Abstract

The purpose of this document is to demonstrate how developers can create a meaningful business application using the most agile and developer friendly stack for creating enterprise class, production ready applications. The full development stack is referred to as ANEML (Angular, Node.js, Express, MarkLogic) and is pronounced "***Animal***". Using a clever moniker, such as ANEML, can be helpful to wining hearts and minds of the development community, similar to the MEAN stack (see http://mean.io)

The demonstration application is a simple application called TweetDeck. This application allows a user to search for tweets stored in a MarkLogic database and plot the geographic location of the Tweeter on Google Maps.

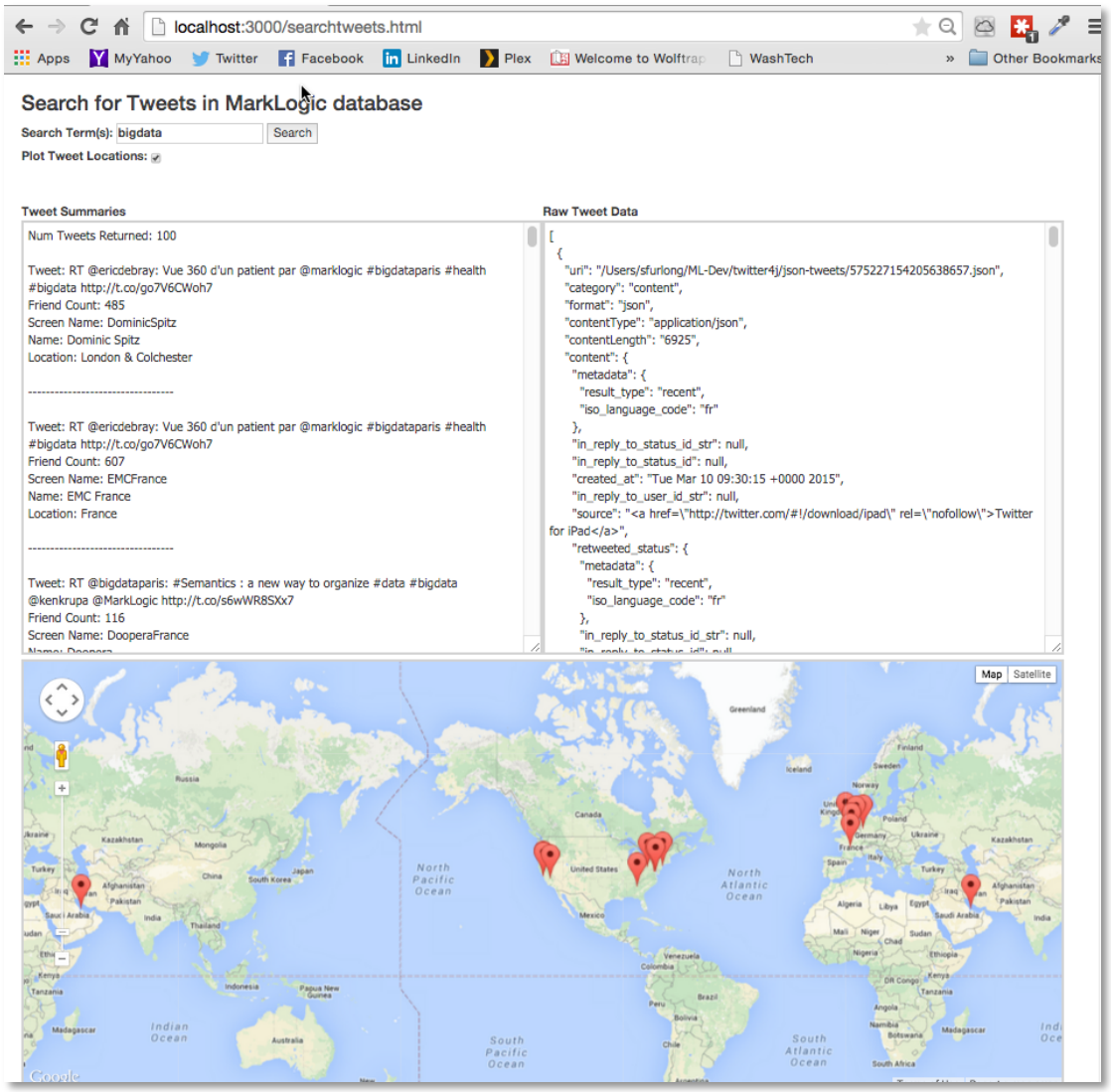


Figure 1 TweetDeck a Sample ANEML Application

The rest of this document describes a step-by-step guide on how to recreate this sample application using the ANEML stack.

MarkLogic Internal Only 2015

I developed this application as a way to learn the MarkLogic 8 platform after joining the company. I hope that others new to MarkLogic 8 will find this project useful. For further information, assistance, or comment, please contact Steve Furlong at steve.furlong@marklogic.com.

Architecture Overview

The ANEML stack is comprised of Angular JS, Node JS, Express, and the MarkLogic database platform. This stack represents a modern development stack where JavaScript is used on the client-tier, middle-tier, and database tier (using JSON document objects). This allows the full stack developer to have language and toolset fidelity through all tiers of an application development cycle.

The best source for more information on each of these components can be found here:

- Angular JS – https://angularjs.org
- Node JS - https://nodejs.org
- Express JS – http://expressjs.com
- MarkLogic – http://developer.marklogic.com/

MarkLogic Internal Only 2015

Installing the ANEML Stack

This section provides a step-by-step guide for installing the minimal components to develop using the ANEML stack. Only the strict minimum components are used to provide the developer with a solid understanding of what layers of the stack contribute to specific architectural capabilities.

Install Node.js

- Download node.js install binaries from https://nodejs.org and install like any other application for your platform (i.e. exe, pkg, rpm, etc)
- Once you've installed node, you will have an "npm" command (node package manager) available to install the remaining components of the stack. Test the installation:
  - `$ node --version`
- Create a directory for the TweetDeck application
  - `$ mkdir aneml-workshop`
  - `$ cd aneml-workshop`
- Create a directory for the NodeJS installation
  - `$ mkdir node-js`
  - `$ cd node-js`
- Create a package.json file in the new directory. This file holds various metadata relevant to the project and gives information to npm that allows it to identify the project as well as handle the project's dependencies. OK to just take the default prompts for this command
  - `$ npm init`

Install Angular JS

- Install Angular using the Node Package Manager in the /node-js directory
  - `$ npm install angular –save`
  - `$ npm install angular`

Install Express

- Install Express using the Node Package Manager in the /node-js directory
  - `$ npm install express –save`
  - `$ npm install express`

Install MarkLogic Node.js Client

- Install the MarkLogic Node JS Client Module in the /node-js directory. This module is used by NodeJS to communicate with the MarkLogic database
  - `$ npm install marklogic –save`

Install Bootstrap CSS Style Sheets (OPTIONAL)

- Bootstrap is a very common client side UI style sheet framework developed by Twitter and now in the public domain. This will help beautify out browser based UI page.
  - `$ npm install bootstrap`

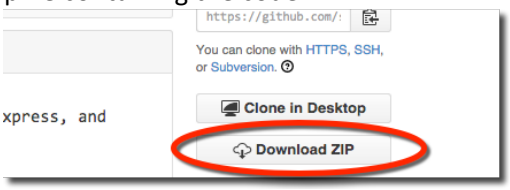
Install MarkLogic

- A very brief description of generic install steps is provided in the appendix of this document.
- Installation of MarkLogic database can be found at http://developer.marklogic.com

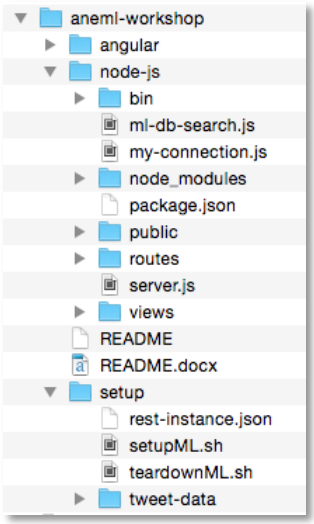
TweekDeck Application Code Deployment

Fetch Code from GitHub

- In your favorite browser, navigate to: https://github.com/sfurlong/aneml
- Download the zipfile containing the code



- Unzip the file into the your /aneml-workshop directory. Note: This should overlay your existing /aneml/workshop/node-js, while keeping its existing contents
- Your directory structure should now look like this:



Configure Your MarkLogic Database

- The database configuration scripts use the MarkLogic Admin REST APIs. To execute the scripts you will need to install CURL on our system. See http://curl.haxx.se/
- The next step will create and load the MarkLogic database using the specifications in the rest-instance.json file. *Note: Port 9010 will be used by the NodeJS to communicate via REST with the MarkLogic database.*

rest-instance.json

```
{
  "rest-api":
  {
    "name": "tweet-deck",
    "group": "Default",
    "database": "tweet-deck-content",
    "modules-database": "tweet-deck-modules",
    "port": "9010",
    "forests-per-host": 1
  }
}
```

- Open the file /aneml-workshop/setup/setupML.sh and configure the environment variables, highlighted below, for your database.

```
ADMIN_UID=admin
ADMIN_PWD=marklogic1
ML_HOST=localhost
ML_ADMIN_PORT=8002
ML_DATA_PORT=9010

# Create the REST instance.  This will also cr
curl --anyauth --user $ADMIN_UID:$ADMIN_PWD -X

# Load Data JSON
curl --anyauth --user $ADMIN_UID:$ADMIN_PWD -X
```

- Run the command to create the database instance, REST API, and Load the tweet data:
  - `$ ./setupML.sh`
- At this time your should have ONE tweet loaded into your tweet-deck database. Roughly 100 other tweets are located in the /setup/tweets directory. There are numerous ways to load the remaining tweets. A good exercise is to use MarkLogic Content Pump (MLCP).

To Run The Demo

Start the Node.js Server

- Open a command shell and change directories to /aneml-workshop/node-js
- Configure the file ml-connection.js to match your MarkLogic database connection details. This will be used by the Node.js server to connect to your database.

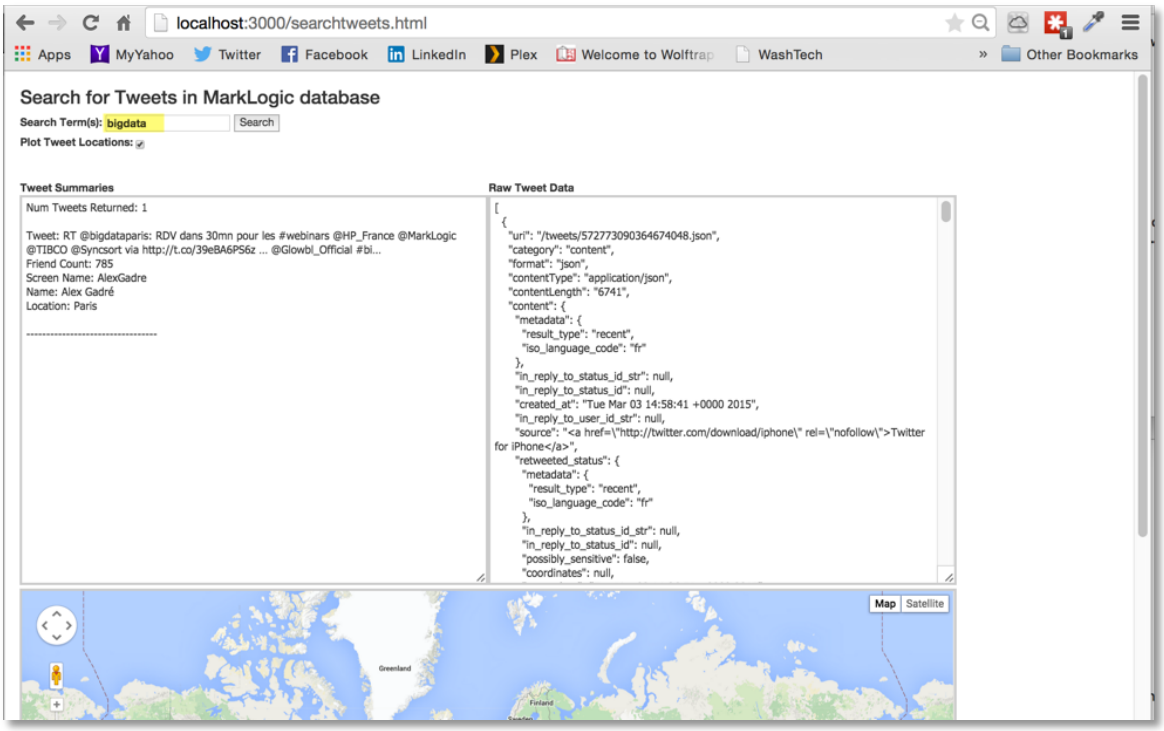
ml-connection.js

```
module.exports = {
  connInfo: {
    host: 'localhost',
    port: 9010,
    user: 'admin',
    password: 'marklogic1'
  }
};
```

- Start the Node JS server using the following command. This will start the serve up the web pages.
  - `$ node server.js`

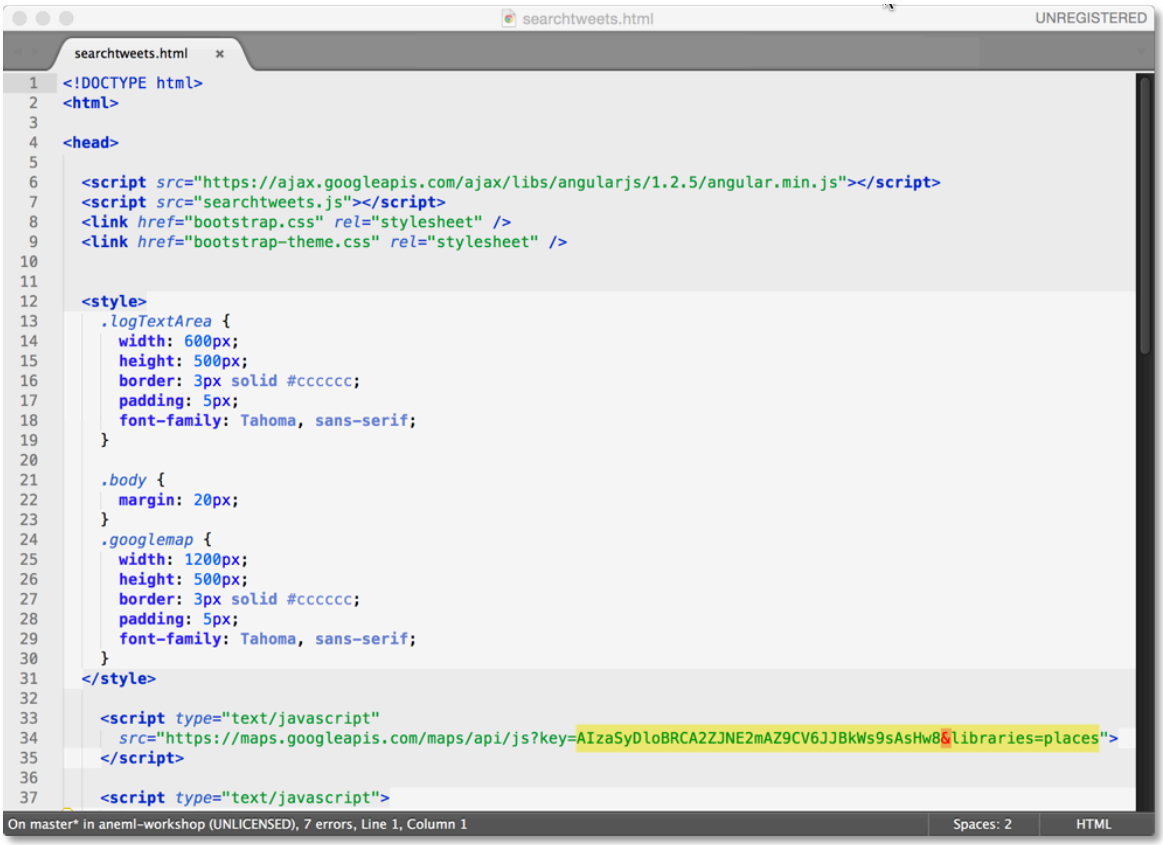
MarkLogic Internal Only 2015

- Open a browser and navigate to http://localhost:3000/searchtweets.html. Use "bigdata" as your search term, this will produce the results shown below.



- Generate a new Google Maps API Key by following the instructions here: https://developers.google.com/maps/documentation/business/mobile/ios/auth
- Update the Google Maps Key in the file /aneml-workshop/node-js/public/search-tweets.html

MarkLogic Internal Only 2015

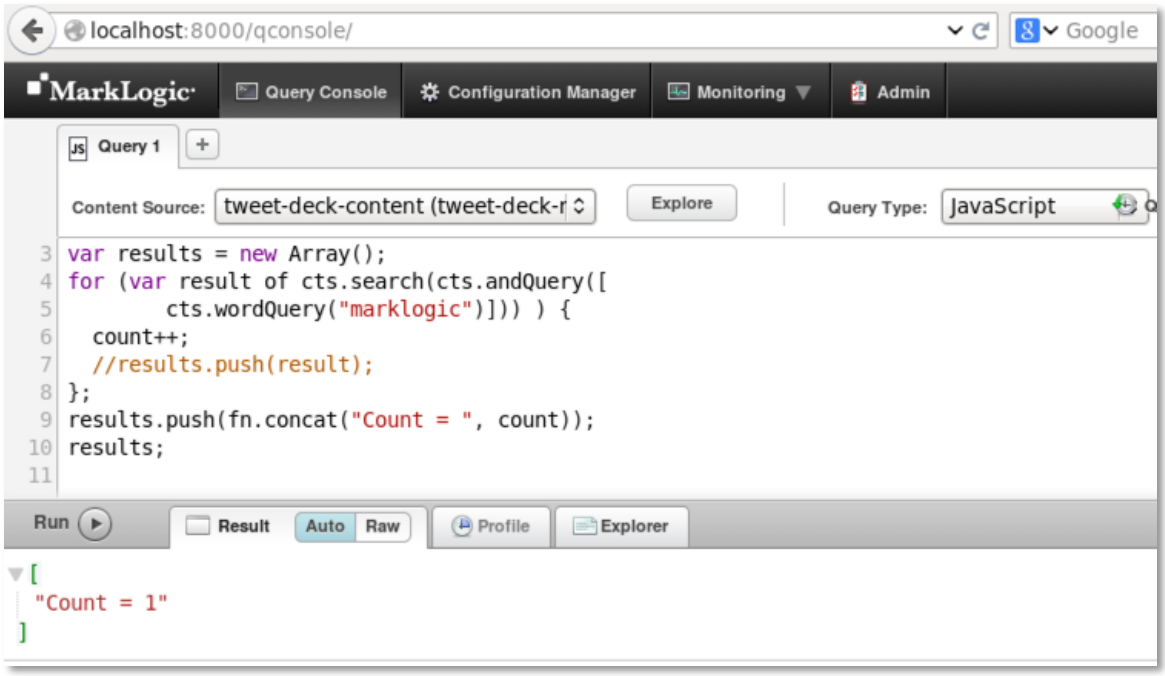


Testing In MarkLogic Query Console

- In your browser, navigate to http://localhost:8000
- In the Query Console, change the Content Source to “tweet-deck-content”
- Set the “Query Type” to “JavaScript”
- Enter the code below and press the “run” button

```
// find all documents with the word "car" and count them
var count = 0;
var results = new Array();
for (var result of cts.search(cts.andQuery([
        cts.wordQuery("marklogic")])) ) {
  count++;
  //results.push(result);
};
results.push(fn.concat("Count = ", count));
results;
```


MarkLogic Internal Only 2015



MarkLogic Internal Only 2015

Appendix

Installing & Uninstalling MarkLogic Database

- `Install: $ rpm –i <<marklogic-rpm-name>>`
- `Start: $ /etc/init.d/MarkLogic start`
- `Stop: $ /etc/init.d/MarkLogic stop`
- `Uninstall: $ rpm –e MarkLogic`
- `Remove DB Files: $ rm –fr /var/opt/MarkLogic`

Working with Git Command Line

Summary of useful Git command line

- Initialize a git repository
  - `$ git init`
- Add files to ignore from git management
  - `$ vi .gitignore`
  - Add any files to be ignored
- Recursively add all files to your local git repository
  - `$ git add -A`
- Commit files to the local repository
  - `$ git commit –m "<<comment>>"`
- Add a remote origin to your local git repository
  - `$ git remote add origin https://sfurlong@github.com/sfurlong/aneml.git`
- Push all local changes to the remote repository
  - `$ git push –u origin master`
- Status commands
  - `$ git status`
  - `$ git log`
  - `$ git diff head`
  - `$ git push`

Future Enhancement Opportunities

1. Query with joins between multiple document types
2. Add pagination to the UI when > 100 tweets are returned
3. AuthN via LDAP
4. AuthZ based on user Role
5. Reporting
   a. Tableau
   b. Pentaho
   c. Qlick
6. Semantics and ontologies

How to Get the Tweets from Twitter

This topic is out of scope for this document. I developed a java program to extract tweets from the Twitter API and write them to disk as JSON formatted files. An open source java library was used called twitter4j. Please contact me for further information.

MarkLogic Internal Only 2015

Change History

Date	Who	Changes
4/30/15	Steve Furlong	Initial Revision

MarkLogic Internal Only 2015