Install Docker

* Make sure my existing packages are up to date ( sudo yum update)
* run a docker installation script (curl -fsSL https://get.docker.com | sh) this script adds the docker.repo repository and installs docker
* sudo service docker start

Run a docker container

Pull a docker image for ubuntu operating system.

Docker images are used to create containers. If the image is not present locally docker will pull the image from registry.hub.docker.com.

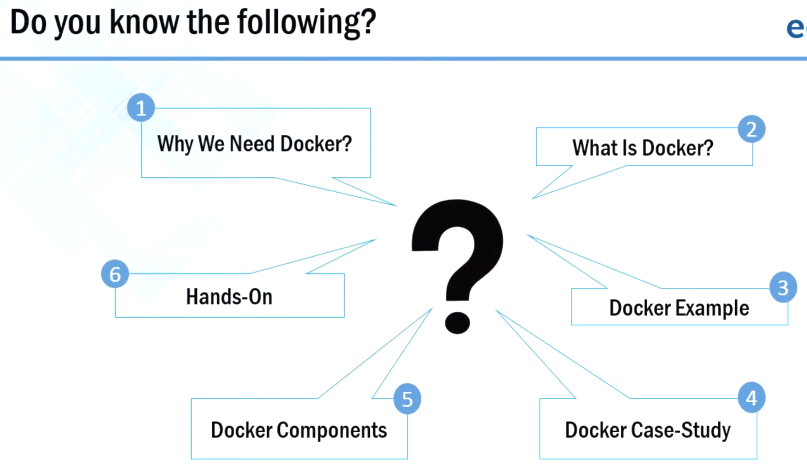
Sudo docker run Ubuntu (Imagename)

Unable to find an image locally =>pulling from registry.hub.docker.com.

Agenda

Start using container 🡪 sudo darker run -it Ubuntu(Imagename)

|  |
| --- |
| 1. why we need docker 2. will focus on various problems that industries were facing before docker was introduced 3. Will understand what exactly docker is ..with an example 4. how industries are using docker with a case study of Indiana University 5. various docket components like images containers etc 6. hands-on part: will focus on installing WordPress and PHPmyadmin using docker compose |







**First problem:** The most common problem that industries were facing:

There is a developer who was build an application, that works fine in his own environment but when it reach production.. There were certain issues with that application.

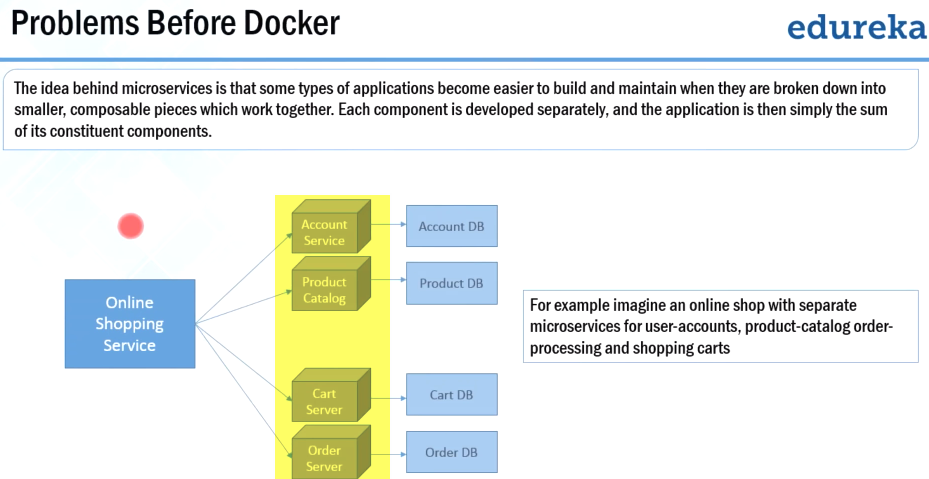
Why does that happen?? Because of difference is the computing environment between dev and product.

Second problem: before we proceed with the second problem it is very important for us to understand microservices.

Microservices & Advantages

micro services : consider a very large application, that application is broken down into smaller services each of those services can be termed as micro services (or) micro services can consider that small processes that communicates with each other over a network to fulfill one particular goal..

Example: you can see that there is an online shopping service application. it can be broken down into smaller microservices like a account service product catalog, card server and order server.



Micro service architecture is gaining a lot of popularity nowadays.., even giants like Facebook and Amazon are adopting micro service architecture.

There are three major reasons (advantages) for adopting micro service architecture

1. there are certain applications which are easier to build and maintain when they are **broken down into smaller** pieces or smaller services
2. suppose if I want to update a particular software or I want a new technology stack in one of my module on one of my service so I can easily do that because the **dependency** concerns will be very less when compared to the application as a whole
3. If any of my module or any of my **service goes down** then my whole application remains largely unaffected.

**Problems in adopting this micro service architecture (2nd problem) :**

One way of implementing microservice architecture:



You can see that there is a host machine and on top of that host machine there are multiple virtual machines.

Each of these virtual machines contains the dependencies for one micro service.. so you must be thinking what is the disadvantage here the

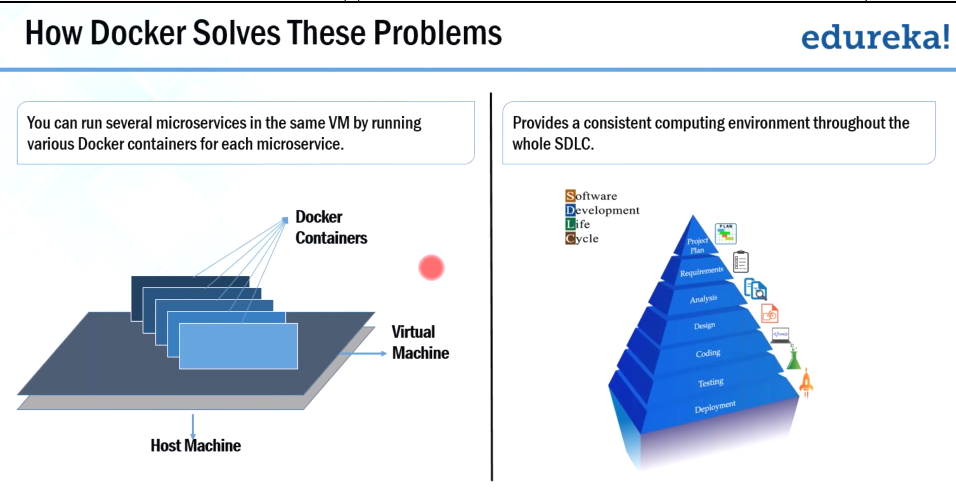
**Major disadvantage:** here is in virtual machines there is a lot of wastage of resources such as ram processor disk space are not utilized completely by the micro service which is running in these virtual machines..

**So it is not an ideal way to implement micro service architecture.**

**Given an example of 5 micro services what if there are more than 5 micro services what if your application is so huge that it requires 50 micro services ---> 50 VMS.. so at that time using virtual machines doesn't make sense because of the wastage of resources .**

**micro service problem--Solution**

**first discuss the implementation of micro service problem that we just saw:**



so what is happening here there's a host machine and on top of that host machine there's a virtual machine and on top of that virtual machine there are multiple docker containers and each of these docker containers contains the dependencies for one micro service …

so you must be thinking what is the difference **here earlier we were using virtual machines now we are using were docker containers on top of virtual machines**

**l**et me tell you guys docker containers: are actually lightweight alternatives of virtual machines..

what does that mean in docker containers you don't need to pre allocate any ram or any disk space so it will take the RAM and disk space according to the requirements of applications all right okay ..

why are we using virtual machine here very good question???

1. first the host machine has to be a Linux or a unix based machine in order to run docker containers..

docker containers does not work on windows system.., so if you have a windows system you require linux or UNIX virtual machine then only you can run docker containers..

1. 2nd reason for using docker containers when a virtual machine is that: you want to segregate it from the rest of the host machine you want to encapsulate that separately in a virtual machine because..

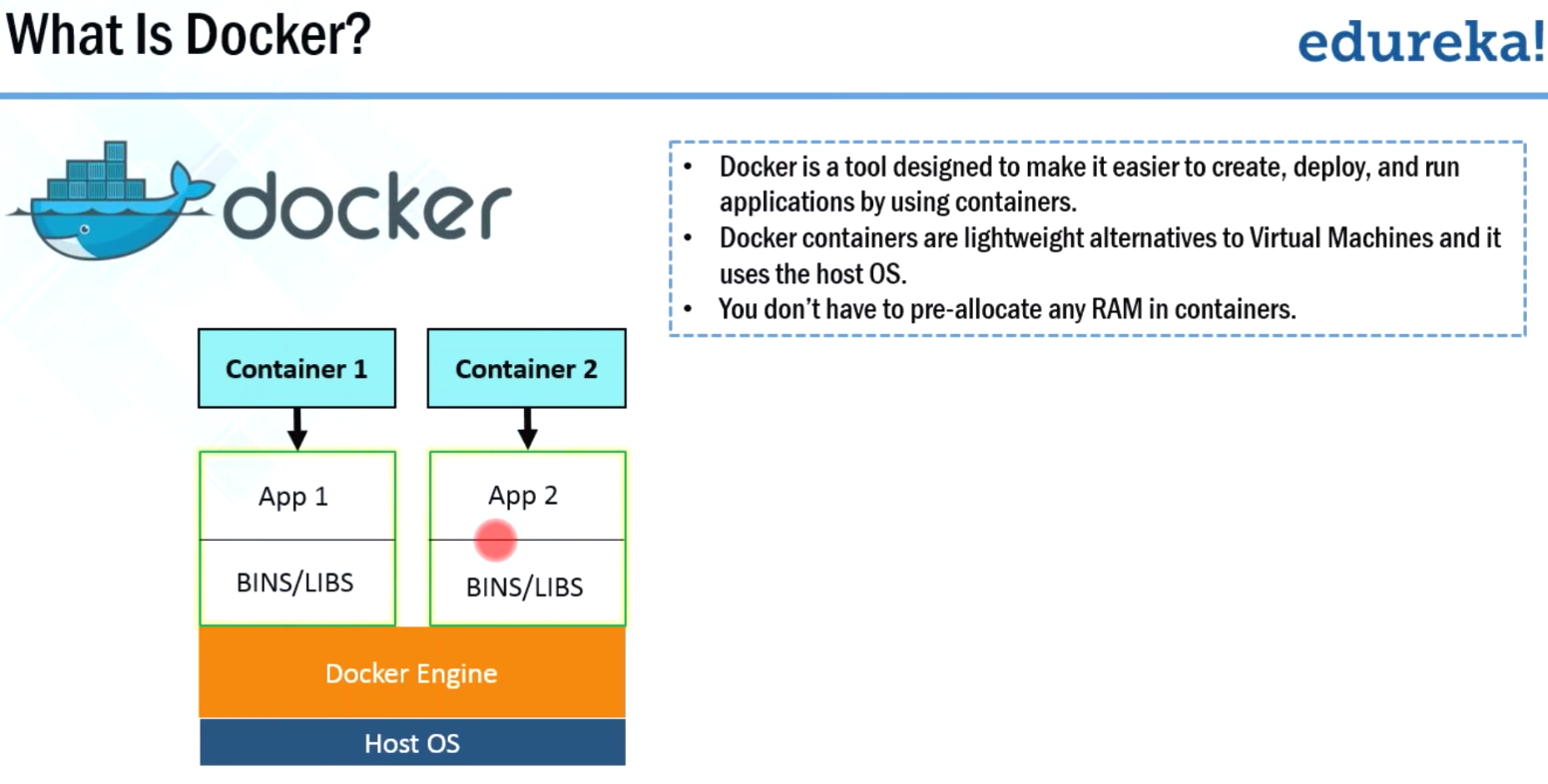
in your host machine suppose you have around 64 gb of ram and around a 20 TB of hard disk and all of your microservices combined will not use more than 20 gb of ram and 200 gb of harddisk.. so there's no point in running docker containers on the host machine.

so I can configure a virtual machine that provides 20 gb of ram and exactly 200 gb of hDD and I can run docker containers on that virtual machine ….

How Docker solves the problem of not having a consistent computing environment throughout the software delivery lifecycle

let me tell you first of all ..docker containers are actually developed by the developers so now let us see how Docker solve the first problem that we saw where an application works fine development environment but not in production ..

so docker containers can be used throughout the SCLC lifecycle in order to provide consistent computing environment.. so the same environment will be present in dev tests and product ..so there won't be any difference in the computing environment..





1.) Docker containers does not use the guest operating system.it uses the host operating system..

2.)On diagram 🡪host operating system 🡪 on top of that host operating system there's a Docker engine and🡪 with the help of this docker engine docker containers are formed and 🡪these containers have applications running in them and the requirements for those applications such as all the binaries and libraries are also packaged in the same container ..

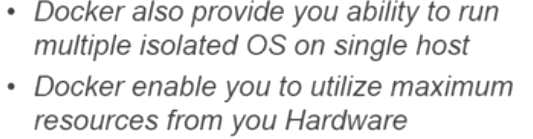
3.)There can be multiple containers running ..

You can see that there are two containers here.. one and two.

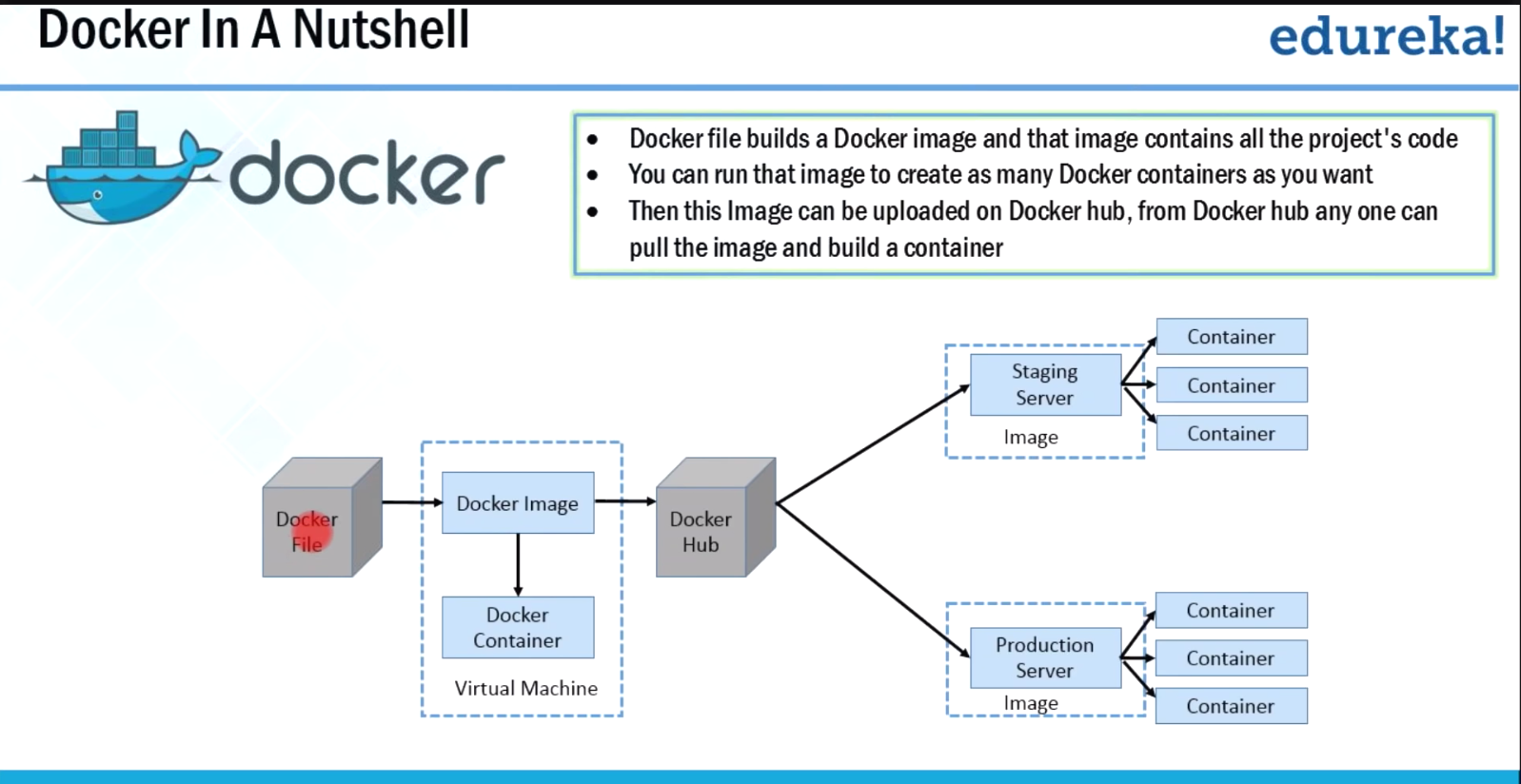
1. On top of the host machine🡪 there's a Docker engine and🡪 on top of the DOcker engine there are multiple containers 🡪On each of those containers will have an application running in them and whatever the binaries and library is required for that application is also packaged in the same container ..

General workflow of Docker or one way of using Docker

Allows to create staging env. Similar to prod..



**Virtualization: Dividing physical hardware logically/virtually**

 1.) Developer writes a code that defines an application requirements or the dependencies in an easy to write docker file.

2.)this docker file produces docker images.. so whatever dependencies are required for a particular application is present inside this image ..

3.) what are darker containers: docker containers are nothing but the runtime instance of docker image.

4.) This particular image is uploaded onto the docker hub now ..

what is darker hub: docker hub is nothing but a git repository for docker images.., it contains public as well as private repositories.. So from public repositories you can pull your image as well and you can upload your own images as well on to the docker hub ..

docker hub is basically a cloud hosted service provided by docker over here.., you can upload your own images like you can write your own image and upload that onto the docker hub and at the same time .., you can even pull the images that are present in the public .So it is just like a git repository for docker images.

5.) from docker hub, various teams such as QA or **production team will pull the image and prepare their own containers** ..

**What is the major advantage we get through this workflow**??

whatever the dependencies that are required for your application is actually present throughout the software delivery life..

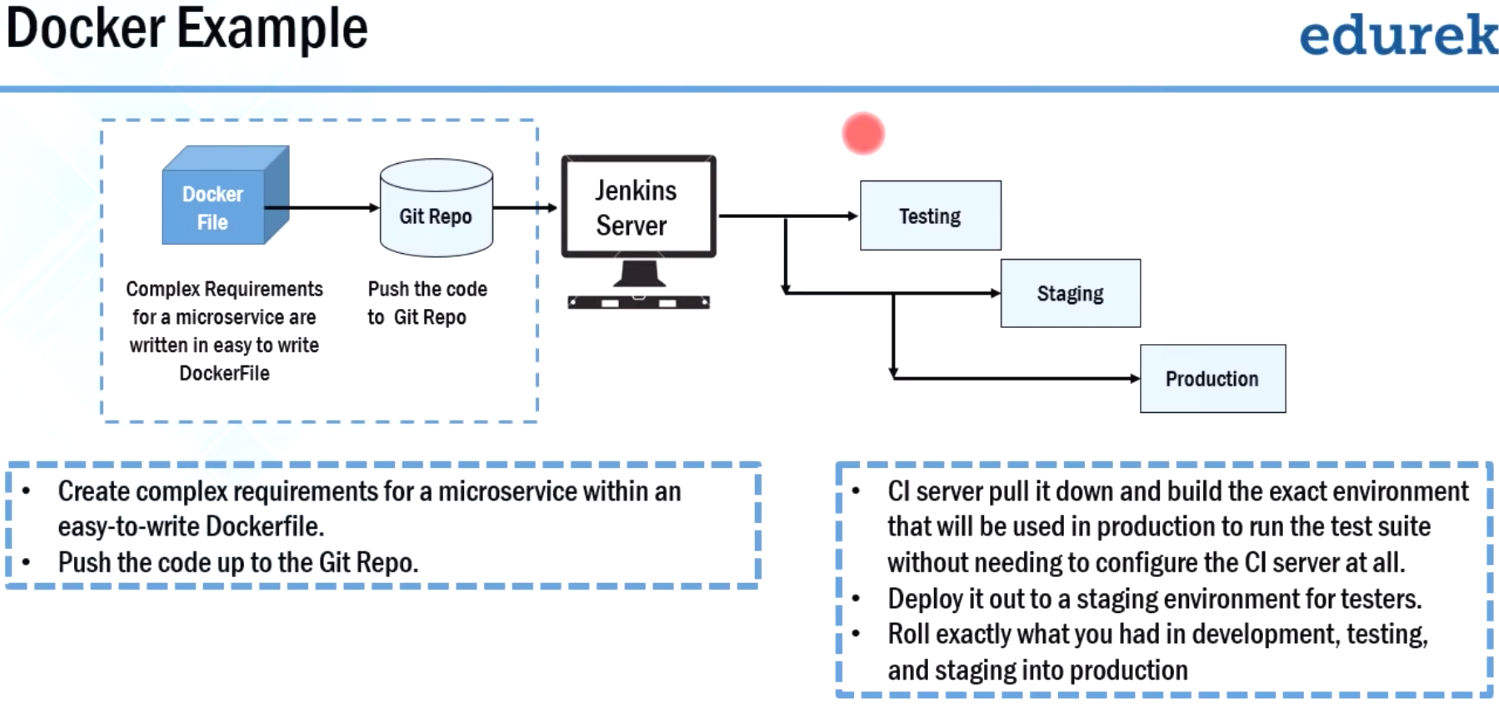
if you can recall the first problem that we saw that an application works fine in development environment but when it reaches production it is not working properly ..,**so that particular problem is easily resolved with the help of this particular workflow ..,because you have a same environment throughout the software delivery lifecycle be a dev test or product.**

Another way of using docker

In the previous example.., we saw that docker images were used and those images were uploaded onto the docker hub.., and from dockerhub various teams were pulling those images and building their own containers.

**But docker images are huge in size and requires a lot of network bandwidth.**

**So in order to save that network bandwidth, we use this kind of a workflow.**



1. we use Jenkins servers or any continuous integration server to build an environment that contains all the dependencies for a particular application or a micro service ..
2. And that build environment is deployed on to various streams like testing staging and production.

what exactly is happening in this particular image??

Over here developer has written complex requirements for a micro service in an easy to write docker file.., and the code is then🡪pushed on to the Git repository 🡪from git repo, continuous integration server like Jenkins will pull that code and build an environment that contains all the app dependencies for that particular microservice ..

and that environment is deployed on to testing staging and production.

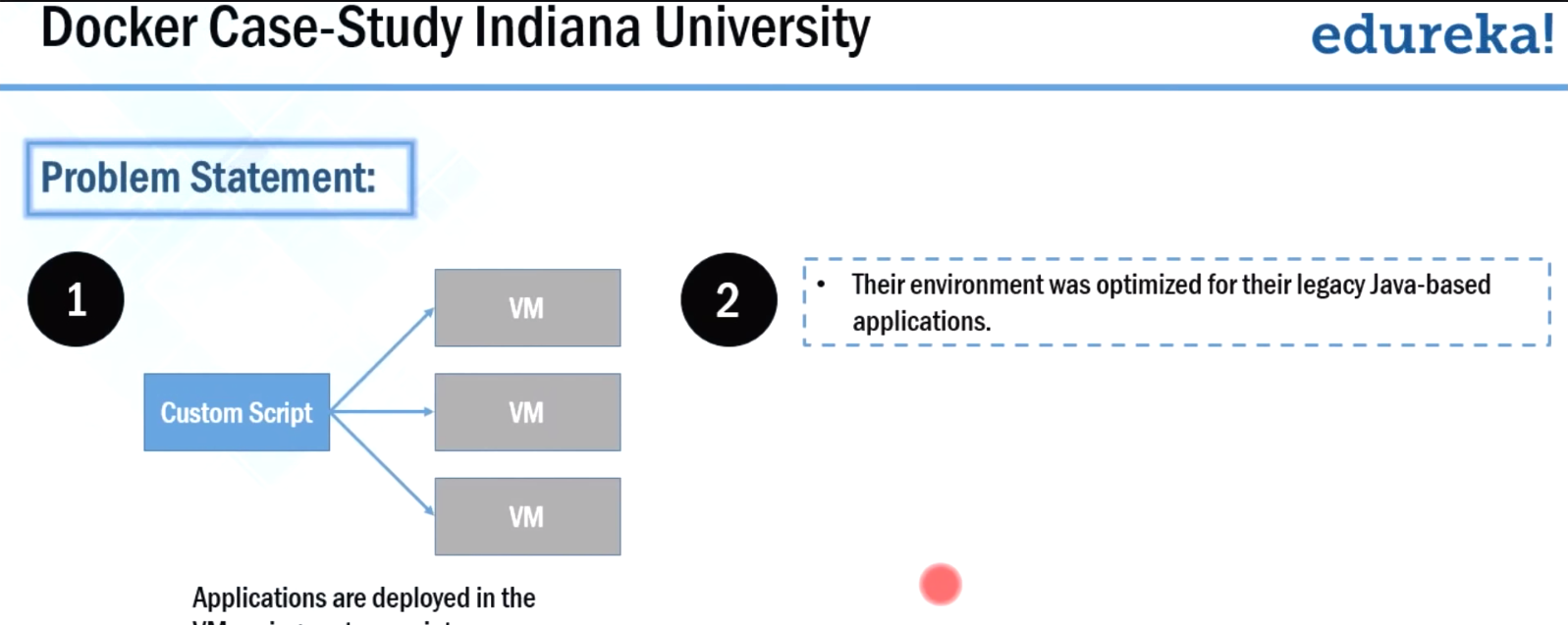
So in this way, whatever requirements are there for your micro service is present throughout the software delivery lifecycle..

The first problem where application works fine in dev but does not work in prod

1. with this workflow we can completely remove that problem.., because the requirements for the micro service is present throughout the software delivery lifecycle .and

2.) This image also explains how easy it is to implement the micro service architecture using docker.

How industries are adopting docker.., so this is the case study of Indiana University

**Before docker:** they were facing many problems.

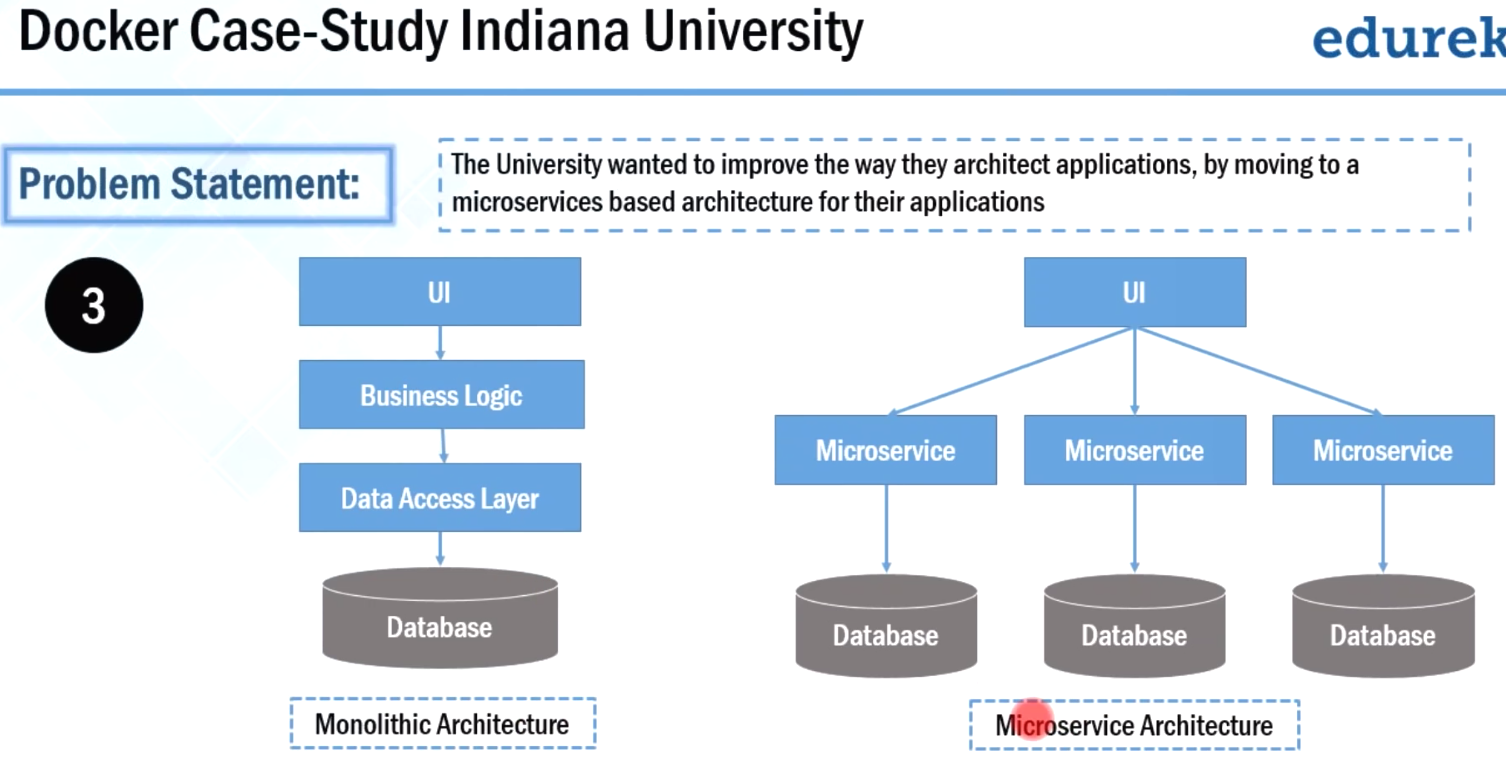
**Pb-1:** they were using custom script in order to deploy that application onto various VMs.

So this requires a lot of manual steps.

**Pb-2:** their environment was optimized for legacy java based applications.. But they're growing environment involves new products that aren't so lowly java-based .

So in order to provide the students the best possible experience they needed to begin modernizing their applications.

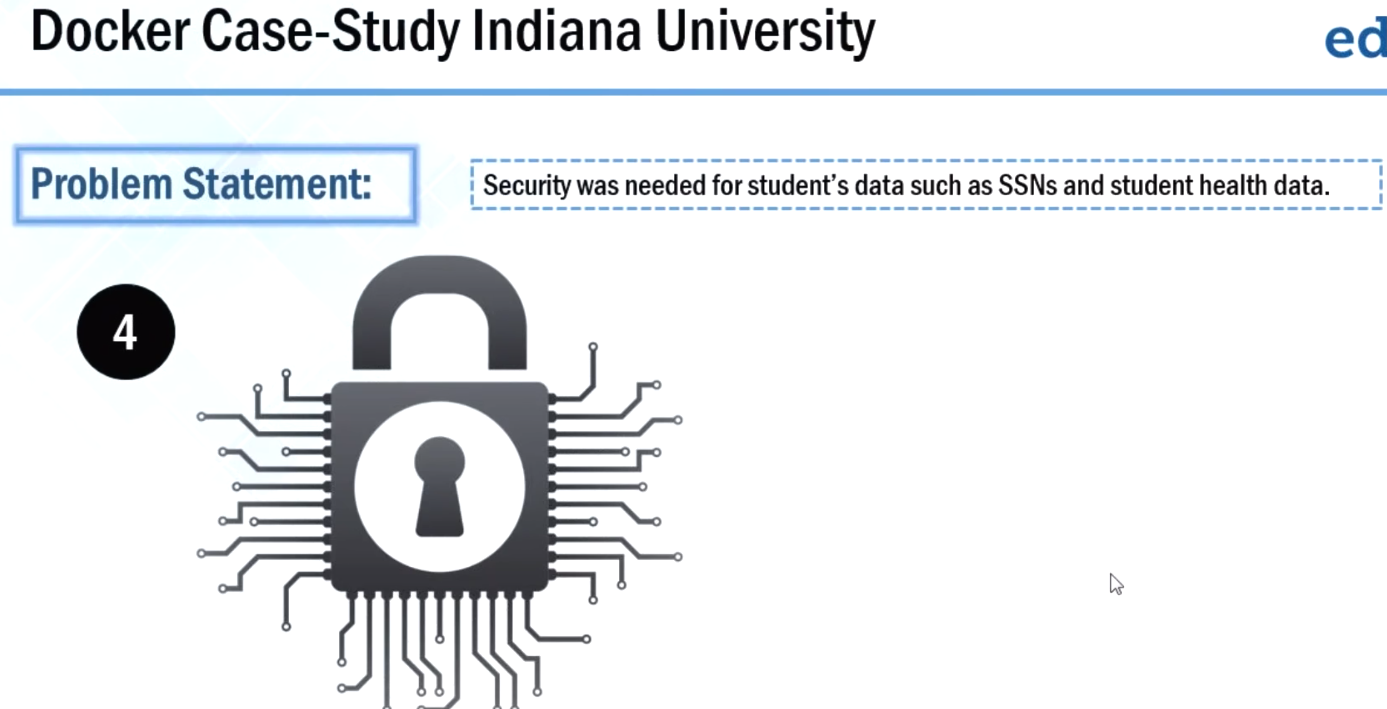
**Pb-3:** Start modernizing their applications 🡪move from a monolithic architecture to a micro service architecture.



And the previous slides we also saw that if you want to update a particular technology in one of your micro service.., it is easy to do that because there will be very less dependency constraints when compared to the whole application

So because of that reason they wanted to start modernizing their application they wanted to move to a micro service architecture.

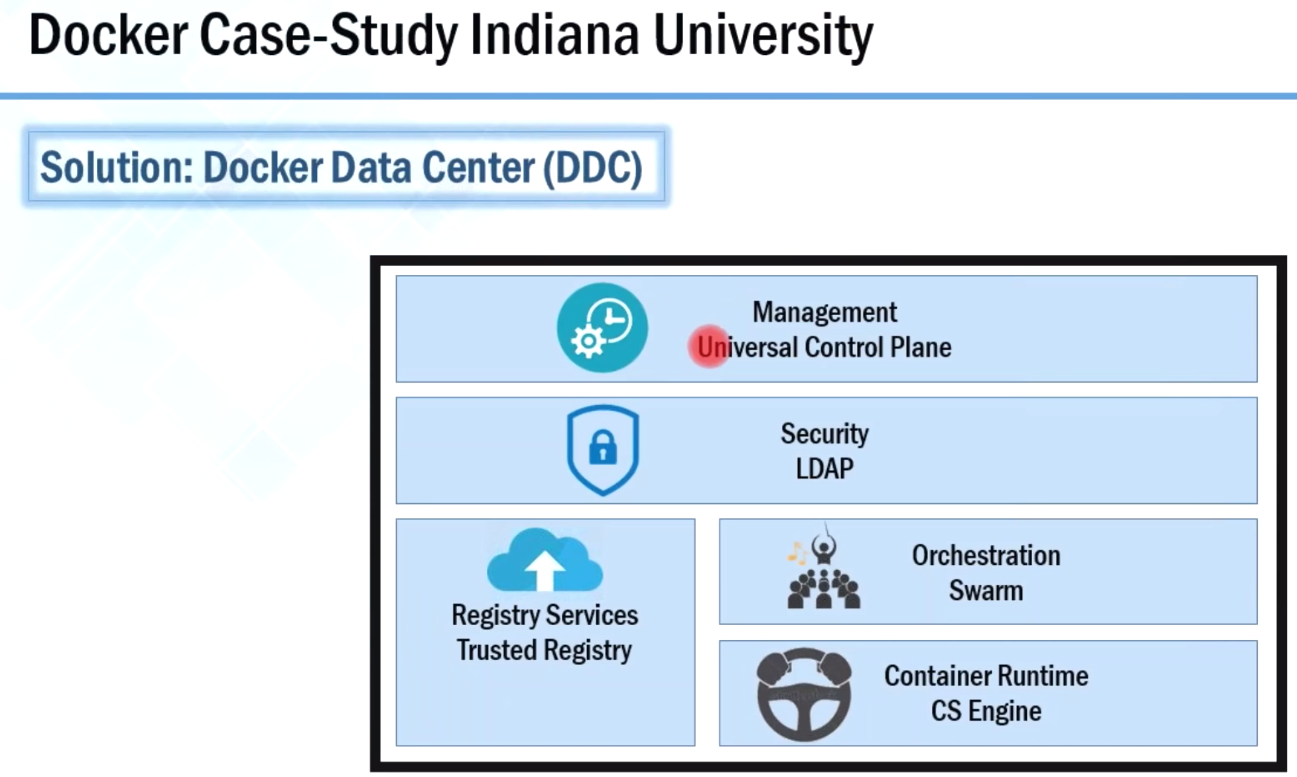
**Pb-4:**

Indiana University also needed security for the sensitive student data such as SSN and student health care data .

These are four major problems that they were facing before Docker.

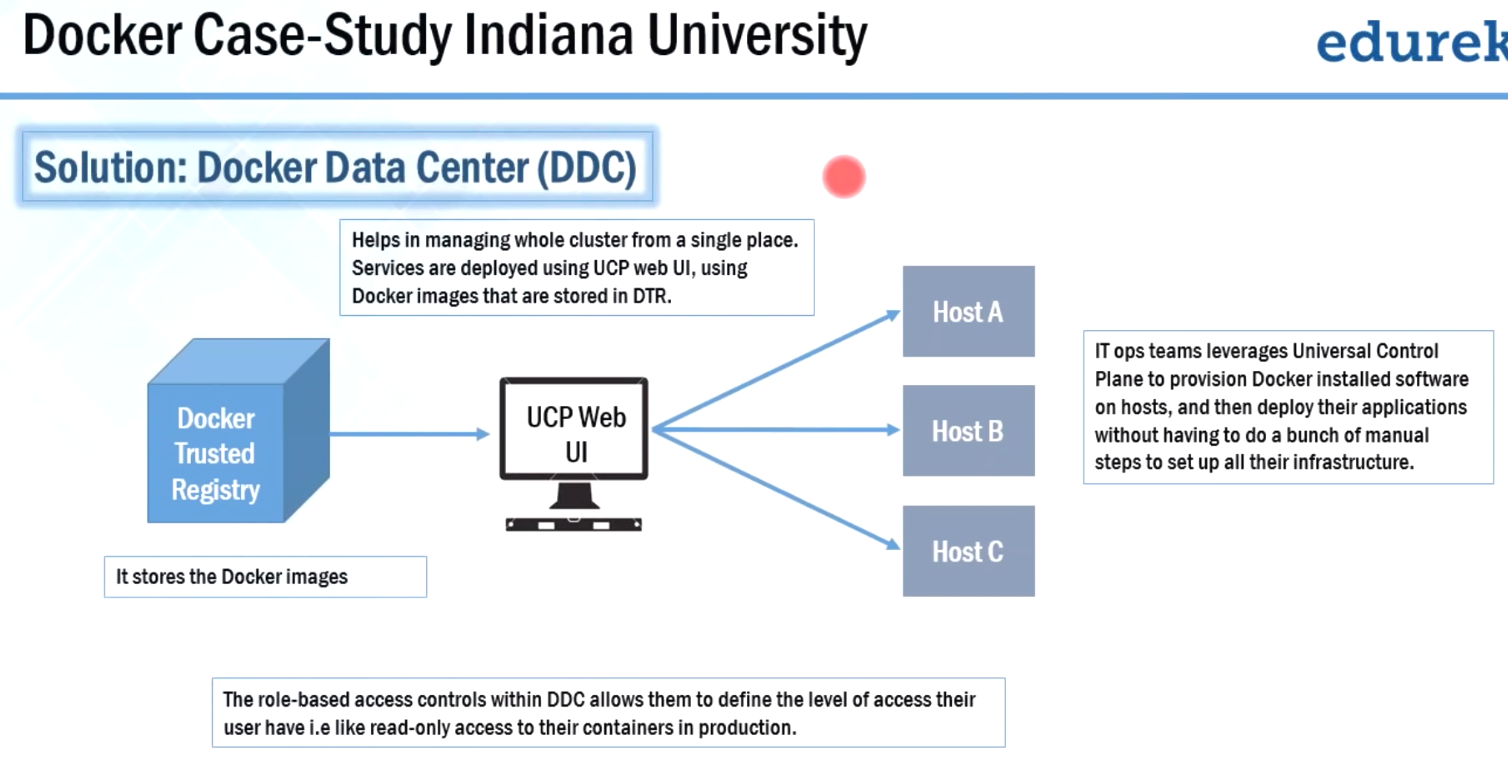
Now let us see how they have implemented Docker to solve all these problems.

**The solution** to all these problems was Docker data center (DDC) and Docker data center has various components which are there in front of your screen..



First is universal control plane, then comes ldap, swarm, CS engine and finally darker trusted registry.

Now let us move forward and see how they have implemented docker data center in their infrastructure.

 this This is a workflow of how Indiana University has adopted DDC.

1.)This is Docker trusted registry .It is nothing but the storage of all your docker images and each of those images contains the dependencies for one micro service.

we saw IU, wanted to move from a monolithic architecture to a micro service architecture .

So, because of that reason these darker images contain the dependencies for one particular microservice but not the whole application.

1. Universal control plane.

It is used to deploy services onto various hosts with the help of docker images that are stored in the darker trusted registry.

So IT ops team can manage their entire infrastructure from one single place with the help of universal control plane web user interface.

They can actually use it to provision Docker installed software on various hosts.., and then deploy application without doing a lot of manual steps.

Previous slides: that Indiana University was earlier using custom scripts to deploy application on two VMs, that requires a lot of manual steps.

That problem is completely removed.

1. When we talk about security the role based access controls within the docker data center allowed Indiana University to define a level of access to various Teams.

For example: they can provide read-only access to docker containers for production team and at the same time they can actually provide read and write access to the dev team

This is how Indiana University has adopted DDC to their questions.

**Q:** what is the difference between docker containers and virtual machines?

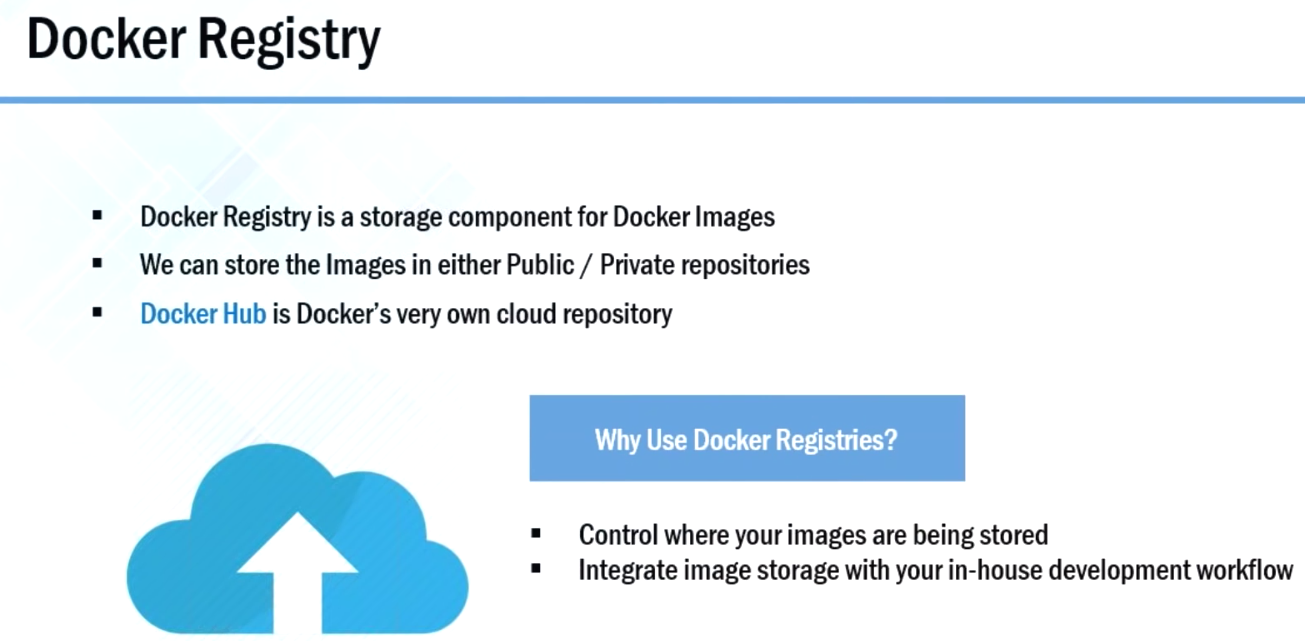
1. Docker containers are nothing but the lightweight alternatives of virtual machines.
2. Docker containers do not don't have their own operating system. They sit on top of the host operating system or you can say that they uses the host operating system.
3. in docker containers you don't need to pre allocate any ram. It takes a ram accordingly but in virtual machines you have to pre allocate a certain amount of RAM.
4. When we talk about runtime, Docker containers have very less run time because you don't need to boot the OS…, whereas in virtual machine it has a run time which is greater than docker containers because you need to boot the OS.

These are the three major difference .

Docker

Components

Docker registry:

 is nothing but the storage of all your darker images.Your images can be stored either in public repositories or in private repositories.

Docker Registry

These repositories can be present locally or it can be present on the cloud.

Docker provides a cloud hosted service called Docker hub.

docker hub as public as well as private repositories.

For public repositories: you can actually pull an image and prepare your own containers at the same time.., you can write an image and upload that onto the docker hub .

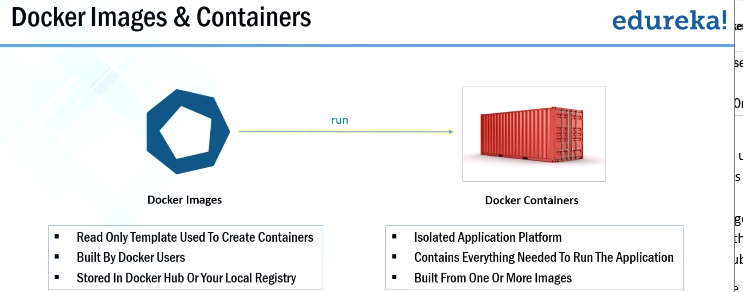
You can upload that into your private repository or you can upload that on a public repository.



Create repository: where you can create your own public or private repositories and upload images and at the same time there is an option called

Explore repositories: this contains all the repositories which are available publicly.

Docker images



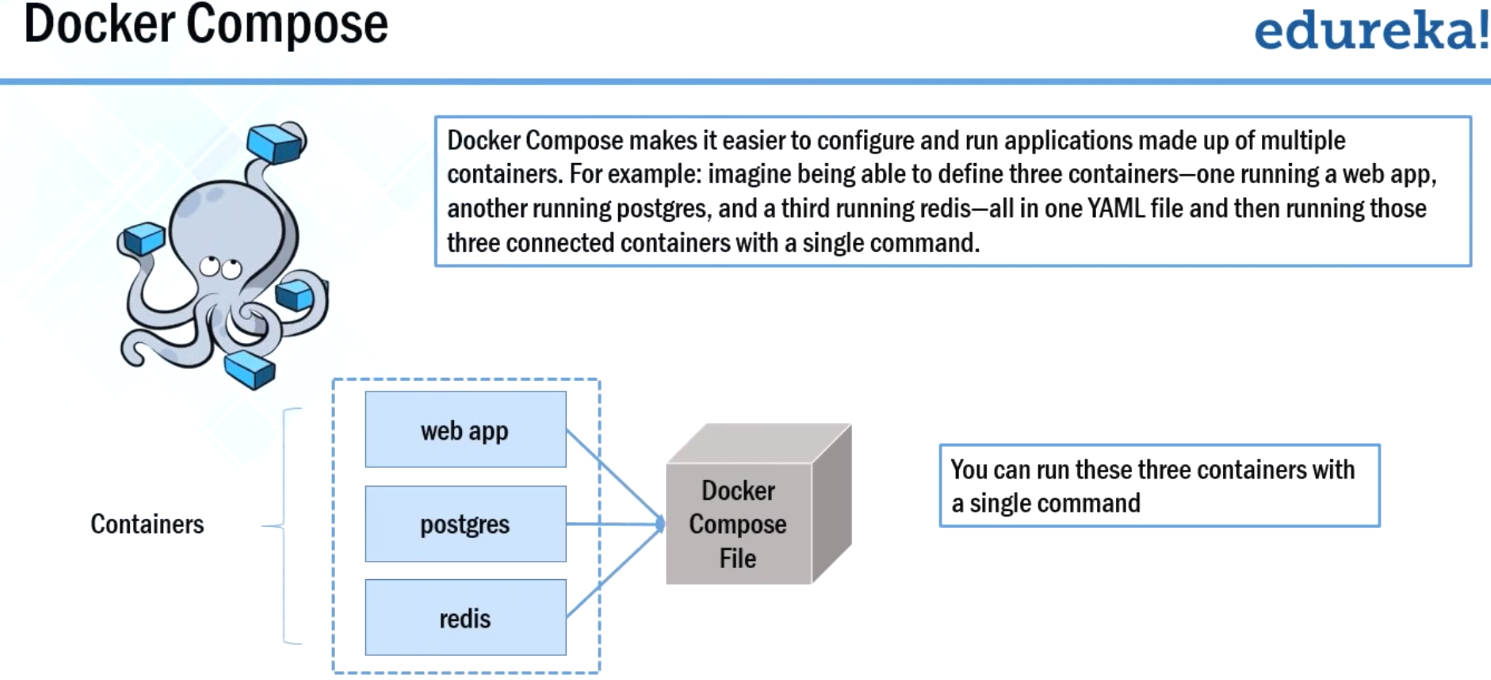
1. Read-only templates that are used to create containers.
2. These Docker images contains all the dependencies for a particular application or a micro service.
3. You can create your own image and upload that onto the Dockerhub and at the same time, you can also pull the images which are available in the public repositories and the entire hub.

Docker containers

1. Docker containers are nothing but the runtime instances of docker images.
2. It contains everything that is required to run an application or a micro service and at the same time it is also possible that .., more than one image is required to create one container.

1st we installed Docker on ubuntu, after that we pulled CentOS image from docker hub and then we build a CentOS container using that CentOS image.

Docker Compose



Suppose you have multiple applications on various containers and all those containers are actually linked together. So **you don't want to actually execute each of those containers one by one but you want to run those containers at once with a single command**

1. With docker compose, you can actually run multiple applications present on various containers with one single command.

Docker -compose up

1. **EX:** imagine you are able to define three containers one running a webapp, another running Postgres, and another running a redis in a YAML file.. That is called Docker compose file.
2. And from there you can actually execute all these three containers with one single command…. that is darker hyphen compose up.

**EX:** suppose you want to publish a blog for that you'll use CMS ( **content management system** )and WordPress is one of the most widely used CMS..

So you need one container for (1)WordPress and you need one more container for (2)MYSQL as back end. And that MYSQL container should be linked with WordPress container.

Apart from that, you need one more container for (3 )PHPmyadmin that should be linked to MYSQL database…, as it is used to access MYSQL database.

**So what??? if you are able to define all these three containers in one YAML file and with one command that is Docker -compose up all three containers are up and running.**

CentOS box-> install docker-> pull CentOS image

Docker-Compose install

sudo yum update

sudo yum install -y python-pip

**pip** is a package management system used to install and manage software packages written in **Python**.

sudo pip install docker-compose

Now I'll make a directory named it as WordPress, mkdir WordPress ..now i'll enter this WordPress directory

mkdir wordpress

cd wordpress/

create new file-> sudo gedit docker-compose.yml

|  |
| --- |
| wordpress:  images: wordpress  links:  - wordpress\_db:mysql  ports:  - 8080:80  wordpress\_db:  image: mariadb  environment:  MYSQL\_ROOT\_PASSWORD: edureka  phpmyadmin:  image: corbinu/docker-phpmyadmin  links:  - wordpress\_db:mysql  ports:  - 8181:80  environemnt:  MYSQL\_USERNAME: root  MYSQL\_ROOT\_PASSWORD: edureka |
| wordpress:-🡪 Defined a container, and named it wordpress  images: wordpress🡺 Build from image wordpress, present on dockerhub.  (But this WordPress image does not have a database ..,so for that i have defined one more container and i have named it as WordPress\_DB.  It is actually built from the image that is called a Mario DB, which is present in the WordPress  ).  links:  - wordpress\_db:mysql🡪 I need to link this WordPress\_DB with the WordPress container.So for that , i have written links wordpress\_DB:mysql  ports:  - 8080:80 🡪 this port 80 of the docker container will actually be linked to port 8080 of my host machine  wordpress\_db:  image: mariadb  environment:  MYSQL\_ROOT\_PASSWORD: edureka🡪 defined a password here, edureka  phpmyadmin:  image: corbinu/docker-phpmyadmin🡪 I've defined one more container called PHPmyadmin. This container is built from the image corbinu /docker-phpmyadmin, that is present on the docker hub  links:  - wordpress\_db:mysql🡪 Again, I need to link this particular container with WordPress\_DB container.I have written links wordpress \_DB:mysql  ports:  - 8181:80🡪 port 80 of my docker container will actually be linked to put 8181 of the host machine  environemnt:  MYSQL\_USERNAME: root🡪 finally i have given a username that is root  MYSQL\_ROOT\_PASSWORD: edureka🡪 password as Edureka |

1. i have defined a container by the name wordpress, it is built from an image called WordPress that is present in the docker hub
2. This image present in dockerhub does not have a database ..,so i need to define one more container that contains a database.

So WordPress\_DB is the name of that container and it is built from an image called Mario DB which is present on the docker hub

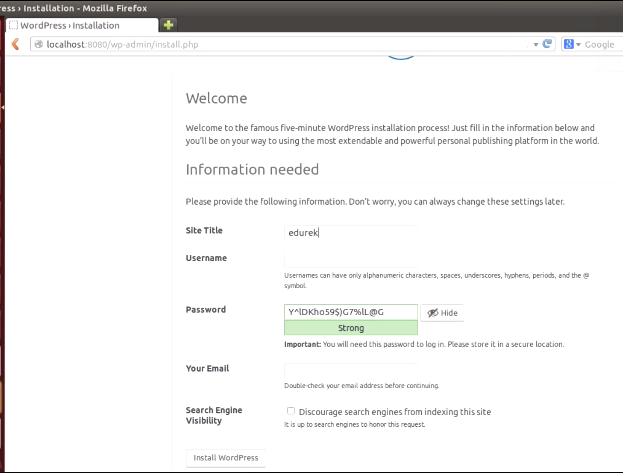
1. I need to link this database with WordPress ..So i'll write links wordpress\_DB colon:mysql
2. Port section-> the port 80 of the docker container will be linked to port 8080 of the host machine
3. After that, we require one more container and I've named it as PHP myadmin ..it is built from the image called corbinu /docker-PHPmy admin.
4. I need to link this particular container with WordPress\_DB :mysql. **Mysql is actually the name that I have given**
5. in the port section, the port 80 of the container is actually linked to put 8181 of the host machine.
6. and finally i have given user name and password.

Sudo Docker-compose up -d

this command will actually pull all the three images and we'll build the three containers ..

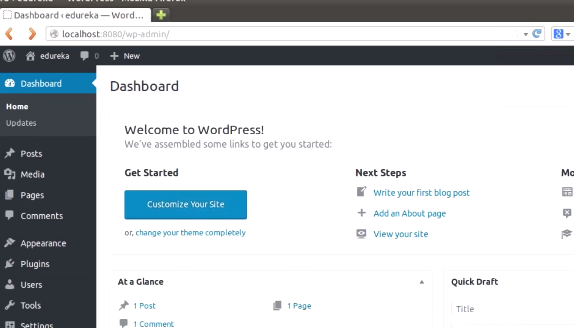
open my browser -> IP address /hostname

localhost:8080.., that I have given for WordPress . So it will direct you to a wordpress installation page.

 over here, fill this particular form which is asking you for :

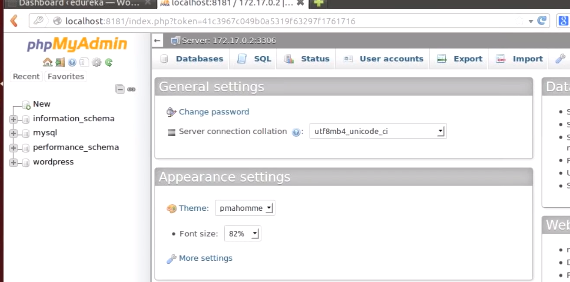
site title->Edureka, username ,password,email

wordpress dashboard and wordpress is now successfully installed.

 On new tab -> localhost:8181 for PHPmyadmin

Enter user,password..,as given YAML file

PHPmyadmin is successfully installed.



this PHPmyadmin is actually used to access a mysql database.

This mysql database is used as back-end for WordPress.