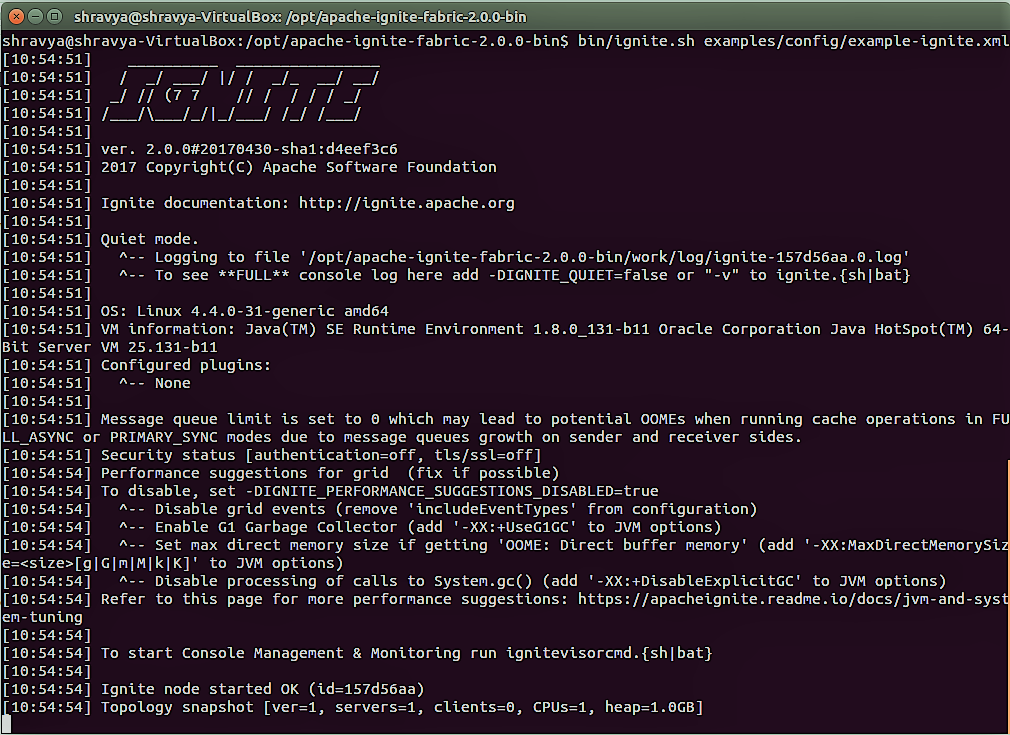
**WORKING WITH APACHE IGNITE**

v2.0.0

The aim of this section of the project is to generate a stream of random messages, store it in an Ignite cache and perform SQL-99 queries on these messages.

**Configuring Ignite**

* To begin with, install JDK(v1.8.0) and Apache ZooKeeper(v3.4.10).
* Download and unzip **apache-ignite-fabric-2.0.0-b1-bin.zip**.
* Set $IGNITE\_HOME to point to this directory.
* Run, **bin/ignite.sh examples/config/example-ignite.xml** to start an Ignite node. This uses the default configurations for the Ignite instance as specified in example-ignite.xml.
* By default, the node runs as server and a space of 1 GB is allocated to the heap.

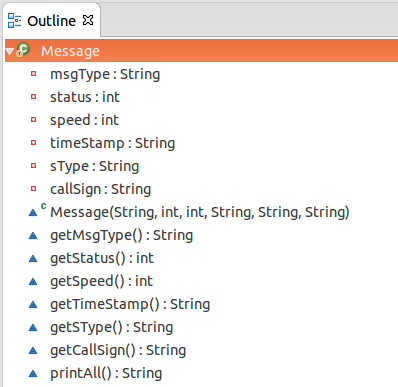


**Creating the Ignite Project**

Dependencies

* Ignite-core : 2.0.0
* Ignite-spring : 2.0.0
* Ignite-indexing : 2.0.0
* Mysql-connector-java : 5.1.42

The Message Class

Create a Java class called Message with the following attributes and member functions. Use the constructor to initialize all the attributes. An object of this class will be passed as value to the key, which is the Message ID (mid:int).

Other than the attributes and the constructor, getter and setter methods can be defined on each attribute if there is need to access them individually outside the class. It can't be done directly since they are private members of the class. A method to display all the attributes can also be defined, which makes it easier to display results on querying.

Message Generator Class

This class connects the different parts together. It generates a stream of 1 million messages. Each attribute of the message is randomly generated, in accordance with the following format-

|  |  |  |
| --- | --- | --- |
| Attribute | Length | Format |
| id (Key) | 9 | MSG000000 - MSG999999 |
| msgType | 6 | Type00 - Type09 |
| status | 2 | 0 - 15 |
| speed | 3 | 0 - 150 |
| timeStamp | 12 | DDHHMMZMONYY (DTG format) |
| sType | 8 | XXXXXXXX |
| callSign | 5 | XXXXX |

Where, X – any alphanumeric character,

DD – date, HH – hour, MM – minute, Z – time zone, MON – month, YY – year.

After the generation of a message, it is loaded into the Ignite cache as <String, Message> type. At the end of the program execution, the cache contains 1 million messages as specified.

These are now ready to be queried.

**Queries**

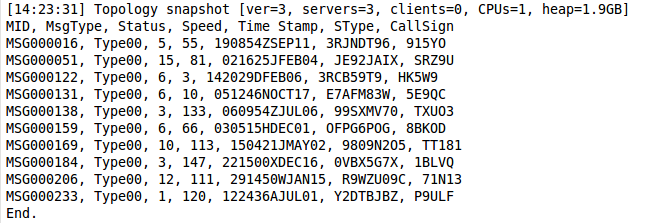
Ignite supports free-form SQL queries. SQL syntax is ANSI-99 compliant. Any SQL function, any aggregation, any grouping can be used and Ignite will figure out where to fetch the results from.

To do this programmatically, SqlQuery and SqlFieldsQuery can be used. SqlQuery is mostly used here, with SqlFieldsQuery being used if there is a need to select only specific fields, instead of selecting the whole object.

The following are a few example queries-

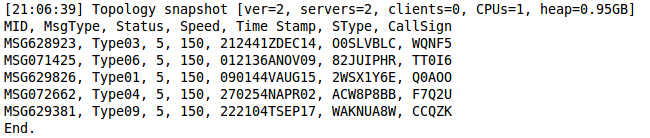
1. To pick first 10 messages with message type "Type00".

SELECT \* FROM Message WHERE msgType='Type00' LIMIT 10



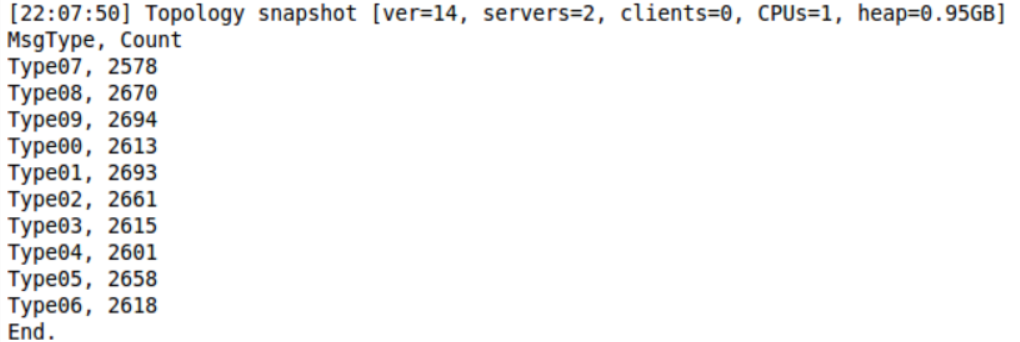
1. To select top 5 fastest messages with status 5.

SELECT \* FROM Message WHERE status=5 ORDER BY speed DESC LIMIT 5



1. To get total number of messages of each type in the year '15.

SELECT msgType, COUNT(timeStamp) FROM Message WHERE timeStamp LIKE '%15' GROUP BY msgType



This