**Agenda: Getting Started with Docker**

* Installing Docker for Windows 10 Professional
* Docker Images for .NET Core.
* Docker CLI Commands
  + Managing Images
  + Managing Containers
* Creating Images from Container

**Docker Commands**

**Listing all images from local registry:**

* docker version
* docker pull hello-world #Pull the image hello-world from the Docker registry
* docker pull nginx
* docker pull alpine
* docker pull ubuntu
* docker inspect nginx
* docker images OR docker image ls
* docker rmi nginx
* docker images --no-trunc #List full-length image IDs
* docker images --filter=reference=alpine
* docker image alpine

**Basic Docker Commands:**

1. docker pull hello-world
2. docker container run hello-world # Run a New Container
3. docker run hello-world # Run a New Container
4. docker ps # List only **Running** Containers
5. docker container ls
6. docker ps -a # **List all** containers including the stopped ones.
7. docker run hello-world # Run a New Container
8. docker run --name **hello** hello-world # The container name is hello if it doesn't exist.
9. docker rm <container-id> # Remove the container
10. docker run **--rm** hello-world # Remove the container as soon as the container stops.
11. docker container prune # Delete all stopped/exited containers
12. docker container prune --filter "until=24h" # Remove only docker stopped containers older than 24 hours
13. docker run -d --name **h1** hello-world #Run container in detached mode
14. docker logs **h1**
15. docker inspect alpine #Note the Cmd value
16. docker run alpine #Container will start and immediately stop
17. docker run alpine **ls -l**  # To override the default cmd
18. docker run alpine **echo "Hello"**
19. docker ps -a

1. docker run **-it** alpine ~~/bin/sh~~
   1. ls -l
   2. echo "hello" > hello.txt
   3. ls
   4. ps
   5. exit
2. docker ps -a #Note the container has exited.
3. docker start <container ID>
4. docker attach <container ID>
   1. ls # Note hello.txt is existing
   2. ctrl + P + Q #To detach from the shell, switching to host.
5. docker ps #Note container is running (not exited)
6. docker **top** <container id> #Display the running processes of a container
7. docker stop <container id>
8. docker ps -a

**To run a sample web application:**

1. docker run -p 8001:80 nginx

# Start a new Terminal / Command Window

1. docker ps
2. docker inspect <container-id>
3. curl <container-ip> - Fails to connect because container is in network with WSL or Linux VM Host machine and NOT with our local machine.
4. curl localhost:8001
5. docker run **-d** --rm -p 8080:80 nginx

-p publishes a port. The IIS image is built to allow traffic in on port 80. This maps port 8080 on the host to port 80 in the container

-d starts in detached mode, Docker runs the container in the background and monitors it

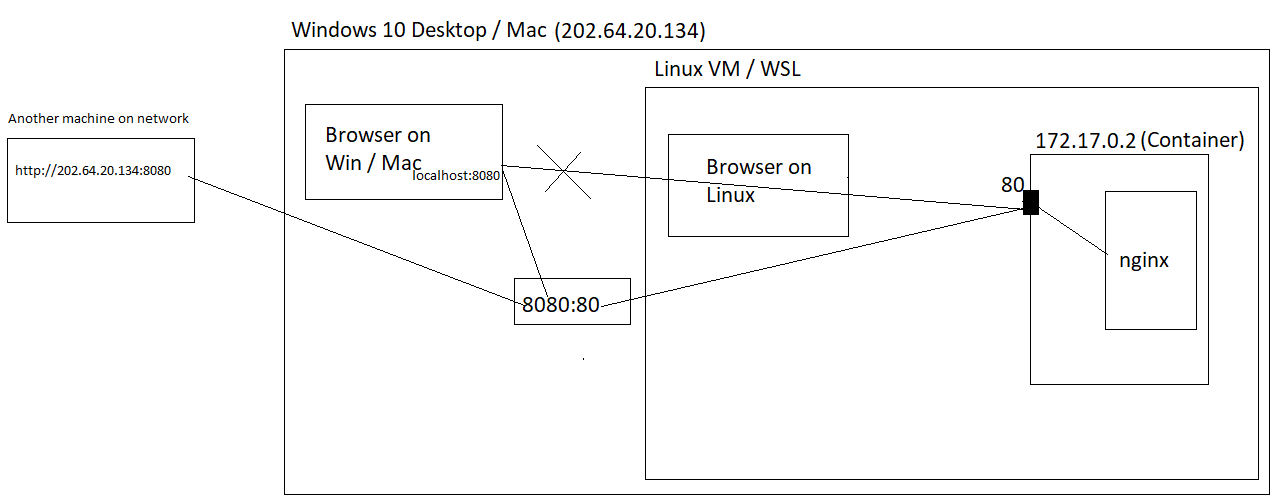
1. docker ps
2. curl http://localhost:8080
3. docker logs <container-id>

**To get terminal access of a container in running state**

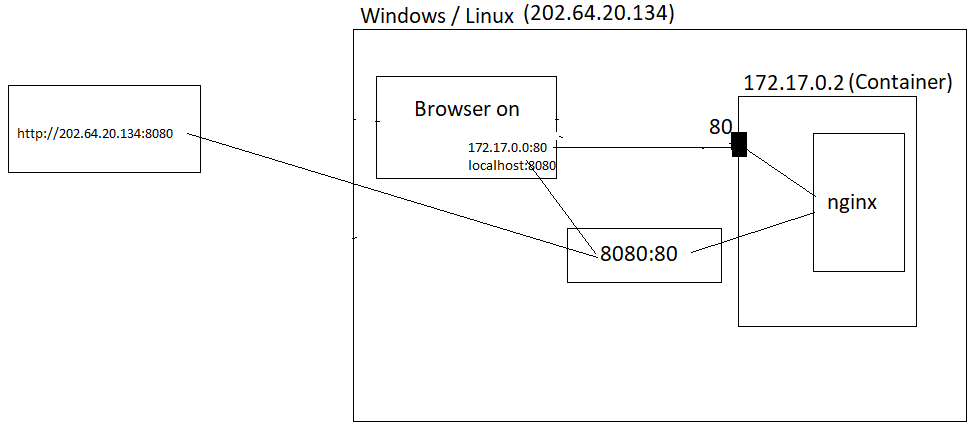
**For Linux:** docker exec -it <containername> **/bin/bash**

**For Windows:** docker exec -it <containername> **cmd.exe**

**If containers are run on Linux VM or WSL on Windows 10:**

****

**If containers are run directly on Windows / Linux Machine:**

****

**Exec Command**

* Start a new container from the nginx image on Docker Hub
* Using exec, get into the container and have a look around
* Visit the web site and make sure you see the default home page.
* Now go change the home page index.html file located in **/usr/share/nginx/html**.
* Add file **/usr/share/nginx/html/hello.html (use echo)**
* Verify that your changes show in the web site.

**Solution:**

**Nginx Layers**

"sha256:ed7b0ef3bf5bbec74379c3ae3d5339e666a314223e863c70644f7522a7527461",

"sha256:fb6d57d46ad57cbb50dfe16eba53a31d8808aa43e3a4a3ddd5c04d2d6ee0ecc5",

"sha256:935b5bd454e1973eb283fbc76d5613def62c190c8c4cc621576ef32f7c7dcb56",

"sha256:f12d4345b7f35505fc42fd613ae2c6749ddba30d6c9025e1a5b070c97c1ad2bb",

"sha256:79974a1a12aa3a3876d8dbcf16b2eda15ba5372574d38dd94bac68d339b6e124",

"sha256:9d907f11dc742442217bda25fde38f11851c7d495cacc6fc4e3869f4cd2c9ad9"

1. docker run -p 8080:80 -d --rm nginx
2. docker ps
3. docker exec -it <container id> /bin/sh
4. # cd /usr/share/nginx/html
5. # echo "hello" > "hello.html"
6. # exit
7. curl <http://localhost:8080>/hello.html
8. mkdir temp #Create a directory on Host
9. cd temp
10. echo This is one > one.html
11. echo This is two > two.html
12. docker ps
13. **docker cp . <container-id>:/usr/share/nginx/html**
14. **docker exec -it <container-id> ls /usr/share/nginx/html**
15. curl <http://localhost:8080/one.html>
16. curl <http://localhost:8080/two.html>
17. **docker diff <container-id> #Changes to the filesystem inside the container after it was created.**
18. docker **commit** <container id> mynginx
19. docker inspect mynginx

mynginx layers:

"sha256:ed7b0ef3bf5bbec74379c3ae3d5339e666a314223e863c70644f7522a7527461",

"sha256:fb6d57d46ad57cbb50dfe16eba53a31d8808aa43e3a4a3ddd5c04d2d6ee0ecc5",

"sha256:935b5bd454e1973eb283fbc76d5613def62c190c8c4cc621576ef32f7c7dcb56",

"sha256:f12d4345b7f35505fc42fd613ae2c6749ddba30d6c9025e1a5b070c97c1ad2bb",

"sha256:79974a1a12aa3a3876d8dbcf16b2eda15ba5372574d38dd94bac68d339b6e124",

"sha256:9d907f11dc742442217bda25fde38f11851c7d495cacc6fc4e3869f4cd2c9ad9",

**"sha256:c58b887ef842a5cd922e8282971ae4da783e021d88b7063195714101550473e3"**

1. docker run -d -p 8081:80 mynginx
2. curl <http://localhost:8081/hello.html>
3. docker exec -it <container-id> bash
   * cd /usr/share/nginx/html
   * rm one.html
   * exit
4. docker diff <container-id>
5. docker commit <container-id> mynginx2.
6. docker inspect mynginx2

**mynginx2:**

"sha256:ed7b0ef3bf5bbec74379c3ae3d5339e666a314223e863c70644f7522a7527461",

"sha256:fb6d57d46ad57cbb50dfe16eba53a31d8808aa43e3a4a3ddd5c04d2d6ee0ecc5",

"sha256:935b5bd454e1973eb283fbc76d5613def62c190c8c4cc621576ef32f7c7dcb56",

"sha256:f12d4345b7f35505fc42fd613ae2c6749ddba30d6c9025e1a5b070c97c1ad2bb",

"sha256:79974a1a12aa3a3876d8dbcf16b2eda15ba5372574d38dd94bac68d339b6e124",

"sha256:9d907f11dc742442217bda25fde38f11851c7d495cacc6fc4e3869f4cd2c9ad9",

"sha256:c58b887ef842a5cd922e8282971ae4da783e021d88b7063195714101550473e3",

**"sha256:c4efa4b124780e7d77f3b5f0cc0003fec7745eb5a8428002c375979cc31f47fa"**