Microsoft Azure Solutions Business

Workshop Azure OSS DB Relational and NoSQL

9h30-12h

- Introduction aux bases OSSDB dans Azure
- OSSDB, Architecture et use cases
- Single server platform
- Flexible server platform
- Hyperscale platform (Citus)

- Cosmos DB Overview
- Cosmos DB Cassandra Instance
- Cosmos DB architecture
- Cosmos DB partitionning strategy
- Cosmos DB Analytical Store

Ali Bouhaddou
Cloud Solution Architect
Azure Data & Analytics

Frédéric Gisbert
Cloud Solution Architect
Azure Data & Analytics

Microsoft Azure Solutions Business

Introduction aux bases OSSDB dans Azure

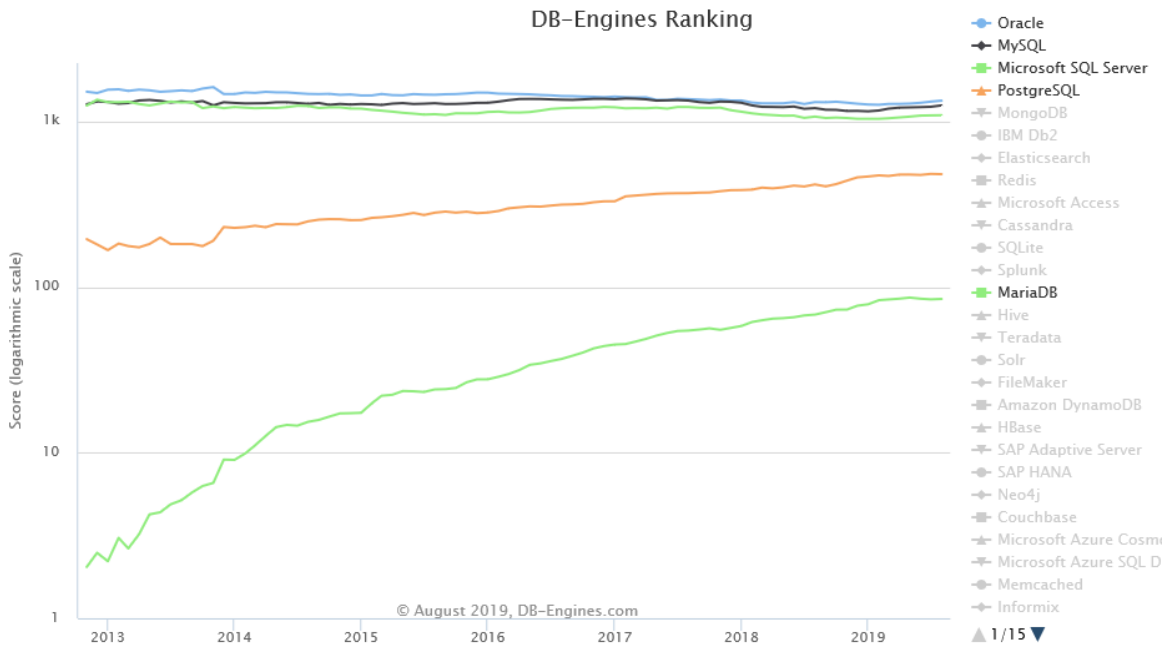
Growth in Popularity of Open Source Databases

80% of companies invest in public cloud technologies[1]

Over 70% of new in-house apps are now developed on OSS databases[2]

PostgreSQL and MariaDB are the fastest growing relational database management systems[3]

PostgreSQL named 2017 **and** 2018 "Database of the year" by DB-Engines.com[4]



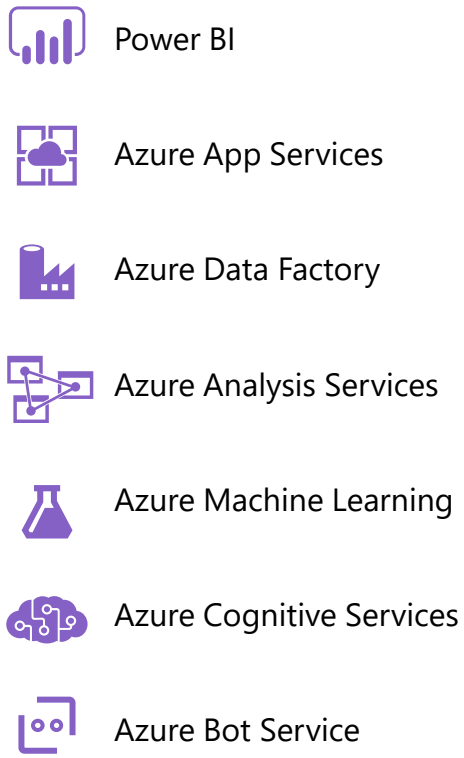
Gartner

Source: State of the Open-Source DBMS Market, 2018 (Feb 2018)

- By 2019, OSDBMS technology will account for **more than 10% of DBMS spending**, reflecting accelerating adoption by enterprise users.
- By 2022, **more than 70% of new in-house applications will be developed on an OSDBMS**, and **50% of existing commercial RDBMS instances will have been converted** or will be in process of converting.
- Through 2020, **relational technology will continue to be used for at least 85% of new applications and projects**.
- Through 2020, **open-source software in IT portfolios will increase at a 30% compound annual growth rate**.

[1]Cisco Global Cloud Index: Forecast and Methodology, editions 2013-2018 & 2015-2020
[2]The State of Open Source RDBMSs, 2015, Gartner (Donald Feinberg and Merv Adrian)
[3]DB-Engines. "DB-Engines Ranking". solid IT Gmbh, 2017. https://db-engines.com/en/ranking_trend
[4]DB-Engines. https://db-engines.com/en/blog_post/76 Kerschberg, Ben. "How Postgres and Open Source are Disrupting the Market for Database Management Systems". Forbes. 8 Mar 2016

A COMMON SERVICE FABRIC ACROSS THE AZURE RELATIONAL DATABASES PLATFORM



Azure SQL Data Warehouse | Azure SQL Database | Azure Database for PostgreSQL | Azure Database for MySQL | Azure Database for MariaDB

Database Services Platform

Intelligent
Advisors, tuning, monitoring

Flexible
On-demand scaling, resource governance

Trusted
HA/DR, backup/restore, security, audit, isolation

Azure compute

Azure storage

Global Azure in 54 regions

BETTER

1 Availability and performance

BETTER

Built-in high availability (99.99% SLA)

Scale in seconds to meet the most demanding spikes by elastically scaling performance

Rely on easy and cost-effective disaster recovery set up with a single click

Build on your terms with a platform that supports your various scale/performance needs

2 Security and compliance

BETTER

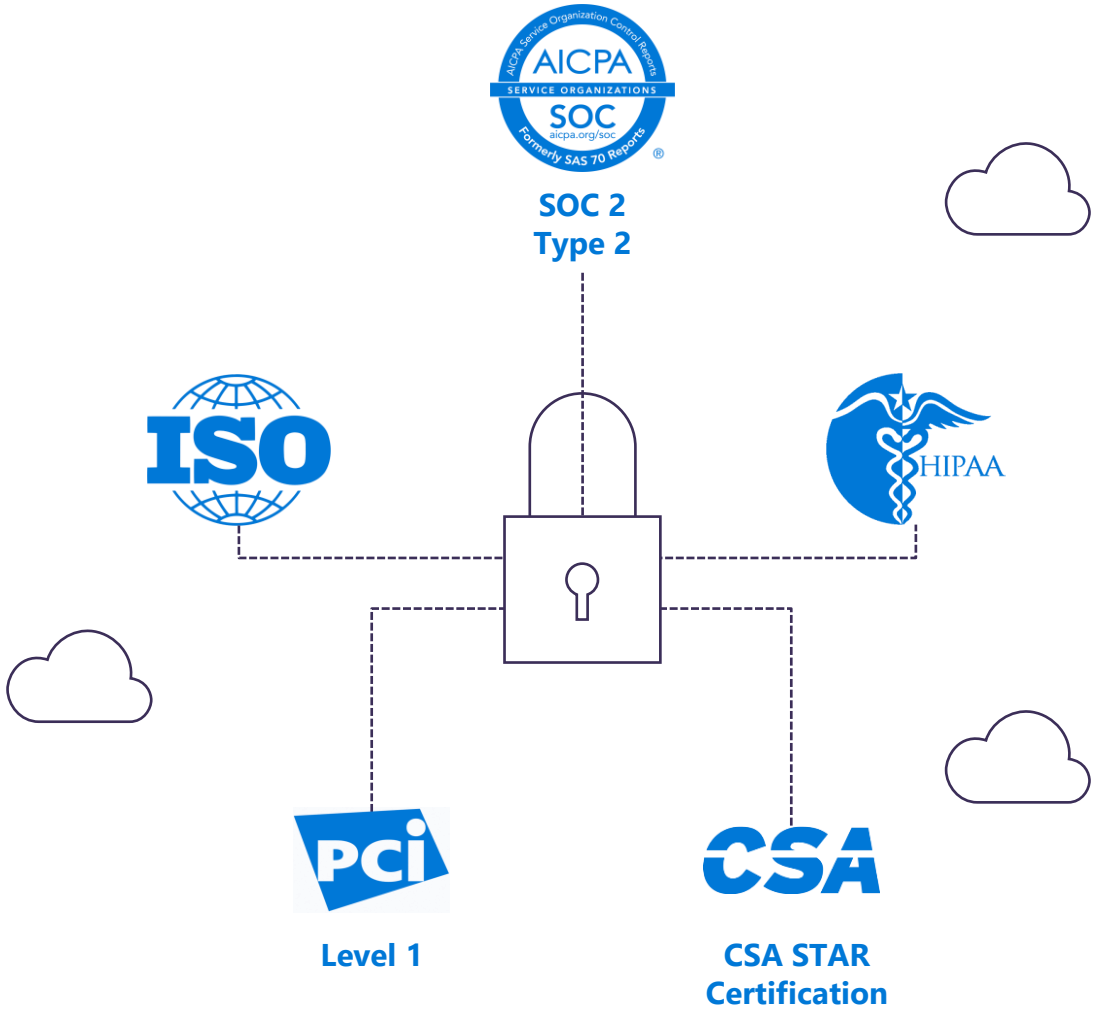
Security built-in with native and AAD integration

Control access with secure SSL, server firewall rules, and VNET

Built-in encryption for data and backups in-motion and at-rest

Protect your data with up-to-date security and compliance features using the Azure IP Advantage

Leading compliance offerings (SOC, ISO, CSA STAR, PCI DSS, HIPAA, etc.)



3 Manageability

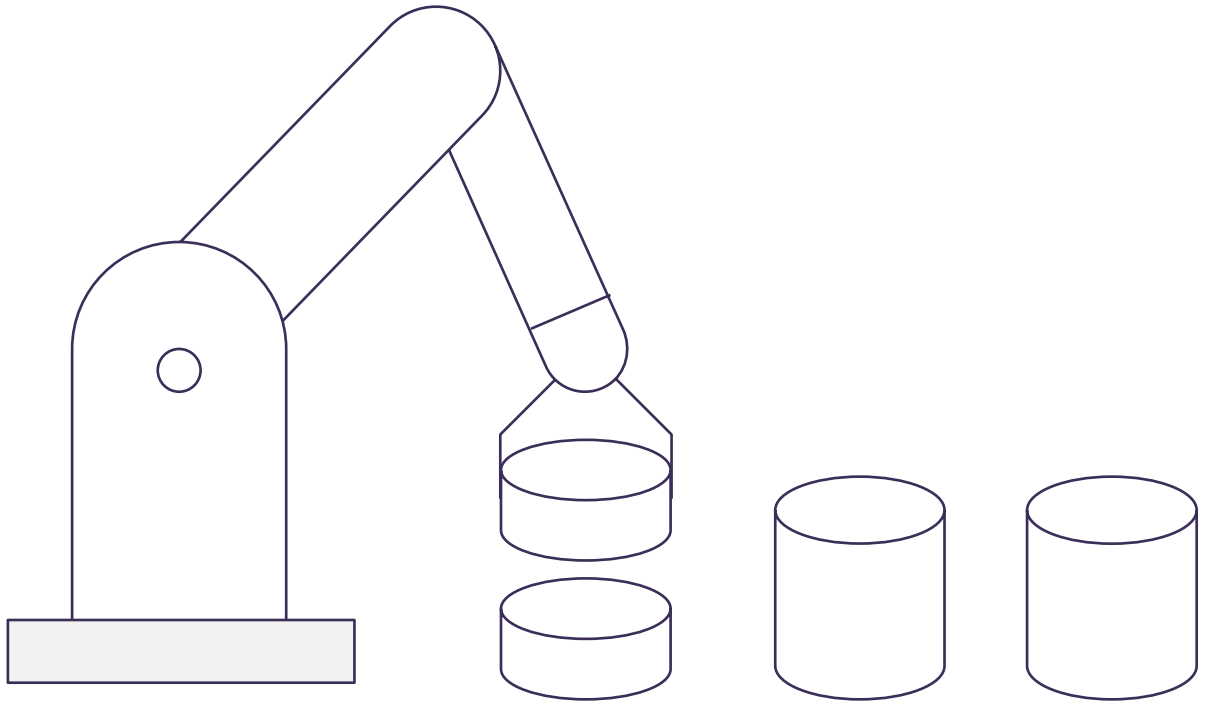
BETTER

Built-in backup and restore

Advanced monitoring and alerting

Intelligent performance tuning

AI-based security protection



4 Azure integration

BETTER

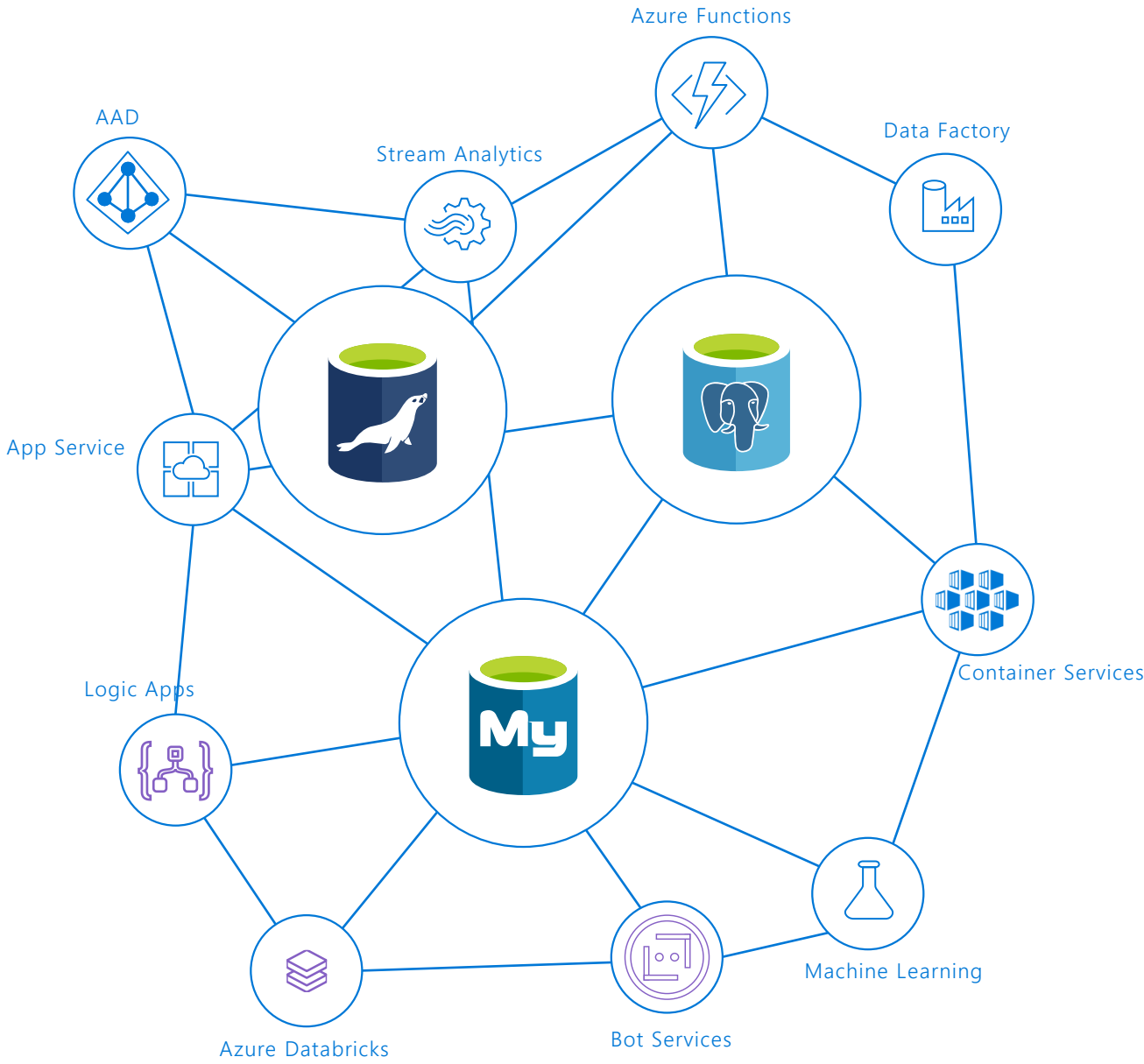
Dev Ops

Manageability and performance tuning

Data visualization

Advanced analytics and AI

Database migrations



Azure Database for PostgreSQL deployment options

Single Server

Fully-managed, single-node PostgreSQL database service with built-in HA

Example use cases

- Transactional and operational analytics workloads
- Apps requiring JSON, geospatial support, or full-text search
- Cloud-native apps built with modern frameworks

Hyperscale

Worry-free PostgreSQL in the cloud with an architecture built to scale out

Example use cases

- Scaling PostgreSQL multi-tenant, SaaS applications
- Real-time operational analytics
- Building high throughput transactional apps

Flexible Server NEW

Maximum control for your database with a simplified developer experience

Example use cases

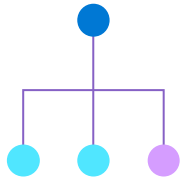
- Support for a variety of workloads with a new simplified architecture
- High-performance apps utilizing zone co-location for low latency
- Apps built on the latest Postgres version

The benefits of Azure Database for PostgreSQL

Build or migrate your workloads with confidence and optimized for value

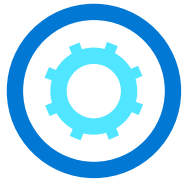
Ultimate control and flexibility for databases

Enjoy maximum control and flexibility with **Custom Maintenance Windows** and additional **configuration parameters** for fine grained tuning. Optimize TCO with **burstable instances** and **stop/start** capabilities.



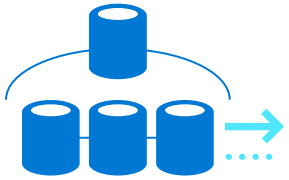
Maximize performance with a fully managed Azure service

Focus on your application innovation, not database management. Enjoy **zone redundant HA**, AI-powered **performance optimization**, and advanced security.



Innovate with open-source tools and extensions

Stay productive with full compatibility with **community PostgreSQL** and support for your favorite PostgreSQL **extensions**.



Build massively scalable PostgreSQL applications

Scale with ease to hundreds of nodes, with no app rewrites. Save time by running transactions and analytics in one database and avoid the costs of manual **sharding**.

Flexible Server

Single Server

Hyperscale

Build on a fully-managed database service with built-in high availability and security

Automatic updates and built-in HA

Azure ensures your data is available and automatically **updates** your database, freeing you to focus on your application

Scale in seconds

Scale your **compute** or **storage** resources **independently** to meet your application's needs

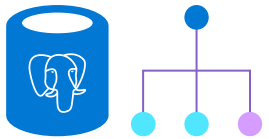
Built-in compliance and security

Automatically leverage enterprise grade security and compliance; proactively receive security alerts with **Advanced Threat Protection**

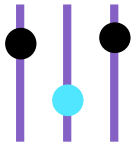
Azure IP Advantage

Rest assured that you have best-in-industry uncapped defense and indemnification coverage

Azure Database for PostgreSQL supports a large variety of PostgreSQL versions



Maximum control and flexibility to meet your workload needs with Flexible Server

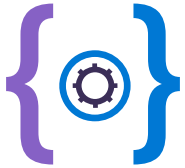


Maximum control for your databases

- Increased control for database management with Custom Maintenance Windows
- Multiple configuration parameters for fine grained database optimization and tuning

Zone redundant high availability

- Ensure data is always available with zone redundant HA
- Synchronous replication across availability zones for high resiliency
- Choose the availability zone for your database for improved connectivity



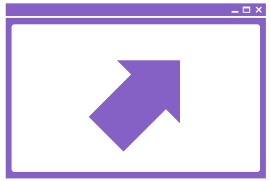
Simplified developer experience

- Guided developer experience increases productivity and simplifies end-to-end deployment
- Full compatibility with community edition PostgreSQL and MySQL
- Pause/Resume capabilities for cost optimization

Primary “Net New” Sales Motions for Orcas in FY21

Lift/shift existing OSS Relational workloads
Move to Azure

Cross-Sell with Cloud Native Apps/DevOps
Develop/Deploy with Azure



Migrate Expensive Oracle and ISV Workloads
Save with Azure

Microsoft Azure Solutions Business

OSSDB, Architecture et use cases

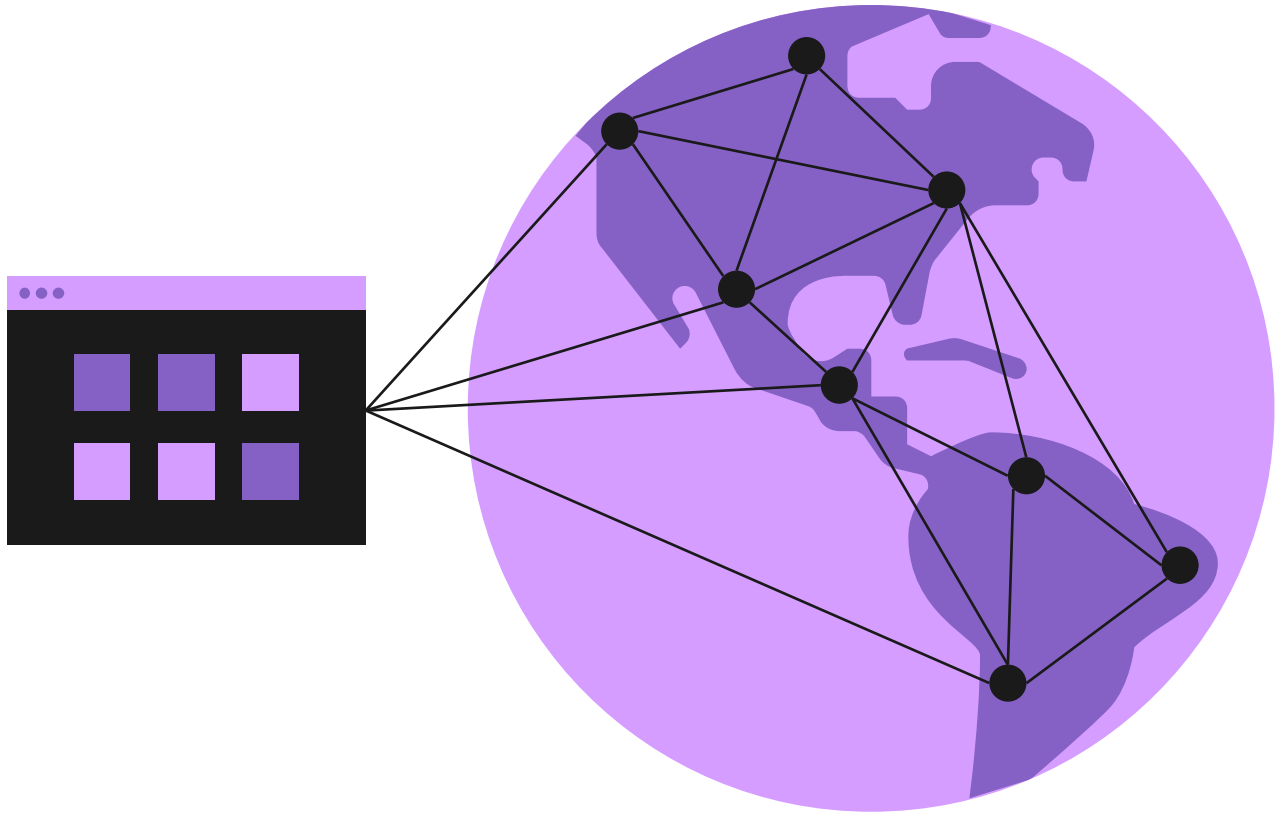
Azure Database Services for MySQL and Azure Database Services for MariaDB

Building next generation applications

From transforming products
to building innovative web apps,
developers use open source to
push the envelope of what's possible

Azure empowers developers with
open source-driven services for apps,
data, infrastructure and analytics

Azure Database for MySQL and
MariaDB illustrate Microsoft's commitment to
open source



Choose the database you prefer

MySQL is a leading open source relational database for LAMP stack apps

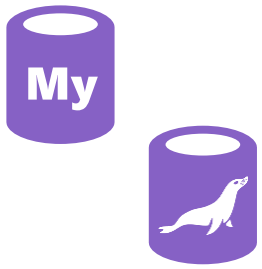
MariaDB is a community-developed fork of MySQL with strong focus on the user community

Enhance your database of choice by uitilizing it as a fully managed service in Azure

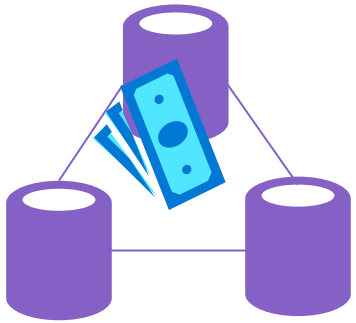
Azure Database Services for MySQL and MariaDB

Fully Managed Community Databases with full integration into Azure's ecosystem and services

Fully managed OSS community databases
with no vendor lock-in

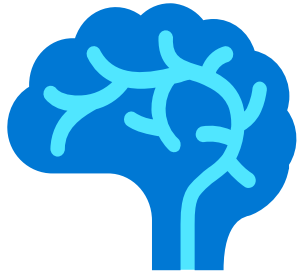


Best total cost of ownership
with built in HA



Secure and compliant
with Advanced Threat Protection and Azure IP Advantage

Built-in intelligence
optimizes performance and security



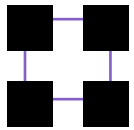
Integrates Azure services
with streamlined provisioning and management experience for common OSS frameworks and languages

Easy Migration → ← Enterprise-ready → ← Developer productivity →

Key use cases for Azure Database for MySQL and MariaDB

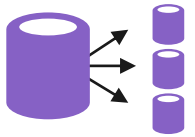
Web and mobile apps

Get to market quickly with LAMP stack



Microservices

Quickly deploy and scale microservices with Azure Kubernetes Service



Read-heavy workloads

Scale out read intensive workloads to ensure high read performance

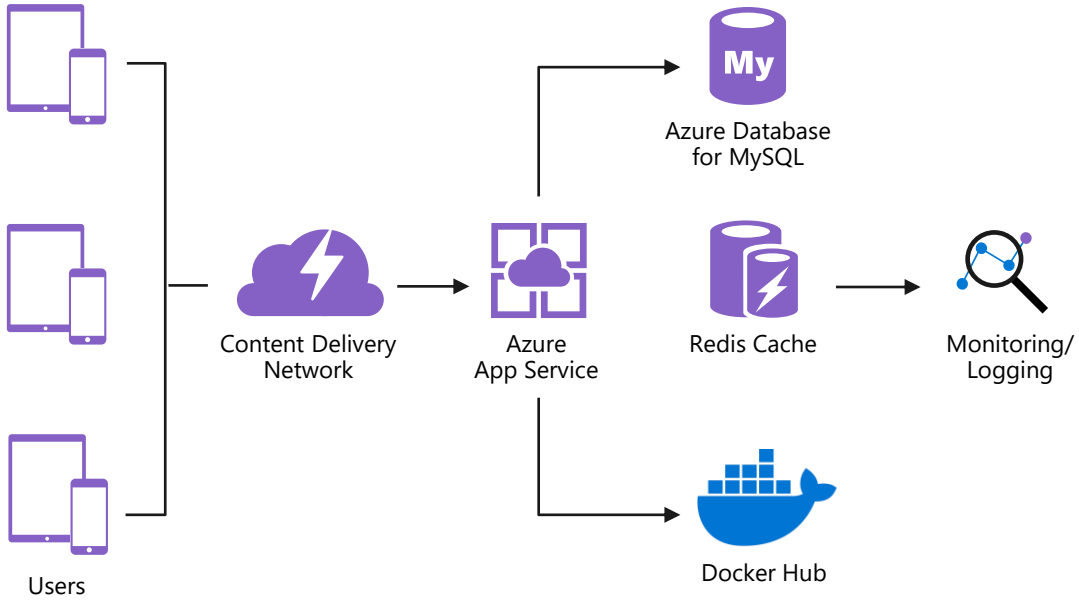
Migration

Reduce database administration by easily migrating to Azure

Get to market quickly with the LAMP open source stack

- Accelerate time to market by leaving database management to Azure Database for MySQL
- Run an Azure Database for MySQL server side-by-side Azure App Service to simplify development
- Integrate effortlessly with popular content management systems like WordPress, Drupal, and Joomla
- Choose from many languages like PHP, Java, Node.js, and more
- **Ideal for web apps, digital marketing, and gaming**

Scalable open source architecture using Azure App Service environment



THE NOBEL PRIZE

Azure Database for MySQL provides a fully managed service that stores vast content as part of the Nobel Prize's WordPress site

Optimize app development with microservice architectures

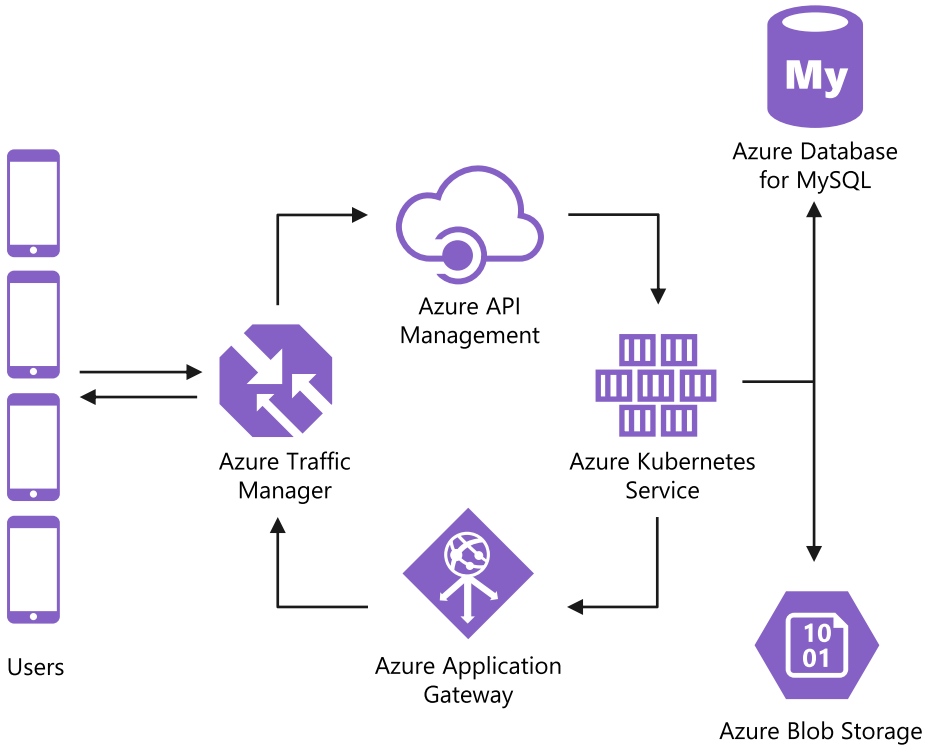
Easily integrate with AKS and other managed services for a quicker time to market

Quickly provision databases with microservices architectures and flexibly scale compute and storage

Easily integrate with Azure Kubernetes Service and other managed services to improve deploying and monitoring microservices

Ideal for retail, e-commerce, manufacturing, and banking applications

Microservice architecture with Azure Database for MySQL and Azure Kubernetes Service



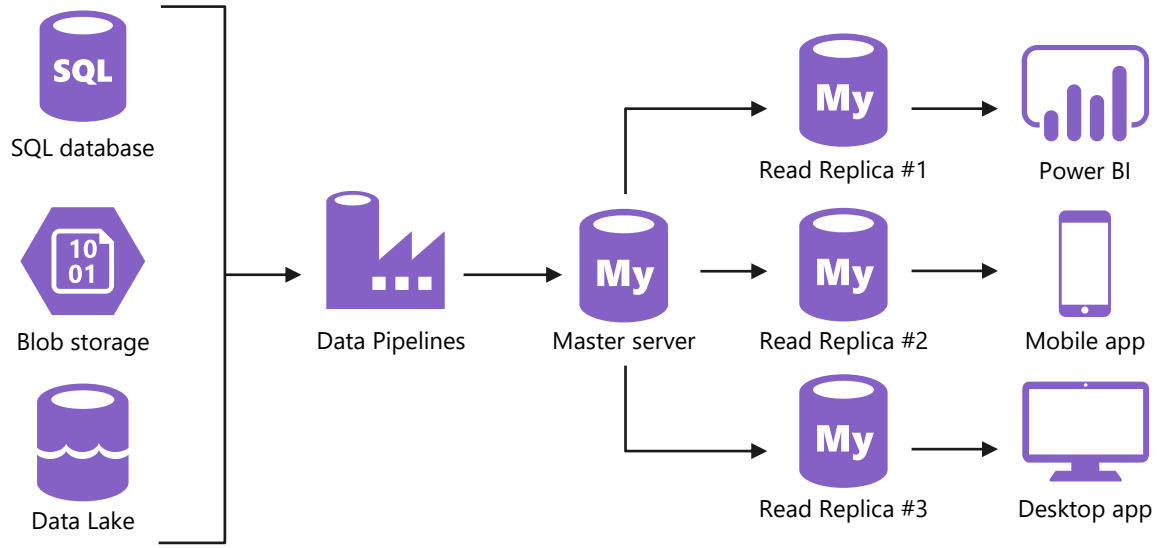
WhiteSource

WhiteSource provides greater value to its customers by delivering new services faster and dynamically scaling database resources up and down

Optimize performance for read-heavy workloads

- Increase performance by deploying read replicas, effectively isolating read and write workloads
- Bring applications closer to users around the world by deploying read replicas to other Azure regions
- **Ideal for read-heavy workloads including BI reporting, social media, web apps, and analytics**

Read-heavy workload architecture with read replicas

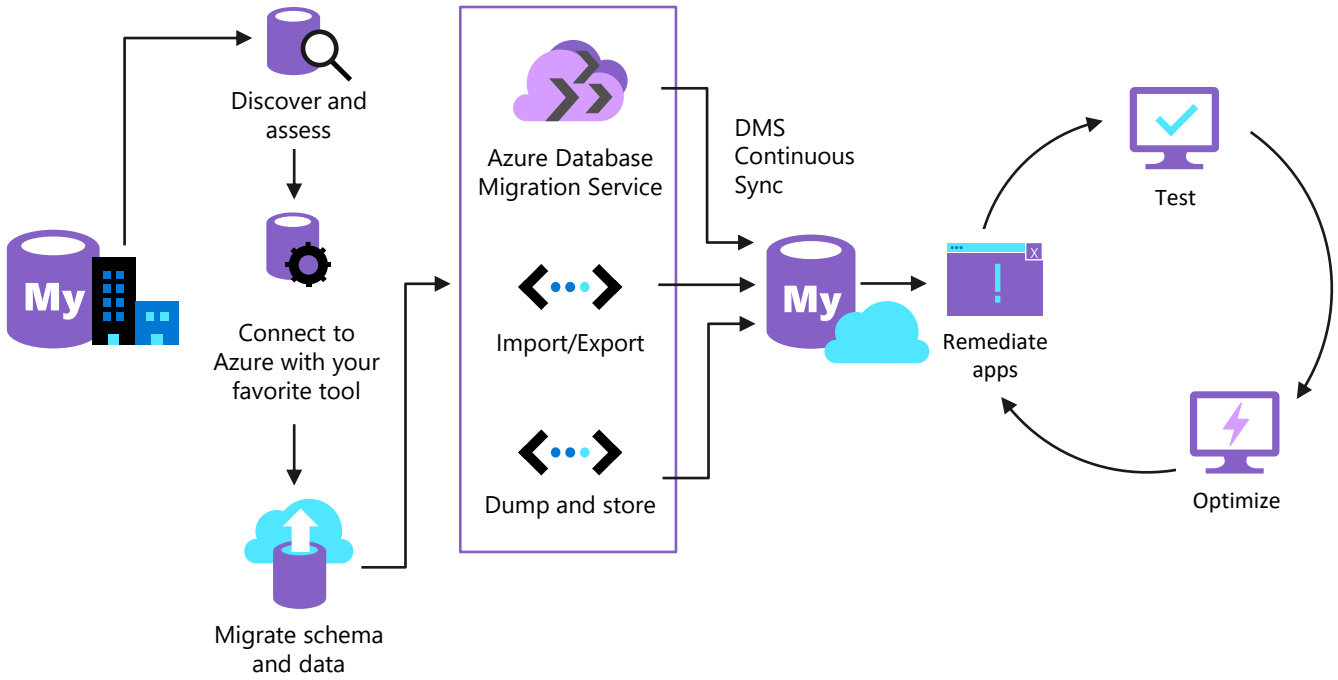


Fotoable deploys photo-sharing apps around the world with Azure Database for MySQL servers deployed across Azure regions

Simplify migration and reduce database administration

- Free your team from database management by migrating on-premises and IaaS workloads to Azure Database for MySQL or MariaDB
- Choose the migration approach that best fits the needs of your workloads
- Reduce downtime on critical databases with Azure Database Migration Service*
- Ideal for on-premises or IaaS MySQL or MariaDB databases

Options for migrating to Azure Database Services



GeekWire

GeekWire removed the pain of patching, scaling, and backing up by migrating its WordPress site to Azure Database for MySQL

* https://datamigration.microsoft.com/scenario/mysql-to-azuremysql?step=1
* https://datamigration.microsoft.com/scenario/postgresql-to-azurepostgresql?step=1
* https://datamigration.microsoft.com/scenario/mariadb-to-azuremariadb?step=1

Elevate MySQL and MariaDB with Azure

Fully managed community database

Built-in High Availability for lowest TCO

Intelligent performance and scale

Industry-leading security and compliance

Integration with the Azure ecosystem

Simplify management, improve performance, and innovate faster with Azure Database for MySQL and MariaDB

Microsoft Azure Solutions Business

OSSDB, Architecture et use cases

Azure Database Services for PostgreSQL

Why PostgreSQL?

- Rich selection of datatypes including text based, numeric, arrays, geospatial, IP, key/value, and JSON
- Powerful built-in functions to make working with data easier
- Multiple index types to allow for fine tuned performance
- Built-in JSON support including indexing of all keys/values within a JSON document for fast retrieval and querying
- Rich set of extensions to enable new functionality without relying on new Postgres versions/releases. This includes things like hypothetical indexing, geospatial support, time series partitioning

Get started with open source extensions

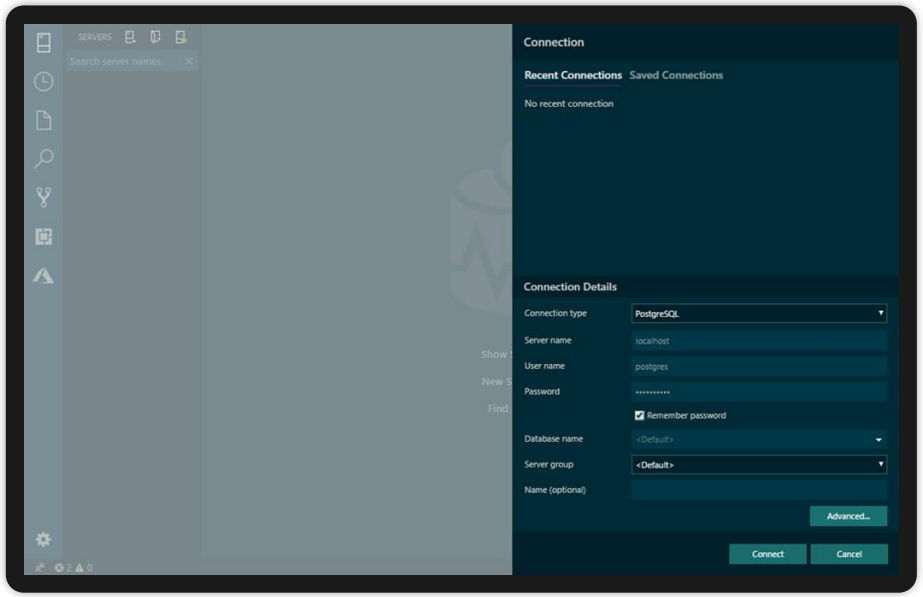
Microsoft is committed to contributing to, working with and supporting access to the Postgres community and its innovations

Microsoft contributions to the Postgres community:

- **Azure Data Studio PostgreSQL extension** enables you to connect to, query, and develop PostgreSQL using a modern data editor
- **Visual Studio Code extension** streamlines code editing with built-in git commands and optimization for quick code-build-debug cycles
- And more, including **pg_auto_failover, cstore_fdw, hll, pg_cron, topN, and Citus Community**

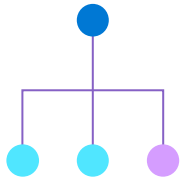
Azure Database for PostgreSQL also supports popular PostgreSQL extensions including **PLV8, TimescaleDB, PostGIS, Hstore, and many more**

Connecting to PostgreSQL on Azure Data Studio



The benefits of Azure Database for PostgreSQL

Build or migrate your workloads with confidence

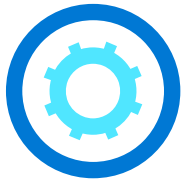


Fully managed and secure

Focus on your apps while Azure manages resource-intensive tasks, supports a large variety of Postgres versions and provides best-in industry indemnification coverage

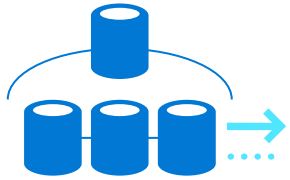
Intelligent performance optimization

Improve performance and reduce cost with customized recommendations



Flexible and open

Stay productive with your favorite Postgres extensions and leverage Microsoft's contributions to the Postgres community



High performance scale-out with Hyperscale (Citus)

Break free from the limits of single-node Postgres and scale out across hundreds of nodes

Single Server

Hyperscale (Citus) NEW

Primary Use-Cases
PostgreSQL with Hyperscale (Citus)

Data estate modernization

Single-server PostgreSQL

Data intensive OSS relational apps

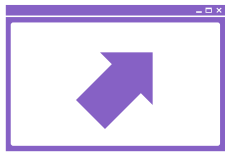
Hyperscale (Citus)

Cross-Sell with Cloud Native Apps/DevOps

Python/Django, Ruby/Rails, AKS

Lift/shift existing OSS Relational workloads

PaaS vs. IaaS, PostgreSQL on-premises

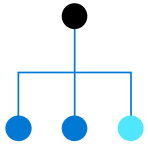


Migrate Expensive Oracle Workloads

TCO, Digital transformations

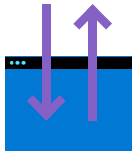
Real-time, operational analytics applications

Analytics on JSON data, Geospatial, Timeseries, In-Memory / HTAP workloads



Multi-tenant & SaaS applications

Sharding, B2B apps built by ISVs, New SaaS applications



Transactional/OLTP applications

Strong consistency, Relational semantics (foreign keys, joins), higher throughput

Single server use cases

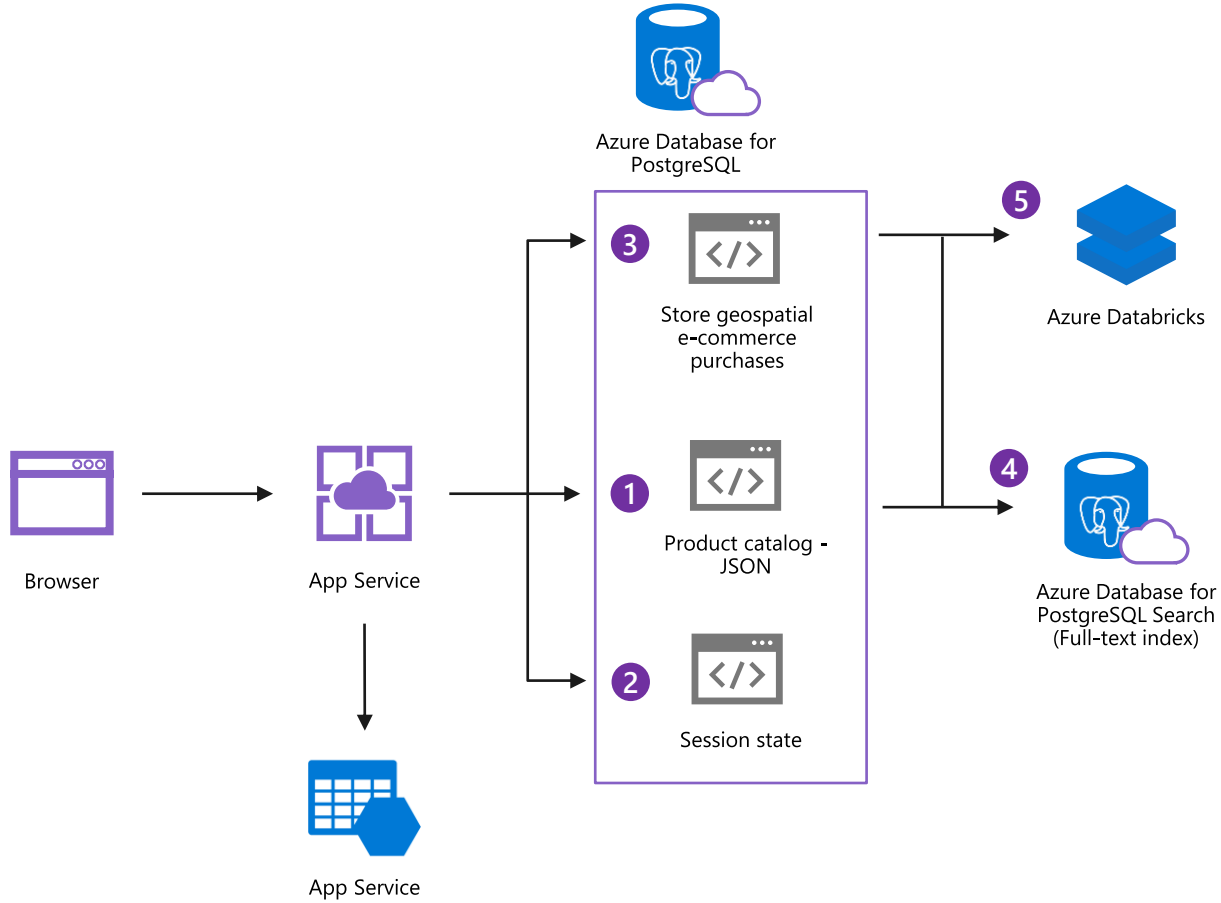
Apps requiring JSON, geospatial support, or full-text search

Cloud-native apps built with modern frameworks

Transactional and operational analytics workloads

Apps requiring JSON, geospatial, full-text search

1. Build your application with both defined schema and unstructured NoSQL type data by leveraging Postgres JSON support
2. Postgres built-in query planner allows for efficient caching which is great for managing user session state
3. Advanced geospatial support allows you to layer in location-aware capabilities into your application
4. Integrated full text search provides rich interactions with data
5. Use scalable machine learning/deep learning techniques, to derive deeper insights from this data



Cloud-native applications built with modern frameworks

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 rich feature set of Postgres you can build design modern experiences free from legacy constraints.

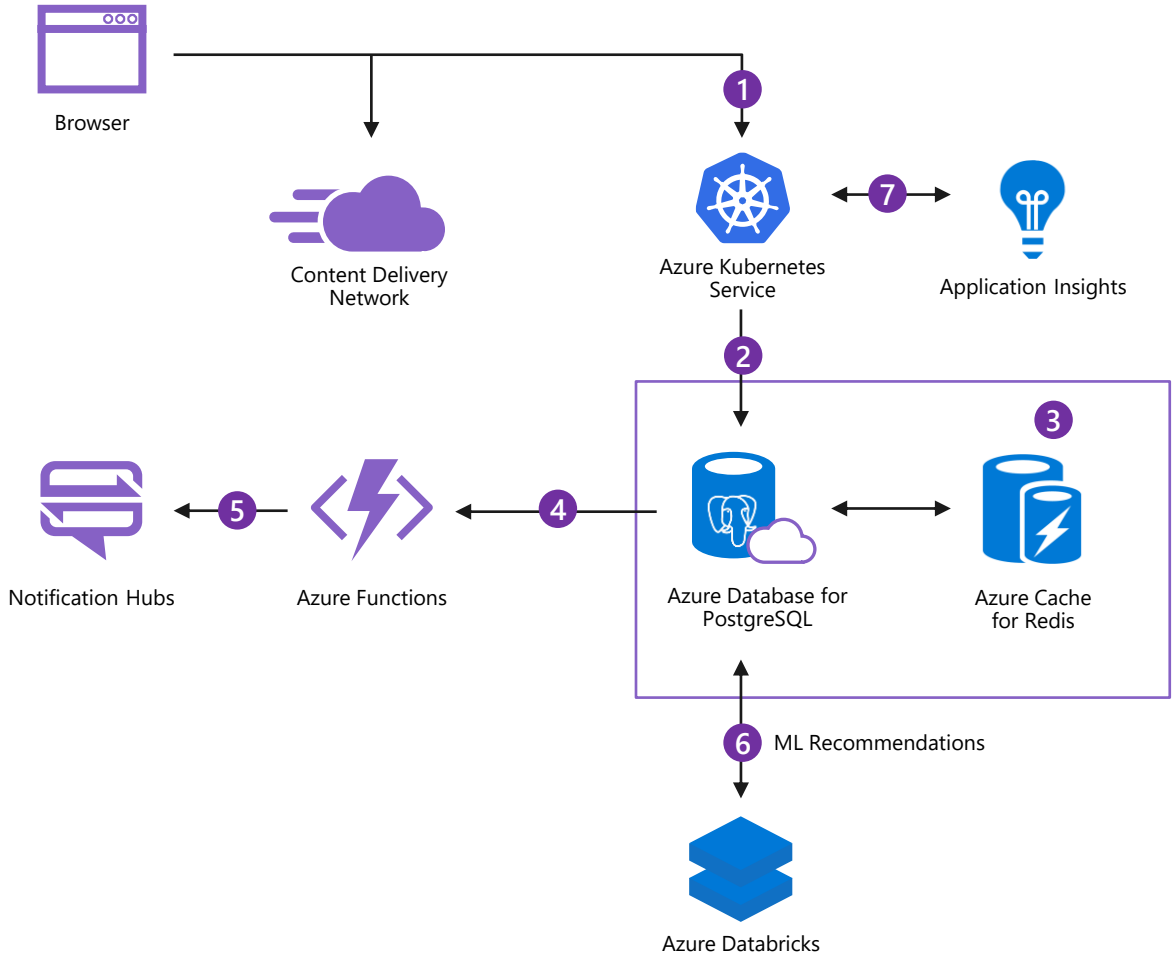
3. Offload database demands by managing sessions state and asset caching with Azure Cache for Redis

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.

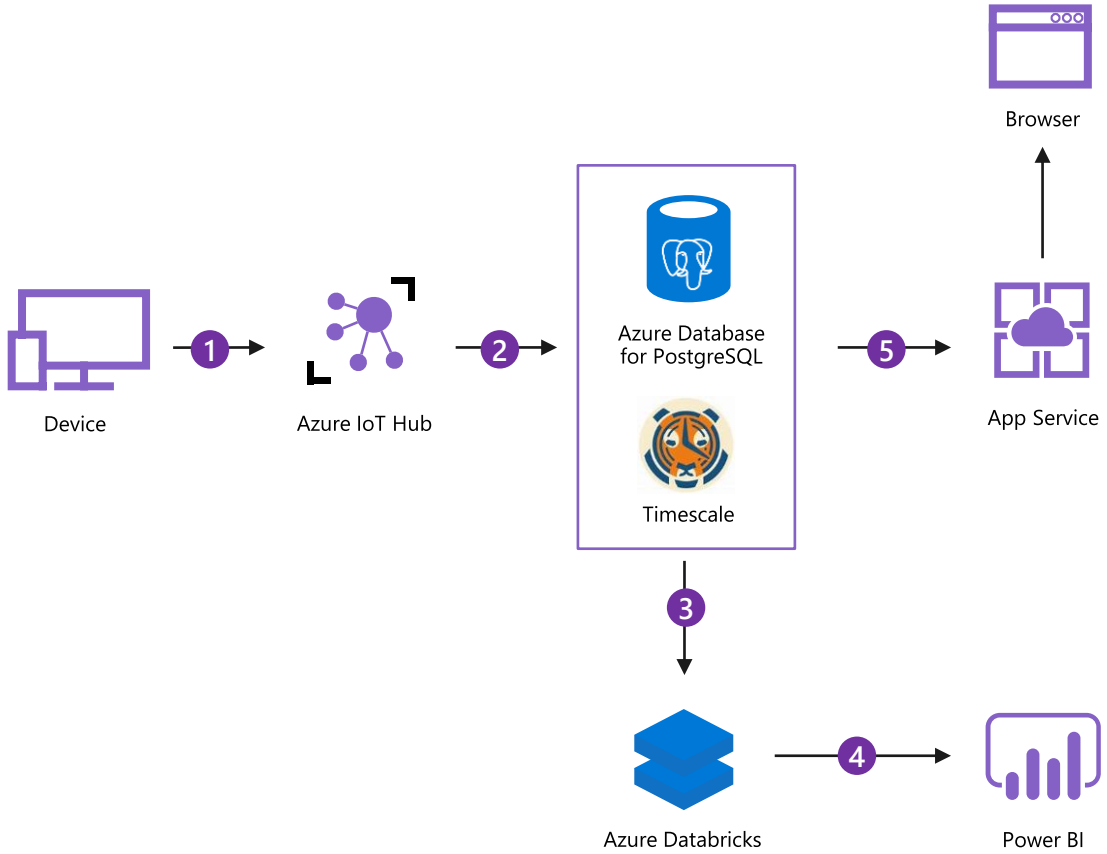
6. Use scalable machine learning/deep learning techniques, to derive deeper insights from this data using Python, R or Scala, with inbuilt notebook experiences in Azure Databricks.

7. Monitor your application's performance for degradation or anomalies and auto-scale your application to changing performance requirements.



Transactional and operational analytics workloads

1. Receive telemetry on state of your devices and message routes from billions of devices
2. Ingest data into a transactional database that allows you to performantly query timeseries data with richness of SQL
3. Perform offline rich machine learning or train predictive models
4. Report and visualize the state of your devices at a granular or aggregated level
5. Provide insights to users and operators on current device status



Hyperscale (Citus) use cases

Scaling PostgreSQL multi-tenant, SaaS applications

Real-time operational analytics

Building high throughput transactional apps

Consider Hyperscale (Citus) when:

- *Database size to grow beyond 100 GB*
- *Query latency of 100 millisecond or less*

Real-time operational analytics and reporting

1. Stream millions of events per second from devices and sensors into a scalable system that speaks and understands Apache Kafka. Build downstream pipelines to process, manipulate, and ingest your data

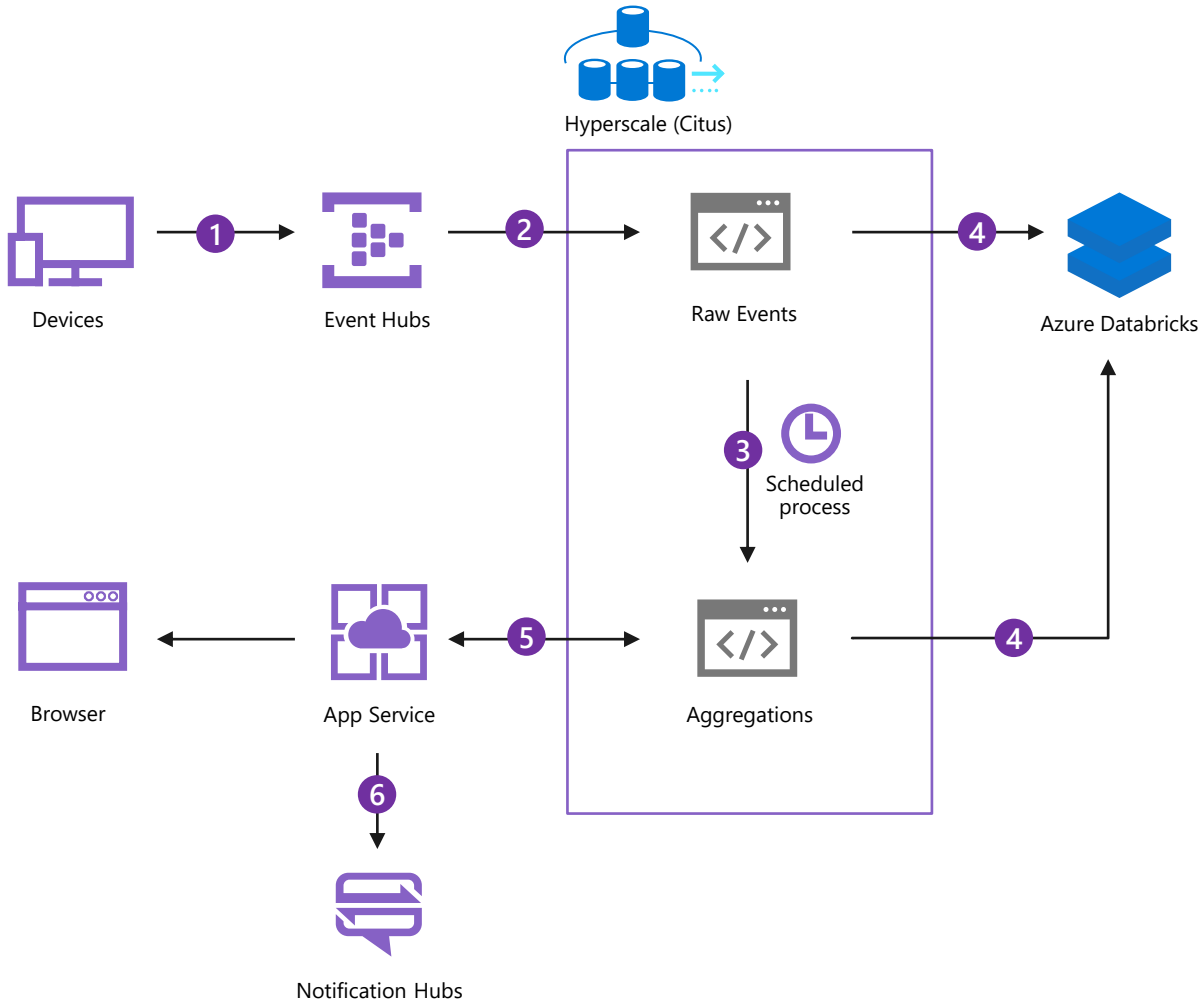
2. Ingest millions of raw transactional events into Hyperscale (Citus) per second, allowing you to query and alert on granular events

3. Perform incremental rollups directly in your database at granularities you define such as minutely, hourly, daily, to real-time reporting and dashboarding for down stream applications

4. Take advantage of Azure Databricks to clean, transform, and analyze the streaming data, and combine it with structured data from operational databases or data warehouses

5. Provide insights to users and operators on current device status

6. Push timely notifications directly to your users on their preferred service or medium



Microsoft Windows relies on Citus for mission-critical decisions

- **“Ship/no-ship decisions for Microsoft Windows are made using Hyperscale (Citus), where our team runs on-the-fly analytics on billions of JSON events with sub-second responses.**
- **Distributed SQL with Citus is a game changer.”**

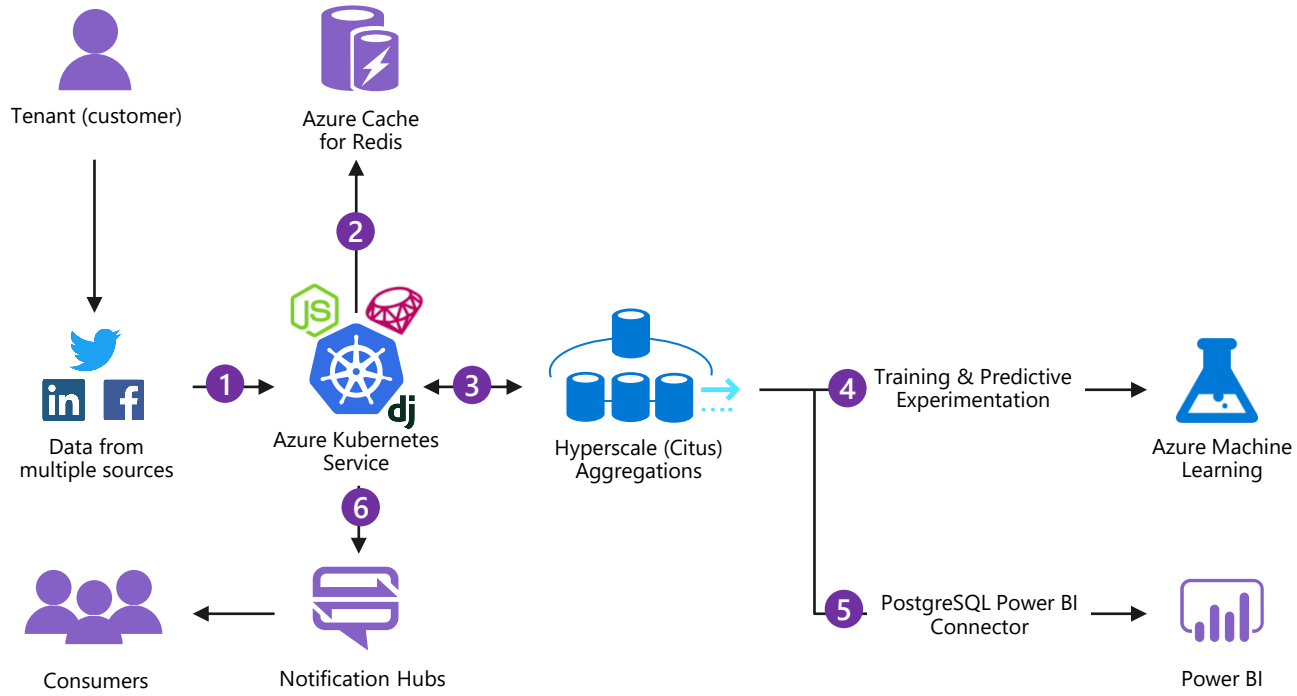
1 PB+ data

6M+ queries per day;

95% of queries completing < 4 secs

Scaling multi-tenant & SaaS applications

1. Ingest and sync data from disparate sources in real time with transactional guarantees
2. Offload database demands by managing sessions state and asset caching with Azure Cache for Redis
3. Shard by tenant and allow Hyperscale to elastically scale out your data. With co-location and tenant isolation features, don't worry about the scale limits of your database
4. Use scalable machine learning/deep learning techniques, to derive deeper insights from this data
5. Report and visualize the state of your devices at a granular or aggregated level
6. Push timely notifications directly to your users on their preferred service or medium



Citus helps ASB onboard customers 20x faster

- "After migrating to Citus, we can onboard Vonto customers 20X faster, in 2 minutes vs. the 40+ minutes it used to take. And with the launch of Hyperscale (Citus) on Azure Database for PostgreSQL, we are excited to
see what we can build next on Azure."

100 GB+ data

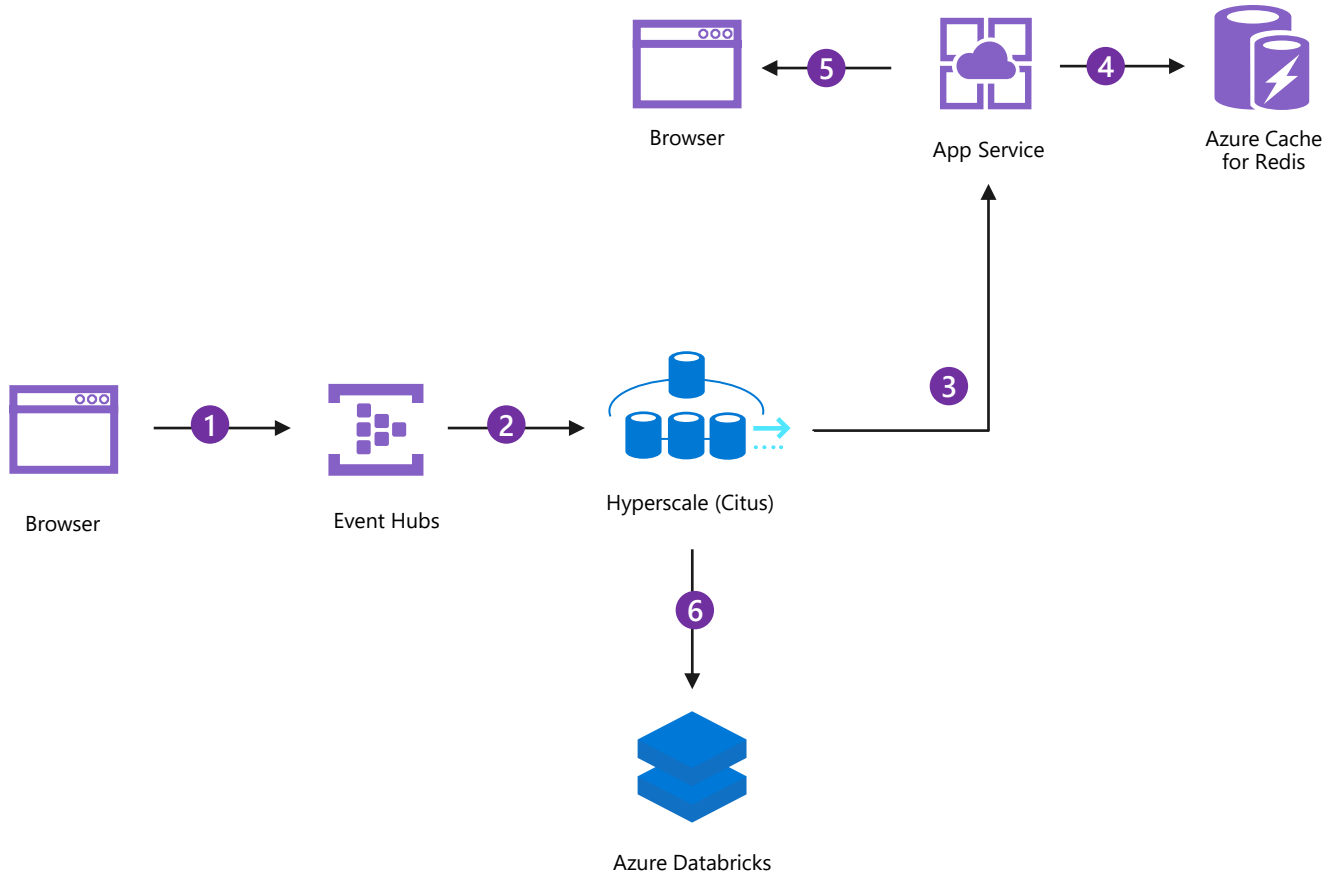
Multi-tenant SaaS application

Milliseconds latency

ASB

Building high throughput transactional apps

1. Ingest millions of events per second from devices and sensors into a scalable system that speaks and understands Apache Kafka. Build downstream pipelines to process, manipulate, and manipulate your data
2. Efficiently store and retrieve billions of records per day with sub second latency with Hyperscale (Citus).
3. Provide millisecond latency lookups across large data sets to apps coordinated with end users
4. Offload database demands by managing sessions state and asset caching with Azure Cache for Redis
5. Deliver data to the application for internal reporting and insights on health of devices and systems
6. Use scalable machine learning/deep learning techniques, to derive deeper insights from this data using Python, R or Scala, with inbuilt notebook experiences in Azure Databricks



Pex cuts hardware costs by 60%, serving sub-second queries on terabytes of data

- High throughput OLTP with analytics
- Low-latency
- Strong consistency
- Relational features including joins, triggers
- Replacing Hadoop / HBase
- Tried Cassandra

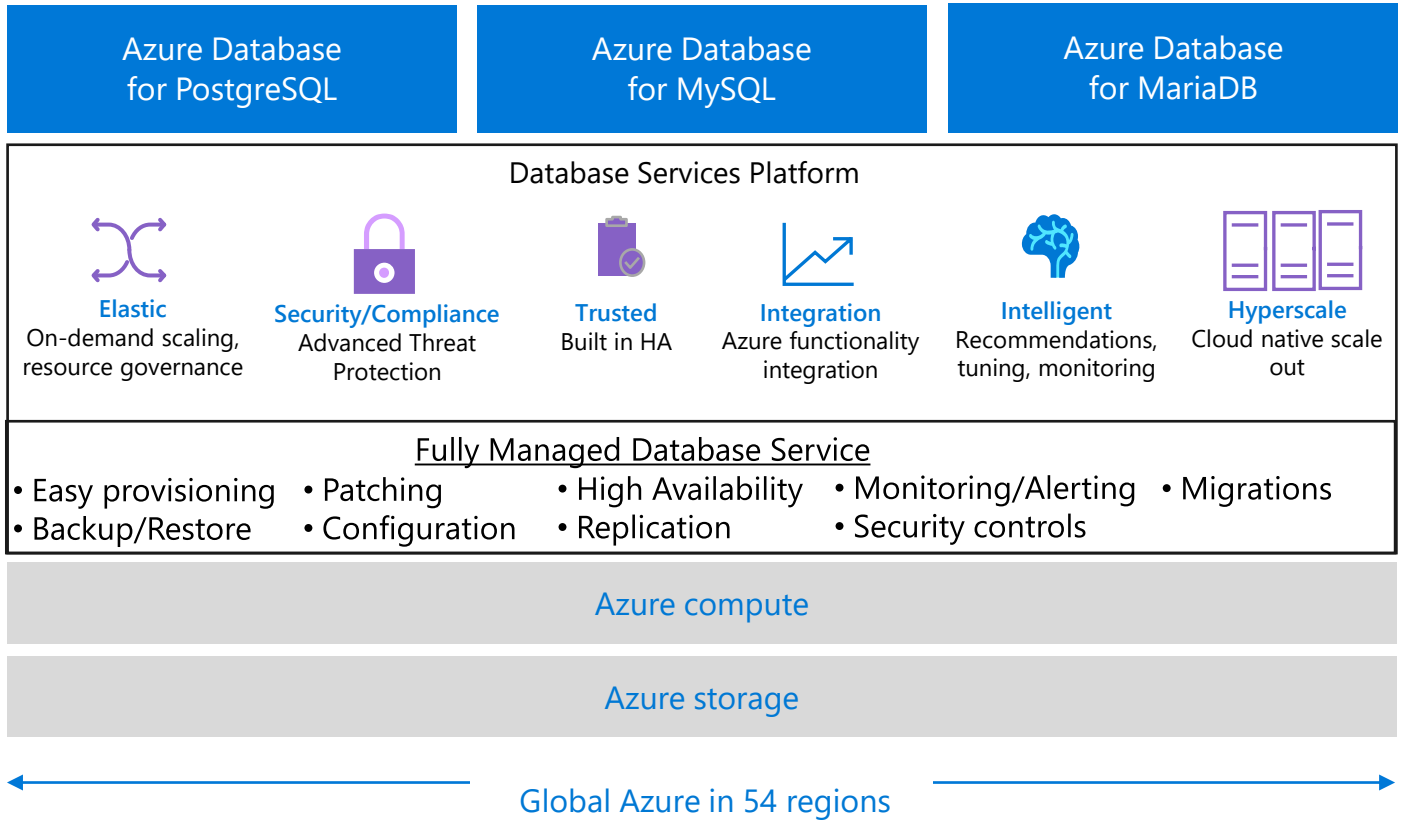
80TB+ of data
60k TPS
80B rows updated / day
1280 CPU cores
21 Citus nodes (**60% reduction** from 55 nodes on HBase)

"We would not be able to fulfill our data processing needs with any other database. And we tried! We really, really tried."
- Rasty Turek, CEO

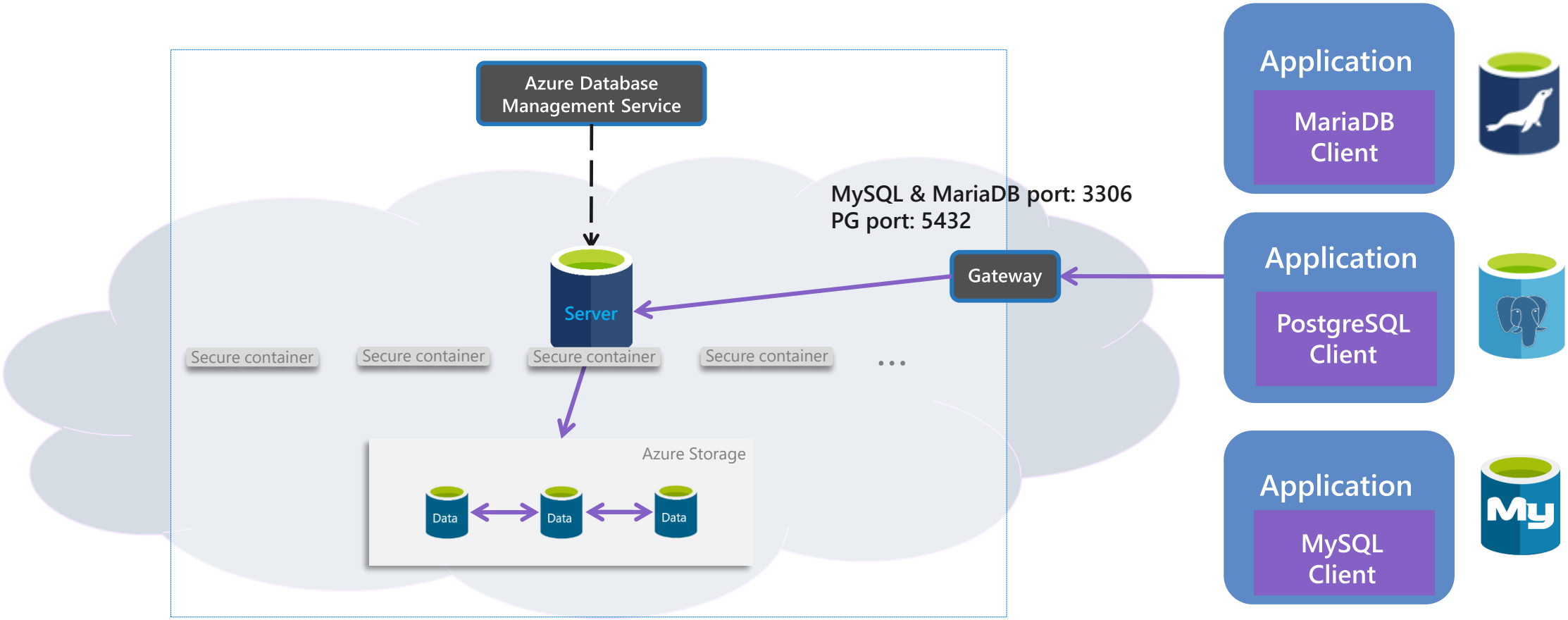
Microsoft Azure Solutions Business

Single server platform overview

Relational Database Services Platform



Azure Database Services Architecture



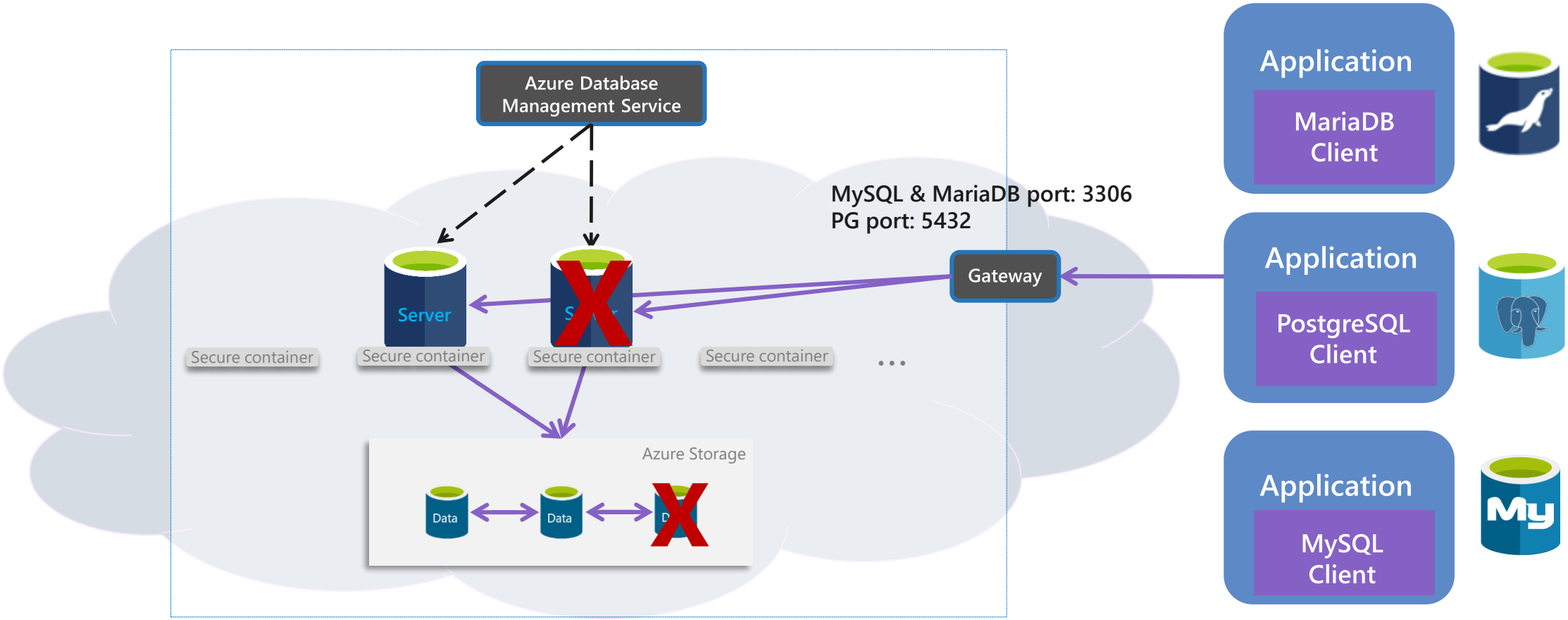
Gateway

- **Routes incoming connections to physical location of server**
- **Provides ability to offer 99.99% HA SLA**
- **Public IP – can be reached from tools like telnet and ping**
  - IP of actual server is still hidden
- **Additional network hop incurs latency**
- **Requires "username@servername" in username field**
  - Format is a change for customers who may already use these DB engines
  - Hostname field in connection string resolves to the gateway
  - Hence "username@servername" needed by gateway to determine backend server to connect to
  - Always remember to pass in the correct server name in the username field
    - *psql –h myserver.postgres.database.azure.com –U myuser@myserver –d postgres*
    - *psql –h myserver2.postgres.database.azure.com –U myuser@myserver –d postgres* **-> will connect to myserver**

High Availability

- **Built-in 99.99% HA SLA available on every single server & pricing tier**
  - Do not need to opt-in
  - Does not require any setup after create or an additional replica server
- **Management service constantly monitors health of servers**
- **Failure detection and spinning up of a new compute process can take 30-45 seconds**
- **When a server fails over, existing connections are dropped**
  - Best practice for an application to retry connections
- **Database recovery**
  - DB engine's native recovery process is used to bring a server back up after failover
  - Recovery may take longer if a large transaction was in progress at the time of failover
  - The service uses the transaction logs to replay transactions to recover

High Availability



Service tiers

	Basic		General Purpose						Memory Optimized				
Intended use case	Built for workloads with **light compute** and **variable I/O** performance.		Ideal for most **business workloads** offering balanced compute and memory with scalable I/O throughput.						Built for high-performance database workloads requiring **in-memory performance for faster transaction processing** and higher concurrency.				
vCore	1	2	2	4	8	16	32	64	2	4	8	16	32
Compute Generation	Gen 4, Gen 5		Gen 4, Gen 5						Gen 5 only				
Memory	2 GB per vCore		5 GB per vCore						10 GB per vCore				
Storage	5 GB – 1 TB Remote Magnetic Media		5 GB – 16 TB Remote SSD						5 GB – 16 TB Remote SSD				
IOPS	Variable		100 – 20000 IOPS						100 – 20000 IOPS				
Backup retention	7 – 35 days		7 – 35 days						7 – 35 days				
Backup storage	Locally redundant		Locally or geographically redundant						Locally or geographically redundant				

Compute

- **Scaling compute follows similar process as HA**
  - Can flexibly scale up/down (2 – 64 vCores depending on tier)
  - Requires a server restart to apply
  - Same notes about HA apply (ex. retry logic as best practice)
- **Memory tightly coupled with vCores**
  - For more memory, scale up vCores or consider using Memory Optimized tier
- **Maximum connections coupled with vCores**
  - Refer to documentation for connection limits
- **Majority of regions are using Gen 5 compute hardware except for China**

Basic vs. GP/MO

- **Basic tier should be used for small workloads that don't need guarantees for IO**
- **Basic cannot scale to GP or MO**
  - If customer is planning to move into production with the service, recommend starting with GP for dev/test
- **Basic uses Azure Standard Storage vs. Premium storage**
  - Standard Storage = no IO guarantee
- **Some features not offered on Basic**
  - Example: VNet service endpoints, geo-redundant backups

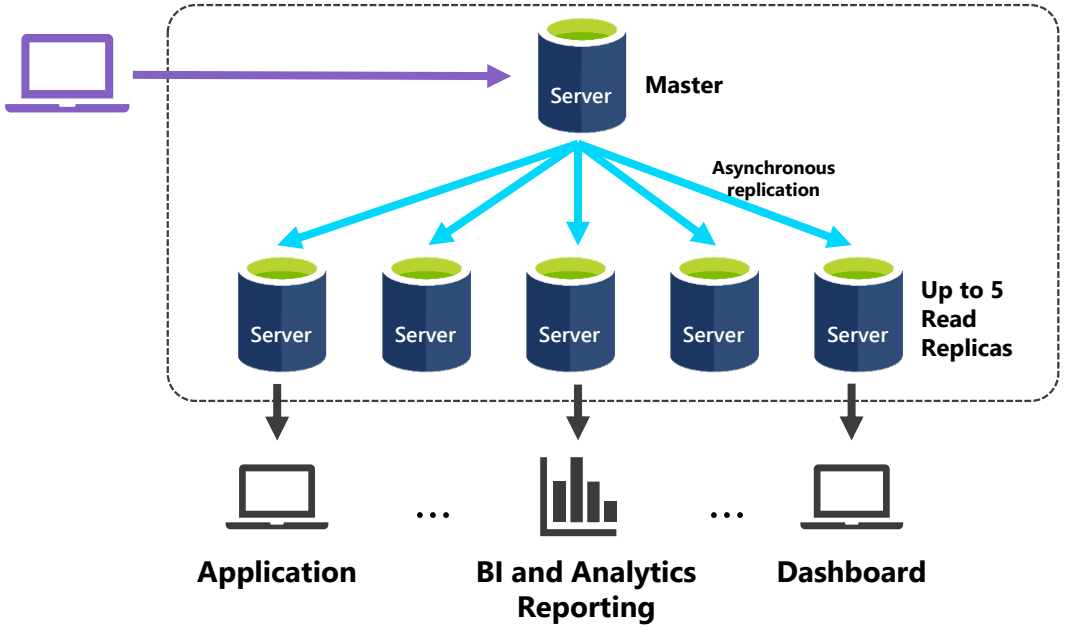
Storage

- Remote blob storage
- Maintain three copies of data files
- Scaling up storage does not require restart
- Scaling storage down not supported
- Storage size tightly coupled with I/O (3 IOPS/GB)
- Original storage
  - Up to 4 TB (GP/MO only)
  - Max 6K IOPs (GP/MO only)
- Large storage
  - Up to 16 TB
  - Higher I/O limits, max 20K IOPs
- Migration between storage options not currently supported
  - If customer requirements are I/O driven, start with large storage

Backup & restore

- **Automated full, differential, transactional backups**
  - Full: once a week
  - Differential: twice a day
  - Transactional: every 5 minutes
- **Backups provide ability to perform point-in-time restores (PITR)**
  - Can PITR to any time within the backup retention period
  - Backup retention period (7-35 days) is always configurable by customer
- **Choice of locally redundant or geo-redundant backup storage**
  - Cannot be changed after server create
- **No direct customer access to backups or transaction logs**

Read replicas



- Ability to create up to 5 read replicas; same or cross-region
- Allows read-heavy workloads to scale out
- Replicas are read-only; can be made read-write by stopping replication
- PostgreSQL uses WAL logs
- MySQL/MariaDB uses binary logs

Read replicas cont'd

- Requires server restart to create first replica
- Depending on location of master server, there are some regional restrictions for read replicas
  - Refer to docs for table of restrictions
- Read replicas are created as new servers. Existing servers can't be made into replicas
- Read replicas have their own distinct hostnames
- Load balancing must be configured by customer
  - Automatic failover is not supported

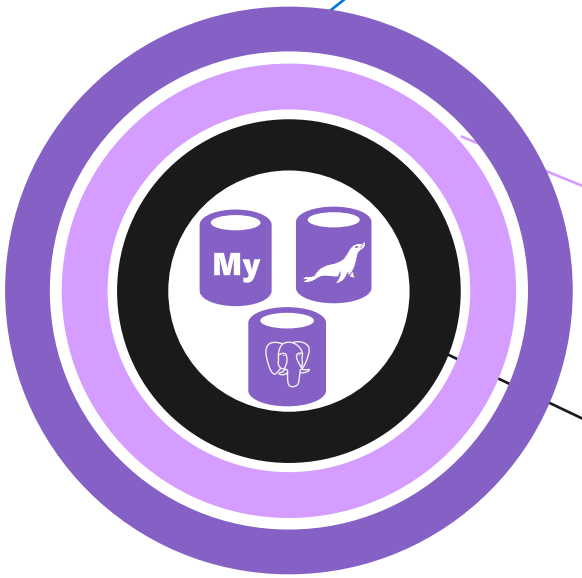
Disaster recovery

- **Geo-restore**
  - Requires server to be created with geo-redundant storage
  - Cost:
    - Before restoring, charged for backup storage
    - After restoring, charged for compute and storage of newly created server
  - Time to recover:
    - When restoring to the geo-paired region, restore time is similar to performing PITR. Time depends on server size and time since last full backup
    - When restoring to a region that is NOT the geo-paired, restore time is longer due to additional network latency
- **Cross-region read replicas**
  - Cost:
    - Replica must be running before disaster occurs. Replica is charged for compute and storage like normal servers
  - Time to recover:
    - Uses continuous asynchronous replication. Server is available for write workloads once “stop replication” command has been run

Protect data with layers of security and leading compliance

Azure provides multiple layers of security to safeguard your data

Support for leading compliance offerings



Monitor access

Native authentication

Advanced Threat Protection

Control access

Secure SSL connectivity

Server firewall rules

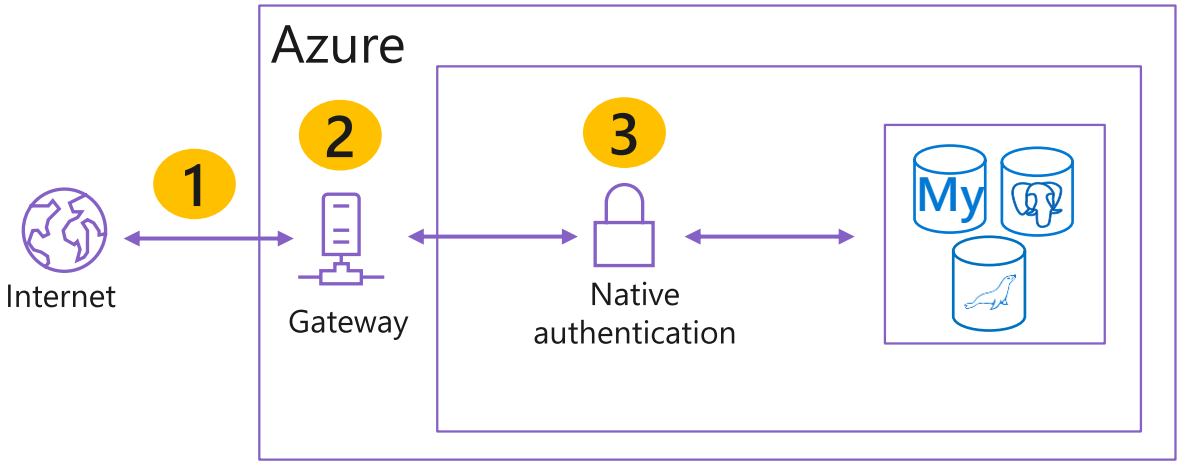
Virtual Network (SE)

Protect data

Built-in encryption for data at rest and in motion

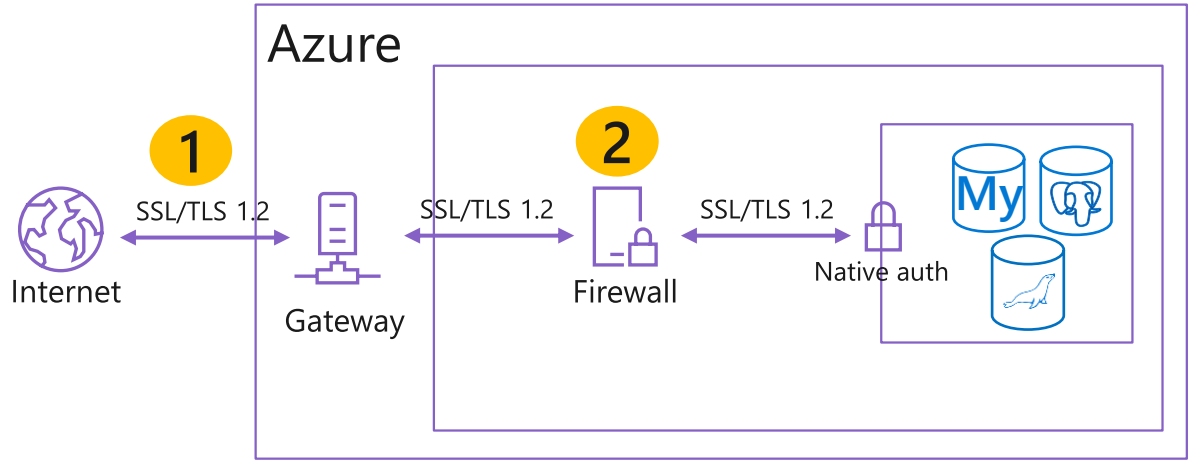
Layered security starts at the edge

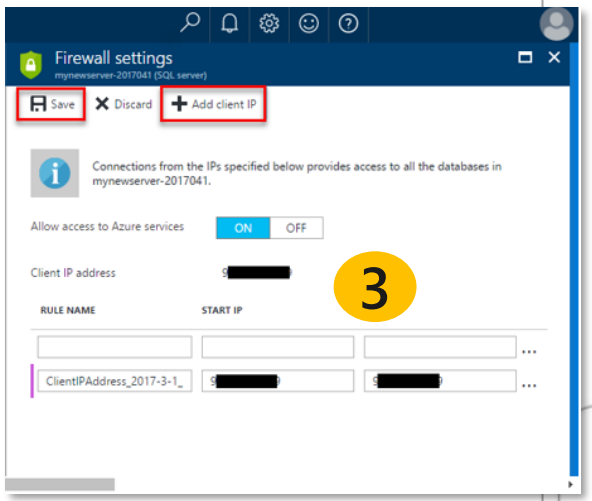
1. A connection is never directly to a database instance. It must first pass through Azure edge network protection.
2. A gateway services the connection request next.
3. Native MySQL, PostgreSQL, MariaDB authentication for connection request.

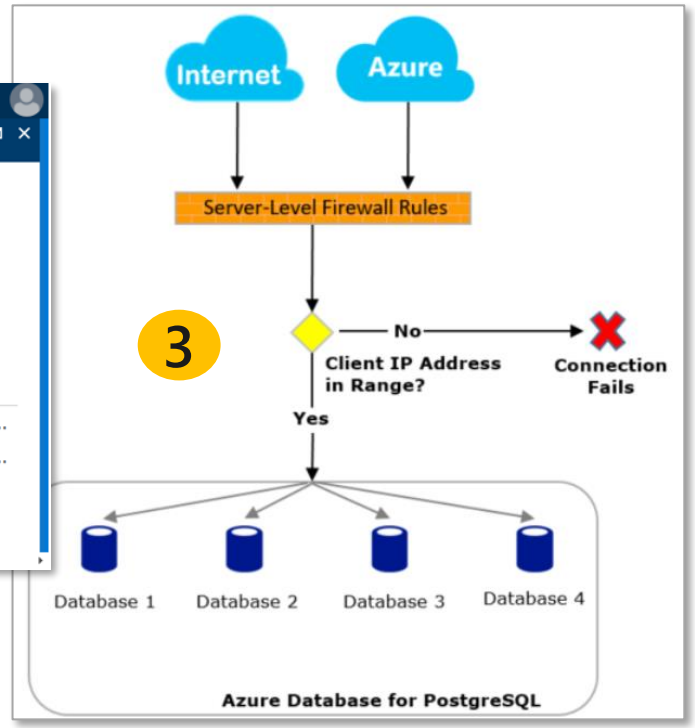


Access control provide additional security

1. **SSL / TLS is enforced by default.**
2. Server firewall prevents all access to your database server until you specify which computers have permission.
3. Configure firewall to allow all connections from Azure including connections from the subscriptions of other customers. Make sure your login and user permissions limit access to only authorized users.

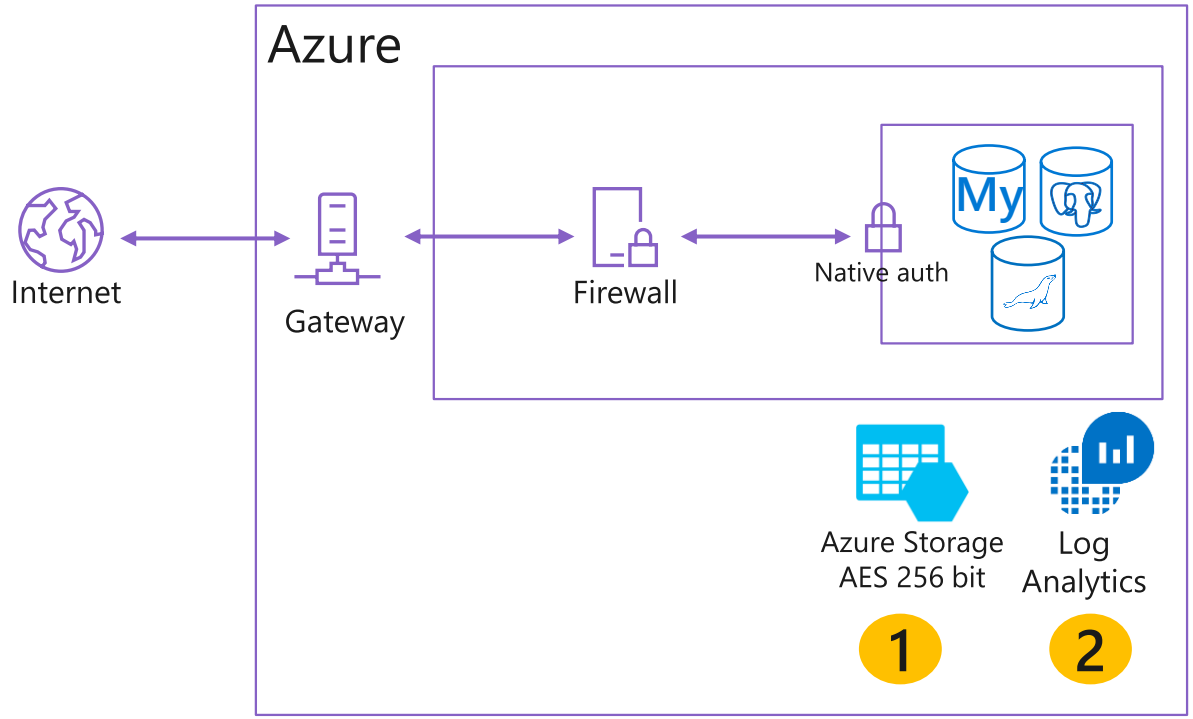






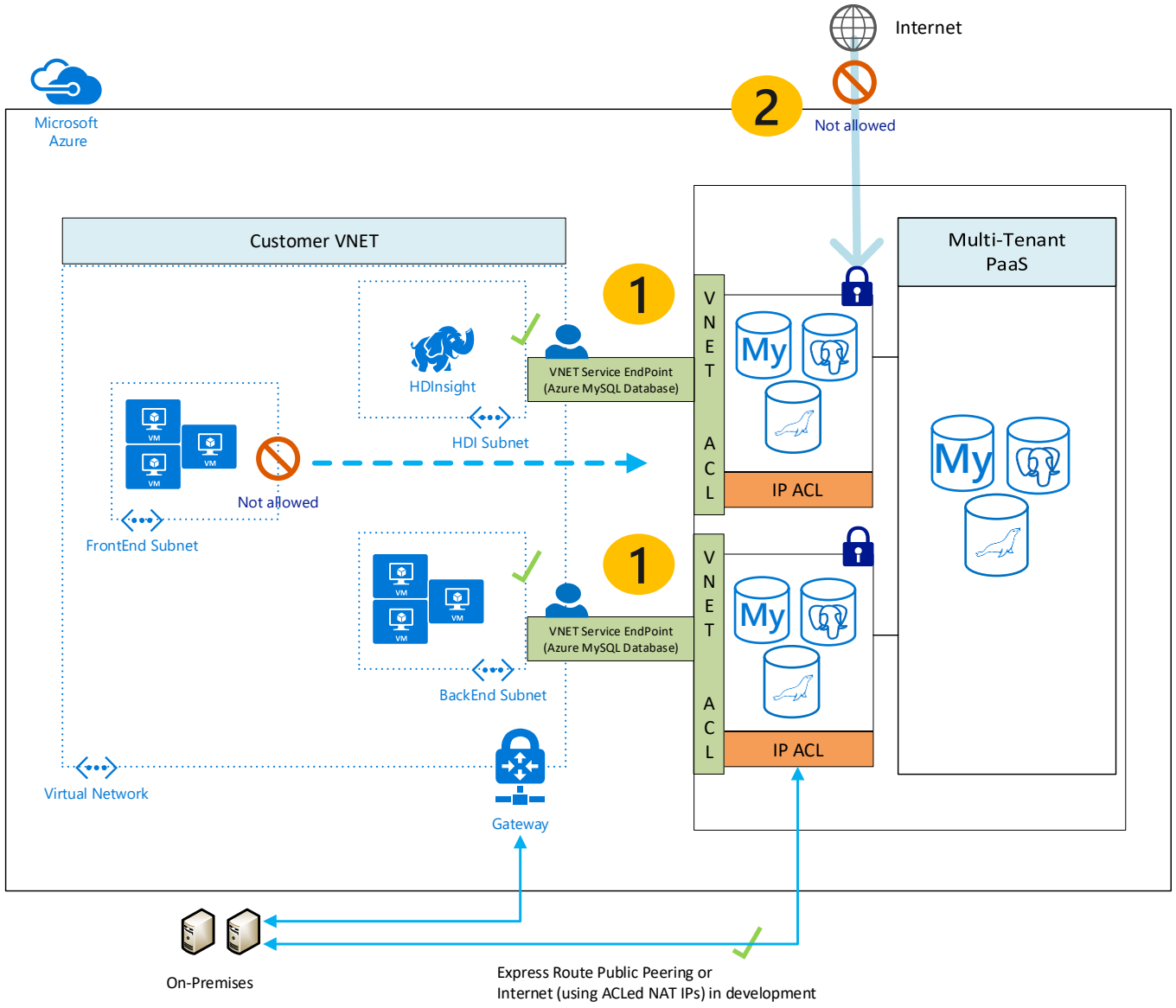
Data encryption and logging ensures protection

1. Encryption of data files and backups at rest is AES 256-bit, system managed and always on.
2. Built in activity logging facilitates security monitoring. Native Azure Monitor integration.



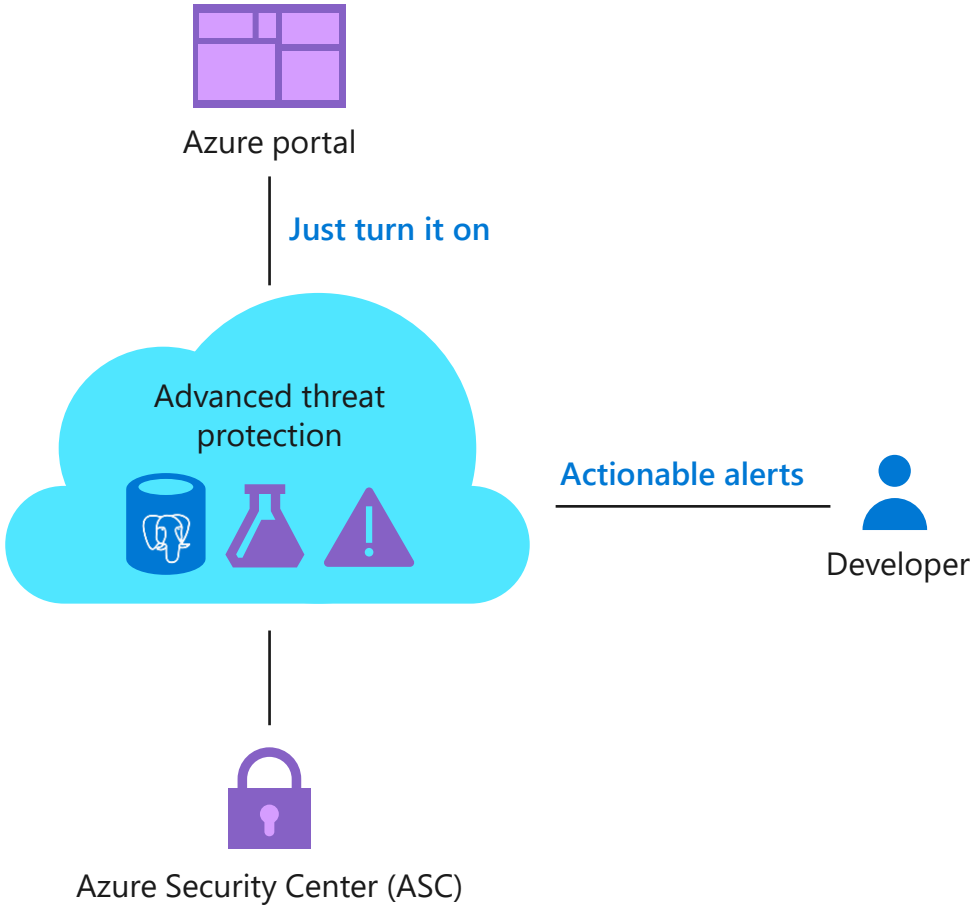
VNet service endpoints enable private, fine-grained access

1. Virtual network rules are one firewall security feature that controls whether your server accepts communications that are sent from particular subnets in virtual networks.
2.
   - VNet enables private access to tagged “Microsoft.SQL” services. These service includes Azure SQL Database, Azure SQL Database Warehouse and Azure OSS Databases.



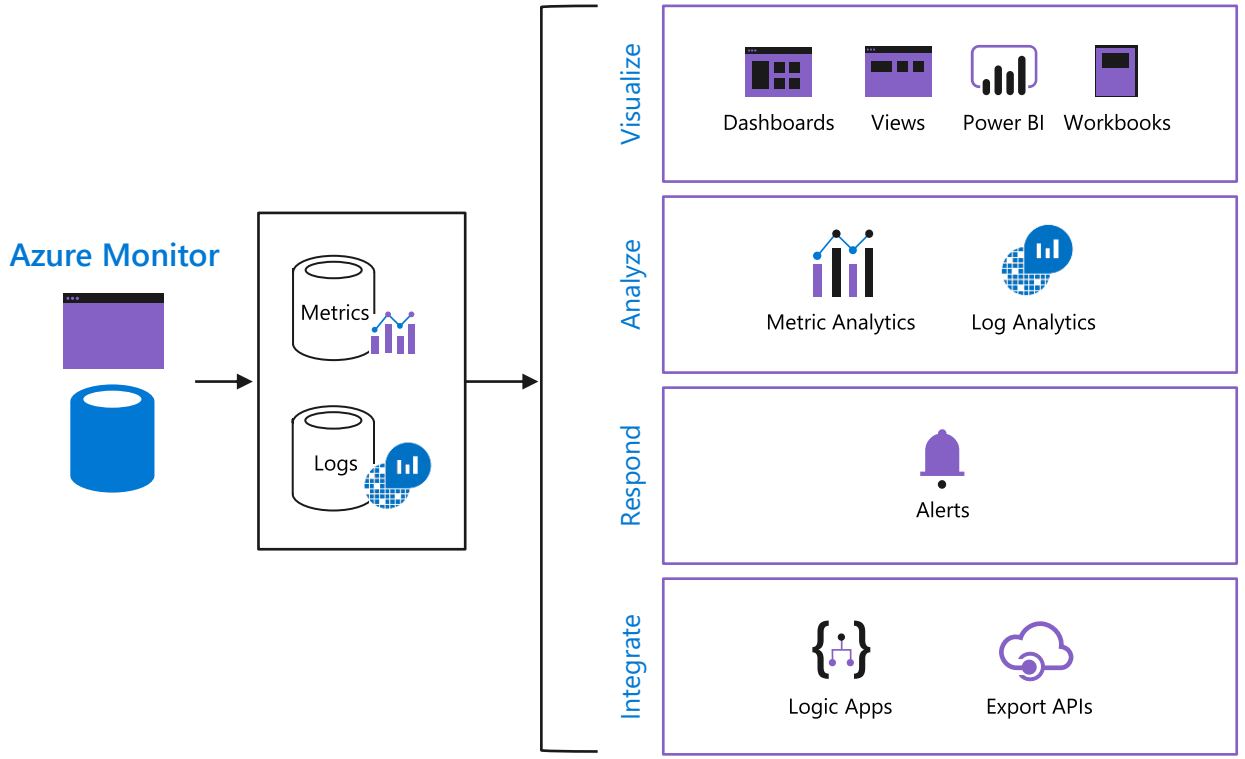
Protect against malicious action with Advanced Threat Protection

- Advanced threat protection (ATP) uses signals to identify and investigate user-based threats
- After creating a behavioral baseline for each user, Azure ATP identifies suspicious activity with built-in adaptive intelligence
- Reduce vulnerabilities with configuration insights and suggested best-practices
- Decrease alert noise with visibility to simple lateral movement paths and real-time attack timelines



Monitor your database with Azure Monitor

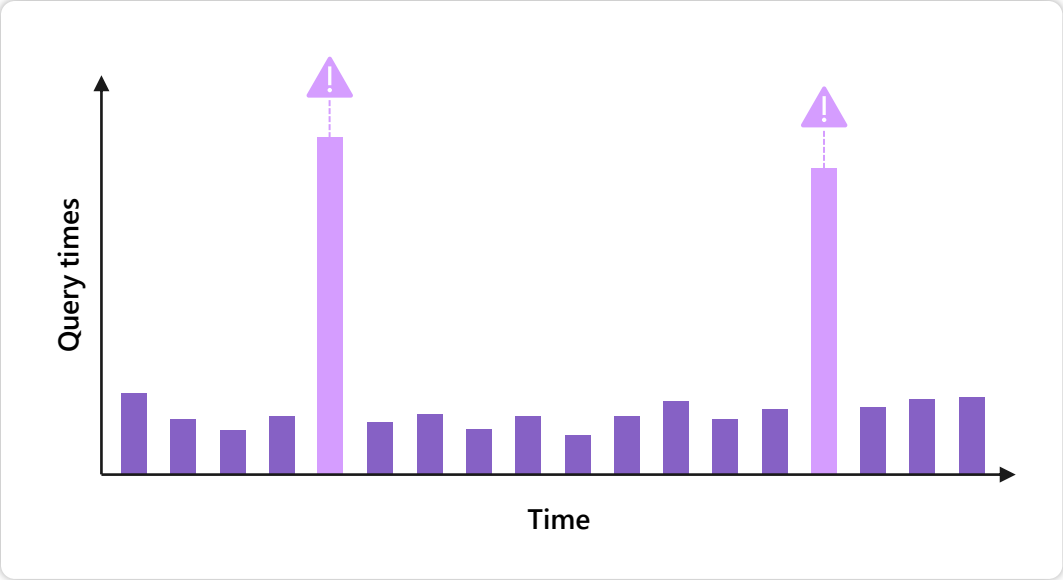
- **All servers are configured with metrics out-of-the-box**
  - Metrics are retained for 90 days
  - Metrics include CPU, storage, memory, connections, etc.
- **Alerts can be configured**
- **Logs can be sent to Azure Monitor Diagnostic logs for additional analysis and longer retention periods**
  - PostgreSQL: server logs, audit logs
  - MySQL/MariaDB: slow query logs, audit logs



Capture query data with Query Store

- Query Store simplifies performance troubleshooting by helping you quickly find the longest running and most resource-intensive queries
- Capture a history of queries and runtime statistics by time windows to reveal usage patterns
- Opt-in to enable Query Store globally across all of your PostgreSQL, MySQL and MariaDB databases

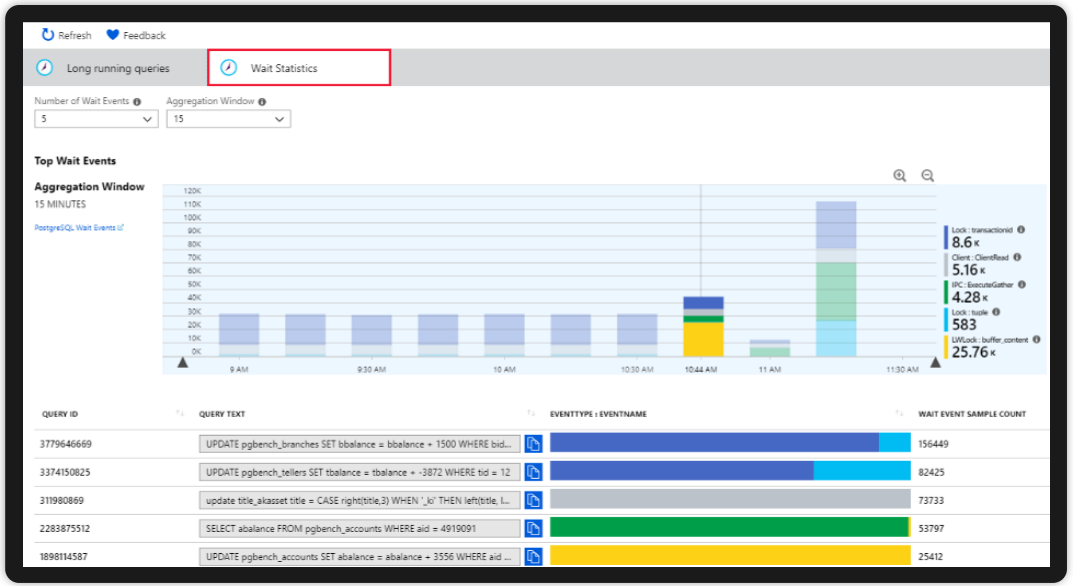
Length of individual queries with Query Store



Review performance details with Query Performance Insight

- Quickly identify your longest running queries,
  how they change over time and what waits are affecting them
- The Query Performance Insight view in the Azure portal surfaces visualizations on key information from Query Store
- The long running queries tab shows the top five queries by average duration per execution, aggregated in 15-minute intervals
- Users can view additional queries and alter the time window

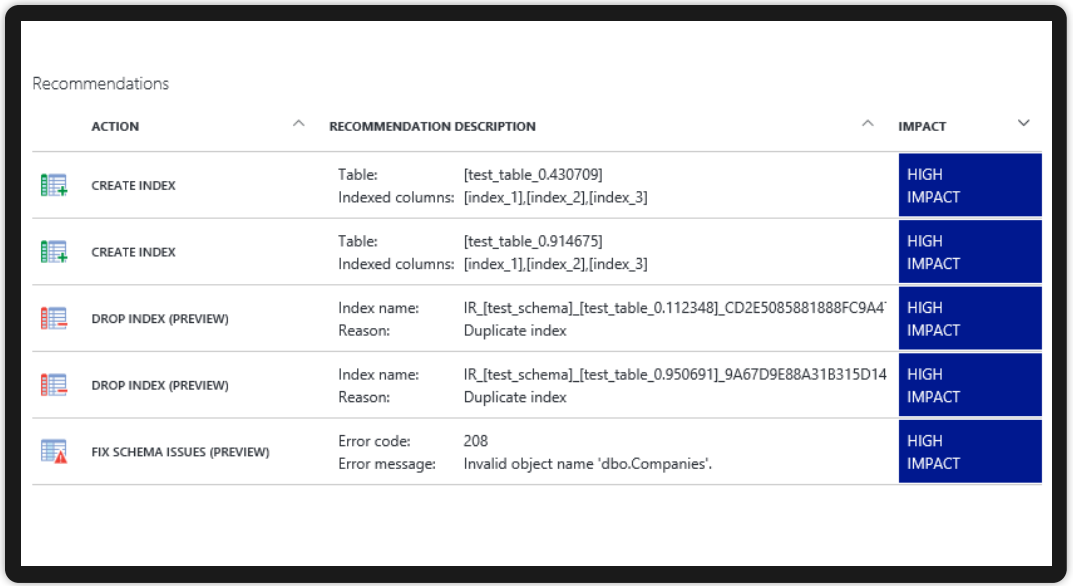
Wait statistics in Query Performance Insight



Customize suggestions with Performance Recommendations

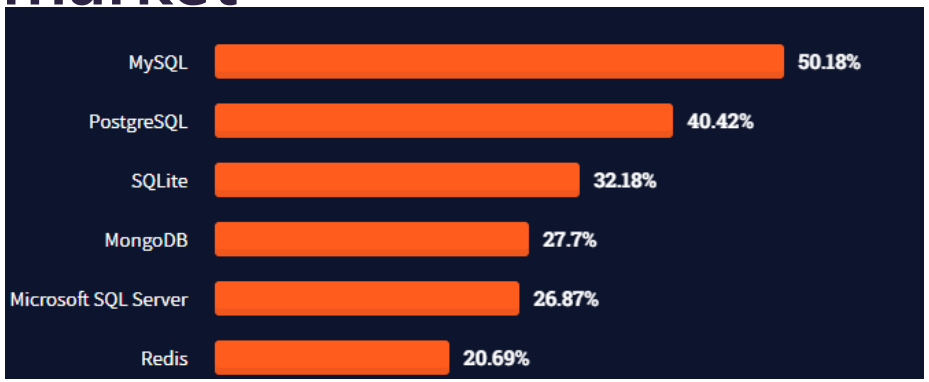
- **Performance Recommendations analyzes your database to create customized suggestions for better performance**
- **Two types of recommendations are supported:**
  - Create Index recommendations suggest new indexes to accelerate the most frequently run or time-consuming queries
  - Drop Index analyzes existing indexes and recommends dropping indexes that are rarely used or redundant (currently only available in PostgreSQL)

Azure portal performance recommendations



Why managed PostgreSQL in Azure?

- **By 2022, 70% of new in-house applications are developed on OSS RDBMs platforms**
- **PostgreSQL is one of the most reliable, scalable, stable and secure open-source RDBMS in the market**
- Benefits of PaaS – built in administration, backups, performance insights and high availability
- Lots of parity with Oracle features and language set
  - More and more customers are migrating to PostgreSQL from Oracle



Most common databases used by developers

Azure PostgreSQL Deployment Options

Single Server

- Best suited for cloud native applications with minimal customization requirements
- Designed to handle automated patching without the need for granular control on the patching schedule

Flexible Server (Currently in Preview, GA November)

- More granular control and flexibility over database management and configuration
- Zone redundant high availability
- Start/Stop capability with burstable SKU for cost optimization

Hyperscale (Citus Extension)

- Horizontally scales queries across multiple machines using sharding
- Parallelizes incoming SQL queries across servers for faster responses on large datasets

Azure PostgreSQL Single Server Challenges

Connections are redirected to database via a Gateway connection	• Changes to typical connection string
No control for co-locating application and database	• Ideally all resources in the same Availability Zone in an Azure Region
Connection pooling	• PostgreSQL connections are resource-intensive
Maintenance	• Customers cannot control maintenance windows

PostgreSQL Flexible Server (Preview, GA November)

- **Hosted on a single tenant VM in Azure**
  - Linux Operating System
  - Community Edition of PostgreSQL (Versions 11,12,13 to date)
- **Connections can be made directly to Flexible Server**
  - Eliminates the need for the Gateway Service
- **Allows for the colocation of the PostgreSQL database server with client tier**
  - Likely the biggest improvement customers will see vs Single Server
- **General Availability: November 30, 2021**

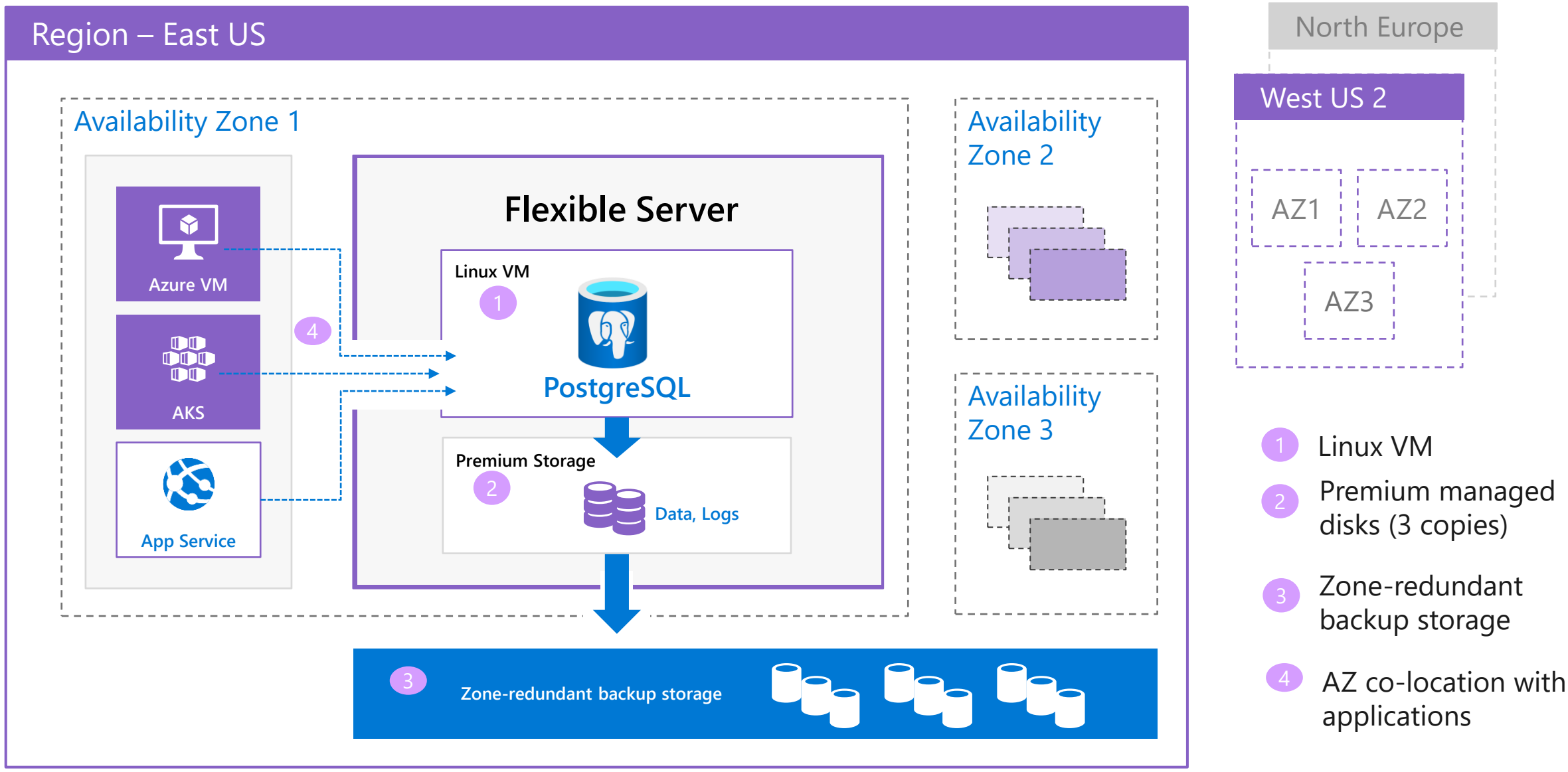
PostgreSQL Flexible Server (Preview)

- **Gives you options to schedule maintenance**
  - One maintenance window every 30 days at your choosing
- **Built-in pgBouncer connection pooling**
  - Use port 6432 instead of default port 5432
  - Up to 5K connections
- **Ability to Start/Stop the server**
  - Some configuration changes require a server reboot

Flexible Server Pricing Tiers

Burstable	General Purpose	Memory Optimized
• Best for workloads that don't need the full CPU continuously • 1-2 vCores • Lost cost development and concurrency needs • No Zone-Redundant HA option	• Workloads that require balanced compute and memory with scalable I/O throughput • Examples: servers for hosting web and mobile apps and other enterprise applications	• High-performance workloads that require faster transaction processing and higher concurrency • Examples: real-time data and high-performance OLTP

Flexible Server Architecture



Flexible Server Zone Redundant High Availability

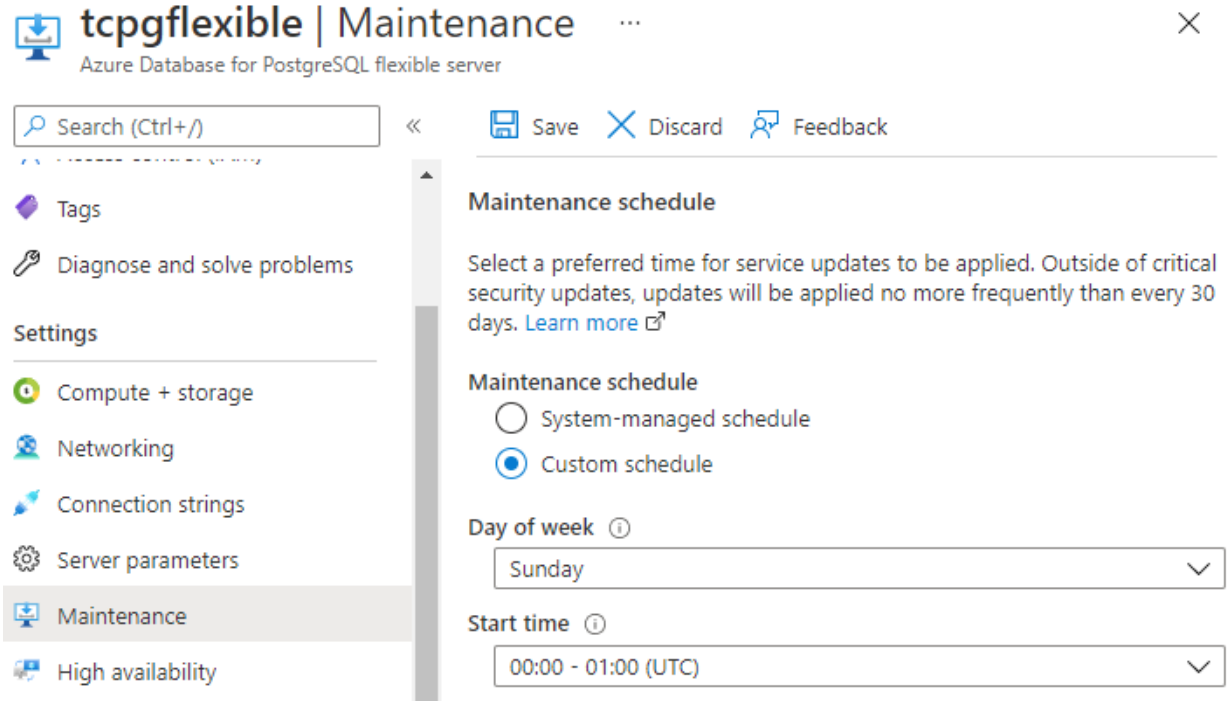
- **When enabled, a hot-standby server with the exact configuration will be spun up**
  - Same configuration for compute and storage in a different Availability Zone (in the same Region)
  - Allows for fast failover and high application availability should the primary server become unavailable
- **Primary Server failures are automatically detected, and failover occurs to the standby**
  - Applications can connect to the new server without a change to the connection string

Flexible Server Updates

- Maintenance occurs to keep databases secure, stable and up-to-date
- When HA is configured, updates are applied to the standby server first
  - Once the standby is updated, connections are drained on the primary and the failover occurs
- Applications reconnect with the same database server name (new primary) and resume operations
- A new standby server is then established in the same zone as the previous primary server

Maintenance Windows

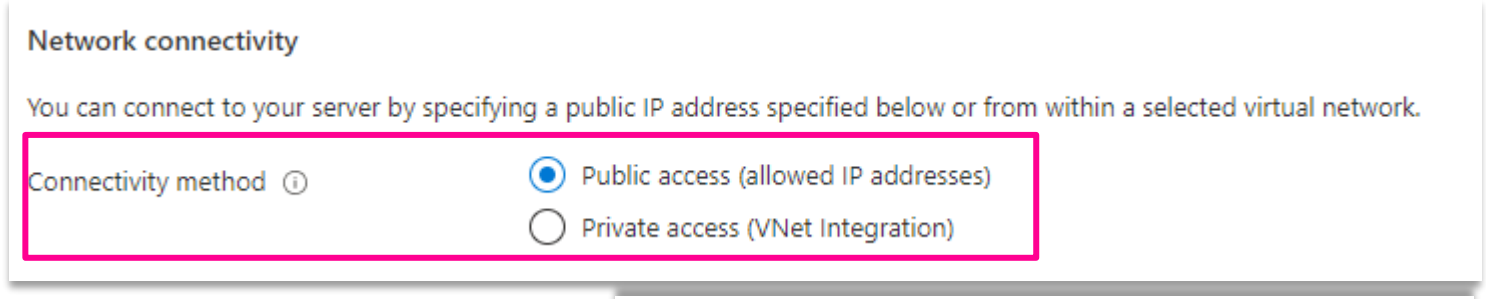
- Pick a day and time when activity is expected to be low
- If you do not choose a window, a 1-hour window between 11pm and 7am local time is chosen for you
- Get alerted 5 days before maintenance is run
- Check Azure Service Health notifications for the server to find downtime

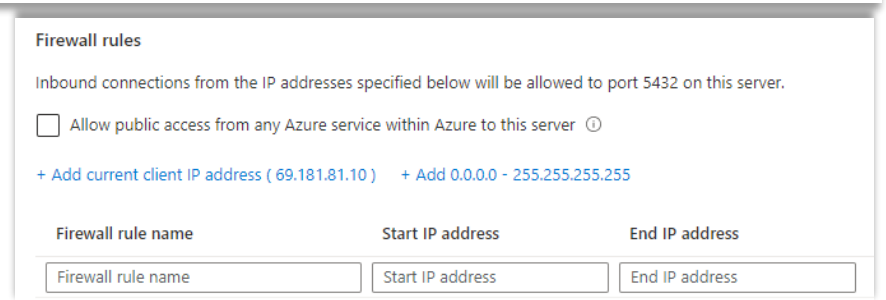


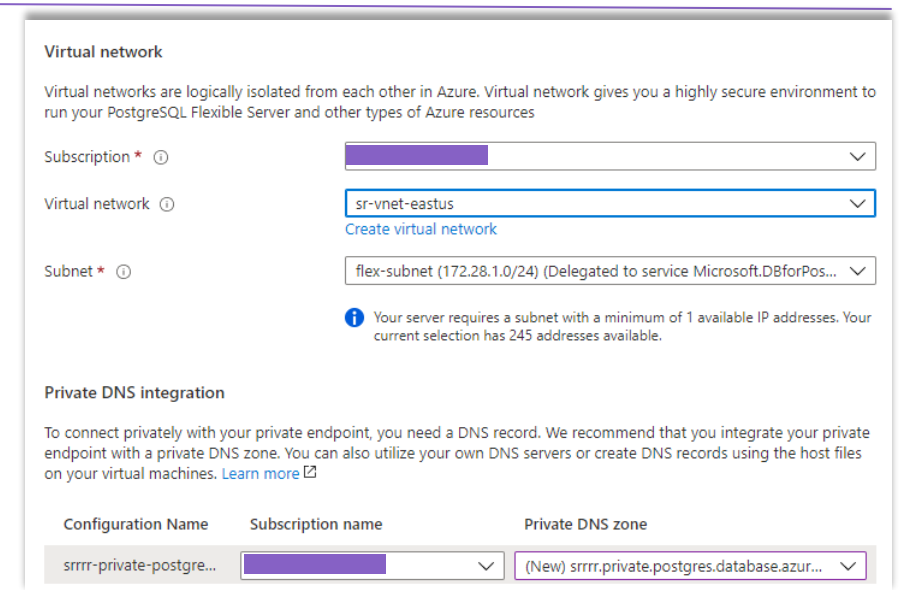
Network Isolation & Security

- **Public access**
- Restricted by firewall rules
- Add client access list

- **Private access** (Virtual Network)
- Inject Flexible servers to a delegated subnet
- Client can be in the same VNET or a different VNET in the same region or a different region







Backup & Restore

Backup

- Backup is taken immediately after the server is created
- Backups are stored in zone redundant storage within a region
- WAL are archived continuously

Restore

- Backup files are restored from snapshot backups
  - WAL files are used to bring the database to a consistent state
- Restore always creates a new server
- Restore time depends on database size, volume of WAL files to process, network bandwidth and region activity

Performance Insights – Coming Soon!

- Based on our Query Store extension
  - Must enable this server parameter
- Shows the most expensive queries on the system
- Gives recommendations for creating/dropping indexes
- Will be released sometime after Flexible Server GA

Flexible Server Roadmap

Feature	Approximate Availability
Updated PowerShell Module	1st Half CY 22
Same region Read-Replicas	1st Half CY 22
In-place Version Upgrade	1st Half CY 22
Azure Active Directory Integration	1st Half CY 22
Bring Your Own Key	1st Half CY 22
Cross-region Read-Replica	2nd Half CY 22

Microsoft Azure Solutions Business

Hyperscale platform (Citus) overview

Microsoft Azure Solutions Business

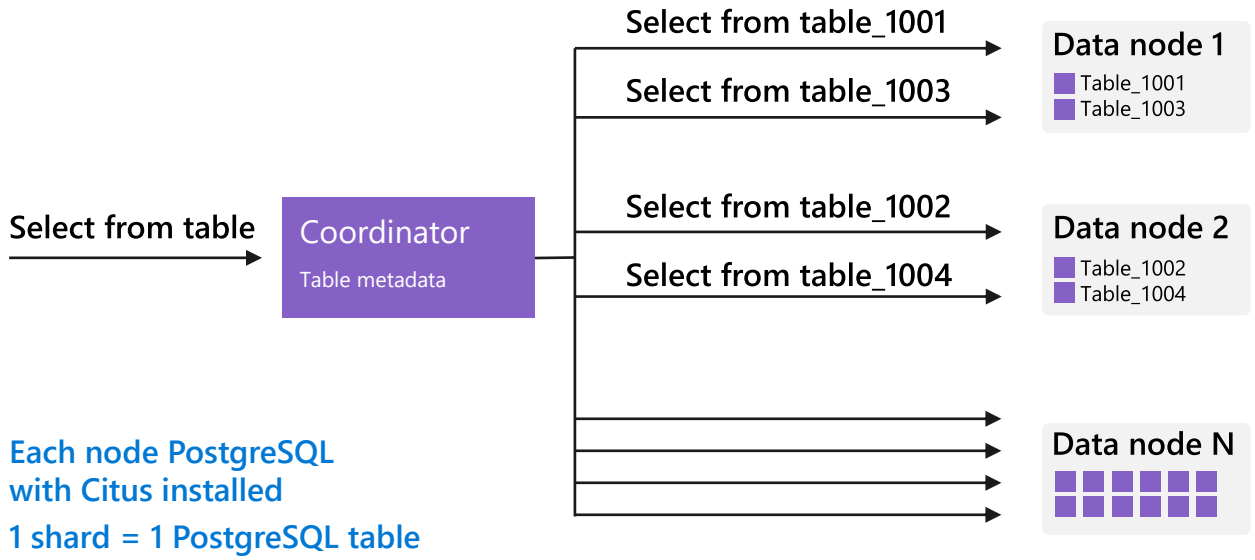
Architecture

Citus Architecture

Shard your PostgreSQL database across multiple nodes to give your application more memory, compute, and disk storage

Easily add worker nodes to achieve horizontal scale, while being able to deliver parallelism even within each node

Scale out to 100s of nodes



Terminology

- Coordinator – Stores Metadata. Node which application connects to.
- Worker / Data nodes – Nodes which store data in form of shards.
- Sharding – Process of dividing data among nodes.
- Shards – A postgres table on the worker containing a subset of rows.
- Distribution column / key – Column used to distribute data among nodes.

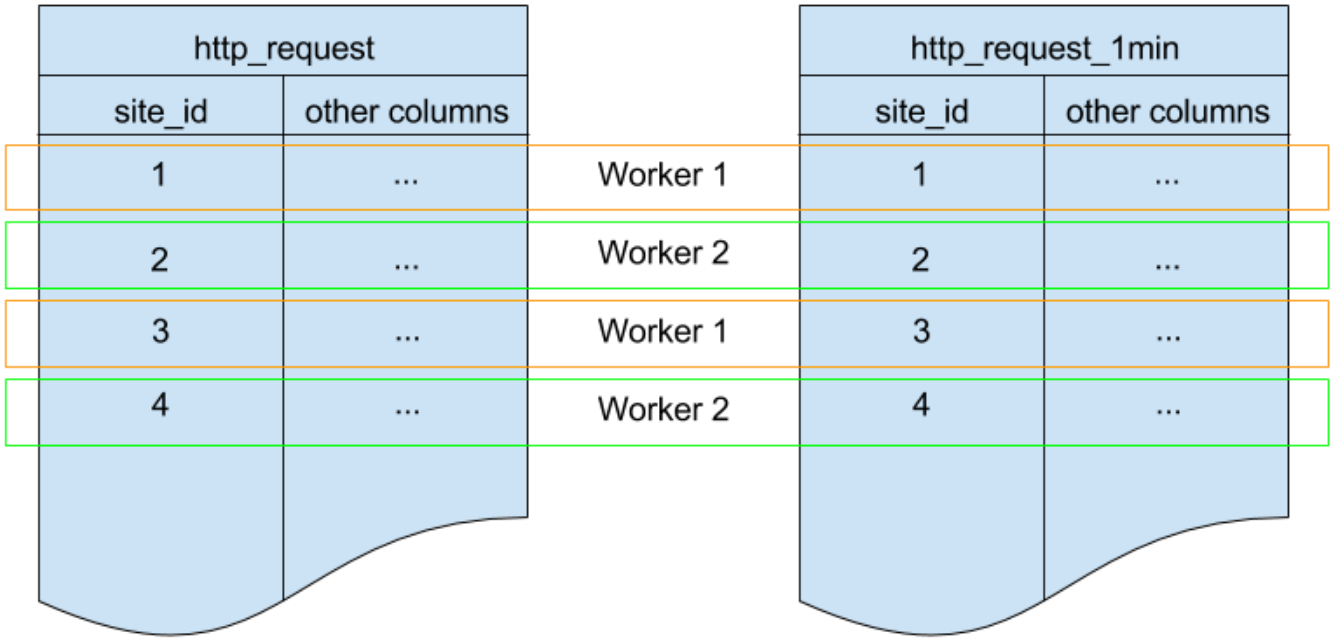
Hash based distribution

- Choose an column - distribution column
- Pre-create shards based on a hash function
- Each shard represents a range of hash values of the distribution column
- Route incoming rows based on the partition column
- Number of shards greater than number of nodes
- We use hash functions that postgres natively provides.
  - Based on datatype: hashint4, uuid_hash, hashtext etc

Co-location

- Tables sharded on the same distribution column are co-located.

Shards of both tables holding the same set of distribution column values are on the same worker.



Co-location based on data-type of the distribution column. Not the name of the column.

Co-location handles

- Joins
- Foreign keys/ Primary keys
- Rollups
- Others in future slides…

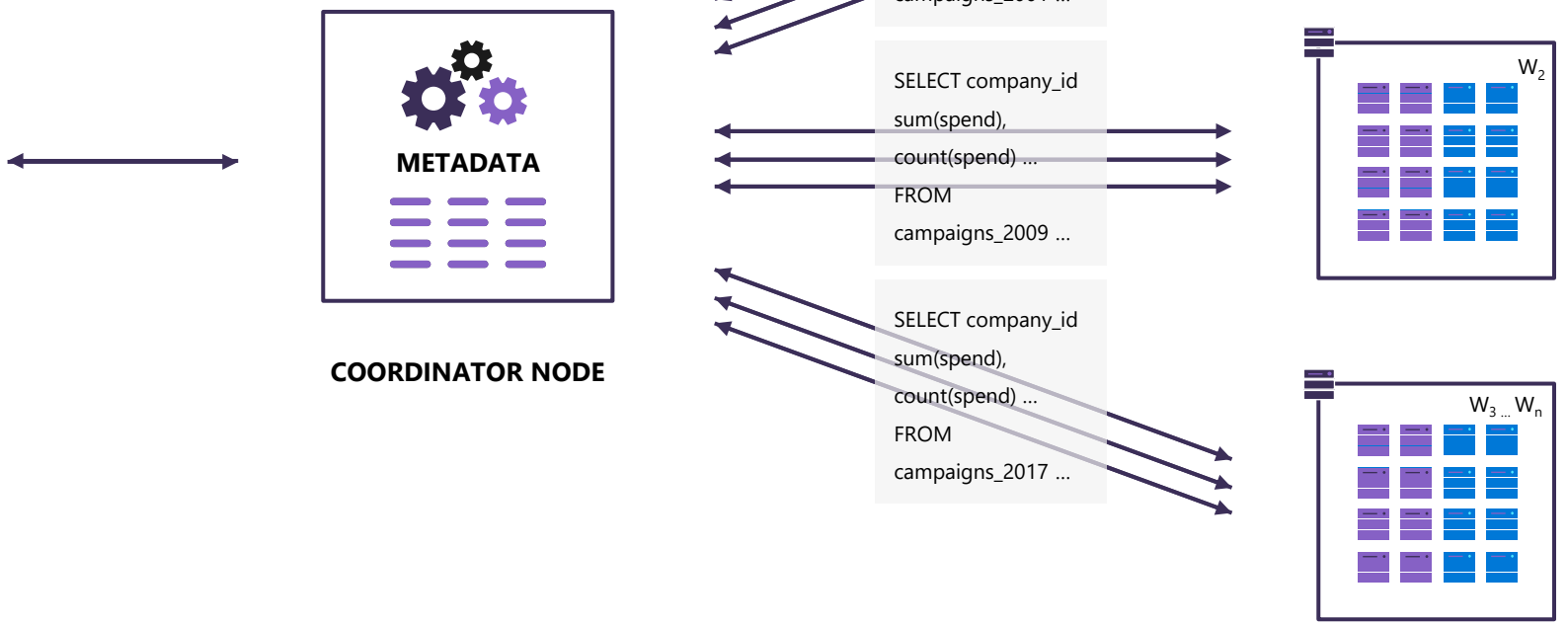
Scaled-out aggregate

Aggregating data before transactions avoids rewriting each row and can save write overhead and table bloat

Bulk aggregation avoids concurrency issues

- **APPLICATION**

```
SELECT  company_id
        avg(spend) AS avg_campaign_spend
FROM    compaigns
GROUP BY company_id
```



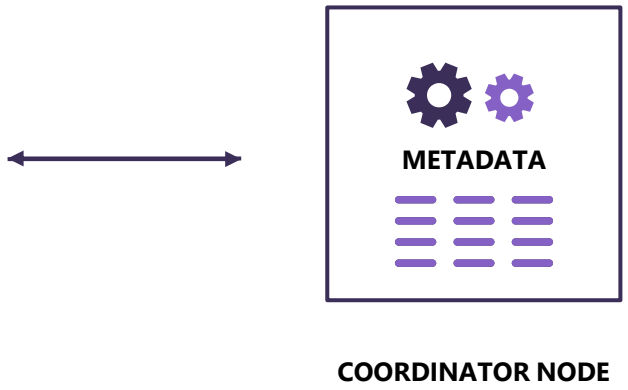
Scaled-out transaction

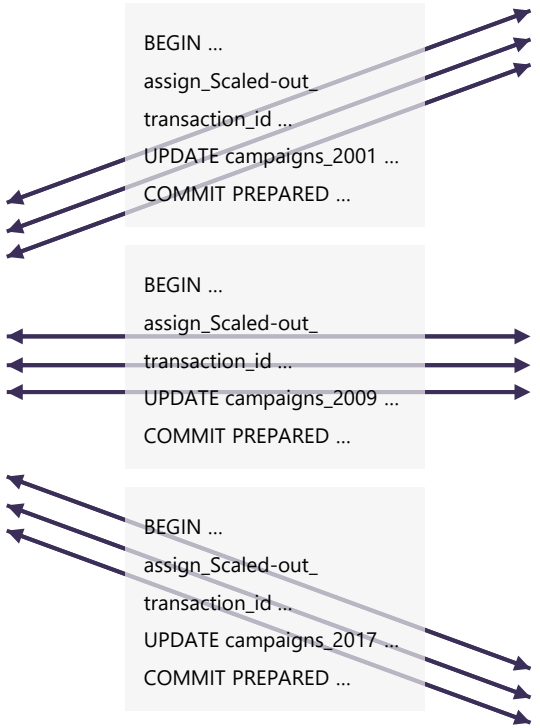
Hyperscale (Citus) leverages built-in 2PC protocol to prepare transactions via a coordinator node

Once worker nodes commit to transactions, release their locks, and send acknowledgements, the coordinator node completes the scaled-out transaction

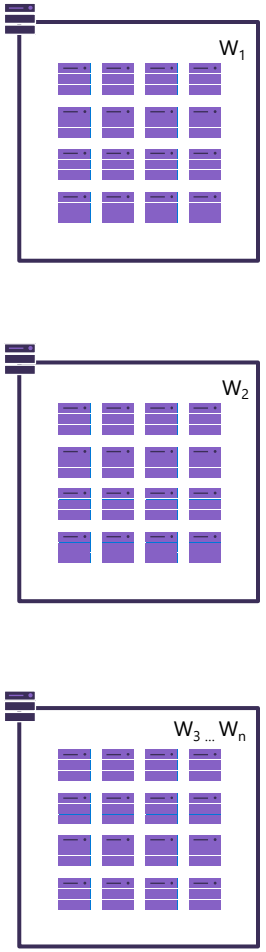
APPLICATION

```
BEGIN;
UPDATE campaigns
   SET feedback 'relevance'
 WHERE company_type 'platinum'
UPDATE ads
   SET feedback 'relevance'
 WHERE company_type 'platinum'
COMMIT;
```





WORKER NODES



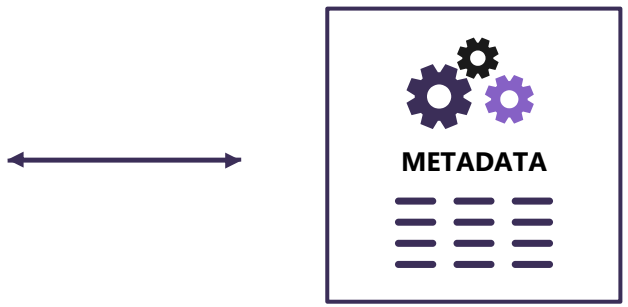
Co-located join

It's logical to place shards containing related rows of related tables together on the same nodes

Join queries between related rows can reduce the amount of data sent over the network

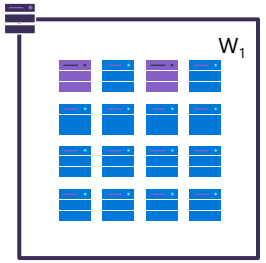
- APPLICATION

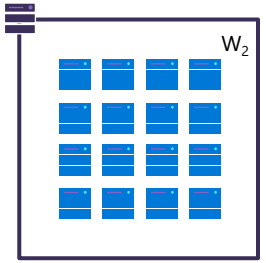
```
SELECT  count(*)
  FROM  ads JOIN campaigns ON
        ads.company_id = campaigns.company_id
 WHERE  ads.designer_name = 'Isaac'
   AND  campaigns.company_id = 'Elly Co'
```

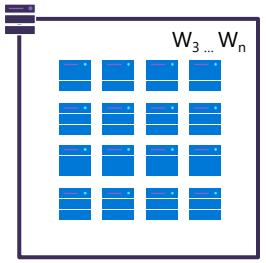


COORDINATOR NODE

WORKER NODES







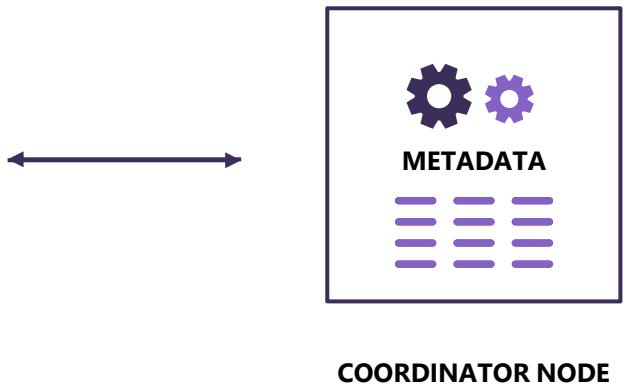
Schema change

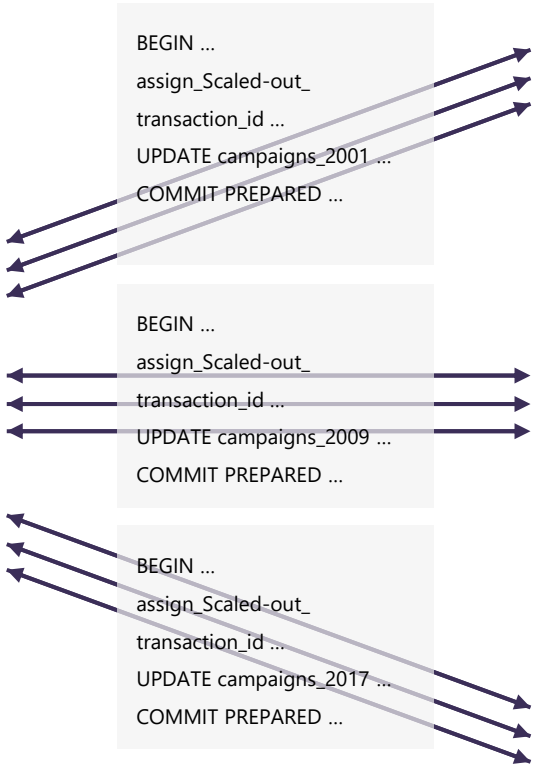
Schema can be updated when types of tables and scale out strategy change

Prepare source tables for migration and add scale out keys

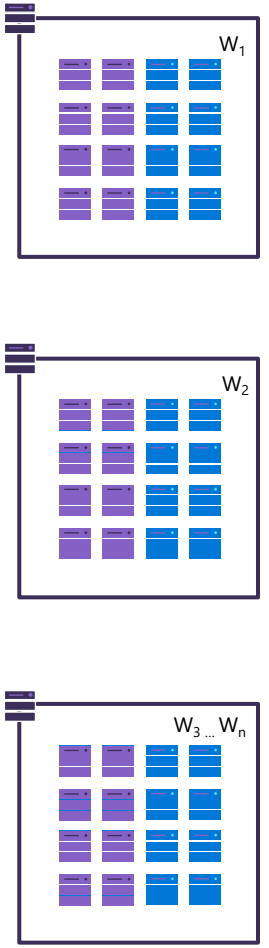
APPLICATION

```
-- Schema Change
ALTER TABLE campaigns
ADD COLUMN company_type text
```





WORKER NODES



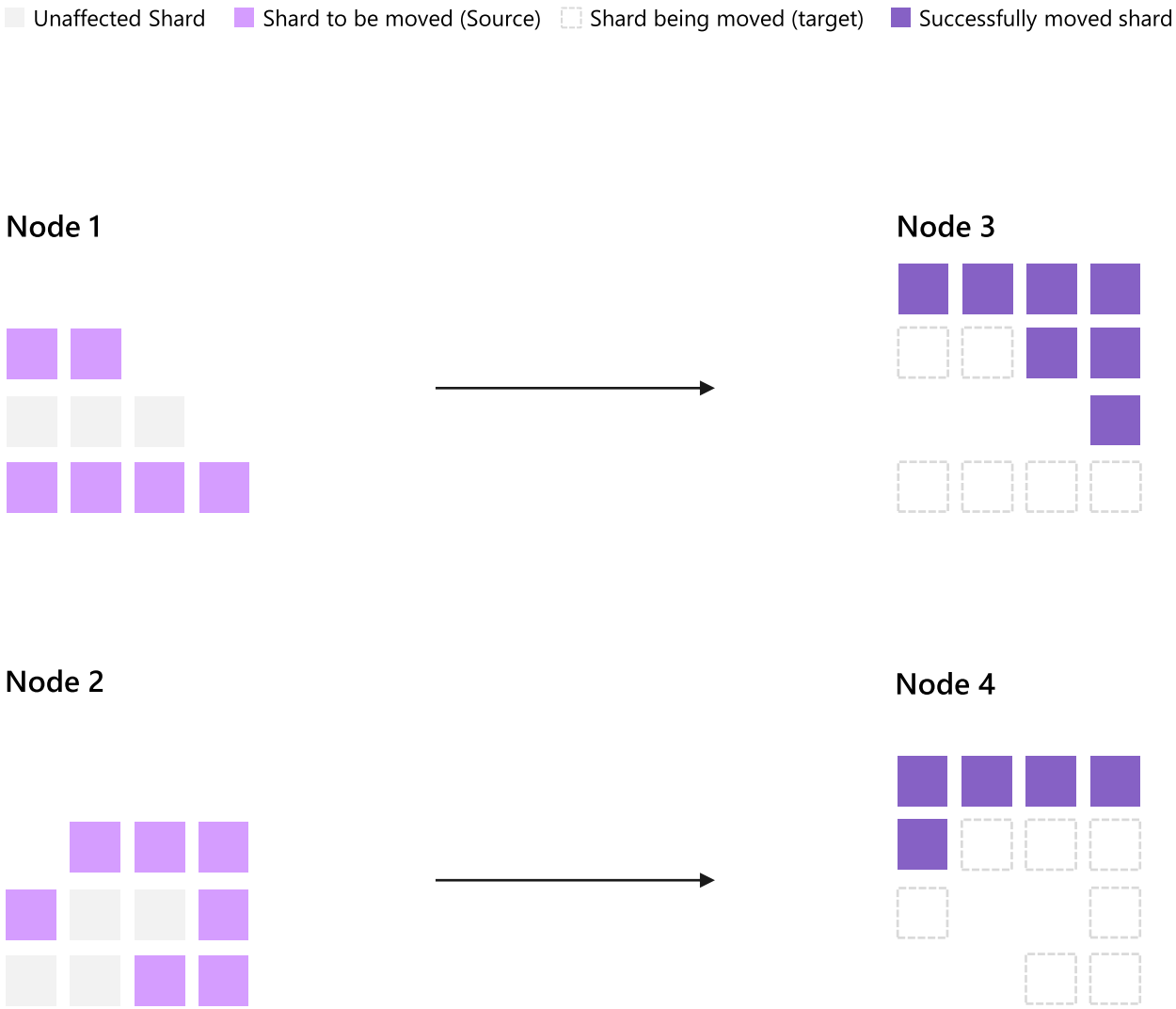
Effectively manage data scale out

Shard rebalancer redistributes shards across old and new worker nodes for balanced data scale out **without any downtime**.

Shard rebalancer will recommend rebalance when shards can be placed more evenly

For more control, use tenant isolation to easily allocate dedicated to specific tenants with greater needs

Shard Rebalancer



End to End Customer Journey

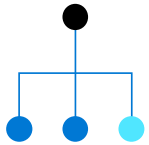
- Technical Qualification
  - Questions/Answers to qualify right workloads to Hyperscale
- Data Modelling
  - Choosing distribution column
  - Table classification
- Performance Considerations
  - Tips/Tricks

- Migration

Microsoft Azure Solutions Business

Technical Qualification

Key uses cases for Hyperscale (Citus)



Multi-tenant & SaaS applications

Scale beyond single node

Minimize hotspots by spreading out tenants

Rebalance data fully online

Isolate large tenants to their own hardware

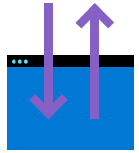
Real-time operational analytics

Ingest terabytes of data per day

Enable sub-second query responses

Parallelize across nodes for 100x performance

Simplify complex ETL processes



Transactional/OLTP applications

Ensure high performance with concurrent users

Avoid single points of failure

Distribute transaction processes across multiple nodes

Manage high volumes of transactions

Common Patterns

- Data Size
  - > 100 GB and expecting to scale further
  - < 100 GB expecting tremendous scale in the near future
    - Ex: Acquiring large customer(s) - 6 months would be a 1TB

- Workload
  - Operational System of record
  - How real-time / Freshness – immediate, 1 min, 30 min, 1hr
  - Read latency – Human real-time (ex: dashboard)
  - Concurrency – High concurrency (100s to 1000s of users)

- Common Sources
  - Single Node Postgres (On-premise, other clouds)
  - Consolidate Relational + NoSQL
  - In-memory databases
  - Oracle

One or more patterns of each bucket = good candidate for Hyperscale

Right Questions?

- Is there a natural dimension to your data?
- What are your queries based on?
- What is the workload / expected latency?
- Is there going to be skew?

General Sharding Patterns

- Tenant
- Entity
- Geography

Tenant Sharding – When?

- If you are B2B or SaaS (i.e. your workload is per-tenant)
- If your tenant distribution is relatively uniform. (i.e. your largest tenant is not 50% of your data)
- If your application is relational i.e. has joins, primary keys, foreign keys, transactions.
- Analytics with small data per customer (OR) can be pre-computed
- Pick Customer_id or tenant_id

Tenant Sharding – Why?

- **Can co-locate related tables together as most tables are related to tenant.**
  - **"All the data from all tables for a given tenant lives on the same worker node"**
- **Can rely on PostgreSQL to execute all queries. Get full SQL coverage per tenant.**
- **Maintain relational parts of the application.**
  - Primary Keys, Joins, Foreign Keys
  - No need of de-normalization
- **Can keep scaling by adding more memory.**

Entity Sharding – When?

- Data grows at a very large pace – Ex: billions of records per day
- Run analytics on that data quickly – Parallelism required across workers.
- Or want to do quick lookups on granular keys while maintaining relational properties – ex IoT applications
- Ex: user_id, device_id, event_id

Entity Sharding – Why?

- You need parallelism to analyze billions of records in sub-second times.
- Can do quick joins based on shard key.
- Can linearly scale out performance with number of cores.
- Data distribution is uniform.

3 Table Types

- Distributed Tables
- Reference Tables
- Local Tables

Distributed Tables

Definition:

- Tables that are sharded.

Classification:

- Large tables (>10GB) – shard on same key (may require addition of shard key)
- All tables are be co-located
- Enables localized and fast joins on workers
- Ex: transactions, events etc

```
SELECT create_distributed_table(table_name, column_name);
```

Reference Tables

Definition:

- Replicated to all the nodes (extra latency)

Classification:

- Small tables < 10GB
- Efficient joins with distributed tables
- Cannot have sharding dimension
- Ex: countries, categories

```
SELECT create_reference_table(table_name);
```

Local Tables

- Plain Postgres tables on the coordinator node.
- Admin Tables that don’t interact with main tables
- Separate micro-service that doesn’t need sharding

Data Modeling Example - Shopify

Stores table
- **Id**
- **Name**

Products table
- **Id**
- **Name**
- **Price**
- **Store id**
- **Description**
- **country_id**

Countries
- Id
- Email
- Name

Purchases table
- Id
- Product id
- Items
- Total

Migrations
- Id
- Migration name
- SQL details

Data Modeling Example - Shopify

Stores table
- **Id**
- **Name**

Products table
- **Id**
- **Name**
- **Price**
- **Store id**
- **Description**
- **country_id**

Countries
- Id
- Email
- Name

Purchases table
- Id
- Product id
- Items
- Total

Migrations
- Id
- Migration name
- SQL details

Distributed
Reference
Local

Proposed Data Model

Table name	Table type	Shard Key in place	Backfilling needed?
Stores	Distributed	Yes	No
Products	Distributed	Yes	No
Countries	Reference	N/A	N/A
Purchases	Distributed	No	Yes
Migrations	Local	N/A	N/A

Main Take-aways

- Choose the right shard key.
  - Customer: For B2B / SaaS, it is a good candidate. Mostly no-brainer for OLTP apps.
  - User / Device: For large datasets requiring parallelism, it is a good candidate.
- Make sure to colocate your tables.
  - In Multi-Tenant systems, shard on top of hierarchy, so that you can maintain colocation.
  - In analytics use cases requiring parallelism, shard tables same way to maintain colocation.
  - If required, add and backfill the column onto the table to co-locate.
- Make your smaller tables reference.

Nouveautés Cosmos DB - Azure Managed Instance for Apache Cassandra

On-premises Apache Cassandra workloads face some common challenges

Difficulty scaling to meet demand

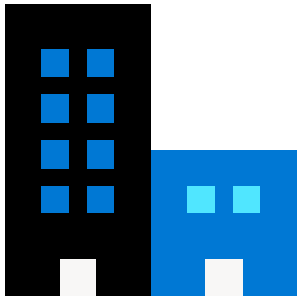
Compatibility issues when adding nodes

Tedious maintenance requirements

Complex licensing structures

Options for hosting Cassandra workloads are typically limited

Organization are usually forced to choose between on-prem and PaaS environments

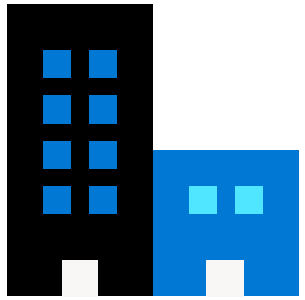


Self hosted Cassandra
On -prem

PaaS

You need more flexible hosting options for your Cassandra workloads

Hybrid environments provide organizations the flexibility they need



Self hosted Cassandra
On -prem

Managed Cassandra
IaaS

PaaS

What if you could easily add performance and flexibility to Apache Cassandra workloads without having to modernize to PaaS?

Limitless scaling

Managed environment

Hybrid deployment

Integrated tools

Apache Cassandra…. The cool bits

- **The backbone of Facebook (where it started) and Netflix!**
- **Short learning curve for app-dev – CQL <= SQL**
- **Distributed, linear scale, write optimized, fast reads**
- Resilience and fault tolerance – multi-master, no SPOF
- Highly Configurable at the platform level (consistency + replication)

Apache Cassandra…. The not-so cool bits

- **Complex to maintain – Cosmos DB is much easier**
- **Lack of complex query engine – Cosmos DB SQL API is richer**
- **Multi-master only - no single region master, multi read region option – Cosmos navigates RPO/RTO better**
- Resilience and fault tolerance – too flexible, data loss?
- Highly Configurable at the platform level – 300 + combinations on consistency. Need a PhD?

https://db-engines.com/en/ranking

Database popularity…

Select a ranking

- Complete ranking
- Relational DBMS
- Key-value stores
- Document stores
- Time Series DBMS
- Graph DBMS
- Search engines
- Object oriented DBMS
- RDF stores
- Wide column stores
- Multivalue DBMS
- Native XML DBMS
- Spatial DBMS
- Event Stores
- Content stores
- Navigational DBMS

Special reports

- Ranking by database model
- Open source vs. commercial

Featured Products

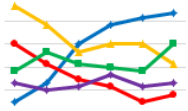
Ranking > Complete Ranking

RSS RSS Feed

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.



trend chart

410 systems in ranking, March 2023

Rank Mar 2023	Rank Feb 2023	Rank Mar 2022	DBMS	Database Model	Score Mar 2023	Score Feb 2023	Score Mar 2022
1.	1.	1.	Oracle	Relational, Multi-model	1261.29	+13.77	+9.97
2.	2.	2.	MySQL	Relational, Multi-model	1182.79	-12.66	-15.45
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	922.01	-7.08	-11.77
4.	4.	4.	PostgreSQL	Relational, Multi-model	613.83	-2.67	-3.10
5.	5.	5.	MongoDB	Document, Multi-model	458.78	+6.02	-26.88
6.	6.	6.	Redis	Key-value, Multi-model	172.45	-1.39	-4.31
7.	7.	7.	IBM Db2	Relational, Multi-model	142.92	-0.04	-19.22
8.	8.	8.	Elasticsearch	Search engine, Multi-model	139.07	+0.47	-20.88
9.	9.	↑ 10.	SQLite	Relational	133.82	+1.15	+1.64
10.	10.	↓ 9.	Microsoft Access	Relational	132.06	+1.03	-3.37
11.	↑ 12.	↑ 14.	Snowflake	Relational	114.40	-1.26	+28.17
12.	↓ 11.	↓ 11.	Cassandra	Wide column	113.79	-2.43	-8.35
13.	13.	↓ 12.	MariaDB	Relational, Multi-model	96.84	+0.03	-11.47
14.	14.	↓ 13.	Splunk	Search engine	87.97	+0.89	-7.39
15.	15.	↑ 16.	Amazon DynamoDB	Multi-model	80.77	+1.08	-1.03
16.	16.	↓ 15.	Microsoft Azure SQL Database	Relational, Multi-model	77.44	-1.31	-7.23
17.	17.	17.	Hive	Relational	70.91	-1.21	-10.31
18.	18.	18.	Teradata	Relational, Multi-model	63.74	+0.71	-5.11
19.	19.		Databricks	Multi-model	60.86	+0.52	
20.	20.	↓ 19.	Neo4j	Graph	53.51	-1.92	-6.16
21.	↑ 22.	↑ 24.	Google BigQuery	Relational	53.44	+0.99	+6.78
22.	↓ 21.	22.	FileMaker	Relational	51.15	-1.66	-1.81
23.	23.	↓ 21.	SAP HANA	Relational, Multi-model	50.84	+1.17	-5.17

NB: wide-column stores Vs columnar databases

Wide-column or column family databases

ID	Product Name	Product Code	Product Category
1	Product1	123	Books
2	Product2	250	Equipment
3	Product3	380	Equipment

1	Product1	123	Books	2	Product2	250	Equipment	3	Product3	380	Equipment

Disk- Logical Representation

Columnar databases

ID	Product Name	Product Code	Product Category
1	Product1	123	Books
2	Product2	250	Equipment
3	Product3	380	Equipment

1	2	3	Product1	Product2	Product3	123	250	380	Books	Equipment	Equipment

Disk- Logical Representation

Introducing Azure Managed Instance for Apache Cassandra

Build and migrate Cassandra workloads to cloud with flexible deployment, elastic scaling, and robust security options

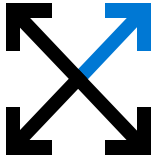
Managed and secure

Streamline operations with VNet Injection, Azure Active Directory integration and automated patches, updates and back-ups

Built in hybrid flexibility and control

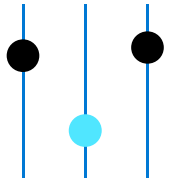
Retain flexibility and control of hardware configurations with hybrid deployment options and simple, instance-based pricing

Limitless and elastic scaling



Meet your business demands with one-click scalability and zero downtime

Fully compatible Cassandra



Keep your development interruption-free by using your existing skills and familiar Cassandra tools, drivers and SDK's

TL;DR… it’s SQL MI, but for Apache Cassandra ☺

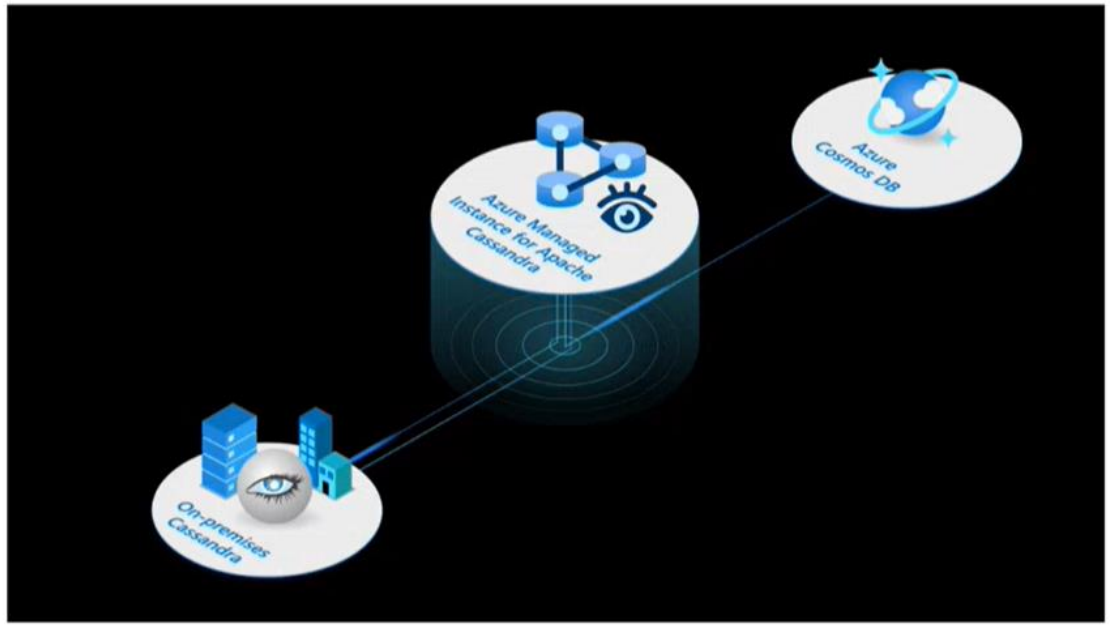
What is Microsoft Azure Managed Instance for Apache Cassandra?

- The simplest way to get an Apache Cassandra datacenter up and running in Azure and connected to your existing cluster.
- On ramp to Azure Cosmos DB Cassandra API
- Reduce the burden of managing the chores of running a cluster (patching, availability, etc.) while getting off-the-shelf Apache Cassandra with most of the control you have with self-hosting with less of the chores.

What is Microsoft Azure Managed Instance for Apache Cassandra?

- Managed Cassandra Instances
  - New Azure service offering
  - Running open source Apache Cassandra software
  - Linux VM – instance based pricing
  - Hosted on Azure compute
  - Premium managed disks for storage
  - Automated operations
  - Single tenant
- Provides Cassandra “Database-as-a-Service”
  - 100% compatibility
  - 100% performance profile
  - Minimal operations overhead

Azure Managed Instance for Apache Cassandra - GA



Fully Managed
- On-demand node provisioning, 3.xx+ Cassandra versions, 100% compat
- Connectivity with high availability, disaster recovery, automated backups
- Azure Monitor, enhanced portal experience, all regions
- Enterprise ready security (VNET injection, BYOK, Audit Logging, LDAP)

Seamless online migration to cloud or hybrid
- Can join existing unmanaged Apache Cassandra clusters on-premises or VMs
- Dual-write proxy for heterogeneous Cassandra migrations

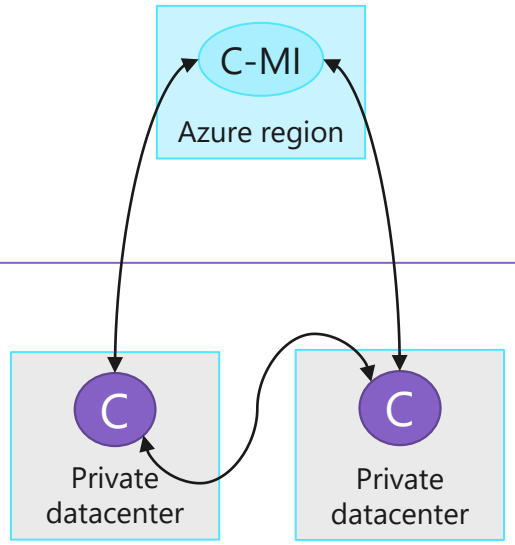
Flexibility & control, from hybrid to serverless
- Single management experience for hybrid on-premise and cloud deployments
- Superior elasticity with Cosmos DB Cassandra API
- Robust security and compliance from Azure
- Easy and transparent instance-based pricing

Psst, remember… it’s SQL MI, but for Apache Cassandra ☺

Journey of Cassandra Customer to Azure Cloud

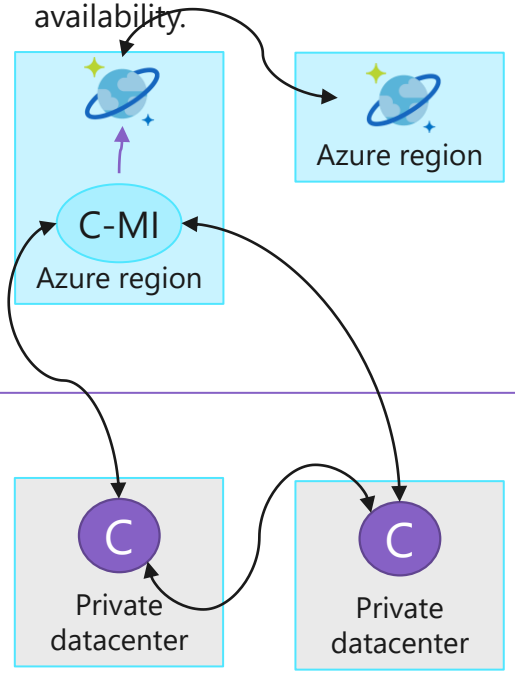
Phase 1

Bring your Cassandra workload to Azure via native Cassandra replication



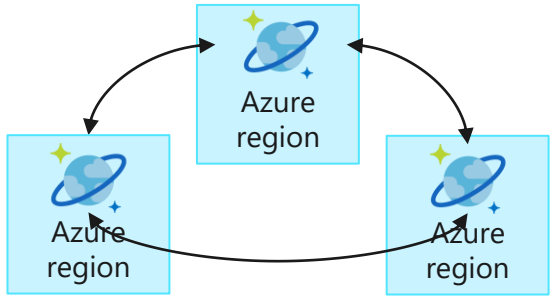
Phase 2

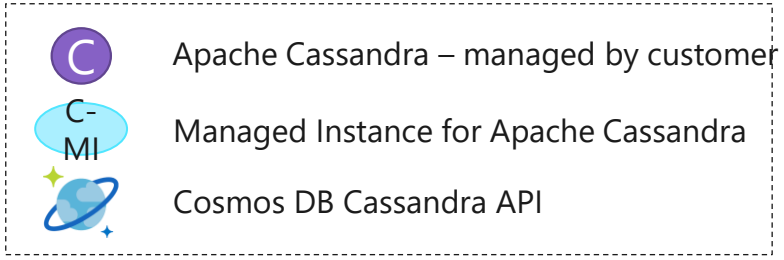
Extend your Cassandra workload with Cassandra API to lower TCO and increased availability.



Phase 3

Extend your Cassandra workload with Cassandra API to lower TCO and increased availability.

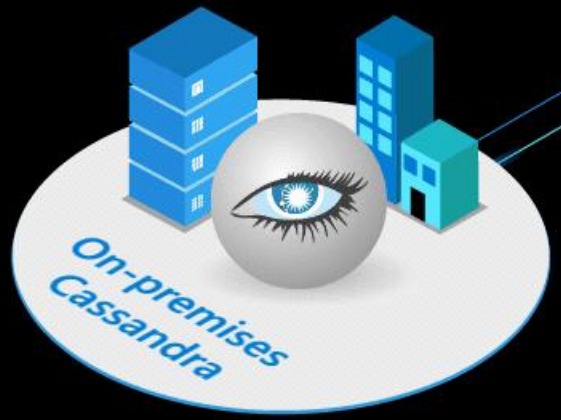




Path to the cloud

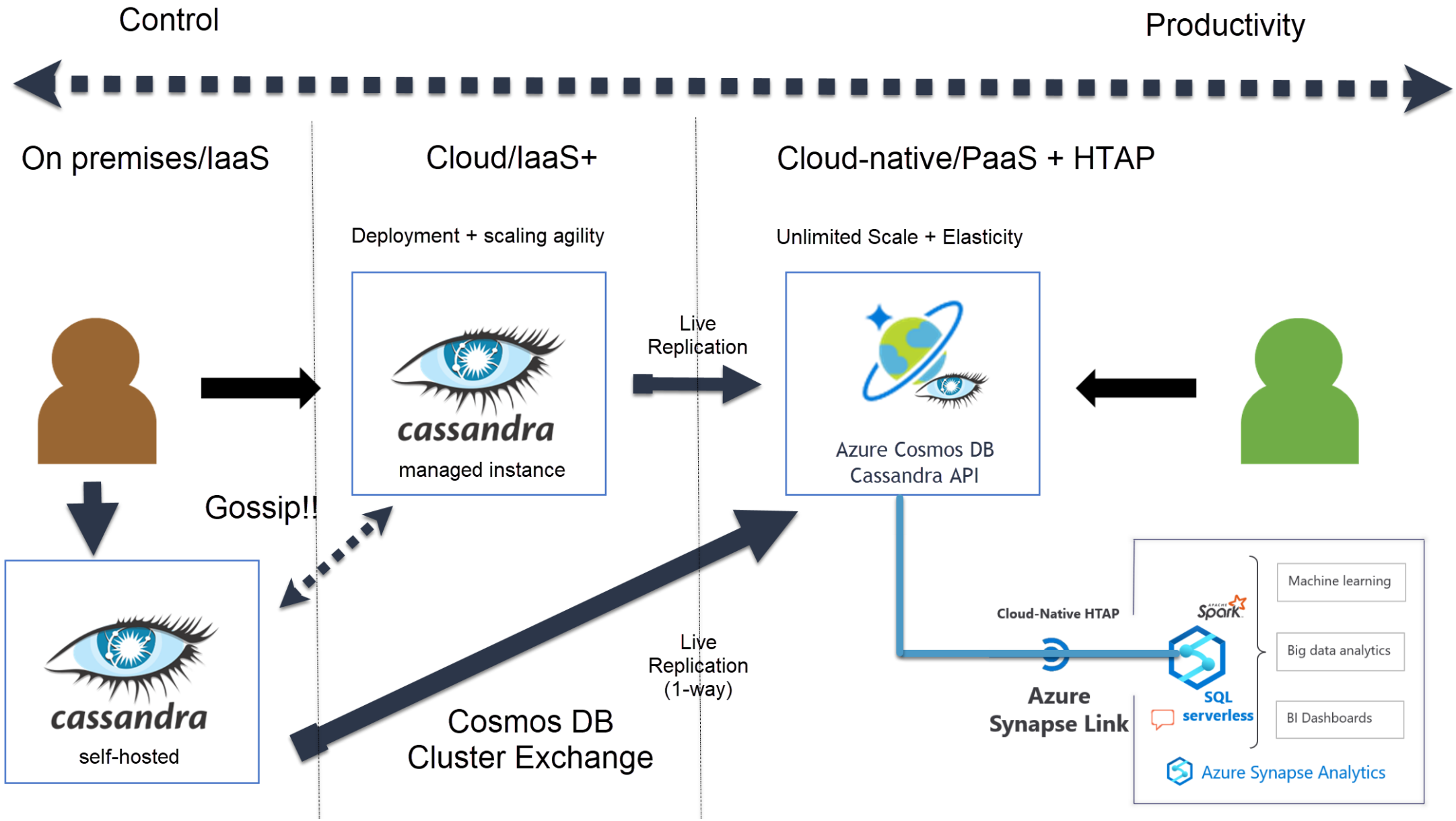
Simple migration to **Azure Managed Instance for Apache Cassandra** frees you from database and infrastructure management with automated deployment and scaling options.

Full modernization to **Azure Cosmos DB Cassandra API** provides speed, flexibility, and high availability at any scale.



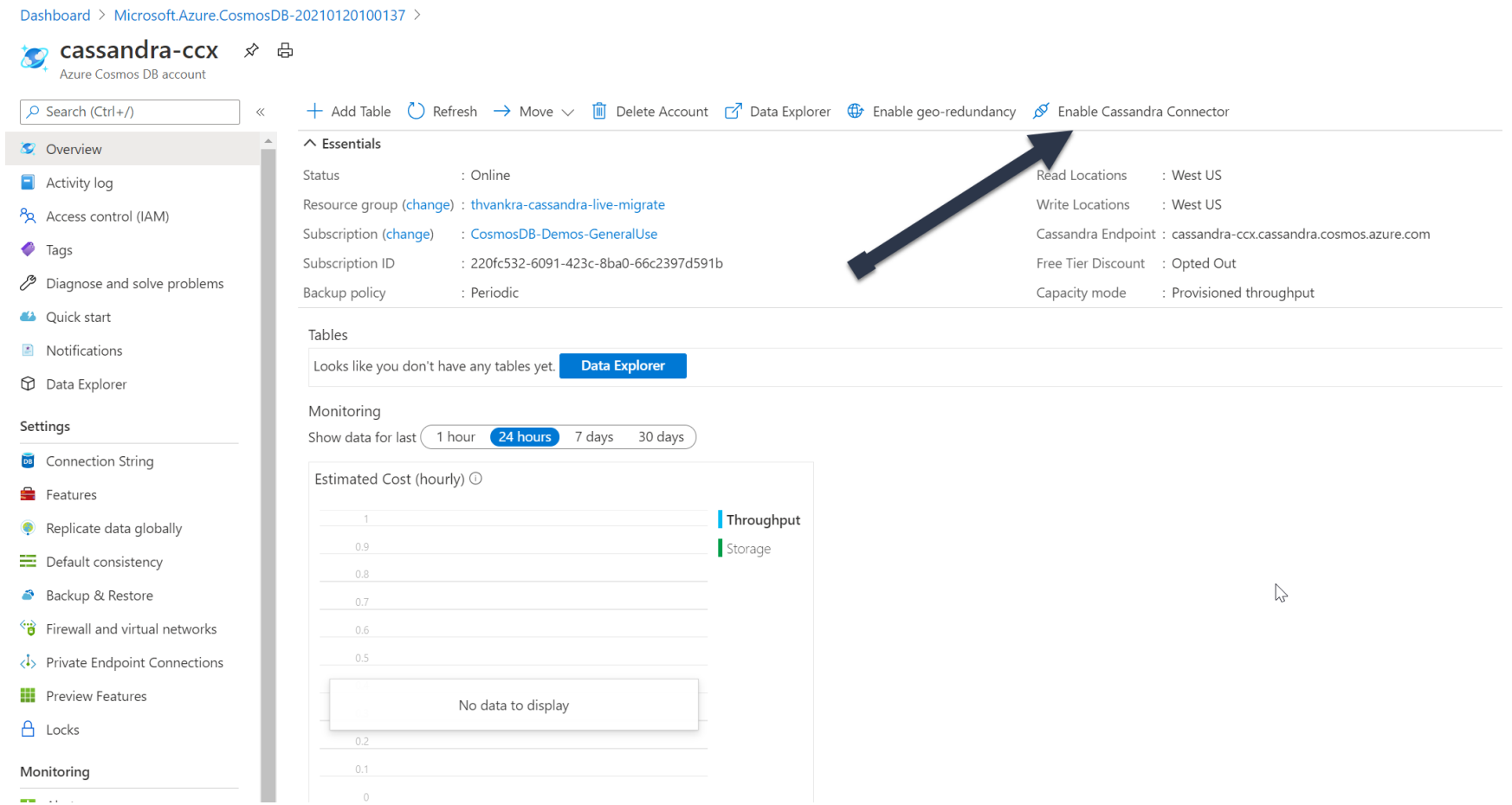
Self-hosting allows for highly customized Apache Cassandra deployment but is limited in terms of scalability, manageability and compatibility.

A continuum of choices



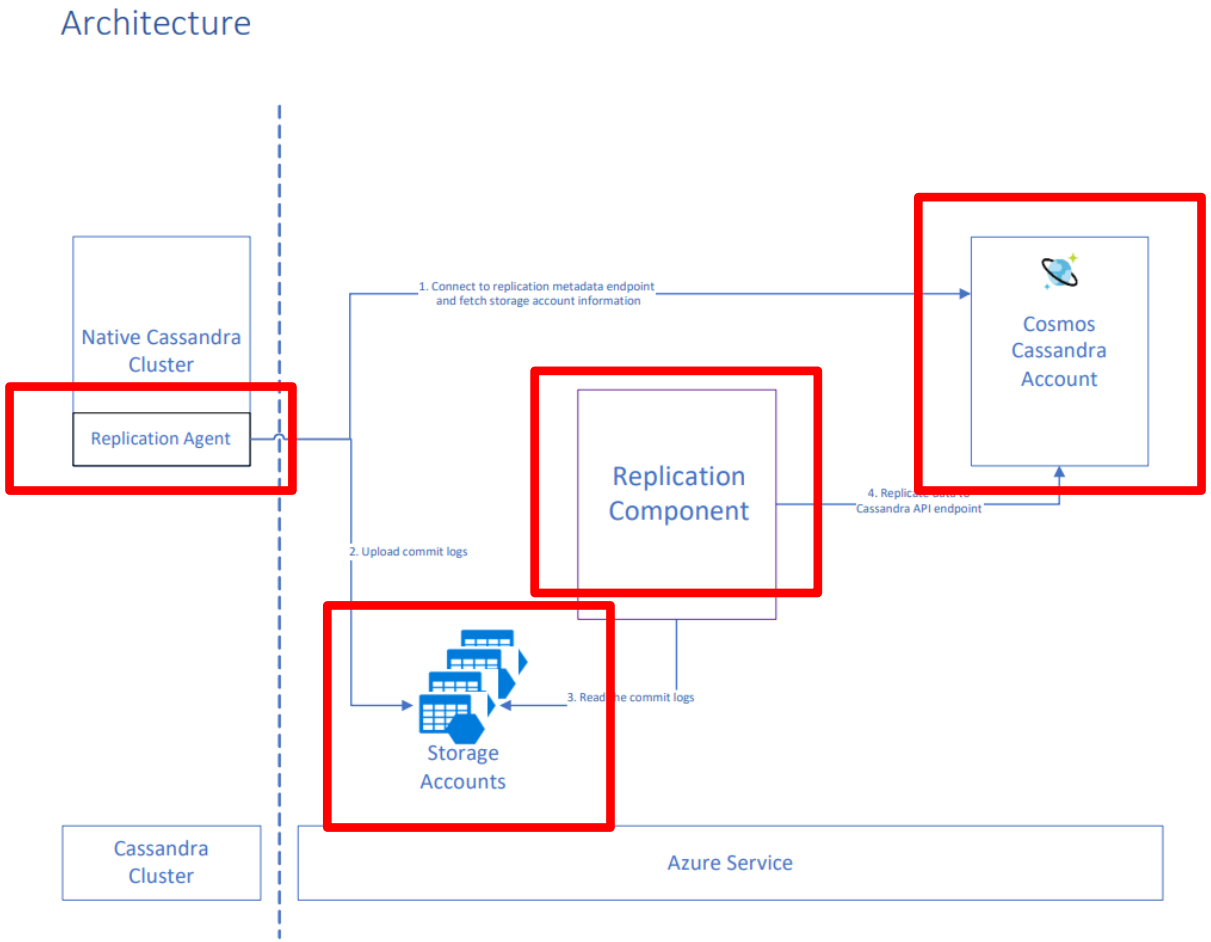
What is Cosmos DB Cluster Exchange?

- Get the $ value of VMs converted Cosmos DB RUs – no double cost of migration.
- Access to a replication agent to move data in real-time



What is Cosmos DB Cluster Exchange?

- Get the $ value of VMs converted Cosmos DB RUs – no double cost of migration.
- Access to a replication agent to move data in real-time (private preview).



How to choose? Options with Cassandra offerings

Control → Productivity

(Self-hosting)

- You have a highly customized Apache Cassandra deployment with custom patches/snitches/etc.
- You have an existing Cassandra SMEs who can deploy, configure, and maintain your clusters

Azure Managed Instance for Apache Cassandra

- You have a standard Apache Cassandra deployment (no custom code)
- Need hybrid connectivity in production to on prem
- You want to lower your operational overhead for your Apache Cassandra nodes
- You want to avoid paying for expensive licenses for additional management software with time-based lock in
- You have Cassandra SMEs you can configure your cluster

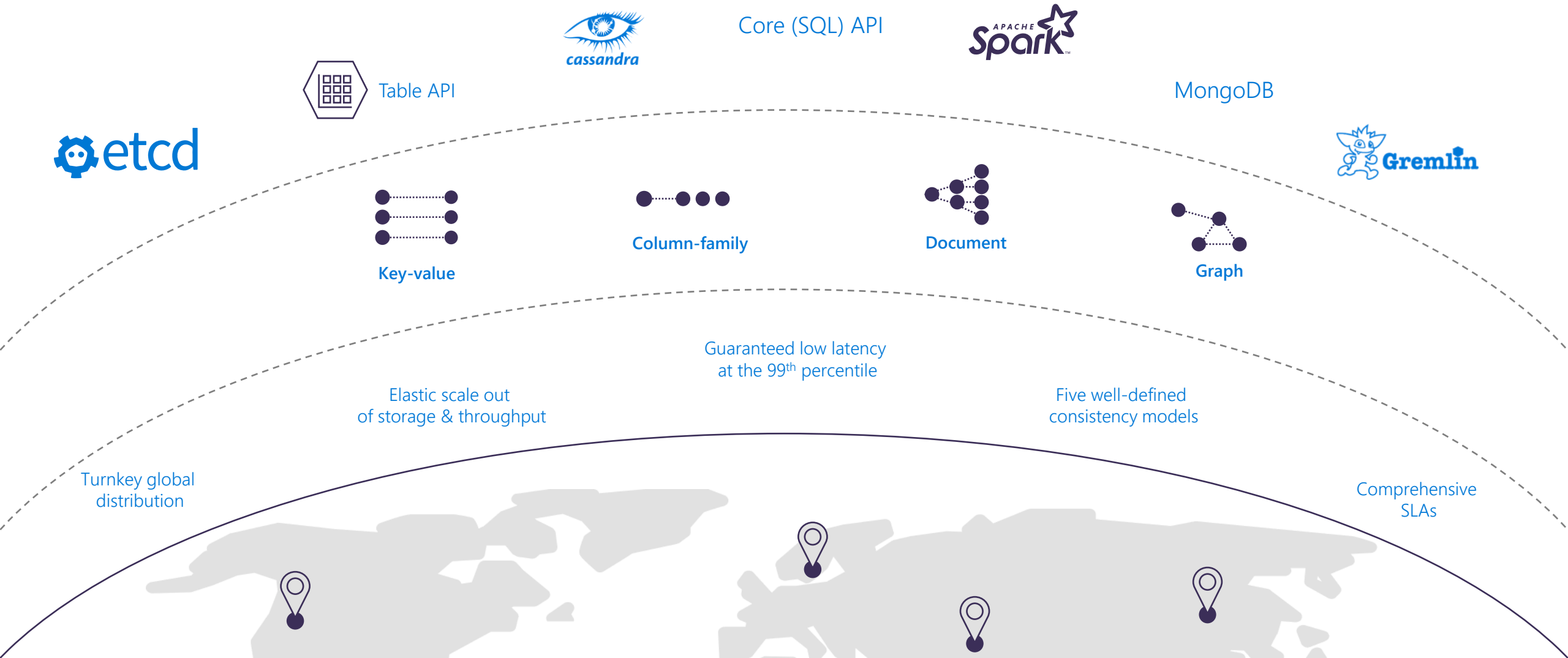
Azure Cosmos DB

- You want lots of elasticity options
- You want cloud native platform features like PITR or Synapse integration
- You don't have any compatibility issues with the Cassandra API
  - Cosmos team is constantly improving compatibility, but there are some gaps
- Read heavy workloads are especially performant/efficient
- No Cassandra SMEs necessary to operate/configure

Microsoft Azure Solutions Business

Cosmos DB - Architecture & Developer Guide

What is…AZURE COSMOS DB?



Azure Cosmos DB

Fast NoSQL database with open APIs for any scale

Azure Cosmos DB is a fully managed NoSQL database for modern app development with SLA-backed speed and availability, automatic and instant scalability, and open source APIs for MongoDB, Cassandra, and other NoSQL engines.

Guaranteed speed at any scale

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

Simplified application development

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

Mission-critical ready

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

Fully managed and cost-effective

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

Top use cases for Azure Cosmos DB

Use Case	Use Case Details	Why Azure Cosmos DB?	Alternatives	Optimization Tips	Customer Examples
Real-time telemetry	**1. IoT** • Logistics (track assets/inventory) • Smart cars (telematics) • Smart Cities • Smart Buildings 2. **User/Customer** • Track digital actions (clicks; app events)	✓ Elastic, **unlimited throughput** (scale) ✓ Can easily **ingest large volumes** of data from write-heavy scenarios ✓ **Real-time queries** backed by <10 ms low-latency SLA ✓ **Auto-indexing** of semi-structured data	• Time-consuming and complex connection, integration, and management bundle of OSS tech (e.g., Redis, Cassandra, etc) • Maintain performance by actively managing, over-allocating, and over-paying for cluster	1. Use Device ID or User ID as partition key 2. Customize index policy (vs auto-index all data) 3. Materialize View design pattern to optimize aggregates 4. Use local region for read/write data in SDK connection policy	• Teijen • Mercedes Benz • Johnson Controls • Exxon Mobil • SitePro
Real-time recommendations and 360-degree customer view	Rapid response recommendations to enhance user experience (e.g., Netflix recommendations; e-commerce product suggestions)	✓ **Spark integration** offers ML and big data processing ✓ **<10 ms low-latency SLA**	• Build and maintain custom solutions; manage data integration (POS, CRM, etc.) • Static, non-responsive, or delayed recommendations based on stale data • Limited personalization (e.g., missing location, daypart, etc.)	1. Custom configuration for Spark connector (e.g., batch size) 2. Scale up RU/s for batch processing then immediately scale down for steady-state 3. Use Product ID or User ID as partition key	• ASOS • Siemens Healthineers • PureFacts
Mission-critical or geo-distributed applications	Deliver consistent real-time, always-on user experience anywhere in the world • Shopping cart/checkout • IoT device registry	✓ One-click, fully-managed **data replication/distribution** around the word ✓ **<10 ms low-latency** SLA ✓ **99.999% high-availability** SLA ✓ Available in **all Azure regions**	• Set up and maintain infrastructure, networks, and databases globally • Manage data replication manually • Time, complexity, and cost to scale capacity and reach	1. Use Session or Bounded Staleness pre-configured data consistency 2. Use local region for read/write data in SDK connection policy 3. Use multi-master writes/replication in production (not dev/test)	• Walmart/Jet.com • Citrix • Chipotle Mexican Grill
NoSQL migration	Move from self-managed NoSQL (e.g., MongoDB, Cassandra) to PaaS solution and stop managing clusters.	✓ **Full PaaS offering** that handles all database/cluster management ✓ **APIs** for MongoDB, Cassandra, and other NoSQL enable migration with minimal code changes	• Create, manage, and tune clusters • Manage upgrades, patching, etc • 24-7 internal team for monitoring, site reliability, and performance optimization	1. Use Azure Data Migration Service to migrate to API for MongoDB 2. Use Azure Data Factory to migrate to SQL (core) API 3. Use Striim (or other preferred partner) for any migration	• Bentley Systems • Symantec (Norton)

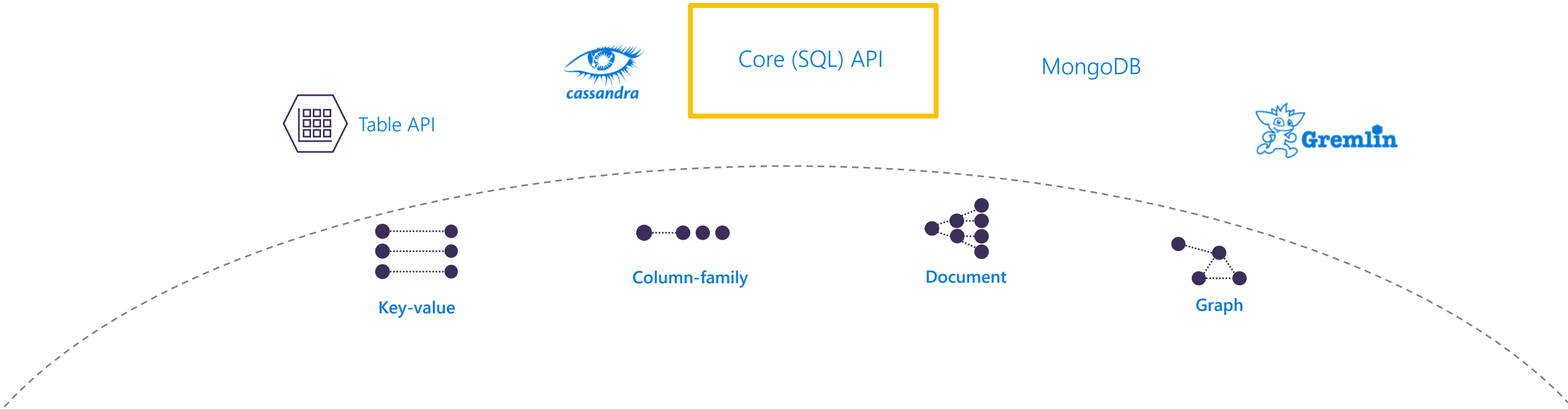
When should the customer use Azure Cosmos DB?

One or more of these requirements…

- **Flexible schema**
  - **Sensor data may change over time** ✓
- **Scalability**
  - Especially for writes
  - **Our use case needs high #writes/sec** ✓
    - Cosmos guarantees throughput for any workload
- **Low latency (fast)**
  - **Cosmos guarantees <10ms at P99 for reads, writes of 1kb** ✓
    - Can use as fast cache for key/val lookups
- **High availability**
  - 99.999% HA with 2+ regions ✓

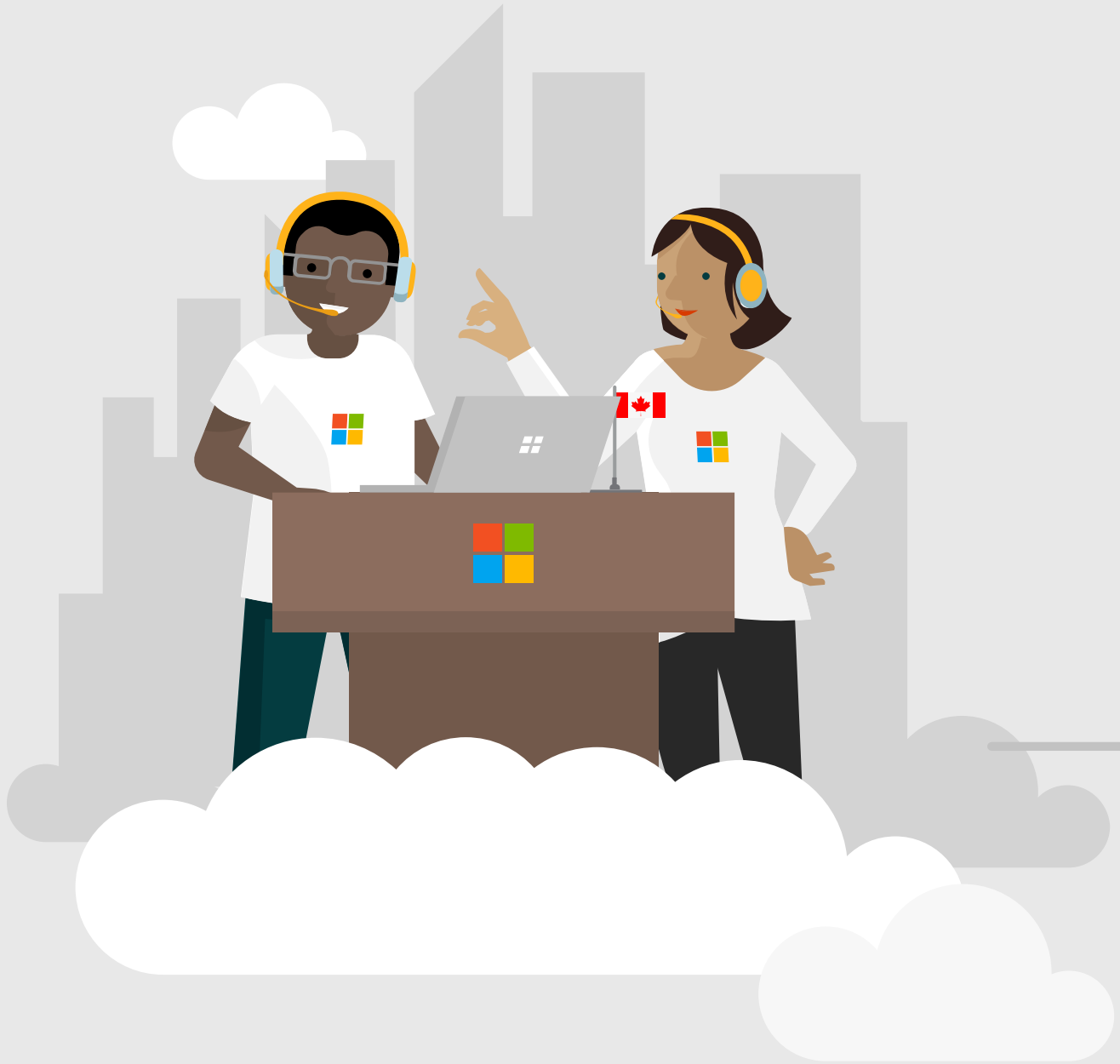
Which API should the customer use for their app?

- **For all new workloads – Core (SQL) API**
  - Best developer experience – Cosmos builds interface, service, and SDKs
  - Gremlin API for graph data
- **Other APIs for migration or DB platform preference/requirement**

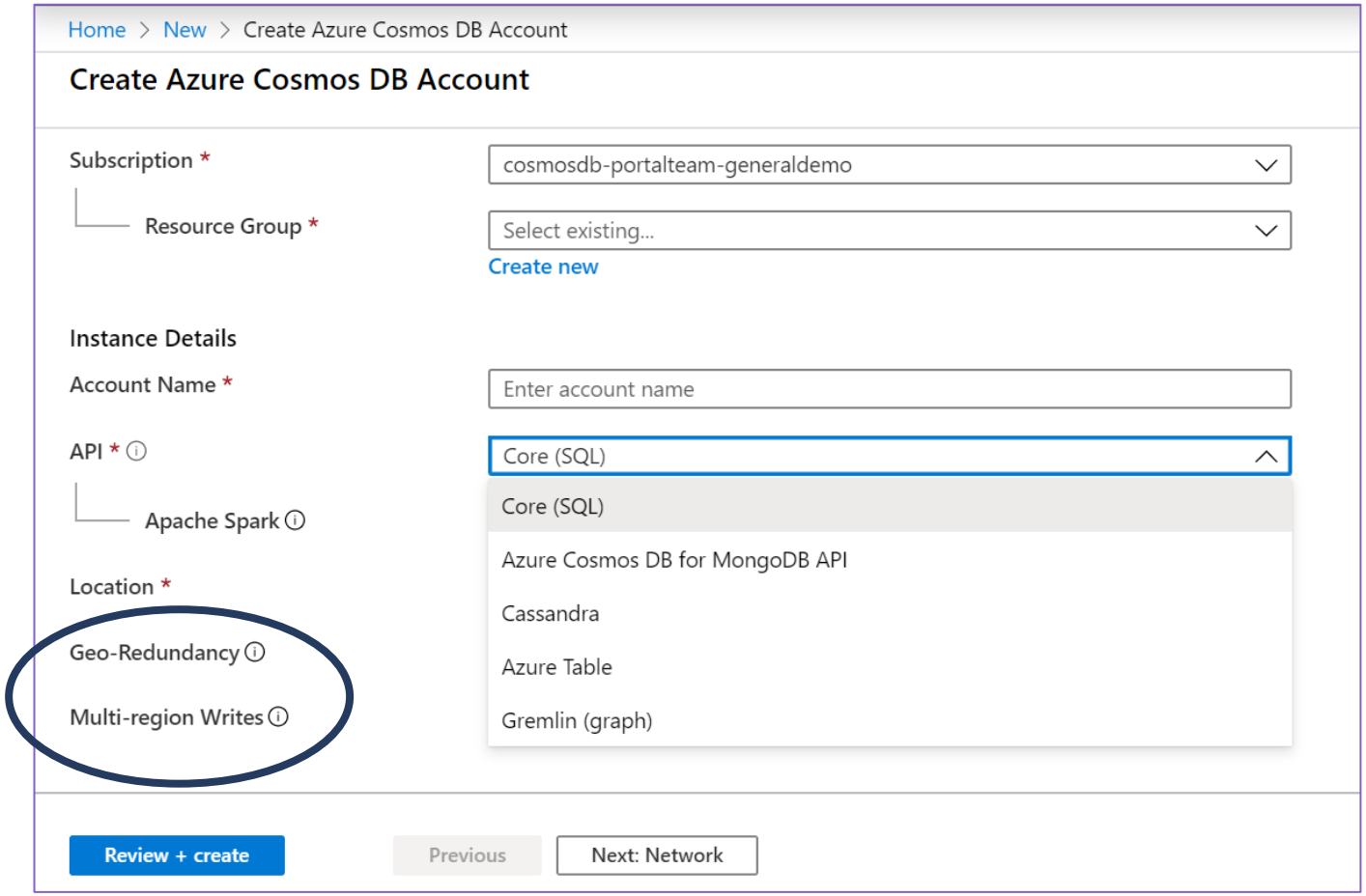


Microsoft

Let’s get started and create CosmosDB

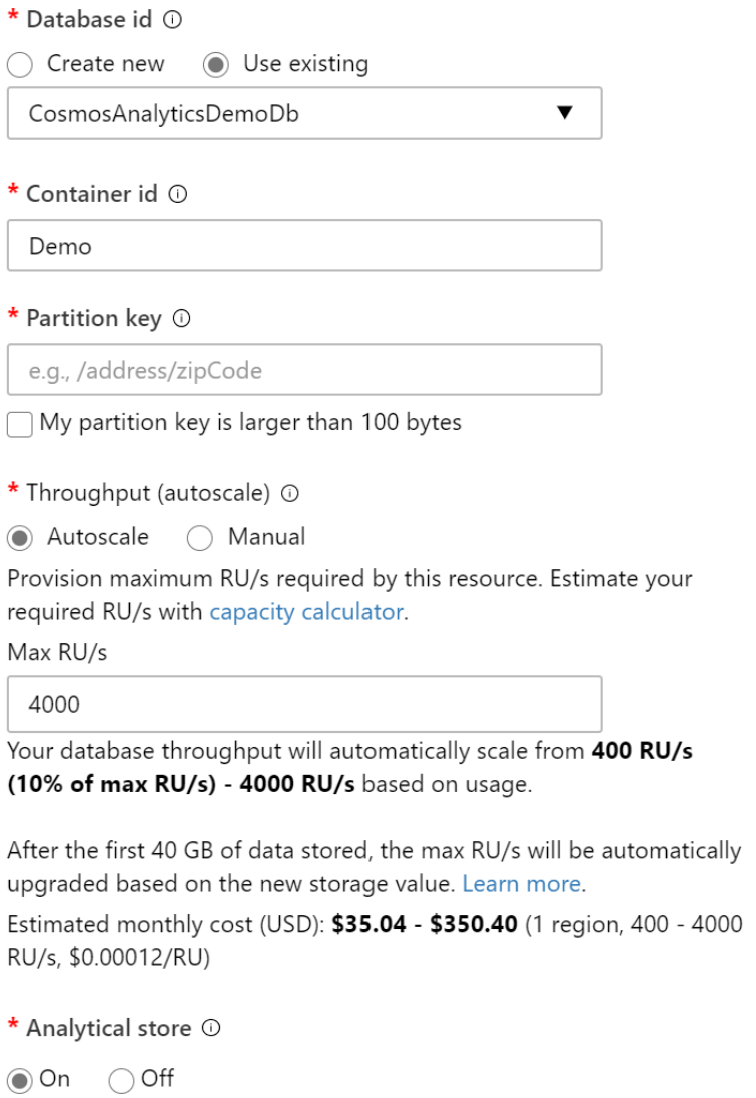


Creating a new account



- Geo-redundancy – starts out with paired 2 regions
- Single-region (N) writes – $0.008 for 100 RUs/hour *N.
- Multi-region (N) writes – $0.016 for 100 RUs/hour *N.
- So 400 RU/s, with 2 regions and multi-master = $93/month
- For dev/test, customer doesn't need these features and can go with single region

Creating a new container



- It's important to think about following before creating a container:

- Request Units
- Partitioning
- Data modelling

- Learning these 3 topics well -> better likelihood of success

What are Request Units (RUs)?

- **In Cosmos DB, you provision the expected capacity/performance**
- **Expressed in Request Units per second (RU/s)**
  - Represents the "cost" of a request in terms of CPU, memory and I/O
- **Performance can be provisioned:**
  - at the database-level
  - at the collection-level
  - or both
- **You can change RU/s programmatically with API calls or…**
- **NEW: Autoscale - to scale based on usage**

What are Request Units (RUs)?

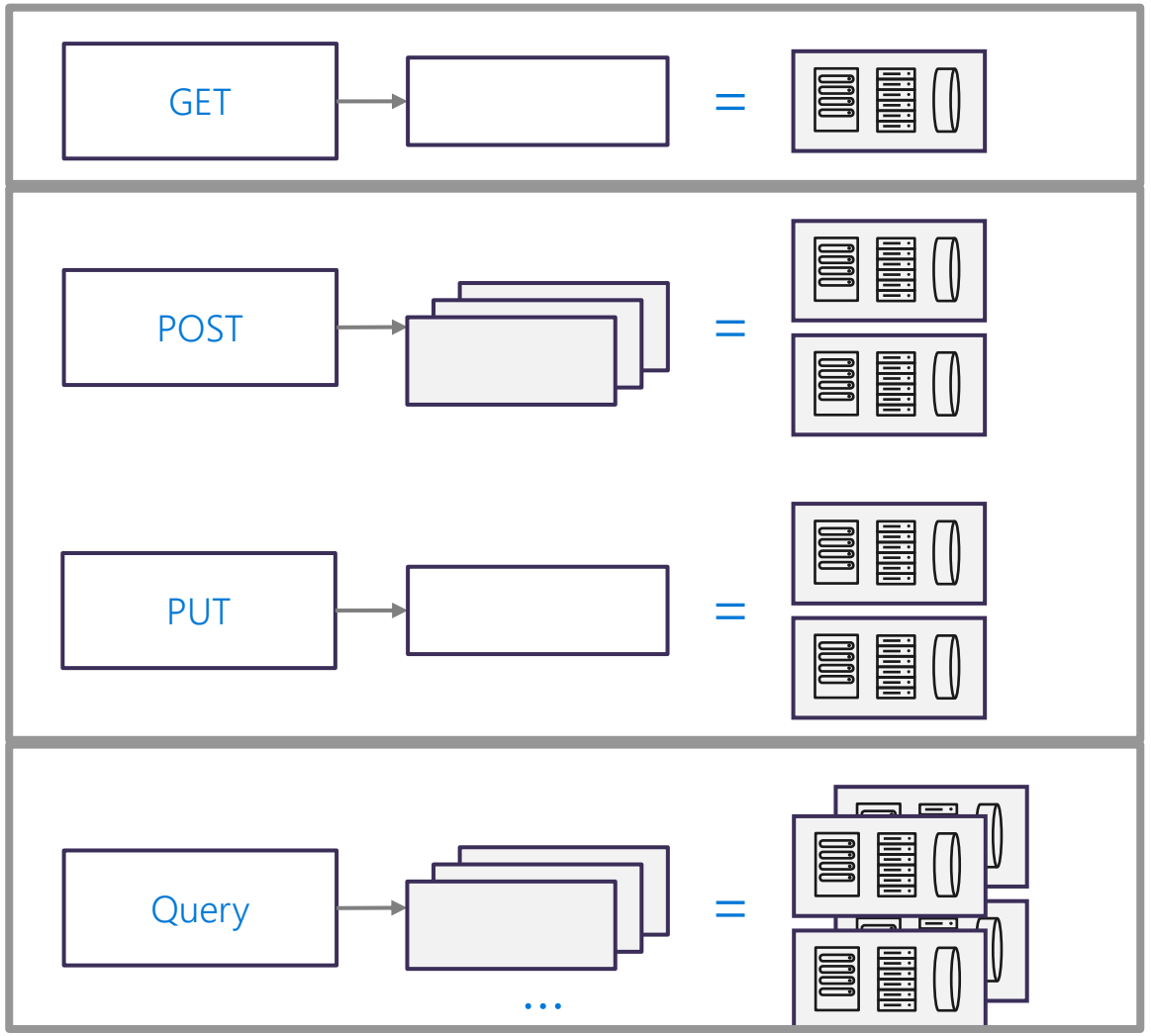
Each request consumes # of RU

1 RU = 1 read of 1 KB document

5 RU = 1 write of a 1KB document

Query: Depends on query & documents involved

(Indexing also affects RU cost)



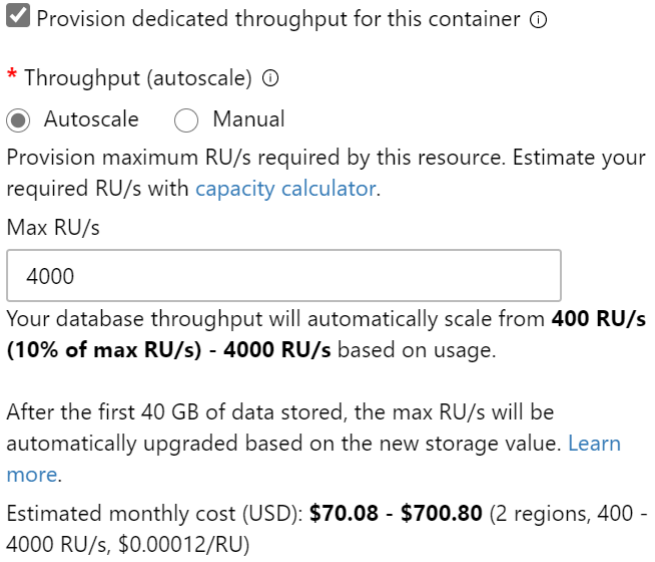
Basic example: Telemetry ingest…let's estimate the RU/s we need

- Contoso Logistics has 1000 vehicles
- Each vehicle sends 1 document (write) every second
- Each document consumes ? RUs
- Total RU/s = $\frac{1000 \text{ writes}}{\text{sec}} * \frac{?\ \text{RU}}{\text{write}}$
- So how many RUs does our document write consume?

Autoscale vs manual provisioning

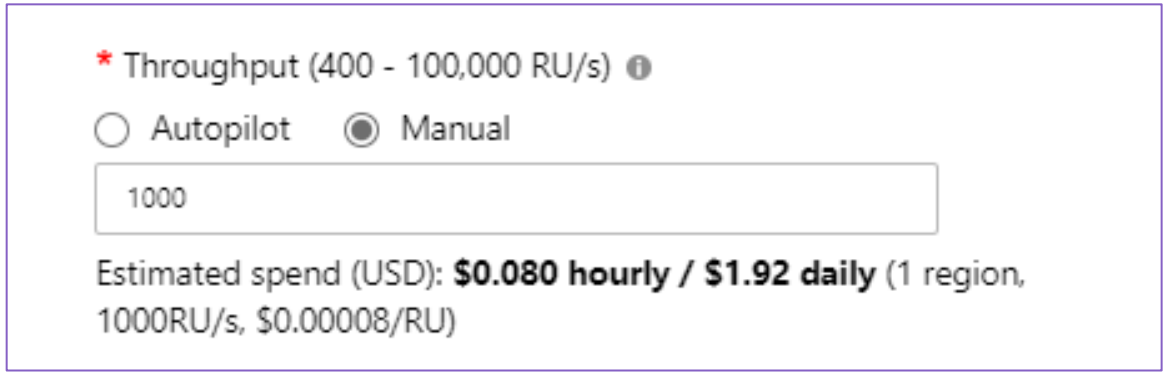
Autoscale

- Good for bursty, unpredictable workloads or periods of idle time
- Scale between 10% and Max



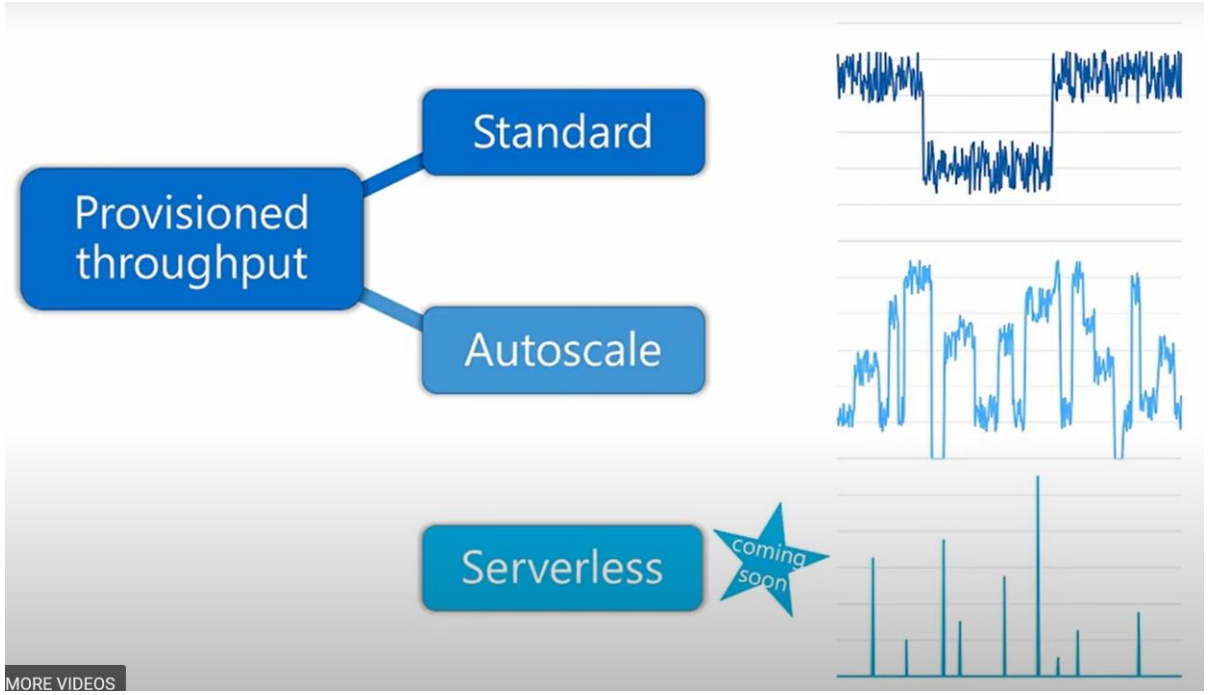
Manual

- Good for predictable workloads
- Change RU/s programmatically with Azure function



CosmosDB Serverless

As a consumption-based option, serverless eliminates the concept of provisioned throughput and instead **charges you for the RUs your database operations consume**.



	Provisioned throughput	Serverless
Maximum number of Azure regions per account	unlimited	1
Maximum throughput burstability per container	unlimited	5,000 RU/s
Maximum storage per container	unlimited	50 GB

Request Units (RU)	Price
1,000,000 serverless request units (RU)	$0.25

- In provisioned throughput mode a container with 500 RU/s monthly cost of: $0.008 * 5 * 730 = **$29.20**
- In serverless mode, you would only pay for the consumed RUs (20M RUs a month): $0.25 * 20 = **$5.00**

Understanding minimum RU/s

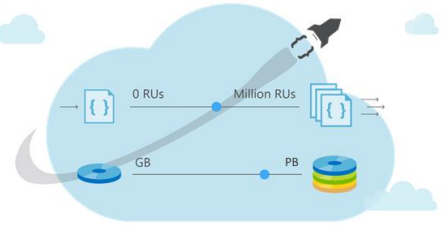
Minimum RU/s:

```
MAX(
        400
        Max Throughput provisioned / 100
        Current storage * 10 RU/GB
)
```

Examples:

- 1 TB of storage = 1000 GB * 10 RU/GB = 10,000 RU/s minimum
- Scale up to 100,000 RU/s -> 1000 RU/s minimum

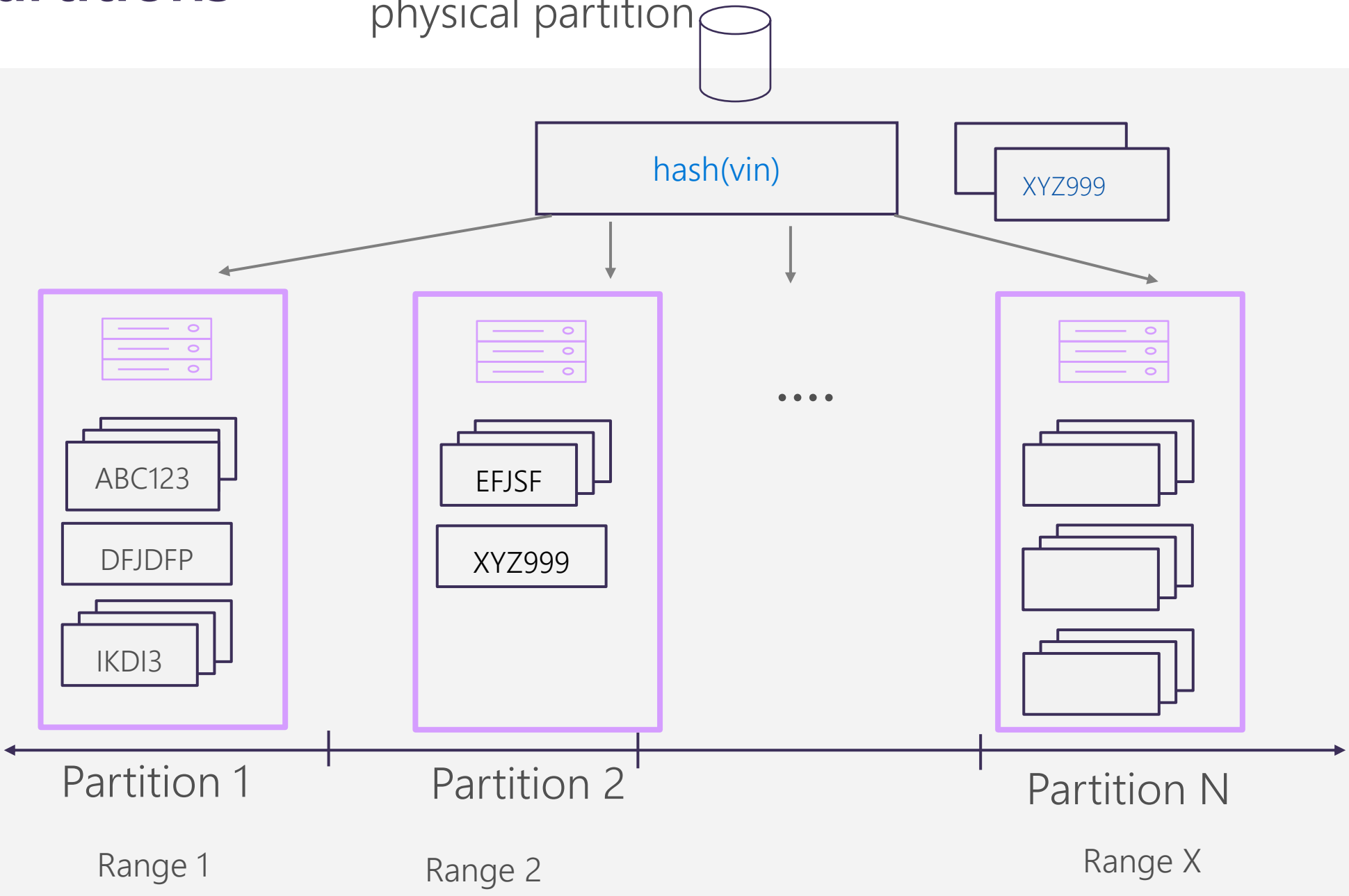
Partitioning



- Why does Cosmos DB ask for a partition key?
- Instead of having 1 larger and larger machine (scaling up), we distribute data among multiple machines
- Each machine is responsible for subset of data
- Analogy: Distributing workload across a team

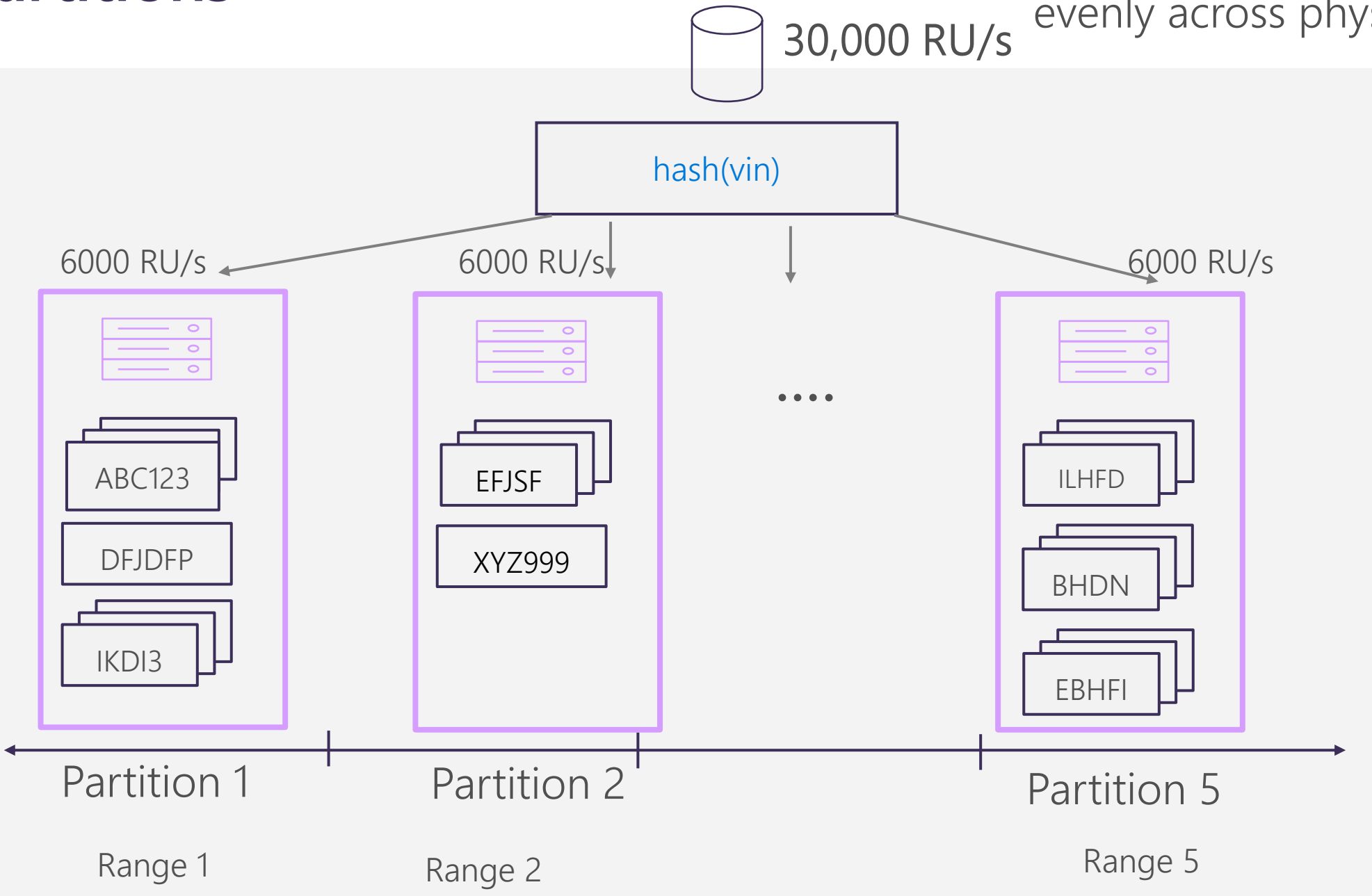
Partitions

Documents with same partition key value (vin) are in same logical, physical partition

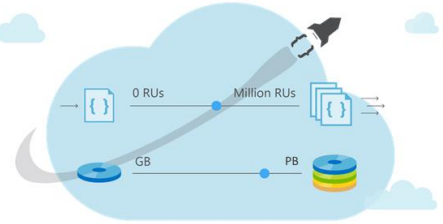


Partitions

Cosmos DB distributes RU/s evenly across physical partitions



Partitioning best practices



- **Write heavy workloads**
  - Want high cardinality of values

- **Read heavy workloads:**
  - You can do key/val lookups with id, partition key)
  - Most of your queries include partition key in the filter

Understanding RUs and physical partitions

- Each physical partition is 50GB in size
- Each logical partition key is 20GB max
- Each physical partition can have 10,000 RU/s max
- Number of Physical Partitions depends on RU Throughput and Data size and provisioned/adjusted automatically

Data modelling

Cosmos DB != relational database

Write out the data access patterns….

Then choose the partition key and data model ☺

Syntetic Partition Key: “telemetry” collection

- We use a property “/partitionKey”
- Syntetic Partition key: vin_year_month
- Why?
  - High cardinality for high write ingest
  - Won’t hit 20GB logical partition key value
  - Can still query on VIN:
    - SELECT * FROM c WHERE c.partitionKey IN (…)
- Optimized for write-heavy scenario


```
{
    "partitionKey": "PB2GAMT1UBQC0N2BW-2019-10",
    "entityType": "VehicleTelemetry",
    "ttl": 5184000,
    "tripId": null,
    "vin": "PB2GAMT1UBQC0N2BW",
    "state": "MN",
    "region": null,
    "odometer": 27929.01,
    …
    "refrigerationUnitKw": 69.99,
    "refrigerationUnitTemp": 17.49,
    "timestamp": "2019-10-20T04:46:52.6877746Z",
    "id": "9c5cb3b6-d47c-4625-b925-460e23a0a0f1"
}
```

Trip entity


```
[{
"id": "e2c7ca5e-2ef1-4ac2-836d-74124811516c",
"partitionKey": "DB2JXBWBLCL09FJFU",
"entityType": "Trip",
"vin": "DB2JXBWBLCL09FJFU",
"consignmentId": "1f145aad-d943-44c7-b0f3-1bc8562c7628",
"plannedTripDistance": 190.53,
"location": "OR",
…
"status": "Pending",
"timestamp": "0001-01-01T00:00:00",
"packages": [
      {
      "packageId": "514edbdc-210b-45f8-b05f-d491d6ded7e4",
      "storageTemperature": 22,
      "highValue": false
      },
……
      {
      "packageId": "26128844-cc3f-4808-99f7-2b142ad002e5",
      "storageTemperature": 32,
      "highValue": false
      }
],
"consignment": {
      "consignmentId": "1f145aad-d943-44c7-b0f3-
      1bc8562c7628",
      "customer": "Bellows College",
      "deliveryDueDate": "2019-10-20T05:56:44.7367547Z"
}]
```


Package entity


```
[{
"id": "ad476b94-8b3a-4ab8-ba5f-b95ad75acd63",
“packageId": " 26128844-cc3f-4808-99f7-2b142ad002e5",
"partitionKey": " 26128844-cc3f-4808-99f7-2b142ad002e5",
"entityType": "Package",
"tripId": "ed96feb9-957e-4d30-873e-7a4f12266022",
"consignmentId": "cb56399c-65b3-452b-89f0-d3b9bed5fa02",
"height": 33.37,
"length": 41.45,
"width": 20.96,
"grossWeight": 223.72,
"storageTemperature": 39,
"highValue": false,
"trip": {
"tripId": "ed96feb9-957e-4d30-873e-7a4f12266022",
"vin": "AESQX3ZKCFM25HNNE",
"plannedTripDistance": 137.09
},
"consignment": {
"consignmentId": "cb56399c-65b3-452b-89f0-d3b9bed5fa02",
"customer": "Lamna Healthcare Company",
"deliveryDueDate": "2019-10-20T08:26:44.7366267Z"
},
"timestamp": "0001-01-01T00:00:00",
}]
```


- Select package info makes it easier to generate Trip summary info view

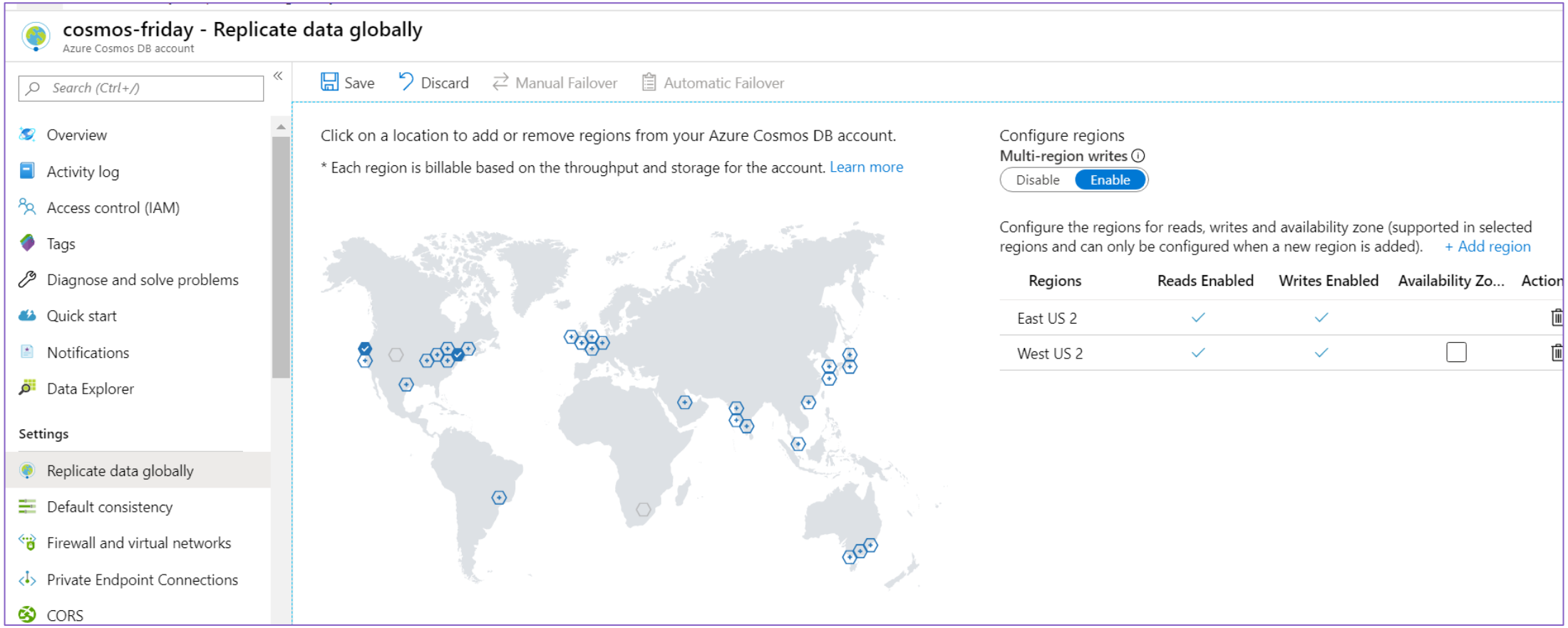
Production readiness

Production readiness checklist

- **Add a second region for 99.999% High Availability**
- **Single region: 99.99% availability** (52.6 min / year)
- **2+ regions: Always 99.999% read availability**
  - Single-region write – 99.99% write availability
  - Multi-region write – 99.999% write availability (5.26min / year)

Production readiness checklist

- Add a second region for 99.999% High Availability



Summary

Setting up the service

- Choose the right API
- Request Units
- Design data model & partition key

Development

- SDK client-side optimizations
- Benchmark performance

Testing

- Local Emulator
- CI/CD pipeline with Cosmos DB

Pre-prod checklist

- Ensure setup for high availability
- Use “ApplicationRegion” (or PreferredLocations) for multi-region deployments.
- Configure logging, monitoring and alerting on key metrics

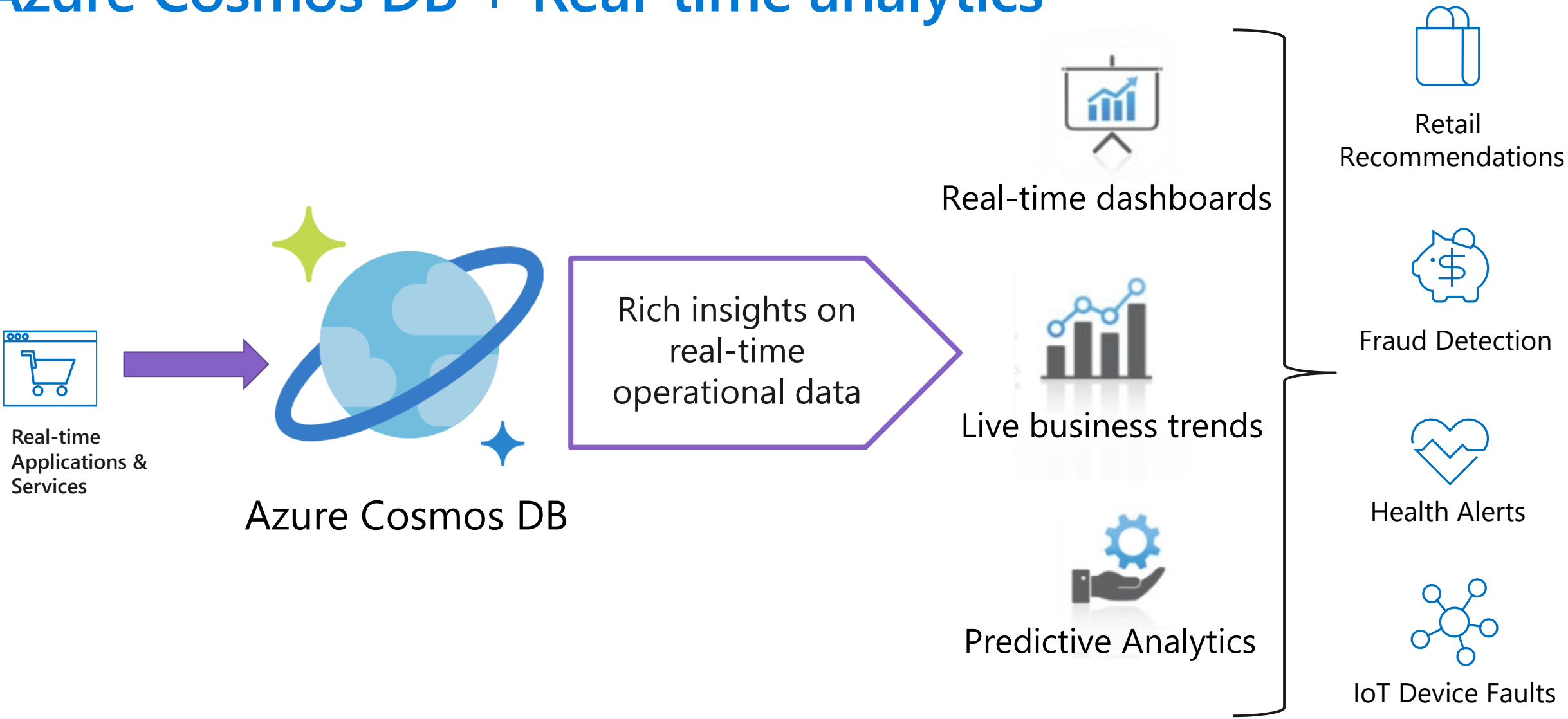
Monitoring in production

- How to use Metrics
- Find expensive operations with Diagnostic Logs

Microsoft Azure Solutions Business

Cosmos DB - Analytical Store

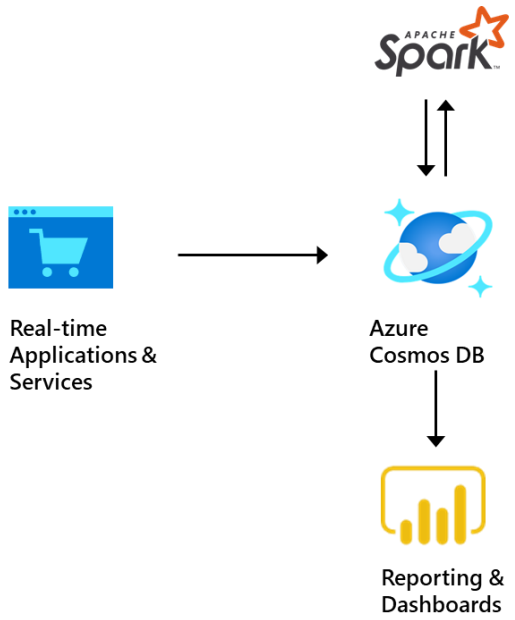
Azure Cosmos DB + Real-time analytics



Transform your business with actionable insights

Azure Cosmos DB analytics : Options

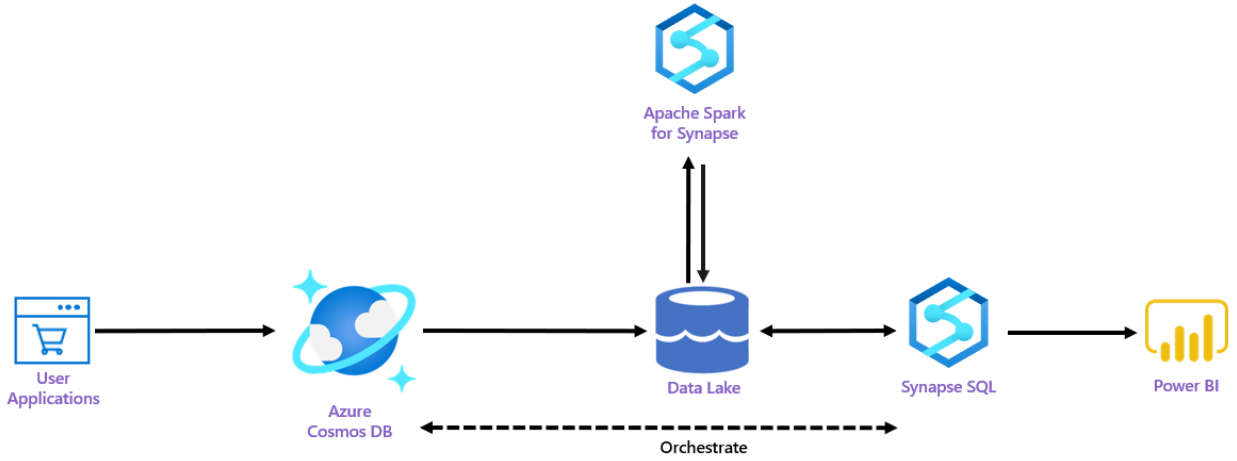
Run OLTP & OLAP workloads on the same database



HUGE performance impact on the OLTP workloads at scale

As volume of operational data increases, latency of analytical queries increases and more resource intensive

Separating OLTP & OLAP



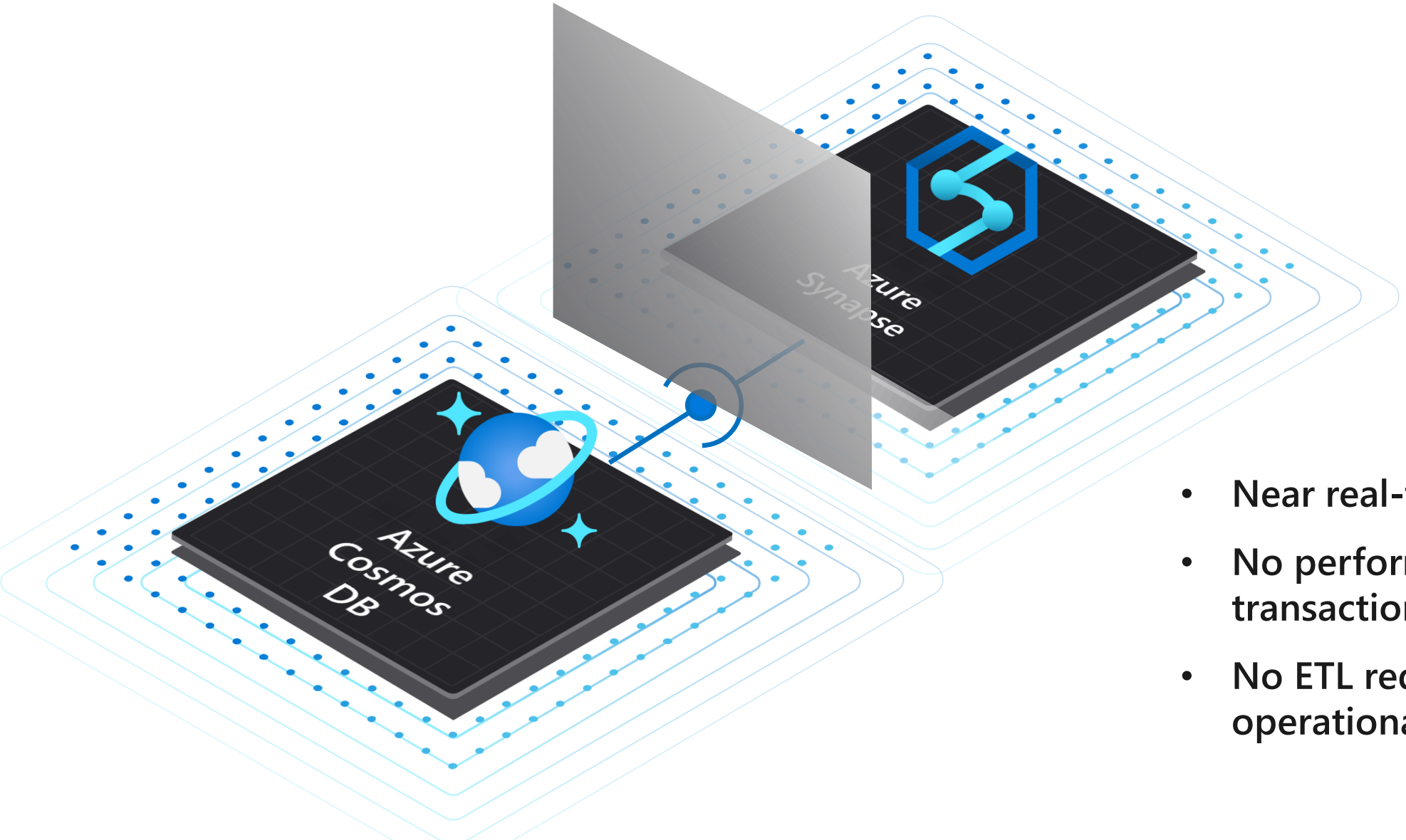
Ingest data periodically from Azure Cosmos DB to Data Lake

Delayed insights on data

Manage data formats and storage layer for analytics

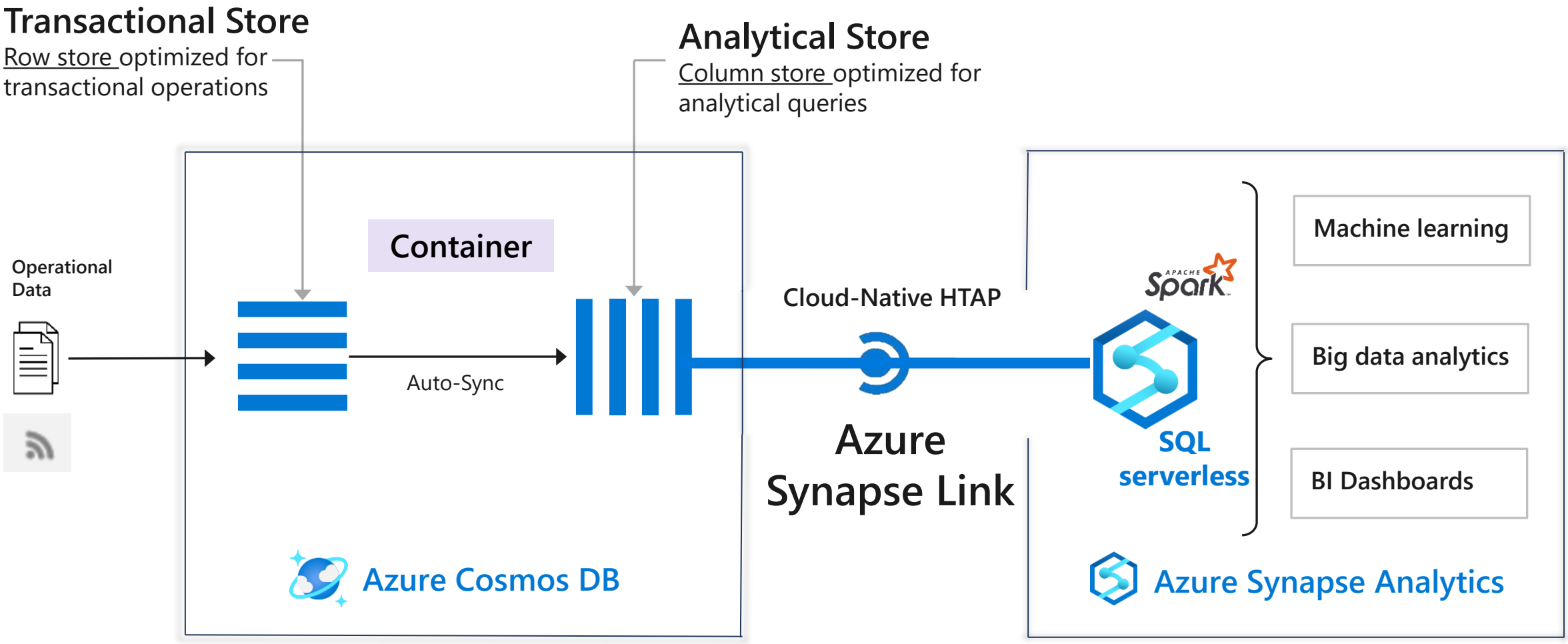
Azure Synapse Link for Azure Cosmos DB

Breaking down the barrier between OLTP & OLAP



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

Azure Synapse Link : How it works?

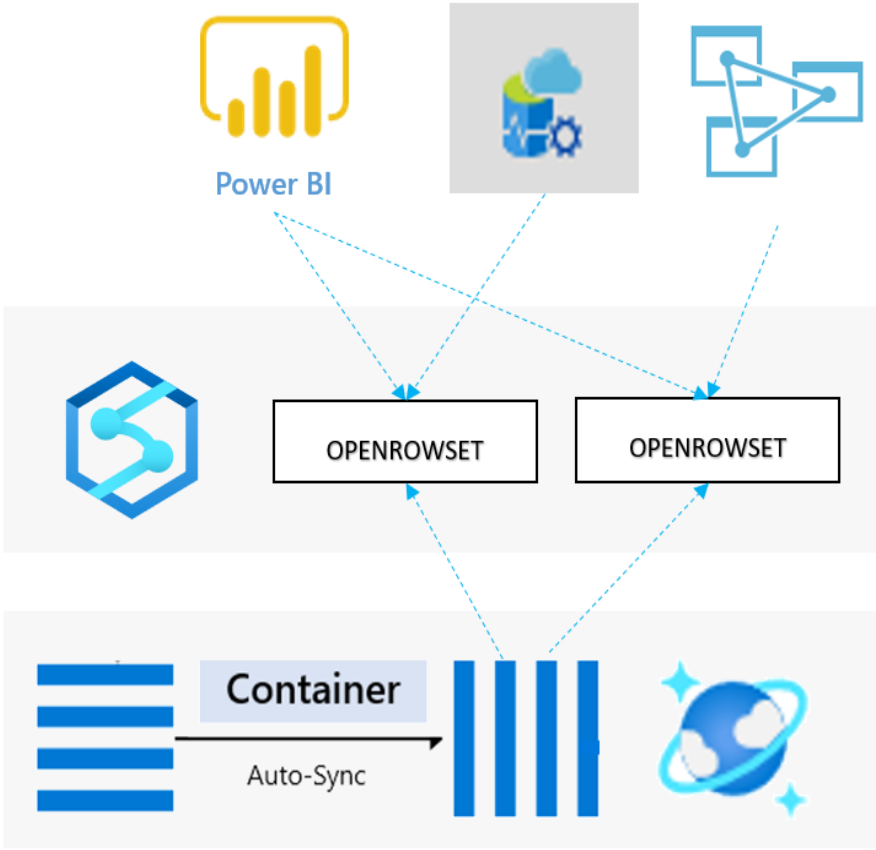


Generate near real-time insights on your operational data

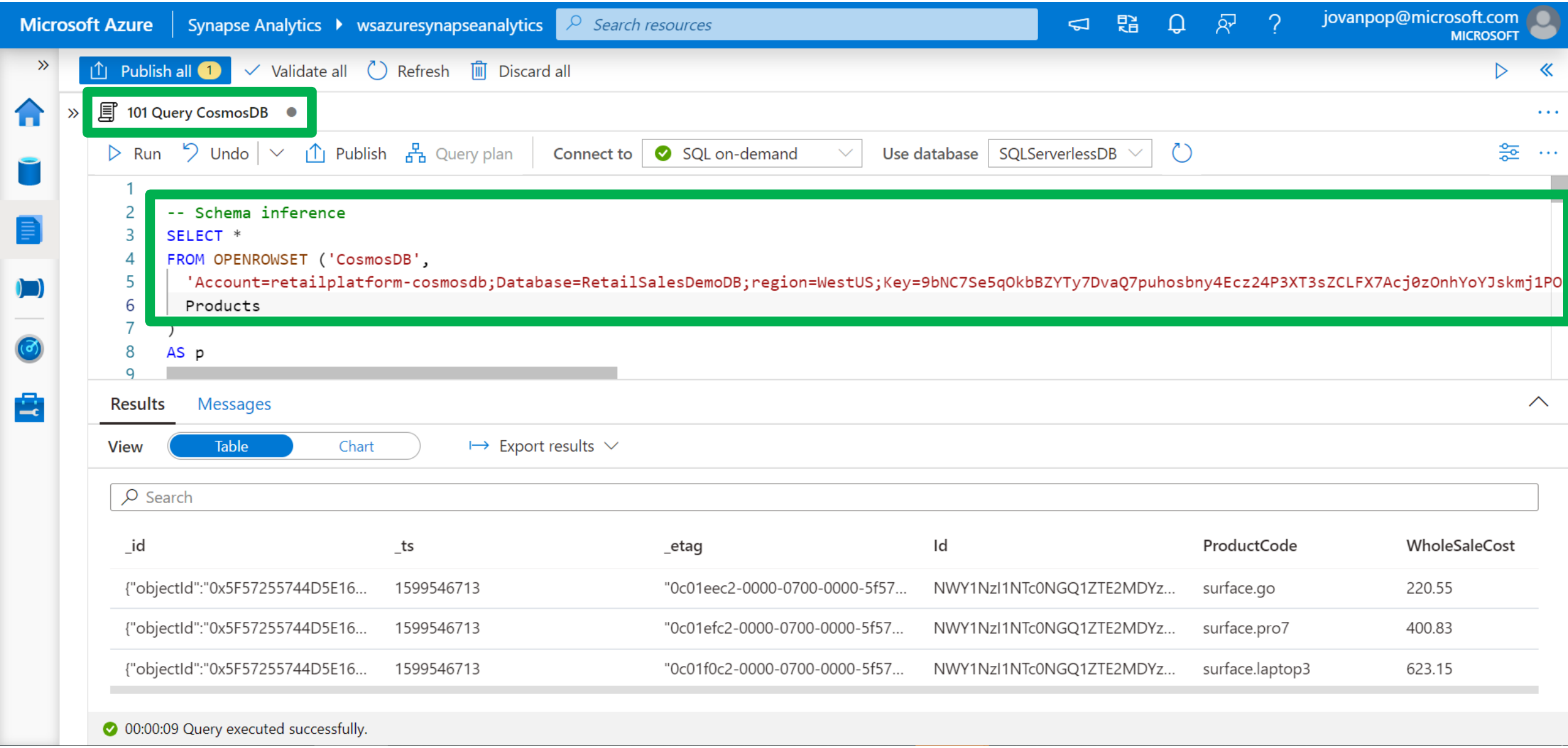
Azure Synapse Link :
Query Azure Cosmos DB Data in Synapse SQL serverless

NEW

- Run *analytical T-SQL queries:*
  Query Azure Cosmos DB data in place, in seconds
- Build rich *near real-time dashboards:*
  Using Power BI integrated in Azure Synapse
- Use wider range of *BI and ad-hoc querying* tools:
  Integrated using T-SQL interface
- Build a *logical data warehouse* :
  Analyze unified view of data across Azure Cosmos DB, Azure Data Lake Storage & Azure Blob Storage



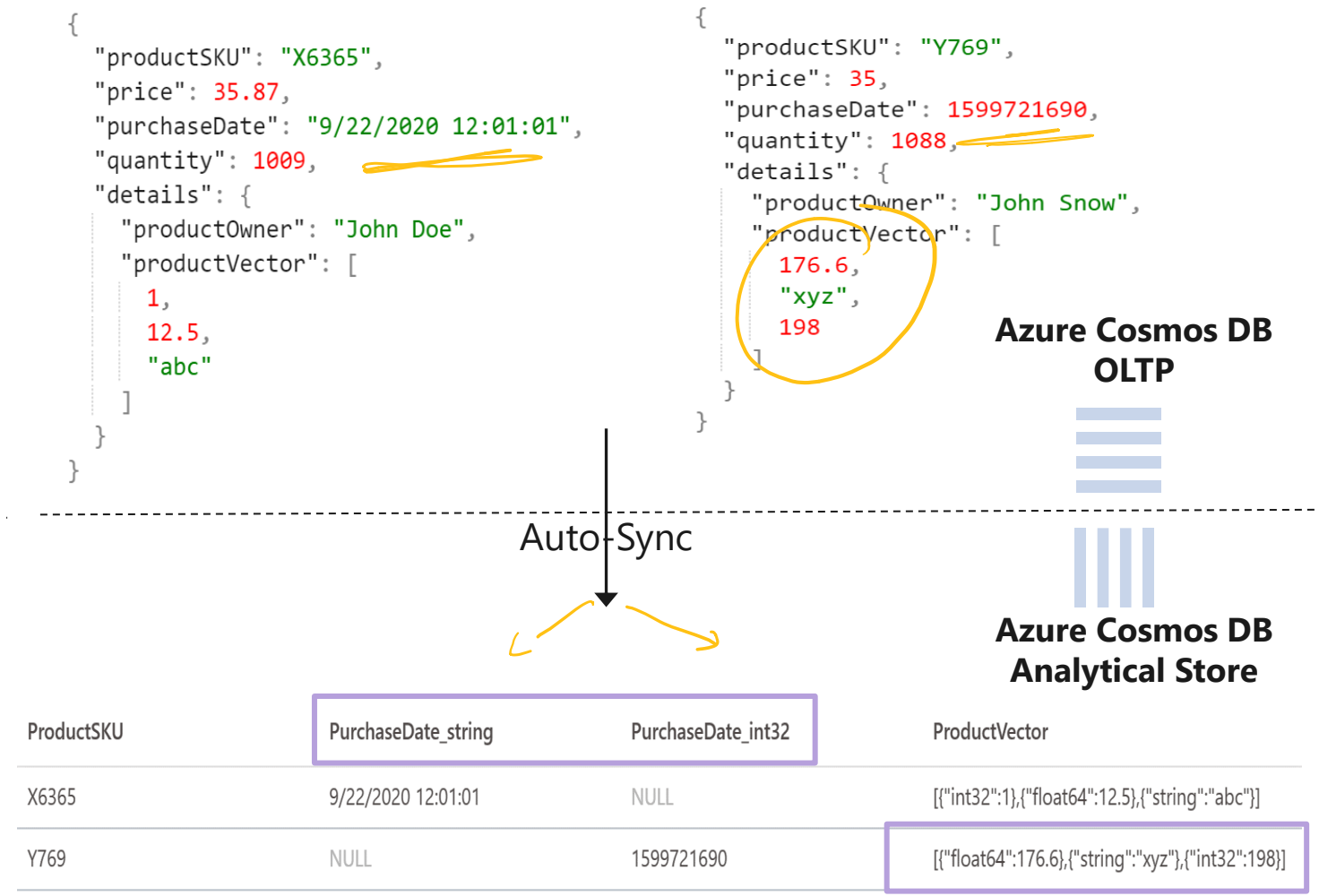
Available in public preview, starting October 2020


Microsoft Azure
Synapse Analytics
wsazuresynapseanalytics
Search resources
jovanpop@microsoft.com
MICROSOFT
Publish all 1
Validate all
Refresh
Discard all
101 Query CosmosDB
Run
Undo
Publish
Query plan
Connect to
SQL on-demand
Use database
SQLServerlessDB
1
2 -- Schema inference
3 SELECT *
4 FROM OPENROWSET ('CosmosDB',
5 'Account=retailplatform-cosmosdb;Database=RetailSalesDemoDB;region=WestUS;Key=9bNC7Se5qOkbBZYTy7DvaQ7puhosbny4Ecz24P3XT3sZCLFX7Acj0zOnhYoYJskmj1PO
6 Products
7)
8 AS p
9
Results
Messages
View
Table
Chart
Export results
Search
_id
_ts
_etag
Id
ProductCode
WholeSaleCost
{"objectId":"0x5F57255744D5E16...
1599546713
"0c01eec2-0000-0700-0000-5f57...
NWY1NzI1NTc0NGQ1ZTE2MDYz...
surface.go
220.55
{"objectId":"0x5F57255744D5E16...
1599546713
"0c01efc2-0000-0700-0000-5f57...
NWY1NzI1NTc0NGQ1ZTE2MDYz...
surface.pro7
400.83
{"objectId":"0x5F57255744D5E16...
1599546713
"0c01f0c2-0000-0700-0000-5f57...
NWY1NzI1NTc0NGQ1ZTE2MDYz...
surface.laptop3
623.15
00:00:09 Query executed successfully.

Azure Synapse Link : Support for Azure Cosmos DB API for MongoDB

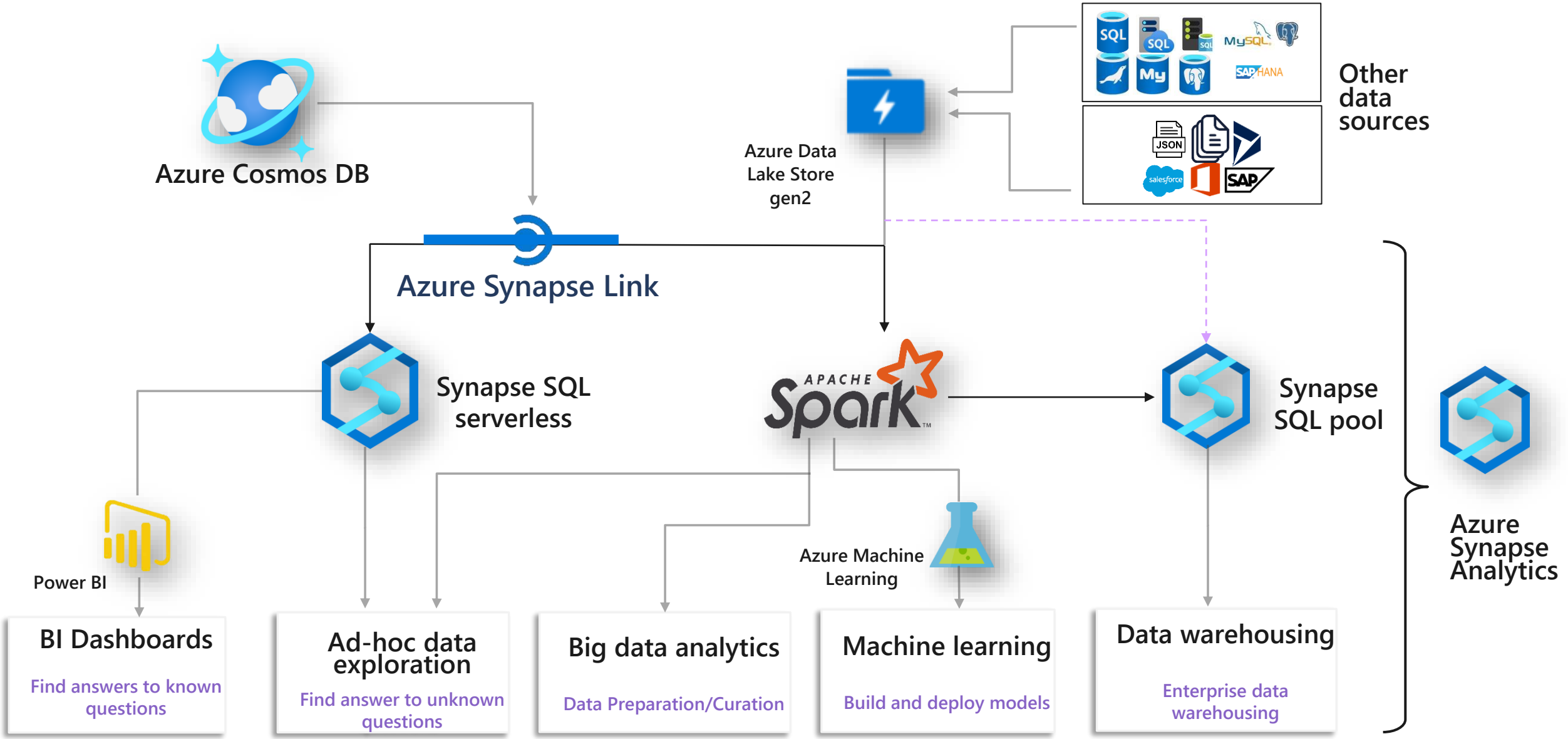
NEW

- Cloud native HTAP for MongoDB data
- Run Synapse T-SQL & Spark queries against MongoDB:
  No impact to OLTP workloads
  No need to ETL to OLAP store
- Analytics against highly polymorphic MongoDB BSON schemas:
  Full-fidelity schema representation in Azure Cosmos DB analytical store (fully managed in auto-sync)



Available now in public preview

Analytics + BI patterns with Azure Synapse Link



Summary

- **Unlock near real-time, no-ETL analytics at scale with Synapse Link**
  - NEW: Support for querying Azure Cosmos DB with SQL serverless in public preview starting Oct 2020
  - NEW: Support for Cosmos DB API for Mongo DB in Synapse Link in public preview
- **NEW: For Spark**
  - Available now
    - Enhanced notebooks
    - Updated magics
    - mssparkutils API
  - Oct
    - Hardware aware Caching and Shuffle Service

Microsoft Azure Solutions Business

Ressources

Self-Service Learning

Read and try out the following sections from the documentation:

a) Quickstart
b) Tutorials
c) Concepts
d) How to guides

https://docs.microsoft.com/en-us/azure/postgresql/
https://docs.microsoft.com/en-us/azure/mysql/

Microsoft Azure Solutions Business

Thank you