



# Dev Workshop | IoT APIs 101 with StrongLoop and IBM Bluemix

API Developer Lab | Miracle Innovation Labs

**Chanakya Lokam**

Director – Marketing and Innovation  
Miracle Software Systems, Inc.

**February 14, 2016**

# Dev Workshop | IoT APIs 101 with StrongLoop and IBM Bluemix

## Goal

In this lab the users will create a REST API for the IoT Data that is being stored in Cloudant. We will be using StrongLoop to create the REST API using the Loopback Framework and String Loop CLI Tools.

## Pre-Requisites

The following are required to complete this lab,

- Web Browser for accessing Cloudant and Cloud9
- SOAP UI 5.0 (or) Curl for testing the API
- Existing Cloudant Account
- Completion of Part 1 of this lab(IoT Data Stored in Cloudant DB)

## Technology Involved

Strong Loop  
Node.js  
REST/SOAP  
Internet of Things  
Cloudant and NoSQL

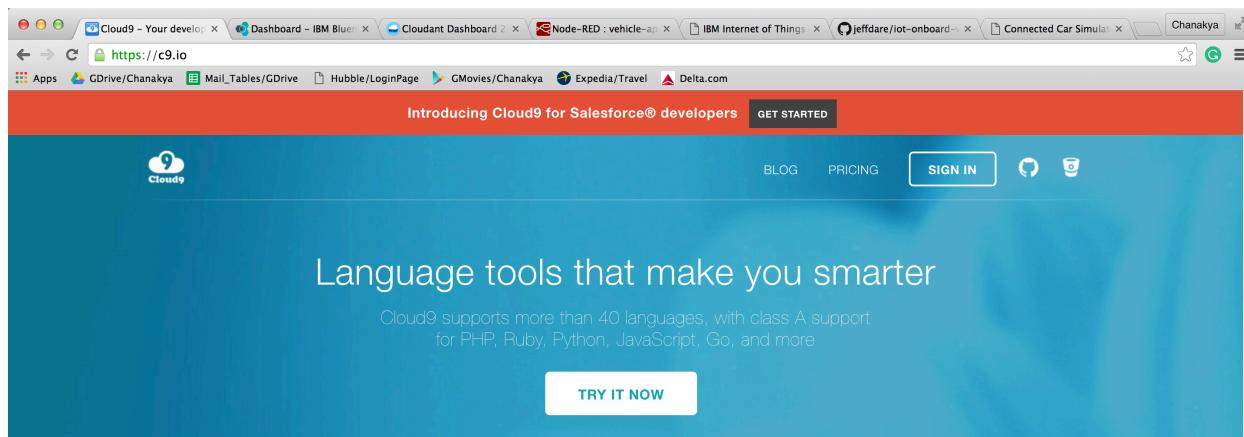
## Lab Steps

So, let us get started with the lab!

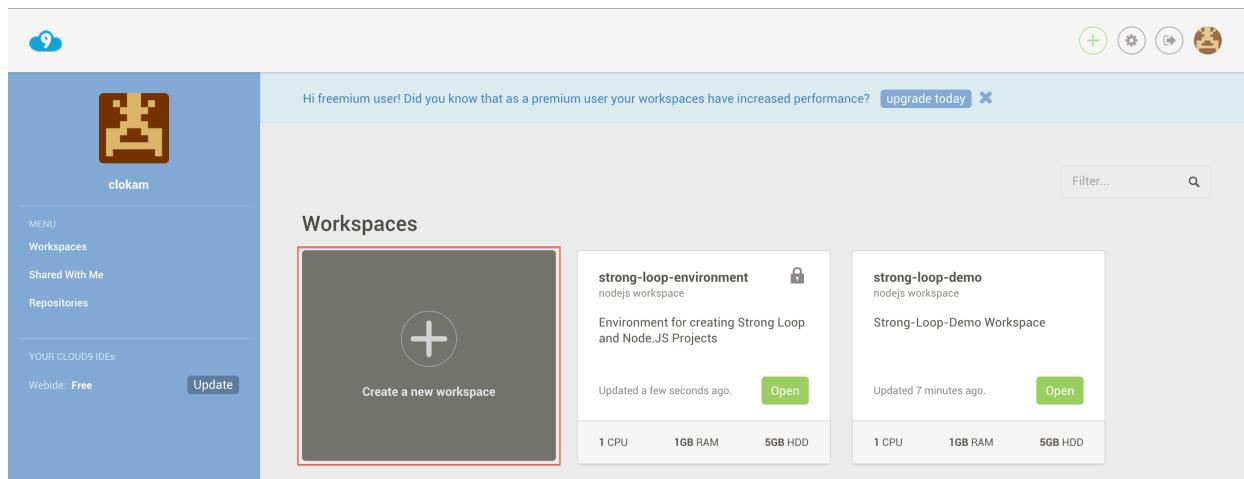
### #1 | Get Access to Cloud9

We will be installing and using StrongLoop with Cloud9 which is an online IDE for developers. You are also free to install StrongLoop locally and follow the same steps with slight alterations.

Go to <http://c9.io> and sign in if you already have any account (or) sign up if you are a new user.



Once you are signed in click on the “Create a new workspace button”.



Create the new workspace with the workspace name(For example, vehicle-api-env) and select the **node.js template** in the “Choose a Template” section.

Owner: clokam      Workspace name: vehicle-api-env

Description: Demo Strongloop environment for creating Vehicle Data Simulator REST API

Hosted workspace    Remote SSH Workspace

**Private** This is a workspace for your eyes only     **Public** This will create a workspace for everybody to see

Clone from Git or Mercurial URL (optional)  
e.g. ajaxorg/ace or git@github.com:ajaxorg/ace.git

Choose a template:

Custom
HTML5
Node.js
Meteor
PHP, Apache &...
Python

## #2 | Installing StrongLoop

Once you have created the workspace you will be directly taken to the new workspace screen and will have your cursor placed within a new terminal.

In the terminal type the following command to install Strong Loop. The installation may take a few moments.

**npm install -g strongloop**

After the installation is complete you should be able to see the following screen within your terminal.

```

bash - clokam-vehic... Immediate +1
└─ lodash@3.10.1
  └─ strong-mesh-models@1.0.0 (user-home@1.1.1, sprint@0.1.5, posix getopt@1.2.0, strong-url-defaults@1.2.1, text-table@0.2.0, serve-favicon@2.3.0, concat-stream@1.5.1, errorhandler@0.4.3, compression@1.6.1, loopback-boot@2.16.0, strong-tunnel@1.1.4, request@2.69.0, strong-npm-ls@1.0.7, http-auth@2.2.8, loopback-explore@1.8.0)
    └─ strong-deploy@0.1.2 (posix-getopt@1.2.0, strong-url-defaults@1.2.1, concat-stream@1.5.1, shells@0.3.0, strong-tunnel@1.1.4, strong-mesh-models@7.1.1)
      └─ less@1.4.2 (mime@1.2.11, mkdirp@0.3.5, ycssmin@1.0.1, request@2.69.0)
        └─ node-inspector@0.7.4 (debug@0.8.1, which@0.9.0, opener@1.5.0, async@0.8.0, strong-data-uri@0.1.1, rc@0.3.5, yargs@1.2.6, glob@3.2.11, ws@0.4.32, express@4.0.0)
          └─ generator-loopback@0.1.5.1 (inflection@0.8.0, async@0.5.2, loopback-api-definition@1.0.0, chalk@1.1.1, mkdirp@0.5.1, yosay@1.1.0, loopback-swagger@2.3.0, js-yaml@3.5.3, request@2.69.0, yeoman-generator@0.21.1, loopback-workspace@3.19.0)
            └─ strong-remoting@0.25.1 (eventemitter2@0.4.14, inflection@1.8.0, jsxmlparser@0.1.9, depd@1.1.0, qs@2.4.2, loopback-phase@1.3.0, sse@0.0.6, traverse@0.6.6, cors@0.7.1, body-parser@1.15.0, express@4.13.4, mux-demux@3.7.9, request@2.69.0, jayson@1.2.2, xmlhttp@0.4.16)
              └─ sqlite@3.1.1 (nan@2.1.0)
                └─ strong-arc@1.8.0 (parse-ms@0.1.0, opener@1.4.1, path@0.11.14, strong-trace-waterfall@0.1.3, minimist@1.2.0, cookie-parser@1.4.1, serve-favicon@2.3.0, sha1@1.1.1, split@0.3.3, http-proxy@0.13.1, node-underscoreify@0.0.14, cxvizi-color@0.0.1, compression@0.6.1, errorhandler@1.4.3, page@0.16.4, acorn@0.2.7.0, express@4.13.4, strong-trace-waterfall@0.1.1, strong-arc@1.8.1, strong-arc@filesystem@0.1.0, detective@0.4.3.1, loopback-boot@2.16.0, fs-extra@0.16.5, loopback-component-expo@0.2.3.0, request@2.69.0, cxvizi-formate@0.0.1, cxvizi-flame@0.0.1, cxvizi-rawtree@0.1.0, cxvizi-eventloop@0.1.0, loopback-workspace@3.19.0, cxvizi-timeseries@0.2.2, strong-trace-waterfall@0.1.0, less@2.6.0, strong-mesh-client@1.6.3)
      clokam:~/workspace $ ]

```

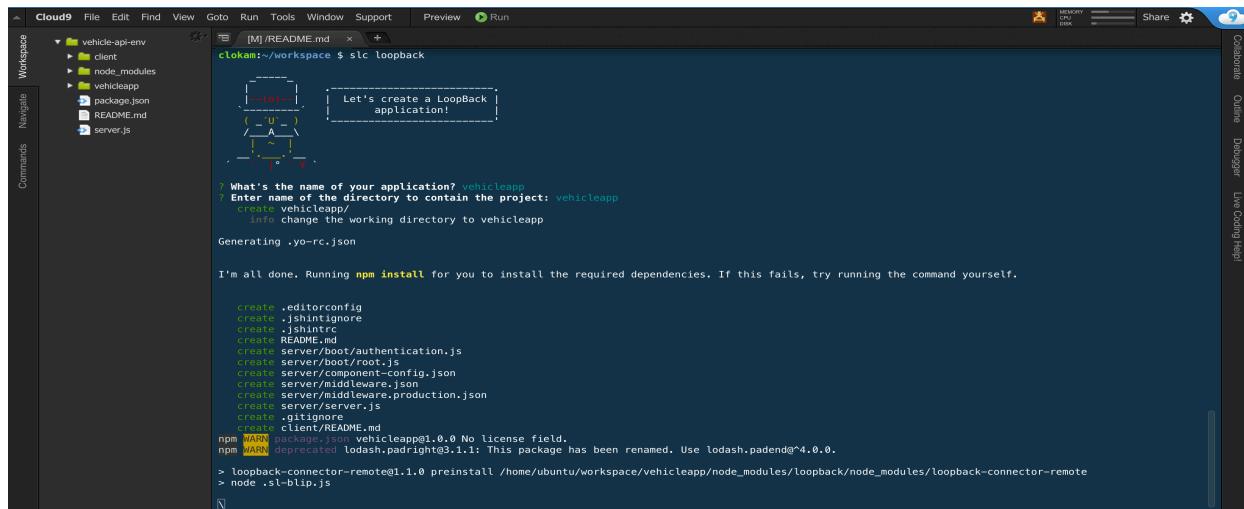
**Note :** The installation process is dependent on the network and is likely to take between 3-5 minutes to complete

## #3 | Create Loopback Application and Model

Once StrongLoop is installed we will now need to create the Application, Data Source and the Application Model. This will define how the REST API will be shaped out and what it will connect to.

Use the **slc loopback** command to create the new application. When prompted give the **Application Name**(For example vehicleapp) and the **Application Directory**(For example vehicleapp). This is based on Yeoman and scaffolds your entire application structure to ensure that you can get things running quickly!

**NPM** will ensure that all the dependencies of the application are installed along with the entire application structure.



```

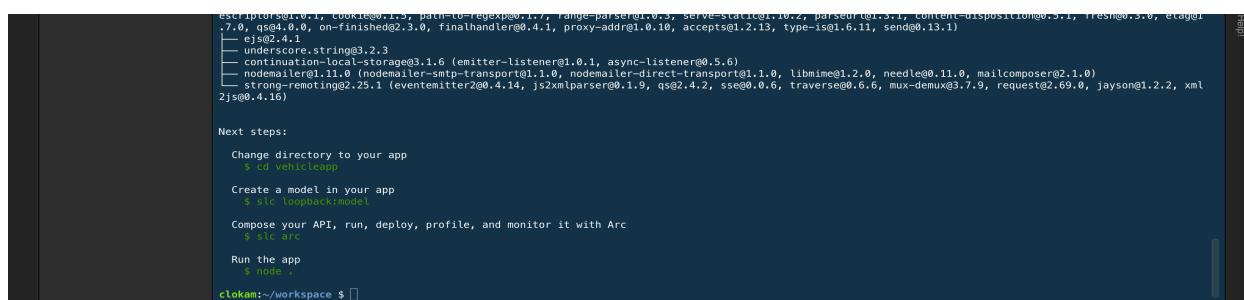
Cloud9 File Edit Find View Goto Run Tools Window Support Preview Run
Workspace Navigate Commands Cloud9 Online Debugging Help
[ M ] README.md
clokam:~/workspace $ slc loopback
Let's create a LoopBack application!
? What's the name of your application? vehicleapp
? Enter name of the directory to contain the project: vehicleapp
  - create vehicleapp/
    - info change the working directory to vehicleapp
Generating .yo-rc.json

I'm all done. Running npm install for you to install the required dependencies. If this fails, try running the command yourself.

  create .editorconfig
  create .jshintrc
  create .shrc
  create README.md
  create server/boot/authentication.js
  create server/boot/root.js
  create server/component-config.json
  create server/config.js
  create server/middleware/production.json
  create server/server.js
  create .gitignore
  - create client/README.md
npm WARN package.json vehicleapp@1.0.0 No license field.
npm WARN deprecated lodash.padright@3.1.1: This package has been renamed. Use lodash.padend@^4.0.0.
> loopback-connector-remote@1.1.0 preinstall /home/ubuntu/workspace/vehicleapp/node_modules/loopback/node_modules/loopback-connector-remote
> node .sl-blip.js

```

Once the installation is complete you should see the next steps as shown below.



```

  esprima@0.7.1, cookie@1.1.3, path@0.12.7, range-parser@0.9.3, serve-static@1.0.2, qs@2.3.1, content-disposition@0.3.1, tessel@0.3.0, etag@1
  .7.0, qs@4.0.0, on-finished@2.3.0, finalhandler@0.4.1, proxy-addr@1.0.10, accepts@1.2.13, type-is@1.6.11, send@0.13.1)
  ejσ@2.4.1
  underscore.string@3.2.3
  continuation-local-storage@3.1.6 (emitter-listener@1.0.1, async-listener@0.5.6)
  nodemailer@1.11.0 (nodemailer-smtp-transport@1.1.0, nodemailer-direct-transport@1.1.0, libmime@1.2.0, needle@0.11.0, mailcomposer@0.2.1.0)
  strong-decoding@2.25.0.1 (eventemitter2@0.4.14, jsxmlparser@0.1.9, qs@2.4.2, sse@0.0.6, traverse@0.6.6, mux-demux@3.7.9, request@2.69.0, jayson@1.2.2, xml
  2js@0.4.16)

Next steps:
  Change directory to your app
  cd vehicleapp

  Create a model in your app
  $ slc loopback:model

  Compose your API, run, deploy, profile, and monitor it with Arc
  $ slc arc

  Run the app
  $ node .

clokam:~/workspace $ 

```

Now we must configure the Data Source and install the connector for using Cloudant with our StrongLoop API.

Navigate to the `<directory-name>/server/datasources.json` file using the left side workspace explorer menu. Remove the existing code and paste the below code.

```
{
  "db": {
    "name": "db",
    "connector": "memory"
  },
  "cloudant": {
    "database": "vehicle-data",
    "username": "fe26cdf0-bd95-485b-aa5e-9862770eb319-bluemix",
    "password": "0cbdea29b0c41a6603ccbee9f5901d278e60ddfb25c3923f1f10103f285f11b6",
    "name": "cloudant",
    "connector": "cloudant"
  }
}
```

For getting the **Cloudant User Name** and **Password** you will have to go back to your Bluemix Application and click on the **Environment Variables** option. Here you can copy and paste the credentials.

The screenshot shows the IBM Bluemix environment variables interface. On the left, there's a sidebar for 'Vehicle-Application' with 'Environment Variables' selected. The main area is titled 'Environment Variables' and has tabs for 'VCAP\_SERVICES' (which is active) and 'USER-DEFINED'. A code editor window displays the following JSON:

```
{
  "cloudantNoSQLDB": [
    {
      "name": "Vehicle-Application-cloudantNoSQLDB",
      "label": "cloudantNoSQLDB",
      "plan": "Shared",
      "credentials": {
        "username": "fe26cdf0-bd95-485b-aa5e-9862770eb319-bluemix",
        "password": "0cbdea29b0c41a6603ccbee9f5901d278e60ddfb25c3923f1f10103f285f11b6",
        "port": 443,
        "url": "https://fe26cdf0-bd95-485b-aa5e-9862770eb319-bluemix:0cbdea29b0c41a6603ccbee9f5901d278e60ddfb25c3923f1f10103f285f11b6@fe26cdf0-bd95-485b-aa5e-9862770eb319-bluemix.cloudant.com"
      }
    },
    "iotf-service": []
  ]
}
```

A red box highlights the 'password' field in the first credential object.

Save the datasources.json file.

```

1  {
2    "db": {
3      "name": "db",
4      "connector": "memory"
5    },
6    "cloudant": {
7      "database": "vehicle-data",
8      "url": "http://Fe26cdf0-bd95-485b-a5e-9862770eb319-bluemix",
9      "username": "Fe26cdf0-bd95-485b-a5e-9862770eb319-bluemix",
10     "password": "0cbdea29b0c41a6603ccbee9f5901d278e60ddfb25c3923f1f10103f285f11b6",
11     "name": "cloudant",
12     "connector": "cloudant"
13   }
14 }
15

```

Now go back to the terminal, navigate to the application directory by using the command **cd <directory-name>**. Then execute the following command to install the Cloudant DB Connector.

**npm install loopback-connector-cloudant**

```

bash -clokam-vehic x Immediate x + 
Next steps:
Change directory to your app
$ cd vehicleapp
Create a model in your app
$ slc looback:model
Compose your API, run, deploy, profile, and monitor it with Arc
$ slc arc
Run the app
$ node .
clokam:~/workspace $ cd vehicleapp/
clokam:~/workspace/vehicleapp $ npm install loopback-connector-cloudant
npm WARN package.json vehicleapp@1.0.0 No license field.
npm WARN engine follow@0.12.1: wanted: {"node":"0.12.x || 0.10.x || 0.8.x"} (current: {"node":"4.1.1","npm":"2.14.4"})
loopback-connector-cloudant@1.0.4 node_modules/loopback-connector-cloudant
├── async@1.5.2
├── loopback-connector@2.3.0
├── debug@2.2.0 (ms@0.7.1)
└── lodash@3.10.1
clokam:~/workspace/vehicleapp $ 

```

The next step is to create the data model within the application. Stay in the same directory and use the following command.

**slc looback:model**

When prompted give the **model name**(For example **vehicledata**). Select the data-source as **cloudant** and the model's base class as **PersistedModel** using the arrows keys.

When asked “Expose **vehicledata** via REST API?” answer with “**Y**”. Click enter when asked about the “Custom Plural Form”. Select Common Model when asked and then we can go about creating the property names.

```

node -clokam-vehic x Immediate x + node -clokam-vehic x Immediate x + node -clokam-vehic x Immediate x +
Run the app
$ node .

clokam:~/workspace $ cd vehicleapp/
clokam:~/workspace/vehicleapp $ npm install loopback-connector-cloudant
npm WARN package.json vehicleapp@1.0.0 No license field.
npm WARN engine follow@0.12.1: wanted: {"node":"0.12.x || 0.10.x || 0.8.x"} (current: {"node":"4.1.1","npm":"2.14.4"})
loopback-connector-cloudant@1.0.4 node_modules/loopback-connector-cloudant
└── async@1.5.2
  ├── loopback-connector@2.3.0
  └── dtrace-provider@0.7.1
    └── led@v0.1.3
      └── Cloudant@0.4.1 (nano@6.2.0)

clokam:~/workspace/vehicleapp $ slc loopback:model
? Enter the model name: vehicledata
? Select the data-source to attach vehicledata to: cloudant (cloudant)
? Select model's base class PersistedModel
? Expose vehicledata via the REST API? Yes
? Custom plural form (used to build REST URL):
? Common model or server only? common
Let's add some vehicledata properties now.

Enter an empty property name when done.
? Property name: fuel
? Property type: string
? Required? Yes

Let's add another vehicledata property.
Enter an empty property name when done.
? Property name: lat
? Property type: string
? Required? Yes

Let's add another vehicledata property.
Enter an empty property name when done.
? Property name: long
? Property type: string
? Required? Yes

Let's add another vehicledata property.
Enter an empty property name when done.
? Property name:
clokam:~/workspace/vehicleapp $

```

Give the first property name as “**fuel**” and select it as **String Type**. Also add two more **String** properties “**lat**” and “**long**”. Select required as “**y**” for all 3 of the properties. Once done click on Enter to finish the Loopback Model creation.

```

node -clokam-vehic x Immediate x + node -clokam-vehic x Immediate x + node -clokam-vehic x Immediate x +
Run the app
$ node .

clokam:~/workspace $ cd vehicleapp/
clokam:~/workspace/vehicleapp $ slc loopback:model
? Enter the model name: vehicledata
? Select the data-source to attach vehicledata to: cloudant (cloudant)
? Select model's base class PersistedModel
? Expose vehicledata via the REST API? Yes
? Custom plural form (used to build REST URL):
? Common model or server only? common
Let's add some vehicledata properties now.

Enter an empty property name when done.
? Property name: fuel
? Property type: string
? Required? Yes

Let's add another vehicledata property.
Enter an empty property name when done.
? Property name: lat
? Property type: string
? Required? Yes

Let's add another vehicledata property.
Enter an empty property name when done.
? Property name: long
? Property type: string
? Required? Yes

Let's add another vehicledata property.
Enter an empty property name when done.
? Property name:
clokam:~/workspace/vehicleapp $

```

## #4 | Editing the Node Red Flow

For the looback model to be able to gather data from the Cloudant DB we have to make sure that we are storing data from Node Red in the same format.

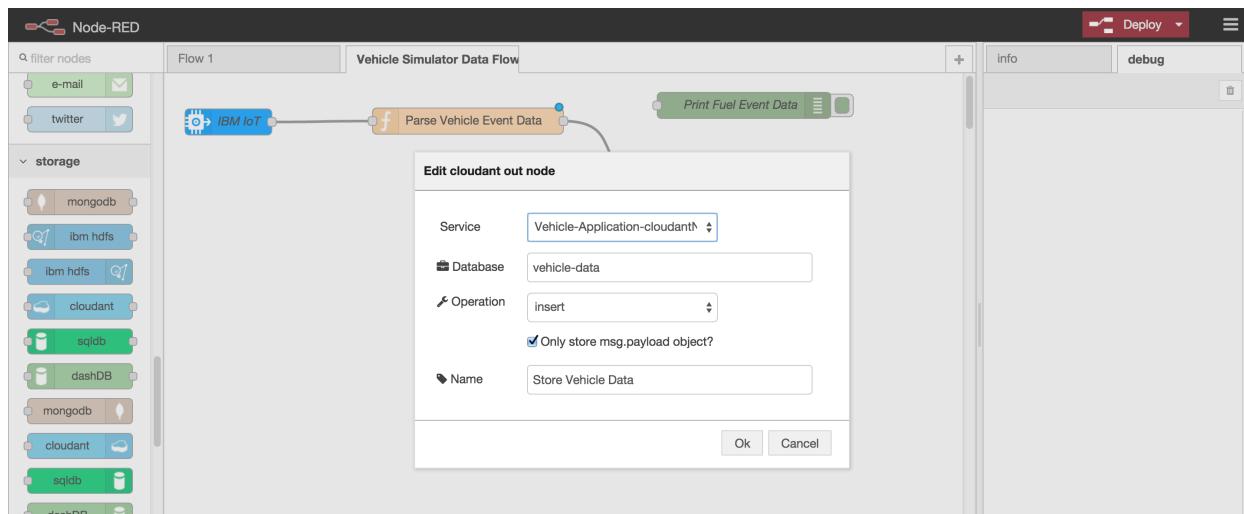
For this we must edit our Node Red Flow. Go back to Node Red and change the code within our “Parse Vehicle Event Data” function node to show the below.

```

msg.fuel = msg.payload.d.fuel
msg.lat = msg.payload.d.lat
msg.long = msg.payload.d.long
msg.loopback_model_name = "<Your Loopback Model Name>"
msg.payload = {"fuel":msg.payload.d.fuel,"loopback_model_name":"vehicle-
model"}
return msg;

```

Also go to the Cloudant node and check the “**Only store msg.payload object?**” as well to ensure that we are stroing only our msg.payload object in to Cloudant.



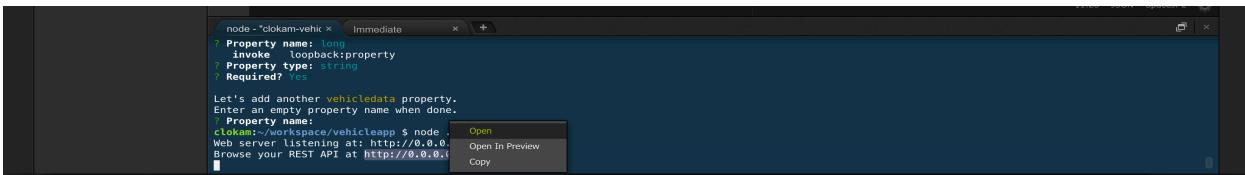
Redeploy the Node Red flow, go back to the Vehicle Simulator and send some more Fuel Events to see if the data gets stored in the Cloudant DB.

## #5 | Testing the StrongLoop API

Now that we have everything set, go back to the Cloud9 Terminal within the application directory and run the following command.

**node .**

This will start the **explorer** on **localhost(0.0.0.0)** on port **8080**. Click on the link in the terminal and select “Open” to open the explore in a new tab.



In the explorer, click on your loopback model and then on the GET Method to open up the following screen,

**StrongLoop API Explorer**

Token Not Set accessToken Set Access Token

User

vehicledata

GET /vehicledata

Find all instances of the model matched by filter from the data source.

Response Class (Status 200)

Model Model Schema

```
[
  {
    "fuel": "string",
    "lat": "string",
    "long": "string",
    "id": "string"
  }
]
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
filter		Filter defining fields, where, include, order, offset, and limit	query	string

Try it out!

Click on “Try it out!” and you should be able to see the data coming in from the API with the Vehicle Data.

**StrongLoop API Explorer**

Token Not Set accessToken Set Access Token

Curl

```
curl -X GET --header "Accept: application/json" "http://vehicle-api-env-clokam.c9users.io:8080/api/vehicledata"
```

Request URL

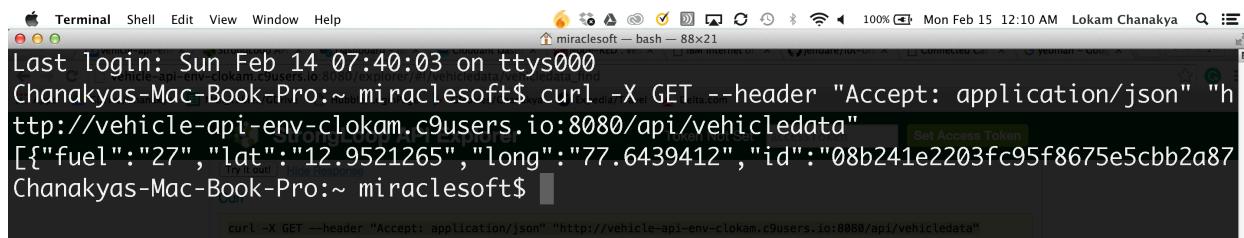
```
http://vehicle-api-env-clokam.c9users.io:8080/api/vehicledata
```

Response Body

```
[
  {
    "fuel": "27",
    "lat": "12.9521265",
    "long": "77.6439412",
    "id": "08b241e2203fc95f8675e5cbb2a87cab"
  },
  {
    "fuel": "234",
    "lat": "12.9521265",
    "long": "77.6439412",
    "id": "ec0963d48bb37fc43dc70e9d5e0ea95e"
  },
  {
    "fuel": "13",
    "lat": "12.9521265",
    "long": "77.6439412",
    "id": "4be7319be5a4cd5a8e8747a467c9e3de"
  }
]
```

You can also try the same using the **CURL** Option given in the explorer (or) try and send the request using an API testing tool such as **SOAP UI** from SmartBear.

Below is a screenshot of a CURL Test from a local machine terminal.



A screenshot of a Mac OS X Terminal window. The window title is "miraclesoft — bash — 88x21". The terminal shows the following command and its output:

```
Last login: Sun Feb 14 07:40:03 on ttys000
Chanakyas-Mac-Book-Pro:~ miraclesoft$ curl -X GET --header "Accept: application/json" "http://vehicle-api-env-clokam.c9users.io:8080/api/vehicledata"
[{"fuel":"27","lat":"12.9521265","long":"77.6439412","id":"08b241e2203fc95f8675e5ccb2a87
Chanakyas-Mac-Book-Pro:~ miraclesoft$
```

Now that you have your devices connected, data stored and api exposed you are ready to tackle the world of IoT. Also try out managing your APIs so that you can control which operations you want to expose, who gets access and how much privileges they can extend on the data.