

AWS 활용 IoT

[7] AWS IoT : Logging



강사 : 고병화

AWS IoT Logging

Logging : AWS IoT 로깅

모니터링은 AWS IoT와 사용자 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 중요한 역할을 한다

로그는 특히 오류의 원인을 찾거나 서비스에서 진행 중인 상황을 이해하는 데 항상 중요한 정보 소스이다

디바이스가 전송하는 메시지는 메시지 브로커 및 규칙 엔진을 경유하므로 AWS IoT가 각 메시지에 대한 이벤트를 전송한다.

로그 항목은 CloudWatch 콘솔의 AWSIoTLogsV2라는 로그 그룹에 나타난다

Logging : Log level(로그 수준)

로그 수준은 AWS IoT를 통해 작성할 로그 메시지의 유형을 말한다

오류(Error)(최소 세부 정보) : 오류 로그 항목만 기록

경고(Warning) : 오류 및 경고 로그 항목을 기록

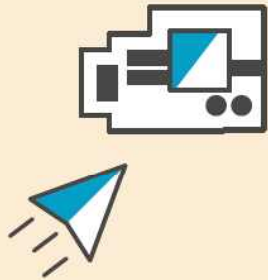
정보(Informational) : 정보, 경고 및 오류 로그 항목을 기록

디버그(Debug)(최대 세부 정보) : 모든 로그 항목을 기록

로깅 비활성화(Disable logging) : 로그 항목이 기록되지 않는다

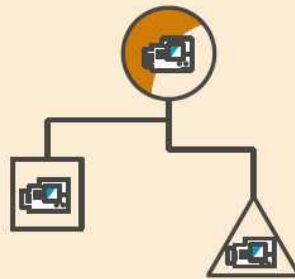


Monitoring Device Events



Jobs

Monitoring of
Device
Updates



Groups

Monitor Devices
Joining Groups



Policies

Monitor Device Security
Policies



Ressource-specific Logging

Groups /
logging-debug-group

Details	Resource-specific logging
Groups	You can choose to override the log level for this resource.
Things	
Security	Local logging setting Set to DEBUG ENABLED
Jobs	
Logs	Global logging setting Set to WARN ENABLED

```
{  
  "timestamp": "2018-04-17 13:50:21.616",  
  "logLevel": "INFO",  
  "traceId": "6753a942-92c3-f979-587c-  
9c634874b672",  
  "accountId": "123456789012",  
  "status": "Success",  
  "eventType": "Publish-In",  
  "protocol": "MQTT",  
  "topicName": "$aws/things/job-agent/jobs/get",  
  "clientId": "job-agent",  
  "principalId":  
  "9187849467e75a1a92cbcf0f3a6a49b4f10d820b  
99dfa62657cf4b6e60c0dac4",  
  "sourceIp": "35.178.51.181",  
  "sourcePort": 46435  
}
```

Logging : AWS IoT Core 로깅 설정

AWS IoT Core 콘솔의 [설정]에서 [로그]의 [로그 관리] 버튼을 클릭한다

로그 정보

AWS IoT 로깅을 관리하여 유용한 정보를 CloudWatch Logs에 기록할 수 있습니다.

로그 관리

디바이스의 메시지가 메시지 브로커 및 규칙 엔진을 통과하면 AWS IoT 로그가 문제 해결에 도움이 될 수 있는 이벤트를 처리합니다.

역할
iotddb_role

로그 수준
오류(최소 상세 수준)

Logging : AWS IoT Core 로깅 설정

역할 선택을

“DeviceManagement
Workshop-
IoTWSIoTServiceRole
-XXXXXXXXXX” 으로
선택하고 로그 수준
을 “**정보**”로 설정하
고 [업데이트] 버튼
을 누른다

로그 역할 정보

CloudWatch Logs에 정보를 기록하는 데 사용할 역할을 생성하거나 선택합니다.

역할 선택

DeviceManagementWorkshop-MiscResour-IoTServiceRole-1G0TRURVZM30Y ▼

역할 생성

☒ IAM 역할에 정책을 연결하여 AWS IoT가 사용자를 대신해 CloudWatch에 로그를 게시하도록 권한을 허용합니다.

로그 수준 정보

로그에 대해 원하는 상세 수준을 선택합니다. 오류(최소 상세 정보 표시)를 선택하면 오류만 기록되며 최소한의 정보만 표시됩니다. 디버그(최대 상세 정보 표시)를 선택하면 가장 세부적인 로그가 생성됩니다. 더 세부적인 로그를 수집하면 로깅 비용이 증가할 수 있습니다.

로그 수준

정보 ▼

취소

업데이트

Logging : AWS IoT 이벤트 활성화

이벤트 구성 상태 가져오기(Cloud 9터미널에서 실행)

```
aws iot describe-event-configurations
```

모든 이벤트를 활성화 시킨다

```
aws iot update-event-configurations --cli-input-json \
```

```
{  
  "eventConfigurations": {  
    "THING_TYPE": {  
      "Enabled": true  
    },  
    "JOB_EXECUTION": {  
      "Enabled": true  
    },  
    "THING_GROUP_HIERARCHY": {  
      "Enabled": true  
    }  
  }  
}
```

Logging : AWS IoT 이벤트 활성화

```
},  
  "CERTIFICATE": {  
    "Enabled": true  
  },  
  "THING_TYPE_ASSOCIATION": {  
    "Enabled": true  
  },  
  "THING_GROUP_MEMBERSHIP": {  
    "Enabled": true  
  },  
  "CA_CERTIFICATE": {  
    "Enabled": true  
  },  
  "THING": {  
    "Enabled": true  
  }
```

Logging : AWS IoT 이벤트 활성화

```
{  
  "JOB": {  
    "Enabled": true  
  },  
  "POLICY": {  
    "Enabled": true  
  },  
  "THING_GROUP": {  
    "Enabled": true  
  }  
}
```

Logging : Fine-grained[세밀한] logging

세밀한 로깅을 사용하면 특정 사물 그룹에 대한 로깅 수준을 설정할 수 있다

세밀한 로그는 로그 그룹 "AWSIoTLogsV2"의 아래 Amazon CloudWatch에 저장된다.

앞에서 사용된 사물 그룹 "building-one"을 사용한다

Cloud9 터미널에서 다음 명령 부터 차례로 수행한다

```
THING_GROUP_NAME=building-one
```

Logging : Fine-grained[세밀한] logging

세밀한 로깅을 위한 현재 로깅 구성을 확인한다

`aws iot get-v2-logging-options`

```
aiore0427:~/provisioning $ aws iot get-v2-logging-options
{
  "roleArn": "arn:aws:iam::844311781633:role/workshop/role/DeviceManagementWorkshop-MiscResour-IoTServiceRole-1G0TRURVZM30Y",
  "defaultLogLevel": "INFO",
  "disableAllLogs": false
}
```

사물 그룹에 대한 로깅 수준을 **DEBUG**로 변경한다

`aws iot set-v2-logging-level --log-level DEBUG \`
`--log-target "{\targetType\": \"THING_GROUP\",`
`\targetName\": \"$THING_GROUP_NAME\"}"`

Logging: Fine-grained[세밀한] logging

logging level을 다시 확인한다
aws iot list-v2-logging-levels

```
aicore0427:~/provisioning $ aws iot list-v2-logging-levels
{
  "logTargetConfigurations": [
    {
      "logTarget": {
        "targetType": "DEFAULT"
      },
      "logLevel": "INFO"
    },
    {
      "logTarget": {
        "targetType": "THING_GROUP",
        "targetName": "building-one"
      },
      "logLevel": "DEBUG"
    }
  ]
}
```

Logging : Fine-grained[세밀한] logging

이전 실습에서 했던 것처럼 게시가 허용된 주제와 게시 권한이 없는 주제에 메시지를 게시한다

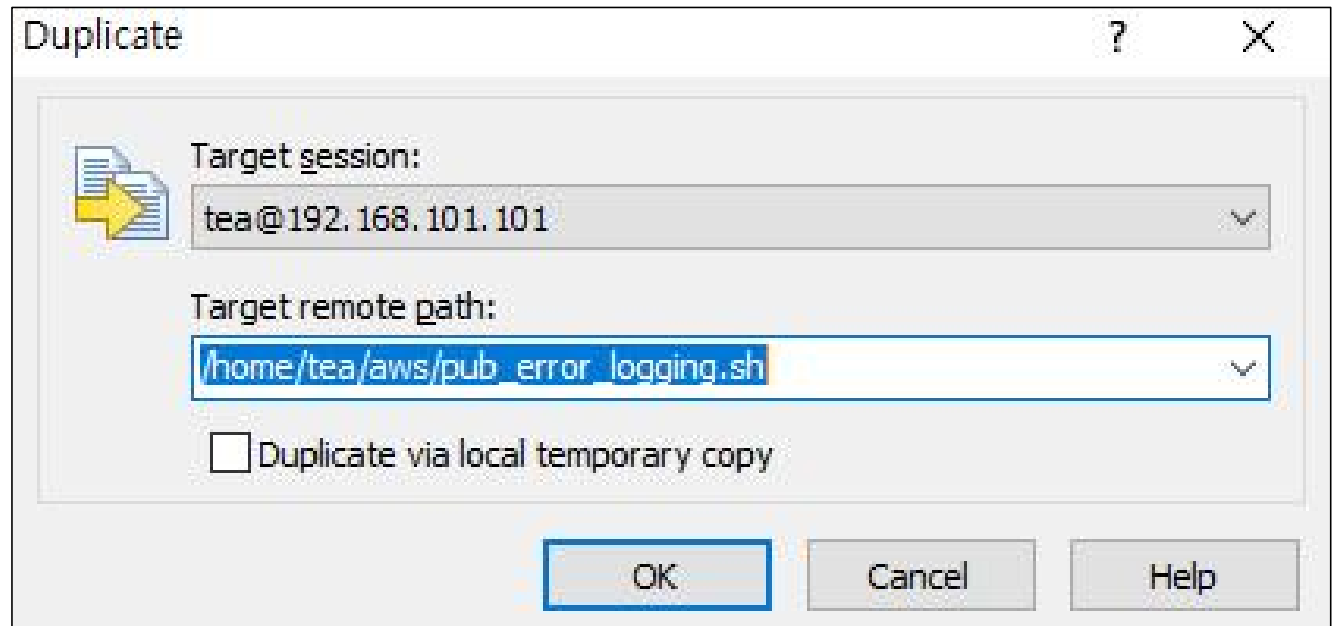
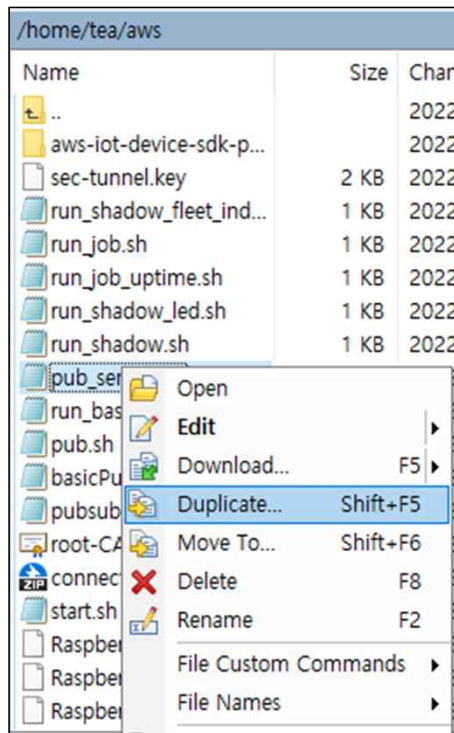
로그가 CloudWatch로 전송될 때까지 약간의 시간이 걸릴 수 있으므로 몇 분 정도 기다린다.

지난 1시간 동안의 이벤트에 대한 로그에서 검색을 수행한다

다양한 로그 수준 또는 클라이언트 ID를 기반으로 검색을 해본다

Logging : Pi3 보드 error logging하기 실습

WinSCP에서 Pi3보드에 연결 후 aws경로안의 pub_sensor.sh파일을 선택하고 우클릭하여 Duplicate를 선택하고 파일명을 pub_error_logging.sh로 변경하여 저장한다



Logging : Pi3 보드 error logging하기 실습

WinSCP에서 pub_error_logging.sh 파일을 클릭하여 편집기에서 뒷부분의 topic을 **iot/sensor**에서 **iot/test_log**로 수정하고 저장한다 (ERROR를 로깅하기 위하여 일부러 정책에 등록되어 있지 않은 권한이 없는 topic을 사용하여 실습한다)

```
python3 aws-iot-device-sdk-python-v2/samples/pub_sensor.py --  
endpoint a1mp2lfc29w0b5-ats.iot.us-east-1.amazonaws.com --  
ca_file root-CA.crt --cert RaspberryPi.cert.pem --key  
RaspberryPi.private.key --client_id basicPubSub --topic iot/test_log  
--count 0
```

Logging : Pi3 보드 error logging하기 실습

AWS 콘솔의 [CloudWatch]로 가서 [로그 그룹]에서
“AWSIoTLogsV2”를 클릭한다 새로 로깅 되는 내용을 확인해 보기
위해서 이전에 생성된 로그 스트림 목록을 전체 선택하고 삭제
한다

로그 스트림 (8/8)	
<input type="text" value="로그 스트림 필터링 또는 접두사 검색 시도"/>	<input type="checkbox"/> 정확히 일치
<input checked="" type="checkbox"/>	로그 스트림
<input checked="" type="checkbox"/>	0e985de0-8102-45f2-bbc5-69572b45bad9_844311781633_0
<input checked="" type="checkbox"/>	e44e0ed3-dc21-4189-aaaa-aec955deec0_844311781633_0
<input checked="" type="checkbox"/>	5ce3c142-8db5-4954-9228-87a678486a6b_844311781633_0
<input checked="" type="checkbox"/>	32b4d7cc-8fdd-4cae-b71d-7d6ee6a72793_844311781633_0
<input checked="" type="checkbox"/>	8d133f3f-ce6c-4b51-8545-7ca0f8522f93_844311781633_0
<input checked="" type="checkbox"/>	52854ae9-59dc-4ffd-b077-b9eaade1f10c_844311781633_0
<input checked="" type="checkbox"/>	af5a1457-e445-4253-87f1-7d2e2c0da4c7_844311781633_0
<input checked="" type="checkbox"/>	e70e557d-26de-4801-bbf5-3204399a8430_844311781633_0

Logging : Pi3 보드 error logging하기 실습

Pi3보드의 터미널로 가서 aws경로로 이동하여

pub_error_logging.sh을 실행 시킨다

아래와 같이 error 메시지가 보이면 CTRL-C로 중지 시킨다

```
192.168.101.101 (planx) - VNC Viewer
./pub_error_logging.sh
tea@planx ~$ cd aws
tea@planx ~/aws$ ./pub_error_logging.sh
Connecting to a1mp2lfc29w0b5-ats.iot.us-east-1.amazonaws.com with client ID 'basicPubSub'...
Connected!
Subscribing to topic 'iot/test_log'...
Connection interrupted. error: AWS_ERROR_MQTT_UNEXPECTED_HANGUP: The connection was closed unexpectedly.
Connection resumed. return_code: 0 session_present: True
Connection interrupted. error: AWS_ERROR_MQTT_UNEXPECTED_HANGUP: The connection was closed unexpectedly.
Connection resumed. return_code: 0 session_present: True
Connection interrupted. error: AWS_ERROR_MQTT_UNEXPECTED_HANGUP: The connection was closed unexpectedly.
Connection resumed. return_code: 0 session_present: True
Connection interrupted. error: AWS_ERROR_MQTT_UNEXPECTED_HANGUP: The connection was closed unexpectedly.
```

Logging: Pi3 보드 error logging하기 실습

AWS 콘솔의 [CloudWatch]로 가서 브라우저의 다시 고침 아이콘을 한번 누르면 새로 로깅 된 목록이 보일 것이다(시간을 확인)
2번째 혹은 3번째를 클릭해보면 우측처럼 ERROR 메시지를 볼 수 있다

<input type="checkbox"/>	로그 스트림
<input type="checkbox"/>	c3ab1dc0-78ba-44d0-883f-87f41163c597_844311781633_0
<input type="checkbox"/>	6653915b-543c-4d4a-9a3e-fae89fc2bcca_844311781633_0
<input type="checkbox"/>	66c09ad0-fd33-4109-acc4-db1aac23f8d7_844311781633_0
<input type="checkbox"/>	4ca45d23-1258-456c-bc76-c2fbefbd355c_844311781633_0
<input type="checkbox"/>	cc773e63-de48-4143-a37d-3ee0df39694a_844311781633_0
<input type="checkbox"/>	dbc6503d-7c61-43e4-9671-4fa743906898_844311781633_0

```
2022-09-19T13:21:19.244+09:00 {"timestamp": "2022-09-19 04:21:19.244",  
{  
  "timestamp": "2022-09-19 04:21:19.244",  
  "logLevel": "ERROR",  
  "traceId": "f6c277b5-39b9-36f8-7136-92e22b0effba",  
  "accountId": "844311781633",  
  "status": "Failure",  
  "eventType": "Subscribe",  
  "protocol": "MQTT",  
  "topicName": "iot/test_log",  
  "clientId": "basicPubSub",  
  "principalId": "20758280db74ba7012ad509cde47043e24e0da",  
  "sourceIp": "123.213.208.180",  
  "sourcePort": 35610,  
  "reason": "AUTHORIZATION_FAILURE",  
  "details": "Authorization Failure"  
}
```

Logging : Fine-grained[세밀한] logging

Cloud9의 터미널에서 아래와 같이 로깅 메시지를 검색할 수 있다

시작시간을 1시간 이전의 시간을 밀리 세컨드로 계산한다

```
starttime=$((($(date '+%s') - 3600) * 1000))
```

```
echo $starttime
```

1시간 이전의 모든 로그를 검색한다

```
aws logs filter-log-events --log-group-name AWSIoTLogsV2 \  
--start-time $starttime
```

Logging : Fine-grained[세밀한] logging

log level 이 **INFO**인 것만 검색한다

```
aws logs filter-log-events --log-group-name AWSSlotLogsV2 \  
--start-time $starttime --filter-pattern "{$.logLevel = INFO}"
```

log level 이 **ERROR**인 것만 검색한다

```
aws logs filter-log-events --log-group-name AWSSlotLogsV2 \  
--start-time $starttime --filter-pattern "{$.logLevel = ERROR}"
```


Logging : Fine-grained[세밀한] logging

log level 이 **ERROR**인 로그만 볼 수가 있다

```
aicore0427:~ $ aws logs filter-log-events --log-group-name AWSIoTLogsV2 \
> --start-time $starttime --filter-pattern "${$.logLevel = ERROR}"

{
  "events": [
    {
      "logStreamName": "c9034025-875d-4dcc-adff-5bacf55f6133_844311781633_0",
      "timestamp": 1663562233033,
      "message": "{\"timestamp\":\"2022-09-19 04:37:13.033\", \"logLevel\":\"ERROR\", \"traceId\":",
      "\", \"status\":\"Failure\", \"eventType\":\"Subscribe\", \"protocol\":\"MQTT\", \"topicName\":\"iot/test_I",
      "2ad509cde47043e24e0daaa1d84aec9b8ad2ac5e03ad766\", \"sourceIp\":\"123.213.208.180\", \"sourcePort\":5678",
      "ilure\"}",
      "ingestionTime": 1663562240807,
      "eventId": "37098677480767556345777268448024255657580310479112634369"
    },
    {
      "logStreamName": "5884b8d5-21e6-45fd-9dcc-3528c4e8fe73_844311781633_0",
      "timestamp": 1663562234392,
      "message": "{\"timestamp\":\"2022-09-19 04:37:14.392\", \"logLevel\":\"ERROR\", \"traceId\":",
      "\", \"status\":\"Failure\", \"eventType\":\"Subscribe\", \"protocol\":\"MQTT\", \"topicName\":\"iot/test_I",
      "2ad509cde47043e24e0daaa1d84aec9b8ad2ac5e03ad766\", \"sourceIp\":\"123.213.208.180\", \"sourcePort\":5556",
      "ilure\"}",
      "ingestionTime": 1663562242929,
      "eventId": "37098677511074269070580385299936475725171692935599226881"
    },
    {
      "logStreamName": "b1112fac-2eec-43fa-9279-7dd2168e3021_844311781633_0",
      "timestamp": 1663562239391,
      "message": "{\"timestamp\":\"2022-09-19 04:37:19.391\", \"logLevel\":\"ERROR\", \"traceId\":",
      "\", \"status\":\"Failure\", \"eventType\":\"Subscribe\", \"protocol\":\"MQTT\", \"topicName\":\"iot/test_I",
      "2ad509cde47043e24e0daaa1d84aec9b8ad2ac5e03ad766\", \"sourceIp\":\"123.213.208.180\", \"sourcePort\":3913",
      "ilure\"}"
    }
  ]
}
```

Logging : Fine-grained[세밀한] logging

Client ID를 필터로 검색하면 해당 디바이스만 볼 수 있다

CLIENT_NAME=basicPubSub

aws logs filter-log-events --log-group-name AWSIoTLogsV2 \
--start-time \$starttime --filter-pattern "{\$.clientId =
\$CLIENT_NAME}"

```
alicore0427:~ $ aws logs filter-log-events --log-group-name AWSIoTLogsV2 \  
> --start-time $starttime --filter-pattern "{$.clientId = $CLIENT_NAME}"  
{  
  "events": [  
    {  
      "logStreamName": "c9034025-875d-4dcc-adff-5bacf55f6133_844311781633_0",  
      "timestamp": 1663562232839,  
      "message": "{$\"timestamp\": \"2022-09-19 04:37:12.839\", \"logLevel\": \"INFO\", \"traceId\": \"7af08b58-0e33-0f7f-345c-329c2c6b\", \"status\": \"Success\", \"eventType\": \"Connect\", \"protocol\": \"MQTT\", \"clientId\": \"basicPubSub\", \"principalId\": \"20758280db74ba7d2ac5e03ad766\", \"sourceIp\": \"123.213.208.180\", \"sourcePort\": 56782}\",  
      "ingestionTime": 1663562240807,  
      "eventId": "37098677476441211777262327558566326312686528346952433664"  
    },  
    {  
      "logStreamName": "c9034025-875d-4dcc-adff-5bacf55f6133_844311781633_0",  
      "timestamp": 1663562233033,  
      "message": "{$\"timestamp\": \"2022-09-19 04:37:13.033\", \"logLevel\": \"ERROR\", \"traceId\": \"74899f2b-2070-4f99-e12a-72ffe46\", \"status\": \"Failure\", \"eventType\": \"Subscribe\", \"protocol\": \"MQTT\", \"topicName\": \"iot/test_log\", \"clientId\": \"basicPubSub\", \"principalId\": \"2ad509cde47043e24e0daa1d84aec9b8ad2ac5e03ad766\", \"sourceIp\": \"123.213.208.180\", \"sourcePort\": 56782, \"reason\": \"AUTHORIZATION_FAILURE\"}\",  
      "ingestionTime": 1663562240807,  
      "eventId": "37098677476441211777262327558566326312686528346952433664"  
    }  
  ]  
}
```


The End