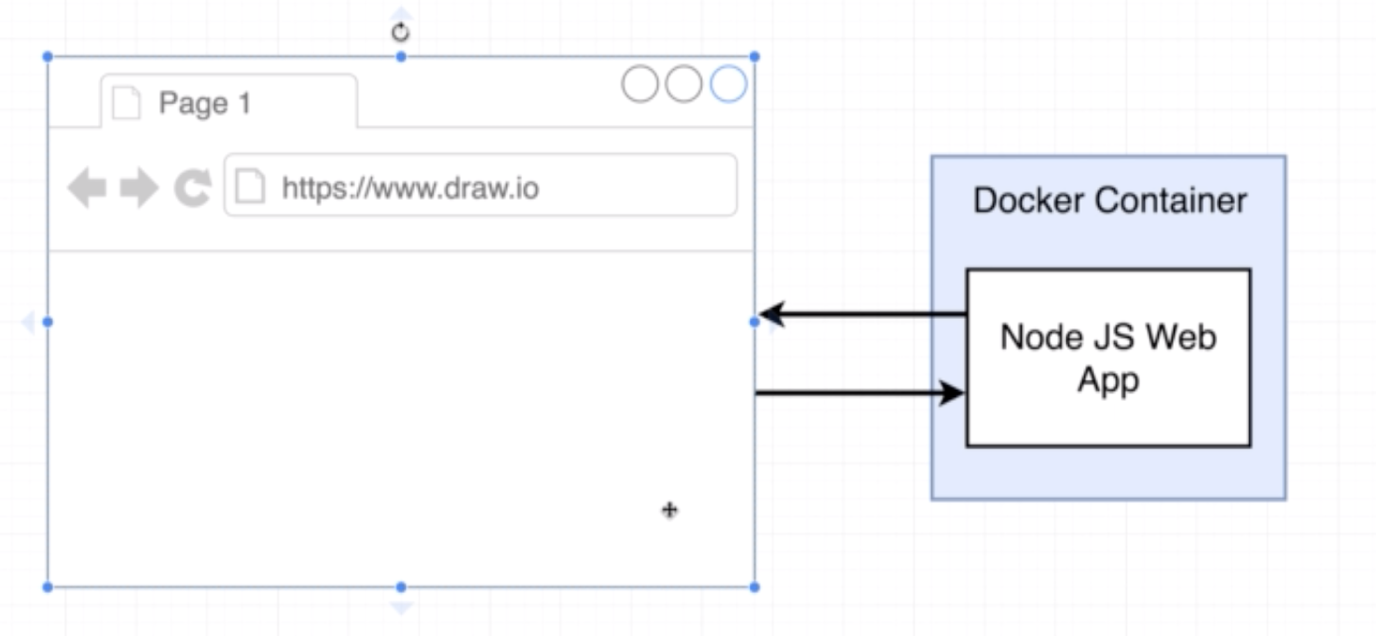
**Making Real Project With Docker**

* **Project Outline:**

****

Steps to running the app as container:

Diagram

Description automatically generated

We are going to do some mistakes just for learning purpose.

A picture containing text

Description automatically generated

* **Node server Setup:**

Create the node files as below

How to run node js app?

Diagram

Description automatically generated

**Diagram

Description automatically generated**

* **Base image issue:**

After creating the necessary file to create the node app (including the Dockerfile) if you try to build an image, it will fail with below error.

Sending build context to Docker daemon 4.096kB

Step 1/3 : FROM alpine

---> cdf98d1859c1

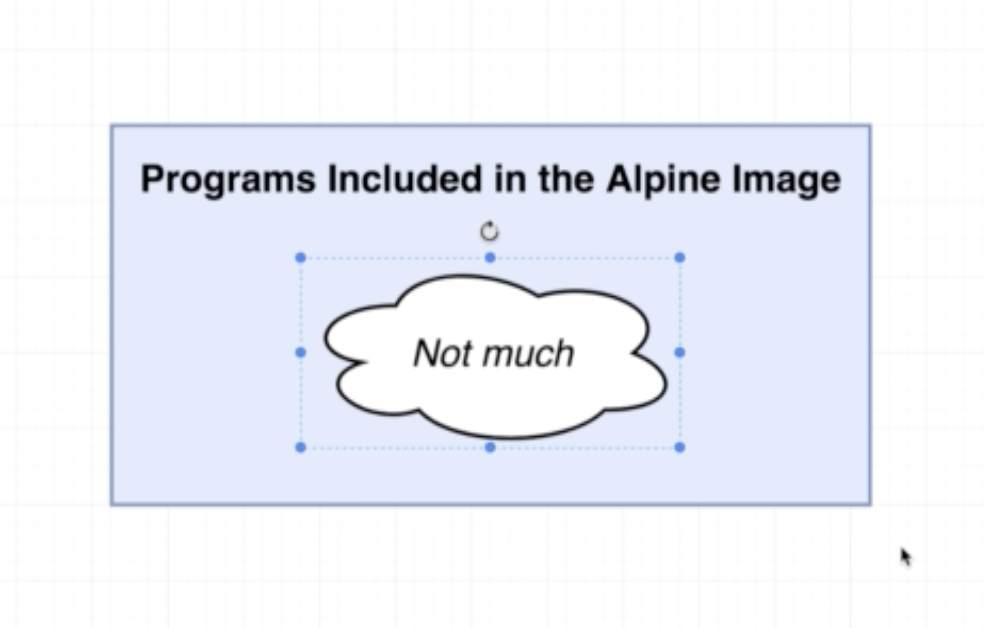
Step 2/3 : RUN npm install

---> Running in b28929d2f5df

/bin/sh: npm: not found

The command '/bin/sh -c npm install' returned a non-zero code: 127

The above error is because the base alpine image doesn’t have the program we require (npm)



There are two options to overcome above issue:

Find the base image which has node and npm installed.

Use alpine image itself and add an additional command to install node and npm.

Note: What does alpine mean in Docker?

Alpine means an image which is small and compact as much as possible. Many popular repositories are providing alpine version of their image.

We are going to use the first approach (use different base image)

In docker hub, you can find a repository for node, node has multiple flavours of image, among them select node:alpine.

Note: node:alpine will contain only the node installation.

Now after changing the base image, run the docker build command.

Now the docker build will fail with below error.

---> Running in 763bbd035a39

npm WARN saveError ENOENT: no such file or directory, open '/package.json'

npm notice created a lockfile as package-lock.json. You should commit this file.

npm WARN enoent ENOENT: no such file or directory, open '/package.json'

npm WARN !invalid#2 No description

npm WARN !invalid#2 No repository field.

npm WARN !invalid#2 No README data

npm WARN !invalid#2 No license field.

* **A few missing files:**

The folder problem with diagrams:

Diagram

Description automatically generated

Diagram

Description automatically generated

When we execute the npm install command, we don’t have “folder or file” named “package.json” as it a segment of the hard-drive.

Diagram

Description automatically generated

The folder or file resides in the rest of the hard drive block. We have to copy that as part of our Dockerfile definition

COPY is the command which copies the files or folder and put it inside the container.

Diagram

Description automatically generated

After adding the copy command above the “npm install”. Run the image, you will get the console output as “Listening on port 8080”.

But when you try to access the <http://localhost8080>” in browser it will give an error message saying “The site can’t be reached”.

* **Container Port Mapping:**

**Diagram

Description automatically generated**

Whenever we access <http://localhost:8080> the request will not be redirected to container, as the container manages individual set of ports.

To forward the traffic to container, we have to do a port mapping.

**Diagram

Description automatically generated**

Note: port mapping is required to route the outside traffic to container, but not for the reverse way (container to outside world). This is done automatically.

Port mapping has to be done at runtime, not on the Dockerfile.

Table

Description automatically generated

Note: source and target port of port forwarding need not to be identical.

* **Specifying a working directory:**

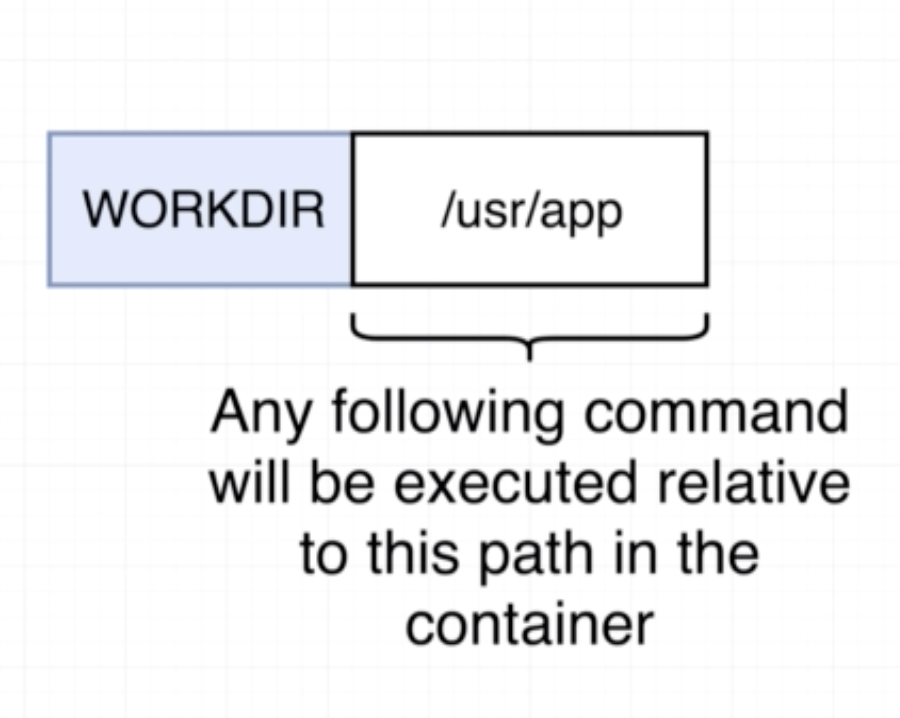
Open a shell using the above image by executing below command.

docker run –it dspandiyan/simpleweb sh

if you see the root folder of the container, it will have all the files we copied as part of COPY command, this is not the right way to place the files, we can create a working directory for us and keep all the files there.

Table

Description automatically generated



Use “WORKDIR” command to place the files under a specified location.

WORKDIR /usr/app

Note: At runtime if the specified directory is not there, it will create a new folder.

* **Unnecessary rebuilds:**

After creating the image and running that image, whatever the changes you will do in the source file, it will affect the running containers, as docker takes a file system snapshot.

If you want to apply the changes we do after the image creation, we have to do additional configuration in the Dockerfile.

Change the value “Bye there” in index.js and build the image again. If you notice the logs, as there is a change just in the index.js, it copied all the files using copy command, post that it ran the “npm install” too. Npm install is not required as we haven’t changed anything in the dependency.

Step 1/5 : FROM node:alpine

---> d151df308b43

Step 2/5 : WORKDIR /usr/app

---> Using cache

---> 77ddb2dff3dc

Step 3/5 : COPY ./ ./

---> 2733a569b452

Step 4/5 : RUN npm install

---> Running in b448eb3d730c

npm notice created a lockfile as package-lock.json. You should commit this file.

npm WARN app No description

npm WARN app No repository field.

npm WARN app No license field.

added 50 packages from 37 contributors and audited 50 packages in 11.398s

found 0 vulnerabilities

Removing intermediate container b448eb3d730c

---> cbd12b189fa1

Step 5/5 : CMD ["npm", "start"]

---> Running in c11a9707f0b1

Removing intermediate container c11a9707f0b1

---> 2085f690641d

Successfully built 2085f690641d

* **Minimizing cache busting and rebuilds:**

Use two COPY command to redo the npm install.

Step 1/6 : FROM node:alpine

---> d151df308b43

Step 2/6 : WORKDIR /usr/app

---> Using cache

---> 77ddb2dff3dc

Step 3/6 : COPY ./package.json ./

---> Using cache

---> 671ff9aa054f

Step 4/6 : RUN npm install

---> Using cache

---> 16932bc9a423

Step 5/6 : COPY ./ ./

---> Using cache

---> 66e9725ee90d

Step 6/6 : CMD ["npm", "start"]

---> Using cache

---> b7b4cc51e821

Successfully built b7b4cc51e821

Successfully tagged dspandiyan/simpleweb:latest

After we use two COPY command (one to copy the package.json and another to copy all the files) if you rebuild the image, it will not execute the “npm install” command on each build, as there is not change on package.json. It uses the cached verson.