

NoSQL 資料庫服務 Azure Cosmos DB

賦能程式開發

Eva Zhao (趙陽)

Microsoft, Asia Data & AI Senior Specialist

Agenda



- Begin with Azure Cosmos DB
- Deep dive inside Azure Cosmos DB
- Integrate with Azure Cosmos DB

Azure Cosmos DB Highlight



Supports “all” popular NoSQL databases as managed DBaaS that provides “limitless” scalability, low latency and 5-nines availability with financially backed SLAs



Support all popular NoSQL databases



Native and open API support for MongoDB, Cassandra, Gremlin data and models

API for
MongoDB



SQL
Core API



Guaranteed speed and <10 ms latency anywhere



Guaranteed <10 ms latency backed by SLA. Automatic scaling with instant limitless elasticity.



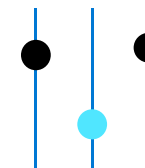
99.999% availability with zero RPO and RTO



Guaranteed 99.999% availability, zero RPO & RTO, backed by SLA and enterprise-level security.



Fully managed DBaaS experience



Auto-sharding, auto-scaling and auto-indexing for blazing fast queries and low TCO

<https://aka.ms/5whyCosmos>

When you begin with Azure Cosmos DB.....



1. Create Azure Cosmos DB Account with an **API** and a **Capacity Model**
2. Create Database and Container with a configuration of **"Throughput"** and **"partition key" / "Shard key"**
3. Connect from application to insert/update items.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with the Microsoft Azure (Preview) logo, a search bar, and various icons. Below the navigation bar, the breadcrumb trail reads 'Dashboard > Create a resource >'. The main heading is 'Create an Azure Cosmos DB account'. Below this, a section titled 'Which API best suits your workload?' provides a brief description of Azure Cosmos DB as a fully managed NoSQL and relational database service. It then presents six options for creating a new account, each with a description and 'Create' and 'Learn more' buttons:

- Azure Cosmos DB for NoSQL**: Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.
- Azure Cosmos DB for MongoDB**: Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.
- Azure Cosmos DB for Apache Cassandra**: Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB.
- Azure Cosmos DB for Table**: Fully managed database service for apps written for Azure Table storage. Recommended if you have existing Azure Table storage workloads that you plan to migrate to Azure Cosmos DB.
- Azure Cosmos DB for Apache Gremlin**: Fully managed graph database service using the Gremlin query language, based on Apache TinkerPop project. Recommended for new workloads that need to store relationships between data.
- Azure Cosmos DB for PostgreSQL**: Fully-managed relational database service for PostgreSQL with distributed query execution, powered by the Citus open source extension. Build new apps on single or multi-node clusters—with support for JSONB, geospatial, rich indexing, and high-performance scale-out.

APIs in Azure Cosmos DB



Azure Cosmos DB for NoSQL
Azure Cosmos DB for MongoDB

Document Database

```
Customer_1167
{
  "_id": "1167",
  "Name": "Lisa Andrews",
  "OrderTerms": "ProductID:1285",
  "OrderDate": "7/7/2013",
  "OrderTotal": "120",
  "Address": "888 Front St, Boise, ID",
  "ad"
}
```

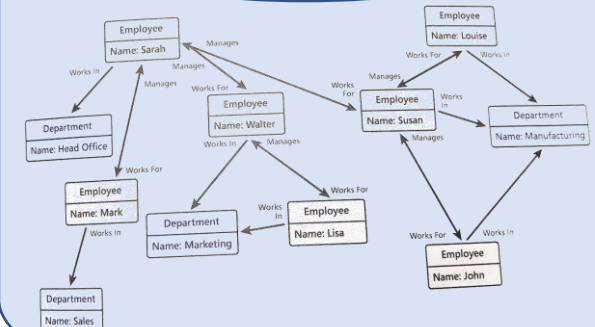
Key/Value Store

Key	Value
Name	Albert
Age	56
Sex	Male
Education	Master
Occupation	Teacher
Yearly Income	100000

Column-family Database

Key	Column Families			
	CustomerInfo		AddressInfo	
CustomerID	CustomerInfo:Title	Mr	AddressInfo:StreetAddress	990 500 Ave
1	CustomerInfo:FirstName	Mark	AddressInfo:City	Bellevue
	CustomerInfo:LastName	Hanson	AddressInfo:State	WA
			AddressInfo:ZipCode	12345
2	CustomerInfo:Title	Mrs	AddressInfo:StreetAddress	888 W. Front St
	CustomerInfo:FirstName	Lisa	AddressInfo:City	Boise
	CustomerInfo:LastName	Andrews	AddressInfo:State	ID
			AddressInfo:ZipCode	22153
3	CustomerInfo:Title	Mr	AddressInfo:StreetAddress	990 500 Ave
	CustomerInfo:FirstName	Walter	AddressInfo:City	Bellevue
	CustomerInfo:LastName	Harp	AddressInfo:State	WA
			AddressInfo:ZipCode	54321

Graph Database



Azure Cosmos DB for Table

Azure Cosmos DB for Apache Gremlin

APIs in Azure Cosmos DB



NOW AVAILABLE

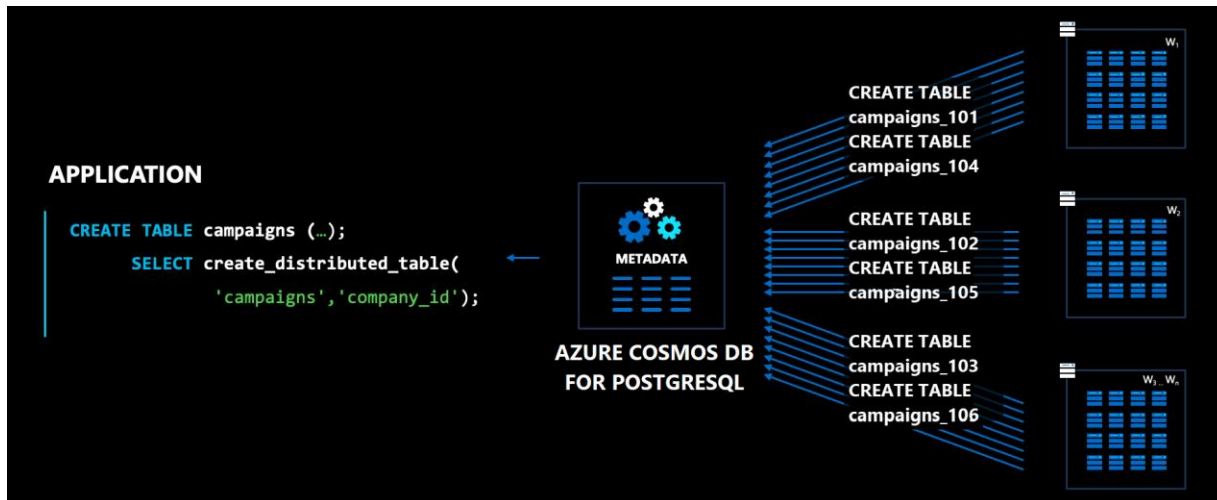
Generally available: Azure Cosmos DB for PostgreSQL

Published date: October 12, 2022

Azure Cosmos DB for PostgreSQL is a fully managed Distributed SQL service that allows you to build modern cloud native applications at any scale. The service is built on open-source PostgreSQL at the core and the Citus extension to horizontally scale workloads. This complements the existing NoSQL APIs in Azure Cosmos DB to provide a common data foundation for a variety of applications including multi-tenant SaaS, IoT, and OLTP apps.

[Learn more.](#)

[Azure Cosmos DB](#) [Azure Database for PostgreSQL](#) [Features](#) [Management](#) [Services](#) [Open Source](#) [Microsoft Ignite](#)



Microsoft Azure Search resources, services, and docs (G+/I)

Home >

azure-cosmos-db-for-postgresql-with-pg15 Azure Cosmos DB for PostgreSQL Cluster

Delete Reset password Upgrade Restore Restart Feedback

Essentials JSON View

Resource group : [nik1020](#) Coordinator name : c.azure-cosmos-db-for-postgresql-with-pg15.postgres...

Location : East US Admin username : citus

Subscription : [Azure SQL DB Project Orcas - CitusData](#) Connectivity method : [Public access \(allowed IP addresses\)](#)

Subscription ID : [\[redacted\]](#) Coordinator node : [96 vCores / 384 GiB RAM, 0.5 TiB storage](#)

PostgreSQL version : 15 Worker nodes : [4 nodes, 32 vCores / 1,024 GiB RAM, 0.5 TiB storage](#)

Citus version : 11.1 Replication role : [Primary](#)

Configuration : [Multi-node configuration](#)

Tags (edit) : [Click here to add tags](#)

Nodes (5) Monitoring Recommendations (0)

Looking for metrics? [Click here](#)

Name	Type	Status	High availability	Availability zone	Fully qualified domain name
azure-cosmos-db-for-po...	Coordinator	Available	No	3	c.azure-cosmos-db-for-postgresql-with...
azure-cosmos-db-for-...	Worker	Available	No	3	w0.azure-cosmos-db-for-postgresql-wit...
azure-cosmos-db-for-...	Worker	Available	No	3	w1.azure-cosmos-db-for-postgresql-wit...
azure-cosmos-db-for-...	Worker	Available	No	3	w2.azure-cosmos-db-for-postgresql-wit...
azure-cosmos-db-for-...	Worker	Available	No	3	w3.azure-cosmos-db-for-postgresql-wit...

Create and use Cosmos DB with simple steps



1. Create Azure Cosmos DB Account with an **API** and a **Capacity Model**
2. Create Database and Container with a configuration of **"Throughput"** and **"partition key" / "Shard key"**
3. Connect from application to insert/update items.

The screenshot shows the 'Create Azure Cosmos DB Account - Core (SQL)' wizard. The 'Project Details' section includes a dropdown for 'Subscription' (Microsoft Azure Internal Consumption) and a dropdown for 'Resource Group' (Create new). The 'Instance Details' section includes a text input for 'Account Name' and a dropdown for 'Location' ((US) West US). The 'Capacity mode' section has two radio buttons: 'Provisioned throughput' (selected) and 'Serverless'. Below this, there is a checkbox for 'Apply Free Tier Discount' (Do Not Apply) and a checkbox for 'Limit total account throughput' (unchecked). At the bottom, there are three buttons: 'Review + create', 'Previous', and 'Next: Global Distribution'.

The screenshot shows the 'New Container' dialog in the Azure Cosmos DB Data Explorer. The 'Database id' field is empty, with a 'Create new' radio button selected. The 'Database throughput (autoscale)' section has 'Autoscale' selected and a 'Database Max RU/s' of 4000. The 'Container id' field contains 'e.g., Container1'. The 'Partition key' field contains 'e.g., /address/zipCode'. At the bottom, there is a '+ Add unique key' button.

Quickstarts



- Fastest path to “running code”
- “I am a developer who has never worked with Cosmos DB before”
- Quickstarts for each Cosmos DB API

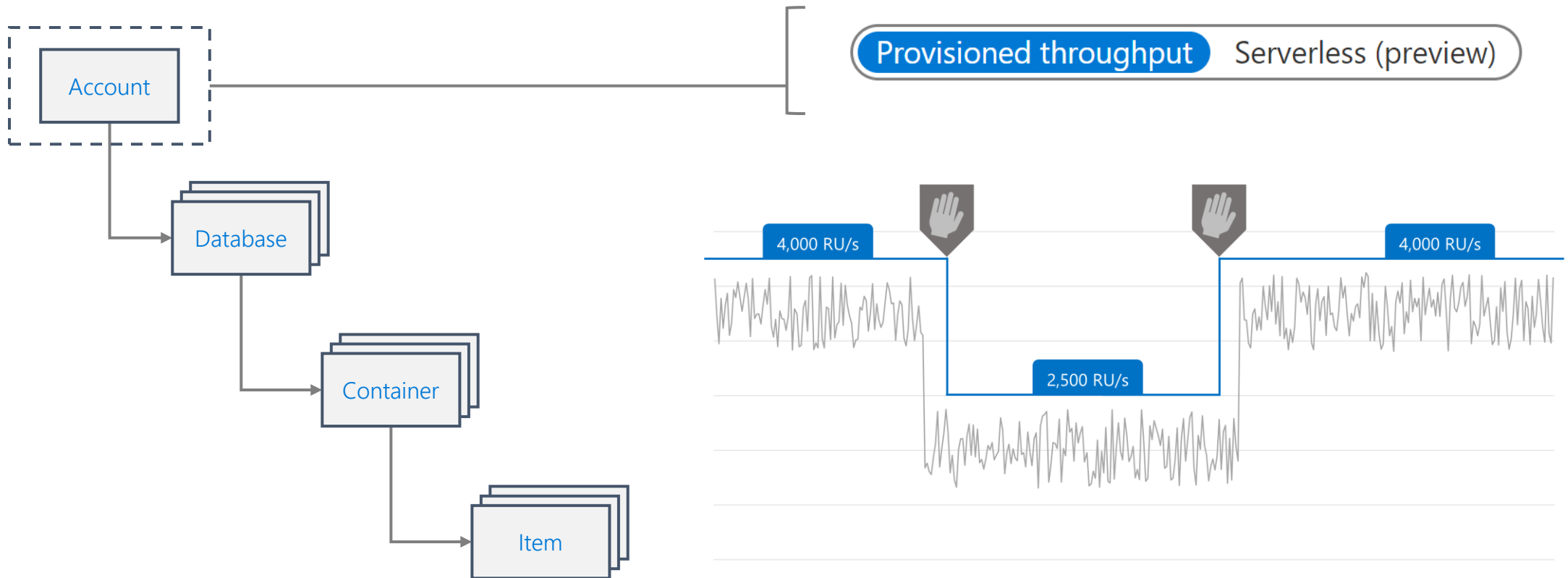
	Core/SQL	MongoDB	Cassandra	Gremlin	Table
.NET	Link	Link	Link	Link	Link
Java	Link	Link	Link	Link	Link
Node.js	Link	Link	Link	Link	Link
Python	Link	Link	Link	Link	Link
Go		Link			
PHP				Link	

Agenda



- Begin with Azure Cosmos DB
- Deep Dive Inside Azure Cosmos DB
 - Capacity Mode
 - Distribute and Scale Automatically
 - Replica Globally & Multi Master
- Integrate with Azure Cosmos DB

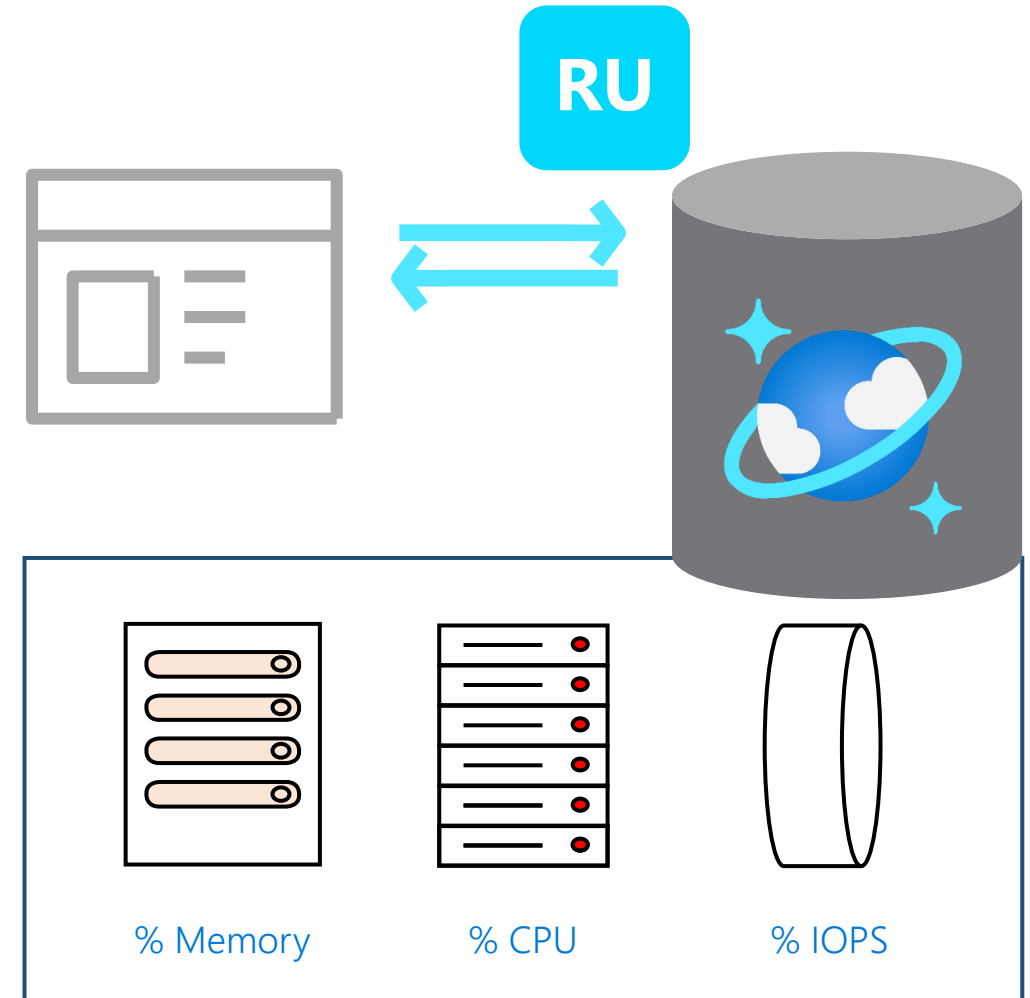
Capacity Mode



Provisioned Throughput is measured with RU



- Request Units (RUs) is a rate-based currency
- Abstracts physical resources for performing requests
- Key to multi-tenancy, SLAs, and cost efficiency
- Covers foreground and background activities



RU is mapping to every



- Normalized across various access methods
1 RU = 1 read of 1 KB item
- Each request consumes fixed RUs
- Applies to reads, writes, query, and stored procedures

RU

Database operations consume a variable number of RUs

Read =



1 RU

Insert =



Upsert =



Delete =



Query =



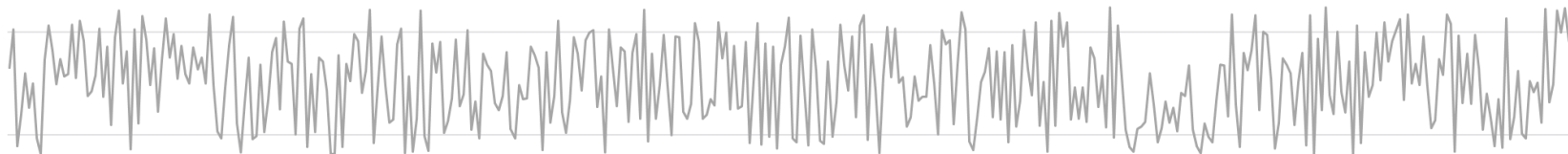
Variable number of RUs

Provisioned Throughput



Provisioned Throughput - Manual

4,000 RU/s

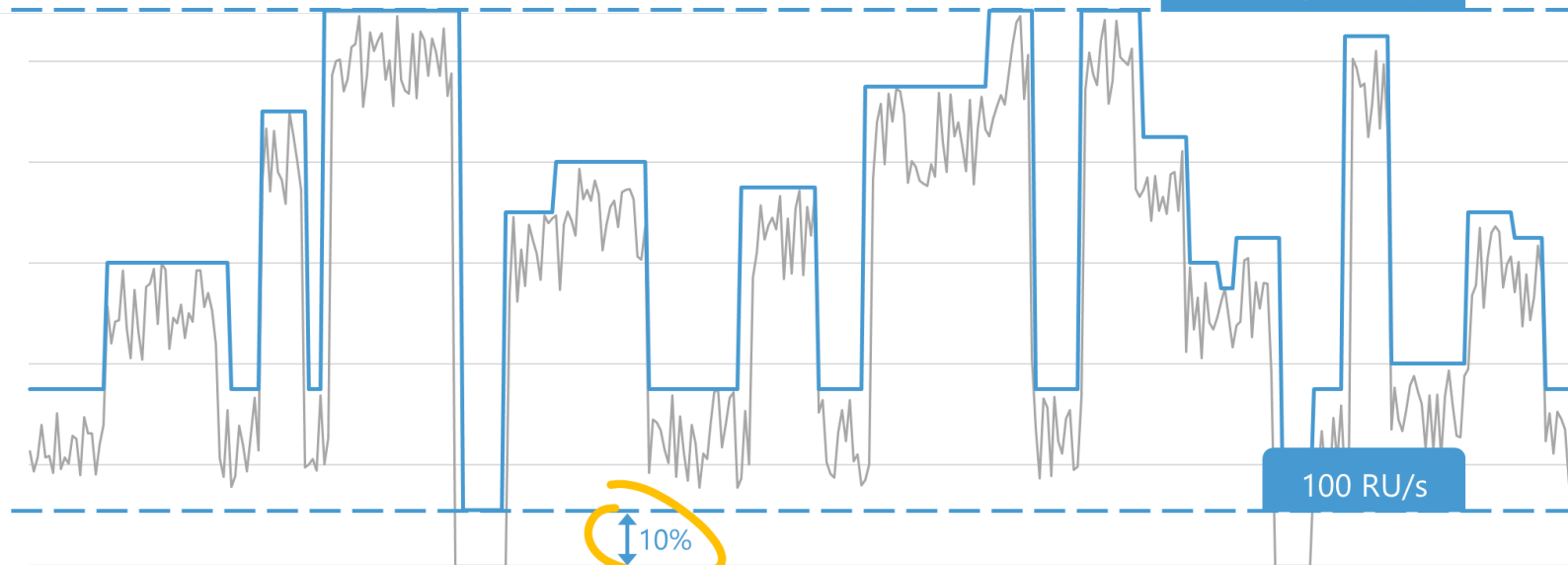


Provisioned Throughput - Autoscale

Max = 1,000 RU/s

100 RU/s

10%



Guaranteed throughput



Guaranteed low latency



Guaranteed availability

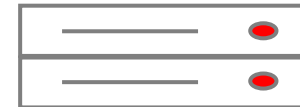
Cosmos DB is distributed and scaled by Partition



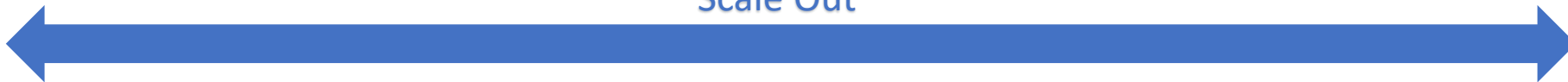
Container



...



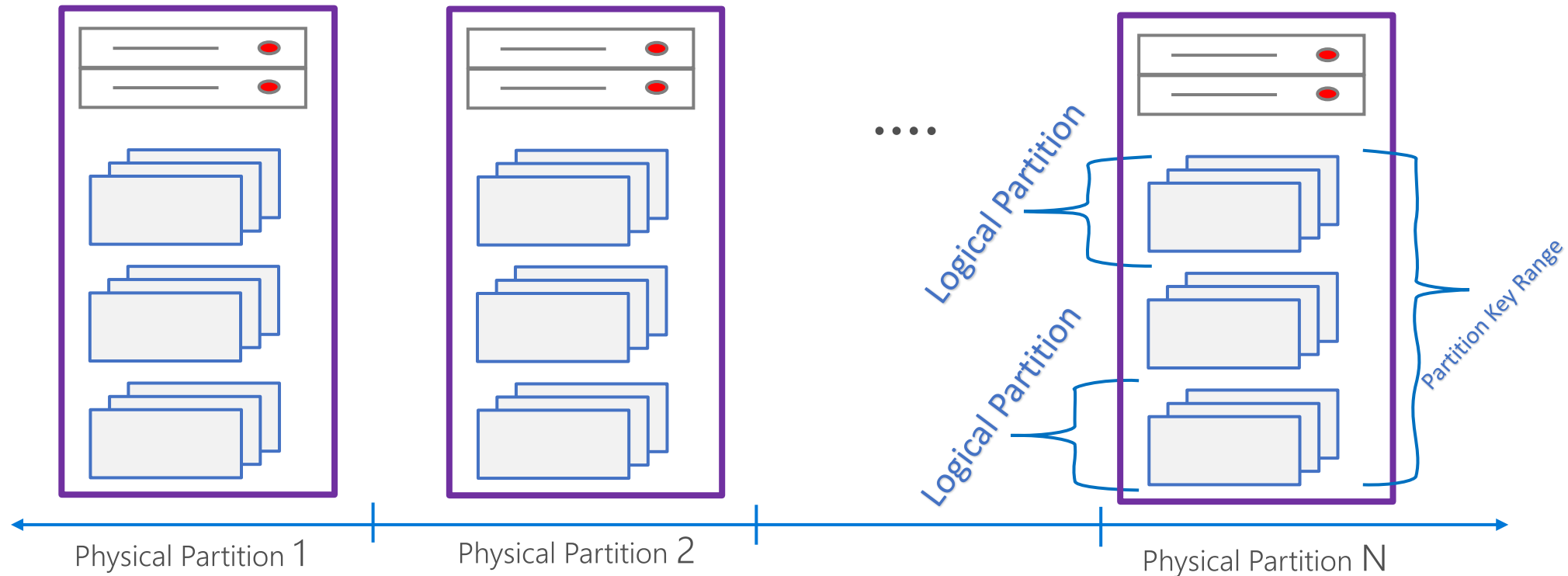
Scale Out



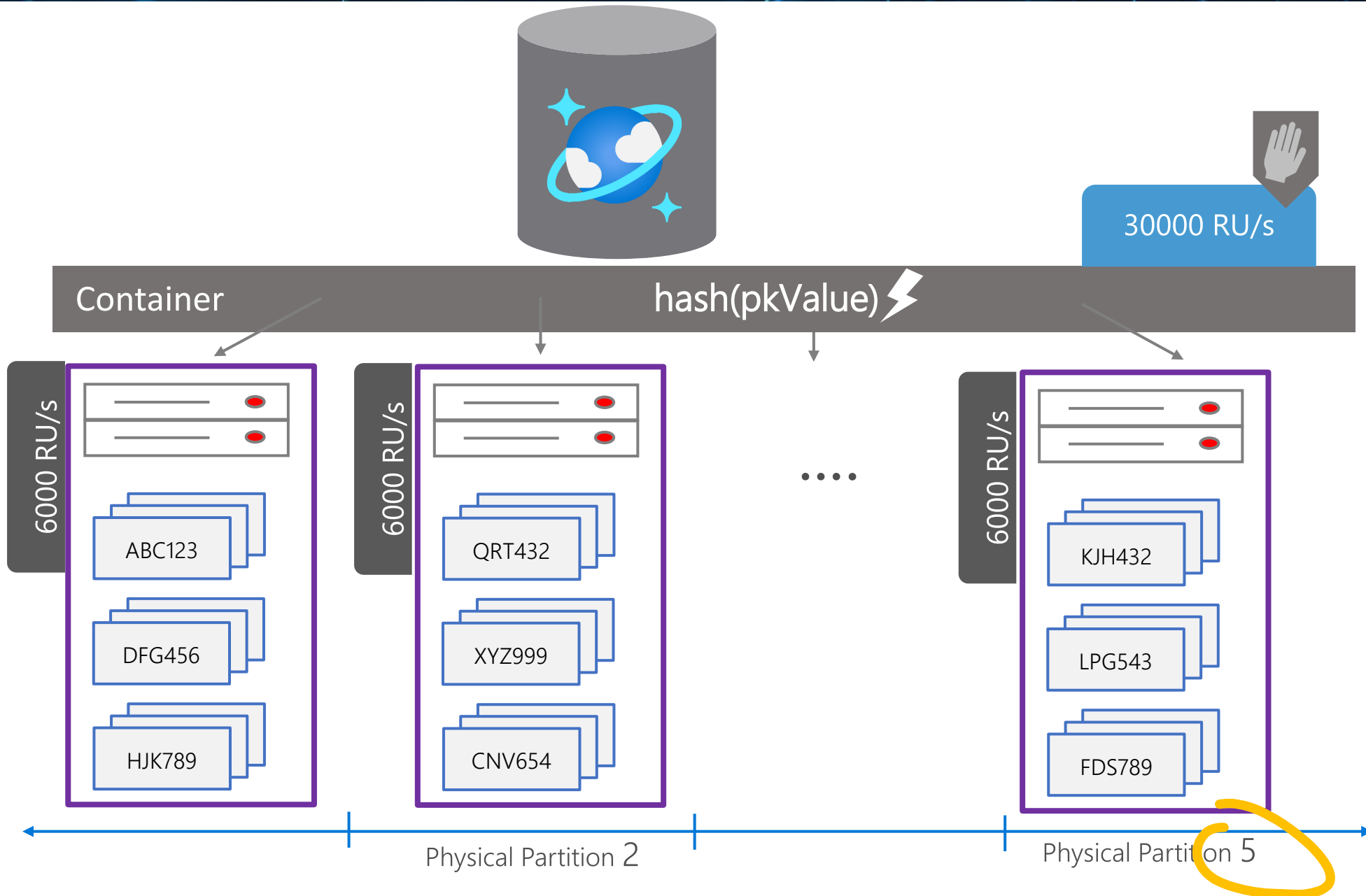
Physical Partition & Logical Partition



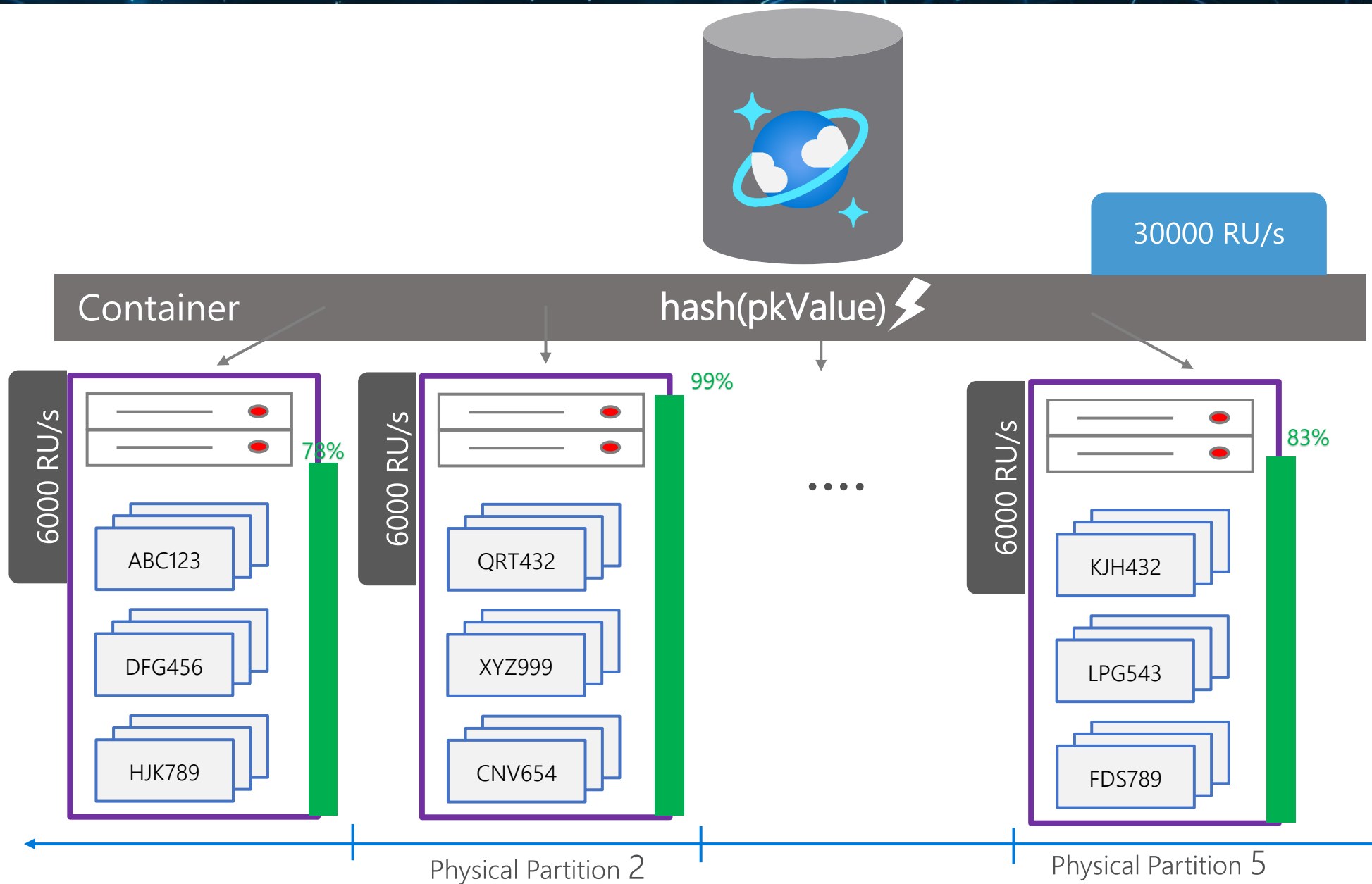
Container



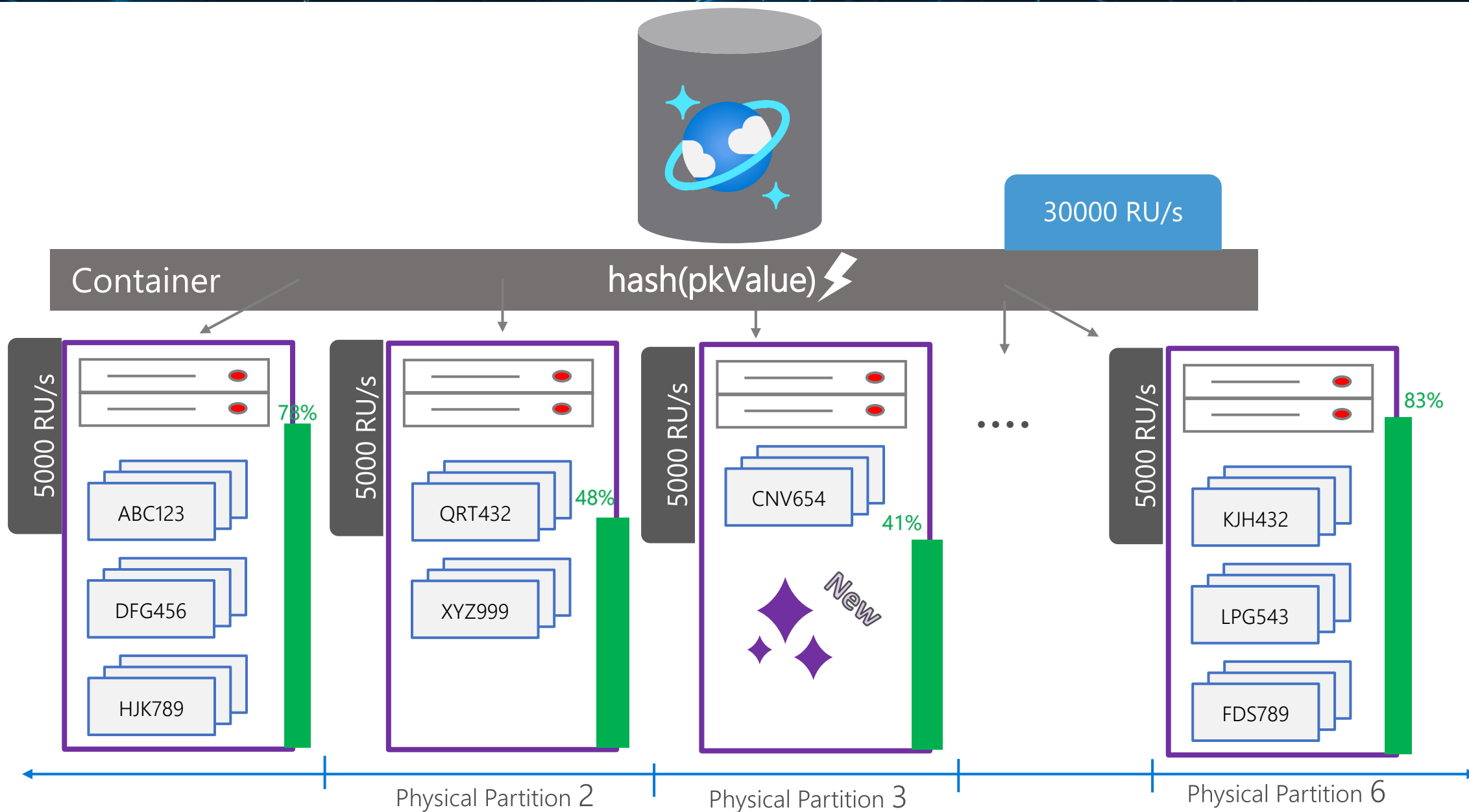
RU is distributed on Partitions



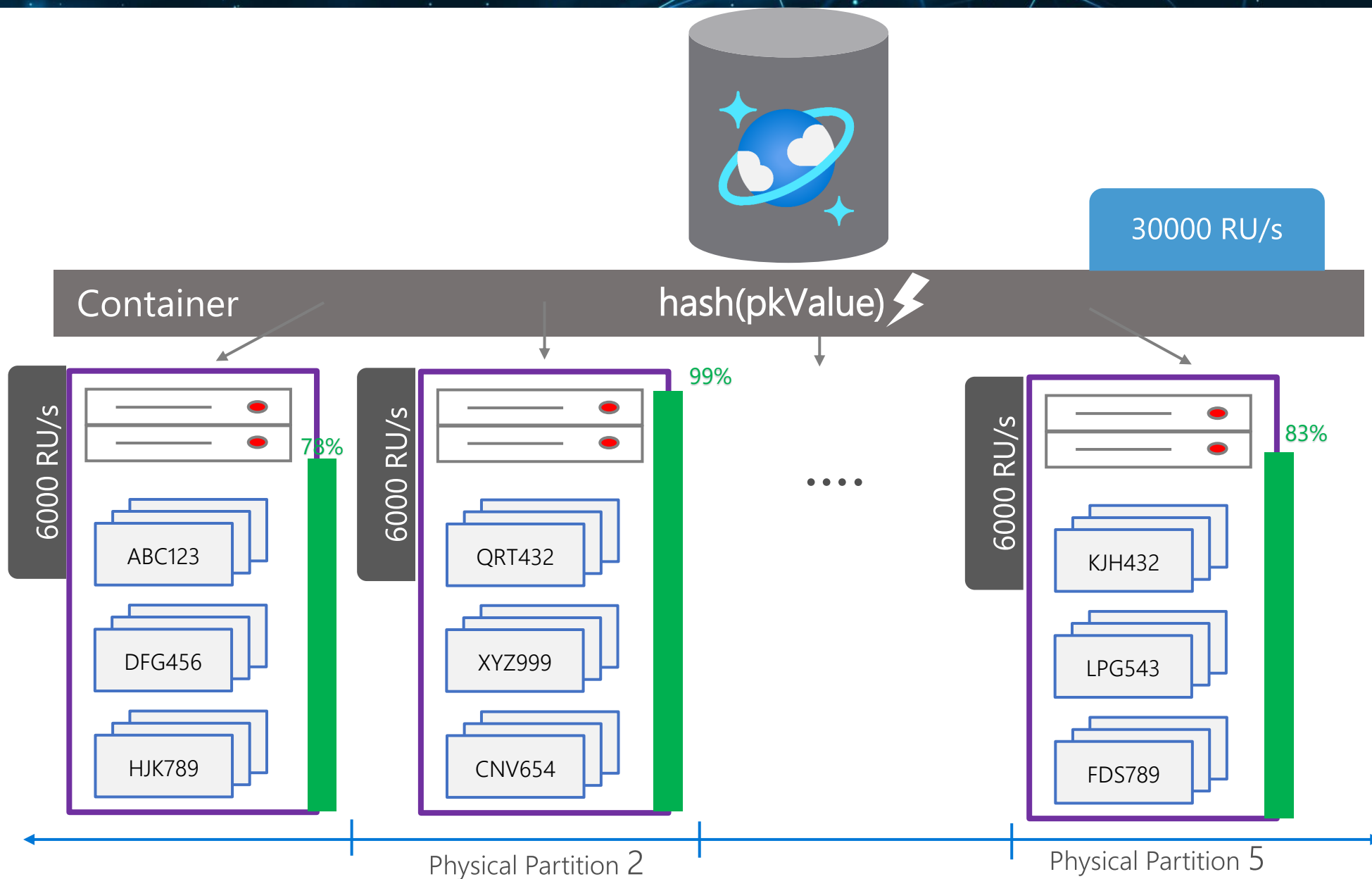
Auto Scale-out with storage growth



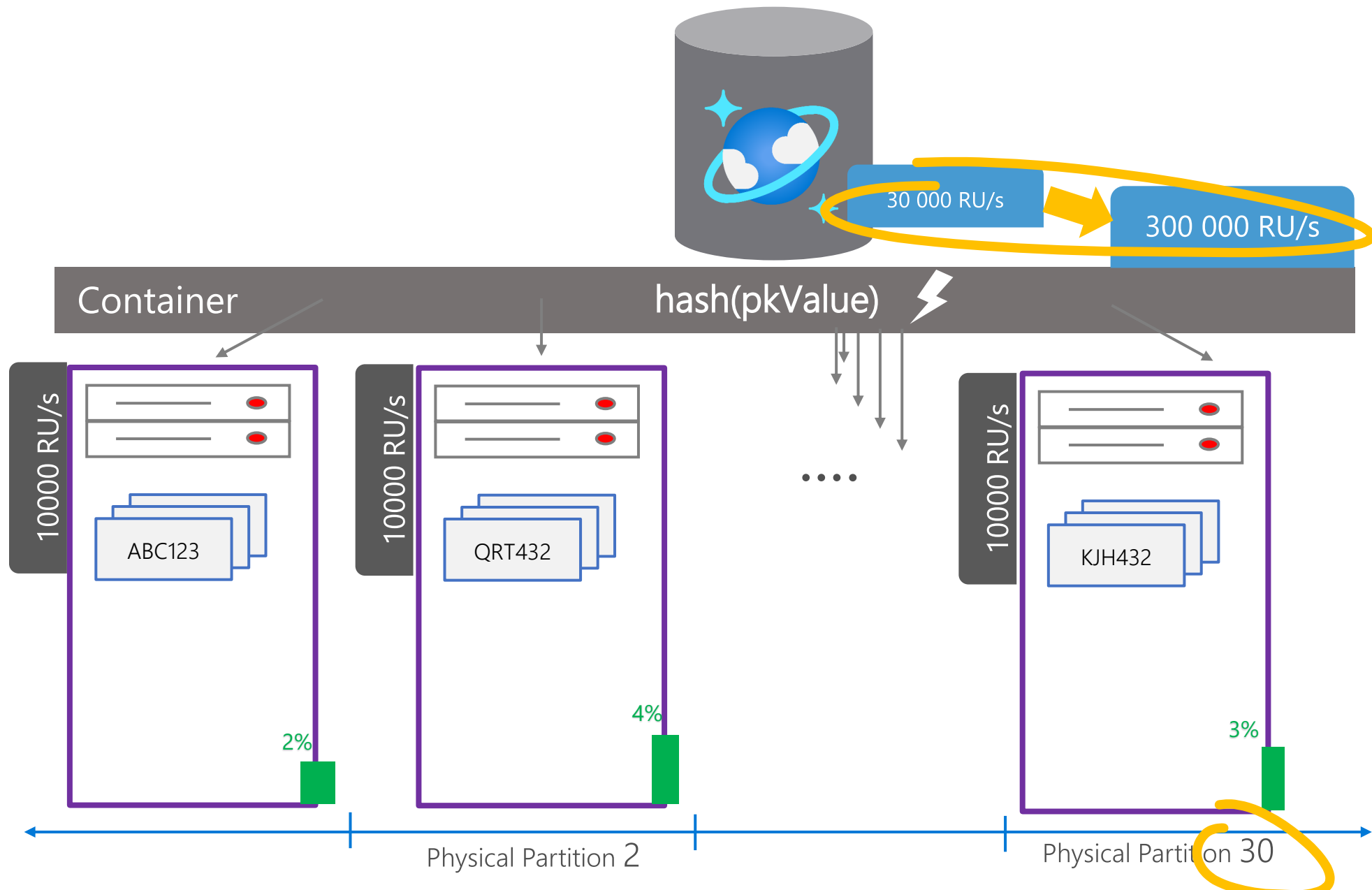
Auto Scale-out with storage growth



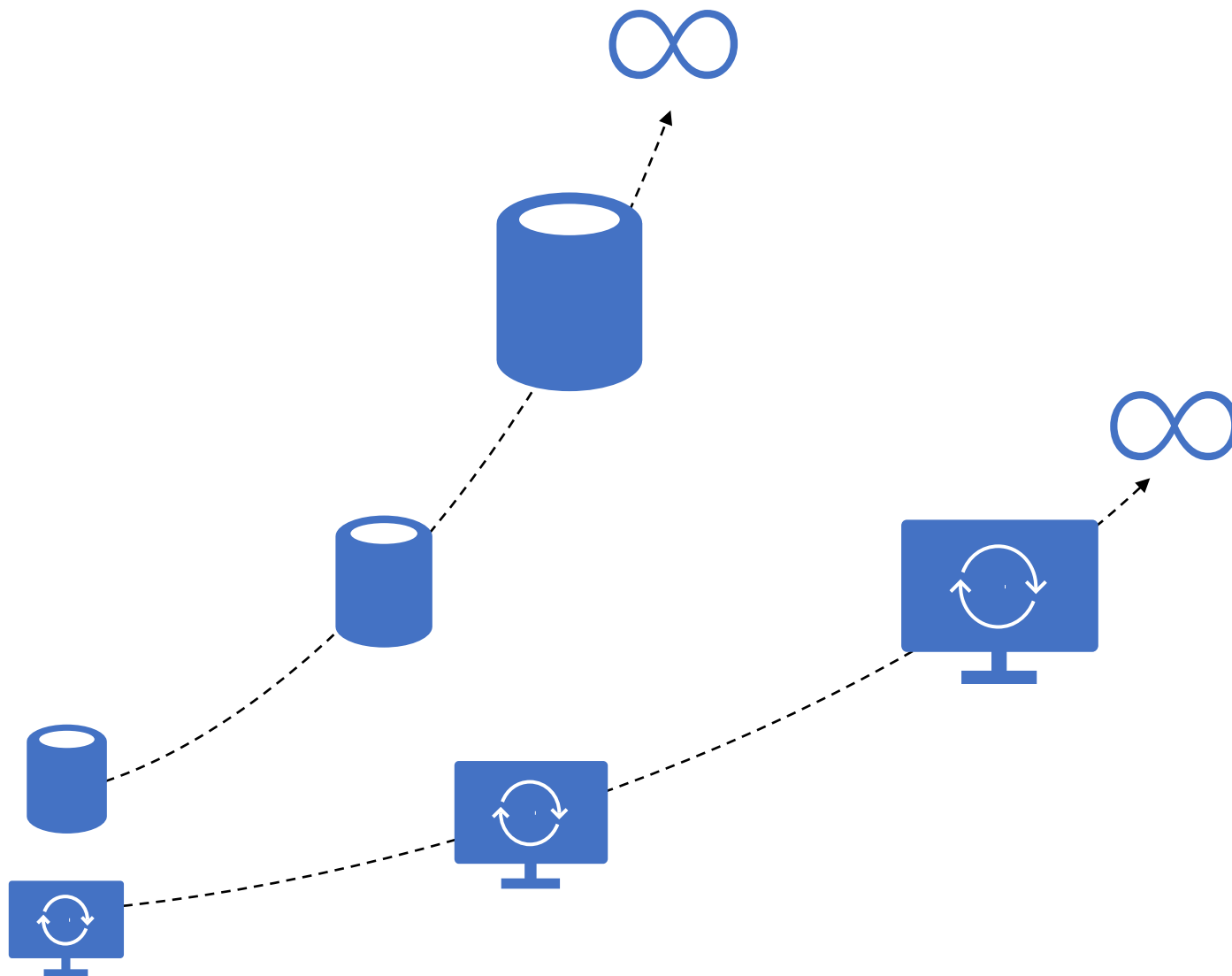
Auto Scale-out with Throughput Assign



Auto Scale-out with Throughput Assign



Take the advantage of Automatic Distribute and Scale



Scale & Setting... x

▼ Scale

Throughput (400 - 10,000 RU/s)

400

Estimated spend (RMB): ¥0.20 hourly / ¥4.90 daily (1 region, 400RU/s, ¥0.00051/RU)

Storage capacity

Unlimited

▼ Settings

Time to Live

Off On (no default) On

			Start Time	End Time	Duration
1	400	10,000	-	-	<10s
2	10,000	20,000	20:51:20	20:55:02	0:03:42
3	20,000	40,000	20:55:02	20:57:45	0:02:43
4	40,000	80,000	20:57:45	21:02:20	0:04:35
5	80,000	160,000	21:02:41	21:05:35	0:02:54
6	160,000	320,000	21:05:54	21:08:26	0:02:32
7	320,000	640,000	21:08:50	21:11:25	0:02:35
8	640,000	1,000,000	21:11:51	21:15:10	0:03:19
					0:22:20



Global Distribute (DR/HA/Multi Master)

Dashboard > ossdatademo > openflight

 **openflight** | Replicate data globally

Azure Cosmos DB account

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Cost Management

Quick start

Notifications

Data Explorer

Settings

Features

Replicate data globally

Default consistency

Backup & Restore

Networking

Keys

Save

Discard

Manual failover

Service-Managed Failover

⚠ When you add a region to your account, you will be billed for the additional RU/s and storage copied to the region. Click here to learn more. →

Click on a location to add or remove regions from your Azure Cosmos DB account.

* Each region is billable based on the throughput and storage for the account. [Learn more](#)



Configure regions

Multi-region writes ⓘ

Disable

Enable

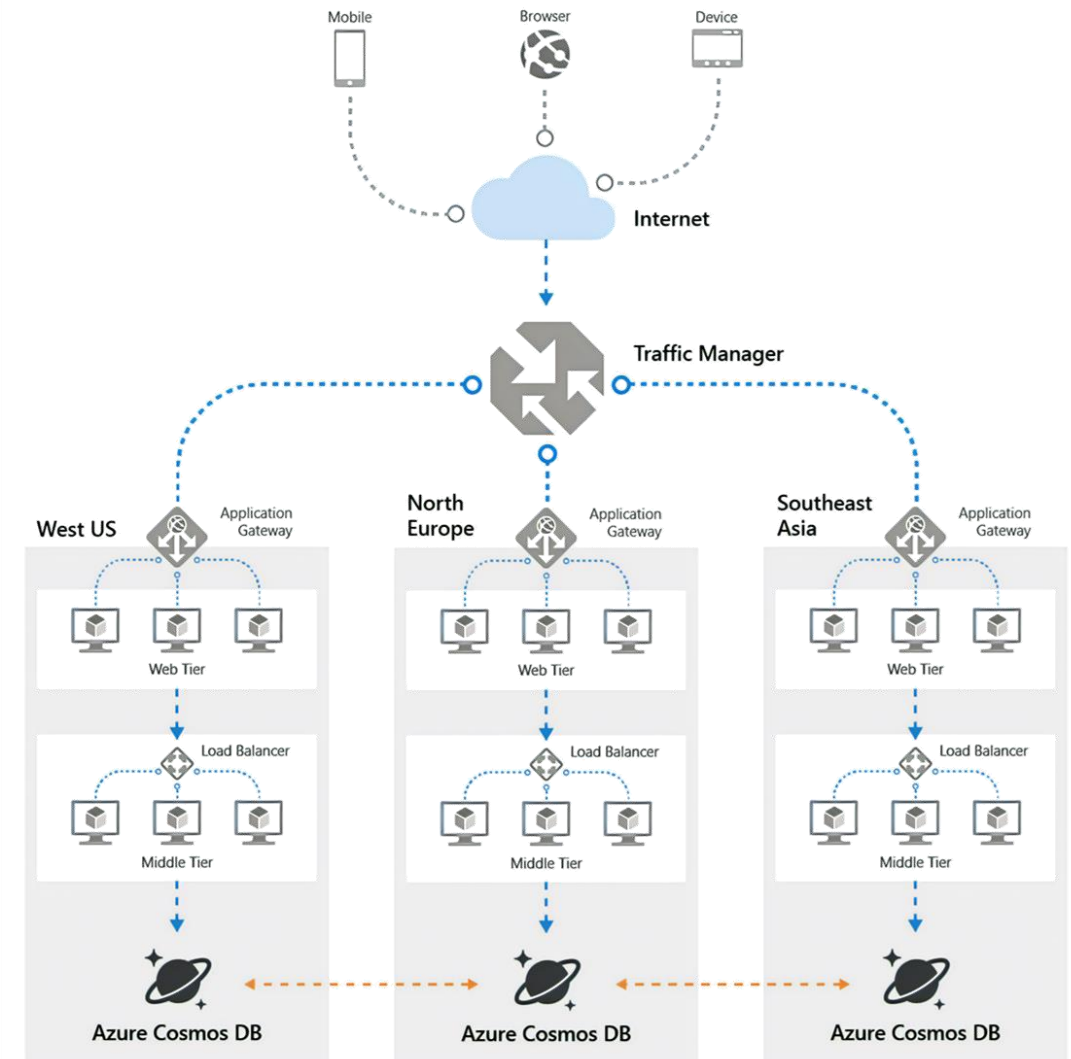
Configure the regions for reads, writes and availability zone (supported in selected regions and can only be configured when a new region is added). [+ Add region](#)

Regions	Reads Enabled	Writes Enabled	Availability Zone	Action
East Asia	✓	✓		
Qatar Central	✓	✓	☐	



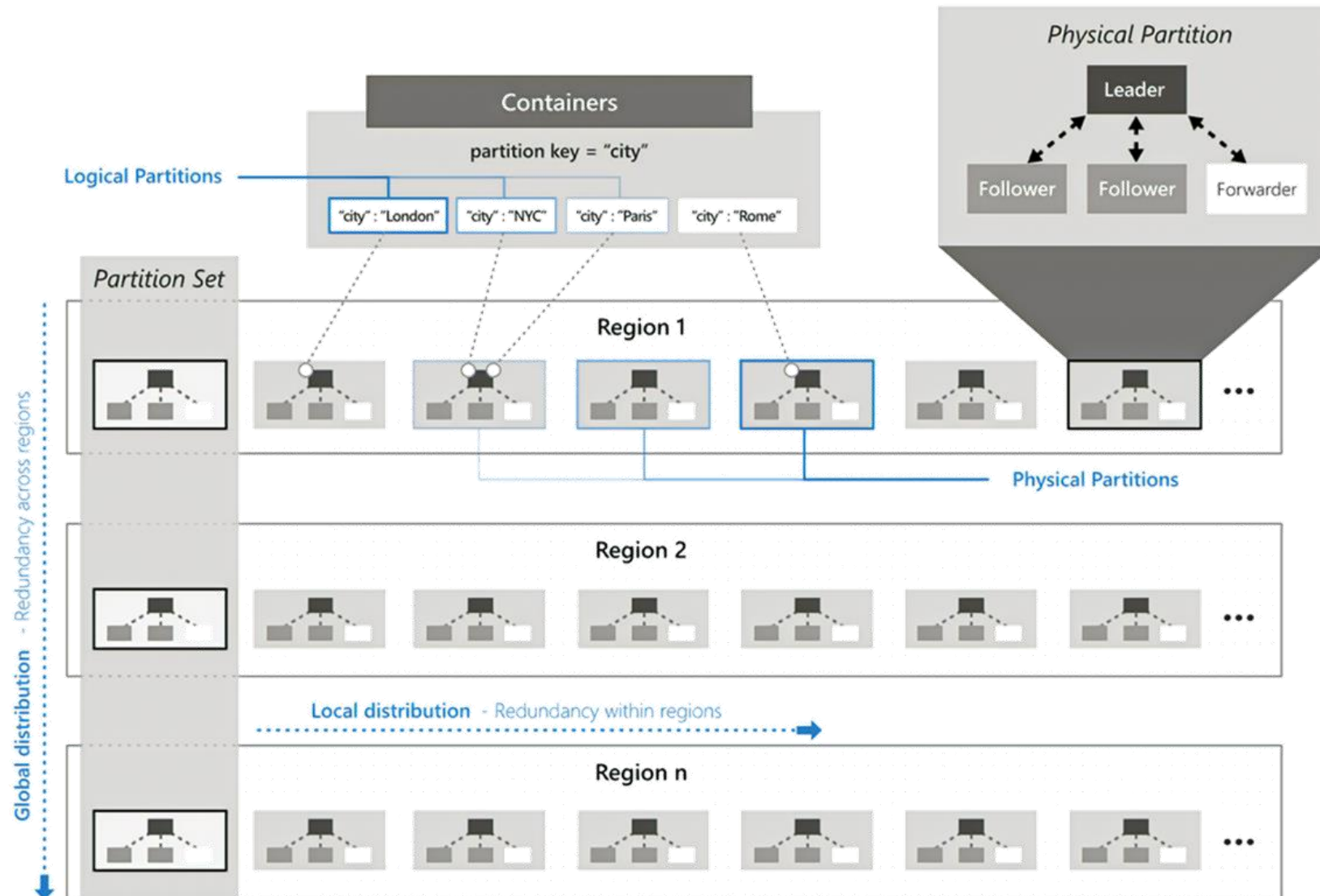
Turnkey Global Distribution

- High Availability
 - Automatic and Manual Failover
 - Multi-homing API removes need for app redeployment
- Low Latency (anywhere in the world)
 - Packets cannot move fast 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





System Topology (Partition = Replica Set)



Multi-Master

Perfect for Intelligent Cloud
and Intelligent Edge Applications

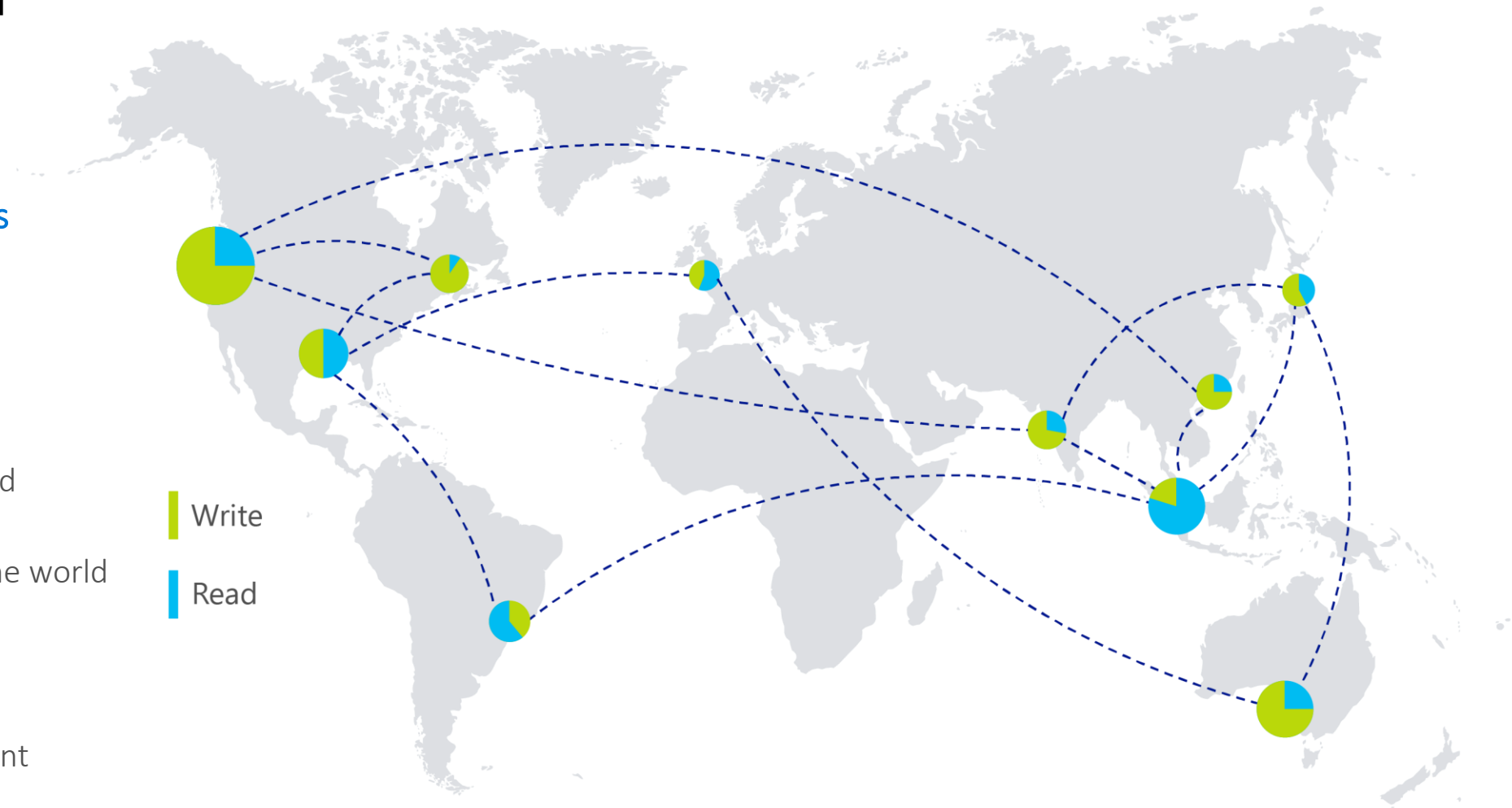
Write scalability around the world

Low latency writes around the world

99.999% High Availability around the world

Well-defined consistency models

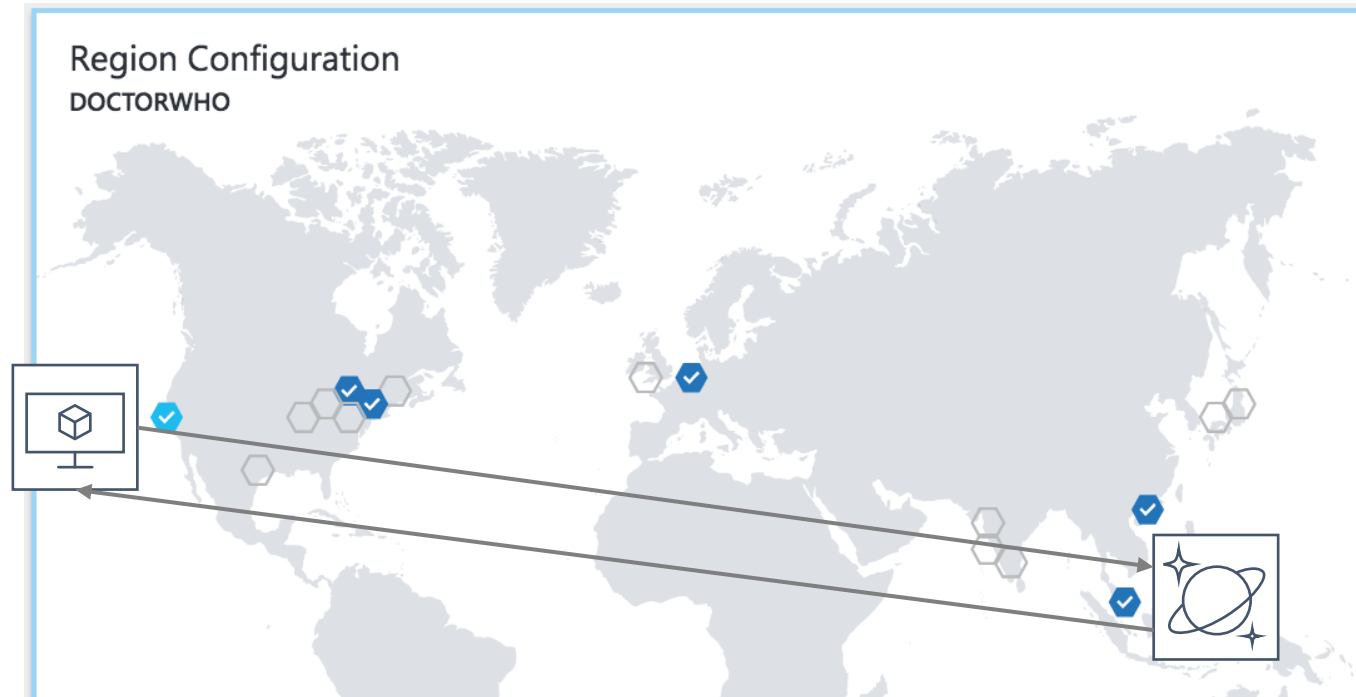
Comprehensive conflict management



Azure Cosmos DB Multi-Master



Replicating Data Globally

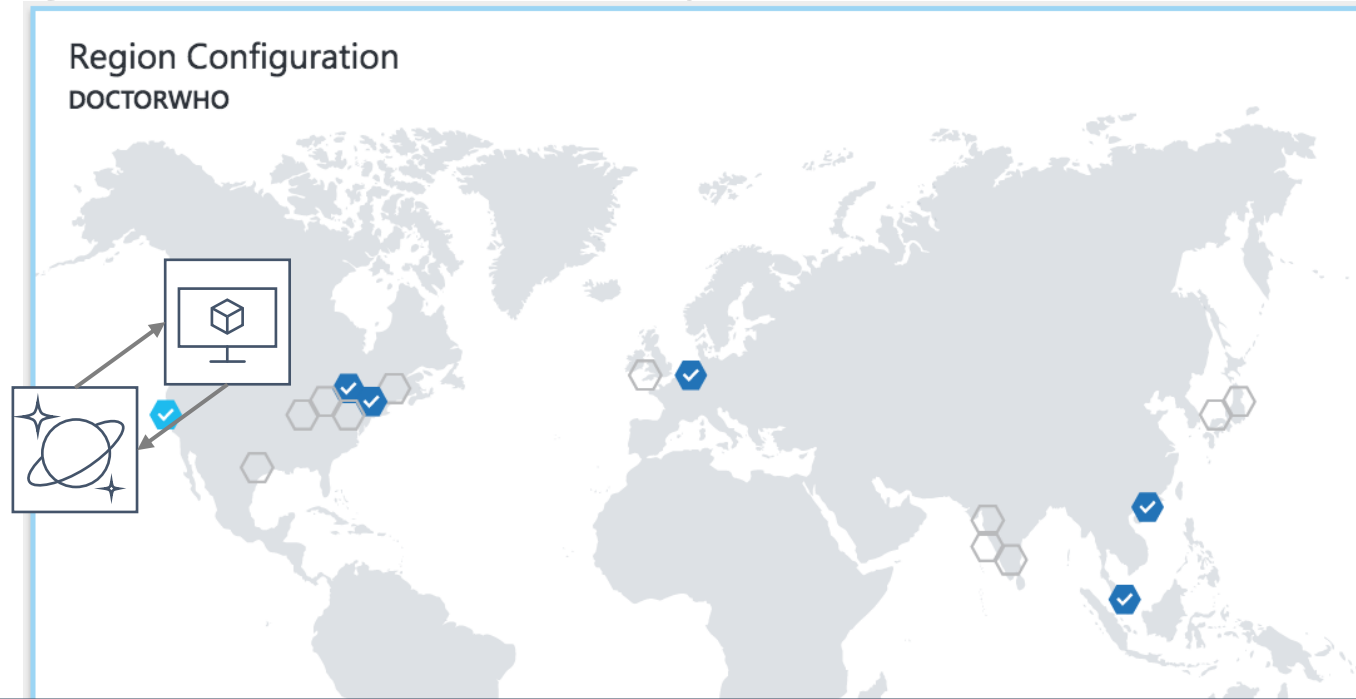


...rkspace/docdb/docdb — amypond@blink: ~/docdb — ssh amypond@104.42.108.173 -p 22

```
[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
```

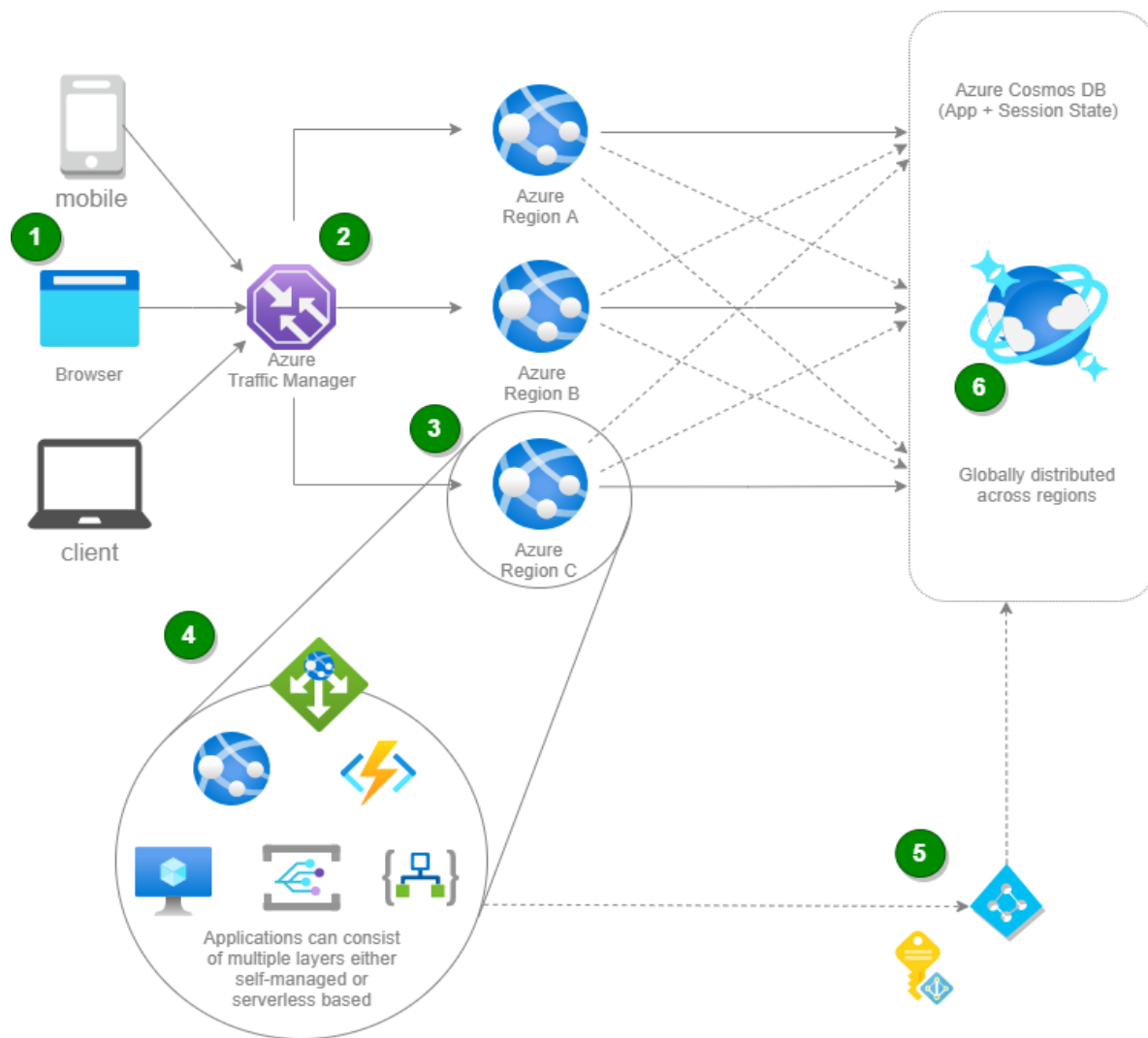


Replicating Data Globally



```
[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
```

Put your data where your users are



Dataflow

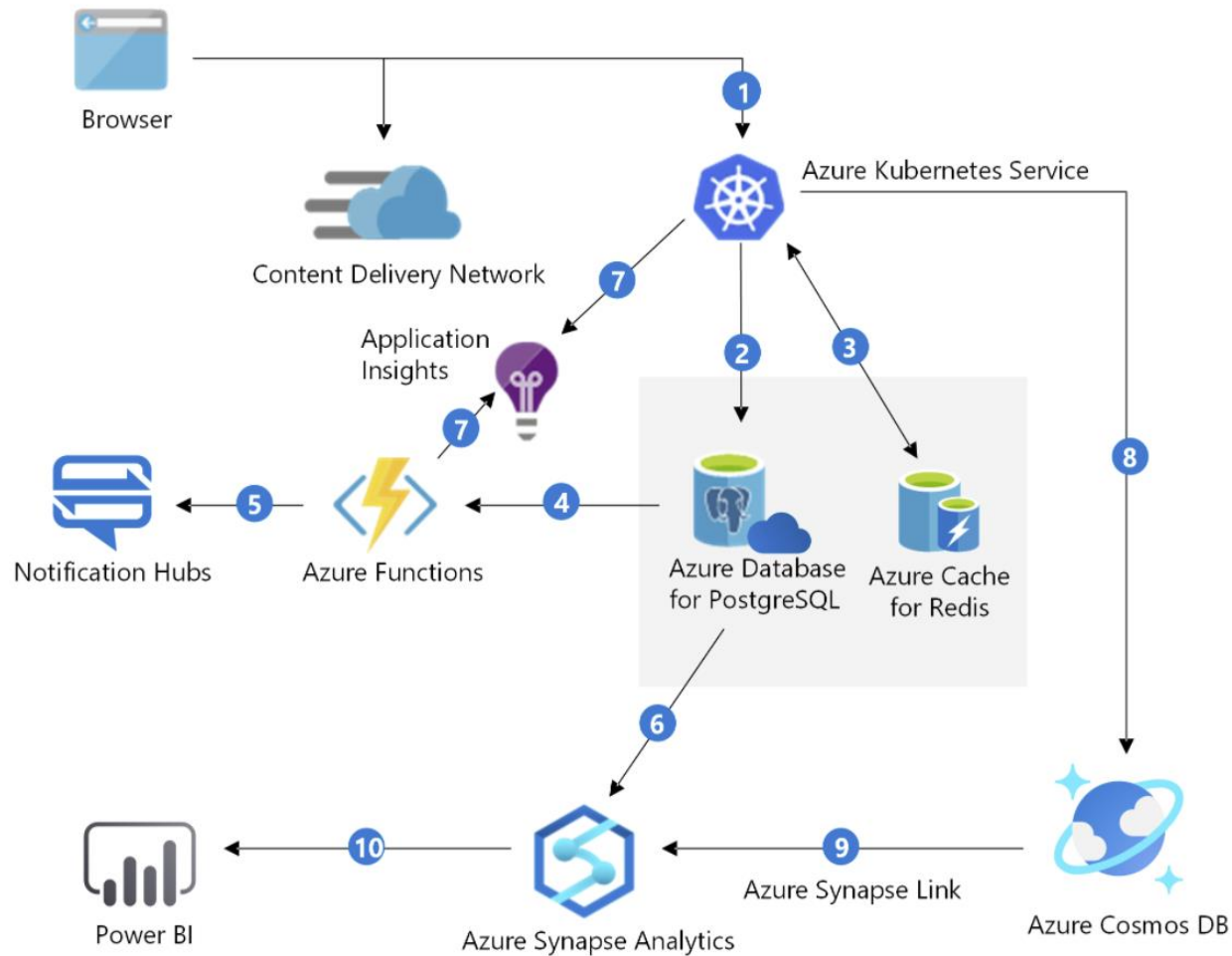
1. User accesses the application through the dedicated client.
2. Azure Traffic Manager will route the user's connection to the best location for accessing the application, based upon a single or nested routing profiles.
3. In the landed region where the application is hosted the application will handle the session and the connection towards the database.
4. This application can range from a simple static page up until a microservices-oriented application hosted in Kubernetes for instance.
5. The connection between the application landscape and the Azure Cosmos DB is handled through an Azure Active Directory User who can pick up the Azure Cosmos DB keys in Key Vault.
6. Your application is aware of the nearest region and can send requests to that region by using the Azure Cosmos DB multi-homing APIs. The nearest region is identified without any configuration changes. As you add and remove regions to and from your Azure Cosmos DB account, your application doesn't need to be redeployed or paused. The application continues to be highly available. Underneath the covers, Azure Cosmos DB will handle the global distribution and replication of the data based upon the number of defined regions. As an addition one should also benefit from the Automatic Failover option to fail over to the region with the highest failover priority with no user action should a region become unavailable. When automatic failover is enabled, region priority can be modified.

Agenda



- Begin with Azure Cosmos DB
- Deep Dive Inside Azure Cosmos DB
- Integrate with Azure Cosmos DB
 - *Analytical Storage*
 - *Change Feed*
 - *Azure Function*
 - *Azure Cognitive Search*
 - *Synapse link*
 - *Power BI*
 - *Databricks*
 - *Synapse*
 - *Azure Data Factory*

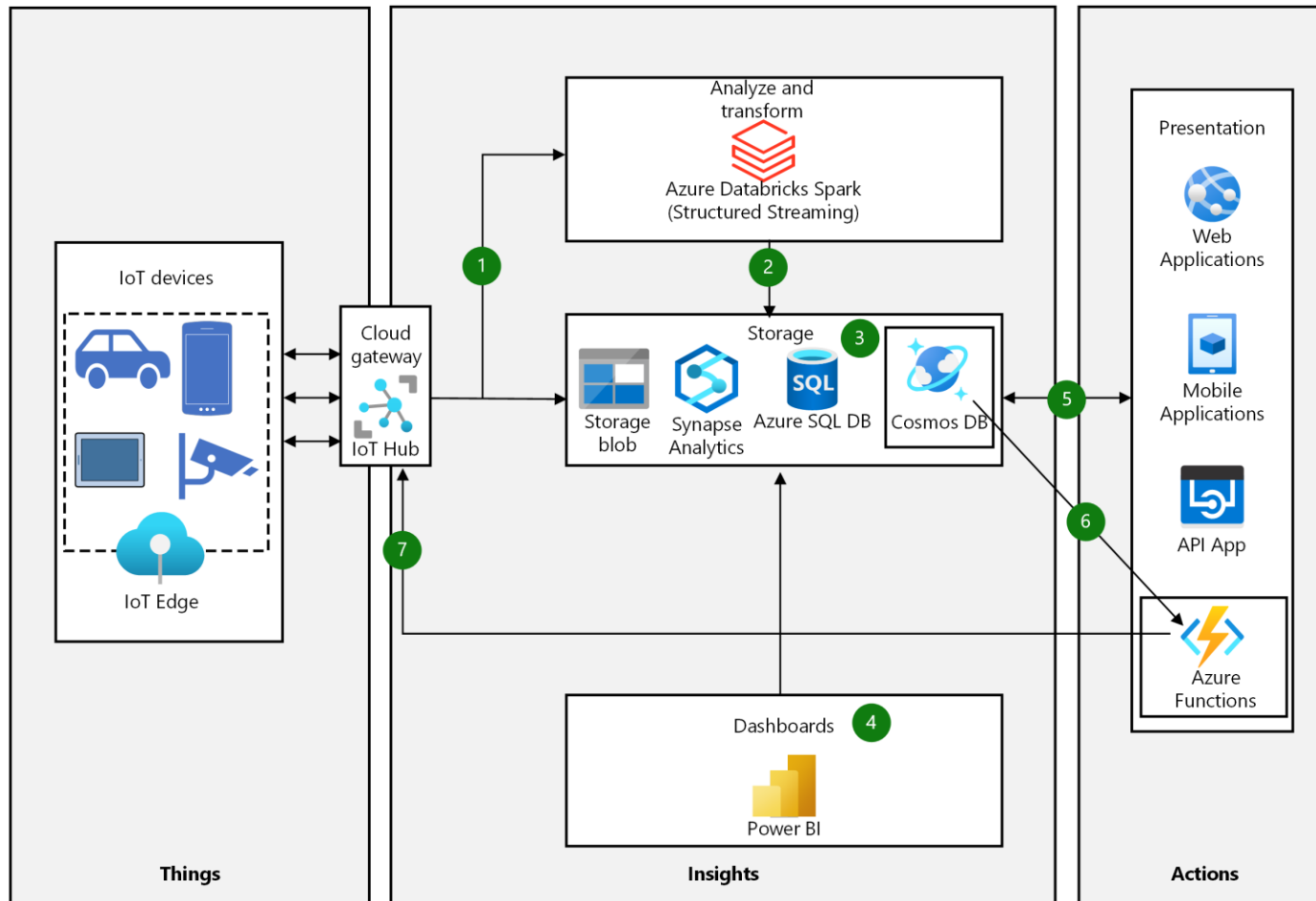
Building Cloud Native Apps on Azure



Data flow

1. Deploy and manage containerized applications easily with a continuous integration and delivery experience (CI/CD), and enterprise grade security and governance.
2. Focus on your app, not the database, with a fully managed database as a service for PostgreSQL. With built-in high availability and the rich feature set of Postgres, you can build design modern experiences free from legacy constraints.
3. Offload database demands by managing sessions state and asset caching with Azure Cache for Redis
4. Alert based on key events such as location or user activity using the serverless compute platform of Azure Functions.
5. Push timely notifications directly to your users on their preferred service or medium.
6. Derive deep insights by analyzing your data with Azure Synapse Analytics, with natively integrated Apache Spark for big data processing and machine learning.
7. Monitor your application's performance for degradation or anomalies, and auto-scale your application to changing performance requirements.
8. Track user interactions with you application at scale using Azure Cosmos DB. Easily scale to meet changing demand requirements with a fully managed NoSQL database.
9. Provide near real-time analytics and insight into user interaction by leveraging Azure Synapse Link for Azure Cosmos DB HTAP capabilities.
10. Finally, surface powerful visualizations of predictive, real-time, and historical transaction data using Power BI.

Azure Cosmos DB in IoT workloads

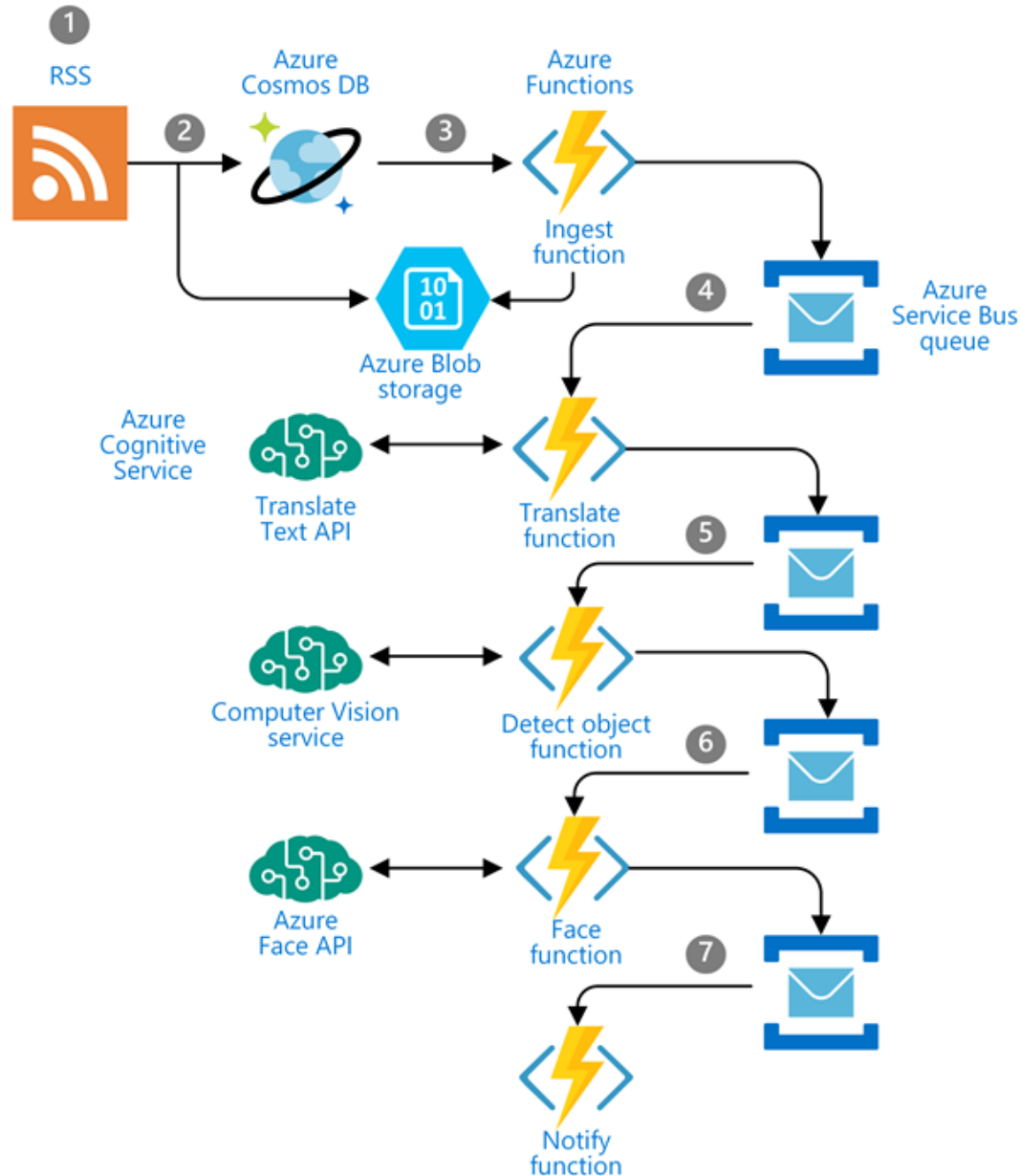


Data flow

- IoT sensors and Edge devices send events as message streams through Azure IoT Hub to the analyze and transform layer. IoT Hub can store data streams in partitions for a specified duration.
- Azure Databricks with Apache Spark Structured Streaming picks up messages from IoT Hub in real time, processes the data based on business logic, and sends the data to storage. Structured Streaming can provide real time analytics, such as calculating moving averages or minimum and maximum values over time periods.
- Azure Cosmos DB stores device messages as JSON documents in the *hot data store*. Azure Cosmos DB can validate against JSON schemas from different device vendors. The storage layer also consists of:
 - Azure Blob Storage. IoT Hub [message routing](#) saves raw device messages to Blob storage, providing an inexpensive, long-term *cold data store*.
 - Azure SQL Database, to store transactional and relational data, such as billing data and user roles.
 - Azure Synapse Analytics data warehouse, populated by [Azure Data Factory](#), which aggregates data from Azure Cosmos DB and Azure SQL DB.
- Microsoft Power BI analyzes the data warehouse.
- The presentation layer uses data from the storage layer to build web, mobile, and API apps.
- Whenever a new or updated device message arrives, Azure Cosmos DB change feed triggers an Azure Functions function.
- The function determines whether the message requires a device action, like a reboot. If so, the function connects to IoT Hub by using the IoT Hub Service API, and initiates the device action. The function can initiate the action by using device twins, cloud to device messages, or direct methods.



Analyze news feeds with near real-time analytics using image and natural language processing



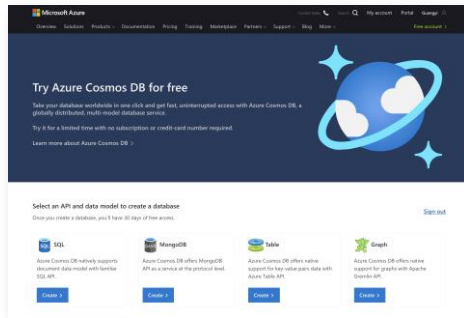
Data flow

1. An RSS news feed acts as the generator that obtains data from a document or article. For example, with an article, data typically includes a title, a summary of the original body of the news item, and sometimes images.
2. A generator or ingestion process inserts the article and any associated images into an Azure Cosmos DB Collection.
3. A notification triggers an ingest function in Azure Functions that stores the article text in Azure Cosmos DB and the article images (if any) in Azure Blob Storage. The article is then passed to the next queue.
4. A translate function is triggered by the queue event. It uses the Translate Text API of Azure Cognitive Services to detect the language, translate if necessary, and collect the sentiment, key phrases, and entities from the body and the title. Then it passes the article to the next queue.
5. A detect function is triggered from the queued article. It uses the Computer Vision service to detect objects, landmarks, and written words in the associated image, then passes the article to the next queue.
6. A face function is triggered from the queued article. It uses the Azure Face API service to detect faces for gender and age in the associated image, then passes the article to the next queue.
7. When all functions are complete, the notify function is triggered. It loads the processed records for the article and scans them for any results you want. If found, the content is flagged and a notification is sent to the system of your choice.

At each processing step, the function writes the results to Azure Cosmos DB. Ultimately, the data can be used as desired. For example, you can use it to enhance business processes, locate new customers, or identify customer satisfaction issues.

Ready to build modern apps using the Azure Cosmos DB ?

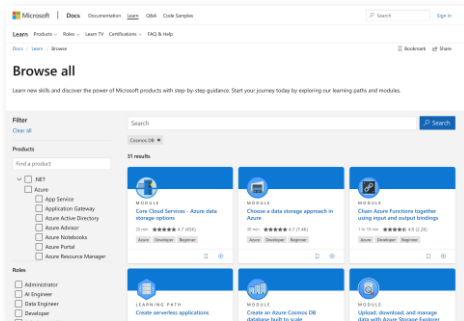
Get started today!



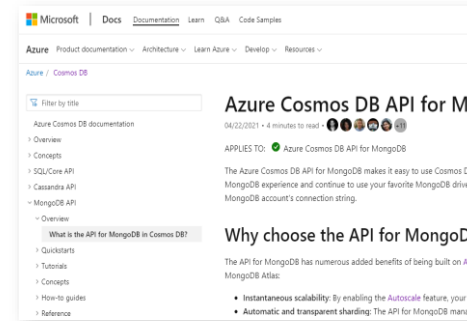
[Try Azure Cosmos DB](#) free or use free-tier to get started



[Learn](#) Azure Cosmos DB optimization & best practices



[Build your skills](#) with learning modules



[Read](#) Cosmos DB documentation



Thank You



Microsoft Azure Taiwan
Facebook 官方粉專



Azure Taiwan User Group
Facebook 交流社團

Eva Zhao (趙陽)

Microsoft, Asia Data & AI Senior Specialist