Docker Primer

Steak Electronics

08/30/19

Contents

1 Overview

Docker is a program in the tradition of Virtualization. However, Docker differs from a Virtual Machine, in that it uses less resources, and allows for more containers (essentially isolated OS') to run. With Docker I can run 8 different websites on a Core 2 Duo. Docker makes it trivial to transport these websites to a new machine. It's as simple as copying the docker-compose (you should be using docker compose) yml configuration, and keeping all permenant volumes, and saved files in one folder. Docker is a little more setup upfront, but promises savings in time plus interest down the line.

2 General Notes

It always helps to read a book on a subject, and then keep it as a Reference. I have read "Using Docker" By Adrian Mouat. It is a decent book. Not bad.

Here are some general tips:

First off, Docker is 64 bit only for i386 architecture. ARM has a separate build. There is no 32 bit, unfortunately.

You will want 'some' RAM. I had 1GB on a P4 machine, and that was not enough. 4GB was enough.

You should always use docker compose. If you read the book above, you will understand why. Docker can run on the command line (commands are

somewhat complex for each container), but with a compose file, you can write everything down in a much simpler fashion. Use compose. It's a separate install, currently. Install it.[1]

One of the benefits of docker, is its simplicity. There are essentially two commands you will ever need to know to use docker. Both must be run as root. One is #docker (e.g. docker restart container_name_here). The other is docker-compose (e.g. #docker-compose up -d).

2.1 Docker Commands Reference

Here is just the good stuff.

- docker-compose up -d (starts the containers in the docker compose file, if they aren't already started. the -d detaches from the stdout logging. You don't need to use stdout logging, you can use docker logs, but its there if you want it)
- docker ps (lists containers running. If one fails to start, you'll see it missing from here)
- docker logs ¡containername¿ (gives you some logging output from the container. Often enough to troubleshoot.)
- docker exec -it ¡containername¿ /bin/bash (this will get you in a shell in the docker container. From here you can do what you need to. Most are debian, and need apt-get install less nano or whatever program you are missing. Ping is missing from possibly all containers, so if you want to test via ping, you'll have to apt-get it).
- docker-compose restart (this will restart all containers. However, I don't recommend it. Initting containers can get corrupted this way, and also its much easier to restart a single faulty container via...)
- docker restart ¡containername¿ (this will restart one single container.)
- docker cp ¡containername¿:/dir/to/file dest (you can copy files from local machine to docker, or vice versa with this. Extremely useful).

[1] Seriously, just ignore the docker command lines. I consider them useless. More of a red herring for rookies.

Less often, you might want to know docker kill ¡containername¿ and docker rmi ¡containername¿. The first will stop a container, the second will remove an image. If you corrupt the install of a container, the second will save you. Alternatively, you can just install a container of the same type with a new name.

3 Specific Tips

3.1 YAML is space sensitive

When you edit the .yml file for docker-compose, you have to hit spaces in a certain pattern (tabs not allowed). This is absurd, but just be aware. The errors are cryptic, and its often just because the spacing doesn't stick to what it expects.

3.2 If you restart a containers namesake process, it will probably restart / reset the container

So if you are troubleshooting an apache container, you edit some files, then /etc/init.d/apache2 restart, uh oh... You just undid all the edits you made, if they aren't in a permanent volume. You can shell in, make edits, and then exit the shell, but a service restart often resets the container.

3.3 Use a single reverse proxy, to handle multiple websites

There are many ways to do this. I use an nginx proxy from scratch. You can also use some containers that are built for this purpose (I personally think it's bloated but a lot of people use Jason Wilder's proxy) [2]

3.4 If you use a single reverse proxy, Lets Encrypt can be done real easy

In this case scenario you would have certbot on the host and a local volume that the proxy has access to which is the webroot of the Lets Encrypt scripts. The nginx proxy entry look something like this:

```
location ^~ /.well-known {
     alias /var/www/html/.well-known/;
      autoindex on;
```

[2] https://github.com/jwilder/nginx-proxy - A lot of people swear by this, but I think it's straying too far from the motorcycle.

```
}
```

And this is put in every server declaration of nginx.conf. Real simple, real easy. The docker compose of the nginx proxy is something like:

```
nginx:
    image: nginx:latest
    container_name: custom_name_for_my_proxy
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
      - /etc/letsencrypt/:/etc/letsencrypt/
      - ./webroot/:/var/www/html/
```

The volumes section is extremely simple, don't be scared. There are two entries. Local and remote. You specify what folder will be the local directory which will be cloned to the host at the remote path you specify. So, the host runs certbot at /etc/letsencrypt, and this folder is cloned to the nginx proxy container, at the same location. Finally, webroot must be set in certbot, but it prompts you for this[3]

3.5 Give every Container a Containername

This makes it easier to refer to them later. All you need to do in the compose file is include container_name: something. Much better than the gibberish they give these names if you don't include it.

3.6 Beware of Interrupting Initting Containers

When you first build a container, it might take 30-60 or more seconds to do whatever it needs to do. If, before then, you restart it... It may get corrupted. This has happened to me more than once. When you are testing a new container, and it doesn't seem to work for some inexplicable reason, create a container with a new name (it will create a new one), or delete the first one, and start it again.

3.7 Put Apache or Program logs from the Container in a volume that is locally accessible

This means you want a volume something like ./containerA_files/logs:/var/www/log/apache2/ so that you can monitor the logs from your host machine easily. docker logs doesn't have everything.

[3] And if you forget or get it wrong, it can be configured somewhere in /etc/letsencrypt. it's a one liner text entry).

3.8 Only Restart Containers you need to Restart

You can restart everything with docker-compose restart, but it's faster, and less prone to break initting containers, if you docker restart containername. Do the latter.