**Jenkins server installation and configuration**

Server Installation

apt-get update

sudo apt-get install default-jdk

wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -

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

sudo apt-get update

sudo apt-get install jenkins

Once installed like this, you can update to the later version of Jenkins

#sudo apt-get update

#sudo apt-get install jenkins

Integrate jenkins with bitbucket

Create public repo in bitbucket

Clone code in development server and push to bitbucket

Install git command in jenkins server

# apt-get update

# apt-get install git

Login to jenkins server

<http://localhost:8080>

unlock jenkins password

/var/lib/jenkins/secrets/InitialAdminPassword

Install suggested plugin

Once the plugins are installed, you will be prompted to set up the first admin user

Update username and password

Install bitbucket plugin

Go to jenkins – manage jenkins – manage plugin – available search – bitbucket

Select bitbucket and install without restart

Create Job

Click new item

Enter item Name **first-project**

Select free style project

Select source code management

Select .git

Add reposit <https://abbasjam@bitbucket.org/abbasjam/test-project.git>

Enable Web-hook

Login bitbucket server

select test-project – setting

Select webhooks – Add webhooks

Title – test-project

URL - <http://jenkins-ip:8080/bitbucket-hook>

Status – Active

Triggers – Repository push

Save.

PHP Unit Testing

Go to Jenkins server terminal

# apt-get update

# apt-get install phpunit

Go to Jenkins URL s

Install junit plugin

Manage plugin – plugin manager – available

Select junit plugin and install

Select job first-project

Go to build Trigger – execute shell

Command : phpunit CalculaterTest.php –log-junit php-result.xml

Post build action : publish junit test result report

Test report xml : php-result.xml

Integrate bitbucket status API

First create OR key

Go to bitbucket – profile (user) - setting - Access management – OAuth

Select Add consumer

Name : first-project

Calback URL : <http://jenkins-ip:8080/>

Repository : select read and write

Install bitbucket OAuth plugin

Go to jenkins URL

Jenkin – manage plugin – plugin manager – available

Select bitbucket OAuth and bitbucket bukld status notified plugin and install

Jenkins - first-project – post build actions

Enable bitbucket notify build status and notify build start , notify build finish

Credentioal - Add OAuth Keys ( from bitbucket server )

Install FTP server [ QA server ]

# apt-get install vsftpd

# vim /et/vsftpd.conf ( go to line no 31 and uncommand write enable = yes)

#systemctl restart vsftpd

# useradd john

# su – john

# mkdir data

# cd data

Install git ftp in Jenking Server Terminal

# apt-get install git-ftp

Install credentioal binding plugin

Jenkins – credentioal – add credentioal

Jenkins – first-project – build Environment – use secret text

Binding – user name and password seprated

Username variable GITUser

Password variable GITPASSWORD

Credential -

Execute shell

Git ftp init –-user $GITUSER –GITPASSWORD <ftp://qaserver-ip/data> ( init command will execute first time only)

Git ftp push –-user $GITUSER –GITPASSWORD <ftp://qaserver-ip/data>

Master Slave configuration

--------------------------

In Slave install openjdk and docker

# apt-get install default-openjdk ( all slave)

# apt-get install docker.io

Deploy two servers ( slave1 and slave2 )

Go to jenkins master – jenkins – configure – global security

Agents : randam

Save

Manage notes

Add new nodes slave1

Permenent agent

Slave1 configure

Launch method – launch agent via java web start

Current working dir /home/ubuntu/jenkins

Save

Same as slave 2

Click on salve1

Download agent.jar

Copy agant.jar file to slave1 server /home/ubuntu/jenkins

Login to slave1 server

( select node slave1 and copy java –jar agant.jar ……………………………… )

Cd /home/ubuntu/jenkins

Run that command ( java –jar agant.jar …………)

After run this command don’t close the terminal

Now slave are connected to jenkins server.

Go to jenkins

Create job

Test and freestyle project

Go to configure github project

Project url : <http://github-server/.../project.git>

Select – source code management

Repository usrl : <http://github-server/.../project.git>

Enable restickt where this project can be run ( this job only run slave1)

Slave1

Save and build

Build – Add built setup – execute shell

Command : Docket rm –f ${sudo docker ps –a –q}

docket build /home/ubuntu/jenkins/workspace/test –t test

Docket run –it –p 82:80 –d test

Create new project

Prod and freestyle project

( same as slave 1)

Go to configure github project

Project url : <http://github-server/.../project.git>

Select – source code management

Repository usrl : <http://github-server/.../project.git>

Enable restickt where this project can be run ( this job only run slave2)

Slave2

Save and build

Command : Docket rm –f ${sudo docker ps –a –q}

docket build /home/ubuntu/jenkins/workspace/prod –t test

Docket run –it –p 82:80 –d test

Go to test job and configure

Post-build actions

Projects to build : prod

Select trigger only if build is statble

Select + sign in dashboard ( for pipeline )

Install buildpipeline

Select + sign in dashboard

View name : CICD

Slect build pipe line view

Click on ok

Then build pipe line view title : CICD

Select initial job : test

For auto deploy

Go to test job – configure - Build trigger

Github hook trigger to gitscm polling

Configure web hook in github

Docker Install and configuration

* The server is the physical server that is used to host multiple virtual machines. So this layer remains the same.
* The Host OS is the base machine such as Linux or Windows. So this layer remains the same.
* Now comes the new generation which is the Docker engine. This is used to run the operating system which earlier used to be virtual machines as Docker containers.
* All of the Apps now run as Docker containers.

# apt-get install docker.io

# systemctl docker enable

# systemctl restart docker

Docker container using the kernel from base os

Every docker container have two layers

Layer 1 - Boot

Layer 2 - lib/bin

Docker working as server and client method

Docker container is running on server and command running on client

Docker Image and container Basic

---------------------------------

Search container image - using docker hub ofiical image

Using command line

# docker search centos

Run the first container

# docker run –i –t –-name mycontainer centos /bin/bash

-i intractive mode

-t terminal

# docker ps ( only running container )

# docker ps –a ( all container)

# docker start mycontainer

Login to Container

------------------

# docker exec –t –i mycontainer /bin/bash

# docker exec –t –i mycontainer touch /root/a.txt

# docker exec –t –i mycontainer ls /root

# docker run –d –i –name test ubuntu /bin/bash

-d daemons

-i intractive mode

# docker instpect mycontainer ( details of on image or container details )

# docker rm mycontainer ( Remove container )

# docker rm –f mycontainer ( Remove container with force option)

# docker rmi <image-id> ( remove image )

# docker histroy centos ( it will show all the command that were run against the centos image)

# docker top container-id ( Top process with in the container )

# docker stats container-id ( Statistics of a running container )

# docker logs <container-id> ( chekcing error log )

Create docker sample docker file

But Docker also gives you the capability to create your own Docker images, and it can be done with the help of Docker Files. A Docker File is a simple text file with instructions on how to build your images

# vim Dockerfile

From ubuntu

MAINTAINER abbas@gmail.com

RUN apt-get update

RUN apt-get install vim git

# docker build -t abbas/ubuntugit:v1.1 .

# docker run –it --name test1 abbas/ubuntugit:v1.1 /bin/bash

Openssh server configuration

#mkdir fileforssh

# cd fileforssh

# vim fileforssh

FROM ubuntu

MAINTAINER [abbas@gmail.com](mailto:abbas@gmail.com)

RUN apt-get update

RUN apt-get install openssh-server –y

RUN cp /etc/ssh/sshd\_config /etc/ssh/sshd-confi.org

RUN sed -i ‘s/#PermitRootLoing Prohibit-password | PermitRootLogin yes’ /et/sshd/sshd-config

RUN echo ‘root:redhat123’ | chpasswd

RUN mkdir /var/run/sshd

CMD [‘/usr/sbin/sshd’, ‘-D’ ]

EXPOSE 22

Note : EXPOSE is used to accept the port forwarding command

# docker build -t abbas/openssh:v1.1 .

# docker run -i -d -P “2525:25” --name myssh1 abbas/openssh:v1.1

-p ( assign port manually )

-P ( assign port automatically )

# docker ps

# ssh –p 2525 root@ip

Entry Point and Work Directory

-------------------------------------------

Docker Is default login in / directory

WORKDIR - change the current work dirctory

# vi Dockerfile

RUN apt-get update

RUN apt-get install curl apache2 –y

WORKDIR /var/www/html

COPY ./entrypoint.sh /root/entrypoint.sh

ENTRYPOINT /root/entrypoint.sh

EXPOSE 80

# vim enterypint.sh

rm –f /var/run/apache2/apache2.pid

/usr/sbin/apache2 -D FORGROUND

# chmod a+x entrypoint.sh

# docker build -t abbas/apache2:v1.1 .

# docker run -d -P --name webserver abbas/apache2:v1.1

# docker exec -t -i webserver /bin/bash