

Microsoft Azure Cloud Services



Authorized & published by Summitworks Technologies Inc

Agenda

- **Azure services**
 - Data services
 - Azure SQL Database
 - Azure CosmosDB

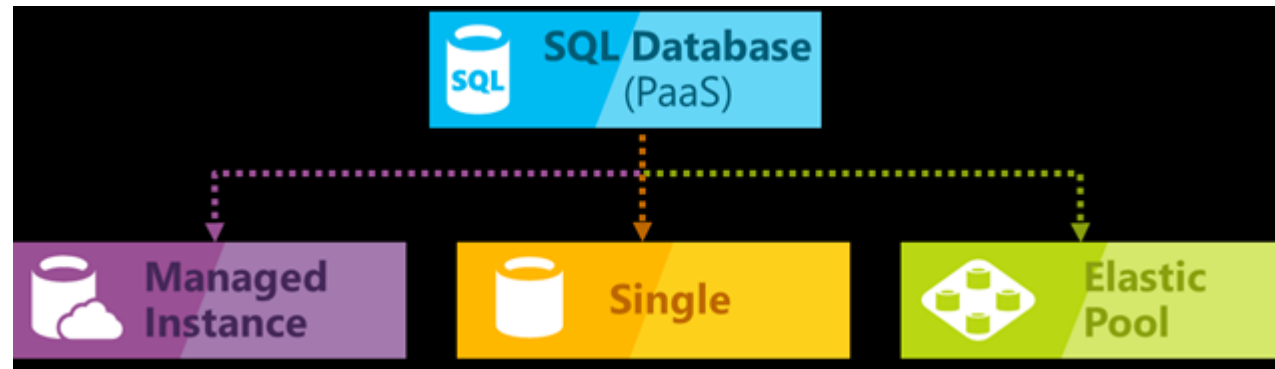
Azure Data Services - Azure SQL Database

Azure SQL Database service

SQL Database is a general-purpose relational database managed service in Microsoft Azure that supports structures such as relational data, JSON, spatial, and XML.

Azure SQL Database provides the following deployment options for an Azure SQL database:

- As a single database with its own set of resources managed via a logical server
- As a pooled database in an elastic pool with a shared set of resources managed via a logical server
- As a part of a collection of databases known as a managed instance that contains system and user databases and sharing a set of resources



Azure SQL Database service

SQL Database shares its code base with the Microsoft SQL Server database engine. With Microsoft's cloud-first strategy, the newest capabilities of SQL Server are released first to SQL Database, and then to SQL Server itself. This approach provides you with the newest SQL Server capabilities with no overhead for patching or upgrading - and with these new features tested across millions of databases.

Scalable performance and pools

SQL Database provides different compute sizes for different needs, and enables databases to be pooled to maximize the use of resources and save money.

Adjust performance and scale without downtime

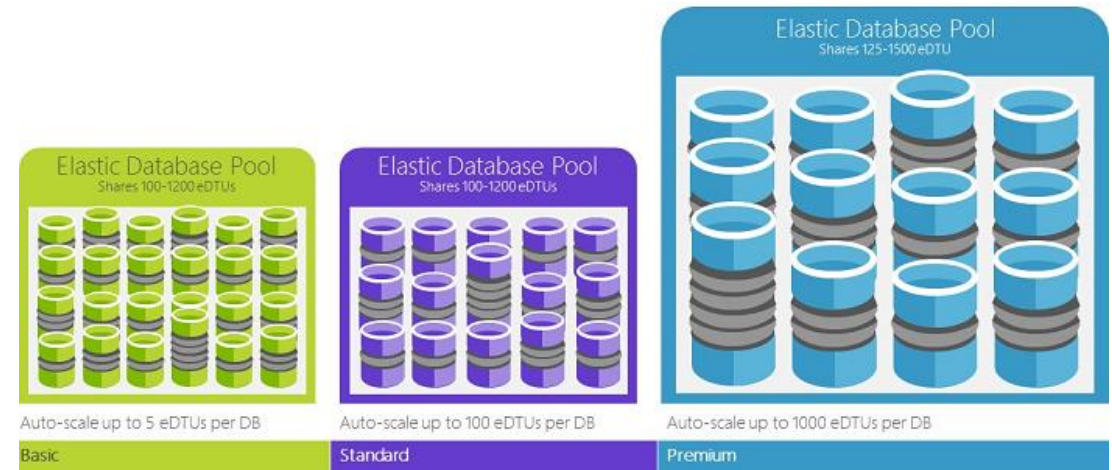
You can build your first app on a small, single database at a low cost per month in the General Purpose service tier and then change its service tier manually or programmatically at any time to the Business Critical Service tier to meet the needs of your solution.

Azure SQL Database service

Elastic pools to maximize resource utilization

For many businesses and applications, being able to create single databases and dial performance up or down on demand is enough, especially if usage patterns are relatively predictable. But if you have unpredictable usage patterns, it can make it hard to manage costs and your business model.

Elastic pools are designed to solve this problem. The concept is simple. You allocate performance resources to a pool rather than an individual database and pay for the collective performance resources of the pool rather than for single database performance.



Azure SQL Database service

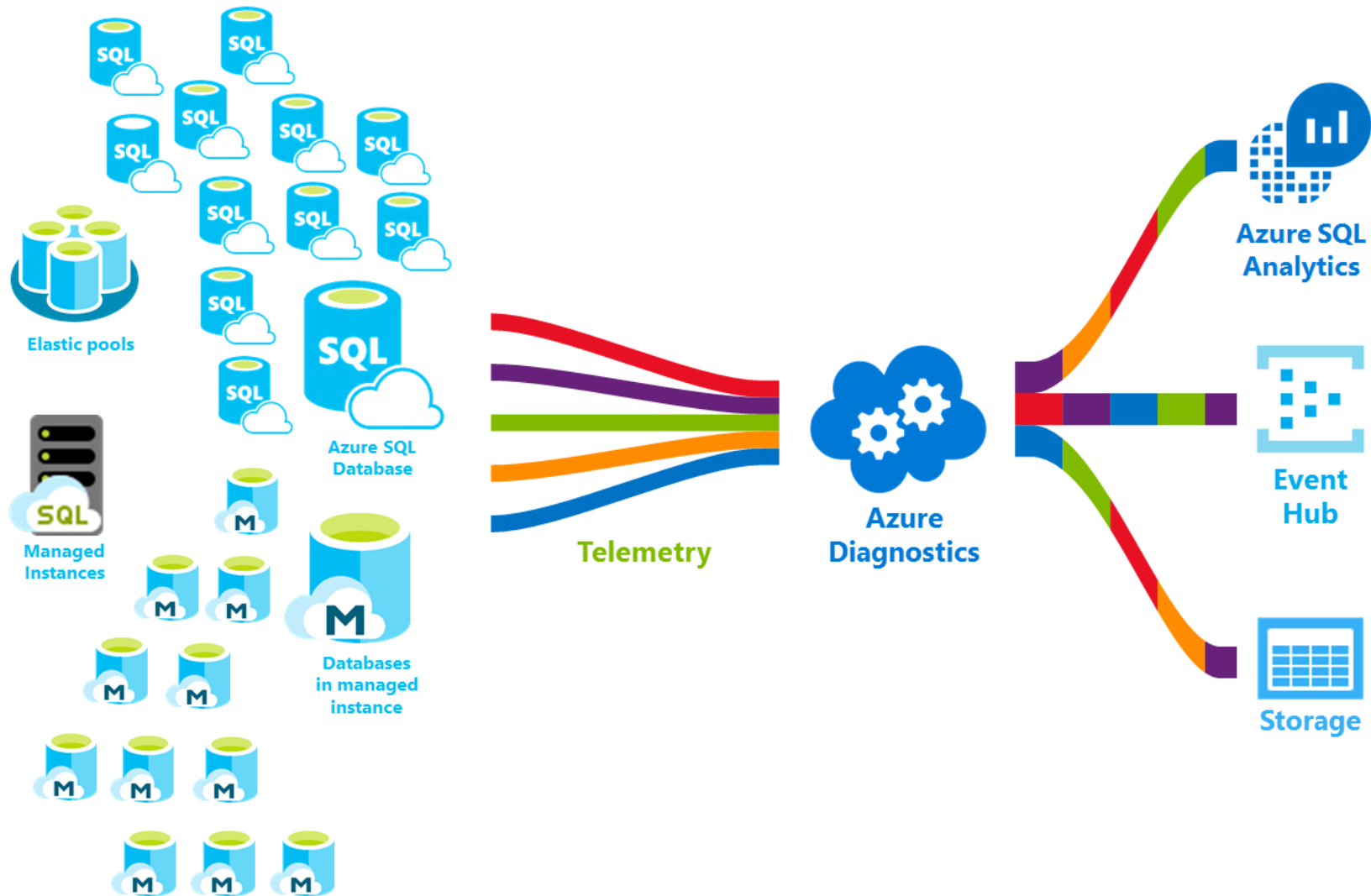
Extensive monitoring and alerting capabilities

But how can you compare the relative performance of single databases and elastic pools? How do you know the right click-stop when you dial up and down? You use the built-in performance monitoring and alerting tools, combined with the performance ratings. Using these tools, you can quickly assess the impact of scaling up or down based on your current or project performance needs.

Additionally, SQL Database can emit metrics and diagnostic logs for easier monitoring. You can configure SQL Database to store resource usage, workers and sessions, and connectivity into one of these Azure resources:

- **Azure Storage:** For archiving vast amounts of telemetry for a small price
- **Azure Event Hub:** For integrating SQL Database telemetry with your custom monitoring solution or hot pipelines
- **Azure Log Analytics:** For built-in monitoring solution with reporting, alerting, and mitigating capabilities.

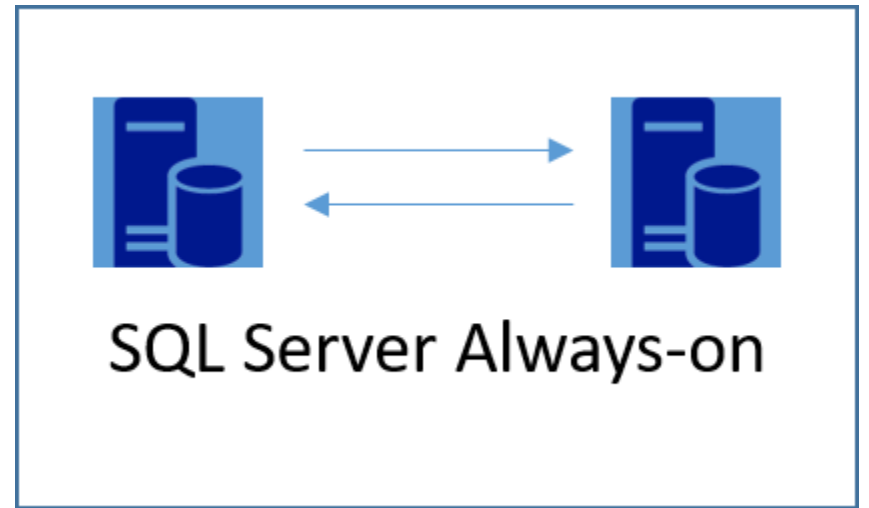
Azure SQL Database service



Azure SQL Database service

Availability capabilities

Azure's industry leading 99.99% availability service level agreement (SLA), powered by a global network of Microsoft-managed datacenters, helps keep your app running 24/7. The Azure platform fully manages every Azure SQL Database and guarantees no data loss and high percentage of data availability. Azure automatically handles patching, backups, replication, failure detection, underlying potential hardware, software or network failures, deploying bug fixes, failovers, database upgrades and other maintenance tasks.



Azure SQL Database service

SQL Database provides built-in business continuity and global scalability features, including:

- **Automatic backups:**

SQL Database automatically performs full, differential, and transaction log backups.

- **Point-in-time restores:**

SQL Database supports recovery to any point in time within the automatic backup retention period.

- **Active geo-replication:**

SQL Database allows you to configure up to four readable secondary databases in either the same or globally distributed Azure data centers. For example, if you have a SaaS application with a catalog database that has a high volume of concurrent read-only transactions, use active geo-replication to enable global read scale and remove bottlenecks on the primary that are due to read workloads.

Azure SQL Database service

- **Failover groups:**

SQL Database allows you to enable high availability and load balancing at global scale, including transparent geo-replication and failover of large sets of databases and elastic pools. Failover groups and active geo-replication enables creation of globally distributed SaaS applications with minimal administration overhead leaving all the complex monitoring, routing, and failover orchestration to SQL Database.

- **Zone-redundant databases:**

SQL Database allows you to provision Premium or Business Critical databases or elastic pools across multiple availability zones. Because these databases and elastic pools have multiple redundant replicas for high availability, placing these replicas into multiple availability zones provides higher resilience, including the ability to recover automatically from the datacenter scale failures without data loss.

Azure SQL Database service

Built-in intelligence

With SQL Database, you get built-in intelligence that helps you dramatically reduce the costs of running and managing databases and maximizes both performance and security of your application. Running millions of customer workloads around-the-clock, SQL Database collects and processes a massive amount of telemetry data, while also fully respecting customer privacy behind the scenes. Various algorithms are continuously evaluating the telemetry data so that the service can learn and adapt with your application. Based on this analysis, the service comes up with performance improving recommendations tailored to your specific workload.



Azure SQL Database service

Easy-to-use tools

SQL Database makes building and maintaining applications easier and more productive. SQL Database allows you to focus on what you do best: building great apps. You can manage and develop in SQL Database using tools and skills you already have.

- The Azure portal
- SQL Server Management Studio
- SQL Server Data Tools in Visual Studio
- Visual Studio Code

Create an Azure SQL database (Portal)

Create a SQL database

1. Click **Create a resource** in the upper left-hand corner of the Azure portal.
2. Select **Databases** from the **New** page, and select **Create** under **SQL Database** on the **New** page.

The screenshot shows the Azure portal's 'New' page. The breadcrumb navigation at the top reads 'Home > New > SQL Database'. The left sidebar lists various categories, with 'Databases' highlighted. The main area displays a 'Featured' list of database services, including 'Azure SQL Managed Instance', 'SQL Database', 'SQL Data Warehouse', 'SQL Elastic database pool', 'Azure Database for MariaDB (preview)', 'Azure Database for MySQL', and 'Azure Database for PostgreSQL'. The 'SQL Database' option is selected, opening a configuration pane on the right. This pane contains the following fields: 'Database name' (with a placeholder 'Enter database name'), 'Subscription' (set to 'BrightRaqce Subscription'), 'Resource group' (with a 'Select existing...' dropdown and a 'Create new' link), 'Select source' (set to 'Blank database'), 'Server' (with a 'Configure required settings' link), 'Want to use SQL elastic pool?' (with 'Yes' and 'Not now' radio buttons, 'Not now' is selected), 'Pricing tier' (with a 'Configure required settings' link and a lock icon), and 'Collation' (set to 'SQL_Latin1_General_CP1_CI_AS'). At the bottom of the pane are 'Create' and 'Automation options' buttons.

Create an Azure SQL database (Portal)

3. Fill out the SQL Database form with the following information.

SQL Database

×

* Database name

mySampleDatabase

✓

* Subscription

BrightRaqqe Subscription

▼

* Resource group

BrightraceRS

▼

Create new

* Select source ⓘ

Sample (AdventureWorksLT)

▼

* Server

Configure required settings

>

Create an Azure SQL database (Portal)

4. Under **Server**, click **Configure required settings** and fill out the SQL server (logical server) form with the following information.

New server

* Server name

brightceserver

.database.windows.net

* Server admin login

br1ghtrac3

* Password

.....

* Confirm password

.....

* Location

Central US

☒ Allow Azure services to access server

Advanced Threat Protection

Start FREE Trial

Not now

5. When you have completed the form, click **Select**.

Create an Azure SQL database (Portal)

- Click **Pricing tier** to specify the service tier, the number of DTUs, and the amount of storage. Explore the options for the amount of DTUs and storage that is available to you for each service tier.
- For this quickstart, select the **Basic** service tier and then use the slider to select **5 DTUs (S0)** and **2 GB** of storage.

Configure


♥ Feedback

Basic For less demanding workloads Starting at 4.99 USD / month	Standard For workloads with typical performance requirements Starting at 15.00 USD / month	Premium For IO-intensive workloads. Starting at 465.00 USD / month
--	---	---

DTUs [What is a DTU?](#)

5 (Basic)

Max data size

100 MB  2 GB

Create an Azure SQL database (Portal)

Query the SQL database

Now that you have created a sample database in Azure, let's use the built-in query tool within the Azure portal to confirm that you can connect to the database and query the data.

1. On the SQL Database page for your database, click **Query editor (preview)** in the left-hand menu and then click **Login**.
2. Select SQL server authentication, provide the required login information, and then click **OK** to log in.

Tags

Diagnose and solve problems

Quick start

Query editor (preview)

Settings

Configure

Geo-Replication

Connection strings

Sync to other databases

Add Azure Search

Properties

Locks

Automation script

SQL

Welcome to SQL Database Query Editor

Authorization type

SQL server authentication

* Login

br1ghtrac3

* Password

OK

OR

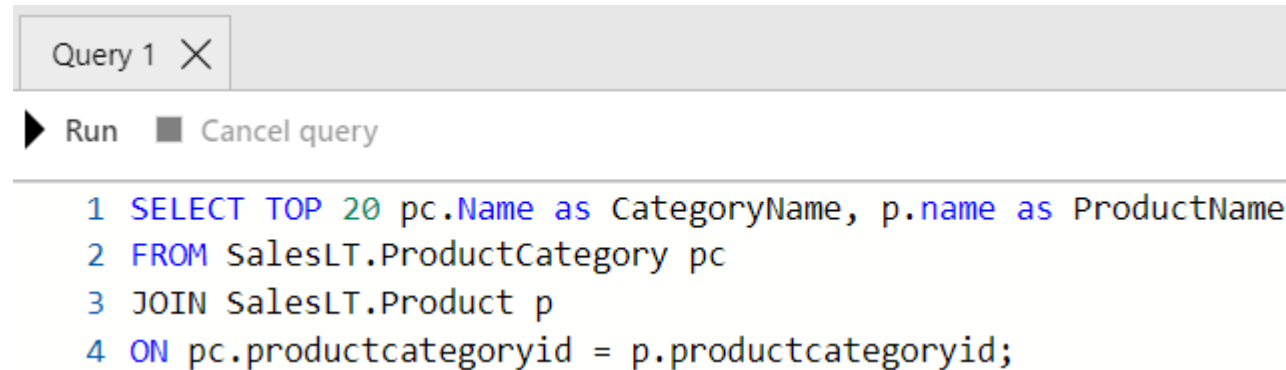
Active Directory single sign on

Error: User 'dubanand@outlook.com' is not authorized to connect to server 'brightraceserver.database.windows.net'.

OK

Create an Azure SQL database (Portal)

3. After you are authenticated as **ServerAdmin**, type the following query in the query editor pane.



```
1 SELECT TOP 20 pc.Name as CategoryName, p.name as ProductName
2 FROM SalesLT.ProductCategory pc
3 JOIN SalesLT.Product p
4 ON pc.productcategoryid = p.productcategoryid;
```

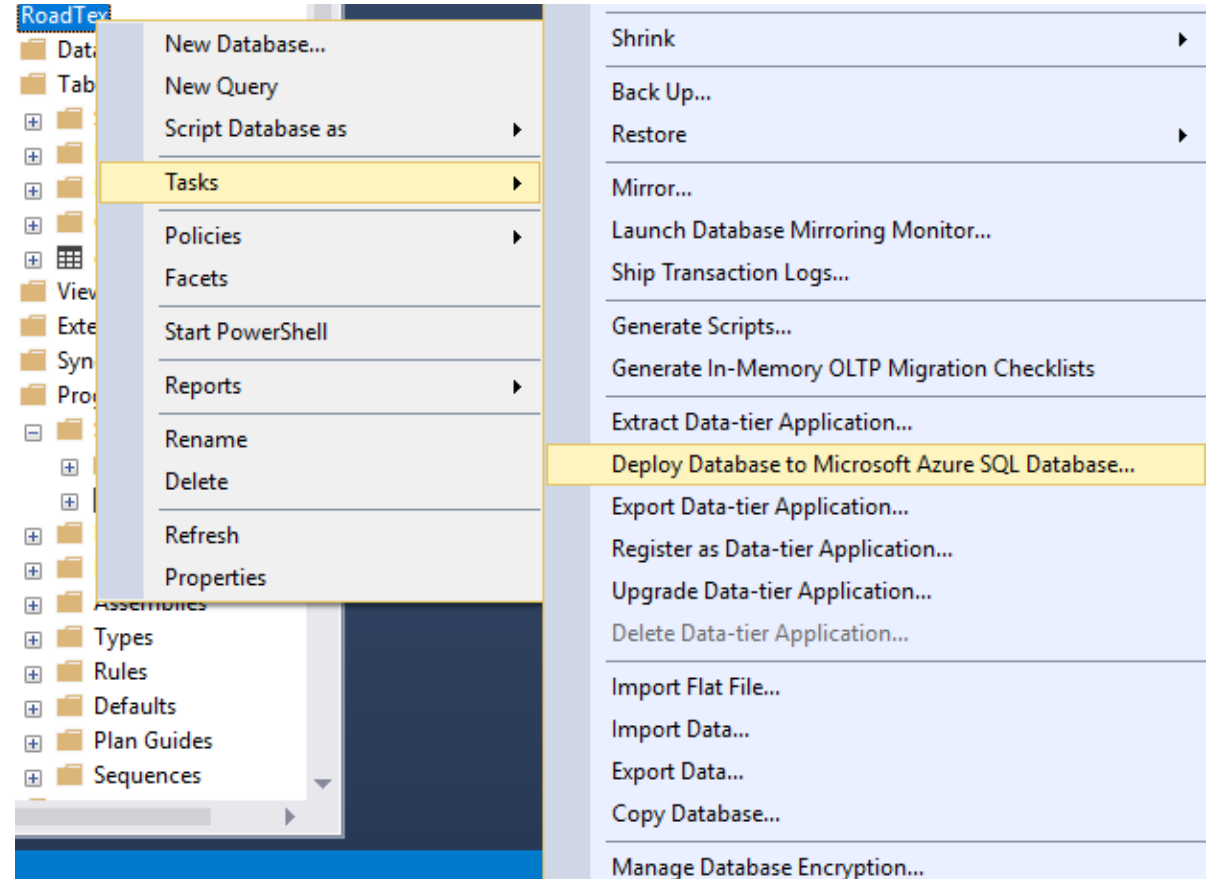
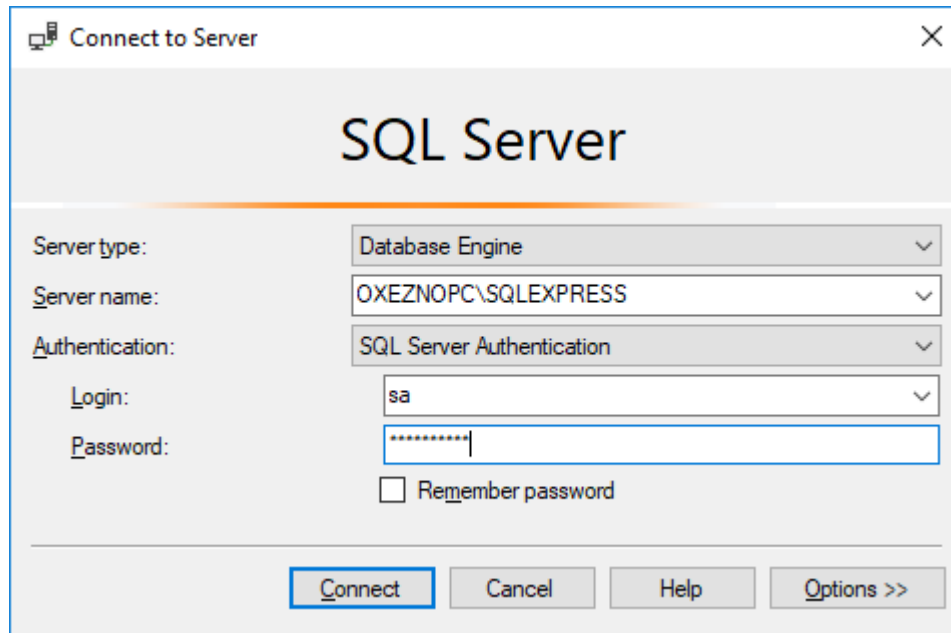
Create an Azure SQL database (Portal)

4. Click **Run** and then review the query results in the **Results** pane.

Results Messages	
<input type="text" value="Search to filter items..."/>	
CATEGORYNAME	PRODUCTNAME
Road Frames	HL Road Frame - Black, 58
Road Frames	HL Road Frame - Red, 58
Helmets	Sport-100 Helmet, Red
Helmets	Sport-100 Helmet, Black

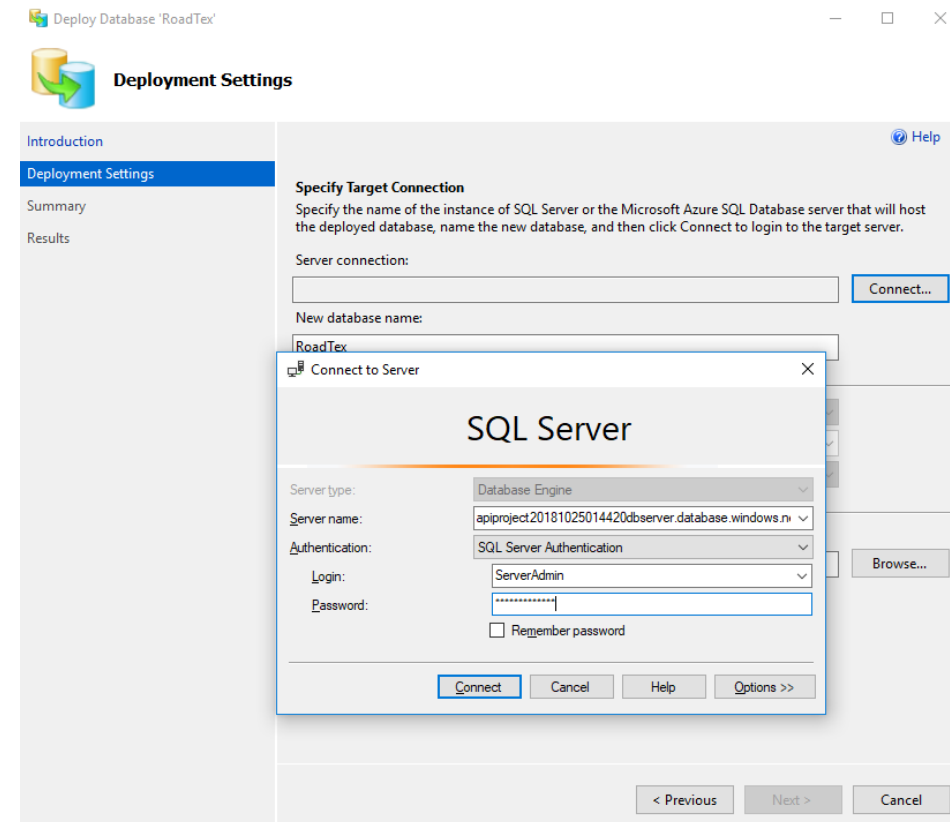
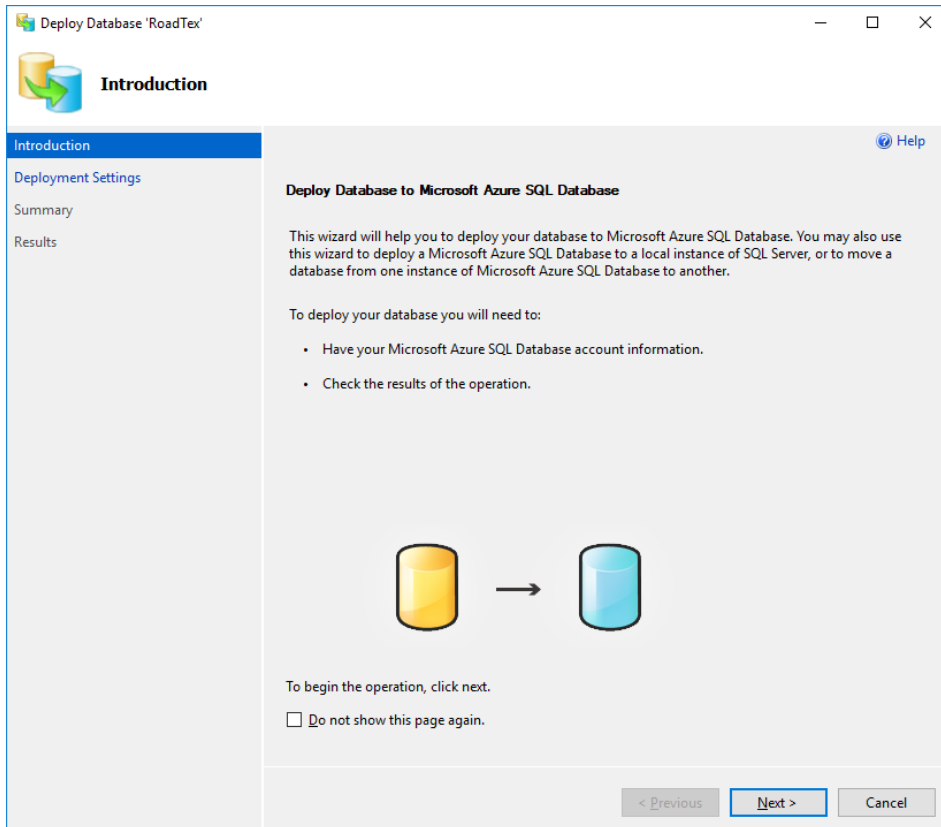
SQL Server Management Studio to SQL Azure

1. Connect to your local DB and select Tools from menu
2. Right click on your DB and select Task/Deploy Database to Microsoft Azure SQL Database



SQL Server Management Studio to SQL Azure

3. Connect to Azure Server using your Azure SQL credentials.



SQL Server Management Studio to SQL Azure

4. Finish the Deploy.

The screenshot shows the 'Deploy Database 'RoadTex'' window with the 'Deployment Settings' tab selected. The left sidebar contains 'Introduction', 'Deployment Settings', 'Summary', and 'Results'. The main area is titled 'Specify Target Connection' and includes instructions: 'Specify the name of the instance of SQL Server or the Microsoft Azure SQL Database server that will host the deployed database, name the new database, and then click Connect to login to the target server.' The 'Server connection:' field contains 'apiproject20181025014420dbserver (ServerAdmin)' with a 'Connect...' button. The 'New database name:' field contains 'RoadTex'. Below, 'Microsoft Azure SQL Database settings' are shown with 'Edition of Microsoft Azure SQL Database' set to 'Basic', 'Maximum database size (GB)' set to '2', and 'Service Objective' set to 'Basic'. The 'Other settings' section shows 'Temporary file name' as 'C:\Users\Oxezno\AppData\Local\Temp\RoadTex-20181025140639.bacpac' with a 'Browse...' button. At the bottom are '< Previous', 'Next >', and 'Cancel' buttons.

The screenshot shows the 'Deploy Database 'RoadTex'' window with the 'Summary' tab selected. The left sidebar contains 'Introduction', 'Deployment Settings', 'Summary', and 'Results'. The main area is titled 'Verify Specified Settings' with the instruction: 'To complete the operation using the specified settings, click Finish.' It displays a tree view of settings: 'Source' (Name: OXEZNOPC, Connected as: sa, Database: RoadTex, Source Database Size: 16.0 MB) and 'Target' (Name: apiproject20181025014420dbserver, Connected as: ServerAdmin, Database: RoadTex, Edition: Basic, Maximum Database Size: 2 GB, Service Objective: Basic). An 'Environment' section shows 'Temporary File Location: C:\Users\Oxezno\AppData\Local\Temp\RoadTex-20181025140639.bacpac' and 'Size available for temporary file: 801.6 GB'. At the bottom are '< Previous', 'Finish', and 'Cancel' buttons.

SQL Server Management Studio to SQL Azure

4. Finish the Deploy.

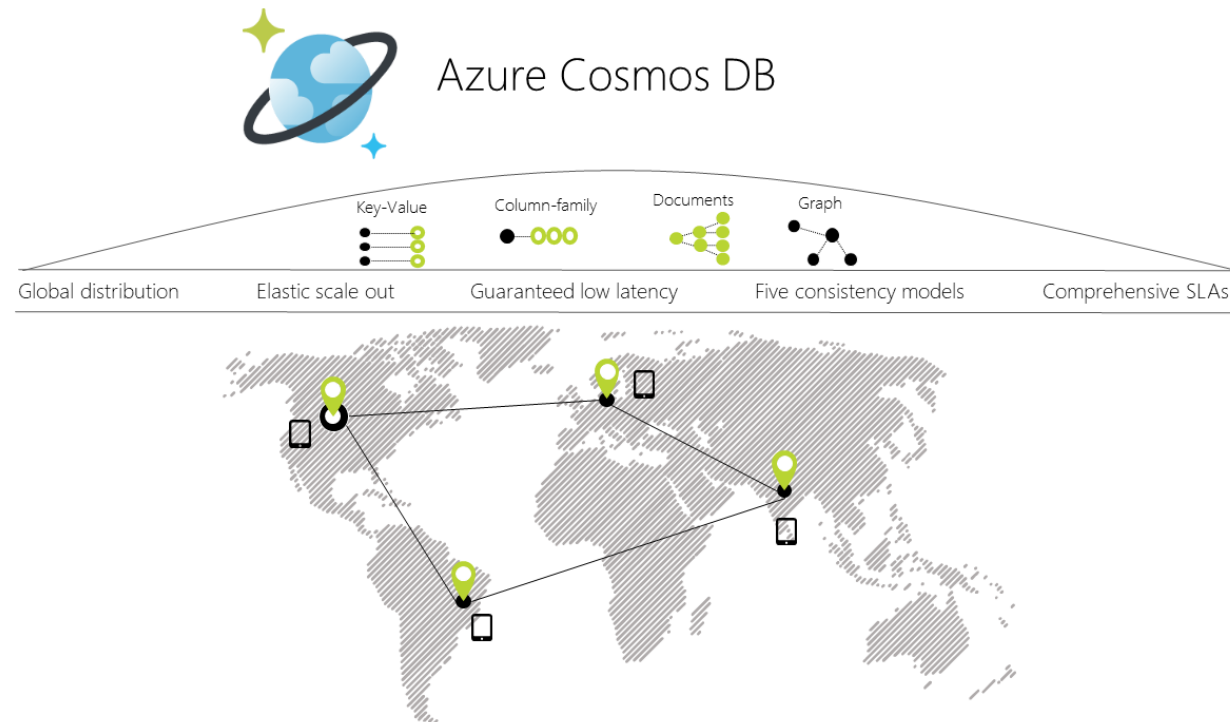
The screenshot shows the 'Deploy Database 'RoadTex'' window with the 'Deployment Settings' tab selected. The left sidebar contains links for 'Introduction', 'Deployment Settings', 'Summary', and 'Results'. The main area is titled 'Specify Target Connection' and includes instructions: 'Specify the name of the instance of SQL Server or the Microsoft Azure SQL Database server that will host the deployed database, name the new database, and then click Connect to login to the target server.' The 'Server connection:' field contains 'apiproject20181025014420dbserver (ServerAdmin)' with a 'Connect...' button. The 'New database name:' field contains 'RoadTex'. Below, 'Microsoft Azure SQL Database settings' are shown with 'Edition of Microsoft Azure SQL Database' set to 'Basic', 'Maximum database size (GB)' set to '2', and 'Service Objective' set to 'Basic'. 'Other settings' include a 'Temporary file name:' field with the path 'C:\Users\Oxezno\AppData\Local\Temp\RoadTex-20181025140639.bacpac' and a 'Browse...' button. At the bottom are '< Previous', 'Next >', and 'Cancel' buttons.

The screenshot shows the 'Deploy Database 'RoadTex'' window with the 'Summary' tab selected. The left sidebar is the same as the previous window. The main area is titled 'Verify Specified Settings' with the instruction: 'To complete the operation using the specified settings, click Finish.' Below this is a tree view showing 'Source' and 'Target' details. The 'Source' section shows: Name: OXEZNOPC, Connected as: sa, Database: RoadTex, and Source Database Size: 16.0 MB. The 'Target' section shows: Name: apiproject20181025014420dbserver, Connected as: ServerAdmin, Database: RoadTex, Edition: Basic, Maximum Database Size: 2 GB, and Service Objective: Basic. An 'Environment' section shows: Temporary File Location: C:\Users\Oxezno\AppData\Local\Temp\RoadTex-20181025140639.bacpac and Size available for temporary file: 801.6 GB. At the bottom are '< Previous', 'Finish', and 'Cancel' buttons.

Azure Data Services - Azure Cosmos DB

Azure Cosmos DB

Azure Cosmos DB is Microsoft's globally distributed, multi-model database. With the click of a button, Azure Cosmos DB enables you to elastically and independently scale throughput and storage across any number of Azure's geographic regions. It offers throughput, latency, availability, and consistency guarantees with comprehensive service level agreements (SLAs), something no other database service can offer.



Azure Cosmos DB Key Capabilities

- **Turnkey global distribution**
 - You can distribute your data to any number of Azure regions, with the click of a button. This enables you to put your data where your users are.
 - Using Azure Cosmos DB's multi-homing APIs, the app always knows where the nearest region is and sends requests to the nearest data center. All of this is possible with no config changes. You set your write-region and as many read-regions as you want, and the rest is handled for you.
- **Elastically and independently scale throughput and storage on demand and worldwide**
 - Easily scale database throughput at a per-second granularity, and change it anytime you want.
 - Scale storage size transparently and automatically to handle your size requirements now and forever.

Azure Cosmos DB Key Capabilities

- **Multiple data models and popular APIs for accessing and querying data**

APIs for the following data models are supported with SDKs available in multiple languages:

- SQL API
- MongoDB API
- Cassandra API
- Gremlin API
- Table API

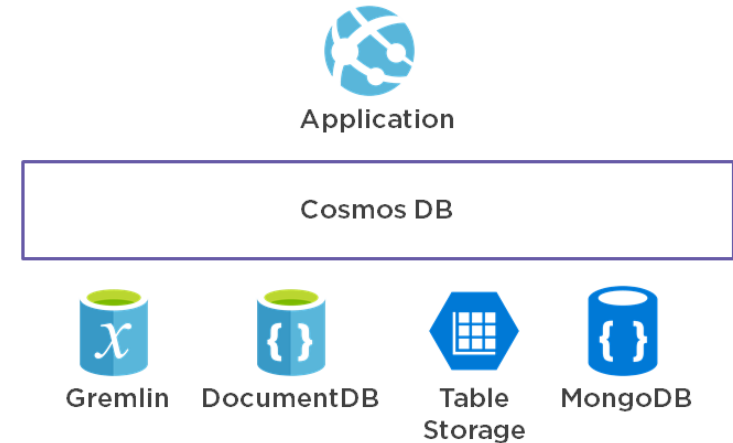
- **Ensure "always on" availability**

- 99.99% availability SLA for all single region database accounts, and all 99.999% read availability on all multi-region database accounts.
- Deploy to any number of Azure regions for higher availability and better performance.

Azure Cosmos DB

Solutions that benefit from Azure Cosmos DB

Any web, mobile, gaming, and IoT application that needs to handle massive amounts of data, reads, and writes at a global scale with near-real response times for a variety of data will benefit from Azure Cosmos DB's guaranteed high availability, high throughput, low latency, and tunable consistency.



Build a .NET app with Cosmos DB / SQL API

Now lets learn how to create an Azure Cosmos DB SQL API account, document database, and collection using the Azure portal. You'll then build and deploy a todo list web app built on the SQL .NET API.

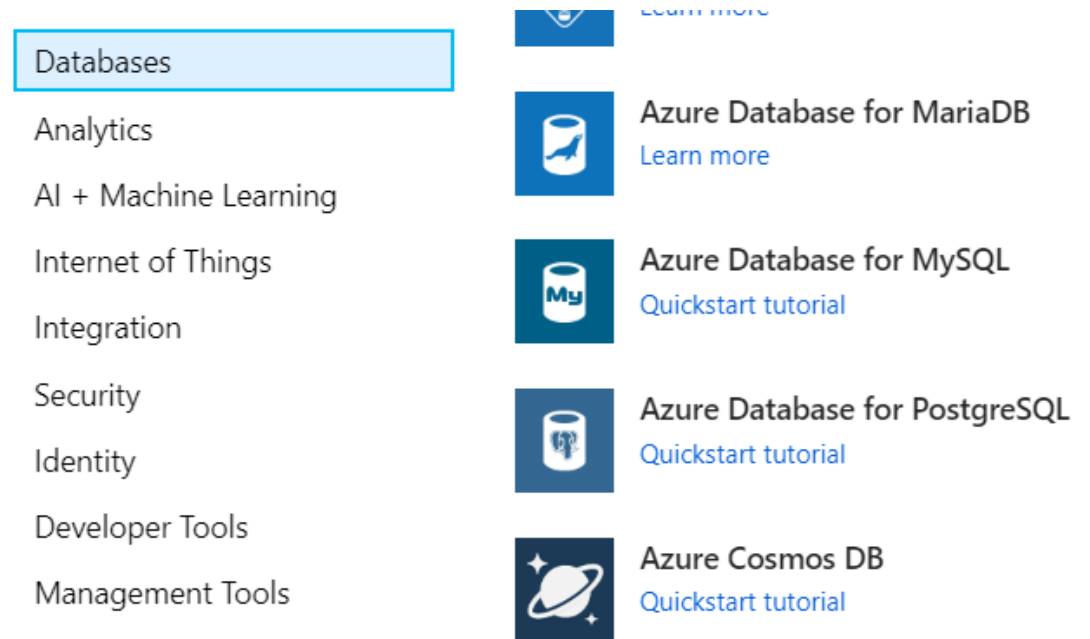
Create a database account

1. Sign in to the Azure portal.
2. Select

Create a resource

> Databases

> Azure Cosmos DB.



Build a .NET app with Cosmos DB / SQL API

3. On the **New account** page, enter the settings for the new Azure Cosmos DB account.
4. Select **Create**.

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription

* Resource Group [Create new](#)

INSTANCE DETAILS

* Account Name documents.azure.com ✓

* API ⓘ

* Location

Geo-Redundancy ⓘ

Multi-region Writes ⓘ

Build a .NET app with Cosmos DB / SQL API

5. The account creation takes a few minutes. Wait for the portal to display the **Congratulations! Your Azure Cosmos DB account was created** page.

✔ Your deployment is complete

[Go to resource](#)



Deployment name: Microsoft.Azure.CosmosDB-20181204142653
Subscription: [BrightRaqce Subscription](#)
Resource group: [BrightraceRS](#)

DEPLOYMENT DETAILS [\(Download\)](#)

Start time: 12/4/2018, 2:33:56 PM

Duration: 3 minutes 56 seconds

Correlation ID: 18e226a4-0a85-4fea-be2c-362703725f75

Build a .NET app with Cosmos DB / SQL API

Add a collection

You can now use the Data Explorer tool in the Azure portal to create a database and collection.

1. Click **Data Explorer > New Collection**.
The **Add Collection** area is displayed on the far right.
2. In the **Add collection** page, enter the settings for the new collection.

Add Collection

×

* Database id ⓘ

☒ Create new ☐ Use existing

Tasks

* Collection Id ⓘ

Items

* Storage capacity ⓘ

Fixed (10 GB)

Unlimited

* Throughput (400 - 10,000 RU/s) ⓘ

1000

−

+

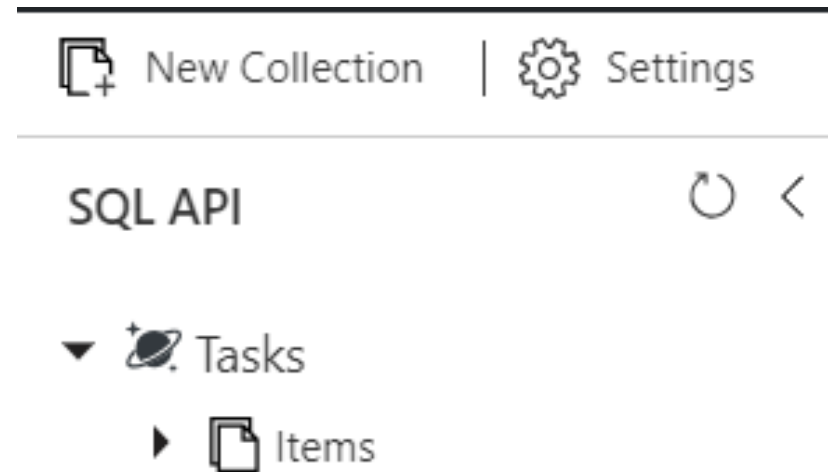
Choose unlimited storage capacity for more than 10,000 RU/s.

Unique keys ⓘ

+ Add unique key

Build a .NET app with Cosmos DB / SQL API

Data Explorer displays the new database and collection.

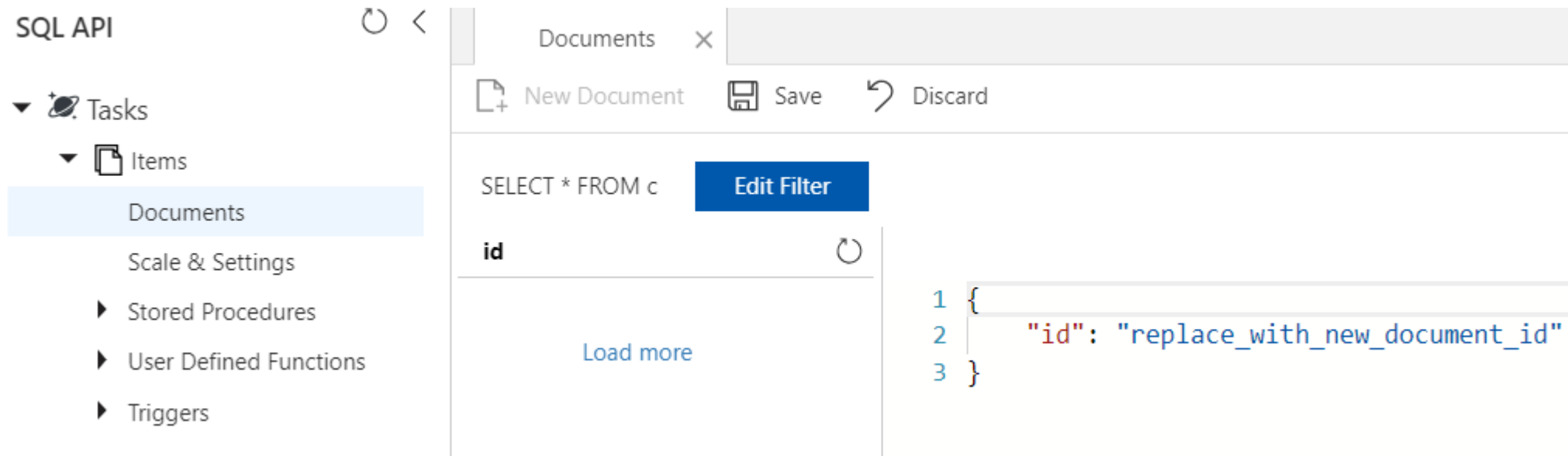


Build a .NET app with Cosmos DB / SQL API

Add sample data

You can now add data to your new collection using Data Explorer.

1. In Data Explorer, the new database appears in the Collections pane. Expand the **Tasks** database, expand the **Items** collection, click **Documents**, and then click **New Documents**.



Build a .NET app with Cosmos DB / SQL API

2. Now add a document to the collection with the following structure.

```
1 {  
2   "id": "1",  
3   "category": "personal",  
4   "name": "groceries",  
5   "description": "Pick up apples and strawberries.",  
6   "isComplete": false  
7 }
```

3. Once you've added the json to the **Documents** tab, click **Save**.

Build a .NET app with Cosmos DB / SQL API

4. Create and save one more document where you insert a unique value for the id property, and change the other properties as you see fit. Your new documents can have any structure you want as Azure Cosmos DB doesn't impose any schema on your data.

```
1 {  
2   "id": "51",  
3   "name": "Daniel",  
4   "lastname": "Jackes",  
5   "address": "147 Road Street, N.J.",  
6   "phone": "857-552-2114",  
7   "email": "danjc@mimail.com"  
8 }
```

5. Once you've added the json to the **Documents** tab, click **Save**.

Build a .NET app with Cosmos DB / SQL API

Query your data

You can now use queries in Data Explorer to retrieve and filter your data.

1. See that by default, the query is set to ***SELECT * FROM c***. This default query retrieves and displays all documents in the collection.

SELECT * FROM c

Edit Filter

id

1

51

Load more

```
1 {
2   "id": "1",
3   "category": "personal",
4   "name": "groceries",
5   "description": "Pick up apples and strawberries.",
6   "isComplete": false,
7   "_rid": "H6tUALiBU0gBAAAAAAAAA==",
8   "_self": "dbs/H6tUAA==/colls/H6tUALiBU0g=/docs/H6tUALiBU0gBAAAAAAAAA==/",
9   "_etag": "\"00000000-0000-0000-8c12-df616f2501d4\"",
10  "_attachments": "attachments/",
11  "_ts": 1543956578
12 }
```

Build a .NET app with Cosmos DB / SQL API

- Stay on the Documents tab, and change the query by clicking the Edit Filter button, adding *ORDER BY c._ts DESC* to the query predicate box, and then clicking Apply Filter.

SELECT * FROM c ORDER BY c._ts DESC

Apply Filter

id
51
1

Load more

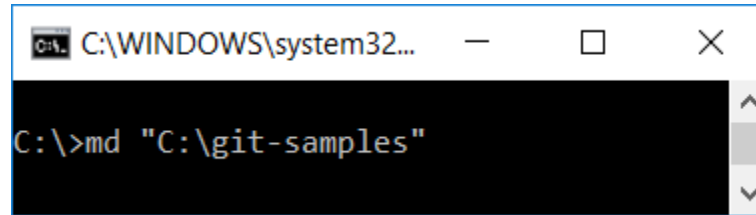
```
1 {
2   "id": "1",
3   "category": "personal",
4   "name": "groceries",
5   "description": "Pick up apples and strawberries.",
6   "isComplete": false,
7   "_rid": "H6tUALiBU0gBAAAAAAAAA==",
8   "_self": "dbs/H6tUAA==/colls/H6tUALiBU0gBAAAAAAAAA==/",
9   "_etag": "\"00000000-0000-0000-8c12-df616f2501d4\"",
10  "_attachments": "attachments/",
11  "_ts": 1543956578
12 }
```

Build a .NET app with Cosmos DB / SQL API

Clone the sample application

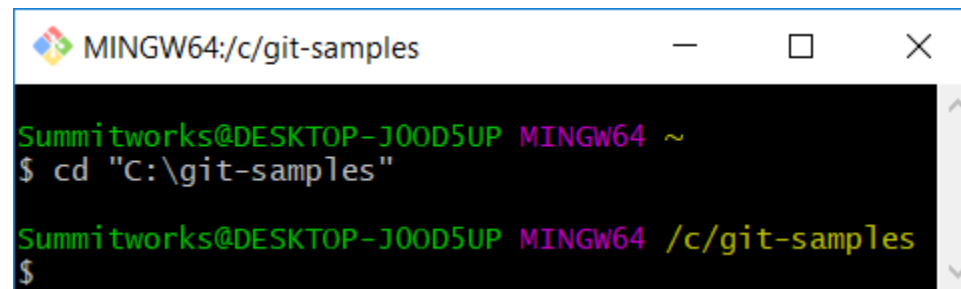
Now let's switch to working with code. Let's clone a SQL API app from GitHub, set the connection string, and run it. You'll see how easy it is to work with data programmatically.

1. Open a command prompt, create a new folder named git-samples, then close the command prompt.



```
C:\WINDOWS\system32...  
C:\>md "C:\git-samples"
```

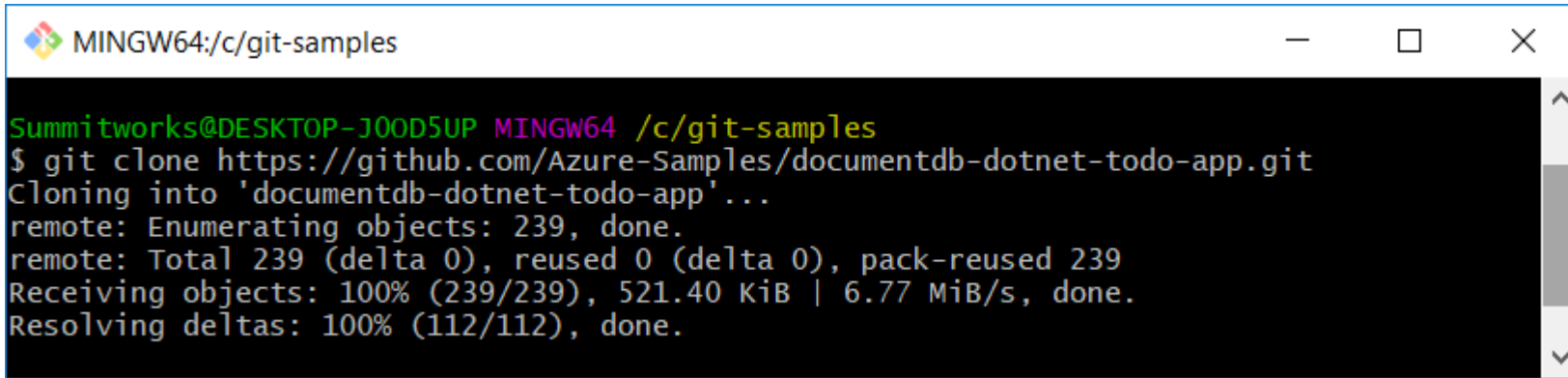
2. Open a git terminal window, such as git bash, and use the cd command to change to the new folder to install the sample app.



```
MINGW64:/c/git-samples  
Summitworks@DESKTOP-JOOD5UP MINGW64 ~  
$ cd "C:\git-samples"  
Summitworks@DESKTOP-JOOD5UP MINGW64 /c/git-samples  
$
```

Build a .NET app with Cosmos DB / SQL API

3. Run the following command to clone the sample repository. This command creates a copy of the sample app on your computer.



```
MINGW64:/c/git-samples  
  
Summitworks@DESKTOP-J00D5UP MINGW64 /c/git-samples  
$ git clone https://github.com/Azure-Samples/documentdb-dotnet-todo-app.git  
Cloning into 'documentdb-dotnet-todo-app'...  
remote: Enumerating objects: 239, done.  
remote: Total 239 (delta 0), reused 0 (delta 0), pack-reused 239  
Receiving objects: 100% (239/239), 521.40 KiB | 6.77 MiB/s, done.  
Resolving deltas: 100% (112/112), done.
```

4. Then open the todo solution file in Visual Studio.

Build a .NET app with Cosmos DB / SQL API

Update your connection string

Now go back to the Azure portal to get your connection string information and copy it into the app.

1. In the Azure portal, in your Azure Cosmos DB account, in the left navigation select **Keys**, and then select **Read-write Keys**. You'll use the copy buttons on the right side of the screen to copy the URI and Primary Key into the web.config file in the next step.

URI

```
https://localhost:8081
```

Primary Key

```
C2y6yDjf5/R+ob0N8A7Cgv30VRDJIWEHLM+4QDU5DE2nQ9nDuVTqobD4b8mGGyPMblZnqyMsEcaGQy67Xlw/Jw==
```

Primary Connection String

```
AccountEndpoint=https://localhost:8081/;AccountKey=C2y6yDjf5/R+ob0N8A7Cgv30VRDJIWEHLM+4QDU5DE2nQ9nDuVTqobD4b8mGGyPMblZnqyMsEcaGQy67Xlw/Jw==
```

Mongo Connection String

```
mongodb://localhost:C2y6yDjf5/R+ob0N8A7Cgv30VRDJIWEHLM+4QDU5DE2nQ9nDuVTqobD4b8mGGyPMblZnqyMsEcaGQy67Xlw/Jw==@localhost:10255/admin?ssl=true
```

Build a .NET app with Cosmos DB / SQL API

2. In Visual Studio 2017, open the web.config file.
3. Copy your URI value from the portal (using the copy button) and make it the value of the endpoint key in web.config.

```
<add key="endpoint" value="FILLME" />
```

4. Then copy your PRIMARY KEY value from the portal and make it the value of the authKey in web.config.

```
<add key="authKey" value="FILLME" />
```

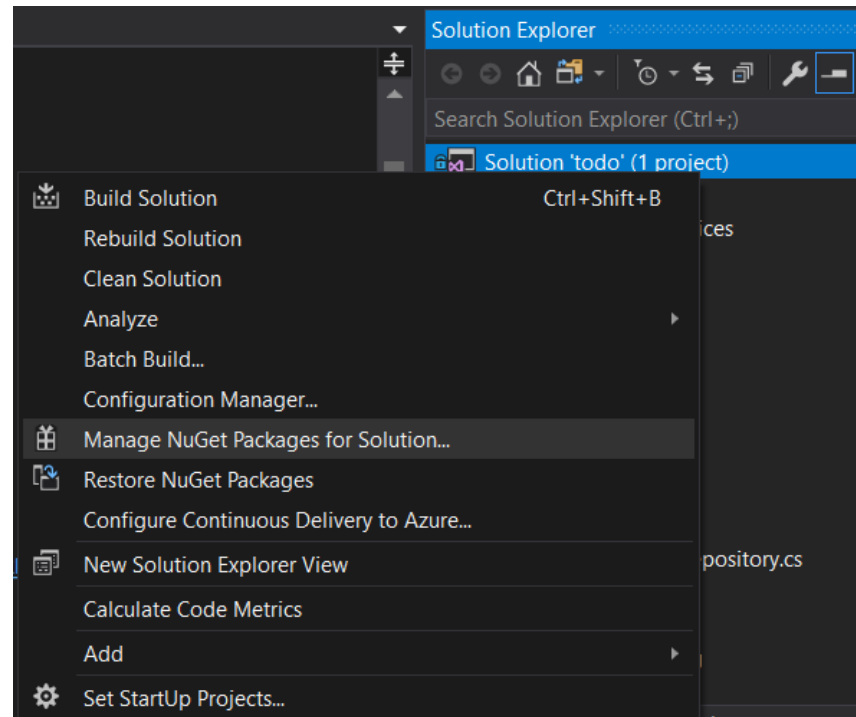
5. Then update the database value to match the name of the database you have created earlier. You've now updated your app with all the info it needs to communicate with Azure Cosmos DB.

```
<add key="database" value="Tasks" />
```

Build a .NET app with Cosmos DB / SQL API

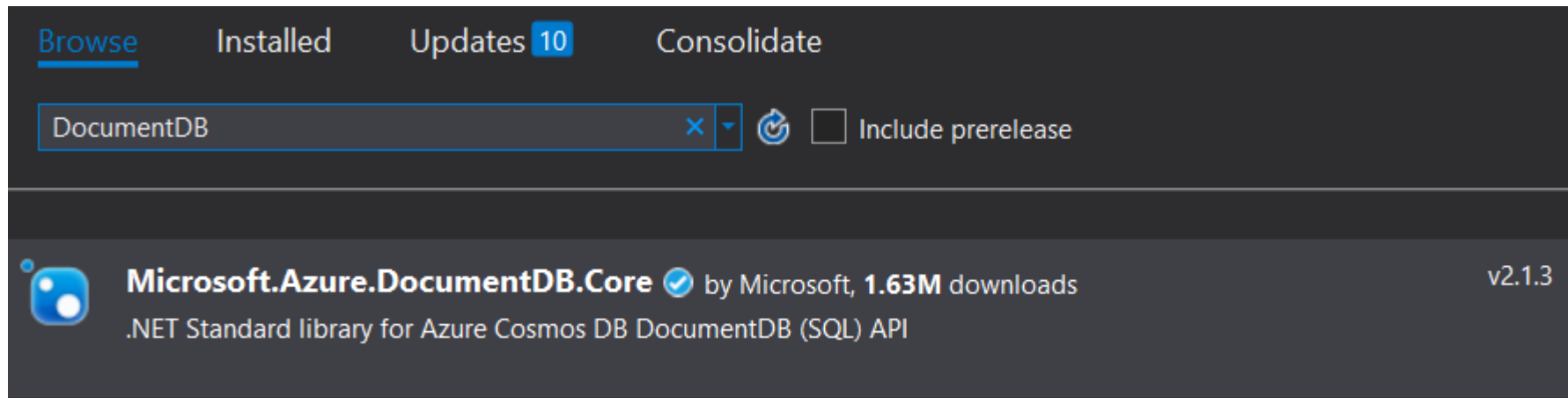
Run the web app

1. In Visual Studio, right-click on the project in **Solution Explorer** and then select **Manage NuGet Packages**.



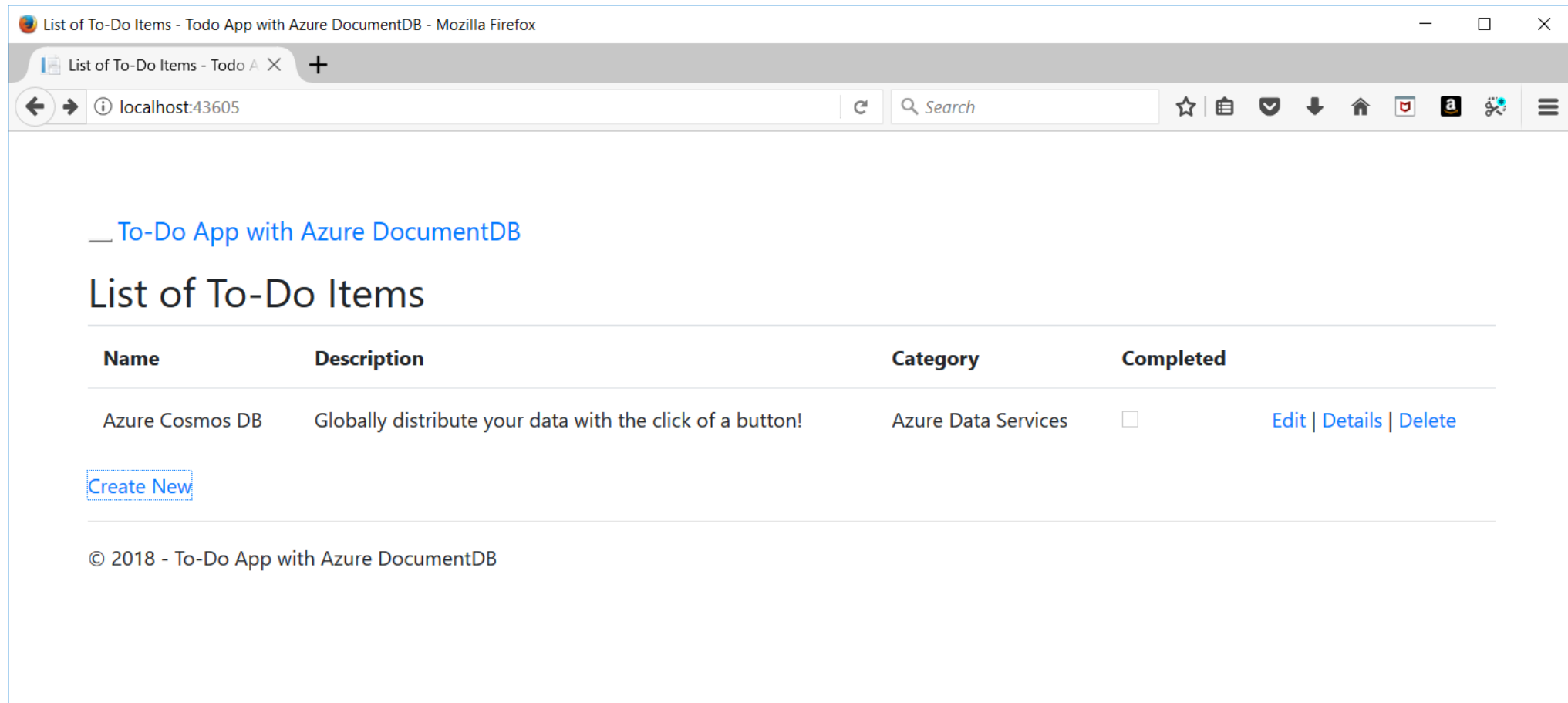
Build a .NET app with Cosmos DB / SQL API

2. In the NuGet **Browse** box, type *DocumentDB*.
3. From the results, install the **Microsoft.Azure.DocumentDB** library. This installs the Microsoft.Azure.DocumentDB package as well as all dependencies.



Build a .NET app with Cosmos DB / SQL API

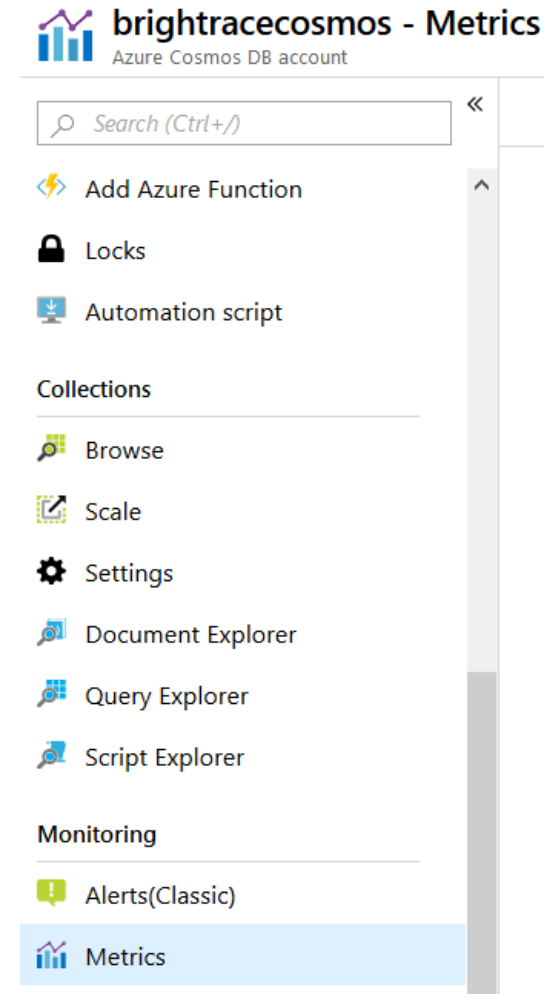
4. Select CTRL + F5 to run the application. Your app displays in your browser.



Review SLAs in the Azure portal

The throughput, storage, availability, latency, and consistency of the resources in your account are monitored in the Azure portal. Let's take a quick look at these metrics.

1. Click **Metrics** in the navigation menu.



Review SLAs in the Azure portal

2. Click through each of the tabs so you're aware of the metrics Azure Cosmos DB provides.

Each chart that's associated with the Azure Cosmos DB Service Level Agreements (SLAs) provides a line that shows if any of the SLAs have been violated. Azure Cosmos DB makes monitoring your SLAs transparent with this suite of metrics.

