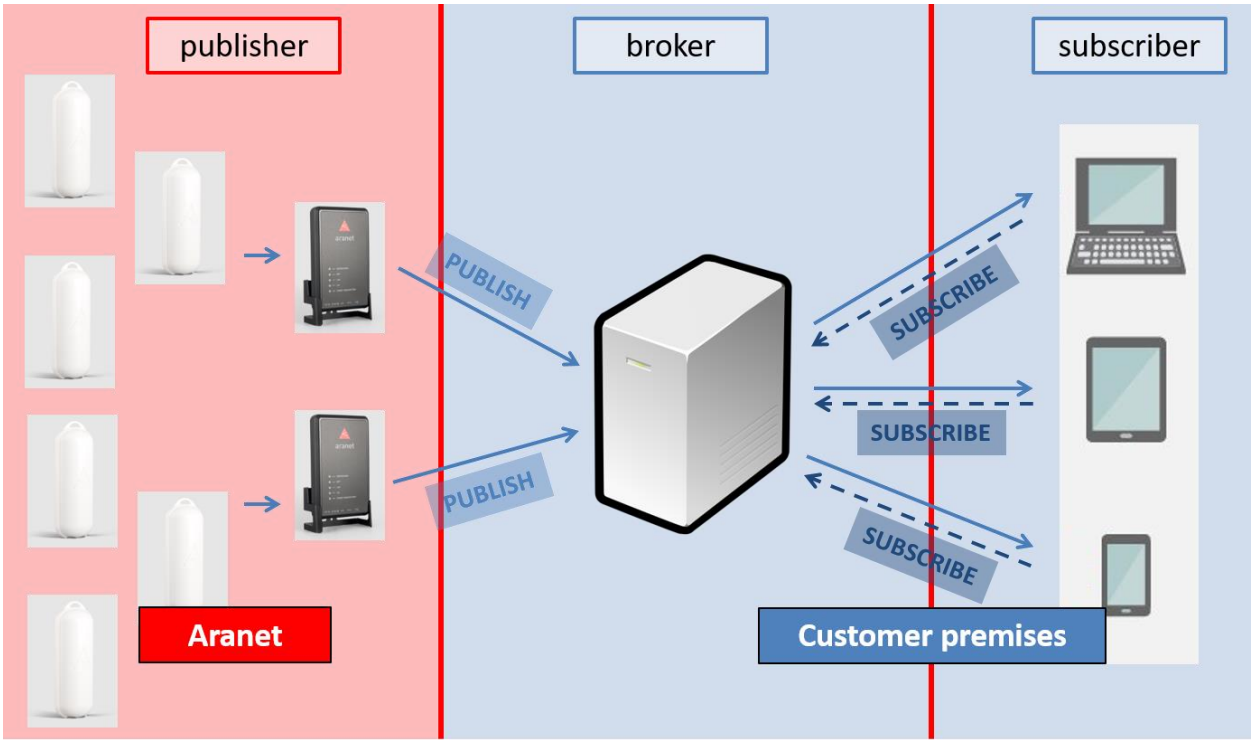
Aranet MQTT functionality and integration with Amazon AWS and Microsoft Azure

1. General MQTT network structure:



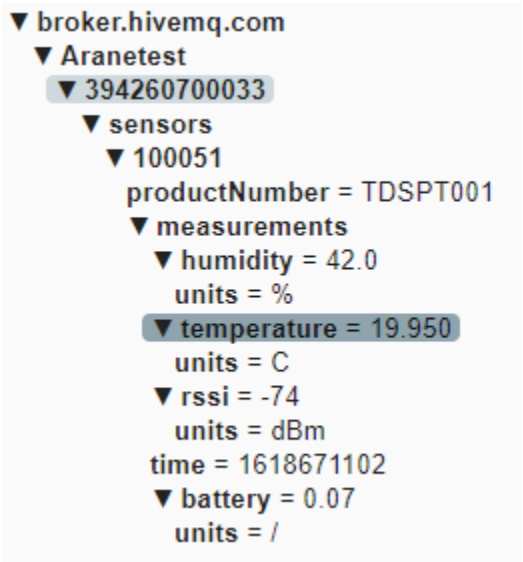
2. MQTT message format

Sensor measurement data messages from the PRO base can be published on the MQTT broker in 3 following formats (hierarchy):

1) **raw**
   in topic structure <root topic name>/*<PRO base serial number>***/sensors/***<sensor ID>***/measurements/***<measurement type>* where
   a) **<root topic name>** - Aranet PRO base station MQTT message identification name which should be configured on the base MQTT page Root topic field. For more details see below *Aranet PRO base station configuration interface*
   b) **<PRO base serial number>** - serial number of PRO base station;
   c) **<sensor ID>** - 6 HEX digit sensor ID where the first digit is the sensor segment (for details see Segments for sensors document) and remaining 5 digits are from sensor marking from the physical label on the sensor body which can be seen also in PRO base station graphical user interface;

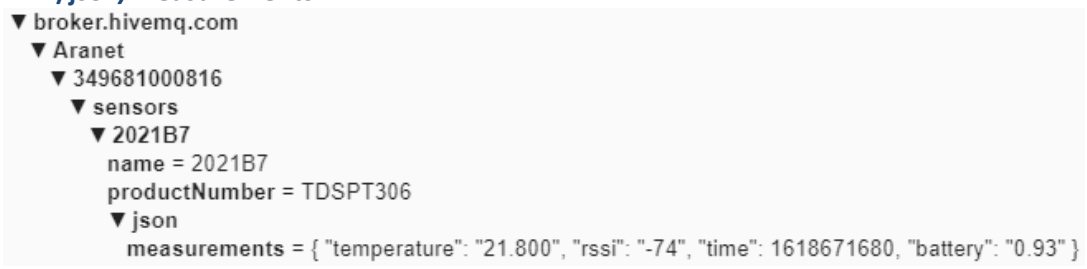
d) **<measurement type>** can be one of the following:
   a. **temperature** – data is given in degrees C (Celsius);
   b. **humidity** – relative humidity data is given in percentage %;
   c. **co2** – carbon dioxide concentration level data given in ppm(parts per million);
   d. **co2Abc** – shows whether CO2 manual (***0***) or automatic (***1***) calibration mode is enabled for the sensor;
   e. **atmosphericpressure** – atmospheric pressure data are given in Pa (Pascal);
   f. **voltage** – data are given in V (Volts);
   g. **current** – electric current data given in A (Ampere);
   h. **weight** – tarred weight in kg (kilogram);
   i. **weight raw** – untarred weight in kg (kilogram);
   j. **illuminance** – data from LUX sensor given in lx (lux);
   k. **distance** – data are given in m (meters);
   l. **vwc** -volumetric water content data of soil/substrate given as a fraction of one whole;
   m. **bec** – bulk electric conductivity data are given in S/m (Siemens per meter);
   n. **pec** - pore water electrical conductivity data are given in S/m (Siemens per meter);
   o. **dp** - dielectric permittivity data of soil or substrate given in absolute numbers;
   p. **ppfd** - photosynthetic photon flux density data are given in micromol/(m^2s) (micromol per square meters multiplied by seconds);
   q. **pulses** – periodic pulses in absolute numbers;
   r. **pulsescumulative** - cumulative pulses in absolute numbers;
   s. **co** – carbon monoxide concentration level data are given in ppm (parts per million);
   t. **differentialpressure** – data are given in Pa (Pascal);
   u. **derived** – derived measurements in user-defined units;
   v. **rssi** – received signal strength data given in dBm;
   w. **battery** – battery charge level which is given as a fraction of one whole;
   x. **time** – measurement time in Unix epoch format: https://www.freeformatter.com/epoch-timestamp-to-date-converter.html

Additionally measurement units for the sensor data according to measurement type is published in topics: <root topic name>/*<PRO base serial number>***/sensors/***<sensor ID>* **/measurements/***<measurement type>***/units**

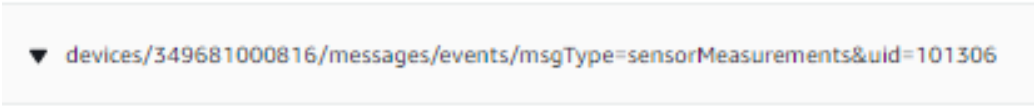


2) **JSON**

in topic structure <root topic name>/*<PRO base serial number>***/sensors/***<sensor ID>***/json/measurements**



3) **Azure** format for sensor data publishing to Azure IoT Hub platform:

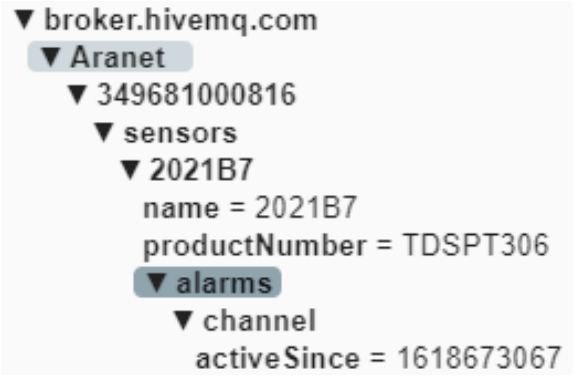



```
{
  "sensors": [
    {
      "uid": "101306",
      "measurements": [
        {
          "measurement": "humidity",
          "value": "38.0",
          "units": "%"
        },
        {
          "measurement": "temperature",
          "value": "21.850",
          "units": "C"
        },
        {
          "measurement": "rssi",
          "value": "-47",
          "units": "dBm"
        },
        {
          "measurement": "time",
          "value": 1618691111
        },
        {
          "measurement": "battery",
          "value": "0.90",
          "units": "/"
        }
      ]
    }
  ]
}
```

Sensor alarm messages from PRO base is published on the MQTT broker in following hierarchy(format): <root topic name>/*<PRO base serial number>***/sensors/***<sensor ID>***/alarms/** +

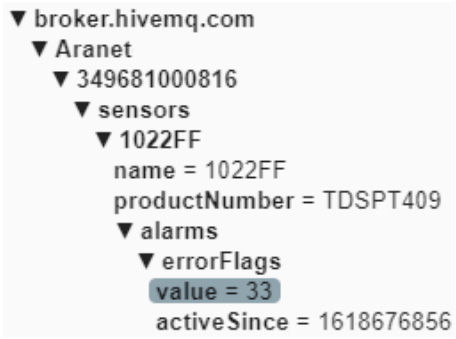
a. **battery/activeSince** – showing time in Unix epoch format when low battery charge alarm appeared in the sensor:

b. **channel/activeSince** – showing time in Unix epoch format when Aranet PRO base station recorded the event when sensor started using different radio channel than configured on the base itself:



c. **packetsLost/activeSince** – showing time in Unix epoch format when Aranet PRO base station recorded that measurement data from some sensor is not received/missing:

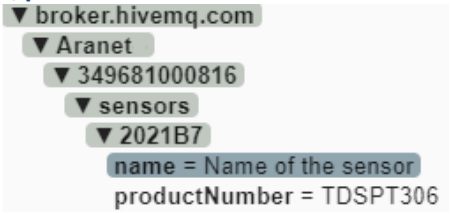
d. **errorFlags/**
    a. **value** - showing number error code value when instead of measurements error message was received from the sensor;
    b. **activeSince** - showing time in Unix epoch format when instead of the measurement error message was received from the sensors:



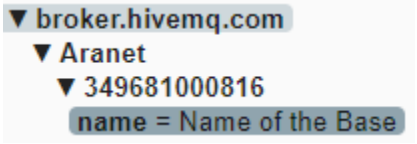
e. **<measurement>** - shows for which measured parameter configured alarm threshold was breached;
    a. **value** – shows measurement value that generated the alarm;
    b. **diff –** shows value by what configured alarm threshold was breached. It is positive when the upper threshold was breached and negative when the lower threshold is breached;
    c. **activeSince -** shows time in Unix epoch format when alarm threshold was breached:

Aranet PRO base station publishes also:

1) name that is assigned to the sensor on the Aranet PRO base station in topic <root topic name>/*<PRO base serial number>***/sensors/***<sensor ID>***/name** and
2) product number of the sensor in topic <root topic name>/*<PRO base serial number>***/sensors/***<sensor ID>***/productNumber**:

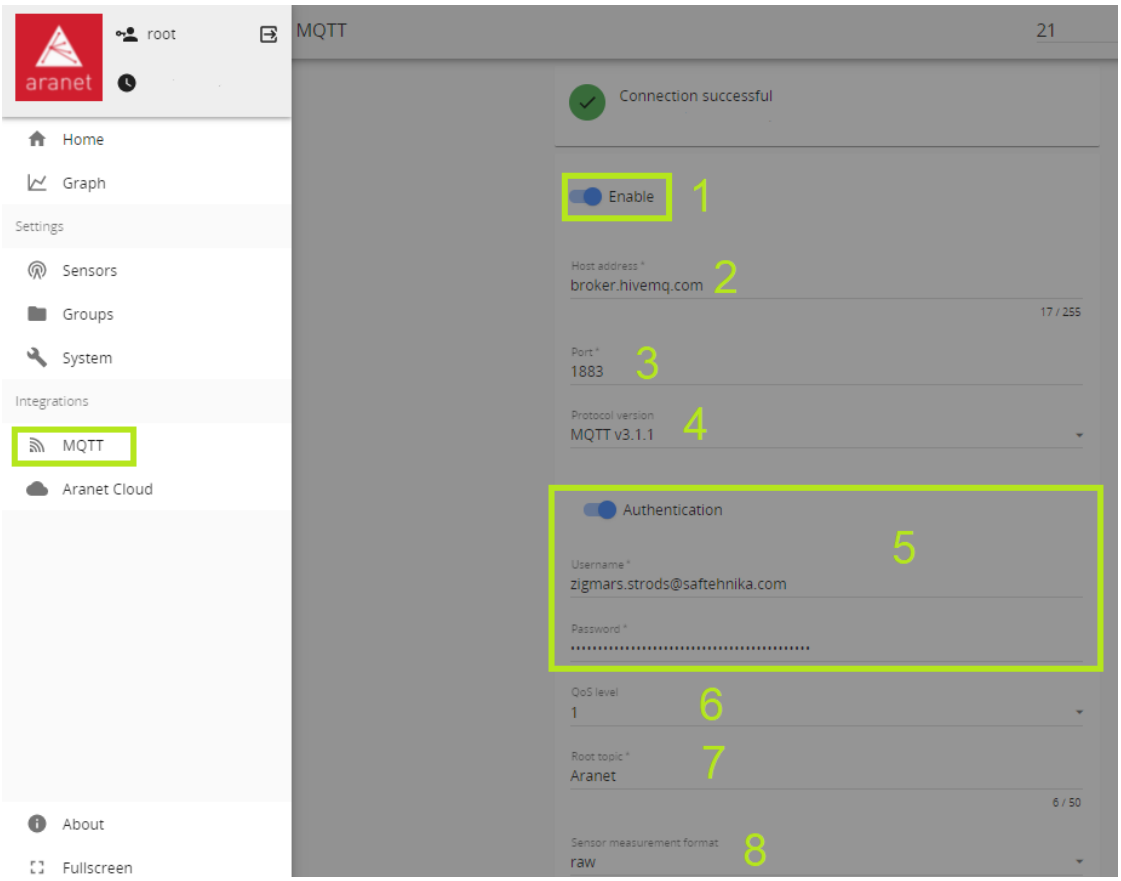


3) name of the Aranet PRO base station itself in topic <root topic name>/*<PRO base serial number>***/name**:

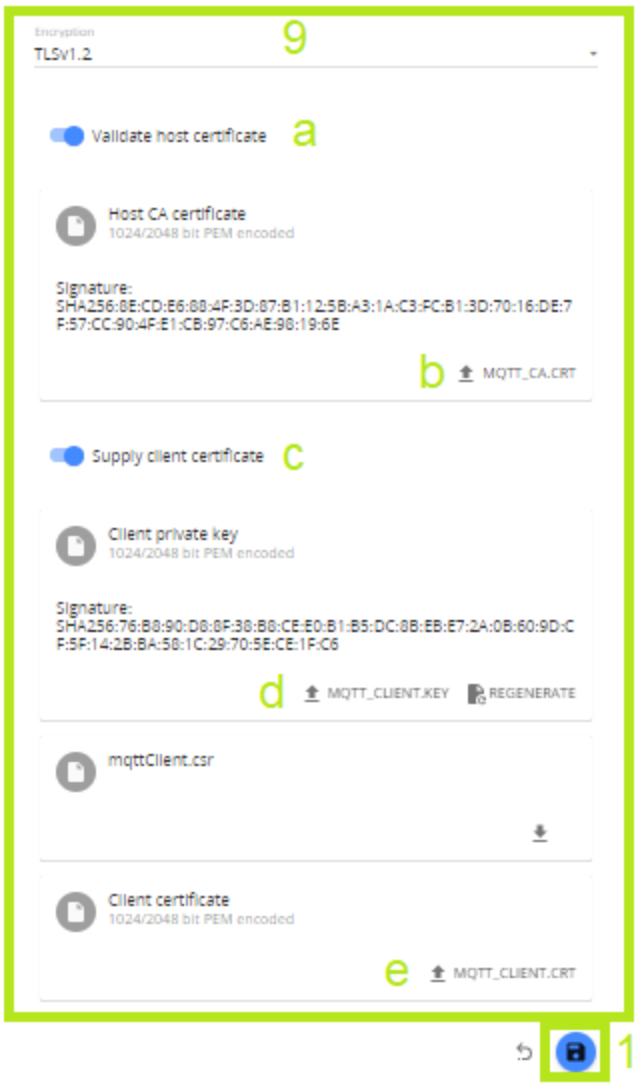


3. Aranet PRO base station configuration interface

Aranet PRO base station connection to MQTT broker is configured in the **MQTT** section of the graphical user interface. In the example below we will use configuration for connection to Hivemq public MQTT broker *broker.hivemq.com*:



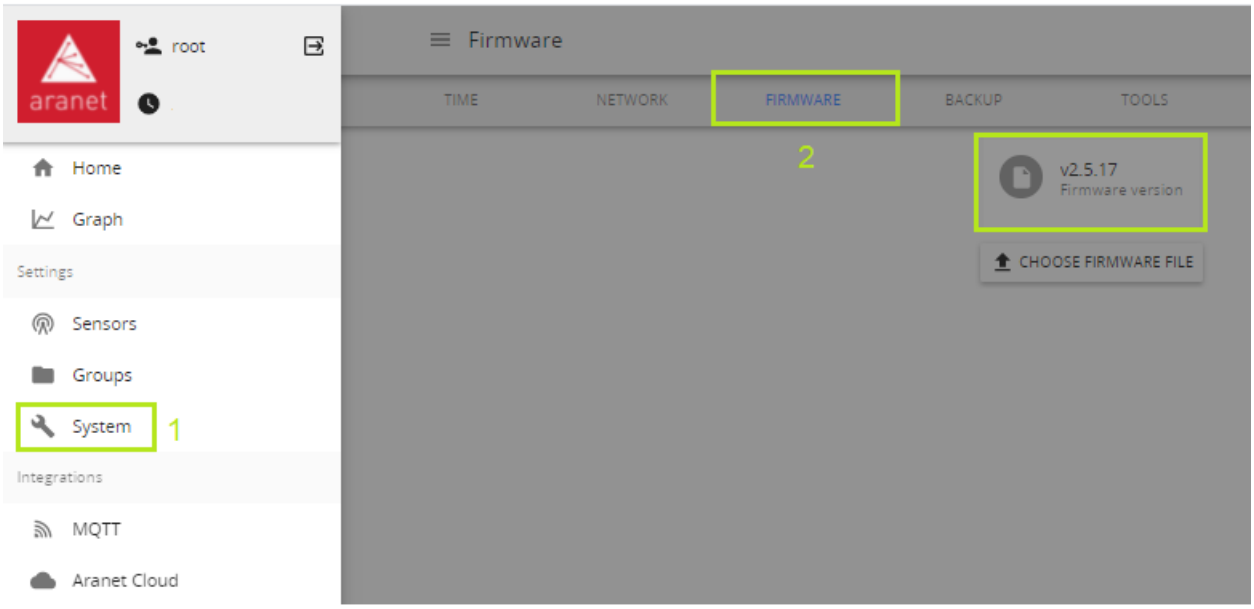
1) **Enable** – allows enabling/disabling MQTT data transmission from Aranet PRO base station;
2) **Host address** – allows configuring IP address or hostname for the MQTT broker;
3) **Port** – allows selecting the TCP port used for the connection to the MQTT broker. The most common ports are ***1883*** or ***8883***;
4) **Protocol version –** allows selecting MQTT protocol version used for connection to MQTT broker. The broker should support this version;
5) **Authentication** - upon necessity allows enabling additional authentication for the connection to MQTT broker and enter
    a. **Username** and
    b. **Password** for such connection authentication;
6) **QoS level** (0, 1 or 2) for MQTT message delivery on the MQTT broker can be selected as necessary http://www.steves-internet-guide.com/understanding-mqtt-qos-levels-part-1/ and http://www.steves-internet-guide.com/understanding-mqtt-qos-2/;
7) **Root topic –** allows selecting root topic name with what MQTT messages will be published from Aranet PRO base station on MQTT broker. In our example, we will use the name ***Aranet***;
8) **Sensor measurement format** – allows selecting format (***raw***, ***JSON*** or ***Azure***) in which MQTT messages from Aranet PRO base station will be published on MQTT broker;



9) **Encryption** - upon necessity allows configuration of additional encrypted certificates (TLS version 1.1, 1.2 or 1.3) to be used for the more secure connection to the MQTT broker;
   a. **Validate host certificate** – enable to upload necessary secure connection certificates;
   b. MQTT_CA.CRT - press to upload root CA certificate in PEM format for MQTT broker;
   c. **Supply client certificate** - enable to upload the device public certificate and private key for secure connection to MQTT broker
   d. MQTT_CLIENT.KEY - press to upload the Aranet PRO base station private key for secure connection to MQTT broker;
   e. MQTT_CLIENT.CRT - press to upload the Aranet PRO base station public key for secure connection to MQTT broker
10) When all necessary configuration parameters are entered, they should be saved by pressing the blue Save icon . If configured MQTT connection is successful, then ***Connection successful*** message will be shown on the top of the page showing also the precise time when the connection was established.

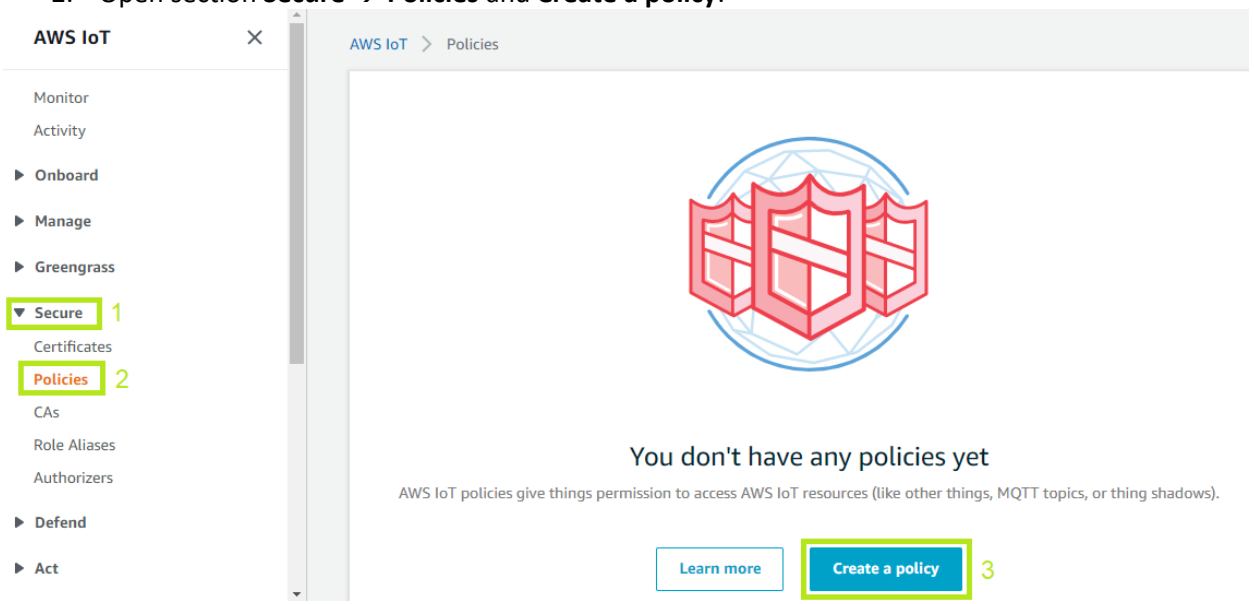
4. MQTT connection configuration with Amazon AWS platform

Aranet PRO base station allows all sensor data publishing directly to AWS IoT Core, but here base only should have firmware version at least 2.5.17. So before proceeding further, please first check the firmware version of Aranet PRO base station in the graphical user interface section **System → FIRMWARE** and if it is older than 2.5.17, then update to the latest version available from https://aranet.com/downloads/ section of our webpage:

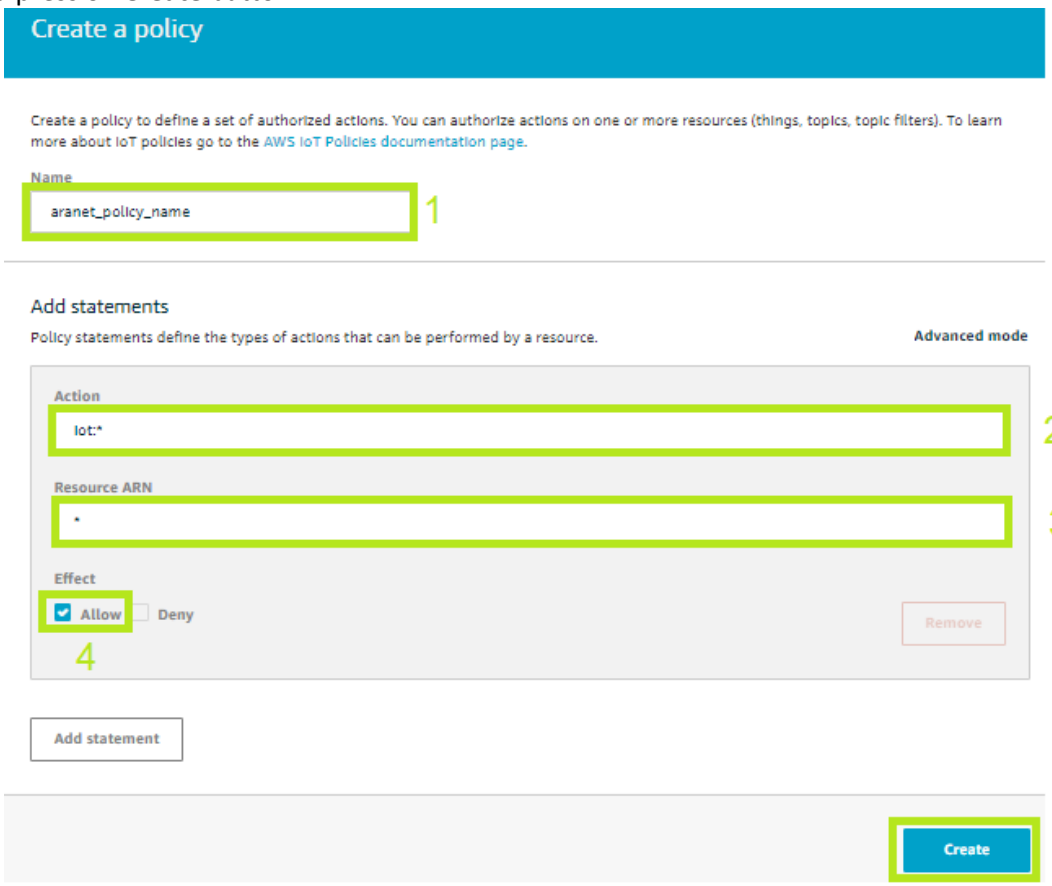


The MQTT connection configuration with the Amazon AWS platform itself can be done in the following steps:

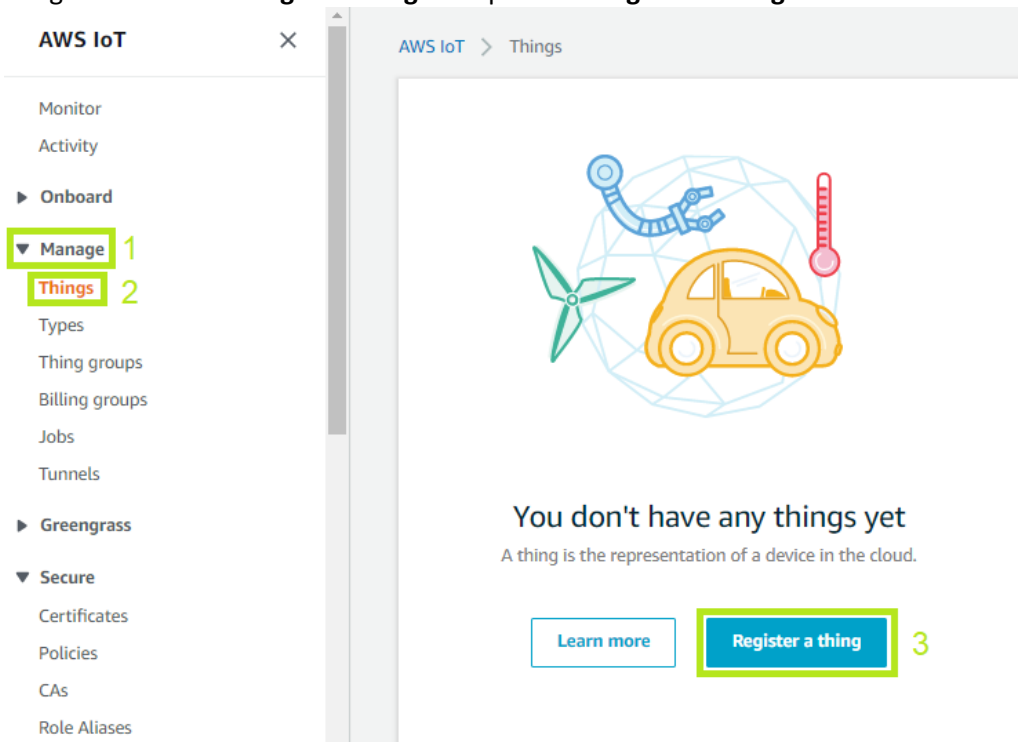
1. Log in to Your AWS account and go to the IoT Core section of the platform;
2. Open section **Secure → Policies** and **Create a policy**:



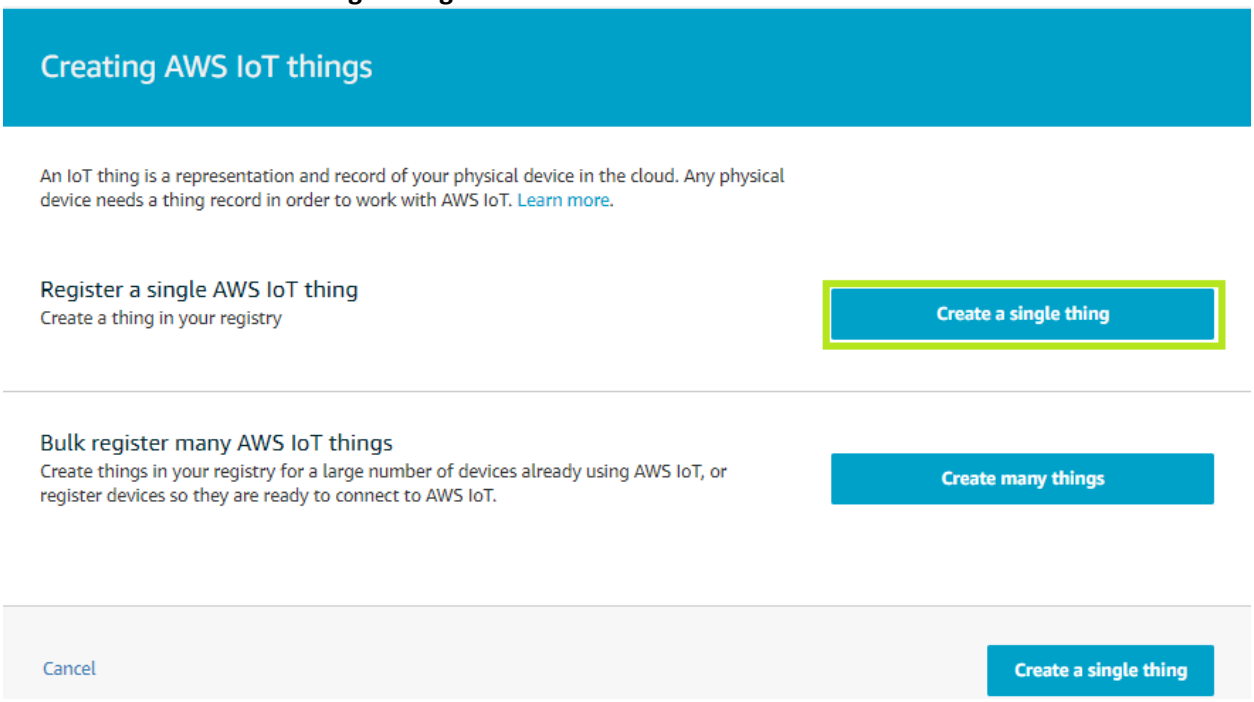
3. Enter the **Name** of the policy, select **Action** as ***iot.**** and **Resource ARN** as *, click to ***Allow*** **Effect** and press on **Create** button:



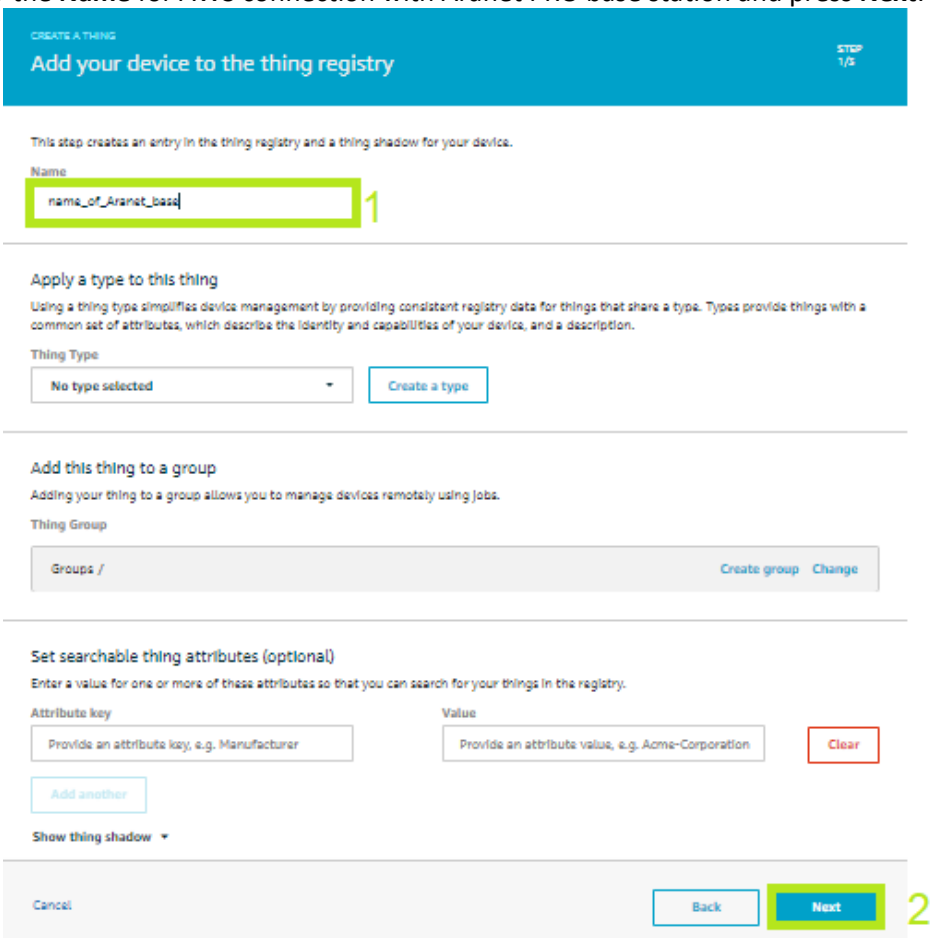
4. Then go to section **Manage** → **Things** and press on **Register a thing**:



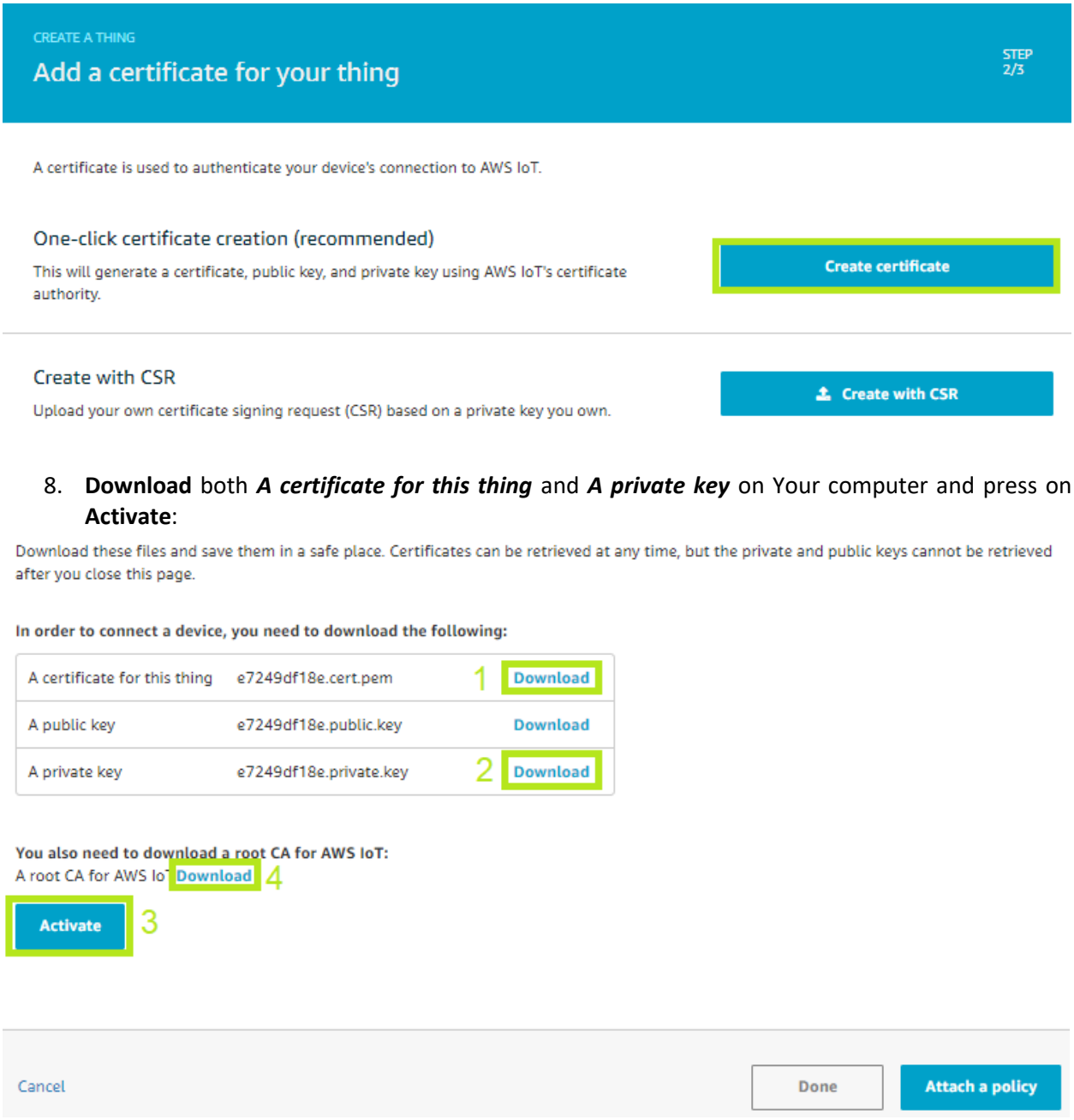
5. Press on **Create a single thing**:



6. Enter the **Name** for AWS connection with Aranet PRO base station and press **Next**:

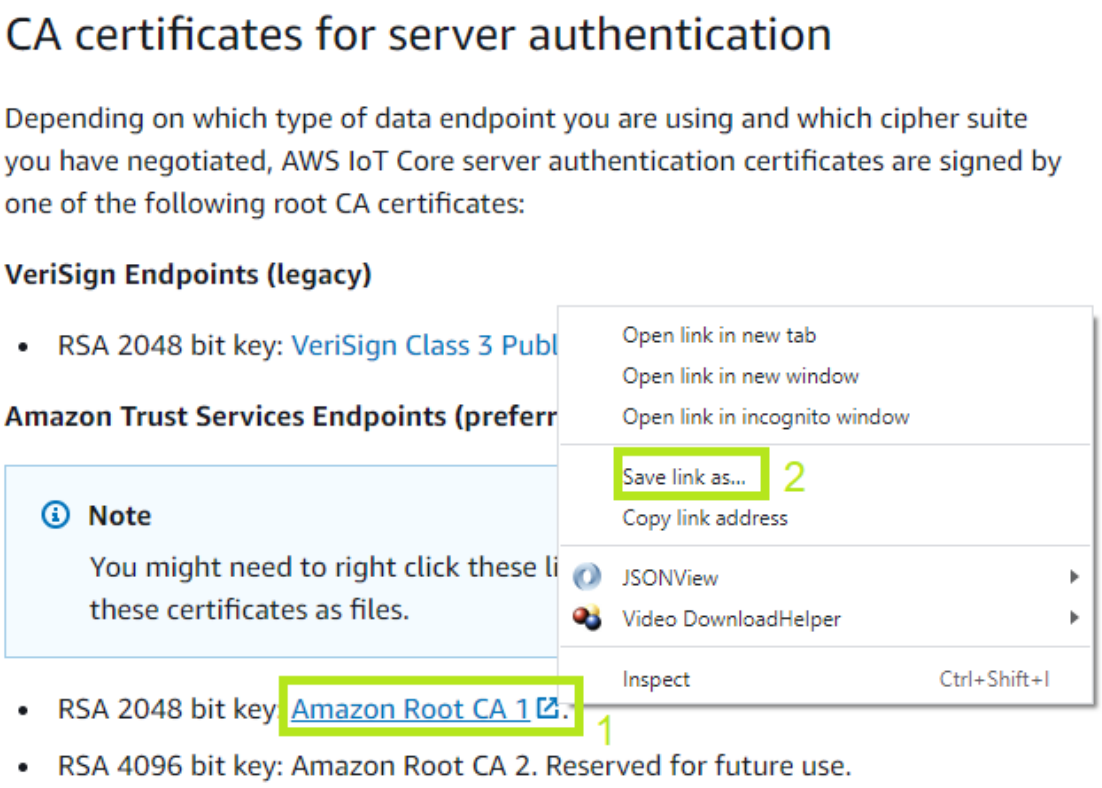


7. Press on **Create certificate**:



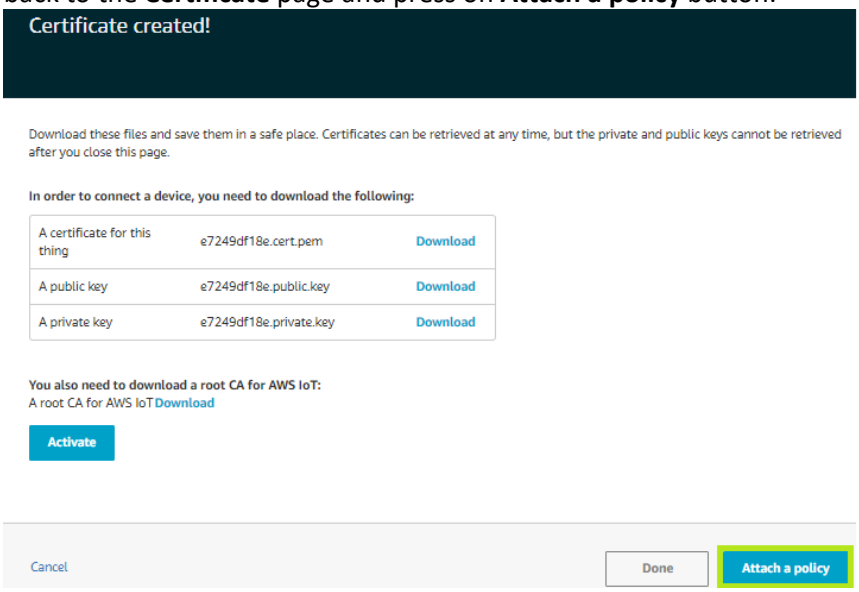
8. **Download** both ***A certificate for this thing*** and ***A private key*** on Your computer and press on **Activate**:

9. After that click on **Download** link next to **A root CA for AWS IoT** and from the opened new page right click with the mouse to ***Save link as***… for ***Amazon Root CA 1***:

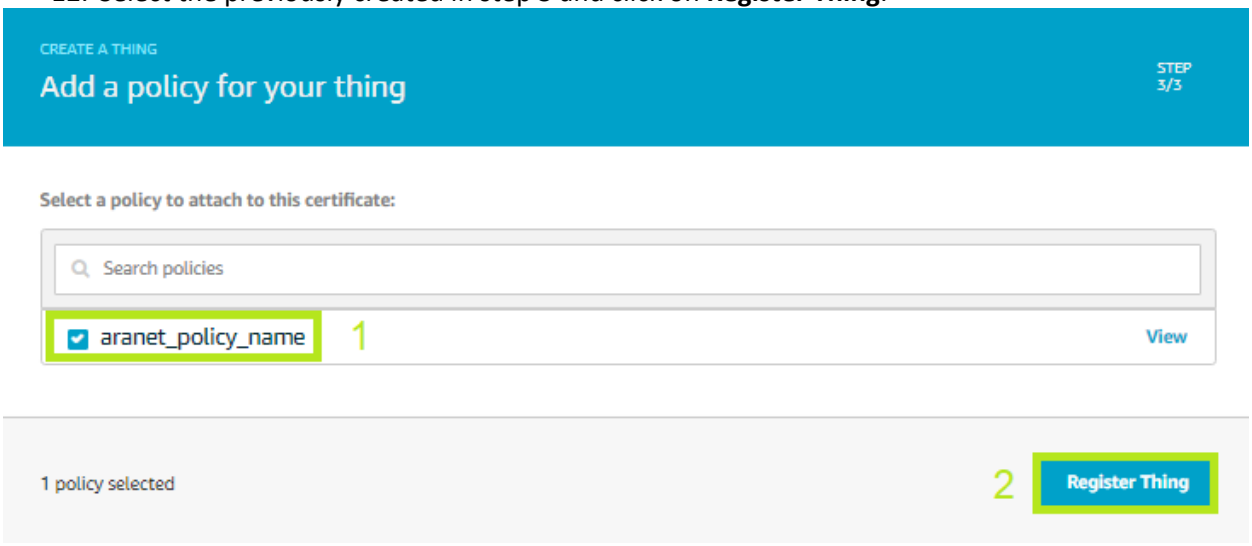


10. Save certificate file on Your computer adding extension .crt

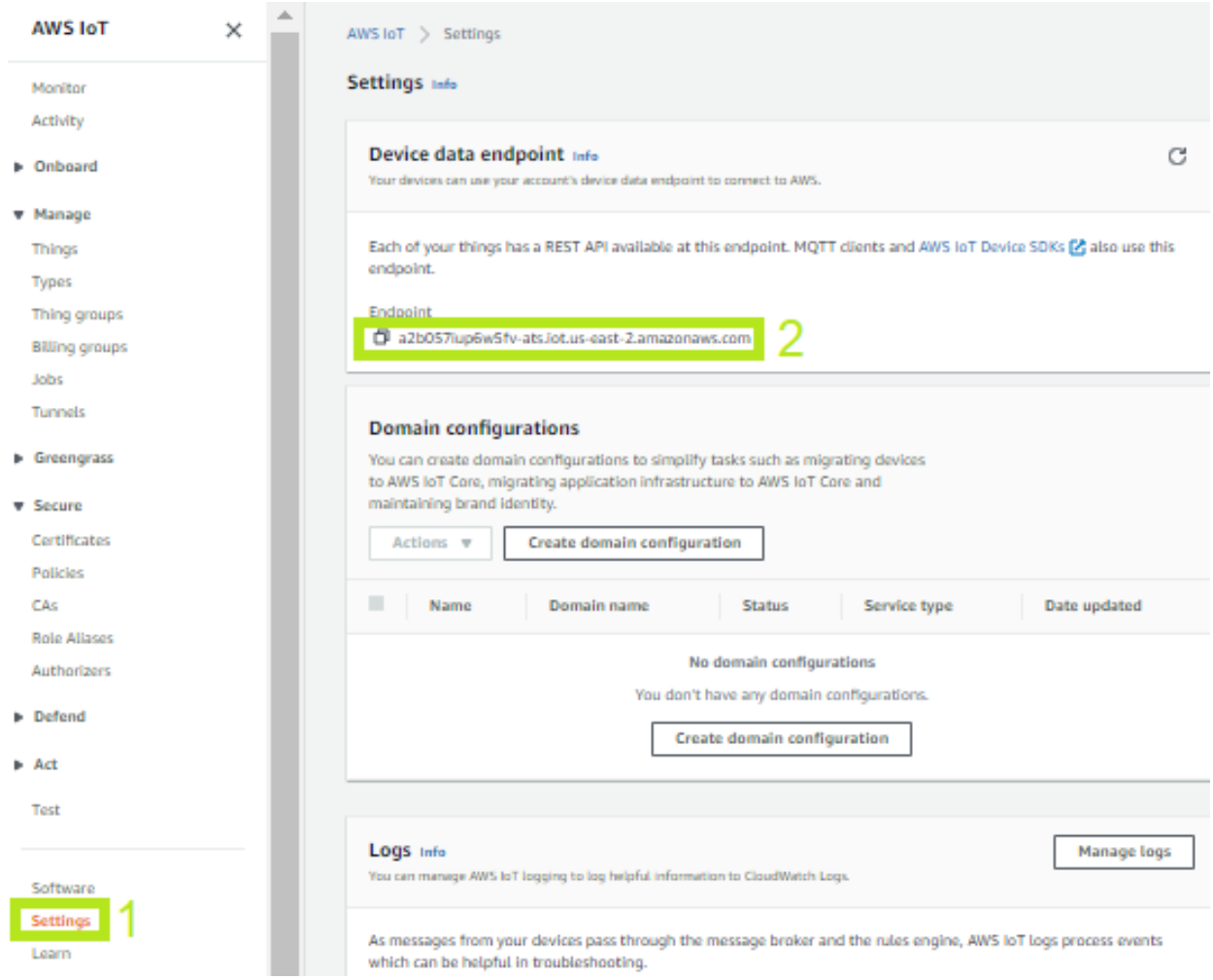
11. Next go back to the **Certificate** page and press on **Attach a policy** button:



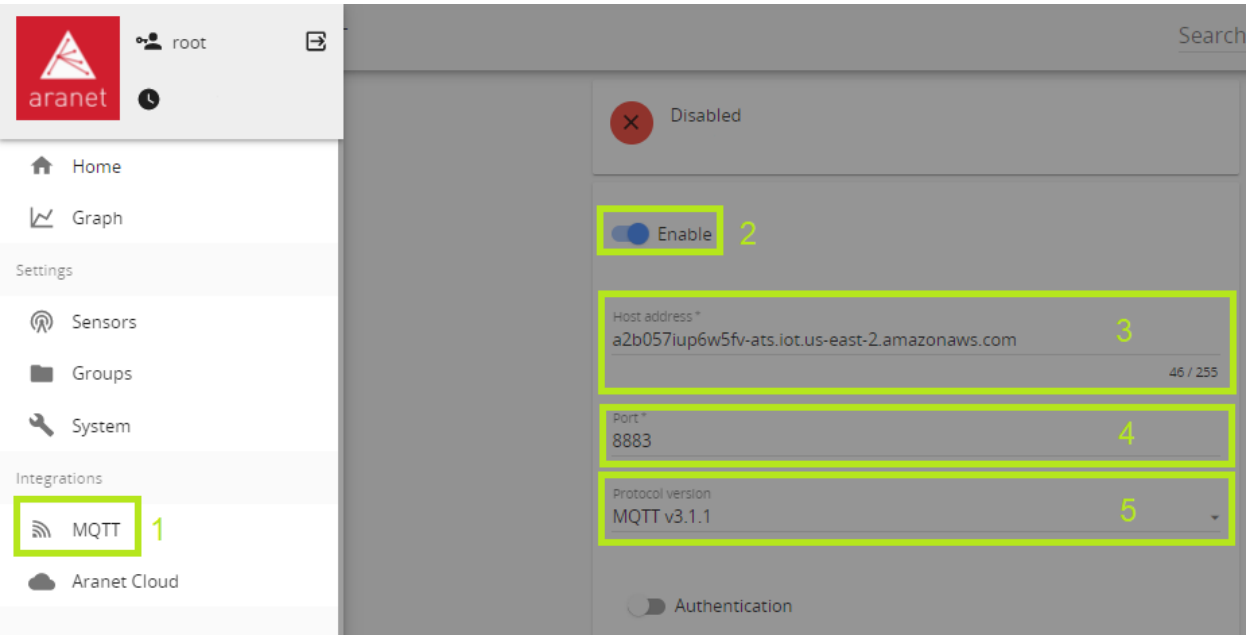
12. Select the previously created in step 3 and click on **Register Thing**:



13. Next go to section **Settings** and copy Your AWS **Endpoint** address:

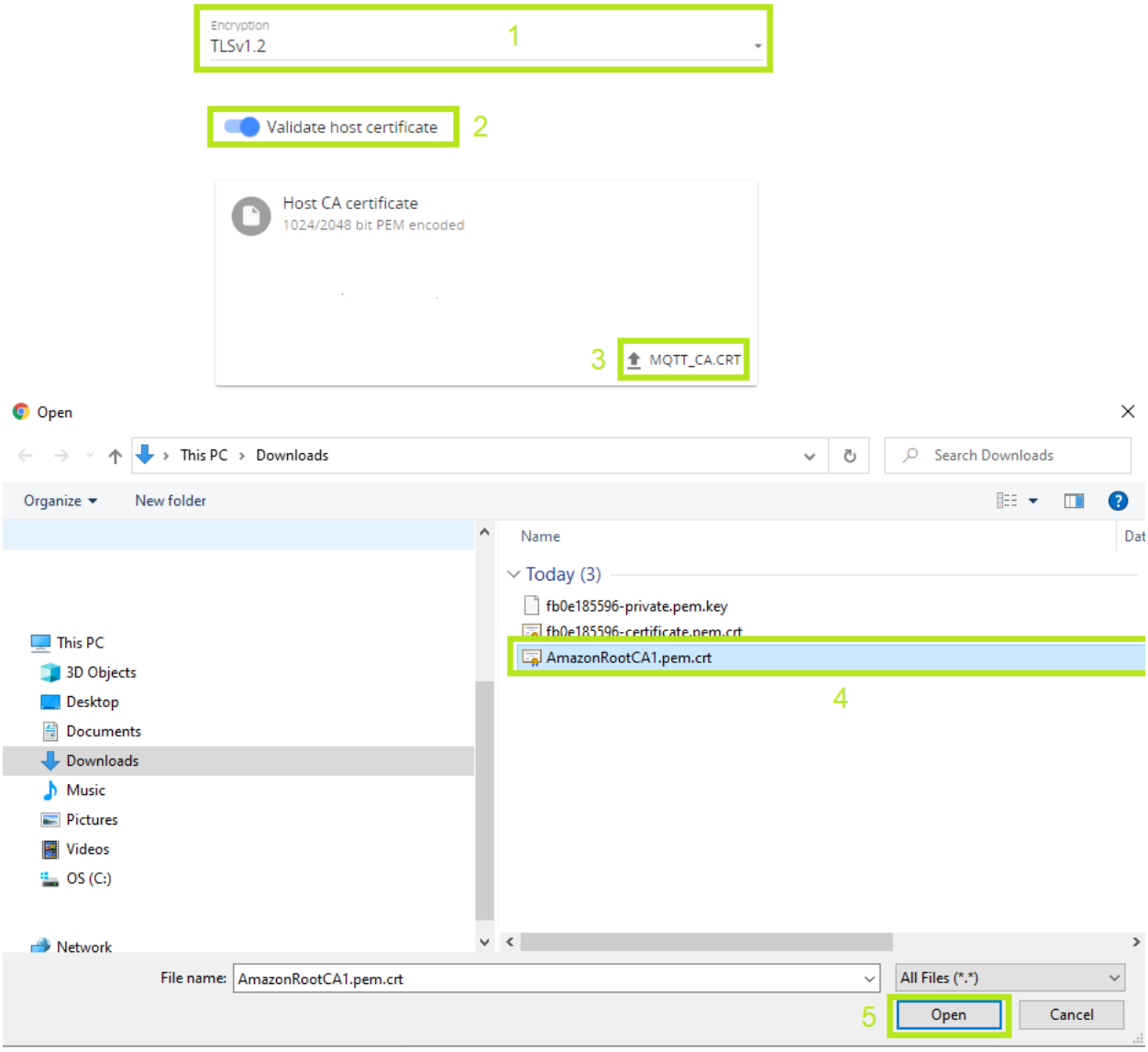


14. Now log in to Aranet PRO base station graphical user interface and go to the **MQTT** section. **Enable** MQTT connection and paste previously copied AWS Endpoint information in **Host address** field. Enter **Port** = ***8883*** and choose **Protocol version** = ***MQTT v3.1.1*** without **Authentification**:

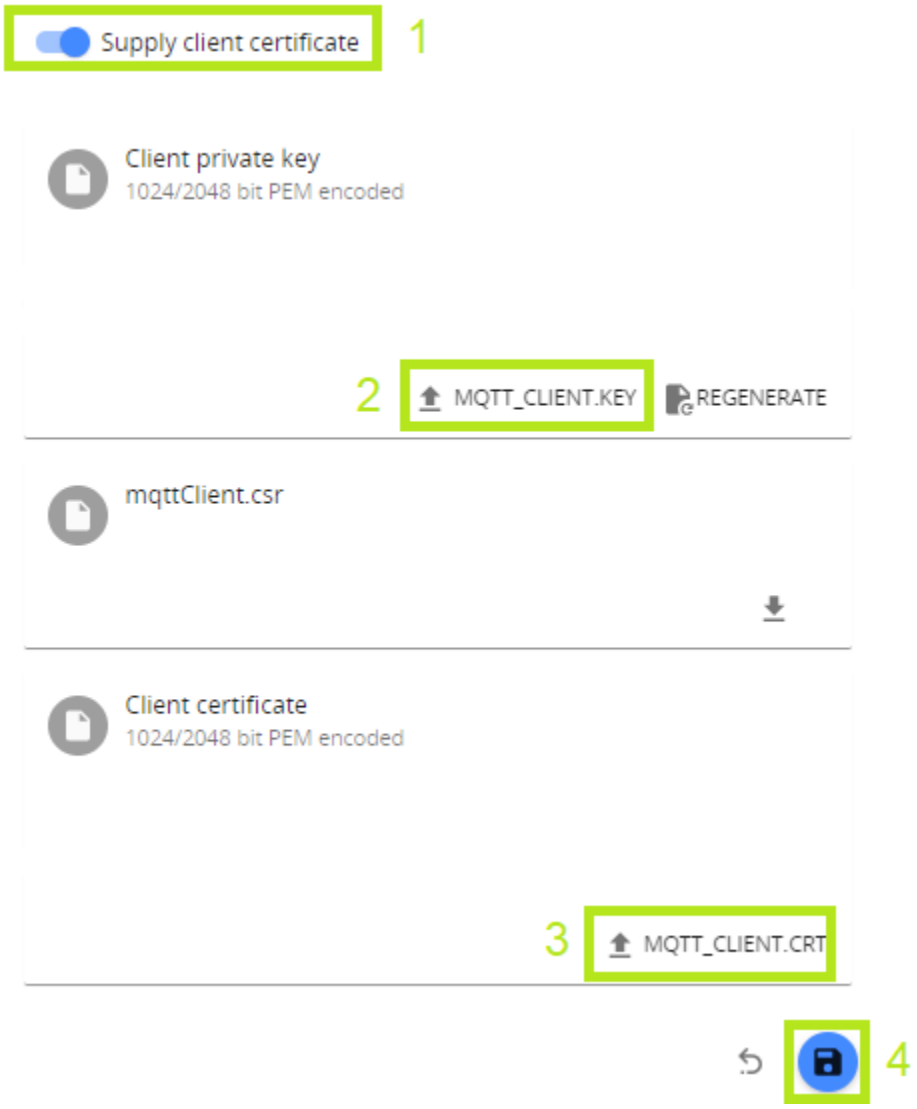


15. Next set necessary **QoS level** (***0*** or ***1***, but ***2*** is not supported by AWS) for MQTT message delivery in AWS system, their **Root topic** and **Sensor measurement format** (***raw*** or ***JSON***)

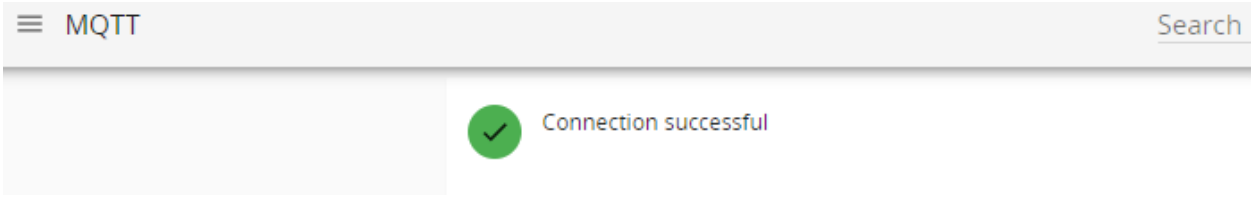
16. Then select **Encryption** = ***TLSv1.2***, enable **Validate host certificate** and clicking on **MQTT_CA.CRT** upload ***AmazonRootCA1.pem.crt*** from step 9:



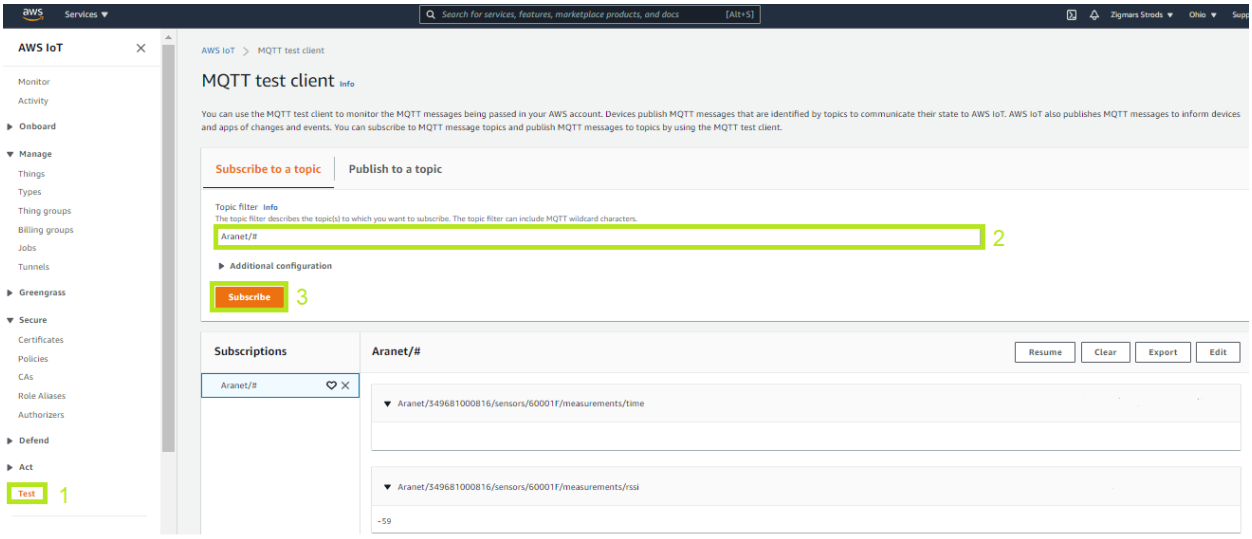
17. Finally enable **Supply client certificate**, clicking on **MQTT_CLIENT.KEY** upload ***…-private.pem.key*** file, but clicking on **MQTT_CLIENT.CRT** upload ***-certificate.pem.crt*** file and then press on **Save** icon:



And now Aranet PRO base station MQTT connection to AWS Core IoT service should be enabled and sensor data should be published on the AWS platform. Connection success should be indicated on the top of the MQTT page of the Aranet Pro base station:

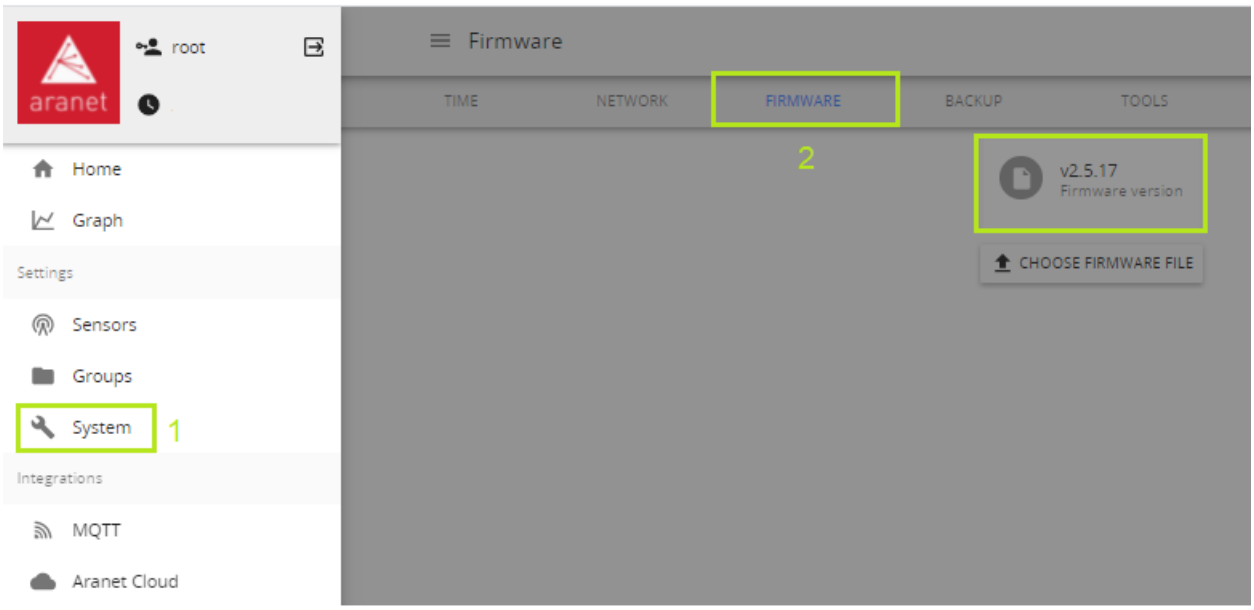


From the AWS platform data reception could be checked from section **Test** entering the Root or any other relevant topic for the previously configured Aranet PRO base station (in our example, ***Aranet/#***) and pressing on **Subscribe** button:

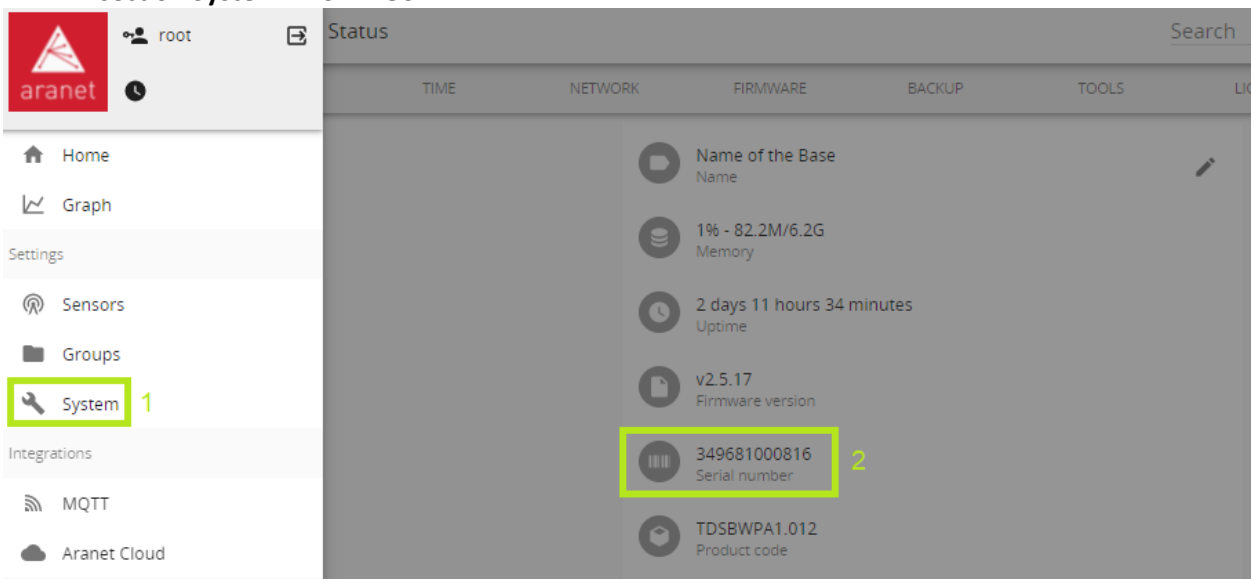


5. MQTT connection configuration with Azure IoT Hub

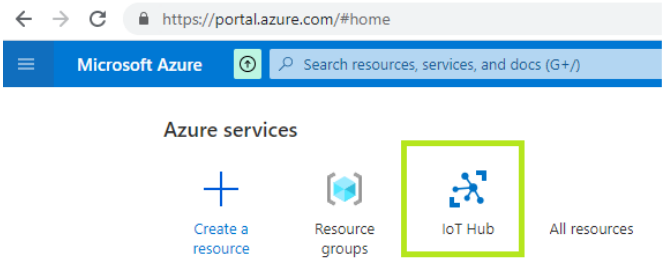
Aranet PRO base station allows also all sensor data publishing directly to Azure IoT Hub, but here the base should also have the firmware version at least 2.5.17. So before proceeding further, please first check the firmware version of the Aranet PRO base station in the graphical user interface section **System** → **FIRMWARE** and if it is older than 2.5.17, then update to the latest version available from https://aranet.com/downloads/ section of our webpage:



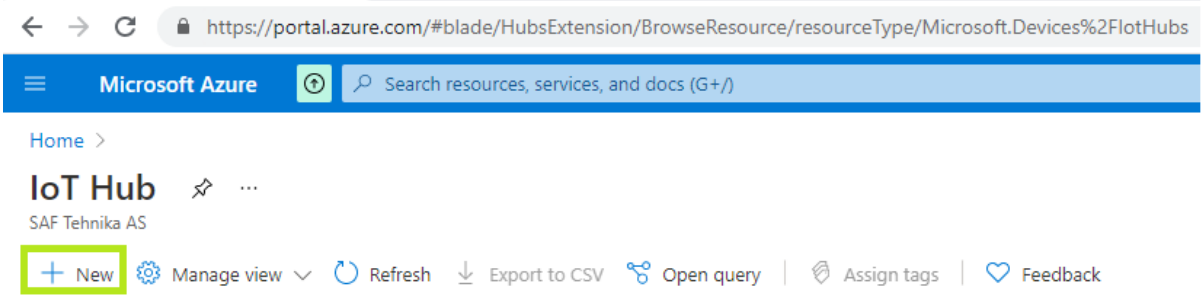
1. Next check and save for further use the **Serial number** of the Aranet PRO base station You have. It can be seen on the label on the back of the physical device or from the graphical user interface section **System → STATUS**:



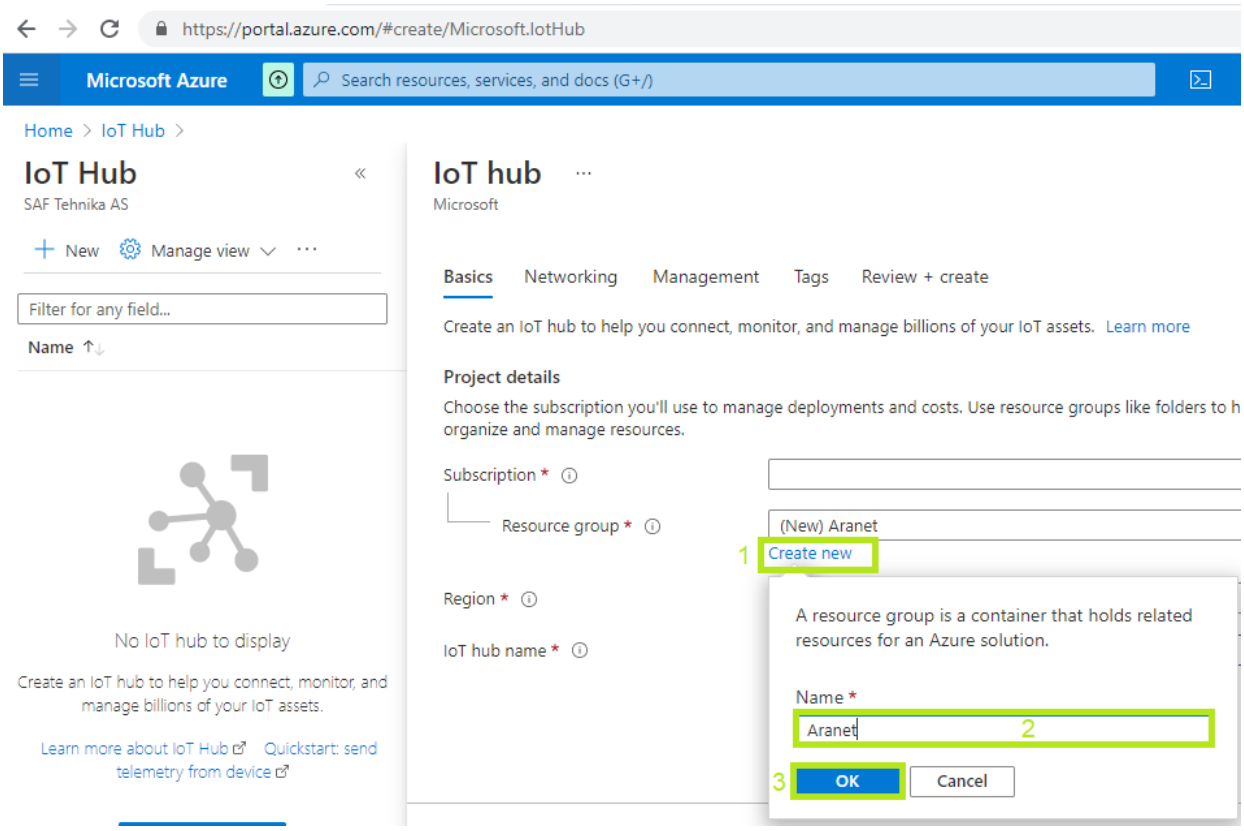
2. Log in to Your Azure account and create a new **IoT Hub** resource:



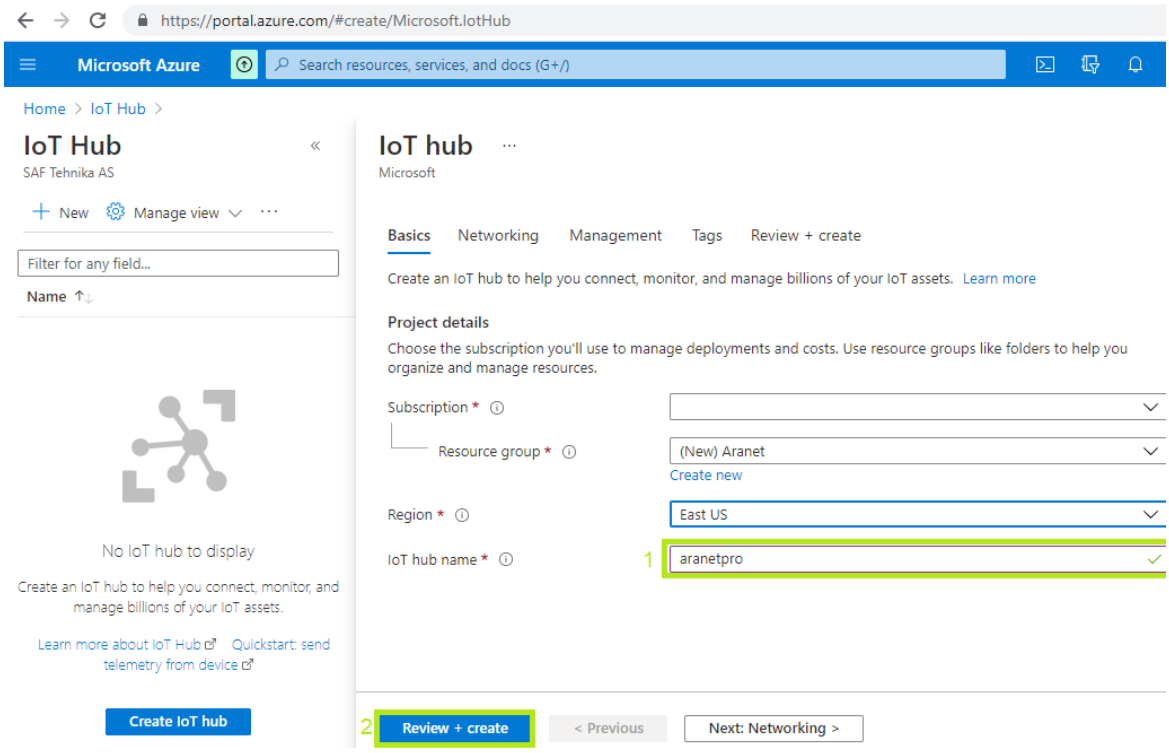
3. Click on **New**:



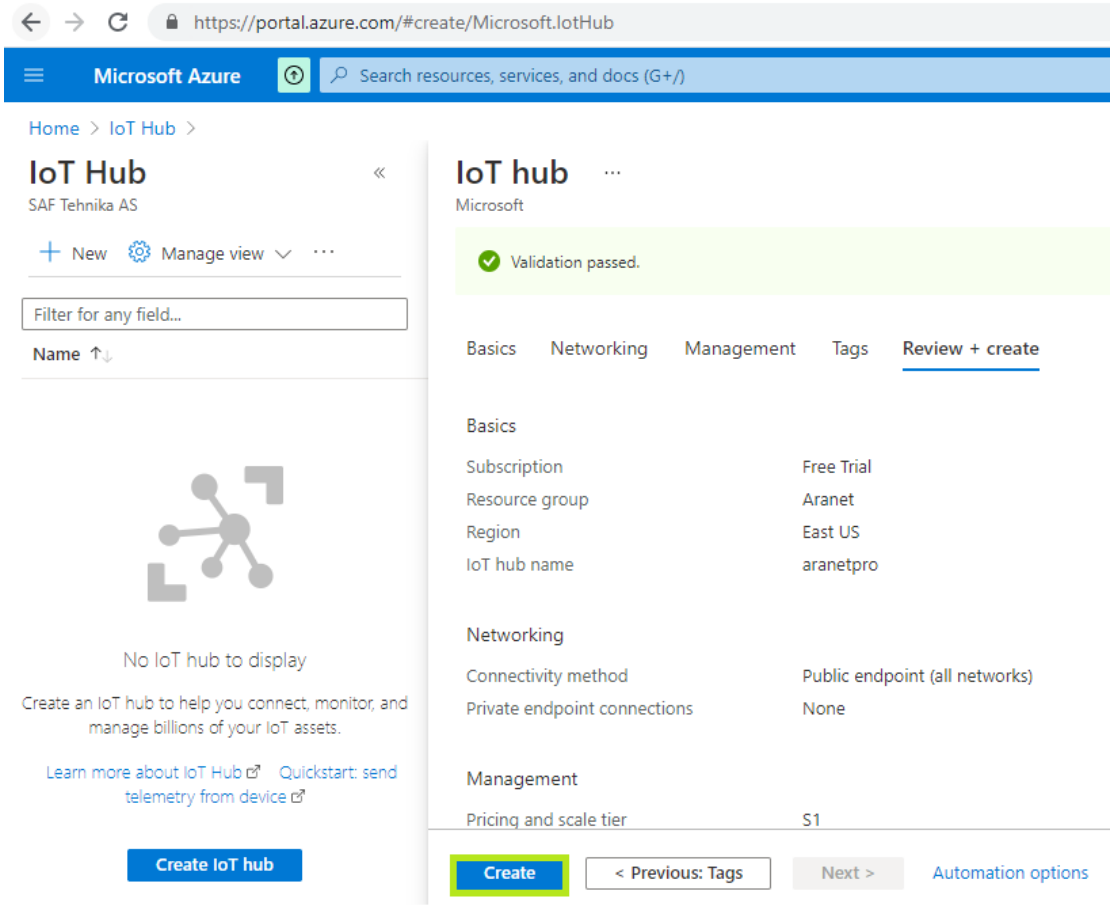
4. In the next window **Create new** for **Resource group**, enter the necessary **Name** and press on **OK** button:



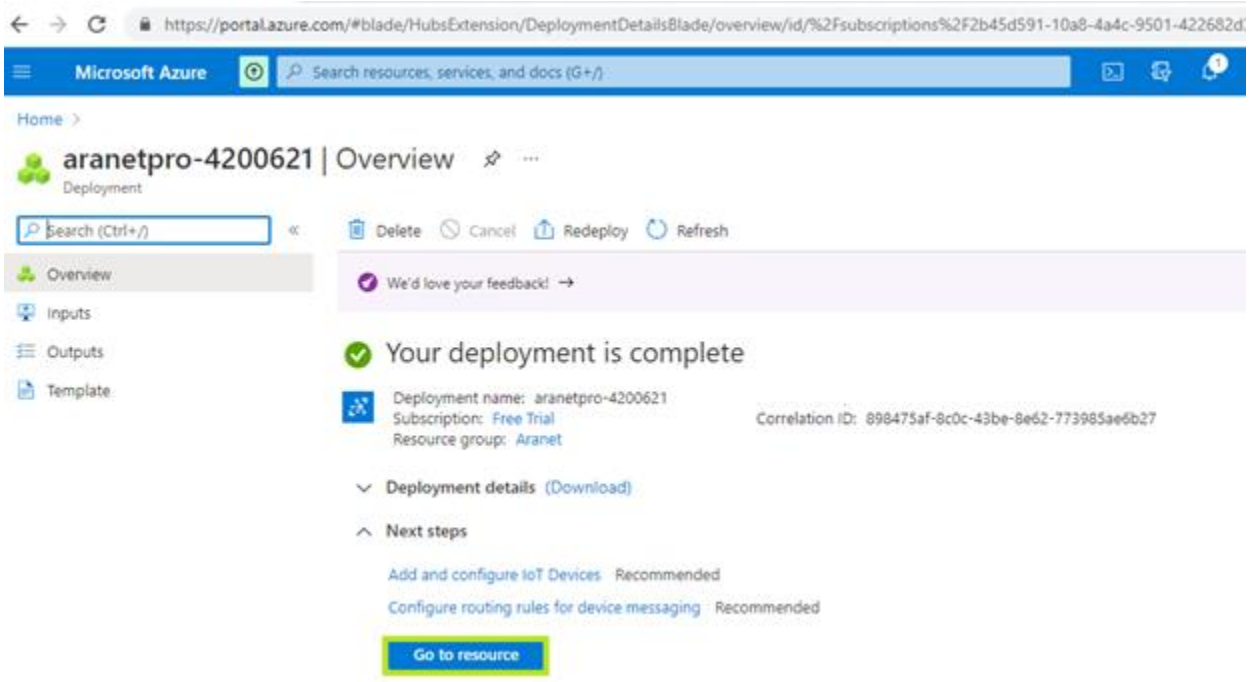
5. In the same window enter necessary **IoT hub name** and press **Review + create** button:



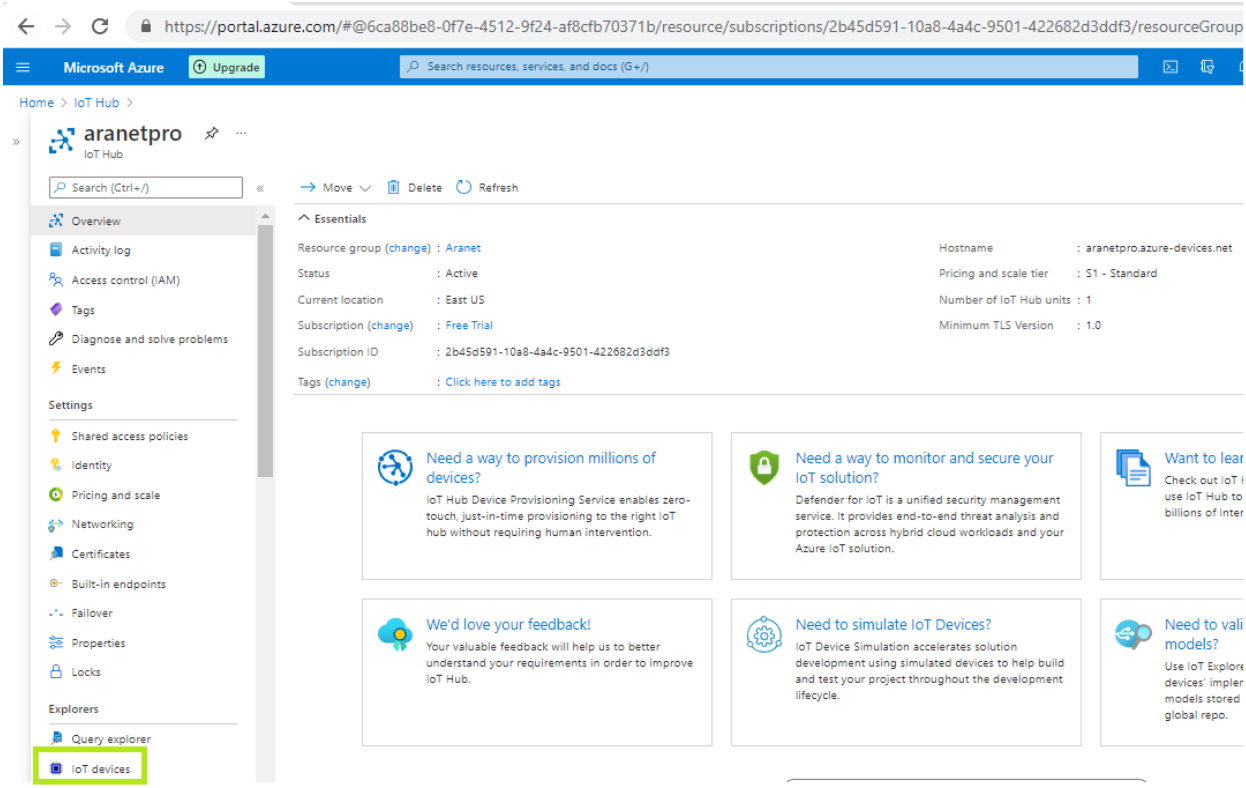
6. Then press **Create** button again:



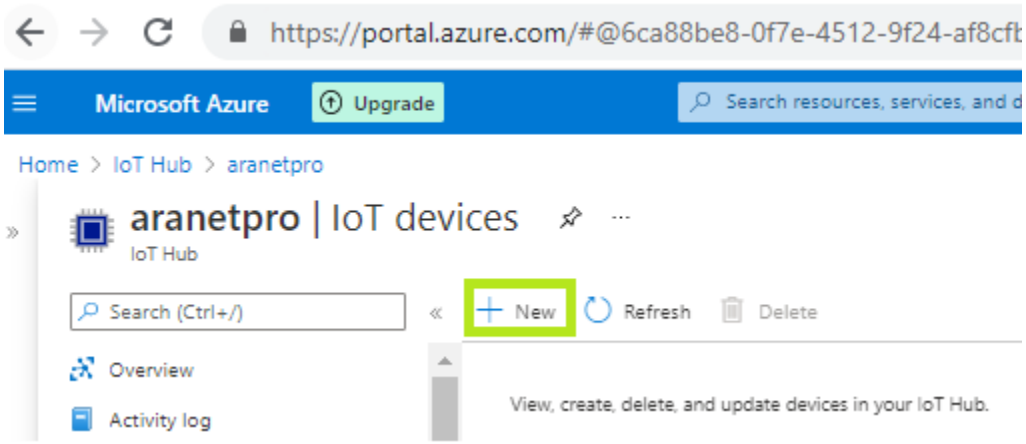
7. Wait for the Azure system to deploy the new resource and when it is done press on **Go to resource** button:



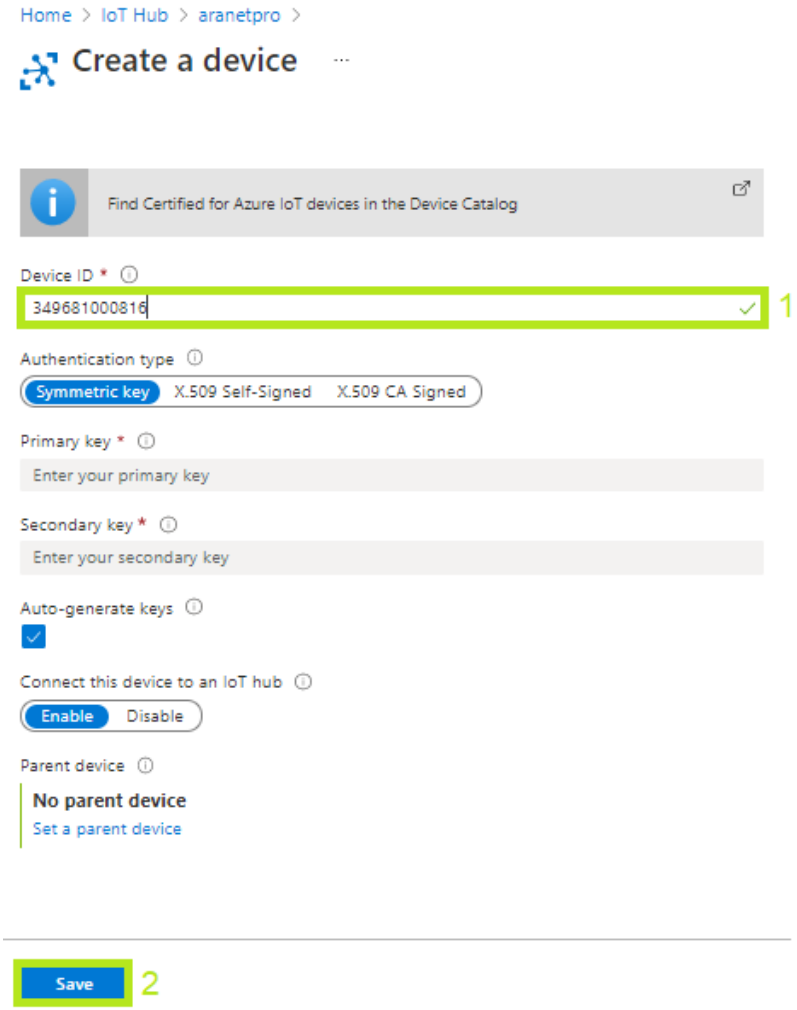
8. Then select the **IoT devices** section from the left side menu:



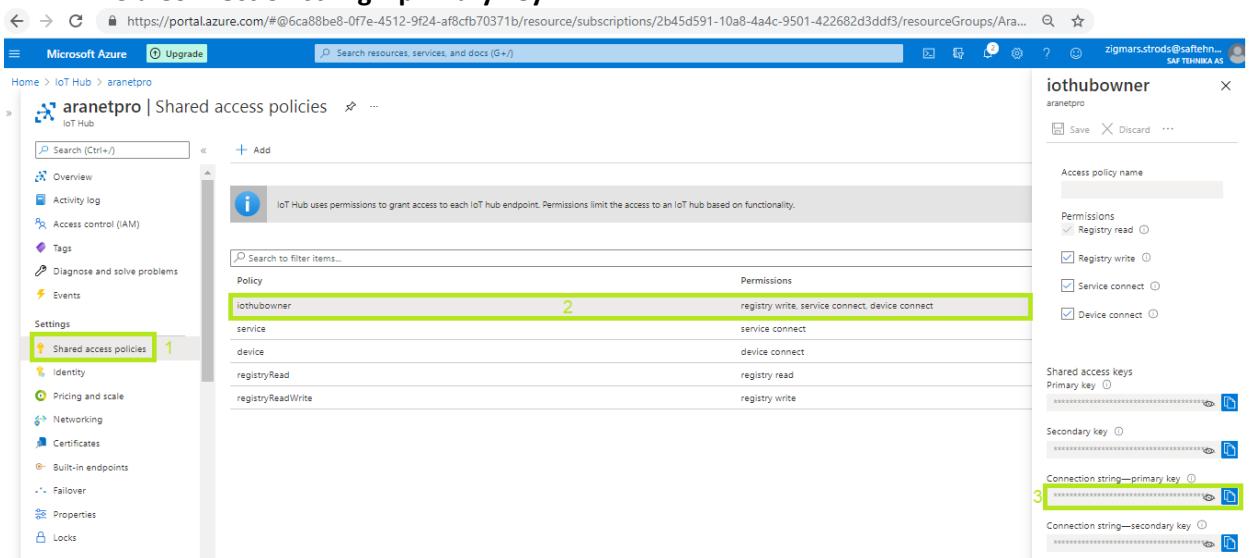
9. Press on **New**:



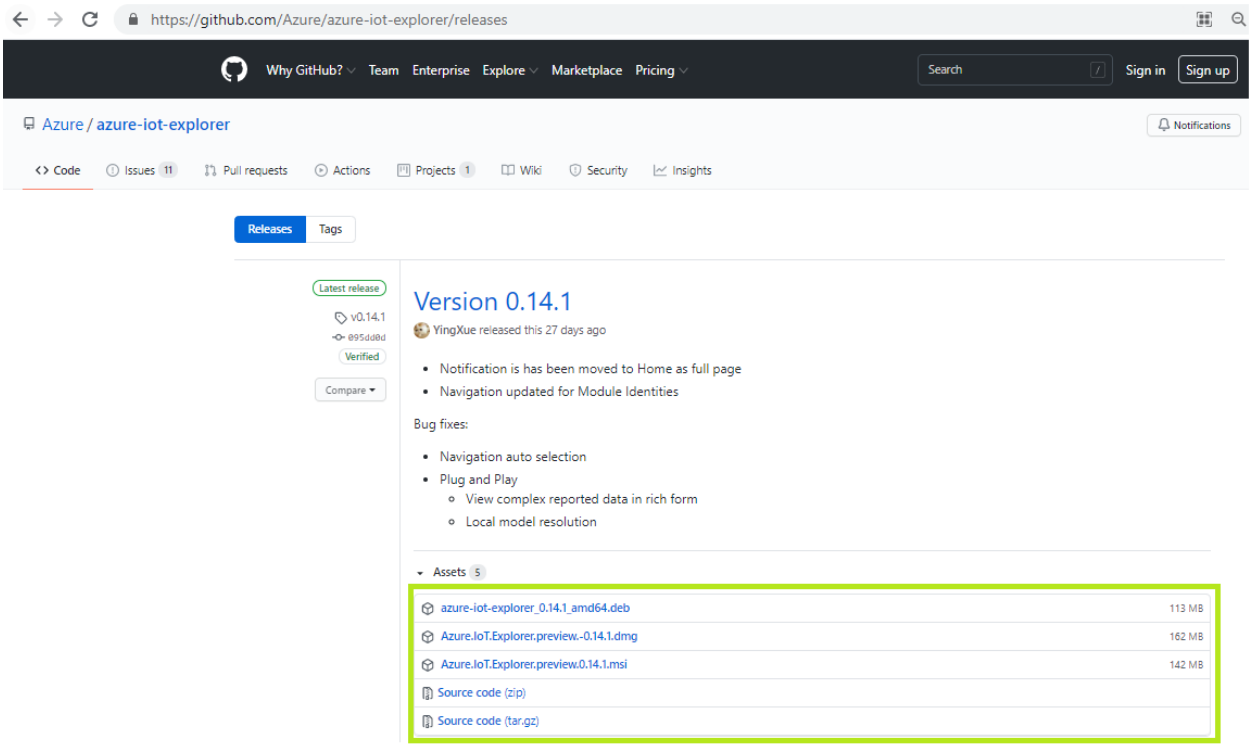
10. Just enter or copy the serial number of the Aranet PRO base station from step 1 in the **Device ID** field without any additional symbols and characters and press the **Save** button:



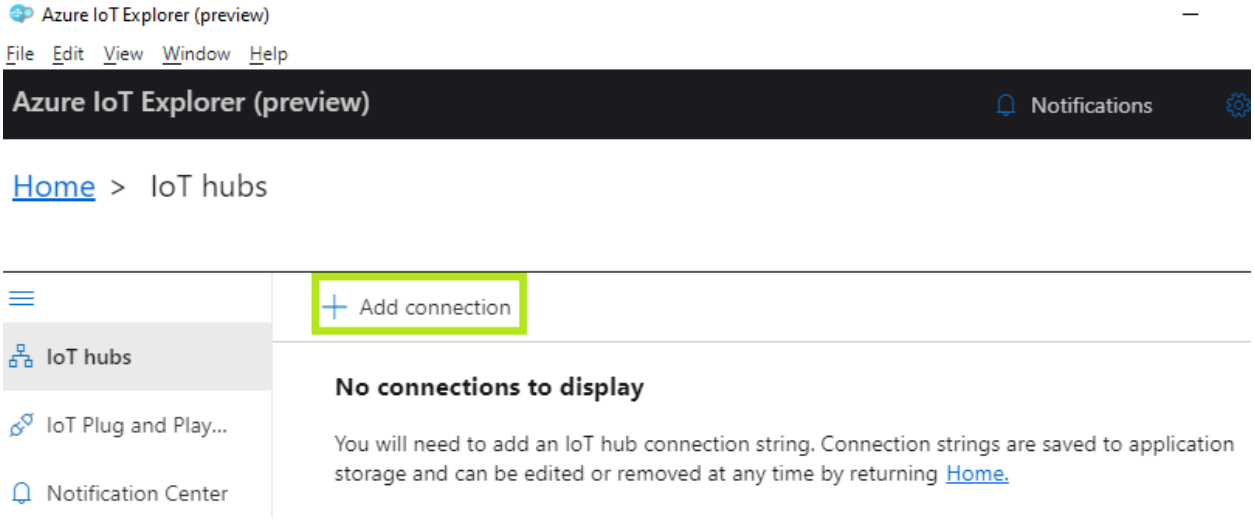
11. Next go to section **Shared access policies**, click on **iothubowner** record and copy information from field **Connection string—primary key**:



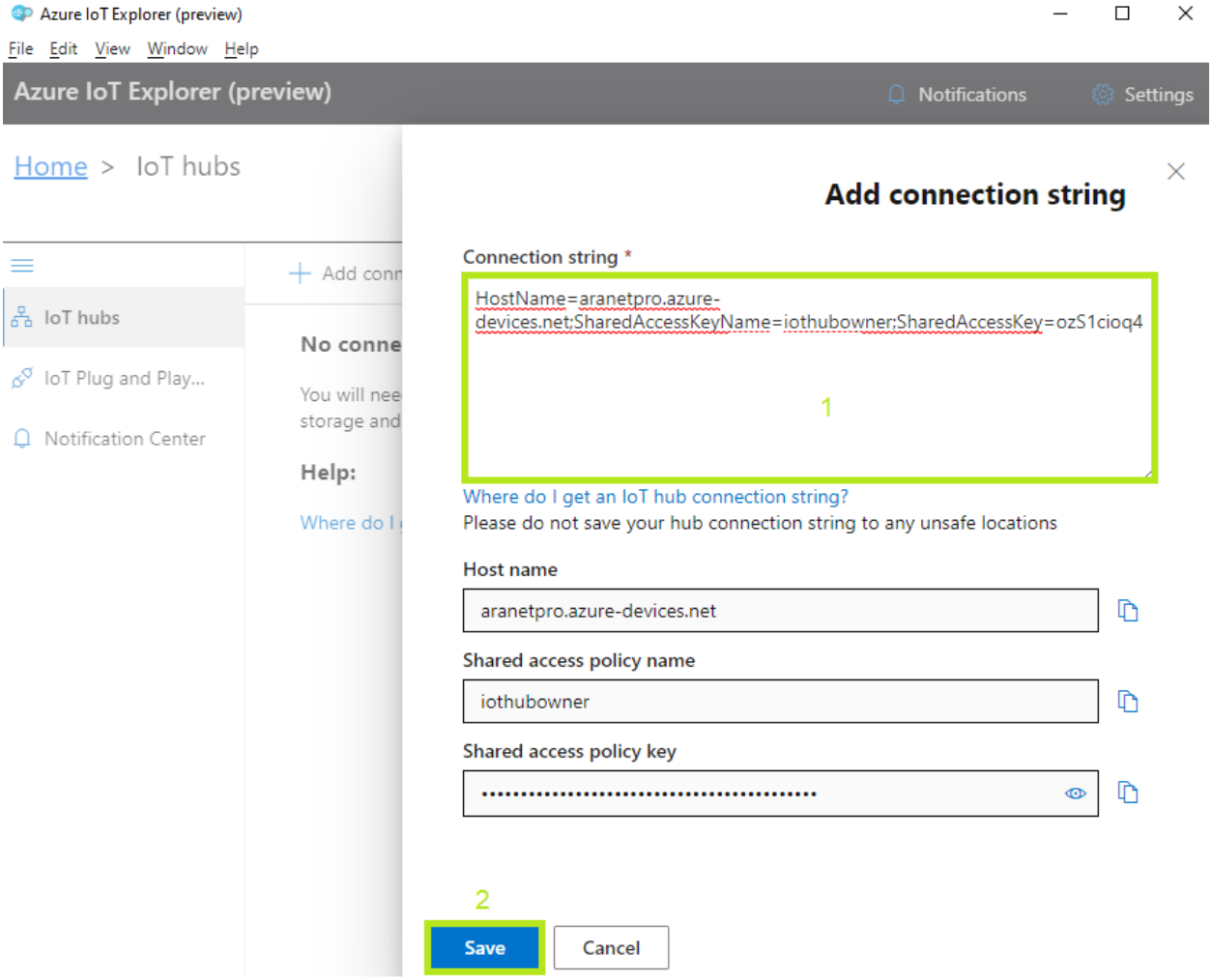
12. Now go to https://github.com/Azure/azure-iot-explorer/releases and install Azure IoT explorer on Your operating system:



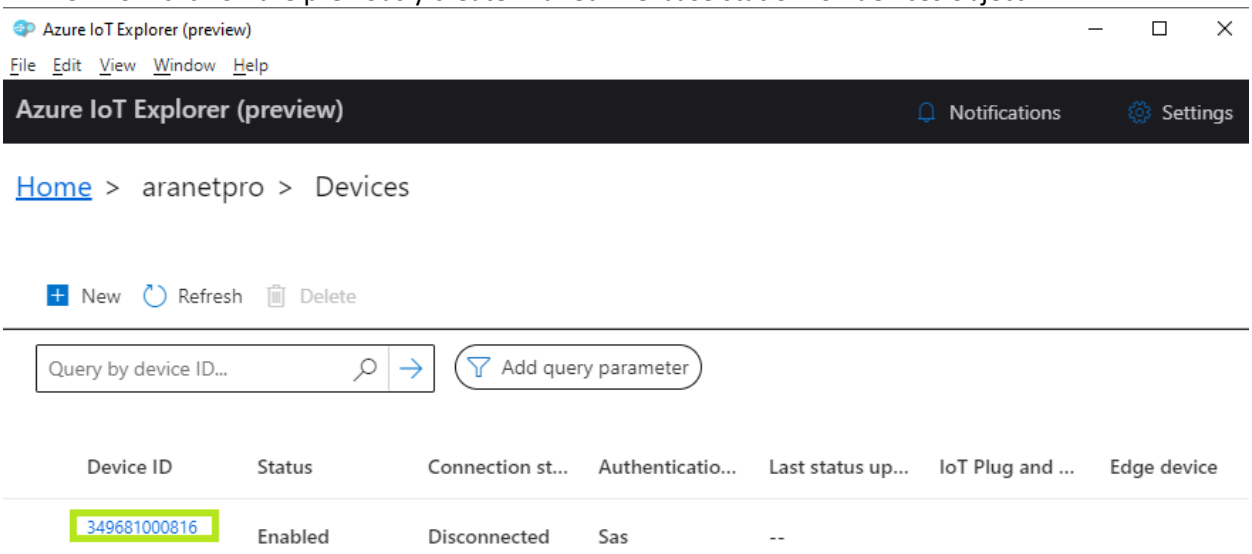
13. Launch Azure IoT explorer application and press on Add connection



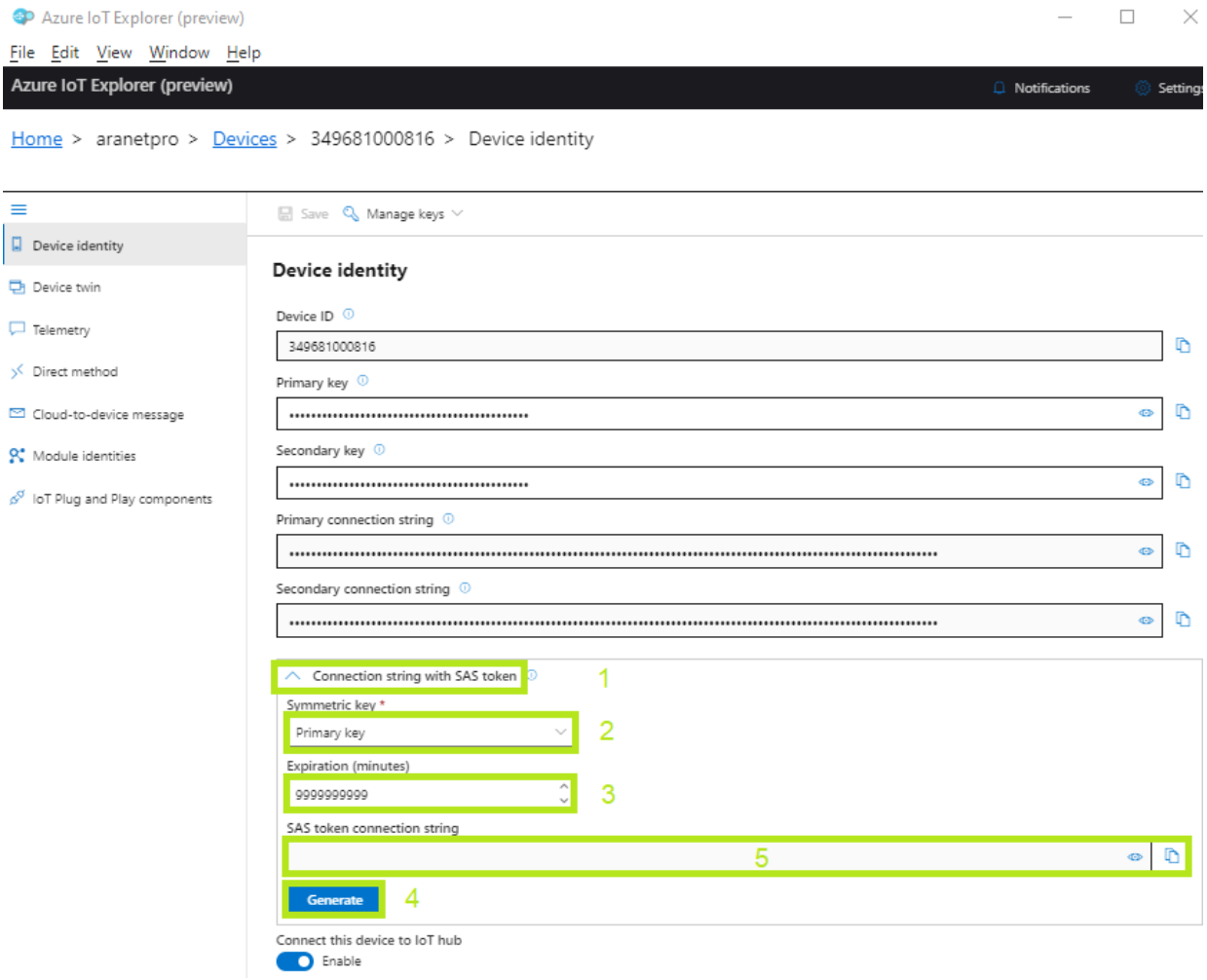
14. In the opened window paste string from step 11 and press the **Save** button:



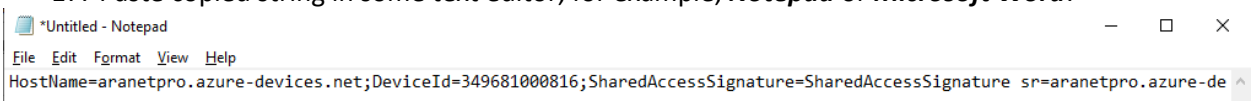
15. Now click on the previously create Aranet PRO base station IoT devices object:



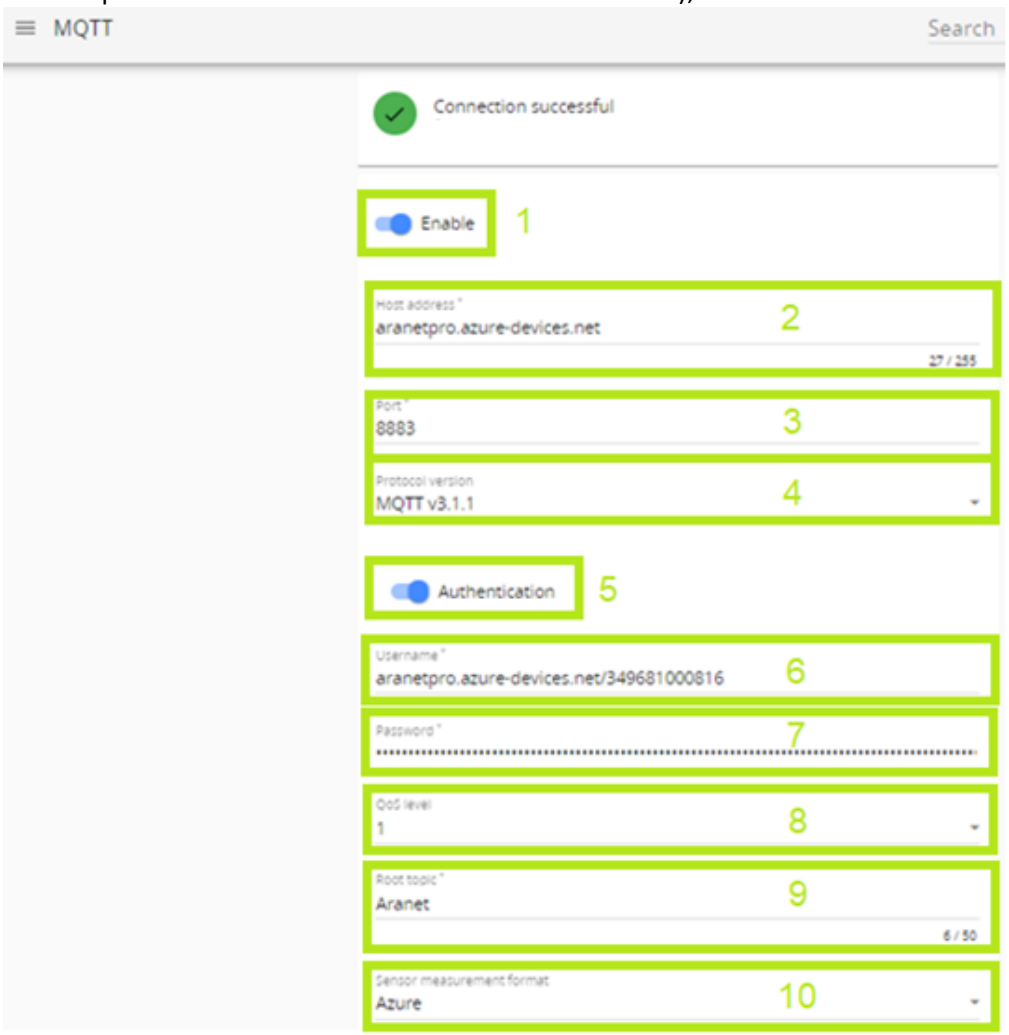
16. Click on **Connection string with SAS token** subsection, select ***Primary key*** as **Symmetric key**, enter necessary **Expiration (minutes)** time, for example, 9999999999, and press on **Generate** button and then copy generated **SAS token connection string**:



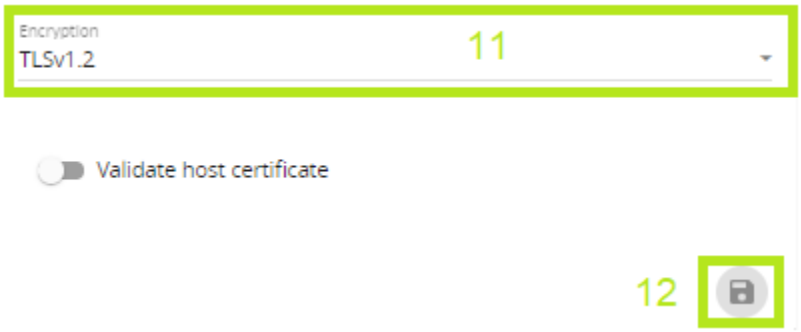
17. Paste copied string in some text editor, for example, ***Notepad*** or ***Microsoft Word***:



18. Now open Aranet Pro base station graphical user interface section **MQTT** and:
    1. **Enable** MQTT connection;
    2. paste ***HostName=*** value from the text editor in the **Host address** field (in our example ***aranetpro.azure-devices.net***);
    3. select **Port** as ***8883***;
    4. select **Protocol version** as ***MQTT v3.1.1***;
    5. ***Enable*** **Authentication**;
    6. in **Username** field paste ***HostName=*** value again, then slash "***/***" and serial number of Aranet PRO base station (in our example, ***aranetpro.azure-devices.net/349681000816***);
    7. in **Password** field paste all the text string that comes after ***SharedAccessSignature=*** (in our example, SharedAccessSignature sr=aranetpro.azure-de…. etc.;
    8. select **QoS level** as ***0*** or ***1*** (2 is not supported by Azure);
    9. enter any **Root topic** value as You want;
    10. select **Sensor measurement format** as ***Azure*** (sensor measurements will not be accepted by Azure platform if the format is selected as raw or JSON);



11. select **Encryption** as ***TLSv1.2***;
12. and finally press on the **Save** icon:



19. If the connection to the Azure platform is successful then the corresponding success message will be shown on top of the MQTT page in Aranet PRO base station graphical user interface. Additionally user can check what data is published on the Azure IoT hub from the Azure IoT explorer application by clicking on the **Telemetry** section and then pressing on **Start** button: