What/Why is Docker Compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

Let’s understand with a simple example, When you are working in the real-time industry there may be a requirement where you need to design an application that has web-tier and data-base tier and if you want to design such application using docker, you need to create two or more container as per the requirement.

So, let’s dockerize a sample word press application which has a web tier and database tier. In word press application we define instructions inside a YAML file and those steps in the YAML file give information to the web tier about the database.

**Note**: In web-based applications, the first DataBase server needs to come up before the webserver.

Steps to dockerize word press application:

* First, we need to make the database container UP and the command to make the data-base container up is:

gaurav@learning-ocean:~$ docker container run --name some-mysql -e MYSQL\_ROOT\_PASSWORD=mypassword -d mysql:5.7

* Get the IP address of data-base container using command below command from field “IP Address”

gaurav@learning-ocean:~$ docker container inspect <first 4-digit container id>

* Now, we need to run a web-tier container using the below command, suppose 172.17.0.2 is the database container's IP address

gaurav@learning-ocean:~$ docker container run --name wordpress -e WORDPRESS\_DB\_HOST=172.17.0.2:3306 -e WORDPRESS\_DB\_USER=root -e WORDPRESS\_DB\_PASSWORD=mypassword -d wordpress

* get the IP of the web-tier container as explained in Step 2.
* After getting the web-tier IP address, you can access the WordPress application by giving the IP address in the URL.

The above steps of sequence define the manual procedure to make web-tier applications up. Now let’s understand how docker-compose will make it easier and efficient.

Let’s create a **docker-compose.yaml** file as shown in below snippet:

version: "3.9"

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: somewordpress

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wordpress

MYSQL\_PASSWORD: wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

volumes:

- wordpress\_data:/var/www/html

ports:

- "8000:80"

restart: always

environment:

WORDPRESS\_DB\_HOST: db:3306

WORDPRESS\_DB\_USER: wordpress

WORDPRESS\_DB\_PASSWORD: wordpress

WORDPRESS\_DB\_NAME: wordpress

volumes:

db\_data: {}

wordpress\_data: {}

The first 12 lines explain the database and later instruction is for web-tier. Now go to the folder where the docker-compose.yaml file is present and run the command

gaurav@learning-ocean:~$ docker-compose up -d

when you hit this command it actually searches for docker-compose.yaml file in the running directory and it executes all the instructions given in YAML file one by one, it will create container application, network, and volume, based on given instruction in YAML file. Run **docker container ls** command to check whether the container came up or not, you should see two containers web-tier and database, to check network run **docker network ls**, it will show default network which will get create for word press application, to check volume run **docker volume ls,**it will show volume related to WordPress application. To delete the application which was created by the docker-compose up command run **docker-compose down** and it will delete all the created containers, network, etc except volume. If you want to delete the volume also then run

gaurav@learning-ocean:~$ docker-compose down --volume

and it will delete volume also.

# Install Docker-Compose in Linux Machine

Run this command to download the current stable release of Docker Compose:

$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

Apply executable permissions to the binary:

$ sudo chmod +x /usr/local/bin/docker-compose

 You can also create a symbolic link to /usr/bin or any other directory in your path.

$ sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

Check the docker-compose version to verify that docker-compose installed or not.

gaurav@learning-ocean:~$ docker-compose --version

docker-compose version 1.29.0, build 07737305

gaurav@learning-ocean:~$

# ****Create First docker-compose File****

To create multiple containers we need to use docker-compose.yaml file, let’s understand how a docker-compose file works and what instructions need to be defined inside it.

To run a nginx container the manual is

gaurav@learning-ocean:~$ docker container run -itd nginx

but if the same container if we want to create using docker-compose file then we need to create a YAML file using **vi docker-compose.yaml** and snippet is shown below

version: '3'

services:

webapp1:

image: nginx

ports:

- "8000:80"

**Version** :-Property is defined by the specification for backward compatibility but is only informative.

**Services:-**It defines the number of applications or containers which need to be created. As shown in the above example under webapp1 it defines the first container which will get created using **image nginx** and expose **ports “8000:80”.** By giving the image name as nginx, we are specifying the container should get built on nginx and we can give any image name as per the requirement. Ports property defines the port mapping of the host to the container, port 8000 is the host port that will map to port 80 of the container.

After creating the docker-compose.yaml file, run **docker-compose up -d** and it will create a new container .

output:

gaurav@learning-ocean:~/docker-compose$ sudo docker-compose up -d

[sudo] password for gaurav:

Creating network "docker-compose\_default" with the default driver

Pulling webapp1 (nginx:)...

latest: Pulling from library/nginx

a330b6cecb98: Already exists

e0ad2c0621bc: Pull complete

9e56c3e0e6b7: Pull complete

09f31c94adc6: Pull complete

32b26e9cdb83: Pull complete

20ab512bbb07: Pull complete

Digest: sha256:853b221d3341add7aaadf5f81dd088ea943ab9c918766e295321294b035f3f3e

Status: Downloaded newer image for nginx:latest

Creating docker-compose\_webapp1\_1 ... done

gaurav@learning-ocean:~/docker-compose$ sudo docker container ls

CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                  NAMES

2939df15d408   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:8000->80/tcp   docker-compose\_webapp1\_1

gaurav@learning-ocean:~/docker-compose$ sudo docker network ls

NETWORK ID     NAME                     DRIVER    SCOPE

0022a9f511fe   bridge                   bridge    local

b96ba06066a8   docker-compose\_default   bridge    local

3a33f83c3663   host                     host      local

e4ebd601732c   none                     null      local

As shown in the above output ,new nginx container is up and the command also shows the mapping of ports along with the name of the container. Now the Nginx application could be accessed by giving localhost and port in URL as shown below.



Docker-compose command creates the new network and docker compose down will delete all the created configuration

output:

gaurav@learning-ocean:~/docker-compose$ docker-compose down

Stopping docker-compose\_webapp1\_1 ... done

Removing docker-compose\_webapp1\_1 ... done

Removing network docker-compose\_default

gaurav@learning-ocean:~/docker-compose$ docker container ls

CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

gaurav@learning-ocean:~/docker-compose$ docker network ls

NETWORK ID     NAME      DRIVER    SCOPE

0022a9f511fe   bridge    bridge    local

3a33f83c3663   host      host      local

e4ebd601732c   none      null      local

Now, suppose we want to create two containers using docker-compose YAML file. It's very easy ,just  require one more webapp to be defined  inside YAML file as shown below

version: '3'

services:

  webapp1:

    image: nginx

    ports:

      - "8000:80"

  webapp2:

    image: nginx

    ports:

      - "8001:80"

Webapp2 is the second container which will bring up the same image as nginx(can be changed as per your choice) and the host port has changed to 8001 and now as shown below webapp2 container is up on 8001 port of host.

output:

gaurav@learning-ocean:~/docker-compose$ docker-compose up -d

Creating network "docker-compose\_default" with the default driver

Creating docker-compose\_webapp2\_1 ... done

Creating docker-compose\_webapp1\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker container ls

CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                  NAMES

4f4a2060650c   nginx     "/docker-entrypoint.…"   6 seconds ago   Up 4 seconds   0.0.0.0:8001->80/tcp   docker-compose\_webapp2\_1

e7923ed200c0   nginx     "/docker-entrypoint.…"   6 seconds ago   Up 5 seconds   0.0.0.0:8000->80/tcp   docker-compose\_webapp1\_1

gaurav@learning-ocean:~/docker-compose$ docker network ls

NETWORK ID     NAME                     DRIVER    SCOPE

0022a9f511fe   bridge                   bridge    local

a6a32f981f30   docker-compose\_default   bridge    local

3a33f83c3663   host                     host      local

e4ebd601732c   none                     null      local

gaurav@learning-ocean:~/docker-compose$ docker volume ls

DRIVER    VOLUME NAME

gaurav@learning-ocean:~/docker-compose$

## **What happens if we re-run the docker-compose file, Will it recreates the same containers again?**

The answer to this question is, docker, compose will read the yaml file instruction and it will only recreate the container if it has any changes in any of the properties.

Let’s change the host port of webapp2 to 8002 and rerun the docker-compose command

output:

gaurav@learning-ocean:~/docker-compose$ cat docker-compose.yaml

version: '3'

services:

  webapp1:

    image: nginx

    ports:

      - "8000:80"

  webapp2:

    image: nginx

    ports:

      - "8002:80"

gaurav@learning-ocean:~/docker-compose$ docker-compose up -d

docker-compose\_webapp1\_1 is up-to-date

Recreating docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$

As shown in above output only webapp2 is getting re-created because the host port for it has been changed from 8001 o 8002

Docker-compose up command always search for **docker-compose.yaml** file in the present working directory and if you change this file name docker compose command will fail as shown below:

output:

gaurav@learning-ocean:~/docker-compose$ mv docker-compose.yaml docker-compose.yaml.bkp

gaurav@learning-ocean:~/docker-compose$ ls

docker-compose.yaml.bkp

gaurav@learning-ocean:~/docker-compose$ docker-compose up -d

ERROR:

        Can't find a suitable configuration file in this directory or any

        parent. Are you in the right directory?

        Supported filenames: docker-compose.yml, docker-compose.yaml, compose.yml, compose.yaml

gaurav@learning-ocean:~/docker-compose$

## **Can we Change the docker-compose.yaml's file name?**

There may be a situation in the future where user doesn’t want to give the default name of compose file due to various reasons, the user wants to give its own file name, this could also be achieved by using -f flag of docker compose command and by specifying -f option docker will not search for default file instead it will search for file provided after -f flag.

gaurav@learning-ocean:~/docker-compose$ docker-compose -f docker-compose.yaml.bkp up -d

Creating network "docker-compose\_default" with the default driver

Creating docker-compose\_webapp1\_1 ... done

Creating docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$

## **can we write a docker-compose file in JSON or any other format ?**

Another important point to highlight is ,it is not mandatory to have o**nly yaml file format** for docker compose command, it can also work with JSON file format. Convert the yaml instruction to json and create a json file using vi docker-compose.json

{

  "version": "3",

  "services": {

    "webapp1": {

      "image": "nginx",

      "ports": [

        "8000:80"

      ]

    },

    "webapp2": {

      "image": "nginx",

      "ports": [

        "8002:80"

      ]

    }

  }

}

output:

gaurav@learning-ocean:~/docker-compose$ docker-compose -f docker-compose.json up -d

Creating network "docker-compose\_default" with the default driver

Creating docker-compose\_webapp2\_1 ... done

Creating docker-compose\_webapp1\_1 ... done

gaurav@learning-ocean:~/docker-compose$

# Docker-Compose Create / Start / Stop / Rm / Up / Down

Let’s see the basic command of docker-compose,

docker-compose --help

will list all the options available for the docker-compose command.

## **Docker-compose create**

This command is used to create a container and it will neither create the network and volume nor will start them, it will just create. As shown below output, the docker-compose create command is deprecated and it is suggested to use **docker-compose up --no-start** flag.

output:

gaurav@learning-ocean:~/docker-compose$ docker-compose create

WARNING: The create command is deprecated. Use the up command with the --no-start flag instead.

Creating docker-compose\_webapp1\_1 ... done

Creating docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker network ls

NETWORK ID     NAME      DRIVER    SCOPE

0022a9f511fe   bridge    bridge    local

3a33f83c3663   host      host      local

e4ebd601732c   none      null      local

gaurav@learning-ocean:~/docker-compose$ docker container ls

CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

gaurav@learning-ocean:~/docker-compose$ docker container ls -a

CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                   PORTS     NAMES

474f9071d846   nginx     "/docker-entrypoint.…"   16 seconds ago   Created                            docker-compose\_webapp2\_1

9cae49d2eb8e   nginx     "/docker-entrypoint.…"   16 seconds ago   Created                            docker-compose\_webapp1\_1

gaurav@learning-ocean:~/docker-compose$

So, if we run **docker-compose up --no-start** flag then it will also create the container but it will also create the default network

gaurav@learning-ocean:~/docker-compose$ docker-compose up --no-start

Creating network "docker-compose\_default" with the default driver

Creating docker-compose\_webapp1\_1 ... done

Creating docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$

## **docker-compose start**

this command will start the container as shown below

gaurav@learning-ocean:~/docker-compose$ ls

docker-compose.yaml

gaurav@learning-ocean:~/docker-compose$ docker-compose up --no-start

Creating network "docker-compose\_default" with the default driver

Creating docker-compose\_webapp1\_1 ... done

Creating docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker container ls -a

CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS    PORTS     NAMES

29dff16ec850   nginx     "/docker-entrypoint.…"   3 seconds ago   Created             docker-compose\_webapp2\_1

b23af0a13110   nginx     "/docker-entrypoint.…"   3 seconds ago   Created             docker-compose\_webapp1\_1

gaurav@learning-ocean:~/docker-compose$ docker-compose start

Starting webapp1 ... done

Starting webapp2 ... done

gaurav@learning-ocean:~/docker-compose$

## **docker-compose stop**

will stop the container

gaurav@learning-ocean:~/docker-compose$ docker-compose stop

Stopping docker-compose\_webapp2\_1 ... done

Stopping docker-compose\_webapp1\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker container ls -a

CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                     PORTS     NAMES

29dff16ec850   nginx     "/docker-entrypoint.…"   About a minute ago   Exited (0) 4 seconds ago             docker-compose\_webapp2\_1

b23af0a13110   nginx     "/docker-entrypoint.…"   About a minute ago   Exited (0) 4 seconds ago             docker-compose\_webapp1\_1

gaurav@learning-ocean:~/docker-compose$

## **docker-compose rm**

will delete the container but it will not remove the network and volume

gaurav@learning-ocean:~/docker-compose$ docker-compose rm

Going to remove docker-compose\_webapp2\_1, docker-compose\_webapp1\_1

Are you sure? [yN] yes

Removing docker-compose\_webapp2\_1 ... done

Removing docker-compose\_webapp1\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker container ls

CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

gaurav@learning-ocean:~/docker-compose$ docker network ls

NETWORK ID     NAME                     DRIVER    SCOPE

0022a9f511fe   bridge                   bridge    local

6866a6234089   docker-compose\_default   bridge    local

3a33f83c3663   host                     host      local

e4ebd601732c   none                     null      local

gaurav@learning-ocean:~/docker-compose$

## **docker-compose down**

will delete the container along with the network and it can also delete the volume if provided with --volume flag

gaurav@learning-ocean:~/docker-compose$ docker-compose up -d

Creating docker-compose\_webapp1\_1 ... done

Creating docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker container ls

CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                  NAMES

0ab41095f13c   nginx     "/docker-entrypoint.…"   7 seconds ago   Up 5 seconds   0.0.0.0:8000->80/tcp   docker-compose\_webapp1\_1

d1a042047493   nginx     "/docker-entrypoint.…"   7 seconds ago   Up 4 seconds   0.0.0.0:8002->80/tcp   docker-compose\_webapp2\_1

gaurav@learning-ocean:~/docker-compose$ docker-compose down

Stopping docker-compose\_webapp1\_1 ... done

Stopping docker-compose\_webapp2\_1 ... done

Removing docker-compose\_webapp1\_1 ... done

Removing docker-compose\_webapp2\_1 ... done

Removing network docker-compose\_default

gaurav@learning-ocean:~/docker-compose$ docker network ls

NETWORK ID     NAME      DRIVER    SCOPE

0022a9f511fe   bridge    bridge    local

3a33f83c3663   host      host      local

e4ebd601732c   none      null      local

gaurav@learning-ocean:~/docker-compose$ docker container ls

CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

gaurav@learning-ocean:~/docker-compose$

# Docker-Compose Ps/Pause/Unpause

## **docker-compose ps**

command is used to check the state of the containers.

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose*\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp*

*docker-compose\_webapp2\_*1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$

## **docker-compose pause**

the command is used to pause the container. you can see the state is pause after**docker-compose pause** command

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$ docker-compose pause

Pausing docker-compose\_webapp2\_1 ... done

Pausing docker-compose\_webapp1\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker ps

CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                       PORTS                  NAMES

aa9371ae4c98   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute (Paused)   0.0.0.0:8000->80/tcp   docker-compose\_webapp1\_1

225a77fbc32b   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute (Paused)   0.0.0.0:8002->80/tcp   docker-compose\_webapp2\_1

## **docker-compose unpause**

will start the container again and will make the container up again.

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State           Ports

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

docker-compose*\_webapp1\_1   /docker-entrypoint.sh ngin ...   Paused   0.0.0.0:8000->80/tcp*

*docker-compose\_webapp2\_*1   /docker-entrypoint.sh ngin ...   Paused   0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$ docker-compose unpause

Unpausing docker-compose\_webapp1\_1 ... done

Unpausing docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

# ****Docker-Compose Kill / Port / Exec / Run / Restart / Pull****

## **docker-compose kill**

command use to kill the container

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$ docker-compose kill

Killing docker-compose\_webapp1\_1 ... done

Killing docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command                State     Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Exit 137

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Exit 137

gaurav@learning-ocean:~/docker-compose$

## **docker-compose port**

command is use to see the mapping of container ports. Command tells about on which port of host, the container port is mapped. The below command shows that port 80 of the webapp1 container is mapped to port 8000 of the host.

gaurav@learning-ocean:~/docker-compose$ docker-compose start

Starting webapp1 ... done

Starting webapp2 ... done

gaurav@learning-ocean:~/docker-compose$ docker-compose port webapp1 80

0.0.0.0:8000

gaurav@learning-ocean:~/docker-compose$

## **docker-compose logs**

is the command to see the container logs.

gaurav@learning-ocean:~/docker-compose$ docker-compose logs

Attaching to docker-compose\_webapp1\_1, docker-compose\_webapp2\_1

webapp2\_1  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration

webapp2\_1  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/

webapp2\_1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh

.....

if you want logs continually from containers then use -f

gaurav@learning-ocean:~/docker-compose$ docker-compose logs -f

## **docker-compose exec**

the command is use to run any command inside, docker exec does not create a new container to run any command ,it will run it on running containers.

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$ docker-compose exec webapp1 ls

bin   docker-entrypoint.d   home   media  proc  sbin  tmp

boot  docker-entrypoint.sh  lib    mnt    root  srv   usr

dev   etc                   lib64  opt    run   sys   var

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$

## **docker-compose run**

docker-compose run command will create**a new container** and then runs the command inside it.

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$ docker-compose run webapp1 ls

Creating docker-compose\_webapp1\_run ... done

bin   docker-entrypoint.d   home   media  proc  sbin  tmp

boot  docker-entrypoint.sh  lib    mnt    root  srv   usr

dev   etc                   lib64  opt    run   sys   var

gaurav@learning-ocean:~/docker-compose$ docker container ls -a

CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS                  NAMES

acc6ef6c3366   nginx     "/docker-entrypoint.…"   32 seconds ago   Exited (0) 31 seconds ago                          docker-compose\_webapp1\_run\_240e22a09a9f

aa9371ae4c98   nginx     "/docker-entrypoint.…"   25 minutes ago   Up 9 minutes                0.0.0.0:8000->80/tcp   docker-compose\_webapp1\_1

225a77fbc32b   nginx     "/docker-entrypoint.…"   25 minutes ago   Up 9 minutes                0.0.0.0:8002->80/tcp   docker-compose\_webapp2\_1

gaurav@learning-ocean:~/docker-compose$

## **What is the Difference between docker-compose exec and docker-compose run Command?**

the command is used to run any command inside, docker exec does **not create a new container** to run any command, it will run it on running containers, whereas the docker-compose run command will **create a new container and then runs** the command inside it.

## **docker-compose restart**

will restart the container

gaurav@learning-ocean:~/docker-compose$ docker-compose restart

Restarting docker-compose\_webapp1\_1 ... done

Restarting docker-compose\_webapp2\_1 ... done

gaurav@learning-ocean:~/docker-compose$

## **docker-compose pull**

will pull the images(that we are using docker-compose.yaml) from the repository

gaurav@learning-ocean:~/docker-compose$ docker-compose pull

Pulling webapp1 ... done

Pulling webapp2 ... done

gaurav@learning-ocean:~/docker-compose$

# Docker-Compose Scale / Top

## **docker-compose scale**

this command is used to scale up a number of containers, let’s say if you want to create 4 containers of webapp1 and 2 containers of webapp2 then using docker-compose scale command it can be done.

gaurav@learning-ocean:~/docker-compose$ cat docker-compose.yaml

version: '3'

services:

  webapp1:

    image: nginx

  webapp2:

    image: nginx

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8002->80/tcp

gaurav@learning-ocean:~/docker-compose$ docker-compose scale webapp1=4 webapp2=2

WARNING: The scale command is deprecated. Use the up command with the --scale flag instead.

Creating docker-compose\_webapp1\_2 ... done

Creating docker-compose\_webapp1\_3 ... done

Creating docker-compose\_webapp1\_4 ... done

Creating docker-compose\_webapp2\_2 ... done

gaurav@learning-ocean:~/docker-compose$

verify using the below command :

gaurav@learning-ocean:~/docker-compose$ docker-compose ps

          Name                        Command               State          Ports

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

docker-compose\_webapp1\_1   /docker-entrypoint.sh ngin ...   Up      80/tcp

docker-compose\_webapp1\_2   /docker-entrypoint.sh ngin ...   Up      80/tcp

docker-compose\_webapp1\_3   /docker-entrypoint.sh ngin ...   Up      80/tcp

docker-compose\_webapp1\_4   /docker-entrypoint.sh ngin ...   Up      80/tcp

docker-compose\_webapp2\_1   /docker-entrypoint.sh ngin ...   Up      80/tcp

docker-compose\_webapp2\_2   /docker-entrypoint.sh ngin ...   Up      80/tcp

gaurav@learning-ocean:~/docker-compose$

As shown in the above output, a total of 6 containers are running and up. Users can configure the load balancer to distribute the load among containers.

Docker-compose top command displays the list of containers along with the number of running processes inside each container.

gaurav@learning-ocean:~/docker-compose$ docker-compose top

docker-compose\_webapp1\_1

  UID       PID    PPID    C   STIME   TTY     TIME                        CMD

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

root       27249   27195   0   06:58   ?     00:00:00   nginx: master process nginx -g daemon off;

systemd+   27421   27249   0   06:58   ?     00:00:00   nginx: worker process

systemd+   27422   27249   0   06:58   ?     00:00:00   nginx: worker process

docker-compose\_webapp1\_2

  UID       PID    PPID    C   STIME   TTY     TIME                        CMD

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

root       27741   27664   0   07:12   ?     00:00:00   nginx: master process nginx -g daemon off;

systemd+   27869   27741   0   07:12   ?     00:00:00   nginx: worker process

systemd+   27870   27741   0   07:12   ?     00:00:00   nginx: worker process

# Docker-Compose Bind Mount

Docker bind and mount provide the options to bind host port to the container port and also user can mount any of the host directories to the container as shown in the below docker-compose file.

version: '3'

services:

  webapp1:

    image: nginx

    ports:

      - "8000:80"

    volumes:

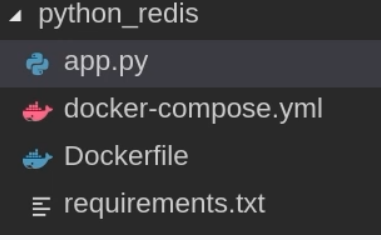
      - ./ot/:/usr/share/nginx/html/

Only 1 service is defined in docker-compose YAML file and ports property specifies the binding of host port to container port and volumes specifies the mount of host directory to the container directory i.e ./ot/ is the host directory which will get mounted to /usr/share/nginx/html of the container.

# ****Docker-compose Image, Build, Dockerfile****

## **How to build a sample project using docker-compose.yaml and Dockerfile?**

As shown below trying to build a small python\_redis project which has 3 files:



Let’s understand the simple Dockerfile, which we will use to build the project.

FROM python:3.4

ADD . /code

WORKDIR /code

RUN pip install -r requirements.txt

CMD ["python", "app.py"]

FROM: It tells that which image needs to pick to build a container. In our case, we are building a small project which has python version 3.4.

ADD: It will copy the host content from the specified directory to the container. A shown above from the present working directory copying everything to the container inside /code directory.

WORKDIR: Whenever the container comes up the default working directory will be set to the directory given in workdir.

RUN: It will install all the required modules which is mentioned in the requirement.txt file

CMD: It will run the specified command in the list inside the container .

requirements.txt file has below contant.

flask

redis

app.py file has below containent:

import os

import time

import redis

from flask import Flask

app = Flask(\_\_name\_\_)

cache = redis.Redis(host='redis', port=6379)

def get\_hit\_count():

retries = 5

while True:

try:

return cache.incr('hits')

except redis.exceptions.ConnectionError as exc:

if retries == 0:

raise exc

retries -= 1

time.sleep(0.5)

@app.route('/')

def hello():

count = get\_hit\_count()

return 'Hello World! I have been seen {} times.\n'.format(count)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0")

Now,let’s look at the docker-compose yaml file

version: '3'

services:

web:

build: .

ports:

- "5000:5000"

redis:

image: "redis:alpine"

In services property we have specified web which is our container name and build is given as dot(.) means present working directory, it searches for Dockerfile inside the current directory and build an image.

We have exposed port 5000 of the host to the 5000 port of the container. Redis is another container database container that has only images.

So, now we have all the files present in our working directory

gaurav@learning-ocean:~/python\_redis$ ls

app.py  docker-compose.yml  Dockerfile  requirements.txt

Let’s run now docker-compose up -d

gaurav@learning-ocean:~/python\_redis$ docker-compose up -d

Building web

Sending build context to Docker daemon  5.632kB

Step 1/5 : FROM python:3.4

3.4: Pulling from library/python

22dbe790f715: Pulling fs layer

0250231711a0: Pulling fs layer

6fba9447437b: Pulling fs layer

c2b4d327b352: Pulling fs layer

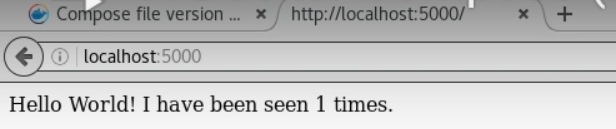
.... output removed....

Status: Downloaded newer image for redis:alpine

Creating python\_redis\_web\_1   ... done

Creating python\_redis\_redis\_1 ... done

Now,try to access application at port 5000



Every time when you reload the URL the count gets incremented and will be stored and taken care of by the Redis database which we have used.

The newly build image will also be visible in the list of images using docker image ls

gaurav@learning-ocean:~/python\_redis$ docker image ls

REPOSITORY         TAG       IMAGE ID       CREATED          SIZE

python\_redis\_web   latest    c27390e2728f   59 seconds ago   935MB

nginx              latest    ad4c705f24d3   10 days ago      133MB

mysql              5.7       1d7aba917169   2 weeks ago      448MB

ansible            latest    c1401d45b590   3 weeks ago      303MB

redis              alpine    f6f2296798e9   3 weeks ago      32.3MB

jenkins/jenkins    lts       3b4ec91827f2   7 weeks ago      568MB

ubuntu             latest    1318b700e415   7 weeks ago      72.8MB

python             3.4       8c62b065252f   2 years ago      924MB

gaurav@learning-ocean:~/python\_redis$

## **How to push this image inside your docker hub account?**

The answer to this question is that we have to change the image name like **dockerhubusername/reponame:tag**

so how can we do that ?

use the below docker-compose.yaml file

version: '3'

services:

 web:

  build: .

  image: 'coolgourav147/mypythonimage'

  ports:

   - "5000:5000"

 redis:

  image: "redis:alpine"

build the image using below command

gaurav@learning-ocean:~/python\_redis$ docker-compose build

check the output of **docker image ls** command

gaurav@learning-ocean:~/python\_redis$ docker image ls

REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE

coolgourav147/mypythonimage   latest    6237e93c0da4   13 seconds ago   935MB

python\_redis\_web              latest    c27390e2728f   6 minutes ago    935MB

nginx                         latest    ad4c705f24d3   10 days ago      133MB

mysql                         5.7       1d7aba917169   2 weeks ago      448MB

ansible                       latest    c1401d45b590   3 weeks ago      303MB

redis                         alpine    f6f2296798e9   3 weeks ago      32.3MB

jenkins/jenkins               lts       3b4ec91827f2   7 weeks ago      568MB

ubuntu                        latest    1318b700e415   7 weeks ago      72.8MB

python                        3.4       8c62b065252f   2 years ago      924MB

gaurav@learning-ocean:~/python\_redis$

# ****Docker-compose Build Args in docker-compose.yaml****

We can make the docker-compose file more generic by passing a build argument to it.

contents of app.py

import os

import time

import redis

from flask import Flask

app = Flask(\_\_name\_\_)

cache = redis.Redis(host='redis', port=6379)

def get\_hit\_count():

retries = 5

while True:

try:

return cache.incr('hits')

except redis.exceptions.ConnectionError as exc:

if retries == 0:

raise exc

retries -= 1

time.sleep(0.5)

@app.route('/')

def hello():

count = get\_hit\_count()

return 'Hello World! I have been seen {} times.\n'.format(count)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0")

contents of requirements.txt

flask

redis

contents of docker-compose.yaml. let's understand the argument along with other options in the docker-compose file.

version: '3'

services:

web:

build:

context: .

dockerfile: Dockerfile

args:

- PYTHON\_VERSION=3.4

image: python-redis-2

ports:

- "5000:5000"

redis:

image: "redis:alpine"

Context : It is used to specify the directory where the docker file resides.

Dockerfile: Specifies the name of the docker file.

Args: It is the main parameter that will specify the python version, which Dockerfile should read and use it accordingly.

Image: It is the same as explained earlier.

Docker file:

#FROM python:3.4-alpine

ARG PYTHON\_VERSION

FROM python:$PYTHON\_VERSION

ADD . /code

WORKDIR /code

RUN pip install -r requirements.txt

CMD ["python", "app.py"]

Here, in the above docker file ARG PYTHON\_VERSION is the parameter which the docker file will read from docker-compose file and based on it,it will use the python image.

# Docker-compose Networks and Volumes

Using docker-compose yaml file, we can create multiple networks and volume in an application.

The below code shows how to create networks using YAML file

version: '3'

services:

  web:

    build:

      context: .

      dockerfile: Dockerfile

      args:

        - PYTHON\_VERSION=3.4-alpine

    image: coolgourav147/python-redis

    ports:

      - "5000:5000"

    networks:

      - appnetwork

  redis:

    image: "redis:alpine"

    volumes:

      - myredisdata:/data

    networks:

      - appnetwork

  redis2:

    image: "redis:alpine"

    volumes:

      - myredisdata2:/data

    networks:

      - appnetwork2

networks:

  appnetwork:

  appnetwork2:

volumes:

  myredisdata:

  myredisdata2:

We have defined 3 containers inside services, each container is using network and volume. At the end network is the property/field which will create 2 networks appnetwork and appnetwork2,likewise, volumes are the property to create volume inside the container.Users can attach the network with any container if and only if it has been created in YAML file.

Before running the docker-compose up -d command ,there are no networks and new volumes created.

gaurav@learning-ocean:~/python\_redis$ docker container ls

CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

gaurav@learning-ocean:~/python\_redis$ docker volume ls

DRIVER    VOLUME NAME

gaurav@learning-ocean:~/python\_redis$ docker network ls

NETWORK ID     NAME      DRIVER    SCOPE

0022a9f511fe   bridge    bridge    local

3a33f83c3663   host      host      local

e4ebd601732c   none      null      local

let run **docker-compose up**

gaurav@learning-ocean:~$ docker-compose up -d

Creating network "docker-volumes\_appnetwork" with the default driver

Creating network "docker-volumes\_default" with the default driver

Building web

Sending build context to Docker daemon  5.632kB

let check network, volume and containers again

gaurav@learning-ocean:~$ docker container ls

CONTAINER ID   IMAGE                        COMMAND                  CREATED         STATUS         PORTS                    NAMES

d0df46f4f293   redis:alpine                 "docker-entrypoint.s…"   9 seconds ago   Up 7 seconds   6379/tcp                 docker-networks\_redis\_1

11f7d05376d2   coolgourav147/python-redis   "python app.py"          9 seconds ago   Up 6 seconds   0.0.0.0:5000->5000/tcp   docker-networks\_web\_1

5f9d87371ad6   redis:alpine                 "docker-entrypoint.s…"   9 seconds ago   Up 6 seconds   6379/tcp                 docker-networks\_redis2\_1

gaurav@learning-ocean:$ docker volume ls

DRIVER    VOLUME NAME

local     docker-networks\_myredisdata

local     docker-networks\_myredisdata2

gaurav@learning-ocean:~$ docker network ls

NETWORK ID     NAME                          DRIVER    SCOPE

0022a9f511fe   bridge                        bridge    local

4dd17646884a   docker-networks\_appnetwork    bridge    local

d124edf4aa4c   docker-networks\_appnetwork2   bridge    local

3a33f83c3663   host                          host      local

e4ebd601732c   none                          null      local

gaurav@learning-ocean:~$

# ****Docker-compose Variable, Environment Variable****

## **How to assign an environment variable to a container?**

we can pass environment variables also in YAML file using the environment tag as shown below.

version: '3'

services:

  web:

    build: .

    ports:

      - "5000:5000"

    environment:

      - Name=Gaurav

      - Add=Rajasthan

  redis:

    image: "redis:alpine"

Let’s rerun the compose file to check if the environment variable is updated or not inside the container.

gaurav@learning-ocean:~$ docker-compose up -d

Creating network "docker-env\_default" with the default driver

Creating docker-env\_redis\_1 ... done

Creating docker-env\_web\_1   ... done

Docker-compose exec <container\_name> env  is the command to check the environment variable in any container.

gaurav@learning-ocean:~$ docker-compose exec web env

PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

HOSTNAME=226286f7d3fd

TERM=xterm

Name=Gaurav

Add=Rajasthan

LANG=C.UTF-8

GPG\_KEY=97FC712E4C024BBEA48A61ED3A5CA953F73C700D

PYTHON\_VERSION=3.4.10

PYTHON\_PIP\_VERSION=19.0.3

HOME=/root

gaurav@learning-ocean:~$

Now, let’s suppose there are 50 environment variables in one container, it is very difficult to write all 50 variables in the YAML file and it will make the YAML file lengthy, So to overcome this problem we can define all environment variables in a file called env.txt.

so env.txt file contains below

Name=Gaurav

ADD=Jaipur

We have to modify the docker-compose yaml file by adding tag env\_file ,So that all the environment variables will get added in the container. We can give multiple env.txt files.

version: '3'

services:

  web:

    build: .

    ports:

      - "5000:5000"

    env\_file:

      - .env.txt

  redis:

    image: "redis:alpine"

 Rerun the YAML file **docker-compose up -d** and you will see using exec command that variables which were defined in env.txt file will get add-in container.

gaurav@learning-ocean:~$ docker-compose exec web env

PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

HOSTNAME=226286f7d3fd

TERM=xterm

Name=Gaurav

Add=Jaipur

LANG=C.UTF-8

GPG\_KEY=97FC712E4C024BBEA48A61ED3A5CA953F73C700D

PYTHON\_VERSION=3.4.10

PYTHON\_PIP\_VERSION=19.0.3

HOME=/root

gaurav@learning-ocean:~$

**.env** file variable also present in docker-compose by default. Users can also define the .env file to make compose file more generic. In .env file versions of application also could be mention as shown below.

PYTHON\_VERSION=3.4

REDIS\_IMAGE=redis:alpine

In Dockerfile, modify the args tag as shown below and args will read the PYTHON\_VERSION from the .env file

version: '3'

services:

  web:

    build:

      context: .

      dockerfile: Dockerfile

      args:

        - PYTHON\_VERSION=${PYTHON\_VERSION}

    image: python-redis-1

    ports:

     - "5000:5000"

  redis:

    image: ${REDIS\_IMAGE}