

Dockerfile

[Official Dockerfile Documentation](#)

[Docker Images Docs](#) is also a good place to get information on using various basic images which can be built off of.

Below, we define a `Dockerfile` within some exclusive directory on our system where we want to work on our docker image. Create this file with any text editor, where the following commands are possible in a `CMD input` format.

`FROM` defines the base image to build off of from a repository on [dockerhub](#)

`LABEL`

`RUN` defines a command to run *in sequence* as the `Dockerfile` is built.

`SHELL` Restarts into a given shell, seen below where we pass `--login` and `-c` parameters to bash

`EXPOSE` defines a port to expose on the container to the host `VOLUME`

`USER`

`WORKDIR`

`ENTRYPOINT`

`ENV`

`ARG`

`COPY` \

```
# Default repository is the same that is used when running `hexo init`
ARG REPO='https://github.com/hexojs/hexo-starter'

# https://hub.docker.com/_/nginx as our base image to build off of
FROM nginx:latest

# Otherwise provide one during build..
# `docker build -t <TAG> . --build-params REPO='https://github.com/username/repo'
ARG REPO

LABEL maintainer='username@gmail.com'

# Install additional packages we need
RUN apt-get update && apt-get -y upgrade && apt install -y curl vim

# Grab NVM and restart shell to load commands into bash
RUN curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash

SHELL ["/bin/bash", "--login", "-c"]

# Install NVM stable version and hexo
RUN nvm install stable
RUN npm install -g hexo-cli

EXPOSE 8080
```

In the Dockerfile above, I use `ARG` to define a default value `REPO` which represents the repository to clone when building this docker image. In this case, the repository is the same that is cloned automatically when running `hexo init`. Since we defined `ARG REPO` after the `FROM` command in the dockerfile, it will be accessible for the entire build process, instead of being limited to `FOR`. If you want to provide a different value for this when building the image, you can do so by using `docker build -t . --build-params REPO=`

`SHELL` restarts our shell to load the `nvm` commands into `bash` so we can in the next step `nvm install stable`. Otherwise, this command would fail saying that `nvm` did not exist.

Building Docker Images

To build a dockerfile into an image, run the following command, where `-t` is tagging the built image with a tag in the preferred format of `dockerhub-username/dockerhub-reponame:version`

```
docker build -t username/nginx-hexo:0.1 .
```

Running Built Images

We can run `docker images` and see the following output displaying all the built docker images on our machine

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|---------------------|-----|--------------|----------------|-------|
| username/nginx-hexo | 0.1 | 86325466e505 | 32 minutes ago | 331MB |

Now to start our newly built image, we run the following command

```
docker container run -d --name nginx-hexo \
username/nginx-hexo:0.1
```

To check that our image is running, run `docker container ls` to see output similar to the below

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS |
|------------------|-------------------------|--------------------------|----------------|---------------|-------|
| NAMES | | | | | |
| 7a74d968f0d2 | username/nginx-hexo:0.1 | "nginx -g 'daemon of..." | 30 minutes ago | Up 30 minutes | |
| 80/tcp, 8080/tcp | nginx-hexo | | | | |

Pushing Images to DockerHub

To login to docker, we need to run `docker login` and follow the prompts, supplying our username and password. On some systems, you could see the below error -

```
error getting credentials - err: exit status 1, out: `GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.freedesktop.secrets was not provided by any .service files`
```

To fix this, we run the following

```
sudo apt install gnupg2 pass
```

After logging into docker on your machine, since we already properly tagged our image when we built it with `docker build -t <TAG>` above, we can simply `docker push <TAG>`. Below, we look up our image's local ID and retag it to ensure this matches our DockerHub username and preferred image name / tag. Then, we push the image to DockerHub, publicly. If you want this image to be private, which it should be if unstable, you can do so by logging into [dockerhub](#) and modifying the repository settings after making the first push.

Get the image ID -

| docker images | | | | |
|---------------------|-----|--------------|----------------|-------|
| username/nginx-hexo | 0.1 | 86513686e505 | 32 minutes ago | 331MB |

Assign the image ID a new tag (This is the same as the old in this case) -

```
docker tag 83213123e515 username/nginx-hexo:0.1
```

Push the docker image to DockerHub -

```
docker push username/nginx-hexo:0.1
```

Saving Images Locally

If you don't want to push to DockerHub for any reason, you can always just save you image locally using `docker save`, and then reload it later either on the same machine or a new one by using `docker load`.

Save the image

```
docker save username/nginx-hexo:0.1 > nginx-hexo.tar
```

Reload the image

```
docker load --input nginx-hexo.tar
```

You should see the following output

a333833f30f7: Loading layer

[=====>] 59.4MB/59.4MB

68a235fa3cf2: Loading layer

[=====>] 119.3kB/119.3kB

b402ba6c11cd: Loading layer

[=====>] 135.7MB/135.7MB

3fc85c9d7bd6: Loading layer

[=====>] 17.61MB/17.61MB

Loaded image: username/nginx-hexo:0.1

Revision #6

Created 26 May 2020 00:37:26 by Shaun Reed

Updated 18 December 2021 17:13:28 by Shaun Reed