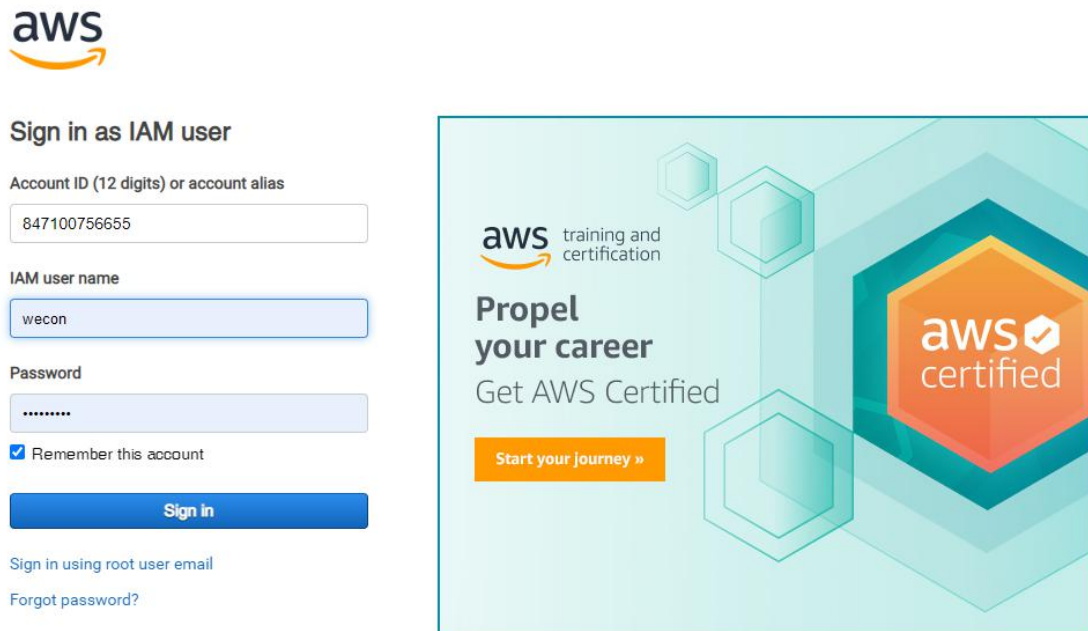




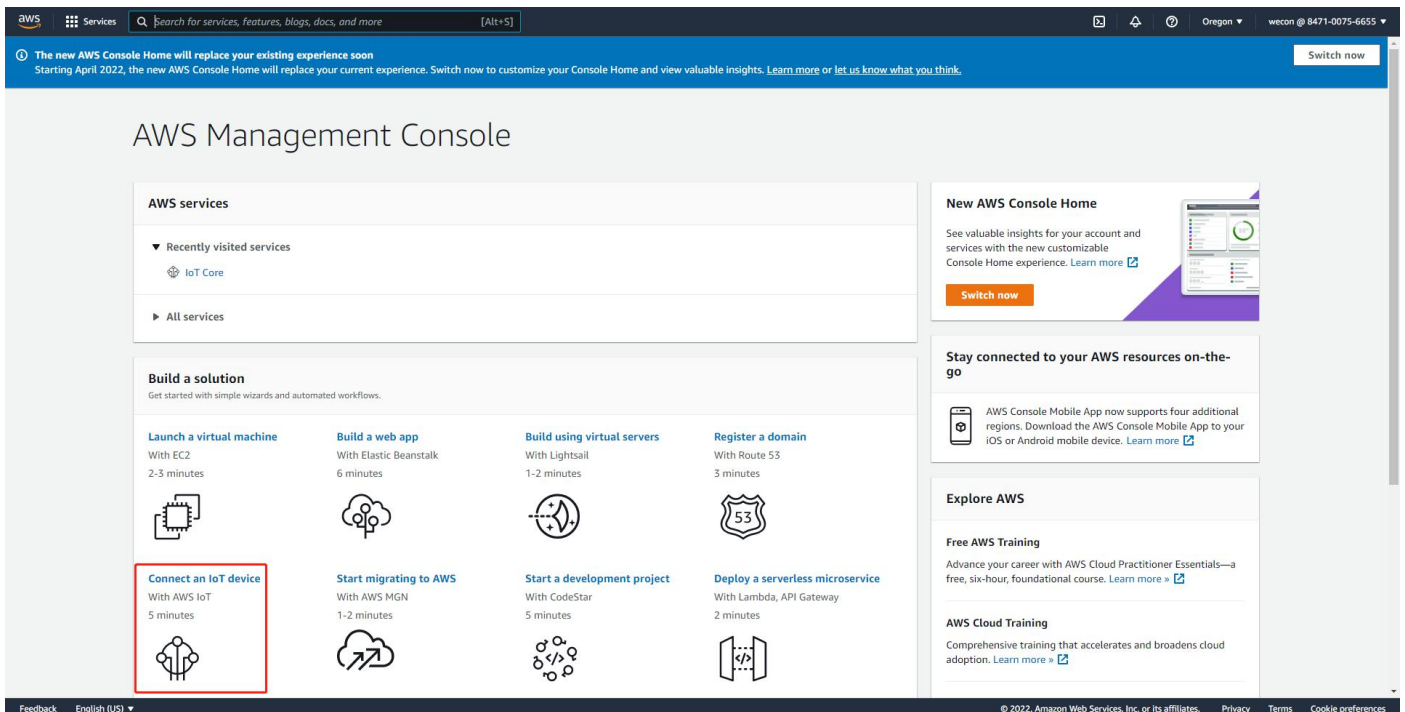
V-Box OpenCloud Mode Connect to AWS IoT

1. Log in AWS

Login aws account and click“Connect an IoT device”



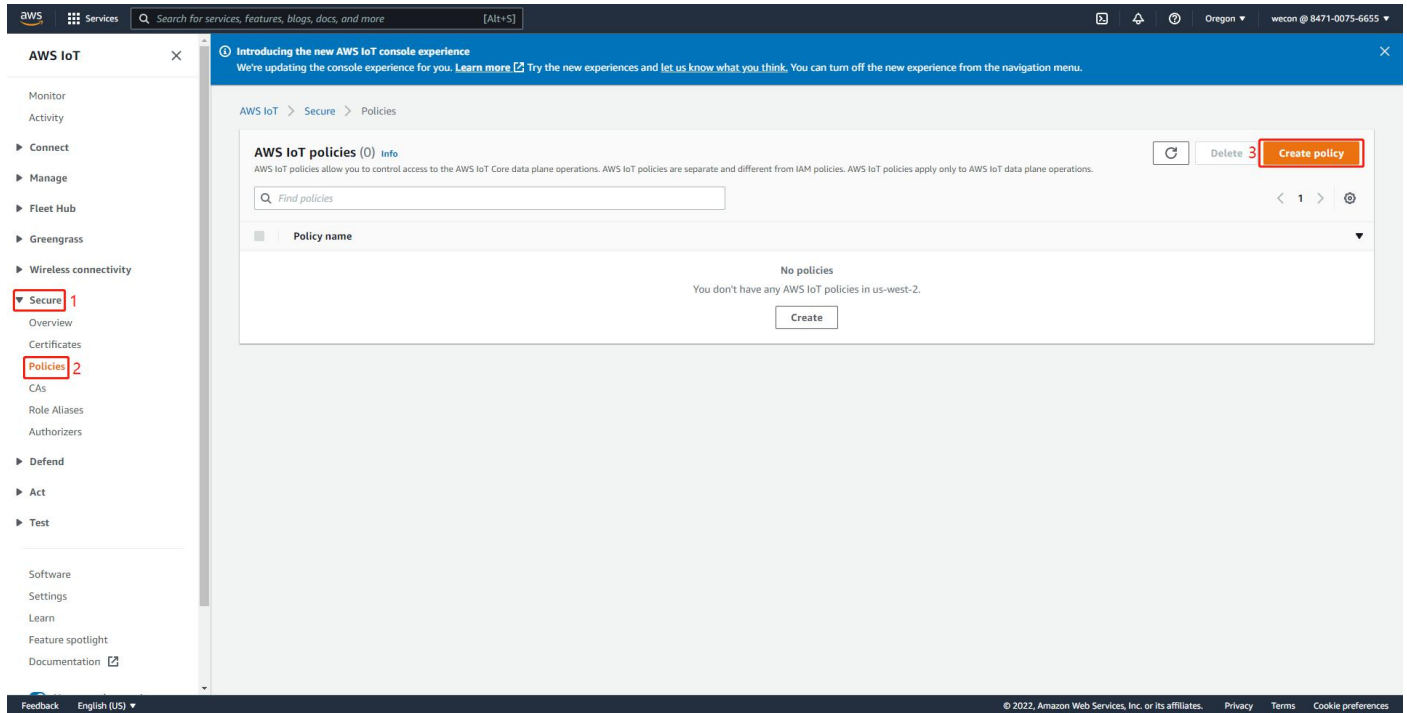
The image shows the AWS sign-in page. On the left, there is a form to sign in as an IAM user. It includes fields for Account ID (12 digits) or account alias (filled with 847100756655), IAM user name (filled with wecon), and Password (masked with dots). There is a checkbox for "Remember this account" and a "Sign in" button. Below the button are links for "Sign in using root user email" and "Forgot password?". On the right, there is a promotional banner for AWS training and certification, featuring the text "Propel your career Get AWS Certified" and a "Start your Journey" button, along with the AWS Certified logo.



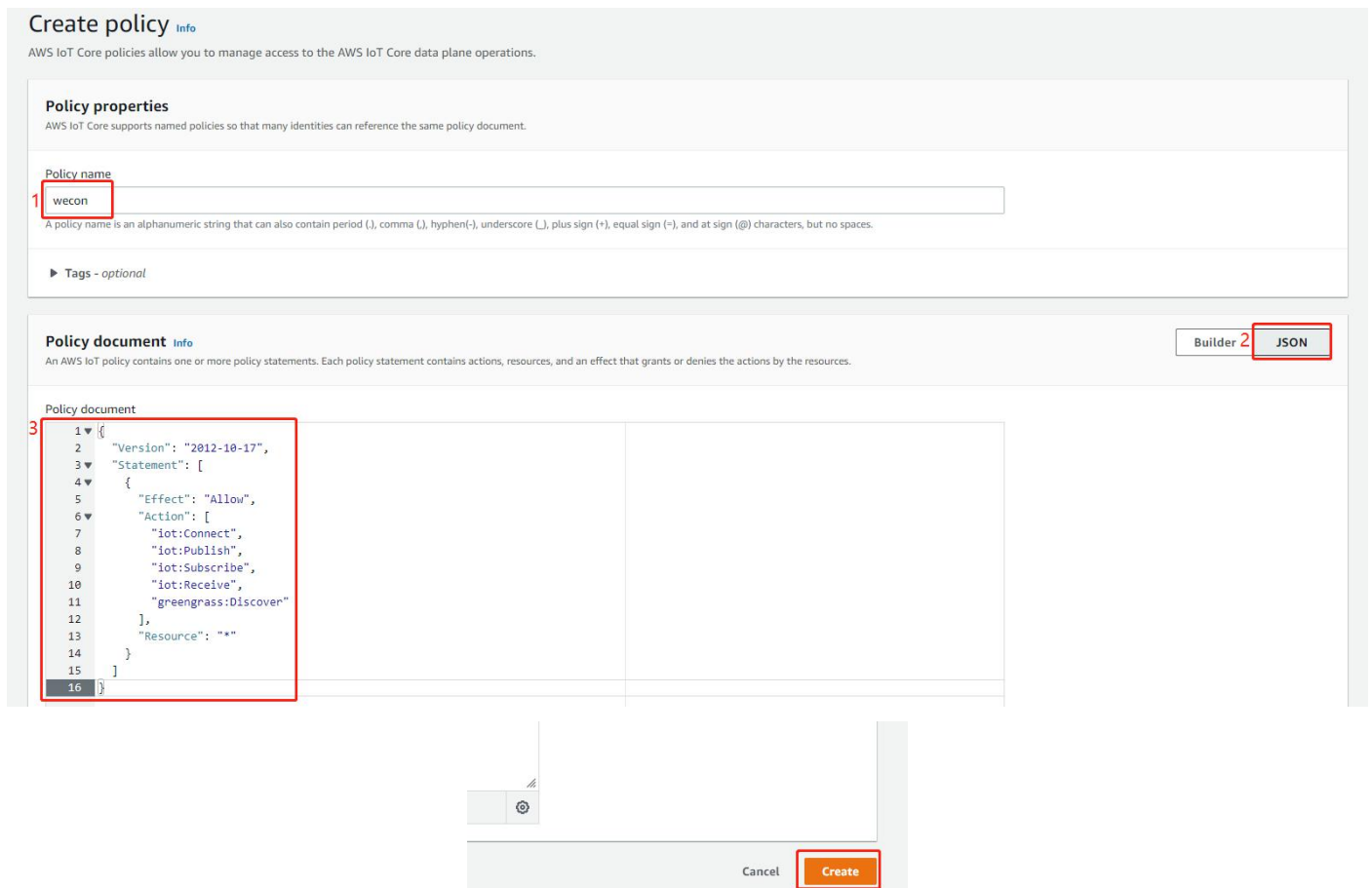
The image shows the AWS Management Console dashboard. At the top, there is a navigation bar with the AWS logo, a search bar, and a user profile dropdown. Below the navigation bar, there is a banner about the new AWS Console Home. The main content area is divided into several sections. On the left, there is a "Recently visited services" section showing "IoT Core". Below that, there is a "Build a solution" section with several cards: "Launch a virtual machine", "Build a web app", "Build using virtual servers", "Register a domain", "Connect an IoT device" (highlighted with a red box), "Start migrating to AWS", "Start a development project", and "Deploy a serverless microservice". On the right, there is a "New AWS Console Home" section, a "Stay connected to your AWS resources on-the-go" section, and an "Explore AWS" section with links to "Free AWS Training" and "AWS Cloud Training".

2. Create policy

Click “Secure”--->“Policies”--->“Create policy”--->Click “Create”



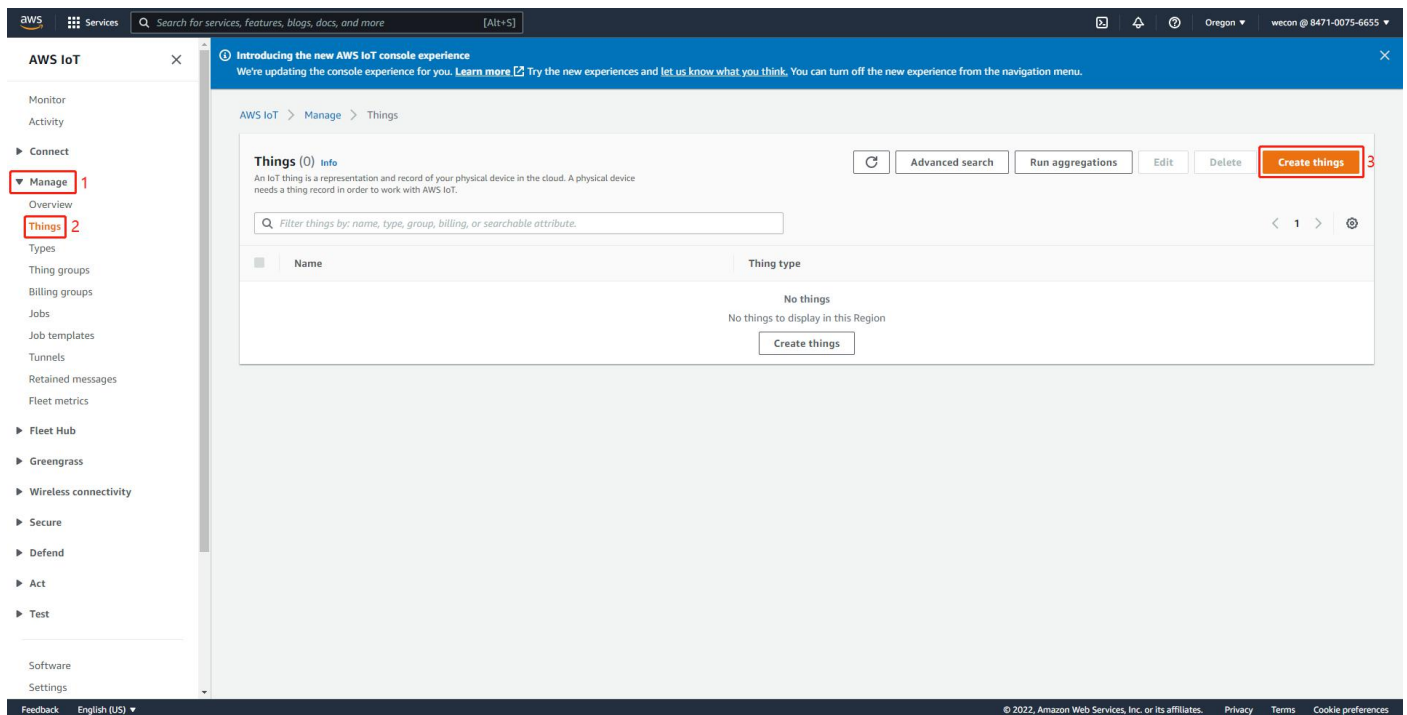
Name the policy--->Click “JSON”--->Copy the following content--->Click “Create”



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "greengrass:Discover"
      ],
      "Resource": "*"
    }
  ]
}
```

3. Create things

Click “Manage”--->“Things”--->“Create things”--->“Create single thing”



Number of things to create

☒ Create single thing

Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

☐ Create many things

Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

Cancel

Next

Name the thing--->Click "Next"

Thing properties [Info](#)

Thing name

1

AWS

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

▶ Thing type - optional

▶ Searchable thing attributes - optional

▶ Thing groups - optional

▶ Billing group - optional

Device Shadow [Info](#)

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.

☒ No shadow

☐ Named shadow

Create multiple shadows with different names to manage access to properties, and logically group your devices properties.

☐ Unnamed shadow (classic)

A thing can have only one unnamed shadow.

Cancel 2

Next

Select the way to create certificate

Device certificate

1
☒
Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

☐
Use my certificate
Use a certificate signed by your own certificate authority.

☐
Upload CSR
Register your CA and use your own certificates on one or many devices.

☐
Skip creating a certificate at this time
You can create a certificate for this thing and attach a policy to the certificate at a later time.

Cancel
Previous 2
Next

Select policy

Policies (1/1)

Select up to 10 policies to attach to this certificate.

< 1 >

<input checked="" type="checkbox"/>	Name
<input checked="" type="checkbox"/>	wecon 1

Cancel
Previous 2
Create thing

Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate


You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate
25d0e54aa0e...te.pem.crt

Deactivate certificate
Download

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

 This is the only time you can download the key files for this certificate.

Public key file
25d0e54aa0ec863e9c11091...3f0c480-public.pem.key

Download

Private key file
25d0e54aa0ec863e9c11091...f0c480-private.pem.key

Download

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1

Download

Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3

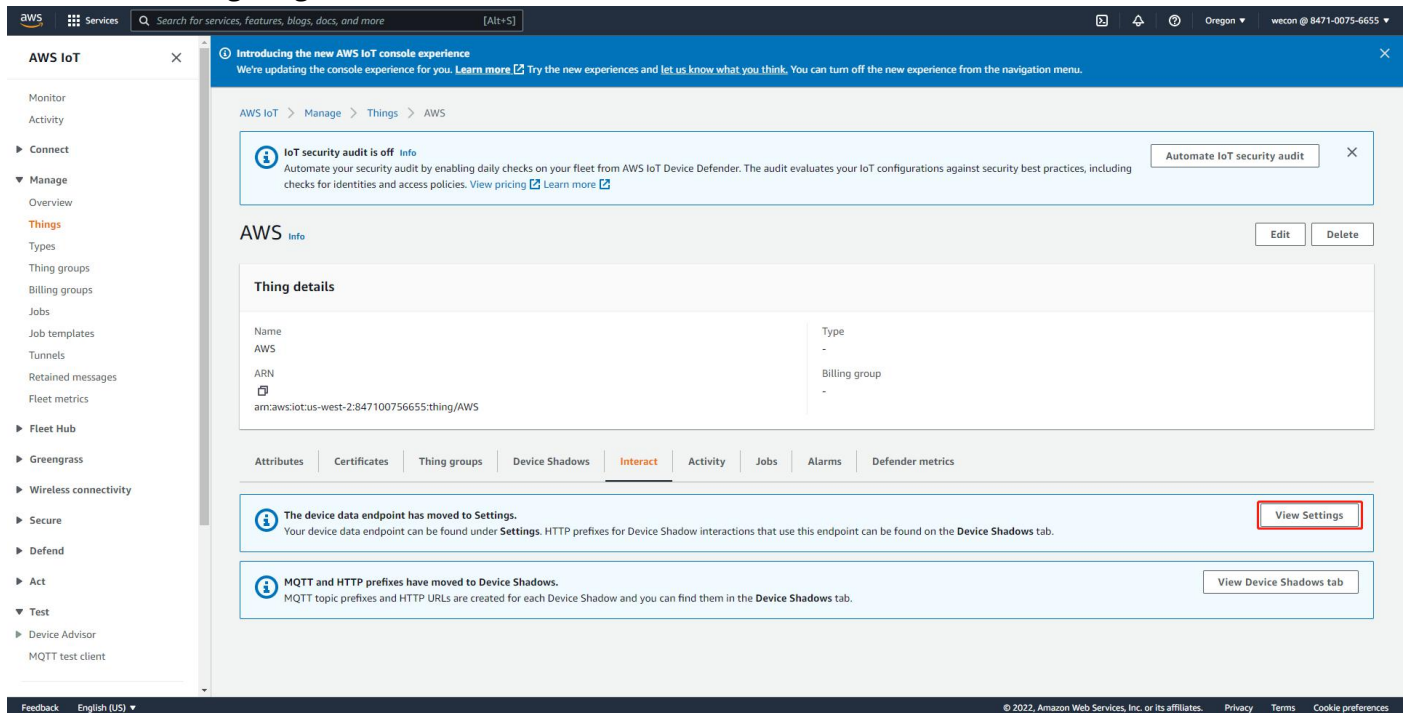
Download

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. [Learn more](#)

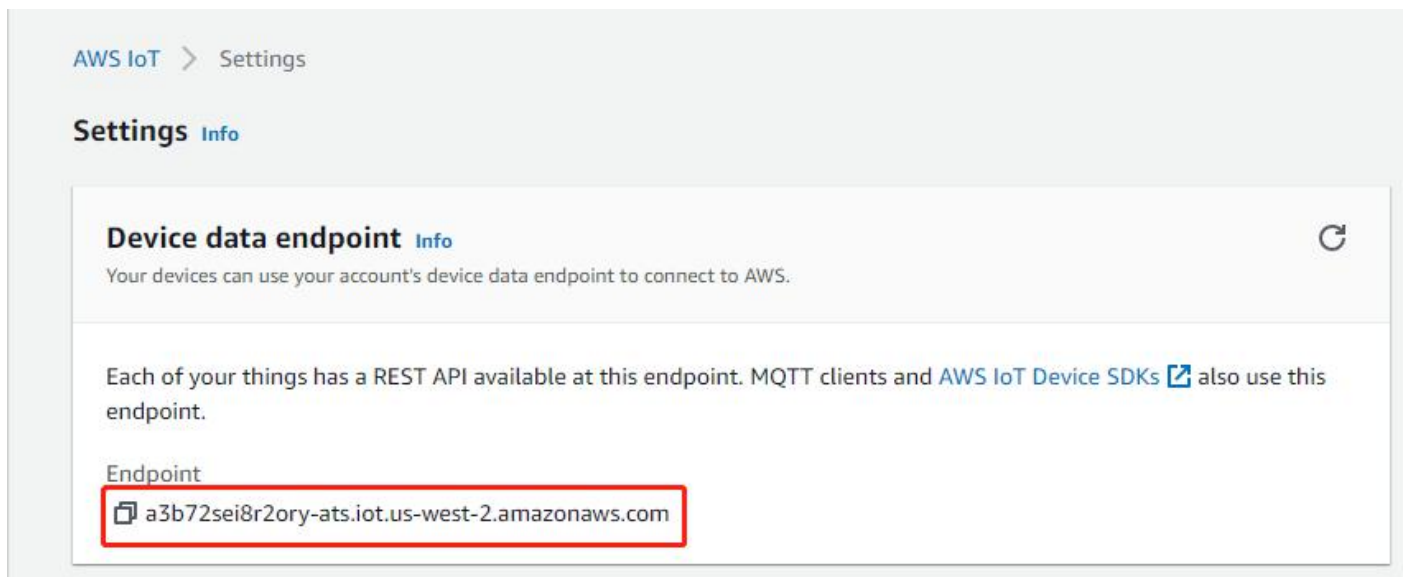
Done

4. Test with MQTT.fx tool

Click “View Setting” to get the “Broker Address”

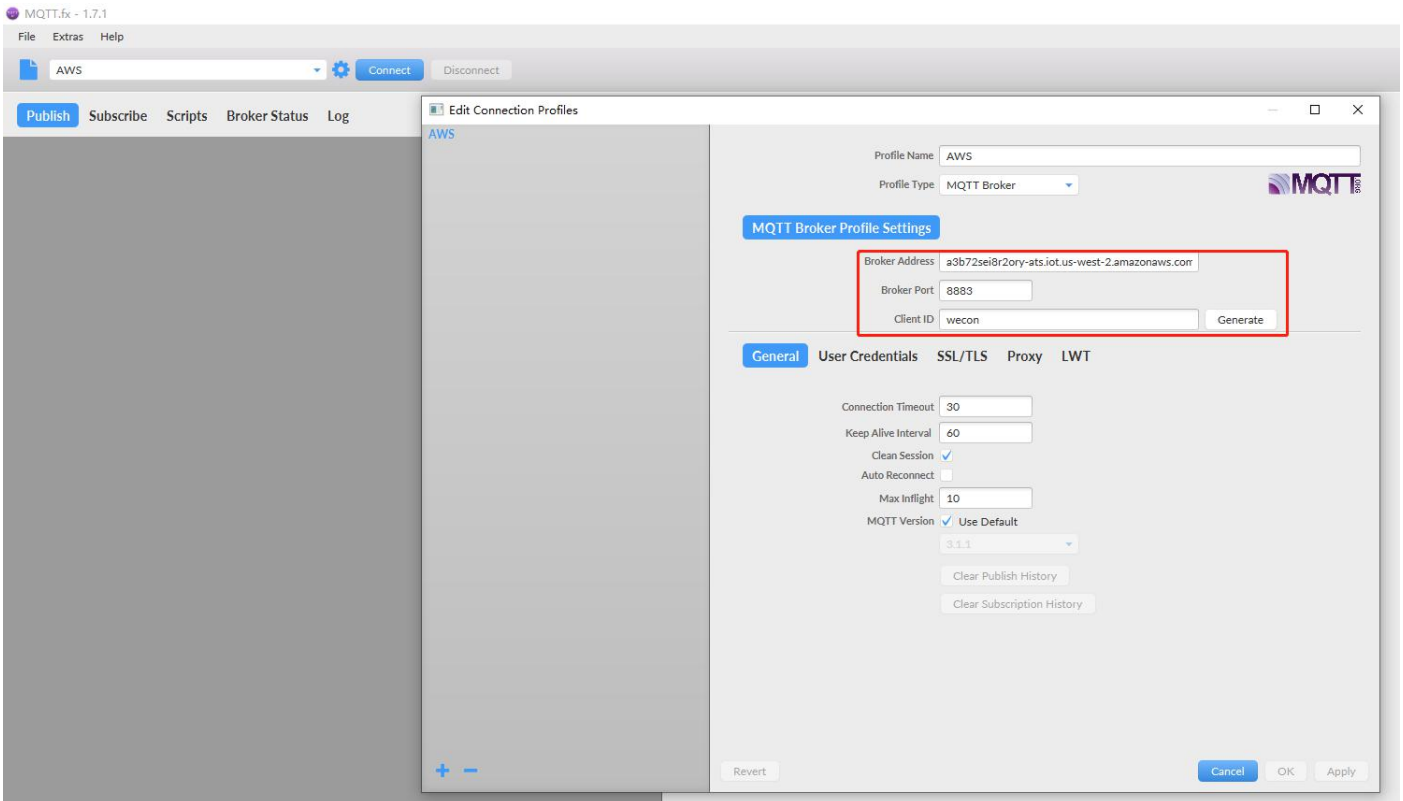


The screenshot shows the AWS IoT console interface. On the left is a navigation menu with options like Monitor, Activity, Connect, Manage, Overview, Things, Types, Thing groups, Billing groups, Jobs, Job templates, Tunnels, Retained messages, Fleet metrics, Fleet Hub, Greengrass, Wireless connectivity, Secure, Defend, Act, Test, Device Advisor, and MQTT test client. The main content area shows the 'Interact' tab for a device. At the top, there's a notification about the new AWS IoT console experience. Below that, there's a section for 'AWS IoT security audit is off'. The 'Thing details' section shows the Name as 'AWS', ARN as 'arn:aws:iot:us-west-2:847100756655:thing/AWS', and Type as 'Billing group'. The 'Interact' tab is selected, showing various sub-tabs like Attributes, Certificates, Thing groups, Device Shadows, Interact, Activity, Jobs, Alarms, and Defender metrics. A notification states that the device data endpoint has moved to Settings, with a 'View Settings' button highlighted by a red box. Another notification mentions that MQTT and HTTP prefixes have moved to Device Shadows, with a 'View Device Shadows tab' button.

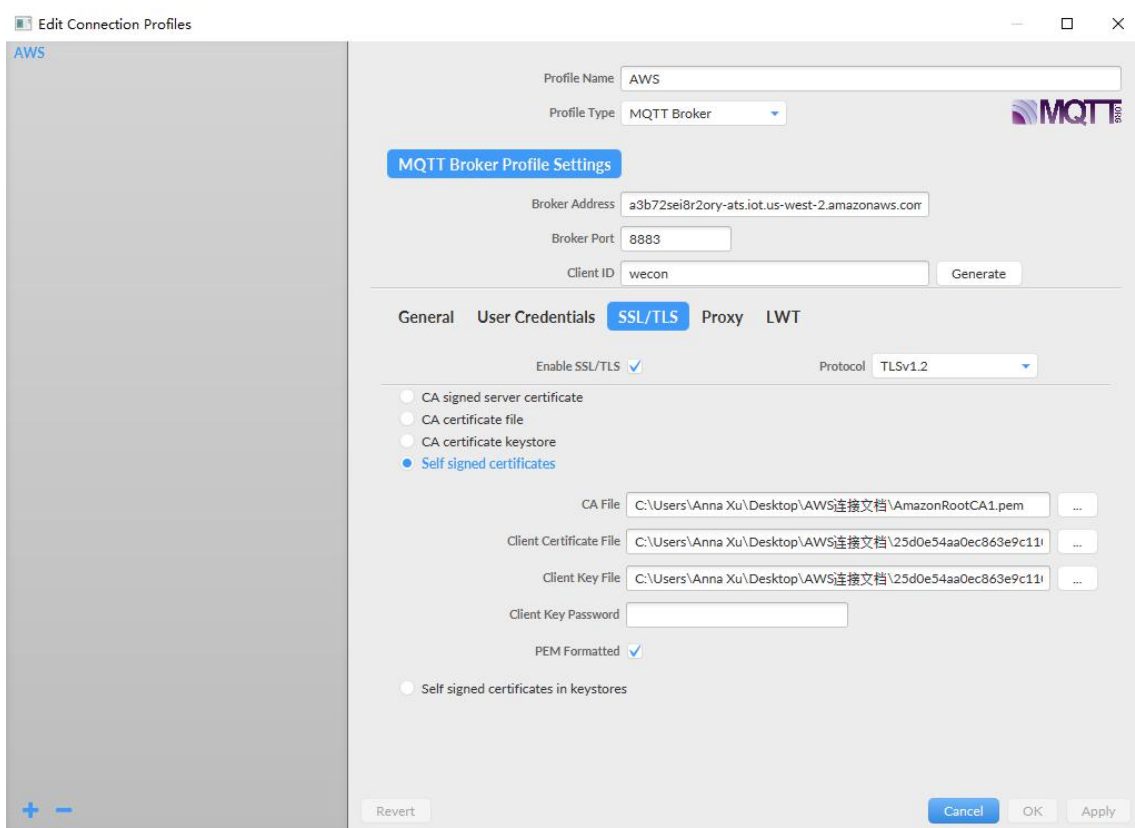


The screenshot shows the AWS IoT Settings page. The 'Device data endpoint' section is highlighted. It states: 'Your devices can use your account's device data endpoint to connect to AWS.' Below this, it explains: 'Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.' The 'Endpoint' section shows the URL 'a3b72sei8r2ory-ats.iot.us-west-2.amazonaws.com' highlighted with a red box.

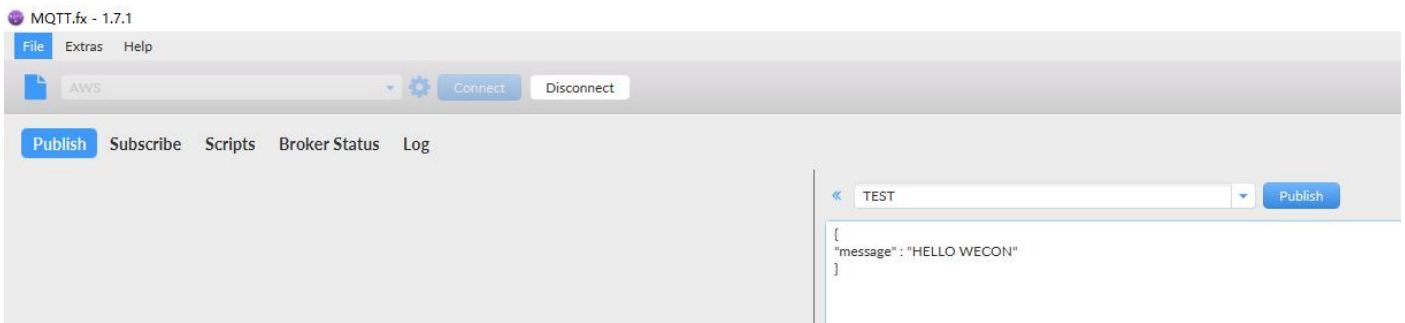
Create one connection in MQTT.fx tool, set broker port as 8883.



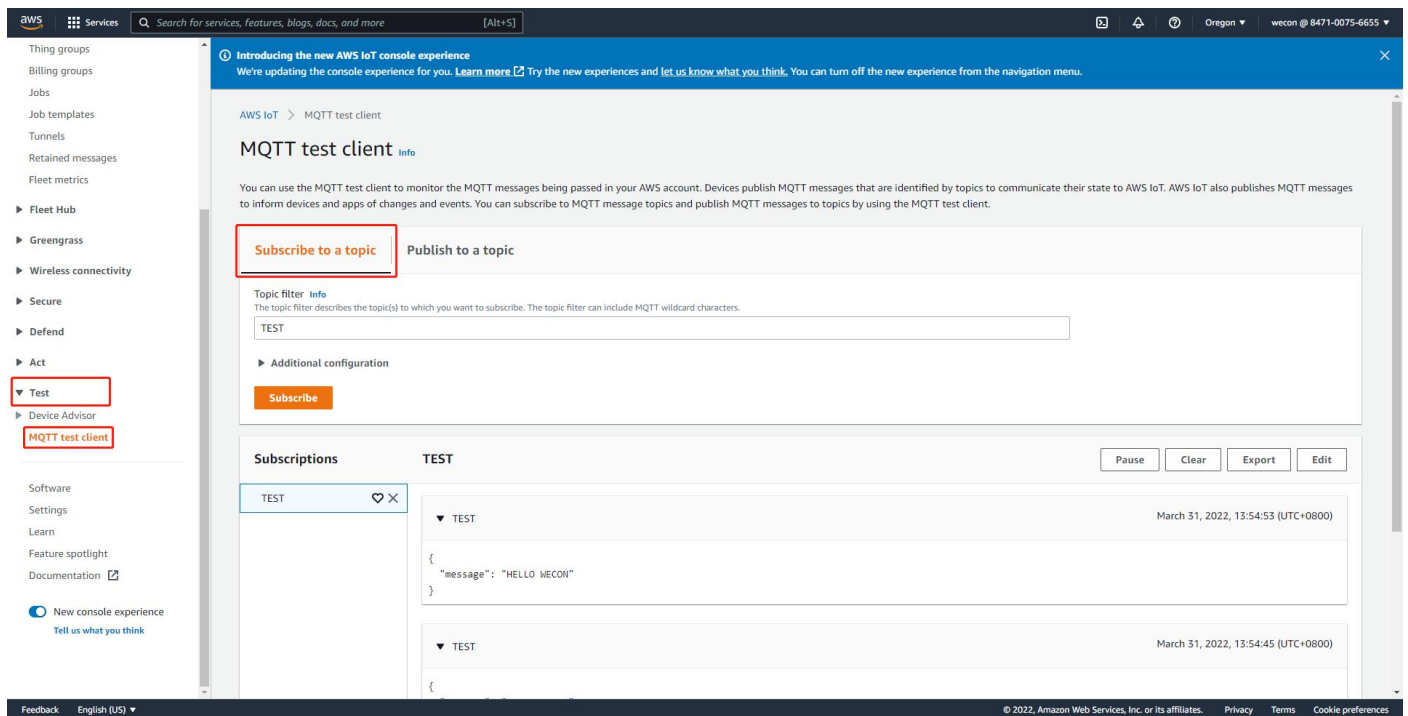
Upload the CA File, Client Certificate File, Client Key File



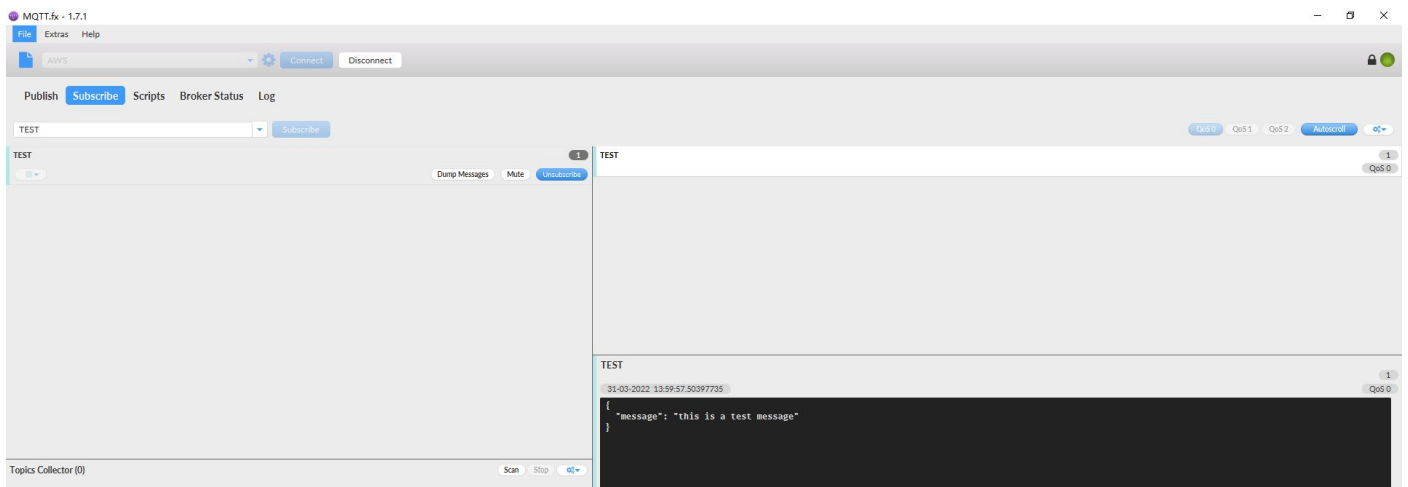
Publish message to topic "TEST"



Click "Test"---->"MQTT test client"---->"Subscribe to a topic", to get message publish from MQTT.fx tool.

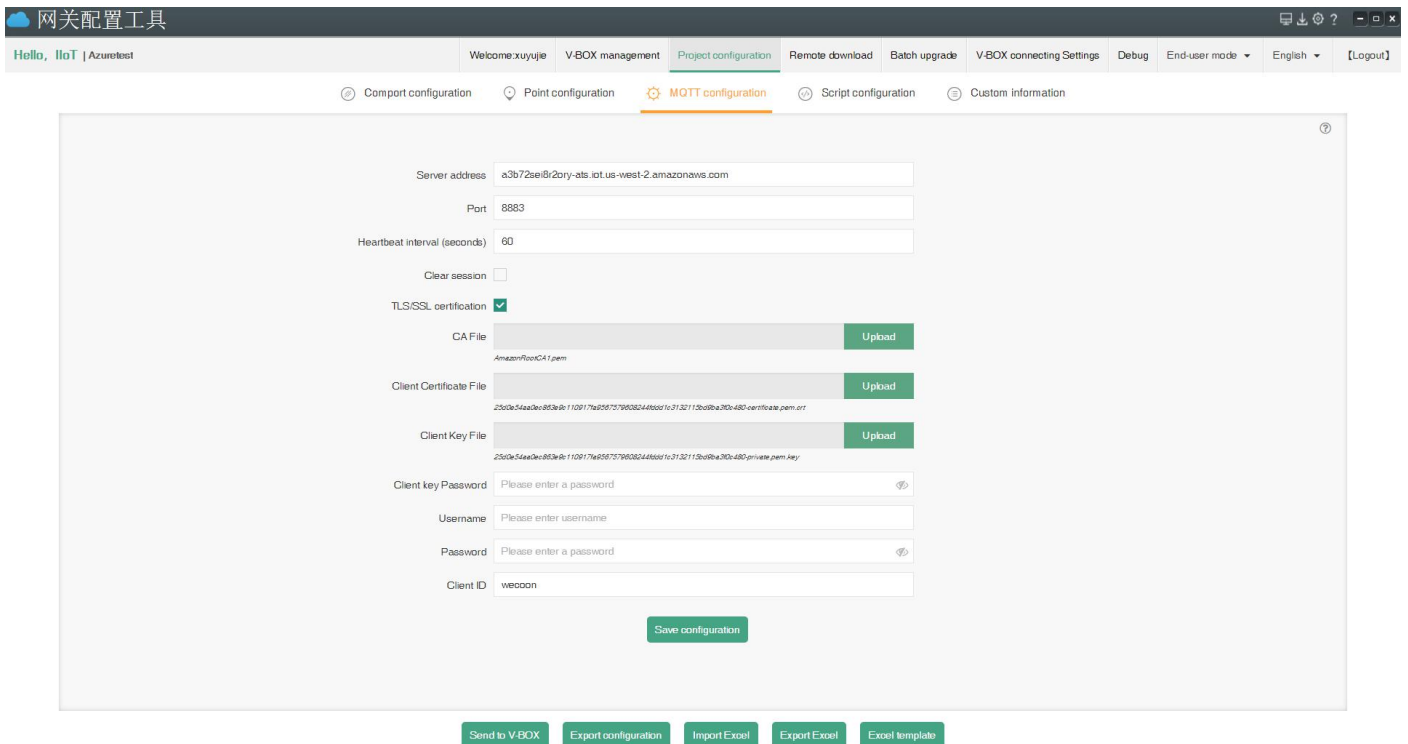


And we can also send message form AWS platform to MQTT.fx tool.



5. Configure in CloudTool

Copy the same setting in MQTT.fx to MQTT configuration



网关配置工具

Hello, IloT | Azuretest

Welcome: xuyujie V-BOX management Project configuration Remote download Batch upgrade V-BOX connecting Settings Debug End-user mode English [Logout]

Comport configuration Point configuration **MQTT configuration** Script configuration Custom information

Server address: a3b72seir2ory-ats.iot.us-west-2.amazonaws.com

Port: 8883

Heartbeat interval (seconds): 60

Clear session: ☐

TLS/SSL certification: ☒

CA File: Upload

Client Certificate File: Upload

Client Key File: Upload

Client key Password:

Username:

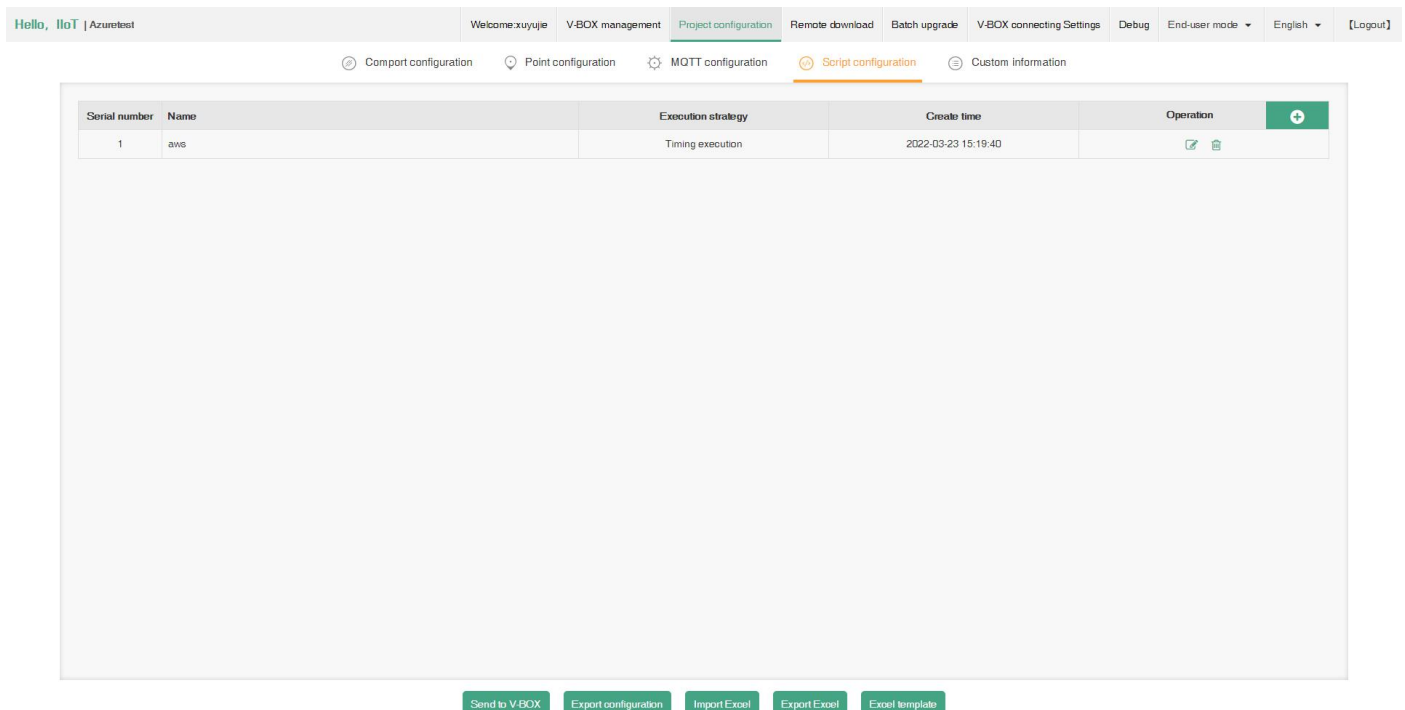
Password:

Client ID: wecon

Save configuration

Send to V-BOX Export configuration Import Excel Export Excel Excel template



Add a lua script and copy the lua demo into it.



Hello, IloT | Azuretest

Welcome: xuyujie V-BOX management Project configuration Remote download Batch upgrade V-BOX connecting Settings Debug End-user mode English [Logout]

Comport configuration Point configuration MQTT configuration **Script configuration** Custom information

Serial number	Name	Execution strategy	Create time	Operation
1	aws	Timing execution	2022-03-23 15:19:40	 

Send to V-BOX Export configuration Import Excel Export Excel Excel template

```
sprint = print
--Cloud mode interface to obtain the MQTT information configured by the cloud platform: (5 returns,
namely the server address, client ID, connection table, last word table, certificate table)
local MQTT_URL, MQTT_CLIENTID, MQTT_CFG, MQTT_LWT, MQTT_CART = mqtt.setup_cfg()
--publish to topics
local pub_RE_TOPIC = string.format('TEST')
--Subscribe topics
local Subscribe_RE_TOPIC1 = string.format('TEST')
--variable
local last_time = 0
--Timing main function
function aws.main()
    sprint(os.date("%Y-%m-%d %H:%M %S", os.time()) .. " aws.main start")
    if g_mq then
        if g_mq:isconnected() then
            send_Data()
        else
            if os.time() - last_time > 5 then
                last_time = os.time()
                mymqtt_connect()
            end
        end
    end
    else
        mymqtt_init()
    end
    sprint(os.date("%Y-%m-%d %H:%M %S", os.time()) .. " aws.main end")
end

-- Initialize MQTT
function mymqtt_init()
    sprint(string.format("mqtt init mqtt_url:%s mqtt_clientid:%s", MQTT_URL, MQTT_CLIENTID))
    g_mq, err = mqtt.create(MQTT_URL, MQTT_CLIENTID) -- Create the object and declare it as a global
variable
    if g_mq then
        g_mq:on("message", mymqtt_msg_callback) -- Register to receive message callbacks
        sprint("mqtt init success")
    else
        sprint("mqtt init failed:", err)
    end
end
```

```

end
-- Connect to MQTT server
function mymqtt_connect()
    sprint("mqtt connecting...")
    local stat, err = g_mq:connect(MQTT_CFG,MQTT_LWT, MQTT_CART)
    if stat == nil then
        sprint("mqtt connect failed:", err)
        return
    else
        sprint("mqtt connected")
    end
    g_mq:subscribe(TEST, 0)
end
-- Receive MQTT message callback function
function mymqtt_msg_callback(topic, msg)
    print("topic:",topic)
    print("revdata:",msg)
    local revData = json.decode(msg)
    print (revData)
    if topic == Subscribe_RE_TOPIC1 then --Process topic information subscribed from the cloud
    if string.match(topic,Subscribe_RE_TOPIC1) then
        --if revData ~= nil then
            for k,v in pairs (revData) do
                print("printing revdata after kv here")
                print (k,v)
            end
            print ("current state is",fanstate)
        --end
    end
end
end
end

--Get real-time data
function getData()
    local jdata = {}
    local addr = bns_get_alldata()
    print(json.encode(addr))
    for i,v in pairs(addr) do
        if v[2] == 1 then

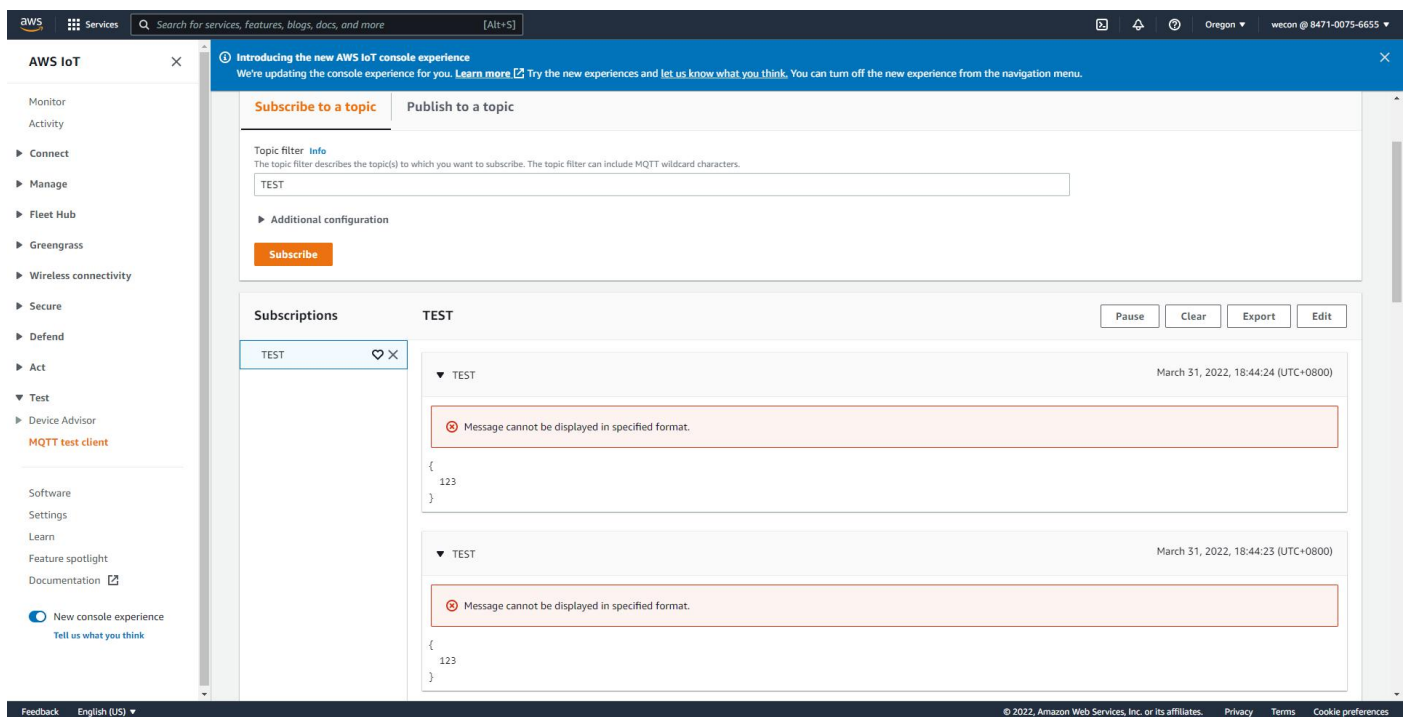
```

```

        jdata[v[3]] = v[4]
    end
end
return jdata
end
--send data
function send_Data()
    local pub_data =
    {
123
    }
    sprint(json.encode(pub_data))
    print(".....",pub_RE_TOPIC)
    return g_mq:publish(pub_RE_TOPIC, json.encode(pub_data), 0, 0)
end

```

Get message in AWS



The screenshot shows the AWS IoT console interface. At the top, there's a navigation bar with the AWS logo, 'Services' link, a search bar, and user information. A blue banner at the top of the console area says 'Introducing the new AWS IoT console experience'. Below this, there are tabs for 'Subscribe to a topic' and 'Publish to a topic'. The 'Subscribe to a topic' tab is active, showing a 'Topic filter' input field with 'TEST' entered. Below the input field, there's an 'Additional configuration' section with a 'Subscribe' button. The 'Subscriptions' section shows a list of subscriptions, with 'TEST' selected. To the right of the subscription list, there are buttons for 'Pause', 'Clear', 'Export', and 'Edit'. The main content area displays two messages for the 'TEST' topic. Each message has a timestamp (March 31, 2022, 18:44:24 (UTC+0800)) and a red error box stating 'Message cannot be displayed in specified format.' Below the error box, the payload is shown as '{ 123 }'.