Internet of Things

IoT Cloud Services and AWS

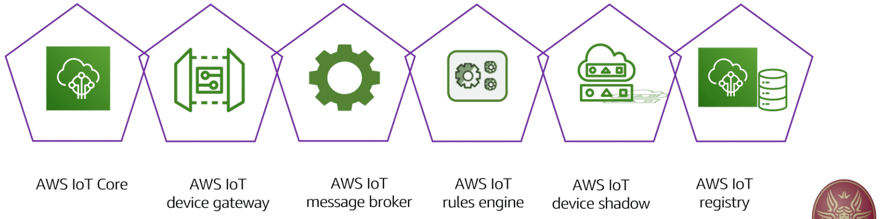
Ioannis Chatzigiannakis

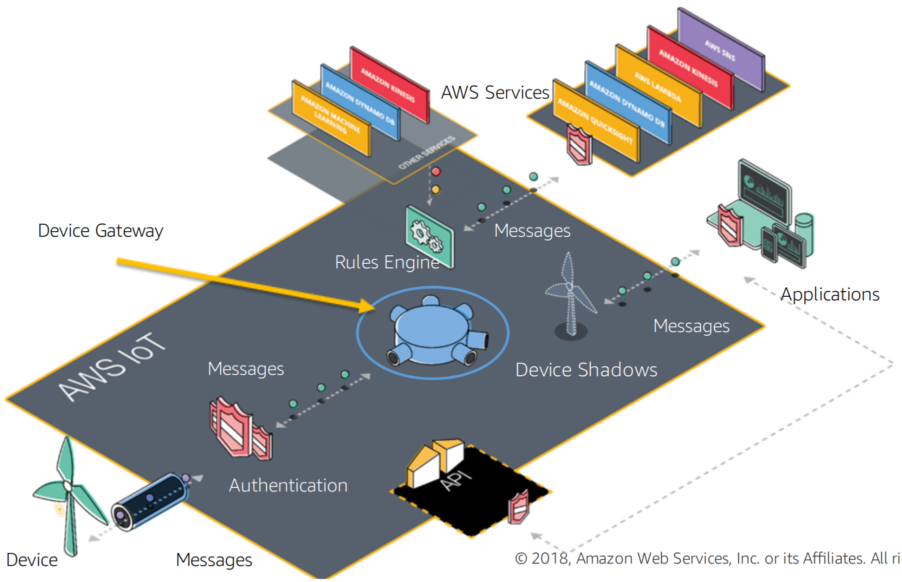
Sapienza University of Rome
Department of Computer, Control, and Management Engineering (DIAG)

Lecture 10:
IoT Cloud Services and AWS

AWS IoT Services

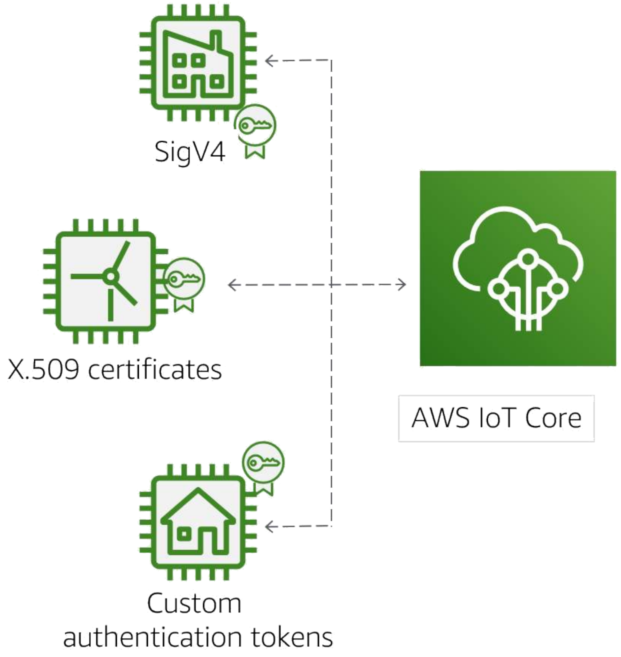
- Secure, bidirectional communication between internet-connected devices:
  - sensors, actuators, embedded microcontrollers, smart appliances, . . .
  - AWS Cloud,
  - Internet.
- Collect, store, and analyze telemetry data from multiple devices.
- Six main components:





Device Identification

- Sensor devices must be identified in order to access the AWS IoT services.
- Authentication is based on pre-deployed certificates.
- Flexible authentication options:
  1. Certificates for mutual authentication by using Message Queuing Telemetry Transport (MQTT) over Transport Layer Security (TLS) v1.2
  2. SigV4 over HTTP
  3. MQTT over WebSockets, which is similar to other AWS services.
- Ensure your devices are TLSv1.2 compliant
  - Not all devices support TLS v1.2.
  - TLS v1.2 ensures security and confidentiality of data exchange.
- Custom authentication tokens provided by our authentication/authorization service also supported.



Authorization & Access management

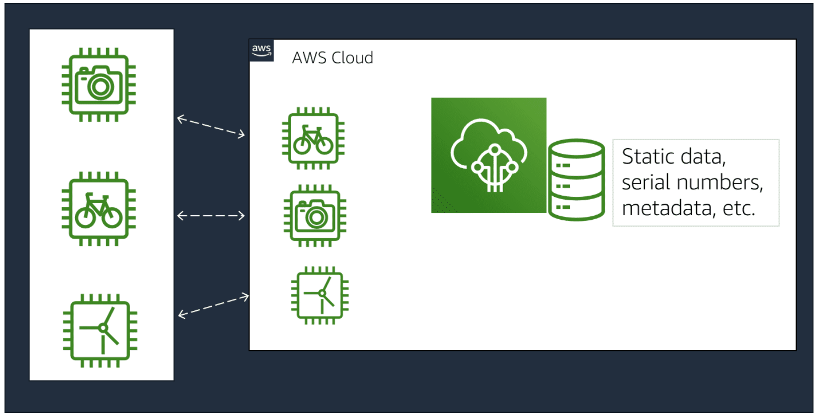
- Authorization is the process of granting permissions to an authenticated identity.
- Fine-grained access control for each User/Device/Service.
- Each device can have different access rules.
- Policies defined using JavaScript Object Notation (JSON).
  - Effect – Allow or Deny.
  - Action - List of actions that the policy allows or denies.
    - iot:Connect – connect to the AWS IoT message broker.
    - iot:Subscribe – subscribe to an MQTT topic or topic filter.
    - iot:GetThingShadow – get a device's shadow.
  - Resource - Lst of resources to which the actions apply.

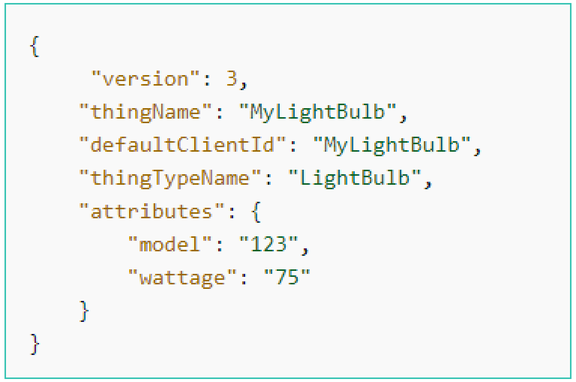
IoT Policy Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:eu-central-1:904534684829:topic/measurements"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:eu-central-1:904534684829:client/measurements"
    },
    {
      "Effect": "Deny",
      "Action": "iot:DeleteThingShadow",
      "Resource": "arn:aws:iot:eu-central-1:904534684829:thing/measurements"
    }
  ]
}
```

IoT registry

- Using the registry is optional.
- Helps you manage your device ecosystem effectively.
- A database of device properties, attributes and tags.
  - A catalog of static metadata.
  - Example: serial numbers, manufacturer, firmware version, . . .
  - Can also store the state of the device and the device shadow.
  - Can acts as a repository for device certificates.
- Fully managed and scales to over a billion devices.
- Enables to search for devices based on attributes and tags.



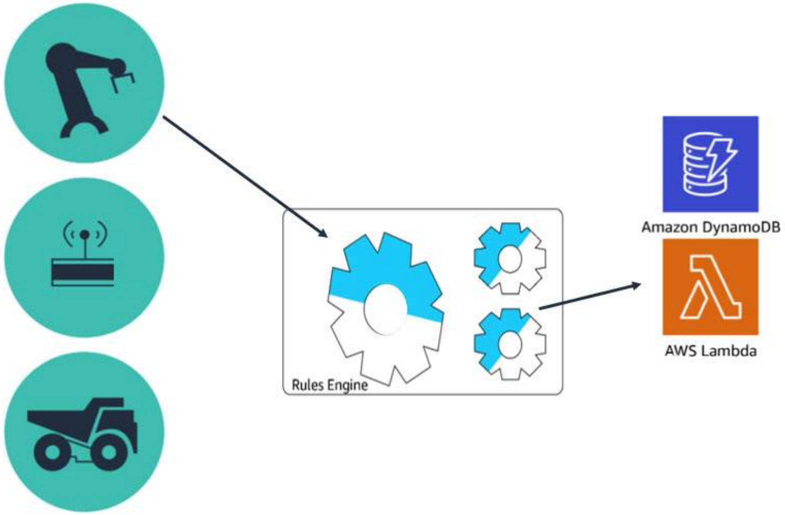


```
{
     "version": 3,
    "thingName": "MyLightBulb",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

AWS IoT message broker

- Processes and routes data from your devices into AWS IoT Core.
- Scalable, reliable, with low-latency, message routing.
- Uses the publish and subscribe model to decouple devices and applications.
- Allows two-way message streaming between devices and applications.
- Allows data transformation, rerouting, and enhancement with external data sources.
- Based on the Message Queuing Telemetry Transport (MQTT) version 3.1.1.
  - Supports MQTT Quality of Service (QoS) levels 0 and 1 only.

AWS IoT rules engine

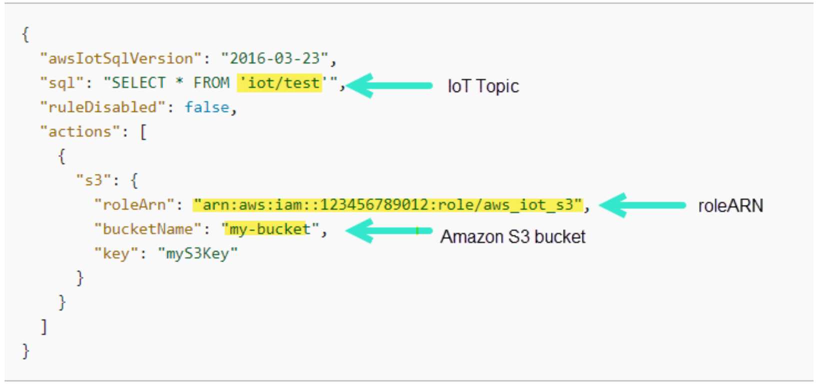


IoT Rules engine

- Sensor publish data continuously or periodically – raw data
- Depending on Variety/Velocity/Volume of raw data we might end up with Big Data.
- Usually not all raw data are useful.
- The rules engine listens for incoming messages that match a rule based on the MQTT topic stream:
  - Saving a file, or a set of data, to an Amazon Simple Storage Service (Amazon S3) bucket.
  - Writing data from a device to an Amazon DynamoDB database.
  - Invoke an AWS Lambda function to extract specific data.
  - Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
  - ...
- The rules allow devices to interact with AWS services.

Rules engine language

Uses SQL-like statements to filter and route MQTT messages.



```
{
  "awsIotSqlVersion": "2016-03-23",
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "actions": [
    {
      "s3": {
        "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
        "bucketName": "my-bucket",
        "key": "myS3Key"
      }
    }
  ]
}
```

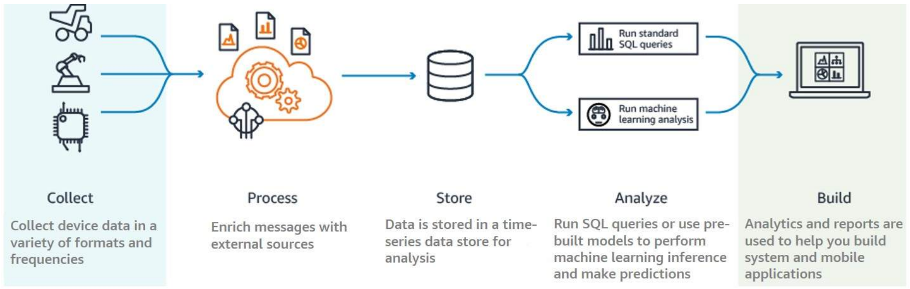
IoT Analytics

- Automates the steps required to analyze IoT data.
- Helps collect only the data you need from your devices.
- Apply transformations to process the data.
- Enrich the data with device-specific metadata, such as device type and location, before storing it.
- Analyze by running queries using the built-in SQL query engine,
- Perform more complex analytics and machine learning inference.

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 enables to process and filter the messages before storing them in a data store.
- Data store – not a database, but a scalable and queryable repository of messages. May have multiple data stores for messages that come from different devices or locations.
- Dataset – retrieve data from a data store by creating a dataset.
  - Enables you to create a SQL dataset or a container dataset.
  - Allows to view dataset contents from the console.

Closer analysis of the process



Run standard SQL queries

Run machine learning analysis

Collect

Collect device data in a variety of formats and frequencies

Process

Enrich messages with external sources

Store

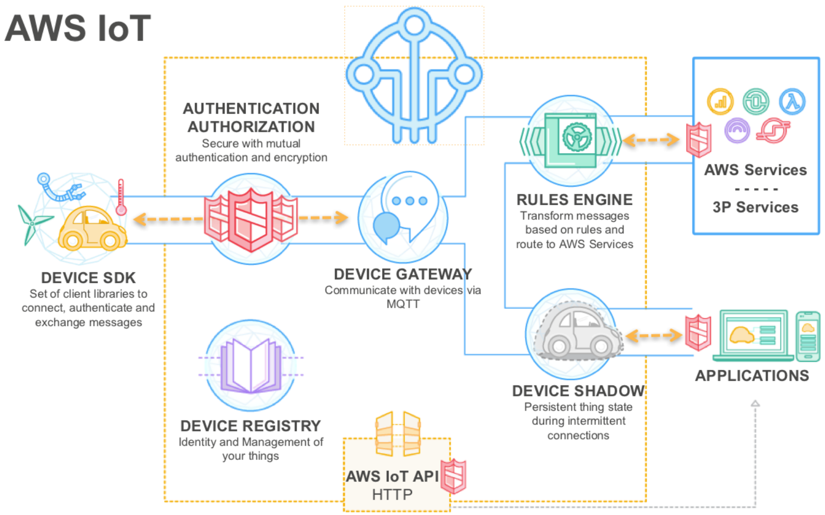
Data is stored in a time-series data store for analysis

Analyze

Run SQL queries or use pre-built models to perform machine learning inference and make predictions

Build

Analytics and reports are used to help you build system and mobile applications



AWS IoT

AUTHENTICATION AUTHORIZATION

Secure with mutual authentication and encryption

DEVICE SDK

Set of client libraries to connect, authenticate and exchange messages

DEVICE GATEWAY

Communicate with devices via MQTT

RULES ENGINE

Transform messages based on rules and route to AWS Services

AWS Services

- - - - -

3P Services

DEVICE SHADOW

Persistent thing state during intermittent connections

APPLICATIONS

DEVICE REGISTRY

Identity and Management of your things

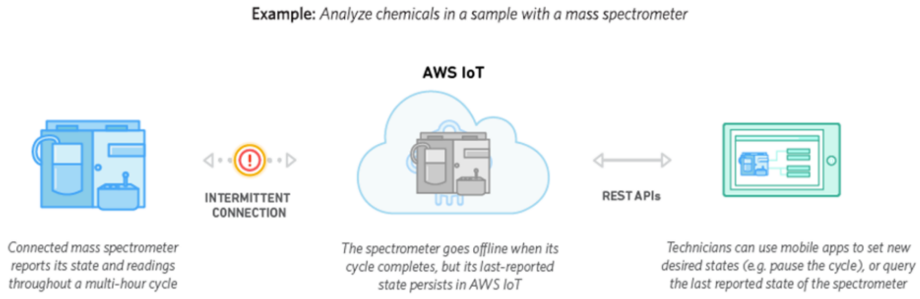
AWS IoT API

HTTP

Device Shadow service

- A Digital Twin.
- Maintains a shadow for each device you connect to AWS IoT.
- Interact directly with the Digital Twin to get/set state over MQTT or HTTP.
  - If Actual Device is connected, changes are propagated.
  - If Actual Device is not connected, changes are kept by Shadow service and propagated when device reconnects.
- Applications are not aware of the connection status of each IoT device.

An Example of Device Shadow Usage



Example: *Analyze chemicals in a sample with a mass spectrometer*

AWS IoT

INTERMITTENT CONNECTION

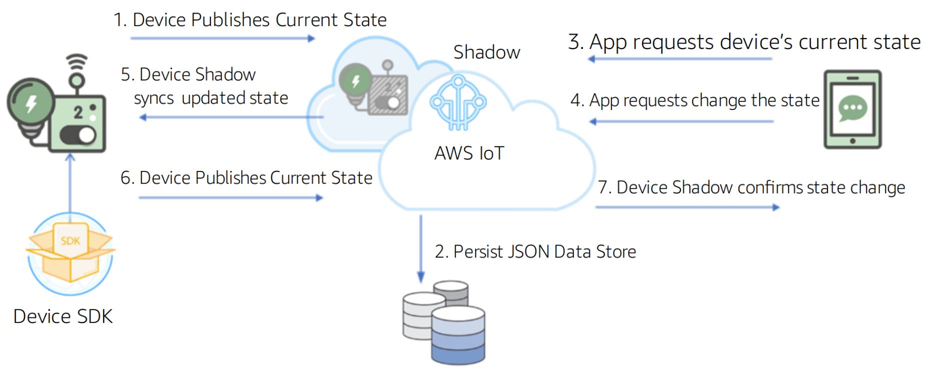
REST APIs

Connected mass spectrometer reports its state and readings throughout a multi-hour cycle

The spectrometer goes offline when its cycle completes, but its last-reported state persists in AWS IoT

Technicians can use mobile apps to set new desired states (e.g. pause the cycle), or query the last reported state of the spectrometer

Device Shadow Lifecycle


1. Device Publishes Current State
Shadow
3. App requests device's current state
5. Device Shadow syncs updated state
4. App requests change the state
AWS IoT
6. Device Publishes Current State
7. Device Shadow confirms state change
SDK
2. Persist JSON Data Store
Device SDK

Δ, Desired and Reported States

Thing

- Reports the current state to the device's shadow
- Retrieves desired state from shadow

Device Shadow

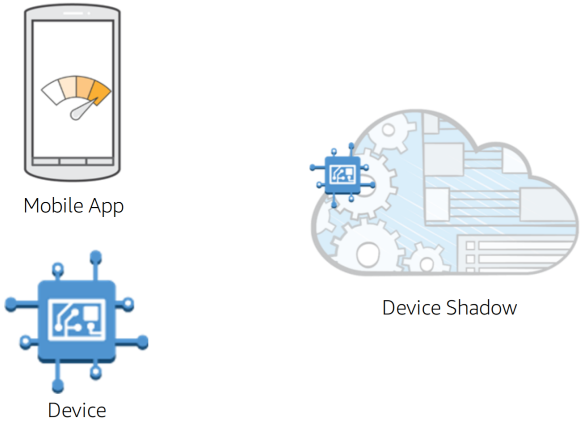
- Coordinates and synchronizes shadow document updates
- Publishes update events on related shadow topics

Mobile App

- Sets the desired state of a device
- Gets the last reported state of the device

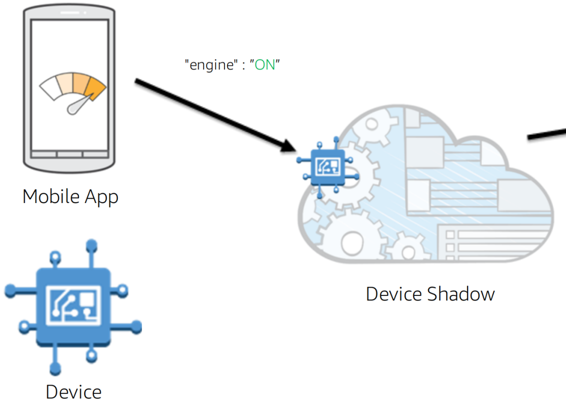
```
{
"state" : {
            "desired" : {
                 "lights": {
"color": "RED" },
                 "engine" : "ON"
            },
            "reported" : {
                 "lights" : {
"color": "GREEN"  },
           "engine" : "ON"
            },
            "delta" : {
                 "lights" : {
"color": "RED"  }
             } },
"version" : 10

}
```

An Example



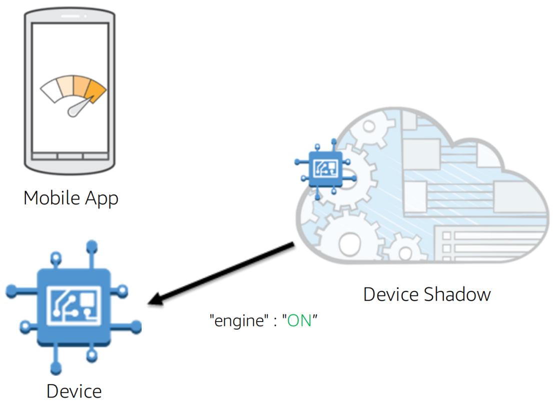
```
{
"state" : {
        "desired" : {
                                   "engine" : ""

        },
        "reported" : {
                                   "engine" : "",


        },
"version" : ,
"clientToken" : "UniqueClientToken",
"timestamp":
}
```

An Example



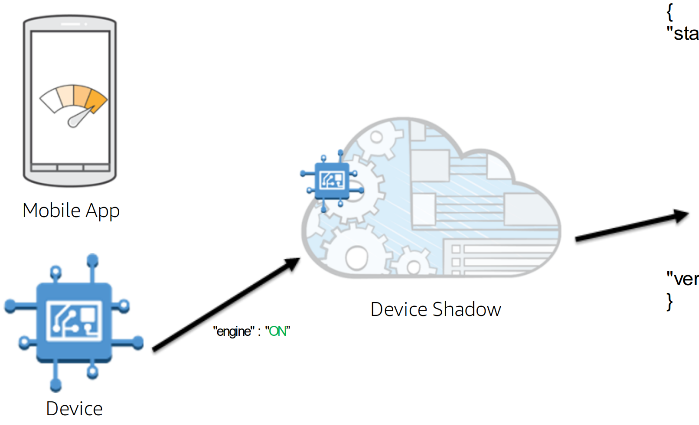
```
{
"state" : {
        "desired" : {
                "engine" : "ON",
                }

        },
        "reported" : {
                "engine" : "OFF",
                }


        },
"version" : 10
}
```

An Example



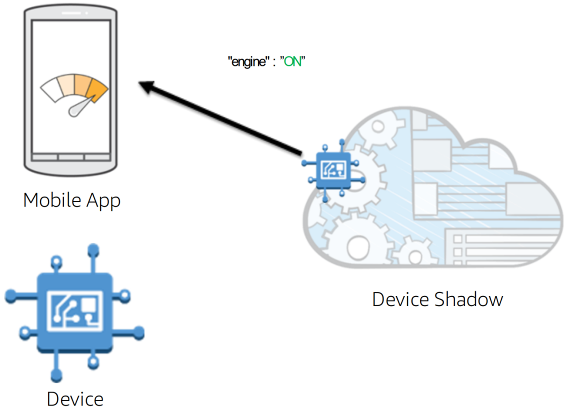
```
{
"state" : {
        “desired" : {
                "engine" : "ON”,
                        }

        },
        "reported" : {
                "engine" : ”OFF”,
                        }

        },

         ”delta” : {
                "engine" : “ON”,
                        }
         },

"version" : 10
}
```

An Example



```
{
"state" : {
        “desired" : {
                "engine" : "ON”,
                        }

        },
        "reported" : {
                "engine" : ”ON”,
                        }

        },
"version" : 10
}
```

An Example



```
{
"state" : {
        “desired" : {
                "engine" : "ON”,
                        }

        },
        "reported" : {
                "engine" : ”ON”,
                        }

        },
"version" : 10
}
```

Interacting with the Device Shadow

Each device is assigned with 4 MQTT topics:

- $aws/things/ThingName/shadow/update
- $aws/things/ThingName/shadow/get
- $aws/things/ThingName/shadow/get/accepted
- $aws/things/ThingName/shadow/delete
- $aws/things/ThingName/shadow/update/delta