Docker

Md. Sabbir Sattar Mukit

October 12, 2022

1 Why docker

Deployments are way too much complicated these days. To deploy a project we have to manage too many languages, frameworks, different kind of databases, dependencies, dev environments, prod, QA and staging.

Suppose, you are making an application using node.js/python. Specific version of node is needed for that.If you work with team other team members should have same sort of environment setup. In this regard docker will help us.Every one shouln't install all of those versions, dependencies, databases.With the help of docker file everyone will be able to run the project in his/her own machine.

We can't install multiple versions of node/python in our local environment at the same time. But sometimes doing different kind of projects we should have those versions/dependencies/databases needed for running a project. Docker file helps us in this regard.Once you make a docker file for your project, sharing this with team member will help them to run the project.

Let's think about the complexities for running different projects in your environment. What if your every team members have to same setup to run every project. Everyone have to maintain same kind of setup for running it. And changing versions/dependencies/databases it too much time consuming. Suppose, you need python version of 3.8.10 for a project. 10 person working on that project every one have to same version installed in their environment.

Docker helps us to solve those problems describe in the top. Docker used for dockering different containers into it and send it to others. Lets, think about the containers which are used to deliver different products. Packing all the product into it and send it to the destination. Docker works like this concept.

2 Docker Installation

1. Install docker desktop from this link : https://docs.docker.com/get-docker/
2. Go to this link and follow the instructions:
https://learn.microsoft.com/en-us/windows/wsl/install-manual

3.Also create an account in docker hub from this link :
https://hub.docker.com/

3 Docker Hub

Docker Hub is a registry service on the cloud that allows you to download Docker images that are built by other communities. You can also upload your own Docker built images to Docker hub. To use docker hub: 1. Go to : https://hub.docker.com/ this website. 2. Singnup to this website. 3. Now you can install various images that are build by other community.

4 Docker Image

Docker Images are like blueprint for containers. Docker image contains different runtime environment, application code, any dependency that needed to run the project, extra configurations like env variables and different commands. Infact images are readonly, once you build an image you can't change it. To make changes of any image , someone need to create a brand new image.

5 Docker Base Image and Parent Image

When you create a new Docker image, you start with a base image that provides the basic environment for your application to run. The base image contains the operating system, libraries, and other software components that are necessary for running your application. You can then add layers to the base image to customize it for your specific needs.

The term "parent image" is often used to describe the relationship between the base image and the new image that you are creating. The parent image is the image that the new image is based on, and it serves as the foundation for the new image.

For example, if you want to create a Docker image for a Python web application, you might start with a base image that contains the Python runtime and libraries that your application requires. This base image would be your parent image for the new image that you are creating. You would then add your application code and any additional dependencies to the parent image to create the final Docker image. Parent image reffers to the image that includes the basic environment for your application. Parent image is not constant as it can be inherited by others.As the inherited image may have some extra characteristics like libraries, runtime environments that are necessary for running the application.

6 Getting Parent Image

Go to dockerHub. DockerHub is the online repository of parent images. It contains premade parent images. We can download images from this repository and use then in our project. Suppose, a project needs to run in node environ-

ment. So we need to have a node as parent image. To download this image from dockerHub run following command in your command prompt:

- docker pull node (specify your version you needed)

7 Containers

Containers are runnable instance of an image.When we run an image it creates a container. To run our application runnable instance of an image will help us.So container is a process which runs our application as outlined by the image we created.

8 Basic Coomands

FROM – Defines the base image to use and start the build process.

RUN – It takes the command and its arguments to run it from the image.

CMD – Similar function as a RUN command, but it gets executed only after the container is instantiated.

ENTRYPOINT – It targets your default application in the image when the container is created.

ADD – It copies the files from source to destination (inside the container).

ENV – Sets environment variables.

COPY –from : This command is used in Docker to copy files or directories from a specific image or container to a new Docker image being built.

COPY –from=¡image/Container ID¿:¡tag¿ ¡src-path¿ ¡dest-path¿

9 Process of Building a Dockerfile

At first you need to add Docker extention to your text editor. Then create a file named **Dockerfile** in the root directory of your project and follow the steps given below:

1. **Choose a Parent Image** : Start by choosing a base image that will serve as the foundation of your Docker image. This image will provide the operating system and other basic software components needed for your application to run.

For adding base image add this command in the tiop of the dockerfile:
From parent image name

2. **Define the working directory**: Specify the working directory where your application code will reside within the container.

Use the following command to define working directory.
WORKDIR /app

This working directory reffers to directory of container from where all instraction will execute.

3. **Install dependencies**: Install any dependencies your application requires to run, such as libraries, frameworks, or packages.
Use the following command to install dependencies:
RUN packageName;
This package name can be varied according to projects.
For python project:
RUN pip install -r requirement.txt/ pip install package name
For node project:

RUN npm install

4. Copy application code : Copy the application code into the container's working directory.

Copying source code : **COPY . .** Here, first . reffers to source directory you want to copy and second . reffers to destination directory where all files will be coppied. First one is for local folder of your project you wanted to copy. Use . if it's in the same directory of dockerfile. Use ./fileName/fileName. For destination which is in dockerimage, give your destination location.

5.Expose ports: If your application requires network connectivity, specify the ports that need to be exposed.

EXPOSE 80
Make port 80 available to the world outside this container.

6. Define environment variables: Set any environment variables that your application needs to run.

ENV NAME World
7. Define the command:
Specify the command that will be run when the container is started. This could be the entry point for your application or a script that starts your application.
CMD ["python", "app.py"]

Here's an example of dockerfile:

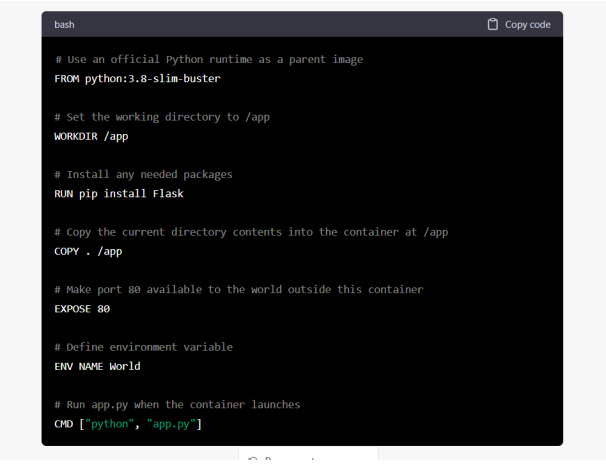

Figure 1: An image of Dockerfile

10 Run a docker project

– docker build -t imagename .
This will create an image in yout docker desktop application. To run this image run the following command :
–docker run –name containerName -p 8000:5000 imageName

11 Docker Ignore file

Actually we add .dockerignore file to reduce redundency . Like while we run a node project it creates a node modules folder which we need not to run every time we run our project. We add such kind of file in dockerignore file same we do in gitignore.

12 Docker Volumes

The purpose of using Docker volumes is to persist data outside the container so it can be backed up or shared.

Docker volumes are dependent on Docker's file system and are the preferred method of persisting data for Docker containers and services. When a container is started, Docker loads the read-only image layer, adds a read-write layer on top of the image stack, and mounts volumes onto the container filesystem.

– docker run (always run a image into a new container)
– docker start (run a container which is already build)

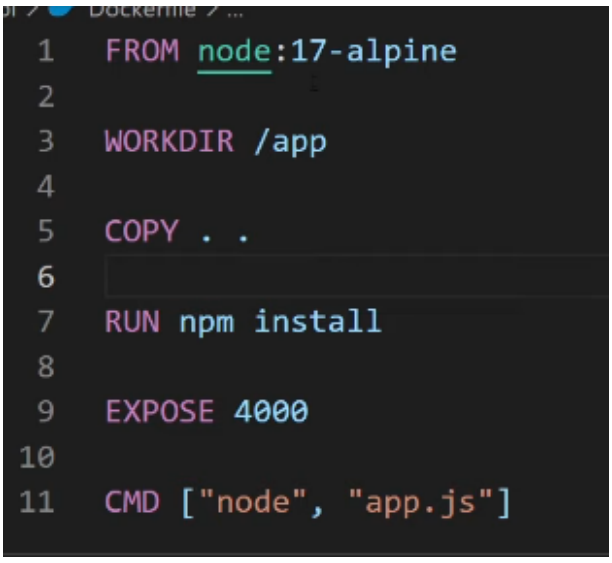

Figure 2: An image of Dockerfile

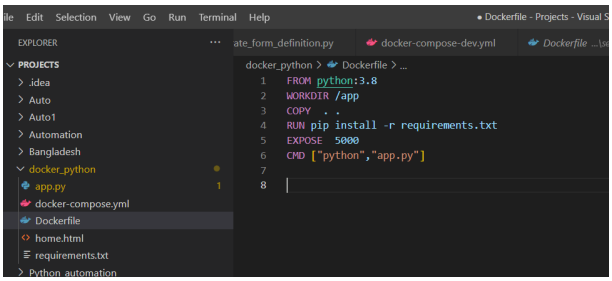

Figure 3: An image of Dockerfile (python project)

– docker stop c (to stop the container)

Volumes always to specify folders on our host computer than can be available to run a container. We can map those folders on our host computer to specific folders inside the container. If we change something on our mapped folders of host computer that will be reflected to the folders of containers.

* we can't app the whole source code of our project.

Using docker volume we can have the host computer's changes and running that containers our project will run with the latest changes. To have all those changes made in host computer we should mapp all those folders with the container. In this process our docker image will not be updated. To update our image we should run our image to another container and rebuild it.

Infact volumes can be used to directory mapping between the project and the container. If something changes to the project, it will reflected to container. After starting this container again we will get all those changes made to our mapped folders.

Implementation of docker volume:

After detecting changes inside containers needs to restart the server. Something that detect changes of our project and restart the server automatically needs to install that globally. After the parent image we should install that package. For example nodemon is a package that detect changes for node project and restart the server automatically.

changes made in host computer should map with the container, If folders inside container changes , after starting it , it will get all changes that made and restart the server.

As our project folder mapped to our container, after deleting files which didn't need to our host computer, it deletes it from the container also.

13 Docker Compose Commands

A Docker Compose file is used to define and run multi-container Docker applications. It's a YAML file that specifies the services, networks, and volumes for your application.

1. version: specifies the version of the Compose file syntax.
2. services: defines the different containers and their configurations.
3. networks: defines the networks that the containers will use to communicate with each other.
4. volumes: defines the volumes that the containers will use to store data.
5. build: specifies the build configuration for the container.
6. image: specifies the Docker image to use for the container.
7. ports: specifies the ports to expose on the container.
8. environment: sets environment variables for the container.

Command section for a service of a docker compose file :

The command section in a Docker Compose file is used to specify the command that should be run when a container based on the service is started. This is typically the main command that the container should run, such as starting a server or running a script.

services:
web:
build: .
ports:
- "8080:8080"
command: python app.py

In this example, the web service is defined with a build configuration and exposed ports, and the command section specifies that the python app.py command should be run when the container starts.

stdin open in docker compose:
The stdin open option in a Docker Compose file is used to open the standard input (stdin) of the container. This option is typically used for interactive applications that require user input or for debugging purposes.

When stdin open is set to true, it allows you to interact with the container's stdin by attaching to the container's console. This means that you can type commands or input directly into the container's console as if you were running the application locally.

services:
web:
build: .
ports:
- "8080:8080"
$\text{stdin}_o pen : true$
$tty : true$
$command : pythonapp.py$

14 Docker Compose

Docker compose contains all the different configurations of the containers(like front end, backend , database) of our project.

1. create docker-compose.yaml file.
2. specify the version of docekr –¿ version: "3.8"
3. services:

api:(frontend/backend/db)

build: ./api (project root directory where docker image created)
containerName: CN(give any name where image will run)
port:
- '4000:4000'

volumes:
- ./api(project directory):/app(directory that processed by docker image)
- ./app/nodeModules.
4. Run a docker compose file :
To cleanup all images, volumes and containers run : docker system prune.

— go to root directory of project.
–run : docker-compose up

– run: docker images (it shows all the images created)

– docker ps (it shows all running containers)
– docker-compose down
–rmi all -v(removes all images, containers and volumes)

15 Links in Docker Compose:

In a Docker Compose file, you can specify links between containers using the links keyword. The links keyword allows you to specify a list of container names that the current container should have access to.

Here's an example of how to use links in a Docker Compose file:

services:
web:
build:
. links:
- db
db:
image: postgres

In this example, we have two services: web and db. The web service is built using the current directory as the build context, and we've specified that it should be linked to the db service.