Integrating NODEJS and Kafka

Below are the list required to integrate NodeJS and Kafka. This is a simple HOWTO to get started.

Table of Contents

1. Integrating NODEJS and Kafka
2. Installing KAFKA Single Node - Quick Start.
    - Download and Extract
    - Now we are ready to start all the services required.
    - Next we start server.
    - Creating Topics
    - Send some message
    - Start a Consumer
3. Installing NodeJS on Centos 6.6.
    - Installing `nodejs` and `npm` on centos is very simple.
    - Installing `gcc-c++` and `make`.
    - Later on we will need `kafka-node` lets install that as well.
    - Lets do a test.
    - Lets make some simple changes to exsisting script to handle JSON.
    - NodeJS Kafka Producer - Using `kafka-node`
        - Step 1 - Copy the below script in a file called `producer_nodejs.js`.
        - Step 2 - Start the `kafka` cluster as we already did in `Installation of Kafka`. Assuming topic as `test`
        - Step 3 - Start the consumer service as in the below command.
        - Step 4 - Execute below command. This will send `This is the First Message I am sending` Message to the Kafka consumer.
        - Step 5 - Check on the consumer you will see the message sent from `nodejs`.
    - Sending JSON to NodeJS to Kafka.
        - Step 1 - Create a script called `json_nodejs_kafka.js` with below script.
        - Step 2 - Start above script on the `nodejs` server.
        - Step 3 - Execute `curl` command to send the JSON to `nodejs`.
        - Step 4 - Output on nodejs console
        - Step 5 - Output on the `kafka` consumer side.
        - Useful Links.

Below are the list required to integrate NodeJS and Kafka. This is a simple HOWTO to get started.

Installing KAFKA Single Node - Quick Start.

Download and Extract

Download the `tgz` file and extract.

```
[kafka-admin@kafka Downloads]$ ls
jdk-7u75-linux-x64.rpm  kafka_2.9.2-0.8.2.0.tgz
[kafka-admin@kafka Downloads]$ sudo rpm -ivh jdk-7u75-linux-x64.rpm
...
[kafka-admin@kafka Downloads]$ sudo tar -xzf kafka_kafka_2.9.2-0.8.2.0.tgz -C /opt
[kafka-admin@kafka Downloads]$ cd /opt
[kafka-admin@kafka opt]$ sudo ln -s kafka_2.9.2-0.8.2.0 kafka
[kafka-admin@kafka opt]$ ls
kafka  kafka_2.9.2-0.8.2.0
[kafka-admin@kafka opt]$ sudo chmod kafka-admin:kafka-admin -R kafka
```

Now we are ready to start all the services required.

```
[kafka-admin@kafka opt]$ cd kafka
[kafka-admin@kafka kafka]$ ls
bin  config  libs  LICENSE  logs  NOTICE
[kafka-admin@kafka kafka]$ bin/zookeeper-server-start.sh config/zookeeper.properties
```

This will start us a zookeeper in `localhost` on port `2181`. This configuration can be changed in the `config/zookeeper.properties` file. NOTE : If you want to run the zookeeper on a separate machine make sure the change in the `config/server.properties` so that the kafka server points to the correct zookeeper. By default it points to `localhost:2181`.

Next we start server.

```
[kafka-admin@kafka kafka]$ bin/kafka-server-start.sh config/server.properties
```

NOTE : If you want to start multiple make sure you make multiple copies of the `server.properties` file and change the below information.

1. `broker.id` is the unique identifier for the service.
2. `port` where this server is going to `listen` on.
3. `log.dir` where to right the log.
   config/server-1.properties: broker.id=1 port=9093 log.dir=/tmp/kafka-logs-1
   config/server-2.properties: broker.id=2 port=9094 log.dir=/tmp/kafka-logs-2

Now our server has started, lets assume we start only one server.

Creating Topics

To create a topic just execute below command, this will create a single partition.

```
[kafka-admin@kafka kafka]$ bin/kafka-topics.sh --create --zookeeper localhost:2181 \
                                        --replication-factor 1 --partitions 1 --topic test
```

To check topics currently running. Execute below command.

```
[kafka-admin@kafka kafka]$ bin/kafka-topics.sh --list --zookeeper localhost:2181
test
[kafka-admin@kafka kafka]$
```

We see currently we have only one topic. Now we are all set to send and recv messages.

Send some message

Open up a new terminal and fire up the Kafka producer script as below. And start typing some message `\n` or `cr` will be end of each message

```
[kafka-admin@kafka kafka]$ bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
This is a message
This is a message2
```

Start a Consumer

Open a new terminal and start the consumer.

Option `--from-beginning` will give all the messages from the beginning. You will see 2 messages as we typed above `This is a message` and `This is a message2`.

```
[kafka-admin@kafka kafka]$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 \
                                                              --topic test --from-beginning
This is a message
This is a message2
```

Our single node Kafka cluster is Ready.

Installing NodeJS on Centos 6.6.

Installing `nodejs` and `npm` on centos is very simple.

```
[nginx-admin@nginx ~]$ sudo su
[nginx-admin@nginx ~]# curl -sL https://rpm.nodesource.com/setup | bash -
[nginx-admin@nginx ~]# yum install -y nodejs
```

Installing `gcc-c++` and `make`.

```
[nginx-admin@nginx ~]$ sudo yum install gcc-c++ make
[sudo] password for nginx-admin:
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
Loading mirror speeds from cached hostfile
 * base: mirrors.123host.vn
 * epel: ftp.cuhk.edu.hk
 * extras: centos-hn.viettelidc.com.vn
 * updates: mirrors.vonline.vn
Package 1:make-3.81-20.el6.x86_64 already installed and latest version
Resolving Dependencies
...

Complete!
```

Later on we will need `kafka-node` lets install that as well.

```
[nginx-admin@nginx ~]$ sudo npm install kafka-node
[sudo] password for nginx-admin:

> snappy@3.0.6 install /home/nginx-admin/node_modules/kafka-node/node_modules/snappy
> node-gyp rebuild

gyp WARN EACCES user "root" does not have permission to access the dev dir "/root/.node-gyp/0.10.36"
gyp WARN EACCES attempting to reinstall using temporary dev dir
                 "/home/nginx-admin/node_modules/kafka-node/node_modules/snappy/.node-gyp"
make: Entering directory `/home/nginx-admin/node_modules/kafka-node/node_modules/snappy/build'
  CXX(target) Release/obj.target/snappy/deps/snappy/snappy-1.1.2/snappy-sinksource.o
  CXX(target) Release/obj.target/snappy/deps/snappy/snappy-1.1.2/snappy-stubs-internal.o
  CXX(target) Release/obj.target/snappy/deps/snappy/snappy-1.1.2/snappy.o
  AR(target) Release/obj.target/deps/snappy/snappy.a
  COPY Release/snappy.a
  CXX(target) Release/obj.target/binding/src/binding.o
  SOLINK_MODULE(target) Release/obj.target/binding.node
  SOLINK_MODULE(target) Release/obj.target/binding.node: Finished
  COPY Release/binding.node
make: Leaving directory `/home/nginx-admin/node_modules/kafka-node/node_modules/snappy/build'
kafka-node@0.2.18 node_modules/kafka-node
   buffer-crc32@0.2.5
   retry@0.6.1
   node-uuid@1.4.1
   async@0.7.0
   lodash@2.2.1
   debug@2.1.1 (ms@0.6.2)
   binary@0.3.0 (buffers@0.1.1, chainsaw@0.1.0)
   node-zookeeper-client@0.2.0 (async@0.2.10, underscore@1.4.4)
   buffermaker@1.2.0 (long@1.1.2)
   snappy@3.0.6 (bindings@1.1.1, nan@1.5.3)
[nginx-admin@nginx ~]$ ls
```

Lets do a test.

Create a script called `example.js` with below code.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

Lets start the server on a terminal.

```
[nginx-admin@nginx nodejs]$ node example.js
Server running at http://127.0.0.1:1337/
```

Hit the URL from the browser and We can see `Hello World`. So we are all set.

NodeJS is Ready.

Lets make some simple changes to exsisting script to handle JSON.

Here is a simple script to handle JSON data.

```
//  Getting some 'http' power
var http=require('http');

//  Setting where we are expecting the request to arrive.
//  http://localhost:8125/upload
var request = {
                hostname: 'localhost',
                port: 8125,
                path: '/upload',
                method: 'GET'
            };

//  Lets create a server to wait for request.
http.createServer(function(request, response)
{
    //  Making sure we are waiting for a JSON.
    response.writeHeader(200, {"Content-Type": "application/json"});

    //  request.on waiting for data to arrive.
    request.on('data', function (chunk)
    {
        //  CHUNK which we recive from the clients
        //  For out request we are assuming its going to be a JSON data.
        //  We print it here on the console.
        console.log(chunk.toString('utf8'))
    });
    //end of request
    response.end();
//  Listen on port 8125
}).listen(8125);
```

Lets fire up the script.

```
[nginx-admin@nginx nodejs]$ node node_recv_json.js
```

On a new terminal send some request to our script. Our script is listening on `8125` port.

```
[nginx-admin@nginx nodejs]$ curl -H "Content-Type: application/json" \
                     -d '{"username":"xyz","password":"xyz"}' http://localhost:8125/upload
```

You will see the message received on the script terminal.

```
[nginx-admin@nginx nodejs]$ node node_recv_json.js
{"username":"xyz","password":"xyz"}
```

Now we are all set to do some RND.

NodeJS Kafka Producer - Using `kafka-node`

Now that we have Kafka and NodeJS ready. Lets some data to our Kafka Cluster.

Below is a basic producer code.

below are the Server Details.

1. `nodejs` is the nodejs server
2. `kafka` is the kafka server (single node)

Step 1 - Copy the below script in a file called `producer_nodejs.js`.

```
/*
    Basic producer to send data to kafka from nodejs.
    More Information Here : https://www.npmjs.com/package/kafka-node
*/

//  Using kafka-node - really nice library
//  create a producer and connect to a Zookeeper to send the payloads.
var kafka = require('kafka-node'),
    Producer = kafka.Producer,
    client = new kafka.Client('kafka:2181'),
    producer = new Producer(client);

    /*
        Creating a payload, which takes below information
        'topic'     --> this is the topic we have created in kafka. (test)
        'messages'  --> data which needs to be sent to kafka. (JSON in our case)
        'partition' --> which partition should we send the request to. (default)

                        example command to create a topic in kafka:
                        [kafka@kafka kafka]$ bin/kafka-topics.sh \
                                    --create --zookeeper localhost:2181 \
                                    --replication-factor 1 \
                                    --partitions 1 \
                                    --topic test
                        If there are multiple partition, then we optimize the code here,
                        so that we send request to different partitions.

    */
    payloads = [
       { topic: 'test', messages: 'This is the First Message I am sending', partition: 0 },
    ];


//  producer 'on' ready to send payload to kafka.
producer.on('ready', function(){
    producer.send(payloads, function(err, data){
        console.log(data)
    });
});

producer.on('error', function(err){}
```

Step 2 - Start the `kafka` cluster as we already did in `Installation of Kafka`. Assuming topic as `test`

Step 3 - Start the consumer service as in the below command.

```
[kafka-admin@kafka kafka]$ bin/kafka-console-consumer.sh --zookeeper localhost:2181 \
                                                             --topic test --from-beginning
```

Step 4 - Execute below command. This will send `This is the First Message I am sending` Message to the Kafka consumer.

```
[nodejs-admin@nodejs nodejs]$ node producer_nodejs.js
```

Step 5 - Check on the consumer you will see the message sent from `nodejs`.

```
[kafka-admin@kafka kafka_2.9.2-0.8.2.0]$ bin/kafka-console-consumer.sh \
                                   --zookeeper localhost:2181 --topic test --from-beginning
This is a message
This is another message here
This is the First Message I am sending
```

Sending JSON to NodeJS to Kafka.

What we are trying to achieve ?

1. Send `json` from and browser/`curl` to `nodejs`.
2. `nodejs` will redirect `json` data to `kafka`.
3. Further processing is done on `kafka`.
4. We can then see the `json` arrival in `kafka`, using `kafka-console-consumer.sh` script.

Step 1 - Create a script called `json_nodejs_kafka.js` with below script.

```
/*
    Getting some 'http' power
*/
var http=require('http');

/*
    Setting where we are expecting the request to arrive.
    http://localhost:8125/upload

*/
var request = {
        hostname: 'localhost',
        port: 8125,
        path: '/upload',
        method: 'GET'
};

/*
    Lets create a server to wait for request.
*/
http.createServer(function(request, response)
{
    /*
        Making sure we are waiting for a JSON.
    */
    response.writeHeader(200, {"Content-Type": "application/json"});

    /*
        request.on waiting for data to arrive.
    */
    request.on('data', function (chunk)
    {

        /*
            CHUNK which we recive from the clients
            For out request we are assuming its going to be a JSON data.
            We print it here on the console.
        */
        console.log(chunk.toString('utf8'))

        /*
            Using kafka-node - really nice library
            create a producer and connect to a Zookeeper to send the payloads.
        */
        var kafka = require('kafka-node'),
        Producer = kafka.Producer,
        client = new kafka.Client('kafka:2181'),
        producer = new Producer(client);

        /*
            Creating a payload, which takes below information
            'topic'     --> this is the topic we have created in kafka.
            'messages'  --> data which needs to be sent to kafka. (JSON in our case)
            'partition' --> which partition should we send the request to.
                          If there are multiple partition, then we optimize the code here,
                           so that we send request to different partitions.

        */
            payloads = [
            { topic: 'test', messages: chunk.toString('utf8'), partition: 0 },
        ];

        /*
            producer 'on' ready to send payload to kafka.
        */
        producer.on('ready', function(){
            producer.send(payloads, function(err, data){
                    console.log(data)
            });
        });

        /*
            if we have some error.
        */
        producer.on('error', function(err){})

    });
    /*
        end of request
    */
    response.end();

/*
    Listen on port 8125
*/
}).listen(8125);
```

Step 2 - Start above script on the `nodejs` server.

```
[nodejs-admin@nodejs nodejs]$ vim json_nodejs_kafka.js
[nodejs-admin@nodejs nodejs]$ node json_nodejs_kafka.js
```

Step 3 - Execute `curl` command to send the JSON to `nodejs`.

```
[nodejs-admin@nodejs nodejs]$ curl -H "Content-Type: application/json" \
                     -d '{"username":"xyz","password":"xyz"}' http://localhost:8125/upload
```

Step 4 - Output on nodejs console

```
[nodejs-admin@nodejs nodejs]$ node json_nodejs_kafka.js
{"username":"xyz","password":"xyz"}
{ test: { '0': 29 } }
```

`{"username":"xyz","password":"xyz"}` request from the `curl` command. `{ test: { '0': 29 } }` response from the kafka cluster that, it has received the `json`.

Step 5 - Output on the `kafka` consumer side.

```
[kafka-admin@kafka kafka_2.9.2-0.8.2.0]$ bin/kafka-console-consumer.sh \
                                   --zookeeper localhost:2181 --topic test --from-beginning
{"username":"xyz","password":"xyz"}
```

`{"username":"xyz","password":"xyz"}` data received from `nodejs` server.

Useful Links.

```
http://kafka.apache.org/documentation.html
http://nodejs.org/
http://nodejs.org/api/http.html
https://www.npmjs.com/package/kafka-node
https://cwiki.apache.org/confluence/display/KAFKA/Index
https://github.com/pelger/Kafkaesque
https://github.com/joyent/node/wiki/installing-node.js-via-package-manager#enterprise-linux-and-fedora
http://stackoverflow.com/questions/7172784/how-to-post-json-data-with-curl-from-terminal-commandline-to-
```