



BACK-END DATABASE AND API SERVER SETUP GUIDE

Abstract

This guide will walk the user through how to setup the backend server software (the API server and helper application) to use as an alternative to our servers.

Keith Erkert

kerkert@asu.edu

Updated 11/24/2020

Table of Contents

Introductory.....	2
Hardware/Software Requirements	2
Environment Setup.....	3
Running the Backend Applications	11

Introductory

As an alternative to using the provided servers from the creators of the IoT monitoring application, users can choose to host their own backend applications for either testing purposes or for their own privacy. This guide will walk users through how to setup the server on a fresh install of Ubuntu 20.04. The backend applications will work on different distros of Linux and will also work on Windows 10/11, there is no guide on how to set the backend applications on those options.

The backend applications are built using .Net 6 framework in the C# language. IotbackendAPI is the web API server that will host the endpoints for the Raspberry Pi to send data, and for the phone apps to retrieve data from. It also handles authentication for users for both the Raspberry Pi and phone apps. The backend helper app runs in a loop and will cache external IP information (such as geolocational data and dns information) so the phone application does not have to query this from 3rd party APIs every time the user inspects an external flow. The application will refresh cached content once it is a week old. Both the web API and helper applications utilize a MySQL database which will be setup in this guide. Mariadb will the engine used to host the MySQL database locally.

Hardware/Software Requirements

Recommended Server Specifications

- Linux or Windows 10 OS

**This guide will use Ubuntu 20.04 LTS, a modern Linux Distro is recommended as the server uses .NET 6*

See <https://docs.microsoft.com/en-us/dotnet/core/install/linux> for supported Distros

- 4-8GB of RAM

**This is heavily dependent on OS and expected workload on this server. If it is being used for personal use only and only running the backend software, the RAM usage should be minimal (around 4GB if using Linux)*

- 1-5GB storage (not including OS size)

**The backend applications and libraries should not use more than 1GB of storage. User data size is dependent on how active the devices are on the pi network, how long the data is saved for, and how many users (if not for personal use).*

Environment Setup

Install .NET libraries

1. Open a terminal
2. Add Microsoft's packages to the package manager

```
wget https://packages.microsoft.com/config/ubuntu/22.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
```

3. Install .NET6 SDK

```
sudo apt-get update
sudo apt-get install -y apt-transport-https
sudo apt-get update
sudo apt-get install -y dotnet-sdk-6.0
```

4. Install .NET6 Runtime

```
sudo apt-get update
sudo apt-get install -y apt-transport-https
sudo apt-get update
sudo apt-get install -y aspnetcore-runtime-6.0
```

Install and Setup MariaDB

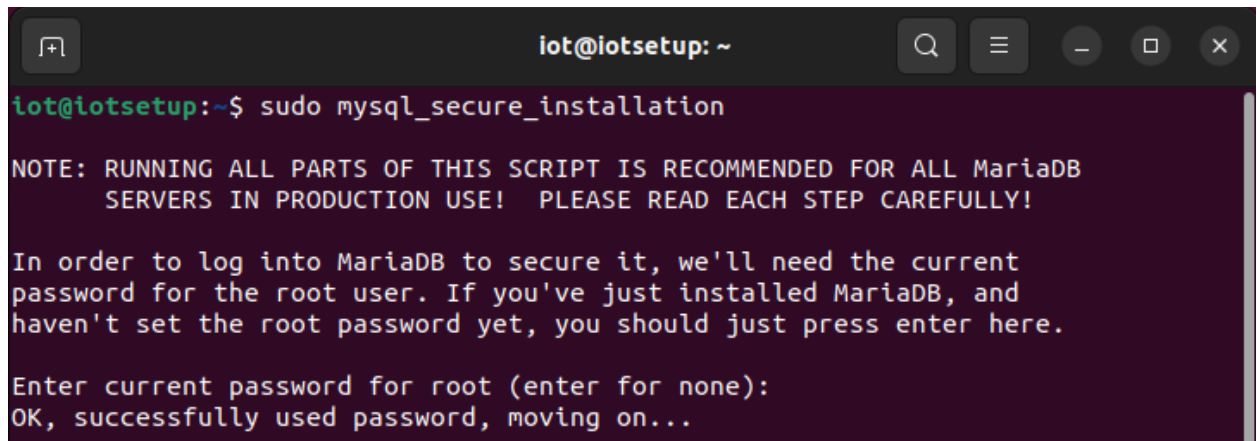
1. Update package manager and install MariaDB

```
sudo apt-get update
sudo apt install mariadb-server
```

2. Run setup for MySQL

```
sudo mysql_secure_installation
```

3. Enter a password (remember this password)

A terminal window titled 'iot@iotsetup: ~' showing the execution of 'sudo mysql_secure_installation'. The output includes a note about production use, instructions for setting a root password, and a confirmation message: 'Enter current password for root (enter for none): OK, successfully used password, moving on...'.

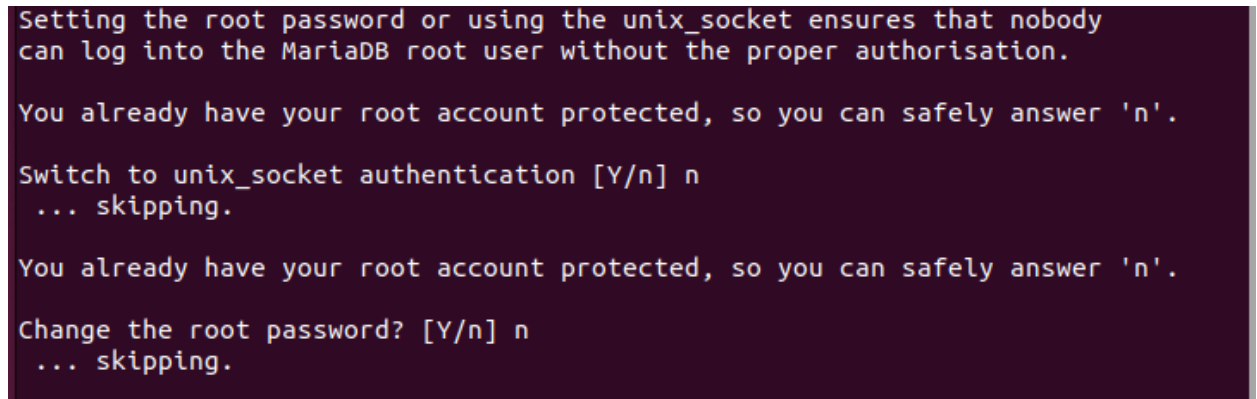
```
iot@iotsetup:~$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

4. Skip unix_socket authentication and change root password (answer n for both)

A terminal window showing the interactive prompts of 'mysql_secure_installation'. It asks to switch to unix_socket authentication (answered 'n') and to change the root password (answered 'n'). Both are skipped.

```
Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] n
... skipping.
```

5. Remove anonymous users, disallow remote access for root, delete the test database, and reload the privileges. (answer y for all 4)

```
By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.
```

```
Remove anonymous users? [Y/n] y
... Success!
```

```
Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.
```

```
Disallow root login remotely? [Y/n] y
... Success!
```

```
By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.
```

```
Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
```

```
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.
```

```
Reload privilege tables now? [Y/n] y
... Success!
```

```
Cleaning up...
```

```
All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.
```

```
Thanks for using MariaDB!
```

6. Open mysql under root user and the password used by ubuntu (not the password during setup)
7. Setup a user for the backend application to connect to the database on

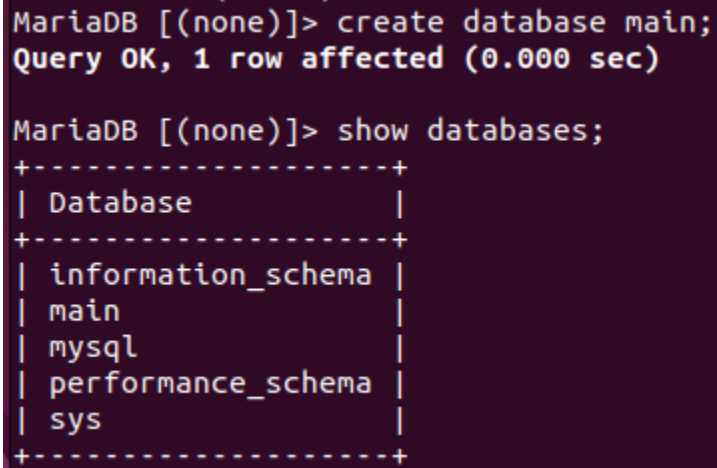
Replace **username and **password** with whatever you like, these will be used in the backend application.*

```
GRANT ALL ON *.* TO '**username**'@'localhost' IDENTIFIED BY '**password**'
WITH GRANT OPTION;
```

```
FLUSH PRIVILEGES;
```

8. Create non-dynamic tables the backend app uses. First make a new database called main

```
CREATE DATABASE main;
```



A terminal window with a dark purple background. The first command is 'MariaDB [(none)]> create database main;' followed by the output 'Query OK, 1 row affected (0.000 sec)'. The second command is 'MariaDB [(none)]> show databases;' followed by a table listing databases: information_schema, main, mysql, performance_schema, and sys.

```
MariaDB [(none)]> create database main;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> show databases;
+-----+
| Database                |
+-----+
| information_schema       |
| main                     |
| mysql                    |
| performance_schema       |
| sys                      |
+-----+
```

9. We need to let the database engine know we are using that database

```
USE main;
```

10. Create the main tables by copying and pasting the following code snippets (to paste in a terminal, use **ctrl+shift+v**)

```

CREATE SCHEMA main_users_blank;

CREATE TABLE main_users_blank.devices (
    device_id      INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    mac            VARCHAR(32) NOT NULL,
    description    VARCHAR(64) NOT NULL,
    nickname       VARCHAR(64) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE main_users_blank.ip_data (
    id             INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    device_ip      VARCHAR(32) NOT NULL,
    source_ip      VARCHAR(32) NOT NULL,
    source_port    VARCHAR(32) NOT NULL,
    destination_ip VARCHAR(32) NOT NULL,
    destination_port VARCHAR(32) NOT NULL,
    packets        INT NOT NULL,
    bytes          INT NOT NULL,
    flows          INT NOT NULL,
    captured       DATETIME NOT NULL,
    device_id      INT NOT NULL,
    protocol       VARCHAR(16) NOT NULL DEFAULT (''),
    duration       DECIMAL(12,4) NOT NULL DEFAULT (0.0000)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE INDEX devices_ip_data ON main_users_blank.ip_data ( device_id );

CREATE TABLE main_users_blank.network_rules (
    rule_id        INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    device_id      INT,
    valid_from_date DATETIME NOT NULL,
    valid_to_date  DATETIME,
    from_time      DATETIME,
    to_time        DATETIME,
    allday         BOOLEAN NOT NULL,
    utc_offset     INT NOT NULL,
    repeat_every   INT,
    repeat_on_day  INT,
    repeat_on_monthly INT,
    repeat_on_yearly BIGINT,
    repeat_end_count INT,
    repeat_end_daye DATETIME
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE INDEX fk_network_rules_devices ON main_users_blank.network_rules ( device_id );

CREATE TABLE main_users_blank.network_rules_block (
    block_id       INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    rule_id        INT NOT NULL,
    to_block       VARCHAR(128) NOT NULL,
    from_device    BOOLEAN NOT NULL,
    block_type     VARCHAR(24) NOT NULL,
    protocol       VARCHAR(16) NOT NULL,
    port           MEDIUMINT NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```



```
CREATE SCHEMA main;
```

```
CREATE TABLE main.ip_geolocation (
  id          INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  ip          VARCHAR(64) NOT NULL,
  added_on    DATETIME NOT NULL,
  updated_last DATETIME NOT NULL,
  continent_name VARCHAR(64) NOT NULL,
  continent_code VARCHAR(6) NOT NULL,
  country_name VARCHAR(64) NOT NULL,
  country_code VARCHAR(6) NOT NULL,
  region_name VARCHAR(64) NOT NULL,
  region_code VARCHAR(6) NOT NULL,
  city        VARCHAR(64) NOT NULL,
  district    VARCHAR(64) NOT NULL,
  zip         VARCHAR(16) NOT NULL,
  timezone_name VARCHAR(64) NOT NULL,
  timezone_offset INT NOT NULL,
  currency    VARCHAR(64) NOT NULL,
  isp_name    VARCHAR(64) NOT NULL,
  org_name    VARCHAR(64) NOT NULL,
  asn         VARCHAR(64) NOT NULL,
  as_name     VARCHAR(64) NOT NULL,
  lat         DOUBLE NOT NULL,
  lon         DOUBLE NOT NULL,
  dns         VARCHAR(128) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=3261 DEFAULT CHARSET=utf8mb4;
```

```
CREATE TABLE main.ports (
  port_id      INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  port_number  VARCHAR(10) NOT NULL,
  description  VARCHAR(512) NOT NULL,
  official    BOOLEAN NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=1206 DEFAULT CHARSET=utf8mb4;
```

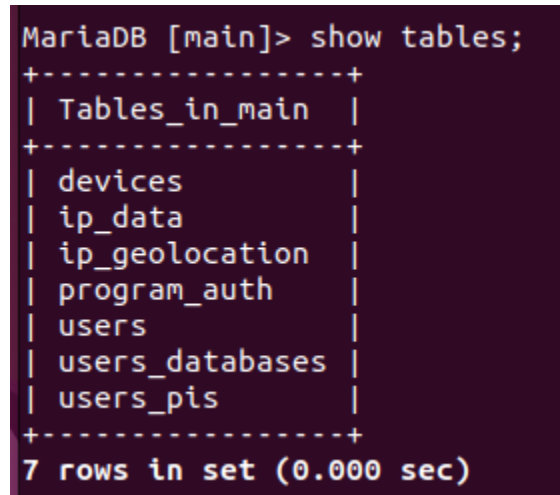
```
CREATE TABLE main.program_auth (
  id          INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nickname    VARCHAR(64) NOT NULL,
  client_token VARCHAR(64) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4;
```

```
CREATE TABLE main.users (
  user_id      INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  username     VARCHAR(128) NOT NULL,
  password     VARCHAR(128) NOT NULL,
  salt         VARCHAR(32) NOT NULL,
  access_token VARCHAR(64) NOT NULL,
  access_expiration DATETIME NOT NULL,
  refresh_token VARCHAR(64) NOT NULL,
  refresh_expiration DATETIME NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4;
```

```
CREATE TABLE main.users_databases (
  id          INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nickname    VARCHAR(64) NOT NULL,
  actualname  VARCHAR(64) NOT NULL,
  user_id     INT NOT NULL
)
```

11. Verify the tables were made

```
SHOW TABLES;
```



```
MariaDB [main]> show tables;
+-----+
| Tables_in_main |
+-----+
| devices        |
| ip_data        |
| ip_geolocation |
| program_auth   |
| users          |
| users_databases|
| users_pis      |
+-----+
7 rows in set (0.000 sec)
```

12. As an extra layer of security, our webAPI verifies the connecting client has a valid client token (to reduce web crawlers from getting more information about our http endpoints), add the following client tokens to the program_auth tables

```
INSERT INTO program_auth (nickname, client_token) VALUES
('pi', '94d9ba988fa2aea8339142716dd809220358a74f393855f09aab268a4f673c6');
INSERT INTO program_auth (nickname, client_token) VALUES ('phone
app', 'b3062fe926f91b22be3941f7aefa9f5b053fad6cb8ea575583fdc127e05b759e');
```

13. Open the port the web API listens on to allow the Raspberry Pi and Phone Application to communicate with the web API. This will also enable the firewall

```
sudo ufw allow 6000/tcp
sudo ufw enable
```

This only opens the port for devices on your LAN/WLAN, not the outside internet

14. To use the phone application outside your home network, forward port 6000 (TCP) in your router for the server's IP. You should also reserve the IP for the server's MAC address, so it will not change if the router resets from a power outage. Please refer to online guides from your router's manufacturer on how to do this.

This will open the webAPI to the internet, other devices from outside your network will be able to connect to the webAPI. There is an authentication layer to the webAPI to prevent anonymous users from accessing your data, but this is a potential security vulnerability, use at your own risk!

Running the Backend Applications

1. Download the WebAPI and backend helper application from GitHub.
 - a. Web API – <https://github.com/ASU-IoT-ResearchProject/IotBackendAPI/releases/tag/v22w45d7a>
 - b. Backend Helper – <https://github.com/ASU-IoT-ResearchProject/IoTBackendHelper/releases/tag/v22w45d7a>
2. Unzip each package into its own folder (if its zipped)
3. Open each folder in its own terminal
4. To start the web API run this command

```
dotnet IotBackendAPI.dll
```

5. To start the backend helper app run this command

```
dotnet backendhelpers.dll
```