ECLIPSE

1.Steps for creating the maven-web project

1. Go to- > File - > New -> Maven Project

2. Select the Maven Project

3. Click -> Next

3. Select the org.apache.maven.archetypes with webapp archetype and click Next

4. Provide the Group id , Artifact id(file nam) and click Finish, Maven web Project is created

5.Here the project created , Check for index .java file containing Hello Welcome to maven web

6. In pom.xml add the dependencies as shown below

7. Right click on server->start server

8. To run the project right click and Run as Maven Clean - It clears out the existing classes previous one

9. Here we can see the console for Build Success.

10. Now again run the project right click and Run as Maven Install – to add artifacts to the project

11. Again run the project Maven test – to test the project

12. Here we can see the console for Build Success

13. Now run the project Run as Maven Build – it creates JAR/WAR files to the project

14. Here specify the goals -> they clean install test or Package , click -> Apply & Run

15.We see the console output

16. Now run the project as Run on server

17. Here we have to select the tomcat server ,Click -> Finish

18. Here we can see output in browser

2.Pushing the maven-web project into github

Now Right click on project ->select show in local terminal->select git bash

Now follow the steps

1. git init
2. git add .
3. git commit –m “maven-web project”
4. create the repo in hub
5. connect to terminal

   git remote add origin <paste the path of http>

6. now push into git hub

   git push –u origin master

7. now project is in github

Jenkins:

Continuous Integration with Jenkins

Prerequisites:

1) Version Control System: Your project should be hosted on a version control system, suchas Git.
2) Mange Jenkins
   - Manage plug-ins :-

   Install the plug-ins which are required for maven project

   i) Maven Integration
   ii) Copy the artifact
   iii) Build pipeline
   iv) Pipeline Utility
   v) Deploy to container
   - Global tool and Configuration

   Set the path for **Maven-Home and Java**

Setting up Continuous Integration in Jenkins:

1) Create a New Jenkins Job:

   Open Jenkins in your web browser and create a new job:

   (a) Click on "New Item" on the Jenkins dashboard.

   (b) Enter a name for your job (e.g., "**Any name**").

   (c) Choose the "Freestyle project" option.

2) Configure Source Code Management:

(a) In the job configuration, go to the "Source Code Management" section.

(b) Choose your version control system (e.g., Git).

(c) Provide the repository URL and credentials if needed…along with branch name where scr code is present

3) Build triggers

(a) Build triggers remotely

We can trigger remotely whenever there is need.

(b) Build after project builds.

(c) Build periodically.

It uses corn job and builds periodically according to the time which is set in corn job irrespective of changes in gihub or not.

(d) Github hook triggers for GITscm polling

Using webhooks

(e) Poll scm

It uses corn job and builds periodically according to the time which is set in corn job irrespective of changes in gihub or not.

4) Configure Build Steps:

(a) Select "Invoke top-level Maven targets."

(b) Enter the goals (e.g., clean install).

5) Configure Post-Build Actions:

(a) Archive the artifacts.

(b) Trigger downstream jobs.

(c) Send email notifications.

6) Save and Run the Job:

(a) Save your job configuration and manually/automatic trigger a build to test if everything is set up correctly. Jenkins will clone your repository, perform the build steps, and report the buildstatus.

2. Continuous Integration with Jenkins with Jenkins pipeline with automatic trigger

```
pipeline {
    agent any
    tools{
        maven 'MAVEN-HOME'
    }
    stages {
        stage('git repo & clean') {
            steps {
                //bat "rmdir  /s /q <repo name of github>"
                bat "git clone <https path of github>"
                bat "mvn clean -f  < repo name of github >"
            }
        }
        stage('install') {
            steps {
                bat "mvn install -f < repo name of github >"
            }
        }
        stage('test') {
            steps {
                bat "mvn test -f < repo name of github >"
            }
        }
        stage('package') {
            steps {
                bat "mvn package -f < repo name of github >"
```

```
        }
      }
    }
}
```

Running a Container:

1.Nginx

- Pull the latest "nginx" Docker image from Docker Hub.
- Run an Nginx container and expose it to port 8080 on your local machine.
- Access the Nginx welcome page from your web browser.

Solution:

docker pull nginx → <search for image name from hub.docker.com>

- Check for docker images
  # docker images
- Now run the container

  # docker run - -name <image name> -d - p 8080:80 <image name>

- now check in browser with public ip of ec2 instance with port number

2. Running the tomcat

- Pull the latest "tomcat" Docker image from Docker Hub.
- Run an tomcat server and expose it to port 6060 on your local machine.
- Access the tomcat page from your web browser

3. Ubuntu

Create a docker container using the official image fro ubuntu and run the container in interactive terminal , execute the some basic commands in it and install the git in it. Push this this image to docker hub

1.pull the official Ubuntu image
docker pull ubuntu:latest
2. check for image
docker images

3. Run the ubuntu container
docker run - it - - name <container name> <image name>
4. Now you can see the ubuntu terminal
ls
apt-get update –y
apt-get install -y git
5. Now check for git version
6. commands for tagging the image
docker tag <image name> <docker hub user name> /<repository name>:<tag>
7. Before pushing the image ,make sure you have are logged in to Docker hub
docker login
8. Push the image
docker push <docker hub user name>/<repository name>:<tag>
9. verify on docker hub

4. Creating a Docker Image with Dockerfile :

Create a docker container using the official image for ubutnu with interactive terminal mode

Vim/vi Dockerfile

FROM ubuntu

MAINTAINER archanareddycse

CMD ["bash"]

:wq

---

1. To build an image from the dockerfile
   # docker build -t myubuntu .

2. To see the image
   # docker images

3. Running conainer from the image

docker run -it myubuntu

5. Creating a Docker Image with Dockerfile :

Create a docker container using the official image for centos and run the container with given CMD

vim dockerfile

FROM centos
MAINTAINER
CMD ["date"]

:wq

---

4. To build an image from the dockerfile
   # docker build -t mycentos .

5. To see the image
   # docker images

6. Running conainer from the image
   # docker run -it   mycentos

6. Creating a Custom Docker Image with Dockerfile :

- Create a new directory on your system and navigate into it.
- Inside the directory, create a simple "Welcome to workshop " HTML file.
- Write a Dockerfile to create a custom Docker image that copies the HTML file into the Nginx default web directory (/usr/share/nginx/html).
- Build the Docker image with a suitable tag.
- Run a container using the custom image and expose it to port 80.
- Verify that you can access your "Hello, World!" page from your web browser.

Solution:

- Create a new directory and navigate into it. Create an "index.html" file with the following content:

Vim/vi index.html

<html>

<head> <title> Welcome to workshop </title>

</head>

<body>

<h1>Welcome to workshop </h1>

</body>

</html>

:wq

- Create a Dockerfile in the same directory with the following content:

Vim/vi Dockerfile

FROM nginx:latest

COPY index.html /usr/share/nginx/html

:wq

- Build the Docker image :
  # docker build -t <image name> .
- Now checks for images
  # docker   images
- Now run the container :
  # docker run - - name <container name> -d - p 9090:80 <image name>
- Now access the nginx page with Hello world by public ip of ec2 instance with port number

Create a docker compose file for setting up dev environment.(DEVELOPMENT SIDE)

mysql container is linked with wordpress container.

```
---
version: '3'
services:  ---------->(root element)
 mydb:      ---------->(1st child)
  image: mysql:5
  environment:
   MYSQL_ROOT_PASSWORD: archana
---------->1st container
 mysite: ---------->2nd child
  image: wordpress
  ports:
   - 5050:80
  links:
   - mydb:mysql
---------> 2nd container
...
:wq
```

To start the above services from dockercompose

```
# docker-compose  up –d
```

(-d is used to avoid allthe logs coming on the screen)

To access wordpress

public_ip:5050

To delete all the containers

```
# docker-compose down
```

2. Steps to create the virtual machine and connecting to it.

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud.

Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

Synonyms

Computer, machine, box, PC, Server = As per AWS terminalogy - Instance

Ex 1: Launch ubuntu instnace

Step 1: Login to AWS

Step 2: Choose region which is near ? (Asia pacific - Mumbai)

Step 3: Services -- EC2

Services -- EC2 --- Launch Instance

Stage 1 --Name (Giving name to the machine) ubuntu

Stage 2 -- Select AMI (Note: Select free tier eligible) ubuntu server

Stage 3 -- Architecture as 64-bit

No of instances -- 1

Stage 4 -- t2.micro

Stage 5 -- Create a new keypair

Stage 6 -- Network Setting ----Create Security group -- (It deals with ports)

We have 0 to 65535 ports

Every port is dedicated to special purpose

Stage 7 -- Storage - 8GB (Observation - we have root - it is same as C Drive)

Stage 8 --- click on launch instance

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

Observation - One machines created.

i)We can access the virtual machine by

powershell /gitbash /webconsole .

ii)Navigate to the path where keypair is present

iii)Paste the ssh -i command from the aws console

a)select the ubuntu server in ubuntu machine

b)click->connect->select SSH client →copy example ssh-i command

iv) Now machine is connected

Docker assignments Questions

1. i) Create an EC2 instance of ubuntu
   ii) install Docker
   iii) Pull image for Nginx Server
   iv) Explore it web browser

   powershell commands

   step :1 cd <paste the path of key pair>

   step :2 <paste the ssh - ikey from ec2 instance>

   step:3 sudo apt-get update

   step:4 sudo apt install docker.io

   step: 5 sudo docker pull nginx

   step:6 check for images

   sudo docker images

   step:7 sudo docker run - it - d - p 8080:80 nginx

   step:6 check in browser with <public ip>:8080

   step:7 will get landing page of nginx page

   Docker assignments Questions on aws

2. i) Create an EC2 instance of ubuntu
   ii) install Docker
   iii) Pull image for Nginx Server
   iv) Explore it web browser

   powershell commands

   step :1 cd <paste the path of key pair>

   step :2 <paste the ssh - ikey from ec2 instance>

   step:3 sudo apt-get update

step:4 sudo apt install docker.io

step: 5 sudo docker pull nginx

step:6 check for images

sudo docker images

step:7 sudo docker run - it - d - p 8080:80 nginx

step:6 check in browser with <public ip>:8080

step:7 will get landing page of nginx page