Docker Tutorial

Docker Tutorial provides basic and advanced concepts of Docker. Our Docker Tutorial is designed for both beginners as well as professionals.

Docker is a **centralized platform** for **packaging, deploying, and running applications**. Before Docker, many users face the problem that a particular **code is running** in the **developer's system but not in the user's system**. So, the main **reason** to develop docker is to help **developers to develop applications** easily, ship them into **containers**, and can be **deployed** anywhere.

Docker was firstly released in March 2013. It is used in the **Deployment stage** of the **software development life cycle** that's why it can efficiently resolve issues related to the application deployment.

What is Docker?

Docker is an **open-source centralized platform designed** to **create, deploy, and run applications**. Docker uses **container** on the **host's operating system** to run applications. It allows **applications** to **use** the **same Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our application **works** in any environment like **development, test, or production.**

Docker includes components such as **Docker client, Docker server, Docker machine, Docker hub, Docker composes,** etc.

Let's understand the Docker containers and virtual machine.

Docker Containers

Docker containers are the lightweight **alternatives** of the **virtual machine**. It allows developers to **package** up the **application** with all its **libraries and dependencies**, and **ship** it as a **single package**. The **advantage** of using a **docker container** is that you **don't need** to **allocate any RAM and disk space** for the applications. It automatically **generates storage and space** according to the application requirement.

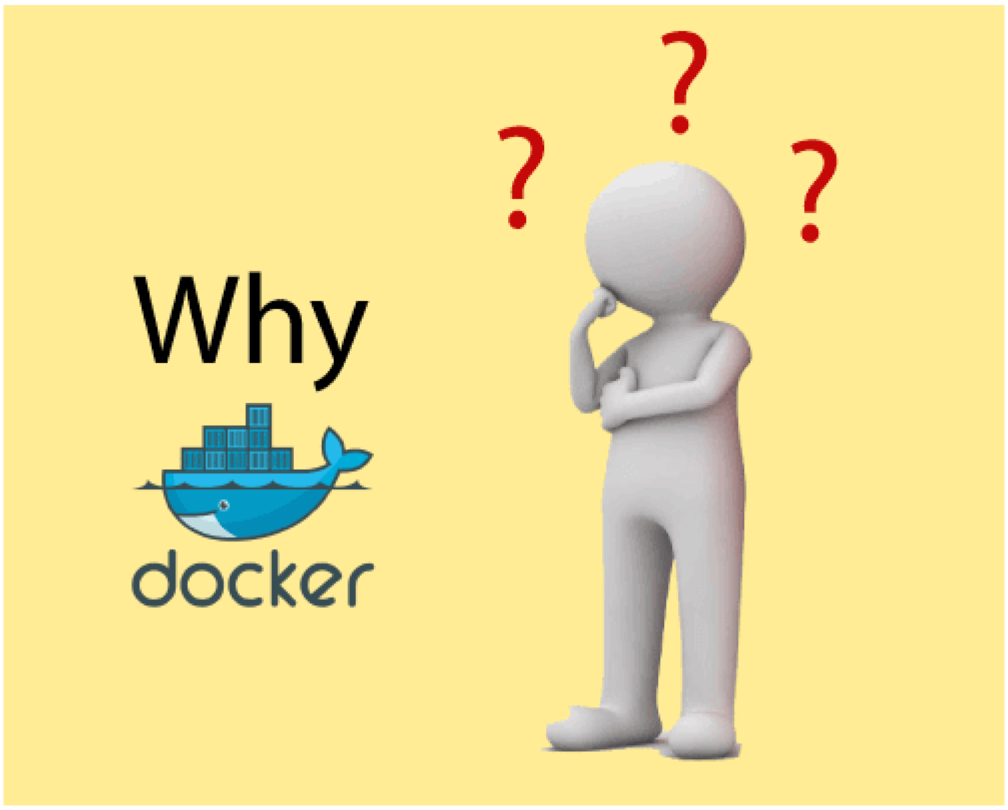
Virtual Machine

A **virtual machine** is a **software** that allows us to **install** and use **other operating systems** (Windows, Linux, and Debian) simultaneously on our machine. The **operating system** in which **virtual machine** runs are called **virtualized operating systems**. These **virtualized operating systems** can **run programs and performs tasks** that we perform in a **real operating system**.

Containers Vs. Virtual Machine

Containers	Virtual Machine
Integration in a container is **faster** and **cheap**.	**Integration** in virtual is **slow** and **costly**.
No wastage of memory.	**Wastage** of memory.
It uses the **same kernel**, but **different distribution.**	It uses **multiple independent operating systems.**

Why Docker?



Docker is **designed** to **benefit** both the **Developer and System Administrator**. There are the following reasons to use Docker -

- Docker **allows** us to **easily install and run software** without worrying about setup or dependencies.
- **Developers** use Docker to **eliminate machine problems**, i.e. "but code is worked on my laptop." when working on code together with co-workers.
- **Operators** use Docker to **run** and **manage apps** in isolated containers for better **compute density**.
- **Enterprises** use Docker to **securely built agile software delivery pipelines** to ship new application features faster and more securely.
- Since docker is not only used for the deployment, but it is also a **great platform** for **development**, that's why we can efficiently **increase** our **customer's satisfaction**.

Advantages of Docker

There are the following **advantages** of Docker -

- It **runs** the **container** in **seconds** instead of minutes.

- It uses **less memory**.
- It **provides** lightweight **virtualization**.
- It does **not** a **require full operating system** to run applications.
- It uses application dependencies to **reduce** the **risk**.
- Docker **allows** you to **use** a **remote repository** to share your container with others.
- It **provides continuous deployment** and **testing environment**.

Disadvantages of Docker

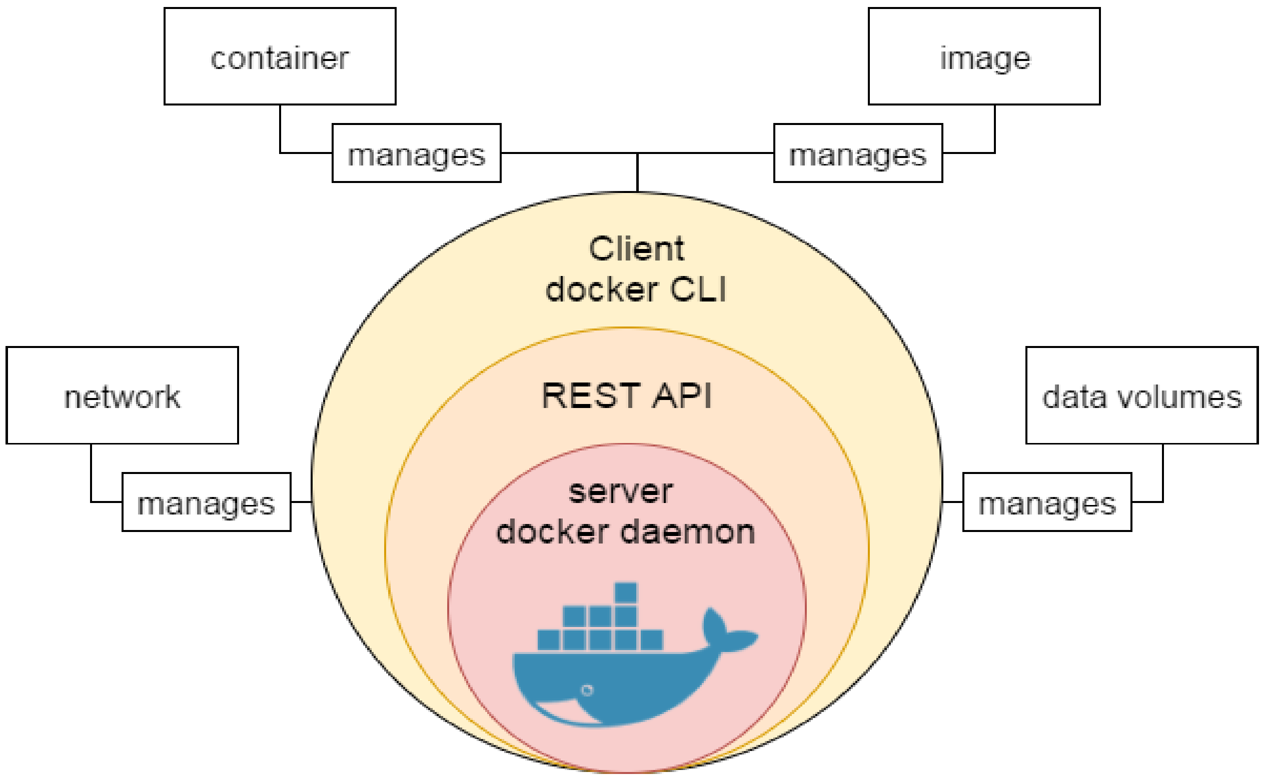
There are the following **disadvantages** of Docker -

- It **increases complexity** due to an additional layer.
- In Docker, it is **difficult** to **manage large number** of **containers**.
- Some **features** such as container self -registration, containers self-inspect, copying files form host to the container, and more are **missing** in the **Docker**.
- Docker is **not a good solution for applications** that require **rich graphical interface.**
- Docker **provides cross-platform compatibility** means if an application is designed to run in a Docker container on Windows, then it can't run on Linux or vice versa.

Docker Engine

It is a **client server application** that contains the following **major components**.

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



Prerequisite

Before learning Docker, you must have the fundamental knowledge of Linux and programming languages such as java, php, python, ruby, etc.

Audience

Our Docker Tutorial is designed to help beginners and professionals.

Problem

We assure that you will not find any difficulty while learning our Docker tutorial. But if there any mistake, kindly post the problem in the contact form.

Docker Features

Although Docker provides lots of features, we are listing some **major features** which are given below.

- **Easy and Faster Configuration**
- **Increase productivity**
- **Application Isolation**
- **Swarm**
- **Routing Mesh**
- **Services**
- **Security Management**

Easy and Faster Configuration

This is a **key feature** of docker that helps us to **configure** the **system** easily and faster.

We can **deploy** our **code** in **less time and effort**. As Docker can be used in a wide variety of environments, the **requirements** of the **infrastructure** are **no longer linked with the environment** of the application.

Increase productivity

By **easing technical configuration** and **rapid deployment of application**. No doubt it has increase productivity. Docker not only helps to execute the application in isolated environment but also it has **reduced the resources**.

Application Isolation

It provides **containers** that are used to **run applications in isolation environment**. Each container is independent to another and allows us to execute any kind of application.

Swarm

It is a **clustering and scheduling tool for Docker containers**. Swarm uses the **Docker API** as its **front end**, which helps us to use various tools to control it. It also helps us to control a **cluster of Docker hosts as a single virtual host**. It's a self-organizing group of engines that is used to **enable pluggable backends**.

Routing Mesh

It **routes** the **incoming requests** for published **ports** on available **nodes to** an **active container**. This feature **enables** the **connection** even if there is no task is running on the node.

Services

Services is a **list of tasks** that lets us **specify** the **state** of the **container** inside a **cluster**. Each **task represents one instance** of a container that should be running and Swarm **schedules** them **across nodes**.

Security Management

It **allows** us to **save secrets** into the **swarm** itself and then choose to **give services access** to certain secrets.

It includes some important commands to the engine like **secret inspect, secret create** etc.

Docker Architecture

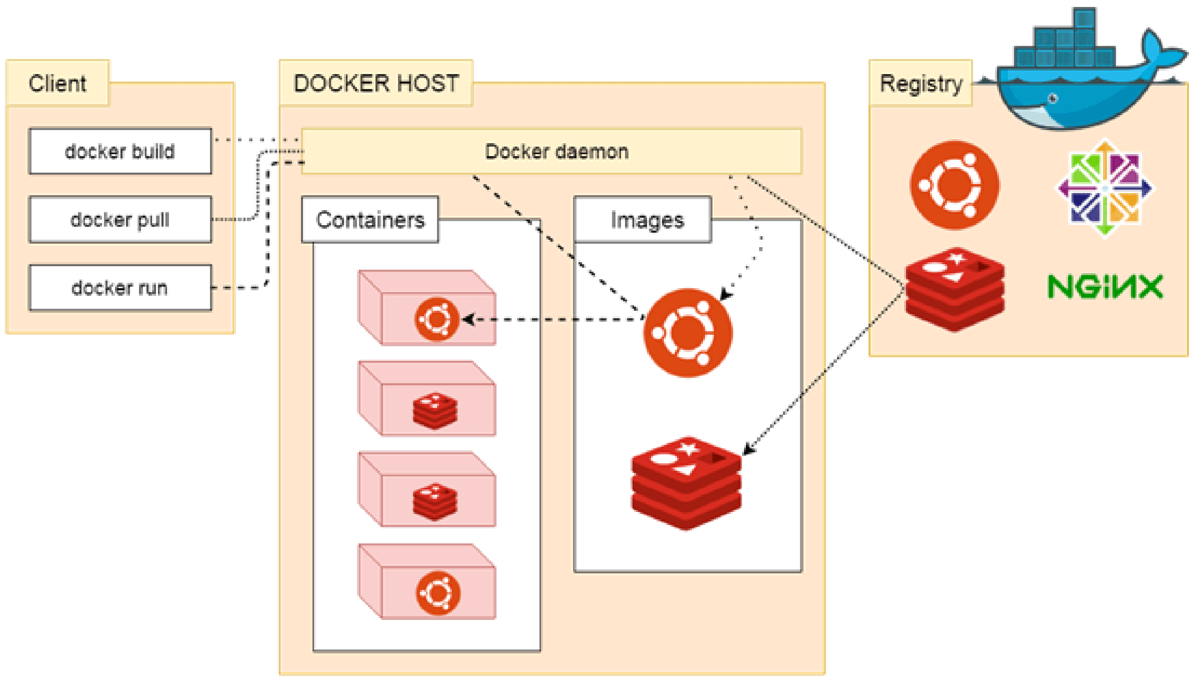
Before learning the Docker architecture, first, you should know about the Docker Daemon.

What is Docker daemon?

Docker **daemon runs** on the **host operating system**. It is **responsible** for **running containers** to **manage docker services**. Docker daemon communicates with other daemons. It **offers** various **Docker objects** such as **images, containers, networking, and storage**.

Docker architecture

Docker follows **Client-Server architecture**, which includes the **three main components** that are **Docker Client**, **Docker Host**, and **Docker Registry**.



1. Docker Client

Docker client uses **commands** and **REST APIs** to **communicate** with the **Docker Daemon (Server)**. When a client **runs** any **docker command** on the docker client **terminal**, the client **terminal sends** these docker **commands** to the **Docker daemon**. Docker **daemon receives** these **commands** from the **docker client** in the form of **command and REST API's request**.

Note: Docker Client has an ability to communicate with more than one docker daemon.

Docker Client uses **Command Line Interface (CLI)** to run the following commands -

docker build

docker pull

docker run

2. Docker Host

Docker Host is used to **provide** an **environment** to **execute** and **run applications**. It contains the **docker daemon, images, containers, networks, and storage**.

3. Docker Registry

Docker **Registry manages** and **stores** the Docker **images**.

There are two types of registries in the Docker -

Pubic Registry - Public Registry is also called as **Docker hub**.

Private Registry - It is used to **share images** within the **enterprise**.

Docker Objects

There are the following **Docker Objects** -

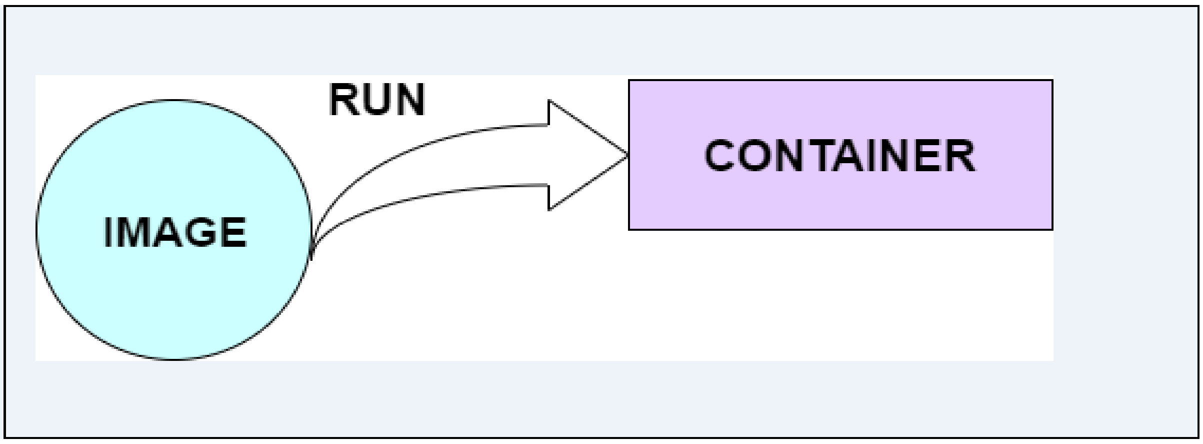
Docker Images

Docker images are the **read-only binary templates** used to **create Docker Containers**. It uses a **private container registry** to **share container images** within the **enterprise** and also uses **public container registry** to **share container images** within the **whole world**. **Metadata** is also used by **docker images** to **describe** the **container's abilities**.

Docker Containers

Containers are the **structural units of Docker**, which is used to **hold** the **entire package** that is needed to run the application. The **advantage** of containers is that it **requires very less resources**.

In other words, we can say that the **image is a template**, and the **container** is a **copy of that template.**



Docker Networking

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

- **Bridge -** Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.
- **Host -** It is used when we don't need for network isolation between the container and the host.
- **None -** It disables all the networking.
- **Overlay -** Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.
- **Macvlan -** Macvlan is used when we want to assign MAC addresses to the containers.

Docker Storage

Docker Storage is used to store data on the container. Docker offers the following options for the Storage -

- **Data Volume -** Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.
- **Directory Mounts -** It is one of the best options for docker storage. It mounts a host's directory into a container.
- **Storage Plugins -** It provides an ability to connect to external storage platforms.

Docker Installation

We can install docker on any operating system whether it is Mac, Windows, Linux or any cloud. Docker Engine runs natively on Linux distributions. Here, we are providing step by step process to install docker engine for Linux **Ubuntu Xenial-16.04 [LTS].**

Prerequisites:

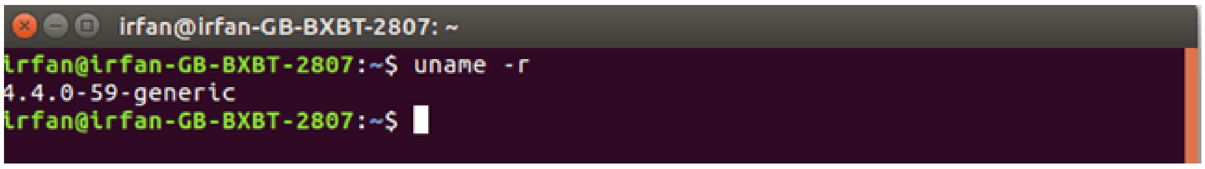
Docker need two important installation requirements:

- It only works on a 64-bit Linux installation.
- It requires Linux kernel version 3.10 or higher.

To check your current kernel version, open a terminal and type ***uname -r*** command to display your kernel version:

Command:

1. $ uname -r



```
irfan@irfan-GB-BXBT-2807:~$ uname -r
4.4.0-59-generic
irfan@irfan-GB-BXBT-2807:~$ 
```

Update apt sources

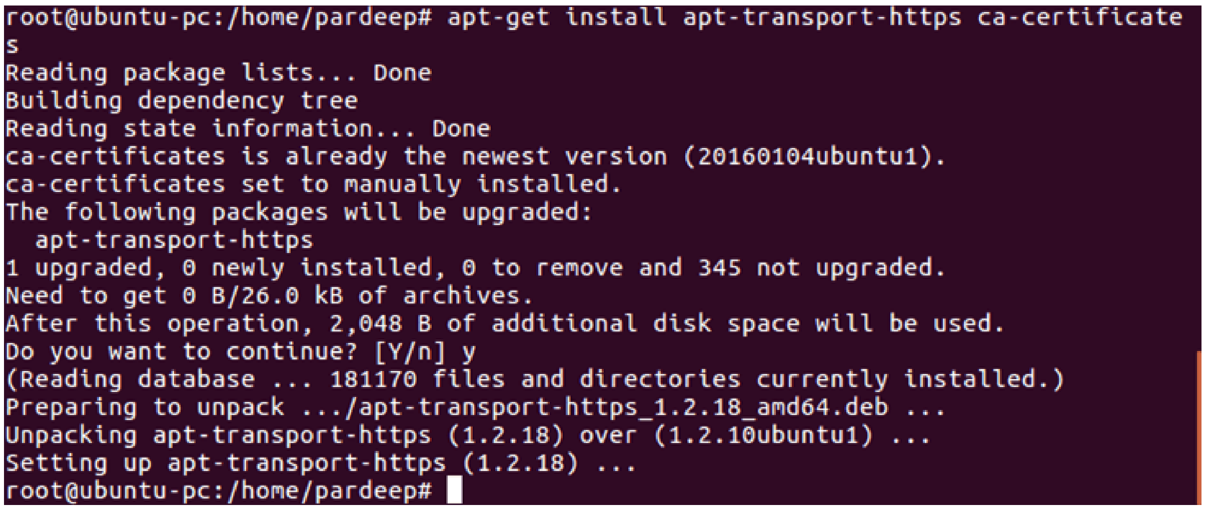
Follow following instructions to update apt sources.

1. Open a terminal window.
2. Login as a *root* user by using *sudo* command.
3. Update package information and install CA certificates.

   Command:

   1. $ apt-get update
   2. $ apt-get install apt-transport-https ca-certificates

   See, the attached screen shot below.



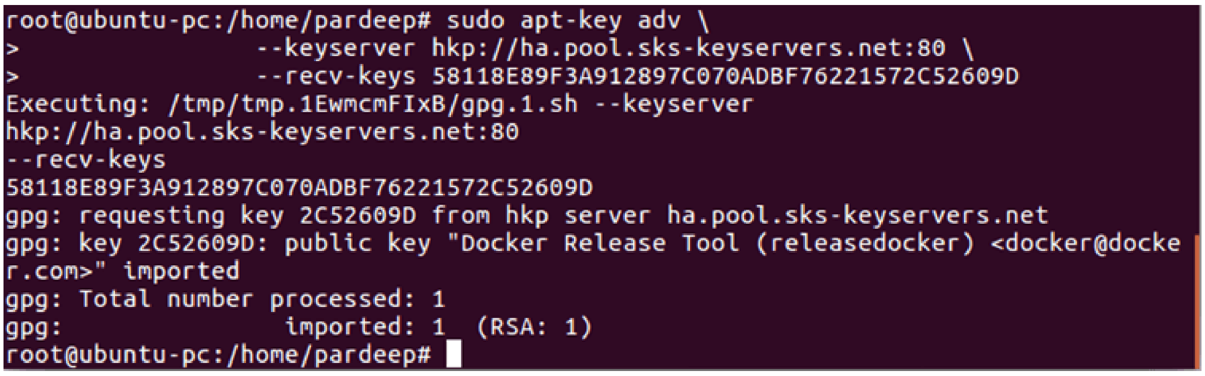
```
root@ubuntu-pc:/home/pardeep# apt-get install apt-transport-https ca-certificate
s
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20160104ubuntu1).
ca-certificates set to manually installed.
The following packages will be upgraded:
  apt-transport-https
1 upgraded, 0 newly installed, 0 to remove and 345 not upgraded.
Need to get 0 B/26.0 kB of archives.
After this operation, 2,048 B of additional disk space will be used.
Do you want to continue? [Y/n] y
(Reading database ... 181170 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_1.2.18_amd64.deb ...
Unpacking apt-transport-https (1.2.18) over (1.2.10ubuntu1) ...
Setting up apt-transport-https (1.2.18) ...
root@ubuntu-pc:/home/pardeep#
```

4. Add the new GPG key. Following command downloads the key.

   Command:

   1. $ sudo apt-key adv \
   2. --keyserver hkp://ha.pool.sks-keyservers.net:80 \
   3. --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

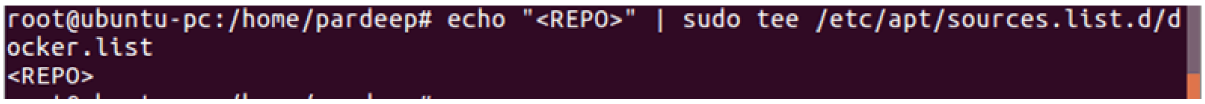
   Screen shot is given below.



```
root@ubuntu-pc:/home/pardeep# sudo apt-key adv \
>                --keyserver hkp://ha.pool.sks-keyservers.net:80 \
>                --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
Executing: /tmp/tmp.1EwmcmFIxB/gpg.1.sh --keyserver
hkp://ha.pool.sks-keyservers.net:80
--recv-keys
58118E89F3A912897C070ADBF76221572C52609D
gpg: requesting key 2C52609D from hkp server ha.pool.sks-keyservers.net
gpg: key 2C52609D: public key "Docker Release Tool (releasedocker) <docker@docke
r.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
root@ubuntu-pc:/home/pardeep#
```

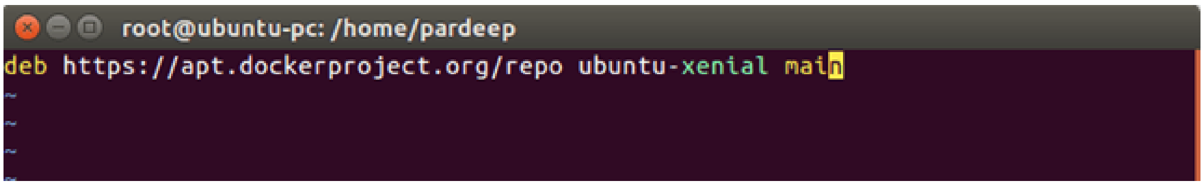
5. Run the following command, it will substitute the entry for your operating system for the file.
   1. $ echo "**<REPO>**" | sudo tee /etc/apt/sources.list.d/docker.list

   See, the attached screen shot below.



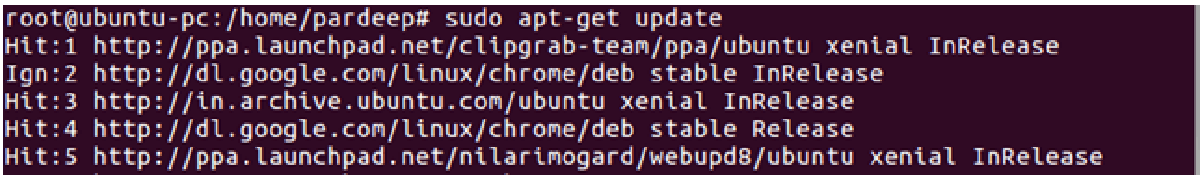
```
root@ubuntu-pc:/home/pardeep# echo "<REPO>" | sudo tee /etc/apt/sources.list.d/d
ocker.list
<REPO>
```

6. Open the file /etc/apt/sources.list.d/docker.list and paste the following line into the file.
   1. deb https://apt.dockerproject.org/repo ubuntu-xenial main



```
root@ubuntu-pc: /home/pardeep
deb https://apt.dockerproject.org/repo ubuntu-xenial main
~
~
~
~
```

7. Now again update your apt packages index.
    1. $ sudo atp-get update

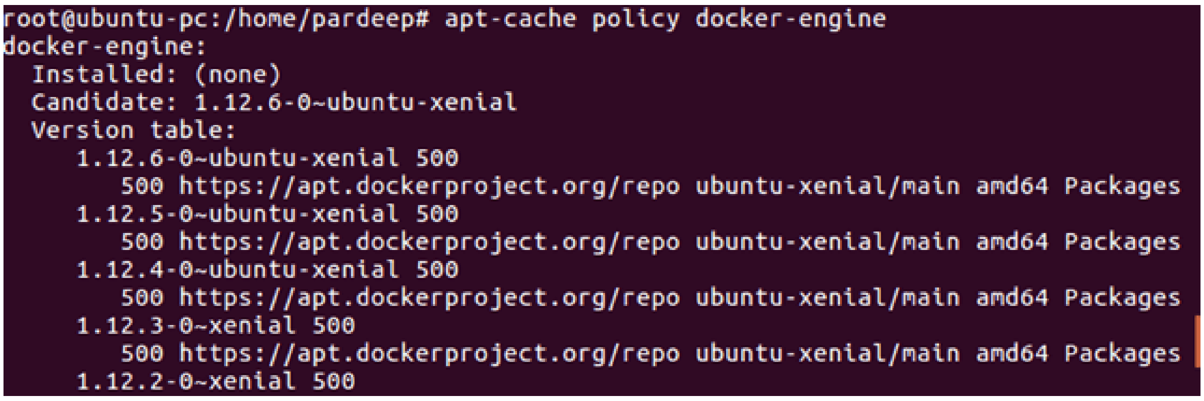


```
root@ubuntu-pc:/home/pardeep# sudo apt-get update
Hit:1 http://ppa.launchpad.net/clipgrab-team/ppa/ubuntu xenial InRelease
Ign:2 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu xenial InRelease
Hit:4 http://dl.google.com/linux/chrome/deb stable Release
Hit:5 http://ppa.launchpad.net/nilarimogard/webupd8/ubuntu xenial InRelease
```

See, the attached screen shot below.

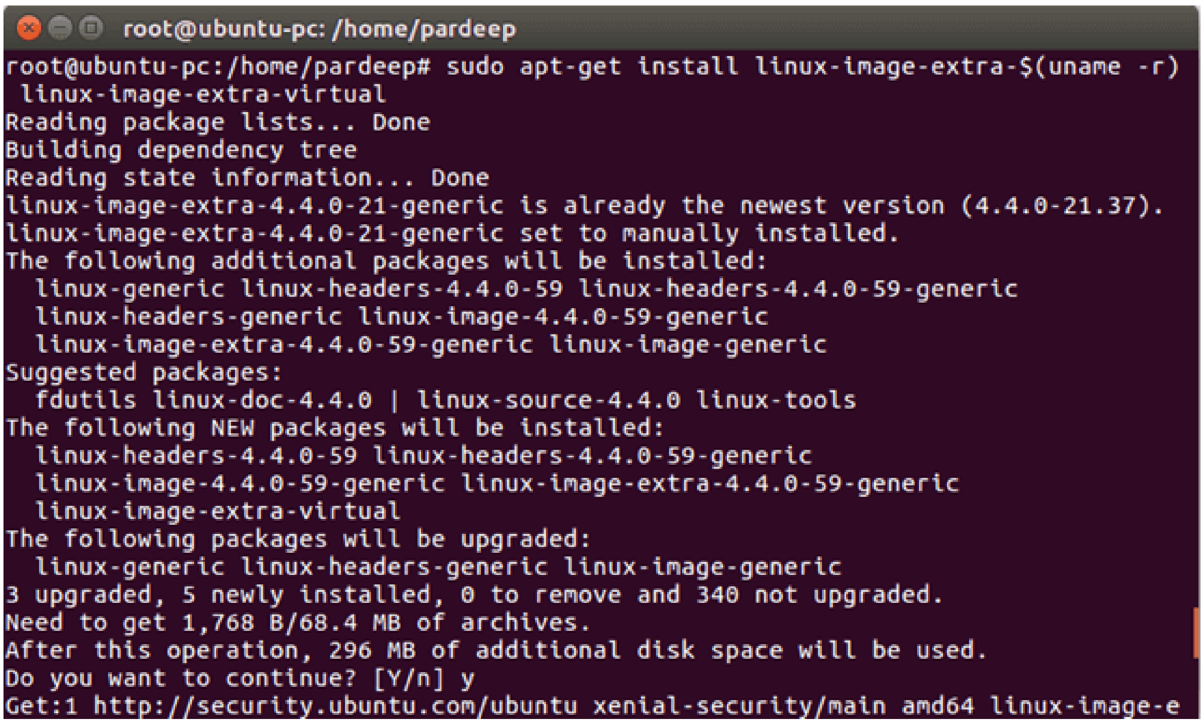
8. Verify that APT is pulling from the right repository.
    1. $ apt-cache policy docker-engine

See, the attached screen shot below.



```
root@ubuntu-pc:/home/pardeep# apt-cache policy docker-engine
docker-engine:
  Installed: (none)
  Candidate: 1.12.6-0~ubuntu-xenial
  Version table:
     1.12.6-0~ubuntu-xenial 500
        500 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 Packages
     1.12.5-0~ubuntu-xenial 500
        500 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 Packages
     1.12.4-0~ubuntu-xenial 500
        500 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 Packages
     1.12.3-0~xenial 500
        500 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 Packages
     1.12.2-0~xenial 500
```

9. Install the recommended packages.
    1. $ sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-virtual

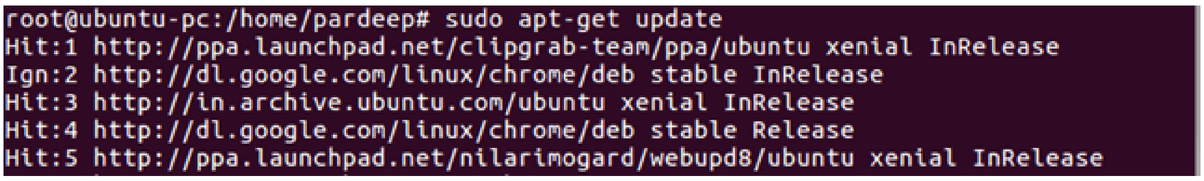


```
root@ubuntu-pc: /home/pardeep
root@ubuntu-pc:/home/pardeep# sudo apt-get install linux-image-extra-$(uname -r)
 linux-image-extra-virtual
Reading package lists... Done
Building dependency tree
Reading state information... Done
linux-image-extra-4.4.0-21-generic is already the newest version (4.4.0-21.37).
linux-image-extra-4.4.0-21-generic set to manually installed.
The following additional packages will be installed:
  linux-generic linux-headers-4.4.0-59 linux-headers-4.4.0-59-generic
  linux-headers-generic linux-image-4.4.0-59-generic
  linux-image-extra-4.4.0-59-generic linux-image-generic
Suggested packages:
  fdutils linux-doc-4.4.0 | linux-source-4.4.0 linux-tools
The following NEW packages will be installed:
  linux-headers-4.4.0-59 linux-headers-4.4.0-59-generic
  linux-image-4.4.0-59-generic linux-image-extra-4.4.0-59-generic
  linux-image-extra-virtual
The following packages will be upgraded:
  linux-generic linux-headers-generic linux-image-generic
3 upgraded, 5 newly installed, 0 to remove and 340 not upgraded.
Need to get 1,768 B/68.4 MB of archives.
After this operation, 296 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://security.ubuntu.com/ubuntu xenial-security/main amd64 linux-image-e
```

Install the latest Docker version.

1. update your apt packages index.
    1. $ sudo apt-get update

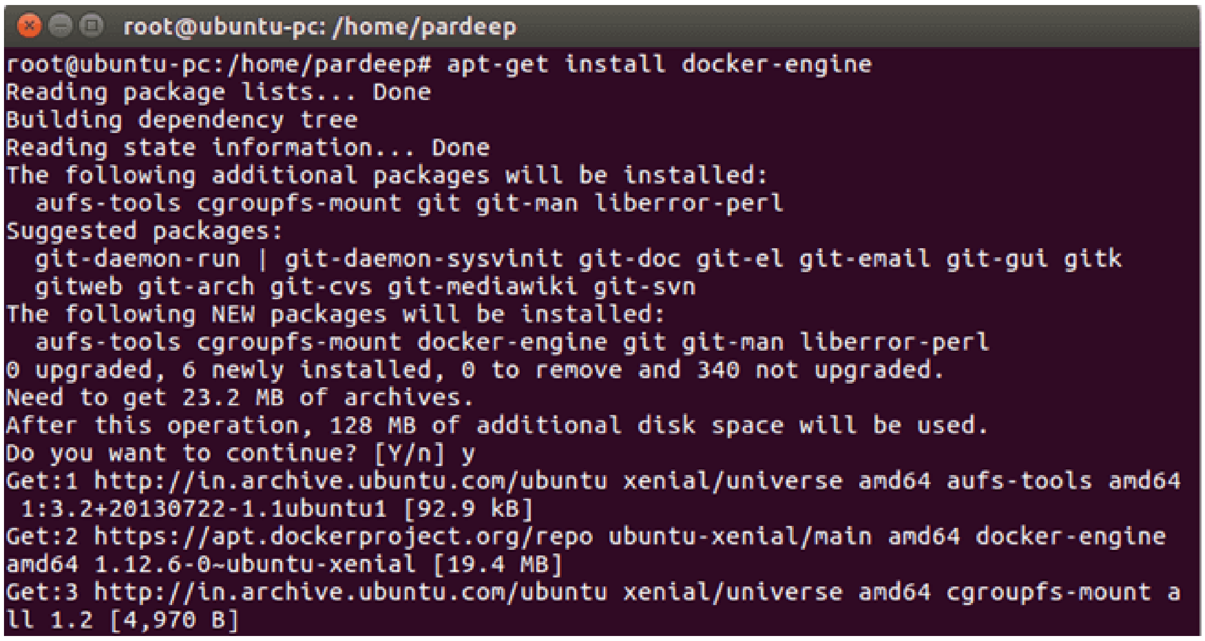
    See, the attached screen shot below.



```
root@ubuntu-pc:/home/pardeep# sudo apt-get update
Hit:1 http://ppa.launchpad.net/clipgrab-team/ppa/ubuntu xenial InRelease
Ign:2 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu xenial InRelease
Hit:4 http://dl.google.com/linux/chrome/deb stable Release
Hit:5 http://ppa.launchpad.net/nilarimogard/webupd8/ubuntu xenial InRelease
```

2. Install docker-engine.
    1. $ sudo apt-get install docker-engine

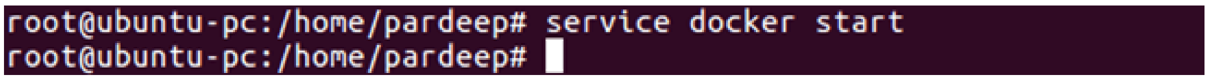
    See, the attached screen shot below.



```
root@ubuntu-pc:/home/pardeep# apt-get install docker-engine
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  aufs-tools cgroupfs-mount git git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  aufs-tools cgroupfs-mount docker-engine git git-man liberror-perl
0 upgraded, 6 newly installed, 0 to remove and 340 not upgraded.
Need to get 23.2 MB of archives.
After this operation, 128 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 aufs-tools amd64
 1:3.2+20130722-1.1ubuntu1 [92.9 kB]
Get:2 https://apt.dockerproject.org/repo ubuntu-xenial/main amd64 docker-engine
amd64 1.12.6-0~ubuntu-xenial [19.4 MB]
Get:3 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 cgroupfs-mount a
ll 1.2 [4,970 B]
```

3. Start the docker daemon.
    1. $ sudo service docker start

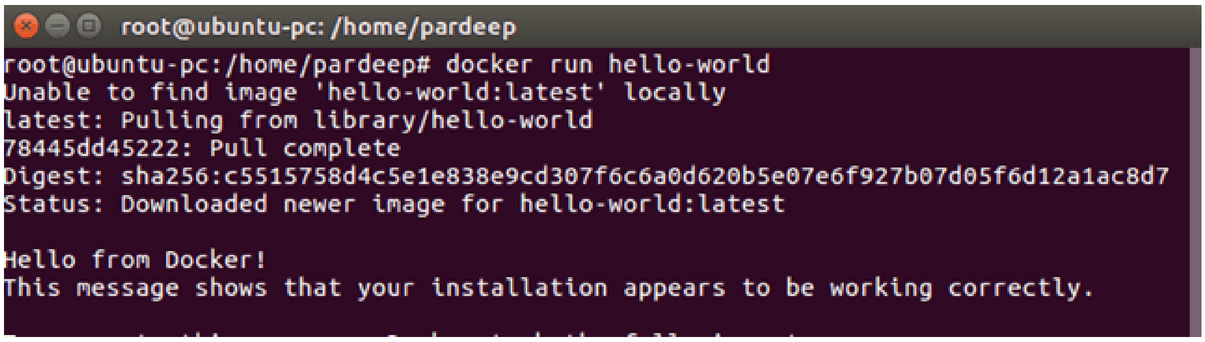
See, the attached screen shot below.



```
root@ubuntu-pc:/home/pardeep# service docker start
root@ubuntu-pc:/home/pardeep# 
```

4. Verify that docker is installed correctly by running the hello-world image.
    1. $ sudo docker run hello-world

See, the attached screen shot below.



```
root@ubuntu-pc:/home/pardeep# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

This above command downloads a test image and runs it in a container. When the container runs, it prints a message and exits.

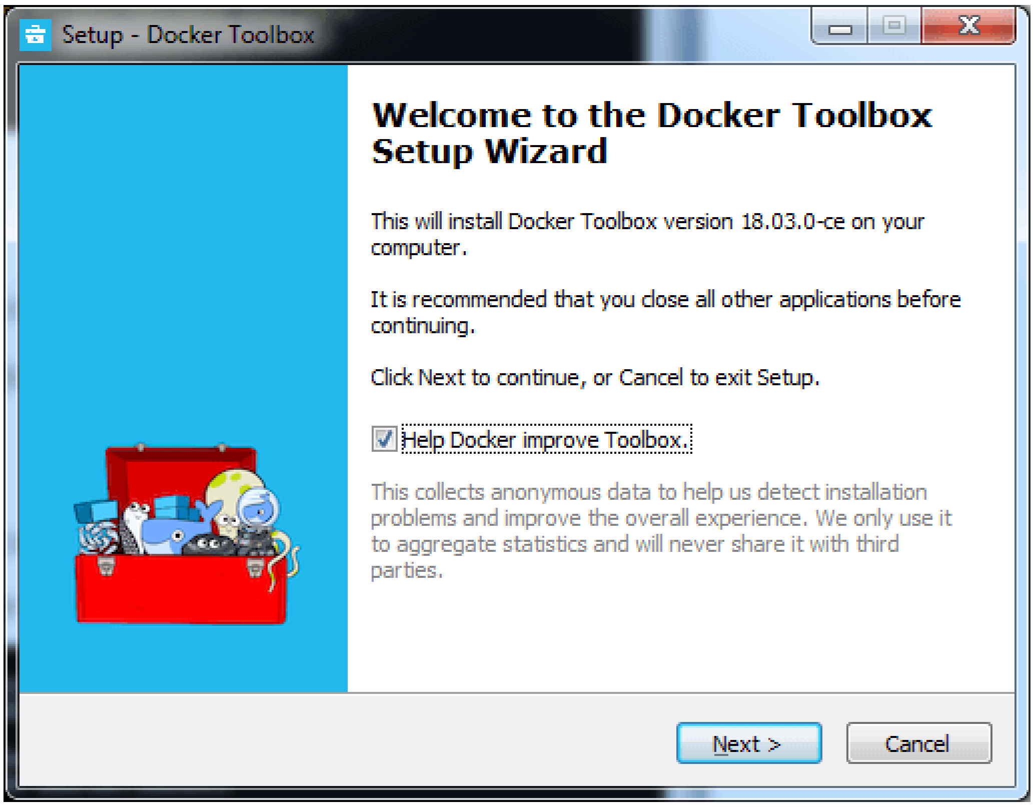
How to install docker on Windows

We can install docker on any operating system like **Windows, Linux,** or **Mac**. Here, we are going to install docker-engine on **Windows**. The main advantage of using Docker on Windows is that it provides an ability to run natively on Windows without any kind of virtualization. To install docker on windows, we need to download and install the **Docker Toolbox**.

Follow the below steps to install docker on windows -

Step 1: Click on the below link to download DockerToolbox.exe. https://download.docker.com/win/stable/DockerToolbox.exe

Step 2: Once the **DockerToolbox.exe** file is downloaded, **double click** on that file. The following window appears on the screen, in which click on the **Next**.



Setup - Docker Toolbox

Welcome to the Docker Toolbox Setup Wizard

This will install Docker Toolbox version 18.03.0-ce on your computer.

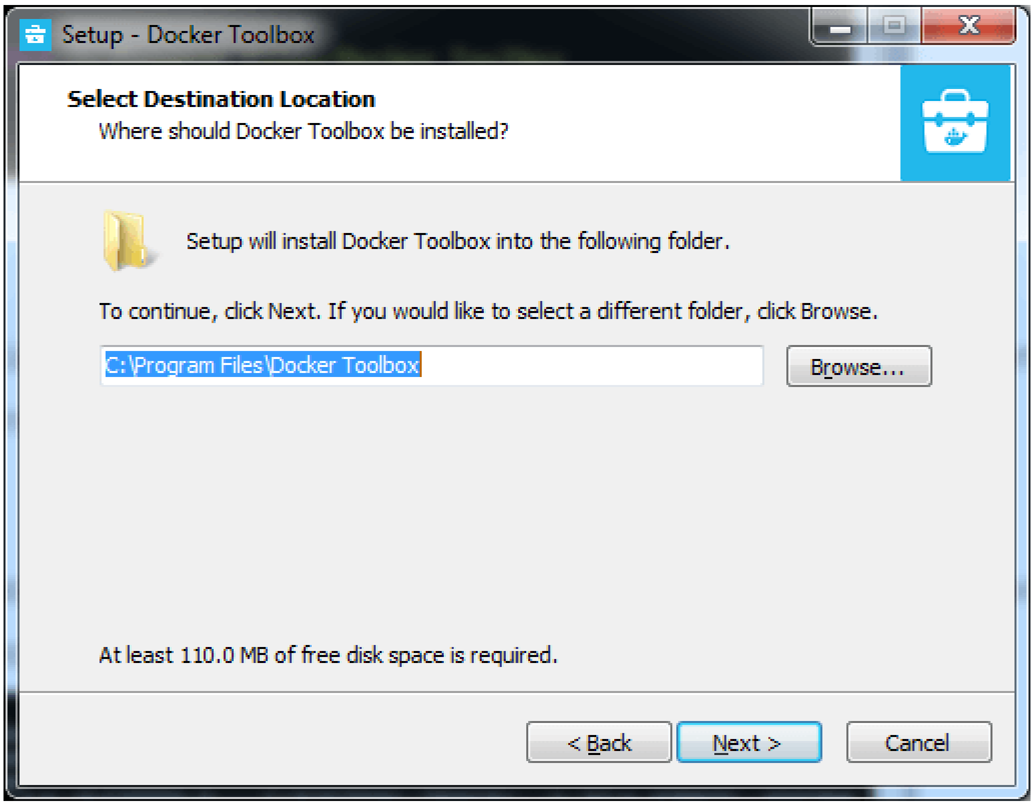
It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

[✓] Help Docker improve Toolbox.

This collects anonymous data to help us detect installation problems and improve the overall experience. We only use it to aggregate statistics and will never share it with third parties.

Step 3: Browse the location where you want to install the Docker Toolbox and click on the Next.


Setup - Docker Toolbox

Select Destination Location

Where should Docker Toolbox be installed?

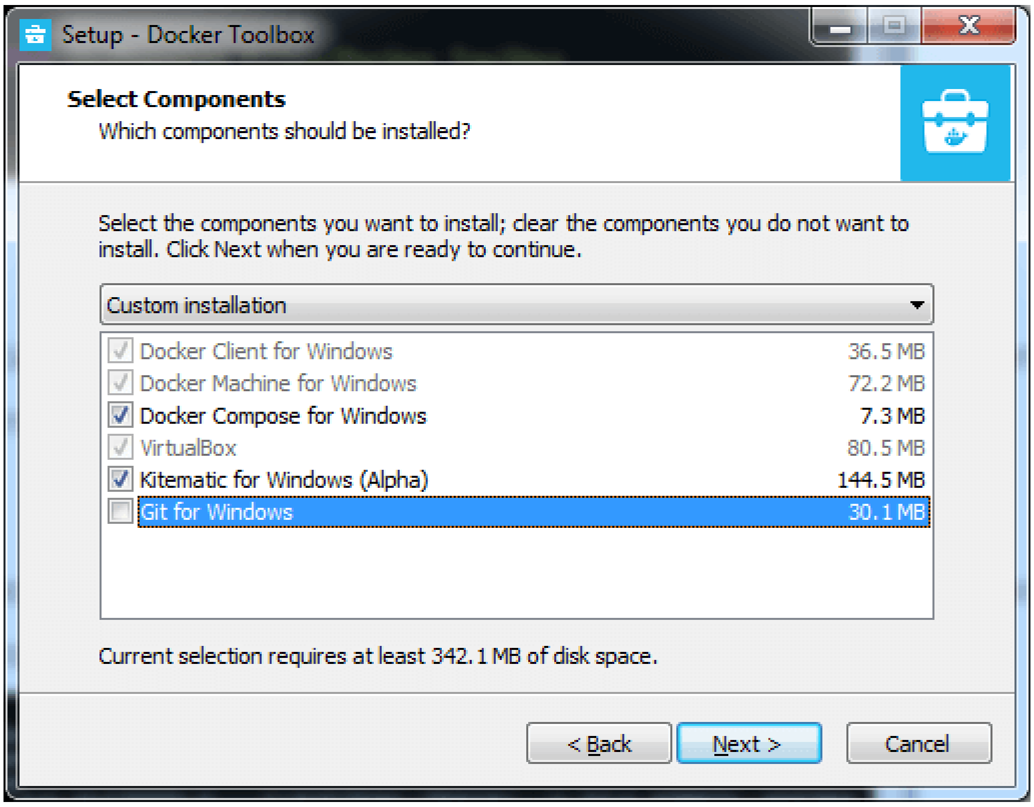
Setup will install Docker Toolbox into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Docker Toolbox

At least 110.0 MB of free disk space is required.

Step 4: Select the components according to your requirement and click on the **Next**.


Setup - Docker Toolbox

Select Components

Which components should be installed?

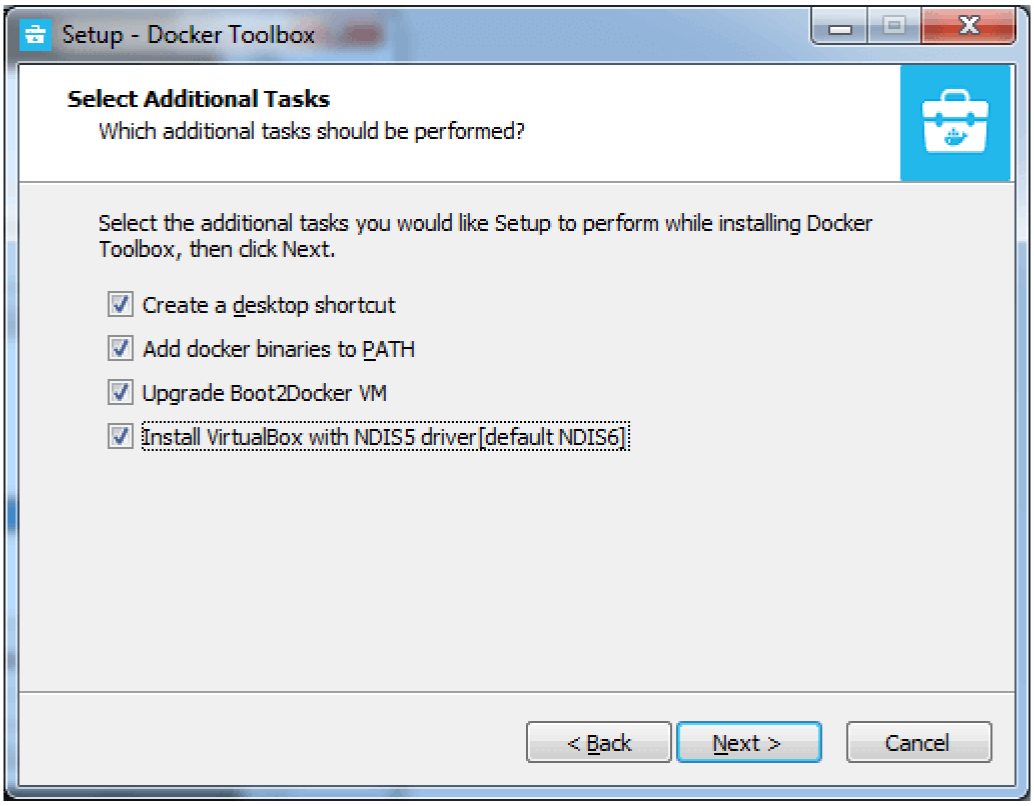
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Custom installation

✓	Docker Client for Windows	36.5 MB
✓	Docker Machine for Windows	72.2 MB
✓	Docker Compose for Windows	7.3 MB
✓	VirtualBox	80.5 MB
✓	Kitematic for Windows (Alpha)	144.5 MB
☐	Git for Windows	30.1 MB

Current selection requires at least 342.1 MB of disk space.

Step 5: Select Additional Tasks and click on the **Next**.



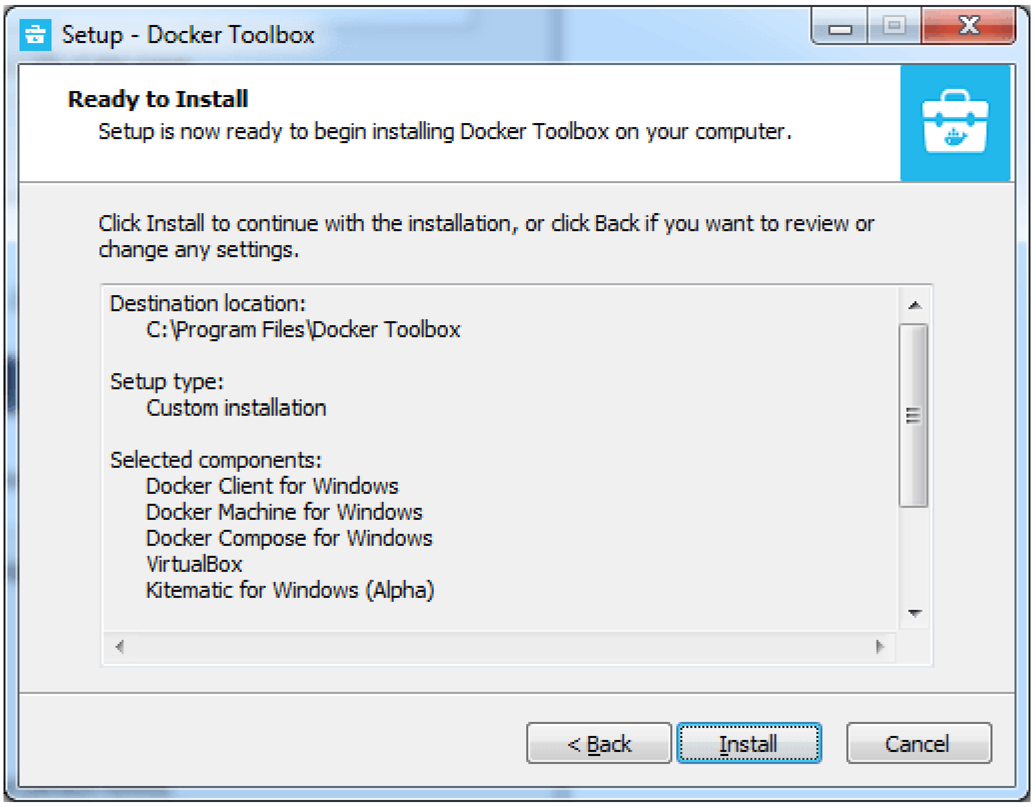
Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Docker Toolbox, then click Next.

- [✓] Create a desktop shortcut
- [✓] Add docker binaries to PATH
- [✓] Upgrade Boot2Docker VM
- [✓] Install VirtualBox with NDIS5 driver[default NDIS6]

Step 6: The Docker Toolbox is ready to install. Click on **Install**.



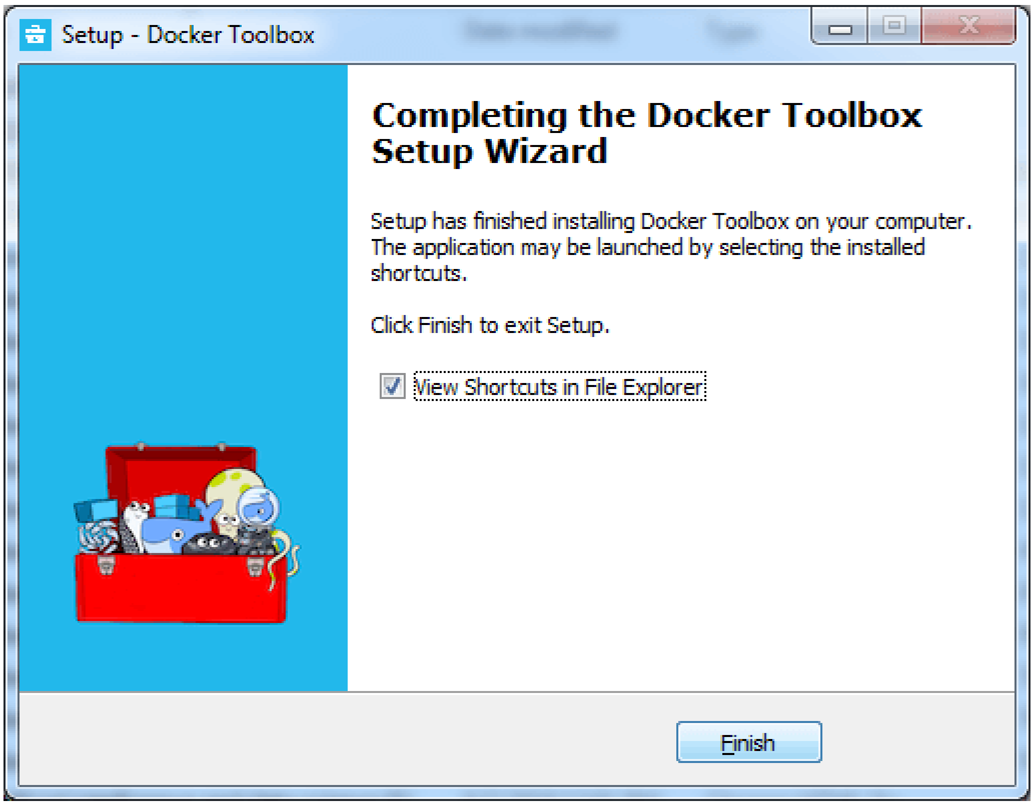
Ready to Install

Setup is now ready to begin installing Docker Toolbox on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Destination location:
    C:\Program Files\Docker Toolbox

Setup type:
    Custom installation

Selected components:
    Docker Client for Windows
    Docker Machine for Windows
    Docker Compose for Windows
    VirtualBox
    Kitematic for Windows (Alpha)
```

Step 7: Once the installation is completed, the following Wizard appears on the screen, in which click on the **Finish**.


Setup - Docker Toolbox

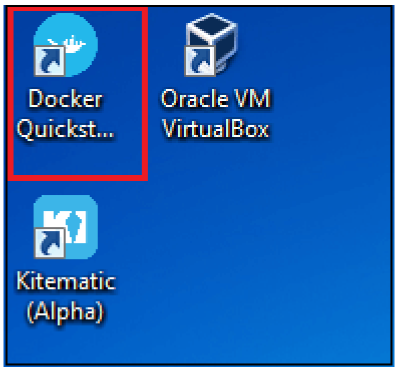
Completing the Docker Toolbox Setup Wizard

Setup has finished installing Docker Toolbox on your computer. The application may be launched by selecting the installed shortcuts.

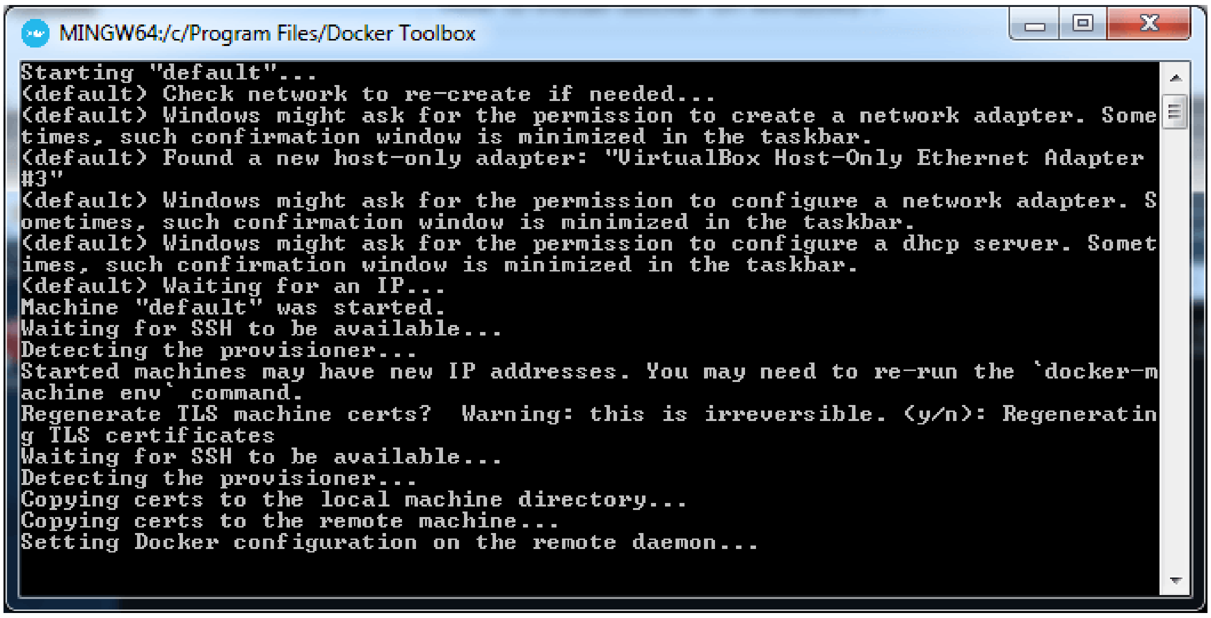
Click Finish to exit Setup.

☑ View Shortcuts in File Explorer

Step 8: After the successful installation, three icons will appear on the screen that are: **Docker QuickStart Terminal, Kinematic (Alpha),** and **OracleVM VirtualBox**. **Double click** on the Docker QuickStart Terminal.



Step 9: A Docker QuickStart Terminal window appears on the screen.

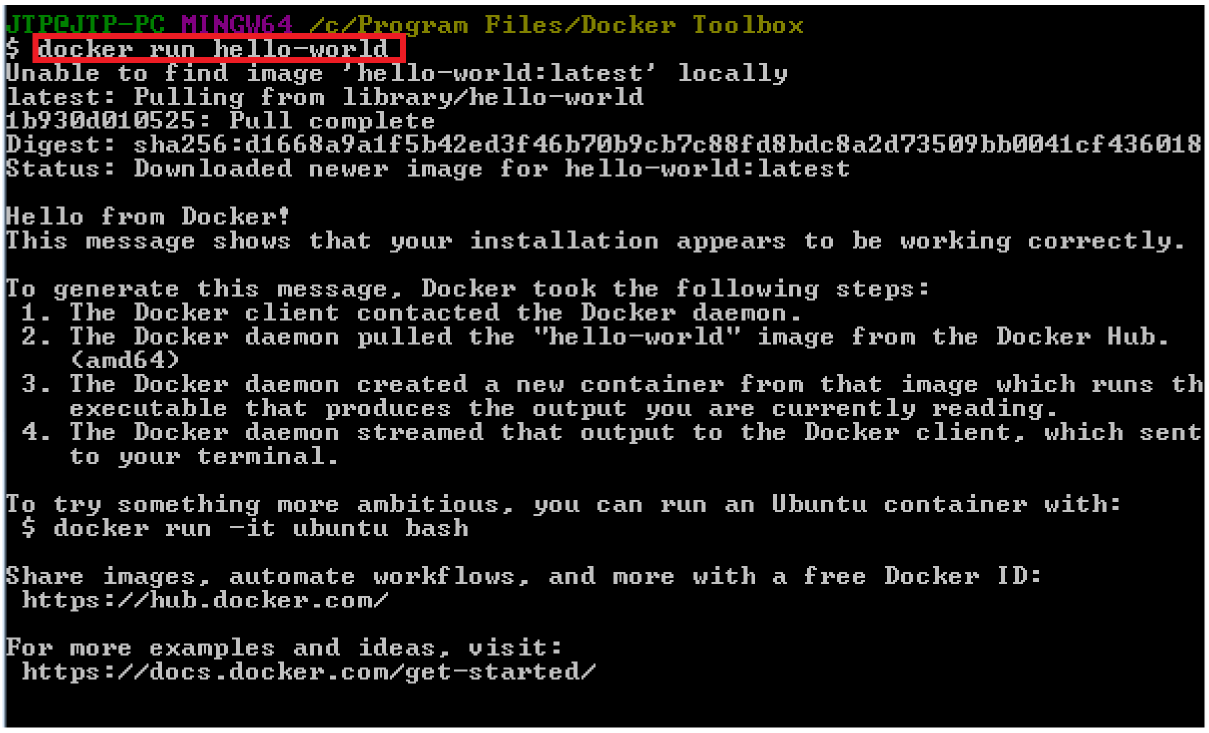


```
Starting "default"...
(default) Check network to re-create if needed...
(default) Windows might ask for the permission to create a network adapter. Some
times, such confirmation window is minimized in the taskbar.
(default) Found a new host-only adapter: "VirtualBox Host-Only Ethernet Adapter
#3"
(default) Windows might ask for the permission to configure a network adapter. S
ometimes, such confirmation window is minimized in the taskbar.
(default) Windows might ask for the permission to configure a dhcp server. Somet
imes, such confirmation window is minimized in the taskbar.
(default) Waiting for an IP...
Machine "default" was started.
Waiting for SSH to be available...
Detecting the provisioner...
Started machines may have new IP addresses. You may need to re-run the `docker-m
achine env` command.
Regenerate TLS machine certs?  Warning: this is irreversible. (y/n): Regeneratin
g TLS certificates
Waiting for SSH to be available...
Detecting the provisioner...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
```

To verify that the docker is successfully installed, type the below command and press enter key.

1. docker run hello-world

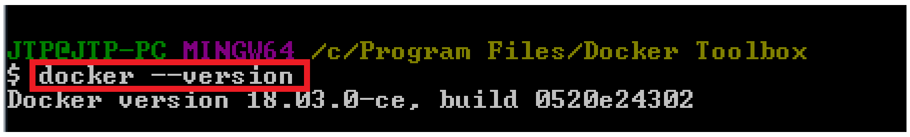
The following output will be visible on the screen, otherwise not.



```
JTP@JTP-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:d1668a9a1f5b42ed3f46b70b9cb7c88fd8bdc8a2d73509bb0041cf436018
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 client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs th
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

You can check the Docker version using the following command.

1. docker -version



```
JTP@JTP-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker --version
Docker version 18.03.0-ce, build 0520e24302
```