Agenda:-
--------
-->Problems before docker
-->why we need docker
-->vertualiztion vs Containerization
-->docker installation, git installation, maven installation
-->basic docker commands
-->what is dockerfile?, how to create dockerfile?, what is dockerimage?, how to create dockerimage?
-->what is dockerdsl?, dockerdsl keywords
-->what is dockerrepo? how to push the dockerimage into dockerrepo?
-->dockerizing springboot application

++++++++++++++++++++++
Problems before docker
++++++++++++++++++++++

-->Compatibility of each service with the underlying OS

-->Compatibility of each service with the libraries and dependencies of OS (One service requires versionX of OS lib rary. Another service - versionY of same library)

-->Every time version of any service updates, you might need to recheck compatibilities with underlying OS infrastr ucture

-->For a new developer to setup the environment with right OS and Service versions

+++++++++++++++++++
Advantages of Docker
+++++++++++++++++++

1.Portable
2.Consume less memory
3.Free and opensource
4.no environmental setup require

+++++++++++++++++++++++
Application Development
+++++++++++++++++++++++

-> Collection of programs is called as software project

-> Software project contains several components

1) Front end components (User interface logic)

2) Backend components (Business Logic)

3) Database Components (Persistence Logic)

-> In order to deploy our application in a machine we need to setup all the Softwares which are required to our appli cation

Ex: OS, Java 1.8v, MYSQL DB, Tomcat Web Server 9.0v etc.....

-> In Realtime project should be deployed into multiple environments for testing purpose

Ex : DEV, SIT, UAT, PILOT and PROD

-> DEV env will be used by Developers to perform integration testing

-> SIT env will be used by Testing team to test functionality of the application

-> UAT env will be used by Client to test functionality of the application

-> PILOT env means pre-production testing env

-> PROD means live environment (It is used to deliver the project)

-> To deploy application to these many enivornments we need to take of all the softwars required to run our applicati
on in all environments. It is very difficult task.

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

++++++++Install Docker in Amazon Linux++++++++++++

*take amazon linux machine.

```
$ sudo yum update -y
$ sudo yum install docker -y
$ sudo service docker start

# add user to docker group by executing below command
$ sudo usermod -aG docker ec2-user

$ docker info

#Restart the session
$ exit
```

+++++++++++++++install git++++++++++++++++++++++++

```
$ sudo yum update
$ sudo yum install git
```

+++++++++++++++++install maven+++++++++++++++++++++

```
$ sudo yum update
$ sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/
epel-apache-maven.repo
$ sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
$ sudo yum install -y apache-maven
```

++++++++++++++++++++Docker Commands+++++++++++++++++++

```
#see docker info
$ docker info

# To see docker images execute below command
$ docker images

# Pulling hello-world docker image
$ docker pull hello-world

# see docker image
$ docker images

# Running hello-world docker image
```

```
$ docker run hello-world
```

############### Note: Create account in Docker Hub (https://hub.docker.com/) ###################

Dockerfile
++++++++++++
Dockerfile is file which contains instructions to create an image. Which contains
Docker Domain Specific Key Words to build image.

DockerImage
++++++++++++

It's a package which contains everything(Softwares+ENV+Application Code) to run your application.

DockerContainer
+++++++++++++++++++++
Run time instance of an image.If you run docker image container will be created that's where our application(process
) is running.

DockerRepo/Registry
++++++++++++++++++++
We can store and share the docker images.

Public Repo
++++++++++++++
Docker hub is a public reposotiry. Which contains all the open source softwares as
a docker images. We can think of docker hub as play store for docker images.

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

USER

WORKDIR

EXPOSE

FROM
++++++++
FROM : It indicates base image to run our application. On top of base image we will create our own image

Syntax : FROM <IMAGE-NAME>

Example :

```
FROM java:jdk-1.8.0
FROM tomcat:9.2
FROM mysql
```

MAINTAINER
+++++++++++
-> It represents who is author or Dockerfile

Ex :   MAINTAINER Saleem <saleem@gmail.com>

COPY
+++++
-> It is used to copy files / folders to image while creating an image

Syntax :    COPY <source> <destination>

Example :

```
# copying war file from target directory to tomcat/webapps directory

COPY target/maven-web-app.war  /usr/local/tomcat/webapp/maven-web-app.war
```

ADD
++++++

-> ADD is also used to copy files to image while creating an image

-> ADD keyword can download files from remote location (http)

-> ADD keyword will extract tar file while copying to image

Note: zip files we have to extract manually

Syntax :

ADD <source> <destination>

ADD <url-to-download> <destination>

Q) What is the difference between COPY and ADD ?

RUN
+++++

-> It is used to execute commands on top of base image

-> Run command instructions will execute while creating an image

-> We can write multiple RUN instructions, they will execute in the order (from top to bottom)

Example :

RUN mkdir  workspace

RUN yum install git

CMD
++++

-> CMD is also used to execute commands

-> CMD instructions will execute while creating container

Example :

CMD sudo start tomcat

-> We can write multiple CMD instructions in Dockerfile but Docker will process only last CMD instruction.

Note: There is no use of writing multiple CMD instructions in Dockerfile

Sample Dockerfile
++++++++++++++++
FROM ubuntu

MAINTAINER  Ashok IT

RUN echo "Run One"

RUN echo "Run Two"

```
CMD echo "CMD One"

CMD echo "CMD Two"

RUN echo "Run Three"
```

```
# build image using docker file
$ docker build -t imageone .
```

syntax:- docker build -t <image-name> .
 where . (current directory)

```
# Run image
$ docker run imageone
```

Note: CMD instruction we can override using runtime CMD

```
#It will print only date (CMD will not execute)
$ docker run imageone date

# We can change docker file name
$ mv Dockerfile Dockerfile_One

# Creating Docker image using Dockerfile_One
$ docker build -f Dockerfile_One -t imagetwo .
```

ENTRYPOINT
+++++++++++
-> ENTRYPOINT instructions will execute while creating container

Note: CMD instructions we can override where as ENTRYPOINT instructions we can't override

Example"

```
ENTRYPOINT [ "echo", "Welcome to Ashok IT "]
```

Difference between CMD and ENTRYPOINT

Let’s take an npm init example for node.

CMD :

Let’s assume below is the initial command we added in dockerfile

```
CMD [ "npm", "init" ]
```
Now, If I run docker run -t node npm install

It will override the npm init command from the dockerfile.

CMD [ "npm", "init" ] This will become  CMD [ "npm", "install" ]
It will execute the npm install command rather than npm init as it overrides with npm install.

Now, Let’s talk about

ENTRYPOINT :

Let’s assume the same command is added in docker file but with ENTRYPOINT

ENTRYPOINT [ "npm", "init" ]
Now, If I run docker run -t node install

It will append the npm init command with npm install in the dockerfile.

ENTRYPOINT [ "npm", "init" ] This will become  ENTRYPOINT [ "npm", "init", "install" ]
It will execute the both npm init & npm install commands.

WORKDIR
+++++++++
-> It is used to set Working Directory for an image / container

Ex: WORKDIR <DIR-PATH>

Note: The Dockerfile instructions which are available after WORKDIR those those instructions will be proess from given working directory

ENV
++++

-> ENV is used to set Environment Variables

Ex:  ENV <key> <value>

LABEL
++++++

-> LABEL will represent data in key value pair

-> It is used to add meta data for our image

Ex: LABEL branchName  release

ARG
++++++

-> It is used to avoid hard coded values in Dockerfile

Ex:

ARG branch=develop
LABEL branch $branch

Note: We can pass argument values in RUNTIME

```
$ docker build -t imageone --build-arg branch=feature
```

USER
++++++
-> We can set user for an image / container

Note: After USER instruction, remaining instructions will be processed with given USER

EXPOSE
++++++++
-> It represents on which port number our container is running

-> It is just like a documentation to understand container running port number

++++++++++++++++++++++++++++++++++
Dockerizing Spring Boot Application
++++++++++++++++++++++++++++++++++

```
FROM java:8-jdk-alpine

COPY ./target/spring-boot-docker-app.jar /usr/app/

WORKDIR /usr/app

ENTRYPOINT ["java","-jar","spring-boot-docker-app.jar"]
```

================================
pushing docker into docker hub
=================================

first login to docker hub by using below command

```
docker login
username : <username>
password : <password>
```

once we enter currect username and password it login successfull

tagging image which want to push

```
docker tag <image-name> <dockerhub-username>/<image-name>

docker push <dockerhub-username>/<image-name>
```

Assigment:-
============

Dockerize one springboot application, create a account in dockerhub and push springboot application image into doc kerhub.