



Exam DP-420: Microsoft Azure Cosmos DB Developer Crash Course

Developing Cloud Native Applications Using Cosmos DB



Reza Salehi

Cloud Consultant and Trainer



@zaalion



Course Overview



Course Repository

<https://github.com/zaalion/oreilly-dp-420>



main 1 branch 0 tags

Go to file Add file <> Code

rezasalehinewsig demo code	
DEMOS/StreamAnalytics	demo code
.gitignore	Updates
OReilly-DP-420-Slide-Deck-final.pdf	Updates
Queries.txt	Updates
README.md	Initial commit
Resources.txt	demo code

README.md

oreilly-dp-420

Local Codespaces New

Clone ?

HTTPS SSH GitHub CLI

https://github.com/zaalion/oreilly-dp-420.git

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Open with Visual Studio

Download ZIP





Official DP-420 Exam Documentation

<https://docs.microsoft.com/en-us/certifications/exams/dp-420>



Two ways to prepare

Online - Free

Instructor-led - Paid

Items in this collection



LEARNING PATH

Get started with Azure Cosmos DB SQL API

2 Modules

Intermediate

Data Engineer

Azure

Start >

+ Save



LEARNING PATH

Plan and implement Azure Cosmos DB SQL API

3 Modules

Intermediate

Developer

Cosmos DB

+ Save



LEARNING PATH

Connect to Azure Cosmos DB SQL API with the SDK

2 Modules

Intermediate

Developer

Cosmos DB



DP-420 Crash Course

- Design and implement data models
- Design and implement data distribution
- Integrate an Azure Cosmos DB solution
- Optimize an Azure Cosmos DB solution
- Maintain an Azure Cosmos DB solution



Design and Implement Data Models

Design a Non-relational Data Model for Azure Cosmos DB for NoSQL

- Develop a design by storing multiple entity types in the same container
- Develop a design by storing multiple related entities in the same document
- Develop a model that denormalizes data across documents
- Develop a design by referencing between documents
- Identify primary and unique keys
- Identify data and associated access patterns
- Specify a default TTL on a container for a transactional store



Cosmos DB is a Schema-free Database

```
1 {  
2   "name": "Reza",  
3   "email": "Reza@contoso.com"  
4 }  
5
```

```
21  
22 {  
23   "title": "C# Cookbook",  
24   "ISBN": "123-456-789"  
25 }  
26
```

```
7  
8 {  
9   "name": "Reza",  
10  "last": "Salehi",  
11  "email": "Reza@contoso.com",  
12  "Courses": [  
13    "DP-420",  
14    "AZ-900",  
15    "AZ-500",  
16    "AI-102"  
17  ]  
18 }  
19
```

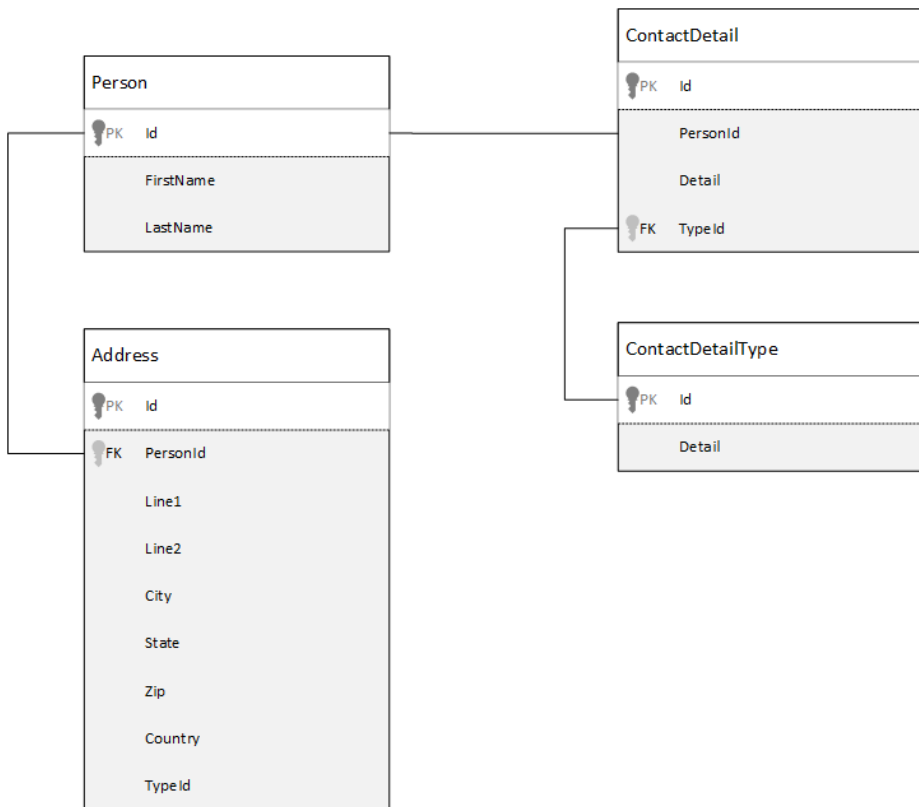


Data Modeling in Azure Cosmos DB

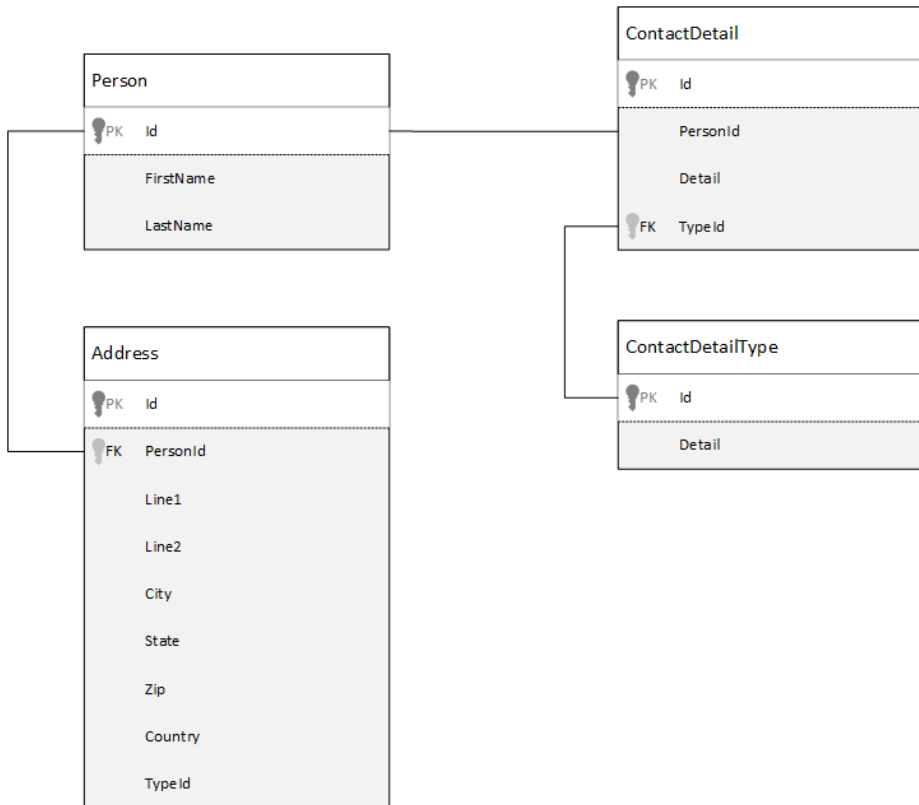
- **Embed data:** Entities will be stored as self-contained items represented as JSON documents. (denormalized)
- **Reference data:** Entities will be stored as separate JSON documents with references to one another. (normalized)
- **Hybrid data models:** Combination of the above



Embed Data



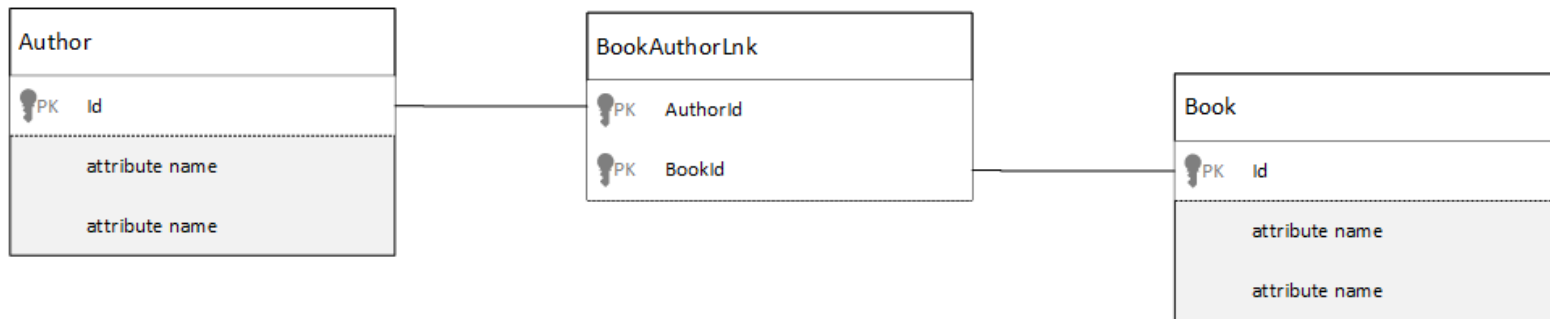
Embed Data



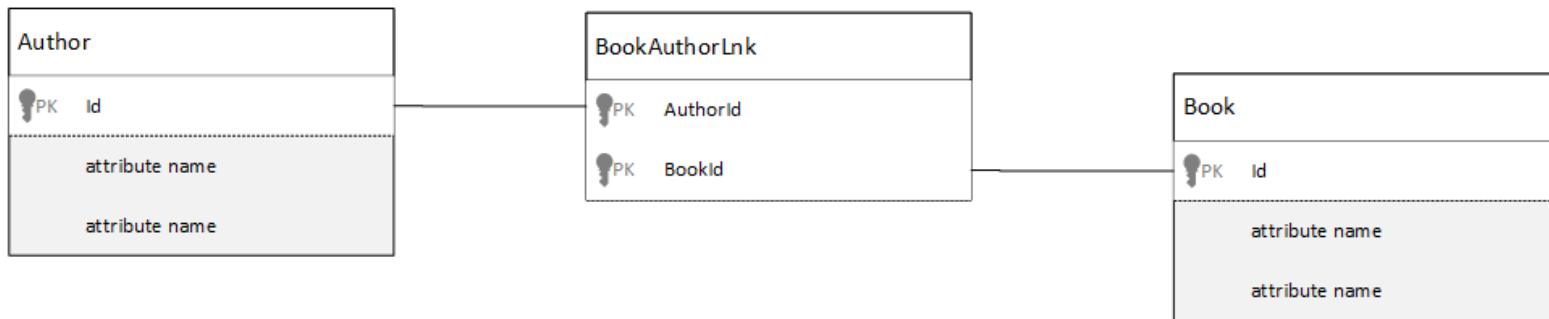
```
{
  "id": "1",
  "firstName": "Thomas",
  "lastName": "Andersen",
  "addresses": [
    {
      "line1": "100 Some Street",
      "line2": "Unit 1",
      "city": "Seattle",
      "state": "WA",
      "zip": 98012
    }
  ],
  "contactDetails": [
    { "email": "thomas@andersen.com" },
    { "phone": "+1 555 555-5555", "extension": 5555 }
  ]
}
```



Reference Data



Reference Data



Publisher document:

```
{
  "id": "mspress",
  "name": "Microsoft Press",
  "books": [ 1, 2, 3, ..., 100, ..., 1000 ]
}
```

Book documents:

```
{ "id": "1", "name": "Azure Cosmos DB 101" }
{ "id": "2", "name": "Azure Cosmos DB for RDBMS Users" }
{ "id": "3", "name": "Taking over the world one JSON doc at a time" }
...
{ "id": "100", "name": "Learn about Azure Cosmos DB" }
...
{ "id": "1000", "name": "Deep Dive into Azure Cosmos DB" }
```



Azure Cosmos DB Keys

- Partition Key (Primary Key)
- Unique Key

Estimated cost (USD) ⓘ: <price>

* Container id ⓘ

Container

* Partition key ⓘ

/partitionKey

Unique keys ⓘ

/firstName, /lastName, /emailAddress



/address/zipCode



+ Add unique key



Time to Live (TTL) in Azure Cosmos DB

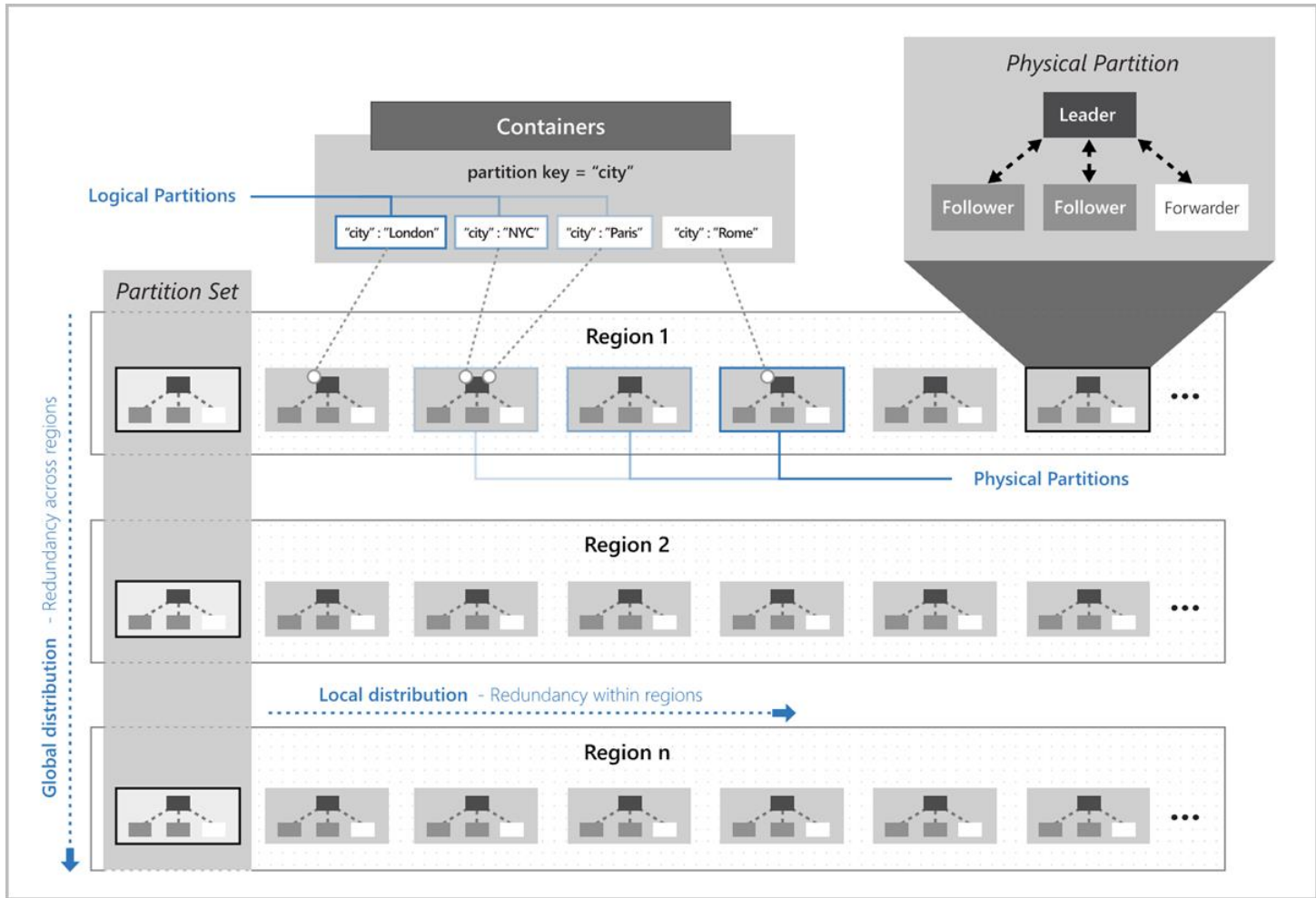
- Ability to delete items automatically from a container after a certain time period.



Design a Data Partitioning Strategy for Azure Cosmos DB for NoSQL

- Choose a partitioning strategy based on a specific workload
- Choose a partition key
- Plan for transactions when choosing a partition key
- Evaluate the cost of using a cross-partition query
- Calculate and evaluate data distribution based on partition key selection
- Calculate and evaluate throughput distribution based on partition key selection
- Construct and implement a synthetic partition key
- Design and implement a hierarchical partition key
- Design partitioning for workloads that require multiple partition keys





Azure Cosmos DB Partitions

- Logical partitions: A set of items that have the same partition key.
- Physical partitions: Smaller containers could have many logical partitions, but they might only need a single physical partition. Physical partitions are entirely managed by Azure Cosmos DB.



Azure Cosmos DB Partition Key

- Its value should not change
- Only “string” values
- This property should have a wide range of possible values to avoid “hot”, or “cold” partitions.
- Spread process, and data storage evenly across all logical partitions as much as possible
- **“A logical partition also defines the scope of database transactions”**



Cosmos DB Query Types

- In-partition query
- Cross-partition query



Cosmos DB Query Types

- In-partition query

SQL

```
SELECT * FROM c WHERE c.DeviceId = 'XMS-0001'
```

- Cross-partition query

SQL

```
SELECT * FROM c WHERE c.Location = 'Seattle'
```



Synthetic Partition Key

- It is best to have a partition key with several values, in the hundreds or thousands
- This results in evenly distributing the data (and workload)
- In many cases, such a property doesn't exist in your data, so you can create a synthetic partition key.



Synthetic Partition Key

```
{  
  "deviceId": "abc-123",  
  "date": 2018  
}
```



```
{  
  "deviceId": "abc-123",  
  "date": 2018,  
  "partitionKey": "abc-123-2018"  
}
```

Synthetic Partition Key Types

- Concatenate multiple properties of an item
- Use a partition key with a random suffix (e.g., 2022-09-09.24)
- Use a partition key with pre-calculated suffixes (e.g., Vehicle-Identification-Number, VIN)



Hierarchical Partition Key Types

- Configure up to a three-level hierarchy for your partition keys to further optimize data distribution and enable higher scale.

C#

 Copy

```
// List of partition keys, in hierarchical order. You can have up to three levels of k
List<string> subpartitionKeyPaths = new List<string> {
    "/TenantId",
    "/UserId",
    "/SessionId"
};
```

[Reference](#)



Plan and Implement Sizing and Scaling for a Database Created with Azure Cosmos DB


- Evaluate the throughput and data storage requirements for a specific workload
- Choose between serverless and provisioned models
- Choose when to use database-level provisioned throughput
- Design for granular scale units and resource governance
- Evaluate the cost of the global distribution of data
- Configure throughput for Azure Cosmos DB by using the Azure portal





Azure Cosmos DB Capacity Calculator

Azure Cosmos DB Account Settings

The simplified Azure Cosmos DB calculator assumes commonly used settings for indexing policy, consistency, and other parameters. For a more accurate estimate, please [sign in](#) to provide your workload details.


API 


Number of regions 


Multi-region writes  ☒ Disabled ☐ Enabled


Workload per region


For a more accurate cost estimate based on your own data, please [sign in](#) and upload your sample data.


Total data stored in transactional store  GB


Use Analytical Store  ☒ Off ☐ On


Item size (upto 2048 KB)  KB

Point reads/sec 

Creates/sec 

Updates/sec 

Deletes/sec 

Queries/sec 

Calculate



Cost Estimate

Transactional Storage

Cost per GB/month	0.25 USD
Total Data stored per region	x 10 GB
EST. STORAGE COST PER MONTH	2.50 USD

Transactional Workload

Cost per 100 RU/s per hour	0.008 USD
EST. THROUGHPUT REQUIRED Show Details	x 400 RU/s
* Minimum throughput required is 400 RU/s	
EST. WORKLOAD COST/MONTH	23.36 USD

Number of regions	x 1
EST. TOTAL COST/MONTH	25.86 USD

Create Cosmos DB account

NEW TO AZURE COSMOS DB? CREATE A NEW FREE TIER ACCOUNT
[Learn more](#)

HAVE AN APP WITH BURSTY TRAFFIC? TRY OUT SERVERLESS
[Learn more](#)

MIGRATE TO AZURE COSMOS DB NOW
[Learn more](#)



Cosmos DB Provisioning Models

- Provisioned: You will set throughput (RUs) for your databases or/and containers
- Auto-scale: “Scale the throughput (RU/s) of your database or container automatically and instantly”
- Serverless: Use the Azure Cosmos account in a consumption-based mode
- Auto scale vs. Serverless



Provisioned throughput in Azure Cosmos DB

- At the Azure Cosmos DB containers level
- At the Azure Cosmos DB databases level

Add Container

Start at \$<price>/mo per database, multiple containers included
[More details](#)

* Database id ⓘ

Create new

Use existing

SharedDB1

* Container id ⓘ

B

* Partition key ⓘ

/id

☐ My partition key is larger than 100 bytes

☒ Provision dedicated throughput for this container ⓘ

* Throughput (400 - 100,000 RU/s) ⓘ

Autoscale

Manual

Estimate your required throughput with [capacity calculator](#)

400

Cost (USD): \$<Price>hourly / \$<price>daily / \$<Price>monthly (1 region, 400RU/s, \$<price>/RU)

**This cost is an estimate and may vary based on the regions where your account is deployed and potential discounts applied to your account*

* Analytical store ⓘ

On

Off

Unique keys ⓘ

+ Add unique key

Govern Azure Cosmos DB

All services > Policy

Policy | Definitions

Search (Ctrl+/) << + Policy definition + Initiative definition Export definitions Refresh

Overview

Getting started

Compliance

Remediation

Events

Authoring

Definitions

Assignments

Exemptions

Now export your definitions and assignments to GitHub and manage them using actions! Click on 'Export definition' menu option. Learn more

Name ↑↓	Definition location ↑↓	Policies ↑↓	Type ↑↓	Defir
Azure Cosmos DB allowed locations			BuiltIn	Po
Azure Cosmos DB throughput should be limited			BuiltIn	Po
Azure Cosmos DB accounts should use customer-managed keys to encrypt data a...			BuiltIn	Po
Azure Cosmos DB key based metadata write access should be disabled			BuiltIn	Po
Cosmos DB database accounts should have local authentication methods disabled			BuiltIn	Po
Azure Cosmos DB should disable public network access			BuiltIn	Po
Configure Microsoft Defender for Azure Cosmos DB to be enabled			BuiltIn	Po
Azure Cosmos DB accounts should have firewall rules			BuiltIn	Po
Microsoft Defender for Azure Cosmos DB should be enabled			BuiltIn	Po
Deploy Advanced Threat Protection for Cosmos DB Accounts			BuiltIn	Po
Configure Cosmos DB database accounts to disable local authentication			BuiltIn	Po
Cosmos DB should use a virtual network service endpoint			BuiltIn	Po



Implement Client Connectivity Options in the Azure Cosmos DB SDK

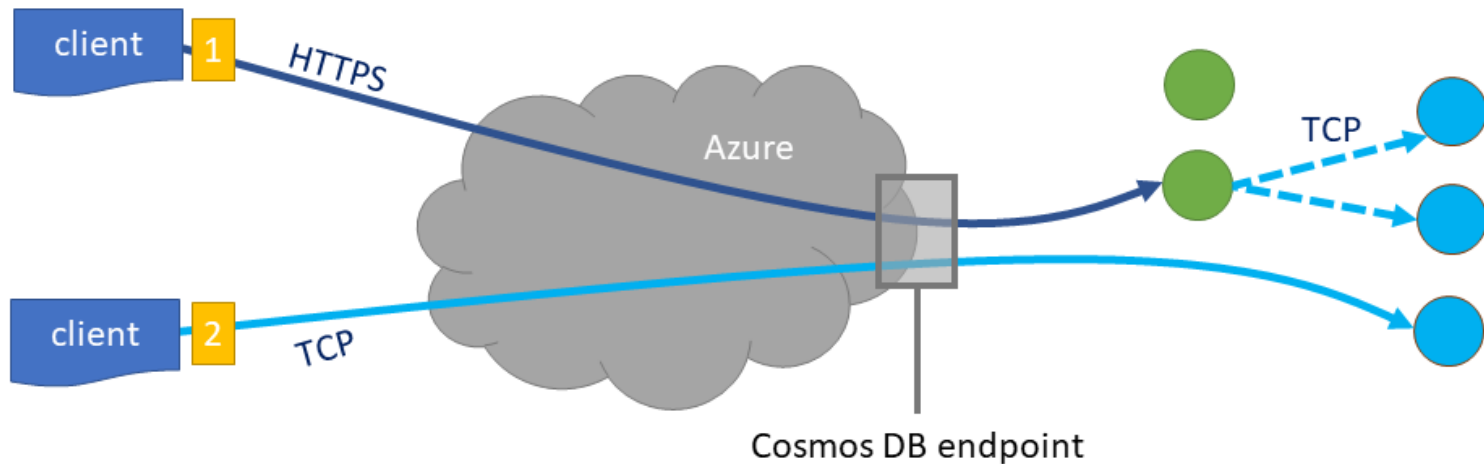
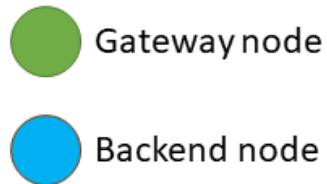
- Choose a connectivity mode (gateway versus direct)
- Implement a connectivity mode
- Create a connection to a database
- Enable offline development by using the Azure Cosmos DB emulator
- Handle connection errors (also see the these)
- Implement a singleton for the client
- Specify a region for global distribution
- Configure client-side threading and parallelism options
- Enable SDK logging



Azure Cosmos DB Connectivity Modes

1 Gateway mode (using HTTPS)

2 Direct mode over TCP



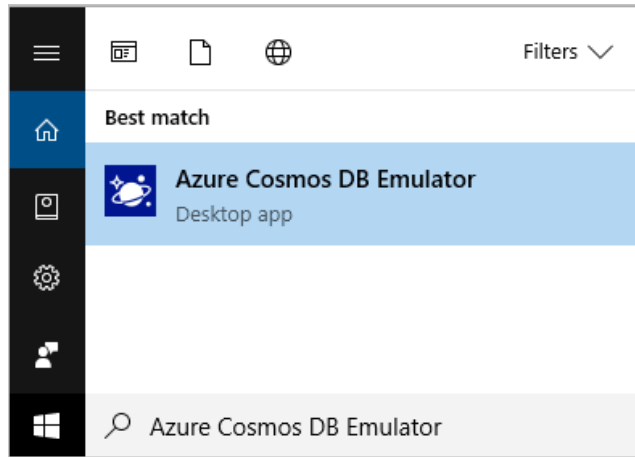
Azure Cosmos DB Connectivity Modes

- Gateway mode: Supported on all SDKs. Best option, if your application runs within a corporate network with strict firewall restrictions. It uses the standard HTTPS port and a single DNS endpoint.
- Direct mode: Supports TCP connectivity, using TLS for authentication and traffic encryption. It offers better performance



Azure Cosmos DB Emulator

- Allows developers to work offline without connecting to a live Azure Cosmos DB instance.
- You can download the emulator from Microsoft



Checklist for Troubleshooting Issues

- Use the latest SDK.
- Review the performance tips and follow the suggested practices.
- Enable the SDK logging to help you troubleshoot an issue.
- Log metrics by using the Azure portal.
- Portal metrics show the Azure Cosmos DB telemetry
- Log the diagnostics string in the V2 SDK or diagnostics in V3 SDK from the point operation responses.
- Log the SQL Query Metrics from all the query responses



Connection Errors

- Request header too large
- Request timeout exceptions
- Slow requests
- Service unavailable exceptions



Implement Data Access by Using the Language for Azure Cosmos DB for NoSQL

- Implement queries that use arrays, nested objects, aggregation, and ordering
- Implement a correlated subquery
- Implement queries that use array and type-checking functions
- Implement queries that use mathematical, string, and date functions
- Implement queries based on variable data



Implement Data Access by Using Azure Cosmos DB for NoSQL SDKs

- Choose when to use a point operation versus a query operation
- Implement a point operation that creates, updates, and deletes documents
- Implement an update by using a patch operation
- Manage multi-document transactions using SDK Transactional Batch
- Perform a multi-document load using Bulk Support in the SDK
- Implement optimistic concurrency control using ETags
- Implement session consistency by using session tokens
- Implement a query operation that includes pagination
- Implement a query operation by using a continuation token
- Handle transient errors and 429s
- Specify TTL for a document
- Retrieve and use query metrics



Point Operation vs. Query Operation

- Ways to read data in Cosmos DB
 - Point reads (SDK): A key/value lookup on a single item ID and partition key.
 - Query reads

	Point read (assumes 1 KB item)	Query
Latency	Typically less than 10 ms	Variable
RU charge	1 RU	At least 2.3 RUs, variable
Number of items returned	1 item	Unlimited (if results size is too large, results are split across multiple pages)
Include partition key?	Required	Recommended

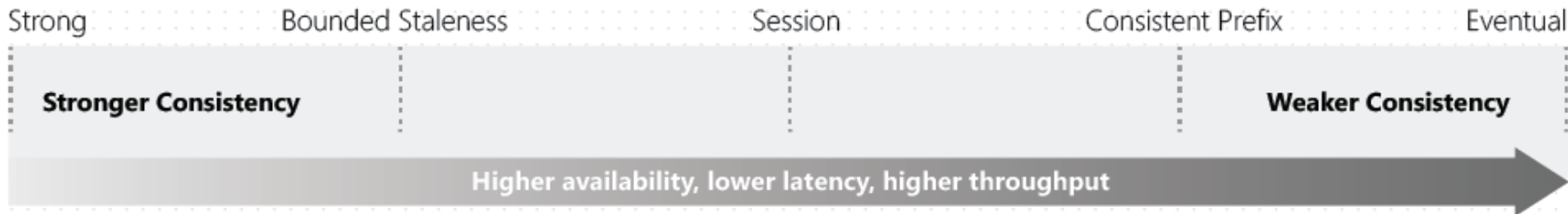


Document Update in Azure Cosmos DB

- Ways to update data in Cosmos DB
 - Update the whole JSON record
 - Partial document update



Azure Cosmos DB Consistency Levels



Specify TTL Using Cosmos DB SDK

```
.NET SDK v3  Java SDK v4  Node SDK  Python SDK

C#  Copy

Database database = client.GetDatabase("database");

ContainerProperties properties = new ()
{
    Id = "container",
    PartitionKeyPath = "/customerId",
    // Never expire by default
    DefaultTimeToLive = -1
};

// Create a new container with TTL enabled and without any expiration value
Container container = await database
    .CreateContainerAsync(properties);
```



Specify TTL Using Cosmos DB SDK

```
.NET SDK v3  Java SDK v4  Node SDK  Python SDK

C#  Copy

Database database = client.GetDatabase("database");

ContainerProperties properties = new ()
{
    Id = "container",
    PartitionKeyPath = "/customerId",
    // Expire all documents after 90 days
    DefaultTimeToLive = 90 * 60 * 60 * 24
};

// Create a new container with TTL enabled and without any expiration value
Container container = await database
    .CreateContainerAsync(properties);
```




Implement Server-side Programming in Azure Cosmos DB for NoSQL by Using JavaScript

- Write, deploy, and call a stored procedure
- Design stored procedures to work with multiple items transactionally
- Implement and call triggers
- Implement a user-defined function



Cosmos DB Stored Procedures


JavaScript

 Copy

```
var helloWorldStoredProc = {  
  id: "helloWorld",  
  serverScript: function () {  
    var context = getContext();  
    var response = context.getResponse();  
  
    response.setBody("Hello, World");  
  }  
}
```


Cosmos DB Pre-triggers

JavaScript

 Copy

```
function validateToDoItemTimestamp() {  
    var context = getContext();  
    var request = context.getRequest();  
  
    // item to be created in the current operation  
    var itemToCreate = request.getBody();  
  
    // validate properties  
    if (!("timestamp" in itemToCreate)) {  
        var ts = new Date();  
        itemToCreate["timestamp"] = ts.getTime();  
    }  
  
    // update the item that will be created  
    request.setBody(itemToCreate);  
}
```



Cosmos DB Post-triggers

JavaScript

Copy

```
function updateMetadata() {
    var context = getContext();
    var container = context.getCollection();
    var response = context.getResponse();

    // item that was created
    var createdItem = response.getBody();

    // query for metadata document
    var filterQuery = 'SELECT * FROM root r WHERE r.id = "_metadata"';
    var accept = container.queryDocuments(container.getSelfLink(), filterQuery,
        updateMetadataCallback);
    if(!accept) throw "Unable to update metadata, abort";

    function updateMetadataCallback(err, items, responseOptions) {
        if(err) throw new Error("Error" + err.message);
        if(items.length != 1) throw 'Unable to find metadata document';

        var metadataItem = items[0];

        // update metadata
        metadataItem.createdItems += 1;
        metadataItem.createdNames += " " + createdItem.id;
        var accept = container.replaceDocument(metadataItem._self,
            metadataItem, function(err, itemReplaced) {
                if(err) throw "Unable to update metadata, abort";
            });
        if(!accept) throw "Unable to update metadata, abort";
        return;
    }
}
```

Reference



Cosmos DB UDF

JavaScript

Copy

```
function tax(income) {  
  if (income == undefined)  
    throw 'no input';  
  
  if (income < 1000)  
    return income * 0.1;  
  else if (income < 10000)  
    return income * 0.2;  
  else  
    return income * 0.4;  
}
```



Design and Implement Data Distribution

Design and implement a replication strategy for Azure Cosmos DB

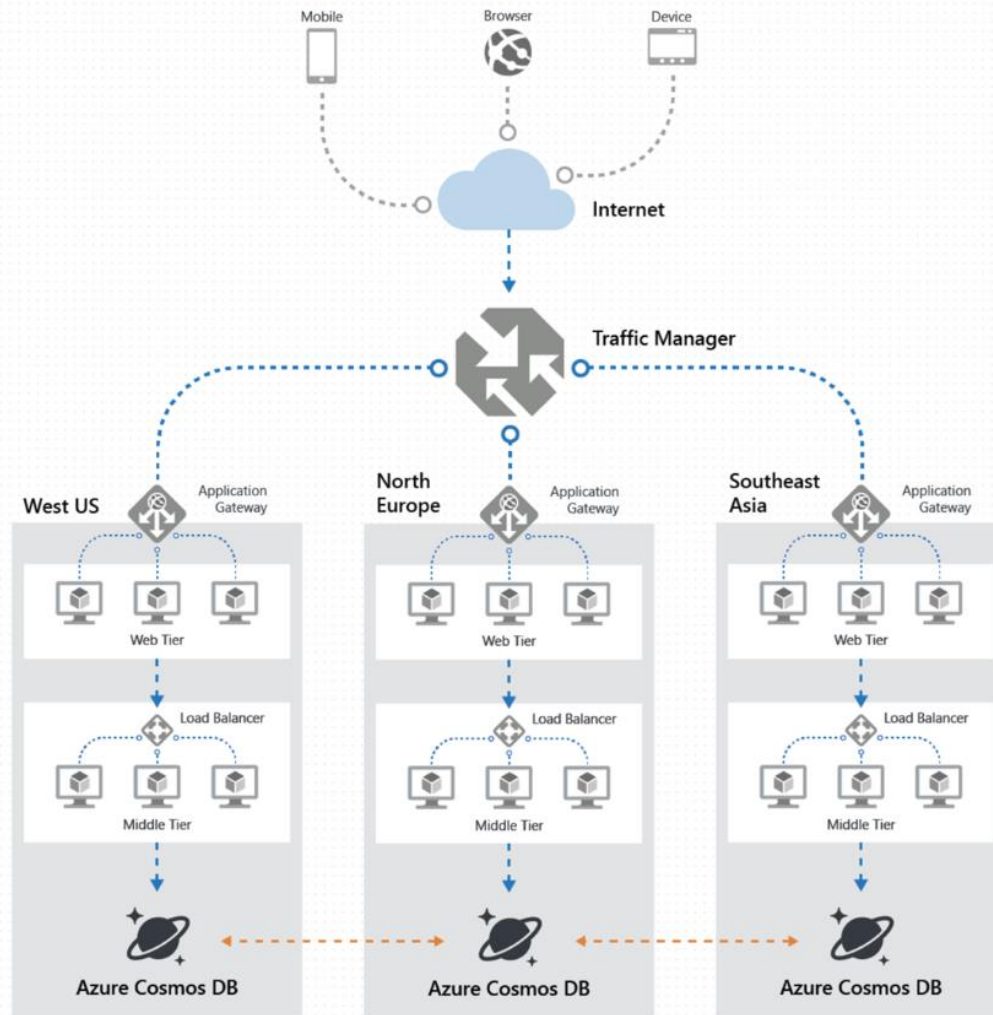
- Choose when to distribute data
- Automatic failover policies for regional failure for Azure Cosmos DB Core API
- Perform manual failovers to move single master write regions
- Choose a consistency model
- Identify use cases for different consistency models
- Evaluate the impact of consistency model choices on availability and associated RU cost
- Evaluate the impact of consistency model choices on performance and latency
- Specify application connections to replicated data



Distribute Data

- Azure Cosmos DB is a globally distributed database system
- Allows you to read and write data from the local region replicas of your database.
- Azure Cosmos DB transparently replicates the data to all the regions associated with your Cosmos account.





notebookaccount | Replicate data globally

Azure Cosmos DB account

Search (Ctrl+ /)



Save



Discard



Manual Failover



Automatic Failover

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Quick start

Notifications

Data Explorer

Settings

Features

Replicate data globally

Default consistency

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

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

Configure regions

Multi-region writes ⓘ

Disable

Enable

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

Regions	Reads Enabled	Writes Enabled	Availability Zone	Action
East US 2	✓	✓		
South Central US	✓	✓		



Design and Implement Multi-region Write

- Choose when to use multi-region write
- Implement multi-region write
- Implement a custom conflict resolution policy for Azure Cosmos DB for NoSQL



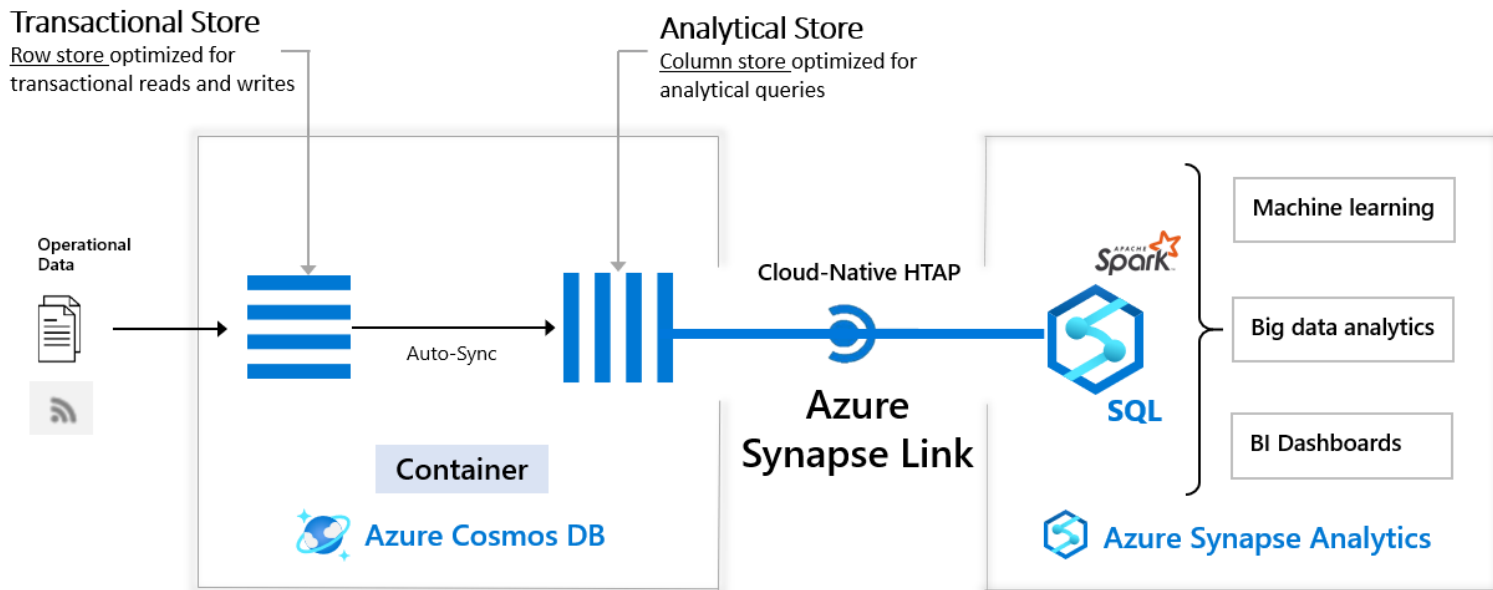
Integrate an Azure Cosmos DB Solution

Enable Azure Cosmos DB Analytical Workloads

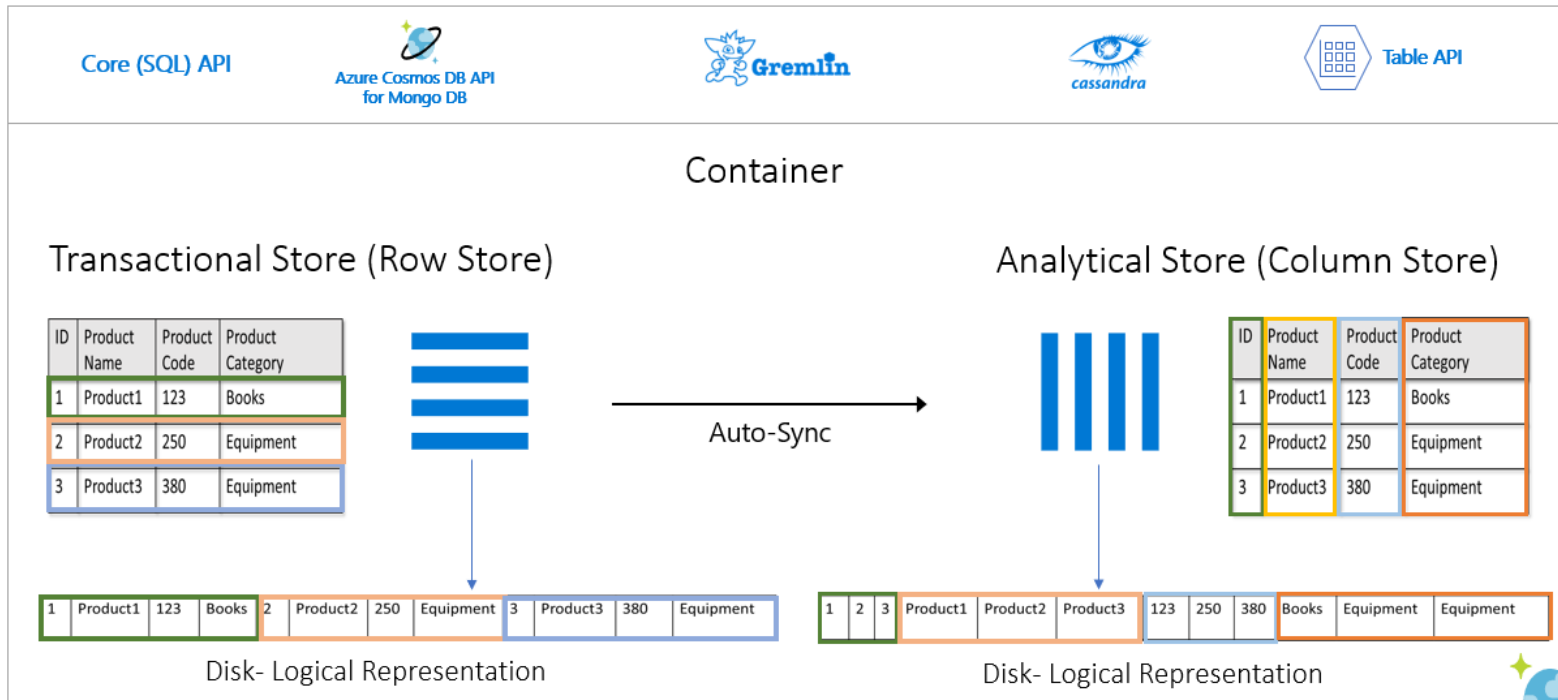
- Enable Azure Synapse Link
- Choose between Azure Synapse Link and Spark Connector
- Enable the analytical store on a container
- Connection to an analytical store and query from Azure Synapse Spark or Azure Synapse SQL
- Perform a query against the transactional store from Spark
- Write data back to the transactional store from Spark

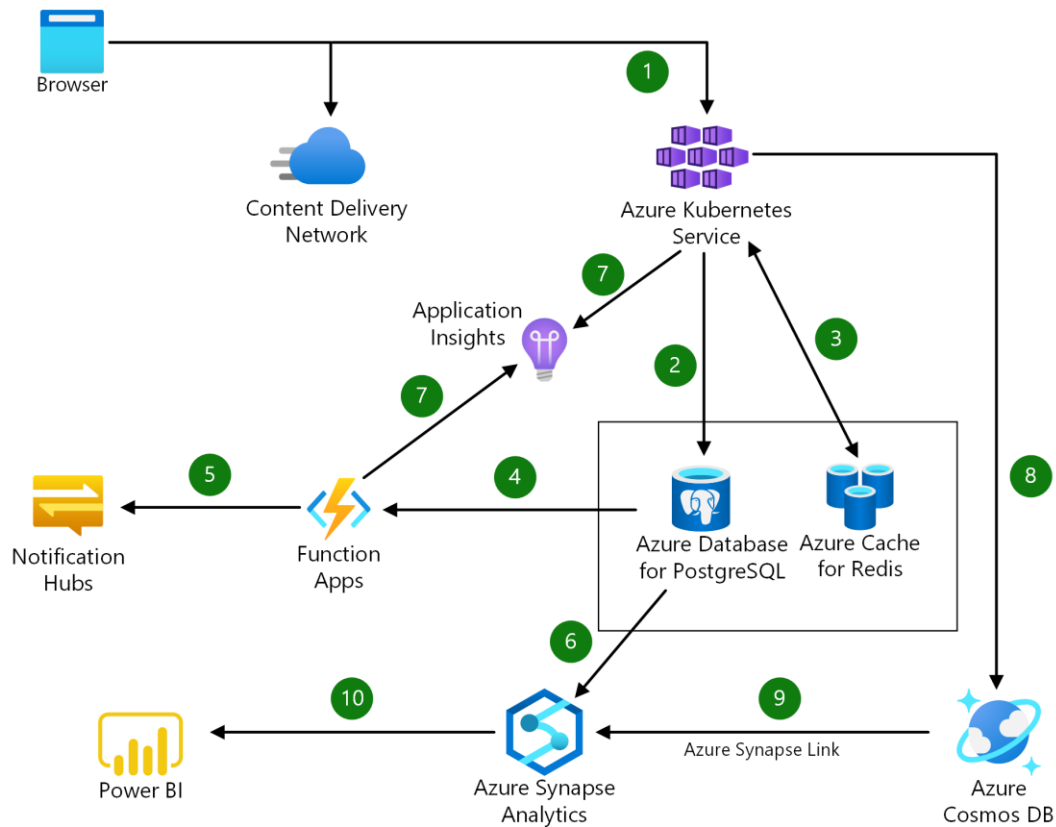


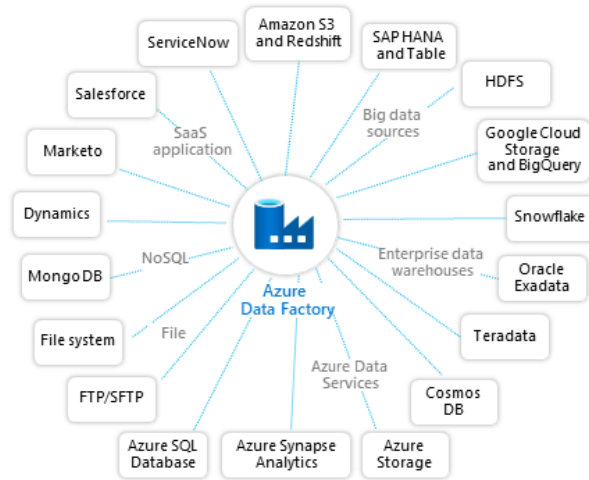
Azure Synapse Link



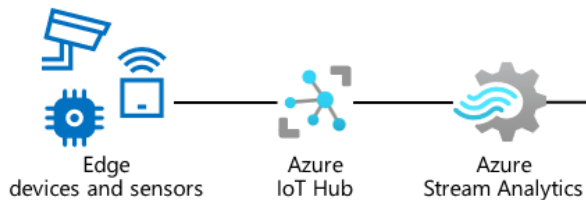
Azure Cosmos DB Analytical Store







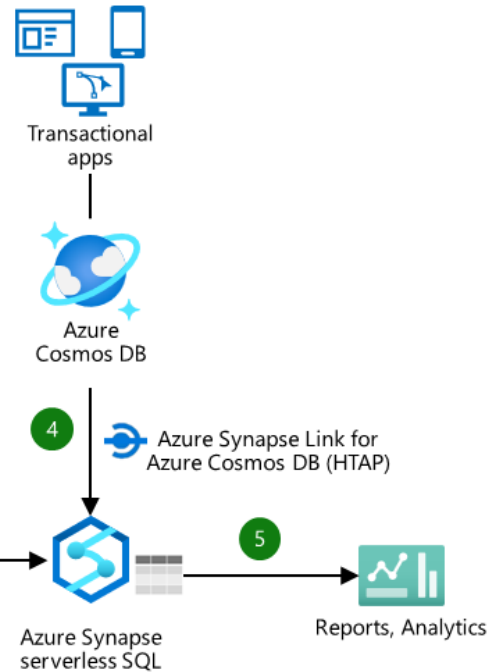
Azure Data Factory
(90+ Connectors)



1

Azure Data Lake

3



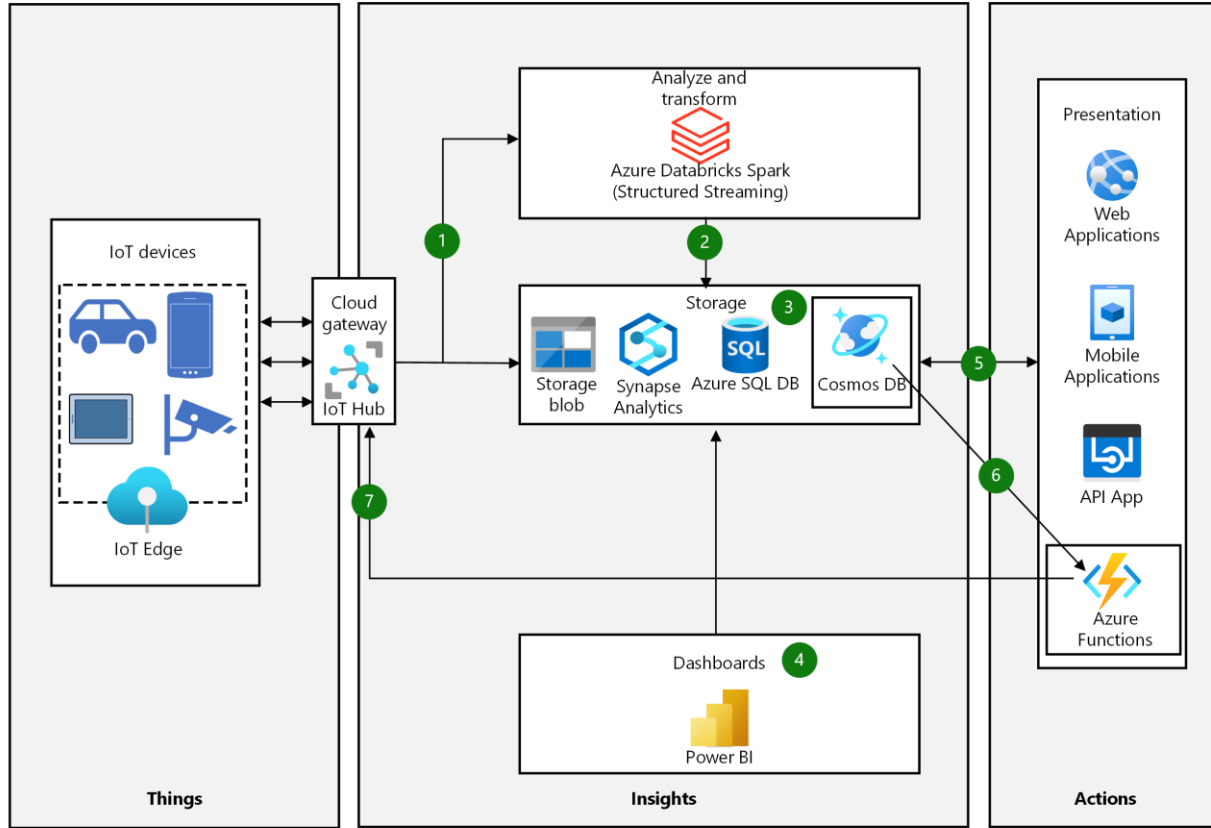
Write Spark DataFrame to Azure Cosmos DB

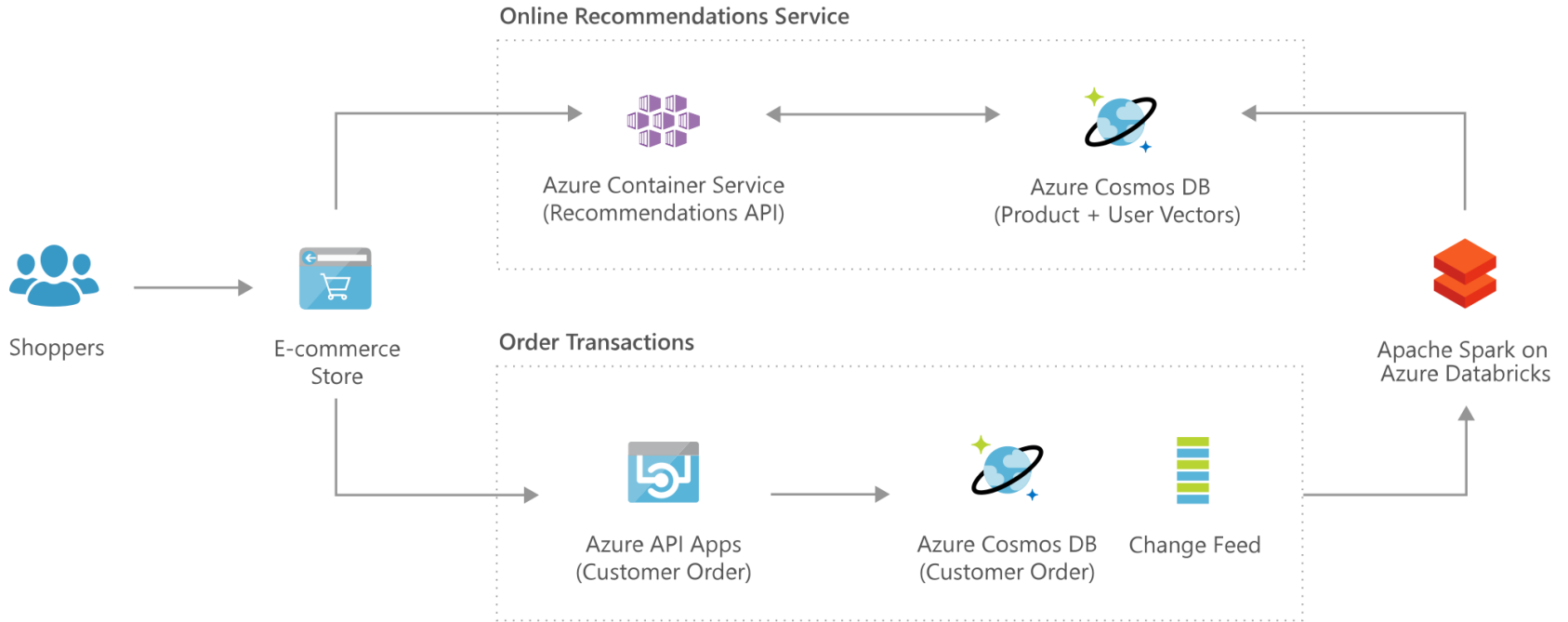
Java

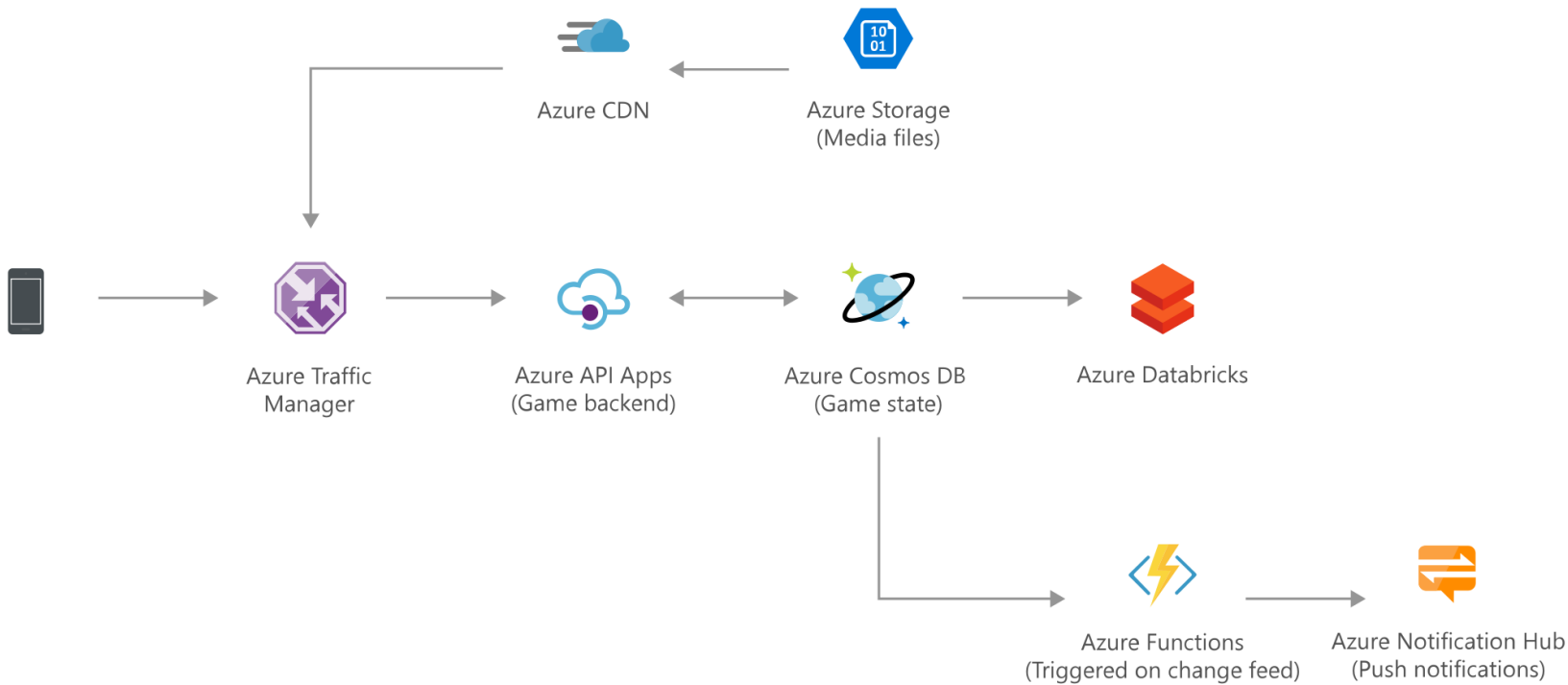
Copy

```
// To select a preferred list of regions in a multi-region Azure Cosmos DB account, add option("spark.cosmos.regions", "  
  
import org.apache.spark.sql.SaveMode  
  
df.write.format("cosmos.oltp").  
    option("spark.synapse.linkedService", "<enter linked service name>").  
    option("spark.cosmos.container", "<enter container name>").  
    option("spark.cosmos.write.upsertEnabled", "true").  
    mode(SaveMode.Overwrite).  
    save()
```





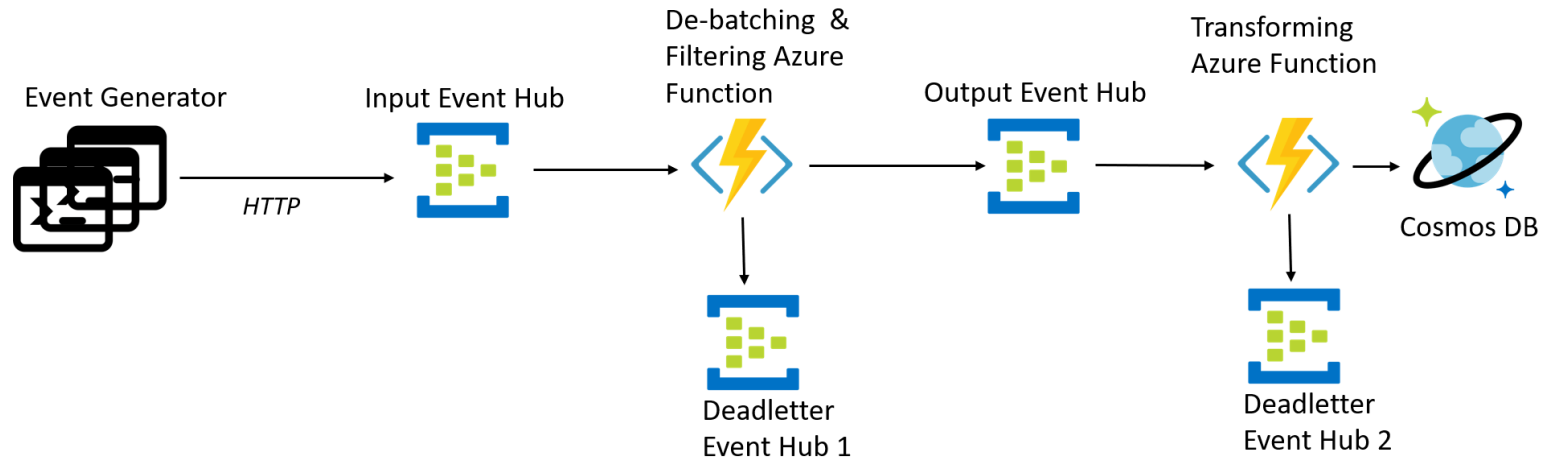




Implement Solutions Across Services

- Integrate events with other applications by using Azure Functions and Azure Event Hubs
- Denormalize data by using Change Feed and Azure Functions
- Enforce referential integrity by using Change Feed and Azure Functions
- Aggregate data by using Change Feed and Azure Functions, including reporting
- Archive data by using Change Feed and Azure Functions
- Implement Azure Cognitive Search for an Azure Cosmos DB solution







Devices

Events



Event Hubs



Function
App

Write



Cosmos DB

Dead letter
messages



Storage queue

CI/CD



Azure
Pipelines

End-to-end
monitoring



Monitor



Optimize an Azure Cosmos DB Solution

Optimize Query Performance in Azure Cosmos DB for NoSQL

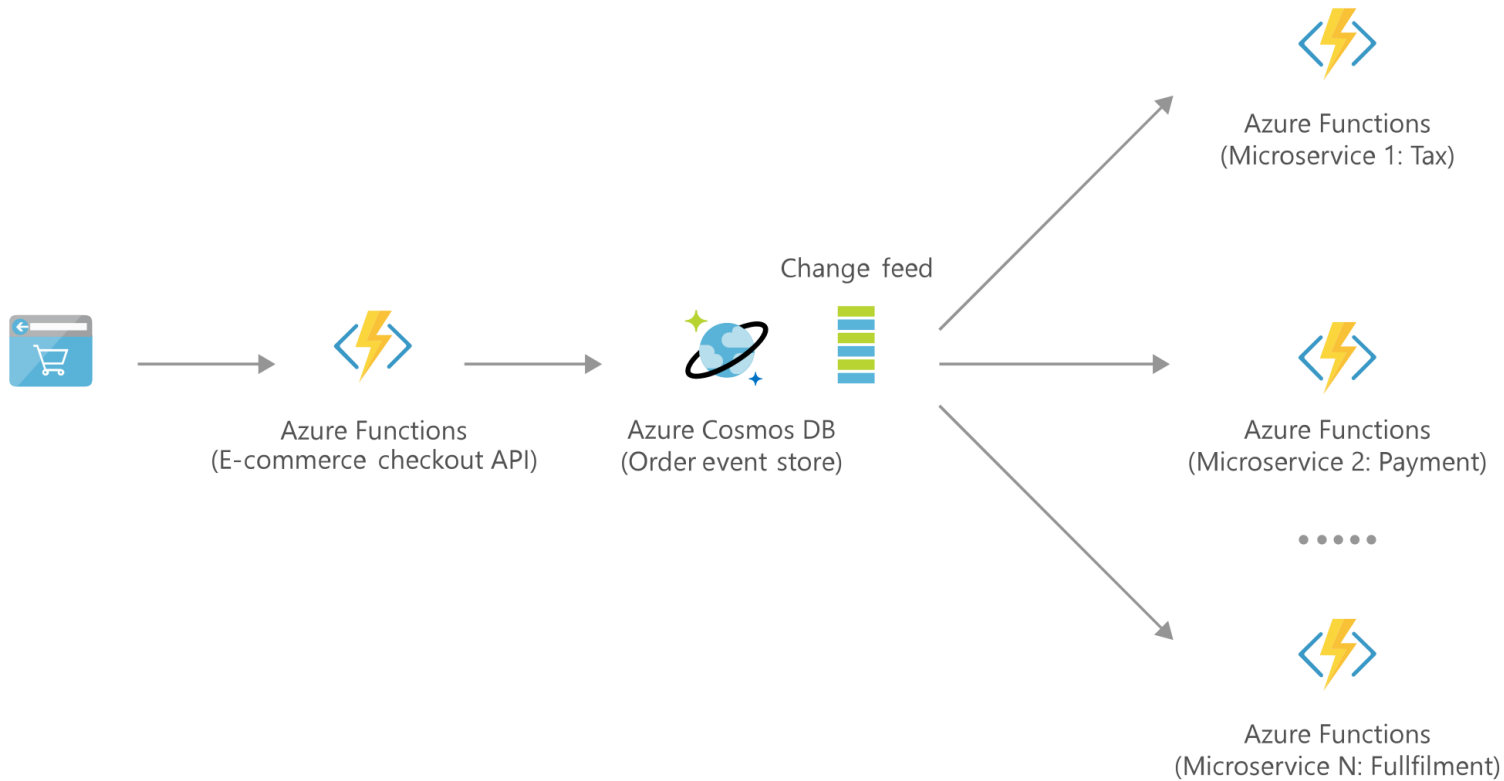
- Adjust indexes on the database
- Calculate the cost of the query
- Retrieve request unit cost of a point operation or query
- Implement Azure Cosmos DB integrated cache



Design and Implement Change Feeds for an Azure Cosmos DB for NoSQL

- Develop an Azure Functions trigger to process a change feed
- Consume a change feed from within an application by using the SDK
- Manage the number of change feed instances by using the change feed estimator
- Implement denormalization by using a change feed
- Implement referential enforcement by using a change feed
- Implement aggregation persistence by using a change feed
- Implement data archiving by using a change feed





Define and Implement an Indexing Strategy for an Azure Cosmos DB for NoSQL

- Choose when to use a read-heavy versus write-heavy index strategy
- Choose an appropriate index type
- Configure a custom indexing policy by using the Azure portal
- Implement a composite index
- Optimize index performance



Maintain an Azure Cosmos DB Solution

Monitor and Troubleshoot an Azure Cosmos DB Solution

- Evaluate response status code and failure metrics
- Monitor metrics for normalized throughput usage by using Azure Monitor
- Monitor server-side latency metrics by using Azure Monitor
- Monitor data replication in relation to latency and availability
- Configure Azure Monitor alerts for Azure Cosmos DB
- Implement and query Azure Cosmos DB logs (and see this)
- Monitor throughput across partitions
- Monitor distribution of data across partitions
- Monitor security by using logging and auditing



Implement Backup and Restore for an Azure Cosmos DB Solution

- Choose between periodic and continuous backup
- Configure periodic backup
- Configure continuous backup and recovery
- Locate a recovery point for a point-in-time recovery
- Recover a database or container from a recovery point



Implement Security for an Azure Cosmos DB Solution

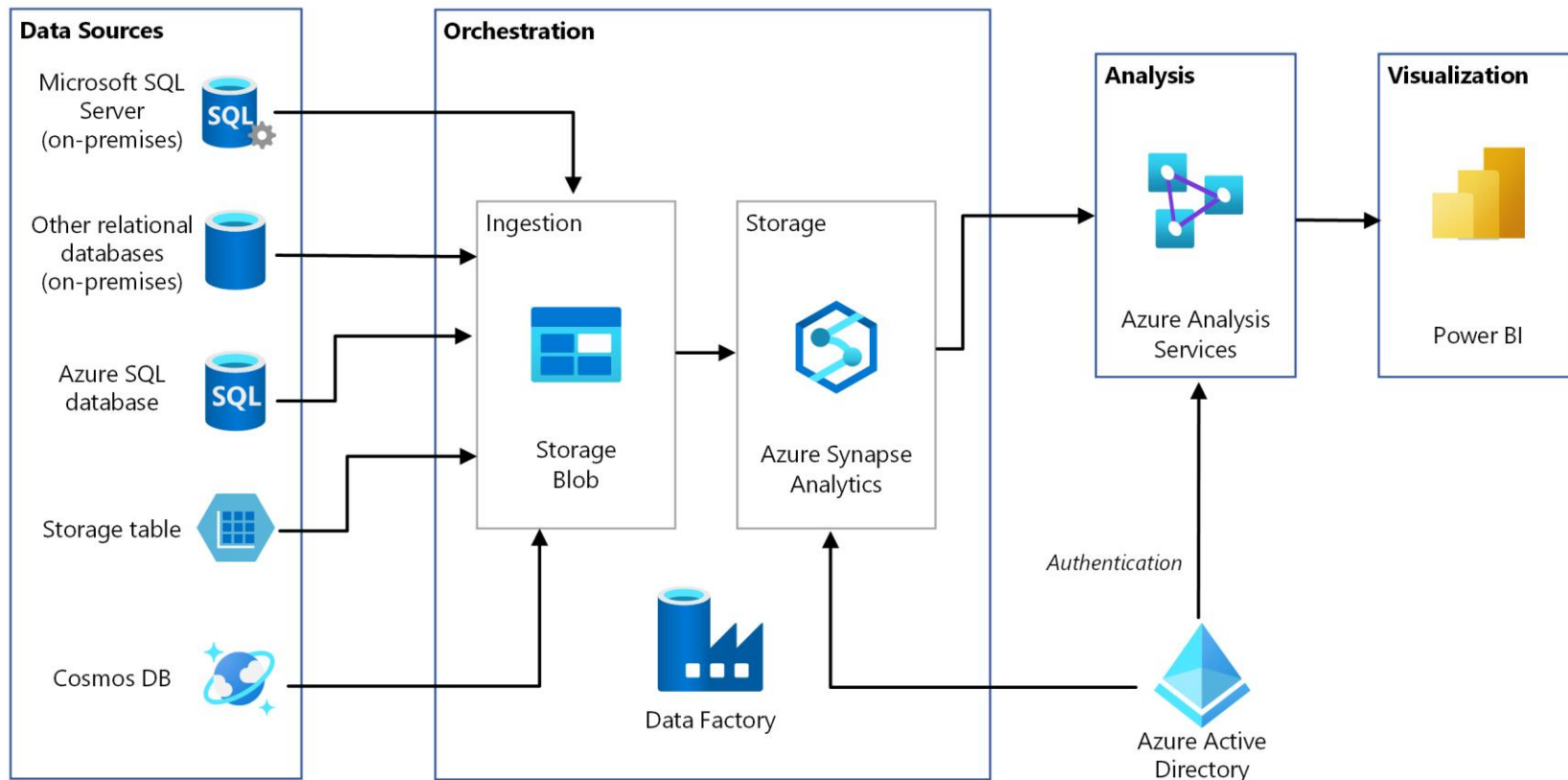
- Choose between service-managed and customer-managed encryption keys
- Configure network-level access control for Azure Cosmos DB
- Configure data encryption for Azure Cosmos DB
- Manage control plane access to Azure Cosmos DB by using Azure role-based access control (RBAC)
- Manage data plane access to Azure Cosmos DB by using keys
- Manage data plane access to Azure Cosmos DB by using Azure Active Directory
- Configure Cross-Origin Resource Sharing (CORS) settings
- Manage account keys by using Azure Key Vault
- Implement customer-managed keys for encryption
- Implement Always Encrypted

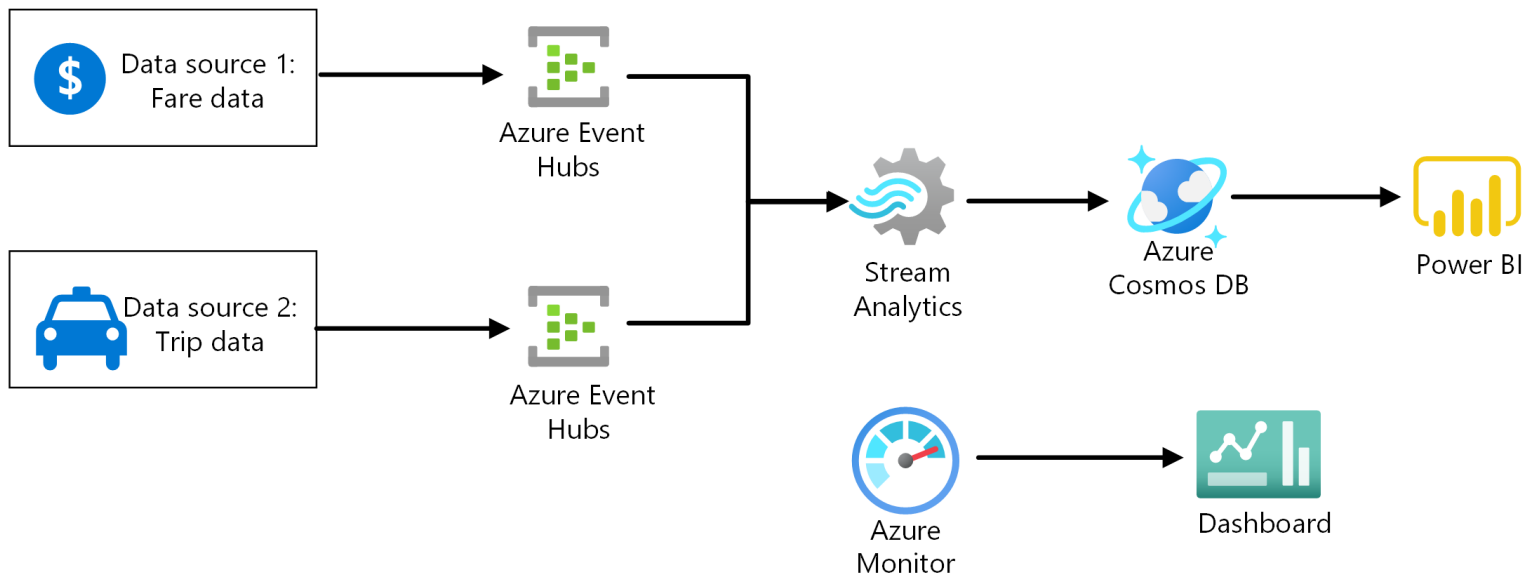


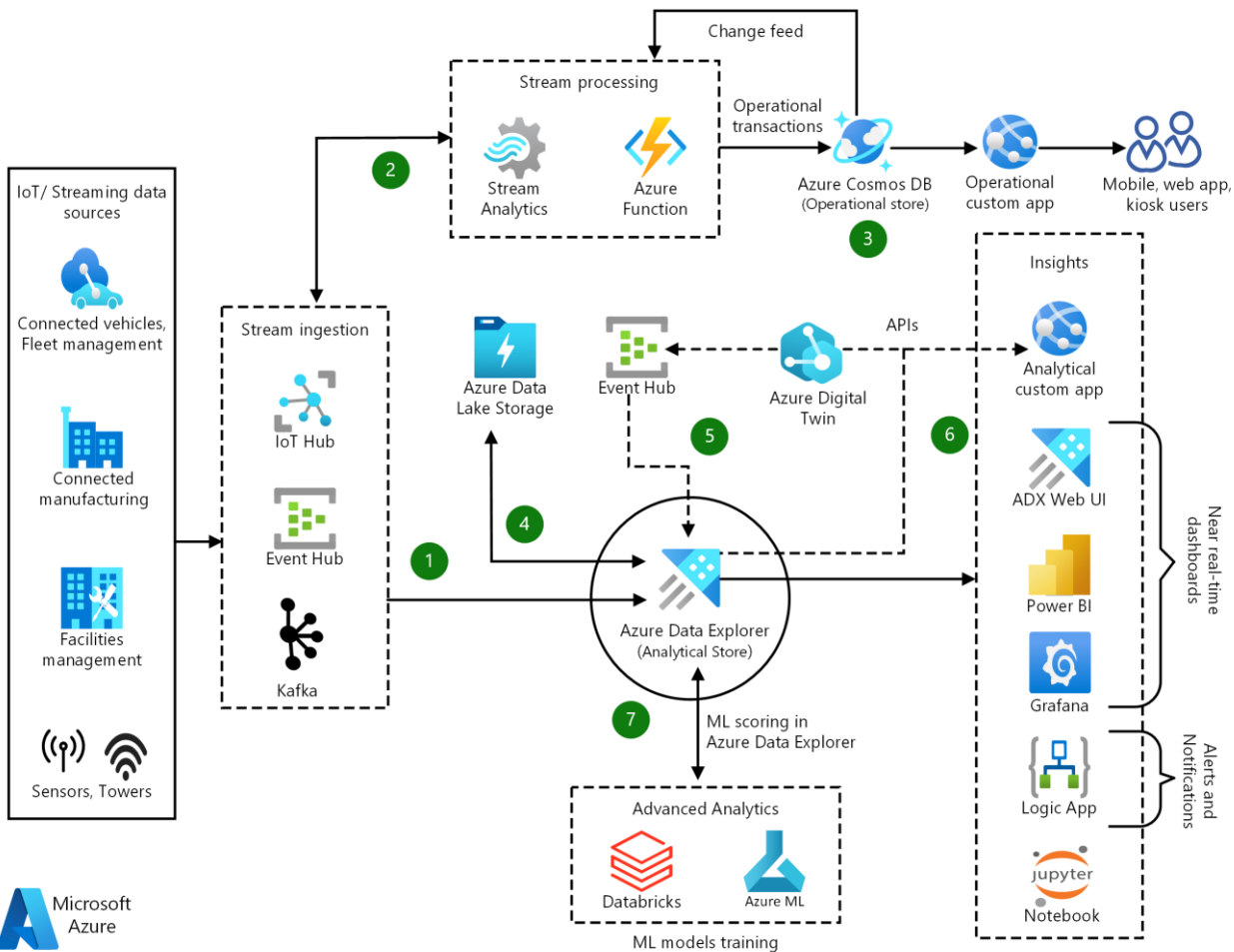
Implement Data Movement for an Azure Cosmos DB Solution

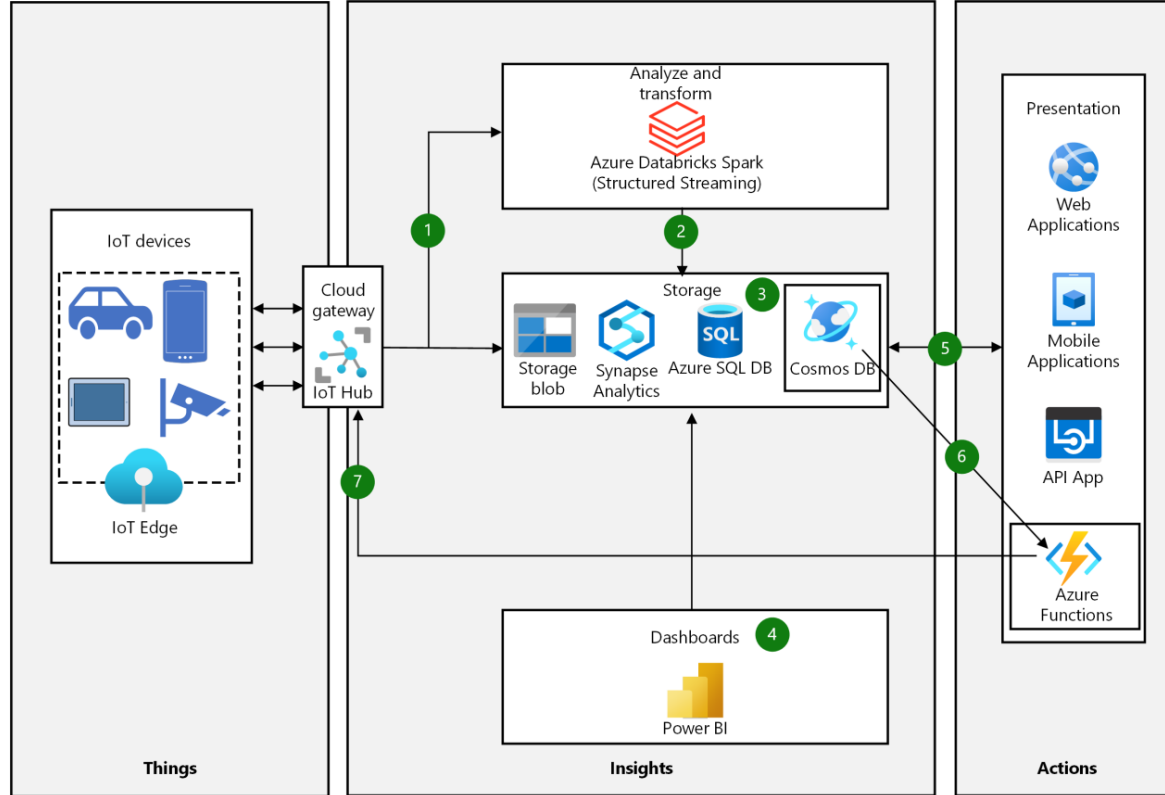
- Choose a data movement strategy
- Move data by using client SDK bulk operations
- Move data by using Azure Data Factory and Azure Synapse pipelines
- Move data by using a Kafka connector
- Move data by using Azure Stream Analytics
- Move data by using the Azure Cosmos DB Spark Connector











Implement a DevOps Process for an Azure Cosmos DB Solution

- Choose when to use declarative versus imperative operations
- Provision and manage Azure Cosmos DB resources by using ARM templates
- Migrate between standard and auto scale throughput by using PowerShell or Azure CLI
- Initiate a regional failover by using PowerShell or Azure CLI
- Maintain index policies in production by using ARM templates



The Exam

DP-420

- [Exam DP-420](#)
- [Skills measured](#)



Tip

- Download the **DP-420 study guide** [↗](#) to help you prepare for the exam
- Demo the exam experience by visiting our **Exam Sandbox** [↗](#)



Two ways to prepare

Online - Free

Instructor-led - Paid

Items in this collection



LEARNING PATH

Get started with Azure Cosmos DB SQL API

2 Modules

Intermediate

Data Engineer

Azure

Start >

+ Save



LEARNING PATH

Plan and implement Azure Cosmos DB SQL API

3 Modules

Intermediate

Developer

Cosmos DB

+ Save



LEARNING PATH

Connect to Azure Cosmos DB SQL API with the SDK

2 Modules

Intermediate

Developer

Cosmos DB



Questions in DP-420

- Number of Questions 40-60 Questions
- Questions
 - Multiple choice
 - Drag and drop
 - Scenario based
- Pass Score 700 (on a scale of 1-1000)
- See [exam sandbox](#)



Schedule exam

Exam DP-420: Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB

Languages: English, Japanese, Chinese (Simplified), Korean, German, French, Spanish, Portuguese (Brazil), Arabic (Saudi Arabia), Russian, Chinese (Traditional), Italian, Indonesian (Indonesia)

Retirement date: none

This exam measures your ability to accomplish the following technical tasks: design and implement data models; design and implement data distribution; integrate an Azure Cosmos DB solution; optimize an Azure Cosmos DB solution; and maintain an Azure Cosmos DB solution.

Schedule exam >

United States

\$165 USD*

Price based on the country or region in which the exam is proctored.

Official [practice test](#) for Designing and Implementing Cloud-Native Applications Using Microsoft Azure Cosmos DB
All objectives of the exam are covered in depth so you'll be ready for any question on the exam.

⊕ Save





Select exam options

AZ-104: Microsoft Azure Administrator

Where do you want to take your exam?



At a test center



Online at my home or office

I have a Private Access Code



Where do you want to take your exam?



At a test center



Online at my home or office

I have a Private Access Code

Prepare for your online exam at your home or office



Your computer

Use a personal computer that has a reliable webcam and internet connection.

Run [system test](#).



Your testing space

The room should be a distraction-free, private place.

See [acceptable spaces](#) and view permitted [comfort aid list](#).



Your photo ID

We'll verify your government-issued identification (ID) when you arrive for your exam.

Review [admission & ID policies](#)



What to expect

Check in for your OnVUE exam 30 minutes before your appointment time.

Watch our [short video](#) to get familiar with the process.

Questions?

Check out the [OnVUE FAQs](#) and [minimum technical requirements](#).



Cart

[Review and confirm](#) contact information to avoid issues on test day.

Description	Details	Price	Actions
		165.00	Remove

Available Products

In addition to scheduling your exam, you might be interested in the following products.



Microsoft Official Practice Test powered by MeasureUp - 30 day online access
Get a discount on available Microsoft Official Practice Test for Microsoft certification exams (Fundamentals, Role-based, or Specialty) 30-day online access.

Special offer: Regularly priced at USD 99.00! [Click here for details](#)

[More Details](#)

USD 80.00

[Add to Order](#)



It's time to test your system

Order #: 0064-8802-7606

Your appointment is confirmed! An order confirmation containing important exam day information has been sent to: zaalion@gmail.com

What's next?

Run a system test

We need to verify that the computer and internet connection you plan to use on exam day meet the [minimum requirements](#) for online testing. It'll just take 5 minutes to run:



Equipment and internet connection checks



Exam simulation

Description

Details

Order Information

Price

165.00



System Test

☐ I confirm that on my exam day I will be using this same testing space, computer, and internet connection.

Alert! Work computers generally have more restrictions that may prevent a successful test. Ensure you are not behind a corporate firewall, and shut down any **Virtual Private Networks (VPNs)** or **Virtual Machines**.

1. Copy Access Code

Click '**Copy Access Code**'.

This code will authorize you to perform a system test.

690-635-235

Copy Access Code

2. Download OnVUE

Click '**Download**'.

Download

3. Run OnVUE

Run the OnVUE application from your Downloads folder.



Course Repository

<https://github.com/zaalion/oreilly-dp-420>



O'REILLY®

Thank you!

Reza Salehi

@zaalion

