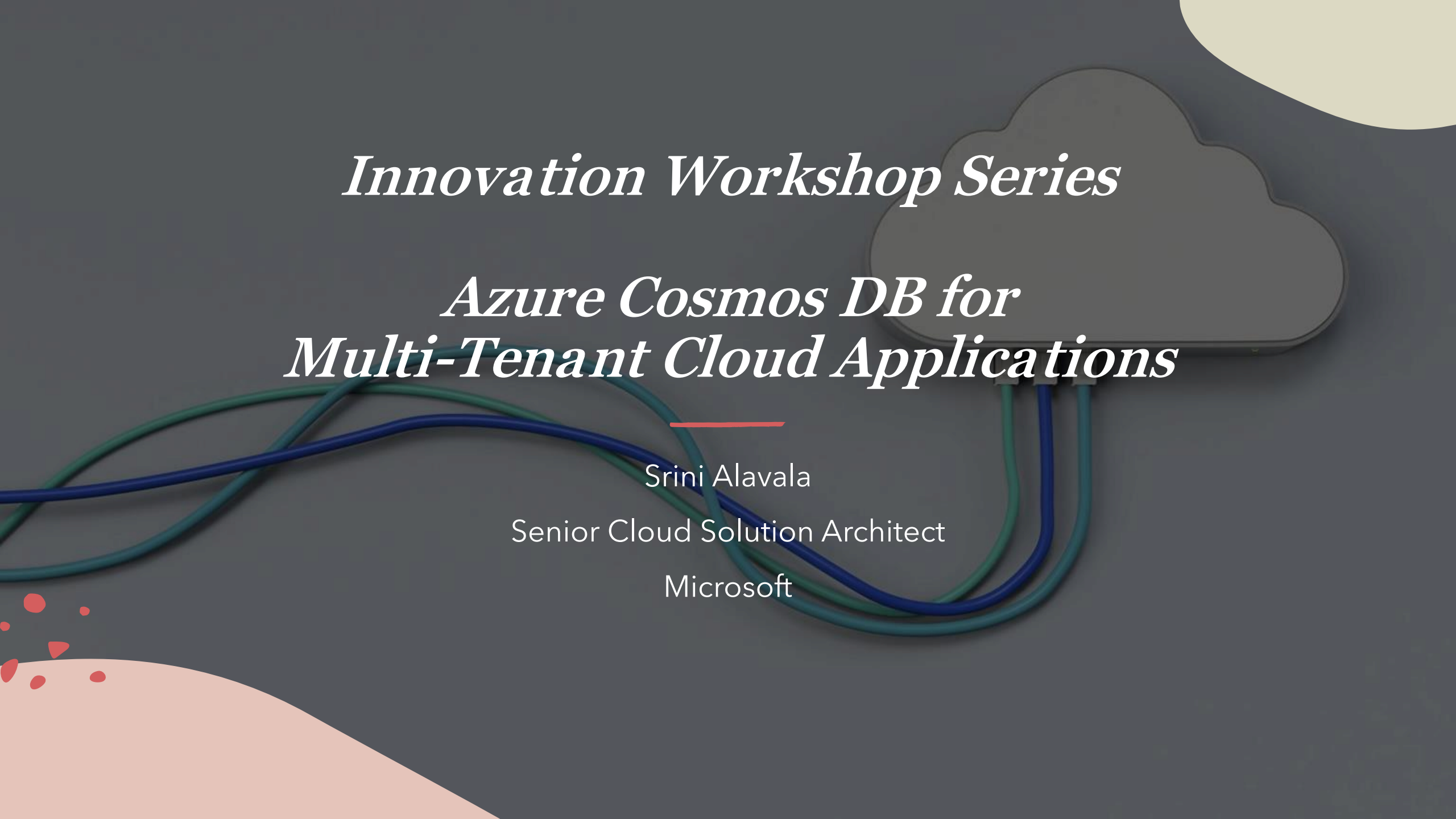


Innovation Workshop Series

Azure Cosmos DB for Multi-Tenant Cloud Applications

Srini Alavala

Senior Cloud Solution Architect

Microsoft

Agenda

Quick Introductions – **5 min**

Cosmos DB Overview – **15 min**

Challenge-1: Provision Azure Cosmos DB Service in Azure Workshop Subscription – **20 min**

Challenge-2: Data Modelling for SaaS Applications – **20 min (Breakout Session)**

Break – 10 min (12 Noon EST)

Challenge-3: Design Cosmos DB for SaaS Data Models and Load Data – **40 min (Breakout Session)**

Challenge-4: Review High Availability, Highly Scalable Throughput & Low Latency Features – **20 min**

Break – 10 min (1:10 – 1:20 EST)

Challenge-6: Build .NET/Java/Node.js/Python application using Cosmos DB Emulator - **60 min**

Recap & Closing **(2:30 – 3:00PM)**

Proctors for the Workshop

Cosmos DB Proctors

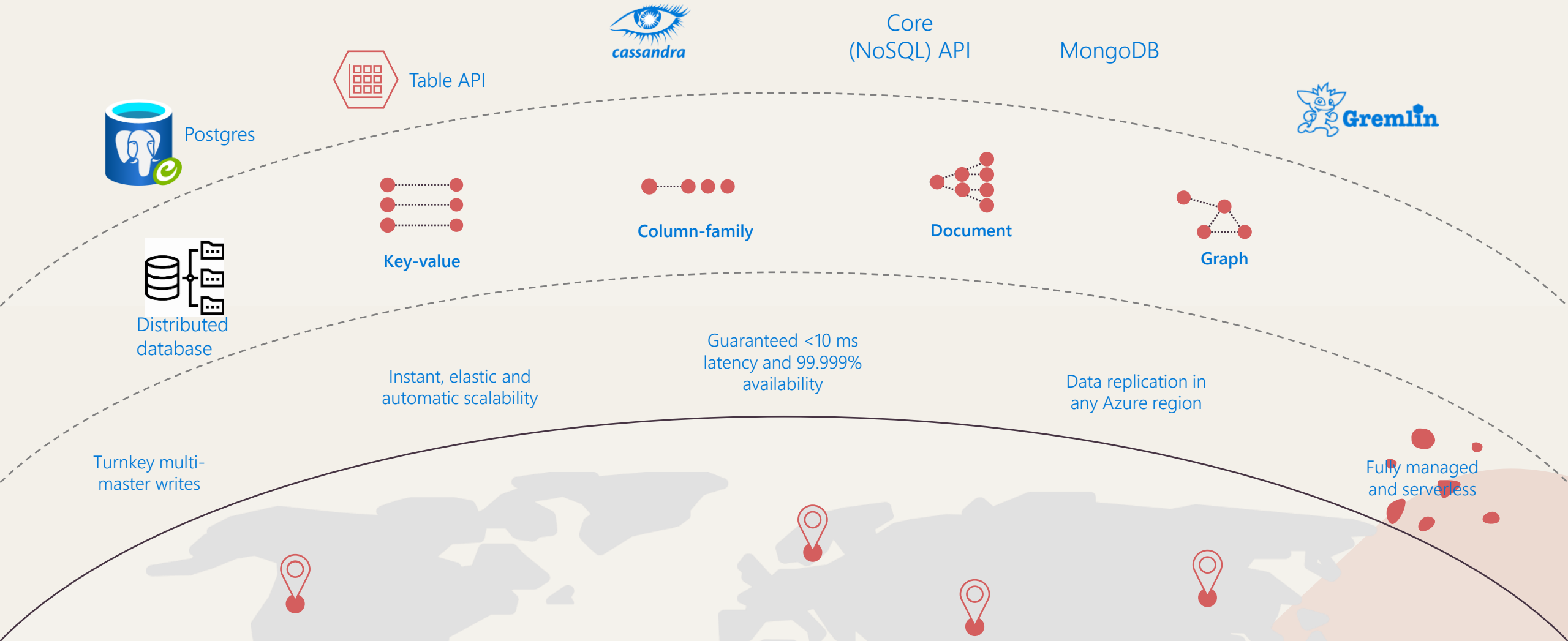
Arif Samad
Kirby Repko
Kirk Hofer
Tim Arif
Tomas Kovarik
Vijay Patil
Geraldine Caszo
Vibha Venkannagari
Venkat Kailasam
Safdar Zaman

Technical & Organizational Proctors

Joyjeet Majumdar
Hosai Yonoszai
Xavier Elizondo
Mark Larson
Austin Hidalgo

AZURE COSMOS DB

Fast NoSQL and relational databases for any scale



Azure Cosmos DB

A fully managed NoSQL database for modern app development with SLA-backed speed and availability, automatic and instant scalability, and open-source APIs for MongoDB, Cassandra, NoSQL engines and PostgreSQL Relational engine.

Guaranteed speed at any scale

Gain unparalleled SLA-backed speed and throughput, fast global access, and instant elasticity.

Faster & Productive application development

Build fast with open source APIs, multiple SDKs, schemaless data, and no-ETL analytics over operational data.

Mission-critical ready

Guarantee business continuity, 99.999% availability, and enterprise-level security for every application.

Fully managed and cost-effective

End-to-end database management with serverless and automatic scaling matching your application and TCO needs.

Customer Use Cases

Serverless applications with low latency, scale rapidly and globally	Well suited for Web, Mobile, Gaming and IOT applications	Workloads with massive amount of data, reads and writes at global scale, near real-time response
IOT applications to ingest bursts of data from sensors distributed across many locations.	Real-time personalization and recommendations.	eCommerce application which store catalogs and manage event data from order processing.
Manage customer generated data from blog posts, ratings and comments.	Gaming services which dynamically share information between players located around the world.	Developers don't have to choose different databases for different data models unlike GCP or AWS.

Azure cosmos DB Provisioning Options

A globally distributed, massively scalable, multi-model database service offers:

- One Free Tier for life per Customer – 1000 RU/s and 25GB Storage
- Container Dedicated or Database Shared throughput
- Auto Scale throughput for bursty, unpredictable workloads. Scale between 10% and Max.
- Serverless with no minimum charge
- Zone and Region Redundancy for HA
- Reserve Capacity for cost savings

Lift and shift MongoDB apps

Run Spark over operational data

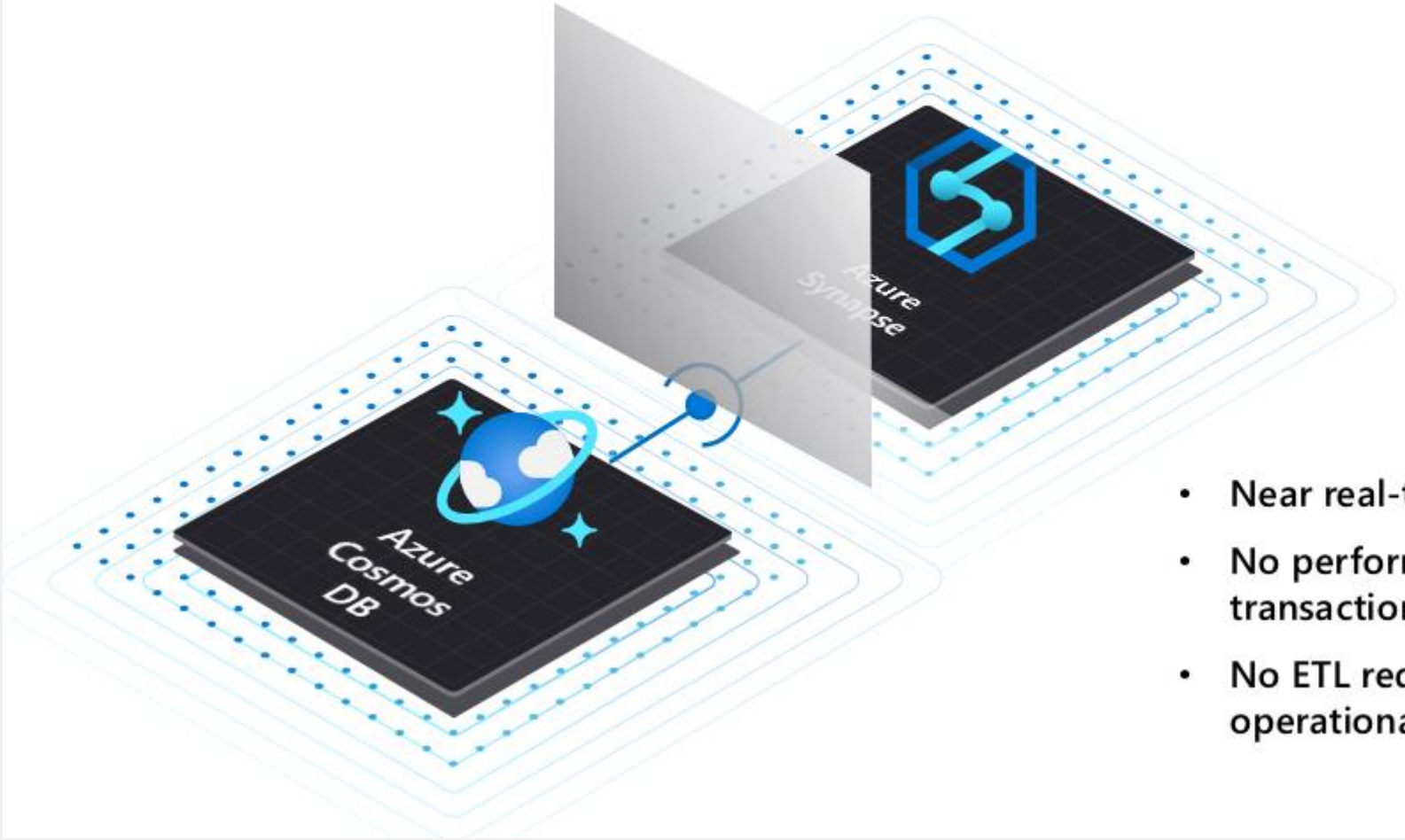
Build real-time customer experiences

Ideal for IoT, gaming and eCommerce

Cosmos DB with synapse link

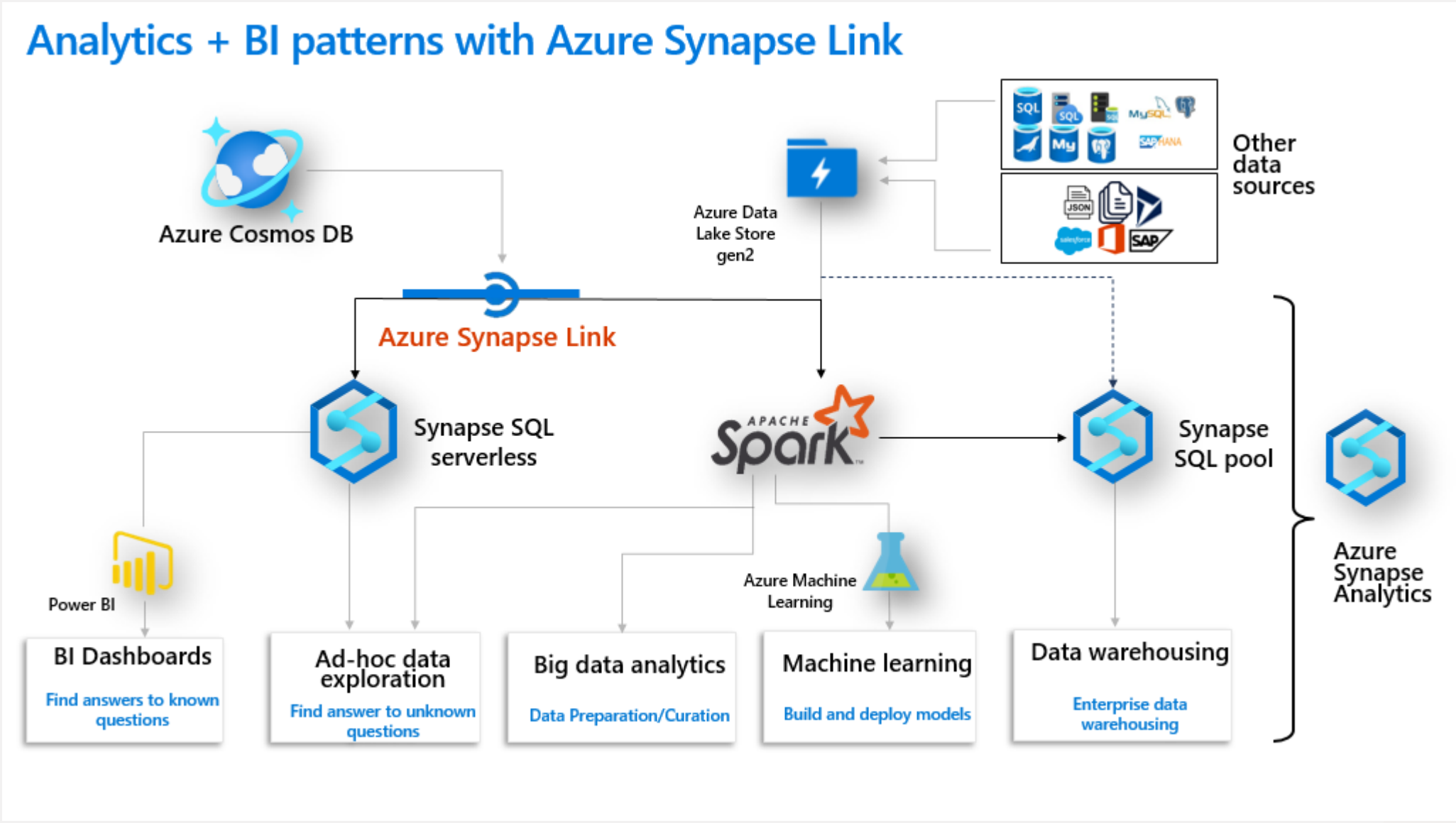
Azure Synapse Link for Azure Cosmos DB

Breaking down the barrier between OLTP & OLAP



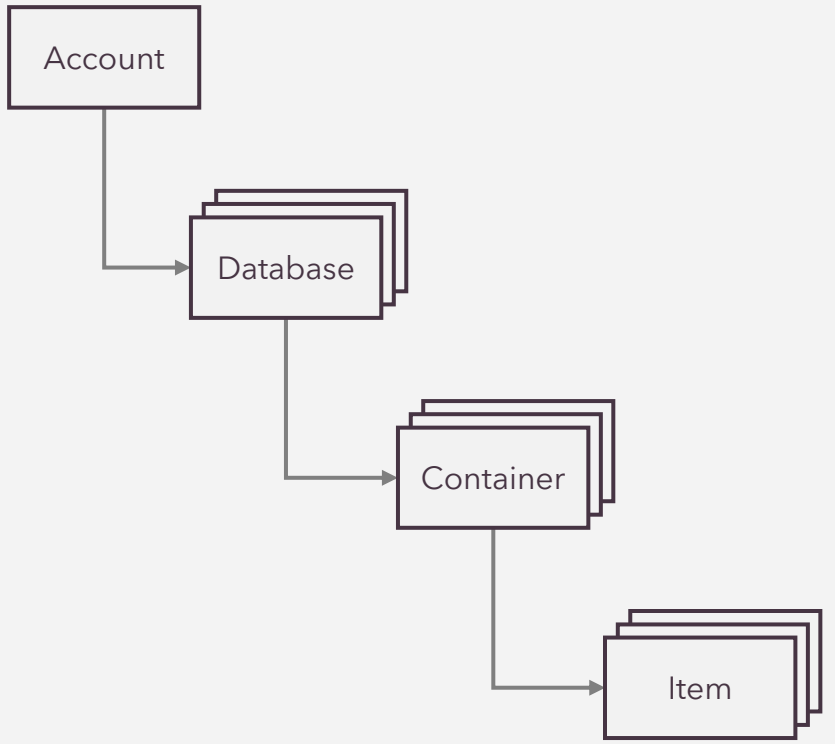
- Near real-time data analytics
- No performance impact on transactional workloads
- No ETL required to analyze operational data

Synapse link

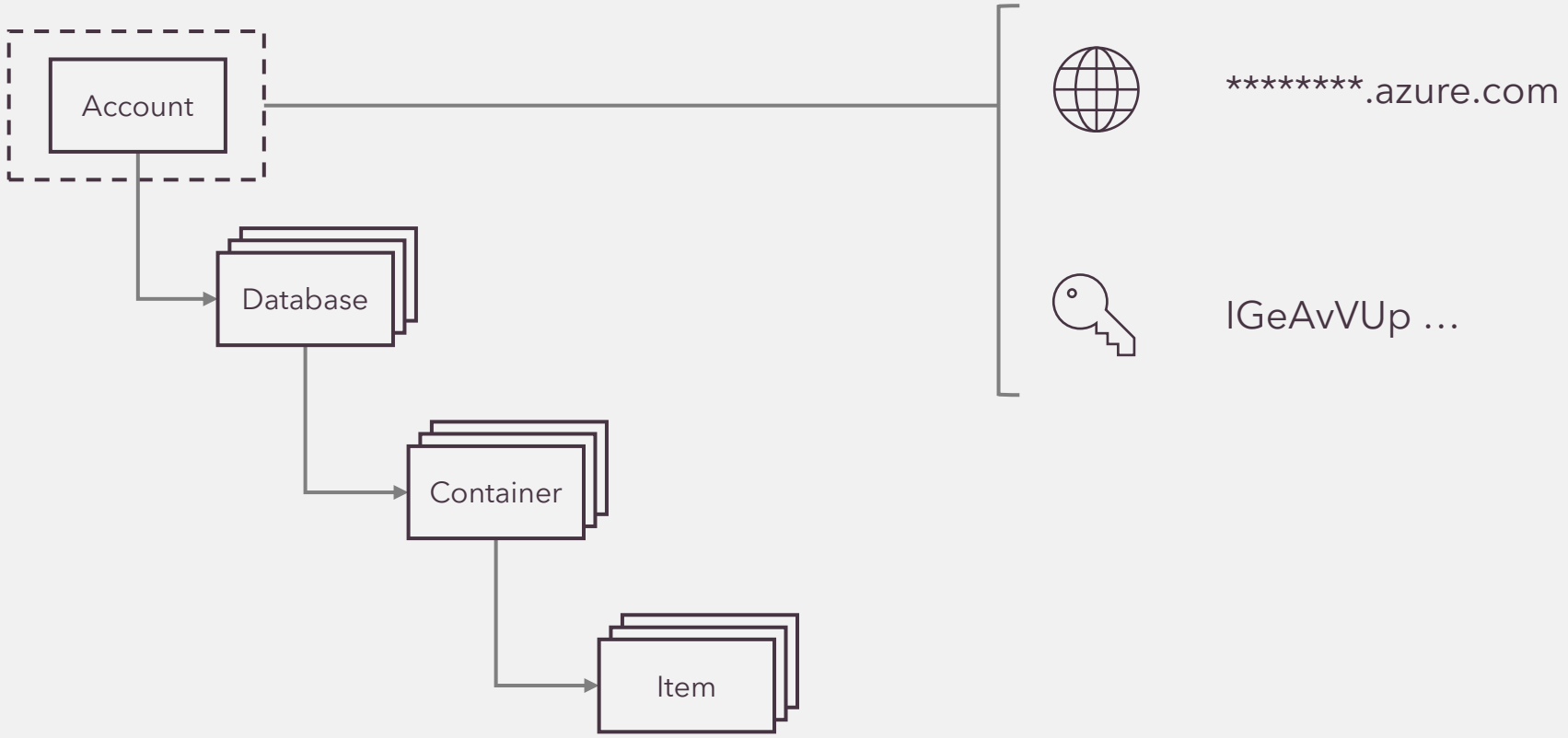


Resource Model

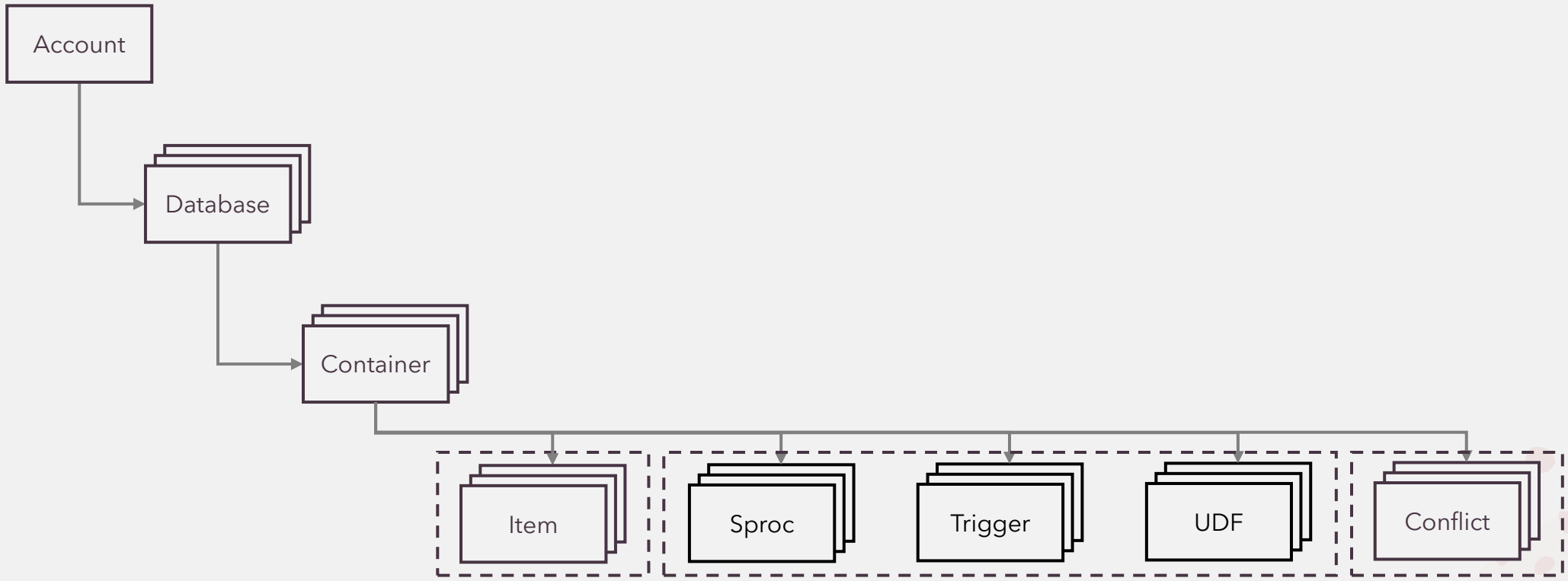
Leveraging Azure Cosmos DB to automatically scale your data across the globe



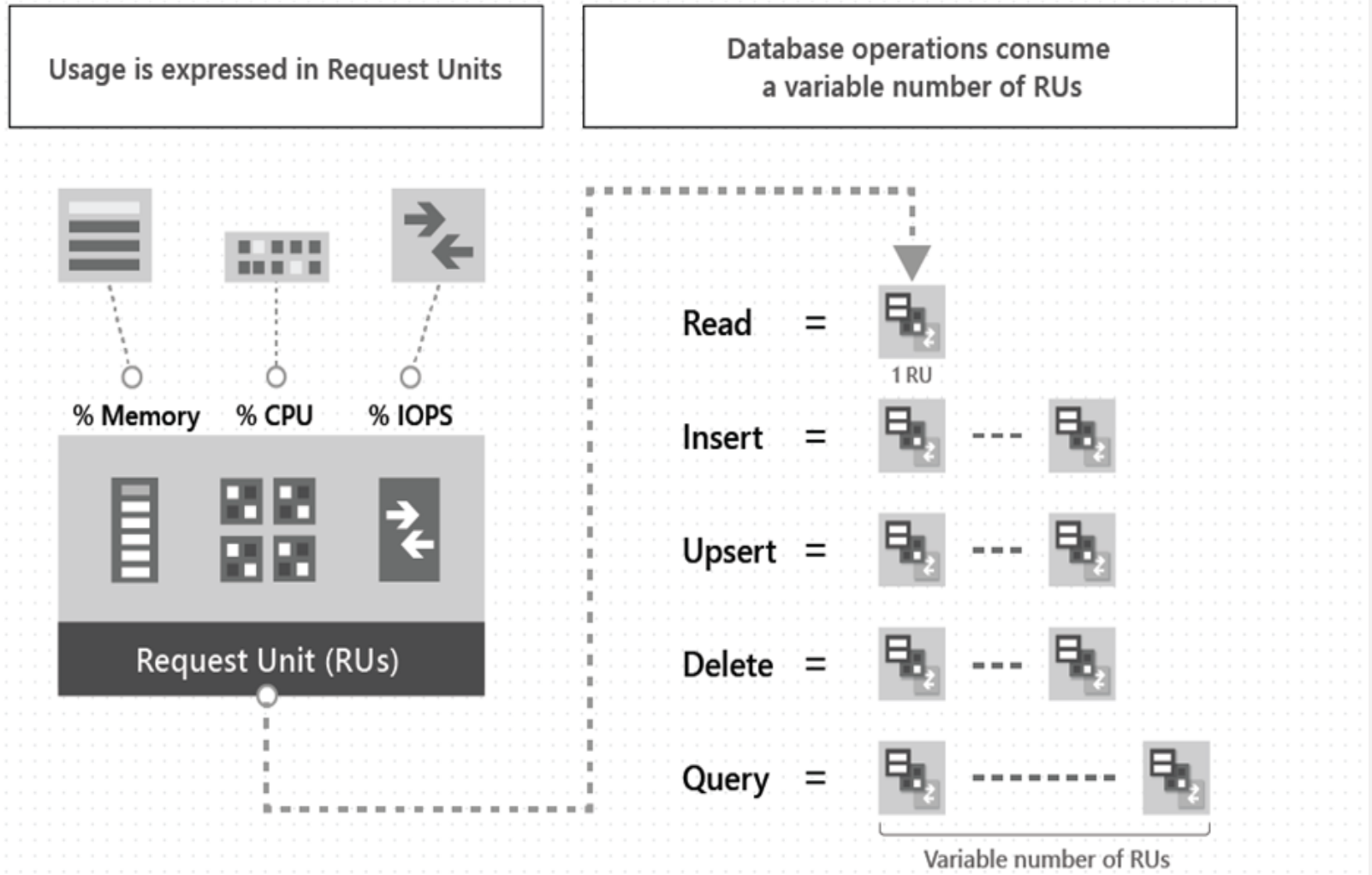
Account URI and Credentials



Container-Level Resources



Request Units(RU)



- Each API has its own database operations and consumes system resources based on the complexity of the operation.
- Cost of DB operations is normalized as Request Units.
- Cost of 1 KB item point read is 1 RU.

Request Units

Provisioned in terms of RU/sec

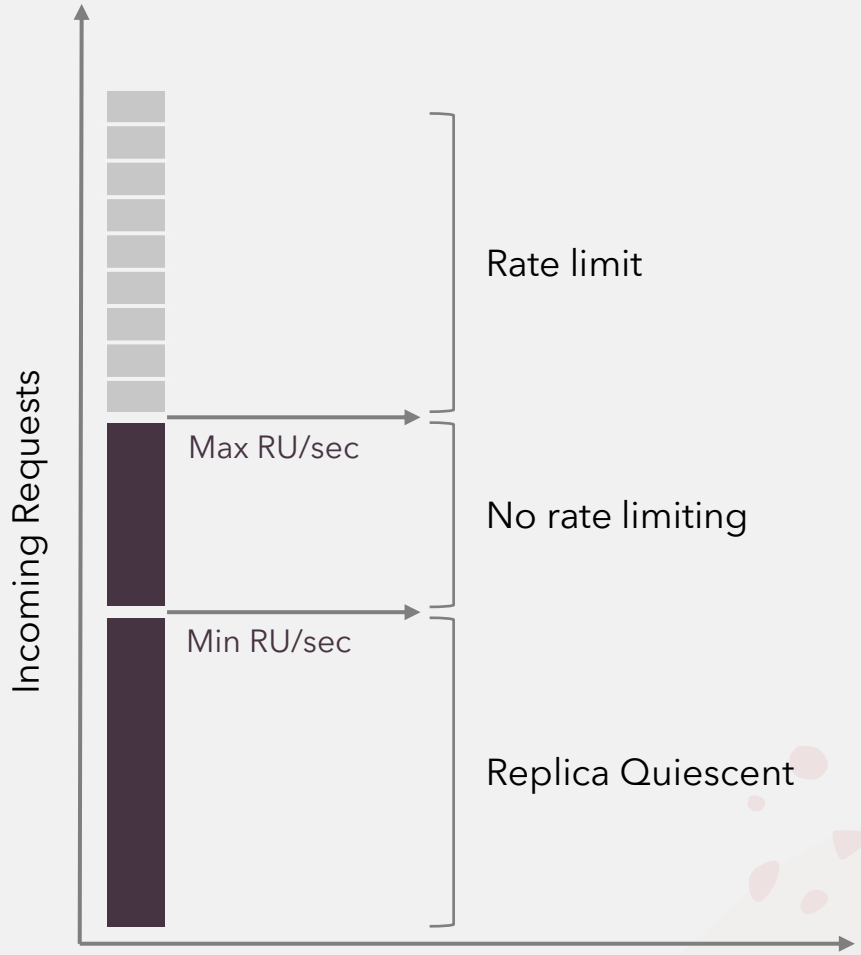
Rate limiting based on amount of throughput provisioned

Can be increased or decreased instantaneously

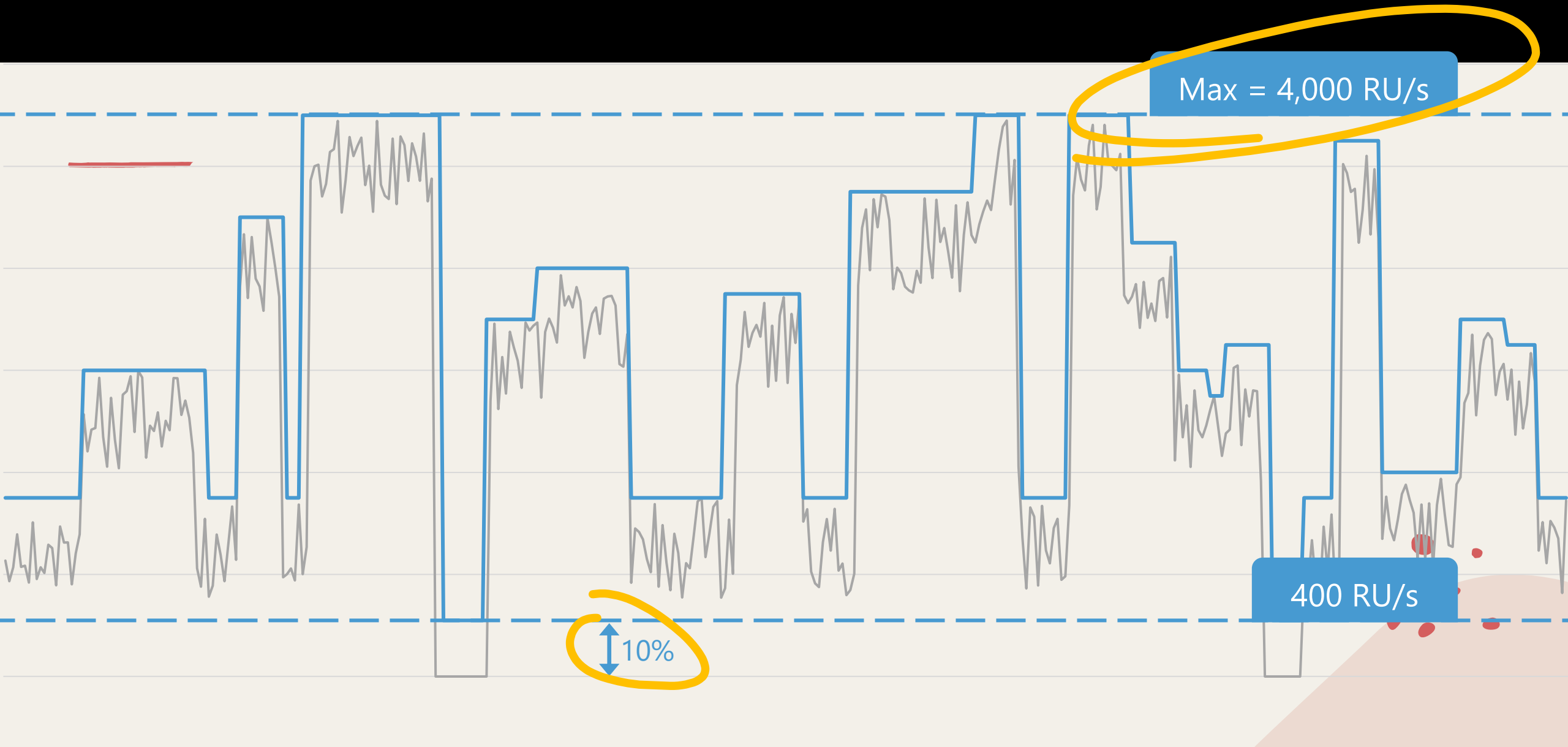
Metered Hourly

Background processes like TTL expiration, index transformations scheduled when quiescent

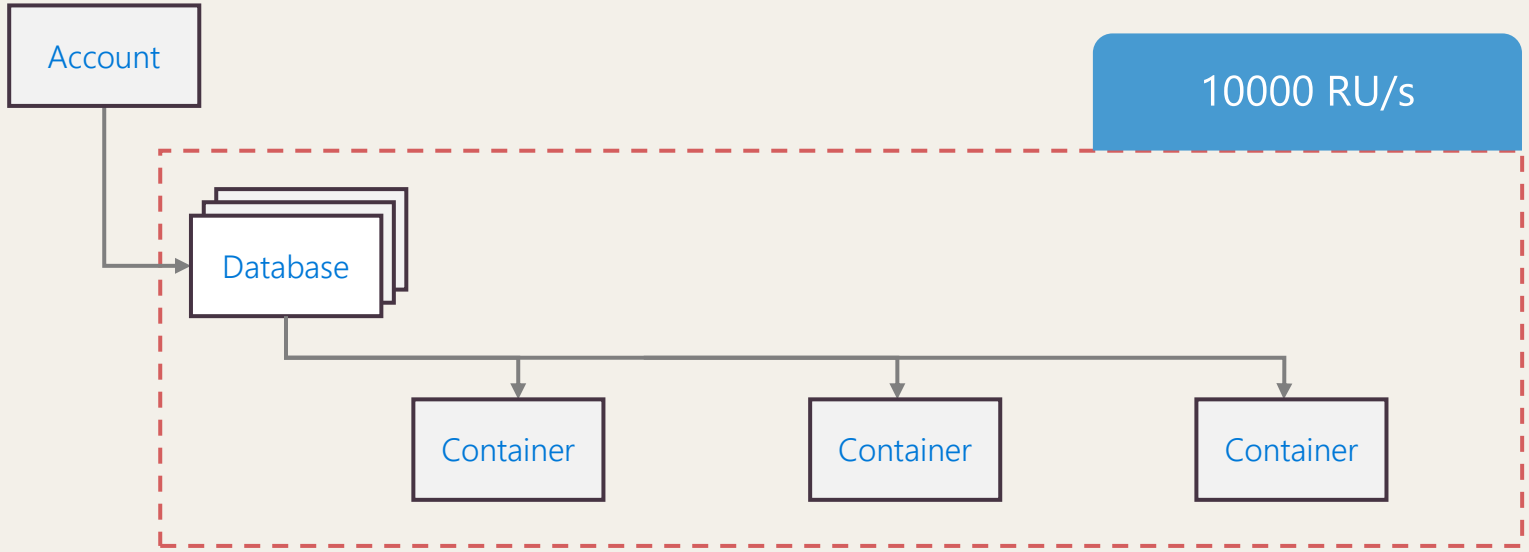
Fixed or Auto Scale options



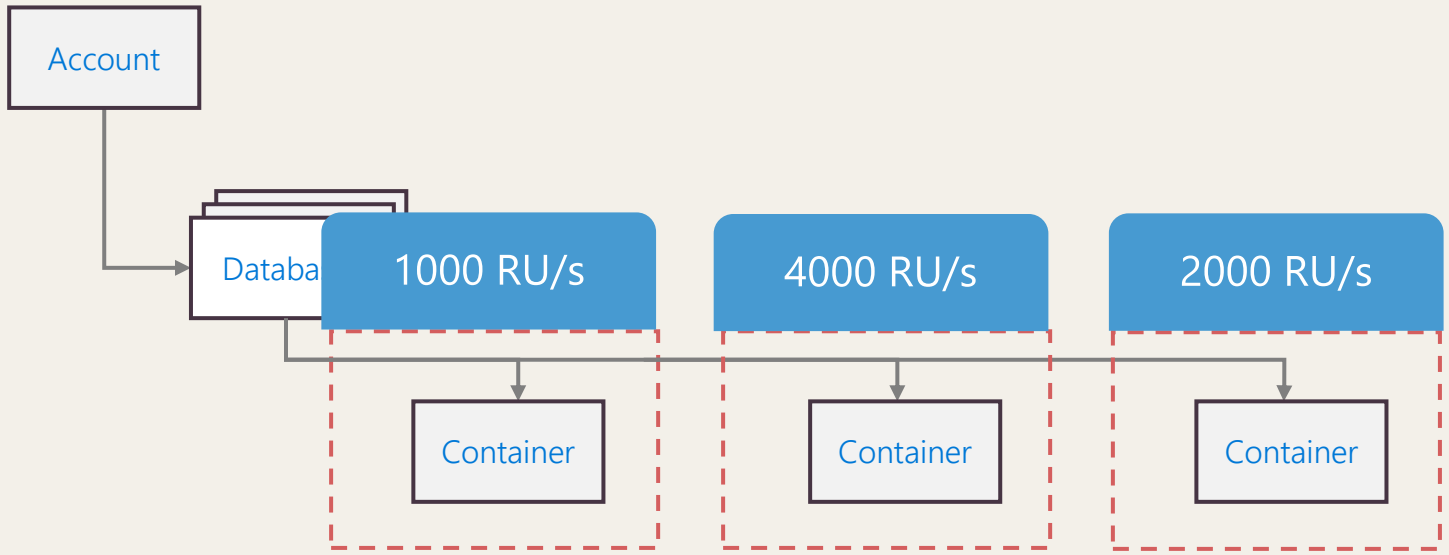
Autoscale Provisioned Throughput



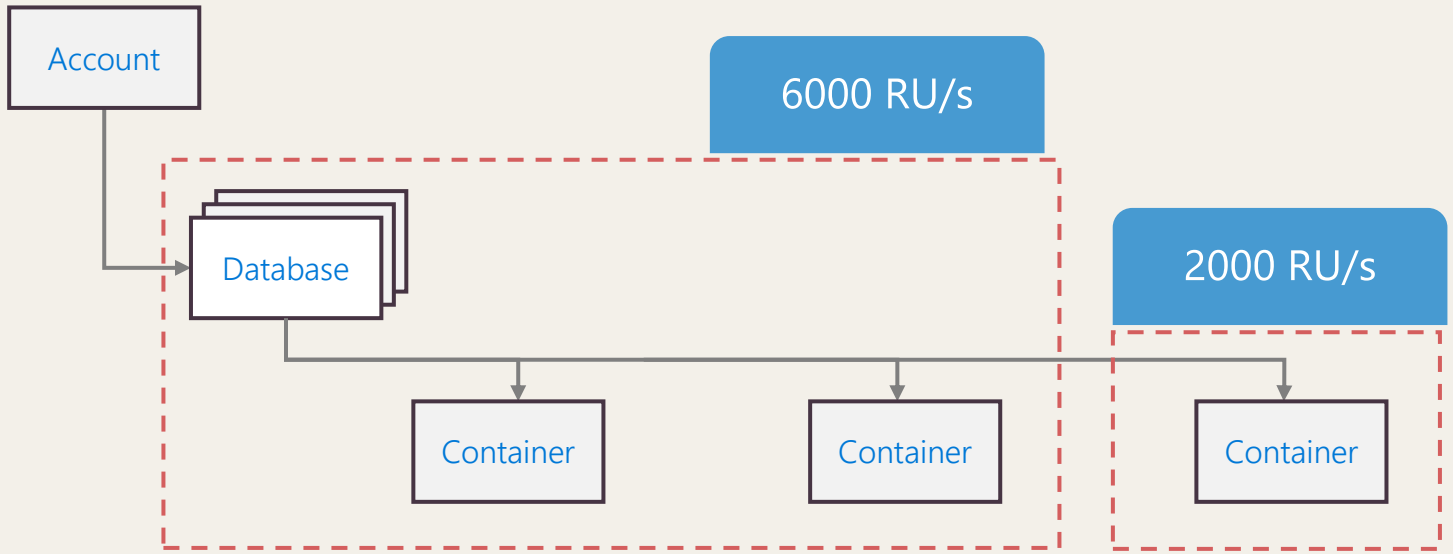
Database Level Throughput Provisioning



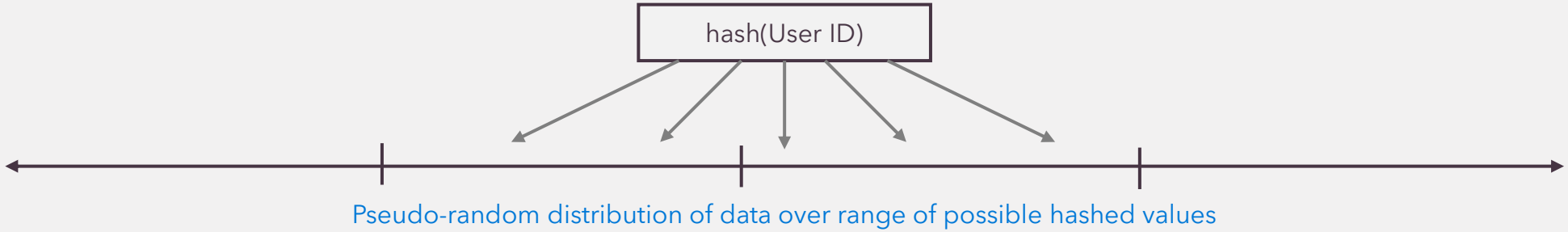
Container Level Throughput Provisioning



Throughput Provisioning - Mixed

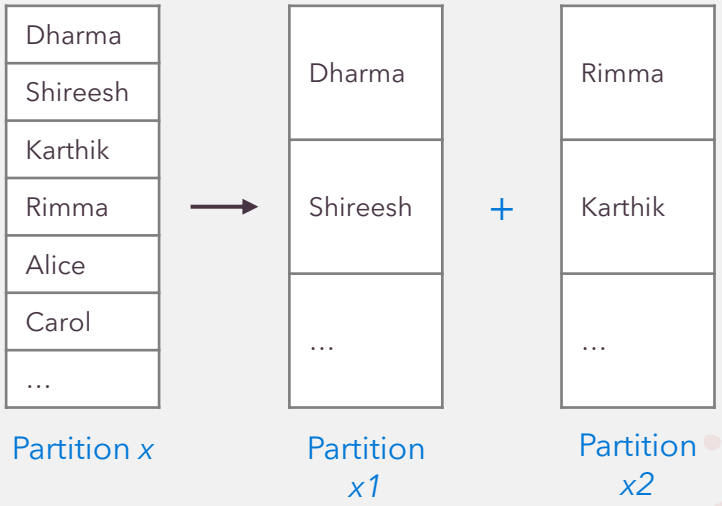


Partitions



Partition Ranges can be dynamically sub-divided to seamlessly grow database as the application grows while simultaneously maintaining high availability.

Partition management is fully managed by Azure Cosmos DB, so you don't have to write code or manage your partitions.



Partition Key storage limits

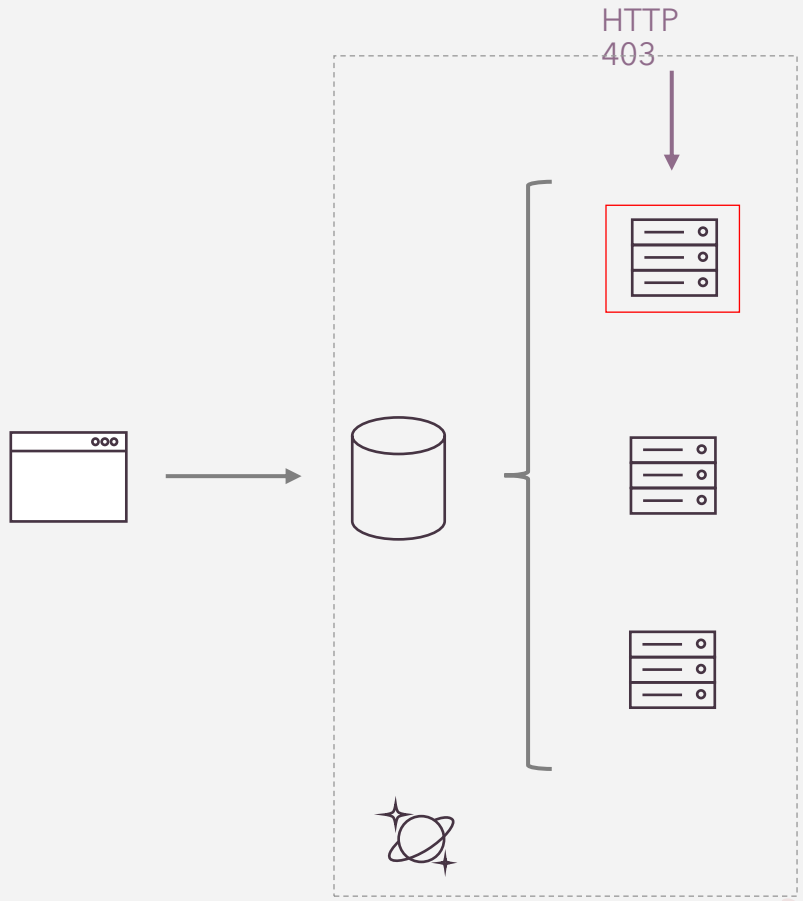
Containers support unlimited storage by dynamically allocating additional physical partitions

Storage for single partition key value (logical partition) is quota'ed to 20GB.

When a partition key reaches its provisioned storage limit, requests to create new resources will return a HTTP Status Code of 403 (Forbidden).

Azure Cosmos DB will automatically add partitions, and may also return a 403 if:

- An authorization token has expired
- A programmatic element (UDF, Stored Procedure, Trigger) has been flagged for repeated violations

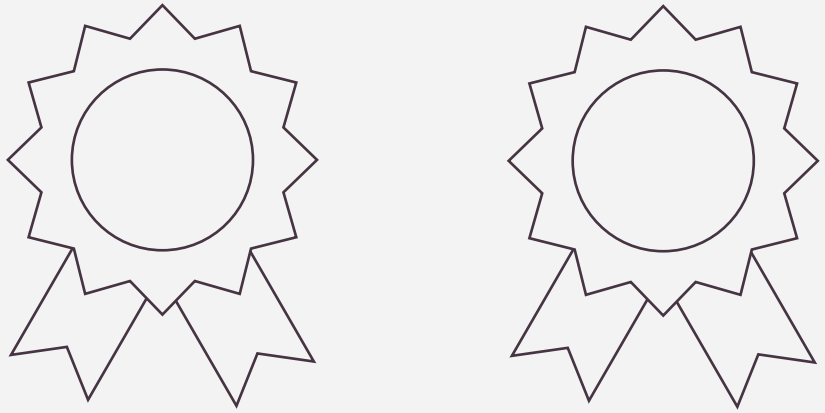


Short-Lifetime Data

Some data produced by applications are only useful for a finite period of time:

- Machine-generated event data
- Application log data
- User session information

It is important that the database system systematically purges this data at pre-configured intervals.



Time-to-Live (TTL)

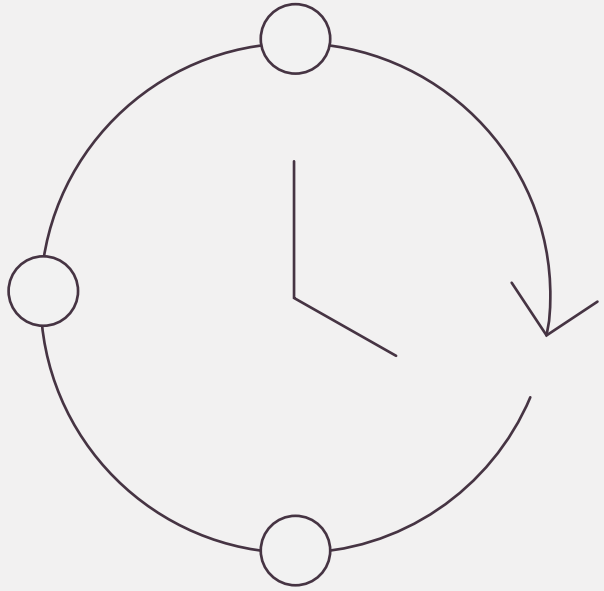
AUTOMATICALLY PURGE DATA

Azure Cosmos DB allows you to set the length of time in which documents live in the database before being automatically purged. A document's "time-to-live" (TTL) is measured in seconds from the last modification and can be set at the collection level with override on a per-document basis.

The TTL value is specified in the _ts field which exists on every document.

- The _ts field is a unix-style epoch timestamp representing the date and time. The _ts field is updated every time a document is modified.

Once TTL is set, Azure Cosmos DB will automatically remove documents that exist after that period of time.



Change Feed

Persistent log of records within an Azure Cosmos DB container. Presented in the order in which they were modified.

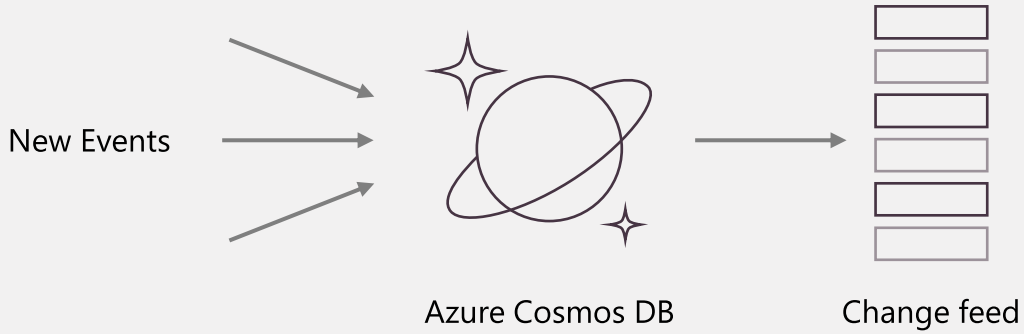
Push Model:

Pushes data to apply business logic for processing. Checking for new data, Storing State handled by Change Feed.

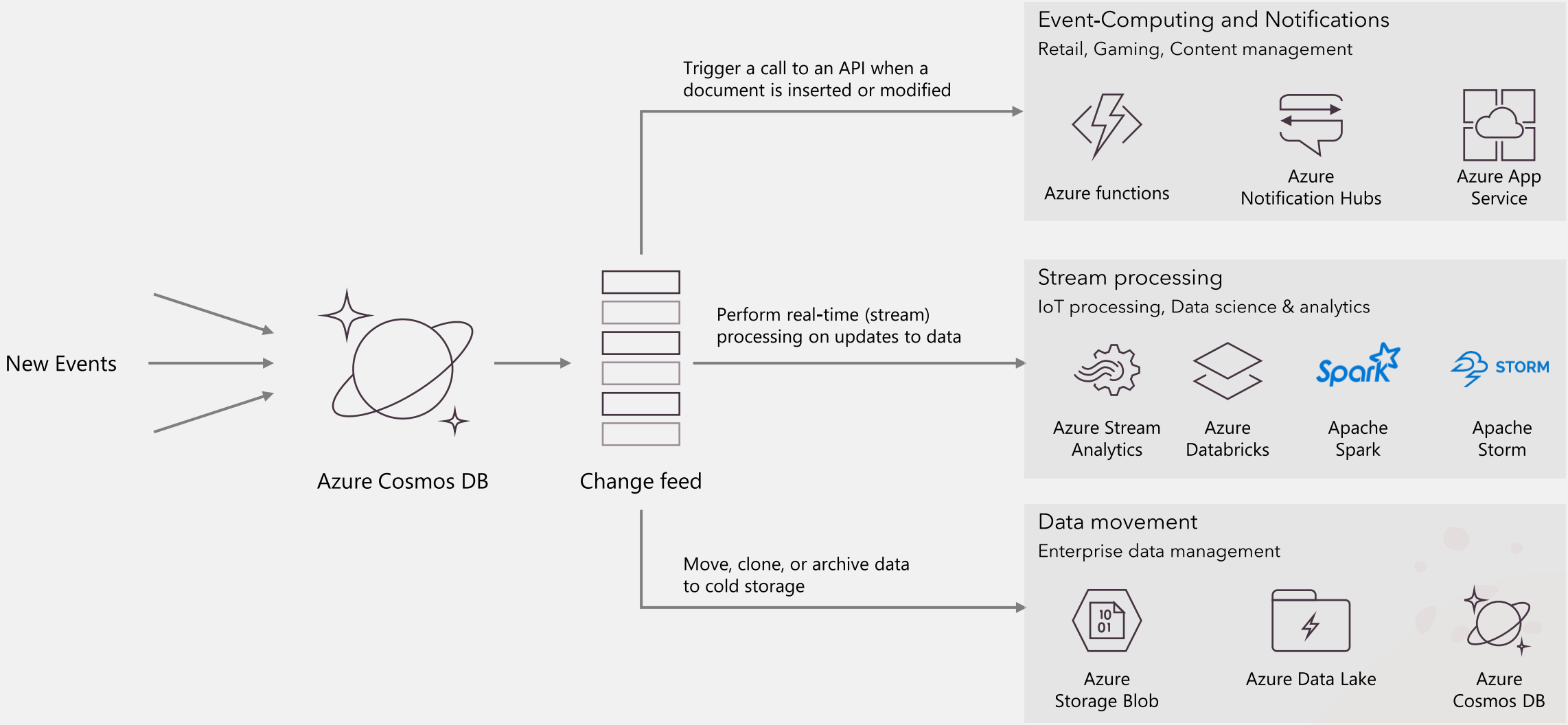
Azure Functions Cosmos DB Triggers and Change Feed Processor Library

Pull Model:

Business logic for processing, handle storing the state, load balancing across multiple clients.



Change Feed Scenarios



Analyzing HTTP Responses

RESPONSE STATUS CODES

When a request is unsuccessful, Azure Cosmos DB responds using well-defined HTTP status codes that can provide more detail into exactly why a specific request failed.

RESPONSE HEADERS

Azure Cosmos DB uses a variety of HTTP headers to offer insight into the result of requests, error conditions, and useful metadata to perform actions such as:

- Resume request
- Measure RU/s charge associated with request
- Access newly created resource directly.

HTTP response codes	
2xx	Success
4xx	Client Errors
5xx	Server Errors

Identifying RATE Limiting

HTTP RESPONSE STATUS CODE

A rate limited request will return a HTTP status code of **429 (Too Many Requests)**. This response indicates that the container has exceeded provisioned throughput limit.

HTTP RESPONSE HEADER

A **rate limited** request will also have a **x-ms-retry-after-ms** header. This header gives the number of milliseconds your application should wait before retrying the current request.

AUTOMATIC RETRY ON THROTTLE

The SDK automatically retries any throttled requests. This can **potentially create a long-running client-side method** that is attempting to retry throttled requests.

Challenge-1

Deploy Azure Services using a script in our Azure Subscription

Use Incognito or Private window to avoid conflict with your Azure Subscription

Challenge-2

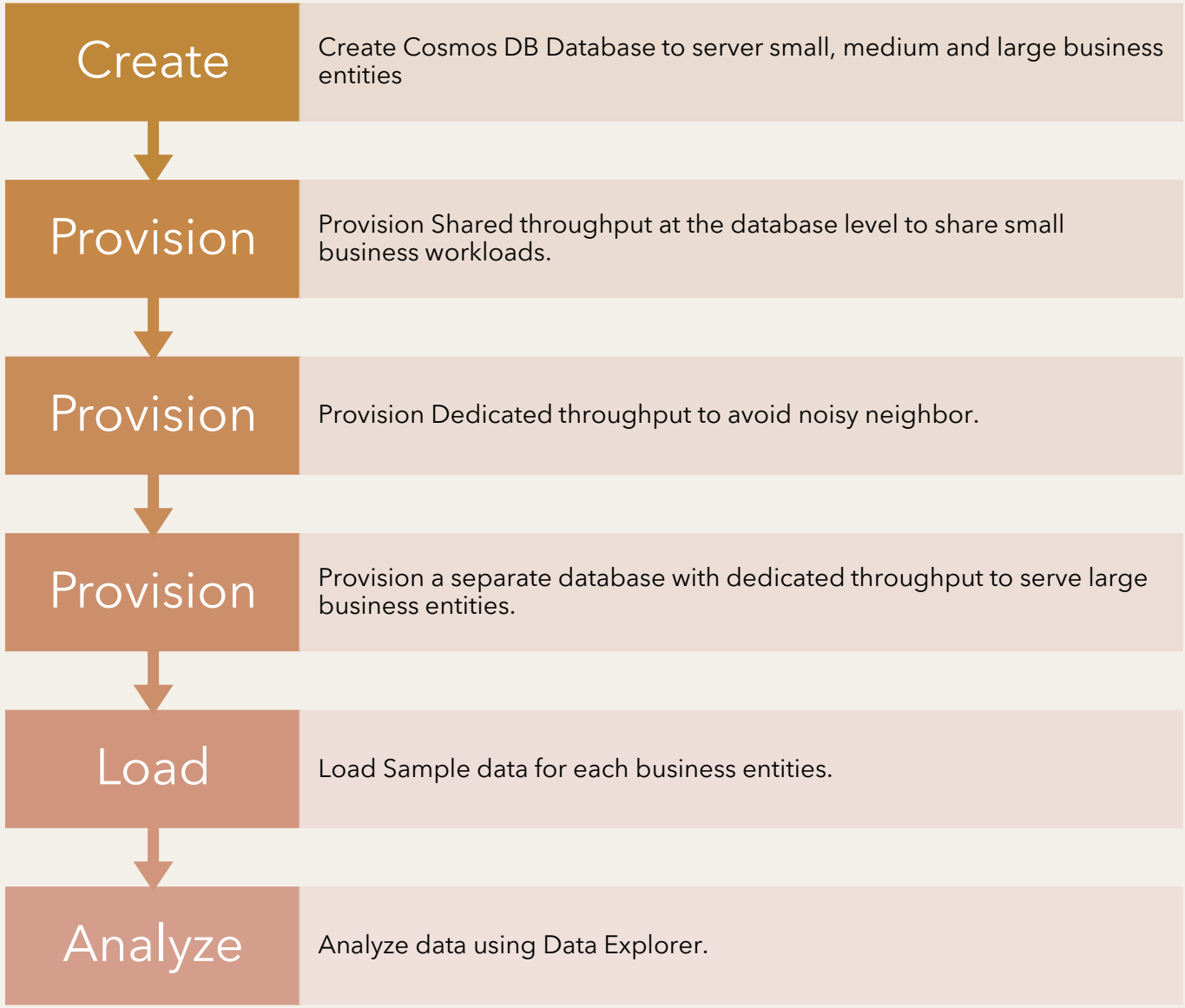
Understand the business use case

Review Software object model for the use case

Review Azure Architecture for building SaaS application

Design Cosmos DB database based on the high-volume access patterns

Challenge-3


Create
Create Cosmos DB Database to server small, medium and large business entities
Provision
Provision Shared throughput at the database level to share small business workloads.
Provision
Provision Dedicated throughput to avoid noisy neighbor.
Provision
Provision a separate database with dedicated throughput to serve large business entities.
Load
Load Sample data for each business entities.
Analyze
Analyze data using Data Explorer.

Challenge-4

Review High Availability with Zone & Region Redundancy

Review Auto Failover Feature

Review Single button Global Replication

Test Sub millisecond Response

Test Autoscale Functionality

Challenge-5

Build a sample application using Quick Start

Run the application from your environment

Modify Cosmos DB database using the application

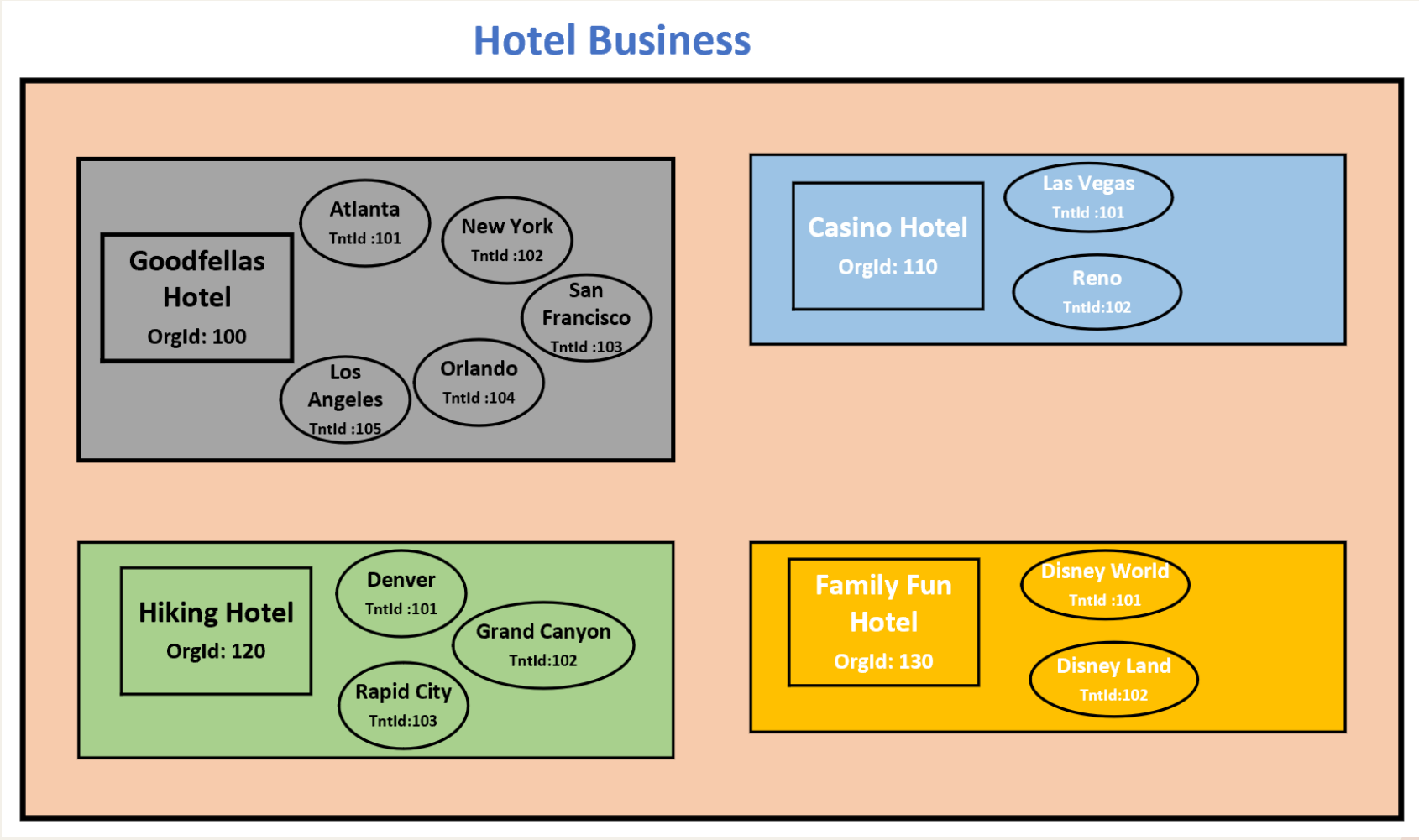
Download Cosmos DB Emulator for local development

Test your Cosmos DB application in your local environment without connecting to Azure

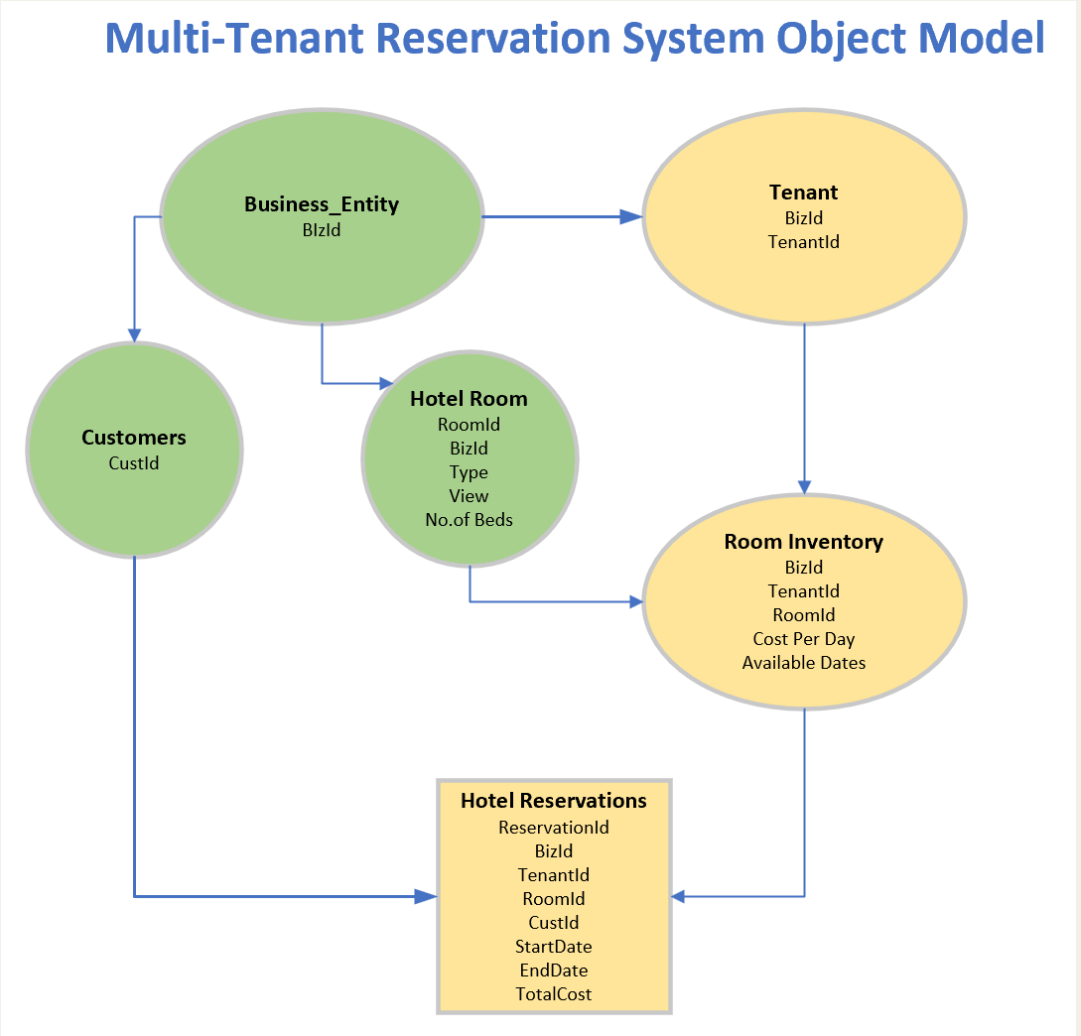
Business Use Case & Data Model Design

Challenge-2

Business Use Case



Software Object Model



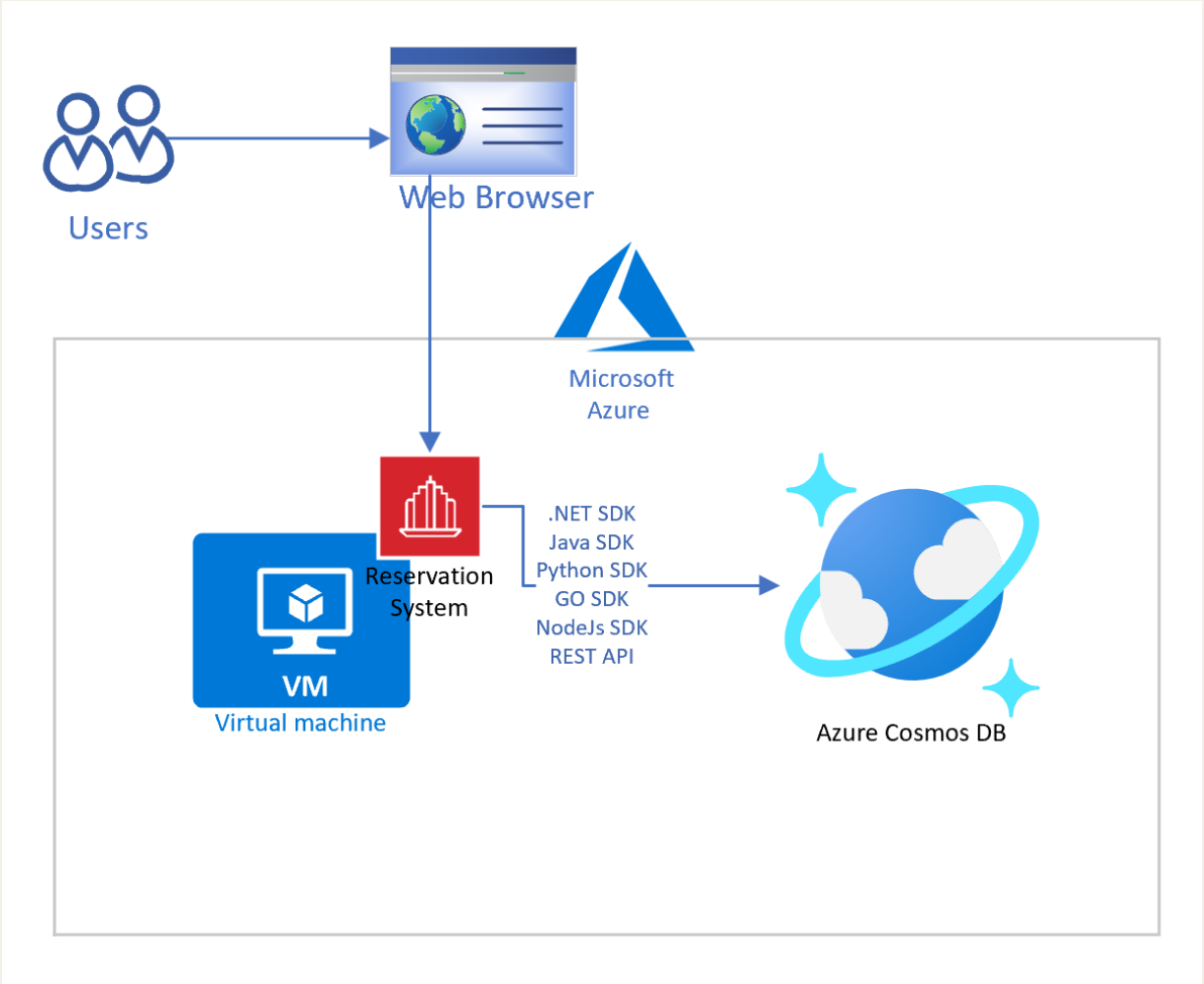
High Volume Access Patterns

Search Functionality to find rooms for a specific location and dates using a business entity website.

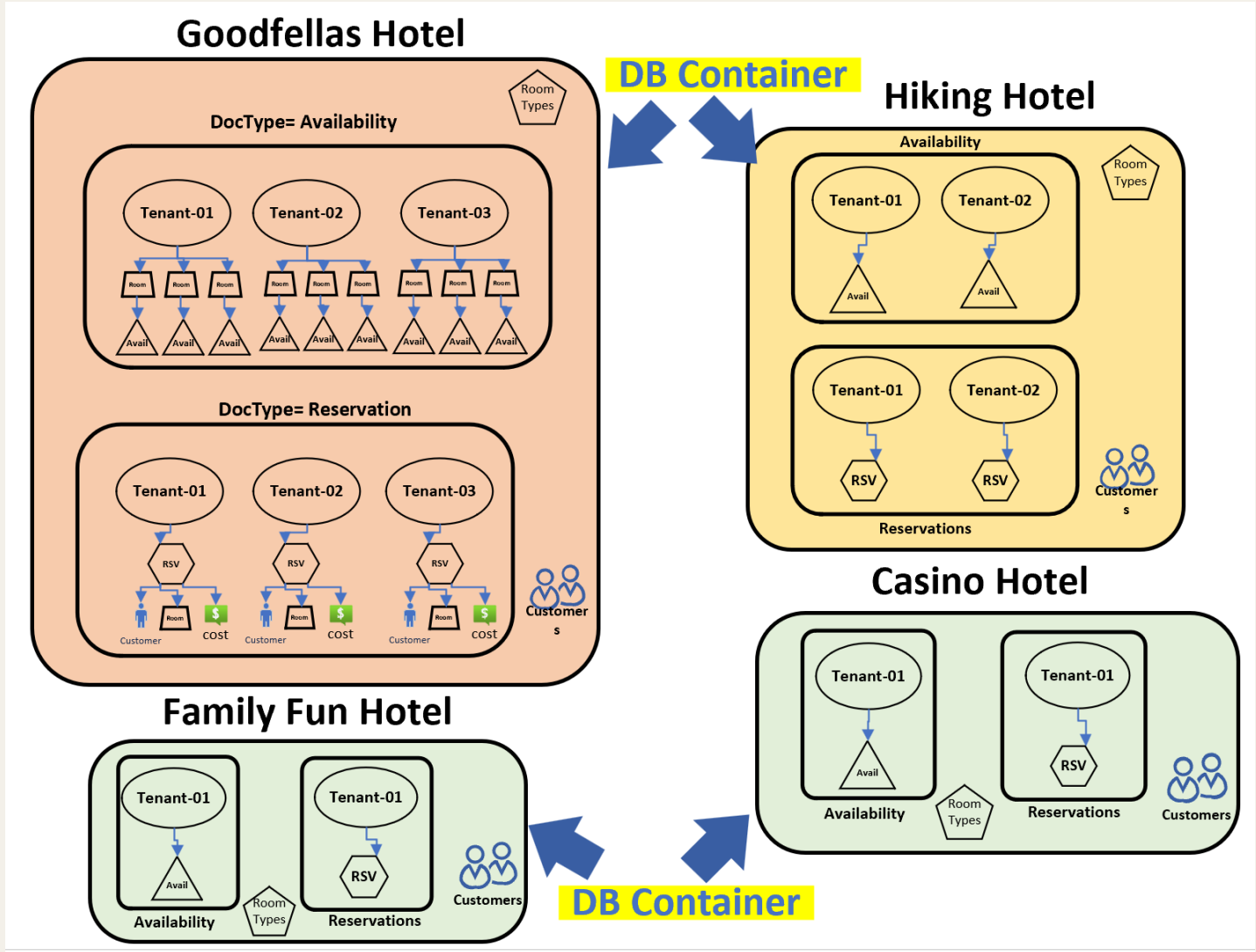
Complete Hotel Reservation transaction and update the room availability at the same time.

Customers & business associates should be able to review, update and delete reservations.

Architecture



Cosmos DB Multi-Tenant Data Model



Microsoft