Joshua Catoe

08/05/18

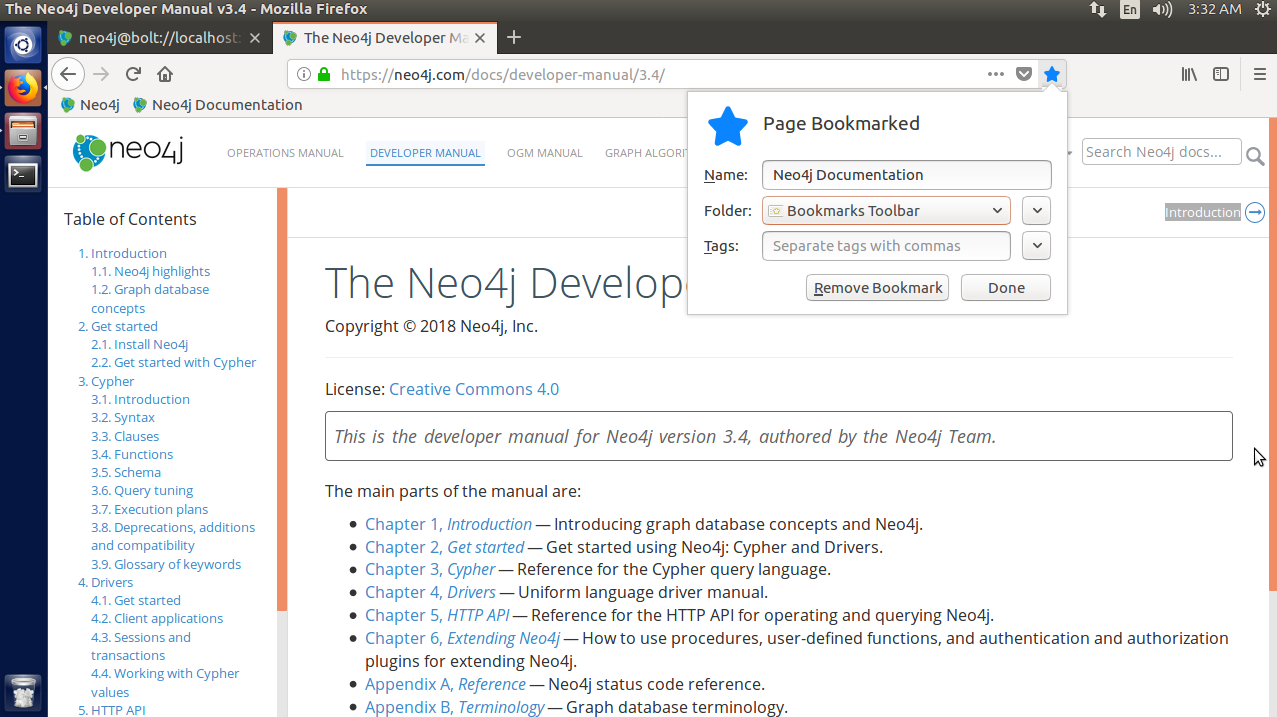
Neo4J

\*Note: I installed the neo4j-gremlin functionality with help from Casey in the discussion forum, but I ran into a problem where I could not create a default Neo4j database/node. I believe this was caused by a missing dependency, but I could not find a solution. In some cases where this prevent me from progressing, I have included code that I’m certain would have worked, had I been able to solve the problem.

**Day 1**

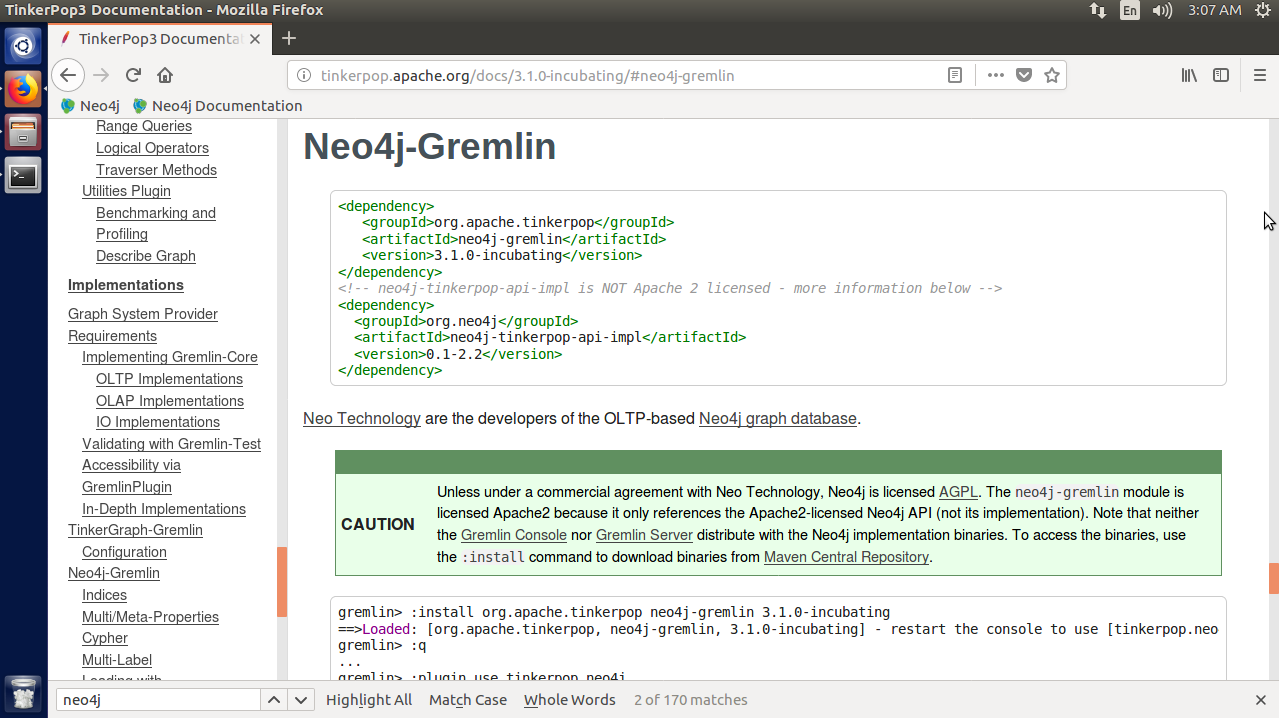
Find:

1. Bookmark the Neo4j wiki.



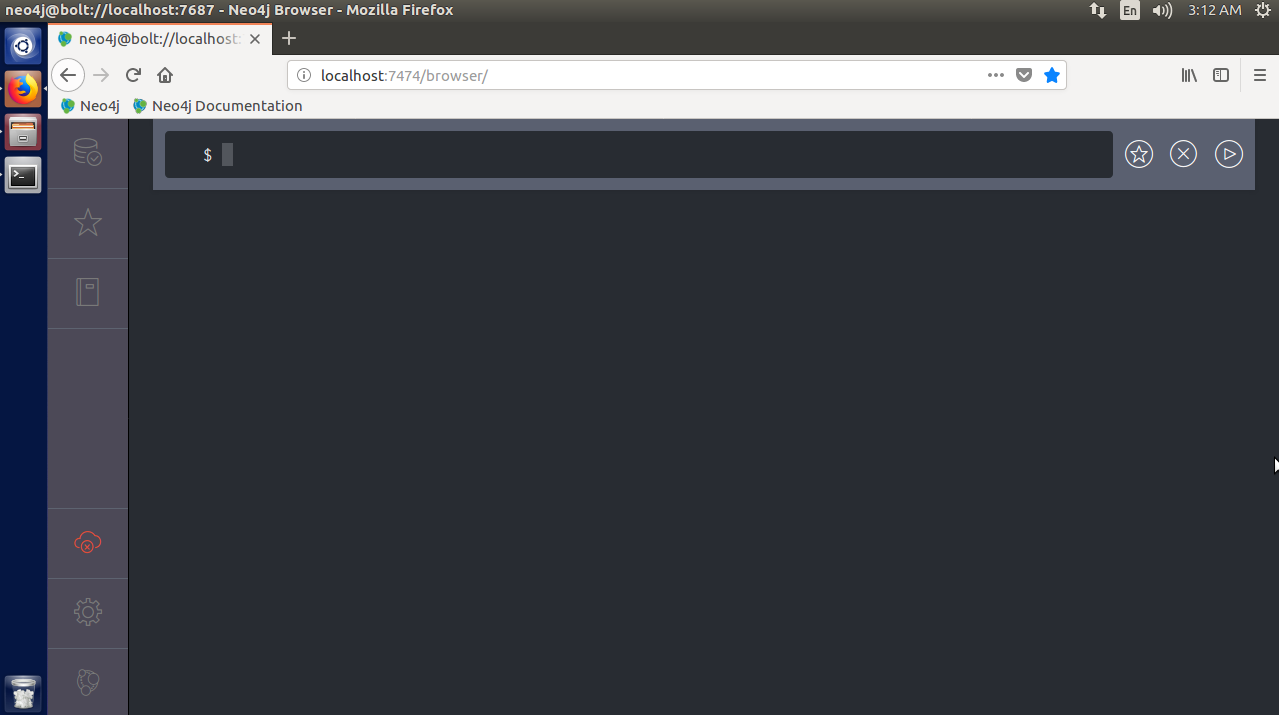
There is no longer a Neo4j wiki. The official documentation can be found by clicking the book icon in the web interface, followed by the Developer Manual link.

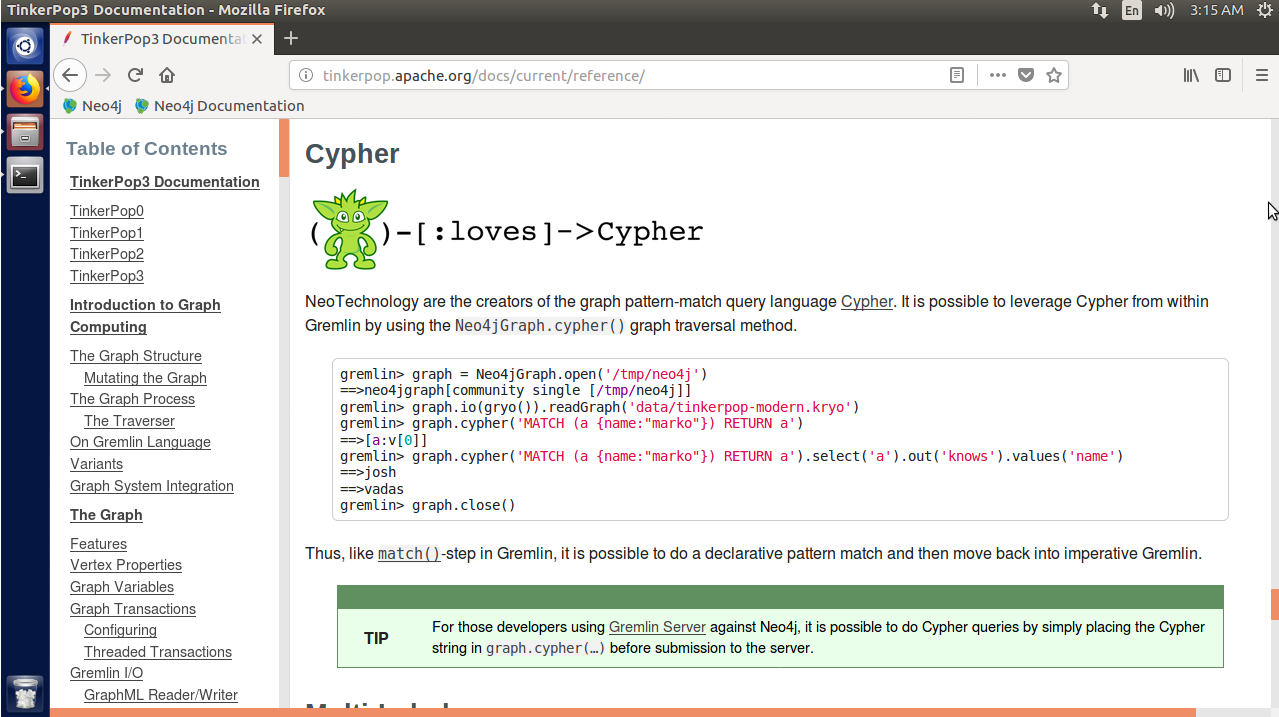
2. Bookmark the Gremlin steps from the wiki or API.



The Neo4j documentation has no instructions for Gremlin, and the web interface has no implementation of it. Instead, you must use the Gremlin Console from Apache and install a plugin for Neo4j. The instructions for doing so can be found in the Tinkerpop 3 documentation under Implementations -> Neo4j-Gremlin.

3. Find two other Neo4j shells (such as the Cypher shell in the admin console).

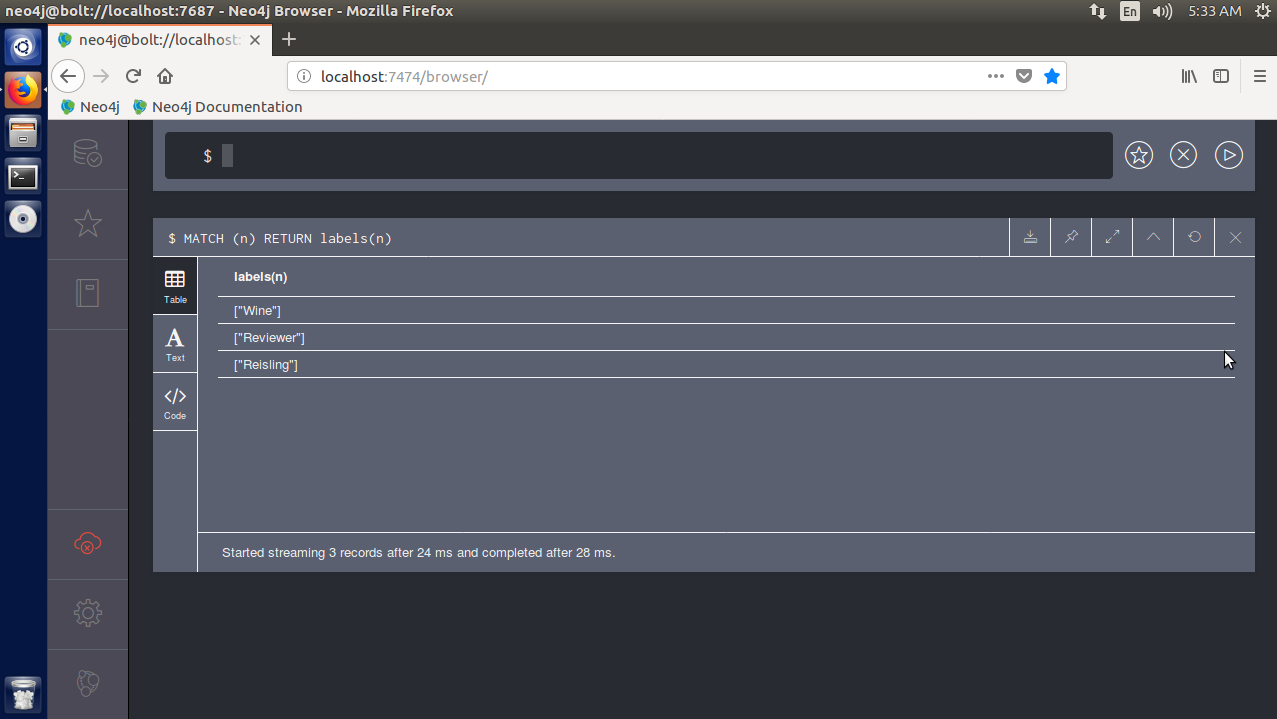




The Cypher shell can be used at <http://localhost:7474/browser>. The only other shell I was able to find was Gremlin. It can be used to run Cypher commands, so I’m assuming that it counts as a Neo4j shell.

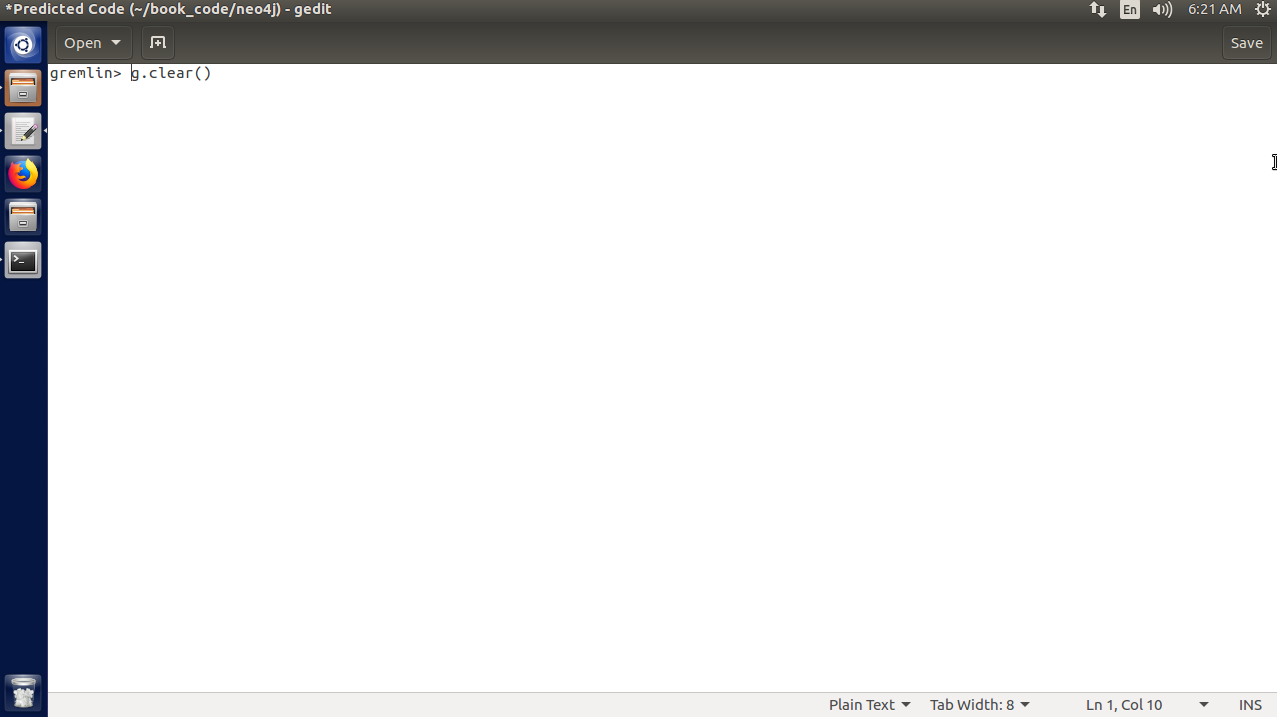
Do:

1. Query all node names with another shell (such as the Cypher query language).



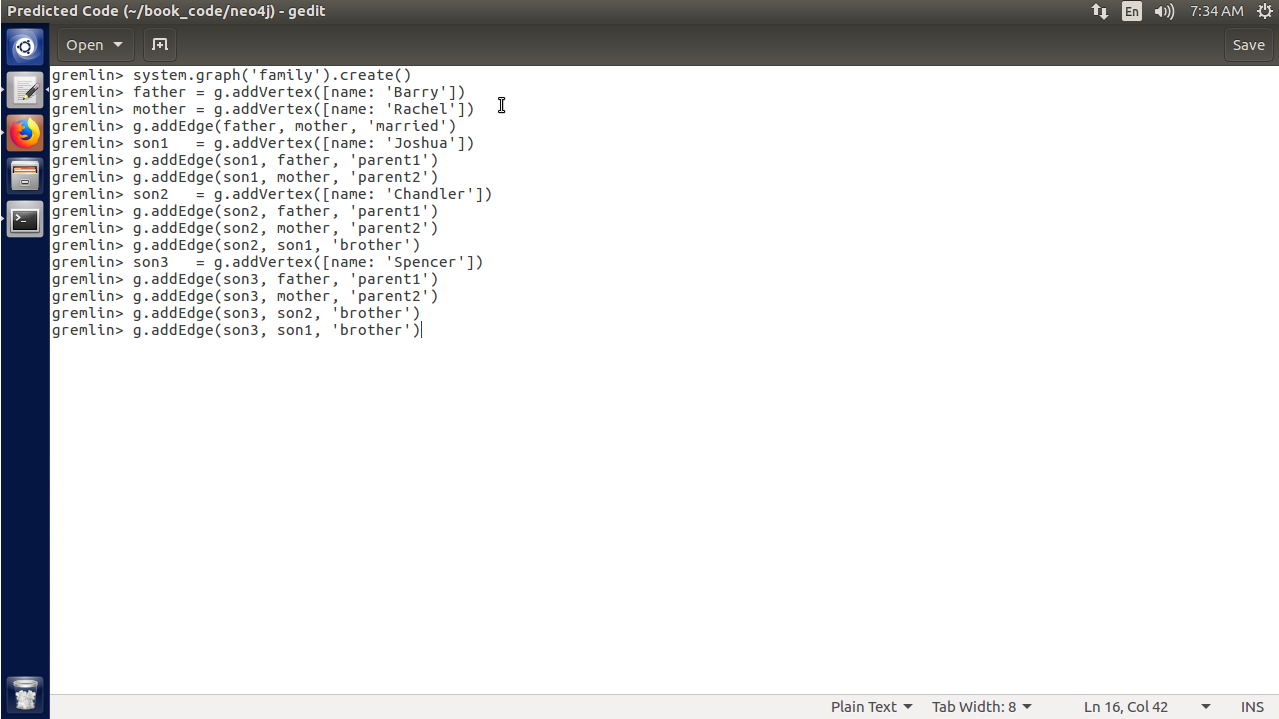
All it takes to query node labels in Cypher is the simple query shown above. **MATCH (n)** to find all nodes and **RETURN labels(n)** to display their label names. The function **labels()** finds all labels for a specific node.

2. Delete all the nodes and edges in your database.



For this, I went with what the book calls the “dangerous” option. **g.clear()** completely clears the graph.

3. Create a new graph that represents your family.

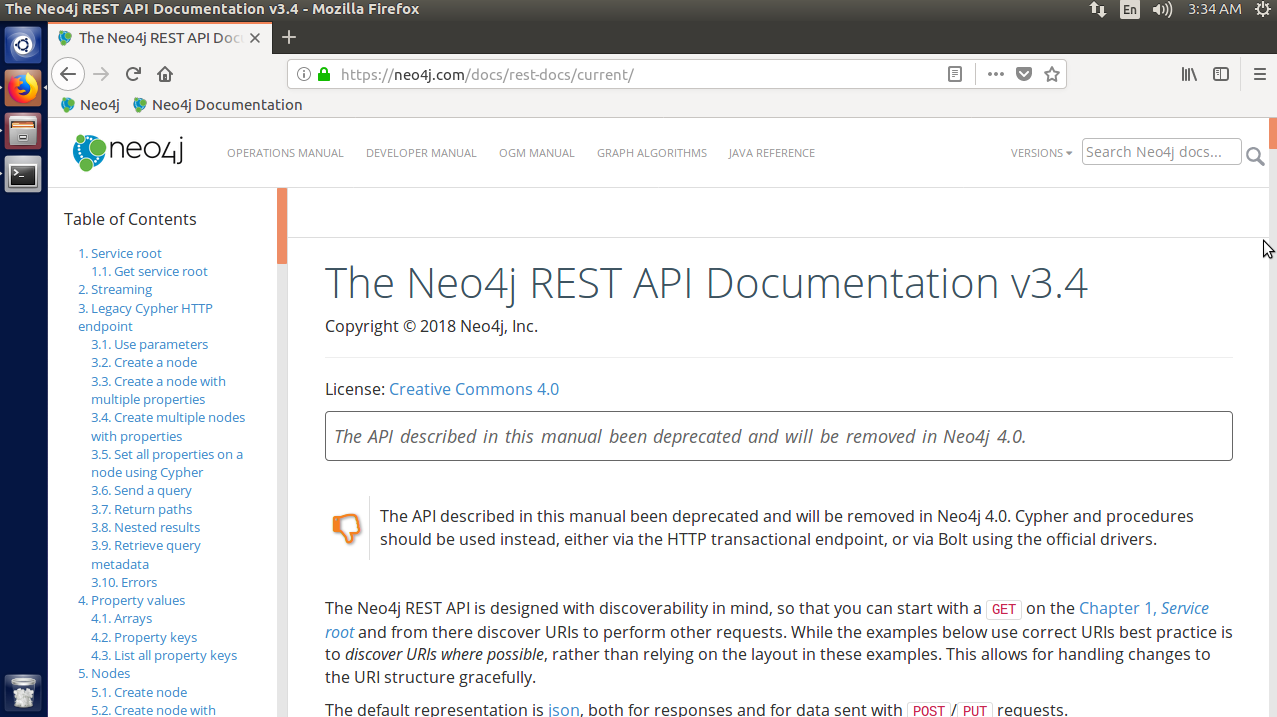


I used the **system.graph().create()** function to actually create the graph, then used **g.addVertex()** to add the vertexes for the family members. In between vertexes, I added the edges with **g.addEdge()** to represent the relationships (married, parent, brother, etc.).

**Day 2**

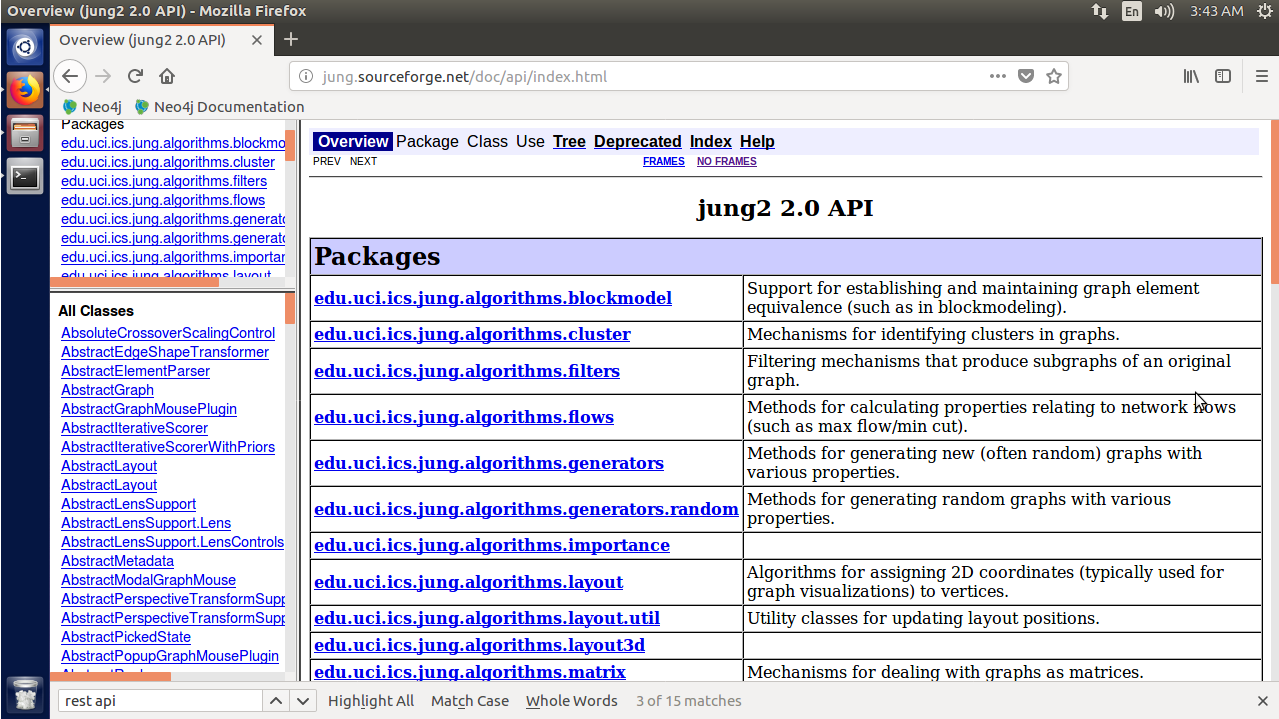
Find:

1. Bookmark the documentation for the Neo4j REST API.



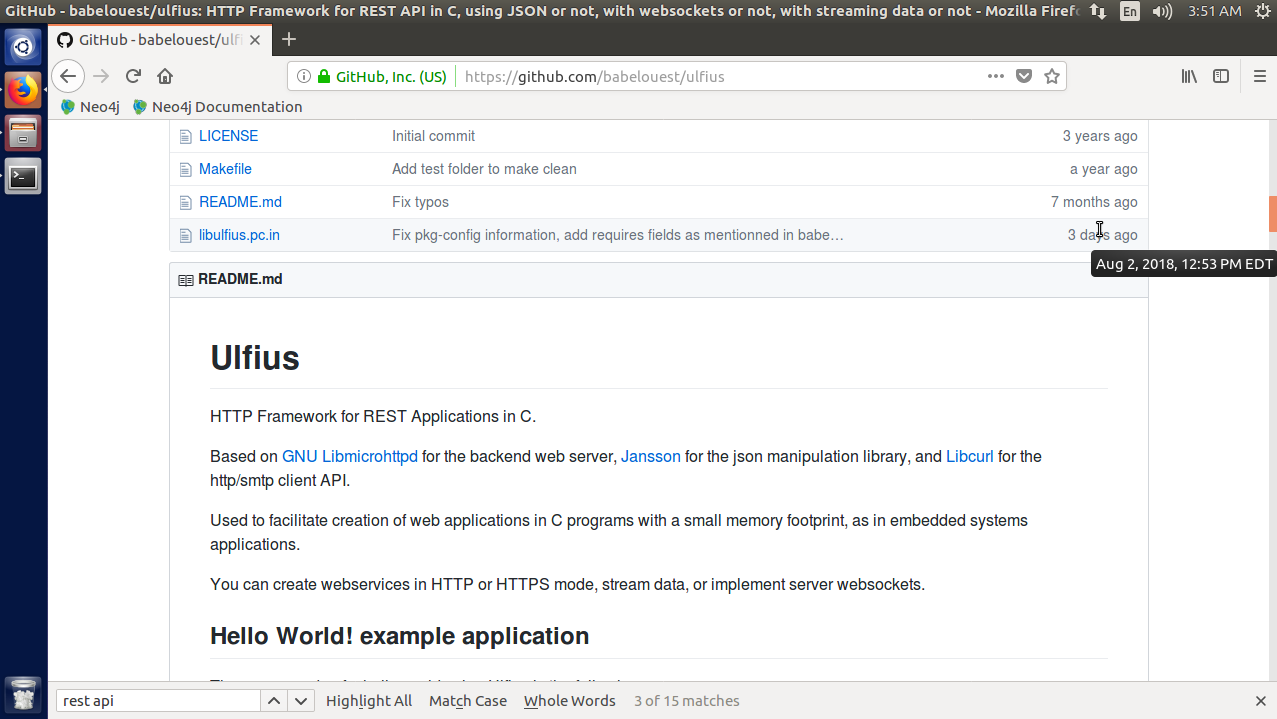
The REST API documentation can be found at <http://neo4j.com/docs/rest-docs/current/>, although it is due to be remove in version 4.0 of Neo4j.

2. Bookmark the API for the JUNG project and the algorithms it implements.



The current JUNG API can be found at <http://jung.sourceforge.net/doc/api/index.html>.

3. Find a binding or REST interface for your favorite programming language.



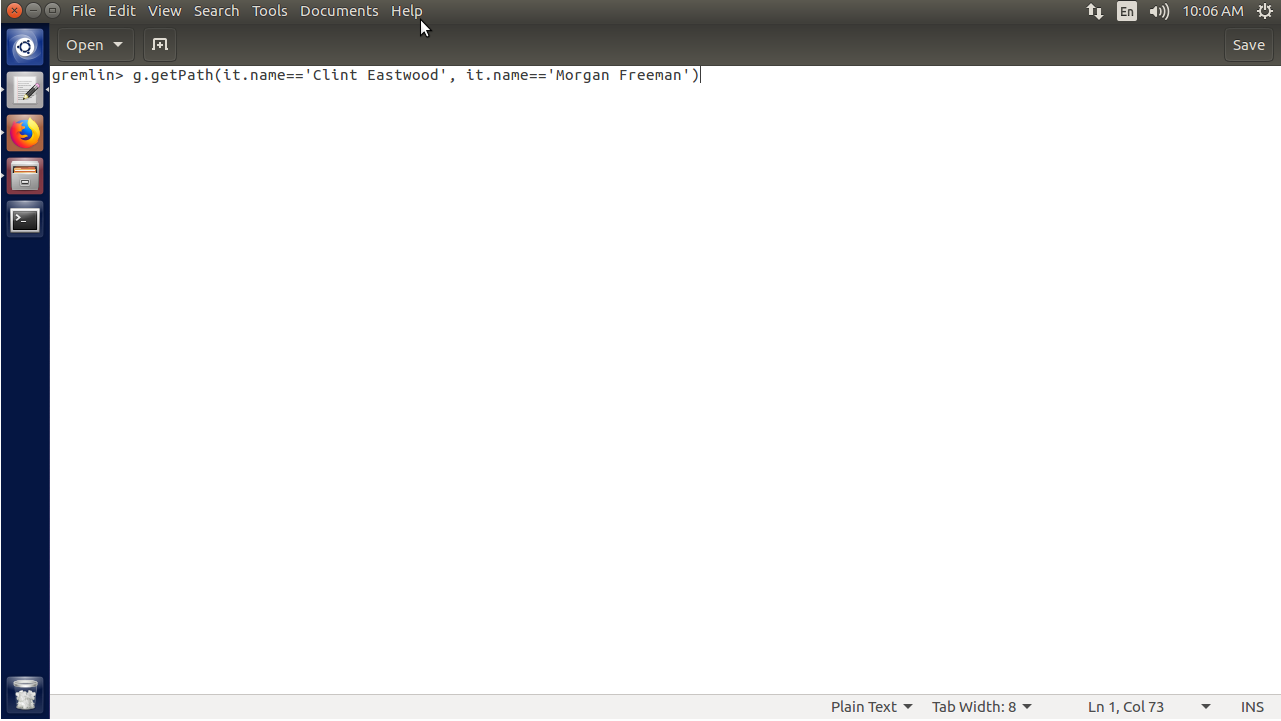
There is a REST interface for C on GitHub called Ulfius.

Do:

1. Turn the path-finding portion of the Kevin Bacon algorithm into its own step. Then implement a general-purpose Groovy function (for example, def actor\_path(g,name1,name2){...}) that accepts the graph and two names and compares the distance.

2. Choose and run one of the many JUNG algorithms on a node (or the data set, if the API demands it).





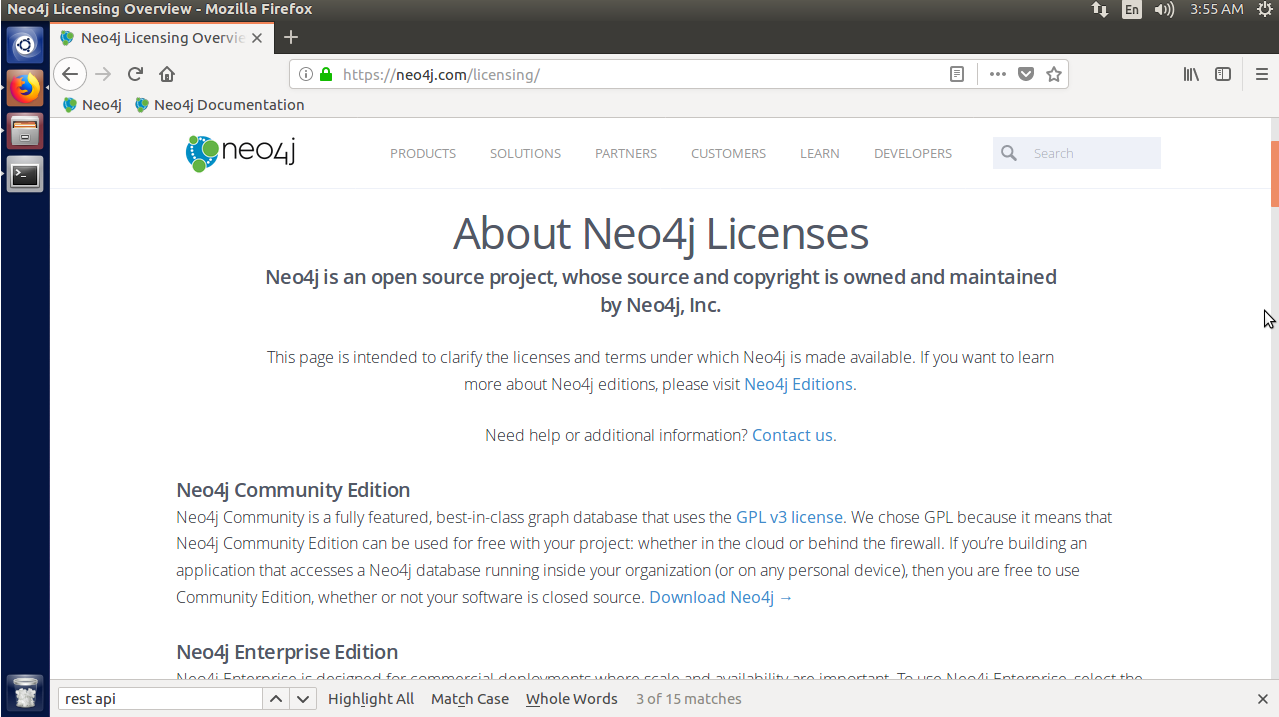
I chose the DijkstraShortestPath algorithm, specifically the **getPath()** method. This takes two vertices as parameters and shortest distance between the two in the graph.

3. Install your driver of choice and use it to manage your company graph with the people and the roles they play, with edges describing their interactions (reports to, works with). If your company is huge, just try your close teams; if you’re with a small organization, try including some customers. Find the most well-connected person in the organization by closest distance to all other nodes.

**Day 3**

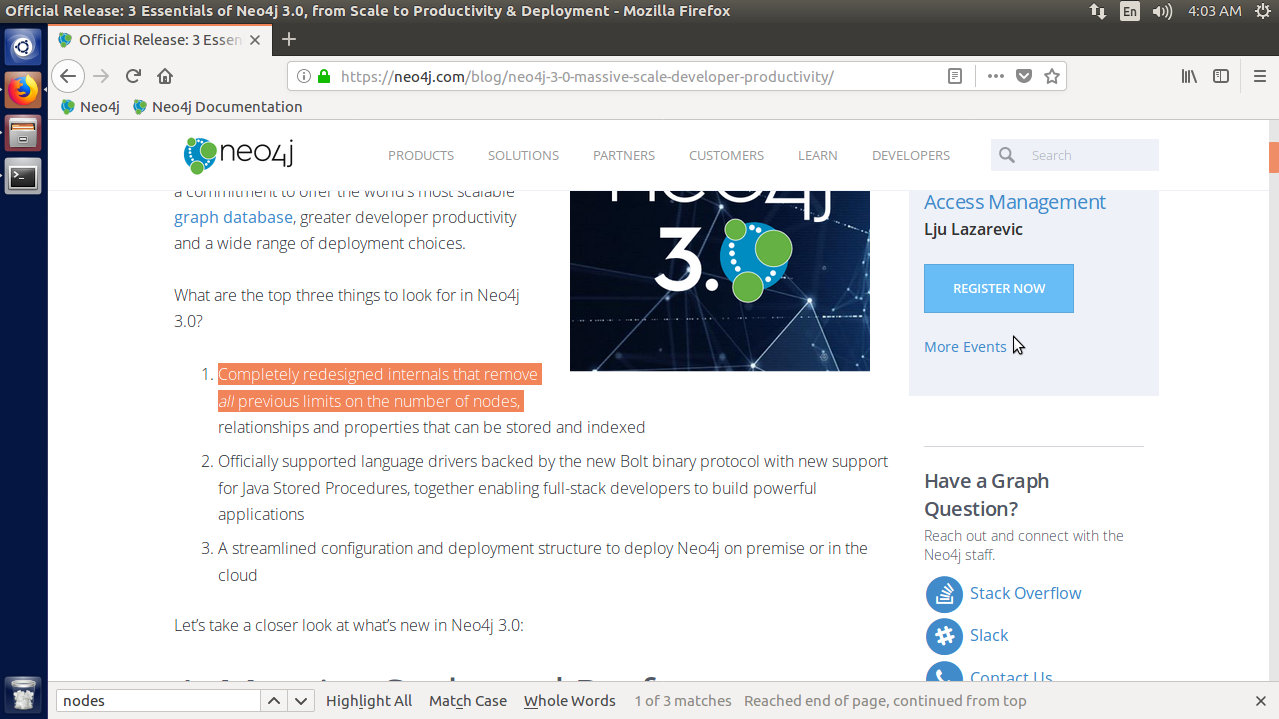
Find:

1. Find the Neo4j licensing guide.



Information about all Neo4j licenses can be found on the Licenses page at <http://neo4j.com/licensing>.

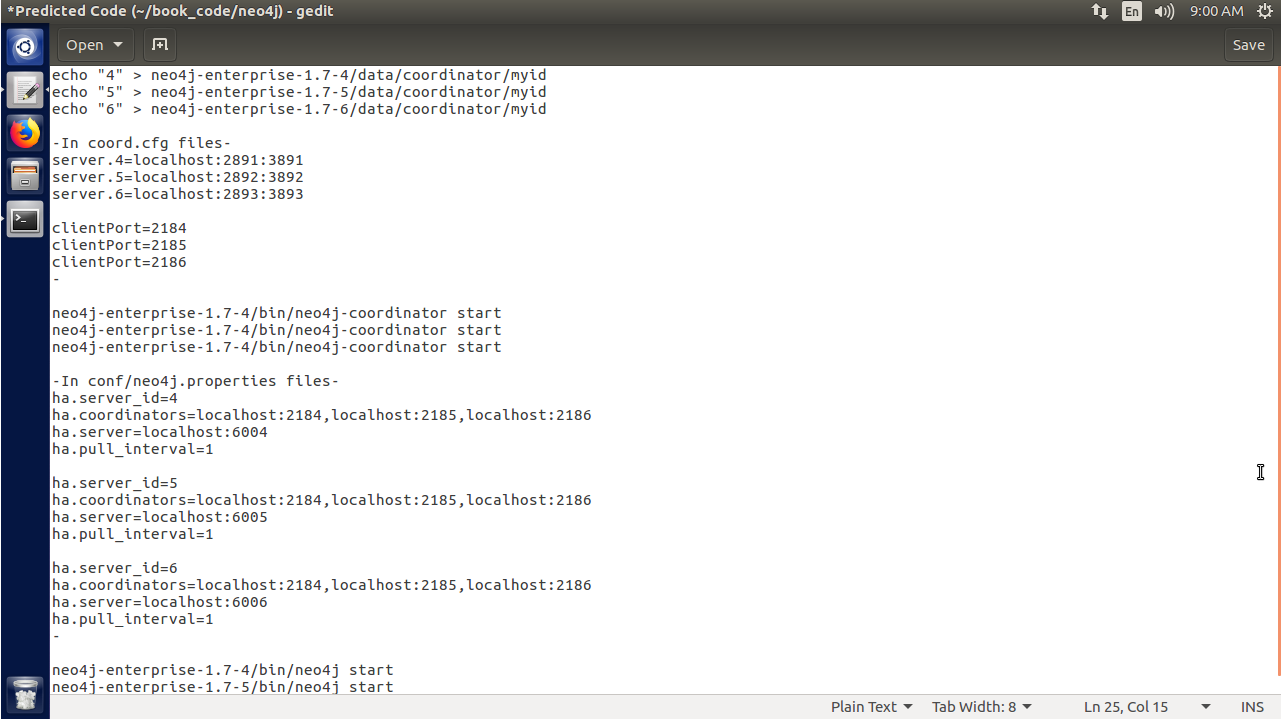
2. Answer the question, “What is the maximum number of nodes supported?”



Detailed in an official Neo4j blog post, as of Neo4j 3.0.0 there are no limits on the number of nodes that can be supported. The previous number was 34 billion (as mentioned in the post).

Do:

1. Replicate Neo4j across three physical servers.



Every Neo4j server must have a different name in a different directory, so I just used 4, 5, and 6. Then I set the ports and client ports for each server in their respective config files and started a coordinator in each server’s directory. Finally, under the properties file for each server I added the high availability settings and then started each server from the console.

2. Set up a load balancer using a web server like Apache or Nginx and connect to the cluster using the REST interface. Execute a Gremlin script command.