

**Applied Accelerated Artificial Intelligence**  
**Prof. Satyadhyan Chickerur**  
**School of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

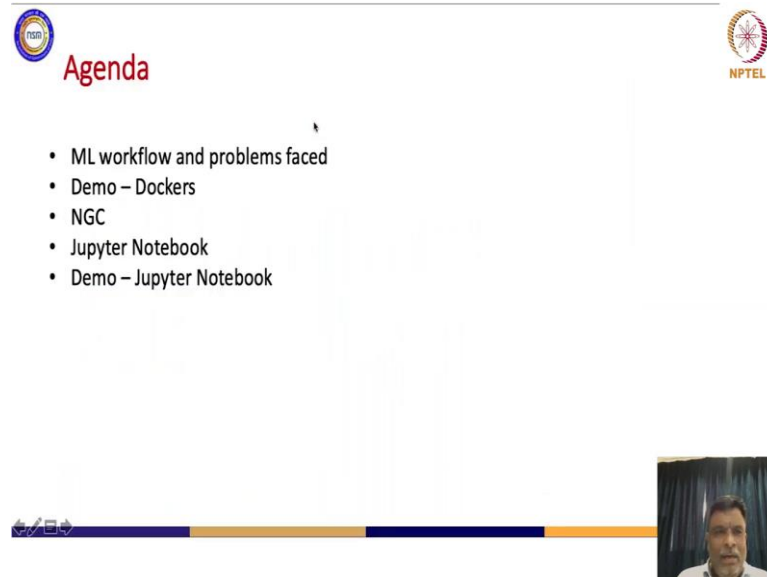
**Lecture - 06**  
**Introduction to Containers and IDE Dockers Part - 1**

Good evening everyone, so, today we are going to actually start with a bit of hands-on, on containers and IDE. And specifically, we will be talking about Dockers and we will do hands on Dockers and I had suggested also for people who would like to do some hands on along with me. And I had actually submitted a link also through which you would have actually tried to download and install Dockers in any of the three types of operating systems right.

The intention today is to just make you comfortable with Dockers. It is also going to be shown as to how do you use GPU with Dockers using optimized containers which NVIDIA also provides which effectively run on NVIDIA GPUs. But the basic essence today is to be accustomed and comfortable with working with Dockers.


So, I am starting with a very basic thing; so, that if you would have installed Dockers, just installed Dockers you can start working with them straight away right. So, that is the intention of this particular session today. In the previous session you had seen the concepts of virtualization, you had seen the concepts of containers right. So, we are taking it forward from there and trying to do hands on today ok.

(Refer Slide Time: 01:43)



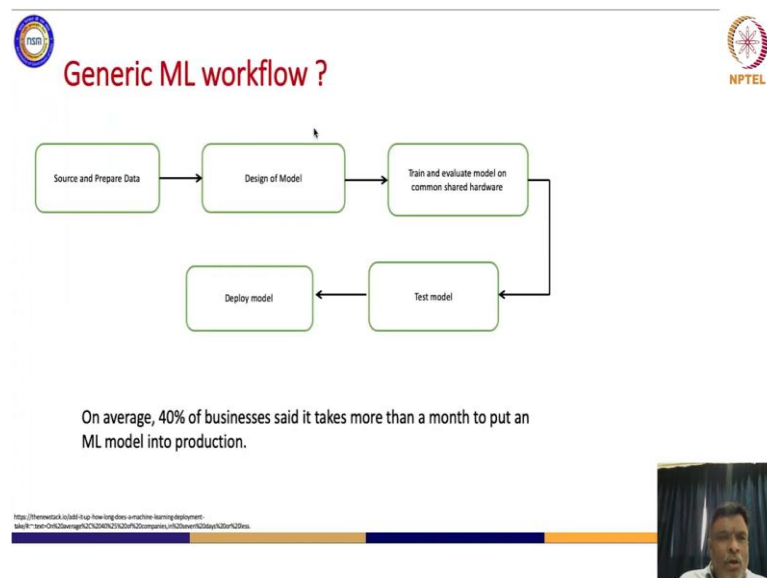
## Agenda

- ML workflow and problems faced
- Demo – Dockers
- NGC
- Jupyter Notebook
- Demo – Jupyter Notebook



So, let us start with the agenda of today, we would be doing ML workflow, why is actually Dockers needed right. We would be discussing about the general machine learning workflow and the problems faced. Then we will do hands on Dockers, I will take you to the NGC website wherein you can register in your leisure and try to download optimize Docker containers. Then a very brief on Jupyter Notebook and then we will try to run a Jupyter Notebook on a DGX using a Docker container right. So, this is the overall agenda of today.

(Refer Slide Time: 02:23)




## Generic ML workflow ?

```
graph LR; A[Source and Prepare Data] --> B[Design of Model]; B --> C[Train and evaluate model on common shared hardware]; C --> D[Test model]; D --> E[Deploy model];
```

On average, 40% of businesses said it takes more than a month to put an ML model into production.

<https://theresetack.co/blog/4-up-how-long-does-a-machine-learning-deployment-take/>

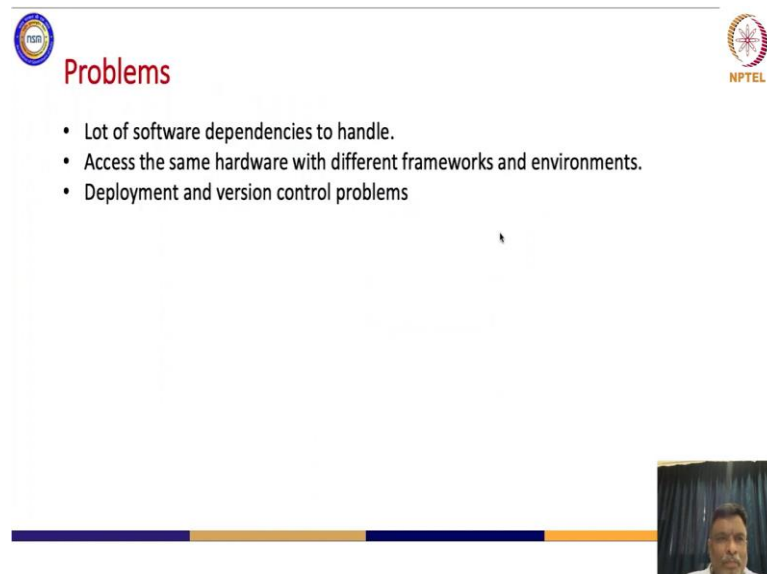


So, this is the generic workflow of any machine learning, deep learning algorithm. What you generally do is, the first step is to source get and prepare your data then you tend to design a model. You need to train and evaluate the model on a common shared platform, why I am telling common shared platform is there is let us say a DGX or there is a cluster which is commonly available to all the teams right.

All of you would be doing your respective models or your respective application, but there is a centralized facility. Because its a big cluster it has to be shared by all the people or it can be a DGX or whatever right, but ultimately you should know how to work on a shared hardware right. So, once you are able to do this training and evaluation of the model on a common hardware, you need to test that model right and then you need to deploy that model.

So, this is a generic machine learning workflow and on an average 40 percent of businesses right said it takes more than a month to put an ML model into production right. If this is the scenario right how are we going to optimize ok, our delivery in such a manner that we put in less amount of effort just to put any ML model which we have designed into production right.

(Refer Slide Time: 04:01)



**Problems**

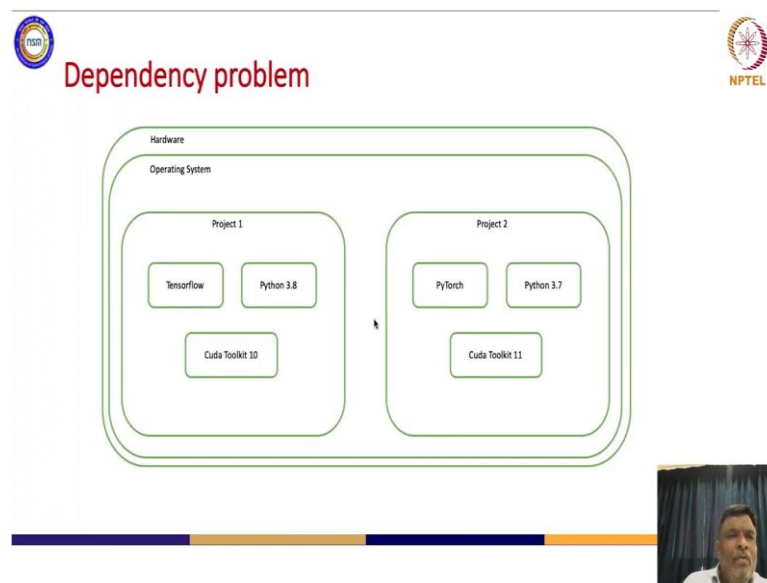
- Lot of software dependencies to handle.
- Access the same hardware with different frameworks and environments.
- Deployment and version control problems

So, what are the problems in such a scenario? There are lot of software dependencies to handle. Because we will see right there will be different teams working on different type

of frameworks, different type of operating systems, the versions and so on and so forth and all have to run on the same hardware right.

So, you need to access the same hardware each of you will be having different frameworks and different approaches to solve. And you need to have deployment and version control problems which will be encountered as you go along ok. So, these are the three problems basically which you have to encounter when you are working on a common shared hardware right.

(Refer Slide Time: 04:49)

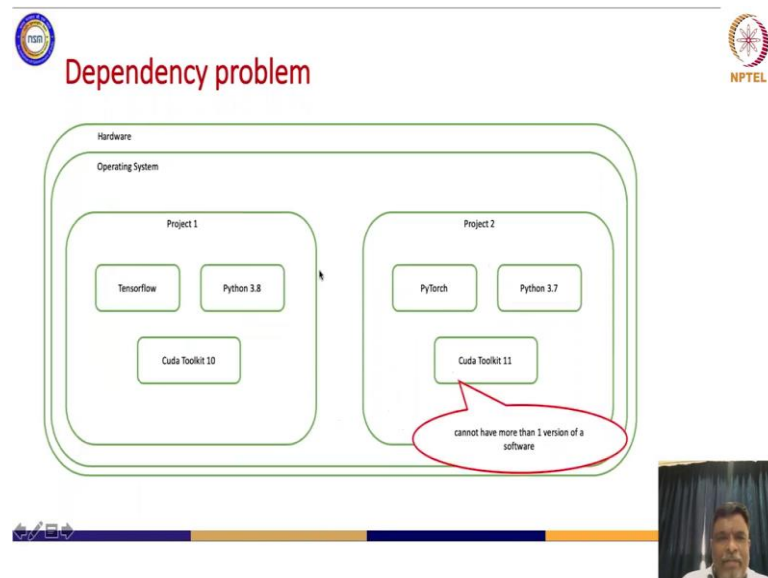


So, let us try to understand this dependency problem what it is. So, if you see this this is the common shared hardware on the slide right this is a shared hardware. So, it can be a cluster, it can be a DGX, it can be a system which has got multiple GPUs on that you would be running a operating system right and that operating system will have to be running certain projects which you would like to do.

Now, let us say from the Layman's perspective, I am just trying to tell you and correlate right whatever we studied in the previous class about virtualization and containers and all that to a very generic simple type of a example. Wherein let us say you have two projects and there is one team which is trying to do project one with TensorFlow with Python 3.8 and you are trying to access a GPU and you are trying to work with CUDA Toolkit 10.

There is another team which is working on project 2 and its working with PyTorch Python 3.7 and CUDA Toolkit 11. So, what is the actual thing here of what we should understand; the problem here is, this particular team requires these things and this particular team requires these things. And all of them are supposed to run on the same operating system and then on the hardware right if this is the scenario.

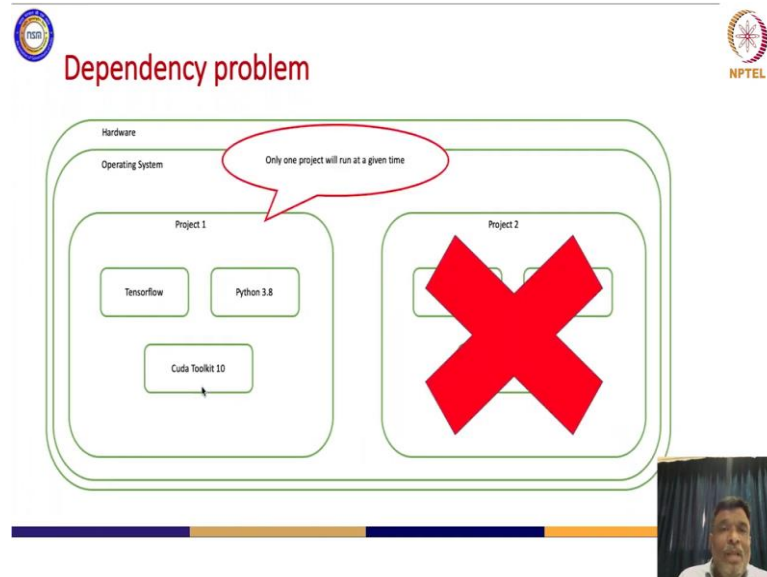
(Refer Slide Time: 06:28)



If this is the scenario what effectively happens is, there is a dependency problem. Now what is that dependency problem? Right, you cannot have more than one version of a software, because you will have clashes about, the drivers right the toolkit all of this right it all gets complicated. So, if this person installs everything on this operating system ok, how are you going to use it?

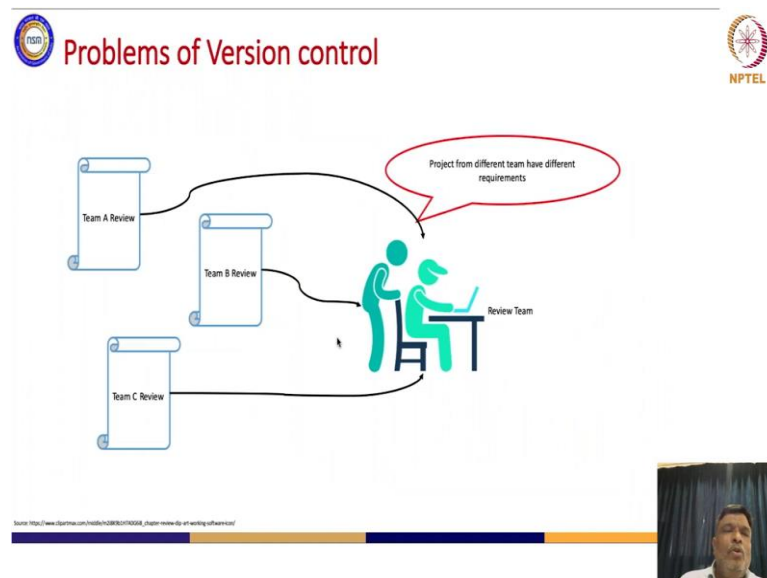
You cannot use it because you want CUDA Toolkit 11 the other person would have installed CUDA Toolkit 10. In that case what are you going to do this is one of the dependency problem this is a problem actually right.

(Refer Slide Time: 07:05)



And then this ultimately means that you can only run one project ok; so, only one project will run at a given time in such a scenario wherein you are bound to be installing everything on that operating system right.

(Refer Slide Time: 07:23)

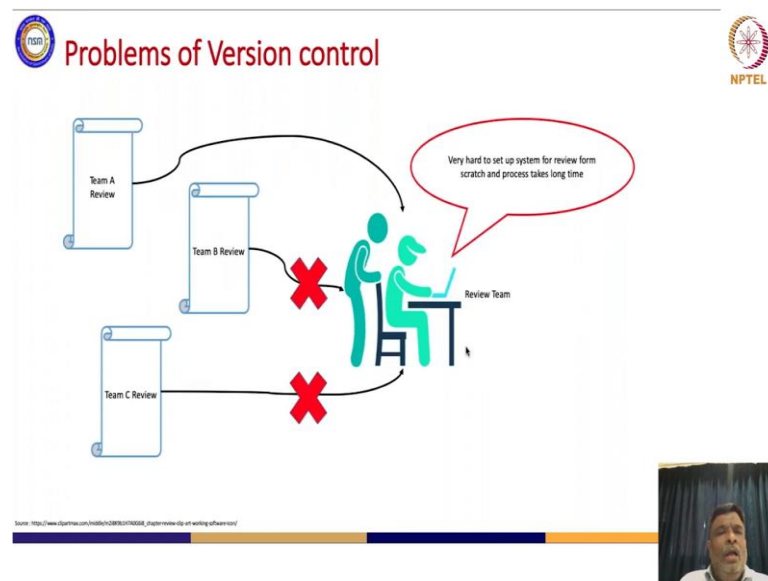


So, to sum up if there is a review team which is going to review let us say for example, they are trying to do some testing right of your code. So, there are there is a team A team, B team, C all three of the teams submitting their code for testing right say. And

these projects are from different teams with different requirement and review team has to do it.

So, how are they going to set up these environments right which you have told that because it is running on your system you want it effectively to be running on somebody else system given the same conditions, given the same framework, given the same dependencies right. So, to do all that the team has to invest lot of time right to do all this.

(Refer Slide Time: 08:10)



So, ultimately what happens again if you set the system for team A review team B and team C would have done something else with some different framework, some different versions of toolkits right. In that case very hard to set up the system for review right from scratch and process is going to take a lot of time.

(Refer Slide Time: 08:30)

**Possible Solutions**

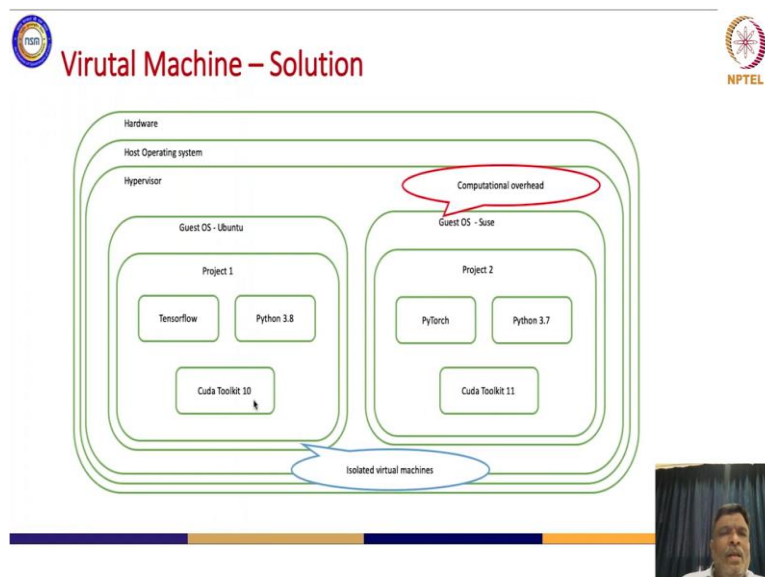
1. Virtual machine
2. Docker and Containers

http://nptel.sru.ac.in/lec02/Project\_Programming\_Patterns.pdf

The slide features the IIT Bombay logo on the top left and the NPTEL logo on the top right. A small video feed of a presenter is visible in the bottom right corner.

So, there are basically two possible solutions ok which you can use which we have seen also in the previous class, there are virtual machines which you can use and then you can use Docker and containers right. So, in case of a virtual machine how is the possible solution going to look like right.

(Refer Slide Time: 08:53)



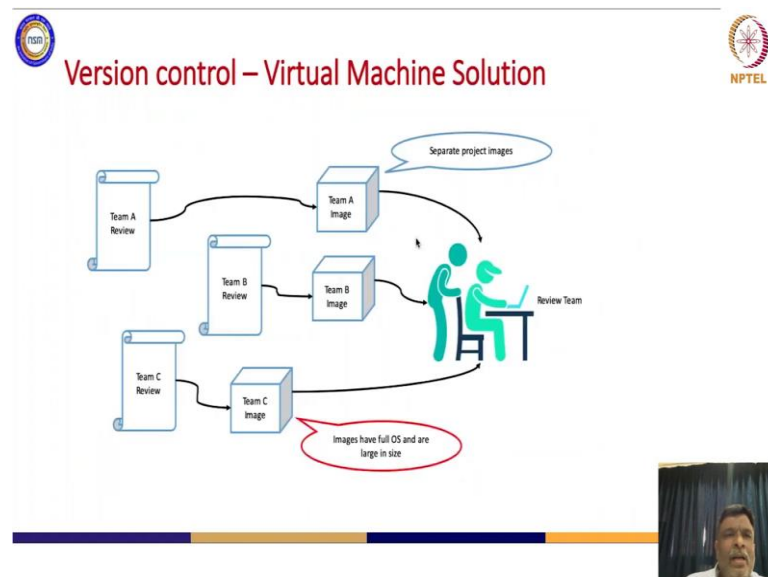
So, this is how it is going to look like, you have the hardware, you have the host operating system, you have the hypervisor and then let us say for this project 1 you work with the guest OS Ubuntu. And then you have TensorFlow Python 3.8 CUDA Toolkit 10



and then there is another virtual machine which basically is again having a guest OS like SUSE Linux or something and then you work with PyTorch Python 3.7 and CUDA Toolkit 11.

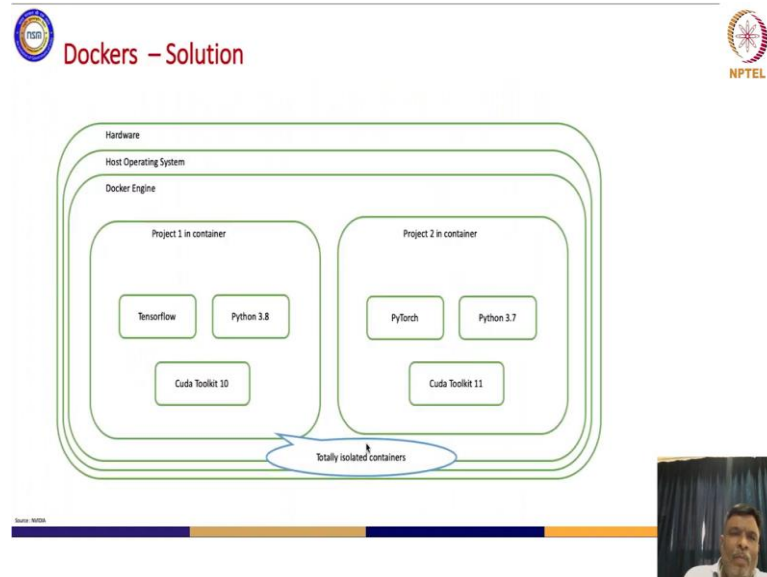
These are isolated virtual machines in the sense that project 1 also could work on this, project 2 also could work on the same system right. But what is the issue? The issue here is that you have got computational overhead in both these cases, because you have a guest OS in both of that both of these virtual machines which will make it more bulky or heavy right it takes lot of memory.

(Refer Slide Time: 10:04)



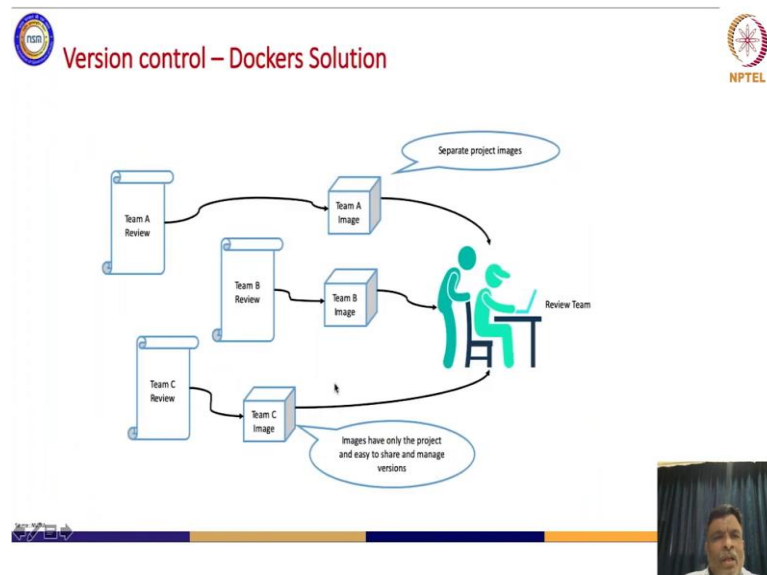
So, this is this ease and was one of the possible solutions which was effective till Docker scale. Now, in this case also let us say team A review you have a separate project image, team B also has a separate project image. So, review team is a bit happy about it, but the only thing is these images have full OS and are large in size; so, that is the only thing which is of a disadvantage here.

(Refer Slide Time: 10:33)



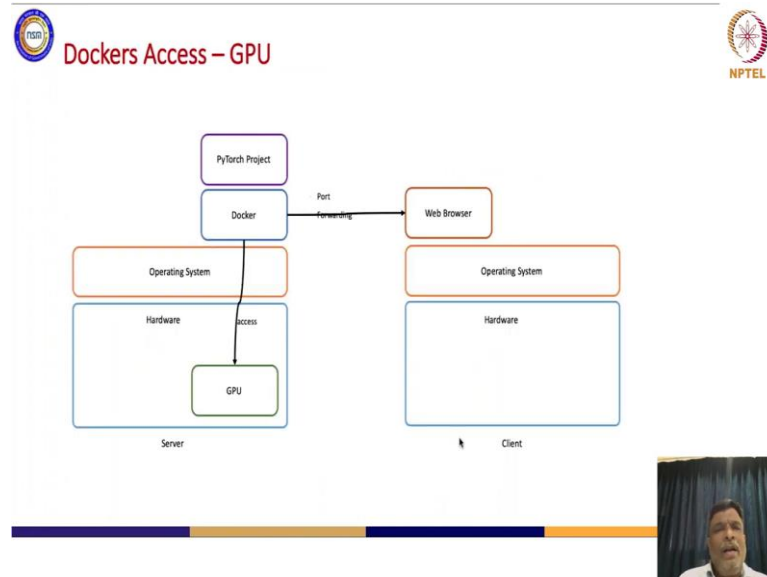
Now, when the Docker came the solution was something like this, you have the hardware you have the host operating system. You have a Docker engine and you have project 1 in container, project 2 in container it may have ok, things wherein it will have its own framework ready everything ready these are all totally isolated containers right.

(Refer Slide Time: 11:02)



So, from the review aspect these images will have the project and they are easy to share and manage you have got separate project images.

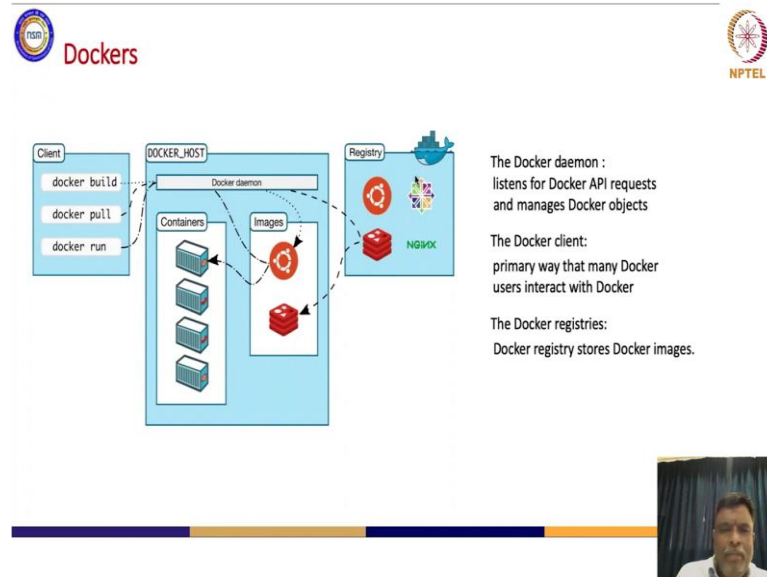
(Refer Slide Time: 11:14)



So, this is how the Docker actually gives you a lightweight solution to a virtual machine problem. And in case you have to have a GPU access also, this is just a very basic diagram for people to understand that let us say this is a server wherein you have your Docker image running PyTorch project through the operating system. The hardware you have the access to the GPU and the thing which gets done here through port forwarding.

This is your system from where you are trying to run this Docker on this hardware; on this hardware and then you are going to project it through your own web browser. And can work with Jupiter Notebook you can work with many things ok and so, this is the overall idea about how you can work with Dockers right.

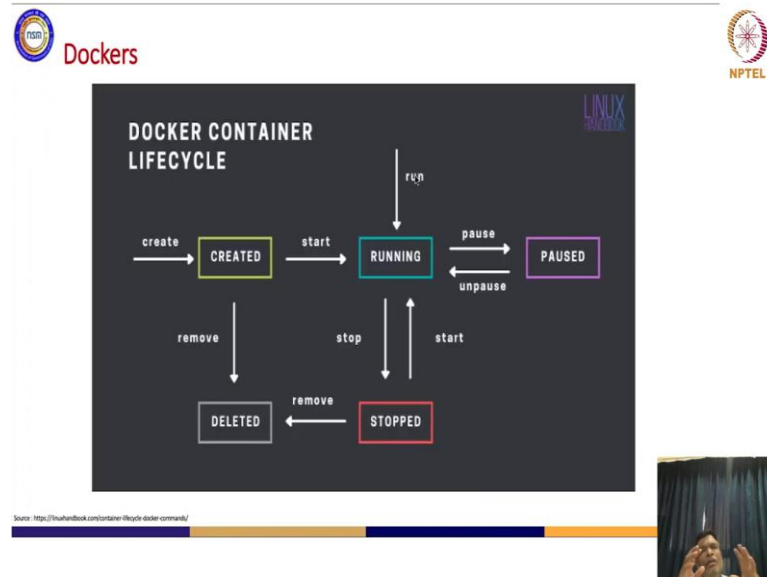
(Refer Slide Time: 12:15)



So, now, before we start to do some hands on, let us try to understand whatever I showed you ok. You have a client this is a Docker host the Docker, host will have a lot of containers, you will have a lot of images this is controlled by Docker demon. And then you have a registry from where these Dockers could be pulled.

So, I am telling you two generic registries today; one is a registry which is basically from the Docker hub, and another one is a registry which is from the NVIDIA NGC cloud. So, you can pull in those containers ok and then try to use it for your own application or project development right. So, this is a basic brief idea about how is the total setup with which you can work with Dockers ok.

(Refer Slide Time: 13:17)



So, the Dockers if you see the container life cycle right, we will see what a container is how it is done. But for the time being just try to understand that a Docker container lifecycle you create a Docker container, you can run it, you can pause it, you can stop it, you can delete it ok. So, at various stages you can do all of these operations on those Docker containers. So, this is just a brief idea of Docker containers.