**Install Docker Toolbox on Windows**

**Legacy desktop solution.** Docker Toolbox is for older Mac and Windows systems that do not meet the requirements of [Docker for Mac](https://docs.docker.com/docker-for-mac/) and [Docker for Windows](https://docs.docker.com/docker-for-windows/). We recommend updating to the newer applications, if possible.

Estimated reading time: 8 minutes

Docker Toolbox provides a way to use Docker on older Windows systems that do not meet minimal system requirements for the [Docker for Windows](https://docs.docker.com/docker-for-windows/) app.

If you have not done so already, download the installer here:

[Get Docker Toolbox for Windows](https://download.docker.com/win/stable/DockerToolbox.exe)

**What you get and how it works**

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

Because the Docker Engine daemon uses Linux-specific kernel features, you can’t run Docker Engine natively on Windows. Instead, you must use the Docker Machine command, docker-machine, to create and attach to a small Linux VM on your machine. This VM hosts Docker Engine for you on your Windows system.

**Tip**: One of the advantages of the newer [Docker for Windows](https://docs.docker.com/docker-for-windows/) solution is that it uses native virtualization and does not require VirtualBox to run Docker.

**Step 1: Check your version**

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:

1. Right click the windows message and choose **System**.

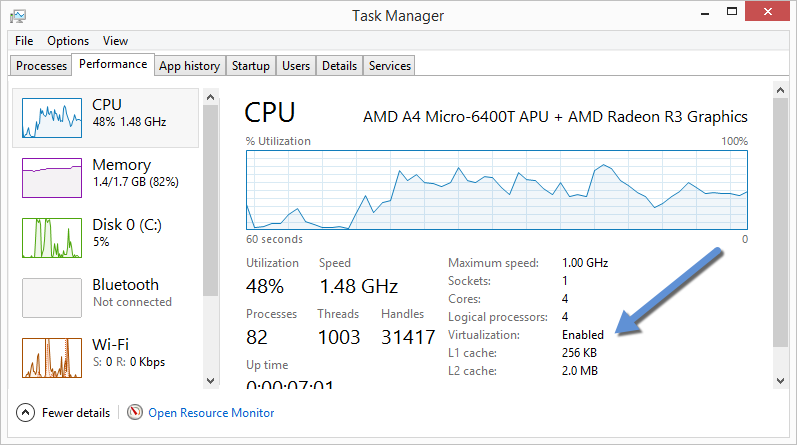
If you aren’t using a supported version, you could consider upgrading your operating system.

If you have a newer system, specifically 64bit Windows 10 Pro, with Enterprise and Education (1511 November update, Build 10586 or later), consider using [Docker for Windows](https://docs.docker.com/docker-for-windows) instead. It runs natively on the Windows, so there is no need for a pre-configured Docker QuickStart shell. It also uses Hyper-V for virtualization, so the instructions below for checking virtualization will be out of date for newer Windows systems. Full install prerequisites are provided in the Docker for Windows topic in [What to know before you install](https://docs.docker.com/docker-for-windows/#what-to-know-before-you-install).

1. Make sure your Windows system supports Hardware Virtualization Technology and that virtualization is enabled.

**For Windows 8 or 8.1**

Choose **Start > Task Manager** and navigate to the **Performance** tab. Under **CPU** you should see the following:



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

**For Windows 7**

Run the [Microsoft® Hardware-Assisted Virtualization Detection Tool](http://www.microsoft.com/en-us/download/details.aspx?id=592) and follow the on-screen instructions.

1. Verify your Windows OS is 64-bit (x64)

How you do this verification depends on your Windows version. For details, see the Windows article [How to determine whether a computer is running a 32-bit version or 64-bit version of the Windows operating system](https://support.microsoft.com/en-us/kb/827218).

**Step 2: Install Docker Toolbox**

In this section, you install the Docker Toolbox software and several “helper” applications. The installation adds the following software to your machine:

* Docker Client for Windows
* Docker Toolbox management tool and ISO
* Oracle VM VirtualBox
* Git MSYS-git UNIX tools

If you have a previous version of VirtualBox installed, do not reinstall it with the Docker Toolbox installer. When prompted, uncheck it.

If you have Virtual Box running, you must shut it down before running the installer.

1. Go to the [Docker Toolbox](https://www.docker.com/toolbox) page.
2. Click the installer link to download.
3. 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.

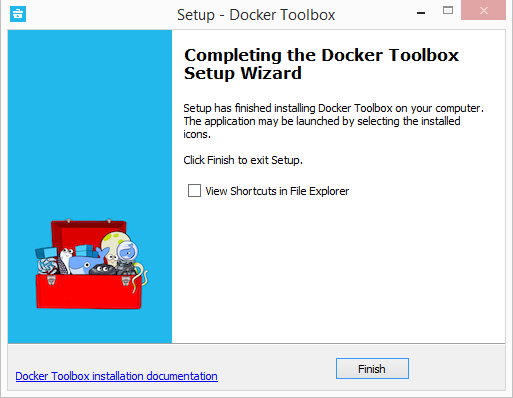


1. Press **Next** to accept all the defaults and then **Install**.

Accept all the installer defaults. The installer takes a few minutes to install all the components:

1. 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:

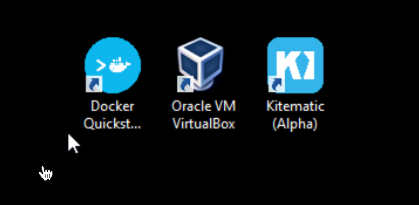


1. Uncheck “View Shortcuts in File Explorer” and press **Finish**.

**Step 3: Verify your installation**

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

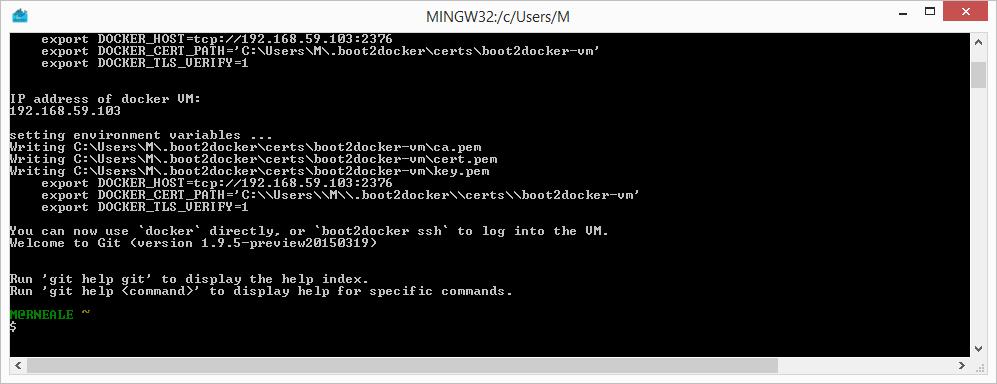
1. On your Desktop, find the Docker Toolbox icon.



1. Click the icon to launch a Docker Toolbox terminal.

If the system displays a **User Account Control** prompt to allow VirtualBox to make changes to your computer. Choose **Yes**.

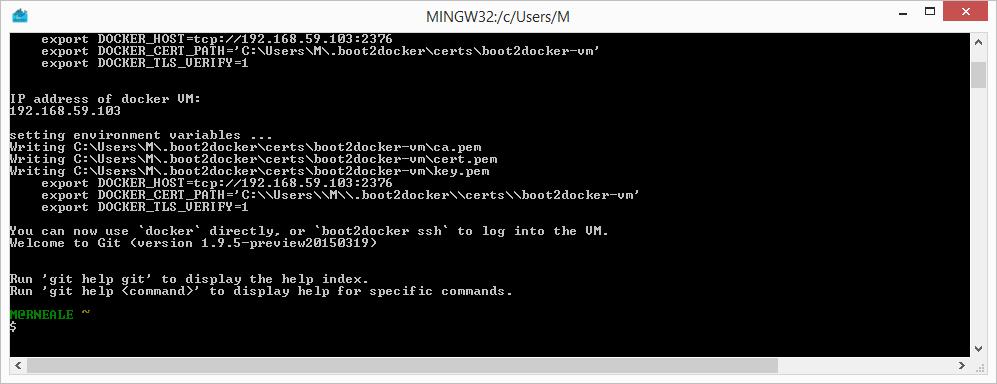
The terminal does several things to set up Docker Toolbox for you. When it is done, the terminal displays the $ prompt.



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 click your mouse next to the $ prompt.

If you aren’t familiar with a terminal window, here are some quick tips.



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 RETURN.

1. Type the docker run hello-world command and press RETURN.

The command does some work for you, if everything runs well, the command’s output looks like this:

$ docker run hello-world

Unable to find image 'hello-world:latest' locally

Pulling repository hello-world

91c95931e552: Download complete

a8219747be10: Download complete

Status: Downloaded newer image for hello-world:latest

Hello from Docker.

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker Engine CLI client contacted the Docker Engine daemon.

2. The Docker Engine daemon pulled the "hello-world" image from the Docker Hub.

(Assuming it was not already locally available.)

3. The Docker Engine daemon created a new container from that image which runs the

executable that produces the output you are currently reading.

4. The Docker Engine daemon streamed that output to the Docker Engine CLI client, which sent it

to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

$ docker run -it ubuntu bash

For more examples and ideas, visit:

https://docs.docker.com/userguide/

**Looking for troubleshooting help?**

Typically, the above steps work out-of-the-box, but some scenarios can cause problems. If your docker run hello-world didn’t work and resulted in errors, check out [Troubleshooting](https://docs.docker.com/toolbox/faqs/troubleshoot/) for quick fixes to common problems.

A Windows specific problem you might encounter has to do with the NDIS6 host network filter driver, which is known to cause issues on some Windows versions. For Windows Vista systems and newer, VirtualBox installs NDIS6 driver by default. Issues can range from system slowdowns to networking problems for the virtual machine (VM). If you notice problems, **re-run the Docker Toolbox installer**, and select the option to ***install VirtualBox with the NDIS5 driver***.

**How to uninstall Toolbox**

Removing Toolbox involves removing all the Docker components it includes.

A full uninstall also includes removing the local and remote machines you created with Docker Machine. In some cases, you might want to keep machines created with Docker Machine.

For example, if you plan to re-install Docker Machine as a part of Docker for Windows you can continue to manage those machines through Docker. Or, if you have remote machines on a cloud provider and you plan to manage them using the provider, you wouldn’t want to remove them. So the step to remove machines is described here as optional.

To uninstall Toolbox on Windows, do the following:

1. List your machines.
2. $ docker-machine ls
3. NAME ACTIVE DRIVER STATE URL SWARM
4. dev \* virtualbox Running tcp://192.168.99.100:2376
5. my-docker-machine virtualbox Stopped
6. default virtualbox Stopped
7. Optionally, remove each machine. For example:
8. $ docker-machine rm my-docker-machine
9. Successfully removed my-docker-machine

This step is optional because if you plan to re-install Docker Machine as a part of [Docker for Windows](https://docs.docker.com/docker-for-windows/), you can import and continue to manage those machines through Docker.

1. Uninstall Docker Toolbox using Window’s standard process for uninstalling programs through the control panel (programs and features).

**Note**: This process does not remove the docker-install.exe file. You must delete that file yourself.

1. Optionally, remove the C:\Users\<your-user>\.docker directory.

If you want to remove Docker entirely, you can verify that the uninstall removed the .docker directory under your user path. If it is still there, remove it manually. This directory stores some Docker program configuration and/or state (e.g., information about created machines such as certificates). Removing this directory is typically not necessary.

1. Uninstall Oracle VirtualBox, which is installed as a part of the Toolbox install.

https://docs.docker.com/toolbox/toolbox\_install\_windows/#how-to-uninstall-toolbox

# How to Use Docker on Windows

## Prerequisites

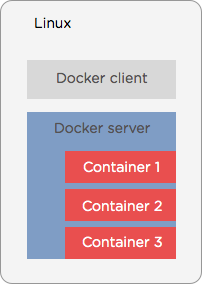
To follow this tutorial, you need a working installation of Windows 7 or 8 on your computer.

## How Docker works and how to make it work on windows

Before jumping into the installation process, let’s see how Docker works in order to understand what we’ll need to do to make it work on Windows.  
We must first view Docker as both a server and a client.

The server lets us build, download, start and stop images or containers.  
The client is just a command line tool that will allow us to communicate with the API of the server.

Here is what it looks like on Linux :

[](https://tutumcloud.files.wordpress.com/2014/11/diagram_linux.png)

The purpose of this tutorial is to use Windows to run both our Docker client and server; therefore using it as a Docker host. To do that we are going to use [Boot2docker](http://boot2docker.io/)

### Boot2docker

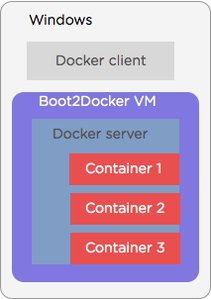
Since we aren’t using Linux, it is not possible ([yet](http://azure.microsoft.com/blog/2014/10/15/new-windows-server-containers-and-azure-support-for-docker/)) to use Docker natively. We are going to need some sort of lightweight VM that emulates a Docker Host.

This is what Boot2docker is for.

Boot2docker is a lightweight Linux distribution based on Tiny Core Linux made specifically to run Docker containers. It runs completely from RAM, weighs ~27MB and boots in ~5s

Source : [boot2docker.io](http://boot2docker.io)

Basically, Boot2Docker will encapsulate our Docker server into a virtual machine and let us access it through the Windows Docker client.

[](https://tutumcloud.files.wordpress.com/2014/11/diagram_win.png)

## Installation

* First, download and run the latest version of Boot2docker [here](https://github.com/boot2docker/windows-installer/releases/tag/v1.3.0). The Windows version of Boot2docker contains the Docker for Windows installer. It will install a set of tools (VirtualBox, Boot2docker ISO, MSYS-git UNIX tools) to help us run Docker.
* We now have a new shortcut on the Desktop called **Boot2Docker Start**. This shortcut is a shell script that will initiate the Boot2docker virtual machine and also the docker client. It will ask you for a ssh passphrase; you can either type one or just hit enter.  
  Boot2docker then starts initializing the VM and the Docker client. However, unlike Docker on OSX, we can’t use the Docker client directly from Windows but only from the Boot2docker VM.  
  We are now logged into the boot2docker VM and ready to use Docker.

## Running Docker containers

Now that the install process is finally finished, we have the ability to launch containers, pull images, etc. as if we were using Docker on Linux.

First, type :

|  |  |
| --- | --- |
| 1 | docker |

to ensure that everything is working.

Boot2docker comes with a sample image named **hello-world** that lets new users quickly launch a container. Let’s give it a try :

|  |  |
| --- | --- |
| 1 | docker run hello-world |

We should see a “hello world” message printed on the screen.

### Port Redirection

Redirecting ports with Boot2docker is really easy. The command is the same as on Linux, i.e. by using the -p flag like this :

|  |  |
| --- | --- |
| 1 | docker run -p PUBLIC\_PORT:PRIVATE\_CONTAINER\_PORT IMAGE CMD |

For instance, if we wanted to run the redis image on port 6379, we would run :

|  |  |
| --- | --- |
| 1 | docker run -p 6379:6379 redis |

The only thing that differs from running this command on Windows instead of Linux is that we will need the Boot2docker IP address in order to access that redis server.

In order to get it we can run the following command :

|  |  |
| --- | --- |
| 1 | boot2docker ip |

### Sharing a Windows folder

This part will explain how to properly set up a shared folder between Windows and Boot2docker. However, as this feature is not officially included we will need to install additional tools.

First, we need to get cifs-utils on the Boot2docker VM

|  |  |
| --- | --- |
| 1  2  3 | wget <http://distro.ibiblio.org/tinycorelinux/5.x/x86/tcz/cifs-utils.tcz>    tce-load -i cifs-utils.tcz |

Then, we create a folder

|  |  |
| --- | --- |
| 1 | mkdir /mnt/sharefolder |

This will be used as a mount point for our shared folder.

We will then need to create and share a new folder on Windows. To keep it simple, let’s just share the folder with ourselves.

Finally, we mount the shared folder on the Boot2docker VM

|  |  |
| --- | --- |
| 1 | sudo mount -t cifs //WINDOWS\_IP/shared /mnt/sharefolder -o username=WINDOWS\_USERNAME |

We will be asked to enter our Windows password and the folder will be mounted.

This will allow us to create volume containers (instructions [here](https://github.com/boot2docker/boot2docker#folder-sharing)) or share Dockerfiles between Windows and the VM for instance.

## Further details on Boot2docker

### User and password

We might need the username and the password of the Boot2docker user. By default the username is **docker** and the password is **tcuser**.

### Upgrade Boot2docker

In order to upgrade Boot2docker:

* Download the latest version of Docker for Windows and run the installer, after which the Boot2docker management tool is updated.
* Upgrade the VM

|  |  |
| --- | --- |
| 1  2  3  4  5 | boot2docker stop    boot2docker download    boot2docker start |

## Conclusion

The main goal of this tutorial was to explain how to install a working Docker environment on Windows using Boot2docker. However, with the recent announcements regarding Docker Windows Support on Azure and Windows Server, a cleaner way to run Docker without VMs may arise.

https://blog.tutum.co/2014/11/05/how-to-use-docker-on-windows/

# Install Docker Enterprise Edition for Windows Server 2016

Docker Enterprise Edition for Windows Server 2016 (*Docker EE*) enables native Docker containers on Windows Server 2016. The Docker EE installation package includes everything you need to run Docker on Windows Server 2016. This topic describes pre-install considerations, and how to download and install Docker EE.

**Already have Docker EE for Windows?** If you already have Docker EE for Windows installed, and you’re ready to get started, skip to [Get started with Docker for Windows](https://docs.docker.com/docker-for-windows/) for a quick tour of the command line, settings, and tools.

**Looking for Release Notes?** [Get release notes for all versions here](https://docs.docker.com/release-notes/) or subscribe to the [releases feed on the Docker Blog](http://blog.docker.com/category/engineering/docker-releases/).

**Install Docker EE**

Docker EE for Windows requires Windows Server 2016. See [What to know before you install](https://docs.docker.com/docker-ee-for-windows/#what-to-know-before-you-install) for a full list of prerequisites. To install Docker Community Edition (*Docker CE*) on a Windows 10 machine, see [Install Docker for Windows](https://docs.docker.com/docker-for-windows/install/).

1. Open a PowerShell command prompt, and type the following commands.
2. PS> Install-Module -Name DockerMsftProvider -Force
3. PS> Install-Package -Name docker -ProviderName DockerMsftProvider -Force
4. PS> Restart-Computer -Force
5. Test your Docker EE installation by running the hello-world container.
6. PS> docker run hello-world:nanoserver
7. Unable to find image 'hello-world:nanoserver' locally
8. nanoserver: Pulling from library/hello-world
9. bce2fbc256ea: Pull complete
10. 3ac17e2e6106: Pull complete
11. 8cac44e17f16: Pull complete
12. 5e160e4d8db3: Pull complete
13. Digest: sha256:25eac12ba40f7591969085ab3fb9772e8a4307553c14ea72d0e6f98b2c8ced9d
14. Status: Downloaded newer image for hello-world:nanoserver
15. Hello from Docker!
16. This message shows that your installation appears to be working correctly.
17. <snip>

**Using a script to install Docker EE**

Use the following steps when you want to install manually, script automated installs, or install on air-gapped systems.

1. In a PowerShell command prompt, download the installer archive on a machine that has a connection.
2. # On an online machine, download the zip file.
3. PS> invoke-webrequest -UseBasicparsing -Outfile docker.zip https://download.docker.com/components/engine/windows-server/17.03/docker-17.03.0-ee.zip
4. Copy the zip file to the machine where you want to install Docker. In a PowerShell command prompt, use the following commands to extract the archive, register, and start the Docker service.
5. # Extract the archive.
6. PS> Expand-Archive docker.zip -DestinationPath $Env:ProgramFiles
7. # Clean up the zip file.
8. PS> Remove-Item -Force docker.zip
9. # Install Docker. This will require rebooting.
10. $null = Install-WindowsFeature containers
11. # Add Docker to the path for the current session.
12. PS> $env:path += "$env:ProgramFiles\docker"
13. # Optionally, modify PATH to persist across sessions.
14. PS> $newPath = "$env:ProgramFiles\docker;" +
15. [Environment]::GetEnvironmentVariable("PATH",
16. [EnvironmentVariableTarget]::Machine)
17. PS> [Environment]::SetEnvironmentVariable("PATH", $newPath,
18. [EnvironmentVariableTarget]::Machine)
19. # Register the Docker daemon as a service.
20. PS> dockerd --register-service
21. # Start the daemon.
22. PS> Start-Service docker
23. Test your Docker EE installation by running the hello-world container.
24. PS> docker run hello-world:nanoserver

**Install Docker EE using OneGet**

If you want to install Docker EE by using [OneGet](https://github.com/oneget/oneget), follow the steps described in [Windows Containers on Windows Server](https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/quick-start-windows-server).

**What to know before you install**

* **What the Docker EE for Windows install includes**: The installation provides [Docker Engine](https://docs.docker.com/engine/userguide/intro/) and the [Docker CLI client](https://docs.docker.com/engine/reference/commandline/cli/).

**About Docker EE containers and Windows Server 2016**

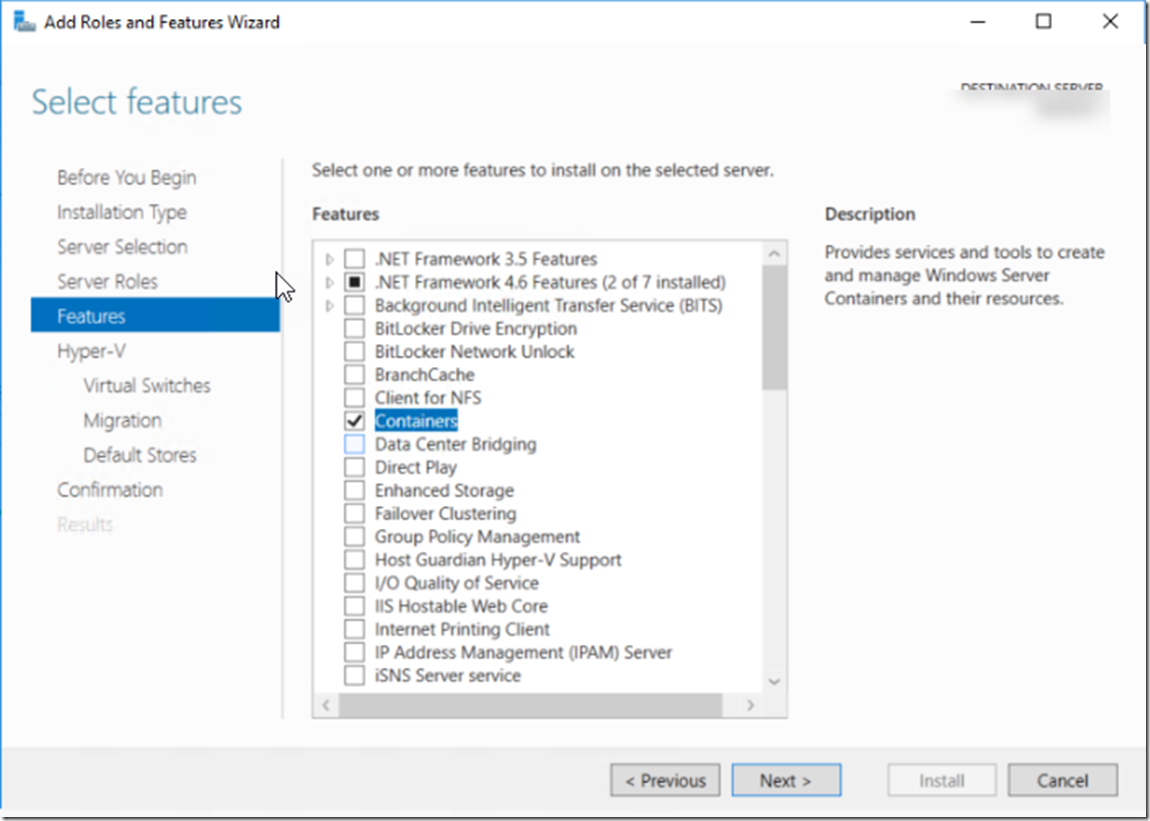
Looking for information on using Docker EE containers?

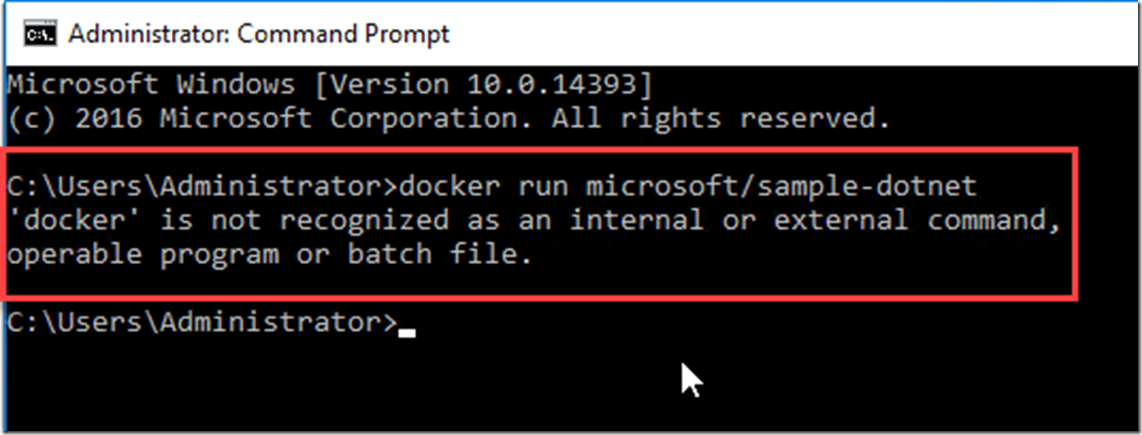
* [Getting Started with Windows Containers (Lab)](https://github.com/docker/labs/blob/master/windows/windows-containers/README.md) provides a tutorial on how to set up and run Windows containers on Windows 10 or Windows Server 2016. It shows you how to use a MusicStore application with Windows containers.
* [Setup - Windows Server 2016 (Lab)](https://github.com/docker/labs/blob/master/windows/windows-containers/Setup-Server2016.md) describes environment setup in detail.
* Docker Container Platform for Windows Server 2016 [articles and blog posts](https://www.docker.com/microsoft/) on the Docker website.

https://docs.docker.com/engine/installation/windows/docker-ee/#where-to-go-next

a couple weeks ago i was delivering a session on Windows Server 2016.  During that session i covered the Container support in Windows Server 2016. I see huge possibilities with containers.  I’m far from a Docker expert but I’m diving head first in that pool.

Anyway,  like i was saying i covered the support for containers in WS2016. Well a few day later one of the attendees reached out and said “this think is busted!” with the two screenshots below.  “I installed the container feature, i rebooted and tried to load a sample container. The darn thing is busted….”

[](https://msdnshared.blob.core.windows.net/media/2016/10/image927.png)

[](https://msdnshared.blob.core.windows.net/media/2016/10/SNAGHTML1875cd73.png)

So, those were my marching orders.

1. Figure it out
2. write about it…

so here we are:

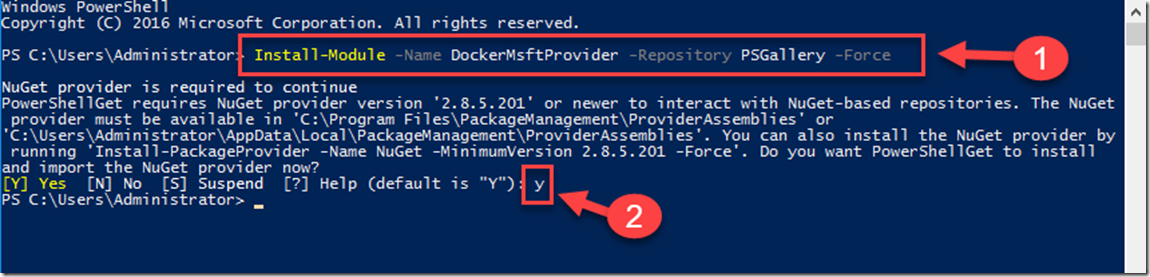
**Step 1- Install the container support.**

Todo so, we’ll use the [OneGet provider PowerShell module](https://github.com/oneget/oneget). The provider will enable the containers feature on your machine ( the same as if you'd use the GUI to Install the feature) and prep for the install of  Docker

The Container Feature is to enable this.  but you still need to install Docker itself ( that includes Docker Engine and the Docker client.) It is required in order to work with Windows containers.

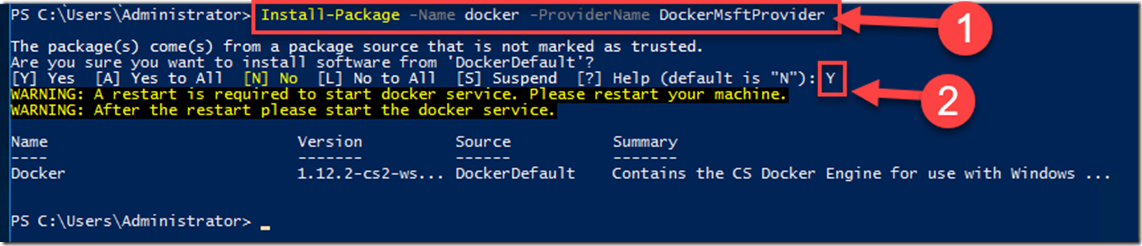
Open an elevated PowerShell session and run the following commands to install the OneGet PowerShell module.

Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

[](https://msdnshared.blob.core.windows.net/media/2016/10/image965.png)

Next we will install the latest version of Docker using the following command.  when prompted to tell you that the source is untrusted and whether or not you want to continue.  type “A” to continue.

Install-Package -Name docker -ProviderName DockerMsftProvider

[](https://msdnshared.blob.core.windows.net/media/2016/10/SNAGHTML1db3b5c3.png)

When the installation is complete, reboot the computer using this powershell command.

Restart-Computer -Force

**Step 2:  Test our Docker support by running a test container from Docker Hub.**

In this step we will download and run a pre-created .NET sample image from the [Docker Hub registry](https://docs.docker.com/docker-hub/).  The container will run a .Net Hello World application and shut itself down .

docker run microsoft/sample-dotnet

https://blogs.technet.microsoft.com/canitpro/2016/10/26/step-by-step-setup-docker-on-your-windows-2016-server/

Is there a full list of URLs that following two commands will need

Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

Install-Package -Name docker -ProviderName DockerMsftProvider

How do we download and install the Docker binaries manually (for a particular version ) ? Iam having issues with Install-Module, Install-Package PS Commands due to proxy. Hopefully there is a manual way that does not involve Powershell.

Invoke-WebRequest <https://dl.bintray.com/docker-compose/master/docker-compose-Windows-x86_64.exe> -UseBasicParsing -OutFile $env:ProgramFiles\docker\docker-compose.exe

It is work in progress but shall be able to get you started <http://goo.gl/l7mjTK>

Pre-Requisites are an active Internet Connection & SxS Source

Steps

1. Add Windows Feature [Containers] & restart.
2. Install these two Docker elements & restart.

Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

Install-Package -Name docker -ProviderName DockerMsftProvider

1. Check all the images available

Docker images

1. Install SQL Image

docker pull [with either of these three]

                                              microsoft/mssql-server-windows-express             [12 GB]

                                               microsoft/mssql-server-windows                              [14 GB]

                                                microsoft/mssql-server-windows-developer        [15 GB]

1. Start a Container

                docker run -d -p 1433:1433 -e sa\_password=Pa58Vv0Pd# -e ACCEPT\_EULA=Y microsoft/mssql-server-windows

1. List all running containers

docker PS

1. Get IP for the container

docker inspect -format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' [cf9 i.e. CONTAINER ID from the command above]

1. Login to container

docker exec -it [cf9 i.e. CONTAINER ID from the command above] sqlcmd

Under [1] Install Docker, the sequence I'd to run was

Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force

Install-Module -Name DockerMsftProvider -Force

Install-Package -Name docker -ProviderName DockerMsftProvider

Otherwise [Install-Package -Name docker -ProviderName DockerMsftProvider] causes

1. Install-Package : Unable to find package providers (DockerMsftProvider).

2. Commandlet Get-WindowsFeature not found.

**1. Install Docker**

To install Docker we'll use the [OneGet provider PowerShell module](https://github.com/oneget/oneget) which works with providers to perform the installation, in this case the [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.

First, install the Docker-Microsoft PackageManagement Provider from the [PowerShell Gallery](https://www.powershellgallery.com/packages/DockerMsftProvider).

none

Install-Module -Name DockerMsftProvider -Repository PSGallery -Force

Next, you use the PackageManagement PowerShell module to install the latest version of Docker.

none

Install-Package -Name docker -ProviderName DockerMsftProvider

When PowerShell asks you whether to trust the package source 'DockerDefault', type A to continue the installation. When the installation is complete, reboot the computer.

none

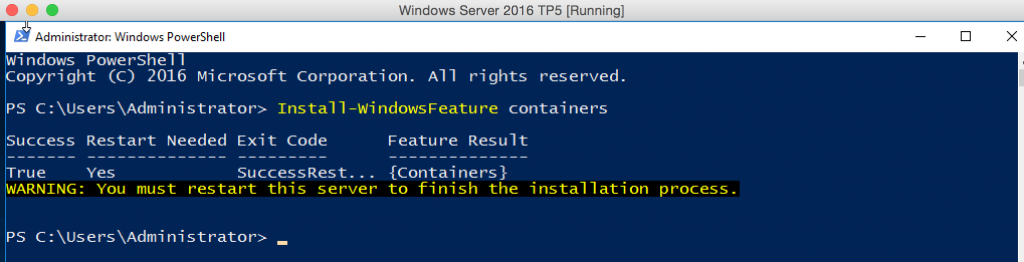
Restart-Computer -Force

Tip: If you want to update Docker later:

* Check the installed version with Get-Package -Name Docker -ProviderName DockerMsftProvider
* Find the current version with Find-Package -Name Docker -ProviderName DockerMsftProvider
* When you're ready, upgrade with Install-Package -Name Docker -ProviderName DockerMsftProvider -Update -Force, followed by Start-Service Docker

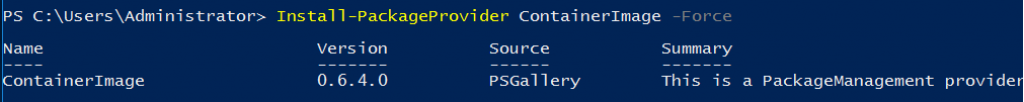
https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/quick-start-windows-server

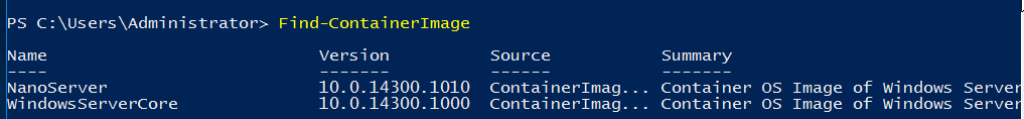
This multi-part blog series will show how to configure, build, and run Docker containers on Windows. The first part showed how to setup [Windows Server 2016 on Virtual Box for Docker Containers](http://blog.couchbase.com/2016/april/windows-server-2016-virtualbox-docker). This part will install the operating system images and configure Docker. Instructions are available at [msdn.microsoft.com/virtualization/windowscontainers/deployment/deployment](https://msdn.microsoft.com/virtualization/windowscontainers/deployment/deployment).

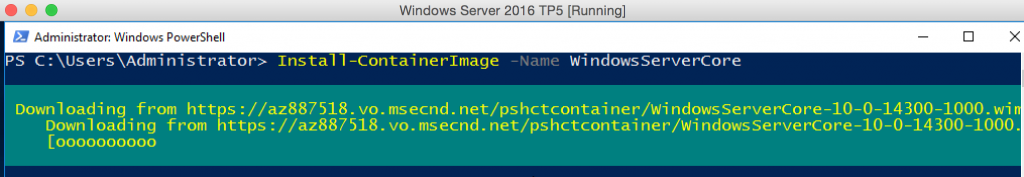
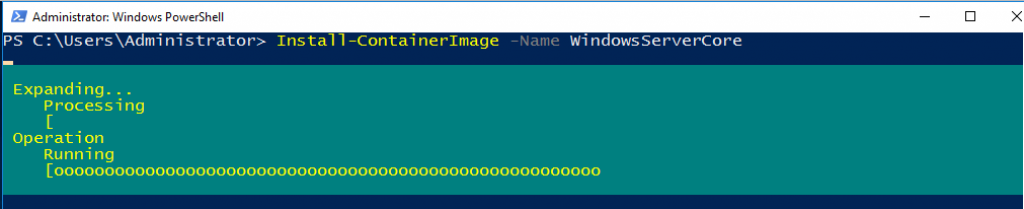
nstall the container feature:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2016-docker-2-1024x262.png)

Restart the Virtual Machine:

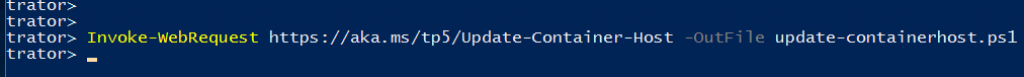
**Install Windows Server 2016 Base Docker Image**

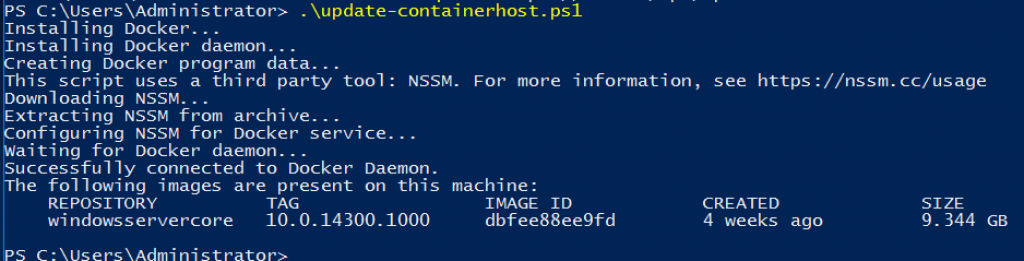
Base operating system can be installed using ContainerImage PowerShell module. Install the module as:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2016-docker-4-1024x102.png)

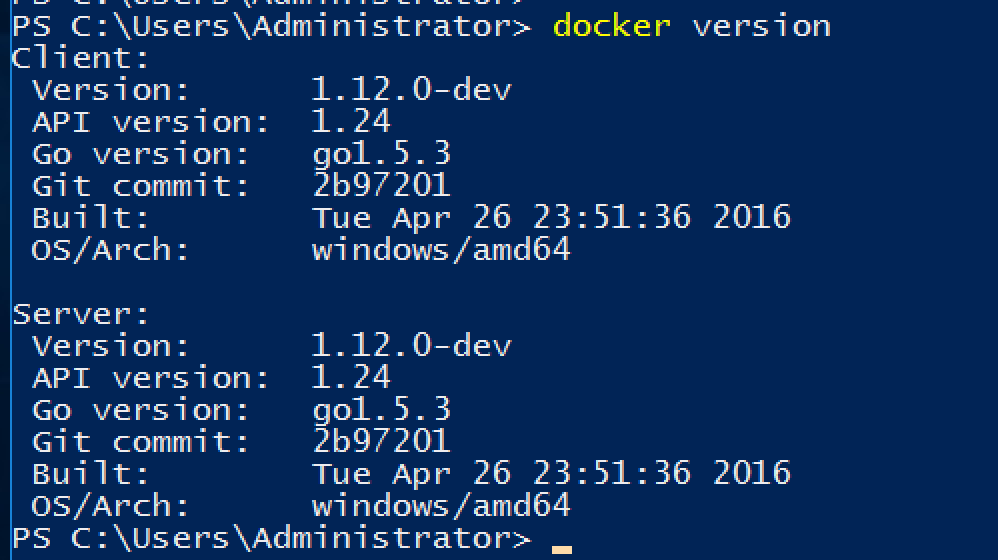
See the list of operating system images that are available:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2016-docker-5-1024x119.png)

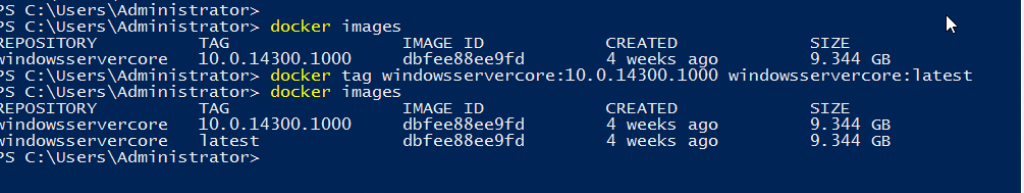
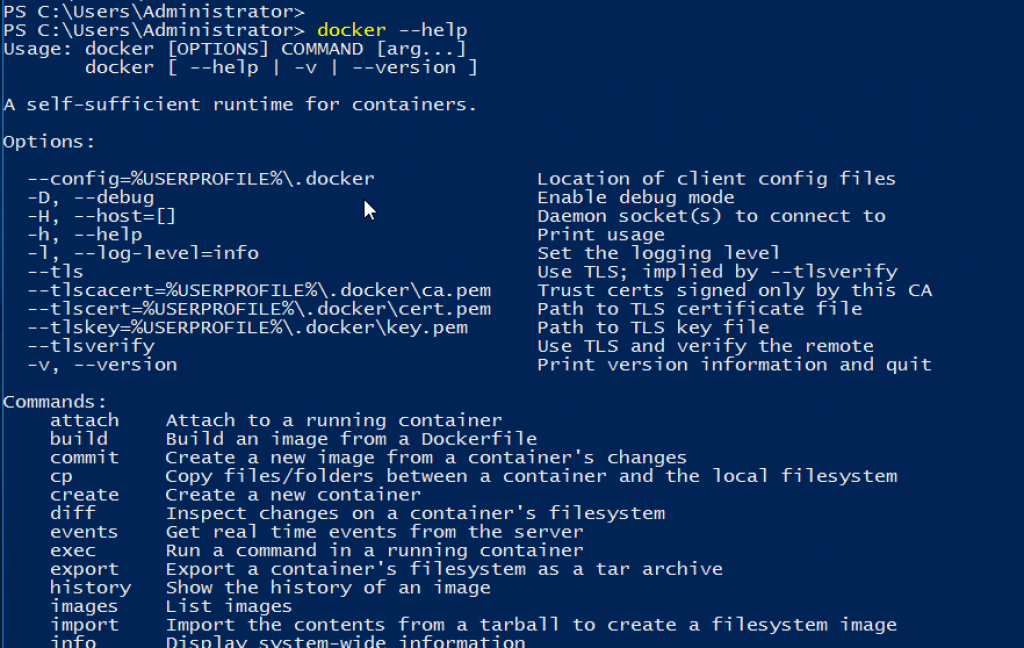
Install the Windows Server Core base OS image:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2016-docker-6-1024x177.png)[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2016-docker-7-1024x209.png)

**Install Docker on Windows Server 2016**

Download the script to install Docker:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2016-docker-8-1024x77.png)

Run the script:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2106-docker-9-1024x261.png)

Check Docker version:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2016-docker-11.png)

Check the list of images and tag the base OS image to latest:  
[](https://blog.couchbase.com/wp-content/original-assets/april-2016/setup-docker-on-windows-server-2016/windows-2106-docker-10-1024x193.png)  
Check docker --help command:  


Subsequent blogs will show how other fun things with Docker on Windows Server 2016!

Some further topics to read …

* Windows Server Containers – Quick Start
* [Windows Server 2016 Technology Preview 5](https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-technical-preview)
* [Docker Documentation](https://docs.docker.com/)
* Getting Started with Docker for Java developers

https://blog.couchbase.com/setup-docker-windows-server-2016/

[Docker](https://www.docker.com/what-docker) and Docker-based containers have been a big deal in the free and open source software (FOSS) space for a while now. In a nutshell, Docker containers are virtualized applications that run in their own isolated memory space and that have their own "sandboxed" file system.

Docker containers are a big deal because they:

* Are much smaller and more agile than full virtual machines
* Can be spun up and destroyed in seconds
* Reduce the attack surface of your applications

Find out how to get a running start on installing the Containers server role and deploying Docker containers by using native Docker commands by following these steps. You can find more at the nearly always-excellent [Microsoft Developer Network (MSDN) documentation](https://msdn.microsoft.com/en-us/virtualization/windowscontainers/quick_start/quick_start_windows_server).

Please keep the following in mind: These steps are based on a prerelease code of Windows Server 2016. Also, you can manage Docker containers by using [native PowerShell commands](https://msdn.microsoft.com/en-us/virtualization/windowscontainers/management/docker-powershell).

Now go ahead and set up a virtual machine running [Windows Server 2016 Technical Preview 5 (TP5)](https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-technical-preview) and let's get busy.

### Installing the Containers Feature and the Docker Engine

Open an administrative PowerShell console and install the new Containers feature:

[Install-WindowsFeature](https://technet.microsoft.com/en-us/library/jj205467%28v=wps.630%29.aspx) -Name Containers -Restart

Let's now create a folder to house the Docker program files:

New-Item -Type Directory -Path 'C:Program FilesDocker' -Force

The following two [Invoke-WebRequest](https://technet.microsoft.com/en-us/library/hh849901.aspx) calls download the Docker engine (daemon in UNIX language) and the Docker client from the Microsoft servers:

Invoke-WebRequest -Uri https://aka.ms/tp5/b/dockerd -OutFile $env:ProgramFilesDockerdockerd.exe -UseBasicParsing

Invoke-WebRequest -Uri https://aka.ms/tp5/b/docker -OutFile $env:ProgramFilesDockerdocker.exe -UseBasicParsing

We should add the Docker directory to our system path so we can call the Docker client from wherever we are in the file system.

[Environment]::SetEnvironmentVariable("Path", $env:Path + ";C:Program FilesDocker", [EnvironmentVariableTarget]::Machine)

You'll have to restart your administrative PowerShell console to put the environment variable change into effect.

To wrap up installation, we'll install the Docker daemon as a Windows service by calling on the **dockerd** executable directly:

dockerd --register-service

And finally fire up the Docker service:

Start-Service -Name Docker -Force

### Downloading the Base Images

In Docker container nomenclature, the image is the template from which you spawn new containers. We can download some pre-built Docker images from the Microsoft servers by installing the container image package provider:

Install-PackageProvider -Name ContainerImage -Force

As of this writing, Microsoft has two container images in their gallery: Nano Server and Server Core. Let's download them both:

Install-ContainerImage -Name WindowsServerCore

Install-ContainerImage -Name NanoServer

Once the image installation process completes (it can take a while, depending on your Internet connection speed), you'll need to restart the Docker service:

Restart-Service -Name Docker -Force

### Deploy Your First Docker Container

Microsoft engineers actually figured out how to run the Windows Server operating system as a container.

To get a list of your Nano Server and Server Core images, run the following command:

docker images

Now we'll use [docker run](https://docs.docker.com/engine/reference/commandline/run/) to deploy a new container named **coreserver3** that uses the Windows Server Core image. The -it switch denotes an interactive session, and **cmd.exe** means that we want to enter the container inside a new cmd.exe console.

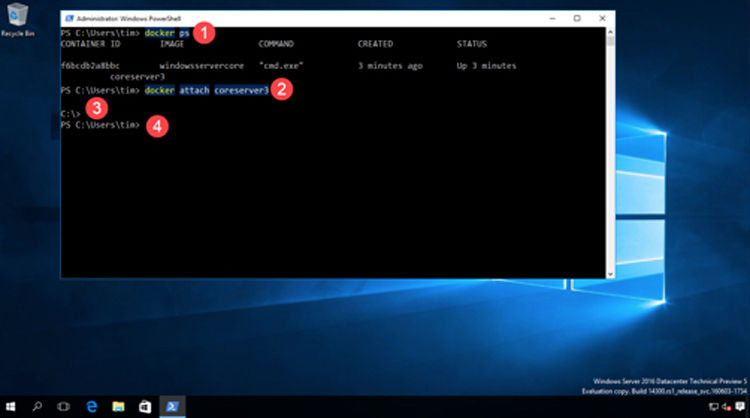
docker run -it --name coreserver3 windowsservercore cmd.exe

Specifically, our **docker run** statement translates to "Run the **cmd.exe** command from within a new Server Core-based container named **coreserver3**."

To switch out of the running container (which sadly runs in the same window as your previously open PowerShell session), use the following keystroke:

CTRL+P,Q

Then you can run [docker ps](https://docs.docker.com/engine/reference/commandline/ps/) to get a list of running containers, [docker attach](https://docs.docker.com/engine/reference/commandline/attach/) **coreserver3** to re-enter the running container, or [docker stop](https://docs.docker.com/engine/reference/commandline/stop/) **coreserver3** to stop the container.



### Windows Containers in Action

Take a look at the previous annotated screenshot and I'll break it all down for you:

1. Here we're in the host operating system at a PowerShell prompt.
2. Here we issue a command to enter the **coreserver3** container.
3. Here we're at the **cmd.exe** prompt within the container.
4. After using the **CTRL+P,Q** keystroke, we exit the container and are back at the host system's PowerShell prompt.

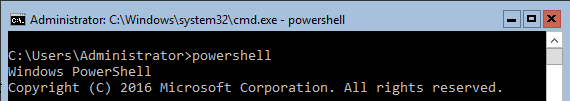
### Bottom Line

There's no question about it — the idea of using Docker containers in Windows Server takes some getting used to. The good news, though, is that I'm positive that the Windows Server engineering team will continue making the Containers feature easier for us Docker newbies to use.

http://www.tomsitpro.com/articles/how-to-deploy-windows-server-docker-containers,1-3326.html

https://dscottraynsford.wordpress.com/2016/10/15/install-docker-on-windows-server-2016-using-dsc/

On a **Windows Server 2016 Server Core** or **Windows Server 2016 Server Core with GUI** server:

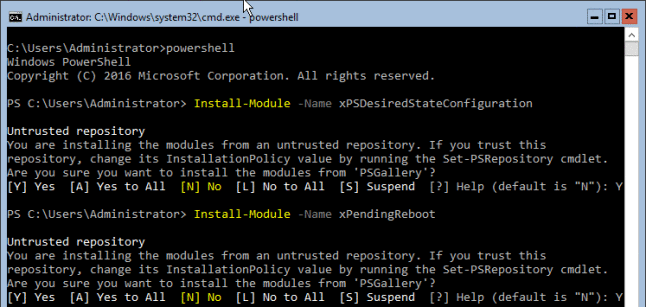
1. Log on as a user with **Local Administrator** privileges.
2. Start an **Administrator PowerShell** console – if you’re using Server Core just enter **PowerShell** at the command prompt:
3. **Install the DSC Resources** required for the DSC configuration by executing these commands:

|  |
| --- |
|  |
| Install-Module -Name xPSDesiredStateConfiguration | | |
|  | | |



|  |
| --- |
| Install-Module -Name xPendingReboot |

[view raw](https://gist.github.com/PlagueHO/71a8cf84d6d13e96a91a2c3ba5209a24/raw/da33705727f75cf4ed5bf1cffcfd499adf345a57/Install-ResourcesforDockerOnWS2016ByDSC.ps1) [Install-ResourcesforDockerOnWS2016ByDSC.ps1](https://gist.github.com/PlagueHO/71a8cf84d6d13e96a91a2c3ba5209a24#file-install-resourcesfordockeronws2016bydsc-ps1) hosted with by [GitHub](https://github.com)

*You may be asked to confirm installation of these modules, answer yes to any confirmations.*  


 **Download the Docker installation DSC script** by executing this command:

|  |
| --- |
|  |
|  Invoke-WebRequest -Uri 'https://gist.githubusercontent.com/PlagueHO/d9595cae1788f436b97bd4c90d50d72e/raw/1146baa2b1e0c8b3869004074b4c97bf71ce9c3c/Install-DockerOnWS2016ByDSC.ps1' -OutFile 'Install-DockerOnWS2016ByDSC.ps1' | |

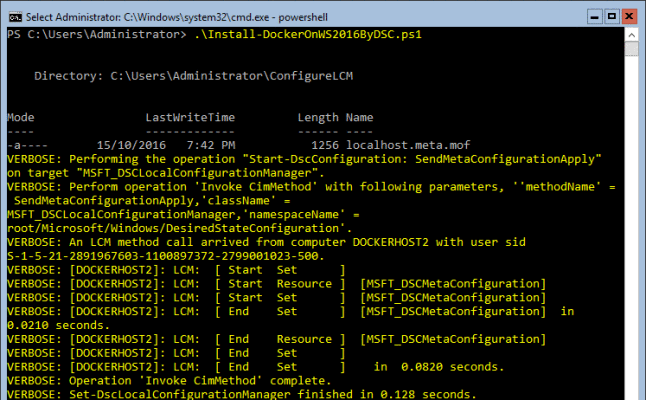
[view raw](https://gist.github.com/PlagueHO/00f302a9299702cebbbd2b40de264f55/raw/9c9669b8fd551653d03f2f7e0373ef296bf0abd0/Download-Install-DockerOnWS2016ByDSC.ps1) [Download-Install-DockerOnWS2016ByDSC.ps1](https://gist.github.com/PlagueHO/00f302a9299702cebbbd2b40de264f55#file-download-install-dockeronws2016bydsc-ps1) hosted with by [GitHub](https://github.com)

ss_dockerdsc_consoledownloadscript

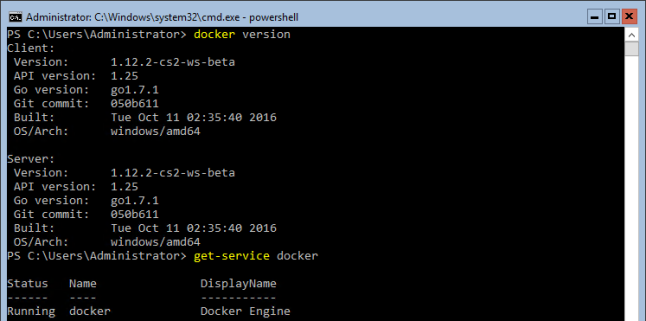
 **Run the Docker installation DSC script** by executing this command:

|  |
| --- |
|  |

|  |
| --- |
| .\Install-DockerOnWS2016ByDSC.ps1 |

1. [view raw](https://gist.github.com/PlagueHO/ef200c931094aa19e752c0429df417bc/raw/fa02a3e6c8d55a78ba9cd24ea8ca525fcc2a9fcd/Run-Install-DockerOnWS2016ByDSC.ps1) [Run-Install-DockerOnWS2016ByDSC.ps1](https://gist.github.com/PlagueHO/ef200c931094aa19e752c0429df417bc#file-run-install-dockeronws2016bydsc-ps1) hosted with by [GitHub](https://github.com)
2. 

The script will run and reboot the server once. Not long after the reboot the Docker service will start up and you can get working with containers:



You’re now ready to start working with Containers.

**What the Script Does**

In case you’re interested in what the script actually contains, here are the components:

1. **Configuration ContainerHostDsc –** the [DSC configuration](https://msdn.microsoft.com/en-us/powershell/dsc/configurations) that configures the node as a Docker Container host.
2. **Configuration ConfigureLCM** – the [LCM meta configuration](https://msdn.microsoft.com/en-us/powershell/dsc/metaconfig) that sets **Push Mode**, allows the LCM to reboot the node if required and configures **ApplyAndAutoCorrect** mode.
3. **ConfigData** – a [ConfigData object](https://msdn.microsoft.com/en-us/powershell/dsc/configdata) that contains the list of node names to apply this DSC Configuration to – in this case LocalHost.
4. **ConfigureLCM** – the call to the **Configuration ConfigureLCM** to [compile the LCM meta configuration MOF file](https://msdn.microsoft.com/en-us/powershell/dsc/metaconfig).
5. **Set-DscLocalConfigurationManager** – this applies the compiled LCM meta configuration MOF file to LocalHost to configure the LCM.
6. **ContainerHostDsc** – the call to the **Configuration ContainerHostDsc** to compile the DSC MOF file.
7. **Start-DSCConfiguration** – this command starts the LCM applying the DSC MOF file produces by the **ContainerHostDsc**.

https://docs.docker.com/docker-for-windows/install/#where-to-go-next

https://blog.docker.com/2016/09/build-your-first-docker-windows-server-container/

https://www.docker.com/docker-windows-server