Amazon Web Services – Internet of Things

# AWS IoT Foundational Concepts

* Device Gateway
* Rules Engine
* Device Shadow
* Registry
* Security

# AWS IoT API and Design Pattern

1. Device SDK
   1. Set of client libraries to connect, authenticate and exchange messages
2. Authentication Authorization
   1. Secure with mutual authentication and encryption
3. Device Gateway
   1. Communicate with devices via MQTT and HTTP
4. Rules Engine
   1. Transform messages based on rules and route to AWS Services
5. Device shadow
   1. Allows persistent Thing state during intermittent connections

# Device Gateway

* Standard Protocol Support (no lock-in)
  + Millions of devices and apps can connect over any protocol starting with MQTT ad HTTP 1.1
* Powerful Pub/Sub 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 Certifications and TLS 1.2 Client Mutual Authentication

# One Service, Two Protocols

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

# Mutual Auth TLS (transport layer security)

* Since devices cannot login with secure passwords a behavior a different pattern is adopted
* AWS IoT API will provide a certificate and private key pair send it over a secure session and bring it into the device firmware
* Devices will not require an AWS ID or IAM credential, but IAM rules and policies can be assigned to the service instead to forgo direct identification management

# Provisioning and Security

* Secure communication with Things
  + Single API call to **CreateKeysAndCertificate()**
  + Client generated **CreateCertificateFromCSR(CSR)**
* Fine-grained Authorization for:
  + Thing management
  + Pub/sub data access
  + AWS Service Access

# Rules Engine

* Simple & Familiar Syntax
  + SQL statement to define topic filter
  + Option WHERE clause
  + Advanced JSON support
* Functions improve signal/noise
  + String manipulation (regex support)
  + Mathematical operations
  + Context based helper functions
  + Crypto support
  + UUID, timestamp, rand, etc.
* Complex evaluations
* Multiple/simultaneous actions
* Connects AWS IoT to external endpoints and AWS services; this relationship is called, Actions
* Actions
  + Rules engine evaluates inbound messages published into AWS IoT, transforms and delivers to the appropriate endpoint based on business rules
  + External endpoints ca be reached via Lambda and Simple Notification Service
* AWS SNS pairing with AWS IoT
  + Enable push notifications
  + Trigger HTTP endpoints, SMS, or email

# Thing Shadow

* Virtual representation of your device in the cloud
* Shadow will persist state of the device until the connected device wakes up
* Flow
  + Actual device publishes current state to its device shadow
  + Shadow will persist data as JSON in the data store of AWS IoT up to 8Kb
  + Applications can request the device(s) current state
  + Application can request to change the device(s) state
  + Device shadow will sync the actual device’s state
  + Actual device can then again publish its current state
* Thing – reports its current stat to one or multiple shadows and retrieves its desired state from a shadow(s)
* Shadow – reports delta, desired and reported states along with metadata and version
* Application – sets the desired state of a device and get the last reported state of the device as well as can delete the shadow
* Message <Object>
  + State <Object>
    - Desired <Object>
    - Reported <Object>
    - Delta <Object>
  + Version <Number>
* Topics
  + UPDATE: **$aws/things/{thingName}/shadow/update**
  + DELTA: **$aws/things/{thingName}/shadow/delta**
  + GET: **$aws/things/{thingName}/shadow/get**
  + DELETE: **$aws/things/{thingName}/shadow/delete**

# Registry

* Static attribute associated to Thing
  + Firmware version
  + Serial number
  + Device type
  + Device group
  + Device description
  + Sensor description
* Support and maintenance
  + Reference manual URL
  + Part # reference
* Reference to external support system

# Device Management

* Ability to update globally or within a Region
* Rules Engine keeps state of updates and tracks progress in a DynamoDB Table
* Store version in Registry Entry
* Functionality examples
  + Hold versioned firmware distributions in AWS S3
  + Message brokers can notify groups in the device fleet using Topic Patterns

# Pay as you go & Predictable Pricing

* Pay as you go with no minimum fees
* $5/million messages published to or delivered in US East, US West, EU
  + $8/million in Asia Pacific
* Free Tier
  + 250k messages/month for the first 12 months

# Pricing Example

* 100 sensors publishing 1x/minute
  + 100 sensors \* 30 days \* 24 hours \* 60 minutes = 4.38 million messages
* Messages deliver to AWS IoT to manage received transmissions
  + Metering Unit: receives all sensor data for AWS IoT
  + DynamoDB Table: receives all sensor data too
* Break down for 4.38 million messages
  + $21.90 to publish messages 4.38 million messages to AWS IoT
  + $21.90 to deliver messages from AWS IoT to Metering Unit
  + $0 to deliver message to DynamoDB

# AWS IoT SDKs

* Android
* Arduino
* Embedded C
* C++
* iOS
* Java
* JavaScript
* Python