Hands-on with AWS IoT

Julien Simon
Principal Technical Evangelist
Amazon Web Services

julsimon@amazon.fr
@julsimon

amazon web services | Pop-up Loft TEL AVIV

Agenda

- Overview of AWS IoT
- Devices & SDKs, with a focus on the Arduino Yún
- The MQTT protocol
- Creating and securing “things”
- Processing data and sending it to other services
- Debugging
- And of course, demos and complete instructions to reproduce them. Nothing will be hidden ☺

amazon web services

Overview of AWS IoT

amazon web services

AWS IoT is a fully managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices.

1

Securely connect and manage any physical device across multiple networks and protocols

2

Extract and filter data from your devices and take action with custom rules

3

Create web and mobile applications that interact with devices reliably at any time

Device SDK

Device Security and Policy Management

Device Gateway

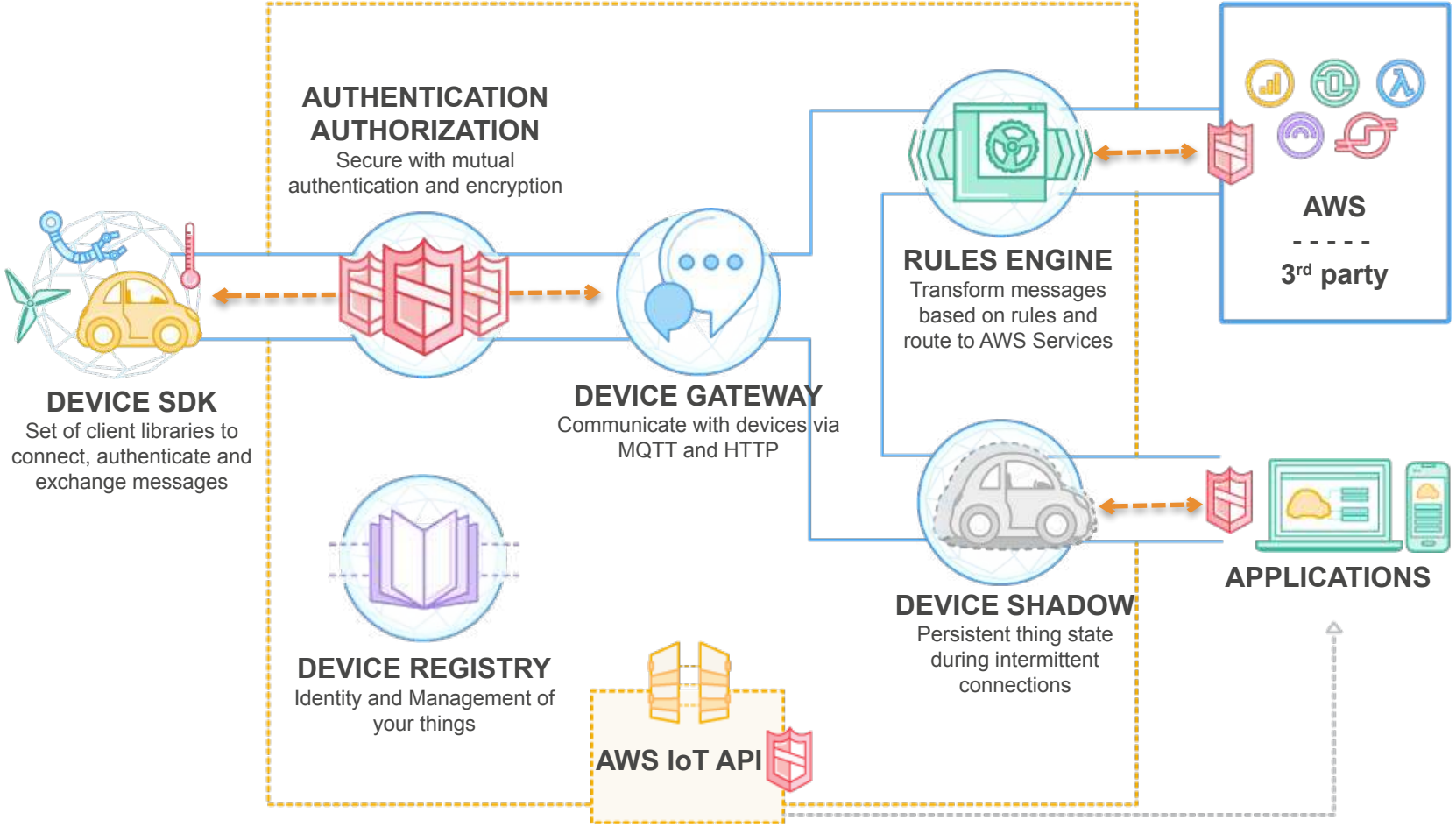
Registry

Rules Engine

Shadow

amazon web services

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 and HTTP
RULES ENGINE
Transform messages based on rules and route to AWS Services
AWS
3rd party
DEVICE SHADOW
Persistent thing state during intermittent connections
APPLICATIONS
DEVICE REGISTRY
Identity and Management of your things
AWS IoT API

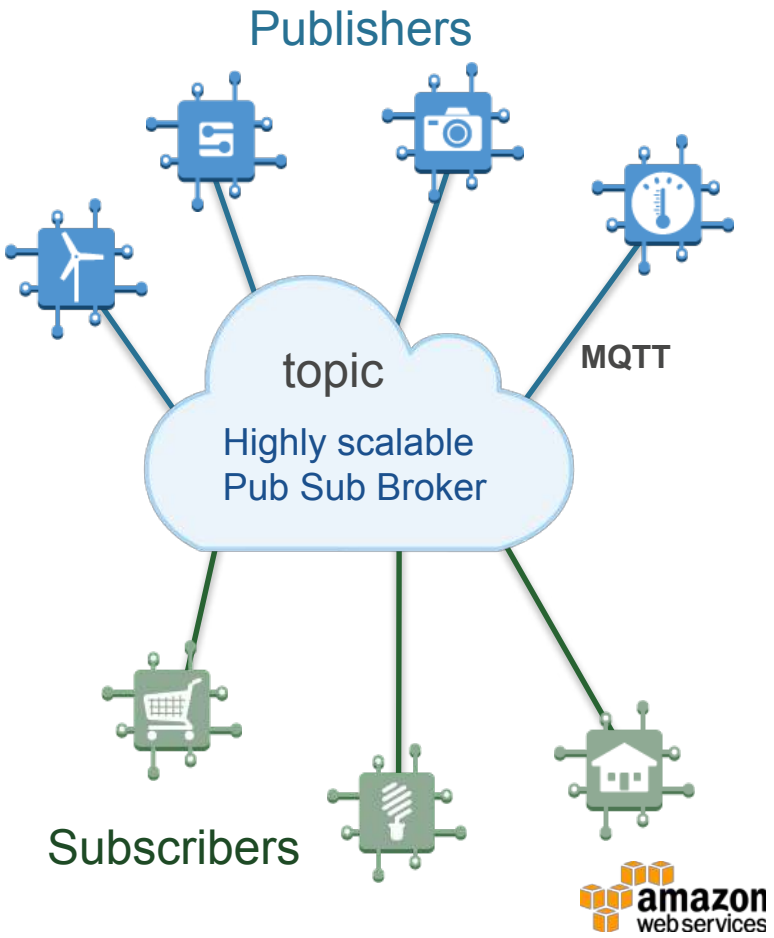
AWS IoT: Securely Connect Devices

Device Registry
Cloud alter-ego of a physical device.
Persists metadata about the device.

Multi-protocol Message Gateway
Millions of devices and apps can connect over MQTT or HTTP

Elastic Publish Subscribe Broker
Go from 1 to 1-billion long-lived connections with zero provisioning

Secure by Default
Connect securely via X509 Certs and TLS v1.2 Client Mutual Auth



AWS IoT: Rules Engine

- Rules Engine evaluates inbound messages published into AWS IoT, transforms and delivers to the appropriate endpoint based on business rules.

- External endpoints can be reached via AWS Lambda and Amazon Simple Notification Service (SNS).

Actions

Invoke a Lambda function

Put object in an S3 bucket

Insert, Update, Read from a DynamoDB table

Publish to an SNS Topic or Endpoint

Publish to a Kinesis stream /

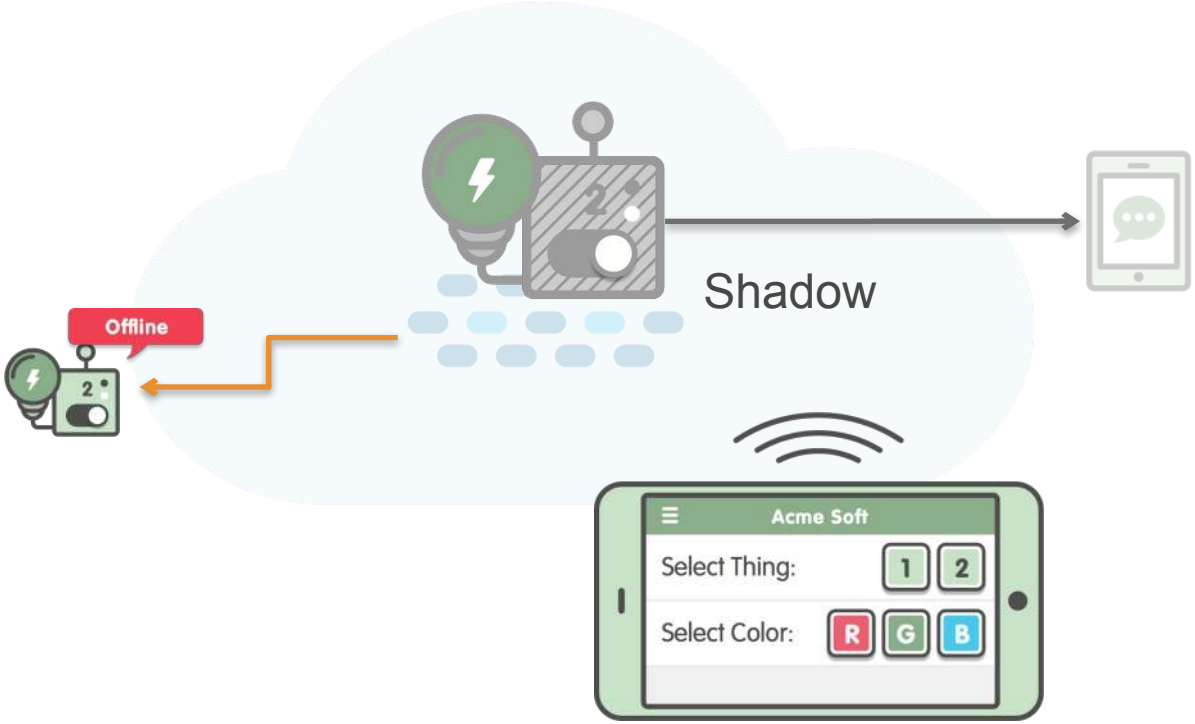
Amazon Kinesis Firehose

Republish to AWS IoT

AWS IoT: Device Shadow

Virtual representation of your device in the cloud

- **Device State**
  - desired
  - reported
- **Device metadata**
  - **Sensors**
- **Version**
- **clientToken**
- **timestamp**



Devices & SDKs

amazon web services

Official AWS IoT Starter Kits

AVNET BROADCOM

MARVELL

RENESAS

TEXAS INSTRUMENTS

MICROCHIP

intel

MEDIATEK

ARROW

QUALCOMM

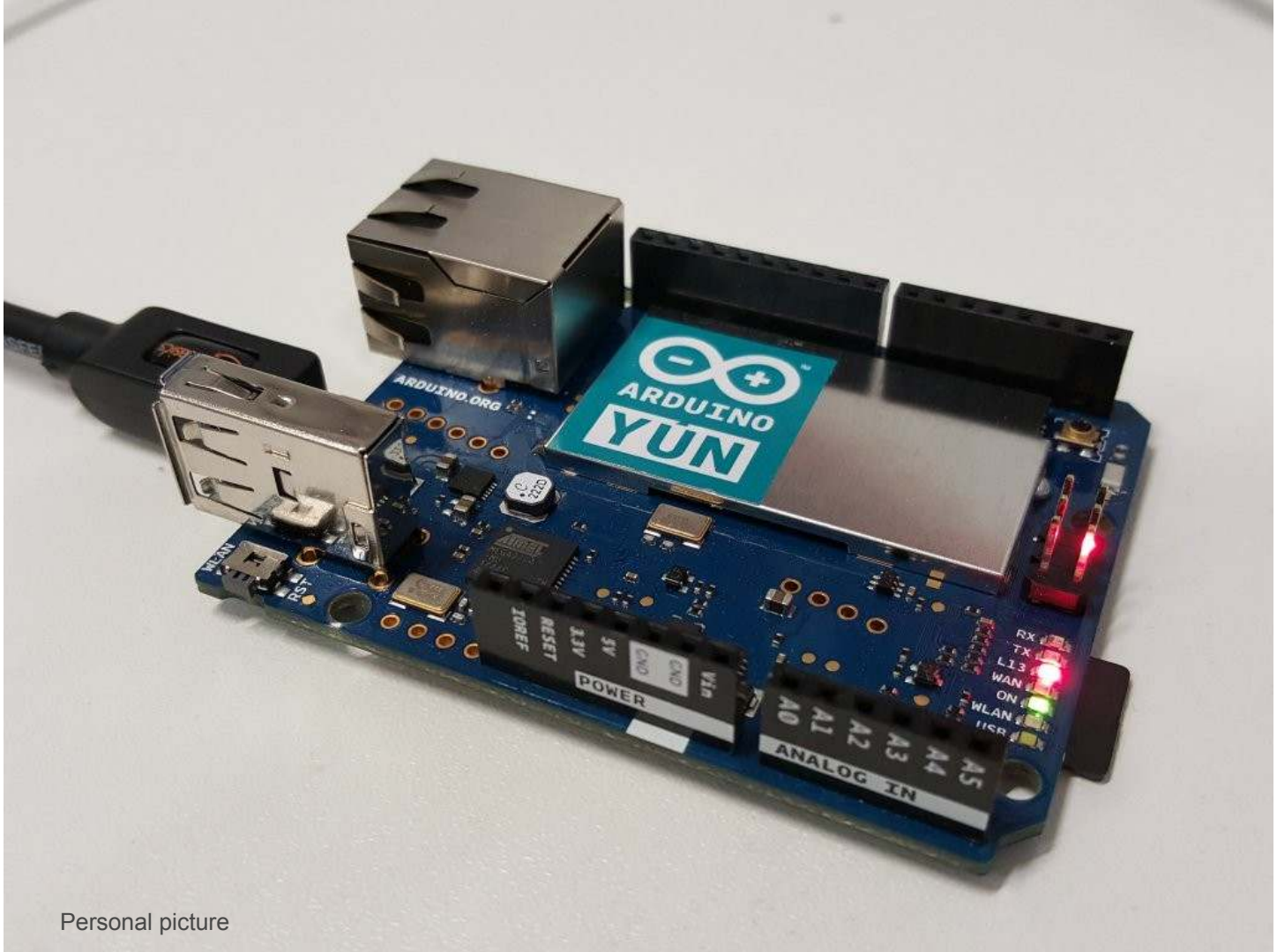
seeed
Grow the Difference

seeed
Grow the Difference

amazon
web services

Device SDKs

- Arduino: Arduino Yún platform
- Node.js: ideal for Embedded Linux
- C: ideal for embedded OS



Personal picture

Arduino Yun ATmega32u4 Microcontroller Board A000008

by Arduino Org

$65.66 ~~$74.95~~ Prime

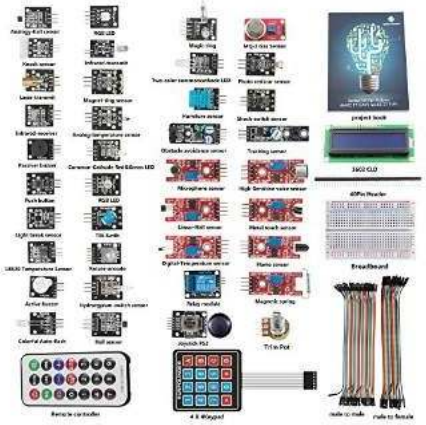
Get it by **Monday, Mar 21**

More Buying Choices
$65.00 new (17 offers)
$59.99 used (1 offer)

★★★★☆ 68

FREE Shipping on eligible orders

Electronics: See all 153 items



SunFounder 37 modules Arduino Sensor Kit for Arduino UNO R3 Mega2560 Mega328 Nano (without controller)

by SunFounder

$68.99 Prime

Get it by **Monday, Mar 21**

More Buying Choices
$68.99 new (64 offers)

★★★★½ 92

FREE Shipping on eligible orders

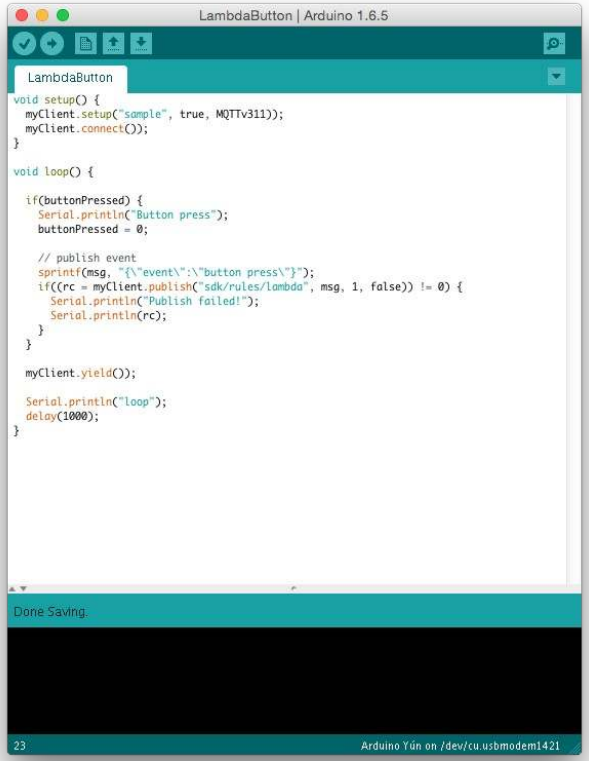
Electronics: See all 76 items

Not an official endorsement by AWS. Just a personal preference ☺

amazon web services

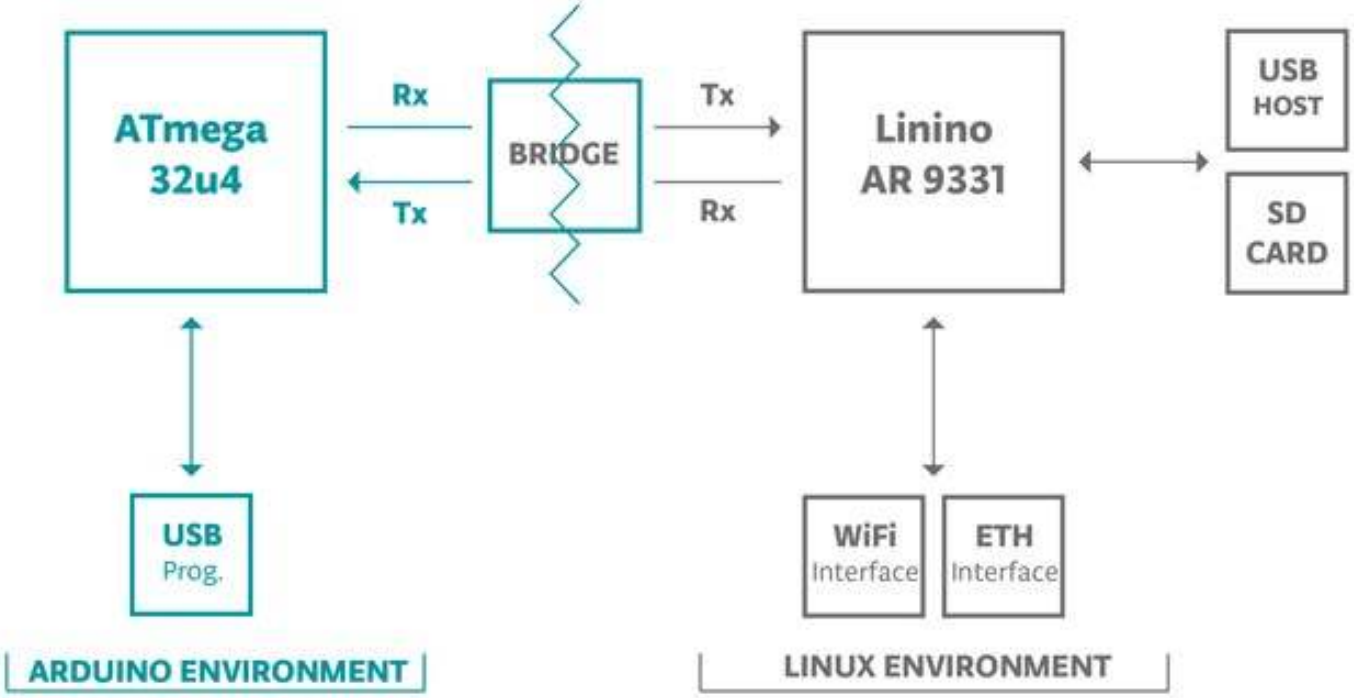
Arduino Yún SDK

- Arduino IDE
- Libraries
- Hardware Ecosystem



amazon web services

Arduino Yún hardware



The MQTT protocol

amazon web services

MQTT Protocol

- OASIS standard protocol (v3.1.1)
- Lightweight, transport protocol that is useful for connected devices
- **Publish-subscribe** with **topics**

- MQTT is used on oil rigs, connected trucks, and many more critical applications
- Customers have needed to build, maintain and scale a broker to use MQTT with cloud applications

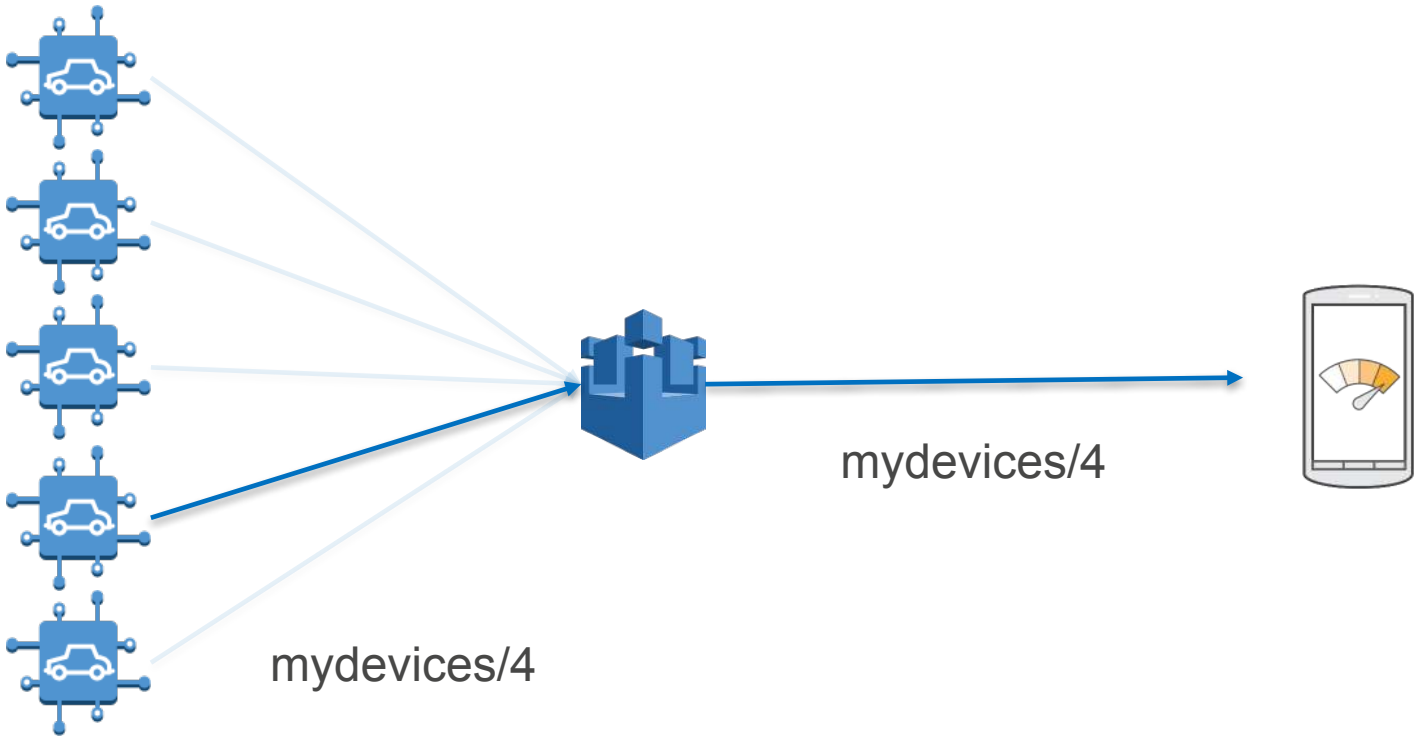
MQTTS vs HTTPS:

- 93x faster throughput
- 11.89x less battery to send
- 170.9x less battery to receive
- 50% less power to stay connected
- 8x less network overhead

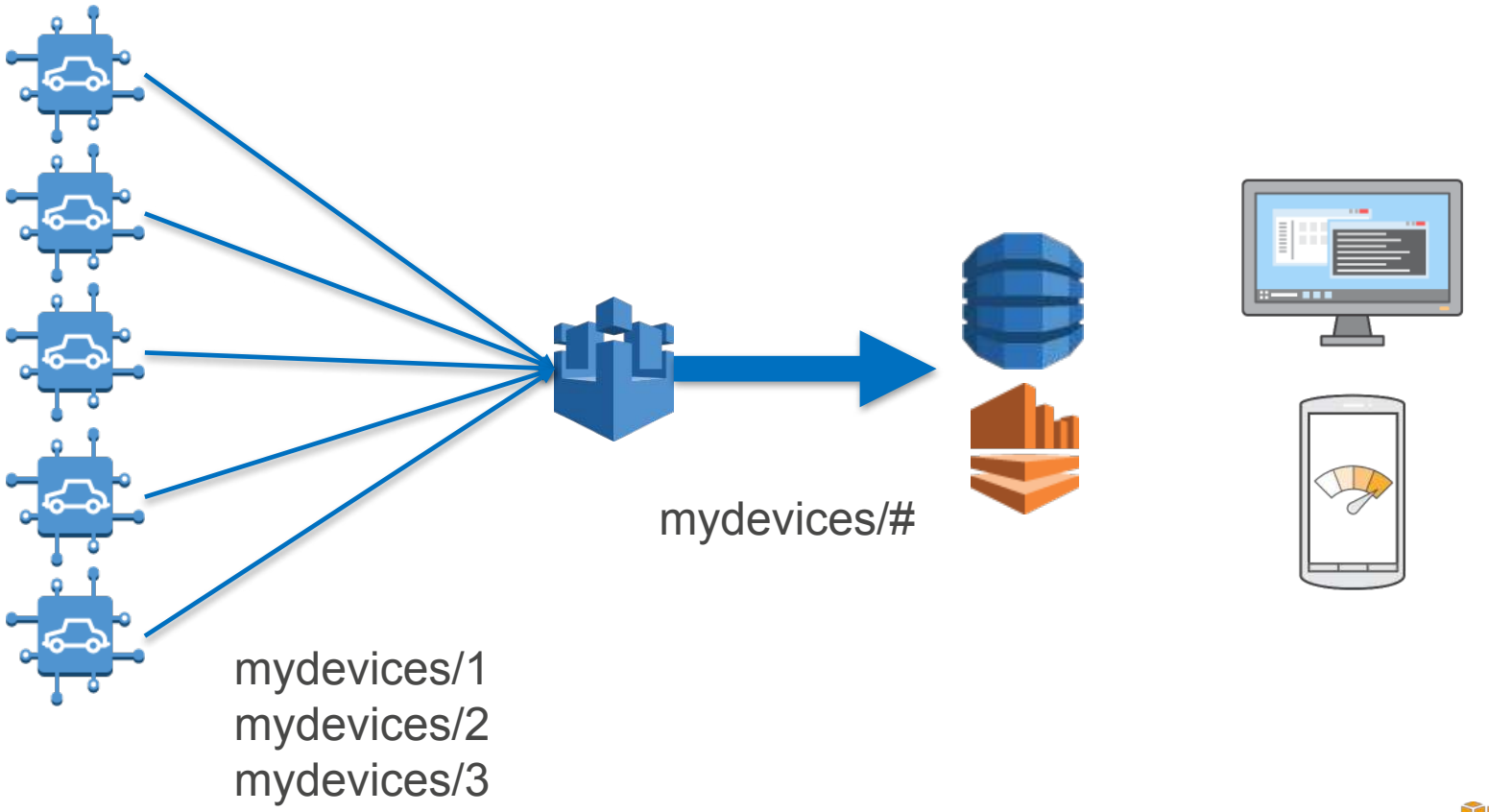
Source: http://stephendnicholas.com/archives/1217

amazon web services

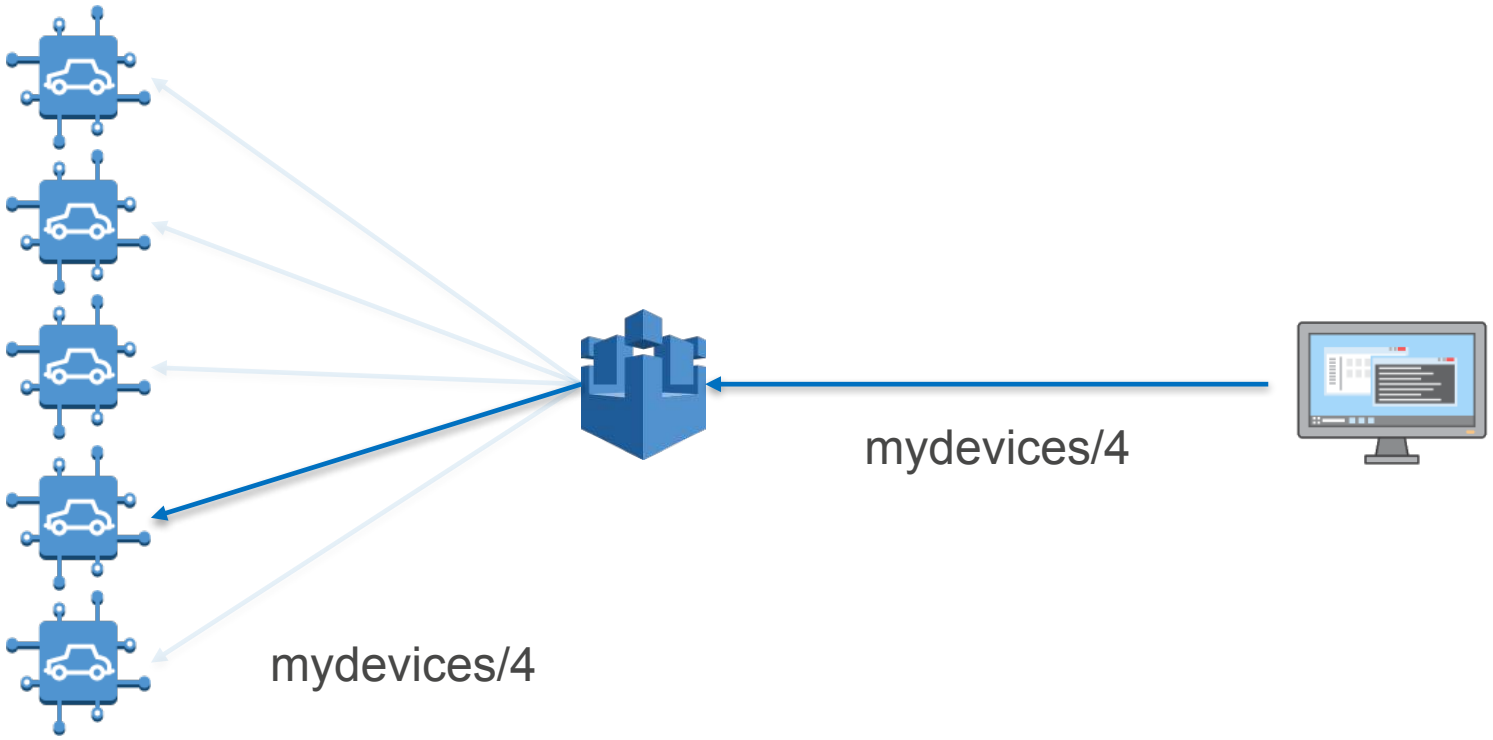
MQTT: collect data from a device



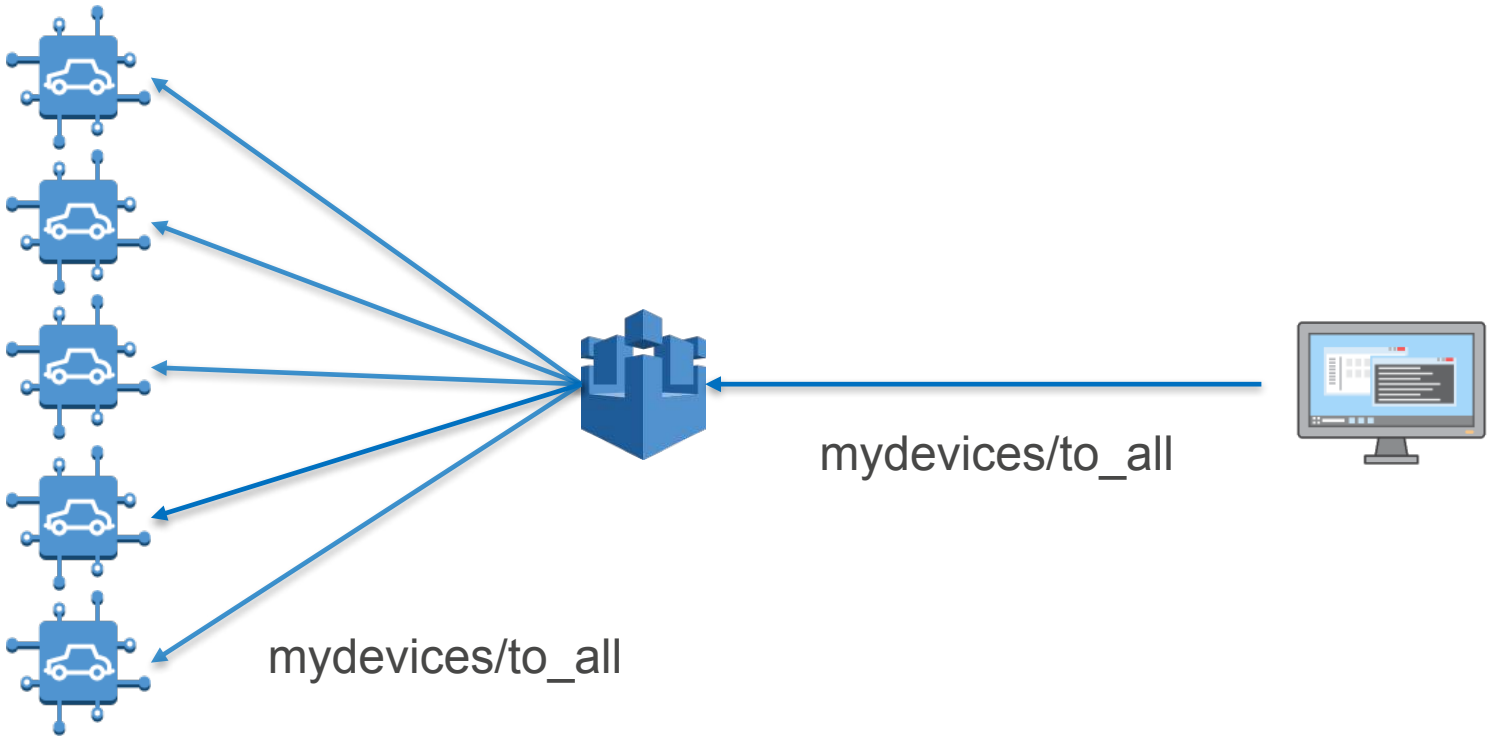
MQTT: aggregate data from many devices



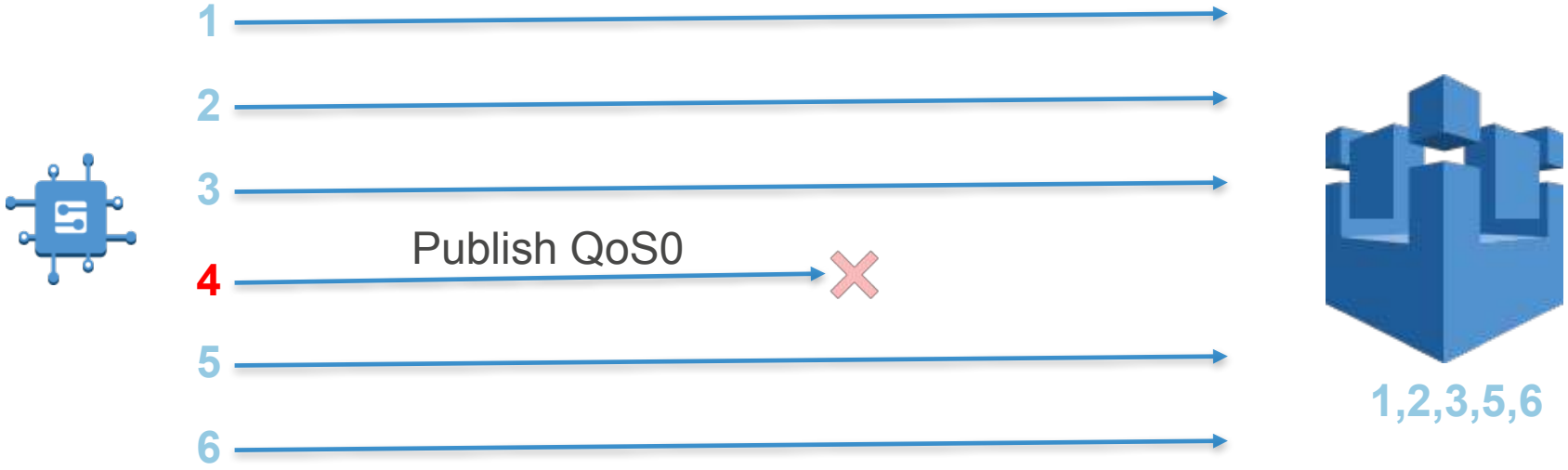
MQTT: update a device



MQTT: update all devices



MQTT: QoS 0 (at most once)



MQTT: QoS 1 (at least once)

1

2

3

PUBLISH QoS1

4

5

PUBLISH QoS1

4

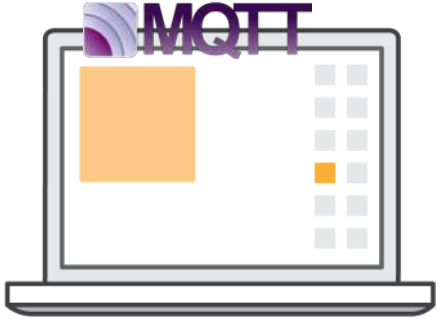
PUBACK

6

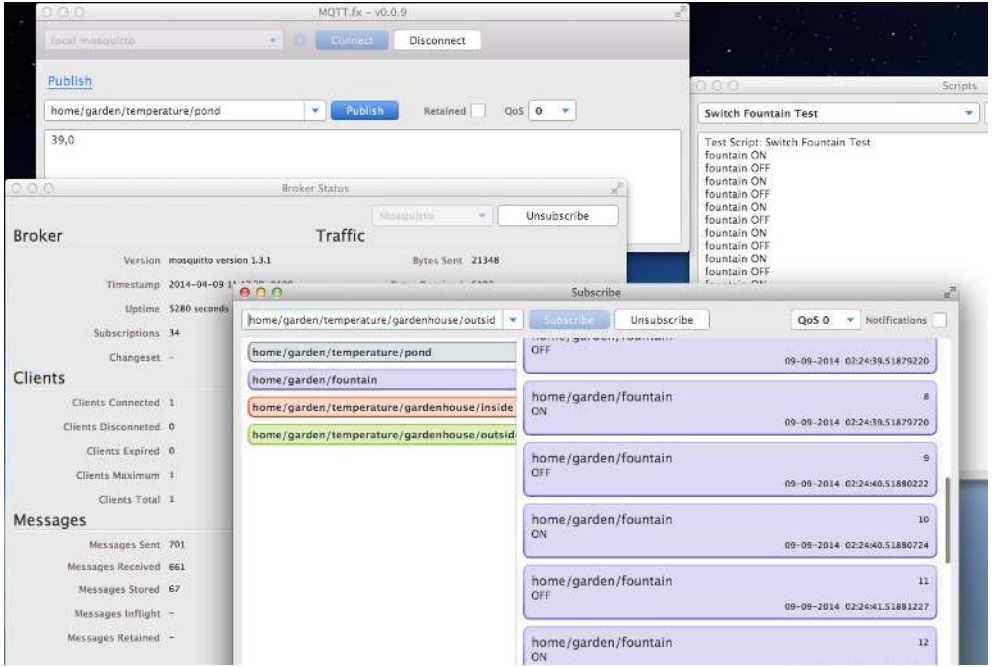
1,2,3,4,5,6

amazon web services

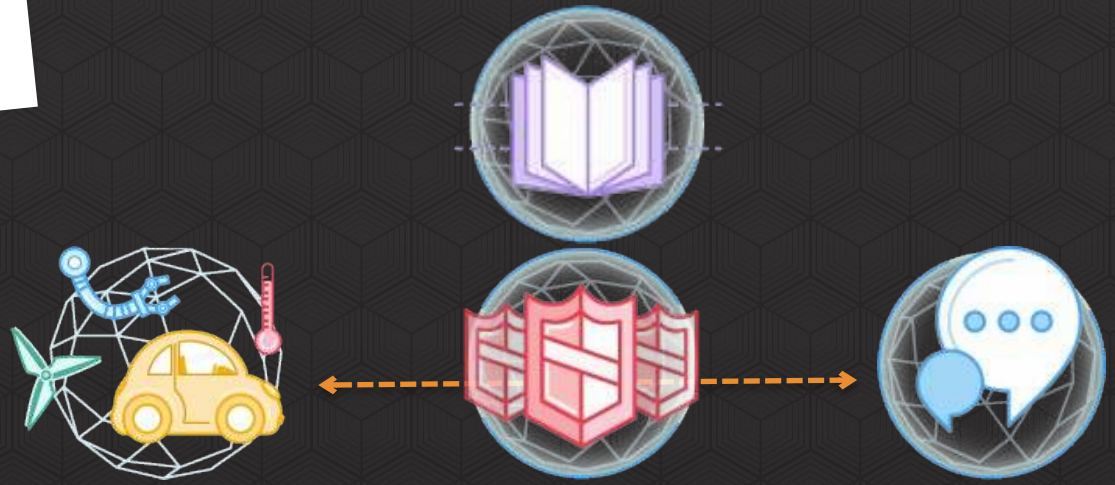
Demo : MQTT Publish / Subscribe with MQTT.fx



http://mqttfx.jfx4ee.org/



amazon web services



Things

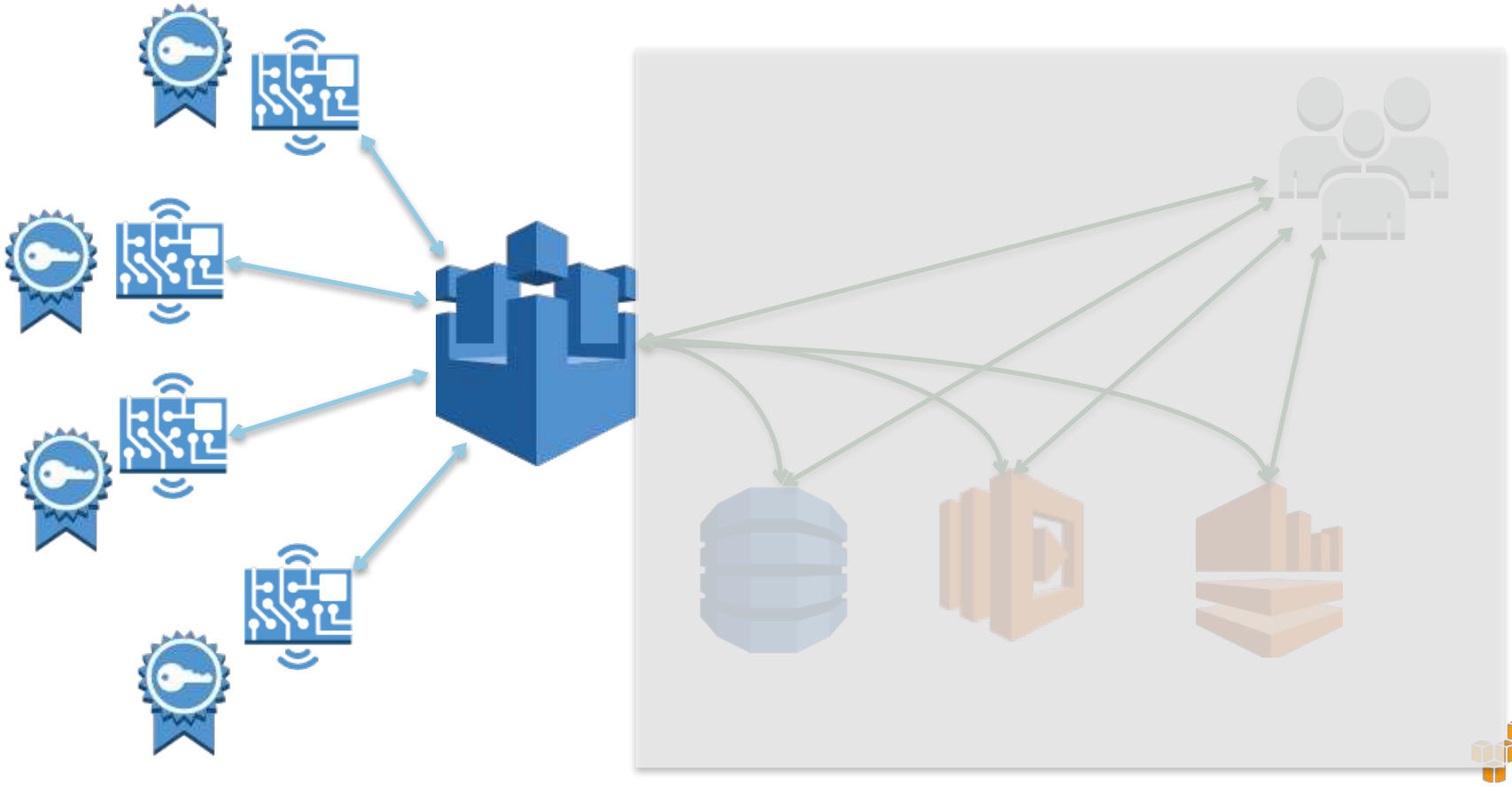
amazon web services

Requirements

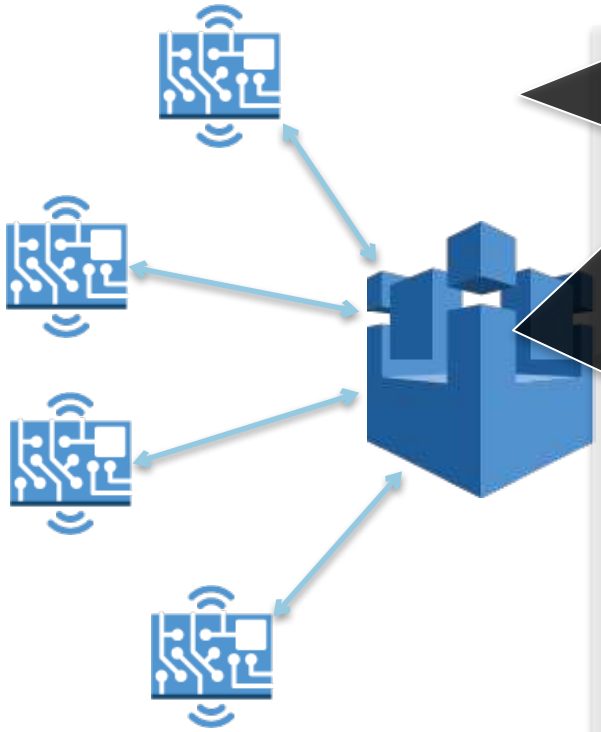
- Secure Identity for Things
- Secure Communications with Things
- Fine-grained Authorization for:
  - Thing Management
  - Pub/Sub Data Access
  - AWS Service Access

amazon web services

Creating Things



Assigning Policies to Things



```
{
  "Version":"2012-10-17",
  "Statement":[ {
        "Effect":"Allow",
        "Action":[ "iot:Connect" ],
        "Resource":"*"
      }, {
        "Effect":"Allow",
        "Action":[ "iot:Publish" ],
        "Resource":[
          "arn:aws:iot:us-east-1:123456972007:
             topic/$aws/things/MyThing/shadow/update"]
      }, {
        "Effect":"Allow",
        "Action":[ "iot:Subscribe", "iot:Receive" ],
        "Resource":[
          "arn:aws:iot:us-east-1:123456972007:
             topicfilter/$aws/things/MyThing/shadow/*"
        ]
      }
  ]
}
```

Creating a thing

```
% aws iot create-thing --thing-name myThing

% aws iot describe-thing --thing-name myThing

% aws iot list-things
```

Creating a certificate and keys

```
% aws iot create-keys-and-certificate
--set-as-active
--certificate-pem-outfile cert.pem
--public-key-outfile publicKey.pem
--private-key-outfile privateKey.pem
```

Don’t forget to install these on your device, e.g. https://github.com/aws/aws-iot-device-sdk-arduino-yun

amazon web services

Creating a policy

```
% cat myPolicy.json
{
   "Version": "2012-10-17",
   "Statement": [{ "Effect": "Allow", "Action":
["iot:*"],
   "Resource": ["*"] }]
}
```

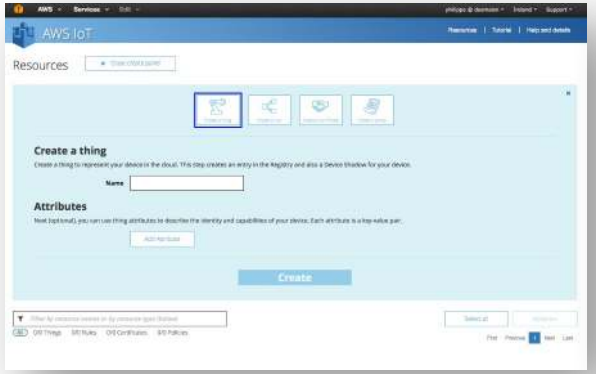
```
% aws iot create-policy
--policy-name PubSubToAnyTopic
--policy-document file://myPolicy.json
```

Assigning an identity to a Policy and a Thing

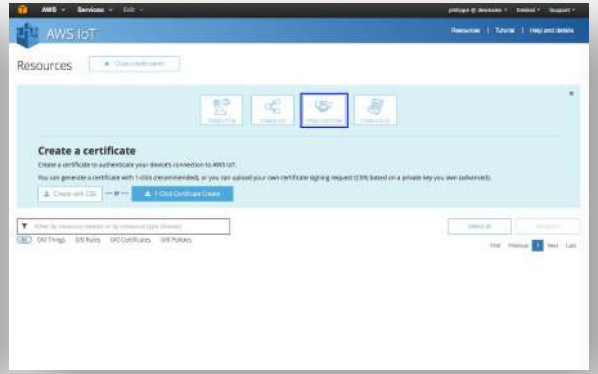
```
% aws iot attach-principal-policy
--policy-name PubSubToAnyTopic
--principal CERTIFICATE_ARN

% aws iot attach-thing-principal
--thing-name myThing
--principal CERTIFICATE_ARN
```

amazon web services

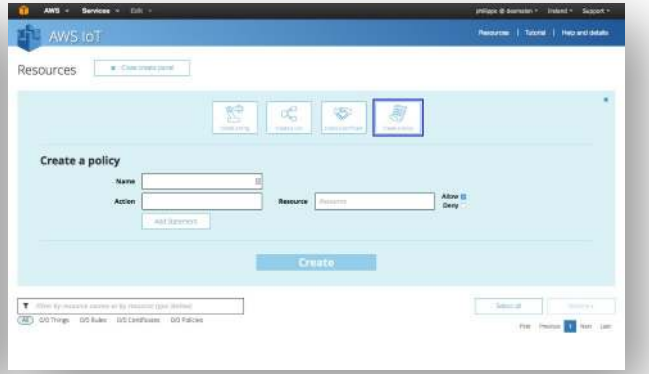
Console mode



Create a Thing



Create a Certificate



Create/Attach a Policy

amazon web services

Demo : MQTT Publish / Subscribe with Arduinos

Yun1

Publish to *topic1*

Subscribe to *topic2*

Yun2

Subscribe to *topic1*

Publish to *topic2*

Subscribe to *topic1* and *topic2*

amazon web services

Connecting to AWS IoT

```
aws_iot_mqtt_client myClient;

if((rc = myClient.setup(AWS_IOT_CLIENT_ID)) == 0) {
  // Load user configuration
  if((rc = myClient.config(AWS_IOT_MQTT_HOST,
AWS_IOT_MQTT_PORT, AWS_IOT_ROOT_CA_PATH,
    AWS_IOT_PRIVATE_KEY_PATH, AWS_IOT_CERTIFICATE_PATH)) == 0) {
        if((rc = myClient.connect()) == 0) {
            // We are connected
            doSomethingUseful();
        }
    }
}
```

amazon web services

Subscribing and publishing to a topic

```
if ((rc=myClient.subscribe(”myTopic", 1, msg_callback)) != 0)
{
   Serial.println("Subscribe failed!");
   Serial.println(rc);
}
```

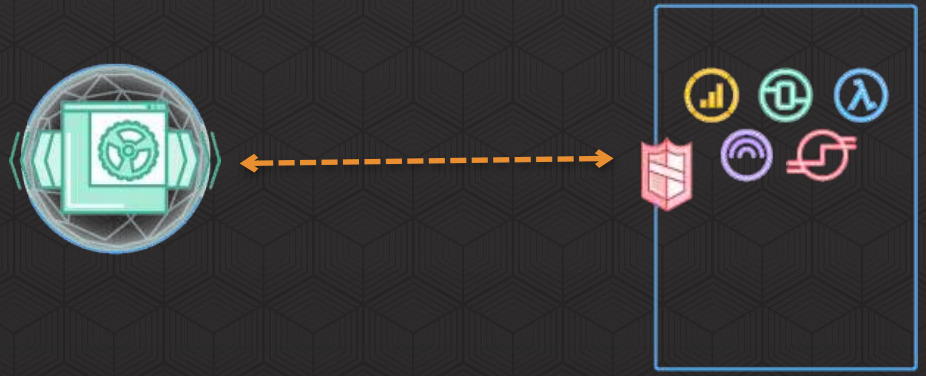
```
if((rc = myClient.publish(”myTopic", msg, strlen(msg),
   1, false)) != 0)
{
   Serial.println("Publish failed!");
   Serial.println(rc);
}
```

amazon web services

Defining a callback for incoming messages

```
// Basic callback function that prints out the message

void msg_callback(char* src, int len) {
   Serial.println("CALLBACK:");
   for(int i = 0; i < len; i++) {
         Serial.print(src[i]);
   }
   Serial.println("");
}
```

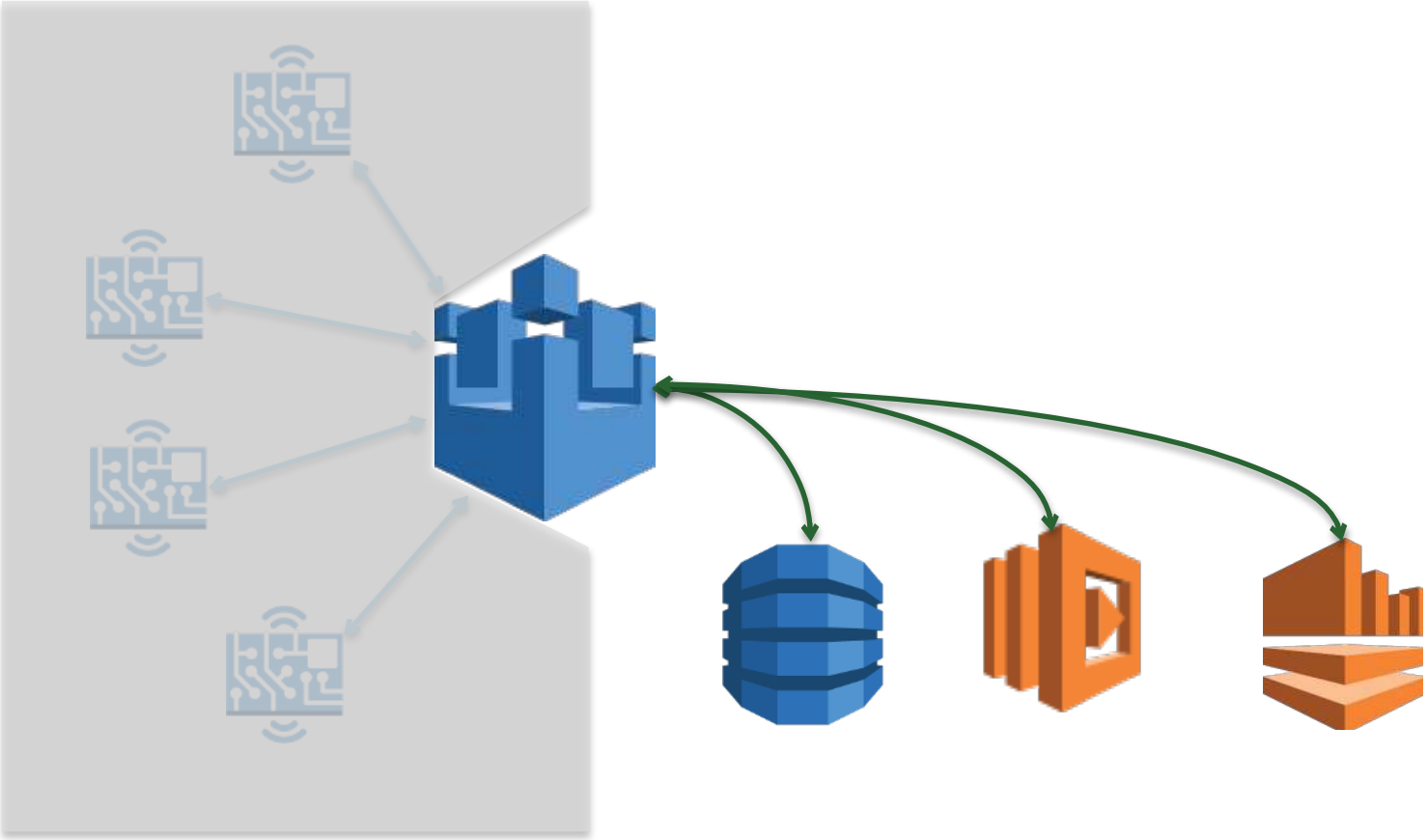
amazon web services



The external world

amazon web services

Granting AWS IoT access to AWS services



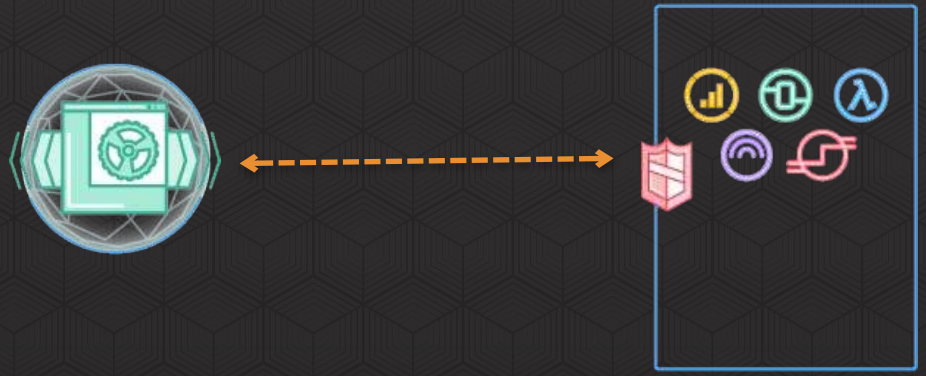
amazon web services

Defining a trust policy for AWS IoT

```
% cat iot-role-trust.json
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Applying the trust policy to AWS IoT

```
% aws iam create-role --role-name my-iot-role
  --assume-role-policy-document file://iot-role-trust.json
{
    "Role": {
        "AssumeRolePolicyDocument": {…},
        "RoleId": "AROAJY7VZX5GEZ3Q7ILU4",
        "CreateDate": "2016-03-19T12:07:03.904Z",
        "RoleName": "my-iot-role",
        "Path": "/",
        "Arn": "arn:aws:iam::613904931467:role/my-iot-role"
    }
}
```

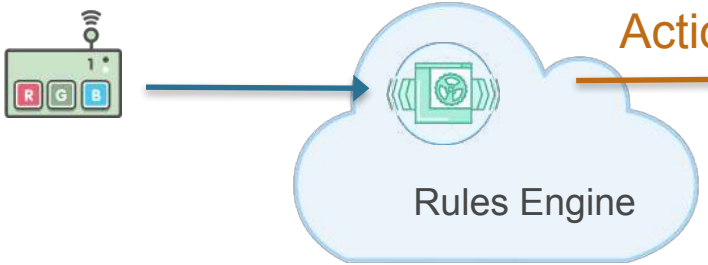


Rules

amazon web services

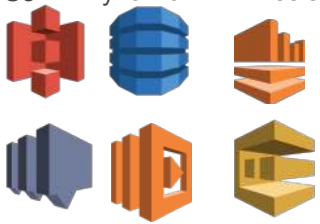
AWS IoT Rules

Rules connect AWS IoT to External Endpoints and AWS Services.

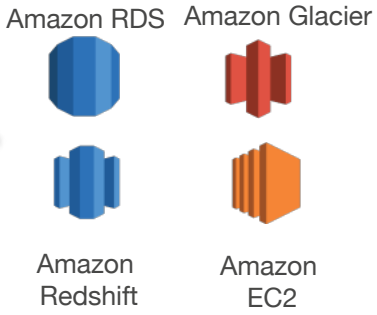


Actions

1. AWS Services
(Direct Integration)



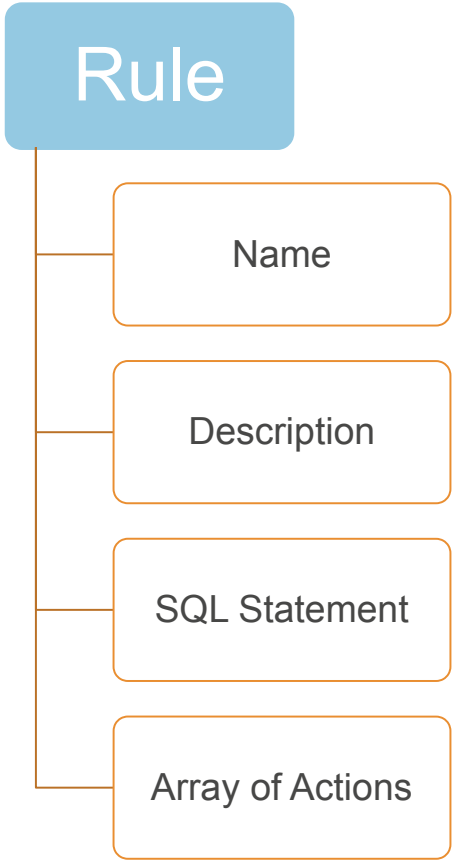
2. Rest of AWS
(via Amazon Kinesis, AWS Lambda, Amazon S3, and more)



3. External Endpoints
(via Lambda and SNS)

amazon web services

AWS IoT Rules Engine



Simple & Familiar Syntax

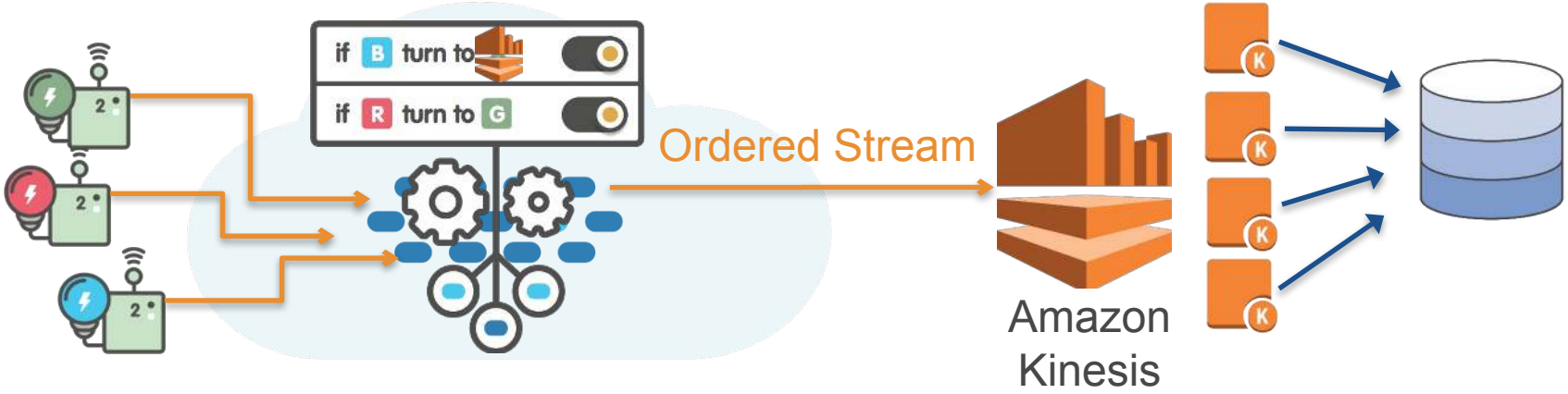
- SQL Statement to define topic filter
- Optional WHERE clause
- Advanced JSON support

Many functions available

- String manipulation (regex support)
- Mathematical operations
- Context based helper functions
- Crypto support
- UUID, Timestamp, rand, etc.

amazon web services

AWS IoT Rules: Streaming Data



N:1 Inbound Streams of Sensor Data

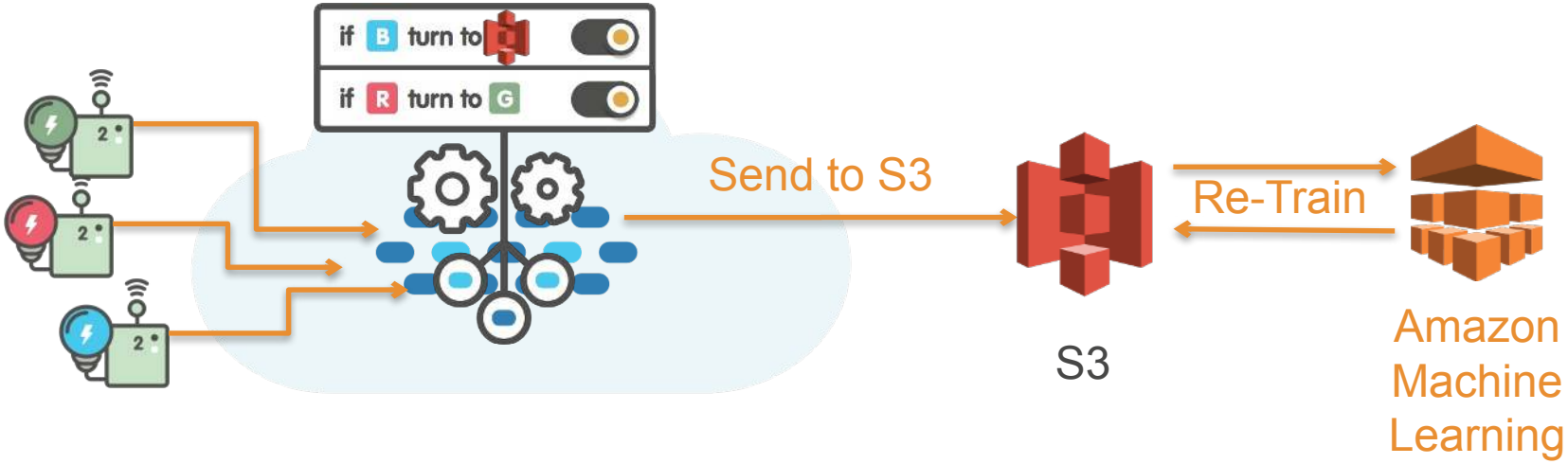
Rules Engine filters, transforms sensor data then sends aggregate to Amazon Kinesis

Amazon Kinesis Streams to Enterprise Applications

Simultaneously stream processed data to databases, applications, other AWS Services

amazon web services

AWS IoT Rules: Machine Learning



Anomaly Detection

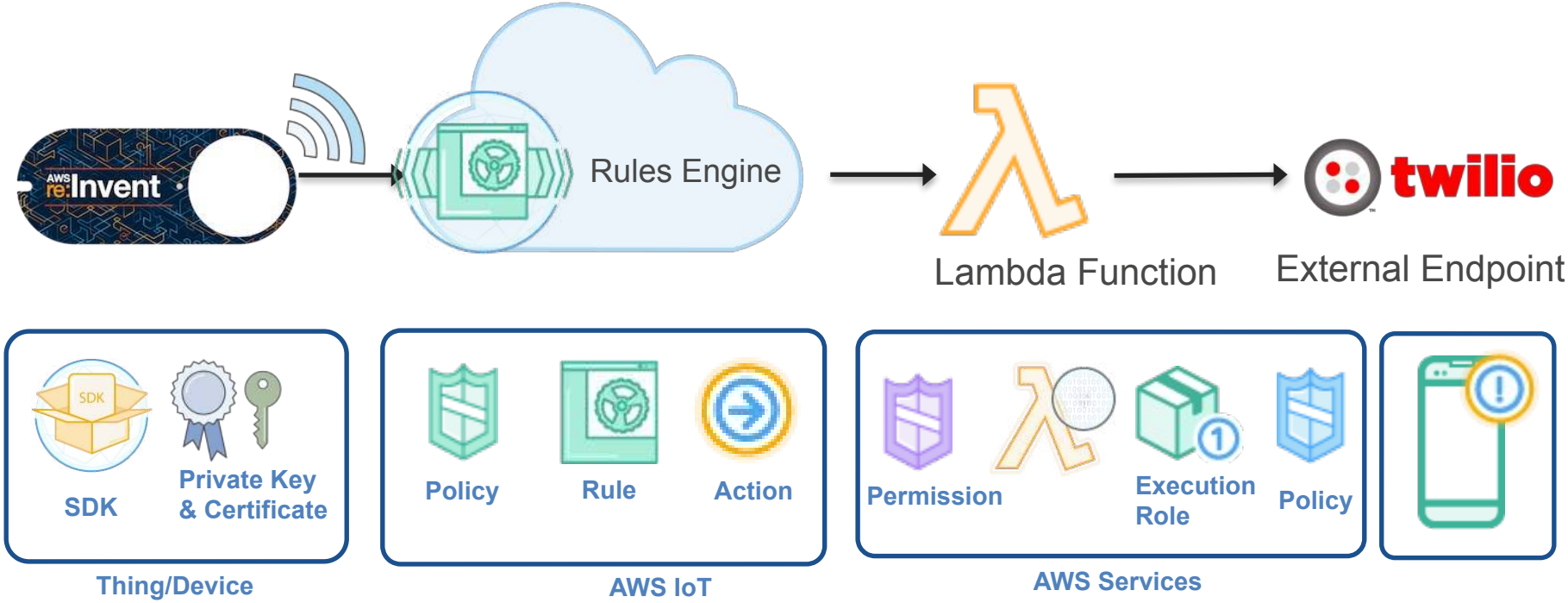
The Rules Engine can feed data to Amazon Machine Learning, for example to predict device failure

Continuous Improvement

Re-train the Amazon Machine Learning model periodically on new data

amazon web services

AWS IoT Rules: External Endpoint



Applying a rule to write to DynamoDB

```
% cat topic1-dynamodb-rule.json
{
  "sql": "SELECT * FROM 'topic1'",
  "ruleDisabled": false,
  "actions": [{
      "dynamoDB": {
          "tableName": "iot-topic1-table",
          "roleArn": "arn:aws:iam::613904931467:role/my-iot-role",
          "hashKeyField": "deviceId",
          "hashKeyValue": "${deviceId}",
          "rangeKeyField": "timestamp",
          "rangeKeyValue": "${timestamp()}"
      }
  }]
}
% aws iot create-topic-rule --rule-name topic1-dynamodb-rule
--topic-rule-payload file://topic1-dynamodb-rule.json
```

Demo : publishing to DynamoDB with MQTT.fx

Publish | Subscribe | Scripts | Broker Status | Log

topic1 [Publish]

```
{
"deviceId": "1234",
"message":"Hello World from MQTT.fx"
}
```

Scan: [Table] iot-topic1-table: deviceId, timestamp

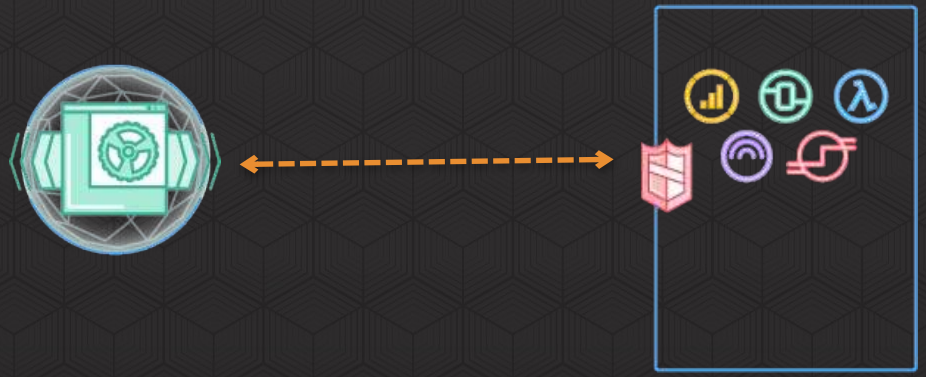
Scan | [Table] iot-topic1-table: deviceId, timestamp

Add filter

Start search

deviceId	timestamp	payload
1234	1458396821732	{ "deviceId" : { "S" : "1234" }, "message" : { "S" : "Hello World from MQTT.fx" }}

amazon web services



Debugging

amazon web services

How can you debug AWS IoT applications?

- Testing with MQTT.fx (or a similar tool) is not enough

- CloudWatch Logs:
  the only way to see what is happening inside AWS IoT
  - Permission issues
  - Rules issues
  - Incorrect JSON messages
  - Etc.

- These logs are not enabled by default:
  - Define a policy allowing AWS IoT to access CloudWatch logs
  - Attach the policy to the AWS IoT role (same one as for external services)

amazon web services

Defining a policy for CloudWatch Logs

```
% cat iot-policy-logs.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy"
             ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

amazon web services

Enabling CloudWatch Logs for AWS IoT

```
% aws iam create-policy
--policy-name my-iot-policy-logs --policy-document file://iot-policy-logs.json
{
    "Policy": {
        "PolicyName": "my-iot-policy-logs",
        "CreateDate": "2016-03-19T12:24:16.072Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ANPAIK73XIV3QG5FF5TX6",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::613904931467:policy/my-iot-policy-logs",
        "UpdateDate": "2016-03-19T12:24:16.072Z"
    }
}

% aws iam attach-role-policy --role-name my-iot-role
--policy-arn "arn:aws:iam::613904931467:policy/my-iot-policy-logs"

% aws iot set-logging-options
--logging-options-payload roleArn="arn:aws:iam::613904931467:role/my-iot-role",logLevel="INFO"
```

Demo : logging events in CloudWatch Logs

```
▼2016-03-19 15:34:23.300 TRACEID:eb1a7666-28c3-4ab4-83a2-f87f66406025
PRINCIPALID:e016283e5191f574f1f76c0278bee9e4d2d4b355d5299b6d16ac4c527f8522b0 [INFO]
EVENT:PublishEvent TOPICNAME:topic1 MESSAGE:PublishIn Status: SUCCESS
▼2016-03-19 15:34:23.403 TRACEID:eb1a7666-28c3-4ab4-83a2-f87f66406025
PRINCIPALID:e016283e5191f574f1f76c0278bee9e4d2d4b355d5299b6d16ac4c527f8522b0 [INFO]
EVENT:MatchingRuleFound TOPICNAME:topic1 CLIENTID:6071974a42ea4594a96446a137b0520b MESSAGE:Matching
rule found: topic1_dynamodb_rule
▼2016-03-19 15:34:23.887 TRACEID:eb1a7666-28c3-4ab4-83a2-f87f66406025
PRINCIPALID:e016283e5191f574f1f76c0278bee9e4d2d4b355d5299b6d16ac4c527f8522b0 [INFO]
EVENT:DynamoActionSuccess TOPICNAME:topic1 CLIENTID:6071974a42ea4594a96446a137b0520b
MESSAGE:Successfully put Dynamo record. Message arrived on: topic1, Action: dynamo, Table:
iot-topic1-table, HashKeyField: deviceId, HashKeyValue: 1234, RangeKeyField: timestamp,
RangeKeyValue: 1458401663404
```

```
▼2016-03-19 17:02:46.691 TRACEID:f8ee7d3f-3c3c-4c23-8458-bf92c6c56c0b
PRINCIPALID:e016283e5191f574f1f76c0278bee9e4d2d4b355d5299b6d16ac4c527f8522b0 [INFO]
EVENT:PublishEvent TOPICNAME:topic1 MESSAGE:PublishIn Status: SUCCESS
▼2016-03-19 17:02:46.804 TRACEID:f8ee7d3f-3c3c-4c23-8458-bf92c6c56c0b
PRINCIPALID:e016283e5191f574f1f76c0278bee9e4d2d4b355d5299b6d16ac4c527f8522b0 [INFO]
EVENT:MatchingRuleFound TOPICNAME:topic1 CLIENTID:6071974a42ea4594a96446a137b0520b MESSAGE:Matching
rule found: topic1_dynamodb_rule
▼2016-03-19 17:02:47.268 TRACEID:f8ee7d3f-3c3c-4c23-8458-bf92c6c56c0b
PRINCIPALID:e016283e5191f574f1f76c0278bee9e4d2d4b355d5299b6d16ac4c527f8522b0 [ERROR]
EVENT:DynamoActionFailure TOPICNAME:topic1 CLIENTID:6071974a42ea4594a96446a137b0520b MESSAGE:Failed
to put Dynamo record. The error received was One or more parameter values were invalid: An
AttributeValue may not contain an empty string (Service: AmazonDynamoDBv2; Status Code: 400; Error
Code: ValidationException; Request ID: CTUP5HKKUONPR9718LQ9QC4J9VVV4KQNSO5AEMVJF66Q9ASUAAJG).
Message arrived on: topic1, Action: dynamo, Table: iot-topic1-table, HashKeyField: deviceId,
HashKeyValue: , RangeKeyField: timestamp, RangeKeyValue: 1458406966804
```

amazon web services

And now it’s your turn!
What will <u>you</u> build?

amazon web services

תודה רבה

Thank You

Julien Simon

julsimon@amazon.fr
Twitter: @julsimon

amazon web services

Pop-up Loft
TEL AVIV