



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

---

**Prepared for:**

**Date:** 18<sup>th</sup> Nov 2021

**Prepared by:**

Document Name: Lab Manual

**Document Number** AZLab309

**Contributor:**

## Contents

1. Introduction .....	3
2. Gremlin Console .....	3
3. Create an Azure Cosmos DB Account with Graph API .....	3
4. Add a Graph .....	8
5. Download and Install Gremlin Agent .....	10
5.1 Step1:.....	10
5.2 Step2:.....	13
6. Configure Remote-secure.yaml file.....	14
7. Connect to your CosmoDB/Graph .....	16
8. Create vertices and edges .....	17
1. Let's begin by adding five person vertices for <i>Thomas, Mary Kay, Robin, Ben, and Jack</i> . .....	17
2. Next, let's add edges for relationships between our people. ....	18
9. Update a vertex.....	18
10. Query your graph.....	19

## 1. Introduction

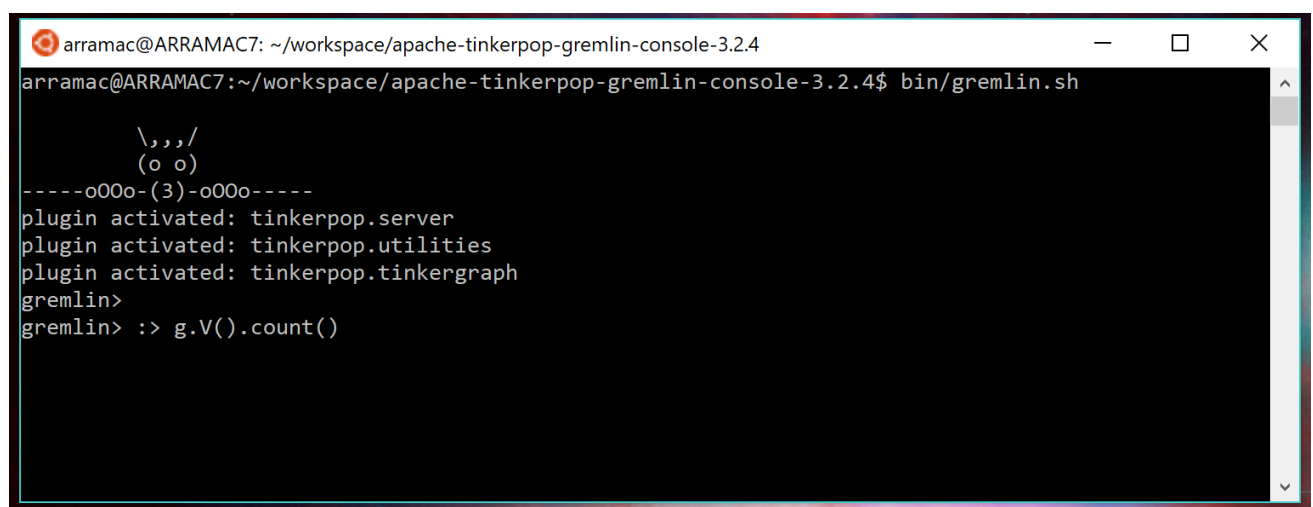
Azure Cosmos DB is a Microsoft Azure database service, fastest-growing Azure service that is available in all Azure regions, that fits any web application, mobile application, gaming or IoT application that requires processing, reading and writing a massive amount of data.

This Lab will help you to create an Azure Cosmos DB **Gremlin API** account, **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>

A screenshot of a terminal window titled 'arramac@ARRAMAC7: ~/workspace/apache-tinkerpop-gremlin-console-3.2.4'. The terminal shows the command 'bin/gremlin.sh' being executed. The output displays a ASCII art logo for Gremlin, followed by the activation of three plugins: 'tinkerpop.server', 'tinkerpop.utilities', and 'tinkerpop.tinkergraph'. The prompt 'gremlin>' is shown, and the command ':> g.V().count()' is entered, though the result is not yet visible.

```
arramac@ARRAMAC7: ~/workspace/apache-tinkerpop-gremlin-console-3.2.4
arramac@ARRAMAC7:~/workspace/apache-tinkerpop-gremlin-console-3.2.4$ bin/gremlin.sh

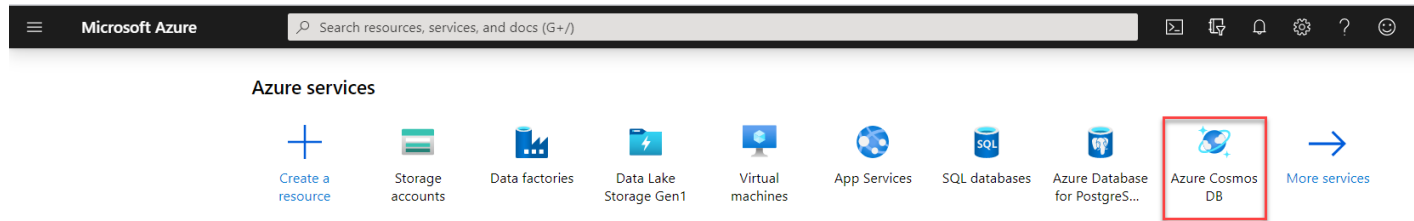
      \,,,/
      (o o)
-----o00o-(3)-o00o-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
gremlin> :> g.V().count()
```

You can also download it from CosmoDb Console

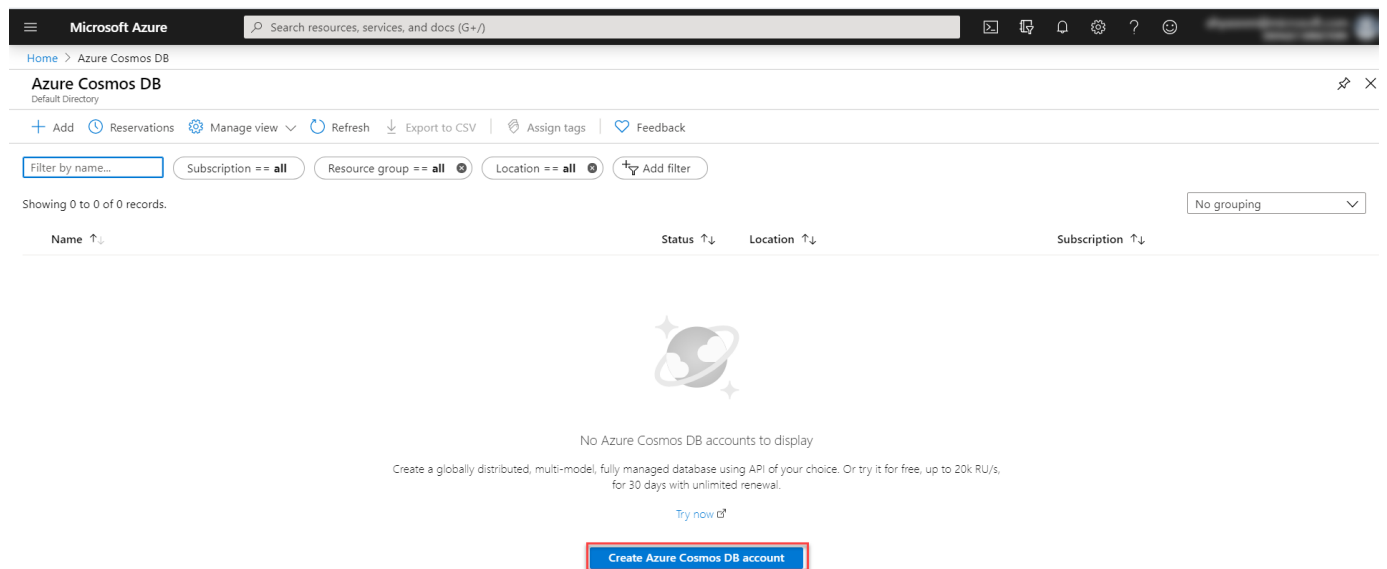
## 3. Create an Azure Cosmos DB Account with Graph API

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

## Select API option ...

### Which API best suits your workload?

Azure Cosmos DB is a fully managed NoSQL database service for building scalable, high performance applications. [Learn more](#)

To start, select the API to create a new account. The API selection cannot be changed after account creation.

#### Core (SQL) - Recommended

Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.

[Create](#)[Learn more](#)

#### Azure Cosmos DB API for MongoDB

Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.

[Create](#)[Learn more](#)

#### Azure Table

Fully managed database service for apps written for Azure Table storage. Recommended if you have existing Azure Table storage workloads that you plan to migrate to Azure Cosmos DB, but do not want to re-write your application to use the SQL API.

[Create](#)[Learn more](#)

#### Gremlin (Graph)

Fully managed graph database service using the Gremlin query language, based on Apache TinkerPop project. Recommended for new workloads that need to store relationships between data.

[Create](#)[Learn more](#)

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

## 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 *	<input type="text" value="Azure Pass - Sponsorship"/>
Resource Group *	<input type="text" value="bipdemorg"/> <a href="#">Create new</a>

### Instance Details

Account Name *	<input type="text" value="graphapidemo"/>
Location *	<input type="text" value="(US) East US"/>
Capacity mode ⓘ	<input checked="" type="radio"/> Provisioned throughput <input type="radio"/> Serverless <a href="#">Learn more about capacity mode</a>

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	<input checked="" type="radio"/> Apply <input type="radio"/> Do Not Apply
Limit total account throughput	<input checked="" type="checkbox"/> Limit the total amount of throughput that can be provisioned on this account <input checked="" type="checkbox"/> ⓘ This limit will prevent unexpected charges related to provisioned throughput. You can update or remove this lin

[Review + create](#) [Previous](#) [Next: Global Distribution](#)

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

## 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

## Microsoft.Azure.CosmosDB-20220908125411 | Overview ...

Deployment


«  Delete  Cancel  Redeploy  Download  Refresh

 Overview


 Inputs

 Outputs


 Template

 We'd love your feedback! →

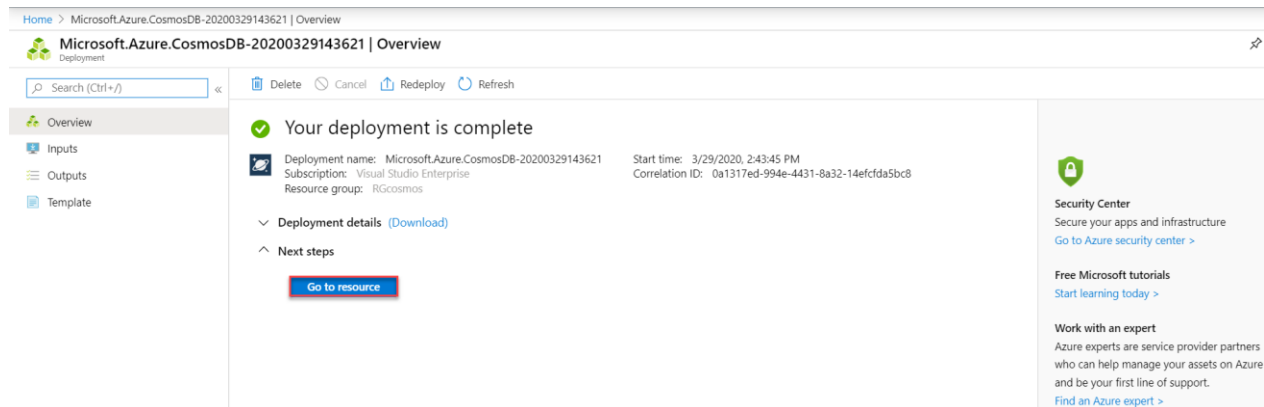
### ... Deployment is in progress

 Deployment name: Microsoft.Azure.CosmosDB-20220908125411   Start time: 9/8/2022, 12:54:19 PM  
Subscription: [Azure Pass - Sponsorship](#)   Correlation ID: 78d90eff-4648-423a-a77c-67758888fb69   
Resource group: [bipdemorg](#)

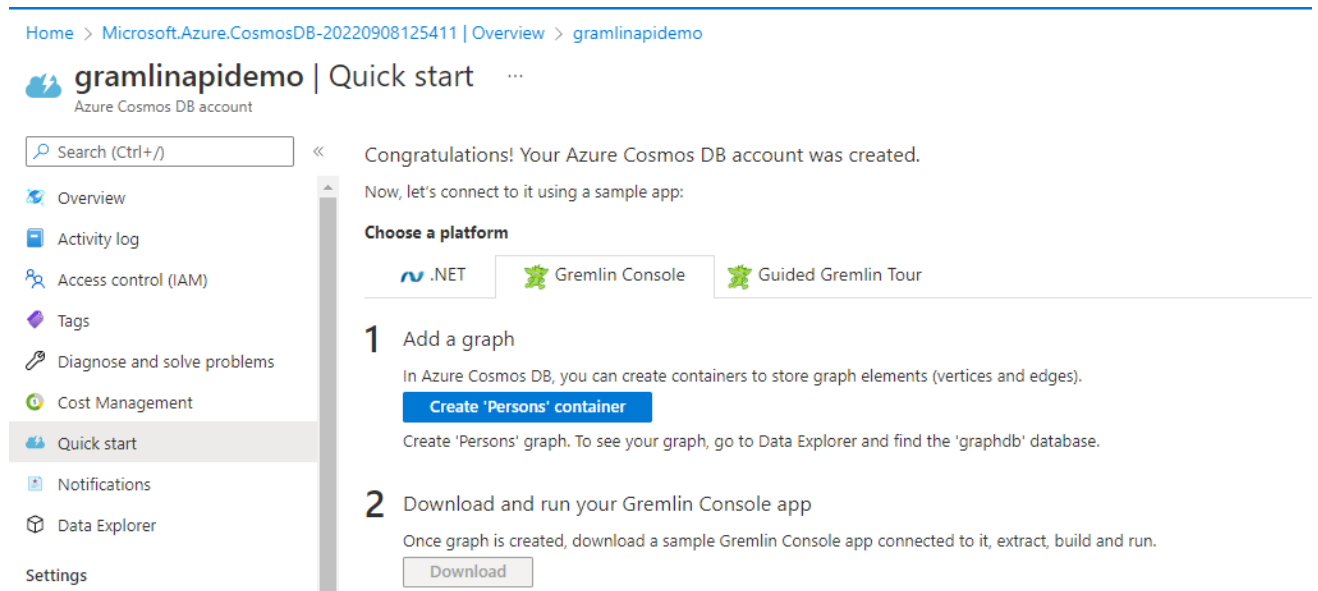
#### Deployment details

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:



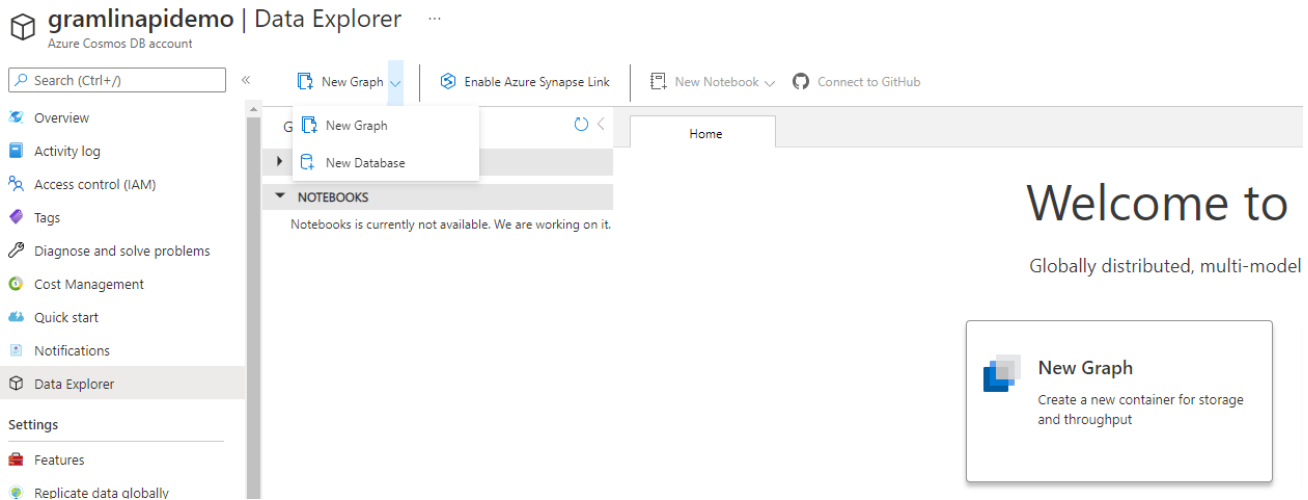
## 4. Add a Graph

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

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



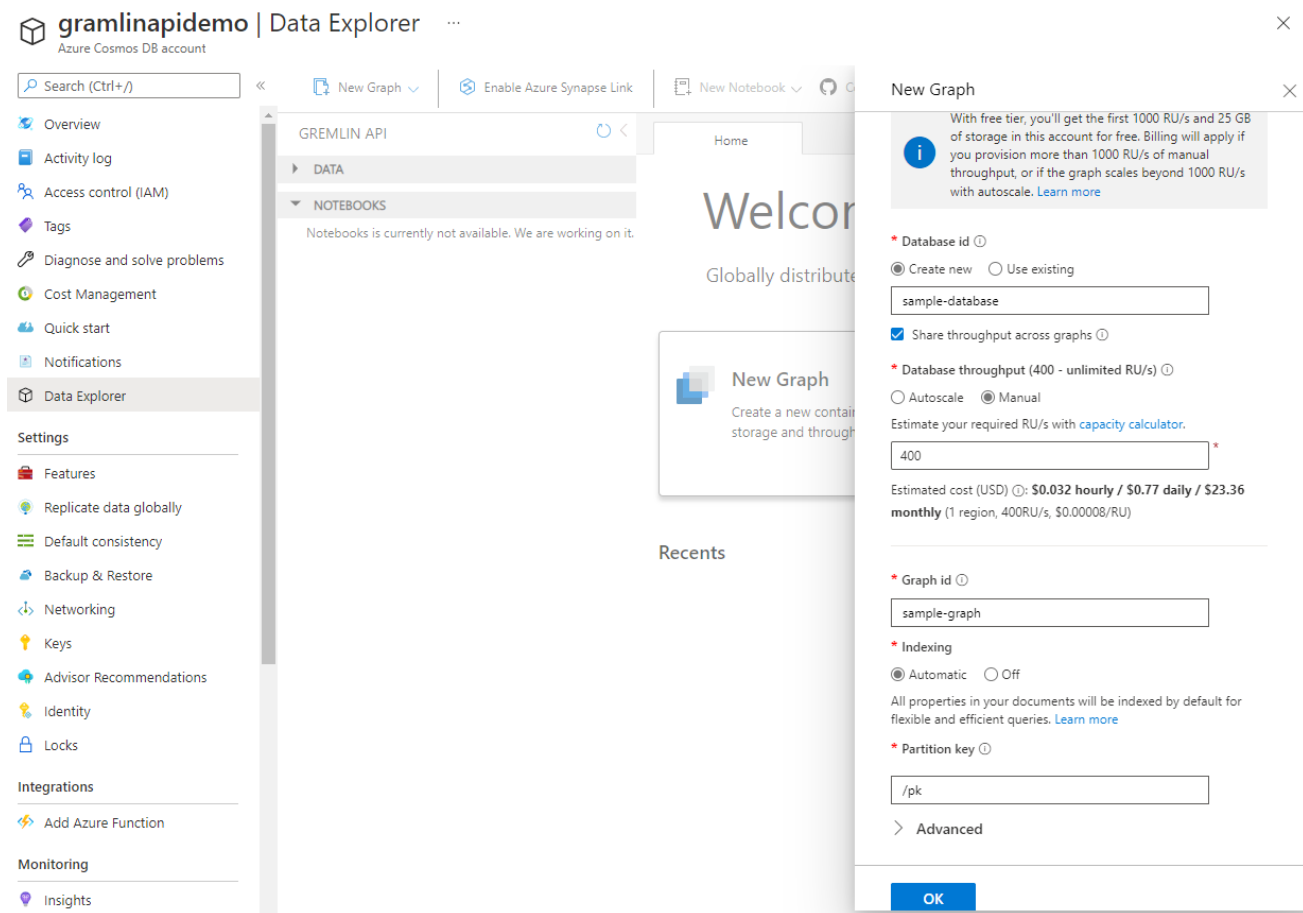
Home > Microsoft.Azure.CosmosDB-20220908125411 | Overview > gramlinapidemo



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

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

Home > Microsoft.Azure.CosmosDB-20220908125411 | Overview > gramlinapidemo



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



## gramlinapidemo | Data Explorer

Azure Cosmos DB account

Search (Ctrl+ /)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Cost Management
- Quick start
- Notifications
- Data Explorer**

Settings

New Graph

Enable Azure Synapse Link

New Notebook

GREMLIN API

### DATA

#### sample-database

Scale

#### sample-graph

Graph

Settings

Stored Procedures

User Defined Functions

Triggers

### NOTEBOOKS

Notebooks is currently not available. We are working on it.

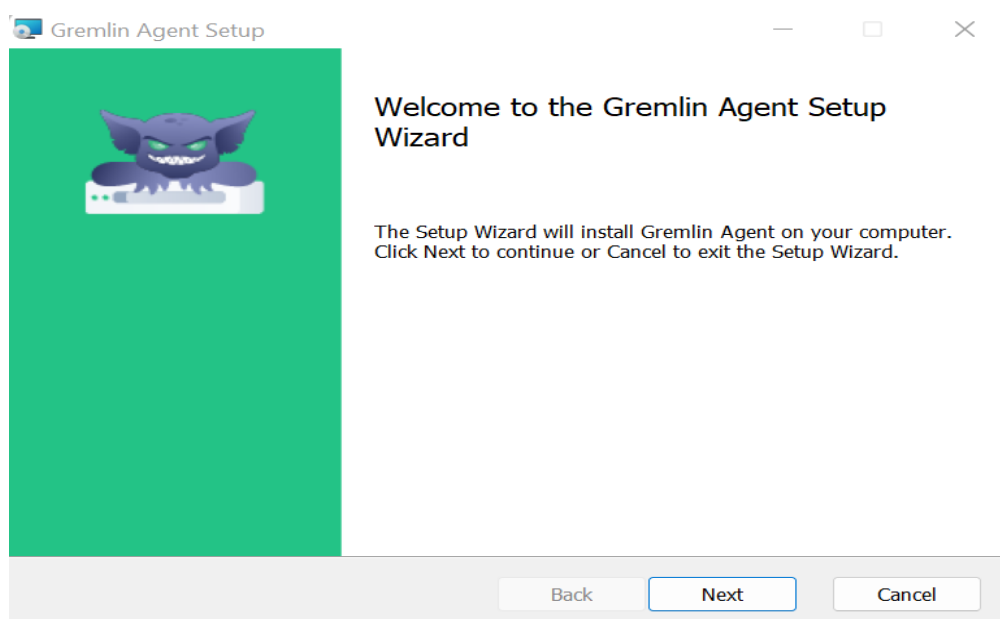
## 5. Download and Install Gremlin Agent

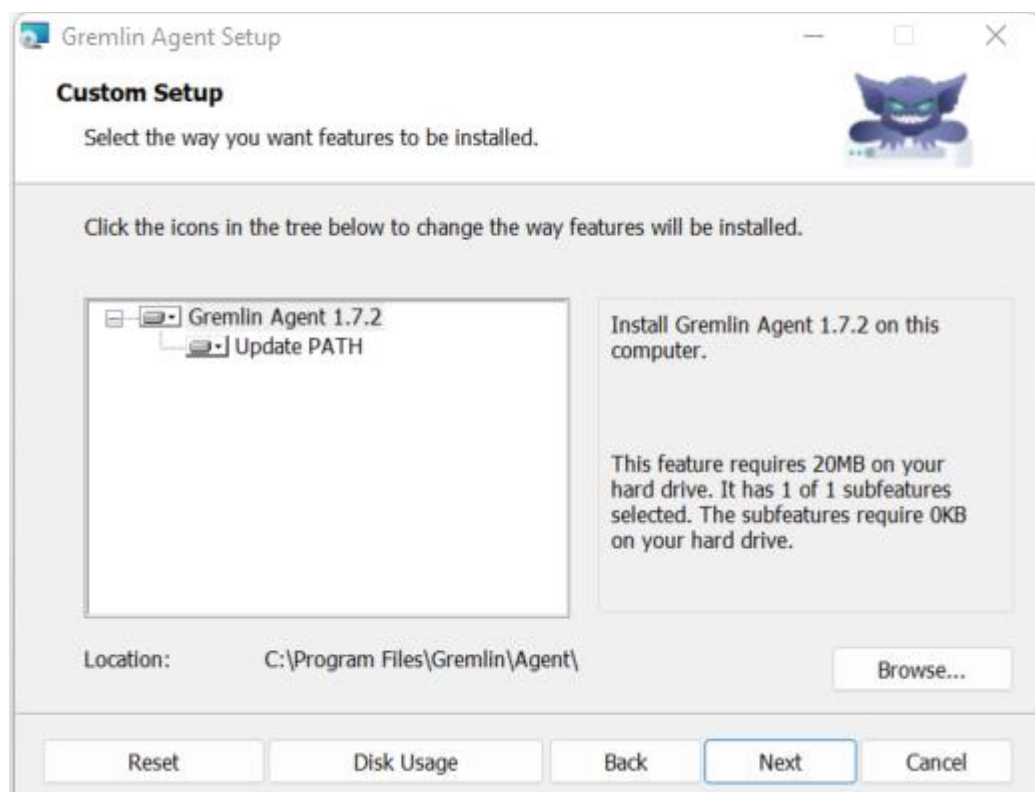
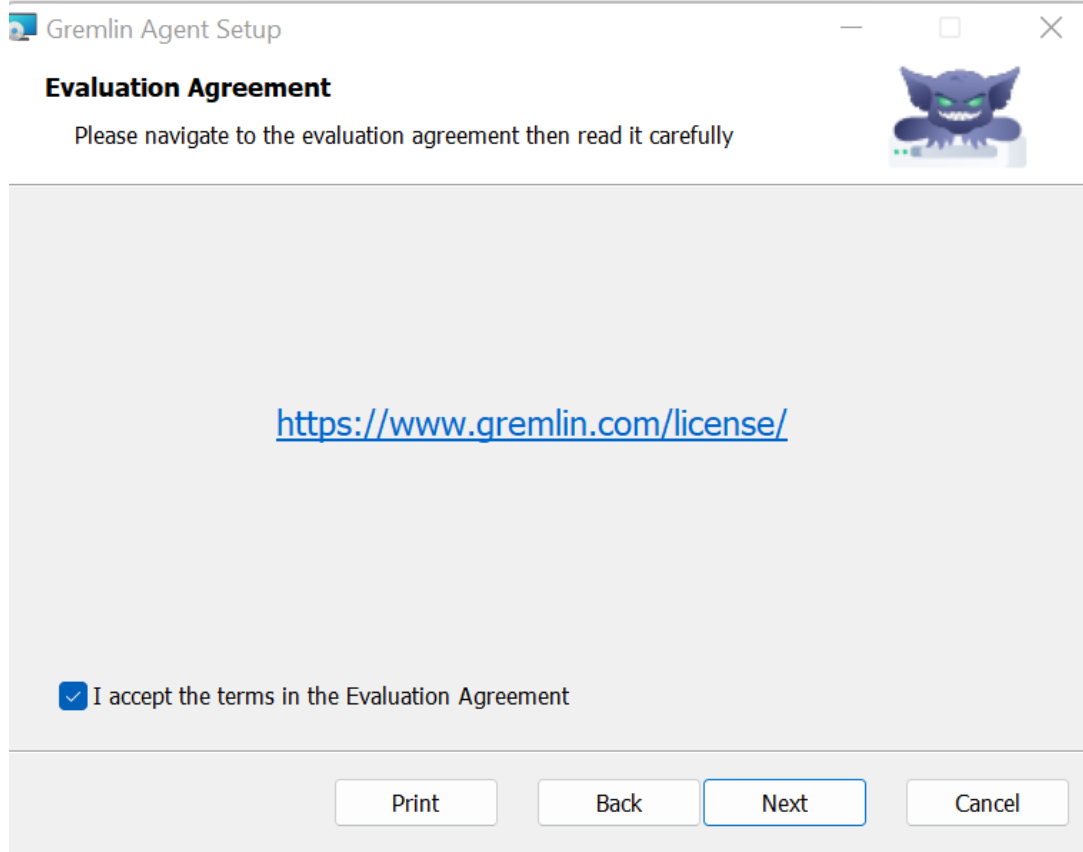
### 5.1 Step1:

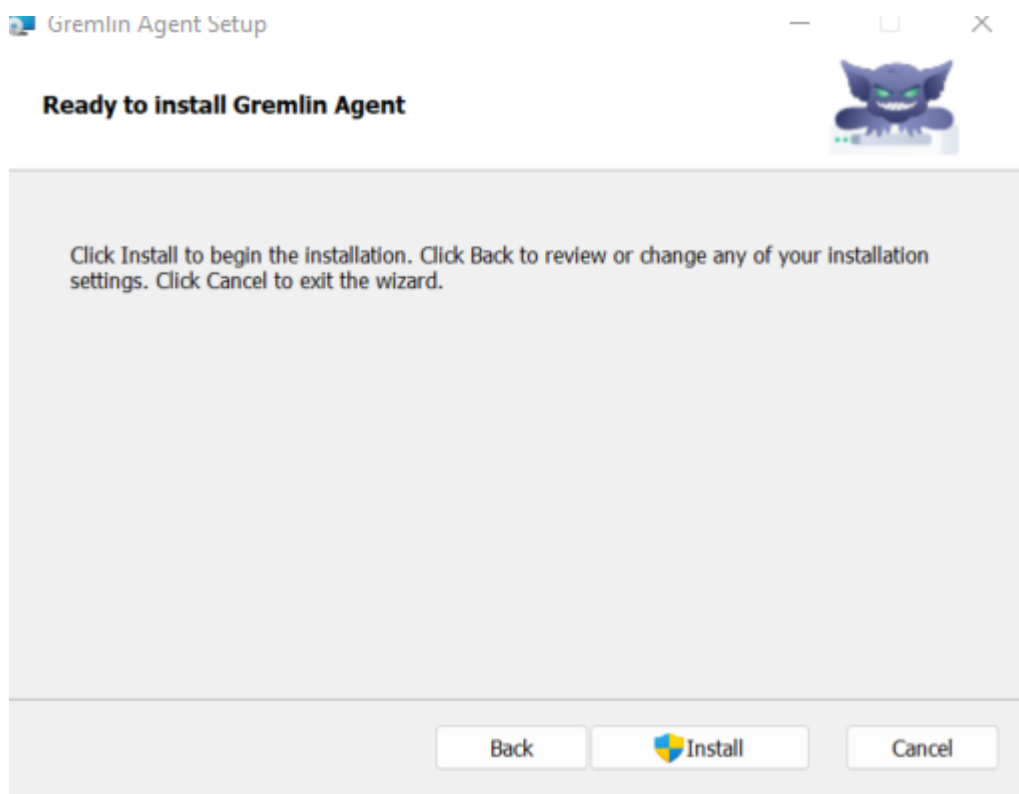
1. Download the Gremlin Installer from this website

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

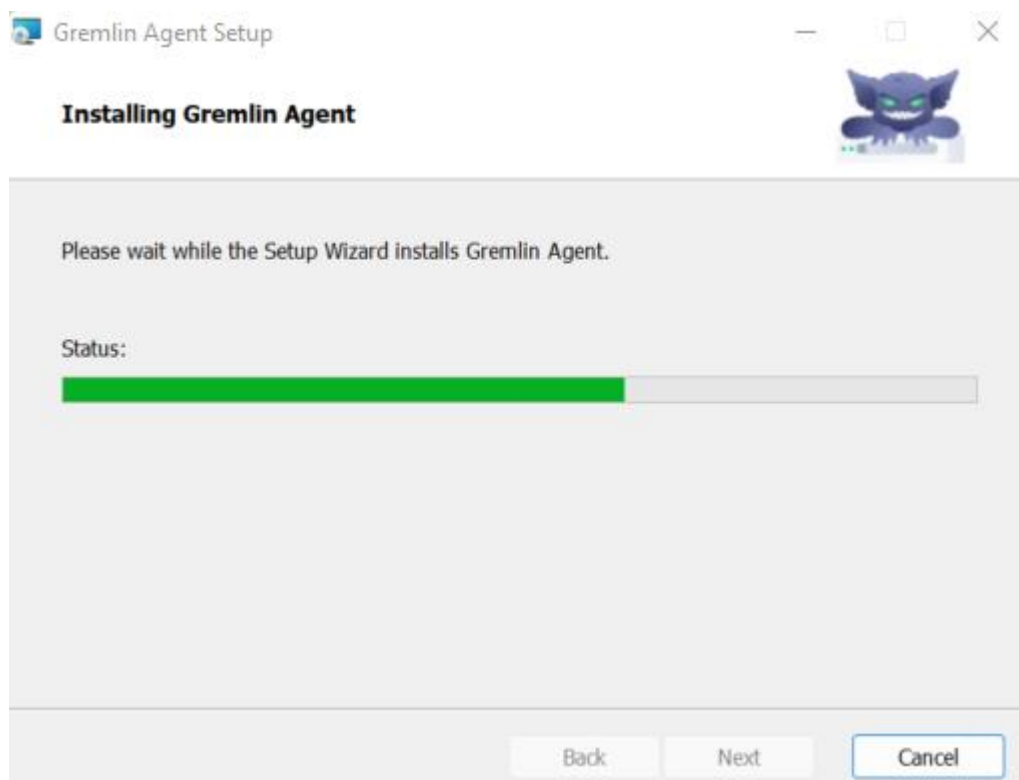
2. Click the MSI to begin the installation and follow the wizard

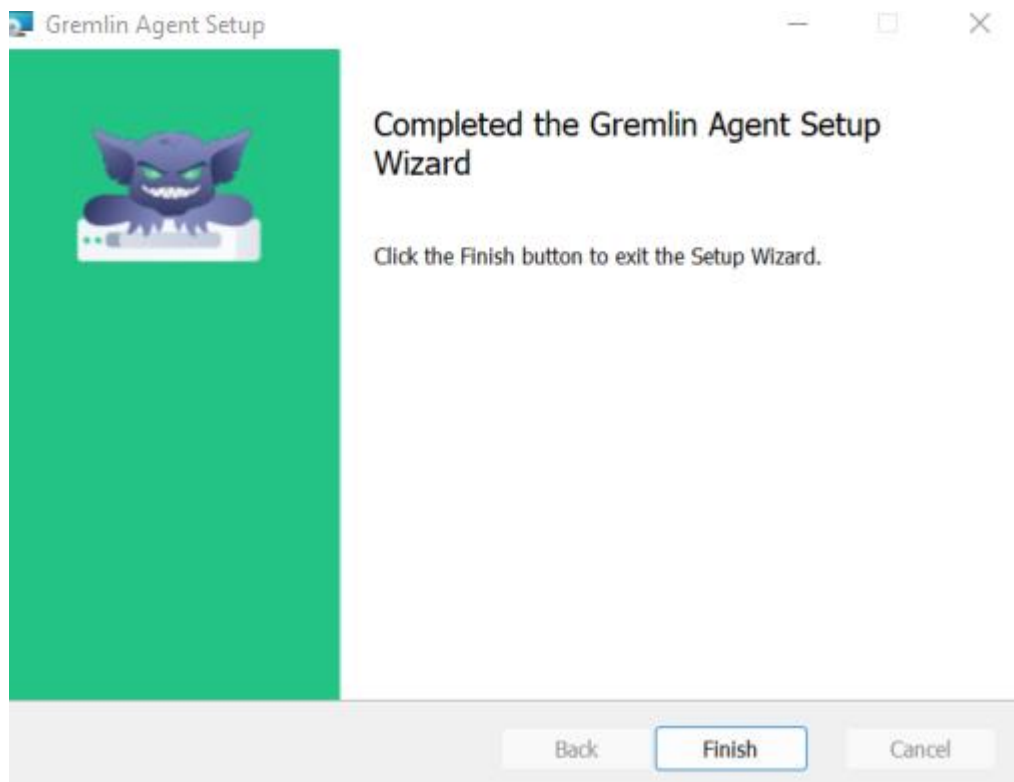






- A dialog box asking to escalate privileges may be displayed. Click Yes.






## 5.2 Step2:

### 1. Create Persons Container

Home > Microsoft.Azure.CosmosDB-20220908125411 | Overview > [gramlinapidemo](#)




 **gramlinapidemo** | Quick start ...  
Azure Cosmos DB account

Search (Ctrl+/) «

Overview  
Activity log  
Access control (IAM)  
Tags  
Diagnose and solve problems  
Cost Management  
**Quick start**  
Notifications  
Data Explorer  
Settings

Congratulations! Your Azure Cosmos DB account was created.  
Now, let's connect to it using a sample app:

**Choose a platform**

 .NET  Gremlin Console  Guided Gremlin Tour

**1 Add a graph**  
In Azure Cosmos DB, you can create containers to store graph elements (vertices and edges).  
[Create 'Persons' container](#)  
Create 'Persons' graph. To see your graph, go to Data Explorer and find the 'graphdb' database.

**2 Download and run your Gremlin Console app**  
Once graph is created, download a sample Gremlin Console app connected to it, extract, build and run.  
[Download](#)

### 2. Download the Gremlin Installer from this CosmoDB Quick start

## gramlinapidemo | Quick start

Search (Ctrl+/)

- 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

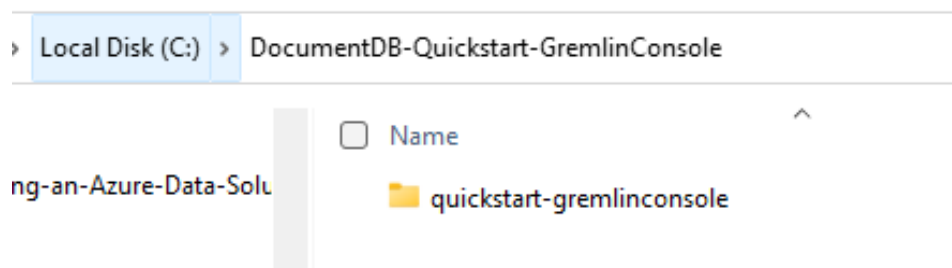
Congratulations! Your Azure Cosmos DB account was created. Now, let's connect to it using a sample app:

**Choose a platform**

- .NET
- Gremlin Console
- Guided Gremlin Tour

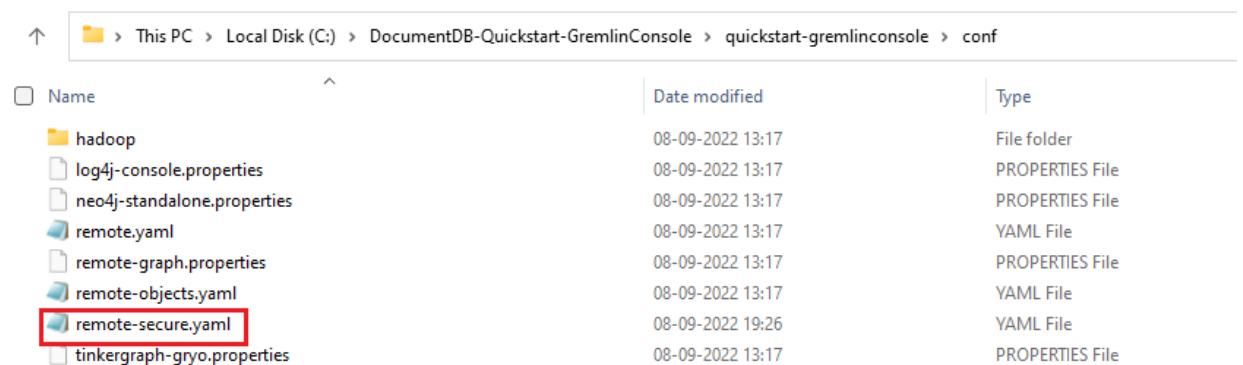
- 1 Add a graph**  
'Persons' graph has been created. To see your graph, go to Data Explorer and find the 'graphdb' database.
- 2 Download and run your Gremlin Console app**  
We created a sample Gremlin Console app connected to your "Persons" graph. Download to query with Gremlin.  
[Download](#)
- 3 Work with data**  
Query and edit your data, add stored procedures, and more using Data Explorer.  
[Open Data Explorer](#)

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



## 6. **Configure Remote-secure.yaml file**

- 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.



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

Setting	Suggested value	Description
hosts	[ <i>account-name</i> .gremlin.cosmos.azure.com]	See the following screenshot. This is the <b>Gremlin URI</b> value on the Overview page of the Azure portal, in square brackets, with the trailing :443/ removed. Note:
port	443	Set to 443.
username	<i>Your username</i>	The resource of the form /dbs/<db>/colls/<coll> where <db> is your database name and <coll> is your collection name.
password	<i>Your primary key</i>	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.

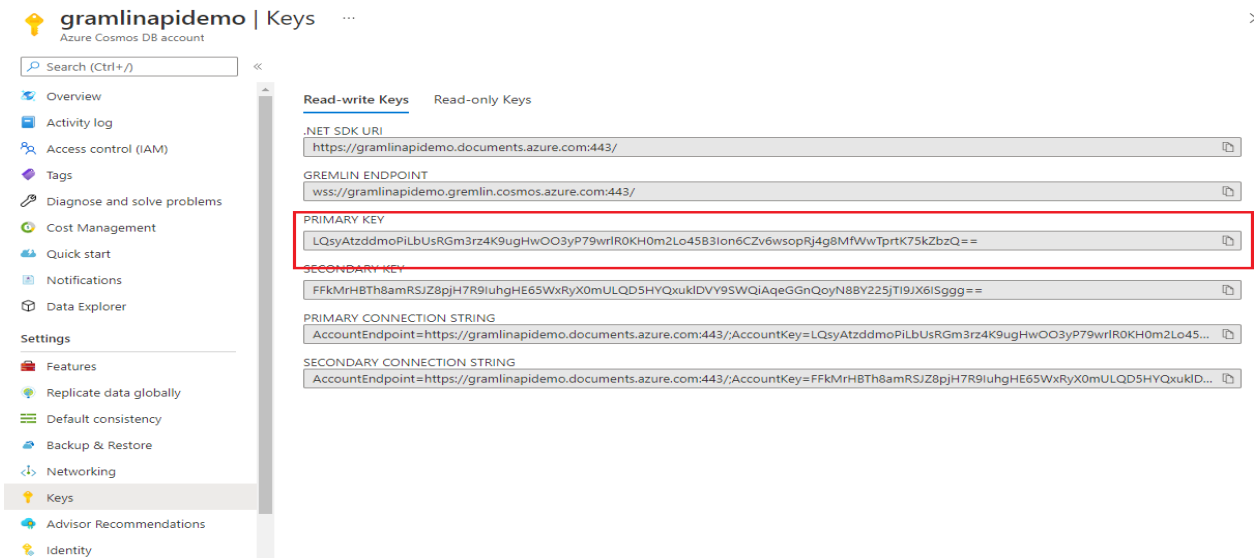
For the hosts value, copy the **Gremlin URI** value from the **Overview** page:

The screenshot shows the Azure portal interface for an Azure Cosmos DB account. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Cost Management, Quick start, Notifications, Data Explorer, and Settings. The main content area displays account details under the 'Essentials' section:

- Status: Online
- Resource group: [bipdemorg](#) (move)
- Subscription: [bipdemorg](#) (move)
- Subscription ID: bc56e867-5e4b-409a-ae12-0ea732bf0203
- Total throughput limit: [1000 RU/s](#)

On the right side, there are links for 'Read Locations' and 'Write Locations', both set to 'West US'. Below these, the 'Gremlin Endpoint' is listed as 'https://gramlinapidemo.documents.azure.com:443/'. The 'Gremlin URI' is highlighted with a red box and is 'wss://gramlinapidemo.gremlin.cosmos.azure.com:443/'. Other settings shown include '.NET SDK URI', 'Free Tier Discount', and 'Opted in'.

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



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

```
remote-secure.yaml - Notepad
File Edit View
# results giving a view into how scripts are executing.
#
# This file will work with:
# - gremlin-server-secure.yaml
#####

hosts: [gramlinapidemo.gremlin.cosmos.azure.com]
port: 443
username: /dbs/sample-database/colls/sample-graph
password: LQsyAtzddmoPiLbUsRGm3rz4K9ugHwOO3yP79wrlR0KH0m2Lo45B3Ion6CZv6wsopRj4g8MfWwTprtK75kZbzQ==
connectionPool: {
  enableSsl: true}
serializer: { className: org.apache.tinkerpop.gremlin.driver.ser.GraphSONMessageSerializerV1d0, config:
{ serializeResultToString: true }}
```

## 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)
-----o00o-(3)-o00o-----
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 tinkerpops.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 gremlinapi.demo.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 - [gremlinapi.demo.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-6c0e-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:b2ad1239-7937-4106-8262-29c12d3d2abd|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:d9e62219-76aa-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:cdd3d924-361f-4a18-b278-6a4c88b088d1,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-bca9-7e8ca4bc3106,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').has('firstName', 'Mary Kay'))
==>[id:8bc40ee1-ec62-4556-bbcc-ae4485460f89,label:knows,type:edge,inVLabel:person,outVLabel:person,inV:8bc40ee1-ec62-4556-bbcc-ae4485460f89,outV:b2ad1239-7937-4106-8262-29c12d3d2abd]
gremlin> g.V().hasLabel('person').has('firstName', 'Robin').addE('knows').to(g.V().hasLabel('person').has('firstName', 'Ben'))
==>[id:6102c9d0-c256-4069-8c24-554178efd59f,label:knows,type:edge,inVLabel:person,outVLabel:person,inV:6102c9d0-c256-4069-8c24-554178efd59f,outV:b42f40ee-56cb-4566-a443-2f3994fb32a1]
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-4644-421ceb836775,value:Thomas]],lastName:[id:3b3834ae-6c0e-4cf0-8c61-b0e8fd205f0a,value:Andersen]],userid:[id:b2ad1239-7937-4106-8262-29c12d3d2abd|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-4644-421ceb836775,value:Thomas]],lastName:[id:3b3834ae-6c0e-4cf0-8c61-b0e8fd205f0a,value:Andersen]],userid:[id:b2ad1239-7937-4106-8262-29c12d3d2abd|pk,value:pk]]]
```

## 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:[age:[id:1217e911-407d-4644-421ceb836775,value:Thomas]],lastName:[id:3b3834ae-6c0e-4cf0-8c61-b0e8fd205f0a,value:Andersen]],userid:[id:b2ad1239-7937-4106-8262-29c12d3d2abd|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-4d
```

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')
```