

What is Docker Container? – Containerize Your Application Using Docker

Last updated on May 22,2019 13.5K Views



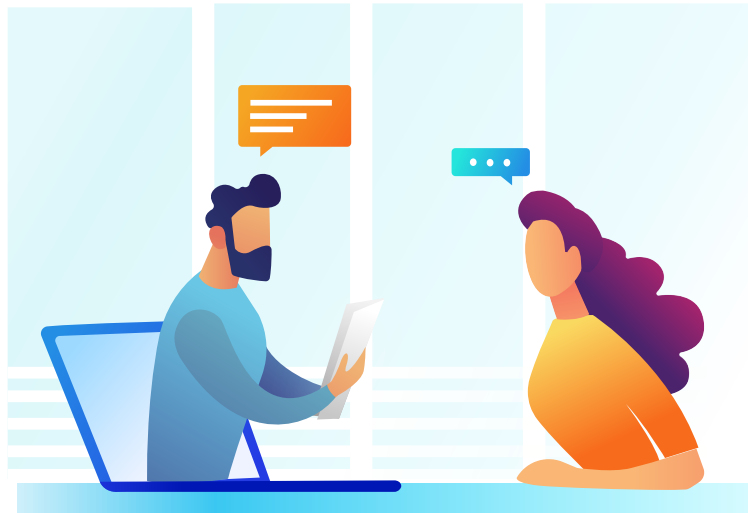
Saurabh

Saurabh is a technology enthusiast working as a Research Analyst at Edureka....

edureka!

NEW
LAUNCH

myMock Interview Service for Real Tech Jobs



- [Mock interview in latest tech domains i.e JAVA, AI, DEVOPS,etc](#)
- [Get interviewed by leading tech experts](#)
- [Real time assessment report and video recording](#)

TRY OUT MOCK INTERVIEW



Well, I am hoping you have read my previous blogs on [Docker](#) where I have covered the basics of Docker. Here, in this Docker Container blog I will be discussing about what are Docker Containers and how it works. Mostly, we will be focusing on Hands-on and use-cases of Docker.

I have listed down the topics for this Docker Container blog:

- Why We Need Docker Containers?
- How Docker Containers work?
- Use-Cases of Docker Container

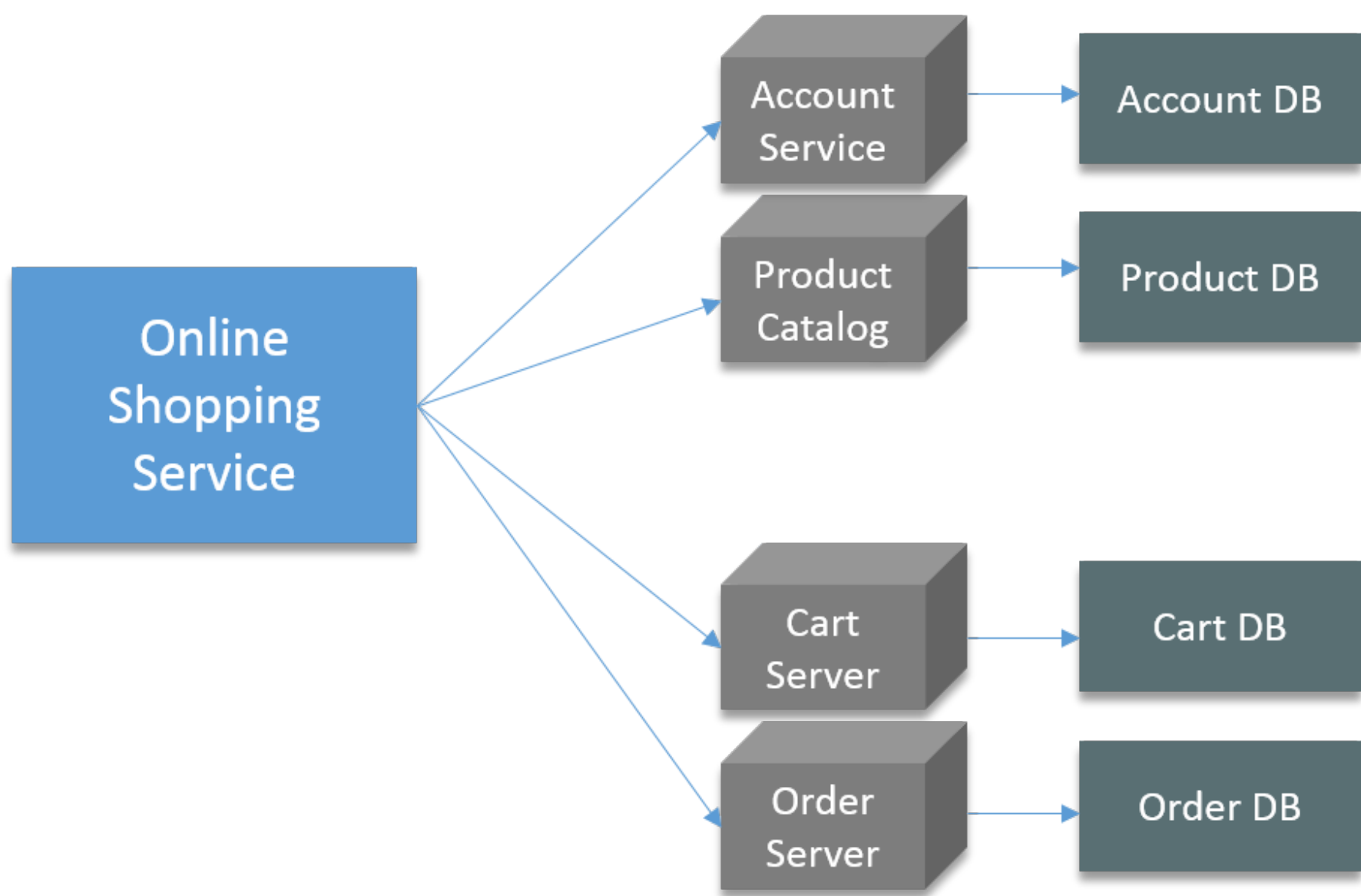
Why We Need Docker Containers?

I still remember it correctly, I was working on a project. In that project we were following the microservice architecture. For those of you who don't know what is microservice, don't worry I will give you an introduction to it.

The idea behind microservices is that certain types of applications become easier to build and maintain when they are broken down into smaller, composable pieces which work together. Each component is developed separately, and the application is then simply the sum of its constituent components.

Consider the example below:





In the above diagram there is an online shop with separate microservices for user-account, product catalog, order processing and shopping carts.

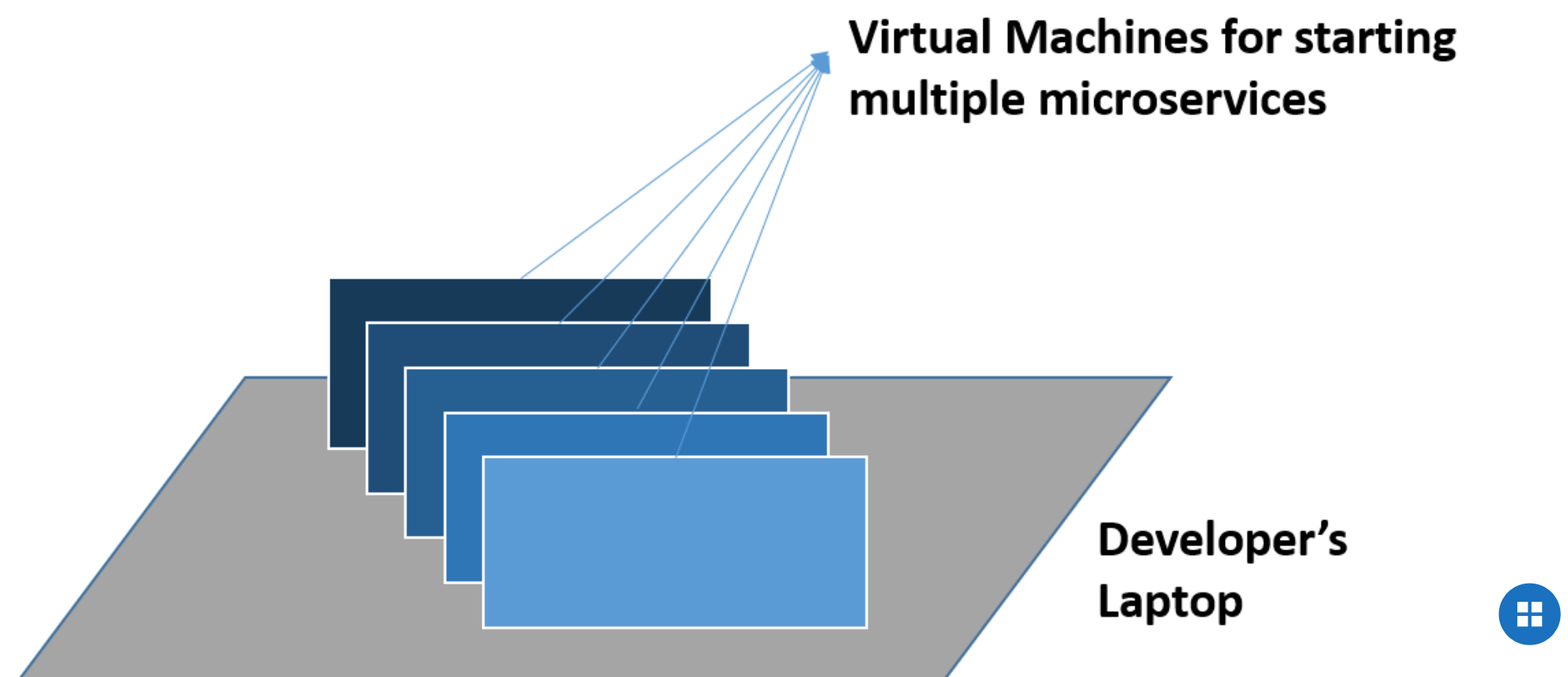
Well, this architecture has a lot of benefits:

- Even if one of your microservice fails, your entire application is largely unaffected.
- It is easier to manage

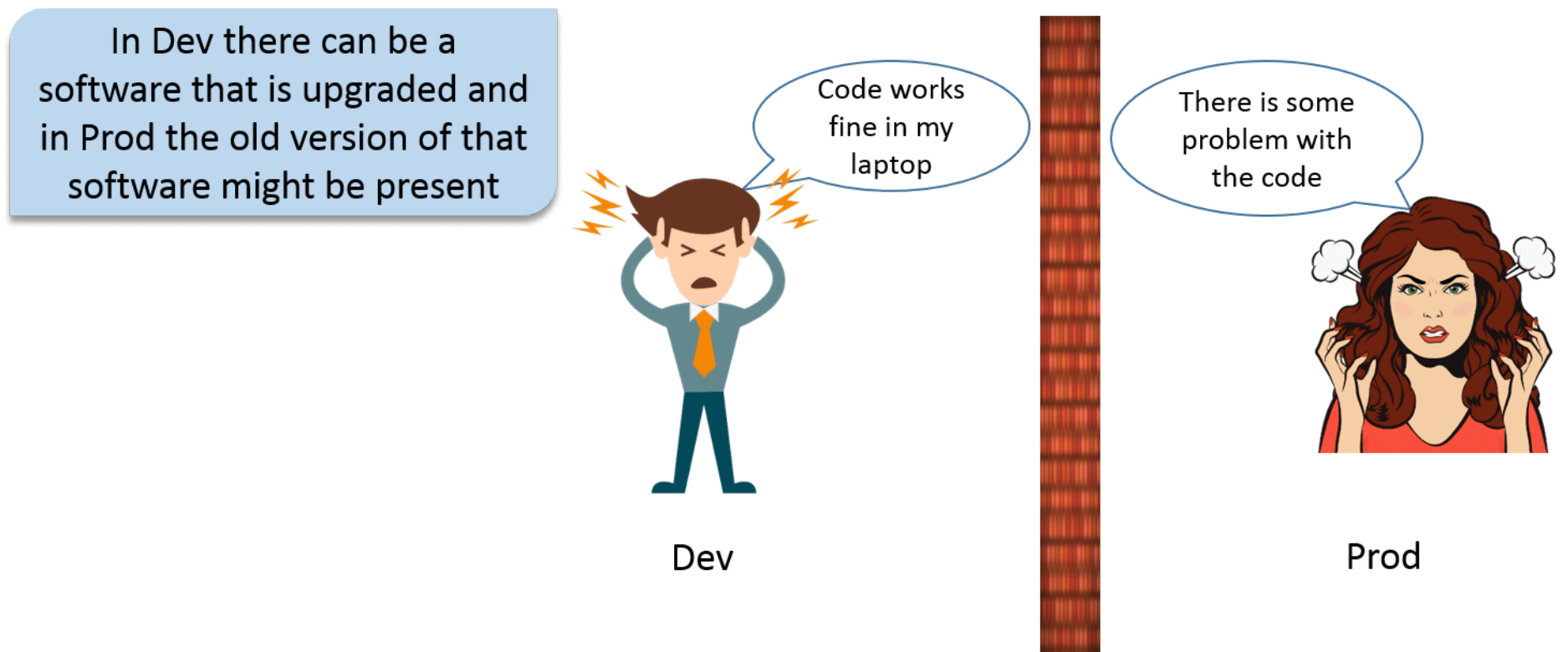
There are many other benefits as well, I won't go into much detail about microservices in this post. But, soon I will be coming up with a couple of blogs on microservices as well.

In this architecture, we were using CentOS Virtual Machines. Those Virtual Machines were configured by writing long scripts. Well, configuring those VMs was not the only problem.

Developing such applications requires starting of several of microservices in one machine. So if you are starting five of those services you require five VMs on that machine. Consider the diagram below:



The other problem is pretty common, I know a lot of you can relate to it. The application works in a developer's laptop but not in testing or production. This can be because of not keeping a consistent computing environment. Consider the diagram below:



There were many other problems apart from this as well, but I feel, these problems are enough for me to explain you the need of Docker Containers.

[Learn How Docker Containers Are Better Than Virtual Machines](#)

So, imagine if I am giving 8 GB of RAM to all my VMs, and I have 5 microservices running on different Virtual Machines. In that case, these VMs will require 40 GB of RAM. Well, now I require the configurations of my host machine to be very high, almost 44 GB of RAM should be there in my host machine. Obviously, this is not a sustainable solution for such an architecture because, I am wasting a lot of resources here.

Fine, I have a lot of resources to waste, but still I have a problem of inconsistency in my software delivery life-cycle (SDLC). I have to configure these VMs in test as well as in prod environment. Somewhere in that process, some software was not updated in the test server, and the Dev team is using the updated version of the software. This leads to conflicts.

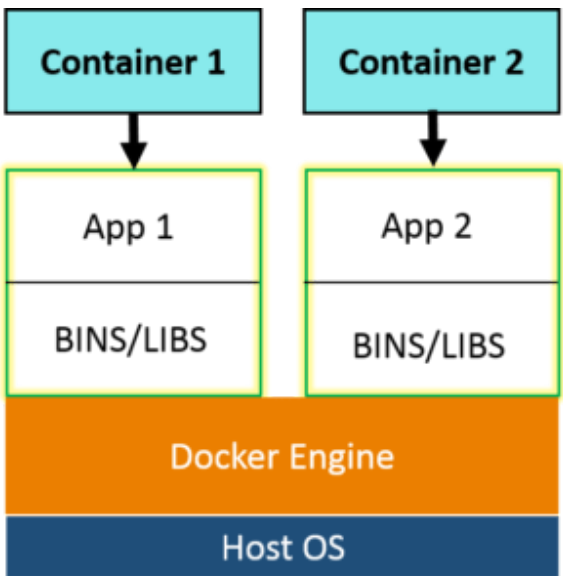
What if I am using 100 VMs, then configuring each VM will take a lot of time, and at the same time it is prone to error as well.

Now, let us understand what is Docker Container and how it works, and how it solved my problem.

What is a Docker Container?

Docker is a tool designed to make it easier to create, deploy and run applications by using containers.

You can create Docker Containers, these containers will contain all the binaries and libraries required for your application or microservice in my case. So your application is present in a container, or you have containerized your application. Now, that same container can be used in the Test and Prod environment.



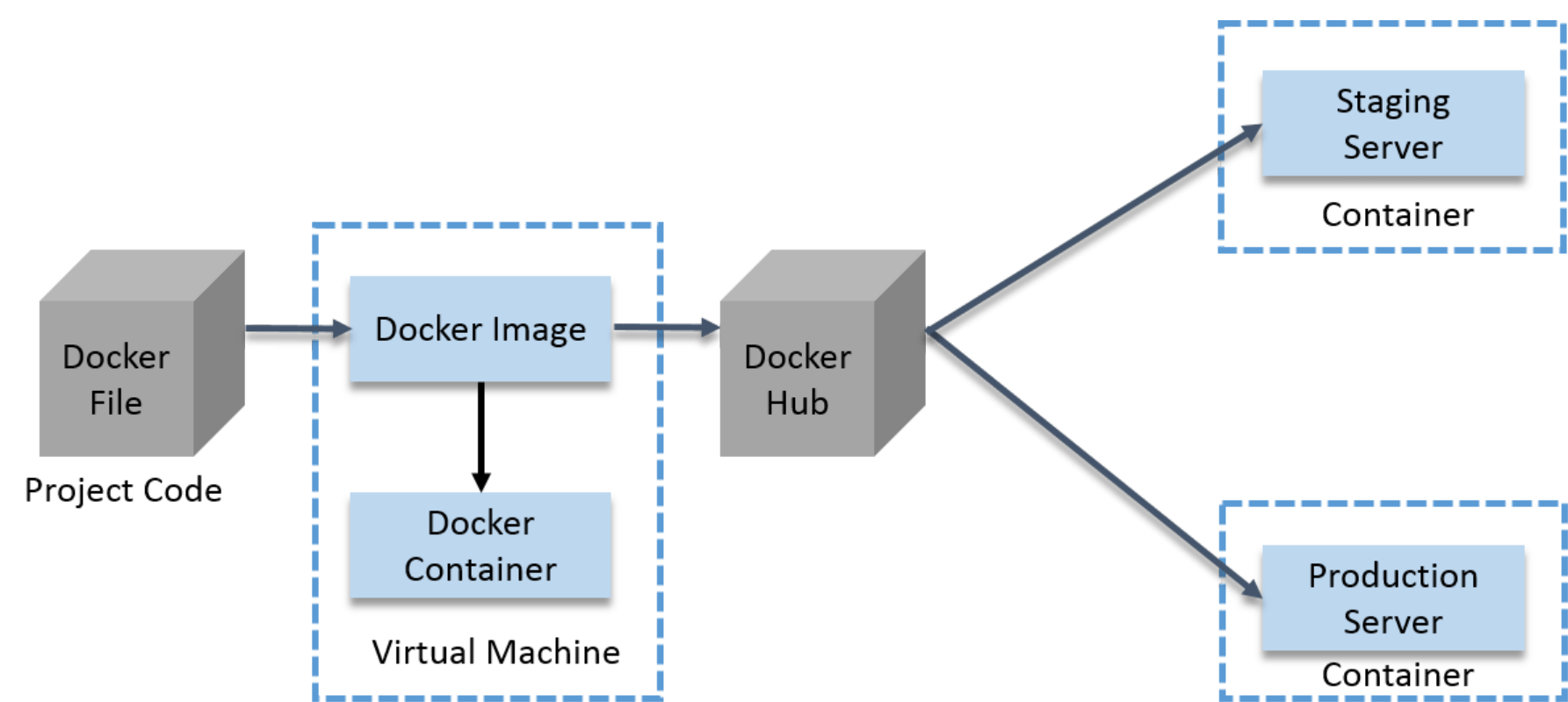
Docker Containers are a lightweight solution to Virtual Machines, and it uses the host OS. The best part, you don't have to pre-allocate any RAM to the Docker Container, it will take it as and when required. So, with Docker Container I don't have to worry about wastage of resources.

Let's understand now, how a Docker Container works.

How a Docker Container Works?



The below diagram is basically, a way to use Docker. And I am assuming that, you have an idea about Docker Image and Dockerfile.



Docker Training and Certification

[Instructor-led Sessions](#)

[Real-life Case Studies](#)

[Assignments](#)

[Lifetime Access](#)

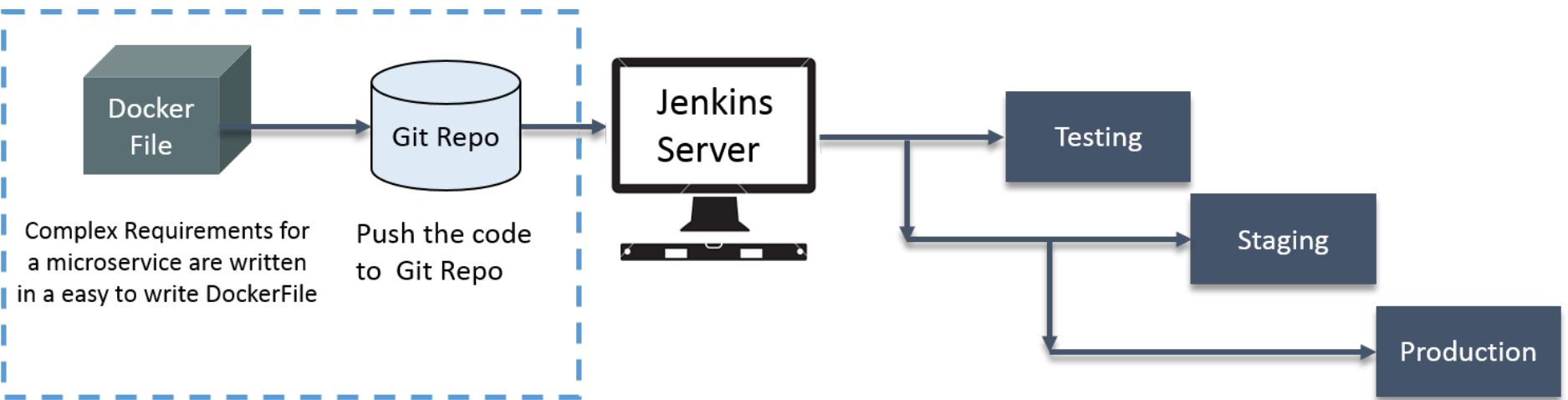
[Explore Curriculum](#)

Guys, I know the diagram looks a bit complex, but trust me it ain't that complex. Below is the explanation of the diagram, even after that you feel it is tough to understand, you can comment your doubt, I will address those questions ASAP.

- A developer will first write the project code in a Docker file and then build an image from that file.
- This image will contain the entire project code.
- Now, you can run this Docker Image to create as many containers as you want.
- This Docker Image can be uploaded on Docker hub (It is basically a cloud repository for your Docker Images, you can keep it public or private).
- This Docker Image on the Docker hub, can be pulled by other teams such as QA or Prod.

This not only prevents the wastage of resources, but also makes sure that the computing environment that is there in a Developer's laptop is replicated in other teams as well. I feel now, I don't have to tell you why we need Docker.

This was one way to use it, I am guessing you guys must be curious to know how I used Docker to solve my problem of microservices. Let me give you an overview on the same.



Below is the explanation of the diagram:

- Firstly, we wrote the complex requirements within a Dockerfile.
- Then, we pushed it on GitHub.
- After that we used a CI server (Jenkins).



- This Jenkins server will pull it down from Git, and the build the exact environment. This will be used in Production servers as well as in Test servers.
- We deployed it out to staging (It refers to deploying your software onto servers for testing purposes, prior to deploying them fully into production.) environments for Testers.
- Basically, we rolled exactly what we had in Development, Testing and Staging into Production.

It will be actually fair to say that, Docker made my life easy.

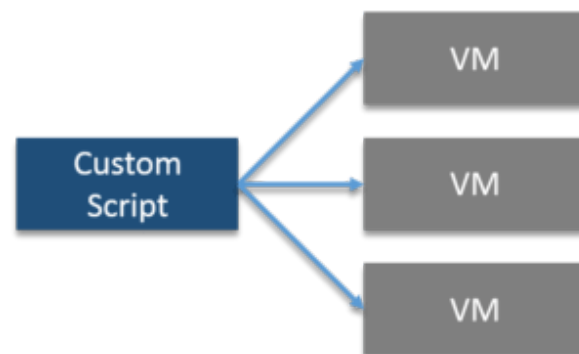
Well, that was the story of my company, let's look at the case-study of Indiana University. How Docker solved their problems.

Indiana University Case-Study:

Indiana University is a multi-campus public university system in the state of Indiana, United States.

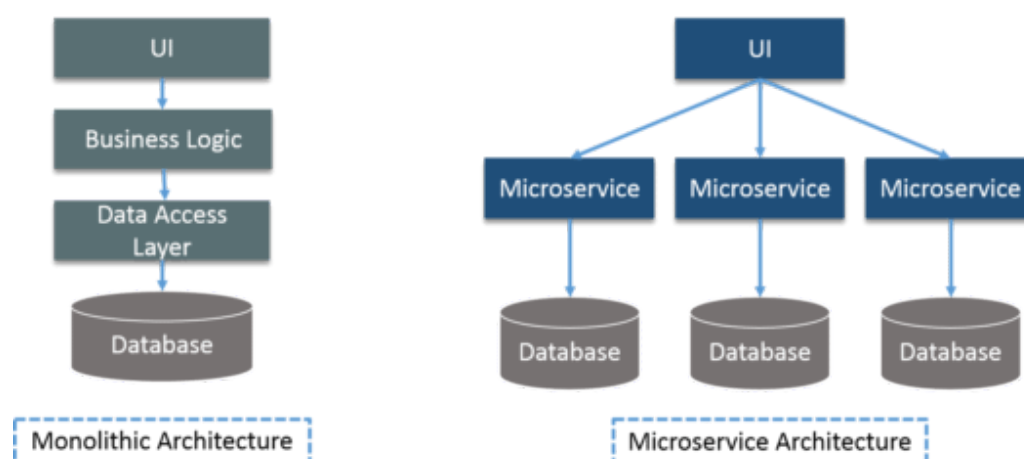
Problem Statement

They were using custom scripts to deploy the applications in the VM.



Their environment was optimized for their legacy Java-based applications. Their growing environment involves new products that aren't solely java based. In order to give their students the best experience possible, the University needed to begin modernizing the applications.

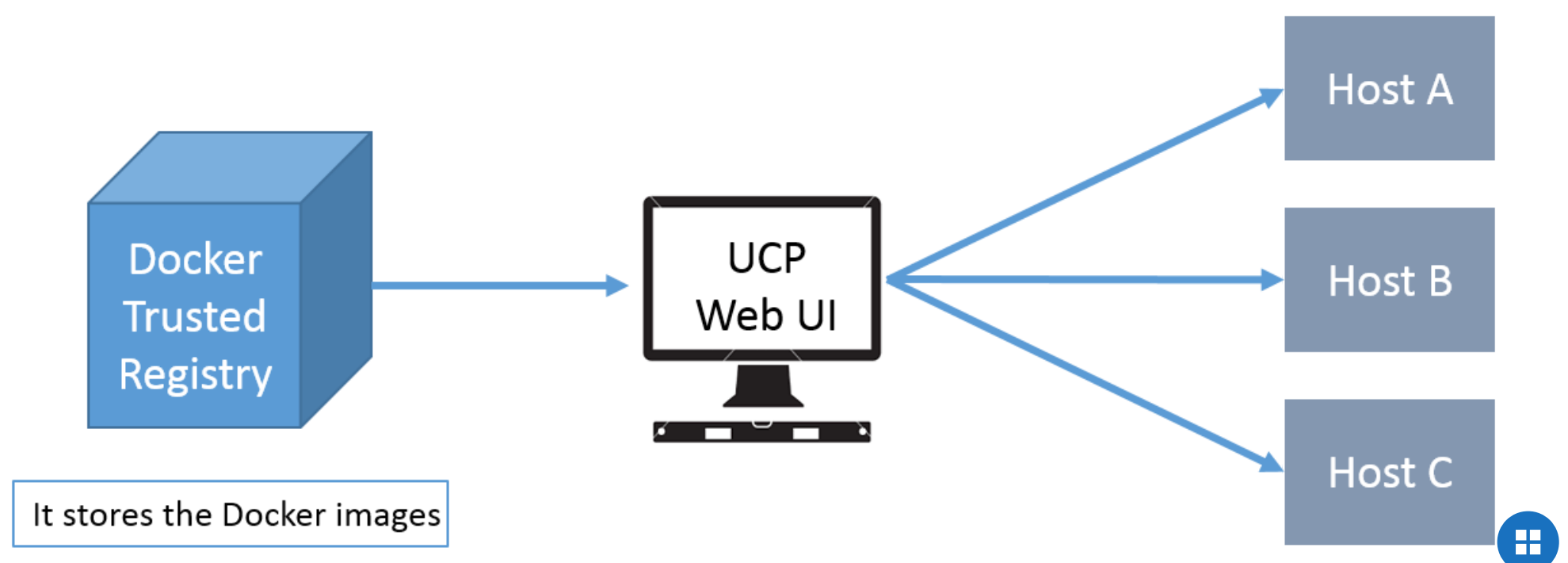
The University wanted to improve the way they architect applications, by moving to a microservices based architecture for their applications.



Security was needed for student's data such as SSNs and student health data.

Solution:

All the problems were addressed by Docker Data Center (DDC), consider the diagram below:



Docker Trusted Registry – It stores the Docker Images.

UCP (Universal Control Plane) Web UI – Helps in managing whole cluster from a single place. Services are deployed using UCP web UI, using Docker images that are stored in DTR (Docker Trusted Registry).

IT ops teams leverages Universal Control Plane to provision Docker installed software on hosts, and then deploy their applications without having to do a bunch of manual steps to set up all their infrastructure.

UCP and DTR integrates with their LDAP server to quickly provision access to their applications.

I am hoping you guys have read the previous blogs to learn the basics of Docker.

Now, I will explain you how we can use Docker Compose for multi container application.

Docker Hands-On:

I am assuming you have installed Docker. I will be using Docker Compose in this post, below I have given a small introduction to Docker Compose.

Docker Compose: It is a tool for defining and running multi-container Docker applications. With Docker Compose, you can use a Compose file to configure your application’s services. Then, using a single command, you can create and start all the services from your configuration.

Suppose you have multiple applications in various containers and all those containers are linked together. So, you don’t want to execute each of those containers one by one. But, you want to run those containers with a single command. That’s where Docker Compose comes in to the picture. With it you can run multiple applications in various containers with a single command. i.e. docker-compose up.

vOps Training



[DEVOPS
CERTIFICATION
TRAINING](#)

DevOps Certification
Training


Reviews
★★★★★ 5(56038)



[KUBERNETES
CERTIFICATION
TRAINING](#)

Kubernetes Certification
Training

Reviews
★★★★★ 5(3808)



[DOCKER TRAINING
AND CERTIFICATION](#)

Docker Training and
Certification

Reviews
★★★★★ 5(4034)

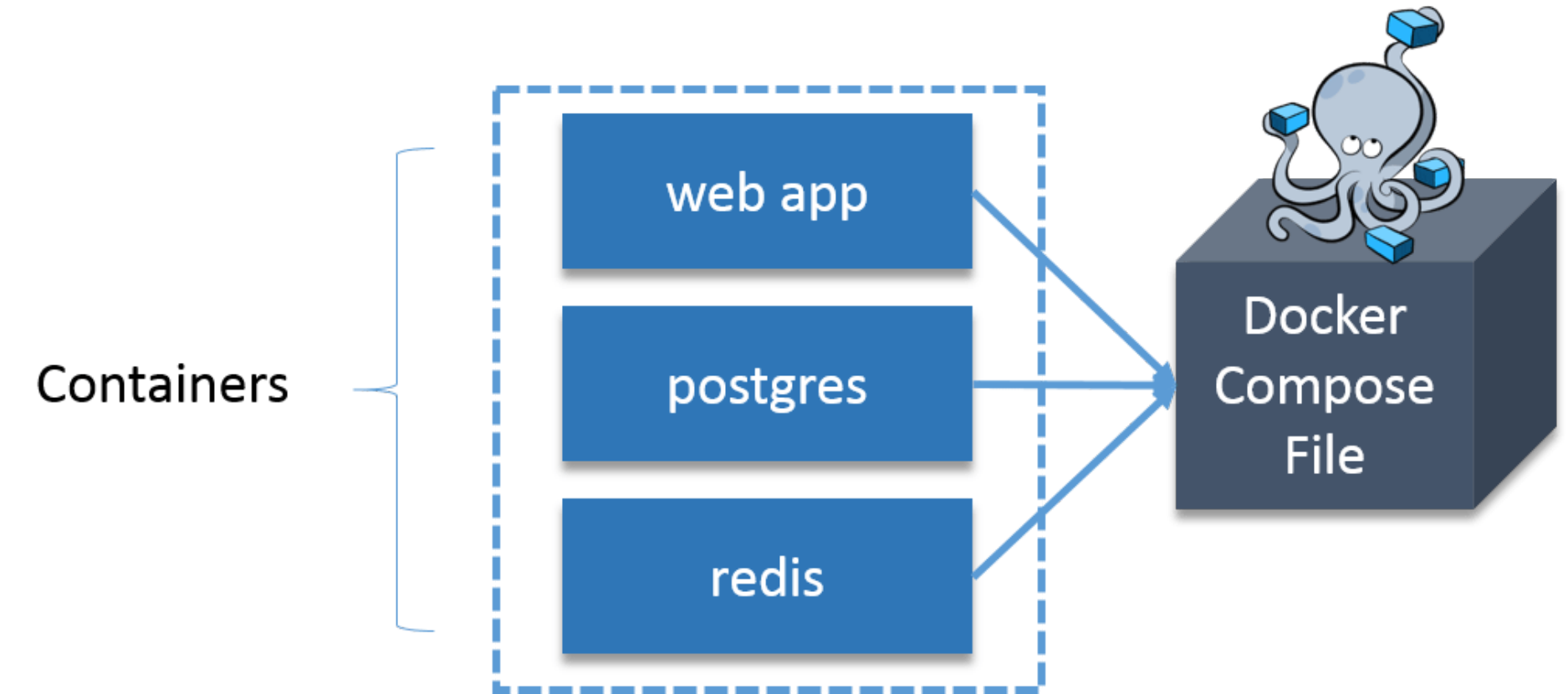


[AWS CERTIFIED
DEVOPS ENGINEER
TRAINING](#)

AWS Certified DevOps
Engineer Training

Reviews
★★★★★ 5(1757)

Example: Imagine you have different containers, one running a web app, another running a postgres and another running redis, in a YAML file. That is called docker compose file, from there you can run these containers with a single command.



Let us take one more example:



Suppose you want to publish a blog, for that you will use CMS (Content Management System), and wordpress is the most widely used CMS. Basically, you need one container for WordPress and you need one more container as MySQL for back end, that MySQL container should be linked to the wordpress container. We also need one more container for Php Myadmin that will be linked to MySQL database, basically, it is used to access MySQL database.

How about I execute the above stated example practically.

Steps involved:

1. **Install Docker Compose:**
2. **Install WordPress:** We'll be using the official [WordPress](#) and [MariaDB](#) Docker images.
3. **Install MariaDB:** It is one of the most popular database servers in the world. It's made by the original developers of MySQL. MariaDB is developed as open source software and as a relational database it provides an SQL interface for accessing data.
4. **Install PhpMyAdmin:** It is a free software tool written in PHP, intended to handle the administration of MySQL over the Web.
5. **Create The WordPress Site:**

Let's get started!

Install Docker Compose:

Install Python Pip first:

```
sudo apt-get install python-pip
```

```
edureka@ubuntu:~$ sudo apt-get install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
algorithm-diff-perl
perl libfakeroot
```

Now, you can install Docker Compose:

```
sudo pip install docker-compose
```

```
edureka@ubuntu:~$ sudo pip install docker-compose
Downloading/unpacking docker-compose
  Downloading docker-compose-1.9.0.tar.gz (156kB): 156kB downloaded
  Running setup.py (path:/tmp/pip_build_root/docker-compose/setup.py) egg_info for package docker-compose

  warning: no previously-included files found matching 'README.md'
  warning: no previously-included files matching '*.pyc' found anywhere in distribution
  warning: no previously-included files matching '*.pyo' found anywhere in distribution
  warning: no previously-included files matching '*.un~' found anywhere in distribution
Downloading/unpacking cached-property>=1.2.0,<2 (from docker-compose)
  Downloading cached_property-1.3.0-py2.py3-none-any.whl
Downloading/unpacking decorator>=0.6.1,<0.7 (from docker-compose)
```

Install WordPress:

Create a wordpress directory:

```
mkdir wordpress
```

Enter this wordpress directory:

```
cd wordpress/
```

```
edureka@ubuntu:~$ mkdir wordpress
edureka@ubuntu:~$ cd wordpress/
edureka@ubuntu:~/wordpress$
```

In this directory create a Docker Compose YAML file, then edit it using gedit:

```
sudo gedit docker-compose.yml
```

```
edureka@ubuntu:~/wordpress$ sudo gedit docker-compose.yml
```

Paste the below lines of code in that yaml file:



```

1  wordpress:
2
3  image: wordpress
4
5  links:
6
7  - wordpress_db:mysql
8
9  ports:
10
11 - 8080:80
12
13 wordpress_db:
14
15 image: mariadb
16
17 environment:
18
19 MYSQL_ROOT_PASSWORD: edureka
20
21 phpmyadmin:
22
23 image: corbinu/docker-phpmyadmin
24
25 links:
26
27 - wordpress_db:mysql
28
29 ports:
30
31 - 8181:80
32
33 environment:
34
35 MYSQL_USERNAME: root
36
37 MYSQL_ROOT_PASSWORD: edureka

```

I know you want me to explain this code, so what I will do, I will take small sections of this code and explain you what's happening.

```

1  wordpress_db:
2  ...
3  environment:
4    MYSQL_ROOT_PASSWORD: edureka
5  ...

```

This will set an environment variable inside the wordpress_db container called MYSQL_ROOT_PASSWORD with your desired password. The MariaDB Docker image is configured to check for this environment variable when it starts up and will take care of setting up the DB with a root account with the password defined as MYSQL_ROOT_PASSWORD.

```

1  wordpress:
2  ...
3  ports:
4    - 8080:80
5  ...

```

The first port number is the port number on the host, and the second port number is the port inside the container. So, this configuration forwards requests on port 8080 of the host to the default web server port 80 inside the container.

```

1  phpmyadmin:
2  image: corbinu/docker-phpmyadmin
3  links:
4    - wordpress_db:mysql
5  ports:
6    - 8181:80
7  environment:
8    MYSQL_USERNAME: root
9    MYSQL_ROOT_PASSWORD: edureka

```

This grabs docker-phpmyadmin by community member corbinu, links it to our wordpress_db container with the name mysql (meaning from inside the phpmyadmin container references to the hostname mysql will be forwarded to our wordpress_db container), exposes its port 80 on port 8181 of the host system, and finally sets a couple of environment variables with our MariaDB username and password. This image does not automatically grab the MYSQL_ROOT_PASSWORD environment variable from the wordpress_db container's environment, the way the wordpress image does. We actually have to copy the MYSQL_ROOT_PASSWORD: edureka line from the wordpress_db container, and set the username to root.



Now start the application group:

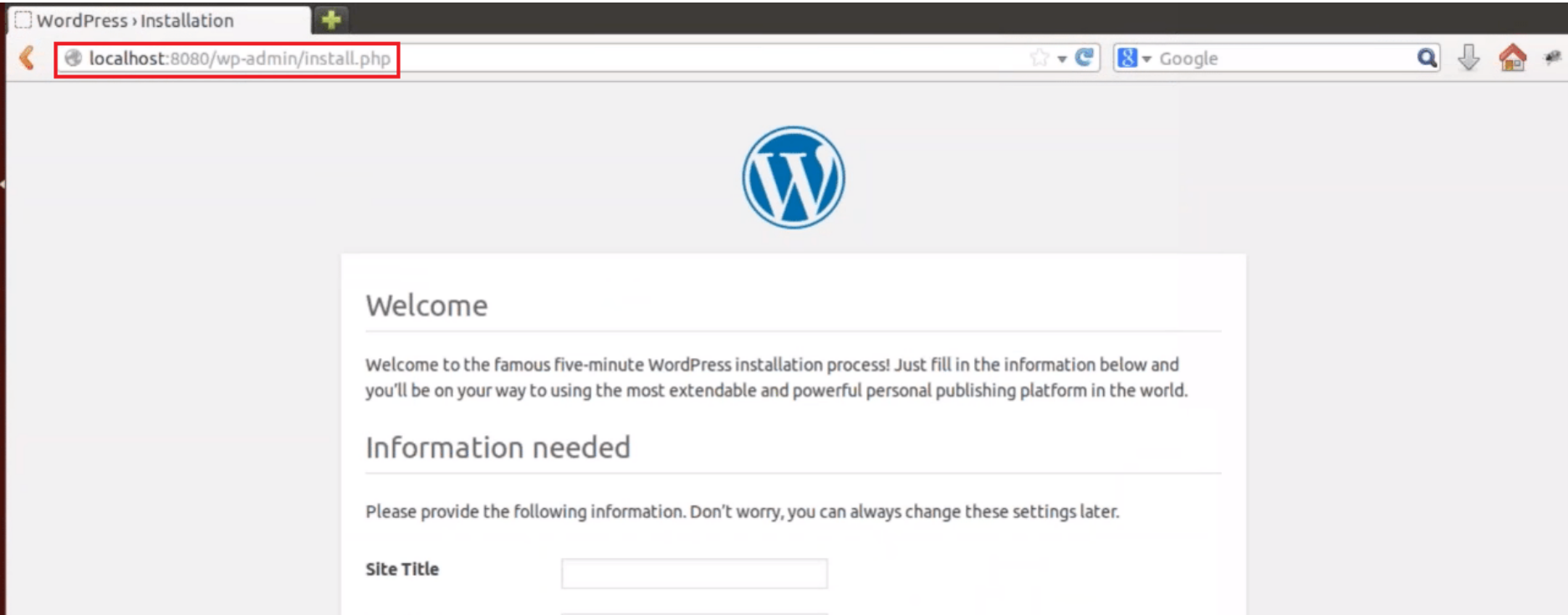
```
docker-compose up -d
```

```
edureka@ubuntu:~/wordpress$ sudo docker-compose up -d
Pulling wordpress_db (mariadb:latest)...
latest: Pulling from library/mariadb
75a822cd7888: Pulling fs layer
b8d5846e536a: Pulling fs layer
b75e9152a170: Pulling fs layer
832e6b030496: Waiting
034e06b5514d: Waiting
374292b6cca5: Waiting
```

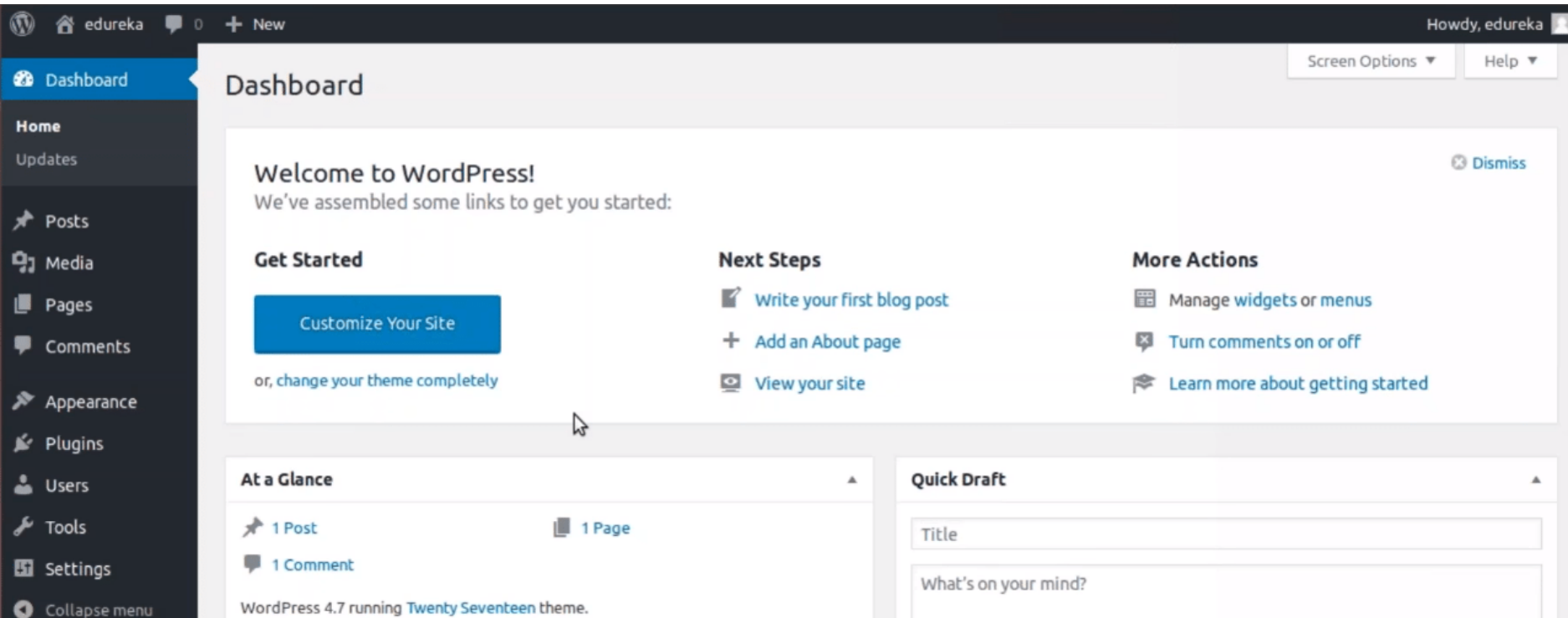
That's all you have to do. You can add as many containers as you like this way, and link them all up in any way you please.

Now, in the browser go to port 8080, using your public IP or host name, as shown below:

```
localhost:8080
```



Fill this form and click on install WordPress.



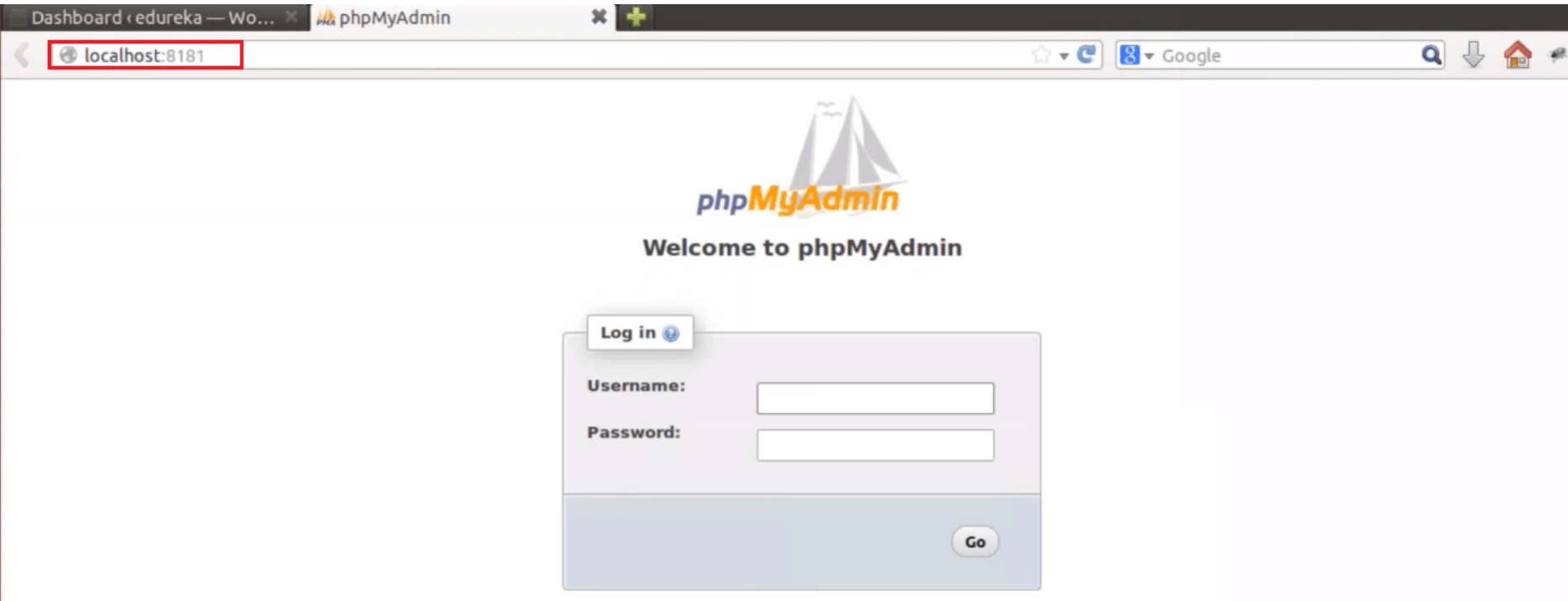
[Docker Training and Certification](#)

[Weekday / Weekend Batches](#)

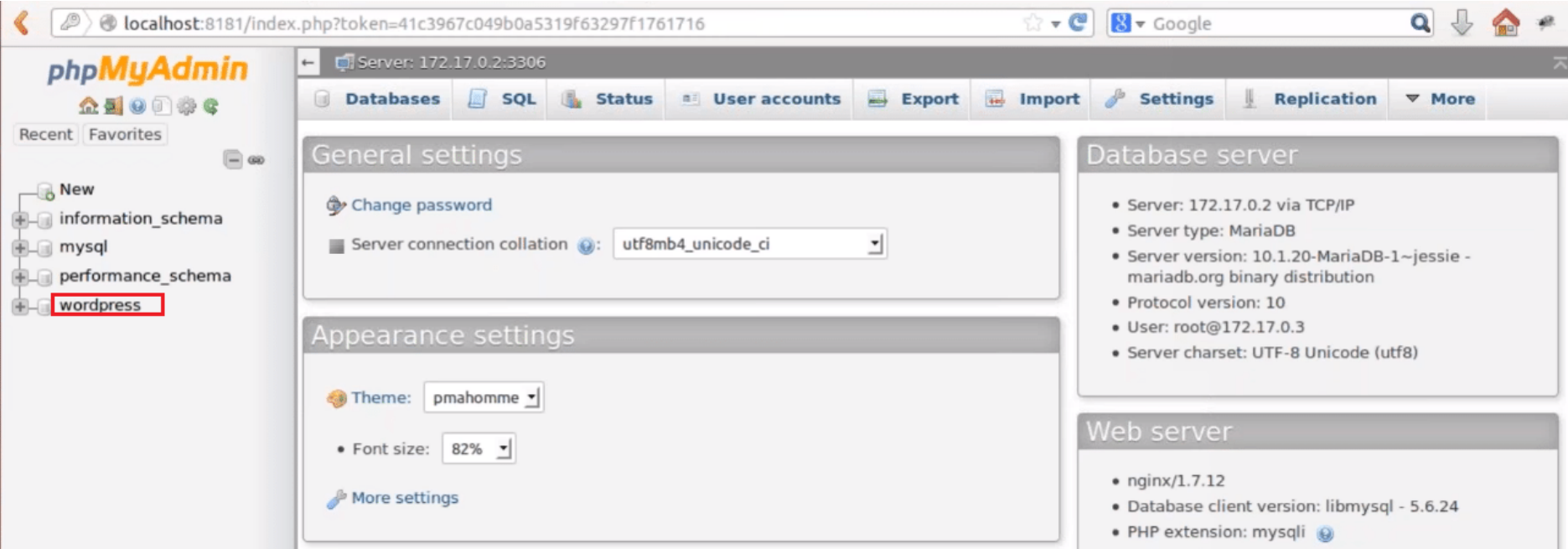
[See Batch Details](#)

Once it is finished, visit your server's IP address again (this time using port 8181, e.g. localhost:8181). You'll be greeted by the phpMyAdmin login screen:





Go ahead and login using username root and password you set in the YAML file, and you'll be able to browse your database. You'll notice that the server includes a wordpress database, which contains all the data from your WordPress install.



Here, I end my Docker Container blog. I hope you have enjoyed this post. You can check [other blogs](#) in the series too, which deal with the basics of Docker.

If you found this Docker Container blog relevant, check out the [DevOps 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, Docker, Nagios, Ansible, Chef, Saltstack and GIT for automating multiple steps in SDLC.

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

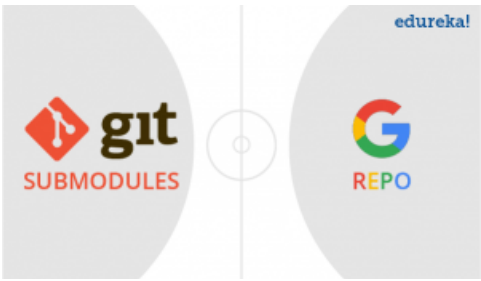
Recommended videos for you

What is DevOps – A Beginners Guide To DevOps
[Watch Now](#)

What is Docker – DevOps Tool For Containerization
[Watch Now](#)

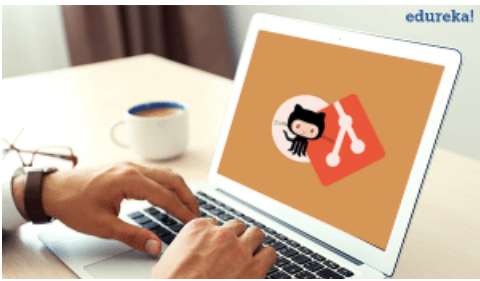
What is Git – A Complete Git Tutorial For Beginners
[Watch Now](#)

Ansible Tutorial For – Ansible Playbook
[Watch Now](#)



Git Submodules vs. Google's Repo Tool

[Read Article](#)



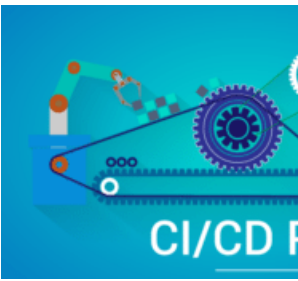
'Git'ting Ahead: Hacking Git and GitHub Part 1

[Read Article](#)



Top Nagios Interview Questions For 2019- All You Need To Know About Nagios

[Read Article](#)



CI CD Pipeline - Learn Setup a CI CD Pipeline from Scratch

[Read Article](#)

<>

Comments

1 Comment

1 Comment

<https://www.edureka.co/blog/>

[Login](#)

[Recommend](#)

[Tweet](#)

[Share](#)

[Sort by Best](#)



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

Name



Basil Joseph • 8 months ago

ERROR: In file './docker-compose.yml', service 'image' must be a mapping not a string.

[^](#) | [v](#) • [Reply](#) • [Share](#)

ALSO ON [HTTPS://WWW.EDUREKA.CO/BLOG/](https://www.edureka.co/blog/)

Git bisect: How to identify a bug in your code?

1 comment • 2 months ago



John — Excellent article! Helped me fix the bug in my code.

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

1 comment • 2 months ago



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

Python Basics: What makes Python so Powerful?

6 comments • a month ago



Techradius Hitech — It is just awesome blog ever

Advantages And Disadvantages Of Ethical Hacking

1 comment • 2 months ago



Tasmia — Can anyone just tell me the disadvantages of this! Please it's important!!

[Subscribe](#) [Add Disqus to your site](#) [Disqus' Privacy Policy](#)

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 DevOps Engineer Training](#)

[2k Enrolled Learners](#)

[Weekend](#)

[Live Class](#)

[Reviews](#)

[★★★★★ 5 \(7500\)](#)



<>

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



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](#)

COMPANY

- [About us](#)
- [News & Media](#)
- [Reviews](#)
- [Contact us](#)
- [Blog](#)
- [Community](#)
- [Sitemap](#)
- [Blog Sitemap](#)
- [Community Sitemap](#)

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](#)

WORK WITH US

- [Careers](#)
- [Become an Instructor](#)
- [Become an Affiliate](#)
- [Become a Partner](#)
- [Hire from Edureka](#)

DOWNLOAD APP



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...](#)



"PMP®","PMI®", "PMI-ACP®" and "PMBOK®" are registered marks of the Project Management Institute, Inc. MongoDB®, Mongo and the leaf logo are the registered trademarks of MongoDB, Inc.

