Dockerfile Commands & Example

CommandDescription

ADD	Copies a file from the host system onto the container
CMD	The command that runs when the container starts
ENTRYPOINT	
ENV	Sets an environment variable in the new container
EXPOSE	Opens a port for linked containers
FROM	The base image to use in the build. This is mandatory and must be the first command in the file.
MAINTAINER	An optional value for the maintainer of the script
ONBUILD	A command that is triggered when the image in the Dcokerfile is used as a base for another image
RUN	Executes a command and save the result as a new layer
USER	Sets the default user within the container
VOLUME	Creates a shared volume that can be shared among containers or by the host machine
WORKDIR	Set the default working directory for the container

Once you've created a Dockerfile and added all your instructions, you can use it to build an image using the `docker build` command. The format for this command is:

```
docker build [OPTIONS] PATH | URL | -
```

The build command results in a new image that you can start using `docker run`, just like any other image. Each line in the Dockerfile will correspond to a layer in the images' commit history.

In the Docker EC2 instance

mkdir test

cd test

vi Dockerfile

FROM ubuntu:latest

MAINTAINER Chaitanya "chaitanya@gmail.com"

RUN apt-get update

RUN apt-get install -y python python-pip wget

RUN pip install Flask

WORKDIR /home

As you can see, it's pretty straightforward: we start from "ubuntu:latest," install dependencies with the `RUN` command, add our code file with the `ADD` command, and then set the default directory for when the container starts. Once we have a Dockerfile itself, we can build an image using `docker build`, like this:

docker build -t image .

Check the images created

docker images

Create a container

docker run -p 5000:5000 imagename