Cloud Tutorial: AWS IoT

CSE 521S Fall
Sep. 17, 2020
Ruixuan (Corey) Dai
XaaS: Basics in Cloud Computing
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

APP
Data
Runtime
Middleware
OS
Virtualization
Server
Storage
Network

You Manage

9/17/20
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 you’re smart A/C controller
PaaS Example: AWS IoT

The essence is **MESSAGING MIDDLEWARE**

Send messages between **sensors** and **servers**...
Software as a Service (SaaS)

- **SaaS**
  - You get a whole solution

- **Example**
  - Gmail
  - Dropbox
  - Office365
XaaS: A Recap

On-Premise
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

IaaS
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

You Manage

PaaS
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

Service Provider Manages

SaaS
- APP
- Data
- Runtime
- Middleware
- OS
- Virtualization
- Server
- Storage
- Network

Service Provider Manages

On-Premise
IaaS
PaaS
SaaS

9/17/20
Tutorial: Hello! AWS IoT!!
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

Source: https://aws.amazon.com/iot-platform/
http://www.brain-smart.net/smart-brain-health-blog/page/2/#axzz4W4oSp8a6
Solution: AWS IoT

United: Connect + Communication
Stated: “Thing Shadow”

Smart: Other Cloud Service
Data Storage
Machine Learning

Source: https://aws.amazon.com/iot-platform/
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.

---

- **Temperature sensor**
  - Period: 5s (0.2Hz)

**Diagram:**
- **AWS IoT**
  - Publish to a topic
  - Forward
  - Subscribe to a topic

**Message middleware:**
- **Amazon SNS**

---

Source: https://aws.amazon.com/iot-platform/
Basic Interact: Subscribe/Publish

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

I am going to publish a tweet. “Today is beautiful”

Received message from Trump: “Today is beautiful”

Subscriber

Publisher

Follow “subscribe”

Tweet “publish”

Twitter account a “Topic”
Resources

- Amazon IoT
  - http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

- Raspberry Pi

- Resource list for course 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/
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

- 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
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 /
```

**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**

**Next**
CREATE A THING

Add a certificate for your thing

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

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

Skip certificate and create thing
You will need to add a certificate to your thing later before your device can connect to AWS IoT.
The keys and cert will be used later.

Download all keys and root CA

Activate keys

<table>
<thead>
<tr>
<th>Description</th>
<th>File Name</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>A certificate for this thing</td>
<td>208f60eb4f.cert.pem</td>
<td>Download</td>
</tr>
<tr>
<td>A public key</td>
<td>208f60eb4f.public.key</td>
<td>Download</td>
</tr>
<tr>
<td>A private key</td>
<td>208f60eb4f.private.key</td>
<td>Download</td>
</tr>
</tbody>
</table>

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

Download

Done

Attach a policy
Create Policy

- A policy is attached to a key/cert
  - It tells what this key/cert can do
Here, we grant it all permissions for demo!
This is the policy you created
Attach Policy

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

Click the cert you just created
CERTIFICATE

208f60eb4fab1b02f5d656963382b05cd5c690b481e710c6a3e899a...

INACTIVE

Details

Certificate ARN

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

arn:aws:iot:us-east-1:006025899016:cert/208f60eb4fab1b02f5d656963382b05cd5c690b481e710c6a3e899a...

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

Actions

Activate
Deactivate
Revoke
Accept transfer
Reject transfer
Revoke transfer
Start transfer
Attach policy
Attach thing
Download
Delete
Attach policies to certificate(s)

Policies will be attached to the following certificate(s):
e1edf9915e9da5a73da0f9385df408d76d28ec364eff1c3e9d4dc5c4b8b34e08

Choose one or more policies

Search policies

- mqtt_gps_uploading
- cse520_policy
- RuiPi_policy
- cse521_policy

0 policies selected  Cancel  Attach
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!
Basic Interact: Publish to the “Shadow”

- Get your “Shadow”
  - In your Thing Page

![AWS IoT Things cse521 Classic Shadow]

**Shadow ARN**
A shadow ARN uniquely identifies the shadow for this thing.

```
arn:aws:iot:us-east-1:00602589016:thing/cse521
```

For more info on using shadow, Learn more

**Shadow Document**

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

**Shadow state:**

```json
{
  "desired": {
    "welcome": "aws-iot"
  },
  "reported": {
    "welcome": "aws-iot"
  }
}
```
Find your “Shadow” Topic

Topic: can be seen as the “address”

Device Shadow MQTT topics

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.

<table>
<thead>
<tr>
<th>ShadowTopicPrefix value</th>
<th>Shadow type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$aws/things/thingName/shadow</td>
<td>Unnamed (classic) shadow</td>
</tr>
<tr>
<td>$aws/things/thingName/shadow/name/shadowName</td>
<td>Named shadow</td>
</tr>
</tbody>
</table>

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

https://docs.aws.amazon.com/iot/latest/developerguide/device-shadow-mqtt.html?icmpid=docs_iot_console
Using Embedded **Test Client** to Test

- In AWS IoT Page
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](https://docs.aws.amazon.com/iot/latest/developerguide/device-shadow-document-syntax.html)
Update Shadow

➤ In your “Thing” Page

THING
cse521
NO TYPE

Details
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-1:00602589016:thing/cse521

For more info on using shadow, Learn more

Shadow Document

Last update: September 16, 2020, 11:49:57 (UTC-0400)

Shadow state:

```
{
  "desired": {
    "welcome": "aws-iot"
  },
  "reported": {
    "welcome": "cse521s",
    "time": "13:45",
    "temperature": "25"
  }
}
```

Metadata:

```
{
  "metadata": {
  
  }
}
```
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

You can subscribe to a topic named `hello_everyone`. After subscribing, you can publish a message to this topic. The message published is: "Hello from AWS IoT console."
Step 2: Connect a “Physical” Device

- AWS IoT SDK
- AWS SDK Client
- Temperature sensor
  Period: 5s (0.2Hz)

AWS IoT

- Virtual “Thing” / Shadow
- Attach
- Certificate
- Policy

Copy
Connect your Device

- Copy certificates to your **physical things**
  - Downloaded before!

- 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)

3. Set up your configuration of the code with SDK

```python
host = "a10qe38noifilm-ats.iot.us-east-1.amazonaws.com"  # Your thing's endpoint. See tutorial slides
rootCAPath = "root-CA.crt"
certificatePath = "e1edf915e-certificate.pem.crt"
privateKeyPath = "e1edf915e-private.pem.key"
port = 8883
clientId = "CSE521"
topic = "$aws/things/cse521/shadow/update"  # Shadow topic of your Thing
```
SDK and Demo Codes

More: Rule Engine, Link with SNS services

- Simple Notification Service

AWS

Virtual "Thing" / Shadow

Rules

Forward

Topic: CSE521_Tutorial

Subscribe

AWS IoT

Publish

Amazon SNS

Subscribe

Ec2 t2.micro
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.

- Add an Action:
  - Forward this message to SNS
  - Specify Dest ARN
  - Enable Rule
Notification on SMS & Email

AWS Notification Message

520Tutor no-reply@sns.amazonaws.com 3:11 PM (28 minutes ago)

"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 below to unsubscribe:
https://sns.us-west-2.amazonaws.com/unsubscribe.html?
SubscriptionArn=arn:aws:sns:us-west-2:401317363811:
CSE520S_Tutorial:00c54352-7d1a-4c09-9cc1-15aedd3c395e3&
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!

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