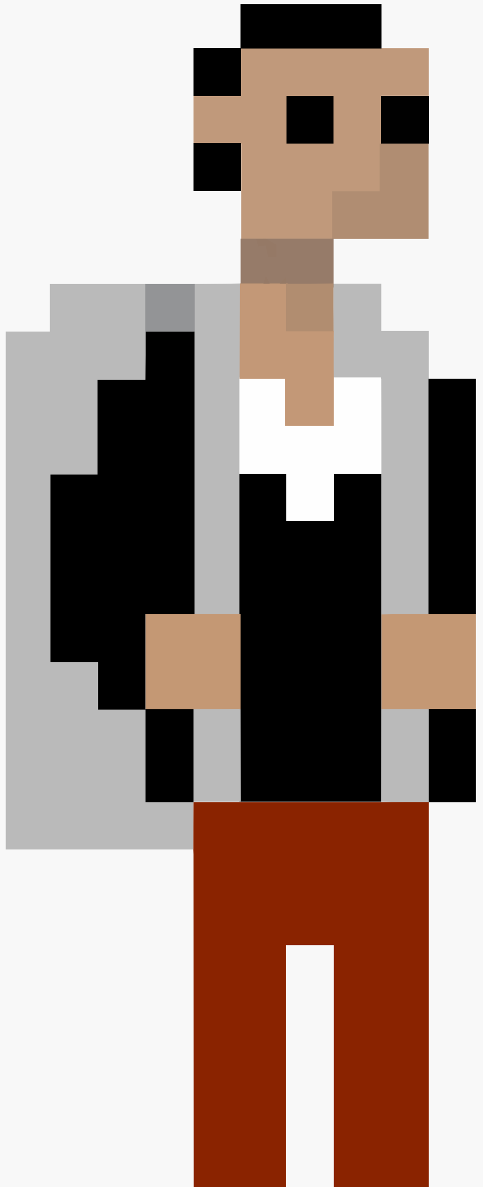
Azure Cloud Application Development
Persistent storage service and databases

Valdemar Zavadsky
Cloud Solution Architect
Microsoft

Who is this guy?



24
years of Dev
(C, C++, C#, SQL, Java)

10
years of R&D

10
years in PCI security

Valdemar
Zavadsky
Cloud Solution Architect at Microsoft

valdemar.zavadsky@microsoft.com
github: valda-z

Platform Services

Security & Management
- Security Center
- Portal
- Azure Active Directory
- Azure AD B2C
- Multi-Factor Authentication
- Automation
- Scheduler
- Key Vault
- Store/ Marketplace
- VM Image Gallery & VM Depot

Media & CDN
- Media Services
- Media Analytics
- Content Delivery Network

Integration
- API Management
- BizTalk Services
- Logic Apps
- Service Bus

Compute Services
- Container Service
- VM Scale Sets
- Batch
- Citrix Xen app
- Dev/Test Lab
- Azure Container Registry

Application Platform
- Web Apps
- Mobile Apps
- API Apps
- Cloud Services
- Service Fabric
- Notification Hubs
- Functions

Developer Services
- Visual Studio
- Mobile Engagement
- VS Team Services
- Xamarin
- Application Insights
- HockeyApp

Data
- SQL Database
- SQL Data Warehouse
- DocumentDB
- SQL Server Stretch Database
- Redis Cache
- Storage Tables
- Azure Search

Intelligence
- Cognitive Services
- Bot Framework
- Cortana

Analytics & IoT
- HDInsight
- Machine Learning
- Stream Analytics
- Data Catalog
- Data Lake Analytics Service
- Data Lake Store
- IoT Hub
- Event Hubs
- Data Factory
- Power BI Embedded

Hybrid Cloud
- Azure AD Health Monitoring
- AD Privileged Identity Management
- Domain Services
- Backup
- Operational Analytics
- Import/Export
- Azure Site Recovery
- StorSimple

Infrastructure Services

Compute
- Virtual Machines
- Containers

Storage
- Blob
- Queues
- Files
- Disks

Networking
- Virtual Network
- Load Balancer
- DNS
- Express Route
- Traffic Manager
- VPN Gateway
- App Gateway

Datacenter Infrastructure (34 Regions, 24 Online)

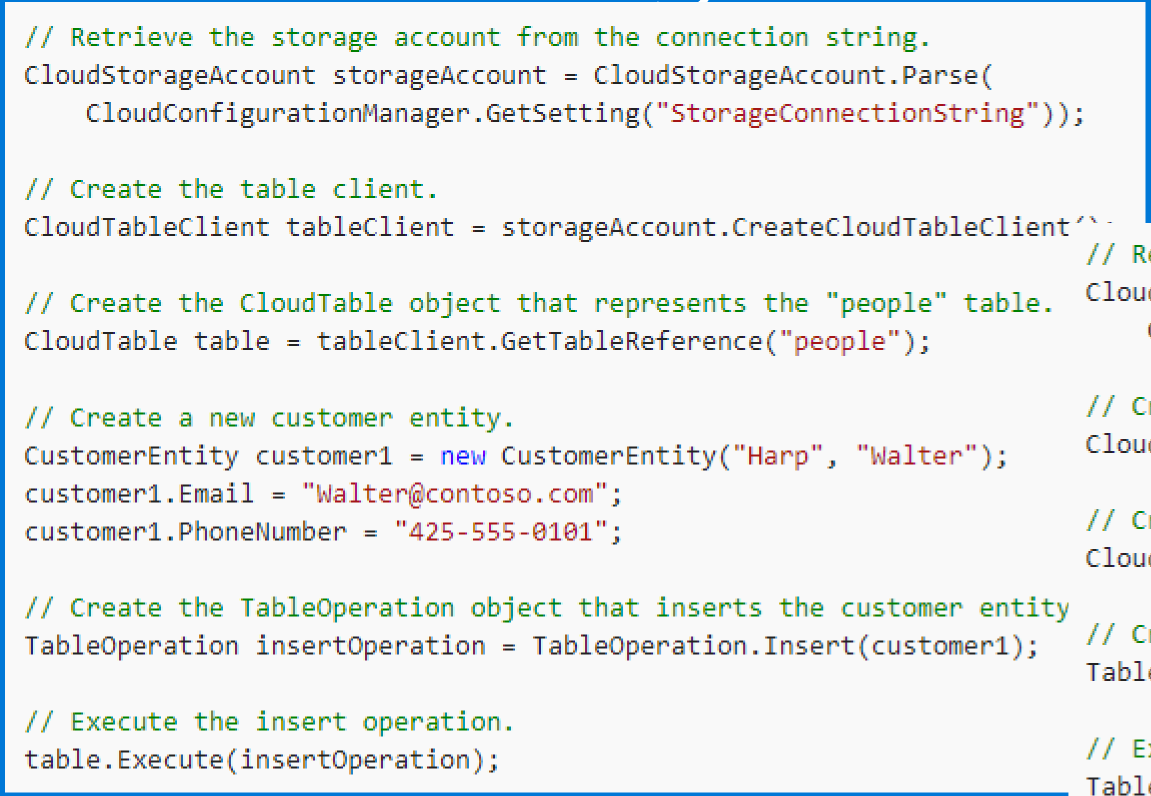
https://docs.microsoft.com/en-us/azure/#pivot=services&panel=all

Blob Storage - Blobs

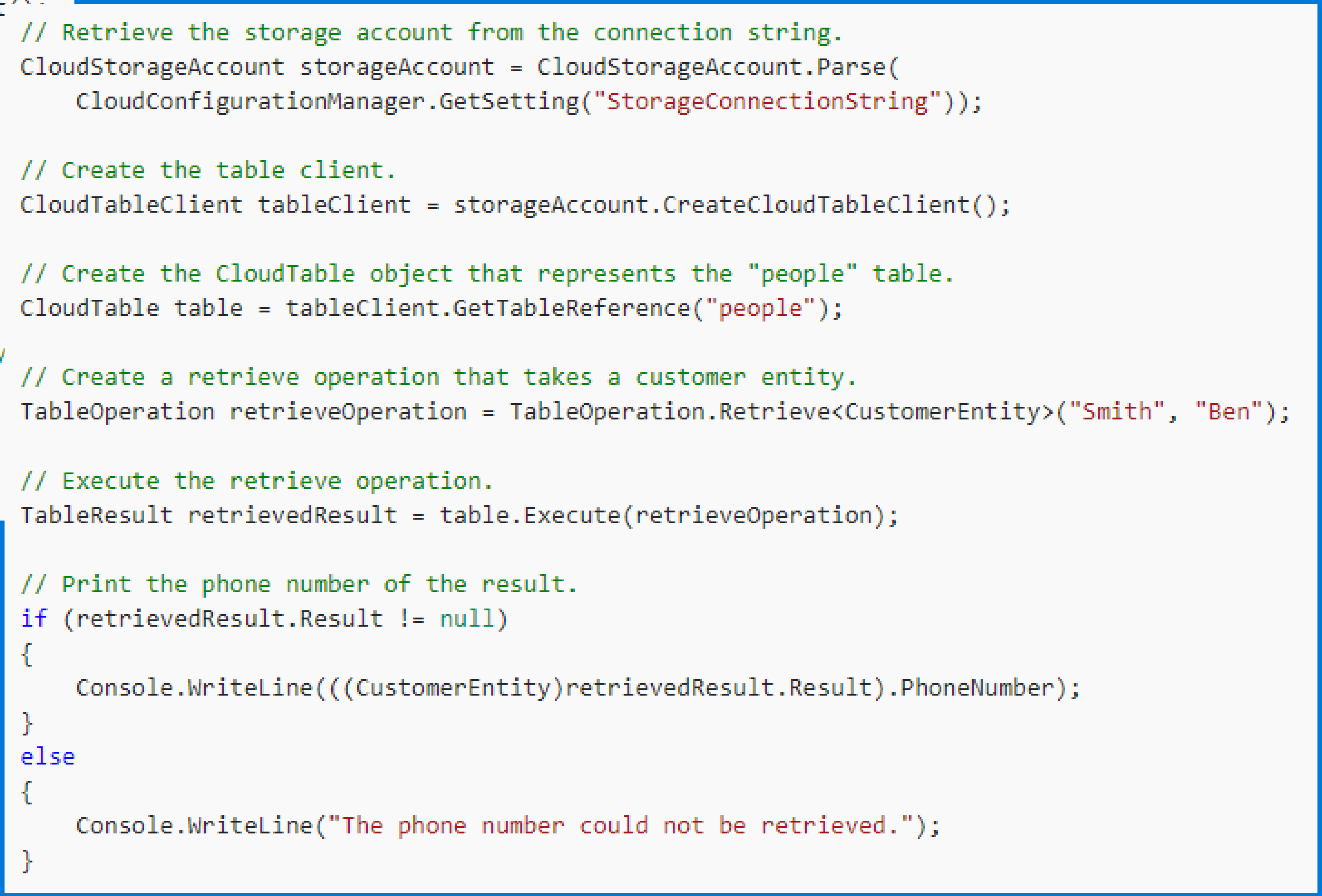
```
// Retrieve storage account from connection string.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the blob client.
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

// Retrieve reference to a previously created container.
CloudBlobContainer container = blobClient.GetContainerReference("mycontainer");

// Retrieve reference to a blob named "myblob".
CloudBlockBlob blockBlob = container.GetBlockBlobReference("myblob");

// Create or overwrite the "myblob" blob with contents from a local file.
using (var fileStream = System.IO.File.OpenRead(@"path\myfile"))
{
    blockBlob.UploadFromStream(fileStream);
}
```

```
// Retrieve storage account from connection string.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the blob client.
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

// Retrieve reference to a previously created container.
CloudBlobContainer container = blobClient.GetContainerReference("mycontainer");

// Retrieve reference to a blob named "photo1.jpg".
CloudBlockBlob blockBlob = container.GetBlockBlobReference("photo1.jpg");

// Save blob contents to a file.
using (var fileStream = System.IO.File.OpenWrite(@"path\myfile"))
{
    blockBlob.DownloadToStream(fileStream);
}
```

Blob Storage - Tables



```
// Retrieve the storage account from the connection string.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the table client.
CloudTableClient tableClient = storageAccount.CreateCloudTableClient(

// Create the CloudTable object that represents the "people" table.
CloudTable table = tableClient.GetTableReference("people");

// Create a new customer entity.
CustomerEntity customer1 = new CustomerEntity("Harp", "Walter");
customer1.Email = "Walter@contoso.com";
customer1.PhoneNumber = "425-555-0101";

// Create the TableOperation object that inserts the customer entity
TableOperation insertOperation = TableOperation.Insert(customer1);

// Execute the insert operation.
table.Execute(insertOperation);
```



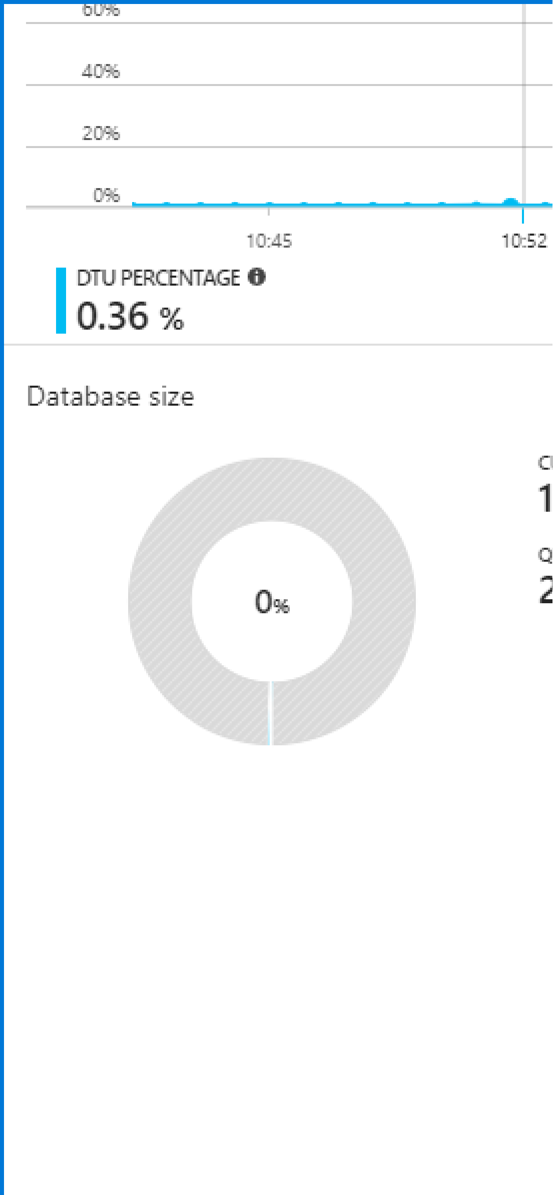
```
// Retrieve the storage account from the connection string.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the table client.
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();

// Create the CloudTable object that represents the "people" table.
CloudTable table = tableClient.GetTableReference("people");

// Create a retrieve operation that takes a customer entity.
TableOperation retrieveOperation = TableOperation.Retrieve<CustomerEntity>("Smith", "Ben");

// Execute the retrieve operation.
TableResult retrievedResult = table.Execute(retrieveOperation);

// Print the phone number of the result.
if (retrievedResult.Result != null)
{
    Console.WriteLine(((CustomerEntity)retrievedResult.Result).PhoneNumber);
}
else
{
    Console.WriteLine("The phone number could not be retrieved.");
}
```

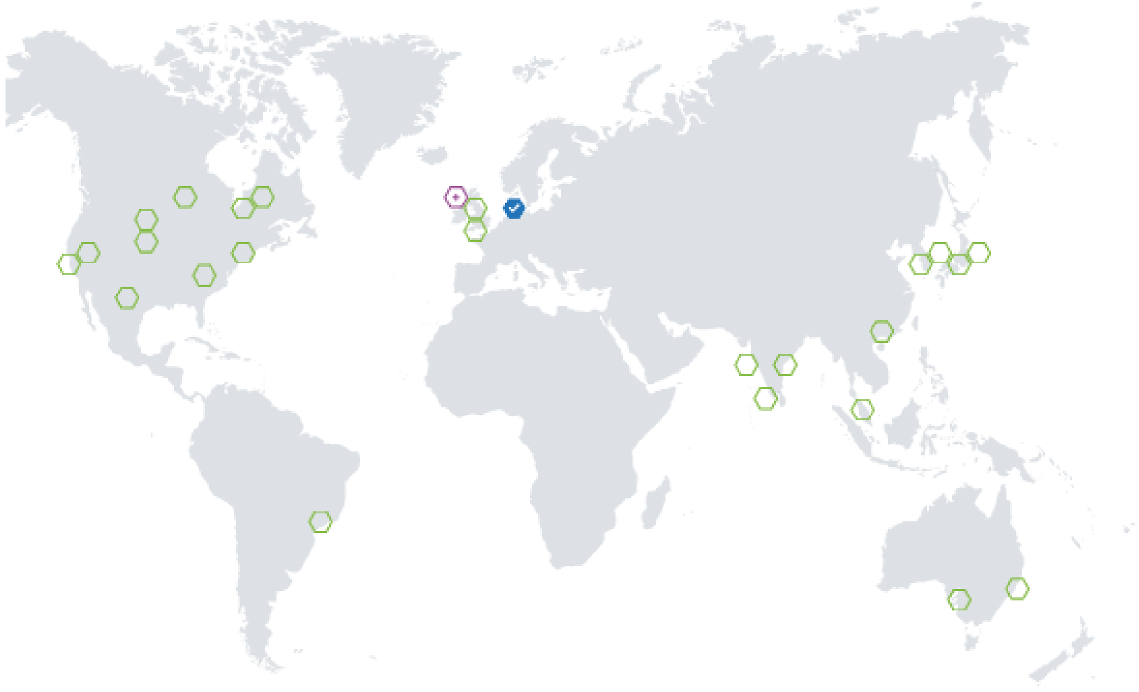
Blob Storage - Queue

```
// Retrieve storage account from connection string.
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the queue client.
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

// Retrieve a reference to a queue.
CloudQueue queue = queueClient.GetQueueReference("myqueue");

// Create the queue if it doesn't already exist.
queue.CreateIfNotExists();

// Create a message and add it to the queue.
CloudQueueMessage message = new CloudQueueMessage("Hello, World");
queue.AddMessage(message);
```

```
// Retrieve storage account from connection string
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
    CloudConfigurationManager.GetSetting("StorageConnectionString"));

// Create the queue client
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

// Retrieve a reference to a queue
CloudQueue queue = queueClient.GetQueueReference("myqueue");

// Peek at the next message
CloudQueueMessage peekedMessage = queue.PeekMessage();

// Display message.
Console.WriteLine(peekedMessage.AsString);
```

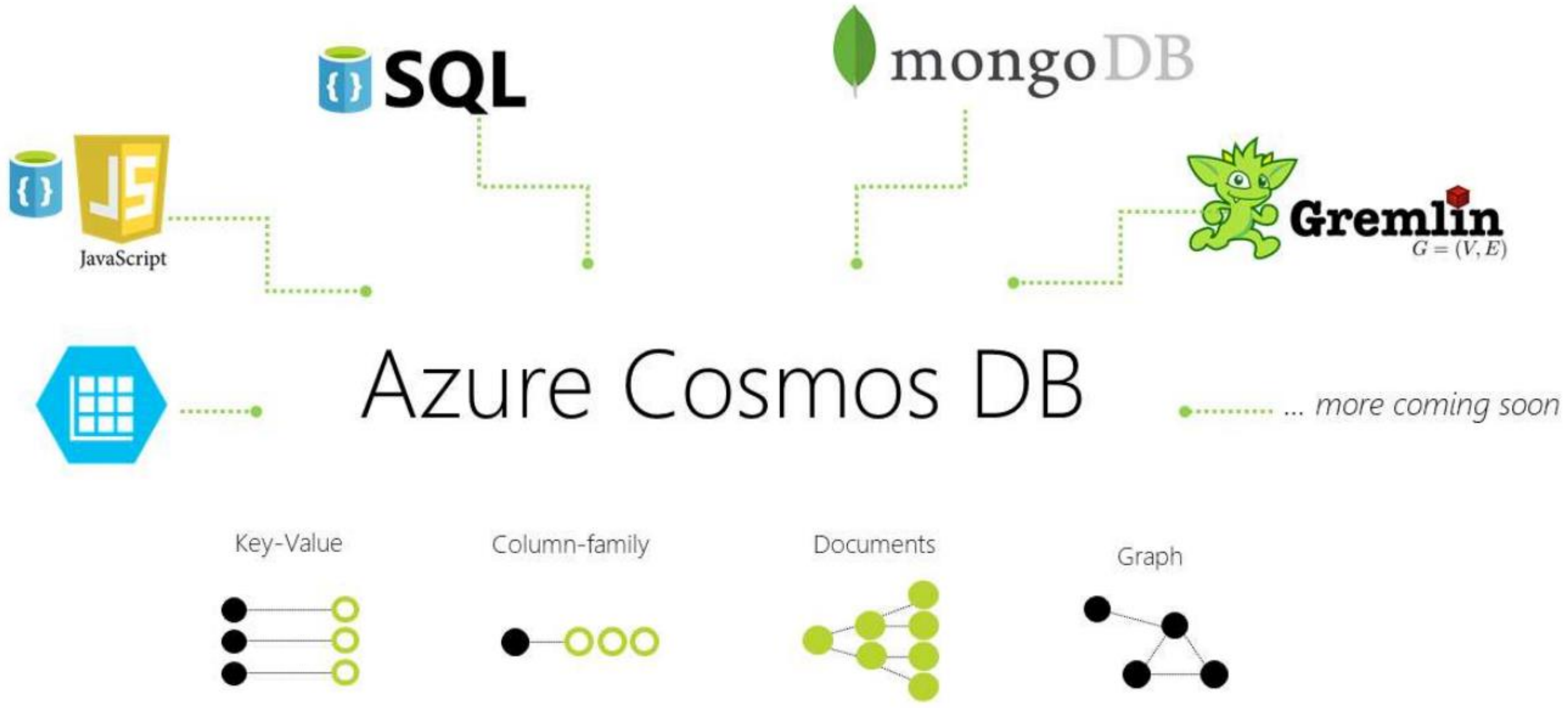
Azure SC





	SERVER/DATABASE	FAILOVER POLICY
PRIMARY		
West Europe	valda/mysmarthome	None
SECONDARIES		
Geo-Replication is not configured		
TARGET REGIONS		
North Europe	*Recommended*	
West US		
West US 2		

Azure Cosmos DB

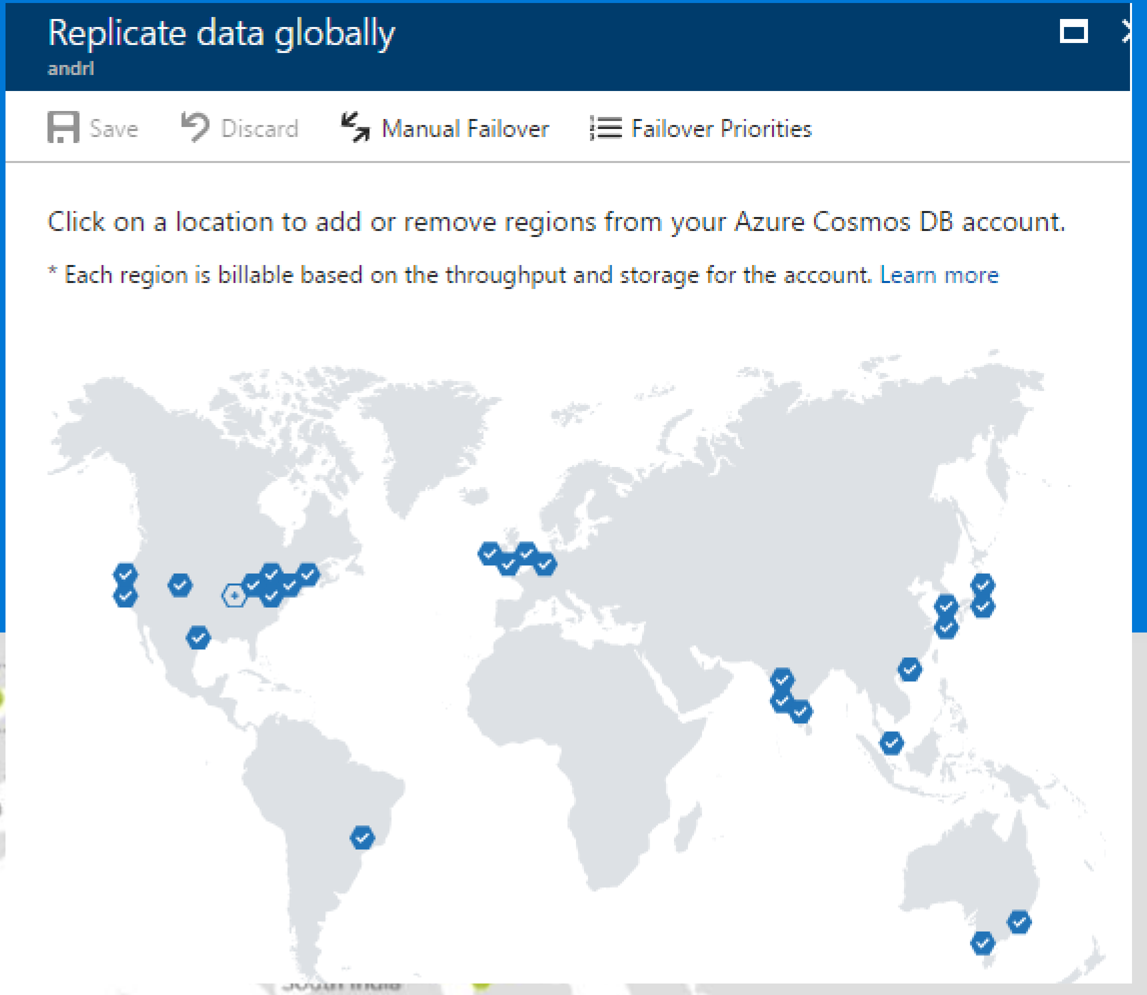


Global distribution | Elastic scale out | Guaranteed low latency | Five consistency models | Comprehensive SLAs

Azure Cosmos DB

Azure DocumentDB is a fully managed NoSQL "database as a service" built for ultra- fast and predictable performance, high availability, elastic scaling, and global distribution, and is especially focused on ease of development.

Canada East
Canada Central
US Gov Iowa
Central US
West US
US Gov Arizona
US Gov Texas
South Central US
North Central US
US DoD East
East US
East US 2
US Gov Virginia
US DoD Central
UK South
North Europe
UK West
France Central
France South
Brazil South
Southeast Asia
Australia East
Australia Southeast
Generally available

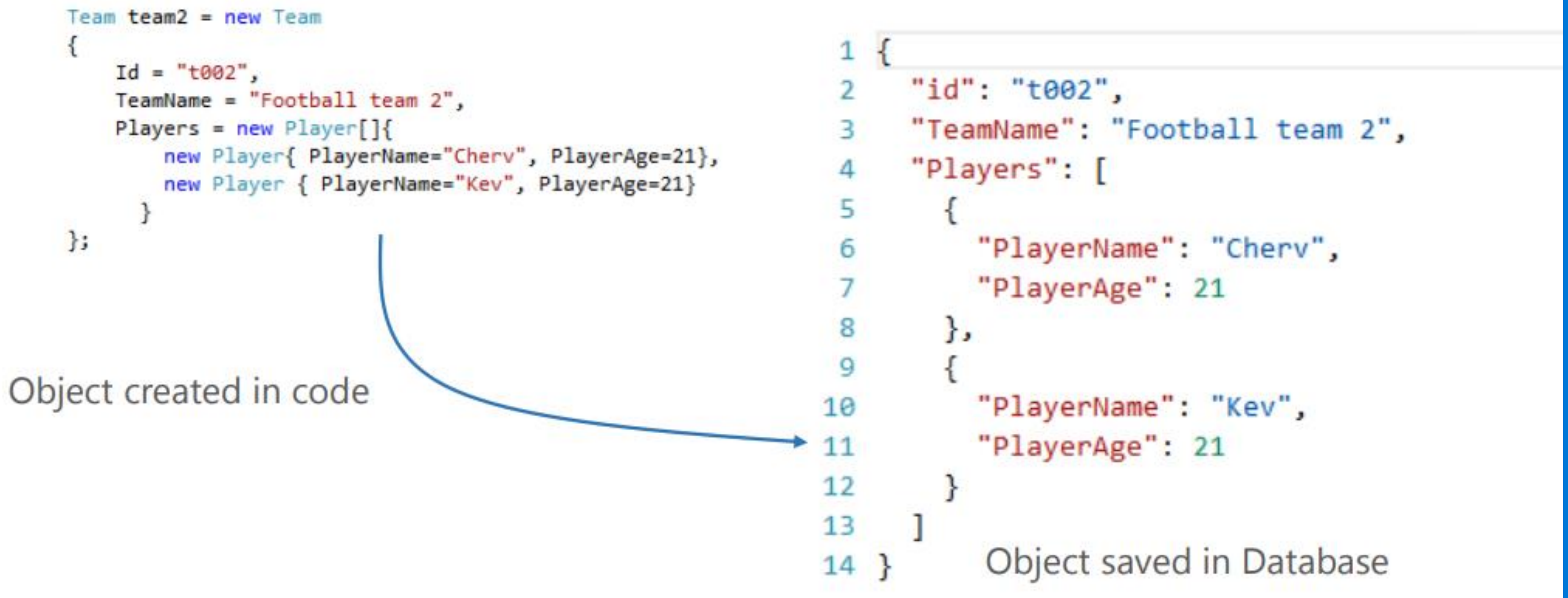


Click on a location to add or remove regions from your Azure Cosmos DB account.

* Each region is billable based on the throughput and storage for the account. Learn more

Azure Cosmos DB – document DB data model

All data is stored in JSON Documents

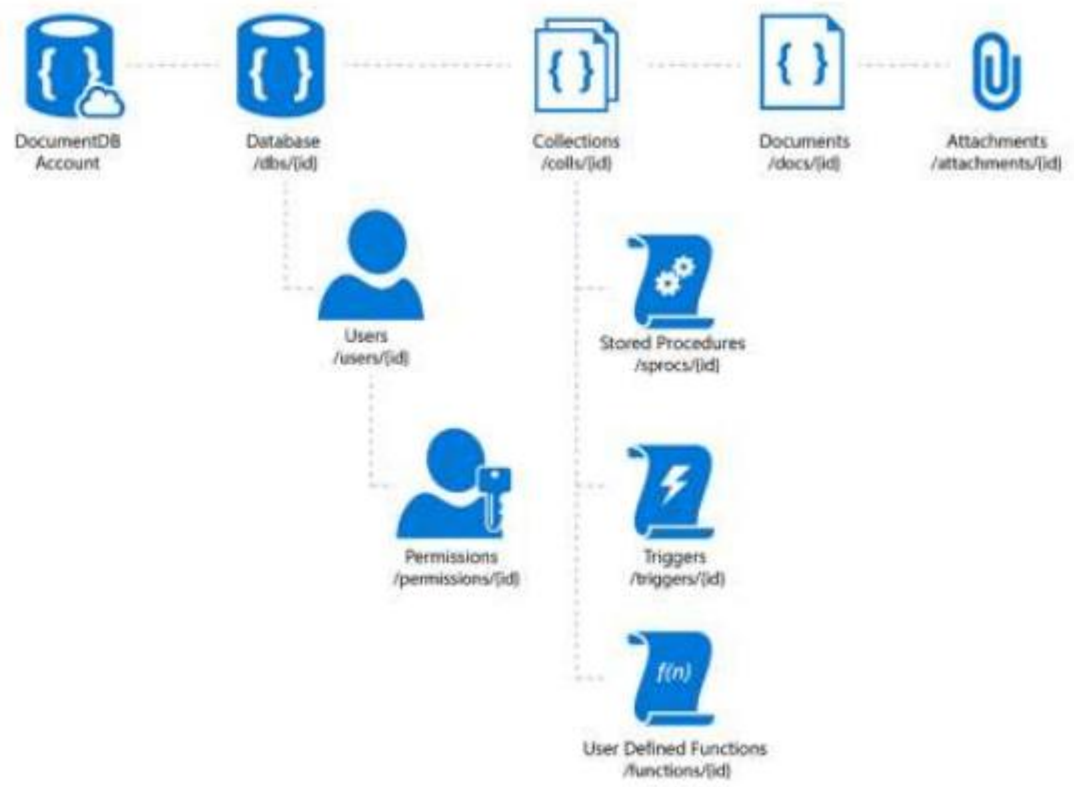


```csharp
Team team2 = new Team
{
    Id = "t002",
    TeamName = "Football team 2",
    Players = new Player[]{
        new Player{ PlayerName="Cherv", PlayerAge=21},
        new Player { PlayerName="Kev", PlayerAge=21}
      }
};
```

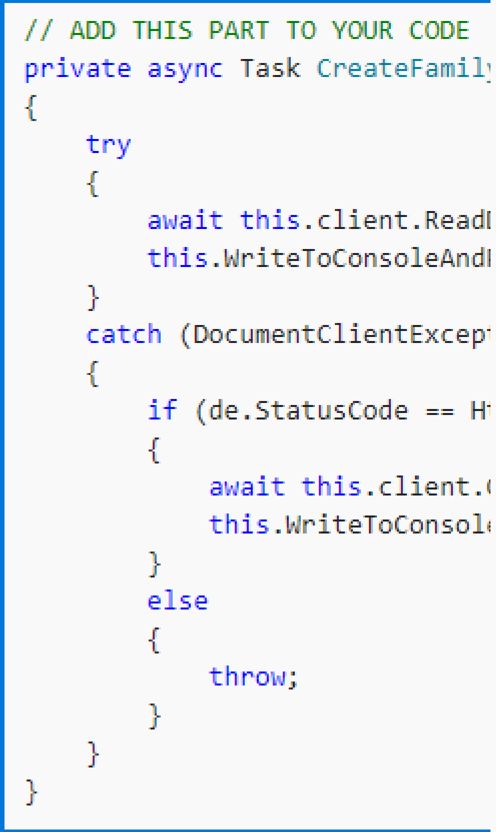
```json
{
  "id": "t002",
  "TeamName": "Football team 2",
  "Players": [
    {
      "PlayerName": "Cherv",
      "PlayerAge": 21
    },
    {
      "PlayerName": "Kev",
      "PlayerAge": 21
    }
  ]
}
```

Azure Cosmos DB – document DB resources

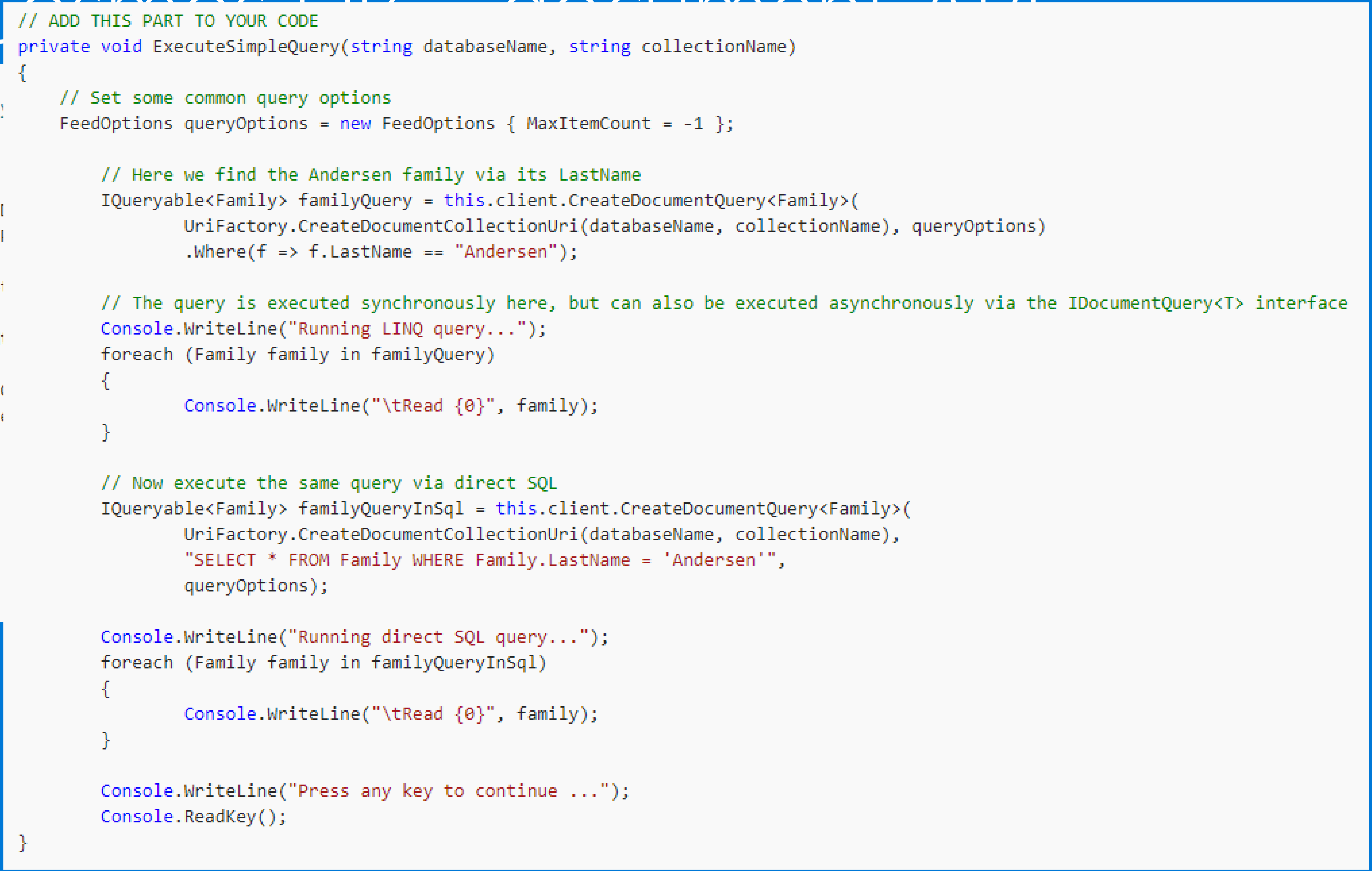
- Database
- User
- Collection
- Stored Procedure
- Trigger
- User-defined Function (UDF)
- Document
- Attachment


DocumentDB Account
Database /dbs/{id}
Collections /colls/{id}
Documents /docs/{id}
Attachments /attachments/{id}
Users /users/{id}
Permissions /permissions/{id}
Stored Procedures /sprocs/{id}
Triggers /triggers/{id}
f(n)
User Defined Functions /functions/{id}

Azure C

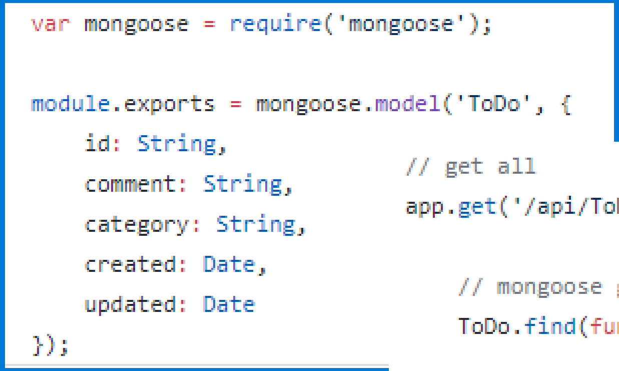


```
// ADD THIS PART TO YOUR CODE
private async Task CreateFamil
{
    try
    {
        await this.client.Read
        this.WriteToConsoleAnd
    }
    catch (DocumentClientExcep
    {
        if (de.StatusCode == H
        {
            await this.client.
            this.WriteToConsol
        }
        else
        {
            throw;
        }
    }
}
```

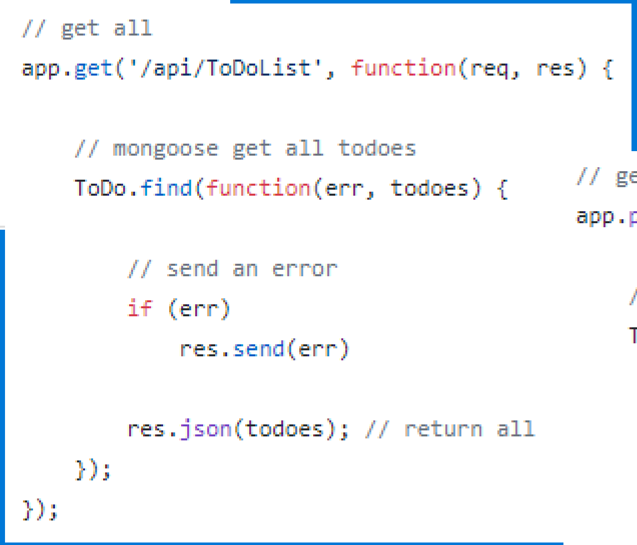


```
// ADD THIS PART TO YOUR CODE
private void ExecuteSimpleQuery(string databaseName, string collectionName)
{
    // Set some common query options
    FeedOptions queryOptions = new FeedOptions { MaxItemCount = -1 };

        // Here we find the Andersen family via its LastName
        IQueryable<Family> familyQuery = this.client.CreateDocumentQuery<Family>(
                UriFactory.CreateDocumentCollectionUri(databaseName, collectionName), queryOptions)
                .Where(f => f.LastName == "Andersen");

        // The query is executed synchronously here, but can also be executed asynchronously via the IDocumentQuery<T> interface
        Console.WriteLine("Running LINQ query...");
        foreach (Family family in familyQuery)
        {
                Console.WriteLine("\tRead {0}", family);
        }

        // Now execute the same query via direct SQL
        IQueryable<Family> familyQueryInSql = this.client.CreateDocumentQuery<Family>(
                UriFactory.CreateDocumentCollectionUri(databaseName, collectionName),
                "SELECT * FROM Family WHERE Family.LastName = 'Andersen'",
                queryOptions);

        Console.WriteLine("Running direct SQL query...");
        foreach (Family family in familyQueryInSql)
        {
                Console.WriteLine("\tRead {0}", family);
        }

        Console.WriteLine("Press any key to continue ...");
        Console.ReadKey();
}
```

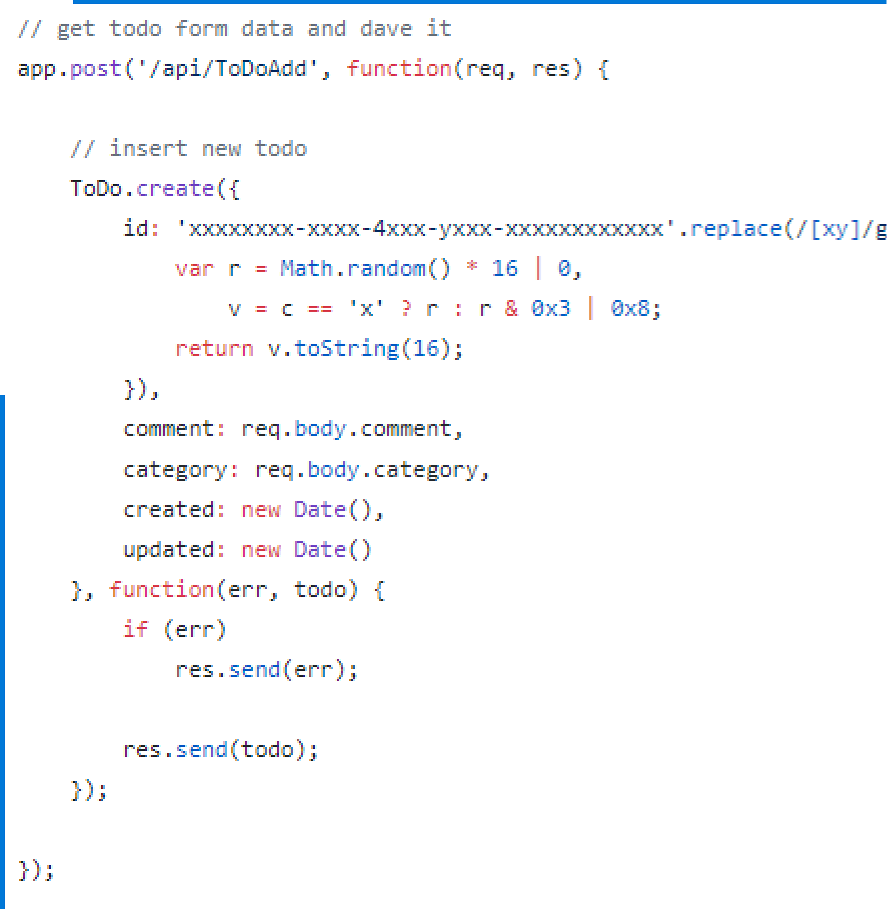
Azure Cosmos DB – MongoDB API



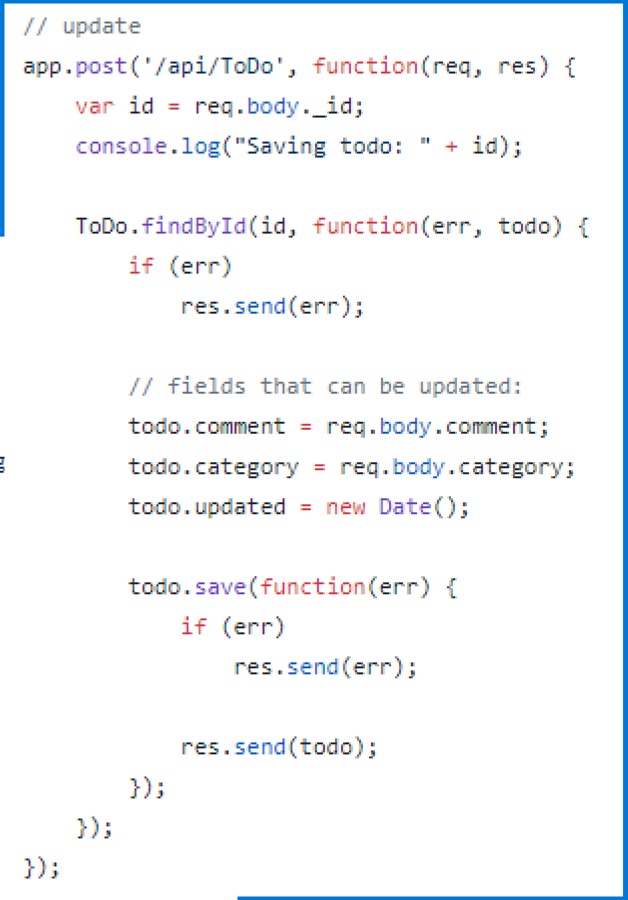
```
var mongoose = require('mongoose');

module.exports = mongoose.model('ToDo', {
    id: String,
    comment: String,
    category: String,
    created: Date,
    updated: Date
});
```



```
// get all
app.get('/api/ToDoList', function(req, res) {

    // mongoose get all todoes
    ToDo.find(function(err, todoes) {

        // send an error
        if (err)
            res.send(err)

        res.json(todoes); // return all
    });
});
```

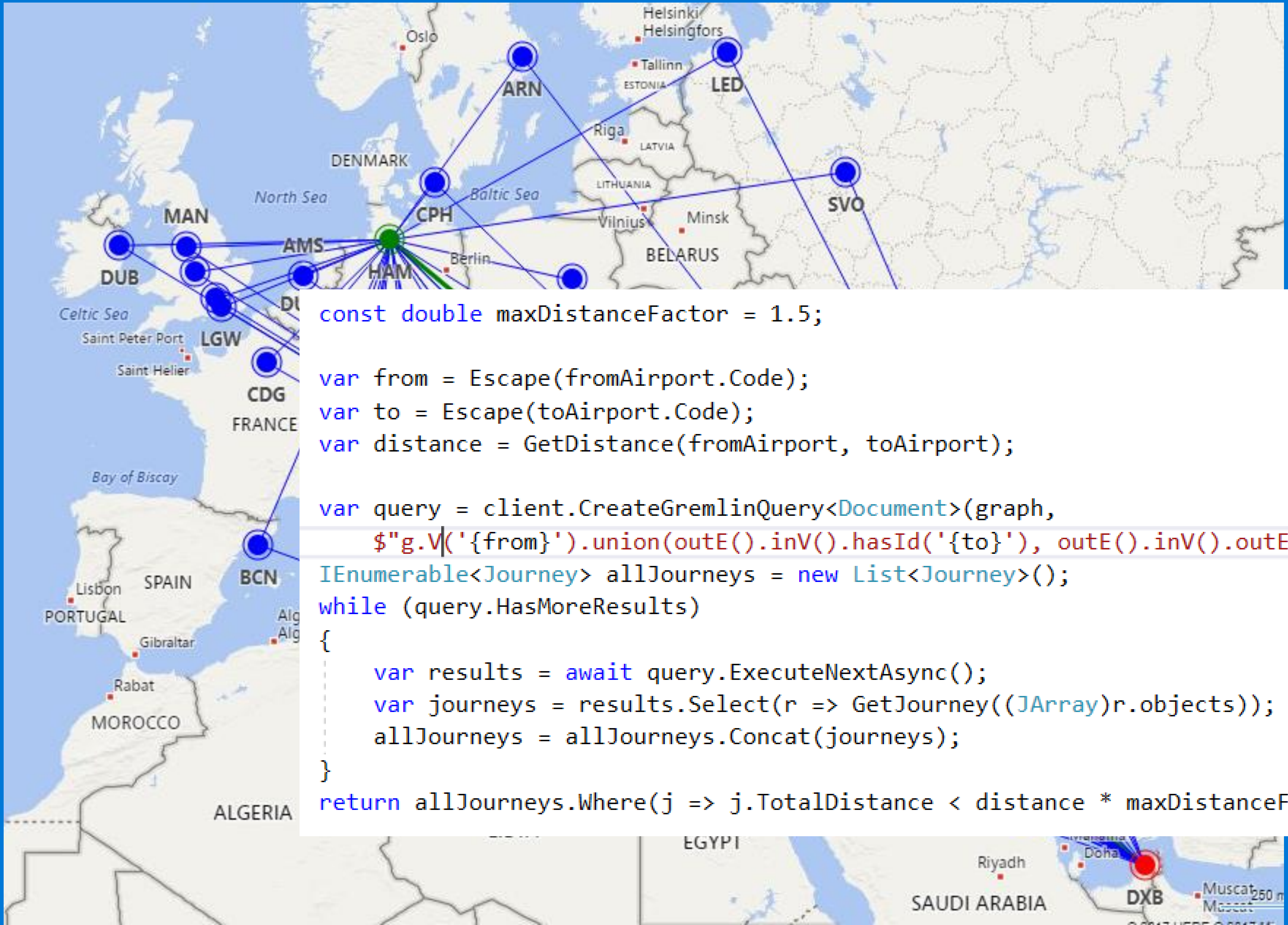


```
// get todo form data and dave it
app.post('/api/ToDoAdd', function(req, res) {

    // insert new todo
    ToDo.create({
        id: 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g
            var r = Math.random() * 16 | 0,
                v = c == 'x' ? r : r & 0x3 | 0x8;
            return v.toString(16);
        }),
        comment: req.body.comment,
        category: req.body.category,
        created: new Date(),
        updated: new Date()
    }, function(err, todo) {
        if (err)
            res.send(err);

        res.send(todo);
    });

});
```



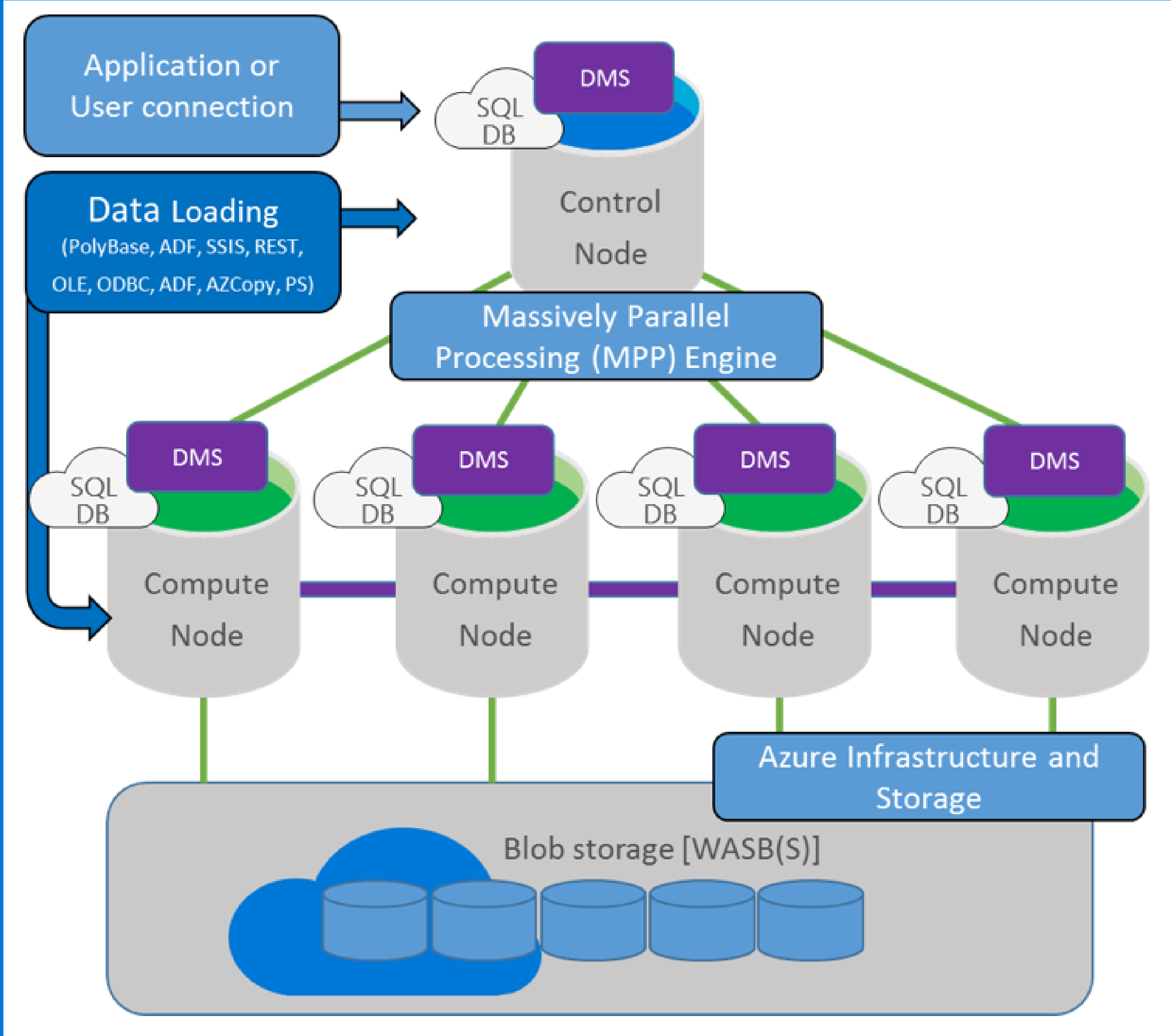
```
// update
app.post('/api/ToDo', function(req, res) {
    var id = req.body._id;
    console.log("Saving todo: " + id);

    ToDo.findById(id, function(err, todo) {
        if (err)
            res.send(err);

        // fields that can be updated:
        todo.comment = req.body.comment;
        todo.category = req.body.category;
        todo.updated = new Date();

        todo.save(function(err) {
            if (err)
                res.send(err);

            res.send(todo);
        });
    });
});
```

Azure Cosmos DB – Graph API (gremlin)

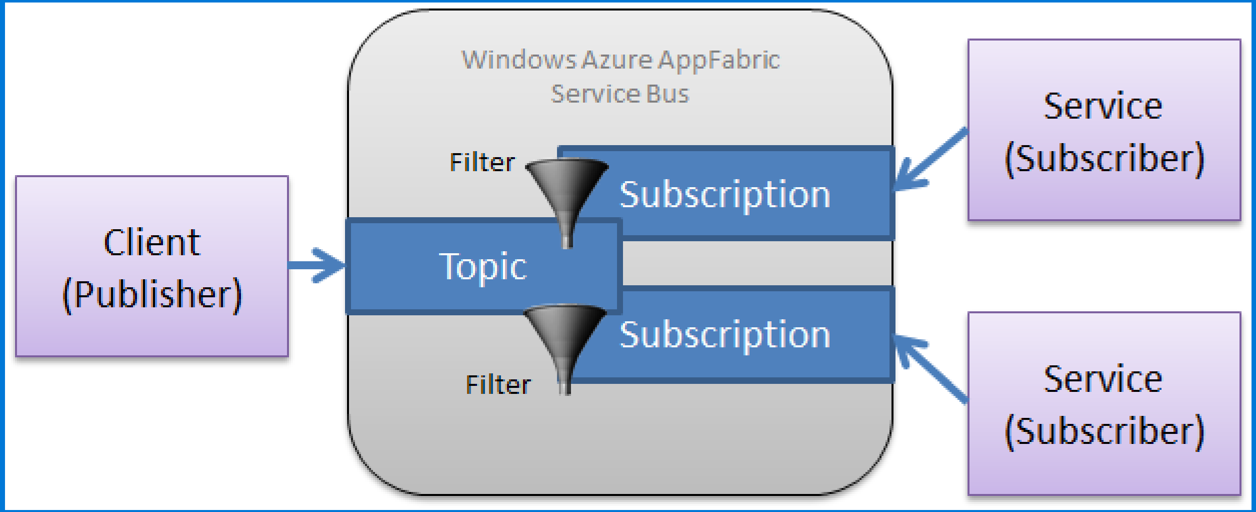


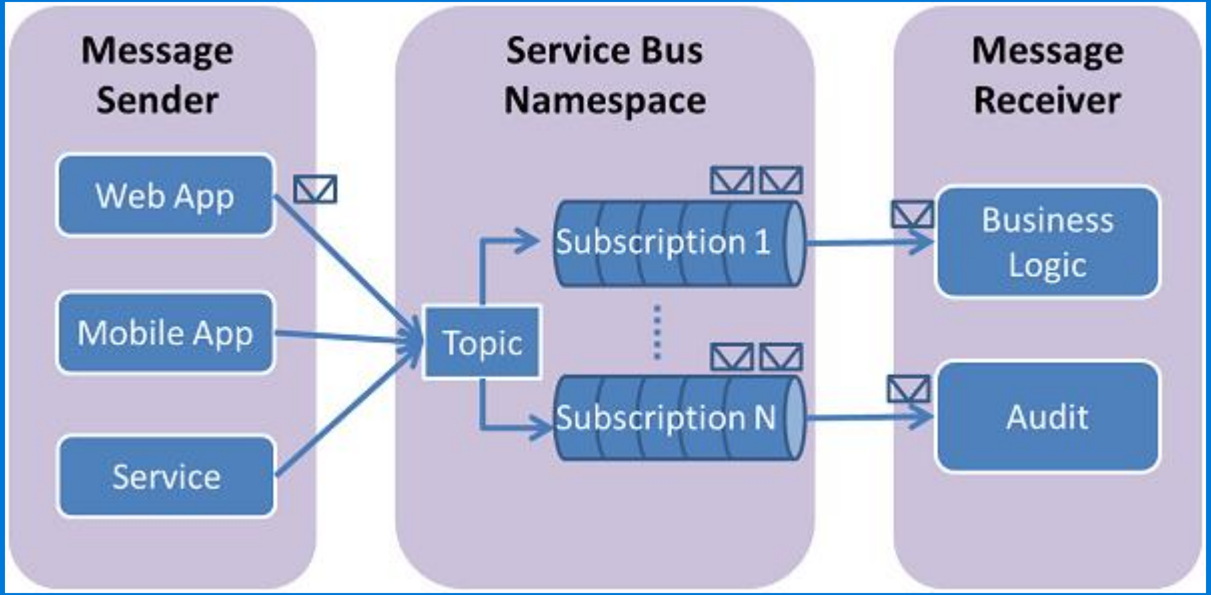
```
const double maxDistanceFactor = 1.5;

var from = Escape(fromAirport.Code);
var to = Escape(toAirport.Code);
var distance = GetDistance(fromAirport, toAirport);

var query = client.CreateGremlinQuery<Document>(graph,
    $"g.V('{from}').union(outE().inV().hasId('{to}'), outE().inV().outE().inV().hasId('{to}')).path()");
IEnumerable<Journey> allJourneys = new List<Journey>();
while (query.HasMoreResults)
{
    var results = await query.ExecuteNextAsync();
    var journeys = results.Select(r => GetJourney((JArray)r.objects));
    allJourneys = allJourneys.Concat(journeys);
}
return allJourneys.Where(j => j.TotalDistance < distance * maxDistanceFactor);
```

Azure SQL Data Warehouse

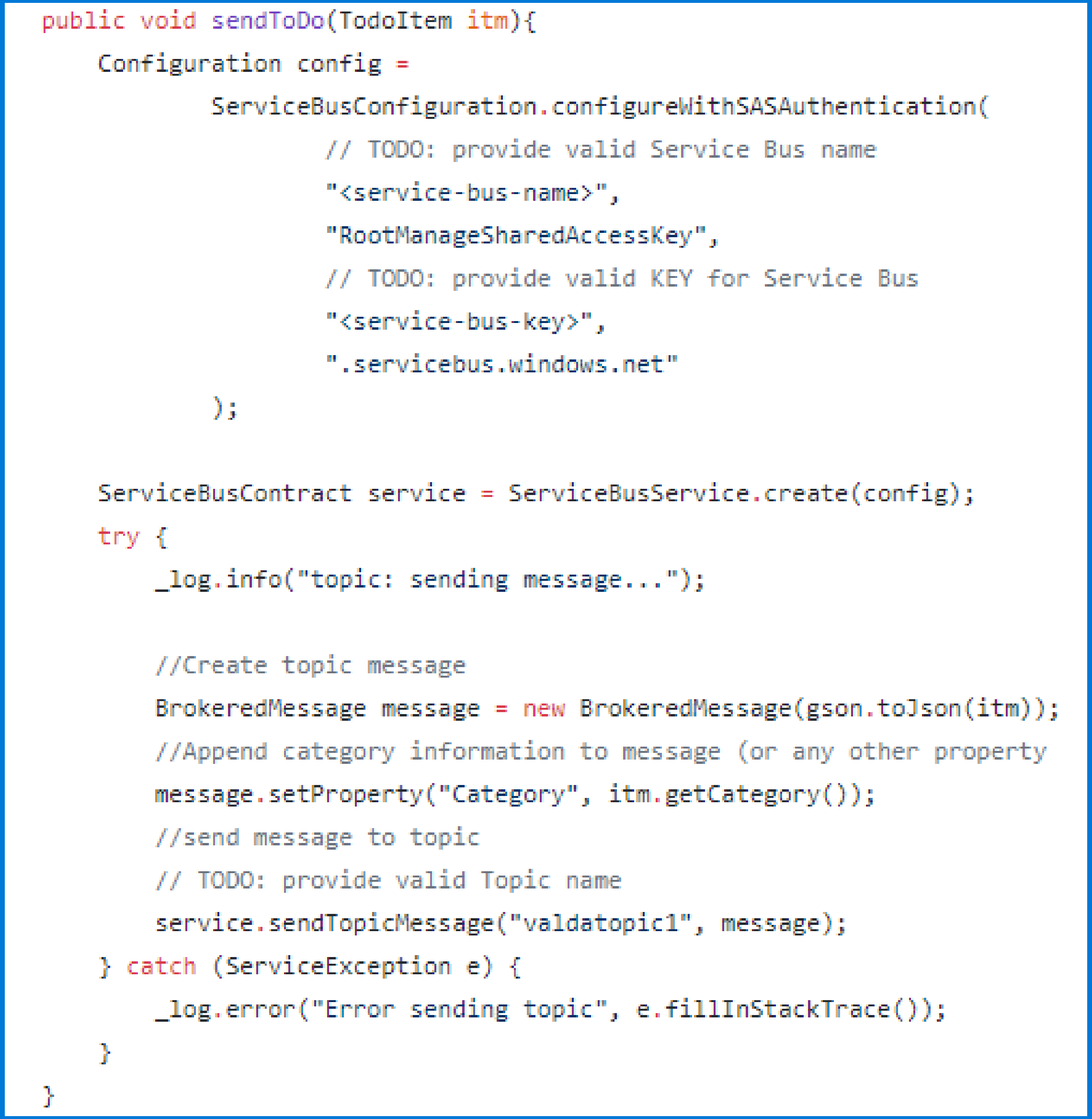

Application or User connection
Data Loading
(PolyBase, ADF, SSIS, REST, OLE, ODBC, ADF, AZCopy, PS)
DMS
SQL DB
Control Node
Massively Parallel Processing (MPP) Engine
DMS
SQL DB
Compute Node
DMS
SQL DB
Compute Node
DMS
SQL DB
Compute Node
DMS
SQL DB
Compute Node
Azure Infrastructure and Storage
Blob storage [WASB(S)]

Azure Service Bus

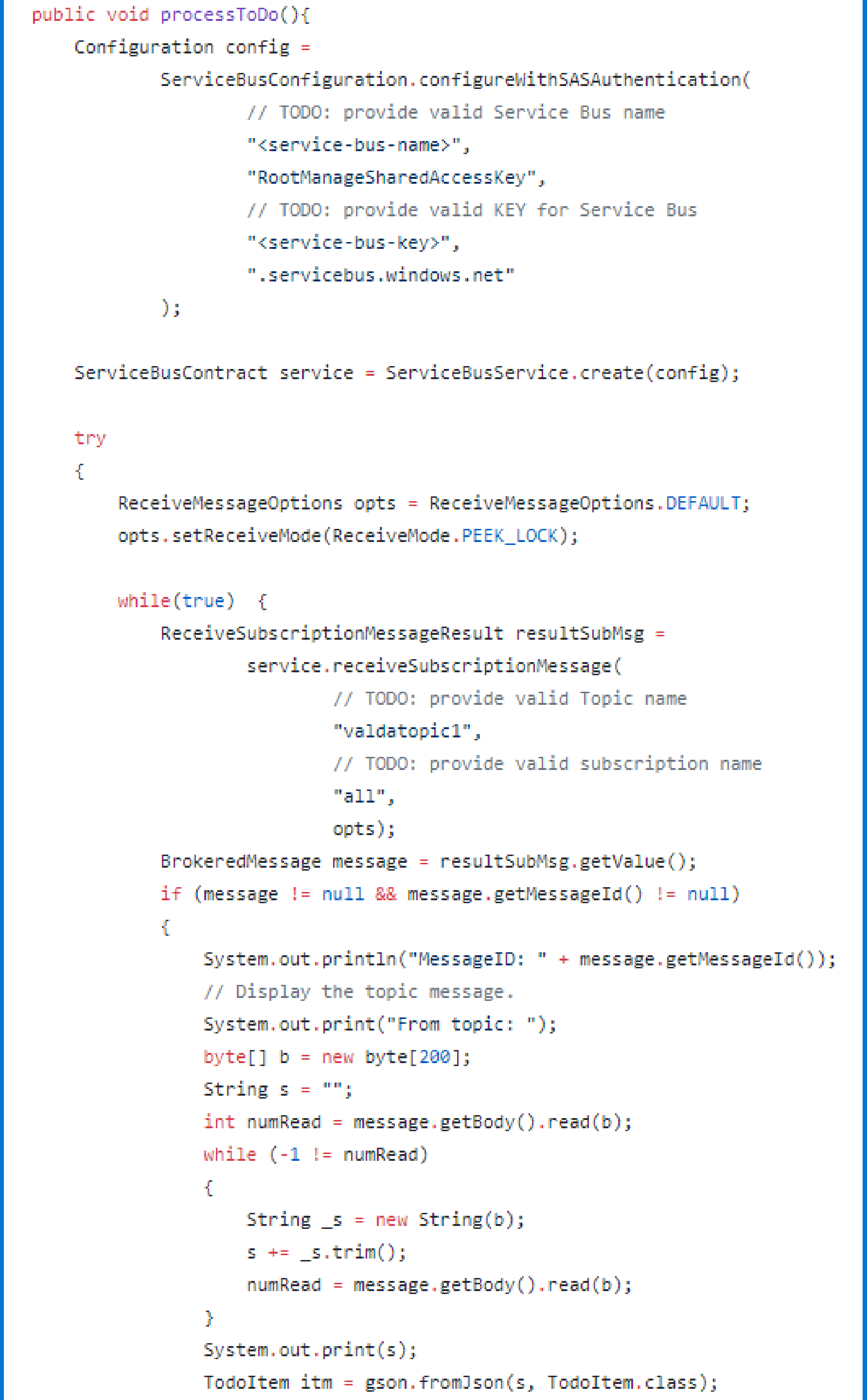




Azure Service Bus



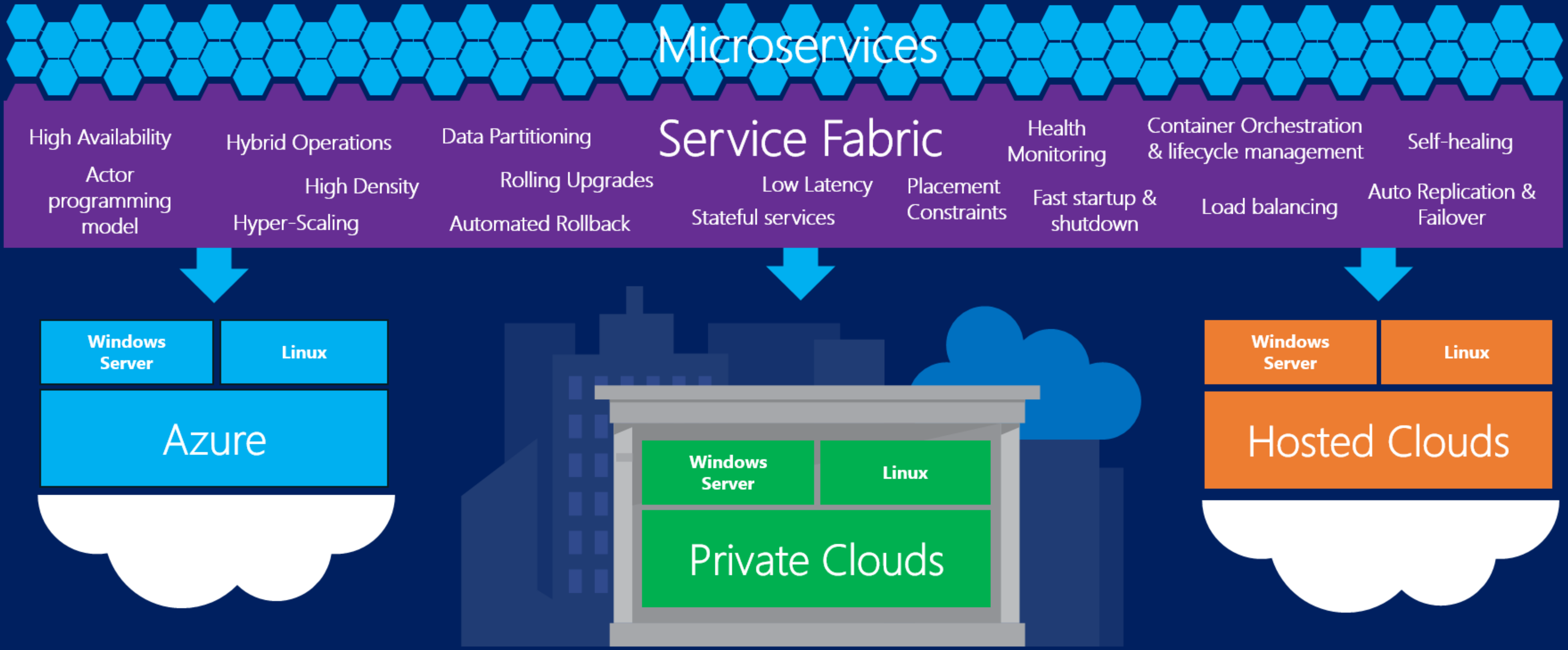
```java
public void sendToDo(TodoItem itm){
    Configuration config =
            ServiceBusConfiguration.configureWithSASAuthentication(
                    // TODO: provide valid Service Bus name
                    "<service-bus-name>",
                    "RootManageSharedAccessKey",
                    // TODO: provide valid KEY for Service Bus
                    "<service-bus-key>",
                    ".servicebus.windows.net"
            );

    ServiceBusContract service = ServiceBusService.create(config);
    try {
        _log.info("topic: sending message...");

        //Create topic message
        BrokeredMessage message = new BrokeredMessage(gson.toJson(itm));
        //Append category information to message (or any other property
        message.setProperty("Category", itm.getCategory());
        //send message to topic
        // TODO: provide valid Topic name
        service.sendTopicMessage("valdatopic1", message);
    } catch (ServiceException e) {
        _log.error("Error sending topic", e.fillInStackTrace());
    }
}
```



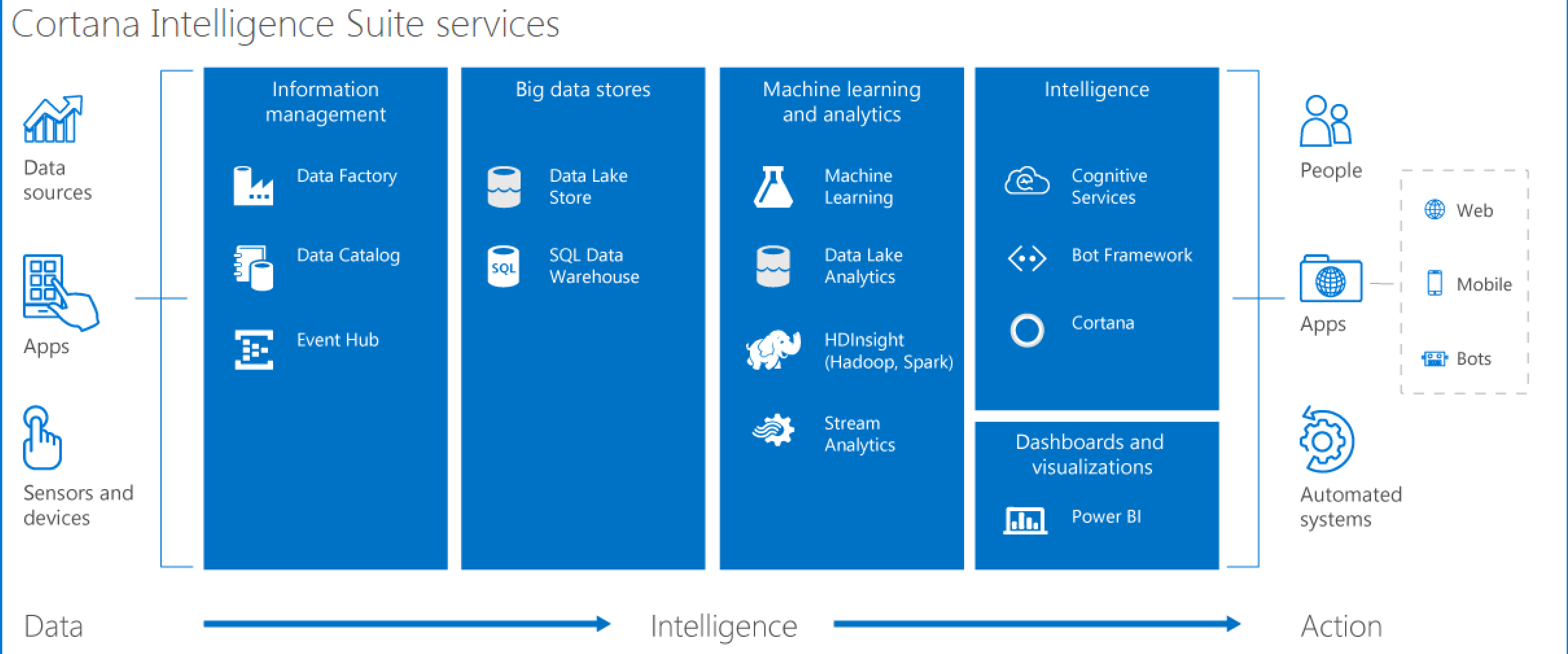
```java
public void processToDo(){
    Configuration config =
            ServiceBusConfiguration.configureWithSASAuthentication(
                    // TODO: provide valid Service Bus name
                    "<service-bus-name>",
                    "RootManageSharedAccessKey",
                    // TODO: provide valid KEY for Service Bus
                    "<service-bus-key>",
                    ".servicebus.windows.net"
            );

    ServiceBusContract service = ServiceBusService.create(config);

    try
    {
        ReceiveMessageOptions opts = ReceiveMessageOptions.DEFAULT;
        opts.setReceiveMode(ReceiveMode.PEEK_LOCK);

        while(true)  {
            ReceiveSubscriptionMessageResult resultSubMsg =
                    service.receiveSubscriptionMessage(
                            // TODO: provide valid Topic name
                            "valdatopic1",
                            // TODO: provide valid subscription name
                            "all",
                            opts);
            BrokeredMessage message = resultSubMsg.getValue();
            if (message != null && message.getMessageId() != null)
            {
                System.out.println("MessageID: " + message.getMessageId());
                // Display the topic message.
                System.out.print("From topic: ");
                byte[] b = new byte[200];
                String s = "";
                int numRead = message.getBody().read(b);
                while (-1 != numRead)
                {
                    String _s = new String(b);
                    s += _s.trim();
                    numRead = message.getBody().read(b);
                }
                System.out.print(s);
                TodoItem itm = gson.fromJson(s, TodoItem.class);
```

Microsoft Azure Service Fabric

A platform for reliable, hyperscale, microservice-based applications



Azure – Data platform


Cortana Intelligence Suite services
Data sources
Apps
Sensors and devices
Information management
Data Factory
Data Catalog
Event Hub
Big data stores
Data Lake Store
SQL Data Warehouse
Machine learning and analytics
Machine Learning
Data Lake Analytics
HDInsight (Hadoop, Spark)
Stream Analytics
Intelligence
Cognitive Services
Bot Framework
Cortana
Dashboards and visualizations
Power BI
People
Apps
Web
Mobile
Bots
Automated systems
Data
Intelligence
Action