ADC 405

Internet of Things

Module – 4

Edge to Cloud Protocol

BY

Mrs. Kanchan Chavan

IoT Application Layer Protocols

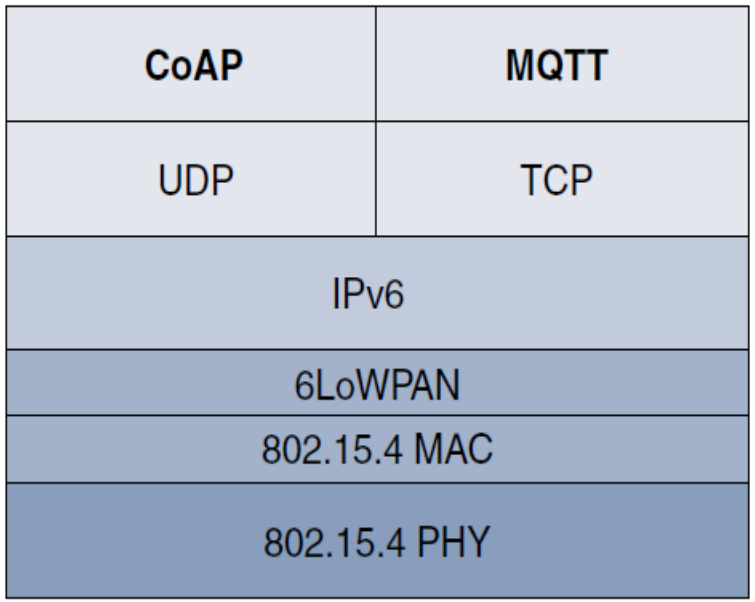


Figure 6-6 *Example of a High-Level IoT Protocol Stack for CoAP and MQTT*

WORLD WIDE WEB – www - Internet

- Web began in 1989 at CERN - European Center for Nuclear Research
- proposal for a web of linked documents - Tim Berners-Lee
- 1994, CERN and M.I.T. signed an agreement - **W3C** (**World Wide Web Consortium**)
- **Web pages -** vast, worldwide collection of content
- **Hypertext -** one page point to another
- **Browser –** program - Pages are viewed - Firefox, Internet Explorer, and Chrome
- **static page, dynamic page**
- **URL** (**Uniform Resource Locator**) **-** identifies pages
- URLs have three parts – protocol, DNS name of the machine, the path uniquely indicating the specific page
  http://www.cs.washington.edu/index.html

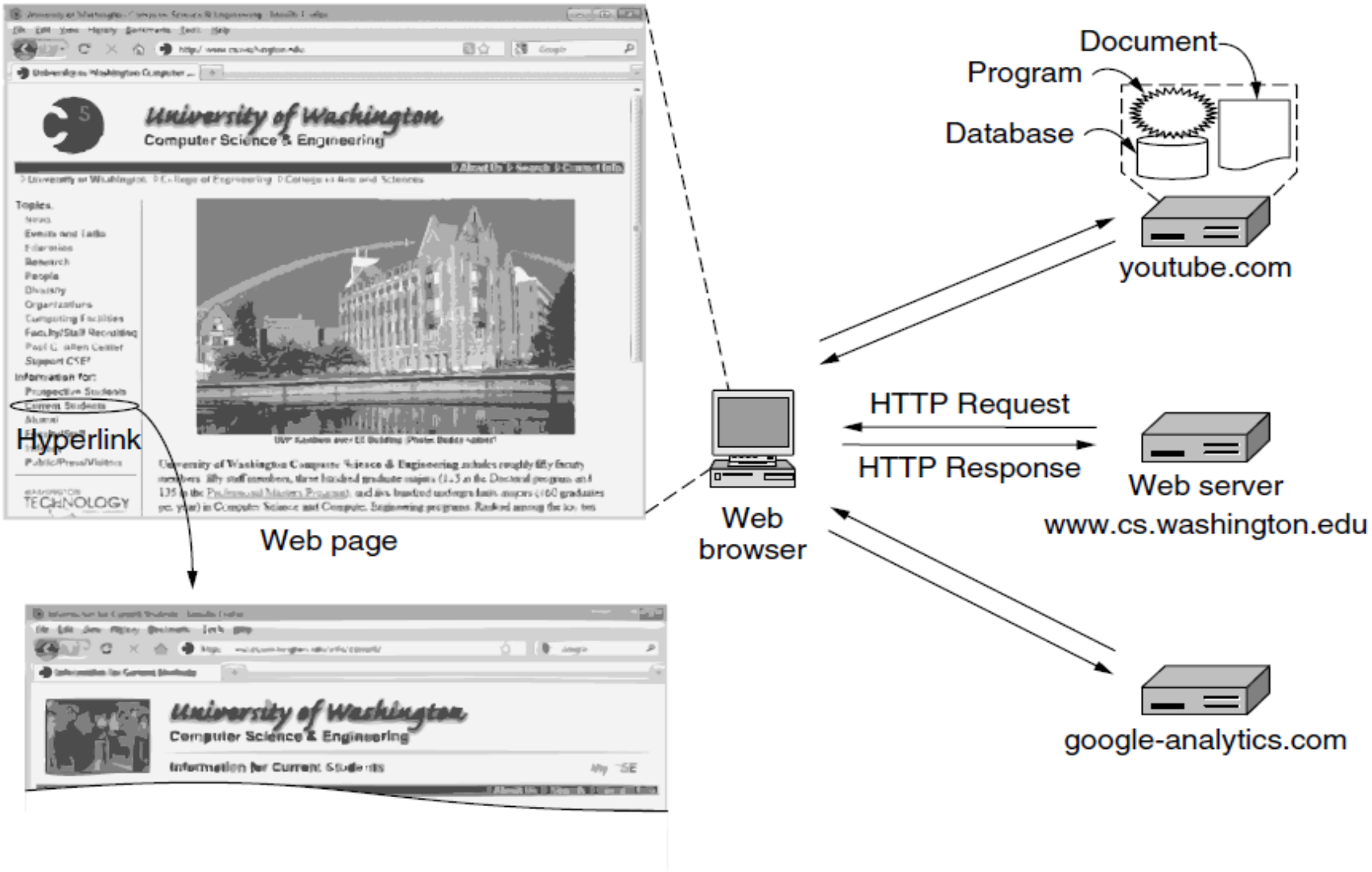


Figure 7-18. Architecture of the Web.

http://www.cs.washington.edu/index.html

HTTP—The HyperText Transfer Protocol

- RFC 2616
- request-response protocol
- ASCII protocol
- runs over TCP (to handle long messages, reliability, or congestion control)
- TCP connection to port 80 on the server's machine
- HTTP 1.0 – connection – single request response
- HTTP 1.1 - **persistent connections - connection reuse ,** relative overhead reduced, possible to pipeline requests

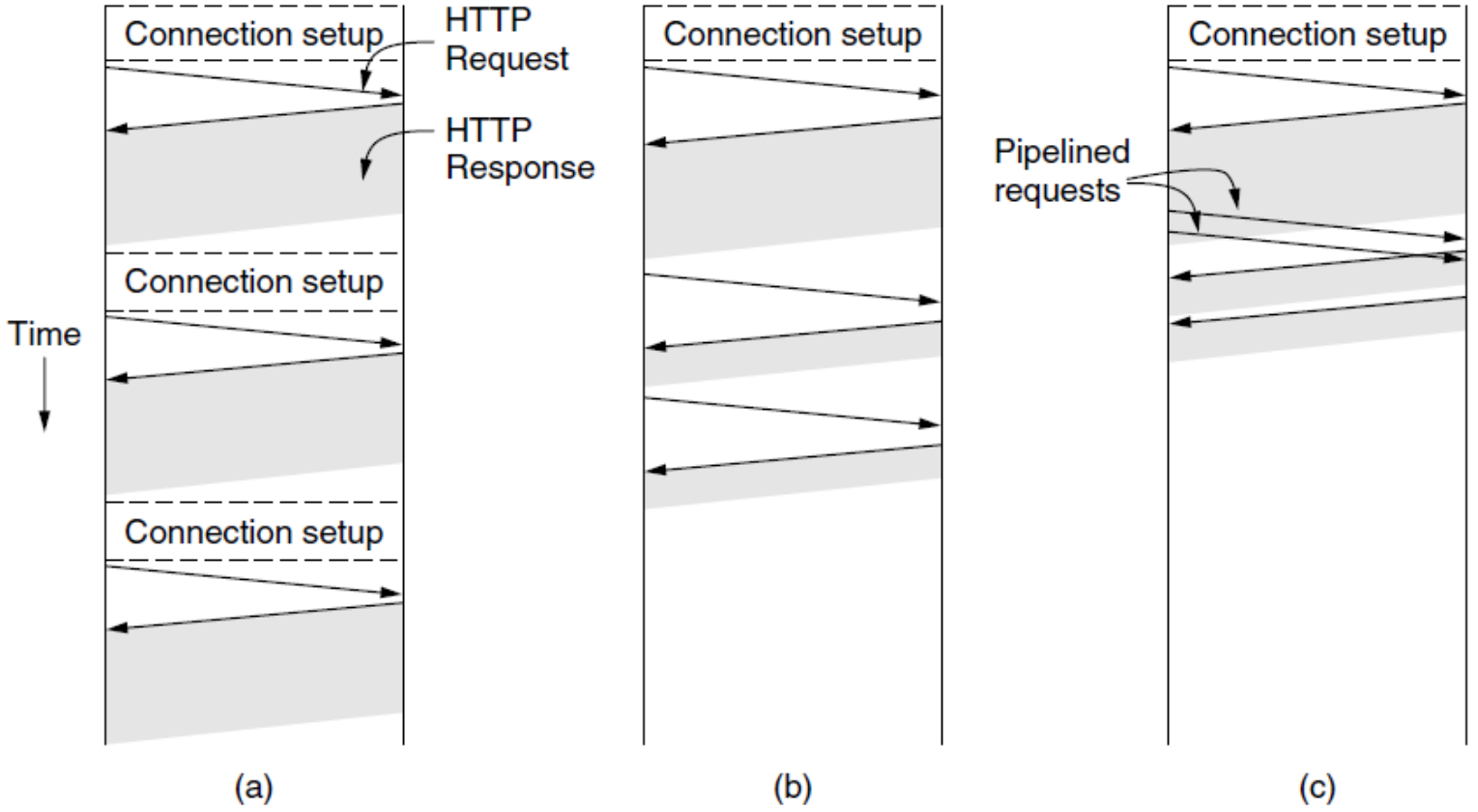


Figure 7-36. HTTP with (a) multiple connections and sequential requests. (b) A persistent connection and sequential requests. (c) A persistent connection and pipelined requests.

- persistent connections open until - idle for a short time (e.g., 60 seconds)
- **parallel connection**

HTTP - Methods

- names are case sensitive
- **Caching**

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

Figure 7-37. The built-in HTTP request methods.

GET filename HTTP/1.1

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Figure 7-38. The status code response groups.

Request Headers – Response Headers

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Content-Encoding	Response	How the content is encoded (e.g., *gzip*)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Content-Range	Response	Identifies a portion of the page's content

HTTP - Caching

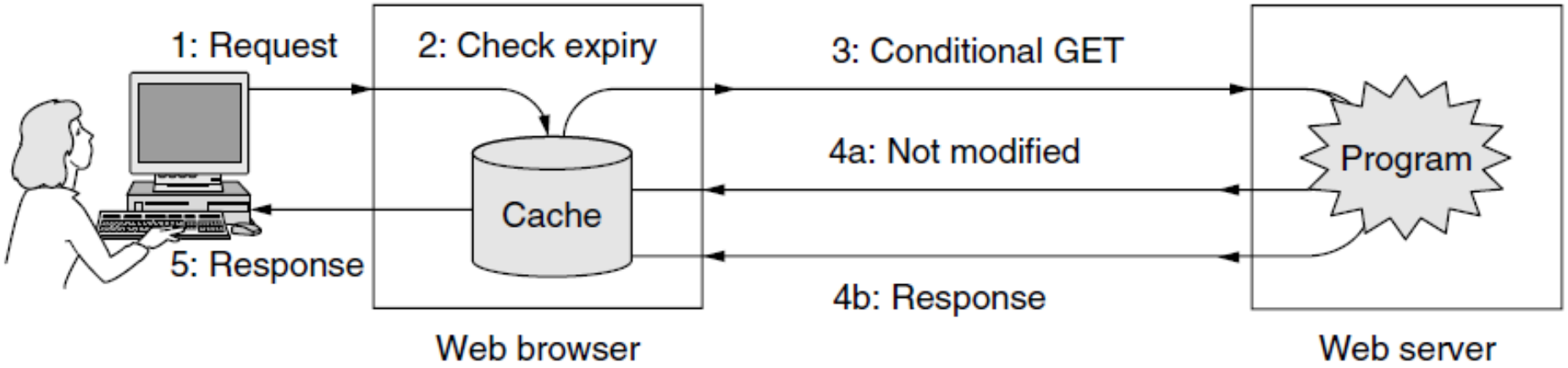


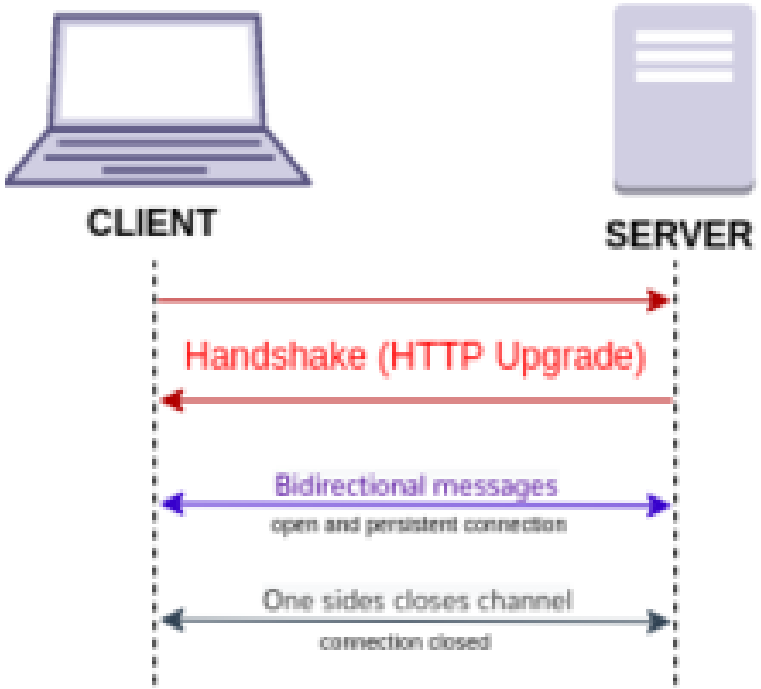
Figure 7-40. HTTP caching.

- *Expires* header
- *Last-Modified* header
- **conditional GET**

```
telnet  www.ietf.org  80
GET  /rfc.html  HTTP/1.1
Host: www.ietf.org
```

WebSocket

- simultaneous two-way communication channels over a single Transmission Control Protocol (TCP) connection
- IETF as RFC 6455 in 2011
- designed to work over HTTP ports 443 and 80 as well as to support HTTP proxies and intermediaries", thus making it compatible with HTTP
- enables real-time, event-driven communication between a client and a server
- Socket is an endpoint for sending and receiving data across the network
- WebSockets maintain a persistent connection



wss://www.example.com/socketserver – ws
websocket, wss – websocket secure

WebSocket

```
GET /chat HTTP/1.1
Host: normal-website.com
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: wDqumtseNBJdhkihL6PW7w==
Connection: keep-alive, Upgrade
Cookie: session=KOsEJNuflw4Rd9BDNrVmvwBF9rEijeE2
Upgrade: websocket
```

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Accept: 0FFP+2nmNIf/h+4BP36k9uzrYGk=
```

CoAP - Constrained Application Protocol

- IETF Constrained RESTful Environments (CoRE) working group's efforts
- IETF – Internet Engg Task Force
- RFC – Request for Comments - **RFC 7252**
- CoAP can run over IPv4 or IPv6
- port 5683 is used for non-secured CoAP, port 5684 is utilized for DTLS-secured CoAP

CoAP Message Field
Ver (Version)
T (Type)
TKL (Token Length)
Code
Message ID
Token
Options
Payload

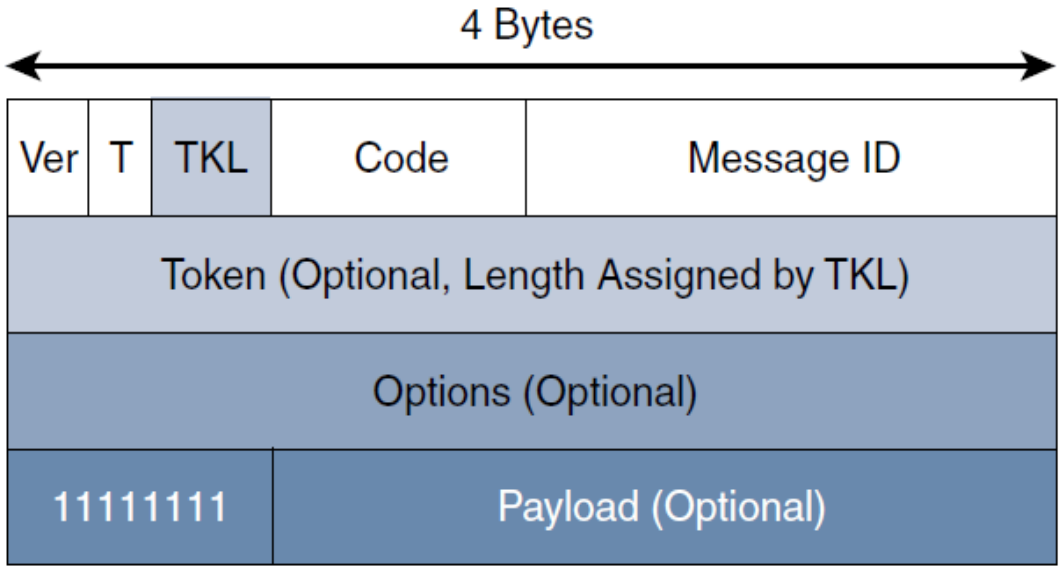


Figure 6-7 *CoAP Message Format*

CoAP - Constrained Application Protocol

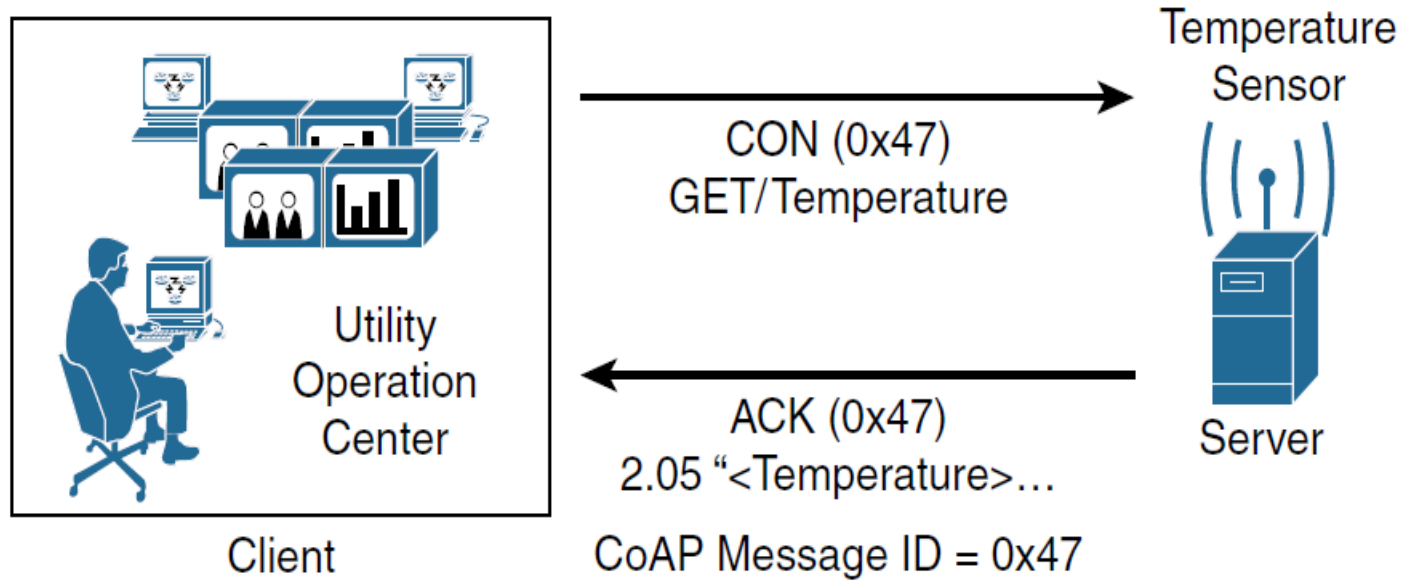


Figure 6-9 *CoAP Reliable Transmission Example*

Message Queuing Telemetry Transport (MQTT)

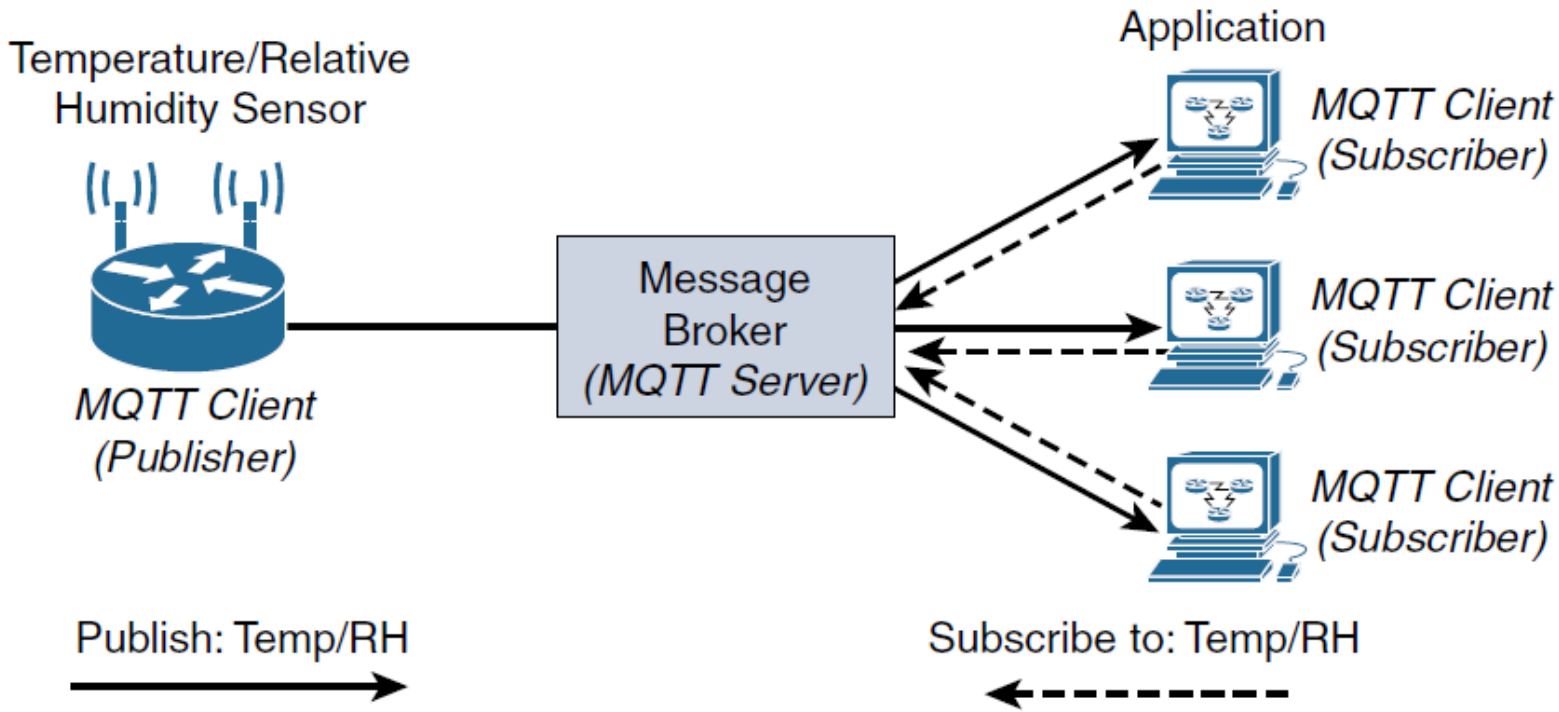


Figure 6-10 *MQTT Publish/Subscribe Framework*

Message Queuing Telemetry Transport (MQTT)

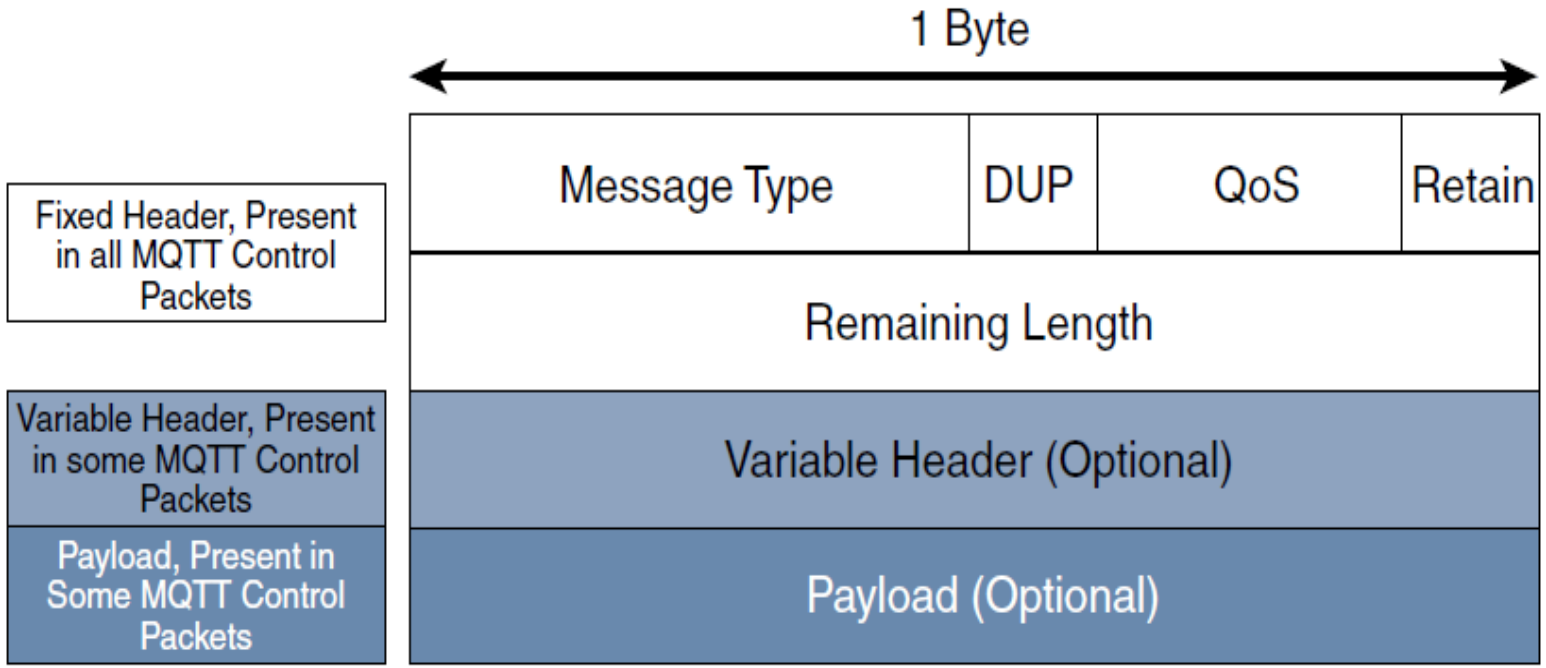


Figure 6-11 *MQTT Message Format*

Message Queuing Telemetry Transport (MQTT)

- lightweight protocol
- run over a TCP transport using port 1883
- MQTT can be secured using TLS on port 8883

Message Queuing Telemetry Transport (MQTT)

Table 6-2 *MQTT Message Types*

Message Type	Value	Flow	Description
CONNECT	1	Client to server	Request to connect
CONNACK	2	Server to client	Connect acknowledgement
PUBLISH	3	Client to server Server to client	Publish message
PUBACK	4	Client to server Server to client	Publish acknowledgement
PUBREC	5	Client to server Server to client	Publish received
PUBREL	6	Client to server Server to client	Publish release
PUBCOMP	7	Client to server Server to client	Publish complete
SUBSCRIBE	8	Client to server	Subscribe request
SUBACK	9	Server to client	Subscribe acknowledgement
UNSUBSCRIBE	10	Client to server	Unsubscribe request
UNSUBACK	11	Server to client	Unsubscribe acknowledgement
PINGREQ	12	Client to server	Ping request
PINGRESP	13	Server to client	Ping response
DISCONNECT	14	Client to server	Client disconnecting

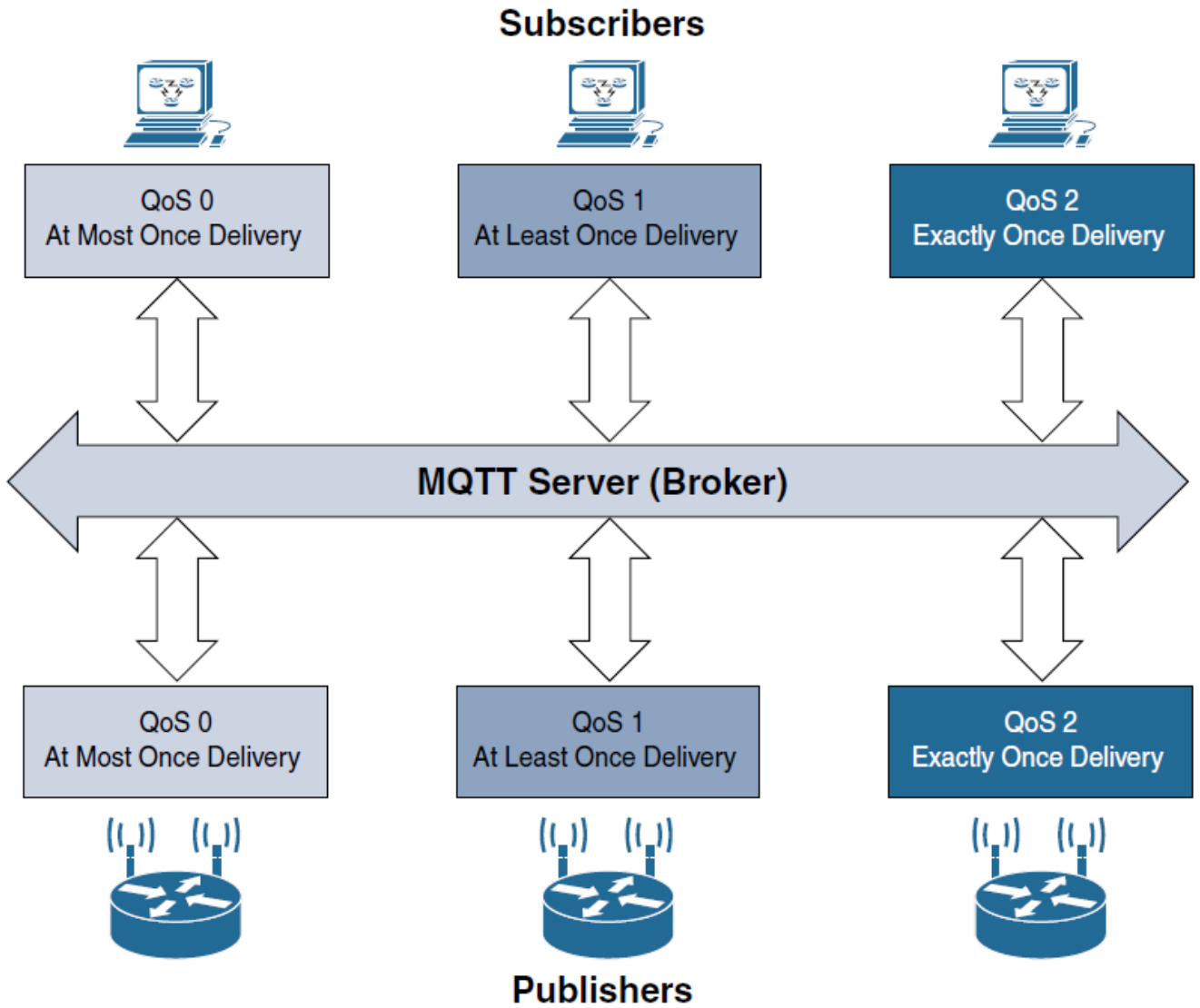


Figure 6-13 *MQTT QoS Flows*

Table 6-3 *Comparison Between CoAP and MQTT*

Factor	CoAP	MQTT
Main transport protocol	UDP	TCP
Typical messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication model	One-to-one	many-to-many
Strengths	Lightweight and fast, with low overhead, and suitable for constrained networks; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages	TCP and multiple QoS options provide robust communications; simple management and scalability using a broker architecture
Weaknesses	Not as reliable as TCP-based MQTT, so the application must ensure reliability.	Higher overhead for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support

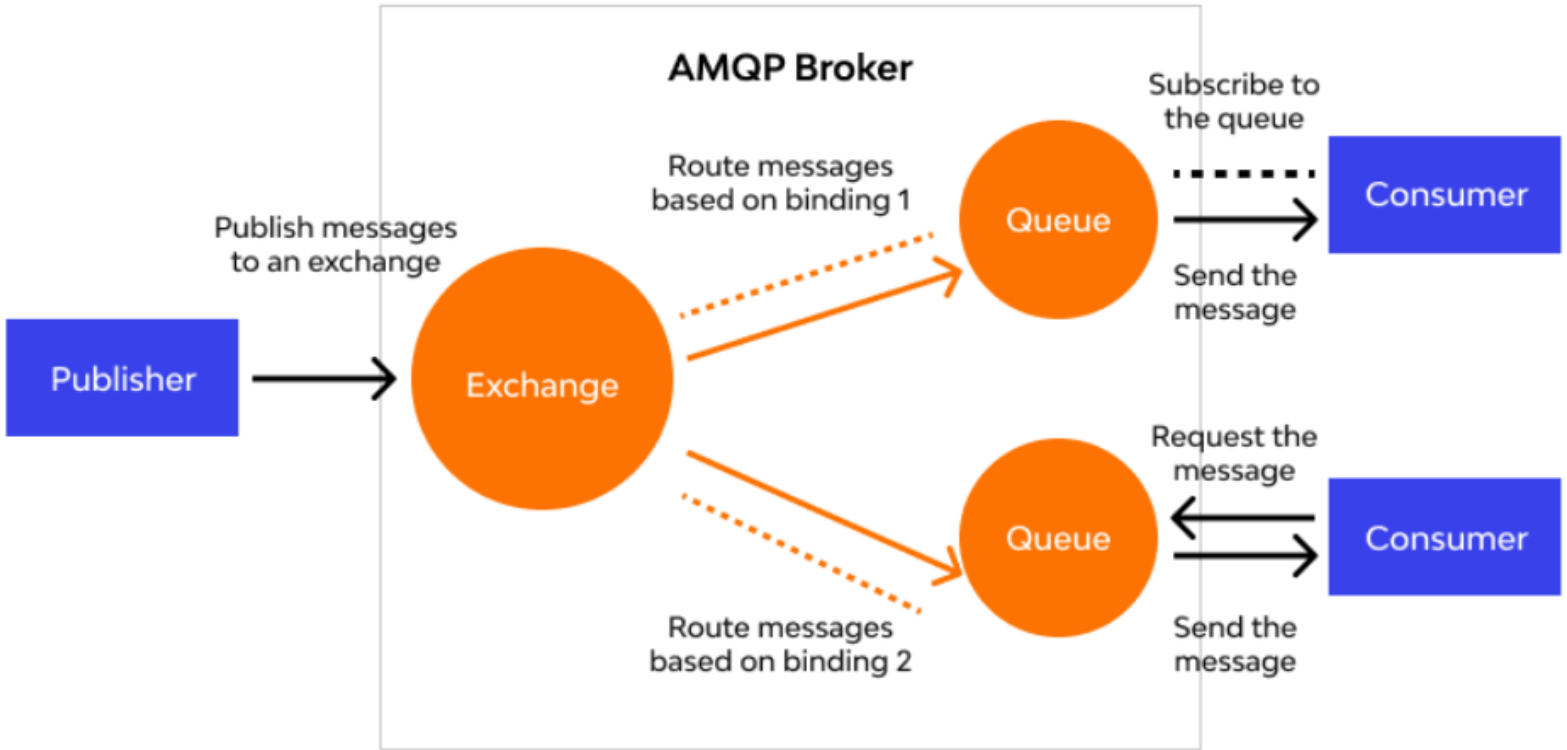
Streaming Text Oriented Message Protocol (STOMP)

- similar to HTTP
- Simple text based protocol
- works over TCP using the commands – **CONNECT, SEND, SUBSCRIBE, UNSUBSCRIBE, BEGIN, COMMIT, ABORT, ACK, NACK, DISCONNECT**
- developed to work with message oriented middleware
- STOMP specifies publish-subscribe mechanism
- interoperable wire format

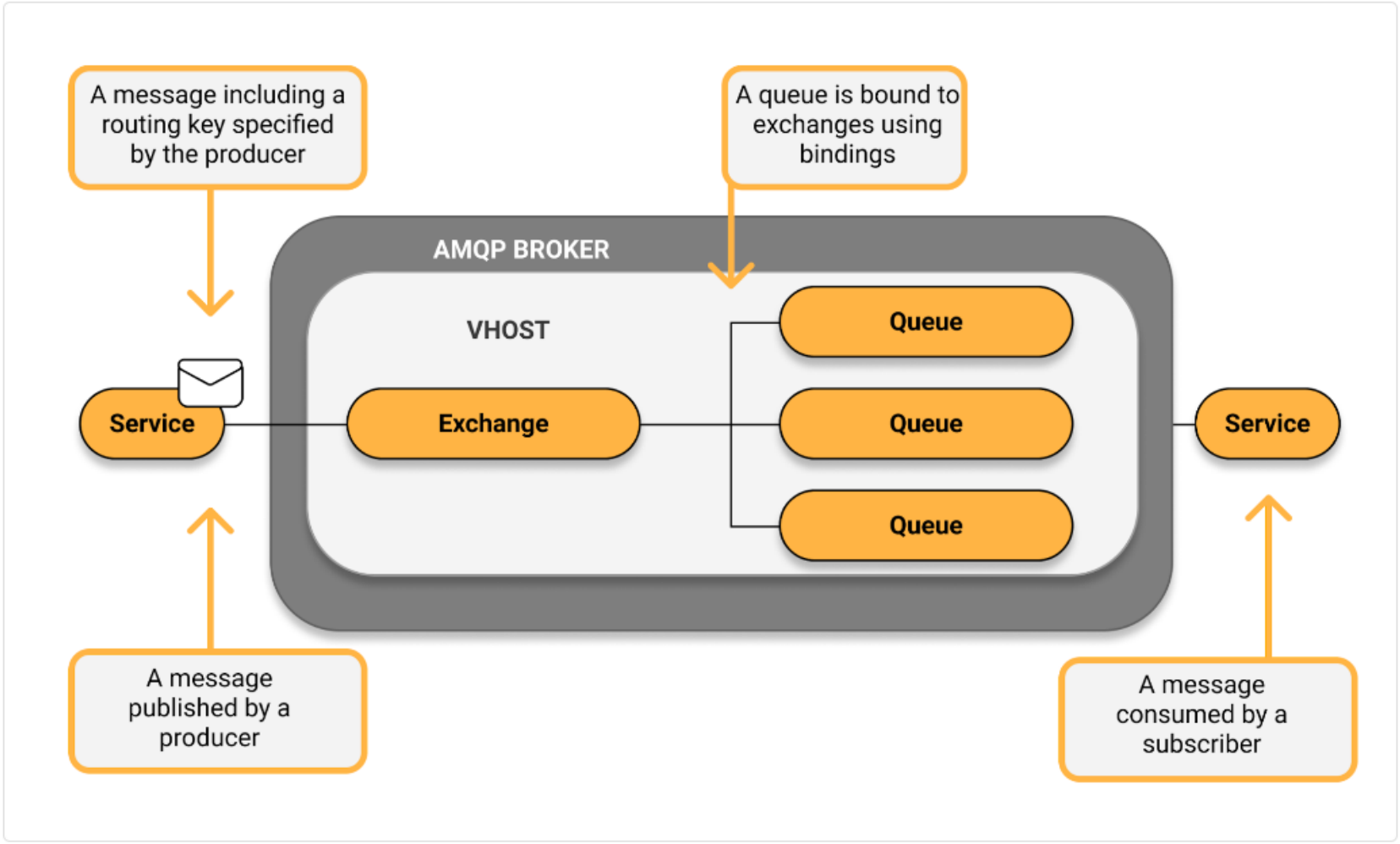
```
SEND
destination:/queue/a
content-type:text/plain

hello queue a
^@
```

Advanced Message Queuing Protocol (AMQP)

- for passing business messages between applications or organizations
- Adding money to your digital wallets, Credit or debit card transaction in retail stores

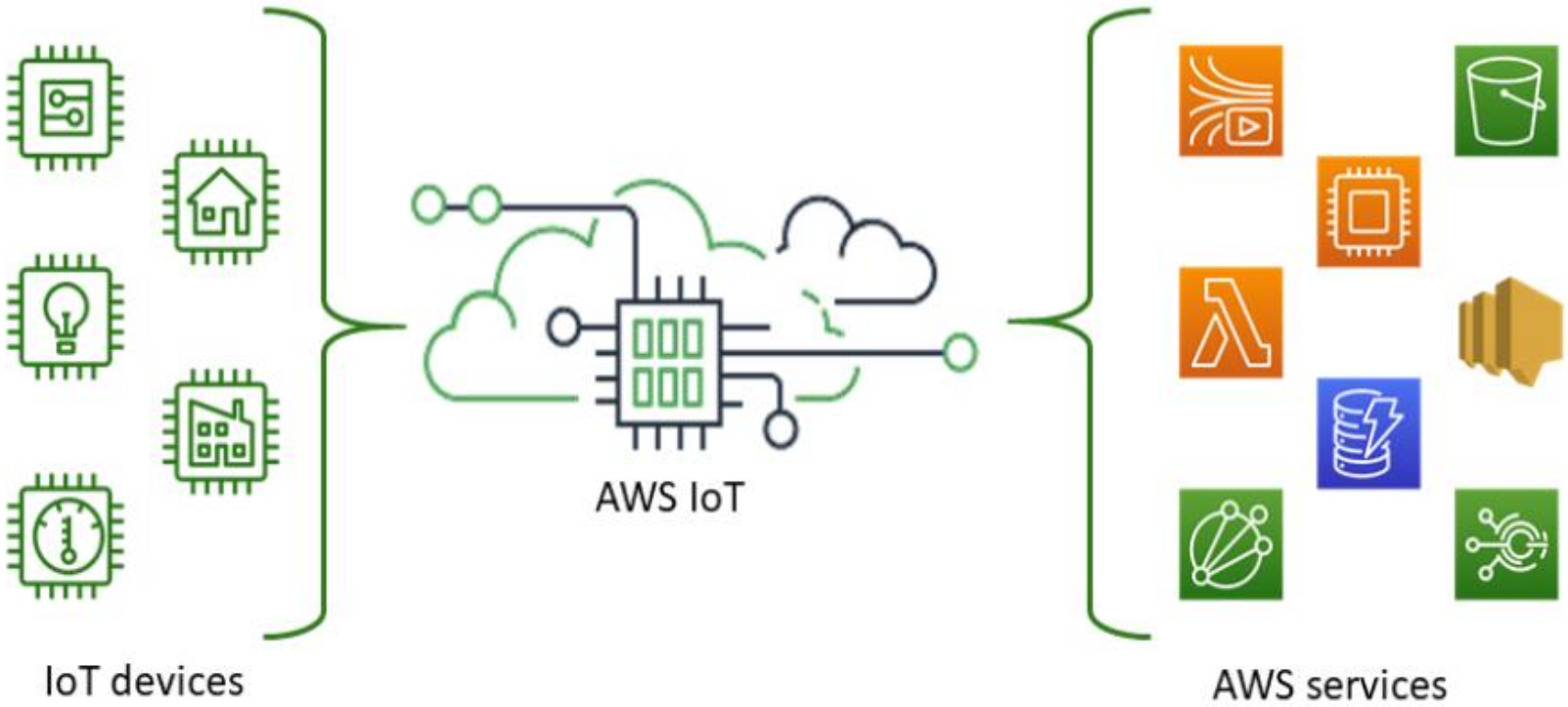


Advanced Message Queuing Protocol (AMQP)

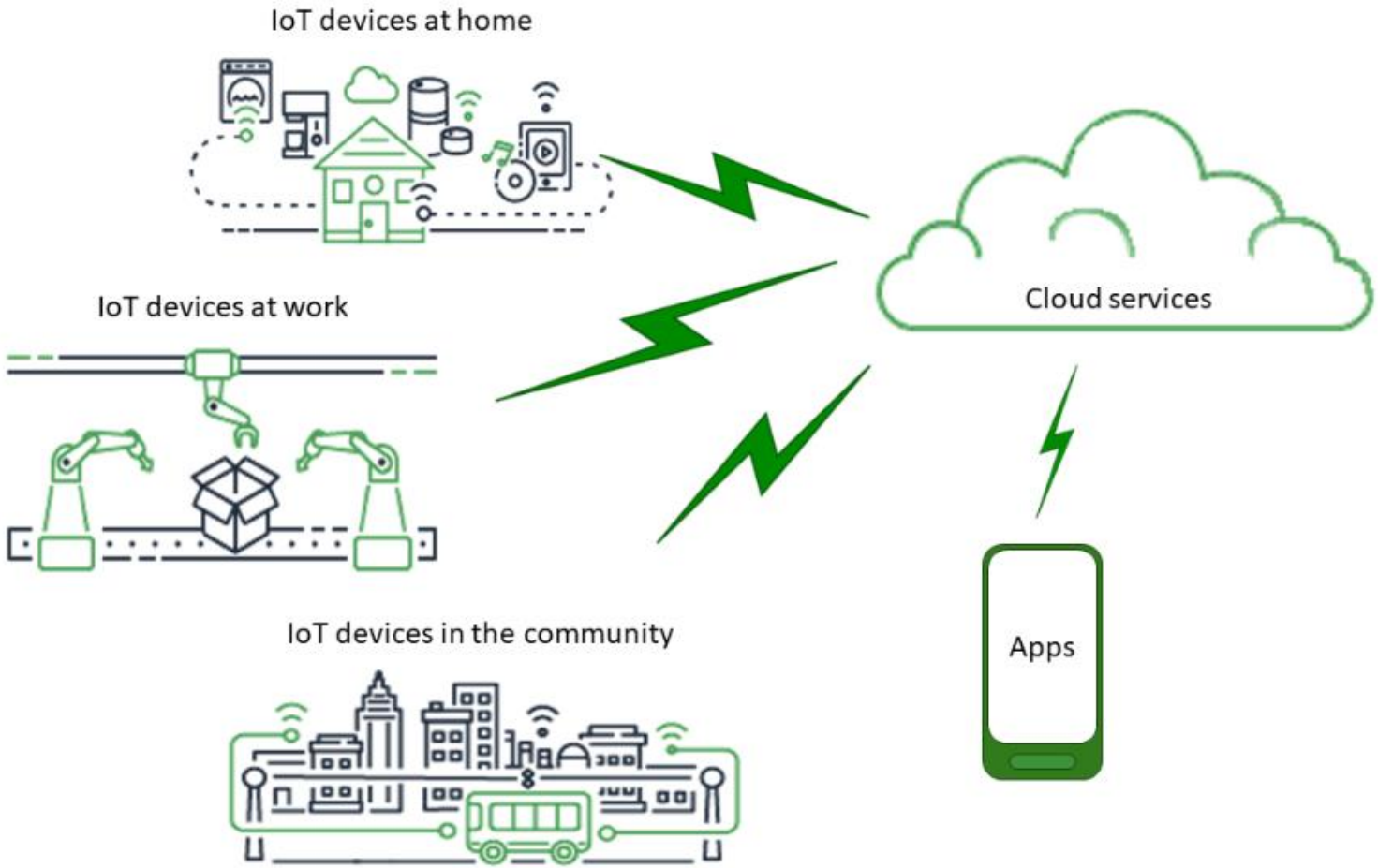


AWS IoT

AWS IoT provides cloud services and device support that you can use to implement IoT solutions



AWS IoT


IoT devices at home
IoT devices at work
IoT devices in the community
Cloud services
Apps

AWS IoT

- **Apps -** Apps give end users access to IoT devices and the features provided by the cloud services to which those devices are connected.

- **Cloud services**
  - IoT connection and management services
  - Compute services, such as Amazon Elastic Compute Cloud and AWS Lambda
  - Database services, such as Amazon DynamoDB

- **Communications**

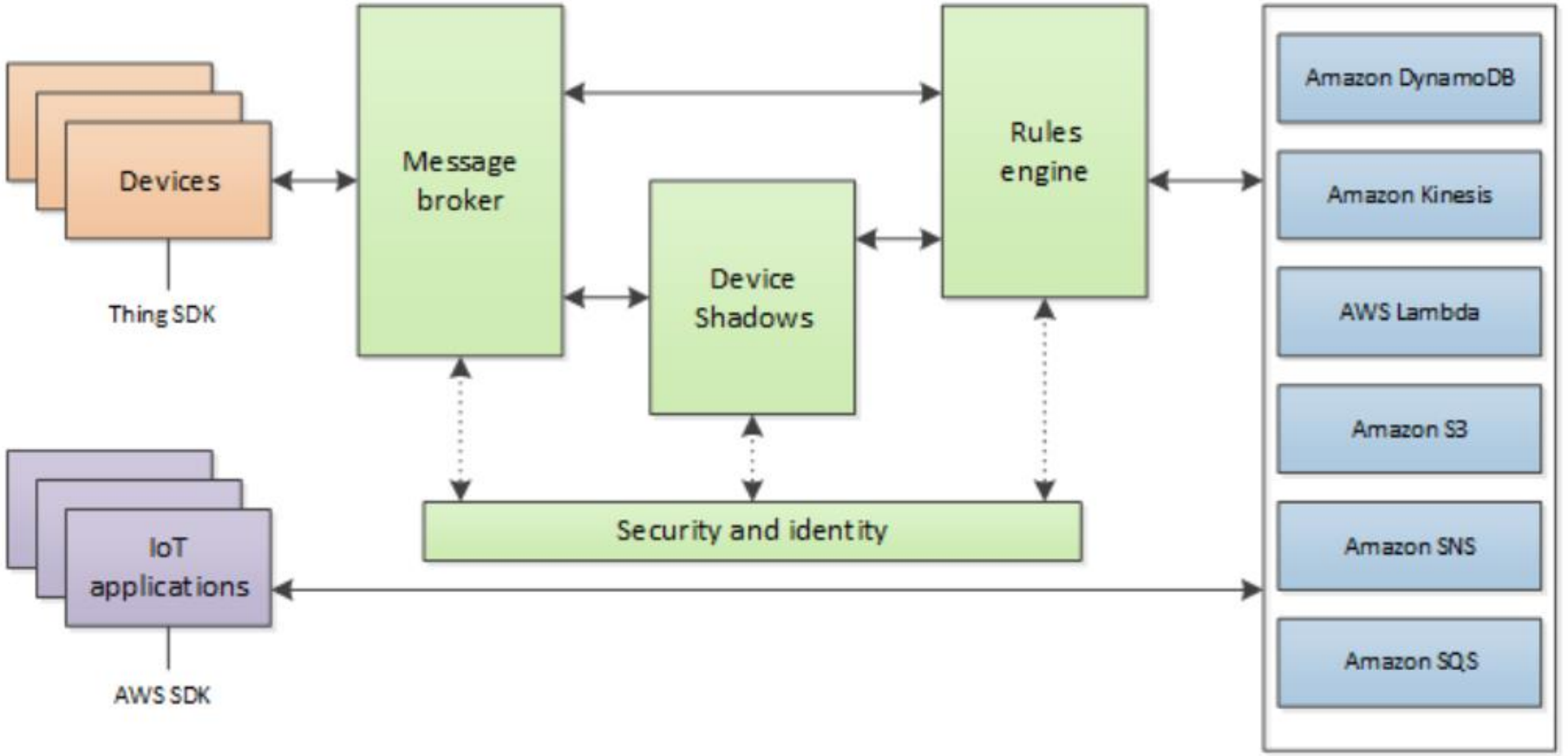
  Devices communicate with cloud services by using various technologies and protocols. Examples include:

  •Wi-Fi/Broadband internet
  •Broadband cellular data
  •Narrow-band cellular data
  •Long-range Wide Area Network (LoRaWAN)
  •Proprietary RF communications

AWS IoT

- **AWS IoT Device SDKs -**Build applications on your devices that send messages to and receive messages from AWS IoT.
- **AWS IoT Core for LoRaWAN -**Connect and manage your long range WAN (LoRaWAN) devices and gateways
- **AWS Command Line Interface (AWS CLI) -**These commands allow you to create and manage thing objects, certificates, rules, jobs, and policies.
- **AWS IoT API -** Build your IoT applications using HTTP or HTTPS requests.
- **AWS SDKs -**Build your IoT applications using language-specific APIs.

AWS IoT



- **Device shadow -** A JSON document used to store and retrieve current state information for a device.
- **Rules engine -** The Rules engine connects data from the message broker to other AWS IoT services for storage and additional processing.

Communication protocols supported by AWS IoT Core

- MQTT (Message Queuing and Telemetry Transport)
- MQTT over WSS (Websockets Secure)
- HTTPS (Hypertext Transfer Protocol - Secure)

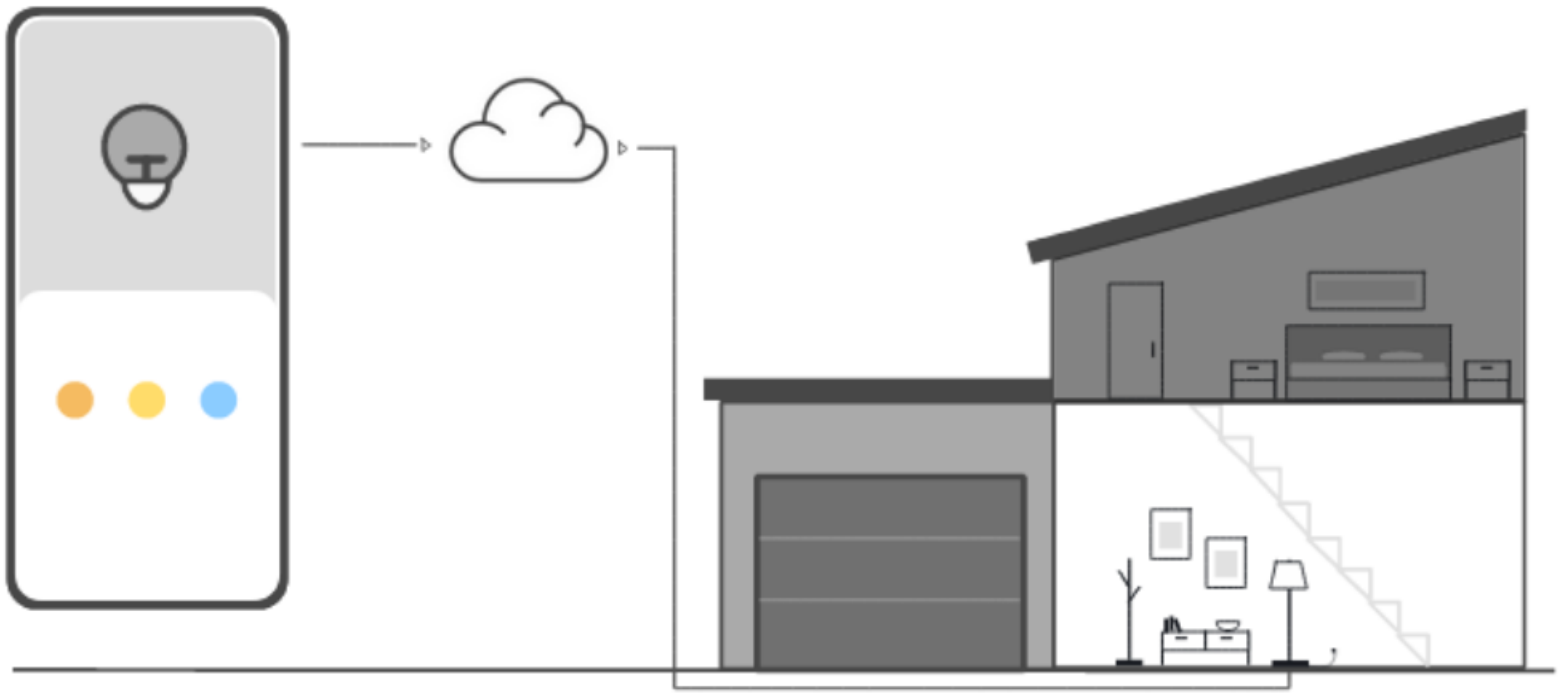
Core

PDF

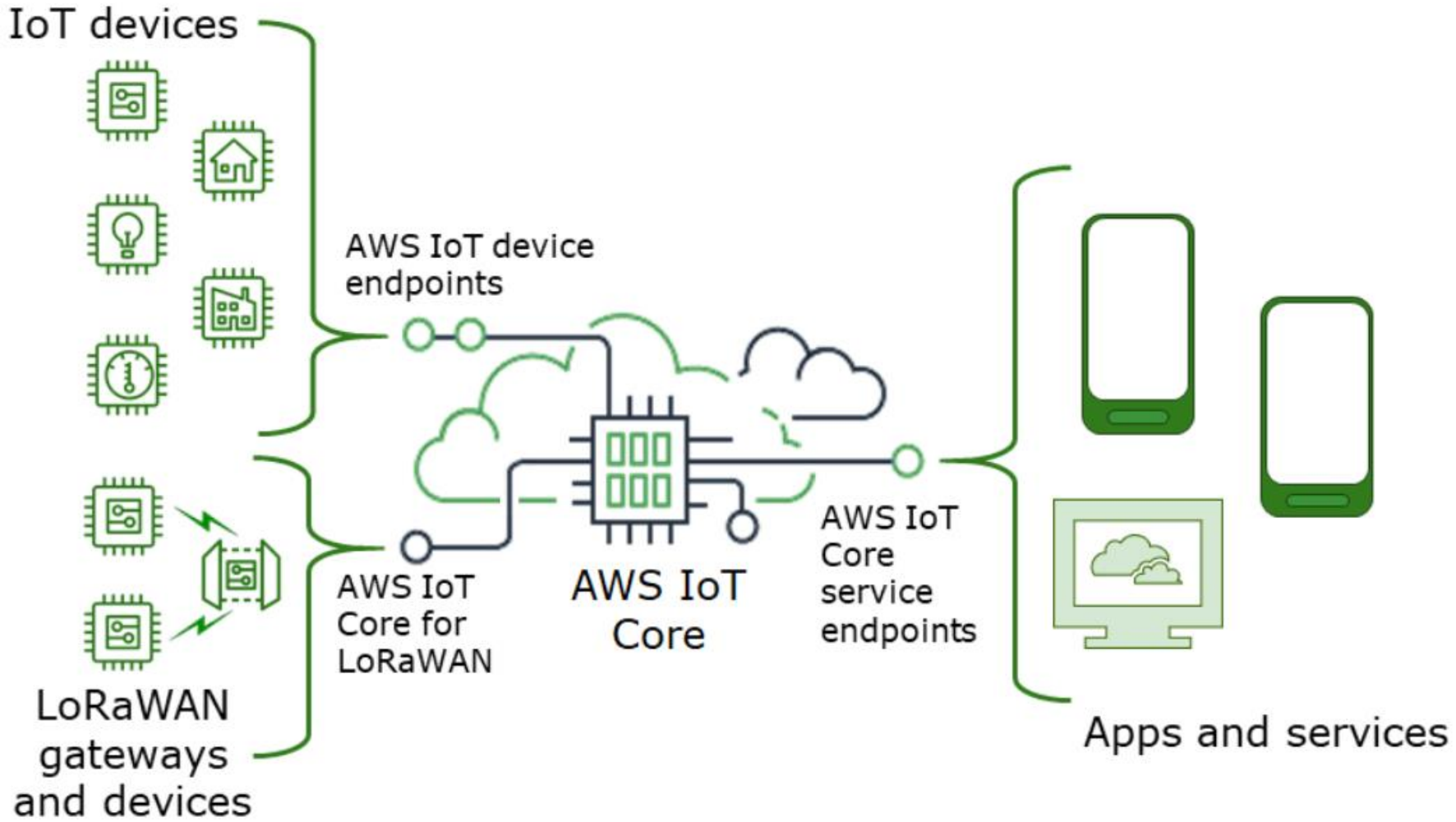
Whether you're new to IoT or you have years of experience, these resources present the AWS IoT concepts and terms that will help you start using AWS IoT.

- **Look** inside AWS IoT and its components in How AWS IoT works.
- Learn more about AWS IoT from our collection of training materials and videos. This topic also includes a list of services that AWS IoT can connect to, social media links, and links to communication protocol specifications.
- **Connect your first device to AWS IoT Core**.
- **Develop** your IoT solutions by Connecting to AWS IoT Core and exploring the AWS IoT tutorials.
- **Test and validate** your IoT devices for secure and reliable communication by using the Device Advisor.

Connecting IoT devices



Connecting IoT devices



https://www.youtube.com/watch?v=TrEtf0R5nzE

AWS IoT

- Monitor
- Onboard
- Manage
  - Things
  - Types
  - Thing Groups
  - Billing Groups
  - Jobs
- Greengrass
- Secure
- Defend
- Act
- Test

You don't have any things yet

A thing is the representation of a device in the cloud.

Learn more | Register a thing

Creating AWS IoT things

An IoT thing is a representation and record of your phyisical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. Learn more.

Register a single AWS IoT thing

Create a thing in your registry

Create a single thing

Bulk register many AWS IoT things

Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

Cancel

Create a single thing

CREATE A THING

Add your device to the thing registry

STEP 1/3

This step creates an entry in the thing registry and a thing shadow for your device.

Name

MyDemoDevice

Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected

Create a type

Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group

CREATE A THING

Add a certificate for your thing

STEP 2/3

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Create certificate

Create with CSR

Upload your own certificate signing request (CSR) based on a private key you own.

Create with CSR

Use my certificate

Register your CA certificate and use your own certificates for one or many devices.

Get started

Skip certificate and create thing

You will need to add a certificate to your thing later before your device can connect to AWS

Create thing without certificate

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	fa0877650f.cert.pem	Download
A public key	fa0877650f.public.key	Download
A private key	fa0877650f.private.key	Download

You also need to download a root CA for AWS IoT:

A root CA for AWS IoT Download

Activate

Server Authentication

Server certificates allow your devices to verify that they're communicating with AWS IoT and not another server impersonating AWS IoT. Service certificates must be copied onto your device and referenced when devices connect to AWS IoT. For more information, see the AWS IoT Device SDKs.

AWS IoT server certificates are signed by one of the following CA certificates:

VeriSign Endpoints (legacy)

- RSA 2048 bit key: VeriSign Class 3 Public Primary G5 root CA certificate

Amazon Trust Services Endpoints (preferred)

- RSA 2048 bit key: Amazon Root CA 1.
- RSA 4096 bit key: Amazon Root CA 2 - Reserved for future use.
- ECC 256 bit key: Amazon Root CA 3.
- ECC 384 bit key: Amazon Root CA 4 - Reserved for future use.

We recommend that all customers create an Amazon Trust Services (ATS) endpoint and load these CA certificates onto their devices to avoid any issues with the widespread distrust by browsers of Symantec CAs (including VeriSign) that occurred in October 2018. For backward compatibility, we still support customers who use these endpoints. Customers can retreive their ATS endpoint by calling the `describe-endpoint` API with the `iot:Data-ATS` endpoint type. Devices operating on ATS endpoints are fully interoperable with devices operating on Symantec endpoints in the same account. They do not need to be reregistered.

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	fa0877650f.cert.pem	**Download**
A public key	fa0877650f.public.key	**Download**
A private key	fa0877650f.private.key	**Download**

You also need to download a root CA for AWS IoT:

A root CA for AWS IoT **Download**

Activate

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	fa0877650f.cert.pem	Download
A public key	fa0877650f.public.key	Download
A private key	fa0877650f.private.key	Download

You also need to download a root CA for AWS IoT:

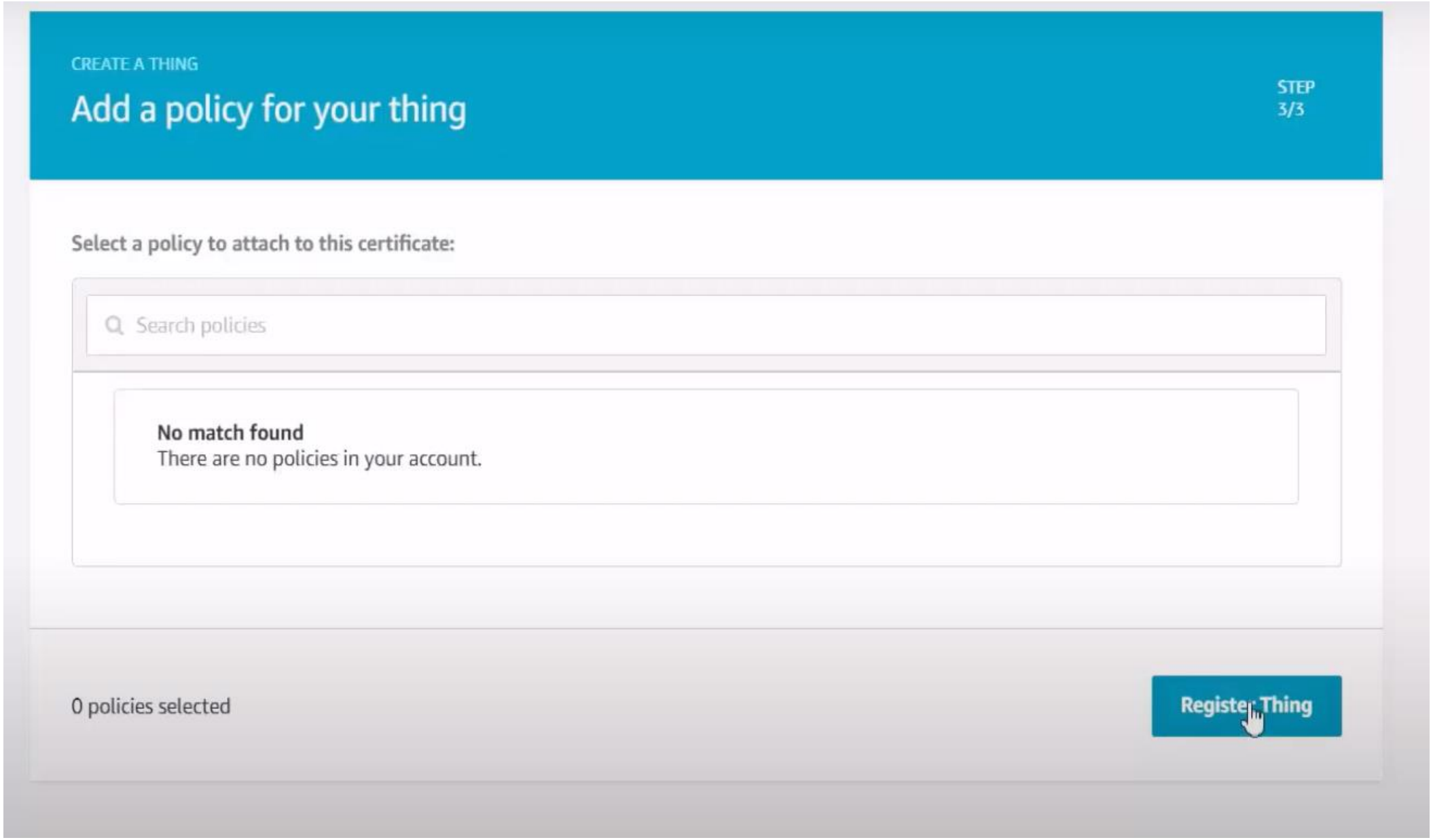
A root CA for AWS IoT Download

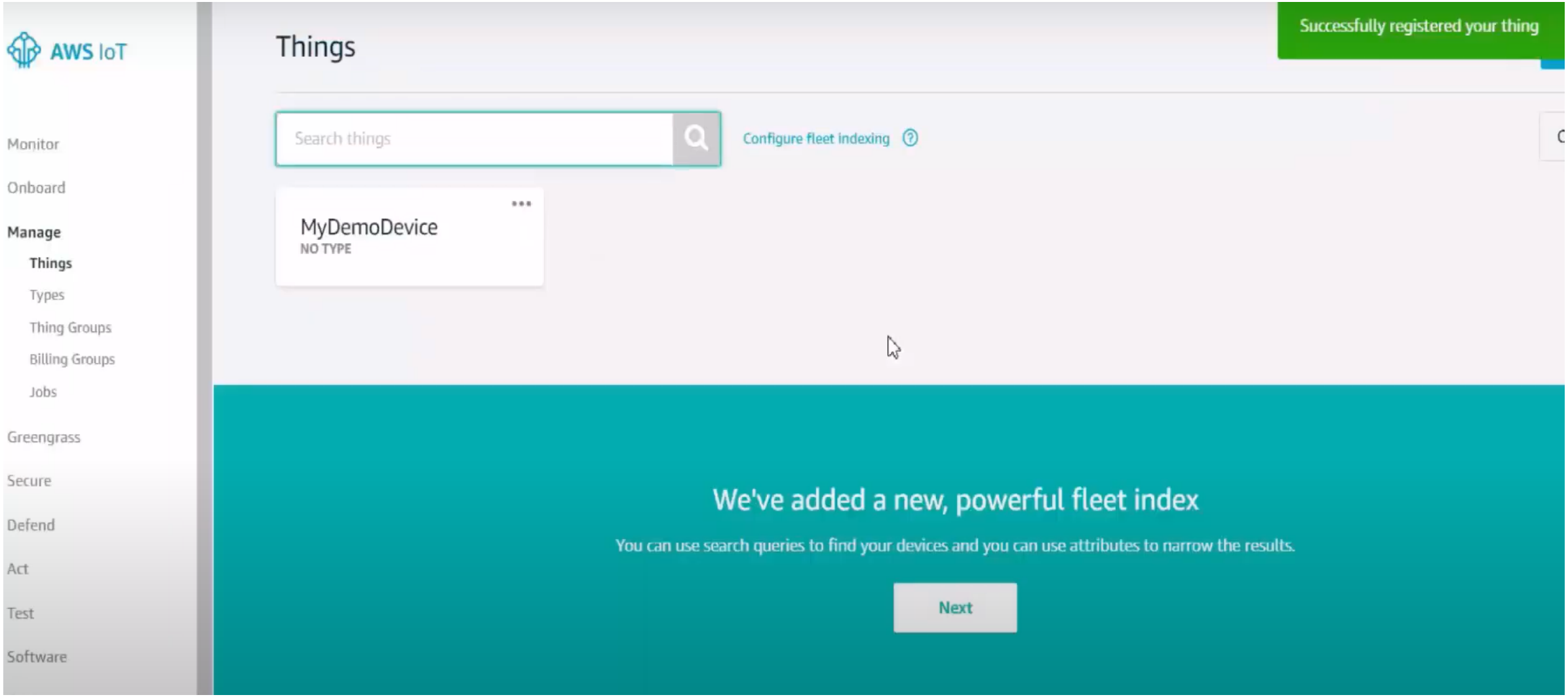
Deactivate

Cancel

Done

Attach a policy


CREATE A THING
Add a policy for your thing
STEP
3/3
Select a policy to attach to this certificate:
Search policies
No match found
There are no policies in your account.
0 policies selected
Register Thing


AWS IoT
Monitor
Onboard
Manage
Things
Types
Thing Groups
Billing Groups
Jobs
Greengrass
Secure
Defend
Act
Test
Software
Things
Successfully registered your thing
Search things
Configure fleet indexing
MyDemoDevice
NO TYPE
We've added a new, powerful fleet index
You can use search queries to find your devices and you can use attributes to narrow the results.
Next

AWS IoT

- Monitor
- Onboard
- Manage
- Greengrass
- **Secure**
  - Certificates
  - **Policies**
  - CAs
  - Role Aliases
  - Authorizers
- Defend
- Act
- Test
- Software
- Settings

You don't have any policies yet

AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadows).

Learn more

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name

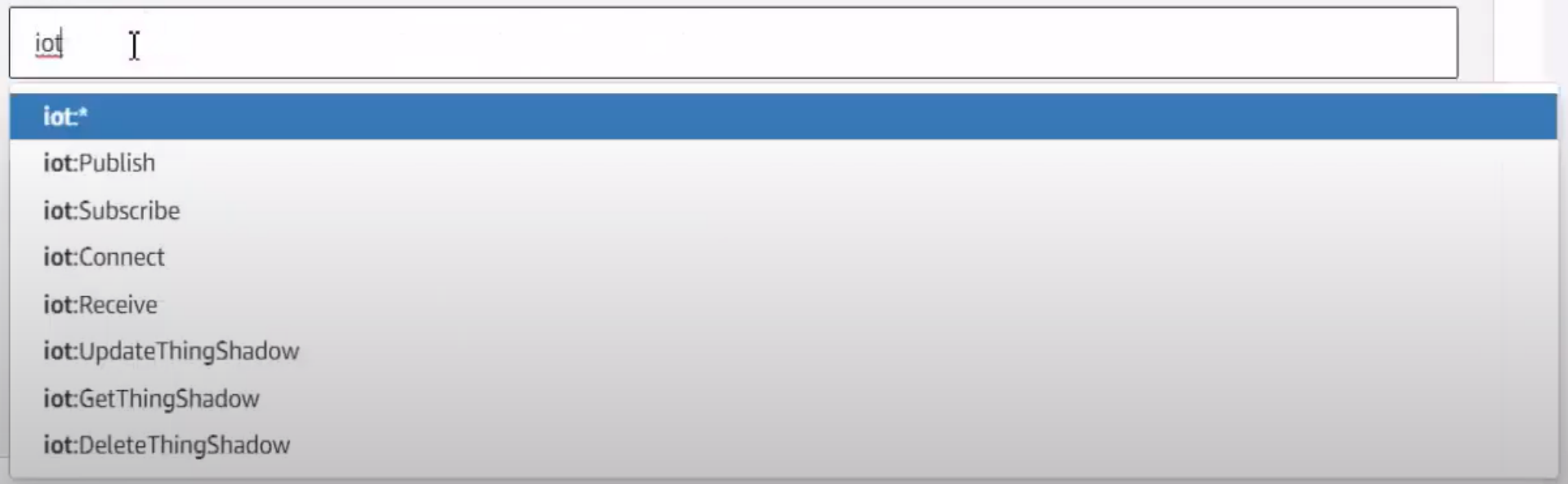
MyIoTPolicy

Add statements

Policy statements define the types of actions that can be performed by a resource.

Advanced mode

Action



Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

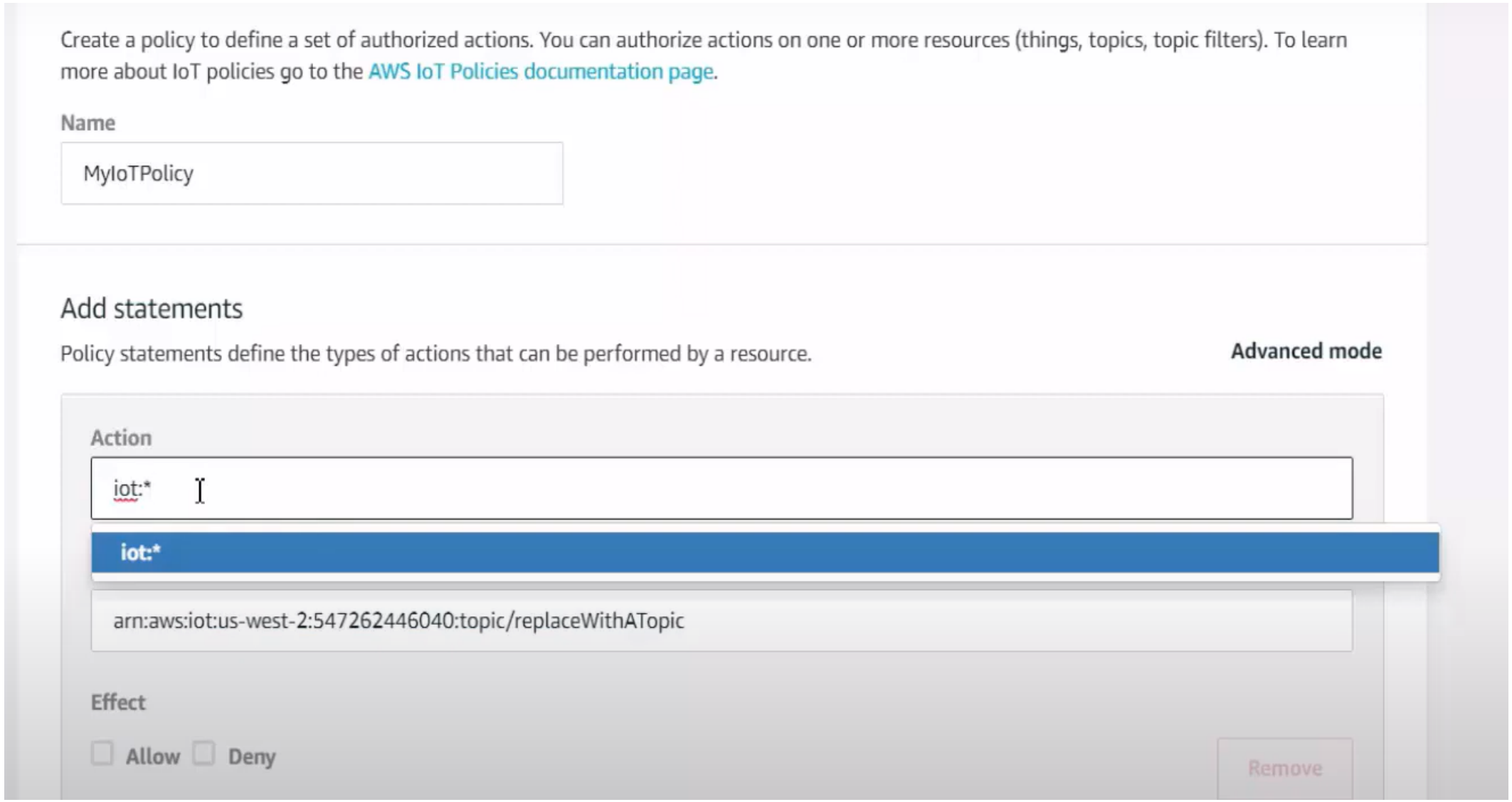
Name

MyIoTPolicy

Add statements

Policy statements define the types of actions that can be performed by a resource.

Advanced mode



Add statements

Policy statements define the types of actions that can be performed by a resource.

Advanced mode

Action

iot:*

Resource ARN

*

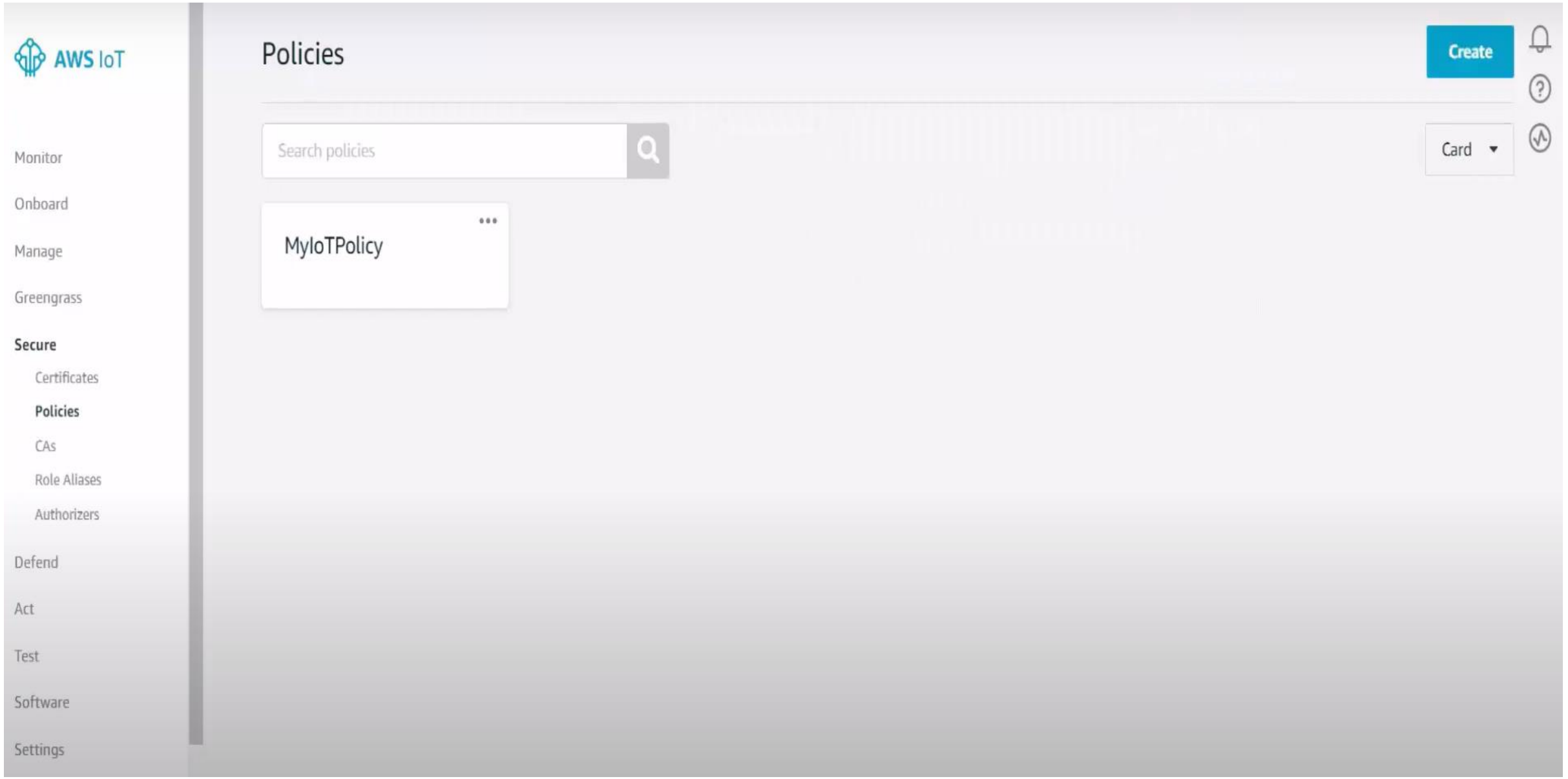
Effect

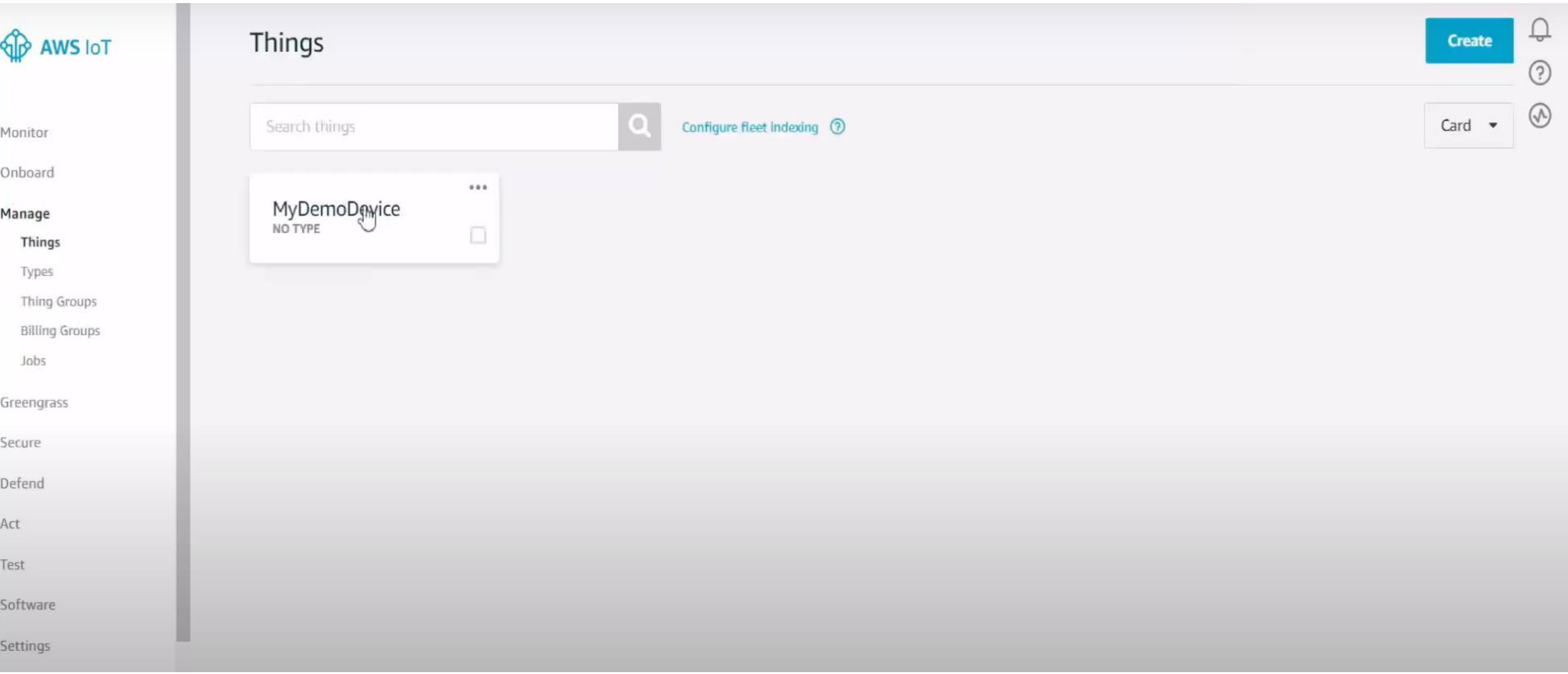
Allow Deny

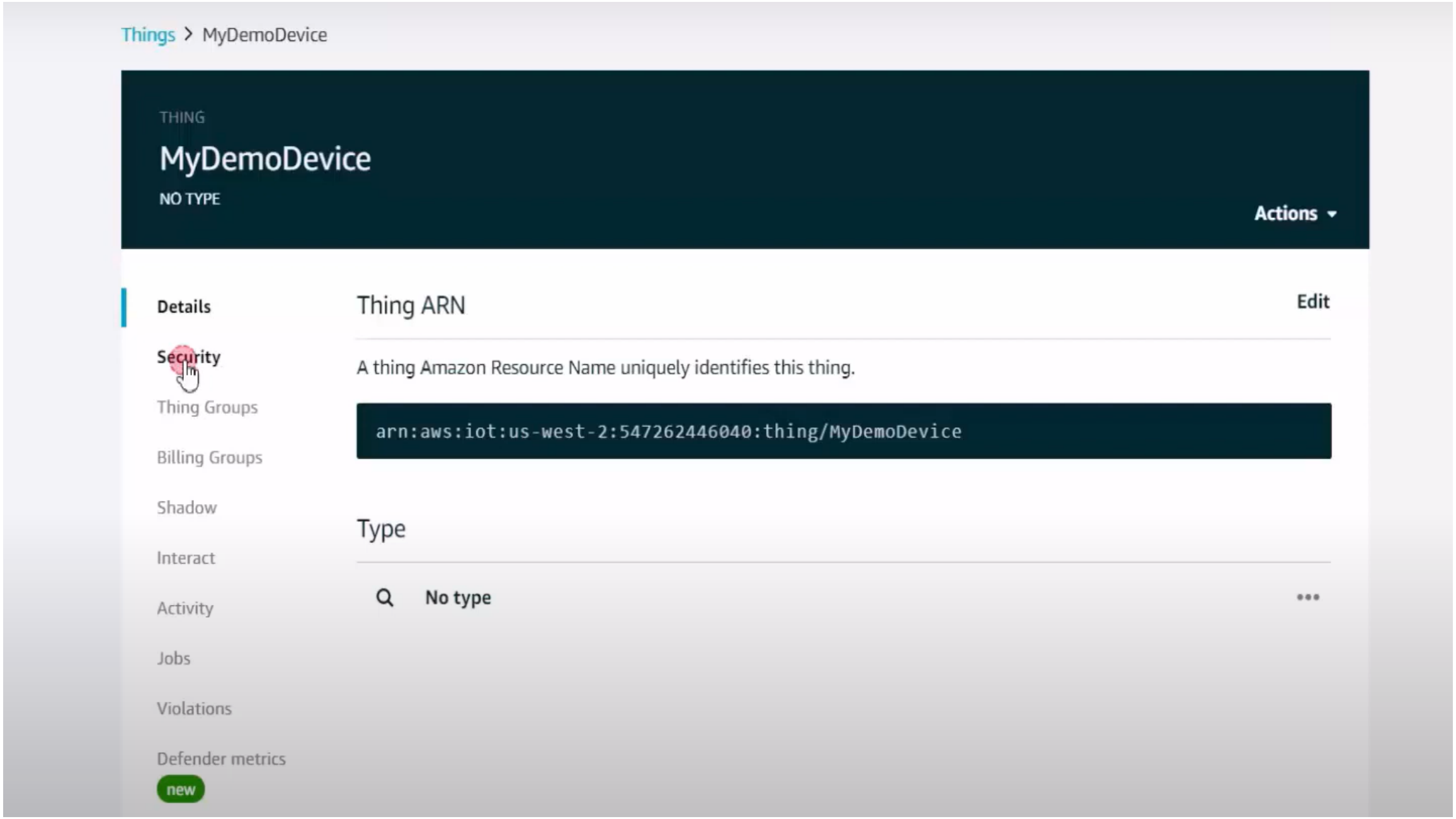
Remove

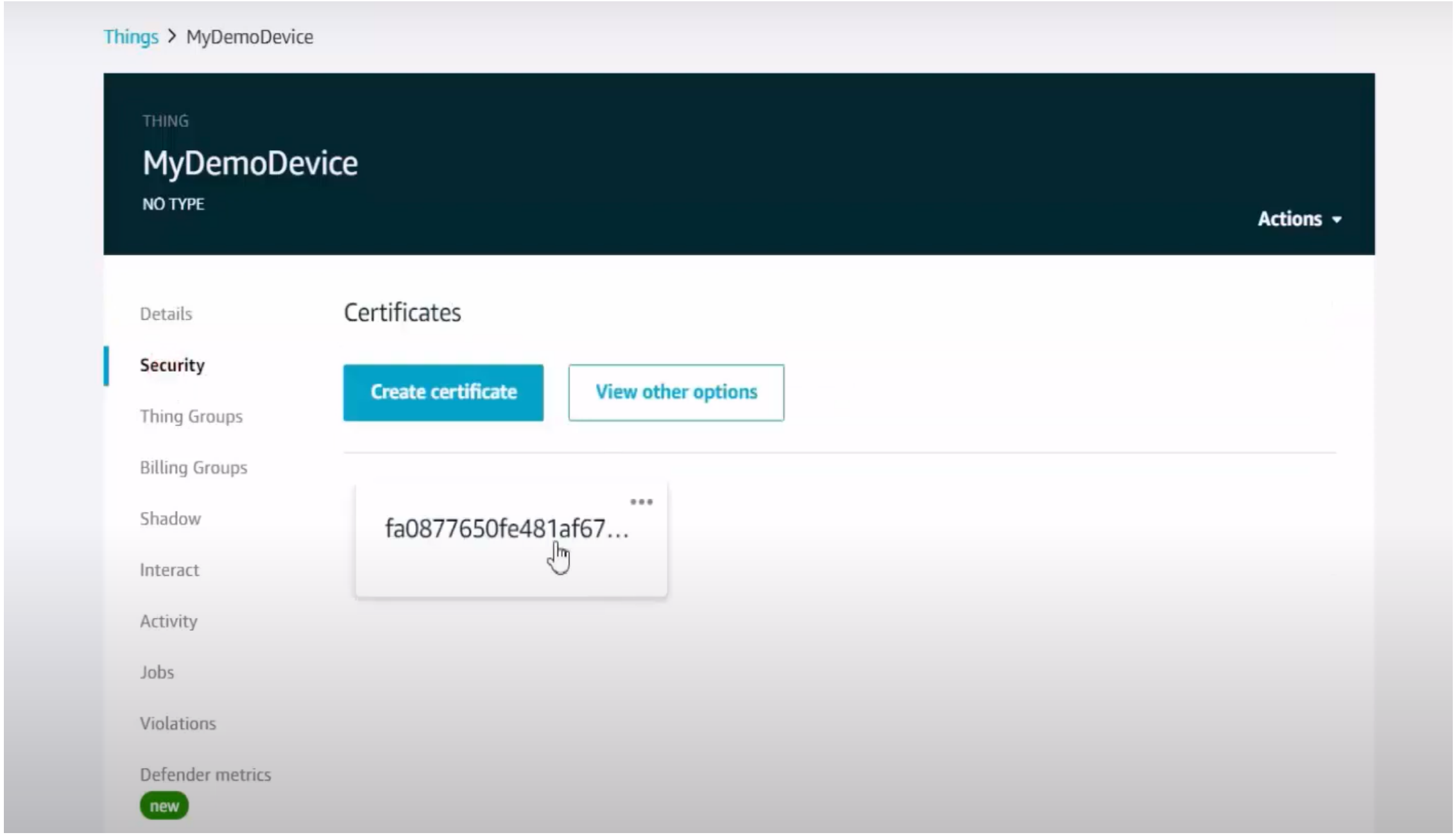
Add statement

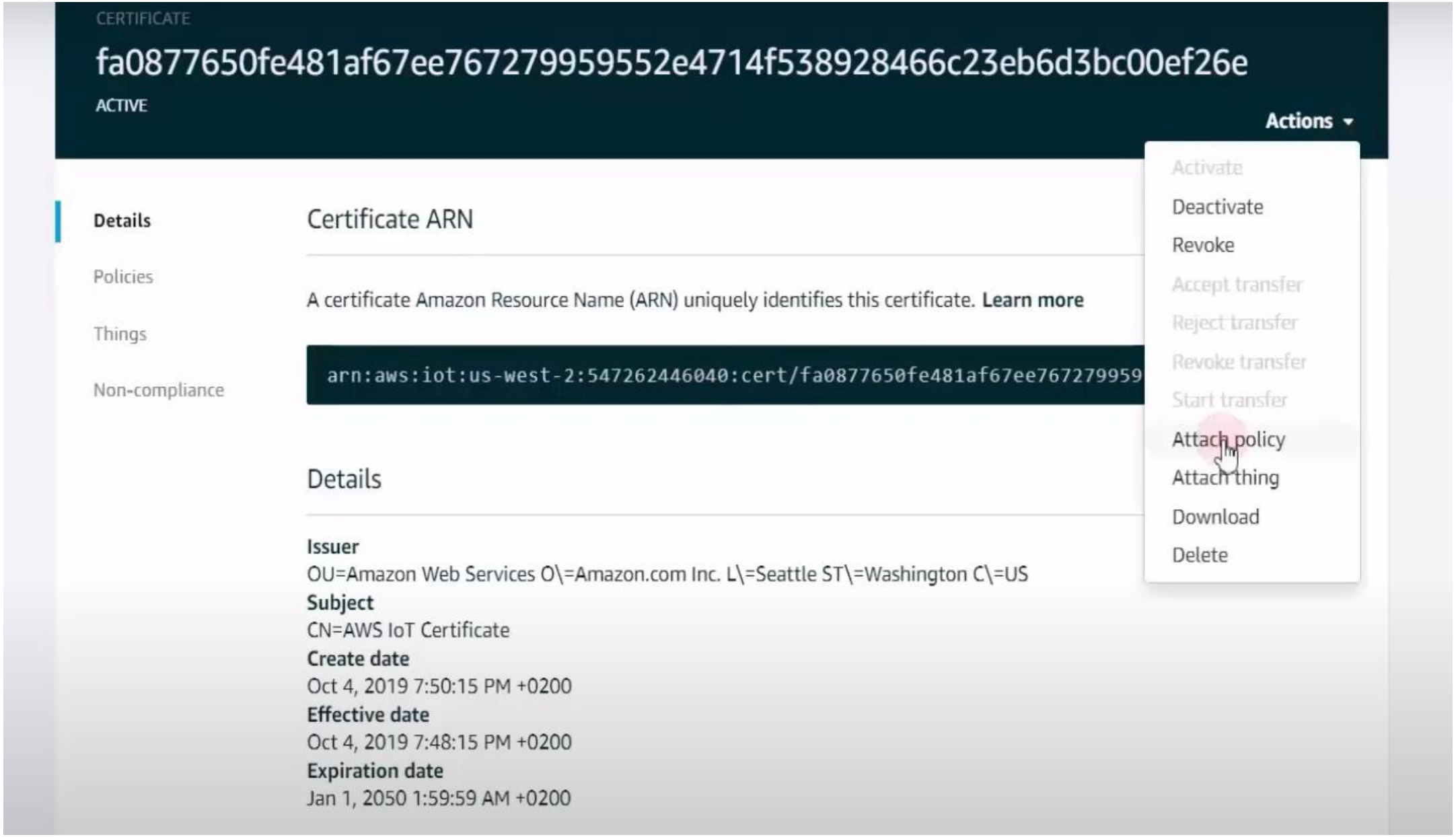
Create

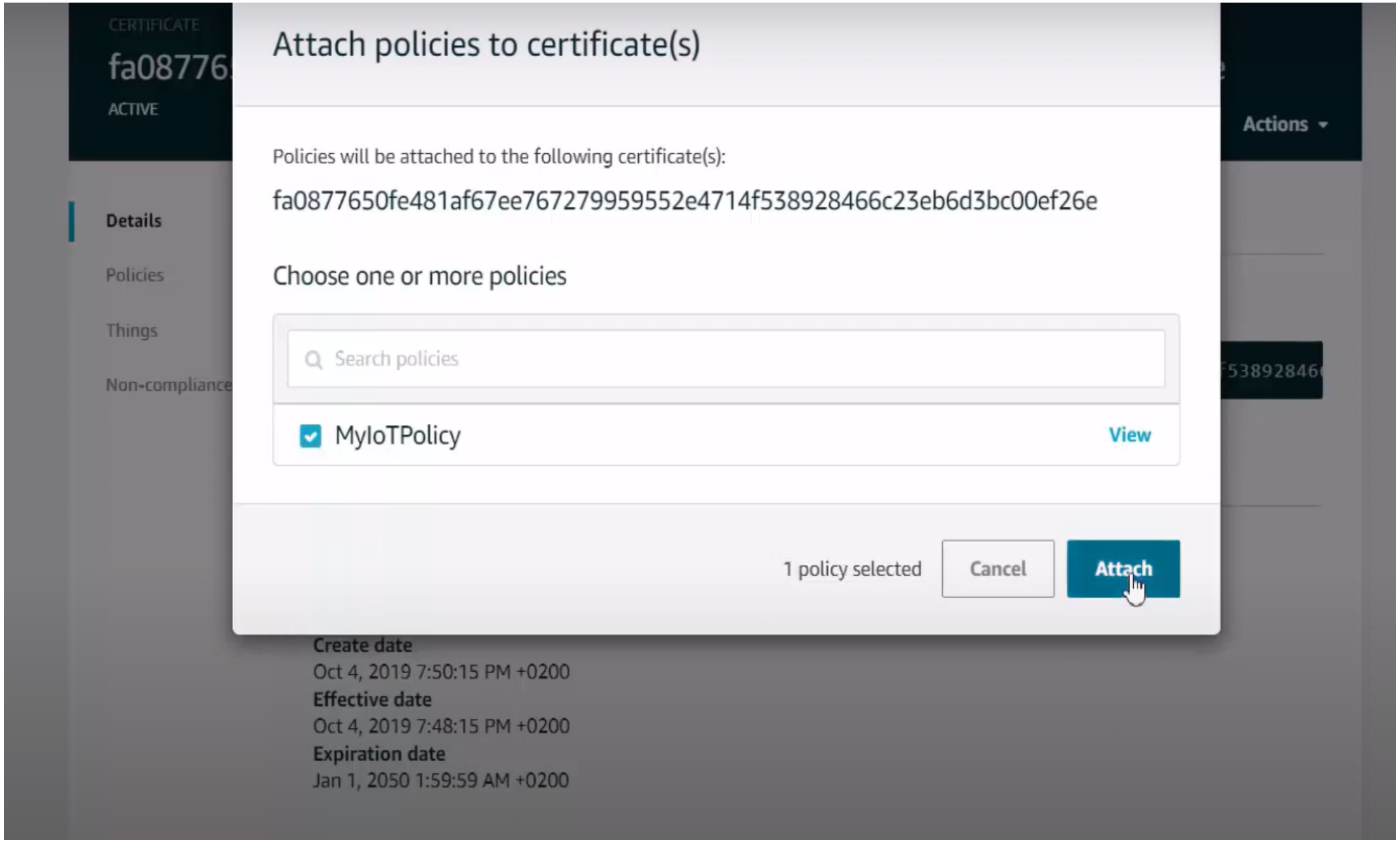

AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Certificates
Policies
CAs
Role Aliases
Authorizers
Defend
Act
Test
Software
Settings
Policies
Create
Search policies
Card
MyIoTPolicy

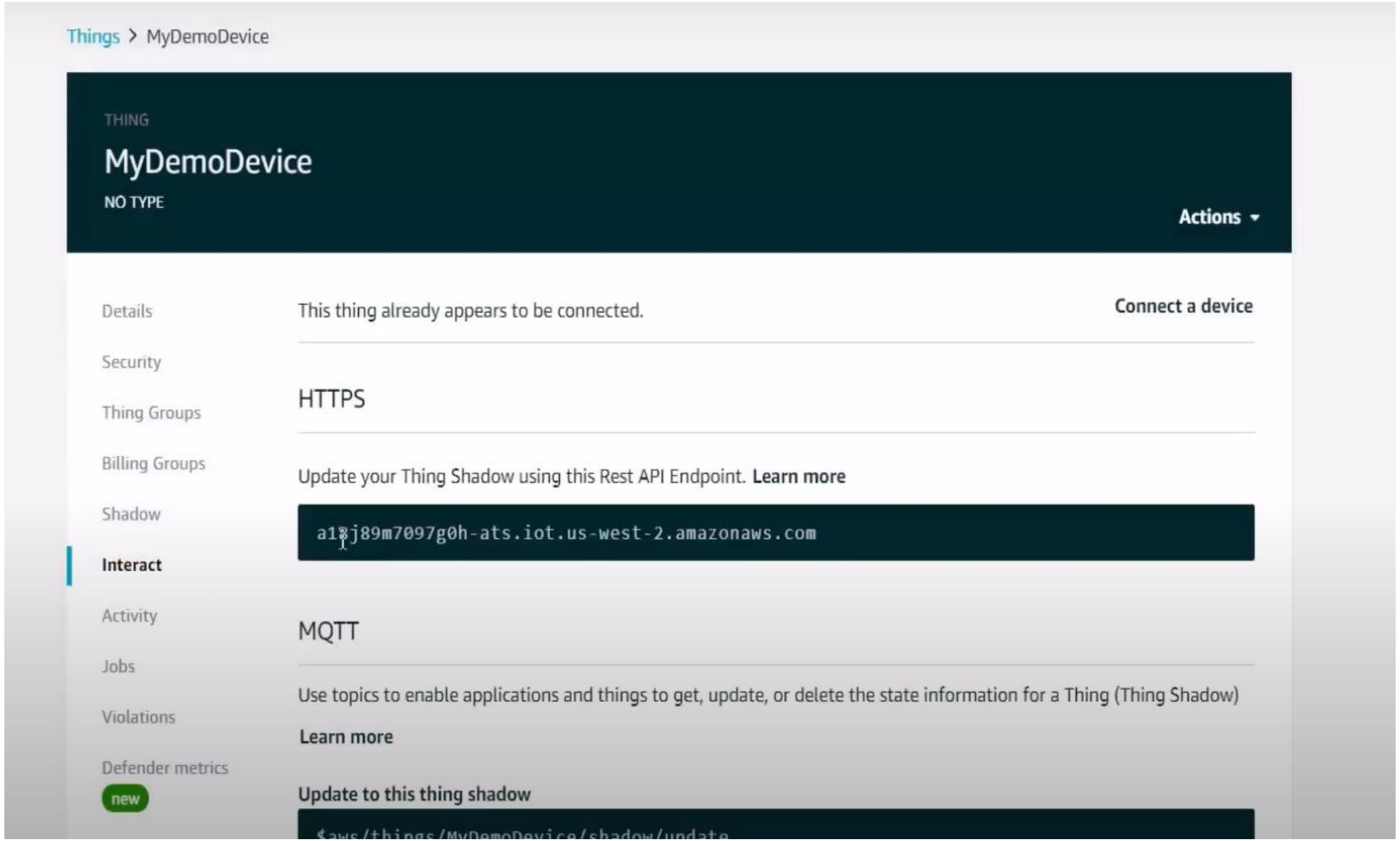

AWS IoT
Monitor
Onboard
Manage
Things
Types
Thing Groups
Billing Groups
Jobs
Greengrass
Secure
Defend
Act
Test
Software
Settings
Things
Create
Search things
Configure fleet indexing
Card
MyDemoDevice
NO TYPE

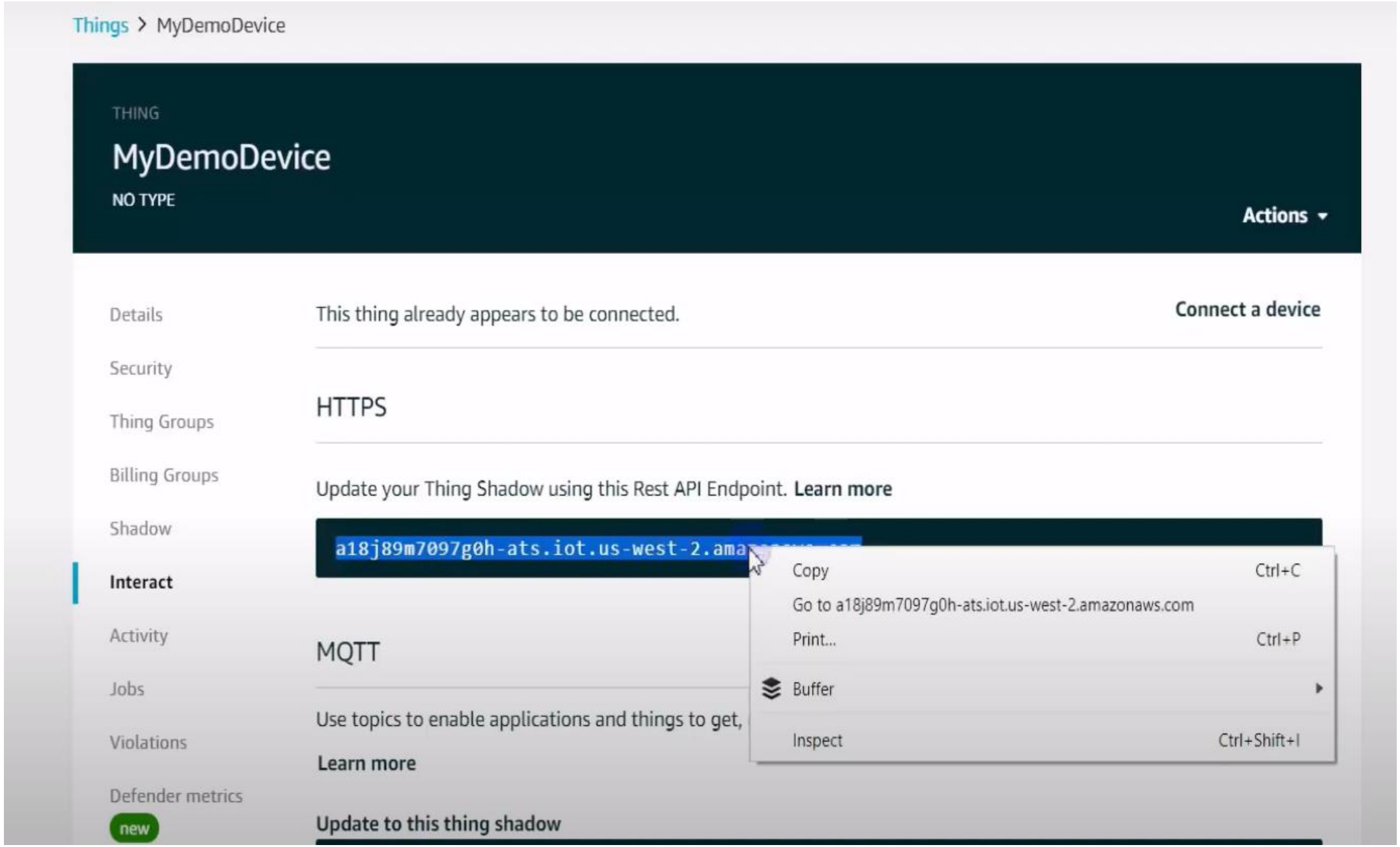

Things > MyDemoDevice
THING
MyDemoDevice
NO TYPE
Actions
Details
Security
Thing Groups
Billing Groups
Shadow
Interact
Activity
Jobs
Violations
Defender metrics
new
Thing ARN
Edit
A thing Amazon Resource Name uniquely identifies this thing.
arn:aws:iot:us-west-2:547262446040:thing/MyDemoDevice
Type
No type

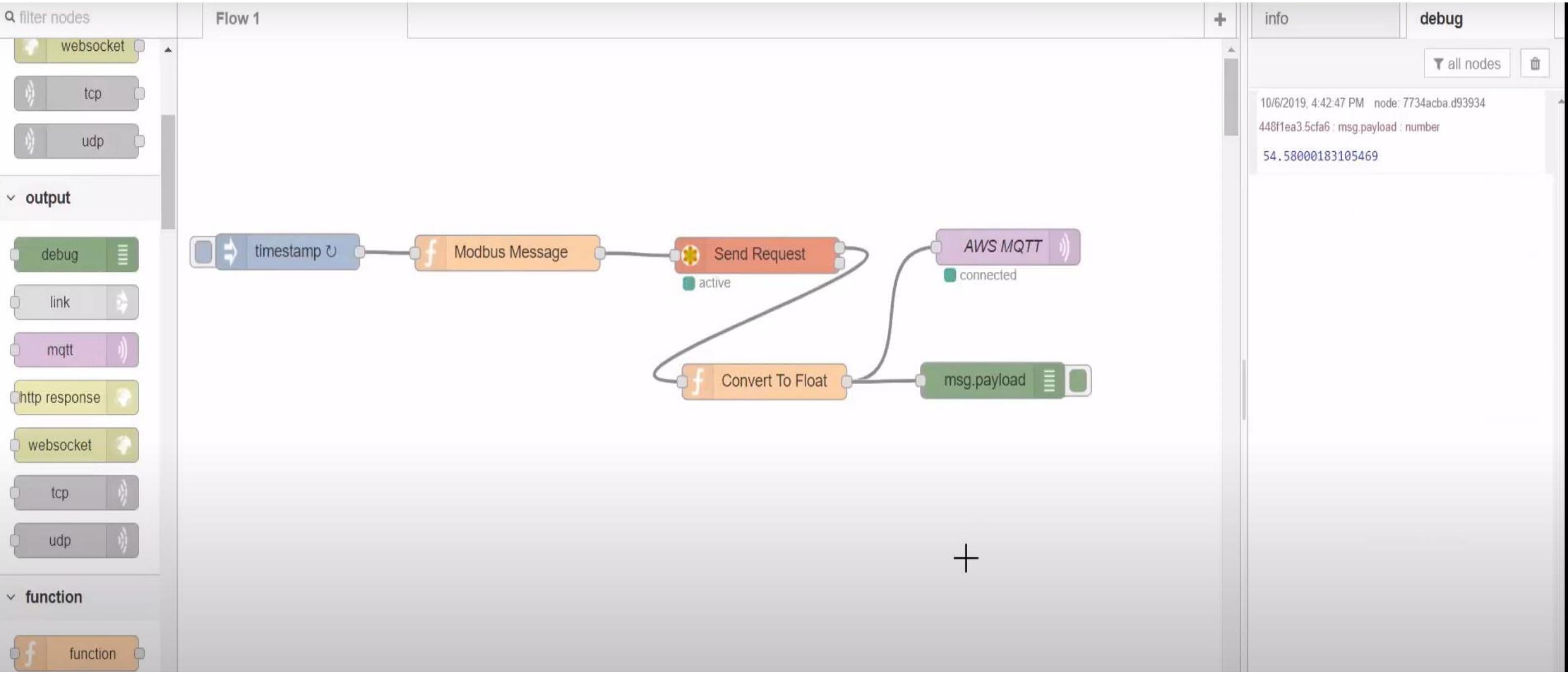

Things > MyDemoDevice
THING
MyDemoDevice
NO TYPE
Actions
Details
Security
Thing Groups
Billing Groups
Shadow
Interact
Activity
Jobs
Violations
Defender metrics
new
Certificates
Create certificate
View other options
fa0877650fe481af67...

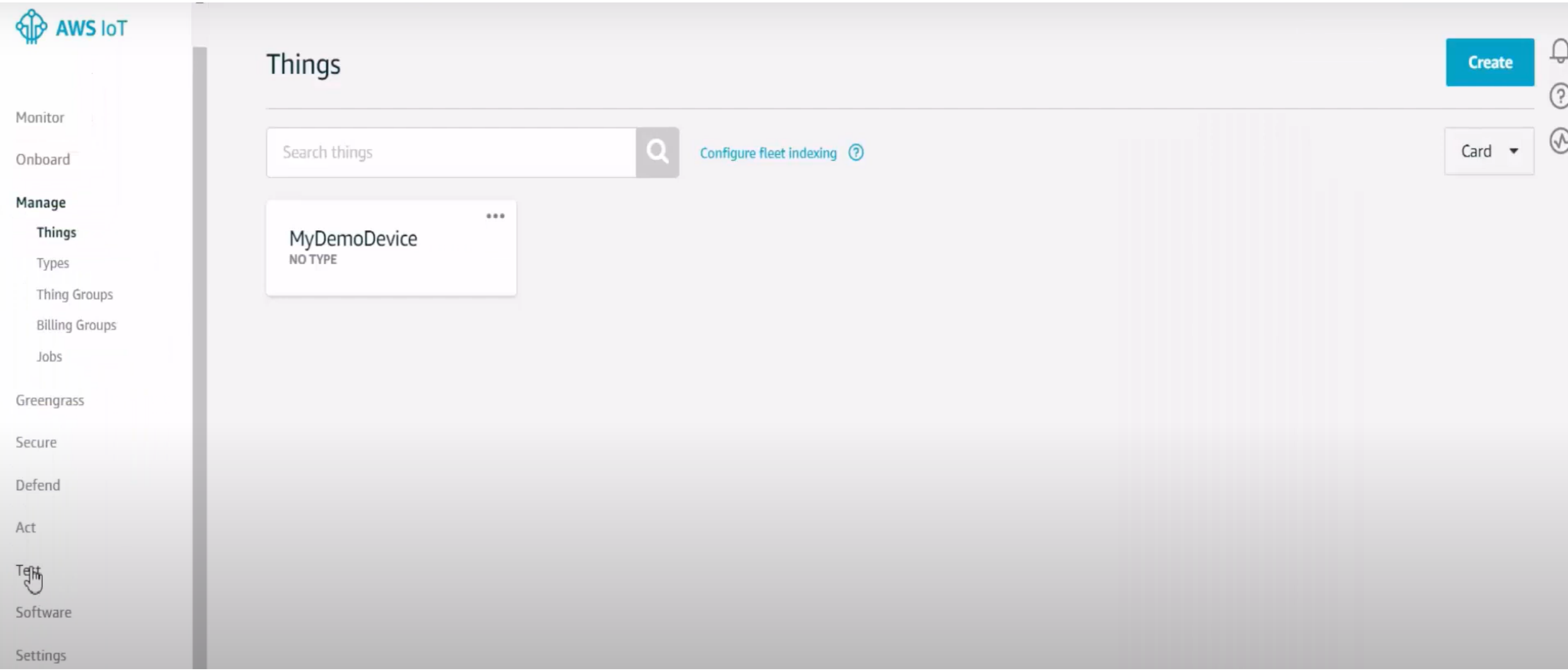

CERTIFICATE
fa0877650fe481af67ee767279959552e4714f538928466c23eb6d3bc00ef26e
ACTIVE
Actions
Activate
Deactivate
Revoke
Accept transfer
Reject transfer
Revoke transfer
Start transfer
Attach policy
Attach thing
Download
Delete
Details
Policies
Things
Non-compliance
Certificate ARN
A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. Learn more
arn:aws:iot:us-west-2:547262446040:cert/fa0877650fe481af67ee767279959
Details
Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Oct 4, 2019 7:50:15 PM +0200
Effective date
Oct 4, 2019 7:48:15 PM +0200
Expiration date
Jan 1, 2050 1:59:59 AM +0200


CERTIFICATE
fa08776
ACTIVE
Actions
Details
Policies
Things
Non-compliance
Attach policies to certificate(s)
Policies will be attached to the following certificate(s):
fa0877650fe481af67ee767279959552e4714f538928466c23eb6d3bc00ef26e
Choose one or more policies
Search policies
MyIoTPolicy
View
1 policy selected
Cancel
Attach
Create date
Oct 4, 2019 7:50:15 PM +0200
Effective date
Oct 4, 2019 7:48:15 PM +0200
Expiration date
Jan 1, 2050 1:59:59 AM +0200


Things > MyDemoDevice
THING
MyDemoDevice
NO TYPE
Actions
Details
Security
Thing Groups
Billing Groups
Shadow
Interact
Activity
Jobs
Violations
Defender metrics
new
This thing already appears to be connected.
Connect a device
HTTPS
Update your Thing Shadow using this Rest API Endpoint. Learn more
a1xxj89m7097g0h-ats.iot.us-west-2.amazonaws.com
MQTT
Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)
Learn more
Update to this thing shadow
$aws/things/MyDemoDevice/shadow/update


Things > MyDemoDevice
THING
MyDemoDevice
NO TYPE
Actions
Details
Security
Thing Groups
Billing Groups
Shadow
Interact
Activity
Jobs
Violations
Defender metrics
new
This thing already appears to be connected.
Connect a device
HTTPS
Update your Thing Shadow using this Rest API Endpoint. Learn more
a18j89m7097g0h-ats.iot.us-west-2.ama
Copy Ctrl+C
Go to a18j89m7097g0h-ats.iot.us-west-2.amazonaws.com
Print... Ctrl+P
Buffer
Inspect Ctrl+Shift+I
MQTT
Use topics to enable applications and things to get,
Learn more
Update to this thing shadow


filter nodes
Flow 1
info
debug
websocket
tcp
udp
output
debug
link
mqtt
http response
websocket
tcp
udp
function
function
timestamp ↻
Modbus Message
Send Request
active
AWS MQTT
connected
Convert To Float
msg.payload
all nodes
10/6/2019, 4:42:47 PM node: 7734acba.d93934
448f1ea3.5cfa6 : msg.payload : number
54.58000183105469


AWS IoT
Monitor
Onboard
Manage
Things
Types
Thing Groups
Billing Groups
Jobs
Greengrass
Secure
Defend
Act
Test
Software
Settings
Things
Create
Search things
Configure fleet indexing
Card
MyDemoDevice
NO TYPE

AWS IoT

MQTT client

Monitor

Onboard

Manage

Greengrass

Secure

Defend

Act

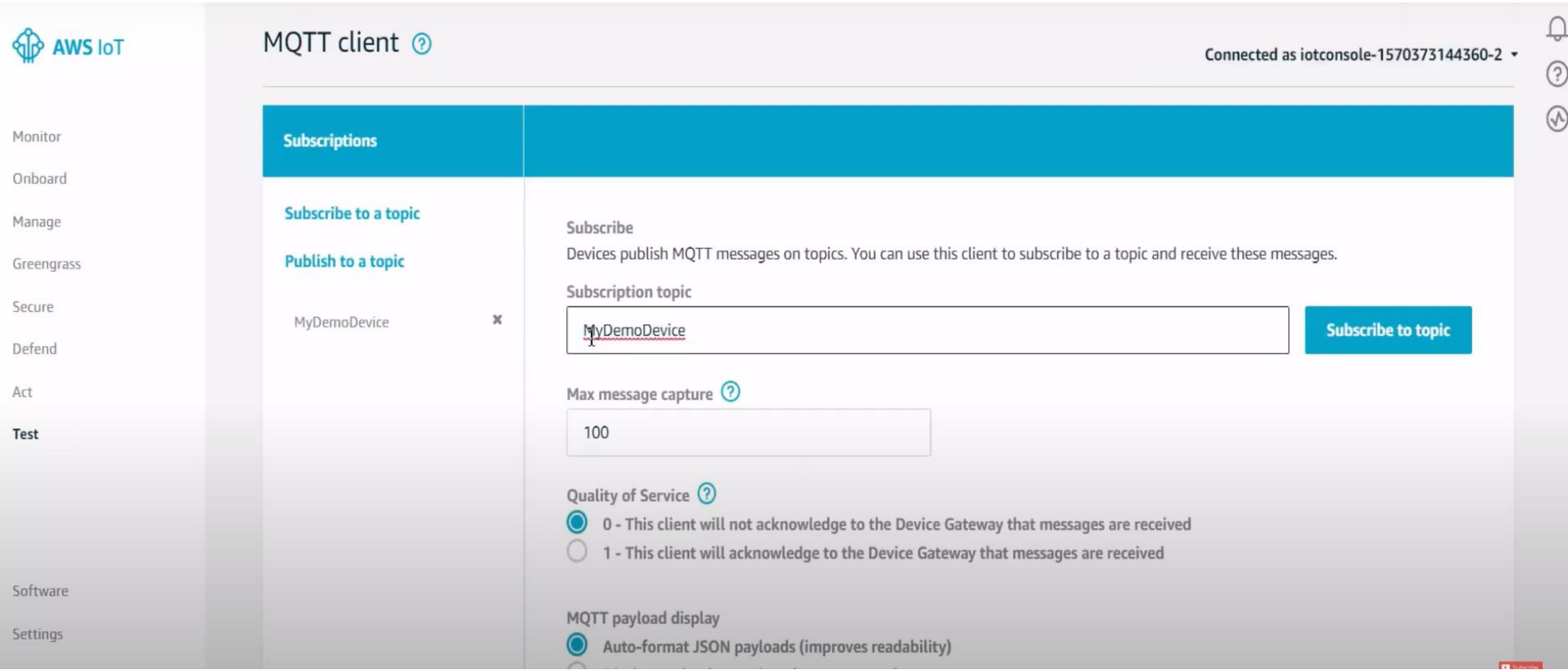
Test

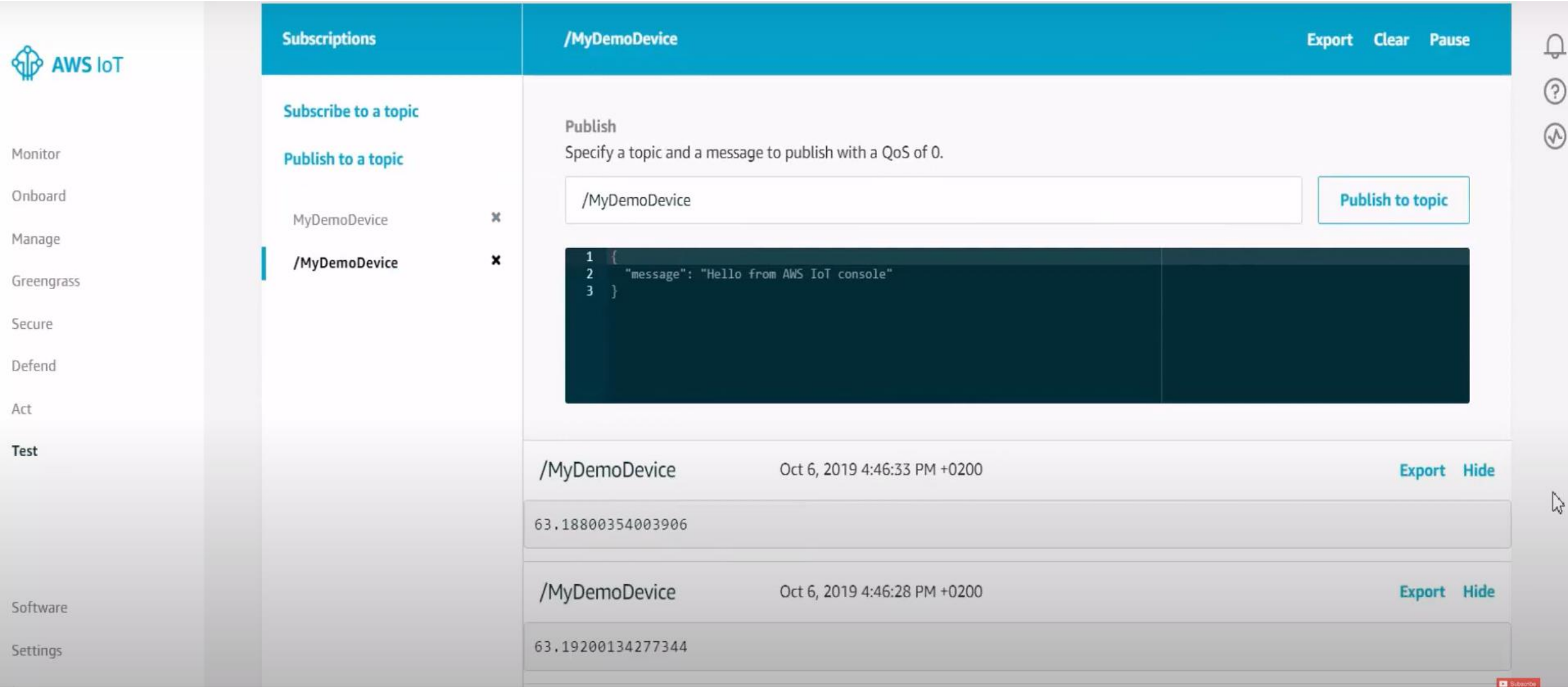
Connecting to Device Gateway...

Software

Settings

Learn


AWS IoT
MQTT client
Connected as iotconsole-1570373144360-2
Monitor
Onboard
Manage
Greengrass
Secure
Defend
Act
Test
Software
Settings
Subscriptions
Subscribe to a topic
Publish to a topic
MyDemoDevice
Subscribe
Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.
Subscription topic
MyDemoDevice
Subscribe to topic
Max message capture
100
Quality of Service
0 - This client will not acknowledge to the Device Gateway that messages are received
1 - This client will acknowledge to the Device Gateway that messages are received
MQTT payload display
Auto-format JSON payloads (improves readability)


AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Defend
Act
Test
Software
Settings
Subscriptions
/MyDemoDevice
Export
Clear
Pause
Subscribe to a topic
Publish to a topic
MyDemoDevice
/MyDemoDevice
Publish
Specify a topic and a message to publish with a QoS of 0.
/MyDemoDevice
Publish to topic
1 {
2 "message": "Hello from AWS IoT console"
3 }
/MyDemoDevice
Oct 6, 2019 4:46:33 PM +0200
Export
Hide
63.18800354003906
/MyDemoDevice
Oct 6, 2019 4:46:28 PM +0200
Export
Hide
63.19200134277344