Max and Kandice rough speech script.

Why we need docker. (the problems)
the most common problem that industries were facing. As you can see that there is a developer who has build an application that works fine in his own environment. but when it reads production there were certain issues with that application. why does that happen that happens because of difference is the computing environment between dev and product.

(Next Slide)(online shopping service)
the second problem it is very important for us to understand what are 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 we can put it in an another way as well. micro services can consider that small processes that communicates with each other over a network to fulfill one particular goal. let us understand this with an example as you can see that there is an online shopping service application. it can be broken down into smaller micro services like account service, product catalog, card server, and orders over. micro service architecture is gaining a lot of popularity nowadays even giants like Facebook and Amazon are adopting micro service architecture there are three

three major advantages of using micro architecture first there are certain applications which are easier to build and maintain when they are broken down into smaller pieces or smaller services. second reason is 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. apart from that. the third reason is if any of my module or any of my service goes down then my whole application remains largely unaffected.

(slide -developing and application requires starting several)
what are the problems in adopting this micro service architecture so this is one way of implementing micro service architecture over here as 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. 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. I have just given an example of five micro services what if there are more than five micro services what if your application is so huge that it requires 50 micro services. so at that time using virtual machines does not make sense because of the wastage of resources.

(slide - Pyramid SDLC Slide)
Let us 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 is 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. let 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.

why are we using virtual machine? the host machine has to be aligned X 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 a Linus or UNIX virtual machine then only you can run docker containers and one more 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 micro services combined will not use more than 20 GB of RAM and 200 GB of hard disk. 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 hardness and I can run docker containers on that virtual machine.

Now let us see 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 test and product so there won't be any difference in the computing environment.

(slide - what is docker)
what exactly docker docker containers does not use the guest operating system it uses the host operating system. let us refer to the diagram that is shown. this is the host operating system and 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. alright and there can be multiple containers running as you can see that there are two. containers here one and two so on top of the host machine there's a docker engine and on top of the docker engine there are multiple containers and 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.

(slide - docker in a nut shell - docker file)
So this is a general workflow of docker or you can say one way of using docker over here what is happening a developer writes a code that defines an application requirements or the dependencies in an easy to write docker file. and this docker file produces docker images so whatever dependencies are required for a particular application is present inside this image. and what are docker containers docker containers are nothing but the runtime instance of docker image. this particular image is uploaded onto the docker hub. now

(Not in slides)
what is docker 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 onto the docker hub all right. from docker hub various teams such as QA or production team will pull the image and prepare their own containers as you can see from the diagram so what is the major advantage we get through this workflow so whatever the dependencies that are required for your application is actually present throughout the software delivery lifecycle. 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, ba, dev, test or product.

(question)
What is 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 repositories alright so it is just like a git repository for docker images.

(jenkins slide)
a docker example so this is 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 Docker hub 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. over here 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. and that build environment is deployed onto various themes like testing staging and production.

so 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 repository

continuous integration servers like Jenkins will pull that code and build an environment that contains all the app dependencies for that particular micro service. 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.

so if you can recall the first problem where application works fine in dev but does not work in prod. so with this workflow we can completely remove that problem because the requirements for the micro service is present throughout the software delivery lifecycle. and this image also explains how easy it is to implement the micro-service architecture using docker.

(Slide) Docker case study (Indiana University)
how industries are adopting docker. so this is the case study of Indiana University. before docker, they were facing many problems. so let us have a look at those problems one by one. the

first problem was they were using custom script in order to deploy that application onto various VMs. so this requires a lot of manual steps and the second problem. was their environment was optimized for legacy Java based applications. but their growing environment involves new products that aren't solely Java based. so in order to provide the students the best possible experience they needed to began modernizing their applications. let us move forward and see what all other problems Indiana University was facing so in the previous problem of docker Indiana University they wanted to start modernizing their applications. so for that they wanted to move from a monolithic architecture to a microservice architecture. and the previous slides we also saw that if you want to update a particular technology in one of your microservice 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.

(slide)
(next problem) Indiana University also needed security for the sensitive student data such as SSN and student health care data. so there 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. and docker data center has various components which are there in front of your screen.

(slide docker data centre)
first is universal control plane, then comes LDAP swarm, CS engine and finally docker trusted registry. now let us move forward and see how they have implemented docker data center in their infrastructure.

(slide: Docker Data Centre)
this is a workflow of how Indiana University has adopted docker data center. 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. as we saw that Indiana University wanted to move from a monolithic architecture to a micro service architecture so because of that reason these docker images contains the dependencies for one particular micro service but not the whole application. after that comes Universal control plane. it is used to deploy services onto various hosts with the help of docker images that are stored in the docker 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 so without doing a lot of manual steps. as we saw in the previous slides that Indiana University was earlier using custom scripts to deploy application onto VMs that requires a lot of manual steps that problem is completely removed here. 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.

(Not in slides)

(questions) what is the difference between docker containers and virtual machines. so docker containers are nothing but the lightweight alternatives of virtual machines. as I've told you earlier is where. 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. 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. when we talk about run time, the ocker 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. so these are the three major difference

(slide - docker registry)
Docker Components. first is a docker registry the orchid registry is nothing but the storage of all your darker images. your images can be stored either in public repositories or in private repositories. 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 from 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 as well that is totally up to you. so for better understanding of docker hub let me just show you how it looks like so this is how docker hub looks like. so first you need to actually sign in with your own login credentials after that you will see a page like this which says welcome to docker hub. over here as you can see that there is an option of 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 Publically. so let us go ahead and explore some of the publicly available repositories. so we have repositories for engineX, redus, Ubuntu then we have docker registry alpine, Mongo, my sequel swarm. so what I'll do I'll show you a center ste repository so this is the center waste repository which contains the center base image. now what I'll do later in the session I will actually

pull a Center wise image from docker hub.

(Slide - Docker Images & Containers)

now let us move forward and see what our docker images and containers. so docker images are nothing but the read-only templates that are used to create containers. these docker images contains all the dependencies for a particular application or a micro service. you can create your own image and upload that onto the docker hub and at the same time you can also pull the images which are available in the public repositories and in docker Hub.

let us move forward and see what our docker containers. docker containers are nothing but the runtime instances of docker images. 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. alright so for better understanding of docker images and docker containers what I'll do on my ubuntu box I will pull a CentOS image and I'll run a CentOS container in that.

Demo
Docker Ubuntu
Docker Compose 489 Project.

(demonstration)
so let us move forward and first install docker in my ubuntu box. so guys this is my ubuntu box. over here first I'll update the packages. so for that I'll type sudo apt-get update. asking for password. it is done now. before installing docker I need to install recommended packages. so for that I'll type sudo apt-get install Linux - image - extra - your name space - R and now a Linux - image - extra - virtual and here we go. press Y.

so we are done with the prerequisites so let us go ahead and install docker. sudo apt-get install darker - engine.

so we have successfully installed docker if you want to install docker and send to us you can refer the sent to its docker installation video. now we need to start this docker service. for that I'll type sudo service docker start. so it says the job is already running. now what I will do I'll pull CentOS. to is image from docker hub and I will run the Centaurs container so for that I'll type sudo docker pull and the name of the image that is Cent OS so first it will check the local registry for centos. its image if it doesn't find there then it will go to the docker hub. for Center wise image and it will pull the image from there so we have successfully pulled a centos image from docker hub.

now I'll run the Centaurs container so for that I'll type sudo docker run - IT cent OS. that is the name of the image. so we are now in the centOS container. let us now recall what we did. first we install docker on Ubuntu. after that we pulled CentOS image from docker hub. and then we build a centaurs container using that sent OS image. so are we clear now I will move forward and I will tell you what exactly docker compose is.

(slide docker-compose slide)

(Demonstration) so let me show you practically how it is done on the same ubuntu box where I've installed docker and I've pulled a Cent os image. so this is my Ubuntu box. first I need to install docker-compose here but before that I need Python pip. so for that I'll type sudo apt-get install Python - PIP and here we go. so it is done now. I will clear my terminal and now I'll install the docker-compose. for that I'll type sudo VIP install docker - compose. and here we go so dr. Campos is successfully installed. now I'll make a directory and I named it as WordPress. mkdir WordPress now I'll enter this WordPress directory. now over here I'll edit docker - compose YML file using. G edit you can use any other editor that you want I'll use e edit so I'll type sudo Gedit docker - compose YML and here we go.

(demo) showing YML code for wordpress.

I have defined a container and I've named it as WordPress. it is built from an image WordPress that is present on the docker hub. but this WordPress image does not have a database. so for that I have defined one more container and I've named it as WordPress underscore DB. it is actually built from the image that is called a Mario DB which is present in the WordPress. and I need to link this WordPress underscore DB with the WordPress container. so for that I have written links WordPress underscore DB : my sequel. alright and in the post section this port 80 of the docker container will actually be linked to put 8 0 8 0 of my host machine.

now what I've done I've defined a password here as nwen405, you can give whatever password that you want and I've defined one more container called PHP myadmin this container is built from the image could be new / stalker - PHP myadmin that is present on the docker hub. again I need to link this particular container with WordPress underscore DB container. for that I have written links WordPress underscore Divi : my sequel and the post section the port 80 of my docker container will actually be linked to put eight one eight one of the host machine. and finally I have given a username that is root and I've given a password as Nwen405

(explaining wordpress docker-compose YML file) I have defined a container by the name WordPress it is built from an image called WordPress that is present in the docker hub alright but this image that is present in the docker hub does not have a database. so I need to define one more container that contains a database. so WordPress underscore DB is the name of that container and it is built from an image called Mario DB which is present on the docker hub. now as I've told you earlier as well . so I need to link this database with WordPress. so what I'll do I'll write links WordPress underscore DB : my sequel . alright and the port section, the port 80 of the docker container will be linked to port eight zero eight zero of the host machine. after that we require one more container and I've named it as PHP Myadmin. (27:50) it is built from the image called Corbino / no / - PHP myadmin and I need to link this particular container with WordPress underscore DB. and my sequel is actually the name. that I have given in the post section the port 80 of the container is actually linked to put eight one eight one of the host machine and finally I have given username and password I've given username as root and password as nwen405.

So let us now save it and we'll quit . let me first clear my terminal and now I'll run a command sudo docker - compose up - D and here we go so this command will actually pull all the three images and we'll build the three containers You

so it is done now let me clear my terminal now what I'll do I'll open my browser and over here I'll type the IP address of my machine or I can type the hostname as well first name of my machine is localhost so I'll type localhost and put eight zero eight zero that I have given for WordPress so it will direct you to a WordPress installation page over here you need to fill this particular form which is asking you for site title I'll give it as ed Eureka username also I'll give us a lyrica password I'll type in nwen405 confirm the user week password then type your email address and it is asking search engine of visibility which I want so I won't click here and finally I click on install WordPress so this is my WordPress dashboard and WordPress is now successfully installed now what I'll do I'll open one more tab and over here I'll type localhost or the IP address of my machine and I go to port 8 1 8 1 for PHP myadmin and over here I need to give the username if you can recall have given root and password I have given as nu Rekha and here we go so PHP myadmin is successfully installed this PHP myadmin is actually used to access a my sequel database and this my sequel database is used as back-end for WordPress