Docker User Manual

SAM MEKALA (smekala@threewiresys.com)

[Docker Overview: 3](#_Toc477328134)

[Revolution of Docker: 3](#_Toc477328135)

[Docker Architecture: 4](#_Toc477328136)

[Docker Engine: 4](#_Toc477328137)

# Docker Overview

Docker is a lightweight virtualization solution to run multiple virtual units (containers) simultaneously on a single control host. Containers are isolated with Kernel’s Control Groups and Namespace of Host OS. Docker only provides operating system-level virtualization where the kernel of HOST OS controls isolated containers.

## Revolution of Docker

There are three different setups that providing a stack to run an application

1) Traditional Servers

2) Virtual machines

3) Containers

### Traditional Servers

Traditional server stack consists of a physical server that runs an operating system and your application.

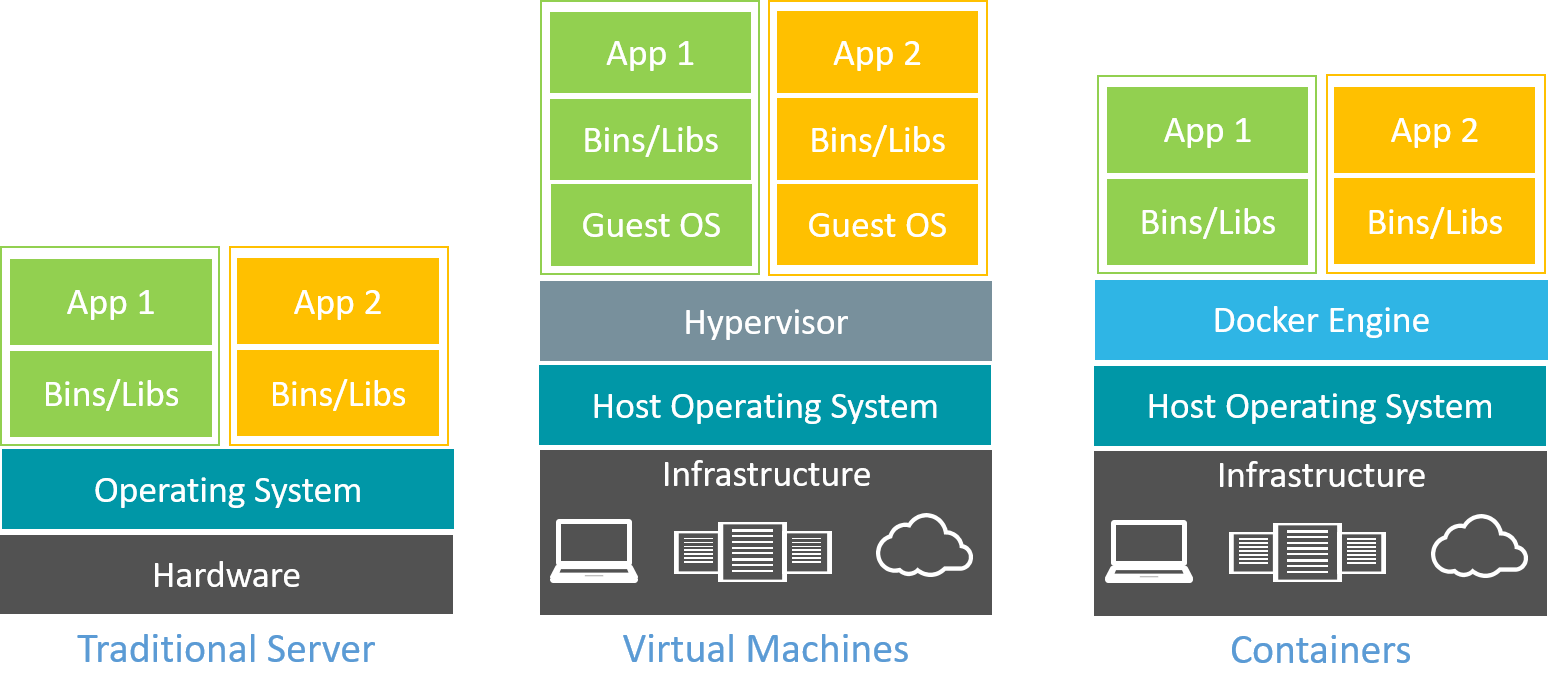
### Virtual machines

The VM stack consist of a physical server which runs an operating system and a hypervisor that manages your virtual machine, shared resources, and networking interface. Each Vm runs a Guest Operating System, an application or set of applications.

### Containers

The key difference with other stack is container-based virtualization uses the kernel of the Host OS to rum multiple isolated guest instances. These guest instances are called as containers. The Host can be either a physical server or VM.

Unlike a virtual machine, a container does not need to boot the operating system kernel, so containers can be created in less than a second. This feature makes container-based virtualization unique and desirable than other virtualization approaches.



# Docker Architecture

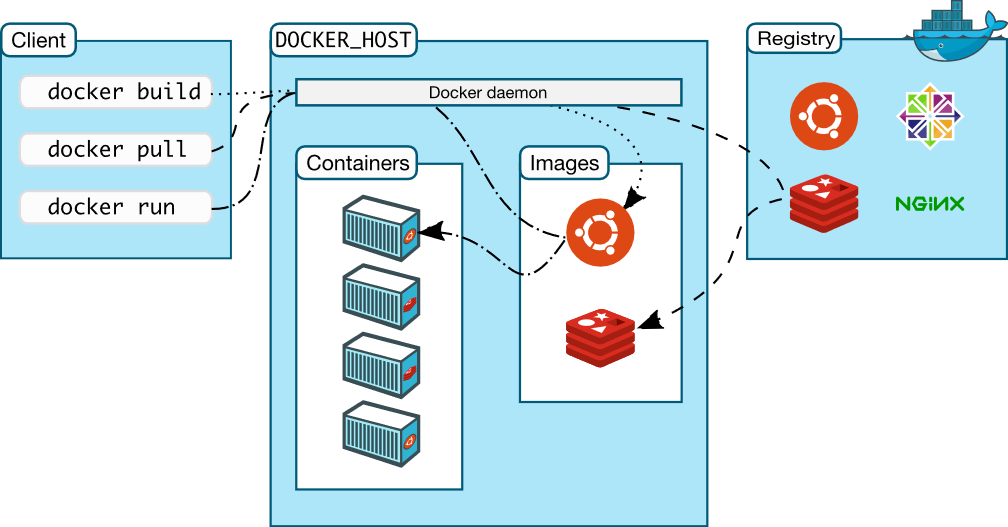
Docker uses a client/server architecture. You can use the CLI client to communicate with the

daemon. The daemon then performs operations with containers and manages images locally or in

registry. The CLI client can run on the same server as the host daemon or on a different machine.

The CLI client communicates with the daemon by using network sockets. The architecture is

depicted below,



## Docker Engine

Docker Engine is a client-server application with these major components:

* A server which is a type of long-running program called a **daemon process**.
* A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
* A command line interface (CLI) client.



## Docker Components

### Docker Daemon

The daemon creates and manages Docker objects, such as images, containers, networks, and data volumes. It is the server side of the Docker engine that manages all Docker objects. It runs on a host machine. The user uses the Docker client to interact with the daemon.

### Docker Client

It is the primary user interface to Docker daemon. It is in the form of Command Line Interface (CLI). It accepts commands and configuration flags from the user and communicates with Docker daemon. The client can reside on the same host as the server or on another host.

### Docker Images

Docker Images are read-only templates that are used to create containers. Each image consists of a series of layers. Docker uses union file systems to combine these layers into a single image. These layers are one of the reasons Docker is so lightweight. When Docker image is changed, such as when we update an application to a new version, a new layer is built and replaces only the layer it updates. The other layers remain same. To distribute the update, the updated layer only needs to be transferred. Layering speeds up distribution of Docker images. Docker determines which layers need to be updated at runtime. Docker image layers can contain multiple components of an application stack. For example, a Docker image might contain operating-system libraries not available on the host in one layer, an application server in another layer, and application binaries in a third layer.

### Docker Containers

A Docker container is a runnable instance of a Docker image. You can run, start, stop, move, or delete a container using Docker API or CLI commands. Each container is an isolated and secure application platform, but can be given access to resources running in a different host or container, as well as persistent storage or databases. Containers use Docker images as a foundation, and they contain all the libraries needed for an application to run, in addition to a file system.

### Docker Registries

Docker registries are repositories for Docker images. These registries are used to store and distribute Docker images. A registry can be public or private, and can be on the same server as the Docker daemon or Docker client, or on a totally separate server.

# Docker Installation

Docker is available on many different operating systems, including most modern Linux distributions, Mac OSX and Windows. Below is the Different platform specification:

### Linux:

* Any distribution running version 3.10+ of the Linux kernel
* Specific instructions are available for most Linux distributions, including RHEL, Ubuntu, SuSE, and many others.

### Microsoft Windows:

* Windows Server 2016
* Windows 10

### Cloud:

* Amazon EC2
* Google Compute Engine
* Microsoft Azure
* Rackspace

## Install Docker in AWS Linux AMI (Amazon EC2)

To install docker open the terminal and run the following commands. Before Installing Docker, update the installed packages and package cache on EC2 instance.

[ec2-user ~]$ sudo yum update -y

Install the Docker

[ec2-user ~]$ sudo yum install -y docker

Start the Docker Service.

[ec2-user ~]$ sudo service docker start

Add the ec2-user to the Docker group so you can execute Docker commands without using sudo.

[ec2-user ~]$ sudo usermod -a -G docker ec2-user

Log out and log back in again to pick up the new Docker group permissions. Sometimes we may need to reboot the instance to provide permissions for ec2-user to access the Docker.

After logging back again, the user can run docker commands without sudo. We can verify weather docker installed properly, by running “docker info” command.

[ec2-user ~]$ docker info

It should show some information as below,

Containers: 26

Running: 0

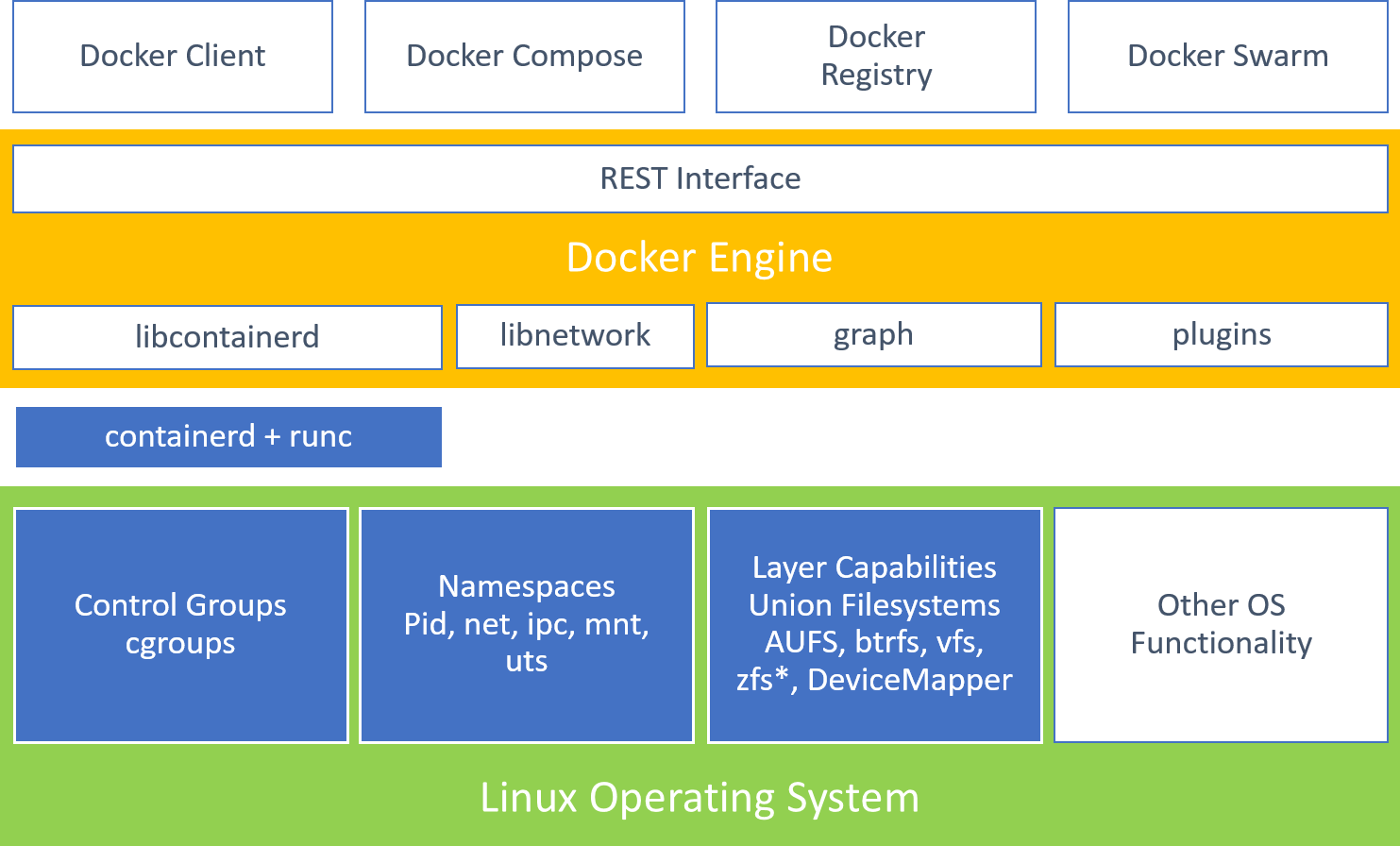
Paused: 0

Stopped: 26

Images: 180

Server Version: 1.12.6

Storage Driver: devicemapper



## Install Docker on Windows Server 2016

To install docker open the powershell and run the following commands.

install the Docker-Microsoft PackageManagement Provider from the PowerShell.

C:\Users\Administrator>**Install-Module** -**Name** DockerMsftProvider -Repository PSGallery -**Force**

Next, you use the PackageManagement PowerShell module to install the latest version of Docker. Install Docker using the module we installed

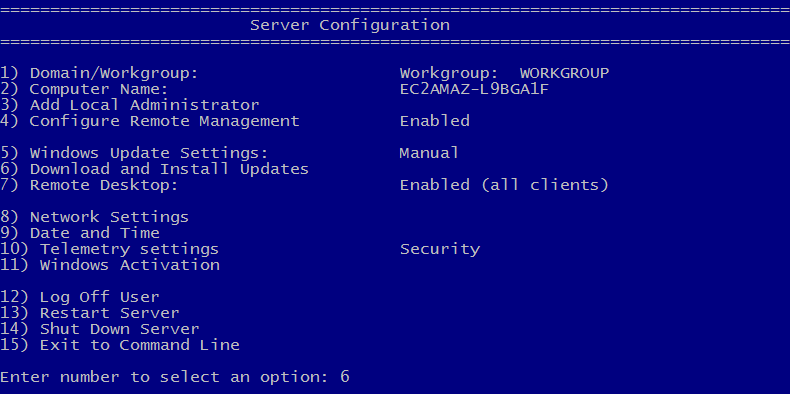
C:\Users\Administrator>**Install-Package** -**Name** docker -**ProviderName** DockerMsftProvider

If Powershell asks to trust the package source “DockerDefault”, type “A” to continue the installation.

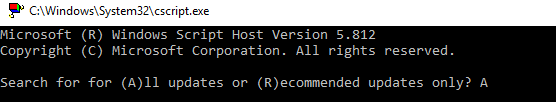
After installation done, update the OS.

C:\Users\Administrator>sconfig

This will show text-based configuration menu, where we need to choose option 6 to download and install updates.



It will prompt new screen. Enter option “A” to install all updates.



After updates are installed, restart the server.

We can verify weather docker installed properly, by running “docker info” command.

C:\Users\Administrator>docker info

It should show some information as below,

Containers: 46

Running: 0

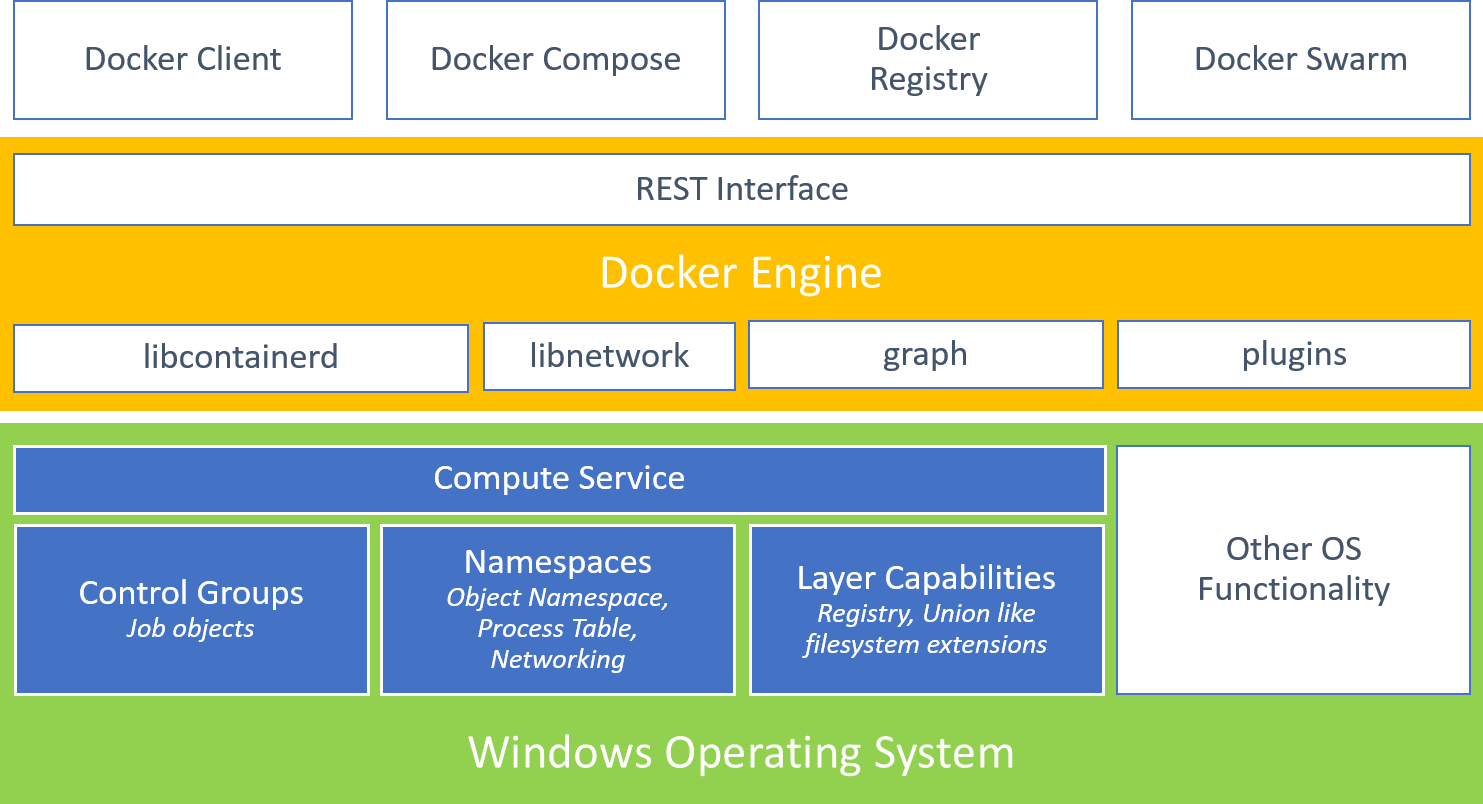
Paused: 0

Stopped: 46

Images: 152

Server Version: 1.12.2-cs2-ws-beta

Storage Driver: windowsfilter



Docker Command Line

Docker commands starts with keyword “docker”. Below, is the Docker commands syntax

docker [COMMAND] [OPTIONS]

To list available commands and options, either run “docker” or “docker help” in command prompt.

Running a Container

Creating Custom Docker Images:

Docker Images can be created in two ways:

1. Commit changes made to a container
2. From a Dockerfile

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using Docker build users can create an automated build that executes several command-line instructions in succession.