DockerAndContainers.md

Docker and Containers

Commands that was used in the Docker and Containers Session conducted by DSC-IIEST facilitated by Arnab Sen.

Exploring the Docker

Basics of `docker` command

```
docker -h
```

Gives an idea of all the commands available. `docker` has some management commands like `container` , `image` . Now run:

```
docker container -h
```

This will show all the `docker container` related commands.

Hello World of Docker

Run the command,

```
docker container run hello-world
```

It will pull `hello-world:latest` from Dockerhub where `:latest` refers to the version of the image that was pulled, since we didn't specify any particular version it automatically chose the latest version.

See all the images

```
$ docker images
# OR
$ docker image ls
```

You will see the `helllo-world` image, once again run the container this time it won't download any image, cause the image now exists locally.

To understand how the image has all the configuration run the `docker image inspect` command.

```
docker image inspect hello-world
# or
docker inspect hello-world
```

List all containers

```
docker ps
```

This will show all the containers that are currently running, to see all the containers that were exited run.

```
docker ps -a
```

Hello World with Ubuntu

First we can pull the image

```
docker image pull ubuntu
```

Now, we will run the container and specify the command:

```
docker container run ubuntu echo "Hello World"
```

To check if it was indeed generated from the ubuntu machine we can run some OS specific commands like:

```
docker container run ubuntu cat /etc/os-release
```

Interact with the container

Keep the container up for sometime by using the sleep command.

```
docker run ubuntu sleep 10
```

This will make the container sleep for 100 seconds. Since we couldn't interact with the container we will now run the container in detached mode.

```
docker run -d ubuntu sleep 300
```

Now to execute a command in the container we will use the `exec` command.

```
docker exec <contianer-id> <command>
```

To get an interactive shell we do

```
docker exec -i <container-id> /bin/bash
```

> -i for the interactive mode

To stop a container we will use the `stop` command.

```
docker container stop <container-id>
```

Dockerizing an application

Repository clone

Get the source code of the application we are about to Dockerise.

```
git clone https://github.com/dsc-iiest/sample-express-server.git
cd sample-express-server
```

Dockerfile

Create a Dockerfile

```
touch Dockerfile
vim Dockerfile
```

Now first we specify a base image.

```
FROM node:alpine
```

Now, copy the files from the current directory into the app directory in the container

```
COPY . /app
```

Let's set the WORKDIR to /app so that it we don't have to specify /app everytime

```
WORKDIR /app
```

Now the command to install dependencies

```
RUN npm install
```

After that we have to specify a CMD that will run everytime we start our container

```
CMD ["npm", "start"]
```

This is how your Dockerfile will look

```
FROM node:alpine
COPY . /app
WORKDIR /app
RUN npm install
CMD ["npm", "start"]
```

Source Link

OR run

```
echo -e "FROM node:alpine\nCOPY . /app\nWORKDIR /app\nRUN npm install\nCMD [\"npm\", \"start\"]\n" > Dockerfile
```

Image from Dockerfile

Now to build an image from the Dockerfile we will use the `docker build` command.

If the Dockerfile is in the same directory we can avoid the `-f` tag. We will use a `-t` tag to name the image

The final command will look like:

```
docker build -t express-app .
```

Container using the image

Now let's start a container from this image.

```
docker run -p5001:5000 --name backend-app express-app
```

`-p` to expose ports. Port 5000 of the container will be mapped to port 5001 of the host machine `--name` is the name of the container

Further Readings

- Overview of Docker
- Install Docker
- My Blog on Docker
- Course on Docker essentials