

Cosmos DB React JS

A. Project Demo Link:

https://mediaspace.illinois.edu/media/t/1_rg741qe3

B. Project Overview:

- Our project focuses on the seamless integration of data extracted from the Youtube API with Azure Cosmos DB, utilizing Cosmos DB's robust features to store and query social graph data obtained from the Youtube platform.
- The primary objective is to develop a web application interface that demonstrates secure querying of the graph data stored in Cosmos DB.
- Our project will emphasize system architecture, performance, security, and database features to ensure an efficient and reliable solution.

C. Data Extraction from Youtube:

- Our data extraction process began by leveraging the YouTube API, facilitated by the Google API client library and Python.
- Utilizing an API key obtained from the Google Developer Console, we gained access to the YouTube Data API, enabling us to authenticate requests.
- Subsequently, we built functions to collect video IDs from specified playlists and to retrieve comments, including replies, for each video as well as scraping comments from all the pages in the comment section of that video.
- Comments and their associated metadata were acquired, including timestamps, usernames, and like counts, we systematically organized this information into Python data structures.
- To streamline analysis and manipulation, we transformed the data into a pandas DataFrame. Finally, we exported the structured data into a JSON format.

D. Upload the Data:

- To upload the extracted YouTube comments data to Azure Cosmos DB, we first obtained the necessary credentials from our Cosmos DB account.
- This involved retrieving the endpoint URL and the primary key, which authenticate our access to the database.
- With these credentials in hand, we instantiated a DocumentClient object from the pydocumentdb library, providing the endpoint URL and the master key.
- Next, we iterated over the JSON data extracted from the YouTube comments file to effectively transfers the YouTube comments data into Azure Cosmos DB, where it is stored and made accessible for further analysis and querying.

The entire process of uploading the data took about 1 hour given the size of the dataset which was scraped from using the API.

E. After data is uploaded:

a. Network Settings:

- The “Network” settings the firewall rules where we restricted the access to my CosmosDB account to only my team members IP address.
- This was done to ensure that security is achieved and the public access to my database is restricted.

b. Role Assignment:

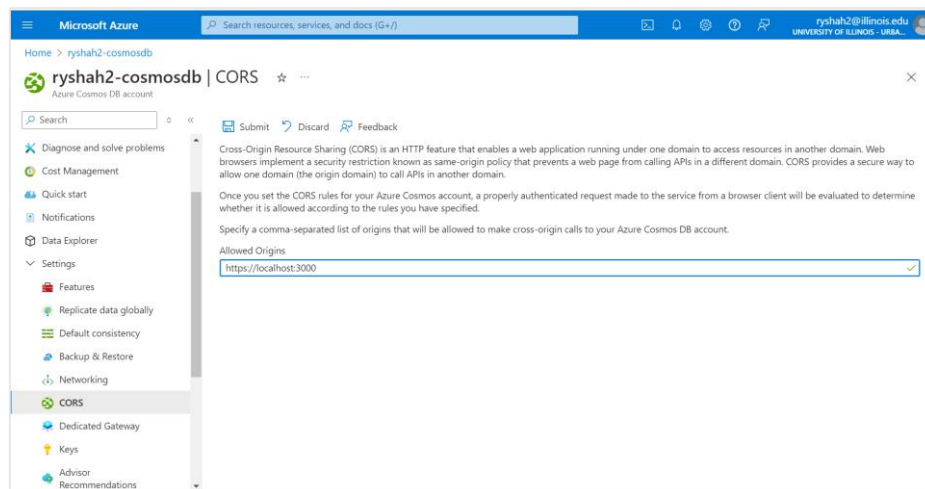
- Through the Azure CLI, accessed via Windows PowerShell, we attempted to assign a read-write role to my team member, Moksha through the the Azure CLI command `az

cosmosdb sql role assignment create`. To accomplish this, I needed to input Moksha's object ID, which was retrieved from the Azure Active Directory. Additionally, I referenced the role definition ID obtained from the Azure documentation to specify the particular role I wanted to assign her.

- As a result, Moksha can now effectively collaborate on managing the Cosmos DB resources within our team.

c. CORS Policy:

- To connect the React web application to our backend we had to enable the CORS policy. Cross-Origin Resource Sharing (CORS) is an HTTP feature that enables a web application running under one domain to access resources in another domain.
- Inside the Allowed Origins section, we had to enter “localhost:3000” which is where our React web application was running.



F. User Interface:

a. Application Structure:

- app.js: Serves as the entry point for the React application, utilizing React Router for navigation.
- Home.js: Represents the homepage, dynamically rendering embedded YouTube videos linked to their respective comments sections.
- styles.css: Contains styling rules ensuring a cohesive and attractive user interface.

b. Cosmos DB to ReactJS Integration (DataDisplay.js):

- MAX_COMMENTS_LIMIT: Defines the maximum number of comments to fetch for performance optimization.
- Function Component: Defines DataDisplay component, initializing state variables for comments data, loading state, error handling, and comment count.
- Fetching Data Effect: Utilizes useEffect hook to fetch comments data from Azure Cosmos DB.
- Queries: Performs two queries on the Cosmos DB container - one to fetch total comment count and another to retrieve comments data for the current video.
- Rendering: Conditionally renders loading indicators, errors, or comments grid based on the current state. Displays comments along with relevant information like username, video ID, and like count.

G. Queries:

Query 1: Retrieving Top 5 Comments with Highest Likes for a Video:

This query retrieves the top 5 comments with the highest number of likes for a specific video. By sorting comments based on their like count in descending order and selecting the top 5, we can quickly identify the most popular comments for any given video. This is particularly useful for understanding user engagement and sentiment towards the video content.

Home	Comme...Items	Comme...Que... x
<pre>1 -- Query 1 2 SELECT TOP 5 * 3 FROM CommentsCollection c 4 WHERE c.VideoID = 'Fsgdznlfe2U' 5 ORDER BY c.LikeCount DESC 6</pre>		
Results	Query Stats	
1 - 5		
<pre>{ { "Timestamp": "2023-10-27T10:03:32Z", "Username": "@BigRockyGaming", "VideoID": "Fsgdznlfe2U", "Comment": "I didn't think this song could get better. It had party vibes always. This version just brings them out so much in a fun way", "Date": "2023-10-27T10:03:32Z", "LikeCount": 2313, "Total_Replies": null, "id": "951cdfe-f73a-4ab9-9f4d-01406d200aba", "_rid": "ekwVAKMqr1S1DQAAAAA==", "_self": "dbs/ekwVAA=/colls/ekwVAKMqr1Q=/docs/ekwVAKMqr1S1DQAAAAA==/", "_etag": "\"0d00afbc-0000-0200-0000-662591b60000\"", "_attachments": "attachments/", "_ts": 1713738166 }, { "Timestamp": "2024-03-09T03:12:01Z", "Username": "@MadalynBoone", "VideoID": "Fsgdznlfe2U", "Comment": "Only true swifties can like this comment♥♥♥", "Date": "2024-03-09T03:12:01Z", "LikeCount": 1460, "Total_Replies": null, "id": "9979b4f7-19e5-4773-9d92-d538283df5aa", "_rid": "ekwVAKMqr1R1C1DQAAAAA==", "_self": "dbs/ekwVAA=/colls/ekwVAKMqr1Q=/docs/ekwVAKMqr1R1C1DQAAAAA==/", "_etag": "\"0d00afbc-0000-0200-0000-662591b60000\"", "_attachments": "attachments/", "_ts": 1713738166 } }</pre>		

Query 2: Finding Comments Containing Specific Keywords

This query searches for comments containing specific keywords, 'Swiftly' or 'Swifties'. By using the 'CONTAINS' function, we can filter comments based on their content, allowing us to analyze sentiments, trends, or topics discussed within the comments section. This helps us gain insights into user interests or reactions related to particular themes or subjects.

Home	Comme...Items	Comme...Que... x
<pre>1 -- Query 2 2 SELECT * FROM CommentsCollection c 3 WHERE CONTAINS(c.Comment, 'Swiftly') OR CONTAINS(c.Comment, 'Swifties') 4 5</pre>		
Results	Query Stats	
1 - 100 Load more		
<pre>{ { "Timestamp": "2024-04-17T11:54:16Z", "Username": "@LillyEvans123", "VideoID": "Fsgdznlfe2U", "Comment": "Swifties 🐣", "Date": "2024-04-17T11:54:16Z", "LikeCount": 1, "Total_Replies": null, "id": "ddb9287e-0cea-4678-bba2-0fbb0aaba383", "_rid": "ekwVAKMqr1Q2AAAAA==", "_self": "dbs/ekwVAA=/colls/ekwVAKMqr1Q=/docs/ekwVAKMqr1Q2AAAAA==/", "_etag": "\"0d0043af-0000-0200-0000-662591170000\"", "_attachments": "attachments/", "_ts": 1713738007 }, { "Timestamp": "2024-04-14T15:08:47Z", "Username": "@AlbertTurikumenayo", "VideoID": "Fsgdznlfe2U", "Comment": "I think I'm a real Swiftly because my birthday is on her new album", "Date": "2024-04-14T15:08:47Z", "LikeCount": 1, "Total_Replies": null, "id": "f9f8a047-84fb-40d2-9ca4-6dd2bf59e3e3", "_rid": "ekwVAKMqr1R1C1DQAAAAA==", "_self": "dbs/ekwVAA=/colls/ekwVAKMqr1Q=/docs/ekwVAKMqr1R1C1DQAAAAA==/", "_etag": "\"0d00afbc-0000-0200-0000-662591b60000\"", "_attachments": "attachments/", "_ts": 1713738166 } }</pre>		

Query 3: Identifying the First Commenter on a Video:

This query determines the first user to comment on a specific video. By ordering comments based on their creation date in ascending order and selecting the top result, we can pinpoint the earliest interaction with the video content. This information is valuable for understanding user engagement dynamics and can be used for various analytical purposes, such as studying the initial response to newly published content.

The screenshot shows the Azure Cosmos DB Data Explorer interface. At the top, a warning message states: "To prevent queries from using excessive RUs, Data Explorer has a 5,000 RU default limit. To modify or remove the limit, go to the Settings cog on the right and find 'RU Threshold'. [Learn More](#)". Below this, the query editor shows the following SQL query:

```
-- Query 3
1 SELECT TOP 5 c.username
2 FROM CommentsCollection c
3 WHERE c.VideoID = 'fsgdzlFE2u'
4 ORDER BY c.Date ASC
```

The results tab shows a list of 5 usernames, ordered by date:

```
{
  "Username": "@sarahhutchins6105"
},
{
  "Username": "@kk12901"
},
{
  "Username": "@paxtonkirkley602"
},
{
  "Username": "@snowdrop-4907"
},
{
  "Username": "@ihaveabigcrushonmadihillpovicz"
}
```

Query Performance metrics:

- The Cosmos DB Data Explorer offers insightful statistics on query performance and resource utilization, providing valuable insights into query execution efficiency and database health. Metrics like Request Units (RUs) represent the computational resources consumed by a query. The higher the value, the more resources it consumed. It's a measure of the query's complexity and the amount of work performed by the database engine to execute it.
- Query Engine Execution Time also denoted the time taken by the query engine to execute the query logic. This includes parsing, optimization, and execution of the query.

H. Backup and restore:

In our application, we've seamlessly integrated continuous backup functionality, leveraging the robust capabilities of Azure Cosmos DB. This means that every time our database is updated, a new backup is triggered automatically. We've demonstrated the power of point-in-time restore by effortlessly restoring our database to a previous state, even after records were deleted.

The screenshot shows the Azure Cosmos DB Point In Time Restore interface. The left sidebar contains a navigation menu with options: Search, Default consistency, Point In Time Restore (selected), Networking, CORS, Dedicated Gateway, Keys, Advisor Recommendations, Microsoft Defender for Cloud, Identity, Locks, Integrations (Power BI, Azure Synapse Link, Add Azure AI Search, Add Azure Function), and Containers. The main panel is titled "mshah15 | Point In Time Restore" and includes tabs for "Restore to new account", "Restore to same account", and "Backup policy". The "Restore to same account" tab is active. The "Backup source" section includes fields for "Restore Point (UTC)" (04/29/2024 00:00:00), "Location" (East US 2), and "Restore Resource" (Selected database/containers, CommentsCollection). The "Restore destination" section includes fields for "Resource group" (mshah15-CosmosDB) and "Restore target account name" (backup-2). On the right, there are links for "Need help with identifying restore point? Search the event feed", "How much will this restore cost? View an estimate", and "Required permissions To perform a restore for this account, the following is required... Learn more".

Home > mshah15-backup1

Azure Cosmos DB account

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Cost Management

Quick start

Notifications

Data Explorer

Settings

Features

Replicate data globally

Default consistency

Point In Time Restore

Networking

CORS

Dedicated Gateway

Keys

Welcome to your Azure Cosmos DB Free Tier account! Your first 1000 RUs and 25 GB of storage will be free for the lifetime of this account. Click here to learn more.

Essentials

Status: Online

Resource group: mshah15-CosmosDB

Subscription: Azure for Students

Subscription ID: f6c3a8fe-e241-4236-8099-7ceec0b33d8

Total throughput limit: No total throughput limit

Read Locations: East US 2

Write Locations: East US 2

URI: https://mshah15-backup1.documents.azure.com:443/

Free Tier Discount: Opted In

Capacity mode: Provisioned throughput

Containers

ID	Database	Throughput (RU/s)
CommentsCollection	YouTubeDB	400

Monitoring

Show data for last: 1 hour 14 hours 7 days 30 days

Requests

Estimated Cost (hourly)

Throughput

Storage

Home > mshah15

mshah15 | Point In Time Restore

Azure Cosmos DB account

Search

Default consistency

Point In Time Restore

Networking

CORS

Dedicated Gateway

Keys

Restore to new account

Restore to same account

Backup policy

Backup policy

Azure Cosmos DB provides two continuous mode backup policies. [Learn more](#) about the differences of the backup policies and pricing details.

☒ Continuous (7 days)

Provides backup window of 7 days / 168 hours and you can restore to any point of time within the window. This mode is available for free.

☐ Continuous (30 days)

Provides backup window of 30 days / 720 hours and you can restore to any point of time within the window. This mode has cost impact.

Home > mshah15-backup1

mshah15-backup1 | Data Explorer

Azure Cosmos DB account

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Cost Management

Quick start

Notifications

Data Explorer

Settings

Features

Replicate data globally

Default consistency

Point In Time Restore

Networking

CORS

Dedicated Gateway

To prevent queries from using excessive RUs, Data Explorer has a 5,000 RU default limit. To modify or remove the limit, go to the Settings cog on the right and find "RU Threshold". [Learn More](#)

DATA

YouTubeDB

CommentsCollection

Items

Scale & Settings

Stored Procedures

User Defined Functions

Triggers

SELECT * FROM c

id

/partitionkey

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

```
{
  "timestamp": "2024-04-28T23:45:07Z",
  "username": "BronEscobar2390",
  "videoID": "Psd0n1Fe2U",
  "comment": "How 🤔🤔 these songs are amazing 🎵🎵",
  "date": "2024-04-28T23:45:07Z",
  "likeCount": 0,
  "total_replies": null,
  "id": "5ed5e5b5-f7ad-463c-bf55-34c028c70ea",
  "_rid": "5ed5e5b5-f7ad-463c-bf55-34c028c70ea",
  "_self": "db/5ed5e5b5-f7ad-463c-bf55-34c028c70ea/documents/5ed5e5b5-f7ad-463c-bf55-34c028c70ea",
  "_etag": "\"0000000f-0000-0200-0000-002091140000\"",
  "_attachments": "attachments/",
  "ts": 1713738884
}
```

Home > mshah15-backup1

mshah15-backup1 | Data Explorer

Azure Cosmos DB account

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Cost Management

Quick start

Notifications

Data Explorer

Settings

Features

Replicate data globally

Default consistency

Point In Time Restore

Networking

CORS

Dedicated Gateway

To prevent queries from using excessive RUs, Data Explorer has a 5,000 RU default limit. To modify or remove the limit, go to the Settings cog on the right and find "RU Threshold". [Learn More](#)

DATA

YouTubeDB

CommentsCollection

Items

Scale & Settings

Stored Procedures

User Defined Functions

Triggers

SELECT * FROM c

id

/partitionkey

1

2

3

4

5

6

7

8

9

10

11

12

13

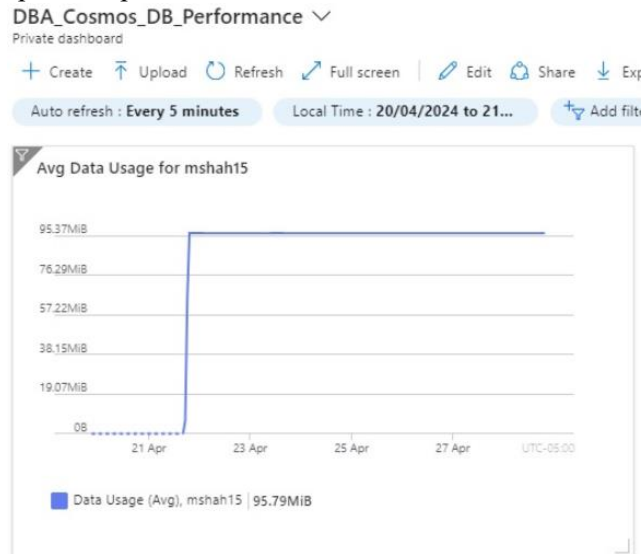
14

15

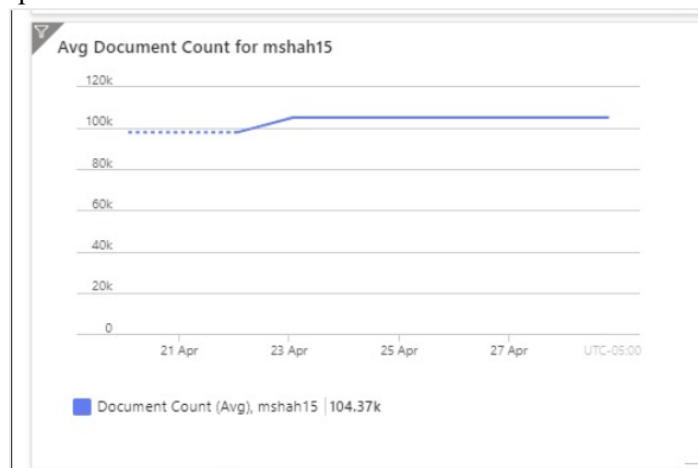
```
{
  "timestamp": "2024-04-28T23:45:07Z",
  "username": "BronEscobar2390",
  "videoID": "Psd0n1Fe2U",
  "comment": "How 🤔🤔 these songs are amazing 🎵🎵",
  "date": "2024-04-28T23:45:07Z",
  "likeCount": 0,
  "total_replies": null,
  "id": "5ed5e5b5-f7ad-463c-bf55-34c028c70ea",
  "_rid": "5ed5e5b5-f7ad-463c-bf55-34c028c70ea",
  "_self": "db/5ed5e5b5-f7ad-463c-bf55-34c028c70ea/documents/5ed5e5b5-f7ad-463c-bf55-34c028c70ea",
  "_etag": "\"0000000f-0000-0200-0000-002091140000\"",
  "_attachments": "attachments/",
  "ts": 1713738884
}
```

I. Performance:

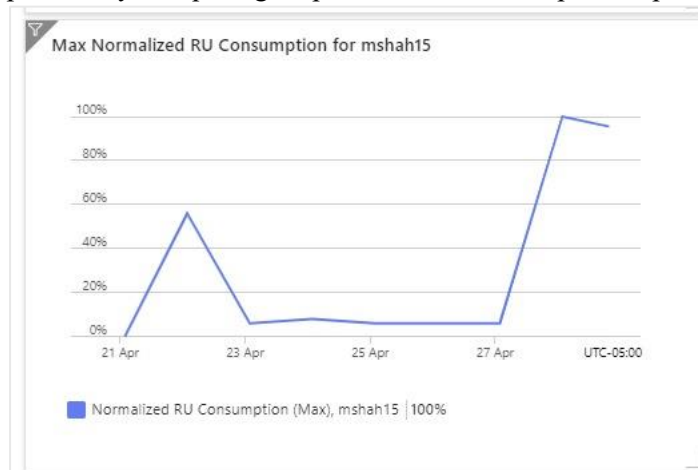
- This metric represents the average amount of data used by the user "mshah15" over a specified period.



- This metric provides a snapshot of the total data consumed by the user "mshah15" within a specific time frame.

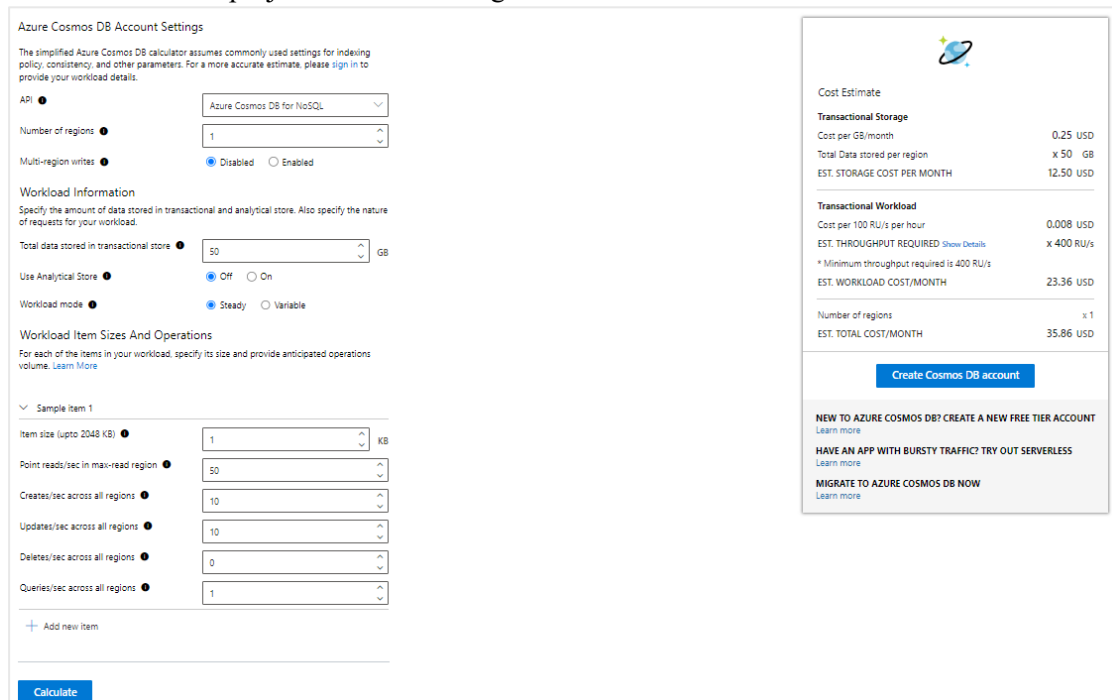


- Normalized RU consumption reflects the database throughput required to serve user requests efficiently. A high value indicates that the user's operations are resource-intensive, potentially requiring optimization to improve performance and reduce costs



J. Cost Analysis:

We used the Azure CosmosDB cost analysis tool (linked below in the resources) to determine what would be the projected cost for using this database tool.



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 Information

Specify the amount of data stored in transactional and analytical store. Also specify the nature of requests for your workload.

Total data stored in transactional store: GB

Use Analytical Store: ☒ Off ☐ On

Workload mode: ☒ Steady ☐ Variable

Workload Item Sizes And Operations

For each of the items in your workload, specify its size and provide anticipated operations volume. [Learn More](#)

Sample Item 1

Item size (upto 2048 KB): KB

Point reads/sec in max-read region:

Creates/sec across all regions:

Updates/sec across all regions:

Deletes/sec across all regions:

Queries/sec across all regions:

+ Add new item

Calculate

Cost Estimate

Transactional Storage

Cost per GB/month	0.25 USD
Total Data stored per region	x 50 GB
EST. STORAGE COST PER MONTH	12.50 USD

Transactional Workload

Cost per 100 RU/s per hour	0.008 USD
EST. THROUGHPUT REQUIRED	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: 35.86 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](#)

For a single region deployment and an estimated data size of 50 GB, our projected monthly cost for utilizing Azure Cosmos DB amounts to \$35.86 USD. This estimate encompasses both storage and transactional workload expenses. In terms of storage, the projected cost is based on storing the 50 GB of data at a rate of \$0.25 USD per GB/month, resulting in a total storage cost of \$12.50 USD per month. Regarding the transactional workload, which includes provisioning throughput for executing database operations, the estimated cost is calculated considering a minimum required throughput of 400 Request Units per second (RU/s) at a rate of \$0.008 USD per 100 RU/s per hour. This translates to an estimated workload cost of \$23.36 USD per month. Notably, the workload cost accounts for various database operations such as queries, point reads, creates, and updates. Thus, for a deployment in a single region and a data size of 50 GB, the total estimated monthly cost for Azure Cosmos DB is \$35.86 USD, covering both storage and transactional workload expenditures.

It's important to note that the projected cost of utilizing Azure Cosmos DB can vary depending on several factors, including the number of regions deployed, the size of the data stored, and the throughput request units provisioned. One needs to carefully assess and adjust these parameters based on specific project requirements to optimize cost-effectiveness and resource utilization.

K. Conclusion:

While our project successfully showcased the integration of YouTube API data with Azure Cosmos DB, we encountered significant hurdles due to the platform's limitations and the relative novelty of Cosmos DB in comparison to more commonly used databases. Issues with troubleshooting and accessing appropriate documentation increased our challenges, particularly in optimizing performance, role assignment and executing complex queries (using subqueries) efficiently. CosmosDB also had a lot of limitations where the Data Explorer did not allow us to use subqueries as the platform did not support it. We also faced significant challenges while connecting the backend component to CosmosDB since we took some time to figure out that

the CORS policy had to be enabled. Despite these obstacles, our project underscores the potential of Cosmos DB for diverse data storage and querying needs, emphasizing the importance of continued improvement and support to realize its full capabilities in real-world applications.

L. Resources:

- <https://learn.microsoft.com/en-us/cli/azure/cosmosdb/sql/role/assignment?view=azure-cli-latest#az-cosmosdb-sql-role-assignment-exists>
- <https://learn.microsoft.com/en-us/azure/cosmos-db/choose-api>
- <https://learn.microsoft.com/en-us/rest/api/storageservices/cross-origin-resource-sharing--cors--support-for-the-azure-storage-services>
- <https://learn.microsoft.com/en-us/samples/azure-samples/react-cosmosdb/react-cosmosdb/>
- <https://stackoverflow.com/questions/61238680/access-to-fetch-at-from-origin-http-localhost3000-has-been-blocked-by-cors>
- <https://learn.microsoft.com/en-us/azure/cosmos-db/continuous-backup-restore-introduction>
- <https://cosmos.azure.com/capacitycalculator/>