DOCKER Containers

DOCKER Container

What is a Container?

It is a way to package applications with all necessary dependencies and configuration.
It is a portable artifact easily shared and moved around.
Using containers makes development and deployment process quiet efficient.

Where do containers live?

They live in a Container Repository (a special type of storage for containers), many companies have very own private repositories to host and store containers. There are also public repositories (Docker and DockerHub) for docker container where we can find and use the containers we want.

Basic Docker Commands

docker pull -> Used to pull latest version of an application to the system.
docker run -> Runs the pulled image in a docker container.
docker run --options -> provides various modes to run a container image.
docker stop -> takes the container id that needs to be stopped.
docker start -> takes the container id that needs to be started.
docker ps -> Provides a list of running docker containers.
docker exec -it -> takes a container id and the preferred CLI to open a CLI for the given container.
docker logs -> takes a container id to display the logs of the particular container.

Docker in Practice

E.g.: Working on a simple JS app with mongoDB and mongoDB Express containers.

Step 1: Create a JS app.
Step 2: Pull mongoDB and mongoDB Express containers from DockerHub.
Step 3: Connect both the containers (using Docker Network).
Step 4: Docker provides certain network by default, to check run "docker network ls".
Step 5: Create a docker network, to do so run:"docker network create <network_name>"

Step 6: Run the docker container for mongoDB, run "docker run -p 27017:27017 -d <environment_variables> --net <network_name> mongo"(has mongo ports, environment variables can be viewed from dockerhub-mongo)

Docker SetUp

Docker Prerequisites:

- Virtualization needs to be enabled.
- Requires Microsoft Hyper-V to run. If not running Docker installer will enable it.
- Once Hyper-V is enabled, VirtualBox will no longer work.
- Works natively on Windows 10 (64bit).
- Docker for Windows include: Docker Engine, Docker CLI client, Docker Compose, Docker Machine & KiteMatic.

Installation for Windows:

- Check Prerequisites.
- Download "Get Docker for Windows[Stable]".
- Once downloaded click on the installer and complete installation.
- Docker for Windows.
- Start Docker manually by searching Docker app and click on it.

Working with Docker

In order to create a docker image from an application we have to copy the contents of the application into the docker file which can be an artifact that was built.

Blueprint for building images, which is called a docker file.

Build a DockerFile:

Step 1: For the image that we build, we will need to base it on another image.
Step 2: Set Environment variables
Step 3: Set RUN,COPY and CMD
Step 4: Save the docker file in the name of "Dockerfile only"

Build a Docker Image from the DockerFile:

Step 1: Check the DockerFile for pre-requisites
Step 2: Run "docker build -t <img-name>:<tag> <directory of Docker-file>"
//Tag can be taken to be the version of an app

Step 3: Once the command is run you get the id when the docker image is successfully created

Delete Docker Image:

Execute: "docker ps -a"	//Gets containers
Execute: "docker rm <container-id>"	//Removes Container
Execute: "docker rmi <img-id>"	//Removes Docker image

Run the Docker image:

Execute: "docker run <img-name>:<tag>"

Open the integrated terminal of the container: docker exec -it <container-id> /bin/sh

Dockerfile syntax:

FROM <docker-image>	
ENV <environment-variables>	
RUN <linux-command>	//Executes inside the container
COPY <directory>	//Executes on the host
CMD ["node","/home/app/server.js"]	//Executes the entry point Linux command. This line translates to "node server.js"...