CS433: Internet of Things
NCS463: Internet of Things

Dr. Ahmed Shalaby

http://bu.edu.eg/staff/ahmedshalaby14

AWS IoT Core

- AWS IoT Core provides **secure, bidirectional communication** between internet-connected devices, such as sensors, actuators, embedded microcontrollers, or smart appliances, and the **AWS Cloud**. Using AWS IoT Core, you can collect, store, and analyze telemetry data from multiple devices. You can also **create applications** where your users can control these devices from their phones or other mobile devices

- AWS IoT Core is composed of six main components:

  1. *Identity service* – Provides authentication, authorization, and device provisioning.
  2. *Device gateway* – Securely connects IP-connected devices and edge gateways to the AWS Cloud and other devices at scale.
  3. *Message broker* – Processes and routes data messages to the AWS Cloud.
  4. *Rules* – Invokes actions in the AWS Cloud.
  5. *Device Shadow service* – Maintains a shadow of your device so the device can be accessed and controlled at any time.
  6. *Registry* – Stores information about devices and their attributes.

Source: AWS IoTCore

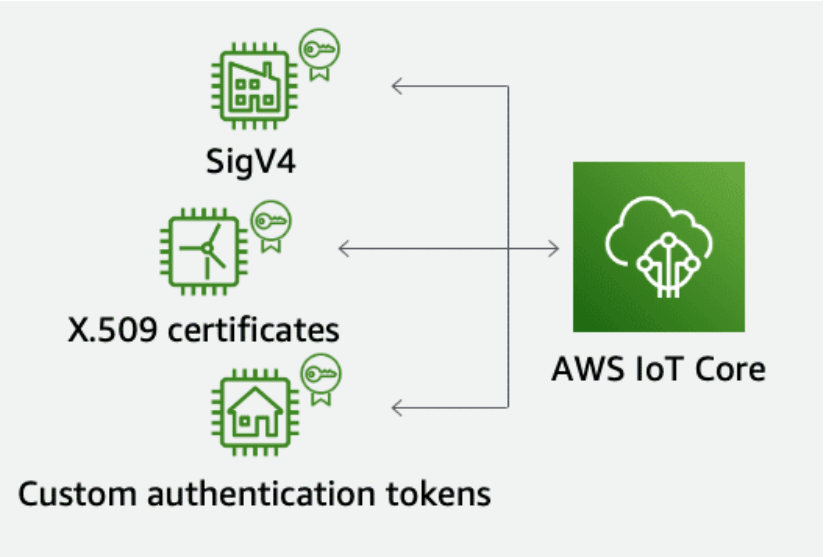
AWS IoT Core

❑ Identity Service

AWS IoT Core provides a secure communication channel for devices to communicate with each other and other services. AWS IoT provides authentication by offering the following options:

- **Certificates** for mutual authentication by using MQTT over Transport Layer Security (TLS) v1.2.
- **Signature Version** 4 (SigV4) signed requests over HTTP.
- **MQTT over WebSocket**, is similar to other AWS services.

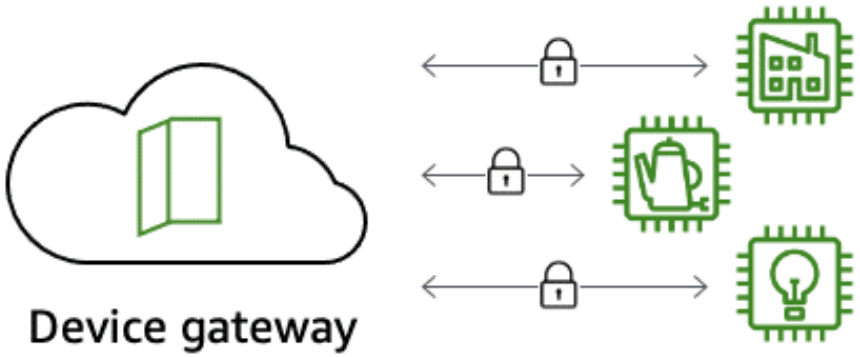
❑ You can also use custom authentication tokens that are provided by your authentication or authorization service.



AWS IoT Core

❑ Device Gateway

The device gateway serves as the **secure entry point** for IoT devices connecting to AWS IoT Core. The device gateway **manages all active device** connections and implements semantics for multiple protocols to ensure that devices are able to securely and efficiently communicate with AWS IoT Core.

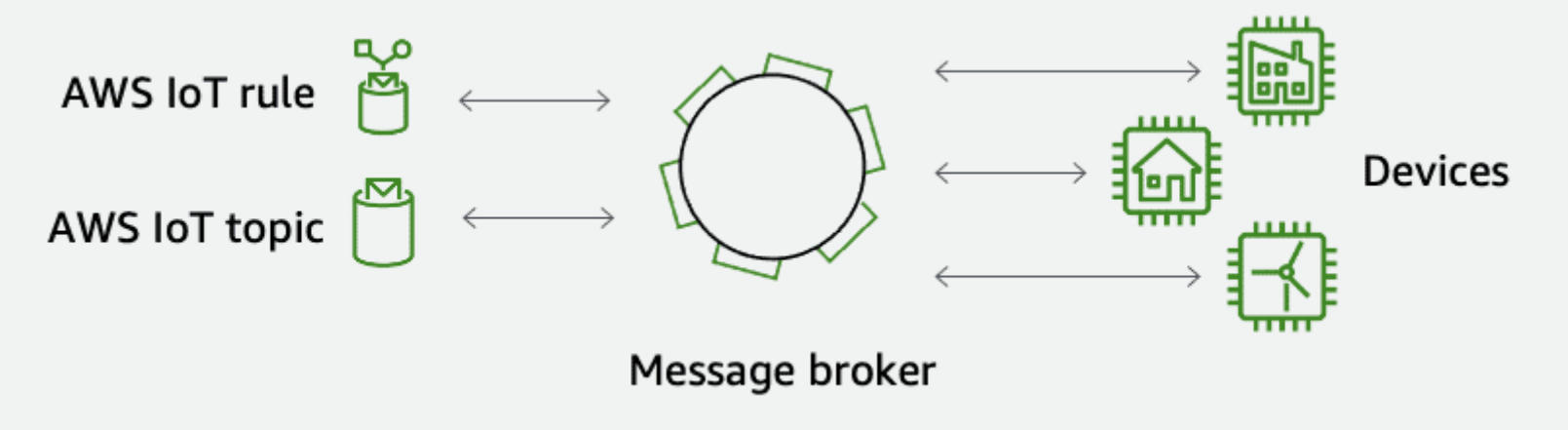


AWS IoT Core

❑ Message broker

The message broker processes and routes data from your devices into AWS IoT Core. The message broker is scalable, has low latency, and provides reliable message routing. It also uses a publish and subscribe model to decouple devices and applications using MQTT.

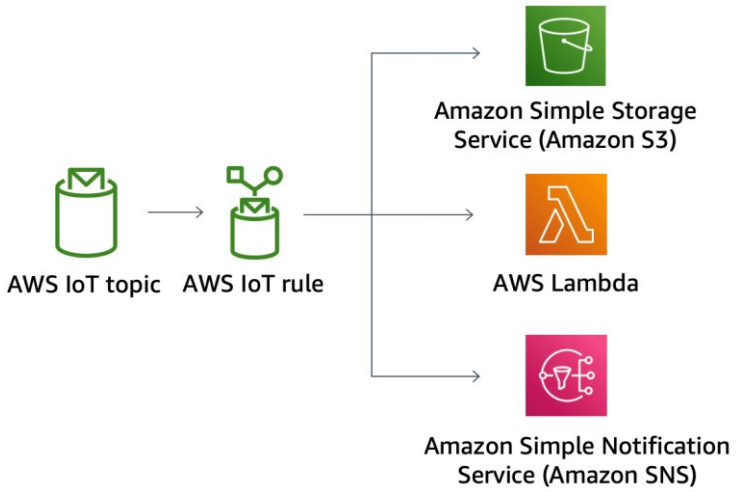
The message broker supports two-way message streaming between devices and applications, including the use of AWS IoT rules and topics. It also provides an opportunity for data transformation, rerouting, and enhancement with external data sources.



AWS IoT Core

❑ Rules

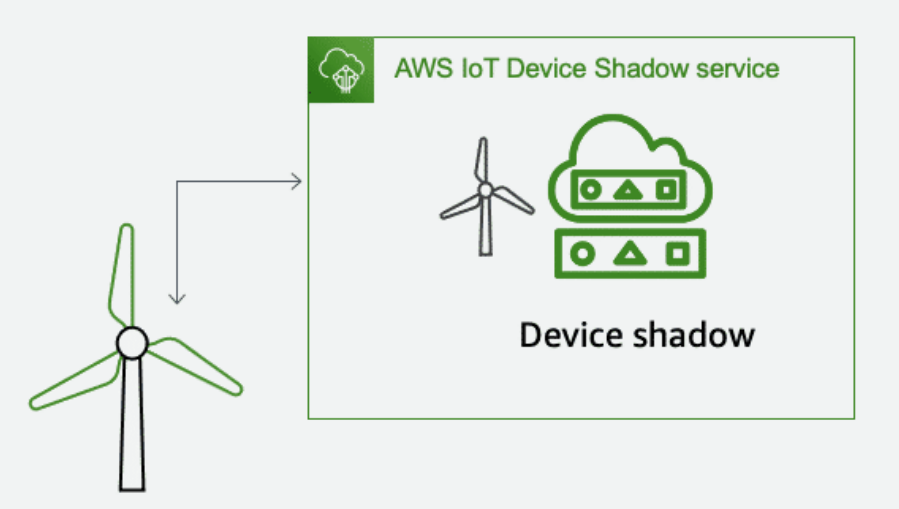
AWS IoT Core processes rules and **finds incoming messages that match the rule query**. When a matching message is received, the **rule action is initiated** such as writing data to an Amazon Simple Storage Service (Amazon S3) bucket, invoking an AWS Lambda function, or sending a message to an Amazon Simple Notification Service (Amazon SNS) topic.



AWS IoT Core

❑ Device Shadow service

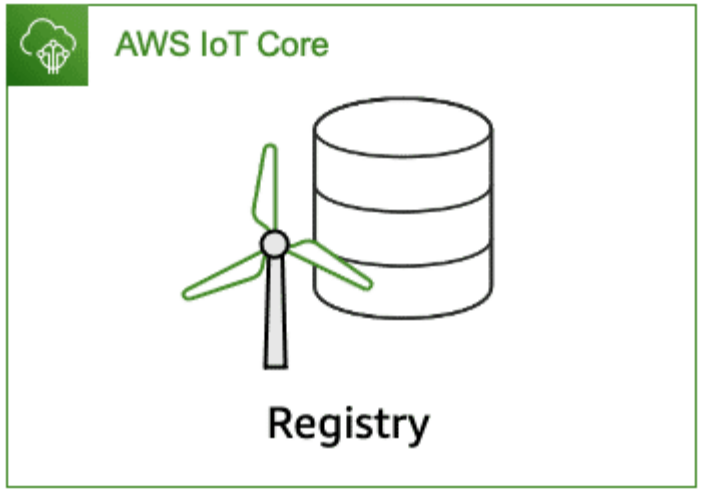
A device shadow can also be referred to as an **AWS IoT thing shadow**. The Device Shadow service **maintains a device shadow** for each device you connect to AWS IoT. You can use the shadow to access a device's state whether the device is **connected to AWS IoT or not**.



AWS IoT Core

❑ Registry

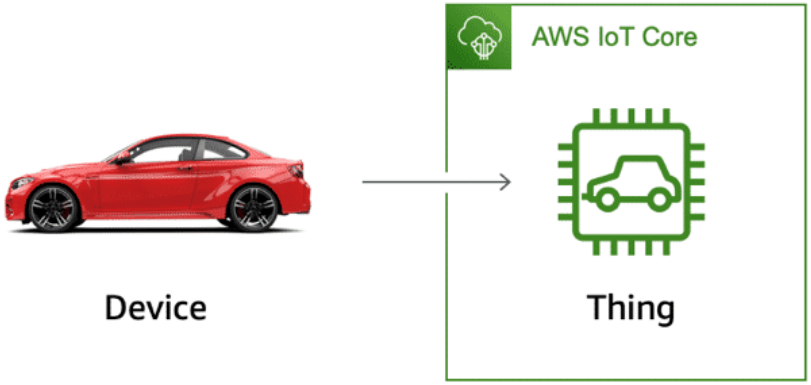
The registry is a **database of devices**. Using the registry for your devices is optional; however, the registry helps you manage your device ecosystem effectively and acts as a repository for device certificates. Using the registry, you can **search registered devices based on attributes and tags**



AWS IoT Core

❑ Things and Devices

- A thing is a representation of a specific device or logical entity. It can represent a physical device or sensor, such as a light bulb or a switch on a wall. It can also represent a logical entity, such as an instance of an application or a physical entity not directly connected to AWS IoT Core but related to devices that do connect to AWS (for example, a car that has engine sensors or a control panel).
- **Information** about a thing is stored in the AWS IoT Core registry as JSON data. Examine the following block of JSON data.




```
{
    "version": 3,
    "thingName": "truckSensor01",
    "defaultClientId": "truckSensor01",
    "thingTypeName": "sensor_DoorAndPower",
    "attributes": {
        "deviceId": "T001",
        "powerRequirements": "12v"
    }
}
```

AWS IoT Core

❑ Things and Devices

- A thing type is a method to organize AWS IoT things into logical categories, such as light bulbs, thermostats, and motion sensors. Thing types allow you to store description and configuration information that is common to all things associated with the same thing type. This simplifies the management of things in the registry. For example, you can define a LightBulb thing type. All things associated with the LightBulb thing type share a set of attributes: **serial number, manufacturer, and wattage**. When you create a thing of type LightBulb (or change the type of an existing thing to LightBulb) you can specify values for each of the attributes defined in the LightBulb thing type.

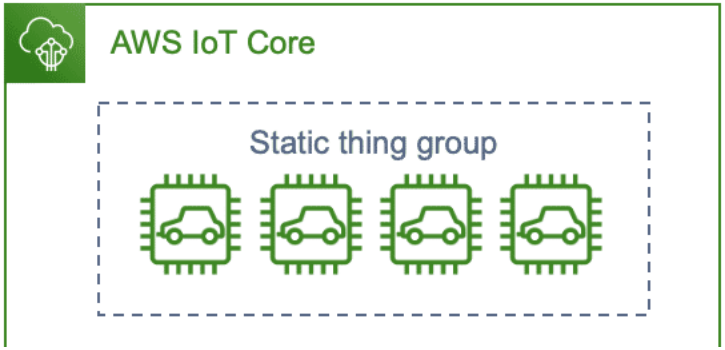
- Example: Thing type JSON

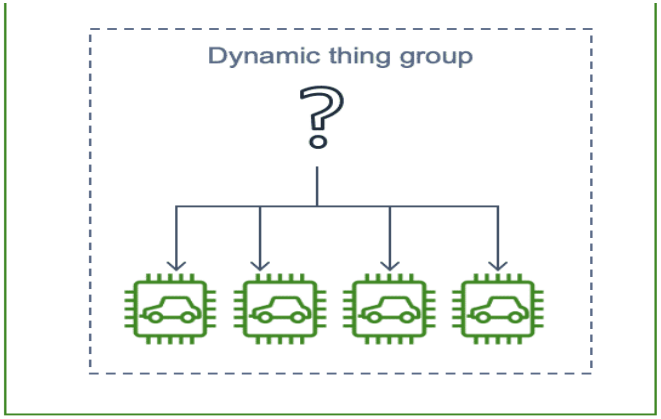
```
{
    "thingTypes": [
        {
            "thingTypeName": "sensor_DoorAndTemp",
            "thingTypeProperties": {
                "searchableAttributes": [
                    "deviceId",
                    "powerRequirements"
                ],
                "thingTypeDescription": "sensor for freezer trucks"
            },
            "thingTypeMetadata": {
                "deprecated": false,
                "creationDate": 1468423800950
            }
        }
    ]
}
```

AWS IoT Core

❑ Thing groups

- There are two types of thing groups.
  - Static thing groups organize devices into groups that you specify. Things are added to a static thing group by using the console, AWS Command Line Interface, or the AWS IoT API.
  - Dynamic thing groups update group membership through search queries. For example, suppose that you want to update the firmware, But, to minimize the chance that the update is interrupted, you only want to update firmware on devices with battery life >80%. You can create a dynamic thing group that only includes devices with a reported battery >80%. Only devices that meet your battery life criteria receive the firmware update. As devices reach the 80% battery life criteria, they are added to the dynamic thing group and receive the firmware update.

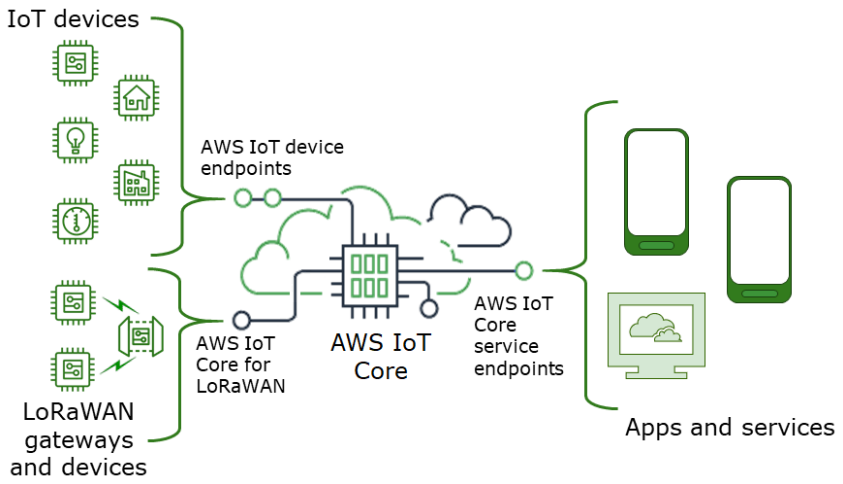




AWS IoT Core

❑ Device software

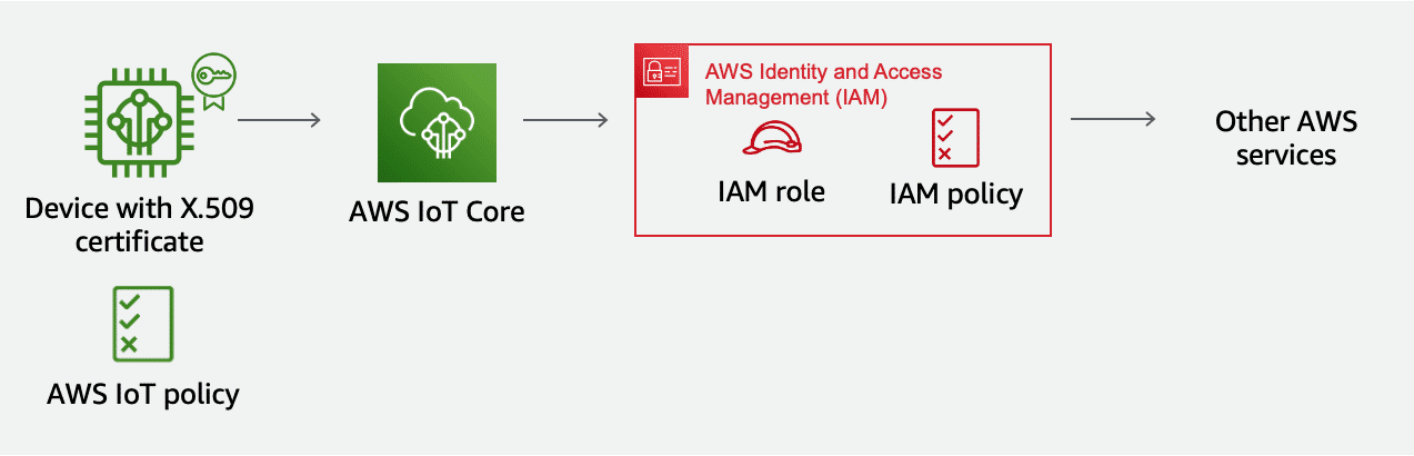
AWS IoT Greengrass is software that extends cloud capabilities to local devices. This enables devices to collect and analyze data closer to the source of information, react autonomously to local events, and communicate securely with each other on local networks.



Device software	Scenario	Device hardware
FreeRTOS	Small, highly constrained, and purpose-built IoT devices and sensors	Microcontroller
AWS IoT Device Software Development Kit (SDK)	IoT devices running Linux distributions (distros) for basic IoT applications and prototyping	Microprocessor
AWS IoT Greengrass	IoT applications managed from the cloud that perform machine learning inference, process data, and are able to handle intermittent connectivity	Microprocessor

AWS IoT Core: Device Security

- AWS IoT Core uses X.509 certificates to grant access to the service for devices. After the devices have been authenticated, their allowed operations and actions in AWS IoT Core are based on an AWS IoT policy. Operations include connecting to the message broker, sending and receiving messages, and getting or updating a thing's device shadow.
- Then, AWS Identity and Access Management (IAM) roles and policies allow AWS IoT to access other AWS resources in your account on your behalf. For example, if you want to have a device publish its state to an Amazon DynamoDB table, IAM roles allow AWS IoT to interact with DynamoDB.

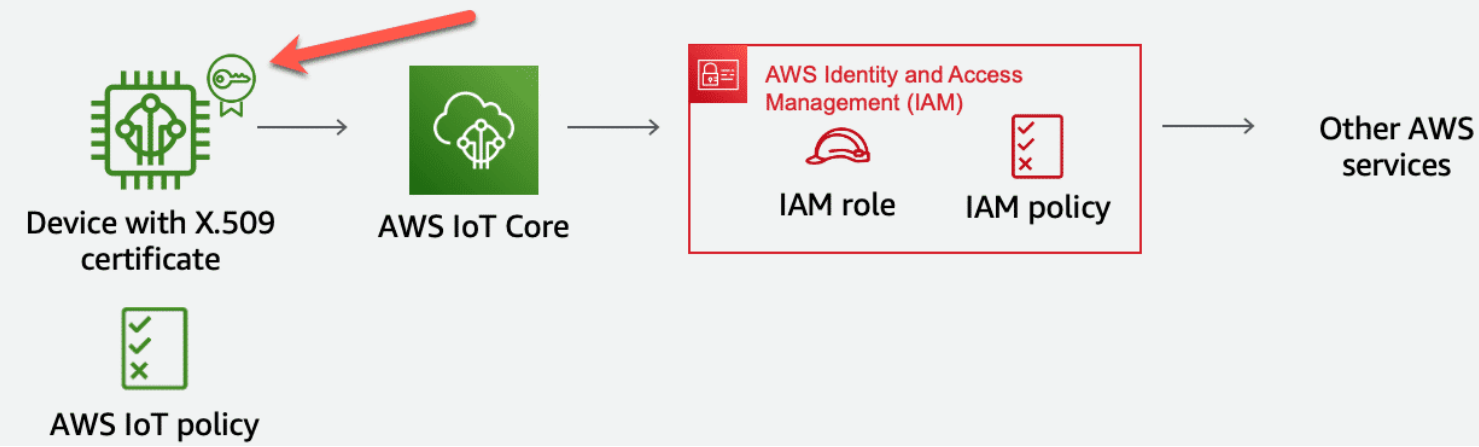


AWS IoT Core: Device Security

- **AWS IoT policy**: AWS IoT policies primarily control a resource's access to the **AWS IoT Core data** plane. When you **send data to and receive data** from AWS IoT Core, you are using the AWS IoT Core data plane. This means that the data plane defines whether you can connect to the message broker, send or receive MQTT messages, or **publish or subscribe to a specific topic**.
- **IAM role**: IAM roles work with IAM policies to grant AWS IoT Core permissions to access the rest of AWS. For example, AWS IoT receives data and communication from a device that is redirected to an AWS resource such as an Amazon Simple Storage Service (Amazon S3) bucket, a DynamoDB table, or any AWS service.
- All communication is encrypted with Transport Layer Security (TLS) version 1.2 to ensure that your application protocols remain secure and confidential. Not all devices support TLS v1.2, so when adding devices to your fleet, make sure that they support TLS v1.2.

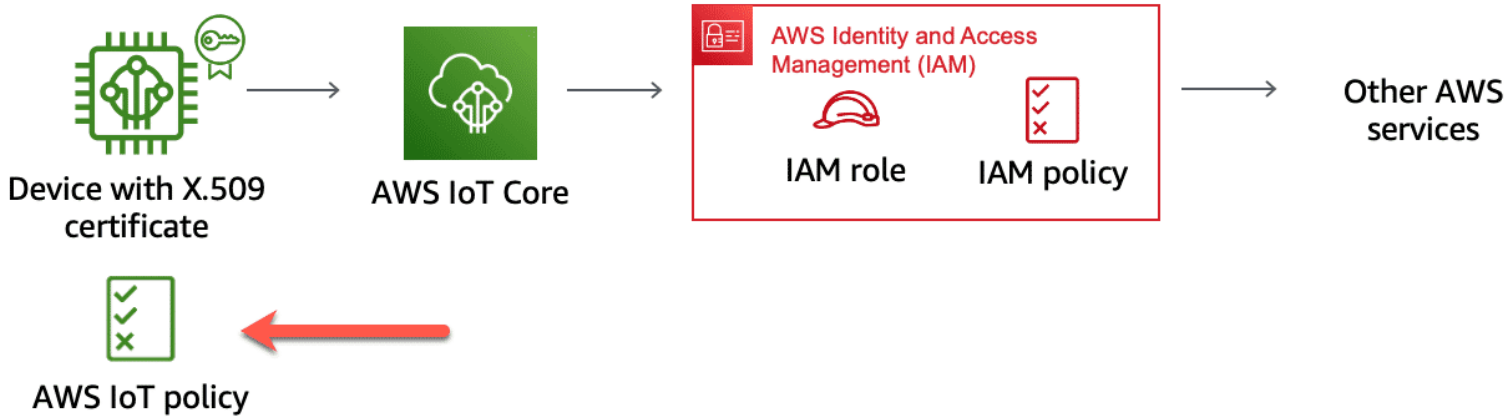
AWS IoT Core: Device authentication

- Authentication is the process of *verifying the identity* of a client or a server.
  - Server authentication is the process by which devices or other clients ensure that they are communicating with an actual AWS IoT endpoint.
  - Client authentication is the process by which devices authenticate with AWS IoT Core.
- Certificates allow devices and clients to connect to the AWS IoT Core. AWS IoT supports the following certificate-signing algorithms:
  - SHA256WITHRSA, SHA384WITHRSA, SHA512WITHRSA
  - DSA_WITH_SHA256
  - ECDSA-WITH-SHA256, ECDSA-WITH-SHA384, ECDSA-WITH-SHA512



AWS IoT Core: Device authorization

- Authorization is the process of granting permissions to an authenticated identity. After identities are authenticated, they still do not have permission to access resources. For devices and users to gain permission, they must be authorized to do so.
- Policies for AWS IoT define what an authenticated identity can do. Devices, mobile applications, web applications, and desktop applications all use an authenticated identity. The identity can execute AWS IoT operations only if it has a policy that grants it permission.



AWS IoT Core: Policies .

- Authorization for actions that can be taken in AWS IoT Core is managed through an AWS IoT policy. Policies **are JSON documents that contain a set of permissions**. The policies are then attached to the device, user, or role to give them the permissions written in the policy.
- An AWS IoT policy contains one or more policy statements. Each policy statement contains the following three keys:
  - Effect – Indicates whether the policy allows or denies **access**.
  - Action – Includes a list of **actions** that the policy allows or denies.
    - **iot:Connect** represents permission to connect to the message broker.
    - **iot:Publish** represents permission to publish to an MQTT topic.
    - **iot:Subscribe** represents permission to subscribe to an MQTT topic or topic filter.
    - **iot:GetThingShadow** represents permission to get a device's shadow.
  - Resource – Specifies a list of **resources** to which the actions apply. If no resource is listed, then the action applies to the resource to which the policy is attached.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Connect"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Publish"],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

AWS IoT Core: Auditing Devices

- **Device Advisor** is a fully managed cloud-based test capability for validating IoT devices during development. You can use pre-built tests to validate your IoT devices are set up for reliable and secure connectivity with AWS IoT Core.
- **AWS IoT Device Defender** is a fully managed service that helps you audit the configuration of your devices, monitor connected devices to detect abnormal behavior and mitigate security risks. It gives you the ability to enforce consistent security policies across your AWS IoT device fleet and respond quickly when devices are compromised.
- Device Defender capabilities include identifying upcoming expiring certificates and using **machine learning models** to determine and audit normal device behavior.
- Device Defender also provides built-in mitigation actions you can take to minimize the impact of security issues such as adding a thing to a thing group (for example, to quarantine a device), updating a device certificate, replacing the default policy version, and turning on IoT logging.

AWS IoT Core: Message Broker

- AWS IoT message broker is how clients and AWS IoT Core communicate. Clients send data by publishing a message on a topic. Clients receive messages by subscribing to a topic. When the message broker receives a message, it forwards the message to all clients subscribed to the topic.
- Topics are hierarchical strings that use a forward slash (/) to separate the levels in the hierarchy. A design best practice is that MQTT topic levels structure follows a general to a specific pattern. As the topic scheme flows from left to right, the topic levels flow generally to specific. For example, this topic refers to a temperature sensor in room1: **sensor/temperature/room1**. Another type of sensor, such as a humidity sensor, for room1 could be **sensor/humidity/room1**.
- The message broker maintains a list of all client sessions and the subscriptions for each session. When a message is published on a topic, the broker checks for sessions with subscriptions that map to the topic. The broker then forwards the published message to all sessions that have a currently connected client.
- The message broker supports the use of the **MQTT** protocol to publish and subscribe and the **HTTPS** protocol to publish. MQTT is a lightweight publish/subscribe communication protocol commonly used by resource-constrained devices. Both MQTT and HTTPS are supported through IPv4 and IPv6. The message broker also supports MQTT over the WebSocket protocol.

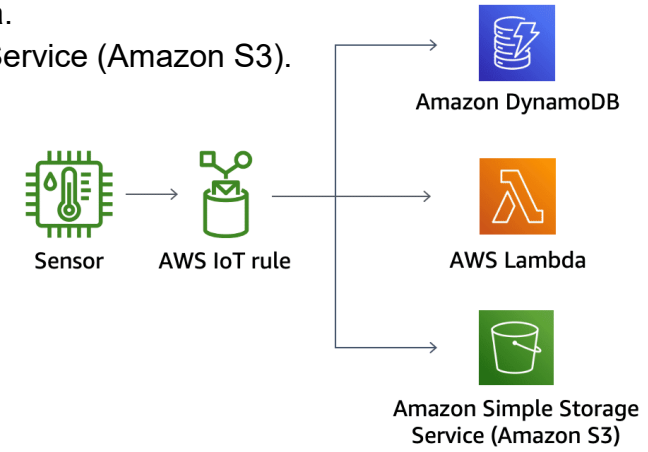
AWS IoT Core: Registry

- AWS IoT provides a registry that helps you manage things and organizes the resources associated with each device in the AWS Cloud. The registry is a **catalog of static metadata** and attributes about the devices, such as **serial numbers, manufacturer, firmware version, internal identifiers, and device capabilities**. You register your devices and associate up to three custom attributes with each one.
- You don't need to create an AWS IoT thing in the registry to connect a device to AWS IoT. However, by using the registry, you can search for and manage devices using the AWS IoT console, AWS IoT API, or the AWS Command Line Interface (AWS CLI).


```
{
    "version": 3,
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

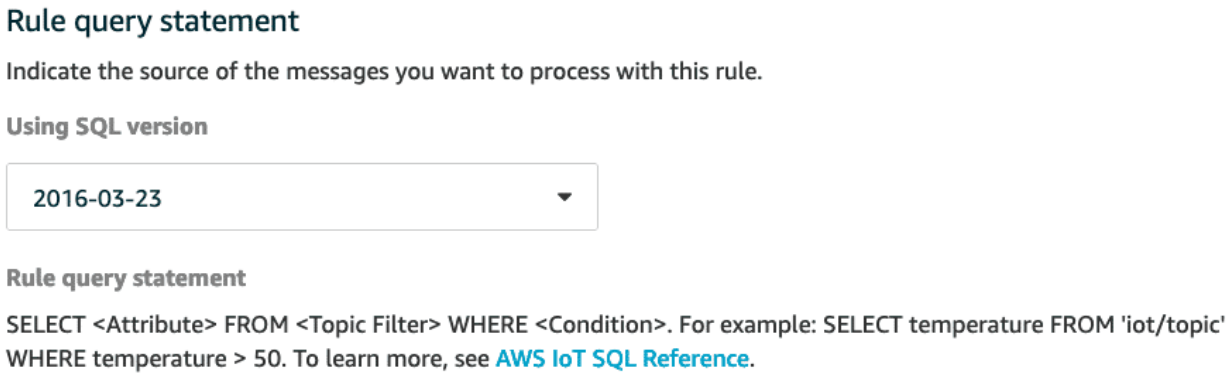
AWS IoT Core: Rules .

- Rules are one of the primary methods of **filtering and directing communication** from AWS IoT Core to other AWS services.
- By definition, a sensor will sense as long as it has power. Accumulating every bit of data from a sensor means that you will have a large amount of received data that is not useful for your business objectives. Because of the amount of data collected, you can **filter or direct data to different locations in the cloud**.
- By using rules, your devices can interact with AWS services. When data comes into the AWS IoT Core, rules are analyzed and actions are performed based on the MQTT topic stream. You can use rules to support tasks, such as the following:
  - Writing data from a device to an Amazon DynamoDB table.
  - Invoking an AWS Lambda function to extract specific data.
  - Saving a file or a set of data to Amazon Simple Storage Service (Amazon S3).



AWS IoT Core: Rules . .

- Rules have at a minimum a name, a query statement, and at least **one action to take**. You can also have additional information, such as a description, an error action (an action that is taken in the case of an error), and tags.
- Rule query statements in AWS IoT are written with a syntax that is similar to structured query language (SQL). The AWS IoT console presents a structured development environment to create query statements to filter and route MQTT messages.
- **Rule action**: The action for this rule is taking the selected messages and storing them in the Amazon S3 bucket called DOC-EXAMPLE-BUCKET.

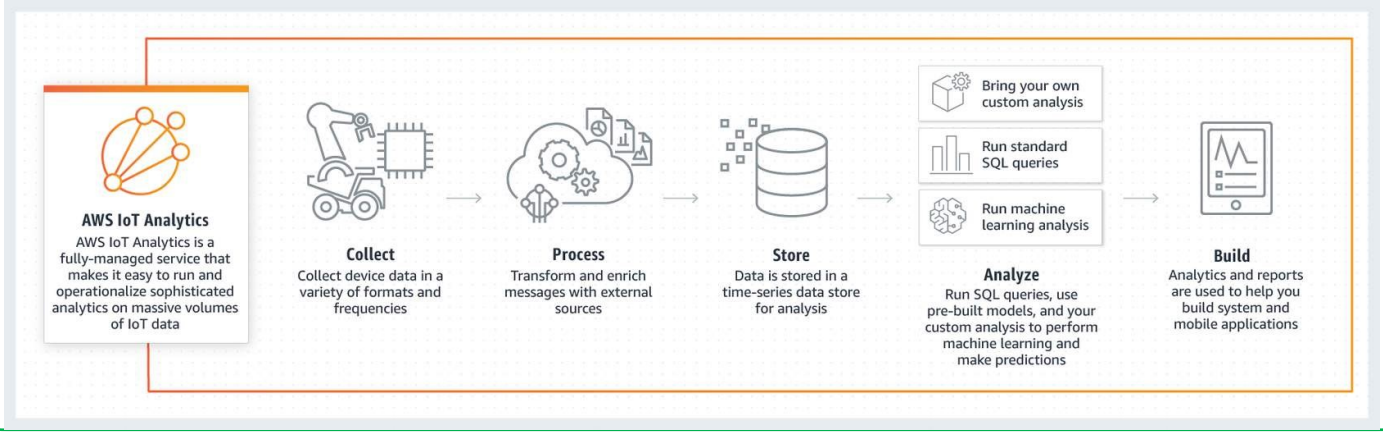


AWS IoT Analytics .

- **AWS IoT Analytics** automates the steps required to analyze data from AWS IoT devices. You configure the service to collect only the data you need from your devices, apply transformations to process the data, and enrich the data with device-specific metadata, such as device type and location, before storing it. Then, you can analyze your data by running queries using the built-in SQL query engine or perform more complex analytics and machine learning inference.
- AWS IoT Analytics terminology
  - **Channel** collects and archives raw, unprocessed message data before publishing this data to a pipeline.
  - **Pipeline** consumes messages from a channel and helps you to process and filter the messages before storing them in a data store.
  - **Data store** is not a database, but it is a scalable and queryable repository of your messages. You can have multiple data stores for messages that come from different devices or locations.
  - **Dataset** contains SQL statements and expressions that you use to query the data store along with an optional schedule that repeats the query at a day and time that you specify.
  - **Dataset contents** are the results of running your dataset. After you create dataset contents, you can view them from the AWS IoT console or by using the AWS IoT API or AWS Command Line Interface (AWS CLI).

AWS IoT Analytics . .

- Analysis Process
  - **Collect**: Begin by defining an AWS IoT Analytics **channel** and selecting the specific data to collect, such as temperature sensor readings.
  - **Process**: Configure AWS IoT Analytics **pipelines** to process your data. AWS IoT Analytics pipelines support transformations, such as Celsius to Fahrenheit conversion, conditional statements, message filtering, and message enrichment, using external data sources and AWS Lambda functions.
  - **Store**: After processing the data in the pipeline, AWS IoT Analytics **stores** it in an IoT-optimized data store for analysis.
  - **Analyze**: Query the data store by using the built-in **SQL query engine** in AWS IoT Analytics to answer specific business questions.
  - **Build**: Build **visualizations** and **dashboards** to get business insights quickly from your AWS IoT Analytics data using Amazon **QuickSight**.



AWS IoT Best Practices

- Best practices for AWS IoT things
  - Use **device IDs** and associate them with the collected and aggregated device data.
  - create the **timestamp** when data is collected instead of when it is synchronized.
- Best practices for AWS IoT security
  - Each device should be given a **unique certificate** and key pair. This gives you fine-grained management of the devices and helps with certificate revocation.
  - Devices must support **rotating and replacing certificates**. This ensures smooth operation and ease of management as certificates expire.
  - AWS IoT policies should follow a strategy of least privilege for permissions. To limit exposure, services **should publish and subscribe to only the topics that they require**.
  - The client ID and thing name on the device metadata should match in the **registry and Device Shadow** service so that there is **no conflict**.
- Best practices for IoT data
  - Before deployment, determine the **frequency of data sent from the device**. Sending data impacts energy consumption, network bandwidth, and the cost of the data sent by your device. Work backwards from your use case to determine the right frequency to sample your data and make that parameter configurable.
  - Data visualization should reflect the KPIs (key point initiatives) defined by **business objectives**. This way, the graphs or images will directly reflect the business objectives and questions.