

AWS 활용 IoT

[3] AWS IoT : Provisioning



강사 : 고병화

실습 AWS 리소스 준비

실습용 AWS 리소스 생성

<https://catalog.us-east-1.prod.workshops.aws/workshops/7c2b04e7-8051-4c71-bc8b-6d2d7ce32727/en-US/launch-workshop-resources-with-cloudformation/launch-cloudformation-stack>

위 링크 주소에서 [Launch CloudFormation stack in us-east-1](#) (N. Virginia)를 찾아 링크 아이콘을 눌러 CloudFormation 스택 생성을 수행한다

Launch a CloudFormation stack

Choose an AWS region below where you want to launch your CloudFormation stack.
By choosing one of the links below you will be automatically redirected to the console where the stack will be launched.

- [Launch CloudFormation stack in eu-central-1](#)  (Frankfurt)
- [Launch CloudFormation stack in eu-west-1](#)  (Ireland)
- [Launch CloudFormation stack in eu-west-2](#)  (London)
- [Launch CloudFormation stack in us-east-1](#)  (N. Virginia)
- [Launch CloudFormation stack in us-west-2](#)  (Oregon)
- [Launch CloudFormation stack in ap-southeast-2](#)  (Sydney)
- [Launch CloudFormation stack in ap-northeast-1](#)  (Tokyo)

실습용 AWS 리소스 생성

AWS CloudFormation으로 실습에 필요한 리소스 생성
CloudFormation Stack은 다음 리소스를 생성한다

- (1) **AWS Cloud9 인스턴스** : 터미널 액세스, 편집기,
aws cli 를 제공
- (2) 보안 터널링을 위한 **Amazon EC2 인스턴스**
- (3) 대량 프로비저닝에 필요한 **s3 버킷**
- (4) **퍼블릭 서브넷이 있는 VPC + 보안 그룹**
- (5) Cloud9 및 EC2 인스턴스용 **인스턴스 프로필**
- (6) 프로비저닝 시나리오에 필요한 **IAM 역할**

실습용 AWS 리소스 생성

약 10분 정도
기다리면 5개
스택의 상태가
“CREATE_COMPLETE” 으로 모두
바뀌고 리소스
생성이 완료 된다

CloudFormation > 스택

스택 (5)

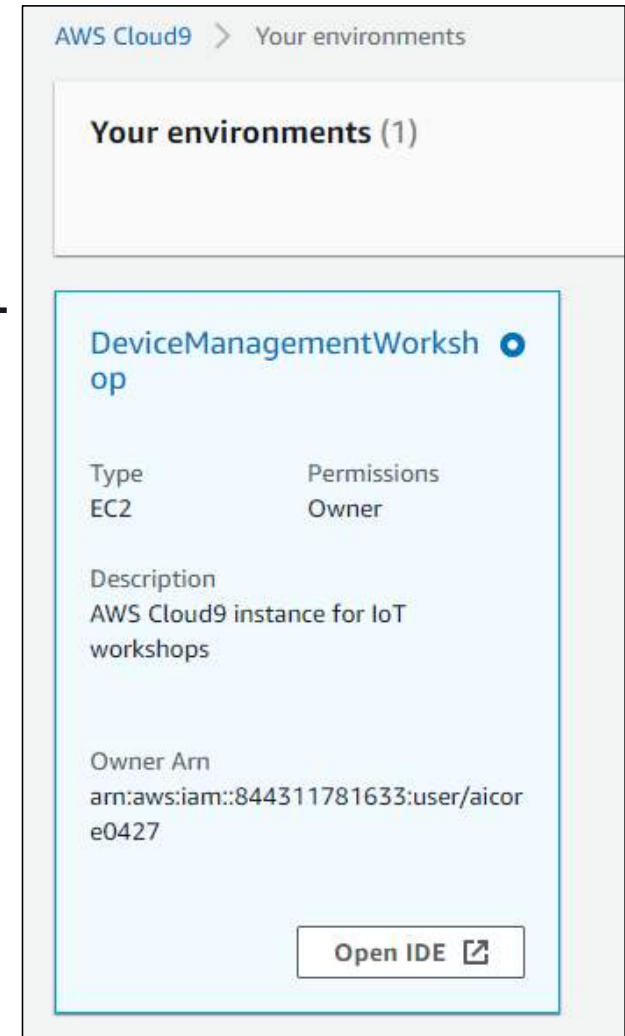
Q 스택 이름으로 필터링 뷰 중첩됨

스택 이름	상태	생성 시간
<input type="radio"/> aws-cloud9-DeviceManagementWorkshop-289d2faf6b2c421ab00d34ddb1b5bb73	✔ CREATE_COMPLETE	2022-09-12 00:28:54 UTC+0900
<input type="radio"/> DeviceManagementWorkshop-C9Instance-1VU7Z58762H2K 중첩	✔ CREATE_COMPLETE	2022-09-12 00:28:45 UTC+0900
<input type="radio"/> DeviceManagementWorkshop-EC2TunnelInstance-Q4MIMTLLHRX0 중첩	✔ CREATE_COMPLETE	2022-09-12 00:28:35 UTC+0900
<input type="radio"/> DeviceManagementWorkshop-MiscResources-15YW8JJX46TKK 중첩	✔ CREATE_COMPLETE	2022-09-12 00:27:13 UTC+0900
<input type="radio"/> DeviceManagementWorkshop	✔ CREATE_COMPLETE	2022-09-12 00:27:07 UTC+0900

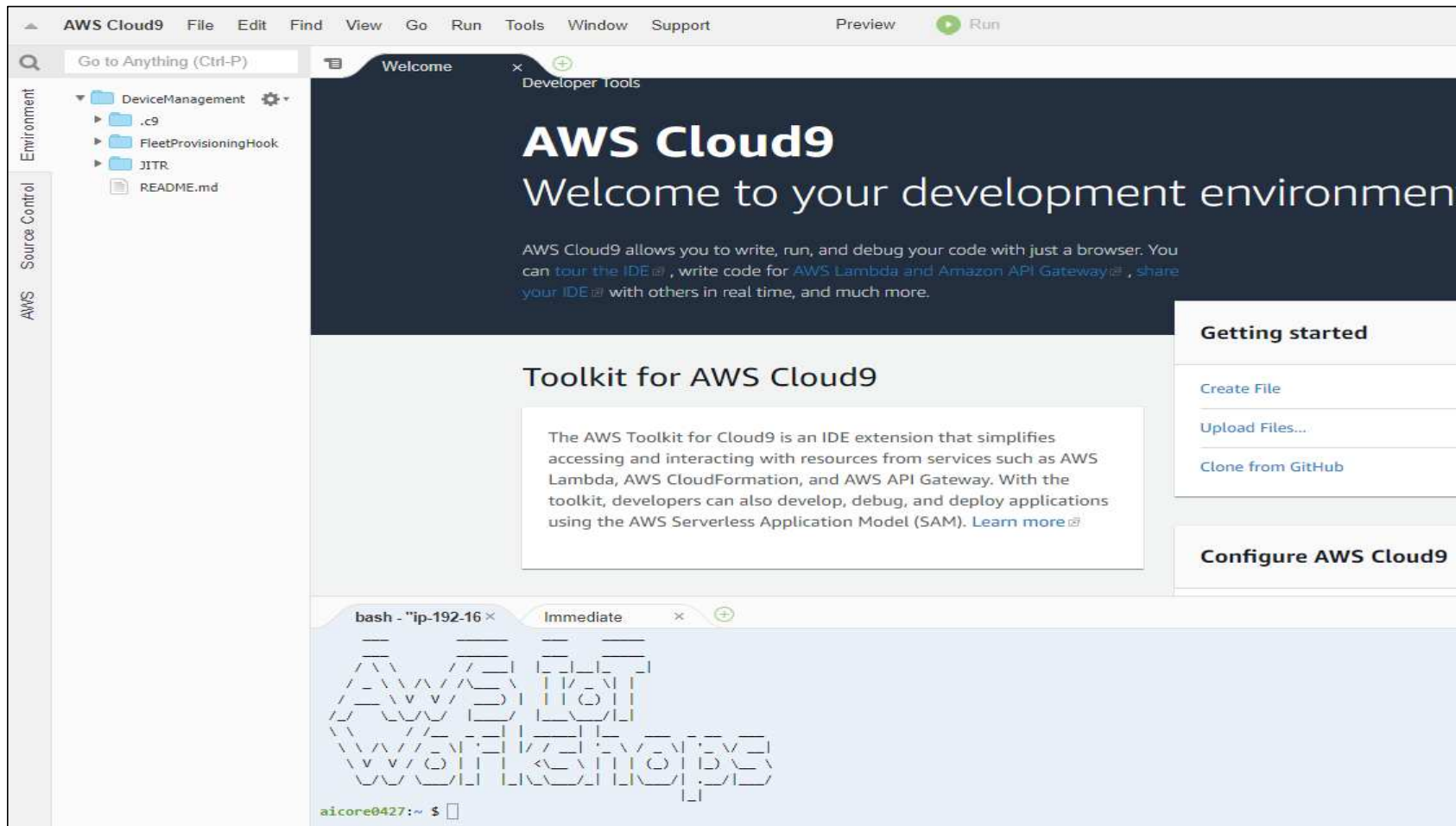
Cloud9 실행

AWS 콘솔의 좌측 상단 [서비스] 에서
[개발자 도구]→“Cloud9”을 클릭한다

하단의 [Open IDE]버튼을 클릭한다



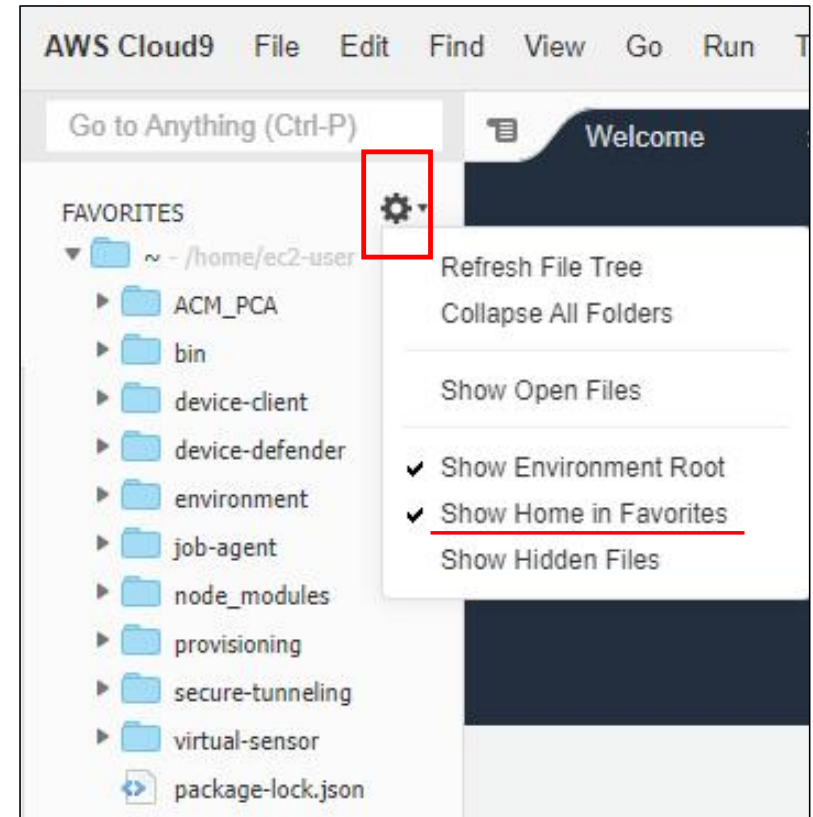
Cloud9 실행



Cloud9 실행

좌측 상단의 기어 모양 아이콘을 클릭하고 **[Show Home in Favorites]** 를 체크해주면 ec2-user의 홈 폴더를 볼 수 있게 된다

가상 머신 사용자 계정 이름은 “**ec2-user**”이다



Cloud9 실행

하단의 터미널에서 실습 중 각종 명령을 실행하게 된다

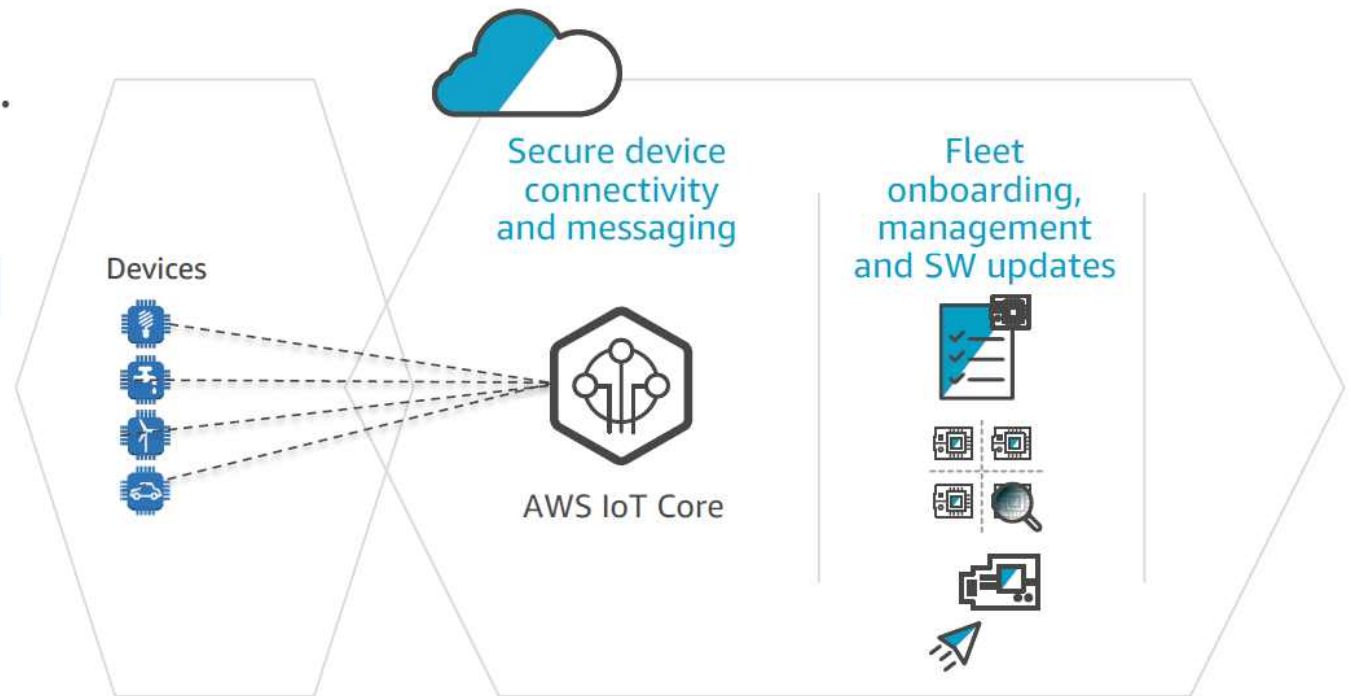
[illegible]

AWS IoT : Provisioning

At xcale - Howto provision devices?

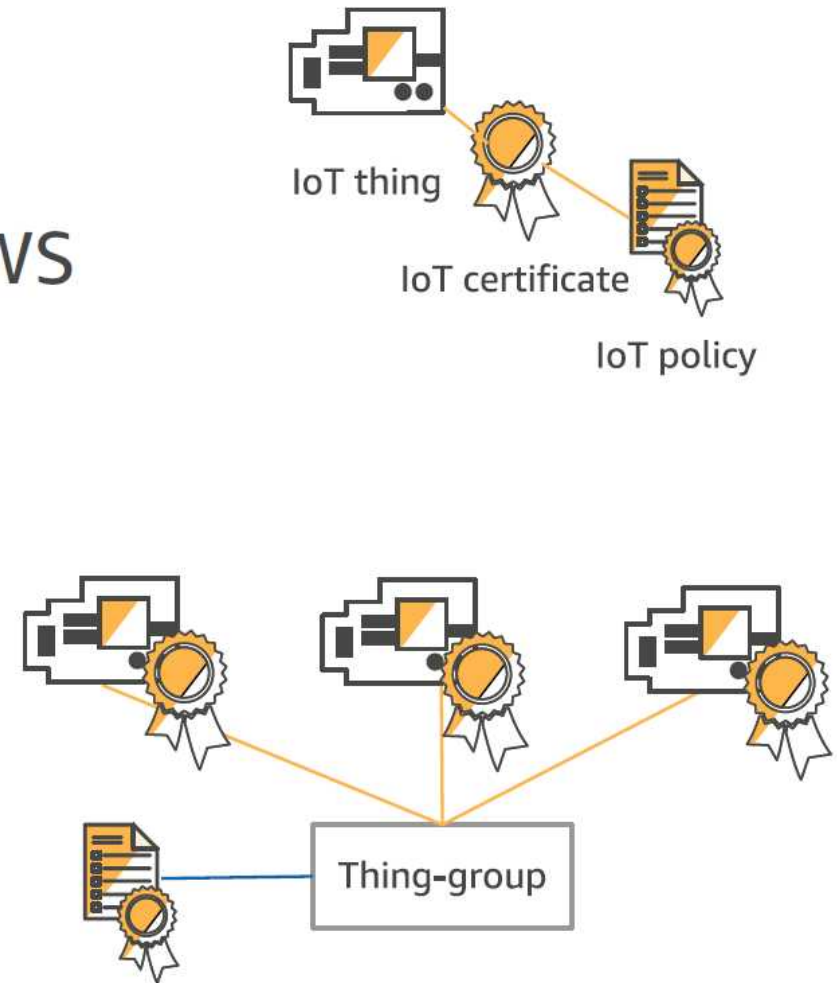
Architecture is developed...

How Do I onboard my devices???



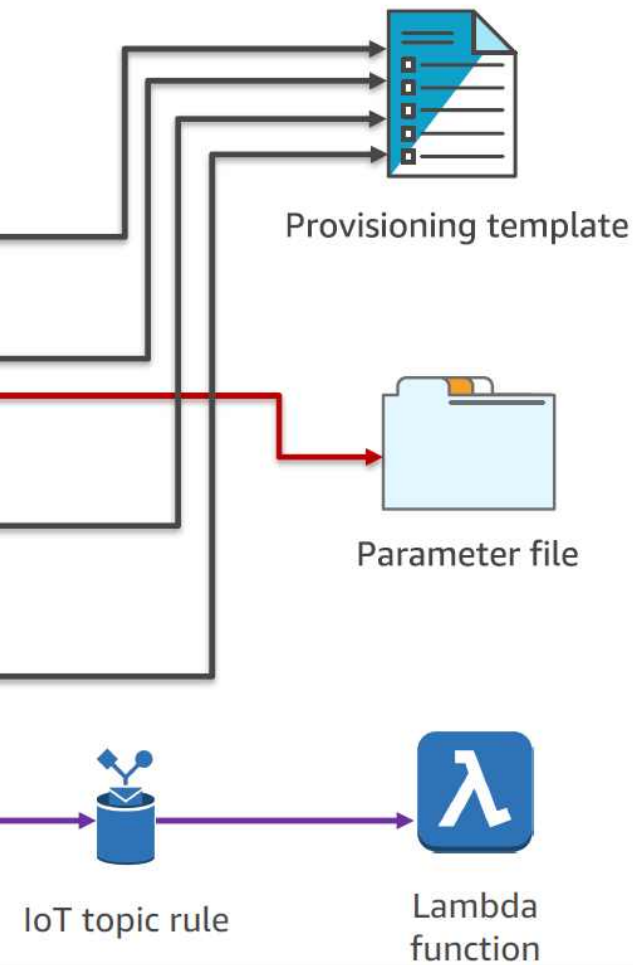
When a device is provisioned

- (Created in the device registry)
- Device certificate registered with AWS IoT Core
- (Certificate attached to the device)
- IoT Policy attached to the device through:
 - Certificate
 - Thing group



AWS IoT provisioning options

- API Calls
- Single Device Provisioning
- Bulk Device Provisioning
- Fleet Provisioning
- Just-in-Time Provisioning
- Just-in-Time Registration



Provisioning template

```
"Parameters" : {  
  "ThingName" : {  
    "Type" : "String",  
    "Required" : true,  
    "Default" : "lightbulb" },  
  "SerialNumber" : {  
    "Type" : "String",  
    "Required" : true,  
    "Default" : "1234567890" },  
  "Location" : {  
    "Type" : "String",  
    "Required" : true,  
    "Default" : "WA" },  
  "CSR" : {  
    "Type" : "String",  
    "Required" : true,  
    "Default" : "" }  
},  
  
"Resources" : {  
  "thing" : {  
    "Type" : "AWS::IoT::Thing",  
    "Properties" : {  
      "ThingName" : {"Ref" : "ThingName"},  
      "AttributePayload" : {  
        "version" : "v1",  
        "serialNumber" : {"Ref" : "SerialNumber"}  
      },  
      "ThingTypeName" : "lightBulb-versionA",  
      "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]  
    }  
  },  
  
  "certificate" : {  
    "Type" : "AWS::IoT::Certificate",  
    "Properties" : {  
      "CertificateSigningRequest": {"Ref" : "CSR"},  
      "Status" : "ACTIVE"  
    }  
  }  
}
```

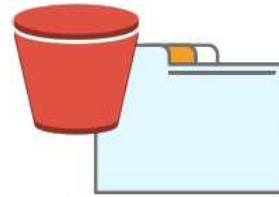

Single/Bulk device provisioning

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"} {"ThingName":  
"bar", "SerialNumber": "456", "CSR": "csr2"}
```

Parameters



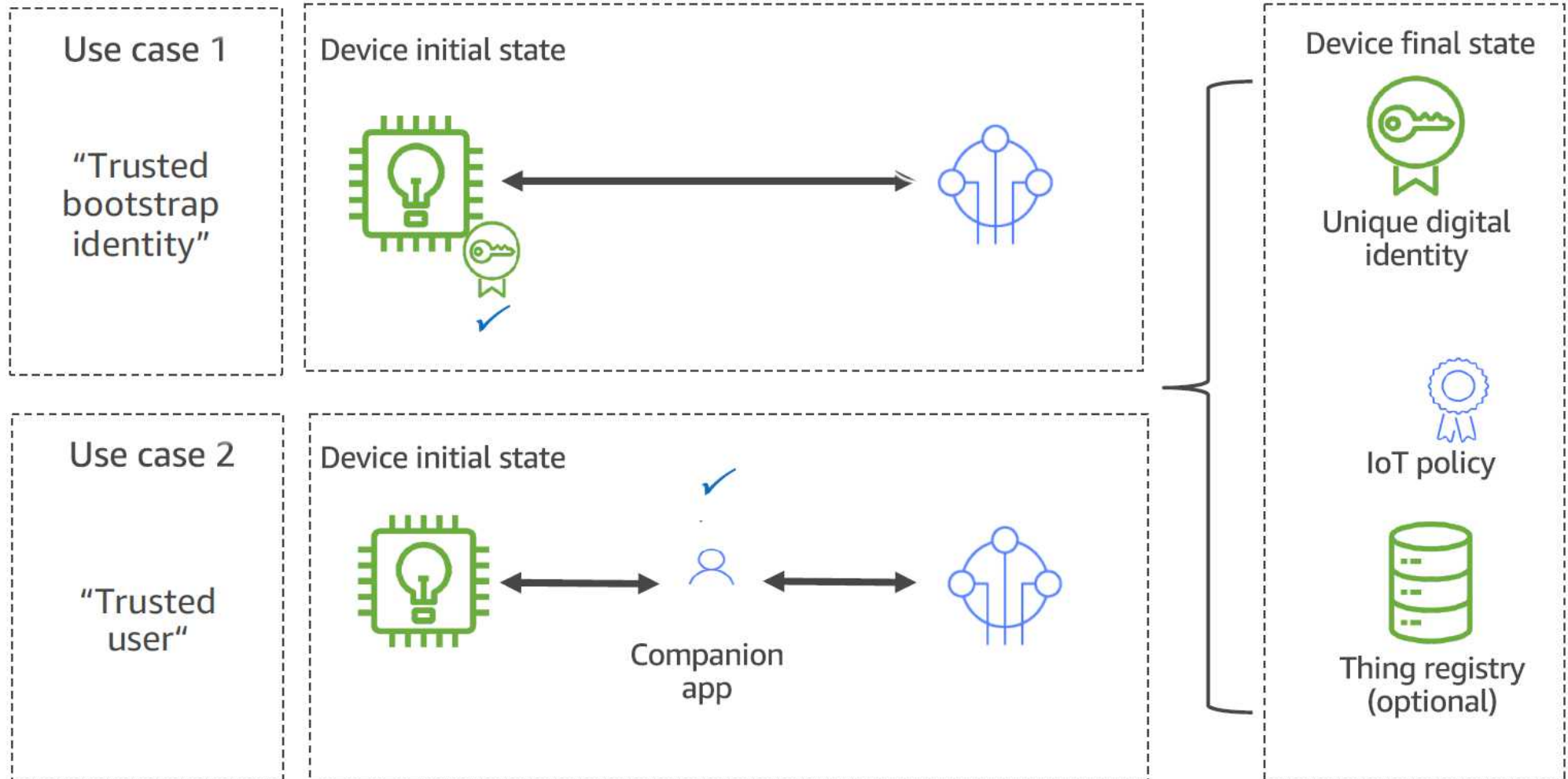
Provisioning template



Parameter file
on S3 Bucket (bulk)

- Parameters with device information are used in the provisioning template
- Single: on "line" as parameter to register a thing
- Bulk: multiple parameter lines in an S3 bucket

Fleet provisioning: How it works



Fleet provisioning



Provisioning template

- Create a provisioning template
 - Optional Lambda-based pre-provisioning hook
- Create provisioning claim key/certificate
- Attach restricted policy to the claim certificate
- Device connects for the first time with claim key/certificate
- Uses the provisioning MQTT API to obtain final certificate and being provisioned in AWS IoT
 - `$aws/provisioning-templates/templateName/provision/payload-format`
 - `$aws/provisioning-templates/templateName/provision/payload-format/#`
 - `$aws/certificates/create-from-csr/payload-format/#`
 - `$aws/certificates/create/payload-format/#`

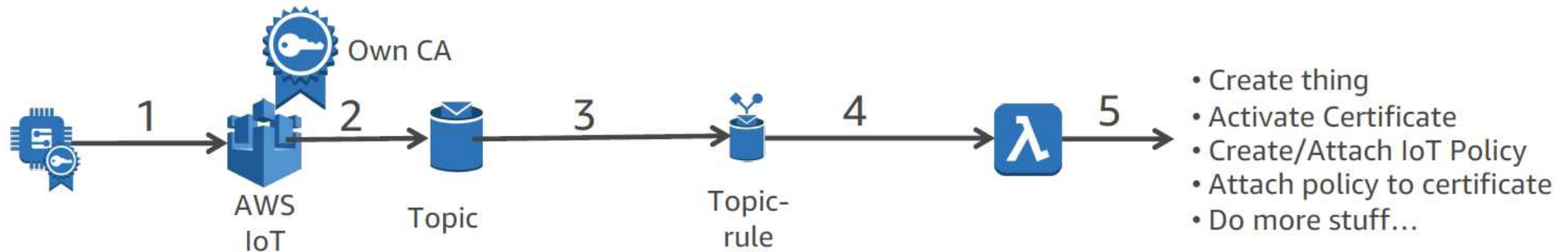
https://docs.aws.amazon.com/ko_kr/iot/latest/developerguide/provision-wo-cert.html

Device Onboarding – JITP



- Own CA required
 - Provisioning Template attached to own CA
1. Device connects to AWS IoT, device certificate gets registered
 2. JITP provisions device according to the provisioning template

Device Onboarding – JITR

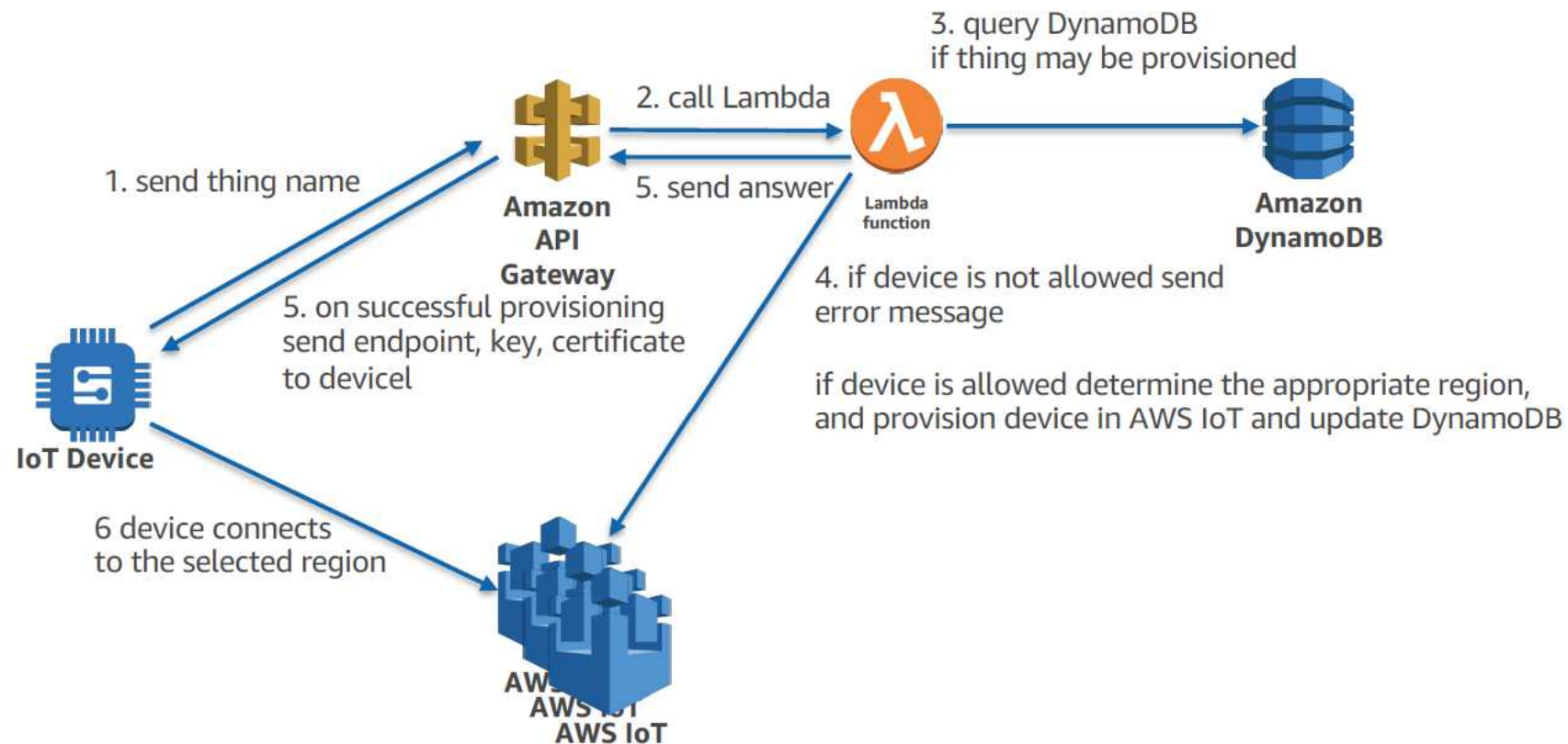


1. Device connects to AWS IoT, device certificate gets registered
2. AWS IoT publishes message to `$aws/events/certificates/registered/<caCertificateID>`
3. Topic Rule is invoked
4. Topic Rule calls Lambda Function as action
5. Lambda provisions device

JITR vs. JITP

JITR	JITP
Topic rule and Lambda function. Code must be written and maintained	No code, only body template attached to CA
Provisioning more complex: Device connects, certificate registers with status PENDING_ACTIVATION, service sends MQTT message, rule triggers Lambda, Lambda does provisioning and optionally more stuff	Easy provisioning: Device connects, provisioning workflow run automatically
Flexible, different policies for different devices can be created/attached. Information from/to the provisioning process can be put/read from other systems, etc.	Static, same provisioning process for every device

Global Device Provisioning



Optional Exercise:

<https://aws.amazon.com/blogs/iot/provision-devices-globally-with-aws-iot/>

IoT Provisioning : [1] API 프로비저닝

Cloud9 터미널에 아래 명령을 차례로 입력하여 Thing을 생성하고 등록한다

실습 디렉토리로 이동

```
cd ~/provisioning
```

Thing 생성

```
THING_NAME=my-first-thing
```

```
aws iot create-thing --thing-name $THING_NAME
```

[링크] <http://aws-workshops-1589389556.eu-west-1.elb.amazonaws.com/iot-device-management-workshop/provisioning-options/provisioning-with-api.html>

IoT Provisioning : [1] API 프로비저닝

key 생성

```
aws iot create-keys-and-certificate --set-as-active \  
  --public-key-outfile $THING_NAME.public.key \  
  --private-key-outfile $THING_NAME.private.key \  
  --certificate-pem-outfile $THING_NAME.certificate.pem >  
/tmp/create_cert_and_keys_response  
cat /tmp/create_cert_and_keys_response
```

```
CERTIFICATE_ARN=$(jq -r ".certificateArn"  
/tmp/create_cert_and_keys_response)  
CERTIFICATE_ID=$(jq -r ".certificateId" /tmp/create_cert_and_keys_response)  
echo $CERTIFICATE_ARN  
echo $CERTIFICATE_ID
```

IoT Provisioning : [1] API 프로비저닝

```
POLICY_NAME=${THING_NAME}_Policy
```

```
aws iot create-policy --policy-name $POLICY_NAME \  
  --policy-document '{"Version":"2012-10-17",  
"Statement":[{"Effect":"Allow","Action": "iot:*","Resource":"*"}]}'
```

```
aws iot attach-policy --policy-name $POLICY_NAME \  
  --target $CERTIFICATE_ARN
```

```
aws iot attach-thing-principal --thing-name $THING_NAME \  
  --principal $CERTIFICATE_ARN
```


IoT Provisioning : [1] API 프로비저닝

등록된 Thing 목록 출력

```
aws iot list-things
```

디바이스 업데이트

```
aws iot update-thing --thing-name $THING_NAME --attribute-payload
```

```
'{"attributes": {"type": "ws-device"}}'
```

```
aws iot list-things
```

IoT Provisioning : [1] API 프로비저닝

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 “iot/ws” 토픽으로 구독한다

주제 구독

주제 게시

주제 필터

정보

주제 필터는 구독할 주제를 설명합니다. 주제 필터에는 MQTT 와일드카드 문자가 포함될 수 있습니다.

iot/ws

▶ 추가 구성

구독

구독

iot/ws

iot/ws

♡ ✕

▼ iot/ws

IoT Provisioning : [1] API 프로비저닝

endpoint 주소 확인

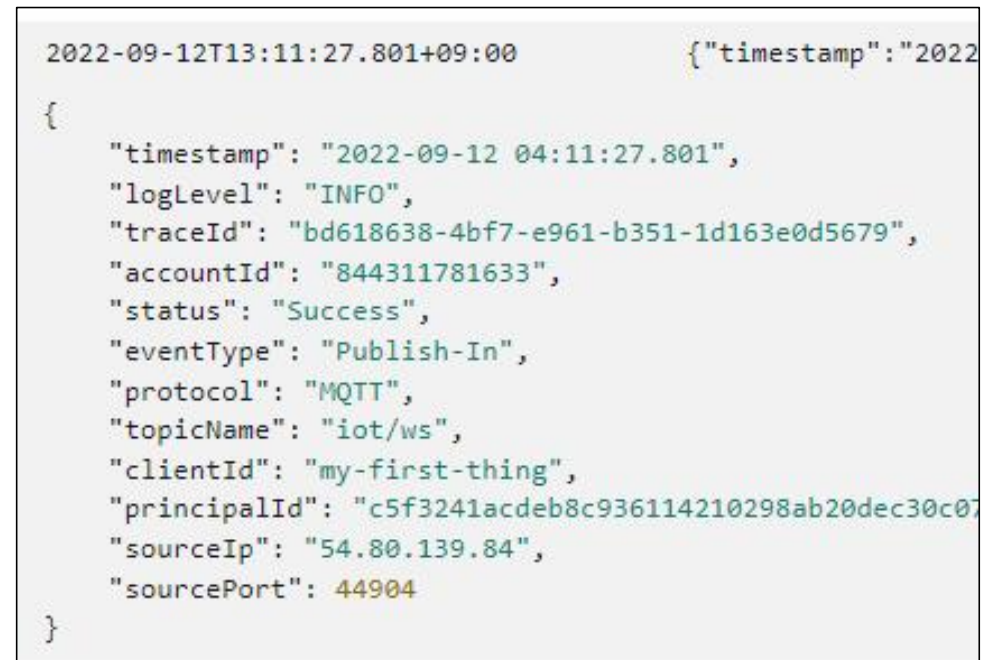
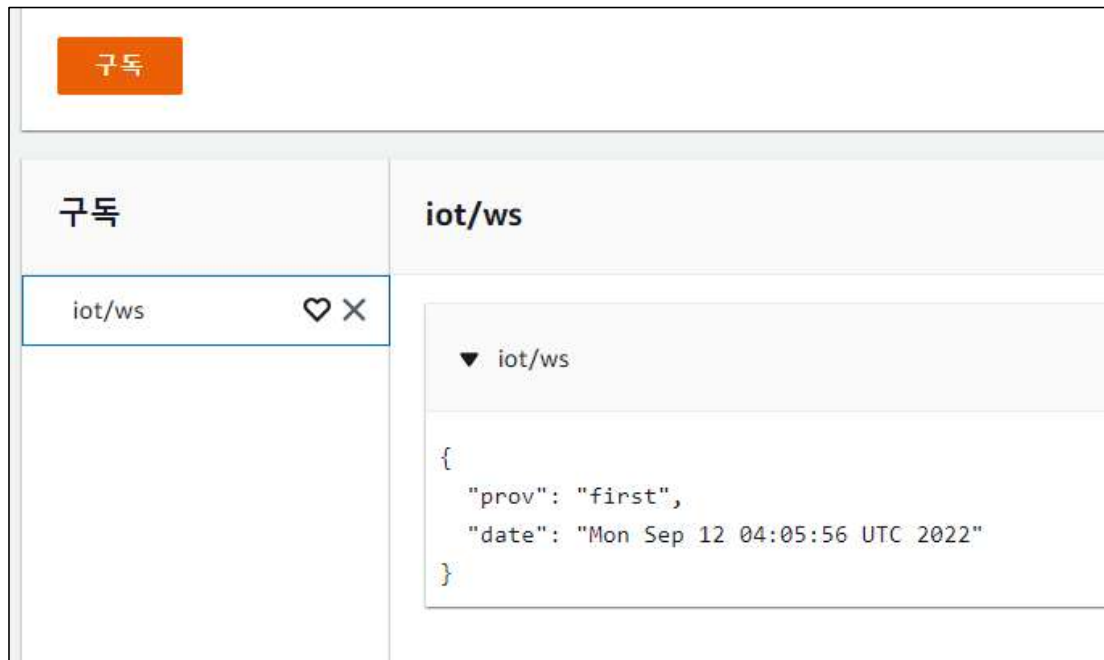
```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

publish a message

```
mosquitto_pub --cafile ~/root.ca.bundle.pem \  
--cert $THING_NAME.certificate.pem \  
--key $THING_NAME.private.key -h $IOT_ENDPOINT -p 8883 \  
-q 0 -t iot/ws -i $THING_NAME --tls-version tlsv1.2 \  
-m "{\"prov\": \"first\", \"date\": \"$(date)\"}" -d
```

IoT Provisioning : [1] API 프로비저닝

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 구독 메시지를 확인한다



IoT Provisioning : [2] 단일 장치 프로비저닝

Cloud9 터미널에 아래 명령을 차례로 입력한다

thing 그룹 생성

```
aws iot create-thing-group --thing-group-name bulk-group
```

thing 타입 생성

```
aws iot create-thing-type --thing-type-name bulk-type
```

#key, CSR 과 parameters 생성

```
THING_NAME=my-second-thing
```

```
mk-prov.sh $THING_NAME
```

```
aicore0427:~/provisioning $ sudo find / -name "mk-prov.sh"
/home/ec2-user/bin/mk-prov.sh
aicore0427:~/provisioning $ cat ~/bin/mk-prov.sh
#!/bin/bash
```

IoT Provisioning : [2] 단일 장치 프로비저닝

`aws iot register-thing --template-body file://~/templateBody.json --parameters`
'앞의 출력 전체를 중괄호 포함하여 복사하여 넣는다'

`echo -e "-----BEGIN CERTIFICATE-----` 앞 명령 출력에서 해당 부분만 복사해서
`넣는다-----END CERTIFICATE-----\n" > $THING_NAME.crt`

MQTT 메시지 publish

```
mosquitto_pub --cafile ~/root.ca.bundle.pem \  
  --cert $THING_NAME.crt --key $THING_NAME.key \  
  -h $IOT_ENDPOINT -p 8883 -q 0 -t iot/ws \  
  -i $THING_NAME --tls-version tlsv1.2 \  
  -m "{\"prov\": \"second\", \"date\": \"$(date)\"}" -d
```



IoT Provisioning : [2] 단일 장치 프로비저닝

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 구독 메시지를 확인한다

iot/ws

▶ 추가 구성

구독

구독	iot/ws
iot/ws  	<div>▼ iot/ws</div> <pre>{ "prov": "second", "date": "Mon Sep 12 04:33:02 UTC 2022" }</pre>

IoT Provisioning : [3] 대량(Bulk) 프로비저닝

Cloud9 터미널에 아래 명령을 차례로 입력하여 Thing을 생성하고 등록한다

```
THING_NAME=bulky  
NUM_THINGS=10  
mk-bulk.sh $THING_NAME $NUM_THINGS
```

새로 생성된 디렉토리로 이동한다(생성한 날짜 값으로 자동 생성됨)

```
cd bulky-2022-09-12_05-17-59  
aws s3 cp bulk.json s3://$S3_BUCKET/  
aws s3 ls s3://$S3_BUCKET/
```


IoT Provisioning : [3] 대량(Bulk) 프로비저닝

```
aws iot start-thing-registration-task \  
  --template-body file://~/templateBody.json \  
  --input-file-bucket $S3_BUCKET \  
  --input-file-key bulk.json --role-arn $ARN_IOT_PROVISIONING_ROLE
```

YOUR_TASK_ID 값은 앞 명령 실행 후 출력 된 것을 복사하여 사용한다

```
aws iot list-thing-registration-task-reports \  
  --report-type ERRORS --task-id [YOUR_TASK_ID]
```

```
aws iot list-thing-registration-task-reports \  
  --report-type RESULTS --task-id [YOUR_TASK_ID]
```

IoT Provisioning : [3] 대량(Bulk) 프로비저닝

AWS IoT 콘솔의 [모든 디바이스]의 [사물] 에 가보면 10개 thing이 등록 된 것을 볼 수 있다

사물 (14) 정보		
IoT 사물은 클라우드상 물리적 디바이스가 드러나는 물체이자 기록입니다. 물체와 함께 작동하려면 사물 레코드가 필요합니다.		
<input type="text" value="Q 사물을 이름, 유형, 그룹, 결제 또는 검색 가능한 속성을 기준"/>		
<input type="checkbox"/>	이름	사물 유형
<input type="checkbox"/>	bulky10	bulk-type
<input type="checkbox"/>	bulky9	bulk-type
<input type="checkbox"/>	bulky8	bulk-type
<input type="checkbox"/>	bulky7	bulk-type
<input type="checkbox"/>	bulky6	bulk-type
<input type="checkbox"/>	bulky5	bulk-type
<input type="checkbox"/>	bulky4	bulk-type
<input type="checkbox"/>	bulky3	bulk-type
<input type="checkbox"/>	bulky2	bulk-type
<input type="checkbox"/>	bulky1	bulk-type

IoT Provisioning : [3] 대량(Bulk) 프로비저닝

```
wget -O results.json $(aws iot list-thing-registration-task-reports --task-id  
[YOUR_TASK_ID] --report-type RESULTS | jq -r '.resourceLinks[]')
```

오류 발생시만 실행한다

```
# wget -O errors.json $(aws iot list-thing-registration-task-reports --task-id  
[YOUR_TASK_ID] --report-type ERRORS | jq -r '.resourceLinks[]' )
```

```
bulk-result.py results.json
```

```
ls -l
```

```
aws iot list-things
```

IoT Provisioning : [3] 대량(Bulk) 프로비저닝

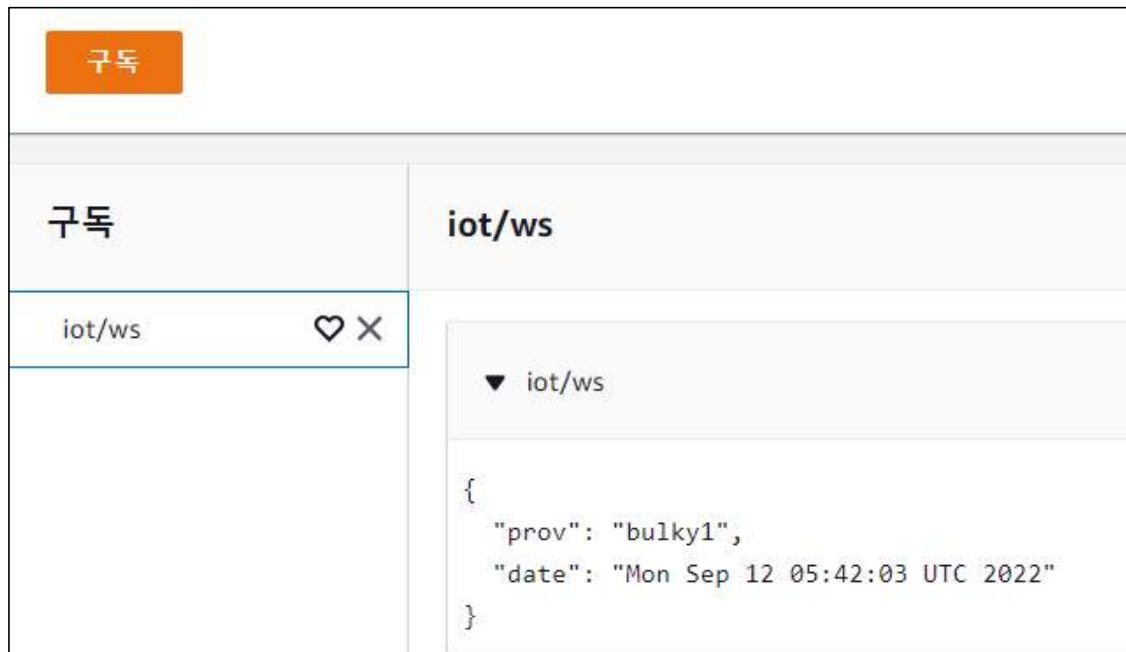
“iot/ws”를 구독하고 buky1을 하나만 테스트 해본다

THING_NAME=bulky1

```
mosquitto_pub --cafile ~/root.ca.bundle.pem \  
--cert $THING_NAME.crt --key $THING_NAME.key \  
-h $IOT_ENDPOINT -p 8883 -q 0 -t iot/ws \  
-i $THING_NAME --tls-version tlsv1.2 \  
-m "{\"prov\": \"bulk\", \"date\": \"$(date)\"}" -d
```

IoT Provisioning : [3] 대량(Bulk) 프로비저닝

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 구독 메시지를 확인한다



IoT Provisioning : [3] 대량(Bulk) 프로비저닝

#thing 10 개 모두 테스트 해본다

```
for i in {1..10}; do
THING_NAME=bulky$i
mosquitto_pub --cafile ~/root.ca.bundle.pem \
--cert $THING_NAME.crt --key $THING_NAME.key \
-h $IOT_ENDPOINT -p 8883 -q 0 -t iot/ws \
-i $THING_NAME --tls-version tlsv1.2 \
-m "{\"prov\": \"bulky$i\", \"date\": \"$(date)\"}" -d
echo $i
done
```

IoT Provisioning : [3] 대량(Bulk) 프로비저닝

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 구독 메시지를 확인한다

```
▼ iot/ws
{
  "prov": "bulky10",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}

▼ iot/ws
{
  "prov": "bulky9",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}

▼ iot/ws
{
  "prov": "bulky8",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}
```

```
▼ iot/ws
{
  "prov": "bulky7",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}

▼ iot/ws
{
  "prov": "bulky6",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}

▼ iot/ws
{
  "prov": "bulky5",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}
```

```
▼ iot/ws
{
  "prov": "bulky3",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}

▼ iot/ws
{
  "prov": "bulky2",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}

▼ iot/ws
{
  "prov": "bulky1",
  "date": "Mon Sep 12 05:43:54 UTC 2022"
}
```

IoT Provisioning : [4] Fleet 프로비저닝

디바이스 인증서가 없는 경우 디바이스 프로비저닝을 위해서 다음 두 가지의 **플릿 프로비저닝**을 사용할 수 있다

[1] 신뢰할 수 있는 사용자(trusted user)에 의한 플릿 프로비저닝

최종 사용자에게 전달되기 전에 고유한 클라이언트 인증서를 IoT 디바이스에 안전하게 설치할 수 없지만

최종 사용자 또는 설치 관리자가 앱을 사용하여 디바이스를 등록하고 고유한 디바이스 인증서를 설치할 수 있는 경우에 사용

[2] 클레임(claim)에 의한 플릿 프로비저닝 (실습에서 사용)

디바이스에 인증서를 설치할 수 없고 최종 사용자가 앱을 사용하여 IoT 디바이스에 인증서를 설치할 수 없는 경우에 사용.

https://docs.aws.amazon.com/ko_kr/iot/latest/developerguide/provision-wo-cert.html

IoT Provisioning : [4] Fleet 프로비저닝

Cloud9 터미널에 아래 명령을 차례로 입력한다

```
cd ~/provisioning/fleet-provisioning
```

사물 그룹 생성

```
aws iot create-thing-group --thing-group-name fleet-provisioning-group
```

프로비저닝 템플릿 생성

```
aws iot create-provisioning-template \  
  --template-name FleetProvisioningTemplate \  
  --provisioning-role-arn $ARN_IOT_PROVISIONING_ROLE \  
  --template-body file:///./fleet-provisioning-template.json \  
  --enabled
```

IoT Provisioning : [4] Fleet 프로비저닝

```
aws iot list-provisioning-templates
```

```
aws iot describe-provisioning-template --template-name  
FleetProvisioningTemplate
```

클레임 인증서 및 키 생성

```
THING_NAME=provision-claim
```

```
aws iot create-keys-and-certificate --set-as-active \  
--public-key-outfile $THING_NAME.public.key \  
--private-key-outfile $THING_NAME.private.key \  
--certificate-pem-outfile $THING_NAME.certificate.pem > provisioning-claim-  
result.json
```

IoT Provisioning : [4] Fleet 프로비저닝

```
CERTIFICATE_ARN=$(jq -r ".certificateArn" provisioning-claim-result.json)
```

```
# 클레임 인증서에 대한 IoT 정책 만들기
```

```
aws iot create-policy --policy-name fleet-provisioning_Policy \  
    --policy-document file:///./fleet-provisioning-policy.json
```

```
# 클레임 인증서에 정책 첨부
```

```
aws iot attach-policy --policy-name fleet-provisioning_Policy \  
    --target $CERTIFICATE_ARN
```

IoT Provisioning : [4] Fleet 프로비저닝

플릿 프로비저닝 프로세스 시작

```
./fleetprovisioning.py --endpoint $IOT_ENDPOINT \  
    --root-ca ~/root.ca.bundle.pem \  
    --cert ./provision-claim.certificate.pem \  
    --key ./provision-claim.private.key \  
    --client-id fleet-device \  
    --templateName FleetProvisioningTemplate \  
    --templateParameters  
"{\"SerialNumber\": \"297468\", \"DeviceLocation\": \"Berlin\"}"
```

```
aws iot describe-thing --thing-name fleety_297468
```

```
aws iot list-things-in-thing-group --thing-group-name fleet-provisioning-group
```

IoT Provisioning : [4] Fleet 프로비저닝

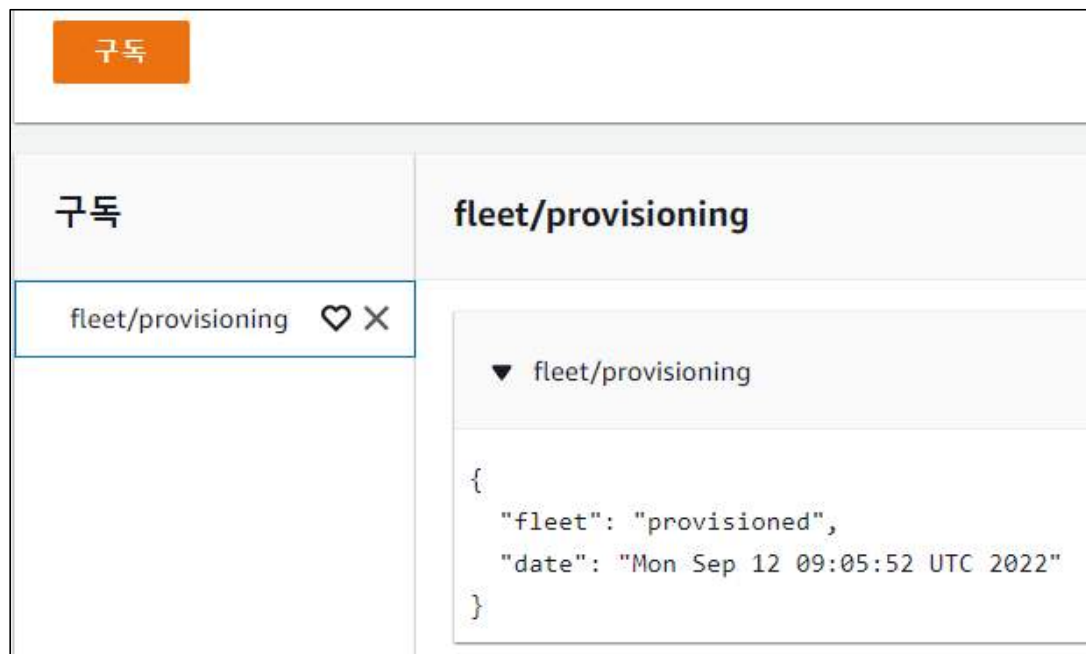
[MQTT 테스트 클라이언트]에서 : "fleet/provisioning" 을 주제로 구독한다

message publish

```
mosquitto_pub --cafile ~/root.ca.bundle.pem \  
--cert fleety_297468.certificate.pem \  
--key fleety_297468.private.key \  
-h $IOT_ENDPOINT -p 8883 -q 1 \  
-t fleet/provisioning \  
-i fleety_297468 \  
--tls-version tlsv1.2 \  
-m "{\"fleet\": \"provisioned\", \"date\": \"$(date)\"}" -d
```

IoT Provisioning : [4] Fleet 프로비저닝

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 구독 메시지를 확인한다



IoT Provisioning : [5] 나만의 CA 가져오기

AWS Certificate Manager 사설 인증 기관 요금 : **[주의 : 유료서비스임!!]**

ACM 사설 인증 기관(CA)은 두 가지 차원에서 요금이 책정됩니다.

고객은 각 사설 CA의 운영에 대해 해당 CA를 삭제할 때까지 월별 요금을 지불하고 매월 발급한 사설 인증서에 대해 요금을 지불합니다.

사설 인증 기관 운영

CA를 삭제할 때까지 각 ACM 프라이빗 CA에 대해 **월별 400.00 USD!!**

ACM 사설 CA 운영에는 CA를 생성하고 삭제하는 시점에 따라 부분 월에 대해 비례 할당으로 계산된 요금이 부과됩니다. 삭제한 후에는 사설 CA에 대한 요금이 부과되지 않습니다. 하지만 삭제된 CA를 복원하는 경우 삭제 시점과 복원 시점 사이의 시간에 대한 요금이 부과됩니다.

<https://aws.amazon.com/ko/certificate-manager/pricing/?nc=sn&loc=3>

IoT Provisioning : [5] 나만의 CA 가져오기

1일 경과 요금 : (비용 과다 발생으로 [5]~[7]번 까지 실습 불가!!)

청구된 요금	절감액	서비스별 요금	계정별 요금	서비스별 세금
Amazon Web Services Korea LLC 서비스별 요금 정보				
총 활성 서비스 11		다음의 총 세전 서비스 요금 USD USD 252.64		
<input type="text" value="서비스 이름으로 필터링"/>		< 1 >		
서비스 이름 리전		의 금액 USD		
Certificate Manager		USD 251.47		
US East (N. Virginia)		USD 251.47		

AWS Certificate Manager 사설 인증 기관 > 프라이빗 인증 기관					
프라이빗 인증 기관 (2)					
	ID	CA 일반 이름	소유자	조직	OU
<input type="radio"/>	8dd02b96-3b83-40a4-8a34-ffb980191a66	IoT Device Management CA	본인	IoT Device Management	IoT Operations
<input checked="" type="radio"/>	ac2dda53-13cd-40fd-ae1b-95e95d200c42	IoT Device Management CA	본인	IoT Device Management	IoT Operations

IoT Provisioning : [5] 나만의 CA 가져오기

Cloud9 터미널에 아래 명령을 차례로 입력한다

```
cd ~/ACM_PCA
```

```
aws acm-pca create-certificate-authority \  
    --certificate-authority-configuration file://./ca-config.json \  
    --revocation-configuration file://./revoke-config.json \  
    --certificate-authority-type "ROOT" \  
    --idempotency-token $(date +%Y%m%d%H%M%S')
```

앞 명령의 출력 값을 복사하여 넣는다

```
CA_ARN=arn:aws:acm-pca:us-east-1:844311781633:certificate-  
authority/ac2dda53-13cd-40fd-ae1b-95e95d200c42
```

```
aws acm-pca describe-certificate-authority --certificate-authority-arn $CA_ARN
```

IoT Provisioning : [5] 나만의 CA 가져오기

상태가 PENDING_CERTIFICATE 이 되므로 CA(인증기관) 설정을 해야 한다.

AWS Certificate Manager 콘솔로 이동한다

프라이빗 인증기관에서 IoT Device Management CA를 선택한다

상태가 보류중인 인증서로 되어 있다

[작업]의 [CA인증서 설치]를 클릭하고

유효성 값: 10 , 유효성 유형: 연수 , 서명 알고리즘: SHA256WTIHRSA

으로(기본값) 설정하고 [다음]버튼을 누른다

[확인 및 설치] 버튼을 누른다

Cloud9 터미널로 이동하여 다음 명령을 입력한다

IoT Provisioning : [5] 나만의 CA 가져오기

```
aws acm-pca get-certificate-authority-certificate --certificate-authority-arn  
$CA_ARN --output text > acm-pca-root-ca.pem
```

```
openssl x509 -text -noout -in acm-pca-root-ca.pem
```

```
REGISTRATION_CODE=$(aws iot get-registration-code --output text)
```

IoT Provisioning : [5] 나만의 CA 가져오기

```
echo $REGISTRATION_CODE
```

```
openssl req -nodes -new -newkey rsa:2048 \  
-keyout iot-registration.key \  
-out iot-registration.csr \  
-subj "/CN=$REGISTRATION_CODE"
```

IoT Provisioning : [5] 나만의 CA 가져오기

```
aws acm-pca issue-certificate \  
  --certificate-authority-arn $CA_ARN \  
  --csr file:///./iot-registration.csr \  
  --signing-algorithm "SHA256WITHRSA" \  
  --validity Value=1,Type="DAYS" \  
  --idempotency-token $(date +%Y%m%d%H%M%S')
```

certificate-arn 값으로 앞 명령의 출력 값을 복사하여 넣는다

```
aws acm-pca wait certificate-issued \  
  --certificate-authority-arn $CA_ARN \  
  --certificate-arn arn:aws:acm-pca:us-east-1:844311781633:certificate-  
authority/ac2dda53-13cd-40fd-ae1b-  
95e95d200c42/certificate/c44d5b855445a3440d61dec2ca7b1b12
```

IoT Provisioning : [5] 나만의 CA 가져오기

앞 명령의 출력 arn값을 복사하여 넣는다

```
aws acm-pca get-certificate \
```

```
--certificate-authority-arn $CA_ARN \
```

```
--certificate-arn arn:aws:acm-pca:us-east-1:844311781633:certificate-  
authority/ac2dda53-13cd-40fd-ae1b-  
95e95d200c42/certificate/c44d5b855445a3440d61dec2ca7b1b12 > iot-  
registration.json
```

```
jq -r '.Certificate' iot-registration.json > iot-registration.crt
```

```
openssl x509 -text -noout -in iot-registration.crt -subject
```

IoT Provisioning : [5] 나만의 CA 가져오기

```
aws iot register-ca-certificate \  
  --ca-certificate file:///./acm-pca-root-ca.pem \  
  --verification-cert file:///./iot-registration.crt
```

앞 명령의 출력 값을 복사하여 넣는다

```
CA_CERTIFICATE_ID=a8e917cb3552675fb8f1cd75a841dfa6c5441323d48e582a  
2a681c4d1f7e1416
```

```
aws iot describe-ca-certificate --certificate-id $CA_CERTIFICATE_ID
```

```
aws iot update-ca-certificate --new-status ACTIVE --certificate-id  
$CA_CERTIFICATE_ID
```

IoT Provisioning : [6] Just-in-time 프로비저닝

Cloud9 터미널에 아래 명령을 차례로 입력한다

```
cd ~/ACM_PCA
```

```
echo $CA_ARN
```

```
echo $CA_CERTIFICATE_ID
```

```
aws acm-pca list-certificate-authorities
```

```
aws iot list-ca-certificates
```


IoT Provisioning : [6] Just-in-time 프로비저닝

```
TB="{ \\\"Parameters\\\" : { \\\"AWS::IoT::Certificate::Id\\\" : { \\\"Type\\\" : \\\"String\\\" }, \\\"AWS::IoT::Certificate::CommonName\\\" : { \\\"Type\\\" : \\\"String\\\" }, \\\"AWS::IoT::Certificate::Country\\\" : { \\\"Type\\\" : \\\"String\\\" } }, \\\"Resources\\\" : { \\\"thing\\\" : { \\\"Type\\\" : \\\"AWS::IoT::Thing\\\", \\\"Properties\\\" : { \\\"ThingName\\\" : { \\\"Ref\\\" : \\\"AWS::IoT::Certificate::CommonName\\\" }, \\\"AttributePayload\\\" : { \\\"Country\\\" : { \\\"Ref\\\" : \\\"AWS::IoT::Certificate::Country\\\" } }, \\\"ThingGroups\\\" : [ \\\"bulk-group\\\" ] } }, \\\"certificate\\\" : { \\\"Type\\\" : \\\"AWS::IoT::Certificate\\\", \\\"Properties\\\" : { \\\"CertificateId\\\" : { \\\"Ref\\\" : \\\"AWS::IoT::Certificate::Id\\\" }, \\\"Status\\\" : \\\"ACTIVE\\\" } }, \\\"policy\\\" : { \\\"Type\\\" : \\\"AWS::IoT::Policy\\\", \\\"Properties\\\" : { \\\"PolicyName\\\" : \\\"$IOT_POLICY\\\" } } } } }
```

IoT Provisioning : [6] Just-in-time 프로비저닝

```
echo $TB
```

```
aws iot update-ca-certificate --certificate-id $CA_CERTIFICATE_ID \  
    --no-remove-auto-registration \  
    --new-auto-registration-status ENABLE \  
    --registration-config "{\"templateBody\": \"$TB\", \"roleArn\":  
    \"$ARN_IOT_PROVISIONING_ROLE\"}"
```

```
aws iot describe-ca-certificate --certificate-id $CA_CERTIFICATE_ID
```

```
deviceCert=deviceJITPCert
```

IoT Provisioning : [6] Just-in-time 프로비저닝

```
openssl req -nodes -new -newkey rsa:2048 \  
-keyout $deviceCert.key \  
-out $deviceCert.csr \  
-subj "/C=DE/CN=my-jitp-device"
```

```
aws acm-pca issue-certificate \  
--certificate-authority-arn $CA_ARN \  
--csr file:///./$deviceCert.csr \  
--signing-algorithm "SHA256WITHRSA" \  
--validity Value=365,Type="DAYS" \  
--idempotency-token $(date +%Y%m%d%H%M%S')
```

IoT Provisioning : [6] Just-in-time 프로비저닝

앞 명령의 출력 값을 복사하여 넣는다

```
aws acm-pca wait certificate-issued \  
    --certificate-authority-arn $CA_ARN \  
    --certificate-arn arn:aws:acm-pca:us-east-1:844311781633:certificate-  
authority/ac2dda53-13cd-40fd-ae1b-  
95e95d200c42/certificate/30b82c555ebbe9b79337f4fb95856abe
```

앞에서 사용한 동일한 값을 복사하여 넣는다

```
aws acm-pca get-certificate \  
    --certificate-authority-arn $CA_ARN \  
    --certificate-arn arn:aws:acm-pca:us-east-1:844311781633:certificate-  
authority/ac2dda53-13cd-40fd-ae1b-  
95e95d200c42/certificate/30b82c555ebbe9b79337f4fb95856abe > $deviceCert.json
```

IoT Provisioning : [6] Just-in-time 프로비저닝

```
cat $deviceCert.json
```

```
jq -r '.Certificate' $deviceCert.json > ${deviceCert}.cert
```

```
jq -r '.Certificate' $deviceCert.json > ${deviceCert}AndCACert.cert
```

```
cat acm-pca-root-ca.pem >> ${deviceCert}AndCACert.cert
```

IoT Provisioning : [6] Just-in-time 프로비저닝

publish

```
mosquitto_pub --cafile ~/root.ca.bundle.pem \  
  --cert ${deviceCert}AndCACert.crt \  
  --key $deviceCert.key \  
  -h $IOT_ENDPOINT -p 8883 -q 1 -t ji/tp \  
  -i $deviceCert --tls-version tlsv1.2 -m '{"let-me": "in"}' -d
```

[바로 실행시 오류 메시지 출력됨(정상)]

Client deviceJITPCert sending CONNECT

Error: The connection was lost.

IoT Provisioning : [6] Just-in-time 프로비저닝

```
aws iot list-things
```

```
aws iot describe-thing --thing-name my-jitp-device
```

[MQTT 테스트 클라이언트]에서 : “ji/tp” 을 주제로 구독한다

```
# message publish
```

```
mosquitto_pub --cafile ~/root.ca.bundle.pem \  
--cert ${deviceCert}.cert \  
--key $deviceCert.key \  
-h $IOT_ENDPOINT -p 8883 -q 1 -t ji/tp -i $deviceCert \  
--tls-version tlsv1.2 -m '{"let-me": "in"}' -d
```

IoT Provisioning : [6] Just-in-time 프로비저닝

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 구독 메시지를 확인한다



IoT Provisioning : [7] Just-in-time 등록

Cloud9 터미널에 아래 명령을 차례로 입력한다

먼저 CA에서 JITP 설정을 제거 한다

```
aws iot update-ca-certificate --certificate-id $CA_CERTIFICATE_ID \  
--remove-auto-registration \  
--new-auto-registration-status DISABLE
```

```
aws iot describe-ca-certificate --certificate-id $CA_CERTIFICATE_ID
```

IoT Provisioning : [7] Just-in-time 등록

CA에 대한 Just-In-Time 등록 활성화

```
aws iot update-ca-certificate --certificate-id $CA_CERTIFICATE_ID \  
--new-auto-registration-status ENABLE
```

```
aws iot describe-ca-certificate --certificate-id $CA_CERTIFICATE_ID
```

```
cd ~/provisioning/jitr
```

IoT Provisioning : [7] Just-in-time 등록

람다 함수 생성

```
aws lambda create-function \  
    --region $REGION \  
    --function-name jitr \  
    --zip-file fileb://./jitr-lambda.zip \  
    --role $ARN_LAMBDA_ROLE \  
    --handler lambda_function.lambda_handler \  
    --runtime python3.7 \  
    --timeout 30 \  
    --memory-size 256
```

IoT Provisioning : [7] Just-in-time 등록

람다 함수 목록에서 확인

```
aws lambda list-functions
```

```
ARN_LAMBDA=$(aws lambda get-function --function-name jitr | jq -r  
'Configuration.FunctionArn')
```

```
echo $ARN_LAMBDA
```

IoT Provisioning : [7] Just-in-time 등록

IoT 주제 규칙 만들기

```
aws iot create-topic-rule --rule-name JITRRule \  
  --topic-rule-payload "{  
    \"sql\": \"SELECT * FROM '$aws/events/certificates/registered/#' WHERE  
certificateStatus = '\"PENDING_ACTIVATION\"'\",  
    \"description\": \"Rule for JITR\",  
    \"actions\": [  
      {  
        \"lambda\": {  
          \"functionArn\": \"$ARN_LAMBDA\"  
        }  
      }  
    ]  
  }"
```

IoT Provisioning : [7] Just-in-time 등록

주제 규칙 생성 확인

```
aws iot get-topic-rule --rule-name JITRRule
```

get you AWS account id

```
ACCOUNT_ID=$(aws sts get-caller-identity | jq -r '.Account')
```

verify that the variable has been set

```
echo $ACCOUNT_ID
```

store topic rule arn

```
ARN_TOPIC_RULE=$(aws iot get-topic-rule --rule-name JITRRule | jq -r  
'ruleArn')
```

IoT Provisioning : [7] Just-in-time 등록

```
# verify that the variable has been set  
echo $ARN_TOPIC_RULE
```

```
# 람다함수에 권한 정책 추가
```

```
aws lambda add-permission --function-name jitr \  
  --region $REGION --principal iot.amazonaws.com \  
  --source-arn $ARN_TOPIC_RULE --source-account $ACCOUNT_ID \  
  --statement-id Id-123 --action "lambda:InvokeFunction"
```

```
# verify the permissions of the function
```

```
aws lambda get-policy --function-name jitr
```

IoT Provisioning : [7] Just-in-time 등록

```
cd ~/ACM_PCA
```

```
deviceCert=deviceJITRCert
```

```
# key 와 CSR 생성
```

```
openssl req -nodes -new -newkey rsa:2048 \  
    -keyout $deviceCert.key \  
    -out $deviceCert.csr \  
    -subj "/C=DE/CN=my-jitr-device"
```


IoT Provisioning : [7] Just-in-time 등록

CSR에서 인증서 발급

```
aws acm-pca issue-certificate \  
    --certificate-authority-arn $CA_ARN \  
    --csr file://./$deviceCert.csr \  
    --signing-algorithm "SHA256WITHRSA" \  
    --validity Value=365,Type="DAYS" \  
    --idempotency-token $(date +%Y%m%d%H%M%S')
```

IoT Provisioning : [7] Just-in-time 등록

앞 명령의 출력 값을 복사하여 넣는다

```
aws acm-pca wait certificate-issued \  
    --certificate-authority-arn $CA_ARN \  
    --certificate-arn arn:aws:acm-pca:us-east-1:844311781633:certificate-  
authority/ac2dda53-13cd-40fd-ae1b-  
95e95d200c42/certificate/0eea455bd254a362daad11f8b7c3db80
```

앞에서 사용한 동일한 값을 복사하여 넣는다

```
aws acm-pca get-certificate \  
    --certificate-authority-arn $CA_ARN \  
    --certificate-arn arn:aws:acm-pca:us-east-1:844311781633:certificate-  
authority/ac2dda53-13cd-40fd-ae1b-  
95e95d200c42/certificate/0eea455bd254a362daad11f8b7c3db80 > $deviceCert.json
```

IoT Provisioning : [7] Just-in-time 등록

```
# publish
mosquitto_pub --cafile ~/root.ca.bundle.pem \
--cert ${deviceCert}AndCACert.crt \
--key $deviceCert.key \
-h $IOT_ENDPOINT -p 8883 -q 1 -t cmd/my-jitr-device/welcome -i my-jitr-
device \
--tls-version tlsv1.2 -m '{"let-me": "in"}' -d
```

[바로 실행 시 오류 메시지 출력됨(정상)]

Client my-jitr-device sending CONNECT

Error: The connection was lost.

IoT Provisioning : [7] Just-in-time 등록

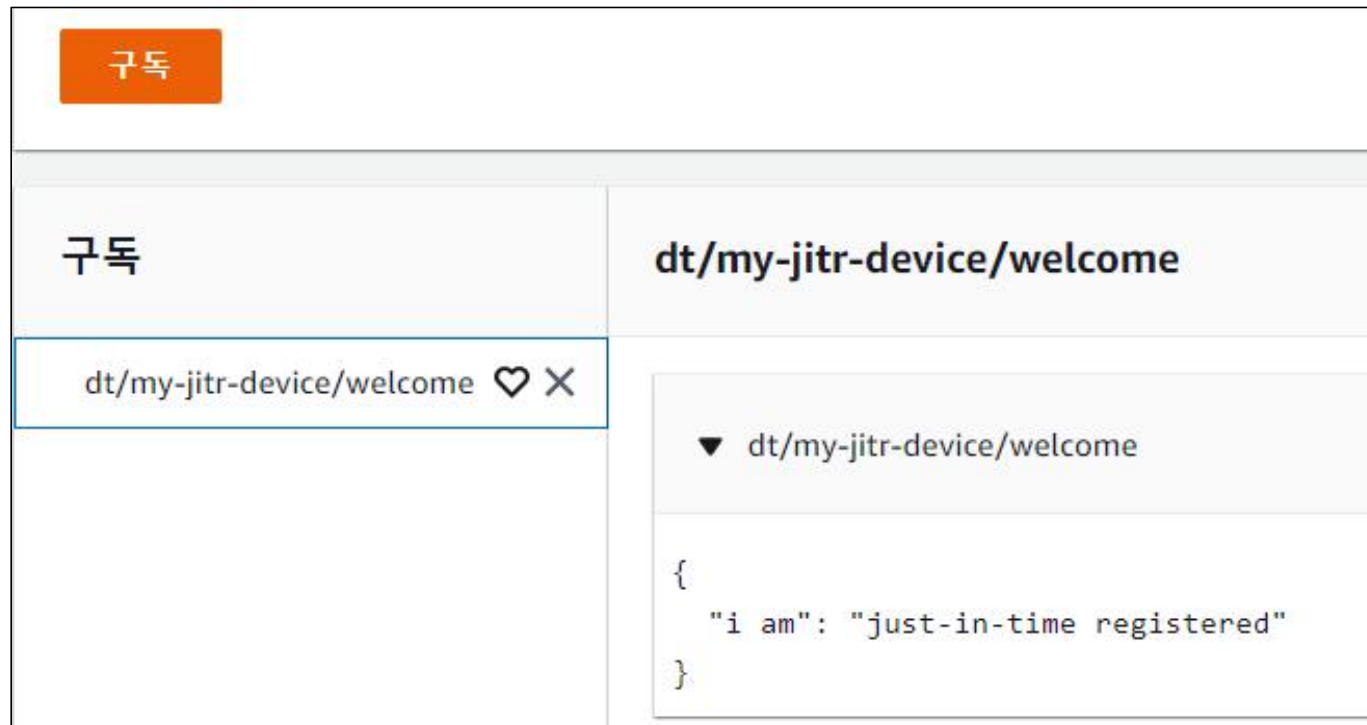
[MQTT 테스트 클라이언트]에서 : “dt/my-jitr-device/welcome ” 을 주제로 구독한다

publish

```
mosquitto_pub --cafile ~/root.ca.bundle.pem \  
  --cert ${deviceCert}.cert \  
  --key $deviceCert.key \  
  -h $IOT_ENDPOINT -p 8883 -q 1 -t dt/my-jitr-device/welcome -i my-jitr-device \  
  --tls-version tlsv1.2 -m '{"i am": "just-in-time registered"}' -d
```

IoT Provisioning : [7] Just-in-time 등록

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에서 구독 메시지를 확인한다



The End