Docker Tutorial – Introduction To Docker & Containerization

Last updated on May 22,2019 *83.3K Views*

Vineet Chaturvedi

I hope you haven't missed the series of earlier DevOps Tutorial blogs. Go through the **DevOps Tutorial blog series** here. The uncontainable trend of Docker container is growing & organizations are looking for professionals possessing ***Docker certification***. Now this time, we will take you through an Introduction To Docker.

Docker Tutorial

This Docker Tutorial blog will give you the conceptual & practical exposure to Docker – A new age containerization technology.

In this blog, we will focus on the below topics:

- What is Virtualization?
- What is Containerization
- Advantages of Containerization over Virtualization
- Introduction to Docker
- Benefits of Docker
- Virtualization vs Containerization
- Docker Installation
- Dockerfile, Docker Image & Docker Container
- What is Docker Hub?
- Docker Architecture
- Docker Compose

Docker is gaining popularity and its usage is spreading like wildfire. The reason for Docker's growing popularity is the extent to which it can be used in an IT organization. Very few tools out there have the functionality to find itself useful to both developers and as well as system administrators. Docker is one such tool that truly lives up to its promise of **Build**, **Ship** and **Run**.

In simple words, Docker is a software containerization platform, meaning you can build your application, package them along with their dependencies into a container and then these containers can be easily shipped to run on other machines.

For example: Lets consider a linux based application which has been written both in Ruby and Python. This application requires a specific version of linux, Ruby and Python. In order to avoid any version conflicts on user's end, a linux docker container can be created with the required versions of Ruby and Python installed along with the application. Now the end users can use the application easily by running this container without worrying about the dependencies or any version conflicts.

These containers uses Containerization which can be considered as an evolved version of Virtualization. The same task can also be achieved using Virtual Machines, however it is not very efficient.

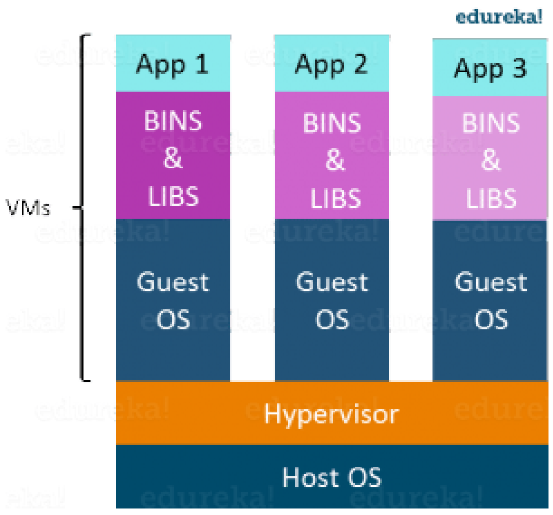
I generally receive a question at this point, i.e. what is the difference between Virtualization and Containerization? These two terms are very similar to each other. So, let me first tell you What is Virtualization?

What is Virtualization?

Virtualization is the technique of importing a Guest operating system on top of a Host operating system. This technique was a revelation at the beginning because it allowed developers to run multiple operating systems in different virtual machines all running on the same host. This eliminated the need for extra hardware resource. The advantages of Virtual Machines or Virtualization are:

- Multiple operating systems can run on the same machine
- Maintenance and Recovery were easy in case of failure conditions
- Total cost of ownership was also less due to the reduced need for infrastructure

In the diagram on the right, you can see there is a host operating system on which there are 3 guest operating systems running which is nothing but the virtual machines.



As you know nothing is perfect, Virtualization also has some shortcomings. Running multiple Virtual Machines in the same host operating system leads to performance degradation. This is because of the guest OS running on top of the host OS, which will have its own kernel and set of libraries and dependencies. This takes up a large chunk of system resources, i.e. hard disk, processor and especially RAM.

Another problem with Virtual Machines which uses virtualization is that it takes almost a minute to boot-up. This is very critical in case of real-time applications.

Following are the disadvantages of Virtualization:

- Running multiple Virtual Machines leads to unstable performance
- Hypervisors are not as efficient as the host operating system
- Boot up process is long and takes time

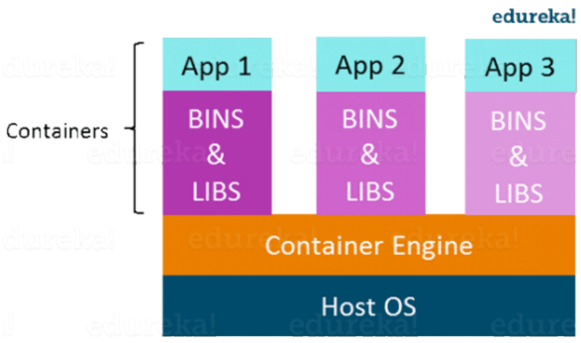
These drawbacks led to the emergence of a new technique called Containerization. Now let me tell you about Containerization.

What is Containerization?

Containerization is the technique of bringing virtualization to the operating system level. While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the operating system. Do note that Containerization is also a type of Virtualization. Containerization is however more efficient because there is no guest OS here and utilizes a host's operating system, share relevant libraries & resources as and when needed unlike virtual machines. Application specific binaries and libraries of containers run on the host kernel, which makes processing and execution very fast. Even booting-up a container takes only a fraction of a second. Because all the containers share, host operating system and holds only the application related binaries & libraries. They are lightweight and faster than Virtual Machines.

Advantages of Containerization over Virtualization:

- Containers on the same OS kernel are lighter and smaller
- Better resource utilization compared to VMs
- Boot-up process is short and takes few seconds

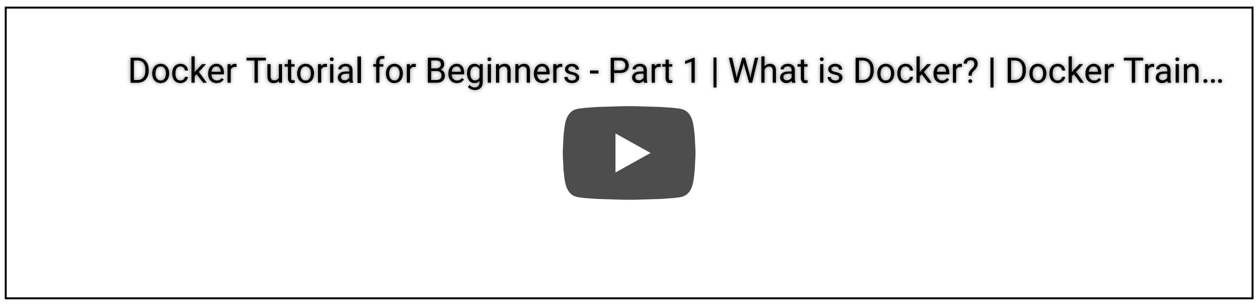


In the diagram on the right, you can see that there is a host operating system which is shared by all the containers. Containers only contain application specific libraries which are separate for each container and they are faster and do not waste any resources.

All these containers are handled by the containerization layer which is not native to the host operating system. Hence a software is needed, which can enable you to create & run containers on your host operating system.

Check out this Docker tutorial video for a deep understanding of Docker.

Docker Tutorial for Beginners | What is Docker? | DevOps Tools | Edureka

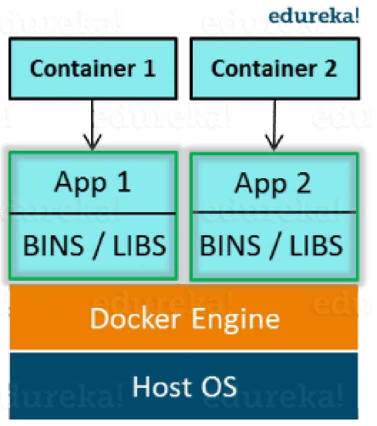


Docker Tutorial for Beginners - Part 1 | What is Docker? | Docker Train...

Now, let me take you through the introduction to Docker.

Docker Tutorial – Introduction To Docker

Docker is a containerization platform that packages your application and all its dependencies together in the form of Containers to ensure that your application works seamlessly in any environment.



As you can see in the diagram on the right, each application will run on a separate container and will have its own set of libraries and dependencies. This also ensures that there is process level isolation, meaning each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.

As a developer, I can build a container which has different applications installed on it and give it to my QA team who will only need to run the container to replicate the developer environment.

Benefits of Docker

Now, the QA team need not install all the dependent software and applications to test the code and this helps them save lots of time and energy. This also ensures that the working environment is consistent across all the individuals involved in the process, starting from development to deployment. The number of systems can be scaled up easily and the code can be deployed on them effortlessly.

Virtualization vs Containerization

Virtualization and Containerization both let you run multiple operating systems inside a host machine.

Virtualization deals with creating many operating systems in a single host machine. Containerization on the other hand will create multiple containers for every type of application as required.

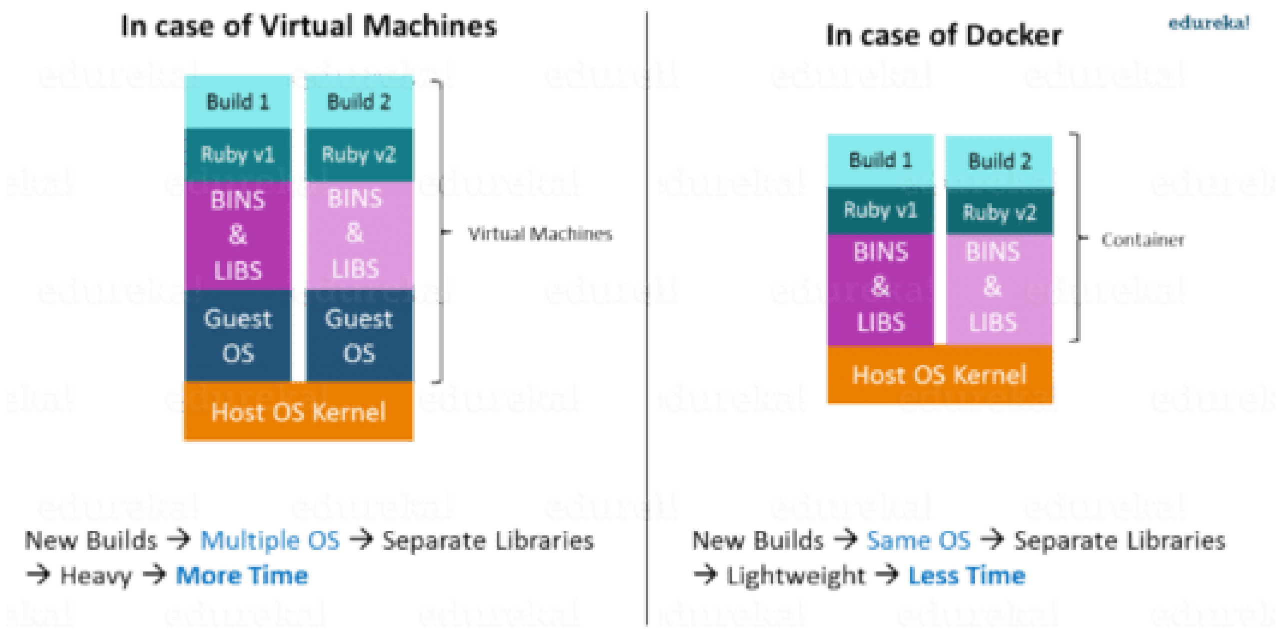


Figure: *What Is Big Data Analytics – Virtualization versus Containerization*

As we can see from the image, the major difference is that there are multiple Guest Operating Systems in Virtualization which are absent in Containerization. The best part of Containerization is that it is very lightweight as compared to the heavy virtualization.

Now, let us install Docker.

Install Docker:

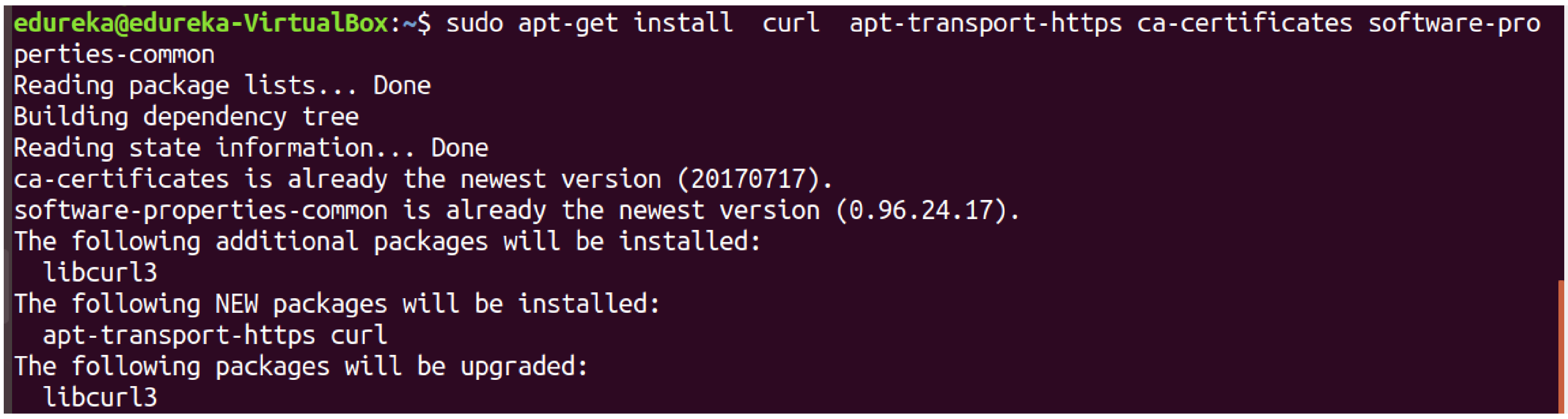
I will be installing Docker in my Ubuntu 17.10 machine. Following are the steps to install Docker:

1. Install required Packages
2. Setup Docker repository
3. Install Docker On Ubuntu

1. Install Required Packages:

There are certain packages you require in your system for installing Docker. Execute the below command to install those packages.

```
sudo apt-get install  curl  apt-transport-https ca-certificates software-properties-common
```

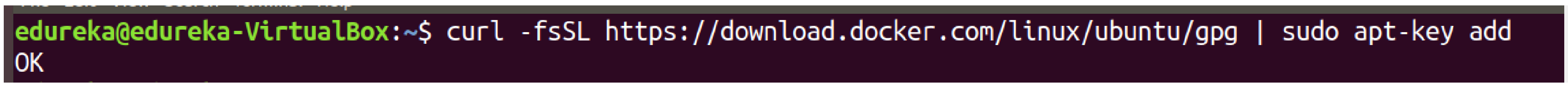


```
edureka@edureka-VirtualBox:~$ sudo apt-get install  curl  apt-transport-https ca-certificates software-pro
perties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20170717).
software-properties-common is already the newest version (0.96.24.17).
The following additional packages will be installed:
  libcurl3
The following NEW packages will be installed:
  apt-transport-https curl
The following packages will be upgraded:
  libcurl3
```

2. Setup Docker Repository:

Now, import Dockers official GPG key to verify packages signature before installing them with apt-get. Run the below command on terminal:

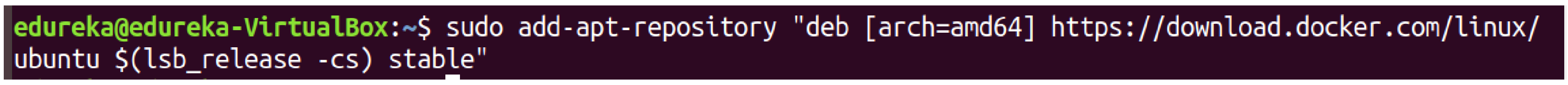
```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```



```
edureka@edureka-VirtualBox:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
OK
```

Now, add the Docker repository on your Ubuntu system which contains Docker packages including its dependencies, for that execute the below command:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

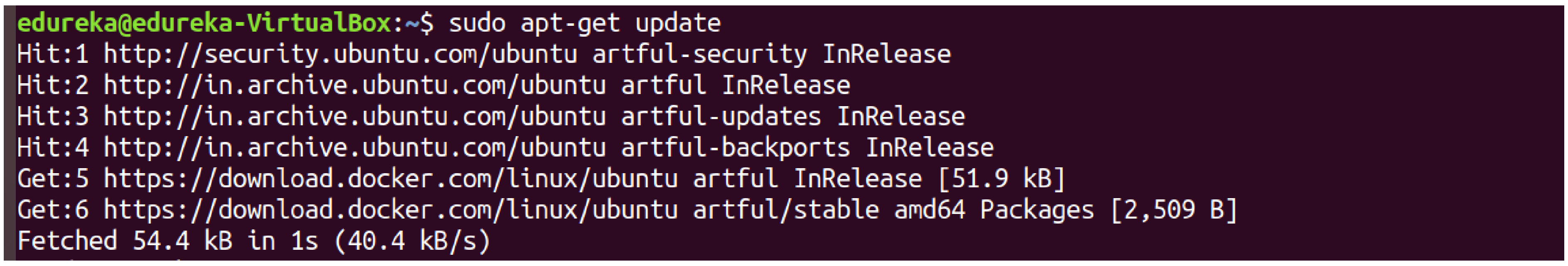


```
edureka@edureka-VirtualBox:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/
ubuntu $(lsb_release -cs) stable"
```

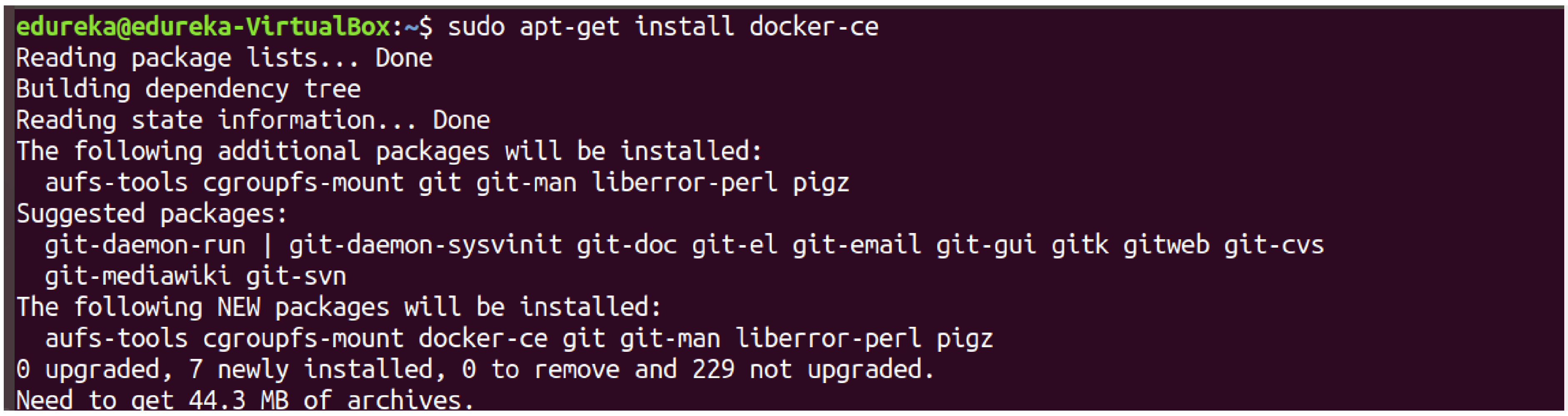
3. Install Docker On Ubuntu:

Now you need to upgrade apt index and install Docker community edition, for that execute the below commands:

```
sudo apt-get update
sudo apt-get install docker-ce
```



```
edureka@edureka-VirtualBox:~$ sudo apt-get update
Hit:1 http://security.ubuntu.com/ubuntu artful-security InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu artful InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu artful-updates InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu artful-backports InRelease
Get:5 https://download.docker.com/linux/ubuntu artful InRelease [51.9 kB]
Get:6 https://download.docker.com/linux/ubuntu artful/stable amd64 Packages [2,509 B]
Fetched 54.4 kB in 1s (40.4 kB/s)
```



```
edureka@edureka-VirtualBox:~$ sudo apt-get install docker-ce
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 pigz
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn
The following NEW packages will be installed:
  aufs-tools cgroupfs-mount docker-ce git git-man liberror-perl pigz
0 upgraded, 7 newly installed, 0 to remove and 229 not upgraded.
Need to get 44.3 MB of archives.
```

Congratulations! You have successfully installed Docker. Also, check out a few commonly used **Docker Commands**.

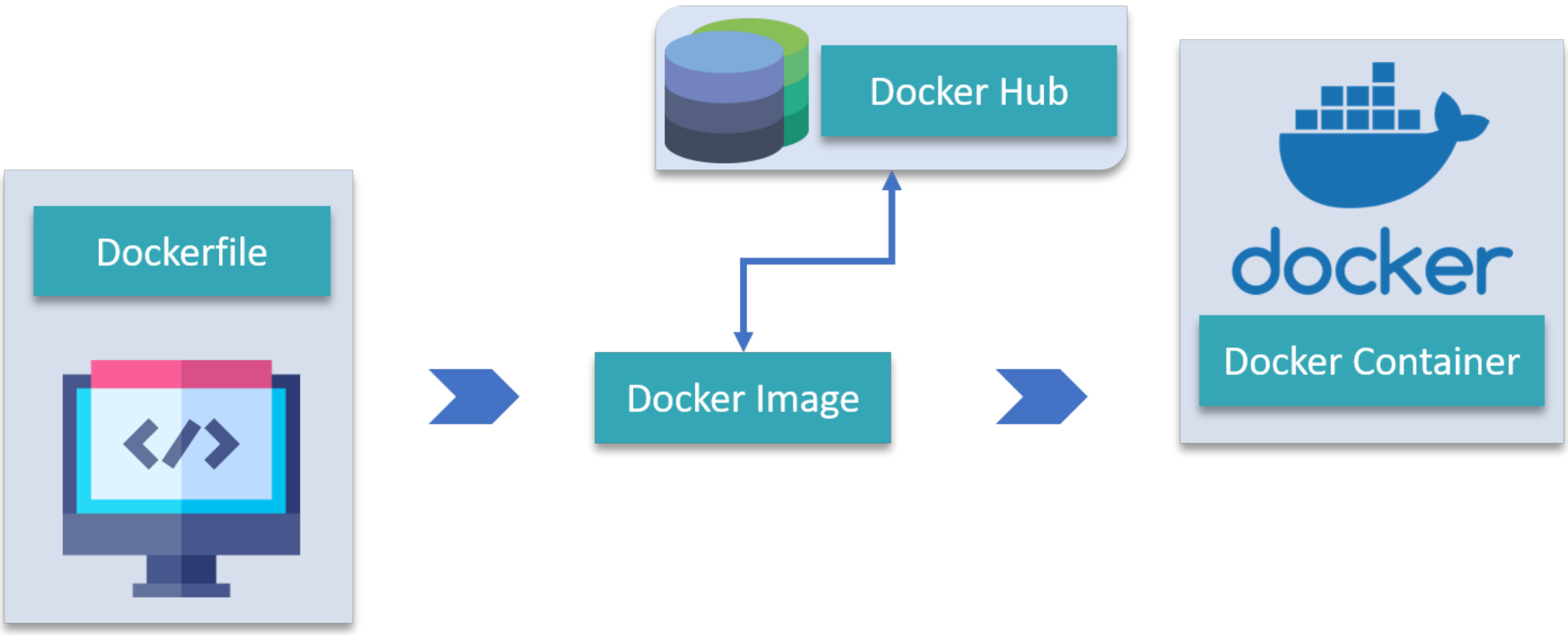
Now let us see a few important Docker concepts.

Dockerfile, Docker Image And Docker Container:

1. A Docker Image is created by the sequence of commands written in a file called as Dockerfile.
2. When this Dockerfile is executed using a docker command it results into a Docker Image with a name.
3. When this Image is executed by “docker run” command it will by itself start whatever application or service it must start on its execution.

Docker Hub:

Docker Hub is like GitHub for Docker Images. It is basically a cloud registry where you can find Docker Images uploaded by different communities, also you can develop your own image and upload on Docker Hub, but first, you need to create an account on DockerHub.



Docker Architecture:

It consists of a Docker Engine which is a client-server application with three major components:

1. A server which is a type of long-running program called a daemon process (the docker command).
2. A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
3. A command line interface (CLI) client (the docker command).
4. The CLI uses the Docker REST API to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI.

Refer this blog, to read more about **Docker Architecture**.

Finally in this Docker Tutorial blog I will be talking about Docker Compose.

Docker Compose:

Docker Compose is basically used to run multiple Docker Containers as a single server. Let me give you an example:

Suppose if I have an application which requires WordPress, Maria DB and PHP MyAdmin. I can create one file which would start both the containers as a service without the need to start each one separately. It is really useful especially if you have a microservice architecture.

Refer my blog on **Docker Container** to understand how to execute it practically.

Also, you can read this blog on how you can containerize a mean stack application using Docker Compose.

With this, we come to an end to Docker Tutorial’s first blog on Introduction To Docker & Containerization.

Go through our next blog on Docker: What Is Docker and Docker Containers?

*Now that you have understood what is DevOps, check out our **DevOps Certification Training** by Edureka, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe. The Edureka DevOps Certification Training course helps learners gain expertise in various DevOps processes and tools such as Puppet, Jenkins, Nagios, Ansible, Chef, Saltstack and GIT for automating multiple steps in SDLC.*

Got a question for us? Please mention it in the comments section and we will get back to you.

Recommended videos for you

Continuous Integration With Jenkins

Watch Now

Devops : Automate Your Infrastructure With Puppet

Watch Now

What is Jenkins? Continuous Integration With Jenkins

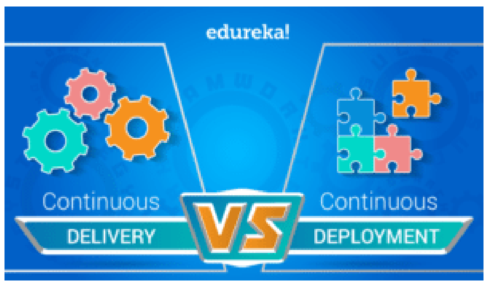
Watch Now

DevOps Tutorial For

Watch Now

‹›

Recommended blogs for you



Continuous Delivery vs Continuous Deployment

Read Article

How To Set Kubernetes Ingress Controller on AWS

Read Article

Setting Up Development Environment Using Vagrant

Read Article

Ansible Tutorial – Le Write Ansible Playbo

Read Article

‹›

Comments

5 Comments

8 Comments https://www.edureka.co/blog/ 1 Login

♡ Recommend 6 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Jeevan Chaitanya • 2 years ago

Understood docker now

1 ^ | ˅ • Reply • Share ›

> **EdurekaSupport** Mod ➦ Jeevan Chaitanya • 2 years ago
>
> We are happy to have helped, Jeevan. Do check out our other blogs on the various tools used for DevOps, you will definitely find it interesting! :)
>
> ^ | ˅ • Reply • Share ›

Paradise Techsoft Solutions • 25 days ago

Great Article. Hope these all will be helpful for me during my learning period. they are very informative. Looking forward to learn more. I am a blogger too so you can visit my blog here: https://blog.paradisetechso...

^ | ˅ • Reply • Share ›

Harvey Jones • 7 months ago

As a beginner in Docker this article cleared most of the confusions I had. This cheat sheet was also useful.

^ | ˅ • Reply • Share ›

Srinimf • 10 months ago

Containerization much useful nowadays. A Docker container can now run in Zo/s mainframe in Linux environment. A light weight Docker container is portable and less maintenance.

^ | ˅ • Reply • Share ›

Pradeep kumar • a year ago • edited

Thanks for the clear explanation.
In virtualization , we can install different operating systems like
windows, ubuntu, on top of host OS, but in container is always only one
host os and we cannot run on different operating system?? Am i correct or wrong??

^ | ˅ • Reply • Share ›

Pradeep kumar • a year ago

In virtualization , we can install different operating systems like windows, ubuntu, on top of host OS, but in container is always only one host os and we cannot run on different operating system??

^ | ˅ • Reply • Share ›

> **Pradeepta Puhan** ➦ Pradeep kumar • 2 months ago
>
> In virtualization, hardware of host os being shared and blocked by Hypervisors like (VMware, Virtualbox..etc). However, in containerization no separate hardware used, which makes it lightning fast. Container can contain any type of images (Unix, Linux, Windows) along with different applications which you can use according to your requirement.
>
> ^ | ˅ • Reply • Share ›

ALSO ON HTTPS://WWW.EDUREKA.CO/BLOG/

How to Get Started With Your Web Developer Career?

1 comment • 10 days ago

Avatar Vignesh. M — I am pursuing by be degree in mechanical engineering but I'm interested in web development. I am learning it. But my

Palindrome in Python: How to check a number is palindrome?

1 comment • 2 months ago

Avatar Mantu — Input a string and if the string is not palindrome add the number of characters to make it palindrome and print the number of

How To Convert Lists To Strings In Python?

3 comments • 25 days ago

Top 8 Data Science Tools Everyone Should Know

1 comment • 3 months ago

Trending Courses in DevOps

DevOps Certification Training

57k Enrolled Learners
Weekend/Weekday
Live Class

Reviews

★★★★★ 5 (22450)

Kubernetes Certification Training

4k Enrolled Learners
Weekend
Live Class

Reviews

★★★★★ 5 (1550)

Docker Training and Certification

5k Enrolled Learners
Weekend
Live Class

Reviews

★★★★★ 5 (1650)

AWS Certified Dev
Engineer Training

2k Enrolled Learner
Weekend
Live Class

Reviews

★★★★★ 5 (750)

‹›

Browse Categories

Artificial Intelligence | BI and Visualization | Big Data | Blockchain | Cloud Computing | Cyber Security | Data Science | Data Warehousing and ETL | Databases | Digital Marketing | Front End Web Development | Mobile Development | Operating Systems | Programming & Frameworks | Project Management and Methodologies | Robotic Process Automation | Software Testing | Systems & Architecture

edureka!

TRENDING CERTIFICATION COURSES

- DevOps Certification Training
- AWS Architect Certification Training
- Big Data Hadoop Certification Training
- Tableau Training & Certification
- Python Certification Training for Data Science
- Selenium Certification Training
- PMP® Certification Exam Training
- Robotic Process Automation Training using UiPath
- Apache Spark and Scala Certification Training
- Microsoft Power BI Training
- Online Java Course and Training
- Python Certification Course

TRENDING MASTERS COURSES

- Data Scientist Masters Program
- DevOps Engineer Masters Program
- Cloud Architect Masters Program
- Big Data Architect Masters Program
- Machine Learning Engineer Masters Program
- Full Stack Web Developer Masters Program
- Business Intelligence Masters Program
- Data Analyst Masters Program
- Test Automation Engineer Masters Program
- Post-Graduate Program in Artificial Intelligence & Machine Learning
- Post-Graduate Program in Big Data Engineering

COMPANY

- About us
- News & Media
- Reviews
- Contact us
- Blog
- Community
- Sitemap
- Blog Sitemap
- Community Sitemap

WORK WITH US

- Careers
- Become an Instructor
- Become an Affiliate
- Become a Partner
- Hire from Edureka

DOWNLOAD APP

Download on the App Store | GET IT ON Google Play

CATEGORIES

CATEGORIES

Cloud Computing | DevOps | Big Data | Data Science | BI and Visualization | Programming & Frameworks | Software Testing |
Project Management and Methodologies | Robotic Process Automation | Frontend Development | Data Warehousing and ETL | Artificial Intelligence |
Blockchain | Databases | Cyber Security | Mobile Development | Operating Systems | Architecture & Design Patterns | Digital Marketing

TRENDING BLOG ARTICLES

TRENDING BLOG ARTICLES

Selenium tutorial | Selenium interview questions | Java tutorial | What is HTML | Java interview questions | PHP tutorial | JavaScript interview questions |
Spring tutorial | PHP interview questions | Inheritance in Java | Polymorphism in Java | Spring interview questions | Pointers in C | Linux commands |
Android tutorial | JavaScript tutorial | jQuery tutorial | SQL interview questions | MySQL tutorial | Machine learning tutorial | Python tutorial |
What is machine learning | Ethical hacking tutorial | SQL injection | AWS certification career opportunities | AWS tutorial | What Is cloud computing |
What is blockchain | Hadoop tutorial | What is artificial intelligence | Node Tutorial | Collections in Java | Exception handling in java |
Python Programming Language | Python interview questions | Multithreading in Java | ReactJS Tutorial | Data Science vs Big Data vs Data Analyt... |
Software Testing Interview Questions | R Tutorial | Java Programs | JavaScript Reserved Words and Keywor... | Implement thread.yield() in Java: Exam... |
Implement Optical Character Recogniti... | All you Need to Know About Implemen...