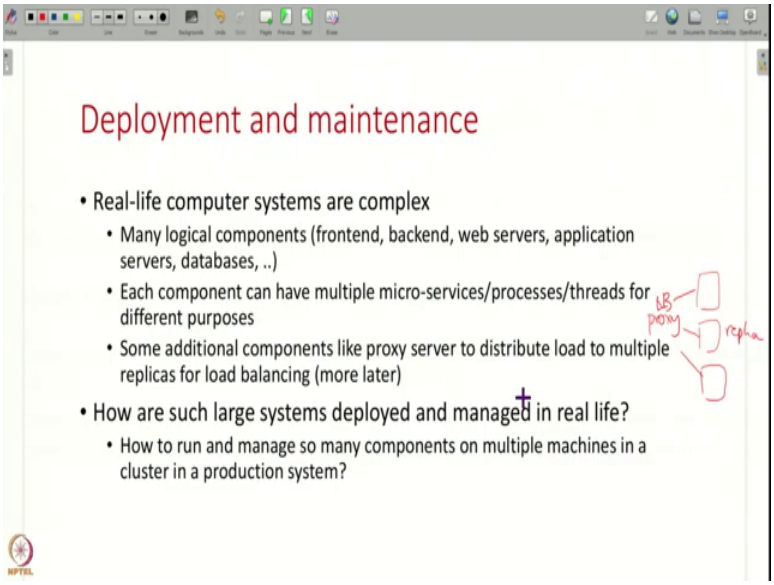
Design and Engineering of Computer Systems
Professor Mythili Vutukuru
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture 42
Deployment of Computer System

Hello everyone, welcome to the 30th lecture in the course Design and Engineering of Computer Systems. So, in the previous two lectures, we have looked at end to end application design, if you have to build a computer system, how do you go about designing it, how do you split it into different modules and how do you design these modules and so on.

So, now in this lecture, we are going to see how are all of these different components of a system deployed and managed in practice when the system is running in real life, how do you take care of all of these different components? So, we are going to study a little bit detail about what are the issues you face in deploying a system. So, let us get started.

(Refer Slide Time: 01:00)



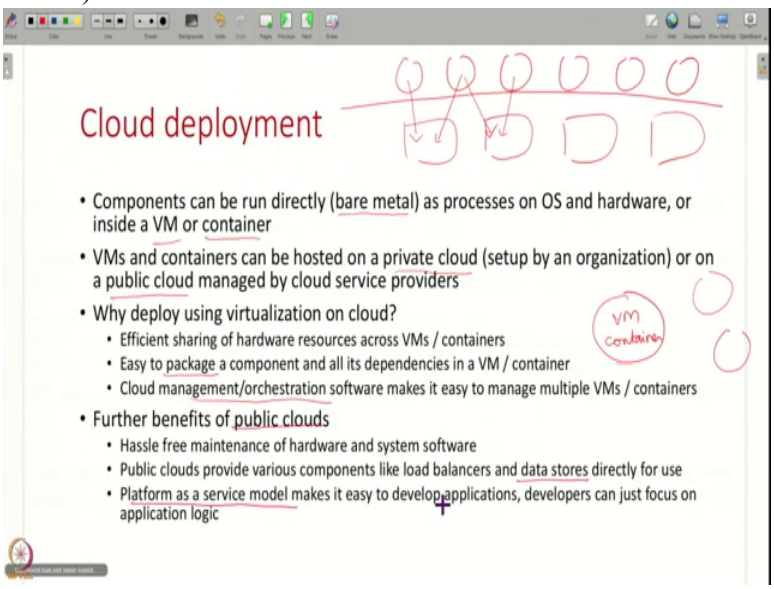
So, it must have been clear to you in the previous two lectures that real life systems are fairly complex, there are many components, we have just seen very toy examples where there like a handful of components that could fit in a slide. But real-life systems have hundreds of such components. And how do you manage all of these, there are front ends at various back end databases, web servers, application servers, all of these things.

And, each component can further have multiple sub components for the different micro services or different functionalities that can have separate sub components or processes or threads, or containers, and so on. And other thing you can have is sometimes one component might get overloaded. So, therefore you will have multiple replicas of the component, and some kind of a proxy server or a load balancer sitting in front distributing the load to all of these companies.

So, we will study this later when we study how to improve performance of systems. But this is a common technique used where, you do not have each of the application server as I showed you, it is not just running in one machine, but it is running in multiple machines with some kind of a load distribution. So, therefore not just what I have shown you, but there will be many more components in real life considering all of this multiple replicas and load balancers and so on.

So, how do you run and manage so many components you cannot be like, single human starting a process here, starting a process here, and taking care of all of these things, a lot of these things have to be automatically managed and automatically taken care of. So, in this lecture, what we are going to do is we are going to study some of the tools that are available in order to automatically deploy and manage the large number of components in a computer system.

(Refer Slide Time: 02:51)



So, the first thing to understand is a lot of real-life systems are not directly run on bare metal, not directly run as processes on the hardware itself, but they are run inside a VM or a container. And these VMs and containers are then hosted on clouds. So, you have a series of hardware servers,

and over this, you have a bunch of different components running on VMs or containers, and this VMs can be placed anywhere.

So, this virtualization layer takes care of these two VMs can be located on this server, and or this VM can move to another server in this way. These VMs and containers are placed on many different servers in the underlying infrastructure. And this infrastructure can either be a private cloud that is set up and managed only by one organization or it can be a public cloud that, many users can use over the internet.

So, you do not usually run components as bare metal directly as processes on the OS, but rather you run them inside VMs or containers of course, this is not always the rule, but this is what is typically done. So, why this virtualization and why deploying on a cloud, what are the advantages of it? One is you will have an efficient sharing of hardware resources, if one server is not able to fully use up all the CPU cores or one application is not able to use up all the CPU cores you can just start another VM also are another container also to the same server.

On the other hand, if one component is facing a lot of load, you can remove all the other VMs or containers and let this one component use up all the resources of the server. So, you will have more flexibility in sharing the hardware resources. If you put things inside VMs and containers because these VMs and containers can be easily moved across servers.

And the other advantages is easy to pack package a component and all its dependencies in a VM, or a container, a web server, it needs these libraries, it needs this software, all of those things you put together, you package them together all the application plus all its dependencies. And this entire piece, this VM, or this container can be moved around different machines without worrying about, does the server have the correct library or not.

So, in real life, these issues are very important. So, VMs, and container provide an easy way of packaging. Then the other thing is, we have seen this before that the cloud systems usually have a management or an orchestration software that makes it easy to manage these multiple different VMs on the underlying machines, which VM is there on which server? Where should I place it, how should I run it, all of these things are taken care of by some of the orchestration software.

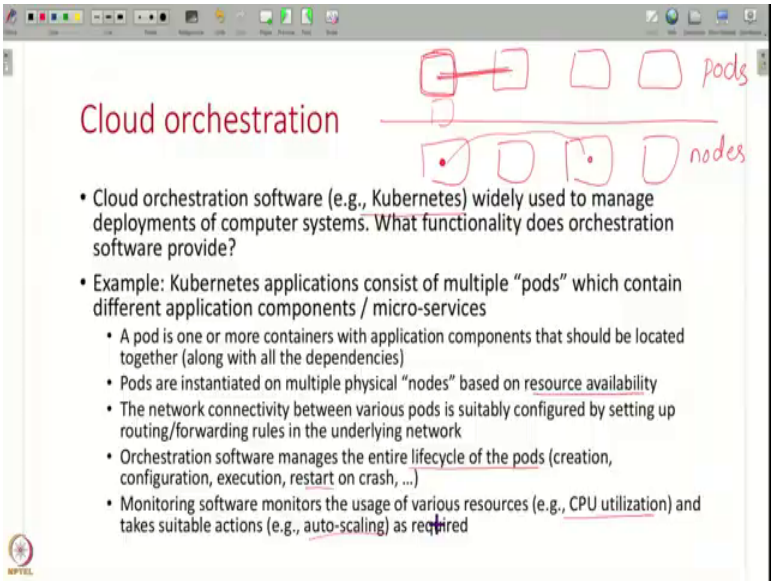
So, for all of these reasons, instead of directly, starting a process on the operating system, bare metal, you might want to build and deploy all of your application components inside VMs, or containers on a cloud. And what is more, if you do not manage this cloud yourself and use a public cloud, there are even more advantages. So, you do not have to worry about the hardware system software, OS upgrades, server upgrades, you do not have to worry about any of that the public cloud provider is taking care of all of that headache for you.

The other thing is public clouds provide you various components, like load balancers, data stores, if you use a public cloud, they will automatically have implementations of a few different data stores, whether traditional databases, no SQL data stores, graph data stores, they will already have some of these readymade components or load balancers. Some standard components will already be available for you to directly use, you do not have to reinvent or bring these components yourself into the cloud.

And also, clouds also have this platform as a service model, where you can just write your application logic, but the entire platform, the web server, the application server database, the IPC between them, coordination between them, all of that is also taken care of for you. So, you can just focus on, my application has to provide these services, how do I implement those services, just focus on the business logic.

So, that is also another advantage. So, overall, in real life, it is not like, you have these 100-200 components in an application and you manually start all of these components on different servers, you package them as VMs containers, and you just give it to a cloud management system that automatically takes care of managing all of these VMs and containers. So, the key part of, the key advantage of putting all of this in the cloud is cloud orchestration.

(Refer Slide Time: 07:51)



So, let us understand a little bit deeper into what are these cloud orchestration software and what kind of functionality do they provide? The common example of a cloud orchestration software that is very popular today is something called Kubernetes. Kubernetes is a software that lets you manage multiple components of your application. So, what developers will do is? They will build these different application components and place them into what are called pods.

A pod is nothing but a group of containers that contain some application components that should be located together, it can be one container or it can be more containers also, that are tightly packaged together in a pod. And then these pods are placed upon the underlying hardware, the servers in the system which are called nodes. So, some components will be placed on some server, some other components will be placed on some other servers and so on.

So, you as a user, you do not have to worry about this headache of which pod is placed on which server and so on, you just build these components, you just build the application. And once you give these parts to Kubernetes, Kubernetes will take care of all the logic like instantiating these pods on the nodes and if some server goes down moving the pods to another server. And the network connectivity, you want these two components to be connected, whereas this component is on this server, this component can be on some other server.

The underlying networking routes have to be established between the servers, all of that is taken care of by the orchestration software. Setting up all the routing forwarding tables in the underlying network. So, that you as a user, you just see this view that these two components are

connected to each other how they are connected the network configuration you do not worry about. Then the other thing that these orchestration softwares do is, they will place these pods on nodes based on resource availability.

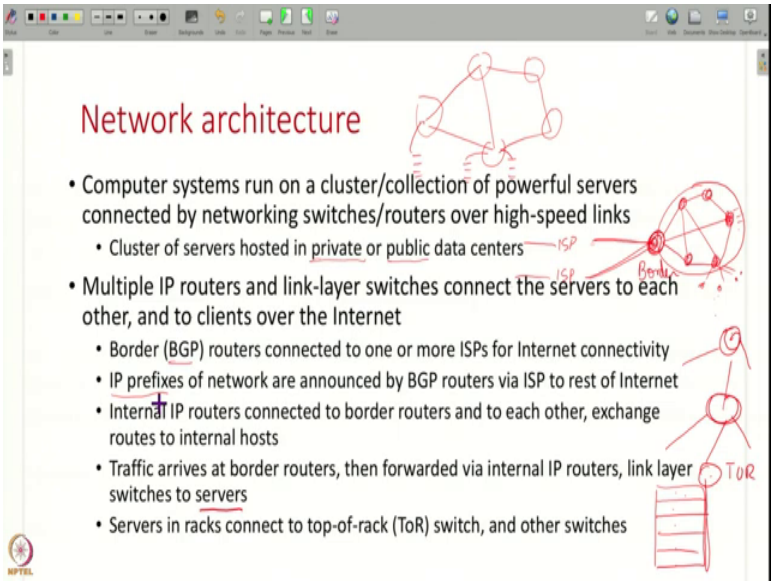
If there are not enough CPU cores free on this node, then a pod will be placed on another node, and what they will do is? They will also monitor this resource usage, is your CPU usage going too high? Is it going too low? They will monitor all of that and if one pod is using up all the CPU on this machine, then maybe you need to do auto-scaling, you need to have two different components that handle the load, all of that auto-scaling is also done.

Then the entire lifecycle of the pod is managed, creating a pod, you will have a nice user interface to create pods to execute pods to start pods to configure them, whatever configuration parameters you have to give, you can easily give if some application component crashes, it will be restarted. Load balancing auto-scaling, a lot of the babysitting that has to be done to these application components in a real system due to failures due to various other bugs, all of that complexity is handled by these orchestration softwares.

So, that you as a application developer can just focus on building the application and lot of the complexity in real life is taken care of by the orchestration software. Note that these orchestration systems are especially important in production. When production if your server suddenly sees a lot of traffic, you want to quickly scale it, you want to quickly have multiple other servers created to absorb this load, you do not have time for some user to come figure out the problem and take action, it has to be done automatically.

That is why the software automatically manages all of these things. Or if some server crashes, you want to quickly move that pod to another server. All of this is taken care of by orchestration software. So, I have just explained the high-level idea of Kubernetes. But there are many such orchestration systems and many such frameworks available to take care of real-life deployments of computer systems. So, now let us understand how the network is designed. All of these different servers are all connected by a network, how do you design this network?

(Refer Slide Time: 12:02)



So, any computer system runs on not just one machine, it usually runs on a cluster of servers. There are many servers, each of which are hosting one or more pods, as we have seen, one or more components of the application are being hosted on all of these servers. So, then how do you connect up all of these servers? Now, these servers can either be in a private cloud that only your organization uses, or the servers can be in a public cloud that many different applications many different systems are using.

Whatever it is, still the problem is the same that you have all of these servers in a large data center, and you have to connect them all up, how do you do it? So, here are some of the basic steps. So, first of all, you have many routers and switches that connect all of these servers to each other. So, there are different servers, they connect to these, different servers, talk to these routers, and switches, and these routers and switches, talk to each other run routing protocols and so on, to provide connectivity across all of these different servers.

So, we have seen in the networking part of the course as to how these routers work and how they run routing protocols and connect up all of these servers. Now, in an organization or in a data center where all the servers are hosted. So, there are some routers that are the border routers, so they are at the edge of the data center. And their job is to talk to other border routers in ISP networks and provide external connectivity.

So, BGP is the routing protocol used by border router. So, these routers are also called BGP routers, So, these BGP routers talk to other border routers in the ISPs. So, the ISPs will provide

network link to connect this data center and these border routers talk to these ISPs and they will announce I have all of these IP addresses and these IP addresses will be announced by the ISP to all the other rest of the internet.

So, that any traffic coming to this data center coming to the servers from the clients can reach this data center. So, the border routers take care of external internet connectivity by announcing IP prefixes and within the data center there are many other routers also, which are the internal IP routers which run some intra domain some other different routing protocols to connect amongst themselves, each internal IP router is managing a bunch of machines and it will say, I have these prefixes this guy will say I have some other prefixes and they exchange information with each other and form these routing tables and forwarding tables.

So, you have internal routers and you have border routers. So, when traffic comes to a computer system from say clients, these border routers have announced the IP addresses of the servers to through the ISP. So, data from clients will come from the ISPs to this border router, then the border router will send the data to the other internal routers, which will then send the data to other internal routers.

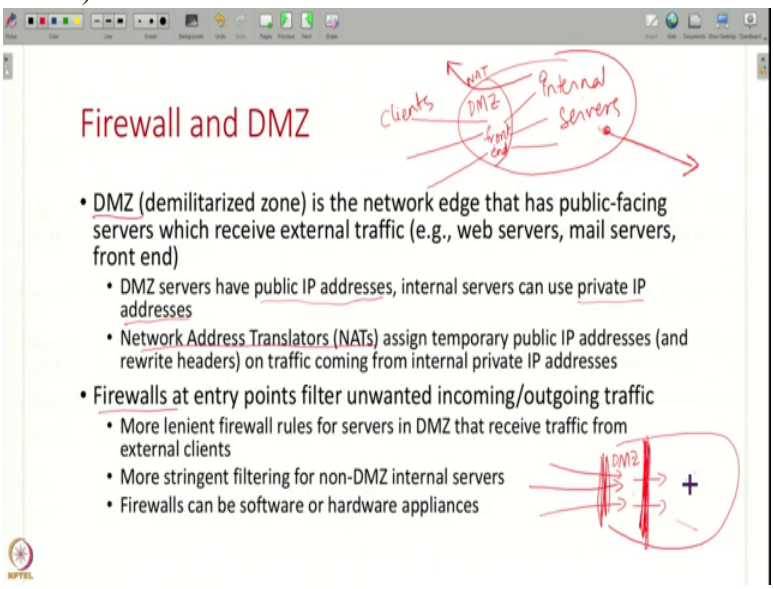
And finally, this is a hierarchical network, it will finally reach the server to which the client has sent the request. So, traffic arrives at the border routers, then it is forwarded via internal IP routers, then between two IP nodes on one IP link, you can have link layer switches through a series of network elements, the final request reaches the servers, the servers the front-end servers, back end servers, whatever all those servers are connected in this way.

So, the other thing to note is these servers typically today, they are blade servers, you have a rack, you have a cupboard kind of thing, if you looked inside data centers, anytime you have a cupboard kind of a thing, and each rack a server is placed, and all of these servers are connected by a top of the rack switch. And many such switches are connected to each other to some IP routers. And many such IP routers are connected to the border router and so on.

In this way, you will have a hierarchical network. And when traffic comes in, it will come to the border router, then it will go to the internal routers, and it will go to the link layers, switches the top of the rack switch. And finally, it will reach your server. So, the IP routers take care of all the IP routing, announcing your prefixes to the rest of the internet so that traffic can come from the

rest of the internet, everybody knows about you and can contact you through these series of routers. So, we have studied all of this in more detail in the networking part of the course, we have also seen how transport protocols work between the clients and the servers to reduce congestion through all of these routers.

(Refer Slide Time: 17:00)



So, the other concept is that of what is called a demilitarized zone, or DMZ in a network. So, in a network, what happens is the edge of the network is usually what is called the demilitarized zone. And this edge of the network has all the public facing servers, your web servers, your front end, that the clients talk to these web servers, email servers, all of these that see external connection from clients, they are placed in a separate area of the network that is called the demilitarized zone or the DMZ.

And these are the servers that typically have public IP addresses and so on. Then the other internal servers are placed inside the network. Why? Because these internal servers, you do not want to expose them to all the outside clients, the front end only will talk to the internal servers, you do not have clients directly connecting to your internal servers, therefore these internal servers will be placed in a more secure area of the network.

And the IP address allocation also differs, these public servers that are there in the DMZ. These will have public IP addresses, the web server will have an IP address that is published by a DNS so that the client can send a connect request and so on. On the other hand, these internal servers can just have private IP addresses only internally, this front end and internal to an organization only they have to communicate, you do not need to publish their IP addresses to any external clients.

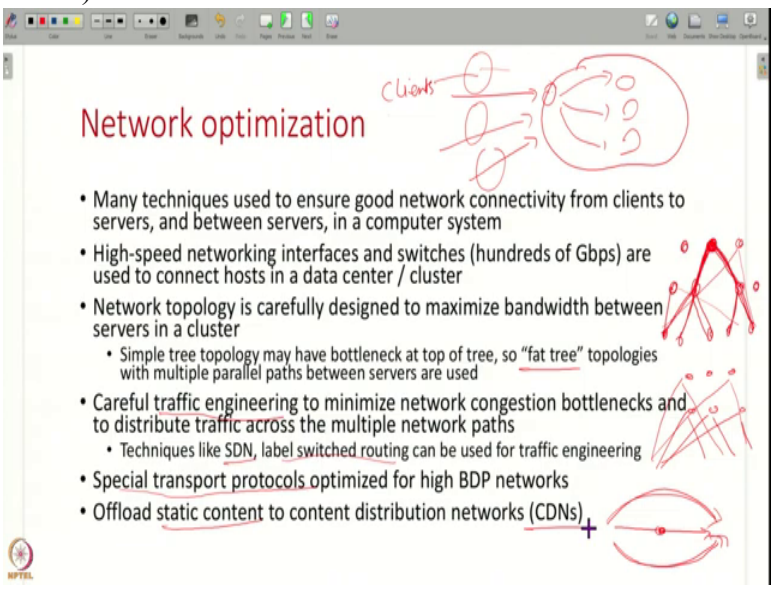
Because nobody from outside will talk to these internal servers. So, these internal servers can use private IP addresses. So, in this way, you can split your IP address allocation between the DMZ and the more secure internal parts. But sometimes if your internal servers also want to talk to the external world, then you can use network address translators are NATs , when a TCP connection is going out temporarily, for this client, some public IP addresses assigned that can happen via NATs.

So, any communication from the internal part of the network that is going out, it will go via a NAT where the NAT will temporarily assign a public IP address. Otherwise, these internal servers are usually isolated from the rest of the internet for security. And you also have firewalls to monitor all the traffic coming into a network. So, if you have a network, in this DMZ you will have a firewall here that is more lenient external requests are coming in.

So, this firewall will allow all sorts of traffic into the DMZ. But there will be another firewall here between your DMZ and your internal network that will be more strict because here you do not have random traffic coming in. You will only have traffic coming in from this front-end servers to your internal application server. So, this firewall will have more strict rules and ensure that your internal servers are protected.

So, your DMZ servers are facing the outside world and they can have more security attacks and so on. But your internal servers where all your application data is there, they are usually kept safer using stricter firewall rules. And of course, we have seen these firewalls they can be software or hardware appliances that basically filter traffic going in and out. So, this is how networks are designed, computer systems are designed where you have a DMZ, demilitarized zone where public facing servers are kept. And all your other internal servers that only your front-end servers will talk to, which only talk to each other, they are placed in a more secure area of the network.

(Refer Slide Time: 20:43)



And once you design a network this way, there are many optimizations also that are done in practice. Here, I will just briefly touch upon them understanding how to optimize these networks, these data center networks for large computer system, that is an active area of research, I would not be able to go into a lot of detail here. But I will just give you a high level some ideas of what you do in real life in order to make your network work better.

So, you have your clients that connect over the regular public internet, to your system. And there are some front-end servers and these servers talk to other internal servers. So, you have an internal network as well as you have the external network, the external internet. Both of these networks should be optimized, you want the internet to be fast, and you want your internal network also to be fast so that the clients can access all the services of your system quickly.

So, what do you do in order to provide good network connectivity, of course, you have to use high speed network cards and high speeds switches if your laptop may only have like, 1 Gbps or 10 Gbps network card. But internally, all of these systems have to exchange large amounts of data, large number of requests are therefore, it is today you have hundreds of Gbps of network interfaces, network cards, and switches to connect all of these different servers in a data center or a cluster.

And also, the network topology how are all of these servers connected to each other that is also carefully designed. So, if you have like a simple tree, kind of a topology, there is a bigger switch,

there is a smaller switch and multiple servers connected like this, if you have a simple tree like this, then if this server wants to send data to this server, this top of the tree usually becomes a bottleneck if everybody is talking to everybody else's servers are communicating with each other over various API's exchanging information, then if you do a simple tree, then the top of the tree becomes a bottleneck.

Everybody has all requests have to go through the top of the tree. So, instead, you will you have newer topologies to connect all of these different servers, you have what are called fact trees, that is you do not have just one path you have multiple switches at each level and this server is connected to this switch also and this server is connected to this switch also. So, you have like, what are called fact tree topologies that is multiple switches at each level.

And a server is connected to multiple switches, not just one switch, that way you have many different paths between two servers that are talking to each other in an application and these multiple parallel paths will basically increase the bandwidth available, and all of this intra component communication can happen much faster. So, this optimizing the network topology is one part of it.

The other part of it is traffic engineering. We have briefly seen this before. If you have multiple parallel paths between servers, if there are two different paths between two components, you want to send some traffic here, you want to send some traffic here, you want to do some load balancing, so that there is no congestion. If everybody is going through the same path, there will be congestion along this path, you want to spread your traffic around inside your data center, so that the servers can communicate quickly without congestion.

That is done we have seen techniques like software defined networking or label switched routing in the networking part of the course that will let you do traffic engineering. That is one component. The other thing is people are also working on special transport protocols. People are optimizing TCP so that this is a high bandwidth delay product network, you have very high bandwidth between components very low delay.

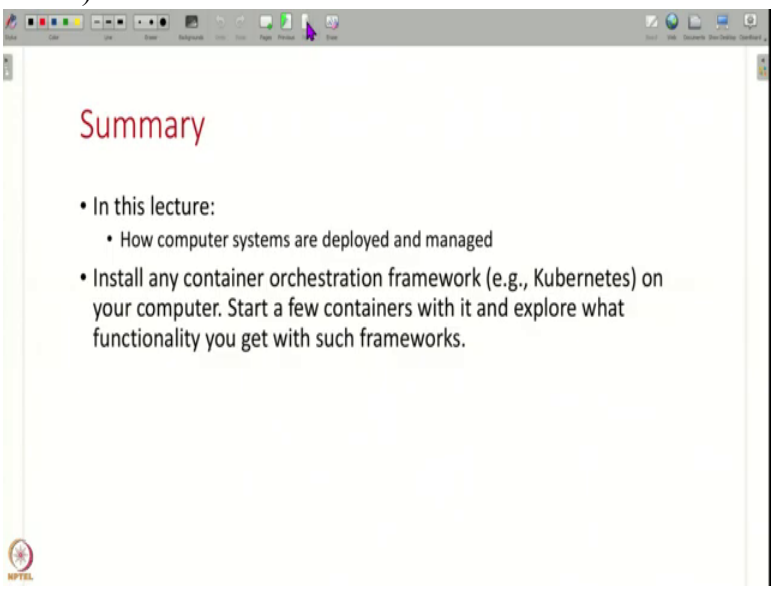
So, for these conditions, you want to optimize your transport protocol. So, you are also having special transport protocols today that work well inside data centers. And all of these techniques are how to optimize your internal network that connects to various components of your computer

system. Of course, there is also work on how to optimize the broader internet itself, so that when clients connect to servers, how to optimize this, that also people are working on and upgrading network switches, optimizing internet protocols so that they work faster and so on.

And the other thing you can think about is, if there is some content that you can offload and provide via CDN, so the clients can just get the content via content distribution networks, and do not have to come to the server all the way. So, static content, that can be distributed via content distribution network, that is also an optimization you can think about.

So, if there are, just images on your web page that can be served by a content distribution network. Only user specific information, you should go all the way to the data center where the computer system is hosted. So, these are all some of the techniques that are in use, in order to optimize the deployment of computer systems.

(Refer Slide Time: 25:58)



So, that is all I have in this lecture. In this lecture, I have given you a little bit of an idea of, once you build a computer system, how you deploy it, and how you manage it, how you take care of the network, and so on. Of course, it is a very complicated topic, I cannot go into all the details. But hopefully this has given you enough of an idea of what are some of the issues you have to deal in real life when you are actually running a real computer system.

So, as an exercise, you can try out one of these container or VM orchestration frameworks, like for example, Kubernetes, you can just install it locally in your laptop, start a few containers, and

just see what is the functionality that these orchestration frameworks provide? So, that is all I have for this lecture. Thank you all. And let us continue our discussion on a new topic of performance engineering next week. Thank you.