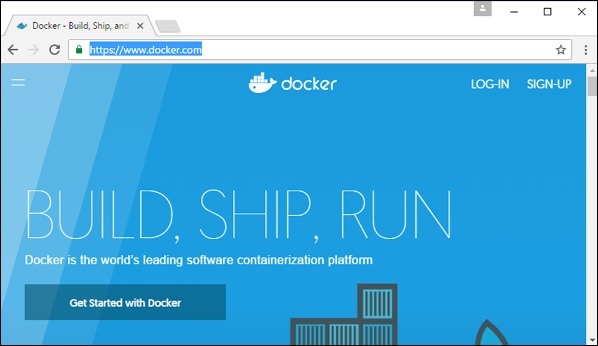
Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.



## Features of Docker

Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.

With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.

You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.

Since Docker containers are pretty lightweight, they are very easily scalable.

## Components of Docker

Docker has the following components

**Docker for Mac** − It allows one to run Docker containers on the Mac OS.

**Docker for Linux** − It allows one to run Docker containers on the Linux OS.

**Docker for Windows** − It allows one to run Docker containers on the Windows OS.

**Docker Engine** − It is used for building Docker images and creating Docker containers.

**Docker Hub** − This is the registry which is used to host various Docker images.

**Docker Compose** − This is used to define applications using multiple Docker containers.

# 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:

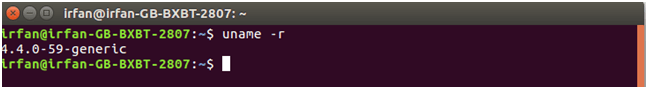
Docker need two important installation requirements:

* 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



## Update apt sources

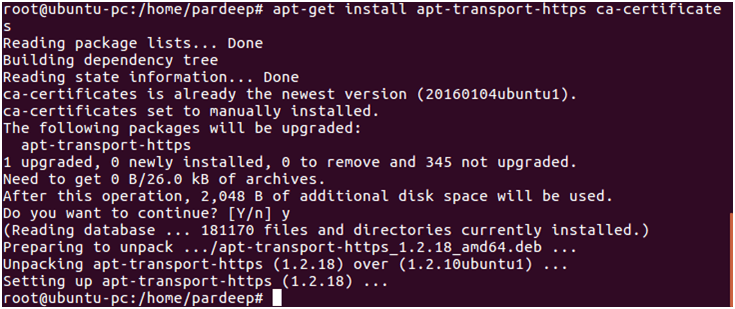
Follow following instructions to update apt sources.

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.

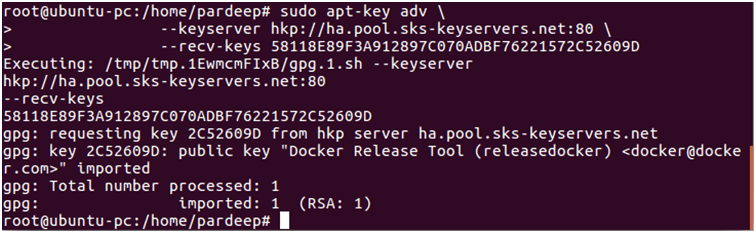


1. Add the new GPG key. Following command downloads the key.

**Command:**

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

Screen shot is given below.

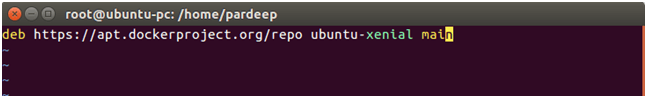


1. Run the following command, it will substitute the entry for your operating system for the file.
   1. $ echo "<REPO>" | sudo tee /etc/apt/sources.list.d/docker.list

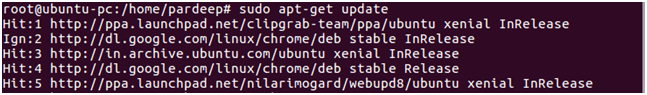
See, the attached screen shot below.



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



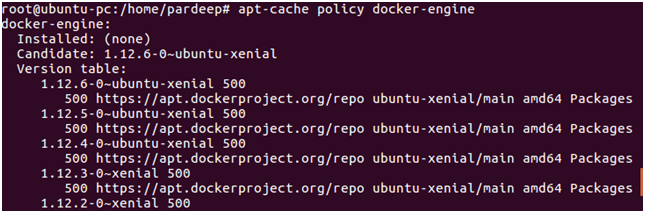
1. Now again update your apt packages index.
   1. $ sudo atp-get update



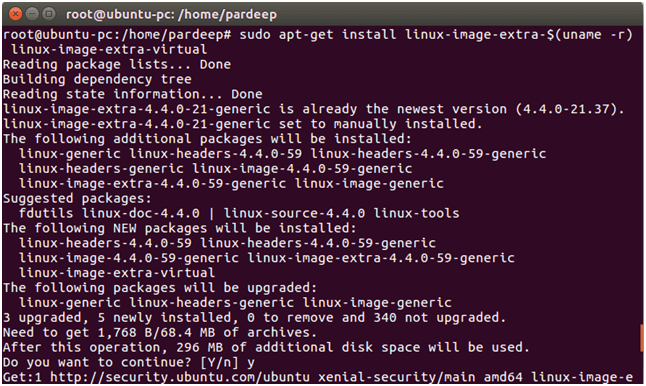
See, the attached screen shot below.

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

See, the attached screen shot below.



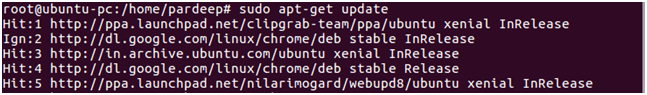
1. Install the recommended packages.
   1. $ sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-virtual



## Install the latest Docker version.

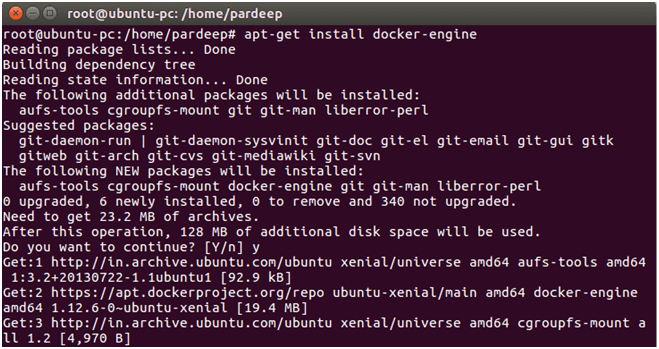
1. update your apt packages index.
   1. $ sudo apt-get update

See, the attached screen shot below.



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

See, the attached screen shot below.



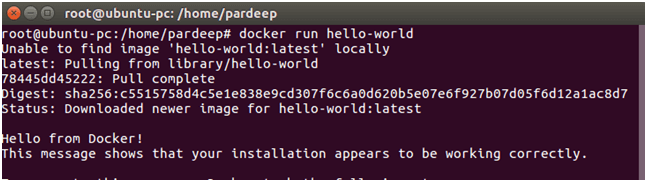
1. Start the docker daemon.
   1. $ sudo service docker start

See, the attached screen shot below.

IMG_266

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

See, the attached screen shot below.



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

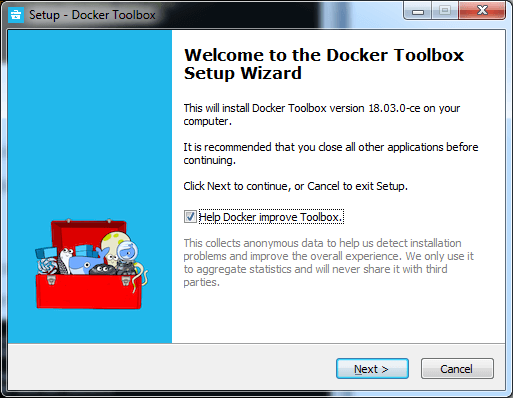
# How to install docker on Windows

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**.

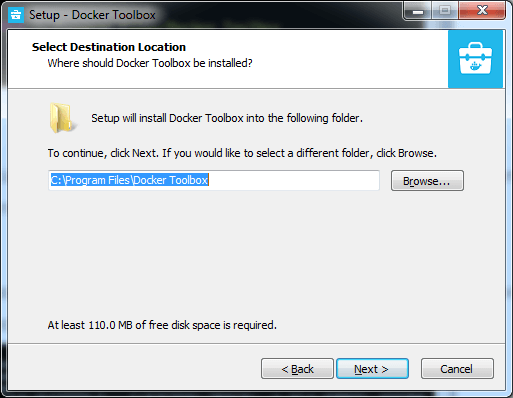
Follow the below steps to install docker on windows -

**Step 1:** Click on the below link to download DockerToolbox.exe. [https://download.docker.com/win/stable/DockerToolbox.exe](https://download.docker.com/win/stable/DockerToolbox.exe" \t "https://www.javatpoint.com/_blank)

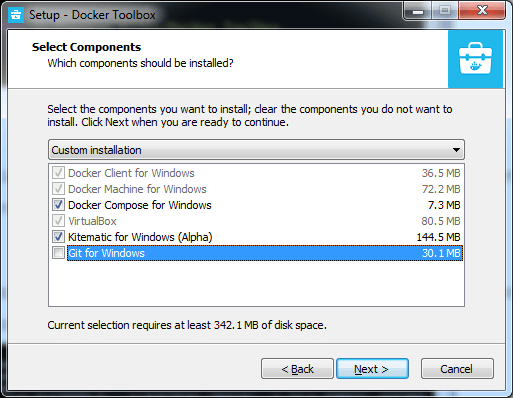
**Step 2:** Once the **DockerToolbox.exe** file is downloaded, **double click** on that file. The following window appears on the screen, in which click on the **Next**.



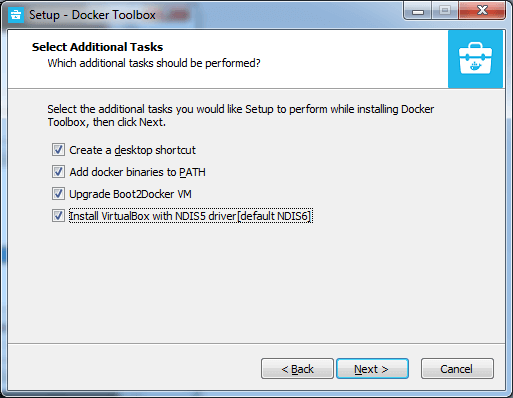
**Step 3: Browse the location** where you want to install the Docker Toolbox and click on the Next.



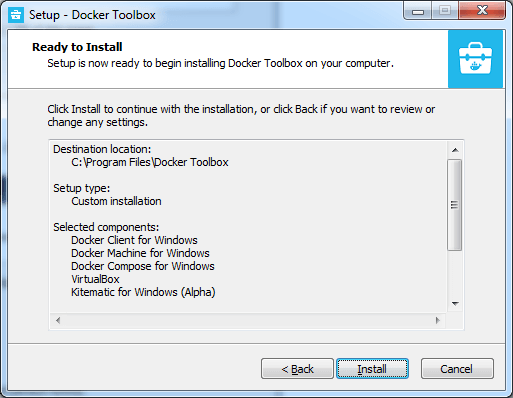
**Step 4: Select the components** according to your requirement and click on the **Next**.



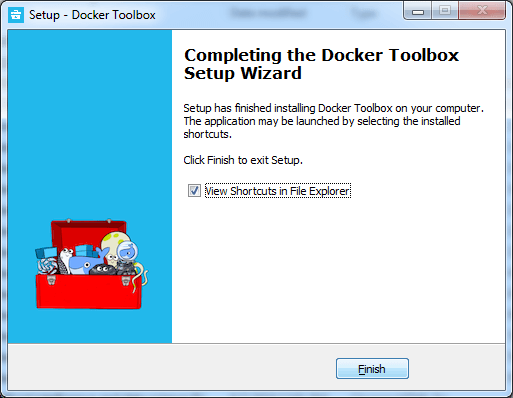
**Step 5: Select Additional Tasks** and click on the **Next**.



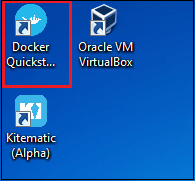
**Step 6:** The Docker Toolbox is ready to install. Click on **Install**.



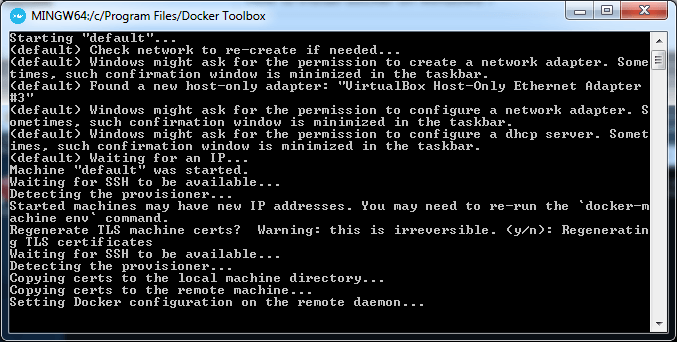
**Step 7:** Once the installation is completed, the following Wizard appears on the screen, in which click on the **Finish**.



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



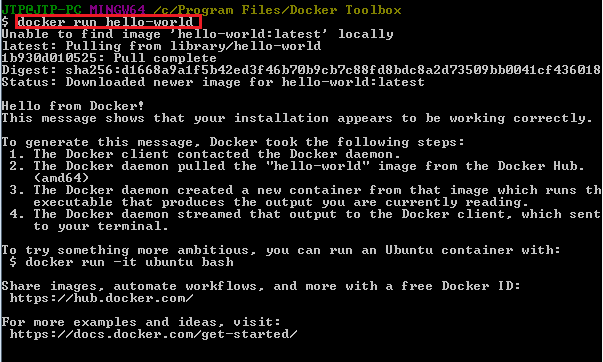
**Step 9:** A Docker Quickstart Terminal window appears on the screen.



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.



You can check the Docker version using the following command.

1. docker -version

IMG_265

# Docker Container and Image

Docker container is a running instance of an image. You can use Command Line Interface (CLI) commands to run, start, stop, move, or delete a container. You can also provide configuration for the network and environment variables. Docker container is an isolated and secure application platform, but it can share and access to resources running in a different host or container.

An image is a read-only template with instructions for creating a Docker container. A docker image is described in text file called a **Dockerfile**, which has a simple, well-defined syntax. An image does not have states and never changes. Docker Engine provides the core Docker technology that enables images and containers.

You can understand container and image with the help of the following command.

1. $ docker run hello-world

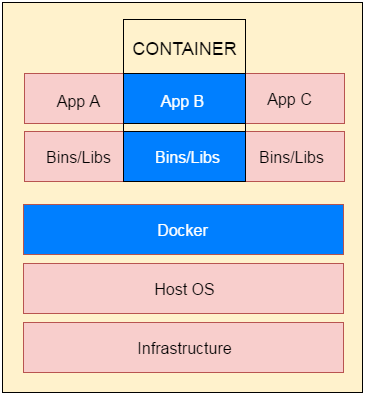
The above command **docker run hello-world** has three parts.

1) **docker:** It is docker engine and used to run docker program. It tells to the operating system that you are running docker program.

2) **run:** This subcommand is used to create and run a docker container.

3) **hello-world:** It is a name of an image. You need to specify the name of an image which is to load into the container.

### Docker Container



**Fig**: docker-container

# Docker Dockerfile

A Dockerfile is a text document that contains commands that are used to assemble an image. We can use any command that call on the command line. Docker builds images automatically by reading the instructions from the Dockerfile.

The docker build command is used to build an image from the Dockerfile. You can use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

1. $ docker build -f /path/to/a/Dockerfile .

## Dockerfile Instructions

The instructions are not case-sensitive but you must follow conventions which recommend to use uppercase.

Docker runs instructions of Dockerfile in top to bottom order. The first instruction must be **FROM** in order to specify the Base Image.

A statement begin with # treated as a comment. You can use RUN, CMD, FROM, EXPOSE, ENV etc instructions in your Dockerfile.

Here, we are listing some commonly used instructions.

### FROM

This instruction is used to set the Base Image for the subsequent instructions. A valid Dockerfile must have FROM as its first instruction.

Ex.

1. FROM ubuntu

### LABEL

We can add labels to an image to organize images of our project. We need to use LABEL instruction to set label for the image.

Ex.

1. LABEL vendorl = "JavaTpoint"

### RUN

This instruction is used to execute any command of the current image.

Ex.

1. RUN /bin/bash -c 'source $HOME/.bashrc; echo $HOME'

### CMD

This is used to execute application by the image. We should use CMD always in the following form

1. CMD ["executable", "param1", "param2"?]

This is preferred way to use CMD. There can be only one CMD in a Dockerfile. If we use more than one CMD, only last one will execute.

### COPY

This instruction is used to copy new files or directories from source to the filesystem of the container at the destination.

Ex.

1. COPY abc/ /xyz

**Rules**

* The source path must be inside the context of the build. We cannot COPY ../something /something because the first step of a docker build is to send the context directory (and subdirectories) to the docker daemon.
* If source is a directory, the entire contents of the directory are copied including filesystem metadata.

### WORKDIR

The WORKDIR is used to set the working directory for any RUN, CMD and COPY instruction that follows it in the Dockerfile. If work directory does not exist, it will be created by default.

We can use WORKDIR multiple times in a Dockerfile.

Ex.

1. WORKDIR /var/www/html