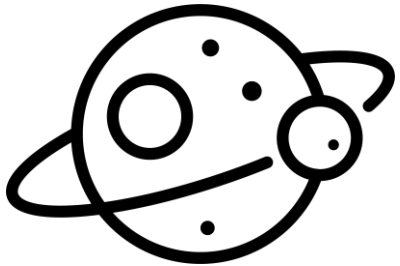


Azure Cosmos DB

Planet-scale, globally distributed, multi-model

Yeah, right…

Andre Essing

Technology Solutions Professional
Microsoft Deutschland GmbH

Andre advises, in his role as Technology Solutions Professional, customers in topics all around the Microsoft Data Platform. He is specialized in mission-critical systems, high-availability, security, database operations and of course the cloud.

andre.essing@microsoft.com | andreessing.de | aessing | Andre_Essing | @aessing | aessing

Azure Cosmos DB

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

SQL

MongoDB

Table API

Gremlin

$G = (V, E)$

cassandra

Key-value

Column-family

Document

Graph

Guaranteed low latency at the 99th percentile

Elastic scale out
of storage & throughput

Five well-defined consistency models

Turnkey global distribution

Comprehensive SLAs

Global Distribution

Turnkey Global Distribution

Worldwide presence as a foundational Azure service

Automatic multi-region replication

Multi-homing APIs

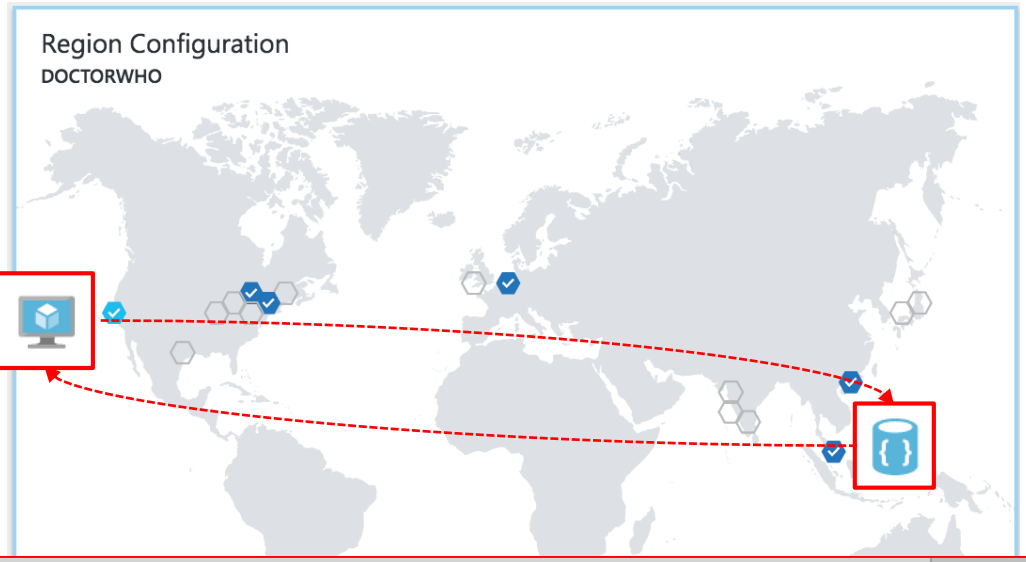
Manual and automatic failovers

Designed for High Availability

Why Global Distribution

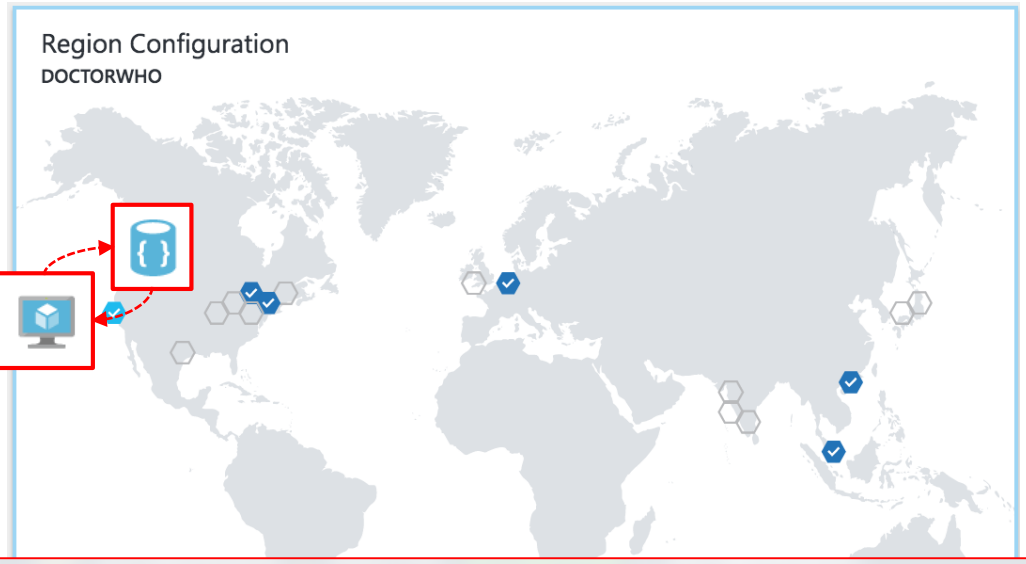
Low Latency (anywhere in the world)

- Packets cannot move faster than the speed of light
- Sending a packet across the world under ideal network conditions takes 100's of milliseconds
- You can *cheat* the speed of light – using data locality
  - CDN's solved this for *static* content
  - Azure Cosmos DB solves this for *dynamic* content
- Single-digit millisecond latency worldwide



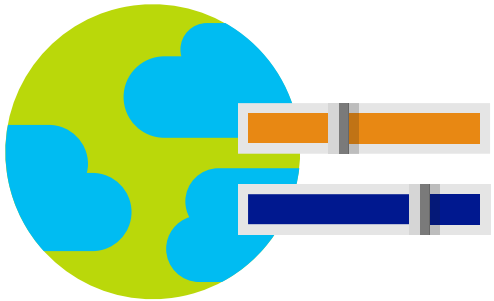
...rkspace/docdb/docdb — amypond@blink: ~/docdb — ssh amypond@104.42.108.173 -p 22

```
amypond@blink:~/docdb$ node testQ2.js
210.788804 milliseconds, 2 RUs, ActivityId: 9025eac6-eb74-4a07-94cf-f2383caffbb3
178.825773 milliseconds, 2 RUs, ActivityId: ea53b736-b629-4290-9cdf-cf80c6139461
178.839173 milliseconds, 2 RUs, ActivityId: f143c992-ba67-4c7b-b7b8-0b5db8df4dbf
178.564573 milliseconds, 2 RUs, ActivityId: 1a8d7b5b-42c5-4c39-9160-d1e5ab2200d0
179.229073 milliseconds, 2 RUs, ActivityId: 483b85de-74e0-4f48-9206-70ac1268c60e
178.653772 milliseconds, 2 RUs, ActivityId: 50fbfe91-f41e-4f14-8a15-0b344c894727
178.464572 milliseconds, 2 RUs, ActivityId: cac6446a-79d4-4886-81d8-1dda835daa72
180.708475 milliseconds, 2 RUs, ActivityId: d40af8e4-582b-4479-bb9c-e3582eac6774
```




```
[amypond@blink:~/docdb$ node testQ2.js
12.736112 milliseconds, 2 RUs, ActivityId: dd3c17b9-1b76-445a-8e27-29b7486bd7e4
4.947605 milliseconds, 2 RUs, ActivityId: e2f4c899-9fb1-4f76-a4ab-e5718fac5742
5.044005 milliseconds, 2 RUs, ActivityId: 0fc5d216-78a0-4d92-a3d0-63efd9dd6552
5.351205 milliseconds, 2 RUs, ActivityId: 861155f0-81ba-4c8a-9933-e50d8708cc21
4.553505 milliseconds, 2 RUs, ActivityId: 3db9641f-70f1-4ef1-84bb-809280bbe1a5
5.427506 milliseconds, 2 RUs, ActivityId: 10d1b2e5-e795-4c77-8655-815f410ba11e
5.900106 milliseconds, 2 RUs, ActivityId: bda1df86-c5ad-45c5-bcfb-93d417c54751
4.895405 milliseconds, 2 RUs, ActivityId: f206d58d-64a2-47f2-9653-145eaf47ff97
5.244306 milliseconds, 2 RUs, ActivityId: 3aa7e177-b1a9-413d-8023-55f5054d1b74
```

Elastic scale out of storage & throughput



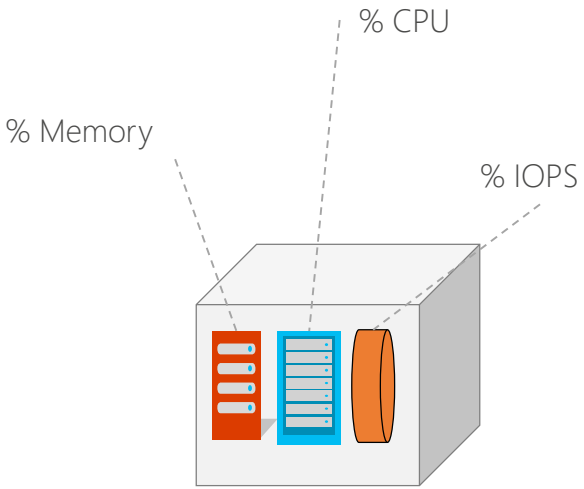
Elastically scalable storage and throughput

Elastically scale storage (GB to PB) and throughput
(100 to 100M req/sec) across many machines and multiple regions

Transparent server-side partition management

Automatic expiration via policy based TTL

Pay by the hour, change throughput at any time for only what you need



Request Units

Request Units (RU) is a rate-based currency

Abstracts physical resources for performing requests

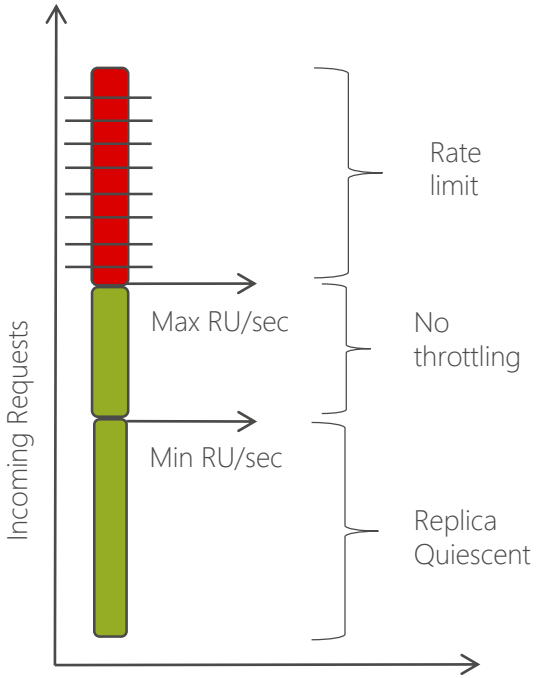
Foreground and background activities

Normalized across various access methods

1 RU = 1 read of 1 KB document

Each request consumes fixed RUs

Applies to reads, writes, queries, and stored procedure execution



Request Units

Provisioned in terms of RU/sec

Rate limiting based on amount of throughput provisioned

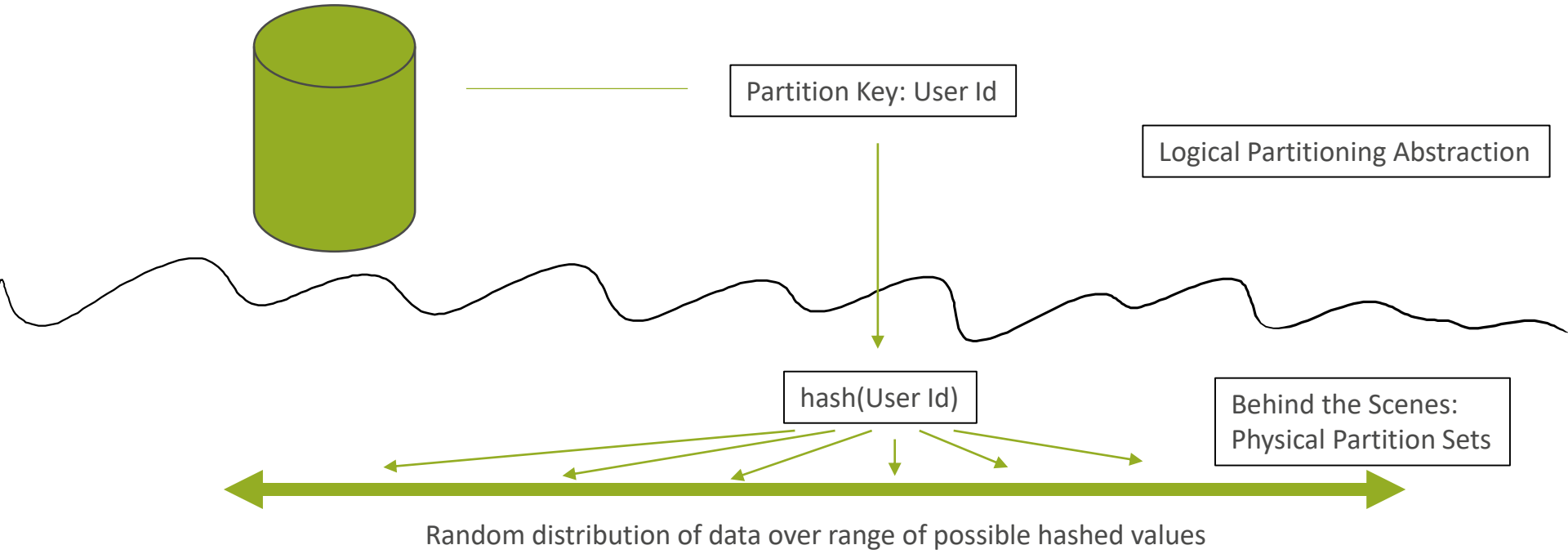
Can be increased or decreased instantaneously

Metered Hourly

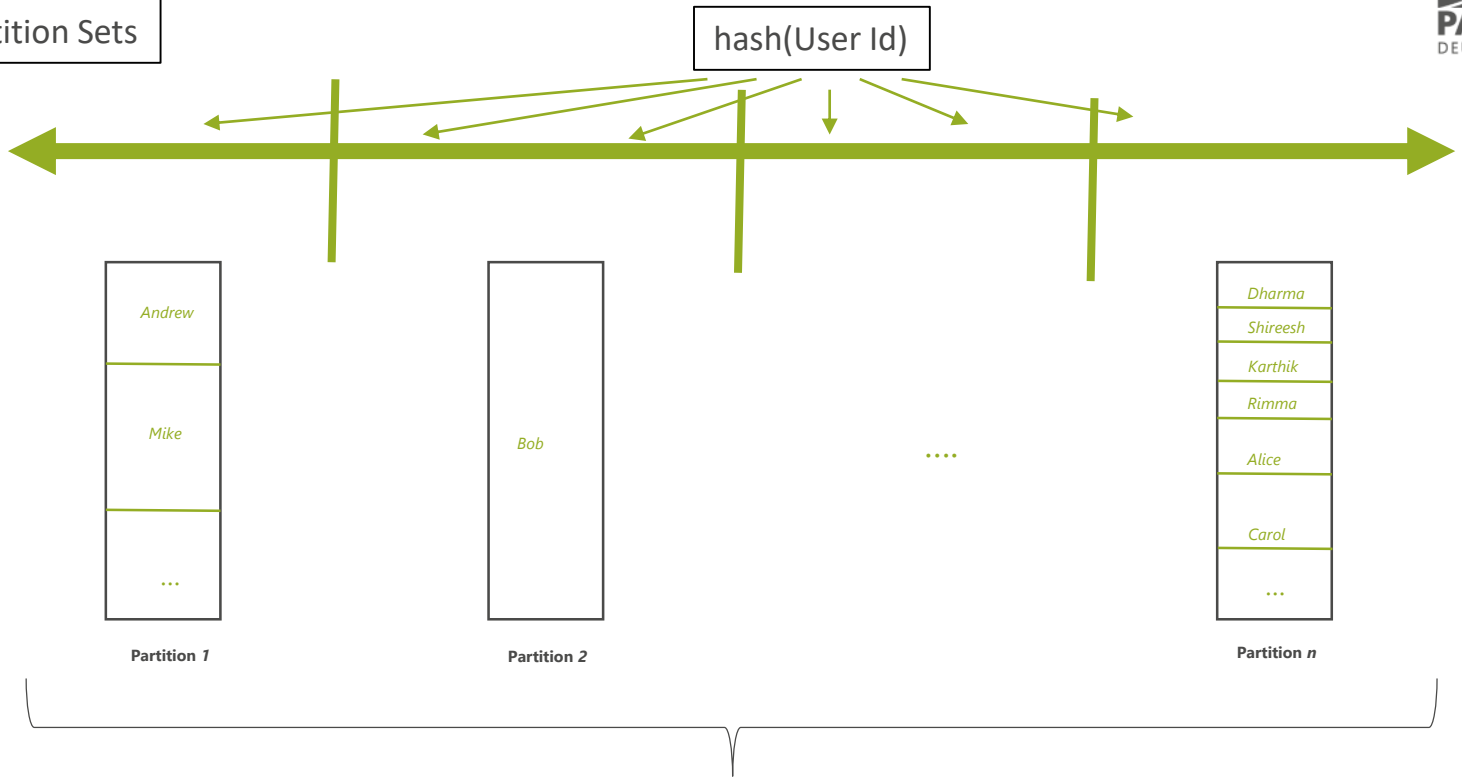
Background processes like TTL expiration, index transformations scheduled when quiescent

Partitioning

Cosmos DB Container (e.g. Collection)

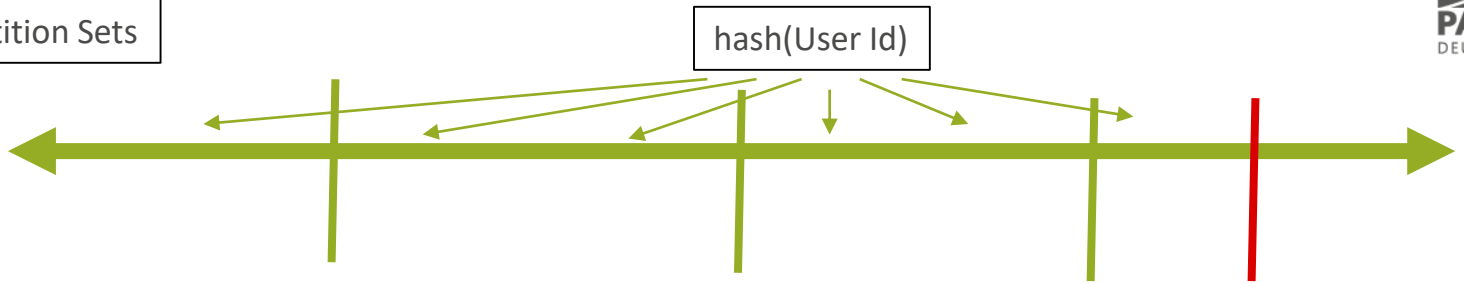


Behind the Scenes: Physical Partition Sets



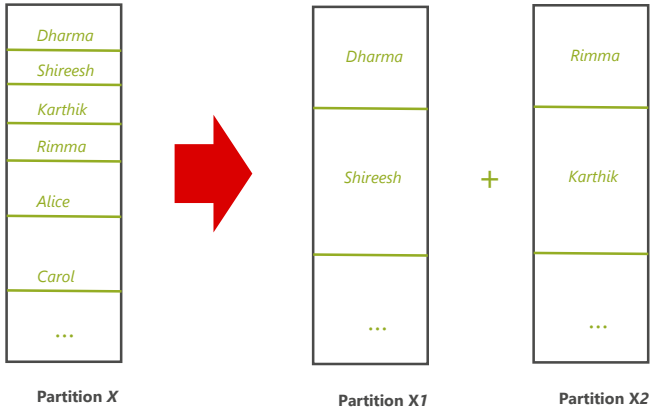
# of partitions based on actual storage and throughput needs
(yielding scalability with low total cost of ownership)

Behind the Scenes: Physical Partition Sets

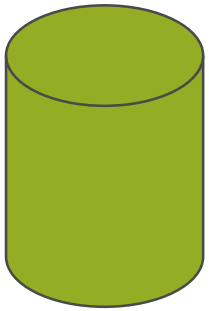


Partition Ranges can be dynamically sub-divided
to seamlessly grow database as the application grows
while sedulously maintaining high availability

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

Steps for Success

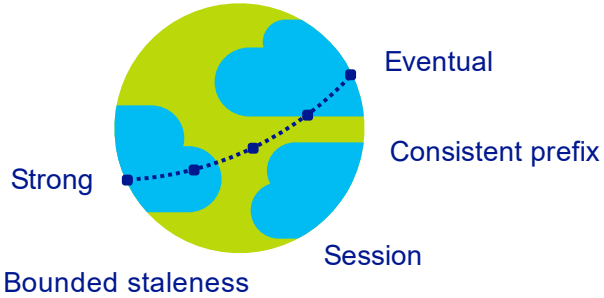
1. Ballpark scale needs (size/throughput)
2. Understand the workload
3. # of reads/sec vs writes per sec
    - Use 80/20 rule to help optimize bulk of workload
    - For reads – understand top X queries (look for common filters)
    - For writes – understand transactional needs
      understand ratio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

Consistency

Choice for most distributed apps

Strong consistency
high latency

Eventual consistency,
low latency



Multiple, well-defined consistency choices

Global distribution forces us to navigate the CAP theorem

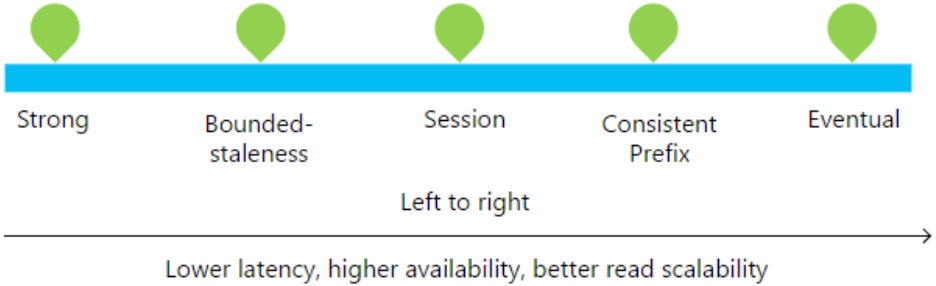
Writing correct distributed applications is hard

Five well-defined consistency levels

Intuitive and practical with clear PACELC tradeoffs

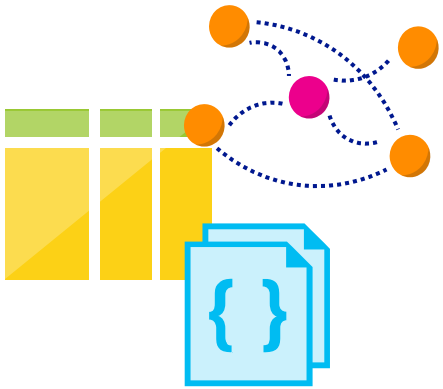
Programmatically change at anytime

Can be overridden on a per-request basis



Consistency Level	Guarantees
Strong	Linearizability (once operation is complete, it will be visible to all)
Bounded Staleness	Consistent Prefix. Reads lag behind writes by at most k prefixes or t interval Similar properties to strong consistency (except within staleness window), while preserving 99.99% availability and low latency.
Session	Consistent Prefix. Within a session: monotonic reads, monotonic writes, read-your-writes, write-follows-reads Predictable consistency for a session, high read throughput + low latency
Consistent Prefix	Reads will never see out of order writes (no gaps).
Eventual	Potential for out of order reads. Lowest cost for reads of all consistency levels.

Multi-Model & Multi-API



Multi-model, multi-API

Database engine operates on Atom-Record-Sequence type system

All data models can be efficiently translated to ARS

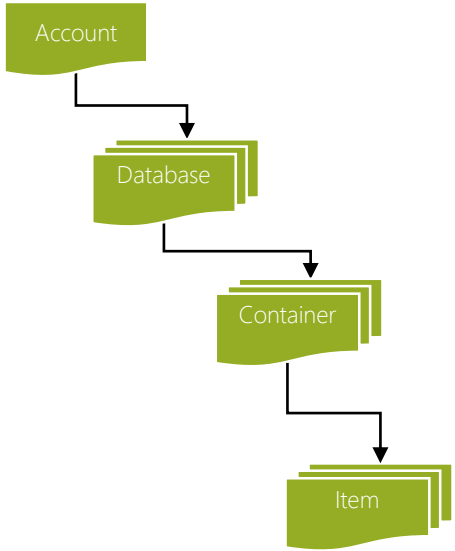
Multi-model: Key-value, Document, and Graph

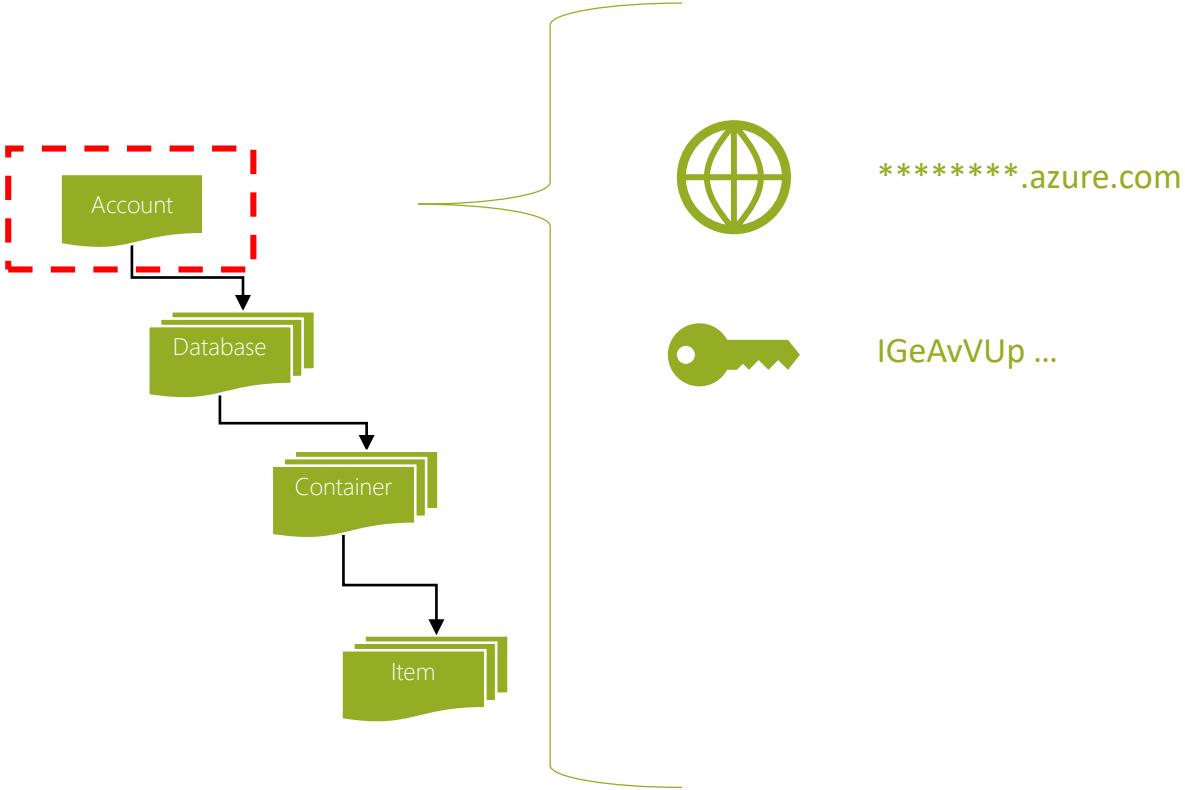
Multi-API: SQL (DocumentDB), MongoDB, Table, and Gremlin

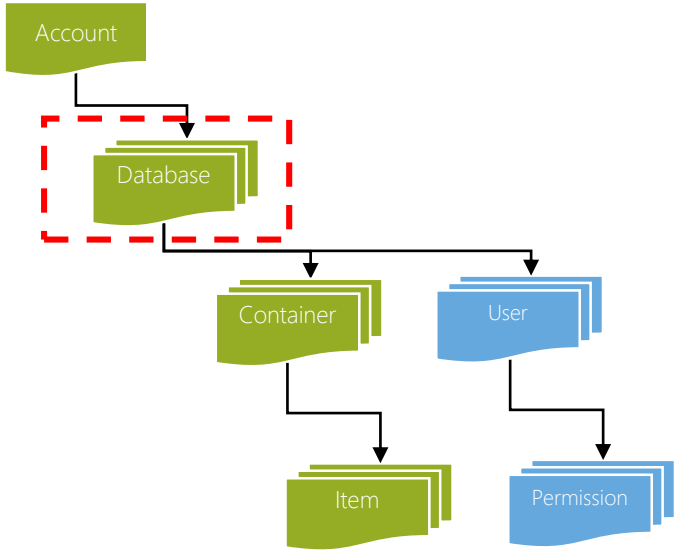
More data-models and APIs to be added (Cassandra, …)

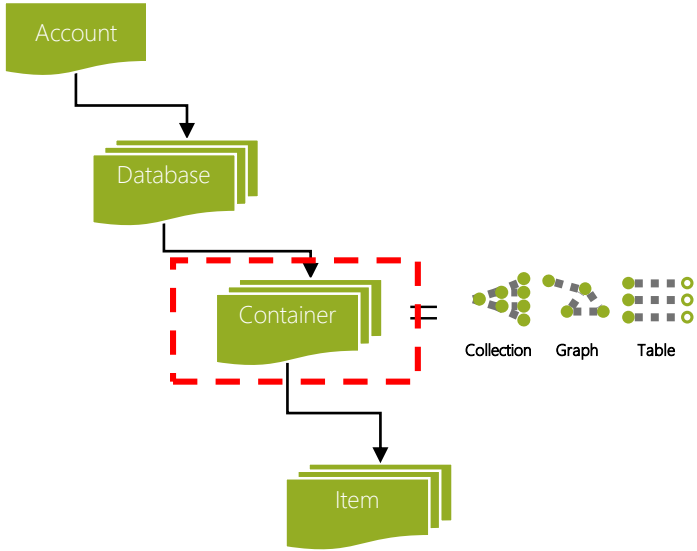
Spark Connector makes analytical scenarios possible

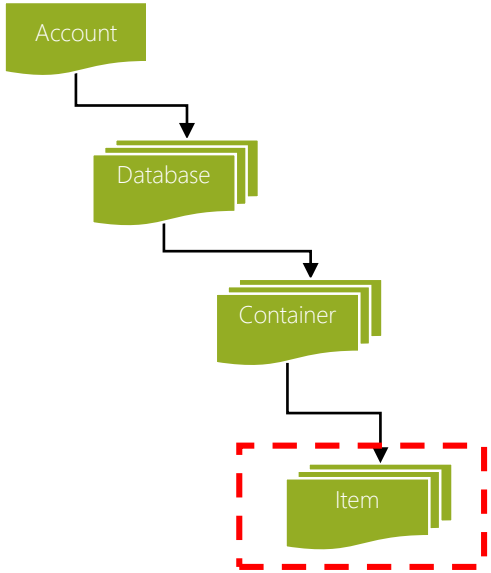
Database for Serverless with Azure Functions integration

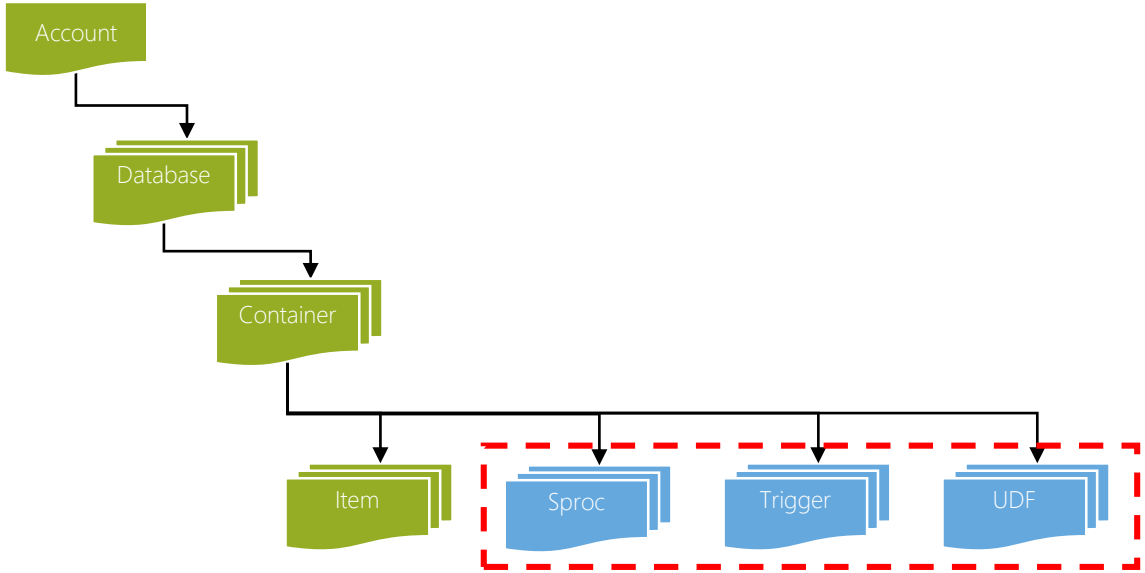

Account
Database
Container
Item

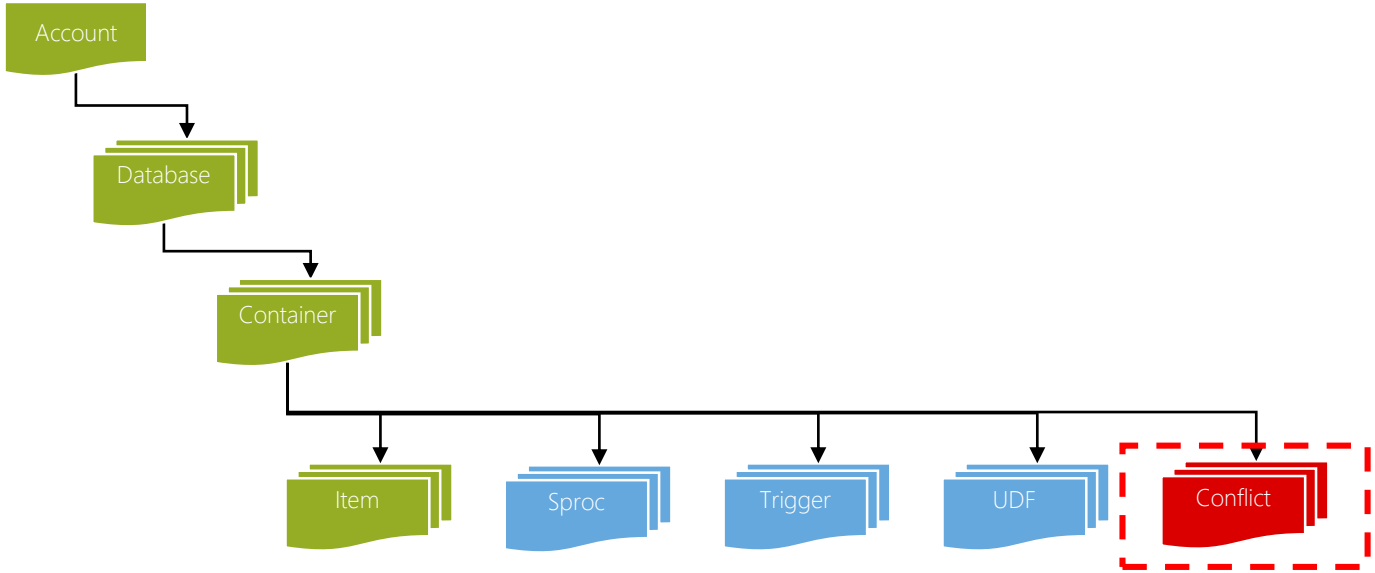

Account
Database
Container
Item
********.azure.com
IGeAvVUp ...


Account
Database
Container
User
Item
Permission

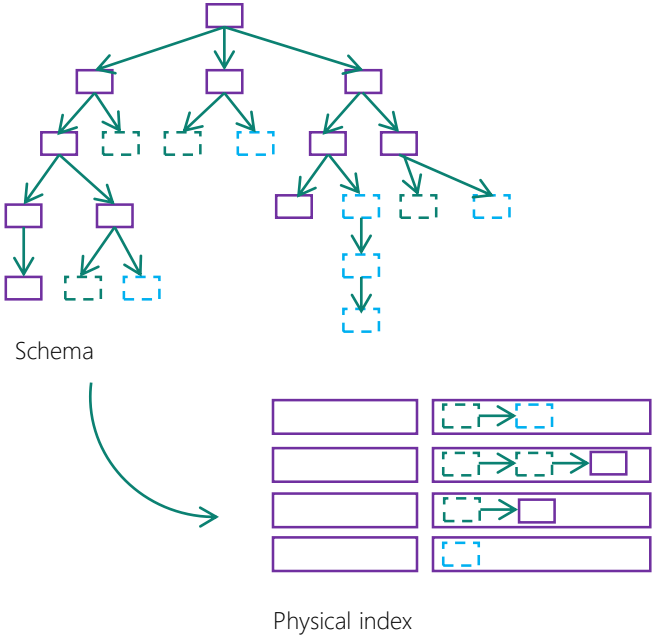

Account
Database
Container
Collection
Graph
Table
Item


Account
Database
Container
Item


Account
Database
Container
Item
Sproc
Trigger
UDF


Account
Database
Container
Item
Sproc
Trigger
UDF
Conflict

Indexing



Schema-agnostic, automatic indexing

Automatically index every property of every record without having to define schemas and indices upfront.

No need for schema and index management

Works across every data model

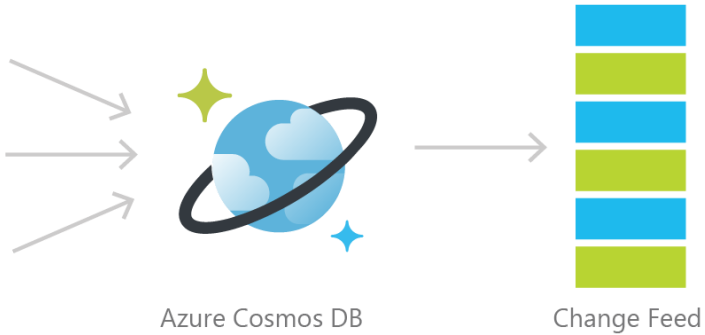
Latch free data structure for highly write-optimized database engine

Multiple index types: Hash, range, and geospatial

Indexing policy can be customized

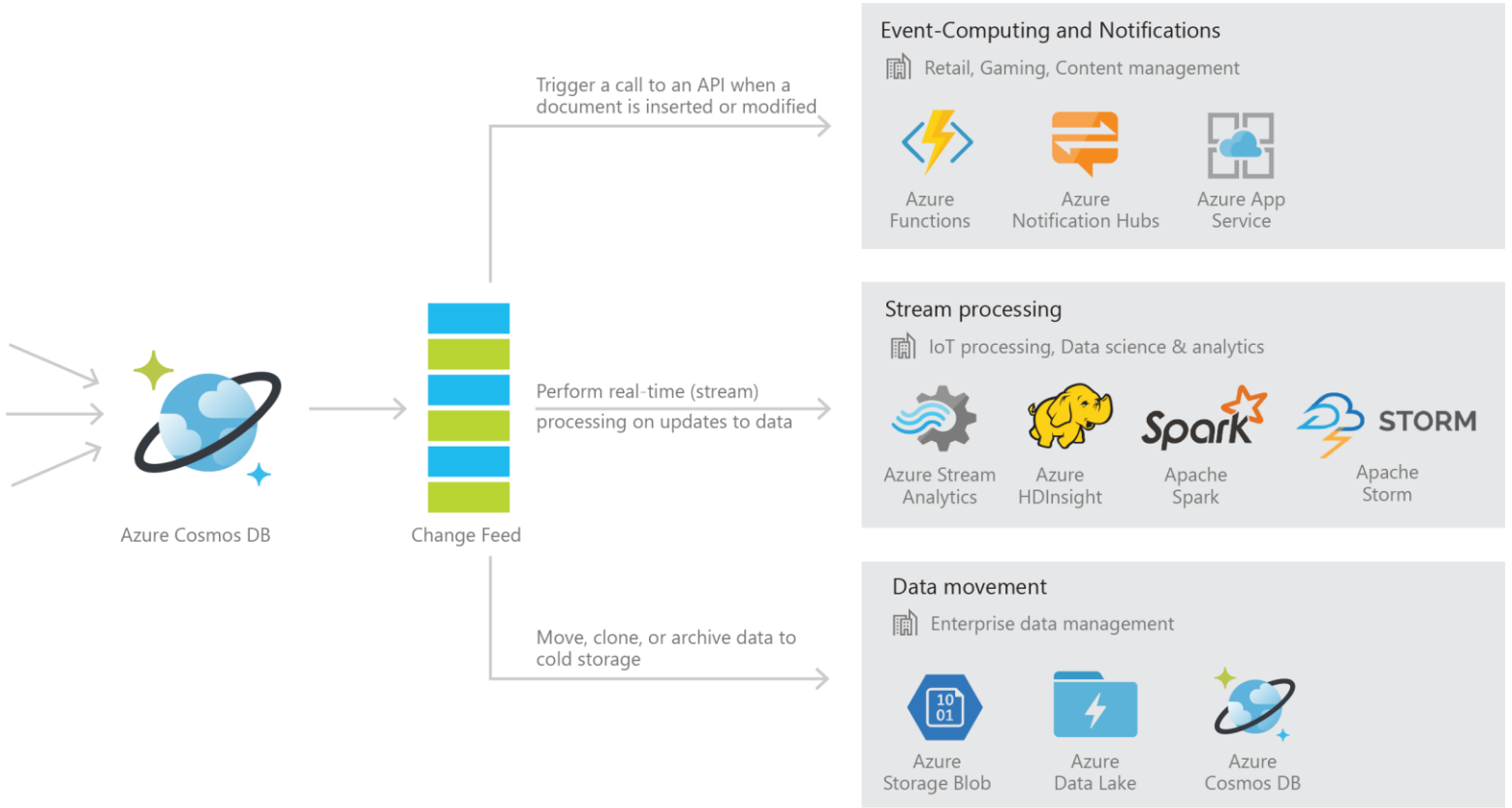
Change Feed

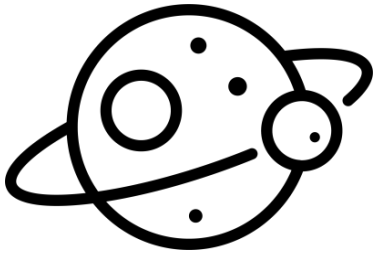
Azure Cosmos DB Change Feed



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

Common Scenarios





What else sets Azure Cosmos DB apart

Security & Compliance

Always encrypted at rest and in motion

Fine grained “row level” authorization

Network security with IP firewall rules

Comprehensive Azure compliance certification:

- ISO 27001
- ISO 27018
- EUMC
- HIPAA
- PCI
- SOC1 and SOC2

Industry-leading, enterprise-grade SLAs

99.99% availability – even with a single region

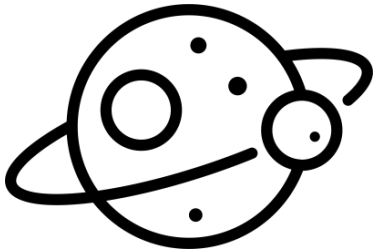
99.999% availability – with multiple regions

Made possible with highly-redundant storage architecture

Guaranteed durability – writes are majority quorum committed

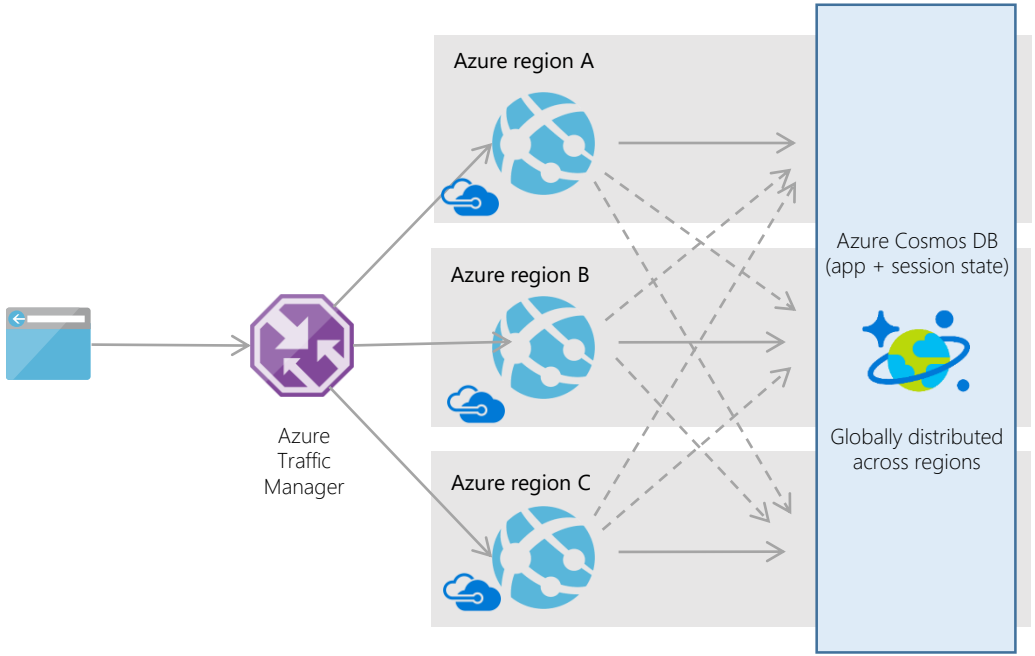
First and only service to offer SLAs on:

- Low-latency
- Consistency
- Throughput

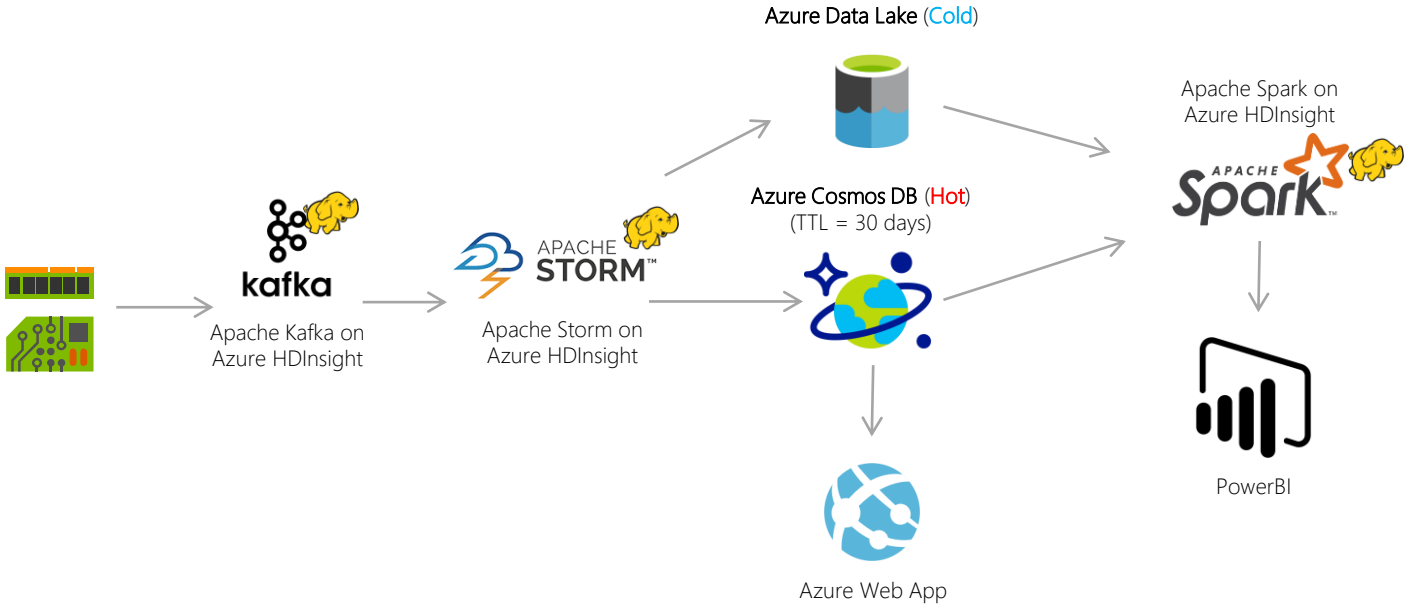


Common Use Cases and Scenarios

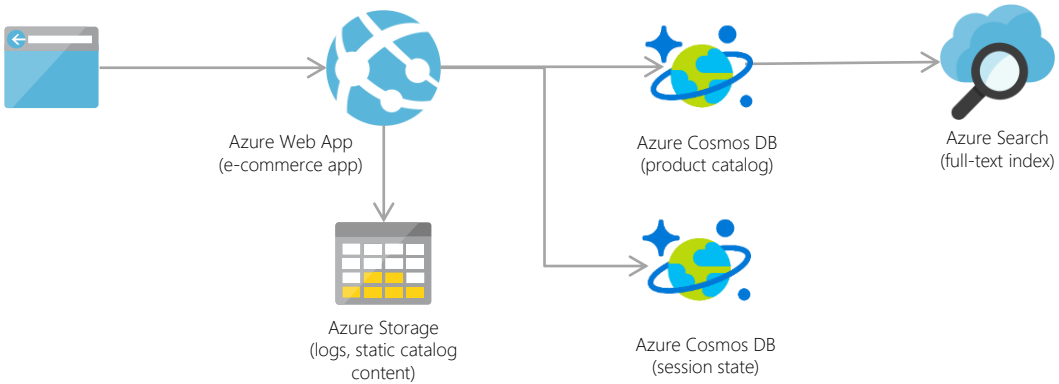
Content Management Systems



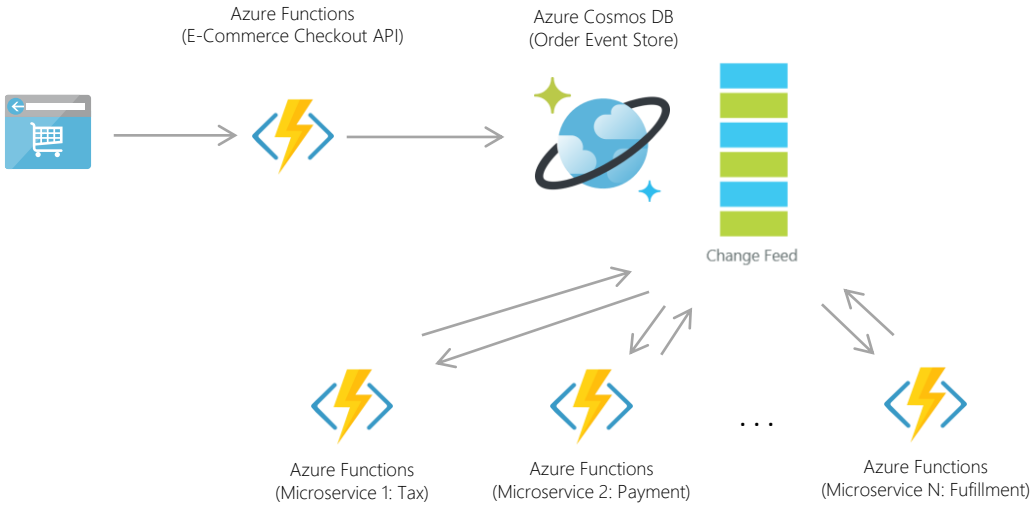
Internet of Things – Telemetry & Sensor Data

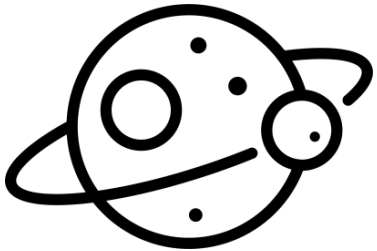


Retail Product Catalogs



Retail Order Processing Pipelines





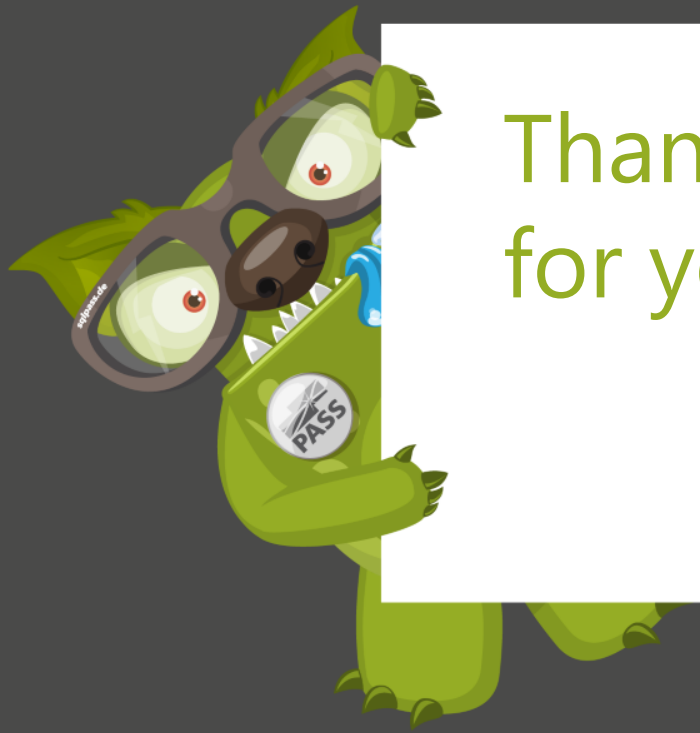
Questions & Answers

cosmosdb.com

Follow @AzureCosmosDB
Use #CosmosDB

stack overflow

#azure-cosmosdb



Thank you very much for your attention.