13 June 2020

14:41

Ansible

Section1

section2

section3

section4

section5

section6

section7

section8

section9

section10

section11

section12

section13

section14

section15

section16

section17

section18

section19

section20

section21

section22

section23

<https://github.com/session>

username:-venkatping

password:-Krish\_1\_2

<https://login.linuxacademy.com/login>

username:-awsvenkat

password:-Krish@123

<https://www.neobux.com/m/l/>

prawin461

Pavan26101994@

<https://sso.redhat.com/auth/realms/redhat-external/protocol/saml/clients/legacy-idp-servlets>

chilakanaveen

Krish@123

<https://hub.docker.com/search?q=&type=image>

69129902

Krish@123

Why use Docker:-

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

Docker makes it really easy to install and run software without

worrying about setup or dependencies

What is docker:-

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

Docker is a platform or ecosystem around creating and running containers

Docker can be Docker client,Docker server,Docker Machine,Docker images,Docker Hub,Docker compose

when i run docker run redis in the backside there will be series of actions executed

and it reaches the docker hub and it downloads a single file called image

This image, is a single file with all the dependencies and config required to run a program

we can use this image to something called container

Container is a instance of an image,Runs a program

Docker Client:-

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

Docker CLI is nothing but Docker Client,By using Docker we are going to

issue commands to

Docker Client doesn't actually do anything with the containers or

images and it is used to interact with Docker Server

Docker Server:-

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

Docker Daemon is nothing but Docker Server,Docker Server is

responsible for creating images,uploading images,Maintaining

containers,running containers,etc

[root@localhost ~]# docker version

Namespacing is used to isolating resources per process or group of processes

Processes,Harddrive,network,Users,Hostnames,Interprocess communication

Control groups(cgroups) limit amount of resources used per process

Memory,CPU Usage,HD I/O,Network Bandwidth

Creating and Running a Container from an image:-

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

Docker(reference the docker client)

run (Try to create and run a container)

<image name>(name of image to use for this container)

docker run command is used to run a container from an image

docker ps command list all running containers

#docker run uuntu

#docker ps

#docker ps -a

#docker stop 1a8ce64ce7da

#docker rm cool\_mestorf

#docker pull nginx we are pulling image but we are notrunning it

#docker rmi ubuntu to remove an image please stop all containers in it

#docker run -p 80:5000 kodekloud/simple-webapp (port mapping)

#docker run -v /opt/datadir:/var/lib/mysql mysql (saving data to external directory)

#docker inspect centos (this command is to get extra information of an image)

#docker logs centos

RUN STDIN:-

---------

To provide an standard input we need to use -i

#docker run kodekloud/simple-prompt-docker

#docker run -i kodekloud/simple-prompt-docker

-t stands for sudo terminal

#docker run -it kodekloud/simple-prompt-docker

with -i and -t we attached to kernel and interactive mode of container

RUN Volume mapping:-

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

we are mapping /var/lib/mysql data in a container to outside of container /opt/datadir

so if we rm mysql image the data in /opt/datadir will not be deleted

/opt/datadir is an external directory

#docker run -v /opt/datadir:/var/lib/mysql mysql

Inspect container:-

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

#docker inspect containerid it will give more information of an container

Container logs:-

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

#docker logs containerid it will give logs of an container

commands:-

--------

#docker run ubuntu cat /etc/\*release\*

NAME="Ubuntu"

VERSION="20.04 LTS (Focal Fossa)"

ID=ubuntu

ID\_LIKE=debian

PRETTY\_NAME="Ubuntu 20.04 LTS"

VERSION\_ID="20.04"

HOME\_URL="<https://www.ubuntu.com/>"

SUPPORT\_URL="<https://help.ubuntu.com/>"

BUG\_REPORT\_URL="<https://bugs.launchpad.net/ubuntu/>"

PRIVACY\_POLICY\_URL="<https://www.ubuntu.com/legal/terms-and-policies/privacy-policy>"

VERSION\_CODENAME=focal

UBUNTU\_CODENAME=focal

#docker run ubuntu:17.10  (to pull/run specific version and here 17.10 is a tag)

#docker run ubuntu:17.10 cat /etc/\*release\*

#docker run -d ubuntu sleep 1000   -d is for dettach mode,detach means it allows us to run another command in shell

#docker attach containerid         attach means it runs in foreground and i t will not allow us to run another command in shell

#docker run timer                  it runs in attach mode

#docker run -d timer  and #docker ps and #docker attach containerid

#docker run jenkins

jenkins is an webserver

port mapping cannot be done when on running container

#docker run -p 8080:8080 jenkins

#docker run jenkins

#mkdir my-jenkins-data

#docker run -p 8080:8080 -v /root/my-jenkins-data:/var/jenkins\_home -u root jenkins

#docker ps

#docker run -p 8080:8080 -v /root/my-jenkins-data:/var/jenkins\_home -u root jenkins

command and encrypt:-

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

- containers are not meant to host an operating system

- containers are meant to host specific task/process

- such as an instance of an webserver/application server,computation

- container lives as long as the process is alive

- if the webserver inside container is stops/crashes the container exits

creating a container:-

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

we have done here is created hello-world and run primary startup command in hello-world

This can be done in another way is

#docker run hello-world  (This can do create and start at sametime)

[root@naveenhost ~]# docker create hello-world

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

0e03bdcc26d7: Pull complete

Digest: sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1

Status: Downloaded newer image for hello-world:latest

816df9bffe82a140aaab04c71f9b621ff6d02bc01b5d844d48fc649104f635ec

[root@naveenhost ~]# docker start hello-world

Error response from daemon: No such container: hello-world

Error: failed to start containers: hello-world

[root@naveenhost ~]# docker start -a hello-world

Error: No such container: hello-world

[root@naveenhost ~]# docker start -a 816df9bffe82a140aaab04c71f9b621ff6d02bc01b5d844d48fc649104f635ec

-a is for attach

-d is for dettach

Restarting stopped container:-

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

[root@naveenhost ~]# docker run centos echo hi there

hi there

[root@naveenhost ~]# docker ps -a

CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                         PORTS               NAMES

bbfe67b5794c        centos              "echo hi there"          6 seconds ago       Exited (0) 4 seconds ago                           xenodochial\_varahamihira

[root@naveenhost ~]# docker start bbfe67b5794c

bbfe67b5794c

[root@naveenhost ~]# docker start -a bbfe67b5794c

hi there

[root@naveenhost ~]# docker logs bbfe67b5794c

hi there

hi there

hi there

[root@naveenhost ~]#

Delete all containers:-

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

#docker system prune

Stop a container:-

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

#docker stop containerid

#docker ps

Multicommand container:-

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

[root@naveenhost ~]# docker ps

CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES

2549ec3965fa        redis               "docker-entrypoint.s…"   6 minutes ago       Up 6 minutes        6379/tcp            focused\_shockley

[root@naveenhost ~]# docker exec 2549ec3965fa redis-cli (here we didn't get prompt to input commands for redis-cli, so we need to use -it)

[root@naveenhost ~]#

[root@naveenhost ~]# docker exec -it 2549ec3965fa redis-cli

127.0.0.1:6379> iam naveen

(error) ERR unknown command `iam`, with args beginning with: `naveen`,

127.0.0.1:6379> set myvalue 5

OK

127.0.0.1:6379> get myvalue

"5"

127.0.0.1:6379>

#docker exec -it 2549ec3965fa sh

sh means command processor in a container

by using the above command we can run as many commands in a running container

#docker exec -it 2549ec3965fa bash

#docker exec -it centos sh

#docker run -p 8000:80 --name webhost -d nginx:1.12 nginx -T

Container:-

---------

- They are just processes

- Limited to what resources they can access

- Exit when process stops

#docker top containerid (it lists running process)

#docker ps  #docker container ls

#docker images

#docker top containerid (it lists process running in one container)

#docker inspect containerid (details of config of container)

#docker stats containerid

#docker stats containerid(performance stats of a container)

#docker run -p 3000:80 --name httpd httpd (ruuning httpd on port 3000)

#docker container top nginx(lists process running on nginx)

#docker container stats(it will show all containers performance)

#docker container stats nginx(it will show container performance of nginx)

Getting a shell inside containers:-

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

#docker container run -it    (start new container interactively)

#docker container exec -it    (run additional command in existing container)

#docaker ps -a/#docker container ls -a

#docker ps/#docker container ls

#docker run -it nginx bash

#docker run -it centos sh

Docker container exec: run additional process in running container

Docker Network concepts:-

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

- Review of docker container run -p

- for local dev/testing,networks usually "just work"

- quick port check with #docker container port <container>

- Learn concepts of Docker Networking

- Understand how network packets move around Docker

Docker Network defaults:-

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

- Each Container connected to a private virtual network "bridge"

- Each virtual container routes through NAT Firewall on host IP

- All containers on a virtual network can talk to each other without -packets

- Best practice is to create a new virtual network for each app:

  - network "my\_web\_app" for mysql and php/apache containers

  - network "my\_api" for mongo and nodejs containers

- Batteries included but removable

  - Defaults work well in many cases,but easy to swap out parts to customize it

- Make new virtual networks

- Attach containers to more than one virtual network (or none)

- skip virtual networks and use host IP(--net=host)

- Use different Docker Network drivers to gain new abilities

Docker Networks:CLI Manager:-

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

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

[root@naveenhost ~]# docker run -p 3000:80 --name webhost -d nginx

ac980dd885428bd65661c3f31a449fd48f7c3d718052a2708130ae8beb8f4b06

[root@naveenhost ~]# docker port webhost

80/tcp -> 0.0.0.0:3000

[root@naveenhost ~]#

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

- Show Networks                     #docker network ls

- Inspect a network                 #docker network inspect

- create a network                     #docker network create --driver

- Attach a network to container     #docker network connect

- Detach a network from container     #docker network diconnect

[root@naveenhost ~]# docker network ls

NETWORK ID          NAME                DRIVER              SCOPE

62c0a2973b31        bridge              bridge              local

ed54fb9b3b25        host                host                local

7d0cb7cc0866        none                null                local

- In some cases we can see docker0 and in some cases we can

  bridge and they two are same

- Default Docker virtual network,which is NAT'ed behind the Host IP

- Here none is not attached to anything

- Spawns a new virtual network for you to attach containers to

[root@naveenhost ~]# docker network create my\_app\_net

57ccb638043ee80944c88dacd80e14e1443b956f862ca511aa0a94f25d08cb78

[root@naveenhost ~]# docker network ls

NETWORK ID          NAME                DRIVER              SCOPE

62c0a2973b31        bridge              bridge              local

ed54fb9b3b25        host                host                local

57ccb638043e        my\_app\_net          bridge              local

7d0cb7cc0866        none                null                local

[root@naveenhost ~]#

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

#docker run -p 3000:80 --name webhost -d nginx

#docker port webhost

#docker inspect --format "{{.NetworkSettings.IPAddress}}" webhost

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

#docker network ls

#docker network create my\_app\_net

#docker container run -d --name new\_nginx --network my\_app\_net nginx

#docker network inspect my\_app\_net

#docker network inspect bridge

#docker network connect --help

#docker network connect my\_app\_net 2b2274116b64 (by using this we can connect my\_app\_net network to any container)

#docker network connect my\_app\_net webhost

#docker inspect webhost

#docker network disconnect my\_app\_net webhost

#docker network inspect my\_app\_net

- create your apps so frontend/backend sit on same docker network

- Their inter-communication never leaves host

- All externally exposed ports closed by default

- you must expose via -p which is better default security

- This gets even better later with swarm and Overlay networks

Docker Networks: DNS:-

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

- Understand how DNS is the key to easy inter-container communications

- Container shouldn't rely on IP's for inter-communication

- DNS for friendly names is built in if you use custom networks

- see how it works by default with custom networks

- Learn how to use --link to enable DNS on default bridge network

- static IP's and using IP's for talking to containers is

  an anti-pattern, do your best to avoid it

- Docker daemon have built in DNS server that containers use by default

- Docker defaults the hostname to the containers name, but you can set aliases if required

#docker container run -d --name my\_nginx --network my\_app\_net

#docker container run -d --name new\_nginx --network my\_app\_net

#docker container exec -it my\_nginx ping new\_nginx

#docker network ls (The default bridge have one disadvantage

here, it is not have DNS server built in by default so use

below command)

#docker container create --link

- Things gets way easier with docker compose in future section

#docker network create app

#docker container run -d --network app --network-alias search elasticsearch:2

#docker container run -d --network app --network-alias search elasticsearch:2

#docker container run --rm --network app alpine nslookup search

#docker container run --rm --network app centos curl -s search:9200

Container images:-

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

- All about images, the building blocks of containers

- What is an image and whats is in it

  - it is an Application binaries and dependencies

  - Metadata about the image data and how to run the image

  - official definition: An image is an ordered collection of

    root filesystem changes and the corresponding execution

    parameters for use within a container runtime

  - Inside this image that is not a complete os, there is no

    kernel no kernel modules(e.g: drivers),it is really just

    binaries that your application needs because the host

    provides the kernel and that is one of the distinct

    characterstics around containers that makes it different

    from virtual machines

  - small as one file(your app binary) like a golang static binary

  - Big as a Ubuntu distro with apt, and apache,php and more installed

- Using Docker Hub Registry

- Managing our local image cache

#docker pull nginx

#docker pull nginx:1.10

images and their layers dicover the image cache:-

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

- image layers

- union file system

- history and inspect commands

- copy on write

images are designed using union file system concept

#docker images

#docker history nginx

#docker inspect nginx

- images are made up of file system changes and metadata

- Each layer is uniquely identified and only stored once on a host

- This saves storage space on host and transfer time on push/pull

- A container is just a single read/write layer on top of image

- inspect command is like metadata

**Image Tagging and Pushing to Docker Hub:-**

#docker image tag --help

latest tag:-

it is just the default tag but image owners should assign it to the newest stable version

#docker image tag nginx bretfisher/nginx

#docker image push bretfisher/nginx (we cannot push becase we need to login first to our repository

#docker login

Username: 69129902

Password:

#docker image push bretfisher/nginx (after login please execute this command

#cat .docker/config.json

{

        "auths": {

                "<https://index.docker.io/v1/>": {

                        "auth": "NjkxMjk5MDI6S3Jpc2hAMTIz"

                }

        },

        "HttpHeaders": {

                "User-Agent": "Docker-Client/19.03.11 (linux)"

        }

}[root@naveenhost ~]#

auth key we can find it from config.json

#docker image tag naveen/nginx naveen/nginx:testing

#docker image push naveen/nginx:testing

properly tagging images

Tagging images for upload to Docker Hub

How tagging is related to image ID

The latest Tag

**Building images The Dockerfile Basics:-**

Eg:-docker build -f some-dockerfile

**package manager:-** pm's like apt and yum are one of the reasons to build containers from Debian,ubuntu,Fedora,centos

**Environment variables:-** one reason they were chosen as preferred way to inject key/value is they work everywhere, on every os and config

#docker image build -t customginx .