

---

# IoT Security and Privacy

## Introduction to Amazon AWS IoT

---

YIER JIN

UNIVERSITY OF FLORIDA

EMAIL: [YIER.JIN@ECE.UFL.EDU](mailto:YIER.JIN@ECE.UFL.EDU)

SLIDES ARE ADAPTED FROM PROF. XINWEN FU @ UCF/UMASS

# Learning Outcomes

Upon completion of this unit:

- Students will be able to understand the architecture of Amazon AWS IoT
- Students will be able to practice the use of AWS IoT managing IoT devices
- Students will be able to practice programming AWS IoT

# Prerequisites and Module Time

## Prerequisites

- Students should have taken classes on operating system and computer architecture.
- Students must have taken crypto and know how public key crypto and symmetric key crypto work.
- Students should have mastered programming Raspberry Pi.
- Students should know basic concepts of networking.

## Module time

- Two-hour lecture
- Two-hour homework

# Outline

Introduction

Device registry - thing, keys, certificate, policy

Security and identity

Device gateway – MQTT

Rules engine

Pricing

Example code with MQTT

# Overview

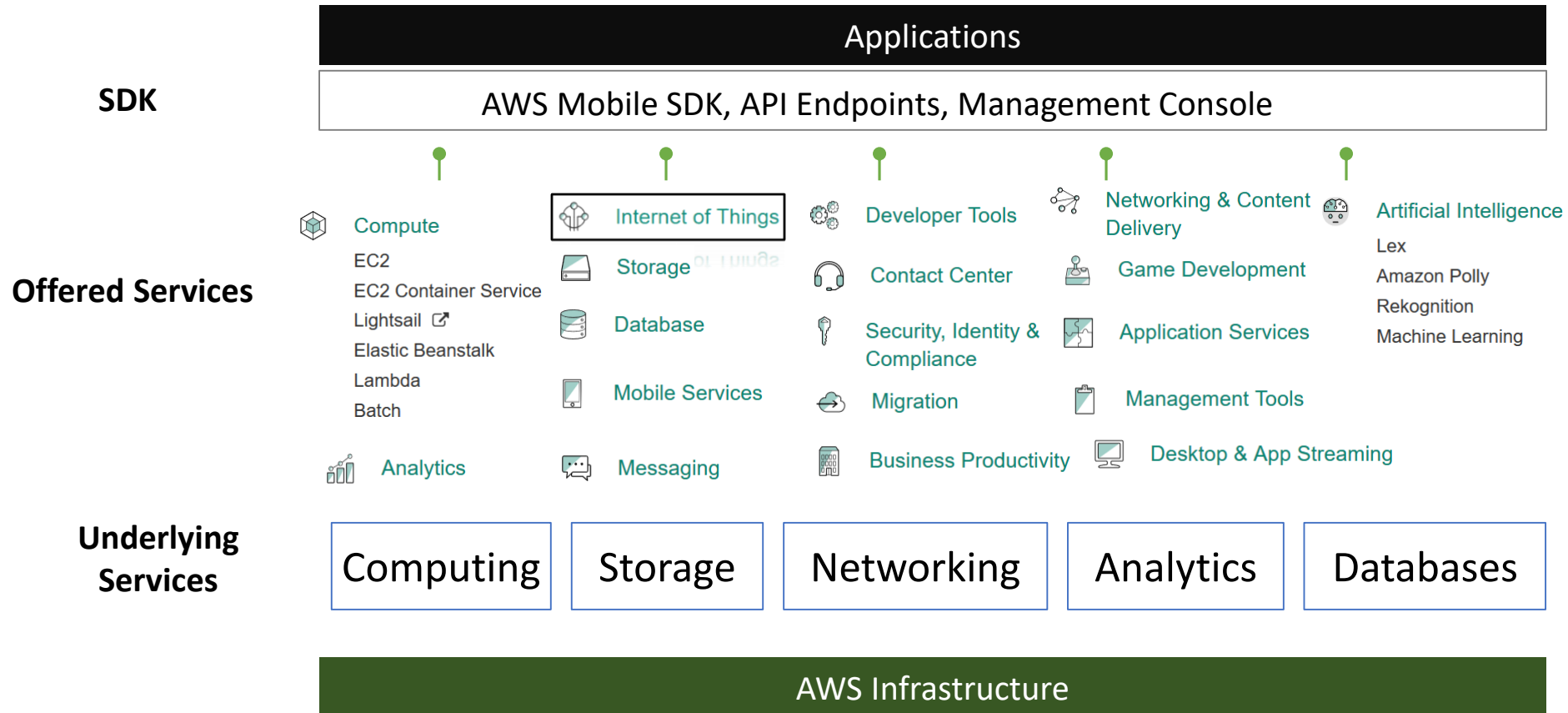
Amazon AWS IoT basically sets up a server such as a MQTT server so that physical IoT devices and applications can use the server to communicate with each other

AWS IoT goes beyond the communication through MQTT and provides other Amazon services that process the data from IoT devices, for example, storing data via Amazon Simple Storage Service (S3).

AWS IoT supports other communication protocol such as REST API (https)

Protocol	Authentication	Port
MQTT	Client Certificate	8883
MQTT over WebSocket	<a href="#">AWS Signature Version 4</a>	443
HTTP	Client Certificate	8443
HTTP	<a href="#">AWS Signature Version 4</a>	443

# Current AWS Services



# AWS IoT service

Beta out in August 2015

Use of standard protocols

SDK, APIs

Partnership with different industry sectors

Bridge to other AWS Services, such as email, SMS, data analytics

Bi-Directional / Long lived connections

## Smart Transportation

## Smart Health

## Smart Agriculture

### AWS IoT Architecture & Ecosystem



*Solutions*

API

Thing  
Shadows

Rules  
Engine

Broker

C / Java  
SDK

Things  
Registry

Secure  
Gateway

*AWS IoT*

Regions & Availability Zones

Security



*IoT Devices*



# AWS IoT - Console Interactive Tutorial

The screenshot displays the AWS IoT console interface during an interactive tutorial. The browser address bar shows the URL: `https://console.aws.amazon.com/iot/home?region=us-east-1#/tutorial/help?step=5`. The page header includes the AWS IoT logo and navigation links for Resources, Tutorial, and Help and details. The main content area is titled "Learn how AWS IoT works with this interactive tutorial" and features a progress bar with six steps: 1. Device Gateway, 2. Rules Engine, 3. Rule Actions, 4. Device Shadows, 5. Build Solutions (current step), and 6. Done.

The diagram illustrates a workflow where a mobile application (Acme Soft) interacts with a light bulb (Device) via a cloud-based Rules Engine. The Rules Engine contains two rules: "if R, forward R" and "if B, transform to G". The mobile app sends a message to the light bulb, which then sends a message to the Rules Engine. The Rules Engine processes the message and sends a response back to the mobile app. The light bulb is represented by a green box with a red light bulb icon and a green box with a blue light bulb icon. The mobile app is represented by a green box with a red light bulb icon and a green box with a blue light bulb icon. The Rules Engine is represented by a blue box with a gear icon and a green box with a blue light bulb icon. The light bulb is labeled "1" and the mobile app is labeled "2".

**Read and Set Device State with Shadows**

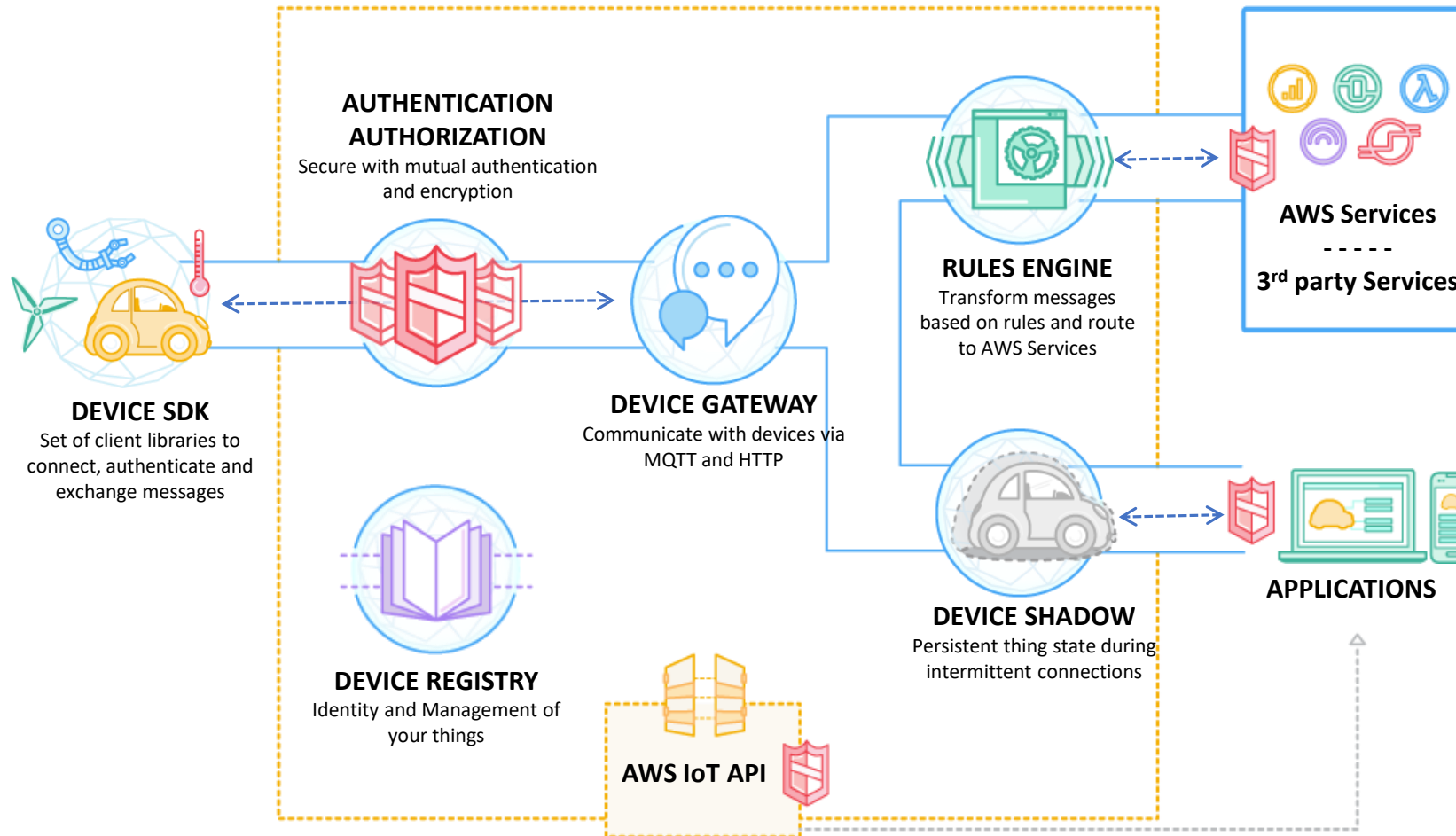
AWS IoT makes it easy to build companion applications that interact with your connected Things.

The example at the left shows a mobile application that reflects the color of your light bulb. The mobile app never communicates directly to the light bulb. Rather, the mobile app uses a REST API to read and set the state of the bulb's Device Shadow.

Try interacting with the bulb via this mobile app.

[Next](#)

# AWS IoT



# AWS IoT Components

## Message broker

- A secure relay between users (subscribers and publishers)
- Protocols: MQTT, HTTP REST interface

## Rules engine

- Rules directing data to other AWS services such as Amazon S3, Amazon DynamoDB, and AWS Lambda

## Thing Registry (Device Registry)

- Virtual devices in the cloud, corresponding to physical things
- Up to three custom attributes for a thing.
- Association of certificates and MQTT client IDs with a thing

# AWS IoT Components (Cont'd)

## Thing Shadows service

- **Synchronization** of states requested by users and at the physical devices (what if the connection is down?)

## Thing shadow

- A JSON document storing state information for a thing

## Device gateway

- Entry point for physical devices into the cloud

## Security and identity service

- Secure communication
- Secure storage of credentials
- Identification, authentication and authorization

# Accessing AWS IoT

## AWS Command Line Interface (AWS CLI)

- Windows, Mac, and Linux
- Refer to the [AWS Command Line Interface User Guide](#).

## AWS SDKs

- Build your IoT applications using language-specific APIs.
- Refer to [AWS SDKs and Tools](#).

## AWS IoT API

- Libraries
- Refer to [Actions in the AWS IoT API Reference](#).

## AWS IoT Thing SDK for C

- For resource-constrained things, such as microcontrollers.

# Closely Related AWS Services

## Amazon Simple Storage Service (S3)

- Scalable storage Refer to [Amazon S3](#).

## Amazon DynamoDB

- NoSQL databases. Refer to [Amazon DynamoDB](#).

## Amazon Kinesis

- Real-time processing of streaming data. Refer to [Amazon Kinesis](#).

## AWS Lambda

- Custom code running on Amazon EC2. Refer to [AWS Lambda](#).

## Amazon Simple Notification Service (SNS)

- Notifications through email, SMS and others. Refer to [Amazon SNS](#).

# Outline

Introduction

Device registry - thing, keys, certificate, policy

Security and identity

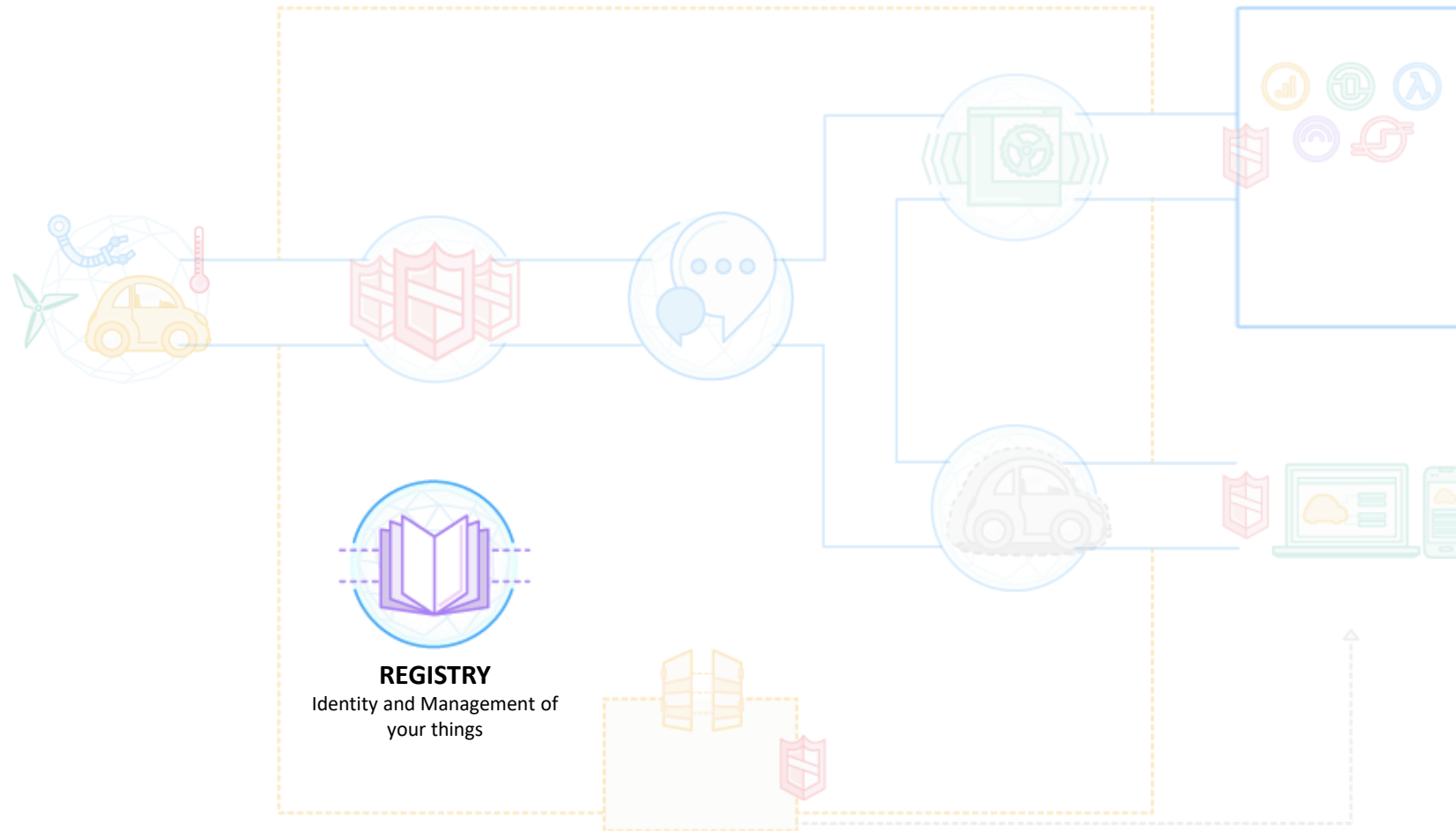
Device gateway – MQTT

Rules Engine

Pricing

Example code with MQTT

# AWS IoT Device Registry





# Get Started with AWS IoT and Raspberry Pi

#1 Sign into AWS Management console from IoT Portal

#2 Create a Raspberry Pi Thing

- A thing represents a physical device in AWS IoT cloud

#3 Create, download and activate Certificate and keys

- A certificate is used to authenticate a physical device with AWS IoT

#4 Create a policy

- A policy specifies what a physical device can do, such as subscribing or publishing to MQTT topics

#5 Attach the thing and the policy with the certificate


- Means the physical device (represented by the certificate) is not associated with the thing in AWS IoT and what the physical device can do

#6 Create a rule (optional)


#7 Connect Raspberry Pi to AWS IoT

<http://www.awsomeblog.com/amazon-web-services-iot/>

# AWS IoT Device Registry - Example 1


AWS
Services
Edit

Xinwen Fu
Oregon
Support


AWS IoT

[Resources](#)
[MQTT Client](#)
[Tutorial](#)
[Settings](#)
1 notification

## Resources

+ Create a resource

Filter by resource names or by resource type (below)

All

1/1 things

1/1 rules

1/1 certificates

1/1 policies

Select all

Actions

First



Previous

1



Next

Last

IoT-motion-sensor






IoT-motion-sensor-Policy



a5aedfc0489  
1fe67afc108f  
b01fb81d2e8

ACTIVE

save

ENABLED

[Learn more](#)
[Detail](#)
[Update shadow](#)
[Edit](#)
×

Name

IoT-motion-sensor


REST API endpoint

https://A3VOQMFBV77HZI.iot.us-west-2.amazonaws.com/things/iot-motion-sensor/shadow

MQTT topic

'\$aws/things/iot-motion-sensor/shadow/update'

Last update

15 days ago 

Attributes

Receiving-Date: Jan-11-2016  
Product-Name: HC-SR501-Infrared-PIR-Motion-Sensor  
Seller: Great-Deal

Linked certificates

[Show all](#)

Shadow status

In sync

Shadow version

140

Shadow state

Create a rule

Connect a device

# AWS IoT Device Registry - Example 2

AWS

Services

Edit

Xinwen Fu

N. Virginia

Support

AWS IoT

Resources

MQTT Client

Tutorial

Settings

1 notification

Resources

Create a resource

Filter by resource names or by resource type (below)

All

1/1 things

1/1 rules

1/1 certificates

1/1 policies

Select all

Actions

First

Previous

1

Next

Last

Fu-MotionSensor

Fu-MotionSensor-Policy

881038f8864646243f52790a92cec5c4b

ACTIVE

FuSNS

ENABLED

Learn more

Detail

Update shadow

Edit

Name

Fu-MotionSensor

REST API endpoint

https://A3VOQMFBV77HZI.iot.us-east-1.amazonaws.com/things/Fu-MotionSensor/shadow

MQTT topic

'\$aws/things/Fu-MotionSensor/shadow/update'

Last update

6 hours ago

Attributes

None

Linked certificates

Show all

Shadow status

In sync

Shadow version

62

Shadow state

```

1 {
2   "reported": {
3     "color": "red",
4     "time": "2016/03/08 15:52:04"

```

Create a rule

Connect a device

# Outline

Introduction

Device registry - thing, keys, certificate, policy

**Security and identity**

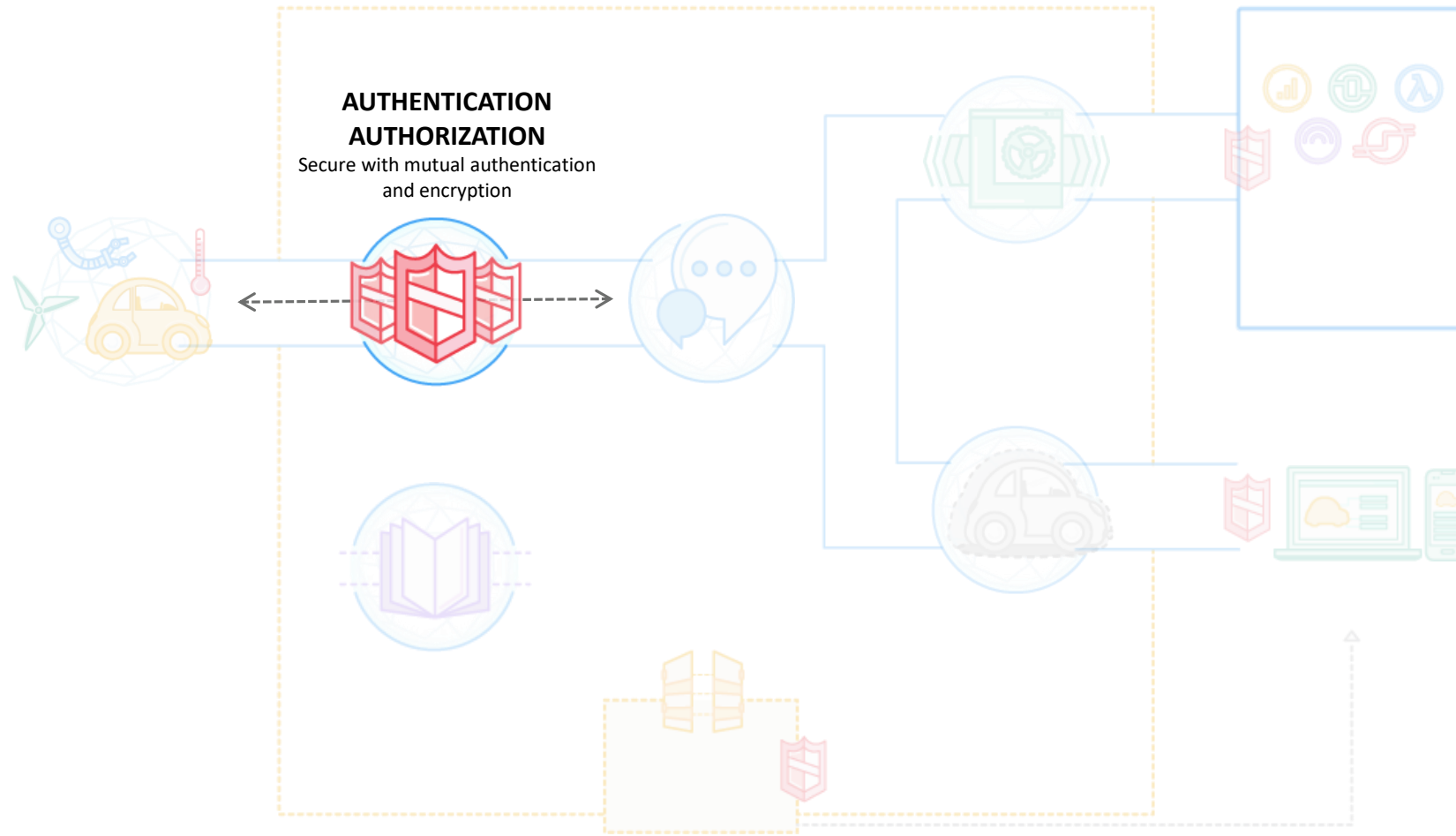
Device gateway – MQTT

Rules Engine

Pricing

Example code with MQTT

# AWS IoT Security



# Securing and Identifying Things: Mutual Authentication through TLS

## **Server authentication**

- Server sends its certificate.
- Then?

## **Client authentication, similar to ssh certificate based authentication**

- Server stores a client's certificate for later identification
- Server performs the challenge response protocol to verify that the client has the private key

# Security, Designed for Connected Devices

	MQTT + Mutual Auth TLS	AWS Auth + HTTPS
Server Auth	TLS + Cert	TLS + Cert
Client Auth	TLS + Cert	AWS Access Keys
Confidentiality	TLS	TLS
Protocol	MQTT	HTTP
Identification	AWS ARNs	AWS ARNs
Authorization	AWS Policy	AWS Policy

Amazon Resource Names (ARNs)

# Outline

Introduction

Device registry - thing, keys, certificate, policy

Security and identity

Device gateway – MQTT

Rules Engine

Pricing

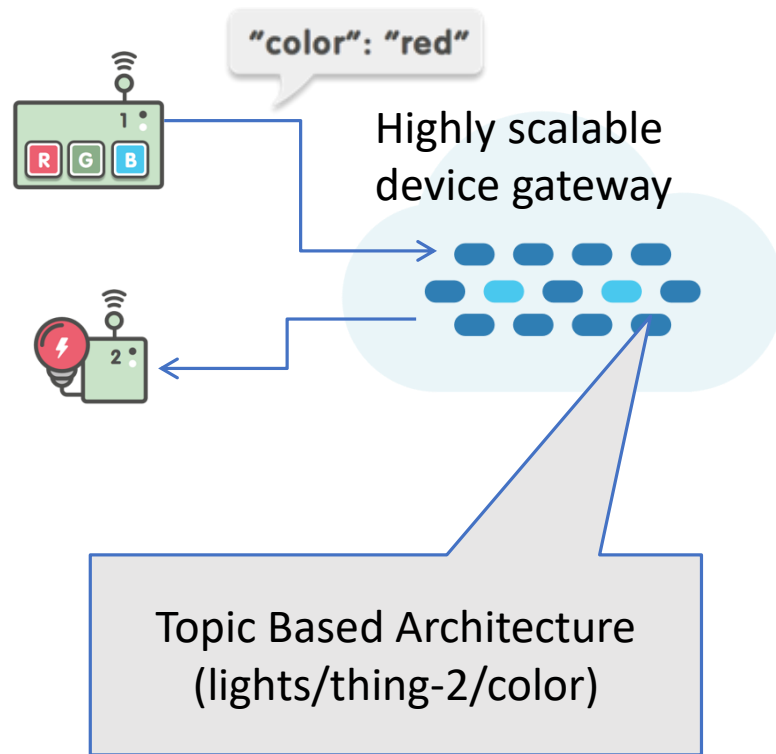
Example code with MQTT



# AWS IoT Device Gateway



# AWS IoT Device Gateway



## Standard Protocol Support:

- MQTT and HTTP

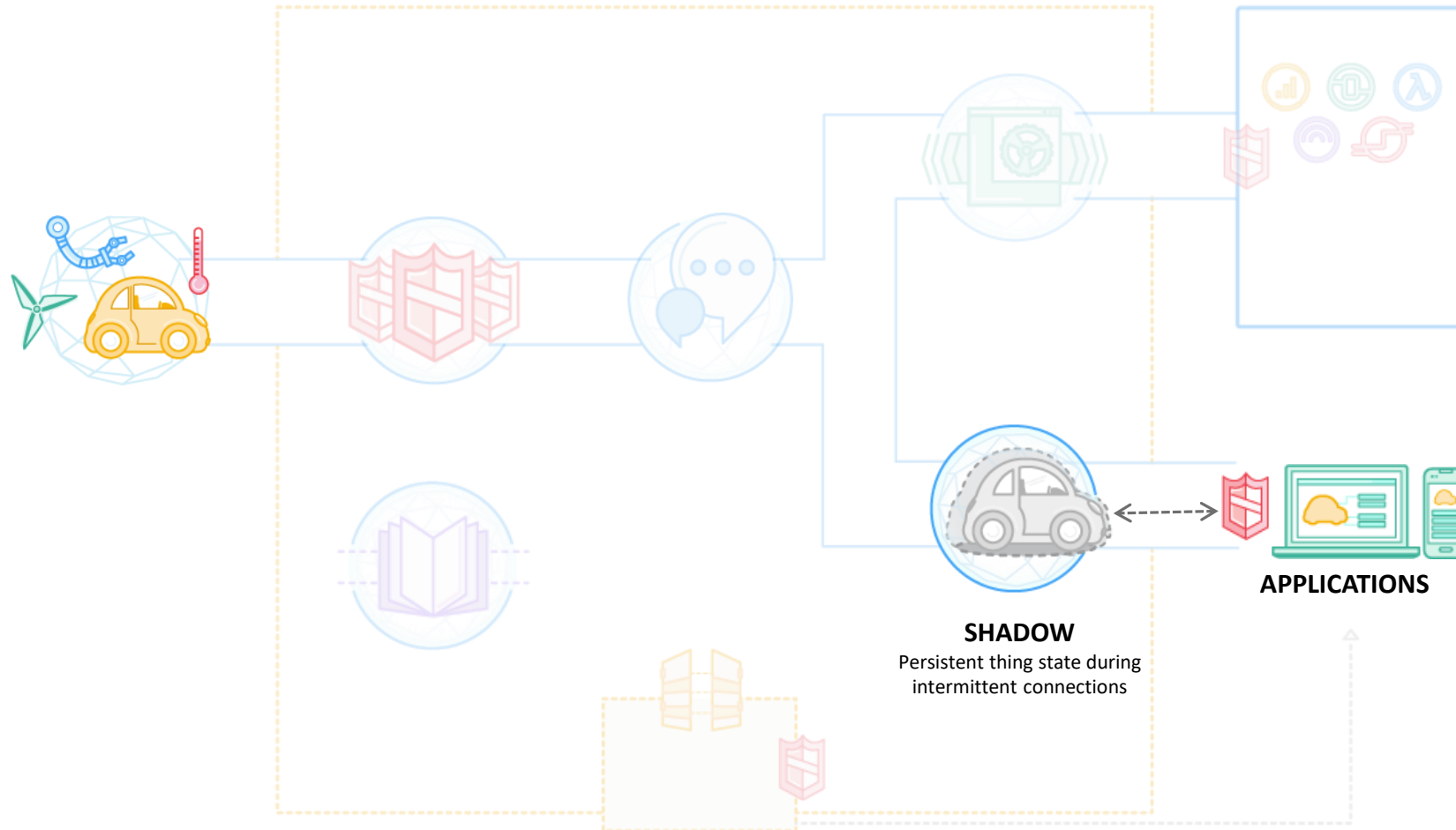
## Publish/Subscribe Broker with Long-lived bi-directional messages

- Clients (Devices and Apps) can receive commands and control signals from the cloud

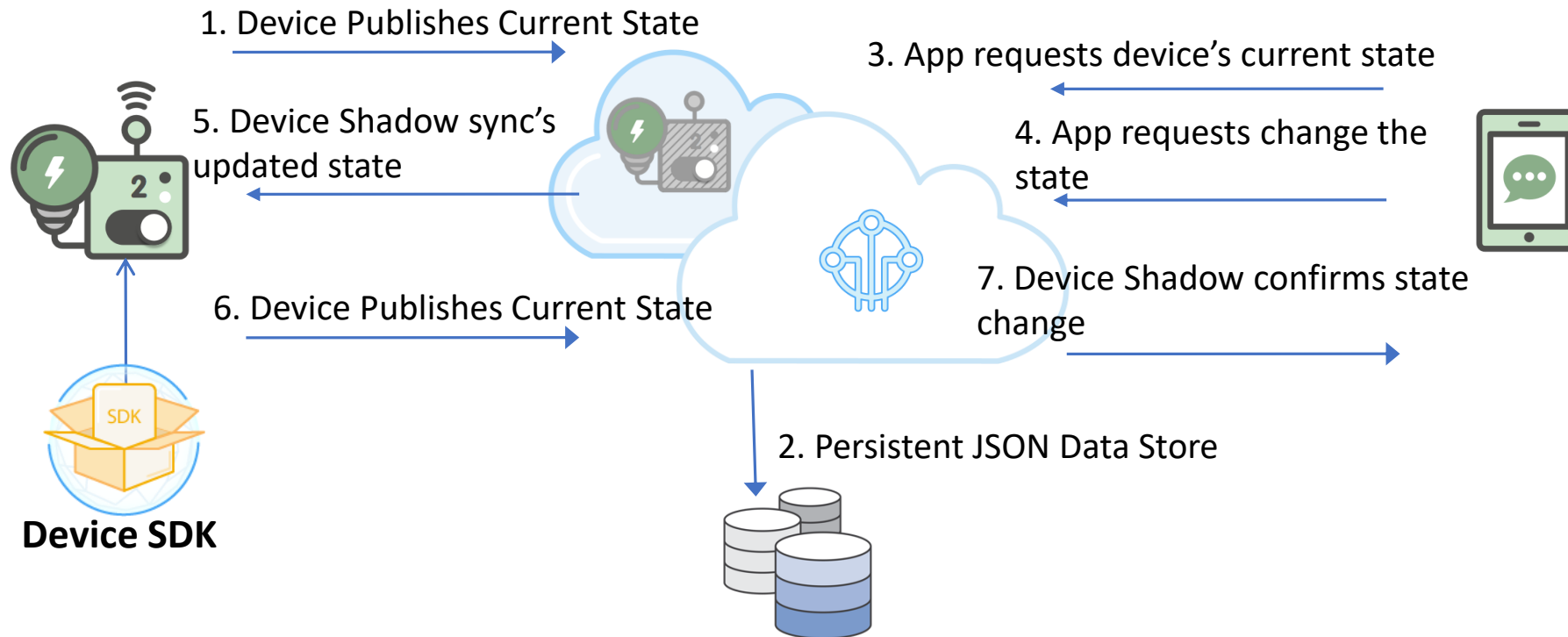
## Secure by Default

- Connect securely via X509 Certs and TLS client mutual authentication

# AWS IoT Device Shadow



# AWS IoT Shadow Flow



# AWS IoT Device Shadow Topics (MQTT)

Thing SDK (C-SDK, JS-SDK)  
makes it easy to build shadow  
functionality into a device so it  
can automatically synchronize  
the state with the device.

Sensor	Reported	Desired	Delta
LED1	RED	YELLOW	LED1 = Yellow TEMP = 60F
ACCEL	X=1,Y=5,Z=4	X=1,Y=5,Z=4	
TEMP	83F	60F	

**Reserved topics starting with \$** (refer to [topics](#))

**UPDATE:** \$aws/things/{thingName}/shadow/update

**DELTA:** \$aws/things/{thingName}/shadow/update/delta

**GET:** \$aws/things/{thingName}/shadow/get

**DELETE:** \$aws/things/{thingName}/shadow/delete

# Publish Using JSON

- **JSON (JavaScript Object Notation)**

- A lightweight data-interchange format
  - Easy for humans to read and write
  - Easy for machines to parse and generate.
- A thing can send its current state to the Thing Shadows service by sending an MQTT message to the topic *\$aws/things/myLightBulb/shadow/update*

```
{
  "state": {
    "reported": {
      "color": "red"
    }
  }
}
```

# RESTful API Accessing Shadow

**curl** is a tool to transfer data from or to a server, using one of the supported protocols including **HTTP** and **HTTPS**

- Delete all data from a thing shadow by setting its state to null  
`curl -H "Content-Type: application/json" -X POST -d '{"state":null}' -k --cert ./a5aedfc048-certificate.pem.crt --key ./a5aedfc048-private.pem.key https://A3VOQMFBV77HZI.iot.us-west-2.amazonaws.com:8443/things/loT-motion-sensor/shadow`
- `curl -H "Content-Type: application/json" -X POST -d '{"state":{"desired":{"motion":"0","time":"hello"}}}' -k --cert ./a5aedfc048-certificate.pem.crt --key ./a5aedfc048-private.pem.key "https://A3VOQMFBV77HZI.iot.us-west-2.amazonaws.com:8443/things/loT-motion-sensor/shadow"`
- `curl -H "Content-Type: application/json" -X POST -d '{"state":{"reported":{"motion":"0","time":"hello"}}}' -k --cert ./a5aedfc048-certificate.pem.crt --key ./a5aedfc048-private.pem.key https://A3VOQMFBV77HZI.iot.us-west-2.amazonaws.com:8443/things/loT-motion-sensor/shadow`

# Outline

Introduction

Device registry - thing, keys, certificate, policy

Security and identity

Device gateway – MQTT

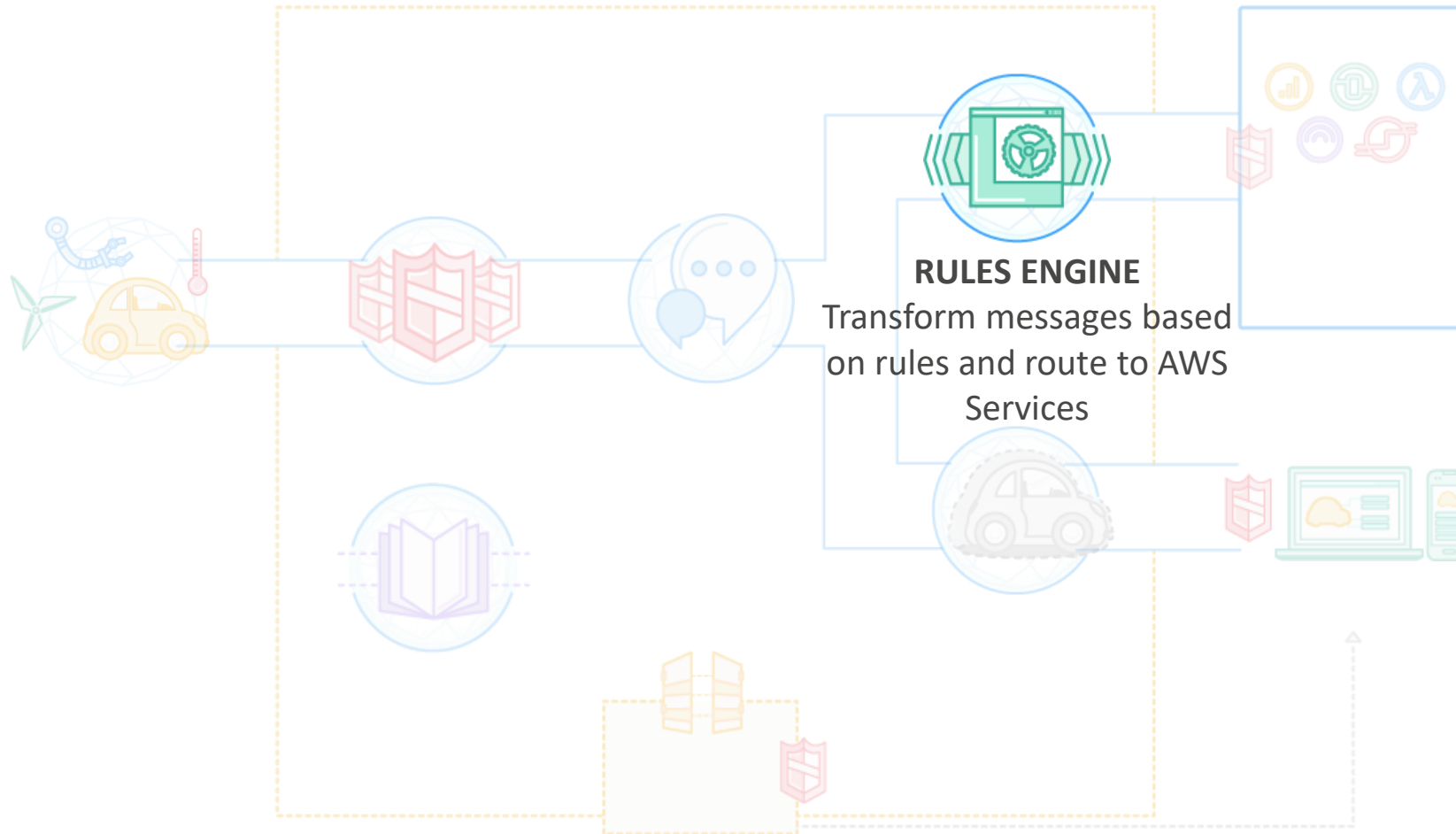
**Rules Engine**

Pricing

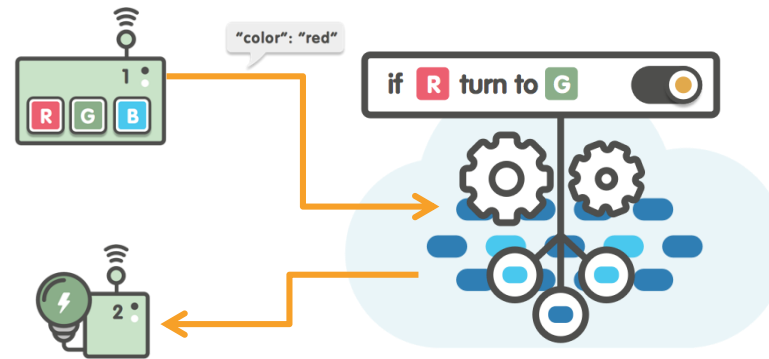
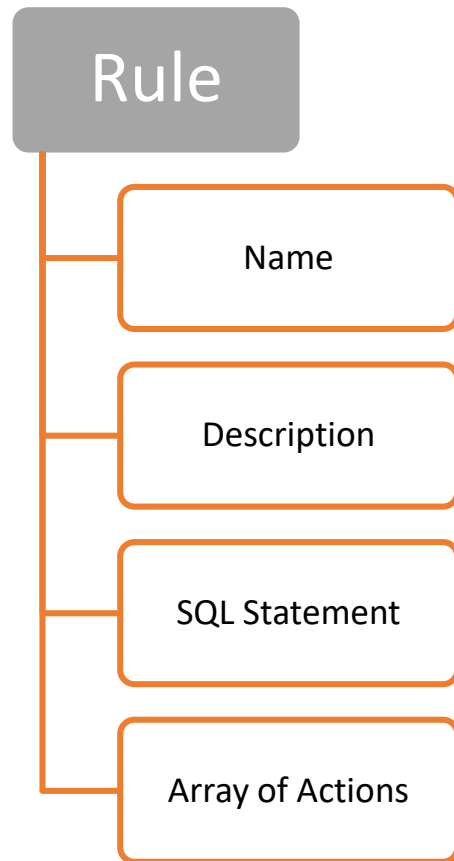
Example code with MQTT



# AWS IoT Rules Engine

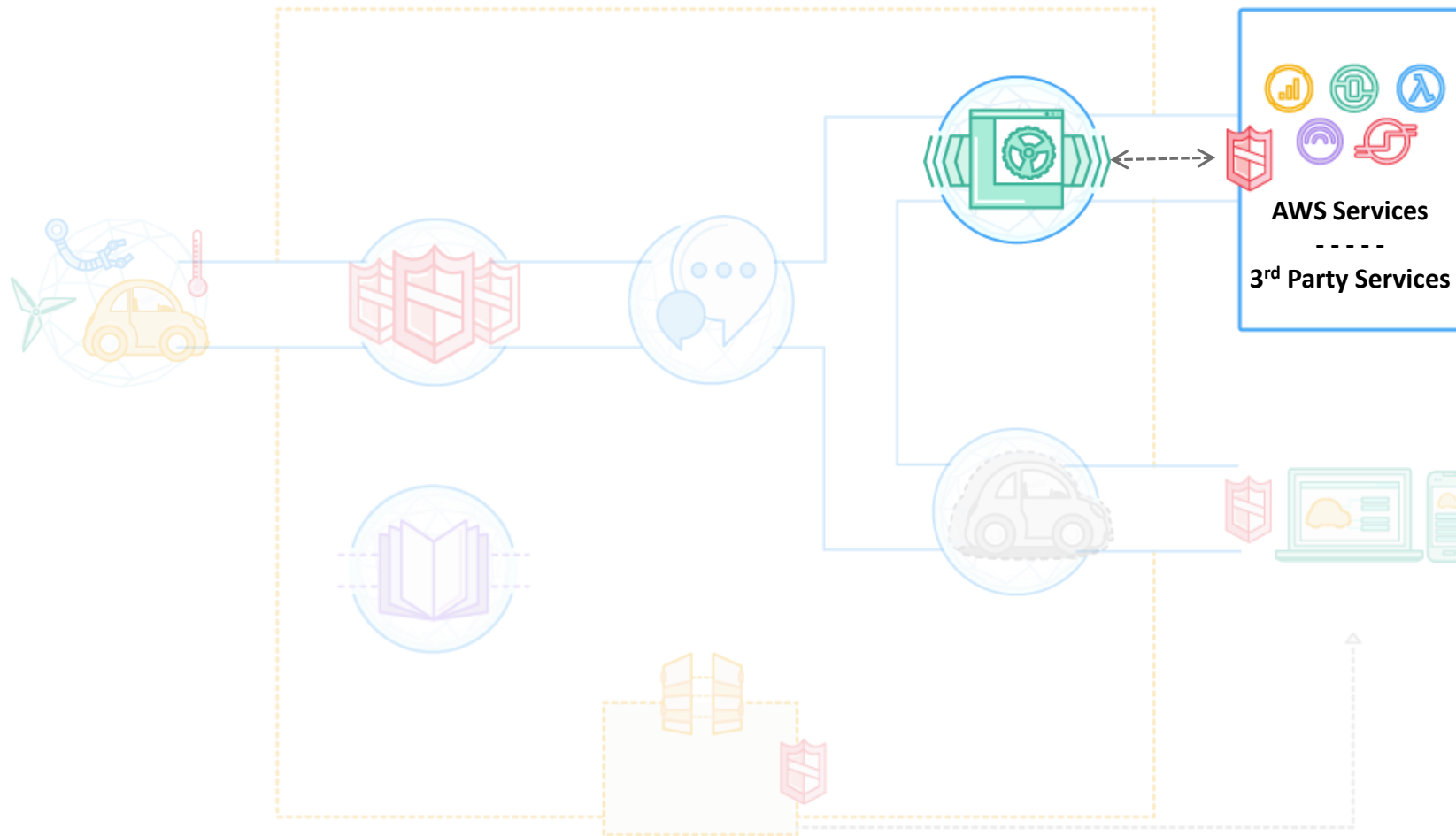


# AWS IoT Rules Engine Basics



```
SELECT * FROM 'things/thing-2/color' WHERE
color = 'red'
```

# AWS IoT Rules Engine Actions






## AWS IoT SQL Reference

# Outline

Introduction

Device registry - thing, keys, certificate, policy

Security and identity

Device gateway – MQTT

Rules Engine

**Pricing**

Example code with MQTT

# Pricing - Pay as You Go

- No minimum
- \$5 per million** messages published to, or delivered in US East (N. Virginia), US West (Oregon), EU (Ireland)
- \$8 per million** in Asia Pacific (Tokyo)
- No fees** for Rules, Shadows, Deliveries to other AWS Services

## Free Tier

250,000 Messages Per Month Free for first 12 Months



# Outline

Introduction

Device registry - thing, keys, certificate, policy

Security and identity

Device gateway – MQTT

Rules Engine

Pricing

Example code with MQTT



## Example code: publish motion sensor data to AWS IoT

```
#!/usr/bin/python3

#required libraries for mqtt and AWS IoT
import sys
import ssl
import json
import paho.mqtt.client as mqtt

# for motion sensor
import RPi.GPIO as GPIO
import time
from datetime import datetime

#called while client tries to establish connection with the server
def on_connect(mqttc, obj, flags, rc):
    if rc==0:
        print ("Subscriber Connection status code: "+str(rc)+" | Connection status: successful")
        mqttc.subscribe("$aws/things/IoT-motion-sensor/shadow/update/accepted", qos=0)
        # mqttc.publish("$aws/things/IoT-motion-sensor/shadow/update", '{"state":{"reported":{"color":"Fu"}}}')
    elif rc==1:
        print ("Subscriber Connection status code: "+str(rc)+" | Connection status: Connection refused")
    # message_json['state']['reported']['color'] == "RED"

#called when a topic is successfully subscribed to
def on_subscribe(mqttc, obj, mid, granted_qos):
    print("Subscribed: "+str(mid)+" "+str(granted_qos)+"data"+str(obj))

#called when a message is received by a topic
def on_message(mqttc, obj, msg):
    print("Received message from topic: "+msg.topic+" | QoS: "+str(msg.qos)+" | Data Received: "+str(msg.payload))

#creating a client with client-id=mqtt-test
mqttc = mqtt.Client(client_id="xinwenfu0")

mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe
mqttc.on_message = on_message
```

```
#Configure network encryption and authentication options. Enables SSL/TLS support.
#adding client-side certificates and enabling tls_v1.2 support as required by aws-iot service
mqttc.tls_set(ca_certs="/home/pi/fu/certs/VeriSign-Class3-Public-Primary-Certification-Authority-G5.pem",
              certfile="/home/pi/fu/certs/a5aedfc048-certificate.pem.crt",
              keyfile="/home/pi/fu/certs/a5aedfc048-private.pem.key",
              tls_version=ssl.PROTOCOL_TLSv1_2,
              ciphers=None)

#mqttc.tls_insecure_set(True)
#connecting to aws-account-specific-iot-endpoint
mqttc.connect("A3VOQMFBV77HZI.iot.us-west-2.amazonaws.com", port=8883)
#AWS IoT service hostname and portno

#automatically handles reconnecting
#mqttc.loop_forever()
# start a new thread handling communication with AWS IoT
mqttc.loop_start()

sensor = 12
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(sensor, GPIO.IN)

rc=0
try:
    while rc == 0:
        i = GPIO.input(sensor)
        print(i)      # i = 1: Motion detected; i = 0: No Motion
        data={}
        data['motion']=i
        data['time']=datetime.now().strftime('%Y/%m/%d %H:%M:%S')
        payload = '{"state":{"reported":'+json.dumps(data)+'}}'
        #json.dumps(data)
        print(payload)

        #the topic to publish to
        #The names of these topics start with $aws/things/thingName/shadow."
        msg_info = mqttc.publish("$aws/things/IoT-motion-sensor/shadow/update", payload, qos=1)

        time.sleep(1)

except KeyboardInterrupt:
    pass

GPIO.cleanup()
```

# References

- [1] [Get started with AWS IoT](#), 2017
- [2] [AWS IoT developer guide](#), 2016
- [3] Onur ŞALK, [Amazon Web Services IoT](#), November 02, 2015
- [4] [Get Started with AWS IoT and Raspberry Pi](#), Oct. 18, 2015
- [5] [AWS January 2016 Webinar Series - Getting Started with AWS IoT](#), Jan 26, 2016
- [6] [AWS Identity and Access Management User Guide](#), 2016
- [7] [paho-mqtt 1.1](#), 2016
- [8] [Introducing JSON](#), 2016