

**Cloud Computing**  
**Prof. Soumya Kanti Ghosh**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 36**  
**Introduction To DOCKER Container**

Hello, so welcome to the course on cloud computing. So, so far what we were discussing is about that how this cloud computing evolved, what are the different advantages and challenges in cloud computing? What are the different associated technologies to this cloud computing aspects? So core architecture and what are the different other type of technologies which came up with those things.

So, as we have seen it may not be a totally new technology or new innovation as such, but it is trying to look at all these innovations, all these new approaches and coming up the things. Another new development what we will look today is a container technology. So, what we are finding; one way as we have seen that there are challenges in infrastructure build up, there are challenges in platform build up, there are challenges in software build up.

That is why we try to migrate from our in house installation to some service providers installation. So, typically as we have discussed on XAS type of services; infrastructure platform and software, these are the different type of services. Another problem which is coming up nowadays is; or rather with the development of software and other developments like there is a redeployment or reconfiguration or recompilation of the software whenever we want to ship one package from one environment to other.

This is one of the major motivation towards these; these days we are having devices, which are pretty resourceful; say your mobile device or some notepad or laptop or even these days smart watches. So, all those things are resourceful and able to run different applications. So, one way is looking at that sort of things; other way another aspects of it is; we are having variety of applications development across the things. So, this type of portability of one application from one environment to another is becoming a major challenge.

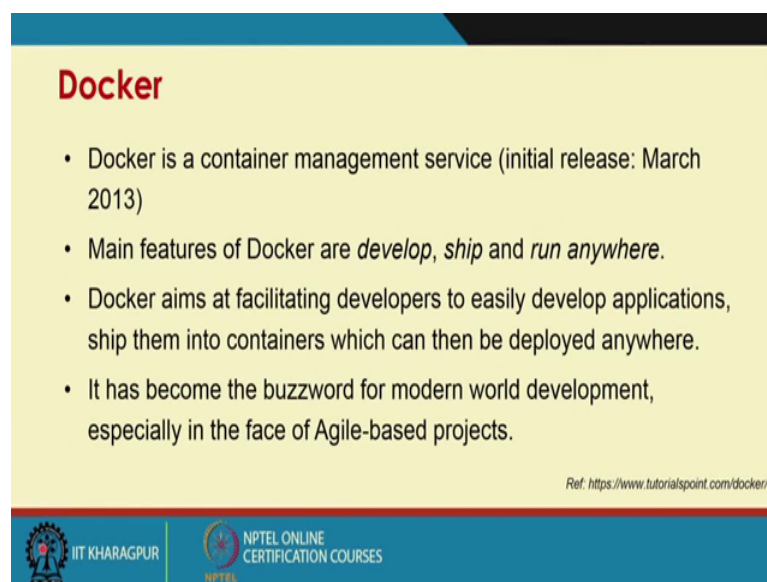


So, there were the container technology came into picture; where we will be able to bundle up the application along with these associated dependencies into a container and push it as a single container to the other type of environment; I mean the amount container fits, your applications start working on that.

So, cloud in such cases can be a platform to host this container; though there are some literature or in some forum people say this container technologies is something a contended to the cloud. But what we see these days not exactly a container to or a something which is fighting with cloud; rather we can cloud can act as a platform to host these container and basically cloud can enhance its capability; by adapting this container technology.

One such container which has become a very popular is the Docker; a open source container which has become a Docker. So, what will see; we will see a brief introduction to this Docker container. It will help us in understanding this how this portability of different systems and software will be there and also help us in looking at this container technology with a background of a cloud computing.



(Refer Slide Time: 04:24)



**Docker**

- Docker is a container management service (initial release: March 2013)
- Main features of Docker are *develop*, *ship* and *run anywhere*.
- Docker aims at facilitating developers to easily develop applications, ship them into containers which can then be deployed anywhere.
- It has become the buzzword for modern world development, especially in the face of Agile-based projects.

Ref: <https://www.tutorialspoint.com/docker/>

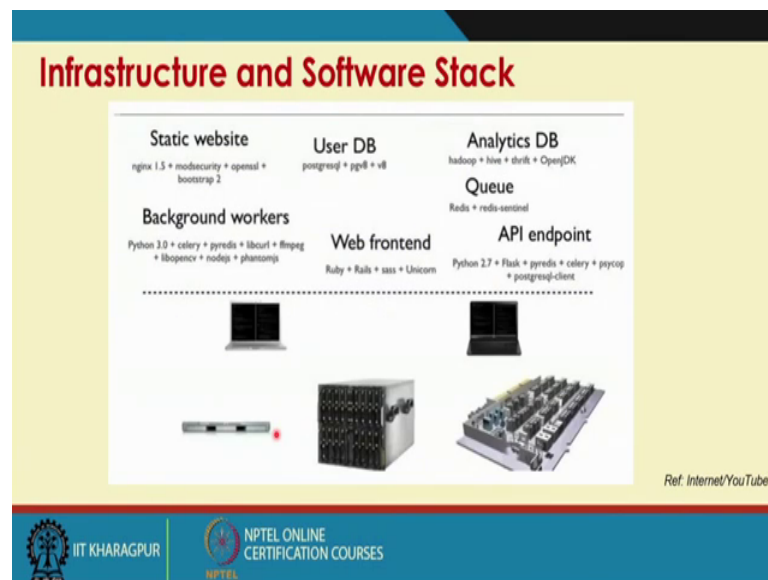
 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

So, if you look at that Docker; it is somewhere initial release where March 2013. So, it is a container management service; main feature of Docker are develop ship run anywhere. So, this key words if we can see like; develop; ship it and run anywhere. So, irrespective of where it is, it should be able to run; this is basic philosophy of Docker which tells



everything like you develop once and ship to any environment and run to this. It can be any your desktop environment, it can be a android environment, it can be a IOS environment, it can be a say any cloud environment, it can be open stack, it can be azure, it can be blue mix or Amazon; anything.

(Refer Slide Time: 05:47)



So, that is exactly the container things trying to say. So, the Docker aims at facilitating developer to easily develop applications, ship them into containers which can then be deployed anywhere. It has become the buzz word for modern world development; especially in the face of agile based projects. So, this is the buzzword these days to look at the these type of things. We will try to make a quick analogy of the things and try to understand what is there in this type of things.

Now, if you look at how our computing paradigm develop. So, it was somewhere a system like it maybe a somewhere; something like a laptop or even desktop or some specialized board or a chassis with blade servers and any type of things, even if I can look at some devices like network devices, where we basically build some applications for them. Now, if I am running something on a particular laptop and then want to run it on a server thing, then in most of the cases I need to recompile the things, I need to align the things or redevelop; not that redevelop the whole thing, but I need to redevelop a major portion of the things or recompile the (Refer Time: 06:51) into the server I want to do for another type of system; I need to again redevelop the things.










So, what is happening that there is a lot of man power uses into the things. Whereas, on the other side; so one side there is a large (Refer Time: 07:11) miss or there is a everyday we are coming or means periodically you are coming with new systems and software's and ways and type of things. On the other hand we have several applications which are coming up on the other hand. Some maybe static website; something like background worker like python and so and so forth, user DB, analytics DB, queue, API endpoints, web frontend and endless things are being developed.

Now, if you look at that for everything; if you want to have. If you want to sip a application from one to another and if we want to this sort of recompilation or reconfiguring the things it is a hell of a job.

(Refer Slide Time: 08:00)

**Goal: Interoperability**

Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?
							

Ref: Internet/YouTube

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, what we try to; when look at that interoperability or portability; what we try to do? We try to set up a matrix. Like say static website; whether it will run on this particular device, whether you run on this device and so and so forth; it is a user DB and type of things.

So, if I put tick on this matrix then this is there. So, our major objective is that with minimal investment in terms of man power, skill, infrastructure and software and so and so forth; I should be able to have all ticks on this matrix; anything can run anywhere, type of situations. So, that is a big challenge to add here, but never the less; the whole computing world wants to achieve.



This leads to commercialization of a product in a better way, marketability of a product in a better way and rather in some cases like if you look at sensor based technologies; if the sensor applications runs only on a particular type of smart phone and or a particular type of OS and it is not running to the other things, then your capability also decreases. I could have captured through different type of mobile devices, but that portability becomes a major challenge.

So, these are some of the things which has driven that whether I can have a; somewhere some intermediate mechanisms which allows me to transfer this data.

(Refer Slide Time: 09:37)



Now, if we try to look at some analogies; look at shipping. I want to ship some product from one to another. So, one side these products are there like; there can be variety of products. Like it can be a box, it can be a car, it can be some barrel, it can be even a piano and drums and servers and etcetera. So, these are the things I want to ship from place A to B and it can be shift in varies form.

It can be in somewhere some trucks and sort of things, somewhere can be dumbled into a racks, some fork lifter to lift on the things, somewhere some locomotive or trains being manufactured out of factory and see it and through crane and put on the ship for shipping from across the continent.



So, there are a variety of products and variety of techniques and technologies which are involved in shipping these products. Unless I standardize; like if you think that the type of mechanisms required for a server will differ from how you do for a piano or what you do for a car. So, unless I make a standardized way of doing this business; then things will not work.

(Refer Slide Time: 11:00)



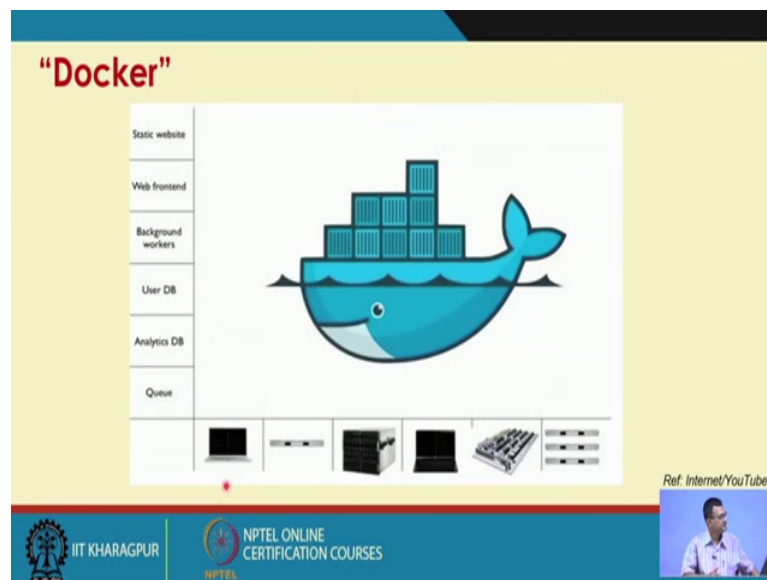
So, what they found out? They found out a concept for this sort of a docking station or this sort of a chamber where if it fits in, it can be shipped. So, what we do; any product, we put into a this sort of docking station or this type of a box. And this box has a standardized size, standardized mechanism of holding and it can be either put on rack, put on fork lifter, put on crane, put on ship; can be taken out from factory or inside the factory or put on the trucks and carry it; these are very popular.

Now, I made this shipping thing to any type of mechanism to transport; by making a intermediate mechanism call shipping containers. Whether, I can make this thing feasible for our applications, for our software applications. Now this container is a standardized size, so if somebody can hold the container; it can hold these products. So, what it does; the inside the container is more on the product which dictates, outside the container the infrastructure which dictates. This outside the environment of the container is more inclined towards this infrastructure by which it is shipped or where it is stored and type of things.



Inside the container is the way where the product is stored; like piano may have a different requirement of environment than car than a server systems. So, the same type of conceptually, the same type of philosophy is being carried to this; when we talk about container platform, when we talk about Docker.

(Refer Slide Time: 13:11)



So, same thing what we say it is a Docker service. We have different applications and there are different devices and if I can dock it in somewhere other; which is standardized and if these devices are able to handle this container or Docker container then my application will run.

So, what inside that container is there; based on the applications all these related libraries, binaries, dependences etcetera are contained in this thing; externally where which infrastructure it is running that is important. So, this sort of philosophy helped in as we stating that you develop and ship to anywhere; develop, build, ship anywhere.

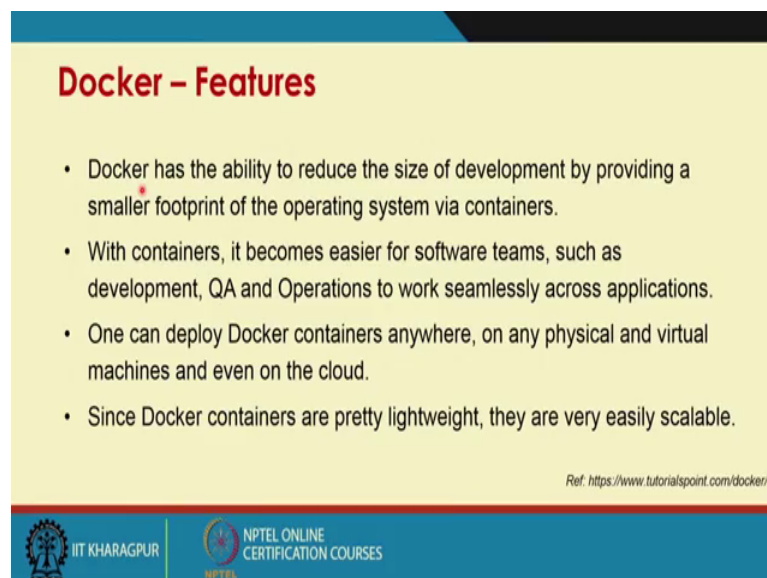
So, if this type of technology is becoming extremely popular and becoming a defect standard for this and what we feel that is necessary to discuss. Because most of the cases what we are doing in a cloud, we are trying to run different applications within this cloud. Like, if you look at say I want to run in a particular banking application or we talk about some special web service applications.



So, what it tries to do? It tries to build a container; particular container class in a sense or a club of services which need to be shipped or needs to be run at different environment. Let it be as your cloud, let it be open stack cloud, let it be any other sort of cloud like IBM Blue Mix or Amazon or Google cloud platform any type of platform. My basic requirement is there; I should not develop, redevelop for every environment and as well it should run, if the resource permits on my smart phone or my desktops systems or server and so and so forth.

So, this sort of portability of the things; this container brings into picture.

(Refer Slide Time: 15:40)



**Docker - Features**

- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for software teams, such as development, QA and Operations to work seamlessly across applications.
- One can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.

Ref: <https://www.tutorialspoint.com/docker/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, if we look at again that the basic features; so, Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers; so, in the container you have a smaller footprint of the OS. With containers, it becomes easier for software teams; such as development, QA teams operations to work seamlessly across applications. So, they are may be at different centers; even for a one organizations and it can work seamlessly because finally, it should fit into the container in being shipped to the other things.

One can deploy Docker containers anywhere; on any physical and virtual machines and even on cloud. So, it is a Docker containers can be deployed anywhere; practically anywhere or any physical or virtual machines in VMs or physical servers and even on cloud. Since Docker containers are pretty lightweight, they are easily scalable; that is an



important factor. So, this container; this Docker container is light weight and as it is light weight then scalability is much easier.

If there is a heavy weight stuff, then there is resource requirement will be much higher and anything portability is becomes or scalability or scaling up or especially scaling up becomes extremely difficult; so, as this is light weight, this is much easier.

There are different variants or what we say different flavors of things or components. So, Docker for Mac, it allows one to run Docker container on Mac OS. Docker on Linux, it allows one to run Docker container on Linux OS. Docker for windows; Docker engine, it is used for building Docker images and creating Docker containers. Some sort of over the bare metal OS; you have the Docker engine, over this Docker containers are placed; some sort of analogy with the hypervisors of a particular virtualization system.

Docker hub; this is a registry the important thing registry which is used to host various Docker images. So, this Docker hubs you will find various Docker images; if you are a developer, you can basically submit in the Docker hub, which can be used by others to use it. Docker compose; this is used to define application using multiple Docker containers. So, it is a sort of composition service which is used to define applications using multiple Docker container. So, these are different components; there are various others or different flavors of these Docker or different components of this Docker.

(Refer Slide Time: 18:42)

### Traditional Virtualization

- Server is the physical server that is used to host multiple virtual machines.
- Host OS is the base machine such as Linux or Windows.
- Hypervisor is either VMWare or Windows Hyper V that is used to host virtual machines.
- One would then install multiple operating systems as virtual machines on top of the existing hypervisor as Guest OS.
- One would then host your applications on top of each Guest OS.

```
graph BT; Server[Server] --> HostOS[Host OS]; HostOS --> Hypervisor[Hypervisor]; Hypervisor --> GuestOS1[Guest OS]; Hypervisor --> GuestOS2[Guest OS]; Hypervisor --> GuestOS3[Guest OS]; GuestOS1 --> App1[App]; GuestOS2 --> App2[App]; GuestOS3 --> App3[App];
```

Ref: <https://www.tutorialspoint.com/docker/>

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

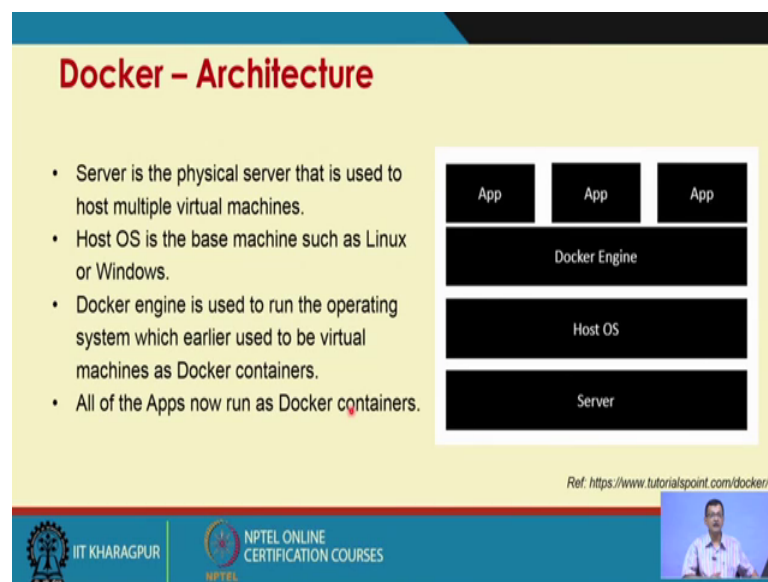


So, if we look at the traditional virtualization; so, we have that underlining server or bare metal or the backbone systems; physical systems. Over that there is a host ways which basically provide this service to this server, over that we have that hypervisor or VMM or Virtual Machine Monitor and this helps us in generating different VMs. And every VMs can have their own guest OS and over the guest OS; different applications runs.

So, this is the basic philosophy of virtualization already we have seen. So, the server is the physical server that is used to host multiple virtual machine. Host OS is the base machine such as Linux or windows, hypervisor as either some sort of hypervisor; it can be VMWare, Xen, KVM and type of things or commercially VMWare or widows hyper V that is used to host virtual machines.

One would then install multiple operating system as virtual machines on the top of the existing hypervisor as guest OS. So, what we do in case of a IAAS type of cloud and then host your application on the top of each guest OS. So, this is the way normally a traditional virtualization work.

(Refer Slide Time: 20:08)



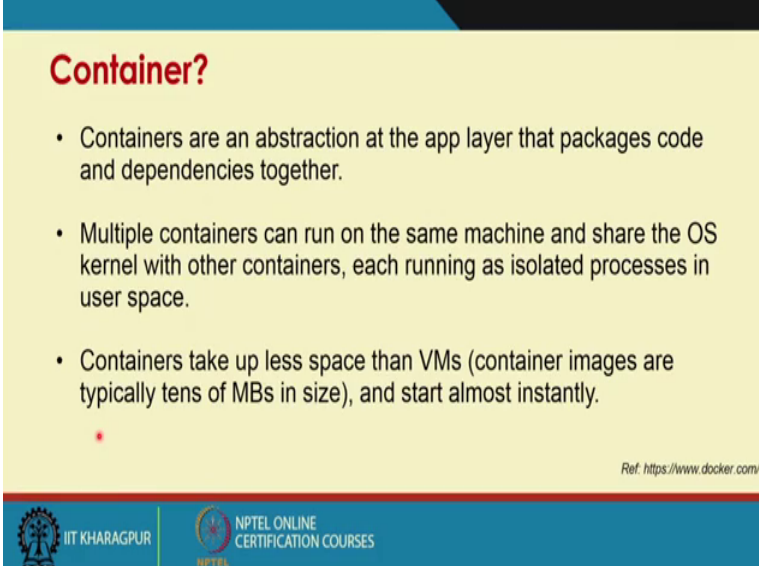
A variant of this Docker architecture is that; it has the server OS; case two s and then we have the Docker engine. So, over the Docker engine we can run different apps; so, it is going to be light weight and of course, you may not do very large applications, but never the less if your application is not so demanding; you can basically deploy out here.



Server is a physical server that is used host multiple virtual machines. Host OS is the base machine such as Linux or windows.

Docker engine is used to run operating systems; which is earlier used to be virtual machines at Docker containers. So, it is running operating systems which is earlier used as a virtual machines as Docker containers. And then finally, all apps now run on the Docker containers; so, it has now Docker containers where this different apps run on this Docker container. So, what we see there is a; though the philosophically maybe same sort of aspects is there, but there is some difference with this traditional virtualization.

(Refer Slide Time: 21:30)



**Container?**

- Containers are an abstraction at the app layer that packages code and dependencies together.
- Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space.
- Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.

Ref: <https://www.docker.com/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, just to recap that containers are abstraction at the app layer that packages code and dependencies together. So, it is a; what is this container including Docker container? It is an abstraction at the app layer; that application layer that packages or bundle code and dependency together; so, it is together bundled. Multiple container can run on the same machine and share the OS kernel and other containers each running on isolated processes in the user space, so that is also possible.

Container takes up less place then virtual machines; container images typically tens of MBs in size and start almost instantaneously as that is exactly what we are discussing, it is a low weight and we can much less resource hungry than virtual machines and it can start instantaneous as it is a low weight.



(Refer Slide Time: 22:38)

**Container (contd...)**

- An **image** is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files.
- A **container** is a runtime instance of an image—what the image becomes in memory when actually executed. It runs completely isolated from the host environment by default, only accessing host files and ports if configured to do so.
- Containers run apps natively on the host machine's kernel. They have better performance characteristics than virtual machines that only get virtual access to host resources through a hypervisor. Containers can get native access, each one running in a discrete process, taking no more memory than any other executable.

Ref: <https://www.docker.com/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

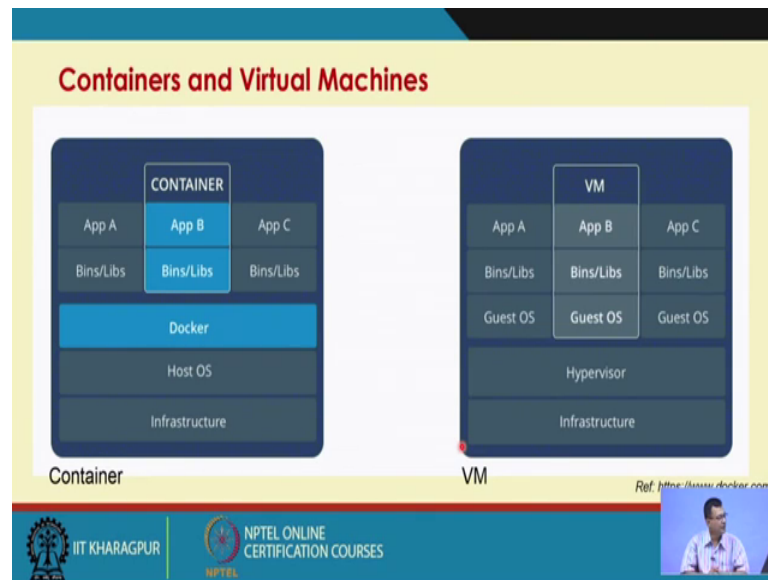
So, there are other few concepts which comes back to back. One is that image so we are talking about images and type of things. So, an image in this context is a light weight, stand-alone, executable package that includes everything needed to run a piece of software, including code, runtime, libraries, environment variables, config files etcetera; so it is important.

So, it is a light weight standalone executable package that includes everything needed to run that particular software package or software including code, runtime, libraries, environment variables, configuration files everything. A container is a runtime instance of an image what the image becomes in memory when actually is executed. So, what is there? so, I have a bundle things once it start running; it is a instantiation of this image that is exactly the container, it runs completely isolated from host environment by default only accessing the host file and ports if considered to do so. So, it is independent of the host environment.

So, other things there container runs apps natively on the host machines kernel; they have better performance characteristics than virtual machine, that only get virtual access to the host resources through a hypervisor. So, it runs directly on the host machines kernel, containers can get native access each one running in a discrete process taking no more memory than any executable. So, it is not only light weight; it has performance wise also instantaneous.



(Refer Slide Time: 24:32)



So, if you look at containers and virtual machines; so, we repeat that things; so we have the underlining infrastructure of the server base. Host base Docker and then we have that container, it has the applications with other dependencies. Whereas in case of a hypervisor or a virtual machine over the hypervisor, we have guest OS and rest of the things.

(Refer Slide Time: 25:05)

**Virtual Machines and Containers**

- **Virtual machines** run guest operating systems - the OS layer in each box.
- Resource intensive, and the resulting disk image and application state is an entanglement of OS settings, system-installed dependencies, OS security patches, and other easy-to-lose, hard-to-replicate ephemera.
- **Containers** can share a single kernel, and the only information that needs to be in a container image is the executable and its package dependencies, which never need to be installed on the host system.
- These processes run like native processes, and can be managed individually
- Because they contain all their dependencies, there is no configuration entanglement; a containerized app "runs anywhere"

Ref: <https://www.docker.com/>

So, that is the difference; so, virtual machine run guest operating system; OS layer in each box. Resource intensive, so as it is requires more resources resulting disc image and application state in a entangled with the OS settings, system-installed dependencies, OS

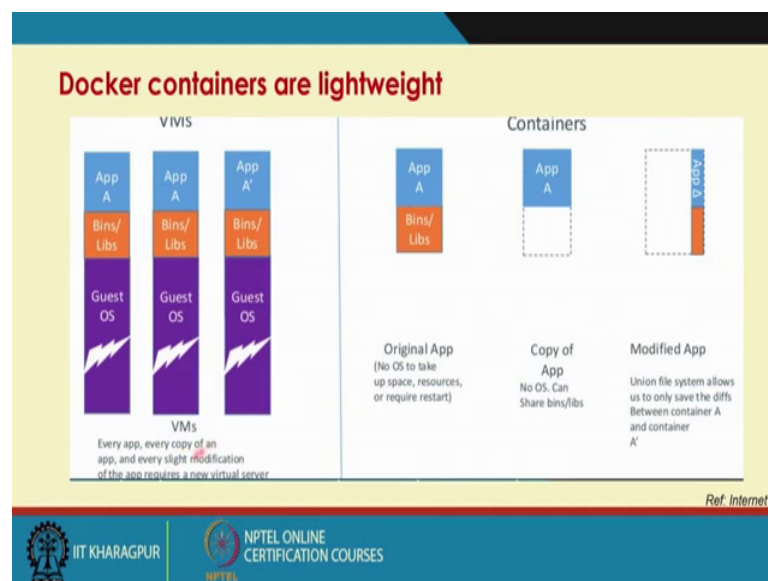


security patches and so and so forth. Whereas, container can share a single kernel and only information that needs to be in a container image is the executable and its package dependence.

So, what are the different other package dependencies which never need to be installed on the host OS. Because it is bundled together; these processes run like native processes and can be managed individual every container. Because they contain their dependency, there is no configuration entanglement and so containerized applications runs anywhere. So; that means, I do not have any so to say any binding with this host machine.

So that means, I isolate this my container based package with the host. So, it increases the portability and can be run any other system. So, same thing; so, if there is a running on a VM, then we have every app, every copy of an app, every slight modification requires a some new virtualized environment.

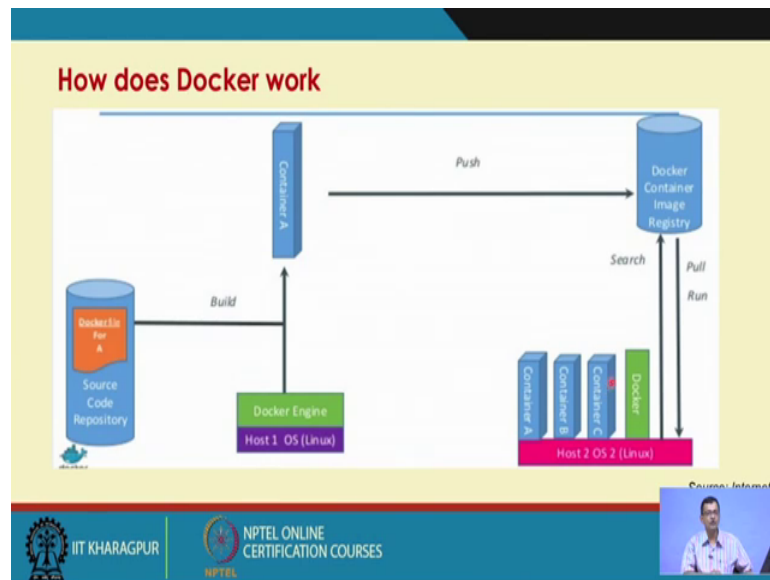
(Refer Slide Time: 26:35)



Whereas in this case slight modification can be done; allows us only to save the difference between container A and container A dash and that can be done at that container level.

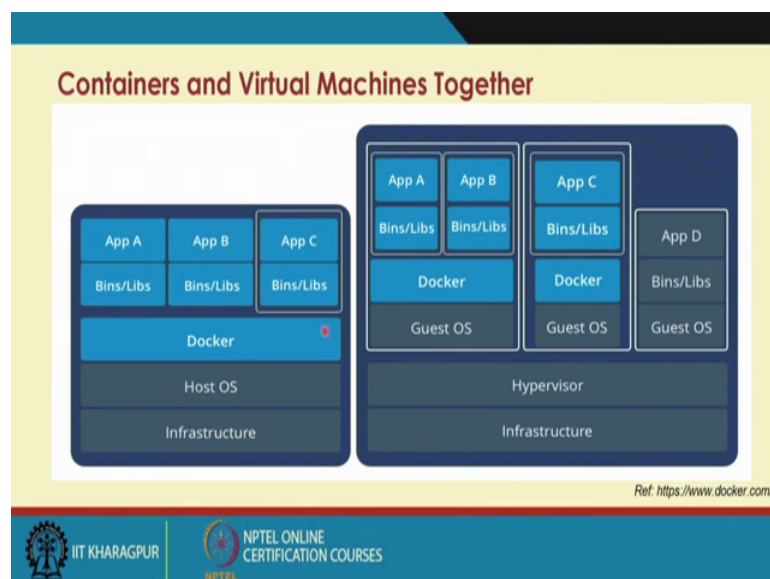


(Refer Slide Time: 26:58)



So, overall working as we have the Docker, as we have seen this is the Docker engine is there. So, from the source repository built over this Docker engine; it is a container, a particular container; class container A, we push it to this container image registry; which can be searched by the users or the consumer. This container finally, is basically this Docker is a some sort of a client server mode operation. So, it search and then pull this things and run on this particular instantiation of the different container classes.

(Refer Slide Time: 27:42)





So, if we try to look at all together like how it can run on the cloud. So, we have infrastructure hypervisor over that guest OS, we can have a Docker services; in this case we have due to different container here only one and so and so forth. So, the cloud can host or can become a platform for running this container type of services. So, that is on some VMs; that is in a IAAS type of cloud, we can run this sort of services.

(Refer Slide Time: 28:25)


**Why is Docker needed for applications?**

- Application level virtualization.
- A single host can run several spatial applications for utilization of resources.
- Build once, deploy anywhere, run anywhere.
- Better collaboration while development of applications.



Ref: <https://www.docker.com/>

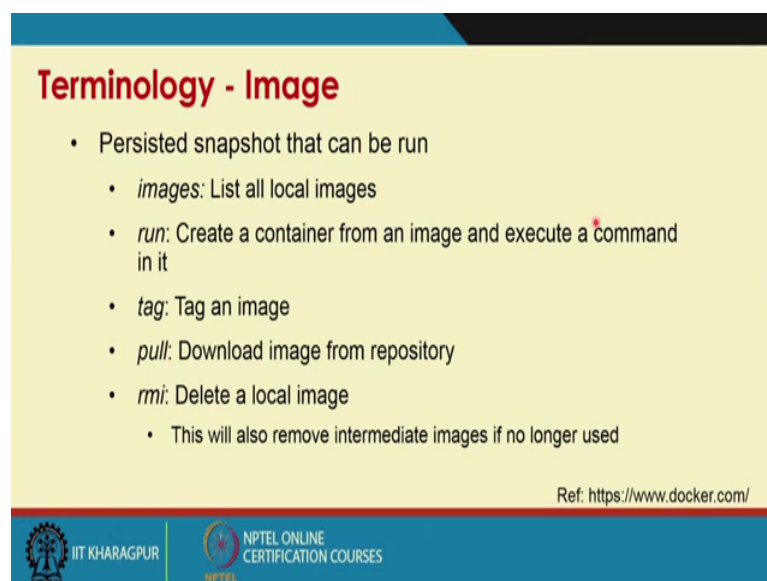
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, there are needs are enormous or needs, requirement, applicability is enormous like application level virtualization is possible. A single host can run several special application for utilization of resources; as we have seen when we discussed about special cloud there are several special applications and now we can have a single host which can run several special applications. Build once, deploy anywhere, run anywhere, type of things; philosophy. So, once I develop and build it and then I deploy anywhere, run anywhere. Better collaboration between developmental of applications, so I can have better collaborative applications into place.



(Refer Slide Time: 29:19)



### Terminology - Image

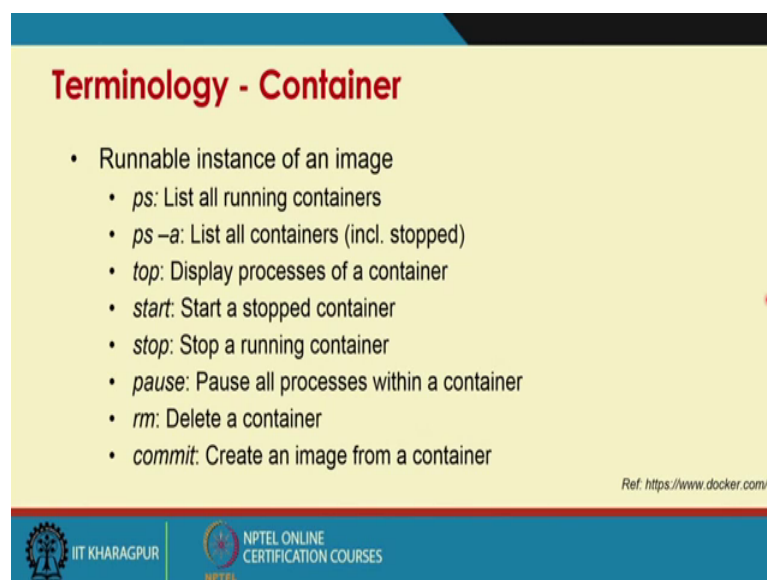
- Persisted snapshot that can be run
  - *images*: List all local images
  - *run*: Create a container from an image and execute a command in it
  - *tag*: Tag an image
  - *pull*: Download image from repository
  - *rmi*: Delete a local image
    - This will also remove intermediate images if no longer used

Ref: <https://www.docker.com/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

There are few terminologies some of them already you have seen. So, like images list of all local images, run; create a container from an image and execute a command in it. Tag an image, pull; download an image from a repository, rmi; delete a local image. So, this is also remove intermediate images if no longer is used so that resources can be released.

(Refer Slide Time: 29:48)



### Terminology - Container

- Runnable instance of an image
  - *ps*: List all running containers
  - *ps -a*: List all containers (incl. stopped)
  - *top*: Display processes of a container
  - *start*: Start a stopped container
  - *stop*: Stop a running container
  - *pause*: Pause all processes within a container
  - *rm*: Delete a container
  - *commit*: Create an image from a container

Ref: <https://www.docker.com/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

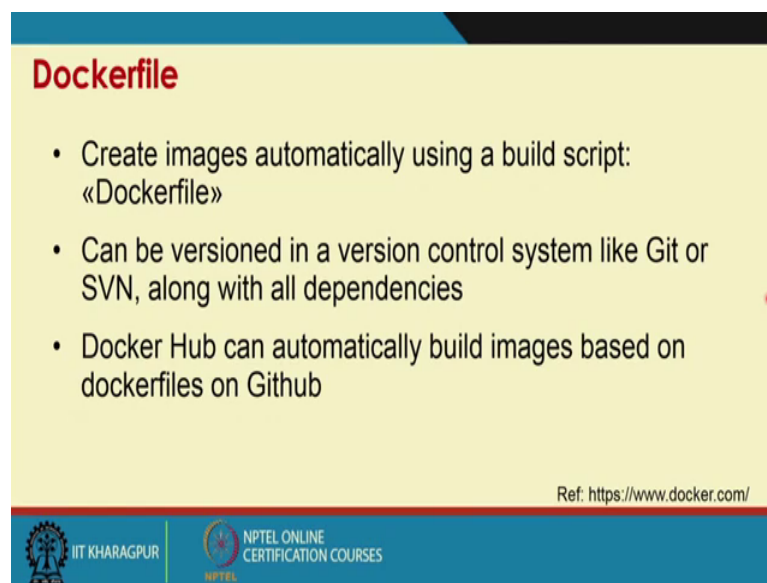
There are some terminology which are more associated with Docker container like ps; list all running containers, ps minus a; list all containers including the stopped one, top; display process in the container, start, stop, pause, rm; delete a container, commit; create



an image on the container. What you may notice that many of them things are some sort of Linux commands; already we are used to it.

So, in this case is also for container we can use those commands those who are interested you can basically hook into Docker dot com and see that how the coding can be done and how a container can be built using a Docker engine. So, this is freely downloadable and you can work and you can run that things on your desktop or server or even on your android devices, make a particular application and run on different devices.

(Refer Slide Time: 30:59)



**Dockerfile**

- Create images automatically using a build script: «Dockerfile»
- Can be versioned in a version control system like Git or SVN, along with all dependencies
- Docker Hub can automatically build images based on dockerfiles on Github

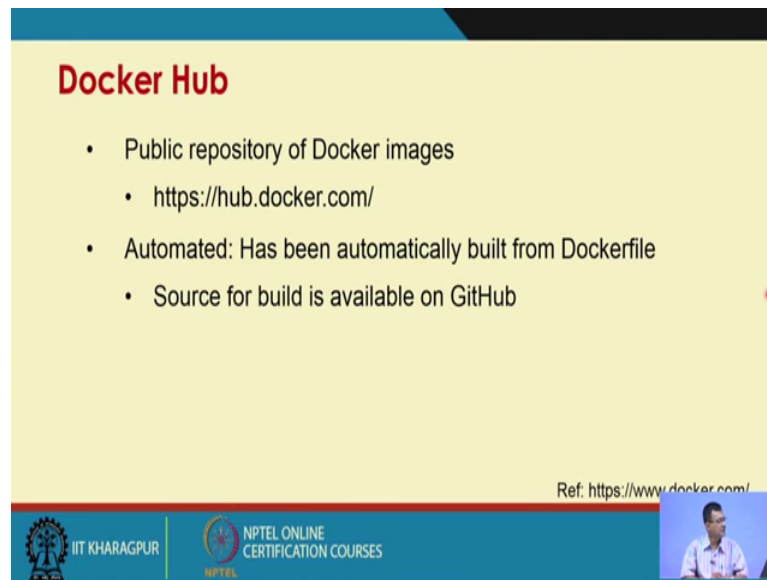
Ref: <https://www.docker.com/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, there are some few more things; one is the Docker file; create an image automatically using a build script called Docker file can be versioned in a version control system like Git or SVN, along with other dependencies. Docker hub can automatically build images based on Docker files on Github, so these things are possible.



(Refer Slide Time: 31:23)



**Docker Hub**

- Public repository of Docker images
  - <https://hub.docker.com/>
- Automated: Has been automatically built from Dockerfile
  - Source for build is available on GitHub

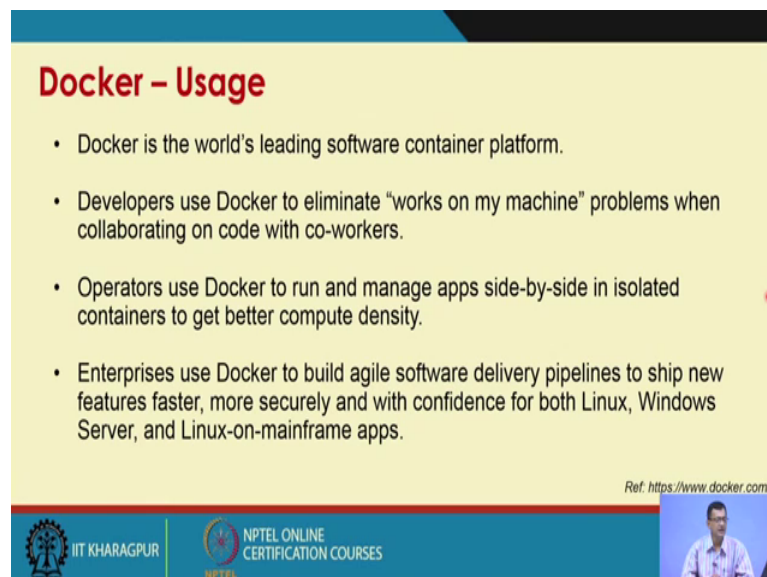
Ref: <https://www.docker.com/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The slide features a yellow background with a blue header and footer. A small video inset of a speaker is visible in the bottom right corner.

And the Docker hub; where the public repository of Docker images are there; like hub dot Docker dot com, there is a good resourceful area for this Docker images. And automated has been automatically built on Docker file. Source for build is available on the Github.

(Refer Slide Time: 31:44)



**Docker - Usage**

- Docker is the world's leading software container platform.
- Developers use Docker to eliminate "works on my machine" problems when collaborating on code with co-workers.
- Operators use Docker to run and manage apps side-by-side in isolated containers to get better compute density.
- Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely and with confidence for both Linux, Windows Server, and Linux-on-mainframe apps.

Ref: <https://www.docker.com/>

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The slide features a yellow background with a blue header and footer. A small video inset of a speaker is visible in the bottom right corner.

So, finally if we look at that different view point or different uses of this Docker is a worldwide claim to be leading software container platform being used at various levels, in different environment and became very popular. So, developers use Docker to



eliminate work on my machine problem. So, that is dependency on the things when collaboration on code with coworkers are much needed.

Operators use Docker to run and manage apps side-by-side in isolated container to get better compute density; that is the view point or the usage of the operators. Enterprises use Docker to build agile software delivery pipelines to ship new features faster, more securely with confidence for both Linux, Windows Servers, Linux-on-mainframe apps and so and so forth.

So, there are different applicability of the Docker and this services becoming pretty popular and those who are interested in this particular Docker technology, you may go through that Docker dot com and there are several other open sources; where you can see that how this compilation can be done and can have your own Docker images and run out on different platform and see the things. So, as far as the cloud computing paradigm is concerned; it acts as a platform and have seen application portability across cloud, making cloud more useful, more acceptable at different environment.

Thank you.