

How to save MQTT messages to MySQL database

Most of the services we enjoy on the Web are provided by web database applications. Web-based email, online shopping, forums and bulletin boards, corporate websites, and sports and news portals are all database-driven.

To build a modern web site, you need to develop a database application. The MySQL database is suitable for a wide variety of use cases, including mission critical apps, dynamic websites, and as an embedded database for software, hardware, and appliances.

Many IOT platforms, including AWS, Google, and IBM, support MQTT, but most online broker, such as Mosquitto, cannot store incoming messages in databases. Most solutions are that we subscribe to the topic and receive all incoming messages that are coming to the topic and then store the incoming messages in the database.

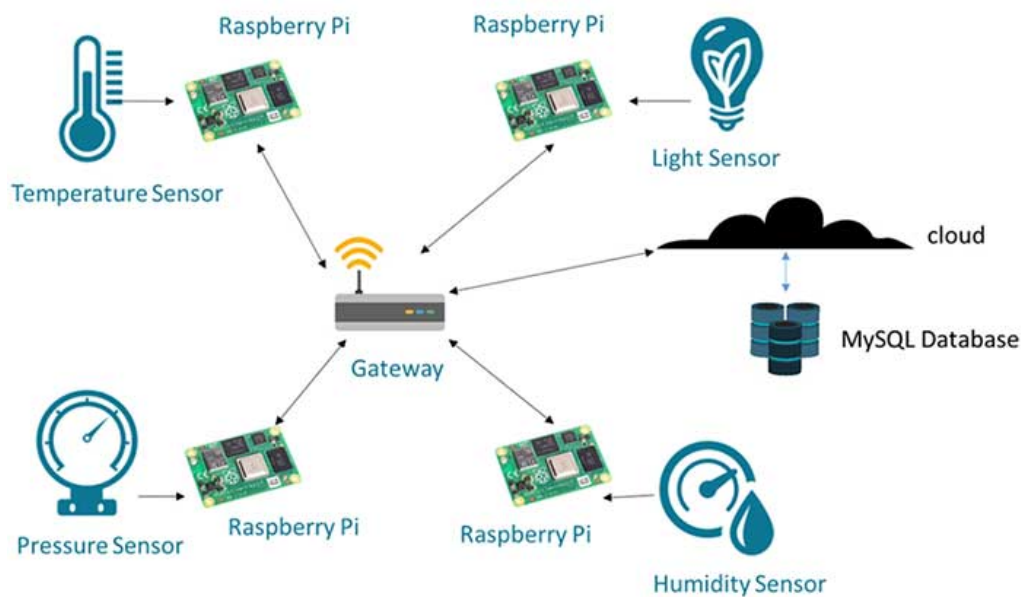
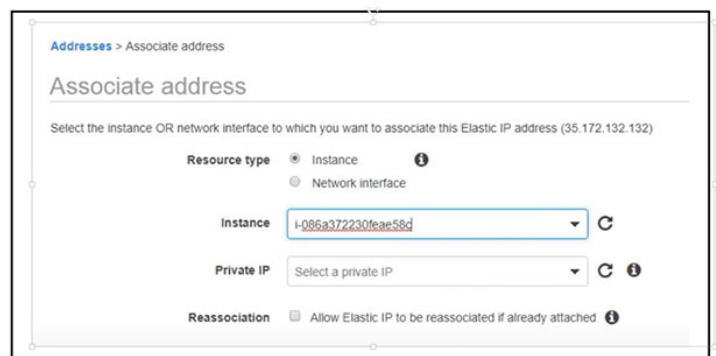
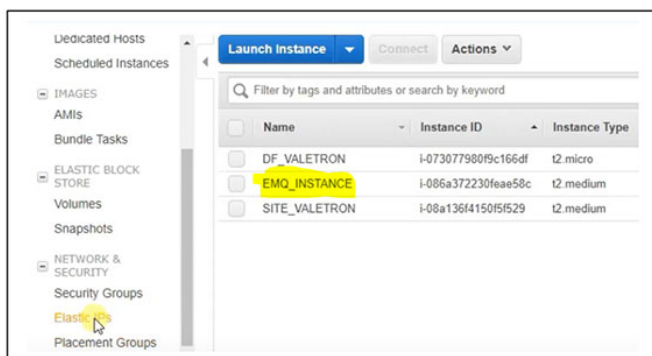


Figure: MQTT with MySQL database

In this blog, you will learn how to use the EMQ broker to store MQTT messages in the database. With EMQ Broker you can write plugins. These plugins can be used to tap the incoming and outgoing messages with 'hooks'. Hooks are the functions that are called when a certain event occurs.

Let's get started by creating a new Ubuntu Instance with the name EMQ instance on Amazon AWS console. Once the instance is created, assign IP address to that instance.



Update security groups in the AWS. Add the inbound rule to give access to port on which the MQTT broker will run.

Edit inbound rules

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0	e.g. SSH for Admin Desktop
Custom TCP	TCP	1883	Custom 0.0.0.0	e.g. SSH for Admin Desktop
Custom TCP	TCP	80	Custom 0.0.0.0, ::0	e.g. SSH for Admin Desktop
Custom TCP	TCP	18063	Custom 0.0.0.0	e.g. SSH for Admin Desktop

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel Save

Now open putty and access Ubuntu on the AWS server via SSH by giving the IP address assigned to the instance. The username by default is Ubuntu.

```

ubuntu@ip-172-31-62-236: ~
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-62-236:~$

```

After login we will install EMQ broker. Type the following commands to install dependency and broker

```

$ sudo apt-get update
$ sudo apt-get install build-essential
$ Sudo apt-get install erlang
$ git clone https://github.com/emqtte/emqx-rel.git
$ cd emqx-rel && make
$ cd _rel/emqxtd && ./bin/emqxtd console

```

Once the EMQ broker is installed we can install MySQL by following the steps below

```

$ sudo apt-get update
$ sudo apt-get install mysql-server
$ mysql_secure_installation
$ sudo apt-get install git curl zip unzip
$ sudo add-apt-repository ppa:ondrej/php
$ sudo apt-get install php7.4-fpm php7.4-common php7.4-xml php7.4-cli php7.4-curl php7.4-json php7.4-mcrypt php7.4-mysqld php7.4-sqlite php7.4-soap php7.4-mbstring php7.4-
zip php7.4-bcmath
$ sudo nano /etc/php/7.4/fpm/php.ini

```

Find the line that reads `cgi.fix_pathinfo=1` and Change it to read `cgi.fix_pathinfo=0`

```

$ cd ~
$ mkdir bin
$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
$ php composer-setup.php --install-dir=/home/ubuntu/bin --filename=composer
$ mysql -u root -p
mysql> CREATE DATABASE dreamfactory;
mysql> GRANT ALL PRIVILEGES ON dreamfactory.* to 'dfadmin'@'localhost' IDENTIFIED BY 'YOUR_PASSWORD_HERE';
mysql> FLUSH PRIVILEGES;
mysql> quit

```

```

ubuntu@ip-172-31-62-236: ~
ubuntu@ip-172-31-62-236:~$ ls
bin composer-setup.php emq-relx
ubuntu@ip-172-31-62-236:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.7.21-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE dreamfactory;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON dreamfactory.* to 'dfadmin'@'localhost' IDENTIFIED BY 'dfadmin';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> FLUSH PRIVILEGES;

```

```

$ sudo mkdir /opt/dreamfactory
$ sudo chown -R ubuntu /opt/dreamfactory
$ cd /opt/dreamfactory
$ git clone https://github.com/dreamfactorysoftware/dreamfactory.git
$ composer install --no-dev --ignore-platform-reqs
$ php artisan df:env

```

Configure the dreamfactory as shown in below figure. The username and password is dfadmin.

```

ubuntu@ip-172-31-62-236: /opt/dreamfactory
Created .env file with default configuration.
Created phpunit.xml with default configuration.

Which database would you like to use for system tables? [sqlite]:
[0] sqlite
[1] mysql
[2] postgresql
[3] sqlsrv
> 1

Enter your mysql Host:
> localhost

Enter your Database Port [3306]:
> 3306

Enter your database name:
> dreamfactory

Enter your database username:
> dfadmin

Enter your database password:
>

```

```
$ nano .env
```

Uncomment (remove the ##) the two lines that read ##DB_CHARSET=utf8 and ##DB_COLLATION=utf8_unicode_ci

```
$ php artisan df:setup
```

Answer the onscreen prompts to create your first admin user for the system

```

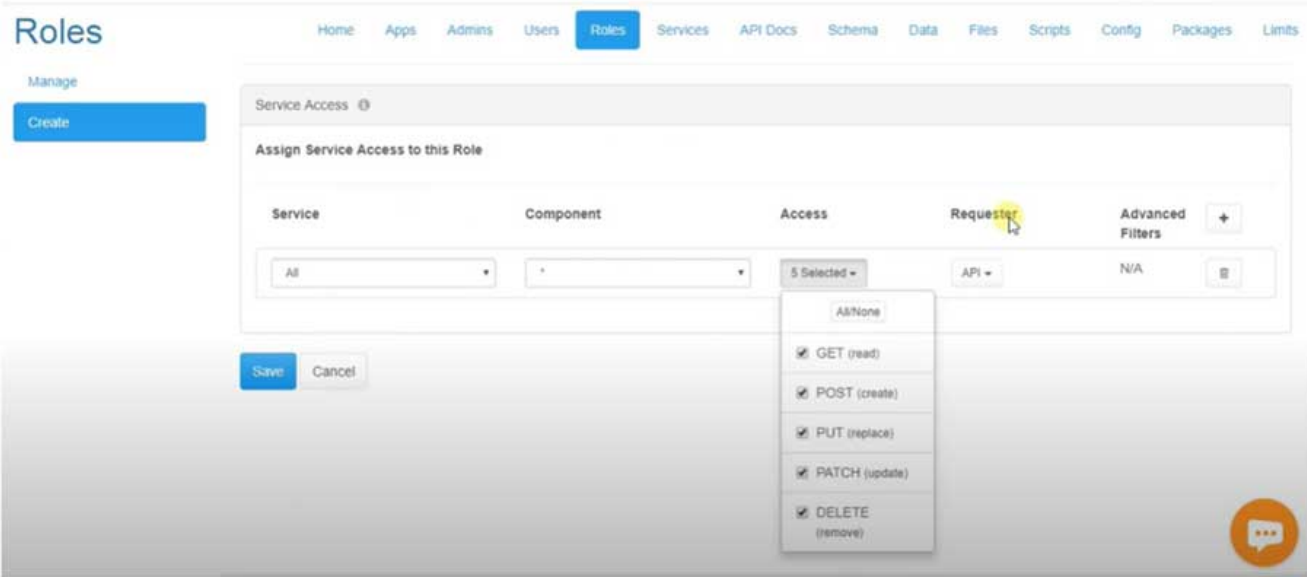
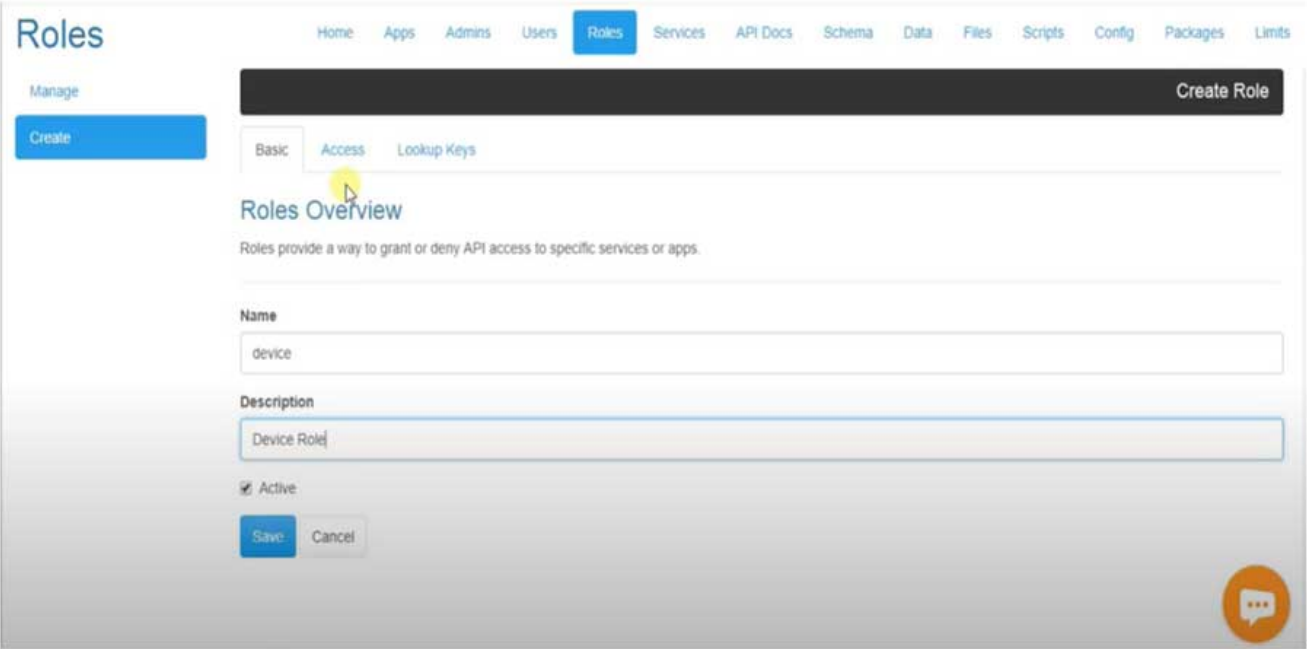
$ sudo chown -R www-data:ubuntu storage/ bootstrap/cache/
$ sudo chmod -R 775 storage/ bootstrap/cache/
$ php artisan cache:clear
$ sudo apt-get install nginx
$ cd /etc/nginx/sites-available
$ sudo cp default default.bak
$ sudo nano default

```

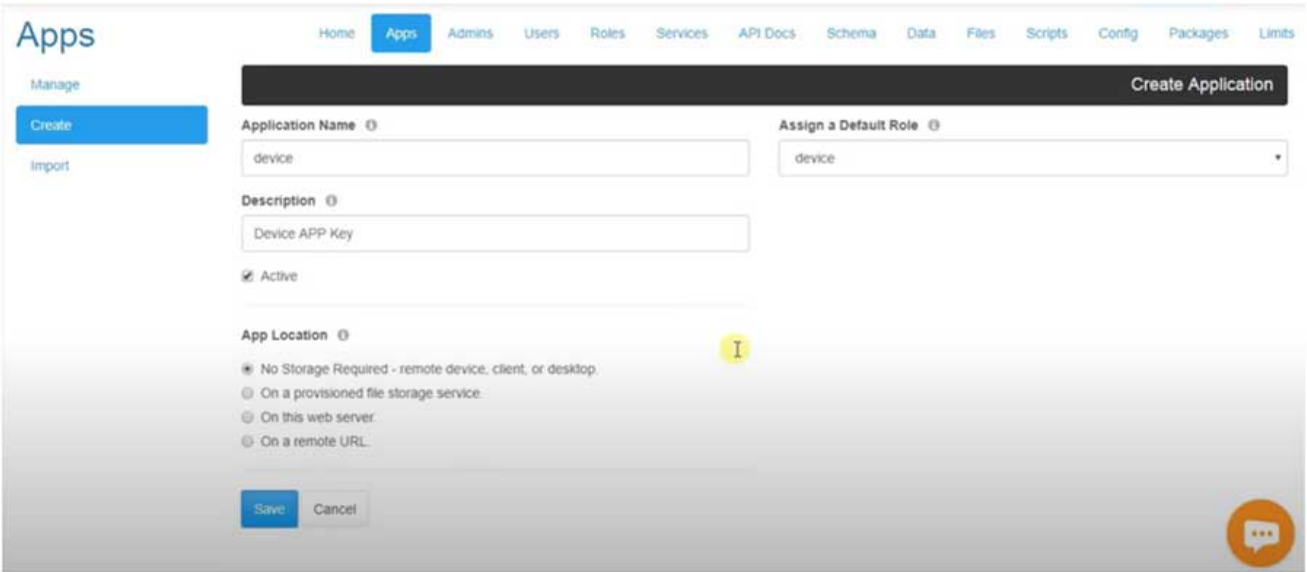
Copy and paste from the file attached and exit editor

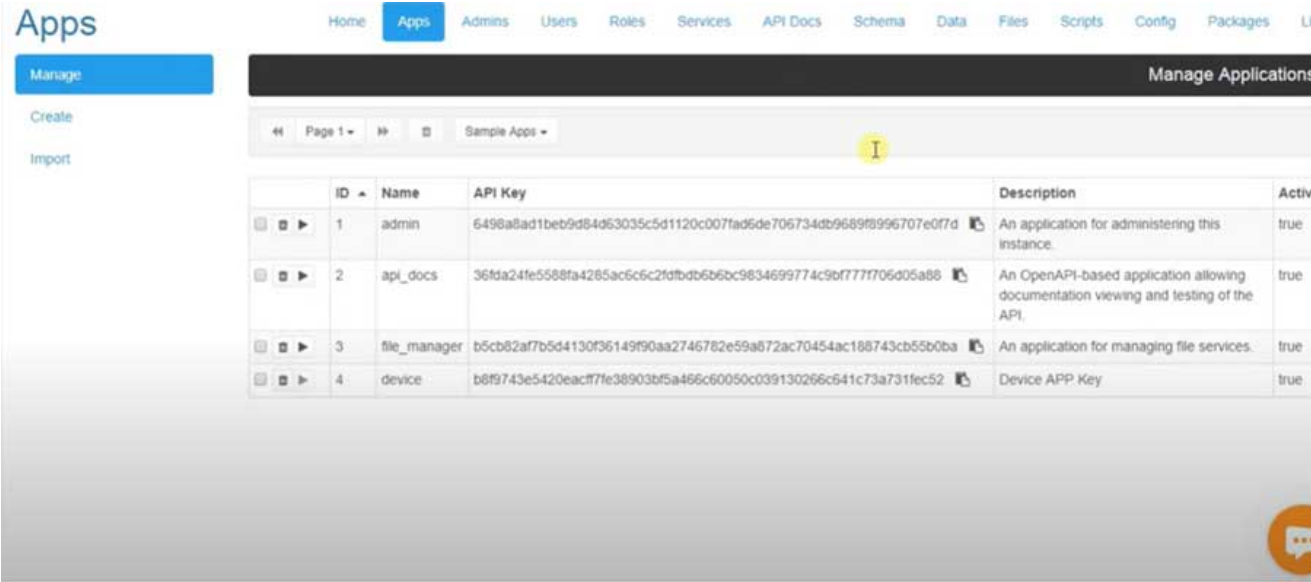
```
$ sudo service php7.4-fpm restart && sudo service nginx restart
```

At this point, the dreamfactory application is installed. DreamFactory is an open source REST API middleware platform that provides RESTful services for building mobile, web, and IoT applications. We will go to the installation of mySql database and create users so that the dreamfactory can access it. Login to dreamfactory by using default username and password which is entered during the setup process. We have to create roles which are permissions for devices. Enter the ip address which is generated at the time of EMQ INSTANCE creation for example 35.172.132.132/dreamfactory/dist/index.html

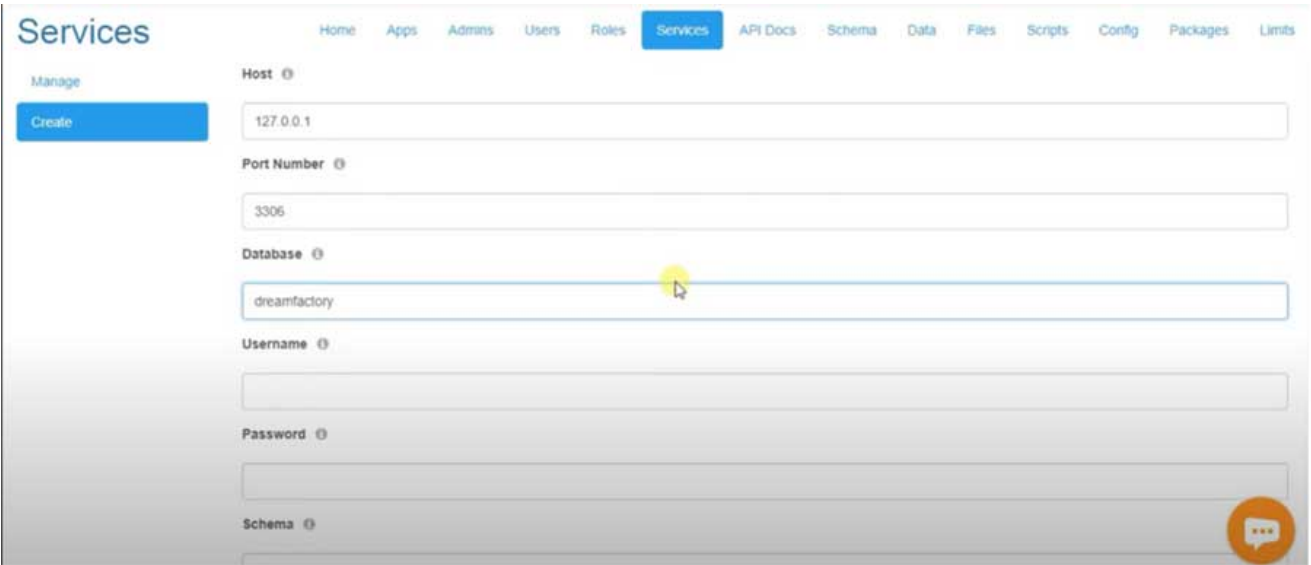
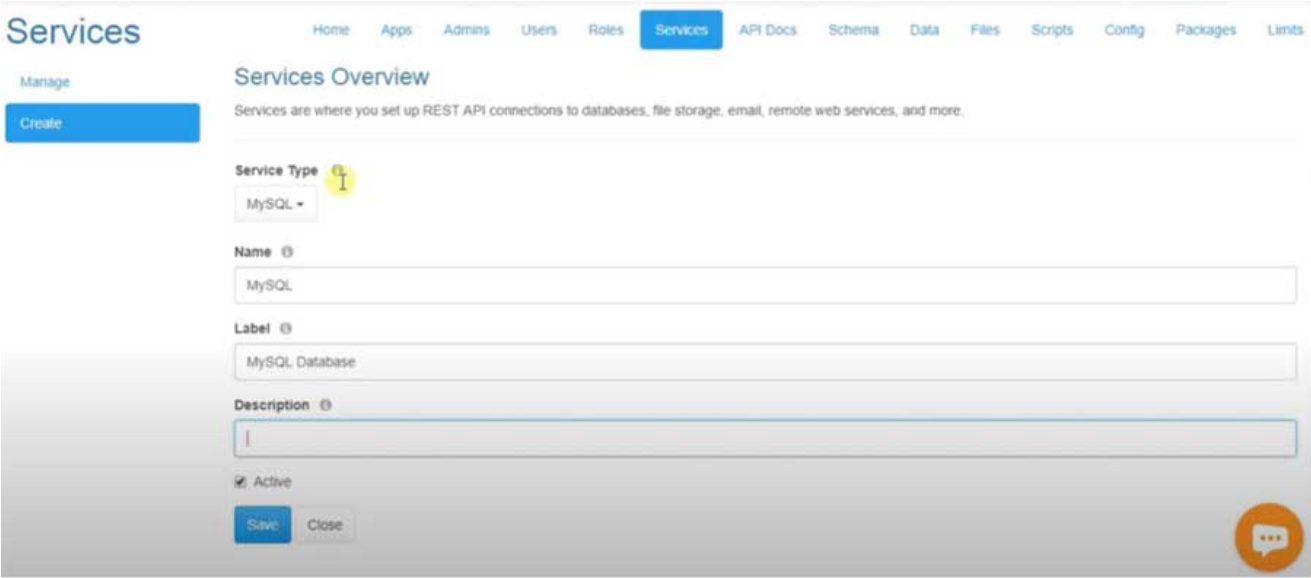


Go to the apps and generate the API key for the device which is the remote access key. Enter details as shown in the below figure





After the API key has been generated we will create a connection to the MySQL database click on the services and create then select MySQL database. Fill username, port number Database, username and Password.



The API key needs to be updated in the emq_plugin_template in order to send data to the MySQL database update the api key as shown below.

ubuntu@ip-172-31-62-236: ~/emq-relx/deps/emq_plugin_template/src

```
ubuntu@ip-172-31-62-236:~/emq-relx$ cd deps
ubuntu@ip-172-31-62-236:~/emq-relx/deps$ ls
bcrypt          emq_auth_mongo    emq_recon         gen_coap          mochiweb
bson            emq_auth_mysql    emq_reloader      gen_logger         mongodb
cliQUE          emq_auth_pgsql    emq_retainer      getopt            mysql
cuttlefish      emq_auth_redis    emq_sn            goldrush           neotoma
ecpooL          emq_auth_username emq_stomp          gproc             pbkdf2
ekka            emq_coap          emqttd            jsx               poolboy
emq_auth_clientid emq_dashboard     emq_web_hook      jwerl             recon
emq_auth_http   emq_lua_hook      epgsql            lager              syslog
emq_auth_jwt    emq_modules       eredis            lager_syslog
emq_auth_ldap   emq_plugin_template esockd             luerl
ubuntu@ip-172-31-62-236:~/emq-relx/deps$ cd emq_plugin_template/
ubuntu@ip-172-31-62-236:~/emq-relx/deps/emq_plugin_template$ ls
s: command not found
ubuntu@ip-172-31-62-236:~/emq-relx/deps/emq_plugin_template$ ls
ebin              erlang.mk         LICENSE           README.md        src
emq_plugin_template.d etc               Makefile         rebar.config     test
ubuntu@ip-172-31-62-236:~/emq-relx/deps/emq_plugin_template$ cd src/
ubuntu@ip-172-31-62-236:~/emq-relx/deps/emq_plugin_template/src$ ls
emq_acl_demo.erl  emq_cli_demo.erl  emq_plugin_template.erl
emq_auth_demo.erl emq_plugin_template_app.erl emq_plugin_template_sup.erl
ubuntu@ip-172-31-62-236:~/emq-relx/deps/emq_plugin_template/src$ nano emq_plugin_template.erl
```

GNU nano 2.5.3 File: emq_plugin_template.erl Modified

```
on_message_publish(Message, _Env) ->
io:format("publish ~s~n", [emqtt_message:format(Message)]),

MessageBin = element(12, Message),
MessageStr = binary_to_list(MessageBin),
inets:start(),
Method = post,
URL = "http://127.0.0.1/api/v2/mysql/_table/log",
Header = [{"X-DreamFactory-Api-Key", "b8f9743e5420eacff7fe38903bf5a466c60050c039"}, {"Content-Type", "application/json"}],
Body = MessageStr,
HTTPOptions = [],
Options = [],
R = http:request(Method, {URL, Header, Type, Body}, HTTPOptions, Options),

(ok, Message).
```

Get Help Write Out Where Is Cut Text Justify Cur Pos
Exit Read File Replace Uncut Text To Spell Go To Line

We can now send data from any client and save it in the database. Let us take Raspberry pi with sensors such as temperature, pressure and humidity connected. The paho-mqtt client running on the Raspberry pi sends the data to the MySQL database. To install paho mqtt in the Raspberry pi type in terminal.

```
$ pip3 install paho-mqtt
```

To connect to the server we can type

```
client = mqtt.Client()
client.on_connect = on_connect
client.connect("35.172.132.132", 1883, 60)
```

The below command will send topic to the broker and it gets saved in the MySQL database.

```
client.publish('raspberrypi/topic', payload=i, qos=0, retain=False)
```

Share

Tweet

Post

Stay informed

Keep up to date on the latest information and exclusive offers!

Subscribe now

Email Address

Submit

By clicking submit you agree to opt into our marketing emails

[Data Protection & Privacy Policy](#)



How To's
Short informative technical tutorials

Technical Resources

Articles, eBooks, Webinars, and more.
Keeping you on top of innovations.

Learn more

Technologies



Power Management



Wireless



Motor Control



Lighting



Sensing



Displays

Applications



Internet of Things



Industrial Automation



Transportation



Alternative Energy



Medical



Robotics



Maintenance & Safety