OCEAN

&Cube:Rosemary Release 2

Software Version: 2.0.0

Installation Guide v2.0.0

Document Release Date: July 2017

Software Release Date: July 2017

OCEAN

Copyright and Disclaimer of Liability

This document may contain technical inaccuracy or type errors, and the author does not have any responsibility on this matter.

The contents of this document can be changed or added regularly, and the relevant corrected version will be added to the document under the title named “New Edition” in consecutive order. The product or program mentioned in this document may be changed or modified without any prior notice.

The source code of &Cube:Rosemary is distributed according to the license policy below.

- The open source code shared by OCEAN (Open allianCE for iot stANdard) is distributed based on the 3-clause BSD-style license. While maintaining copyright header in the source code file, the open source code can be used freely in the purpose of commercial or non-commercial systems.
- License of OCEAN does not force users to share the developed source code with others. The ownership of the developed source code belongs to the developer and (s)he has no obligation to share it.
- Anyone can contribute to improvement of the open source environment of OCEAN. If so, the developed source code should follow the license policy of OCEAN.

Copyright (c) 2017, OCEAN
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

1. &Cube:Rosemary

1.1. Introduction

The &Cube:Rosemary is the open source IoT gateway platform based on the oneM2M (http://www.oneM2M.org) standard. As one of the oneM2M platforms, &Cube-Rosemary provides common services functions to oneM2M applications and other oneM2M devices. So the &Cube-Rosemary can be used to provide proximity based IoT services. As defined in the specifications, &Cube-Rosemary as MN-CSE also provides interworking functionalities via IPE (Interworking Proxy Entity).

The &Cube:Rosemary implements CSE which acts like a server so due to the similarities defined in the oneM2M specifications, it shares the source codes with Mobius. As allowed in supported configurations in oneM2M Functional Architecture TS (TS-0001), &Cube:Rosemary also has registration procedures to other CSEs like IN-CSE (i.e. Mobius). Except this, currently &Cube:Rosemary works as the same as Mobius. For the installation, &Cube:Rosemary does not require MQTT broker installation since it uses the broker of Mobius.

The following figure shows the deployment configuration using &Cube:Rosemary. It can locally host oneM2M resources within the MN-CSE so it provides REST APIs to other devices/applications.

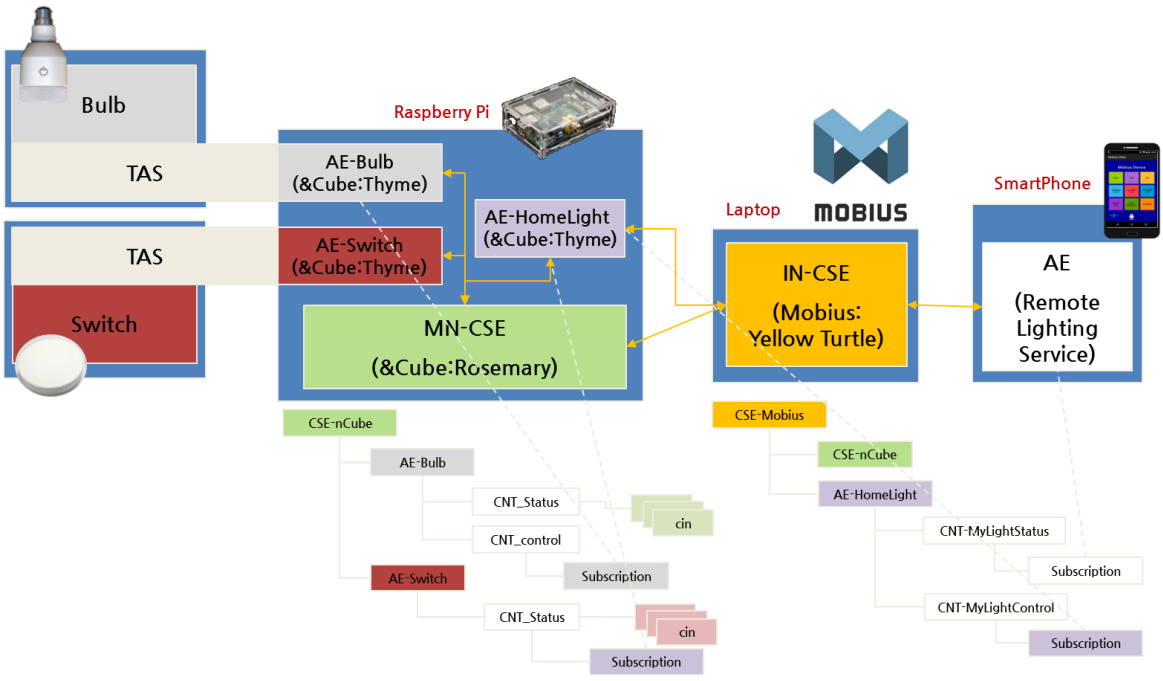

Figure 1 IoT system configuration using &Cube:Rosemary

1.2. &Cube:Rosemary Source Code Directory

The figure below shows the composition of the &Cube:Rosemary source codes. It consists with mobius folder and rosemary.js, app.js, pxy_coap, pxy_ws, pxy_mqtt.js, package.json files. The mobius folder is the same as the Mobius server platform. Other sources codes are generic modules like db_action.js, resource.js, responder.js, security.js, sgn.js, sql_action.js, ts_agent.js and oneM2M resource type specific ones.
For more details, please refer to the Table 1.

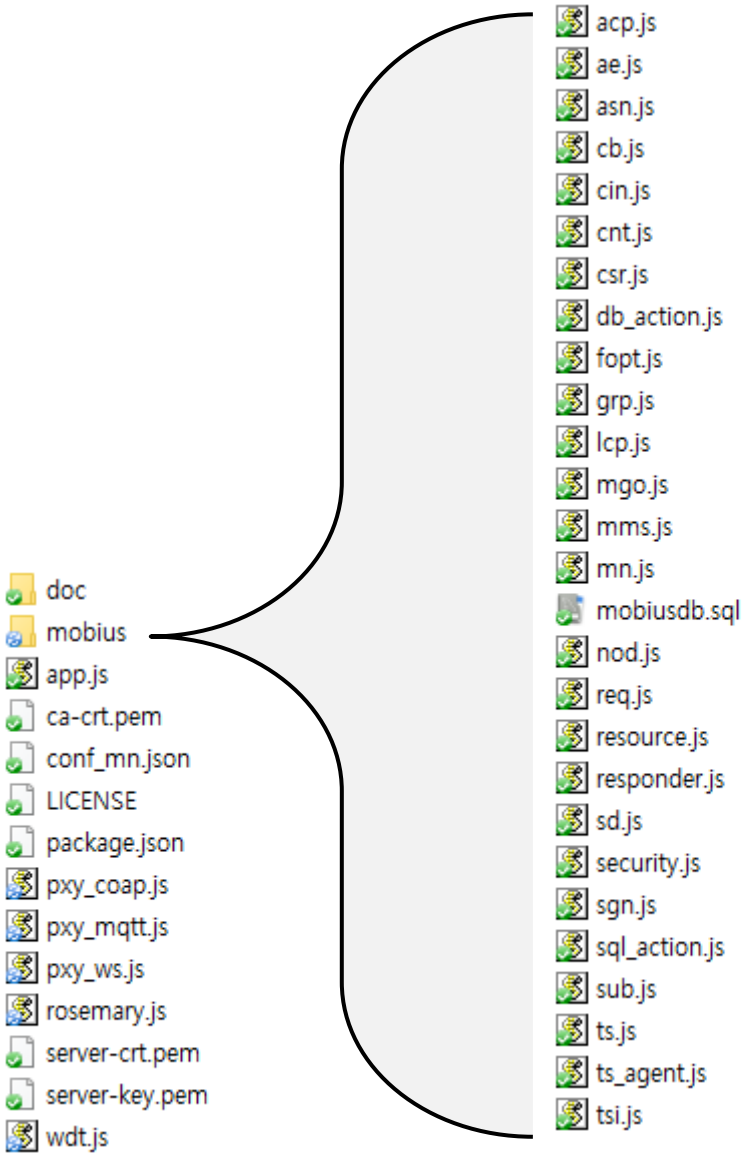

Figure 2 &Cube:Rosemary source code directory

Table 1 Function Reference Table for Node JS Files

Source File	Role and Function
rosemary.js	It configures settings of &Cube:Rosemary and also is used to run &Cube:Rosemary.
app.js	This file acts as role of flow router and it is the main code running Mobius server.

	① It handles initial processing of received packets. ② It initiates HTTP server with 'listening' mode to wait for HTTP requests target to the Mobius HTTP server. ③ It handles the parsing of URL of packets and evaluate the correctness of the request body resulted of parsing. It then sends the request to resource.js to continue the processing if the request is valid one, otherwise throws exceptions. This file also implements the server clustering algorithms to improve the performance of HTTP server.
mobius/resource.js	It is core file to process the CREATE, RETRIEVE, UPDATE, DELETE, NOTIFY and DISCOVERY operations for oneM2M resource primitives. This file undertakes the processing of parsed request URI and request body received from app.js according to corresponding operation. It converts the data into a format to process the data and connect to mysql database. The mysql database is initialized and handled by db_action.js and sql_action.js module.
mobius/responder.js	It is responsible for handling the response process. It receives processing results from app.js and resource.js modules and generates responses from the processing results following format requested by originator either XML or JSON serialization.
mobius/db_action.js	This file contains parameters used to connect and access to the database and parameters for returning response results from the database.
mobius/sql_action.js	This file contains functions to receive data and parameters required for a series of database operations and functions to call db_action.js module to return data from database.
mobius/sgn.js	This module file is responsible for checking the existence of <subscription> child resource under the requested target resource. If it exists, the module checks the event type and retrieves the field value of attribute *notificationUril.* Then it generates and sends a notification message to the address indicated by field value of attribute *notificationUril.*
mobius/security.js	This file contains functions to check the access privileges of originators for a requested resource when the resource applies with <*accessControlPolicy*> resource. The originator ID is abstracted from request header field of X-M2M-Origin. The process checks the access right of current originator ID to a target resource and responds with ACESS_DENIED error when the originator ID has no privileges to access to the resource.
mobius/fopt.js	This file contains function to handle operations target to <*fanOutPoint*> virtual resource of a <*group*> resource. It transmits operation request to all group members contained in <group> resource and aggregates responses from all group members into an aggregated response.
mobius/ts_agent.js	This file contains functions to manage <*timeSeriesInstance*> resource. It monitors the mssing data in the <*timeSeriesInstance*> resource and then stores the missing data.
pxymqtt.js	This file contains functions implementing mqtt proxying function to handle mqtt protocol messages with http protocol module as following procedures:

	① It creates a oneM2M mqtt topic with configuration information of Mobius server and then subscribe to the topic to receive mqtt requests targeted to this topic later; ② Whenever receiving mqtt protocol messages, it generates and sends a http request wrapping mqtt request message to Mobius server and waits for http response from Mobius server. It then abstracts http response and generates a mqtt response message correspondingly. ③ It responds with mqtt protocol message.
pxy_coap.js	This file contains functions implementing coap proxying function to handle coap protocol messages with http protocol module.
pxy_ws.js	This file contains functions implementing websocket proxying function to handle websocket protocol messages with http protocol module.
mobius/ae.js	It contains functions to parse AE request
mobius/cb.js	It contains functions to parse CSEBase request. The CSEBase resource contains configuration information of Mobius server
mobius/cin.js	It contains functions to parse contentInstance request
mobius/cnt.js	It contains functions to parse container request
mobius/csr.js	It contains functions to parse remoteCSE request
mobius/grp.js	It contains functions to parse group request
mobius/lcp.js	It contains functions to parse locationPolicy request
mobius/mms.js	It contains functions to parse multimediaSession request
mobius/mn.js	It contains functions to register this CSE to the other CSE.
mobius/sd.js	It contains functions to parse semanticDescriptor request
mobius/sub.js	It contains functions to parse subscription request.
mobius/ts.js	It contains functions to parse timeSeries request.
mobius/tsi.js	It contains functions to parse timeSeriesInstance request.

2. &Cube:Rosemary Installation

2.1. Introduction

Installation procedures of &Cube:Rosemary is basically the same as Mobius, starting with MySQL installation. Unlike Mobius, &Cube:Rosemary does not require MQTT broker installation since it uses the broker of Mobius.

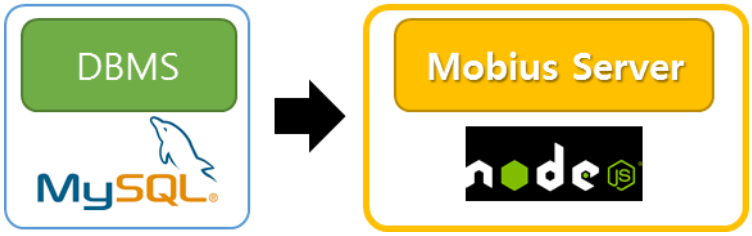

Figure 2 &Cube:Rosemary installation procedures

2.2. Pre-requisites Installation (Windows)

2.2.1. MySQL Installation

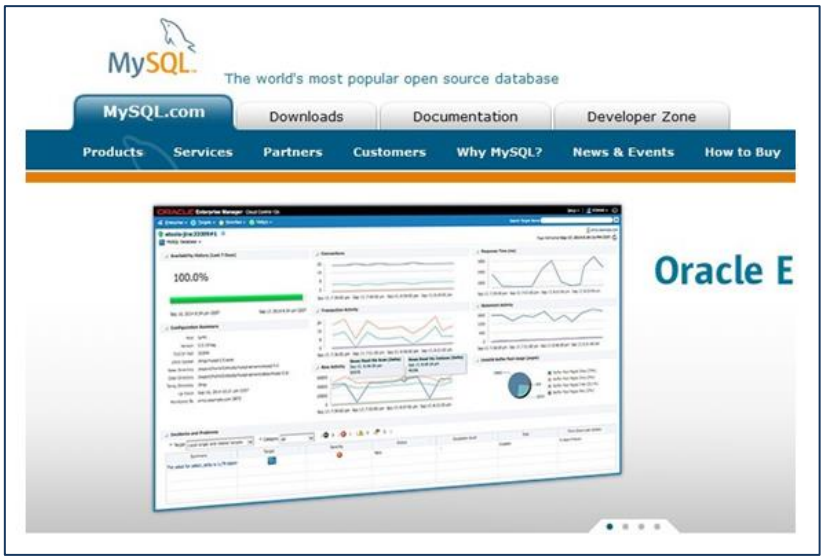

Figure 3 MySQL Introduction

The &Cube:Rosemary stores the data uploaded from devices into MySQL database. MySQL is an open source RDBMS and the installation procedures are introduced as following:

Figure 4 MySQL community server

Different versions of MySQL are provided for download at below link:

http://dev.mysql.com/downloads/mysql.

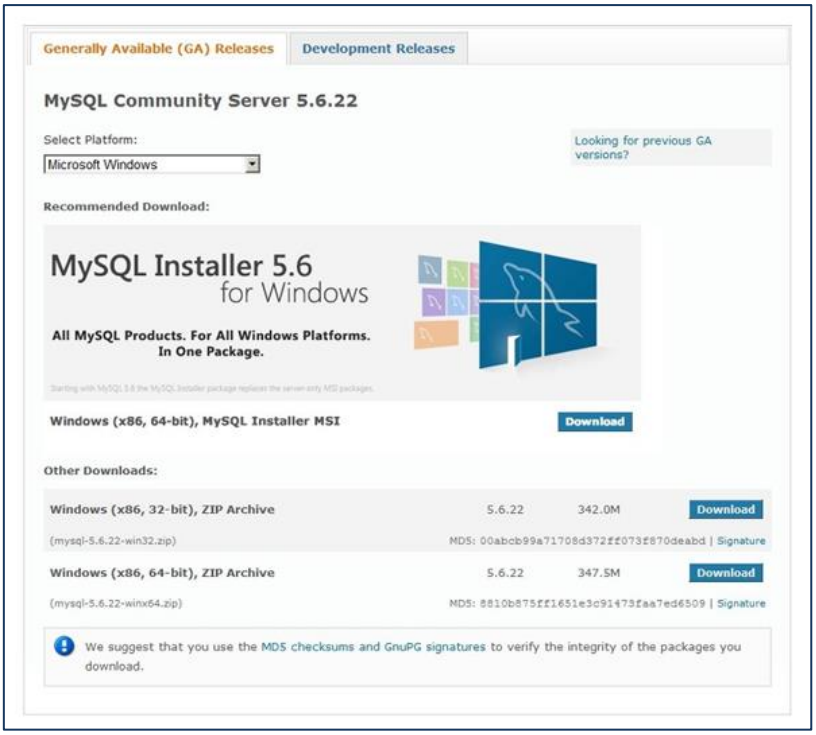

Figure 5 Download MySQL Installer

There is a MySQL community server version for free downloading. Users could also select to download MySQL Enterprise Edition or MySQL Cluster CGE version which need to buy licenses before starting use. User can choose to install a light MSI installer which installs all required MySQL packages online or a standalone version that contains all required packages. If users prefer to install using zip archive, you can download and install as shown below.

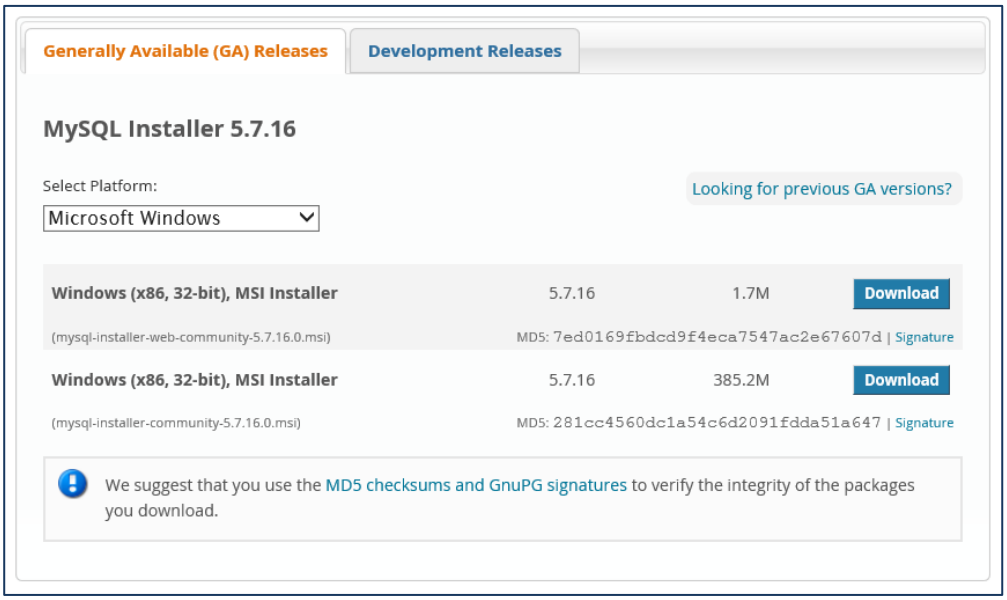

Figure 6 Select MySQL Installer for downloading

Here we demonstrate downloading the free version MySQL community server and you can find it from the top 'Downloads' menu tab and we select to install a Windows 64bit version MySQL through MySQL installer as shown as Figure .

The MySQL can be easily installed following the procedures as depicted as Figure .

① Read and accept the license agreement, then

② Choose a Setup type: A list of setup types "Developer Default, Server only, Client only, Full or Custom" are available to install. Here we recommend to select "Custom" and then select only two packages "MySQL Server" and "MySQL Workbench" to install.

③ Install automatically the selected packages and after successful installation, then

④ Configure the MySQL runtime environment shown as Figure .

i. Configure the machine type to either 'Development Machine', 'Server Machine', or 'Dedicated Machine'. Here we select option 'Development Machine' to configure.

ii. Set 'root' account passwords and optionally create users with different privileges to access to the MySQL.

iii. Configure MySQL server as a Window Service and finally apply the user configurations to MySQL server.

iv. Input the root password and check the connection status to the configured MySQL as shown as Figure

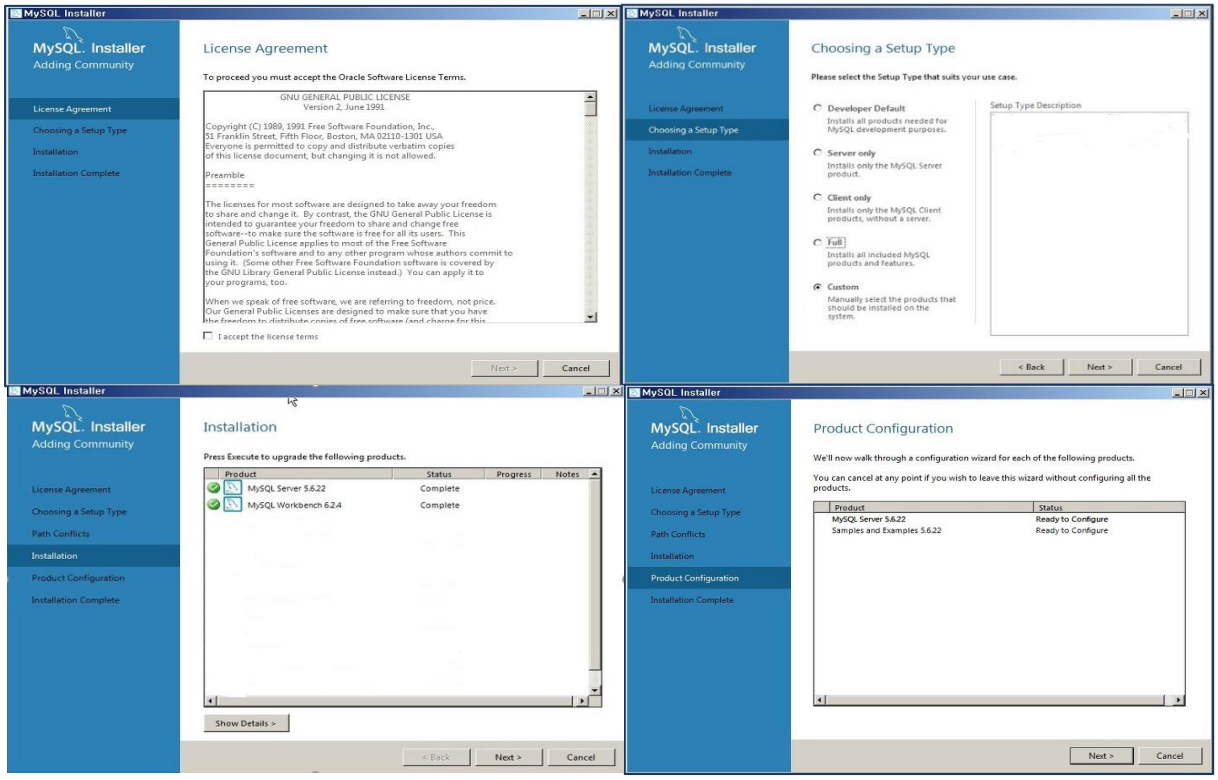

Figure 7 MySQL installation procedures

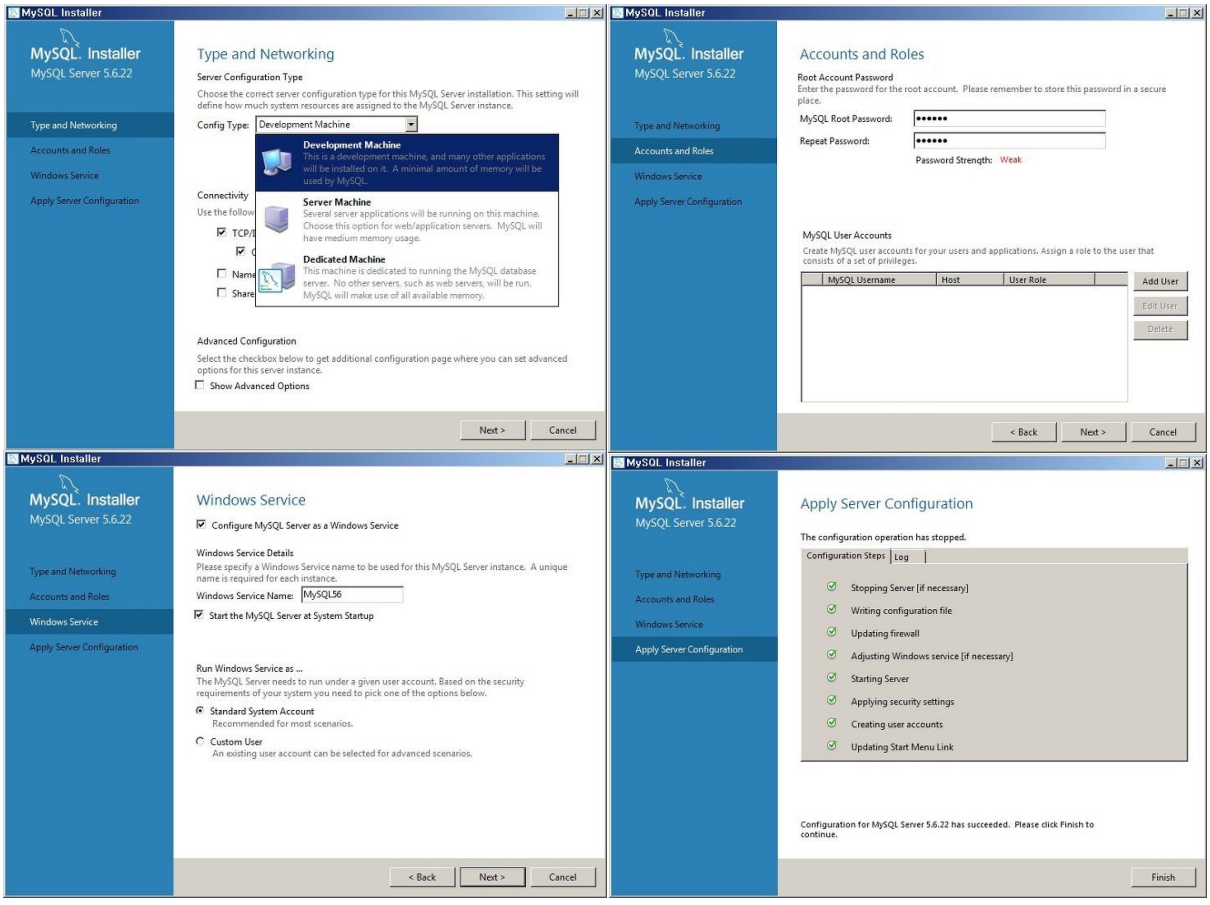

Figure 8 MySQL Runtime configurations

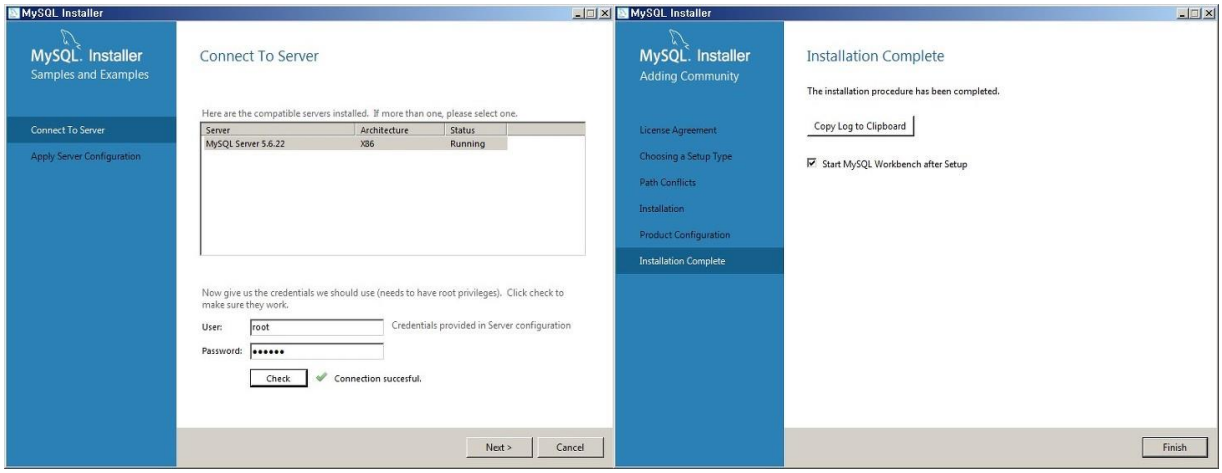

Figure 9 Connect to MySQL server for test

Run MySQL workbench and create a MySQL connection and then connect to MySQL server with root password as shown as Figure .

Figure 10 Create a MySQL connection and connect to the server

MySQL workbench provides both a graphical user interface (GUI) and command line to manage the database, data tables, import and output existing database as well as execution of queries. If users are familiar with MySQL commands, you can choose command line to input commands to control MySQL database. Here we demonstrate how to configure MySQL database through workbench GUI.

Until this step, we have installed MySQL successfully and then we have to configure the MySQL in terms of creation of a database for Mobius. In OCEAN alliance website download page, a MySQL database file can be downloaded to use. How to download and import that file will be explained later.

Step 1: Create a Mobiusdb Schema

After the MySQL is connected properly, right click the mouse at the workbench GUI as depicted as Figure and create a schema using popped tab. Name the new schema as ‘mobiusdb’ and apply the schema creation. Then you can see the new created ‘mobiusdb’ schema at the SCHEMA column at the left down side of GUI, select the mobiusdb schema and right click to select ‘Set as Default Schema’ to set the current schema as default.

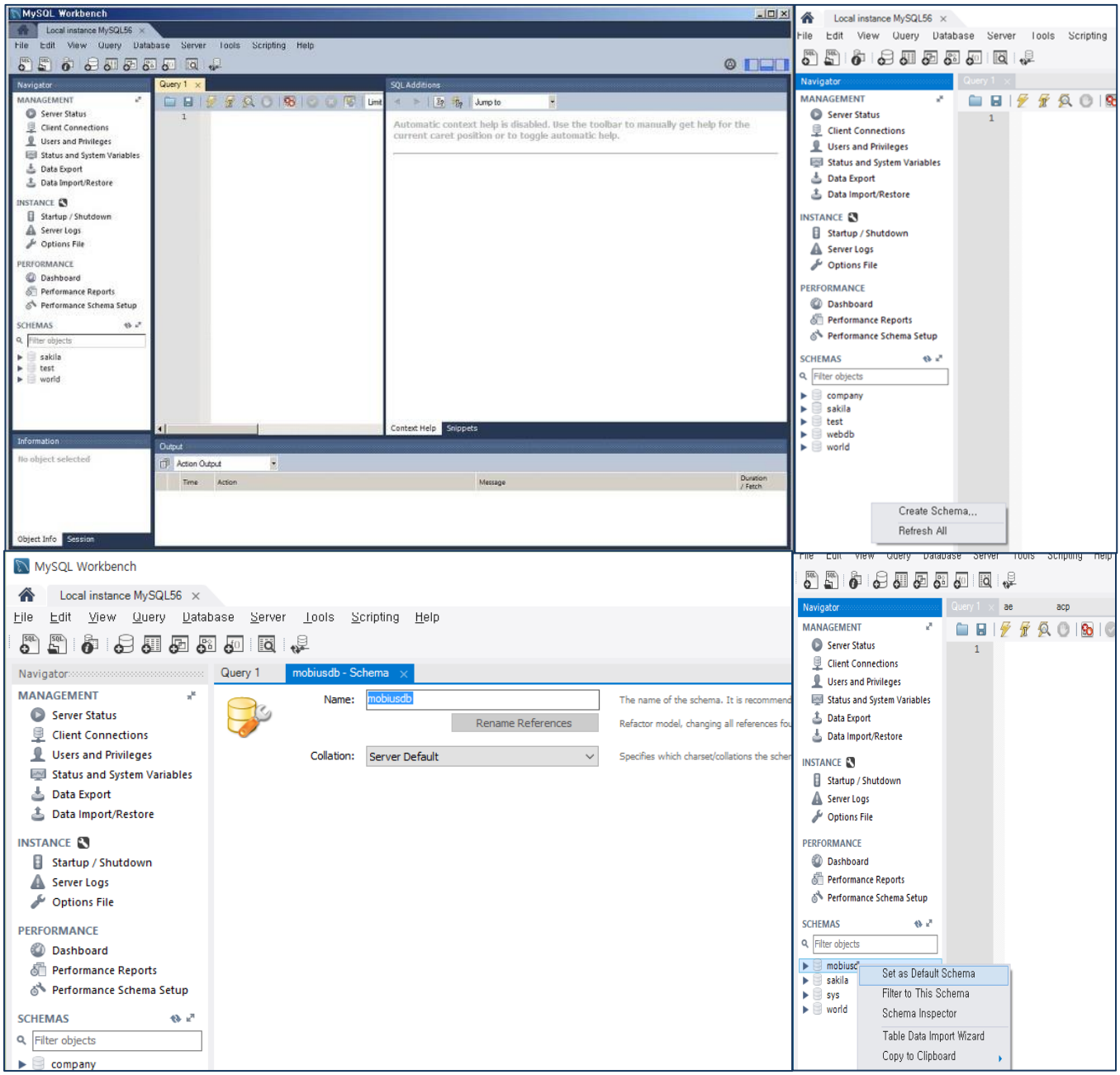

Figure 11 Create a MySQL connection and connect to the server

Step 2: Import Mobiusdb database

All we need regarding for configuring Mobius database are bundled into a sql script file (mobiusdb.sql). Users can download freely the .sql scipt file from OCEAN alliance website Mobius page.

As depicted as Figure 12, import the downloaded .sql script file through ‘Data Import’ popped window. In ‘Import from Disk tab’, select option ‘Import from Self-Contained File’ and browse to the downloaded .sql script and select ‘the default schema to be imported to’ as created ‘mobiusdb’ schema. Finally click the ‘Start Import’ to apply the change.

After the .sql file is imported, refresh the mobiusdb schema and you can see the mobiusdb schema is updated with empty tables are updated.

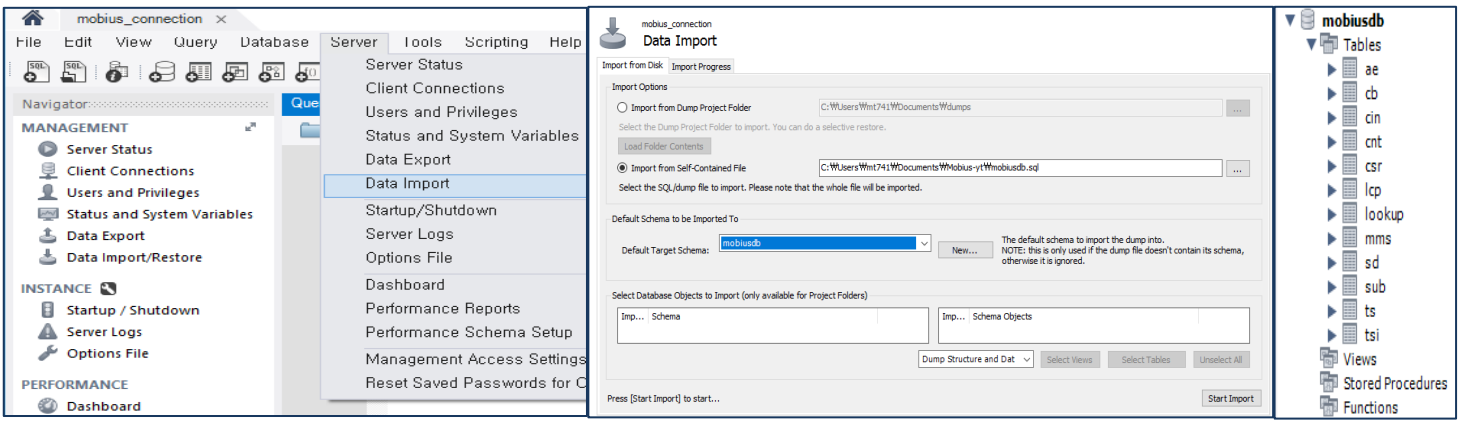

Figure 1 Import Mobius sql script file

2.2.2. Node JS Installation

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. In contrast to multi-thread based servers, node.js operating on a single thread using non-blocking I/O calls allows it to support tens of thousands of concurrent connections. With node.js, users don't need worry about the dead-locking of processes because almost no function in node.js directly performs I/O so there is no locks problem at all and therefore, scalable systems can be reasonably developed through node.js.

Figure 13 Introduction of Node.Js

Node.js was originally written by developer Ryan Dahl in 2009 and distributed as an open source and currently a diverse of modules developed from contributors are included covering file system I/O, networking, binary data (buffers), cryptography and data streams etc.

For more details about Node.js, please visit Node.js homepage.

A number of versions (binary/source) support for different platforms are provided in NodeJs downloads page. Users can download a preferred version from here and install it following set up wizard.

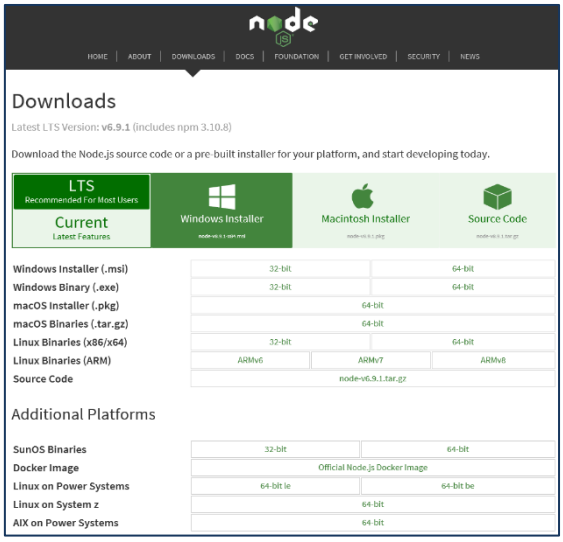

Figure 14 Download Node.js package

After installation of Node.js, check the system environment variables table (here taking Windows system as an example) to ensure that the installation path of node.js (default is C:\Program File\nodejs) is already added automatically in PATH environment variable and if there is no node.js path, update the PATH with node.js path as shown as Figure .

To test whether Node.js is installed properly, open a Windows Command Prompt window and under the user directory, run `node` command to enable node.js mode. In the node.js mode, input `console.log("hello node.js");` if receives response `hello node.js` then it indicates the Node.JS are installed and works properly.

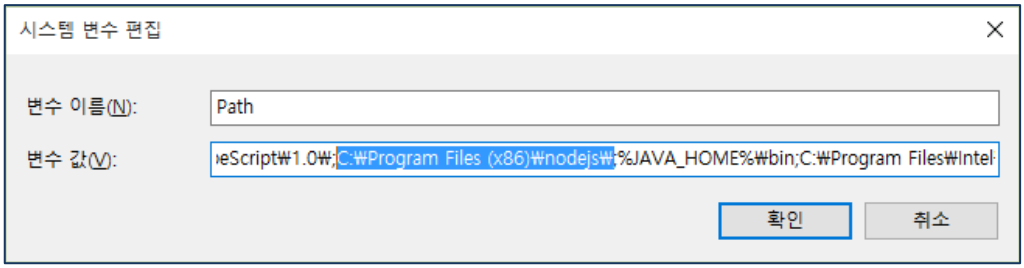

Figure 15 Update PATH environment variable with Node.js installation path

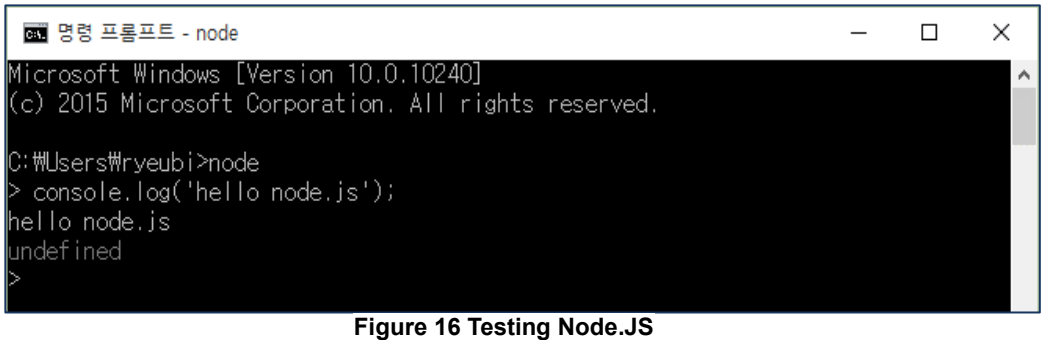

Figure 16 Testing Node.JS

2.3. &Cube:Rosemary Installation

&Cube:Rosemary is distributed with the source code package in OCEAN alliance website (http://www.iotocean.org) or the GitHub (https://github.com/IoTKETI/nCube-Rosemary).

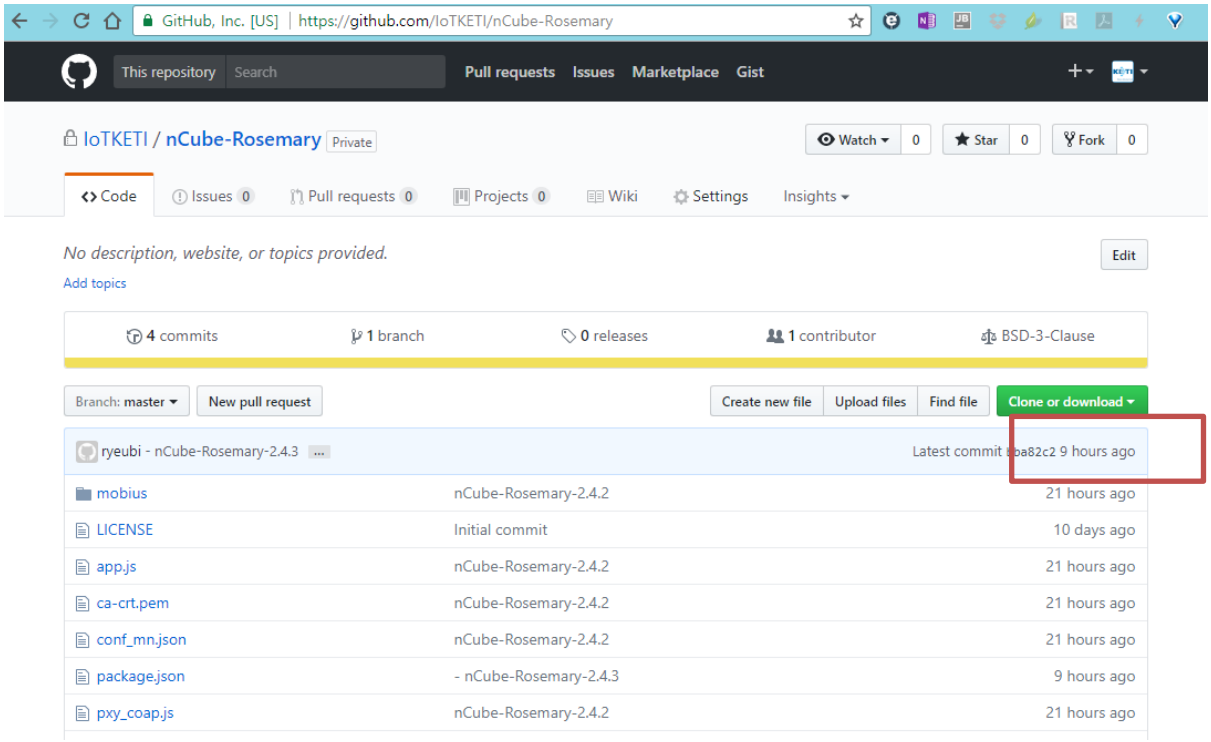

Figure 17 Download from GitHub

After downloading, unzip the package and you will see the included folders and files as Figure . The roles functions of the files are described in Table 1.

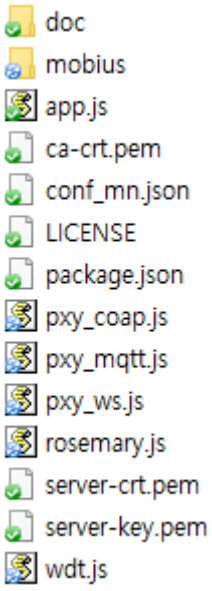

Figure 18 Unzipped package files

The unzipped node.js files can be directly executed without any additional compiling operation.

But there are several node.js modules are not yet installed. To installed the required additional node.js modules, open a Windows command prompt window and go to the directory where the unzipped package and run `npm install` command as shown as below.

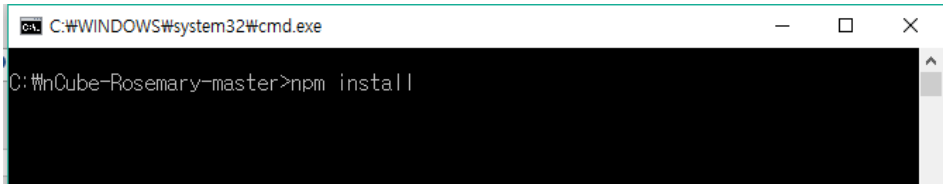

Figure 19 Run `npm install` command to install additional node.js modules

After running the npm install command, all additional required node.js modules will be generated and stores in node_modules folder as shown as Figure 20. You can see there is a node_modules folder in the unzipped source folder.

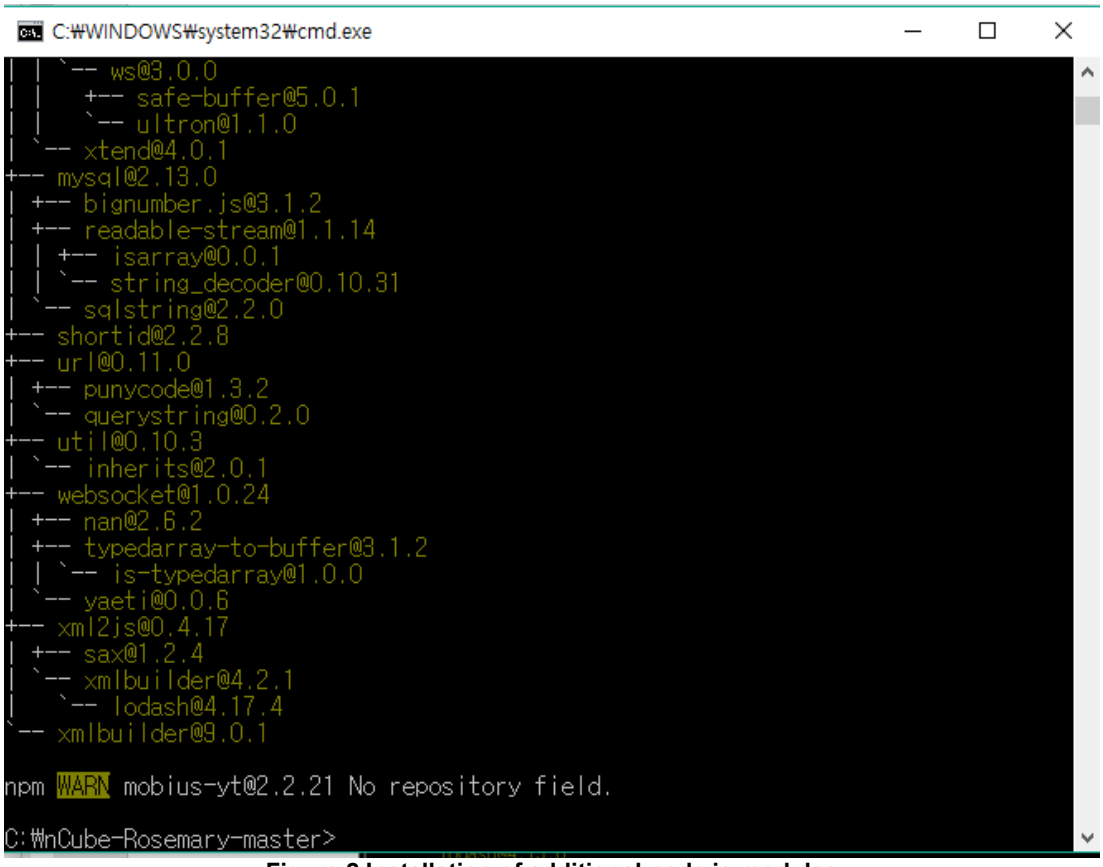

Figure 2 Installation of additional node.js modules

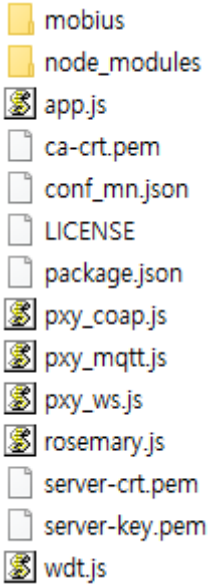

Figure 3 A node_modules folder created in unzipped source folder

Until this step, we have installed all required software and modules to run &Cube:Rosemary. Users can run node mobius.js in the command line to activate the &Cube:Rosemary. The installation is almost done and only a few configurations are left.

3. Running &Cube:Rosemary

3.1. Runtime Environment Configuration

A configuration file conf_mn.json is included in the unzipped folder. User can customize the communication port `csebaseport`, and the MySQL database connection password `dbpass`. These configurations are required to guarantee the &Cube:Rosemary is accessible from the CSEBase port and the it can access to the MySQL database with the configured password.

```
{
    "parent" : {
        "cbname": "Mobius",
        "cbcseid": "/Mobius",
        "cbhost": "203.253.128.161",
        "cbhostport": "7579",
        "cbprotocol": "http",
        "mqttbroker": "203.253.128.161"
    },
    "csebaseport": "7599",
    "dbpass": "dksdlfduq2"
}
```

Figure 4 conf_mn.js configuration file

In the figure above, the “parent” represents the settings of the Registrar CSE that &Cube:Rosemary registers with.

The following figure shows the list of additional configurations in the rosemary.js file.

```
// my CSE information
global.usecsetype           = 'mn'; // select 'in' or 'mn' or 'asn'
global.usecsebase           = 'rosemary';
global.usecseid             = '/rosemary';
global.usecsebaseport       = conf.csebaseport;

global.usedbhost            = 'localhost';
global.usedbpass            = conf.dbpass;


global.usepxywsport         = '7573';
global.usepxymqttport       = '7574';


global.usetsagentport       = '7572';

global.usemqttbroker        = parent_mqttbroker; // mobius to mqttbroker
global.usesecure            = 'disable';
```

Figure 5 Other configurations in rosemary.js

3.2. Running &Cube:Rosemary

Now users can run `node rosemary.js` command to run &Cube:Rosemary.

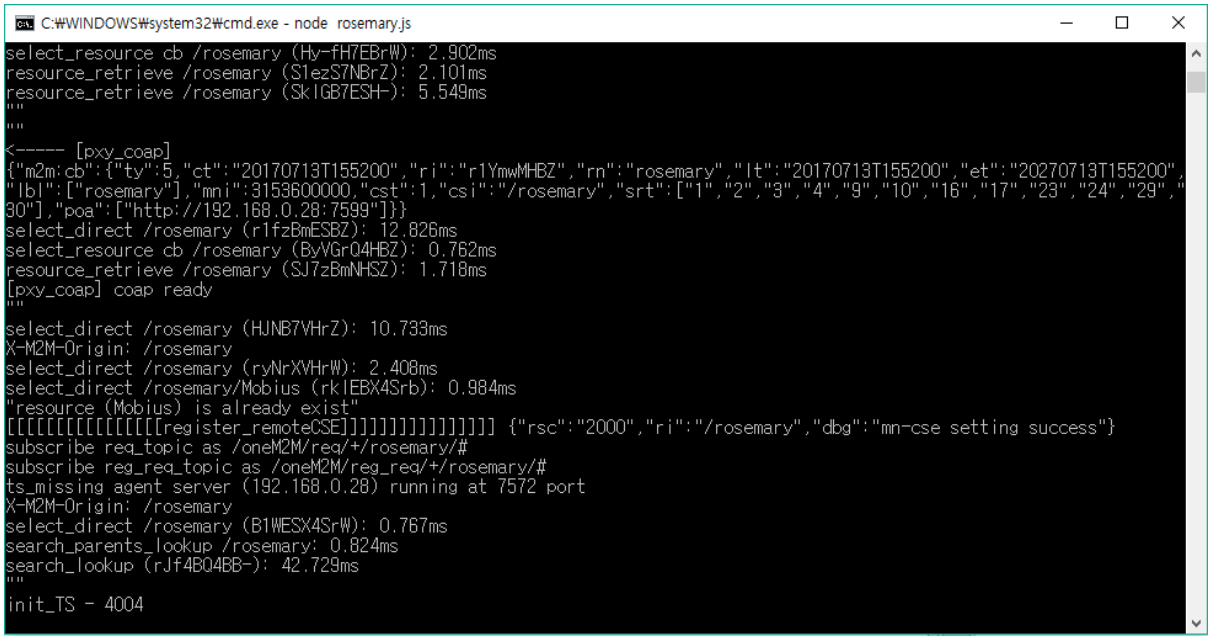

Figure 24. Running &Cube:Rosemary