**Deploying Artificial Intelligence**

**In Enterprise**

**Powered by Docker**

**Proposed Table of Contents**

**Note: Kindly re-arrange the content as shown below. Add the missing content under your name.**

1. Executive Summary (Jothi)
2. Deploying Artificial Intelligence in Enterprise (Jothi)?
3. What is Docker? (DP)
   1. Docker Engine - The Runtime
   2. Docker Hub - The Registry
4. Who is Docker for? (DP)
5. Role of Docker in AI Deployment (DP)
6. Why Docker for AI Deployment (Jothi)
7. Installing Docker on Windows (DP)
8. System Requirements (DP)
9. Installing Docker on Windows that meets all the Requirements (DP)
10. Start Docker for Windows (DP)
11. Installing Docker on Windows that does not meet the Requirements (DP)
12. Verifying Your Docker Installation (DP)
    1. Docker Basic Terminologies
    2. Docker Image
    3. Docker Container
    4. Docker Engine - The Runtime
    5. Docker Hub - The Registry
13. How to Create Docker Image for AI Application- Step by Step (DP)
    1. Steps (Visual - Jothi & DP)
    2. Implementation
14. How to Run AI Application from Docker Container - Step by Step (DP)
    1. Steps (Visual - Jothi & DP)
    2. Implementation
15. Appendix
    1. AI Application Architecture (Jothi)
    2. AI System Landscape (Jothi)
    3. Docker Architecture (Jothi)

Table of Contents

[Installing Docker on Windows 3](#_Toc531714693)

[ System Requirements 3](#_Toc531714694)

[Installing Docker on Windows that meets all the Requirements 4](#_Toc531714695)

[Start Docker for Windows 5](#_Toc531714696)

[Installing Docker on Windows that does not meet the Requirements 8](#_Toc531714697)

[Verifying Your Installation 12](#_Toc531714698)

[What is Docker 15](#_Toc531714699)

[Docker Engine - The Runtime 15](#_Toc531714700)

[Docker Hub - The Registry 15](#_Toc531714701)

[Who is Docker for ? 16](#_Toc531714702)

[Role of Docker in AI Deployment 17](#_Toc531714703)

[Docker Basic Terminologies 18](#_Toc531714704)

[ Docker Image 18](#_Toc531714705)

[ Docker Container 19](#_Toc531714706)

[ Docker File 19](#_Toc531714707)

[Docker Basic Terminologies 20](#_Toc531714708)

[ Docker Compose 21](#_Toc531714709)

[ Docker Volume 22](#_Toc531714710)

[Docker Basic Commands 24](#_Toc531714711)

# Executive Summary

# << Jothi to Write>>

# Docker Architecture

# Installing Docker on Windows

Docker is a full development platform for creating containerized apps, and Docker for Windows is the best way to get started with Docker on Windows. Docker for Windows requires Microsoft Hyper-V to run. The Docker for Windows installer enables Hyper-V for you, if needed, and restarts your machine. After Hyper-V is enabled, VirtualBox no longer works, but any VirtualBox VM images remain. VirtualBox VMs created with docker-machine (including the default one typically created during Toolbox install) no longer start. These VMs cannot be used side-by-side with Docker for Windows. However, you can still use docker-machine to manage remote VMs.

## System Requirements

* Windows 10 64bit: Pro, Enterprise or Education (1607 Anniversary Update, Build 14393 or later).
* Virtualization is enabled in BIOS. Typically, virtualization is enabled by default. This is different from having Hyper-V enabled. For more detail see [Virtualization must be enabled](https://docs.docker.com/docker-for-windows/troubleshoot/#virtualization-must-be-enabled) in Troubleshooting.
* CPU SLAT-capable feature.
* At least 4GB of RAM.

# Installing Docker on Windows that meets all the Requirements

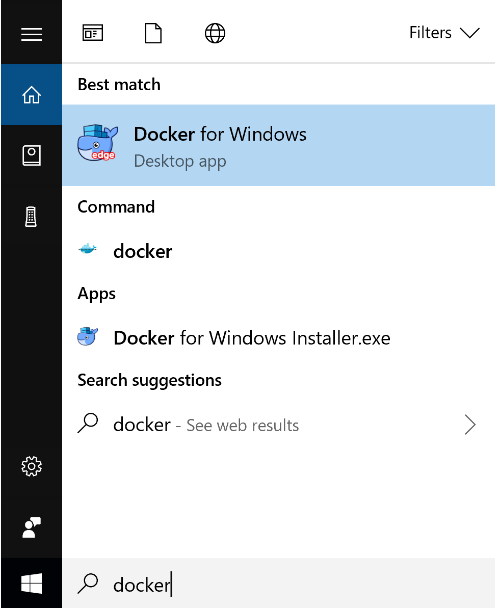
Visit the below link

<https://docs.docker.com/docker-for-windows/install/#about-windows-containers>



* Follow the install wizard to accept the license, authorize the installer, and proceed with the install. You are asked to authorize Docker.app with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.
* Click **Finish** on the setup complete dialog to launch Docker.

# Start Docker for Windows

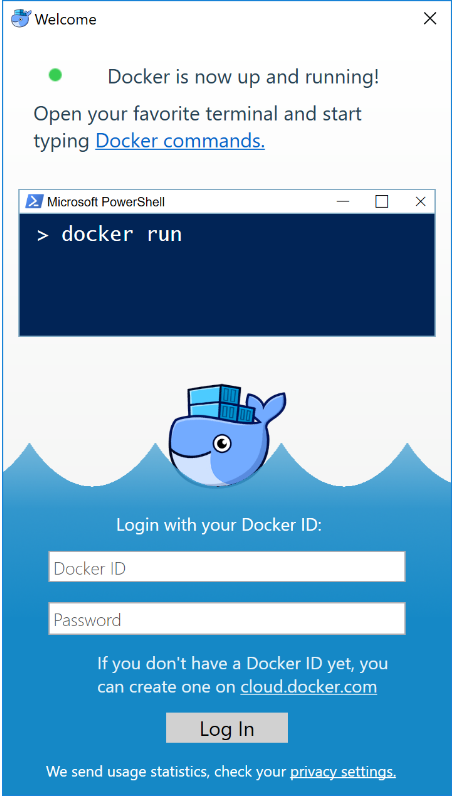


Docker does not start automatically after installation. To start it, search for Docker, select **Docker for Windows** in the search results, and click it (or hit Enter).

When the whale in the status bar stays steady, Docker is up-and-running, and accessible from any terminal window.



If the whale is hidden in the Notifications area, click the up arrow on the taskbar to show it. To learn more, see [Docker Settings](https://docs.docker.com/docker-for-windows/install/#docker-settings-dialog).



# Installing Docker on Windows that does not meet the Requirements

Visit the link : <https://docs.docker.com/toolbox/toolbox_install_windows/>. Click on Get Docker Toolbox for Windows



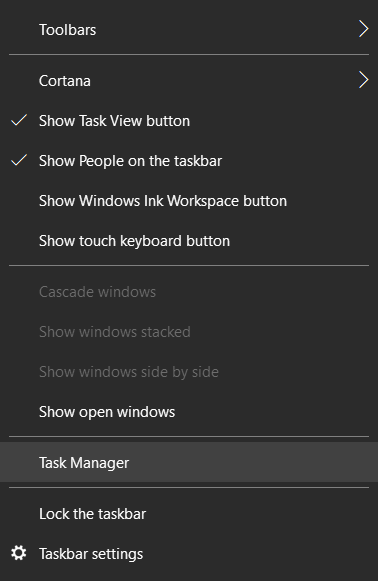
Docker Toolbox includes the following Docker tools:

* Docker CLI client for running Docker Engine to create images and containers
* Docker Machine so you can run Docker Engine commands from Windows terminals
* Docker Compose for running the docker-compose command
* Kitematic, the Docker GUI
* the Docker QuickStart shell preconfigured for a Docker command-line environment
* Oracle VM VirtualBox

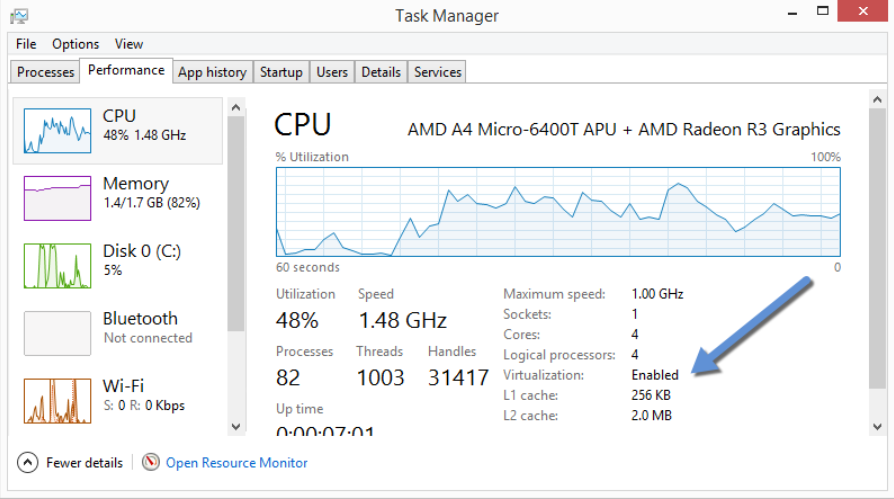
One of the advantages of the newer Docker for Windows solution is that it uses native virtualization and does not require VirtualBox to run Docker

To run Docker, your machine must have a 64-bit operating system running Windows 7 or higher. Additionally, you must make sure that virtualization is enabled on your machine. To verify your machine meets these requirements, do the following:

Right Click on the Taskbar >>> Open Task Manager

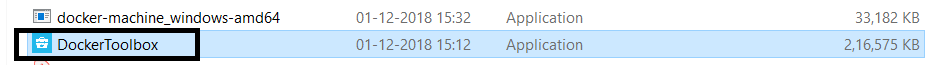
****

Check if virtualization enable or not. If virtualization is not enabled on your system, follow the manufacturer’s instructions for enabling it.

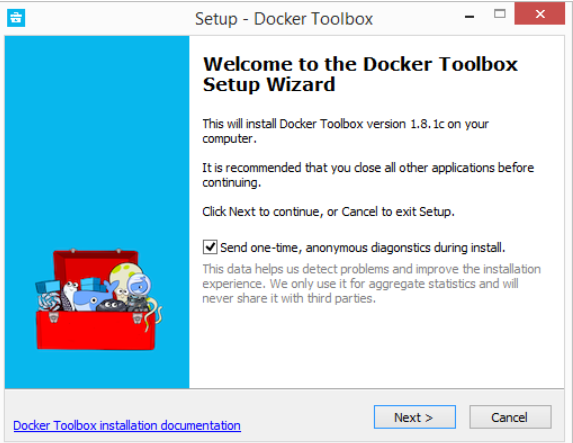


Now Install Docker Toolbox as shown below:

1. Go to the Downloads folder where Docker Toolbox is downloaded



1. Install Docker Toolbox by double-clicking the installer. The installer launches the “Setup - Docker Toolbox” dialog. If Windows security dialog prompts you to allow the program to make a change, choose **Yes**. The system displays the **Setup - Docker Toolbox for Windows** wizard.

****

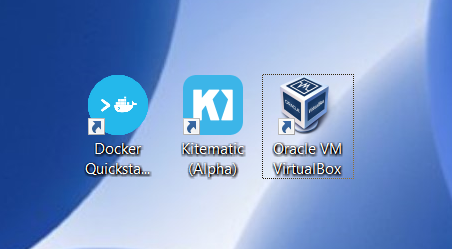
1. Press **Next** to accept all the defaults and then **Install**. Accept all the installer defaults. The installer takes a few minutes to install all the components:
2. When notified by Windows Security the installer will make changes, make sure you allow the installer to make the necessary changes. When it completes, the installer reports it was successful.

****

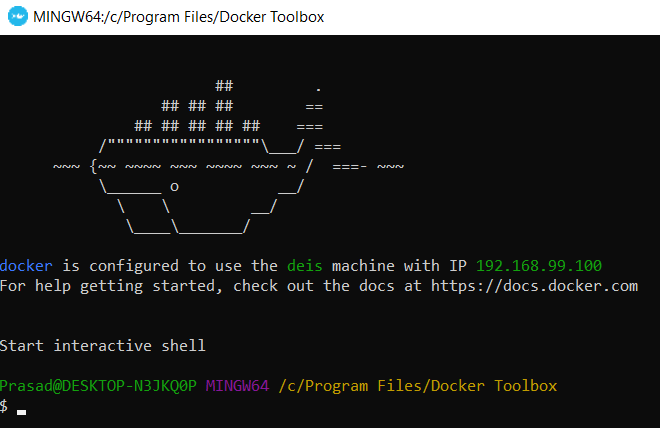
# Verifying Your Installation

The installer adds Docker Toolbox, VirtualBox, and Kitematic to your **Applications** folder. In this step, you start Docker Toolbox and run a simple Docker command.

1. On your Desktop, find the Docker QuickStart Terminal icon

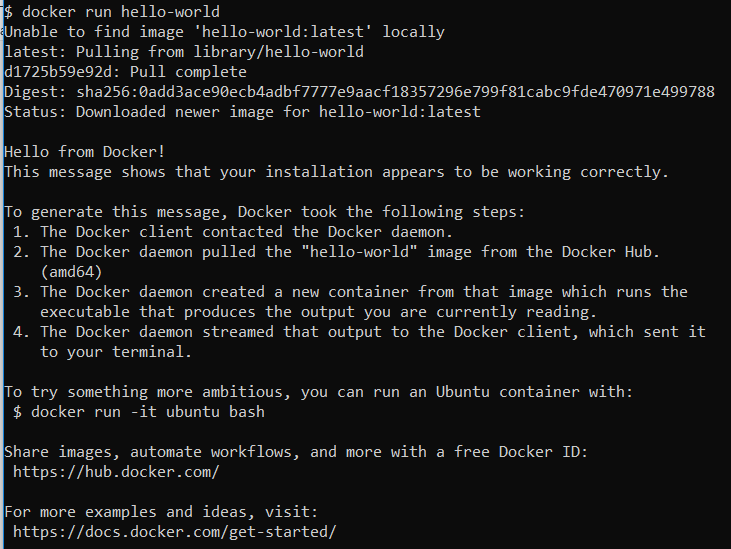
****

1. Click the Docker QuickStart icon to launch a pre-configured Docker Toolbox terminal. If the system displays a User Account Control prompt to allow VirtualBox to make changes to your computer. Choose Yes. The terminal does several things to set up Docker Toolbox for you. When it is done, the terminal displays the $ prompt.



The terminal runs a special bash environment instead of the standard Windows command prompt. The bash environment is required by Docker.

1. Make the terminal active by clicking your mouse next to the $ prompt. The prompt is traditionally a $ dollar sign. You type commands into the command line which is the area after the prompt. Your cursor is indicated by a highlighted area or a | that appears in the command line. After typing a command, always press ENTER.
2. Type the docker run hello-world command and press ENTER.

****

# What is Docker

Docker is a containerization platform that packages your application and all its dependencies together in the form of a docker container to ensure that your application works seamlessly in any environment. In simple words, Docker is a tool designed to make it easier to create, deploy & run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries & other dependencies, & ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other linux machine regardless of any customized settings that machine might have that could differ from the machine for writing & testing the code.

In a Way, Docker is bit like a Virtual machine. But unlike a Virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same linux Kernel as the system that they are running on & only requires applications to be shipped with things not already running on the host computer. This gives a significant performance boost & reduces the size of the application.

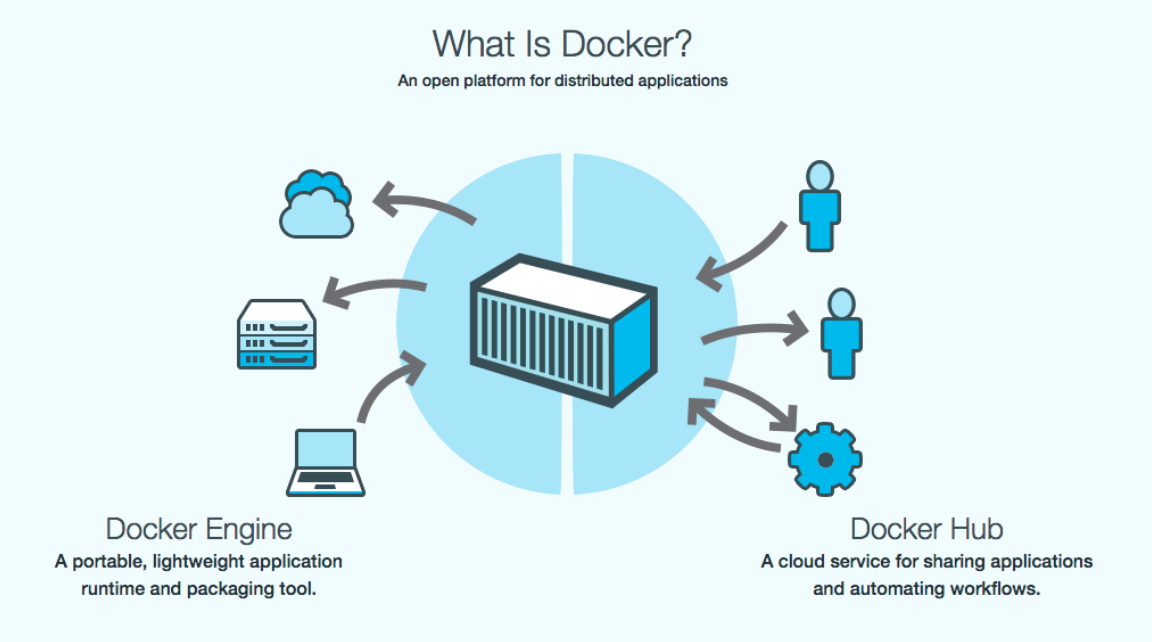
These containers uses Containerization which can be considered as an evolved version of Virtualization. The same task can also be achieved using Virtual Machines, however it is not very efficient. There are two key parts to Docker: Docker Engine, and Docker Hub. It’s equally important to secure both parts of the system. And to do that, it takes an understanding of what they each consist of, which components need to be secured, and how. Let’s start with Docker Engine

## ****Docker Engine - The Runtime****

Docker Engine hosts and runs containers from the container image file. It also manages networks, and storage volumes. There are two key aspects to securing Docker Engine: namespaces, and cgroups

## ****Docker Hub - The Registry****

While Docker Engine manages containers, it needs the other half of the Docker stack to pull container images from. That part is Docker Hub—the container registry where container images are stored, and shared.



# Who is Docker for ?

Docker is a tool that is designed to benefit both developers & system administrators, making it a part of many DevOps (Developers + Operations) toolchains. For Developers, it means that they can focus on writing code without worrying about the system that will ultimately running on. It also allows them to get a head start by using one of the thousands of programs already designed to run in a Docker Container as a part of their application. For Operations staff, Docker gives flexibility & potentially reduces the number of systems needed because of its small footprint & lower overhead.

# Role of Docker in AI Deployment

**To be Added by JOTHI**

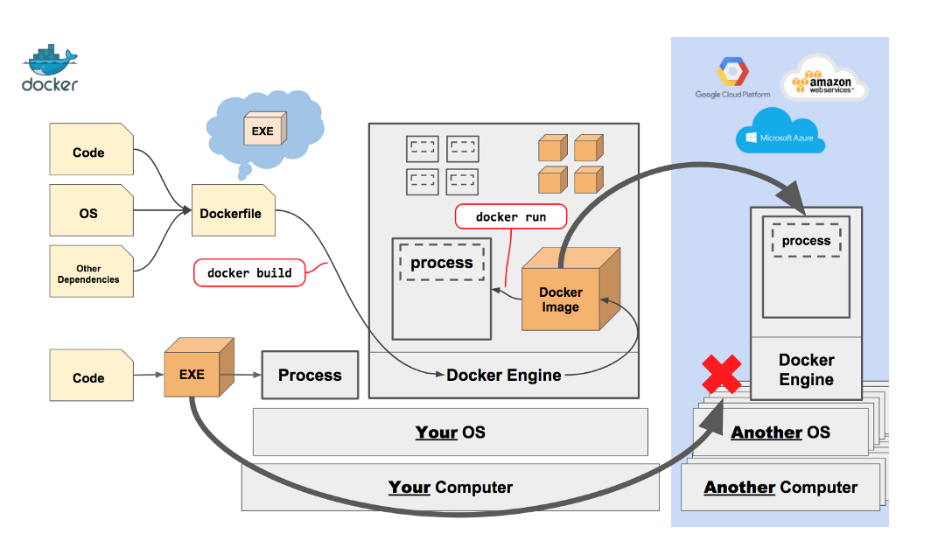
# Docker Basic Terminologies

**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

## Docker Image

Docker Image can be compared to a template which is used to create Docker Containers. They are the building blocks of a Docker Container. These Docker Images are created using the build command. These Read only templates that are used for creating containers by using the run command.

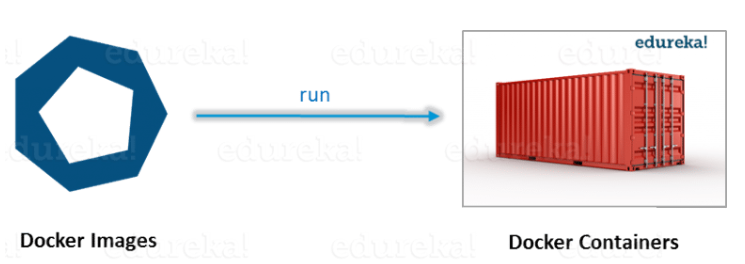
In other Words A Docker image is a file, comprised of multiple layers, used to execute code in a Docker container. An image is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel. When the Docker user runs an image, it becomes one or multiple instances of that container.



**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

## Docker Container

Containers are the ready applications created from Docker Images or you can say a Docker Container is a running instance of a Docker Image and they hold the entire package needed to run the application. This happens to be the ultimate utility of Docker.



**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

## Docker File

When you run the docker run command, Docker uses the Dockerfile to build the image itself. The Dockerfile is essentially the build instructions to build the image. The advantage of a Dockerfile over just storing the binary image (or a snapshot/template in other virtualization systems) is that the automatic builds will ensure you have the latest version available. This is a good things from a security perspective, as you want to ensure you are not installing any vulnerable software’s.



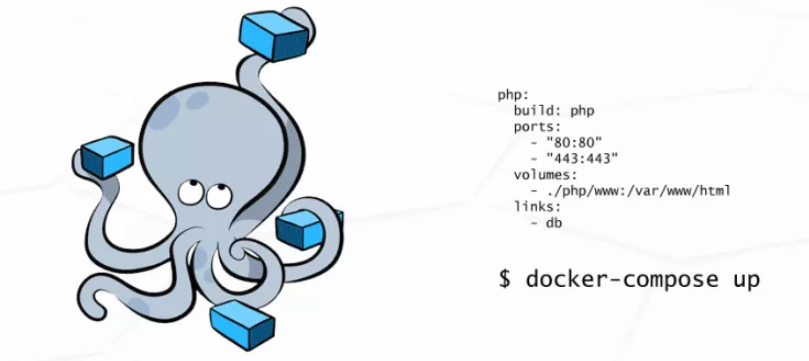
## Docker Basic Terminologies

* **FROM --** The Image the new image will be based on.
* **MAINTAINER --** Name & email of the maintainer of this image.
* **COPY --** Copy a file or a directory into the image.
* **ADD --** Same as COPY, but handles URLS & unpack tarballs automatically.
* **RUN --** Run a command inside the container, such as apt-get install.
* **ONBUILD --** Run commands when building an inherited Dockerfile.
* **CMD --** Default command to run when running the container. Can be overridden with command line parameters.
* **ENV --** Set environment variable in the container.
* **EXPOSE --** Expose ports from the container. Must be explicitly exposed by the  
  run command to the Host with -p or -P.
* **VOLUME --** Specify that a directory should be stored outside the union file  
  system. If is not set with docker run -v it will be created in /var/lib/docker/volumes.
* **ENTRYPOINT --** Specify a command that is not overridden by giving a new  
  command with docker run image cmd. It is mostly used to give a default  
  executable and use commands as parameters to it.
* **USER --**  Set the user for RUN, CMD and ENTRYPOINT.
* **WORDIR --** Sets the working directory for RUN, CMD, ENTRYPOINT, ADD and COPY.

**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

## Docker Compose

Docker Compose is a extremely easy to use and a useful tool for running a multi-container Docker application. Docker Compose, however, is limited to single node environments which makes it more appropriate for development and CI environments. In my next article in this series I will talk about how we take things one step further into production.

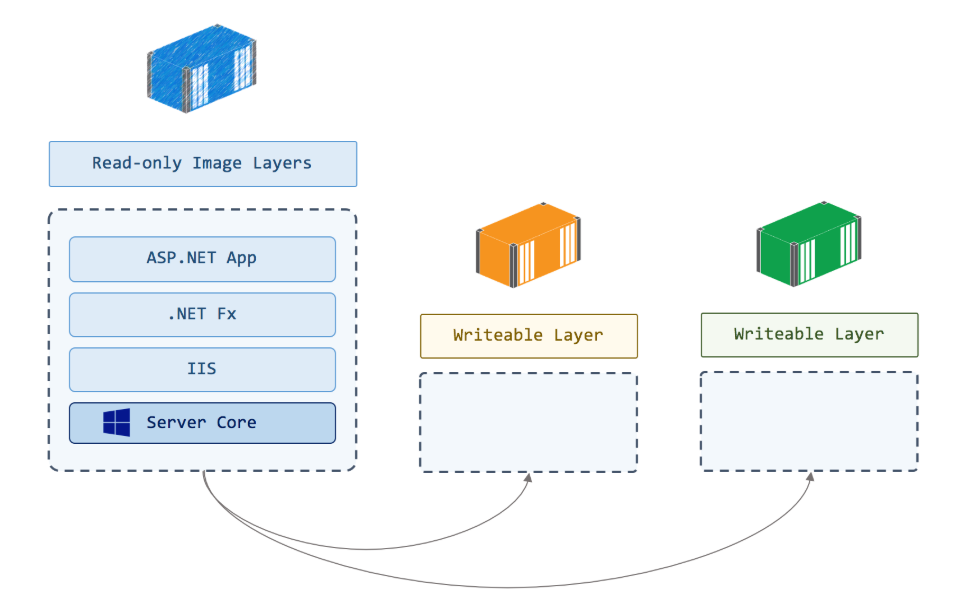


Your applications can be defined in a YAML file where all the options that you used in `docker run` are now defined. Compose also allows you to manage your application as a single entity rather than dealing with individual containers.

**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

## Docker Volume

Docker volumes to store state outside of containers, so your data survives when you replace the container to update your app. Docker uses symbolic links to give the volume a friendly path inside the container, like C:\data. Some application runtimes try to follow the friendly path to the real location - which is actually outside the container - and get themselves into trouble.



* Volumes are easier to back up
* You can easily manage values using Docker CLI commands
* Volumes can be more safely shared among multiple containers
* Volume’s contents exist outside the lifecycle of a given container

**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

# Docker Basic Commands