Contents

[**Module 2 - Working with Data Storage** 1](#_Toc39350463)

[**Module 5 - Working with Relational Data Stores in the Cloud** 11](#_Toc39350464)

**Module 2 - Working with Data Storage**

This module talks about variety of ways to store data in Azure.

**The Benefits of using Azure to Store Data**

The Azure data storage options are cloud-based, secure, and scalable, reliable and durable.  
Imp Benefits of Azure Storage:

1. **Automated Backup and Recovery:** Mitigates risk of data loss in case of unforeseen failure or interruption.
2. **Global replication:** Data copies are made to protect against planned and unplanned events. We choose to replicate data at multiple locations across globe.
3. **Support performing Analytics** on data consumption.
4. **Encryption Capabilities:** Data is encrypted to make it secure. We can have tight control over who can access data.
5. **Multiple Data types**: Azure can store almost any kind of data. Ex: Video, Text, binary files like VHD, relational, NoSQL.
6. **Data storage on Virtual disks:** Azure can store up to 8Tb of data on its virtual disks. Useful for storing videos and simulations
7. **Storage Tiers:** Storage tiers allow prioritizing access to data based on the frequency of usage.

**Comparing Azure to On-Premises Storage**

On Premises = storage and maintenance of data on local hardware and servers.

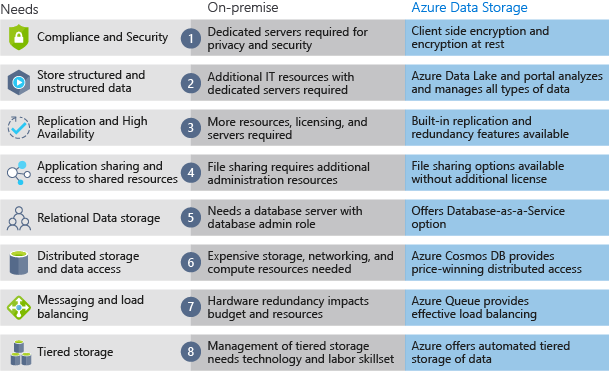
Factors to consider while comparing Azure to On-premises:

**Cost Effectiveness:** An on-prem storage solution needs dedicated hardware. This can be significant up-front expense (or capital cost). Changing requirement may need changing hardware. Also we need to buy hardware in a way to support handling of Peak Demand (leaving under-utilized during off-peak times). Azure provides Pay-as-You-Go pricing model. We are charged only for services that we use. (Expenses are treated as operating expenses instead of upfront cost). Azure is scalable (allows us to scale up/out as demand dictates and scale back when demand is low.

**Reliability:** On-premises storage requires data backup, load balancing, and disaster recovery strategies which can be challenging and expensive. Azure data storage provides data backup, load balancing, disaster recovery, and data replication as services to ensure data safety and high availability.

**Storage types:** When an application demands different types of storage, an on-prem approach requires numerous servers and administrative tools for each storage type. Azure data storage provides a variety of different storage options including distributed access and tiered storage. We can integrate a combination of storage technologies providing the best storage choice for each part of your solution.

**Agility:** on premises, changing services or technology would mean provisioning and deploying new servers and infrastructure, which can be time consuming and expensive. Azure data storage gives you the flexibility to create new services in minutes.



**Summary:**

Azure provides the following features:

* Storage of both structured and unstructured data
* High security that supports global compliance standards
* Load balancing, high availability, and redundancy capabilities
* The ability to store large volumes of data directly to the browser using features such as Azure Blob storage

These capabilities of Azure data storage make it an ideal platform for hosting any large global application or portal.

**Introducing Azure Storage:**

Most organizations have diverse requirements for their cloud-hosted data.   
Ex: storing data in a specific region, or needing separate billing for different data categories. Azure storage accounts let you formalize these types of policies and apply them to your Azure data.

**Storage Accounts:**

Organizations can have multiple storage accounts to let them implement different sets of requirements. Ex: one storage account for the private business data and one for the consumer-facing files. Policy factors that are controlled by storage account which helps decide how many accounts we need.

**What is a Storage Account?**

* Azure provides multiple ways to store data.   
  Ex:  
  Database Options: Azure SQL Server, Azure Cosmos DB, and Azure Table Storage.   
  Store and send messages: Azure Queues and Event Hubs.  
  Store Loose files: Azure Files and Azure Blobs.

Azure selected four of the data services (Azure Blobs, Azure Files, Azure Queues, and Azure Tables) and placed them together under the name Azure Storage.

These four were given special treatment because they are all primitive, cloud-based storage services and are often used together in the same application.

* A storage account is a container that groups a set of Azure Storage services together. Only data services from Azure Storage can be included in a storage account (Azure Blobs, Azure Files, Azure Queues, and Azure Tables).
* Combining data services into a storage account lets us manage them as a group. The settings you specify when you create the account, or any that you change after creation, are applied to everything in the account.
* Deleting the storage account deletes all of the data stored inside it.
* A storage account is an Azure resource and is included in a resource group. Azure subscription can contain multiple resource groups, where each group contains one or more storage accounts.
* Other Azure data services like Azure SQL and Cosmos DB are managed as independent Azure resources and cannot be included in a storage account.

**How many Storage accounts do we need?**

* A storage account represents a collection of settings like location, replication strategy, and subscription owner. We need one storage account for every group of settings that we want to apply to our data.   
  One difference is enough to require separate storage accounts.
* The number of storage accounts we need is typically determined by our data diversity, cost sensitivity, and tolerance for management overhead.
* Data Diversity: Orgs generate data that differs in where it is consumed (specific region or country the data needs to be stored), how sensitive it is (proprietary or public), which group pays the bills, etc. Diversity along any of these vectors can lead to multiple storage accounts.
* Cost Sensitivity: A storage account by itself has no financial cost; however, the settings we choose for the account influence the cost of services in the account.   
  Ex: Geo-redundant storage costs more than locally-redundant storage. Premium performance and the Hot access tier increase the cost of blobs.   
  We can use multiple storage accounts to reduce costs. Ex: We could place our critical data into a storage account with geo-redundant storage and put our non-critical data in a different storage account with locally-redundant storage.
* Management Overhead: Each storage account requires some time and attention from an administrator to create and maintain. It also increases complexity for anyone who adds data to your cloud storage; everyone in this role needs to understand the purpose of each storage account so they add new data to the correct account.
* A typical strategy is to start with an analysis of your data and create partitions that share characteristics like location, billing, and replication strategy, and then create one storage account for each partition.

**How many Storage accounts do we need?**

A storage account defines a policy that applies to all the storage services in the account.  
Settings that are controlled by a storage account:  
  
1) **Subscription**: Azure subscription that will be billed for the services in the account.  
2) **Location**: datacenter that will store the services in the account.  
3) **Performance**: Determines the data services in our storage account and type of hardware disks used to store.

* Standard allows you to have any data service (Blob, File, Queue, Table) and uses traditional magnetic disk drives.
* Premium limits you to one specific type of blob called a page blob and uses solid-state drives (SSD) for storage.

4) **Replication:** Strategy used to make copies of our data to protect against hardware failure or natural disaster. Azure storage accounts are always replicated to ensure high availability - this option lets you choose how far away the replication occurs to match your durability requirements

* Locally redundant Storage - Automatically maintains a copy of our data within the data center associated with the storage account. This guards against hardware failures, but not against event that incapacitates entire datacenter.
* Geo-redundant storage – Allows replication at different data centers across the world.

**5) Access Tier:** Controls how quickly we will be able to access blobs in this storage account. Applies only to Blob storage. Serves as default value for new blobs. Hot gives quicker access than Cool, but at increased cost. The Hot Access Tier is ideal for frequently accessed data, and the Cool Access Tier is better for infrequently accessed data. Note that this only sets the default value - when you create a Blob, you can set a different value for the data.

**6) Secure transfer required:** A security feature that determines the supported protocols for access. Enabled requires HTTPs, while disabled allows HTTP.

**7) Virtual Network:** A security feature that allows inbound access requests only from the virtual network(s) you specify.

Following 3 settings apply only to account, rather than the data stored in the account.

1. **Name:** Each storage account has a Globally Unique Name. Use only lowercase letters and digits and be between 3 and 24 characters.  
   The name will be used to generate the public URL used to access the data in the account.
2. **Deployment Model**: A deployment model is the system Azure uses to organize your resources. This model defines the API that you use to create, configure, and manage those resources. Azure has 2 deployment models:
   1. Resource Manager: the current model that uses the Azure Resource Manager API
   2. Classic: a legacy offering that uses the Azure Service Management API

Most Azure resources only work with Resource Manager.  
Storage accounts, virtual machines, and virtual networks support both, so we must choose one or the other when we create.

The Resource Manager model adds the concept of a resource group, which is not available in the classic model.

Microsoft recommends we use Resource Manager for all new resources.

1. **Account Kind:** Storage account kind is a set of policies that determine which data services we can include in the account and the pricing of those services. There are 3 kinds of accounts:
   1. StorageV2 (general purpose v2): Current offering that supports all storage types and all of the latest features
   2. Storage (general purpose v1): Legacy kind that supports all storage types but may not support all features
   3. Blob storage: Legacy kind that allows only block blobs and append blobs. No page blobs.

Microsoft recommends that we use the General-purpose v2 option for new storage accounts.   
Few exceptions: pricing for transactions is lower in general purpose v1, which would allow you to slightly reduce costs if that matches your typical workload.

The core advice here is to choose the Resource Manager Deployment model and the StorageV2 (general purpose v2) account kind for all your storage accounts.

**Storage account creation tool:**

Options are:

* Azure Portal
* Azure CLI (Command-line interface)
* Azure PowerShell
* Management client libraries

The portal provides a GUI with explanations for each setting. Portal is easy to use and helpful for learning about the options.

The other tools support automation. The Azure CLI and Azure PowerShell let us write scripts, while the management libraries allow us to incorporate the creation into a client app.

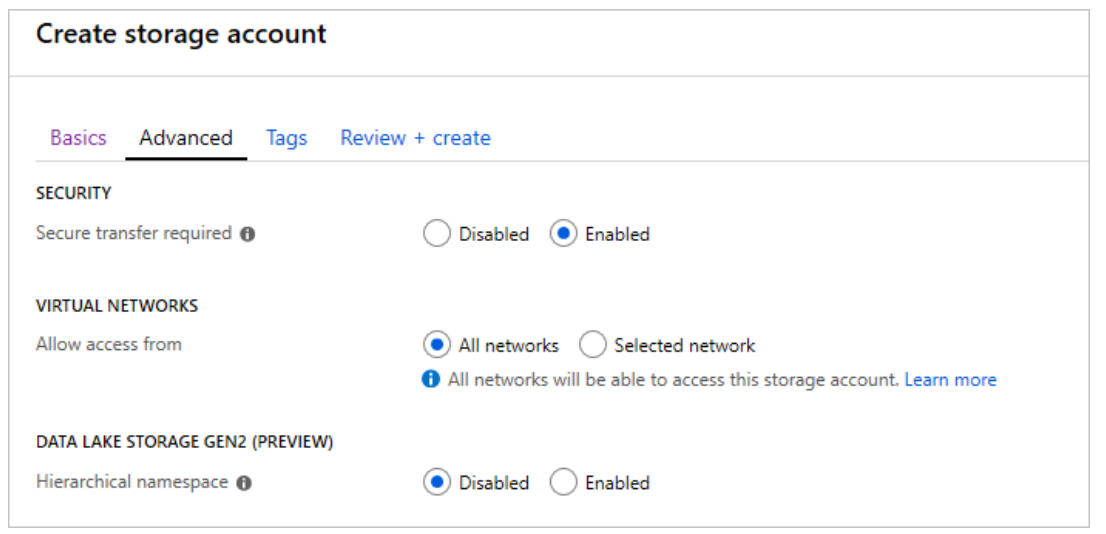
Storage-account creation is usually a one-time operation done at the start of a project. For one-time activities, the portal is the most common choice.

If we have an existing client application, the management libraries might be an attractive choice; otherwise, scripts will likely be a better option. Scripts are typically faster to create and less work to maintain.

**Advanced Options while creating storage account:**

1. Secure transfer required: This setting controls whether HTTP can be used for the REST APIs used to access data in the Storage account. Setting this option to Enabled will force all clients to use SSL (HTTPS). (Using HTTPS over the network is considered a best practice).
2. Virtual Networks: This option allows us to control access to the storage account for specific Vnet or All networks. Use “All Networks” to allow full public internet access.
3. DataLake Storage Gen2: Turns the storage account into a Datalake.

Storage Account Tags: This lets you associate key/value pairs to the account for your categorization and is a feature available to any Azure resource.



**Review Questions:**

Replication policy is a characteristic of a storage account. Every member in the storage account must use the same policy. If you need some data to use the geo-replication strategy and other data to use the local replication strategy, then you will need two storage accounts.

The storage account name is used as part of the URI for API access, so it must be globally unique.

**Introduction to Data Lake Storage Gen 2**

Azure Data Lake Storage provides a repository where we can upload and store huge amounts of unstructured data with an eye toward high-performance Big Data Analytics. It leverages Azure's storage capabilities to store large amounts of unstructured data to add value to their BI solution.

A data lake: Repository of data stored in its natural format (as blobs or files).

Azure Data Lake Storage is a comprehensive, highly scalable, and cost-effective data lake solution for big data analytics built in Azure.

It combines power of a high-performance file system, with a massive storage platform. This allows us to have quick insights in to our data.

It is built on Azure Blob Storage capabilities to optimize it specifically for analytics workloads. It also provides Blob Storage's tiering and data lifecycle management capabilities and the high-availability, security, and durability capabilities of Azure Storage.

**Benefits:**

* Designed to deal with this variety and volume of data at Exabyte scale while handling hundreds of gigabytes of throughput that is secure by design.
* This gives us the flexibility to use Data Lake Storage Gen2 as the basis for both real-time and batch solutions.
* HDFS compatible access: Data can be treated as if it's stored in a HDFS.  
  This allows us to store data in one place and access it through a wide range of compute technologies like Azure Databricks, HDInsight, and SQL Data Warehouse without moving the data between environments, enabling efficient usage while minimizing the cost.
* Security: ADLS Gen2 supports Access Control Lists (ACL) and POSIX permissions. We can set a granular level of permissions at a directory or file level for the data stored within the Data Lake. This Security is configurable through variety of tech like Spark, Hive, and Azure Storage Explorer.
* All Data Stored can be stored in encrypted at rest using either Microsoft of Customer Managed Keys.
* Performance: ADLS organizes stored data in a hierarchy of directories and subdirectories (like a file system) allowing for easier navigation. Thus, data processing requires less computational resources, which in turn reduces both the time and cost.
* Data Reduncancy: ADLS takes advantage of the Azure Blob replication models that provide data redundancy in a single data center (LRS), or to a secondary region using the GRS option. This allows data to be always available and protected if Catastrophe strikes.

ADLS Gen2 setup requires StorageV2 (General Purpose V2) Azure Storage account with the Hierarchical namespace enabled.

**Comparing Azure Blob Storage to Azure Data lake Storage:**

**Azure Blob:** Used to store large amount of unstructured (“object”) data in a single hierarchy (Flat Namespace) and can be accessed using HTTP or HTTPs. A great use case for blob storage is archiving rarely used data or storing website assets such as images and media.

**ADLS:** Built on Blob Storage optimizing I/O of high-volume data using hierarchical namespaces. Hierarchical namespaces organize blob data into directories and store metadata about each directory and the files within it. This structure allows operations like directory renames and deletes to be done in a single atomic operation. In contrast Flat namespaces, require several operations proportionate to the number of objects in the structure. This yields better storage and retrieval performance for an analytical use case, which lowers the cost of analysis.

Applications can use either the Blob APIs or Azure Data Lake Storage Gen2 file system APIs for accessing data.

**Stages of Big Data Processing:**

ADLS Gen2 plays a fundamental role in wide range of big data architectures like:

1. Modern Data warehouse.
2. Advanced Analytics against Big Data.
3. A Real-time Analytics solution.

4 stages for processing big Data (that are common to all, regardless of the architecture):

1. **Ingestion**: This identifies the technology and processes used to acquire the source data. This data can come from files, logs, and other types of unstructured data that must be put into the Data Lake Store. The technology used will vary depending on the frequency the data is transferred.
2. **Store**: This identifies where the ingested data should be placed. In this case its ADLS Gen2.
3. **Prep & Train**: This identifies the technologies that are used to perform data preparation and model training and scoring for data science solutions. Ex: Azure Databricks or Machine Learning Services.
4. **Model and Serve:** The model and serve step involves the technologies that will present the data to users. These can include visualization tools such as Power BI, or other data stores such as SQL Data Warehouse, Cosmos DB, Azure SQL, or Azure Analysis Services. Often, a combination of these technologies will be used.

**Big Data Use Cases:**

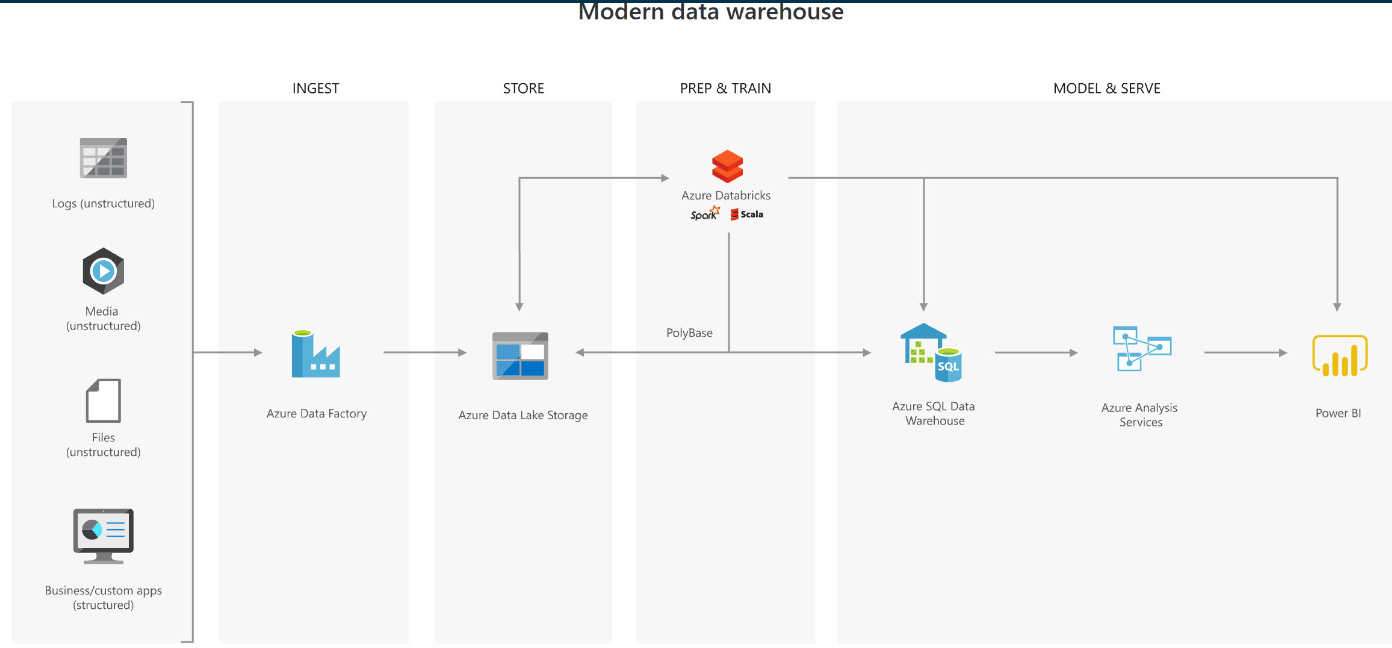
1. **Creating a Modern Data Warehouse:**

Older On-premises business intelligence solution: A Microsoft SQL Server Database Engine, Integration Services, Analysis Services, and Reporting Services to provide historical reports. Analysis Services Data mining component can be used to create a predictive analytics solution to predict the buying behavior of customers.   
Problems with this approach: Worked well for low volumes of data, but could not scale for larger volumes. Also it was difficult to deal with JSON Data.

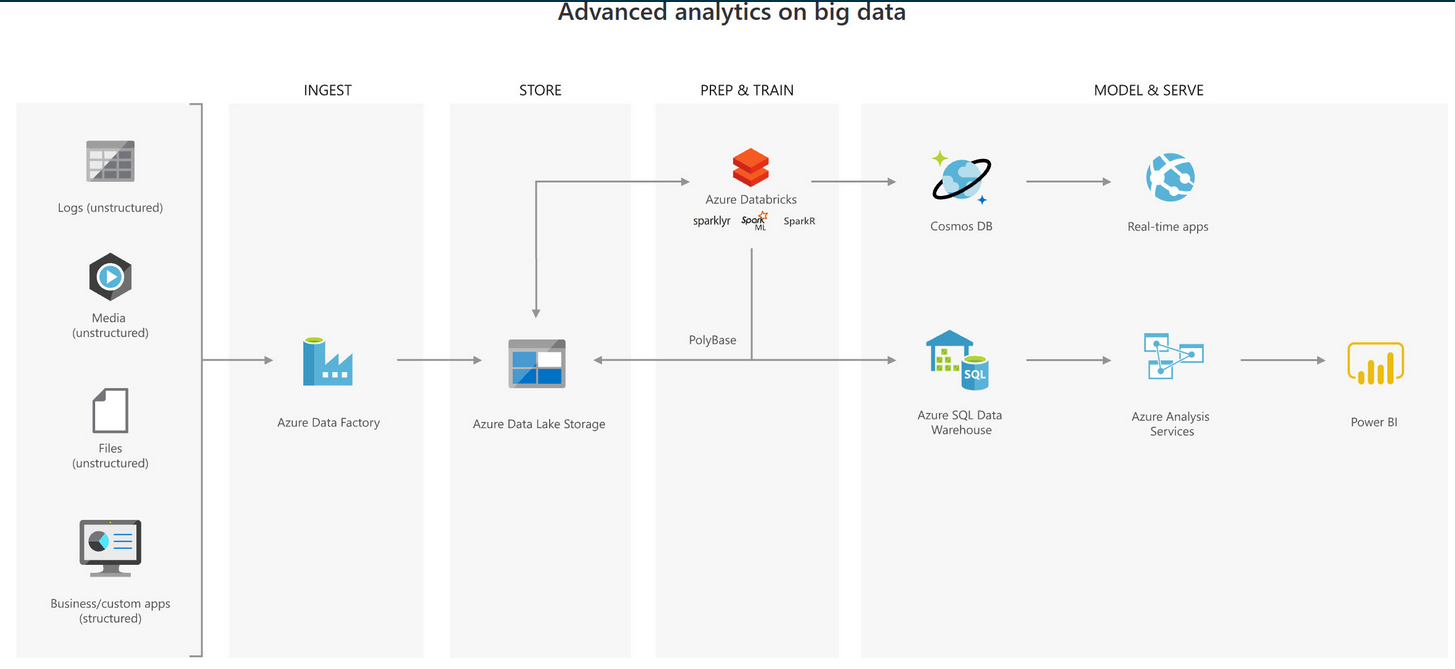
Modern Data Warehouse:

The modern data warehouse uses Azure Data Lake storage at its heart.

Integration Services is replaced by Azure Data Factory to ingest data into the Data Lake from a business application. This serves as the source of truth for the predictive model built in Azure Databricks, and PolyBase is used to transfer the historical data into a big data relational format held in using Azure Synapse Analytics. This in turn also stores the results of the trained model from Databricks.



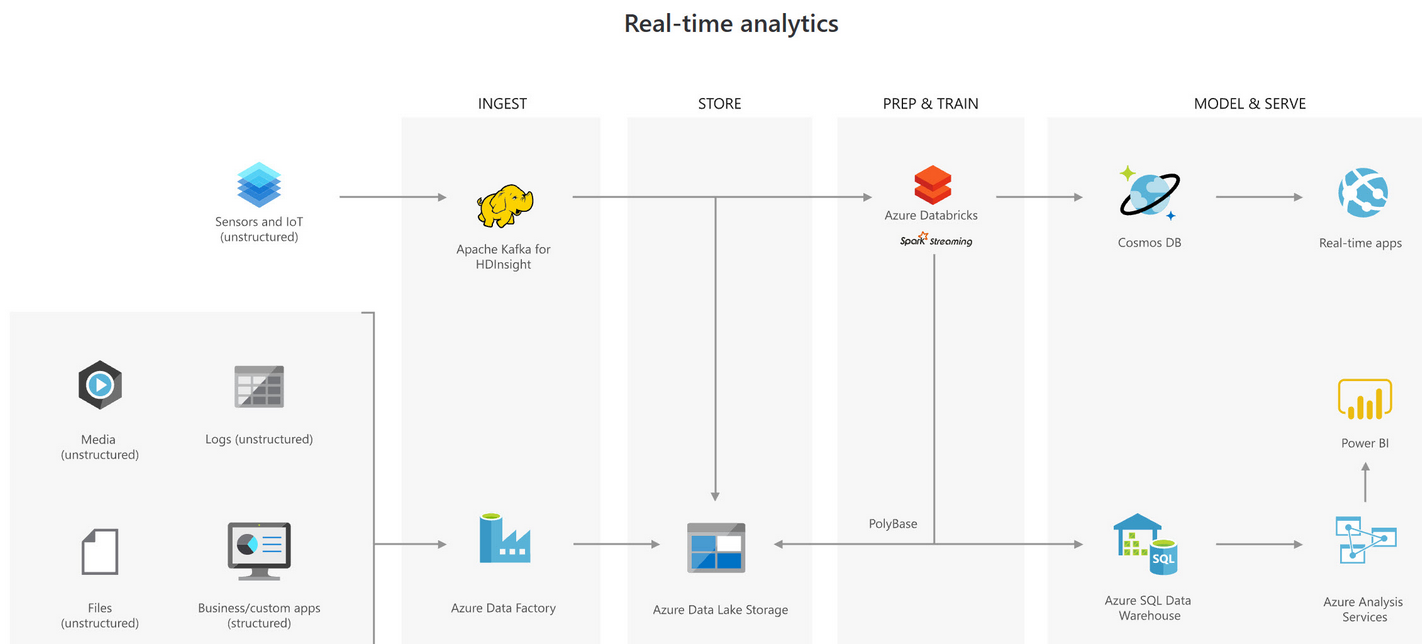
1. **Advanced Analytics for Big Data:**



In this solution, ADF transfers terabytes of web logs from a web server to the Data Lake on an hourly basis. This data is provided as features to the predictive model in Azure Databricks, which is then trained and scored. The result of the model is then distributed globally using Azure Cosmos DB that the real-time app will use to provide recommendations to the customers as they add products to their online basket.

To complete this architecture, PolyBase is used against the Data Lake to transfer descriptive data to the Azure Synapse Analytics for reporting purposes. This data is presented to Analysis Services to provide the caching capability for Azure Synapse Analytics to service many users and display the data through reports using Power BI.

1. **Real Time analytics Solution:**



In Real Time analytical solution, the major change that occurs is to the ingestion phase for processing big data solutions. In above architecture Apache Kafka for HDInsight is used to ingest streaming data from an Internet of Things (IoT) device, although this could be replaced with IoT Hubs and Stream Analytics. The key point is that the data is persisted in Data Lake Storage to service other parts of the solution.

**Summary:**

Azure Data Lake Storage Gen2 provides a cloud storage service that is highly available, secure, durable, scalable, and redundant.

Azure Data Lake Storage brings new efficiencies to process big data analytics workloads and can provide data to a multitude of compute technologies including HDInsight and Azure Databricks without needing to move the data around.

**Review Questions:**

If you want to enable the best performance for Analytical Workloads in Data Lake Storage Gen 2, then on the Advanced tab of the Storage Account creation set the Hierarchical Namespace to enabled.

Store is the phase in which Azure Data Lake Storage resides for processing big data solution. Ingestion is the phase that typically includes technologies for ingesting data from a source such as Azure Data Factory. Model & Serve is the phase that includes technologies for presenting data for end users such as Power BI.

**Upload Data into Azure Data Lake Storage Gen2**

* ADLS Gen2 is a data lake solution explicitly designed for enterprises to run large scale analytical workloads in the cloud.
* ADLS Gen2 takes the core capabilities from ADLS Gen1 including file system semantics and security and scale and combines it with the low-cost, highly available capabilities of Azure Blob Storage.

**Creating Azure Data Lake storage Gen 2 Account using Powershell:**

Open up the PowerShell console within the Azure Portal and type out the following:

$resourceGroup = "mslearn-datalake-test"   
$location = "westus2"   
New-AzResourceGroup -Name $resourceGroup -Location $location

New-AzStorageAccount -ResourceGroupName $resourceGroup `   
-Name "dlakedata001" `   
-Location $location `   
-SkuName Standard\_LRS `   
-Kind StorageV2 `   
-EnableHierarchicalNamespace $True

Creating an Azure Data Lake Storage Account Gen2 is the same as creating an Azure Blob Store, there's just one setting that is different. (Set Hierarchical namespace to Enabled on the Advanced Tab)

The storage account name must be unique across all of Azure. The portal will display a green checkmark next to the name when you have a valid entry.

Also note that for creating ADLS Gen2, the storage account kind should be StorageV2 (general-purpose-V2)

**Upload Data using Azure Storage explorer:**

Use Azure Storage Explorer to upload files and make ad-hoc transfers into an Azure Data Lake Storage.

It is a free application available for Windows, macOS, and Linux. The app is designed to manage unstructured data in Azure such as tables, blobs, queues, and files. It also supports data in Azure Cosmos DB and Azure Data Lake Storage.

Once installed, we can use Azure storage Explorer to perform several operations against data in Azure Storage account (including Data Lake):

* Upload files or folders from your local computer into Azure Storage.
* Download cloud-based data to your local computer.
* Copy or move files and folders around in the storage account.
* Delete data from the storage account.

Multiple options for connecting to storage account using Storage Explorer:

* Sign in with your Azure account to provide access to all your subscriptions.
* Use a connection string to access a specific Azure Storage account.
* Use a storage account name and access key.

**Create a File System using Azure Storage Explorer:**

Blobs are always uploaded into folders. This allows you to organize groups of blobs much like you organize files on your computer.

When working with ADLS, we start by creating a File system. This defines the specific container in Blob storage that will hold your data lake. We can then create folders and files within this dedicated area.

Note that when creating File System, the option says “Create Blob Container”. Container is blob accounts is equivalent to File System in ADLS.

Adding a folder provides a hierarchical structure for managing your data. We can create multiple levels in the account. However, we must ensure that parent folders exist before you create children.

**Azure Data Factory:**

ADF is a cloud-based data integration service that creates workflows in the cloud for orchestrating batch data movement and transformations.

Using ADF, we can create and schedule workflows (called pipelines) to ingest data from disparate data stores.

The data can then be processed and transformed with services such as:

* Azure HDInsight Hadoop
* Spark
* Azure Data Lake
* Azure Machine Learning

Many data orchestration tasks that can be conducted using Azure Data Factory. Ex: copy data from ADLS Gen1 to ADLS Gen2.

ADF name is also globally Unique. ADF version should be V2.

When we creating ADF we need to specify the location for the ADF. The data stores that are used by the data factory can be in other locations and regions.

**Module 5 - Working with Relational Data Stores in the Cloud**

Azure Relational Data Platform options:

* Azure SQL Database and
* The enterprise data warehouse capabilities using Azure Synapse Analytics.

**Azure SQL Database:**

Azure SQL Database are a Platform-as-a-Service (PaaS) offering, meaning much less infrastructure and maintenance to manage ourselves.

Azure SQL Database is a convenient, cost-effective, and secure way to host your relational databases.

* **Convenience:** Setting up SQL Server on a VM or on physical hardware requires us to know about hardware and software requirements, understand the latest security best practices and manage operating system and SQL Server patches on a routine basis, manage backup and data retention issues yourself.

With Azure SQL Database, Microsoft manage the hardware, software updates, and OS patches for you. All we specify is the name of your database and a few options.

We can bring up and tear down Azure SQL Database instances at our convenience.

* **Cost:** Azure SQL Database has several pricing options. These pricing options enable you to balance performance versus cost. We can start with just few dollars per month.
* **Scale:** With Azure SQL Database, we can adjust the performance and size of our database on the fly when your needs change.
* **Security:** Azure SQL Database comes with a firewall that's automatically configured to restrict connections from the Internet.

We can “whitelist” IP addresses we trust.

**Creating an Azure SQL Database:**

1. **One Server, Many Databases:**When we create first Azure SQL database, we also create an Azure SQL logical server. Logical server is an administrative container for our databases. This can include both Azure SQL Databases and Azure SQL Data Warehouse databases. We can control logins, firewall rules, and security policies through the logical server. We can also override these policies on each database within the logical server.
2. **DTUs Versus vCores:**Azure SQL Database has two purchasing models: DTU and VCore.

**DTU:** Database transaction Unit. It is a combined measure of compute, storage, and IO resources. DTU is a simple, preconfigured purchase option.

**eDTU :** Since a logical server can hold more than one DB, there is the idea of eDTUs (Elastic Transaction units). This option enables us to choose one price, but allow each database in the pool to consume fewer or greater resources depending on current load.

**VCore:** While the DTU model provides fixed combinations of compute, storage, and IO resources, the vCore model enables us to configure resources independently.

vCore gives us greater control over what compute and storage resources we use.

1. **SQL Elastic Pools:**

SQL elastic pools relate to eDTUs.

They enable us to buy a set of compute and storage resources that are shared among all the databases in the pool. Each database can use the resources they need, within the limits you set, depending on current load.

When we create an Azure SQL Database, we can create a SQL Elastic Pool.

1. **SQL Managed Instances:**

This option creates a database with near 100% compatibility with the latest SQL Server on-premises Enterprise Edition database engine.

Favorable for “Lift and Shift” scenarios to migrate on-premises SQL Server.

They allow us to migrate huge databases into Azure cloud without changing any code and stay compliant with all the applications.

1. **Collation:**

This refers to rules that sort and compare data.

Collation helps us define sorting rules when case sensitivity, accent marks, and other language characteristics are important.

Ex: SQL\_Latin1\_General\_CP1\_CI\_AS, means.

* + *Latin1\_General* refers to the family of Western European languages.
  + *CP1* refers to code page 1252 - a popular character encoding of the Latin alphabet.
  + *CI* means that comparisons are case insensitive. For example, “HELLO” compares equally to "hello".
  + *AS* means that comparisons are accent sensitive. For example, “résumé” doesn't compare equally to "resume".

If we don't have specific requirements around how data is sorted and compared, we choose the default collation.

1. **Resource limits:**

Azure SQL Database has a number of different offering that can be purchased including basic, standard and premium tiers. The tier selected differs in the maximum resource limits, which therefore has an impact on cost.

Over time if we realize we need additional compute power to keep up with demand, we can adjust performance options or even switch between the DTU and vCore performance models.

1. **Logical Server settings:**

| **Setting** | **Value** |
| --- | --- |
| Server name | A globally unique [server name](https://docs.microsoft.com/azure/architecture/best-practices/naming-conventions). |
| Server admin login | A [database identifier](https://docs.microsoft.com/sql/relational-databases/databases/database-identifiers) that serves as your primary administrator login name. |
| Password | Any valid password that has at least eight characters and contains characters from three of these categories: uppercase characters, lowercase characters, numbers, and non-alphanumeric characters. |
| Location | Any valid location from the available list. |

1. **SQL Server Firewall**: We can also specify which systems can access our database through the firewall. Initially, the firewall prevents all access to our database server from outside of Azure.

**Adding Data to Azure SQL Database:**

Microsoft maintain the infrastructure, software updates, and patches for your Azure SQL database. Beyond that, we can treat your Azure SQL database like you would any other SQL Server installation.   
For example, we can use Visual Studio, SQL Server Management Studio, Azure Data Studio, or other tools to manage our Azure SQL database.

**Azure Cloud Shell:**

Azure Cloud Shell is a browser-based shell experience to manage and develop Azure resources. (Interactive console that runs in the cloud).

Behind the scenes, Cloud Shell runs on Linux. But we have two experiences to choose from: Bash and PowerShell.

Cloud Shell is accessible from anywhere. Besides the portal, we can also access Cloud Shell from [shell.azure.com](https://shell.azure.com/), the Azure mobile app, or from Visual Studio Code.

Cloud Shell includes popular tools and text editors. Ex: az, jq, sqlcmd

* **az** is also known as the Azure CLI. It's the command-line interface for working with Azure resources
* **jq** is a command-line JSON parser. We can pipe output from az commands to this tool to extract important fields from JSON output.
* **sqlcmd** enables us to execute statements on SQL Server. We can use sqlcmd to create an interactive session with our Azure SQL database.

Cloud Shell makes it easy to access and work with your Azure resources. Because Cloud Shell is browser-based, we can access it from Windows, macOS, or Linux – essentially any system with a web browser.

**Getting Information about our Azure SQL DB:**

We can use the az utility to list our databases and show information about our database.

The az commands we run require the name of our resource group and the name of our Azure SQL logical server. To save typing, run this azure configure command to specify them as default values.

*az configure --defaults group=<rgn>[sandbox resource group name]</rgn> sql-server=<server-name>*

Note that we only need the logical name without the .database.windows.net suffix.  
  
Run az sql db list to list all databases on your Azure SQL logical server.

*az sql db list*

The output gives a large JSON output. Since we just want the database name, we should pipe the output to jq to print only the name fields.

*az sql db list | jq '[.[] | {name: .name}]'*

Above commands give the output as below:

*[*

*{*

*"name": "Logistics"*

*},*

*{*

*"name": "master"*

*}*

*]*

Like SQL Server, master includes server metadata, such as sign-in accounts and system configuration settings.

To get information about our database we can execute the below command:

*az sql db show --name Logistics*

This also gives a large output. To only view the name, maximum size, and status of our DB we can execute below:

*az sql db show --name Logistics | jq '{name: .name, maxSizeBytes: .maxSizeBytes, status: .status}'*

Above command shall give an output like below:

*{*

*"name": "Logistics",*

*"maxSizeBytes": 2147483648,*

*"status": "Online"*

*}*

**Connecting to Database:**

We can connect to a database using sqlcmd.  
CRUD – Create, Read, update and Delete.

Run the ‘az sql db show-connection-string’ command to get the connection string to the Logistics database in a format that sqlcmd can use.

*az sql db show-connection-string --client sqlcmd --name Logistics*

The output resembles this.

*"sqlcmd -S tcp:contoso-1.database.windows.net,1433 -d Logistics -U <username> -P <password> -N -l 30"*

We can make use of the output of the previous step to create an interactive session.

*sqlcmd -S tcp:contoso-1.database.windows.net,1433 -d Logistics -U martina -P "password1234$" -N -l 30*  
Place your password in quotes so that “&” and other special characters aren't interpreted as processing instructions.  
  
From the sqlcmd session, we can run following TSQL commands:

*CREATE TABLE Drivers (DriverID int, LastName varchar(255), FirstName varchar(255), OriginCity varchar(255));*

*GO*

*SELECT name FROM sys.tables;*

*GO*

*INSERT INTO Drivers (DriverID, LastName, FirstName, OriginCity) VALUES (123, 'Zirne', 'Laura', 'Springfield');*

*GO*

*SELECT DriverID, OriginCity FROM Drivers;*

*GO*

*UPDATE Drivers SET OriginCity='Boston' WHERE DriverID=123;*

*GO*

*SELECT DriverID, OriginCity FROM Drivers;*

*GO*

*DELETE FROM Drivers WHERE DriverID=123;*

*GO*

*SELECT COUNT(\*) FROM Drivers;*

*GO*

Using the above method, we can get the connection string for our favorite SQL management tool – whether that's from SQL Server Management Studio, Visual Studio, or something else.