How to understand building images with docker-compose

Zhao Li Follow

Jun 10, 2019 · 6 min read ★

Problem

When we were downloading images and then running the image as a container, we were using images that others have provided via Docker Hub. How do we create a Docker image that is customized for our own use?

Solution

To create a custom image that meets our own requirements, we will need to build an image that contains what we need.

This tutorial will go through building a Docker image and then explaining what happened.

Here are the steps we will take:

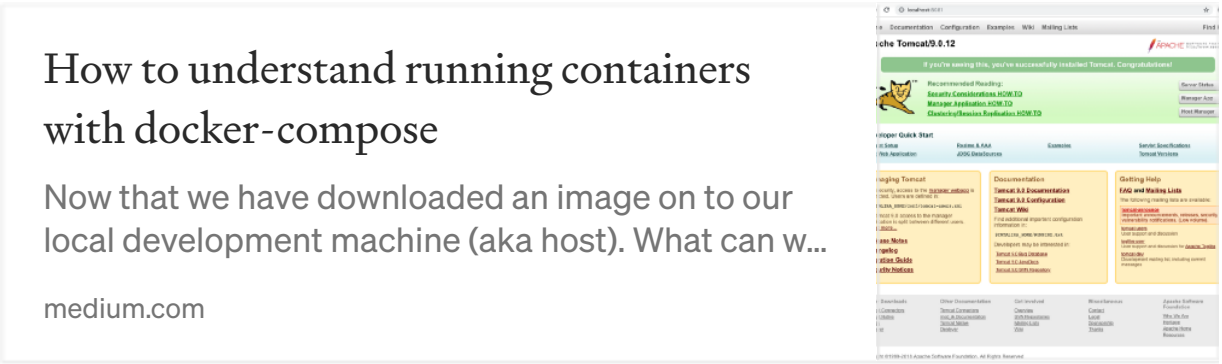
1. update our existing `docker-compose.yml` file so that it builds a custom image
2. add instructions to a `Dockerfile` that Docker Compose follows to build a new image with
3. use `docker-compose build` to build our custom image
4. use `docker-compose up` to run our custom image

This diagram shows what is happening behind the scenes:

Docker-Compose Build

Front Matter

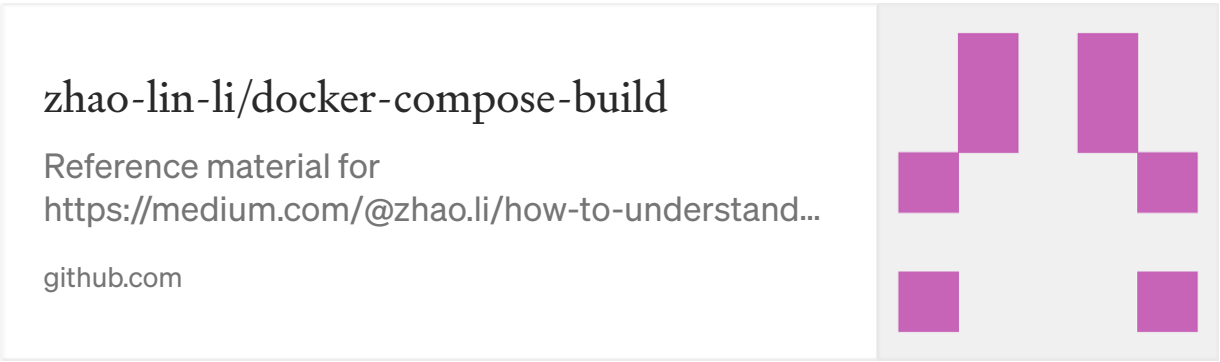
This tutorial assumes you have gone through the steps from this tutorial:



Here is the relevant code from that tutorial, pasted here for convenience:

```
version: '3'
  services:
    app:
      image: tomcat:9.0.12
      ports:
        - "8081:8080"
```

This tutorial will be updating the code from the previous tutorial and the new code from this tutorial can be found here:



Update `docker-compose.yml` to Build

We have been telling docker-compose to use the official `tomcat` image tagged at `9.0.12` that is provided by Docker Hub. In order for us to build our own image we will need to change our `docker-compose.yml` to this:

```
version: '3'
services:
  app:
    build: .
    ports:
      - "8081:8080"
```

The `build` key allows us tell docker-compose where to find the build instructions as well as the files and/or folders used during the build process.

Add Dockerfile

We can now provide our build instructions via a `Dockerfile`. Let's then create a file named `Dockerfile` and add this content into it.

```
FROM tomcat:9.0.12
COPY index.html /usr/local/tomcat/webapps/ROOT/index.html
```

The line, with the `FROM` instruction, tells docker compose to use `tomcat:9.0.12` as our base image. A base image is the starting

point of where we will start building our custom image. We could have chosen any image as a base image. We could have also chosen to build our custom image from scratch: https://docs.docker.com/develop/develop-images/baseimages/

But basing off of `tomcat:9.0.12` will keep things familiar for this exercise.

The line, with the `COPY` instruction, tells docker compose to copy `index.html` from our host file system into our image files system at `/usr/local/tomcat/webapps/ROOT/index.html`. This file location is where tomcat locates its splash page and is not related to anything about Docker or containers.

By copying `index.html` to `/usr/local/tomcat/webapps/ROOT/index.html` inside the container, we have effectively overwritten tomcat's default splash page with a web page we can customize.

We could have left this out, and just build and run an image that is identical to the vanilla `tomcat:9.0.12` image. But making this modification to the `tomcat:9.0.12` image allows us to visually see that we have indeed built a custom image. The `COPY` instruction also serves as a simple example of how we can customize images.

With the `Dockerfile` now requiring an `index.html`, we need to create this file and add some simple HTML to it:

```
<html>
  <body>
    <h1>docker-compose build worked!</h1>
  </body>
</html>
```

Now that we have all of our files in place, we are ready to build.

Docker Compose Build

We simply need to issue `docker-compose build` to build our custom image.

```
$ docker-compose build
Building app
Step 1/2 : FROM tomcat:9.0.12
9.0.12: Pulling from library/tomcat
38517b303516: Pull complete
d518e0b86e7e: Pull complete
75323769698f: Pull complete
7961e608858b: Pull complete
c08b11391f09: Pull complete
315ba632da3b: Pull complete
c6e12a686ad3: Pull complete
978161388bc0: Pull complete
e798a276443d: Pull complete
f8723d658b52: Pull complete
84398876c738: Pull complete
Digest:
sha256:2c1f49c0e9b5d32cec3ee7ad18b140acc44eeab727a7912928
8d859c1a6931e9
Status: Downloaded newer image for tomcat:9.0.12
 ---> 7c5bb46055d4
Step 2/2 : COPY index.jsp
/usr/local/tomcat/webapps/ROOT/index.jsp
```

```
 ---> b81f7351823f
Successfully built b81f7351823f
Successfully tagged docker-compose-build_app:latest
```

If the image already exists on your host, then the output will look more like this:

```
$ docker-compose build
Building app
Step 1/2 : FROM tomcat:9.0.12
 ---> 287b3865bc34
Step 2/2 : COPY index.jsp
/usr/local/tomcat/webapps/ROOT/index.jsp
 ---> d2ca3af22e07
Successfully built d2ca3af22e07
Successfully tagged docker-compose-build_app:latest
```

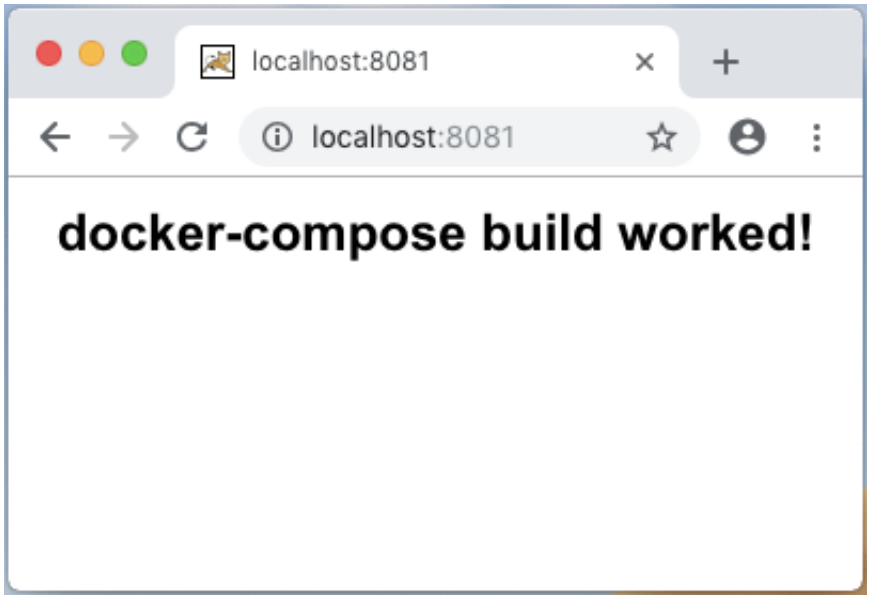

Now that we have built our image, we are ready to run it.

Docker Compose Up

We can then issue `docker-compose up` to run our custom image.

```
$ docker-compose up
Creating network "docker-compose-build_default" with the
default driver
Creating docker-compose-build_app_1 ... done
Attaching to docker-compose-build_app_1
app_1  | NOTE: Picked up JDK_JAVA_OPTIONS:  --add-
opens=java.base/java.lang=ALL-UNNAMED --add-
opens=java.base/java.io=ALL-UNNAMED --add-
opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
...
app_1  | 21-Oct-2018 23:02:14.423 INFO [main]
org.apache.catalina.startup.Catalina.start Server startup
in 2018 ms
```

If we then browse to `localhost:8081` on our browser, we should see our custom splash page:



Custom Tomcat Splash Page

Congratulations! You have just created your own image and ran it.

If you found this tutorial article useful and have other technologies that you are interested in learning how to get started with, please submit your ideas to https://gitlab.com/zhao-li/tutorial-articles.

Thank you for your time 🙏

Notes

Tomcat Base Image

We could have chosen any image as a base image, but because, we were making a small modification to the Tomcat image, it makes sense to base our image off of the tomcat image.

Common Instructions

There are a ton of instructions you can add to a `Dockerfile`. Here is a list of available instructions: https://docs.docker.com/engine/reference/builder/

The two instructions I most commonly use in my `Dockerfile`'s are `COPY` and `RUN`.

Development Workflow

If you were actively developing a custom image, the process above would be extremely slow, because it would entail doing this, repeatedly:

```
docker-compose down
# make modifications
docker-compose build
docker-compose up
# test modifications
```

Instead you can use the `volumes` key in the `compose-compose.yml`. This article talks about it more: https://medium.com/@zhao.li/how-to-understand-working-with-files-in-containers-using-docker-compose-6d8e68ed790b

You can sometimes skip the extra step of `docker-compose build` because `docker-compose up` will automatically build the container if it does not exist.

```
$ docker-compose up
Creating network "docker-compose-build_default" with the
default driver
Building app
Step 1/2 : FROM tomcat:9.0.12
9.0.12: Pulling from library/tomcat
38517b303516: Pull complete
...
84398876c738: Pull complete
Digest:
sha256:2c1f49c0e9b5d32cec3ee7ad18b140acc44eeab727a7912928
8d859c1a6931e9
Status: Downloaded newer image for tomcat:9.0.12
 ---> 7c5bb46055d4
Step 2/2 : COPY index.html
/usr/local/tomcat/webapps/ROOT/index.html
 ---> 72e4b1d2276d
Successfully built 72e4b1d2276d
Successfully tagged docker-compose-build_app:latest
WARNING: Image for service app was built because it did
not already exist. To rebuild this image you must use
`docker-compose build` or `docker-compose up --build`.
Creating docker-compose-build_app_1 ... done
Attaching to docker-compose-build_app_1
app_1  | NOTE: Picked up JDK_JAVA_OPTIONS:  --add-
opens=java.base/java.lang=ALL-UNNAMED --add-
opens=java.base/java.io=ALL-UNNAMED --add-
opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
...
app_1  | 21-Oct-2018 23:43:46.830 INFO [main]
org.apache.catalina.startup.Catalina.start Server startup
in 1724 ms
```

But to rebuild the container to pick up any changes, you'll need to use `docker-compose build` or `docker-compose up --build`, just like the message in the output above says.

Analogy of Images and Containers

Let's continue with our analogy comparing container & images to computers and virtual machines to understand what happened when we built a custom image.

```
Analogies to Docker (and Docker Compose) images and
containers:
╔═══════════════╦═══════════════════════════╦═══════════════════════
═══╗
║    Docker     ║         computer          ║ virtual machine
(VM) ║
╠═══════════════╬═══════════════════════════╬═══════════════════════
═══╣
║     image     ║       file-system         ║       VMDK file
   ║
╠═══════════════╬═══════════════════════════╬═══════════════════════
═══╣
║ custom image ║ modified file-system ║  modified VMDK
file  ║
╚═══════════════╩═══════════════════════════╩═══════════════════════
═══╝
```

When we `COPY`'d our `index.html` file into the container, we are analogously adding `index.html` file into a computer's file-system or VMDK file, respectively.

Typically, whenever we build an image we are installing files, packages, libraries, dependencies, etc. on top of the base image.

Once you are more familiar with containers, the house vs apartment analogy described in this post will better capture the nuances of containers:

https://blog.docker.com/2016/03/containers-are-not-vms/

References

How to get started with docker-compose — Zhao Li — Medium

You want to learn how to use docker-compose but are not sure how to go about learning it. To...

medium.com

EE491F Data Driven Web Development

Course material for University of Hawaii at Manoa's EE491F - Data Driven Web Development

zhaol.github.io

Docker | Docker Compose | Containers | How To | Programming