How to Connect Your Industrial Assets to AWS IoT Using N3uron's MQTT Module

Connecting AWS IoT: Overview

As stated in our previous article, MQTT: The Universal Messaging Protocol for Cloud Providers and IIoT Systems, MQTT has emerged as the defacto standard for IIoT and of course, is also supported by AWS IoT. OT infrastructure can be connected to AWS IoT Core as well as AWS IoT Greengrass Core using MQTT, enabling access to the whole ecosystem of services currently provided by AWS.

In short, AWS IoT Core is the service that receives and routes MQTT messages from edge devices and applications such as N3uron This guide explains in detail how to communicate your industrial assets bi-directionally with AWS IoT Core in a secure way by means of N3uron's MQTT module and thus, bridge the gap between OT and IT.

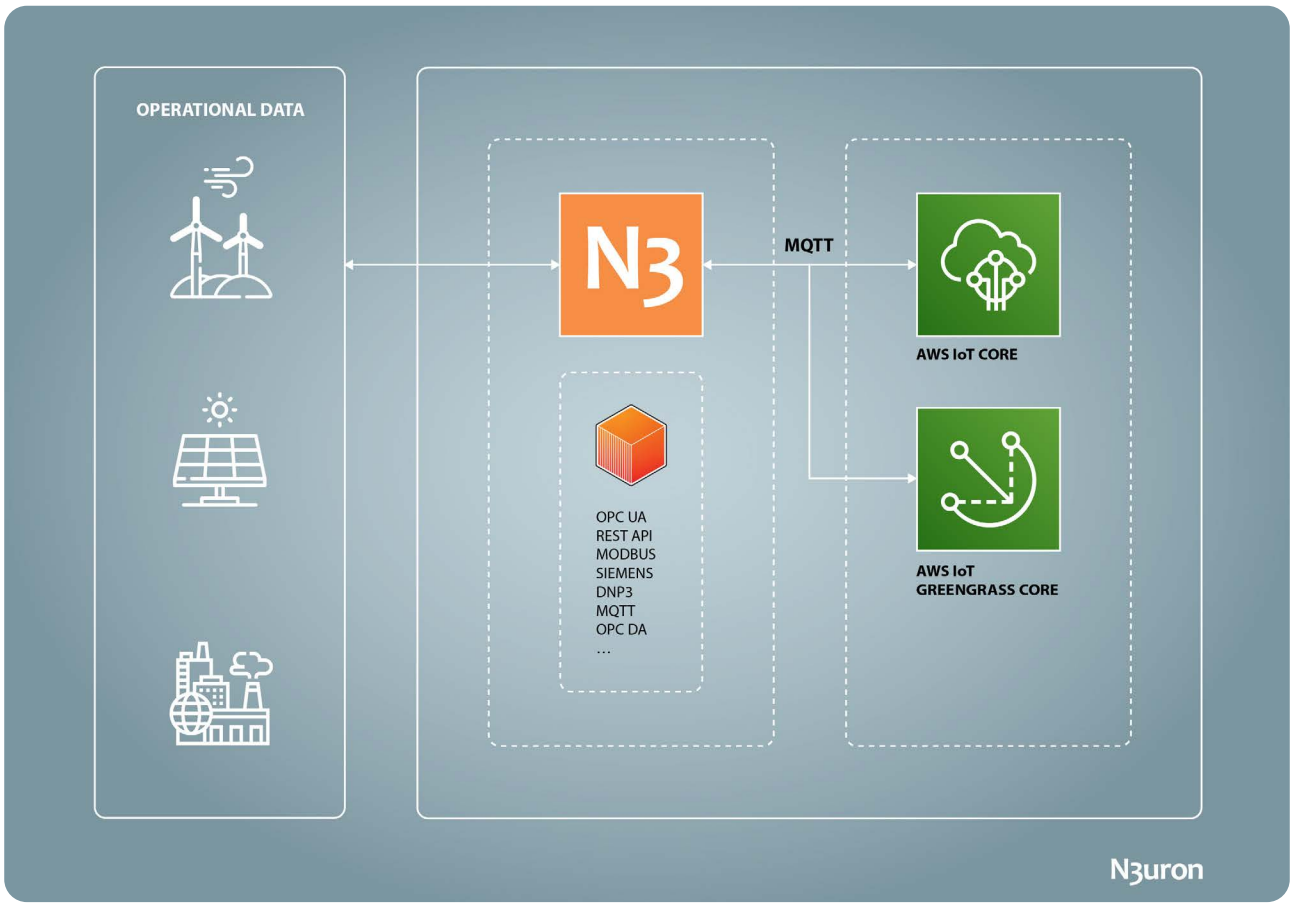


Diagram displaying operational data exchange between OT assets and Amazon IoT using N3uron IIoT protocols

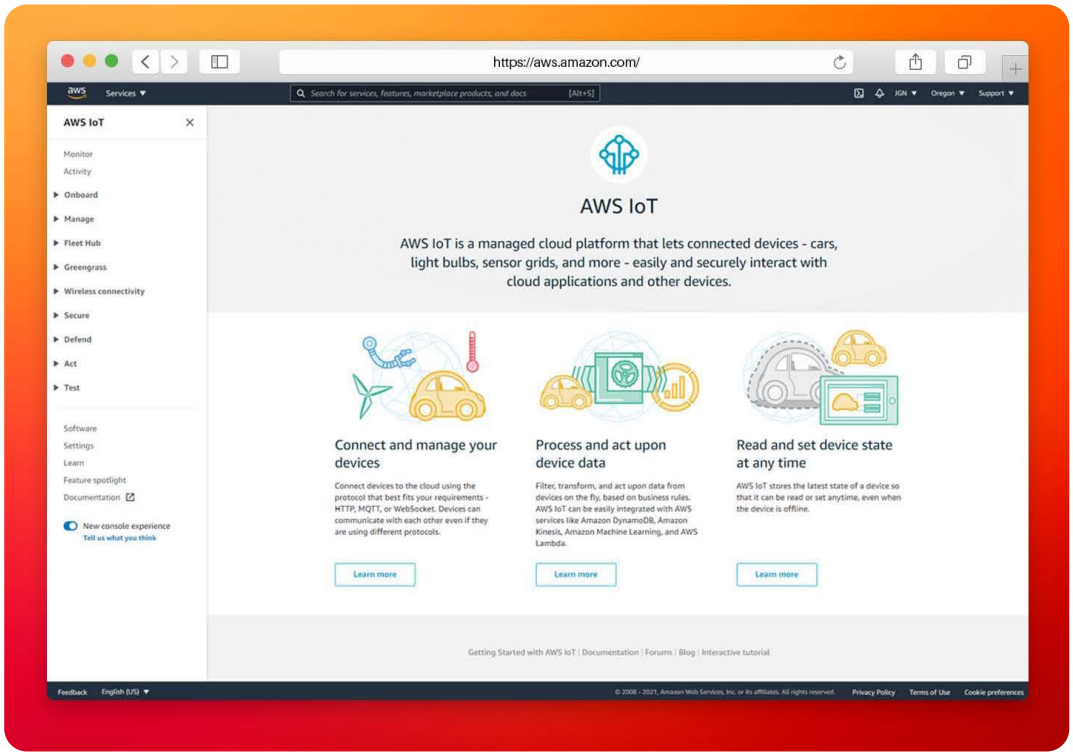
N3uron and AWS IoT Requirements

It is assumed that you already have an AWS account. If not, you can create one at https://aws.amazon.com. If you haven't downloaded N3uron yet, you can do so at https://n3uron.com/downloads/. If this is the first time installing N3uron, our Quick User Guide will guide you through the entire installation process.

Configuring AWS IoT Core

Log Into Amazon and Open AWS IoT Console

Click on the link to get to the AWS IoT console.



Screenshot displaying the Amazon IoT platform console panel

Once logged in, you'll create the AWS IoT resources that a device will require in order to connect to AWS IoT and exchange messages.

Create a Policy using the AWS IoT Console

This policy will authorize your device to interact with AWS IoT services.

Certificates are used to authenticate your device with AWS IoT Core. AWS IoT policies are attached to the certificate authenticating the device to determine the AWS IoT operations, such as subscribing or publishing to MQTT topics that this device is permitted to perform. The device will present its certificate whenever it connects and sends messages to the AWS IoT Core.

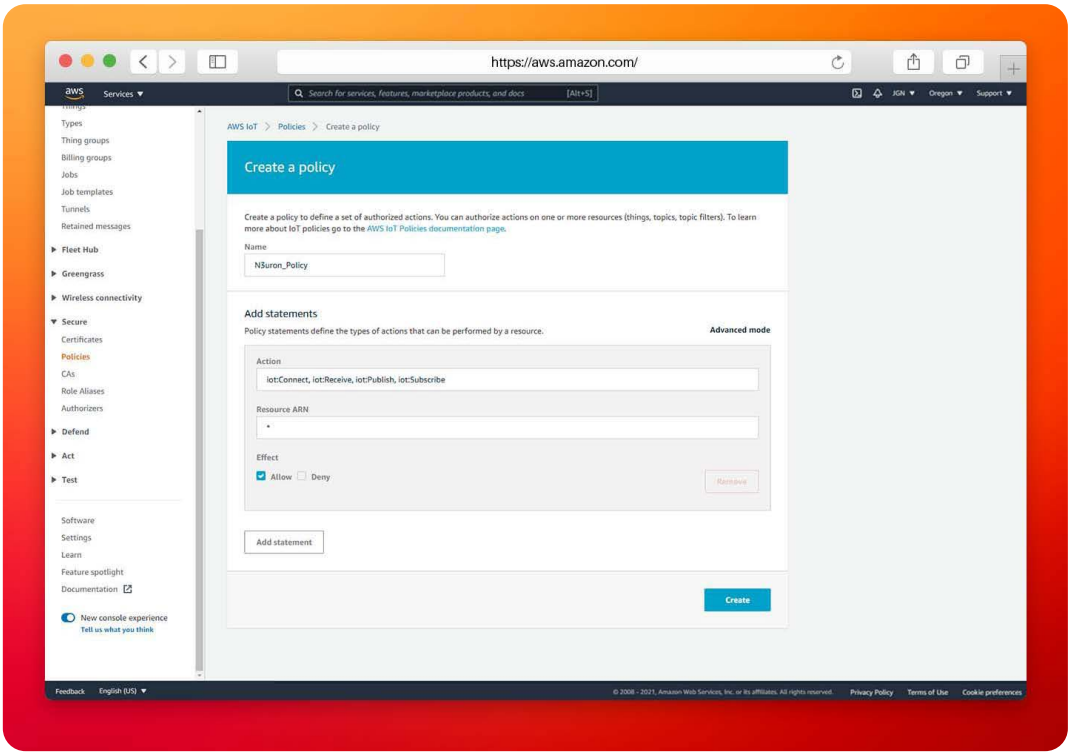
The following procedure will create a policy that allows your device to perform the AWS IoT operations necessary for this example. You must create the AWS IoT policy first, which will then allow you to attach it to the device certificate that you will be creating later.

- **Step 01:** Within the AWS IoT console, in the left-hand menu, first select **Secure**, and then **Policies**. On the **You don't have a policy yet** page, choose **Create a policy**. If your account has existing policies, choose Create.
- **Step 02:** On the **Create a Policy** page:
  A: In the Name field, enter a name for the policy (for example, N3uron_Policy).
  B: In the Action field, enter iot:Connect, iot:Receive, iot:Publish, iot:Subscribe. These are the actions that the device will need permission to perform.
  C: In the **Resource ARN** field, enter *. This selects any client (device). For increased security, it's highly recommended that access is restricted by specifying a client **ARN** (Amazon resource name) once your **Thing** has been created.

D: Select the **Allow** check box. These values allow all clients that have this policy attached to their certificate to perform the actions listed in the **Action** field.

– **Step 03:** After you have entered the information for your policy, choose **Create**.

For more information, see IAM policies.

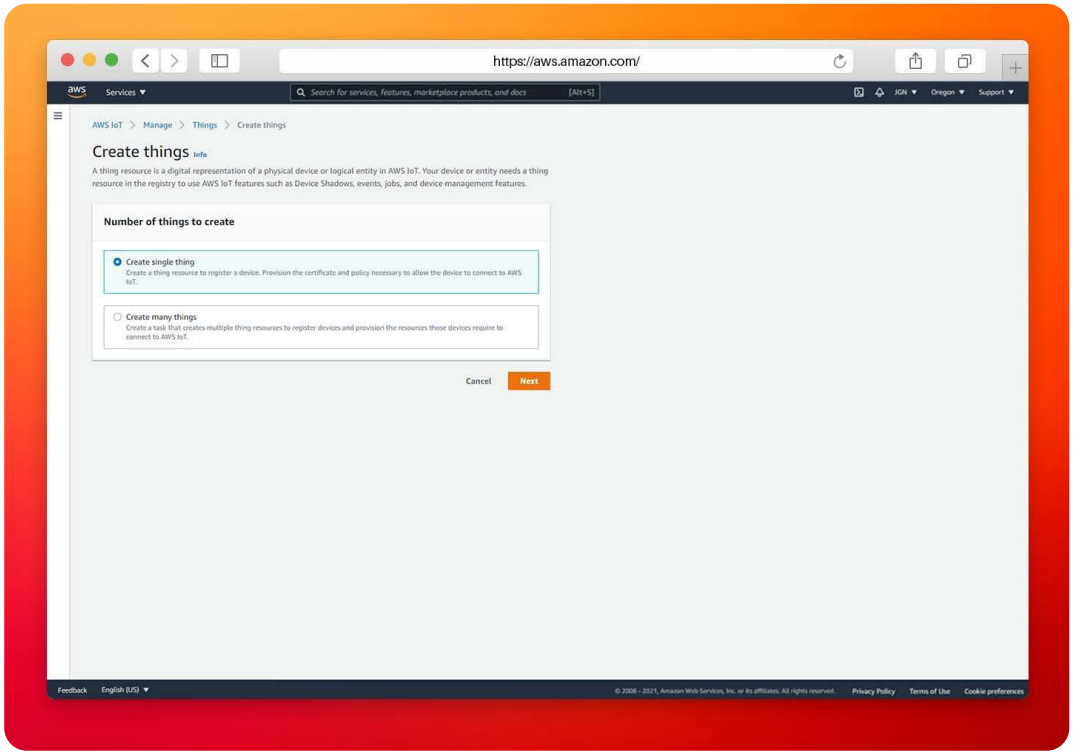


Screenshot displaying the “create policy” panel in the Amazon IoT platform

Create a Thing in the AWS IoT Console

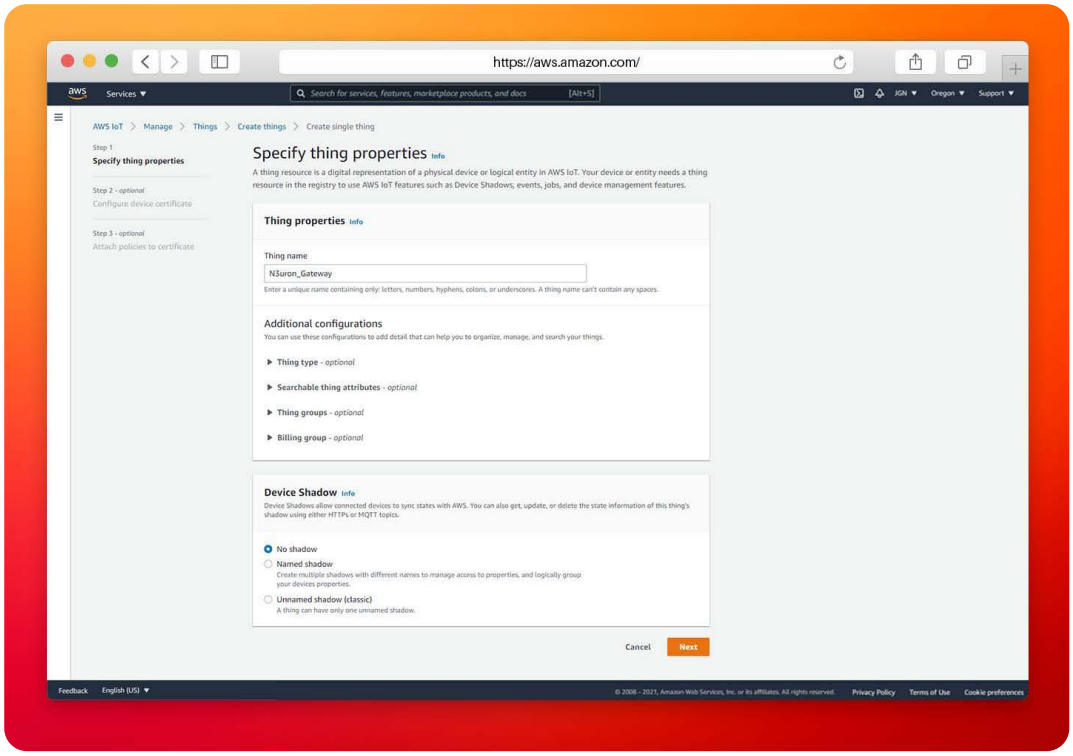
Devices connected to AWS IoT are represented by Thing objects in the AWS IoT registry. A Thing object represents a specific device or logical entity.

– **Step 01:** Within the AWS IoT console, in the left-hand menu, select **Manage**, then choose **Things**.
– **Step 02:** On the **Things** page, select **Create Things**.
– **Step 03:** On the **Create Things** page, select **Create a single thing**, then select **Next**.



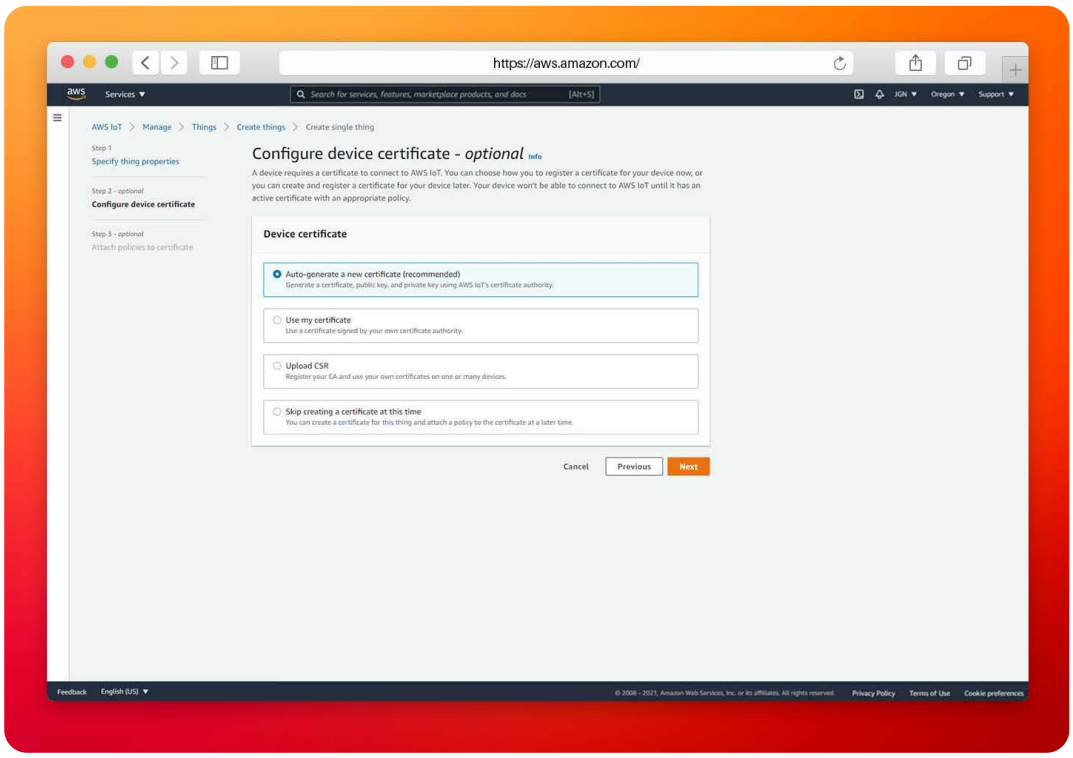
Screenshot displaying the “create things” panel in the Amazon IoT platform

- **Step 04:** On the **Specify thing properties** page, for **Thing name**, enter a name for your Thing. In this example, we have named it “N3uron_Gateway”. When naming Things, it’s important to choose the name carefully because a thing name cannot be changed after it has been created.
- **Step 05:** Leave the rest of the fields on this page empty. Select **Next**.



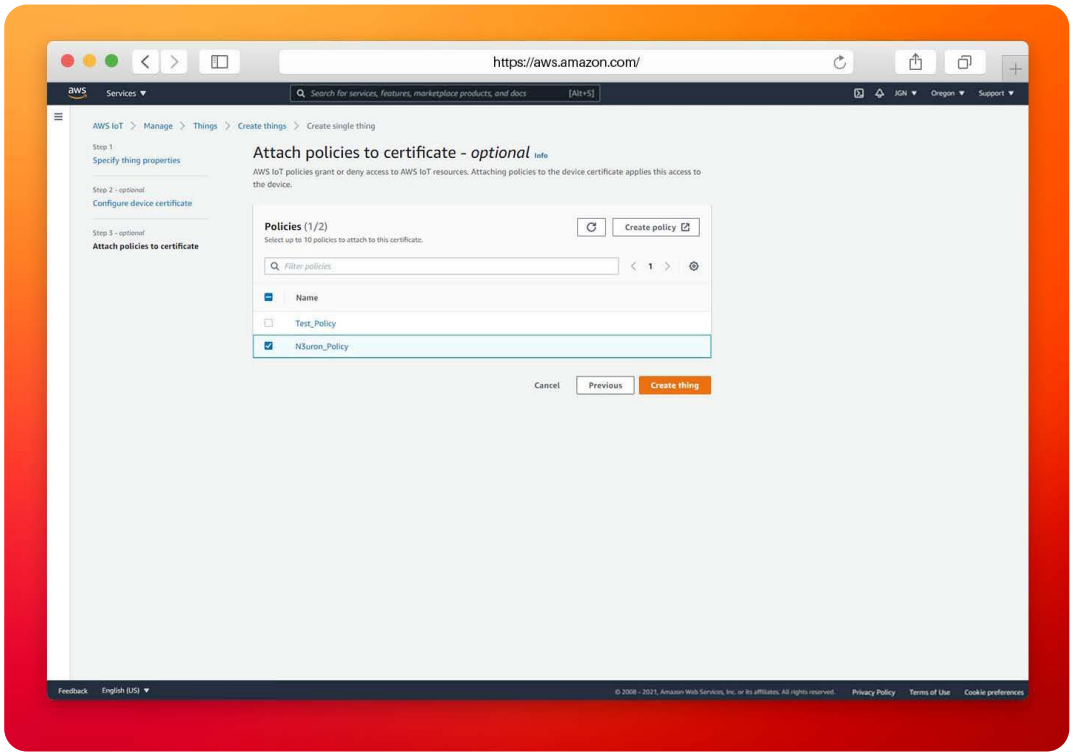
Screenshot displaying the “specify thing properties” panel in the Amazon IoT platform

– **Step 06:** On the **Configure device certificate – optional** page, select **Auto-generate a new certificate (recommended)**. Select **Next**.



Screenshot displaying the “configure device certificate” panelinf the Amazon IoT platform

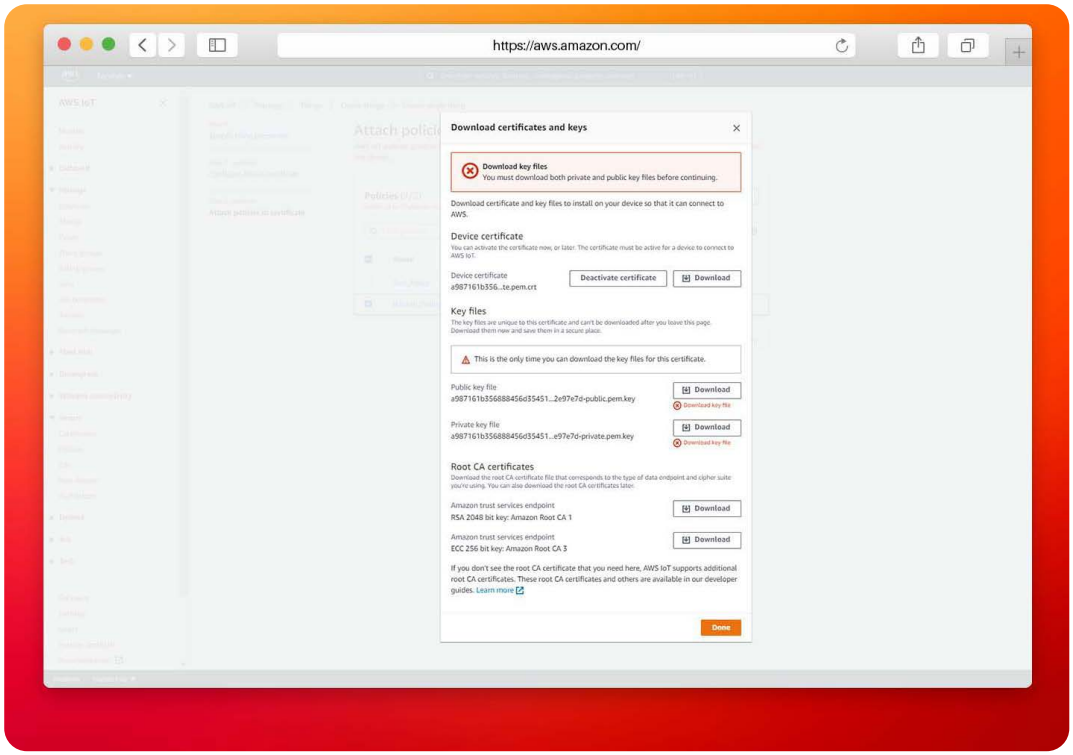
– **Step 07:** On the **Attach policies to certificate – optional** page, select the policy you created in the previous section. In the previous section, this policy was named, N3uron_Policy. Choose **Create thing**.



Screenshot displaying the “attach policies” panel in the Amazon IoT platform

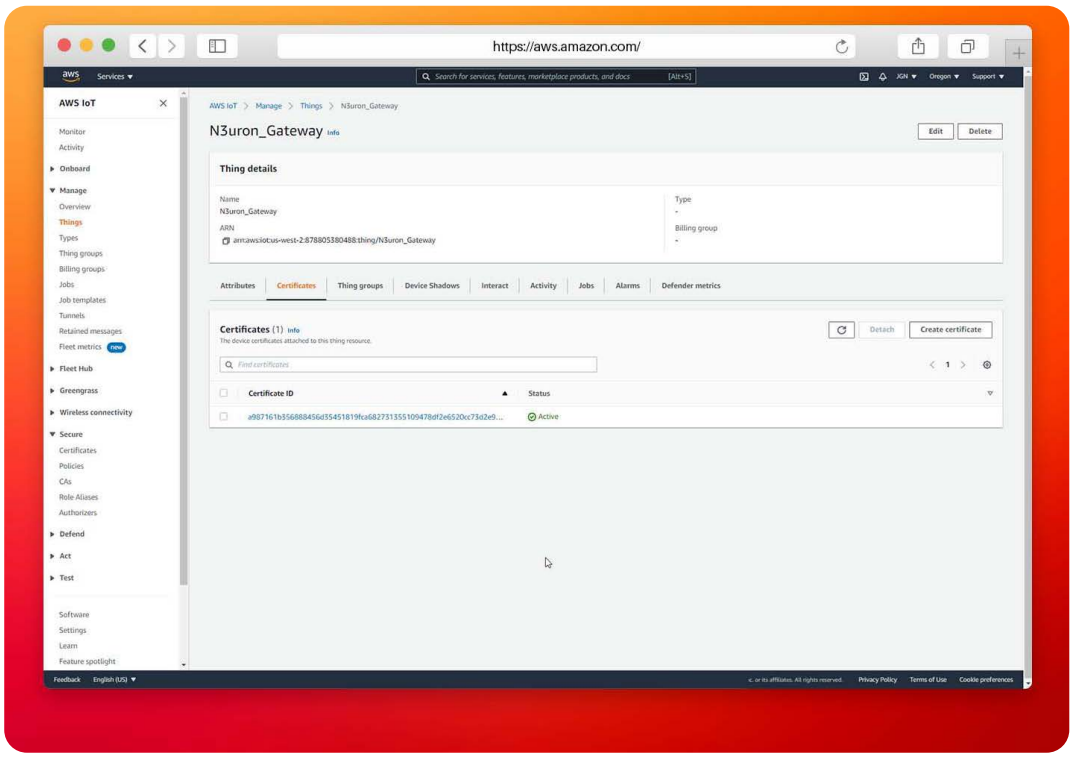
- **Step 08:** On the **Download certificates and keys** page:
  A: Download each of the certificates and key files and save them for later. You'll need to install these files on your device. See below for the required files:
  - Private key.
  - Public key.
  - Device certificate.
  - Root CA certificate.

  B: Download the **Amazon root CA 1**.
  C: Choose **Done**.



Screenshot displaying the "download certificates and keys" window in the Amazon IoT platform

After this procedure is complete, you should be able to see the new thing object in your list of Things. Click on the Thing you've just created, N3uron_Gateway, select the **Certificates** tab, and make sure the certificate is active.

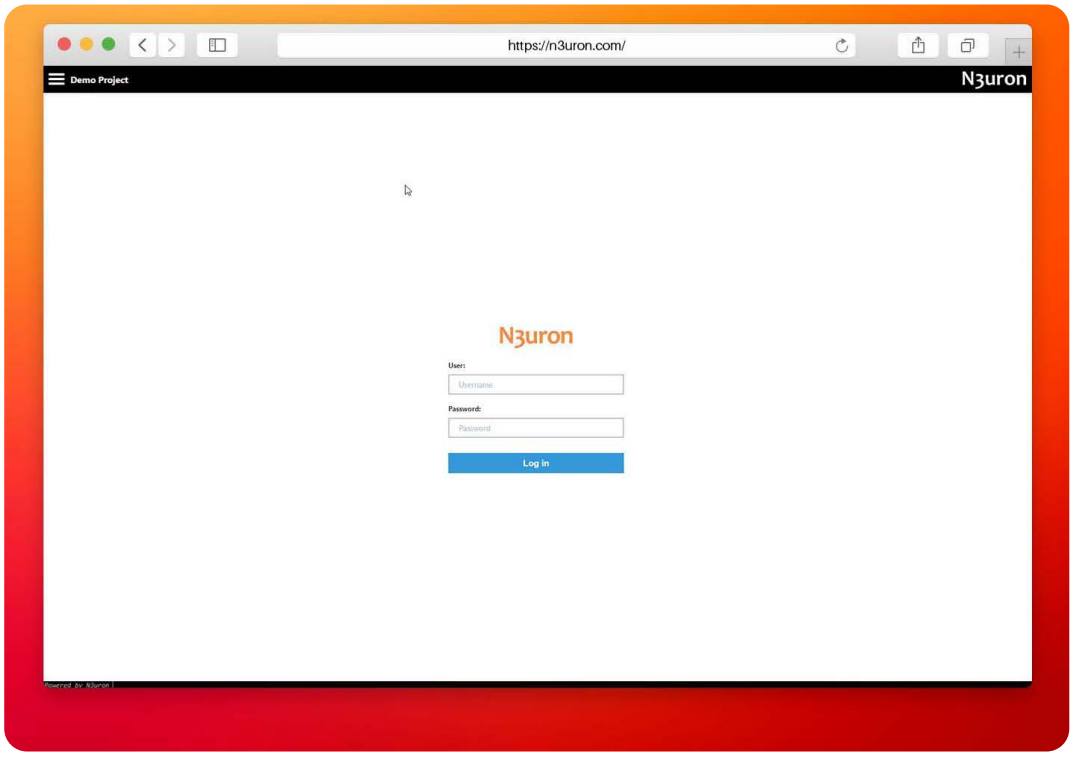


Screenshot displaying the “Thing details” panel in the Amazon IoT platform

Configure N3uron IIoT Platform

Log Into the N3uron IIoT Platform Using a Web Browser

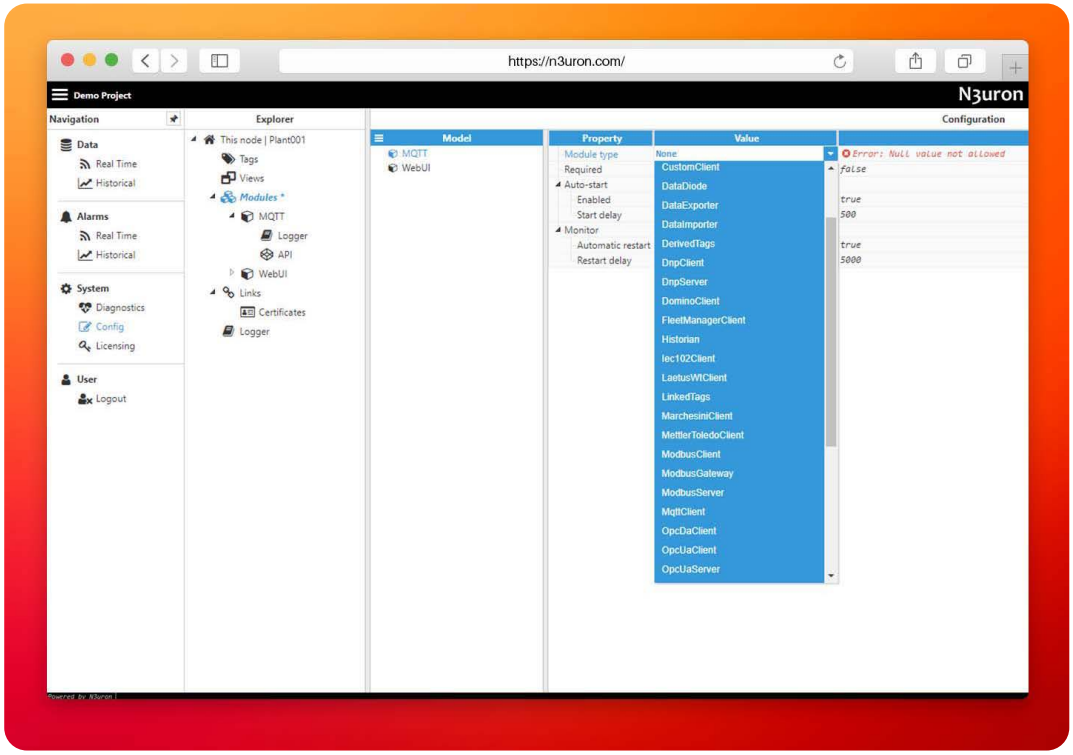
If this is your first time accessing N3uron, open your web browser and type http://localhost:8003. By default, the **User** and **Password** is admin and n3uron respectively.



Screenshot displaying the log in interface in N3uron’s IIoT platform WebUI

Create a Module Instance Within N3uron's WebUI Interface

- **Step 01:** In the **Navigation** panel, select **Config**.
- **Step 02:** In the **Explorer** panel, select **Modules**.
- **Step 03:** Click on the **Model** menu and select **New Module**.
- **Step 04:** The instance can be given any name but for the example we will use MQTT.
- **Step 05:** Set the **Module Type** property to **MqttClient**. Leave the rest of the properties as their default values and click **Save**.



Screenshot displaying how to create an instance using N3uron's MQTT Module panel

Configure N3uron's MQTT Module within the WebUI's Explorer Panel

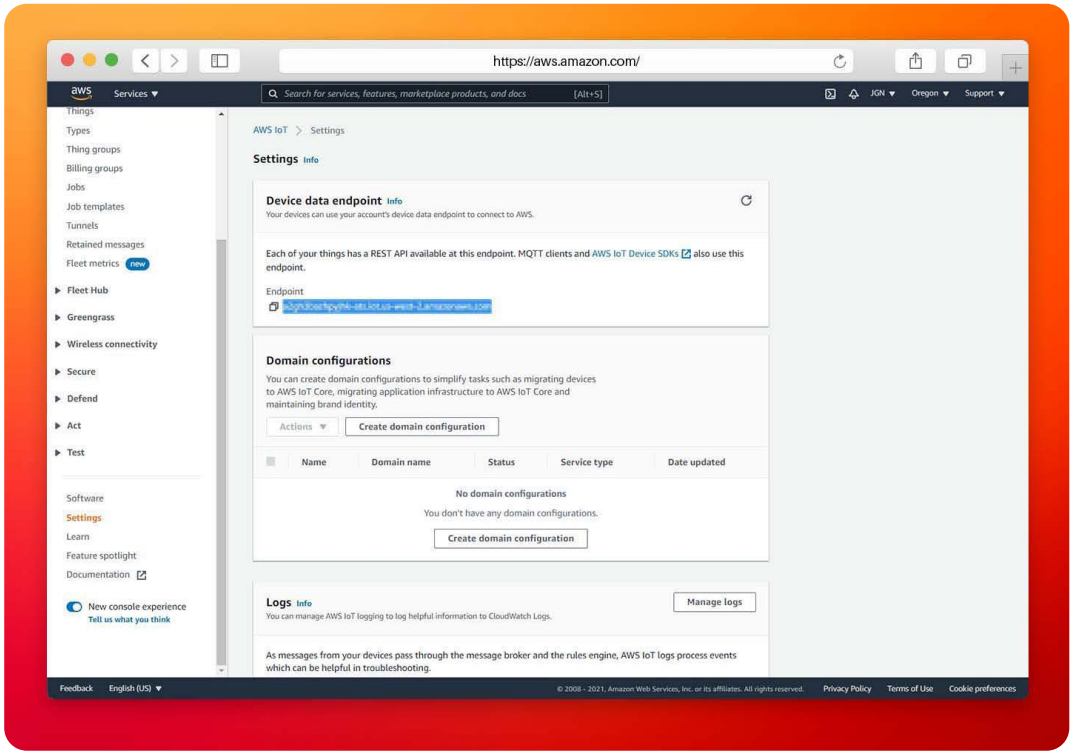
- **Step 01:** In the **Explorer** panel, select the MQTT instance you have just created.
- **Step 02:** Click on the Model menu button and select **New Connection**.
- **Step 03:** Give the New connection a name. In this example, it has been named AWS.
- **Step 04:** Configure the connection properties:

  A: Select **Amazon Web Services** from the **Destination Broker** drop down menu.

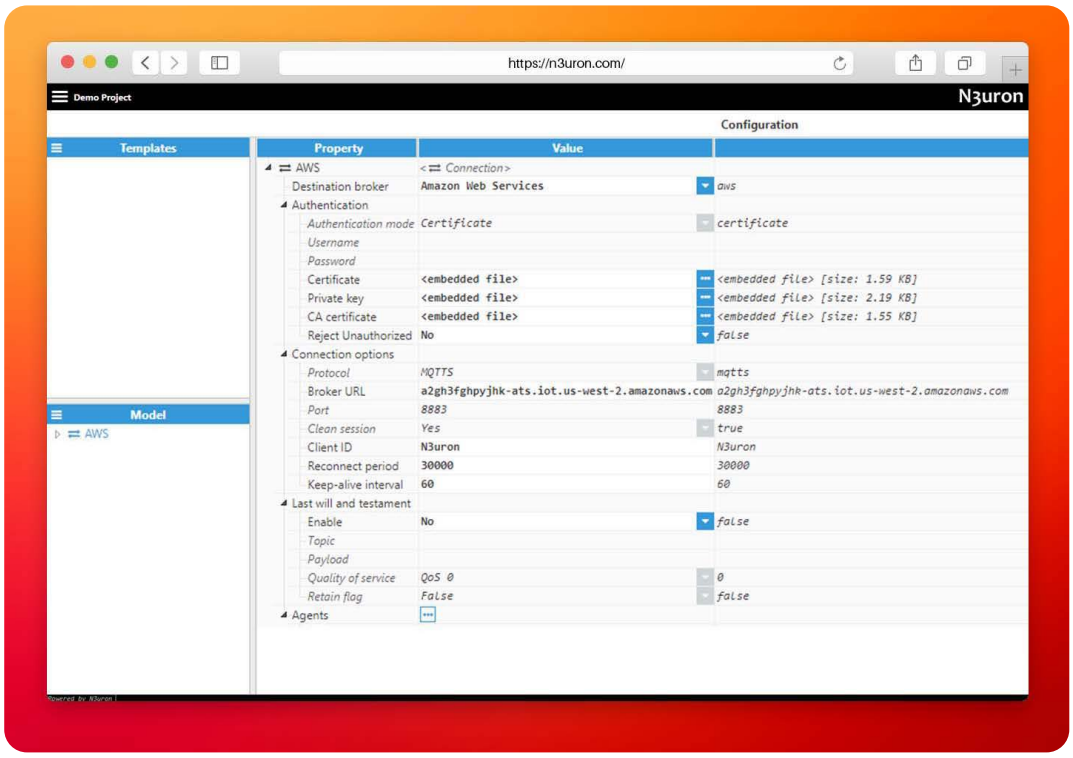
  B: Load the **Certificate, Private key** and **CA certificate** you downloaded and saved when you created the Thing in the AWS IoT Console.

  C: In the AWS IoT console, in the left-hand menu, go to **Settings** and copy your **Device Data Endpoint**. Go back to N3uron and paste it in the **Broker URL** field.

  D: Leave the rest of the properties as their default values and click on **Save**.



Screenshot displaying the Endpoint in the Amazon IoT platform

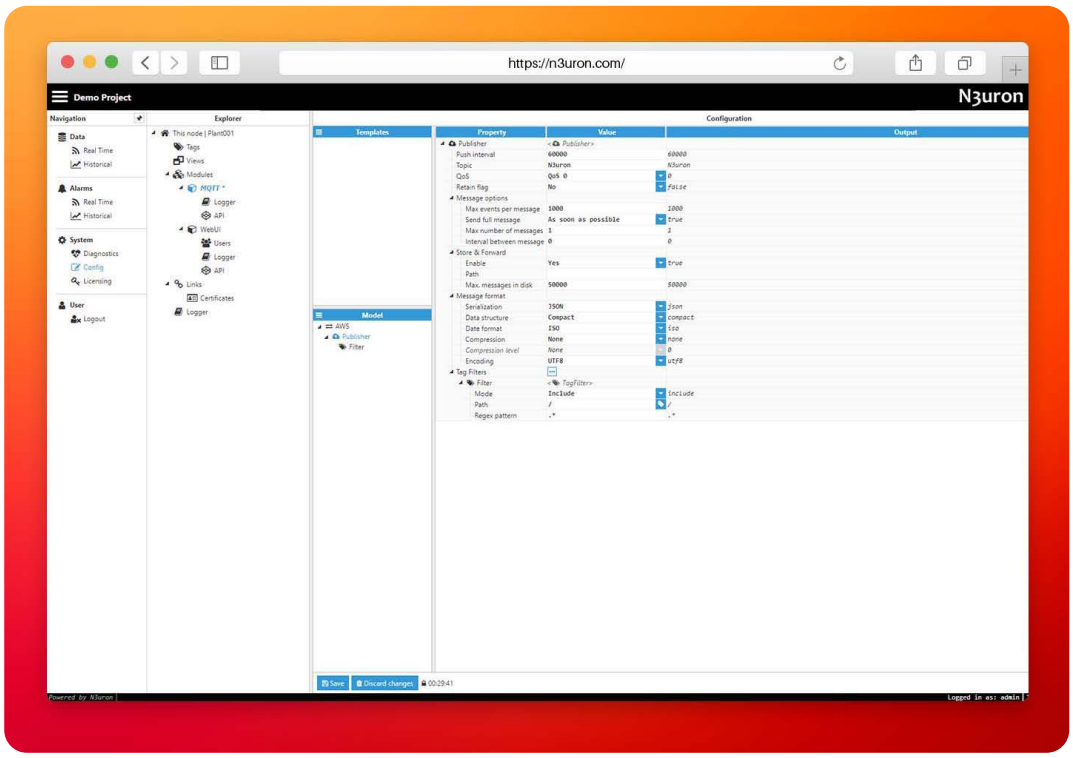


Screenshot displaying AWS IoT platform connection configuration in N3uron's MQTT module panel

Publish Data Using N3uron's MQTT Module

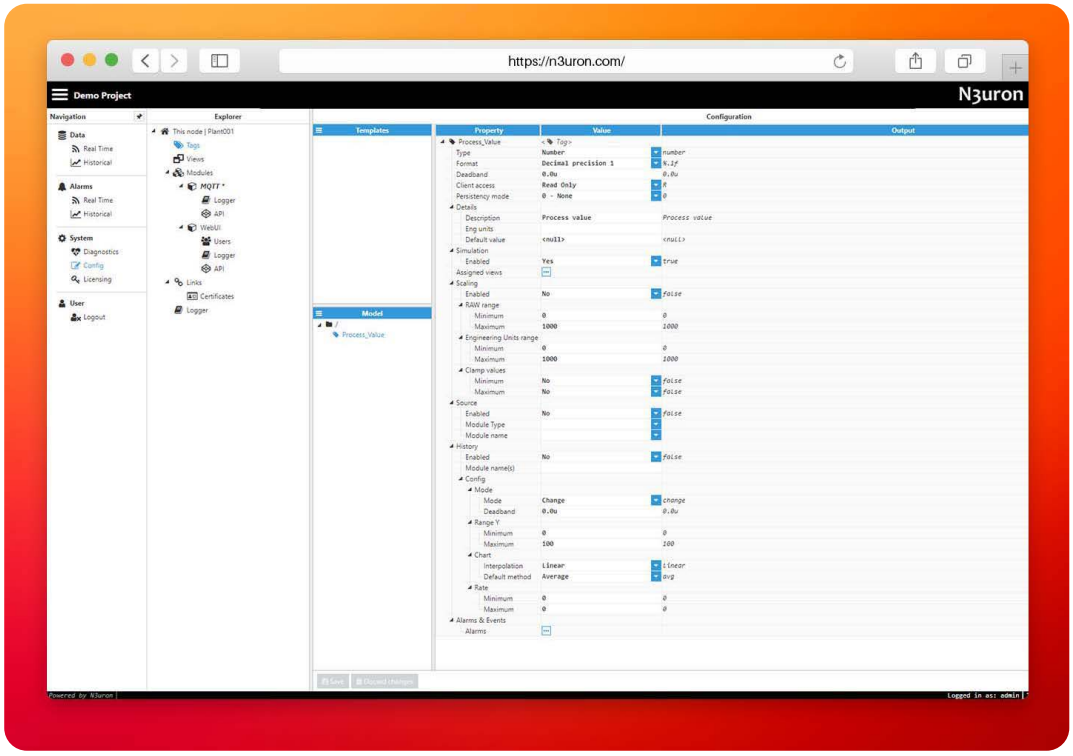
- **Step 01:** Within the **Model** panel, right-click on the AWS Connection you have just configured, select – New Publisher, and give it a name. In this example, we will simply use Publisher.
- **Step 02:** Click on it and add a name in the **Topic** field. In this example, we have used N3uron.
- **Step 03:** Click on the Tag Filter button, select **New Tag Filter**, and change the default name. In this example

we have used **Filter**. Leave **Mode**, **Path**, and **Regex pattern** as their default values. With this configuration, every tag configured in N3uron will be published to our AWS Broker.



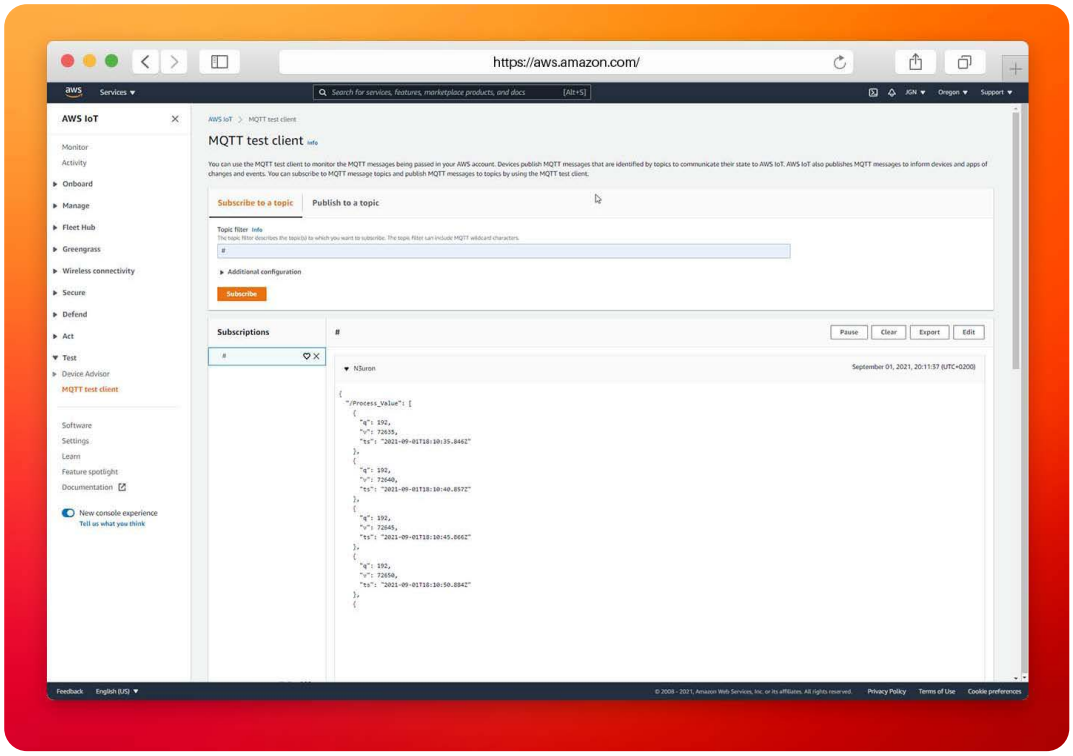
Screenshot displaying the publisher configuration setting in N3uron's MQTT module panel

- **Step 04:** In the **Explorer panel**, select **Tags**.
- **Step 05:** In the **Model** menu, right-click on the folder icon, select **New Tag**, and give it a name. In this example, we will use Process_value.
- **Step 06:** Within the **Configuration** panel, set the following properties using the values shown below, leaving the rest of them as their default values:
  - **Type:** Number.
  - **Simulation/Enabled:** Yes



Screenshot displaying the tag configuration settings in N3uron's MQTT module panel

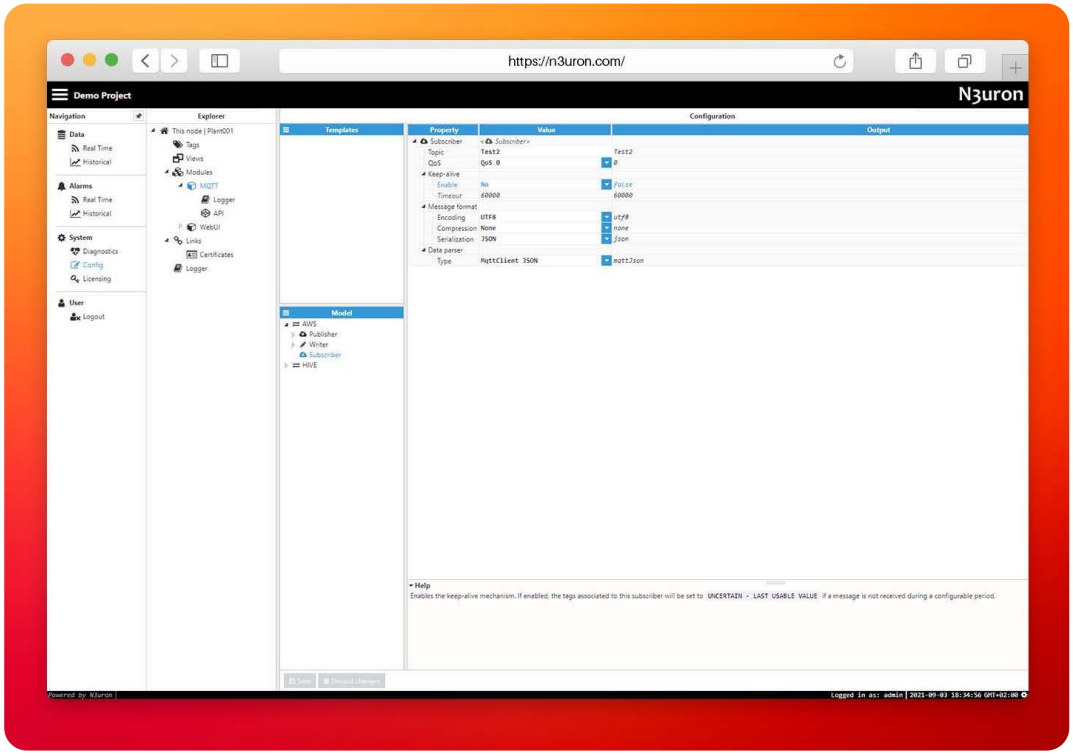
- **Step 07:** Go to the AWS IoT console and in the left-hand menu, select **MQTT test client**.
- **Step 08:** Click on the **Subscribe to a topic** tab, **enter #** in the **Topic filter** to subscribe to everything, and click on **Subscribe**. Within a few seconds you should see the messages published to the N3uron/Process_Value topic that corresponds to our previous configuration.



Screenshot displaying the MQTT test client panel in the Amazon IoT platform

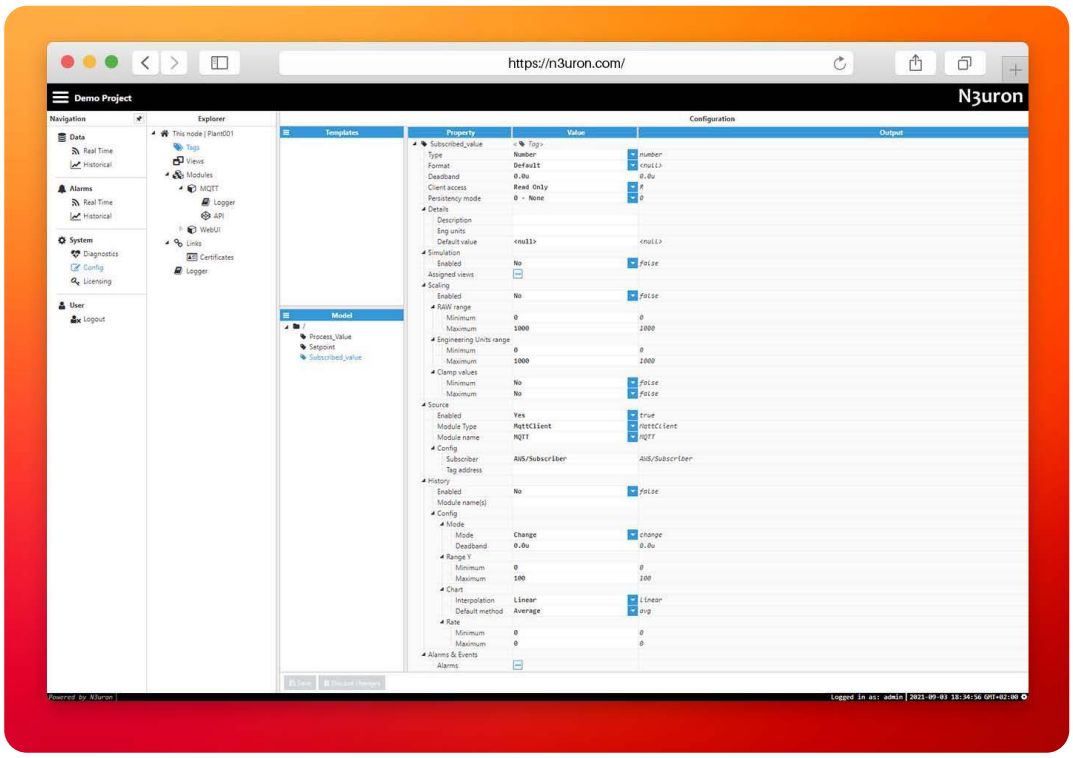
Subscribe to a Topic Using N3uron's MQTT Module

- **Step 01:** In the **Model** panel, right-click on the AWS Connection, select **New Subscriber**, and give it a name. In this example, we will simply use Subscriber.
- **Step 02:** Click on it and add a name in the **Topic** field. In this example, we have used Test2.
- **Step 03:** Set the following properties using the values shown below, leaving the rest of them as their default values:
  - **Qos:** Qos 0.
  - **Encoding:** UTF8
  - **Compression:** None
  - **Serialization:** JSON
  - **Data parser/Type:** MqttClient JSON



Screenshot displaying the subscriber configuration settings in N3uron's MQTT module panel

- **Step 04:** Within the **Explorer panel**, select **Tags**.
- **Step 05:** In the **Model** menu, right-click on the folder icon, select **New Tag**, and give it a name. In this example, we will use Subscribed_value.
- **Step 06:** In the **Configuration** panel, set the following properties using the values shown below, leaving the rest of them as their default values:
  - **Type:** Number.
  - **Source/Enabled:** Yes
  - **Module Type:** MqttClient
  - **Module name:** MQTT
  - **Config/Subscriber:** AWS/Subscribe.
- **Step 07:** Click on **Save**.

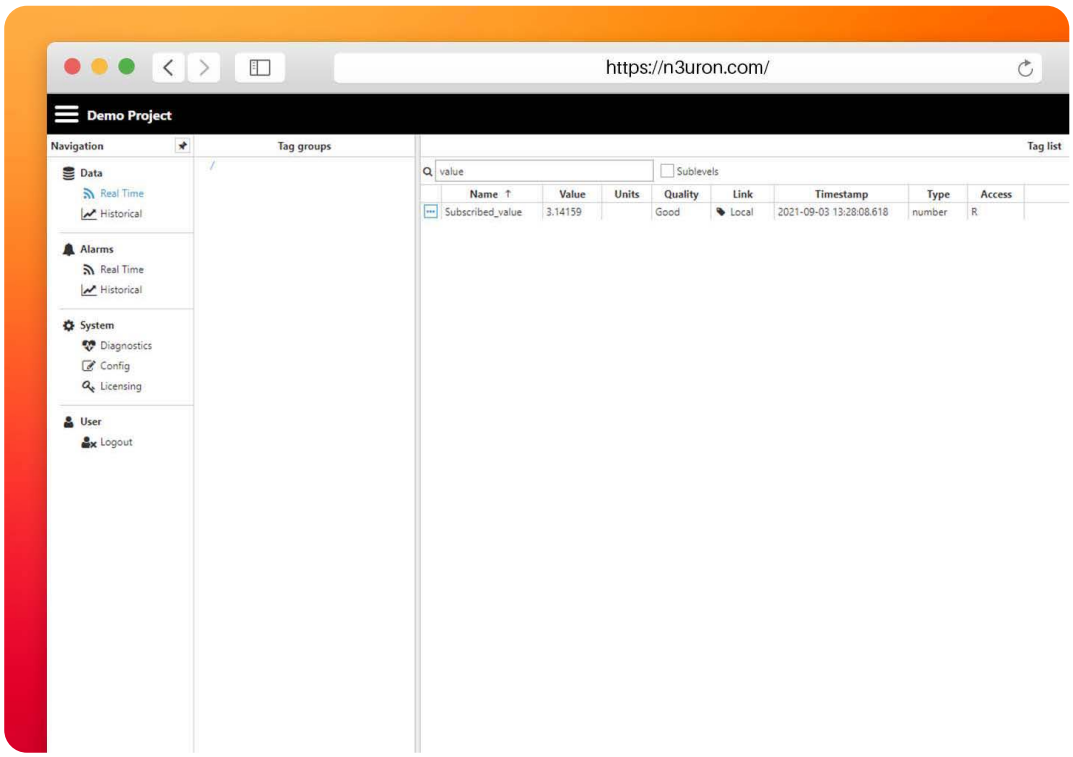


Screenshot displaying the subscribed tag configuration settings in N3uron's WebUI interface

- **Step 08:** Go to the AWS IoT Console and in the left-hand menu select MQTT test client.
- **Step 09:** Click on the **Publish to a topic** tab, enter Test2 in the **Topic filter**, and enter the following in **Message Payload:**

```
{
        "/Subscribed_value": [{
            "v": 3.14159,
        "q": 192,
        "ts": 1630668488618
    }]
}
```

- **Step 10:** Click on the Publish button.
- **Step 11:** Go back to the N3uron WebUI interface and in the left-hand panel, select **Data/Real Time**. You should now see the Subscribed_Value tag you created before with a value of 3.14159.

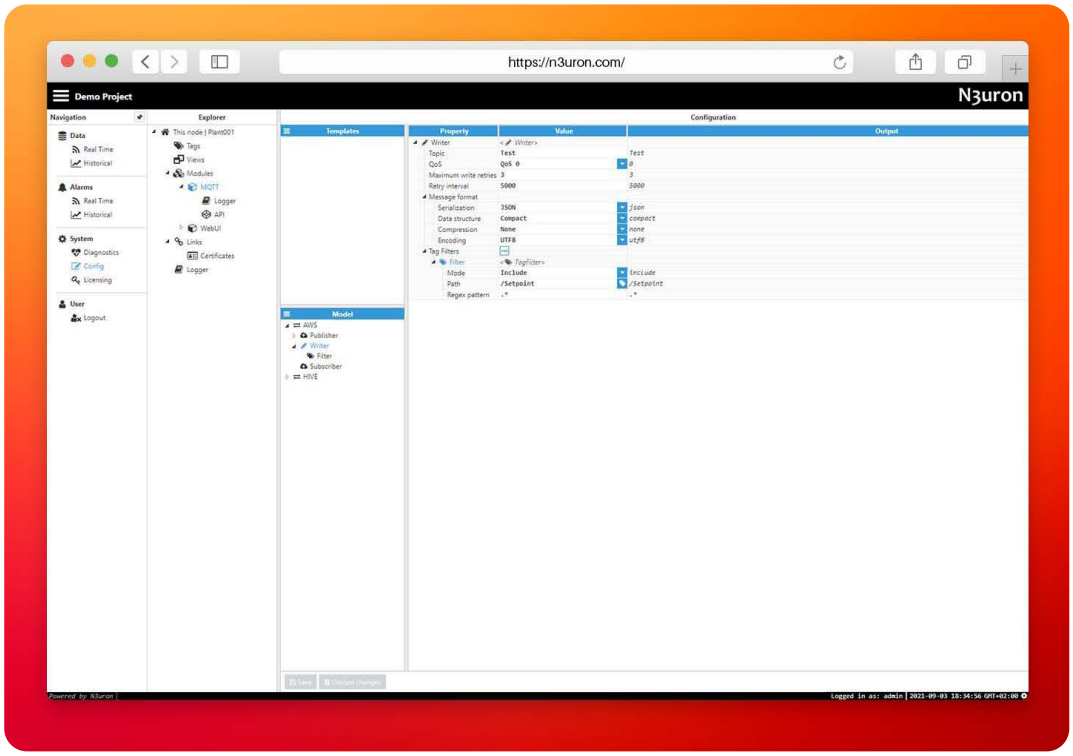


Screenshot displaying real-time values in N3uron's WebUI interface

Create a Writer Using N3uron's MQTT Module

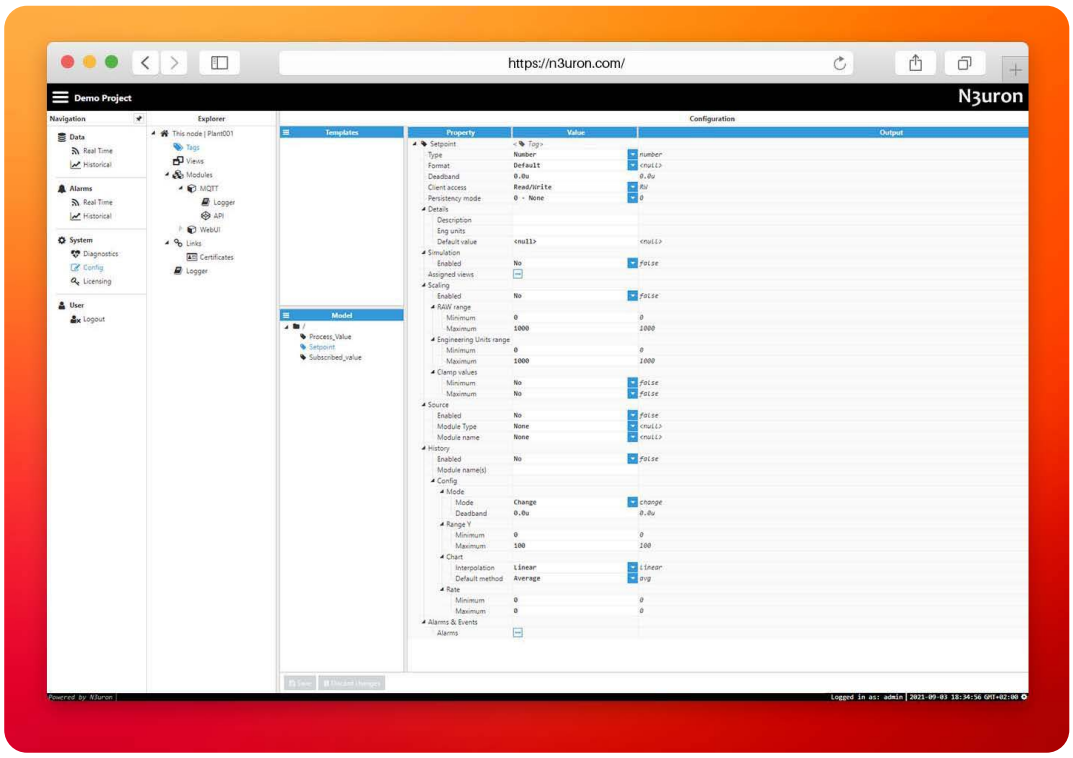
Writers are similar to Subscribers, the main difference between them being that Writers issue a write command to the underlying data provider for a specific tag, while a Subscriber acts directly as a data provider.

- **Step 01:** Within the **Model** panel, right-click on the AWS Connection, select **New Writer**, and give it a name. In this example, we will simply use Writer.
- **Step 02:** Click on it and add a name in the **Topic** field. In this example, we have used Test.
- **Step 03:** Set the following properties using the values shown below, leaving the rest of them as their default values:
  - **Qos:** Qos 0.
  - **Serialization:** JSON
  - **Data Structure:** Compact
  - **Compression:** None
  - **Encoding:** UTF8
- **Step 04:** Click on the **Tag Filter** button, select **New Tag Filter**, and change the default name. In this example, we have used Filter.
- **Step 05:** Enter /Setpoint in the **Path** field and leave **Mode & Regex pattern** as their default values.



Screenshot displaying the writer configuration settings in N3uron's MQTT module panel

- **Step 06:** Within the **Explorer panel**, select **Tags**.
- **Step 07:** In the **Model** menu, right-click on the folder icon, select **New Tag**, and give it a name. In this example, we will use Setpoint.
- **Step 08:** In the **Configuration** panel, set the following properties using the values shown below, leaving the rest of them as their default values:
  - **Type:** Number.
  - **Source/Enabled:** No
  - **Client access:** Read/Write
- **Step 09:** Click on **Save**.



Screenshot displaying the writer tag configuration settings in N3uron's WebUI interface

- **Step 01:** Go to the AWS IoT console and in the left-hand menu, select **MQTT test client**.
- **Step 02:** Click on the **Publish to a topic** tab, enter Test in the **Topic** filter, and enter the following in the **Message Payload**:

```
{
        "/Setpoint": 25.8
}
```

- **Step 03:** Click on the **Publish** button.
- **Step 04:** Go back to the N3uron WebUI interface and in the left-hand panel, select **Data/Real Time**. You should now see the Setpoint tag you created before with a value of 25.8.

Conclusion: How to Connect AWS IoT Using N3uron's MQTT Module

Connecting your assets to the AWS infrastructure is extremely easy using N3uron's MQTT Client module. If you're ready to go using MQTT, download the N3uron free trial version and read our MQTT Client Manual on how to implement and use N3uron's MQTT software module on our communication platform. Download the MQTT Client Manual