Setting up Titan 1.0 with Apache HBase

2 February 2016 | Arielle Bonnici | Big Data, Software Analytics, Software Development

During a recent project we researched a number of graph databases. One of these was Titan, currently regarded as one of the leading graph databases having native integration with the Apache TinkerPop graph stack. The plan was to create a test setup using an **HBase** storage backend which we had running on **Hadoop2**.

Although the official Titan Documentation is quite extensive and a good resource to start wrapping your head around how it works, the examples are quite generic and do not provide a clear picture on getting started with your choice of backend and putting the pieces together.

Getting it all up and running involved several steps and hiccups along the way which consumed a bunch of research time on blogs, forums and discussion groups. Therefore we decided to gather all the info and put together these basic steps in a single blog post in the hope that it will save you some time if you are also trying to get started with Titan on HBase.

This post will guide you to:

- Install Titan
- Install Elasticsearch
- Run the Gremlin console and create a new graph
- Run the Gremlin server for remote connections

Search ...

Do you know how people use your software?
Find Out With Trackerbird
TRY IT FOR FREE

Articles by Category
Articles by Category
Select Category

Recent Posts
Using Open Source Libraries with Closed Source Software
Collecting Custom Data with Event Tracking
Setting up Titan 1.0 with Apache HBase

Related Links
Trackerbird API Reference
Trackerbird Pricing
Trackerbird Support Helpdesk

Archives

Pre-Requisite

Titan requires Java 8 and the `$JAVA_HOME` environment variable needs to be configured to point to the directory where the JRE or JDK is installed.

We also had HBase 0.98.12 running on Hadoop 2.7.1, all set up on an Ubuntu 14.04 machine.

Installing Titan

The first thing you'll need to do is download Titan from https://github.com/thinkaurelius/titan/wiki/Downloads. At the time of writing the latest version of Titan is 1.0 and you will need to download the **titan-1.0.0-hadoop1.zip** file.

Unzip the file you downloaded from a shell:

```
$ unzip titan-1.0.0-hadoop1.zip
```

Let me side-track for a minute here and say that initially we were led to believe that since we have Hadoop 2 in our setup, we needed to use the titan-1.0.0-hadoop2.zip file. When we tried to start the Gremlin console we encountered problems right away and got an error to resolve 'com.thinkaurelius.titan.hadoop.MapReduceIndexManagement'.

> *Invalid import definition: 'com.thinkaurelius.titan.hadoop.MapReduceIndexManagement'; reason: startup failed: script14522627513712042298222.groovy: 1: unable to resolve class com.thinkaurelius.titan.hadoop.MapReduceIndexManagement*

Then if you try to open the graph by executing: graph = TitanFactory.open('conf/titan-hbase-es.properties'), you will get the error below:

> *Could not instantiate implementation: com.thinkaurelius.titan.diskstorage.hbase.HBaseStoreManager*

This issue occurs because the `$TITAN_HOME`/lib/titan-hadoop-

1.0.0.jar is missing when using titan-1.0.0-hadoop2.zip.

Installing Elasticsearch

Titan comes with native support for composite indexes, however if you want to make use of mixed indexes you will need to configure an indexing backend. Titan currently supports Lucene, Solr and Elasticsearch. We chose to install Elasticsearch for the test setup which was pretty straight forward and "just worked". Note that we didn't bother with security as at this stage we just needed to get things going, but that is something you will need to look into if you decide to take it further. Have a look here.

To install Elasticsearch:

Go to the `$TITAN_HOME`/bin folder and search for the elasticsearch jar. In the case of Titan 1.0 it comes with Elasticsearch 1.5.1 therefore you should find elasticsearch-1.5.1.jar. This is the Elasticsearch version you should install.

Open a shell and download the corresponding Elasticsearch version:

```
$ wget
https://download.elastic.co/elasticsearch/elasticsearch
1.5.1.deb
```

Install it by running the `dpkg` command:

```
$ sudo dpkg -i elasticsearch-1.5.1.deb
```

Elasticsearch will be installed in /usr/share/elasticsearch and its configuration files will be located in /etc/elasticsearch.

Run the following command so that Elasticsearch starts and stops automatically:

```
$ sudo update-rc.d elasticsearch defaults
```

Open the /etc/elasticsearch/Elasticsearch.yaml file and set a name for the node and cluster (remember to remove the # at the beginning of the lines):

```
cluster.name: mycluster
```

```
node.name: mynode
```

For testing purposes, you can set the number of shards and replicas as follows:

```
index.number_of_shards: 1
index.number_of_replicas: 0
```

Save the changes and start Elasticsearch:

```
$ sudo service elasticsearch start
```

Running the Gremlin Console

You are now ready to connect to Titan, create a graph and start playing with it. Using the Gremlin console is a good starting point to do that. This will create a new table in HBase called 'titan'.

Open `$TITAN_HOME`/conf/titan-hbase-es.properties, set the storage hostname / IP of your HBase server(s) and save the file:

```
storage.hostname: 127.0.0.1
```

We have left the default 127.0.0.1 since we have everything set up locally.

Open a shell and change the working directory to the Titan home folder:

```
$ cd titan-1.0.0-hadoop1
```

Start the Gremlin console:

```
$ sudo ./bin/gremlin.sh
```

If you omit the `sudo`, you'll encounter an access denied error when Gremlin is loading:

> *Exception in thread "main" java.io.FileNotFoundException: /usr/local/titan-1.0.0-hadoop1/ext/plugins.txt (Permissions denied)*

Open the graph and specify the Titan / HBase / Elasticsearch properties file:

```
graph = TitanFactory.open('conf/titan-hbase-
es.properties')
```

If that works you will see:

```
==>standardtitangraph[hbase:[127.0.0.1]]
```

The path to the properties file you specify is relative to the current working directory. If you didn't change the working directory to the Titan home folder before running the Gremlin console, you will encounter an error here:

> *Backend shorthand unknown: /conf/titan-hbase-es.properties*

Or if you try to specify the complete path to the properties file:

> *Could not open global configuration*

At this stage you can start adding vertices and edges to your graph. If you're new to the Gremlin language, you will find that this presentation on The Gremlin Graph Traversal Language will help to get you started, as well as the TinkerPop3 Documentation.

Running the Gremlin Server for Remote Connections

When you connect to the Titan graph from the Gremlin console, the graph is only accessible through that session and once the Gremlin console is stopped Titan will also stop, since its main thread of execution was part of the Gremlin shell. To run Titan as a long running process which you'll be able to connect to remotely, you will need to configure and run the Gremlin server.

Copy $TITAN_HOME/conf/titan-hbase-es.properties to the $TITAN_HOME/conf/gremlin-server directory and add the following:

```
gremlin.graph=com.thinkaurelius.titan.core.TtanFactory
storage.hbase.tablename = titan
```

Open the$TITAN_HOME/ conf/gremlin-server/gremlin-server.yaml and change:

```
host: localhost
```

to:

```
host: 0.0.0.0
```

Find the graph entry and change:

```
graph: conf/gremlin-server/titan-berkelyje-
server.properties
```

to:

```
graph: conf/gremlin-server/titan-hbase-es.properties
```

Open a shell and change the working directory to the Titan home folder:

```
$ cd titan-1.0.0-hadoop1
```

Start the Gremlin server:

```
$ sudo ./bin/gremlin-server.sh
```

You can test the connection by opening another shell and starting the Gremlin console (see above), then run the following command:

```
:remote connect tinkerpop.server conf/remote.yaml
```

If the connection is successful you will see:

```
==>Connected – localhost/127.0.0.1:8182
```

You will then need to enter `:>` before your Gremlin commands, so they will be sent to the currently active remote connection:

```
:> g = graph.traversal()
```

We hope this blog post has provided some useful information to make your Titan implementation less painful.

G+1 1 Like 8

Comments are closed.

EXPLORE TRACKERBIRD

Overview

How It Works

Why Trackerbird

Business Benefits

ABOUT US

About Trackerbird

Supporting Startups

CONTACT

Contact Trackerbird

Partners & Affiliates

FOLLOW US:

rss

facebook

twitter

linkedin