**Docker**

Docker is a platform to develop, deploy, and run applications inside containers. It provides a way to run applications securely isolated in a container, packaged with all its dependencies , settings and libraries and allows them to be portable among any system.

Docker container is an open source software development platform. A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it. For example, differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure.

It is available for both Linux and Windows based apps, containerized software will always run the same, regardless of the environment.

**How is id different from Virtual Machine**

Virtual machines are the precursors to Docker containers. Virtual machines also isolate an application and its dependencies. However, Docker containers are superior to virtual machines because they take fewer resources, are very portable, and are faster to spin up.

The main difference between containers and VMs is in their architectural approach. Containers \*share\* the host system’s kernel with other containers. VMs run on top of a physical machine using a “hypervisor” which itself acts as host machine which has its own operating system .

**Why docker is important**

**1. Ease of use:** The main advantage of containers that it allows us to build quickly and test portable applications. It allows anyone to package an application on their laptop, which in turn can run unmodified on any public cloud, private cloud, or even bare metal*. This feature helps developers, systems admins, architects and others just by building the app once and deploy it anywhere.*

**2. Speed:** Since the docker containers are very lightweight, fast and they take up fewer resources while running on the kernel, you can create and run a it in seconds, compared to VM.

**3. Docker Hub:** Docker users also benefit from the increasingly rich ecosystem of Docker Hub, which acts as app store for Docker images. Docker Hub has tens of thousands of public images created by the community that are readily available for use and we can easily pull them and use it directly without any modifications.

**4. Modularity and Scalability:** Docker makes it easy to break out your application’s functionality into individual containers. For example, you might have your Postgres database running in one container and your Redis server in another while your Node.js app is in another. With Docker, it’s become easier to link these containers together to create your application, making it easy to scale or update components independently in the future.

**Building Blocks of Docker:**

Images: Docker images are more like blueprints which can be immutable master template that is used to pump out containers that are all exactly alike. It contains the Dockerfile, libraries, and code your application needs to run, all bundled together.

Dockerfile: A Dockerfile is a file with instructions for how Docker should build your image.

Container: A container is the runtime instance of image. When we run the image container will be created and the instructions in the dockerfile will be executed.

Container Registry: If you want other people to be able to make containers from your image, you send the image to a container registry. Docker Hub is the largest registry and the default

**How to create custom image:**

**Steps:**

1. Create dockerfile with the set instructions which should run required actions.

Eg: If we want to run simple python program which uses packages like pandas, then instructions will be as follows:

**FROM** "ubuntu:bionic"

**RUN** apt-get update && yes | apt-get upgrade

**RUN** apt-get install -y git python-pip  
**RUN** pip install --upgrade pip

**RUN** pip install pandas

COPY scriptname.py /usr/local/share/ # Copies the file from host system

CMD ["scriptname.py", "-flag"] ## Executes the program on command prompt

1. Build docker file to create the image.

Example : Here python\_test is the name of image

*sudo docker build -t python\_test . It creates the python\_test image*

1. Run the containers by running image

*sudo docker run python\_test*

**How to run existing image or container to run sample python code:**

Let’s say we pulled tensorflow-gpu from docker hub and we want to run the .py file on command line. This can be done as follows :

Use the latest TensorFlow GPU image to start a bash shell session in the container. '/home/Kaggle/DockerLearn/Dockerfiles/' is the path where required scripts are present. These files will be temporarily copied in '/notebooks' folder of container ('tensorflow'):

Run this command to execute the .py file.

*sudo docker run --runtime=nvidia -it -v /home/Kaggle/DockerLearn/Dockerfiles/:/notebooks tensorflow/tensorflow:latest-gpu bash*

*Appendix:*

*Some useful commands to deal with docker*

*##Stop the container*

*sudo docker stop container\_id*

*##Pause the container*

*sudo docker pause container\_id*

*##Start the container*

*sudo docker start container\_id*

*##View the containers*

*docker ps*

*or*

*docker ps -a*

*##Remove the container*

*docker rm container\_id*

*##Kill the container*

*docker kill container\_id*

*##Check the number of images*

*docker images -a*

*##Remove the images*

*docker rmi img\_id*

*## To check the docker information*

*docker info*

*## To see usage of commands*

*docker help*

*##One liner to stop / remove all of Docker containers:*

*docker stop $(docker ps -a -q)*

*docker rm $(docker ps -a -q)*

*Note: We cannot remove the images if the container related to that image is on run. SO we have to stop the container and remove the image*

*If any image is removed then the its dependent images get also removed*

*nvidia-docker is only available for Linux,*

*DIERCTORY OF DOCKER - /var/lib/docker*