**Notes**

**Cloud Infrastructure**

**Overview of Cloud Infrastructure**

After choosing the cloud service model and the cloud type offered by vendors, customers

need to plan the infrastructure architecture. The infrastructure layer is the foundation of the cloud. This layer consists of physical resources that are housed in Regions, Zones and Data

Centers. A Cloud provider’s IT environment is typically distributed across many regions around the world. A cloud region is a geographic area or location where a cloud provider’s infrastructure is clustered and may have names like NA South or US East. The cloud regions are isolated from each other so that if one region was impacted by a natural disaster like an earthquake, the cloud operations in other regions would keep running. Each Cloud Region can have multiple Zones (or Availability Zones or AZ for short), which are typically distinct Data Centers with their own power, cooling and networking resources. These Zones can have names like DAL-09 or us-east-1. The isolation of zones improves the cloud’s overall fault tolerance, decreases latency, and avoids creating a single shared point of failure. The Availability Zones (and DataCenters within them) are connected to other AZs and regions,

private datacenters and the Internet using very high bandwidth network connectivity.

A cloud data center is a huge room or a warehouse containing cloud infrastructure.

These data centers contain pods and racks or standardized containers of computing resources such as servers, as well as storage, and networking equipment - virtually everything that a physical IT environment has.

**Computing Resources:** Cloud providers offer several compute options – Virtual Servers, Bare Metal Servers, and “Serverless” computing resources.

Most of the servers in a cloud datacenter run hypervisors to create virtual servers

or virtual machines (also called VMs for short), that are software-based computers, based on

virtualization technologies.

Other servers in the racks are bare metal servers that are physical servers that aren’t

virtualized. Customers can provision VMs and Bare Metals servers as and when they need them and run their workloads on them. Cloud users can also run their workloads on serverless computing resources, which are an abstraction layer on top of virtual machines.

We will talk about all three compute options in greater detail in subsequent videos.

**Storage:**

Information and data can consist of files, code, documents, images, videos, backups, snapshots, and databases and can be stored in many different types of storage options on the Cloud. Bare Metal Servers and Virtual Servers are provisioned with default storage in local

drives. Since these cloud servers can be provisioned and decommissioned by customers on demand and freed up for use by other users, any information stored in a local drive can be lost when you delete or decommission a cloud server.

However there are other storage options available on the cloud to persist data that you can

choose depending on factors like how important your data is, how quickly you want to be able to access it, how often you access it, and how secure you need it to be.

These additional storage options include Block storage, File storage, and Object storage.

Block and file storage modes are commonly used in traditional data centers, but often

struggle with scale, performance and distributed characteristics of cloud.

Object storage is the most common mode of storage in the cloud as it’s both highly distributed and resilient. We will examine Object Storage and the other storage options in more detail in later videos.

**Networking:**

Networking infrastructure in a cloud data center includes traditional networking hardware like routers and switches, but more importantly for users of the Cloud, the Cloud providers have Software Defined Networking (or SDN) options where certain networking resources are virtualized or made available programmatically, through APIs. This allows for easier network provisioning, configuration, and management in the cloud. When servers in the cloud are provisioned, you need to setup their public and private network interfaces. The public network interfaces, as the name suggests, connect the servers to the public internet, whereas the private ones provide connectivity to your other cloud resources and help keep them secure. As in the physical IT world, network interfaces in the cloud need to have IP addresses and subnets either assigned automatically or configured. In a cloud environment it is even more important to configure which network traffic and users can access your resources, which can be done by setting up Security Groups and Access Control Lists (or ACLs).

For further security and isolation of your resources in the cloud, most Cloud providers

provide Virtual Local Area Networks (VLANs), Virtual Private Clouds (VPCs), and Virtual

Private Networks (VPNs). Some of the traditional hardware appliances such as firewalls, load balancers, gateways and traffic analyzers can also be virtualized and made available as services in the cloud. Another networking capability provided by the Cloud Providers is Content Delivery Networks or CDNs, that distribute content to multiple points throughout the world so users accessing the content can access it more quickly by getting it from a point nearest to them. We will learn more about some of these cloud networking options and terminology in subsequent videos. Cloud infrastructure is constantly advancing and improving. In the next video, we explain virtualization and virtual machines.

**Virtualization and Virtual Machines Explained**

Virtualization is a fairly old technology, but it's still super relevant to building our cloud computing strategy today. So, first off, what is virtualization?

virtualization is the process of creating a software based, or virtual, version of something, whether that be compute, storage, networking, servers, or applications and what makes virtualization feasible, is something called the hypervisor. We're going to write that here.

**A hypervisor** is simply a piece of software that runs above the physical server, or host. And there are a couple different types of hypervisors out there.

**The use of a hypervisor:** It essentially pulls the resources from the physical server and allocate them to your virtual environments.

**Two main types of hypervisors:** One being Type 1. Very simple to remember. And 2, you guessed it, Type 2

**1. A Type 1 hypervisor** is a hypervisor that is installed directly on top of the physical server. They're also called bare-metal hypervisors. They are the most frequently type of use hypervisors and they're most secure, they lower the latency, and these are the ones that you'll see in the market the most. Some examples would be VMware, ESXi, or Microsoft Hyper-v, or even open-source KVM.

**2. A Type 2 hypervisor**: what makes these different is that there is a layer of host OS that sits between the physical server and the hypervisor. So, by that nature they are also called, Hosted. These are a lot less frequent. They're mostly used for end-user virtualization and you might see some in the market that are called: Oracle, VirtualBox, or VMware Workstation.  They have a higher latency than a Type 1 hypervisor. So once you have your hypervisor installed, you can build virtual environments, or virtual machines, or simply put, VMs. So let's spin up some environments. So, what makes a VM a VM? A VM is simply a software based computer. They're run like a physical computer. They have an operating system and applications, and they're completely independent of one another, but you can run multiple of them on a hypervisor and the hypervisor manages the resources that are allocated to these virtual environments from the physical server. So, because they're independent you can run different operating systems on different virtual machines. You could run Windows here, or Linux here, or UNIX here for example. Because they're independent they're also extremely portable. You can move a virtual machine from one hypervisor to another hypervisor on a completely different machine almost instantaneously, which gives you a lot of flexibility and a lot of portability within your environment. So looking at all of this - this is the core virtualization as a process. So let's talk about a couple key benefits that you want to take away from this.

**Benefits of Hypervisors**

1)Cost savings. When you think about this and the fact that you can run multiple virtual environments from one piece of infrastructure, means that you can drastically reduce your physical infrastructure footprint. This is consolidation at its core. And the fact that you don't have to maintain nearly as many servers, run as much electricity, save on maintenance costs, means that you save on your bottom line at the end of the day.

2)**Would be agility and speed**. Like I said, spinning up a virtual machine is relatively easy and quick - a lot more simple than provisioning an entire new environment for your developers if they say they want to spin up a new environment so that they can run a test scenario. Whatever it might be, virtualization makes that process a lot simpler and quicker

3) **lowers your downtime**. So, let's say that this host goes out unexpectedly. The fact that you can move virtual machines from one hypervisor to another, on a different physical server, means that you have a great backup plan in place right? So, if this host goes down

you can simply move your VMs very quickly to another hypervisor on a machine that is working. Virtualization and VMs are at the center of cloud computing and provide many benefits. In the next video, we will discuss the types of virtual machines.

**Types of Virtual Machines**

Virtual Machines or VMs are also known as Virtual Servers or Virtual Instances, or simply

Instances, depending on the cloud provider.

The various cloud providers make VMs available in a variety of configurations and deployment options to serve different use cases. When you create a virtual server in the cloud, you specify the Region and Zone or Data Center you want the server to be provisioned in and the Operating System you want on it. You can choose between shared (that is, a multi-tenant) VMs or dedicated (that is, a single-tenant) VMs. You can also choose between hourly or monthly billing, and select storage and networking options for the virtual server.

Now let’s look at a few different types of VMs that can be provisioned in the cloud.

Shared or Public Cloud VMs are provider-managed, multi-tenant deployments that can be provisioned on-demand with predefined sizes. Being multi-tenant means that the underlying physical server is virtualized and is shared across other tenants or users. To satisfy different workloads, cloud providers offer predefined sizes and configurations ranging from a single virtual core and a small amount of RAM to multiple virtual cores and much larger amounts of RAM. For example there can be configurations for Compute Intensive workloads, Memory intensive workloads, or High Performance I/O. Rather than pick from only pre-defined sizes, some providers also offer custom configurations that allow users to define the number of cores and RAM and local storage characteristics. Public VMs are usually priced by the hour (or in some cases even seconds) and configurations start as low as pennies per hour. Some providers also let you get monthly VMs, which can result in some cost savings if you know you will run the VM for at least a month, but if you decide to de-commision the VM in the middle of the month, you will still be charged for the full month.

**Types of Virtual Machines**

Transient or Spot VMs take advantage of unused capacity in a cloud data center. Cloud providers make this unused capacity available to users at a much lower cost than regular VMs of similar sizes. Although the Transient VMs are available at a huge discount, the Cloud provider can choose to de-provision them at any time and reclaim the resources for provisioning regular, higher-priced, VMs. Because you run the risk of losing these VMs when capacity in the data center decreases, these VMs are great for non production workloads such as testing and developing applications. They are also useful for running stateless workloads, testing scalability, or running big data and high performance computing (HPC) workloads at a low cost.

Reserved virtual server instances allow you to reserve capacity and guarantee resources for future deployments. You reserve desired amount of virtual server capacity, provision instances from that capacity when you need them, and choose a term, such as 1 year or 3 years, for your reserved capacity. You're guaranteed this capacity within the data center of your choice for the life of the contract term. By committing to a longer term, you can also lower your costs compared to hourly or monthly instances. This can be useful when you know you require at least a certain level of cloud capacity for a specific duration. If you exceed your reserved capacity, you can always choose to supplement your unplanned usage and capacity requirements with hourly or monthly VMs. Note however that not all predefined VMs families or configurations may be available as reserved.

Dedicated hosts offer single-tenant isolation. This means that only your VMs run on a given host so they can make exclusive use of full capacity and resources of the underlying hardware. When provisioning a dedicated host you to specify the data center and POD in which you want your host placed. You then assign instances, or virtual machines, to a specific host. This allows for maximum control over workload placement. Dedicated hosts are typically used for meeting compliance and regulatory requirements or meet specific licensing terms. Virtualization and VMs are at the center of cloud computing and provide many benefits. In the next video, we will discuss bare metal servers, what they are and what they provide.

**Bare Metal Servers**

A bare metal server is a single-tenant, dedicated physical server. In other words, it's dedicated to a single customer.

**Responsibilities**

The cloud provider actually takes the physical server and plugs it into a rack in a data center for customers. The cloud provider manages the server up to the operating system or OS, which means if anything goes wrong with the hardware or rack connection, they will fix or replace it and then reboot the server.

**The customer is responsible** for administering and managing everything else on the server. Bare metal servers are either preconfigured by the cloud provider to meet workload packages or they can be custom-configured as per customer specifications. This includes the processors, RAM, hard drives, specialized components, and the OS. Customers can also install their own OS and can install certain hypervisors that aren't available from the cloud provider, and thus create their own virtual machines and farms. With bare metal servers you can also add GPUs, which are designed for accelerating scientific computation, data analytics, and rendering professional grade virtualized graphics. Because bare metal servers are physical machines, they take longer to provision than virtual servers. Pre-configured builds of bare metal can take 20 to 40 minutes to provision and custom-builds can take around three or four hours, but these provisioning times can vary by Cloud provider. As Bare Metal servers are dedicated for use by a single client at any given time, they tend to be more expensive than similarly sized Virtual Machines. Also note that unlike virtual servers, not all cloud providers provide Bare Metal servers. Since bare metal servers are fully customizable, they can do what a customer wants in the most demanding environments. Bare metal servers are dedicated and intended for long term, high performance use in highly secure and isolated environments.

Clients have full access and control of bare metal servers because there’s no hypervisor

required. As there is no sharing underlying server hardware with other customers, Bare metal servers fulfil the demanding needs of high-performance computing or HPC and data intense applications that require minimal latency-related delays.

These servers also excel in big data analytics applications and GPU-intensive solutions.

Some workload examples that bare metal servers satisfy are ERP, CRM, AI, Deep Learning, and virtualization. If you use any applications that require high degrees of security control or apps that you’ve typically run in an on-premises environment, then a bare metal server is a good alternative in the cloud. When comparing bare metal servers to virtual servers, some of the most important considerations are found in customer need.

**compare bare metal servers with virtual machines**

 Bare metal servers work best for: CPU and I/O intensive workloads, excel with highest performance and security, satisfy strict compliance requirements, and offer complete flexibility, control and transparency but come with added management and operational overhead. Whereas, virtual servers are rapidly provisioned, provide an elastic and scalable environment, and are low cost to use, however since they share underlying hardware with other virtual servers, they can be limited in throughput and performance.

**Secure Networking in Cloud**

As cloud environments gain greater adoption, and digital data invites rapidly increasing

cybersecurity threats, building secure networks on the cloud is crucial. Let’s look at how we can build a secure cloud networking presence. As one might expect, the notion of building a cloud network is not much different from deploying a network in an on-premises data center.

The main difference stems from the fact, that in the cloud, we use logical instances of

networking elements as opposed to physical devices.

For example, Network Interface Controllers (NICs) would be represented by vNICs in cloud

environments. In the cloud, networking functions are delivered as a service rather than in the form of rack-mounted devices.

**Steps to create a network in the cloud**

one starts by defining the size of the network, or the IP address range that establishes the boundaries or the cloud network. Cloud networks are deployed in networking spaces that are logically separated segments of the networks using options, including Virtual private Cloud (VPC) that in turn can be divided into smaller segments called subnets. Logically segmented cloud networks are private carveout of the cloud that offer customers the security of private clouds and the scalability of public clouds. Cloud resources, such as VMs or Virtual Server Instances (VSIs), storage, network connectivity and load balancers are deployed into subnets.

Using subnets allows users to deploy enterprise applications using the same multi-tier concepts used in on-premises environments. Subnets are also the main area where security is implemented in the cloud. Every subnet is protected by Access Control Lists (ACLS) that serve as a subnet-level fire wall. Within the subnet, one could create Security Groups that provide security at the instance level such as VSIs. Once you build a subnet, then it is time to add some VSIs and storage to it so that you could run your applications.

Let’s say you have a 3-tier application that requires web access VSIs, applications

tier VSIs and backend database VSIs. In this case, we would place the web facing VSIs into one Security Group, the Application VSIs in a second Security Group, while the database VSIs in a third Security Group. It goes without saying that the web-facing VISs need Internet access.

A public Gateway instance is added to the network to enable users’ access to the application

in the internet tier. While public gateways are great for Internet access to the cloud, enterprises are interested in extending their on-premises resources to the cloud by securely connecting them using Virtual Private Networks, or VPNs.

When building many subnets and deploying several workloads, it becomes necessary to ensure that applications continue to be responsive. That is achieved with Load Balancers that ensure availability of bandwidth for the different applications.

Enterprises with hybrid cloud environment find using dedicated high-speed connections

between clouds and on-premises resources is a more secured and more efficient way than

public connectivity solutions. Some cloud service providers offer such connectivity, such as IBM Cloud and its Direct Link solution that enables extending on-premises resources to the cloud as needed. Building a cloud network entail creating a set of logical constructs that deliver networking functionality that is akin to the data center networks that all IT professionals have come to rely on for securing their environments and ensuring high performing business applications. In the Next video we’ll look at containerization technology and why containers have become a de-facto element of Cloud Native computing.

**Containers**

Containers are an executable unit of software in which application code is packaged, along

with its libraries and dependencies, in common ways so that it can be run anywhere, whether

it be on desktop, traditional IT, or the cloud. Containers are small, fast, and portable

, and unlike virtual machines, they do not need to include a guest OS in every instance and can, instead, simply leverage the features and resources of the host OS.

In the rest of this video, we will see how container-based technology really works. Hi everyone. My name is Sai Vennam and I'm a developer advocate with IBM. Today I want to talk about containerization. Whenever I mention containers, most people tend to default to something like Docker or even Kubernetes these days. But container technology has actually been around for quite some time. It's actually back in 2008 that the Linux kernel introduced C groups, and control groups, that basically paved the way for all the different container technologies we see today. So that includes Docker, but also things like Cloud Foundry, as well as Rocket and other container runtimes out there. Let's get started with an example, and we'll say that I was a developer. I've created a node.js application and I want to push it into production. We'll take two different form factors to kind of explain the advantages of containerization. Let's say that first we'll talk about VMs, and then we'll talk about containers. So, first things first, let's introduce some of the things that we've got here.

We've got the hardware itself, just a big box. We've got the guest, or rather, the host, operating system, as well as a hypervisor. Hypervisor is actually what allows us to spin up VMs. Let's take a look at this shared pool of resources with the host OS and hypervisor.

We can assume that some of these resources have already been consumed.

Next, let's go ahead and take this .js application and push it in. And to do that, I need a Linux VM. So let's go ahead and sketch out that Linux VM. In this VM there's a few things to note here. So, we've got another operating system, in addition to the host OS, it's gonna be the guest OS, as well as some binaries and libraries. That's one of the things about Linux VMs, that even though we're working with a really lightweight application, to create that Linux VM, we have to put that guest OS in there, in a set of binaries and libraries. That really bloats it out.

In fact, you know, I think the smallest node .js VM that I've seen out there is 400 plus mega

bytes, whereas the node.js runtime and app itself would be under 15. So we've got that and we'll go ahead and let's push that .js application into it. Just by doing that alone, we're gonna consume a set of resources.

Next, let's think about scaling this out. Right. So we'll create two additional copies of it, and you'll notice that even though it's the exact same application, we have to use and deploy that separate guest OS and libraries every time. And so we'll do that three times. And by doing that, essentially, we can assume that for this particular hardware we've consumed all of the resources.

And there's another thing that I haven't mentioned here, but this .js application, I developed

it on my MacBook. So when I pushed it into production to get it going on the VM, I noticed that there were some issues and incompatibilities. This is the kind of foundations is big, he said, she said issue. Where things might be working on your local machine, and work great, but when you try to push it into production, things start to break. and this really gets in the way of doing agile DevOps, and continuous integration and delivery. That's solved when you use something like containers. There's a three-step process when kind of doing anything container related, and then pushing, or creating, containers.

1.Starts with first, some sort of a manifest. So something that describes the container itself. So in the Docker world, this would be something like a Docker file and in Cloud Foundry, this would be a manifest Channel.

2.Next, what you'll do is create the actual image itself. So for the image, again, if you're working with something like Docker, that could be something like a Docker image. If you're working with Rocket it would be an ACI or application container image. So regardless of the different containerization technologies, this process stays the same.

3.The last thing you end up with is an actual container itself. You know, that contains all of the runtimes, and libraries, and binaries needed to run an application. That application runs on a very similar set up to the VMS, but what we've got on this side is, again, a host operating system.

The difference here, instead of a hypervisor, we're gonna have something like a runtime

engine. So if you're using Docker this would be the Docker engine and, you know, different containerization technologies would have a different engine. Regardless, it's something that runs those containers. Again, we've got this shared pool of resources, so we can assume that that alone consumes some set of resources. Next, let's think about actually containerizing this technology.

**Three-step process of containerization.**

1.We create a docker file.

2.We build out the image.

3.We push it to a registry, and we have our containers, and we can start pushing this out as containers.

The great thing is, these are going to be much more lightweight. So deploying out multiple containers, since you don't have to worry about a guest OS this time, you really just have the libraries, as well as the application itself.

So we've scaled that out three times, and because we don't have to duplicate all of those operating system dependencies and create bloated VMs, we actually will use less resources.

So use a different color here... and scaling that out three times, we still have a good

amount of resources left.

Next, let's say that my coworker decides, hey, for this .js application, let's take

advantage of a third party you know let's say a cognitive API - to do something like image recognition. Let's say that we've got our third party service, and we want to access that using maybe a Python application. So he's created that service that acts as that third party APIs

and with our node.js application, we want to access that Python app, to then access

that service. If we wanted to do this in VMs, I'm really tempted to basically create a VM out of both the .js application and the Python application because essentially that would allow me to continue to use the VMs that I have. But that's not truly cloud native, right? Because if I wanted to scale out the .js, but not the Python app, I wouldn't be able to if they were running in the same VM. So to do it in a truly cloud native way, essentially I would have to free up some of these resources. Basically get rid of one of these VMs, and then deploy the Python application in it instead. And you know, that's not ideal. But with the container based approach what we can do is simply say, since we're modular, we can say, okay, just deploy one copy of the Python application. So we'll go ahead and do that. There's a different color here. And that consumes a little bit more resources and then with those those remaining resources, the great thing about container technology, that actually becomes shared between all the processes running. In fact, another advantage if some of these container processes aren't actually utilizing the CPU or memory, all of those shared resources become accessible for the other containers running within that hardware. So with container-based technology, we can truly take advantage of cloud native based architectures. We talked about things like portability of the containers. We talked about how it's easier to scale them out.

And then overall, with this kind of three-step process and the way we push containers, allows

for more agile devops and continuous integration and delivery. Containers streamline development and deployment of Cloud Native applications. In the next lesson, we will cover cloud storage.

Lesson Summary

**In this lesson, you have learned:**

Cloud infrastructure consists of data centers, storage, networking components, and compute resources.

Virtualization is the process of creating a software-based version of physical resources, made possible through the use of hypervisors.

A few different types of Virtual Machines can be provisioned on the cloud. These include:

* Shared or Public Cloud VMs that are provider-managed, multi-tenant deployments that can be provisioned on-demand with predefined sizes
* Transient or Spot VMs that take advantage of unused capacity in a cloud data center
* Reserved VMs that allow you to reserve capacity and guarantee resources for future deployments
* Dedicated hosts that offer single-tenant isolation

Bare metal servers are single-tenant physical servers that are dedicated to a single customer. Bare metal servers fulfill the demanding needs of high-performance computing (HPC) and data intense applications and are ideal for applications that have a high degree of security or compliance requirements.

Networking capabilities in the cloud are delivered as a service rather than in the form of rack-mounted devices. Cloud resources, such as VMs (or VSIs), storage, network connectivity, and load balancers, are deployed into subnets within Virtual Private Clouds (VPCs). Using private and public subnets allows users to deploy multi-tier enterprise applications securely. Load balancers distribute the traffic and allow applications to be responsive.

Containers are an executable unit of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere—desktops, traditional IT, or the cloud. Containers are lighter weight and consume fewer resources than Virtual Machines - helping streamline the development and deployment of cloud native applications.

**Cloud Storge and content Delivery Network**

**Basics of Storage on Cloud**

Cloud storage is where you save data and files in the cloud.

Certain storage must be attached to a compute node before the storage can be accessed, whereas other storage types can be directly accessed either through the public Internet or a dedicated private network connection. Cloud providers host, secure, manage, and maintain the cloud storage and associated infrastructure to ensure you have access to your data when you need it. Cloud storage services allow you to scale your capacity as you need so you only pay for what you provision, usually on a ‘per gigabyte’ basis. The cost of storage will vary by type but in general, the faster the read / write speed of the storage, the higher the per gigabyte cost. Cloud storage is available in four main types – Direct Attached, File Storage, Block Storage and Object Storage. Direct Attached storage, sometimes referred to as ‘Local Storage’, is storage which is presented directly to a cloud-based server and is effectively either within the host server chassis or within the same rack. This storage is fast and normally only used to store a server’s operating system, although it can have other use cases.The main two reasons why direct attached storage is not so great for other uses besides to store the operating system is that it’s typically ‘Ephemeral’ , meaning that it only lasts as long at the compute resource it’s attached to – it cannot be shared with other nodes and while you can use RAID techniques, it’s not as resilient to failure as other types of storage. File storage is typically presented to compute nodes as ‘NFS Storage’. NFS stands for Network File System and means that the storage is connected to compute nodes over a standard ethernet network. NFS-mounted storage is common-place but it tends to be slower than either direct-attached storage or block storage because the data travels over an ethernet network.

It also tends to be lower cost than either direct attached or block storage.

One advantage of File Storage is that it can be mounted or used on multiple servers at

once. File-based storage is a simple, straightforward approach to data storage and works well for organizing data in a hierarchical folder structure, that desktop users are familiar with.

Block storage is presented to compute nodes using high-speed fibre connections, which

means that read and write speeds are typically much faster and reliable than with file storage, making block storage suitable for use with databases and other applications where disk speed is important. You typically provision block storage in ‘volumes’, which can then be mounted onto a compute node, which it then effectively sees as another hard drive.

Volumes can normally only be mounted onto one compute node at a time. With both File and Block storage, you may also hear the term ‘IOPS’. IOPS stands for ‘Input/Output Operations Per Second’ and relates to the speed of the storage or to put it another way, how quickly data can be read from or written to the storage. We’ll cover this in a little more detail in a later video. Persistence is a term that is used when provisioning File or Block storage and relates to what happens to the storage once the compute node it is attached to is terminated.

If the storage is set to ‘persist’ then it will not be deleted along with the compute

node, meaning that it and its data are preserved and available to mount onto another compute node, though you will continue to pay for the storage.

You can also, in some cases, set the storage so that it is automatically deleted with the

compute node that it is mounted onto – in this case, as we know, it becomes Ephemeral Storage. Here, you will also stop paying for the storage but you will lose any data unless it is backed up somewhere. There are several ways to backup data in the cloud but one way to back up both File and Block storage is to take a Snapshot. As the term implies, this is a point in time image of the storage. Snapshots are usually fast to create (they don’t actually write any data, or rather they create metadata), don’t require downtime and subsequent snapshots record only changes to the data. They are great for returning storage to the way it was at a particular snapshot, though note, they cannot be used to recover individual files.

The fourth kind of storage is Object storage. This is a different type of storage in so much as it’s not attached to a compute node, rather it is accessed via an API. Of all the storage types, Object Storage is by far the cheapest and also the slowest in terms of read and write speeds, but it is infinite in size to the end user. Unlike File and Block storage where you provision a certain storage capacity and it fills upover time, with Object Storage you can keep adding files to it and it never fills up, you just pay for what you use. This makes Object Storage a fantastic repository for all sorts of unstructured data types, large and small, including documents, video, logs, backups, data from IoT, application binaries and virtual machine images.

In the following videos, there will be more detailed information on the different types

of storage.

**File Storage**

n this video, we’re going to talk about File Storage in greater detail.

Like direct attached storage, file storage must be attached to a compute node before

it can be accessed and have data stored on it. However, File Storage can be less expensive, more resilient to failure, and involve lesser disk management and maintenance for you as the user to do , as compared to direct attached storage. You can also provision much larger amounts of File Storage and present it as a disk to a server. File storage is mounted from remote storage appliances. That is, the physical disks are contained in a separate, specialised piece of hardware and they are then connected to the compute node via the underlying infrastructure in the datacenter. These storage appliances are not only extremely resilient to failure, the data is also far more secure in them as these storage appliances offer services such as encryption in transit and encryption at rest. These appliances are all managed by the service provider. File Storage is mounted to compute nodes via an ethernet network – the same kind of network that you might receive email or browse the internet over, although this ethernet network is normally dedicated to the task. This means it can sometimes be referred to as ‘Network Attached Storage’, ‘Network File Storage’ or simply ‘NFS. One of the issues with ethernet networks is that their speed can vary – the more loaded an ethernet network is, the more likely it becomes that it’s speed or bandwidth will be affected. Of course, Cloud Providers build their storage networks to handle very high volumes of traffic. But even so, consistent speed cannot be guaranteed. Therefore, File storage tends to be used for workloads where consistently high network speeds are not a requirement.

In terms of workloads, File Storage can typically be mounted onto more than one compute node at a time, where the mounted disk or volume looks just like another drive on the compute node. The ability for File Storage to be mounted to multiple compute nodes at a time make it an ideal solution where some sort of common storage is required – for example, a departmental file share, a ‘landing zone’ for incoming files that need to be processed by an application, or a repository of files that a web service might access. In these applications, the potential variance in the speed of the connecting network is not really an issue.

Of course, where cost is an issue, you can use file storage for other applications such as databases, but the trade-off is speed. When you provision file storage, one consideration you need to take into account is the IOPS capacity of the storage. IOPS stands for Input/Output Operations Per Second and refers to the speed at which the disks can write and read data (note this is not the speed of the network between the storage and the compute node).

The higher the IOPS value, the faster the speed of the underlying disk. A higher IOPS will also normally cost more. Understanding IOPS is important because if the IOPS value is too low for your application, the storage can become a bottleneck and cause your application to run slowly. Alternatively, if the IOPS is too high, you will probably be paying more that you need

to for your storage. For example, a file share may be mounted on 30 different compute nodes and an application writes and requests data to and from that share 60 times per minute.

You can average that out to 1 operation per second. With this simple example, you can see that each application has different IOPS requirements.

In the next video, we’re going to talk more about Block Storage, how it compares with

File Storage, and when you would typically use one over the other.

**Block Storage**

In this video, we will discuss Block Storage and how it compares to File Storage in the

Cloud. Block storage breaks files into chunks (or blocks) of data and Stores each block separately under a unique address. Like direct attached storage and file storage, block storage also must be attached to a compute node before it can be utilized for your workloads. Block storage, like file storage, can be mounted from remote storage appliances, making it extremely resilient to failure, and keeping data far more secure in them, on account of encryption in transit, and encryption at rest services, available on these appliances.

Block storage is mounted as a volume to compute nodes using a dedicated network of fibres,

through which signals move at the speed of light. These fibre optic networks are more expensive to build than the ethernet ones which deliver File Storage, which is one reason why Block Storage tends to have a higher price-point. However, since the traffic is moving faster and with speed consistency, they are perfect for workloads that need low-latency storage to work effectively. In terms of workloads, it is important to note that unlike File Storage, which can be mounted onto 80 computer nodes or more, Block storage is normally mounted onto only one compute node at a time. Since these disks run at a consistent high speed, they are perfect for workloads that need consistently fast storage, such as databases and mail servers. Block storage is not suitable for workloads where there needs to be some level of disk sharing between compute nodes. For block storage, as it is for file storage, you need to take the IOPS capacity of the storage into account. Most cloud providers will allow you to specify IOPS characteristics when you provision storage and, in some cases, adjust the IOPS of your storage as you need, so if the requirements or usage behaviour of an application changes, you can adjust accordingly. So, to summarise the commonalities and differences between these two storage types: Block and File Storage is taken from appliances which are maintained by the service provider. Both are normally highly available and resilient and will often include data encryption at rest and in transit. File storage is attached to compute nodes using an ethernet network, so it is sometimes called Network attached or NFS Storage.

File storage is very reliable, but the speed of the connecting network can vary, based

on load. Block storage is attached via a high-speed fibre network, which is very reliable and

consistent. File storage can be attached to multiple compute nodes at once. Block storage can only be attached to one node at a time. File storage is a good choice where file shares are required, where workloads do not require lightning-fast connectivity to storage, or where cost is a factor. Block storage is a good choice when supporting an application that needs consistent fast access to disk, such as databases. Remember to consider the IOPS requirements of the application when provisioning either file or block storage.

In the next video, we’ll start to look at Object Storage.

**Object Storage Overview**

In this video, we’re going to start to understand what Object Storage is, how data is stored

in Object Storage, and how it differs from the more traditional storage types such as File and Block Storage. The first thing to note about Object Storage is that you do not connect it to a particular compute node in order to use it. Instead, you provision an Object Storage service instance and use an API (or Application Program Interface) to upload, download, and manage your data. This means you can directly use Object Storage with anything that can call an API and you don’t need an underlying compute node. The second thing to note about Object Storage is that it’s less expensive that other cloud storage options. It’s per gigabyte cost is typically a couple of US cents per month and in some cases, even less, depending on the storage tier used. More on storage tiers later. The third and possibly most important thing to note about Object Storage is that it’s effectively infinite. With file and block storage, you specify the size of the storage you want in gigabytes or terabytes and then pay a fee based on the size you provisioned. With Object Storage, you just consume the storage you need and pay per gigabyte cost for what you use. You can keep uploading files and the storage will never run out. So, when would you use Object Storage? Well, Object Storage is great for storing large amounts of unstructured data. By unstructured this means that the data is not stored in any kind of hierarchical folder or directory structure – Object Storage uses ‘buckets’, and objects are stored within these buckets in a structurally flat way.

A bucket is a bit like a folder, in the sense that you can give them meaningful names, and

of course have different buckets for different object-types but you cannot place a bucket

within a bucket. When an object is placed in a bucket, it also has some metadata (data about the data) added to it, such as an object ID. This metadata helps applications to both locate and access the object, as well as provide information on the time that the data was stored or last accessed. When you create a bucket, you don’t need to provide or define any sizing information. The bucket will just hold the data that you place inside it and the service provider ensures that there is sufficient storage capacity available. Buckets can hold as little as a few bytes of data, right up to multiple petabytes and you can build up the amount of data stored as slowly or quickly as you like, as well as shrink it back down again.

The service provider also takes care of resilience and making sure that the Object Storage solution is highly available.

Some cloud providers offer different types of buckets with different levels of resilience.

For example, they offer buckets which are resilient, but the data is only stored in

one data centre. This is a good option where data needs to reside in a particular geographical location or in situations where high availability is less of an issue. They will then offer buckets which are highly available across regions, where the data is stored multiple times in different datacentres (or zones) in the same region or even in multiple regions. These options usually cost more but they provide both the highest level of resilience as well as availability for your data. Object Storage has a very ‘flat’ storage structure, which we’ll explain in the next

lesson. This data can be anything from text files to audio and video files, from IOT data to

virtual machine images, from backup files to data archives. Pretty much any data which is static and where fast read and write speeds are not necessary would make a good fit for object storage. Object Storage would, however, not be suitable for running operating systems, nor applications such as databases or anything else where the contents of the files changes. So, to summarize what we have learned in this lesson: Object Storage is used to store files or Objects which are static. The data that you can store using Object Storage can be anything from text files to audio and video files, from IOT data to virtual machine images, from backup files to data archives.

You cannot run operating systems or other applications such as databases using Object

Storage. Objects are stored in Buckets. You can have multiple buckets, but you cannot place buckets within buckets. You do not need to specify a size for a bucket, you can just use as little or as much space as you need. Many providers offer different types of buckets with different charges for each. Some are based on resilience and availability, while others are based on the frequency at which the objects inside are accessed.

In the next video, we’ll be diving into Object Storage data tiers and Object Storage

APIs.

Object Storage - Tiers and APIs

In this video, we're going to look more closely at Object Storage data tiers and

Object Storage APIs. Object Storage buckets also have storage tiers, or classes, associated with them, and these tiers are based on how frequently the data is accessed.

A standard tier bucket is where you would store objects that are frequently

accessed. This tier tends to have the highest per gigabyte cost associated with it.

A vault or archive tier is where you might store documents that are only accessed perhaps only once or twice a month, or less, and this will be offered at a lower storage cost.

Whereas there may also be cold vault tier, where you would store data that is typically accessed only once or twice a year. This storage often costs just a fraction of a US cent per gigabyte per month. Often, you can also set up automatic archiving rules for

your data, meaning that if an object isn't accessed for a period of time, it will automatically be moved to a cheaper storage tier. The rule uses some of the object's metadata to determine when it should be archived. Note that, Object Storage does not come with IOPS options.

Object Storage tends to be very slow in comparison with file or block storage, where downloads typically takes seconds if not longer to complete. Where providers offer cold vault buckets, data retrieval from these tiers can sometimes even take hours because the storage is kept offline. If your application needs fast access to files, then object storage may not be a good option. We've mentioned that object storage is priced per gigabyte used, but there can also be other costs related to retrieval of the data. These costs are similarly low, but access charges can be higher for data that is in a vault or cold vault tiers, so it is important to

ensure that the data is in the correct tier based on its frequency of access. Object Storage does not need to be attached to a compute node for you to access it, rather you access object storage through an application program interface, or API. The most common API for object storage is called the S3 API, which is a standard based on the S3 object storage offered by AWS. Many providers offer APIs to their object storage, which is S3 compatible, which is useful, because it means developers can write code which is able to access multiple vendors object storage. The API itself is an HTTP based RESTful API, or RESTful web service. The API call allows applications to manage object storage and buckets, as well as put, upload, or get download objects to and from them. Object Storage is not just for new applications, but can be used to meet requirements for existing ones. It can also be used as an effective solution for backup and disaster recovery as a replacement for off-site tape based solutions,

reducing the time to restore data. Many backup packages now include the ability to backup data up into the Cloud using object storage. Object Storage is more efficient than tape backup solutions, which require tapes that need to be physically loaded into, and removed from, tape drives and moved off-site for geographical redundancy. So, to summarize what we have learned in this lesson, object storage has different tiers with different charges for each.

Some are based on the frequency at which the objects inside are accessed. Object Storage is priced per gigabyte of storage used per month plus some charges for data retrieval. Object Storage is much cheaper than file or block storage. Object Storage is very slow in comparison with file and block storage. You can often create rules which allow the automatic

archiving of objects to cheaper tiers when they are in frequently accessed. Object Storage is accessed using an API. Many Object Storage providers have an S3 compatible API which means developers can create code that will work against multiple vendor Object Storage solutions. Object Storage in the Cloud offers an effective Backup and Disaster Recovery Solution. In the next video, we will be covering Content Delivery Network,

or CDN, which is driven by Object Storage.

**CDN - Content Delivery Networks**

A content delivery network, or CDN, is a distributed server network that delivers temporarily stored, or cached, copies of website content to users, based on the user's geographic location. A CDN stores this content in distributed locations and reduces the distance between your website visitors, and your website server. In the rest of the video, we'll learn more about Content Delivery Networks. Hi. I'm Ryan Sumner, I'm a Chief Network Architect with IBM cloud, and today I'm going to help you answer: what is a content delivery network? So, in short, a content delivery network, or CDN, is a service that accelerates Internet content delivery. In other words, the main benefit of a CDN is that it makes your website faster. Before I get into describing to you how it accomplishes that, and some of the other benefits, first I want to talk to you about some of the challenges that we have where we have users all around the world, but we don't have servers all around the world, and the experience that those users have due to that dynamic. So, I've got a simple diagram here showing a server hosted down in Dallas. This is my website. And then I have users all around the world. So, in Sidney I might have five.In London I've got five. New York I might have ten. LA I might have ten. I've got 30 users around the world that are accessing my server and my website down in Dallas. Let's kind of follow a set of these users in their journey. Let's look at their users down in Sydney. They make a request to the website. They've got an 8,600 mile hike to Dallas, and then an 8,600 mile hike back. The amount of time that that takes is usually measured and measured in milliseconds, and just that round-trip might be about 170 milliseconds. For our users up in London, that might be about 100 milliseconds. Our users in New York City can probably experience about a 40 millisecond round-trip time. And over in LA, about 30. So as you can see, the further you're away, the longer it takes ultimately, the slower the website will be for you. So this is where the CDN comes into play, and this is how it actually accomplishes the increase in speed, which is by reducing the amount of distance between the user and the content, or the server providing the content. What it does by doing that is, it places these content delivery network endpoints in as many locations around the world as possible. And in our case, we're going to assume we've got one in just about every location where our users exist. So now when the user in Sydney, or London, or New York City, or LA tries to access some content, it's first retrieved by the content delivery network service and then distributed around the world. So we have a single request down to the Dallas server. It's

now then distributed all around the world, and our users in London now instead of going all the way to Dallas, they're able to retrieve that content directly from their closest geographical location, drastically reducing the amount of time that it takes to retrieve that

content. As you can see here, it's very basic how a CDN is able to provide the benefits to the end-user by reducing the amount of time that it takes to deliver the service. But what you're not seeing here, is an indirect benefit, is the reduction in the amount of traffic that actually hits the Dallas server. So the indirect benefit is that you actually see a reduction in the load, or a reduction in the amount of capacity that you need in Dallas, to serve all these users. So another indirect benefit because of there's this much less validity, and so much less stuff happening in Dallas, because all these users are having to make these trips. And

I'm also not having to communicate with with users so far away. The Dallas environment may also see an increase in uptime. And then lastly, because the users are not really directly communicating with the servers down in Dallas, you have the indirect benefit of an

increase in security through obscurity. It's pretty basic to understand how a CDN works in the end to provide a better benefit to the end user.

**Lesson Summary**

**In this lesson, you have learned:**

* Cloud storage is available in four main types–Direct Attached, File, Block, and Object Storage. These storage types differ in how they can be accessed, the capacity they offer, how much they cost, the types of data they are best suited to store, and their read-write speed.
* Direct Attached (or Local) Storage is storage that is presented directly to a cloud-based server and is effectively either within the host server chassis or within the same rack.
* File Storage is typically presented to compute nodes as a Network File System (NFS), which means that the storage is connected to compute nodes over a standard ethernet network.
* Block Storage is presented to compute nodes using high-speed fiber connections, typically provisioned in volumes, which are mounted onto a compute node.
* Object Storage is accessed via an API and doesn’t need an underlying compute node. Object Storage offers infinite capacity as you can keep adding files to it and just pay for what you use. Compared to the other storage types, object storage is slowest in terms of read and write speeds.
* A Content Delivery Network (CDN) is a distributed server network that accelerates internet content delivery by delivering temporarily stored or cached copies of website or media content to users based on their geographic location.