Creating a new Docker Image/container from existing Container

-And run the new container

**Step 1: Create a initial Base Container – for an App**

* Let’s get started by creating a running container.
* So that we don’t get bogged down in the details of any particular container,
* we can use [nginx](http://nginx.org/).
* The Docker create command will create a new container .

docker create --name nginx\_base -p 80:80 nginx:alpine

****

**Step 2: Inspect Images**

* If you look at the list of images on your system
* you will now see the nginx:alpine image:

docker images -a

****

**Step 3: Inspect Containers- created but not started**

* Note here that the container is not running
* so you won’t see it in the container list unless you use the -a flag (-a is for all).

****

**Step 4: Start the container – which should expose port 80**

* Let’s start the container and see what happens.

$ docker start nginx\_base

nginx\_base



**Step 5: Access the container App - from browser with localhost:**

* The browser will show the default message “Welcome to nginx”

****

**Step 5: Modify the Running Container – by creating index.html**

* Create a file index.html with content “Hello World” in the local directory in local host
* And copy this file in to container location /usr/share/nginx/html/index.html

$ pwd

/c/Users/Shanmugavelu/

Check the file in the container by connecting to container 🡪 before copy

$ winpty docker exec -it nginx\_base sh

/ #

/ # cd /usr/share/nginx/html/

/usr/share/nginx/html # ls -ltr

total 8

-rw-r--r-- 1 root root 615 Feb 14 16:20 index.html

-rw-r--r-- 1 root root 497 Feb 14 16:20 50x.html

/usr/share/nginx/html #

/usr/share/nginx/html # cat index.html

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

html { color-scheme: light dark; }

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>

$ docker cp index.html nginx\_base:/usr/share/nginx/html/index.html

****

****

**Step 6: Commit the Running Container nginx\_base**

* create a new image using the commit command.

$ docker commit nginx\_base

sha256:284608bfc7628ea86036fbbac9cbf0038dcb2322f161a8730e630ee6c29e44a4

****

**Step 7: Create the TAG for the newly created image:**

docker tag 284608bfc762 hello\_world\_nginx

****

**Step 8: Stop and remove the initial base container which is running.**

****

**Step 9: Start the new App container with the new image**

* Create and start the new container using the run command .

docker run --name hello\_world -p 80:80 hello\_world\_nginx

****

Shanmugavelu@DESKTOP-QQ3CB6K MINGW32 ~ (master)

$ docker run --name hello\_world\_cont -d -p 80:80 hello\_world\_nginx

38a152f386311ae2453b1297e78a0a536b08738ae1fcf70c392b0564cc909704

Shanmugavelu@DESKTOP-QQ3CB6K MINGW32 ~ (master)

$ docker start hello\_world\_cont

hello\_world\_cont

Shanmugavelu@DESKTOP-QQ3CB6K MINGW32 ~ (master)

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

38a152f38631 hello\_world\_nginx "/docker-entrypoint.…" 37 seconds ago Up 36 seconds 0.0.0.0:80->80/tcp hello\_world\_cont

Shanmugavelu@DESKTOP-QQ3CB6K MINGW32 ~ (master)

$

**Step 10: Access the new App container**

 **- from browser with localhost:**

* The browser will show the new message “Hello World”
* This is as per modified container image , where we added the index.html with “Hello World” message.

****

##

## **Drawbacks of Using Docker Containers**

Some of the drawbacks of using Docker containers include:

1. **Security concerns:**

Containers share the host operating system, so if a container is compromised, the host can also be affected. This is why proper measures must be taken to ensure the security of containers.

1. **Limited Resource Management:**

Containers rely on the host for resource management, so there may be limitations on the resources available to each container, such as CPU, memory, and storage.

1. **Networking Complexity:**

Networking between containers can be complex, especially in multi-host deployments.

1. **Persistent Storage:**

 Persistent storage can be a challenge with containers, as containers are ephemeral by nature and data can be lost when a container is deleted or recreated.

1. **Scalability:**

 Scaling containers can be difficult, as each container runs in isolation, and multiple containers must be coordinated to scale an application.

1. **Support for Legacy Applications:**

 Legacy applications that are not designed to run in containers may not work well in a containerized environment.

These are some of the challenges that can be faced while using Docker containers, but with proper planning and management, these challenges can be overcome to achieve the benefits of containerization.

## **Written by Vinod Sairam**

The Database Administrator who helps to manage Enterprise Databases who develops Devops Engineering ideas to support Database tasks, Infrastructure tasks and to reach the right audience.

**Assisted by: Shanmugavelu (Database-Devops Engineer)**