusUrl: '13.127.185.241:8081',
nexusVersion: 'nexus3', protocol: 'http', repository:
'ashokit-snapshot-repository', version: '1.0-SNAPSHOT'
}

=========================
5) Create Deploy Stage
=======================
-> Start Tomcat Server

-> Install SSH Agent plugin using Manage Plugins

-> Generate SSH Agent and configure stage

stage ('Deploy'){
       sshagent(['Tomcat-EC2-Credentials']) {
                 sh 'scp -o StrictHostKeyChecking=no target/01-maven-web-app.war
ec2-user@65.2.30.68:/home/ec2-user/apache-tomcat-9.0.68/webapps'
           }
    }


################
Final Pipeline
################


node {

    stage ('Git Clone'){
        git credentialsId: 'c87aff7e-f5f1-4756-978f-3379694978e6', url:
'https://github.com/ashokitschool/maven-web-app.git'
    }

    stage ('Maven Clean Build'){
        def mavenHome = tool name: "Maven-3.8.6", type: "maven"
        def mavenCMD = "${mavenHome}/bin/mvn"
        sh "${mavenCMD} clean package"
    }

    stage('SonarQube analysis') {
                        withSonarQubeEnv('Sonar-Server-7.8') {
                        def mavenHome = tool name: "Maven-3.8.6", type: "maven"
                        def mavenCMD = "${mavenHome}/bin/mvn"
                        sh "${mavenCMD} sonar:sonar"
        }
    }

    stage('upload war to nexus'){
        nexusArtifactUploader artifacts: [[artifactId: '01-maven-web-app',
classifier: '', file: 'target/01-maven-web-app.war', type: 'war']], credentialsId:
'NEXUS_CREDENTIALS', groupId: 'in.ashokit', nexusUrl: '65.0.182.179:8081',
nexusVersion: 'nexus3', protocol: 'http', repository: 'ashokit-snapshot', version:
'1.0-SNAPSHOT'
    }


stage ('Deploy'){
       sshagent(['Tomcat-Server-Agent']) {
                sh 'scp -o StrictHostKeyChecking=no target/01-maven-web-app.war
ec2-user@13.235.75.254:/home/ec2-user/apache-tomcat-9.0.64/webapps'
           }
    }

}
```

---====================

Application Environments
=====================

-> Environment means the platform which is used to run our application

-> In Realtime, we will use multiple environments to run our application

-> DEV Environment

-> SIT Environment

-> UAT Environment

-> Pilot Environment

-> Prod Environment

-> Developers will use DEV environment for integration testing.

-> SIT env will be used by Software Testing team to test our application functionality. SIT stands for System Integration Testing.

-> UAT means User Acceptance Testing. UAT will done by Client. For client testing we will give UAT environment.

Note: After UAT testing, client will decide GO or No-GO. GO means go for production deployment. No-GO means production deployment cancelled.

-> When client identify some critical bugs in UAT then they will say No GO.

-> PILOT env is called as pre-prod environment. It is used for live data testing.

-> Prod env is called live environment. The application which is deployed in PROD env will be available for public users access.

Note: To deploy the project into these environments we will use Jenkins software.

1) Maven : Build Tool

2) Git Hub : Source Code Repository s/w

3) Nexus : Artifactory Repository to upload artifacts & to create shared libraries

4) SonarQube : Code Quality Checking / Code Review S/w

5) Tomcat : Web Server to run our web applications

6) Jenkins: CI CD software which is used to automate build & deployments

Note: Created one pipeline using all above tools to perform build & deployment

----
--===================
Ansible Tutorial
==================

1) What is Ansible
2) Configuration Management
3) Push Based vs Pull Based
4) How to setup Ansible
5) Host Inventory file
6) Ansible Modules
7) YAML
8) Playbooks
9) Hands On
10) Ansible Vault
11) Ansible Roles
12) Conclusion

=========================
Configuration Management
=========================

It is a method through which we automate admin tasks.
Configuration management tool turns your code into infrastructure.
So your code would be testable, repeatable and versionable.

Infrastructure refers to the composite of â€”

Software
Network
People
Process

==========
Ansible
==========

-> Ansible is one among the DevOps configuration management tools which is famous for its simplicity.

-> It is an open source software developed by Michael DeHaan and its ownership is on RedHat

-> Ansible is an open source IT Configuration Management, Deployment & Orchestration tool.

-> This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.

-> Ansible is an automation tool that provides a way to define infrastructure as code.

-> Infrastructure as code (IAC) simply means that managing infrastructure by writing code rather than using manual processes.

-> The best part is that you donâ€™t even need to know the commands used to accomplish a particular task.

-> You just need to specify what state you want the system to be in and Ansible will take care of it.

-> The main components of Ansible are playbooks, configuration management and deployment.

-> Ansible uses playbooks to automate deploy, manage, build, test and configure anything

-> Ansible was written in Python

=================
Ansible Features
==================

-> Ansible manages machines in an agent-less manner using SSH

-> Built on top of Python and hence provides a lot of Python's functionality

-> YAML based playbooks

-> Uses SSH for secure connections

-> Follows push based architecture for sending configuration related notifications

======================
Push Based Vs Pull Based
=====================

-> Tools like Puppet and Chef are pull based

-> Agents on the server periodically checks for the configuration information from central server (Master)

-> Ansible is push based

-> Central server pushes the configuration information on target servers.

=====================
What Ansible can do ?
=====================
1) Configuration Management
2) App Deployment
3) Continous Delivery

==========================
How Ansible works ?
=======================

Ansible works by connecting to your nodes and pushing out a small program called Ansible modules to them.

Then Ansible executed these modules and removed them after finished.The library of modules can reside on any machine, and there are no daemons, servers, or databases required.

The Management Node is the controlling node that controls the entire execution of the playbook.

The inventory file provides the list of hosts where the Ansible modules need to be run.

The Management Node makes an SSH connection and executes the small modules on the hostâ€™s machine and install the software.

Ansible removes the modules once those are installed so expertly.

It connects to the host machine executes the instructions, and if it is successfully installed, then remove that code in which one was copied on the host machine.

Ansible basically consists of three components

Ansible requires the following components in order to automate Network Infrastructure.

1) Controlling Nodes
2) Managed Nodes
3) Ansible Playbook

====================
Controlling Nodes
===================
are usually Linux Bastion Servers that are used to access the switches/routers and other Network Devices.
These Network Devices are referred to as the Managed Nodes.

+++++++++++++++++++
Managed Nodes
++++++++++++++
Managed Nodes are stored in the hosts file for Ansible automation.

==================
Ansible Playbook
==================

Ansible Playbooks are expressed in YAML format and serve as the repository for the various tasks that will be executed on the Managed Nodes (hosts).

Playbooks are a collection of tasks that will be run on one or more hosts.

=================
Inventory file
=================

Ansible's inventory hosts file is used to list and group your servers.

Its default locaton is /etc/ansible/hosts

Note: In inventory file we can mention IP address or Hostnames also

====================================
Few Important Points About Inventory File
===================================

-> Comments begins with '#' character
-> Blank lines are ignore
-> Groups of hosts are delimited by '[header]' elements
-> You can enter hostnames or ip addresses
-> A hostname/ip can be a member of multiple groups
-> Ungrouped hosts are specifying before any group headers like below

Ansible's inventory hosts file is used to list and group your servers. Its default locaton is /etc/ansible/hosts

```
Note: In inventory file we can mention IP address or Hostnames also


Sample Inventory File
++++++++++++++++++++
#Blank lines are ignore

#Ungrouped hosts are specifiying before any group headers like below

192.168.122.1
192.168.122.2
192.168.122.3

[webservers]
192.168.122.1
#192.168.122.2
192.168.122.3

[dbserver]
192.168.122.1
192.168.122.2
ashokit-db1.com
ashokit-db2.com


================
Ansible Setup
================

=> Create 3 Red Hat Systems in AWS (Free Tier Eligible - t2.micro)

1 - Control Node
2 - Managed Nodes

=> Connect to all the 3 systems and create ansible user

$ sudo useradd ansible
$ sudo passwd ansible

pwd
confirm pwd

$ sudo visudo
ansible ALL=(ALL) NOPASSWD: ALL

$ sudo vi /etc/ssh/sshd_config

-> comment PasswordAuthentication no
-> un-comment PasswordAuthentication yes

-> Restart the server

$ sudo service sshd restart

Note: Do the above steps in all the 3 machines

=========================
Install Ansible in Control Node
=========================

-> Switch to Ansible user

$ sudo su ansible
```

```
-> Install Python

$ sudo yum install python3 -y

-> Check python version

$ python --version (it will fail bcz we used python3)
$ python3 --version

-> Update python alternatives

$ sudo alternatives --set python /usr/bin/python3

-> Check python version

$ python --version

-> Install PIP (It is a python package manager)

$ sudo yum -y install python3-pip

-> Install Ansible using Python PIP

$ pip3 install ansible --user

-> Verify ansible version

$ ansible --version

->  Create ansible folder under /etc

$ sudo mkdir /etc/ansible

-> create ansible.cfg file under /etc/ansible And paste complete content from below
git link.

Open : https://raw.githubusercontent.com/ansible/ansible/devel/examples/ansible.cfg

Copy the content and paste it in ansible.cfg file with below command

$ sudo vi /etc/ansible/ansible.cfg

-> Create hosts file under /etc/ansible. Sample content can found in below git link

Open : https://raw.githubusercontent.com/ansible/ansible/devel/examples/hosts
Copy the content and paste it in hosts file with below command

$ sudo vi /etc/ansible/hosts

******************** With this Ansible setup completed
**********************************

1) Update Host Inventory in Ansible Server to add host servers details to test
connection

$ sudo vi /etc/ansible/hosts

$ Connect using username and password
HOST-NODE-IP ansible_user=ansible ansible_password=password

2) Use ping module to test Ansible and after successful run you can see the below
output.

$ ansible all -m ping
```

Note : Install sshpass in Ansible server if you get below error .

"to use the 'ssh' connection type with passwords, you must install the sshpass program

```
$ sudo yum install sshpass

$ ansible all -m ping
```

===
Generate SSH Key In Control Node and Copy SSH key into Managed Nodes
===

1) Now generate SSH key in Ansible Server (Control Node):

```
$ sudo su ansible

# Generate ssh key using below command
$ ssh-keygen
```

2) Copy it to Managed Nodes as ansible user

```
$ ssh-copy-id ansible@<ManagedNode-Private-IP>

Ex : $ ssh-copy-id ansible@172.31.8.95
```

Note: Repeat above command by updating HOST IP for all the managed Servers.

3) Update Host Inventory in Ansible Server to add host servers details.

```
# vi /etc/ansible/hosts

# Connect using username and password  ( not recommended)
#192.168.1.105 ansible_user=ansible ansible_password=password

# If ssh keys are copied
172.31.43.23
```

4) Use ping module to test Ansible and after successful run you can see the below output.

```
# ping all managed nodes listed in host inventory file
$ ansible all -m ping

#ping only webservers listed in host inventory file
$ ansible webservers -m ping

#ping only dbservers listed in host inventory file
$ ansible dbservers -m ping
```

=========================
Ansible AD-HOC Commands
=========================

-> switch to ansible user and run ansible ad-hoc commands

```
$ sudo su ansible
```

=> To run any ansible command we will follow below syntax

```
# ansible [ all / groupName / HostName / IP ] -m <<Module Name>> -a <<args>>
```

Note: Here -m is the module name and -a is the arguments to module.

Example:

```
# It will display date from all host machines.
$ ansible all -m shell -a date

# It will display uptime from all host machines.
$ ansible all -m shell -a uptime

There are two default groups, all and ungrouped. all contains every host. ungrouped
contains all hosts that donâ€™t have another group

# It will display the all the modules available in Ansible.
$ ansible-doc -l

# To display particular module information
$ ansible-doc <moduleName>

# To display shell module information
$ ansible-doc shell

# it will display details of copy module
$ ansible-doc -l | grep "copy"

#It will display more information about yum module
$ ansible-doc yum


=============
Ping Module
==============

# It will ping all the servers which you have mentioned in inventory file
(/etc/ansible/hosts)
$ ansible all -m ping

# It will display the output in single line.
$ ansible all -m ping -o

# Date of all machines
$ ansible all -m shell -a 'date'

# Redhat release of all the machines
$ ansible all -m shell -a 'cat /etc/*release'

# Kind of mount on all the machines
$ ansible all -m shell -a 'mount'

# Check the service status on all the machines
$ ansible all -b -m shell -a 'service sshd status'

# Here it will check the disk space use for all the nodes which are from dbservers
group
$ ansible dbservers -a "df -h"

# Here it will check the disk space use for all the nodes which are from webservers
group
$ ansible webservers -a "free -m"

# Here it will display date from from webservers group
$ ansible webservers -a "date"

================
Yum Module
===============

# It will install vim package in all node machine which you have menyioned in host
```

```
inventory file.
$ ansible all -b -m yum -a "name=vim"

# Check git version in all machines
$ ansible all -m shell -a "git --version"

# to install git client in all node machines
$ ansible all -m shell -b -a "yum install git -y"

# To installl git only in webserver nodes
$ ansible webservers -m shell -b -a "yum install git -y"

# To install webserver only in particular machine
$ ansible 172.1921.1.0 -m shell -b -a "yum install git -y"

$ ansible all -m shell -b -a "name=git state=present"
$ ansible all -m shell -b -a "name=git state=latest"
$ ansible all -m shell -b -a "name=git state=absent"

present : install
latest : update to latest
absent : un-install

# to install any software in ubuntu server then we should use apt package manager

$ ansible all -m apt -a "name="git state="present"

# To install httpd package in all node machines
$ ansible all -b -m yum -a "name=httpd state=present"

Note: Here state=present, is not a mandatory, it is by default.

# To update httpd package in all node machines.
$ ansible all -b -m yum -a "name=httpd state=latest"

# To remove httpd package in all node machines.
$ ansible all -b -m yum -a "name=httpd state=absent"

$ ansible all -m copy -a "src="index.html dest=/var/www/html/index.html"

start httpd service

$ ansible all -b -m service -a "name=httpd state=started"

$ ansible all -b -m shell -a "service httpd start"


Note: For privilige escalations we can use -b option


Q) Irrespective of underlying OS which module we can use to manage
packages(softwares) using package manager in Ansible ?

Ans)  Ansible introduced "package manager" to work with underlying package manager


======================================
YAML (Yet Another Markup Language )
=====================================

-> YAML Ainâ€™t markup language

-> We can make use of this language to store data and configuration in a
human-readable format.
```

```
-> YAML files will have .yml as an extension

-> Official Website :  https://yaml.org/

===================
Sample YML File Data
===================
Fruit: Apple
Vegetable: Carrot
Liquid: Water
Meet: Chicken

Array/List
++++++++++++
Fruits:
 - Orange
 - Apple
 - Banana
 - Guava

Vegetables:
 - Carrot
 - Cauliflower
 - Tomoto

Here - dash indicate the element of any array.


name: Ashok
age: 29
phno: 123456
email: ashokitschool@gmail.com
hobbies:
  - cricket
  - dance
  - singing

# person data in yml

person:
  id: 101
  name: Raju
  email: raju@gmail.com
  address:
    city: Hyd
    state: TG
    country: India
  job:
    companyName: IBM
    role: Tech Lead
    pkg: 25 LPA
  hobbies:
    - cricket
    - chess
    - singing
    - dance



# using --- hypens to seperate the data

---
person:
  id: 101
  name: Raju
  email: raju@gmail.com
  address:
    city: Hyd
    state: TG
    country: India
  job:
    companyName: IBM
    role: Tech Lead
    pkg: 25 LPA
  hobbies:
    - cricket
    - chess
    - singing
    - dance
---
movie:
  name: Bahubali
  hero: Prabhas
  heroine: Anushka
  villian: Rana
  director: SS Rajamouli
  budget: 100cr
...
```

==============
Playbooks
============

-> Playbook is a single YAML file, containing one or more â€˜playsâ€™ in a list.

-> Plays are ordered sets of tasks to execute against host servers from your inventory file.

-> Play defines a set of activities (tasks) to run on managed nodes.

-> Task is an action to be performed on the managed node

Examples are

 a) Execute a command
 b) Run a shell script
 c) Install a package
 d) Shutdown / Restart the hosts

Note : Playbooks YML / YAML starts with the three hyphens (---) and ends with three dots (â€¦)

Playbook contains the following sections:

1) Every playbook starts with 3 hyphens â€˜---â€˜

2) Host section â€“ Defines the target machines on which the playbook should run. This is based on the Ansible host inventory file.

3) Variable section â€“ This is optional and can declare all the variables needed in the playbook. We will look at some examples as well.

4) Tasks section â€“ This section lists out all the tasks that should be executed on the target machine. It specifies the use of Modules. Every task has a name which is a small description of what the task will do and will be listed while the playbook is run.

```
=============================
Playbook To Ping All Host Nodes
=============================
---
- hosts: all
  gather_facts: no
  remote_user: anisble
  tasks:
   - name : Ping
     ping:
     remote_user: ansible
...

#hosts: The tasks will be executing in specified group of servers.

#name: which is the task name that will appear in your terminal when you run the
playbook.

#remote_user: This parameter was formerly called just user. It was renamed in
Ansible 1.4 to make it more distinguishable from the user module (used to create
users on remote systems).

Note : Remote users can also be defined per task.

# Run the playbook Using below command
$ ansible-playbook <<Playbbok file name>>

# It will run the playbook.yml playbook in verbose

$ ansible-playbook playbook.yml -v
$ ansible-playbook playbook.yml -vv
$ ansible-playbook playbook.yml -vvv

$ It will provide help on ansible_playbook command
$ ansible-playbook --help

#  It will check the syntax of a playbook
$ ansible-playbook playbook.yml --syntax-check

# It will do in dry run.
$ ansible-playbook playbook.yml --check

# It will display the which hosts would be effected by a playbook before run
$ ansible-playbook playbook.yml --list-hosts

# It execute one-step-at-a-time, confirm each task before running with
(N)o/(y)es/(c)ontinue
$ ansible-playbook playbook.yml --step


======================================
Install HTTPD + copy index.html + Start Service
======================================

-> Create index.html file in the location where our playbook is creating

-> Create yml file with below content

---
- hosts: all
  become: true
  tasks:
   - name: Install Httpd
     yum:
      name: httpd
      state: present
   - name: Copy index.html
     copy:
        src: index.html
        dest: /var/www/html/index.html
   - name: Start Httpd Server
     service:
       name: httpd
       state: started
...
```

-> Execute the playbook yml using ansible-playbook command

```
===========
Variables
===========
```

-> Variables are used to store the data

ex:

id = 101, name = ashok, age = 30

-> We can use 3 types of variables in Ansible

1) Local Variables
2) Group Variables
3) Host Variables

```
---
- hosts: all
  become: true
  tasks:
   - name: Install Httpd
     yum:
      name: "{{package_name}}"
      state: present
   - name: Copy index.html
     copy:
        src: index.html
        dest: /var/www/html/index.html
   - name: Start Http Server
     service:
       name: "{{package_name}}"
       state: started
...
```

=> We can pass variable value in run time like below

```
$ ansible-playbook <filename.yml> --extra-vars package_name=httpd
```

-> We can define variables with in the playbook also

```
---

- hosts: all
  become: true
  vars:
        package_name: httpd
  tasks:
   - name: Install Httpd
     yum:
      name: "{{package_name}}"
      state: present
   - name: Copy index.html
     template:
        src: index.html
        dest: /var/www/html/index.html
   - name: Start Http Server
     service:
       name: "{{package_name}}"
       state: started
...
```

```
---
- hosts: all
  become: true
  tasks:
   - name: install software
     yum:
       name: "{{package_name}}"
       state: present
...
```

```
$ ansible-playbook filename --extra-vars package_name=mysql
```

=================
Group Variables
=================

-> For webservers i want to install git software

-> For dbservers i want to install mysql software

-> We can achieve this using group variables

-> group vars files should be created at host inventory location

-> host-inventory location : /etc/ansible

group_vars/all.yml
group_vars/<groupName>.yml

Ex:

```
$ mkdir /etc/ansible/group_vars

$ sudo vi /etc/ansible/group_vars/webservers.yml
package: git

$ sudo vi /etc/ansible/group_vars/dbservers.yml
package: mysql
```

============
Host vars
=============

-> Server specific variables

-> In one group we will have multiple servers

-> For every host if we wan seperate variables then we should go for host vars

-> mkdir /etc/ansible/host_vars

-> create a file with host name or ip

-> vi /etc/ansible/host_vars/172.138.1.1.yml

=====================================
Variable Value we can declare with in playbook

Variable value we can supply in runtime

Variable value we can declare in hosts_vars

Variable value we can declare in group_vars
=====================================

===============
Ansible Vault
===============

-> It is used to secure our data

-> When we configure uname and pwd in variables files everybody can see them which is not a good practice

-> When we are dealing with sensitive data then we should secure that data

-> Using Ansible Vault we can protect or secure our data

-> Using Ansible vault we can encrypt and we can decrypt data

Encryption ---> Converting from readable format to un-readable format

De-Cryption ---> Converting un-readable format to readable format (bringing file back to normal state)

=======================
Ansible Vault Commands
=======================

```
$ ansible-vault encrypt <playbook-yml> : To encrypt our yml file

$ ansible-vault view <playbook-yml> : To see original data from encrypted file

$ ansible-vault edit <playbook-yml> : To edit data in original format

$ ansible-vault decrypt <playbook-yml> : To decrypt the file (bring the file to normal state)

$ ansible-vault rekey <playbook-yml> : To change vault password
```

-> To encrypt a playbook we need to set one vault password

-> while executing playbook we need to pass vault password

```
$ ansible-playbook <filename>.yml --ask-vault-pass
```

-> You can store vault password in a file and you can give that file as input to execute playbook

```
$ vi valutpass
$ ansible-playbook filename.yml --vault-password-file=~/vaultpass

# We can see encrypted file in human readable format
$ ansible-vault view /etc/ansible/group_vars/all.yml

# We can edit encrypted file in human readable format
$ ansible-vault edit /etc/ansible/group_vars/all.yml

# We can decrypt the file
$ ansible-vault decrypt /etc/ansible/group_vars/all.yml

# To update vault password we can use rekey
$ ansible-vault rekey /etc/ansible/group_vars/all.yml
```

=================
Handlers and Tags
================

-> Handlers are used to notify the tasks to execute

-> Using Handlers we can execute tasks based on other tasks status

Note: If second task status is changed then only i want to execute third task in playbook

-> To inform the handler to execute we will use 'notify' keyword

-> Using Tag we can map task to a tag-name

-> Using tag name we can execute particular task and we can skip particular task also

```
---
- hosts: all
  become: true
  gather_facts: yes
  vars:
    package_name: httpd
  tasks:
    - name: install httpd
      yum:
        name: "{{package_name}}"
        state: present
      tags:
        - install
    - name: Copy index.html
      copy:
        src: index.html
        dest: /var/www/html/
       tags:
          - copy
      notify:
        Start Httpd Server
  handlers:
    - name: Start Httpd Server
      service:
        name: "{{package_name}}"
        state: started
```

```
...


# to display all tags available in playbook
$ ansible-playbook filename.yml --list-tags

# Execute a task whose tag name is install
$ ansible-playbook filename.yml --tags "install"

# Execute the tasks whose tags names are install and copy
$ ansible-playbook filename.yml --tags "install,copy"

# Execute all the tasks in playbook by skipping install task
$ ansible-playbook filename.yml --skip-tags "install"
```

```
++++++++++++++++++++++++++++++
Installing Multiple Softwares
++++++++++++++++++++++++++++++
- hosts: all
  tasks:
  - name: install softwares
    yum:
          name: "{{item}}"
          state: present
        with_items:
        - wget
        - zip
        - unzip

++++++++++++++++++++++++++++++
Another approach
++++++++++++++++++++++++++++++
- hosts: all
  tasks:
  - name: install softwares
    yum:
          name: ['wget', 'zip', 'unzip']
          state: present


======================
```

Q) Write a playbook to install maven in Web servers. In Webservers group few machines are RED HAT based and few machines are Ubuntu based.

```
==> Amazon Linux / Cent OS / RED HAT OS   -----------> yum as package manager

==> Ubuntu / Debian  OS   -------> apt as package manager

---
- hosts: webservers
  tasks:
   - name: install maven
      yum:
        name: maven
        state: present
      when: ansible_os_family == 'RedHat'
  -  name : install maven
     apt:
        name: maven
        state: present
     when: ansible_os_family == 'Debian'
...
```

-----------------================
Ansible Roles
================

-> Roles are a level of abstraction for Ansible configuration in a modular and reusable format

-> As you add more and more functionality to your playbooks, they can become difficult to maintain

-> Roles allow you to break down a complex playbook into separate, smaller chunks that can be coordinated by a central entry point.

# Sample playbook with Role

```
---
- hosts: all
  become: true
  roles:
    - apache
```

1. Ansible roles are consists of many playbooks, which is similar to modules in puppet and cook books in chef.
    We term the same in ansible as roles.

2. Roles are a way to group multiple tasks together into one container to do the automation in very effective manner with clean directory structures.

3. Roles are set of tasks and additional files for a certain role which allow you to break up the configurations.

4. It can be easily reuse the codes by anyone if the role is suitable to someone.

5. It can be easily modify and will reduce the syntax errors.

++++++++++++++++++++++++++++++++
How do we create Ansible Roles?
++++++++++++++++++++++++++++++++

-> To create an Ansible role, use "ansible-galaxy" command which has the templates to create it.

```
$ sudo su ansible

$ cd /home/ansible

$ mkdir roles

$ cd roles

$ ansible-galaxy init apache
```

> where, ansible-glaxy is the command to create the roles using the templates.

> init is to initiliaze the role.

> apache is the name of the role

List out the directory created under apache

```
$ sudo yum install tree
$ tree apache
```

We have got the clean directory structure with the ansible-galaxy command. Each directory must contain a main.yml file, which contains the relevant content.

+++++++++++++++++++++
Directory Structure:
+++++++++++++++++++++

tasks â€“ contains the main list of tasks to be executed by the role.

handlers â€“ contains handlers, which may be used by this role or even anywhere outside this role.

defaults â€“ default variables for the role.

vars â€“ other variables for the role. Vars has the higher priority than defaults.

files â€“ contains files required to transfer or deployed to the target machines via this role.

templates â€“ contains templates which can be deployed via this role.

meta â€“ defines some data / information about this role (author, dependency, versions, examples, etc,.)

+++
Lets take an example to create a role for Apache Web server.
+++

Below is a sample playbook to deploy Apache web server. Lets convert this playbook code into Ansible role.

```
- hosts: all
  become: true
  tasks:
   - name: Install Httpd
     yum:
      name: httpd
      state: present
   - name: Copy index.html
     template:
       src: index.html
       dest: /var/www/html/index.html
   - name: Start Http Server
     service:
       name: httpd
       state: started
```

First, move on to the Ansible roles directory and start editing the yml files.

```
$ cd roles/apache
```

1. Tasks
+++++++++++++++

Edit main.yml available in the tasks folder to define the tasks to be executed.

```
$ vi tasks/main.yml

---
# tasks file for roles/apache
- name: install httpd
  yum:
   name: httpd
   state: present
- name: Copy index.html
  copy:
    src=index.html
    dest=/var/www/html/
  notify:
  - restart apache
```

2. Files
+++++++++++++
-> Copy required files into files directory or create index.html file with content

```
$ vi files/index.html

<write content here>
```

3. Handlers
++++++++++++++
Edit handlers main.yml to restart the server when there is a change. Because we have already defined it in the tasks with notify option. Use the same name â€œrestart apacheâ€ within the main.yml file as below.

```
$ vi handlers/main.yml

- name: restart apache
  service:
   name: httpd
   state: restarted
```

We have got all the required files for Apache roles. Lets apply this role into the ansible playbook â€œrunsetup.ymlâ€ as below to deploy it on the client nodes.

```
$ vi /home/ansible/runsetup.yml

---
- hosts: all
  become: true
  roles:
   - apache
...
```

-> Execute playbook which contains apache role

```
        $ ansible-playbook runsetup.yml
```

If you have created multiple roles, you can use the below format to add them in the playbook

```
---
- hosts: all
  become: true
  roles:
   - apache
   - jenkins
   - java
   - maven
   - sonar

-----------------------------------------------------------------------------------
--------------------------------------------------------==================
Docker & Kubernetes
===================

Application Tech Stack :  It represents technologies used in the application


1) Frontend Stack : HTML, CSS, JS, BS & Angular / React JS

2) Backend Stack : Java / .Net / Python / Node JS

3) Database : Oracle / MySQL / PostgresSQL / Mongo DB


========
Docker
========

-> Docker is a containizeration platform

-> Docker is used to simplify application deployment process in Multiple
Environments
                (DEV, SIT, UAT, PILOT and PROD)

-> Docker is used to package application code + application dependencies for easy
execution

-> Using Docker we will create Docker images

-> Docker Image contains App code + App dependencies

-> We can run docker image in any machine. It will take care of dependencies &
execution

-> When we run Docker image, it will create Docker container

-> Docker Container will run our application




===============
Virtualization
===============

-> Installing Multiple Guest Operating Systems in one Host Operating System

-> Hypervisior S/w will be used to achieve this

-> We need to install all the required softwares in HOST OS to run our application

-> It is old technique to run the applications

-> System performance will become slow in this process

-> To overcome the problems of Virtualization we are going for Containerization
concept

=================
```

Containerization
==================

-> It is used to package all the softwares and application code in one container for execution

-> Container will take care of everything which is required to run our application

-> We can run the containers in Multiple Machines easily

-> Docker is a containerization software

-> Using Docker we will create container for our application

-> Using Docker we will create image for our application

-> Docker images we can share easily to mulitple machines

-> Using Docker image we can create docker container and we can execute it

============
Conclusion
============

-> Docker is a containerization software

-> Docker will take care of application and application dependencies for execution

-> Deployments into multiple environments will become easy if we use Docker containers concept

Container = Application Code + Application Libraries + Application Dependencies

Docker is a containerization software

Using Docker we will create Docker image

Docker Image =
 Application code + Application libs (maven dependencies) + Application Dependencies(java, tomact, mysql etc...)

Once docker image is created then we can use jenkins to run docker image in multiple machines

Jenkins is just deployment software. We will use jenkins to run docker images in all environments

When we run Docker image it will create Docker container

Docker Container means Runtime instance of our application

===================
Docker Architecture ?
===================

1) Dockerfile

2) Docker Image

3) Docker Registry

4) Docker Container

-> Dockerfile contains set of instructions to build docker image. In dockerfile we will specify which sotwares are required to run our code/application.

-> Docker image means a package which contains application code + app libs + app dependencies

-> Docker Registry is a place which is used to store docker images for future purpose

                Ex: Docker Hub, Amazon ECR etc....

-> Docker container is runtime instance of our application. When we run Docker Image it will create Docker Container. Inside Container our application and application dependencies will be available.

1) What is Docker
2) What is Virtualization
3) What is Containerization
4) What is Docker Architecture
5) Dockerfile
6) Docker Image
7) DockerHub
8) Docker Container

=======================
Install Docker in Linux VM
=======================

-> Loging into AWS account

-> Create Linux Virtual Machine using Amazon Linux AMI

-> Connect to Linux VM using MobaXterm

-> Execute below commands to install Docker s/w

```
$ sudo yum update -y
$ sudo yum install docker -y
$ sudo service docker start

# add ec2-user to docker group by executing below command (to give docker permissions to ec2-user accnt)
$ sudo usermod -aG docker ec2-user

# Close the terminal
$ exit
```

Then press 'R' to restart the session (This is in MobaXterm)

```
#execution below command to see docker status

$ docker info
```

```
====================
Basic Docker Commands
====================

# display docker images available in our machine

$ docker images

# download docker image

$ docker pull <image-name / image-id>

# Run docker image

$ docker run <image-name / image-id>

# Delete docker image

$ docker rmi <image-name / image-id>

# Display all running docker containers

$ docker ps

# display all running and stopped containers

$ docker ps -a

# Delete docker container

$  docker rm <container-id>

# Delete docker image forcefully

$  docker rmi  -f <image-id>

# Stop Docker container

$ docker stop <container-id>

# Delete all stopped containers and un-used images and un-used networks

$ docker system prune -a

==========
Dockerfile
=========

-> Dockerfile contains instructions to build docker image

-> In Dockerfile we will use DSL (Domain Specific Language) keywords

-> Docker engine will process Dockerfile instructions from top to bottom

-> Below are the Dockerfile Keywords

FROM
MAINTAINER
COPY
ADD
RUN
CMD
ENTRYPOINT
ENV
```

LABEL
WORKDIR
EXPOSE
VOLUME

=============
FROM
=============

-> FROM keyword is used represent base image to create our our image
-> On Top of base image our image will be created

Syntax:

```
FROM java:jdk-1.8
FROM tomcat:9.5
FROM mysql:6.8
FROM python:3.3
```

=============
MAINTAINER
=============

-> MAINTAINER keyword is used to specify Dockerfile author information

Syntax:

```
MAINTAINER  Ashok <ashok.b@oracle.com>
```

=======
COPY
=======

-> COPY command is used to copy the files from source to destination while creating docker image

Syntax:

```
COPY <source-location>  <destination-location>
```

Ex:

```
COPY  target/sbi-app.war   /app/tomcat/webapps/sbi-app.war
```

=======
ADD
=======

-> ADD command is also used to copy files from source to destination while creating docker image

Syntax:

```
ADD <source-location>  <destination-location>

ADD <url>  <destination-location>
```

Ex:

```
ADD  <URL>   /app/tomcat/webapps/sbi-app.war
```

Q) What is the difference between COPY and ADD commands ?

-> Using COPY command we can just copy the files from one path to another path with in the machine

-> Using ADD command we can copy files from one path to another path and it supports source location as URL also.

=======
RUN
=======

-> RUN instructions will execute while creating the image

-> Using RUN we can give instructions to docker to execute commands

-> We can write multiple RUN instructions, docker will process all the RUN instructions from top to bottom

Example

```
RUN yum install maven
RUN yum install git
RUN git clone repo-url
RUN mvn clean package
```

=======
CMD
=======

-> CMD instructions will execute while creating the container

-> Using CMD command we can run our application package file jar / war file

Example

```
CMD  sudo start tomcat
```

Note: If we write multiple CMD instructions also docker will process only last CMD instruction. There is no use of writing multiple CMD instructions in one Dockerfile.

Q) What is the difference between RUN and CMD in Dockerfile ?

-> RUN is used to execute instructions while creating image
-> CMD is used to execute instruction while creating Container

-> We can write multiple RUN instructions in Dockerfile, docker will process all those instructions one by one.
-> If we write multiple CMD instructions in Dockerfile, docker willl process only last CMD instruction.

==================
Sample Dockerfile
=================

```
FROM ubuntu

MAINTAINER Ashok<ashokit@gmail.com>

RUN echo "Hi, i am RUN-1"

RUN echo "Hi, i am RUN-2"

CMD echo "Hi, I am CMD-1"

RUN echo "Hi, i am RUN-3"

CMD echo "Hi, i am CMD-2"
```

-> Save the above content in docker file

                filename : Dockerfile

# Command to create docker image using dockerfile

Syntax : $ docker build -t <image-name> .

Ex : $ docker build -t myfirstimage .

# Command to run docker image

```
$ docker run myfirstimage
```

# Command to login with dockerhub account

```
$ docker login
```

Note: We need to enter our dockerhub account credentials correctly (it will ask only first time)

# Command to tag our docker image

```
$ docker tag  <image-name>  <tag-name>
```

Ex: $ docker tag myfirstimage ashokit/myfirstimage

# command to push docker image to docker hub account

```
$ docker push <tag-name>
```

Note: Delete all unused images and stopped containers

```
$ docker system prune -a
```

# Pull the image from docker hub

```
$ docker pull ashokit/myfirstimage
```

# Run the image

```
$ docker run ashokit/myfirstimage
```

Note: We can use customized name also for the dockerfile. When we change dockerfile name we need to pass filename as input for docker build command using -f option like below.

```
$ docker build -f <dockerfile-name>  -t <image-name> .
```

============
ENTRYPOINT
============

-> ENTRYPOINT instructions will execute while creating container

Syntax

```
ENTRYPOINT [ "echo"  , "Welcome to Ashok IT" ]

ENTRYPOINT [  "java" , "-jar" , "target/boot-app.jar"  ]
```

Q) What is the difference between CMD and ENTRYPOINT ?

-> We can override CMD instructions in runtime while creating container

-> We can't override ENTRYPOINT instructions

==========
WORKDIR
==========

-> It is used to set working directory for an image / container

Ex:

```
WORKDIR     /app/
```

Note: The Dockerfile instructions which are available after WORKDIR those instructions will be processed from given working directory.

======
ENV
======

-> ENV is used to set Environment Variables

Ex:

```
ENV <key> <value>

ENV   java   /etc/softwares/java
```

====
ARG
====

-> It is used to remove hard coded values

-> Using ARG we can pass values in the runtime like below

Ex:

ARG branch

RUN git clone -b $branch <repo-url>

$ docker build -t imageone --build-arg branch=master

=====
USER
=====

-> We can set user for creating image / container

Note: After USER instruction all the remaining commands will execute with given user permissions

========
EXPOSE
========

-> It is used to specify our container running PORT

Ex:

EXPOSE 8080

Note: It is just like a documentation command to provide container running port number.

========
VOLUME
========

-> VOLUME is used to specify docker container data storage location.

Note: Volumes are used for storage.

FROM
MAINTAINER
COPY
ADD
RUN
CMD
ENTRYPOINT
WORKDIR
USER
ENV
ARG
EXPOSE
VOLUME

==============================
Dockerize Spring Boot Application
==============================

-> Spring Boot is one ready made java based framework available in the market to develop java based applications quickly

-> Spring Boot is providing emedded server (internal server will be available, we no need to configure server for execution)

-> Spring Boot application will be packaged as jar file (mvn clean package goal will do that package)

Note: When we do maven package, project jar will be created in project target folder

-> To run spring boot applications we just need to run jar file like below

```
$  java -jar <boot-app-jar-file>
```

--------------------------------Dockerfile------------------------------------

```
FROM openjdk:11

COPY target/spring-boot-docker-app.jar  /usr/app/

WORKDIR /usr/app/

ENTRYPOINT ["java", "-jar", "spring-boot-docker-app.jar"]
```

Spring Boot App Git Repo URL :
https://github.com/ashokitschool/spring-boot-docker-app.git

```
# install git client s/w
$ sudo yum install git

# Clone Git Repo
$ git clone https://github.com/ashokitschool/spring-boot-docker-app.git

# Navigating to project folder
$ cd spring-boot-docker-app

# install maven s/w
$ sudo yum install maven

# execute maven goals
$ mvn clean package
```

Note: After package got success, we can see project jar file in target folder.

```
# create docker image
$ docker build -t sb-app .

# run docker image with port mapping
$ docker run -p 8080:8080 sb-app
```

Note: Enable 8080 port number in EC2 VM security group

URL To Access Application :   http://ec2-vm-public-ip:8080/welcome/Ashok

======================================
How to run Docker container in Detached Mode
======================================

=> With below command our terminal will be blocked, we can't execute any other command. To execute other command we need to type CTRL+C then terminal will open for commands execution but our container gets stopped.

```
$ docker run -p 8080:8080 ashokit/sb-app
```

Note: To overcome above problem we can pass '-d' to run container in detached mode. When we execute below command it will run the container in detached mode and it will open terminal for commands execution.

```
$ docker run -d -p 8080:8080 ashokit/sb-app
```

=> Once above command is executed, we can see running containers using below command

```
$ docker ps
```

=> We can check logs of the container using below command

```
$ docker logs <container-id>
```

==
Dockerizing Java Web Application (Without SpringBoot)
==

-> Java web applictaions will be packaged as war file

-> WAR (Web Archive) contains application code

-> To run the war file we need webserver (Ex: Apache Tomcat)

-> We need to deploy war file in Tomcat Server for Execution

-> In Tomcat server we will have "webapps" folder for deployment

Note: To run normal java web applications we need "java & tomcat" as dependencies

```
---------------------------------------------------Dockerfile---------------------------------------------------

FROM tomcat:8.0.20-jre8

COPY target/01-maven-web-app.war   /usr/local/tomcat/webapps/maven-web-app.war

------------------------------------------------------------------------------------------------------------------
```

############### Java Web App Git Repo : https://github.com/ashokitschool/maven-web-app.git ###########

```
$ git clone https://github.com/ashokitschool/maven-web-app.git

$ cd maven-web-app

$ mvn clean package

$ docker build -t maven-web-app .

$ docker images

$ docker run -d -p 8080:8080 maven-web-app

$ docker ps

$ docker logs <container-id>
```

URL To access The Application :   http://ec2-vm-public-ip:host-port/maven-web-app/

Note: In the above url "maven-web-app" is called as context path (name of the war file will become context path)

========================
Dockerize Python Application
========================

-> Python is a general purpose scripting language

-> Python programs will have .py as extension

-> Compilation is not required for python programs

```
------------------Dockerfile---------------------

FROM python:3.6

MAINTAINER Ashok Bollepalli "ashokitschool@gmail.com"

COPY . /app

WORKDIR /app

EXPOSE 5000

RUN pip install -r requirements.txt

ENTRYPOINT ["python", "app.py"]

-----------------------------------------------------------------------------
```

Python Flask app Git hub Repo:
https://github.com/ashokitschool/python-flask-docker-app.git

```
$ git clone https://github.com/ashokitschool/python-flask-docker-app.git

$ cd python-flask-docker-app

$ docker build -t python-flask-app .

$ docker images

$ docker run -d -p 5000:5000 python-flask-app

$ docker ps

$ docker logs <container-id>
```

```
# Command to enter into docker container
$ docker exec -it <container-id> /bin/bash
```

Note: Execute 'exit' command to come out from docker container vm.

======================================

Assignment -1 : Dockerize React JS Application

Assignment-2 : Dockerize Node JS Application

======================================

================
Docker Network
================

-> Network is all about communication

-> Docker network is used to provide isolated network for Docker Containers

-> In Docker we will have below 3 default networks

1) none

2) host

3) bridge

-> In Docker we have below 5 network drivers

1) Bridge ----> This is default network driver in Docker
2) Host
3) None
4) Overlay ----> Docker Swarm
5) Macvlan

-> Bridge driver is recommended driver when we are running standalone container. It will assign one IP for container

-> Host Driver is also used for standalone container. IP will not be assigned for container

-> None means no network will be provided by our Docker containers

-> Overlay network driver is used for Orchestration. Docker Swarm will use this Overlay network driver

-> Macvlan driver will assign MAC address for a container. It makes our container as Physical.

```
# display docker networks available
$ docker network ls

# Create docker network
$ docker network create ashokit-network

# Inspect network
$ docker network inspect <network-name>

# delete docker network
$ docker network rm <network-id>

# Run a container with given network
$ docker run -d -p hport:cport --network ashokit-network <imagename>
```

===============
Docker Compose

===============

Monolith Application : One application which contains all the functionalities is called as Monlith App.

-> If we make any small code change then we need to re-deploy entire application
-> If we make any code change in one functionality there may be a impact on another functionality
-> Maintenence will become very difficult when we go for Monolith Based Application

Note: To overcome the problems of Monlith we are using Microservices Architecture.

Microservices Application : Application functionality will be developed as micro services (rest apis)

-> Every functionality will be developed as individual project (individual API)

ADMIN_API
REPORT_API
PAYMENT_API
CART_API
TRACKING_API
PRODUCT_API
CANCEL_API

=> Every API should run in a seperate container

=> Running Multiple containers manully for all the apis is difficult job

**************** To solve this problem Docker-Compose came into picture ********************************

=> Docker Compose is a tool which is used to manage multi container based applications

=> Using Docker Compose we can easily setup & deploy multi container based applications

=> We will give containers information to Docker Compose using YML file (docker-compose.yml)

=> Docker Compose YML should have all the information related to containers creation

======================
Docker Compose YML File
======================

```
version:

services:

network:

volumes:
```

=====================

=> Docker Compose default file name is "docker-compose.yml"

```
# Create Containers using Docker Compose
$ docker-compose up

# Create Containers using Docker Compose with custom file name
$ docker-compose -f <filename> up

# Display Containers created by Docker Compose
$ docker-compose ps

# Display docker compose images
$ docker-compose images

# Stop & remove the containers created by docker compose
$ docker-compose down



=====================
Docker Compose Setup
=====================

# download docker compose
$ sudo curl -L
"https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname
-s)-$(uname -m)" -o /usr/local/bin/docker-compose

# Give permission
$ sudo chmod +x /usr/local/bin/docker-compose

# How to check docker compose is installed or not
$ docker-compose --version



=======================================
Spring Boot with MySQL using Docker Compose
=======================================

=> Spring Boot App with MySQL DB Git repo URL

                URL :
https://github.com/ashokitschool/spring-boot-mysql-docker-compose.git

=> Below is the docker-compose file


---
version: "3"
services:
  application:
    image: spring-boot-mysql-app
    networks:
      - springboot-db-net
    ports:
      - "8080:8080"
    depends_on:
      - mysqldb
  mysqldb:
    image: mysql:5.7
    networks:
      - springboot-db-net
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=sbms
```

```
networks:
  - springboot-db-net:
...
```

```
===============================================================
Steps to Run Spring Boot application with MySQL DB using Docker Compose
================================================================

# Clone git repo url
$ git clone https://github.com/ashokitschool/spring-boot-mysql-docker-compose.git

# Get into project directory
$ cd spring-boot-mysql-docker-compose

# Build Maven Project
$ mvn clean package

# Create Docker Image  ( Image Name : spring-boot-mysql-app )
$ docker build -t spring-boot-mysql-app .

# Check docker image created or not
$ docker images

# Run containers using docker compose ( docker-compose.yml available )
$ docker-compose up -d

# Check containers which are created
$ docker-compose ps

# Check logs of application contianer
$ docker logs -f <container-name>

Note: Access the application in browser

        URL : http://ec2-vm-public-ip:host-port/


# Get into App container to check jar file
$ docker exec -it <container-name> /bin/bash/


# Check DB tables by entering into container
$ docker exec -it <db-container-name> /bin/bash

$ mysql -u root -p

$ show databases;

$ use <db-name>   (our db name is sbms)

$ show tables;

$ select * from table-name  (our table name is book)
```

```
====================================
Stateful Conatiners Vs Stateless Containers
====================================
```

-> Stateless Container means container will not remember the data which got generated by that container. When we re-create the new container we will loose old data.

Note: By default docker containers are stateless containers.

-> In above springboot application when we recreate the containers we lost our old data database which we inserted through application. (This is not accepted in the realtime).

Note: Even if we deploy latest code or if we re-create the containers we should not loose our old data. Our data should remain in the database.

-> If we don't want to loose the data even if we re-create the container then we need to make our Docker Container as Statefull Container.

-> To make Docker Containers as statefull, we need to use Docker Volumes.

========================
Docker Volumes
========================

-> Applications we are executing as Docker Containers

-> Docker containers are by default stateless

-> Once container is removed then we will loose the data that got stored in the container

-> In realtime we shouldn't loose the data even if container got removed

For Example : Database container

-> Application will store data in database, even if we delete application container or db container data should be available.

-> To make sure data is available even after the container is deleted then we need to use Docker Volumes concept

********************* Docker Volumes are used to store container data permanently *************************************

=> Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.

=> We have 3 types of volumes in Docker

1) Anonymous Volumes (without name)

2) Named Volumes (Will have a name) ----> Recommended

3) Bind Mounts (Storing on Host Machine)

Q) What is Dangling volume in Docker ?

-> The volumes which are created but not associated to any container are called as Dangling Volumes

```
# Delete all dangling volumes
$ docker volume rm $(docker volume ls -q -f dangling=true);

# Create Docker volume
$ docker volume create <vol-name>
```

```
# Display all docker volumes
$ docker volume ls

# Inspect Docker Volume
$ docker volume inspect <vol-name>

# Delete docker volume
$ docker volume rm <vol-name>

# Delete all docker volumes
$ docker system prune --volumes
```

```
---------------------------------------- Docker Compose with Docker Named Volumne
----------------------------------
version: "3"
services:
  application:
    image: spring-boot-mysql-app
    ports:
      - "8080:8080"
    networks:
      - springboot-db-net
    depends_on:
      - mysqldb
  mysqldb:
    image: mysql:5.7
    networks:
      - springboot-db-net
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=sbms
    volumes:
      - /var/lib/mysql
networks:
  springboot-db-net:
-----------------------------------------------------------------------------------
-----
```

=============
Docker Swarm
=============

Docker : It is a containerization platform. It is used to deploy the applications as containers.

Docker Swarm : It is an Orchestration Platform. It is used to manage docker containers.

-> Managing docker containers nothing but creating / updating / scale up / scale down / remove containers

Note: In market we have docker swarm, kubernetes, open shfit as Orachestration platforms.

-> Docker Swarm is used to setup Docker Cluster

-> Cluster means group of servers

-> Docker swarm is embedded in Docker engine (No need to install Docker Swarm Seperatley)

-> We will setup Master and Worker nodes using Docker Swarm cluster

-> Master Node will schedule the tasks (containers) and manage the nodes and node failures

-> Worker nodes will perform the action (containers will run here) based on master node instructions

==================
Swarm Features
==================
1) Cluster Management
2) Decentralize design
3) Declarative service model
4) Scaling
5) Multi Host Network
6) Service Discovery
7) Load Balancing
8) Secure by default
9) Rolling Updates

============================
Docker Swarm Cluster Setup
============================

-> Create 3 EC2 instances (ubuntu) & install docker in all 3 instances using below 2 commands

```
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh
```

Note: Enable 2377 port in security group for Swarm Cluster Communications

1 - Master Node
2 - Worker Nodes

-> Connect to Master Machine and execute below command

```
# Initialize docker swarm cluster
$ sudo docker swarm init --advertise-addr <private-ip-of-master-node>
```

```
Ex : $ sudo docker swarm init --advertise-addr 172.31.37.100
```

```
# Get Join token from master  (this token is used by workers to join with master)
$ sudo docker swarm join-token worker
```

Note: Copy the token and execute in all worker nodes with sudo permission

```
Ex: sudo docker swarm join --token
SWMTKN-1-4pkn4fiwm09haue0v633s6snitq693p1h7d1774c8y0hfl9yz9-8l7vptikm0x29shtkhn0ki8
wz 172.31.37.100:2377
```

Q) what is docker swarm manager quarm?

Ans) If we run only 2 masters then we can't get High Availability

Formula : (n-1)/2

If we take 2 servers

2-1/2 => 0.5 (It can't become master)

3-1/2 => 1 (it can be leader when the main leader is down)

Note: Always use odd number for Master machines

-> In Docker swarm we need to deploy our application as a service.

====================
Docker Swarm Service
====================

-> Service is collection of one or more containers of same image

-> There are 2 types of services in docker swarm

1) Replica (default mode)
2) global

```
$ sudo docker service create --name <serviceName> -p <hostPort>:<containerPort>
<imageName>
```

```
Ex :  $ sudo docker service create --name java-web-app -p 8080:8080
ashokit/javawebapp
```

Note: By default 1 replica will be created

Note: We can access our application using below URL pattern

        URL : http://master-node-public-ip:8080/java-web-app/

```
# check the services created
$ sudo docker service ls

# we can scale docker service
$ docker service scale <serviceName>=<no.of.replicas>

# inspect docker service
$ sudo docker service inspect --pretty <service-name>

# see service details
$ sudo docker service ps <service-name>

# Remove one node from swarm cluster
$ sudo docker swarm leave

# remove docker service
$ sudo docker service rm <service-name>
```

=========
Summary
=========

1) What is Application Stack
2) Life without Docker
3) Life with Docker
4) Docker introduction
5) Virtualization vs Containerization

6) Docker Installation in Linux
7) Docker Architecture
8) Docker Terminology
9) Dockerfile & Dockerfile Keywords
10) Writing Dockerfiles
11) Docker image commands
12) Docker container commands
13) Dockerizing Java Spring Boot Application
14) Dockerizing Java Web Application with External Tomcat
15) Dockerizing Python Flask Application
16) Docker Network
17) Monlolith Vs Microservices
18) Docker Compose
19) Docker Compose File Creation
20) Docker Volumes
21) Spring Boot with MySQL DB Dockerization using Docker Compose
22) What is Orchestration ?
23) Docker Swarm
24) Docker Swarm Cluster Setup
25) Deployed java web app as docker container using swarm cluster

```
$ docker info
$ docker images
$ docker rmi <imagename>
$ docker pull <imagename>
$ docker run <imagename>
$ docker run -d -p host-port : container-port <image-name>
$ docker tag <image-name> <image-tag-name>
$ docker login
$ docker push <image-tag-name>

$ docker ps
$ docker ps -a
$ docker stop <container-id>
$ docker rm <container-id>
$ docker rm -f <container-id>
$ docker system prune -a
$ docker logs <container-id>
$ docker exec -it <container-id> /bin/bash

$ docker network ls
$ docker network create <network-name>
$ docker network rm <network-name>
$ docker network inspect <network-name>

$ docker-compose up -d
$ docker-compose down
$ docker-compose ps
$ docker-compose images
$ docker-compose stop
$ docker-compose start

$ docker volume ls
$ docker volume create <vol-name>
$ docker volume inspect <vol-name>
$ docker volume rm <vol-name>
$ docker system prune --volumes


$ sudo docker service --name <service-name> -p 8080:8080 <img-name>
$ sudo docker service scale <service-name = replicas
$ sudo docker service ls
$ sudo docker service rm <service-name>
```

```
# To check os version
$ cat /etc/os-release

# To check kernal version
$ uname -r
-----------------------------------------------------------------------------------
-----------------------------------------------====================================
============
Kubernetes HA Set up  (Multi Node Cluster with kubeadm)
================================================

-> Create one security group with below inbound rules

                Rule-1 : "Type : SSH (22), Source : Anywhere"

                Rule-2 : "Type : ALL TCP , Source: Anywhere"

-> Create 3 virtual machines using Ubuntu 20.04 version AMI  by selecting above
created security group

                1 Master Node : t2.medium

                2 Worker Nodes :  t2.medium

*******************************  Note: Use UBUNTU 20.04 version AMI for all 3
machines *************************************


================== Part-1 : Master & Worker Nodes Common Commands Execution start
====================


# Upgrade apt packages
$ sudo apt-get update

#Create configuration file for containerd:
$ cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf overlay br_netfilter
EOF

#Load modules:

$ sudo modprobe overlay
$ sudo modprobe br_netfilter


#Set system configurations for Kubernetes networking:

$ cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF


#Apply new settings:
$ sudo sysctl --system

#Install containerd:

$ sudo apt-get update && sudo apt-get install -y containerd


# Create default configuration file for containerd:
```

```
$ sudo mkdir -p /etc/containerd


#Generate default containerd configuration and save to the newly created default
file:

$ sudo containerd config default | sudo tee /etc/containerd/config.toml


#Restart containerd to ensure new configuration file usage:

$ sudo systemctl restart containerd


#Verify that containerd is running (optional)

$ sudo systemctl status containerd (presss ctrl+c for exit)


#Disable swap:

$ sudo swapoff -a


#Disable swap on startup in /etc/fstab:

$ sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab


#Install dependency packages:

$ sudo apt-get update && sudo apt-get install -y apt-transport-https curl


# Download and add GPG key:

$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add
-


# Add Kubernetes to repository list:

$ cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF


Update package listings:

$ sudo apt-get update


# Install Kubernetes packages (Note: If you get a dpkg lock message, just wait a
minute or two before trying the command again):

$ sudo apt-get install -y  kubelet kubeadm kubectl kubernetes-cni nfs-common


# Turn off automatic updates:

$ sudo apt-mark hold kubelet kubeadm kubectl kubernetes-cni nfs-common


=====================Common Commands Execution
End============================================
=================Part-2 : Only Master Node Commands Execution
Start===================================


# Initialize the Cluster
$ sudo kubeadm init

# Set kubectl access:
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config


# Test access to cluster:
$ kubectl get nodes


# Install the Calico Network Add-On -
# On the Control Plane Node, install Calico Networking:

$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
$ kubectl get nodes


# Join the Worker Nodes to the Cluster
$ kubeadm token create --print-join-command

Note : In both Worker Nodes, paste the kubeadm join command to join the cluster.
Use sudo to run it as root:

$ do kubeadm join ...

In the Control Plane Node, view cluster status (Note: You may have to wait a few
moments to allow all nodes to become ready):

#command to join other nodes as master


=====================================================
Validate the setup by executing below command in master-node
=====================================================
$  kubectl get nodes


-----------------------------------------------------------------------------------
--------------------------------------------
```

=============
K8S HELM
============

-> We deployed our apps in Kubernetes cluster using Manifest files

-> Manifest files we can write in 2 ways

1) JSON
2) YML (more demand)

-> It is difficult to write manifest files for our applications

-> Helm is a package manager for k8s applications

-> Helm allows you to install or deploy applications on kubernetes cluster in a similar manner to yum/apt for linux distributions.

-> Helm lets you fetch, deploy and manage the lifecycle of applications both 3rd party apps and your own applications

Ex: prometheus, grafana, nginx-ingress, ELK stack are third party apps

-> Helm introduces several familiar concepts such as

Helm Chart (package contains k8s manifests - templates)

Helm Repositories which holds helm charts/packages

A CLI with install/upgrade/remove commands

================
Why to use Helm?
================

-> Deploying application on K8S cluster is little difficult

-> As part of app deployment we need to create below k8s objects

Deployment
Service
ConfigMaps/Secrets
Volumes
Ingress Rules
HPA

-> Helm greatly simplifies the process of creating, deploying and managing applications on k8s cluster

-> Heml also maintains a versioned history of very chart (application) installation. If something goes wrong , you can simply call 'helm rollback'.

-> Setting up a single application can involve creating multiple independent k8s resources and each resource requires a manifest file.

####################
What is Helm Chart
####################

-> HELM chart is a basically just a collection of manifest files organized in a specific directory structure that describe a related K8S resource.

-> There are two main components in HELM chart

1) template
2) value

-> Templates and values renders a manifest which can understand by k8s

-> Helm uses charts to pack all the required k8s components (manifests) for an application to deploy, run and scale.

-> charts are very similar to RPM and DEB packages for Linux.

Ex: yum install git

Note: it will interact with repo and it will download git

```
##############
HELM Concepts
##############

-> Helm packages are called charts, and they consist of a few YML configuration
files and some templates that are rendered into K8S manifest files. Here is the
basic directory structure of a chart.

charts : dependent charts will be added here

templates: contains all template files

values : It contains values which are required for templates

##################
HELM Architecture
##################

what-the-helm
â”œâ”€â”€ Chart.yaml
â”œâ”€â”€ charts
â”œâ”€â”€ templates
â”‚   â”œâ”€â”€ NOTES.txt
â”‚   â”œâ”€â”€ _helpers.tpl
â”‚   â”œâ”€â”€ deployment.yaml
â”‚   â”œâ”€â”€ ingress.yaml
â”‚   â”œâ”€â”€ service.yaml
â”‚   â””â”€â”€ tests
â”‚       â””â”€â”€ test-connection.yaml
â””â”€â”€ values.yaml


##################
Helm Installation
##################

$ curl -fsSl -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3

$ chmod 700 get_helm.sh

$ ./get_helm.sh

$ helm

-> check do we have metrics server on the cluster

$ kubectl top pods

$ kubectl top nodes

# check helm repos
$ helm repo ls

# Before you can install the chart you will need to add the metrics-server repo to
helm
$ helm repo add metrics-server https://kubernetes-sigs.github.io/metrics-server/

# Install the chart
$ helm upgrade --install metrics-server metrics-server/metrics-server

$ helm list
$ helm delete <release-name>
```

```
================================
Metric Server Unavailability issue fix
================================
URL : https://www.linuxsysadmins.com/service-unavailable-kubernetes-metrics/
```

```
$ kubectl edit deployments.apps -n kube-system metrics-server
```

=> Edit the below file and add new properties which are given below

```
--------------------------- Existing File--------------------
 spec:
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443

--------------------------- New File--------------------
---
    spec:
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-insecure-tls=true
        - --kubelet-preferred-address-types=InternalIP

-------------------------------------------------------------------
```

```
$ kubectl top pods

$ kubectl top nodes
```

```
-------------------------------------------------------------------------------------
-------------------------------------------------------

#############
Kubernetes Monitoring
#######################
```

=> We can monitor our k8s cluster and cluster components using below softwares

1) Prometheus
2) Grafana

```
=============
Prometheus
=============
```

-> Prometheus is an open-source systems monitoring and alerting toolkit

-> Prometheus collects and stores its metrics as time series data

-> It provides out-of-the-box monitoring capabilities for the k8s container orchestration platform.

```
=============
Grafana
```

=============

-> Grafana is a analysis and monitoring tool

-> Grafana is a multi-platform open source analytics and interactive visualization web application.

-> It provides charts, graphs, and alerts for the web when connected to supported data sources.

-> Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore and share dashboards.

Note: Graphana will connect with Prometheus for data source.

###
How to deploy Grafana & Prometheus in K8S
##

-> Using HELM charts we can easily deploy Prometheus and Grafana

##
Install Prometheus & Grafana In K8S Cluster using HELM
###

```
# Add the latest helm repository in Kubernetes
$ helm repo add stable https://charts.helm.sh/stable

# Add prometheus repo to helm
$ helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts

# Update Helm Repo
$ helm repo update

# Search Repo
$ helm search repo prometheus-community

# install prometheus
$ helm install stable prometheus-community/kube-prometheus-stack

# Get all pods
$ kubectl get pods
```

Node: You should see prometheus pods running

```
# Check the services
$ kubectl get svc

# By default prometheus and grafana services are available within the cluster as
ClusterIP, to access them outside lets change it to NodePort.

# Edit Prometheus Service & change service type to NodePort then save and close
that file
$ kubectl edit svc stable-kube-prometheus-sta-prometheus

# Now edit the grafana service & change service type to NodePort then save and
close that file
$ kubectl edit svc stable-grafana


# Verify the service if changed to LoadBalancer
```

```
$ kubectl get svc

# Check in which nodes our Prometheus and grafana pods are running
$ kubectl get pods -o wide

=> Access Promethues server using below URL

                URL : http://node-ip:nodeport/

=> Access Grafana server using below URL

                URL : http://node-ip:nodeport/

=> Use below credentials to login into grafana server

UserName: admin
Password: prom-operator

=> Once we login into Grafana then we can monitor our k8s cluster. Grafana will
provide all the data in charts format.



##########
ELK Stack
##########

-> The ELK Stack is a collection of three open-source products â€" Elasticsearch,
Logstash, and Kibana

-> ELK stack provides centralized logging in order to identify problems with
servers or applications

-> It allows you to search all the logs in a single place


E stands for : Elastic Search --> It is used to store logs

L stands for : Log Stash --> It is used for processing logs

K stands for : Kibana --> It is an visualization tool


FileBeat : Log files

MetricBeat : Metrics

PacketBeat : Network data

HeartBeat : Uptime Monitoring


->  Filebeat collect data from the log files and sends it to logstash

-> Logstash enhances the data and sends it to Elastic search

-> Elastic search stores and indexes the data

-> Kibana displays the data stored in Elastic Search based on the request recieved

###########################
EFK Installation using HELM
###########################
```

Pre-requisites :

EKS Cluster
Nodes : 4 GB RAM
Client Machine with kubectl & helm configured

```
$ kubectl create ns efk

$ kubectl get ns

$ helm ls

$ helm repo add elastic https://helm.elastic.co

$ helm repo ls

$ helm show values elastic/elasticsearch >> elasticsearch.values

$ vi elasticsearch.values
```

-> replicas as 1 & masternodes as 1

```
$ helm install elasticsearch elastic/elasticsearch -f elasticsearch.values -n efk

$ helm ls -n efk

$ kubectl get all -n efk

$ helm show values elastic/kibana >> kibana.values

$ vi kibana.values
```

-> Change Service Type from ClusterIP to LoadBalancer

-> Change Port to 80 (default port is 5601)

```
$ helm install kibana elastic/kibana -f kibana.values -n efk

$ kubectl get all -n efk

$ helm install filebeat elastic/filebeat -n efk

$ helm install metricbeat elastic/metricbeat -n efk
```

Note: Access Kibana using Load Balancer DNS

======================
Kubernetes Running Notes
======================

```
1) What is Kubernetes
                        - Orchestration Platform
                        - To manage containers
                        - Developed by Google using Go language
                        - Google donated K8S to CNCF
                        - K8S first version released in 2015
                        - It is free & Open source
```

2) Docker Swarm Vs K8S

                        - Docker Swarm doesn't have Auto Scaling (Scaling is manual process)
                        - K8S supports Auto Scaling
                        - For Production deployments K8S is highly recommended
                        - Kubernetes is replacement for Docker Swarm

3) What is Cluster

- Group Of Servers
- Master Node(s)
- Worker Node(s)
- DevOps Enginner / Developer will give the task to K8S Master Node
- Master Node will manage worker nodes
- Master Node will schedule tasks to worker nodes
- Our containers will be created in Worker Nodes

4) Kubernetes Architecture

- Control Plane / Master Node / Manager Node
    - Api Server
    - Schedular
    - Control Manager
    - ETCD

- Worker Node (s)
    - Pods
    - Containers
    - Kubelet
    - Kube Proxy
    - Docker Runtime

5) How to communicate with K8S control plane ?

- Kubectl (CLI tool)

- Web UI Dashboard

===============================
Kubernetes Architecture Components
===============================

=> API Server : It is responsible to handle incoming requests of Control Plane

=> Etcd : It is internal database in K8S cluster, API Server will store requests / tasks info in ETCD

=> Schedular : It is responsible to schedule pending tasks available in ETCD. It will decide in which worker node our task should execute. Schedular will decide that by communicating with Kubelet.

=> Kubelet : It is a worker node agent. It will maintain all the information related to Worker Node.

=> Conroller-Manager : After scheduling completed, Controller-Manager will manage our task execution in worker node

=> Kube-Proxy : It will provide network for K8S cluster communication (Master Node <---> Worker Nodes)

=> Docker Engine : To run our containers Docker Engine is required. Containers will be created in Worker Nodes.

=> Container : It is run time instance of our application

=> POD : It is a smallest building block that we will create in k8s to run our containers.

========================
Kubernetes Cluster Setup
========================

1) Self Managed Cluster (We will create our own cluster)

    a) Mini Kube (Single Node Cluster)

    b) Kubeadm (Multi Node Cluster)

2) Provider Managed Cluster (Cloud Provide will give read made cluster) ---> Charges applies

    a) AWS EKS

    b) Azure AKS

    c) GCP GKE

=================================
Minikube Cluster setup on windows OS
================================

1) Download and install Docker Desktop software in Windows
        (URL : https://docs.docker.com/desktop/install/windows-install/)

2) Download and install Minikube (URL : https://minikube.sigs.k8s.io/docs/start/)

3) Download and install Kubectl (URL : https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/)

====================
Kubernetes Components
====================

1) Pods
2) Services
3) Namespaces
4) ReplicationController
5) ReplicaSet
6) DaemonSet
7) Deployments
8) StatefulSet
9) K8S Volumes
10) ConfigMap & Secrets
11) Ingress Controller
12) K8S Web Dashboard
13) RBAC (Role Based Access in K8S)
14) HELM Charts (Package Manager)
15) Grafana & Promethues (Monitoring Tools)
16) ELK Stack (Log Monitoring)
17) EKS Cluster (Provider Managed Cluster - Paid Service)

=============
PODS
=============

-> POD is a smallest building block in k8s cluster

-> In K8S, every container will be created inside POD only

-> POD always runs on a Node

-> POD represents a running process

-> POD is a group of one or more containers running on a Node

-> Each POD will have unique IP with in the cluster

-> We can create K8S pods in 2 ways

        1) Interactive Mode (By using kubectl command directley)

```
                Ex: $ kubectl run --name <pod-name>  image=<image-name>
--generator=run-pod/v1
```

        2) Declarative Mode (K8S Manifest YML file)

```
---
apiVersion :
kind:
metadata:
spec:
...
```

-> Once K8S manifest yml is ready then we can execute that using below kubectl command

```
                                        $ kubectl apply -f <file-name>
```

```
================================
Kubernates Sample POD Manifest YML
================================
---
apiVersion: v1
kind: Pod
metadata:
   name: javawebapppod
   labels :
      app: javawebapp
spec:
   containers:
    - name: javawebappcontainer
      image: ashokit/javawebapp
      ports:
         - containerPort: 8080
...
```

```
# To all get all pods which are created
$ kubectl get pods

# To create pod using pod manifest yml
$ kubectl apply -f <pod-manifest-yml>

# Checking pod creation status
$ kubectl get pods

# To describe the pod information
$ kubectl describe pod <pod-name>
```

```
# To get pod logs
$ kubectl logs <pod-name>

# To check in which node pod is running
$ kubectl get pods -o wide
```

***** Note: By default PODS are accessible only with in the Cluster, Outside of the Cluster We can't access PODS*******

=> To provide PODS access outside of the cluster we will use 'Kubernetes Service' concept

=============
K8S Service
==============

-> K8S service makes PODs accessible outside of the cluster also

-> In K8S we have 3 types of Services

1) Cluster IP
2) Node Port
3) Load Balancer   (Will work only with Provider Managed Cluster - we will learn this in EKS)

-> We need to Create k8s service manifest to expose PODS outside the cluster

```
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
spec:
   type: NodePort
   selector:
        app: javawebapp
   Ports:
        - port: 80
          targetPort: 8080
...
```

```
$ kubectl get svc

$ kubectl apply -f <service-manifest-yml>

$ kubectl get svc
```

Note: NodePort service will map our pod to a random port Number (Ex: 30002)

-> Enable Node Port in Security Group Inbound Rules

```
$ kubectl describe pod <pod-name>
```

Note: Here we can see in which Node our POD is running

-> We can access our application using below URL

URL : http://node-ip:node-port/java-web-app/

==============
Cluster IP
==============

-> It will expose our k8s service on a cluster with one internal ip

-> Cluster IP type service is accessible only with in cluster using Cluster IP

-> When we access cluster ip, it will redirect the request to POD IP

Note: POD is very short lived object, when pod is re-created POD ip will change hence it is not at all recommended to access pods using pod ips. To expose PODS with in cluster we can use 'Cluster IP' service

Note: ClusterIP service is accessible only with in cluster (can't accessed outside the cluster)

-> To expose POD using service, we will use POD label as a Selector in Service Manifest file like below

```
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
spec:
   type: ClusterIP
   selector:
        app: javawebapp # this is pod label
   Ports:
        - port: 80
          targetPort: 8080
...
```

```
$ kubectl apply -f <svc-manifest-yml>

$ kubectl get svc
```

===========
Node Port
==========

-> Node Port Service is used to expose our PODS outside the cluster also

-> When we use NodePort Service we can specify PORT Number, if we don't specify port number then k8s will assign one random port number for our service.

Q) What is the range of NodePort service PORT Number in k8S ?

Ans)   30000 - 32767

```
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
spec:
   type: NodePort
   selector:
        app: javawebapp
   Ports:
        - port: 80
          targetPort: 8080
          nodePort: 30002
...

$ kubectl get svc

$ kubectl apply -f <svc-manifest-yml>

$ kubectl get svc

$ kubectl delete service <service-name>
```

Note: Once we expose our POD using NodePort service then we can access our pod outside the cluster also.

```
# Get POD IP and POD Running Node IP
$ kubectl get pod -o wide
```

```
#URL To access our application

                http://pod-running-node-public-ip:nodeport/<context-path>/
```

===
Comibining Pod manifest and Service Manifest using single YML
===

```
---
apiVersion: v1
kind: Pod
metadata:
   name: javawebapppod
   labels :
      app: javawebapp
spec:
   containers:
    - name: javawebappcontainer
      image: ashokit/javawebapp
      ports:
         - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
spec:
  type: NodePort
  selector:
    app: javawebapp
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30002
...

$ kubectl apply -f <manifest-yml>
```

============

POD Lifecycle
============

-> When we make a request to create a POD then API Server will recieve our request

-> API Server will store our POD creation request in ETCD

-> Schedular will find un-scheduled pods and it will schedule them in Worker Nodes

-> The Node Agent (Kubelet) will see POD Schedule and it will fire Docker Engine

-> Docker Engine runs the Container

-> The entire POD lifecycle is store in ETCD.

Note : POD is ephemeral (very short lived object)

===============
K8S Namespaces
===============

-> Namespace represents a cluster inside the cluster

-> Namespaces are used to group k8s components

-> We can create multiple namespaces in single k8s cluster

-> Namespaces are logically isolated with each other

Note: In java we have packages concept to group the classes in k8s we have namespaces to group k8s components

-> We can get k8s namespaces using below command

```
                $ kubectl get ns  ( or )  $ kubectl get namespaces
```

-> K8S cluster providing below namespaces by default

```
                1) default
                2) kube-node-lease
                3) kube-public
                4) kube-system
```

Note: If we create any k8s component without giving namespace then k8s will consider 'default' namespace for that.

-> The remaining 3 namespaces will be used by k8s for cluster management.

-> It is not recommended to create our k8s components under kube-node-release, kube-public and kube-system.

```
# Command to get all k8s components

$ kubectl get all

# Command to get all k8s components of particular namespace

$ kubectl get all -n <namespace>

Ex : $ kubectl get all -n kube-system
```

-> It is highly recommended to create our k8s components under a custom namespace

# Command to create custom namespace - interactive approach

```
$ kubectl create ns ashokitns
```

-> We can create namespace using declarative approach also (manifest yml)

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: <insert-namespace-name-here>
...
```

```
$ kubectl apply -f <namespace-name>
```

Note: If we delete namespace all the components of that namespace also gets deleted.

# Command to delete namespace

```
$ kubectl delete ns ashokitns
```

=> When we execute below command the components of 'default' namespace got deleted.

```
                $ kubectl delete all --all
```

===================================
POD & Service Under Custom Namespace
===================================

```
---
apiVersion: v1
kind: Pod
metadata:
   name: javawebapppod
   labels :
      app: javawebapp
   namespace: ashokitns
spec:
   containers:
    - name: javawebappcontainer
      image: ashokit/javawebapp
      ports:
         - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
  namespace: ashokitns
spec:
  type: NodePort
  selector:
    app: javawebapp
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30002
...
```

```
$ kubectl apply -f <manifest-yml>

$ kubectl get all -n ashokitns

$ kubectl get pods -n ashokitns

$ kubectl get svc -n ashokitns

$ kubectl describe pod <pod-name> -n ashokitns

$ kubectl logs <pod-name> -n ashokitns

$ kubectl delete ns ashokitns
```


-> As of now we have created K8S POD manually using POD manifest file.

-> If we delete the POD then our application will be down, k8s not re-creating the POD

```
# command to delete the pod
$ kubectl delete pod <pod-name>

# check the pods
$ kubectl get pods
```

-> POD is not re-created by k8s because we have created POD manually.

-> It is not all recommended to create pods manully.

-> K8S provided below components/resources to create the PODS. Always we have to create pods using below resources.

1) ReplicationController
2) ReplicaSet
3) Deployment
4) DaemonSet
5) StatefulSet

-> If we create the PODS using above resources then K8S will take care of pods & Pods lifecycle.

===================

Replication Controller

===================

-> It is one of the key feature of K8S

-> It is responsible to manage POD life cycle

-> It is responsible to make sure given no.of pods are running for our application all the time.

Note: If any pod is crashed / deleted then it will replace that pod with new pod (Automatically it will create it, it is called as self healing capability).

-> Using Replication Controller we can scale up and scale down our PODS count

-> Create below manifest file to work with 'Replication Controller"

```
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: javawebapprc
spec:
  replicas: 3
  selector:
    app: javawebapp
  template:
    metadata:
      name: javawebapppod
      labels:
       app: javawebapp
    spec:
      containers:
      - name: javawebappcontainer
        image: ashokit/javawebapp
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
spec:
  type: NodePort
  selector:
    app: javawebapp
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30002
...
```

```
$ kubectl apply -f <rc.yml>

$ kubectl get rc

$ kubectl get pods

$ kubectl get svc

$ kubectl delete pod <pod-name>

$ kubectl scale rc <rc-name> --replicas <count>
```

Ex: $ kubectl scale rc javawebapprc --replicas 3

```
$ kubectl get pods -o wide
```

==============
ReplicaSet
==============

-> ReplicSet is the replacement for ReplicationController (It is next gen component in k8s)

-> ReplicaSet also manages Pod Lifecycle

-> ReplicaSet also mantains given no.of pod replicas at any point of time

-> We can scale up and we can scale down our POD replicas using ReplicasSet also

-> The only difference between ReplicationController and ReplicaSet is in 'selector'

-> ReplicationController supports 'Equlity Based Selector'

-> ReplicaSet supports 'Set Based Selector'

Equality Based Selector

```
selector:
   app: javawebapp
```

Set Based Selector

```
selector:
   matchLabels:
           app: javawebapp
           version: v1
           type: backend
```

-> Create below manifest to work with ReplicaSet

```
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: javawebapprs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: javawebapp
  template:
    metadata:
      name: javawebapppod
      labels:
        app: javawebapp
    spec:
      containers:
      - name: javawebappcontainer
        image: ashokit/javawebapp
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
spec:
  type: NodePort
  selector:
    app: javawebapp
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30002
...
```

```
$ kubectl apply -f <rs.yml>

$ kubectl get rs

$ kubectl get pods -o wide

$ kubectl delete pod <pod-name>

$ kubectl scale rs <rs-name> --replicas <count>

Ex: $ kubectl scale rs javawebapprs --replicas 3

$ kubectl get pods -o wide
```

================
K8S Deployments
================

-> Deployment is the most recommended approach to deploy our application in k8s cluster

-> Deployment is used to tell how to create pods on the k8s cluster

-> Using Deployment we can scale up and we can scale down our POD Replicas

-> Deployment supports Roll Out and Roll Back

-> Below are the key benefits of k8s deployment

1) Deploy a RS

2) Update PODS

3) Rollback to older deployment

4) Scale up & scale down

Note: When we use ReplicationController or ReplicaSet latest images cant be updated directley. We have to delete RC or RS to deploy lastest code (when we delete RC or RS all pods gets deleted then application will be down)

-> When we use Deployment concept we can easily update latest code without deleting Deployment. We can achieve zero downtime.

-> K8S deployment having below deployment strategies

1) ReCreate

2) RollingUpdate

-> ReCreate strategy means it will delete all the existing pods and it will create new pods (downtime will be there)

-> RollingUpdate strategy means it will delete the pod and it will re-create the pod one by one.

Note: If we don't specify deployment strategy in manifest yml, then k8s will consider 'RollingUpdate' as default deployment strategy.

-> We can use below 'Deployment' manifest yml to create deployment of our java web application in K8S cluster

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: javawebappdeployment
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: javawebapp
  template:
    metadata:
      name: javawebapppod
      labels:
        app: javawebapp
    spec:
      containers:
      - name: javawebappcontainer
        image: ashokit/javawebapp
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: javawebappsvc
spec:
  type: NodePort
  selector:
    app: javawebapp
  ports:
    - port: 80
      targetPort: 8080
      nodePort: 30002
...
```

```
$ kubectl get deployment

$ kubectl apply -f <deployment-manifest-yml>

$ kubectl get deployment

$ kubectl get pods

$ kubectl get svc

$ kubectl get pods -o wide

$ kubectl scale deployment javawebappdeployment --replicas 3

$ kubectl logs <podname>
```

=======================
Blue & Green Deployment
=======================

=> It is one of the application release model with zero downtime

```
=======================================
Steps to perform Blue-Green Deployment Model
=======================================
```

-> Create 'blue deployment' using below manifest yml (blue-deployment.yml)

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-boot-demo-deployment-blue
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: k8s-boot-demo
      version: v1
      color: blue
  template:
    metadata:
      labels:
        app: k8s-boot-demo
        version: v1
        color: blue
    spec:
      containers:
        - name: k8s-boot-demo
          image: ashokit/javawebapp
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
...
```

```
$ kubectl apply -f blue-deployment.yml
$ kubectl get pods
```

=> Create a service for blue pods exposing as NodePort (service-live.yml)

```
---
apiVersion: v1
kind: Service
metadata:
  name: k8s-boot-live-service
spec:
  type: NodePort
  selector:
    app: k8s-boot-demo
    version: v1
  ports:
    - name: app-port-mapping
      protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 30002
...
```

```
$ kubectl apply -f live-service.yml
$ kubectl get svc
```

=> After creating the service access our application using below URL

                http: // node-ip : 30002 / java-web-app/

```
===============================
Deploying Latest Code as Green
================================
```

=> Create green pods using below deployment manifest (green-deployment.yml)

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-boot-demo-deployment-green
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: k8s-boot-demo
      version: v2
      color: green
  template:
    metadata:
      labels:
        app: k8s-boot-demo
        version: v2
        color: green
    spec:
      containers:
        - name: k8s-boot-demo
          image: ashokit/mavenwebapp
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
...
```

```
$ kubectl apply -f green-deployment.yml
$ kubectl get pods
```

=> To test green pods we are creating Pre-Prod Service (service-preprod.yml)

```
---
apiVersion: v1
kind: Service
metadata:
  name: k8s-boot-service-preprod
spec:
  type: NodePort
  selector:
    app: k8s-boot-demo
    version: v2
  ports:
    - name: app-port-mapping
      protocol: TCP
      port: 8080
      targetPort: 8080
      nodePort: 30092
...
```

```
$ kubectl apply -f service-preprod.yml
$ kubectl get svc
```

=> Access the application using pre-prod service

                http://node-ip:30092/maven-web-app/

Note: Once pre-prod testing completed then v2 pods we need to make live

==============================
How to make Green PODS as Live ?
================================

-> Go to service-live.yml and change selector to 'v2' and apply

```
$ kubectl apply -f service-live.yml
```

-> After applying live service with v2 then our live service will point to green pods (latest code)

        URL : http://node-ip:30002/maven-web-app/

================
DeamonSet
================

-> It is also one of the k8s resource used to create PODS in k8s cluster

-> DeamonSet will create copy of the pod on each worker node

==================================
Some typical uses of a DaemonSet are:
==================================

1) Running a cluster storage daemon on every node
2) Running a logs collection daemon on every node
3) Running a node monitoring daemon on every node

=> Create fluentd-elasticsearch pod using daemonset

```
$  kubectl apply -f https://k8s.io/examples/controllers/daemonset.yaml
```

=> fluentd-elasticsearch will be created on 'kube-system' namespace

=> Get all the k8s components belongs to kube-system namespace

```
$ kubectl get all -n kube-system

$ kubectl get pods -o wide -n kube-system
```

Note: We can see that fluentd pods are runnnig on all the nodes

===================
Config Map & Secrets
===================

-> We shouldn't hardcode properties in the application, because from environment to environment application properties might change.

                        Ex: DEV DB and PROD DB properties will be different

-> ConfigMap & Secrets are used to avoid hard coding properties in the application

                Ex: database properties, smtp properties

-> ConfigMap is used to store the data in the form of key-value (non-confidential)

-> A ConfigMap allows you to decouple environment-specific configuration from your container images, so that your applications are easily portable.

-> If we use ConfigMap concept for application environment properties then we can deploy our application in any environment without re-creating images.

-> Secrets is also one of the kubernetes resource

-> Secrets is used to store confendential data in key-value format

                Ex: username, password, token etc...

-> Using Secrets we will store confidentials data in encrypted format

```
=====================
Working with ConfigMap
======================
```

-> Below is the example for configmap

-> create a manifest file like below

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: weshopify-db-config-map
  labels:
    storage: weshopify-db-storage
data:
 DB_DRIVER_NAME_VALUE: com.mysql.cj.jdbc.Driver
 DB_HOST_SERVICE_NAME_VALUE: weshopify-app-db-service
 DB_SCHEMA_VALUE: weshopify-app
 DB_PORT_VALUE: "3306"
...
```

```
$ kubectl apply -f <configMap-manifest.yml>
```

-> To get/refer data from config-map, in our pod manifest we will use below tag

```
- name: DB_DRIVER_CLASS
              valueFrom :
                configMapKeyRef :
                  name : weshopify-db-config-map
                  key :  DB_DRIVER_NAME_VALUE
```

```
==================
Working with Secrets
==================
```

-> Below is the example for Secrets

-> Create secrets yml file with below content

```
---
apiVersion: v1
kind: Secret
metadata:
 name: weshopify-db-config-secrete
 labels:
  secrete: weshopify-db-config-secrete
data:
 DB_USER_NAME_VALUE: cm9vdA==
 DB_PASSWORD_VALUE: cm9vdA==
type: Opaque
...
```

```
$ kubectl apply -f <secrets-manifest-yml>
```

=> We can get data from Secrets in our POD manifest using below tag

```
 - name: DB_PASSWORD
              valueFrom :
                secretKeyRef :
                  name : weshopify-db-config-secrete
                  key :  DB_PASSWORD_VALUE
```

==============================
Hardcoded Application Properties
=============================

```
spring:
  datasource:
    username: ashokit
    password: ashokit@123
    url: jdbc:mysql://localhost:3306/sbms
    driver-class-name: com.mysql.jdbc.Driver
```

===
Application Properties with Environment Variables
===

```
spring:
  datasource:
    username: ${DB_USERNAME:ashokit}
    password: ${DB_PASSWORD:ashokit@123}
    url: ${DB_URL:jdbc:mysql://localhost:3306/sbms}
    driver-class-name: ${DB_DRIVER: com.mysql.jdbc.Driver}
```

====================================
Deploying MySQL Database in K8S Cluster
===================================

****** Git Url for K8S Manifest Files :
https://github.com/ashokitschool/kubernetes_manifes_ymls.git ********

```
# Create Config Map
$ kubectl apply -f <config-manifest-yml>

# Create Secret
$ kubectl apply -f <Secret-manifest-yml>

# Create PV
$ kubectl apply -f <PV-manifest-yml>
```

```
# Create PVC
$ kubectl apply -f <PVC-manifest-yml>

# Create Database Deployment
$ kubectl apply -f <Database-manifest-yml>

# Check pods which are created
$ kubectl get pods
```

Note: We should be able to get 'database' as a pod

******************************* With above steps our Database Deployment Completed **************************************

```
# Connecto database pod
$ kubectl exec -it <pod-name> bash

# Connect to database
$ mysql -h localhost -u root -p
```

Note: It will ask for password, enter password as root.

```
# Check databases available in mysql
$ show databases;

# Use the database
$ use weshopify-app;

# create a table
$ create table emp(emp_id int, emp_name varchar(50));

# display all tables
$ show tables;

# insert record
$ insert into emp values(101, 'Raju');
$ insert into emp values(102, 'Rani');

# Retrieve records from table
$ select * from emp;

# exit from mysql database
$ exit

# exit from the pod
$ exit
```

Note: Finally we are back to control-plane

=====================================
Stateless application vs Stateful Application
=====================================

=> Stateless applications can't store the data permanentley. For every request they will get new data and it will process that data

Ex: Node JS app, Nginx app, Boot app etc....

=> Stateful applications means they will store the data permanentley and they will keep track of data

Ex: MySQL, Oracle, Postgres etc....

=> To run stateful containers in k8s cluster we will use StatefulSet concept.

=> StatefulSet will assign a sticky identity for each pod starting from zero (0)

=> In Stateful set primary and secondary pods will be created.

=> Once primary pod is created then by copying primary pod data secondary pod will be created

Note: New POD will be created by copying previous pod data. If previous pod is pending state then new pod will not be created.

=> In StatefulSet, pods will be deleted in reverse order (last to first)

===
What is the difference between Deployment and StatefulSet ?
===

-> Deployment will create the PODS with random ids
-> Deployment will scale down pods in random order
-> Deployment pods are stateless pods

-> StatefulSet will create the PODS with sticky identity
-> StatefulSet will scale down pods in reverse order
-> StatefulSet pods are statefull

Note:

-> For application deployment we will use 'DEPLOYMENT'
-> For database deployment we will use 'STATEFULSET'

#### Stateful Set Manifest YMLS Repo :
https://github.com/ashokitschool/kubernetes_statefulset_ymls.git ####

############################
Kubernetes web dashboard
##########################

=> We can communicate with Kubernetes cluster in 2 ways

1) Kubectl (CLI)

2) Web dashboard (UI Based)

=> Using kubectl we can execute commands to deploy our components in k8s cluster

=> Web Dashboard will provide user interface to perform our operations in k8s cluster

======================
K8S Web Dashboard Setup
======================

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.0/aio/deploy/recommende
d.yaml

$ kubectl -n kubernetes-dashboard get pods -o wide
$ kubectl -n kubernetes-dashboard get svc

# Edit k8s dashboard service and change it to NodePort
$ kubectl -n kubernetes-dashboard edit svc kubernetes-dashboard
```

Note: Check kubernetes-dashboard node port and enable that Node PORT in security group

```
# Check in which node kubernete-dashboard POD is running
$ kubectl get pods -o wide -n kubernetes-dashboard


# Access k8s web ui dashboard using below URL
```

URL : https://node-public-ip:node-port/

```
#create admin user with below yml

$ vi create-user.yml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
...


$ kubectl apply -f create-user.yml


# create cluster role binding
$ vi cluster-role-binding.yml

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
...

$ kubectl apply -f cluster-role-binding.yml


# Get the bearer token
$ kubectl -n kubernetes-dashboard create token admin-user
```

Note: Copy the token and enter in kubernetes web dashboard login.

======================
Kubernetes Ingress
======================

-> Deploy two application Into K8S using Cluster IP Service

```
--------------------------------
java-web-app-deploy.yml-------------------------------------
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: javawebappdeployment
spec:
 replicas: 1
 strategy:
    type: Recreate
 selector:
   matchLabels:
     app: javawebapp
 template:
  metadata:
   name: javawebapppod
   labels:
     app: javawebapp
  spec:
    containers:
    - name: javawebappcontainer
      image: ashokit/javawebapp
      ports:
      - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
 name: javawebappsvc
spec:
  type: ClusterIP
  selector:
   app: javawebapp
  ports:
   - port: 80
     targetPort: 8080
...

---------------------------------------------------------maven-web-app-deploy.yml--
---------------------------------------------------------
apiVersion: apps/v1
kind: Deployment
metadata:
 name: mavenwebappdeployment
spec:
 replicas: 2
 selector:
   matchLabels:
     app: mavenwebapp
 template:
  metadata:
   name: mavenwebapppod
   labels:
     app: mavenwebapp
  spec:
    containers:
    - name: mavenwebappcontainer
      image: ashokit/mavenwebapp
      ports:
      - containerPort: 8080
      resources:
        requests:
         cpu: 200m
         memory: 1Gi
        limits:
         cpu: 500m
         memory: 2Gi
---
apiVersion: v1
kind: Service
metadata:
 name: mavenwebappsvc
spec:
  type: ClusterIP
  selector:
   app: mavenwebapp
  ports:
   - port: 80
     targetPort: 8080
-----------------------------------------------------------------------------------
-----------------------------------------------------------------
```

```
$ kubectl apply -f javawebapp.yml

$ kubectl apply -f mavenwebapp.yml
```

-> Now we have 2 services running in K8S cluster with Cluster IP service.

-> We will use Ingress to provide routing for these two services from external traffic

-> K8S ingress is a resource to add rules for routing traffic from external sources to the services in the k8s cluster

-> It requires an ingress controller for routing the rules specified in the ingress object

-> Ingress controller is typically a proxy service deployed in the cluster. It is nothing but a Kubernetes deployment exposed to a service.

############
Ingress Setup
############

```
# git clone k8s-ingress
$ git clone https://github.com/ashokitschool/kubernetes_ingress.git

$ cd kubernetes_ingress

# Create namespace and service-account
$ kubectl apply -f common/ns-and-sa.yaml

# create RBAC and configMap
$ kubectl apply -f common/

# Deploy Ingress controller
```

-> We have 2 options to deploy ingress controller

1) Deployment
2) DaemonSet

```
$ kubectl apply -f daemon-set/nginx-ingress.yaml

# Get ingress pods using namespace
$ kubectl get all -n nginx-ingress

# create LBR service

$ kubectl apply -f service/loadbalancer-aws-elb.yaml
```

Note: It will generate LBR DNS

```
*******************   Map LBR dns to route 53 domain  *************************
```

-> Create Ingress kind with rules

```
============================
Path Based Routing
============================
```

```
$ vi ingress-rules-routes.yml

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-resource
spec:
  ingressClassName: nginx
  rules:
  - host: ashokit.org
    http:
      paths:
      - pathType: Prefix
        path: "/java-web-app"
        backend:
         service:
          name: javawebappsvc
          port:
           number: 80
      - pathType: Prefix
        path: "/maven-web-app"
        backend:
         service:
          name: mavenwebappsvc
          port:
           number: 80
...
```

```
=================================
Access the application using below URL
=================================
```

URL-1 : www.ashokit.org/java-web-app

URL-2 : www.ashokit.org/maven-web-app

```
#######################
Kubernetes Monitoring
#######################
```

=> We can monitor our k8s cluster and cluster components using below softwares

1) Prometheus
2) Grafana

=============
Prometheus
=============

-> Prometheus is an open-source systems monitoring and alerting toolkit

-> Prometheus collects and stores its metrics as time series data

-> It provides out-of-the-box monitoring capabilities for the k8s container orchestration platform.

=============
Grafana
=============

-> Grafana is a analysis and monitoring tool

-> Grafana is a multi-platform open source analytics and interactive visualization web application.

-> It provides charts, graphs, and alerts for the web when connected to supported data sources.

-> Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore and share dashboards.

Note: Graphana will connect with Prometheus for data source.

###
How to deploy Grafana & Prometheus in K8S
##

-> Using HELM charts we can easily deploy Prometheus and Grafana

##
Install Prometheus & Grafana In K8S Cluster using HELM
###

```
# Add the latest helm repository in Kubernetes
$ helm repo add stable https://charts.helm.sh/stable

# Add prometheus repo to helm
$ helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts

# Update Helm Repo
$ helm repo update

# Search Repo
$ helm search repo prometheus-community

# install prometheus
$ helm install stable prometheus-community/kube-prometheus-stack

# Get all pods
$ kubectl get pods
```

Node: You should see prometheus pods running

```
# Check the services
$ kubectl get svc
```

```
# By default prometheus and grafana services are available within the cluster as
ClusterIP, to access them outside lets change it to NodePort.
```

```
# Edit Prometheus Service & change service type to NodePort then save and close
that file
$ kubectl edit svc stable-kube-prometheus-sta-prometheus
```

```
# Now edit the grafana service & change service type to NodePort then save and
close that file
$ kubectl edit svc stable-grafana
```

```
# Verify the service if changed to LoadBalancer
$ kubectl get svc
```

```
# Check in which nodes our Prometheus and grafana pods are running
$ kubectl get pods -o wide
```

=> Access Promethues server using below URL

```
                URL : http://node-ip:nodeport/
```

=> Access Grafana server using below URL

```
                URL : http://node-ip:nodeport/
```

=> Use below credentials to login into grafana server

UserName: admin
Password: prom-operator

=> Once we login into Grafana then we can monitor our k8s cluster. Grafana will provide all the data in charts format.

##########
ELK Stack
##########

-> The ELK Stack is a collection of three open-source products â€” Elasticsearch, Logstash, and Kibana

-> ELK stack provides centralized logging in order to identify problems with servers or applications

-> It allows you to search all the logs in a single place

E stands for : Elastic Search --> It is used to store logs

L stands for : Log Stash --> It is used for processing logs

K stands for : Kibana --> It is an visualization tool

FileBeat : Log files

MetricBeat : Metrics

PacketBeat : Network data

HeartBeat : Uptime Monitoring

-> Filebeat collect data from the log files and sends it to logstash

-> Logstash enhances the data and sends it to Elastic search

-> Elastic search stores and indexes the data

-> Kibana displays the data stored in Elastic Search based on the request recieved

##########################
EFK Installation using HELM
##########################

Pre-requisites :

EKS Cluster
Nodes : 4 GB RAM
Client Machine with kubectl & helm configured

```
$ kubectl create ns efk

$ kubectl get ns

$ helm ls

$ helm repo add elastic https://helm.elastic.co

$ helm repo ls

$ helm show values elastic/elasticsearch >> elasticsearch.values

$ vi elasticsearch.values
```

-> replicas as 1 & masternodes as 1

```
$ helm install elasticsearch elastic/elasticsearch -f elasticsearch.values -n efk

$ helm ls -n efk

$ kubectl get all -n efk

$ helm show values elastic/kibana >> kibana.values

$ vi kibana.values
```

-> Change Service Type from ClusterIP to LoadBalancer

-> Change Port to 80 (default port is 5601)

```
$ helm install kibana elastic/kibana -f kibana.values -n efk

$ kubectl get all -n efk

$ helm install filebeat elastic/filebeat -n efk

$ helm install metricbeat elastic/metricbeat -n efk
```

Note: Access Kibana using Load Balancer DNS

---===============
AWS - EKS
===============

-> EKS stands for "Elastic Kubernetes Service"

-> EKS is a fully managed AWS service

-> EKS is the best place to run K8S applications because of its security, reliability and scalability

-> EKS can be integrated with other AWS services such as ELB, CloudWatch, AutoScaling, IAM and VPC

-> EKS makes it easy to run K8S on AWS without needing to install, operate and maintain your own k8s control plane.

-> Amazon EKS runs the 'K8S control Plane' across three availability zones in order to ensure high availability and it automatically detects and replaces unhealthy masters.

-> AWS will have complete control over Control Plane. We don't have control on Control Plane.

-> We need to create Worker Nodes and attach to Control Plane.

Note: We will create Worker Nodes Group using ASG Group

-> Control Plane Charges + Worker Node Charges (Based on Instance Type & No.of Instances)

###############
Pre-Requisites
###############

=> AWS account with admin priviliges

=> Instance to manage/access EKS cluster using Kubectl (K8S-Client-VM)

=> AWS CLI access to use kubectl utility

#################################
Steps to Create EKS Cluster in AWS
#################################

Step-1) Create VPC using Cloud Formation (with below S3 URL)

URL :
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml

Stack name : EKSVPCCloudFormation

Step-2) Create IAM role in AWS

-> Entity Type : AWS Service

-> Select Usecase as 'EKS' ==> EKS Cluster

-> Role Name : EKSClusterRole (you can give any name for the role)

Step-3) Create EKS Cluster using Created VPC and created IAM Role

-> Cluster endpoint access : Public & Private

Step-4) Create RedHat ec2 Instance (K8S_Client_Machine)

-> Connect to K8S_Client_Machine using Mobaxterm

######## Install Kubectl with below commands ###############

```
$ curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

$ kubectl version --client
```

########## Install AWS CLI in K8S_Client_Machine with below commands ##############

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$ sudo yum install unzip
$ unzip awscliv2.zip
$ sudo ./aws/install
```

############ Configure AWS Cli with Credentials ###############

-> We can create AWS Access Keys using IAM service

Access Key ID: AKIAYG3UVCXWDIVZ6GMD
Secret Access Key: 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

```
$ aws configure
```

Note: We can use root user accesskey and secret key access

##

```
$ aws eks list-clusters

$ ls ~/.
```

############# Update kubeconfig file in remote machine from cluster using below command ##############

```
$ aws eks update-kubeconfig --name <cluster-name> --region <region-code>
```

Ex: aws eks update-kubeconfig --name EKSCluster --region ap-south-1

############# ##
##############

Step-5) Create IAM role for EKS worker nodes (usecase as EC2) with below policies

Entity Type : AWS Service

a) AmazonEKSWorkerNodePolicy
b) AmazonEKS_CNI_Policy
c) AmazonEC2ContainerRegistryReadOnly

Step-6) Create Worker Node Group (Go to cluster -> Compute -> Node Group)

-> Select the Role we have created for WorkerNodes

-> Use t2.medium

-> Min 2 and Max 5

Step-7) Once Node Group added then check nodes in K8s_client_machine

```
$ kubectl get nodes

$ kubectl get pods --all-namespaces
```

Step-8) Create POD and Expose the POD using NodePort service

Note: Enable NODE PORT in security Group to access that in our browser

#########
Conclusion
#########

1) Delete Worker Node Group

2) Delete EKS Cluster

3) Delete VPC + NAT Gateway + Elastic IP

#################################
RDS (Relational Database Service)
#################################

-> Every application will contain 3 layers

1) Front End (User interface)

2) Back End (Business Logic)

3) Database

-> End users will communicate with our application using frontend (user interface)

-> When end-user performs some operation in the front-end then it will send request to backend. Backend contains business logic of the application.

-> Backend logic will communicate with Database to perform DB operations

(insert / update / retrieve / delete)

#######################################
Challenges with Database Setup On our own
#######################################

1) Setup Machine to install Database Server
2) Purchase Database Server License
3) Install Database Server in our Machine
4) Setup Network for our machine
5) Setup power for machine
6) Setup a server room to keep our machines

```
7) Setup AC for room for cool temperature
8) Setup Security for room
9) Setup Database backups


-> If we use AWS cloud, then AWS will take care of all the above works which are
required to setup Database for our application

-> In AWS, we have RDS service to create and setup database required for our
applications

-> We just need to pay the money and use Database using AWS RDS service. DB setup
and maintenence will be taken care by AWS people.


********* Using RDS we can Easily set up, operate, and scale a relational database
in the cloud *******

######################################
Steps to create MYSQL DB using AWS RDS
######################################

1) Login into AWS management console

2) Goto RDS Service

3) Click on 'Create Database'

                Choose a database creation method : Standard Create
                Engine Option : MySQL
                Template : Free Tier
                DB instance Identifier : ihis (Note : you can give anything)
                Username : admin
                Password : Choose a passord
                public access: Yes
                initial database name : hdfcdb (Note: you can give anything)

4) Click on 'Create Database' (It will take few minutes of time to create)

        Note: Notedown username and password of the database

5) Once Database created, it will provide database Endpoint URL to access

###################
MySQL DB Properties
###################

Endpoint : database-1.cnhsodrv3nqx.ap-south-1.rds.amazonaws.com
Uname : admin
Pwd : ashokit123
Port : 3306 (it is default port for mysql db)
Database Name : hdfcdb


Note: We need to provide DB properties to project development team / testing team

##########################
Steps to test MYSQL DB Setup
###########################

1) Download and install Visual Studio using below link

        Link : https://aka.ms/vs/17/release/vc_redist.x64.exe

2) Download and install MySQL Workbench using below link
```

        Link : https://dev.mysql.com/downloads/workbench/

3) Create Database Connection in MySQL workbench using Database properties

4) Once we are able to connect with Database then we can execute below queries in Workbench

```
###############
MySQL Queries
###############

show databases;

use hdfcdb;

show tables;

create table emp_dtls(emp_id integer(10), emp_name varchar(20), emp_salary
integer(10));

insert into emp_dtls values(101, 'Raju', 5000);

insert into emp_dtls values(102, 'Rani', 6000);

insert into emp_dtls values(103, 'Ashok', 7000);

select * from emp_dtls;
```

```
#################################
Working with MySQL client in Ubuntu
#################################

$ sudo apt-get update

$ sudo apt-get install mysql-client

$ mysql -h <endpoint> -u <username> -p (click enter and give password)
```

```
#######################################
Working with MySQL client in AMAZON Linux
#######################################

$ sudo yum update

$ sudo yum install mysql

$ mysql -h <endpoint-url> -u <username> -p (click enter and give password)
```

-h : It represents host (DB edpoint URL)

-u : It represents DB username

-p : It represents DB password

```
Ex:   mysql -h database-1.cqrc41bryfex.ap-south-1.rds.amazonaws.com -u admin -p
```

IAM (Identity & Access Management)

==
-> An AWS Identity and Access Management (IAM) user is an entity that you create in AWS to represent the person or
 application that uses it to interact with AWS Services.

-> AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.

We can use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

-> IAM helps protect against security breaches by allowing administrators to automate numerous user account related tasks.

-> Best practice of using the root user only to create your first IAM user.

-> Enable Multi Factory Authentication (MFA) for Root Usr

-> By using Google Authenticator App we can configure "Virtual MFA"

Best Practices:
===============
- When we login AWS using 'email' and 'password', that has complete access to all AWS services and resources in   the account (Root account).

- Strongly recommended that you do not use the "root user" for your everyday tasks, even the administrative ones.

- Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

- IAM user is truely global, i.e, once IAM user is created it can be accessible in all the regions in AWS.

- Amazon S3 also considered as Global but, it is not truely global. When we create a bucket in S3
  it displays all the buckets of other regions in one place , so that is the reason we are calling AmazonS3 is Global (but partly global).

- But IAM is 100% Global. Once you create IAM user you can use it anywhere in all the regions.

```
1. Main things in IAM is
        -Roles
        -Users
        -Policies / Permissions
        -Groups

2. IAM users can be accessible by the following 3 ways.
        -through AWS console
        -CLI (Command Line Interface)
        -API (fast glaciers)
```

3. In MNCs , permissions will not be provided for individual users. Create the Groups and add the users into it.
    Users & Groups are for the Endusers.
    Roles are for the AWS Services.

Steps:
====
1. Create an IAM user

```
   Services - Security, Identity, & Compliance - IAM
   Users---<Add user>
        User name* = Iamuser1
        Access type  = 'select' both "Programmatic Access"
                                "AWS Management Console access"


        Console password = 'select'
                           custom Password =  (********somepassword eg:test1234)

                click <NextPermissions>
        (Note: we are not providing any permissions as of now, just <create user>)

        Once the IAM user has been created.
                AccessKeyID =AKIAIEJH7Z3FDKH36YWQ
                Secretaccesskey=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

        (Note: Once you close this window, AccessKeyID and Secret Accesskey has
gone, so save it somewhere)

  - Best Practice is never give an individual permissions to the user, as users
will be changed frequently, when they left the organization.
    So Need to create the Groups and assign the users to it.

2. Group
        <create new group>
        Groupname =admins
        (Note: no need add any policy now).
        <creategroup>

3. Add user to this group

        click on newly create group 'admins'
        <Add users to Group>

        GroupARN =arn:aws:iam::540105522204:group/admins

  -Always add the permissions to the 'Groups' level not  to the 'users' level. Its
a Best Practice in the real-time.




*****************
Policies:
*****************
 - When we want to add the permissions to the the groups is through the 'Policies'.
 - Default AWS Policies are appear in'Orange color Icons'
 - One disadvantage of AWS Default Policies are , we can't customize the policies
to apply to the Groups.
 - To provide customized policies to apply to Groups, we need to create the new one
and apply to the Groups.

4. Now, we will add 'Administrator Access' Permissions to the user(Iamuser1) we
create.

        Groups -Admins-tab<permissions> ---<AttachPolicy>---'select' Administrator
Access----<AttachPolicy>

    -Dashboard -Customize the IAM link replacing the ID with any name. To Hide the
ID need to customize.

        IAM user sigin in

        https://4234324234.signin.aws.amazon.com/console
        After Customize

        https://classroomuser.signin.aws.amazon.com/console

- Open the new tab in the browser
        https://classroomuser.signin.aws.amazon.com/console

        IAMuser =Iamuser1
        password=test1234

5. Now need to login using the IAM user, which we created.

        Once login , we can launch an EC2 instance.As this user(Iamuser1) is
provided with Admin access.

=======================================================================
Requirement:
=========++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

I got an requirement to create a new user and he should be able to do only 'stop'
and 'start' , 'reboot' select instances only.
He should not have the permissions to terminate the EC2 Instances.
He should not have the permissions to create the new EC2 Instance.

1. Login to your AWS Console with your root login.

2. IAM -Create another user
        User name* = Iamuser2
        Access Type ='select'  "AWS Management Console access"

        'select' CustomPassword ="<somepassword>"
        <NextPermissions>
        Not selecting any group here
        <createuser>

3. Signout and Login using the 'Iamuser2' and its credentials

        Open browser
                https://classroomuser.signin.aws.amazon.com/console
        login with Iamuser2 credentials

        Services ---EC2
        you will get an 'Authorization Error'

4.  To view EC2 instances need to provide read permission to the user 'Iamuser2'.
     - using Tags, we can provide permissions to this user.

     Login using the Root user
        EC2 Instances
        Select the Running Instance
         click on tab <Tags>
                add new tag
                   Key =user
                   Value=Iamuser2
                    <save>

5. Using this we can restrict the user to create EC2 instances. We can allow him to
do only 'stop' and 'start' Instances.
    For this, need to write the custom scripts.

        Open the browser search for ='restrict aws user ec2 instance'
        https://aws.amazon.com/premiumsupport/knowledge-center/restrict-ec2-iam/

        copy the script and open in any editor and customize it.
        arn:aws:ec2:us-east-1:111122223333:instance/*"
        (Note: For every service we have arn (amazon resource name), but for EC2
there is no arn naming)

        InterviewQuestion:If anyone ask you , arn is not displaying for the EC2
instances?
        Ans:Simply say that, ARN is not visible for the EC2 instances, but for the
other services like S3, we have                         ARN url.



copy the script

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:RebootInstances"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/Owner": "Bob"
                }
            },
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/*"
            ],
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        }
    ]
}



After Customization
===================

        {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:RebootInstances"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/user": "Iamuser2"
                }
            },
            "Resource": [
                "arn:aws:ec2:us-east-1:449938344550:instance/*"
            ],
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        }
    ]
}
```

=============
Note:
        449938344550 = Root AccountID

```
6. copy the script after customization
    IAMUser
      Policies ---<createPolicy>---'select' JSON tab
        paste the customized script.

        <ReviewPolicy>

7. Review Policy
        Name ='UserRestrictEC2Instance'
        <createpolicy>

8. Now, need to add this policy to the user or groups.
        select Users
             'Iamuser2' ---Permissions(tab)-- Add Permissions ---AttachExisting
Policies directly
             Filter policies ='UserRestrictEC2Instance'

              Select the policy(UserRestrictEC2Instance')
---<Review>--<AddPermissions>

9. Login to IAM user console
        Iamuser2/password

   - Now Try to Terminate the EC2 Instance. It throws an error
   - Try to Launch an EC2 instance , it throws an error.
     Like this we can restrict the user by creating some policies and apply to it.
     AWS provides the readymade(default) policies we need to customize as per our
requirement.
```

=======================
Simple Storage Service (S3)
=======================

######### Amazon S3 : Object storage built to store & retrieve any amount of data from anywhere ########

######### Note: 5 GB of S3 standard storage for 12 months with the AWS Free Tier ########

-> S3 is a storage service in AWS cloud

-> S3 is unlimited storage

-> S3 is object based storage

-> In s3 we can store all flat files (any type of file)

-> We can upload, download and access files from S3 at any point of time

-> The files in s3 can't be executed

-> We can't install OS, DB etc in S3

-> We can't attach S3 to EC2 instance but we can access s3 buckets data from EC2 instance

-> S3 supports static website hosting

-> S3 is cheaper than EC2

-> S3 is serverless

-> In S3 we will store data in buckets. Bucket is a container & bucket contains objects

```
        object = file
        key is name of the object
```

-> S3 is global but buckets are regional

-> Bucket names are universal or unique

Note: always create a bucket with your company name or project name.

-> We can't create one bucket inside another bucket

-> We can create multiple buckets in multiple regions

-> Max no.of buckets you can create in S3 is 100 (soft limit)

-> By default buckets are private, if required we can make it public.

create bucket -> inside that create folder called photos -> inside that upload puppy.jpg

Note: Every object will have its own url/endpoint.

Ex: http://8pmbukcet.s3.amazonaws.com/photos/puppy.jpg

bucketname+ domain + object name

-> S3 uses WORM model (Write Once and Read Many)

=============================
Static Website Hosting using S3
=============================

=> Collection of web pages is called as Website

=> Websites are 2 types

1) Static Website

2) Dynamic Website

=> The website which will display same content/response for every user is called as Static website

=> The website which will display response based on user and user request is called as Dynamic Website.

########## Website Code Git Repo :

https://github.com/ashokitschool/s3_static_website_hosting.git ###########

1) Create a bucket in S3

        - Enter unique name for bucket

        - uncheck block public access

2) Upload Website content files in bucket (assets folder, index.html and error.html) make sure you given public access for files which we are uploading.

3) Go to Bucket Properties tab -> Enable Static Website hosting and configure index and error pages

index.html for main content
error.html for wrong url

4) It will display URL, access that URL

#############
Versioning
#############

-> Versioning is like a backup tool

-> By default versioning is not enabled on the bucket

-> Versioning is enabled on bucket level but applied on object level.

-> when we upload same file multiple times then versions will be created.

corejava.jpg (v1, v2, v3) - v3 latest version

advjava.jpg (v1, v2, v3) - v3 is latest version

-> If somebody deleted my original object by mistake, for latest version delete marker label will be applied.

-> Version ID is always unqiue

-> versioning files we can download at any time

-> If you delete the original object, delete marker is applid on the latest versioning

-> If you want the object to be re-stored, delete the delete marker and your object is re-stored.

-> You can't download delete marker version, you can only delete it.

-> Once you have enabled the versioning, you can't disable it. You can only suspend it.

-> AWS charges for Versioning, becareful while you enable versioning for huge files.

-> S3 is unlimited storage

min obj size = 0 Bytes
Max obj size = 5 TB

We can have unlimited no.of 5TB objs in a single bucket

For single PUT, we can upload max 5 GB.

-> If we want to upload bigger file then we shud go for Multi_part upload (MPU)
(break files in smaller chunks and upload it)

-> AWS recommended, if you have >100 MB use MPU

###############
Storage Classes
###############

-> While uploading the objects into S3 , selecting storage class is mandatory

Scenario : some customers wants to store and wants to access data frequently...
some other customers wants to store data but don't want to access frequently... we
can't charge same bill for both customers because they have diff businss
requirements.

-> To meet business requirements of clients, AWS provided several storage classes
in S3

Standard Frequently Access(FA)

This is used for frequently access data

Default storage class
Regular purpose (storing, website, images etc)
No retrival charges
Availability = 99.9 %
Durability=999999999 (11 9's)
Min Obj size is 0 Bytes

Standard In-Frequently Access (IA)

Frequently access but not critical
Not Retrival charges
AWS doesn't recommend to use this
Cheaper than others
Availability = 99.9 %
Durability=99.99%
min obj size = 128 kb
min duration : 30 days

One Zone IA

In-frequently access but not critical

retrival charges apply
availability=99.99%
Durability= 11 9's
Min Obj size = 128 KB
min obj size = 128 kb
min duration : 30 days

```
Intelligent Tiering
------------------------
Unknown access pattern
Based on access it moves from FA to IA
availability=99.99%
Durability= 11 9's

min duration : 30 days

Glacier
-----------
Infrequently access data
archiving purpose
vault : container of archives
Archive : Object /Archive(zip) -> 40 TB
unlimited no.of archives
1000 vaults
Retrival charges apply


Glacier has retrival options
--------------------------------------
Expedited : 1 to 5 mins
Standard : 3 to 5 hours
Bulk : 5 to 12 hours
availability=99.99%
Durability= 11 9's
Min duration : 90 days

Note: Deep Glocier min duration is 180 days

-> It is possible to move the objects from one storage class to another storage
class automatically (LCM) -> Life Cycle management.


-> LCM is created on bucket level and applied on object level

-> Life cycle rule  (current version & previous version)

-> my obj moving from FA -> IA (30 days) -> Glacier(60 days) ->this is called
transition

-> Delete after 365 days (Expiration)

-> Object Lock (Permanent Lock & certain period lock)

-> I have a bucket named as movies.

-> We can enable bucket logs to identify who is accessing our bucket

-----------------------
Encryption
---------------------
Encryption is used for security

Encryption can be done in 2 ways

In-Transit : Encryption while data is moving/flow HTTPS

Data At Rest : Encryption while data is at rest


-> Amazon S3 has 3 types of encryptions

serer-side encryption
```

SSE - S3 (AWS Managed Key)
SSE - KMS (AWS KMS Key)
SSE - C (Customer Provided Key)

client-side encryption (should be handled by customer, how to reach aws is our headache)
in-transit encryption (using https)

-> AWS Certification manager (ACM)
is where you can generate HTTPS certificates (SSL/TLS/HTTPS)

-> S3 data consistency models - 2 types

Read after write consistency for PUTS of new objects (immediatley)
Eventually consistency for overwrites of puts and deletes

-> Pre-Signed URL (it will be accessible for limited time)

##########################
Transfer Acceleration
##########################

-> If we want to transfer the data from our place to AWS bucket it will use our own internet.

-> We can speed up transfer process using Transfer Acceleration.

-> It is used to transfer data fastly (It will use CDN concept)

-> With CDN it will use AWS internal network

---################
Route 53
################

-> Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service.

-> It is used to map application domain name with IP addr / LBR URL / S3 Bucket URL

```
                                                          Route 53
                        www.ashokit.in ===============> https://192.168.1.1/ (EC2
IP)

                                                                Route 53
                        www.facebook.com ===============>
https://facebook.com.alb.aws.amazon.com/ (LBR URL)

                                                                Route 53
                        www.ashokit.org  ================> S3 Bucket URL


                S3 Bucket URL :
http://ashokit.org.s3-website.ap-south-1.amazonaws.com/
```

-> Using Route 53 service we can map our application domain name to EC2 instance IP address

-> Using Route 53 service we can map our application domain name to Load Balancer URL

-> Using Route 53 service we can map our application domain name to S3 Bucket URL

Steps to configure DNS name using Route 53 service
----

1) Goto Route 53 service

2) Register Domain in AWS

Note: Based on domain extension charges will be applied

```
                        .com ===> 12 $

                        .link ===> 5 $

                        .click  ===> 3 $
```

3) Go to billing dashboard and complete payment for the domain you have purchased

Note: When we purchase domain we will get one hosted zone with Name servers by default

Note: Domain Name also will get Name Servers by default

Note: "Domain Name" Name Servers and "Hosted Zone" Name Servers should be same

Note: Name servers are used to route the traffic to Domain Name

4) Go to hosted zone and create a new record

```
                        Record Name : Leave it blank

                        Record Type : keep it default value (A - Route to ipv4 or
Aws resources)

                        Value : Select Alias
                                        Choose Endpoint as "Load Balancer"
                                        Choose  Region
                                        Choose  LBR
                                        Click on Create Record
```

Note: We can route the traffic to Specific IP / Load Balancer URL / S3 Endpoint etc..

Note: Here i am using Load Balancer Endpoint URL to route the traffic

5) Once Record is created wait for sometime (2 to 3 mins)

6) Access Domain Name URL in browser

Ex: www.ashokit.org

+++

---=================
Elastic Beanstalk
=================

-> End-to-end web application management service in AWS

-> In AWS, Elastic Beanstalk provides Platform As a Service (PaaS)

-> Easily we can run our web aplications on AWS cloud using Elastic Beanstalk service.

-> We just need to upload our project code to Elastic Beanstalk it will take care of deployment

-> Elastic Beanstack will take care of softwares and servers which are required to run our application.

-> Elastic Beanstalk will take care of deployment, capacity provisioning, load balancer and auto scaling etc..

-> To deploy one java web application manually we need to perform below operations

1) Create Security Group

2) Create Network

3) Create Virtual Machine (s)

4) Install Java software in Virtual machine

5) Install Webserver to run java web application

6) Deploy application to server

7) Re-start the server

8) Create LBR

9) Create AutoScaling etc...

-> AWS providing infrastructure, we are creating platform using AWS infrastructure to run our java application
   (IaaS Model)

=> Instead of we are preparing platform to run our application, we can use Elastic Beanstalk service to run our web applications.

=> Elastic Beanstalk is providing Platform as a service.

```
========================
Create Sample Application
=========================
```

-> Create Application

-> Choose the name

-> Select the platform

```
======================
Below Beanstalk events
======================
Env Creation started
S3 bucket created to store the code
```

Security Group
VPC
EC2 instances
Webserver installation
Load Balancer
Autoscaling
Cloud watch

URL : http://webapp1-env.eba-rc8b64vg.ap-south-1.elasticbeanstalk.com/

===============================
Advantages with Elastic Beanstalk
===============================

1) Fast and simple to begin

2) Developer productivity

3) Impossible to outgrow

4) Complete resource control

====================
Elastic Beanstalk Pricing
====================

Thereâ€™s no additional charge for Elastic Beanstalk. You pay for Amazon Web Services resources that we create to store and run your web application, like Amazon S3 buckets and Amazon EC2 instances.

==
Procedure to deploy java-spring-boot-application
==

1) Create one application in Beanstalk

2) Choose Platform as Java

3) Select Upload Your Code option and upload spring-boot-jar file

-> Go to our application environment -> Configuration -> Software -> Edit -> Environment properties -> Configure below property

```
                        SERVER_PORT = 5000
```

Note: After changing the port Environment will be re-started

URL : http://sbrestapp-env.eba-5fsapphh.ap-south-1.elasticbeanstalk.com/welcome

=====================
Application Versions
======================

-> In Beanstalk we can maintain mutliple versions of our application

```
                        sb-rest-api-v1.jar
                        sb-rest-api-v2.jar
```

sb-rest-api-v3.jar

-> We can deploy particular version of jar file based on demand

----

=================
Amazon Cloud Watch
=================

Amazon CloudWatch is a component of Amazon Web Services that provides monitoring for AWS resources and the customer applications running on the Amazon infrastructure.

CloudWatch enables real-time monitoring of AWS resources such as Amazon Elastic Compute Cloud (EC2) instances, Amazon Elastic Block Store (EBS) volumes, Elastic Load Balancing and Amazon Relational Database Service (RDS) instances.

The application automatically collects and provides metrics for CPU utilization, latency and request counts.

Users can also stipulate additional metrics to be monitored, such as memory usage, transaction volumes or error rate.

==========================
Cloud Watch & SNS - Lab Task
==========================

1) Create SNS Topic with Email Notification

2) Configure Email Subscription in SNS Topic (Confirm Subscription recieved in email)

3) Select EC2 Instance -> Action -> Monitor and trouble Shoot -> Manager Cloud Watch Alarms -> Create cloud watch Alarm

  Alaram Notification : Select SNS Topic which we have created
  Alaram Thresh Hold : Avg CPU >= 5%

4) Connect to EC2 instance

5) Increase load on the EC2 instance using "stress" software

```
  $ sudo amazon-linux-extras install epel -y
  $ sudo yum install stress -y
  $ sudo stress --cpu 8 -v --timeout 180s
```

6) Observe the behaviour of Cloud Watch / Alaram / SNS
    (We should get Email Notification)

Note: When Alarm got triggered, its status will be changed to 'In Alarm'

=> We can Monitor our Alarm history (how many times it got triggered)

        (Goto Cloud Watch -> Select Alarms -> Click Alarm -> Click on History)

=======================
VPC : Virtual Private Cloud
======================

-> We are creating Virtual Machines in AWS to setup our infrastructure

-> In order to create Virtual Machine mainley we will use 3 services

1) EC2 (Elastic Compute Cloud)
2) EBS (Elastic Block Storage)
3) VPC (Virtual Private Cloud)

Note: Without using these services we can't launch Virtual VM

-> In AWS we have total 200+ Services

-> Those 200 services we can categorize them into 2 types

1) AWS Managed Services (SaaS)
2) Customer Managed Services (IaaS & PaaS)

Note: VPC is one of the customer managed service in AWS

Example

Take a 2 BHK flat
(Land, House (BR-1, BR-2, Kitchen, Hall))

Land Gate (To enter into our land)
Hall Gate (To enter into our house)
Kitchen Gate, BR-1 gate and BR-2 gate

-> Consider VPC also like above house

-> For every room seperate door (In VPC seperate network (Subnets))

-> We need main gate to enter into house (In VPC we will use Internet Gateway)

-> To enter into hall we will use hall gate (In VPC we will use Routing Table)

-> People will come into our house and sit in hall for discussion then they will leave
    (we will be monitoring manually)

Note: To avoid public access we will keep our components in private subnets

(Webservers, Appservers, DB Servers)

-> Public subnet means Internet connection will be there

-> Private subnet means no internet connection

-> If our components are in private subnet then how to take code or update code ?

Note: NAT gateway comes into picture (it will be created in public subnet)

-> Our public and private subnets will be connected to NAT Gateway

===
Q) What is the difference between Ineternet Gateway and NATGateway ?
===
-> IG will have both incoming and outgoing

-> Nategateway will have only outgoing

Note: Outside ppl can't access NAT Gateway

VPC
Subnets (private & public)
Internet Gateway (Ingres & Agress)
Route Table (Connections to Subnets)
NAT Gateways (no incoming only outgoing)

###########################
Then how to provide security ?
###########################

-> In houses manually we will be monitoring who is entering into our house and who is entering into rooms

-> In VPC we have to configure security rules

-> In VPC to provide security we have 2 options

1) Security Group
2) NACL (N/W Access Control List)

Note: Who can access which subnet we will configure in NACL

-> Every subnet will have its own NACL (both private and public)
-> NACL will work at subnet level

-> Security group is used to open ports and who can access those ports
-> Security groups works at resource level

-> If we want to apply same rule for multiple resources then we will go for security group and will add that for all resources

-> If we want to apply security policy for entire subnet then we will use NACL

-> In security group we can only allow the traffic
-> In NACL we can do both allow and deny the traffic

Note : For our aprtment we will have security to notedown incoming and outgoing ppl/vehicles recrods

-> If we want same functionality for our VPC then we can use VPC FLOW Logs
-> VPC Flow Logs will track incoming and outgoing traffic of our VPC
-> By default VPC is disabled (we can enable)

Note: Check default VPC configurations Route Tables and NACL

-> To construct house we will take land measurements (length & width)

-> How to take measurements in VPC Networking ?

-> We will do VPC sizing based on IPS

Example

If we take 100 ip's then in VPC we will allocate those 100 ip's to 2 subnets

Subnet-1 : 50 ips
Subnet-2 : 50 ips

-> All my 100 ips got completed. In future if i get new requirement i m running out of ips

-> To overcome these problems we don't allocate all ips to subnets
   (we will allocate few ips and will keep few ips in reserve mode)

-> Before entering into ip sizing lets understand IPs first

Note: We can create VPC is very easily but we will get multiple errors while creating subnets and ip allocation.

Note: there are several websites will help us in ip division

-> Lets understand ips

-> I have 2 subnets connected with LAN

```
                                private ip
        subnet-1 <------------------------> subnet-2
```

-> If 2 systems not connected with LAN then we should go for public ip

```
                                 public ip
        System-1 <------------------------> System-2
```

-> IPs are divided into 2 types

1) IPV4
2) IPV6

Note: Earlier we have only IPV4

-> In IPV4 we have 5 classes
        (A, B, C, D, E)

Note: Class E is reserved for researching purpose

-> IP ranges we will do with CIDR

-> CIDR stands for Class less inter domain range

Note: These CIDR classes are required for only public ips and not required for private ips

-> Daily billions of new devices launching and they are using internet

-> If a device wants to use internet then ip is mandatory (we are running out of ips)

-> To overcome this 1PV4 problem IPV6 came into picture

IPV4
++++
10 . 0 . 0 . 1
32-Bits
We have 4 parts here
Every part is called as one octect
In every octect we have 8 bits
4 parts * 8 bits = 32 bits

IPV6
++++

```
10.4.1 -  -  -
128 bits
We have 8 parts
In evary part we have 16 bits
8 parts * 16 bits = 128 bits

==================
IP ranges in IPV4
==================
-> IP ranges will be calculated in 2 power

-> If we give range as /16

10.0.0.1/16 = > 2 power (32-16) => 2 Power 16 => 65536
10.0.0.1/32 = > 2 power (32-32) => 2 Power 0  => 1
10.0.0.1/31 = > 2 power (32-31) => 2 Power 1  => 2
10.0.0.1/30 = > 2 power (32-30) => 2 Power 2  => 4
10.0.0.1/29 = > 2 power (32-29) => 2 Power 3  => 8
10.0.0.1/28 = > 2 power (32-28) => 2 Power 4  => 16 (AWS supports from /28)

=> /29  to /32 AWS won't support

-> We can start giving subnet ranges from /16 to /28

-> Recommended to use /24

        10.0.0.1/24 = > 2 power (32-24) => 2 Power 8 => 256

Note: /24 we will get 256 IPs those are sufficient for our usecases in realtime


#################
VPC Lab Task
#################

-> Create a VPC

-> CIDR Block : 10.0.0.0/16

-> Select No IPV6 CIDR blok

-> Select Default Tenancy

-> Create VPC

Note: After creating VPC verify its details

(DNS hostnames -> Disabled)

#############
Create Subnets
################

Note:  By default 3 subnets will be available under default vpc

-----------------------
Create Subnet-1
----------------------
-> Create Subnet

   Name : public-subnet-az-1

-> select availability zone as 1a

-> CIDR Block : 10.0.0.0/24  (It will take 256 ips)
```

```
------------------------
Create Subnet-2
-------------------------
-> Create Subnet

   Name : private-subnet-az-1b

-> select availability zone as 1b

-> CIDR Block : 10.0.1.0/24  (It will take 256 ips)


Note: Every subnet will have Route Table and NACL

-> AWS will reserve 5 ips in every subnet (we will get only 251)

---------------------------------
Create Internet gateway
---------------------------------
Note: By default one IGW will be available and it will be attached to default VPC

-> Create custom Internet Gateway (ashokit-vpc-igw)

-> Attach this IGW to VPC (we can attach IGW to only one VPC)

----------------------------
Create Route Table
----------------------------

Note: When we create VPC, we will get only route table by default. It is called as
Main route table.

Note : Change existing route table name  'ashokit-vpc-private-rt'

-> Create one new Route Table (ashokit-vpc-public-rt)
-> Choose vpc and create it

Now We have 2 route tables

-> Goto route table and attach route tables to subnets  (Subnets association for
Route Tables)

Private Route Table should have Private Subnet
Public Route Table should have Public Subnet


++++++++++++++++++++++
Making Subnet as public
++++++++++++++++++++++

-> Goto public Route Table -> Edit Routes

-> Add Destination as 0.0.0.0/0 and Target as IGW -> Save

-> Subnet Associations -> Edit SNET -> Select Public Subnet

------------------------------------
Create EC2 (Public EC2)
--------------------------------------
-> Choose AMI
-> Select VPC
-> Select Public Subnet
-> Assign Public IP as Enable
-> Add SSH and Http Protocols
```

-> Download KeyPair
-> Launch Instance

Note: Goto VPC and Enable DNS Host Enable

Create EC2 (Private EC2)

-> Choose AMI
-> Select VPC
-> Select Private Subnet
-> Assign Public IP as Enable
-> Add SSH (source : custom, Range : 10.0.0.0/16)
-> Download KeyPair
-> Launch Instance

====================
Test EC2 Connections
====================

-> Connect to Public EC2 using MobaXterm (It should allow to connect)

-> Connect to Private EC2 using MobaXterm (It shouldn't allow to connect)

==
Connect with 'private-ec2' from 'public-ec2' using 'ssh' connection
==

Note: As both Ec2 instances are available under same VPC, we should be able to one machine from another machine.

Procedure to access

-> Upload pem file into public-ec2 machine (in mobaxterm we have upload option)

-> Execute below command to provide permission to access pem file

```
                $ chmod 400 <pem-file-name>
```

-> Execute below command to make ssh connection from public-ec2 to private-ec2

```
                $ ssh -i "pem-file-name"  ec2-user@private-ec2-vm-private-ip
```

Note: It should establish connection (this is internal connection)

-> Try to ping google from private ec2 (it should not allow because igw is not available)

=============================
VPC with NAT Gateway Lab Task
=============================

1) Create NAT gateway in public subnet

2) Add NAT gateway in 'private-subnet-routute-table'

3) After NAT Gateway, we should be able to ping google from 'private-ec2' also

===

Note: Delete Elastic IP and NAT Gateway after practise

===

=======================
What is VPC Peering
=======================

VPC Peering: IPV4 or IPV6 traffic routes between VPCs created to establish communication between one or more multiple VPCs.

=======================
AWS definition:
=======================
=> â€œA VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses.

=> Instances in either VPC can communicate with each other as if they are within the same network. â€œ

1) Through VPC Peering, traffic stays within the AWS network and not go over the internet.

2) Non-overlapping CIDRs â€“ The 2 VPCs you are trying to peer, must have a mutually exclusive set of IP ranges.

3) Transitive VPC Peering â€“ not allowed i.e

(If VPC A & B have peered and VPC A & C have peered, VPC B & C cannot share contents until there is an exclusive peering done between VPC B & C)

=======================
Will VPC Peering Cost me?
=======================
No. VPC itself wonâ€™t cost you, however, the resources deployed inside the VPC and data transfers are done will cost you.

==
Letâ€™s create VPC Peering to enable communication
==

To establish the connection, lets create VPC peering

=> On the left navigation panel under VPC -> Peering Connections:

VPC (Requester) = ashokit_aws_custom_vpc

VPC (Accepter) = default_vpc

=> Now you would see the status Pending Acceptance which means, Requestor has sent a request to the peer now target VPC needs to accept the request.

=> Go to VPC Peering -> Click on Actions -> Accept Request

=> Now we need to make entries in Route Tables

Now navigate to Route Tables, in Default VPC RT(Route Table) -> Edit routes

########## Default VPC Route Table should have 3 routes #########
(Local + all traffic with IGW + ashokit_aws_custom_vpc IP Range)

```
172.31.0.0/16 - local
0.0.0.0/0 - Internet-gateway
10.0.0.0/16 - vpc peering  (We need to add this)

########### Custom VPC Route Table should have 3 routes #########
(Local + All traffic with IGW + Default VPC IP Range)

10.0.0.0/16 - local
0.0.0.0/0 - Internet-gateway
172.31.0.0/16 - vpc  (We need to add this)

########### Allow Traffic in VPC Security Groups ###########

Edit Security Group of Default and Custom VPC to allow traffic from each other

Default VPC Security Group looks like
SSH - 22 - all
HTTP - 80 - all

Custom VPC Security Group would look like
SSH - 22 - all
HTTP - 80 - all

############ Create EC2 instance under Customer VPC with below user data script
###########

sudo yum update -y
sudo yum install -y httpd.x86_64
sudo  systemctl start httpd.service
sudo  systemctl enable httpd.service
echo "Hello world from $(hostname -f)" > /var/www/html/index.html

############ Create EC2 instance under Default VPC with below user data script
###########

sudo yum update -y
sudo yum install -y httpd.x86_64
sudo  systemctl start httpd.service
sudo  systemctl enable httpd.service
echo "Hello world from $(hostname -f)" > /var/www/html/index.html


==========================Test VPC Peering Connectivity ==========================

# Access Webserver that is available in Custom VPC from Default-vpc-ec2-instance
with below command
$ curl <private-ip>

Note: We should get response

# Access Webserver that is available in Default VPC from Custom-vpc-ec2-instance
with below command
$ curl <private-ip>

Note: We should get response


======================================================
Q ) What is the difference between NACL and Security Groups ?
======================================================

================
Security Group
================
```

-> Security Group acts as a Firewall to secure our EC2 instances

-> Security Group contains Inbound Rules & Outbound Rules

```
                        inbound rules ---> incoming traffic
                        outbound rules ---> outgoing traffic
```

-> In One security group we can add 50 Rules

-> Security Group supports only Allow rules (by default all rules are denied)

-> We can't configure deny rule in security group

Ex : 172.32.31.90 ----> don't accept request from this IP (we can't do this in SG)

-> Security Groups are applicable at the instance level (manually we have to attach SG to instance)

-> Multiple Security Groups can be attached to single instance & one instance can have 5 security groups

-> Security Groups are statefull
(Any changes applied to incoming rules then those changes will be applicable for Outgoing Rules also)

-> In Security Group we can configure Rule destination as CIDR and IP

-> Security Group acts as First Level of defense for Outgoing traffic

======
NACL
======

-> NACL stands for Network Access Control List

-> NACL acts as a firewall for our Subnets in VPC

-> NACL applicable at the subnet level

-> NACL rules are applicable for all the resources which are part of that Subnet

-> NACL rules are stateless
(any changes applied to incoming rules will not be applicable for outgoing rules, we need to do that manually)

-> In NACL we can configure both Allow & Deny rules

Ex: We can block particual IP address (192.168.2.4) to connect with EC2 instance

-> One subnet can have only one NACL

Note: One NACL can be added to multiple subnets

-> NACL supports rule destination as only CIDR

-> NACL acts as first level of Defense for Incoming Traffic (Security Group acts as First Level of defense for Outgoing traffic)

--=================
Terraform (IAAC s/w)
=================

-> Terraform is an open source s/w created by HashiCorp and written in Go programming language

-> Terraform is an infrastructure as code (IaaC) software tool,

-> Infrastructure as code is the process of managing infrastructure in a file or files rather than manually configuring resources using user interface (UI)

-> In Terraform resources are nothing but Virtual machines, Elastic IPs, Security Groups, Network interfaces, RDS, LBR etc..

-> Terraform code is written in the HashiCorp Configuration langauge (HCL) in files with the extension .tf

-> Terraform allows users to use HashiCorp Configuration Language (HCL) to create the files containing definitions of the their desired resources

-> Terraform Supports all most all cloud providers (AWS, AZURE, GCP, Openstack etc..)

-> To automate infrastructure creation in cloud platforms we will use Terraform.

==========================
Terraform vs Cloud Formation
==========================

-> Terraform developed by HashiCorp
-> CloudFormation developed by AWS

-> Terraform supports many cloud providers
-> Cloud Formation willl support only in AWS

-> Terraform uses HashiCorp configuration language (HCL) which built by HashiCorp. It is fully compatible with JSON.

-> AWS Cloud Formation utilizes either JSON or YAML. Cloud formation has a limit of 51,000 bytes for the template body itself.

==========================
Terraform Vs Ansible
==========================

-> Terraform developed by HashiCorp
-> Ansible is also an open source software

-> Terraform is an infrastructure as a Code, which means they are designed to provision the servers themselves.
-> Ansible is a configuration management tool. Which means ansibled designed o install and manage software on existing servers.

-> Terraform is ideal for creating, managing and improving infrastructure.
-> Ansible is ideal for software provisioning, application deployment and configuration management.

==========================
Terraform Setup - Pre-Requisites
==========================

1) Cloud Platform Account (AWS, Azure, GCP, Openstack etc..)
2) IAM User account (Secret Key and Access Key)
3) IAM User should have resources Access

###############################

```
Terraform Installation
#############################

1) Create EC2 instance (RED HAT Linux)

2) Connect to EC2 VM using Mobaxterm

3) Swith to root user
        $ sudo su -

4) Install unzip software
        $ sudo yum install wget unzip vim -y

5) Download Terraform Software (https://www.terraform.io/downloads)

$ sudo yum install -y yum-utils
$ sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
$ sudo yum -y install terraform

6) Check Terraform Version

$ terraform -v


#######################################
Working with EC2 Instance using Terraform
########################################


1) Create IAM user with Programmatic Access   (IAM user should have EC2FullAccess)

2) Download Secret Key and Access Key

3) Write First Terraform Script

$ mkdir terraformscript
$ cd terraformscripts
$ vi FirstTFScript.tf

provider "aws" {
   region = "ap-south-1"
   access_key = "AKIA4MGQ5UW757KVKECC"
   secret_key = "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"
}

resource "aws_instance" "AWSServer" {
  ami = "ami-08df646e18b182346"
  instance_type = "t2.micro"
  key_name = "linux"
  security_groups = ["ashokit_security_group"]
  tags = {
   Name = "MyEC2-VM"
  }
}

10) Initialize Terraform using init command

        $ terraform init

11) Format your script (indent spaces)

        $ terraform fmt

12) Validate Your Script
```

```
        $ terraform validate
```

13) Create Execution Plan For Your Script

```
        $ terraform plan
```

14) Create Infrastructure

```
        $ terraform apply
```

Note: When the script got executed it will store that state in a file. If we execute script again it will not create. If you delete that state file and execute script again then it will create it.

15) Destory Infrastructure

```
        $ terraform destroy -auto-approve
```

-> In first script we kept provider and resources info in single script file. We can keep provider and resources information in seperate files

Ex : proder.tf & main.tf

```
###################################
Script to create multiple Ec2 instances
###################################

provider "aws" {
   region = "ap-south-1"
   access_key = "AKIA4MGQ5UW757KVKECC"
   secret_key = "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"
}

resource "aws_instance" "AWSVM_Server" {
  count           = "2"
  ami             = "ami-05c8ca4485f8b138a"
  instance_type   = "t2.micro"
  key_name        = "linux"
  security_groups = ["ashokit_security_group"]
  tags = {
    Name = "REDHAT-EC2-VM"
  }
}
```

Note: Once it is created, then destory infrastructure using below command

```
$ terraform destroy -auto-approve
```

=====================
Variables in TypeScript
=====================

-> We can maintain variables in seperate file

```
$ vi vars.tf

variable "ami"{
 description="Amazon Machine Image value"
 default = "ami-05c8ca4485f8b138a"
}
```

```
variable "instance_type"{
   description="Amazon Instance Type"
   default = "t2.micro"
}

variable "instances_count"{
        description="Total No.of Instances"
        default = "2"
}
```

-> Create main tf file using variables

```
$ vi main.tf

provider "aws" {
  region = "ap-south-1"
   access_key = "AKIA4MGQ5UW757KVKECC"
   secret_key = "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"
}

resource "aws_instance" "AWSServer" {
  count="${var.instances_count}"
  ami = "${var.ami}"
  instance_type = "${var.instance_type}"
  key_name = "linux"
  security_groups = ["ashokit_security_group"]
  tags = {
   Name = "EC2 VM - ${count.index}"
  }
}
```

Note: We can supply variables in runtime also

-> Remove instances_count variable from var.tf file and pass like below

```
$ terraform apply -var instances_count="2" -auto-aprove
```

=============================
Comments in Terraform Script
=============================

# - single line comment

// - single line comment (java style)

/* and */ - Multi line comments

================================
Dealing with Secret Key and Access Key
================================

-> We have configure secret_key and access_key in terraform script file. Instead of that we can configure them as environment variables.

```
$ export AWS_ACCESS_KEY_ID="AKIA4MGQ5UW7UPMSQKXK"
$ export AWS_SECRET_ACCESS_KEY="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"
```

-> To verify environment variables we can use echo command

```
$ echo $AWS_ACCESS_KEY_ID
$ echo $AWS_SECRET_ACCESS_KEY
```

-> Now remove credentials from terraform script and execute it.

Note: We are setting provider credentials in terminal so these variables will be available for current session. If we want to set permanently add them in .bashrc file

```
============================
Working with User Data
============================
```

-> It is used to execute script when instance launched for first time.

-> Create Userdata in one file

```
$ vi installHttpd.sh

#!/bin/bash
sudo su
yum install httpd -y
cd /var/www/html
echo "<html><h1>Welcome to Ashok IT...!!</h1></html>" > index.html
service httpd start

$ chmod u+x installHttpd.sh
```

-> create main scrit in main.tf file

-> vi main.tf

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "AWSServer" {
  ami = "ami-05c8ca4485f8b138a"
  instance_type = "t2.micro"
  key_name = "linux"
  security_groups = ["ashokit_security_group"]
  user_data = "${file("installHttpd.sh")}"
  tags = {
   Name = "Web-Server"
  }
}
```

```
==================================
Creating S3 bucket using Terraform script
==================================
```

-> Add S3 policy for IAM user

-> Execute below terraform script to create s3 bucket in AWS

```
provider "aws"{
  region = "ap-south-1"
}

resource "aws_s3_bucket" "s3bucketashokit"{

bucket = "s3bucketashokit"
acl="private"

versioning{
        enabled = true
}

 tags = {
   Name = "S3 Bucket By Ashok"
 }
}
```

===================================
Create MySQL DB in AWS using Terraform
===================================

-> Provider RDS access for IAM user

-> Execute below script to create MySQL DB in AWS cloud

```
provider "aws"{
  region = "ap-south-1"
}

resource "aws_db_instance" "default" {
  allocated_storage    = 100
  engine                    = "mysql"
  engine_version        = "5.7"
  instance_class        = "db.t3.micro"
  name                        = "mydb"
  username                = "foo"
  password                = "foobarbaz"
  parameter_group_name = "default.mysql5.7"
  skip_final_snapshot  = true
}
```

---======================
Elastic File System (EFS)
======================

-> AWS EFS lets you create scalable file storage to be used on EC2.

-> You donâ€™t have to bother about capacity forecasting as it can scale up or down on-demand.

===============================
Some of the EFS advantages are:
===============================

1) Fully managed by AWS
2) Low cost, pay for what you use
3) Highly available & durable service
4) Automatically scale up or down
5) Scalable performance

========================
Steps to work with EFS
========================
=> Login into the AWS console
=> Go to Services and select EFS under storage
=> Click on â€œCreate file system.â€

File System ID : fs-00fc4e2c6aee19e2e

```
=======================
Mounting EFS on EC2
======================

Before mounting, you need to install the NFS client. If you expand the list and
click on â€œAmazon EC2 mount instructionsâ€, you will get the details.


# Login to EC2 instance and install the NFS client

$ sudo yum install -y amazon-efs-utils

# Letâ€™s create a folder where you want to mount the EFS.

$ sudo mkdir efsdir

#  Mount EFS Filesystem (Make sure you changed FileSystem ID)
$ sudo mount -t efs -o tls fs-0a37cdf9d7d2d8f82:/ efsdir

Note: Make EC2 VM security Group inbound rule having "NFS" protocol with 2049 port
number

# Change the directory to the mount point that is created above using the command:

$ cd efsdir

# Create a new sub directory with following command:

$ sudo mkdir begin

# Change the permissions of the above subdirectory with the following command:

# sudo chown ec2-user begin

# Change the directory to begin directory with following command:

$ cd begin

# Create a sample text file:

$ touch myfile.txt

# Run ls command to list the contents of directory.

============================================================
Note: Create another ec2 instance and mount EFS file system
============================================================

# Login to EC2 instance and install the NFS client

$ sudo yum install -y amazon-efs-utils

# Letâ€™s create a folder where you want to mount the EFS.

$ sudo mkdir efsdir

# Mount EFS Filesystem (Make sure you changed FileSystem ID)
$ sudo mount -t efs -o tls fs-00fc4e2c6aee19e2e:/ efsdir

# Change the directory to the mount point that is created above using the command:

$ cd efsdir

# check the files available
```

```
$ ls
```

Note : The files we have created in First EC2 instance, should display in second ec2 instance.

========================
Serverless Computing
========================

-> Serverless computing means run the application without thinking about servers

-> AWS will take care of servers required to run our application

-> AWS lambdas are used to implement serverless computing

=============
AWS Lambdas
=============
AWS Lambda is a way to run code without creating, managing, or paying for servers.

You supply AWS with the code required to run your function, and you pay for the time AWS runs it, and nothing more.

Your code can access any other AWS service, or it can run on its own. While some rules about how long a function has to respond to a request, thereâ€™s almost no limit to what your Lambda can do.

The real power, though, comes from the scalability that Lambda offers you.

AWS will scale your code for you, depending on the number of requests it receives. Not having to build and pay for servers is nice. Not having to build and pay for them when your application suddenly goes viral can mean the difference between survival and virtual death.

=====================
Should I Use It?
=====================
While AWS Lambda and serverless architecture have some really cool benefits, there are caveats. State management in a serverless architecture requires a bit of a mindset shift. Thereâ€™s no local state, and traditional database connections are not really applicable to short-lived functions.

Because Lambdas are not persistent, you cannot use connection pooling from the Lambda. If you try to use traditional database connections and AWS spins up 20 lambdas to service 20 requests that come in during a short window, youâ€™re likely to run into data access issues.

Amazon wants to help you solve this data access problem by providing additional services such as DynamoDB or Aurora. This is great to help you solve the connection pooling issue. However, one price youâ€™ll pay for this is vendor lock-in. The more you rely on AWS to manage your infrastructure, the harder itâ€™s going to be to decide to move away from that later. That might be fine for your situation, but itâ€™s good to keep in mind.

Lambdas have a hard limit on execution time. After 15 minutes, your function will be stopped whether itâ€™s finished or not. If this is a problem, then AWS Lambdas and functions-as-a-service arenâ€™t a good fit for your application.

This limit comes with a few others: AWS doesnâ€™t allow more than 512MB of disk space for your functions and the invocation payload (request and response) is limited to 256 KB (asynchronous) and 6 MB (synchronous). Here again, these limits have to do with what AWS Lambdas are intended for; small discrete computing jobs.

Another potential issue is the cold start.

Lambdas can be really cheap because you only pay for what you use, but that means that when the lambda is finally triggered, there will be some overhead to turn it on and service the event. Itâ€™s not a significant amount of time, on the order of seconds. Your lambdas will persist for a period of time after the initial cold start, so the cost is not felt on every request.

=======================================
AWS Lambda vs. AWS EC2 vs. Elastic Beanstalk
==

AWS Lambda is not the only computing service that AWS provides. So how is it different from other services like EC2 and Beanstalk?

Lambda is a Platform as a Service(PaaS) whereas EC2 is an Infrastructure as service(IaaS). EC2 is more flexible and customizable when compared to Lambda. If youâ€™re using EC2, you can choose Operating Systems, Networks, and also customize them. You can also change network configurations between different EC2 instances or between an instance or a VPC and the internet. But using EC2 is like setting up a server and lambda leans more towards serverless architecture.

Beanstalk is a Platform as a Service(PaaS) to deploy and manage applications on the cloud. Beanstalk provides default provisioning, load balancing, and other features and you just need to enable them to use them for your application. So, most of the things are taken care of by AWS and you focus on building, testing, and deploying the application. The main difference between Beanstalk and Lambda is that with Beanstalk, you put the whole application on the cloud, but when using Lambda, only the functional part of the code goes on the cloud.

The bottom line is that all these computing services have their own pros and cons, and are specialized for different purposes. Which suits you best depends on your use case. Among the 3, AWS Lambda is the most suitable and cheap to run back-end codes without the need to set up a server.

So it looks like there are some things to consider regarding data and state management, vendor lock-in, and performance. That being said, you shouldnâ€™t be scared of using lambdas, just be aware of what youâ€™re buying. Now, letâ€™s move on to making your first AWS Lambda!

==================================
Running Java Code with AWS Lambda
==================================

1) Create Lambda Function with 'java 11' runtime

2) Upload jar file in 'Code Source'

3) Configure Handler in Runtime

Class Name : in.ashokit.LambdaHandler

Method Name : handleRequest

Handler Syntax : className :: methodName

Ex: in.ashokit.LambdaHandler::handleRequest

```
===============
Cloud Formation
===============
```

=> Cloud Formation service is used to provision infrastructure in AWS Cloud.

=> Cloud Formation works based on 'Infrastructure as a code' (IAAC)

=> Cloud Formation supports JSON and YML configurations

Note: If we create infrastructure manually it takes lot of time and it is error prone.

=> If we design cloud formation template to create Infrasture then we can re-use that template.

Note: Cloud Formation service works only in AWS Cloud.

Note: The alternate for 'Cloud Formation' service is 'TERRAFORM' tool.   Terraform works with almost all cloud platforms available in the market.

```
=====================================
Creating EC2 instance using Cloud Formation
=====================================
```

=> Goto AWS Management Console and Navigate to 'Cloud Formation'

=> Save below template file with .yml extension

=> Click on Create Stack and upload below Template File

------------------------------------ Ec2 - creation - yml - file -------------------------

```
Description:  Ashok IT - Build Linux Web Server
Parameters:
  LatestAmiId:
    Description: AMI Linux EC2
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'
Resources:
  webserver1:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: "t2.micro"
      ImageId: !Ref LatestAmiId
      SecurityGroupIds:
        - !Ref WebserverSecurityGroup
      Tags:
        - Key: Name
          Value: webserver1
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum update -y
          yum install httpd -y
          service httpd start
          chkconfig httpd on
          cd /var/www/html
          echo "<br>" >> index.html
          echo "<h2><b>Ashok IT EC2 Linux Demo</b></h2>" >>index.html
  WebserverSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable Port 80
      Tags:
      - Key: Name
        Value: webserver-sg
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
```

=> Verify EC2 dashboard, we can see EC2 instance created

=> Access EC2 VM public in browser.

===================
Part-1 : Introduction
===================
1) What is DevOps
2) Why DevOps
3) DevOps Enginner Roles & Responsibilities
4) Developers Roles & Responsibilities
5) DevOps Tools Overview
6) What is SDLC
7) Waterfall Methodology
8) Agile Methodology

==================
Part-2 : Linux OS
==================
1) What is Linux & History
2) Linux Distrubutions
3) Linux VM Setup in AWS
4) Connecting with Linux VM
5) Linux Commands
6) Working with files & directories
7) Working with Editors
8) File Permissions
9) User Management
10) find & update
11) Software Installations

====================
Part-3 : DevOps Tools
====================
1) Maven
2) Gradle
3) Git Hub
4) BitBucket
5) Apache Tomcat
6) SonarQube
7) Nexus Repo
8) Terraform
9) Ansible
10) Docker
11) Kubernetes (Kubeadm + EKS Cluster)
12) JIRA

==================
Part-4 : AWS Cloud
==================
1) What is Cloud Computing
2) On Prem Vs Cloud Infra
3) Advantages with Cloud
4) AWS Introduction
5) AWS Services Overview
6) AWS Free Tier Account Creation
7) Region & AZ's
8) EC2
9) EBS Volumes + Snapshots
10) LBR + ASG
11) S3 Buckets
12) Static Website Hosting using S3
13) Route 53 (DNS Mapping)
14) IAM (Groups + Users + Policies)
15) Cloud Watch + SNS
16) RDS
17) EKS Cluster
18) AWS Lambads
19) Elastic Beanstack
20) VPC
21) EFS

================
Part-5 : Projects
================

1) Project-1 (CI CD Pipeline) : GIT + Maven + Sonar + Nexus + Tomcat + Jenkins

2) Project-2 (Fullstack App) : Docker + Kubernetes

Frontend : Angular

Backend : Spring Boot

3) Project-3 : Final CI CD Pipline : Jenkins + Docker + K8S (Pending)

=====================
Part-6 : Interview Guide
=====================

1) Resume Building Workshop

2) How to explain self introduction

3) How to explain Roles & Responsibilities

4) How to explain project in interview

5) 130 Interview Questions on DevOps

6) 100 Interview Questions on AWS

===
==