Working with Azure Cosmos DB - MongoDB API

Reza Salehi
CLOUD CONSULTANT

@zaalion linkedin.com/in/rezasalehi2008

Overview

What is MongoDB?

Why migrating to Azure Cosmos DB MongoDB API?

- Benefits of a managed service

What you need to do before & after migration

Demo: Migrating to Cosmos DB MongoDB API

MongoDB is a cross-platform document-oriented database program.

MongoDB

Is a NoSQL database

Documents are stored in JSON format

Wire Protocol Compatibility

Azure Cosmos DB implements wire protocols of common NoSQL databases including Cassandra, MongoDB, Gremlin, and Azure Table Storage

This allows existing client SDKs, drivers, and tools of these NoSQL databases to interact with Cosmos DB

You can migrate your application to Cosmos DB while preserving most of its logic

Containers in Each API

Azure Cosmos API	Specialized Entity
SQL API	Collection
Table API	Table
MongoDB API	**Collection**
Cassandra API	Table
Gremlin API	Graph

Why Would You Migrate to Cosmos DB?

Get financially backed SLAs for the NoSQL APIs powered by Cosmos DB

Global distribution with multi-master replication

Elastically scale the provisioned throughput and storage for your Cosmos databases

Pay only for the throughput and storage you need

Pre-migration Steps

Pre-migration Steps

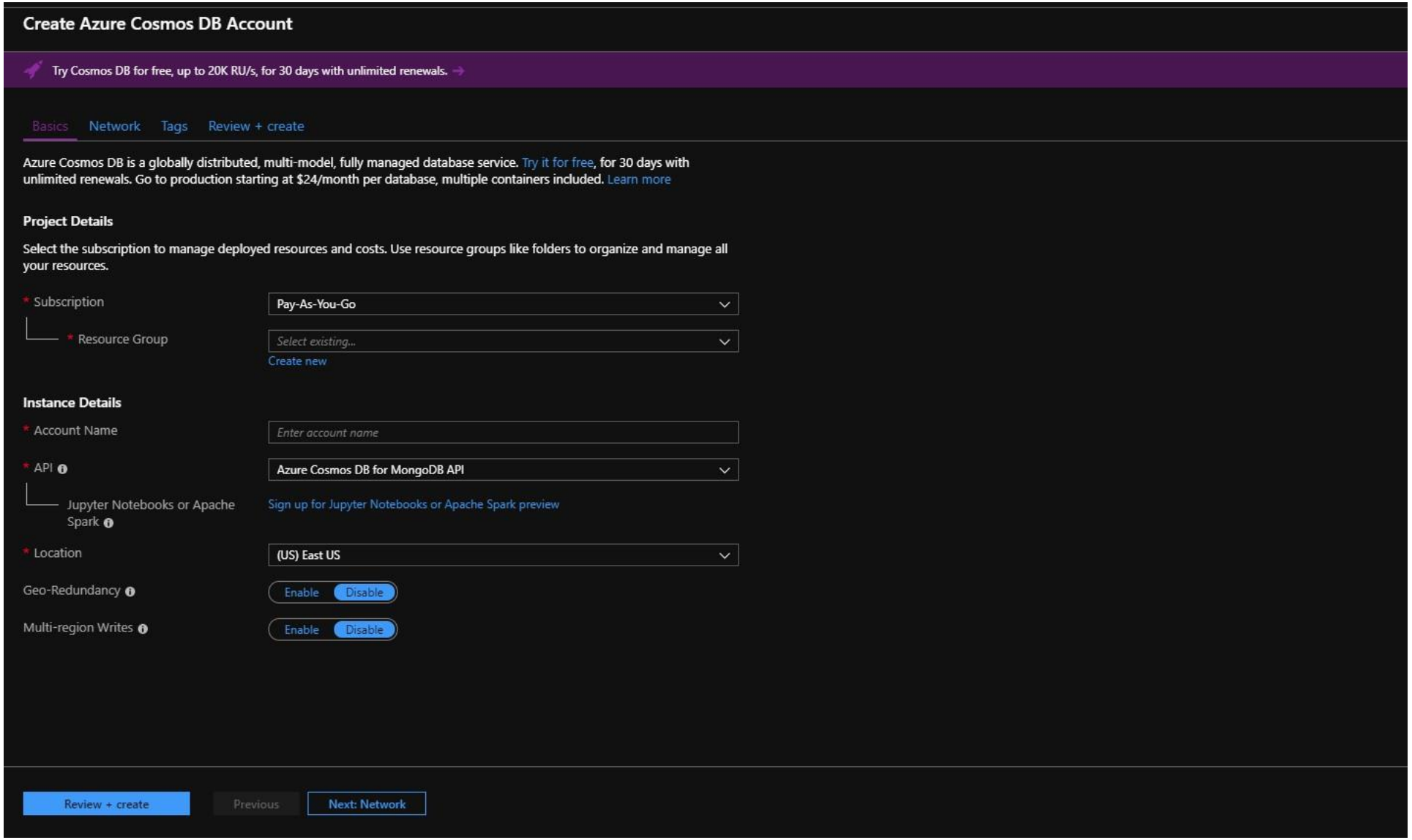
Create an Azure Cosmos DB account

Estimate the throughput needed for your workload

Pick an optimal partition key for your data

Understand the indexing policy that you can set on your data

Create an Azure Cosmos DB Account



“Before starting the migration, you should estimate the amount of throughput to provision for your Azure Cosmos databases and collections.”

Microsoft

Estimate the Throughput for Your Workload

Throughput can be provisioned on both collection & database

You can choose database-level throughput if not sure

Throughput is measured in Request Units (RUs) / second

RU is an abstraction of physical resources, (Memory, CPU, IOPs)

~DTUs in Azure SQL Database

Key Factors Affecting Needed RUs

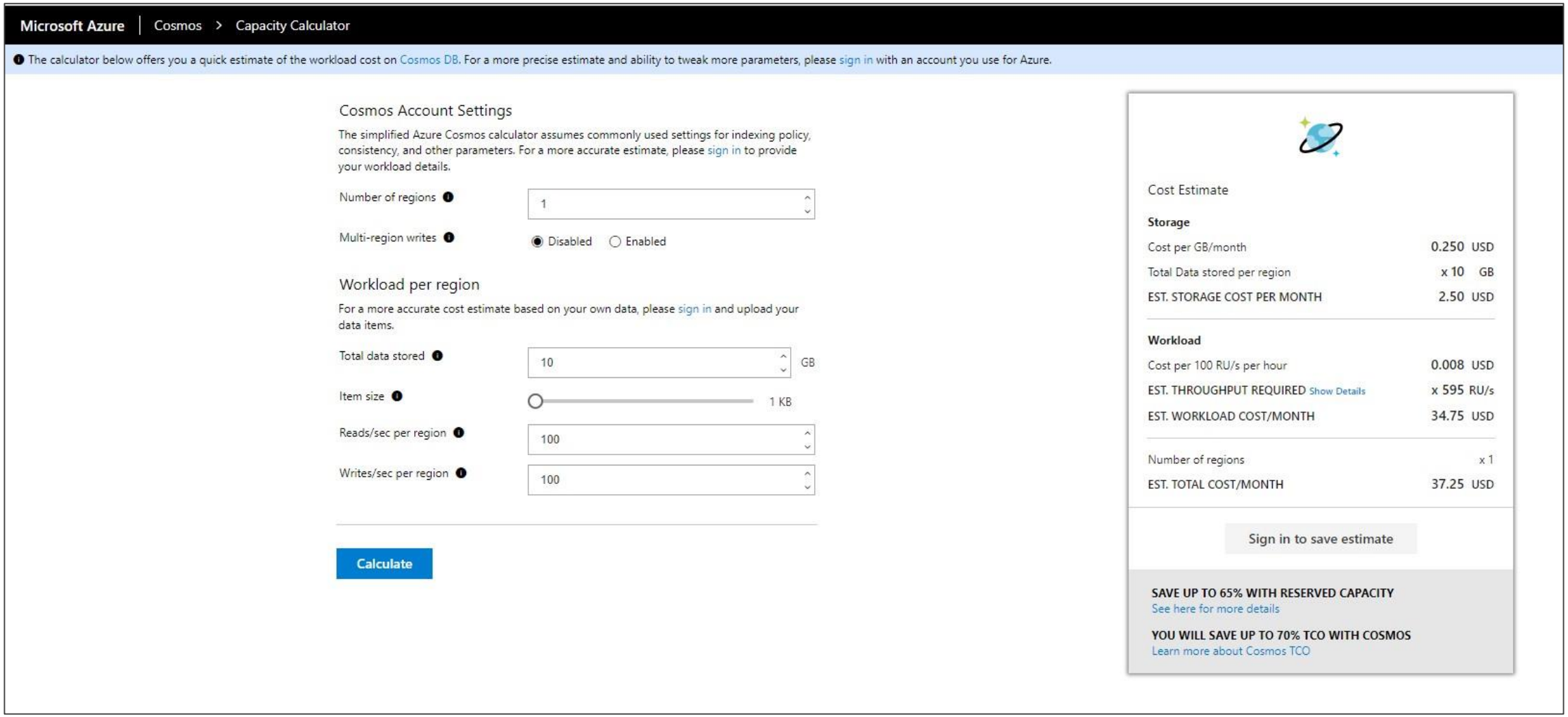
The size of an item effects the number of RUs consumed to read/write the item

The number of RUs consumed to write an item increases as the item property count increases

The frequency of CRUD operations (create, read, update, delete)

The complexity of queries (Query patterns)

Estimate the Throughput for Your Workload



You can change the number
of provisioned RUs
within seconds.

Choose the Partition Key

Partitioning is a key point of consideration before migrating to a globally distributed Database

Cosmos DB uses partitioning to scale individual containers in a database to meet the scalability and performance needs of your application

Follow the mentioned best practices to avoid “hot” partitions (choose the right partition key)

Index Your Data

Azure Cosmos DB indexes all your data fields upon ingestion

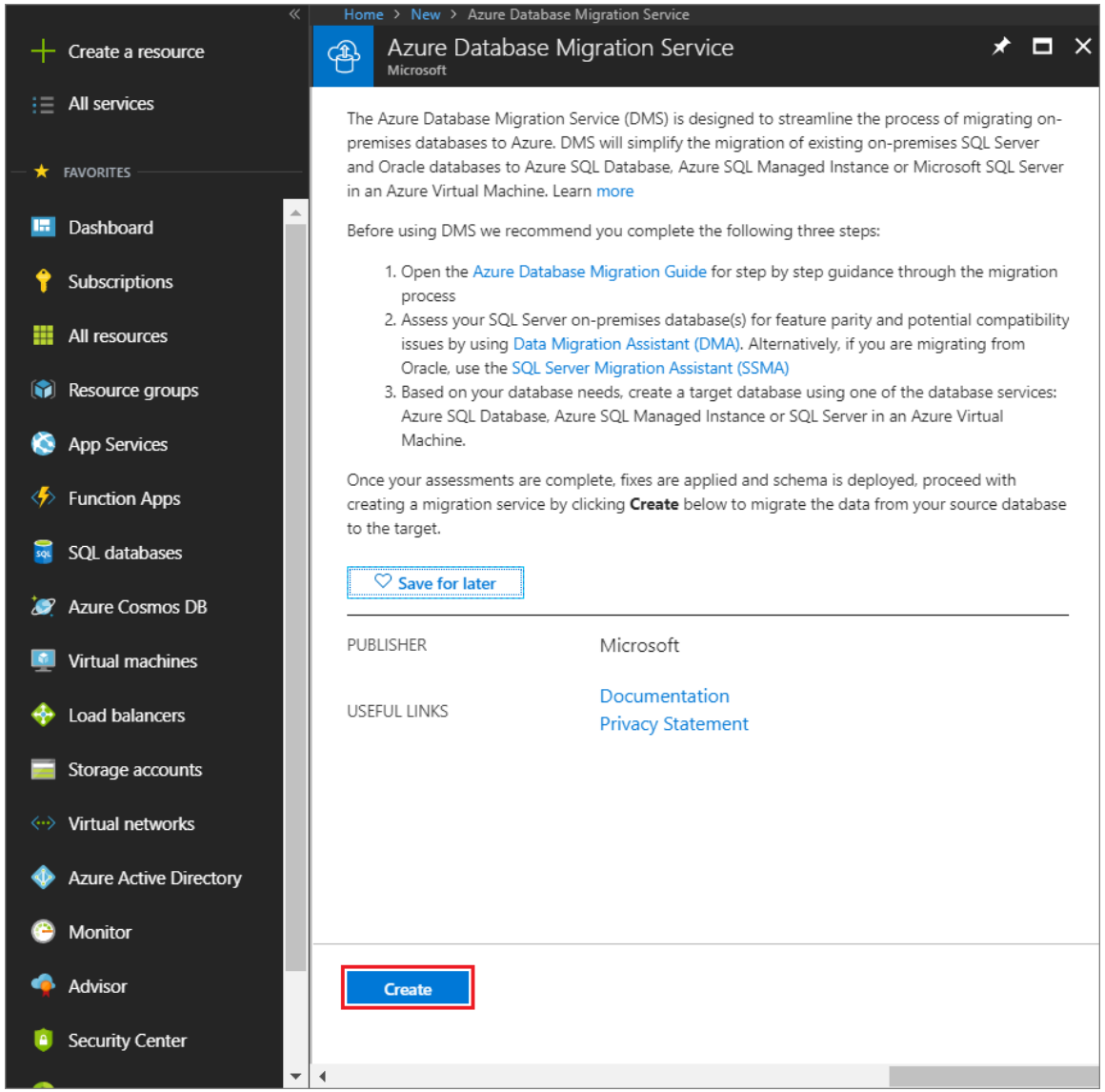
It is recommended to turn off indexing when migrating data

Azure DMS migrates MongoDB collections with unique indexes

The unique indexes must be created in destination before the migration

Can not create unique indexes, when there is already data in collections

Use Azure Database Migration Service



Post-migration Steps

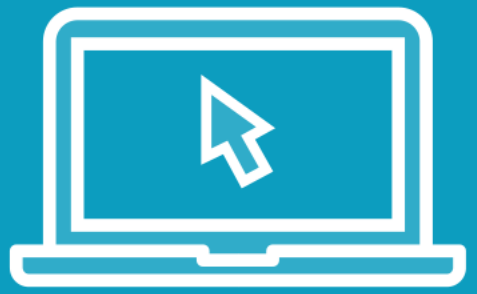
Connect your application

Optimize the indexing policy (optional)

Globally distribute your data (optional)

Set consistency level (optional)

Demo



Create a MongoDB database in MongoDB Atlas

- Skip this demo if you already have a MongoDB database to migrate

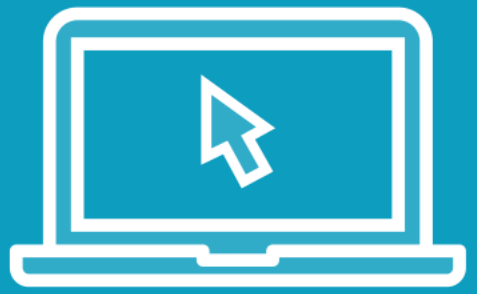
Demo

Using the MongoDB SDK in a .NET app

Pre-migration steps

- Provision a Cosmos DB MongoDB API instance
- Register Microsoft.DataMigration provider in your subscription

Demo



Migrate a Mongo DB database to Azure Cosmos DB

Azure Data Migration Service (DMS)

Update our .NET app to work with Azure Cosmos DB MongoDB

Summary

An overview of MongoDB

Why would you migrate to Cosmos DB MongoDB API?

What you need to do before & after migration

Demo: Migrating to Cosmos DB MongoDB API