From Zero to Docker

Training | 2019.05.16 | Mário Dagot, Jorge Dias

Docker is an open platform for developing, shipping, and running applications. Through the course of this training we will guide you to the most common feature and use cases of docker. Take this as an introduction and an opportunity to dive into the docker world.

AGENDA

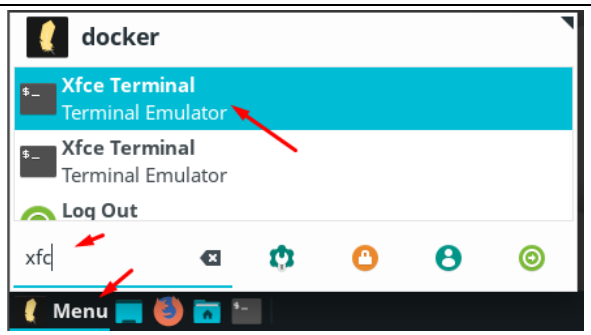
- 01 - Install Vim and Terminator and VSCode
- 02 - Install Docker CE for Ubuntu
- 03 - Hello from Busybox
- **04 - Webapp with Docker**
- 05a - Webapp with Docker - My first Dockerfile – Nginx
- 05b.1 - Webapp with Docker - My first Dockerfile - Dotnet Core
- 05b.2 - Webapp with Docker - My first Dockerfile MultiStage - Dotnet Core
- 06 - Save and Restore and Push to Docker Hub
- 07a - Webapp with database integration - My first network – SpringBoot
- 07b - Webapp with database integration - My first docker-compose – SpringBoot

04 - WEBAPP WITH DOCKER

Objective

- Learn about nginx web server
- Learn how to run containers in detach mode
- Learn about base images
- Learn how run command inside a container
- Learn how to expose container ports to the host environment

Step by Step

Open a terminal windows	

Pull nginx docker image. Nginx is a web server, similar to the well-known apache web server	docker ~ docker pull nginx Using default tag: latest latest: Pulling from library/nginx 743f2d6c1f65: Pull complete 6bfc4ec4420a: Pull complete 688a776db95f: Pull complete Digest: sha256:a08f3331865d6072d7a28a5b943d1526dbd3fe3b4ca723c0438750d5e23f21a1 Status: Downloaded newer image for nginx:latest
List images, you will see that the image is now available	docker ~ docker images REPOSITORY TAG IMAGE ID CREATED SIZE nginx latest 53f3fd8007f7 6 hours ago 109MB busybox latest af2f74c517aa 5 weeks ago 1.2MB
Let's create a new running instance of the nginx container. We use -d to allow the container to run in detach mode. This way it will run in the background.	docker ~ docker run -d --name mynginx nginx d6e29cb571aac41b19a13feae76d96426d8dc7c2308bc3379eca054512e388b2
We can list the running containers using the following command. Similar to the docker ps -a command we have been using.	docker ~ docker container ls CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES d6e29cb571aa nginx "nginx -g 'daemon of…" 8 seconds ago Up 7 seconds 80/tcp mynginx
Let's use curl to perform a HTTP request to port 80 at localhost. HINT: Alternatively you can open your browser and go to the same address. We should get a connection refused. Again, as we saw before, docker containers live in isolation and if we don't explicitly allow access they cannot see or be seen by the host.	docker ~ curl localhost:80 curl: (7) Failed to connect to localhost port 80: Connection refused
Let's remove the container. We can use the command	docker ~ docker container rm mynginx

we have used several times during our course. An error! What happened?! The docker engine doesn't allow the removal of running containers. This is precaution measure. Remember, containers by default are ephemeral. Once they are removed we will lose all saved state. Be warned!	`Error response from daemon: You cannot remove a running container d6e29cb571aac41b19a13feae76d96426d8dc7c2308bc3379eca054512e388b2. Stop the container before attempting removal or force remove`

We can stop the container and then remove it.	`docker ~ docker container stop mynginx`
	`mynginx`
	`docker ~ docker container rm mynginx`
	`mynginx`

Using the -p 8080:80 we can expose the nginx 80 port to the host on port 8080. Basically, when we access localhost at port 8080 it will be as if we were accessing the port 80 inside the container. How cool is that?	`docker ~ docker run -d --name mynginx -p 8080:80 nginx` `97a9e96fa765c7eedb67bdbd0a6ff74e98fbec40ec0d837c658e18afb135ace7`

Doing a docker ps we have the confirmation that the container is running and the we have the port mapping.	`docker ~ docker ps -a`
	`CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES`
	`97a9e96fa765 nginx "nginx -g 'daemon of…" 8 seconds ago Up 7 seconds 0.0.0.0:8080->80/tcp mynginx`

Try now to access the previous URL.
 	
Success!
	`docker ~ curl localhost:8080`
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<title>Welcome to nginx!</title>`
	`<style>`
	`  body {`
	`  width: 35em;`
	`  margin: 0 auto;`

```
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

All docker images have a base image as its parent. The nginx image is based on debian.

Since it's debian based we can try to gain access to its internal filesystem using bash (or if it doesn't exist sh)

We use docker exec to perform commands on a running container. The -it flag allow us to use bash in interactive mode.

Play a bit around.

You can change the filesystem and see its impact. Lets try to edit the index.html of the base nginx webserver.

We use sed to find&replace a specific phrase.

```
docker ~ docker exec -it mynginx bash
root@97a9e96fa765:/# cd /usr/share/
X11/            bash-completion/ debianutils/     dpkg/           gdb/
libc-bin/       menu/            pam-configs/     terminfo/
adduser/        bug/             dict/            fontconfig/     info/
lintian/        misc/            perl5/           xml/
base-files/     common-licenses/ doc/             fonts/          java/
locale/         nginx/           pixmaps/         zoneinfo/
base-passwd/    debconf/         doc-base/        gcc-6/          keyrings/
man/            pam/             tabset/
root@97a9e96fa765:/# cd /usr/share/nginx/html/
root@97a9e96fa765:/usr/share/nginx/html# ls -l
total 8
-rw-r--r-- 1 root root 494 Apr 16 13:08 50x.html
-rw-r--r-- 1 root root 612 Apr 16 13:08 index.html
root@97a9e96fa765:/usr/share/nginx/html# sed -i 's/Welcome to nginx!/Welcome to
nginx, from docker!/g' index.html
root@97a9e96fa765:/usr/share/nginx/html# exit
exit
```

Curl again the url.
 	
Voila. We do see our changes in index.html.
	docker ~ curl localhost:8080
	<!DOCTYPE html>
	<html>
	<head>
	<title>Welcome to nginx, from docker!</title>
	<style>
	  body {
	  width: 35em;
	  margin: 0 auto;
	  font-family: Tahoma, Verdana, Arial, sans-serif;
	  }
	</style>
	</head>
	<body>
	 Welcome to nginx, from docker!
	 If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
	
	 For online documentation and support please refer to
	nginx.org.
	
	Commercial support is available at
	nginx.com.
	
	 Thank you for using nginx.
	</body>
	</html>

What if I want to have multiple instances of nginx running on the same machine?! No problem. Again, each container is running in isolation. Just do a docker run again. Beware and expose to a different local port on the host. Just think broadly:	docker ~ docker run -d --name mynginx-new -p 8081:80 nginx 2019676d293d39449fcc126fc1aec4fb128c69f89328be363dee04a717c3b3de docker ~ docker container ls CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 2019676d293d nginx "nginx -g 'daemon of…" 9 seconds ago Up 8 seconds 0.0.0.0:8081->80/tcp mynginx-new 97a9e96fa765 nginx "nginx -g 'daemon of…" 3 minutes ago Up 2 minutes 0.0.0.0:8080->80/tcp mynginx docker ~ curl localhost:8081 <!DOCTYPE html>

Multiple versions of the same application running on the same machine
Consistent setup – each container instance of the same image have the same configuration internally
No library conflicts
Easy to install & remove
 	
Great isn't it?! But there's more, just keep on.
	<html>
	<head>
	<title>Welcome to nginx!</title>
	<style>
	  body {
	  width: 35em;
	  margin: 0 auto;
	  font-family: Tahoma, Verdana, Arial, sans-serif;
	  }
	</style>
	</head>
	<body>
	 Welcome to nginx!
	 If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
	
	 For online documentation and support please refer to
	nginx.org.
	
	Commercial support is available at
	nginx.com.
	
	 Thank you for using nginx.
	</body>
	</html>

How to copy to and from the container?
 	
Easy… Let's copy the index.html from the first container. Then do some changes.
	docker ~ docker cp mynginx:/usr/share/nginx/html/index.html index.html
	docker ~ ls -l index.html
	-rw-r--r-- 1 docker docker 638 mai 8 09:55 index.html
	docker ~ vi index.html
	docker ~ cat index.html
	<!DOCTYPE html>
	<html>
	<head>
	<title>Welcome to nginx, from docker!</title>
	<style>
	  body {
	  width: 35em;

```
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx, from docker in a second container!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

We can copy now, from the host to the second container.

Did it work?

Curl the localhost and the port exposed for the second nginx container.

Nice!

```
docker ~ docker cp index.html mynginx-new:/usr/share/nginx/html/index.html
docker ~ curl localhost:8081
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx, from docker!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx, from docker in a second container!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```



```
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Lessons learned

We learned how to start the nginx container and execute commands on the running container. The containers has a base image, like debian, Ubuntu, etc, and due to that is also possible to run a linux shell and access the containers as if it was a virtual machine or remote server.

As we already knew, containers are isolated from the outside world. We can however expose ports from the container and make them accessible to the host environment.

Another advantage of the isolation characteristic is that we can spin up as many instances as we want and they are completely independent of each other. No conflicts, same configuration, etc.

Revision History

Version	Date	Author	Description
1.0	2019.05.01	Mário Dagot, Jorge Dias	Initial Version