Docker file set of instruction commands

02:49 PM

docker image inspect myimage

FROM - base image we add
The **CMD** instruction has three forms:

CMD ["executable","param1","param2"] (exec form, this is the preferred form)
CMD ["param1","param2"] (as default parameters to ENTRYPOINT)
CMD command param1 param2 (shell form)

CMD ["/usr/bin/wc","--help"]
CMD echo "This is a test." | wc -

LABEL
LABEL <key>=<value> <key>=<value> <key>=<value> ...

The LABEL instruction adds metadata to an image
LABEL "com.example.vendor"="ACME Incorporated"
LABEL com.example.label-with-value="foo"
LABEL version="1.0"

EXPOSE
The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.

EXPOSE 80/tcp
EXPOSE 80/udp

ENV
ENV <key>=<value> ...

ENV MY_NAME="John Doe"
ENV MY_DOG=Rex\ The\ Dog
ENV MY_CAT=fluffy

ADD
ADD has two forms:

ADD [--chown=<user>:<group>] <src>... <dest>
ADD [--chown=<user>:<group>] ["<src>",... "<dest>"]

The ADD instruction copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>.

COPY
COPY has two forms:

```
COPY [--chown=<user>:<group>] <src>... <dest>
COPY [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

The COPY instruction copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.

ENTRYPOINT
ENTRYPOINT has two forms:

The exec form, which is the preferred form:

```
ENTRYPOINT ["executable", "param1", "param2"]
```

An ENTRYPOINT allows you to configure a container that will run as an executable.

//npm run start

For example, the following starts nginx with its default content, listening on port 80:

```
 docker run -i -t --rm -p 80:80 nginx
```

Port mapping

-p external port : container port 8080

192.168.10.20 --- 10 -- jenking, apche, nginx - they listing on diff ports.

192.168.10.20:80 ---> 80

Exposing external port

```
$ docker run --name thbs-nginx -d -p 8080:80 nginx
```

Then you can hit http://localhost:8080 or http://host-ip:8080 in your browser.

```
docker run -dit --name thbs-apche -p 8085:80 httpd:2.4
```

WORKDIR

WORKDIR /path/to/workdir
The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile. If the WORKDIR doesn't exist, it will be created even if it's not used in any subsequent Dockerfile instruction.

The WORKDIR instruction can be used multiple times in a Dockerfile. If a relative path is provided, it will be relative to the path of the previous WORKDIR instruction. For example:

```
WORKDIR /a
WORKDIR b
WORKDIR c
RUN pwd
```

The output of the final pwd command in this Dockerfile would be /a/b/c.

--- Try on build cheche

If you do not want to use the cache at all, you can use the **--no-cache=tru**e option on the docker build command.

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

How to build image from docker file

```
docker build -t mynode-app -f "Dockerfile-node" .
```

generate image from custome docker file

push custome image docker registory

```
cd /docker/
```

```
git clone https://github.com/sshelake25/thbs_aws_devops.git
cd thbs_aws_debops
```

create repo inside hub

on machine

login to hub from your in order to push $ docker login

```
docker push sshelake25/thbs_aws_dev:tagname
```

```
docker build -t sshelake25/thbs_aws_dev:s-nodeapp -f
"Dockerfile-node" .
```

```
docker push sshelake25/thbs_aws_dev:s-nodeapp
```

If someone want to user my image to run container docker pull sshelake25/thbs_aws_dev:s-nodeapp

```
docker run -d -p 8090:3000 sshelake25/thbs_aws_dev:s-nodeapp
```

http://ipaddress:8090/list_movies