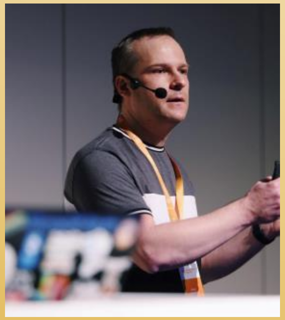
Performance Tuning For Azure Cosmos DB

Cloud Databases



Hasan Savran

SavranWeb Consulting

FUTURE DATA DRIVEN SUMMIT

Hasan Savran (He/him)

Owner

SavranWeb Consulting

- MS Data Platform MVP
- Azure Cosmos DB SME
- From Cleveland, USA
- 15+ years Web Development
- 8+ years Business Intelligence

https://h-savran.blogspot.com/

hasansavran

@savranweb

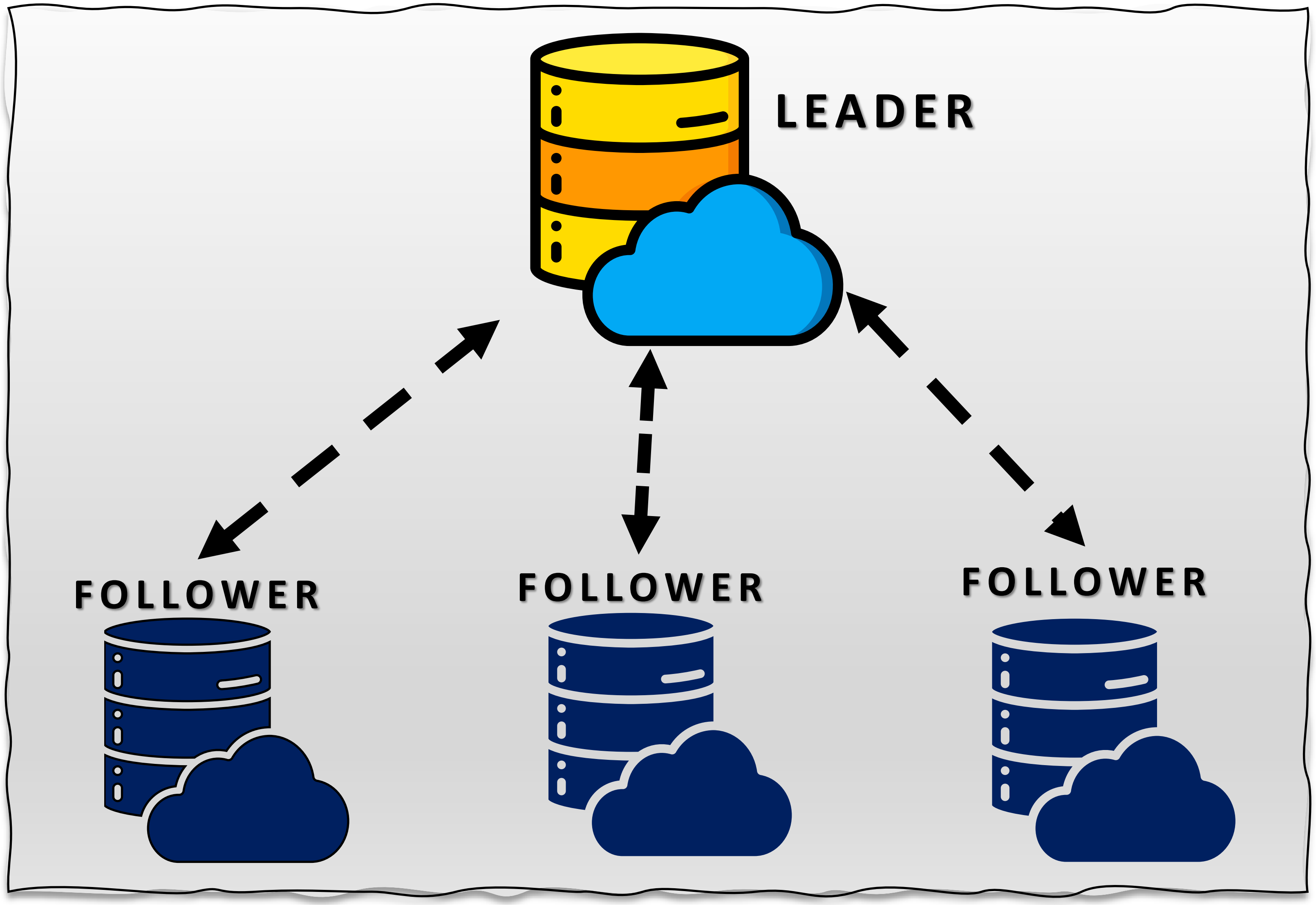
Microsoft® MVP Most Valuable Professional

COSMOS DB PHYSICAL PARTITION

Limits

50 GB Data

10,000 RU



Query Engine

Storage Engine

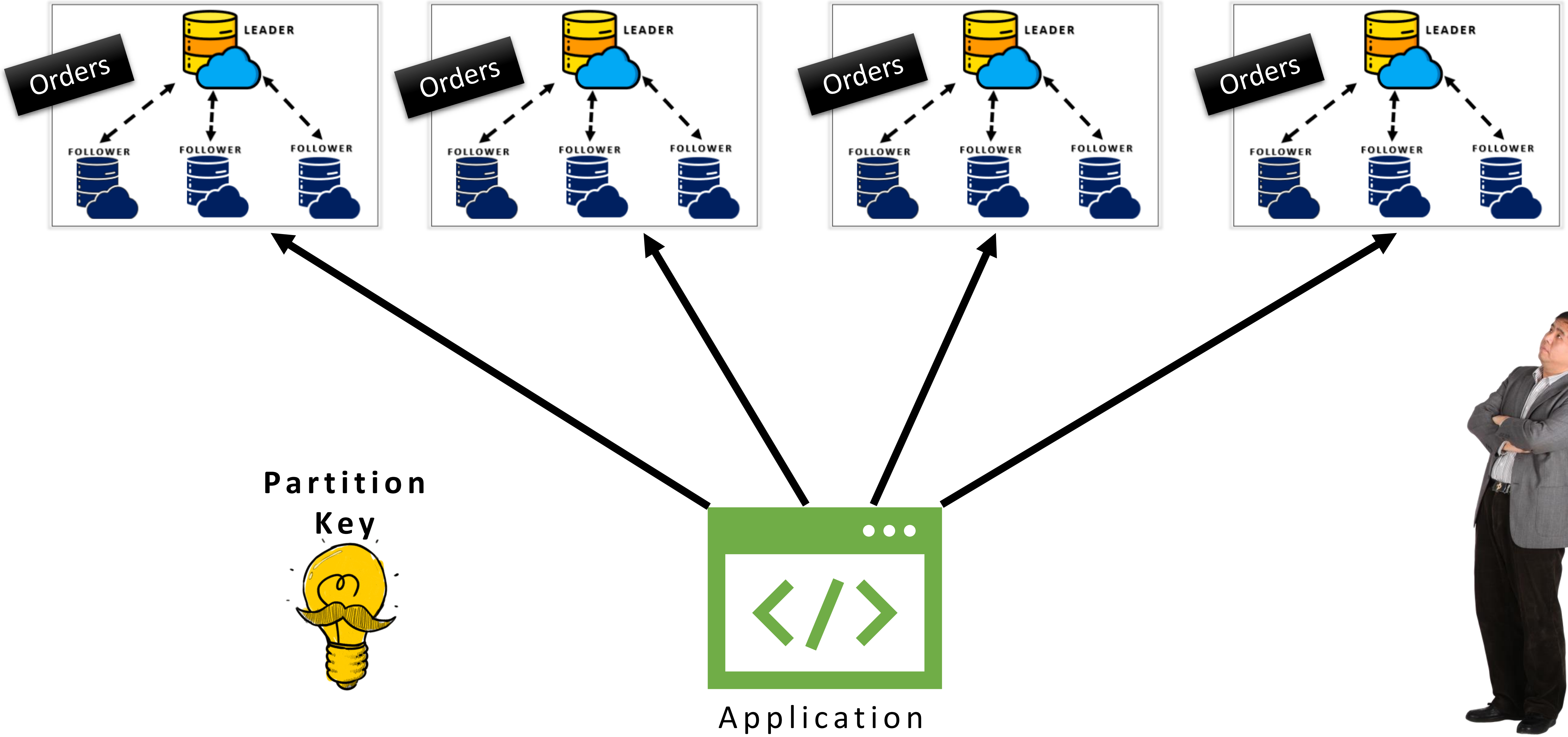
Indexing Engine

WITHOUT GLOBAL DISTRIBUTION

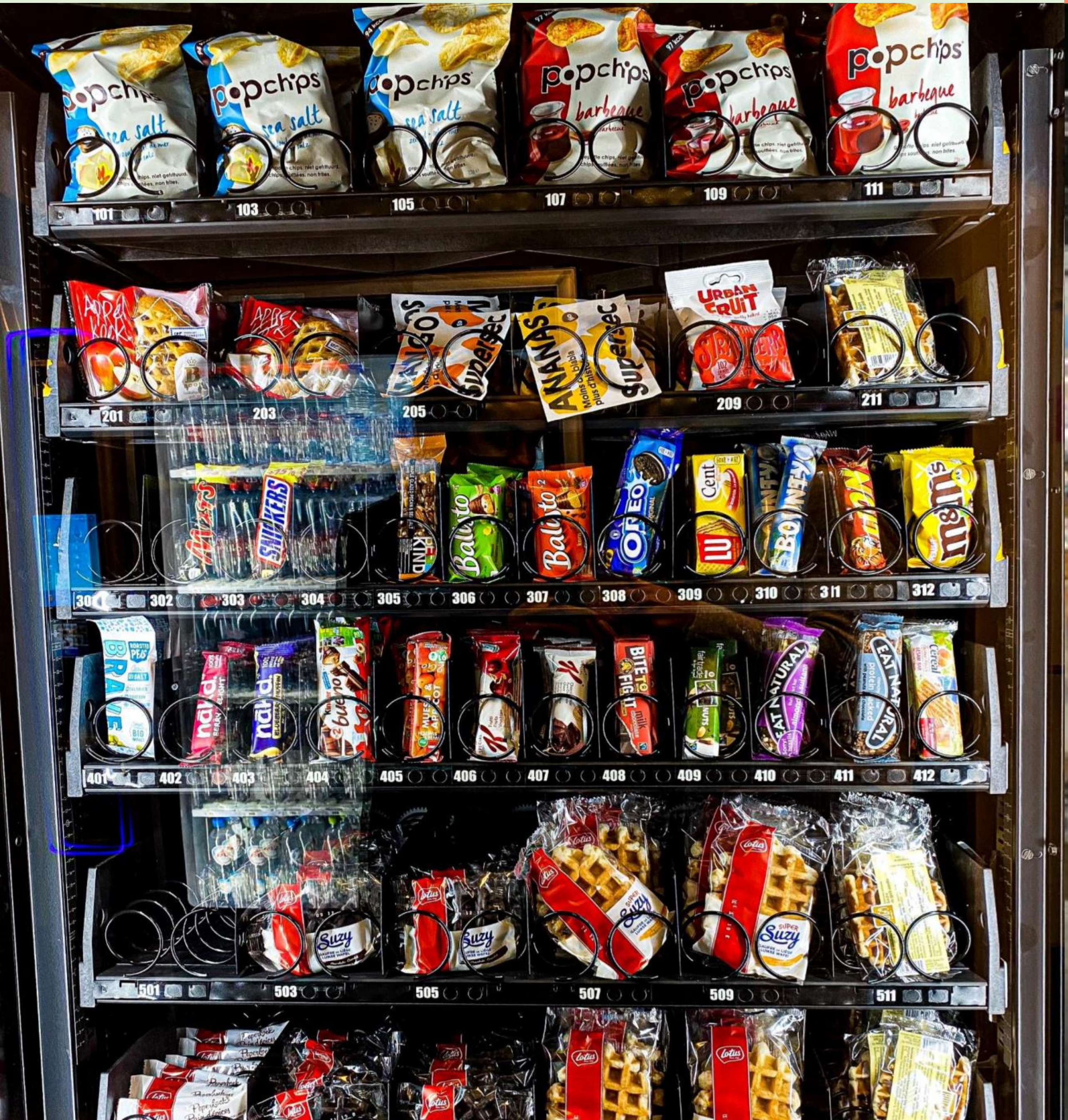
Database with 180 GB Data

OR

Database with 20,000 Request Unit



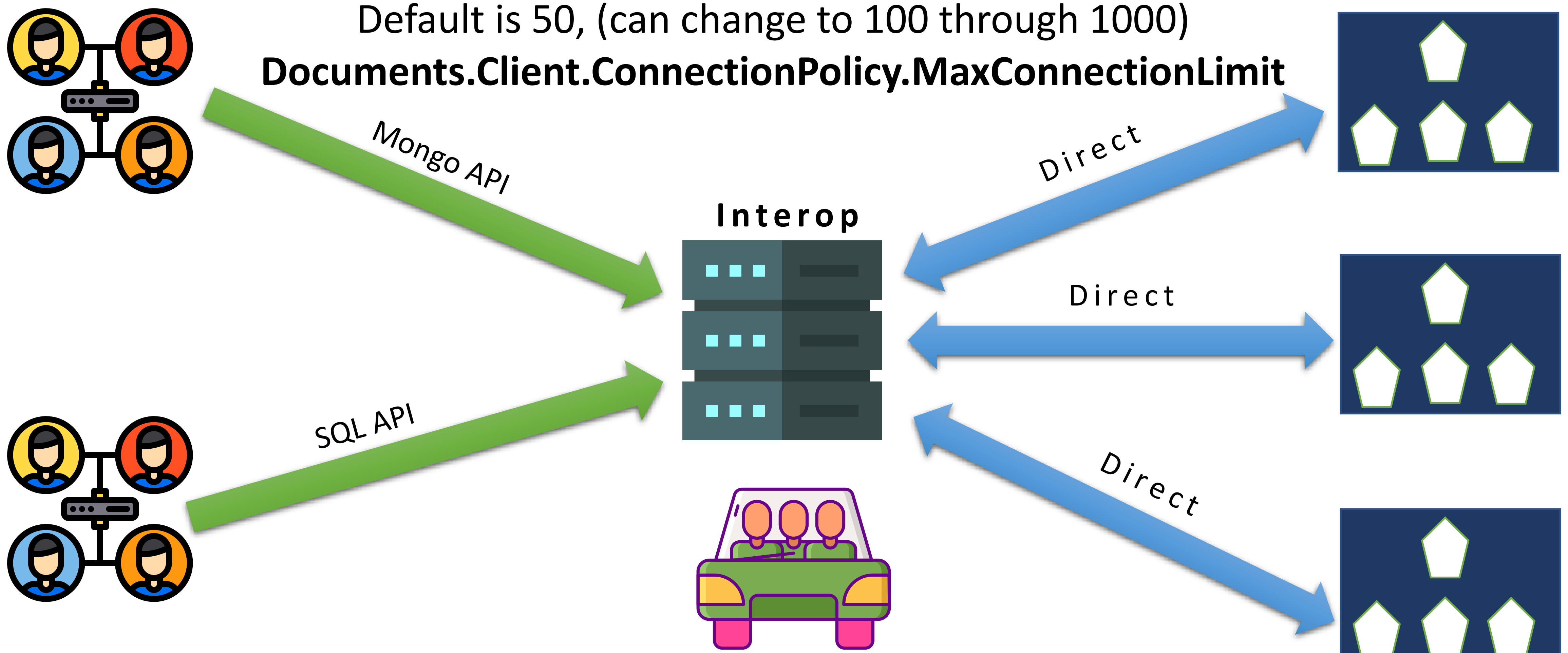
LOW RU & High Scalability



HIGH RU & BIG MESS

Connecting to Cosmos DB

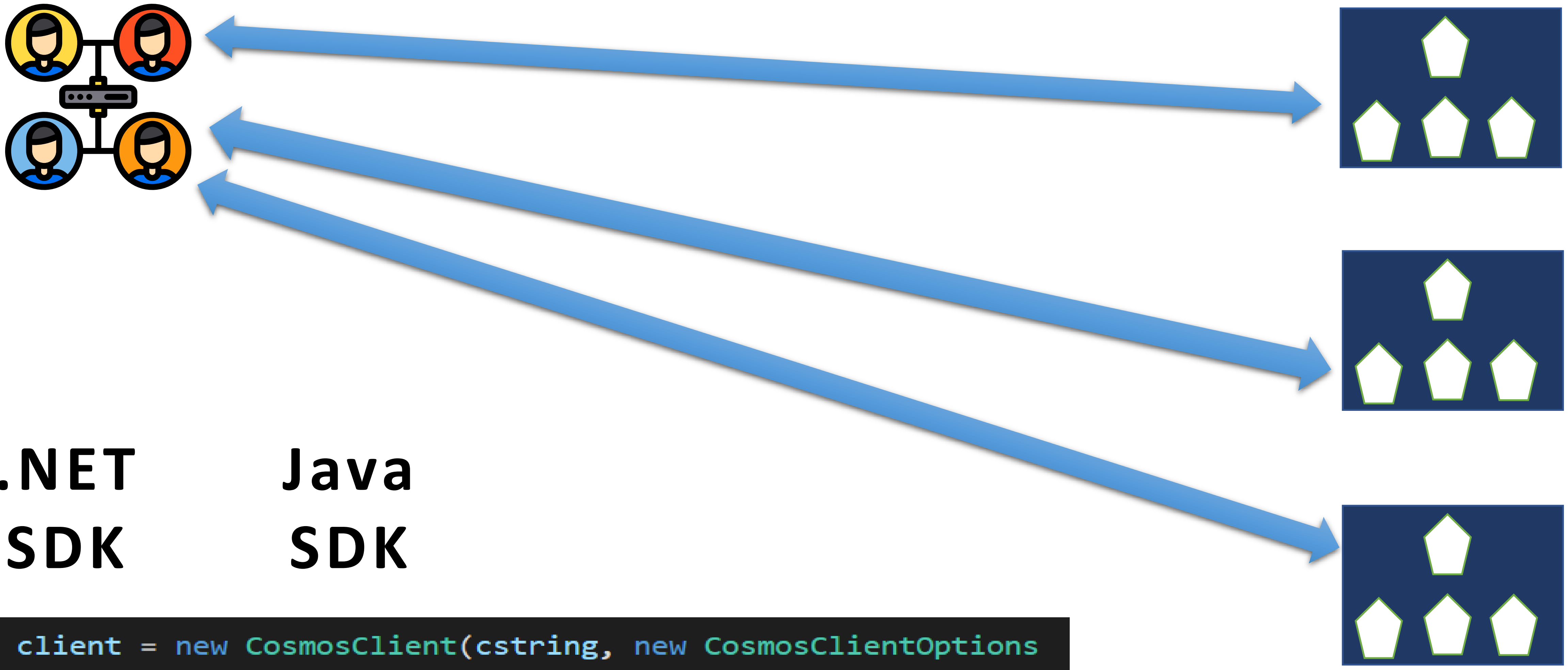
Increase **MaxConnections** per host/IP
Default is 50, (can change to 100 through 1000)
Documents.Client.ConnectionPolicy.MaxConnectionLimit



Gateway Mode uses **HTTPS & single DNS endpoint**

Connecting to Cosmos DB

Direct Connection(**TCP**)

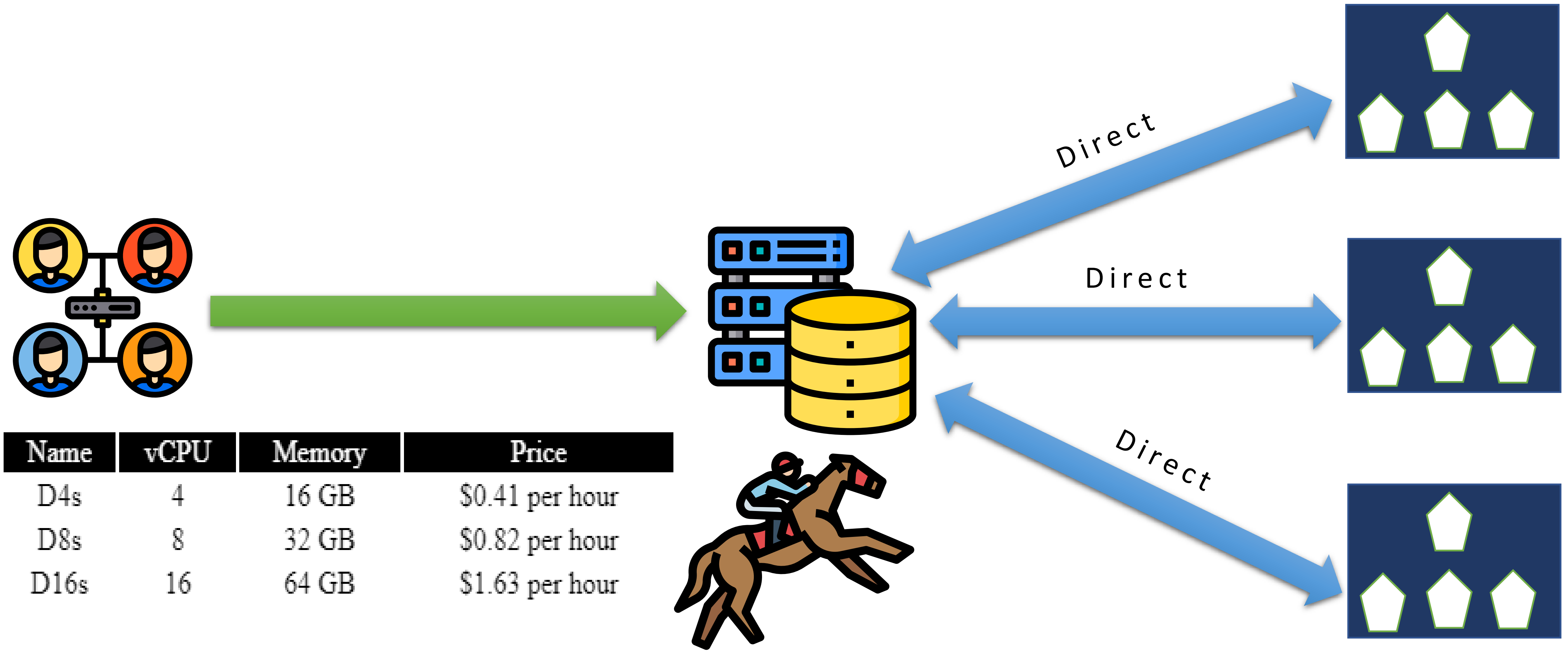


.NET SDK

Java SDK

```
var client = new CosmosClient(cstring, new CosmosClientOptions
{
    ConnectionMode = ConnectionMode.Direct
});
```

Connecting to Cosmos DB

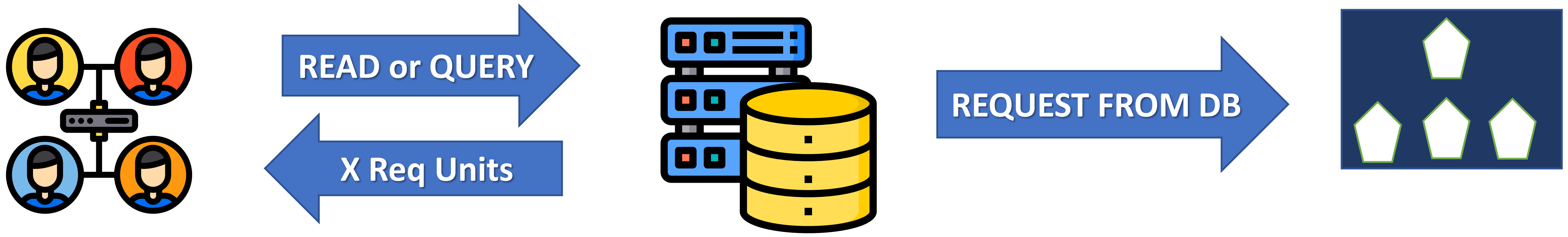


Name	vCPU	Memory	Price
D4s	4	16 GB	$0.41 per hour
D8s	8	32 GB	$0.82 per hour
D16s	16	64 GB	$1.63 per hour

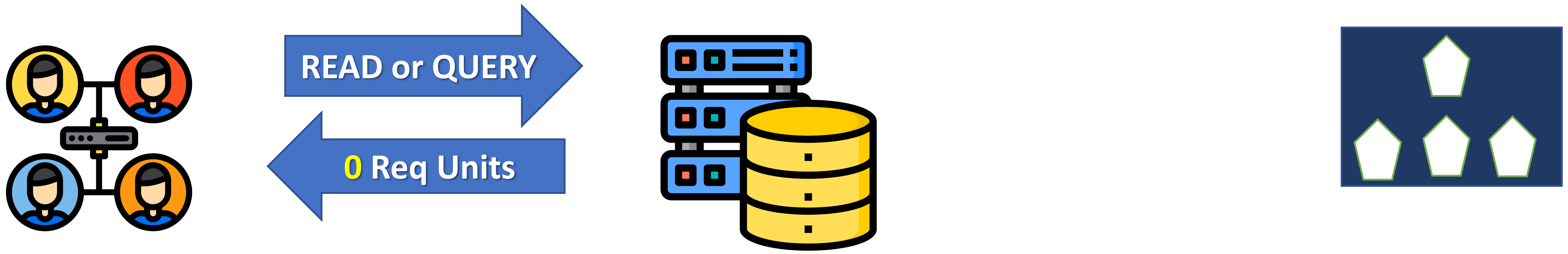
Dedicated Gateway Mode

Caching Data with Integrated Gateway

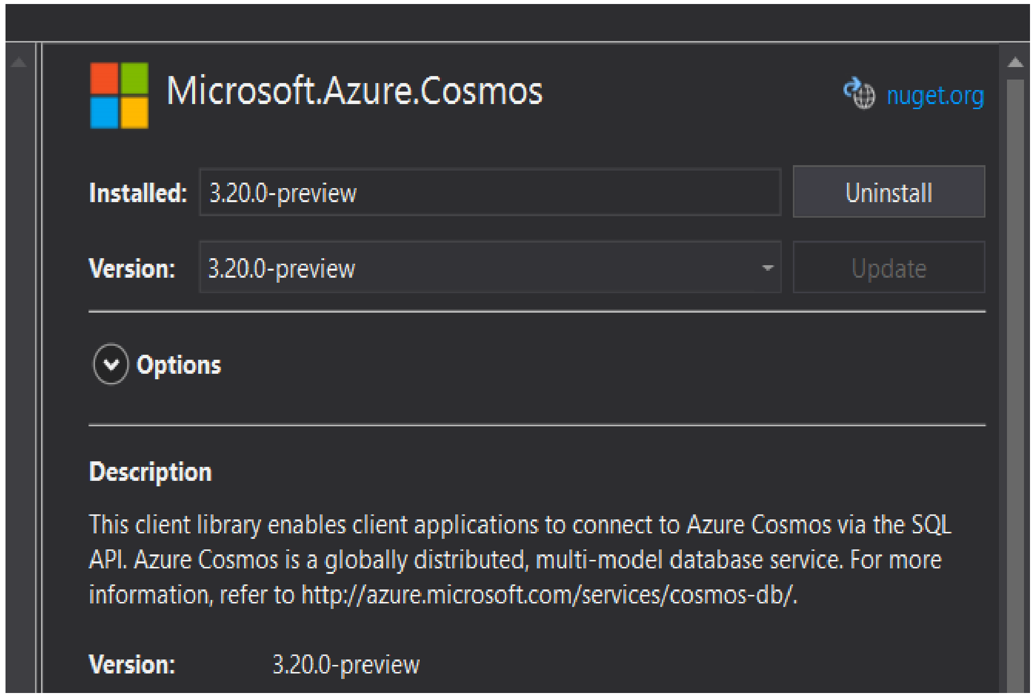
Consistency Level must be **EVENTUAL**



SAME Request comes later



Caching Data with Integrated Gateway



Microsoft.Azure.Cosmos

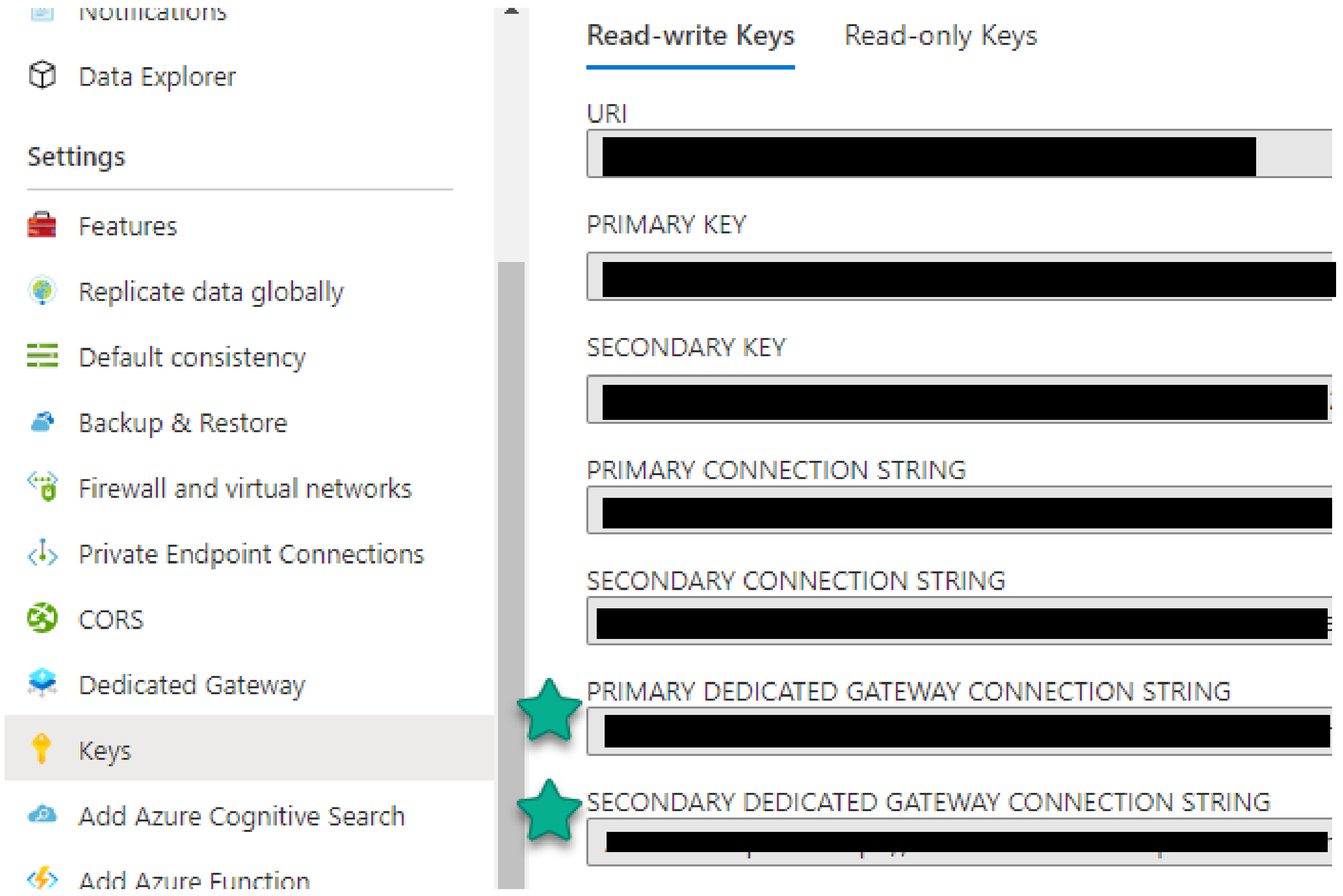
Installed: 3.20.0-preview

Version: 3.20.0-preview

Description

This client library enables client applications to connect to Azure Cosmos via the SQL API. Azure Cosmos is a globally distributed, multi-model database service. For more information, refer to http://azure.microsoft.com/services/cosmos-db/.

Version: 3.20.0-preview



URI: [redacted]

PRIMARY KEY: [redacted]

SECONDARY KEY: [redacted]

PRIMARY CONNECTION STRING: [redacted]

SECONDARY CONNECTION STRING: [redacted]

PRIMARY DEDICATED GATEWAY CONNECTION STRING: [redacted]

SECONDARY DEDICATED GATEWAY CONNECTION STRING: [redacted]

SDK 3.2.1 or later

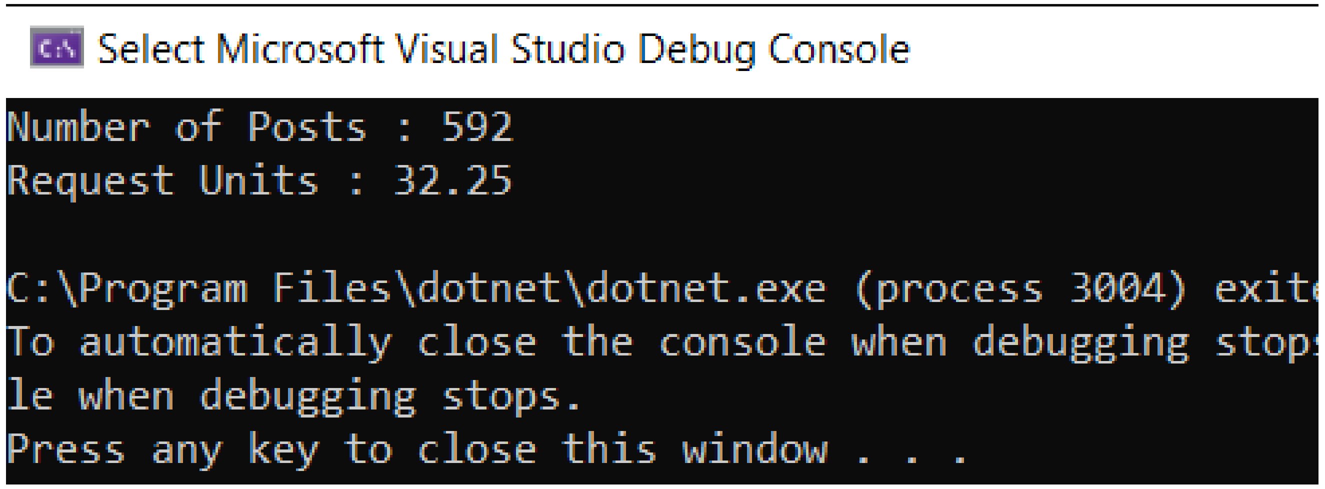
Use Dedicated Gateway Connection String

Caching Data with Integrated Gateway

```
static async Task<List<StackOverflowPost>> TestCaching(int postid=0)
{
     var cosmosClient = new CosmosClient(connectionString,
         new CosmosClientOptions { ConnectionMode= ConnectionMode.Gateway});
     Container container = cosmosClient.GetContainer("Stackoverflow", "Posts");
     var cmd = "SELECT * FROM Posts o WHERE o.PostId < 500";
     var query = new QueryDefinition(cmd);
     var queryResultSetIterator = container.GetItemQueryIterator<StackOverflowPost>(query,
          requestOptions: new QueryRequestOptions
         {
             ConsistencyLevel = ConsistencyLevel.Eventual,
             DedicatedGatewayRequestOptions = new DedicatedGatewayRequestOptions
             {
                 MaxIntegratedCacheStaleness = TimeSpan.FromMinutes(30)
             }
         }
          );
     var posts = new List<StackOverflowPost>();
     double rq = 0;
     try
       {
         while (queryResultSetIterator.HasMoreResults)
         {
```

Caching Data with Integrated Gateway

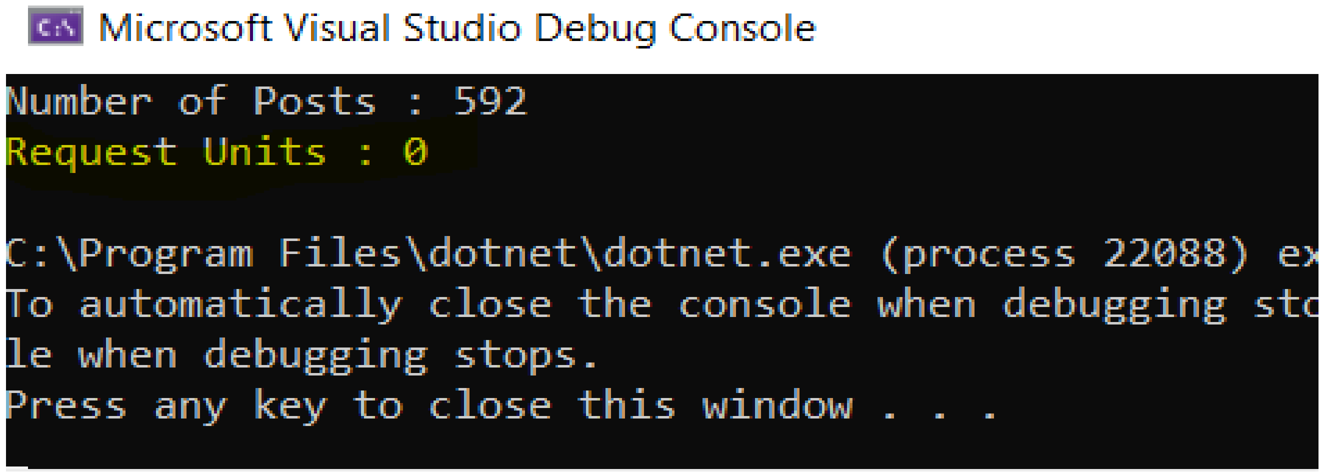
FIRST TIME



```
Number of Posts : 592
Request Units : 32.25

C:\Program Files\dotnet\dotnet.exe (process 3004) exite
To automatically close the console when debugging stop
le when debugging stops.
Press any key to close this window . . .
```

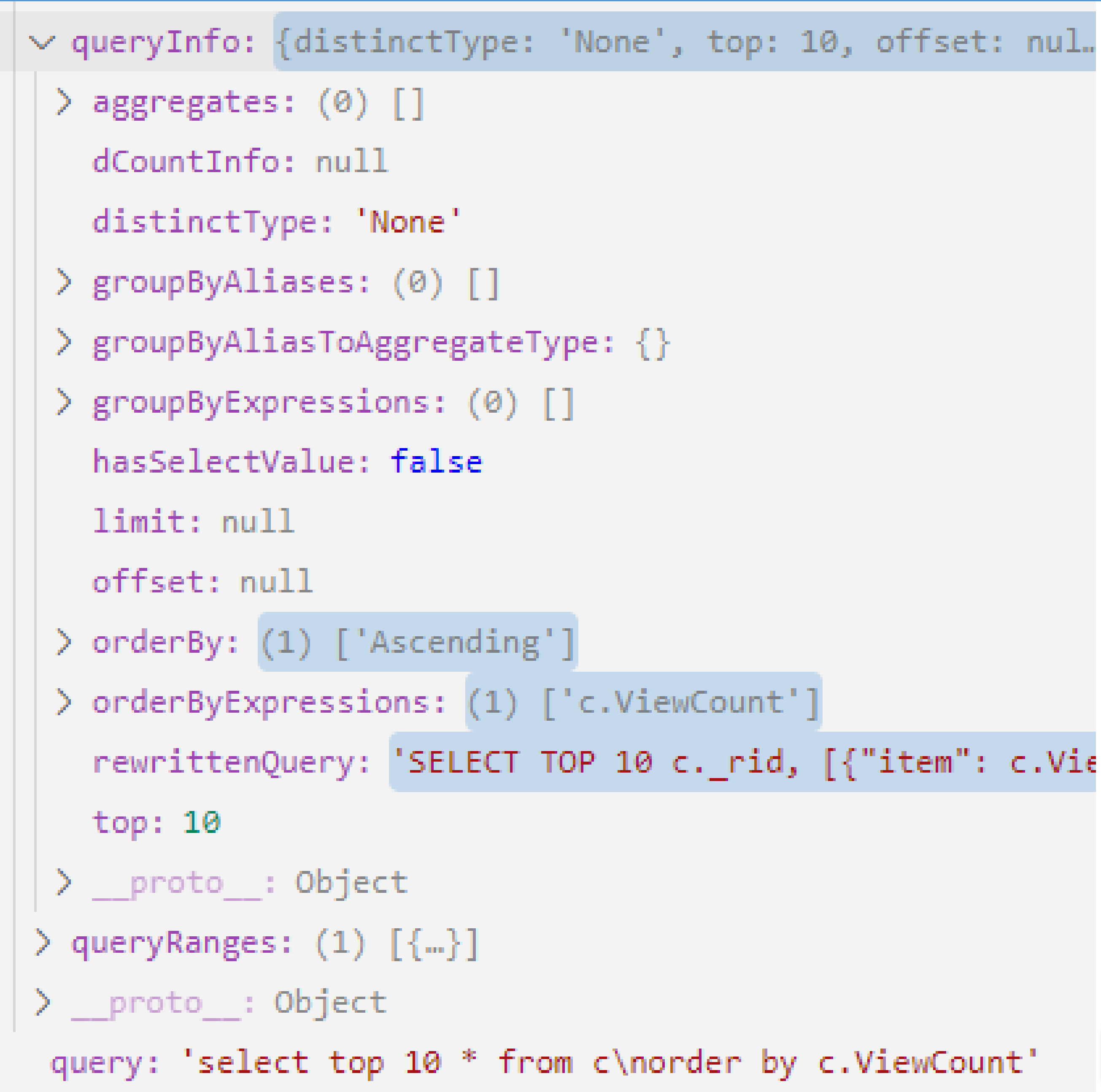
LATER



```
Number of Posts : 592
Request Units : 0

C:\Program Files\dotnet\dotnet.exe (process 22088) ex
To automatically close the console when debugging sto
le when debugging stops.
Press any key to close this window . . .
```

SDK



```
∨ queryInfo: {distinctType: 'None', top: 10, offset: nul…
  > aggregates: (0) []
    dCountInfo: null
    distinctType: 'None'
  > groupByAliases: (0) []
  > groupByAliasToAggregateType: {}
  > groupByExpressions: (0) []
    hasSelectValue: false
    limit: null
    offset: null
  > orderBy: (1) ['Ascending']
  > orderByExpressions: (1) ['c.ViewCount']
    rewrittenQuery: 'SELECT TOP 10 c._rid, [{"item": c.Vie
    top: 10
  > __proto__: Object
> queryRanges: (1) [{…}]
> __proto__: Object
 query: 'select top 10 * from c\norder by c.ViewCount'
```

SELECT TOP 10 c._rid,
[{"item": c.ViewCount}] AS orderByItems, c AS payload
FROM c
WHERE ({documentdb-formattableorderbyquery-filter})
ORDER BY c.ViewCount

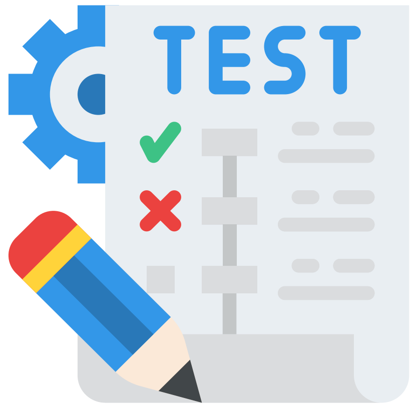
.NET & Hosting Recommendations

- Use 64-bit Windows host processing
- ServiceInterop.dll to parse and optimize queries locally for SQL SDK

***runtimeconfig.json* file:**

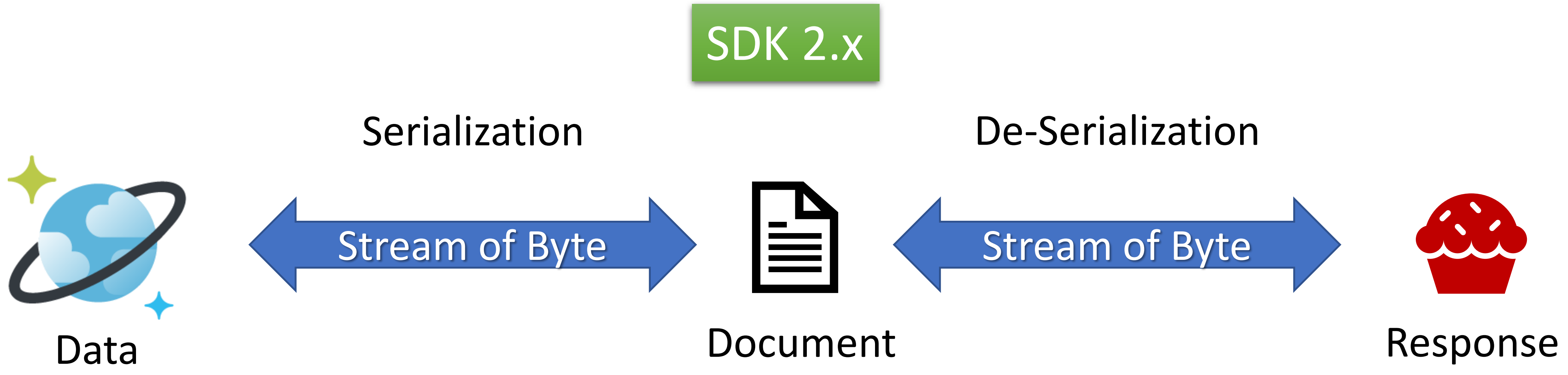
```
{
  "runtimeOptions": {
   "configProperties": {
    "System.GC.Server": true
   }
} }
```

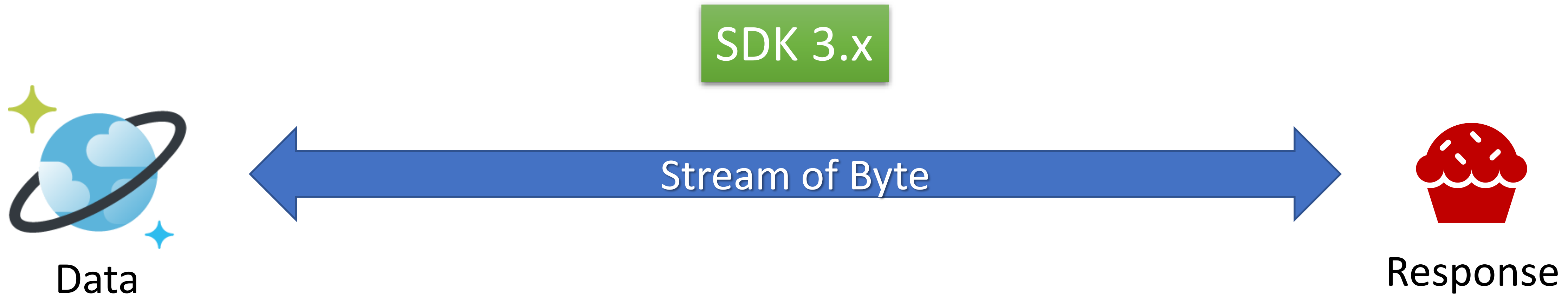
High CPU can cause increased latency and request timeout exceptions.

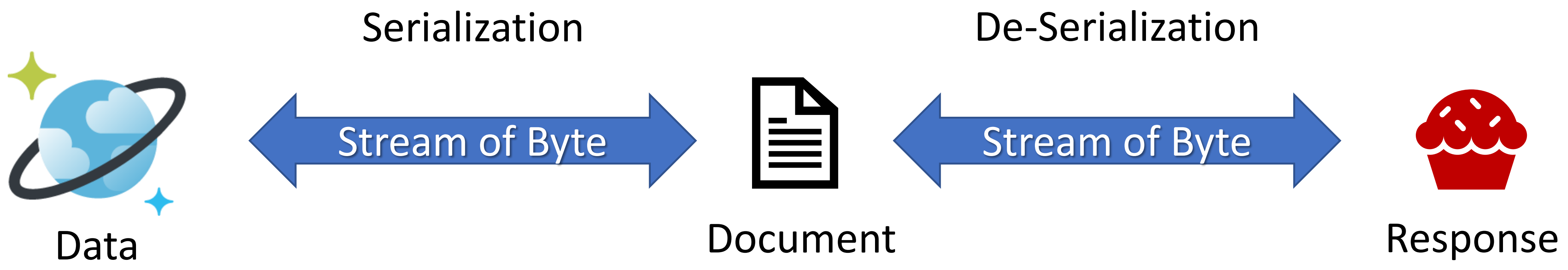


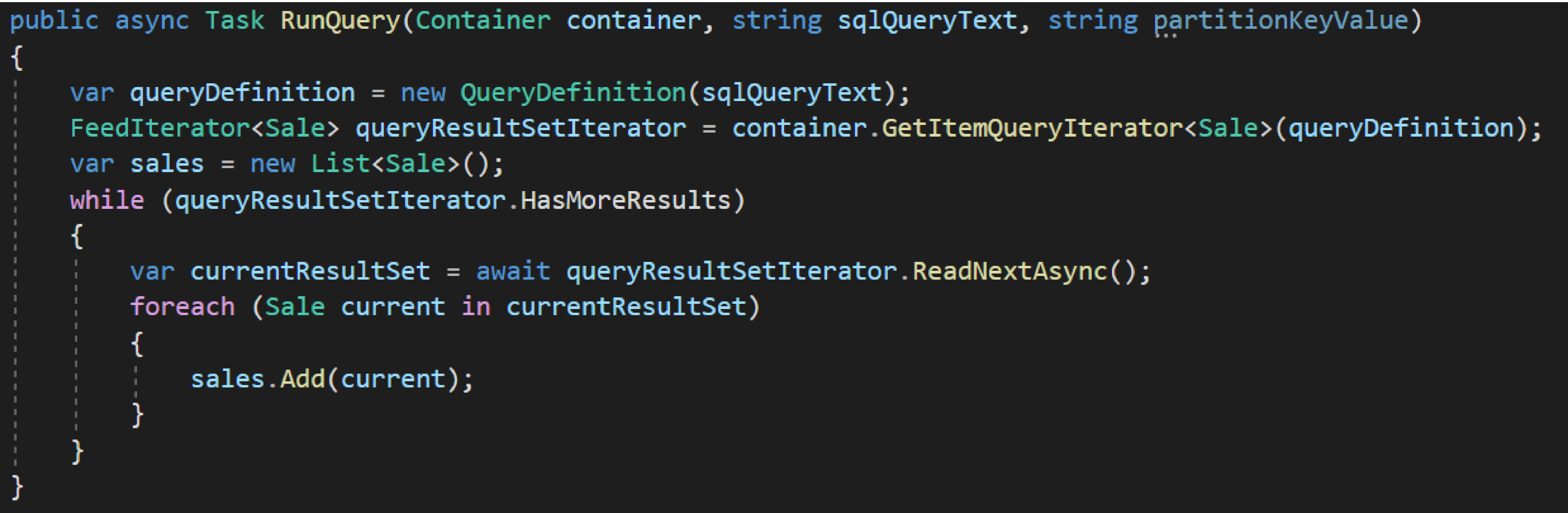
50.000 R/U

Use STREAM API in SDK 3



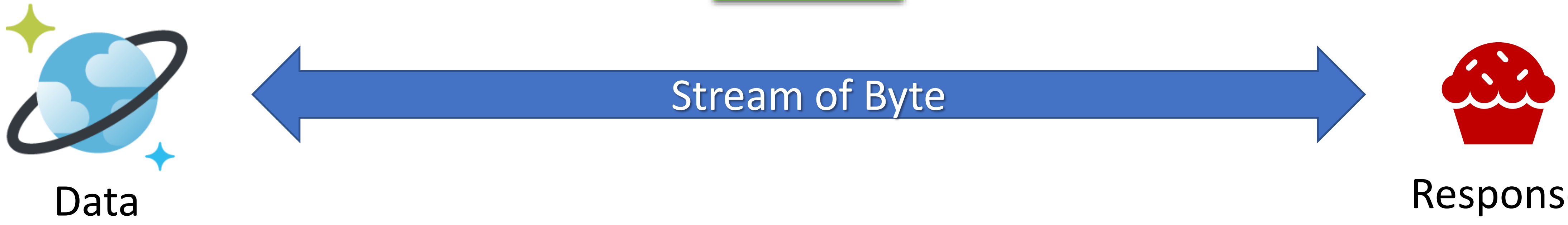


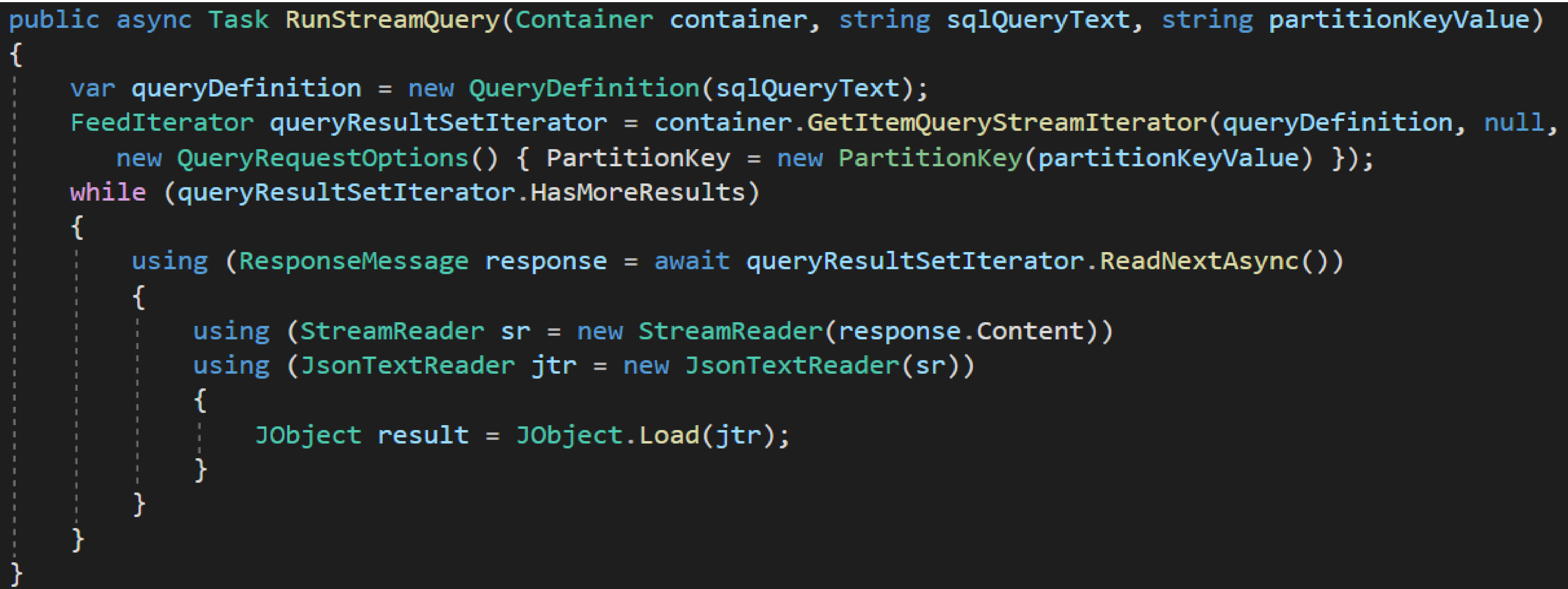




```
public async Task RunQuery(Container container, string sqlQueryText, string partitionKeyValue)
{
    var queryDefinition = new QueryDefinition(sqlQueryText);
    FeedIterator<Sale> queryResultSetIterator = container.GetItemQueryIterator<Sale>(queryDefinition);
    var sales = new List<Sale>();
    while (queryResultSetIterator.HasMoreResults)
    {
        var currentResultSet = await queryResultSetIterator.ReadNextAsync();
        foreach (Sale current in currentResultSet)
        {
            sales.Add(current);
        }
    }
}
```

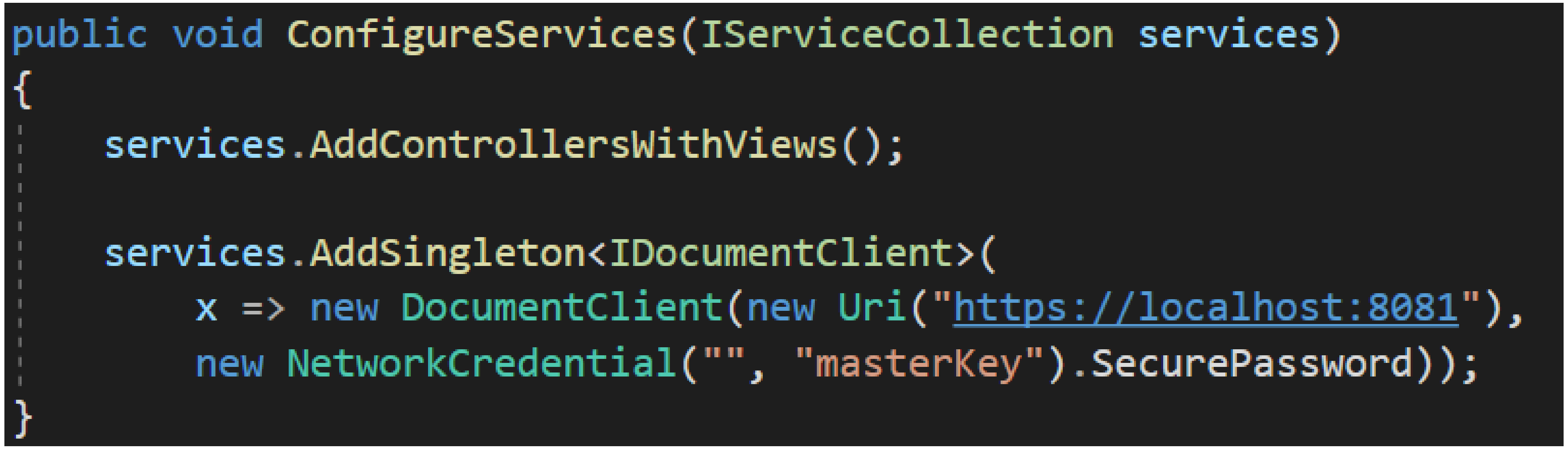
SDK 3.x





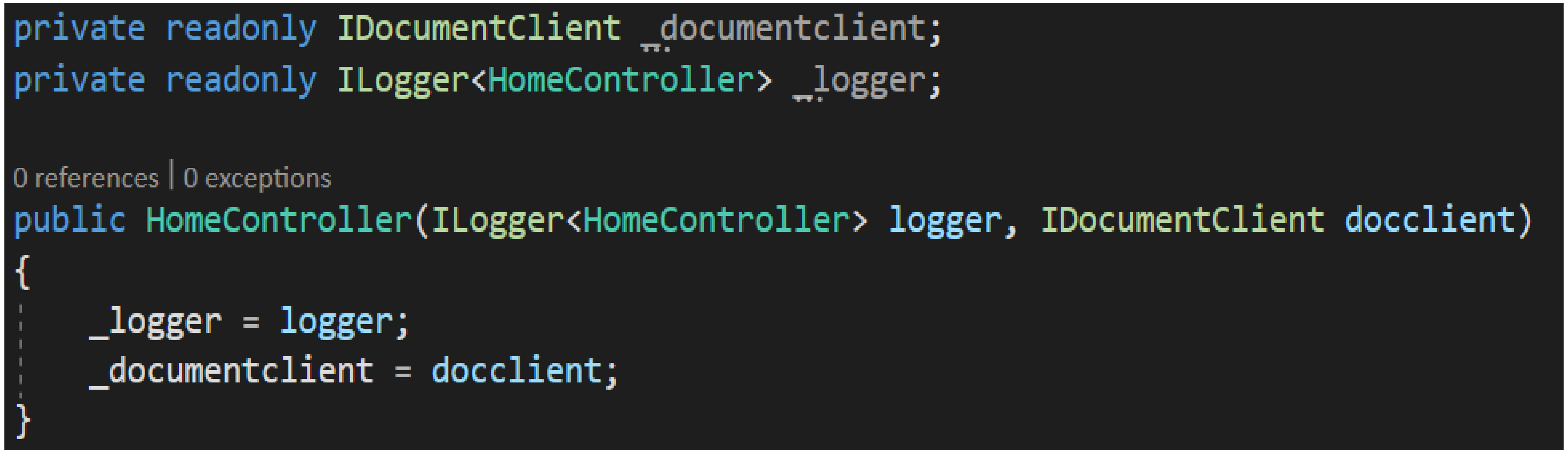
```
public async Task RunStreamQuery(Container container, string sqlQueryText, string partitionKeyValue)
{
    var queryDefinition = new QueryDefinition(sqlQueryText);
    FeedIterator queryResultSetIterator = container.GetItemQueryStreamIterator(queryDefinition, null,
       new QueryRequestOptions() { PartitionKey = new PartitionKey(partitionKeyValue) });
    while (queryResultSetIterator.HasMoreResults)
    {
        using (ResponseMessage response = await queryResultSetIterator.ReadNextAsync())
        {
            using (StreamReader sr = new StreamReader(response.Content))
            using (JsonTextReader jtr = new JsonTextReader(sr))
            {
                JObject result = JObject.Load(jtr);
            }
        }
    }
}
```

DocumentClient



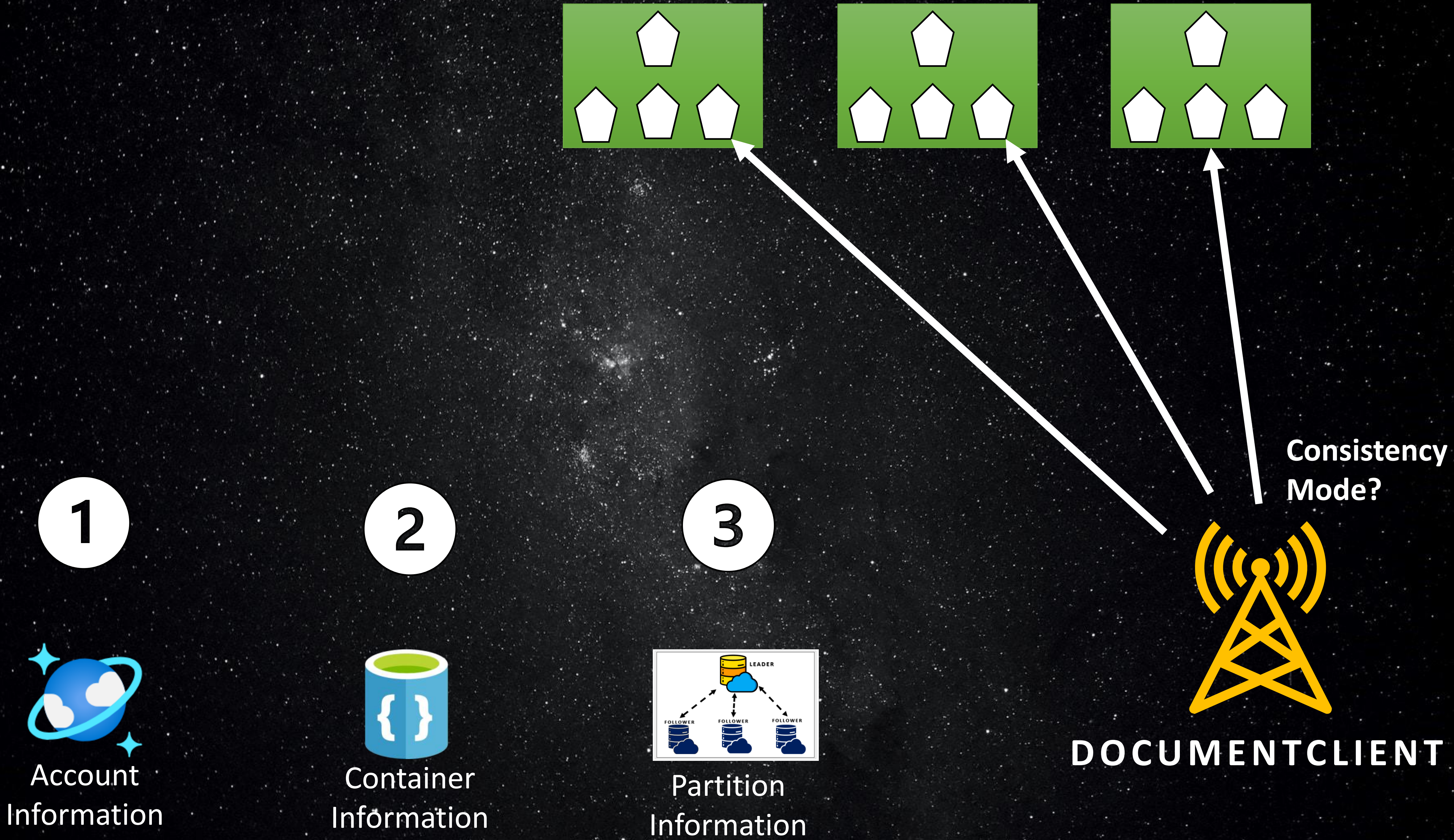
```csharp
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddSingleton<IDocumentClient>(
        x => new DocumentClient(new Uri("https://localhost:8081"),
        new NetworkCredential("", "masterKey").SecurePassword));
}
```

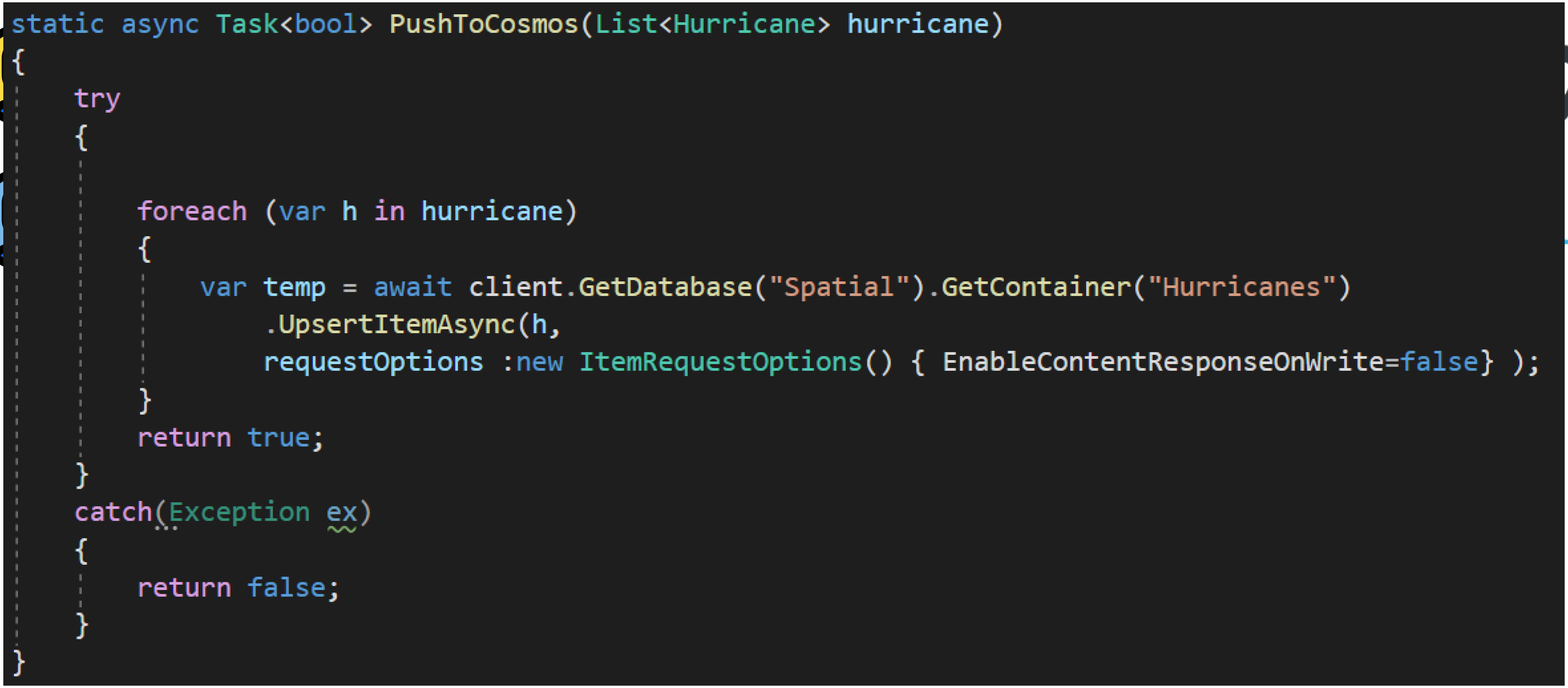


```csharp
private readonly IDocumentClient _documentclient;
private readonly ILogger<HomeController> _logger;

0 references | 0 exceptions
public HomeController(ILogger<HomeController> logger, IDocumentClient docclient)
{
    _logger = logger;
    _documentclient = docclient;
}
```


1
2
3
Account Information
Container Information
LEADER
FOLLOWER
FOLLOWER
FOLLOWER
Partition Information
Consistency Mode?
DOCUMENTCLIENT

Optimize bandwidth



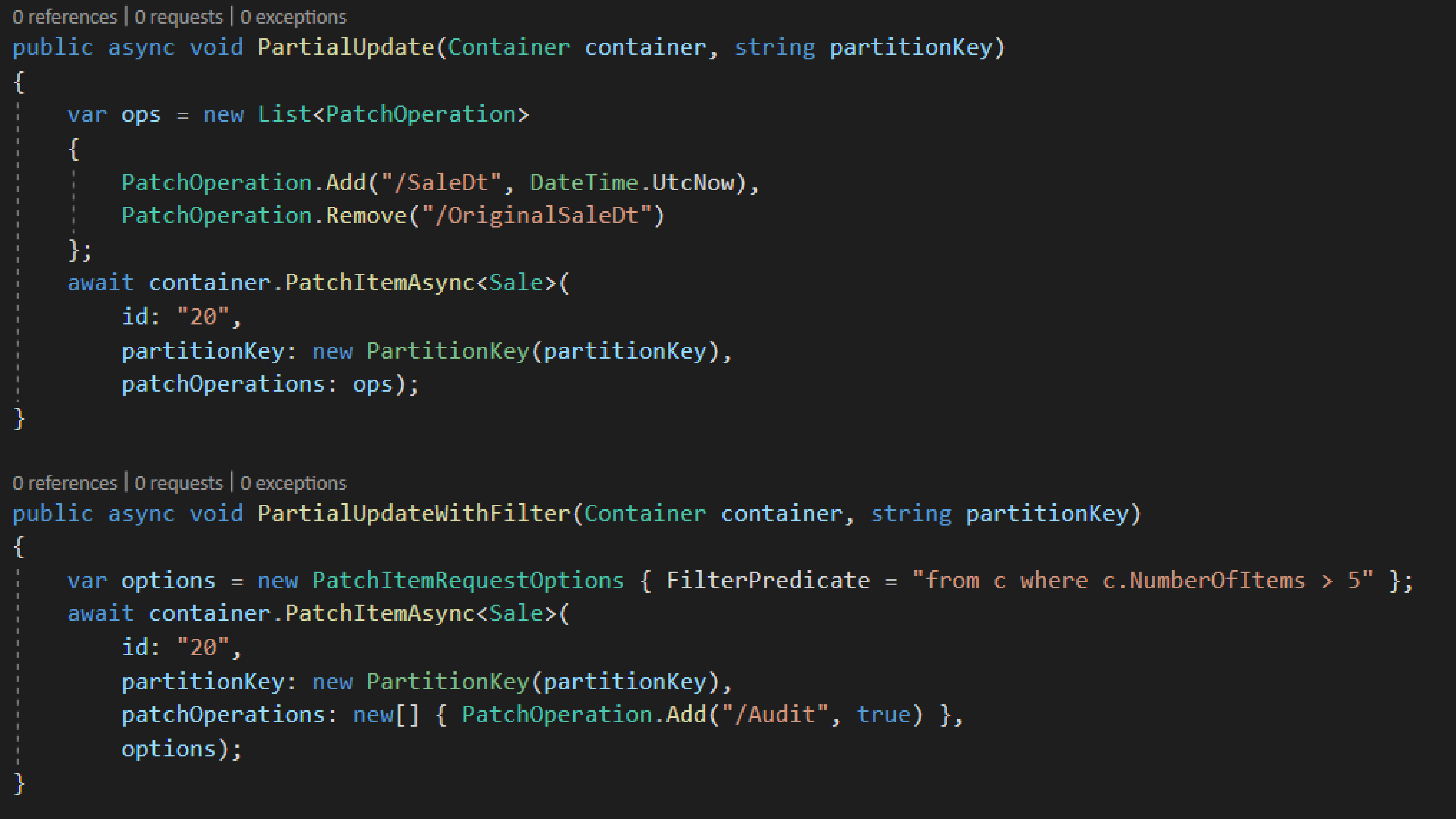
```
static async Task<bool> PushToCosmos(List<Hurricane> hurricane)
{
    try
    {

        foreach (var h in hurricane)
        {
            var temp = await client.GetDatabase("Spatial").GetContainer("Hurricanes")
                .UpsertItemAsync(h,
                requestOptions :new ItemRequestOptions() { EnableContentResponseOnWrite=false} );
        }
        return true;
    }
    catch(Exception ex)
    {
        return false;
    }
}
```

Partial Document Update

- Combine multiple operations
- Conditional Updates with SQL-like filters
- Supports Transactions with Transactional batch
- Conflict Resolution Support

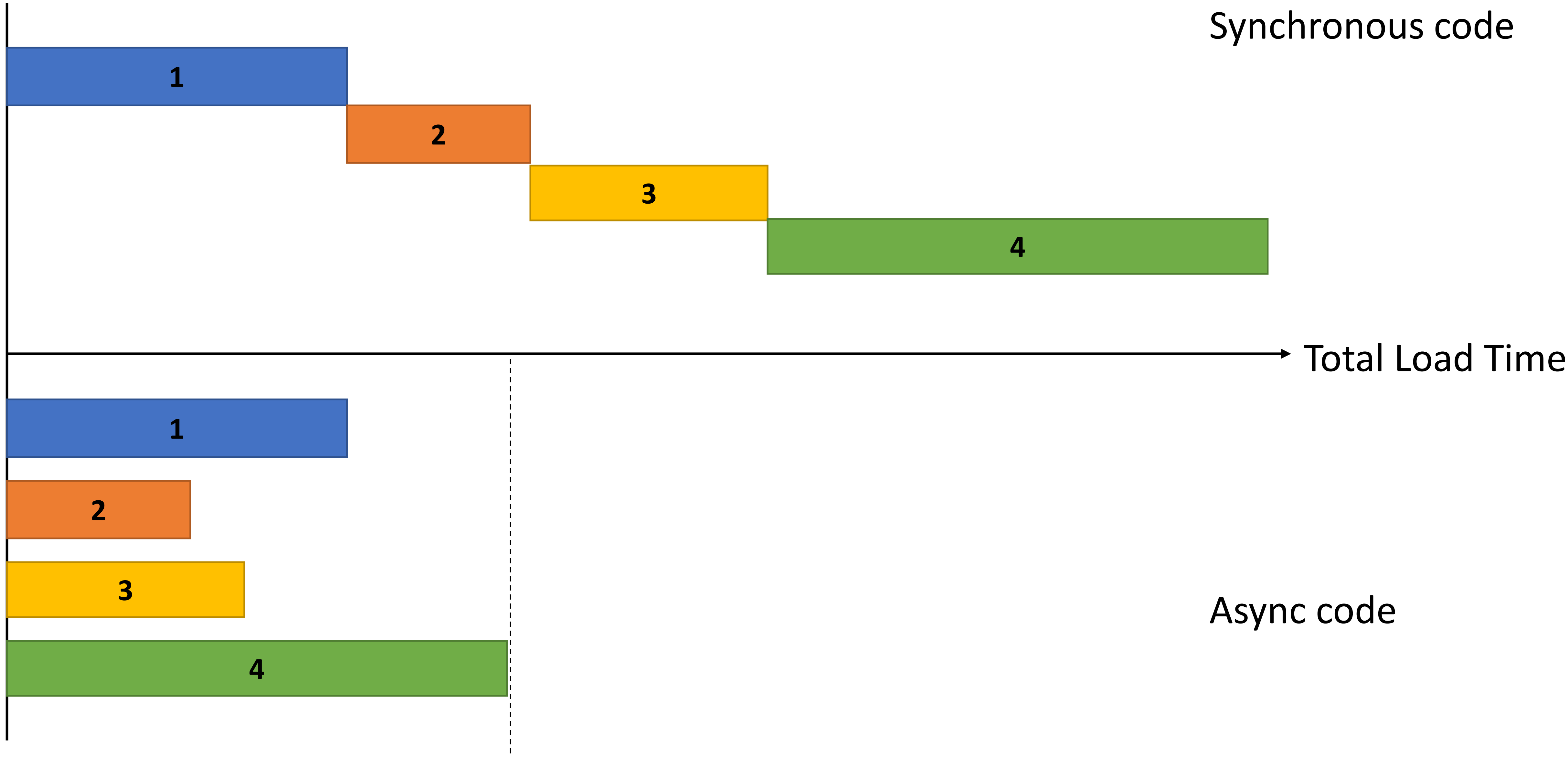
PARTIAL DOCUMENT OPERATIONS	
Add	Creates a new element
Set	Updates an element or creates one if it does not exist.
Replace	Updates an element only if exists.
Remove	Deletes an existing element
Increment	Increases or decreases by specified value



```
0 references | 0 requests | 0 exceptions
public async void PartialUpdate(Container container, string partitionKey)
{
    var ops = new List<PatchOperation>
    {
        PatchOperation.Add("/SaleDt", DateTime.UtcNow),
        PatchOperation.Remove("/OriginalSaleDt")
    };
    await container.PatchItemAsync<Sale>(
        id: "20",
        partitionKey: new PartitionKey(partitionKey),
        patchOperations: ops);
}

0 references | 0 requests | 0 exceptions
public async void PartialUpdateWithFilter(Container container, string partitionKey)
{
    var options = new PatchItemRequestOptions { FilterPredicate = "from c where c.NumberOfItems > 5" };
    await container.PatchItemAsync<Sale>(
        id: "20",
        partitionKey: new PartitionKey(partitionKey),
        patchOperations: new[] { PatchOperation.Add("/Audit", true) },
        options);
}
```

Avoid Blocking Calls

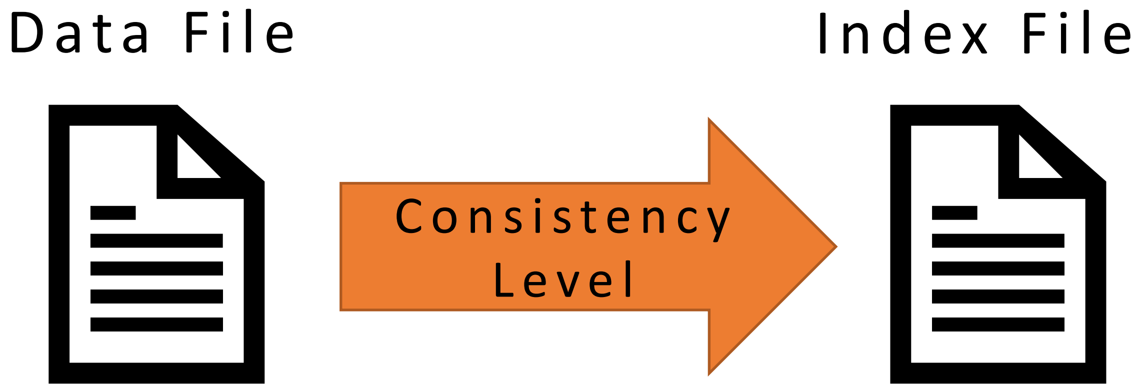


Indexing Policies

Available Indexing Modes

CONSISTENT **LAZY** **NONE**

COSMOS DB INDEXES EVERYTHING



Index Types

RANGE INDEX

- Equality Queries
- Equality matches
- Range Queries
- String system functions
- ORDER BY
- JOIN

SPATIAL INDEX

- Spatial functions
- Point
- LineString
- Polygon
- MultiPolygon

COMPOSITE INDEX

- ORDER BY multiple properties
- Filter and ORDER BY
- Filter on multiple properties

Composite Indexes

- Order BY clause with 2 or more properties requires Composite Index
- Optimize queries that have filters on multiple properties

A Composite Index can only optimize a **single** range filter or system function.
All Composite Indexed properties must be in the query's filter.
Property with a range filter must be defined **last** in the composite index.

Range Filters

Age **>** 10	Age **<** 10	Age **>=** 10	Age **<=** 10	Age **!=** 10

Modify Indexes

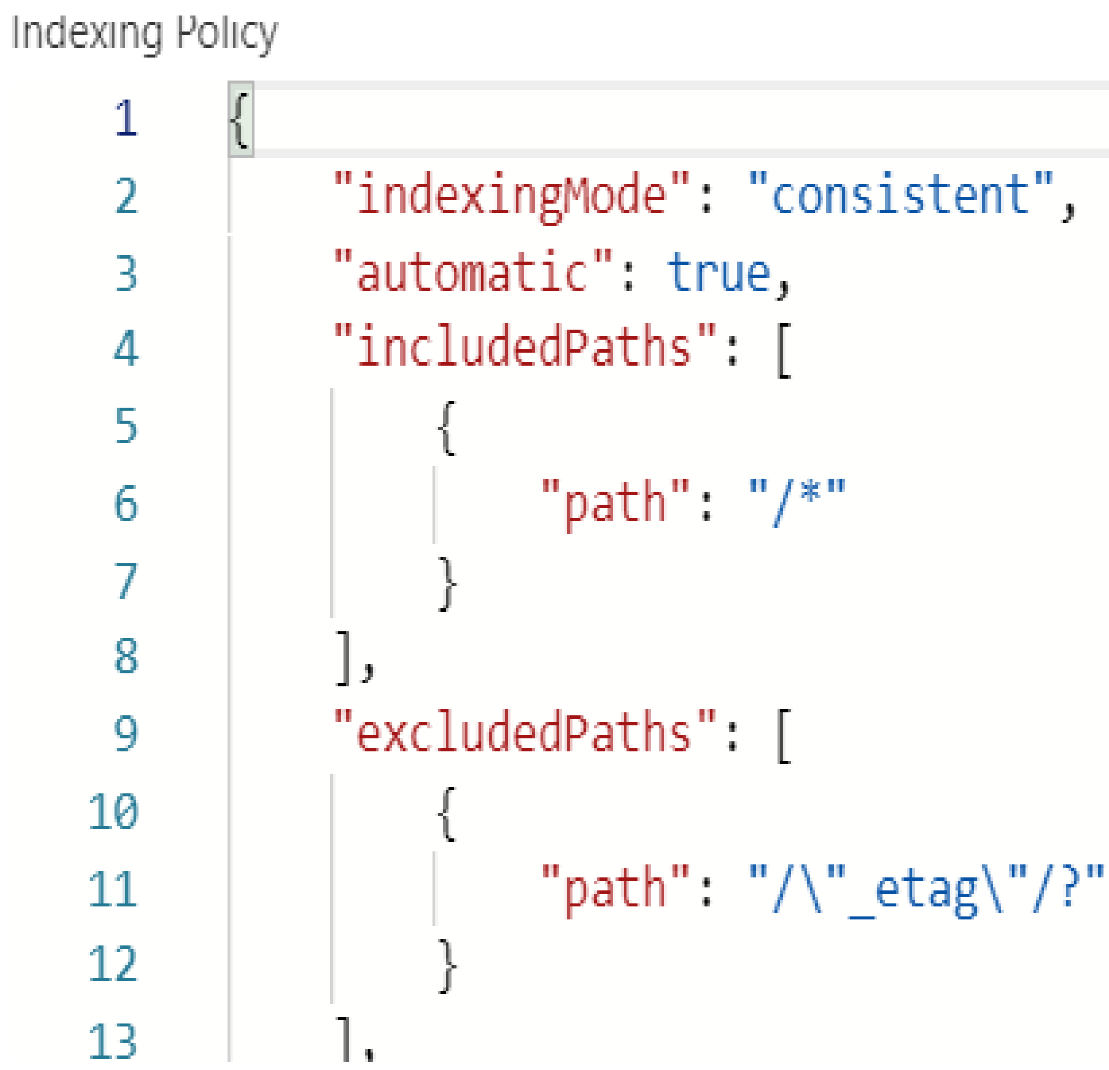
Exclude unused paths will

- Improve write performance
- Reduce R/U charges on write operations
- Reduce index storage

Do all indexing changes in one policy modification.

Do not use Lazy Indexing Mode

Take advantage of Composite Indexes



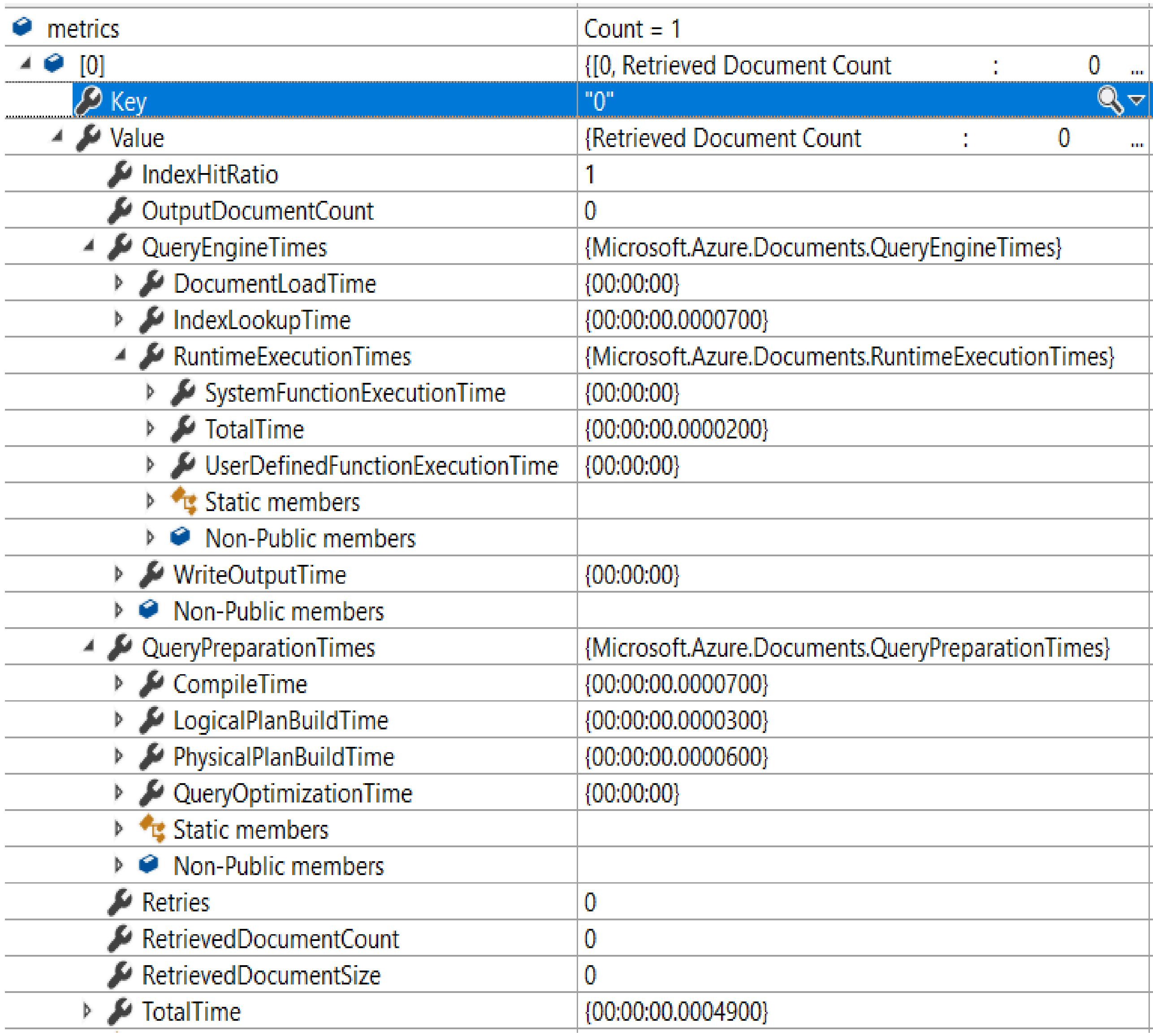
Indexing Policy

```
{
    "indexingMode": "consistent",
    "automatic": true,
    "includedPaths": [
        {
            "path": "/*"
        }
    ],
    "excludedPaths": [
        {
            "path": "/\"_etag\"/?"
        }
    ],
```

Provisioned	Auto Scale	Serverless
Daily used Applications	Infrequently Used Applications	Infrequently Used Applications
Predictable/Stable Workloads	Unpredictable/Critical Workloads	Limited Workloads
Any Traffic	Any Traffic	Light Traffic / Long Idle times
All Types of Applications	New Applications	Proof of Concept Applications
Production friendly	Dev & Test Environments	Dev & Test Environments
Minimum: 400 RU/s	Minimum: 400 – 4000 RU/s	No Global Distribution
		5000 RU/s and 50 GB per container
$0.008/hour for 100 RU/s	**$0.008/hour for 100 RU/s**	**$0.25 per 1M RU**

Execution Metrics

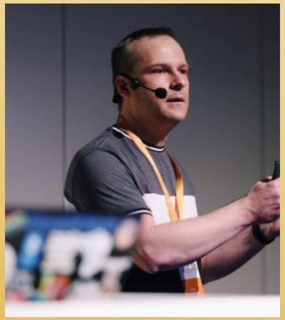
```
var query = client.CreateDocumentQuery(
            UriFactory.CreateDocumentCollectionUri(_dbname, _container),
            $"SELECT * FROM Orders o WHERE o.OrderId = {@orderid}",
            new FeedOptions {
              PopulateQueryMetrics = true,
              EnableCrossPartitionQuery =true }).AsDocumentQuery();
          FeedResponse<dynamic> result = await query.ExecuteNextAsync();
          IReadOnlyDictionary<string, QueryMetrics> metrics = result.QueryMetrics;
```



```
metrics                                          Count = 1
◢ [0]                                            {[0, Retrieved Document Count          :          0 ...
    Key                                          "0"
  ◢ Value                                        {Retrieved Document Count          :          0 ...
      IndexHitRatio                              1
      OutputDocumentCount                        0
    ◢ QueryEngineTimes                           {Microsoft.Azure.Documents.QueryEngineTimes}
      ▷ DocumentLoadTime                         {00:00:00}
      ▷ IndexLookupTime                          {00:00:00.0000700}
      ◢ RuntimeExecutionTimes                    {Microsoft.Azure.Documents.RuntimeExecutionTimes}
        ▷ SystemFunctionExecutionTime            {00:00:00}
        ▷ TotalTime                              {00:00:00.0000200}
        ▷ UserDefinedFunctionExecutionTime       {00:00:00}
        ▷ Static members
        ▷ Non-Public members
      ▷ WriteOutputTime                          {00:00:00}
      ▷ Non-Public members
    ◢ QueryPreparationTimes                      {Microsoft.Azure.Documents.QueryPreparationTimes}
      ▷ CompileTime                              {00:00:00.0000700}
      ▷ LogicalPlanBuildTime                     {00:00:00.0000300}
      ▷ PhysicalPlanBuildTime                    {00:00:00.0000600}
      ▷ QueryOptimizationTime                    {00:00:00}
      ▷ Static members
      ▷ Non-Public members
      Retries                                    0
      RetrievedDocumentCount                     0
      RetrievedDocumentSize                      0
    ▷ TotalTime                                  {00:00:00.0004900}
```

Thank you for joining us

Future Data Driven 2022



Scan the QR Code

Join The Data Driven Community

FUTURE
DATA
DRIVEN
SUMMIT

Thank you!

You can reach me on the following platforms.

SavranWeb

hasansavran

https://h-savran.blogspot.com/