**Introduction to Docker**

**Container**

* Containers share the machine’s OS system kernel and therefore do not require an OS per application
* A container is a standard unit of software that packages up
  + Code
  + Dependencies
* Containers run from a directory
  + /var/lib/docker – Lists container, or rather image id
* Containers run directly from images
* Container is a process running from a directory, and all its data is coming from an image

**Docker Container**

Docker containers that run on Docker Engine:

* Standard: Docker created the industry standard for containers so that they could be portable anywhere
* Lightweight: Containers share the machine’s OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs
* Secure: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry

**Docker Image**

Docker Hub is a registry for docker images

* A stopped container like vm image
* Consist of multiple layers
* An app will be bundled in an image
* Containers run from images
* Images are called repositories in registries

**Docker Registries**

* Storage for Docker images
* Dockerhub is default registry
* Cloud-based registries
  + Dockerhub
  + GCR (Google Container Registry)
  + Amazon ECR
* In-house or Local Registries
* Nexus 3+
* Jfrog Artifactory
* DTR (Docker Trusted Registry)

**VM vs Container**

* Container offers isolation, not virtualization
* Containers are OS virtualization
* VMs are Hardware virtualization
* VM needs an operating system
* Containers don’t need an operating system
* Containers use Host operating system for computer resource

**Docker Commands**

* **docker images** – To list all the images in the local machine
* **docker run <image name>** - To create a container from the image name
* **docker stop <image name or ID>** - To stop the images process
* **docker ps** – To check the running container
* **docker ps** –a – To list all containers and dead containers.
* **docker exec <container name> <command>** – executes commands on containers/attached containers
* **docker start/restart/rm <container name/ID> –** To start, restart, or remove docker
* **docker rmi <image name>** – To remove docker images
* **docker inspect** – Details of container & image
* **docker pull <image name> -** Pulls the image from docker hub by defaults
* **docker run –name <container name> -p <host port eg 7090:<container port 80> –d <image name>** - This runs the image nginx, with option –name enables us to give the container a name. With –d options allowing us to run the container in the background. Lastly the host port and container port 80.
* **docker exec –it <container name> /bin/bash** – To attach to a container process (like ssh)
* **docker inspect <image name>** - To know the image volume, process it runs, port, etc
* **docker logs <container name>** - To see the logs of the container
* **apt install procps** – To install the process command

**Steps for deleting Docker image**

* **docker stop <container ID>** - Stops the container
* **docker rm <container ID>** - Removes the container
* **docker rmi <image name>** – Removes the image

**Container Volumes – Persistent Storage for volatile containers**

**Container Data**

* The data doesn’t persist when that container no longer exists, and it can be difficult to get the data out of the container if another process needs it
* A container’s writeable layer is tightly coupled to the host machine where the container is running. You can’t easily move the data somewhere else.

**Two ways to store files in the container host machine**

* Volumes
  + Managed by Docker (/var/lib/docker/volumes/) on Linux
  + **docker volume create <volume name>** - To create a volume
* Bind Mounts
  + Stored anywhere on the host machine
  + **docker run --name mysql-db -v /home/ubuntu/vprodbdata:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=my-secret-passwd -d -p 3030:3306 mysql:5.7** – Command to create a Bind Mounts

**Building Docker Images**

**Dockerfile Instructions**

* **FROM** - Base Image
* **LABELS** - Adds metadata to an image
* **Run** - Execute commands in a new layer and commit the results
* **ADD/COPY** - Add files and folders into an image
* **CMD** – Runs binaries/commands on docker run
* **ENTRYPOINT** – Allows you to configure a container that will run as an executable
* **VOLUME** – Creates a mount point and marks it as holding an externally mounted volumes
* **EXPOSE** – Container listens on the specified network ports at runtime
* **ENV** – Sets the environment variable
* **USER** – Sets the username or (UID)
* **WORKDIR** – Sets the working directory
* **ARG** – Defines a variable that users can pass at build time
* **ONBUILD** – Adds to the image a trigger instruction to be executed at a later time

**Building Image Command**

* **docker build –t <image name> <image path name>** - To build a docker image