Introduction to Docker part 2

Note

This is the second part of the lesson on Docker. You need to understand the first part (Introduction to Docker part 1) before taking this other lesson.

That said and done, Let's get started

Table of content

1

Review and Tips

What is docker

Review and Tips

- **Docker** enables us to **create containers.**
- To use docker, we first need to create our account on hub.docker.com
- Secondly, we need to download and install docker on our system with yum
- Always remember that when you install **Docker**, you need to check if **the daemon is up and running**: **# systemctl status docker**
- Enable the docker daemon for it to start when the server is booted up

Review and Tips

- To run a docker container, we need to pull a **Docker image** that will be install in that container
- The **Docker images** come from the **Docker Hub**
- The images on **Docker hub** can be Official ones from companies like **Centos, Ubuntu** etc.
- The **images can also be built and pushed on Docker hub by individuals** in their **repositories** or **registry**

Review and Tips

- To display all the images we have on our system, we run the command:
  - **$ docker images**
- To list the various containers we have on the system, we can run
  - **$ docker ps** or **$ docker ps -a**
- To run a container with and interactive shell in detached mode, we use the image ID in the command:
  - **$ docker run -itd imageID bash**
- We have many other docker commands we can use: docker info, docker exec, docker attach, docker stop, docker rm, docker rmi etc…

The /var/lib/docker directory

What does it contain?

Review and Tips

- And important thing you need to know is that, when you install **Docker**, **a path is created on your system**. That is the **/var/lib/docker**
- You can **cd** in there and check its content
  - **$ sudo ls /var/lib/docker**
- If you are logged in as user student you might get a **permission denied** or and error: **student is not in the sudoers file**
- To solve this, you need to add user student to the sudoers file or to the wheel group with the commands:
- **$ exit** then **$ sudo usermod -aG wheel student** (here we are logged in as user vagrant)

Review and Tips

- Now, switch again to user student: $ sudo su student
- List the content of /var/lib/docker: sudo ls /var/lib/docker
- You may need to enter the password for user student (school1)
- In that path, you can see **subdirectories** like: **containers, image, network, volumes** etc..

```
[student@localhost vagrant]$ sudo ls /var/lib/docker
[sudo] password for student:
buildkit    image    overlay2  runtimes  tmp    volumes
containers  network  plugins   swarm     trust
```

Review and Tips

- If some of these directories are deleted, then **you may get some issues cause some stuff will no more work.**
- You can easily understand that:
  - **Images** are stored in the **image directory**
  - **Containers** are stored in the **containers directory**
  - **Network** informations are stored in the **Network folder**
  - **Etc.**

Docker network interfaces

Review and Tips

- Another important thing you need to know is that when Docker is install on your system, running the **ifconfig** command **will display more network interfaces than usual**
- Hence, the **first network interface** you see will generally be **docker0**
- This interface is the one dealing with all the **Docker containers** you will create on this host machine (ie the machine on which we have installed Docker)
- **The docker interface has its own little network**. The IP address here is **172.17.0.1** and the subnet mask is **255.255.0.0**

```
[root@localhost vagrant]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        inet6 fe80::42:c2ff:fe3d:e53a  prefixlen 64  scopeid 0x20<link>
        ether 02:42:c2:3d:e5:3a  txqueuelen 0  (Ethernet)
        RX packets 2945  bytes 127202 (124.2 KiB)
```

Questions:

1. What is the class of this IP address?
2. How many addresses can be available here?

NetID and HostID in docker containers

Review and Tips

- The docker's network is set up to build **thousands of containers!**
- All our containers will have the same **NetID** (the two first octets) and only the **HostID** will be changing for each container
- Let's check that out!

  Run a container for **httpd** in **detached mode**

  - **$ docker run -d --name=web1 -p 81:80 httpd**

NB: Make sure the specified **name (web1)** and the **port (81)** are not yet used by another container. If that they are already used, just modify it or continue with that container.

Review and Tips

- The **docker ps** command will show that there is a new container created

```
[student@localhost vagrant]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                   NAMES
35a9a51bc3d1   httpd     "httpd-foreground"       27 seconds ago   Up 23 seconds   0.0.0.0:81->80/tcp, :::81->80/tcp       web1
9cac82d010cb   httpd     "httpd-foreground"       4 hours ago      Up 4 hours      0.0.0.0:85->80/tcp, :::85->80/tcp       web
```

- Let's access it from the browser to make sure it works!
- Check your IP address with **$ ifconfig** (Take the server's address in eth1: mine here is **192.168.56.37**)
- Open the browser and paste the IP address with the port number: 192.168.56.37:81
- We get the message **"It works!"**

Review and Tips

- Now, let's check if the IP address of that container starts with **172.17** as we said earlier (ie the **NetID does not change**).
- To do that, we use the command: **# docker inspect ContainerID**
- **Example:** **# docker inspect 35a** (35a here are the first 3 characters of my containerID when I run **docker ps**)
- This displays a whole bunch of informations about our container. Just check the **IP address**. Mine is **172.17.0.3**

```
"Gateway": "172.17.0.1",
"IPAddress": "172.17.0.3",
"IPPrefixLen": 16,
"IPv6Gateway": "",
```

- You can scroll down to get in touch with the container's informations

Review and Tips

- If we run another container, the **HostID** of the IP address will go to **0.4 , 0.5 ...**
- Let's run another httpd container
  - **$ docker run -d --name web2 -p 82:80 httpd**
- Remember, we can't use the same ports nor the same name while running another container.
- The name must be **unique** because the system has a **DNS record** of all the created containers.
- If two containers have the same name, **that will bring confusions**

Review and Tips

- You can inspect the second container created to check its **IP address**

  # docker inspect dac (here are the first 3 characters of my container ID)

- The **IP address HostID** for this container is **0.4**

```
"Gateway": "172.17.0.1",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"IPAddress": "172.17.0.4",
```

2

Mounting Volumes in container

Why should we do that?

Mounting volumes in a container

- When a container is in use, **data collected from the users of that container are stored in there**
- How can you get that data since **the container is an isolated entity on the system?**
- Now, **if someone deletes a container on your system, then that container will go with all that data**. **We Can't let that happen in the enterprise environment !**
- To avoid this, we mount volumes inside the containers

u train

Mounting a volume in our container means to connect a folder, from our host to a folder inside the container so that the data collected in the container might be store on the host server as well

The docker volume command

Docker volume

- We can use the docker volume command to create and manage volumes in our containers
- To create a volume, we use: $ **docker volume create volumeName**
- To list volumes, we run: $ **docker volume ls**
- To inspect a volume, we use: $ **docker volume inspect volumeName**
- To remove a volume, we run: $ **docker volume rm volumeName**
- The volumes are created in the **/var/lib/docker/volumes** folder

Docker volume

Example: Let's create and manage a docker volume called **my-vol**

- Create the volume: $ **docker volume my-vol**
- Let's start a httpd container with that volume: (choose one way -v or --mount)
  - Using the **-v** flag

```
$ docker run -d --name=web3 -p 84:80 -v my-vol:/usr/local/apache2/htdocs httpd
```

  - Using the **--mount** flag (recommended)

```
$ docker run -d --name=web4 -p 86:80  --mount source=my-vol1,target=/usr/local/apache2/htdocs httpd
```

Docker volume

- If you start a container with a volume that does not yet exist, Docker creates the volume for you.
- You can inspect the volumes created with $ **docker volume inspect my-vol**

```
[student@localhost html]$ docker volume inspect my-vol
[
    {
        "CreatedAt": "2022-02-23T14:13:33Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/my-vol/_data",
        "Name": "my-vol",
        "Options": null,
        "Scope": "local"
    }
```

- List the volumes with $ **docker volume ls**

```
[student@localhost html]$ docker volume ls
DRIVER    VOLUME NAME
local     my-vol
local     my-vol1
```

Docker volume

- Let's access the web3 container from the browser using the IP address and the port number: **192.168.56.37:84**
- You will get the page **it works!**
- Now let's modify that content from our server in the **index.html** file located at **/var/lib/docker/volumes/my-vol/_data/index.html**
  - **$ sudo vi /var/lib/docker/volumes/my-vol/_data/index.html**
  - Replace the It works! With Good Job!!
- Save and exit the file
- Refresh your browser's page and check what you get

Create containers using the same volume

Docker volumes

- You can **create many containers pointing to the same volume. That is, the containers will have the same content and behaviour**
- Let's create another container here with the volume my-vol

```
$ docker run -d --name=web5 -p 88:80 -v my-vol:/usr/local/apache2/htdocs httpd
```

- Let's access it in the browser with the IP address and the port number

  192.168.56.37:88

- You can realize that this **container is similar to the web3 container**

What happens if we remove a container created with a volume?

Docker volumes

- Remember you need to stop a container before removing it. You can also use the -f flag to force the removal
- We can **remove many containers at the same time: $ docker ps**

```
[student@localhost html]$ docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED          STATUS          PORTS                                   NAMES
e7ff9ad4ce04   httpd     "httpd-foreground"   33 minutes ago   Up 33 minutes   0.0.0.0:88->80/tcp, :::88->80/tcp       web5
28014d2145e7   httpd     "httpd-foreground"   6 hours ago      Up 6 hours      0.0.0.0:86->80/tcp, :::86->80/tcp       web4
c3aa97d557aa   httpd     "httpd-foreground"   6 hours ago      Up 6 hours      0.0.0.0:84->80/tcp, :::84->80/tcp       web3
5aa92629b110   httpd     "httpd-foreground"   7 hours ago      Up 7 hours      0.0.0.0:83->80/tcp, :::83->80/tcp       quirky_mayer
dac52b06c62b   httpd     "httpd-foreground"   7 hours ago      Up 7 hours      0.0.0.0:82->80/tcp, :::82->80/tcp       web2
35a9a51bc3d1   httpd     "httpd-foreground"   7 hours ago      Up 7 hours      0.0.0.0:81->80/tcp, :::81->80/tcp       web1
9cac82d010cb   httpd     "httpd-foreground"   11 hours ago     Up 11 hours     0.0.0.0:85->80/tcp, :::85->80/tcp       web
```

- Let's stop the containers: $ **docker stop e7f 280 c3a 5aa dac 35a 9ca**
- Let's delete the containers: $ **docker rm e7f 280 c3a 5aa dac 35a 9ca**

Docker volumes

- You can realize that even though our containers are all deleted, **the web content is still in the /home/student/html folder on our host**
- Our **index.html** file is saved

```
[student@localhost html]$ cd /home/student/html
[student@localhost html]$ ls
index.html
[student@localhost html]$ cat index.html
<h1> This is my first docker httpd content delivery </h1>
```

- **Thus, we can easily create another container and mount the web content volume to it**
- It will work just the same as our previously deleted container

3

Build a Docker image

How can one build a Docker image?

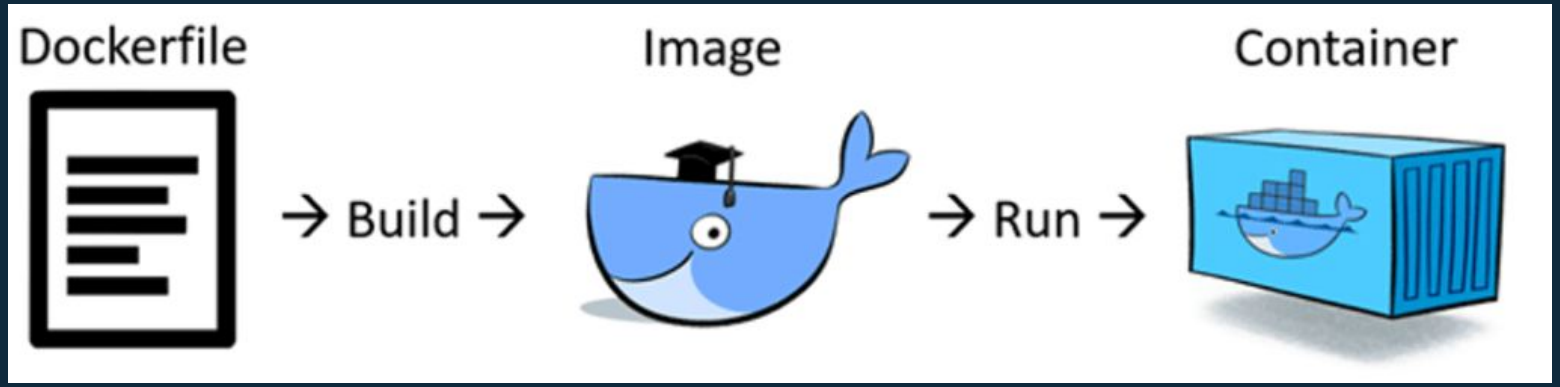
Build a Docker image

We have been using images from the Docker Hub

- What if we are not satisfied with the image we see there?
- What if an image is not exactly what we are looking for?
- What if we want something else that is not from Docker hub?
- How can we build our own Docker image?

Build a Docker image

The process



Build a Docker image

- To build our own **Docker image**, we need a **Docker file** in which we are going to put **our image building instructions**
  - Open the browser and go to the **Docker hub** (**hub.docker.com**)
  - Sign in with your credentials (**username and password**)
- Search for **httpd** and read the instructions on **how to build an image from it**
- **That said and done, let's go into practice**

Build a Docker image

- In your home directory (**/home/student** for me), create a new file called **Dockerfile**
  - **$ cd ~**
  - **$ touch Dockerfile** (create it with capital D)
  - **$ vim Dockerfile**
- Write the following content in the file:

  FROM httpd:2.4

  MAINTAINER student

  RUN apt-get update

  COPY ./index.html /usr/local/apache2/htdocs
- **Save this and quit**

Build a Docker image

- Now, create the **index.html** file in your **home directory**

  $ vim index.html

  <h1>This is my Docker image of httpd</h1>

- Save and exit

- Now, we can build that image by running:

  $ docker build -t student-httpd .

- **student-httpd** is just the name I want for my image
- Don't forget the **.** (dot at the end)
- You can check if the image was created: **docker images**

Build a Docker image

- We see that a new image was created with the TAG **latest.**

```
[student@localhost ~]$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
student-httpd   latest    0944b905414d   17 seconds ago   161MB
ubuntu          latest    54c9d81cbb44   3 weeks ago      72.8MB
httpd           2.4       a8ea074f4566   4 weeks ago      144MB
httpd           latest    a8ea074f4566   4 weeks ago      144MB
```

- That is because we did not precise the tag (the version) while building the image. We can do that with: $ **docker build -t student-httpd:1.0 .**

  (the dot '.' at the end simply signifies the current directory)

- Now, in **docker images** we will see the tag **1.0**

Exercise: Build another image with the **tag 2.0**

Build a Docker image

- Let's run one of our images to see if it is working!

  $ docker run -d --name apache2 -p 89:80 student-httpd

- If you run **docker ps** , you will see **a new container running**
- You can access it from the browser with your **IP address** and the port number as usual: **192.168.56.37:89**
- **Remember to refresh the page!**

Build a Docker image

- Let's add some lines in our **index.html** file

  $ vim index.html

  <h1>Learning docker is a must and I am enjoying it</h1>

- Let's build another image with the version 3.0

  $ docker build -t student-httpd:3.0 .

  $ docker images to check if the image is created

Build a Docker image

- Let's **run a container** with our new image and access it from the browser with the **IP address and the port number**
  - **$ docker run -d --name apache1 -p 90:80 student-httpd:3.0**
- In the browser **192.168.56.37:90**

This is my Docker image of httpd

Learning Docker is a must and I am enjoying it

4

Push a Docker image

How can one push a Docker image to docker hub?

Push a Docker image

- To do that, let's login to our **docker hub** account right in the **Terminal**
  - **$ docker login** then enter the **username** and the **password** of your docker account if necessary
- Now, we use the docker tag command to create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

$ docker tag YourimageName yourRepositoryName/imageName:tag

Exemple: (Put your own **Docker ID** at the place of **kserge2001)**

$ docker tag student-httpd:3.0 kserge2001/serge-httpd:1.0

- You can run **docker images**. You will see a new image in your repository

Push a Docker image

- Now, let's push it to docker hub. The syntax is:

  docker push username/imageName:tag

- **Exemple: docker push kserge2001/serge-httpd:1.0**
- When the process is done,
  - Open your account on **hub.docker.com**
  - Click on your **Repositories**
  - You should see the **new image you pushed in there**
- **Now, anybody in the whole world can access that image and pull it on their system to run a container**

Build an image from scratch

That is only using the OS version

Push a Docker image

Let's do that from a **CentOS version (CentOS 6.10)**

Your Dockerfile should look like this:

```
FROM kserge2001/centos-ssh

MAINTAINER student

RUN yum update -y

RUN yum install httpd -y

RUN service httpd start

EXPOSE 80

COPY ./index.html /var/www/html

CMD apachectl -D FOREGROUND
```

EXPOSE is used to open a port

CMD is used to precise the program that will run in the container when it is spun up. (we don't want the container to exit)

Save and quit!

Push a Docker image

- Let's build that image: **$ docker build -t kserge2001/web-serge .**
- It might take some time to build
- If you run **docker images**, you will see **a new image in the list**
- Now, let's add some content in the **index.html** file to make a difference

  $ vim index.html

  <h1>This is an image built from scratch</h1>
- **Save and quit**
- Build the image once more: **$ docker build -t kserge2001/web-serge .**

Push a Docker image

- Check it in **docker images**

```
REPOSITORY                TAG                 IMAGE ID
kserge2001/web-serge      latest              6fa97b4afdc0
```

- Let's modify the index file here: vim index.html

  <h1>This is the one built from scratch </h1>

- Let's build that image:

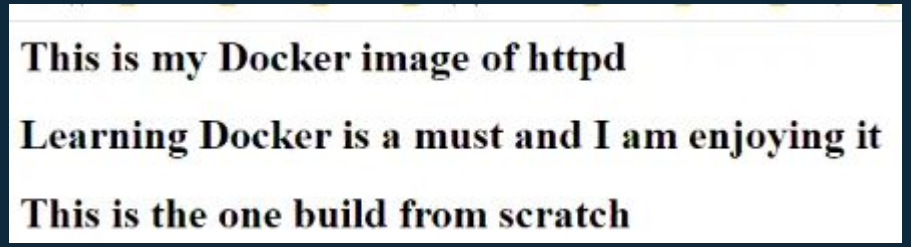
  $ docker build -t kserge2001/web-serge .

Push a Docker image

- Now, run the container with that image ($ **docker images** to check)

  docker run -d -p 88:80 kserge/web-serge

- The **docker ps** command will show a new container running
- You can access it from your browser **192.168.56.37:88**



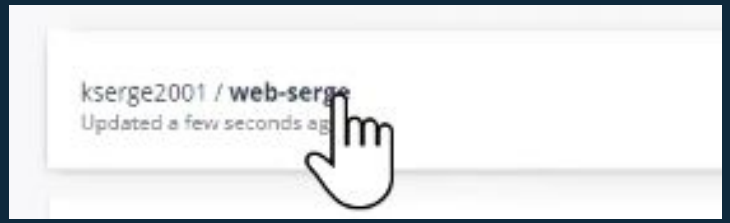
- You can run containers with other images we built and access them from the browser to check what will be displayed

Push a Docker image

- Let's push that image on the **docker hub**:

  docker push kserge2001/web-serge

- Check the **Repositories** on your account (refresh the page)



- **Exercice: Check how to build an ubuntu 18.04 image from scratch**

These concept might seem confusing but you will better understand with research and practice.

Play with this and don’t forget to clean up the images and containers when you are done.

see you guys in the next lesson!

Thanks!

Any questions?

You can find us at:

website: http://utrains.org/

Phone: +1 (302) 689 3440

Email: contact@utrains.org

Click on the link below to contact the support team for any issue!

utrains.org/support/

Create a ticket for your problem and we will get back to you soon!