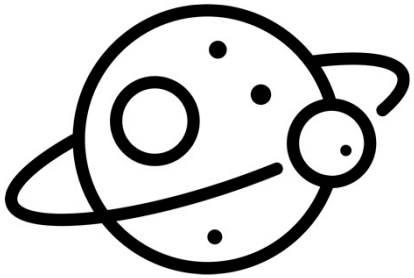


Azure Cosmos DB

Chris Campbell
Cloud Solution Architect
Advanced Analytics and AI
Microsoft

Different types of data are…different

Data comes in may shapes and sizes

Data is used for many different purposes

Some data relates to other data

Some data is related to many other pieces of data which are interrelated

Some data is self-contained

Some data is queried by keys

Some data is queried by properties

Some data is scanned

Why do we treat them all the same way?

Let's review…what is a relational database?

Data stored as two-dimensional tables

Set-based queries and operations

Data is strongly typed

Schemas are rigidly defined

The system enforces the schema

Queries filter data (WHERE…)

Queries shape data (SELECT…)

Queries return the Cartesian product of tables (JOIN…)

Why use relational?

Robust, stable and proven

Ideal when data conforms to a structured, rigid schema

When you need very flexible queries

When data needs to be very consistent and durable

When you need ACID (Atomic/Consistent/Isolated/Durable) transactions

When is relational less than optimal?

When you need to scale-out. Relational is typically scale-up only.

When your schema needs to be flexible.

When you're trying to support code-first development

It's not just about relational data anymore…we have lots of choices…

Relational

Time-tested architecture

Best when the data is strongly typed and the schema is rigid and normalized

Fast and flexible queries

Full SQL Support

Key/Value

List of keys and values

Rapid, discreet lookups

When you always know the key

Built for speed

Document

JSON-based

Built for scale and simplicity

Schema-free

Code-first development

Columnar

Data keyed by row and stored by column

Very large volumes

Very fast when queries are well-known

Scalable and resilient

Graph

Nodes and relationships

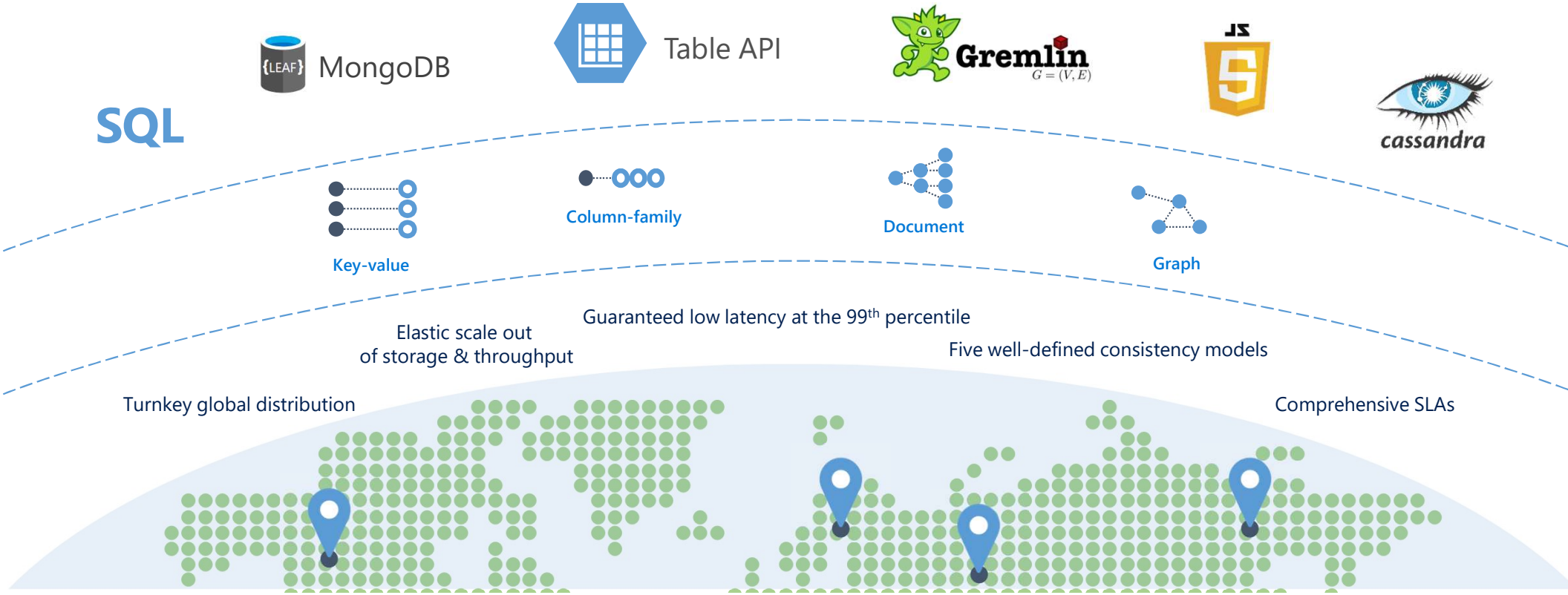
Complex connections

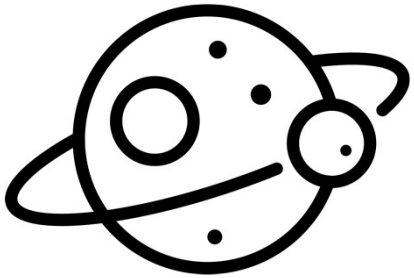
Find related objects

Find the shortest path

Azure Cosmos DB

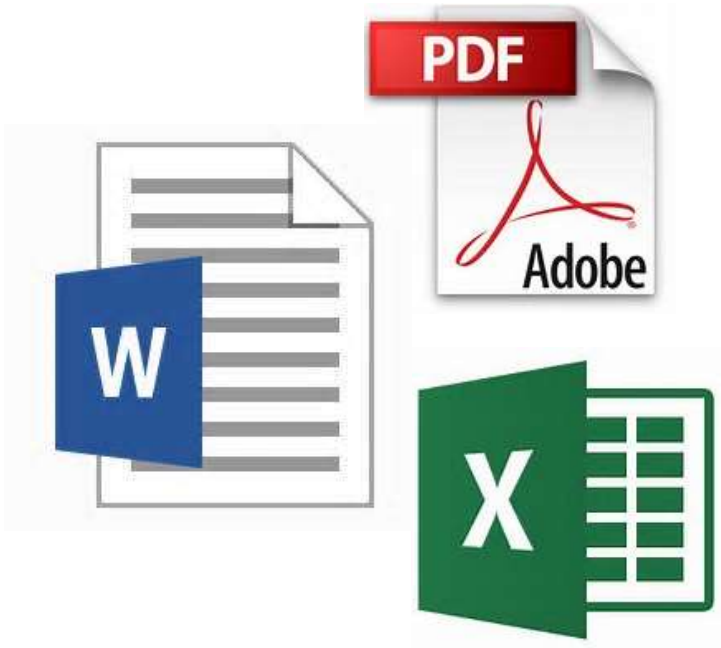
A globally distributed, massively scalable, multi-model database service





SQL API (Formerly known as DocumentDB API)

Not this kind of document…



This kind of document...


```
{
  "id": "19015",
  "description": "Snacks, granola bars, hard, plain",
  "tags": [
    {
      "name": "snacks"
    },
    {
      "name": "granola bars"
    },
    {
      "name": "hard"
    },
    {
      "name": "plain"
    }
  ],
  "version": 1,
  "isFromSurvey": false,
  "foodGroup": "Snacks",
  "servings": [
    {
      "amount": 1,
      "description": "bar",
      "weightInGrams": 21
    },
    {
      "amount": 1,
      "description": "bar (1 oz)",
      "weightInGrams": 28
    },
    {
      "amount": 1,
      "description": "bar",
      "weightInGrams": 25
    }
  ]
}
```

What's a document database?

Data stored JSON documents

Non-relational

Promotes code-first development

Built for simplicity, scale and performance

No schema enforced

Very flexible query options

Resilient to schema changes

Richer querying and indexing (compared to K/V stores)

Fast

What's it good for?

Mobile and web apps – metadata for content-driven apps

User data store – user profile and behavior

IoT – Queryable device and event store

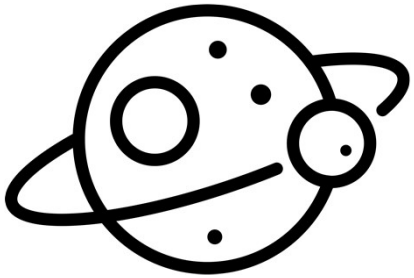
Workflow logging

Data ingestion when schema is variable

Product catalogs

Anything requiring a very flexible schema

Demo – SQL API



MongoDB API

MongoDB API

Move MongoDB apps to DocumentDB without changing code

No Server Management

Limitless Scale

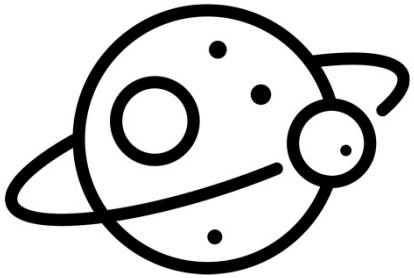
99.99% availability

Local data redundancy

Global replication

Use existing tools like MongoChef or Robo 3T (Formerly RoboMongo)

Demo – MongoDB API



Graph API

Graph API

Graphs store related data as Vertices and Edges

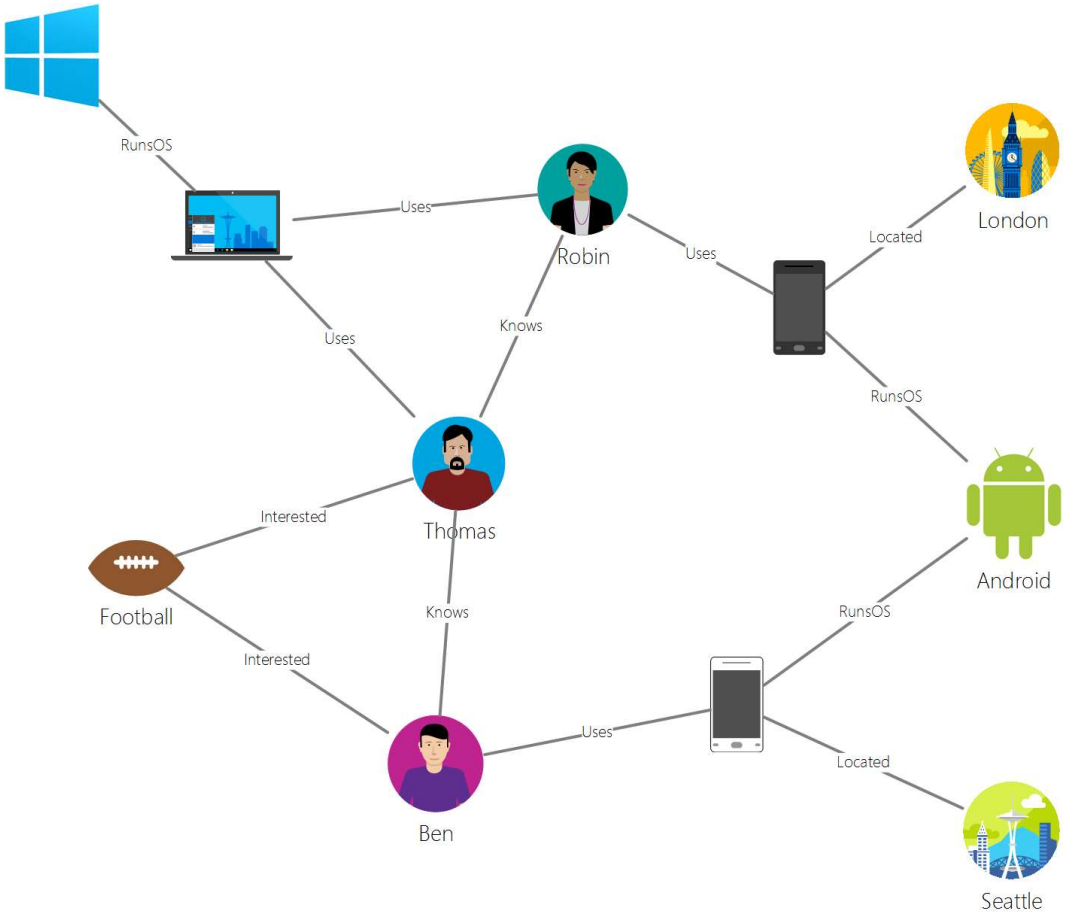
Vertices (or Nodes) represent entities

Edges represent directed relationships between Vertices

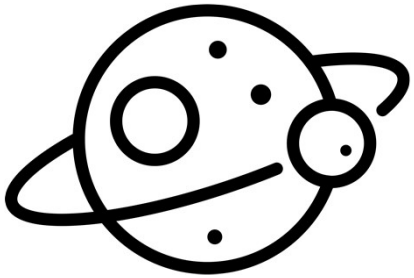
Ideal for answering relationship-based queries (social, properties, assemblies, etc.)

Think Kevin Bacon connector

Graph Example

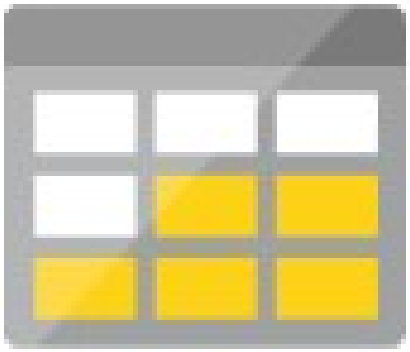


Demo – Graph API



Other API’s

Table Storage API



Premium alternative to Azure Table Storage

Store terabytes of structured data

Great for data that can be denormalized and queried by a primary key

Turnkey global distribution

Dedicated throughput worldwide

Single-digit millisecond latencies at the 99th percentile

Guaranteed high availability

Automatic secondary indexing

Cassandra API

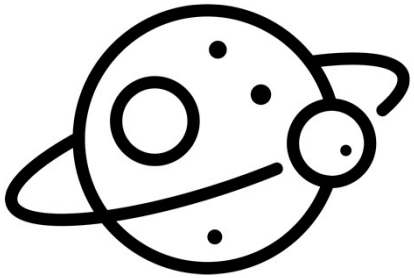
Column-family storage

No operations management

Easily migrate applications and use existing code and tools

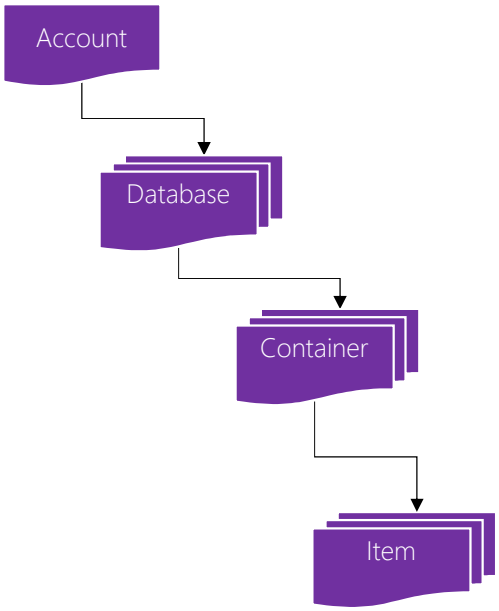
Use Cassandra Query Language based tools (like CQLSH)

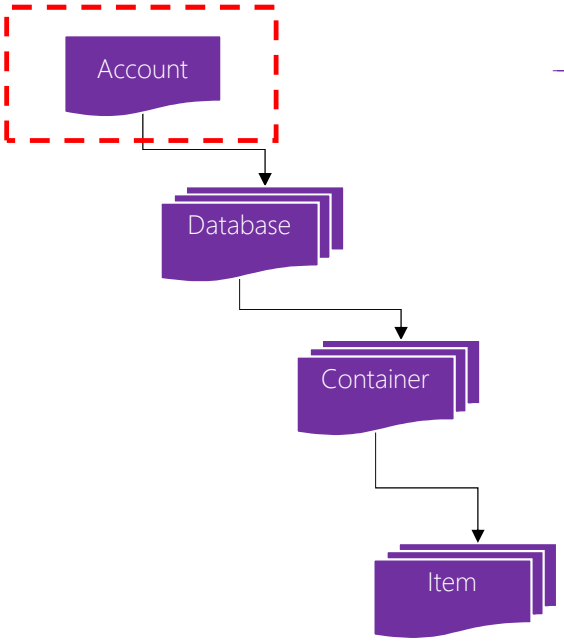
Use existing drivers



Resource Model

Cosmos DB Resource Model



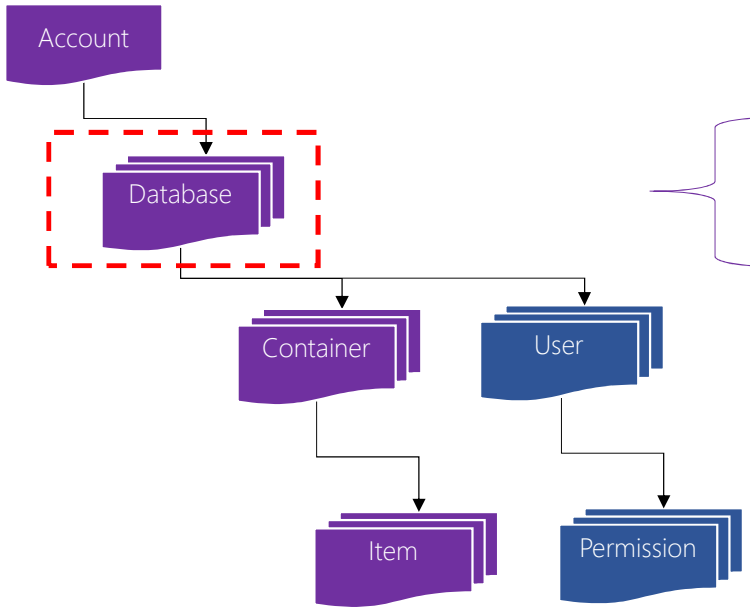


Contains one or more databases

Tied to a single Azure Subscription

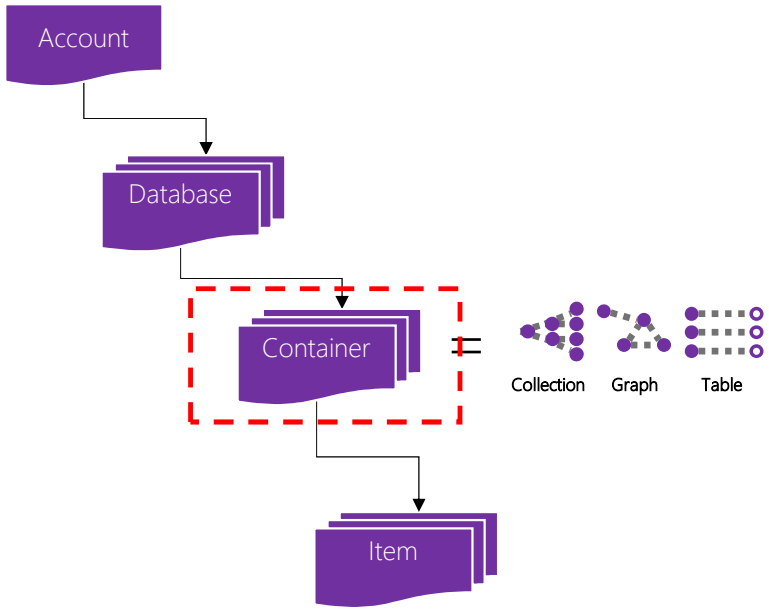
Firewall rules and replication

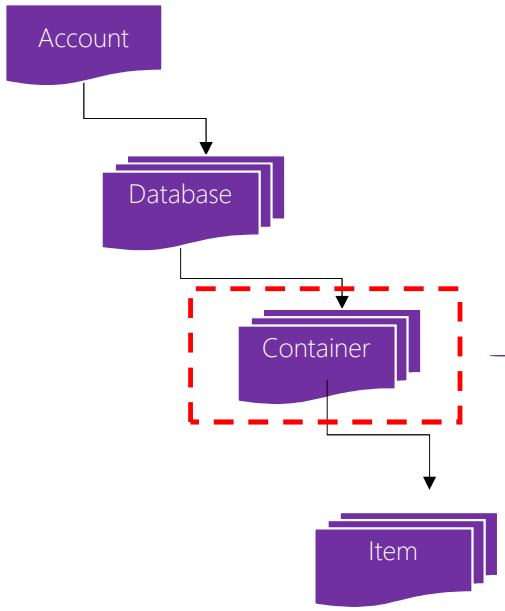
One API



Holds one or more containers (collections)

Also manages users and permissions


Account
Database
Container
=
Collection
Graph
Table
Item



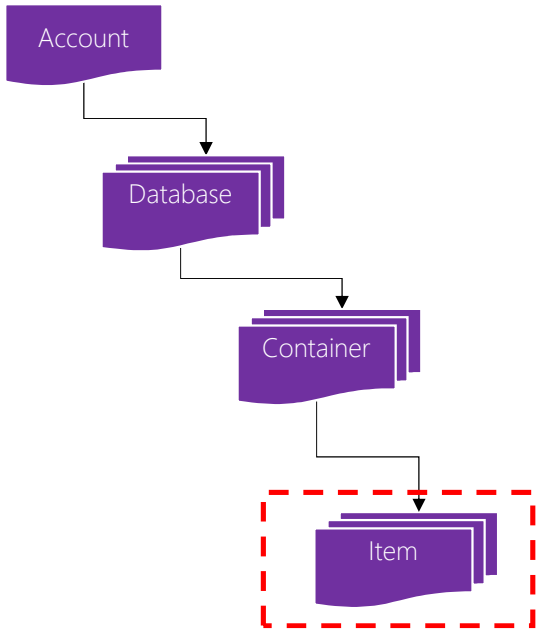
Containers are created with a specific API

Contains user-generated JSON documents

Provisioned throughput

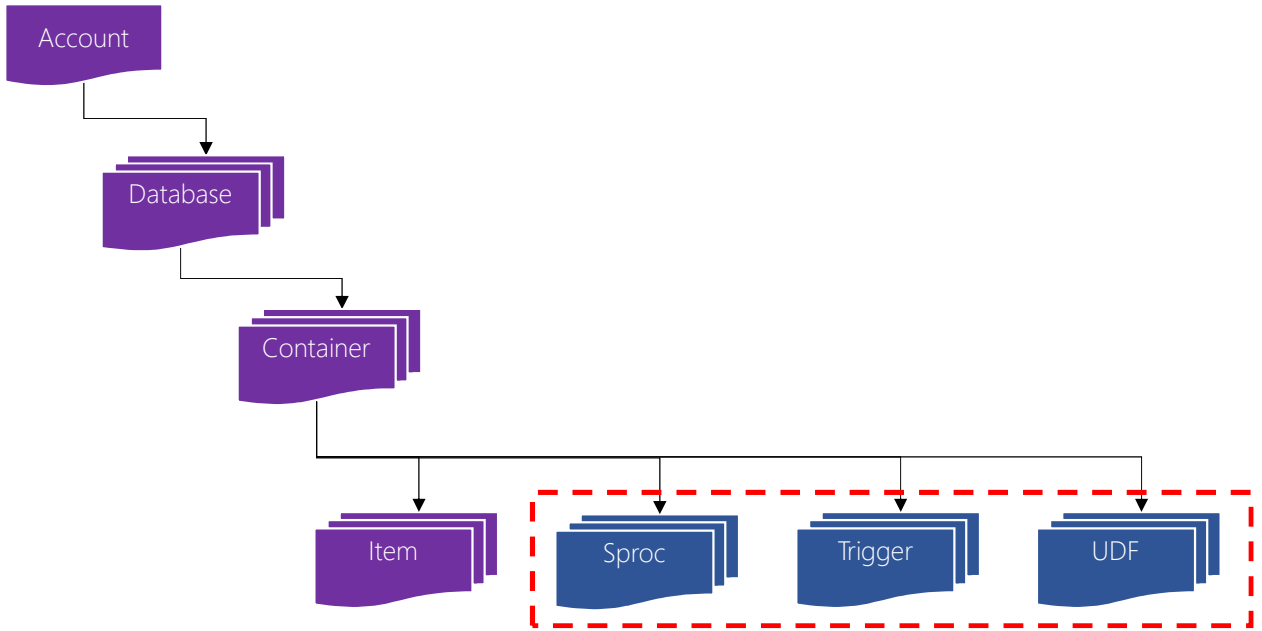
JavaScript stored procedures, UDF's and triggers

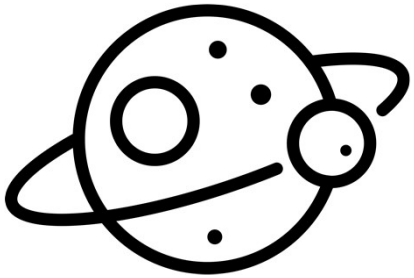
Transaction boundary



Items are your data expressed as JSON documents

No schema is enforced


Account
Database
Container
Item
Sproc
Trigger
UDF



Global Distribution

Turnkey Global Distribution

Worldwide presence as a Foundational Azure service

Automatic multi-region replication

Multi-homing APIs

Manual and automatic failovers

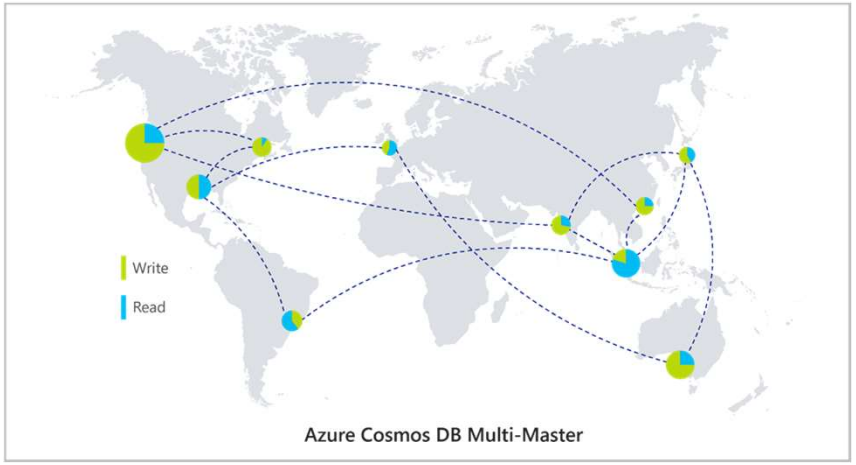
Designed for High Availability

Azure regions

Azure has more global regions than any other cloud provider—offering the scale needed to bring applications closer to users around the world, preserving data residency, and offering comprehensive compliance and resiliency options for customers.

54 regions worldwide **140** available in 140 countries

* Two Azure Government Secret region locations undisclosed



Azure Cosmos DB Multi-Master

Multi-master Support

Better disaster recovery, write availability and failover

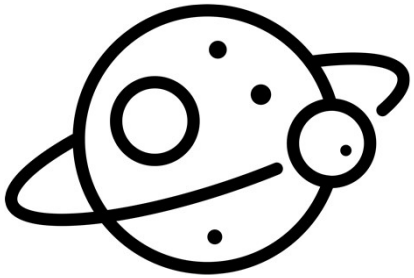
Improved write latency for end users

Improved write scalability and write throughput

Load balancing

Better use of provisioned capacity

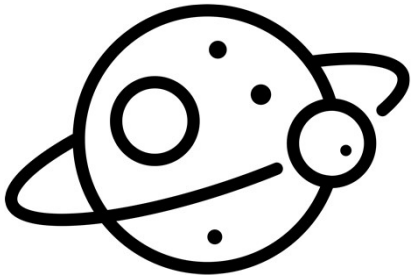
Simpler and more resilient architectures



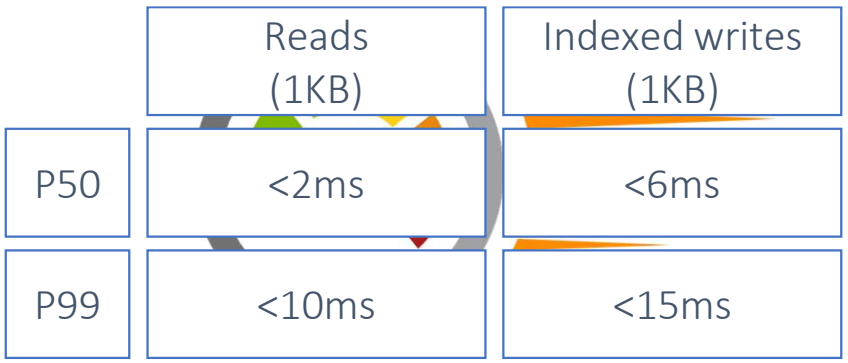
Consistency

5 consistency levels let you make predictable trade-offs between consistency, availability, and performance

Strong	Bounded Staleness	Session	Consistent Prefix	Eventual
Writes can't be read until durably committed	Reads can lag behind writes by a specified limit	You are guaranteed to read your own writes	Reads may lag behind writes but read order is guaranteed	Weakest consistency but lowest latency



Performance and Throughput



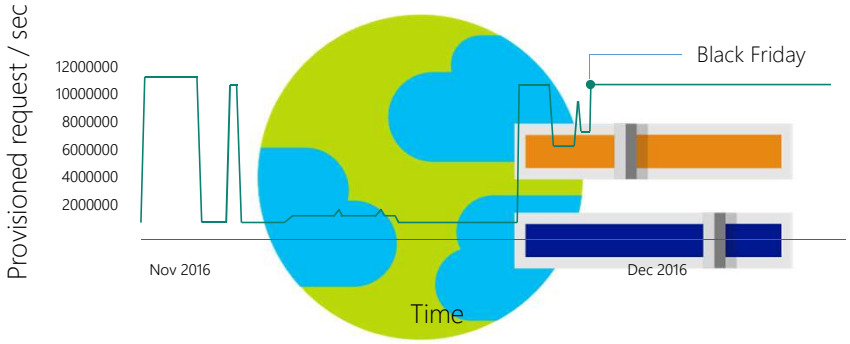
Guaranteed low latency at P99 (99th percentile)

Requests are served from local region

Single-digit millisecond latency worldwide

Write optimized, latch-free database engine designed for SSD

Synchronous automatic indexing at sustained ingestion rates



Hourly throughput (request/sec)

Elastically scalable storage and throughput

Single machine is never a bottle neck

Transparent server-side partition management

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

Automatic expiration via policy based TTL

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

Request Units

RU's measure the amount of reserved resources available to your Collection

You are charged by how much capacity you reserve

Reserving capacity ensures predictable performance

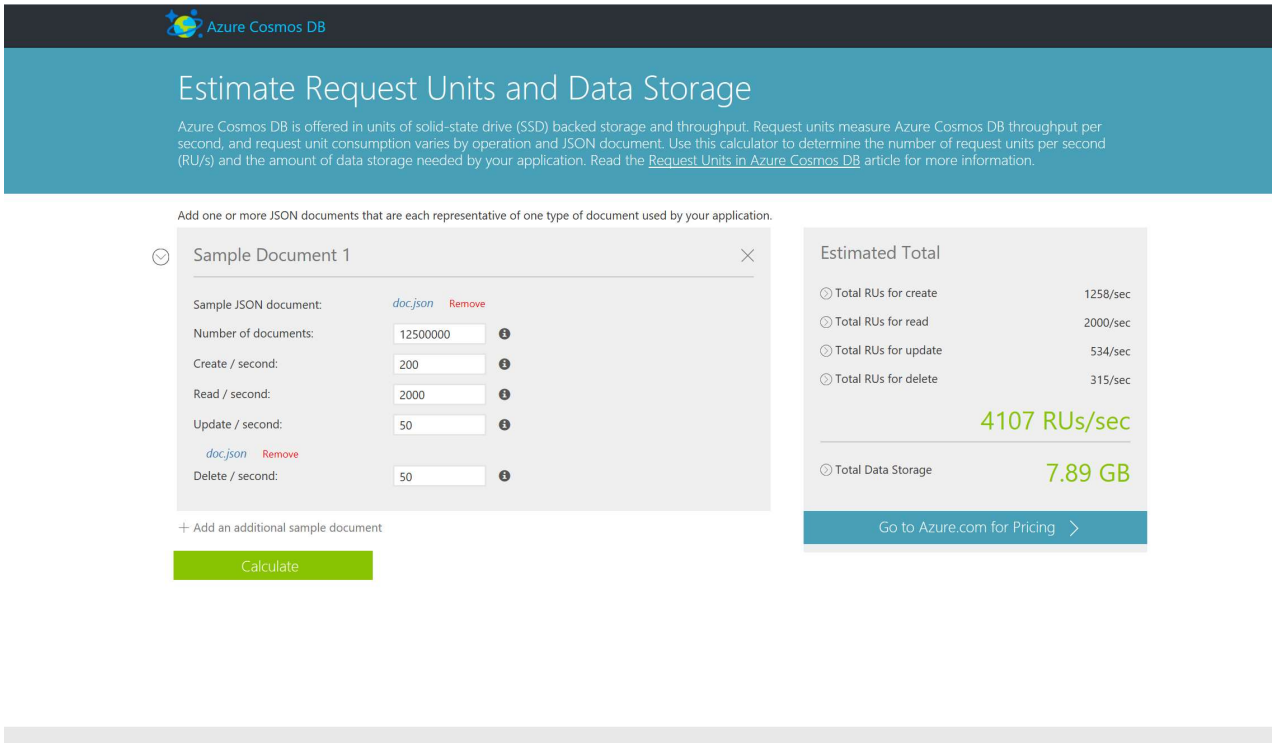
Allocated in RUs per second

Every operation incurs a Request Charge expressed in RUs

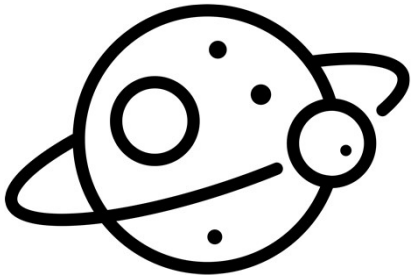
Request Charges are impacted by…

- Document Size
- Property Count
- Consistency Level
- Indexed Properties
- Indexing Policy
- Query Patterns
- Script Usage

Request Unit Calculator



https://www.documentdb.com/capacityplanner



Indexing

Indexing

At global scale ALTER TABLE, CREATE INDEX, DROP INDEX are a non-starter

Logical index layouts (inverted, tree, columnar, …)

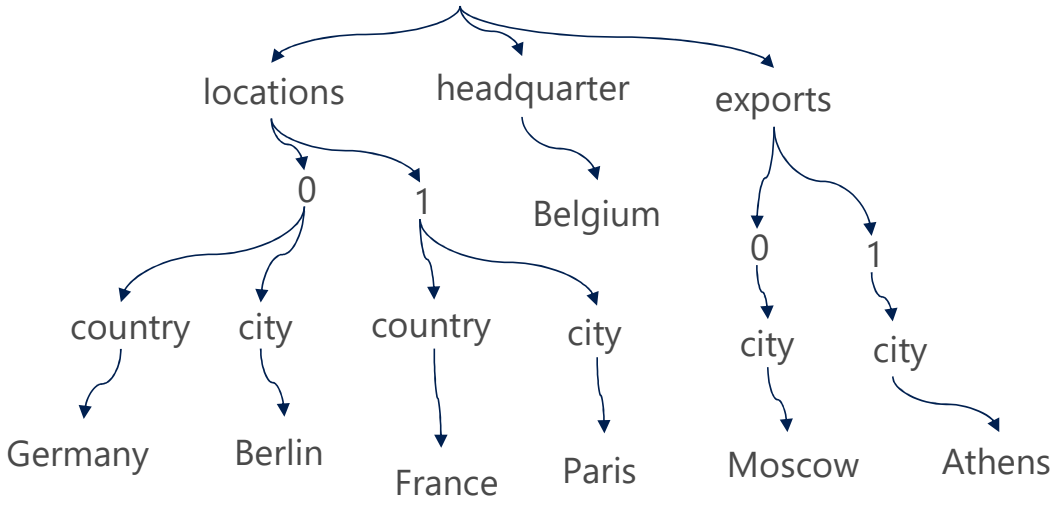
Automatic and synchronous indexing of all ingested content

Hash, Range, Geo-spatial and Columnar

No schemas or secondary indices ever needed

Resource governed, write optimized database engine

```
{
  "locations":
  [
    { "country": "Germany", "city": "Berlin" },
    { "country": "France", "city": "Paris" }
  ],
  "headquarter": "Belgium",
  "exports":[{"city":"Moscow"}, {"city":"Athens"}]
}
```



http://www.vldb.org/pvldb/vol8/p1668-shukla.pdf

Indexing Modes

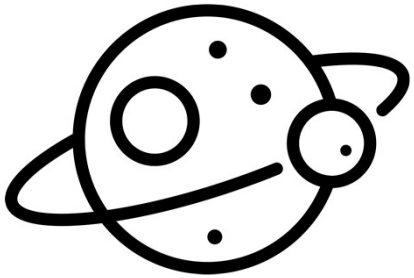
Consistent	Lazy	None
Default Mode Index updated synchronously on writes	Indexing lags behind writes Good for bulk insert scenarios	Only the ID is indexed Best for key/value scenarios

All properties are indexed by default

You can include or exclude paths to customize

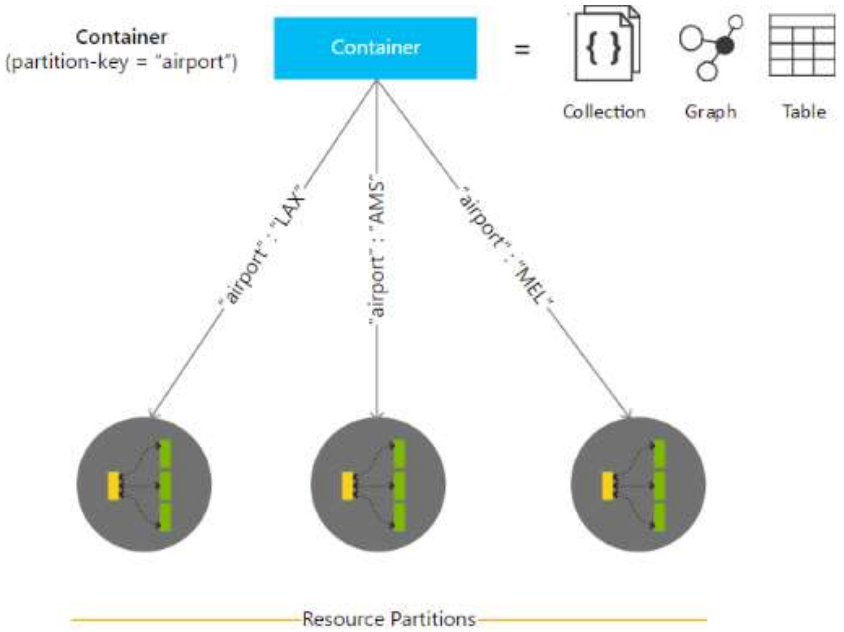
Remember, indexing impacts RU consumption

Index types include Hash (strings and numbers), Range (numbers) and Geo-spatial



Other Features

Partitioning



Cosmos DB partitions your data across multiple logical partitions allocated to physical partitions

A physical partition is a fixed amount of SSD storage that is replicated for high-availability

A logical partition is a sub-division of a physical partition that stores all data for a single partition key value

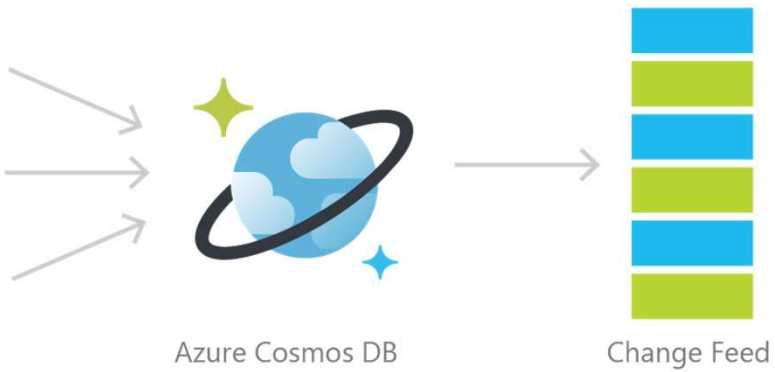
Partition management is fully automated

Your only job is to select a good partition key

Partition keys should…

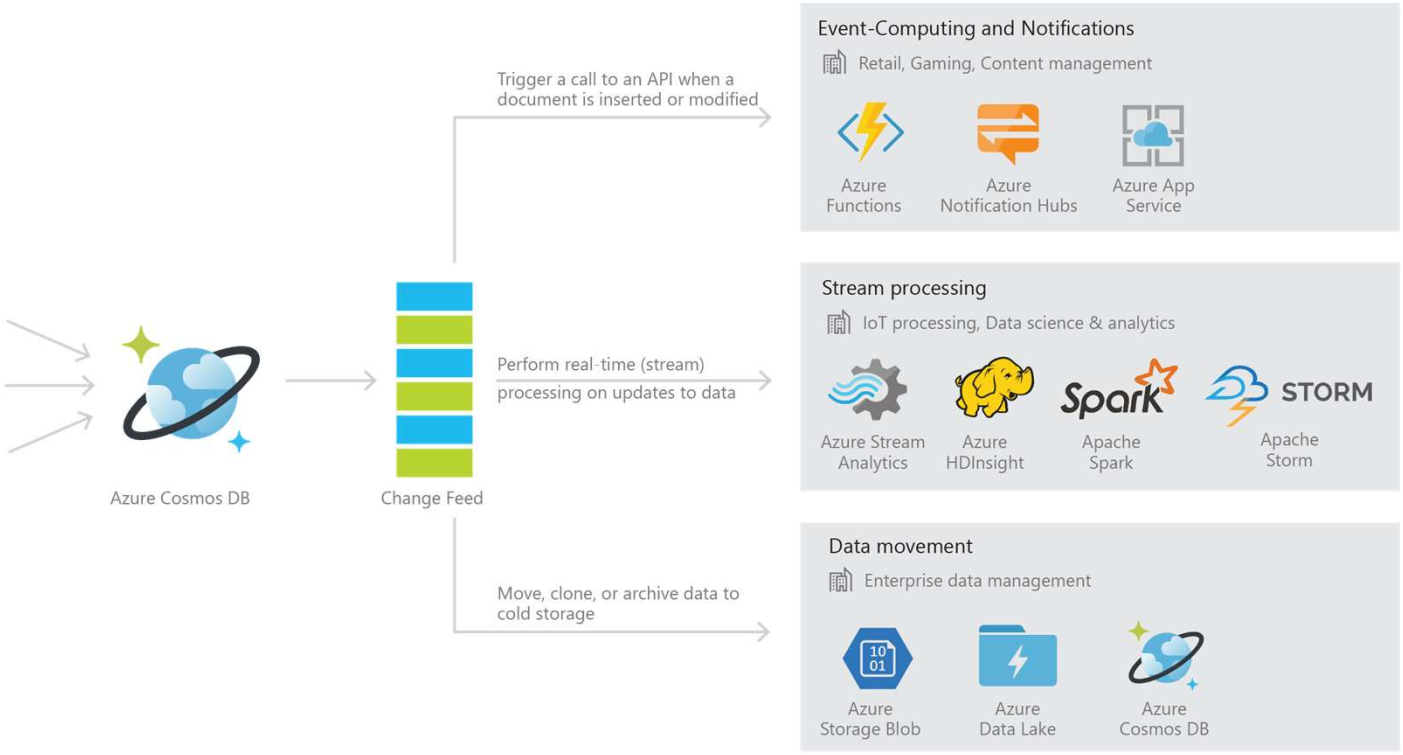
- Have a large number of values
- Be chosen to distribute data as evenly as possible

Azure Cosmos DB Change Feed



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

Common Change Feed Use Cases



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

Other features and capabilities

Automatically expire data (TTL)

Unique Keys

Azure Functions integration

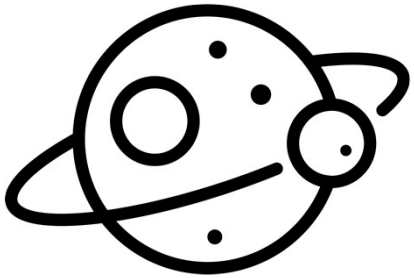
Azure Search integration

VNET Service Endpoints

Automatic Backups

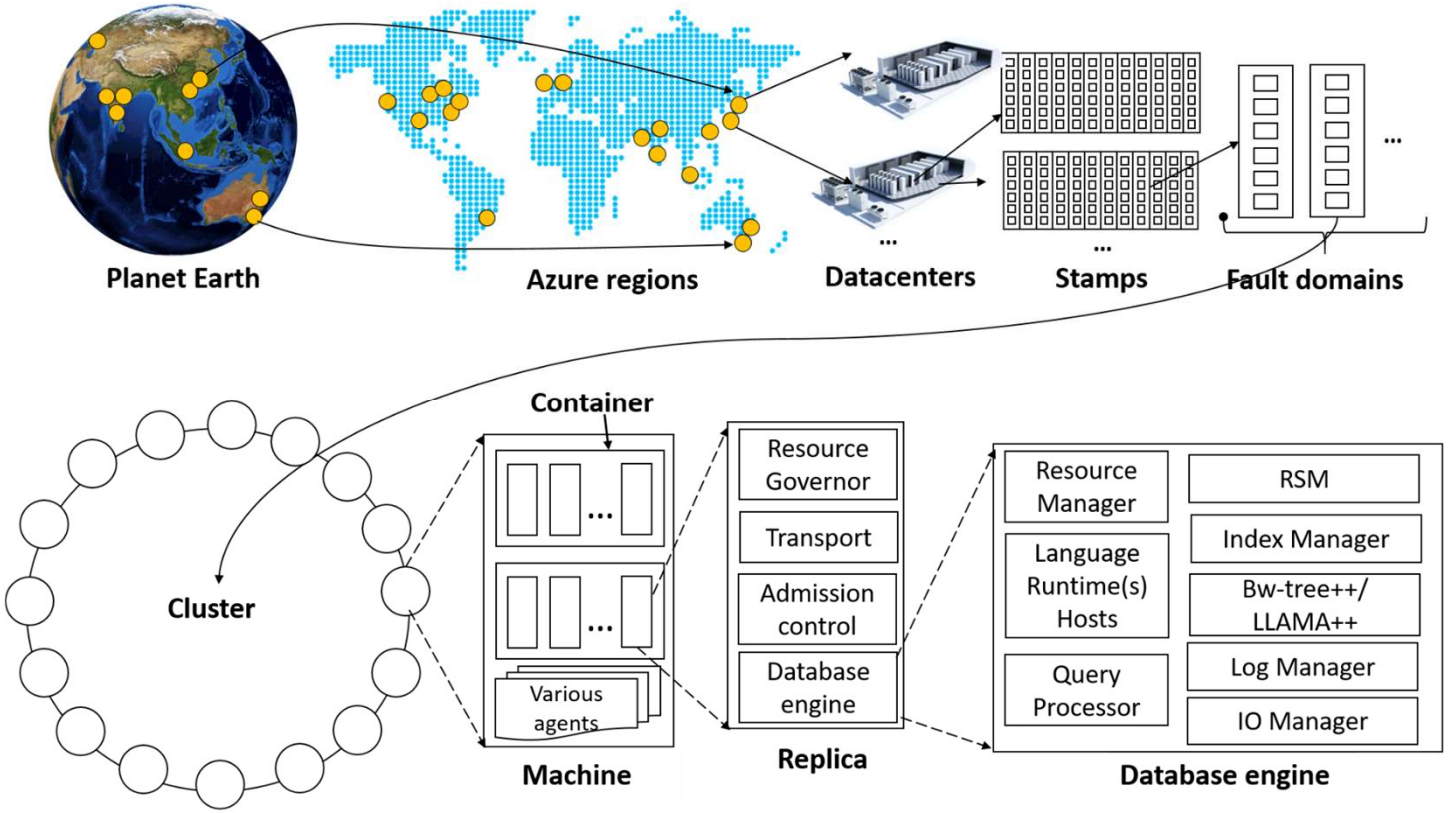
Automatic Regional Failover

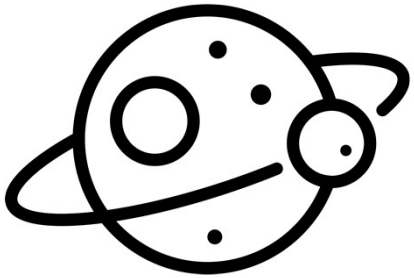
Bulk Executor Library for batch operations



System Internals

System topology (behind the scenes)



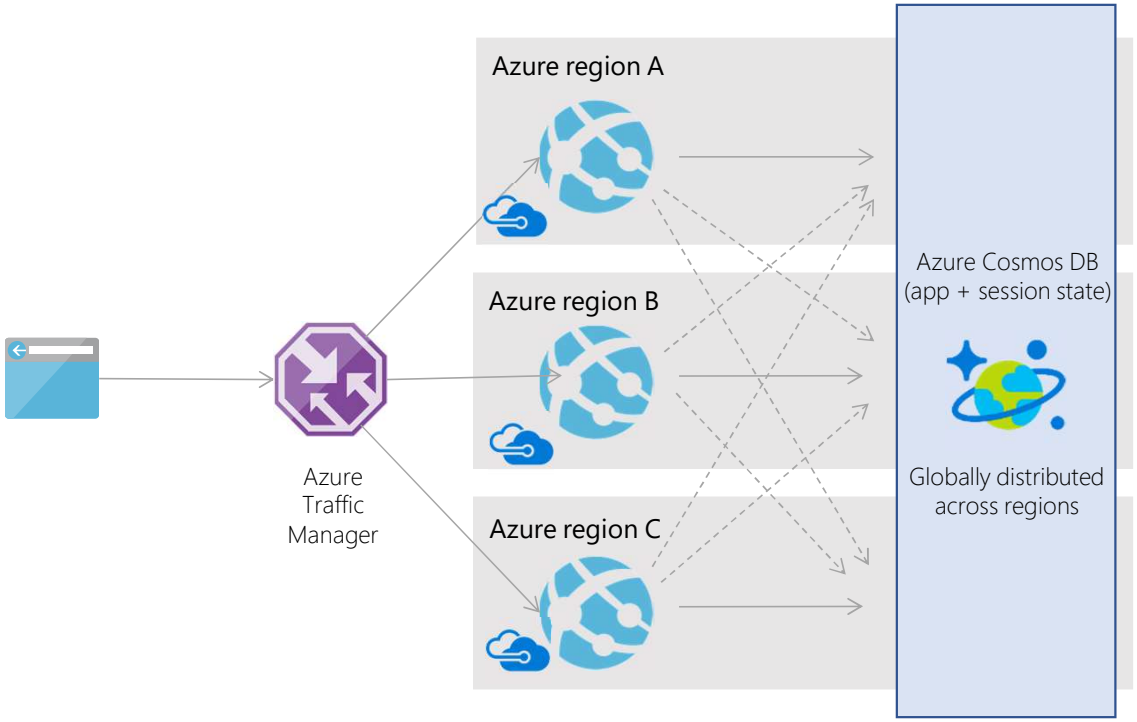


Customer Use Cases and Architectures

Globally Distributed Apps

- **Business Needs:**
  - High scalability to support massive user base
  - Low latency access across multiple geographies to build highly responsive applications
  - Automatic indexing over flexible schemas to support rich queries over user-defined content (e.g. custom form builder)
  - High availability across multiple data centers

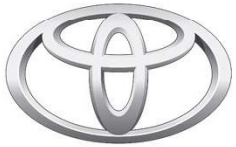
Globally distributed apps



Internet of Things – Telemetry & Sensor Data

- **Business Needs:**
  - High scalability to ingest large # of events coming from many devices
  - Low latency queries and changes feeds for responding quickly to anomalies
  - Schema-agnostic storage and automatic indexing to support dynamic data coming from many different generations of devices
  - High availability across multiple data centers

Microsoft Azure



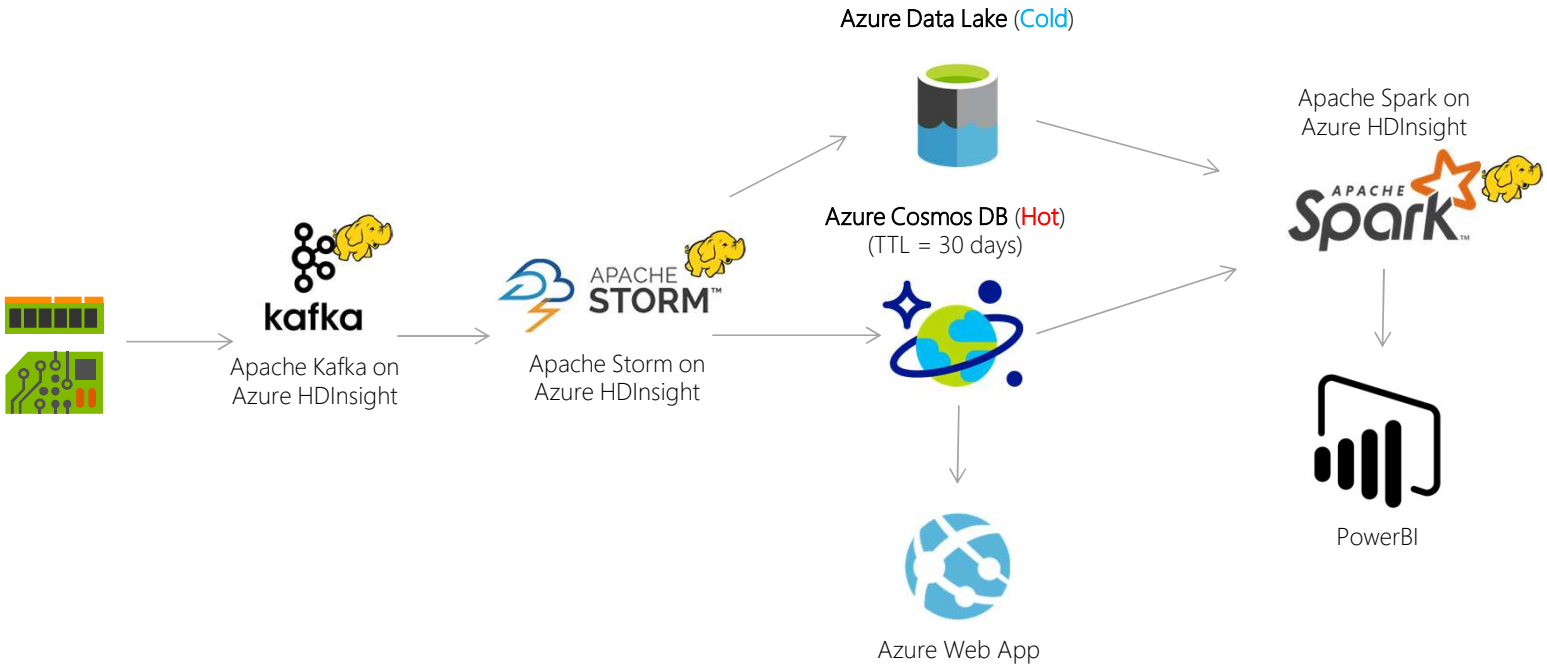
TOYOTA

Honeywell

LG CNS

Johnson Controls

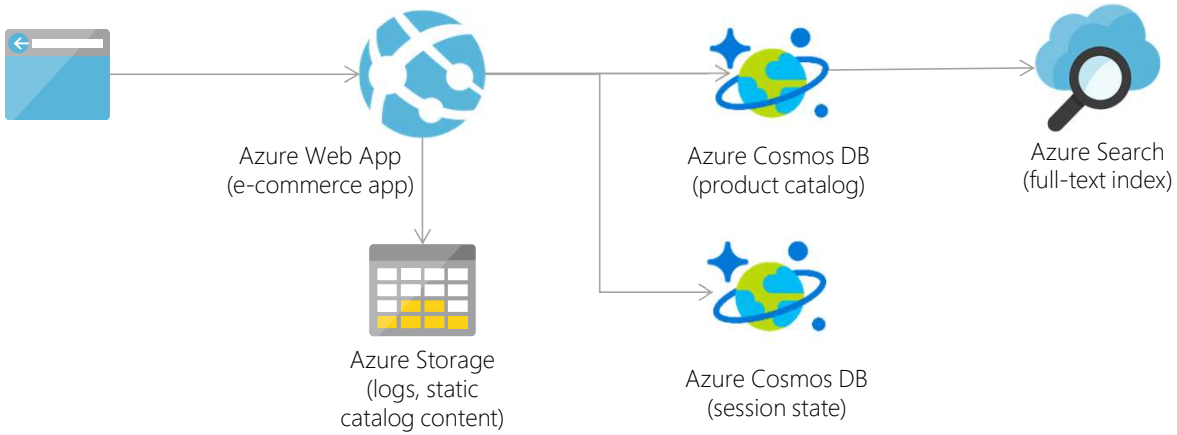
Internet of Things – Telemetry & Sensor Data



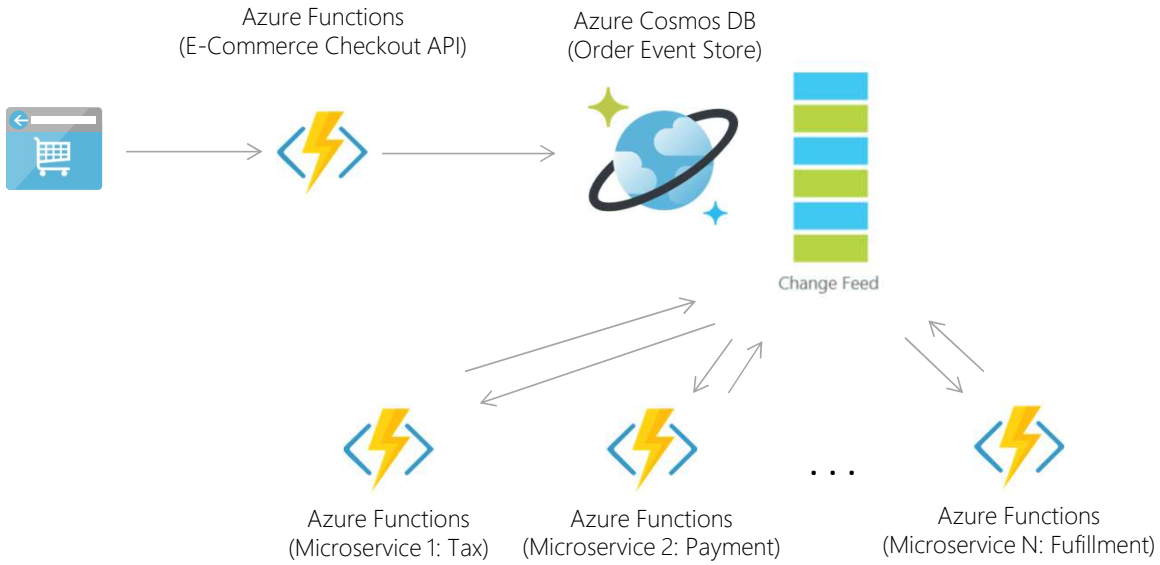
Retail – Product Catalog & Order Processing

- **Business Needs:**
  - Elastic scale to handle seasonal traffic (e.g. Black Friday)
  - Low-latency access across multiple geographies to support a global user-base and latency sensitive workloads (e.g. real-time personalization)
  - Schema-agnostic storage and automatic indexing to handle diverse product catalogs, orders, and events
  - High availability across multiple data centers

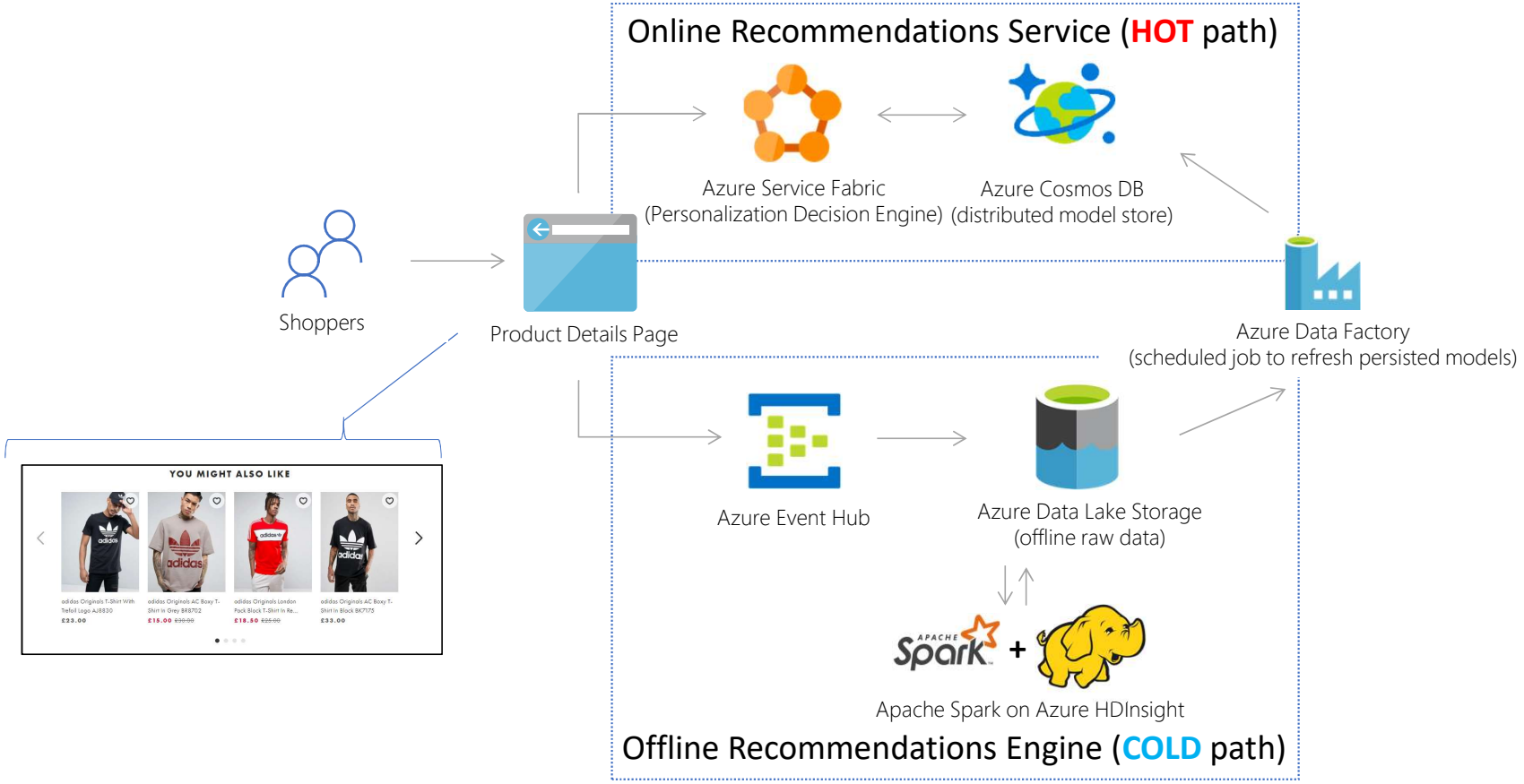
Retail Product Catalogs



Retail Order Processing Pipelines



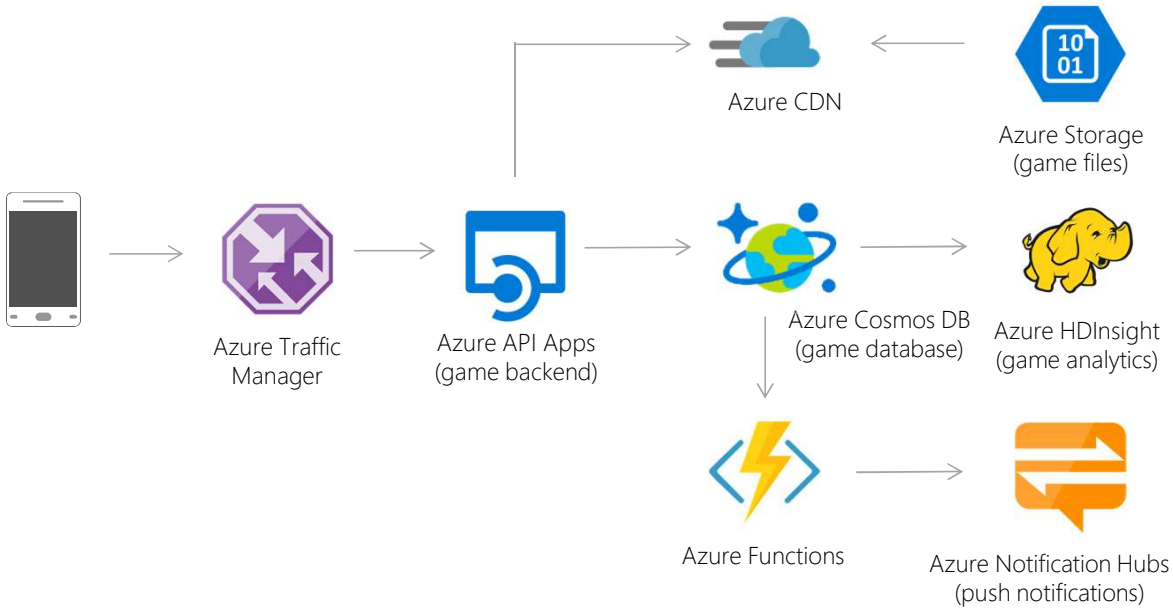
Real-time Recommendations



Multiplayer Gaming

- **Business Needs:**
  - Elastic scale to handle bursty traffic on day
  - Low-latency queries to support responsive gameplay for a global user-base
  - Schema-agnostic storage and indexing allows teams to iterate quickly to fit a demanding ship schedule
  - Change-feeds to support leaderboards and social gameplay

Multiplayer Gaming

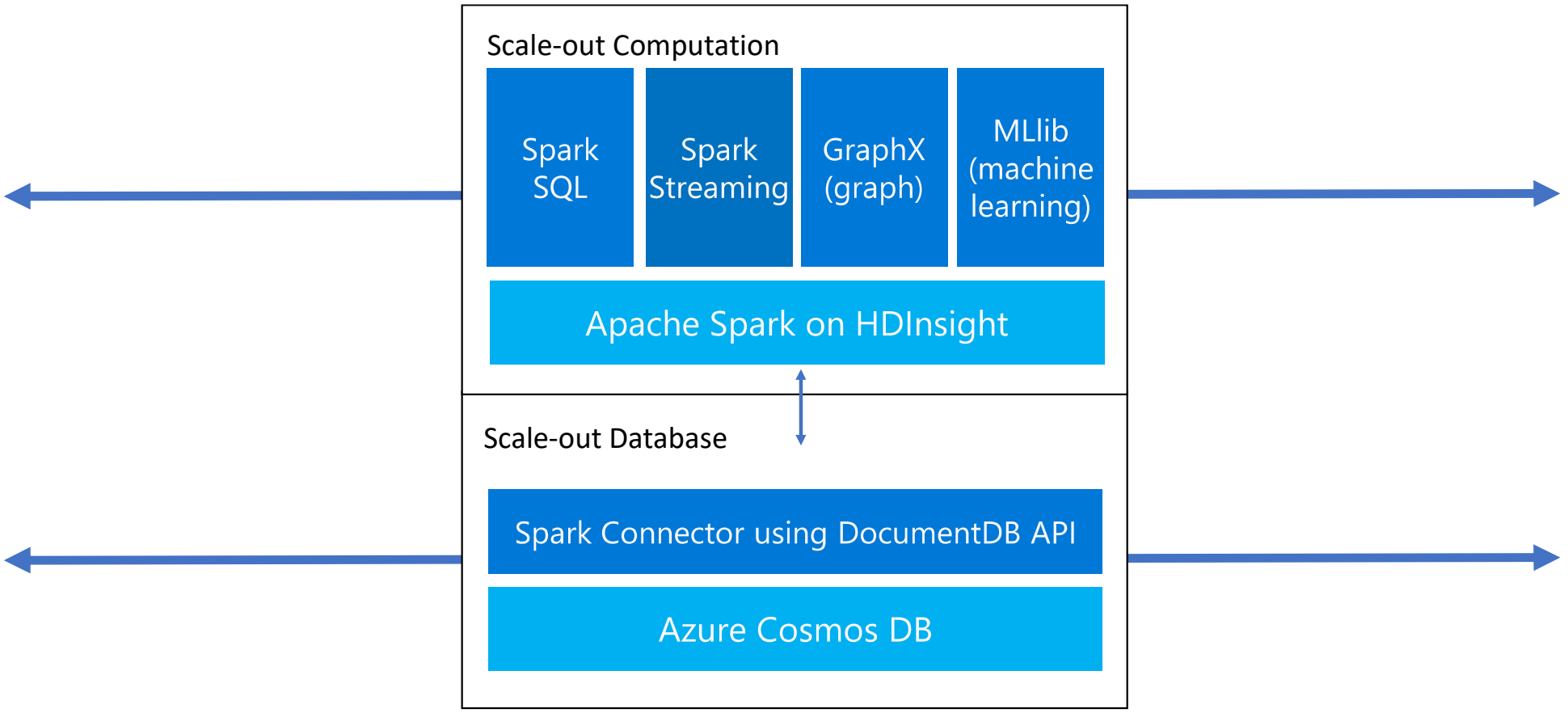


Customer 360 + Operational Analytics

- **Business Needs:**
  - High scalability to handle large user base
  - Rich queries and automatic indexing over flexible schemas to consolidate data from a variety of sources
  - Built-in aggregates and efficient spark connector to analyze user behavior, drive insights, and build a single view of the customer.

Affinio

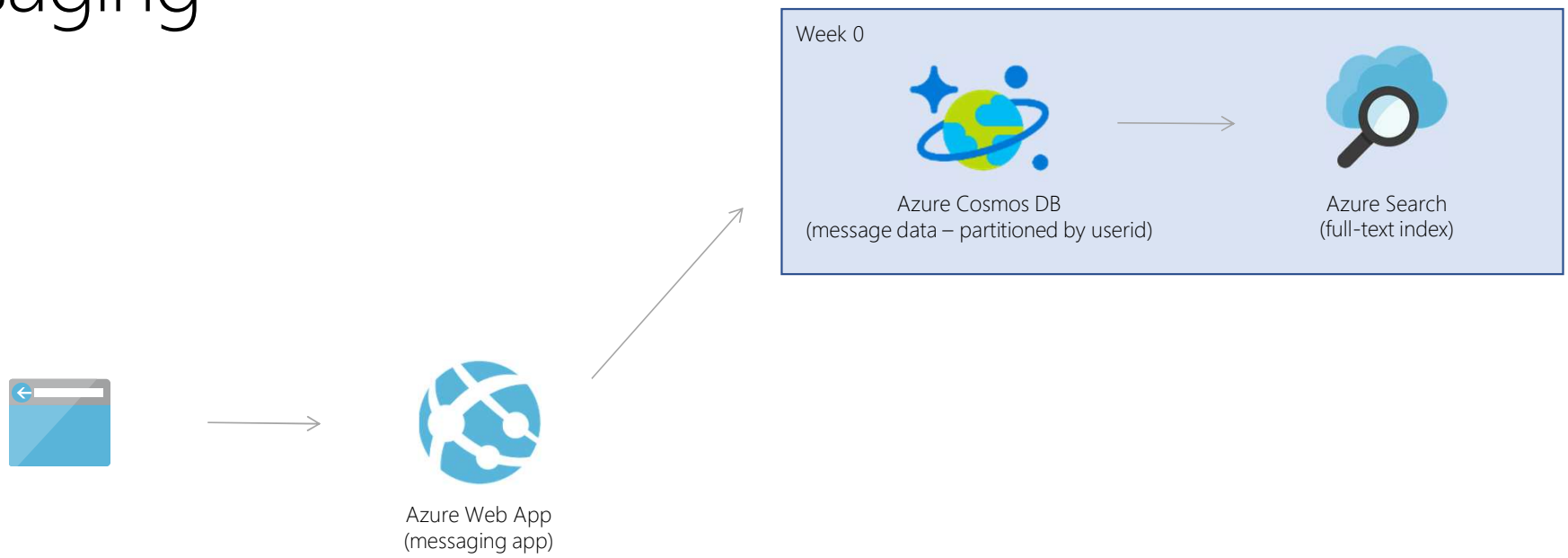
Customer 360 + Social Analytics


Scale-out Computation
Spark SQL
Spark Streaming
GraphX (graph)
MLlib (machine learning)
Apache Spark on HDInsight
Scale-out Database
Spark Connector using DocumentDB API
Azure Cosmos DB

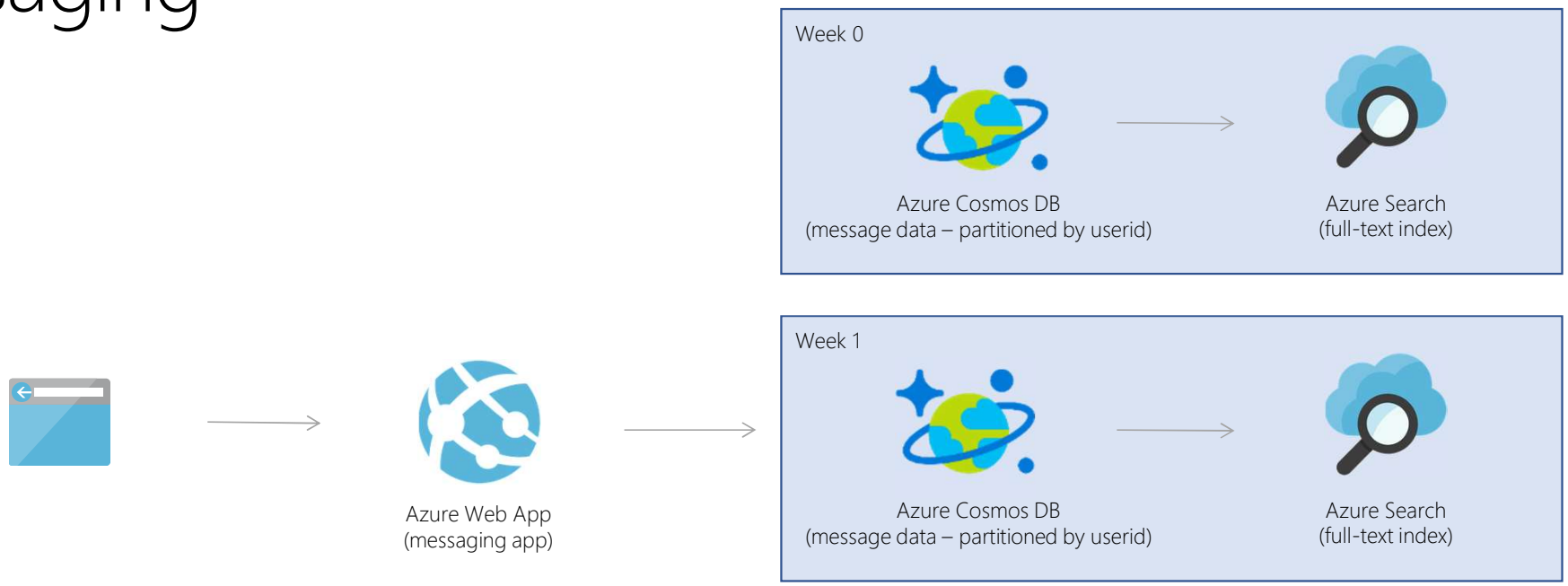
Messaging

- **Business Needs:**
  - High scalability to ingest and store messages from a massive user base
  - Low latency access across multiple geographies to support global user-base
  - High availability across multiple data centers

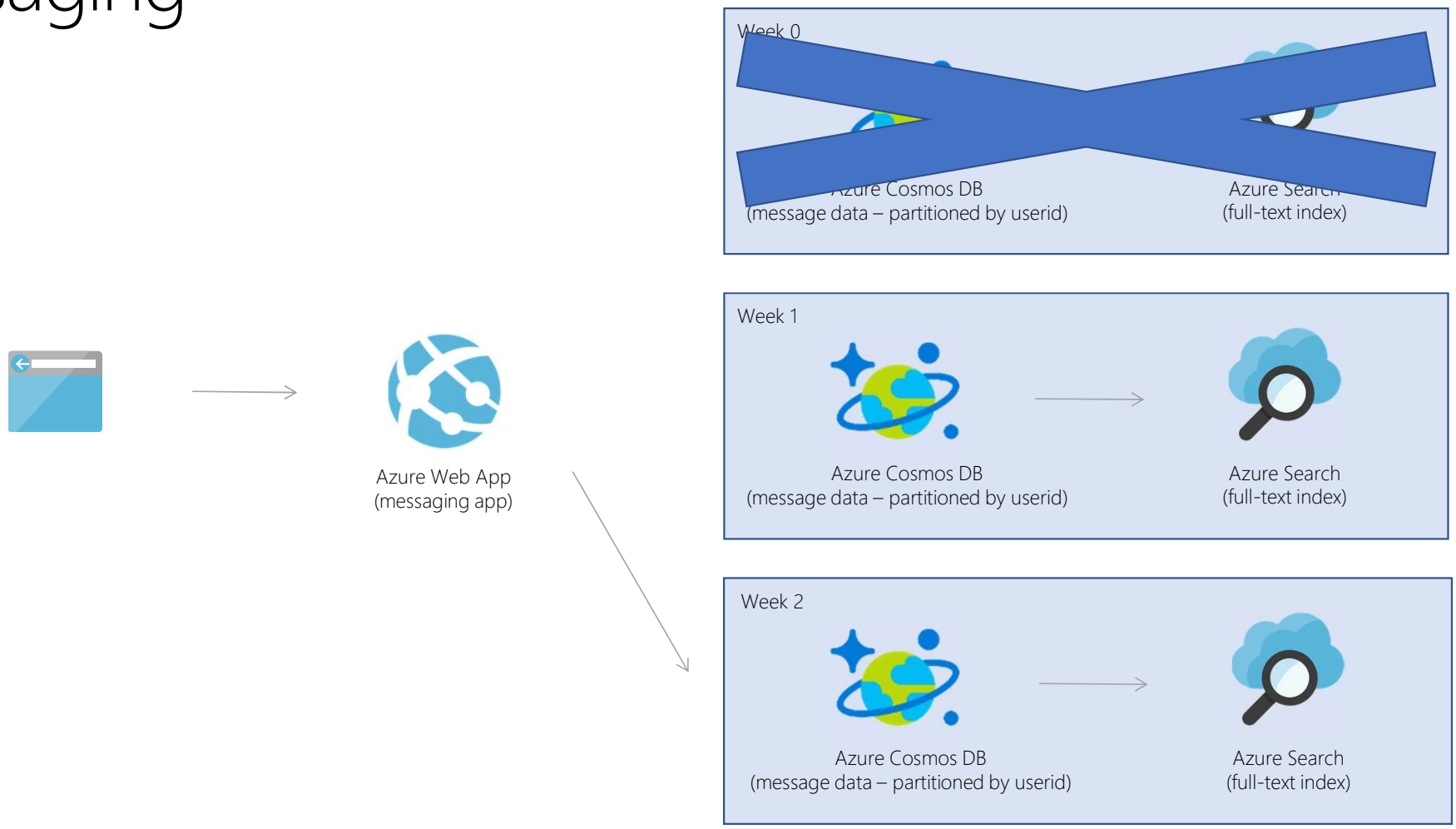
Messaging

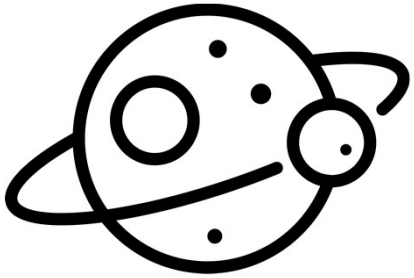

Week 0
Azure Cosmos DB
(message data – partitioned by userid)
Azure Search
(full-text index)
Azure Web App
(messaging app)

Messaging


Week 0
Azure Cosmos DB
(message data – partitioned by userid)
Azure Search
(full-text index)
Week 1
Azure Cosmos DB
(message data – partitioned by userid)
Azure Search
(full-text index)
Azure Web App
(messaging app)

Messaging


Week 0
Azure Cosmos DB
(message data – partitioned by userid)
Azure Search
(full-text index)
Week 1
Azure Cosmos DB
(message data – partitioned by userid)
Azure Search
(full-text index)
Azure Web App
(messaging app)
Week 2
Azure Cosmos DB
(message data – partitioned by userid)
Azure Search
(full-text index)



Thank you!

cosmosdb.com

Follow @AzureCosmosDB
Use #CosmosDB

stack overflow

#azure-cosmosdb