**Dockerfile Instructions with Examples**

A **Dockerfile** is a text document that contains all the commands a user can call on the command line to build the Docker image.

Below is workflow to create Docker Container from Dockerfile

**Dockerfile –> Docker Image –> Docker Container**

**What is Dockerfile Instructions ?**

Dockerfile contains a set of Instructions to build Docker Image -> from Docker Image -> running Docker container

**#1: FROM –**

**FROM** in Dockerfile Instruction used to specify Docker Image Name and start the build process

**Example 1:**

#specify a Base Image

FROM ubuntu:latest

**Example 2:**

#specify a Base Image

FROM node:12

**#2: MAINTAINER –**

**MAINTAINER** in Dockerfile Instruction is used to about the person who creates the Docker Image

**Example:**

MAINTAINER support@anusoften.com

**#3: CMD –**

**CMD** in Dockerfile Instruction is used to execute a command in Running container, There should be one **CMD** in a Dockerfile.

**CMD** executes the commands when your Docker Image is deployed.

**Example 1:**

# To run apache2 in foreground

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]

**Example 2:**

FROM ubuntu:latest

CMD /bin/bash

**#4: RUN** –

**RUN** in Dockerfile Instruction is used to execute any commands on top of current Docker Image

**RUN** executes the command when you are building Image.

**Example 1:**

FROM ubuntu:latest

MAINTAINER support@anu.com

RUN apt-get update

RUN apt-get install -y apache2

If you want to run .sh(shell script) file inside Dockerfile

COPY test.sh .

RUN ./test.sh

#OR

RUN /path/to/test.sh

**#5: LABEL –**

**LABEL** in Dockerfile Instruction is used to specify metadata information of Docker Image.

**Example:**

FROM ubuntu:latest

LABEL "author"="FOSS TechNIx"

LABEL "Date"="2020-09-29"

**#6: EXPOSE –**

**EXPOSE** in Dockerfile Instruction is used to specify Network port for Docker container

**Example 1:**

# To Expose port 80 of Docker container

EXPOSE 80

**Example 2:**

EXPOSE 8080/tcp

**#7: ENV –**

**ENV** in Dockerfile Instruction is used to set Environment Variables with key and value.

**Example 1:**

FROM node:12  
ENV workdirectory /usr/node

**#8: ADD –**

**ADD:** Copies a file and directory from your host to Docker image, however can also fetch remote URLs, extract TAR/ZIP files, etc. It is used downloading remote resources, extracting TAR/ZIP files.

**Syntax:**

ADD <source>... <destination>

**Example 1:**

ADD java/jdk-8u231-linux-x64.tar /opt/jdk/

**Example 2:**

ADD https://soften.com/test.tar.xz /home/ubuntu/test/

**#9: COPY –**

**COPY** in Dockerfile Instruction used to Copies a file or directory from your host to Docker image, It is used to simply copying files or directories into the build context.

**Syntax:**

COPY <source>... <destination>

**Example 1:**

# To Install All dependencies for Node.js App

COPY package\*.json ./

RUN npm install

# To copy all application packages

COPY . .

**Example 2:**

COPY index.html /var/www/html

**#10: ENTRYPOINT –**

**ENTRYPOINT** in Dockerfile Instruction is used you to configure a container that you can run as an executable.

**ENTRYPOINT** specifies a commands that will executes when the Docker container starts.

**Example 1:**

FROM ubuntu:latest

ENTRYPOINT ["ls"]

**#11: VOLUME –**

**VOLUME** in Dockerfile Instruction is used to create or mount volume to docker container.

**Example 1:**

FROM node:12

RUN mkdir /node

WORKDIR /node

RUN echo "Welcome to Node.js" > node

VOLUME /node

**#12: USER –**

**USER** in Dockerfile Instruction is used to set the user name and UID when running container

**Example 1:**

**USER** admin

To create new user in Dockerfile and login to user.

**Example 2:**

**RUN** adduser -D admin

**USER** admin

**#13: WORKDIR –**

**WORKDIR** in Dockerfile Instruction is used to set the working directory.

**Example 1:**

# To Create nodejsapp directory

WORKDIR /nodejsapp

**#14: ARG –**

**ARG** in Dockerfile Instruction is used to set Environment variables with key and value during the image build .

**Example 1:**

**ARG** JAVA\_PATH=/opt/jdk/jdk1.8.0\_251

**ENV** JAVA\_HOME ${JAVA\_PATH}

**#15: ONBUILD –**

**ONBUILD** in Dockerfile Instruction is used to specify command that runs when the image in Dockerfile is used as base image for another image.

**Examples 1:**

FROM node:12

RUN mkdir -p /usr/node/app

WORKDIR /usr/node/app

**ONBUILD** COPY package.json /usr/node/app/

**ONBUILD** RUN npm install

**ONBUILD** COPY . /usr/node/app

CMD [ "npm", "start" ]

**#16: STOPSIGNAL –**

**STOPSIGNAL**in Dockerfile Instruction is used to set the system call signal that will be sent to the container to exit

**Example 1:**

**STOPSIGNAL** SIGQUIT

**#17: SHELL –**

**SHELL** in Dockerfile Instruction is used to set the default shell.

**Example:**

**SHELL** ["/bin/bash", "-c", "echo hello"]

**#18: HEALTHCHECK –**

**HEALTHCHECK** in Dockerfile Instruction is used to Check container health by running a command inside the container

**Example 1:**

FROM ubuntu:latest

HEALTHCHECK --interval=60s --timeout=5s \

CMD curl -f http://soften.info/ || exit 1

EXPOSE 80

**#19: .dockerignore –**

**.dockerignore** in Dockerfile Instruction is used to prevent copy local modules and other unwanted file being copied into Docker Image.

Create a .dockerignore in same directory and you can add unwanted modules/files into it.

sudo nano .dockerignore

\*.yaml

\_\_pycache\_\_/

.git

.aws

.env

**Conclusion**:

We have covered Dockerfile Instructions with Examples/Dockerfile Instructions Explained with Examples.

**Dockerfile Instructions FAQ (Frequently Asked Questions)**

**What is difference between Dockerfile ADD vs COPY ?**

**ADD:** Copies a file and directory from your host to Docker image, however can also fetch remote URLs, extract TAR/ZIP files, etc. It is used downloading remote resources, extracting TAR/ZIP files.  
**Syntax:**  
ADD <source> <destination>  
**Example:**  
ADD java/jdk-8u231-linux-x64.tar /opt/jdk/

**COPY :** Copies a file or directory from your host to Docker image, It is used to simply copying files or directories into the build context.  
**Syntax:**  
COPY <source> <destination>  
**Example:**  
COPY index.html /var/www/html

**What is the difference between CMD and ENTRYPOINT in a Dockerfile?**

**CMD** in Dockerfile Instruction is used to execute a command in Running container, There should be one **CMD** in a Dockerfile.

**ENTRYPOINT** in Dockerfile Instruction is used you to configure a container that you can run as an executable.

**How to Create Docker Image for Node JS Application [2 Steps]**

[Node.js](https://nodejs.org/en/) is free and an open-source cross-platform JavaScript run-time environment that allows server-side execution of JavaScript code.  [**NPM**](https://www.npmjs.com/)(Node Package Manager) is command line tool for Node.js packages that installs, updates and uninstall packages in your projects.

Docker provides a robust client-server application architecture with a powerful server, REST API and command-line interface client.

**Prerequisites**

* Ubuntu 16/18/20.04 LTS
* SSH access with sudo privileges
* Firewall Port: 3000

**Install Node JS and NPM on Ubuntu**

If you are using Ubuntu OS then Install Node JS and NPM on Ubuntu using below articles

[How to Install Node.js and NPM on Ubuntu 20.04 LTS](https://www.fosstechnix.com/how-to-install-node-js-and-npm-on-ubuntu/)

[How to Install Latest Node.js and NPM on Ubuntu 19.04,18.04/16.04 LTS](https://www.fosstechnix.com/install-latest-node-js-and-npm-on-ubuntu/)

If you are using other OS then follow [Node JS Official Site](https://nodejs.org/en/download/package-manager/) to Install if not installed

**Step 1: Creating Node.js Application**

Lets create the directory named **nodejsdocker** to add node js files to test.

$ sudo mkdir nodejsdocker

Navigate to **nodejsdocker** directory

$ cd nodejsdocker

Create the package.json file where you will specify all dependencies of your Node JS application

$ sudo nano package.json

paste the below lines into it

{

"name": "Docker\_NodeJS\_App",

"version": "0.1",

"description": "Node.js Application with Docker",

"main": "server.js",

"scripts": {

"start": "node server.js"

},

"dependencies": {

"express": "^4.17.1"

}

}

Next create the server.js page to test Node JS application with express framework

$ sudo nano server.js

Paste the below lines in it

'use strict';

const express = require('express');

// Constants

const PORT = 3000;

const HOST = '0.0.0.0';

const app = express();

app.get('/', (req, res) => {

res.send('Testing Node JS Application');

});

app.listen(PORT, HOST);

console.log(`Running on http://${HOST}:${PORT}`);

**Step 2: How to Create Docker Image for Node JS Application**

Before creating Docker Image for Node JS application install the docker using below link if not installed.

[How to Install Docker on Ubuntu 19.10/18.04/16.04 LTS](https://www.fosstechnix.com/install-docker-on-ubuntu/)

[How to Install Docker on Windows 10](https://www.fosstechnix.com/install-docker-on-windows/)

For other OS follow [Docker Official Guide](https://docs.docker.com/engine/install/)

Next create the Dockerfile with below command in Project root directory

$ sudo nano Dockerfile

Paste the below Dockerfile instructions in it

FROM node:12

# To Create nodejsapp directory

WORKDIR /nodejsapp

# To Install All dependencies

COPY package\*.json ./

RUN npm install

# To copy all application packages

COPY . .

# Expose port 3000 and Run the server.js file to start node js application

EXPOSE 3000

CMD [ "node", "server.js" ]

Now build the Docker Image using below command

$ sudo docker build -t nodejsdocker .

**Sample Output:**

added 50 packages from 37 contributors and audited 50 packages in 6.293s

found 0 vulnerabilities

Removing intermediate container 3e97b890d792

---> 6a354a5d9d56

Step 5/7 : COPY . .

---> 9f75f8ce0d6f

Step 6/7 : EXPOSE 3000

---> Running in 66c4a65f7392

Removing intermediate container 66c4a65f7392

---> 9009232a55e9

Step 7/7 : CMD [ "node", "server.js" ]

---> Running in 8d4d96bc2877

Removing intermediate container 8d4d96bc2877

---> ba86a68c01e4

Successfully built ba86a68c01e4

Successfully tagged nodejsdocker:latest

once build is successful , you can see list of docker images using below command

$ docker images

**Output:**

REPOSITORY TAG IMAGE ID CREATED SIZE

nodejsdocker latest ba86a68c01e4 5 minute ago 922MB

<none> <none> 5c85c5a4da87 5 minute ago 922MB

<none> <none> 29fa5eed7d5e 5 minute ago 918MB

node 12 e163934eebb0 5 minute ago 918MB

alpine latest a24bb4013296 5 minute ago 5.57MB

We have covered, How to create Docker Image for Node JS Application.

Run the Docker container , **-p**is used to map the public port to docker container internal port, Here I am using same port for both.

$ docker run -p 3000:3000 nodejsdocker

**Output:**

docker run -p 3000:3000 nodejsdocker

Running on http://0.0.0.0:3000

If you want to run Docker container in detached mode use below command

$ docker run -p 3000:3000 -itd CONTAINER\_ID

To check docker Process

$ docker ps

**Output:**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

ac89891c8dfb nodejsdocker "docker-entrypoint.s…" 10 minute ago Up 37 minutes 0.0.0.0:3000->3000/tcp nostalgic\_robinson

To check Docker Container logs

$ docker logs **CONTAINER\_ID**

**Output:**

Running on http://0.0.0.0:3000

**Testing the Docker Container**

Open your favorite browser and type server IP with container public port

$ Server\_IP:port

You can test also using curl command

$ curl -i http://localhost:3000

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: text/html; charset=utf-8

Content-Length: 27

ETag: W/"1b-uO/TfUtLPwJy4DS2pakC/kf+9oA"

Date: Thu, 11 Jun 2020 04:55:34 GMT

Connection: keep-alive

Testing Node JS Application

To stop Docker container

$ docker stop **CONTAINER\_ID**

**Pushing Docker Image to Docker Hub Repository**

If you want to push the docker image to Docker Hub Registery. First login to [https://hub.docker.com](https://hub.docker.com/)  with ID and password using command line

$ docker login

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.

Username: **anusoften**

Password:

Login Succeeded

Now push Docker Image to Docker Hub Repository

$ docker push nodejsdocker

Error: denied: requested access to the resource is denied:docker

If you are getting above error while pushing docker images to docker hub repository first time then first tag the Docker Image and try to push again

$ docker tag nodejsdocker anusoften/nodejsdocker

Push the Docker Image again

$ docker push anusoften/nodejsdocker