**Docker - Building a Web Server Docker File**

Let’s see how we can build a web server image which can be used to build containers.

In our example, we are going to use the Apache Web Server on Ubuntu to build our image. Let’s follow the steps given below, to build our web server Docker file.

**Step 1** − The first step is to build our Docker File. Let’s use **vim** and create a Docker File with the following information.

RUN apt-get update

RUN apt-get install –y apache2

RUN apt-get install –y apache2-utils

RUN apt-get clean

EXPOSE 80 CMD [“apache2ctl”, “-D”, “FOREGROUND”]

The following points need to be noted about the above statements −

* We are first creating our image to be from the Ubuntu base image.
* Next, we are going to use the RUN command to update all the packages on the Ubuntu system.
* Next, we use the RUN command to install apache2 on our image.
* Next, we use the RUN command to install the necessary utility apache2 packages on our image.
* Next, we use the RUN command to clean any unnecessary files from the system.
* The EXPOSE command is used to expose port 80 of Apache in the container to the Docker host.

**Step 2** − Run the Docker **build** command to build the Docker file. It can be done using the following command −

sudo docker build –t=”mywebserver”

**Step 3** − Now that the web server file has been built, it’s now time to create a container from the image. We can do this with the Docker **run** command.

sudo docker run –d –p 80:80 mywebserver

The following points need to be noted about the above command −

* The port number exposed by the container is 80. Hence with the **–p** command, we are mapping the same port number to the 80 port number on our localhost.
* The **–d** option is used to run the container in detached mode. This is so that the container can run in the background.

**Docker - Building Files**

The Docker File can be built with the following command −

docker build

**Docker build**

This method allows the users to build their own Docker images.

Syntax

docker build -t ImageName:TagName dir

**Docker - Networking**

Docker takes care of the networking aspects so that the containers can communicate with other containers and also with the Docker Host. If you do an **ifconfig** on the Docker Host, you will see the Docker Ethernet adapter. This adapter is created when Docker is installed on the Docker Host.

## IFCONFIG

## **Listing All Docker Networks**

This command can be used to list all the networks associated with Docker on the host.

Syntax

docker network ls

**Docker - Compose**

Docker Compose is used to run multiple containers as a single service. For example, suppose you had an application which required NGNIX and MySQL, you could create one file which would start both the containers as a service without the need to start each one separately. how to get the docker-compose version.

sudo ./docker-compose -version

**Docker – Storage**

**Storage Drivers**

Docker has multiple storage drivers that allow one to work with the underlying storage devices. The following table shows the different storage drivers along with the technology used for the storage drivers.

|  |  |
| --- | --- |
| **Technology** | **Storage Driver** |
| OverlayFS | overlay or overlay2 |
| AUFS | aufs |
| Btrfs | brtfs |
| Device Manager | devicemanager |
| VFS | vfs |
| ZFS | zfs |

Let us now discuss some of the instances in which you would use the various storage drivers −

**AUFS**

* This is a stable driver; can be used for production-ready applications.
* It has good memory usage and is good for ensuring a smooth Docker experience for containers.
* There is a high-write activity associated with this driver which should be considered.
* It’s good for systems which are of Platform as a service type work.

**Device mapper**

* This is a stable driver; ensures a smooth Docker experience.
* This driver is good for testing applications in the lab.
* This driver is in line with the main Linux kernel functionality.

**Btrfs**

* This driver is in line with the main Linux kernel functionality.
* There is a high-write activity associated with this driver which should be considered.
* This driver is good for instances where you maintain multiple build pools.

**Ovelay**

* This is a stable driver and it is in line with the main Linux kernel functionality.
* It has a good memory usage.
* This driver is good for testing applications in the lab.

**ZFS**

* This is a stable driver and it is good for testing applications in the lab.
* It’s good for systems which are of Platform-as-a-Service type work.

To see the storage driver being used, issue the **docker info** command.

Syntax

docker info