Lab Manual- Create, query, and traverse an Azure Cosmos DB graph database using the Gremlin console

Prepared for:

Date: 18th Nov 2021

Prepared by:

Document Name: Lab Manual

Document Number AZLab309

Contributor:

Contents

1. Introduction

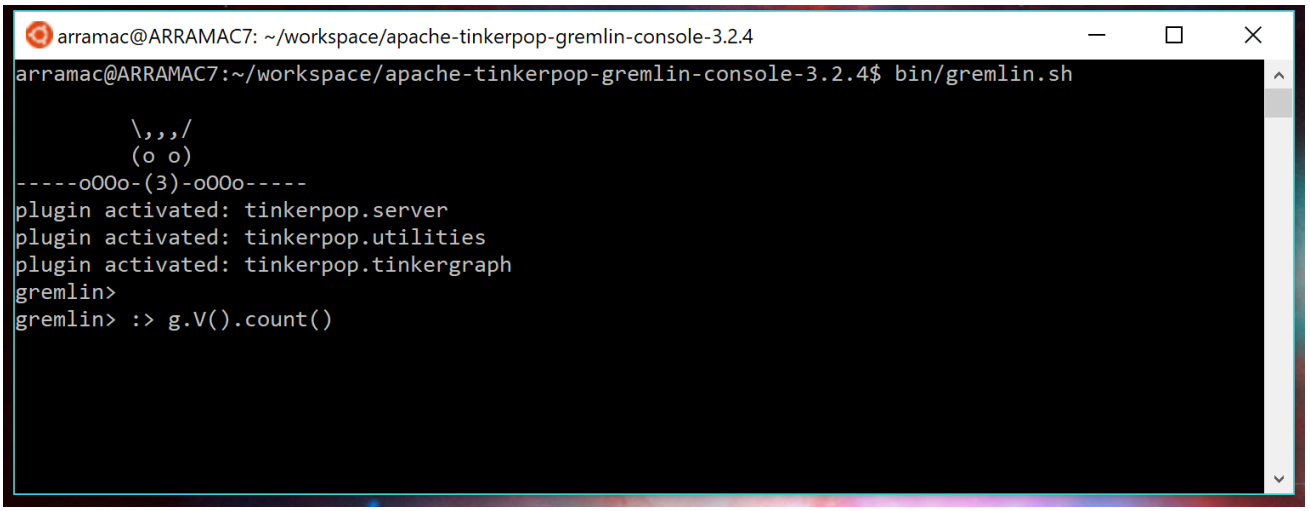
Azure Cosmos DB is a Microsoft Azure database service, fastest-growing Azure service that is available in all Azure regions, that fits any web application, mobile application, gaming or IoT application that requires processing, reading and writing a massive amount of data.

This Lab will help you to create an Azure Cosmos DB **Gremlin AP**I account**t, database, and graph (container) using the Azure portal and then use the Gremlin Console from Apache TinkerPop** to work with Gremlin API .

2. Gremlin Console

The Gremlin console is Groovy/Java based and runs on Linux, Mac, and Windows. You can download it from the

https://tinkerpop.apache.org/download.html

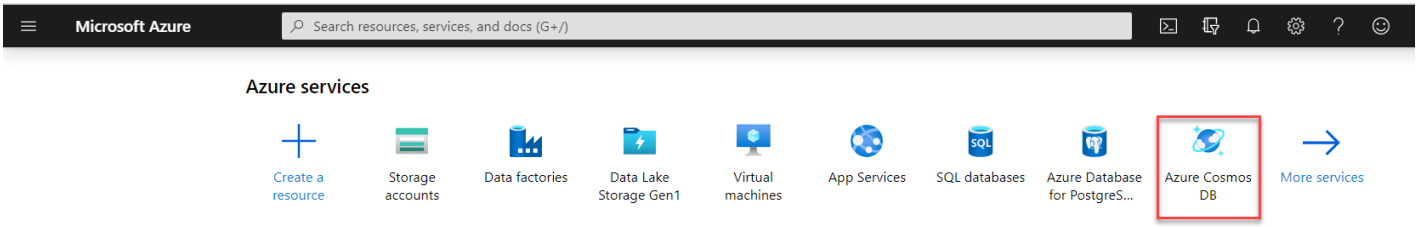


You can also download it from CosmoDb Console

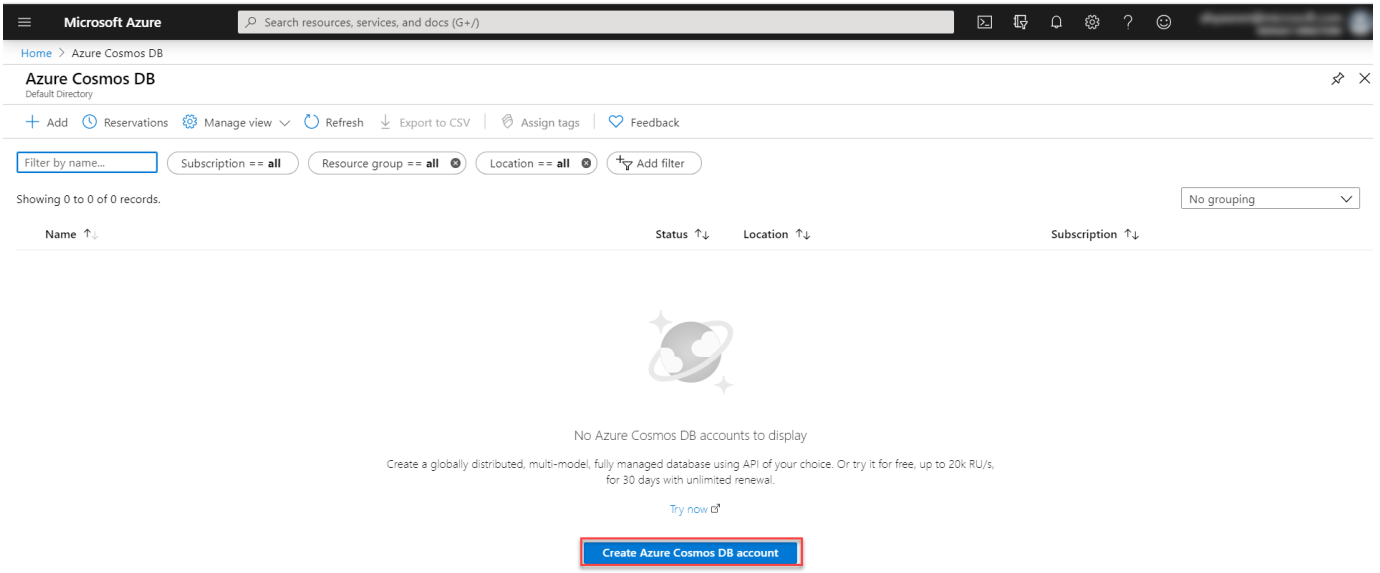
3. Create an Azure Cosmos DB Account with Graph API

1. In a new browser window, sign in to the Azure portal.

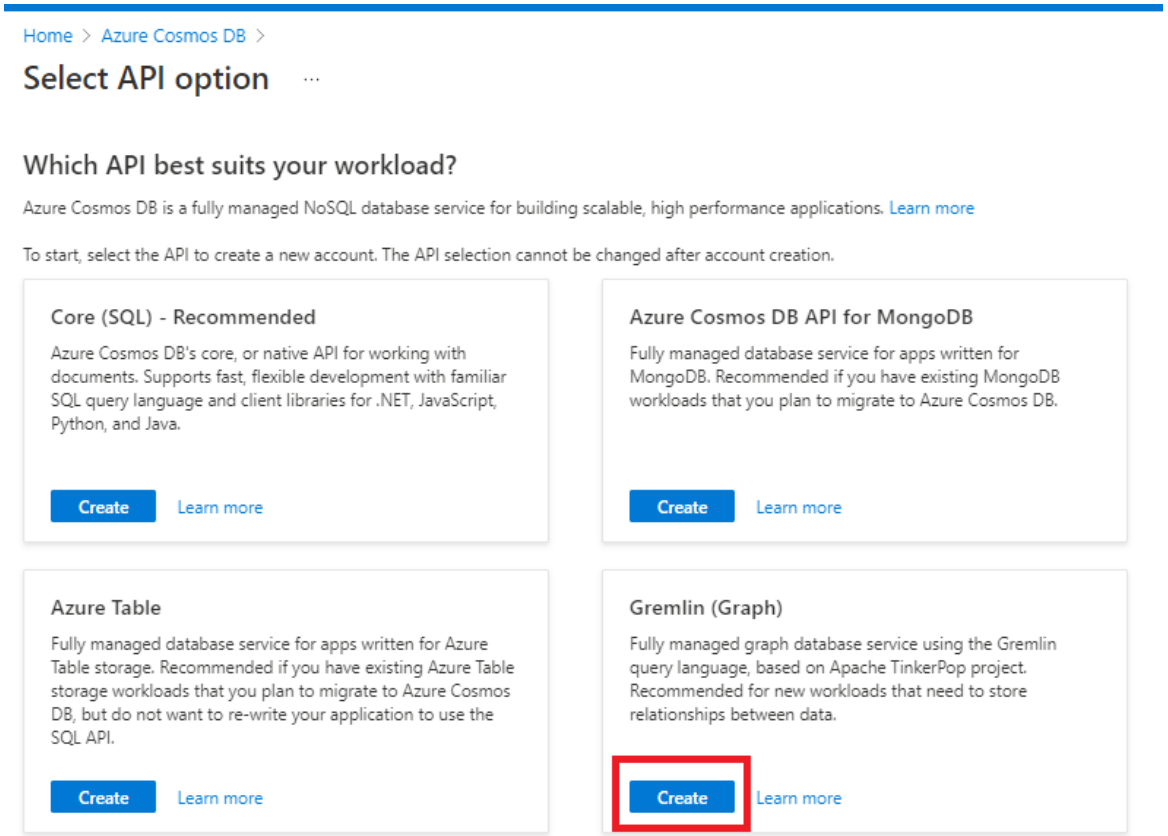
2. Search or select Azure ComsoDB



3. In the opened page, click on the **Create Azure Cosmos DB account** option, to create a new Cosmos DB account, as shown below:



4. Select API Gremlin and click **Create** Button



5. From the Create Azure Cosmos DB Account page, select the **Azure subscription** under which you plan to create the Cosmos DB account and the **resource group**.
6. Select your region and Leave all option default

Home > Azure Cosmos DB > Select API option >

Create Azure Cosmos DB Account - Gremlin (Graph) ...

Basics | Global Distribution | Networking | Backup Policy | Encryption | Tags | Review + create

Azure Cosmos DB is a fully managed NoSQL database service for building scalable, high performance applications. Try it for free, for 30 days with unlim

Project Details

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

Subscription * Azure Pass - Sponsorship

Resource Group * bipdemorg

Create new

Instance Details

Account Name * graphapidemo

Location * (US) East US

Capacity mode ⓘ ◉ Provisioned throughput ○ Serverless

Learn more about capacity mode

With Azure Cosmos DB free tier, you will get the first 1000 RU/s and 25 GB of storage for free in an account. You can enable free tier on up to one acco

Apply Free Tier Discount ◉ Apply ○ Do Not Apply

Limit total account throughput ☑ Limit the total amount of throughput that can be provisioned on this account

ⓘ This limit will prevent unexpected charges related to provisioned throughput. You can update or remove this lin

Review + create | Previous | Next: Global Distribution

7. After providing your selections click on **Review + Create** .

Home > Azure Cosmos DB > Select API option >

Create Azure Cosmos DB Account - Gremlin (Graph)

Validation Success

Basics | Global Distribution | Networking | Backup Policy | Encryption | Tags | Review + create

Creation Time

Estimated Account Creation Time (in minutes)	2

The estimated creation time is calculated based on the location you have selected

Basics

Subscription	Azure Pass - Sponsorship
Resource Group	bipdemorg
Location	West US
Account Name	(new) gramlinapidemo
API	Gremlin (graph)
Capacity mode	Provisioned throughput
Geo-Redundancy	Disable
Multi-region Writes	Disable

Backup Policy

Backup policy	Periodic
Backup storage redundancy	Locally-redundant backup storage

Networking

Connectivity method	All networks

Create | Previous | Next | Download a template for automation

8. In the Review + Create page, review all the selections then click on the **Create** option to create your Azure Cosmos DB account

Home >

Microsoft.Azure.CosmosDB-20220908125411 | Overview

Deployment

Search (Ctrl+/)

- Overview
- Inputs
- Outputs
- Template

Delete | Cancel | Redeploy | Download | Refresh

We'd love your feedback! →

Deployment is in progress

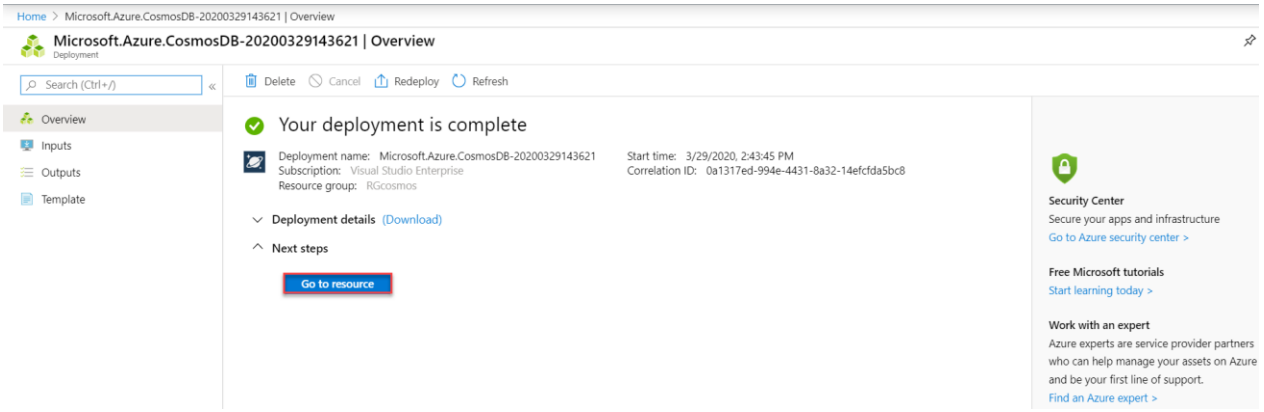
Deployment name: Microsoft.Azure.CosmosDB-20220908125411
Subscription: Azure Pass - Sponsorship
Resource group: bipdemorg

Start time: 9/8/2022, 12:54:19 PM
Correlation ID: 78d90eff-4648-423a-a77c-67758888fb69

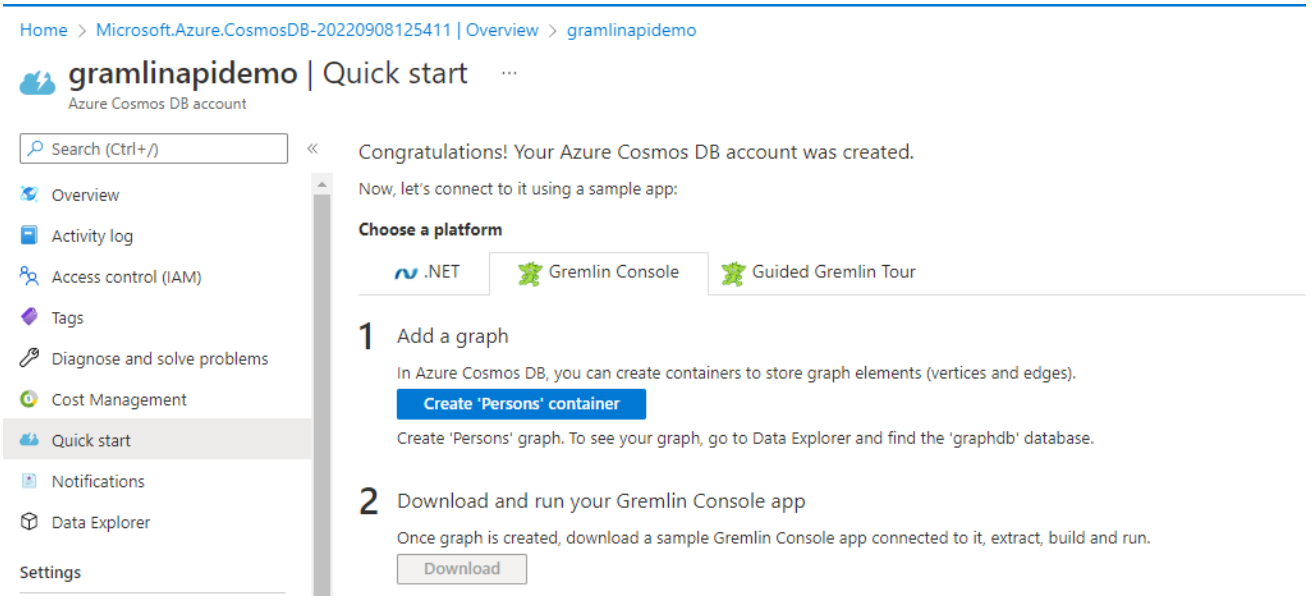
Deployment details

Resource	Type	Status
gramlinapidemo	Microsoft.DocumentDb/databaseAccounts	OK

9. After a few minutes, the database account will be created and deployed completely. Click on the **Go to resource** option to start working in the database account, as shown below:



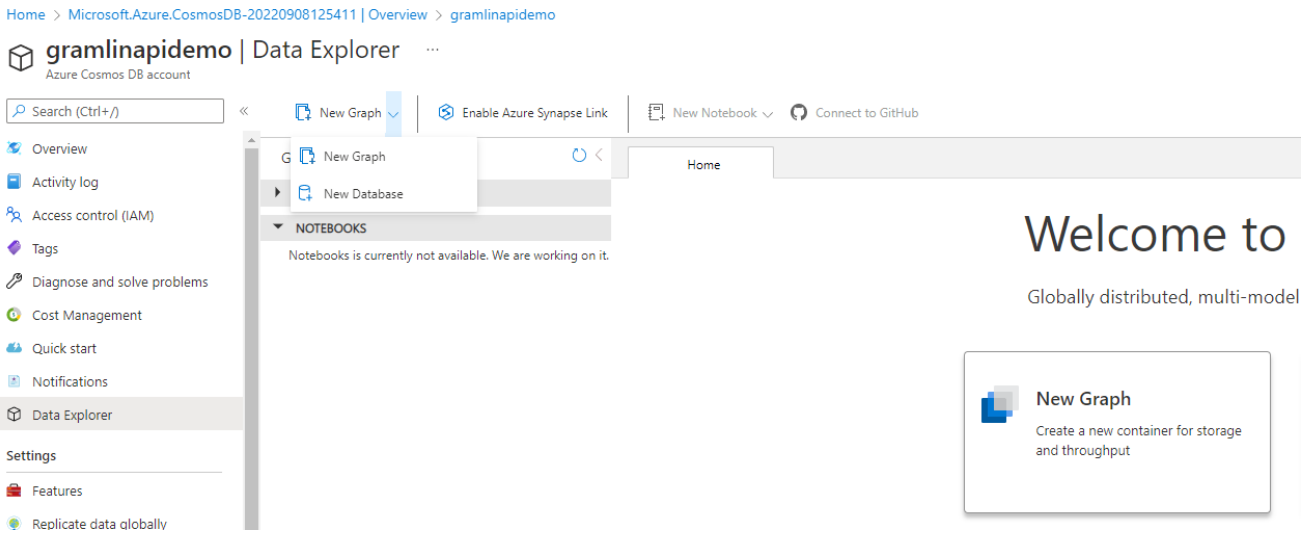
10. The first opened page is the Quickstart page that allows you to select from the available platforms and the next step to proceed with the Azure Cosmos DB account. Click on the Overview page from the below:



4. Add a Graph

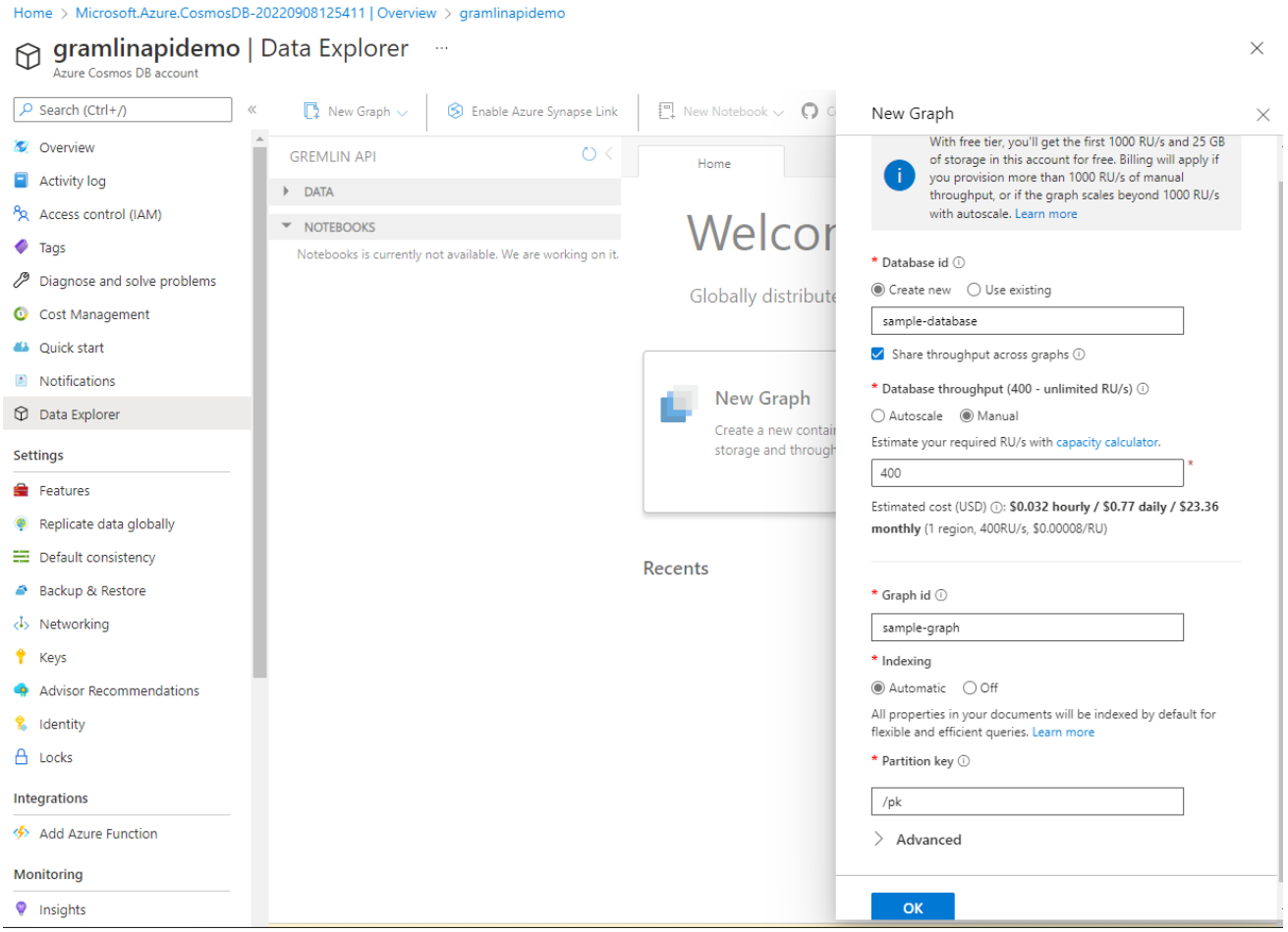
You can now use the Data Explorer tool in the Azure portal to create a graph database.

1. Select **Data Explorer** > **New Graph**.

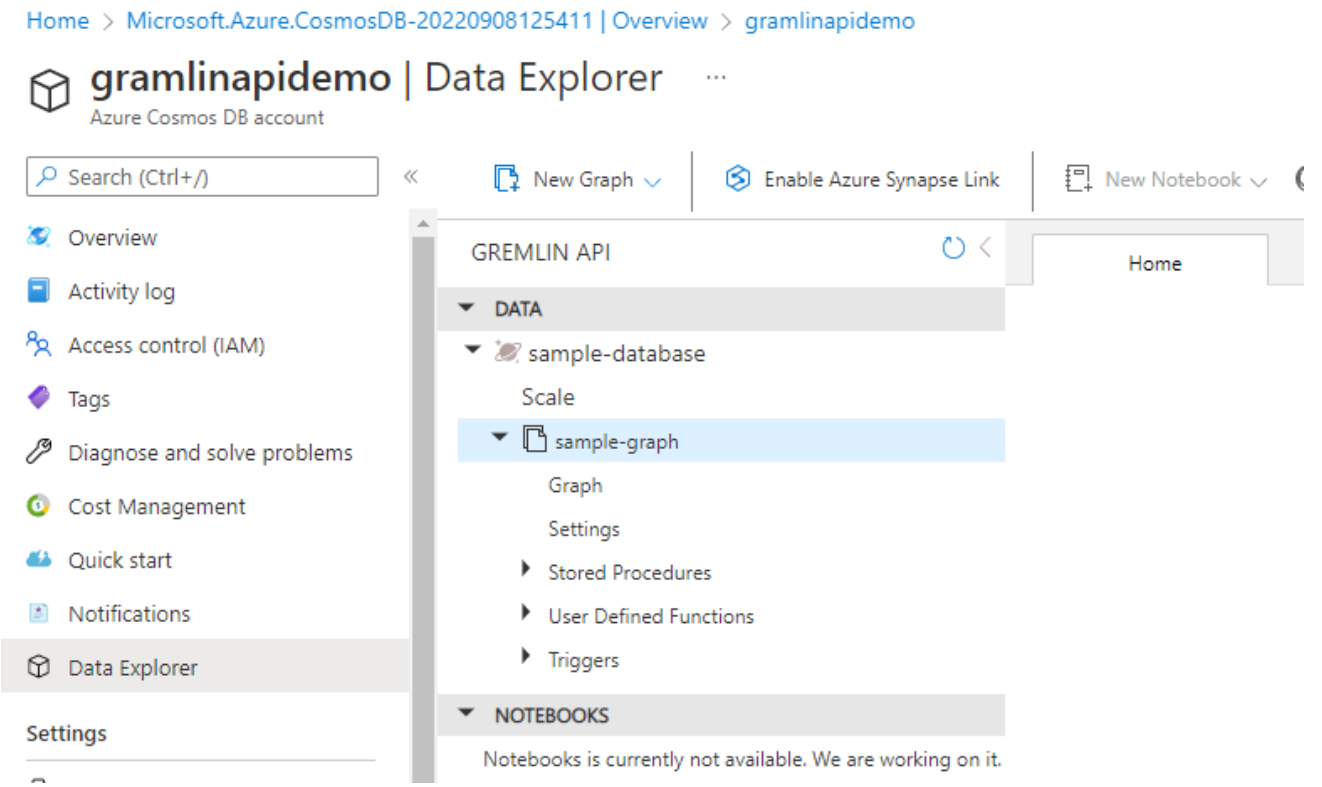


2. In the **Add graph** page, enter the settings for the new graph.

Setting	Suggested value	Description
Database ID	sample-database	Enter *sample-database* as the name for the new database.
Throughput	400 RUs	Change the throughput to 400 request units per second (RU/s).
Graph ID	sample-graph	Enter *sample-graph* as the name for your new collection.
Partition Key	/pk	All Cosmos DB accounts need a partition key to horizontally scale..



3. Once the form is filled out, select **OK** and you will get screen similar to below.



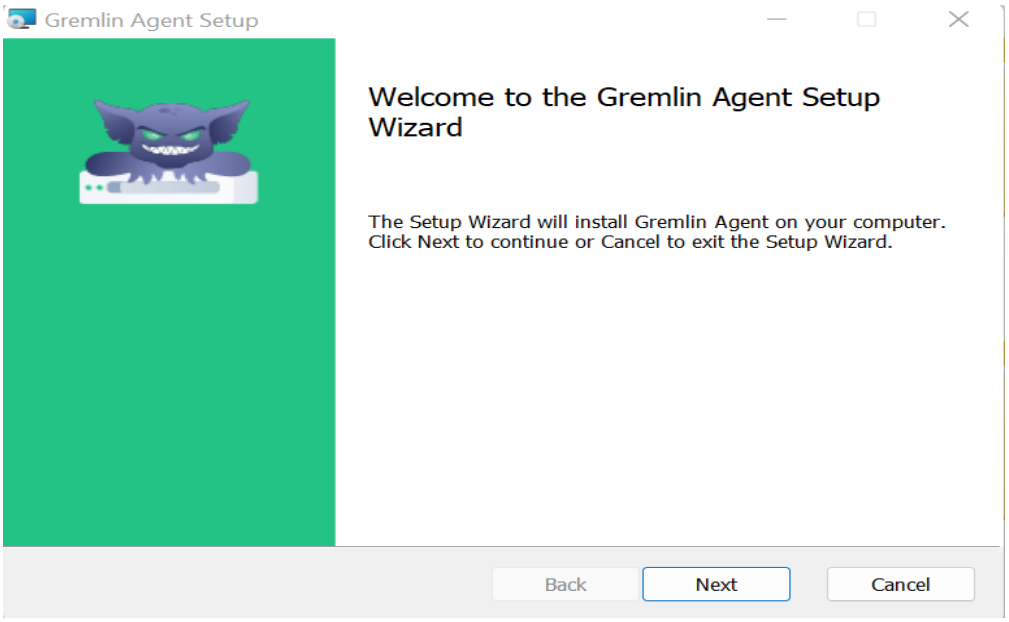
5. Download and Install Gremlin Agent

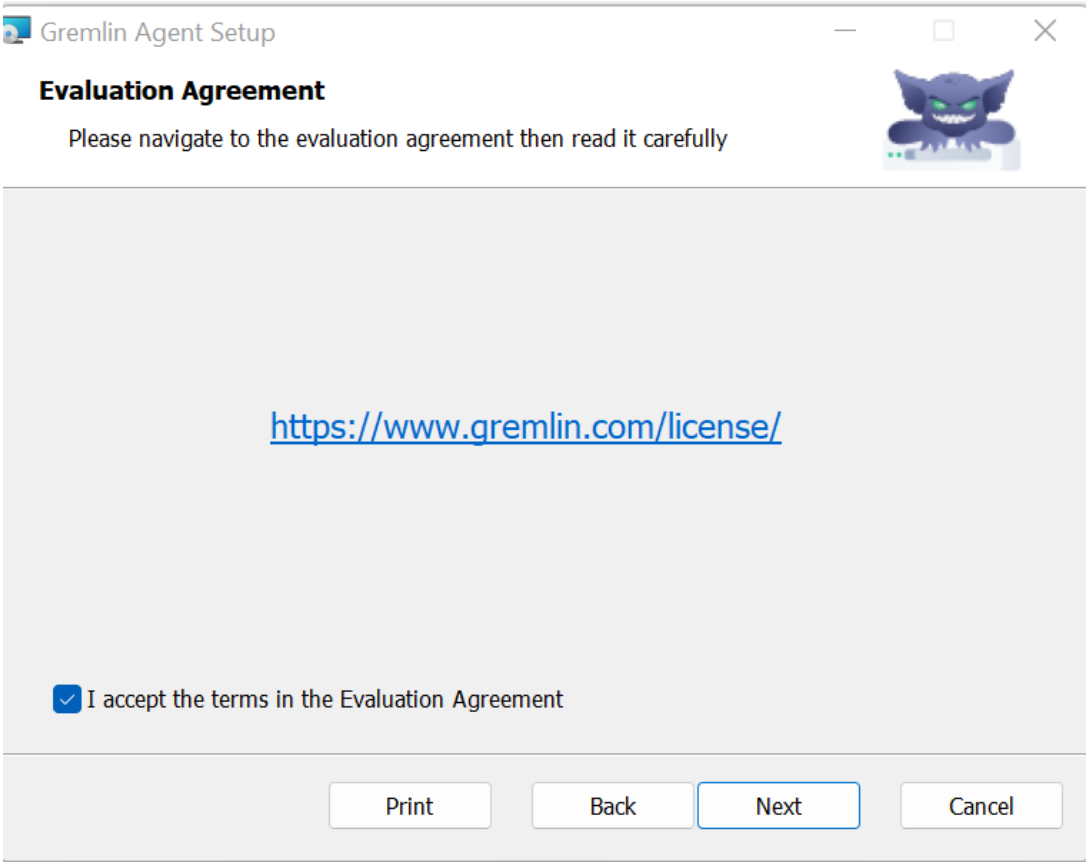
5.1 Step1:

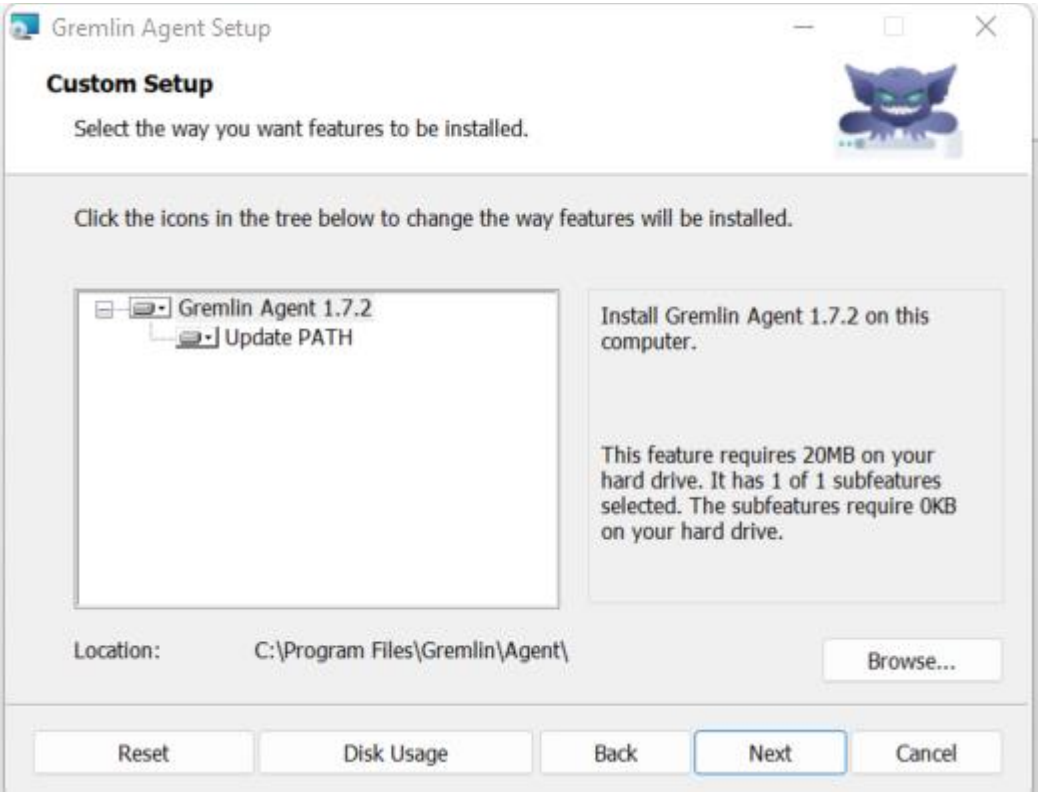
1. Download the Gremlin Installer from this website

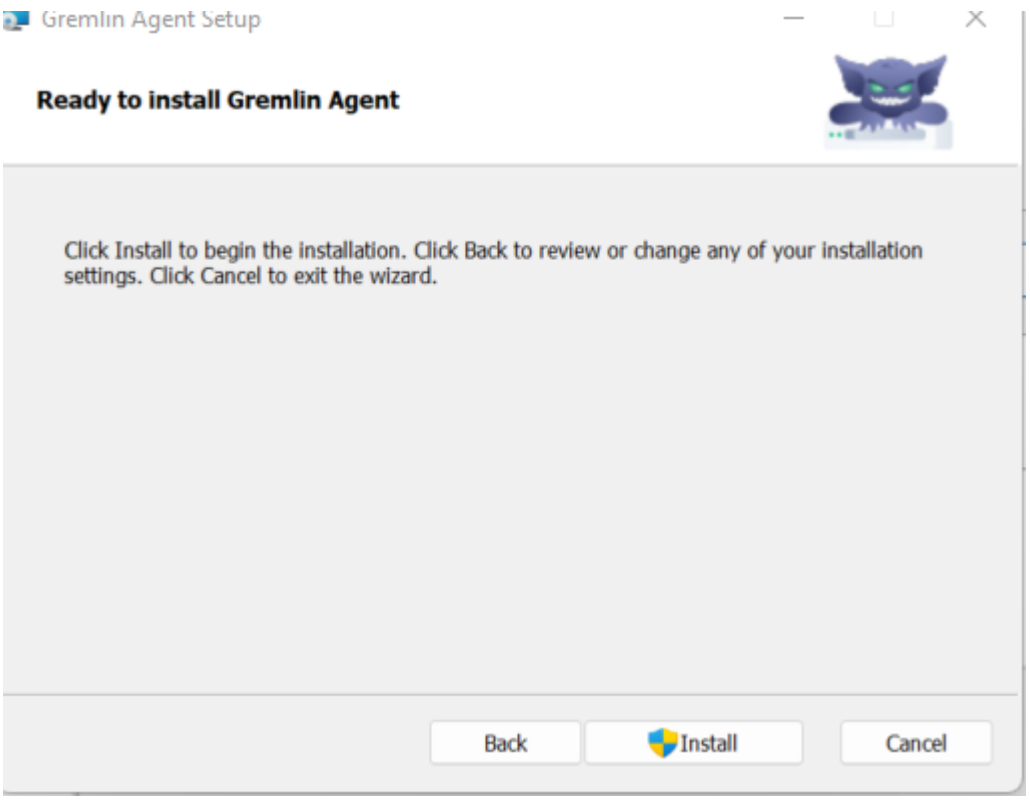
https://www.gremlin.com/community/tutorials/how-to-install-and-run-gremlin-on-windows/

2. Click the MSI to begin the installation and follow the wizard

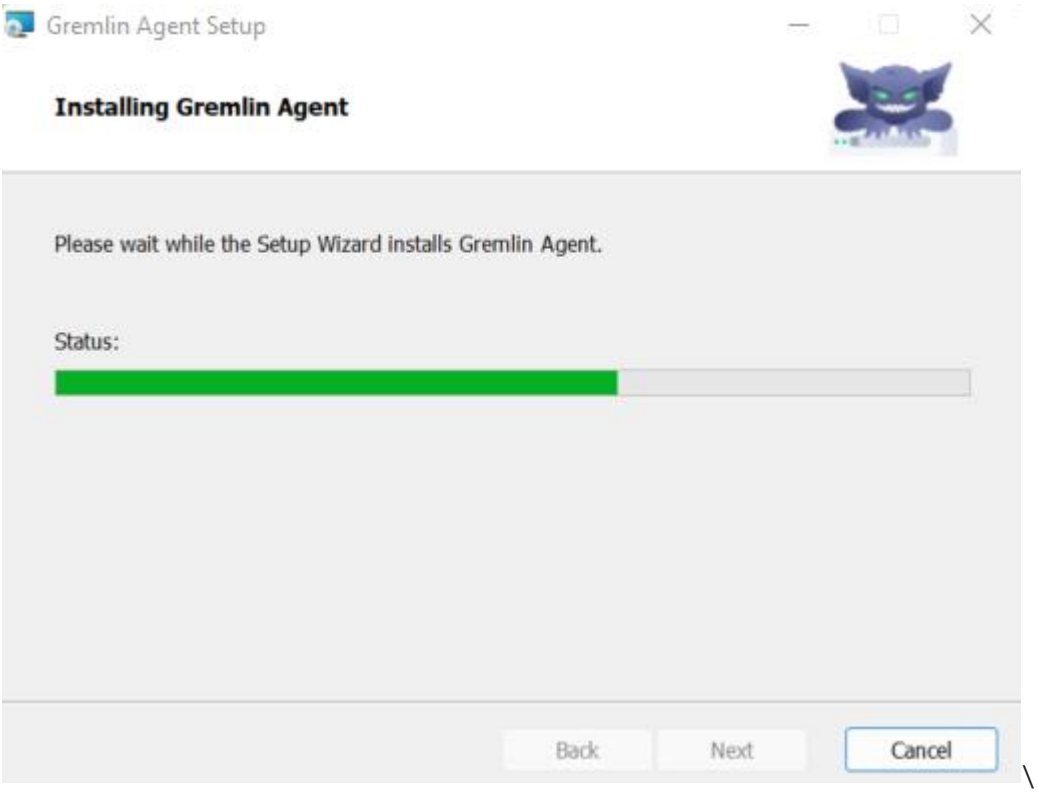


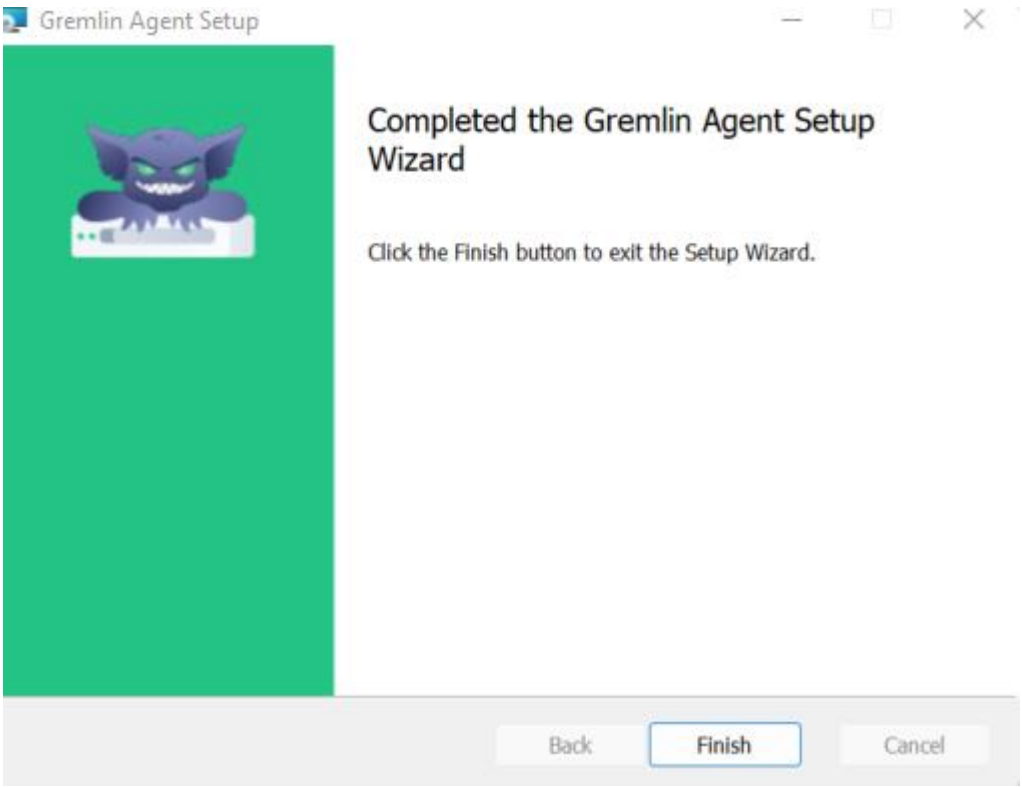

Gremlin Agent Setup
Evaluation Agreement
Please navigate to the evaluation agreement then read it carefully
https://www.gremlin.com/license/
I accept the terms in the Evaluation Agreement
Print
Back
Next
Cancel


Gremlin Agent Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
Gremlin Agent 1.7.2
Update PATH
Install Gremlin Agent 1.7.2 on this computer.
This feature requires 20MB on your hard drive. It has 1 of 1 subfeatures selected. The subfeatures require 0KB on your hard drive.
Location:
C:\Program Files\Gremlin\Agent\
Browse...
Reset
Disk Usage
Back
Next
Cancel



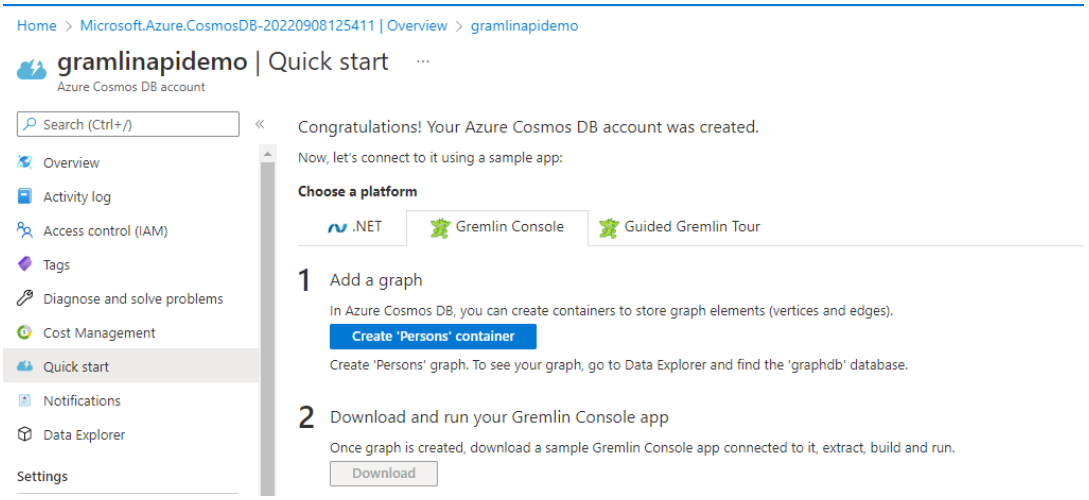
- A dialog box asking to escalate privileges may be displayed. Click *Yes*.



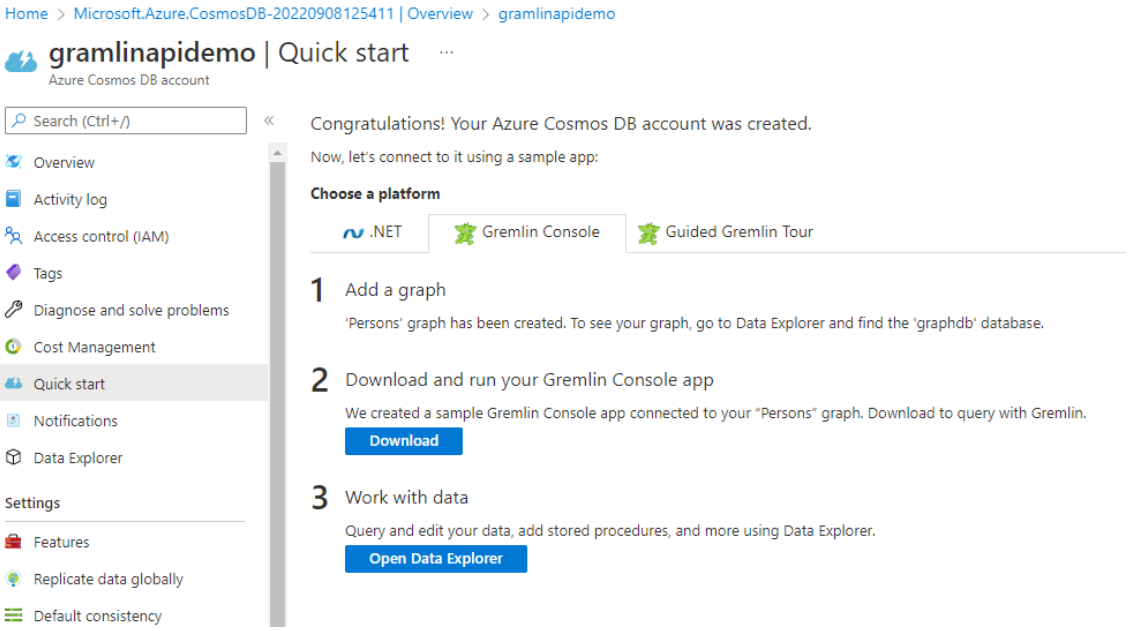


5.2 Step2:

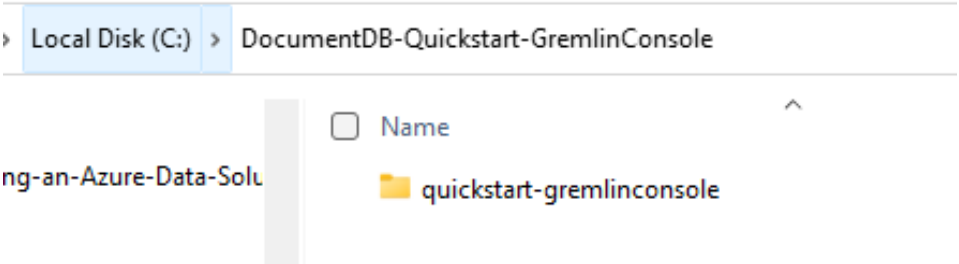
1. Create Persons Container



2. Download the Gremlin Installer from this CosmoDB Quick start

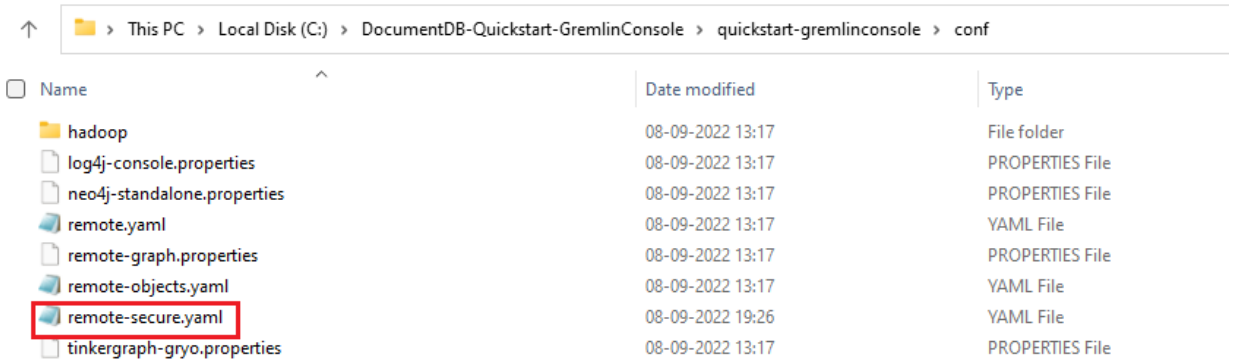


3. Once the Download Complete , extract the Zip and Copy the Folder to C:\



6. *Configure Remote-secure.yaml file*

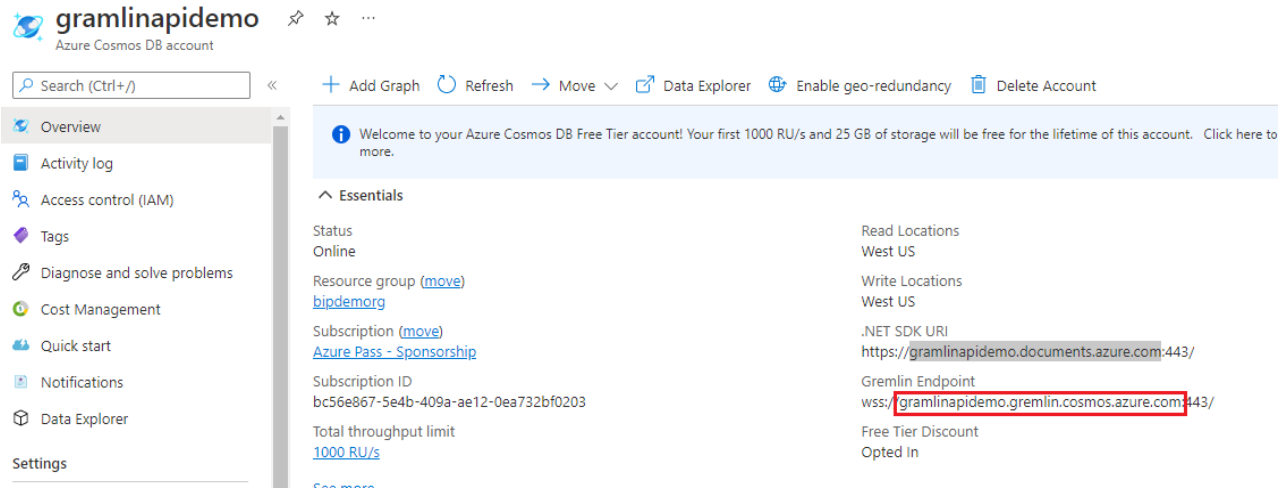
1. Before starting the Gremlin Console, create or modify the remote-secure.yaml configuration file in the apache-tinkerpop-gremlin-console-3.2.5/conf directory.



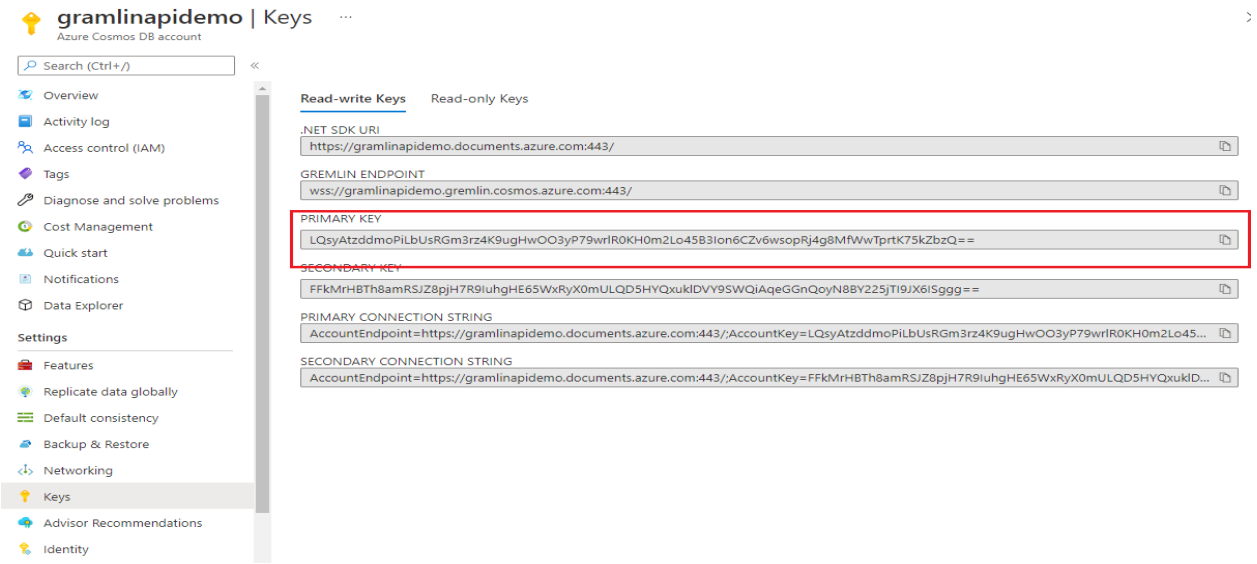
2. Fill in your *host*, *port*, *username*, *password*, *connectionPool*, and *serializer* configurations as defined in the following table:

Setting	Suggested value	Description
hosts	[*account-name*.**gremlin**.cosmos.azure.com]	See the following screenshot. This is the **Gremlin URI** value on the Overview page of the Azure portal, in square brackets, with the trailing :443/ removed. Note:
port	443	Set to 443.
username	*Your username*	The resource of the form /dbs/<db>/colls/<coll> where <db> is your database name and <coll> is your collection name.
password	*Your primary key*	See second screenshot below. This is your primary key, which you can retrieve from the Keys page of the Azure portal, in the Primary Key box. Use the copy button on the left side of the box to copy the value.
connectionPool	{enableSsl: true}	Your connection pool setting for TLS.
serializer	{ className: org.apache.tinkerpop.gremlin.driver.ser.GraphSONMessageSerializerV2d0, config: { serializeResultToString: true }}	Set to this value and delete any \n line breaks when pasting in the value.

For the hosts value, copy the **Gremlin URI** value from the **Overview** page:



For the password value, copy the **Primary key** from the **Keys** page:



Your **remote-secure.yaml** file should look like this. Save the File.

7. Connect to your CosmoDB/Graph

1. In your terminal, run bin/gremlin.bat or bin/gremlin.sh to start the Gremlin Console

```
C:\DocumentDB-Quickstart-GremlinConsole\quickstart-gremlinconsole\bin>gremlin.bat

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
```

2. In your terminal, run :remote connect tinkerpop.server conf/remote-secure.yaml to connect to your app service.

```
gremlin> :remote connect tinkerpop.server conf/remote-secure.yaml
log4j:WARN No appenders could be found for logger (io.netty.util.internal.log
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more
==>Configured gramlinapidemo.gremlin.cosmos.azure.com/40.112.241.114:443
```

3. Next run **:remote console** to redirect all console commands to the remote server.

```
gremlin> :remote console
==>All scripts will now be sent to Gremlin Server - [gramlinapidemo.gremlin.cosmos.azure.com/40.112.241.114:443]
n to local mode
```

4. Let's try a simple **count()** command. Type the following into the console at the prompt

g.V().count()

```
gremlin> g.V().count()
==>0
```

8. Create vertices and edges

1. **Let's begin by adding five person vertices for *Thomas, Mary Kay, Robin, Ben,* and *Jack.***

Input (Thomas):

g.addV('person').property('firstName', 'Thomas').property('lastName', 'Andersen').property('age', 44).property('userid', 1).property('pk', 'pk')

```
gremlin> g.addV('person').property('firstName', 'Thomas').property('lastName', 'Andersen').property('age', 44).property('userid', 1).property('pk', 'pk')
```

Input (Mary Kay):

g.addV('person').property('firstName', 'Mary Kay').property('lastName', 'Andersen').property('age', 39).property('userid', 2).property('pk', 'pk')

Input (Robin):

g.addV('person').property('firstName', 'Robin').property('lastName', 'Wakefield').property('userid', 3).property('pk', 'pk')

Input (Ben):

g.addV('person').property('firstName', 'Ben').property('lastName', 'Miller').property('userid', 4).property('pk', 'pk')

Input (Jack):

g.addV('person').property('firstName', 'Jack').property('lastName', 'Connor').property('userid', 5).property('pk', 'pk')

```
gremlin> g.addV('person').property('firstName', 'Thomas').property('lastName', 'Andersen').property('age', 44).property('userid', 1).property('pk', 'pk')
==>[id:b2ad1239-7937-4106-8262-29c12d3d2abd,label:person,type:vertex,properties:[firstName:[[id:eb4c7738-9334-4050-a644-421ceb836775,value:Thomas]],lastName:[[id:3b3834ae-6c
0e-4cf0-8c61-b0e8fd205f0a,value:Andersen]],age:[[id:41c1e627-2808-4477-82ec-8bbbe2e29144,value:44]],userid:[[id:c53c55ff-7fbc-46ce-bef4-47b39097d939,value:1]],pk:[[id:b2ad12
39-7937-4106-8262-29c12d3d2abd|pk,value:pk]]]]
gremlin> g.addV('person').property('firstName', 'Mary Kay').property('lastName', 'Andersen').property('age', 39).property('userid', 2).property('pk', 'pk')
==>[id:d9e62219-76aa-4e94-a102-7c25d0dc3443,label:person,type:vertex,properties:[firstName:[[id:1a50635b-69e6-4fc9-8a9d-e28d30edc5de,value:Mary Kay]],lastName:[[id:1198be18-
f442-4c9b-9870-ad194569a7ec,value:Andersen]],age:[[id:10c4da44-572d-43fb-9295-30e6ad49afcb,value:39]],userid:[[id:81b0dbeb-e128-48f5-8057-4e5f919cf3df,value:2]],pk:[[id:d9e6
2219-76aa-4e94-a102-7c25d0dc3443|pk,value:pk]]]]
gremlin> g.addV('person').property('firstName', 'Robin').property('lastName', 'Wakefield').property('userid', 3).property('pk', 'pk')
==>[id:b42f40ee-56cb-4566-a443-2f3994fb32a1,label:person,type:vertex,properties:[firstName:[[id:11198a82-0d30-4ab4-aaf8-53006c7a59f6,value:Robin]],lastName:[[id:cdd3d924-361
f-4a18-b278-6a4c88b808d1,value:Wakefield]],userid:[[id:e6fd21fc-90a4-4d38-8456-c6425a0d255a,value:3]],pk:[[id:b42f40ee-56cb-4566-a443-2f3994fb32a1|pk,value:pk]]]]
gremlin> g.addV('person').property('firstName', 'Ben').property('lastName', 'Miller').property('userid', 4).property('pk', 'pk')
==>[id:b5f79eaf-517e-4609-be9c-2466ed81681a,label:person,type:vertex,properties:[firstName:[[id:a9ac60ab-e921-4ad2-bbde-a90d7d296186,value:Ben]],lastName:[[id:6b3a9185-7ad3-
494a-8959-21027d26164c,value:Miller]],userid:[[id:f991a7e7-fa11-4161-a18d-6bf5325060b9,value:4]],pk:[[id:b5f79eaf-517e-4609-be9c-2466ed81681a|pk,value:pk]]]]
gremlin> g.addV('person').property('firstName', 'Jack').property('lastName', 'Connor').property('userid', 5).property('pk', 'pk')
==>[id:5f36157f-578e-4831-875c-a6bacd1ba251,label:person,type:vertex,properties:[firstName:[[id:b16727d5-83fd-4e50-a426-877a406c4cb1,value:Jack]],lastName:[[id:8f0988c7-2931
-4207-bca9-7e0ca4bc3106,value:Connor]],userid:[[id:582ba15d-08da-429d-b2be-bc9fd4047f82,value:5]],pk:[[id:5f36157f-578e-4831-875c-a6bacd1ba251|pk,value:pk]]]]
```

2. **Next, let's add edges for relationships between our people.**

Input (Thomas -> Mary Kay):

g.V().hasLabel('person').has('firstName', 'Thomas').addE('knows').to(g.V().hasLabel('person').has('firstName', 'Mary Kay'))

Input (Thomas -> Robin):

g.V().hasLabel('person').has('firstName', 'Thomas').addE('knows').to(g.V().hasLabel('person').has('firstName', 'Robin'))

Input (Robin -> Ben):

g.V().hasLabel('person').has('firstName', 'Robin').addE('knows').to(g.V().hasLabel('person').has('firstName', 'Ben'))

```
gremlin> g.V().hasLabel('person').has('firstName', 'Thomas').addE('knows').to(g.V().hasLabel('person').
==>[id:8bc40ee1-ec62-4556-bbcc-ae4485460f89,label:knows,type:edge,inVLabel:person,outVLabel:person,inV:
d3d2abd]
gremlin> g.V().hasLabel('person').has('firstName', 'Robin').addE('knows').to(g.V().hasLabel('person').h
==>[id:6102c9d0-c256-4069-8c24-554178efd59f,label:knows,type:edge,inVLabel:person,outVLabel:person,inV:
4fb32a1]
gremlin> g.V().hasLabel('person').has('firstName', 'Thomas').property('age', 45)
==>[id:b2ad1239-7937-4106-8262-29c12d3d2abd,label:person,type:vertex,properties:[age:[[id:1217e911-407d
44-421ceb836775,value:Thomas]],lastName:[[id:3b3834ae-6c0e-4cf0-8c61-b0e8fd205f0a,value:Andersen]],use
9-7937-4106-8262-29c12d3d2abd|pk,value:pk]]]]
gremlin> g.V().hasLabel('person').has('age', gt(40))
==>[id:b2ad1239-7937-4106-8262-29c12d3d2abd,label:person,type:vertex,properties:[age:[[id:1217e911-407d
44-421ceb836775,value:Thomas]],lastName:[[id:3b3834ae-6c0e-4cf0-8c61-b0e8fd205f0a,value:Andersen]],use
```

9. Update a vertex

Let's update the *Thomas* vertex with a new age of *45*.

g.V().hasLabel('person').has('firstName', 'Thomas').property('age', 45)

```
gremlin> g.V().hasLabel('person').has('firstName', 'Thomas').property('age', 45)
==>[id:b2ad1239-7937-4106-8262-29c12d3d2abd,label:person,type:vertex,properties:[
644-421ceb836775,value:Thomas]],lastName:[[id:3b3834ae-6c0e-4cf0-8c61-b0e8fd205f0
39-7937-4106-8262-29c12d3d2abd|pk,value:pk]]]]
```

10. Query your graph

First, let's try a query with a filter to return only people who are older than 40 years old.

Input (filter query):

g.V().hasLabel('person').has('age', gt(40))

```
gremlin> g.V().hasLabel('person').has('age', gt(40))
==>[id:b2ad1239-7937-4106-8262-29c12d3d2abd,label:person,type:
644-421ceb836775,value:Thomas]],lastName:[[id:3b3834ae-6c0e-4c
```

Next, let's project the first name for the people who are older than 40 years old.

Input (filter + projection query):

g.V().hasLabel('person').has('age', gt(40)).values('firstName')