Containerization – Docker

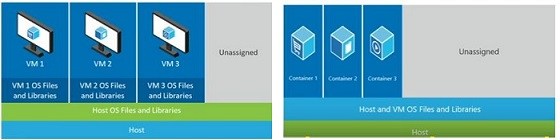
## **What is Containerization?**

Containerization is a lightweight alternative to full machine [virtualization](http://www.webopedia.com/TERM/V/virtualization.html) that involves encapsulating an application in a container with its own operating environment.

Containers are essentially operating system sandboxes that are hosted on full operating systems. In the case of Windows containers, these are all hosted on Windows Server. Each container contains the minimal amount of resources, and any operating system-specific resources are gathered from the operating system “beneath” them. Windows containers have their own networking, file system and registry. When it needs the ability to interact with these systems, it will use the hosting operating system.

**Containers Vs Virtual Machines**

Containers and virtual machines (VMs) are complementary. VMs excel at providing extreme isolation (for example with hostile tenant applications where you need the ultimate break out prevention). Containers operate at the process level, which makes them very lightweight and perfect as a unit of software delivery. While VMs take minutes to boot, containers can often be started in less than a second.

Containers vs VMs

## **Advantages of containerization**

Containerization gained prominence with the open source [Docker](http://searchenterpriselinux.techtarget.com/definition/Docker), which developed a method to give containers better portability allowing them to be moved among any system that shares the host OS type without requiring code changes. With Docker containers, there are no guest OS environment variables or library dependencies to manage.

Proponents of containerization point to gains in efficiency for memory, CPU and [storage](http://searchstorage.techtarget.com/definition/storage) as key benefits of this approach, compared with traditional virtualization. Because containers do not have the overhead required by VMs separate OS instances it is possible to support many more containers on the same infrastructure. As such, containerization improves performance because there is just one OS taking care of hardware calls.

A major factor in the interest in containers is they can be created much faster than hypervisor-based instances. This makes for a much more agile environment and facilitates new approaches, such as microservices and continuous integration and delivery.

## **Why to use Containers?**

Containers have some characteristics that make them a favorable choice for hosting workloads on cloud, which are described in the sections that follow.

Agility  
Because starting a new container instance is essentially the same as starting a new process, you can start containers quickly usually in time frames less than a second.

Compute Density  
Because container instances are just processes, you can run a large number of container instances on a single physical server or VM. A higher compute density means that you can provide cheaper and more agile compute services to your customers.

Decouple compute and resource  
Another major benefit of using containers is that the workloads running in them are not bound to specific physical servers or VMs. Traditionally, after a workload is deployed, it’s pretty much tied to the server where it’s deployed. If the workload is to be moved to another server, the new one need to be repurposed for the new workload, which usually means the entire server, needs to be rebuilt to play its new role in the datacenter. With containers, servers are no longer assigned with specific roles. Instead, they form a cluster of CPUs, memory, and disks within which workloads can roam almost freely.

## **How can I use containers?**

There are a different ways to get started with Windows Server containers. You may choose a cloud service such as [Azure Container Service](https://docs.microsoft.com/en-us/azure/container-service/container-service-intro). This prevents from having to either bring up physical hardware or even provision a new virtual machine to host a container on. Or, [by using PowerShell](https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/deploy-containers-on-server), any machine running Windows Server 2016 can be configured to host containers.

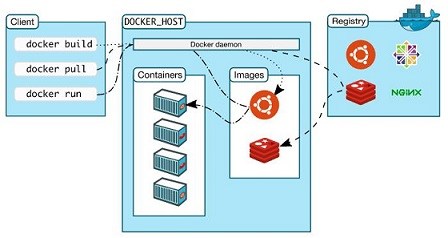
There are many tools available in market which will help you build containers.

|  |  |
| --- | --- |
| 1.    Docker | 2.    Packer |
| 3.    Kubernetes | 4.    Mesos |
| 5.    Rocket (rkt) | 6.    CloudSlang |
| 7.    Marathon | 8.    Nomad |
| 9.    Swarm | 10.  Fleet |
| 11.  OpenVZ | 12.  Rancher |
| 13.  Containership | 14.  Solaris Containers |
| 15.  Tectonic |  |

## **Docker overview**

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

## **Docker Architecture**

Docket Architecture

* Docker Client: Is a command line interface

* The Docker daemon(dockerd): It listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

* Docker Host: Is a host VM/Machine with O/S Linux or Windows running docker daemon.

* Docker Image: Is the definition with single file.

* Docket Container: Is an instance of an image, there can be many instances of image.

* Docker Hub: Is a collection of reusable images. 50K+ images of WordPress, Redis, MySql and your images. We can use public hub or can create private hub

* Docker Resigtries: A Docker registry stores Docker images. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

* Services: allow you to scale containers across multiple Docker daemons, which all work together as a swarm with multiple managers and workers. Each member of a swarm is a Docker daemon, and the daemons all communicate using the Docker API.

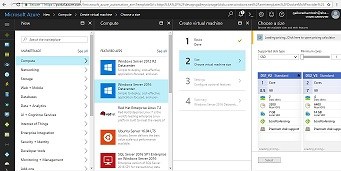
## **Create docker container from  image in docker Hub**

For detailed information about Containerization and Docker please read [**Containerization – Docker – Part1**](http://www.sharecareinspire.com/containerization-docker/)

This article focuses on containerization using Windows Azure and Docker.

Note: Installation of Docker requires computer system (physical or virtual) running Windows Server 2016.

Prerequisite: Create Windows Server 2016 VM in Azure

Create Windows Server 2016 VM

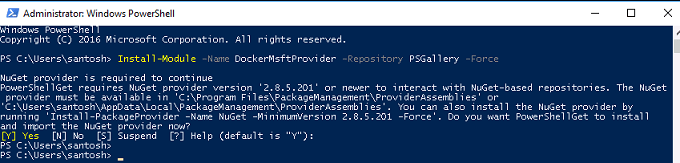
## **Steps by step installation of docker in VM**

**1. Install Docker**

To install Docker we need [MicrosoftDockerProvider](https://github.com/OneGet/MicrosoftDockerProvider). The provider enables the containers feature on your machine. You also install Docker which requires a reboot. Docker is required in order to work with Windows containers. It consists of the Docker Engine and the Docker client.

* Open an elevated PowerShell session and run the following commands and run following PowerShell command

Install-Module -Name DockerMsftProvider -Repository PSGallery –Force

Docker Microsoft Provider

* Use the PackageManagement PowerShell module to install the latest version of Docker. PowerShell asks you whether to trust the package source ‘DockerDefault’, typeA to continue the installation. When the installation is complete, reboot the computer.

Install Docker

* Run following command to restart computer

Restart-Computer –Force

* Run following commands to update docker to most recent version

#Check the installed version with

Get-Package -Name Docker -ProviderName DockerMsftProvider

#Find the current version with

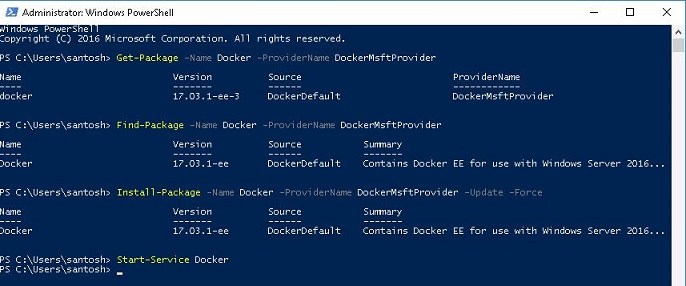
Find-Package -Name Docker -ProviderName DockerMsftProvider

#Upgrade docker with

Install-Package -Name Docker -ProviderName DockerMsftProvider -Update -Force

#Restart Docker service with

Start-Service Docker

Update Docker

**2. Install Windows Updates**

Ensure your Windows Server system is up-to-date by running with

Sconfig

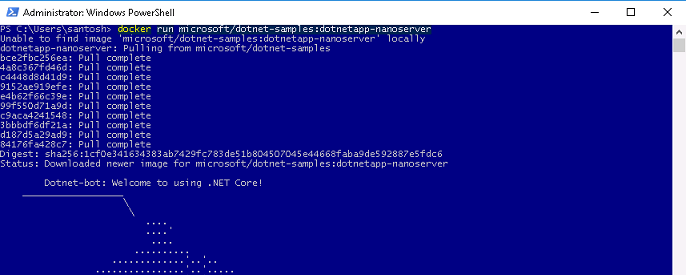
Choose option 6 to Download and Install Updates. When prompted, choose option A to download all updates.

Install Windows Updates

**3. Install Your First Container**

Let’s download existing .NET sample image from the Docker Hub registry and deploy a simple container running a .Net Hello World application.

docker run microsoft/dotnet-samples:dotnetapp-nanoserver

Install .NET docker image