Building images with Dockerfiles

As we saw in the Docker Walkthrough chapter, the general Docker workflow is:

- start a container based on an image in a known state
- add things to the filesystem, such as packages, codebases, libraries, files, or anything else
- commit the changes as layers to make a new image

In the walkthrough, we took a very simple approach of just starting a container interactively, running the commands we wanted (like "apt-get install" and "pip install"), and then committing the container into a new image.

In this chapter, we'll look at a more robust way to build an image. Rather than just running commands and adding files with tools like *wget*, we'll put our instructions in a special file called the *Dockerfile (https://docs.docker.com/reference/builder/)*. A Dockerfile is similar in concept to the recipes and manifests found in infrastructure automation (IA) tools like Chef (http://www.getchef.com/) or Puppet (http://puppetlabs.com/).

Overall, a Dockerfile is much more stripped down than the IA tools, consisting of a single file with a DSL that has a handful of instructions. The format looks like this:

```
# Comment
INSTRUCTION arguments
```

The following table summarizes the instructions; many of these options map directly to option in the "docker run" command:

Command	Description
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.

Example of building an image from a Dockerfile

Perhaps the best way to understand a Dockerfile is to dive into an example. Let's take a look at the example we went through in our overview chapter and condense it into a Dockerfile:

```
#
# Super simple example of a Dockerfile
#
FROM ubuntu:latest
MAINTAINER Andrew Odewahn "odewahn@oreilly.com"

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

ADD hello.py /home/hello.py

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 "simple_flask:dockerfile" .
```

The "-t" flag adds a tag to the image so that it gets a nice repository name and tag. Also not the final ".", which tells Docker to use the Dockerfile in the current directory. Once you start the build, you'll see it churn away for a while installing things, and when it completes, you'll have a brand new image. Running `docker history` will show you the effect of each command has on the overall size of the file:

```
$ docker history simple_flask:dockerfile
IMAGE               CREATED             CREATED BY                                      SIZE
9ada423c0a60        3 days ago          /bin/sh -c #(nop) WORKDIR /home                 0 B
5c3625267cd9        3 days ago          /bin/sh -c #(nop) ADD file:96e699cd177f1a3f3c   163 B
9c20a6548fbe        3 days ago          /bin/sh -c pip install Flask                    4.959
 MB
7195370ae6e1        3 days ago          /bin/sh -c apt-get install -y python python-p   136.1
 MB
761bf82875cc        3 days ago          /bin/sh -c apt-get update                       19.94
 MB
40b29df1d2c2        3 days ago          /bin/sh -c #(nop) MAINTAINER Andrew Odewahn "   0 B
c4ff7513909d        9 days ago          /bin/sh -c #(nop) CMD [/bin/bash]               0 B
cc58e55aa5a5        9 days ago          /bin/sh -c apt-get update && apt-get dist-upg   32.67
 MB
0ea0d582fd90        9 days ago          /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$/   1.895
 kB
d92c3c92fa73        9 days ago          /bin/sh -c rm -rf /var/lib/apt/lists/*          0 B
9942dd43ff21        9 days ago          /bin/sh -c echo '#!/bin/sh' > /usr/sbin/polic   194.5
 kB
1c9383292a8f        9 days ago          /bin/sh -c #(nop) ADD file:c1472c26527df28498   192.5
 MB
511136ea3c5a        14 months ago                                                       0 B
```

Finally, you can start the container itself with the following command:

```
$ docker run -p 5000:5000 simple_flask:dockerfile python hello.py
```

Notice that in this example we're running the Flask app directly when we start the container, rather than just running the bash shell and starting it as we've done in other examples.

Dockerfiles vs. Infrastructure Automation (IA)

Dockerfiles provide a relatively simple way to create a base image. And, because you can use the FROM command to chain Dockerfiles together into increasingly complex images, you can do quite a lot, even with Docker's (refreshingly!) minimal command set. But, if you already have an existing IA tool (and you should!), such as Chef (http://www.getchef.com/), Puppet (http://puppetlabs.com/), Ansible (http://www.ansible.com/home), Salt (http://www.saltstack.com/), it's very unlikely you could or even should rewrite everything. So, if you're in this situation what can you do?

I HAVE NO IDEA! I NEED TO RESEARCH THIS MORE, BUT I THINK YOU CAN USE A TOOL LIKE Packer]((https://github.com/jeroenjanssens/data-science-at-the-command-line/tree/master/dst/build)http://www.packer.io/ (http://www.packer.io/)). MAYBE I CAN CONVINCE Jeroen Janssens to do something with his Ansible stuff

‹ Previous (example.html) Next › (dockerhub.html)

Matt • 4 years ago

I have done a little research around this, specifically regarding Ansible. I'm doing some proof-of-concept work over the next few weeks, and will be sure to include what I've found here.

1 • Reply • Share ›

Sukesh Kasam ➚ Matt • 2 years ago

Hi Matt I am doing proof of concept in docker swarm, have you done with creating docker file to docker image.

• Reply • Share ›

odewahn Mod ➚ Matt • 4 years ago

Great! Please do and send a pull request to a comment with what you discover.

• Reply • Share ›

ChaZ • 18 days ago

Dockerfiles vs. Infrastructure Automation (IA)

I think both can be used in a complementary way: Utilize Docker to create images and use IA to run docker generated images. But in this approach, IA's utility becomes limited.

• Reply • Share ›

Binh Thanh Nguyen • 2 years ago

Thanks, nice post

• Reply • Share ›

Prayag Upd • 2 years ago

docker run -p 5000:5000 simple_flask:dockerfile does not do anything for me.

My dockerfile here -> https://github.com/prayagup...

• Reply • Share ›

Siva Ram ➚ Prayag Upd • a year ago

Try writing EXPOSE 5000 in your dockerfile of Github.

• Reply • Share ›