Romin Irani Follow
My passion is to help developers succeed. ¯_(ツ)_/¯
Jul 26, 2015 · 5 min read

Docker Tutorial Series—Part 5 : Building your own Docker Images

This is part 5 of the Docker Tutorial Series.

In this section, we are going to take our first steps to building our own image. We are going to keep it simple at first by adding our software on top of images that are already present. Later on in another chapter, we will look at writing our own files.

Once we have created our image, we will also push this image to the Docker Hub. Keep your Docker Hub username handy. **In case you have not yet registered for the Docker Hub, I suggest that you do so now at the following link.**

What happened to my data?

It is important that we first understand the key difference between containers and images. To summarize it, it is important to remember that "Images are Immutable and Containers are Ephemeral". Let us see that in action now.

Let us begin with starting **boot2docker** utility and then working with the base **ubuntu** image that we have already downloaded earlier. First make sure that you do have ubuntu latest image to get going here.

Fire up the following command:

```
docker@boot2docker:~$ docker images
```

This should give you a list of Docker images that you have and that **ubuntu:latest** should be present on it. If not, I suggest that you do a

```
docker@boot2docker:~$ docker pull ubuntu:latest
```

Let us launch a container from the **ubuntu:latest** image by giving the following command:

```
docker@boot2docker:~$ docker run -it --name mycontainer1
--rm ubuntu:latest
```

This should lead you to a prompt. For e.g. on my machine, I have the following prompt:

```
root@ea503e60bae3:/#
```

Now, let's install the popular Git software on this container instance by running the following commands:

```
sudo apt-get update
```

This should output a stream of messages. Let the process continue its work till you see a message as given below:

```
Reading package lists... Done
```

You will be back at the prompt. Now, let us install Git via the command given below:

```
sudo apt-get install git
```

It will prompt you with a message:

```
Do you want to continue? [Y/n]
```

Please go ahead with a **Y**.

It will take a few seconds to download the Git software and install it. You should finally be at the prompt once everything is done.

To verify that Git is installed, simply type git at the prompt as shown below:

```
$ git --version
```

This should print out the Git version at the console as shown below:

```
git version 1.9.1
```

Now, let us exit the container by typing exit. Keep in mind that we had provided the—rm flag while starting the container, which means that the container is removed on termination.

Now, let us launch another instance of the same ubuntu container as shown below.

```
docker@boot2docker:~$ docker run -it --name mycontainer1
--rm ubuntu:latest
```

Type **git** at the prompt. It displays the message that git is not found:

```
root@42f5b5340f27:/# git
bash: git: command not found
root@42f5b5340f27:/#
```

What happened? Didn't we just install Git on ubuntu and expecting it to be present. The point is that each Container instance launched this way is independent and is depending on the master image i.e. ubuntu:latest, which does not have Git installed.

Committing your Image

To ensure that next time we have a version of ubuntu with Git installed, we need to commit our image. What this means is that we started with a container based on the ubuntu image. Then we added some software to it i.e. git. This means that we modified the state of the container. Hence we need to save that state of the container as an image, so that can relaunch additional new containers from that image. This way they will have the git software installed too.

Let us exit from the container in the previous section and following these steps:

- Launch the container based on ubuntu, this time without the—rm flag

```
docker@boot2docker:~$ docker run -it — name mycontainer1
ubuntu:latest
```

- Update the OS and install Git via the commands given below:

```
$ sudo apt-get update
```

```
$ sudo apt-get install git
```

- Verify that Git is installed via the command given below:

```
$ git — version
```

- Exit the container by typing **exit**.
- Run the docker ps -all command to see the mycontainer1 that we launched:

```
docker@boot2docker:~$ docker ps -all
```

- This should show you the **mycontainer1** that we had launched. It is currently in exited state.
- Commit the container image via the docker commit command as shown below:

```
docker commit [ContainerID] [Repository[:Tag]
```

- In our case, we use the mycontainer1 as the containerid since we have given it a name. If you had not started that container with a name, you could have used the Container Id that is visible in the **docker ps -all** command.
- The Repository name is important. Eventually (and which is what we will do) we will want to push this to the Docker Hub. So the format of the Repository name that is recommended is the following:

```
<dockerhubusername>/<repositoryname>
```

- In our case, the repository-name can be something like ubuntu-git and you will need to substitute your <dockerhubusername> tag above with your Docker Hub user name.
- If you do not give the tag, it will be marked as '**latest**', which is fine for us. For e.g. my Docker Hub user name is rominirani and hence I will commit it in the following manner:

```
docker@boot2docker:~$ docker commit mycontainer1
rominirani/ubuntu-git
```

- This will give you back the image ID.
- Check the list of docker images

```
docker@boot2docker:~$ docker images
```

- You should see at the top of the list an image that is like the following entry that I have:

```
docker@boot2docker:~$ docker images
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
rominirani/ubuntu-git latest 23cf273fafed About a minute
ago 247.1 MB
```

This shows that our Repository **rominirani/ubuntu-git** has got created.

Launching a Container from your Image

We can now launch a container from our newly created Docker image i.e. **yourusername/ubuntu-git** via the docker run command.

```
docker@boot2docker:~$ docker run -it — name c1
<yourusername>/ubuntu-git
```

The above command will launch the container based on our image. It will then take you to the prompt, where you can verify that it comes with Git by giving the git—version command.

Pushing the Image to the Docker Hub

To push the image to the Docker Hub via the following steps:

1. Get an account at Docker Hub
2. From **boot2docker** prompt, execute the command **docker login** and follow the instructions for your username and password.
3. On successful login, you should get a **Login Succeeded** message.
4. To a docker push as shown below:

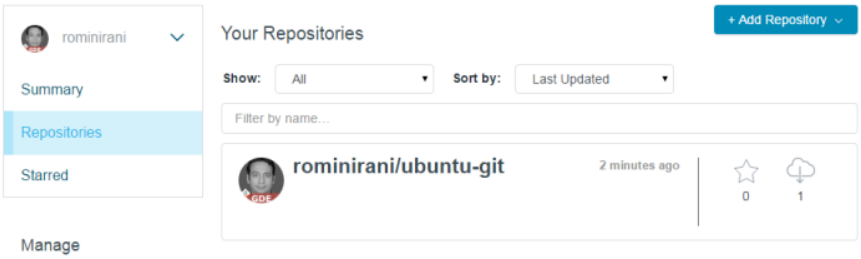
```
docker@boot2docker:~$ docker push <yourusername>/ubuntu-
git
```

This will take a while to upload the details. Note that Docker is smart in the sense that it already will detect if the base image i.e. ubuntu already

exist and hence it will cleverly chose only those layers that need to be uploaded. A sample run of the process is shown below:

```
docker@boot2docker:~$ docker push rominirani/ubuntu-git
The push refers to a repository [rominirani/ubuntu-git]
(len: 1)
0644edf7f9c6: Image already exists
23cf273fafed: Pushing 5.505 MB/37.93 MB
23cf273fafed: Image successfully pushed
d0955f21bf24: Image successfully pushed
9fec74352904: Image successfully pushed
a1a958a24818: Image successfully pushed
f3c84ac3a053: Image successfully pushed
511136ea3c5a: Image successfully pushed
Digest:
sha256:397eac492cd67b1e4b1371fc74a676c58d059756fe20022f6
346d6f99b908a1b
docker@boot2docker:~$
```

Check for the repository in Docker Hub (web):



You can also search for the repository via the docker search command. Try the following command:

```
docker@boot2docker:~$ docker search ubuntu-git
```

You should be able to locate your repository in the search results.

Exercise: Try out some other repositories, put your files on top of it and then commit your own image.