AWS IoT 개요

- AWS IoT 구조 및 Rule Engine -

부산대학교

김호원

부산대
정보보호 및 IoT 연구실,
블록체인 보안 전문연구실,
사물인터넷연구센터

2018.11

목차

AWS IoT 개요

I. AWS IoT 구조

AWS IoT 개요

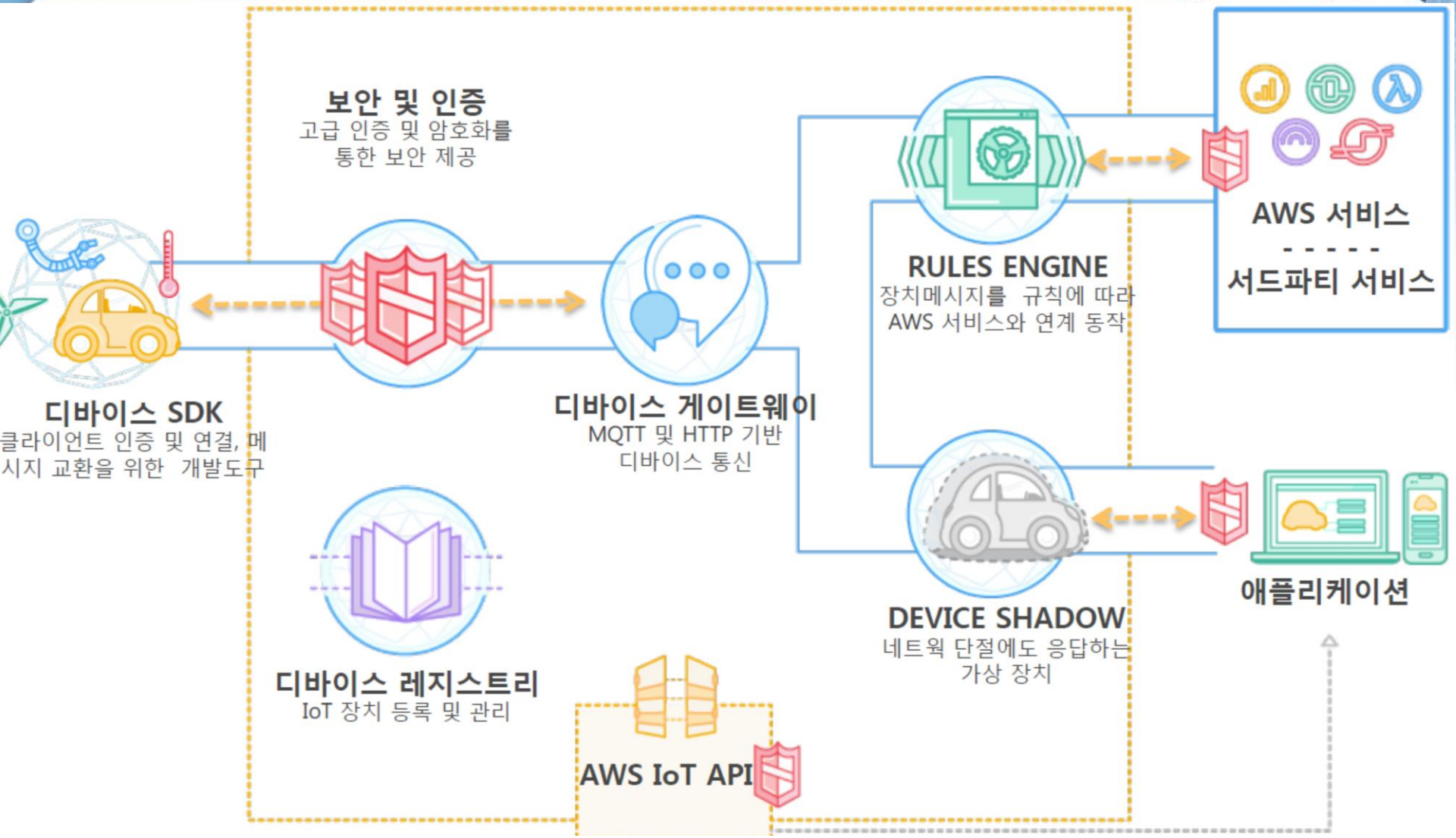
AWS IoT

- Amazon에서 만든 IoT 클라우드 플랫폼
- 센서, actuator, embedded processor, smart device, open source H/W, 컴퓨터 등이 AWS 클라우드와 연동되는 양방향 통신 환경
- 디바이스에서 정보 센싱하여, 해당 정보를 가공/저장/분석/응용 연동 가능

AWS 구성 요소

- 디바이스 게이트웨이
- 메시지 브로커
- Rule engine
- 보안 및 자격 증명 서비스
- 레지스트리
- 그룹 레지스트리
- 디바이스 Shadow
- 디바이스 Shadow 서비스
- 디바이스 provisioning 서비스
- 사용자 지정 인증 서비스
- 작업 서비스

AWS IoT 개요



https://developer.amazon.com/blogs/post/Tx3828JHC709GZ9/Using-Alexa-Skills-Kit-and-AWS-IoT-to-Voice-Control-Connected-Devices

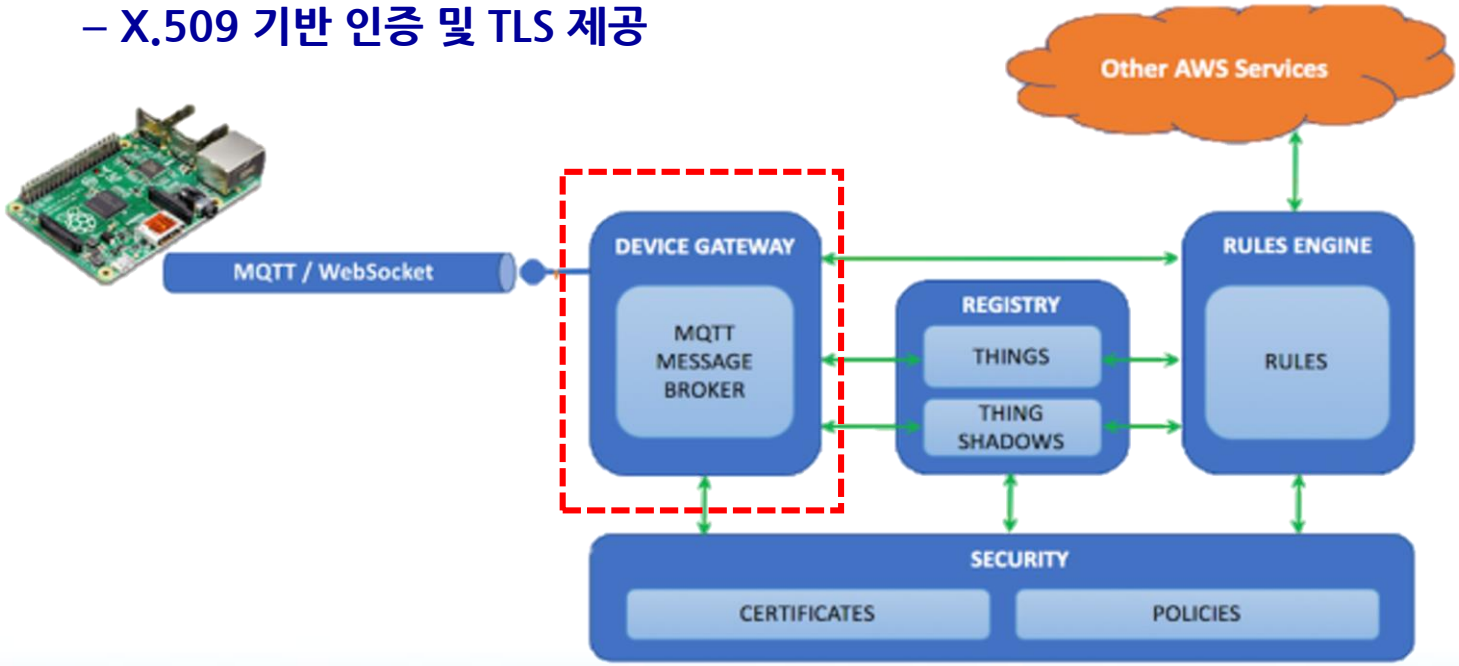
AWS IoT - 게이트웨이/메시지 Broker

AWS 게이트웨이

- 디바이스가 안전하고 효율적으로 AWS IoT와 통신할 수 있게 함
- 메시지 Broker를 가짐

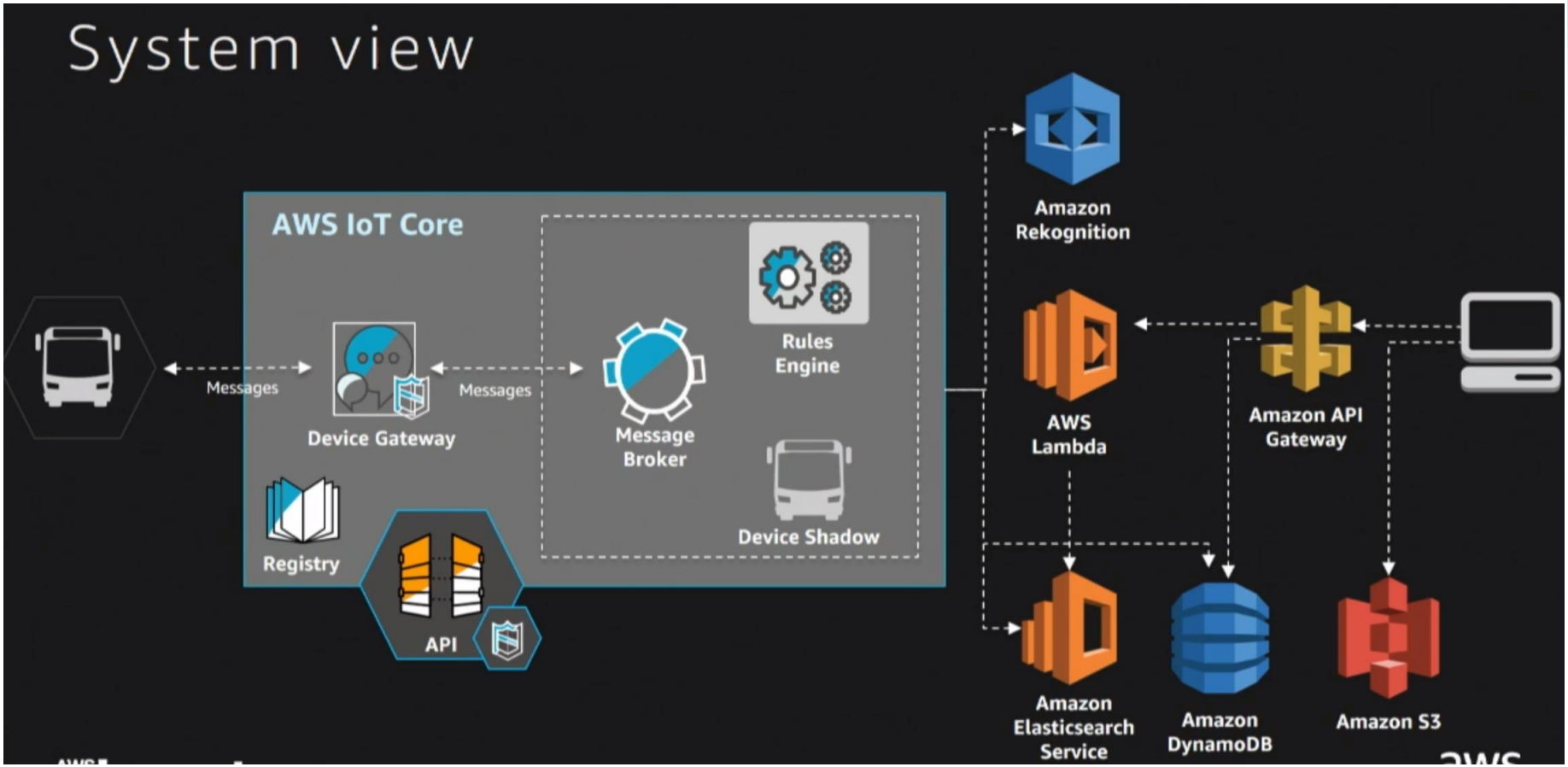
메시지 Broker

- MQTT, HTTP, websocket 기반 통신 제공
- Pub/Sub 서비스 제공
- X.509 기반 인증 및 TLS 제공

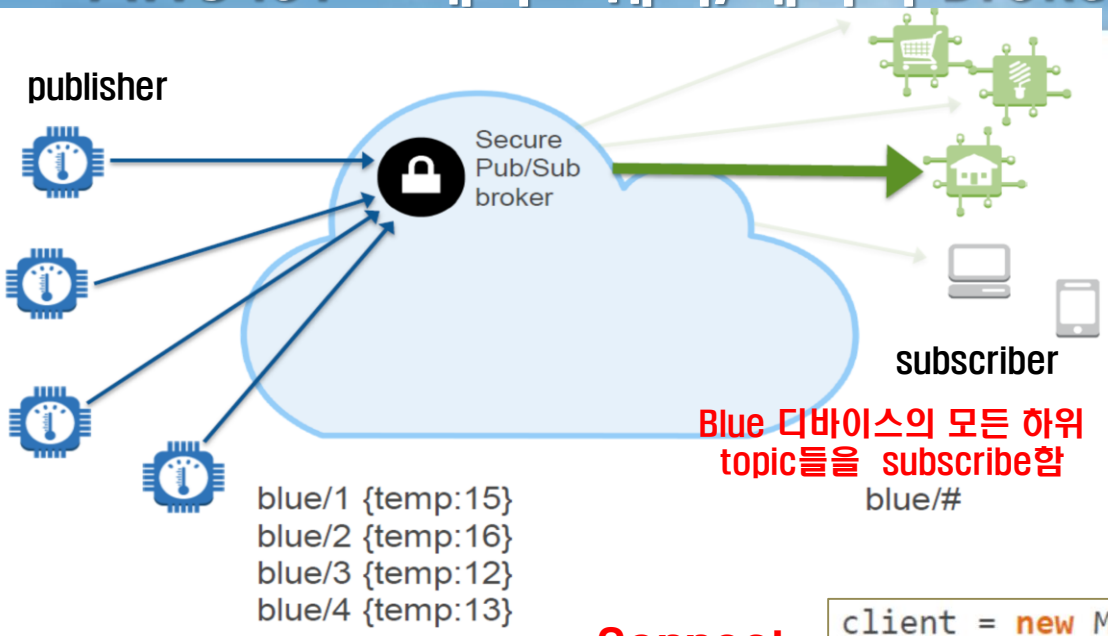


AWS IoT - 게이트웨이/메시지 Broker

- 메시지 Broker가 디바이스 게이트웨이가 아닌 서비스 서버에 설치된 경우



AWS IoT - 게이트웨이/메시지 Broker



Blue 디바이스의 모든 하위 topic들을 subscribe함

blue/#

< MQTT code example, http://wiki.eclipse.org/Paho, javascript >

Connect

Subscribe
Publish

Unsubscribe
Disconnect

```
client = new Messaging.Client(hostname, port, clientId)
client.onMessageArrived = messageArrived;
client.onConnectionLost = connectionLost;
client.connect({ onSuccess: connectionSuccess });

function connectionSuccess() {
    client.subscribe("planets/earth");
    var msg = new Messaging.Message("Hello world!");
    msg.destinationName = "planets/earth";
    client.publish(msg);
}

function messageArrived(msg) {
    console.log(msg.payloadString);
    client.unsubscribe("planets/earth");
    client.disconnect();
}
```

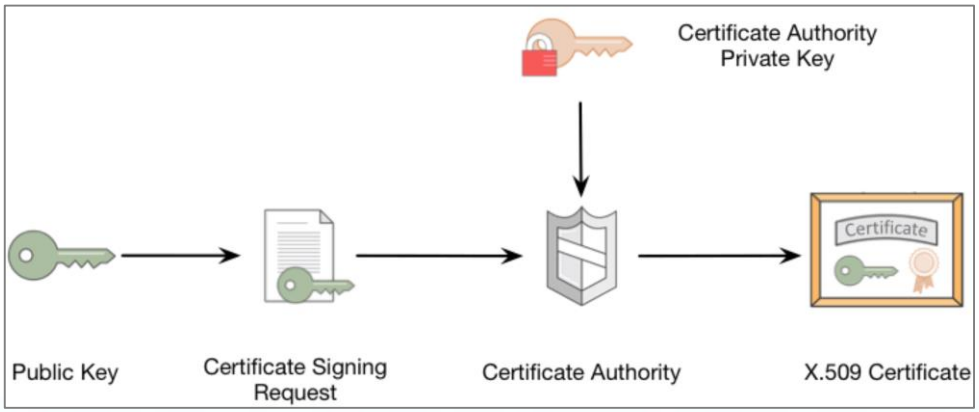
AWS IoT 인증 및 보안

AWS IoT 보안 기술

- X.509 기반의 인증
- TLS 기반의 암호 통신 제공

IoT 디바이스용 인증서 생성

- X.509 certificate은 해당 subject의 public key에 대한 인증서임
- 새로운 X.509 certificate을 만들기위해 CSR(Certificate Signing Request) 작성후 → CA 서버에 전송 필요
  - CSR: subject의 public key와 추가 식별 정보 가짐
- CA에서는 CSR의 추가 식별 정보를 확인 후, 검증되면 public key를 signing하여 돌려줌(즉, 인증서 생성함)



Root Certificate:
- Self-Signed, well known
- Has Root Certificate public key
- Signed by Root Certificate private key

Certificate 1:
- Has own public key
- Signed by Root Certificate private key
- Use Root Certificate public key to prove authenticity

Certificate 2:
- Has own public key
- Signed by Certificate 1 private key
- Use Certificate 1 public key to prove authenticity

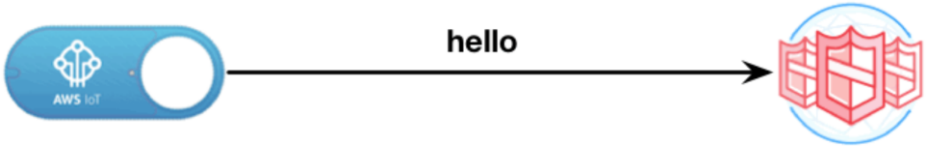
https://aws.amazon.com/ko/blogs/iot/understanding-the-aws-iot-security-model/

AWS IoT 인증 및 보안

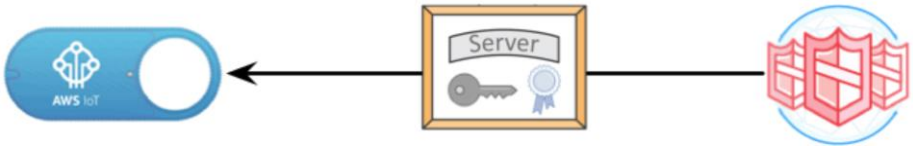
IoT 디바이스가 자신의 인증서와 private key를 사용하여 AWS IoT에 자신을 인증하는 절차 (1/4)

– IoT 디바이스가 자신의 private key와 certificate(public key)를 가지고 있어야 함

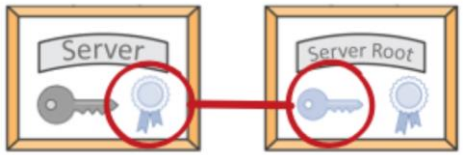
1. IoT 디바이스가 AWS IoT 서버에 hello를 전송하면서, 인증/인가 절차 시작함 (이는 TLS handshake 의 시작 신호임)



2. hello를 수신한 AWS IoT 서버는 서버의 certificate과 자신이 사용하고 싶은 cryptographic method와 자신(서버)의 인증서를 IoT 디바이스에 전송함



3. IoT 디바이스는 서버의 certificate을 검증(root CA의 인증서로 이를 검증할 수 있음)하여, 자신이 정상적인 AWS IoT 서버와 통신함을 확인함 → 디바이스는 이제 서버를 신뢰할 수 있게 됨

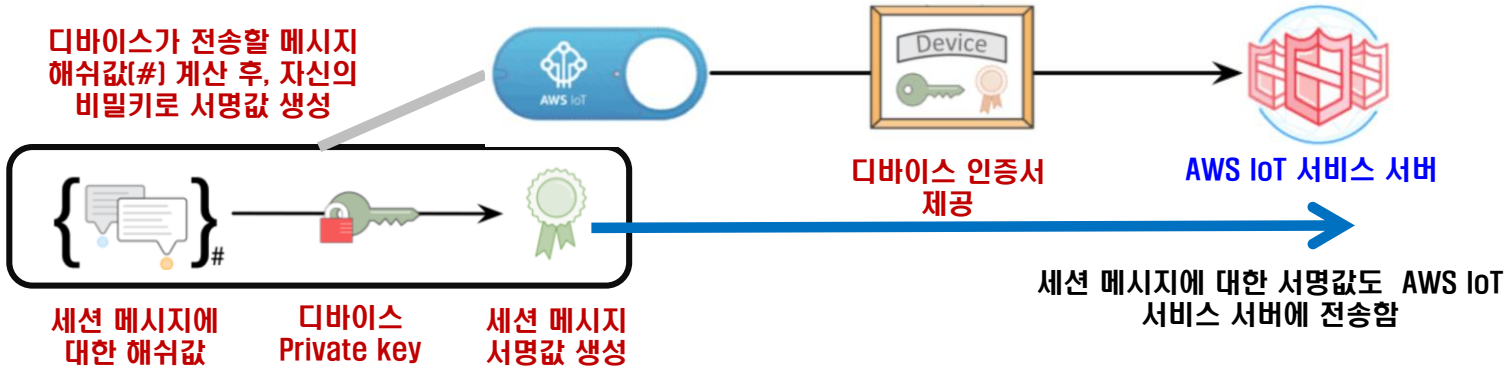


https://aws.amazon.com/ko/blogs/iot/understanding-the-aws-iot-security-model/

AWS IoT 인증 및 보안

IoT 디바이스가 자신의 인증서와 private key를 사용하여 AWS IoT에 자신을 인증하는 절차 (2/4)

4. IoT 디바이스는 AWS IoT 서비스 서버에 자신을 인증해야 함. 또한, 상호 비밀 통신을 위해, shared secret을 공유해야 함
   - 먼저 디바이스 인증서를 서비스 서버에 보냄
   - 디바이스는 자신이 전송할 현재 session의 모든 메시지에 대한 해쉬값을 계산 후, 자신의 private key를 사용하여 서명값을 생성함
   - 이 서명값도 서비스 서버에 보냄

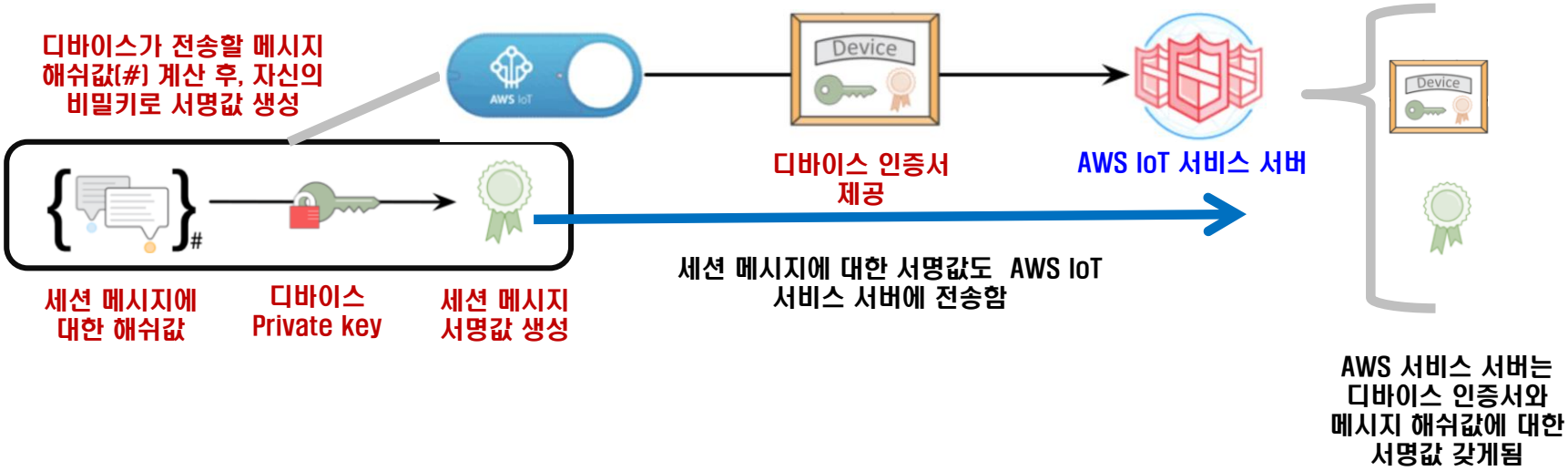


https://aws.amazon.com/ko/blogs/iot/understanding-the-aws-iot-security-model/

AWS IoT 인증 및 보안

IoT 디바이스가 자신의 인증서와 private key를 사용하여 AWS IoT에 자신을 인증하는 절차 (3/4)

4. IoT 디바이스는 AWS IoT 서비스 서버에 자신을 인증해야 함. 또한, 상호 비밀 통신을 위해, shared secret을 공유해야 함
   - 먼저 디바이스 인증서를 서비스 서버에 보냄
   - 디바이스는 자신이 전송할 현재 session의 모든 메시지에 대한 해쉬값을 계산 후, 자신의 private key를 사용하여 서명값을 생성함
   - 이 서명값도 서비스 서버에 보냄



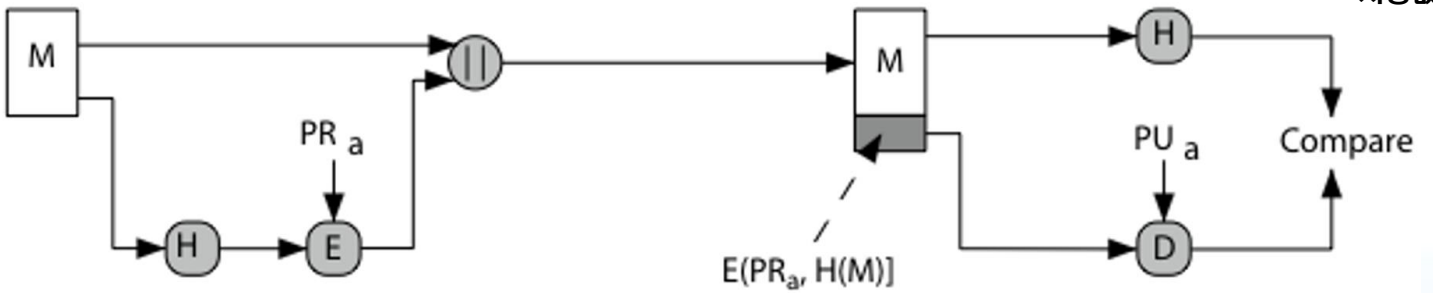
https://aws.amazon.com/ko/blogs/iot/understanding-the-aws-iot-security-model/

AWS IoT 인증 및 보안

IoT 디바이스가 자신의 인증서와 private key를 사용하여 AWS IoT에 자신을 인증하는 절차 (4/4)

5. AWS IoT 서비스 서버는 디바이스 공개키, 메시지 해쉬에 대한 서명값을 가지고 있으며, 서명값이 정당하다고 검증함(디바이스 공개키를 사용하여, 서명 검증)

- 디바이스로부터 메시지를 수신한 후, 서비스 서버는 메시지에 대한 해쉬값 생성
- 서비스 서버는 수신한 서명값을 공개키(디바이스 인증서)로 복호함
- 복호화된 해쉬값과 수신된 메시지로 생성한 해쉬값 비교하여 제대로 메시지가 수신됨을 확인함



https://aws.amazon.com/ko/blogs/iot/understanding-the-aws-iot-security-model/

AWS IoT 인증 및 보안

MQTT, HTTPS 프로토콜 지원

- 서버 인증: 인증서 기반 + TLS 활용
- 클라이언트 인증: 인증서 기반 + TLS, 혹은 API 키 + TLS
- 통신 메시지 기밀성 : TLS 기반
- ID 식별 체계 : ARN(AWS Resource Name) 사용

	MQTT + Mutual Auth TLS	AWS Auth + HTTPS
Server Auth	TLS + Cert	TLS + Cert
Client Auth	TLS + Cert	AWS API Keys
Confidentiality	TLS	TLS
Protocol	MQTT	HTTP
Identification	AWS ARNs	AWS ARNs
Authorization	AWS Policy	AWS Policy

AWS IoT 인증 및 보안

AWS 식별 체계 (ARN :AWS Resource Name)

– 구문

```
arn:aws:iot:your-region:account-id:cert/cert-ID
arn:aws:iot:your-region:account-id:policy/policy-name
arn:aws:iot:your-region:account-id:rule/rule-name
arn:aws:iot:your-region:account-id:client/client-id/rule-name
```

– 예 1:

```
arn:aws:iot:your-region:123456789012:cert/123a456b789c123d456e789f123a456b789c123d456e789f123a456b789c123c456d7
arn:aws:iot:your-region:123456789012:policy/MyIoTPolicy
arn:aws:iot:your-region:123456789012:rule/MyIoTRule
arn:aws:iot:your-region:123456789012:client/client101
```

– 예 2 (IoT 디바이스): arn:aws:iot:us-west-2:313[redacted]97:thing/myIoTButton1

– 예 3 (인증서):

arn:aws:iot:us-west-2:3137[redacted]97:cert/535a0ef6493d4ba4f0691[redacted]7e63f90daa5ad0487096

발행자
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
제목
CN=AWS IoT Certificate
생성 날짜
2018. 11. 2. 오전 10:59:44
발효 날짜
2018. 11. 2. 오전 10:57:44
만료 날짜
2050. 1. 1. 오전 8:59:59

AWS IoT 권한 관리

- 세분화된 권한 관리 제공
  - 장치 관리 제공
  - Pub/Sub 데이터 접근 관리
  - AWS 서비스 접근 제어

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": ["iot:Publish"],
      "Resource":
         ["arn:aws:iot:us-east-1::topic/foo"]
    },
    {
        "Effect": "Allow",
        "Action": ["iot:Subscribe"],
      "Resource":
         ["arn:aws:iot:us-east-1::topicfilter/foo/bar/*"]
}]}
```

〈 Subscribe 정책 〉

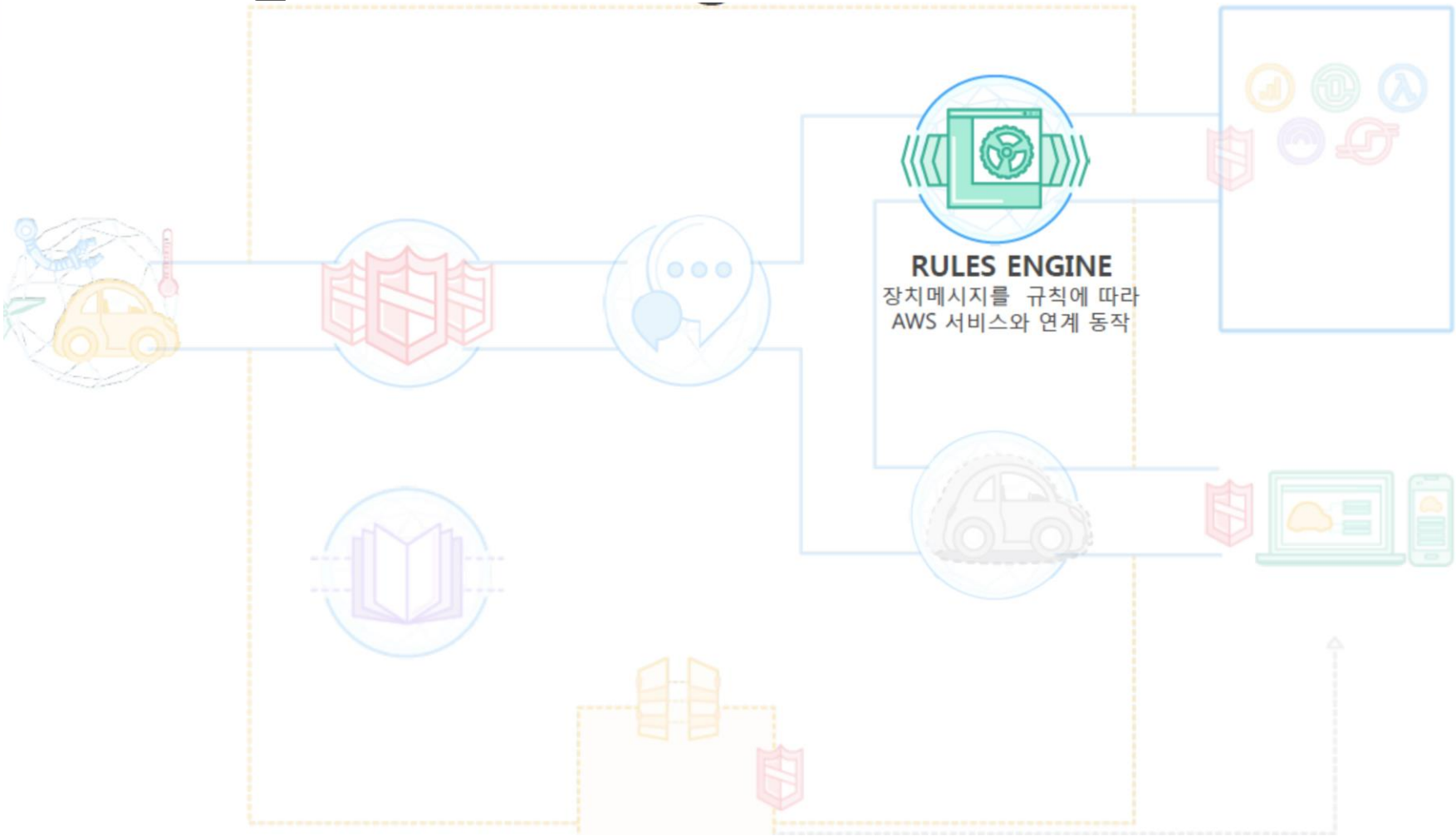
버전 2 업데이트 시간: 2018. 11. 2. 오전 11:09:56 정책 문서 편집

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:us-west-2:3137[redacted]97:topic/IoTButton/G030MD046075LKV0"
    }
  ]
}
```

II. AWS IoT Rule Engine

AWS IoT Rules Engine

AWS IoT 룰 엔진



AWS IoT Rules Engine

- Rule을 사용하여 AWS 서비스와 상호 작용 가능. MQTT topic stream을 기반으로 rule이 분석되고 작업 수행됨.
- Rule을 사용하여 다음과 같은 작업 수행 가능
  - 디바이스로부터 수신한 데이터를 보강 또는 필터링
  - 디바이스로부터 수신한 데이터를 Amazon DynamoDB 데이터베이스에 기록
  - 파일을 Amazon S3에 저장
  - Amazon SNS을(를) 사용하여 모든 사용자에게 푸시 알림 발송
  - Amazon SQS 대기열에 데이터 게시
  - Lambda 함수를 호출하여 데이터 추출 등

- 그런데 위의 작업이 수행되기 위해선, **AWS 리소스에 access 권한이 부여되어야 함**
- **즉, IoT role을 먼저 만들고 → rule을 갖도록 함**

AWS IoT Rules Engine - 리소스에 access 권한 부여

- **AWS IoT에게 role 제공을 통해, 접근 권한(access) 부여함**
  - Rule이 리소스에 접근하는 것을 제어하기 위해, IAM role을 사용함
    - 즉, IAM role은 리소스/서비스 접근을 제어하게됨
  - 이에, 리소스에 접근하는 정책을 가지는 IAM role을 먼저 만듦
  - AWS IoT 서비스는 rule을 실행할 때, 이 role을 수임(take/assume) 함

- IAM: AWS Identity & Access Mgmt, AWS ID 및 액세스 관리
- AWS 서비스 및 자원에 대해 권한을 부여할 수 있는 권한 관리 솔루션

AWS IoT Rules Engine

이에, IAM role을 생성해야 함

- trust 파일(iot-role-trust.json)을 만든 후, role 파일을 생성함
- *aws iam create-role --role-name my-iot-role --assume-role-policy-document file://iot-role-trust.json*

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

iot-role-trust.json (trust policy)

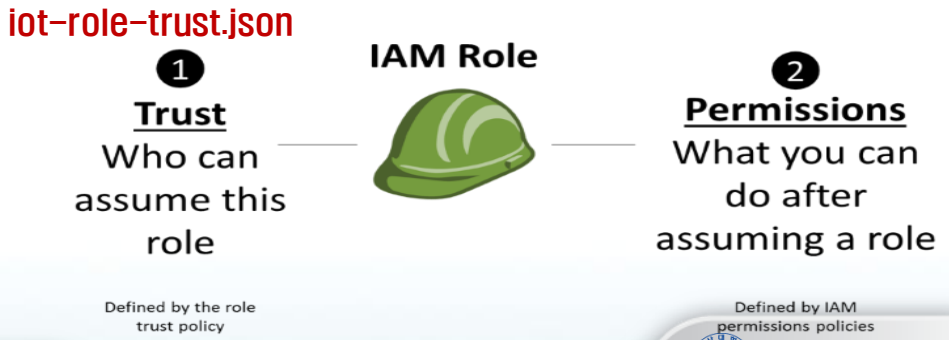
Principal, iot.amazonaws.com이 role을 assume 함

```
{
  "Role": {
      "AssumeRolePolicyDocument": "url-encoded-json",
      "RoleId": "AKIAIOSFODNN7EXAMPLE",
      "CreateDate": "2015-09-30T18:43:32.821Z",
      "RoleName": "my-iot-role",
      "Path": "/",
      "Arn": "arn:aws:iam::123456789012:role/my-iot-role"
  }
}
```

my-iot-role

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "dynamodb:*",
        "Resource": "*"
    }]
}
```

iot-policy.json (permission policy)



AWS IoT Rules Engine

Rule 사례

- 다음은 iot / test 항목으로 보낸 모든 메시지를 지정된 DynamoDB 테이블에 rule임
- SQL 문은 메시지와 역할을 필터링함.
- role ARN grants AWS IoT permission to write to the DynamoDB table.

```
{
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "dynamoDB": {
          "tableName": "my-dynamodb-table",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
          "hashKeyField": "topic",
          "hashKeyValue": "${topic(2)}",
          "rangeKeyField": "timestamp",
          "rangeKeyValue": "${timestamp()}"
      }
  }]
}
```

AWS IoT Rules Engine

참고) trust policy와 permission policy 예

(trust policy)

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "ec2.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

This trust policy allows the Amazon EC2 service to assume the role.

(permission policy)

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

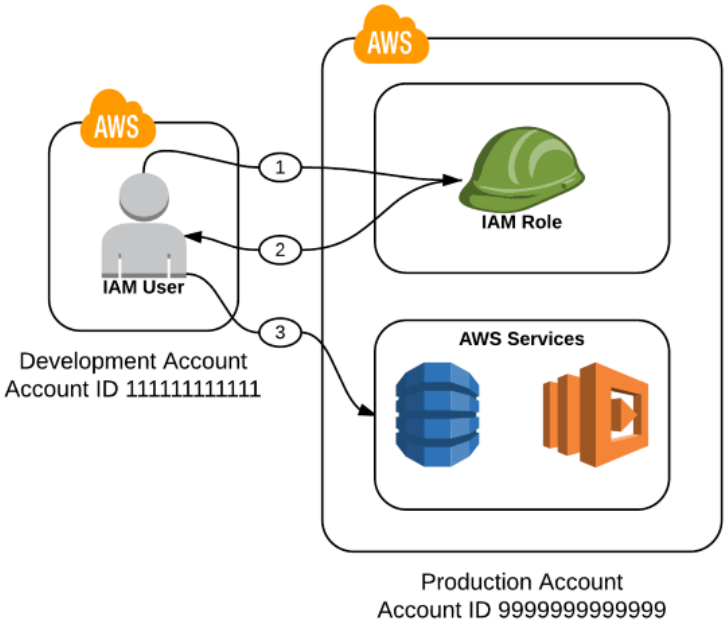
This permissions policy allows the role to perform only the ListBucket action on the example_bucket Amazon S3 bucket.

참고) IAM role의 역할

IAM role 이란? (1/3)

- IAM의 role은 두가지 형태의 policy를 가짐
- Trust policy: 이 정책은 어느 entity가 role을 수임(assume)할 것인지를 정의함
- Permissions policy: role이 어느 AWS resource를 접근하며, 어떤 action을 취할 수 있는지를 정의

사례



(1) IAM 사용자는 AWS STS(Security Token Service)에 연결하여, production 계정에서 role을 assume 함

(2) AWS STS는 임시 credential을 return 함

(3) IAM user는 임시 credential을 사용하여, resource와 서비스에 접근함

참고) IAM role의 역할

IAM role 이란? (2/3)

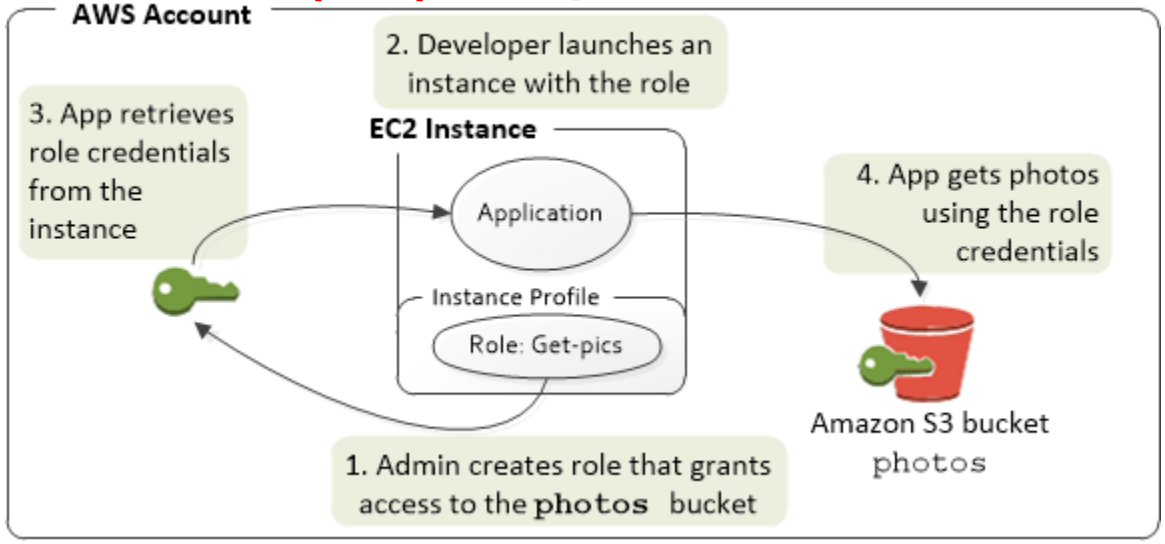
- An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS.

- However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials (password or access keys) associated with it. Instead, if a user assumes a role, temporary security credentials are created dynamically and provided to the user.

- You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account.

참고) IAM role의 역할

IAM role 이란? (3/3) - 사용예

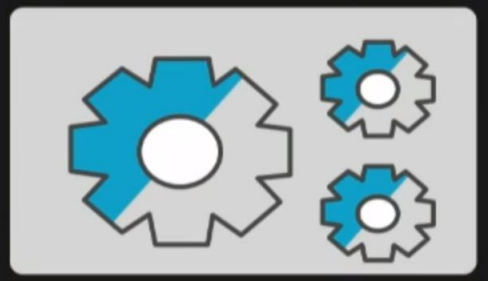


1. 관리자는 IAM을 사용하여 photo를 읽는 role (get-pics)을 만듦.
   - 관리자는 trust 정책에 EC2 인스턴스만이 role을 가질 수 있도록 지정하고 photos 버킷에서 사진을 읽기만 할 수 있도록 함
2. 개발자는 EC2 인스턴스를 시작하고 이 인스턴스에 get-pics role을 할당함
3. App이 실행되면 app은 EC2 인스턴스에서 "임시 보안 자격 증명" 을 가져옴. 임시 보안 자격 증명은 제한된 시간 동안만 유효
4. App은 "임시 보안 자격 증명" 을 사용하여 사진이 있는 bucket에 접근함. 애플리케이션은 읽기 권한만 있음

AWS IoT Rules Engine

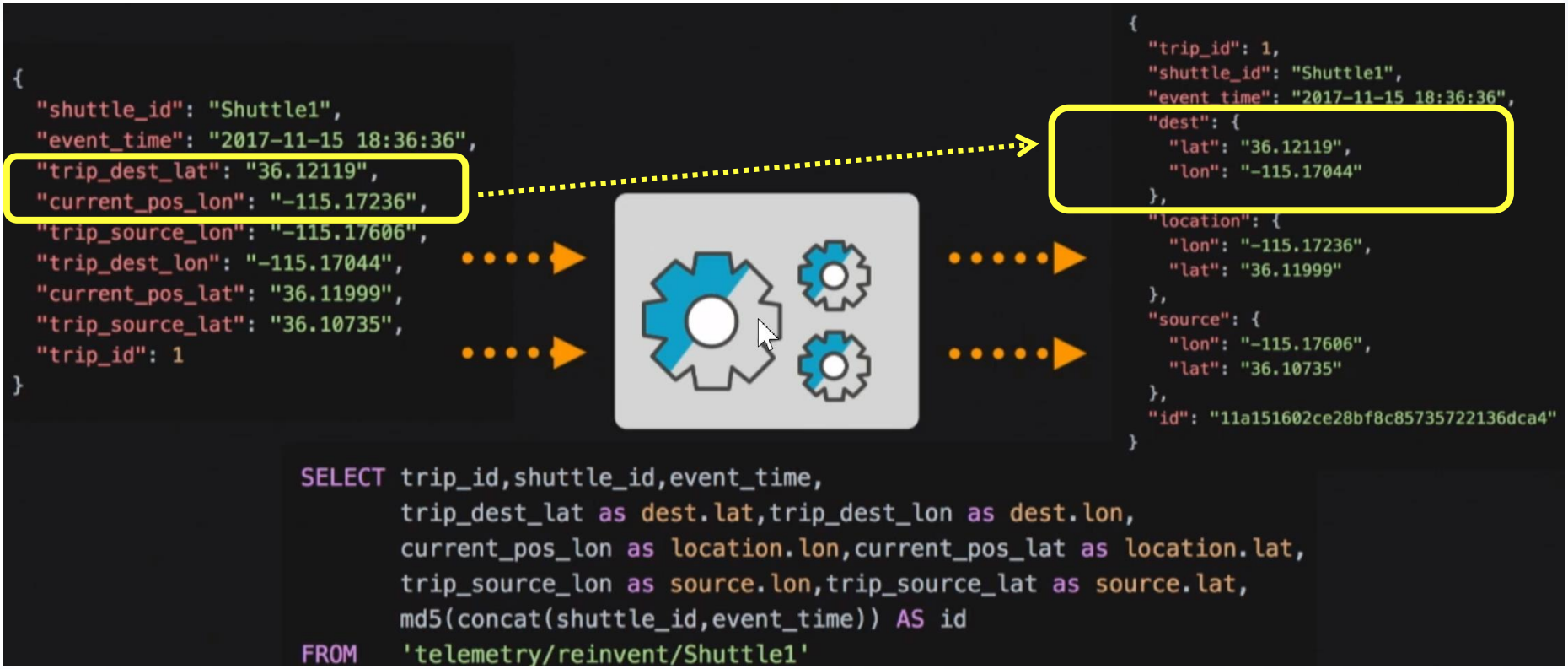
Rule 설명

- SQL
  - SELECT * FROM *topic* WHERE *condition*
  - SELECT *status.space_id AS room_id* FROM *'iot/tempSensors/#'* WHERE *temp > 50*
- Functions (in SELECT or WHERE)
  - String and mathematical operations
  - Get Shadow data
  - AML predict function
  - UUID, timestamp, rand, and so on



AWS IoT Rules Engine

- Rule 설명
- Substitution template
  - …, trip_dest_last as dest.lat, trip_dest_lon as dest.lon, …
    - trip_dest_last 값 → dest.lat
    - Trip_dest_lon 값 → dest.lon



감사합니다

Q & A

부산대학교 전기컴퓨터공학부 김호원
부산대학교 사물인터넷 연구센터장
howonkim@pusan.ac.kr

참고 사항

- 역할(role) …
  - Creating a Role to Delegate Permissions to an AWS Service.
  - role을 만드는 이유는 AWS 서비스에서 접근 권한을 위임하기 위한 목적임
  -
  - Many AWS services require that you use roles to allow the service to access resources in other services on your behalf.
  - A role that a service assumes to perform actions on your behalf is called a service role.

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-service.html

참고 사항

역할(role) …

- 서비스에 대한 역할 생성(AWS CLI)

AWS CLI에서 역할을 만들려면 여러 단계를 거쳐야 합니다. 콘솔을 사용하여 역할을 만들 때는 많은 단계가 자동으로 수행되지만 AWS CLI를 사용하면 각 단계를 직접 명시적으로 수행해야 합니다. 역할을 만든 다음 권한 정책을 역할에 할당해야 합니다. 작업 중인 서비스가 Amazon EC2인 경우에도 인스턴스 프로파일을 만들어 거기에 역할을 추가해야 합니다. 선택적으로 역할에 대한 권한 경계를 설정할 수 있습니다.

AWS CLI에서 AWS 서비스에 대한 역할을 만들려면

1. 역할 생성: aws iam create-role
2. 역할에 관리형 권한 정책 연결: aws iam attach-role-policy

   또는

   역할을 위한 인라인 권한 정책 생성: aws iam put-role-policy
3. (선택 사항) 역할(aws iam put-role-permissions-boundary)에 대한 권한 경계를 설정합니다.

   이 권한 경계는 역할이 가질 수 있는 최대 권한을 관리합니다. 권한 경계는 고급 AWS 기능입니다.

Amazon EC2 또는 Amazon EC2를 사용하는 다른 AWS 서비스에 대해 역할을 사용할 경우 인스턴스 프로파일에 역할을 저장해야 합니다. 인스턴스 프로파일은 시작할 때 Amazon EC2 인스턴스에 연결할 수 있는 역할을 위한 컨테이너입니다. 하나의 인스턴스 프로파일은 하나의 역할만 포함할 수 있으며 이 제한은 늘릴 수 없습니다. AWS Management 콘솔을 사용하여 역할을 생성한 경우 역할과 동일한 이름을 지닌 인스턴스 프로파일이 자동으로 생성됩니다. 인스턴스 프로파일에 대한 자세한 내용은 인스턴스 프로파일 사용 단원을 참조하십시오. 역할을 사용하여 EC2 인스턴스를 시작하는 방법에 대한 자세한 내용은 *Linux 인스턴스용 Amazon EC2 사용 설명서*에서 Amazon EC2 리소스에 대한 액세스 제어를 참조하십시오.