

# Cloud Tutorial: AWS IoT

---

CSE 521S Fall  
Sep. 17, 2020  
Ruixuan (Corey) Dai



Washington University in St. Louis

# XaaS: Basics in Cloud Computing

---



Washington University in St.Louis

# Cloud Computing

- Cloud computing provides **shared pool of configurable computing resource** to end users **on demand**
- Three service models
  - **IaaS (Infrastructure as a Service)**: virtual machines, storage, network ...
 

  - **PaaS (Platform as a Service)**: execution runtime, middleware, web server, database, development tool ...
 


  - **SaaS (Software as a Service)**: email, virtual desktop, games ...
 


# Cloud Services: On-premise Software

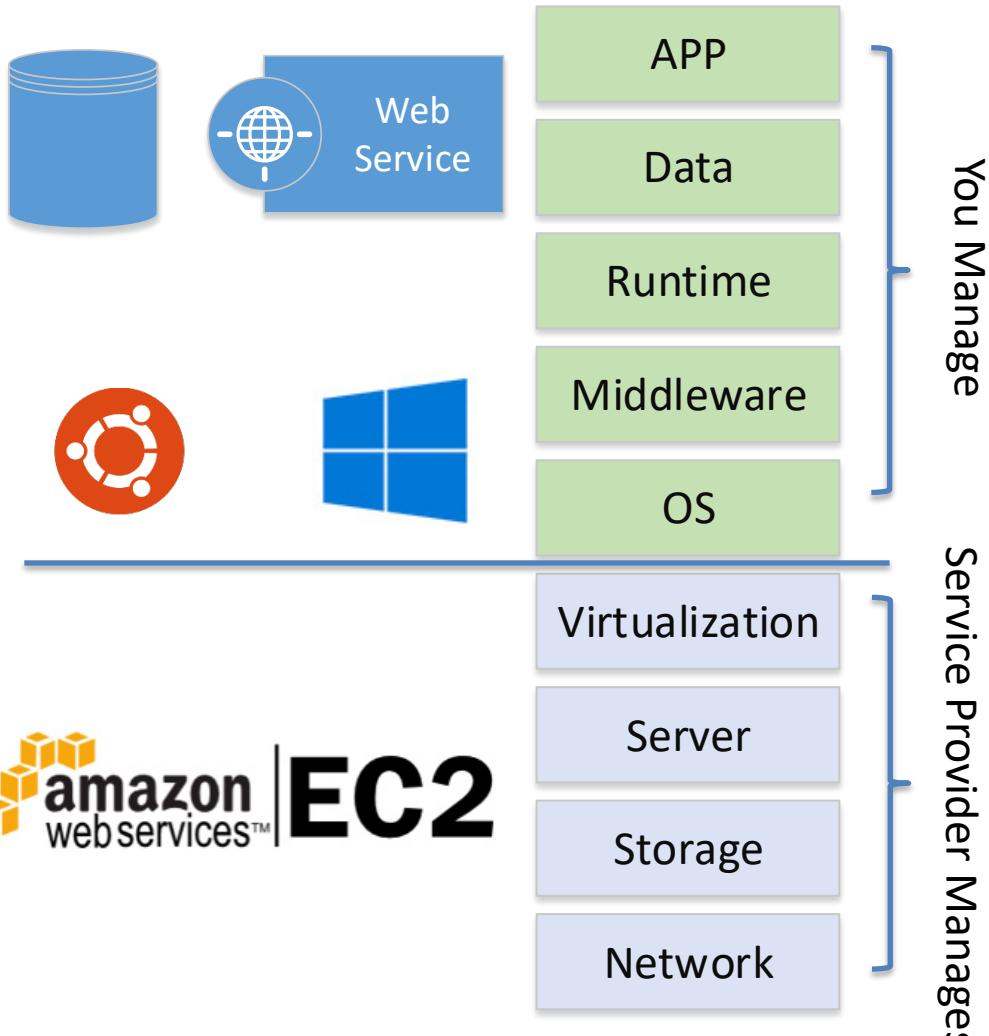
- Traditional
- installed and runs on personal computer
- You Manage and Deploy
  - ❑ Hardware
  - ❑ OS
  - ❑ Software
- Example
  - ❑ This presentation



# Infrastructure as a Service (IaaS)

## ➤ IaaS

- ❑ "physical server box"
- ❑ Virtual Machine
  - Memory
  - Storage
  - CPU
  - Network



## ➤ Example

- ❑ AWS EC2
- ❑ AWS EFS

## ➤ Use case

- ❑ Build up your VM cluster

# Platform as a Service (PaaS)

## ➤ PaaS

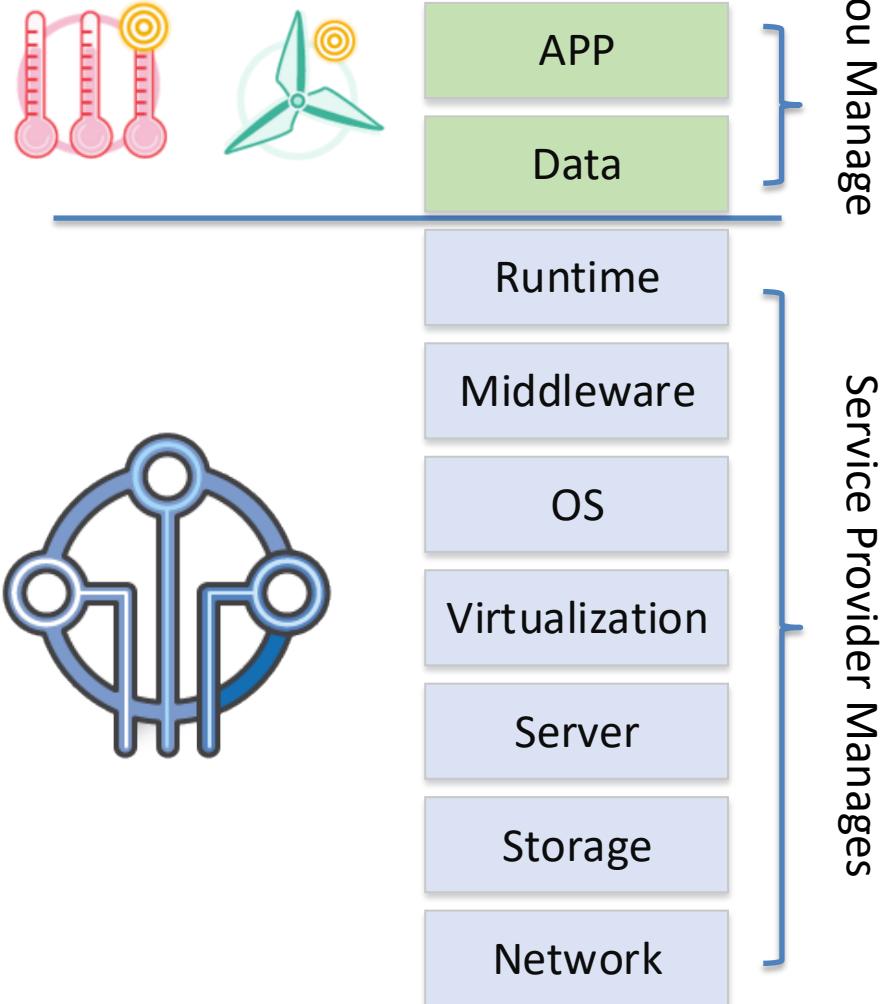
- ❑ You get a framework
- ❑ Host Application
- ❑ Tools

## ➤ Example

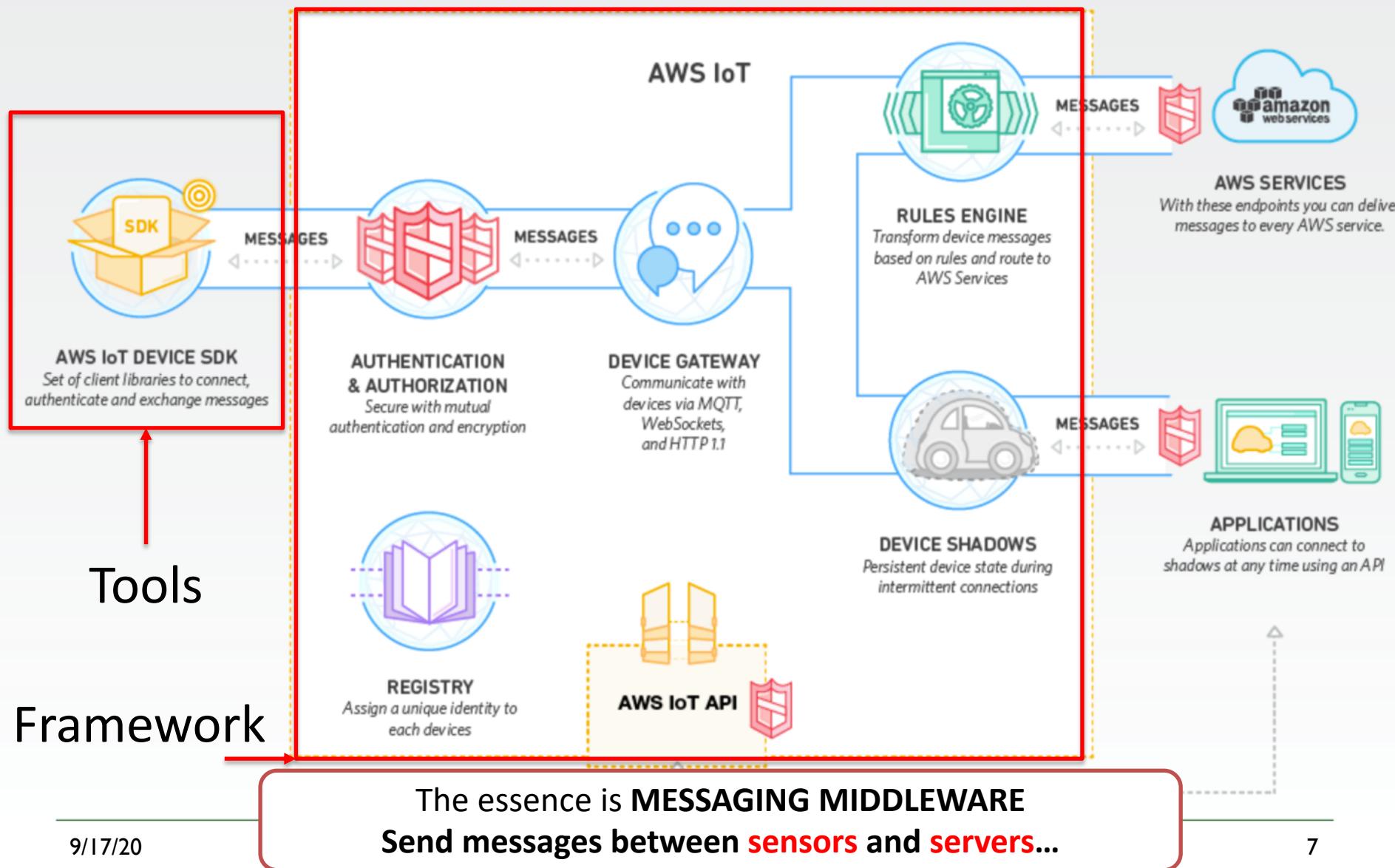
- ❑ AWS IoT

## ➤ Use case

- ❑ Build up your smart A/C controller



# PaaS Example: AWS IoT



# Software as a Service (SaaS)

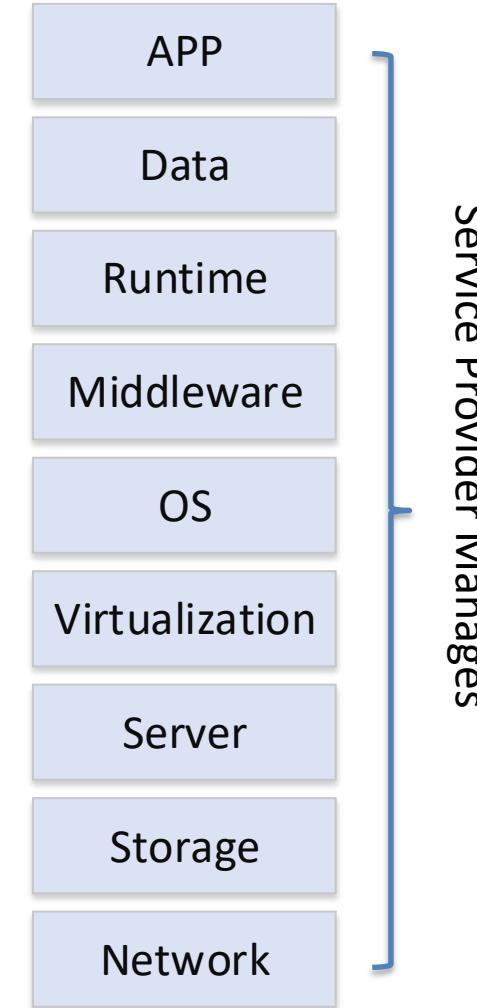
## ➤ SaaS

- ❑ You get a whole solution

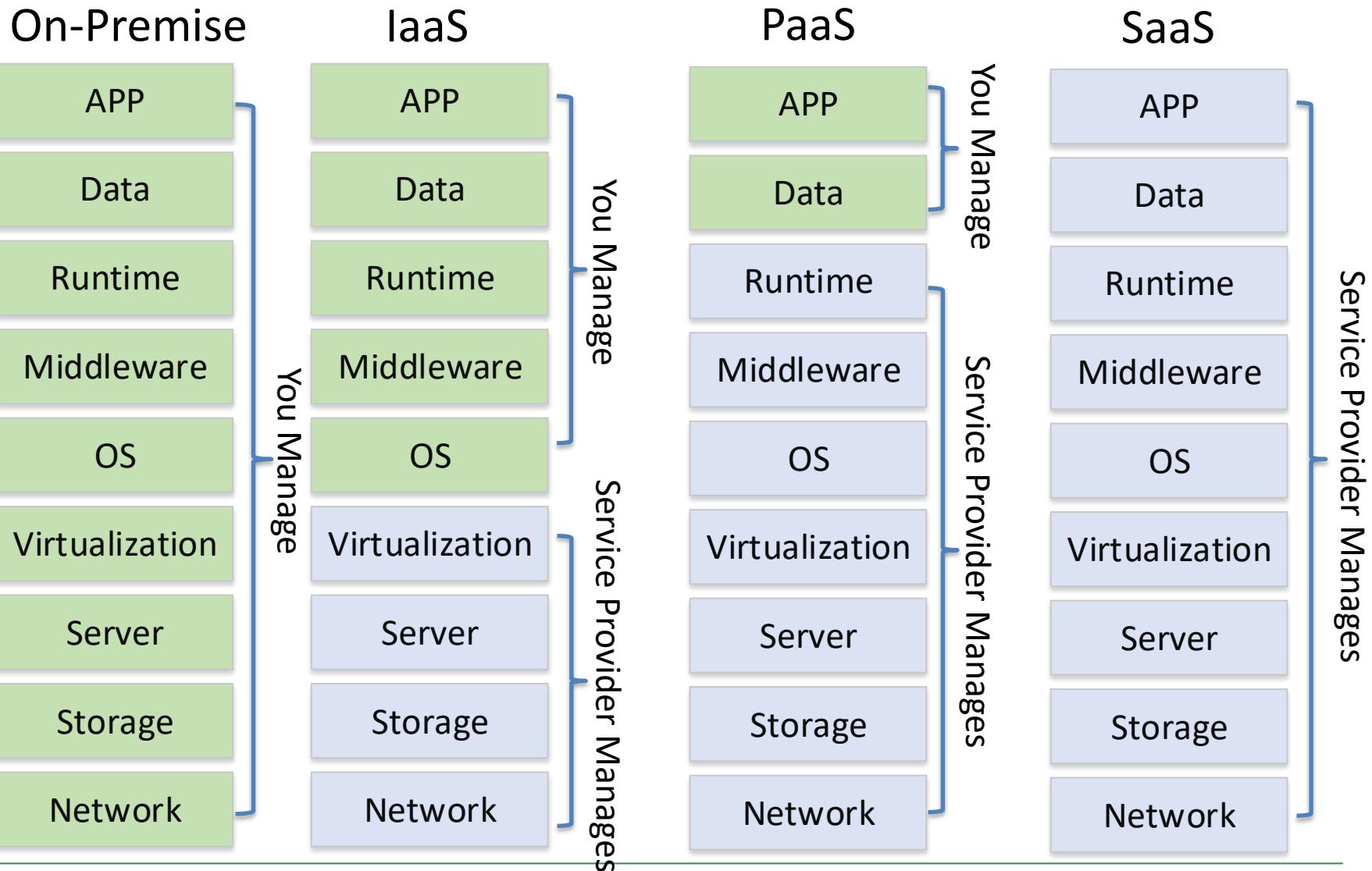


## ➤ Example

- ❑ Gmail
- ❑ Dropbox
- ❑ Office365



# XaaS: A Recap



# Tutorial: Hello! AWS IoT!!

---



Washington University in St.Louis

# Internet-of-Things

## ➤ Internet-of-Things

### □ Devices

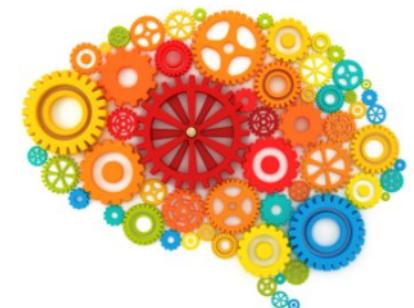
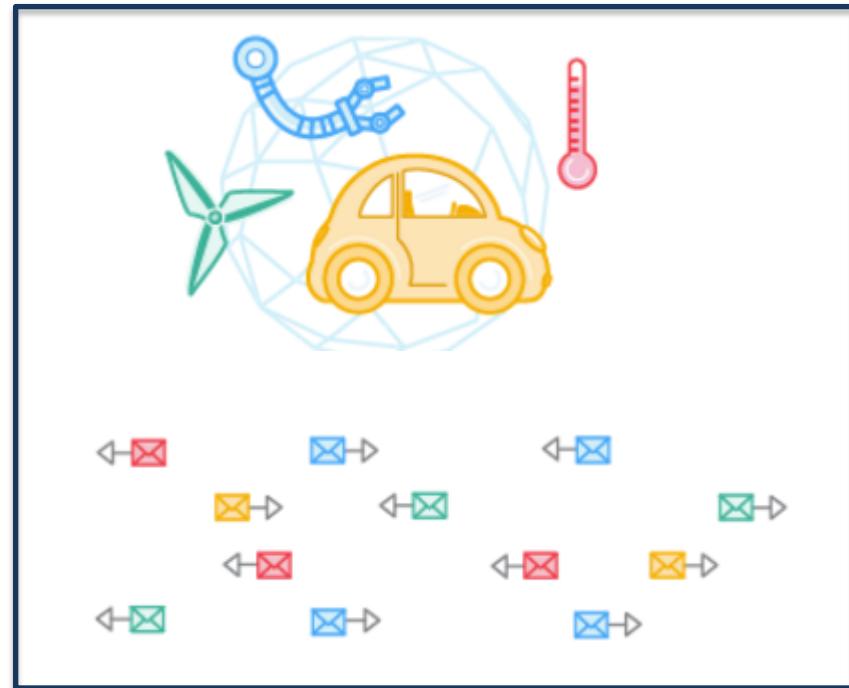
- Different Types
  - Sensors, actuators

### □ Data and Command

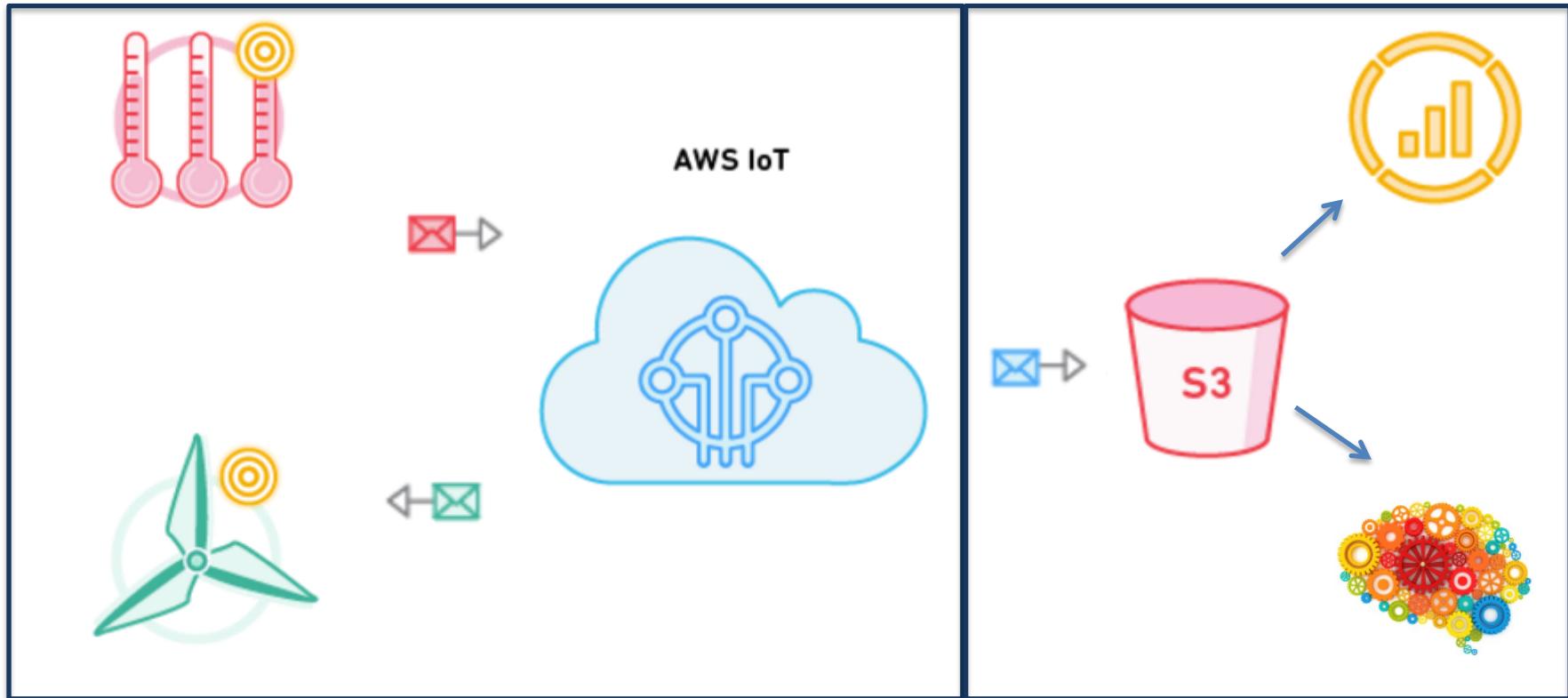
- Sensing the world
- Give Response

### □ Challenge

- United: Connected + Communication
- Smart: Data Analytics + Strategy



# Solution: AWS IoT



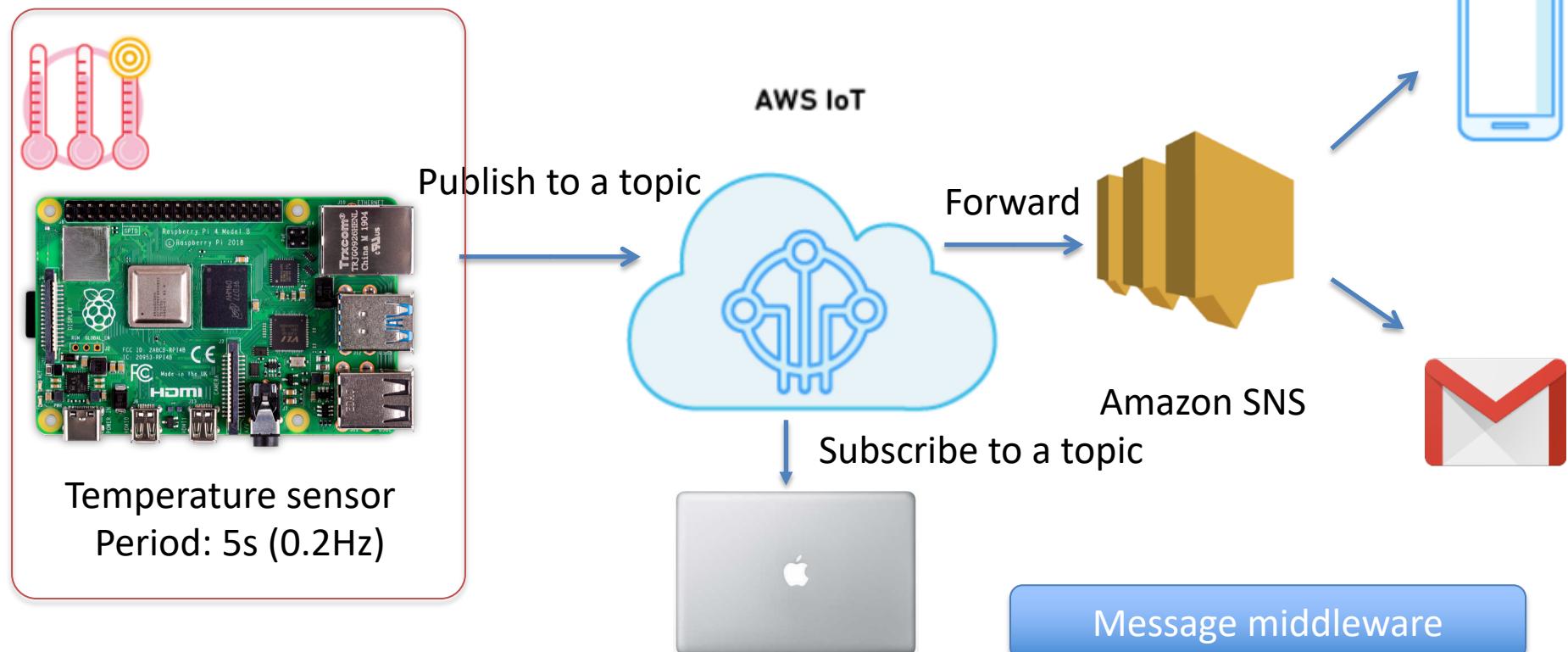
United: Connect + Communication

Stated: “Thing Shadow”

Smart: Other Cloud Service  
Data Storage  
Machine Learning

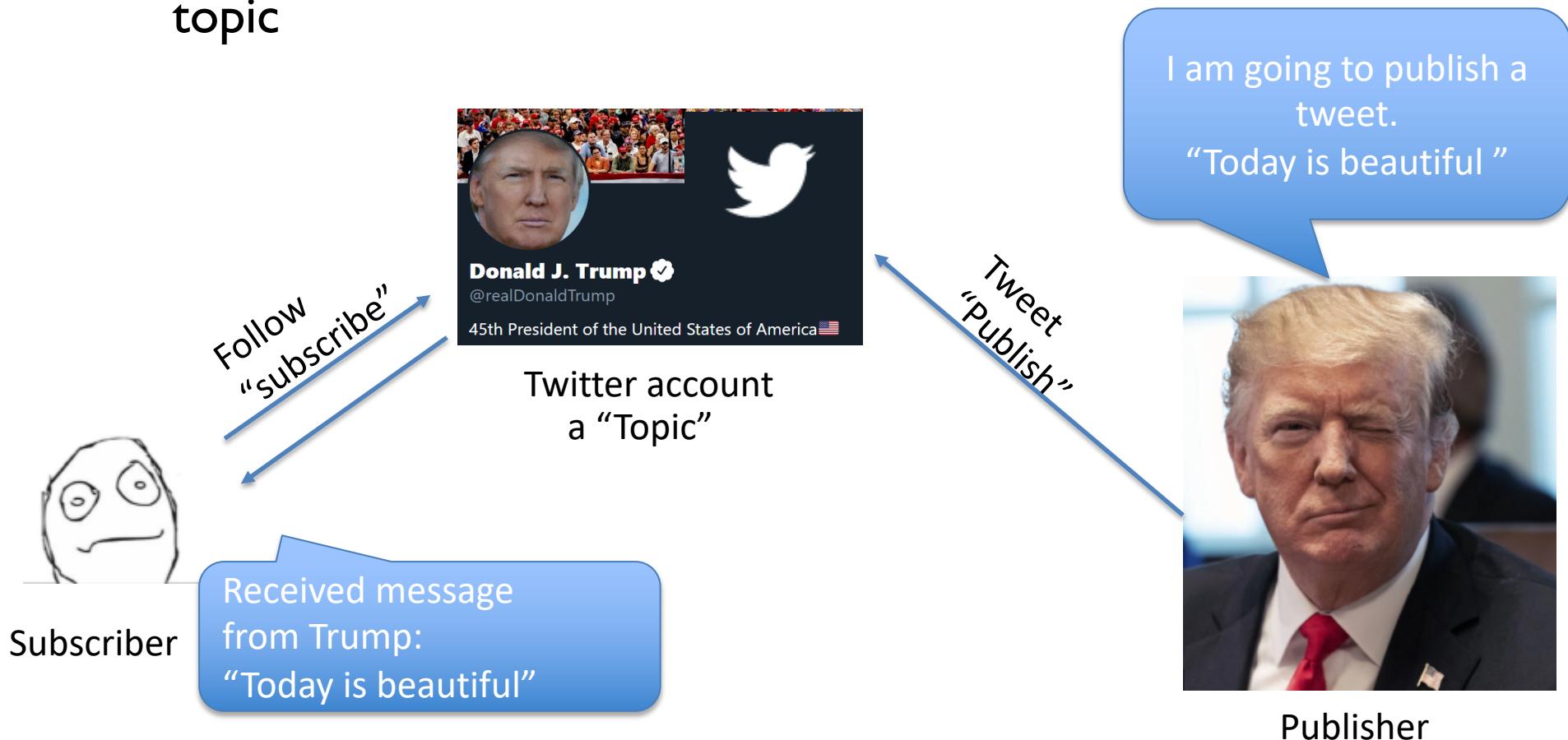
# Tutorial: Hello AWS IoT!

- Key concepts:
  - Publisher (e.g. Sensor), Subscriber (e.g. Server), **Topic**
    - **Topic is used to identify the message.**
- Not a traditional “peer-to-peer” communication.



# Basic Interact: Subscribe/Publish

- You can define your own Topic (Twitter Account)
- Subscriber can receive the message you published to your topic



# Resources

---

## ➤ Amazon IoT

- <http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html>

## ➤ Raspberry Pi

- <https://www.raspberrypi.org/>

## ➤ Resource list for course projects

- [http://cps.cse.wustl.edu/index.php>List\\_of\\_Projects](http://cps.cse.wustl.edu/index.php>List_of_Projects)

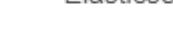
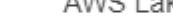
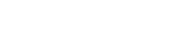
## ➤ Apply for \$40 credits for Amazon AWS

- <https://aws.amazon.com/education/awseducate/apply/>

# Get into AWS Manage Console

- Create your own AWS account
- Sign In IoT Manage Console
  - <https://aws.amazon.com/iot/>

Group
A-Z

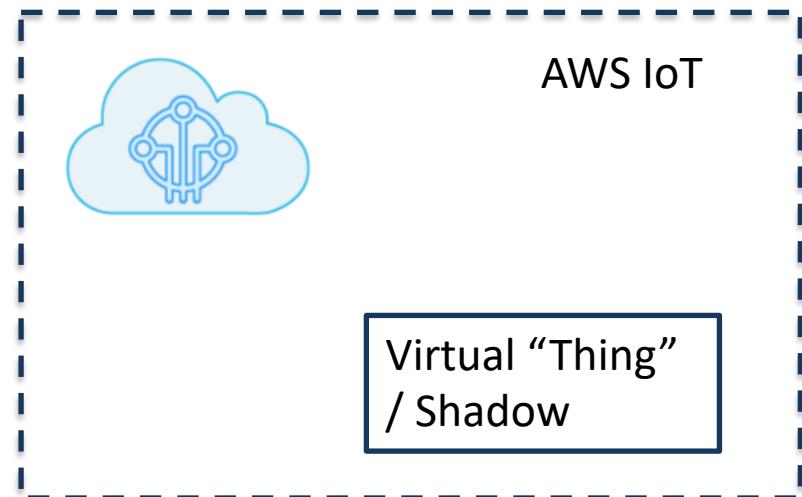
<div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Compute</b> </div> <div style="margin-bottom: 5px;">  EC2         </div> <div>  Lightsail          </div> <div>  ECR         </div> <div>  ECS         </div> <div>  EKS         </div> <div>  Lambda         </div> <div>  Batch         </div> <div>  Elastic Beanstalk         </div> <div>  Serverless Application Repository         </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Robotics</b> </div> <div style="margin-bottom: 5px;">  AWS RoboMaker         </div> <div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Blockchain</b> </div> <div style="margin-bottom: 5px;">  Amazon Managed Blockchain         </div> <div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Satellite</b> </div> <div style="margin-bottom: 5px;">  Ground Station         </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Analytics</b> </div> <div style="margin-bottom: 5px;">  Athena         </div> <div>  EMR         </div> <div>  CloudSearch         </div> <div>  Elasticsearch Service         </div> <div>  Kinesis         </div> <div>  QuickSight          </div> <div>  Data Pipeline         </div> <div>  AWS Glue         </div> <div>  AWS Lake Formation         </div> <div>  MSK         </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Business Applications</b> </div> <div style="margin-bottom: 5px;">  Alexa for Business         </div> <div>  Amazon Chime          </div> <div>  WorkMail         </div> <div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>End User Computing</b> </div> <div style="margin-bottom: 5px;">  WorkSpaces         </div> <div>  AppStream 2.0         </div> <div>  WorkDocs         </div> <div>  WorkLink         </div>
<div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Storage</b> </div> <div style="margin-bottom: 5px;">  S3         </div> <div>  EFS         </div> <div>  FSx         </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Management &amp; Governance</b> </div> <div style="margin-bottom: 5px;">  AWS Organizations         </div> <div>  CloudWatch         </div>	<div style="border-bottom: 1px solid #ccc; padding-bottom: 10px;"> <b>Security, Identity, &amp; Compliance</b> </div> <div style="margin-bottom: 5px;">  IAM         </div> <div>  Resource Access Manager         </div>	<div style="border: 2px solid red; padding: 5px;"> <b>Internet Of Things</b> </div> <div style="margin-bottom: 5px;">  IoT Core         </div> <div>  Amazon FreeRTOS         </div> <div>  IoT 1 Click         </div>

# Step 1: Create a Virtual "Thing"

- Virtual “Thing”: a **virtual copy** of your thing
  - A Thing in AWS IoT has a **“shadow”**
    - a JSON document that is used to **store and retrieve current state information for a device.**
      - **E.g.** Battery level, Connectivity, data  
A “Dashboard” to show some info
  - **Shadow is a special topic in AWS IoT**

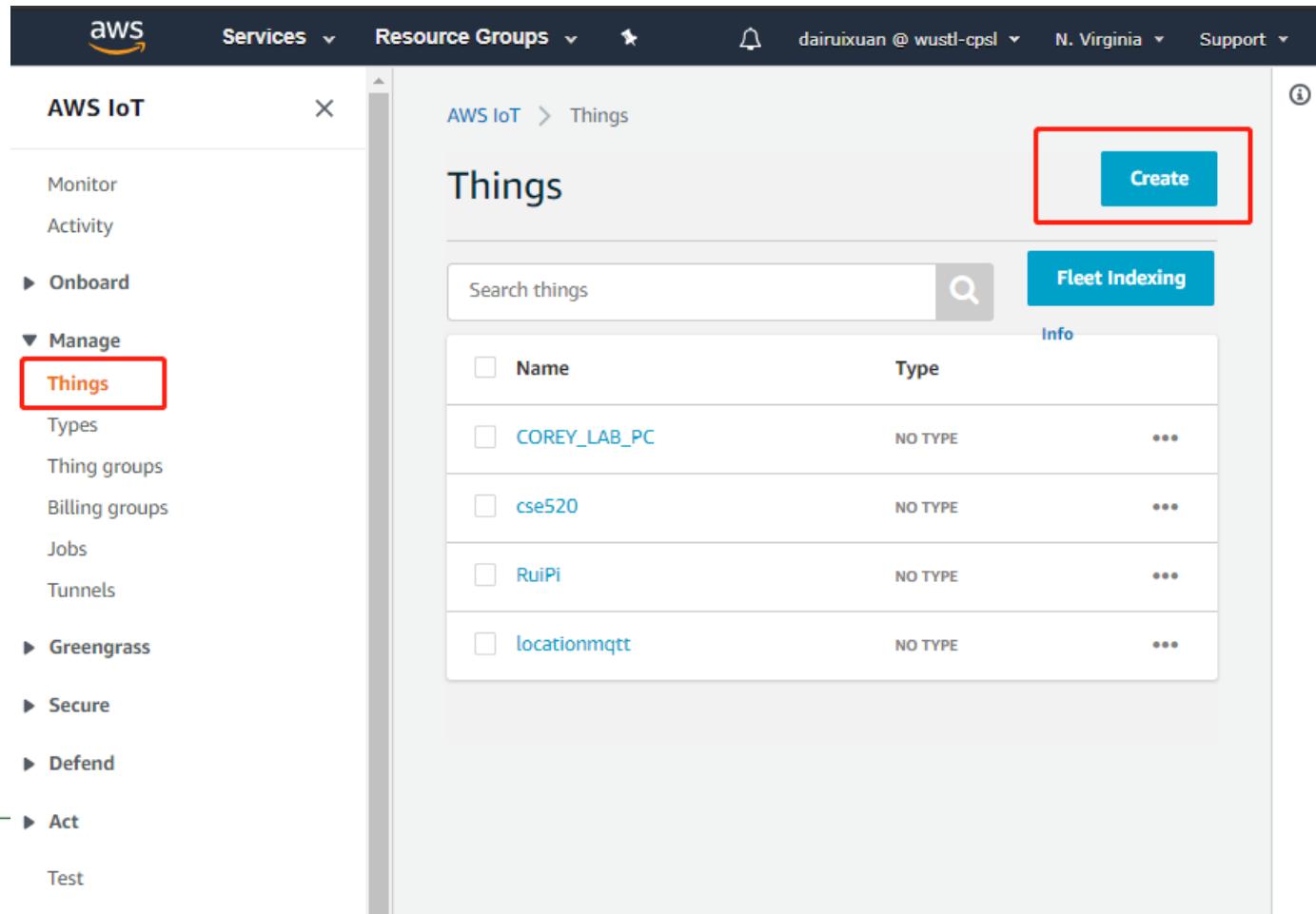
- **Certificates and policy**
  - **Authentication, Security**
  - **Permission and roles**

Certificates → your ID  
 Policy → your permission book



# Create a thing

- 1.AWS IoT Menu
  - Things → Create
- 2. Give a name



The screenshot shows the AWS IoT Things management interface. On the left, a sidebar menu is open under the 'Manage' section, with 'Things' selected and highlighted with an orange box. At the top right of the main content area, a large blue 'Create' button is also highlighted with a red box. The main table lists five existing things: COREY\_LAB\_PC, cse520, RuiPi, and locationmqtt, each with a checkbox next to its name and 'NO TYPE' listed under 'Type'. A search bar and a 'Fleet Indexing' button are visible above the table.

Name	Type
COREY_LAB_PC	NO TYPE
cse520	NO TYPE
RuiPi	NO TYPE
locationmqtt	NO TYPE

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

Name

cse521

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

Groups /

Create group Change

Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key

Provide an attribute key, e.g. Manufacturer

Value

Provide an attribute value, e.g. Acme-Corporation

Clear

Add another

Show thing shadow ▾

Cancel

Back

Next

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.

[!\[\]\(82ace3c1cdce20e5f8670b9f0a4207cd\_img.jpg\) 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 IoT.

[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	208f60eb4f.cert.pem	<a href="#">Download</a>
A public key	208f60eb4f.public.key	<a href="#">Download</a>
A private key	208f60eb4f.private.key	<a href="#">Download</a>

**Download all keys and root CA**

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

A root CA for AWS IoT [Download](#)

[Activate](#)

**Activate keys**

The keys and cert will be used later

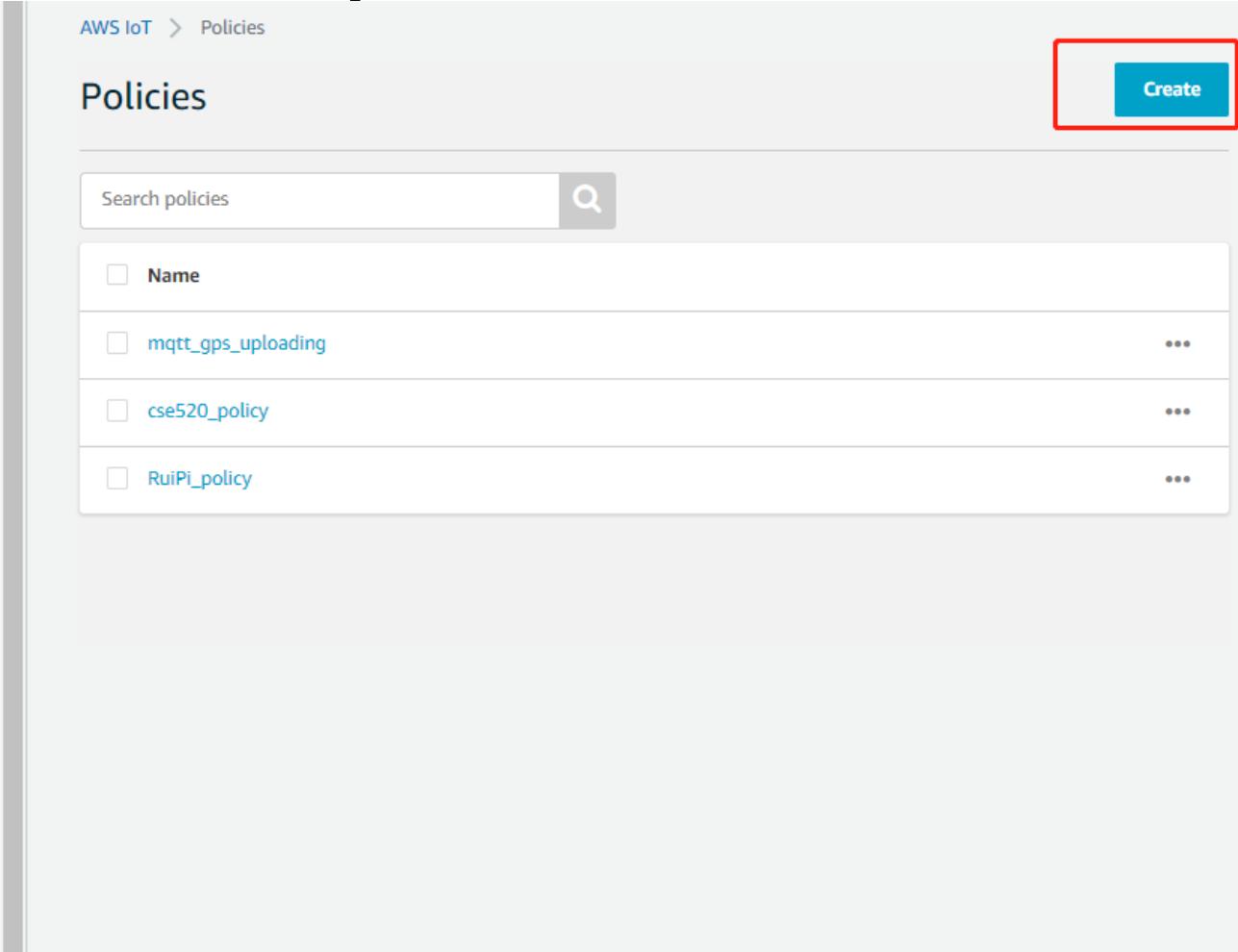
[Cancel](#)

[Done](#)

[Attach a policy](#)

# Create Policy

- A policy is attached to a key/cert
  - It tells what this key/cert can do



The screenshot shows the AWS IoT Policies interface. On the left, there's a navigation sidebar with sections like Activity, Onboard, Manage (Things, Types, Thing groups, Billing groups, Jobs, Tunnels), Greengrass, Secure (Certificates, Policies, CAs, Role Aliases, Authorizers), Defend, Act, and Test. The 'Policies' link in the Secure section is highlighted with a red box. The main area is titled 'Policies' and contains a search bar labeled 'Search policies'. Below the search bar is a table with four rows, each representing a policy: 'Name' (checkbox), 'mqtt\_gps\_uploading' (checkbox), 'cse520\_policy' (checkbox), and 'RuiPi\_policy' (checkbox). Each row has a '...' button on the right. A large red box highlights the 'Create' button in the top right corner of the main content area.

# 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

cse521\_policy

## Add statements

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

[Advanced mode](#)

Action

iot:\*

Here, we grant it all permissions for demo!

Resource ARN

\*

Effect

Allow  Deny

[Remove](#)

[Add statement](#)

[Create](#)

AWS IoT > Policies

## Policies

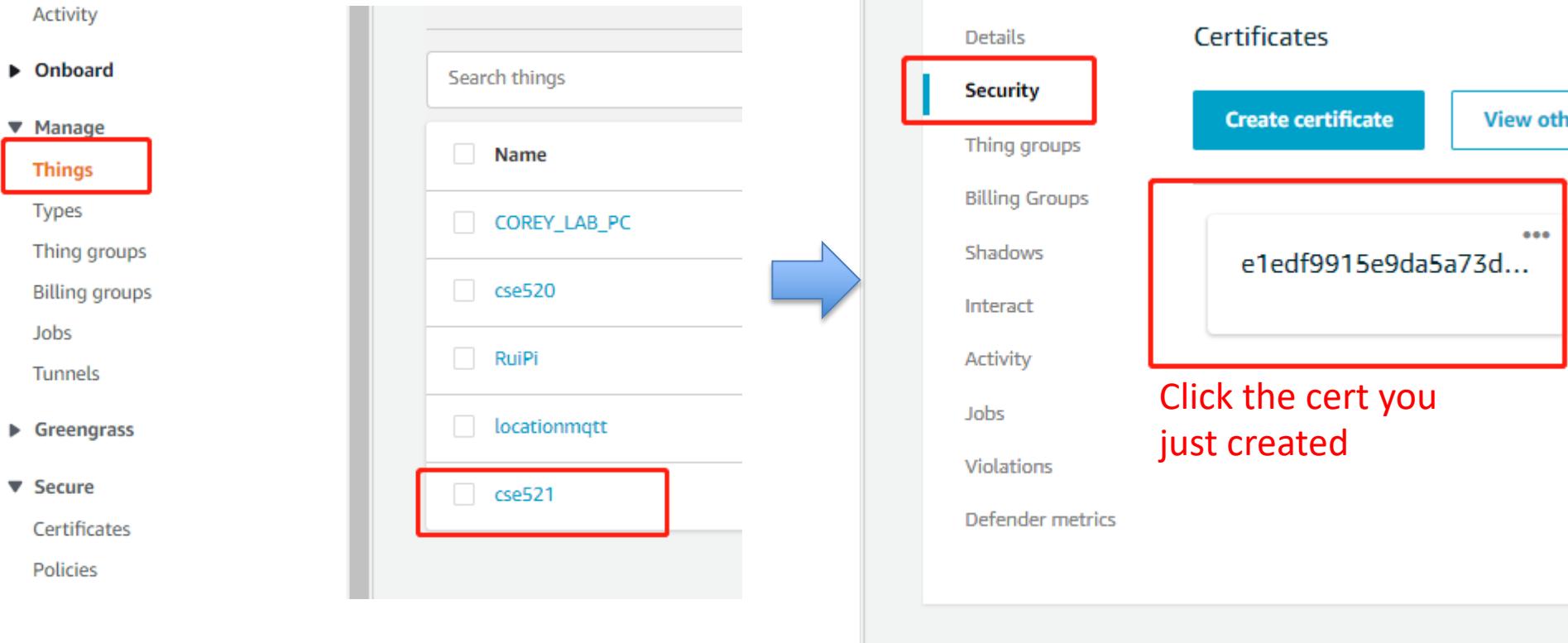
[Create](#)

Search policies  

<input type="checkbox"/> Name	
<input type="checkbox"/> mqtt_gps_uploading	...
<input type="checkbox"/> cse520_policy	...
<input type="checkbox"/> RuiPi_policy	This is the policy you created
<input type="checkbox"/> cse521_policy	...

# Attach Policy

- Attach Policy to the key/cert
- A policy tells what this key/cert can do



The screenshot shows the AWS IoT Core interface. On the left, the navigation menu includes 'Activity', 'Onboard', 'Manage' (with 'Things' selected), 'Greengrass', and 'Secure' (with 'Certificates' selected). The main area displays a list of things under 'Manage/Things', with 'cse521' highlighted by a red box. An arrow points from this list to the right pane. The right pane shows the 'Certificates' section, which includes a 'Create certificate' button and a list of certificates. One certificate, starting with 'e1edf9915e9da5a73d...', is also highlighted by a red box. The 'Security' tab is selected in the top navigation bar of the right pane.

Click the cert you just created

CSE\_521\_Temp > 208f60eb4fab1b02f5d6...

CERTIFICATE  
**208f60eb4fab1b02f5d656963382b05cd5c690b481e710c6a3e899a...**  
INACTIVE

**Actions** ▾

- Activate
- Deactivate
- Revoke
- Accept transfer
- Reject transfer
- Revoke transfer
- Start transfer
- Attach policy**
- Attach thing
- Download
- Delete

**Details**

**Certificate ARN**

A certificate Amazon Resource Name (ARN) uniquely identifies this certificate.

`arn:aws:iot:us-east-1:006025899016:cert/208f60eb4fa...`

**Details**

**Issuer**  
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US

**Subject**  
CN=AWS IoT Certificate

**Create date**  
Aug 25, 2019 12:56:24 PM -0500

**Effective date**  
Aug 25, 2019 12:54:24 PM -0500

**Expiration date**  
Dec 31, 2049 5:59:59 PM -0600

X

## Attach policies to certificate(s)

Policies will be attached to the following certificate(s):

e1edf9915e9da5a73da0f9385df408d76d28ec364eff1c3e9d4dc5c4b8b34e08

Choose one or more policies

<input type="checkbox"/> mqtt_gps_uploading	<a href="#">View</a>
<input type="checkbox"/> cse520_policy	<a href="#">View</a>
<input type="checkbox"/> RuiPi_policy	<a href="#">View</a>
<input checked="" type="checkbox"/> cse521_policy	<a href="#">View</a>

0 policies selected

[Cancel](#)

[Attach](#)

September 16, 2020, 11:11:39 (UTC-0400)

**Expiration date**

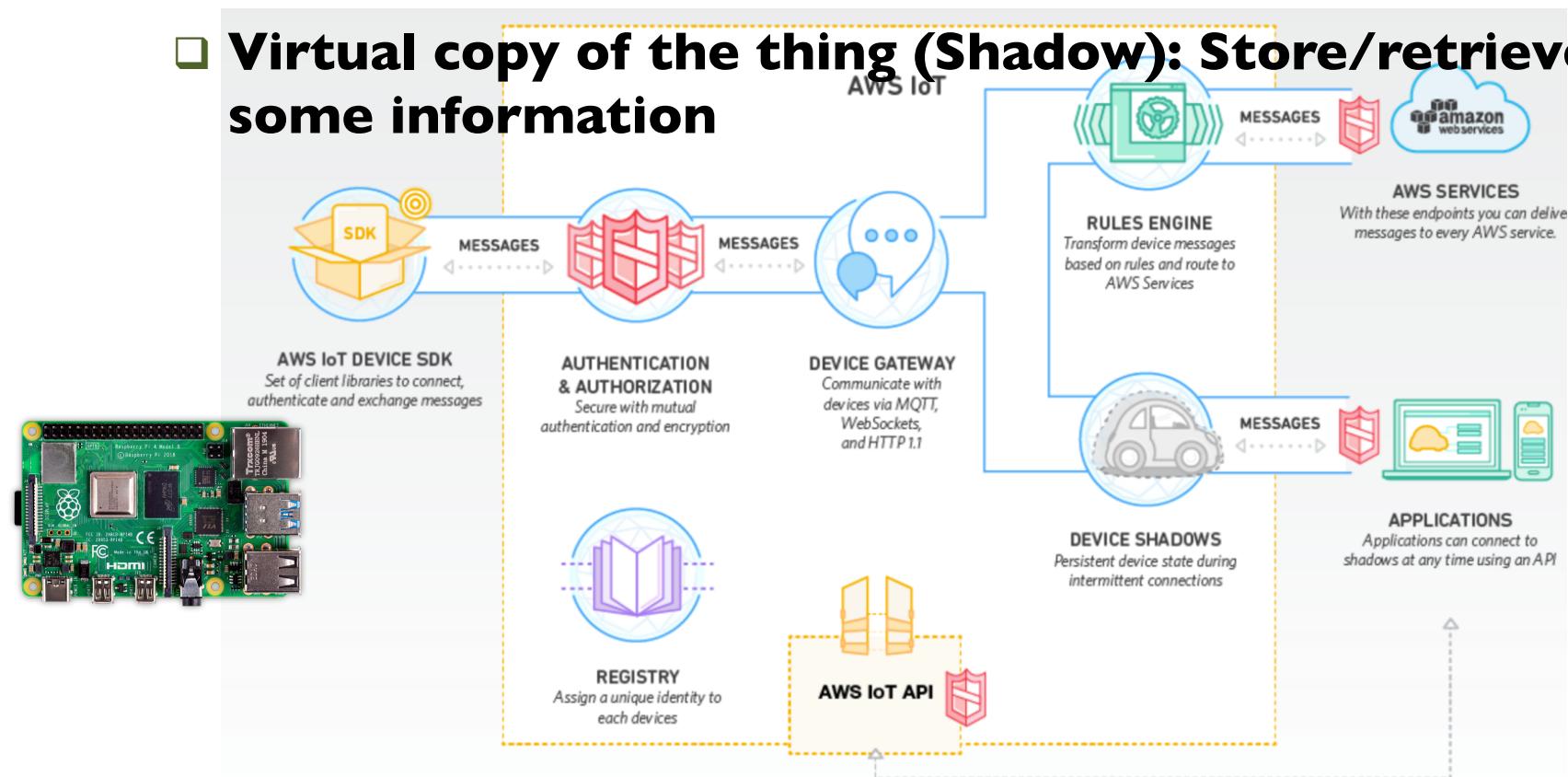
December 31, 2049, 18:59:59 (UTC-0500)

---

**Now, you have a virtual thing on AWS!**

# AWS Things Summary

- ❑ **Certificate: authenticate the device**
- ❑ **Policy: define the roles/permissions of the certificate**
- ❑ **Virtual copy of the thing (Shadow): Store/retrieve some information**



# Let's test it online!

---

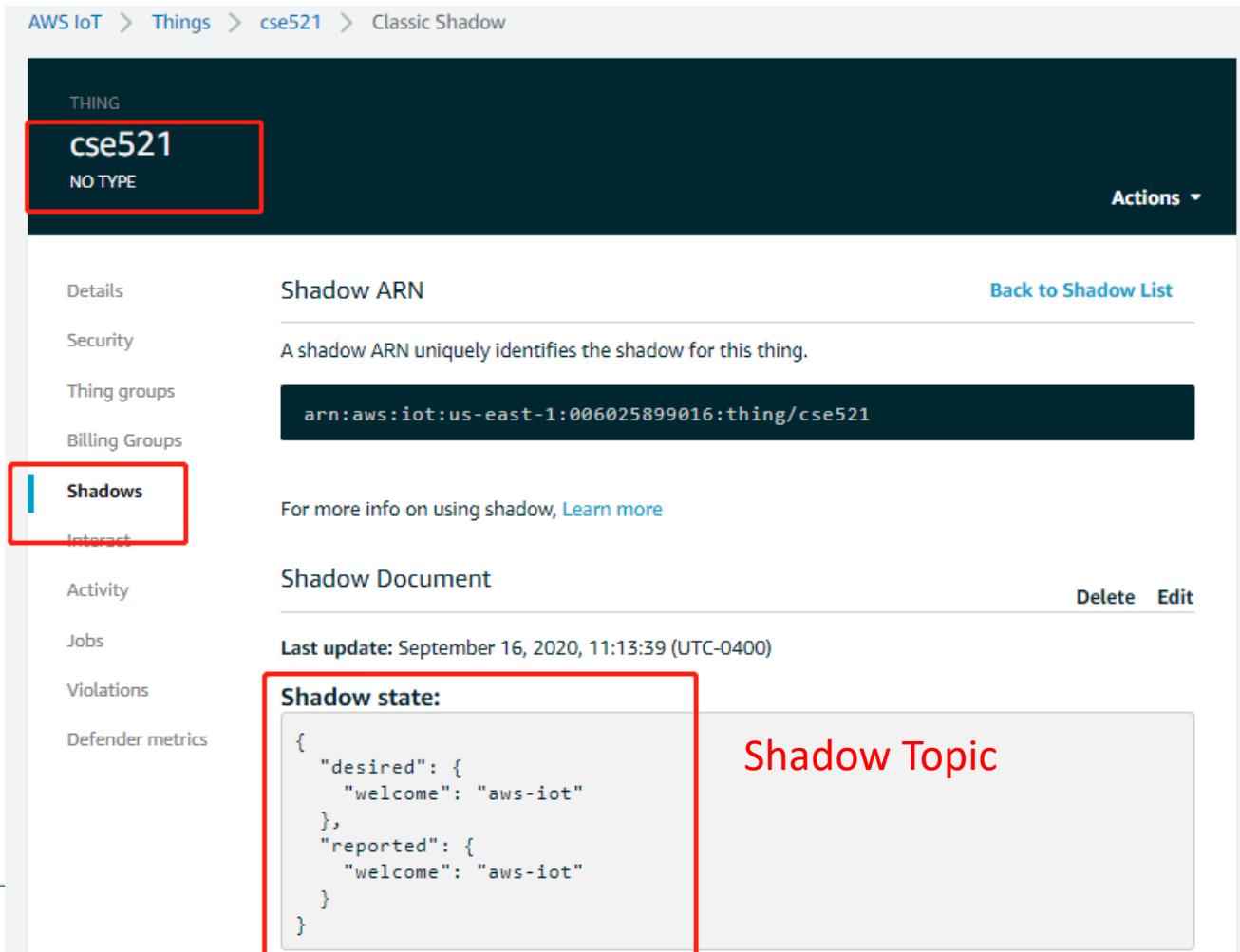


Washington University in St. Louis

# Basic Interact: Publish to the “Shadow”

- Get your “Shadow”
- In your Thing Page

Shadow is a special topic



AWS IoT > Things > cse521 > Classic Shadow

THING

cse521

NO TYPE

Actions ▾

Details      Shadow ARN      Back to Shadow List

Security

Thing groups

Billing Groups

**Shadows**

Interact

Activity

Jobs

Violations

Defender metrics

Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing.

arn:aws:iot:us-east-1:006025899016:thing/cse521

For more info on using shadow, [Learn more](#)

Shadow Document

Last update: September 16, 2020, 11:13:39 (UTC-0400)

Delete   Edit

**Shadow state:**

```
{
  "desired": {
    "welcome": "aws-iot"
  },
  "reported": {
    "welcome": "aws-iot"
  }
}
```

**Shadow Topic**

# Find your “Shadow” Topic

## ➤ Topic: can be seen as the “address”

### Device Shadow MQTT topics

[PDF](#) | [Kindle](#)

**Shadow Topic:**

\$aws/things/*cse521*/shadow/**update**

The Device Shadow service uses reserved MQTT topics to enable devices and apps to get, update, or delete the state information for a device (shadow).

Publishing and subscribing on shadow topics requires topic-based authorization. AWS IoT reserves the right to add new topics to the existing topic structure. For this reason, we recommend that you avoid wild card subscriptions to shadow topics. For example, avoid subscribing to topic filters like \$aws/things/thingName/shadow/# because the number of topics that match this topic filter might increase as AWS IoT introduces new shadow topics. For examples of the messages published on these topics see [Interacting with shadows](#).

Shadows can be named or unnamed (classic). The topics used by each differ only in the topic prefix. This table shows the topic prefix used by each shadow type.

<i>ShadowTopicPrefix</i> value	Shadow type
\$aws/things/ <i>thingName</i> /shadow	Unnamed (classic) shadow
\$aws/things/ <i>thingName</i> /shadow/name/ <i>shadowName</i>	Named shadow

To create a complete topic, select the *ShadowTopicPrefix* for the type of shadow to which you want to refer, replace *thingName*, and *shadowName* if applicable, with their corresponding values, and then append that with the topic stub as shown in the following sections.

**/update**

Publish a request state document to this topic to update the device's shadow:

*ShadowTopicPrefix*/update



Monitor

Activity

► Onboard

▼ Manage

- Things
- Types
- Thing groups
- Billing groups
- Jobs
- Tunnels

► Greengrass

► Secure

► Defend

► Act

**Test**

---

Software

Setting

Learn

Documentation

▶ Using Embedded **Test Client** to Test Your Shadow Topic

□ In AWS IoT Page

Subscriptions

Subscribe to a topic

Publish to a topic

\$aws/things/cse521/shadow... ×

Subscription topic

\$aws/things/cse521/shadow/update

Subscribe to topic

Max message capture Info

100

Quality of Service Info

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)

Display payloads as strings (more accurate)

Display raw payloads (in hexadecimal)

Publish

Specify a topic and a message to publish with a QoS of 0.

\$aws/things/cse521/shadow/update

Publish to topic

```

1 { "state": {
2   "reported": {
3     "time": "13:45",
4     "temperature": "25"
5   },
6   "message": "Hello from CSE521s console"
7 }
  
```

Topic Message

# Shadow Message

```
{ "state": {  
    "reported": {  
        "welcome": "CSE521s",  
        "time": "13:45",  
        "temperature": "25"  
    },  
    "message": "Hello from CSE521s console"  
}
```

A Shadow Message is a JSON object.

**Shadow message has strict formats.**

Please see

<https://docs.aws.amazon.com/iot/latest/developerguide/device-shadow-document-syntax.html>

# Update Shadow

## ➤ In your “Thing” Page

THING

cse521

NO TYPE

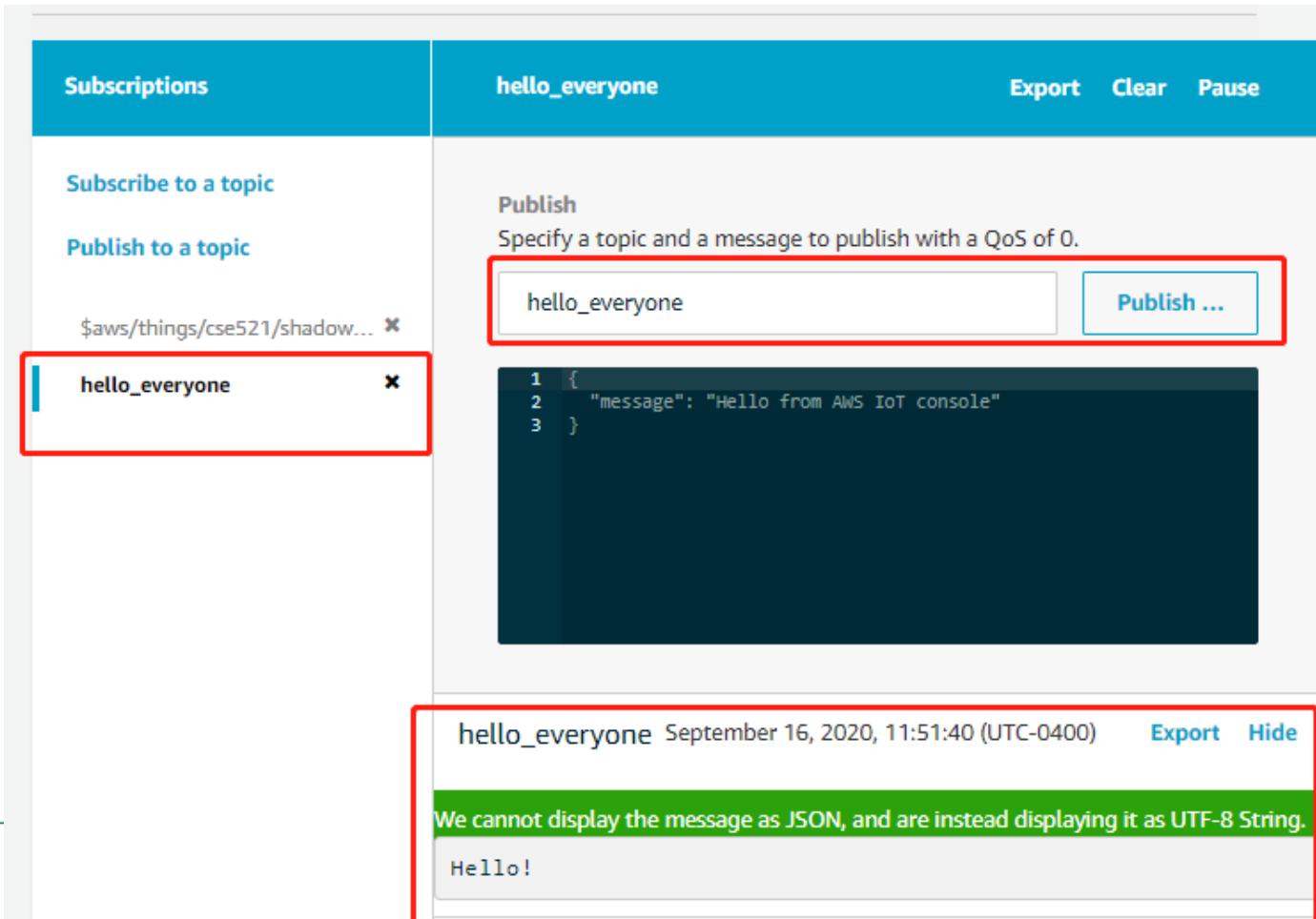
Actions ▾

---

<a href="#">Details</a> <a href="#">Security</a> <a href="#">Thing groups</a> <a href="#">Billing Groups</a> <b><a href="#">Shadows</a></b> <a href="#">Interact</a> <a href="#">Activity</a> <a href="#">Jobs</a> <a href="#">Violations</a> <a href="#">Defender metrics</a>	<p><b>Shadow ARN</b></p> <p>A shadow ARN uniquely identifies the shadow for this thing.</p> <pre>arn:aws:iot:us-east-1:006025899016:thing/cse521</pre> <p>For more info on using shadow, <a href="#">Learn more</a></p> <p><b>Shadow Document</b></p> <p>Last update: September 16, 2020, 11:49:57 (UTC-0400)</p> <p><b>Shadow state:</b></p> <pre>{   "desired": {     "welcome": "aws-iot"   },   "reported": {     "welcome": "CSE521s",     "time": "13:45",     "temperature": "25"   } }</pre> <p><b>Metadata:</b></p> <pre>{   "metadata": {   }</pre>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

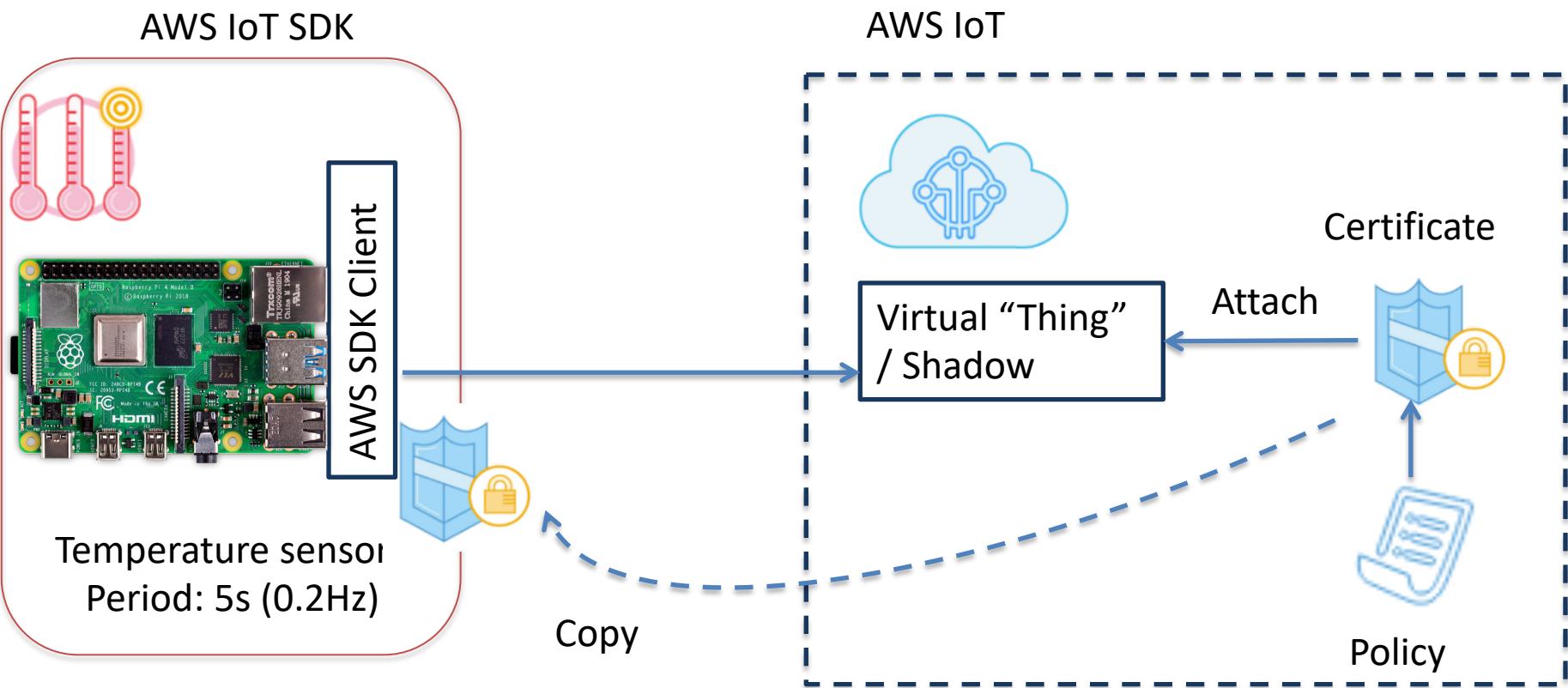
# Basic Interact: Subscribe/Publish

- You can define your own Topic
- Once you have a subscriber that is subscribed to the topic, the subscriber can receive the message



The screenshot shows the AWS IoT Subscriptions interface. On the left, under 'Subscriptions', there is a list with one item: 'hello\_everyone' (highlighted with a red box). In the center, the 'hello\_everyone' tab is active. The 'Publish' section contains a text input field with 'hello\_everyone' and a 'Publish ...' button (also highlighted with a red box). Below this, a code editor window shows a JSON message: { "message": "Hello from AWS IoT console" }. At the bottom, a message history shows a single entry: 'hello\_everyone September 16, 2020, 11:51:40 (UTC-0400)' followed by 'Hello!' (highlighted with a red box). A note at the bottom states: 'We cannot display the message as JSON, and are instead displaying it as UTF-8 String.'

# Step 2: Connect a “Physical” Device



# Connect your Device

## ➤ Copy certificates to your **physical things**

- Downloaded before!

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

A certificate for this thing	208f60eb4f.cert.pem	<a href="#">Download</a>
A public key	208f60eb4f.public.key	<a href="#">Download</a>
A private key	208f60eb4f.private.key	<a href="#">Download</a>

You also need to download a root CA for AWS IoT:  
 A root CA for AWS IoT [Download](#)

[Download all keys and root CA](#)

## ➤ Choose your **AWS SDK** (support MQTT)

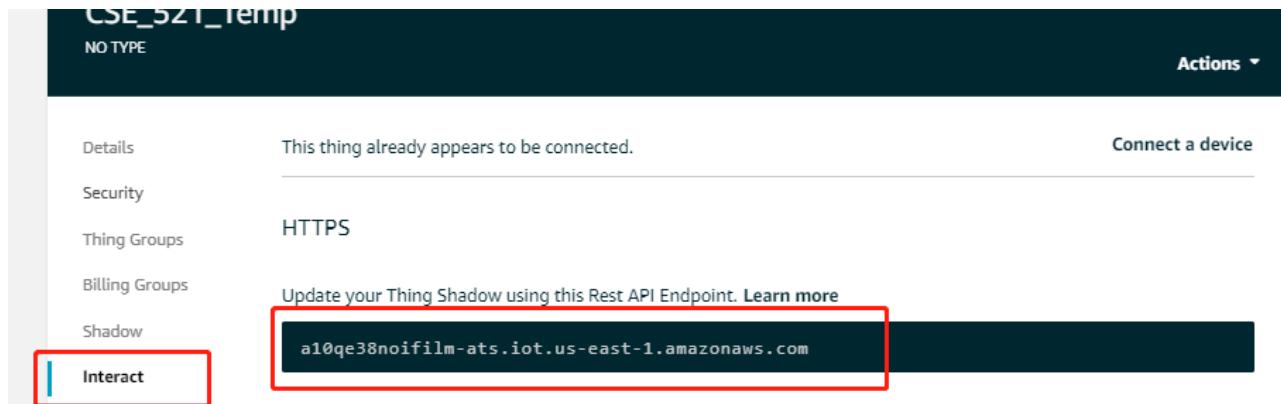
- Node JS
- Python (pip install AWSIoTPythonSDK)
- Java
- ...

## ➤ Set up your client with SDK and the certificates



# Some Notes

- 1. Copy the certificates/keys to your real thing
- 2. You will need the endpoint and port (8883)



Host(Endpoint)

- 3. Set up your configuration of the code with SDK

```
host = "a10qe38noifilm-ats.iot.us-east-1.amazonaws.com" # Your thing's endpoint. See tutorial slides
rootCAPath = "root-CA.crt"
certificatePath = "e1edf9915e-certificate.pem.crt"
privateKeyPath = "e1edf9915e-private.pem.key"
port = 8883
clientId = "CSE521"
topic = "$aws/things/cse521/shadow/update" # Shadow topic of your Thing
```

Change your code accordingly!

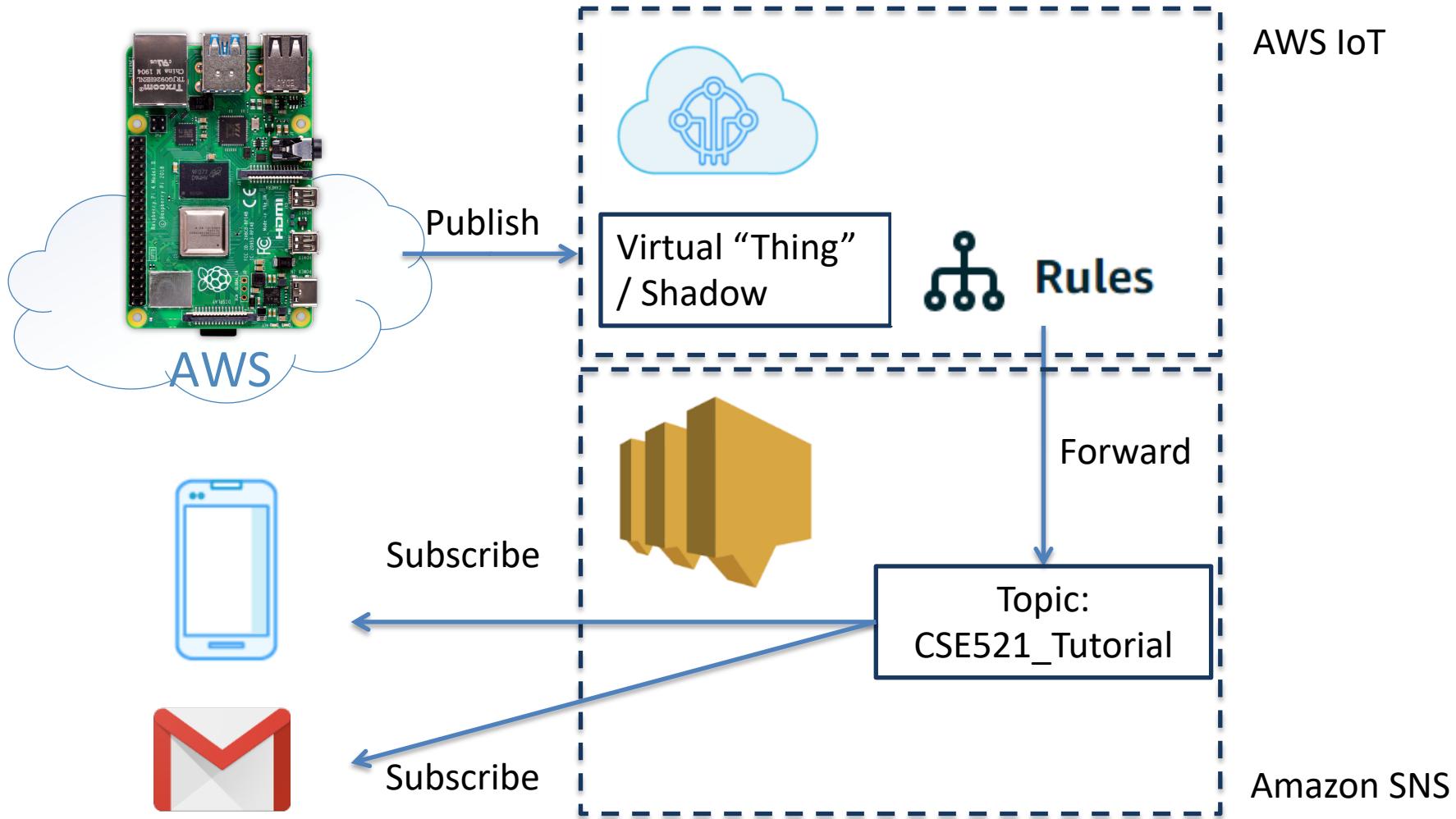
# SDK and Demo Codes

---

- <https://github.com/aws/aws-iot-device-sdk-python>
- <https://github.com/aws/aws-iot-device-sdk-python/blob/master/samples/basicPubSub/basicPubSub.py>

# More: Rule Engine, Link with SNS services

## ➤ Simple Notification Service



# Create a Rule in Amazon IoT

## ➤ Add a query to filter your interesting topic (event)

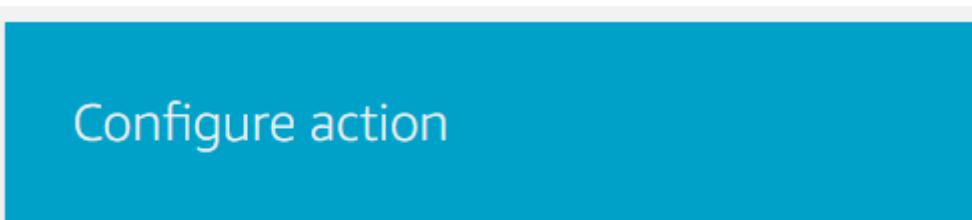
### Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

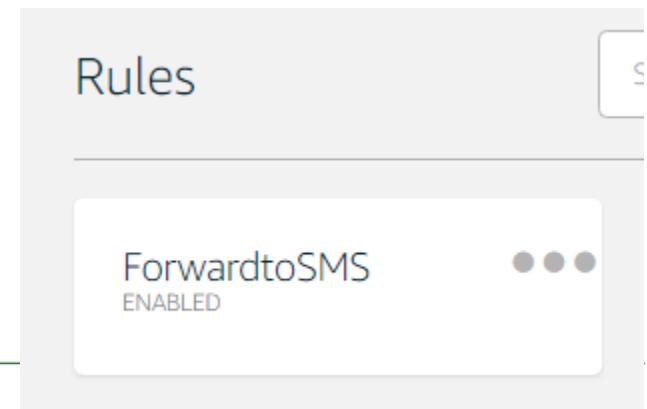
```
1 SELECT * FROM '$aws/things/cse521/shadow/update'
```

## ➤ Add an Action:

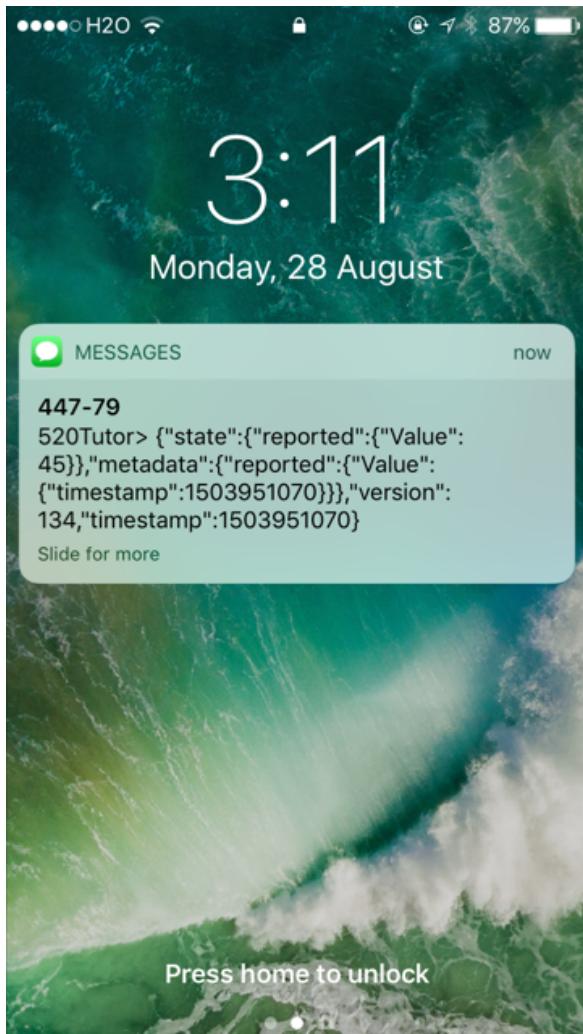
- Forward this message to SNS
- Specify Dest ARN
- Enable Rule



Send a message as an SNS push notification



# Notification on SMS & Email



AWS Notification Message Inbox x

 **520Tutor** no-reply@sns.amazonaws.com 3:11 PM (28 minutes ago) ☆ ↻ ▾  
to me ▼

{"state":{"reported":{"Value":45}}, "metadata":{"reported":{"Value":{"timestamp":1503951070}}}, "version":134, "timestamp":1503951070}

--  
If you wish to stop receiving notifications from this topic, please click or visit the [link](https://sns.us-west-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-west-2:401317363811:CSE520S_Tutorial:00c54352-7d1a-4c09-9cc1-15aed3c395e3&Endpoint=naroahlee@gmail.com) below to unsubscribe:  
[https://sns.us-west-2.amazonaws.com/unsubscribe.html?  
SubscriptionArn=arn:aws:sns:us-west-2:401317363811:  
CSE520S\\_Tutorial:00c54352-7d1a-4c09-9cc1-15aed3c395e3&  
Endpoint=naroahlee@gmail.com](https://sns.us-west-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-west-2:401317363811:CSE520S_Tutorial:00c54352-7d1a-4c09-9cc1-15aed3c395e3&Endpoint=naroahlee@gmail.com)

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at  
<https://aws.amazon.com/support>

# One More Thing: Account Security

➤ DON'T UPLOAD YOUR KEY PUBLICLY!!!

Time to Open Source!



# What if... \$50,000 AWS Bill!

Quora

Ask or Search Quora

Ask Question

Fraud

Amazon Web Services

Amazon.com (product)

Hackers

+3



## My AWS account was hacked and I have a \$50,000 bill, how can I reduce the amount I need to pay?

For years, my bill was never above \$350/month on my single AWS instance. Then over the weekend someone got hold of my private key and launched hundreds of instances and racked up a \$50,000 bill before I found out about it on Tuesday. Amazon had sent a warning by email at \$15,000 saying they had found **our key posted publicly**, but I didn't see it. Naturally, this is a devastating amount of money to pay. I'm not saying I shouldn't pay anything, but this just a crazy amount in context. Amazon knew the account was compromised, that is why they sent an email, they knew the account history and I had only spent \$213 the previous month. I almost feel they deliberately let it ride to try to earn more money. Does anyone have any experience with this sort of problem?

# Some project examples

- Gesture recognition with smartwatch
  - ❑ Recognize the specific gesture to control the light
- Smart pet feeder
  - ❑ Food dispenser with schedules and smart control
- Smart mirror
  - ❑ Show personalized info in the mirror

Sensing, Connecting, Smart

If you have any question about the project,  
feel free to send me an Email

# Thanks!

---

Ruixuan Dai



Washington University in St. Louis