DevOps project3

* Developer will write code and upload it in github and will give that URL to Devops engineer.
* Devops engineer will write Dockerfile and upload it in his github.
* Github and Jenkins will be integrated.
* Jenkins will download that Dockerfile into Jenkins-server and will copy that file in Ansible-server using rsync command
* Docker package must be installed in ansible-server
* Command will be defined in Jenkins which will build/tag/push in dockerhub. All these commands will be fired from Jenkins on ansible-server

For building image🡪

docker image build -t myimage:v1

for tagging image as we want it to push in dockerhub🡪

docker image tag myimage:v1 shubhamthopate/myimage:v1

for pushing image in dockerhub🡪

docker image push shubhamthopate/myimage:v1

* These commands will create docker-image using this dockerfile in ansible server. (myimage:v1) and will upload that docker-image in dockerhub after tagging
* We’ll write a playbook in ansible-server which will create containers using docker-image. (mycontainer)
* But there will be problem when developer changes his sourcecode. When we will replace that sourcecode url in our playbook and try building/tagging/pushing job will failed as previous image will be there with same name.
* We could resolve this problem by making changes in Jenkins commands and also in ansible-playook BUT that’s not a solution because we want fully automation (changes in name of image)
* For that we’ll have to use Jenkins predefined variables.

Following are Jenkins predefined variables:

* **BUILD\_NUMBER** - The current build number. For example "153"
* **BUILD\_ID** - The current build id. For example "2018-08-22\_23-59-59"
* **BUILD\_DISPLAY\_NAME** - The name of the current build. For example "#153".
* **JOB\_NAME** - Name of the project of this build. For example "foo"
* **BUILD\_TAG** - String of "jenkins-${JOB\_NAME}-${BUILD\_NUMBER}".
* **EXECUTOR\_NUMBER** - The unique number that identifies the current executor.
* **NODE\_NAME** - Name of the "slave" or "master". For example "linux".
* **NODE\_LABELS** - Whitespace-separated list of labels that the node is assigned.
* **WORKSPACE** - Absolute path of the build as a workspace.
* **JENKINS\_HOME** - Absolute path on the master node for Jenkins to store data.
* **JENKINS\_URL** - URL of Jenkins. For example *http://server:port/jenkins/*
* **BUILD\_URL** - Full URL of this build. For example *http://server:port/jenkins/job/foo/15/*
* **JOB\_URL** - Full URL of this job. For example *http://server:port/jenkins/job/foo/*
* Hence command for building image becomes🡪

docker image build -t $JOB\_NAME:v1.$BUILD\_ID .

* Command for tagging becomes 🡪

docker image tag $JOB\_NAME:v1.$BUILD\_ID shubhamthopate/$JOB\_NAME:V1.$BUILD\_ID

* Command for pushing image in dockerhub 🡪

docker image push shubhamthopate/$JOB\_NAME:V1.$BUILD\_ID

* But then also problem isn’t totally resolved. Our playbook will push latest image only and for that we need to fire some more commands in Jenkins so that versioning of images along with latest image will get done in dockerhub.
* Command will be for setting new tag ‘latest’ to latest image 🡪

docker image tag $JOB\_NAME:v1.$BUILD\_ID shubhamthopate/$JOB\_NAME:latest

* Next command will be for pushing this new tag

docker image push shubhamthopate/$JOB\_NAME:latest

* Now we’ll define a playbook in such a way that there will be only name of image and no tag/version of image will be mentioned. So that latest image available in dockerhub will get pushed

- hosts: all

tasks:

- name: create container

shell: docker container run -itd --name cloudknowledge-container -p 9000:80 shubhamthopate/cloudknowledge-job

* As job is going to run multiple times there will be so many images in ansible server which will consume unnecessary storage. We’ll set one more command in Jenkins which will delete images (which is getting build/latest image) from ansible server

docker image rmi $JOB\_NAME:V1.$BUILD\_ID shubhamthopate/$JOB\_NAME:V1.$BUILD\_ID shubhamthopate/$JOB\_NAME:latest

* One more problem arises when docker try to fetch latest image. Old latest image will be there in dockerhost and hence it will not fetch new latest image from dockerhub. Hence we need to delete previous latest image. For that we’ll make changes in playbook.

- hosts: docker-host

tasks:

- name: stop container

shell: docker container stop cloudknowledge

- name: remove container

shell: docker container rm cloudknowledge

- name: remove docker image

shell: docker image rmi shubhamthopate/cloudknowledge

- name: create new container

shell: docker container run -itd --name cloudknowledge-container -p 9000:80 shubhamthopate/cloudknowledge-job

1. Launch 3 ec2 (amazon linux)🡪 Jenkins, ansible and dockerhost
2. Connect to Jenkins and install Jenkins, git

# cd

# yum install java\* -y

# wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

# rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io.key>

# yum install Jenkins -y

# systemctl start jenkins

# systemctl enable jenkins

# yum install git -y



connect to ansible server--> install ansible

# amazon-linux-extras install ansible2

vim /etc/ansible/hosts

add -->

[dockerhost]

private\_ip\_of\_dockerhost

1. create pw-less authentication between ansible-server and docker-host

Get connected with docker-host server

passwd root

vim /etc/ssh/sshd\_config

PermitRootLogin yes

passwordAuthentication yes

:wq

systemctl restart sshd

Get connected with ansible-server

# ssh-keygen

# ssh-copy-id -i root@private\_ip\_of\_docker-host

Finally make sure whether passwordless authentication is successful or not

# ssh root@ private\_ip\_of\_docker-host

1. create pw-less authentication between jenkins-server and ansible-server

Get connected with ansible-server

passwd root

vim /etc/ssh/sshd\_config

PermitRootLogin yes

passwordAuthentication yes

:wq

systemctl restart sshd

Get connected with jenkins-server

# ssh-keygen

# ssh-copy-id -i root@private\_ip\_of\_ansible-server

Finally make sure whether passwordless authentication is successful or not

# ssh root@ private\_ip\_of\_ansible-server

1. install docker in ansible-server and docker-host server

# yum install docker -y

# systemctl start docker

# systemctl enable docker

1. Get connected with jenkins-server

passwd root

vim /etc/ssh/sshd\_config

PermitRootLogin yes

passwordAuthentication yes

:wq

1. systemctl restart sshd
2. install 'publish over ssh' plugin in Jenkins
3. manage jenkins--> configure system --> scroll down--> SSH servers--> Add
4. fill details of jenkins server, give root id and pw,

name: Jenkins

Hostname: private-ip-of-jenkins-server

1. Test Configuration--> make sure whether successful or not
2. do same for Ansible server
3. integrate github with Jenkins

go to github 🡪 create new repo 🡪 create file- Dockerfile 🡪 paste following data

FROM centos:latest

MAINTAINER shubham

RUN yum install httpd -y \

zip \

unzip

ADD https://www.free-css.com/assets/files/free-css-templates/download/page247/kindle.zip /var/www/html/

WORKDIR /var/www/html

RUN unzip kindle.zip

RUN cp -rvf markups-kindle/\* .

RUN rm -rf \_MACOSX markups-kindle kindle.zip

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]

EXPOSE 80

Save 🡪 commit

go to jenkins

copy jenkins url

go to gihub page

settings --> webhooks--> add webhook

Payload url -->http://jenkins\_ip:8080/github-webhook/

content type--> application/json

go to Jenkins page

cloud--> configure --> add new token--> generate --> copy token keep it somewhere--> apply and save

go to github page

secret--> paste token

add webhook

create new job--> freestyle --> give name ‘cloudknowledge-job’

source code management --> git

give url of Dockerfile in github

change ‘master’ to ‘main’

1. Build Trigger

Github hook trigger got GITScm polling

1. Build

Add build step

Send file or execute commands over ssh

Name: Jenkins

Exec command:

rsync -avh /var/lib/jenkins/workspace/cloudknowledge-job/Dockerfile [root@private-ip-of-ansible-server:/opt](mailto:root@172.31.10.81:/opt)

1. Once again Add build step

Name: ansible

Exec command:

cd /opt

docker image build -t $JOB\_NAME:v1.$BUILD\_ID .

docker image tag $JOB\_NAME:v1.$BUILD\_ID shubhamthopate/$JOB\_NAME:V1.$BUILD\_ID

docker image tag $JOB\_NAME:v1.$BUILD\_ID shubhamthopate/$JOB\_NAME:latest

docker image push shubhamthopate/$JOB\_NAME:V1.$BUILD\_ID

docker image push shubhamthopate/$JOB\_NAME:latest

docker image rmi $JOB\_NAME:V1.$BUILD\_ID shubhamthopate/$JOB\_NAME:V1.$BUILD\_ID shubhamthopate/$JOB\_NAME:latest

1. Go to ansible server and login to docker

# docker login

1. Lets create playbook in ansible server

# mkdir /sourcecode

# cd /sourcecode

# vim docker.yml

- hosts: docker-host

tasks:

- name: stop container

shell: docker container stop cloudknowledge

- name: remove container

shell: docker container rm cloudknowledge

- name: remove docker image

shell: docker image rmi shubhamthopate/cloudknowledge

- name: create new container

shell: docker container run -itd --name cloudknowledge-container -p 9000:80 shubhamthopate/cloudknowledge-job

1. Go to Jenkins 🡪 add postbuild actions 🡪 send build artifatcs over ssh 🡪

Name: ansible

Exec command: ansible-playbook /sourcecode/docker.yml

Apply 🡪 save 🡪 build now

1. Connect to dockerhost and check whether container is created or not
2. Go to dockerhub 🡪 check image is there or not
3. Connect to ansible server 🡪 check image is deleted or not
4. Open public-ip of docker-host🡪 webpage shall appear
5. Lets change url in dockerfile and check whether pipeline runs automatically

Go to [www.free-css.com](http://www.free-css.com)

Select any template 🡪 copy link

Go to ansible server🡪

# cd /opt

# ls

# unzip file\_name.zip

# ls

1. Go to dockerfile 🡪 paste following content

FROM centos:latest

MAINTAINER shubham

RUN yum install httpd -y \

zip \

unzip

ADD https://www.free-css.com/assets/files/free-css-templates/download/page254/photogenic.zip /var/www/html/

WORKDIR /var/www/html

RUN unzip photogenic.zip

RUN cp -rvf photogenic/\* .

RUN rm -rf photogenic photogenic.zip

CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]

EXPOSE 80

Save and commit

1. Check whether new container gets created