IOT

Services

# Index

Table of Contents

[Index 2](#_Toc28020834)

[IoT Core 3](#_Toc28020835)

[Definition 3](#_Toc28020836)

[IoT Device Gateway 3](#_Toc28020837)

[IoT Thing Registry 3](#_Toc28020838)

[IoT Message Broker 4](#_Toc28020839)

[IoT Shadow Service 4](#_Toc28020840)

[IoT Rules Engine 5](#_Toc28020841)

[IoT Job Service 5](#_Toc28020842)

[Security 6](#_Toc28020843)

# IoT Core

## Definition

* It is a managed cloud platform that lets connected device (such as IOT sensors, actuators, embedded micro-controllers or smart appliances) easily and securely interact with cloud applications and other devices.
* It can support billions of devices and trillions of messages and can process and route those messages to AWS endpoints and to other devices reliably and securely.
* It enables one to create applications can keep track of and communicate with all devices, all the time, even when they aren’t connected.

## IoT Device Gateway

* The Device Gateway forms the backbone of secure, low-latency, low-overhead, bi-directional communication between connected devices and the cloud capabilities such as the Rules Engine, Device Shadow, and other AWS and 3rd-party services.
* The Device Gateway supports the pub/sub messaging pattern, which enables scalable, low-latency, and low-overhead communication. It is particularly useful for IoT scenarios where billions of devices are expected to communicate frequently and with minimal delay.
  + Pub/sub involves clients publishing messages on logical communication channels called ‘topics’ and clients subscribing to topics to receive messages. The device gateway enables the communication between publishers and subscribers.
  + Traditionally, organizations have had to provision, operate, scale, and maintain their own servers as device gateways to take advantage of pub/sub. AWS IoT Core has eliminated this barrier by providing the Device Gateway.
* It allows for
  + Communications across multiple protocols such as:
    - MQTT
    - WebSockets
    - HTTP 1.1
  + Devices connecting on MQTT and WebSockets to maintain long-lived two-way connections.

## IoT Thing Registry

* It is used to keep track of all of one’s internet-connected things to enable effective interaction from applications and other devices.
* It can be used to create things by registering them inside the registry, list/ describe and delete things as well as search for things based on thing name or thing type.
* It allows one to associate a physical device with a X.509 certificate to allow authenticated communication between the thing and AWS IoT.
* Thing Types
  + AWS allows one to use thing types to store configuration information that is same for all things associated with said thing type. (Think template)
  + For example, all things of thing type ‘X’ will share a set of attributes like *model number*, *wattage*, etc.
  + Each thing type attribute can have different values specific to the thing in question.
  + Things within a thing type can have up to 50 attributes. Without a thing type, things can have up to 3 attributes.
  + A thing can only be associated with a single thing type.
* Thing Groups
  + AWS allows one to categorize things into groups.
  + One can add, update or delete attributes of a group.
  + One can manage permissions over multiple things by attaching or detaching polices to or from a group.
  + One can configure logging at the group level.

## IoT Message Broker

* It is a publish and subscribe service (similar to AWS SNS) that enables sending and receiving messages to and from IoT.
* Publishers
  + The entities that send out messages to a Message Broker Topic.
* Topics
  + This is where messages are published.
  + Topics would have names such as ‘*sensor/humidity/station01*’.
  + Topic names are isolated at AWS Account & Region level.
  + Topics can be published to IoT devices or external applications and users.
  + There are existing, reserved topics for lifecycle events and the Device Shadow Service.
    - Some lifecycle events include connect/ disconnect events, subscribe/ unsubscribe events, etc.
* Subscribers
  + The entities that register to receive messages from a topic.
* Message Broker supports MQTT to publish and subscribe and HTTPS to publish over IPv4 and IPv6. MQTT is also supported over web sockets.

## IoT Shadow Service

* An IoT device’s shadow is a virtual device/JSON document that is used to store the state of a device.
* Every device connected to AWS IoT has a device shadow that can be queried over MQTT even if the device itself is not connected to the internet.
  + Once the device comes back online, it can inherit the latest state information from it’s shadow.
* Shadow document contains properties such as:
  + **State**
    - **Desired:** The desired state of a thing. Applications can write to this document field to update the state of a thing without directly being connected to the thing.
    - **Reported:** The reported state of a thing. Things write to this field to report their new state and applications read it to determine the state of that thing.
  + **Metadata:** Information (such as timestamps) about the data stored inside the state portion of the document.
  + **Timestamp:** Indicates when IoT transmitted the message.
  + **Client Token:** A string unique to thing (Enables associating MQTT responses with the requests).
  + **Version:** The document’s incremented version.
* The shadow service uses IoT Message Broker topics to publish and receive information related to interacting with device shadows.
  + The topics are specific to each device and start with ‘*$aws/things/<thingName>/shadow’.*
  + For example, a full topic name for getting the device shadow for a device called ‘car1’ might look like ‘*$aws/things/car1/shadow/get*’

## IoT Rules Engine

* IoT Rules allows devices to interact with other AWS Services.
* Rules are analyzed and trigger actions based on the MQTT topic streams.
* Rules allows one to do things like:
  + Save files to S3.
  + Invoke a Lambda function.
  + Send information to AWS SQS, SNS, DynamoDB, Kinesis etc.
* How it works:
  + An IoT SQL statement is used to filter incoming MQTT messages and push the data elsewhere.
  + One or more actions can be configured when such messages are received.
    - Proper IAM permissions should be configured as well.
  + One can set an error when earlier actions fail.

## IoT Job Service

* IoT Jobs are a set of operations sent to, and executed on, AWS IoT devices.
* Jobs can instruct devices to download application updates, perform troubleshooting, rotate certificates, and other tasks.
* Some key concepts include:
  + **Job documents:** A UTF-8 encoded JSON document that contains the information a device needs to perform the job. Usually, this will include URLs where the devices can download an update or other data.
  + **Target:** A specification of a thing or thing groups that should perform the job.
  + **Job Execution:** An instance of a job on a target. Targets report their progress to AWS IoT.
  + **Rollouts:** A way to determine how quickly devices will be notified of a pending job execution. Value can range from 1 to 1000 jobs per minute.
  + **Job Types:**
    - **Snapshot Job:** A type of Job that is run once for all targets and is completed when the targets finish the job or report a failure to do so. (This is default job type)
    - **Continuous Job:** A type of job that is sent to all targets specified and continues to run. The job will also be sent to new devices added to a Thing Group.

## Security

* IoT devices use X.509 certificates for authentication and authorization.
  + Certificates can be created and registered with specific IoT devices.
  + Multiple polices granting or denying permissions can be attached to each certificate.
  + It is best practice to assign a unique certificate to each IoT device.
* For mobile devices, AWS Cognito can be used.
* For desktop devices, IAM & federated Identities can be used.

# AWS Greengrass

## Definition

* AWS Greengrass extends AWS Cloud capabilities to local devices by bringing the compute layer directly to the device in the form of AWS Lambda.
* This allows those devices to collect and analyze data close to the source of information and to communicate with each other on local networks securely.

## Greengrass Groups

Greengrass groups are a collection of settings for AWS Greengrass core devices and the devices that communicate with them.

* **Greengrass Group definition:** A definition that contains a collection of information about the AWS Greengrass group.
* **Greengrass Group Settings:** Settings for a Greengrass group including an AWS Greengrass group role, log configuration, certificate and connection information, and AWS Greengrass core connectivity information.
* **AWS Greengrass core:** An AWS IoT Thing that represents the Greengrass core.
* **Lambda function definition:** Lambda functions to be deployed in the AWS Greengrass core.
* **Subscription definition:** This definition is a list of subscriptions that enable communication with MQTT messages and contains:
  + **A message source:** An AWS IoT Thing, Lambda function, AWS IoT, or the local AWS Greengrass shadow device.
  + **A subject:** An MQTT topic or topic filter that’s used to filter message data.
  + **A message target:** Identifies the destination for messages (can be any of the same things as a message source)
* **Device definition:** A list of devices that are members of the Greengrass group with their configuration data.
* **Resource definition:** A list of local resources and machine learning resources on the AWS Greengrass core and configuration data.

## Types of Greengrass Devices

* AWS Greengrass Core
  + These devices run a special runtime that allows them to communication with AWS IoT and AWS Greengrass cloud services.
  + These devices are essentially AWS IoT devices that run a local AWS Lambda runtime.
  + They have their own certificates for authentication with AWS IoT, as well as device shadows and exist in the IoT Registry.
  + They run software to allow AWS IoT devices to find their group and core information.
* AWS IoT Devices
  + Other non-core devices can connect to the AWS Greengrass cores.
  + These devices run software written by AWS IoT Device SDK.
  + They can be sensors or actuators.
  + They can run outside the AWS Greengrass group