**Docker Technology**

* Docker technology is one implementation of container based virtualization technologies.
* Docker is a software which provides centralized platform to execute your application. It wraps software components into a complete standardized unit which contains everything require to run.
* Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.
* The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.

**Features of Docker**

* Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
* With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
* You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
* Since Docker containers are pretty lightweight, they are very easily scalable.

**Components of Docker**

Docker has the following components

* **Docker for Mac** − It allows one to run Docker containers on the Mac OS.
* **Docker for Linux** − It allows one to run Docker containers on the Linux OS.
* **Docker for Windows** − It allows one to run Docker containers on the Windows OS.
* **Docker Engine** − It is used for building Docker images and creating Docker containers.

**When Do You Need to Use a Docker?**

* To run your code locally on your laptop while replicating the environment on your server.
* During various development phases (dev/test/QA), Docker CI/CD was used.
* As a version control system and for distributing your app's OS with a team.

**How Do You Setup a Docker Locally**

* Download the Docker Toolbox and a Docker edition.
* Check to see if your BIOS supports Virtualization Technologies, AMD-V, or KVM.
* Install the Oracle VirtualBox Extension Pack.
* Run the Setup.

**How Do You Use a Docker?**

The most significant benefit of virtual machines is that they create snapshots that can be reverted to at any time.

Docker containers improve lightweight process virtualization by being OS agnostic and utilizing the Linux Kernel's capabilities.

They're made from Docker images, similar to snapshots. A Docker file is used to create Docker images, which can be customized or used as is 'libcontainer' is the default execution driver for docker containers.

Docker Hub can be used to look up docker images and see how they were created.

To make a Docker container, type the following command in the terminal to download the 'hello world' image –

$ docker run hello world

Use the following command to determine the number of images on your system –

$ docker images

Using the Docker Hub to find an image –

$ docker search <image>

**Here’s a List of Docker Commands**

* docker run – Starts a new container and executes a command.
* docker start – Starts one or more containers that have been stopped.
* docker stop – Puts an end to one or more currently running containers.
* Docker file- It is a command that creates an image.
* Docker pull - Pulls an image or a repository from a registry.
* Docker push. Pushes an image or a repository to a registry.
* docker export – Creates a tar archive of a container's filesystem.
* docker exec – Executes command in a container at runtime.
* Docker search – Looks for images on the Docker Hub.
* docker attach. Attaches to a running container
* docker commit – Creates a new image based on the changes made to a container.

**Examples of Using a Docker**

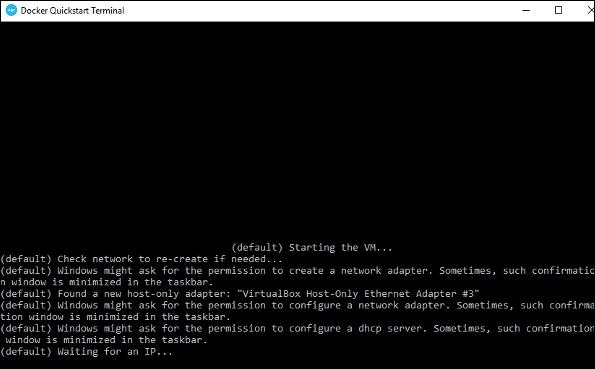
* By downloading Docker, you can run WordPress on your laptop without having to install Apache, PHP, MySQL, or other software. In order to run Docker in a virtual machine, the Docker Toolbox creates a containerized version of Linux.
* Install Oracle VirtualBox using Docker Tool Box.
* Open VirtualBox and install the Extension Pack.
* To verify that your installation was successful, type $ docker run hello-world in the terminal.
* To install WordPress locally, search for a WordPress image on the Docker Hub.
* Dockers can also be used to set up DokuWiki.
* Testing SDN components with Dockers is possible.

Here are a few examples to help you get started with your Docker engine.

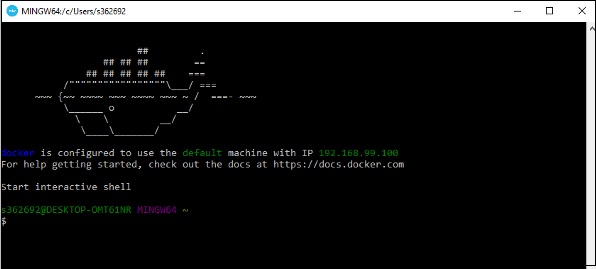
## Working with Docker Toolbox

Let’s now look at how Docker Toolbox can be used to work with Docker containers on Windows. The first step is to launch the Docker Toolbox application for which the shortcut is created on the desktop when the installation of Docker toolbox is carried out.

Next, you will see the configuration being carried out when Docker toolbox is launched.



Once done, you will see Docker configured and launched. You will get an interactive shell for Docker.



To test that Docker runs properly, we can use the Docker **run command** to download and run a simple **HelloWorld Docker container**.

The working of the Docker **run command** is given below −

docker run

This command is used to run a command in a Docker container.

### **Syntax**

docker run image

### **Options**

* **Image** − This is the name of the image which is used to run the container.

### **Return Value**

The output will run the command in the desired container.

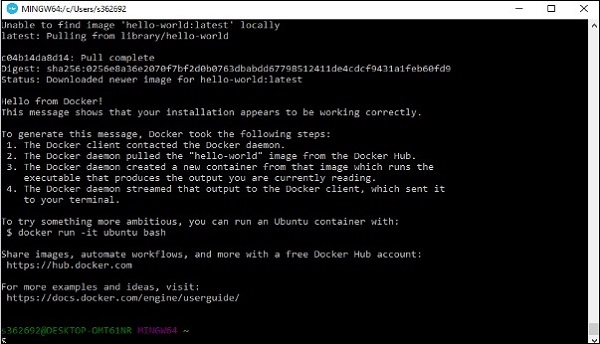
### **Example**

sudo docker run hello-world

This command will download the **hello-world** image, if it is not already present, and run the **hello-world** as a container.

### **Output**

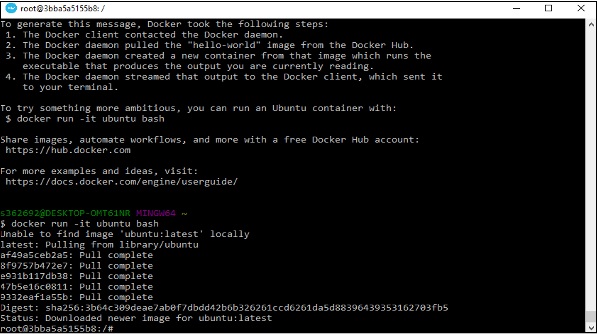
When we run the above command, we will get the following result −



If you want to run the Ubuntu OS on Windows, you can download the Ubuntu Image using the following command −

Docker run –it ubuntu bash

Here you are telling Docker to run the command in the interactive mode via the **–it** option.



In the output you can see that the Ubuntu image is downloaded and run and then you will be logged in as a root user in the Ubuntu container.

**Important Docker Concepts Images**

• Images are read only templates used to create containers.

• Images are created with the docker build command, either by us or by other docker users.

• Images are composed of layers of other images.

• Images are stored in a Docker registry.

**Containers**

• If an image is a class, then a container is an instance of a class - a runtime object.

• Containers are lightweight and portable encapsulations of an environment in which to run applications.

• Containers are created from images. Inside a container, it has all the binaries and dependencies needed to run the application.

**Registries and Repositories**

• A registry is where we store our images.

• You can host your own registry, or you can use Docker’s public registry which is called DockerHub.

• Inside a registry, images are stored in repositories.

• Docker repository is a collection of different docker images with the same name, that have different tags, each tag usually represents a different version of the image.

**Why Using Official Images**

• Clear Documentation

• Dedicated Team for Reviewing Image Content

• Security Update in a Timely Manner

**Docker commit**

• Docker commit command would save the changes we made to the Docker container’s file system to a new image.

docker commit container\_ID repository\_name:tag

**Dockerfile and Instructions**

• A Dockerfile is a text document that contains all the instructions users provide to assemble an image.

• Each instruction will create a new image layer to the image.

• Instructions specify what to do when building the image.

**Docker Build Context**

• Docker build command takes the path to the build context as an argument.

• When build starts, docker client would pack all the files in the build context into a tarball then transfer the tarball file to the daemon.

• By default, docker would search for the Dockerfile in the build context path.

**Dockerfile In Depth Steps**

1. Spin up a container from a base image.

2. Install Git package in the container.

3. Commit changes made in the container.

Chain RUN Instructions

• Each RUN command will execute the command on the top writable layer of the container, then commit the container as a new image.

• The new image is used for the next step in the Dockerfile. So each RUN instruction will create a new image layer.

• It is recommended to chain the RUN instructions in the Dockerfile to reduce the number of image layers it creates.

**Sort Multi-line Arguments Alphanumerically**

• This will help you avoid duplication of packages and make the list much easier to update.

**CMD Instructions**

• CMD instruction specifies what command you want to run when the container starts up.

• If we don't specify CMD instruction in the Dockerfile, Docker will use the default command defined in the base image.

• The CMD instruction doesn’t run when building the image, it only runs when the container starts up

. • You can specify the command in either exec form which is preferred or in shell form.

**Docker Cache**

• Each time Docker executes an instruction it builds a new image layer.

• The next time, if the instruction doesn't change, Docker will simply reuse the existing layer.

**Dockerfile with Aggressive Caching**

FROM ubuntu:14.04

RUN apt-get update

RUN apt-get install -y git

**Cache Busting**

FROM ubuntu:14.04

RUN apt-get update && apt-get install -y \

git \

curl

**Cache Busting**

• You can also achieve cache-busting by specifying a package version. This is known as version pinning.

RUN apt-get update && apt-get install -y \

package-bar \

package-baz \

package-foo=1.3.\*

**How container links work behind the scenes?**

**Benefits of Docker Container Links**

• The main use for docker container links is when we build an application with a microservice architecture, we are able to run many independent components in different containers.

• Docker creates a secure tunnel between the containers that doesn’t need to expose any ports externally on the container.

**Why Docker Compose?**

Manual linking containers and configuring services become impractical when the number of containers grows

. • Docker compose is a very handy tool to quickly get docker environment up and running.

• Docker compose uses yaml files to store the configuration of all the containers, which removes the burden to maintain our scripts for docker orchestration.

**Docker Compose Commands**

• docker compose up starts up all the containers.

• docker compose ps checks the status of the containers managed by docker compose.

• docker compose logs outputs colored and aggregated logs for the compose-managed containers. • docker compose logs with dash f option outputs appended log when the log grows.

• docker compose logs with the container name in the end outputs the logs of a specific container. • docker compose stop stops all the running containers without removing them.

• docker compose rm removes all the containers.

• docker compose build rebuilds all the images

**Introduction to Docker Networking**

**Docker Network Types**

• Closed Network / None Network

• Bridge Network

• Host Network

• Overlay Network

**None Network**

**None Network Isolated Isolated Isolated None Network**

• Provides the maximum level of network protection.

• Not a good choice if network or Internet connection is required.

• Suites well where the container require the maximum level of network security and network access is not necessary.

**Host Network**

• The least protected network model, it adds a container on the host's network stack

. • Containers deployed on the host stack have full access to the host's interface.

• This kind of containers are usually called open containers.

• Minimum network security level.

• No isolation on this type of open containers, thus leave the container widely unprotected.

• Containers running in the host network stack should see a higher level of performance than those traversing the docker0 bridge and iptables port mappings.

Define Container Networks with Docker Compose

• Unit tests should test some basic functionality of our docker app code, with no reliance on external services.

• Unit tests should run as quickly as possible so that developers can iterate much faster without being blocked by waiting for the tests results.

• Docker containers can spin up in seconds and can create a clean and isolated environment which is great tool to run unit tests with. Unit Tests in Containers

**Pros:** • A single image is used through development, testing and production, which greatly ensures the reliability of our tests.

**Cons:** • It increases the size of the image.

**Introduction to Docker Swarm and Set up Swarm cluster**

**How Swarm cluster works**

• To deploy your application to a swarm, you submit your service to a manager node.

• The manager node dispatches units of work called tasks to worker nodes.

• Manager nodes also perform the orchestration and cluster management functions required to maintain the desired state of the swarm.

• Worker nodes receive and execute tasks dispatched from manager nodes.

• An agent runs on each worker node and reports on the tasks assigned to it. The worker node notifies the manager node of the current state of its assigned tasks so that the manager can maintain the desired state of each worker

**Docker Services**

• The services can be defined in our Docker compose file.

• The service definition includes which Docker images to run, the port mapping and dependency between services.

**Docker Stack**

• A docker stack is a group of interrelated services that share dependencies, and can be orchestrated and scaled together

. • You can image that a stack is a live collection of all the services defined in your docker compose file.

• Create a stack from your docker compose file:

– docker stack deploy

• In the Swarm mode,

– Docker compose files can be used for service definitions.

– Docker compose commands can’t be reused. Docker compose commands can only schedule the containers to a single node.

– We have to use docker stack command. You can think of docker stack as the docker compose in the swarm mode.

**How to update our services in Production?**

**Provision a Swarm Cluster**

• Step 1: Deploy two VMs, one will be used for the Swam manager node, and the other one will be used as a worker node.

• Step 2: Appoint the first VM as Swarm manager node and initialize a Swarm cluster. – docker swarm init

• Step 3: Let the second VM join the Swarm cluster as a worker node.

– docker swarm join

**Docker Swarm commands**

• docker swarm init

– Initialize a swarm. The docker engine targeted by this command becomes a manager in the newly created single-node swarm.

• docker swarm join

– Join a swarm as a Swarm node.

• docker swarm leave

– Leave the swarm