→ Exec - execute a command

Use the docker exec command to execute a command on my Docker containers,

e.g docker exec [container name] cat /etc/hosts

→ Run - attach and detach

docker run -d kodekloud/simple-webapp

This will run the docker containers in the background mode, and you will be back to your prompt immediately. The container will continue to run in the backend.

Now if you would like to attach back to the running container later, run the docker attach command:

docker attach [container ID or name]

List of Docker Commands

① Run - start a container
② PS - list containers
③ Stop - stop a container
④ Rm - remove a container
⑤ Images - list images

① Rmi - remove images
② Pull - download an image
③ Exec - execute a command
④ Run - attach and detach.

- Run - tag

docker run redis → redis version = 5.0.5

docker run redis:4.0 → tag

- Run - STDIN

If you'd like to provide your input, you must map the standard input of your host to the docker container using the dash i parameter.

docker run -i kodekloud/simple-prompt-docker

The -i parameter is for interactive mode.

docker run -it kodekloud/simple-prompt-docker

The -t stands for a pseudo terminal.

So with the combination of -it, we're now attached to the terminal as well as interactive mode on the container.

- Run - Port Mapping

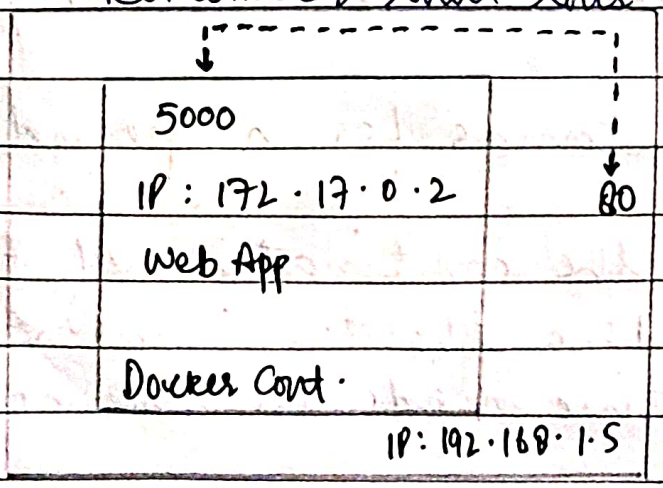
docker run kodekloud/webapp
* Running on http://0.0.0.0:5000/

The underlying host where Docker is installed is called Docker host or Docker engine.
When we run a containerized web application it runs and we're able to see that the server is running.
But how does the user access my application?

The application is listening on port 5000.
So we can access the application by using port 5000.

But what IP do I use to access it from a web browser?

① To use the IP of the docker container.
Every docker container gets an IP assigned by default, in this case it is 172.17.0.2. Remember that this is an



internal IP and is only accessible with in the docker host.
So if you open a browser from within the Docker host, you can go to HTTP://172.17.0.2:5000 to access the IP address.

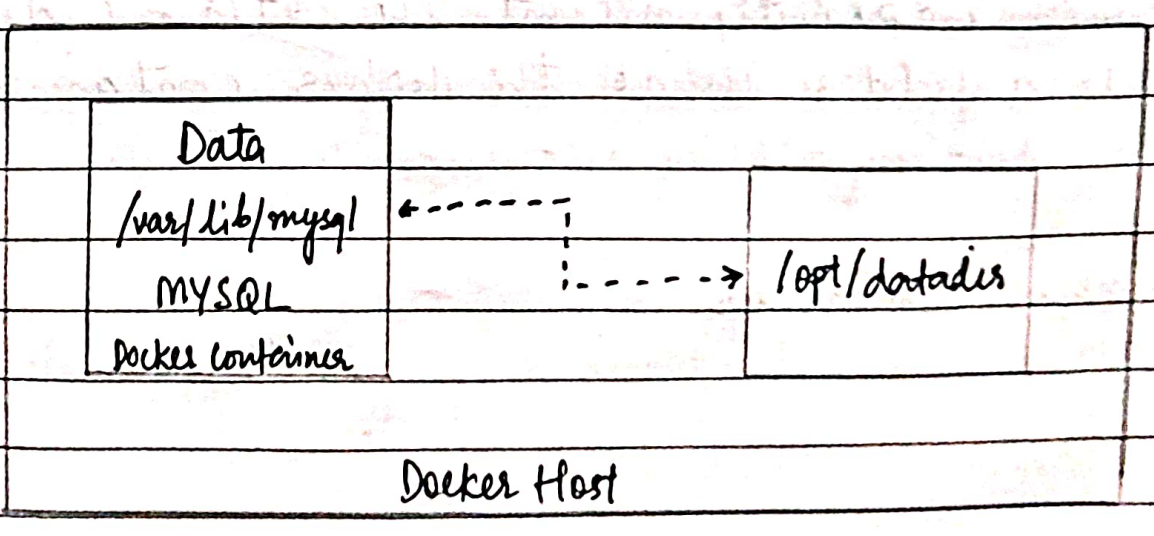
But since this is an internal IP, users out of the Docker host cannot access it using this IP. For this we could use the IP of the Docker host, which is 192.168.1.5.
But for that to work we must have mapped the port inside the Docker container to a free port on the Docker host.

```
docker run -p 80:5000 kodekloud/simple-webapp
```

- Run - Volume Mapping

```
docker run mysql
```

When databases and tables are created, the



data files are stored in location: /var/lib/mysql, inside the docker container. The docker container has its own isolated file system and any changes to any files happen within the container.

What happens if we were to delete the Mysql container and remove it?

As soon as we do that, the container along with all the data inside it gets blown away, meaning all your data is gone.

If you would like to persist data, you would want to map a directory outside the container on the Docker host to a directory inside the container.

```
docker run -v /opt/datadir:/var/lib/mysql
                    mysql
```

This way when Docker container runs, it will implicitly mount the external directory to a folder inside the docker container.

- Inspect Container

If you would like to see additional details about a specific container, use the docker inspect command and provide the container name or ID. It returns all details of a container in a JSON format.

ENVIRONMENT VARIABLES:

Use the Docker inspect command to inspect the properties of a running container. Under the config section, you will be able to find the list of environment variables set on the container.

IMAGES:

Q. Why would you need to create your own image?

It could be either because you cannot find a component or a service that you want to use as a part of your application on Docker Hub already, or your team decided that the application you're developing will be dockerized for ease of shipping and deployment.

Q. How to create my own image?

1. OS - Ubuntu

2. Update apt repo

3. Install dependencies using apt

4. Install python dependencies using pip

5. Copy source code to /opt folder

6. Run the web server using "flask" command.

Dockerfile :

A text file with instructions to build image

DOCKERFILE

```
FROM Ubuntu

RUN apt-get update
RUN apt-get install python

RUN pip install flask
RUN pip install flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP = /opt/source-code/
                       app.py flask run.
```

- docker build -t imageName:tagName "location of Dockerfile"

- docker push "location of dockerfile".
  - ↳ This will make it available on the public Docker Hub registry.

↳ This will create an image locally on your system.

DOCKERFILE

INSTRUCTION ARGUMENT

↓ Everything on the left in caps is an instruction

→ Everything on the right is an argument to these instructions.

LAYERED ARCHITECTURE:

Layer 1. Base Ubuntu layer

Layer 2. Changes in apt packages.

Layer 3. Changes in pip packages.

Layer 4. Source Code

Layer 5. Update Entrypoint with "flask" command.