Design multi-tenant SaaS applications with Azure Cosmos DB

By Sandeep S. Nair
Sr. Program Manager
Azure Cosmos DB Engineering Team
Feb 2023

Contents

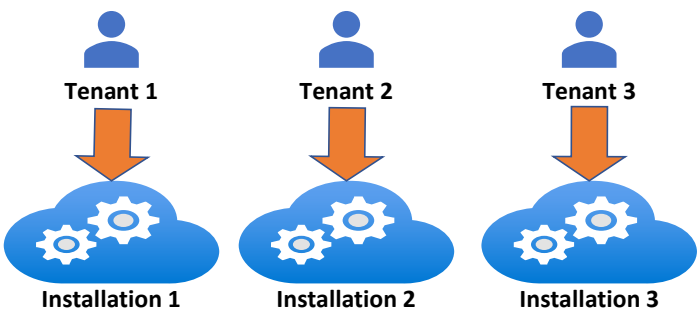
What is SaaS?

SaaS provides a complete software solution that the customer purchases on a pay-as-you-go basis from a cloud service provider. The customer rents the use of an app for your organization, and the customer's users connect to it over the Internet, usually with a web browser. All the underlying infrastructure, middleware, app software, and app data are in the service provider's data centre. The service provider manages the hardware and software, and with the appropriate service agreement, will ensure the availability and the security of the app and your data as well. SaaS allows the customer organization to get quickly up and running with an app at minimal upfront cost.

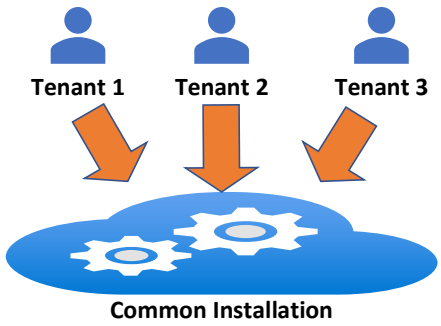
Gartner defines SaaS as "Software that is owned, delivered, and managed remotely by one or more providers. The provider delivers software based on one set of common code and data definitions that is consumed in a one-to-many model at all contracted customers at any time on a pay-for-use basis or as a subscription based on use metrics."

In the SaaS world, tenant refers to the customers/users that are going to rent/use the service. The individual or organization that develops, markets, and sells SaaS software is called an independent software vendor (ISV).

Technically speaking, SaaS services can be single tenant also, but dedicated installation and management per tenant/customer is expensive and non-scalable model. Hence this is not a preferred option in the SaaS world.



Single Tenant Design



Mutl-Tenant Design

Multi-tenant service where a single installation provides multiple logical installations by sharing the same installation across multiple customers/tenants is the chosen method. Since you don't need to provision any extra resources for each tenant, and you maintain only one set of your application across all your customers, it helps you lower the operational cost and makes it easier to maintain over time. Therefore, SaaS has become analogous with multi-tenant offerings.

While multi-tenant services provide many advantages, designing it well is critical for it to be successful. The following are minimum expectations from any SaaS service:

- **Security**: Data stored in the platform is safe from exfiltration. Data needs to be protected at rest and in motion using industry best practices.
- **Isolation**: Each customer would want themselves to be isolated from other tenants.
- **Scalability**: Since customer traffic can grow rapidly due to lower costs, trial periods, sale cycles etc. The service should be able to rapidly and automatically scale.
- **Reliability**: Service should have high redundancy and should be able to recover instantaneously in case of any failure. Service operation disruption should be almost zero, and there should be no loss of data even in case of disaster.
- **Geographically Distributed**: As the service gains popularity, you may add customers from varied and multiple geographical locations. You service availability in multiple geographies would be critical for the service growth.
- **Monitoring**: In a SaaS offering the cost of running your operations is a critical piece of information in calculating your subscription charges. Being able to monitor resource consumption granularly will help you understand the usage and price your subscription charges.
- **Consistent High Performance**: The service should be provided very fast responses even during peak load, also it should not bog down because of noisy neighbours. The service should be always available. High Availability and Low Latency are critical for success in todays' highly competitive world.

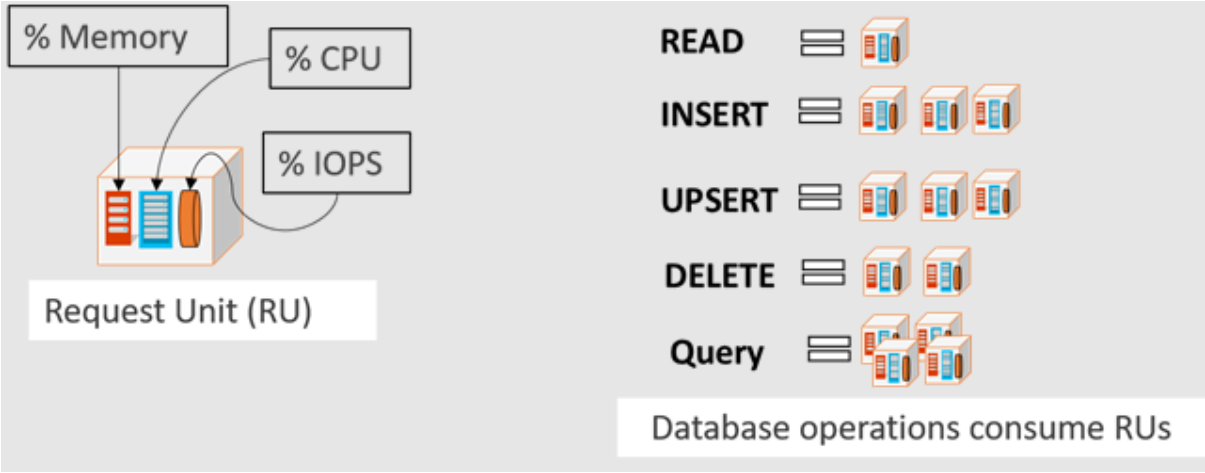
Azure Cosmos DB: The data backbone for SaaS applications

SaaS applications need to respond in real time to large changes in usage store ever increasing volumes of data, and make this data available to users in milliseconds. The performance of the SaaS Application is directly correlated to lower latencies in queries and lookups, higher throughput in terms on transactions per second/operations per minute, higher availability and consistency of data at global scale, and ability to cost efficiently access the underlying database by providing limitless, instant elasticity. Azure Cosmos DB is a fully managed database for modern app development, it offers single-digit millisecond response times, automatic and instant scalability, along with guaranteed speed at any scale. It provides cost-effective options for unpredictable or sporadic workloads of any size or scale, enabling developers to get started easily without having to plan or manage capacity. It takes care of all database administration and management activities saving developers time and money.

Limitless Scaling

Azure Cosmos DB is designed for unparalleled performance at any scale with instant and limitless elasticity. It allows customers to elastically scale throughput (number of database operations issued within a unit of time) based on their fluctuating workload traffic patterns.

Each Azure Cosmos DB operation ranging from simple point read/write to complex queries consumes system resources based on the complexity of the operation. The cost of all database operations in Azure Cosmos DB is expressed as Request Units (or RUs, for short). RU is a performance currency abstracting the system resources such as CPU, IOPS, and memory that are required to perform the database operations supported by Azure Cosmos DB.



RU/s is the normalized currency of throughput for various database operations.

Learn more about Request Units as a throughput and performance currency in Azure Cosmos DB

You can scale your Cosmos DB limitlessly simply by provisioning the appropriate number of RUs. You don't have to worry about any scaling operations like capacity management, replication, or hardware. Just pay for what you use and let Azure Cosmos DB automatically scale elastically with your app.

Serverless

The Azure Cosmos DB serverless offering lets you use your Azure Cosmos DB account in a consumption-based fashion. With serverless, you're only charged for the Request Units (RUs)

consumed by your database operations and the storage consumed by your data. Serverless containers can serve thousands of requests per second with no minimum charge and no capacity planning required, but it doesn't offer predictable throughput or latency guarantees.

Azure Cosmos DB serverless should be considered when you are developing, testing, prototyping new applications where the traffic pattern is unknown. It best fits scenarios where you expect intermittent and unpredictable traffic with long idle times. Please note that serverless accounts cannot be converted to provisioned throughput mode. You will have to copy the data from the serverless account to a new provisioned throughput account when migrating.

How to choose between provisioned throughput and serverless

Partitioning

An Azure Cosmos DB container is where data is stored. Unlike most relational databases which scale up with larger VM sizes, Azure Cosmos DB scales out by adding one or more servers, called partitions. To increase throughput or storage, more partitions are added. This scale-out architecture provides virtually unlimited throughput and storage for a container. Azure Cosmos DB elastically scales out storage and throughput without compromising the availability, consistency, and latency for your SaaS applications. Partitioning improves scalability, reduces contention, and optimizes performance. A well-designed partitioning strategy can deliver the same performance for a container regardless of its size, whether its 1MB or 1PB in size.

The items in a container are divided into distinct subsets called logical partitions based on the value of a partition key that is associated with item. All the items in a logical partition have the same partition key value.

Learn about Partitioning and horizontal scaling in Azure Cosmos DB

Hierarchical PK

A logical partition in Azure Cosmos DB cannot exceed 20 GB of storage, this can be a limiting factor if tenant Id is used as a partition key. Hierarchical partition keys help sub partitioning the container to overcome these limits.

Learn more https://learn.microsoft.com/azure/cosmos-db/hierarchical-partition-keys

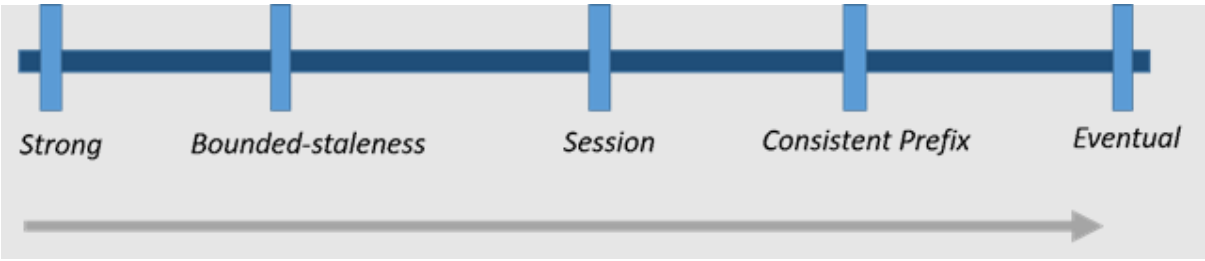
Choosing a partition key is one of most important decision as it determines whether your workload will scale or not. Your partition key choice affects the throughput control, latency, data isolation, storage limits, cost, and many other aspects of the database. Also you cannot change the partition key of an existing container.

Low Latency Guarantees at the 99th Percentile.

As part of its SLAs, Azure Cosmos DB guarantees end-to-end low latency at the 99th percentile to its customers. For a typical 1KB item, Azure Cosmos DB guarantees end-to-end latency of reads under 10ms and indexed writes under 15ms at the 99th percentile within the same Azure region. The average latencies are significantly lower (under 5ms). With an upper bound of request processing on every database transaction, Azure Cosmos DB allows clients to clearly distinguish between transactions with high latency vs. a database being unavailable.

Multiple, Well-Defined Consistency Models Backed by SLAs

Distributed databases that rely on replication for high availability, low latency, or both, must make a fundamental trade-off between the read consistency, availability, latency, and throughput as defined by the PACELC theorem. Azure Cosmos DB allows developers to choose between five well-defined consistency models along the consistency spectrum.



Learn more consistency levels in Azure Cosmos DB

Strong consistency model is the gold standard of data programmability, but it adds a steep price from higher write latencies. While eventual consistency offers higher availability and better performance, but it doesn't guarantee consistency. Session consistency is the most widely used consistency level for both single region as well as globally distributed applications. It provides write latencies, availability, and read throughput comparable to that of eventual consistency but also provides the consistency guarantees that suit the needs of applications written to operate in the context of a user. The default consistency level configured on your account applies to all Azure Cosmos DB databases and containers under that account.

Globally Distributed

Azure Cosmos DB transparently replicates data to all the regions associated with your Azure Cosmos DB account and allows you to read and write data from the regional replica of choice. Each partition in your primary region can be replicated to other Azure regions. When an application elastically scales throughput or storage, Azure Cosmos DB transparently performs partition management operations across all the regions.

Azure Cosmos DB being a ring zero Azure service will be available in any new Azure data centre as soon as it goes online. At any time, with a few clicks or programmatically with a single API call, you can add or remove the geographical regions associated with your Azure Cosmos DB database without requiring your application to be paused or redeployed.

Key benefits of global distribution in Azure Cosmos DB

99.99% High Availability

Azure Cosmos DB guarantees 99.99% availability SLA for every database account. The availability guarantees are agnostic of the scale (throughput and storage associated with a customer's database), number of regions (at least 2 required), or geographical distance between regions associated with a given database.

To always ensure high availability it's recommended to set up your Azure Cosmos DB account with a single write region and at least a second (read) region and enable Service-Managed failover. Azure Cosmos DB can be configured to accept writes in multiple regions, however multiple write regions doesn't guarantee write availability during a region outage. The best configuration to achieve high availability during a region outage is single write region with service-managed failover. Review your application design against these best practices.

Learn about High availability in Azure Cosmos DB

Resource Governance

Azure Cosmos DB distributes the provisioned throughput equally among the resource partitions. The automatic resource governance ensures that each partition delivers as per its assigned throughput and if it receives more requests than it was assigned, the client will receive "request rate too large" (throttling) with a back-off interval after which the client can retry.

To manage and plan capacity, Azure Cosmos DB ensures that the number of RUs for a given database operation over a given dataset is deterministic. You can examine the response header to track the number of RUs that are consumed by any database operation.

Understand the factors that affect RU charges and your application's throughput requirements, to run your application cost effectively. Optimizing the cost of your requests in Azure Cosmos DB | Microsoft Learn

The type of Azure Cosmos DB account determines the way consumed RUs get charged. There are three modes in which you can create an account, choose between manual and auto scale on Azure Cosmos DB based on your application requirements.

Priority Based Throttling (in Preview)

Priority based throttling allows users to specify priority for each data request sent to Azure Cosmos DB. Based on the priority specified, if there are more requests than provisioned throughput, then Azure Cosmos DB will throttle low priority requests to allow high priority requests to execute. This capability allows a user to perform more important tasks while delaying lesser important tasks and adds predictability while lowering cost.

Burst Capacity (in Preview)

Azure Cosmos DB burst capacity (preview) allows you to take advantage of your database or container's idle provisioned throughput capacity to handle spikes of traffic. With burst capacity, each physical partition can accumulate up to 5 minutes of idle capacity, which can be consumed at a rate up to 3000 RU/s. With burst capacity, requests that would have otherwise been rate limited can now be served with burst capacity while it's available.

Redistribute Throughput Across Partitions (in Preview)

By default, Azure Cosmos DB distributes the provisioned throughput of a database or container equally across all physical partitions. However, scenarios may arise where due to a skew in the workload or choice of partition key, certain logical (and thus physical) partitions need more throughput than others. For these scenarios, Azure Cosmos DB gives you the ability to redistribute your provisioned throughput across physical partitions. Redistributing throughput across partitions helps you achieve better performance without having to configure your overall throughput based on the hottest partition.

Learn More: Redistribute throughput across partitions (preview) in Azure Cosmos DB

Isolation Models

Azure Cosmos DB supports several isolation models, the isolation model you select will depend on your application design, throughput requirements, and the cost. Some of the design scenarios are listed below.

Refer to Azure Cosmos DB considerations for multitenancy - Azure Architecture Center to select your isolation model.

Dedicated Account per Tenant

Each Tenant has its own dedicated Azure Cosmos DB account. This design is a good choice if you have:

- Small number of regulated tenants who need very high level of isolation as part of their compliance requirements.
- Tenants who want control on data geo replication regions.
- Single-tenant application design.

Since each tenant resides in their own Azure Cosmos DB account this is not a multi-tenant database design. It provides the highest level of isolation available but adds complexity in the application code and increase maintenance cost and effort.

Pro	Con
Highest level of Isolation Full Control on throughout Control on Geo-replication regions Fully isolated from noisy neighbour Customer can use BYOK CMK for extra security of data.	Azure Cosmos DB SDKs are designed for singleton design pattern. An SDK instance can only connect to one Azure Cosmos DB account at any instance; hence this design is not compatible with singleton pattern. This makes the application code complicated. High cost as there is no sharing of resources. Would require provisioning/deprovision an account every time a tenant is added/removed from the application. Also, it's difficult to maintain. Cross tenant queries not possible

Named Container with Dedicated Throughput

All tenants share same Azure Cosmos DB account, and each tenant has its own dedicated container with dedicated throughput.

This can be a good choice if you have:

- Less than 500 large tenants (assuming one container/tenant)
- Tenants who need control on throughput.
- Tenants with widely varying throughput needs.
- Tenants who need isolation from Noisy neighbour.

This is a multi-tenant model where each tenant gets their own dedicated containers. This model provides high level of isolation as each tenant resides on their own physical partition(s), but it adds some complexity in the application code as each request will have to dynamically point to the appropriate container. You can manage the provisioned throughput for each container independently hence gives you flexibility to accommodate very large customers along with smaller customers. However, there is a limit of 500 containers in an account, hence you will not be able to provision large number of tenants using this model.

Pro	Con
High level of Isolation	Control on Geo-replication regions per tenant not possible
Can have dedicated throughout control per tenant.	Throughput sharing not possible, high cost/tenant.
Fully isolated from noisy neighbour	Complicated application code
Tenant size is not limited (using hierarchical partitioning)	Cross Tenant Queries not possible

Shared Containers Dedicated Throughput

All tenants share same Azure Cosmos DB account and container, the collection has dedicated throughput which is shared across tenants. Provides for unlimited automatic scaling of tenant size and tenant count.

This is the most scalable design and provides throughput guarantees, when a tenant is small it is accommodated in a shared container partition, but with increase in storage the tenant may

get an independent container partition. However, you cannot control the partition assignment to tenants and hence can't guarantee isolation.

Pro	Con
Unlimited automatic scaling	Control on Geo-replication regions per tenant not possible
Throughput sharing keeps cost low.	Cross Tenant Queries possible, but expensive
Tenant size is not limited (using hierarchical partitioning)	Not isolated from noisy neighbour.
Simpler application code and no maintenance overheads.	Mixed level of Isolation
Throughput redistribution can be used to provide more throughput to some tenant partitions.	

The delete by partition key feature (in preview) deletes all documents with the same logical partition key value, this could be useful in removing data of customers who exit your application.

Hybrid Approaches

- You can consider a combination of the above approaches to suit different tenants' requirements and your pricing model. For example:
- Provision all free trial customers within a shared container and use the tenant ID or a synthetic key partition key.
- Offer a higher silver tier that provisions dedicated throughput for the tenant's container.
- Offer the highest gold tier, and provide a dedicated database account for the tenant, which also allows tenants to select the geography for their deployment.

Multi Model with Multiple API Options

Azure Cosmos DB is a multi-model database service, which offers an API projection for all the major NoSQL model types; Column-family, Document, Graph, and Key-Value. Azure Cosmos DB offers multiple database APIs, which include NoSQL, MongoDB, PostgreSQL Cassandra, Gremlin, and Table. Azure Cosmos DB offers multiple database APIs. All the APIs offer automatic scaling of storage and throughput, flexibility, and performance guarantees.

The NoSQL API provide support for querying items using a Structured Query Language (SQL) syntax designed for querying JSON data. There are SDKs available for a popular language like .NET, Java, Node.js, Spark, and Python.

However, if you want to use the open-source developer ecosystem without the overhead of management, and scaling approaches, you can use the Cosmos DB API for MongoDB, PostgreSQL, Cassandra, or Gremlin. These API's implement the wire protocol of open-source database engines and allow your applications to treat Azure Cosmos DB as if it were various other database technologies. Azure Cosmos DB helps you to use the ecosystems, tools, and skills you already have for data modelling and querying with its various APIs.

Azure Cosmos DB NoSQL API is the native API and offers the best end-to-end experience as it has full control over the interface, service, and custom-built SDK client libraries. Any new feature that is rolled out to Azure Cosmos DB is first available on API for NoSQL accounts.

Considerations when choosing an API

Security

Azure Cosmos DB is a ring zero Azure service. This means it must keep all its compliance certificates current. Azure Cosmos DB has a plethora of certifications that you can read more about in the blog post "Azure #CosmosDB: Secure, private, compliant. Azure Cosmos DB also implements stringent security practices like encryption at rest and in transit without any configuration by the customer. Learn about how Azure Cosmos DB secures your database.

To help ensure the best level of network security, Azure Cosmos DB supports a secret based authorization model that utilizes a strong Hash-based Message Authentication Code (HMAC), it also enforces TLS 1.2 on all network traffic. Azure Cosmos DB provides the following additional layer of network security.

- Azure Cosmos DB provides IP-based access controls for inbound firewall support like the firewall rules of a traditional database system.
- Azure private endpoint for Azure Cosmos DB connects your application privately and securely to the Azure Cosmos DB account. The private endpoint is a set of private IP addresses in a subnet within your virtual network. You can then limit access to an Azure Cosmos DB account over private IP addresses. When Private Link is combined with restricted NSG policies, it helps reduce the risk of data exfiltration.

Authentication and Access Control

Azure Cosmos DB accounts are by default authenticated using shared secrets (Primary/secondary keys) allowing any management or data operation. It comes in both read-write and read-only variants. However, Azure Cosmos DB customers need to securely manage

these shared secrets, any exposure of these keys can lead to data breach. To overcome this challenge Azure Cosmos DB exposes a built-in fine-grained role-based access control (RBAC) system using Azure Active Directory identities.

Its recommend to disable shared secrets if you are using RBAC for authentication and authorization.

BYOK Customer Managed Keys

Data stored in your Azure Cosmos DB account is automatically and seamlessly encrypted with keys managed by Microsoft (service-managed keys). Optionally, you can choose to add a second layer of encryption with keys you manage (customer-managed keys or CMK). Currently, customer-managed keys are available only at an account level.

Soon you will be able to extend the CMK encryption mechanism to individual containers. This would allow each Tenant to manage their keys and add another layer of encryption for their data.

Know More: Configure cross-tenant customer-managed keys for your Azure Cosmos DB account with Azure Key Vault (preview)

Client-Side Encryption

Always Encrypted is a feature designed to protect sensitive data, such as credit card numbers or national identification numbers (for example, U.S. social security numbers), stored in Azure Cosmos DB. Always Encrypted allows clients to encrypt sensitive data inside client applications and never reveal the encryption keys to the database. Always Encrypted allows clients to encrypt sensitive data inside their applications and never reveal the plain text data or encryption keys to the Azure Cosmos DB service. Always Encrypted for Azure Cosmos DB is currently supported in In .NET and Java.

You can use up to 20 encryption keys per database. Adding additional encrypted properties to an existing encryption policy or performing a rotation of a data encryption key isn't supported. Hence plan your encryption policy carefully. In case of a dire requirement please migrate to a new container with an appropriate encryption policy.

Know More: Use client-side encryption with Always Encrypted for Azure Cosmos DB

Monitoring

Monitoring your Application resources for their availability, performance, and operation is a fundamental requirement to ensure your critical applications and business processes run as expected. You can monitor the metrics of your Azure Cosmos DB account and create

dashboards from the Azure Monitor. Azure Monitor collects the Azure Cosmos DB metrics by default, you will not need to explicitly configure anything. These metrics are collected with one-minute granularity, the granularity may vary based on the metric you choose. The Azure Cosmos DB portal allows you to monitor the metrics including throughput, storage, availability, latency, consistency, and system level metrics. Additionally, you can monitor your Azure Cosmos DB account programmatically by using the .NET, Java, Python, Node.js SDKs, and the headers in REST API.

Measuring RU Costs

Azure Cosmos DB reports the RU per operation as part of the response header for each request.

All the responses in the SDK, including CosmosException, have a Diagnostics property that records all the information related to the single request, including RUs consumed, no of retries or any transient failures, activity ID, exception/stack trace information, HTTP status/sub-status code, diagnostic string. Alternatively, you can measure consumption by using the Azure portal.

Having a good understanding of the RU consumption pattern in your application is critical to operate a highly scalable, yet cost effective solution in Cosmos DB, you may have to periodically review and fine tune your database operations to bring in efficiency in your solution.

Analytics

To analyse large operational datasets while minimizing the impact on the performance of mission-critical transactional workloads, traditionally, the operational data in a database is extracted and processed by Extract-Transform-Load (ETL) pipelines. ETL pipelines require many layers of data movement resulting in much operational complexity, and performance impact on your transactional workloads. It also increases the latency to analyse the operational data from the time of origin. Hybrid transactional and analytical processing (HTAP) is an emerging application architecture that breaks the wall between transaction processing and analytics. Azure Synapse Link for Azure Cosmos DB is a cloud native HTAP capability that enables near real time analytics over operational data in Azure Cosmos DB. Azure Synapse Link creates a tight seamless integration between Azure Cosmos DB and Azure Synapse Analytics. The benefits of using Synapse link are:

- Reduced complexity with No ETL jobs to manage.
- No impact on operational workloads
- Cost-effective

You can also use the Azure Cosmos DB Analytical Store Spark Connector (in Preview) with Azure Databricks to work with a centralized data lake and join with diverse data sources.

Learn More : Azure Synapse Link for Azure Cosmos DB, benefits, and when to use it

You can connect to Azure Cosmos DB from Power BI desktop by using one of these options:

- Use Azure Synapse Link to build Power BI reports with no performance or cost impact to your transactional workloads, and no ETL pipelines.
- You can either use DirectQuery or import mode. With DirectQuery, you can build dashboards/reports using live data from your Azure Cosmos DB accounts, without importing or copying the data into Power BI.
- Connect Power BI Desktop to Azure Cosmos DB account with the Azure Cosmos DB connector for Power BI. This option is only available in import mode and will consume RUs allocated for your transactional workloads.

SDK in All Major Languages

Azure Cosmos DB SDK provides client-side logical representation to access the Azure Cosmos DB for NoSQL. SDKs are available for a choice language like .NET, Java, Node.js, Spark, and Python.

First Party Integrations

Azure Data Factory

Using Azure Data Factory can ISVs enrich their SaaS apps with integrated hybrid data as to deliver data-driven user experiences. Pre-built connector with Azure Cosmos DB and integration at scale enables you to focus on your users while Data Factory takes care of the rest.

Azure Kubernetes Service

Azure Kubernetes Service (AKS) offers the quickest way to start developing and deploying cloud-native apps in Azure, datacentres, or at the edge with built-in code-to-cloud pipelines and guardrails. Azure Kubernetes Services provides a means for Kubernetes to provision and manage Azure services through its Azure Service Operator (ASO). ASO consists of custom resource definitions (CRD) that define Azure Cosmos DB services. Users can provision and manage Azure Cosmos DB service in a clean and concise manner using ARM, BICEP, or ASO through their CI/CD pipeline.

Sample : Build and deploy containerized apps with Azure Kubernetes Service & Azure Cosmos DB - Azure Cosmos DB Blog (microsoft.com)

Azure Cognitive Search

To make full use of their data applications often require gaining more insights, explore and search this data. Azure Cognitive Search, an AI enabled PAAS search service, can be integrated with Cosmos DB to not only extract critical knowledge from the stored data, but then enable users to search, find answer to questions and extract deep insights found within this data.

Learn how to index data from Azure Cosmos DB in Azure Cognitive Search

Azure Functions

Azure Functions can connect with Azure Cosmos DB without the users having to write their own integration code. These bindings, which represent both input and output, are declared within the function definition.

Learn how to work with Azure Cosmos DB trigger and bindings for Azure Functions.

Azure Synapse Analytics

Azure Synapse Link for Azure Cosmos DB is a cloud-native hybrid transactional and analytical processing (HTAP) capability that enables near real time analytics over operational data in Azure Cosmos DB. Azure Synapse Link creates a tight seamless integration between Azure Cosmos DB and Azure Synapse Analytics.

Azure Databricks (in Preview)

Enables ISVs to query Cosmos DB analytical store from Databricks to support customer scenarios where they want a centralized data lake and join with diverse data sources. Lights up end-to-end analytical story, providing ISVs with the flexibility to use analytical platform of their choice in a seamless way.