Run Your Cloud MQTT Mosquitto Broker (access from anywhere using Digital Ocean)

Learn how to install Mosquitto Broker for MQTT communication on a Linux Ubuntu VM (Virtual Machine) using Digital Ocean. Running an MQTT Mosquitto Broker in the cloud allows you to connect several ESP32/ESP8266 boards and other IoT devices from anywhere using different networks as long as they have an Internet connection. We'll also cover how to connect your ESP boards to the cloud MQTT broker using Arduino IDE.

Updated 3 March 2023

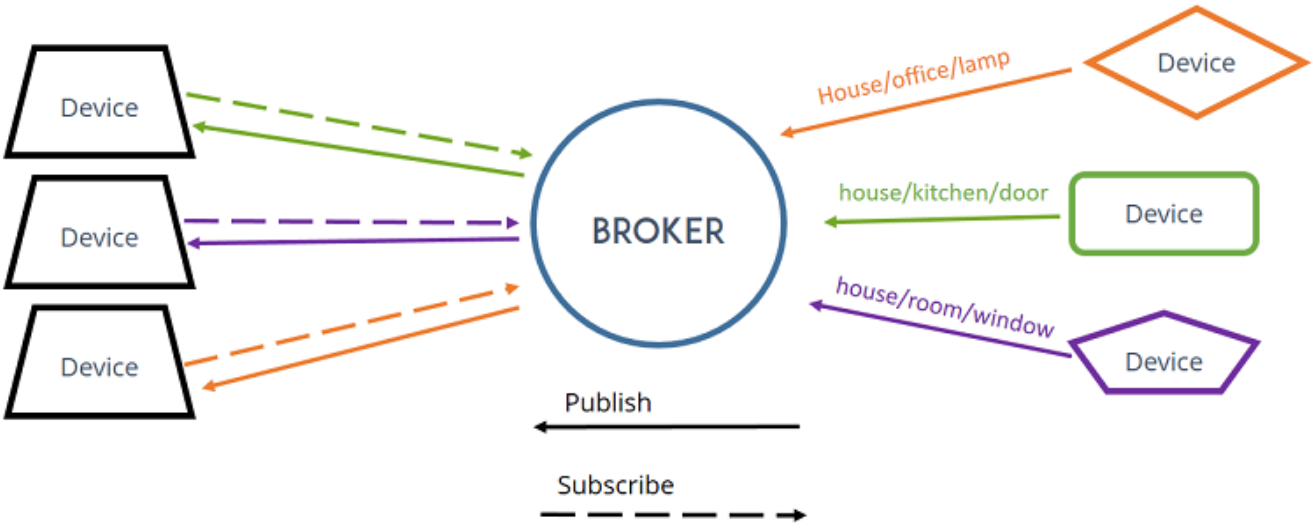
You might like: **SMART HOME with Raspberry Pi, ESP32, and ESP8266**—learn Node-RED and InfluxDB on a Raspberry Pi to build a Home Automation System with the ESP32 and ESP8266.

Introducing MQTT Protocol

MQTT stands for Message Queuing Telemetry Transport. It is a lightweight publish and subscribe system where you can publish and receive messages as a client. It is widely used in the home automation and IoT fields.

> To learn more about MQTT, read our complete guide: What is MQTT and how it works.

An MQTT broker is primarily responsible for receiving all MQTT messages, filtering the messages, decide who is interested in each message and then, publishing the messages to all subscribed clients.



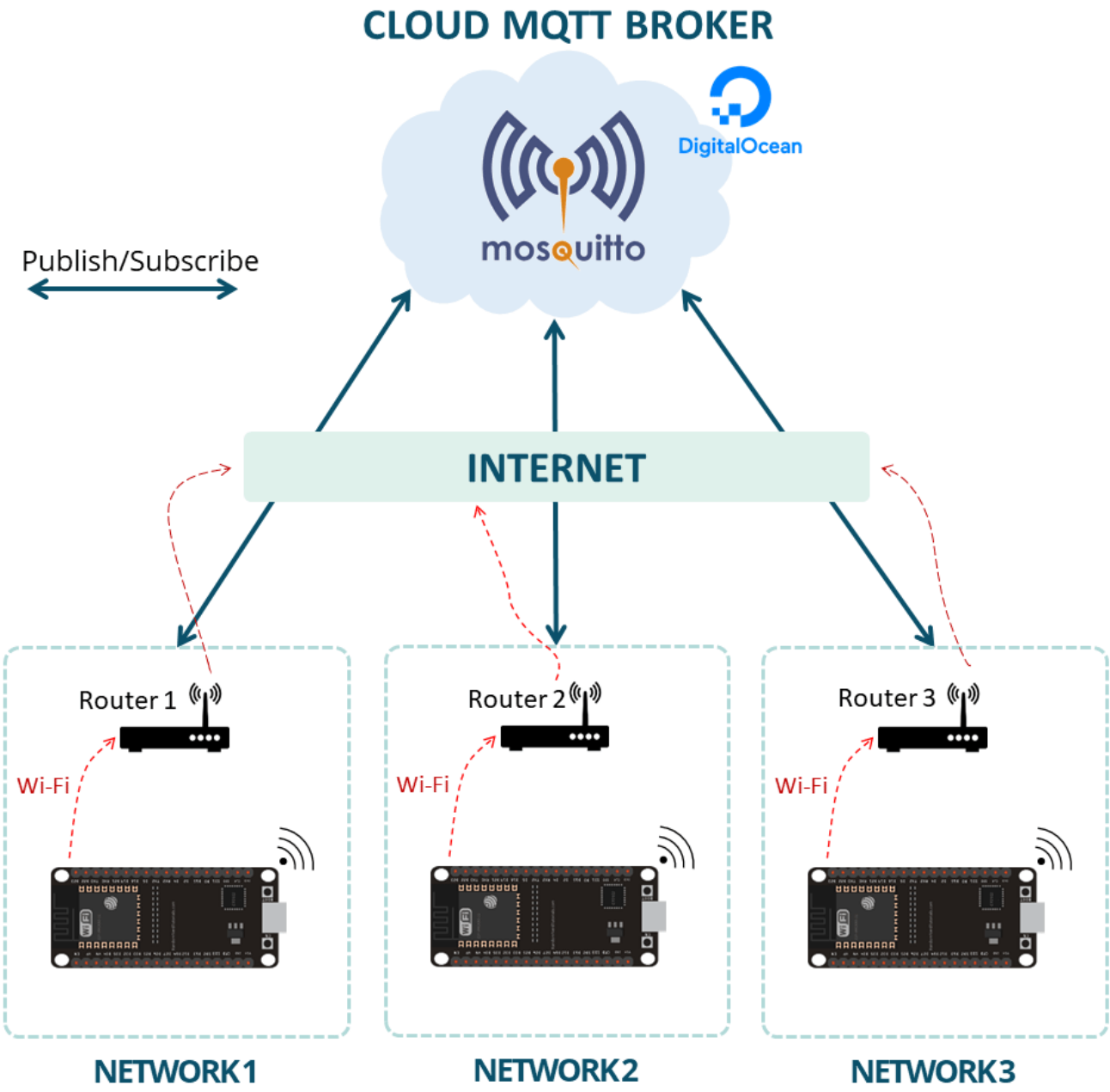
There are several brokers you can use. In our Home Automation projects and tutorials we use the popular Mosquitto MQTT Broker. It is easy to install, configure and use.

In this tutorial, we'll show you how to install Mosquitto MQTT broker on the cloud —a Linux Ubuntu VM (virtual machine) running on Digital Ocean hosting service.

Cloud MQTT Broker Overview

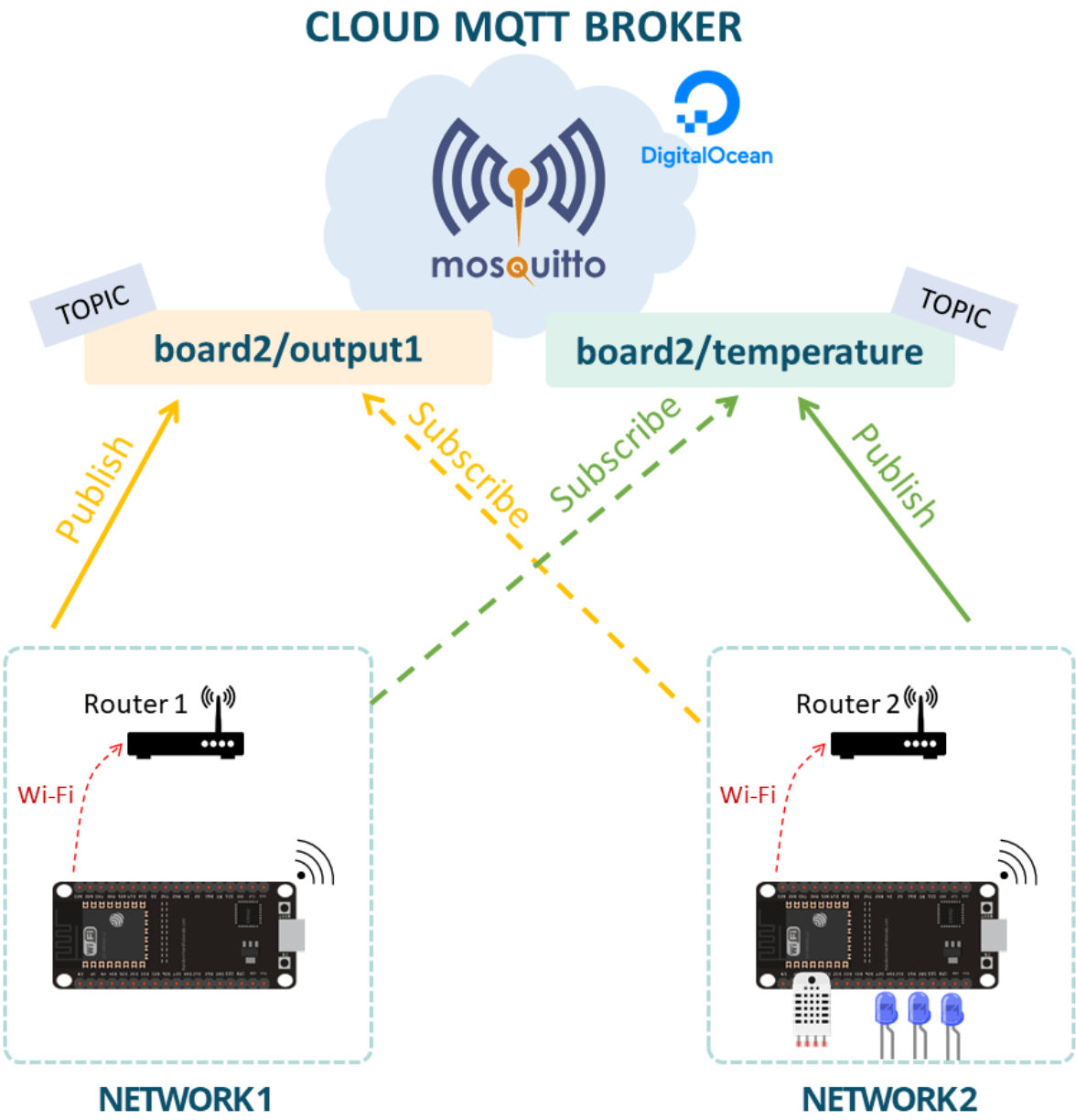
What's the advantage of using a Cloud MQTT broker and how it works?

Using a Cloud MQTT broker allows several IoT devices (like ESP32 and ESP8266 boards) to communicate with each other using MQTT, even if they are on different networks (different locations connected to different routers). Here's an overview.



- Mosquitto MQTT broker is running on the cloud (host service provided by Digital Ocean). So, it can receive messages from IoT devices all around the world.
- You can have several ESP boards on different networks that connect to the same Cloud MQTT broker.
- Each ESP board needs to be connected to a router that allows access to the internet in order to connect with the broker.
- Because the boards use the same MQTT broker, they can communicate with each other by publishing and subscribing to the same topics.

The following diagram shows an example of a possible application:



- The previous image shows two ESP32 boards on different networks. Each board is connected to a different router with access to the internet.
- Even though they are on different networks, they can communicate with each other via the Cloud MQTT broker by subscribing and publishing on the same topics.
- ESP32 #1 publishes on a topic that ESP32 #2 is subscribed to (`board2/output1`). The message can indicate whether ESP32 #2 should turn an output on or off. So, ESP32 #1 can control the ESP32 #2 outputs.
- Similarly, ESP32 #2 publishes temperature readings on the `board2/temperature` topic. ESP32 #1 is subscribed to that topic, so it receives board2 sensor readings.

computer or your smartphone. You can follow this tutorial: Access Node-RED Dashboard from Anywhere using Digital Ocean

Hosting Service – Digital Ocean

To run your Cloud MQTT Mosquitto Broker, you need to use a hosting service that allows you to have access to the command line and install any software that you need. I recommend using Digital Ocean that offers an Ubuntu server that you can manage through a command line.

I've been using it since 2015 and I personally recommend it, but you can use any other hosting service. Any hosting service that offers a Linux Ubuntu VM with full console access should work.

If you don't have a hosting account, I recommend signing up for Digital Ocean. When you sign up for Digital Ocean, you can try it for 60 days (they give you free credits to test the platform). You need to go to this link in order to claim the free credits: https://randomnerdtutorials.com/digitalocean.

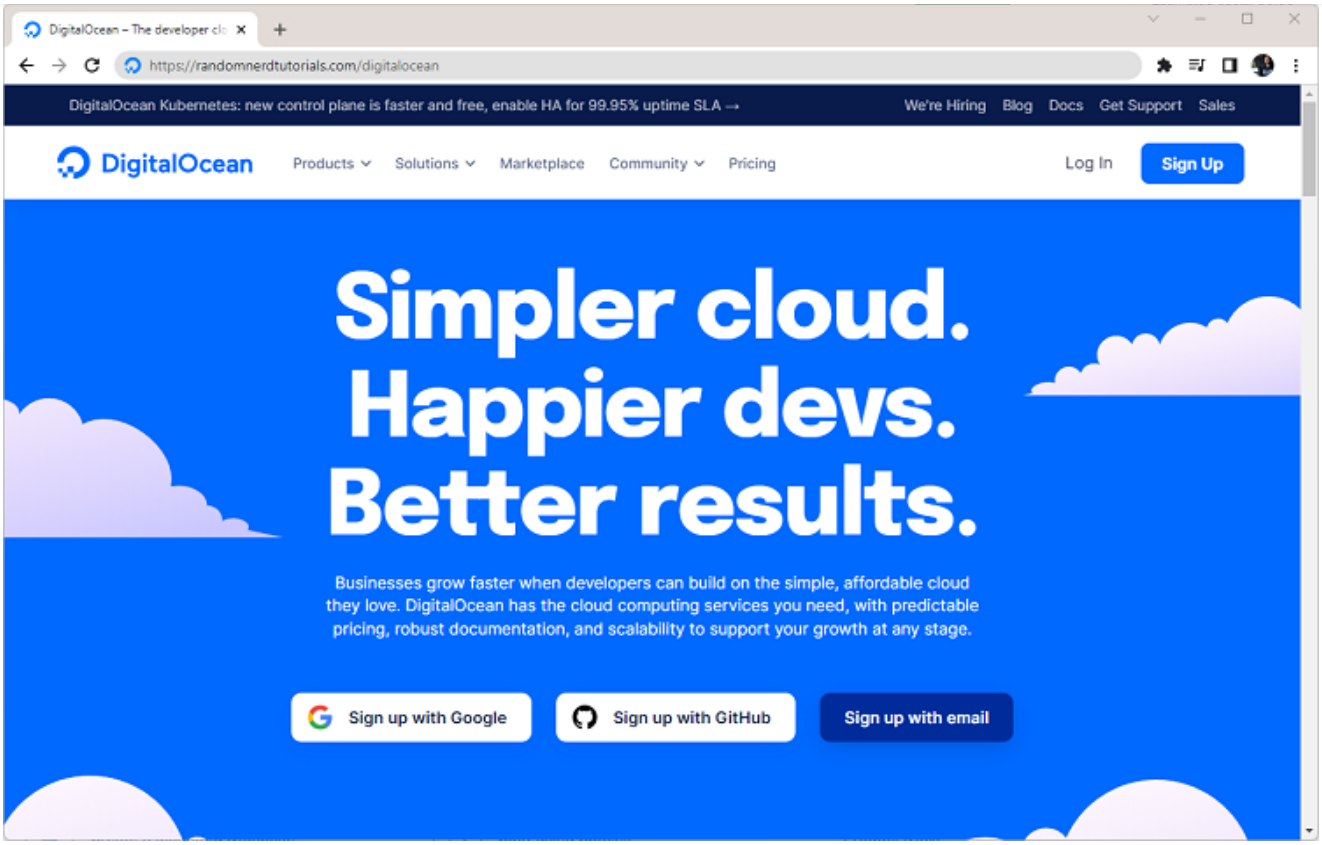
Grab Linux Ubuntu VM on Digital Ocean »

If you like our projects, you might consider signing up to the recommended hosting service, because you'll be supporting our work.

> **Note:** you can also run Mosquitto MQTT Broker in your local network using a Raspberry Pi board. However, the purpose of this tutorial is to run an MQTT broker in the cloud to communicate with boards (or other IoT devices) across different networks.

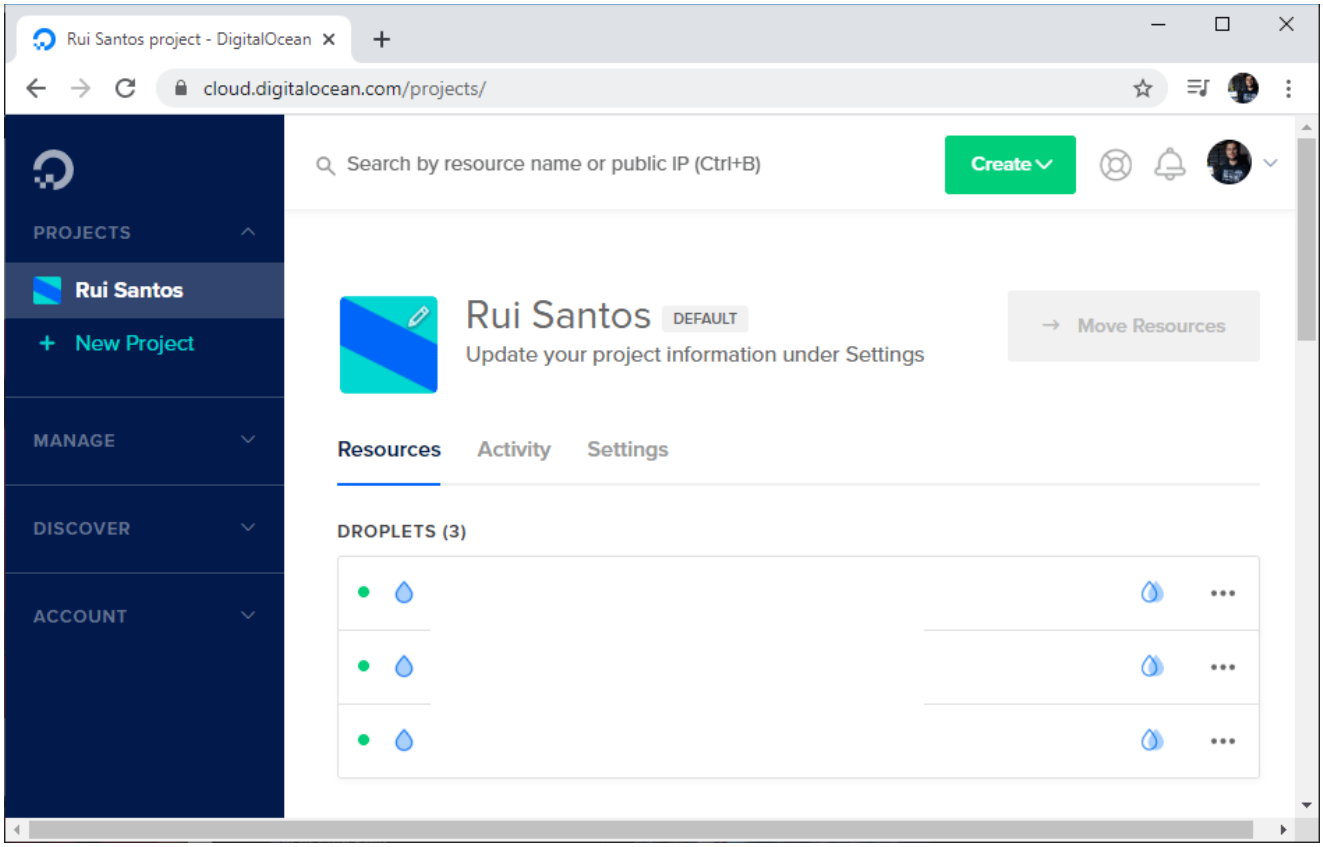
Creating Digital Ocean Account

To create a Digital Ocean Account, go to Digital Ocean and sign up using one of the available options.



Create your account, and you'll receive a $200 credit that you can use for 60 days to test the platform. You might need to enter a valid credit card, but you can cancel your account anytime if you're no longer interested in using the service after the free 60 days trial.

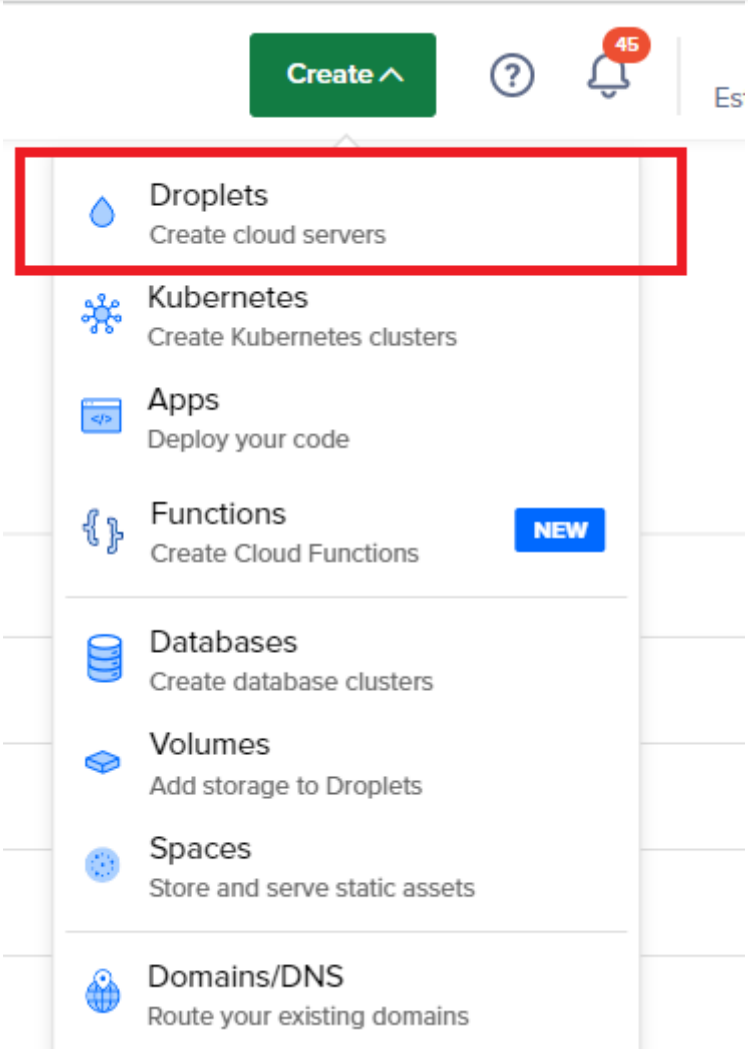
Confirm your account and login. On the **Project** tab, click on your name. You should see a similar Dashboard.



Create a Droplet (Linux Ubuntu VM)

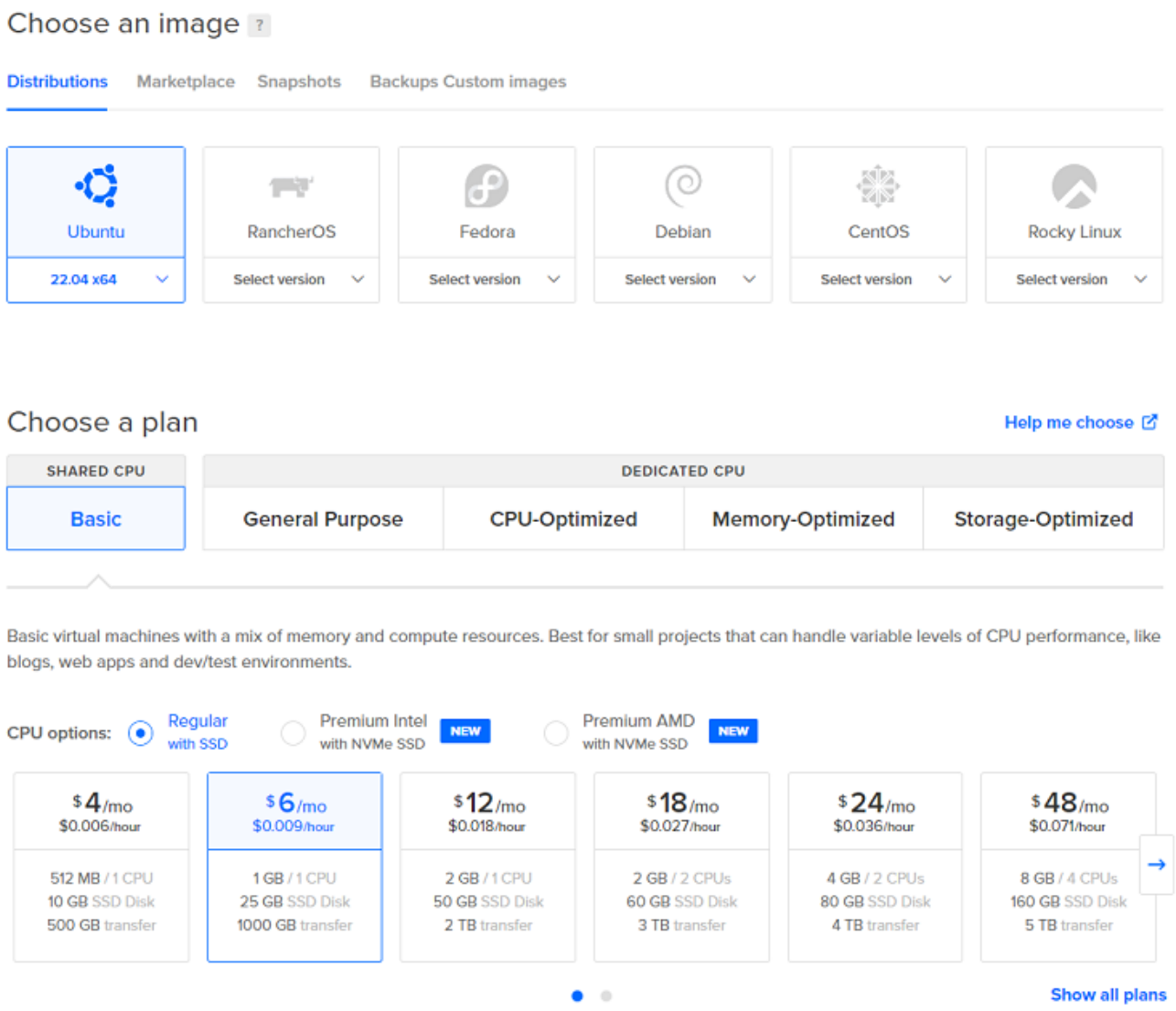
To create a new VM, press the “**Create**” button on the top right corner and select the “**Droplets**” option. Digital Ocean calls **Droplets** to its VMs.

> **Important:** if you’re already running a Droplet with Node-RED installed, you can skip these next steps (creating a Droplet). You can run both Node-RED and Mosquitto MQTT broker on the same server.

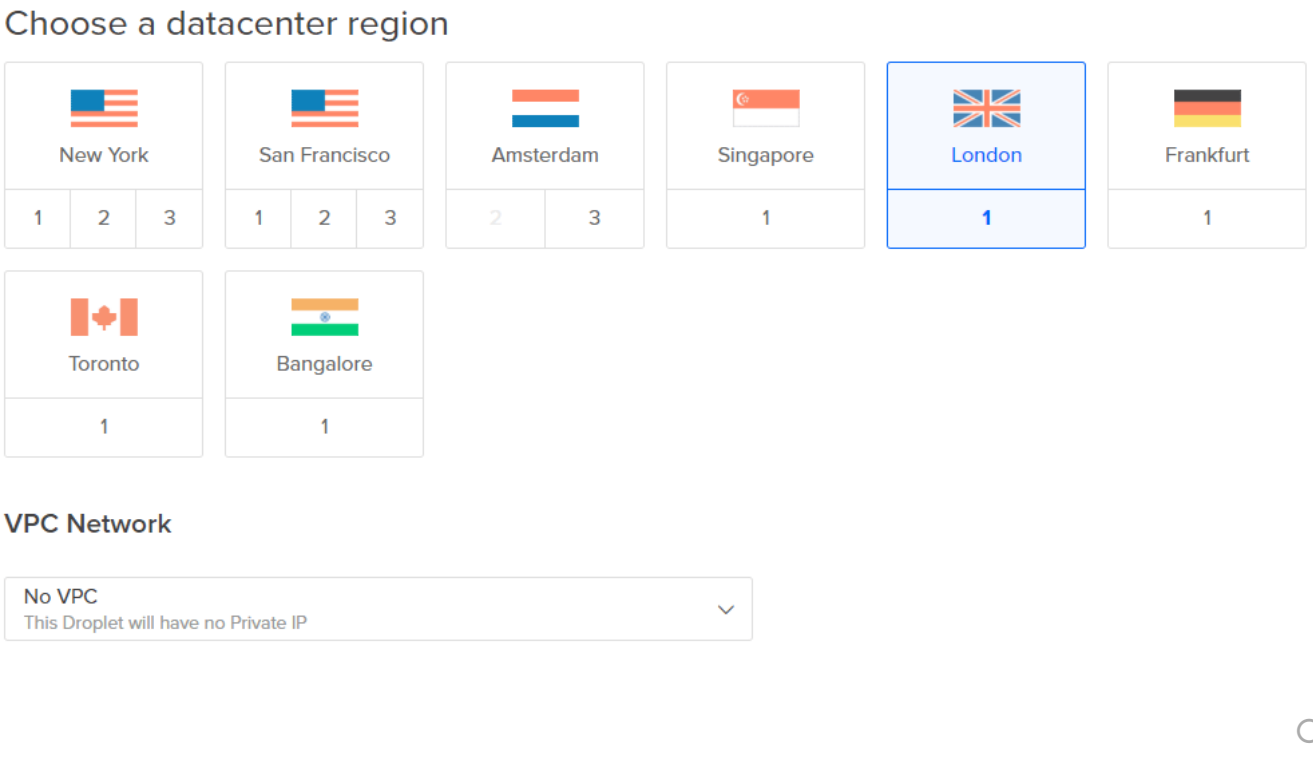


Then, select the following options:

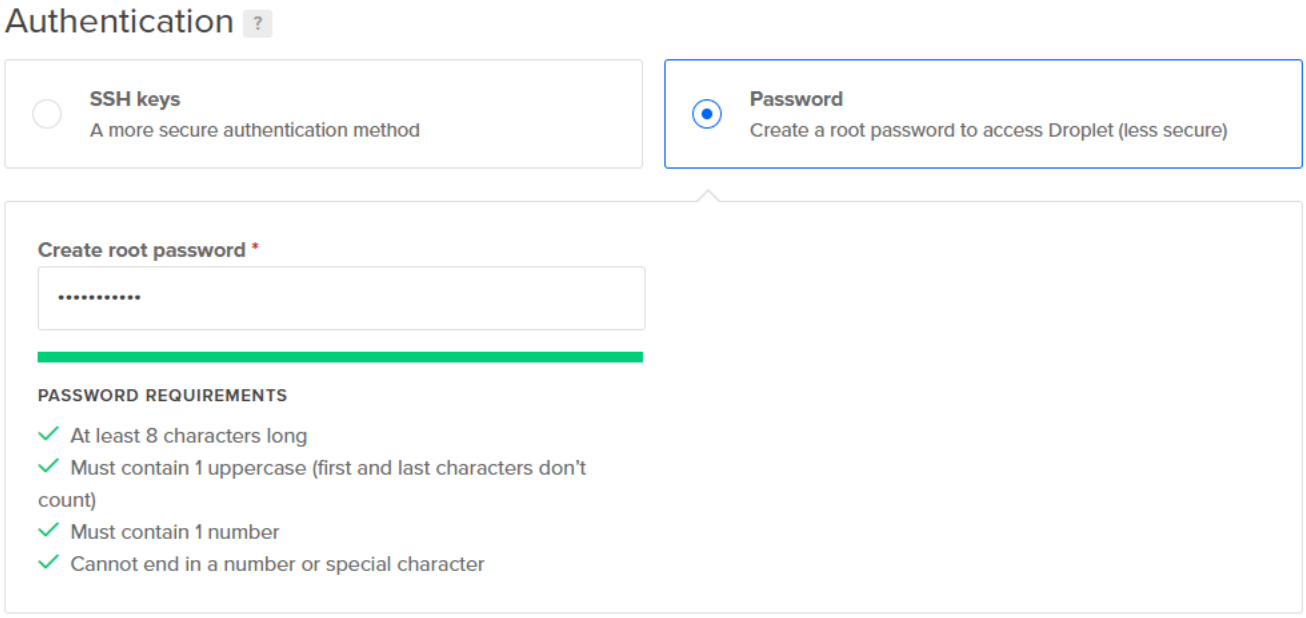
- Distributions: Ubuntu
- Choose a plan: Shared CPU Basic—we recommend choosing the $6/month option (the $4 plan will also work, but might be a bit slow).



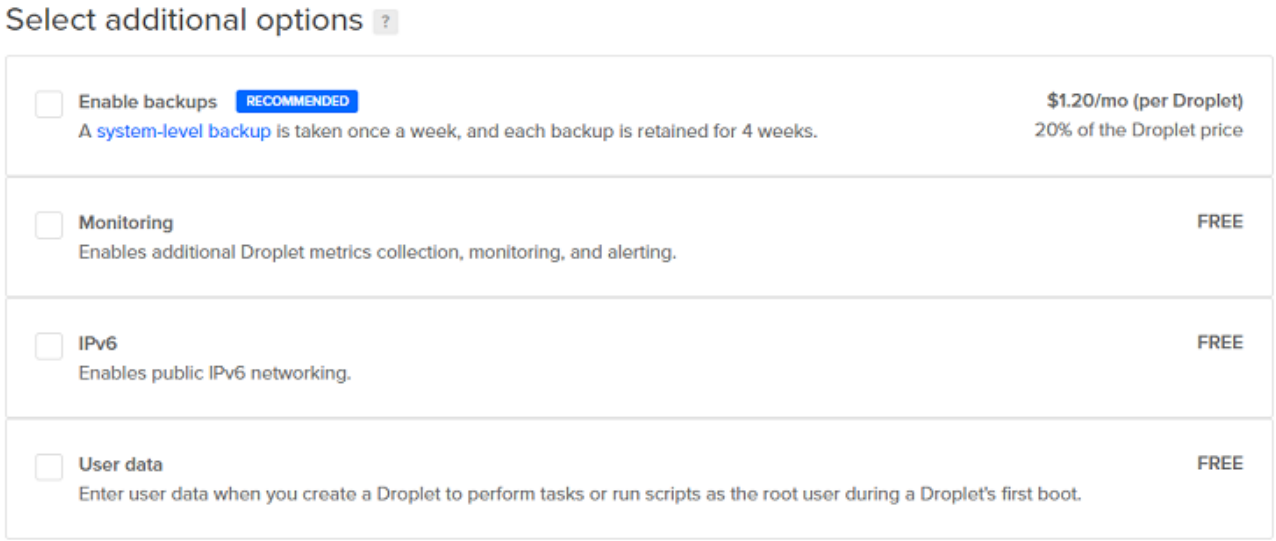
Choose a datacenter region—choose the closest to your location.



Create the root password that allows you to access your Droplet (save this password, because you'll need it to access your server).

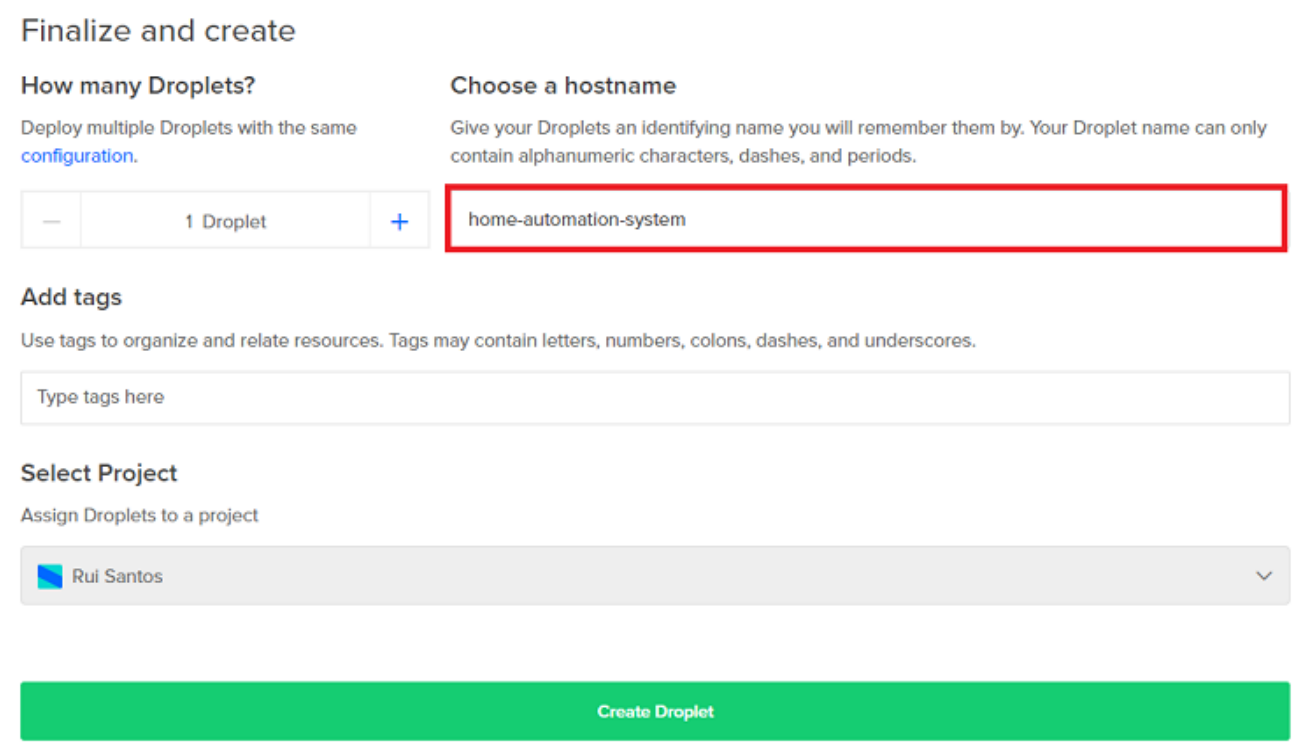


Then, you can select any additional options you think might be useful for your project.



Finally, choose a **hostname** to easily identify which Virtual Machine you are working with. I've named my Droplet `home-automation-system`.

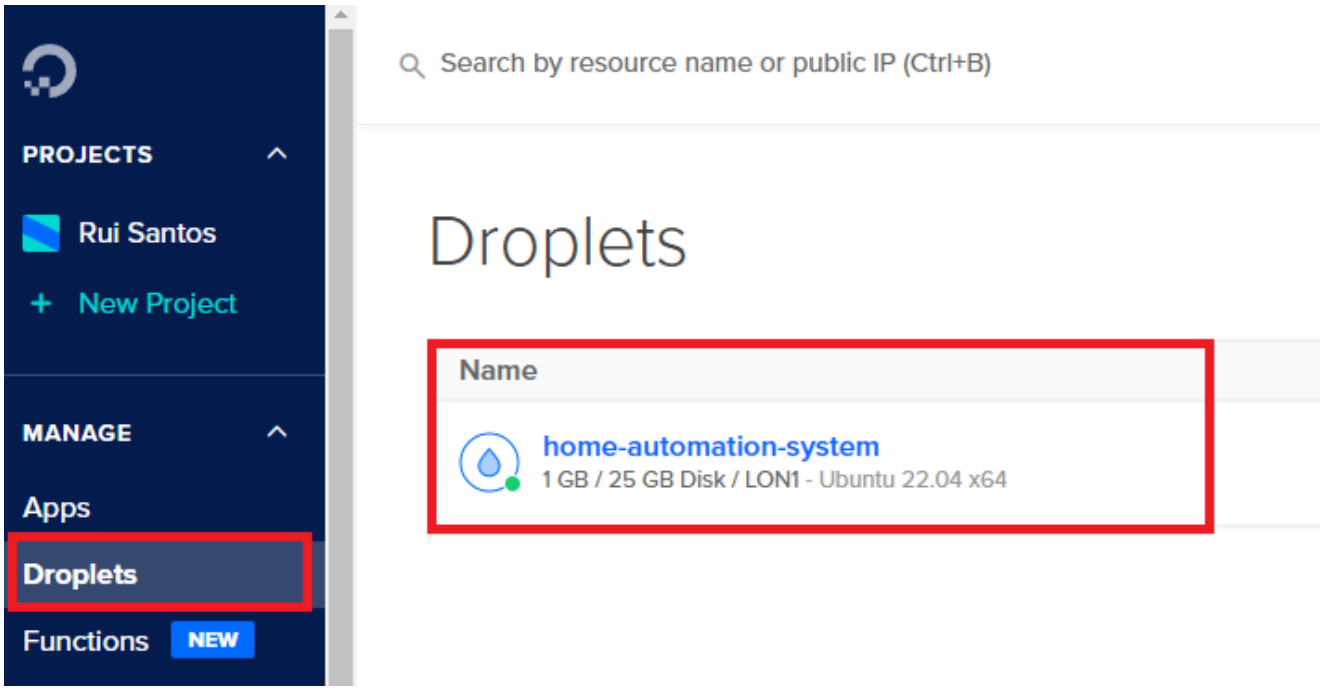
That's it, you just need to press the big green button **Create Droplet** to finish the process.



Wait a few minutes and when the progress bar ends, your Droplet is ready.

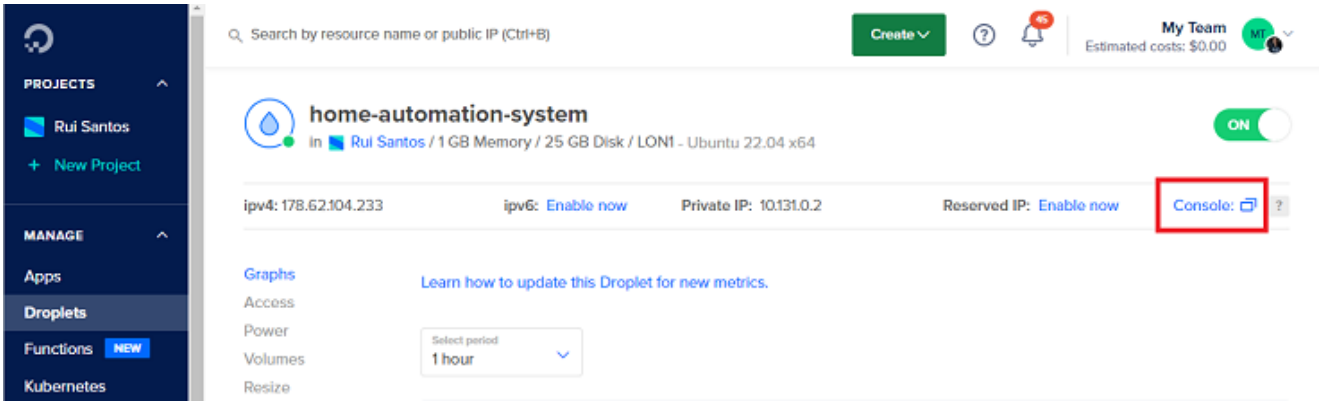
Accessing Your Linux Ubuntu VM Console

Now, if you click on the **Droplets** tab, your newly created droplet should be there.

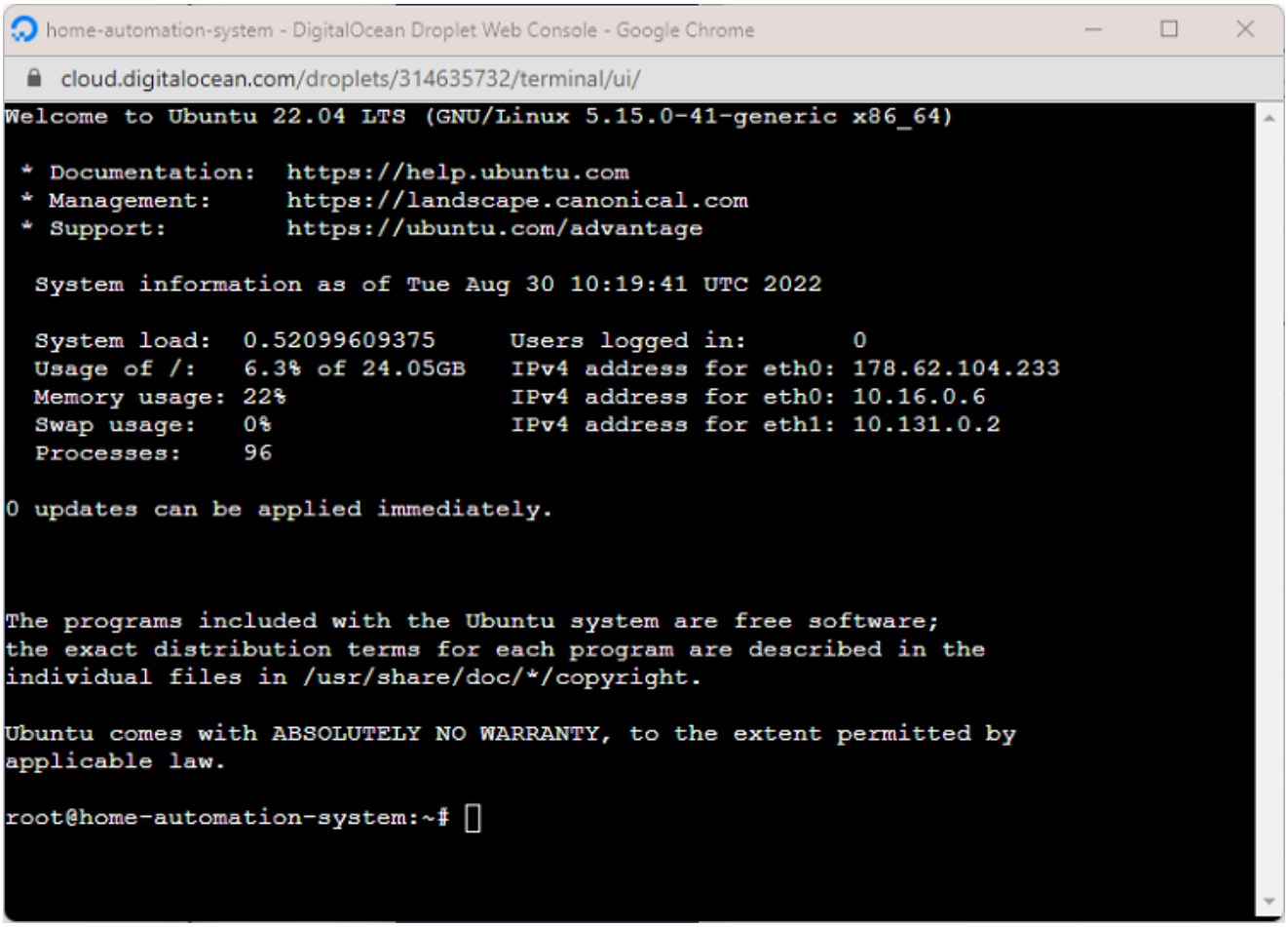


Click on the droplet name. A new page will open. At the top right corner, there's a Console link. If you click there, it will open a new console/terminal window where

you can type Linux commands to install software or run commands the same way you do on your Raspberry Pi via SSH.



Type your login username (`root`) and the password defined earlier, press the Enter key to access your server.



There's an ***optional*** step, but it goes beyond the scope of this tutorial. It's ***not*** required to make this project work: prepare your server with non-root, sudo-enabled user and basic firewall with this Initial Server Setup with Ubuntu 20.04.

Installing Mosquitto MQTT Broker on Linux Ubuntu VM Digital Ocean

Let's install the **Mosquitto Broker** on Digital Ocean.

1) Run the following command to upgrade and update your system:

```
sudo apt update && sudo apt upgrade -y
```

2) When asked, press **Y** and **Enter**. It will take some time to update and upgrade.

3) To install the Mosquitto Broker enter the next command:

```
sudo apt install -y mosquitto mosquitto-clients
```

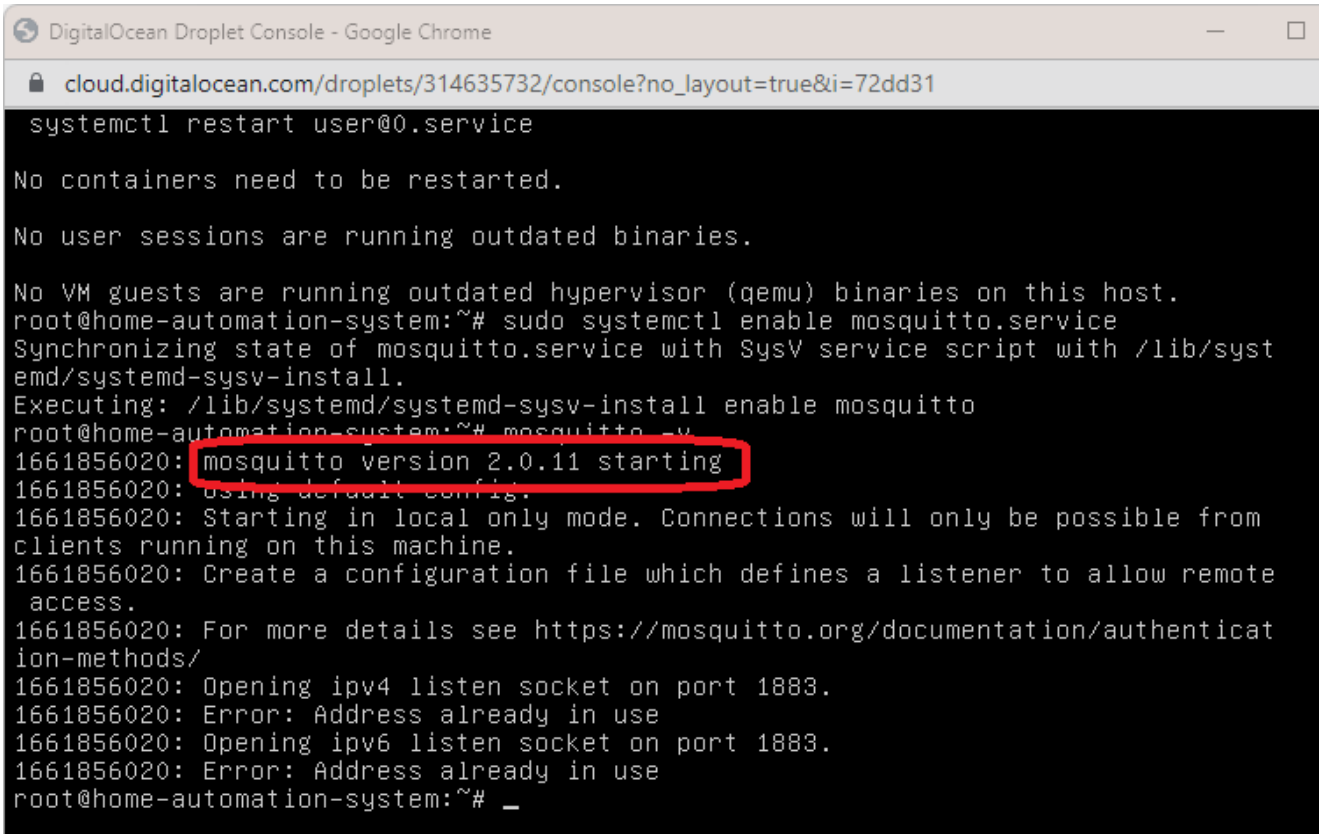
That's it! Mosquitto MQTT broker is installed.

4) To make Mosquitto auto start when the server boots, you need to run the following command (this step is optional, but it ensures that as long as the server is running, Mosquitto will be running even after a server restart):

```
sudo systemctl enable mosquitto.service
```

5) Now, test the installation by running the following command:

This returns the Mosquitto version that is currently running on your server. It will be 2.0.11 or above.



You can ignore the error message “Error: Address already in use”.

Enable Remote Access/ Authentication

To enable remote access so that we can communicate with IoT devices, we need to edit/create a configuration file.

We’ll add authentication with user and password.

1) Run the following command, but replace `YOUR_USERNAME` with the username you want to use:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd YOUR_USERNAME
```

I’ll be using the MQTT user `sara`, so I run the command as follows:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd sara
```

When you run the preceding command with the desired username, you'll be asked to enter a password. No characters will be displayed while you enter the password. Enter the password and memorize the user/pass combination, **you'll need it late**r in your projects to make a connection with the broker.

This previous command creates a password file called `passwd` on the `/etc/mosquitto` directory. Now, we need to edit the mosquitto configuration file so that it only allows authentication with the username and password we've defined.

2) Run the following command to edit the configuration file:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

3) Add the following line at the top of the file (make sure it is at the top of the file, otherwise it won't work):

```
per_listener_settings true
```

4) Also add the following three lines to allow connection for authenticated users and tell Mosquitto where the username/password file is located.

```
allow_anonymous false
listener 1883
password_file /etc/mosquitto/passwd
```

Your configuration file will look as follows (the new lines are in bold):

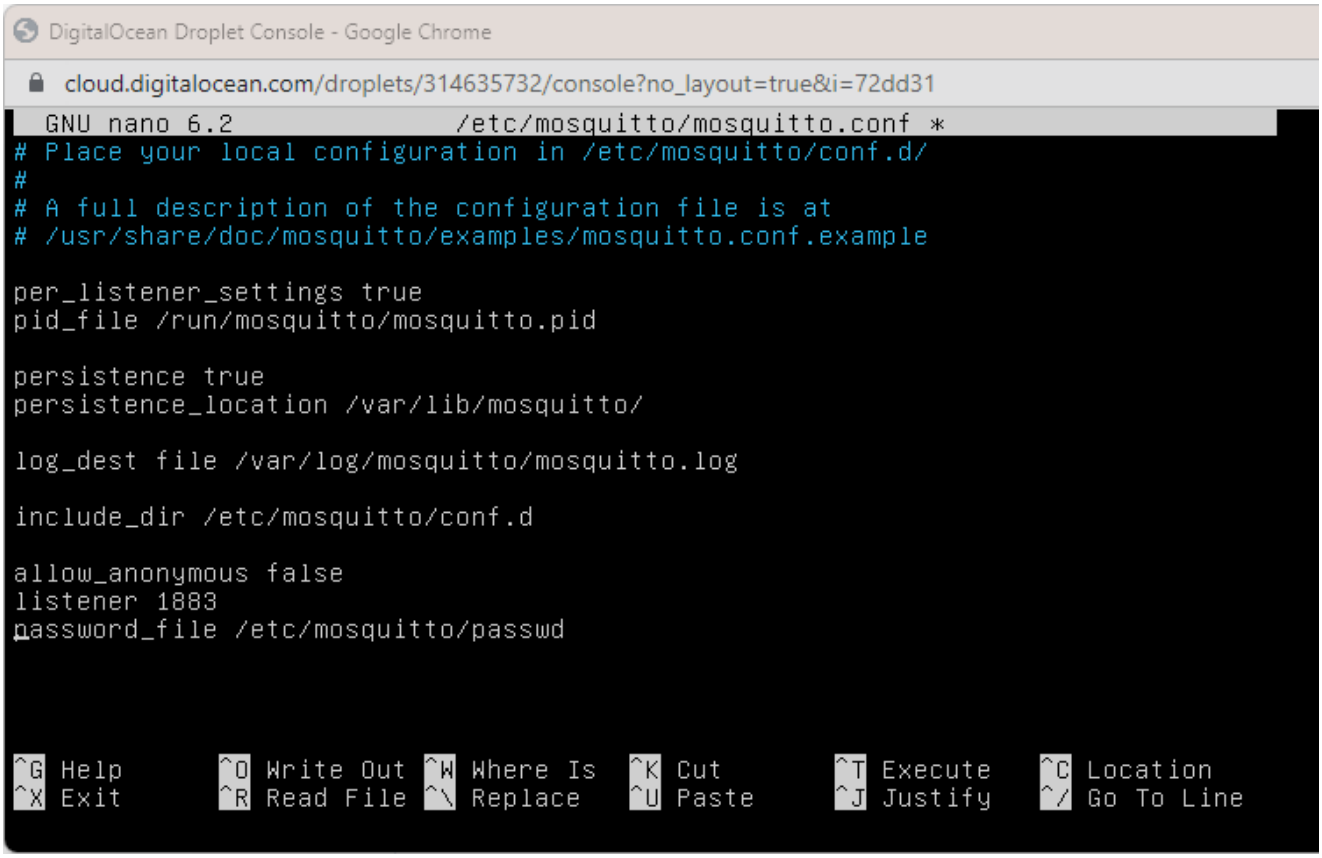
```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

per_listener_settings true

pid_file /run/mosquitto/mosquitto.pid
```

```
log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
allow_anonymous false
listener 1883
password_file /etc/mosquitto/passwd
```



5) Press **CTRL-X**, then **Y**, and finally press **Enter** to exit and save the changes.

6) Restart Mosquitto for the changes to take effect.

```
sudo systemctl restart mosquitto
```

7) Wait a few seconds. To check if Mosquitto is running, you can type the following command:

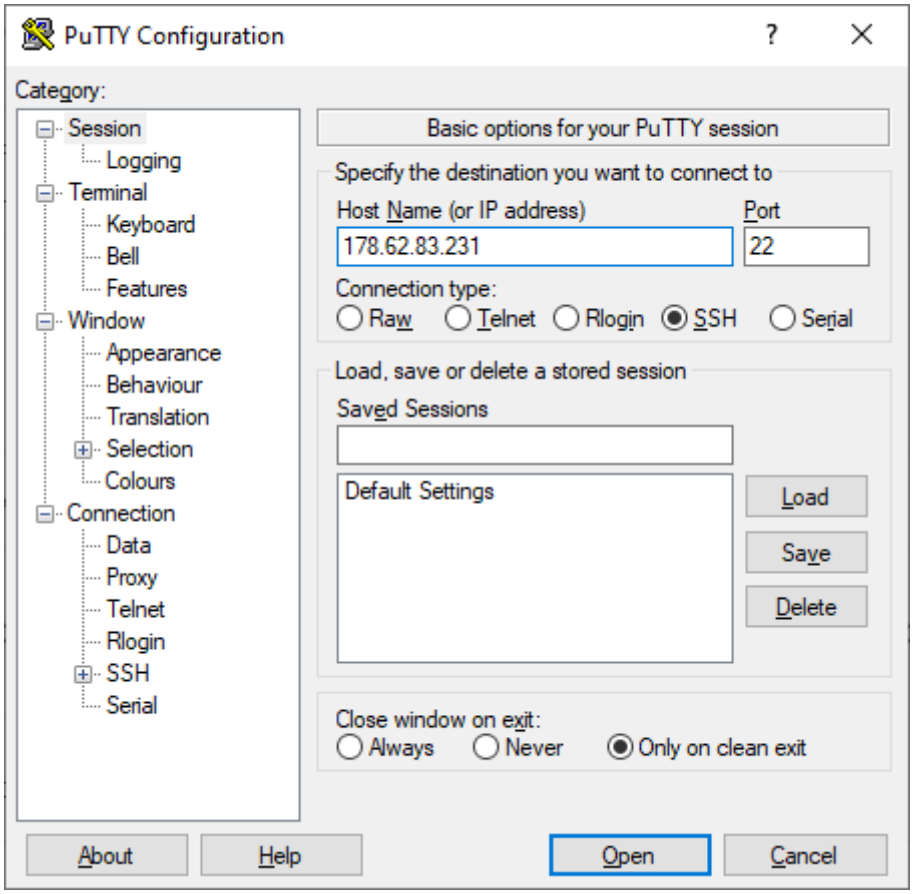
```
sudo systemctl status mosquitto
```

Now, you have Mosquitto MQTT broker installed on the cloud with authentication

On your ESP32/ESP8266 Arduino code, on the MQTT Host, you should use your droplet IP address.

Testing MQTT Mosquitto Broker Installation

To test your MQTT broker installation, you can use another terminal window (Terminal window #2) and establish an SSH communication with your server (you can use PuTTY or a similar SSH client). Enter the droplet IP address and try to establish an SSH connection.



Login as `root` and enter your password.

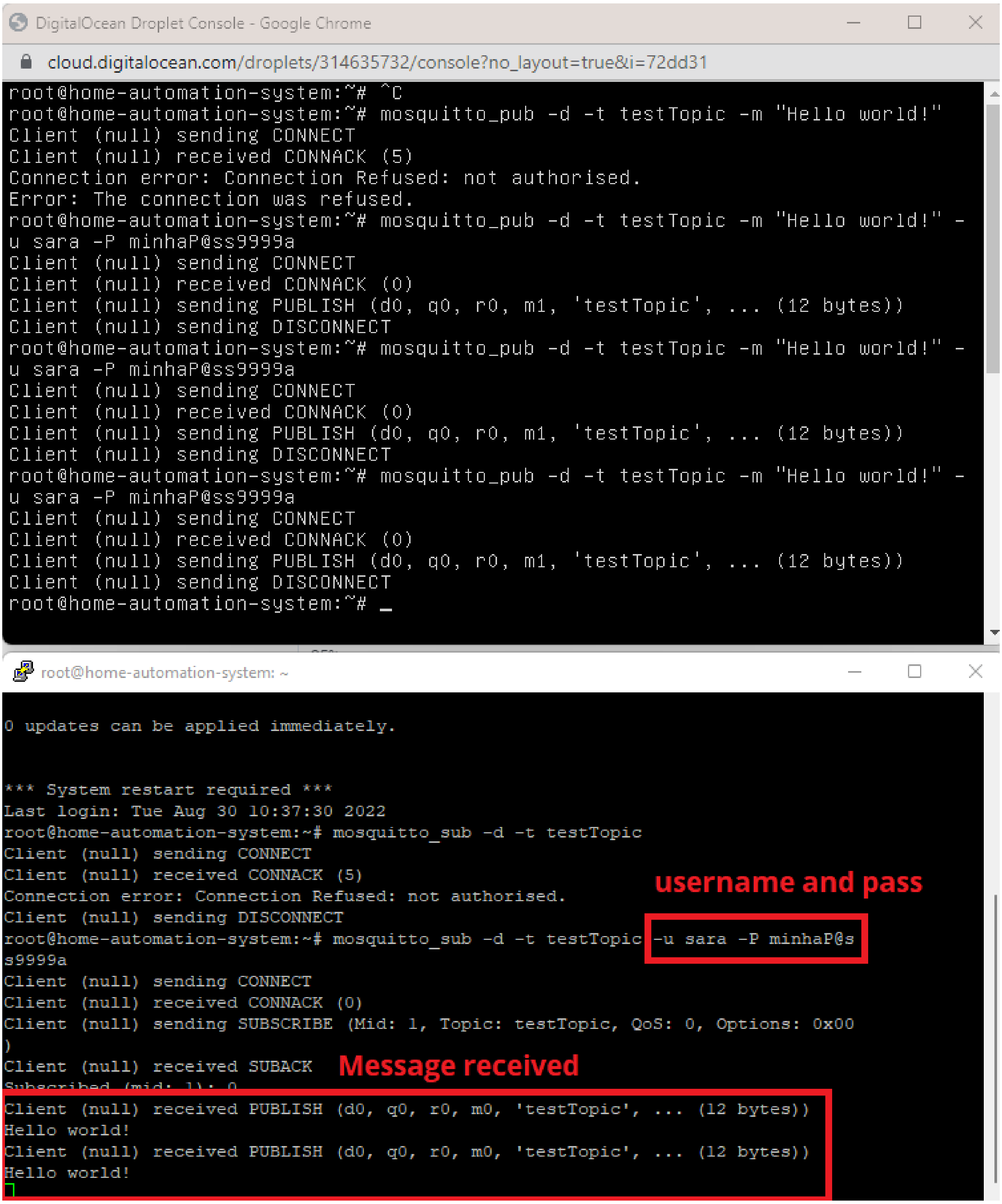
Then, enter the following command to subscribe to the `testTopic` topic. Replace `user` with your username and `pass` with your password.

```
mosquitto_sub -h localhost -t testTopic -u user -P pass
```

In your Terminal window #1, use the next command to publish the message "**Hello, world!**" in the `test` topic. Replace `user` with your username and `pass`

```
mosquitto_pub -h localhost -t testTopic -m "Hello, world!" -u user -P pass
```

Terminal window #2 should receive the message.



You can use the next table as a reference for the parameters you can pass in

-h	Hostname
-t	MQTT topic
-m	MQTT message
-u	MQTT username
-P	MQTT Password

Connecting Your ESP32 to MQTT Mosquitto Broker

We often use our ESP32 and ESP8266 boards in our MQTT projects. So, we'll show you how you can connect the ESP32 board to your Cloud MQTT Broker—it's the same for an ESP8266 board, just make sure you use the ESP8266 specific functions.

Before proceeding with this tutorial, make sure you complete the following prerequisites.

Arduino IDE

We'll program the ESP32 board using Arduino IDE, so make sure you have the ESP32 add-on installed.

- Installing the ESP32 Board in Arduino IDE (Windows, Mac OS X, Linux)

MQTT Libraries

To use MQTT with the ESP32 we'll use the Async MQTT Client Library.

Installing the Async MQTT Client Library

1. Click here to download the Async MQTT client library. You should have a .zip folder in your *Downloads* folder
2. Go to **Sketch** > **Include Library** > **Add .ZIP** library and select the library

Installing the Async TCP Library

To use MQTT with the ESP, you also need the Async TCP library.

1. Click here to download the Async TCP client library. You should have a .zip folder in your Downloads folder
2. Go to **Sketch** > **Include Library** > **Add . ZIP** library and select the library you've just downloaded.

ESP32 MQTT Publish Messages to Cloud MQTT Broker

Copy the following code to your Arduino IDE. To make it work for you, you need to insert your network credentials as well as the MQTT broker details (your Digital Ocean Droplet's IP Address, broker username and password).

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/c

  Permission is hereby granted, free of charge, to any person o
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall b
  copies or substantial portions of the Software.
*/

#include <WiFi.h>
extern "C" {
  #include "freertos/FreeRTOS.h"
  #include "freertos/timers.h"
}
#include <AsyncMqttClient.h>

#define WIFI_SSID "REPLACE_WITH_YOUR_SSID"
#define WIFI_PASSWORD "REPLACE_WITH_YOUR_PASSWORD"

// Digital Ocean MQTT Mosquitto Broker
```

```
// For a cloud MQTT broker, type the domain name
//#define MQTT_HOST "example.com"
#define MQTT_PORT 1883
```

View raw code

Type your network credentials on the following lines.

```
#define WIFI_SSID "REPLACE_WITH_YOUR_SSID"
#define WIFI_PASSWORD "REPLACE_WITH_YOUR_PASSWORD"
```

Insert the Digital Ocean Droplet IP address, so that the ESP32 connects to your broker (in my case, it is 178.62.83.231).

```
#define MQTT_HOST IPAddress(178, 62, 83, 231)
```

If your broker requires authentication, type your MQTT username and MQTT password.

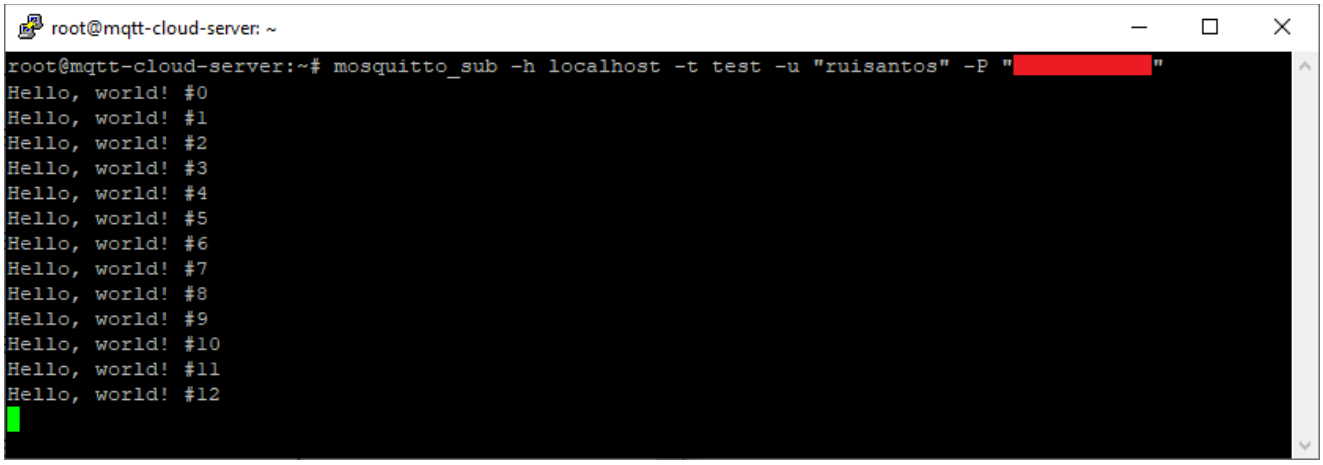
```
#define MQTT_USERNAME "YOUR_USER"
#define MQTT_PASSWORD "YOUR_PASSWORD"
```

Testing ESP32 MQTT Publishing Messages

If you have your ESP32 running the uploaded code and you open your Arduino IDE Serial monitor, you'll see that your ESP32 is publishing new messages every 5 seconds.

Establish an SSH connection with your cloud server (using PuTTY, for example) and type (replace `user` with your username and `pass` with your password.):

```
mosquitto_sub -h localhost -t test -u user -P pass
```



Cloud MQTT Broker Publish Messages to ESP32

The next sketch makes the ESP32 subscribe to a cloud MQTT topic to receive messages. Copy it to your Arduino IDE, then insert your network credentials as well as the MQTT broker details (your Digital Ocean Droplet's IP Address and the broker username and password).

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/c

  Permission is hereby granted, free of charge, to any person o
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall b
  copies or substantial portions of the Software.
*/

#include <WiFi.h>
extern "C" {
  #include "freertos/FreeRTOS.h"
  #include "freertos/timers.h"
}
#include <AsyncMqttClient.h>

#define WIFI_SSID "REPLACE_WITH_YOUR_SSID"
#define WIFI_PASSWORD "REPLACE_WITH_YOUR_PASSWORD"
```

```
#define MQTT_HOST IPAddress(XXX, XXX, XXX, XXX)
// For a cloud MQTT broker, type the domain name
//#define MQTT_HOST "example.com"
#define MQTT_PORT 1883
```

View raw code

Type your network credentials on the following lines.

```
#define WIFI_SSID "REPLACE_WITH_YOUR_SSID"
#define WIFI_PASSWORD "REPLACE_WITH_YOUR_PASSWORD"
```

Insert the Digital Ocean Droplet IP address, so that the ESP32 connects to your broker (in my case, it is 178.62.83.231).

```
#define MQTT_HOST IPAddress(178, 62, 83, 231)
```

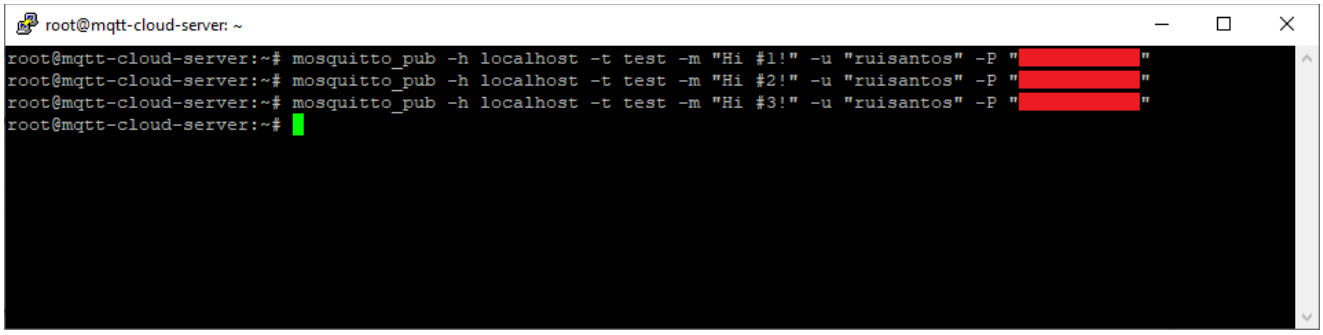
If your broker requires authentication, type your MQTT username and MQTT password.

```
#define MQTT_USERNAME "YOUR_USER"
#define MQTT_PASSWORD "YOUR_PASSWORD"
```

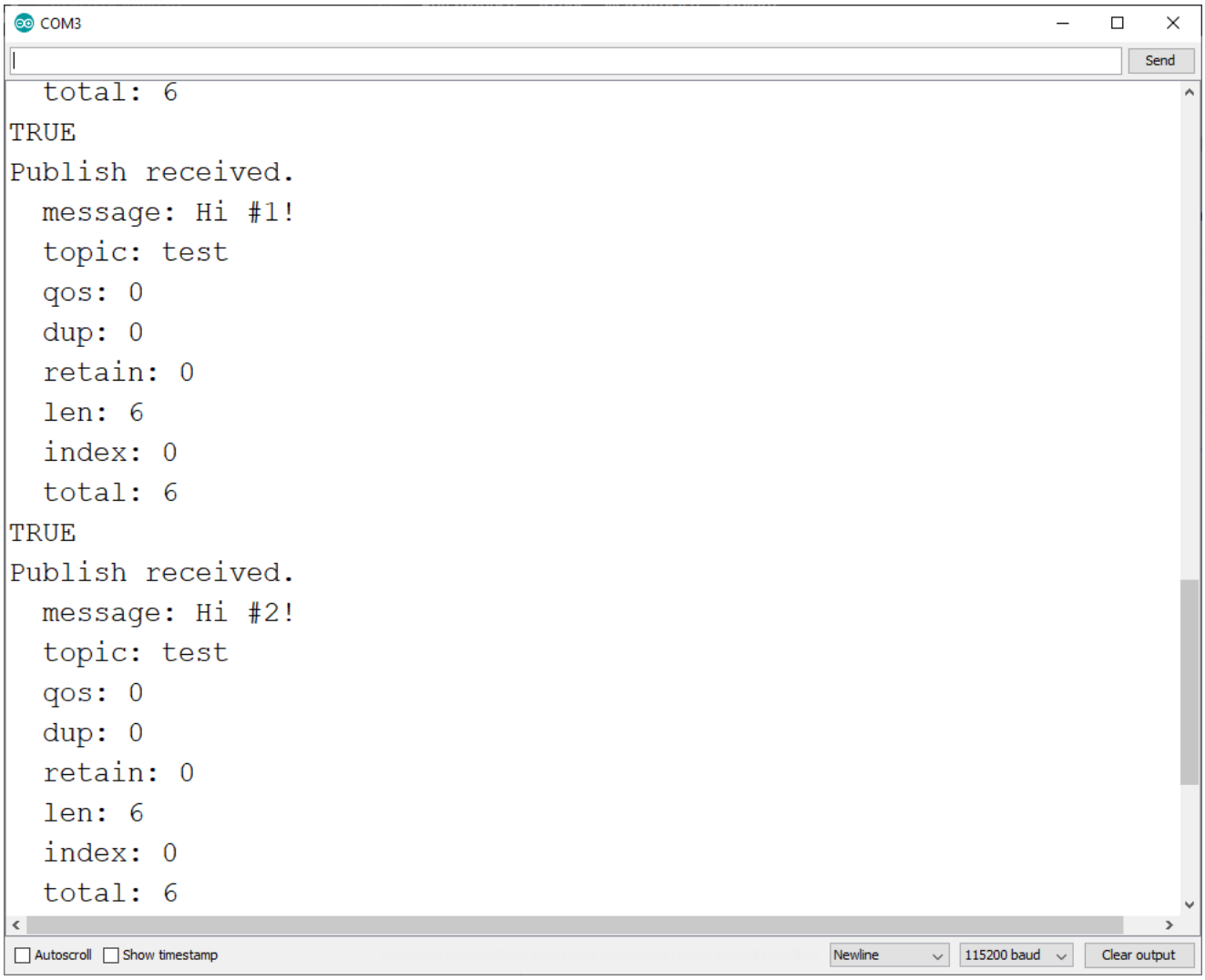
Testing ESP32 Subscribe to MQTT Topic

To test if your ESP32 is receiving MQTT messages, in your Digital Ocean console start publishing different messages (for example “**Hi #1!**“, “**Hi #2!**“, etc). Replace `user` with your username and `pass` with your password.

```
mosquitto_pub -h localhost -t test -m "Hi #1!" -u user -P pass
mosquitto_pub -h localhost -t test -m "Hi #2!" -u user -P pass
mosquitto_pub -h localhost -t test -m "Hi #3!" -u user -P pass
```



Your ESP32 should receive each message and print it in the Serial Monitor, as shown in the image below.



In these quick examples, we've shown you how to publish and subscribe MQTT messages using the Cloud MQTT broker. The idea is to use several ESP32 or ESP8266 boards that publish and subscribe to the same topics to communicate with each other and/or use Node-RED on the cloud to interact with those boards.

(Optional) Taking It Further – MQTT Mosquitto

The best method to add an SSL certificate to your server is by having a domain name pointed at your server and using Let's Encrypt certificates.

- You'll have to buy a domain name and point it to Digital Ocean Name Servers.
- You'll also need to follow these next instructions to prepare Certbot Standalone Mode to Retrieve Let's Encrypt SSL Certificates on Ubuntu (this guide also works with Ubuntu 20.04).

Having a domain name and Let's Encrypt SSL Certificates ready, follow the next instructions to secure your Mosquitto broker.

To enable SSL encryption, we need to tell Mosquitto where our Let's Encrypt certificates are stored. Open up the configuration file we previously started:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Add the next lines to make your default.conf add the Let's Encrypt certificates.

```
allow_anonymous false
password_file /etc/mosquitto/passwd

listener 1883 localhost

listener 8883
certfile /etc/letsencrypt/live/example.com/cert.pem
cafile /etc/letsencrypt/live/example.com/chain.pem
keyfile /etc/letsencrypt/live/example.com/privkey.pem
```

Listener 1883 is the standard unencrypted MQTT port. The localhost instructs Mosquitto to only bind this port to the localhost interface, so it's not longer accessible externally.

On the other hand, listener 8883 sets up an encrypted listener on port 8883. The next three lines point Mosquitto to the appropriate Let's Encrypt files to set up the encrypted connections.

Save and exit the file (Ctrl+X, Y, Enter key), then restart Mosquitto to update the

```
sudo systemctl restart mosquitto
```

Update the firewall to allow connections to port 8883.

```
sudo ufw allow 8883
```

Now, you subscribe to the `test` MQTT topic in the encrypted port (8883). Don't forget to replace example.com with your domain name in the subscribe and publish commands.

```
mosquitto_sub -h example.com -t test -p 8883 --capath /etc/ssl/certs/ -u
user -P pass
```

You can publish encrypted messages:

```
mosquitto_pub -h example.com -t test -m "Secure message" -p 8883 --capath
/etc/ssl/certs/ -u user -P pass
```

With this setup, you'll need to prepare your ESP32/ESP8266 to make encrypted MQTT requests on port 8883.

Wrapping Up

This complete guide was tested and it should work. There are many steps and they must be followed exactly as we describe in the right order. Otherwise, something might not work properly.

In all our guides and projects we always try to help if anyone gets stuck. However, in this particular case, there are so many steps that it can be tough to help you without having access to the server and testing it (of course, we don't have the resources to help everyone personally).

If you have any problem installing Mosquitto MQTT broker, preparing your Linux

service since 2015 and they always have an extremely helpful support team (or just use their Forum).

Now, if you want to install Node-RED on Digital Ocean, follow the next tutorial: Access Node-RED Dashboard from Anywhere using Digital Ocean.

If you like this type of project, make sure you take a look at our SMART HOME course, where you'll learn how to setup a home automation system using MQTT, Node-RED, InfluxDB, and much more:

- **SMART HOME with Raspberry Pi, ESP32, and ESP8266**

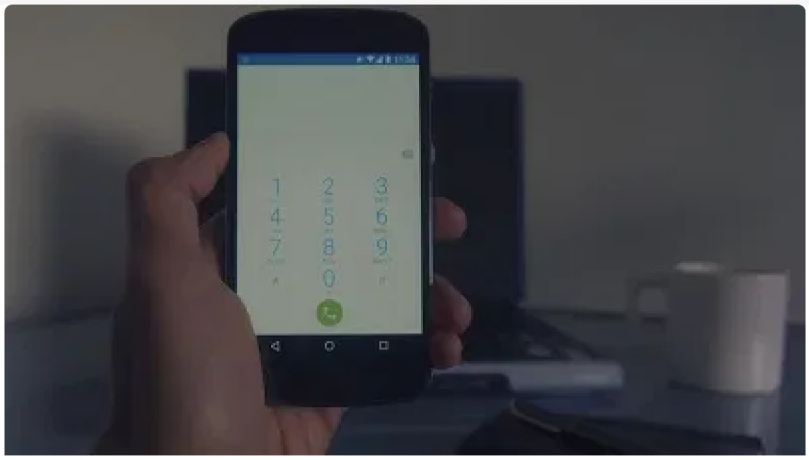
Read the next guides to learn more about MQTT:

- What is MQTT and How It Works
- ESP32 MQTT – Publish and Subscribe with Arduino IDE
- ESP32 MQTT – Publish BME280 Sensor Readings (Arduino IDE)

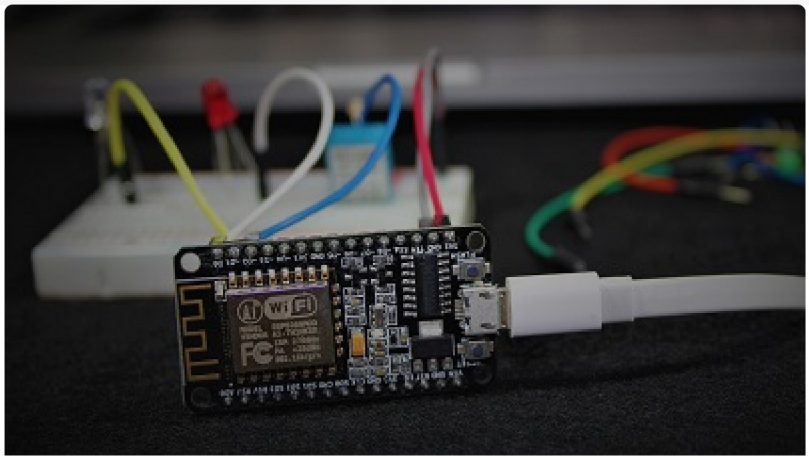
Thanks for reading.

communication protocols **DOWNLOAD »**

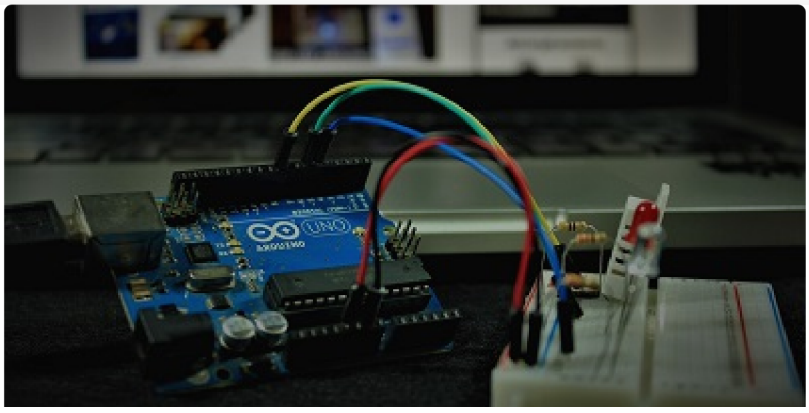
Recommended Resources



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.



Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



What to Read Next…

MicroPython: MQTT – Publish DS18B20 Temperature Readings (ESP32/ESP8266)

ESP32 with Stepper Motor (28BYJ-48 and ULN2003 Motor Driver)

ESP32/ESP8266: Firebase Web App to Display Sensor Readings (with Authentication)

23 thoughts on “Run Your Cloud MQTT Mosquitto Broker (access from anywhere using Digital Ocean)”

Nyacuma

June 28, 2020 at 7:17 pm

Rui you are a real clever man! How do you do all these?
Any way I struggled to make Cloud MQTT Mosquitto Broker at DigitalOcean during these Covid-19 scare period. I also succeeded in capturing my messages into a mysql database.
I borrowed heavily on your ‘esp-weather-station.php’! and others of your writings.

Reply

Rui Santos

June 29, 2020 at 10:03 am

I'm glad you find these guides useful! You can definitely run everything on DigitalOcean.

Reply

Dusan

June 28, 2020 at 8:23 pm

Good evening. That didn't work. I got into Digital Ocean, but no hundred dollar credits were offered to me, i paid five dollars. I stop at the "Launch Console", reports "incorrect login" (root). But today I'm done, continuing tomorrow. Good night.

Reply

Rui Santos

June 29, 2020 at 10:02 am

Please contact the DigitalOcean support team (in your account, there's a contact meny), they'll gladly help you access the console and see why the $100 credit is missing in your account. Regards,
Rui

ONG KHEOK CHIN

June 29, 2020 at 1:59 am

I have Bluehost hosting account. Is Bluehost provides Linux Ubuntu VM service ?

Reply

Rui Santos

June 29, 2020 at 10:04 am

Yes, but unfortunately I think Bluehost charges like $20/month for a Linux VM… (while with DigitalOcean you can use the $5/month plan).

Reply

David Williams

July 1, 2020 at 3:55 pm

Rui / Sara,
It would be useful if you presented a method for hosting the readers own mqtt type or linux service using an ESP32 as a web server and instructing how to create a DMZ on the users own router to save us having to fork-out for these expensive hosting services – to my pocket anything above £0 is out of my range. I beg borrow and steal (not really) to get ESP modules and components and just don't have the monthly surplus to afford these

Just a thought
Keep safe and Keep up the good work
D

Reply

Dhanavath Vishnu
July 6, 2020 at 10:48 am

Hello Rui/Sara,
It is glad to see this guide finally comes from you guys, I was waiting for this guide. I have implemented using your other guides to run Mosquitto MQTT Broker in my local network using a Raspberry Pi board and developed DHT22+ESP32 sensor to send data to the mosquito broker after the hosting it on Node-red Dashboard. Thanks for that.

Can we use this method of hosting Mosquitto on the digital ocean for a commercial project? will this method satisfy the IoT security requirement?

can we host the same mosquito broker on AWS? If so what might be the expected budget.

Do you have any guide of implementing DHT22+ESP32 integration and publishing the data to the cloud server over HTTPS? If you are having can you please share the link with me.

Thank you very much your help.

Reply

Rui Santos
July 8, 2020 at 9:04 am

I think so. Digital Ocean is a very reliable company that is used in many enterprises. The security mostly depends on how you configure your server.
You should be able to also do it on AWS, but I've never tried it before.
Thanks for asking!
Rui

Reply

allan
November 22, 2020 at 3:02 pm

i like to install Mosquitto MQTT on blue host and send data from Mosquitto MQTT to mongo atlas cloud . any plugin needed for it .

thanks

Reply

Amin Shahsavar
December 27, 2020 at 1:42 pm

Hi Rui, your excellent website has two different tutorials on how to connect ESP32s to the internet. This tutorial here is based on MQTT (using cloud-based MQTT and Node-Red), and another one based in HTTP GET (using cloud-based MySQL + PHP scripts), found here: https://randomnerdtutorials.com/control-esp32-esp8266-gpios-from-anywhere/

Maybe there is even more? I'm a little lost – what are the differences and which method do you recommend for which use case?

From my limited knowhow it looks quite the same; both will have a web page UI that is accessable from anywhere in the world and in it you can receive and send data to one or more ESP32s simultaneously?

Reply

James TI

February 14, 2021 at 2:34 pm

Great guide as always! Will you do a follow-up guide to the line you left hanging 'With this setup, you'll need to prepare your ESP32/ESP8266 to make encrypted MQTT requests on port 8883'?

Reply

Jim

March 1, 2021 at 6:34 am

Hi Rui, I purchased the Digital Ocean MQTT service and tried this tutorial. I was able to publish and subscribe through the putty and everything was ok until I tried to test it on ESP32. I keep getting the followings:
WiFi connected
IP address:
192.168.1.148
Connecting to MQTT…
Disconnected from MQTT.

Connecting to MQTT…
Disconnected from MQTT.
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Connecting to MQTT…
Disconnected from MQTT.
Connecting to MQTT…
Disconnected from MQTT.
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76

What is causing this problem?
Thank you for your response.

Reply

Rui Santos
March 3, 2021 at 10:30 am

Does your ESP32 have a stable internet connection? Is it fairly close to your router?

Reply

Moyu
May 1, 2021 at 5:49 pm

Hi Rui,

Thanks for the tutorial.

I am too facing same issue.

All steps worked within digitalocean local host. but cannot connect matt

Does any additional steps required for listening into 1883 port?

Output:
WiFi connected
IP address:
192.168.14.135
Connecting to MQTT…
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Disconnected from MQTT.
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Connecting to MQTT…
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76
Publishing on topic test at QoS 1, packetId: 0 Message: 1.76

Thanks

Reply

Moyu
May 1, 2021 at 6:33 pm

Hi Rui,

It works now.

I tried everything in my existing LAMP server, where it did not work.

Created new droplet same as your steps, then started working, Thanks!

ilker

July 16, 2021 at 4:41 pm

```
extern “C” {
#include “freertos/FreeRTOS.h”
#include “freertos/timers.h”
}
#include <AsyncMqttClient.h>

#define TINY_GSM_MODEM_SIM800 // Modem is SIM800L
#define SerialMon Serial
#define SerialAT Serial1
#define TINY_GSM_DEBUG SerialMon

const char apn[] =”internet”;//internet
const char gprsUser[] =””;
const char gprsPass[] =””;

#include <Wire.h>
#include <TinyGsmClient.h>

#ifdef DUMP_AT_COMMANDS
#include <StreamDebugger.h>
StreamDebugger debugger(SerialAT, SerialMon);
TinyGsm modem(debugger);
#else
TinyGsm modem(SerialAT);
#endif

TinyGsmClient client(modem);

// Digital Ocean MQTT Mosquitto Broker
```

```
#define MQTT_USERNAME “REPLACE_WITH_YOUR_MQTT_USER”
#define MQTT_PASSWORD
“REPLACE_WITH_YOUR_MQTT_PASSWORD”

// Test MQTT Topic
#define MQTT_PUB_TEST “led”

// TTGO T-Call pins
#define MODEM_RST 5
#define MODEM_PWKEY 4
#define MODEM_POWER_ON 23
#define MODEM_TX 27
#define MODEM_RX 26
#define I2C_SDA 21
#define I2C_SCL 22

TwoWire I2CPower = TwoWire(0);

#define IP5306_ADDR 0x75
#define IP5306_REG_SYS_CTL0 0x00

bool setPowerBoostKeepOn(int en){
I2CPower.beginTransmission(IP5306_ADDR);
I2CPower.write(IP5306_REG_SYS_CTL0);
if (en) {
I2CPower.write(0x37); // Set bit1: 1 enable 0 disable boost keep on
} else {
I2CPower.write(0x35); // 0x37 is default reg value
}
return I2CPower.endTransmission() == 0;
}

AsyncMqttClient mqttClient;
TimerHandle_t mqttReconnectTimer;

void connectToMqtt() {
Serial.println(“Connecting to MQTT…”);
```

```
void onMqttConnect(bool sessionPresent) {
Serial.println("Connected to MQTT.");
Serial.print("Session present: ");
Serial.println(sessionPresent);
}

void onMqttDisconnect(AsyncMqttClientDisconnectReason reason) {
Serial.println("Disconnected from MQTT.");

xTimerStart(mqttReconnectTimer, 0);

}

/void onMqttSubscribe(uint16_t packetId, uint8_t qos) {
Serial.println("Subscribe acknowledged.");
Serial.print(" packetId: ");
Serial.println(packetId);
Serial.print(" qos: ");
Serial.println(qos);
}
void onMqttUnsubscribe(uint16_t packetId) {
Serial.println("Unsubscribe acknowledged.");
Serial.print(" packetId: ");
Serial.println(packetId);
}/

void onMqttPublish(uint16_t packetId) {
Serial.print("Publish acknowledged.");
Serial.print(" packetId: ");
Serial.println(packetId);
}

void setup() {
SerialMon.begin(115200);
delay(10);

I2CPower.begin(I2C_SDA, I2C_SCL, 400000);
```

```
bool isOk = setPowerBoostKeepOn(1);
SerialMon.println(String("IP5306 ") + (isOk ? "OK" : "FAIL"));

pinMode(MODEM_PWKEY, OUTPUT);
pinMode(MODEM_RST, OUTPUT);
pinMode(MODEM_POWER_ON, OUTPUT);
digitalWrite(MODEM_PWKEY, LOW);
digitalWrite(MODEM_RST, HIGH);
digitalWrite(MODEM_POWER_ON, HIGH);

SerialAT.begin(115200, SERIAL_8N1, MODEM_RX, MODEM_TX);
delay(6000);

modem.restart();
//modem.init();

String modemInfo = modem.getModemInfo();
SerialMon.print("Sim : ");
SerialMon.println(modemInfo);

if(!modem.gprsConnect(apn, gprsUser, gprsPass))
SerialMon.println("error!!!!");
else
SerialMon.println("success…..");

if (modem.isGprsConnected()) {
SerialMon.println("GPRS success!!");
}

mqttReconnectTimer = xTimerCreate("mqttTimer",
pdMS_TO_TICKS(2000), pdFALSE, (void*)0,
reinterpret_cast(connectToMqtt));

mqttClient.onConnect(onMqttConnect);
mqttClient.onDisconnect(onMqttDisconnect);
/mqttClient.onSubscribe(onMqttSubscribe);
mqttClient.onUnsubscribe(onMqttUnsubscribe);/
```

```
// If your broker requires authentication (username and password), set them below
//mqttClient.setCredentials(MQTT_USERNAME, MQTT_PASSWORD);

}

void loop() {

}
```

I use ESP32 SIM800L GPRS is connecting but xTimerCreate does not work and when I invoke mqtt.connect() function manually ,it reboots.

Reply

Redhill

August 16, 2021 at 8:01 am

Hi Rui,
Can I use ESP8266 (instead of ESP32) to access this mqtt server? Will there be certificate issue? Which mqtt library to use? Any working example ?
Thanks

Reply

Joao Pedro

November 28, 2021 at 5:25 pm

Hello Rui

My question is if it's possible for the VM to have the MQTT broker, Node-red and InfluxDB on the same machine.
My only goals is to have a database that i could export the .csv eventually and a Real-Time Dashboard. I was thinking about a Node-Red Flow that would get the Mqtt data and update the Dashboard and write to the InfluxDB in Parallel. Is it Possible?

Reply

ilker

February 16, 2022 at 1:13 pm

Hi,How can we open web socket with mosquitto? I use mosca and https://unpkg.com/mqtt@3.0.0/dist/mqtt.min.js var client = mqtt.connect(address); with this way it works for mosca but it does not work for mosquitto .What should I do for mosquitto?

Reply

Juan

March 23, 2022 at 8:10 pm

Hi Rui, can this project work with ttgo Lora esp32 boards?

Greetings

Reply

Hi.
Yes, you can replace those “regular” ESP32 boards with any other ESP32 board model.
Regards,
Sara

Reply

ilker
April 26, 2022 at 6:36 am

Hi,How can we open web socket with mosquitto? I use mosca and https://unpkg.com/mqtt@3.0.0/dist/mqtt.min.js var client = mqtt.connect(address); with this way it works for mosca but it does not work for mosquitto .What should I do for mosquitto?

Reply

Leave a Comment

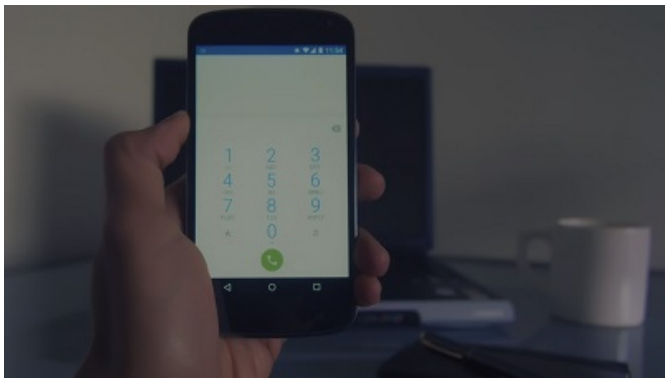
Email *

Website

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Post Comment



Build Web Servers with ESP32 and ESP8266 » boards to control outputs and monitor sensors remotely.