Gremlin is part of an Apache project providing a Graph computing framework called Tinkerpop. The main page of the project is here: https://tinkerpop.apache.org/. This document is based on the 3.3.2 version, which is the current version at the time of authoring this document.

Some resources for your Gremlin learning journey (it is an alien world for relational folks);

Documentation and Tutorials
Introduction - https://tinkerpop.apache.org/gremlin.html
Getting Started - http://tinkerpop.apache.org/docs/current/tutorials/getting-started/
Gremlin Console - http://tinkerpop.apache.org/docs/current/tutorials/the-gremlin-console/
Formal Documentation - http://tinkerpop.apache.org/docs/3.3.3/reference/
Recipes - http://tinkerpop.apache.org/docs/3.3.3/recipes/

Helpful Resources
Kelvin Lawrence Gremlin Book

- Latest - http://kelvinlawrence.net/book/Gremlin-Graph-Guide.html
- PDF - http://www.kelvinlawrence.net/book/Gremlin-Graph-Guide.pdf
- GitHub - https://github.com/krlawrence/graph

https://academy.datastax.com/resources/getting-started-graph-databases

Videos
Azure Friday - Graphs with Azure Cosmos DB Gremlin API

BRK3311 @ Build 2018 – How to take advantage of scale out graph in Azure Cosmos DB

Channel 9 - Tips for using the Gremlin API with Azure Cosmos DB

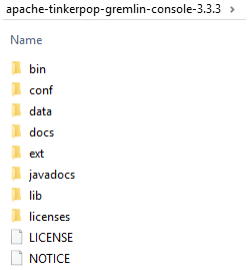
BRK3183 @ Ignite 2018 - Traversing Scalable Graphs with Azure Cosmos DB's Gremlin API

THR2053 @ Ignite 2018 - Common data problems solved with graphs using Azure Cosmos DB Gremlin API

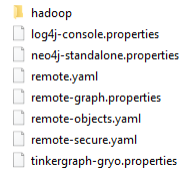
Installing Gremlin Console

Download the Gremlin console zip from a link on the above site – the 3.3.3 version is available here - https://www.apache.org/dyn/closer.lua/tinkerpop/3.3.3/apache-tinkerpop-gremlin-console-3.3.3-bin.zip or look on the download page for the latest stable version - https://tinkerpop.apache.org/downloads.html

Unzip to a location on your computer (i.e. your desktop or documents) – your folder should look like;



Open the conf folder;



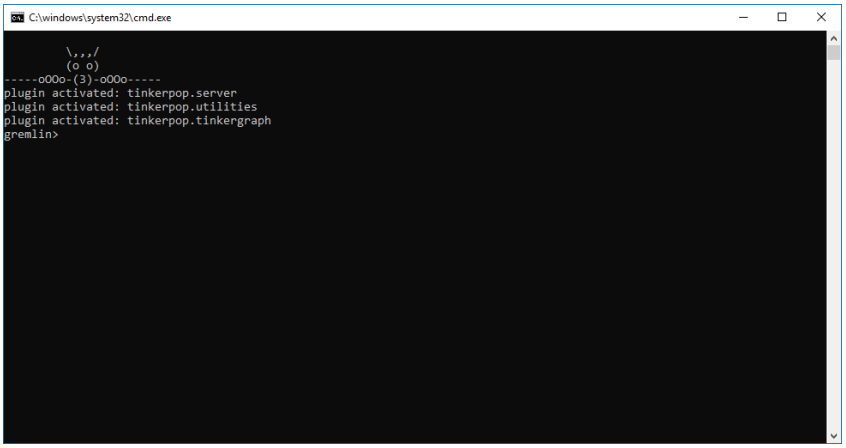
Edit the remote-secure.yaml file and add your Cosmos DB Name, your Graph Database Name, your Graph Name and your Cosmos DB SAS key.

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.

############################################################
# This configuration is meant to have Gremlin Server return
# text serialized objects. The server will toString()
# results giving a view into how scripts are executing.
#
# This file will work with:
# - gremlin-server-secure.yaml
############################################################

hosts: [mvhcosmosdb.gremlin.cosmosdb.azure.com]
port: 443
username: /dbs/GraphDB/colls/testgraph
password: XQetWREqEzxF6fDAdZ1xrwSWvnbUSBCE5n1VAmXrhtouoDHiuXdvej4a0PbXtTIQ6K2Doh7VXX9pzPcer31vJw==
connectionPool: {
  enableSsl: true}
serializer: { className: org.apache.tinkerpop.gremlin.driver.ser.GryoMessageSerializerV3d0, config: { serializeResultToString: true }}
```

Change folders to the \bin folder – there are 2 files – gremlin.bat and gremlin.sh;

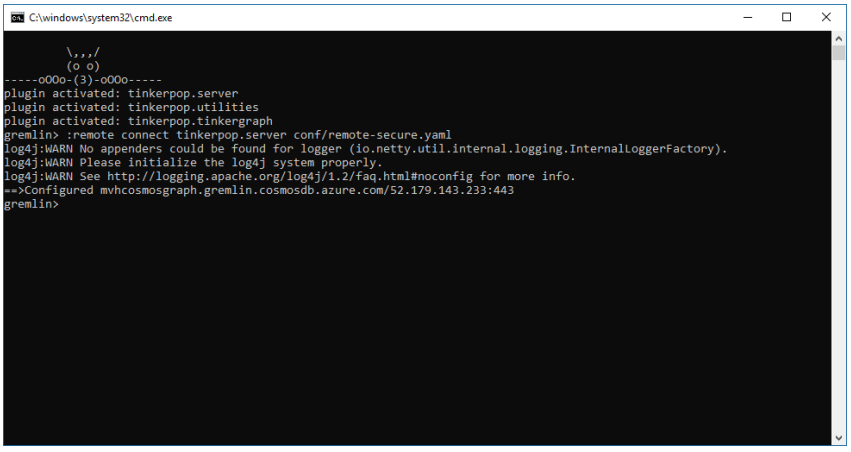
The first time – run the gremlin.bat file as Administrator – this gives the console to create some settings in the registry – otherwise you get some additional error messages when the console starts.

After the initial start, you no longer need to run the bat file as administrator. You can create a shortcut to the .bat and put it on your desktop, if you have put the console files elsewhere.
When Gremlin starts you will see;



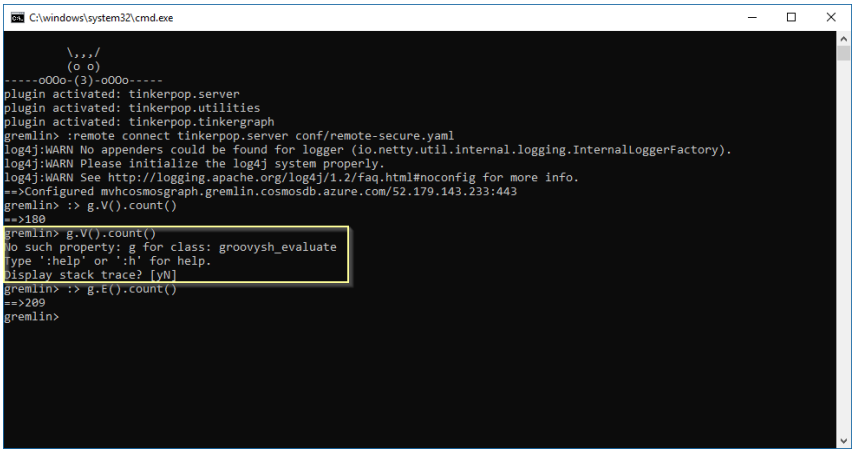
To work with Cosmos DB, you need to connect to it;

:remote connect tinkerpop.server conf/remote-secure.yaml



Ignore the log4j error message (unless you really want to fix it), the console still works.

Note to issue commands to the remote graph server, you need to prefix every command with :> (shortcut for :submit) otherwise you get a No such property error, since there is no local graph set up.



You can play with local graphs and even the Tinkerpop server as an alternative to Cosmos DB.

To load data into a Cosmos DB graph you can use Gremlin commands in the console, use Gremlin.NET in a C# app (or another API in a programming language) or the Graph Bulk Load API (you need special permission to get access to this at the moment: https://github.com/Microsoft/Microsoft.Azure.Graphs.BulkImport/).

Below is a snapshot of a C# sample that creates a graph from a json sample file (from a customer);

```
20 references
static string DoGremlinCmd(string cmd)
{
    string ret = "";

    Task<IReadOnlyCollection<dynamic>> task = gc.SubmitAsync<dynamic>(cmd);
    task.Wait();
    foreach (var result in task.Result)
    {
        string output = JsonConvert.SerializeObject(result);
        Console.WriteLine(String.Format("Cmd:\t{0}\nResult:\t{1}", cmd, output));
        if (ret.Length > 0) ret += " | ";
        ret += output;
    }
    return ret;
}

0 references
static void Main(string[] args)
{
    GremlinServer gs = new GremlinServer(hostname, port, true, "/dbs/" + database + "/colls/" + collection, authKey);
    gc = new GremlinClient(gs, new GraphSON2Reader(), new GraphSON2Writer(), GremlinClient.GraphSON2MimeType);

    string text = File.ReadAllText(@"C:\temp\ResumeData_sample.json");
    List<PersonType> input = JsonConvert.DeserializeObject<List<PersonType>>(text);

    DoGremlinCmd("g.V().drop()");                                   // remove everything
    // string x = DoGremlinCmd("g.V().haslabel('skill').count()");  // get count of skills (if we were adding)
    int skill = 1, job = 1;
    for (int p = 1; p <= input.Count; p++)
    {
        PersonType person = input[p-1];
        DoGremlinCmd("g.addV('person').property('id', 'person." + p.ToString() + "').property('unique_id', '" + person.unique_id + "').property('modified_date', '" + person.modified_date + "')");
        DoGremlinCmd("g.addV('location').property('id', 'location." + p.ToString() + "').property('city', '" + person.location.city + "').property('state', '" + person.location.state +
            "').property('lat', '" + person.location.coordinates[0] + "').property('long', '" + person.location.coordinates[1] + "')");
        DoGremlinCmd("g.V('person." + p.ToString() + "').addE('lives_in').to(g.V('location." + p.ToString() + "'))");
        DoGremlinCmd("g.addV('industry').property('id', 'industry." + p.ToString() + "').property('name', '" + person.industry + "')");
        DoGremlinCmd("g.addV('sub_industry').property('id', 'subindustry." + p.ToString() + "').property('name', '" + person.sub_industry + "')");
        DoGremlinCmd("g.V('subindustry." + p.ToString() + "').addE('parent_industry').to(g.V('industry." + p.ToString() + "'))");
        DoGremlinCmd("g.V('person." + p.ToString() + "').addE('industry_map').to(g.V('industry." + p.ToString() + "'))");
        DoGremlinCmd("g.V('person." + p.ToString() + "').addE('sub_industry_map').to(g.V('subindustry." + p.ToString() + "'))");
        foreach (string s in person.skills.Skills)
        {
            DoGremlinCmd("g.V().has('skill', 'name', '"+ s + "').fold().coalesce(unfold(), addV('skill').property('id', 'skill." + skill.ToString() + "').property('name', '" + s + "'))");
            DoGremlinCmd("g.V('person." + p.ToString() + "').addE('has_skill').to(g.V().has('skill', 'name', '" + s + "')).property('recent', 'no')");
            skill++;
        }
        foreach (string s in person.skills.RecentJobSkills)
        {
            DoGremlinCmd("g.V().has('skill', 'name', '" + s + "').fold().coalesce(unfold(), addV('skill').property('id', 'skill." + skill.ToString() + "').property('name', '" + s + "').property('recent', 'yes'))");
            DoGremlinCmd("g.V('person." + p.ToString() + "').out('has_skill').has('name', '" + s + "').fold().coalesce(unfold().property('recent', 'yes'), addE('has_skill').to(g.V().has('skill', 'name', '" + s + "')).property('recent', 'yes'))");
            skill++;
        }
        foreach(ExperienceType e in person.experience_normalized)
        {
            DoGremlinCmd("g.V().has('job', 'name', '" + e.JobTitle + "').fold().coalesce(unfold(), addV('job').property('id', 'job." + job.ToString() + "').property('name', '" + e.JobTitle + "'))");
            DoGremlinCmd("g.V().has('company', 'name', '" + e.Company + "').fold().coalesce(unfold(), addV('company').property('id', 'company." + job.ToString() + "').property('name', '" + e.Company + "'))");
            DoGremlinCmd("g.V('person." + p.ToString() + "').addE('employment').to(g.V().has('job', 'name', '" + e.JobTitle + "'))");
            DoGremlinCmd("g.V('person." + p.ToString() + "').addE('worked_at').to(g.V().has('company', 'name', '" + e.Company + "'))");
            DoGremlinCmd("g.V().has('company', 'name', '" + e.Company + "').addE('title').to(g.V().has('job', 'name', '" + e.JobTitle + "'))");
            DoGremlinCmd("g.V().has('company', 'name', '" + e.Company + "').addE('in_industry').to(g.V('industry." + p.ToString() + "'))");
            job++;
        }
    }

    DoGremlinCmd("g.V('person.1').outE('has_skill').has('recent', 'yes').inV().hasLabel('skill').count()");
}
```

Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data Migration Jumpstart Team (askdmjfordmtools@microsoft.com). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the Azure Database Migration Guide.