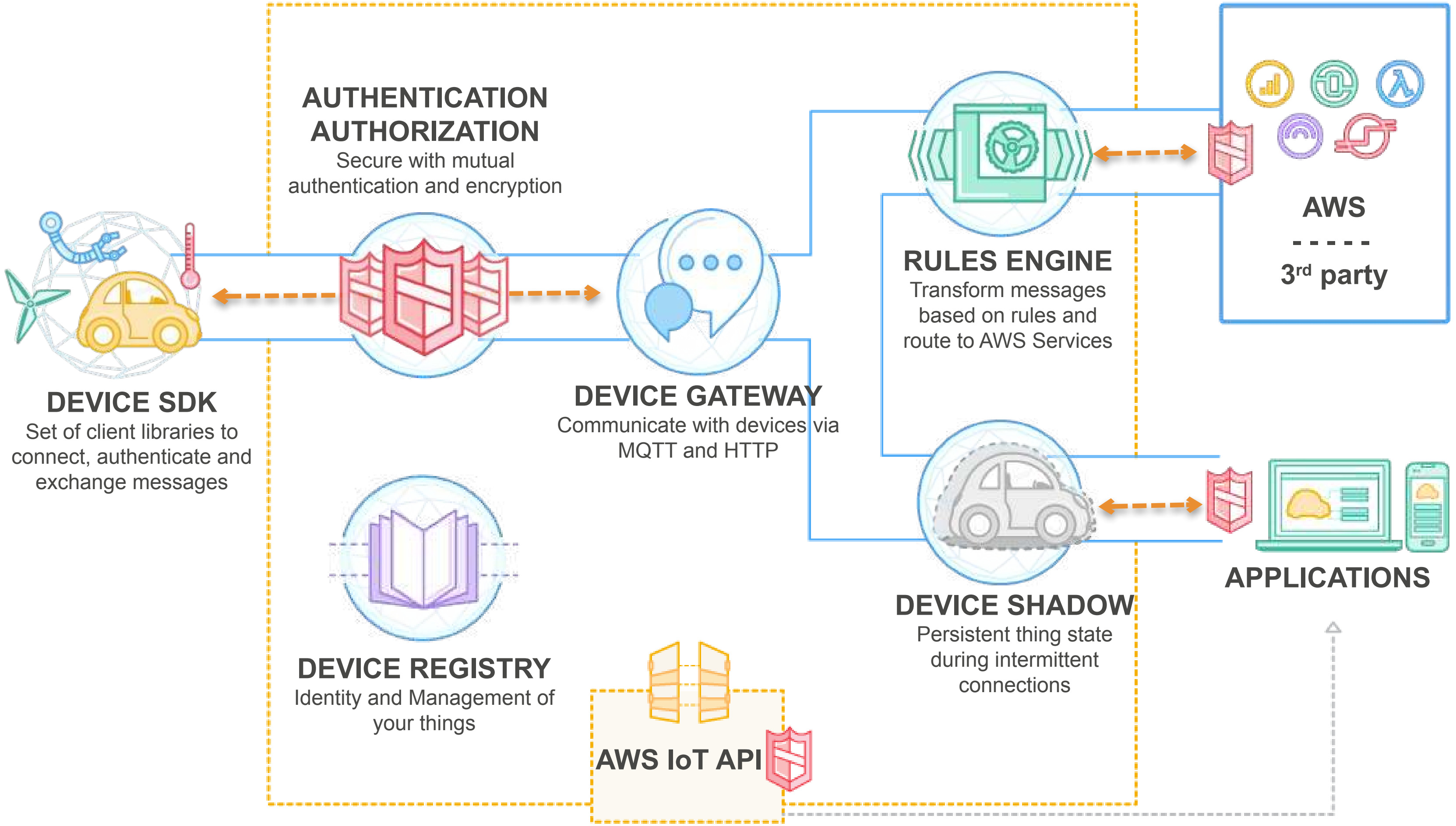
Hands-on with AWS IoT

Julien Simon
Principal Technical Evangelist
Amazon Web Services

julsimon@amazon.fr
@julsimon

Agenda

- Overview of AWS IoT
- Devices & SDKs, with a focus on the Arduino Yún
- The MQTT protocol
- Creating and securing “things”
- Routing AWS IoT messages to other AWS services
- Debugging AWS IoT applications
- And lots of AWS CLI, yeah!


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

*** **NEW (April 7)** : AWS IoT is now available in eu-central-1 (Frankfurt)

Devices & SDKs

Official AWS IoT Starter Kits

AVNET BROADCOM

powered by amazon web services
MARVELL
Marvell EZ-Connect™ MW302 IoT Starter Kit Powered by AWS

MARVELL

RENESAS

TEXAS INSTRUMENTS

MICROCHIP

intel

MEDIATEK

ARROW
QUALCOMM

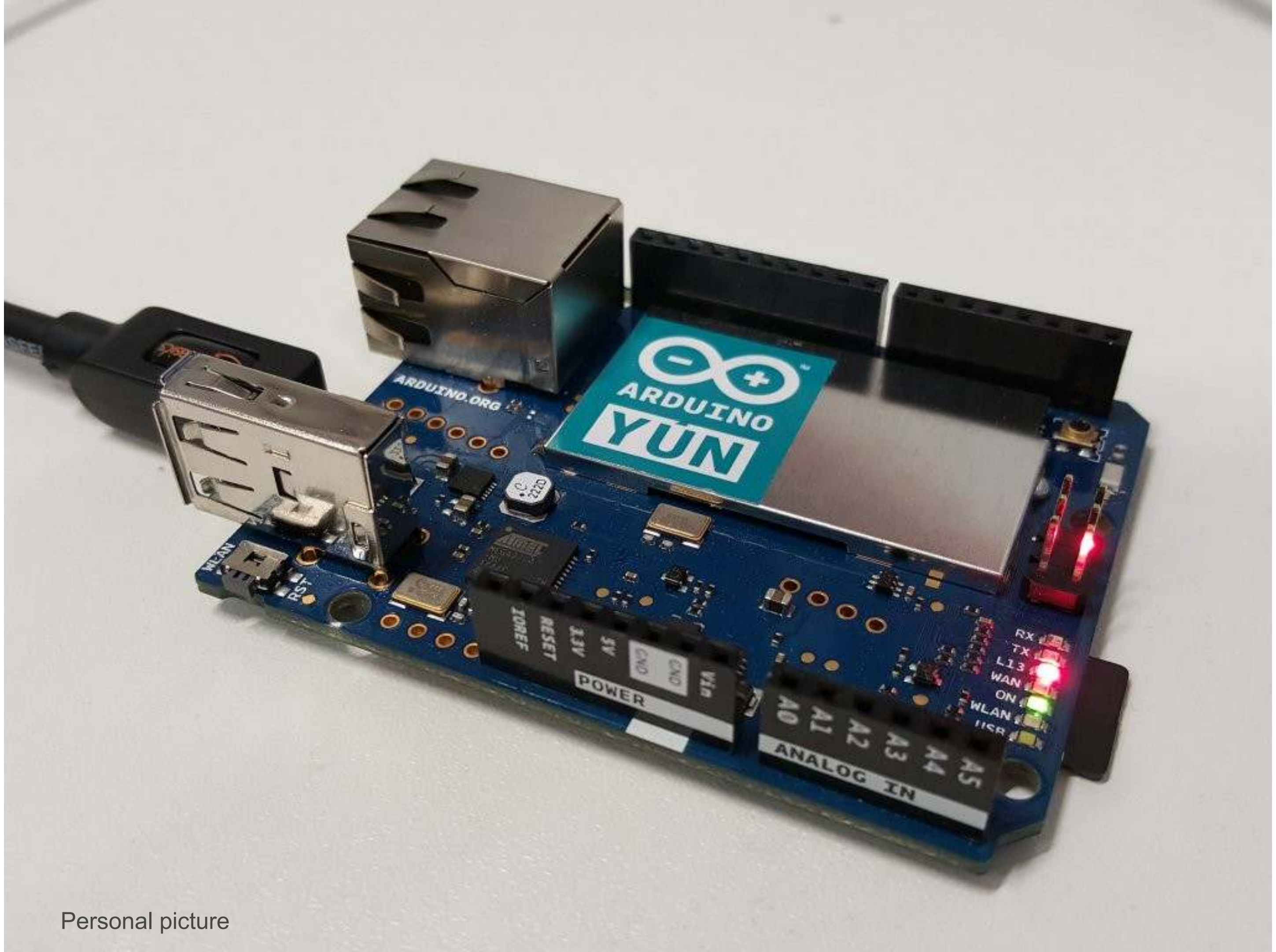
seeed
Grow the Difference

seeed
Grow the Difference

amazon web services

Software platforms supported by AWS IoT

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



Personal picture

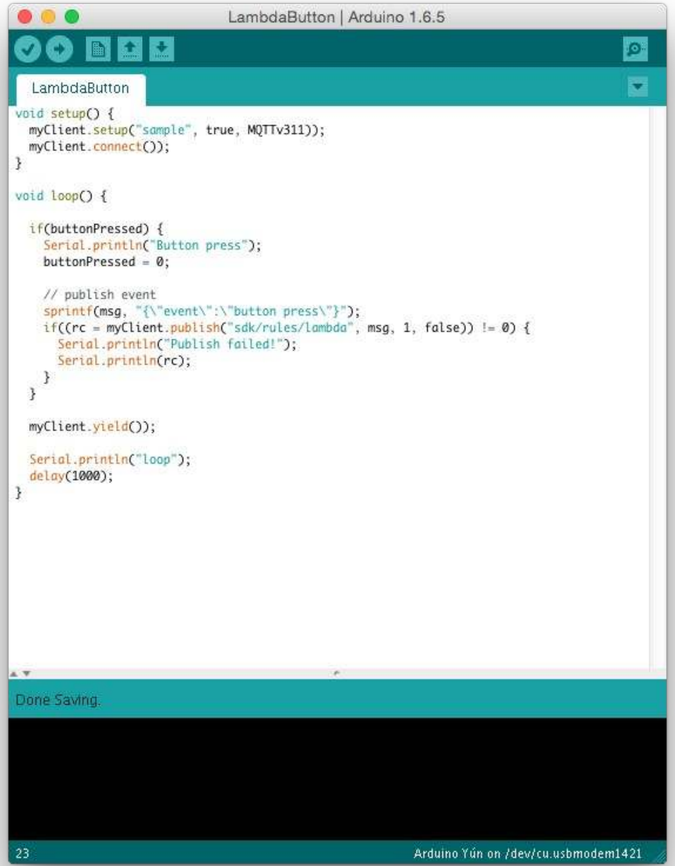
Arduino Yún SDK

Arduino IDE and librairies

http://arduino.org/software

AWS IoT SDK

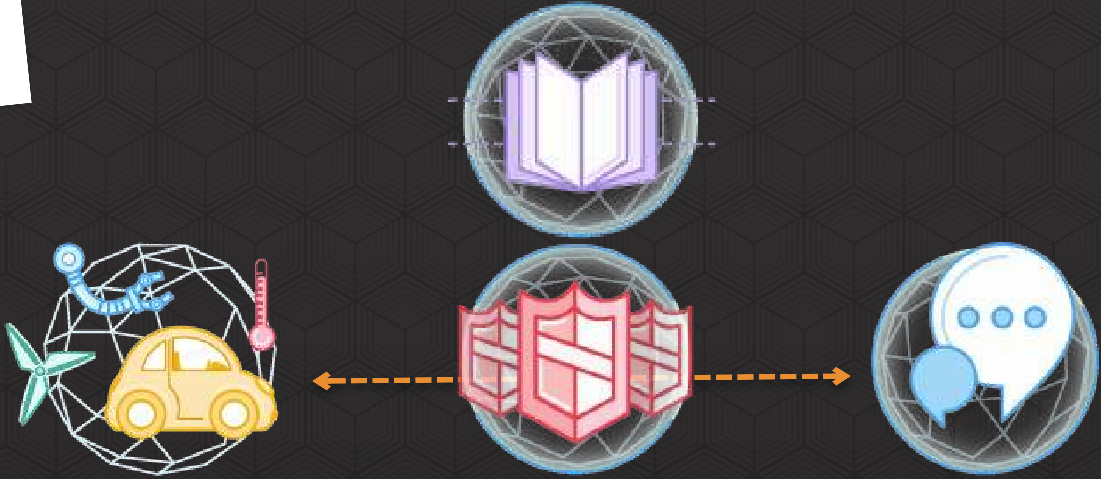
https://github.com/aws/aws-iot-device-sdk-arduino-yun



```
void setup() {
  myClient.setup("sample", true, MQTTv311));
  myClient.connect());
}

void loop() {

  if(buttonPressed) {
    Serial.println("Button press");
    buttonPressed = 0;

    // publish event
    sprintf(msg, "{\"event\":\"button press\"}");
    if((rc = myClient.publish("sdk/rules/lambda", msg, 1, false)) != 0) {
      Serial.println("Publish failed!");
      Serial.println(rc);
    }
  }

  myClient.yield());

  Serial.println("loop");
  delay(1000);
}
```

Done Saving.

23   Arduino Yún on /dev/cu.usbmodem1421



Things

Requirements

- Thing Registry
- Secure Identity for Things
- Secure Communications with Things
- Fine-grained Authorization for:
  - Thing Management
  - Publish / Subscribe Access
  - AWS Service Access

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
```

*** **NEW (April 11)** : You can now use your own certificates

The AWS IoT root certificate, the thing certificate and the thing private key must be installed on your device, e.g. https://github.com/aws/aws-iot-device-sdk-arduino-yun

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
```

Arduino : 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();
       }
   }
}
```

The MQTT protocol

MQTT Protocol

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

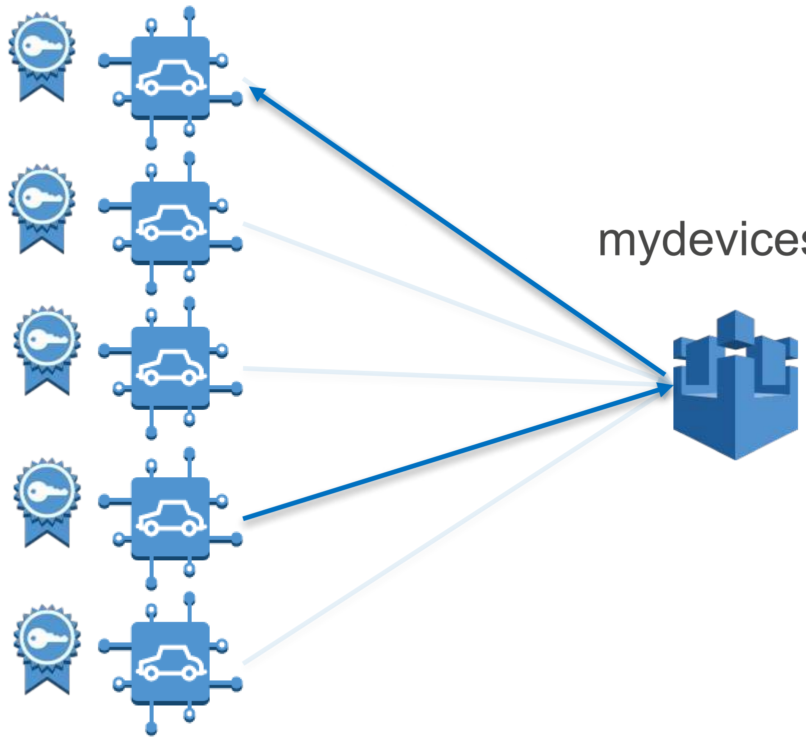
- MQTT is used on oil rigs, connected trucks, and many more critical applications
- Until now, customers had to build, maintain and scale a broker to use MQTT with cloud applications

MQTTS vs HTTPS:

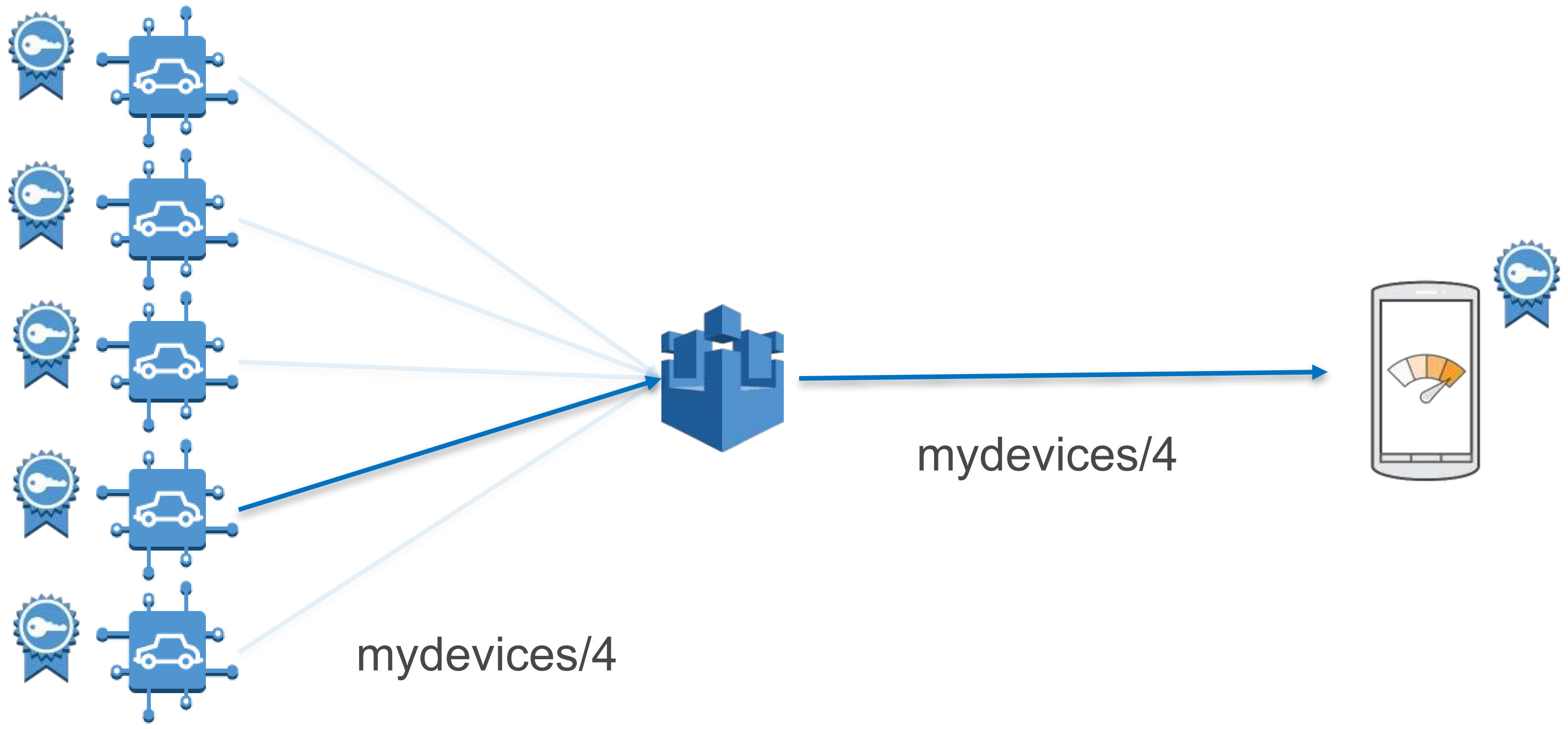
93x faster throughput
11.89x less battery to send
170.9x less battery to receive
50% less power to stay connected
8x less network overhead

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

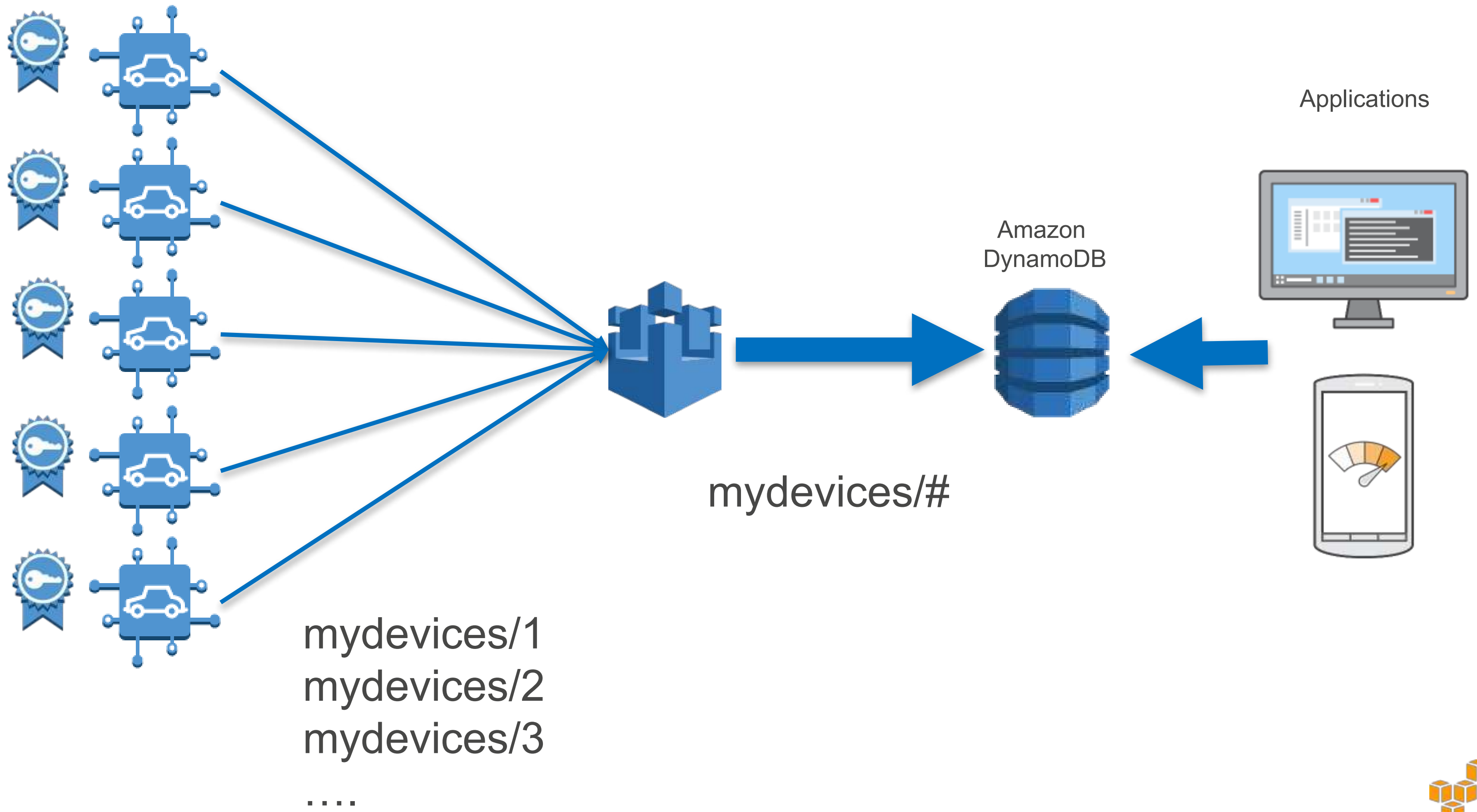
MQTT: device-to-device communication



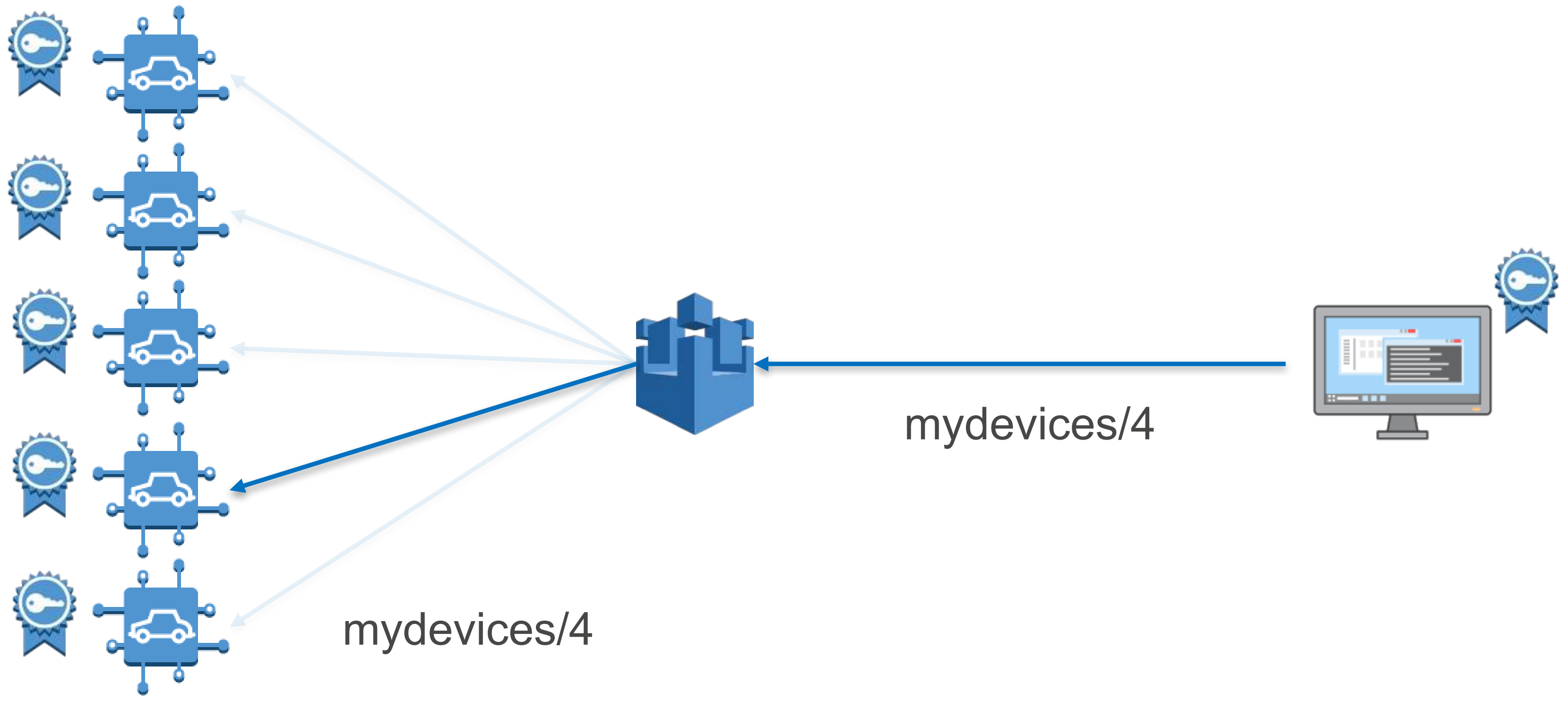
MQTT: collect data from a device



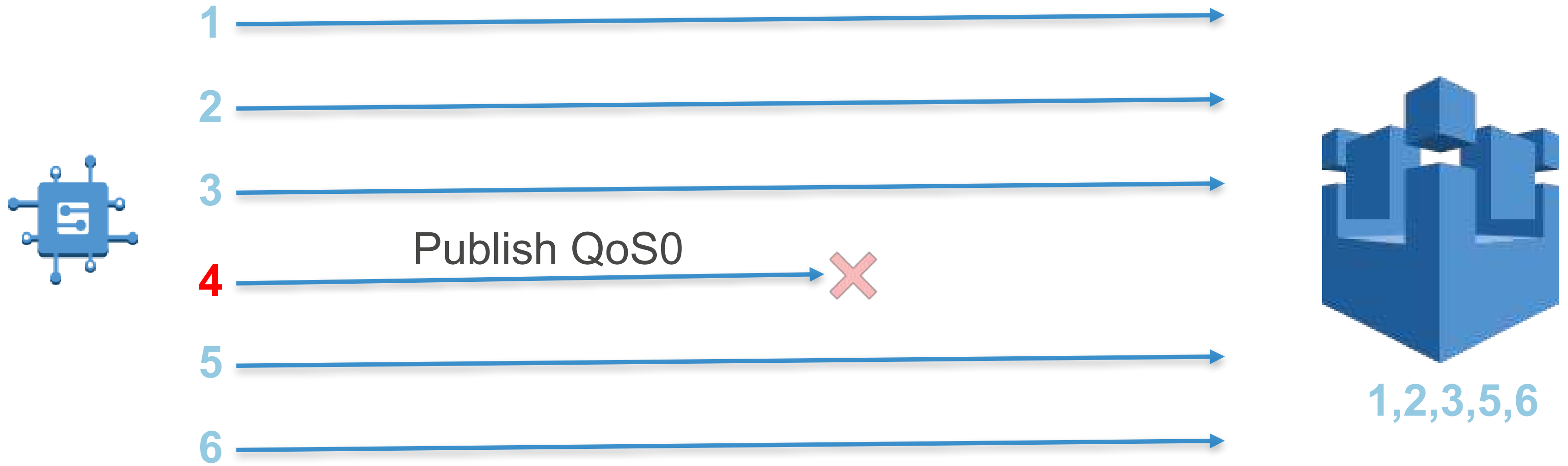
MQTT: aggregate data from many devices



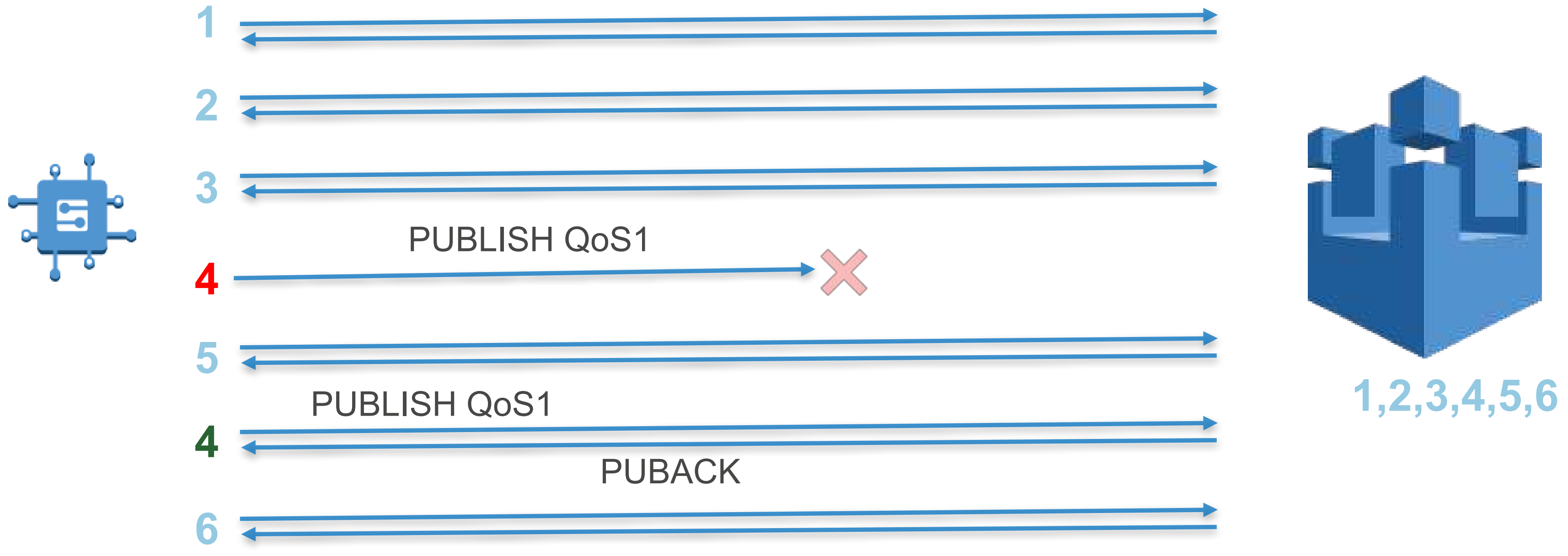
MQTT: update a device



MQTT: QoS 0 (at most once)

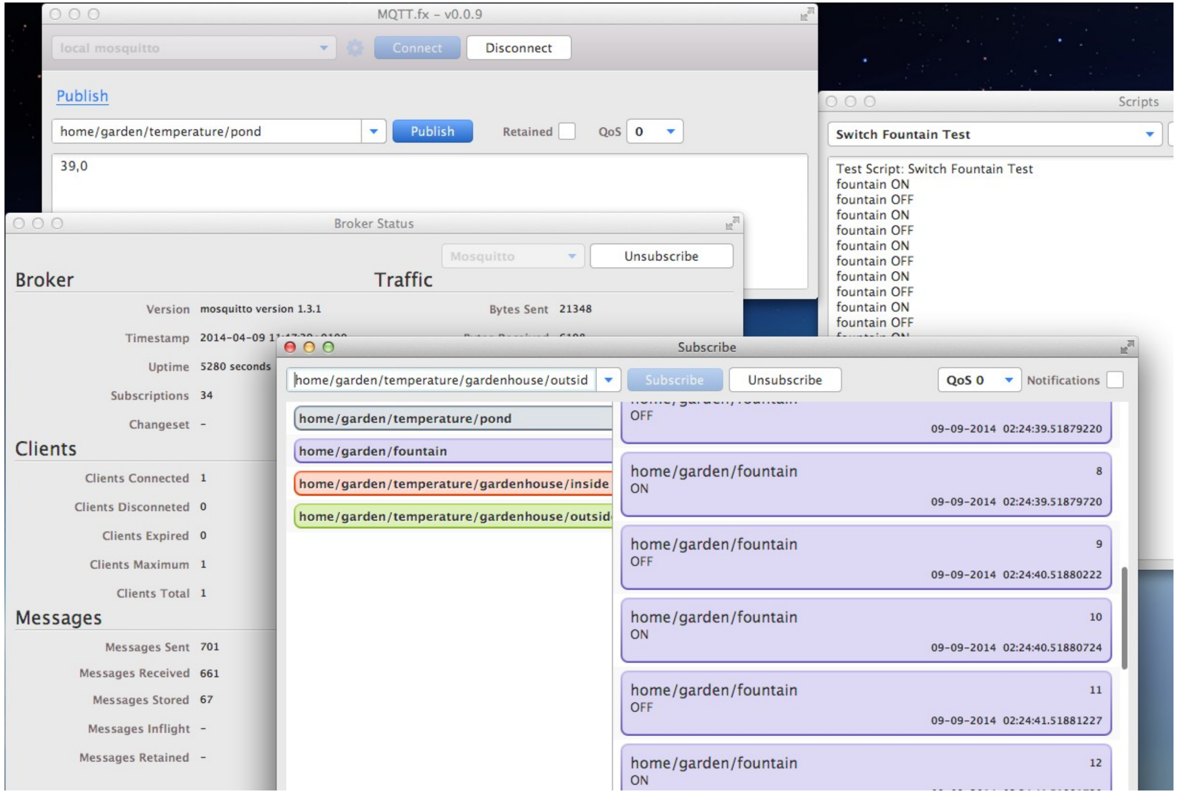


MQTT: QoS 1 (at least once)



MQTT.fx

http://mqttfx.jfx4ee.org/



home/garden/temperature/pond — QoS 0

39,0

Switch Fountain Test

Test Script: Switch Fountain Test
fountain ON
fountain OFF
fountain ON
fountain OFF
fountain ON
fountain OFF
fountain ON
fountain OFF
fountain ON
fountain OFF

Broker

Version	mosquitto version 1.3.1
Timestamp	2014-04-09 1
Uptime	5280 seconds
Subscriptions	34
Changeset	-

Traffic

Bytes Sent	21348

Clients

Clients Connected	1
Clients Disconnected	0
Clients Expired	0
Clients Maximum	1
Clients Total	1

Messages

Messages Sent	701
Messages Received	661
Messages Stored	67
Messages Inflight	-
Messages Retained	-

home/garden/temperature/gardenhouse/outsid — QoS 0

- home/garden/temperature/pond
- home/garden/fountain
- home/garden/temperature/gardenhouse/inside
- home/garden/temperature/gardenhouse/outsid

```
OFF
                                   09-09-2014  02:24:39.51879220
home/garden/fountain                                           8
ON
                                   09-09-2014  02:24:39.51879720
home/garden/fountain                                           9
OFF
                                   09-09-2014  02:24:40.51880222
home/garden/fountain                                          10
ON
                                   09-09-2014  02:24:40.51880724
home/garden/fountain                                          11
OFF
                                   09-09-2014  02:24:41.51881227
home/garden/fountain                                          12
ON
```

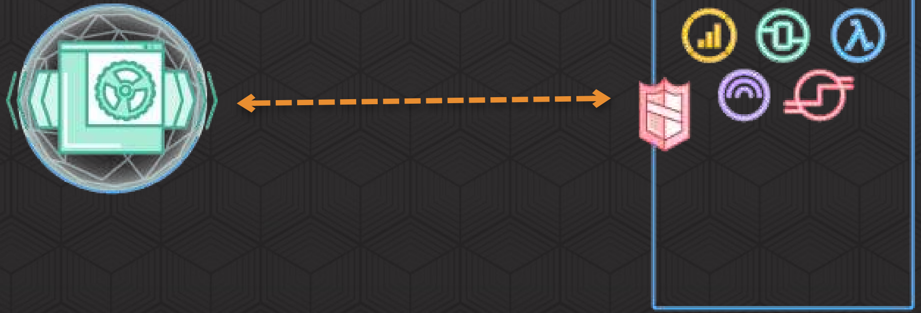
Arduino : subscribing and publishing to a topic

```
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);
}
```

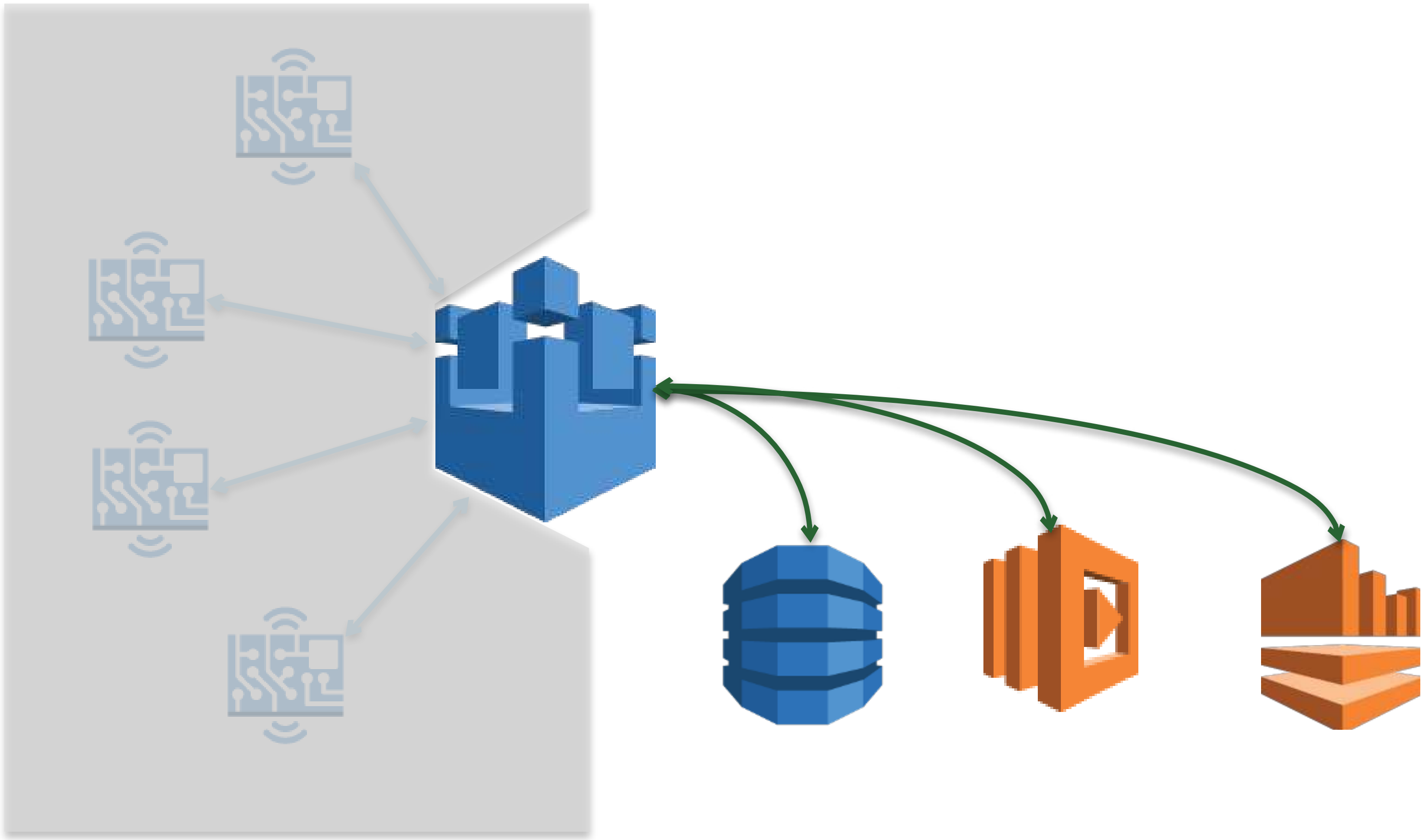
Arduino : callback for incoming messages

```
// 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("");
}
```



Rules

Granting AWS IoT access to AWS services



Defining a trust policy for AWS IoT

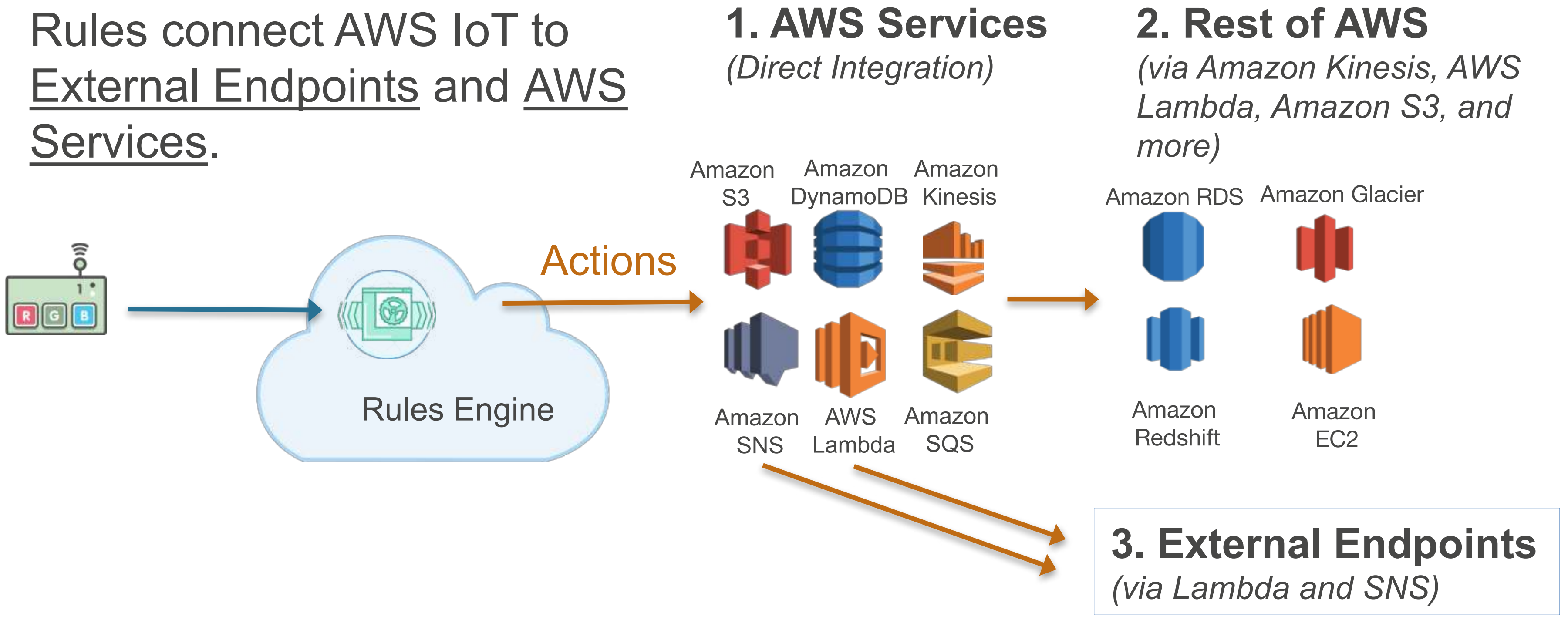
```
% cat iot-role-trust.json
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

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"
    }
}
```

AWS IoT Rules

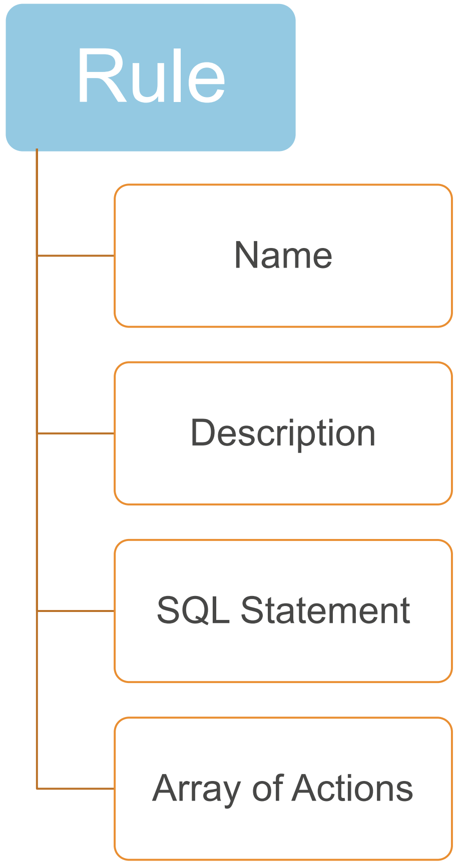
Rules connect AWS IoT to External Endpoints and AWS Services.



*** **NEW (March 16)** : direct integration with Amazon Elasticsearch & CloudWatch

*** **NEW (April 11)** : direct integration with Amazon Machine Learning

AWS IoT Rules Engine



Simple & Familiar Syntax

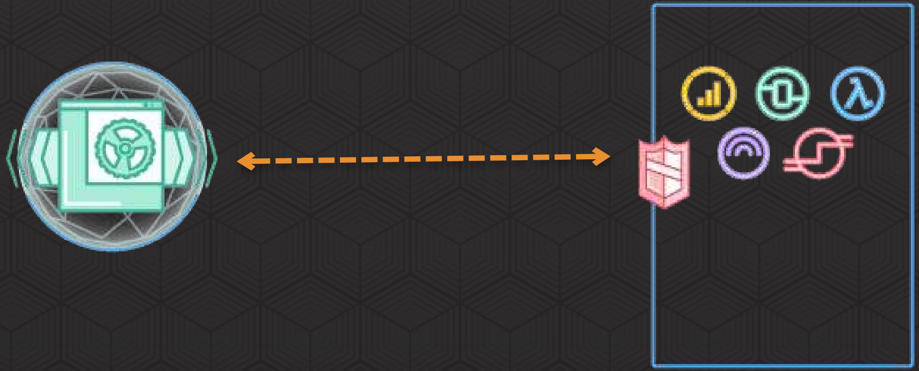
- SQL Statement to define topic filter
- Optional WHERE clause
- Advanced JSON support

Many functions available

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

Creating 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
```



Debugging

How can you debug AWS IoT applications?

- Testing with MQTT.fx (or a similar tool) is not enough
- CloudWatch Logs: <u>the only way to see what is happening inside AWS IoT</u>
  - Permission issue
  - Rule issue
  - Incorrect JSON message
  - 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 (<u>same one as for external services</u>)

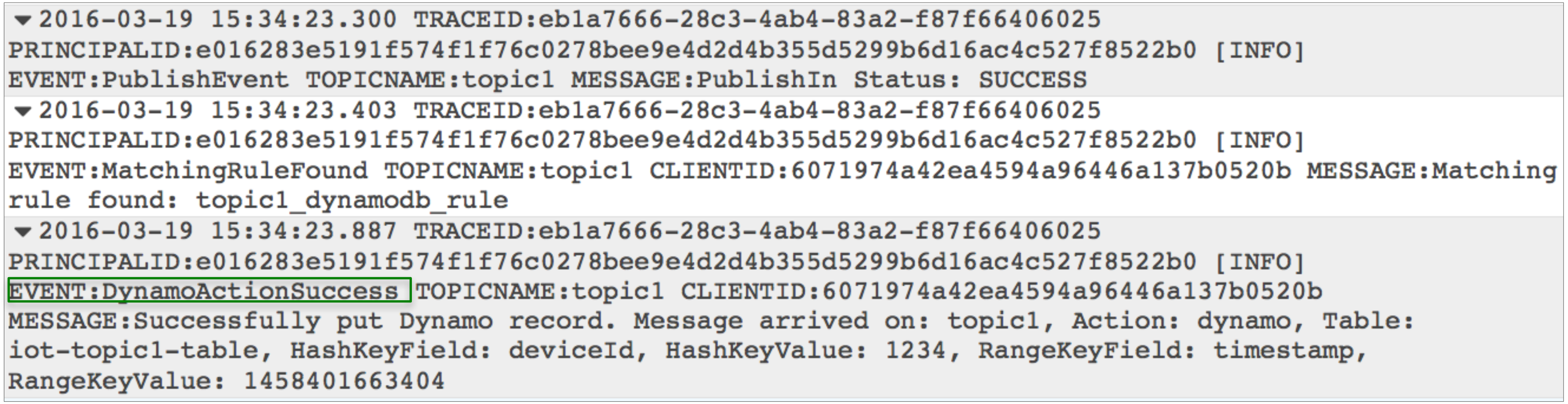
Defining a policy for CloudWatch Logs

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

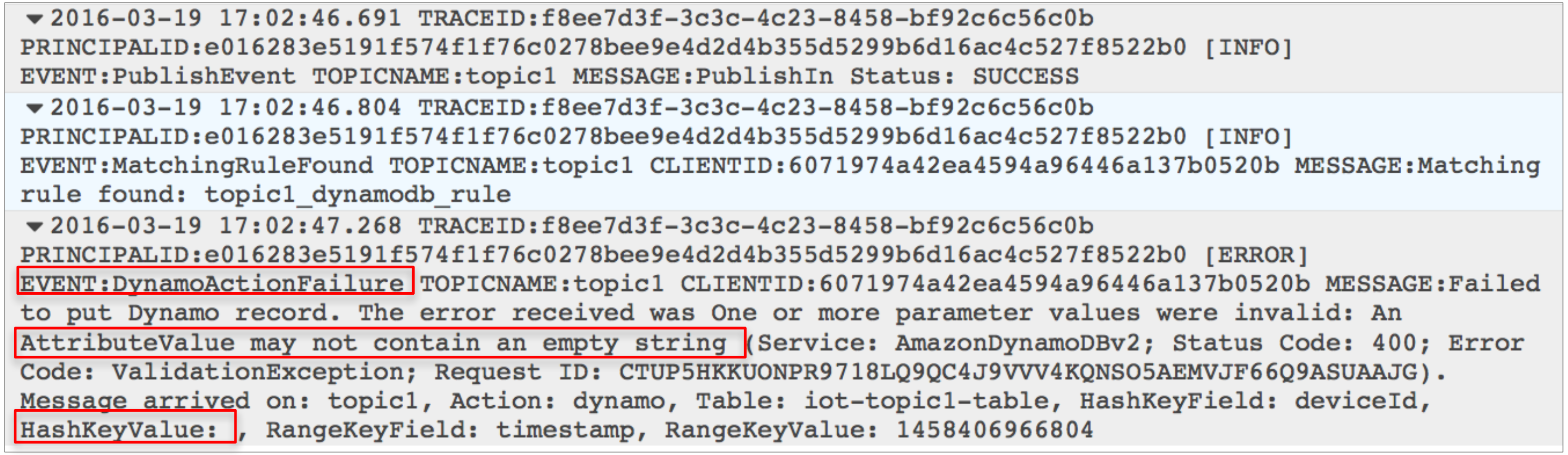
Enabling CloudWatch Logs for AWS IoT

```
% 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 webservices

Next events

May 31st

AWSome Day

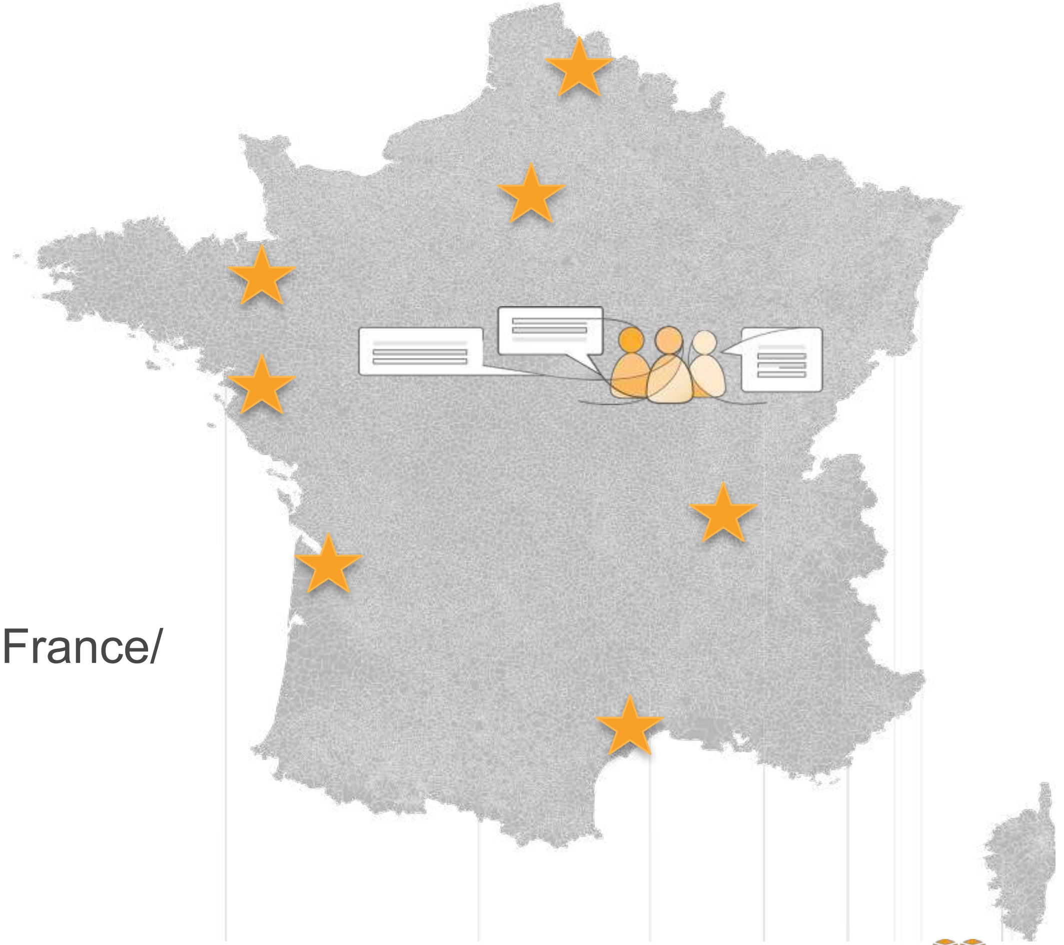
June 28
September 27
December 6

AWS User Groups

Lille
Paris
Rennes
Nantes
Bordeaux
Lyon
Montpellier

facebook.com/groups/AWSFrance/

@aws_actus



Thank You !

Julien Simon
julsimon@amazon.fr
@julsimon