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:

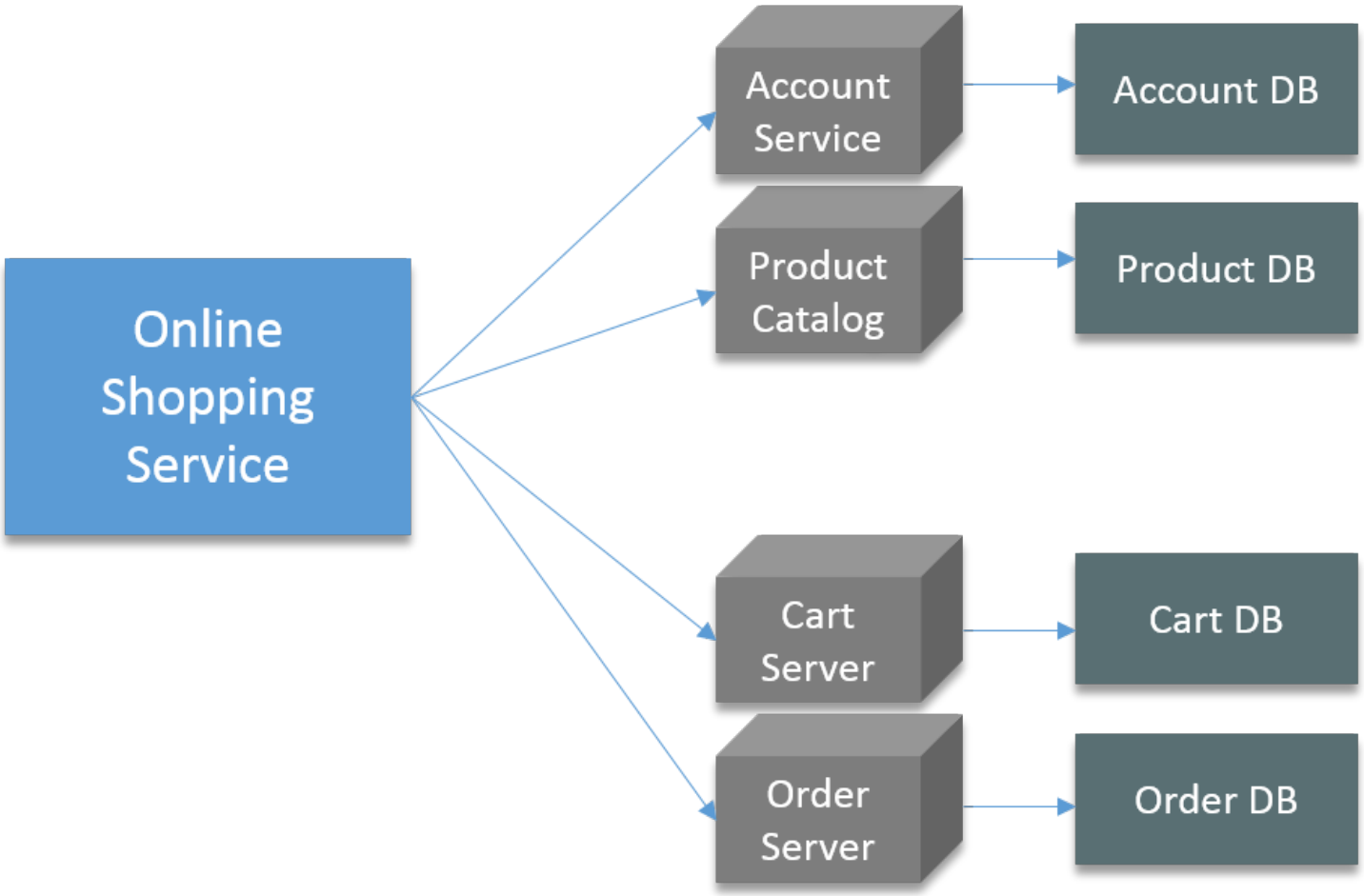
- Why We Need Docker Containers?
- How Docker Containers work?
- Use-Cases of Docker Container

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.

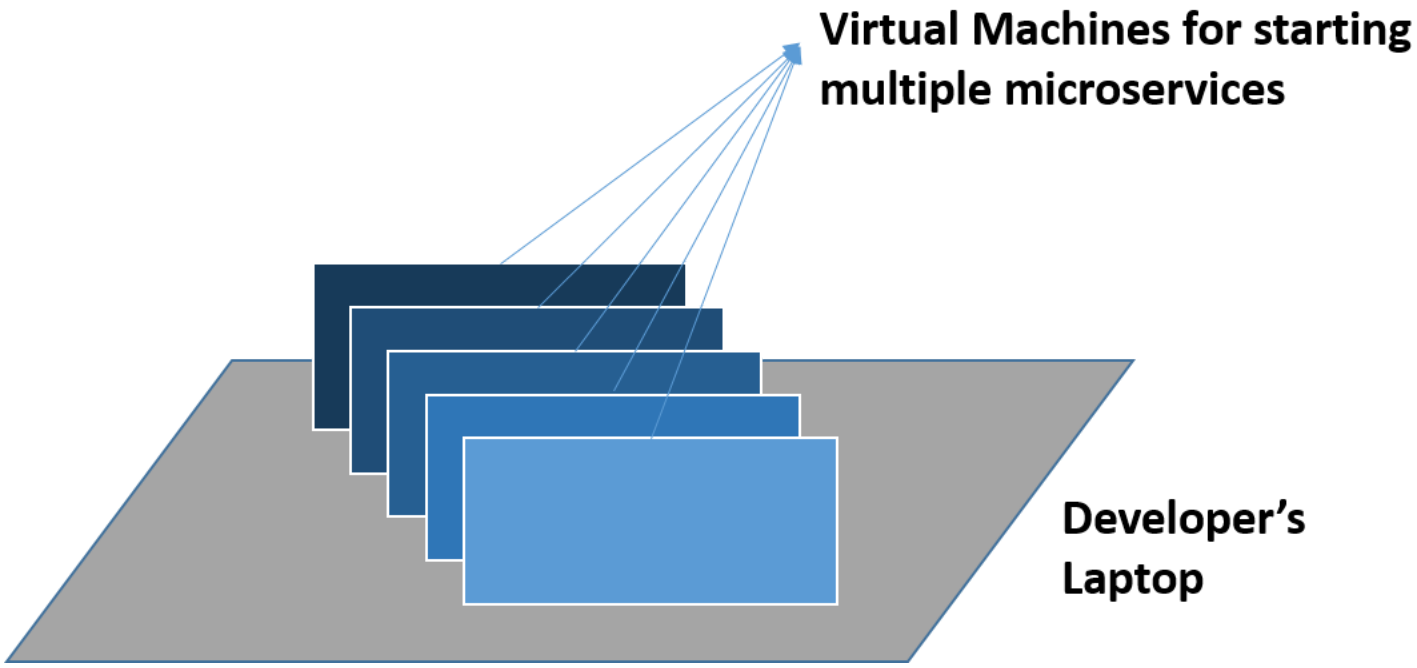
Well, this architecture has a lot of benefits:

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

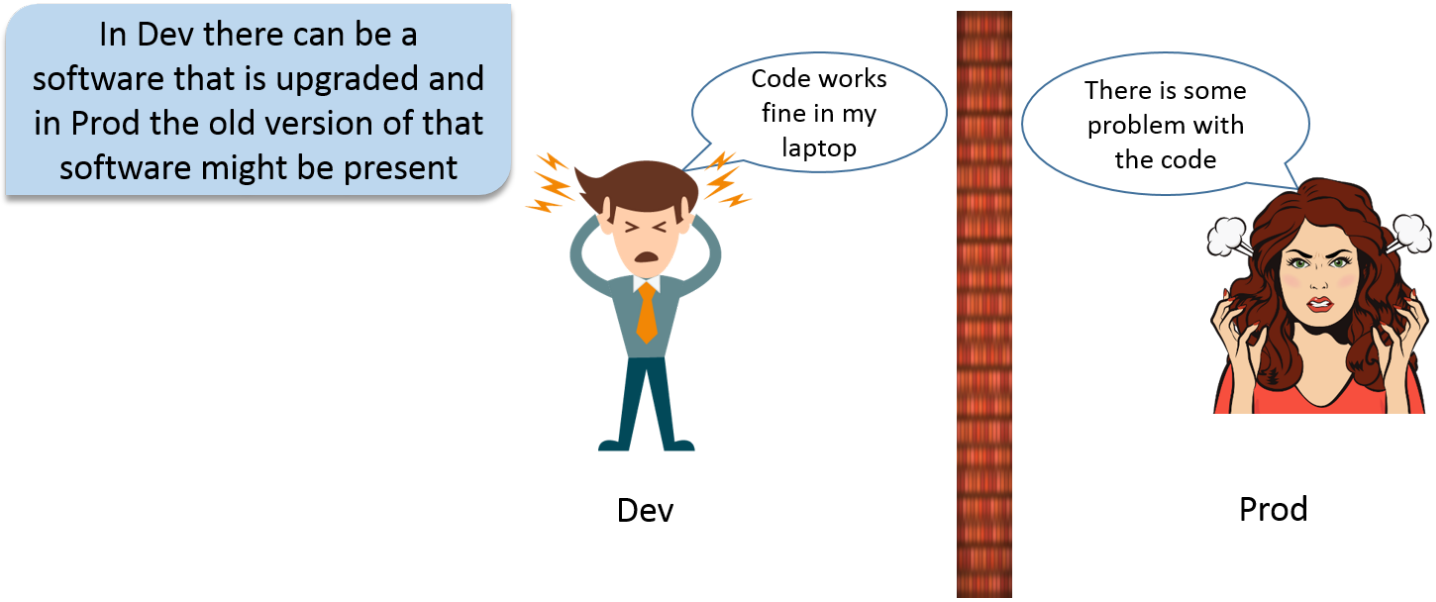
There are many other benefits as well, I won't go into much detail about microservices in this post. But, soon I will be coming up with a couple of blogs on microservices as well.

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.

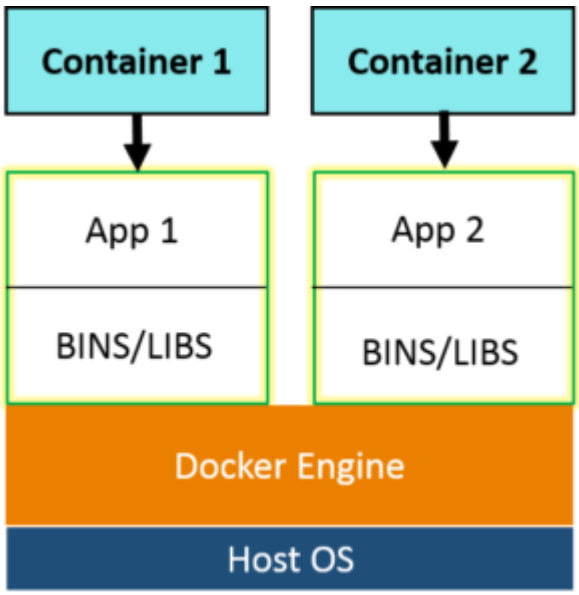
What if I am using 100 VMs, then configuring each VM will take a lot of time, and at the same time it is prone to error as well.

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.

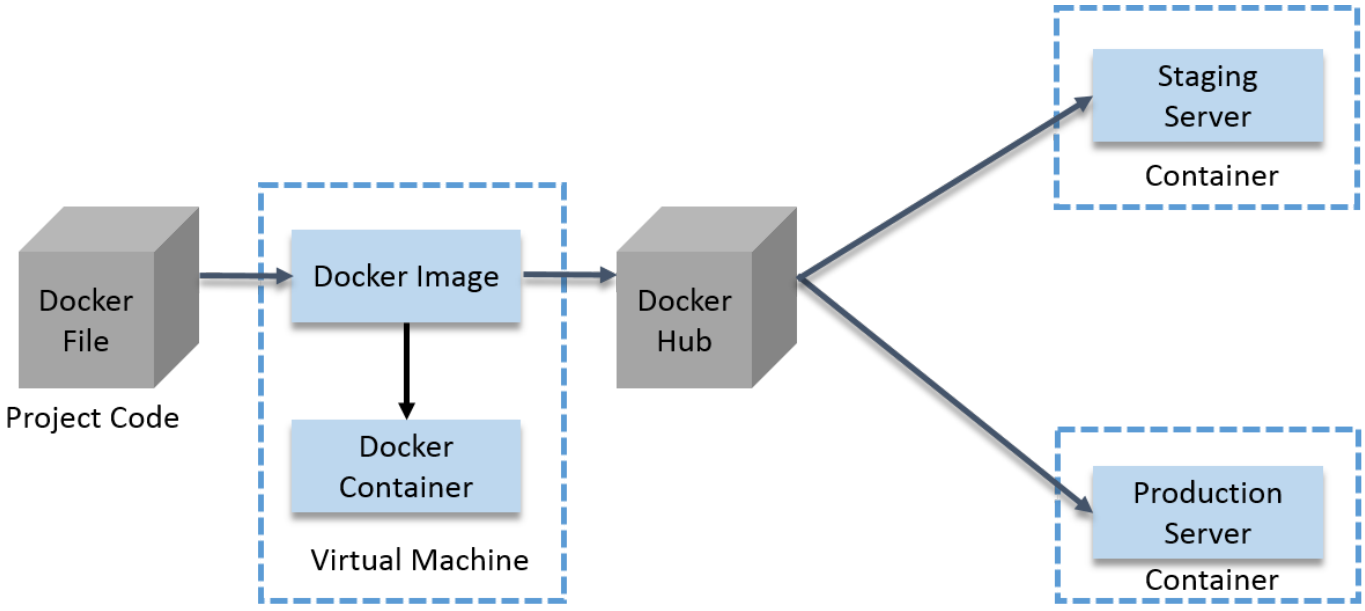


Docker Containers are a lightweight solution to Virtual Machines, and it uses the host OS. The best part, you don't have to pre-allocate any RAM to the Docker Container, it will take it as and when required. So, with Docker Container I don't have to worry about wastage of resources.

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.



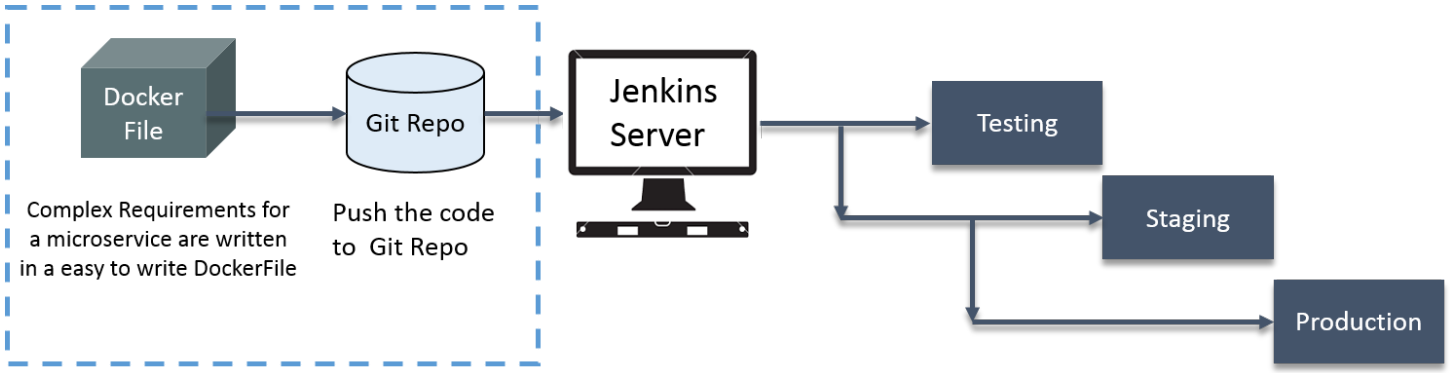
Guys, I know the diagram looks a bit complex, but trust me it ain't that complex. Below is the explanation of the diagram, even after that you feel it is tough to understand, you can comment your doubt, I will address those questions ASAP.

- 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.

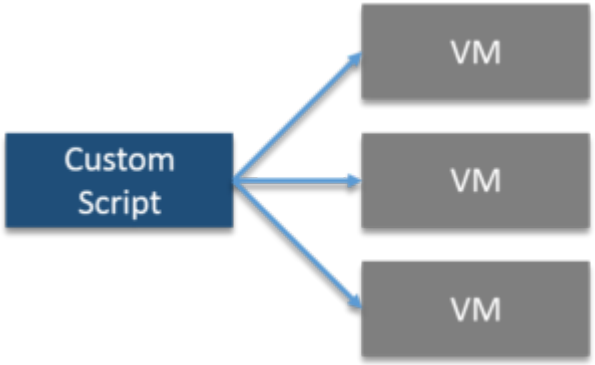
Well, that was the story of my company, let's look at the case-study of Indiana University. How Docker solved their problems.

Indiana University Case-Study:

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

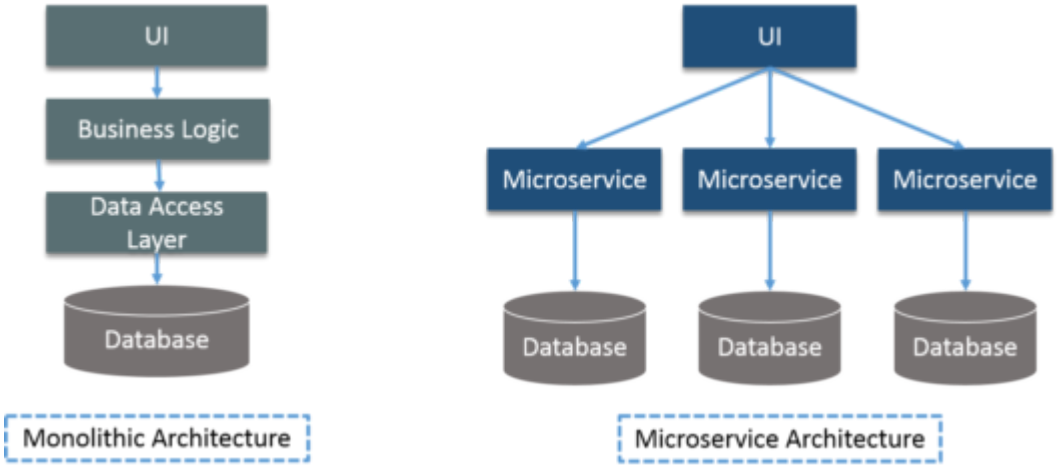
Problem Statement

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.

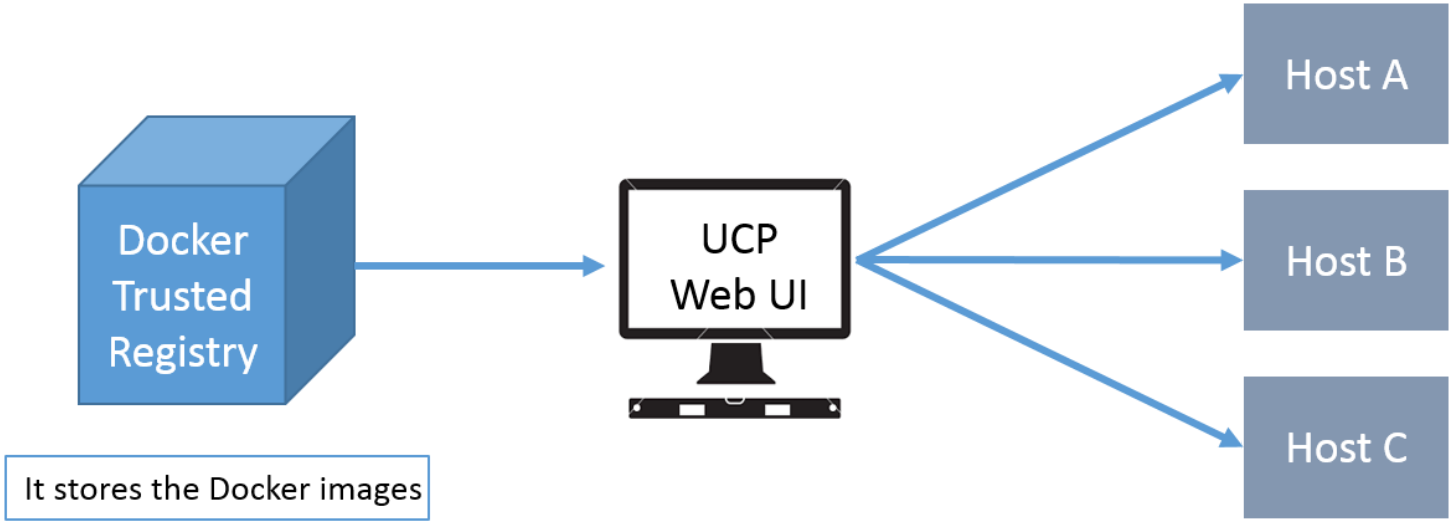
The University wanted to improve the way they architect applications, by moving to a microservices based architecture for their applications.



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.

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