Azure Cosmos DB

Globally distributed, multi-model database service

Dirk Johann, Cloud Solution Architect @ Microsoft

23.01.2018

Developing planet-scale apps comes with planet-scale challenges

- **Write accurate, globally distributed apps**
- **Managing and versioning complex schemas**
- **Scaling both throughput and storage based on global demand**
- **Balancing the needs for strong and eventual consistency**
- **Delivering highly-responsive experiences**
- **Ensuring an always-on system**

Put your data where your users are

Introducing Azure Cosmos DB

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

Global distribution

- Available in all Azure regions
- Multi-homing APIs
- Comprehensive SLA
- Manual and automatic failover
- Automatic & synchronous multi-region replication

Automatically replicate all your data around the world – across more regions than Amazon and Google combined

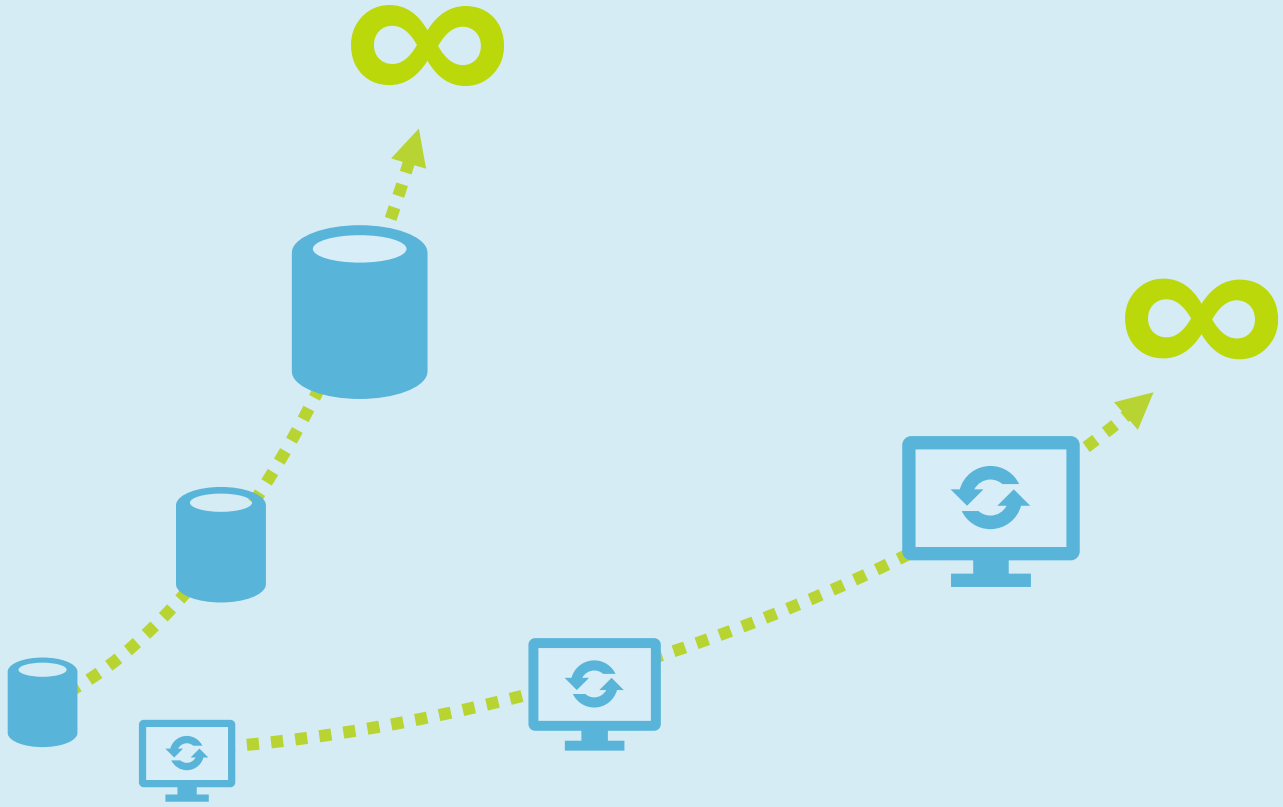
Introducing Azure Cosmos DB

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

Global distribution

Elastic scale-out

Independently and elastically scale storage and throughput across regions



Introducing Azure Cosmos DB

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

- Global distribution
- Elastic scale-out
- **Guaranteed single-digit latency**

Serve <10 ms read and <15 ms write requests at the 99th percentile from the nearest region while delivering data globally

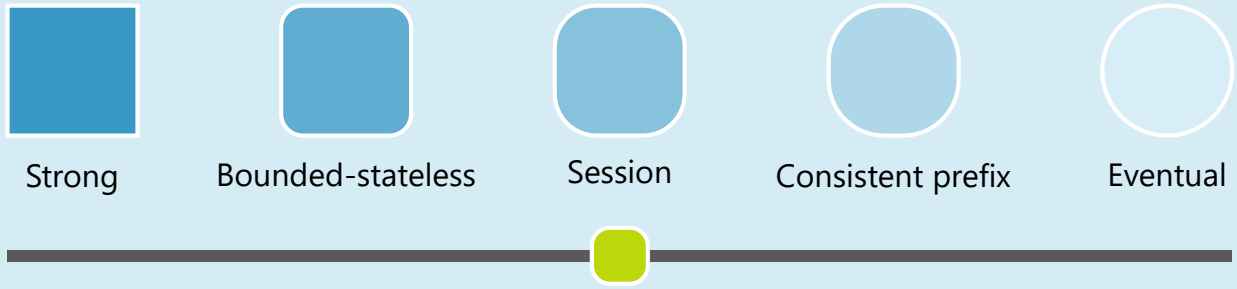
Guaranteed global millisecond latency at the 99th percentile

Introducing Azure Cosmos DB

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

- Global distribution
- Elastic scale-out
- Guaranteed single-digit latency
- **Choice of consistency**

Choose from five defined consistency levels for low latency and high availability

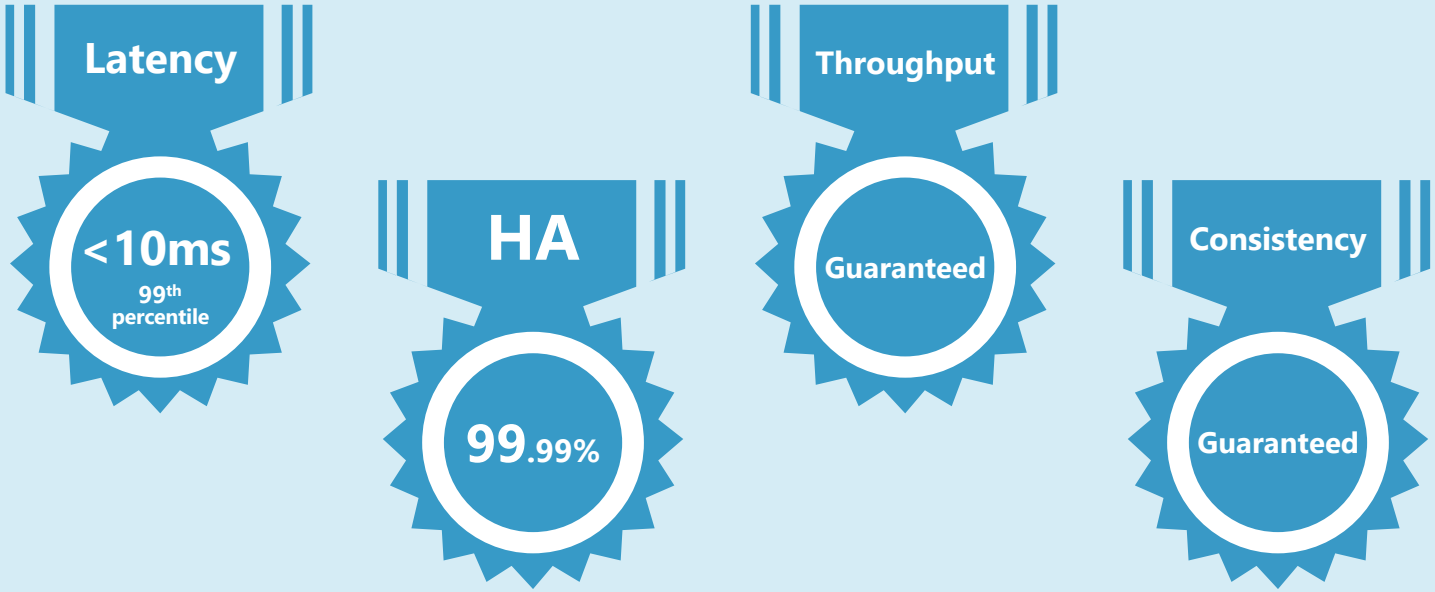


Introducing Azure Cosmos DB

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

- Global distribution
- Elastic scale-out
- Choice of consistency
- Guaranteed single-digit latency
- **Enterprise-level SLAs**

Only service with financially-backed SLAs for millisecond latency at the 99th percentile, 99.99% HA and guaranteed throughput and consistency



Introducing Azure Cosmos DB

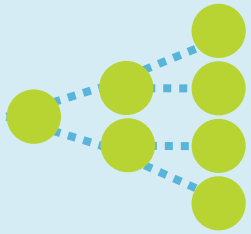
A globally distributed, massively scalable, multi-model database service

- Global distribution
- Elastic scale-out
- Guaranteed single-digit latency
- Choice of consistency
- Enterprise-level SLAs
- **Multi-model + multi API**

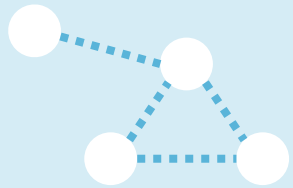
Use key-value, graph, and document with a schema-agnostic service that doesn't require any schema or secondary indexes

KEY-VALUE

COLUMN-FAMILY



DOCUMENT



GRAPH

Introducing Azure Cosmos DB

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

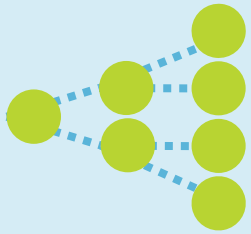
Multi-model + multi API

- Database engine operates on atom-record-sequence (ARS) based type system
  - All data models are efficiently translated to ARS
- API and wire protocols are supported via extensible modules
- Instance of a given data model can be materialized as trees
- Graph, documents, key-value, column-family, … *more to come*

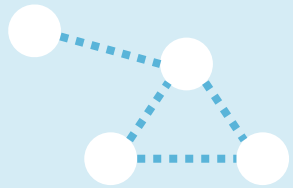
Use key-value, graph, and document with a schema-agnostic service that doesn't require any schema or secondary indexes

KEY-VALUE

COLUMN-FAMILY



DOCUMENT



GRAPH

Introducing Azure Cosmos DB

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

Multi-model + multi API

Cosmos DB offers a multitude of APIs to access and query data including, SQL and various popular OSS APIs.

Table API

SQL

Schema-agnostic, automatic indexing

- At global scale, schema/index management is hard
- Automatic and synchronous indexing of all ingested content - hash, range, geo-spatial, and columnar
- No schemas or secondary indices ever needed
- Resource governed, write optimized database engine with latch free and log structured techniques
- Online and in-situ index transformations
- While the database is fully schema-agnostic, schema-extraction is built in
- Customers can get Avro schemas from the database

Security & Compliance

Enterprise grade security

Encryption at Rest

- Always encrypted at rest and in motion
- Data, index, backups, and attachments encrypted

Encryption is enabled automatically by default

- No impact on performance, throughput or availability
- Transparent to your application

Comprehensive Azure compliance certification

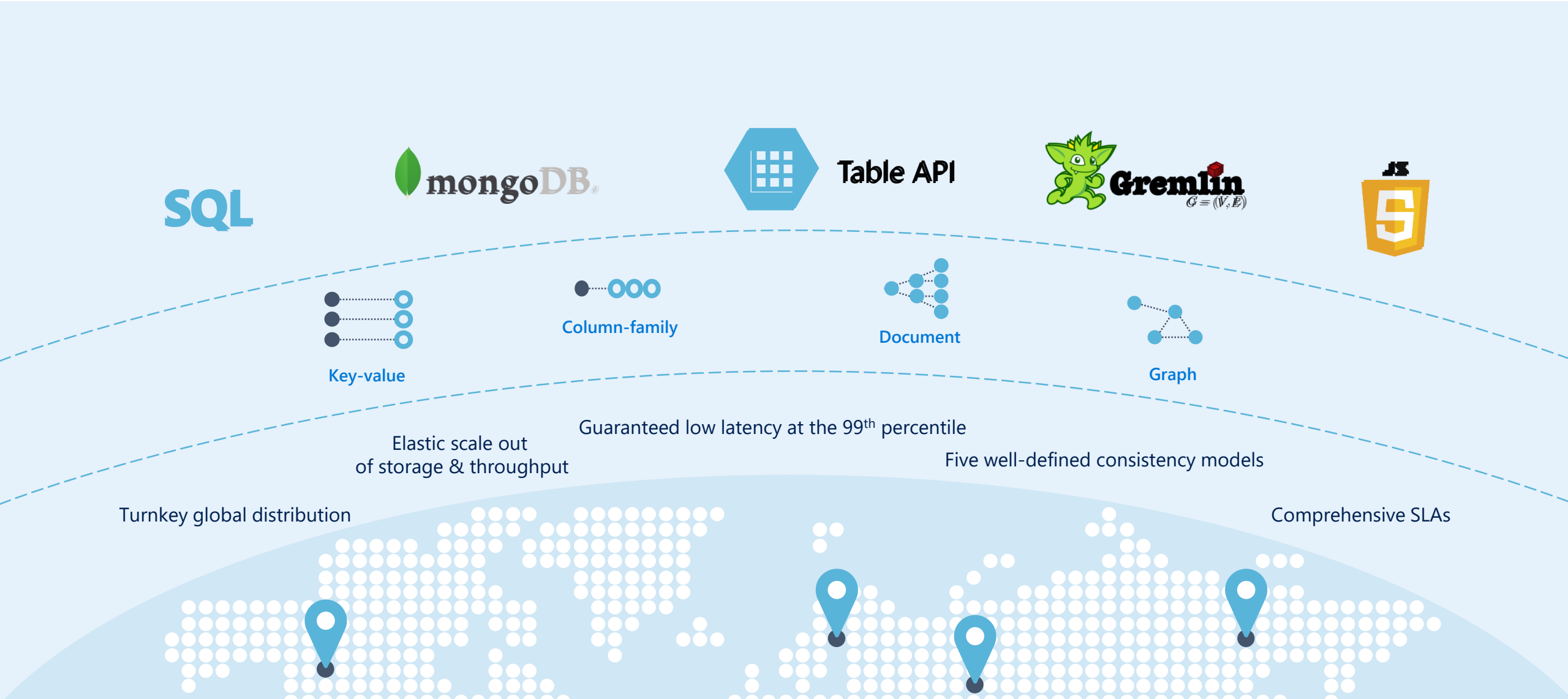
- ISO 27001, ISO 27018, EUMC, HIPAA, PCI
- SOC1 and SOC2 (Audit complete, Certification in Q2 2017)
- FedRAMP, IRS 1075, UK Official (IL2) (Q2 2017)
- HITRUST (H2 2017)

Azure Cosmos DB: Value to Customer

- Save money
- Become more productive
- Become more flexible
- Become more responsive
- Become more innovative

Introducing Azure Cosmos DB

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



Azure Cosmos DB Evolution

- Originally started to address the problems faced by large scale apps inside Microsoft
- Built from the ground up for the cloud
- Started as a MongoDB NoSQL compete
- Now going after everything and anything scale

2010 — Project Florence

2014 — DocumentDB

2015

2017 — Cosmos DB

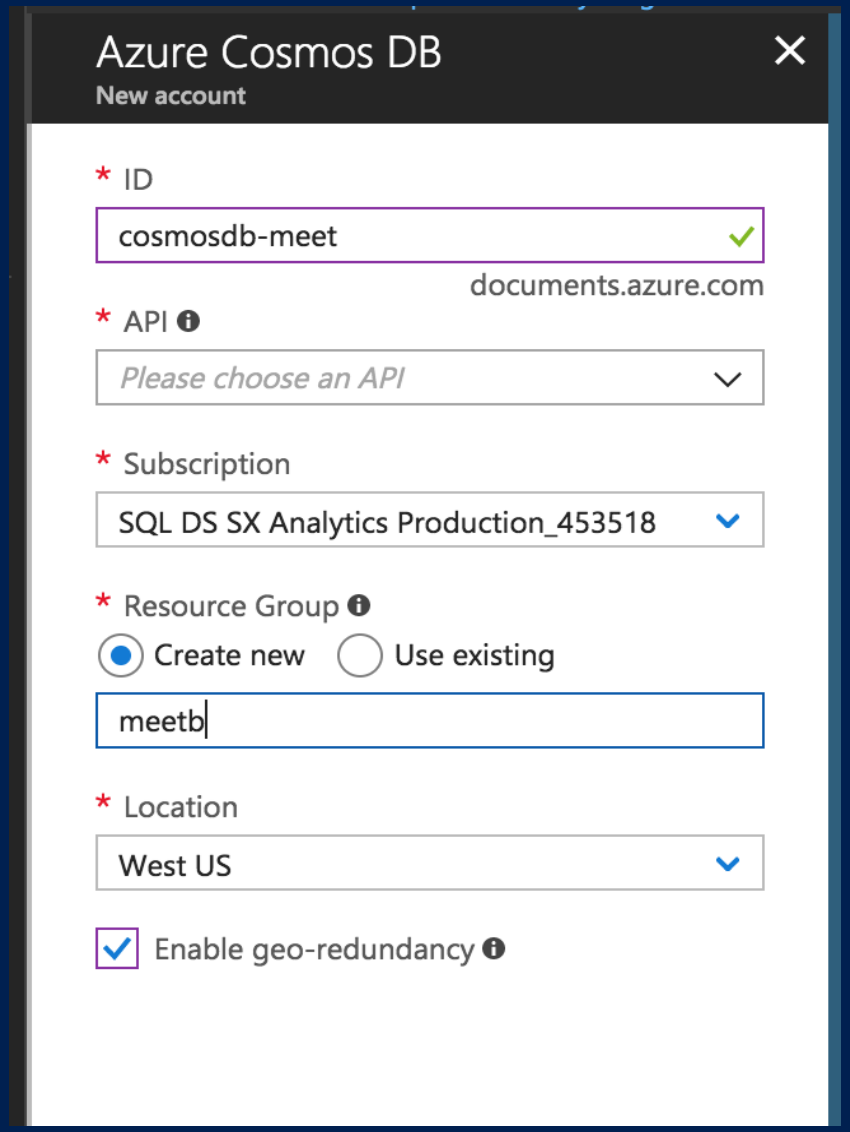
What is CosmosDB?

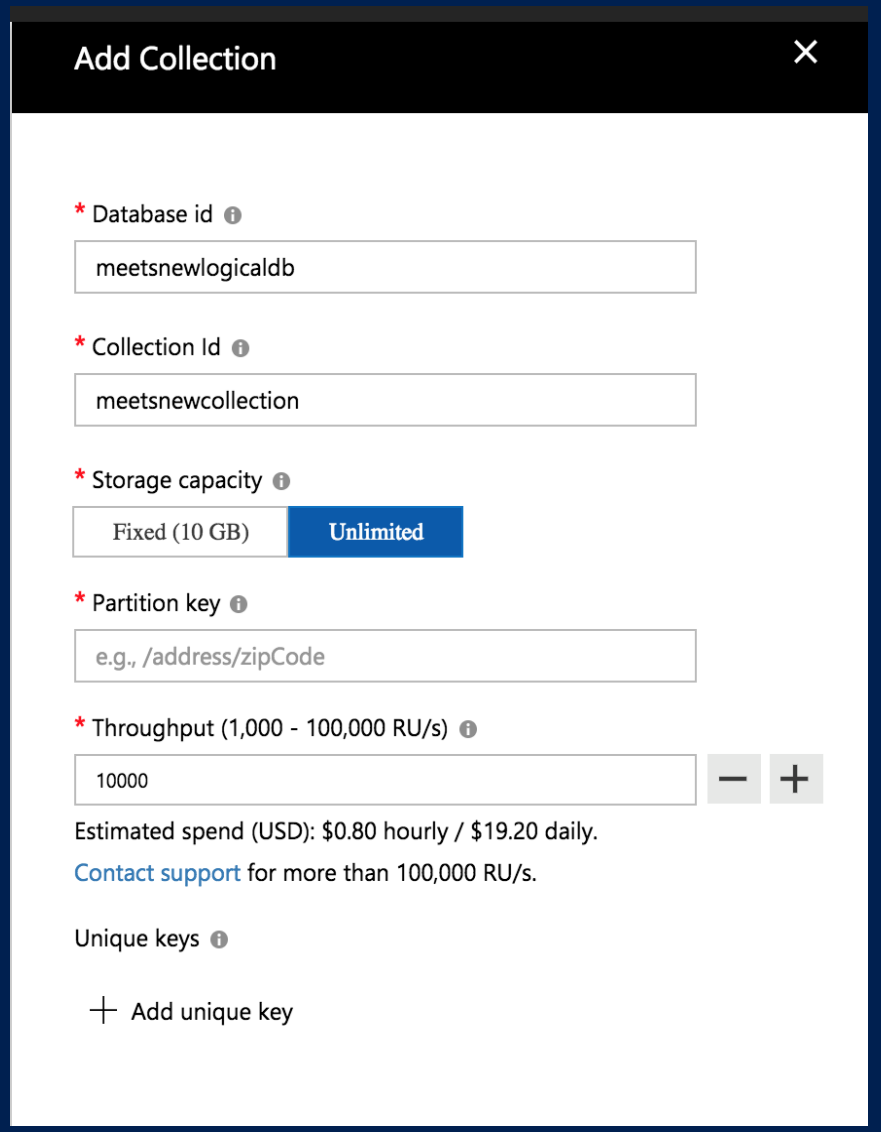
Microsoft's proprietary globally-distributed, multi-model database service "for managing data at planet-scale"

CosmosDB promises

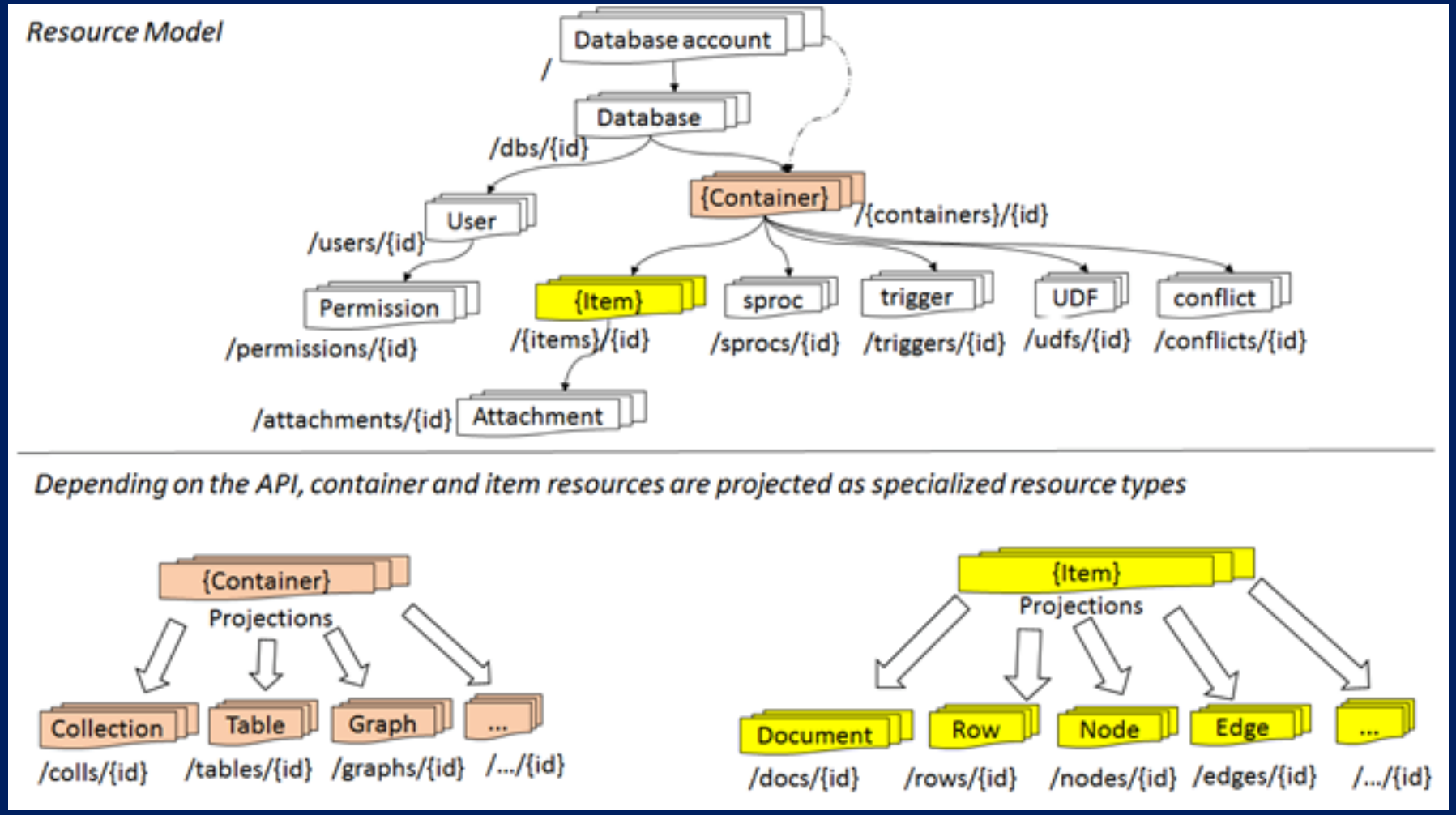
- Turnkey global distribution
- Guaranteed Single Digit ms Latency at the 99th Percentile
- Unlimited Elastic Scaleout
- Choice of Consistency
- Schema-agnostic, automatic indexing
- Multiple Models + APIs

Portal Experience





Terminology



Turnkey global distribution

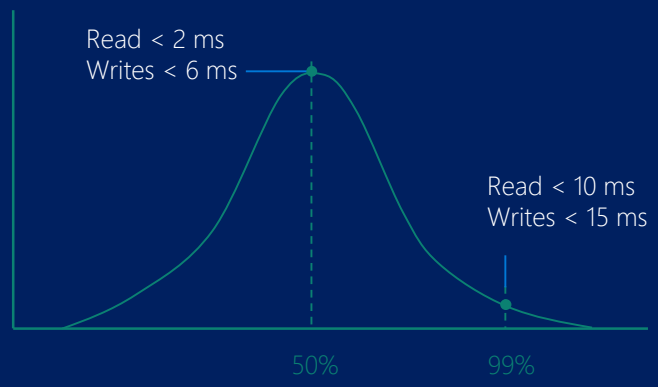
Global Distribution From The Ground-Up

- Transparent and automatic multi-region replication
  - Associate *any* number of regions with your database account, *at any time*
  - Policy based geo-fencing
- Multi-homing APIs
  - All endpoints are logical, by default
  - Apps don't need to be redeployed during regional failover
  - Apps can also access physical endpoints if needed
- Support for both manual and automatic failover
- Designed for high availability
  - Allows for dynamically setting *priorities* to regions
  - *Simulate* regional disasters via API
  - Test the *end-to-end availability* for the entire app (beyond just the database)

Guaranteed Single Digit ms Latency at the 99th Percentile

Guaranteed Low Latency

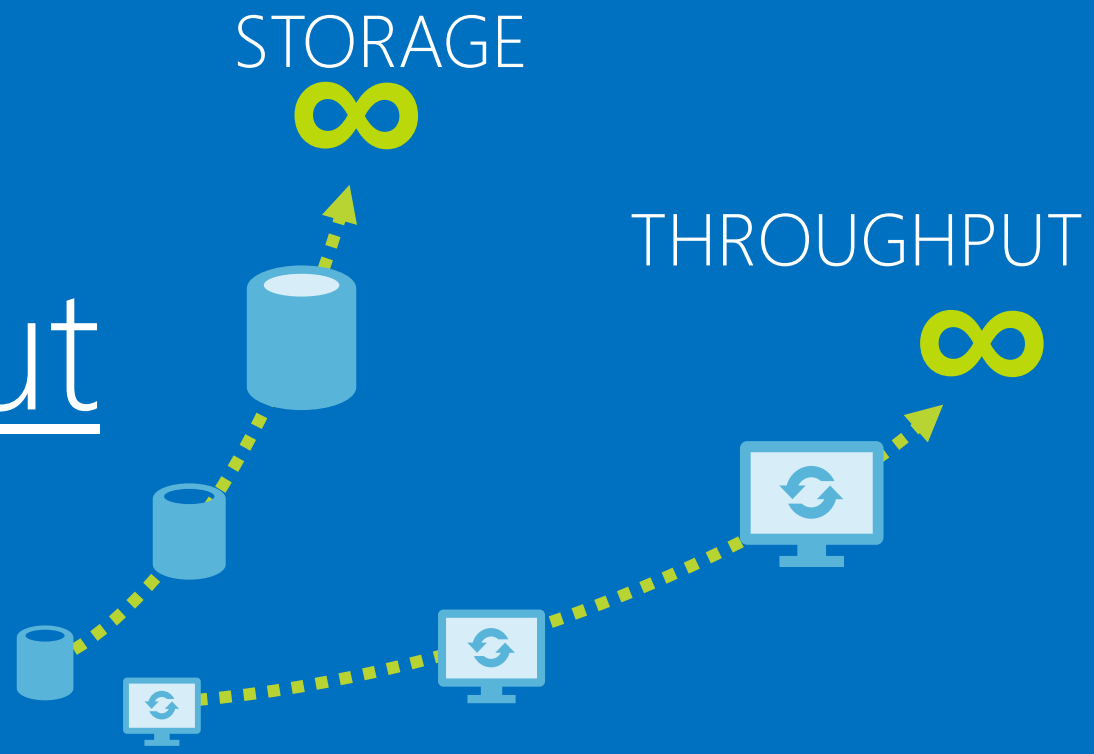
- Reads and writes served from local region
- Guaranteed millisecond latency worldwide
- Write optimized, latch-free database engine
- Automatically indexed SSD storage
- Synchronous and automatic indexing at sustained ingestion rates



	Reads (1KB)	Indexed writes (1KB)
50th	<2ms	<6ms
99th	<10ms	<15ms

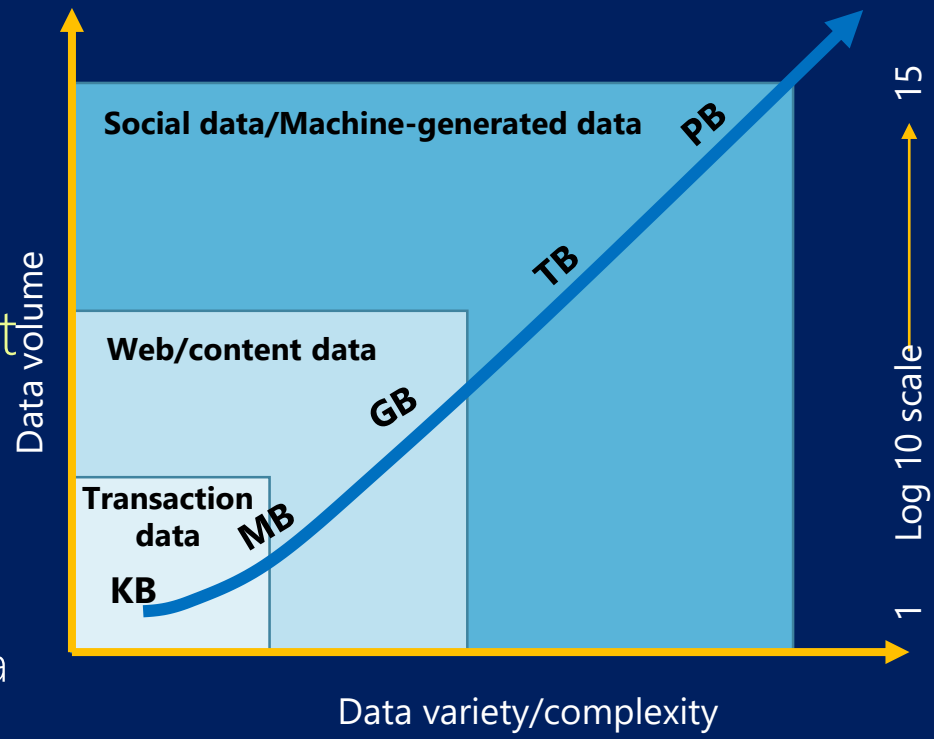
3

Elastic Scaleout



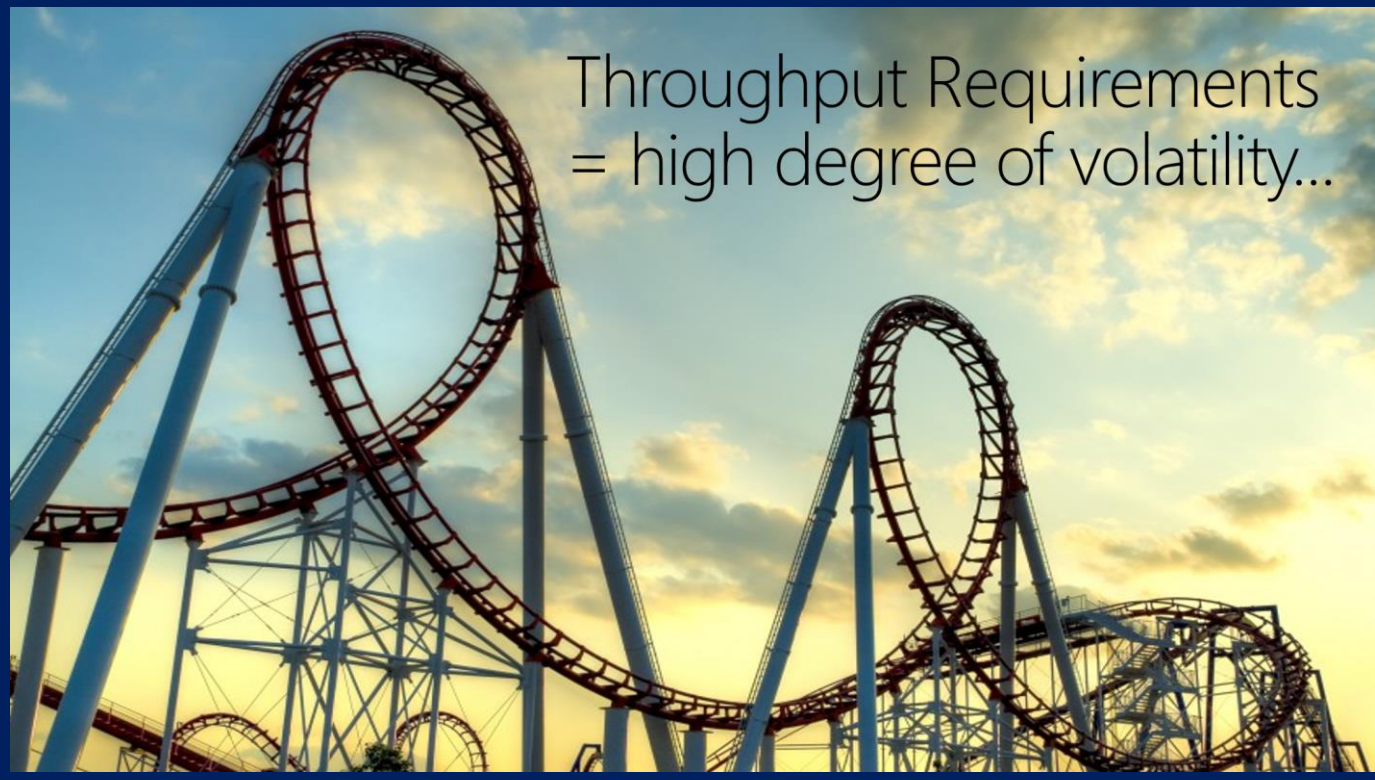
Azure Cosmos DB: Elastically Scalable Storage

- Single machine is never a bottleneck
- A single table can scale from GB-PBs, across many machines, and regions
- Semi Transparent server side partition management and routing
- Optionally evict old data using built-in support for TTL
  - Policy based, automatic tiering to any HDFS compatible data lake (e.g. ADLS or Azure Storage)
- Customers pay only for the throughput and storage they need

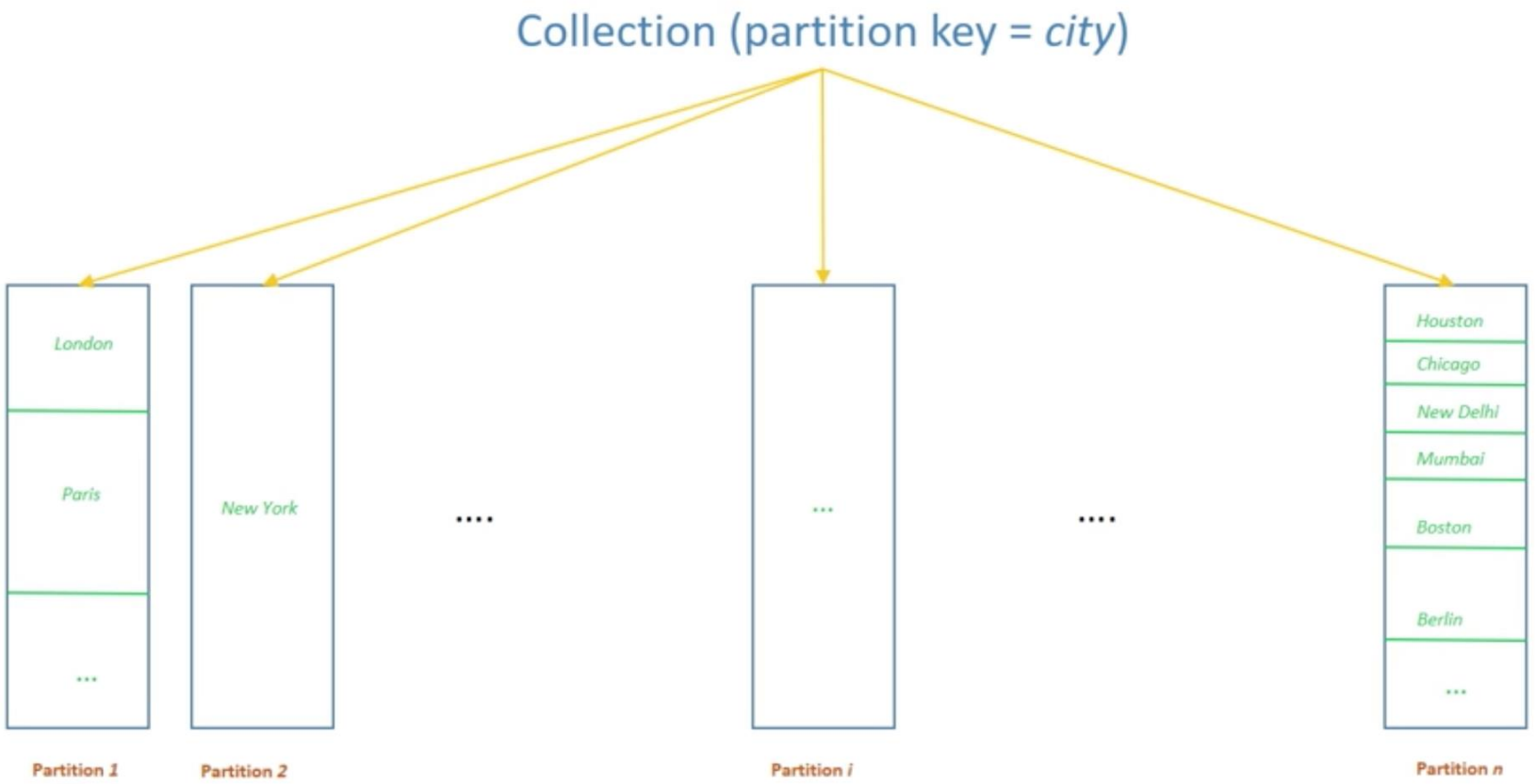


Azure Cosmos DB: Elastically Scalable Throughput

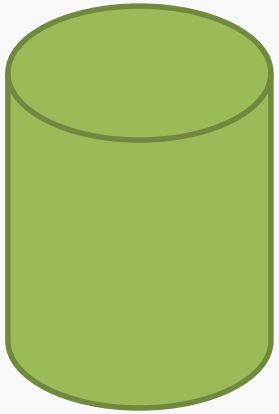
- Elastically scale throughput from 10 to 100s of millions of requests/sec across multiple regions
- Support for requests/sec for different workloads
  - This ensures that never have to provision for the peak
- Customers pay only for the throughput and storage they need
- Customers pay by the hour for the provisioned throughput



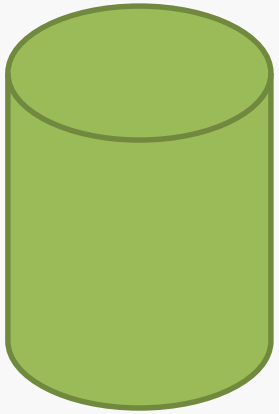
What are partitions?

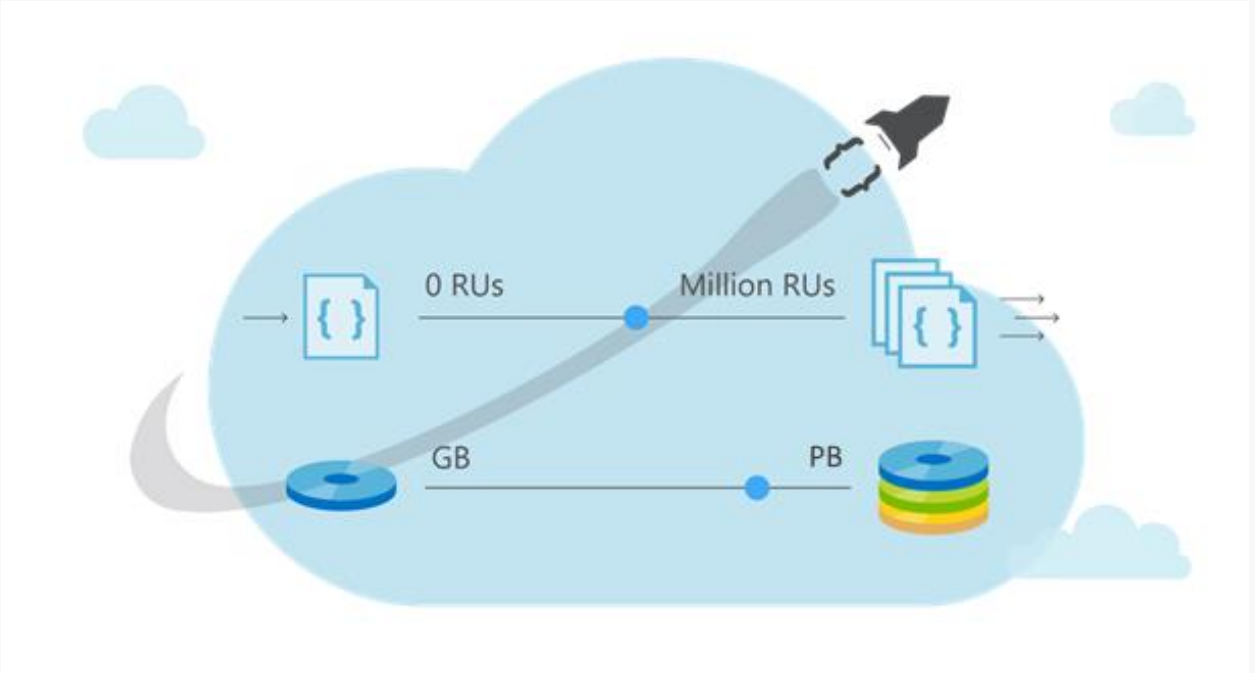


Cosmos DB Container (e.g. Collection)

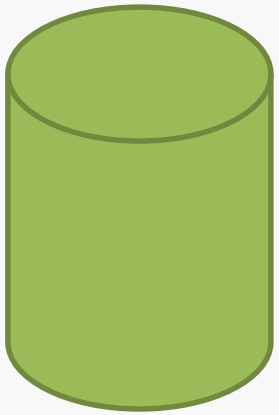


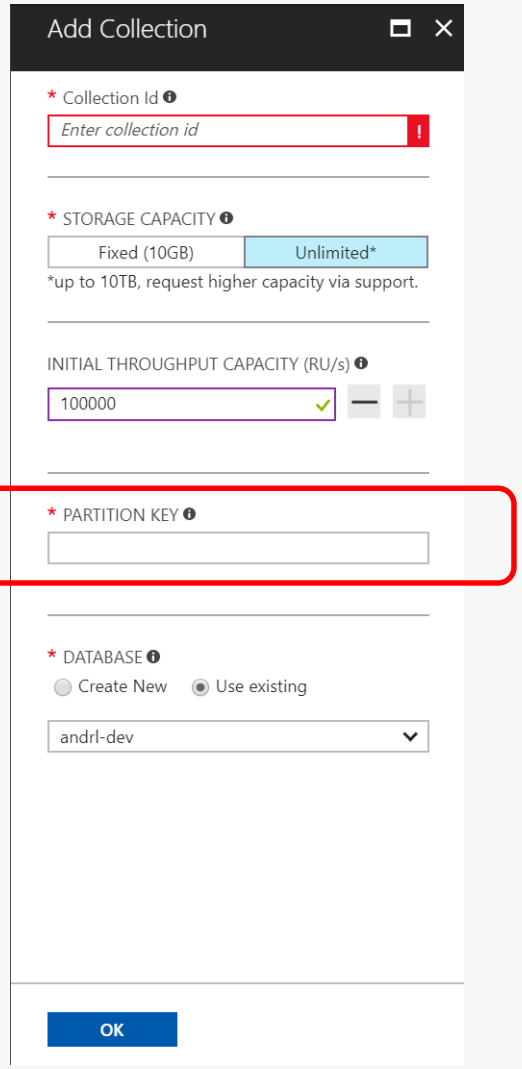
Cosmos DB Container (e.g. Collection)





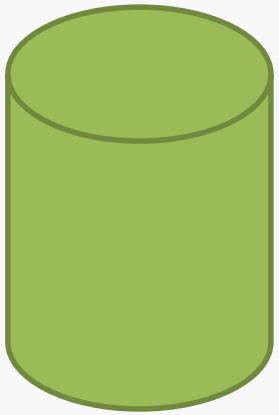
Cosmos DB Container (e.g. Collection)





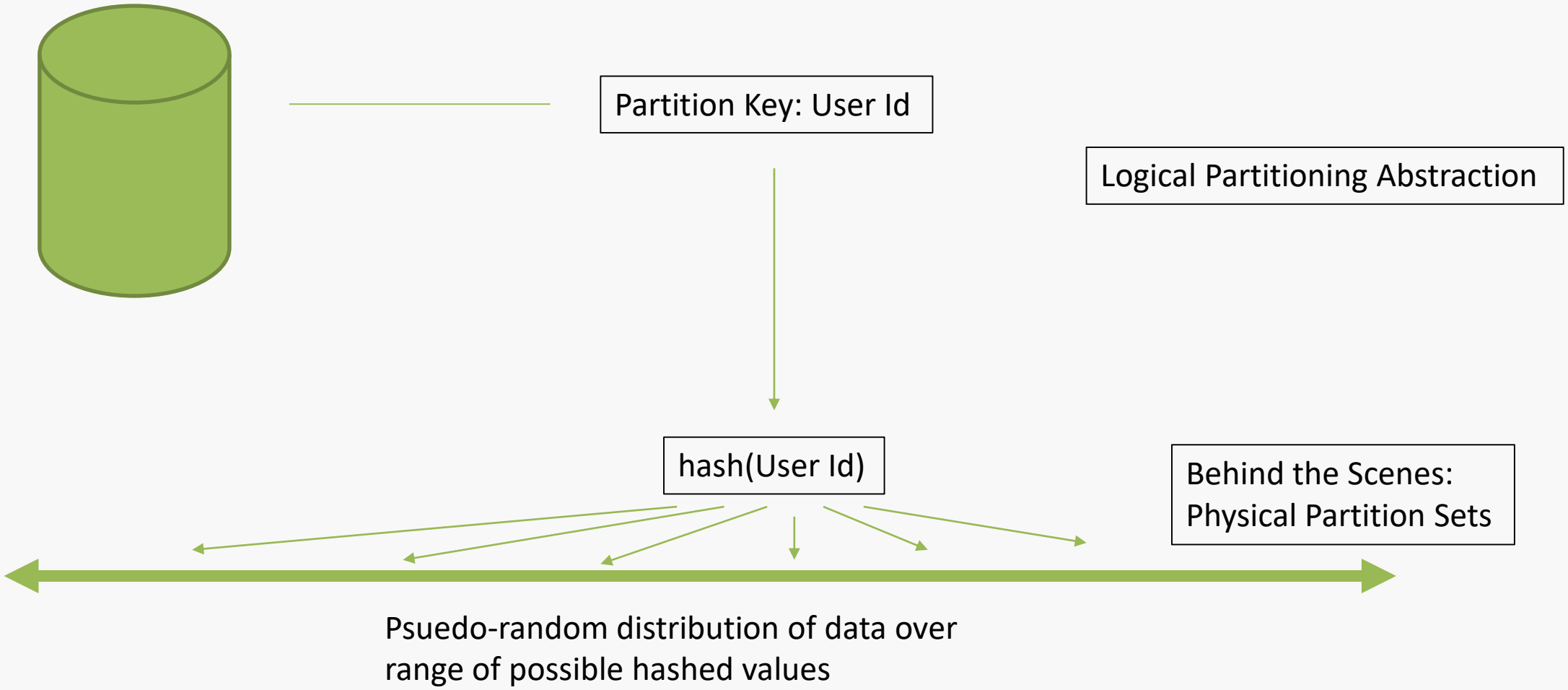
Partitioning Scheme: top-most design decision in Cosmos DB

Cosmos DB Container (e.g. Collection)

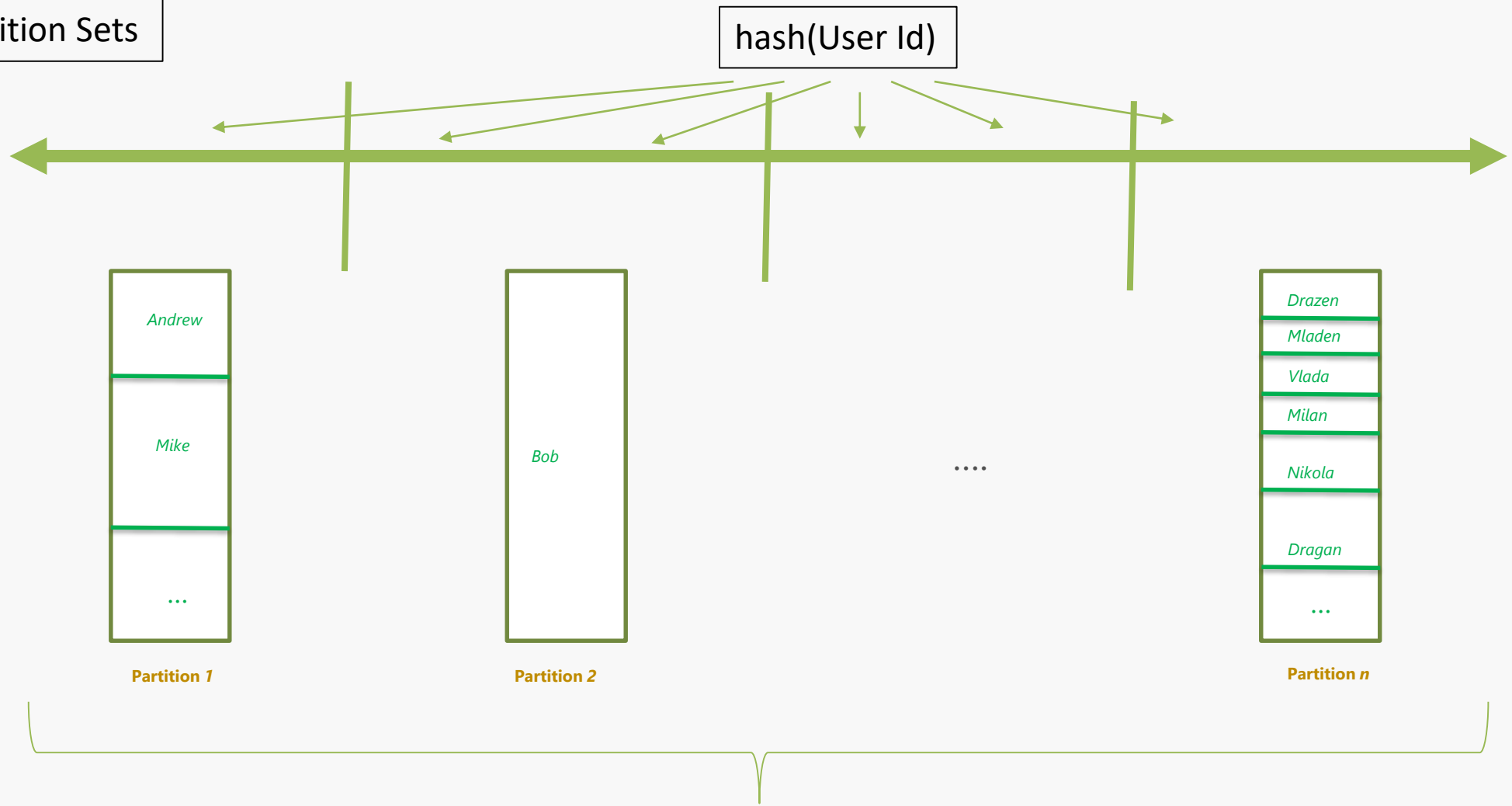


Partition Key: User Id

Cosmos DB Container (e.g. Collection)

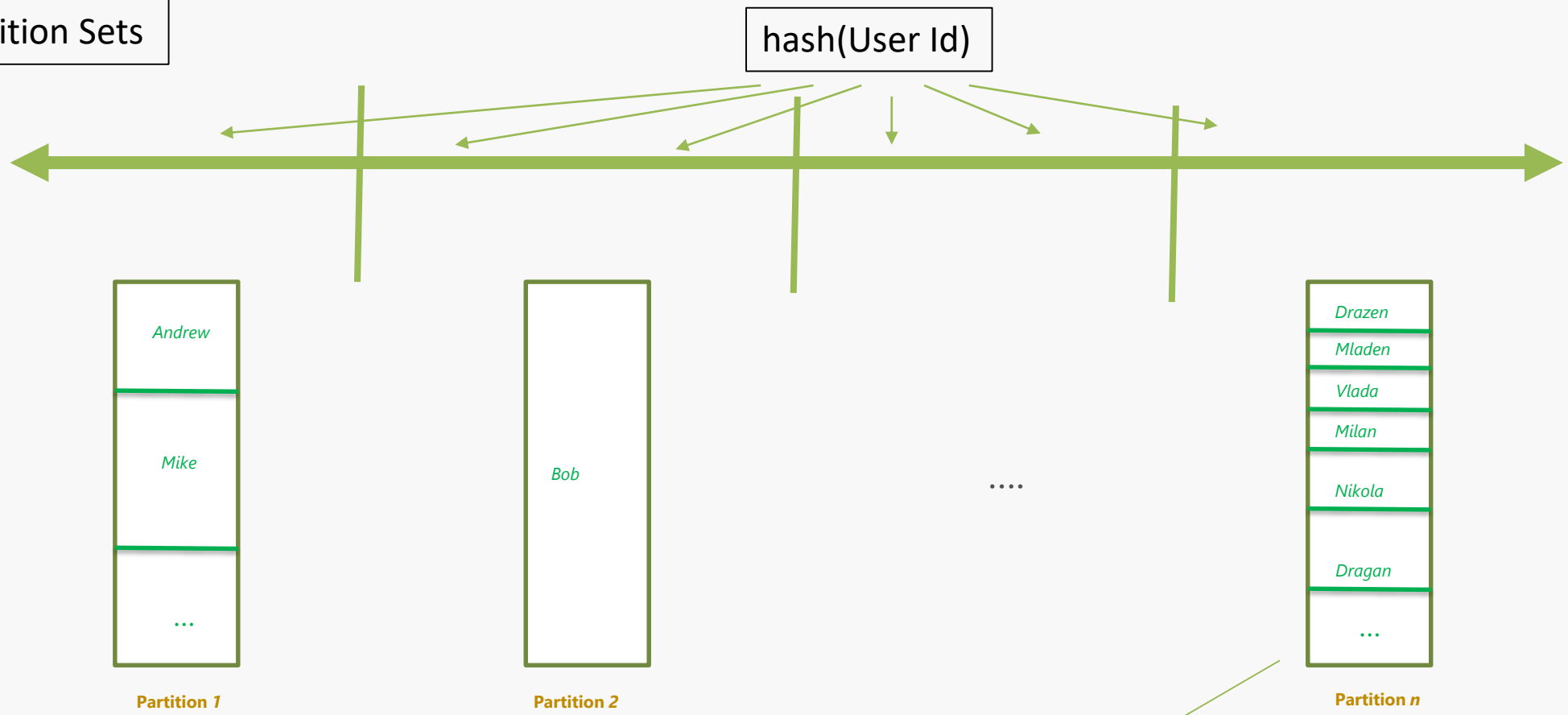


Behind the Scenes:
Physical Partition Sets



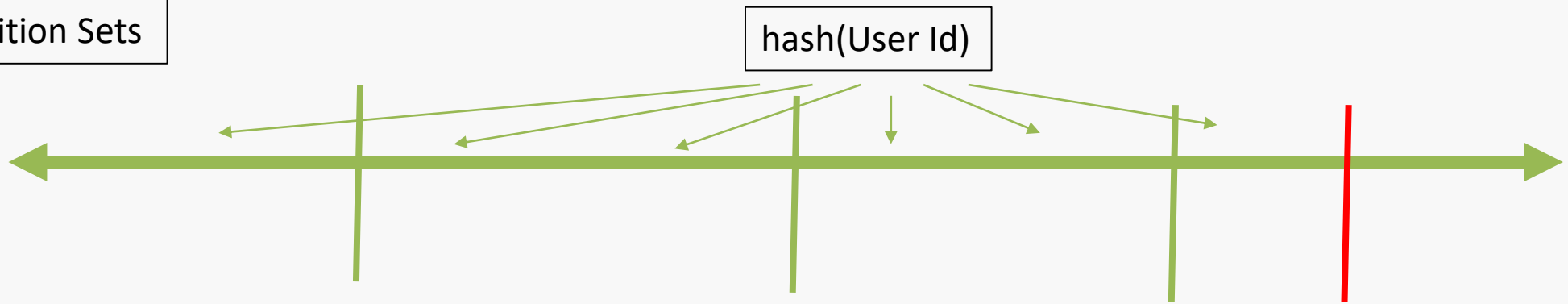
Frugal # of Partitions based on actual storage and throughput needs
(yielding scalability with low total cost of ownership)

Behind the Scenes: Physical Partition Sets

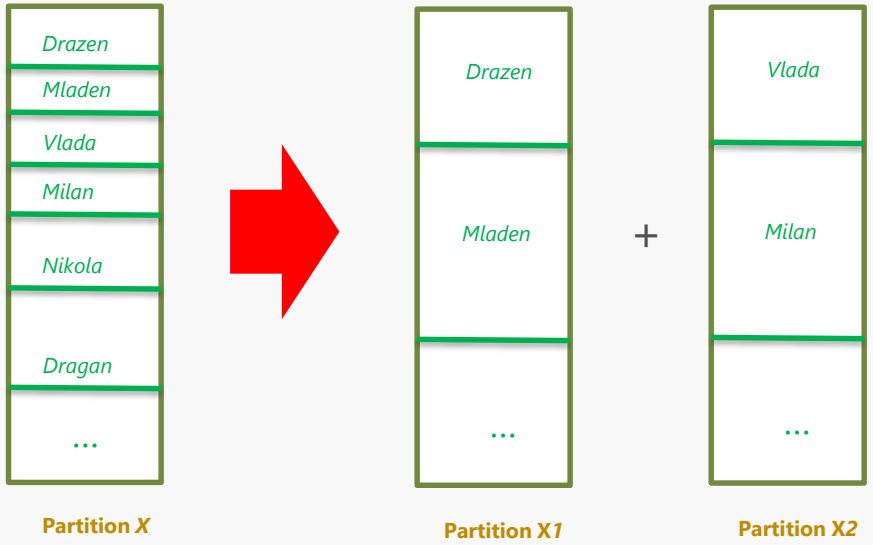


What happens when partitions need to grow?

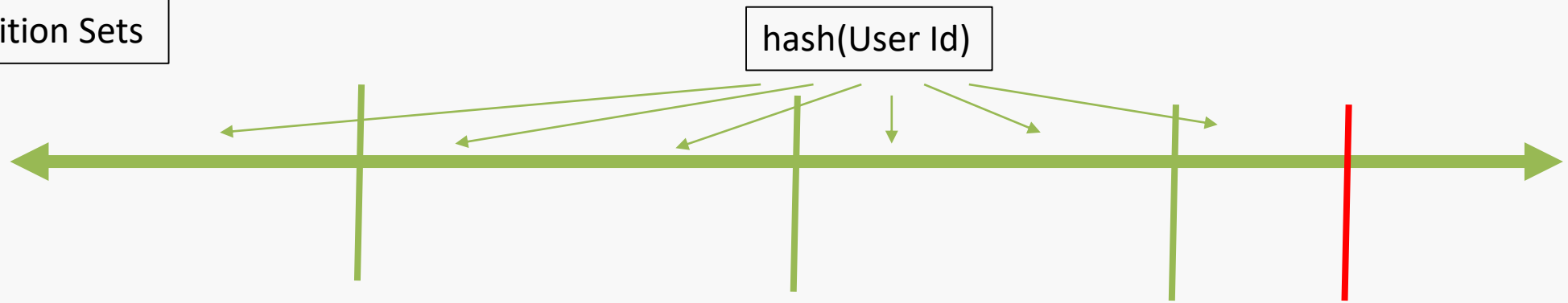
Behind the Scenes:
Physical Partition Sets



Partition Ranges can be dynamically sub-divided
To seamlessly grow database as the application grows
While sedulously maintaining high availability

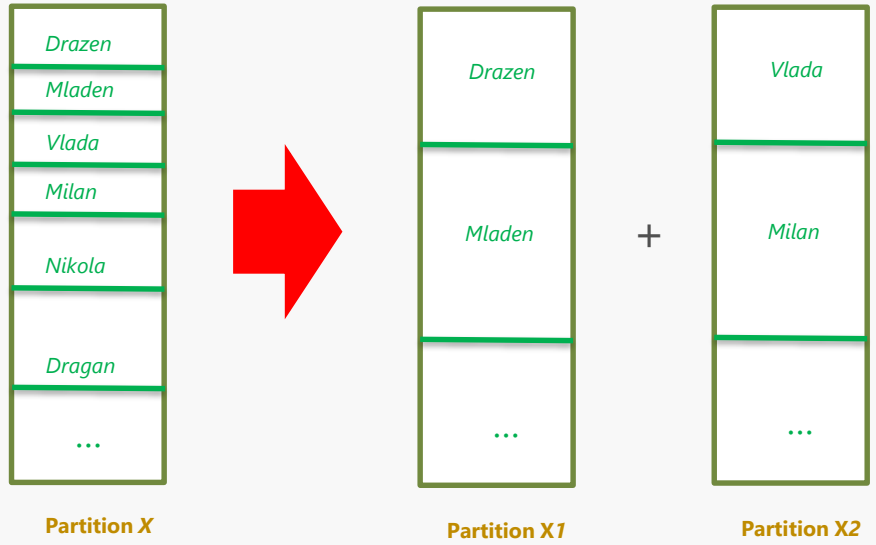


Behind the Scenes:
Physical Partition Sets

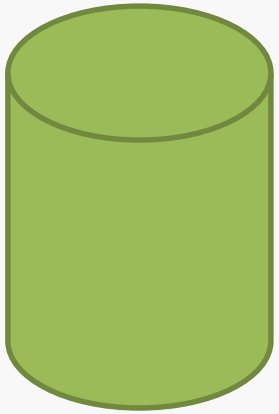


Partition Ranges can be dynamically sub-divided
To seamlessly grow database as the application grows
While sedulously maintaining high availability

Best of All:
Partition management is completely taken care of by the system
You don't have to lift a finger… the database takes care of you.



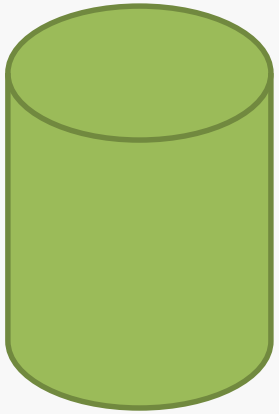
Cosmos DB Container (e.g. Collection)



Best Practices: Design Goals for Choosing a Good Partition Key

1) Distribute the overall request + storage volume
   - Avoid “hot” partition keys
2) Partition Key is scope for [efficient] queries and transactions
   - Queries can be intelligently routed via partition key
   - Omitting partition key on query requires fan-out

Cosmos DB Container (e.g. Collection)



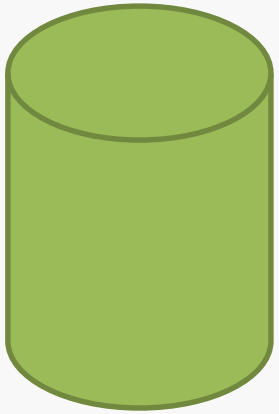
Best Practices: Design Goals for Choosing a Good Partition Key

1) Distribute the overall request + storage volume
   - Avoid “hot” partition keys
2) Partition Key is scope for [efficient] queries and transactions
   - Queries can be intelligently routed via partition key
   - Omitting partition key on query requires fan-out

Steps for Success

1. Ballpark scale needs (size/throughput)
2. Understand the workload
3. # of reads/sec vs writes per sec
   - Use 80/20 rule to help optimize bulk of workload
   - For reads – understand top X queries (look for common filters)
   - For writes – understand transactional needs
     understand ratio of inserts vs updates

Cosmos DB Container (e.g. Collection)



Best Practices: Design Goals for Choosing a Good Partition Key

1) Distribute the overall request + storage volume
   - Avoid “hot” partition keys
2) Partition Key is scope for [efficient] queries and transactions
   - Queries can be intelligently routed via partition key
   - Omitting partition key on query requires fan-out

Steps for Success

1. Ballpark scale needs (size/throughput)
2. Understand the workload
3. # of reads/sec vs writes per sec
   - Use 80/20 rule to help optimize bulk of workload
   - For reads – understand top X queries (look for common filters)
   - For writes – understand transactional needs
     understand ratio of inserts vs updates

Azure Cosmos DB each partition size limit

2

In Azure Cosmos DB partinioned collection, does each partition has any size limit?

As per this old document, they have a size limit of 10 GB. Is that the same now also?

https://azure.microsoft.com/en-in/blog/10-things-to-know-about-documentdb-partitioned-collections/

Regards, Karthikeyan V.

azure azure-cosmosdb

share improve this question

edited Jun 14 '17 at 16:23

lambodar
1,191 ● 1 ● 13 ● 31

asked Jun 14 '17 at 10:14

user2461022
26 ● 1 ● 3

The truly baffling thing is why there is an arbitrary limit at all. It seems like the 21st century version of the apocryphal "640K ought to be enough for anybody" gaffe... – McGuireV10 Nov 20 '17 at 20:33

add a comment

2 Answers

active oldest votes

2

A partitioned collection has individual 10GB partition spaces. For a given partition key, you cannot exceed 10GB of data. This has not changed.

You'll need to pick a partition key which distributes your data across many partitions, vs creating "hot" partitions which could fill up (where you'd then get an error when attempting to write content).

share improve this answer

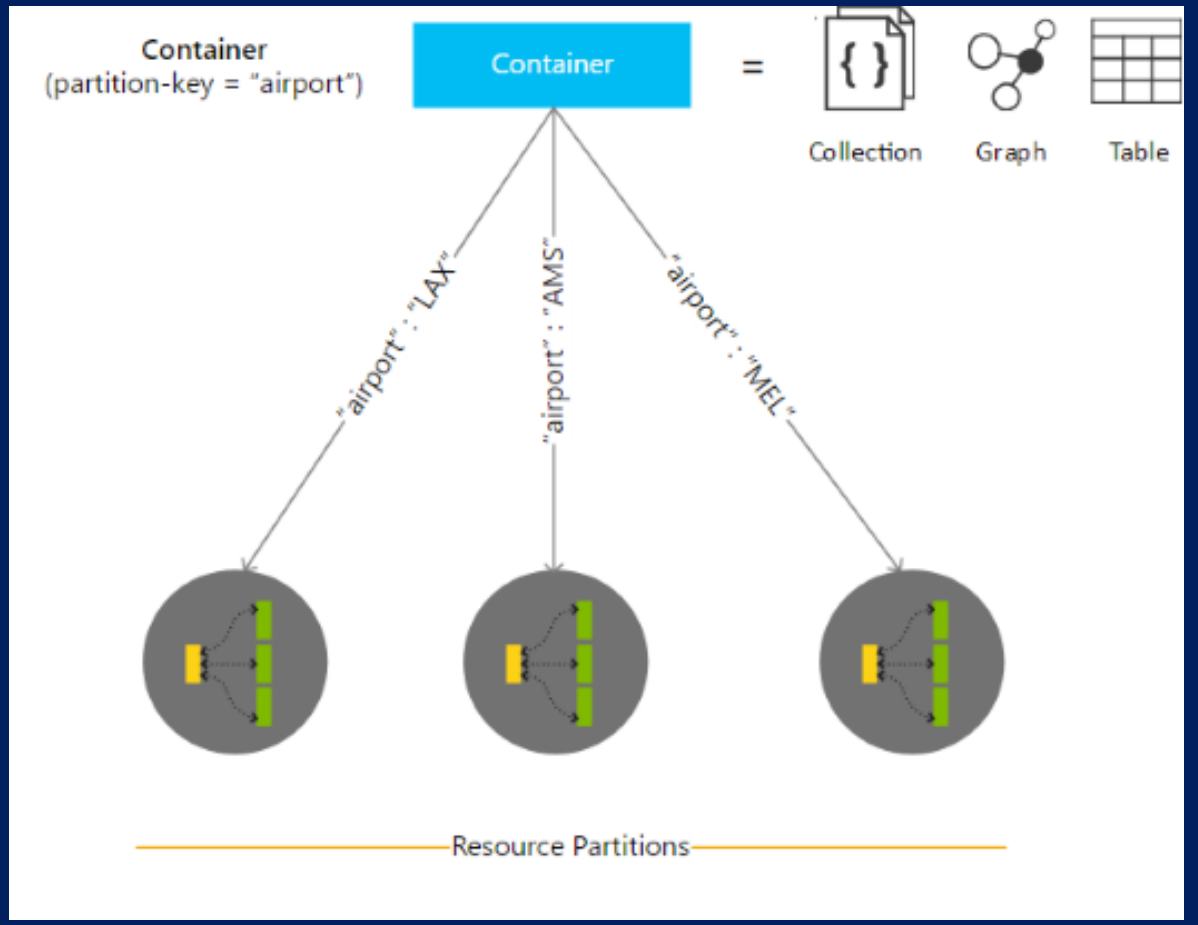
answered Jun 14 '17 at 12:12

David Makogon
52.1k ● 14 ● 93 ● 137

add a comment

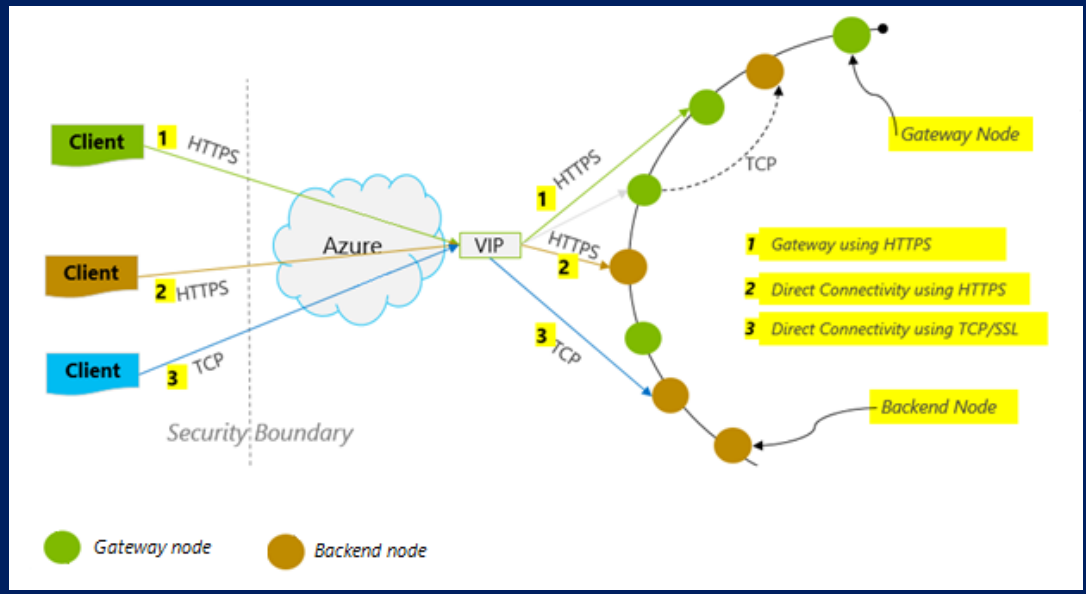
Partitioning

- Collections (Containers) for storing documents, graphs, or tables.
- Hash based partitioning
- Data in collections is subdivided into logical partitions using a partition key.
- Logical partitions are grouped into physical partitions
- By default logical partitions are 10gb (limit).
- The number of partitions is managed by Cosmos DB based on storage throughput requirements.
- Collections are redistributed (and split) to physical partitions in real time.
- An ideal partition key enables you to use efficient queries and has sufficient cardinality to ensure solution is scalable



Direct vs Gateway Connections

- Direct is normally faster as it allows direct connection to backend nodes.
- Only Direct connections support TCP connections
- Gateway can be efficient for cross partition queries. HTTP connections.
- Partition + key: resolvable to physical record.
- On startup the Cosmos client library needs to download partition ranges and routing information from the master DB.



4 Choice of Consistencies

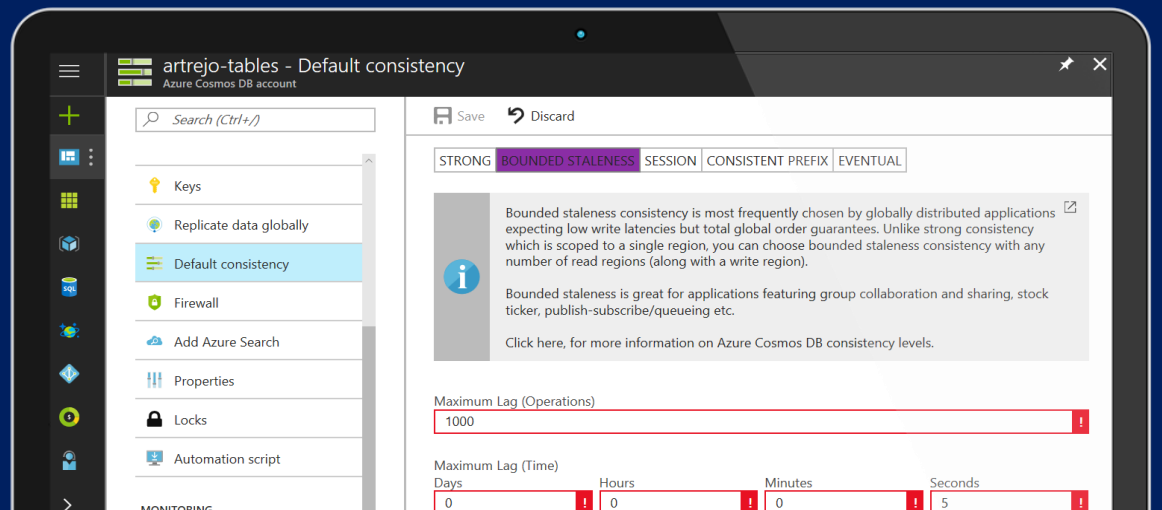
Navigating CAP theorem
Consistent data worldwide

Azure Cosmos DB

5 well-defined consistency models

Clear Tradeoffs

- Latency
- Availability
- Throughput



Scope of Consistencies

- The granularity of consistency is scoped to a single user request.
- A write request may correspond to an insert, replace, upsert, or delete transaction.
- The user may be required to paginate over a large result-set, spanning multiple partitions
- But each read transaction is scoped to a single page and served from within a single partition.

Schema-agnostic, automatic indexing

Schema-agnostic, automatic indexing

- At global scale, schema/index management is hard
- Automatic and synchronous indexing of all ingested content - hash, range, geo-spatial, and columnar
  - No schemas or secondary indices ever needed
- Resource governed, write optimized database engine with latch free and log structured techniques
- Online and in-situ index transformations
- While the database is fully schema-agnostic, schema-extraction is built in
  - Customers can get Avro schemas from the database

Multi-Model

Azure Cosmos DB

Multi-model and multi-API

DATA MODEL

Key-value

Document

Graph

…

APIs

SQL (DocumentDB)

MongoDB

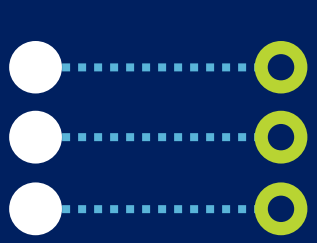
Tables

Gremlin Graph

..

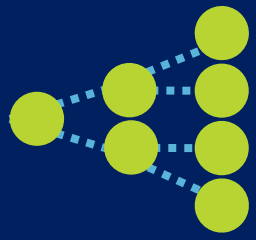
Native Support for Multiple Data Models

- API and wire protocols are supported via extensible modules
- Instance of a given data model can be materialized as trees
- Graph, documents, key-value, column-family, … *more to come*

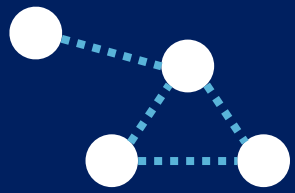


KEY-VALUE

COLUMN-FAMILY



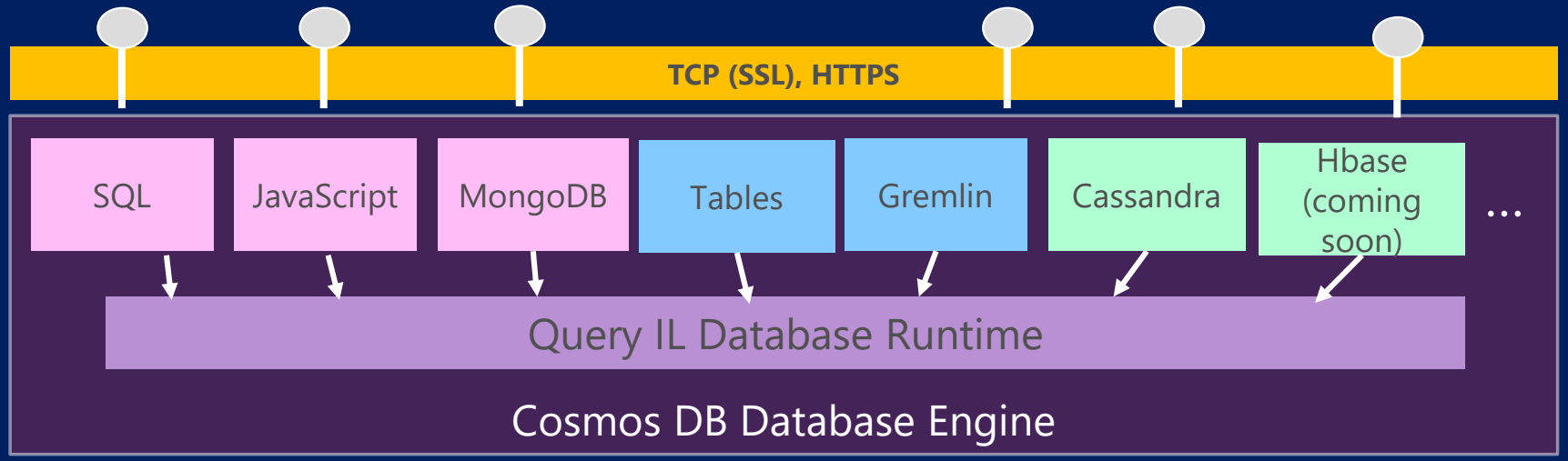
DOCUMENT



GRAPH

Native Multi-API

Native Support for Multiple APIs, formats & Wire Protocols



Powering global solutions

Field-tested by Microsoft's planet-scale services and industry-leading enterprises apps

Globally-distributed mission-critical apps

Guarantee uptime to users worldwide with high-availability and low-latency

IoT

Scale instantly for uncertain IoT workloads without sacrificing performance

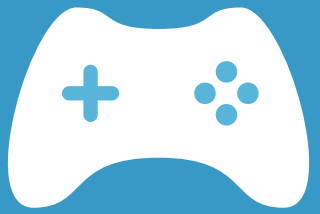
Personalization

Generate personalized service through low-latency and tunable consistency settings

Retail and e-commerce

Support queries over product catalogs, traffic spikes, and rapidly changing inventory

Gaming

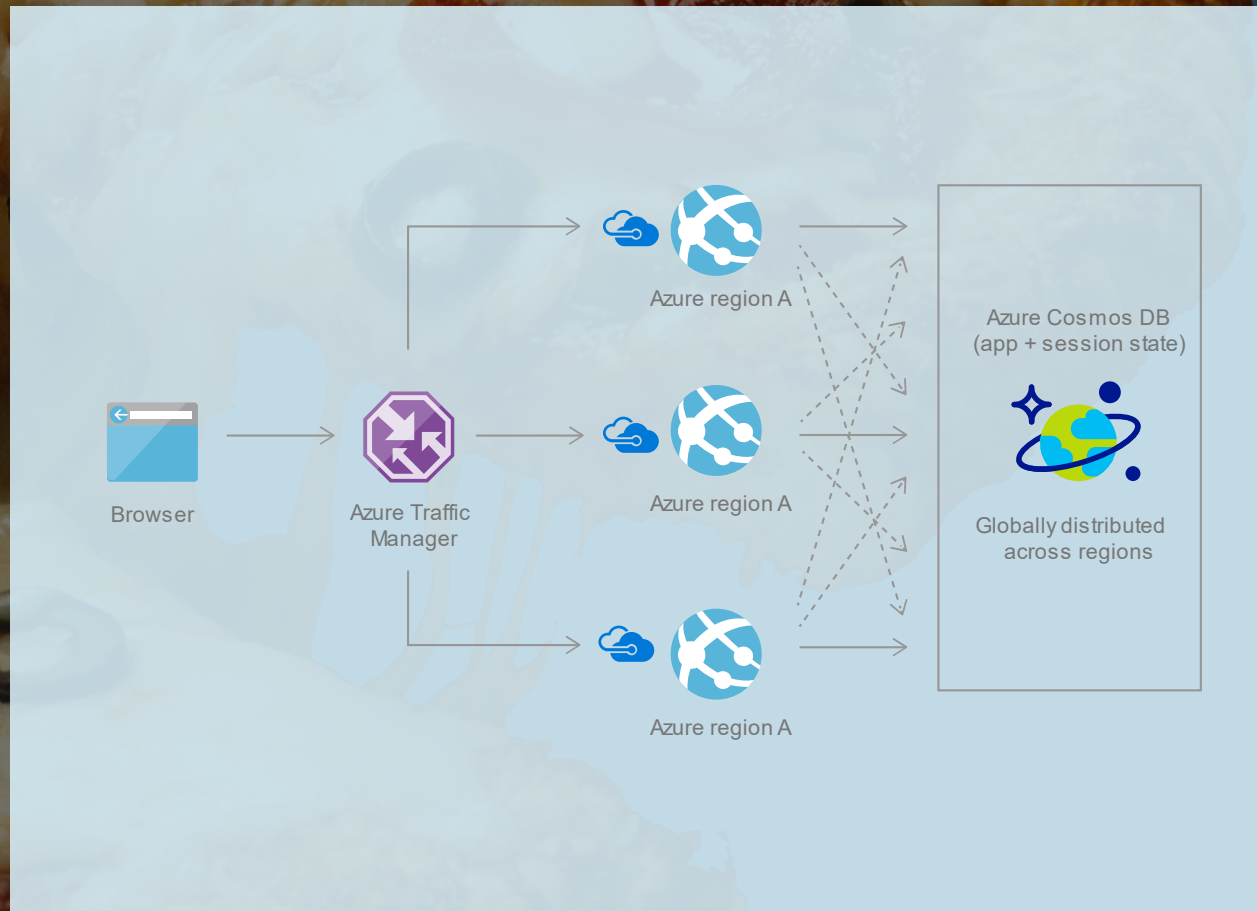


Accommodate bursts of traffic and deliver low-latency multiplayer experiences

Domino's Pizza delivers through globally-distributed apps

Tech-centric consumers across continents demand instant access and uninterrupted service

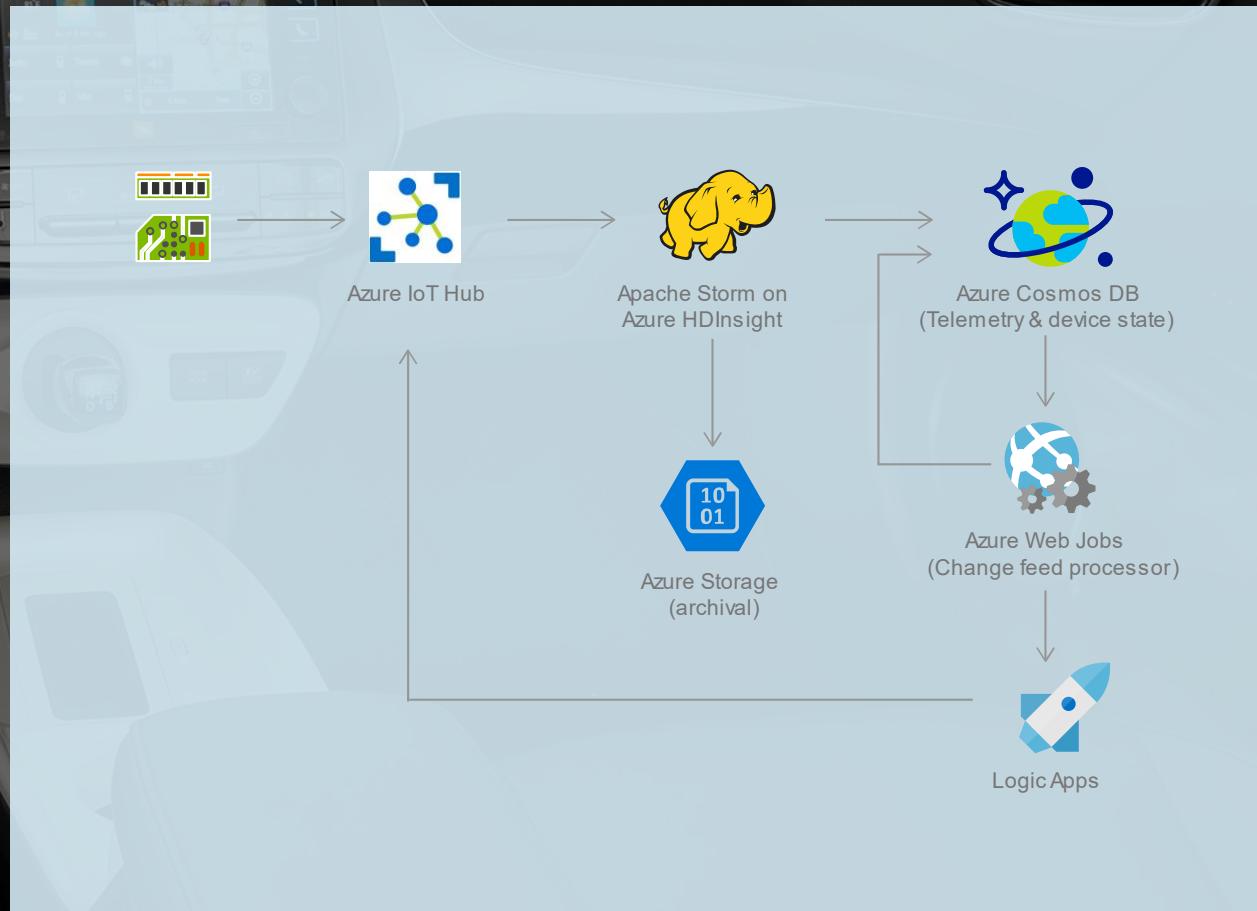
- 99.99% uptime
- Millisecond load latency
- Globally distributed order-processing



Toyota steers IoT telemetrics toward the future

Diverse and unpredictable IoT sensor workloads require a responsive data platform

- Real-time vehicle diagnostics
- Instant elastic scaling
- No loss in ingestion or query performance

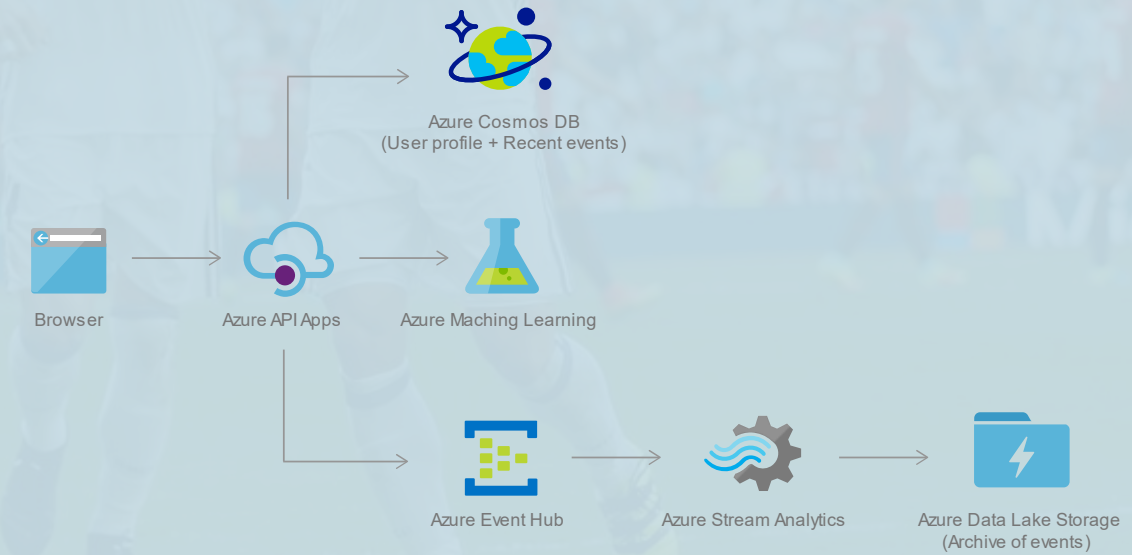


Personalized services score big with Real Madrid fans

Realmadrid

450M global supporters want direct engagement with the football club

- Tunable consistency settings for rapid insight
- Customized experiences through digital interactions
- High app performance worldwide

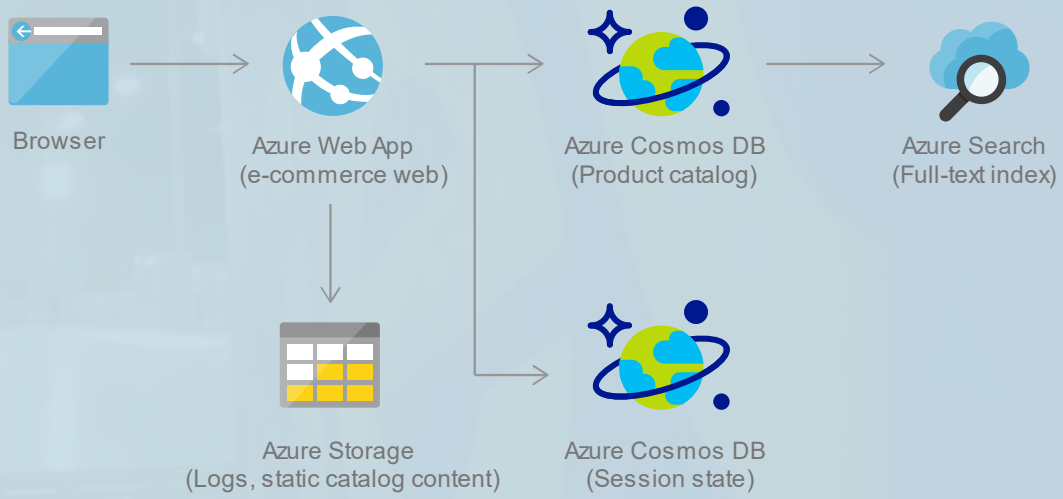


Retail and e-commerce

Jet.com flies through busy retail peaks

Black Friday, Cyber Monday, and other high traffic periods threaten service quality

- Immediate inventory updates
- Real-time change feeds
- Low latency for swift processing

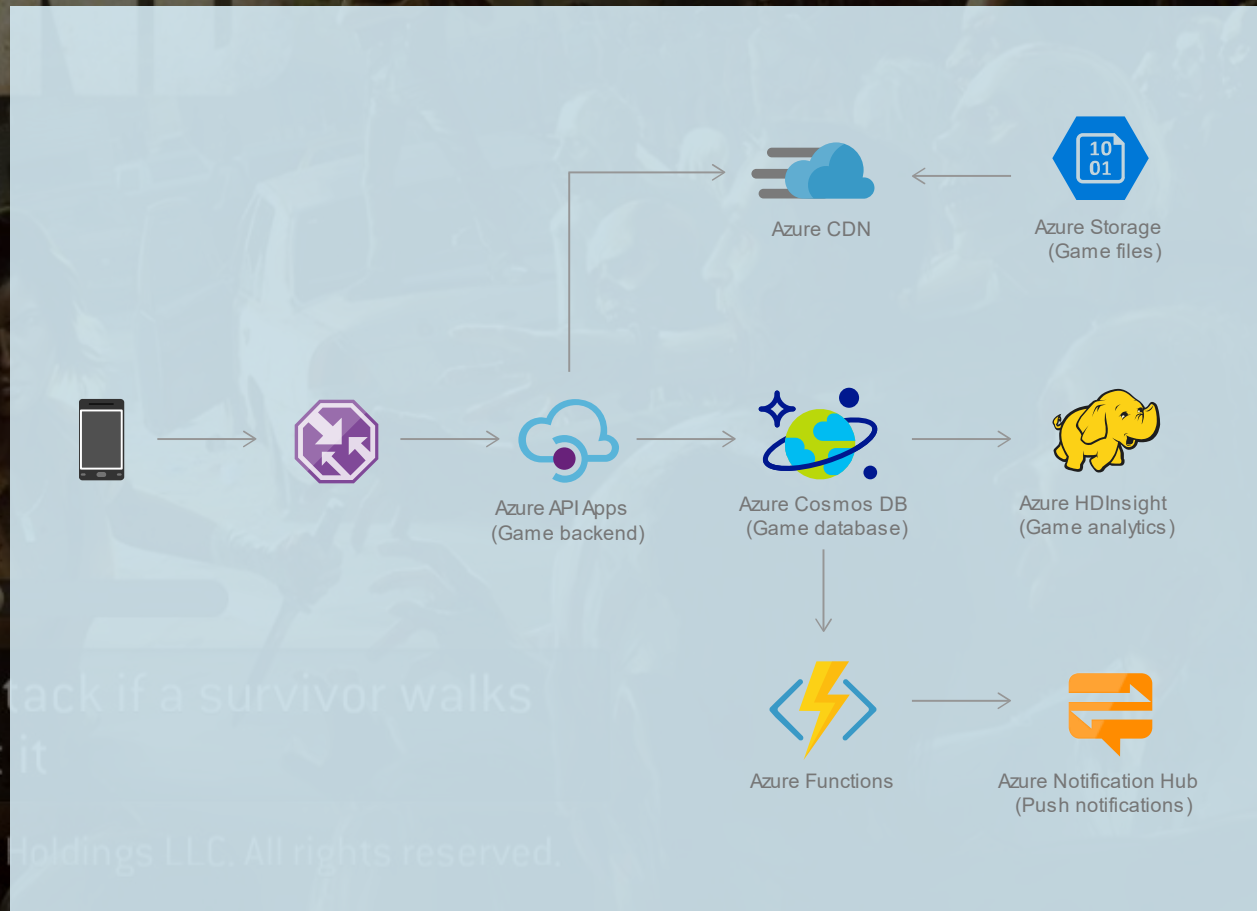


Gaming

Next Games RPG springs to life with Azure Cosmos DB

Need for a DB that to seamlessly respond to massive scale and performance demands

- Multi-player game play with low latency
- Instant capacity scaling from launch onward
- Uninterrupted global user experience



Global apps need global data
from a service that's out of this world.

Welcome to Azure Cosmos DB

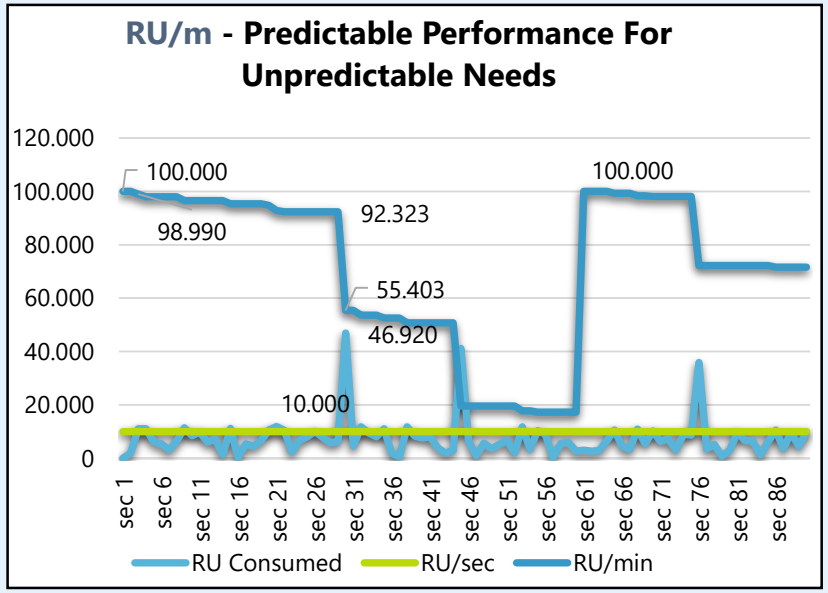
Start free

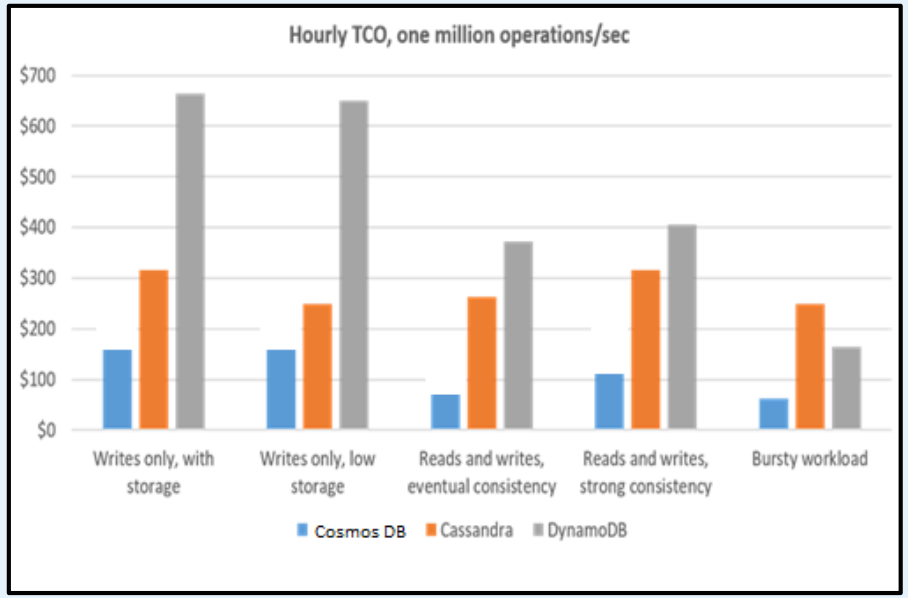
Azure Cosmos DB – Lowest TCO

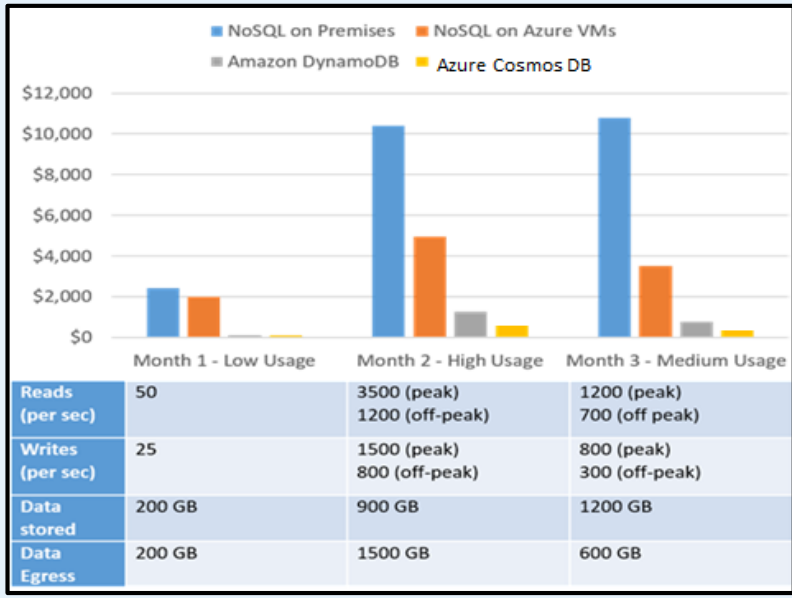
Deeply exploits cloud core properties and economies of scale

- Commodity hardware
- Fine-grained multi-tenancy
- End to end resource governance
- Optimal utilization of resources

Cosmos DB: 5-10X more cost-effective







	Month 1 - Low Usage	Month 2 - High Usage	Month 3 - Medium Usage
Reads (per sec)	50	3500 (peak) 1200 (off-peak)	1200 (peak) 700 (off peak)
Writes (per sec)	25	1500 (peak) 800 (off-peak)	800 (peak) 300 (off-peak)
Data stored	200 GB	900 GB	1200 GB
Data Egress	200 GB	1500 GB	600 GB

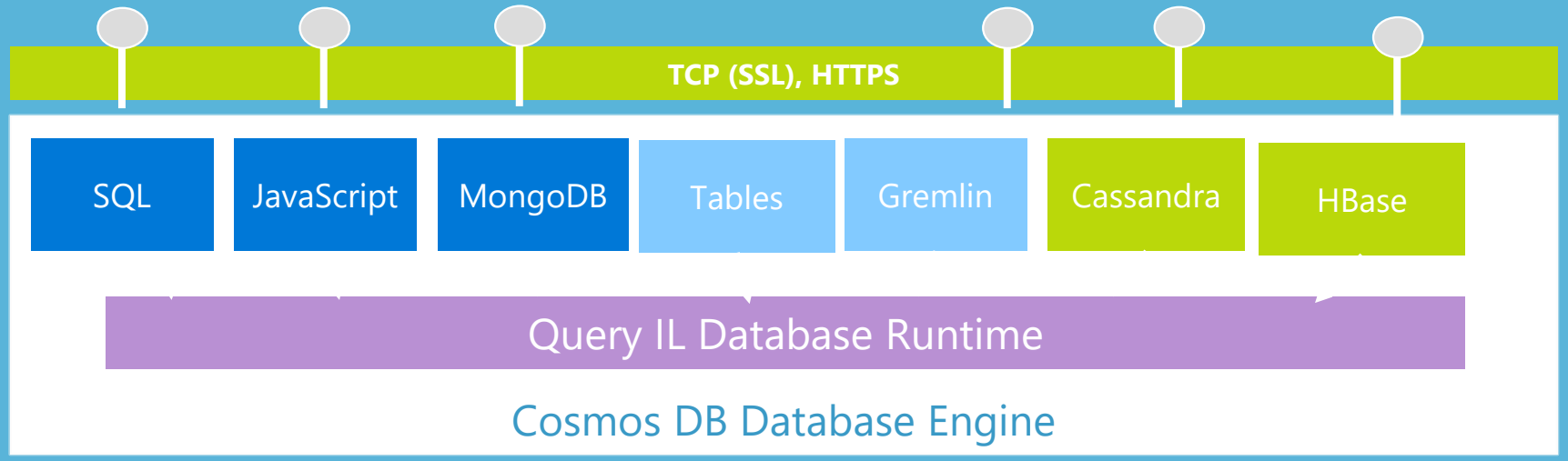
Customers save 60-73% in provisioning cost!

Azure Cosmos DB: TCO Factors

Deeply exploits cloud core properties and economies of scale

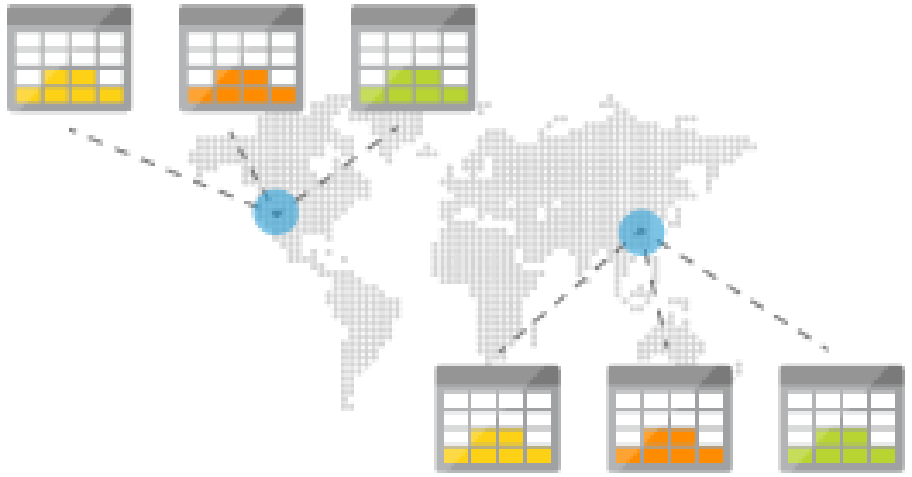
- Fine-grained multi-tenancy.
- No administration dev/ops required
- Superior elasticity
- Economy of Scale
- Cloud-optimized

Native Support for Multiple APIs, formats & Wire Protocols



Azure Tables

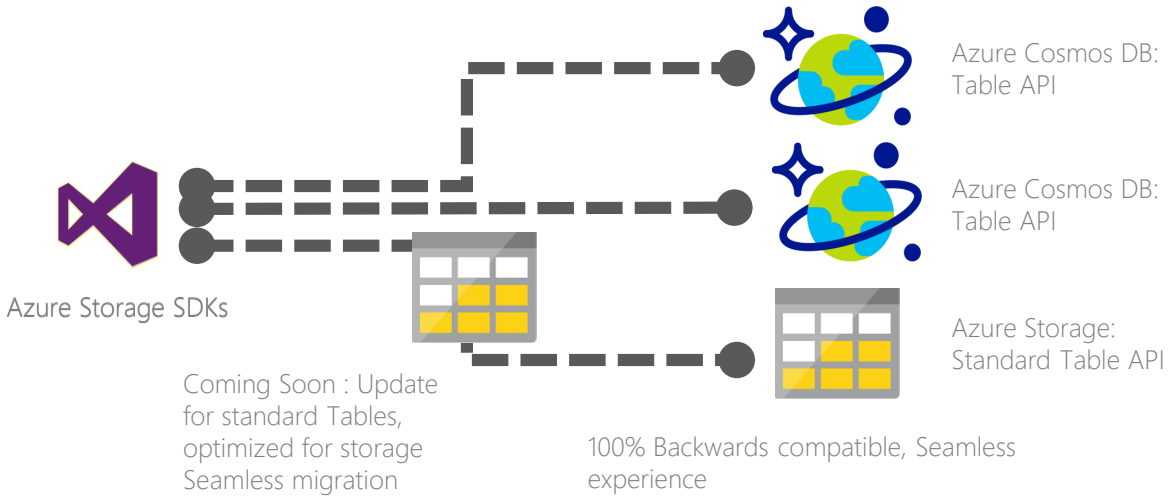
- Key-Value store for rapid development
- Low Cost
- Simplicity
- Massively Scalable
- Flexible data schema
- Globally replicated
- Enterprise ready



Tables API in Azure Cosmos DB

- Premium experience (low latency, well-defined consistency)
- Globally Distributed
- Secondary Indexes for user-defined queries
- Millisecond latency, Guaranteed throughput
- We heard you – *"Top user voice asks"*

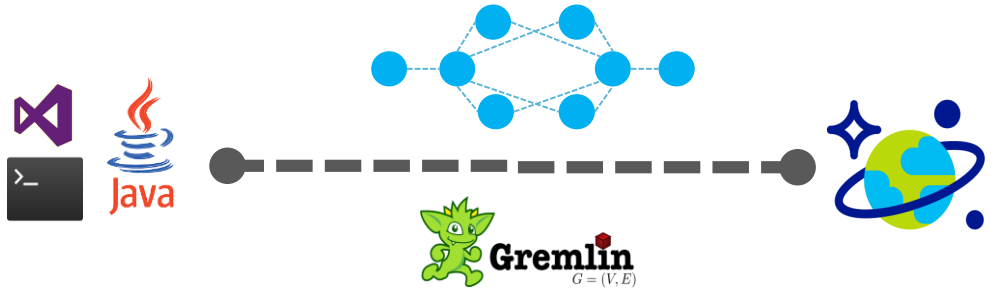
preview



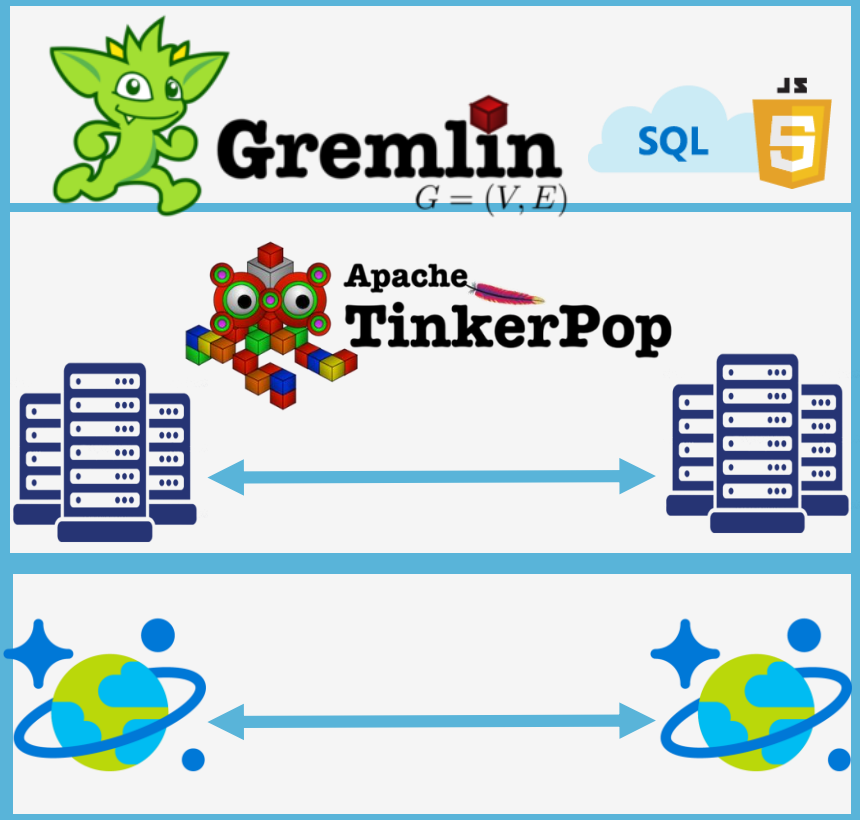
preview

Gremlin API in Azure Cosmos DB

- Model the real world
- Relationship as first-class entities
- Optimized for graph storage & traversal
- Gremlin standard



Native Graph Processing

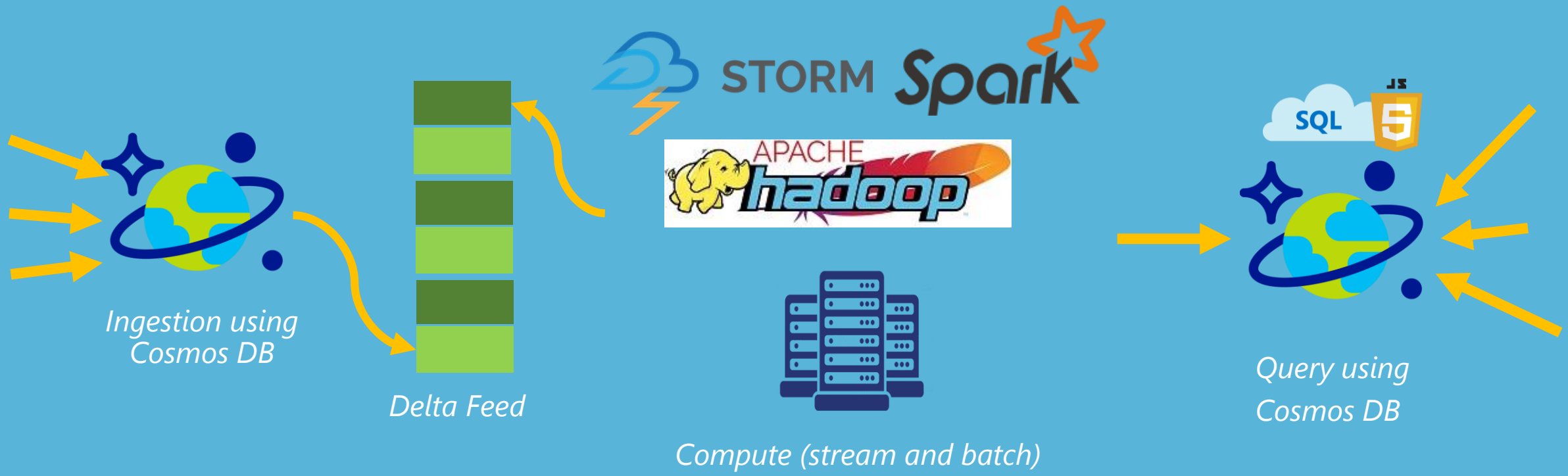


Gremlin and SQL query languages

Independently scalable graph engine (using Tinkerpop framework)

Globally distributed, elastically scalable, low latency, auto-indexed service

Change Feed



Lambda pattern with significantly lower TCO
Single scalable database solution for **both** ingestion and query

Spark Connector to Azure Cosmos DB

RDD and Dataset-based connectors available

Native integration with Spark SQL

Direct mapping to Cosmos DB partitions

Natively leverage Cosmos DB index

Predicate pushdown

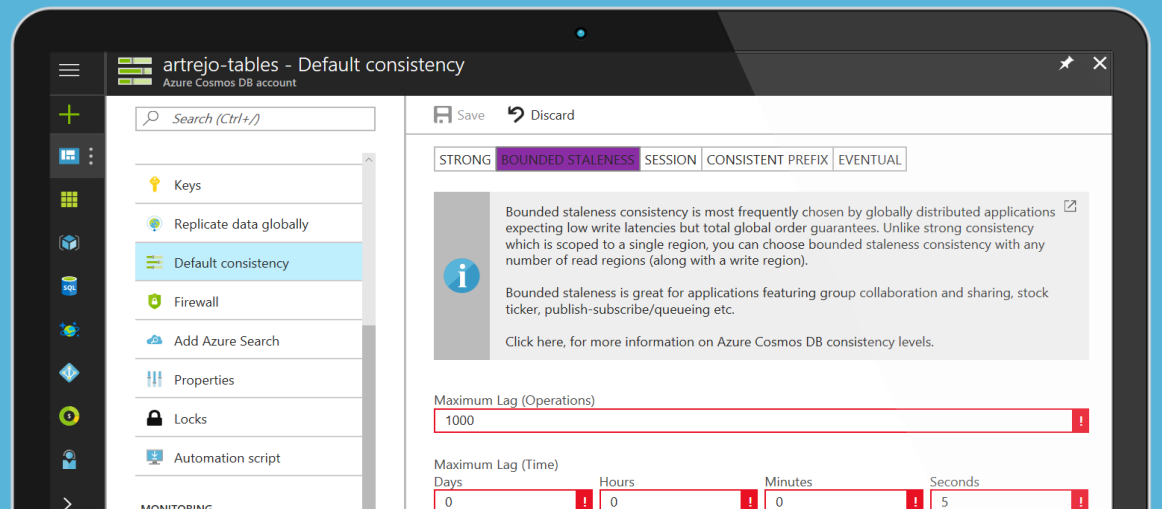
Available now

Azure Cosmos DB

5 well-defined consistency models

Clear Tradeoffs

- Latency
- Availability
- Throughput



Powering global solutions

Field-tested by Microsoft's planet-scale services and industry-leading enterprises apps

Globally-distributed mission-critical apps	IoT	Personalization	Retail and e-commerce	Gaming
Domino's	TOYOTA	Realmadrid	jet	THE WALKING DEAD NO MAN'S LAND
Digital ordering rapidly processed across four continents	Massive volumes of car sensor data drives customer service and vehicle diagnostics	Customized, real-time experiences for fans through live and digital interactions	Heavy volume managed with real-time change feeds and elastic scaling	Seamless gameplay despite traffic spikes from launch through growth

Next Steps

Getting Started

- cosmosdb.com
- portal.azure.com
- aka.ms/cosmosdb
- aka.ms/cosmosdb-Tables
- aka.ms/cosmosdb-Graph
- aka.ms/cosmosdb-MongoDB
- aka.ms/cosmosdb-DocumentDB
- cosmosdb.com/capacityplanner

Download

- aka.ms/CosmosDB-emulator

Re-visit Build session recordings on Channel 9.

Continue your education at Microsoft Virtual Academy online.