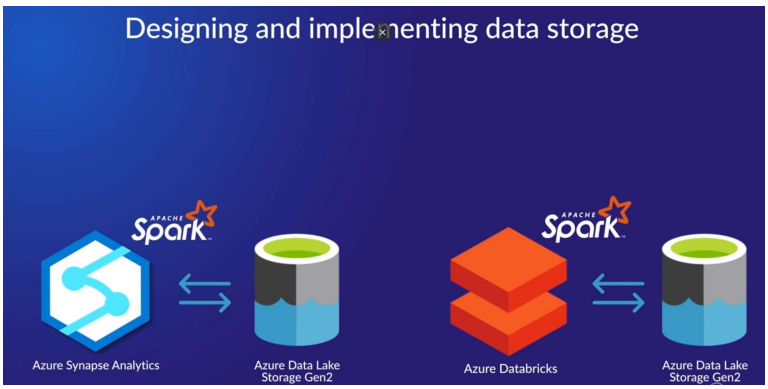
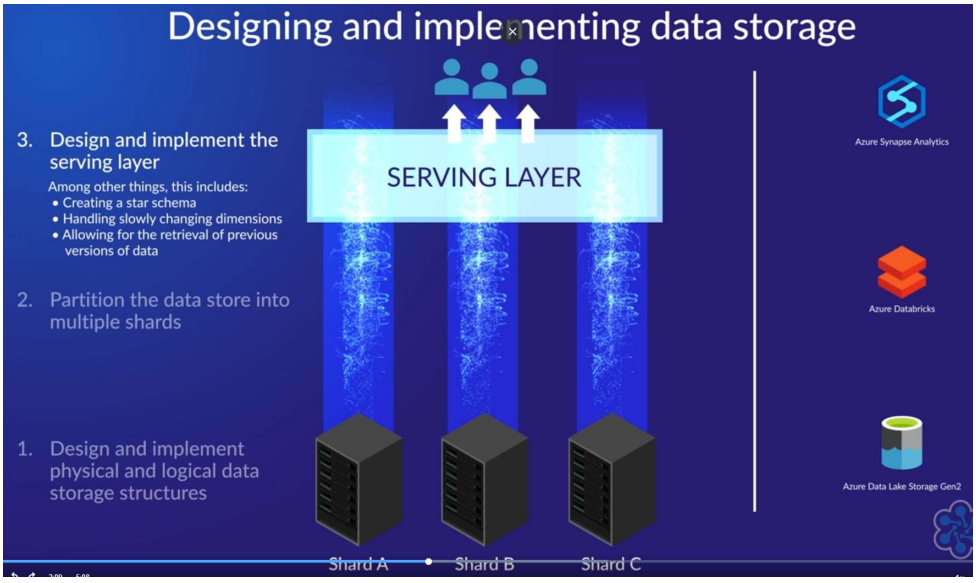
Friday, April 1, 2022 11:23 AM

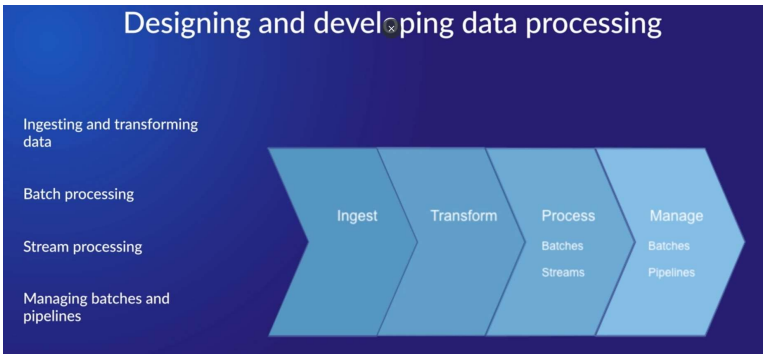
Screen clipping taken: 4/1/2022 11:28 AM



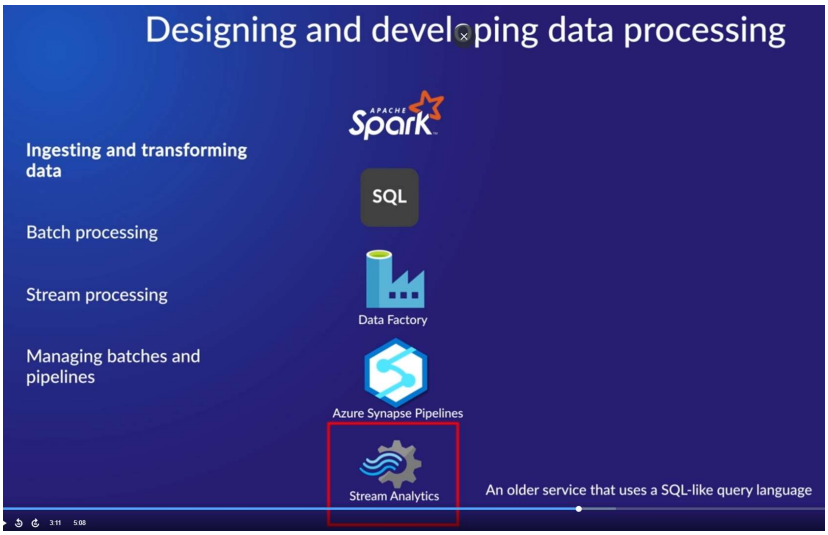
Screen clipping taken: 4/1/2022 11:33 AM



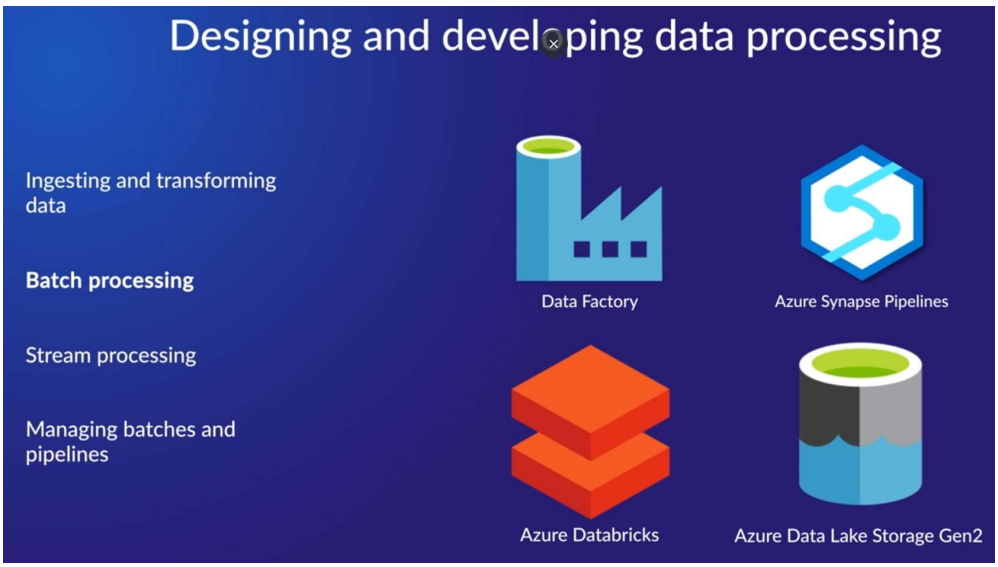
Screen clipping taken: 4/1/2022 11:34 AM



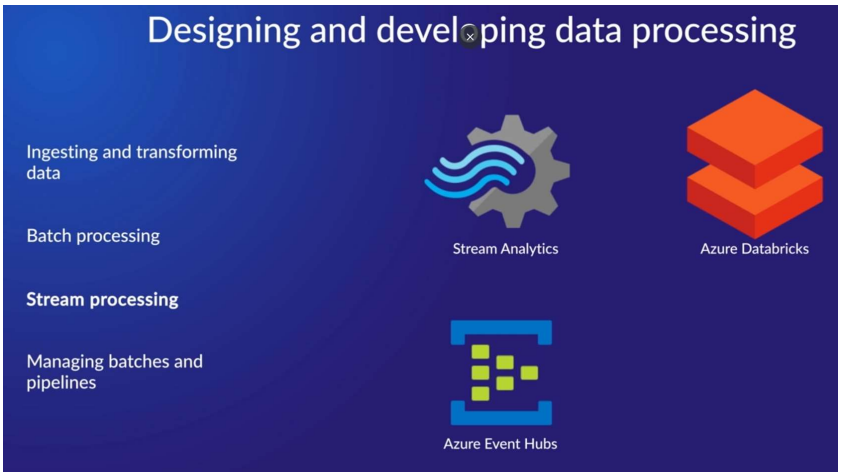
Screen clipping taken: 4/1/2022 11:35 AM



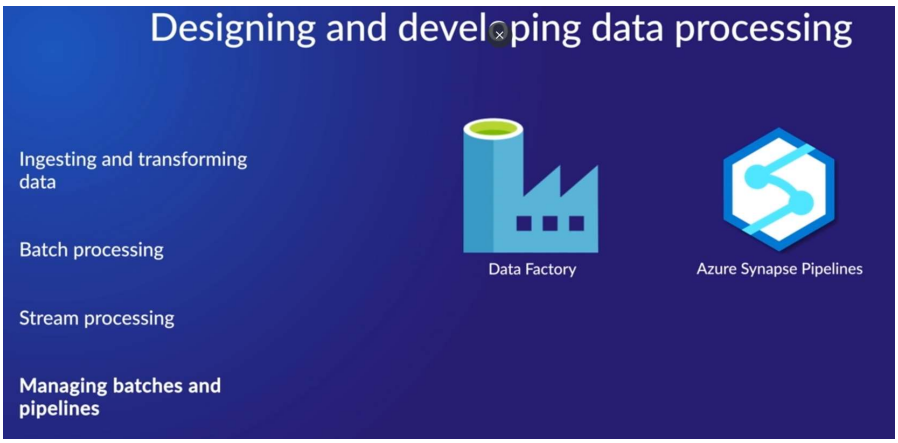
Screen clipping taken: 4/1/2022 11:37 AM



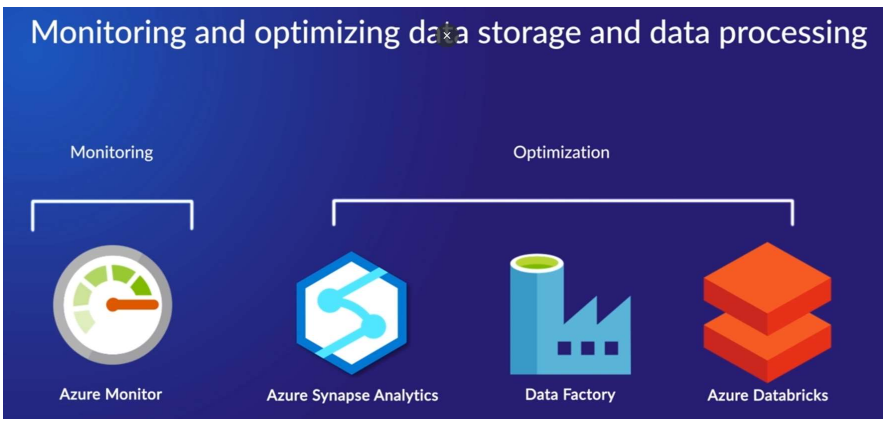
Screen clipping taken: 4/1/2022 11:42 AM



Screen clipping taken: 4/1/2022 11:43 AM



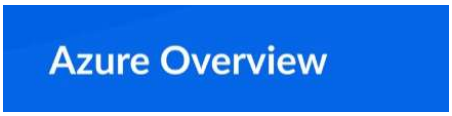
Screen clipping taken: 4/1/2022 11:43 AM



Screen clipping taken: 4/1/2022 11:45 AM

C02OAZ-Azure Overview

Friday, April 1, 2022 11:48 AM



Screen clipping taken: 4/1/2022 11:49 AM

Screen clipping taken: 4/1/2022 11:49 AM

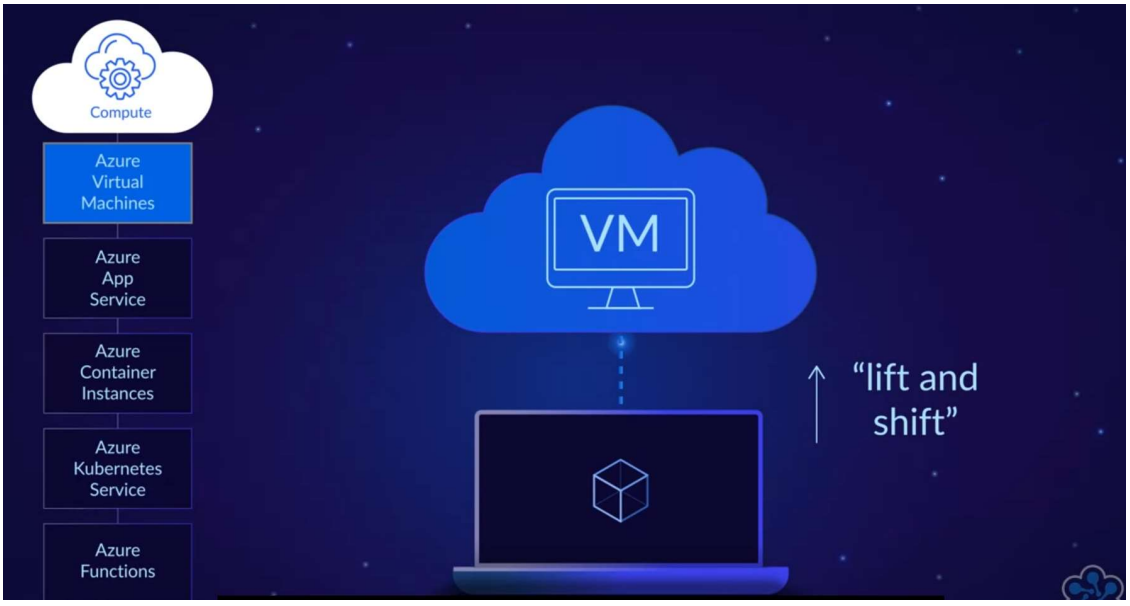
So what is Microsoft Azure? To put it simply, it's a collection of online services that organizations can use to build, host, and deliver applications.

Screen clipping taken: 4/1/2022 11:53 AM

- The best part is that you don't need to have your own data center or even any servers because Azure runs in Microsoft's data centers around the world, which your users can access over the internet.
- Not only does this approach save you the trouble of having to build and maintain your own on-premises IT infrastructure, but it can also save you money because you only have to pay for what you use, and you can scale your Azure resources up and down as needed.
- For most applications, you need three core elements: compute, storage, and networking.

Screen clipping taken: 4/1/2022 12:05 PM

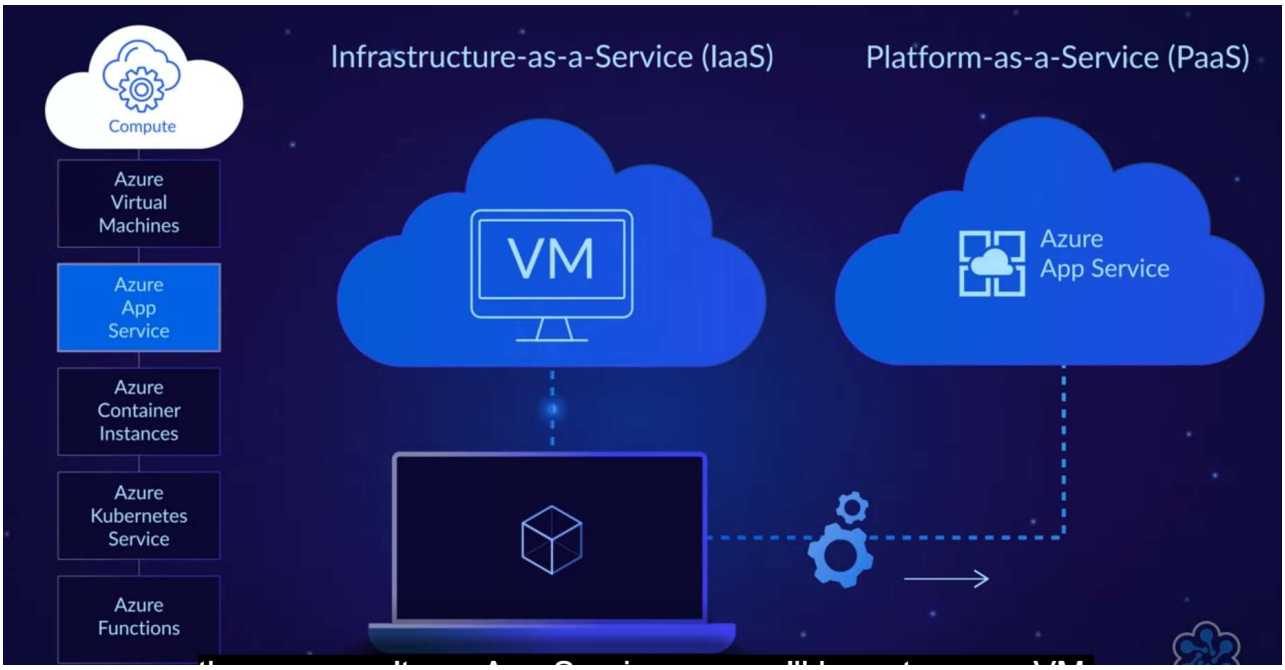
- **compute**. In Azure's early days, Microsoft offered only one type of compute service: virtual machines, or VMs for short.
- These are machines that run either Windows or Linux.
- If you currently have an application running on a Windows or Linux server, then the most straightforward way to migrate it to Azure is to do what's called a **"lift and shift" migration**.
- **"lift and shift" migration** That is, you simply lift the application from your on-premises server and shift it to a virtual server in the cloud.



Screen clipping taken: 4/1/2022 12:55 PM

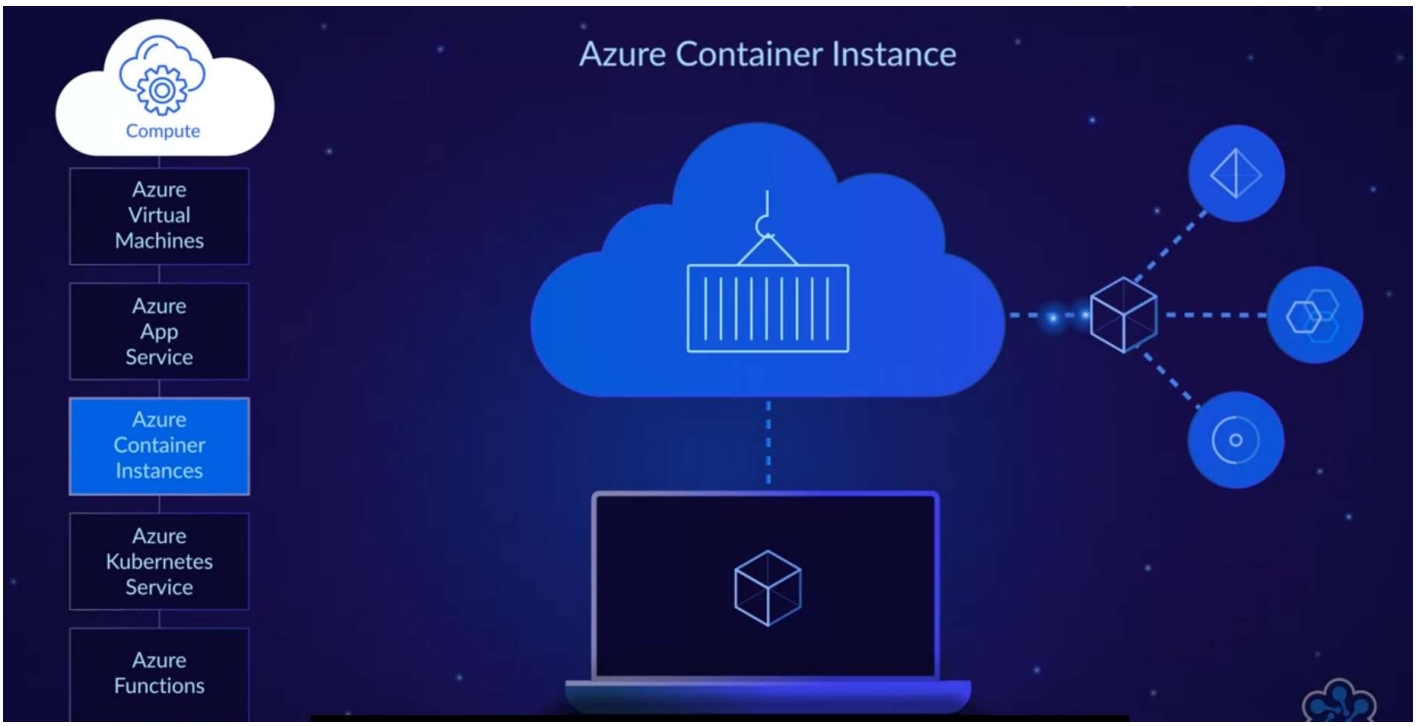
- **Azure VMs** are known as **Infrastructure-as-a-Service** because they're traditional IT infrastructure components that are offered as a service.
- Later, Microsoft came out with **what's known as a Platform-as-a-Service** offering

called **Azure App Service**. This platform lets you host web and mobile applications without having to worry about the underlying infrastructure. After doing a minor amount of configuration, you can just upload your code to an App Service instance and let Azure take care of the details. In most cases, this is a better solution than using virtual machines, but there are times when it makes more sense to use VMs. For example, if you have an application that's not a web or mobile app, then you can't use App Service, so you'll have to use a VM.



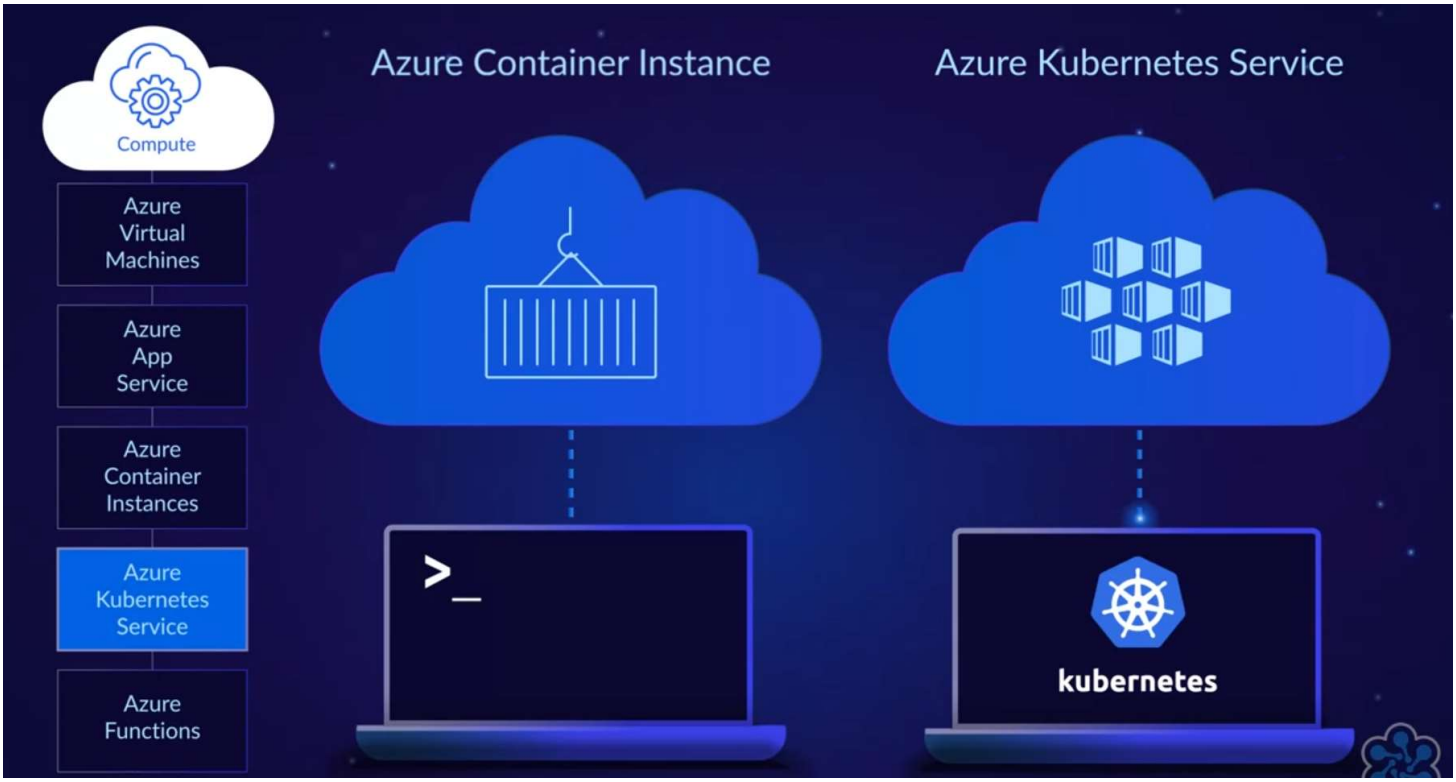
Screen clipping taken: 4/1/2022 12:56 PM

These days, the **hottest compute technology is containers**. These are self-contained software environments. For example, a **containe**r might include a complete application plus all of the third-party packages it needs.



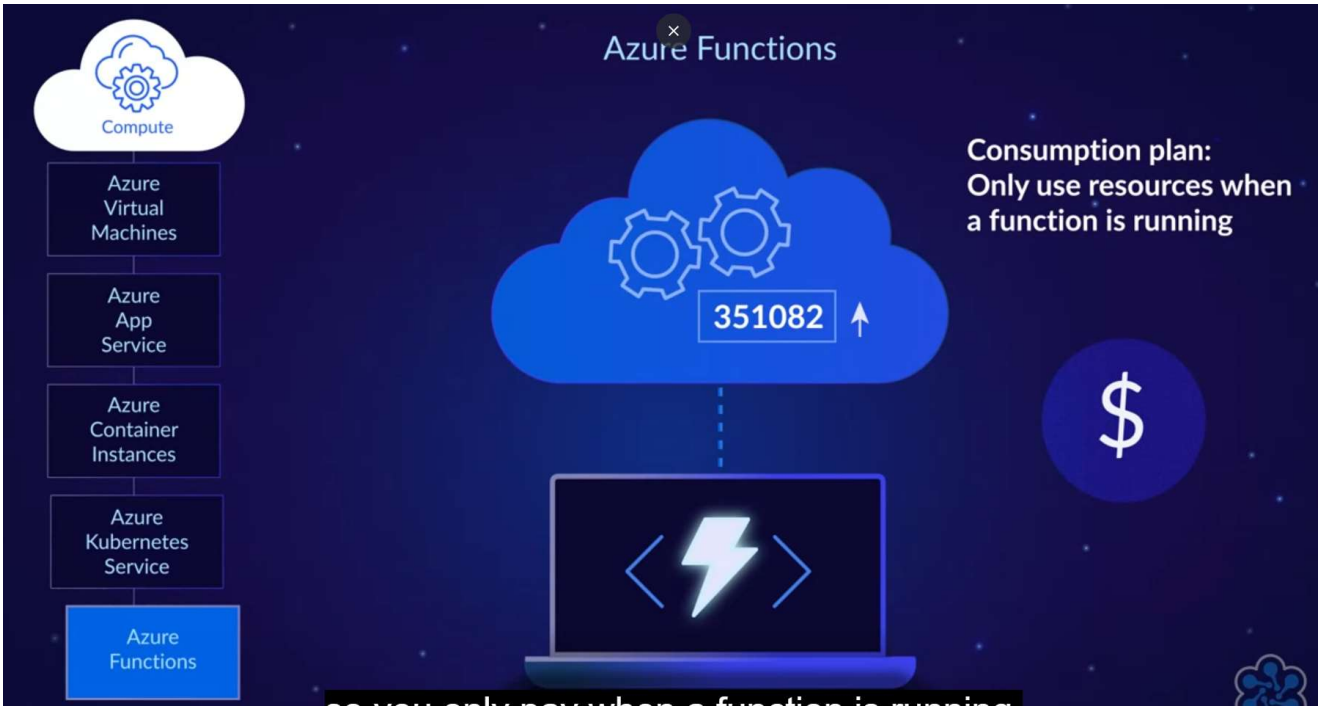
Screen clipping taken: 4/1/2022 12:58 PM

Containers are somewhat like virtual machines except they don't include the operating system. This makes it easy to deploy them because they're very lightweight compared to virtual machines. In fact, containers run on virtual machines.
Microsoft provides **a variety of ways to run containers**. The simplest way is to use **Azure Container Instances**. This service lets you run a container using a single command. If you have a more complex application that involves multiple containers, then you'll probably want to use **Azure Kubernetes Service**, which is what's known as a container orchestrator. It makes it easy to deploy and manage multi-container applications.



Screen clipping taken: 4/1/2022 12:59 PM

Before we move on, I should mention one more compute service. It's called **Azure Functions**, and it's **Microsoft's main "serverless" offering.** Azure Functions is kind of like Azure App Service except that it executes individual functions rather than entire applications, and you only pay for it when it gets used. When you provision an App Service instance, it runs until you shut it down, and you pay for it the whole time it's running. Although it's possible to configure Azure Functions in the same way, it's usually better to use the Consumption plan, which means that it only uses resources when a function is running, so you only pay when a function is running.



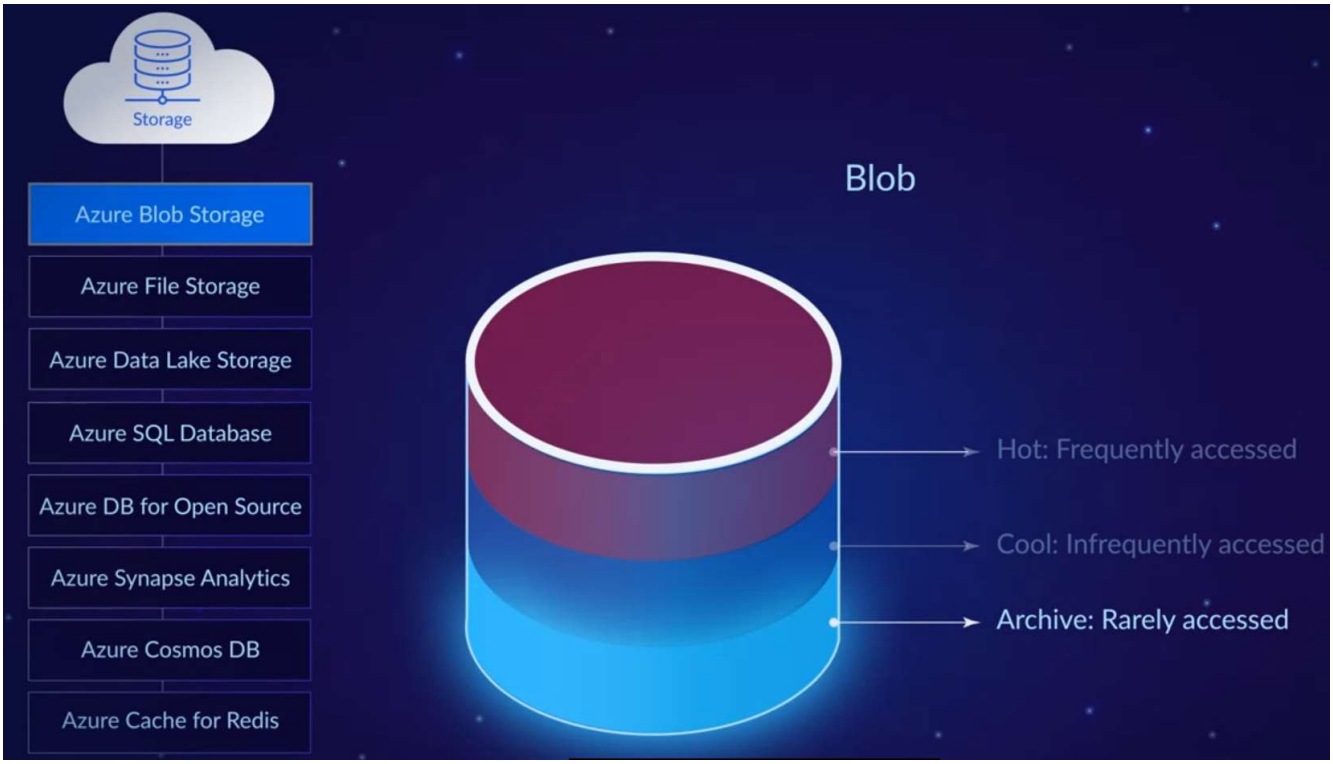
Screen clipping taken: 4/1/2022 1:01 PM

Okay, that was a lot of compute options. Now let's move on to **storage**. Believe it or not, there are even more options for storage than for compute. That's because I'm also including databases and other data stores in the storage category.
The simplest form of storage is called **Blob storage**. It's referred to as object storage, but really it's just a collection of files. It's not like a normal file system, though, because it doesn't have a hierarchical folder structure. It has a **flat structure**. It's typically used for **unstructured data**, such as images, videos, and log files.

Screen clipping taken: 4/1/2022 1:02 PM

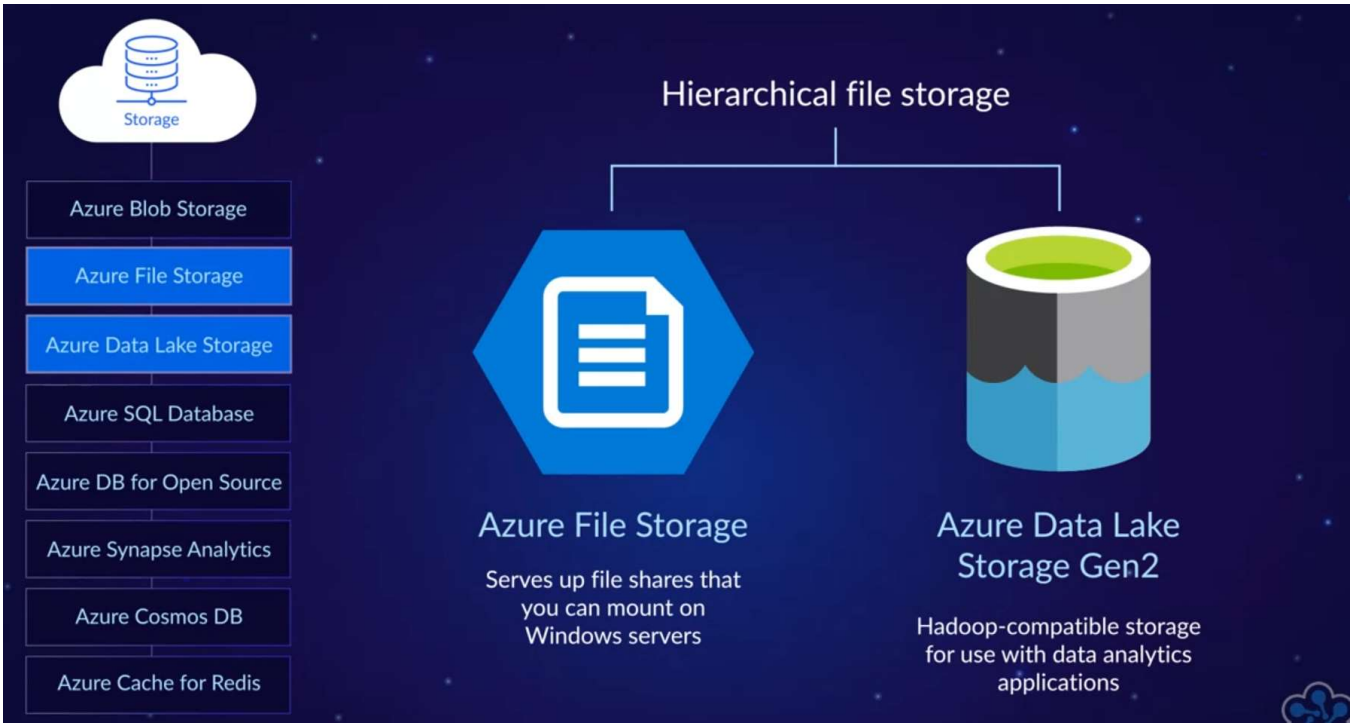
One of the great things about it is that it has multiple **access tiers**: hot, cool, and

archive. The **hot tier** is for frequently accessed files. The **cool tie**r is for files you expect to access only about once a month or less. The advantage is that it costs less than the hot tier as long as you don't access it frequently. The **archive tier** is for files that are rarely accessed, such as backup files. It has the lowest storage costs but the highest retrieval costs. It also takes several hours to retrieve files from the archive tier.



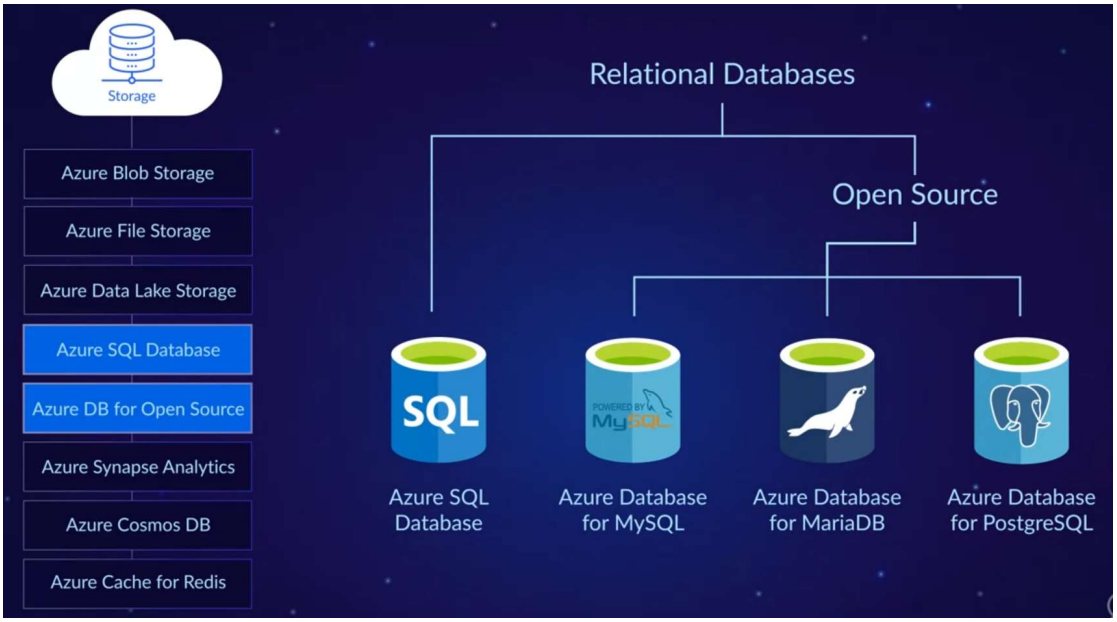
Screen clipping taken: 4/1/2022 1:04 PM

If you need **hierarchical file storage**, there are a couple of options. The one that will probably seem more familiar is **Azure File Storage**, which is like a typical **SMB** file server. It serves up file shares that you can mount on Windows servers. The less familiar option is **Azure Data Lake Storage Gen2**. This is Hadoop-compatible storage for use with data analytics applications.



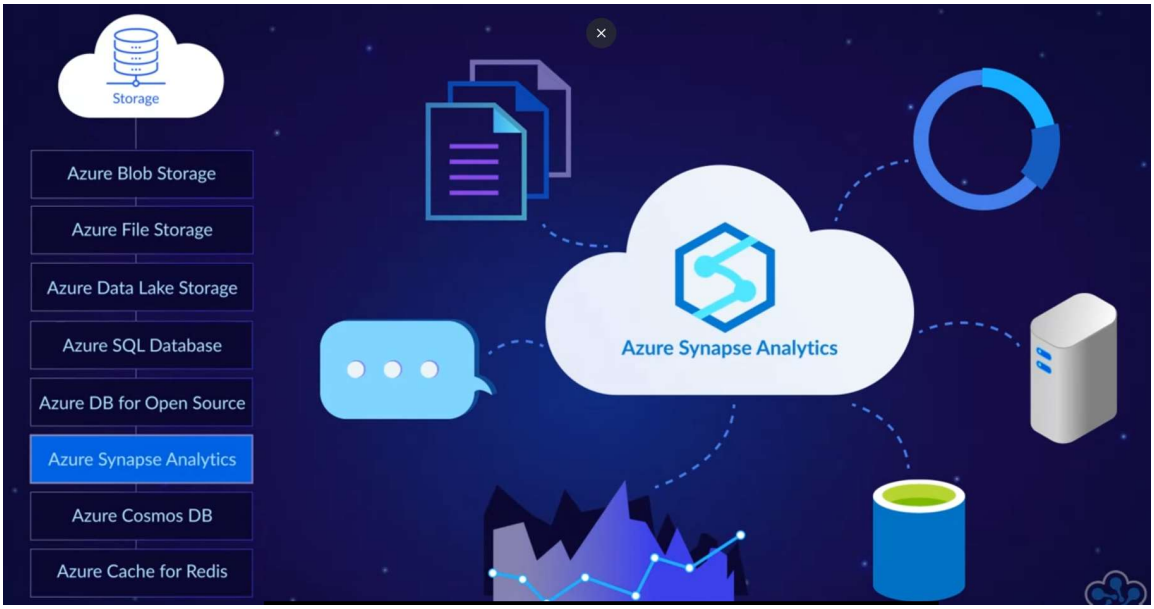
Screen clipping taken: 4/1/2022 1:06 PM

Okay, now how about **relational databases**? In an on-premises Microsoft environment, **SQL Serve**r is the most commonly used database. The cloud equivalent is **Azure SQL Database**. It's very similar to SQL Server, although it's not 100% compatible. If you need to run an **open source database**, then Microsoft still has you covered. It offers Azure Database for MySQL, MariaDB, and PostgreSQL.



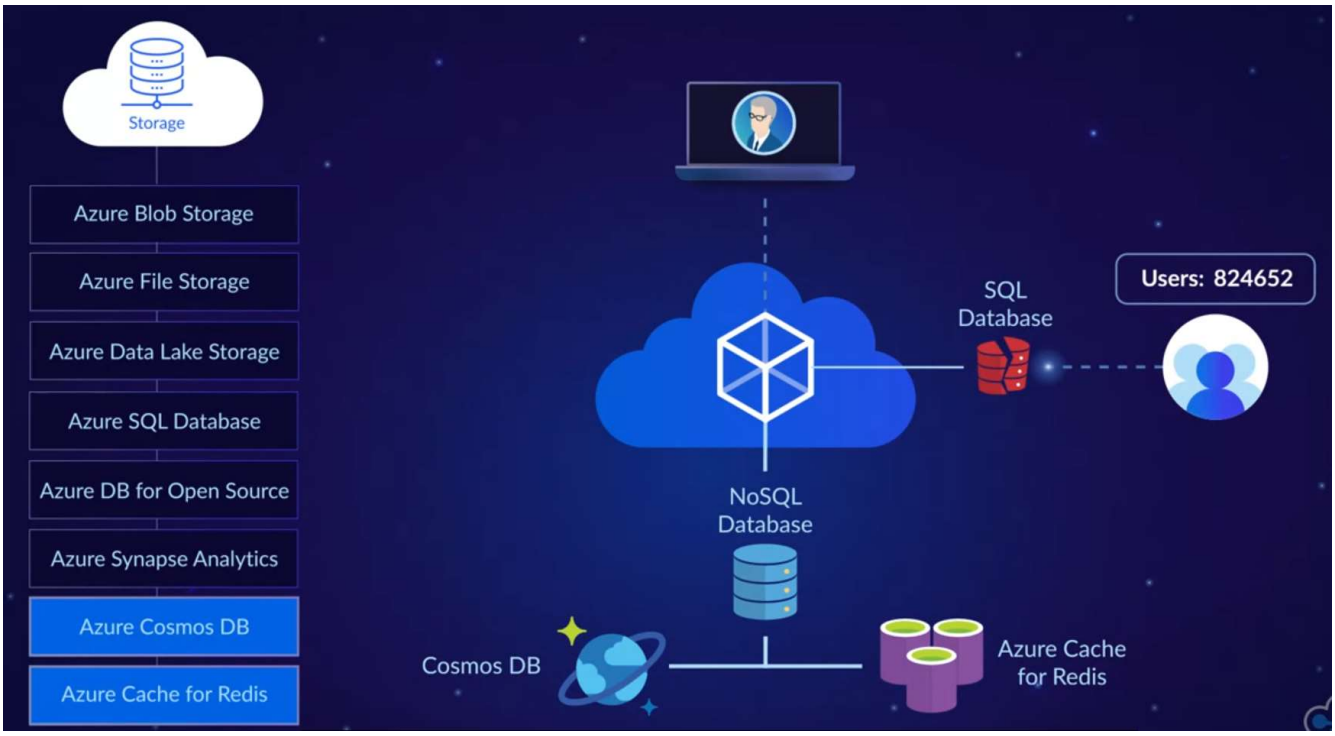
Screen clipping taken: 4/1/2022 1:07 PM

- All of these databases, including both SQL Database and the open source options, are suitable for online transaction processing. On the other hand, if you need to build a data warehouse, then **Azure Synapse Analytics** is the best choice.



Screen clipping taken: 4/1/2022 1:08 PM

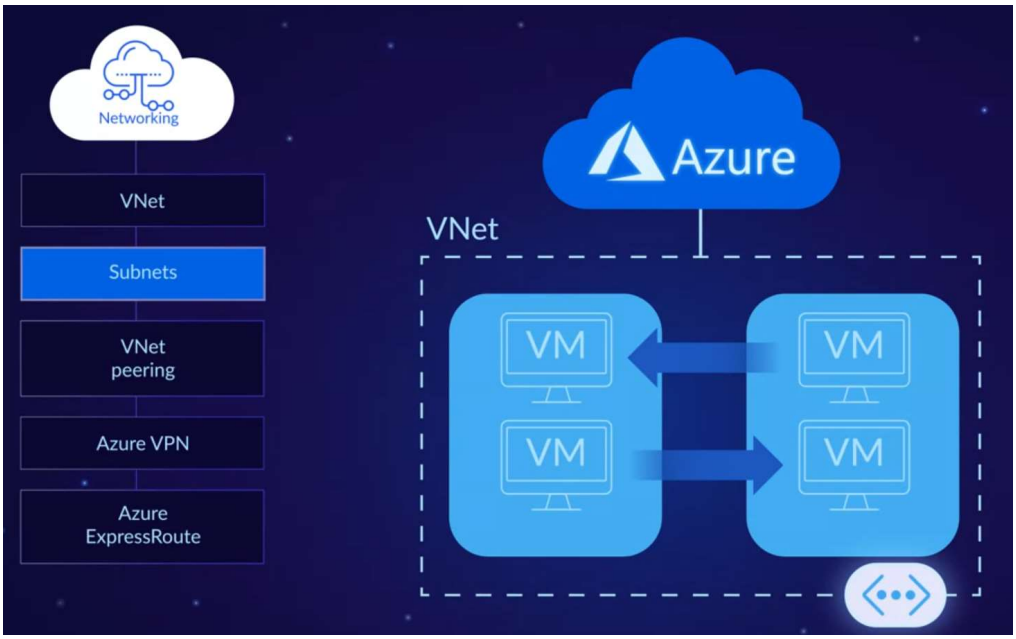
- If you release an application that attracts a very large number of users, you may find that a traditional relational database can't scale to meet the demand. One common solution is to use a so-called **NoSQL database**. These databases are designed to handle far more data than relational databases. However, in order to achieve that massive scalability, they have to sacrifice something, so they don't support all of the features of relational databases. Nonetheless, they have become a cornerstone of many cloud-based applications.
- Microsoft's main NoSQL offering is called **Cosmos DB.** It's an amazing database service that can scale globally. Another NoSQL service is **Azure Cache for Redi**s, which is typically used to speed up applications by caching frequently requested data.



Screen clipping taken: 4/1/2022 1:11 PM

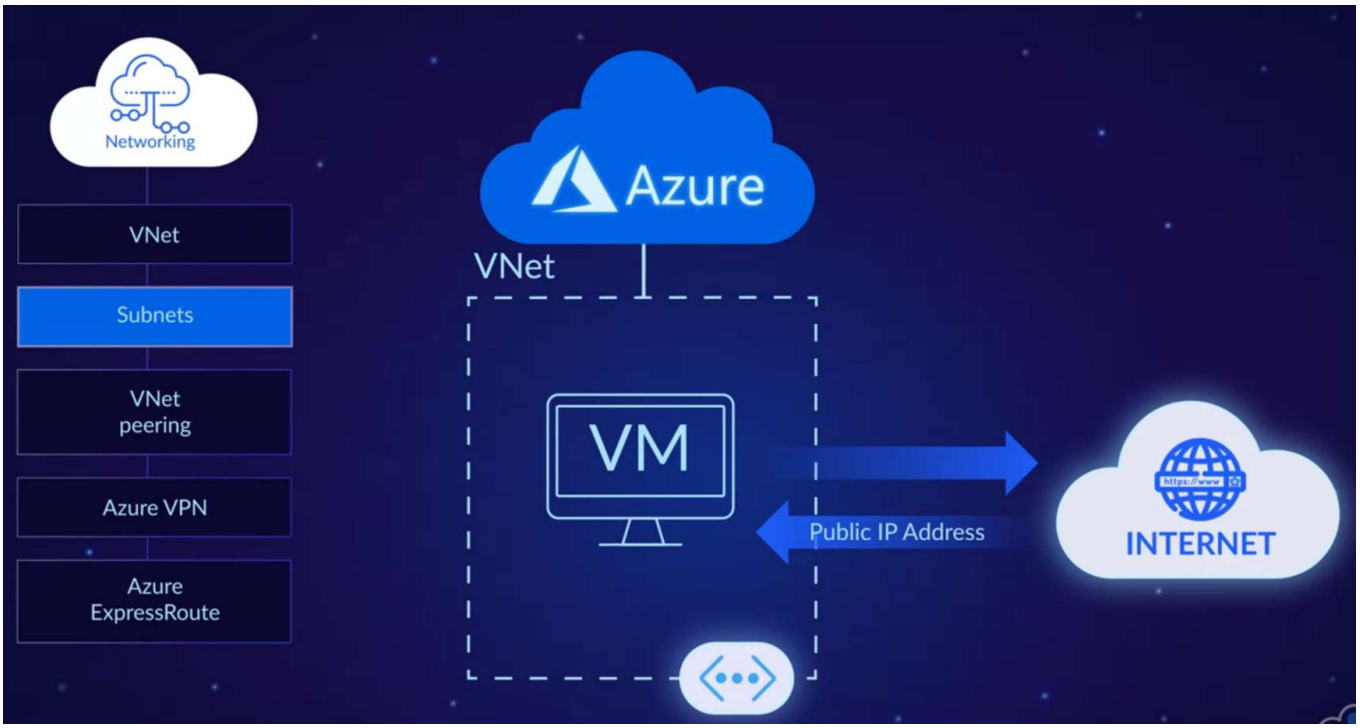
Networking:

- When you create a virtual machine on Azure, you have to put it in a virtual network, or **VNet**. A **virtual network** is very similar to an on-premises network. Each virtual machine in a VNet gets an IP address, and it can communicate with other VMs in the same VNet. You can also divide a VNet into subnets and define **routes** to specify how traffic should flow between them.



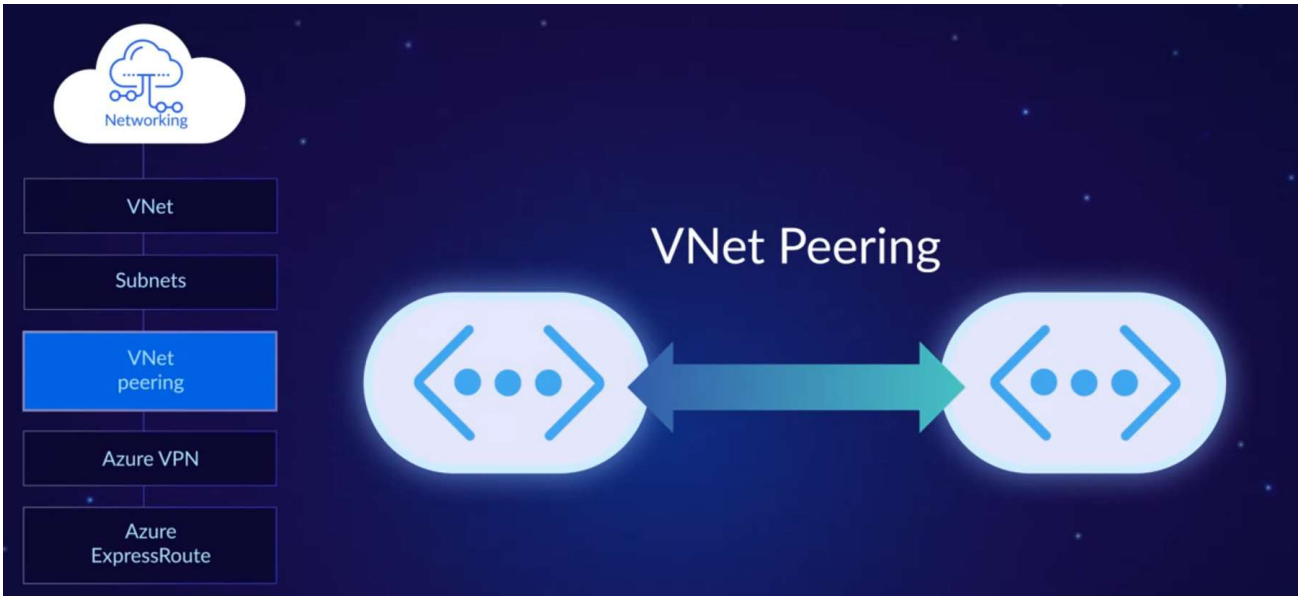
Screen clipping taken: 4/1/2022 1:13 PM

By default, all **outbound** traffic from a VM to the internet is allowed. If you also want to allow **inbound** traffic, then you need to assign a public IP address to the VM.



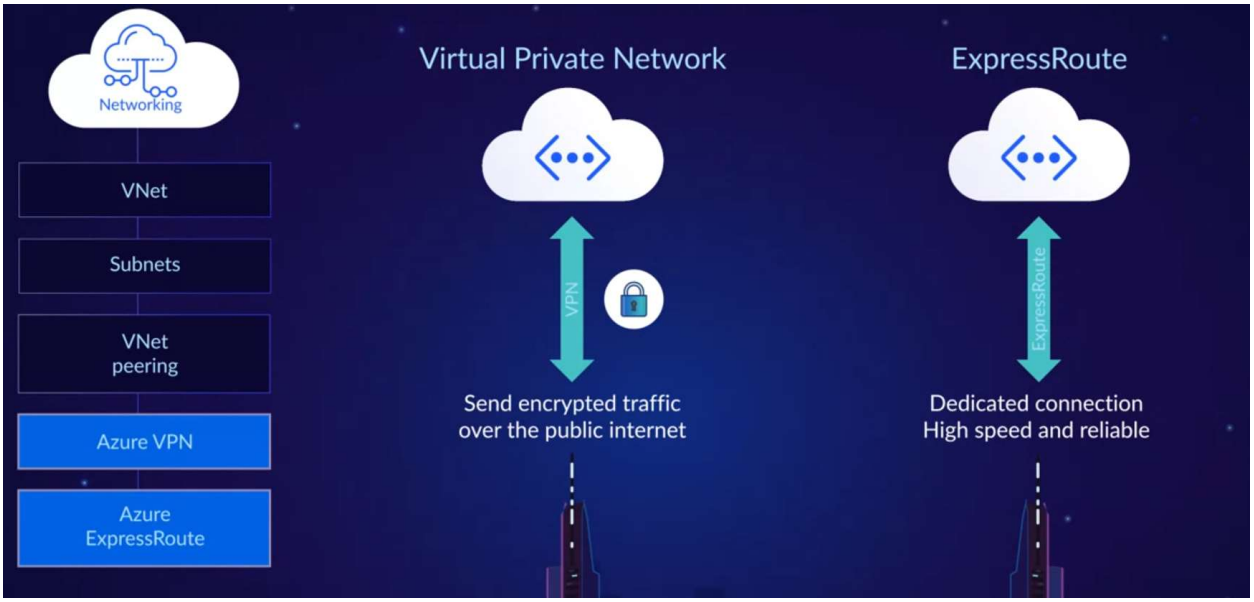
Screen clipping taken: 4/1/2022 1:14 PM

If you want VMs in one VNet to be able to communicate with VMs in another VNet, then you can connect the VNets together using **VNet peering**. By the way, so far, I've only been talking about virtual machines, but other resources, such as Kubernetes clusters, can be in VNets, too.



Screen clipping taken: 4/1/2022 1:14 PM

If you want to create a secure connection between a VNet and an on-premises network, then you can use either a VPN, which stands for **Virtual Private Networ**k, or **Azure ExpressRoute**. A **VPN** sends encrypted traffic over the public internet, whereas **ExpressRoute** communicates over a private, dedicated connection between your site and Microsoft's Azure network. ExpressRoute is much more expensive than a VPN, but it provides higher speed and reliability since it's a dedicated connection.



Screen clipping taken: 4/1/2022 1:16 PM

There are plenty of other network services available as well, but the ones I've covered are enough to give you a good high-level understanding of Azure networking.

Azure also provides a wide variety of other services outside of the core compute, storage, and networking categories, such as in **hot areas like artificial intelligence and DevOps**. I'll go over some of those later.
The easiest way to understand how Azure works is to actually use it, so in the next lesson, we'll create a virtual machine. If you're ready, then go to the next video.

From <https://cloudacademy.com/course/overview-of-azure-services/azure-overview/?context_id=3191&context_resource=lp>

Using the Azure Portal

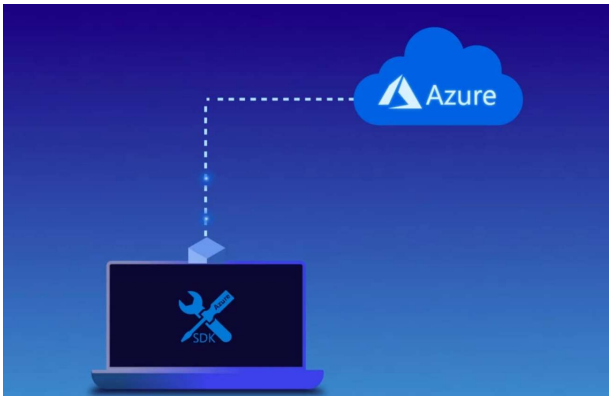
Friday, April 15, 2022 2:55 PM

Now we're going to dive into using Azure. Suppose you have a server application that you want to migrate to the cloud. As I mentioned earlier, the most straightforward way to do this is to move the application to a virtual machine on Azure.
There are many ways to interact with Azure, the Azure portal runs in
-a browser, so you don't need to install anything to use it.
-Alternatively, you can install the CLI, which stands for command-line interface,
-or Azure PowerShell or the SDK, which stands for Software Development Kit.

If you want to use a command-based interface, and you have a lot of experience with PowerShell, then that's probably your best bet.
-Otherwise, use the CLI, which is easier to learn than PowerShell.
-It's actually possible to use the CLI and PowerShell from inside the browser too without having to install anything on your desktop. I'll show you how to do that in the next lesson.

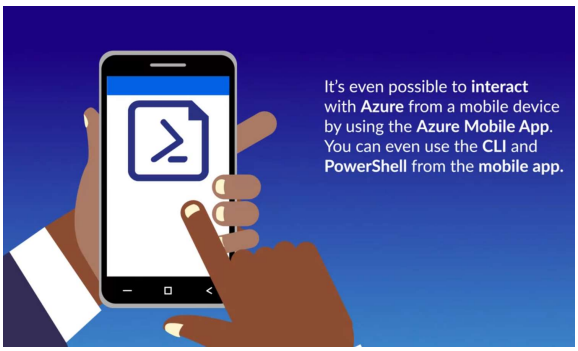
Screen clipping taken: 4/15/2022 2:57 PM

The SDK is what you need to use if you're going to add code in your applications to interact with Azure.



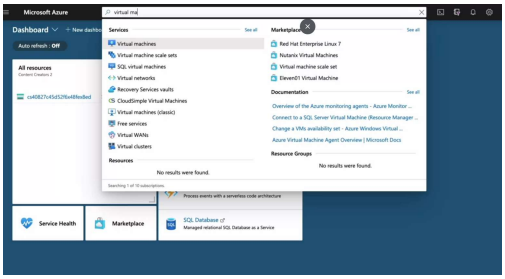
Screen clipping taken: 4/15/2022 2:58 PM

It's even possible to interact with Azure from a mobile device by using the Azure mobile app. You can even use the CLI and PowerShell from the mobile app.



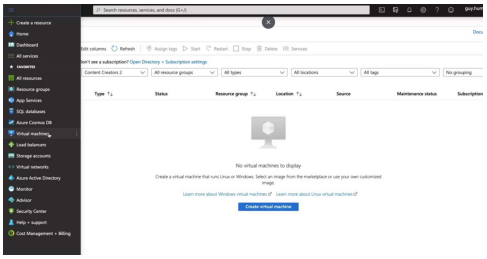
Screen clipping taken: 4/15/2022 2:58 PM

- The easiest way to get started, though, is to use the Azure portal. So if you're following along on your own account, go to portal.azure.com.
- There are a few different ways to get to the place where you can create a VM. In this search box, you can start typing virtual machine and you'll see it come up at the top of the search results.



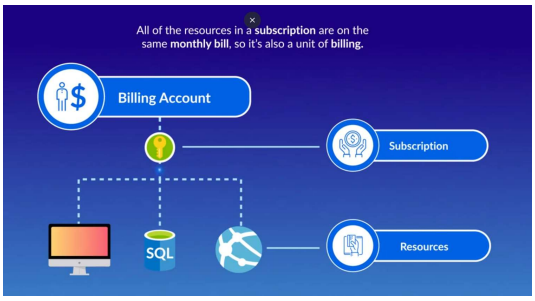
Screen clipping taken: 4/15/2022 2:59 PM

- Another way is to go into the menu on the left and select virtual machines. It takes you to the same place. Once you're here, you can click Add.



Screen clipping taken: 4/15/2022 3:00 PM

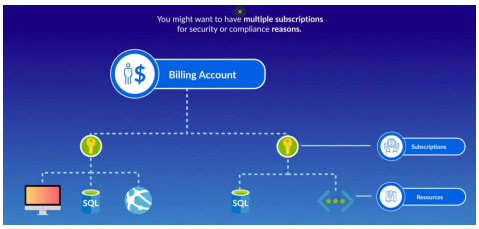
- The first thing it wants to know is what subscription to put this VM in. It always sets this to your default subscription, so you can usually just leave it that way. But I should explain what a subscription is.
- When you sign up for an Azure account, Microsoft creates both a billing account and a subscription.
- It's easy to get the two mixed up because they're both involved in billing. A *billing account* is an agreement you or your organization sign to use Microsoft services. A *subscription* is actually just a collection of Azure resources. But all of the resources in a subscription are on the same monthly bill, so it's also a unit of billing.



Screen clipping taken: 4/15/2022 3:01 PM

So why do you need to have both a billing account and a subscription?

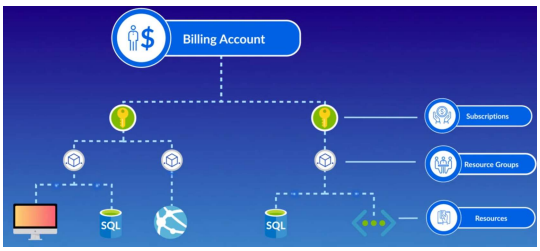
- Well, you might want to have multiple subscriptions in your billing account. Since each subscription generates a separate invoice, it can be useful to have a separate subscription for each department in your organization. Also, since the resources in different subscriptions are isolated from each other, you might want to have multiple subscriptions for security or compliance reasons.



Screen clipping taken: 4/15/2022 3:02 PM

Okay, so we'll leave the subscription with the default. Next, it's asking for the resource group.

- Like a subscription, a resource group is a collection of resources, but a subscription can have multiple resource groups so it's a way of further grouping the resources within a subscription.

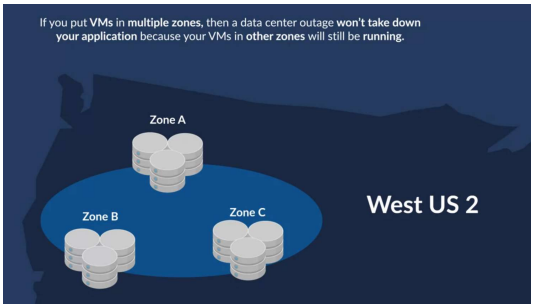


Screen clipping taken: 4/15/2022 3:03 PM

- There are a variety of ways to divide resources into resource groups. But the best practice is to group related resources together, such as a VM and its associated storage account. Generally speaking, the resources in a group should be created and deleted at the same time, which makes sense if they're components that work together to provide a solution.
- All right, so let's create a new resource group to hold this VM. I'll call it example-RG. Now we need to give the VM a name. I'll call it example-VM. Next we need to decide which region the VM should run in.
- Microsoft has data centers on every continent except Antarctica. Each of these dots is a region, and each region contains multiple Azure data centers. You usually want to choose a region that's closest to where your users are located.

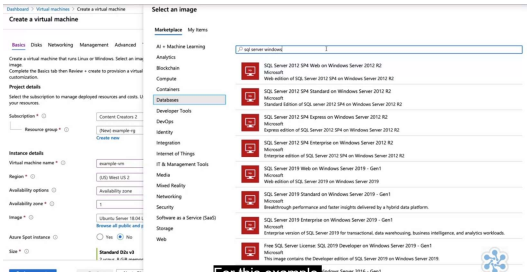
Screen clipping taken: 4/15/2022 3:05 PM

- I'll choose West US 2. Notice that it has a diamond in it. That's because it has availability zones. I'll explain what those are in a minute.
- You need to set an availability option if you want to make sure your application will still be available, even if Azure experiences an outage that affects this VM. The first option is **"Availability zone"**, which is what I just showed you on the map.
- A region that supports availability zones has at least three data centers, each of which is called a zone. If you put VMs in multiple zones, then a data center outage won't take down your application, because your VMs and other zones will still be running.



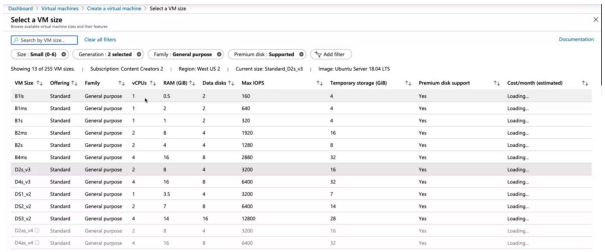
Screen clipping taken: 4/15/2022 3:08 PM

- It added another field called "Availability zone". We can choose 1, 2, or 3. It doesn't really matter which one we choose for this VM, but if we were to create a second VM, we'd need to put it in a different availability zone. I'll choose "1".
- Next, we need to choose a **VM image**. This is a copy of the disk that will be used for the VM. The choices are different versions of Linux and Windows, such as Ubuntu, Red Hat, and Windows Server. This is only a small subset of the images that are available. You can click on this link to see more. For example, suppose you want an image that includes not only Windows Server but also SQL Server, click on Databases and then type "sql server". Surprisingly, there are images of SQL Server on Linux first. I'll add windows to the search to narrow it down. There we go. There are lots of SQL Server image options. For this example, we'll just go back and pick one of the basic options though. I'll pick the Ubuntu server option.



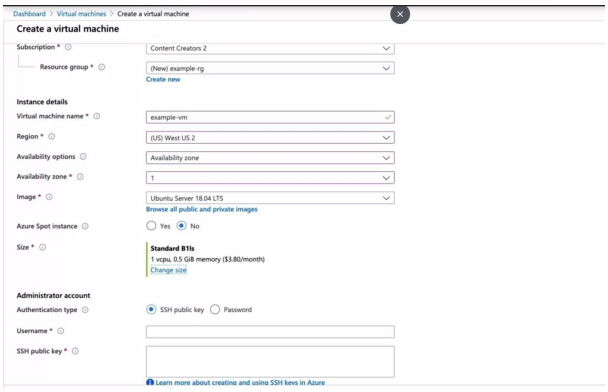
Screen clipping taken: 4/15/2022 3:10 PM

- If you use an Azure spot instance for this VM, then the cost will be dramatically lower, but the VM might be shut down with only 30 seconds' notice. So you should only use this option for non-critical workloads.
- There are lots of **options for the size of the VM**.
- At the top of the list, there's a size with only one virtual CPU, half a gig of memory, and four gigs of temporary storage. You can see that the cost per month is very low. Some of these other options are much bigger, but so is the cost. Since we're not actually going to use this VM, I'll choose the cheapest option.



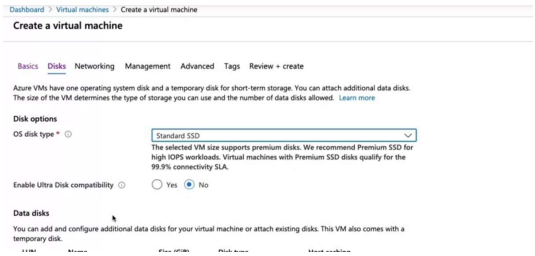
Screen clipping taken: 4/15/2022 3:11 PM

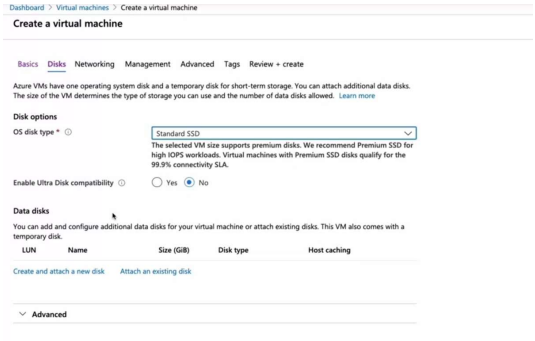
- Here, we need to create the **administrator account**. In most cases, you should use an **SSH public key** for authenticating to the VM. But to keep **it simple for this demo, I'll use a password.** You can name your administrator account almost anything.



Screen clipping taken: 4/15/2022 3:12 PM

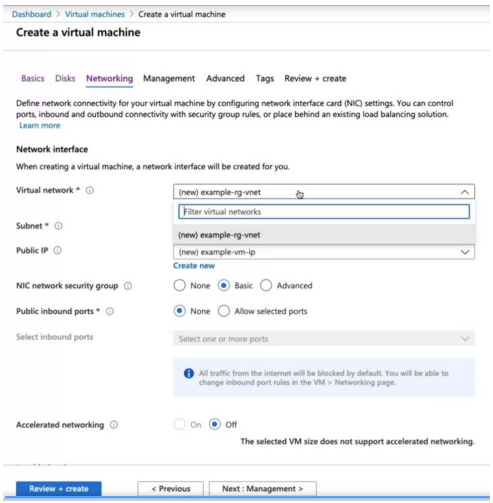
- The inbound port rules let you open up the VM to access from the internet. If this VM will be acting as a web server, then you'll wa**nt to open up ports 80 and 443.** There's also an option to open up the SSH port so you can log into the VM remotely.
- The problem with opening up the SSH port here is that it will allow access from all internet addresses, which is dangerous. There are other more secure ways to give yourself access to the VM. So you shouldn't select this unless you're just doing some testing. I'm going to set this to "None".
- That's it for the basic settings, but there are a few more settings we should have a look at. We'll click this button to go to **the disks tab**. The operating system disk is set to premium SSD by default. That option has the highest performance, but it's much more expensive than the other two. I'll set it to Standard SSD. You can also add data disks if you want.





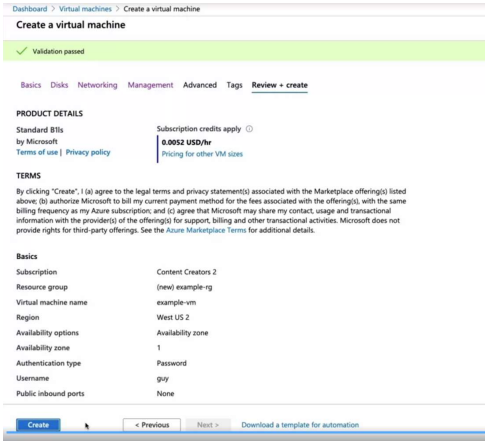
Screen clipping taken: 4/15/2022 3:17 PM

- And now let's have a look at the **networking options.** By default, it creates a virtual network, a subnet inside that virtual network, and a public IP address. If we already had another virtual network, we could put it in there instead if we wanted to.



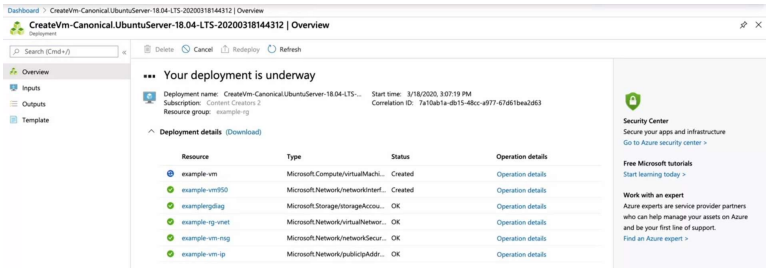
Screen clipping taken: 4/15/2022 3:18 PM

- Okay, that takes care of the compute, storage, and networking options for this VM. So let's create it.
- First, click "Review + Create". It says that our settings have passed validation testing. Then it summarizes all of our settings. Now click the Create button.



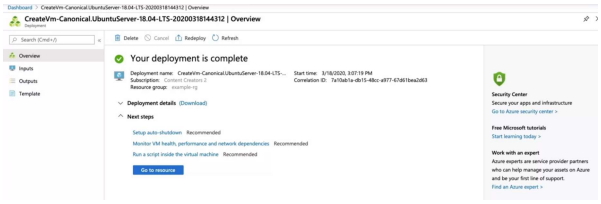
Screen clipping taken: 4/15/2022 3:19 PM

- It'll take a little while to create, so I'll fast forward.



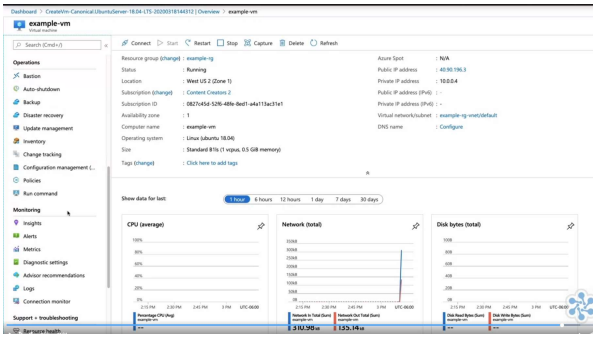
Screen clipping taken: 4/15/2022 3:20 PM

- All right, it's done. We can click on "Go to resource" to have a look at the VM.



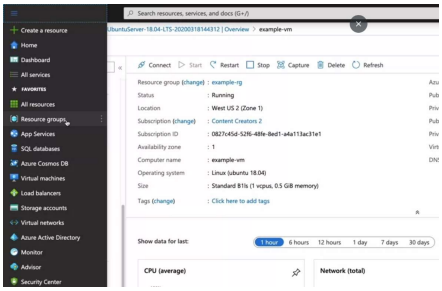
Screen clipping taken: 4/15/2022 3:20 PM

- Here it shows the details of this VM, such as its status, and its public IP address. Up here it has some controls that let you stop and restart the VM or even delete it. If we had set up SSH authentication, then we could connect to it from here too. In the menu on the left, there are all kinds of options for things like activity logs, security, backups, and monitoring.

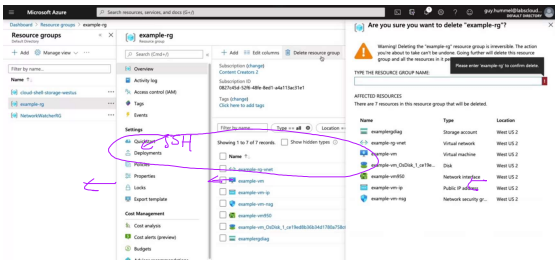


Screen clipping taken: 4/15/2022 3:22 PM

- That's it for this demo, you should have a pretty good idea of how to create resources in the portal now, because the process is fairly similar, regardless of what kind of Azure resources you're creating. If you're following along on your own Azure account, you might want to look around and see what you can do with your new VM.
- **When you're done, make sure you delete the VM so you don't incur any ongoing charges.** The best way to delete everything associated with the VM, including the public IP address, etc., is to **delete its resource group**.
- Select Resource groups in this menu, and then click on "example-rg". Here's a list of all the resources associated with the VM. Now click "Delete resource group". Then type the name of the resource group. Then click the Delete button.



Screen clipping taken: 4/15/2022 3:24 PM



Screen clipping taken: 4/15/2022 3:25 PM

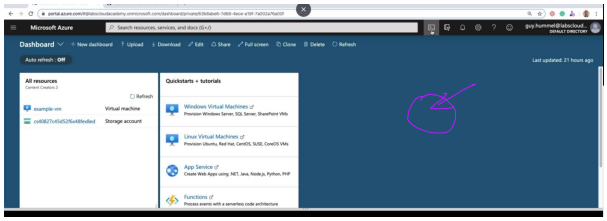
Now if you're ready to see how to work with Azure using the *command-line interface*, then please go to the next video.

From <https://cloudacademy.com/course/overview-of-azure-services/using-azure-portal/>

Using the Azure CLI

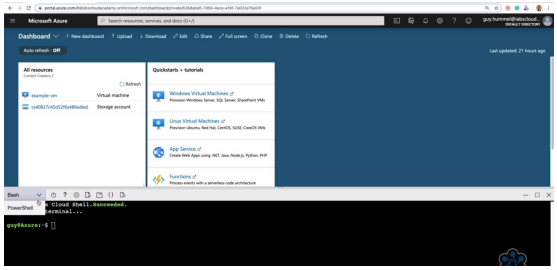
Friday, April 15, 2022 3:26 PM

- As you saw in the last demo, it's pretty easy to use the portal to create Azure resources but it's definitely not the most efficient way to do it because it requires a lot of pointing and clicking.
- An alternative is to use the command line. Although it can be more difficult since you have to know the exact names of all the command line options. Once you've mastered it, you can often create and manage resources much more quickly.
- In this demo, I'm going to show you how to create a website using Azure App Service. Do you remember when I said that it's possible to use the command line without having to install anything on your desktop? Well, now that you're familiar with the portal, I'll show you how.
- You see this icon to the right of the search bar? That's how you start something called the Cloud Shell. It's a very small virtual machine that you can use to run Azure commands. Click on it.



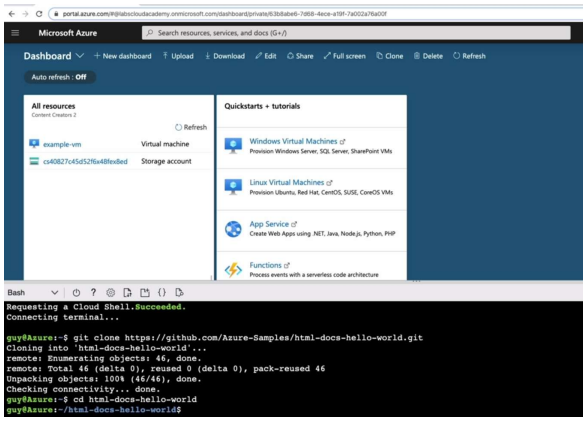
Screen clipping taken: 4/15/2022 4:42 PM

- The first time you run Cloud Shell, it'll ask you for permission to create a storage account that the Cloud Shell VM can use. This isn't the first time I've used Cloud Shell, so my storage account is already set up. If you do get a dialogue box asking for permission, then click the option to create a storage account.
- Cloud Shell supports both PowerShell and the Bash shell. You can switch between them using this menu. We're going to use the Bash Shell because the commands are simpler. If you're familiar with Linux commands, then it will be especially easy.

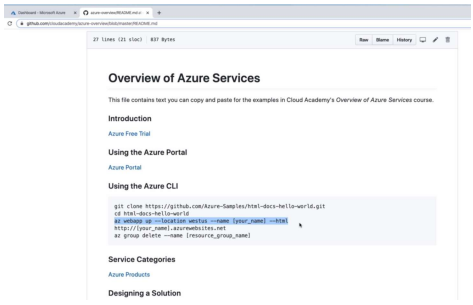


Screen clipping taken: 4/15/2022 4:44 PM

- First, we need to download the files for a sample website. The command to do that is pretty long, so it's easiest to copy and paste it.
- You can get it from the read me file in the GitHub repository for this course.
- You can find a link to the repository at the bottom of the overview tab below this video.
- Okay, copy this git clone command and paste it into the Cloud Shell window. It created a directory (which is the Linux term for folder) called html-docs-hello-world.
- Change into that directory with the cd command. I'm just going to use keyboard shortcuts to do the copying and pasting now.
- Now we can create an Azure App Service Web App using one command. Here it is.

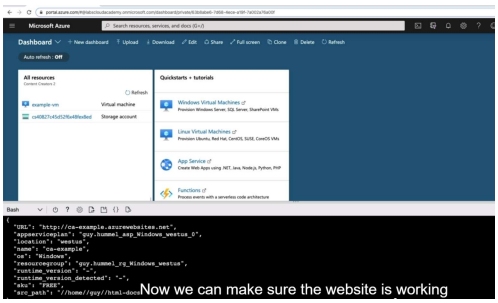


Screen clipping taken: 4/15/2022 4:46 PM

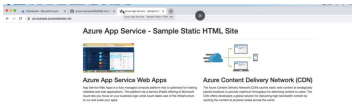


Screen clipping taken: 4/15/2022 4:46 PM

- It starts with **az which means it's an Azure command**. Then we say **webapp w**hich is the command for Azure App Service.
- Then **up** which means to create the web app using the code in the current directory.
- Then we say **--location** and the name of the region where we want to deploy the web app.
- I've set it to westus. Then we say **--name** and what we want to call the app.
- I'm calling it ca-example, but you'll have to call it something else because the name has to be unique among all of the out service names across all Azure customers.
- Then we say **--html** to tell it that we're creating a static HTML website. Now hit Enter.
- That's all you have to do to create a website running on Azure App Service. It'll take a little while to deploy so I'll fast forward.
- Okay, it's done. Now we can make sure the website is working by going to this URL. Remember to change this name to the one you used. Great, there's the sample website.

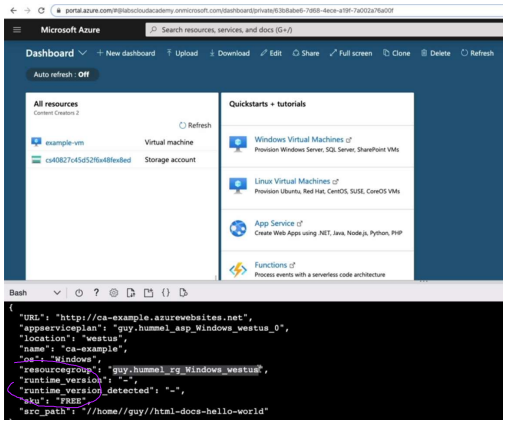


Screen clipping taken: 4/15/2022 4:49 PM



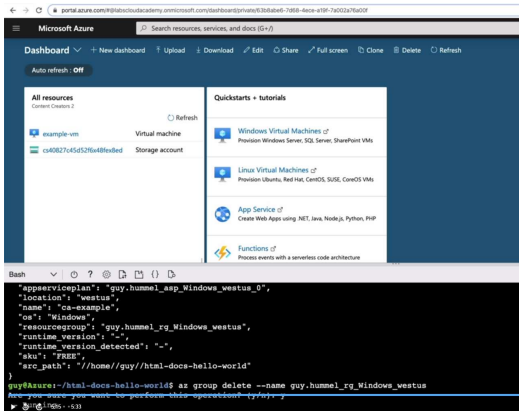
Screen clipping taken: 4/15/2022 4:49 PM

- All right, now let's delete the web app, so we don't incur any more charges. In the VM demo, we deleted everything associated with the VM by deleting its resource group.
- We can do the same thing here but since we didn't specify a resource group name in the command, how do we do that? App Service created a resource group for us.
- You'll see it in the output from the command.



Screen clipping taken: 4/15/2022 4:50 PM

- Type az group delete --name then copy and paste the resource group name from here. Type "y".



Screen clipping taken: 4/15/2022 4:52 PM

- While that's getting deleted, I should mention that there are many ways to deploy a web app.
- For example, both Visual Studio and Visual Studio Code can publish an app directly to Azure App Service without you needing to use the Azure portal or the command line.
- Once the app is deployed, though, you'll need to use the portal or the CLI to manage it.

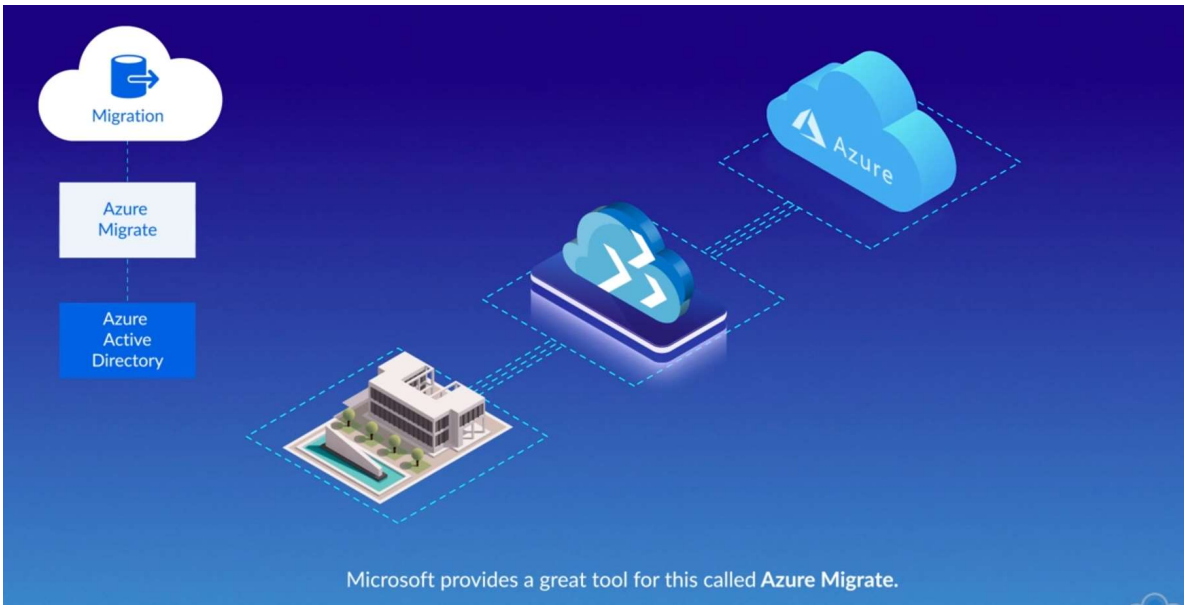
And that's it for using the CLI.

From <https://cloudacademy.com/course/overview-of-azure-services/using-azure-cli/>

Service Categories

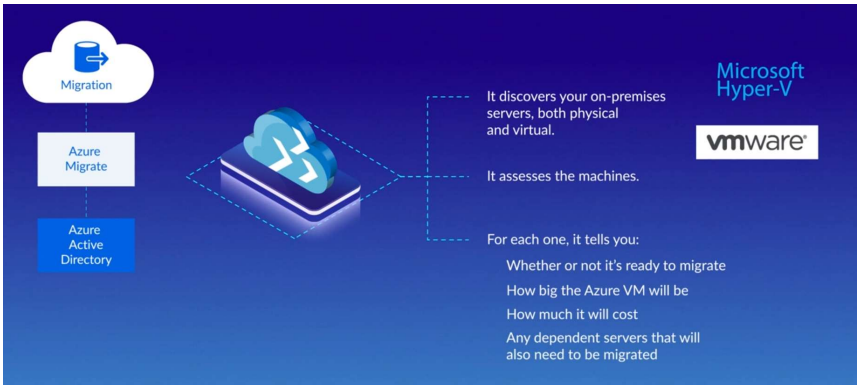
Friday, April 15, 2022 5:00 PM

So far, I've only talked about compute, storage, and networking services, but Microsoft offers services in many other categories as well. I won't talk about all of the Azure services available, but I'll go over some of the highlights.
If your organization is just getting started with Azure, then one of the first things you'll want to do is figure out how you can migrate at least some of your existing applications to Azure. Microsoft provides a great tool for this called Azure Migrate.



Screen clipping taken: 4/15/2022 5:03 PM

First, it discovers your on-premises servers, both **physical and virtua**l. On the **virtual side**, this includes both **Hyper-V and VMware**. Then it assesses these machines. For each one, it tells you whether or not it's ready to migrate, how big the Azure VM will be, how much it will cost, and any dependent servers that will also need to be migrated. When you're ready, it will even help you do the migration.

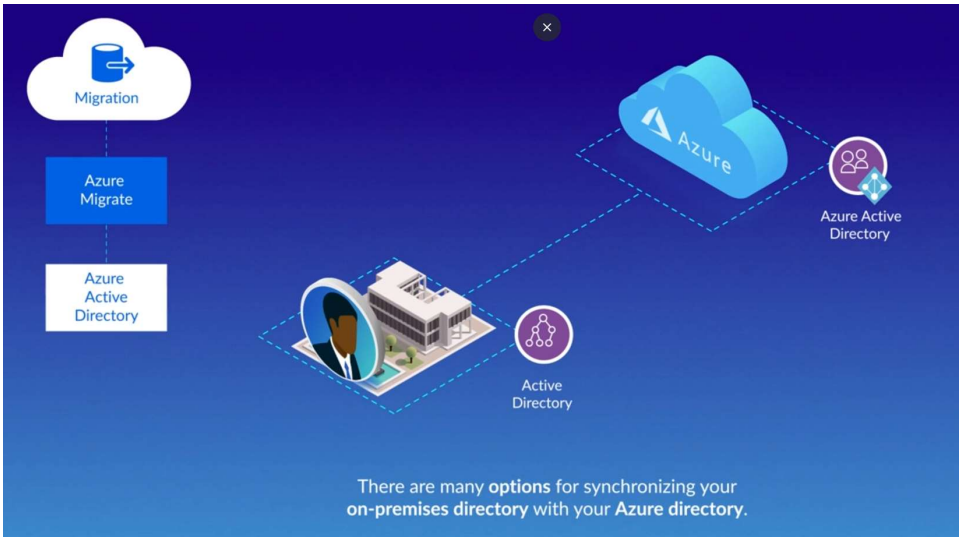


Screen clipping taken: 4/15/2022 5:04 PM

Azure Migrate is also integrated with other tools to help you migrate **SQL Server databases, web apps, and data.** Also, if you have a virtual desktop infrastructure, there's a tool that will do an assessment to help you migrate it to Windows Virtual Desktop, which is hosted on Azure.

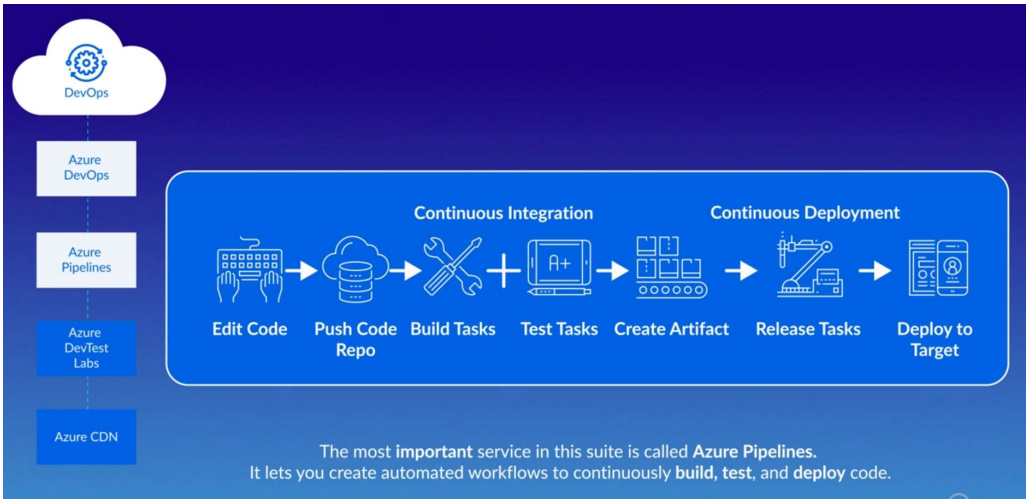
Screen clipping taken: 4/15/2022 5:05 PM

Another question that always comes up is, "How do we integrate our on-premises identities with Azure?" In most cases, organizations are using Active Directory for identity management. Naturally, Microsoft has a good solution for these customers. And, not surprisingly, it's called **Azure Active Directory.** This is a managed identity service that takes care of authentication. It's not exactly the same as Active Directory, but it's very similar, and there are many options for synchronizing your on-premises directory with your Azure directory.



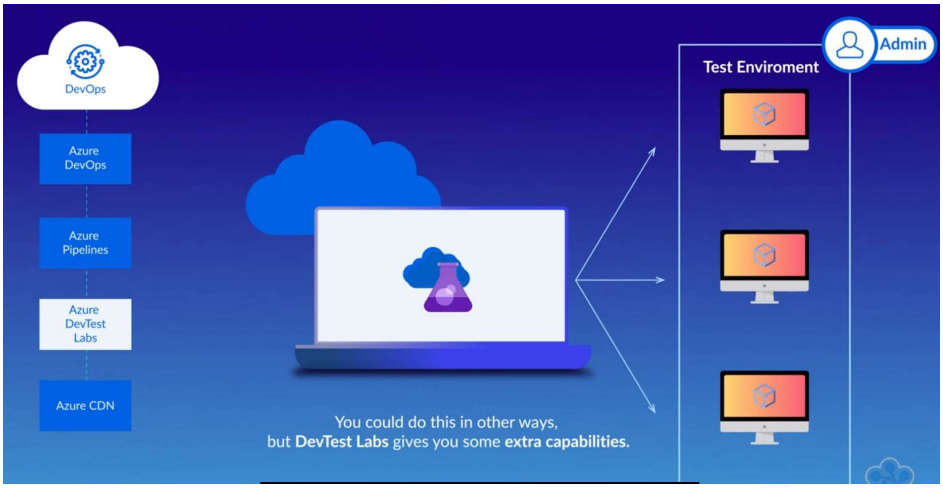
Screen clipping taken: 4/15/2022 5:06 PM

Let's move on to the development area. The DevOps approach has spread rapidly in organizations around the world. If you're not familiar with it, the idea is that you can automate large portions of the building, testing, and releasing of application updates. Microsoft offers a suite of services called **Azure DevOps** to help you implement these processes. The most important service in this suite is called **Azure Pipeline**s. It lets you create automated workflows to continuously build, test, and deploy code.



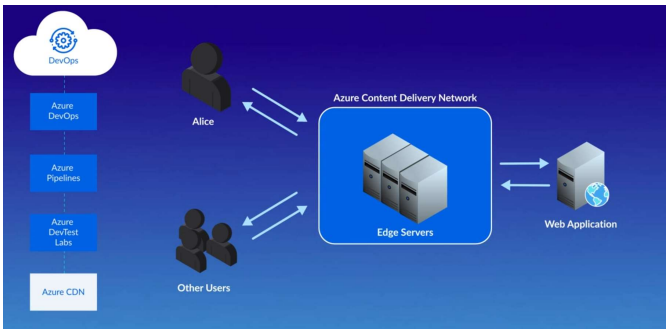
Screen clipping taken: 4/15/2022 5:16 PM

Another service that helps with the development process is **Azure DevTest Labs**, which makes it easy to spin up non-production environments. You could do this in other ways, but DevTest Labs gives you some extra capabilities, such as allowing administrators to control costs by setting limits on how many VMs can be deployed at once and ensuring that VMs are shut down when they're not in use.



Screen clipping taken: 4/15/2022 5:17 PM

One really helpful service for speeding up the responsiveness of your applications **is Azure Content Delivery Network**, which lets you take advantage of Microsoft's extensive global network. It caches your most frequently accessed content in locations around the world so your end-users will retrieve it from the closest point on the network. This really helps with making your web applications feel more like local applications.

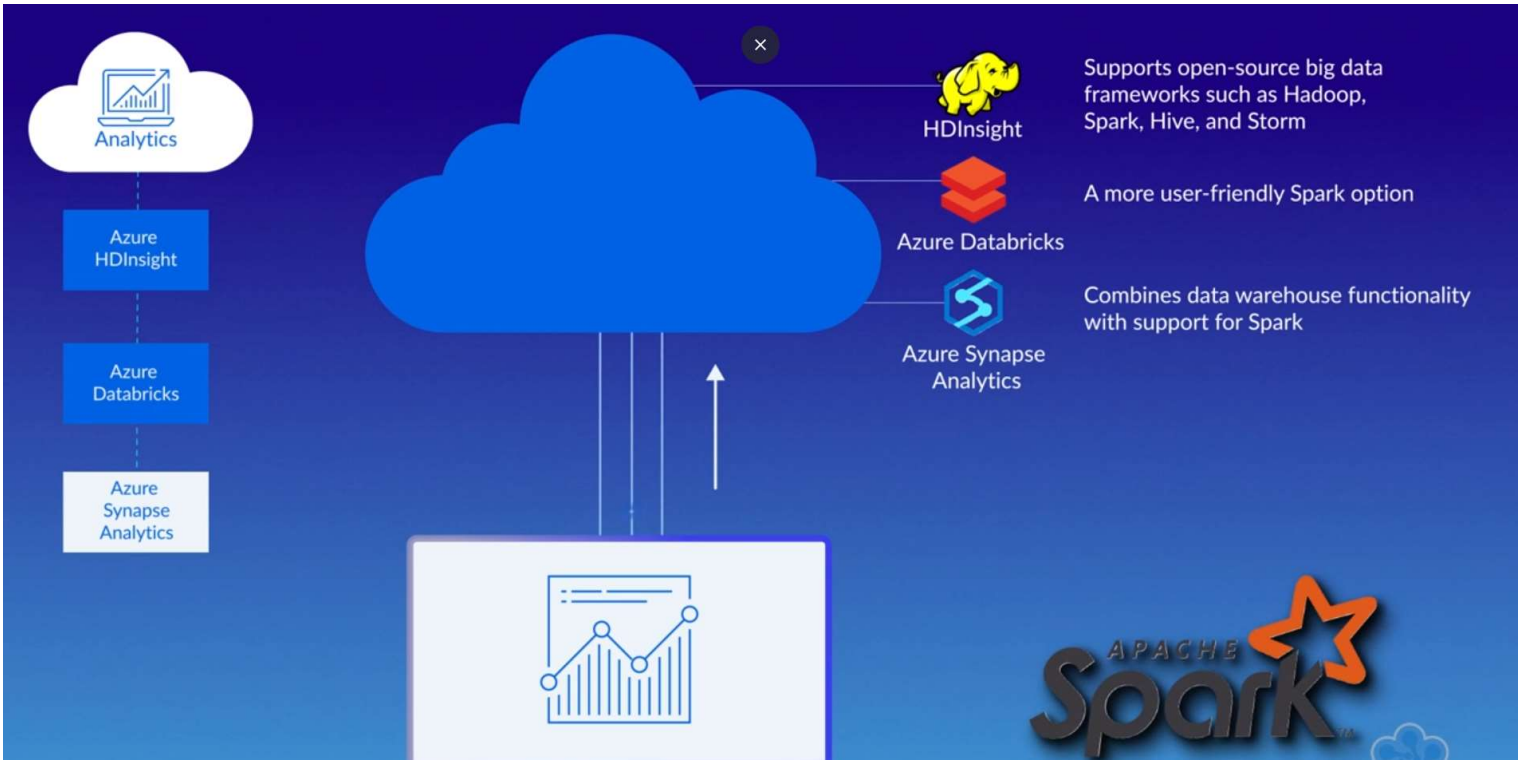


Screen clipping taken: 4/15/2022 5:19 PM

Although there are billions of computers connected to the internet, they're dwarfed by the number of other devices connected to the internet, such as smart thermostats or power meters. This is often referred to as the Internet of Things or IoT.
Microsoft offers a suite of services to help organizations connect, monitor, and control IoT devices. The simplest way to get started is to use Azure IoT Central, which is a fully

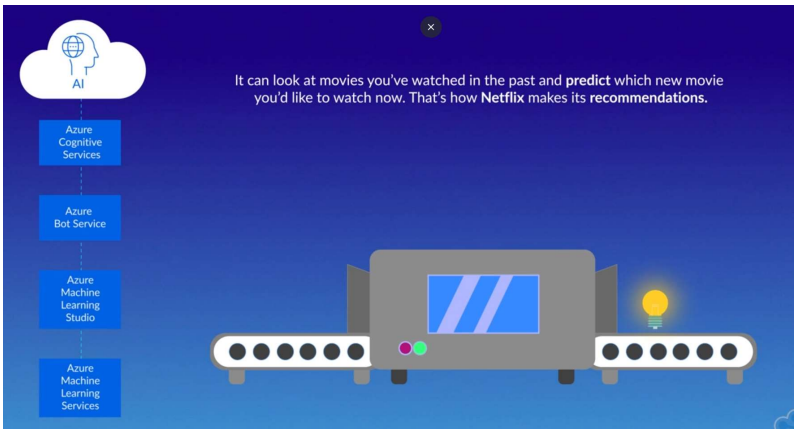
managed SaaS solution that takes care of the technical details for you. It lets you create IoT applications without writing any code.
If you need something more customized, then you can integrate your applications with Azure IoT Hub. It's a service that handles secure communications with thousands, or even millions, of IoT devices. In fact, it's the service that IoT Central uses behind the scenes.
Microsoft also offers a solution called Azure Sphere to make your IoT devices more secure. It includes certified chips, the Azure Sphere operating system, and the Azure Sphere Security Service, all of which provide layers of protection for your IoT devices.
When you have a large volume of data coming in, whether it's from IoT devices or applications, you'll probably want to perform analytics on it. Microsoft's analytics offerings have evolved over time, which is why you'll see a variety of services in this area.

- The oldest one is HDInsight. It supports a wide variety of open-source big data frameworks, including Hadoop, Spark, Hive, Storm, and many others.
- Azure Databricks is a similar service because it runs Spark as well, but it's more user-friendly and easier to manage than HDInsight.
- Azure Synapse Analytics is the new version of Azure SQL Data Warehouse. It includes all of the old data warehouse functionality, but it also supports Spark analytics.
- Have you noticed a common theme? Apache Spark seems to be the king of big data analytics, and it's just a question of which service you want to use to run it.

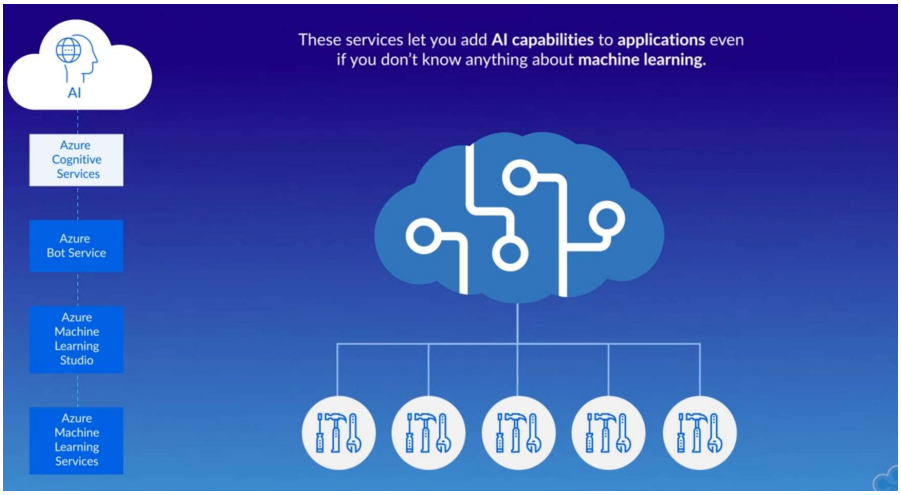


Screen clipping taken: 4/16/2022 2:59 PM

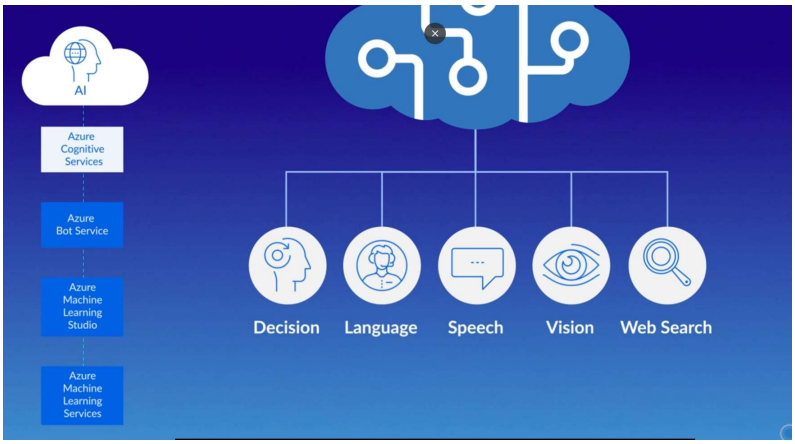
- A more sophisticated type of analytics is artificial intelligence.
- You've probably heard about the amazing advances in AI that have enabled computers to do everything from language translation to facial recognition to beating humans at games like chess and Go.
- Even though AI seems like it must be incredibly complex, the basic idea is fairly simple.
- The most common method is called machine learning. The way it works is you feed lots of real-world data into a program and the program tries to make generalizations about the data. This is known as **training a model**.
- It then uses these generalizations to make predictions when it's given new data. For example, it can analyze the viewing habits of millions of Netflix customers and make generalizations about the kinds of movies that different types of people like to watch. Then it can look at movies you've watched in the past and predict which movie you'd like to watch now. That's how Netflix makes its recommendations.



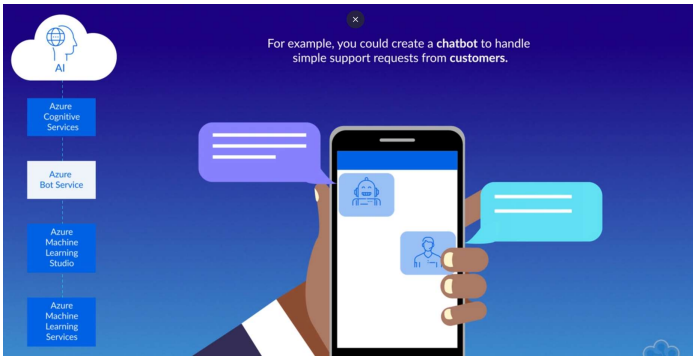
Screen clipping taken: 4/16/2022 3:01 PM



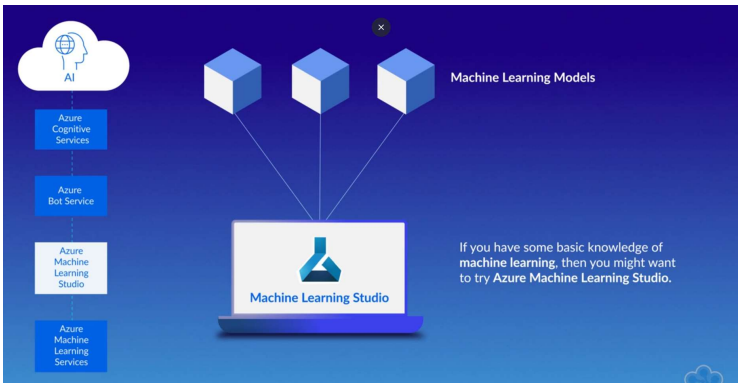
Screen clipping taken: 4/16/2022 3:02 PM



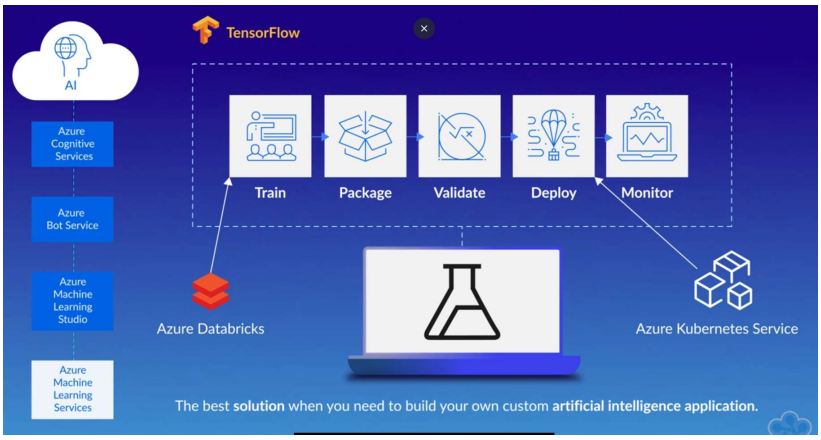
Screen clipping taken: 4/16/2022 3:02 PM



Screen clipping taken: 4/16/2022 3:03 PM

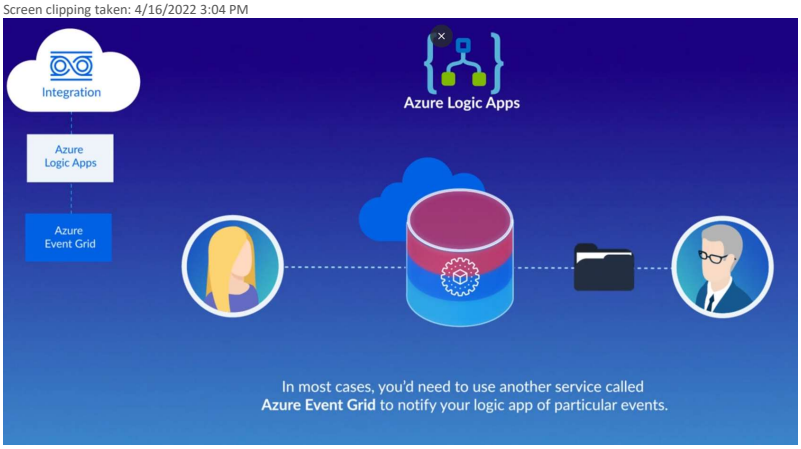


Screen clipping taken: 4/16/2022 3:03 PM

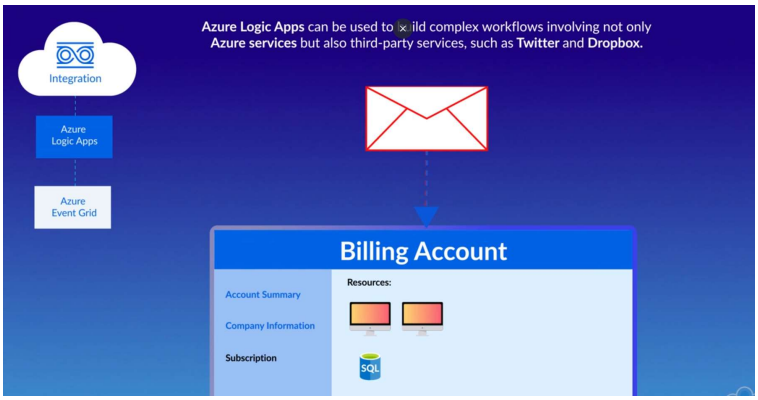


Screen clipping taken: 4/16/2022 3:04 PM

Screen clipping taken: 4/16/2022 3:04 PM



Screen clipping taken: 4/16/2022 3:05 PM



Screen clipping taken: 4/16/2022 3:05 PM

Screen clipping taken: 4/16/2022 3:06 PM

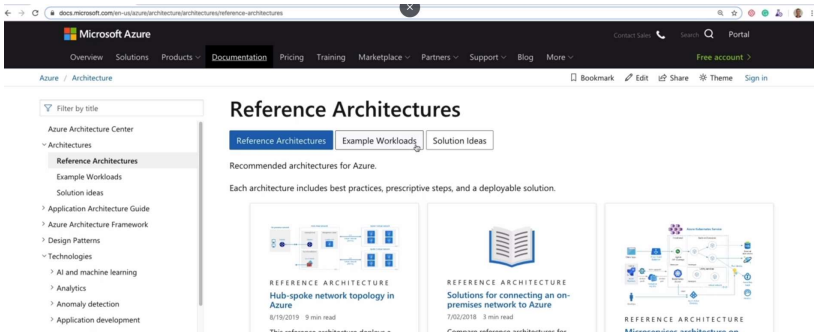
Screen clipping taken: 4/16/2022 3:06 PM

Microsoft offers lots of different AI services. If you're new to AI, then the best place to start is Azure Cognitive Services. This is a collection of pre-built artificial intelligence tools. These services let you add AI capabilities to applications even if you don't know anything about machine learning.
They're grouped into five categories: decision, language, speech, vision, and web search. For example, the vision category includes the Computer Vision API, which can classify images, and the Face API, which can detect faces in images.
A related offering is Azure Bot Service, which gives you the tools to create a chatbot. This is an intelligent agent that can answer questions. For example, you could create a chatbot to handle simple support requests from customers.
If you have some basic knowledge of machine learning, then you might want to try Azure Machine Learning Studio. It lets you train and deploy machine learning models without any coding, using a drag-and-drop interface. I highly recommend it for learning the basics of machine learning.
A much more sophisticated option is Azure Machine Learning Services, which gives you full control over every stage of the machine learning process. You can use any Python-based machine learning framework, such as TensorFlow or PyTorch, train models using services such as Azure Databricks, and deploy models using services such as Azure Kubernetes Service. Azure Machine Learning Services is usually the best solution when you need to build your own custom artificial intelligence application.
The last category I'll mention is integration tools. Since we use so many applications and services, it would be nice to be able to perform certain tasks on them automatically. For example, suppose you have an Azure blob container that your customer uploads documents to, and you'd like to be notified by email as soon as one arrives so that you can respond to it as quickly as possible. Microsoft offers a service called Azure Logic Apps that lets you automate this sort of task without writing any code. You can create a logic app using a drag-and-drop interface.
In this example, the logic app would be able to detect events that occur in Blob storage, but in most cases, you'd need to use another service called Azure Event Grid to notify your logic app of particular events. For example, if you want to get an email every time a virtual machine is created in a subscription, then you would configure Event Grid to send a message to your logic app whenever this occurs.
These examples are actually pretty trivial. Azure Logic Apps can be used to build complex workflows involving not only Azure services but also third-party services, such as Twitter and Dropbox.

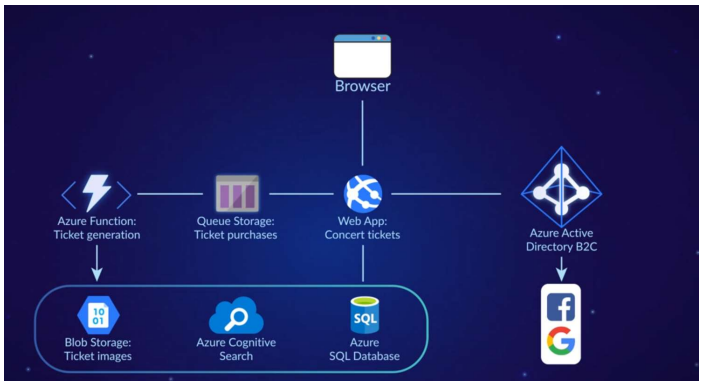
I've covered some of the most important Azure services, but there are many more that I didn't cover. So if you have a need that doesn't seem to fit with any of these services, there are plenty of other options. First, you can look through Microsoft's Azure Products page. Another way is to select all services from the main menu in the Azure Portal and then search for what you need. If you still can't find what you're looking for, then you can search the Azure Marketplace from the search bar at the top of the portal. The Marketplace includes a wide variety of third-party solutions that work with Azure, such as Docker Enterprise or Barracuda Firewall.
Okay, that's it for my services overview.

From <https://cloudacademy.com/course/overview-of-azure-services/azure-service-categories/>

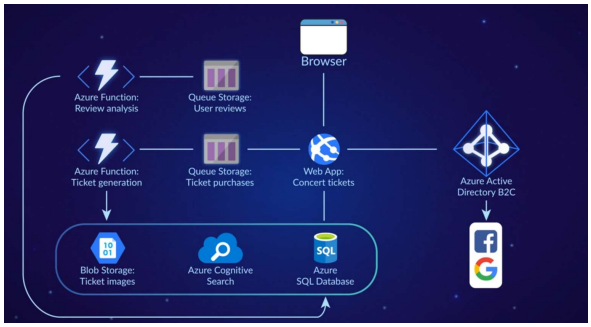
Saturday, April 16, 2022 3:07 PM



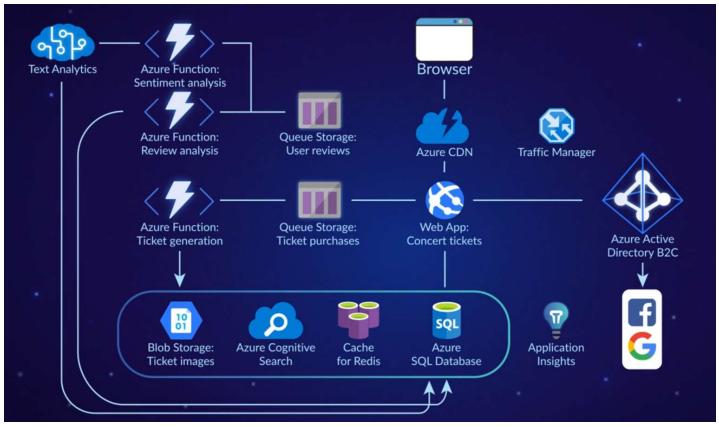
Screen clipping taken: 4/16/2022 3:07 PM



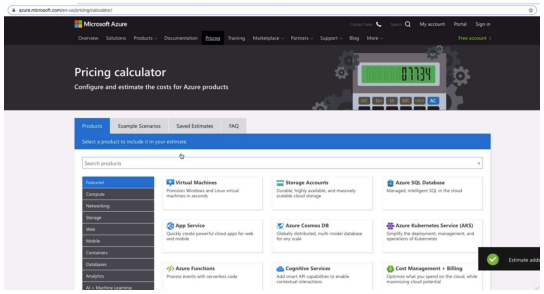
Screen clipping taken: 4/16/2022 3:10 PM



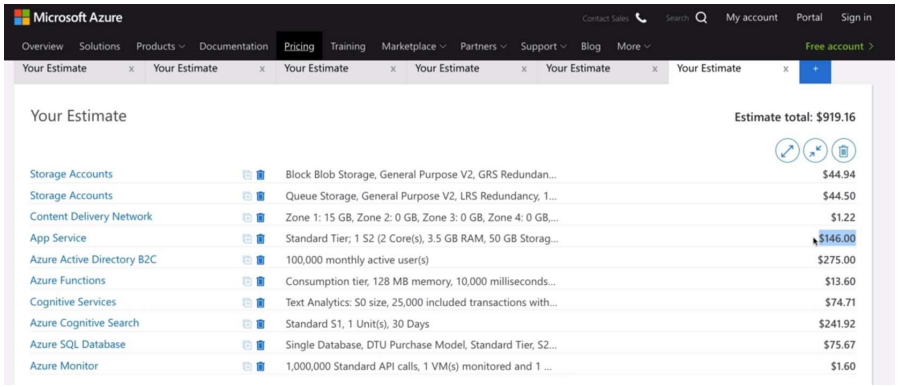
Screen clipping taken: 4/16/2022 3:11 PM



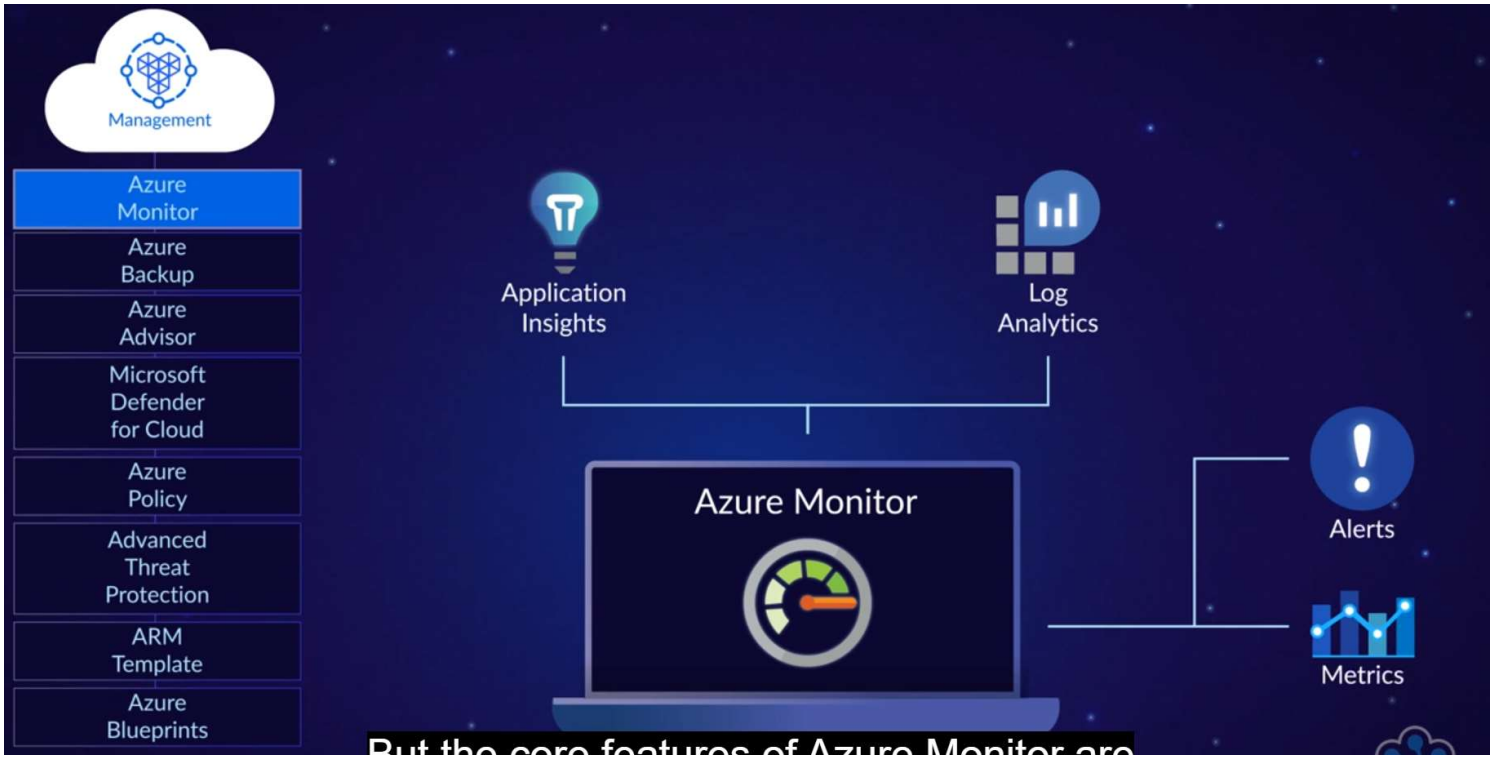
Screen clipping taken: 4/16/2022 3:13 PM



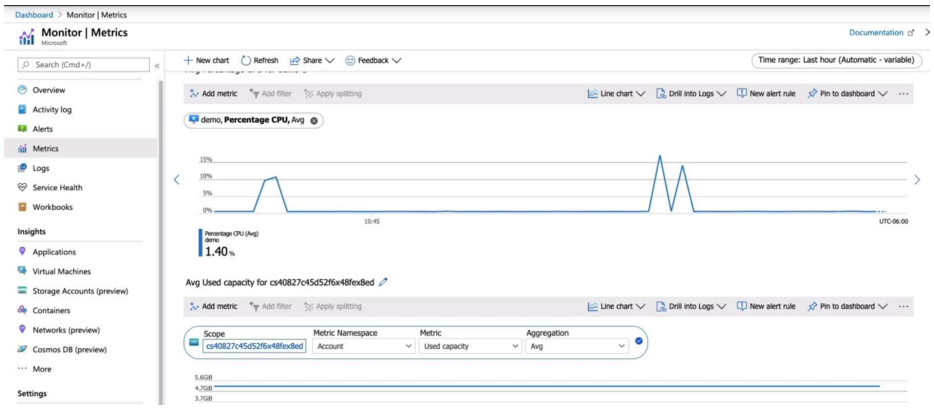
Screen clipping taken: 4/16/2022 3:13 PM



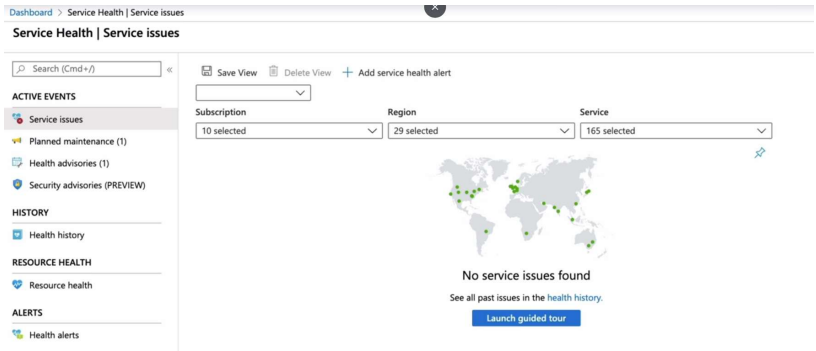
Screen clipping taken: 4/16/2022 3:15 PM



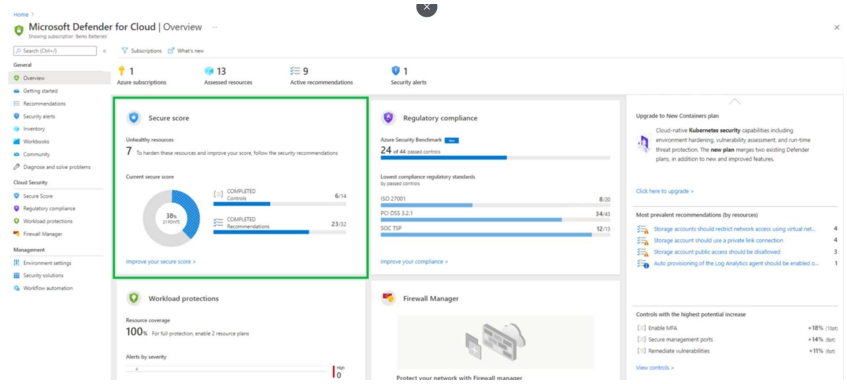
Screen clipping taken: 4/16/2022 3:16 PM



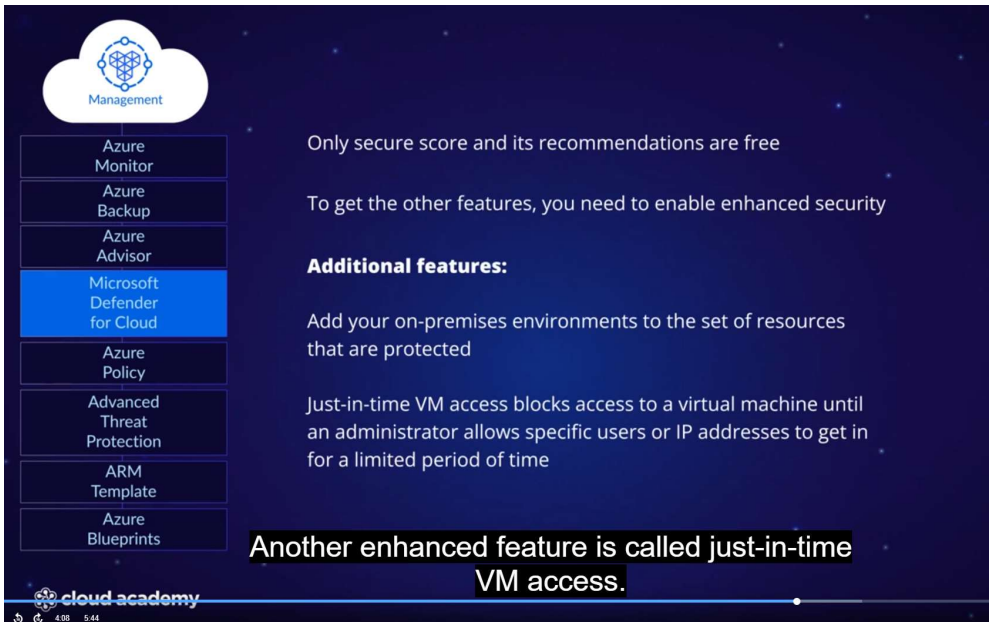
Screen clipping taken: 4/16/2022 3:17 PM



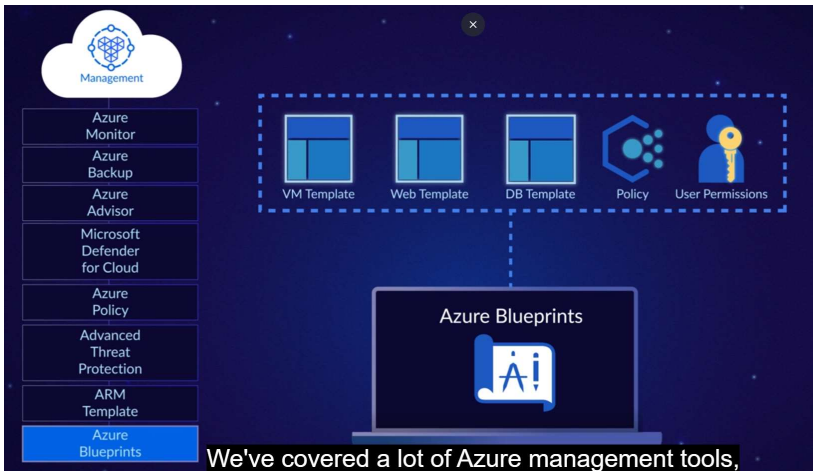
Screen clipping taken: 4/16/2022 3:17 PM



Screen clipping taken: 4/16/2022 3:20 PM



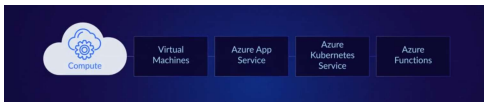
Screen clipping taken: 4/16/2022 3:22 PM



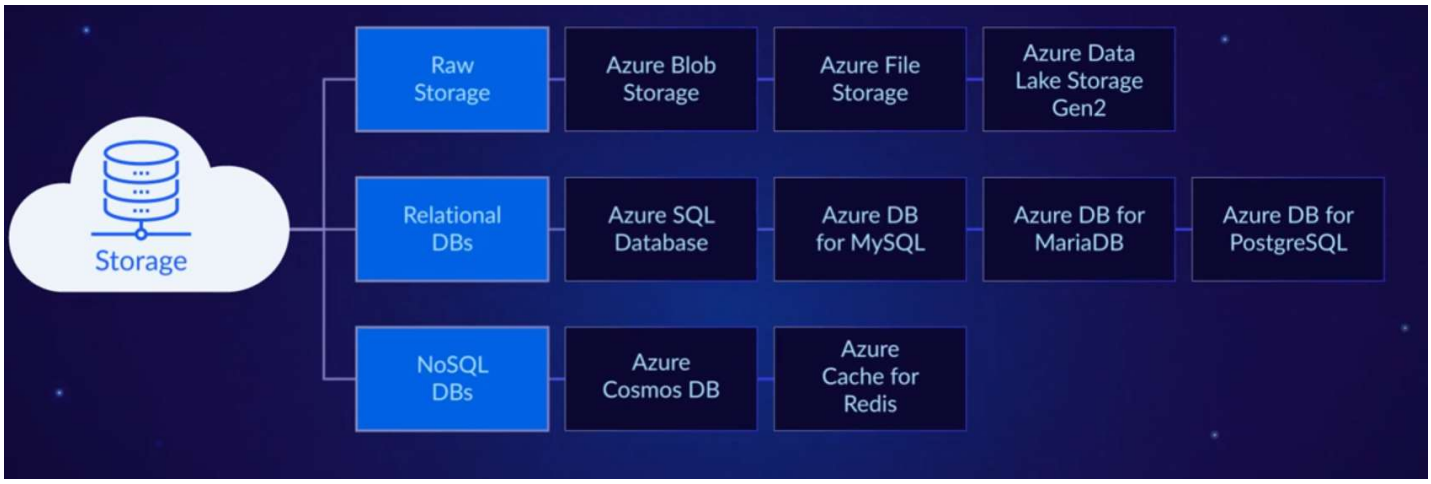
Screen clipping taken: 4/16/2022 3:23 PM

Screen clipping taken: 4/16/2022 3:16 PM

Saturday, April 16, 2022 3:24 PM

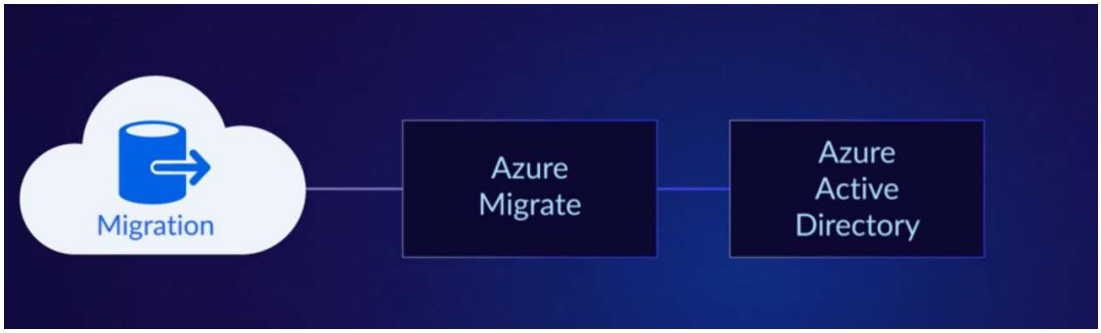


Screen clipping taken: 4/16/2022 3:25 PM



Screen clipping taken: 4/16/2022 3:26 PM

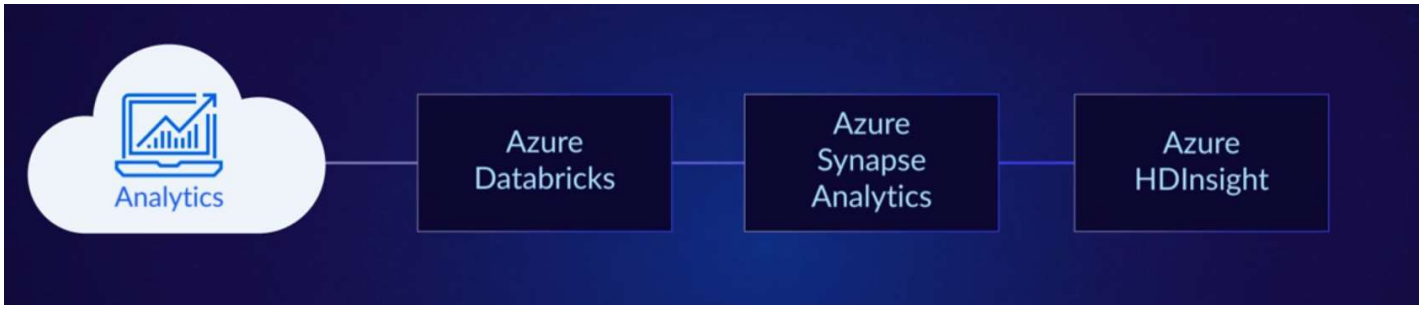
Screen clipping taken: 4/16/2022 3:26 PM



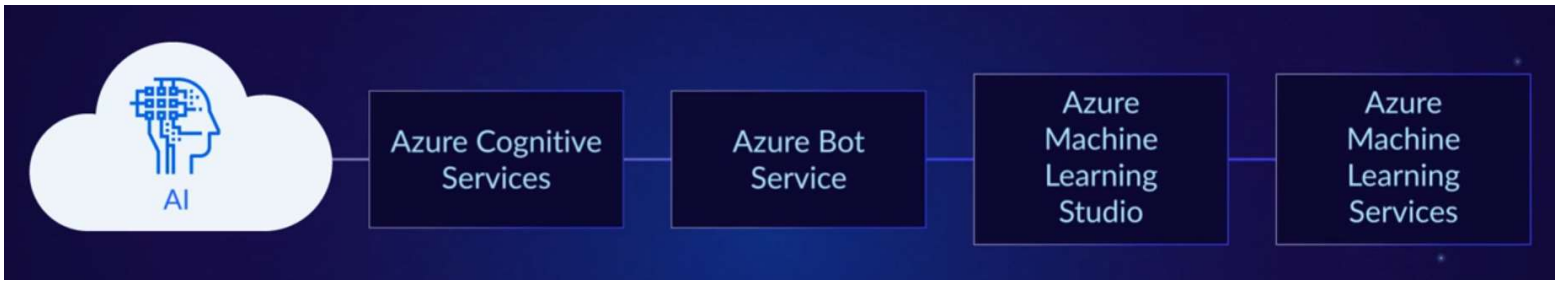
Screen clipping taken: 4/16/2022 3:27 PM

Screen clipping taken: 4/16/2022 3:27 PM

Screen clipping taken: 4/16/2022 3:28 PM



Screen clipping taken: 4/16/2022 3:28 PM

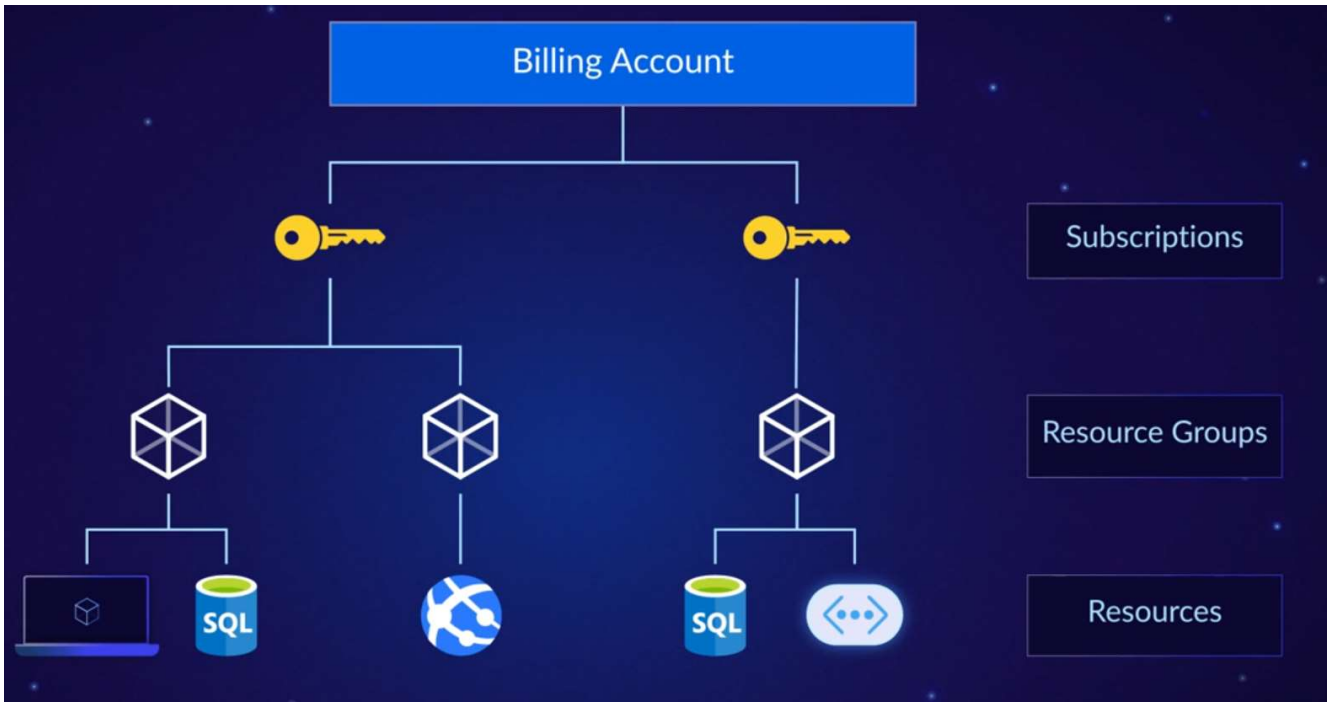


Screen clipping taken: 4/16/2022 3:28 PM

Screen clipping taken: 4/16/2022 3:29 PM

Screen clipping taken: 4/16/2022 3:29 PM

Screen clipping taken: 4/16/2022 3:30 PM



Screen clipping taken: 4/16/2022 3:31 PM

Screen clipping taken: 4/16/2022 3:29 PM

ADI - Introduction

Saturday, April 16, 2022 3:35 PM

Screen clipping taken: 4/16/2022 3:35 PM

Welcome to “Designing an Azure Data Implementation”. My name’s Guy Hummel and I’ll be helping you with the data aspects of architecting an Azure solution. I’m a Microsoft Certified Azure Solutions Architect, and I’m the Azure Content Lead at Cloud Academy. If you have any questions, feel free to connect with me on LinkedIn and send me a message, or send an email to support@cloudacademy.com.
This course is intended for people who want to become Azure cloud architects.
To get the most from this course, you should have a general knowledge of IT architecture, especially databases.
We’ll start with an overview of Azure Storage. Then we’ll go over some data services that don’t fit in the usual storage or database categories, such as Data Factory and HDInsight.
Next, we’ll look at relational databases, especially Azure SQL Database.
After that, we’ll go through Azure’s NoSQL services, such as Azure Data Lake and Azure Redis Cache.
Finally, we’ll go into quite a bit of detail on Cosmos DB, Microsoft’s global, multi-model database.
By the end of this course, you should be able to identify the most appropriate Azure services for various data-related needs; design an Azure SQL Database implementation for scalability, availability, and disaster recovery; and design an Azure Cosmos DB implementation for cost, performance, consistency, availability, and business continuity.
We’d love to get your feedback on this course, so please give it a rating when you’re finished.
Now, if you’re ready to learn how to get the most out of Azure’s data services, then let’s get started.

From <https://cloudacademy.com/course/designing-an-azure-data-implementation/designing-an-azure-data-implementation-introduction/?context_resource=lp&context_id=3191>

ADI - Azure Storage

Saturday, April 16, 2022 3:36 PM

If you need to store data, but you don't need a full-blown database, then Azure Storage is the way to go. First, it's durable and highly available, because it stores all data redundantly. Second, it's secure, because all data is encrypted automatically, and you can set fine-grained access control to it. Third, it's scalable, because you can always add more data without having to worry about provisioning hardware to hold it. There *is* a 500 terabyte limit, but if you need more than that, you can contact Azure Support. Fourth, it's a managed service, so you don't have to worry about maintenance. And finally, it's accessible over the web.
Although data is always stored redundantly, there are four different options, depending on your needs. Note that not all of these options are available in every region or for every type of data. We'll go from cheapest and least redundant to most expensive and most redundant.
Locally-redundant storage (LRS) is replicated across racks in the same data center. This means that if there is a disaster at that data center, your data could be lost. Although this is highly unlikely, you should only use locally-redundant storage if you can easily reconstruct your data.
Zone-redundant storage (ZRS) is replicated across three zones within one region, so if an entire zone goes down, your data will still be available.
Geo-redundant storage (GRS) is replicated across two regions, so even if *an entire region* goes down, your data will still be available. However, in the case of a regional disaster, you'd have to wait for Microsoft to perform a geo-failover before you could access your data in the secondary region.
That's why you may want to consider using read-access geo-redundant storage (RA-GRS). It's the same as geo-redundant storage except that if there's a disaster in your primary region, then you can read your data from the secondary region immediately. You won't have write access, though, so if you can't wait until Microsoft restores availability in the primary region, then you'll have to copy your data to yet another region and point your applications to the new location.
You can copy data to and from Azure Storage using the AzCopy utility, Azure PowerShell, or the Azure Storage SDK, which is available for a variety of programming languages.
Azure Storage supports four types of data: blobs, files, queues, and tables. Blob stands for binary large object, but really a blob is just a file. So why is there a distinction between blob storage and file storage? The difference is in how they're organized. Blobs aren't really organized at all. Sure, you can use slashes in their names, which makes it look like they have a folder structure, but they're not actually stored that way.
File storage, on the other hand, has the sort of hierarchical structure you'd expect in a filesystem. In fact, it's SMB-compliant, so you can use it as a file share. This makes it easy to move an on-premises file server to Azure. Even better, you can make this file share globally accessible over the web, if you want. To do that, users need a shared access signature token, which allows access to particular data for a specific amount of time.
You might be tempted to use File storage instead of Blob storage, even when you don't need an SMB-compliant file share, but bear in mind that it's significantly more expensive than Blob storage. If you just need a place to put files, whether they're documents or videos or logs or anything else, then you should use Blob storage, which is by far the cheapest of all the storage types.
There are options for making Blob storage even cheaper too. You can choose from three storage tiers: hot, cool, and archive. Hot storage is the tier you'll probably use the most often. It's intended for data that gets accessed frequently. If you have data that *doesn't* get accessed frequently, then you should consider the cool storage tier.
It's optimized for data that still needs to be retrieved immediately when requested, even though it doesn't get accessed very often. An example would be a video file that people rarely watch. The cool tier has a much lower storage cost, but a much higher cost for reads and writes. The data also needs to be in the cool tier for at least 30 days.
If you have data that will almost never be accessed and you can live with it taking up to 15 hours to access when you do need it, then the archive tier is a way to save lots of money. It's 5 times cheaper than the cool tier for storage costs, but it's dramatically more expensive for read operations. The data also needs to reside in the archive tier for at least 180 days.
You can move data between the tiers anytime you want, but if you do it before the minimum

duration for the cool or archive tiers, then you'll be charged an early deletion fee. For example, if you put data in the archive tier and then move it back to the cool tier 90 days later, you'll be charged half of the early deletion fee, since you moved the data when there was still half of the 180-day minimum left to go.
Queue storage is a very different option. It's intended for passing messages between applications. One application pushes messages onto the queue and another application asynchronously retrieves those messages from the queue, one at a time, and processes them. We'll talk more about messaging systems in another course.
I find Table storage to be the most surprising type of Azure Storage. It's a NoSQL datastore with storage costs that are about the same as File storage and with way cheaper transaction costs. I think Microsoft realized what a good deal this is too, because they now have a premium version of Table storage that's part of their CosmosDB service. We'll talk about both versions of Table storage again in other courses.
I should also mention that there's another type of Azure Storage. Disk storage is what's used for the disks that are attached to virtual machines, but we don't need to go into the details here.
Microsoft is very good about providing options for hybrid solutions where you can use both Azure and your existing on-premises infrastructure. StorSimple is Microsoft's hybrid offering in the storage area. It's a virtual array that you install at your own site and it's primarily used to do backup, recovery, and storage tiering.
Of course, you don't need to use StorSimple to copy or move your data to the Azure cloud, but it makes the process so much easier. For example, here's how storage tiering works. You provision enough local storage for your current needs, and then as you need more capacity, it moves your infrequently accessed data to the cloud. It even deduplicates and compresses the data before sending it, so your Azure storage costs are as low as possible. This all happens seamlessly in the background.
If you have a disaster locally and need to recover data from the cloud, StorSimple restores the metadata first and then restores the data itself as needed. That way, you can get up and running again very quickly.
And that's it for Azure Storage.

From <https://cloudacademy.com/course/designing-an-azure-data-implementation/designing-an-azure-data-implementation-azure-storage/?context_resource=lp&context_id=3191>

ADI - Azure Data Services

Saturday, April 16, 2022 3:36 PM

In this lesson, we'll cover services that don't fall into the usual categories of basic storage, transactional databases, or NoSQL datastores. These services help you work with data in other ways, such as finding, transforming, and analyzing it.
First up is Azure Purview. Organizations typically have so much data in so many different places that it's hard to find what you're looking for. The purpose of Azure Purview is to act as an index to all of those data sources, so you can discover them. Of course, in order for this to work, your employees need to register their data sources in the catalog. The data itself stays where it is, but its location and the metadata about it get added to the catalog. The metadata includes things like column names and data types. Users can also add additional information about a data source, such as a description or some tags.
Once various data sources are registered, people can search the catalog to find what they're looking for using Purview Studio
Another way to deal with pockets of data is to collect it all in either a data lake or a data warehouse. These serve two different, but related, needs.
Data warehouses store data in structured, relational tables, while data lakes store any kind of data, whether it's structured or not. For example, you could store everything from documents to images to social media streams.
Data warehouses are generally used for business reporting, while data lakes are more often used for data analytics and exploration. In fact, one common setup is to process data in a data lake and then export it to a data warehouse. Both types of services are designed for performing massive queries at high speed.
Azure Data Lake Storage is built on top of Azure Blob Storage, and it provides the additional capabilities needed for a modern data lake. Its most important feature is that it's compatible with Hadoop and Spark, which are the most popular open-source software systems for doing data analytics.
Azure Synapse Analytics (formerly known as SQL Data Warehouse) offers an interesting mix of data warehouse and data lake capabilities. If you need a data warehouse, you can create a SQL pool, which lets you run SQL queries on structured, relational tables. If you want a data lake, then you can create a Spark pool, which lets you use Spark to query both structured and unstructured data.
Spark has become so popular that Microsoft has many services that let you use Spark for data analytics. In addition to Data Lake Storage and Synapse Analytics, you can also use Azure Databricks and Azure HDInsight. Databricks is a managed Spark implementation that was developed by the people who created Apache Spark. HDInsight supports a wide variety of open-source big data frameworks, including Hadoop, Spark, Hive, Storm, and many others.
One difference between Databricks and HDInsight is ease of use. For example, to run a processing job with either service, you need to spin up a cluster, but Azure Databricks can be configured to automatically spin up a cluster when a job runs and shut it down after the job is finished. In contrast, HDInsight doesn't have a built-in way to spin up a cluster automatically. So if you need to run HDInsight jobs quite often, you can leave a cluster running all the time, which would be expensive, or you could spin clusters up and down as you need them, which would be kind of a pain.
One way to make HDInsight work in a more automated fashion is to use yet another service, Azure Data Factory. It lets you create workflows to automate data movement and data transformation. One of its many capabilities is spinning up and down HDInsight clusters as needed, but it can do far more than that.

With Data Factory, you can create data processing pipelines. For example, a pipeline could copy data from SQL Server to Data Lake Storage, run a Spark job on the data using an HDInsight cluster, and store the results in Synapse Analytics, all without any human intervention. It can even automate machine learning jobs. It's such a useful tool that Microsoft even includes a stripped-down version of it in Synapse Analytics.
One more data analytics tool is Azure Analysis Services. It lets you create data models that make sense of existing data. One of the problems with the multitude of data in organizations is that it can be hard to understand how all of that data relates to the real world. Using a data model is easier than working with the raw data. Analysis Services also makes browsing large amounts of data faster because it uses in-memory caching. However, end users don't browse directly through Analysis Services. Instead, they use one of the supported client tools, such as Power BI, Tableau, or Excel.
And that's it for Azure Data Services.

From <https://cloudacademy.com/course/designing-an-azure-data-implementation/designing-an-azure-data-implementation-azure-data-services/?context_resource=lp&context_id=3191>

ADI - Relational Database Storage

Saturday, April 16, 2022 3:37 PM

SQL Server has been one of the most popular relational databases for a very long time, so it's not surprising that Microsoft has made it available as a managed Azure service as well. But Azure supports other relational databases too. Microsoft offers a surprisingly high level of support for the most popular open-source relational databases, MySQL and PostgreSQL.

If you have existing databases running on either of these systems and you want to migrate them to Azure, then you should use Azure Database for MySQL and Azure Database for PostgreSQL. These are managed services that give you lots of extra goodies, including high availability, backups, security, and compliance. There's even a free service that can migrate your existing databases to Azure with minimal downtime. The managed service price is quite reasonable, so there isn't much reason to run these databases directly on virtual machines, because you'd have to manage them yourself.

OK, back to SQL Server. The most obvious way to migrate your SQL Server databases is to use Azure SQL Database. However, if you're currently running SQL Server 2016 or higher, there's another option you may want to consider. It's called SQL Server Stretch Database.

It's very similar to StorSimple, except it's for database records instead of files. It migrates cold table rows (that is, infrequently queried rows) to Azure. This happens in the background automatically. And you can still query the data that has migrated to Azure too. The only difference is that the query will take a bit longer than usual.

Stretch Database is a good solution if your on-site data keeps growing. First, instead of buying more storage to handle the growth, you can just migrate your cold data to Azure, which is usually cheaper and easier. Second, as your data grows, backups take longer to run. Eventually, your backups may not be able to run within your backup window. By migrating your cold data to the cloud, your on-site backups run faster and your cloud backups happen automatically.

Of course, if your users almost never need to query your cold data, then it would be cheaper to store it offline than in Azure. But in most organizations, people do need to query cold data occasionally, and it would be a pain to bring that data back online every time it's needed, so storing it in Azure is usually a better choice.

Microsoft even provides a tool called Data Migration Assistant that makes recommendations on many aspects of data migration, including Stretch Database. It tells you which databases and tables would be good candidates for Stretch Database. It also tells you about potential blocking issues, because there are a number of constraints that can prevent data from being migrated, such as tables that have more than 1,023 columns.

If you'd rather bite the bullet and just move an entire database to Azure, then there are two ways to do it. Ideally, you would move it to Azure SQL Database, which is a managed version of SQL Server. One of the big advantages is that you wouldn't need to worry about maintenance anymore, because Microsoft takes care of all patches and updates, as well as all of the underlying infrastructure.

It isn't always possible to move an existing SQL Server database to Azure SQL Database, though, because they're not 100% compatible with each other. In the same Data Migration Assistant that I mentioned earlier, it tells you what reengineering, if any, would be required to move a database to the SQL Database service.

If the reengineering effort would be too great, then one option is to move the database to a SQL Server instance running on a virtual machine in Azure. The disadvantage is that you would have to take care of maintaining the instance.

So, Microsoft has an offering called SQL Managed Instance that gives you the best of both

worlds. It's nearly 100% compatible with SQL Server, but it's managed, so you don't have to worry about maintenance.

Microsoft provides a surprising number of options for running and scaling SQL Database. First, there are three service tiers: General Purpose, Hyperscale, and Business Critical. These service tiers have very different architectures from each other.

The General Purpose tier is the default. If you don't have any special size, performance, or availability requirements, then this is the best and least expensive choice. Despite its budget pricing, it still offers a low 5 to 10-millisecond latency and 99.99% availability. One of its limitations, though, is database size. It can't hold more than 4 terabytes (or 8 terabytes for a Managed Instance).

If you need more space than that, then the Hyperscale tier is the way to go. It supports databases of up to 100 terabytes. Not only that, but it can scale its compute resources up and down very quickly. It also provides instant backups and fast database restores.

If you need lightning-fast performance, then the Business Critical tier is the best choice. It provides 1 to 2-millisecond latency. It does this by using local SSD storage. Normally, it's risky to use local storage because that makes it more difficult to provide high availability, but Microsoft has solved that problem by using a 4-node cluster. Of course, that makes it more expensive, too. You can even get higher availability than the General Purpose tier by choosing the zone redundant option, which spreads the three read replicas in the cluster across the availability zones in a region. This gives it a 99.995% availability. You can't scale a database beyond 4 terabytes, though, so if you have a database that's bigger than that, then you'll still need to use the Hyperscale tier and lose the advantages of the Business Critical tier.

Okay, we've gone through the service tiers, but there are lots of other options as well. The next decision to make is which resource type to use. The choices are Single Database, Elastic Pool, and Managed Instance.

A single database is pretty self-explanatory. If you only need one database, then this is usually a good option.

The difference between the single database model and the elastic pool model is that an elastic pool can contain multiple databases that share resources with each other. This model works well if your databases are not all busy at the same time.

For example, suppose that you have five databases, and they all experience spikes in demand occasionally, but they don't all experience these spikes at the same time. Putting them in a pool of shared resources would be much cheaper than provisioning five single databases because you'd have to provision enough resources for each of the single databases to handle spikes, even though they wouldn't need that capacity most of the time. With elastic pools, you can also set minimum and maximum resources for each database to ensure that one database doesn't consume all of the resources in the pool.

As I mentioned earlier, a Managed Instance has the highest compatibility with SQL Server. If you need to migrate an existing SQL Server database to Azure, then Managed Instance is the easiest way to do it. One disadvantage, though, is that Managed Instance doesn't work with the Hyperscale tier, so it doesn't support databases larger than 8 terabytes. I should also mention that elastic pools don't work with the Hyperscale tier either.

All right, there's yet another option you need to know: purchasing models. There are two different ways you can pay for your databases: DTUs or vCores.

Database Transaction Units, or DTUs, used to be the only option. A DTU is a balanced bundle of compute, storage, and I/O resources. The problem with this model is that it's not very flexible. For example, if you have a workload that needs a lot of storage but not much compute power, then you'll be paying for a lot of excess compute power that you don't need.

Microsoft introduced the Virtual Core, or vCore, model to provide more flexibility, and it's now the recommended purchasing model. It allows you to select the number of virtual CPU cores separately from the maximum storage space size. It also lets you specify the hardware generation, which includes options for memory-optimized and compute-optimized configurations.

And we're not done yet because there's another purchasing option within the vCore model. You can choose either Provisioned or Serverless. With the Provisioned model, your SQL Database instance gets provisioned with the exact resources you requested, and you get charged for the database as long it's running.

With the Serverless purchasing model, you specify the minimum and maximum number of vCores, and it will autoscale based on workload demand. Then you can configure it so that if there's no activity for a period of time, it will pause the database, and you won't be charged for any compute resources until new activity causes the compute resources to come back up.

This sounds great, doesn't it? So why wouldn't you always choose the Serverless option? Well, the cost per vCore is higher for Serverless than it is for Provisioned, so if your database was active most of the time, then the Serverless option would be more expensive.

By the way, even if you don't choose the Serverless option, you can still scale your database manually when you need to. You can just edit the database's configuration and change the number of vCores (or DTUs if you're using that model). There will be a brief loss of connectivity while it switches over to the new resources, but it generally takes less than four seconds, so it shouldn't be very noticeable.

Regardless of which service tier you choose, high availability within a region is handled automatically. SQL Database provides HA through a technology similar to Always ON Availability Groups in SQL Server. The good news is that you don't have to do anything, even if there is a failure, because failover is automatic too. It may take up to 30 seconds to recover, but that should be acceptable in most cases.

To protect against regional failures, you need to configure active geo-replication. This lets you create up to four read replicas in different regions. If the primary database becomes unavailable, then you (or your application) can failover to one of the secondary databases. That secondary then becomes the new primary.

To make this work, you need to ensure that your secondaries have user authentication configured in the same way as the primary. One way to do this is to use Azure Active Directory so you don't need to manage credentials manually on your databases.

You should also use the same firewall rules for secondaries as you do for your primary. One way to do this is to use database-level firewall rules, which are replicated with the database.

Active geo-replication also has the side benefit that you can use the secondary databases to make queries faster for users in other regions. This is possible because the secondaries are read replicas, so as long as a user doesn't need to write to the database, they can connect to the nearest secondary instead of the primary.

If you're using a Managed Instance, then disaster recovery is handled by auto-failover groups, which is a layer on top of active geo-replication. Auto-failover groups allow you to failover multiple databases at the same time, which is a really nice feature.

Even with high availability, backups are still important, so you can recover from cases of accidental corruption or deletion. Luckily, SQL Database automatically takes care of backups too. It saves them in read-access geo-redundant storage (RA-GRS) to protect against regional failures. Transaction log backups happen about every 5 or 10 minutes, so you won't lose much data if you do a point-in-time recovery.

Backups are saved for 7 days by default, but you can increase the retention period to up to 35

days. If you need to save them for longer than that, then you can configure a long-term backup retention policy for each database. You can keep backups for up to 10 years.

Normally, you would restore data to the same server where it was originally located, but if you need to restore a database to a different server, then you can use Geo-restore. As the name implies, this is usually used to restore a database to another geographic region. If you're not using active geo-replication, then this is the way to failover to another region. It typically takes about 12 hours, though, so it's not a good solution for mission-critical applications. It can also result in the loss of up to an hour's worth of data, since it can take up to an hour for the backup copy in the primary region to be replicated to the secondary region. If the primary region goes down, then the secondary region may not have the latest backup.

And that's it for relational database storage.

From <https://cloudacademy.com/course/designing-an-azure-data-implementation/designing-an-azure-data-implementation-relational-database-storage/?context_resource=lp&context_id=3191>

ADI - NoSQL Storage

Saturday, April 16, 2022 3:38 PM

Relational databases are great for online transaction processing, but the rise of big data has led to the need for bigger databases. NoSQL datastores are able to scale better because they satisfy fewer requirements than relational databases. Considering their name, it's not surprising that one requirement they don't fulfill is support for SQL queries.
Azure offers a variety of NoSQL datastores, some of which are intended for very specialized purposes. We'll start with the more general-purpose options.
Azure Table storage is a key/attribute store, so it's intended for simple, but structured, data. For example, you could use it for address books or user profiles. It has a schemaless design, so the structure of your data can change without any trouble. It also automatically indexes the records, so queries are fast, although it doesn't create secondary indexes, so some queries could be slower. If you need secondary indexes or global distribution, then you can use the premium version that's part of Cosmos DB. If you don't need those features, then the basic version of Table storage is a great choice because it's an extremely cheap NoSQL datastore that's easy to use. If you need more sophisticated features like complex joins, foreign keys, or stored procedures, then you should use a relational database instead.
Azure Redis Cache is another general-purpose datastore, but it's used under specific conditions. It's intended to speed up data retrieval in applications. It's a managed service for Redis, which is an open-source, in-memory database. Since its data resides in memory, it's very fast, which is why it's used as a cache. It's even simpler than Azure Table storage, because it just stores key/value pairs.
Azure Redis Cache is offered in 3 tiers. Basic doesn't have an SLA, so it should only be used for testing and development. The Standard tier provides a replicated, high availability cache. The Premium tier has better performance and can handle bigger workloads, disaster recovery, and other advanced features.
Azure Data Lake Storage is different from Table storage and Redis because it's not a key/value store. It's intended to hold large quantities of any kind of data. It's essentially a data warehouse for unstructured data and its main purpose is data analytics.
Azure Search is a very specialized database. It creates an index of text data so your users can easily run searches on it. You can then embed search functionality into your web, mobile, and enterprise applications. Azure Search also provides a wide variety of features, such as search suggestions, language analyzers, and fuzzy searches.
Time Series Insights (or TSI) is another highly specialized database. It collects time-stamped data, such as data from IoT devices. Since this is a very common use case, TSI integrates with Azure IoT Hub and Azure Event Hubs. Once the data resides in TSI, you can run queries on billions of events and get a response in seconds. You can run these queries either interactively or from your applications. When you run queries interactively using TSI explorer, you can see visualizations of the data.
Azure has another NoSQL storage solution, Cosmos DB, but it's such a big topic that it'll take up a whole lesson. So, if you're ready, then go to the next video.

From <https://cloudacademy.com/course/designing-an-azure-data-implementation/designing-an-azure-data-implementation-nosql-storage/?context_resource=lp&context_id=3191>

ADI - Cosmos DB

Saturday, April 16, 2022 3:38 PM

Cosmos DB is a pretty amazing database service. It used to be called DocumentDB, but Microsoft added so much new functionality to it that they had to rename it. Cosmos DB is such an unusual database service that it can take a while to understand it. I'll try to sum it up as briefly as I can.

Cosmos DB's two most important features are that it's global and it's multi-model. Not only is it extremely easy to replicate a Cosmos DB database to multiple regions around the world, but Microsoft even provides service level agreements for *latency*, which isn't something you can get with normal database services.

The multi-model feature may be even more innovative. Cosmos DB supports many different types of data models, including document, key-value, graph, and wide column. It isn't the only database system that supports multiple models, but it's the first to be offered as a global cloud service.

As I mentioned in the NoSQL lesson, Azure Table storage now has a premium offering as part of Cosmos DB. It's accessed using the Table API. It's still a key/attribute store, but it offers these additional features:

- Global distribution
- Dedicated throughput worldwide
- Single-digit millisecond latencies at the 99th percentile
- Guaranteed high availability, and
- Automatic secondary indexing

These features are not unique to the Table API. They are inherent capabilities of all Cosmos DB data models.

Next, let's look at the document data model. It's very similar to the key/value model, but it provides richer querying capabilities. At the moment, Cosmos DB supports two different options for document databases, the MongoDB API and the SQL API.

MongoDB is currently the most popular document database and it's open source. It stores data in a JSON-like format, without a schema. With the MongoDB API, you can take applications that were written to work with a MongoDB database and point them to a Cosmos DB database instead. In many cases, you only need to change a connection string to make this work.

Cosmos DB's second document database offering is the SQL API. This used to be called the DocumentDB API. The new name is actually kind of confusing because a document database is technically a NoSQL database. So isn't it a contradiction to call it the SQL API? Well, it's called that because it lets you use a SQL-like language to query JSON documents (which is how Cosmos DB stores the data).

Since SQL was designed to work with relational database tables, Microsoft couldn't just use SQL as-is to query JSON documents, so it's not standard SQL. Furthermore, Microsoft incorporated JavaScript's programming model in their query language, so it's even more different from SQL than you would expect. Despite all of this, the SQL API still looks very similar to standard SQL, which makes it easier for people with SQL experience to use than the MongoDB API.

Graph databases are more complex than document databases because they also store information about relationships between entities. For example, two people may be connected to each other on LinkedIn or Facebook. A graph database stores this relationship data separately from the user entities.

Graphs are a great way to model many real-world relationships, but you can still find ways to model them using other types of databases. Where graph databases really shine is with applications that need to *traverse* a graph from one entity to another, with other entities in between, such as a friend of a friend. Graph databases can perform these operations orders of magnitude faster than other types of databases. Some examples of application types that can benefit from graph databases are social networking, content management, geospatial, and product recommendations.

The last data model supported by Cosmos DB is the wide column model. This is the model used by the Apache Cassandra database system. Cosmos DB provides the Cassandra API for

applications that are written to use a Cassandra database.
Yet another unusual feature of Cosmos DB is that you can choose from five different consistency levels. When you build a database system that's distributed across different locations (as is the case with Cosmos DB), you face the problem of how to keep all of the locations in sync.
If data is written in one location, it has to be replicated to all of the other locations as well. But what if a read request comes in at one of the other locations immediately after the data was written at the first location? Since the replication wouldn't have happened yet, the read request wouldn't return the new data. But a read request against the first location *would* return the new data, so that would be inconsistent.
This consistency problem is why relational databases usually have only one location. This makes it hard to scale relational databases and is the biggest reason why NoSQL databases became popular. There are many different ways to handle the consistency problem and Cosmos DB supports five of them.
Strong consistency guarantees that a read operation will return the most recent version of an item. This is the type of consistency that relational databases have. For a distributed database to achieve strong consistency, it has to ensure that each write operation has been propagated to all of the replicas before the operation is considered complete. There's an obvious problem with this approach. It's very slow, especially if the database is distributed over a wide geographic region. So how can Cosmos DB provide strong consistency for a global database? Well, it can't. If you choose strong consistency, then the database can only be in one region.
At the other end of the spectrum is eventual consistency. The only guarantee made by this approach is that if no new writes are made to an item, then eventually all of the replicas will have the same value for that item. This is a very weak guarantee because not only could a request return an old value, but it could return a value older than the one that you retrieved previously. This could happen if your second request connected to a replica that hadn't been updated yet. This is the level of consistency typically offered by NoSQL databases. It has the lowest latency, but the worst consistency.
Cosmos DB offers three other consistency levels that are in between these two extremes. As you add more consistency, both the latency and the cost generally go up.
Consistent Prefix is the same as eventual consistency except that it guarantees that read operations will never see out-of-order writes. That is, reads can still return older values, but never out of order.
Session consistency guarantees consistency for each client session. So a client will never see data older than what it has written during a session. This is relatively easy to provide because the system doesn't have to worry about conflicts between multiple clients. It just needs to keeps things consistent for an individual client. This approach offers the lowest latency reads and writes. It's also by far the most popular consistency level chosen by Cosmos DB customers.
Bounded staleness guarantees that reads may lag behind writes by a limited amount of time. This costs as much as strong consistency, but it allows you to distribute your database across regions and has lower latency. This is the second most popular consistency level chosen by customers.
Believe it or not, Cosmos DB allows you to specify a different consistency level *on each request*, although most customers don't do that. It also provides an SLA on both consistency *and* latency.
Cosmos DB's pricing model is based on Request Units, or RUs. All API requests have a number of RUs assigned to them, based on the resources required to fulfill that request. For example, a complex SQL query would require more RUs than reading a single key/value pair. Writes take more resources to execute than reads do, so they're five times more expensive.
You need to tell Cosmos DB how many RUs *per second* to provision. This throughput capacity is applied to each region associated with your database account. You have to pay for this amount of capacity regardless of how much you actually use, but you can increase or decrease the RUs per second at any time.
The number of RUs required to fulfill a request depends on many factors, including:

- Item size
- Number of properties in an item
- Consistency level
- Number of properties indexed

- Document indexing
- Query complexity, and
- Script usage (i.e. stored procedures and triggers)

To estimate the number of RUs per second you'll need, you can use the request unit calculator. But it would probably be easier to run a few typical operations and see how many RUs they required. You can get that number from either the header in the response to your request or from the Cosmos DB Data Explorer in the Azure portal. Then multiply that by the number of operations you expect to run per second.

If your application exceeds the number of RUs per second that you reserved, then your requests will be rate-limited until the rate drops below the amount you reserved. On the plus side, Microsoft provides an SLA on the throughput level you reserve.

Cosmos DB also provides an SLA on availability. It guarantees 99.99% availability for both single-region and multi-region databases. It also provides a 99.999% *read* availability SLA on multi-region databases.

It can provide such strong HA guarantees because of its highly redundant architecture. Not only is the database replicated *across* regions, but it's also replicated *within* each region.

In the unlikely event of a regional failure, Azure will automatically failover your Cosmos DB databases to another region. You can make this process more efficient if you set a preferred regions list for each region. For example, if the West US region goes down, and your preferred regions list for that region has North Europe as the next region on the list, then that's where Azure will failover to. When the region comes back online, Azure will automatically perform a recovery and bring the database back online in that region.

You can also test how your application will respond to an outage by performing a manual failover.

Cosmos DB automatically backs up your databases too. It takes snapshots every four hours and stores them in geo-redundant Blob storage. Only the last two snapshots are retained. If you delete a database or a container within that database, then the snapshot will be retained for 30 days. If this backup schedule isn't sufficient for your needs, then you can schedule additional backups using the Cosmos DB Data Migration tool.

If you accidentally delete a database or a container, then you have to submit a ticket with Azure Support to restore it from the latest backup. If you have a data corruption issue, then you should delete the corrupted container as soon as possible. Otherwise your backups will be overwritten with corrupted data. Then you can submit a support ticket to get it restored.

Now I'm going to go over a few specific capabilities of the SQL API. Since it used to be the only supported API back when the service was called DocumentDB, it's the most mature.

As I mentioned earlier, one of the great features of Cosmos DB is automatic indexing. However, there may be times when you'll want to customize the indexing. For example, you could reduce your costs by eliminating some of the indexes, or you could make document updates faster by not indexing them immediately.

You can make these changes by setting an indexing policy on a collection. A collection is a group of documents. When you override the default indexing policy, you can do things like exclude specific documents or JSON properties. You can also configure the index update mode. There are three options for that: Consistent, Lazy, and None. The update mode is usually set to Consistent, which means that the index gets updated when a document gets updated. If the update mode is set to Lazy, then the index gets updated when the collection's throughput capacity isn't being maxed out by user requests. In other words, the index is often out-of-date and has *eventual* consistency regardless of the consistency level for the document updates. The final update mode is None, which just means that the collection won't have an index.

You can also make some of your queries more efficient by customizing the data type of an index. For example, suppose you're storing time-stamped IoT data and you often need to run queries on things like the average temperature reading for the previous day. To do this, you need to do a range query, and to make the query efficient, you need to create a range index. This is easy to do by setting a custom indexing policy.

Another type of data that needs special indexing is geospatial data. The SQL API supports the GeoJSON specification. You can use GeoJSON Point to represent the longitude and latitude of a location, such as a building, or GeoJSON LineString to represent two or more points connected by line segments, such as a river, or GeoJSON Polygon to represent a closed area, such as a lake.

If you have geospatial data and you want to, for example, search for all of the houses within a 10-mile radius of a city's center, then you'll need to set a custom indexing policy for geospatial data.
And that's it for Cosmos DB.

From <https://cloudacademy.com/course/designing-an-azure-data-implementation/designing-an-azure-data-implementation-cosmos-db/?context_resource=lp&context_id=3191>

ADI - Conclusion

Saturday, April 16, 2022 3:39 PM

I hope you enjoyed learning about Azure's data services. Let's do a quick review of what you learned.

Azure Storage has 4 different redundancy options. Locally-redundant storage is replicated across racks in the same data center. Zone-redundant storage is replicated across three zones within one region. Geo-redundant storage is replicated across two regions. Read-access geo-redundant storage is the same as geo-redundant storage except that if there's a disaster in your primary region, then you can read your data from the secondary region immediately.

Azure Storage supports five types of data: blobs, files, queues, tables, and disks. Blob storage holds data objects. You can choose from three storage tiers. Hot storage is for data that gets accessed frequently. Cool storage is for data that doesn't get accessed more than once every 30 days and that needs to be retrieved immediately when requested. Archive storage is for data that doesn't get accessed more than once every 180 days and that can take up to 15 hours to access when you do need it.

File storage is SMB-compliant, so you can use it as a file share. Queue storage is for passing messages between applications. Table storage is a very simple and inexpensive NoSQL datastore. Disk storage is for the disks that are attached to virtual machines.

StorSimple is a virtual array that you install at your own site. It moves your infrequently used data to the cloud and lets you access that data seamlessly when you need it.

Azure Purview is an index to all of an organization's data. Each data source has to be manually registered in the catalog.

Azure Synapse Analytics includes both a SQL-based data warehouse and a Spark-based data lake. SQL pools are used primarily for business reporting, and Spark pools are used primarily for data exploration, processing, and analytics.

Two other services that can run Spark jobs are Azure Databricks and HDInsight.

With Azure Data Factory, you can create data processing pipelines. This lets you automate data movement and transformation.

Azure Analysis Services lets you create data models to make sense of existing data. It sits between databases and business intelligence clients, such as Power BI.

Azure Database for MySQL and Azure Database for PostgreSQL are managed services that provide high availability, backups, security, and compliance for MySQL and PostgreSQL.

SQL Server Stretch Database migrates cold table rows to Azure but still lets you query them. This saves you money and helps your backups run faster.

Azure SQL Database is the preferred option for moving from SQL Server to Azure, but it's not 100% compatible with SQL Server, so there may be some reengineering required. It offers three service tiers. Basic is for databases with less than 2 gig of data. Standard can hold up to 1 terabyte and Premium can hold up to 4 terabytes. Premium offers the fastest I/O and support for in-memory processing.

Standard and Premium offer a number of different performance levels, measured in DTUs, which stands for Database Transaction Units. A DTU represents a bundle of compute, storage, and I/O resources.

You also need to choose between the single database model and the elastic pool model. An elastic pool can contain multiple databases that share resources with each other. This model

works well if the usage levels of your various databases are unpredictable.

SQL Database is highly available within a region automatically. To make it highly available across regions, you need to configure active geo-replication. This lets you create up to four read replicas in different regions.

SQL Database takes care of backups automatically, too, but it even protects against regional failures out of the box. Transaction log backups happen about every 5 or 10 minutes. Geo-restore lets you restore a database to another geographic region from where it was backed up.

Azure Table storage is a key/attribute store. It's schemaless and it automatically creates a primary index. Azure Redis Cache is a simple key/value store. It runs in memory, which is why it's used as a cache. Azure Data Lake Storage is a NoSQL repository for all kinds of data. Azure Search creates an index of text data so your users can run searches. Time Series Insights collects time-stamped data, such as data from IoT devices, and lets you run queries on it.

Cosmos DB is globally distributed and it provides SLAs for latency, throughput, consistency, and availability.

It has APIs to support various data models, including Table storage, MongoDB, SQL, Graph, and Cassandra.

It offers 5 data consistency levels. Strong consistency guarantees completely consistent reads, but it's only available for single-region databases. Bounded staleness guarantees that reads may lag behind writes by a limited amount of time. Session consistency guarantees consistency for each client session. Consistent Prefix guarantees that read operations will never see out-of-order writes. Eventual consistency guarantees that if no new writes are made to an item, then eventually all the replicas will have the same value for that item.

You need to tell Cosmos DB how many RUs per second to provision. This throughput capacity is applied to each region associated with your database account.

Cosmos DB guarantees 99.99% availability for both single-region and multi-region databases. It also provides a 99.999% read availability SLA on multi-region databases. To make the automatic failover process more efficient, set a preferred regions list for each region.

Cosmos DB automatically takes snapshots of your databases every four hours. Only the last two snapshots are retained.

You can change the indexing behavior on a Cosmos DB collection by configuring a custom indexing policy. The three options for the index update mode are Consistent, Lazy, and None.

Now you know how to identify the most appropriate Azure services for various data-related needs; design an Azure SQL Database implementation for scalability, availability, and disaster recovery; and design an Azure Cosmos DB implementation for cost, performance, consistency, availability, and business continuity.

To learn more about Azure's data services, you can read Microsoft's documentation. Also watch for new Microsoft Azure courses on Cloud Academy, because we're always publishing new courses. Please give this course a rating, and if you have any questions or comments, please let us know. Thanks and keep on learning!

From <https://cloudacademy.com/course/designing-an-azure-data-implementation/designing-an-azure-data-implementation-conclusion/?context_resource=lp&context_id=3191>

Saturday, April 16, 2022 4:44 PM

Data Warehouses and Data Lakes

Data Warehouse	Data Lake
• Structured data (relational tables) • Collect data from many databases • Designed for reporting • Query using SQL	• Unstructured data (documents, etc.) and structured data • Collect all kinds of data in one place • Designed for big data analytics • Query using Apache Spark

Screen clipping taken: 4/16/2022 4:44 PM

Azure Synapse Analytics

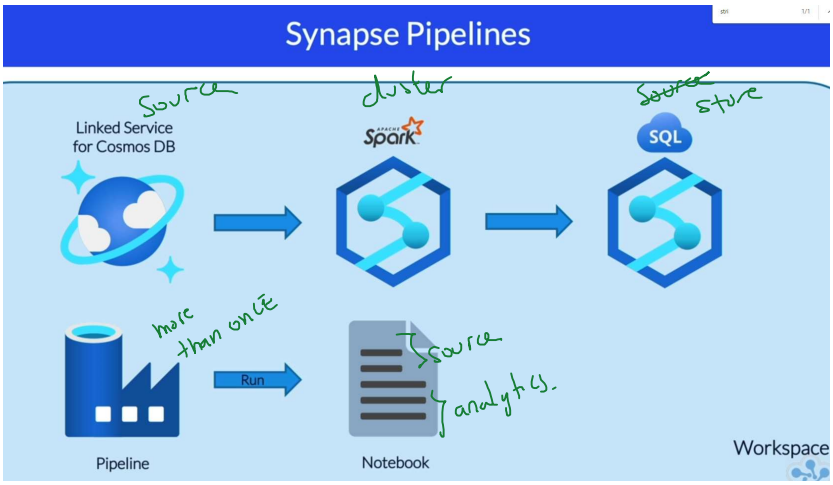
Dedicated SQL Pool	Spark Pool
• Lets you run SQL queries on structured, relational tables	• Lets you use Spark to query both structured and unstructured data

Screen clipping taken: 4/16/2022 4:45 PM

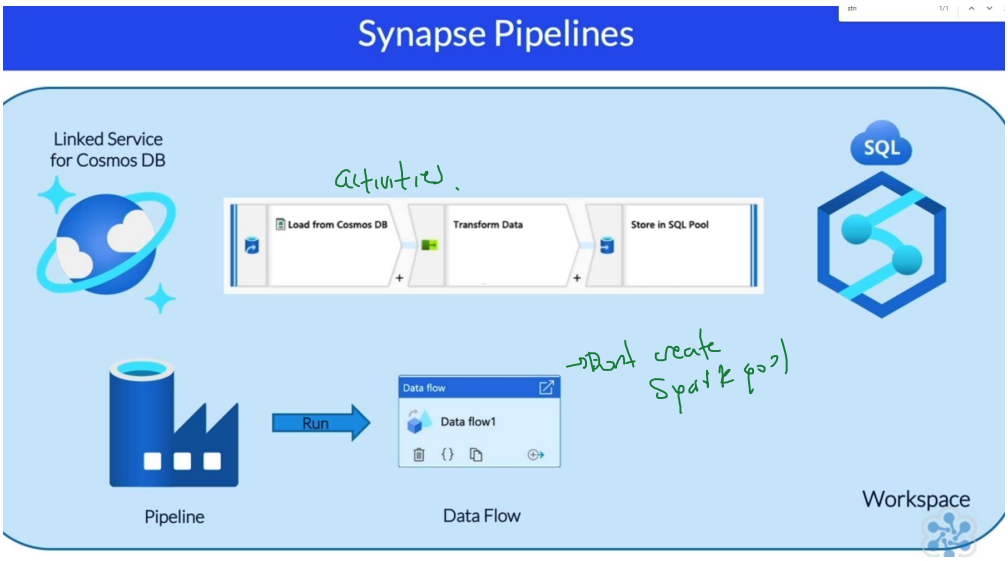
Synapse Pipelines

- Stripped-down version of Azure Data Factory
- Lets you create data processing pipelines
- For example, a pipeline could:
  - Copy data from Azure Cosmos DB to a Spark pool
  - Run a Spark job that creates statistics about the data
  - Store the results in a SQL pool

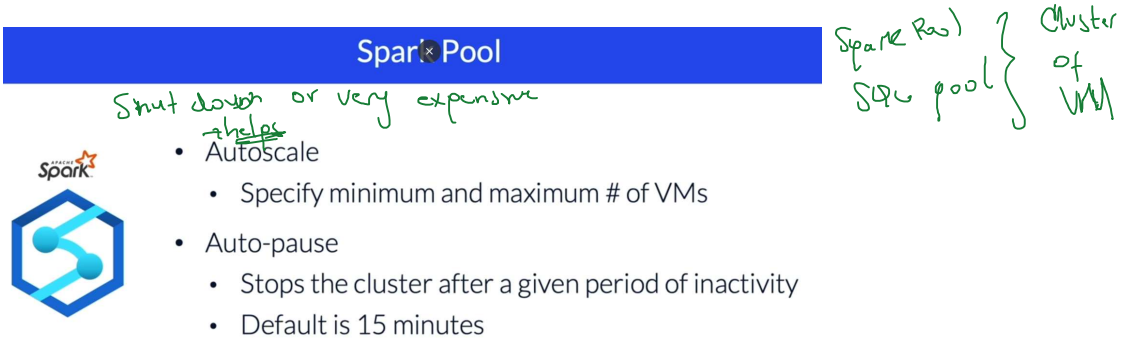
Screen clipping taken: 4/16/2022 4:47 PM



Screen clipping taken: 4/16/2022 4:51 PM



Screen clipping taken: 4/16/2022 4:53 PM



Screen clipping taken: 4/16/2022 4:55 PM

Dedicated SQL Pool

- Formerly known as Azure SQL Data Warehouse
- Provides a compute cluster and storage
- DWUs (Data Warehousing Units) set the amount of CPU, memory, and I/O in the compute cluster
- You can only increase or decrease the # of DWUs manually because there's no autoscaling feature
- Storage space is provided by Azure Storage, so it scales independently from the compute cluster
- You can manually pause it when it's not in use
- When it's paused, you won't pay for the compute cluster, but you'll still pay for the storage

Screen clipping taken: 4/16/2022 4:57 PM

Serverless SQL Pool

- Don't have their own storage
- Don't have access to data in dedicated SQL pools
- Can query data in other places, such as
  - CSV, Parquet, or JSON files in Azure Storage
  - Azure Open Datasets
  - External tables in Azure Storage created by Spark pools
- You don't have to pay for the compute resources in them
- You only have to pay for the amount of data processed by your queries
- When you create an Azure Synapse Workspace, it will automatically create a serverless SQL pool

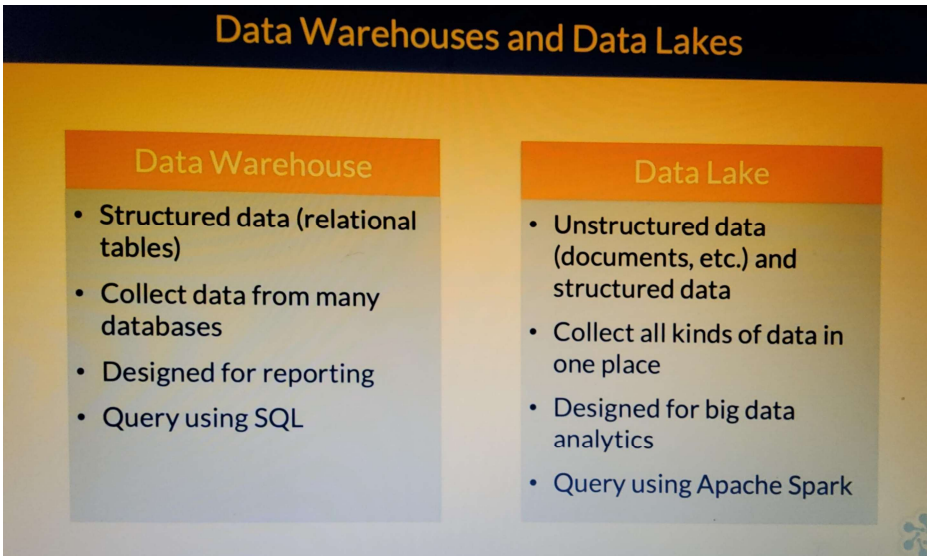
Screen clipping taken: 4/16/2022 4:59 PM

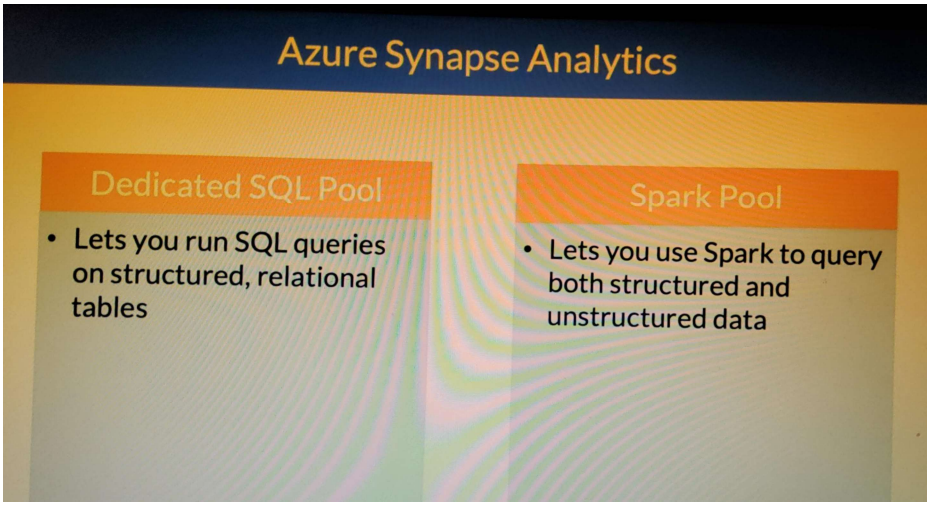
Screen clipping taken: 4/16/2022 4:56 PM

Intro synapses

Thursday, 15 December 2022 1:52 PM

Intro to Synapse.





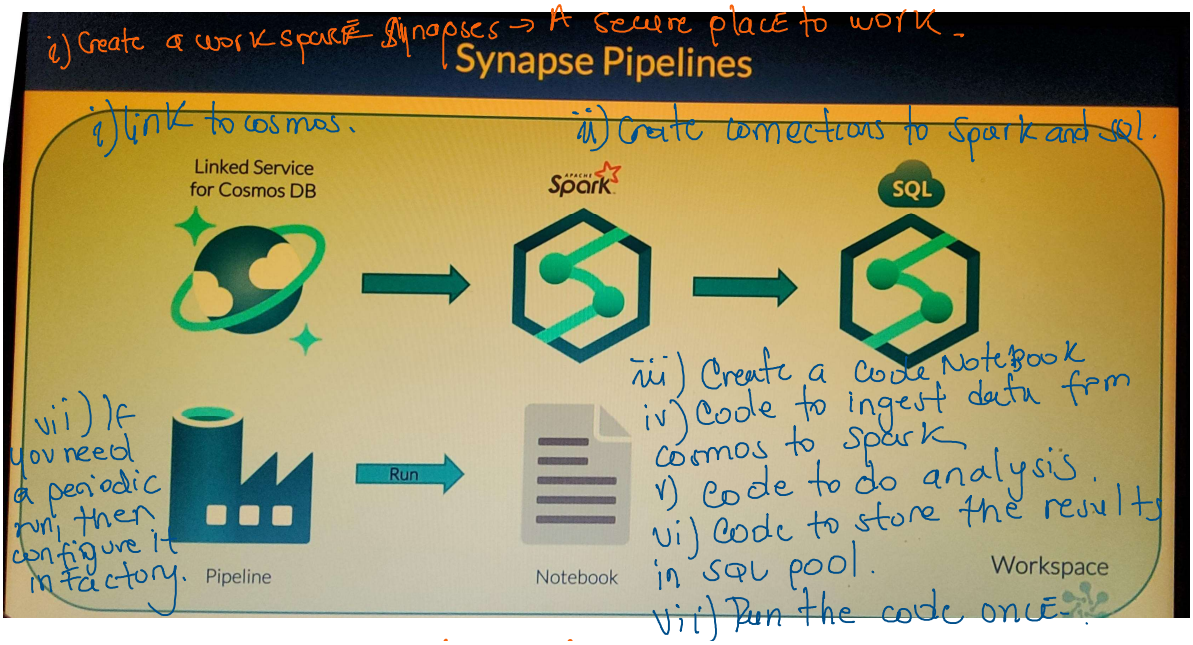
Synapse offers solutions to work with both worlds: unstructured and structured data, in only one service.

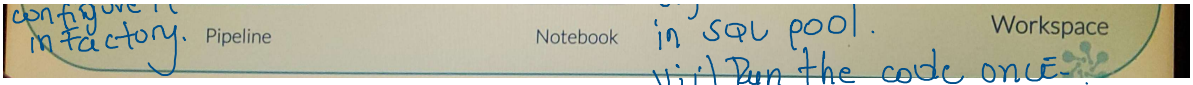
→ Synapse also provides a sophisticated tool for getting data from other Azure services and transforming it.

→ Azure Synapse Pipeline is a stripped down version of Data Factory (lets you create data pipelines) → simplificada

→ 2 approches → with code
→ without code.

i) Code option.



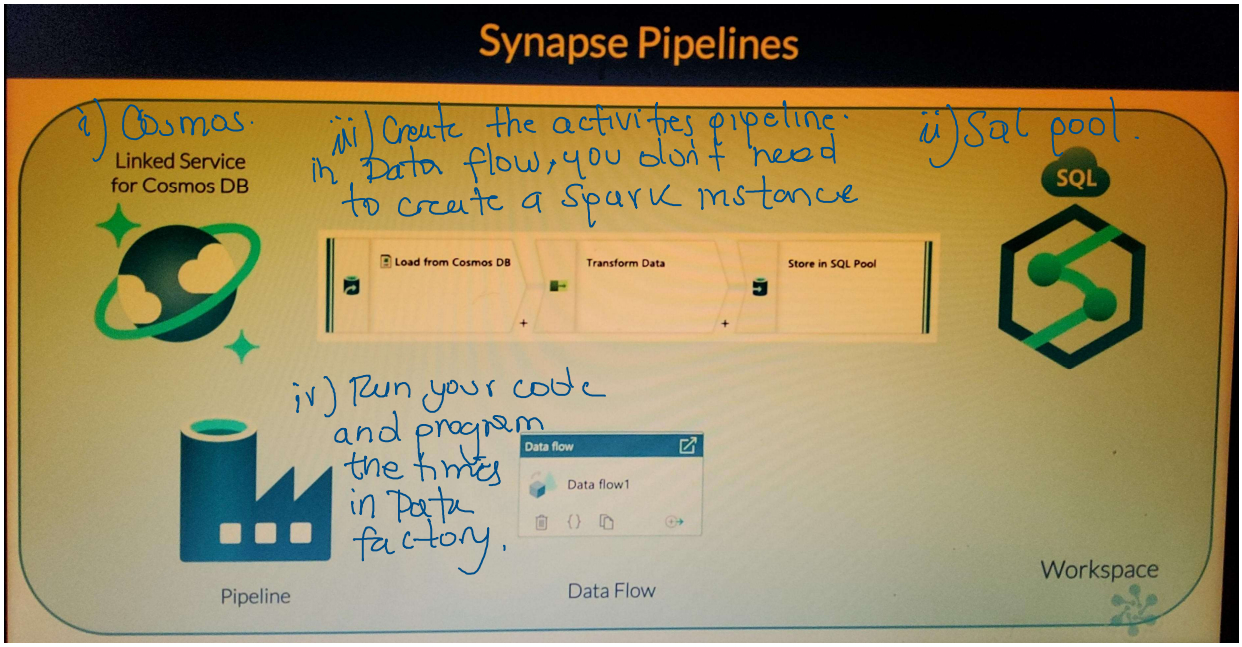


configure it in factory.

in sql pool.

vii) Run the code once.

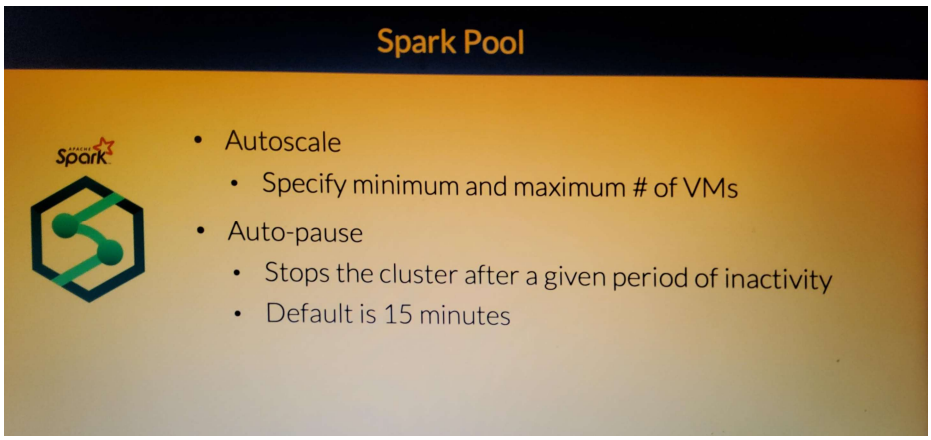
ii) Without code option



Spark and SQL pools → Both are clusters of VM
→ Cost out a lot of money. Be careful.
→ Different underlying architecture.

Mechanisms given for Microsoft to manage Spark and SQL pool clusters

Spark



SQL

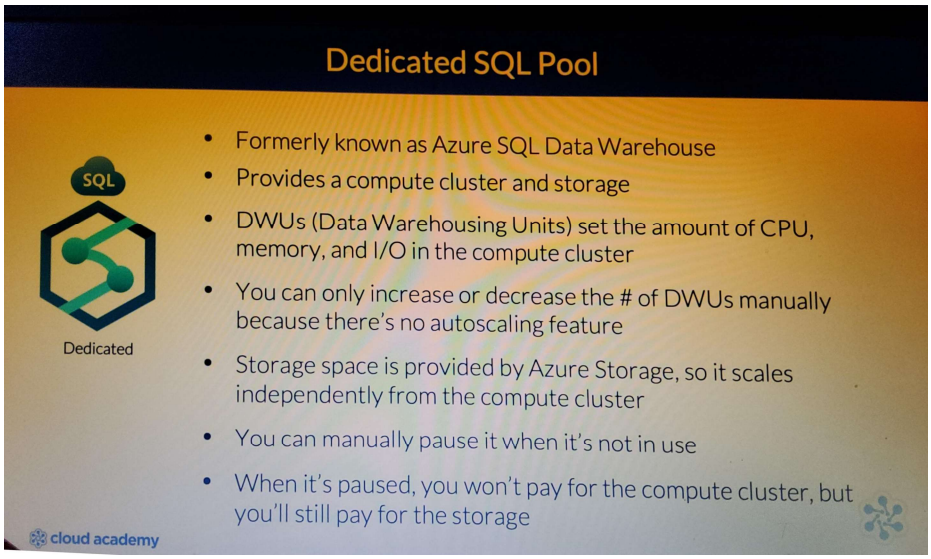
SQL has dedicated and serverless pools.

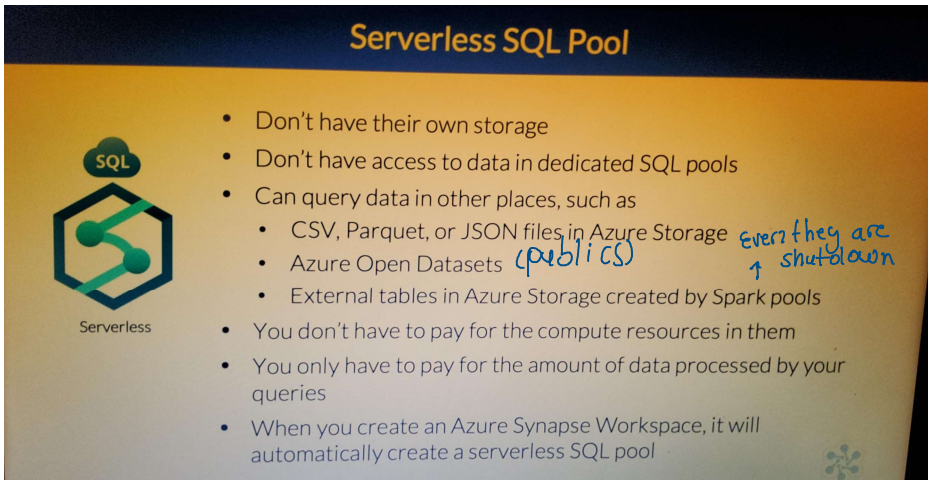
Synapse

→ SQL has dedicated and serverless pools.

Dedicated Pools

→ New name for Azure SQL Data Warehouse.





Summary

Review

- Data warehouse: Dedicated SQL pool → To run SQL queries on structured data.
- Data lake: Spark pool → to run Spark in structured and unstructured data.
- Both types of pools are clusters of virtual machines

Dedicated SQL Pool

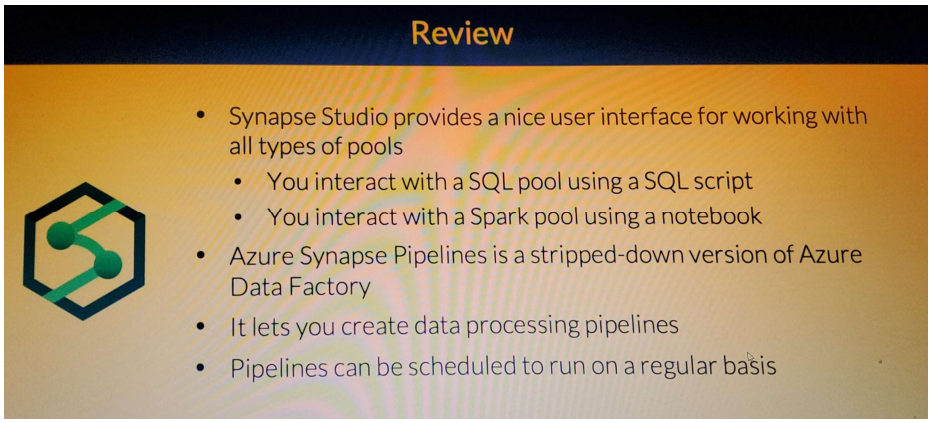
Dedicated

- DWUs (Data Warehousing Units) set the amount of CPU, memory, and I/O in the compute cluster
- You can only increase or decrease the # of DWUs manually
- Storage space is provided by Azure Storage, so it scales independently from the compute cluster
- You can manually pause it when it's not in use
- When it's paused, you won't pay for the compute cluster, but you'll still pay for the storage

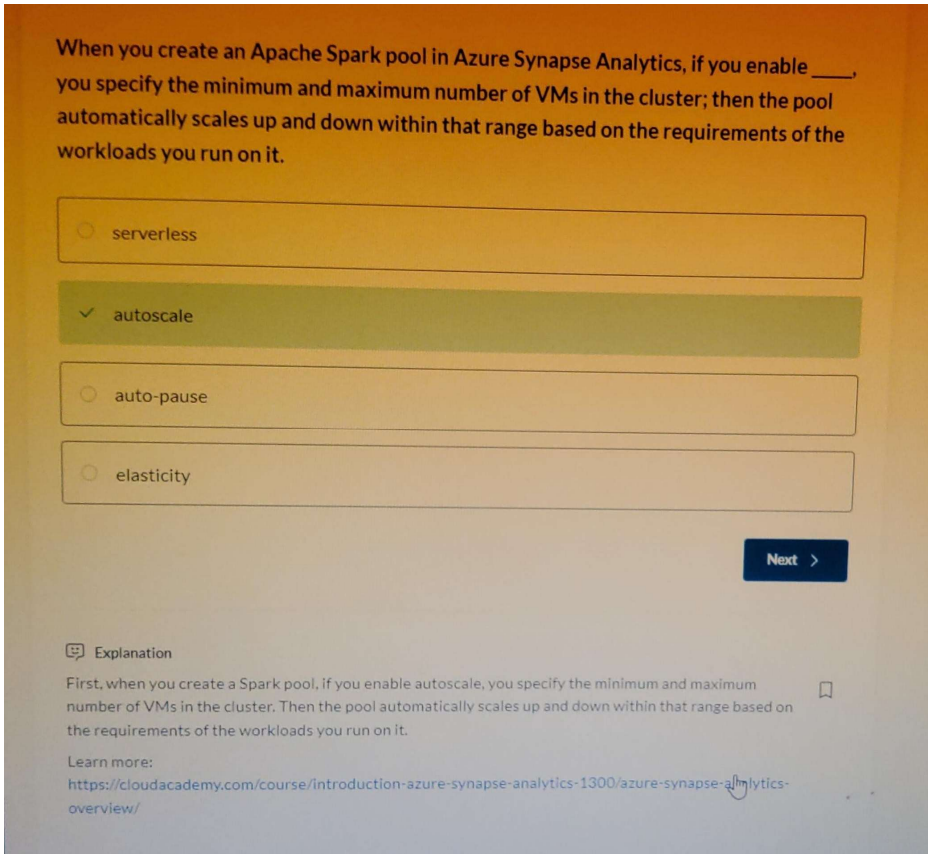
Serverless SQL Pool

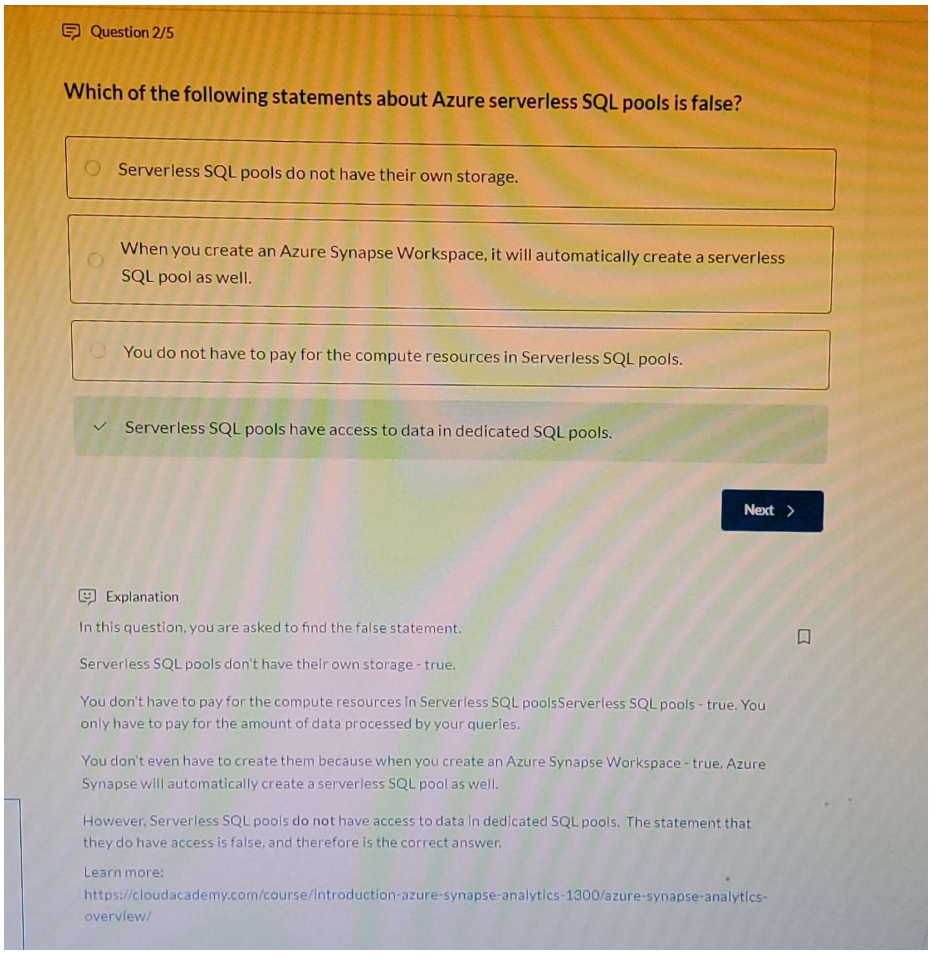
Serverless

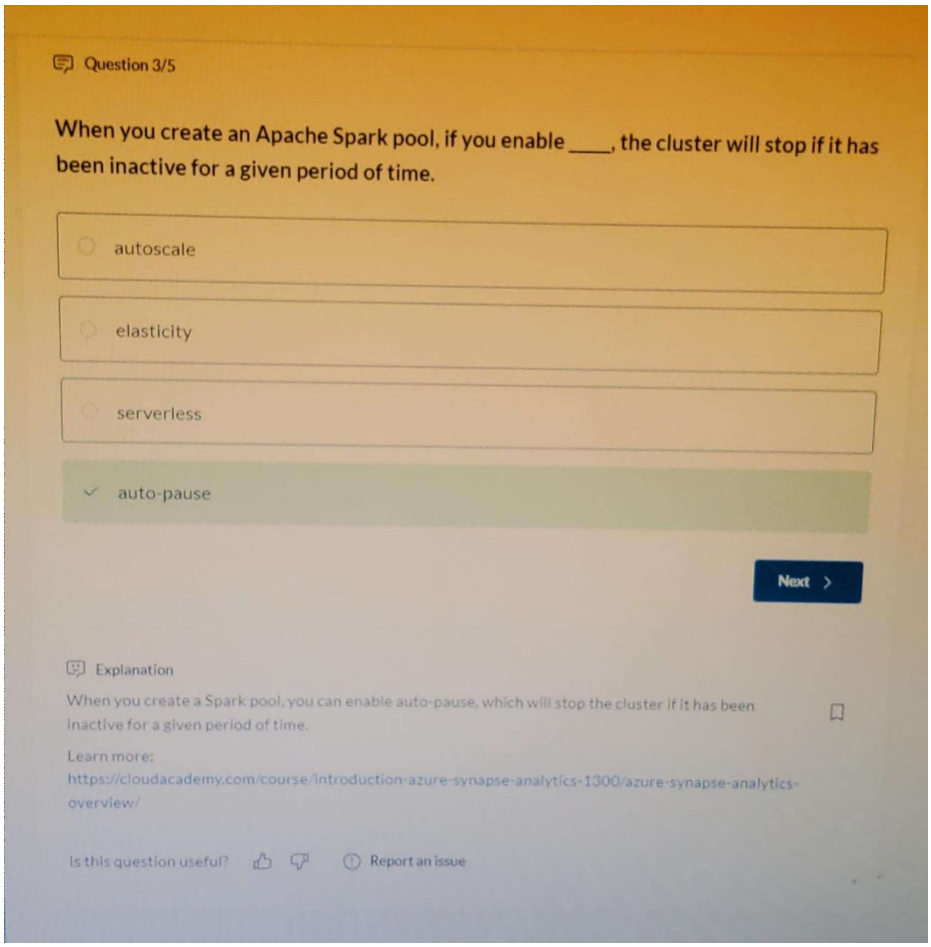
- Don't have their own storage
- Don't have access to data in dedicated SQL pools
- Can query data in other places, such as
  - CSV, Parquet, or JSON files in Azure Storage
  - Azure Open Datasets
  - External tables in Azure Storage created by Spark pools
- You only have to pay for the amount of data processed by your queries
- When you create an Azure Synapse Workspace, it will automatically create a serverless SQL pool

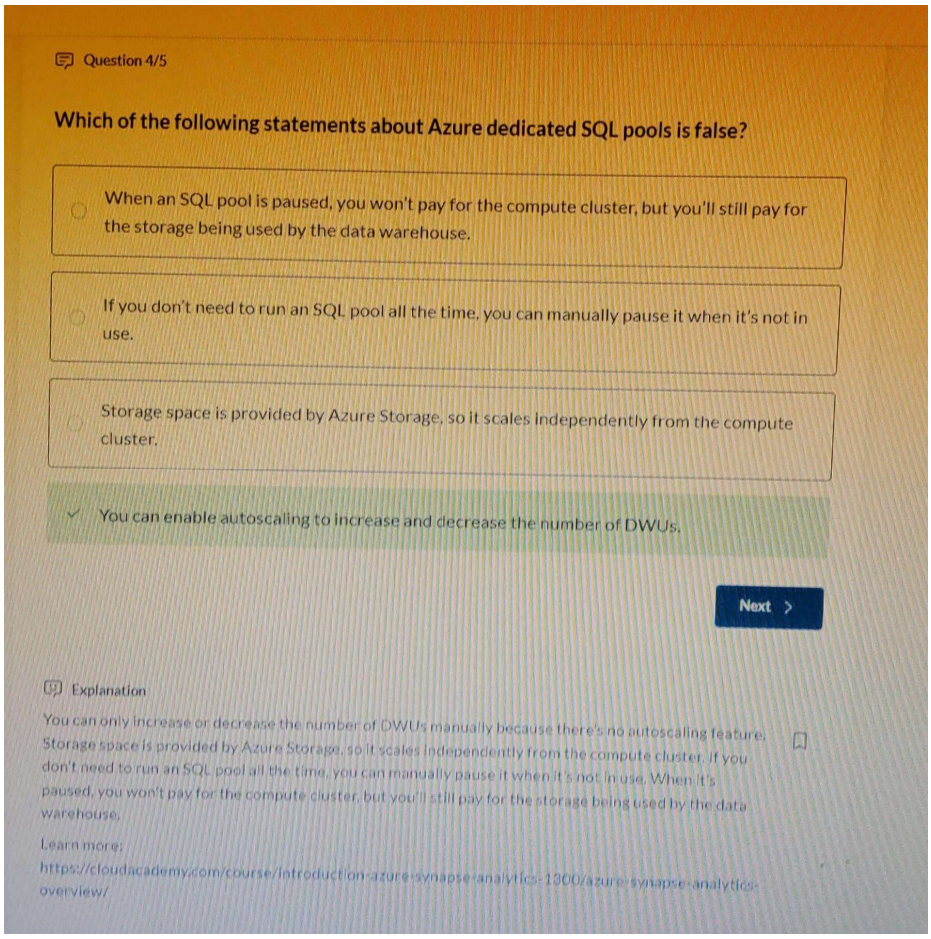

Review
Synapse Studio provides a nice user interface for working with all types of pools
You interact with a SQL pool using a SQL script
You interact with a Spark pool using a notebook
Azure Synapse Pipelines is a stripped-down version of Azure Data Factory
It lets you create data processing pipelines
Pipelines can be scheduled to run on a regular basis

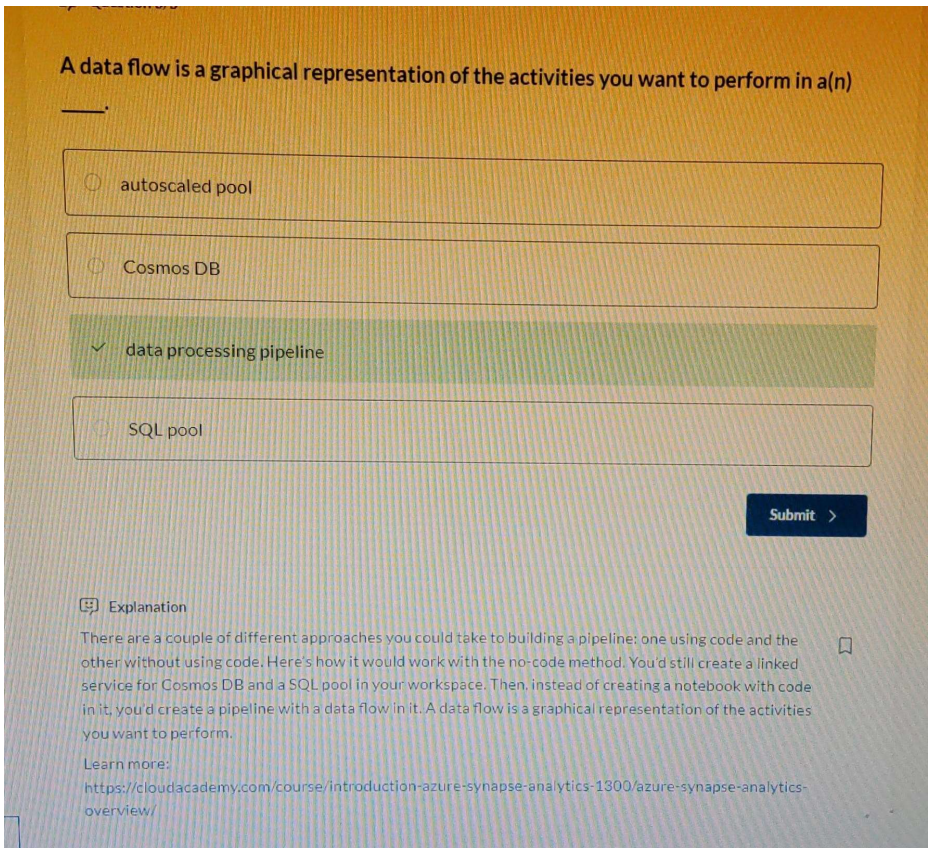
Check


When you create an Apache Spark pool in Azure Synapse Analytics, if you enable ____, you specify the minimum and maximum number of VMs in the cluster; then the pool automatically scales up and down within that range based on the requirements of the workloads you run on it.
serverless
autoscale
auto-pause
elasticity
Next
Explanation
First, when you create a Spark pool, if you enable autoscale, you specify the minimum and maximum number of VMs in the cluster. Then the pool automatically scales up and down within that range based on the requirements of the workloads you run on it.
Learn more:
https://cloudacademy.com/course/introduction-azure-synapse-analytics-overview/


Question 2/5
Which of the following statements about Azure serverless SQL pools is false?
Serverless SQL pools do not have their own storage.
When you create an Azure Synapse Workspace, it will automatically create a serverless SQL pool as well.
You do not have to pay for the compute resources in Serverless SQL pools.
Serverless SQL pools have access to data in dedicated SQL pools.
Next
Explanation
In this question, you are asked to find the false statement.
Serverless SQL pools don't have their own storage - true.
You don't have to pay for the compute resources in Serverless SQL poolsServerless SQL pools - true. You only have to pay for the amount of data processed by your queries.
You don't even have to create them because when you create an Azure Synapse Workspace - true. Azure Synapse will automatically create a serverless SQL pool as well.
However, Serverless SQL pools do not have access to data in dedicated SQL pools. The statement that they do have access is false, and therefore is the correct answer.
Learn more:
https://cloudacademy.com/course/introduction-azure-synapse-analytics-1300/azure-synapse-analytics-overview/


Question 3/5
When you create an Apache Spark pool, if you enable ____, the cluster will stop if it has been inactive for a given period of time.
autoscale
elasticity
serverless
auto-pause
Next
Explanation
When you create a Spark pool, you can enable auto-pause, which will stop the cluster if it has been inactive for a given period of time.
Learn more:
https://cloudacademy.com/course/introduction-azure-synapse-analytics-1300/azure-synapse-analytics-overview/
Is this question useful?
Report an issue

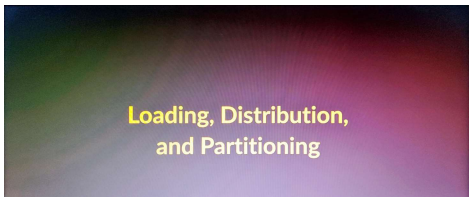

Question 4/5
Which of the following statements about Azure dedicated SQL pools is false?
When an SQL pool is paused, you won't pay for the compute cluster, but you'll still pay for the storage being used by the data warehouse.
If you don't need to run an SQL pool all the time, you can manually pause it when it's not in use.
Storage space is provided by Azure Storage, so it scales independently from the compute cluster.
You can enable autoscaling to increase and decrease the number of DWUs.
Next
Explanation
You can only increase or decrease the number of DWUs manually because there's no autoscaling feature. Storage space is provided by Azure Storage, so it scales independently from the compute cluster. If you don't need to run an SQL pool all the time, you can manually pause it when it's not in use. When it's paused, you won't pay for the compute cluster, but you'll still pay for the storage being used by the data warehouse.
Learn more:
https://cloudacademy.com/course/introduction-azure-synapse-analytics-1300/azure-synapse-analytics-overview/

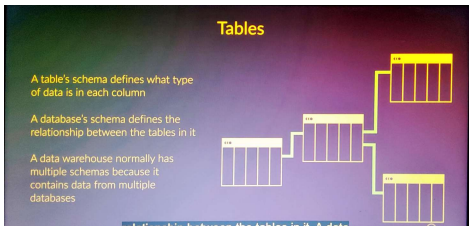

A data flow is a graphical representation of the activities you want to perform in a(n) ____.
autoscaled pool
Cosmos DB
data processing pipeline
SQL pool
Submit
Explanation
There are a couple of different approaches you could take to building a pipeline: one using code and the other without using code. Here's how it would work with the no-code method. You'd still create a linked service for Cosmos DB and a SQL pool in your workspace. Then, instead of creating a notebook with code in it, you'd create a pipeline with a data flow in it. A data flow is a graphical representation of the activities you want to perform.
Learn more:
https://cloudacademy.com/course/introduction-azure-synapse-analytics-1300/azure-synapse-analytics-overview/

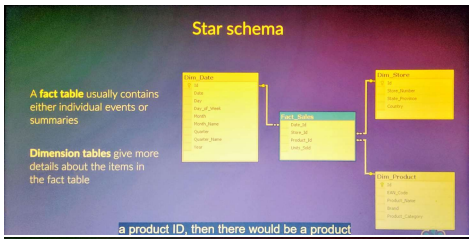
Optimizing dedicated sql pools in A Synapse Analytics

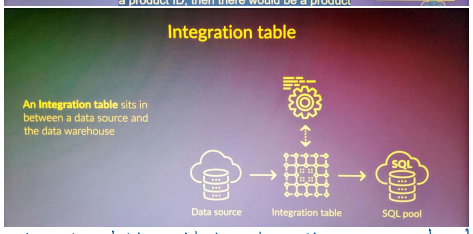
Wednesday, January 4, 2023 7:53 PM

CourseOptimizing dedicaredsql poolsin A sYNAPSE Analytics



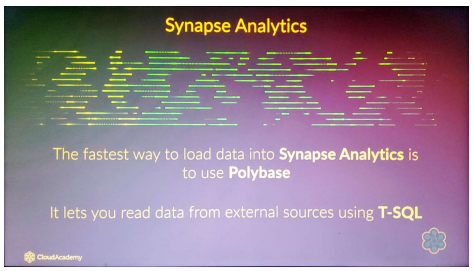




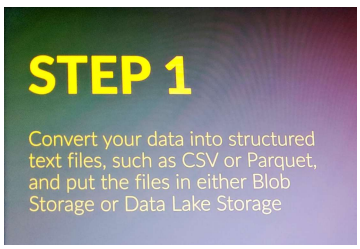


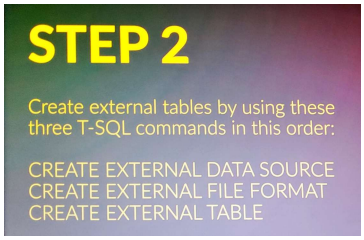
Integration table → it is not part of a start schema.

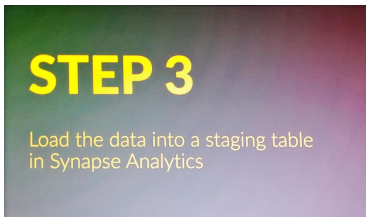
→ A common practice when loading data into a dedicated SQL pool is to first load the data into a staging table, perform some transformations on that data, and then load it into the SQL pool.



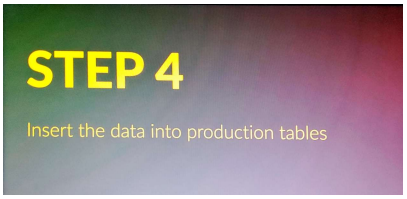
↳ Steps:

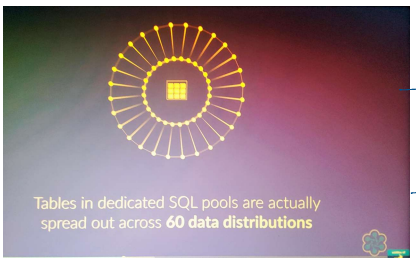






→ this is a best practice, so you can deal with data loading issues without affecting production tables.





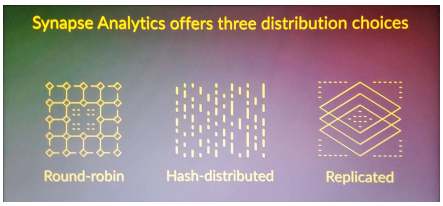
esta es la razón

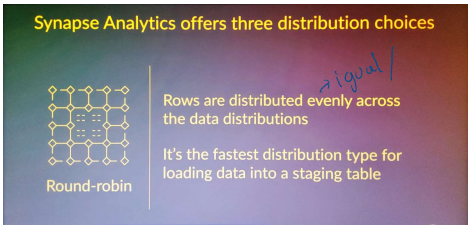
→ this is why queries are so fast on this service, they're massively parallelized.

engendra

→ when you run a query, it spawns 60 queries that each run on one data distribution.

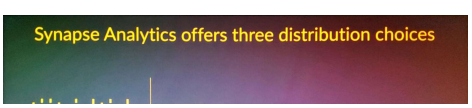
→ to make this work efficiently, you have to decide how the data will be distributed. (Known as sharding), 3 options





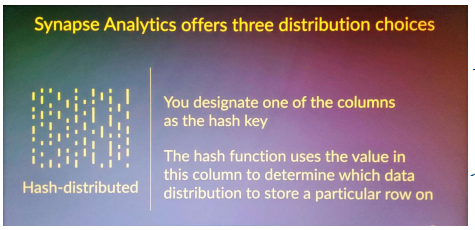
igual

→ the simplest
No optimize.
It doesn't perform any optimization.



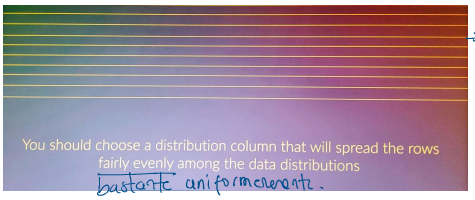
→ A bit more complicated.

→ As long as you choose a hash

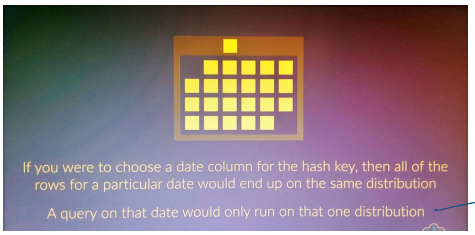


→ A bit more complicated.

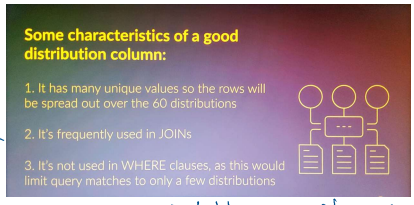
→ As long as you choose a hash key that is appropiate for the most commonly run queries on this table, then query performance will be much better than it would be with a round-robin table.



→ If too many of the rows are on the same data distribution, the it will be a hot spot that reduces the advantages of Synapse Analytics' massively parallel architecture.

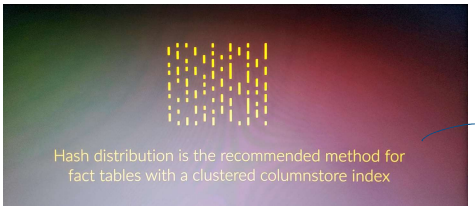


→ which would make the query take much longer than if it were to run across all 60 distributions in pacallel.

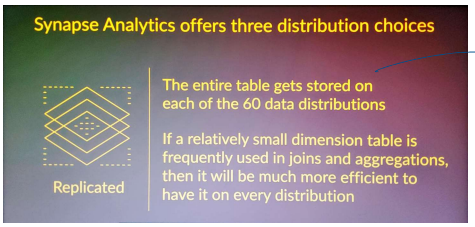


→ If two fact tables are often joined together, then distribute both of the tables on the same join column. that way, rows from the 2 tables that have the same value in the join column will be stored on the same distribution, so they can be joined together easily.

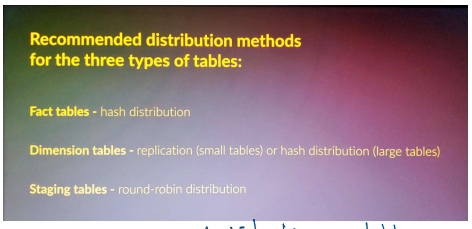
→ If you don't have frequent joins, then choose a column that's often in group by clauses.



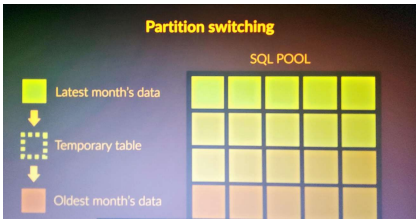
→ this is the default.



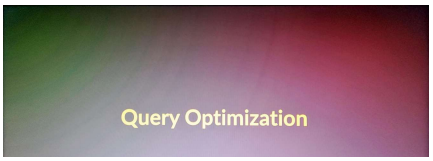
→ It's simpler.



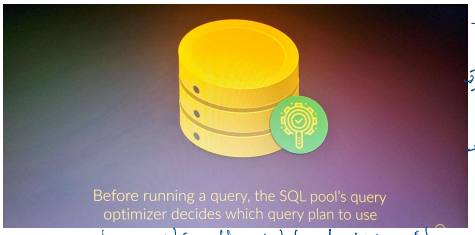
→ In addition to distributing a table, you can also partition it by date range. (each month in separated partition), the benefit of doing this is that you could use something called partition switching



→ you can load the latest month's data into a temporary table, and then in the production table, replace the old partition with the new one.

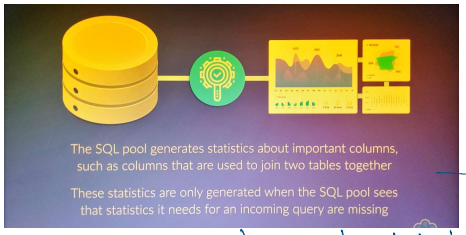


→ to reduce both time queries take and the resources they consume.

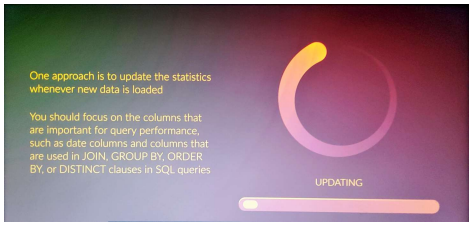


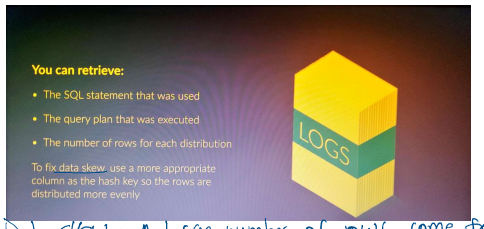
→ for ex, it estimates the number of rows to return.
→ If it is a small amount, only a plan is used.
→ If it is larger, it uses a different query plan.

In order to make the right decision, it needs to know your data pretty well. To do that, SQL generates statistics automatically, by default.



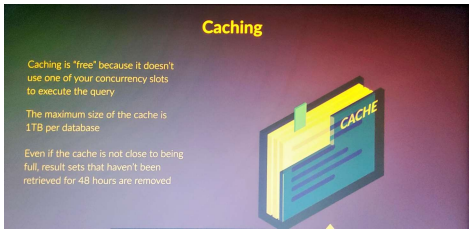
→ If the statistics aren't already there, the query will be slower the first time, it's run because the statistics have to be generated.
→ this is why it's important to generate statistics ahead of time, if possible
→ also, even if statistics have already been created they will become out-of-date. as new data gets added.

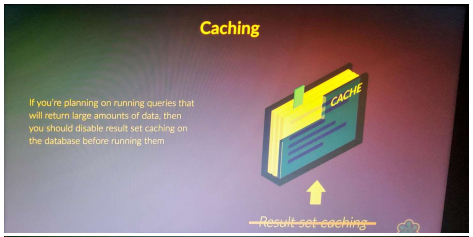


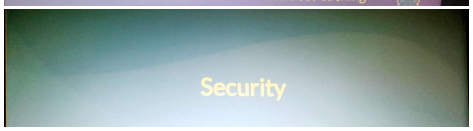


~~Data skew~~ = A large number of rows come from one distribution (hotspot)
↳ Sesgo de datos.

Caching
One feature that dramatically speeds up queries is caching.
→ If you enable result set caching, then the results from all queries (with a few exceptions) get cached.
→ If you run again the same query, and the data hasn't changed in the meantime, then the results are retrieved from the cache instead of from executing the query.







Synapse offers a wide variety options for security for dedicated SQL pools including:
i) → Data Discovery & Classification.
ii) → Dynamic data Masking
iii) → Vulnerability Assesment
iv) → Advanced threat Protection
v) → transparent Data Encryption.

i) → scans your database looking for sensitive data, such as names, addresses, and credit card numbers. It then gives you a list of recommendations for how these columns should be classified. such as "Confidential" or "Highly Confidential"
→ If you accept the recommendations then those columns will be labeled with those classifications.
→ You can label them manually.
→ After labelling, you can use the database auditing feature, to monitor access to the sensitive data.
ii) → will obscure some of the info in a particular column when

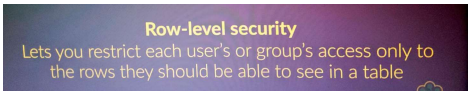
→ After labelling, you can use the [illegible] access to the sensitive data.

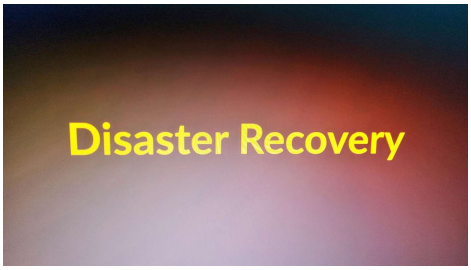
ii) → will obscure some of the info in a particular column when it is retrieved in a query.

iii) → scans your database looking for potential security issues, such as loose permissions and dangerous firewall settings. then you can go through the list of issues and decide whether or not they truly are issues that need to be addressed.
→ If you want to address an issue, there will be a recommendation to remediate, in many cases there will be a remediation script you can run.

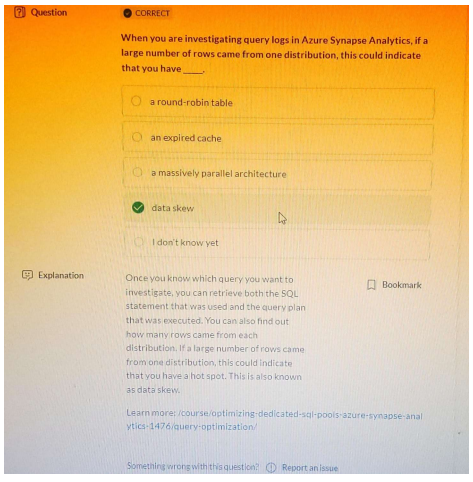
iv) → look for unusual attempts to access or exploit databases.

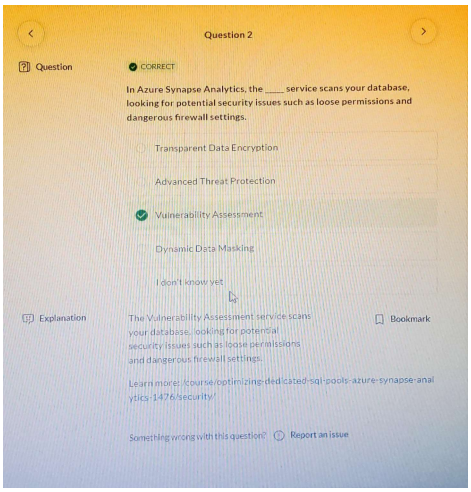
v) → is used to encrypt the entire database, including the log files, and the backups. If a hacker gets a copy of the data, they don't be able to read any of the data in it

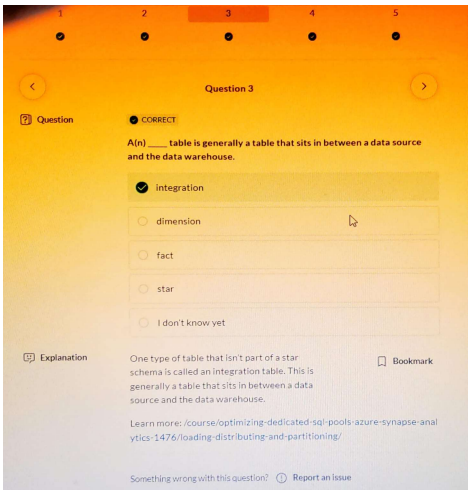


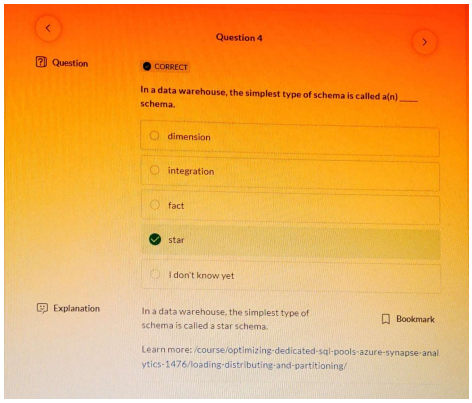


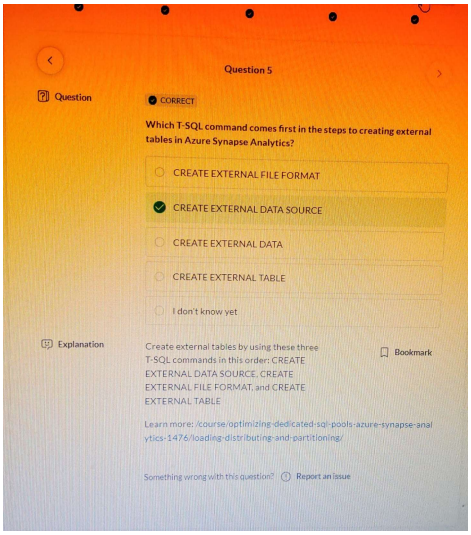
→ If you need to restore your data warehouse to a previous state, you can simply choose which of the automatic restore points you want to go back to.
or you can create an user defined restore point, in addition, to the automatic restore points.
→ Both the automatic and user-defined restore points are stored in the primary region where your data warehouse runs, if it is down, you won't be able to access any of those restore points.
→ to ensure that you will still be able to bring up your data warehouse in another region, Azure makes a geo-redundant backup once a day to a paired data center
→ If a disaster strikes, you can use that backup to restore a copy of your data warehouse to any region that supports Synapse Analytics.








Question 4
Question
CORRECT
In a data warehouse, the simplest type of schema is called a(n) ____ schema.
dimension
integration
fact
star
I don't know yet
Explanation
In a data warehouse, the simplest type of schema is called a star schema.
Bookmark
Learn more: /course/optimizing-dedicated-sql-pools-azure-synapse-analytics-1476/loading-distributing-and-partitioning/


Question 5
Question
CORRECT
Which T-SQL command comes first in the steps to creating external tables in Azure Synapse Analytics?
CREATE EXTERNAL FILE FORMAT
CREATE EXTERNAL DATA SOURCE
CREATE EXTERNAL DATA
CREATE EXTERNAL TABLE
I don't know yet
Explanation
Create external tables by using these three T-SQL commands in this order: CREATE EXTERNAL DATA SOURCE, CREATE EXTERNAL FILE FORMAT, and CREATE EXTERNAL TABLE
Bookmark
Learn more: /course/optimizing-dedicated-sql-pools-azure-synapse-analytics-1476/loading-distributing-and-partitioning/
Something wrong with this question?
Report an issue