DOCKER

Create a docker registry on centos :

1. Yum –y update
2. Install : Yum –y install docker-distribution
3. Edit : vi /etc/docker-distribution/registry/config.yml
4. check firewalld

* firewall-cmd --add-port=5000/tcp –permanent
* firewall-cmd –reload

5.Start docker registry service

* Systemctl start docker-distribution
* Systemctl enable docker-distribution
* Systemctl status docker-distribution

1. *confirm port :*

* *telnet <machine ip:5000>*

1. Add Insecure Registry to Docker Engine

* *Edit the file :* /etc/docker/daemon.json

{

“insecure-registries”:[“myregistry.local:5000”]

}

1. *Restart docker engine :*

* *Systemctl restart docker*

1. *Push docker images to local registry*

* *Cat /etc/hosts , and map ip with registry :*
* *172.26.41.59 myregistry.local*
* *Pull : Docker pull centos:latest*
* *Tag : Docker tag centos:latest myregistry.local:5000/centos:latest*
* *Push : docker push myregistry.local:5000/centos:latest*

*If u get Sha256: means that the image is pushed*

1. *Check the Images stored under :*

* *Ls /var/lib/registry/docker/registry/v2/repositories*

1. *download image on local registry :*

* *docker pullmyregistry.local:5000/centos:latest*

Open ports on running container

1. systemctl status iptables
2. yum install iptables-services -y
3. systemctl status iptables
4. systemctl enable iptables
5. systemctl start iptables
6. rpm -qc iptables-services
7. hostname -i
8. ifconfig
9. iptables -nvL
10. yum install -y ifconfig
11. ifconfig
12. netstat -ntlp | grep 80
13. yum install netstat
14. iptables -t nat -A POSTROUTING --source 172.17.0.8 --destination 172.17.0.8 -p tcp --dport 80 -j MASQUERADE
15. iptables -t nat -A DOCKER ! -i docker0 --source 0.0.0.0/0 --destination 0.0.0.0/0 -p tcp --dport 80 -j DNAT --to 192.68.122.1:80
16. iptables -m state -h
17. iptables -p icmp -h
18. iptables -j DROP -h
19. iptables -A DOCKER ! -i docker0 -o docker0 --source 0.0.0.0/0 --destination 172.17.0.8 -p tcp --dport 80 -j ACCEPT
20. systemctl status firewalld.service
21. firewall -cmd --zone=public --add-port=443/tcp --permanent
22. yum install firewalld
23. firewall -cmd --zone=public --add-port=443/tcp --permanent
24. systemctl status firewalld.service
25. systemctl start firewalld.service
26. firewall -cmd --zone=public --add-port=443/tcp --permanent
27. firewall-cmd --zone=public --add-port=443/tcp --permanent
28. firewall-cmd --reload
29. firewall-cmd --zone=public --add-port=80/tcp --permanent
30. firewall-cmd --reload
31. firewall-cmd --listen-all
32. firewall-cmd --list-all
33. iptables-save | grep 80
34. iptables-save | grep 443
35. firewall-cmd --zone=public --query-port=80/tcp
36. firewall-cmd --zone=public --query-port=443/tcp
37. firewall-cmd --zone=public –query-port=23/tcp

**readme to setup hdp-3.0 follow the following instructions**

1.cd /root/ hdp\_test\_ali/ hdp\_bridge\_3.0

execute : sh automation.sh

2. On the completion of the script

Login into the containers and get the hostname & ipaddress of the all containers.

docker exec –it container-name /bin/bash

docker ps

ostname –f

hostname –i

Install jq if not there :

wget -O jq https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64

chmod +x ./jq

cp jq /usr/bin

4. Login into the container which has ambari-server .

* cd /HDP/group\_vars/ and edit vi HDP\_Variables.yml
* edit : HDP\_XML url , HDP\_REPO , HDP\_REPO , AMBARI\_REPO
* if you want to use exiting mysql then Hive\_DB: "mysql"
* change the value , From here : HDP\_XML url , HDP\_REPO , HDP\_REPO
* change the ambary repo valur from cd /etc/yum.repos.d

<http://172.26.41.85/HDP_3.0/>

<http://172.26.41.85/HDP_3.0/HDP/centos7/3.0.1.0-187/HDP-3.0.1.0-187.xml>

* Replace “bd1d31a758a1” with hostname of "ambari-server container"
* Replace “1248bb9b1fca” with the hostname of your "agent’s containers"
* cd /HDP/scripts/bin and execute mainSetup.sh

5. exit from the container :exit

6. docker port <container-name> and select the port on with ambari-server port 8080 is bind

8080/tcp -> 0.0.0.0:32777

7. Open the browser and type your <system\_ip: port>

8. You can see now Ambari-ui and login in with default credentials

Username:admin

Password:admin

7. if you want to use existing database for hive then go to ambari-ui and follow these steps

* services -> hive -> configs -> database .
* select Existing MySQL / MariaDB Database
* Database Name : hive
* Database Username : root
* Database Password : ambari
* Database Url : replace hostname with your master i.e your ambari-server hostname

8. execute following commands

* docker exec -it ambari\_server\_3.0\_demo /bin/bash
* cd /usr/share/java
* ambari-server setup --jdbc-db=mysql --jdbc-driver=/usr/share/java/mysql-connector.jar
* and then restart hive metastore and hive server

9. steps to stop and delete the container

* docker stop CONATINER-NAME
* docker rm CONTAINER-NAME

10. delete images

* docker rmi ambari\_agent\_uniquename ambari\_server\_uniquename aishwarya0296/centos7

To setup hdp-2.5 follow the following instructions:

1. Change the directory to : cd /root/hdp\_bridge\_2.50 and execute automation.sh

If you get following issue during the execution of automation.sh script follow the steps to clean and start the system and execute automation.sh again after deleting containers and images . Also remove the network created .

ERRORS:

* Failed to get D-Bus connection: No such file or directory
* Failed to get D-Bus connection: No such file or directory
* Failed to get D-Bus connection: No such file or directory
* Enter password:
* mysqladmin: connect to server at 'localhost' failed
* Error: 'Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock' (2)'
* Check that mysqld is running and that the socket: '/var/lib/mysql/mysql.sock' exists!

To delete container and images :

* stop and delete the container on your master i.e your ambari-server

docker stop CONATINER-NAME

docker rm CONTAINER-NAME

* delete image

docker rmi ambari\_agent\_2.5 ambari\_server\_2.5 aishwarya0296/centos7

To remove network :

* Docker network ls
* Docker network rm <network name>

#### rexecute automation.sh#####

2. on successful execution of automation.sh , there will be two input parameter which will ask you for the number of cluster you want to create and the unique identification name of your cluster.

* Enter the No of agents : enter the no of ambari-agents you want in the cluster
* Enter the appender : Make sure that There should be no special character in the name of appender , and it should be not as same as existing identifier .

3. On the completion of the script

To get the container-name run "docker ps" command see the last column where you will find the unique name that you provided earlier.

Login into the containers (all) : "docker exec –it container-name /bin/bash"

get the hostname of the containers with hostname –f & hostip of the all the containers with hostname –I .

Also make entries of the nodes present in your cluster in "/etc/hosts` file" of each container.

Note the hostname and hostip of all the containers present in the cluster

Also Check the heartbeat of the container as already in container . check it by moving into /var/log/ambar-agent and then open vi ambari-agent.log to check heartbeat .

:/heartbeat

4. Login into the container which has ambari-server with "docker exec -it container-name /bin/bash"

* This will open an interactive terminal for you and you will be on root directory then change directory
  + -cd /HDP/group\_vars/ and open vi HDP\_Variables.yml
* Edit HDP\_Variables.yml with the following details –
  + If you want to use exiting mysql then Hive\_DB: "existingmysql”
  + Replace “bd1d31a758a1” with hostname of "ambari-server-container” wherever occurred in the file
  + Replace “1248bb9b1fca” with the hostname of your "ambari-agent-container" wherever occurred in the file.
  + At a point hostip needs to be changed , there also mention the hostnames not the hostips , which are the hostnames of “ambari-server container” and “ambari-agent container”
  + Replace exiting variables with your variables value , the variables whose values needs to be changed are : AMBARI\_REPO, HDP\_REPO, and HDP\_UTILS\_REPO with your repos URL
  + The base URL for the repos are available at : /etc/yum.repos.d
  + Update all the variables in HDP\_Variables.yml , whichever required and save the file.
* After saving change the directory to : cd /HDP/scripts/bin and execute mainSetup.

As: " sh mainSetup.sh" .

5. once u see accepted status , exit from the container and to exit from the container type exit

6. to check the ui , follow the steps :

docker port <container-name>

output will be :

* + 10000/tcp -> 0.0.0.0:10005
  + 8188/tcp -> 0.0.0.0:8193
  + 8088/tcp -> 0.0.0.0:8093
  + 8190/tcp -> 0.0.0.0:8195
  + 9083/tcp -> 0.0.0.0:9088
  + 18081/tcp -> 0.0.0.0:18086
  + 4040/tcp -> 0.0.0.0:4045
  + 50070/tcp -> 0.0.0.0:50075
  + 10001/tcp -> 0.0.0.0:10006
  + 11000/tcp -> 0.0.0.0:11005
  + 19888/tcp -> 0.0.0.0:19893
  + 8080/tcp -> 0.0.0.0:32777
  + 10002/tcp -> 0.0.0.0:10007
  + 2181/tcp -> 0.0.0.0:2186
  + 8042/tcp -> 0.0.0.0:8046
* Now , select the port with which ambari-server port 8080 is bind i.e 32777

Here 32777 is the port of your system expose to access UI of Ambari-Server

7. Open the browser and type your <system\_ip: port> -> 172.26.41.56: 32777

8. Ui for Ambari will be prompted, login it with default credentials

Username:admin

Password:admin

9. if you want to use existing database for hive then go to ambari-ui and follow these steps

* + services -> hive -> configs -> advanced .
    - * select Existing MySQL / MariaDB Database
      * Database Name : ambari
      * Database Username : root
      * Database Password : ambari
      * Database Url : replace hostname with your master i.e your ambari-server hostname
      * save the changes made above and restart hive metastore and hive server
  + If you get error like this while updating the hive database , Then execute following commands :
    - "apache.hadoop.hive.metastore.HiveMetaException: Failed to load driver"
    - docker exec -it ambari\_server\_2.5\_demo /bin/bash
    - cd /usr/share/java
    - ambari-server setup --jdbc-db=mysql --jdbc-driver=/usr/share/java/mysql-connector.jar
    - Then restart hive metastore and hive server

#####################################################################################

steps to stop and delete the container on your master i.e your ambari-server

* docker stop CONATINER-NAME
* docker rm CONTAINER-NAME

to delete images

* docker rmi ambari\_agent\_2.5 ambari\_server\_2.5 aishwarya0296/centos7

<https://labs.play-with-docker.com/>

docker login

anamikanagar01

PAPAa1234@

docker commands :

when the Dockerfile is built, it becomes a Docker Image and when we run the Docker Image then it finally becomes a Docker Container.

**Dockerfile:**A Dockerfile is a text document which contains all the commands that a user can call on the command line to assemble an image.docker build to create an automated build to execute several command-line instructions in succession.

**Docker Image** can be compared to a template which is used to create Docker Containers. Use docker run to run the image and create a container.

**Docker Container** is a running instance of a Docker Image as they hold the entire package needed to run the application

**create image using**

docker commit : by saving state of a docker container

docker build : by writing a docker file

docker file is better as it reduces the size , also it is better bcz by saving state of container we cant save the state of a process (start/stop , eg:nginx)

**Docker Engine** is simply the docker application that is installed on your host machine. It works like a client-server application which uses:

* A **server** which is a type of long-running program called a daemon process
* A command line interface (CLI) **client**
* REST API is used for communication between the CLI client and Docker Daemon

**Docker Registry** is where the Docker Images are stored. The Registry can be either a user’s local repository or a public repository like a Docker Hub allowing multiple users to collaborate in building an application.

### **Advantages of Containerization over Virtualization:**

* Containers on the same OS kernel are lighter and smaller
* Better resource utilization compared to VMs
* Boot-up process is short and takes few seconds

Virtualization and Containerization both let you run multiple operating systems inside a host machine.

**installing docker :**

sudo apt-get update

sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-virtual

sudo apt-get install docker-engine

sudo service docker start

sudo service docker status

sudo service docker stop

sudo service docker restatrt

docker -v

sudo docker pull centos : verify by pulling an image from docker hub

sudo docker run -it centos : creating docker container

**creating a dockerfile :**

nano Dockerfile:

FROM ubuntu:latest

MAINTAINER Sahiti (email@domain.com)

RUN apt-get update

RUN apt-get install -y nginx

ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]

EXPOSE 80

create image : docker build .

Check images :docker images

create container : docker run -it -p port\_number -d image\_id

check conatiners: docker ps