

IoT 프로젝트 특별과정

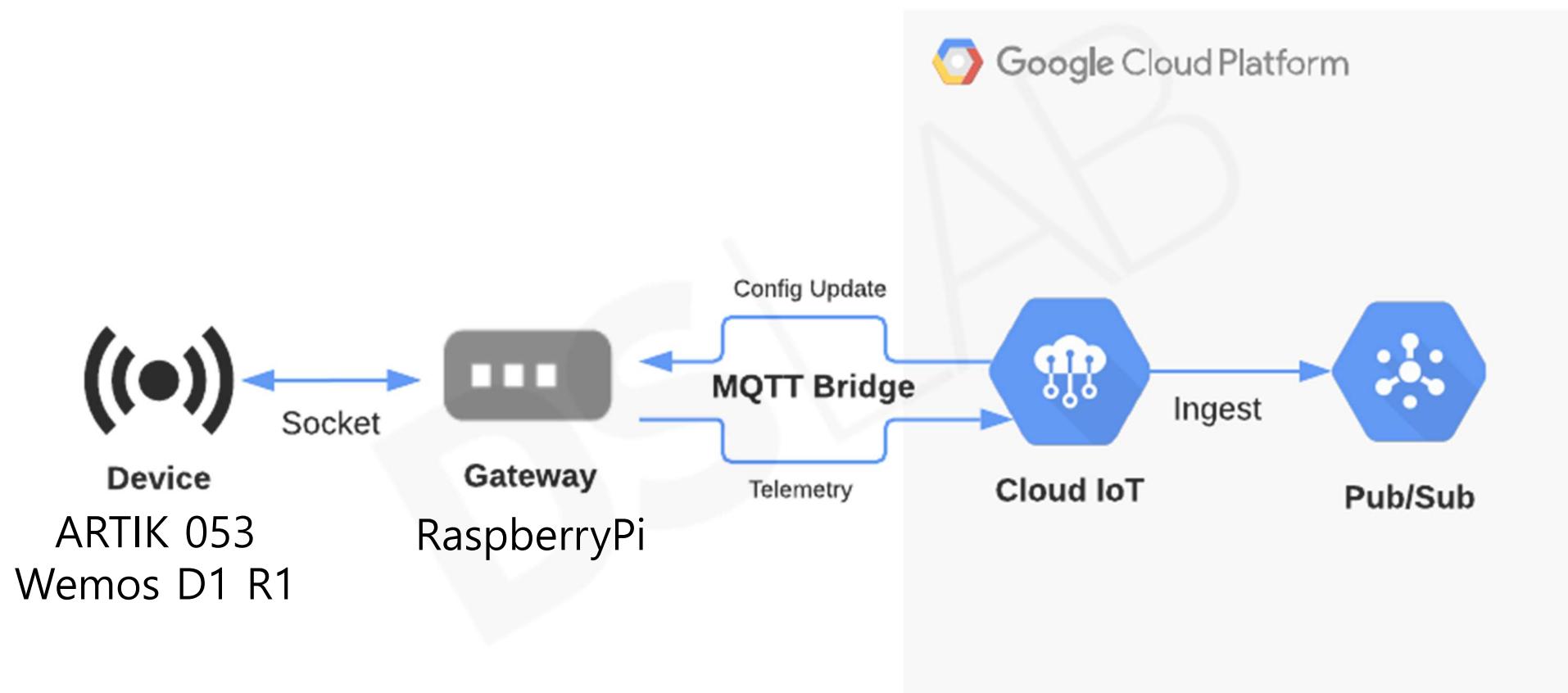
– Part 5. Google IOT Core –

Contents

- ❖ RaspberryPi Gateway 구축하기
- ❖ Device 만들기 - ARTIK053
- ❖ Device 만들기 - Arduino(WeMos D1 R1)
- ❖ OAuth 2.0 Access Token 받기



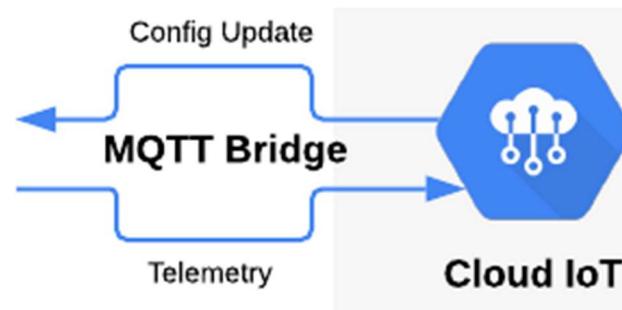
Google IOT Core 전체적 흐름도



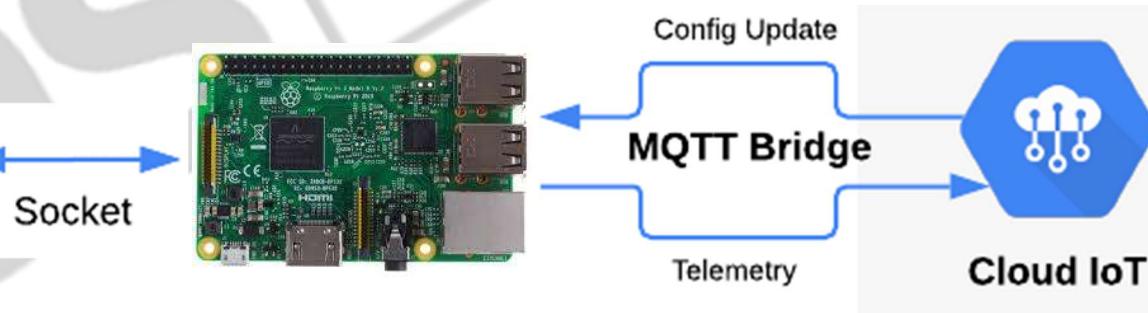
RaspberryPi Gateway 구축하기

RaspberryPi Gateway 구축하기

❖ Gateway를 구축하는 이유



- 각각의 디바이스 키값을 발급 및 관리
- 멀티프로세싱으로 인한 성능 저하
- C언어 기반



- 하나의 키값으로 디바이스 여러 개를 묶어서 관리
- 소켓통신만을 사용하여 성능 저하를 최소화
- Python, node.js 등을 사용

RaspberryPi Gateway 구축하기

- ❖ 프로젝트 생성 - <https://console.cloud.google.com> 접속

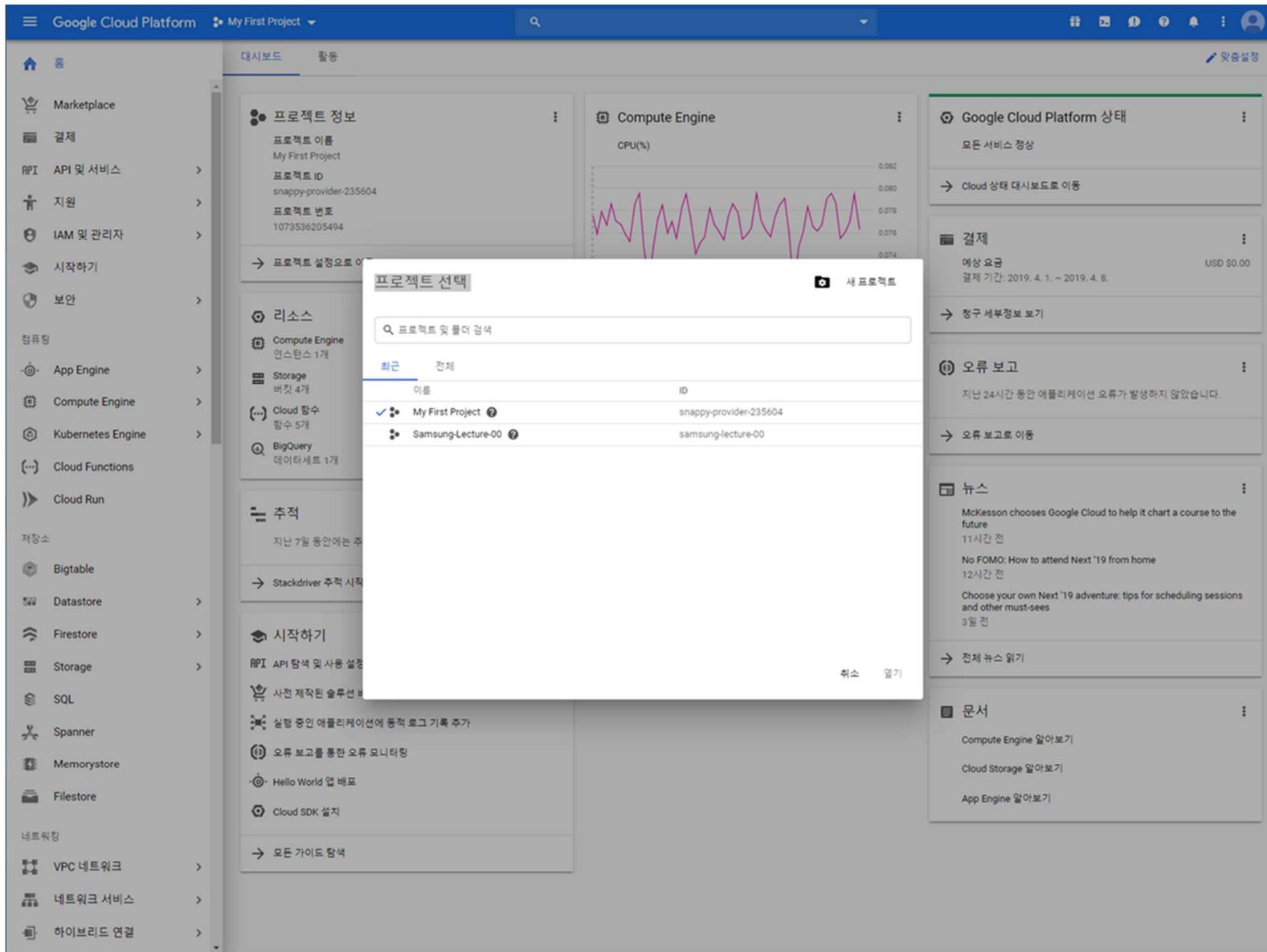
The screenshot shows the GCP dashboard with the following sections:

- 프로젝트 정보**: Shows the project name (Samsung-Lecture-00), ID (samsung-lecture-00), and number (70115256646). It also includes a link to "프로젝트 설정으로 이동".
- API API**: Displays API usage metrics over time, showing requests (요청) and responses (응답) for various APIs.
- Google Cloud Platform 상태**: Monitors service status and provides links to the Cloud Status Dashboard and regional billing information.
- 리소스**: A section stating "이 프로젝트에는 리소스가 없습니다." (No resources available).
- 결제**: Shows estimated charges (예상 요금) for the period from April 1 to April 7, 2019, amounting to USD \$0.00.
- 추적**: Provides access to Stackdriver Trace data.
- 오류 보고**: Offers error reporting and monitoring options.
- 뉴스**: Features news articles related to GCP, such as "Choose your own Next '19 adventure: tips for scheduling sessions and other must-sees" and "Want repeatable scale? Adopt infrastructure as code on GCP".
- 문서**: Links to documentation for Compute Engine, Cloud Storage, and App Engine.
- 시작하기**: A sidebar containing links to various quick-start guides, including "Hello World" and "Cloud Storage 버킷 생성".

The left sidebar lists various GCP services: Marketplace, 결제, API 및 서비스, 지원, IAM 및 관리자, 시작하기, 보안, 컴퓨팅 (App Engine, Compute Engine, Kubernetes Engine, Cloud Functions, Cloud Run), 저장소 (Bigtable, Datastore, Firestore, Storage, SQL, Spanner, Memystore, Filestore), 네트워킹 (VPC 네트워크, 네트워크 서비스, 하이브리드 연결).

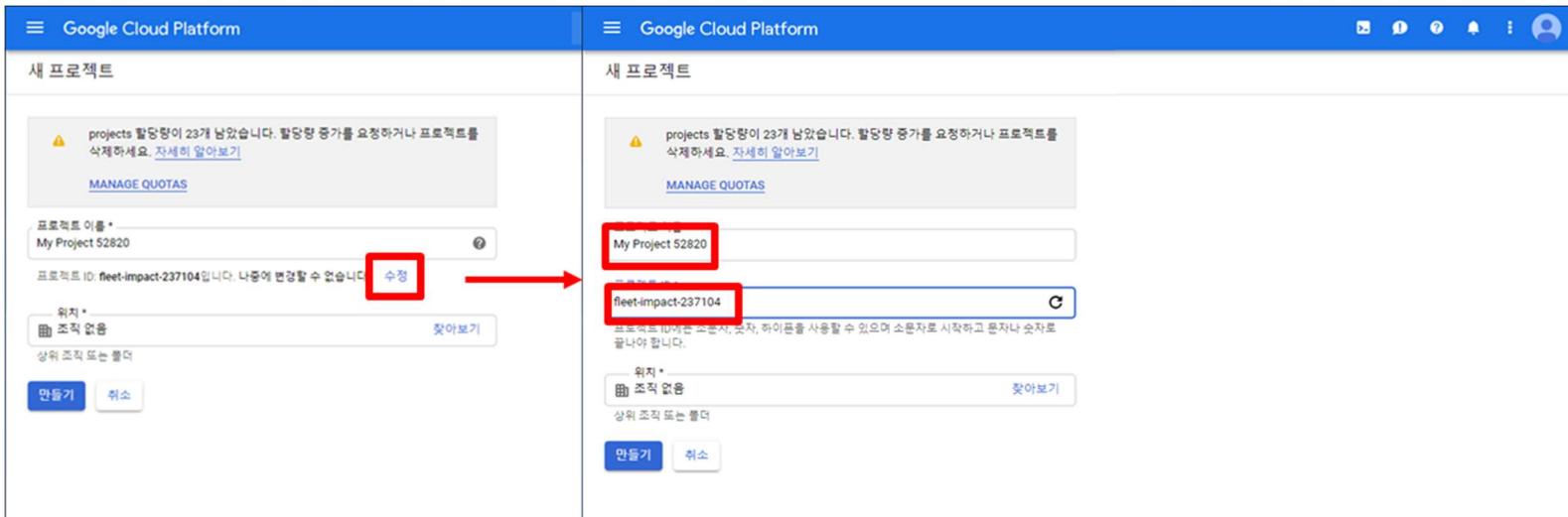
RaspberryPi Gateway 구축하기

❖ 프로젝트 생성 - 프로젝트 선택 클릭 및 새 프로젝트



RaspberryPi Gateway 구축하기

❖ 프로젝트 생성 - 프로젝트 이름 및 프로젝트 ID 설정



RaspberryPi Gateway 구축하기

❖ 프로젝트 생성 - 프로젝트 선택

The screenshot shows the Google Cloud Platform (GCP) dashboard for the project "Samsung-Lecture-01". The left sidebar lists various services like Marketplace, Compute Engine, and Storage. The main area displays the "Project Selection" dialog, which lists recent projects. The project "Samsung-Lecture-01" is selected and highlighted with a red box. Other listed projects include "My First Project" and "Samsung-Lecture-00". The right side of the screen shows the "Google Cloud Platform Status" section, which indicates all services are running normally.

이름	ID
Samsung-Lecture-01	samsung-lecture-01
My First Project	snappy-provider-235604
Samsung-Lecture-00	samsung-lecture-00

RaspberryPi Gateway 구축하기

❖ 레지스트리 생성 - IoT 코어 클릭

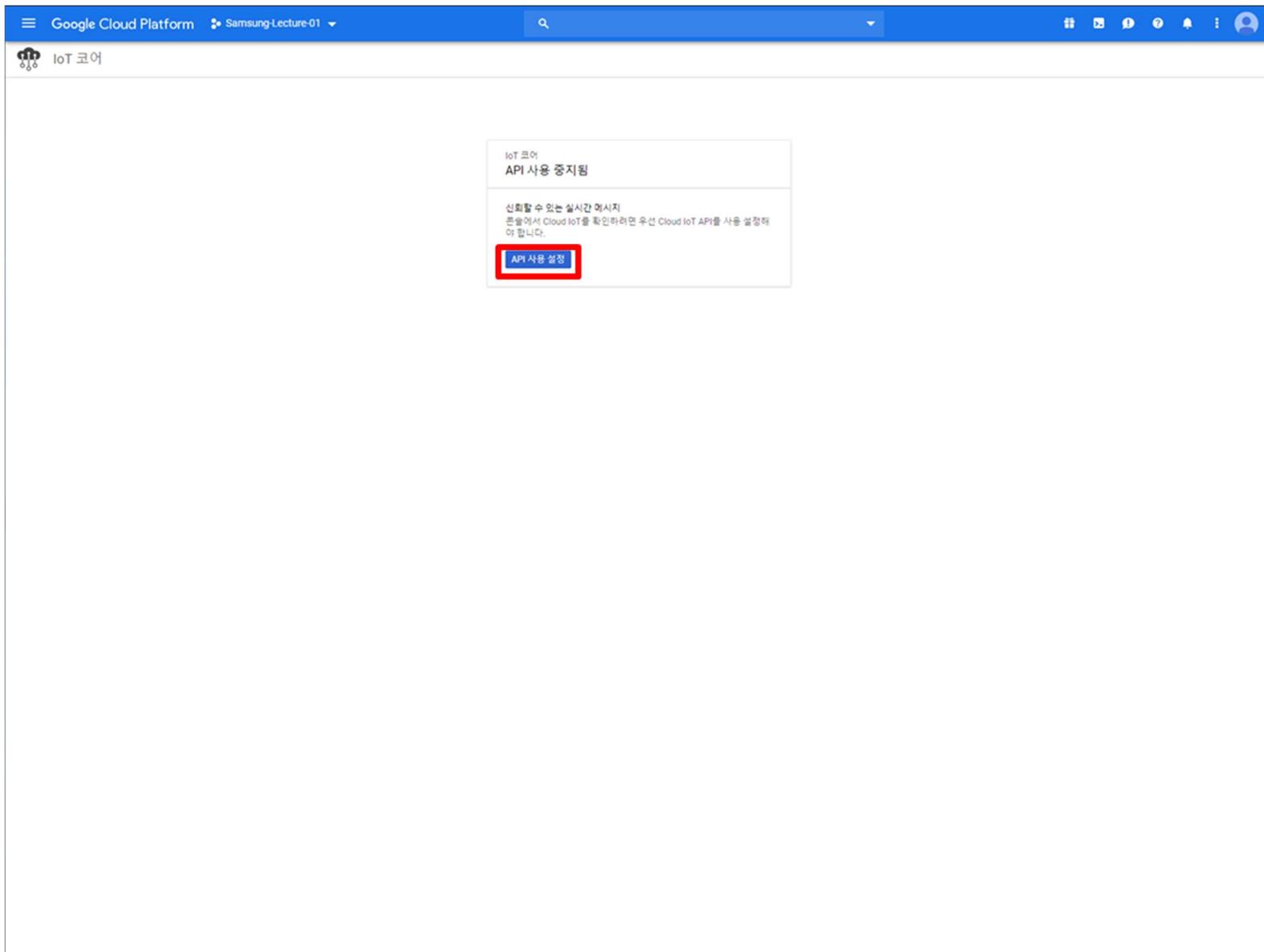
The screenshot shows the Google Cloud Platform (GCP) dashboard for a project named 'My First Project'. The left sidebar navigation bar has 'IoT 코어' (IoT Core) highlighted with a red box. The main content area displays several cards:

- 프로젝트 정보**: Shows the project name 'My First Project', ID 'snappy-provider-235604', and number of instances (1).
- Compute Engine**: A chart showing CPU utilization over time, with a value of 0.075 Instance/cpu/utilization.
- Google Cloud Platform 상태**: Shows all services are running.
- 결제**: Shows estimated charges for the period April 1, 2019 - April 8, 2019.
- 오류 보고**: No errors reported.
- 뉴스**: News items from Google Cloud.
- API API**: A chart showing API requests over time, with a value of 3,283 요청(requests).
- 주석**: No annotations.
- 시작하기**: Step-by-step guides for RPI API analysis and usage settings, Stackdriver monitoring setup, logs, metrics, and log export.

At the bottom of the page, there is a URL: <https://console.cloud.google.com/iot/authuser=18&hl=ko&organizationId=0&project=snappy-provider-235604>.

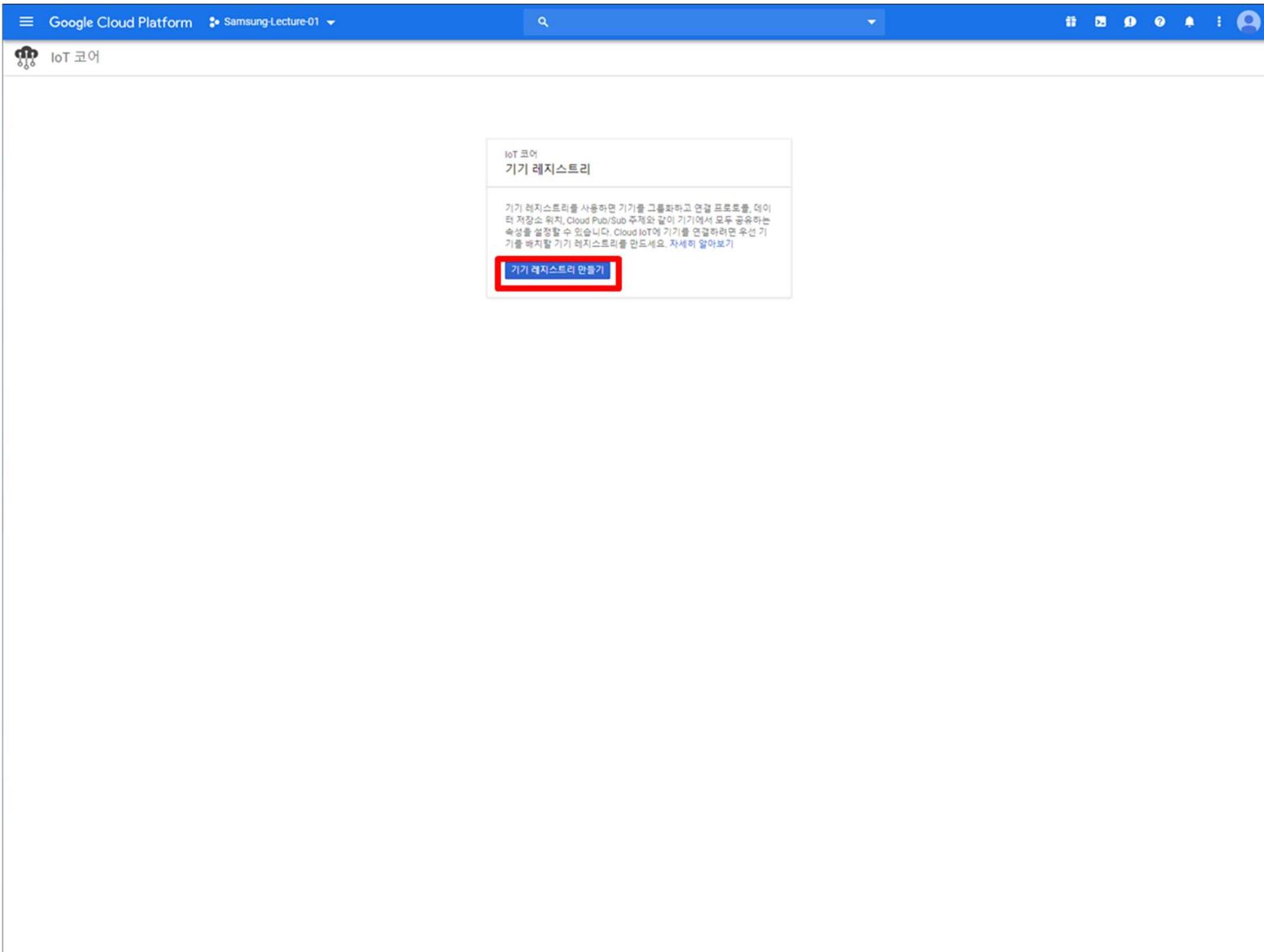
RaspberryPi Gateway 구축하기

❖ 레지스트리 생성 - API 사용 설정



RaspberryPi Gateway 구축하기

- ❖ 레지스트리 생성 - 레지스트리 만들기 클릭



RaspberryPi Gateway 구축하기

❖ 레지스트리 생성 - ID 및 Region, Topic 설정

The screenshot shows the 'Registry 만들기' (Create Registry) page in the Google Cloud Platform. The 'Registry ID' field is highlighted with a red box and contains the value 'EX)Samsung.Lecture.01'. The 'Region' dropdown is also highlighted with a red box and contains the value 'EX)asia-east1'. The 'Cloud Pub/Sub 주제' dropdown is highlighted with a red box and contains the value 'EX)projects/Samsung-lecture-01/topics/pubsub01'. Other fields like 'Protocols' (MQTT, HTTP), 'Cloud Pub/Sub 주제 추가' (Cloud Pub/Sub topic selection), and 'Stackdriver Logging' (logging levels: 연습, 오류, 정보, 디버그) are visible but not highlighted.

RaspberryPi Gateway 구축하기

❖ 라즈베리파이 세팅 - 게이트웨이 코드 다운로드

```
pi@raspberrypi: ~/RaspberryPi-Gateway
pi@raspberrypi: ~$ login as: pi
pi@raspberrypi: ~$ pi@192.168.0.21's password:
Linux raspberrypi 4.14.98-v7+ #1200 SMP Tue Feb 12 20
The programs included with the Debian GNU/Linux system,
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Mar 25 16:16:30 2019
pi@raspberrypi: ~$ git clone https://github.com/dslab-lws/RaspberryPi-Gateway
'RaspberryPi-Gateway'에 복제합니다 ...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 4), reused 9 (delta 3), pack-reused 0
오브젝트 묶음 푸는 중 : 100% (10/10), 완료 .
```

git clone https://github.com/dslab-lws/RaspberryPi-Gateway

RaspberryPi Gateway 구축하기

❖ 라즈베리파이 세팅 - 디렉토리 변경

```
pi@raspberrypi: ~/RaspberryPi-Gateway
pi@raspberrypi: ~$ login as: pi
pi@raspberrypi: ~$ pi@192.168.0.21's password:
Linux raspberrypi 4.14.98-v7+ #1200 SMP Tue Feb 12 20:27:48 GMT 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Mar 25 16:16:30 2019
pi@raspberrypi: ~$ 
pi@raspberrypi: ~$ 
pi@raspberrypi: ~$ 
pi@raspberrypi: ~$ 
pi@raspberrypi: ~$ 
pi@raspberrypi: ~$ git clone https://github.com/dslab-lws/RaspberryPi-Gateway
'RaspberryPi-Gateway'에 복제합니다 ...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 4), reused 9 (delta 3), pack-reused 0
오브젝트 풀어 푸는 중: 100% (10/10), 완료.
pi@raspberrypi: ~$ cd RaspberryPi-Gateway/
pi@raspberrypi: ~/RaspberryPi-Gateway $ 
```

cd RaspberryPi-Gateway

RaspberryPi Gateway 구축하기

❖ 라즈베리파이 세팅 - 키 생성

```
pi@raspberrypi:~/RasberryPi-Gateway $ chmod +x generate_keys.sh
pi@raspberrypi:~/RasberryPi-Gateway $ ./generate_keys.sh
Generating RSA private key, 2048 bit long modulus
.....+++++
.....+.....
e is 65537 (0x010001)
writing RSA key
```

실행 권한 주기 -> chmod +x generate_keys.sh
실행 -> ./generate_keys.sh

RaspberryPi Gateway 구축하기

❖ 라즈베리파이 세팅 - 생성된 키 파일 열기 및 키 복사

```
pi@raspberrypi: ~/RasberryPi-Gateway
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAOCAQ8AMIIBCgKCAQEA8rRZJTMG+YUfIpFmdS4H
UMqvazDRlWqoYw47tMhvU/efNsVyWXbsWt1Toi0f5aGhaP7Oh0mqvfgvYgs23s1C
AkXhQT7Zk6wxWYXKWETcNMciTExXvc9FqhZJ+M0DsqbebdgzTbyBpO0tDWzKRG0E
TE1E7v8TwNevK94oECT7gPF9/ildBp3swiOBRZ5GnuYxnMm6y0Flr6DZaD3DJ1po
QoxfiZbz/hgdaChiNsczKNOFLPLjv/RSr/tlujhWI+uo+c9omInCfBI7XGqe01nA
E6ImwWIg2djNDS7WIu6gFA7Ps+CEeSXIfdSYX9j+ooxpPMfhqnxnMJqw0nK7IWse
iWIDAQAB
-----END PUBLIC KEY-----
```

vi rsa_public.pem
복사 후 :q + Enter

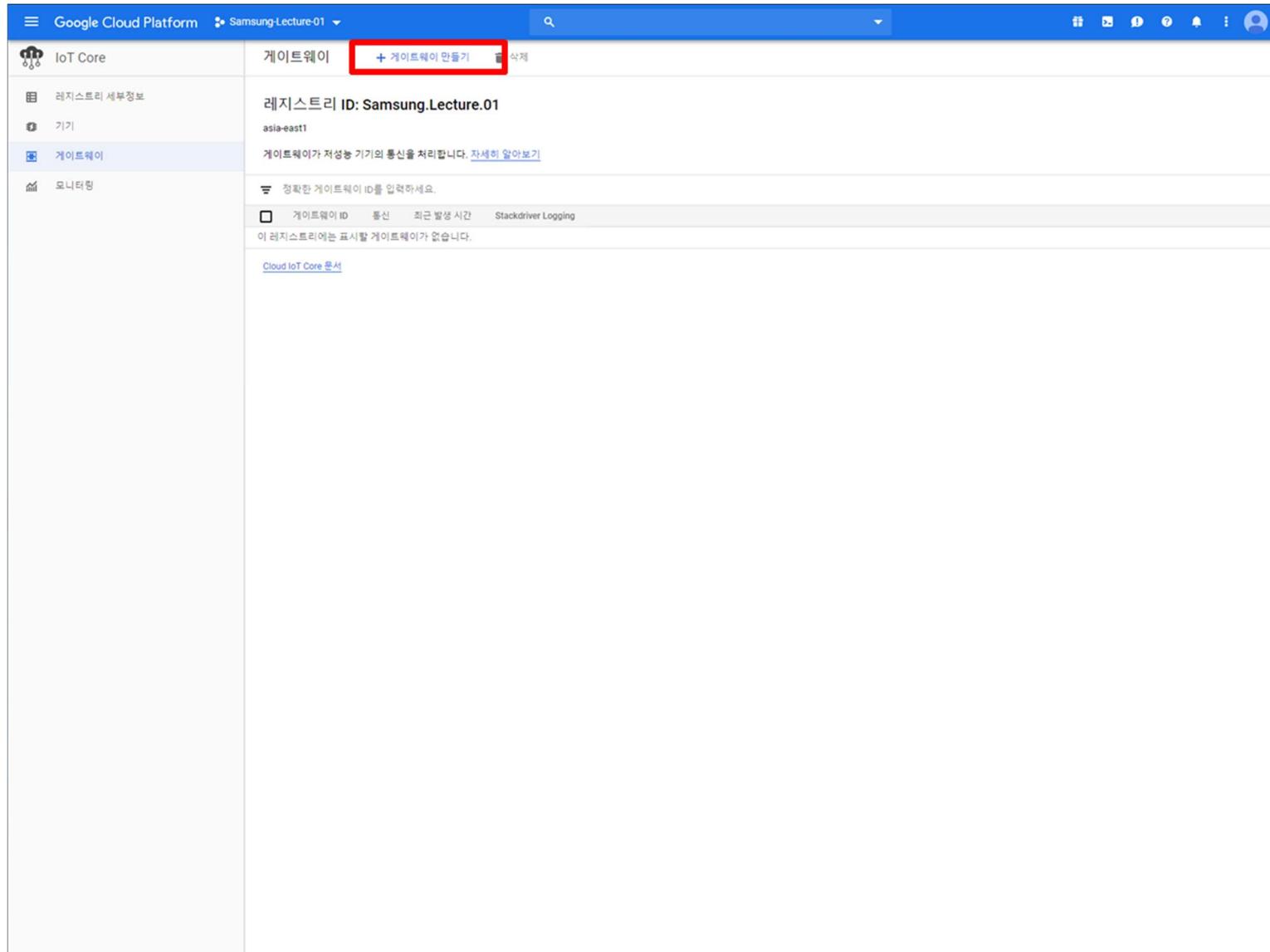
RaspberryPi Gateway 구축하기

❖ 게이트웨이 생성 - 게이트웨이 클릭

The screenshot shows the Google Cloud Platform IoT Core interface for the project "Samsung-Lecture-01". On the left sidebar, under the "기기" (Devices) section, the "게이트웨이" (Gateway) option is highlighted with a red box. The main pane displays the configuration for a device registry with the ID "Samsung.Lecture.01". It includes fields for location ("리전" - asia-east1), connection protocols ("프로토콜" - MQTT, HTTP), and Stackdriver Logging. Below this, the "Cloud Pub/Sub 주제" (Cloud Pub/Sub Topic) section lists two topics: "projects/samsung-lecture-01/topics/pubsub01" (Topic type: 기본 원격 분석, Handler: -) and another unnamed topic (Topic type: 기본 상태, Handler: -). A collapsed section for "CA CERTIFICATES" is also visible.

RaspberryPi Gateway 구축하기

❖ 게이트웨이 생성 - 게이트웨이 만들기 클릭



RaspberryPi Gateway 구축하기

❖ 게이트웨이 생성 - Gateway ID 및 키값 설정

Google Cloud Platform Samsung-Lecture-01 게이트웨이 만들기

Samsung Lecture.01 레지스트리에서 게이트웨이를 만듭니다.

Gateway ID: ex)gateway01

게이트웨이 허신: 적용

인증: 선택사항

인증 방식: 직접 입력

공개 키 험식: RS256

공개 키 값

```
-----BEGIN PUBLIC KEY-----  
(Public key value must be in PEM format)  
-----END PUBLIC KEY-----
```

공개 키 인증서 (선택사항)

연결만으로 설정(키 값만 같으면 연결 가능하게 설정)

연결만으로 설정(키 값만 같으면 연결 가능하게 설정)

Stackdriver Logging

게이트웨이의 로그 설정을 선택하세요. 이 기기에 함께 레지스트리 기본값을 재정의합니다.

기본 설정 보기

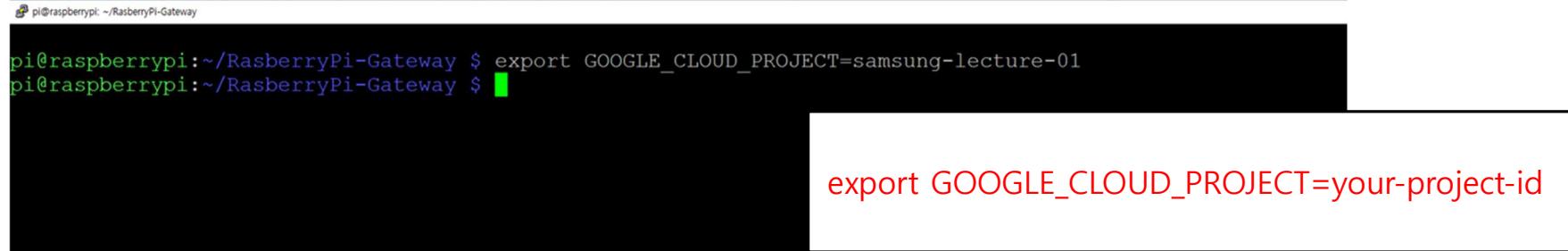
레지스트리 기본 설정 사용

작성하기

취소

RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - Project ID 전역변수 설정



```
pi@raspberrypi: ~/RaspberryPi-Gateway $ export GOOGLE_CLOUD_PROJECT=samsung-lecture-01  
pi@raspberrypi: ~/RaspberryPi-Gateway $ █
```

export GOOGLE_CLOUD_PROJECT=your-project-id

RaspberryPi Gateway 구축하기

- ❖ 게이트웨이 접속 - run-gateway 파일 열기 및 수정

vi run-gateway

```
pi@raspberrypi: ~/RaspberryPi-Gateway
#!/bin/bash

# Copyright 2018 Google LLC
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

python ./cloudiot_mqtt_gateway.py \
    --registry_id Samsung.Lecture.01 \
    --gateway_id gateway01 \
    --cloud_region=asia-east1 \
    --private_key_file=rsa_private.pem \
    --algorithm=RS256 \
    --ca_certs=roots.pem \
    --mqtt_bridge_hostname= mqtt.googleapis.com \
    --mqtt_bridge_port=8883 \
    --jwt_expires_minutes=1200

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~

"run-gateway" 26 lines, 909 characters
```

Registry ID

Gateway ID

RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - 게이트웨이에 필요한 모듈 다운로드

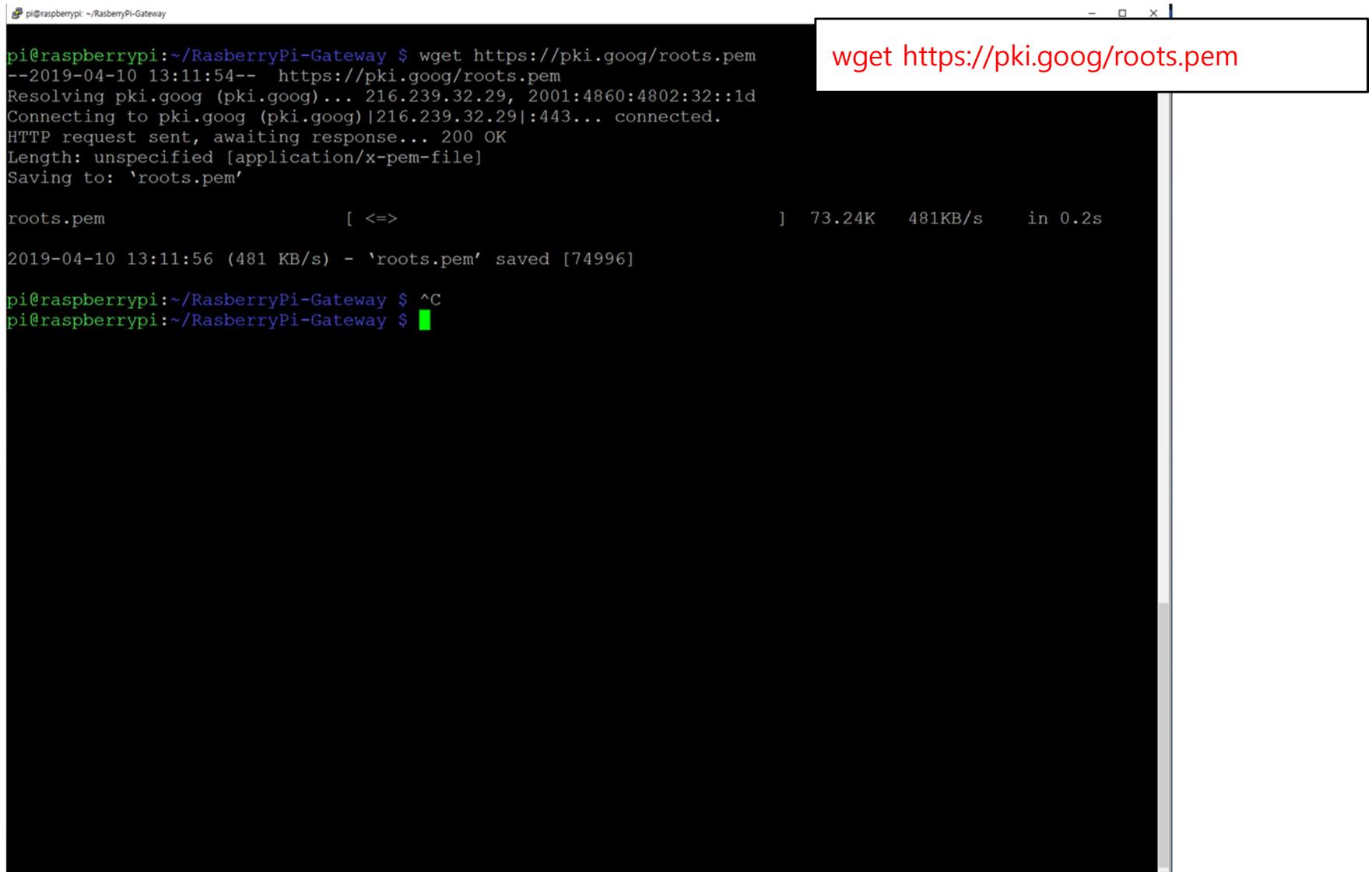


```
pi@raspberrypi:~/RasberryPi-Gateway $ pip install -r requirements-gateway.txt
Collecting cryptography==2.4.1 (from -r requirements-gateway.txt (line 1))
  Collecting paho-mqtt==1.4.0 (from -r requirements-gateway.txt (line 2))
    Collecting pyjwt==1.6.4 (from -r requirements-gateway.txt (line 3))
      Using cached https://files.pythonhosted.org/packages/93/d1/3378cc8184a6524dc92993090ee8b4c03847c567e298305d6cf86987e005/PyJWT-1.6.4-py2.py3-none-any.whl
      Collecting httplib2==0.12.1 (from -r requirements-gateway.txt (line 4))
        Collecting pathlib==1.0.1 (from -r requirements-gateway.txt (line 5))
          Collecting enum34; python_version < "3" (from cryptography==2.4.1->-r requirements-gateway.txt (line 1))
            Using cached https://files.pythonhosted.org/packages/c5/db/e56e6b4bbac7c4a06de1c50de6felef3810018ae11732a50f15f62c7d050/enum34-1.1.6-py2-none-any.whl
            Collecting asn1crypto>=0.21.0 (from cryptography==2.4.1->-r requirements-gateway.txt (line 1))
              Using cached https://files.pythonhosted.org/packages/ea/cd/35485615f45f30a510576f1a56d1e0a7ad7bd8ab5ed7cdc600ef7cd06222/asn1crypto-0.24.0-py2.py3-none-any.whl
            Collecting idna>=2.1 (from cryptography==2.4.1->-r requirements-gateway.txt (line 1))
              Using cached https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl
            Collecting cffi!=1.11.3,>=1.7 (from cryptography==2.4.1->-r requirements-gateway.txt (line 1))
            Collecting six>=1.4.1 (from cryptography==2.4.1->-r requirements-gateway.txt (line 1))
              Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe898238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
            Collecting ipaddress; python_version < "3" (from cryptography==2.4.1->-r requirements-gateway.txt (line 1))
          )
            Using cached https://files.pythonhosted.org/packages/fc/d0/7fc3a811e011d4b388be48a0e381db8d990042df54aa4ef4599a31d39853/ipaddress-1.0.22-py2.py3-none-any.whl
            Collecting pycparser (from cffi!=1.11.3,>=1.7->cryptography==2.4.1->-r requirements-gateway.txt (line 1))
              Using cached https://www.piwheels.org/simple/pycparser/pycparser-2.19-py2.py3-none-any.whl
            Installing collected packages: enum34, asn1crypto, idna, pycparser, cffi, six, ipaddress, cryptography, paho-mqtt, pyjwt, httplib2, pathlib
            Successfully installed asn1crypto-0.24.0 cffi-1.12.2 cryptography-2.4.1 enum34-1.1.6 httplib2-0.12.1 idna-2.8 ipaddress-1.0.22 paho-mqtt-1.4.0 pathlib-1.0.1 pycparser-2.19 pyjwt-1.6.4 six-1.12.0
세 그 멘 테 이 선 오 류
pi@raspberrypi:~/RasberryPi-Gateway $
```

pip install -r requirements-gateway.txt

RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - Google CA 루트인증서 받아오기



```
pi@raspberrypi:~/RaspberryPi-Gateway $ wget https://pki.goog/roots.pem
--2019-04-10 13:11:54-- https://pki.goog/roots.pem
Resolving pki.goog (pki.goog)... 216.239.32.29, 2001:4860:4802:32::1d
Connecting to pki.goog (pki.goog)|216.239.32.29|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-pem-file]
Saving to: 'roots.pem'

roots.pem [ <=> ] 73.24K 481KB/s in 0.2s

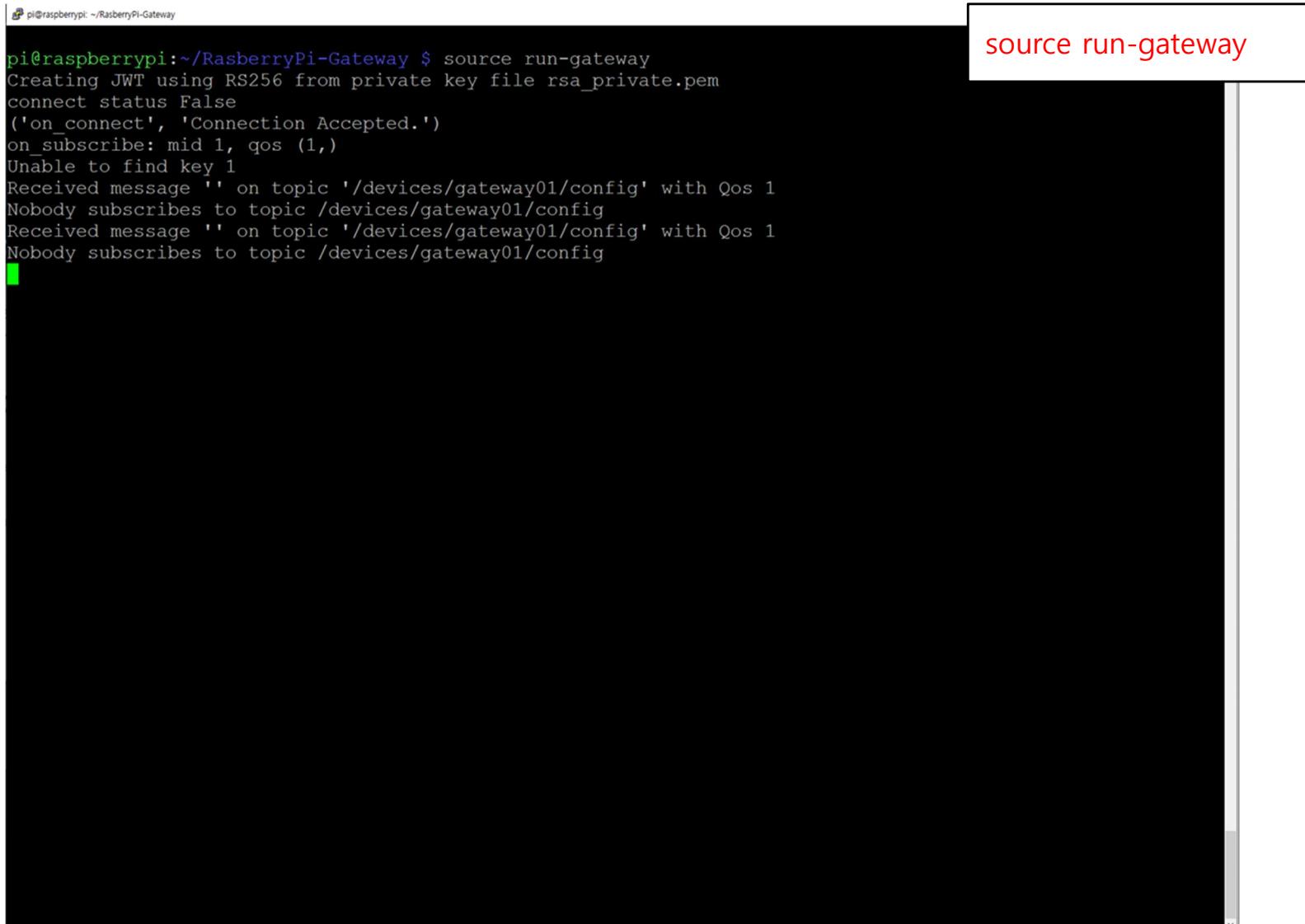
2019-04-10 13:11:56 (481 KB/s) - 'roots.pem' saved [74996]

pi@raspberrypi:~/RaspberryPi-Gateway $ ^C
pi@raspberrypi:~/RaspberryPi-Gateway $
```

The terminal window shows the command `wget https://pki.goog/roots.pem` being run. The output indicates the file is being downloaded from Google's PKI server. A red box highlights the command in the input field.

RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - 게이트웨이 실행



A terminal window showing the output of a command. The command is highlighted in red at the top right of the window. The terminal shows the process of generating a JWT using RS256 from a private key file, connecting to a broker, and receiving configuration messages for a gateway device.

```
pi@raspberrypi:~/RaspberryPi-Gateway $ source run-gateway
Creating JWT using RS256 from private key file rsa_private.pem
connect status False
('on_connect', 'Connection Accepted.')
on_subscribe: mid 1, qos (1,)
Unable to find key 1
Received message '' on topic '/devices/gateway01/config' with Qos 1
Nobody subscribes to topic /devices/gateway01/config
Received message '' on topic '/devices/gateway01/config' with Qos 1
Nobody subscribes to topic /devices/gateway01/config
```

source run-gateway

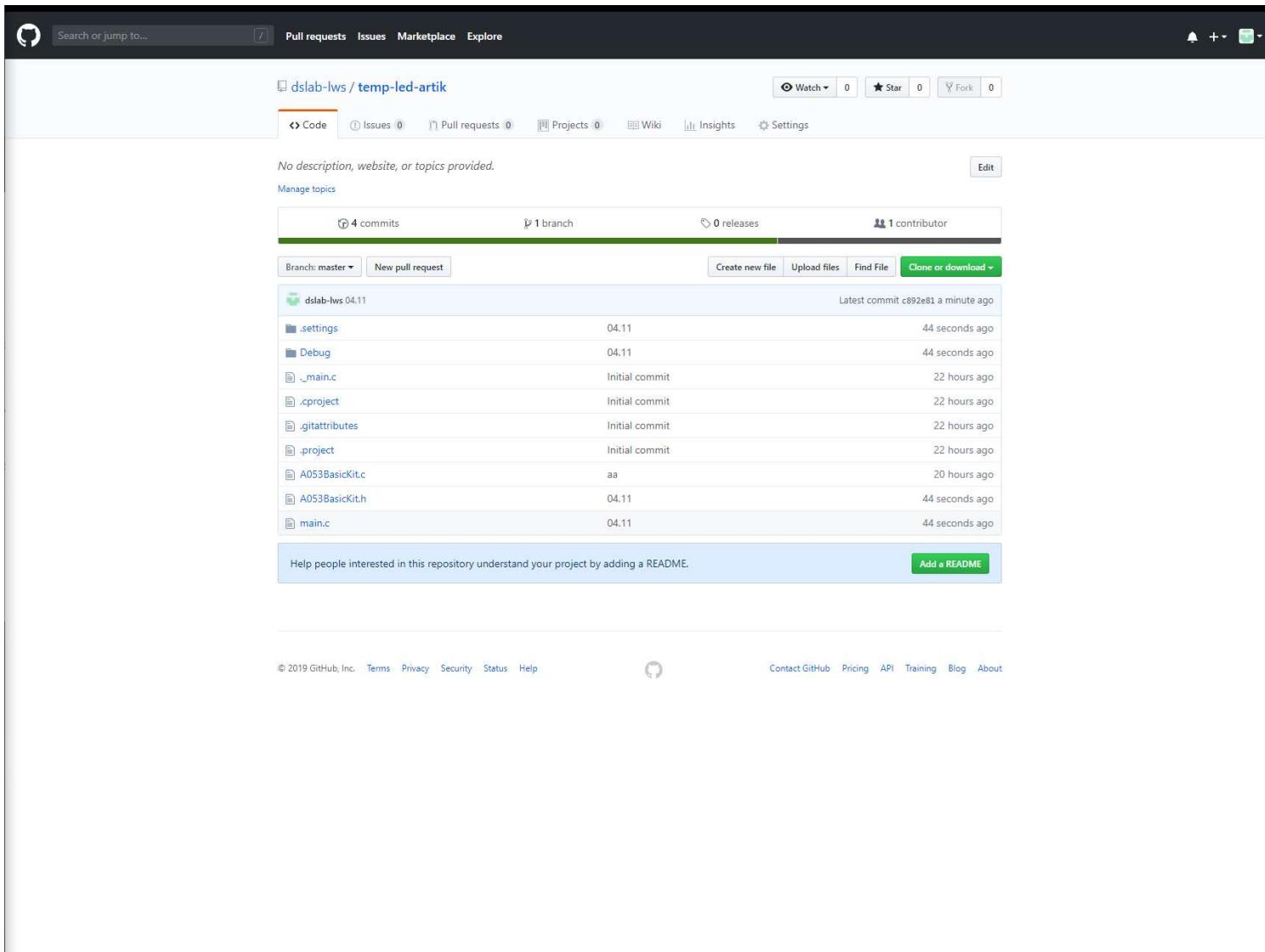
디바이스 만들기 -ARTIK 053



디바이스 만들기 -ARTIK 053

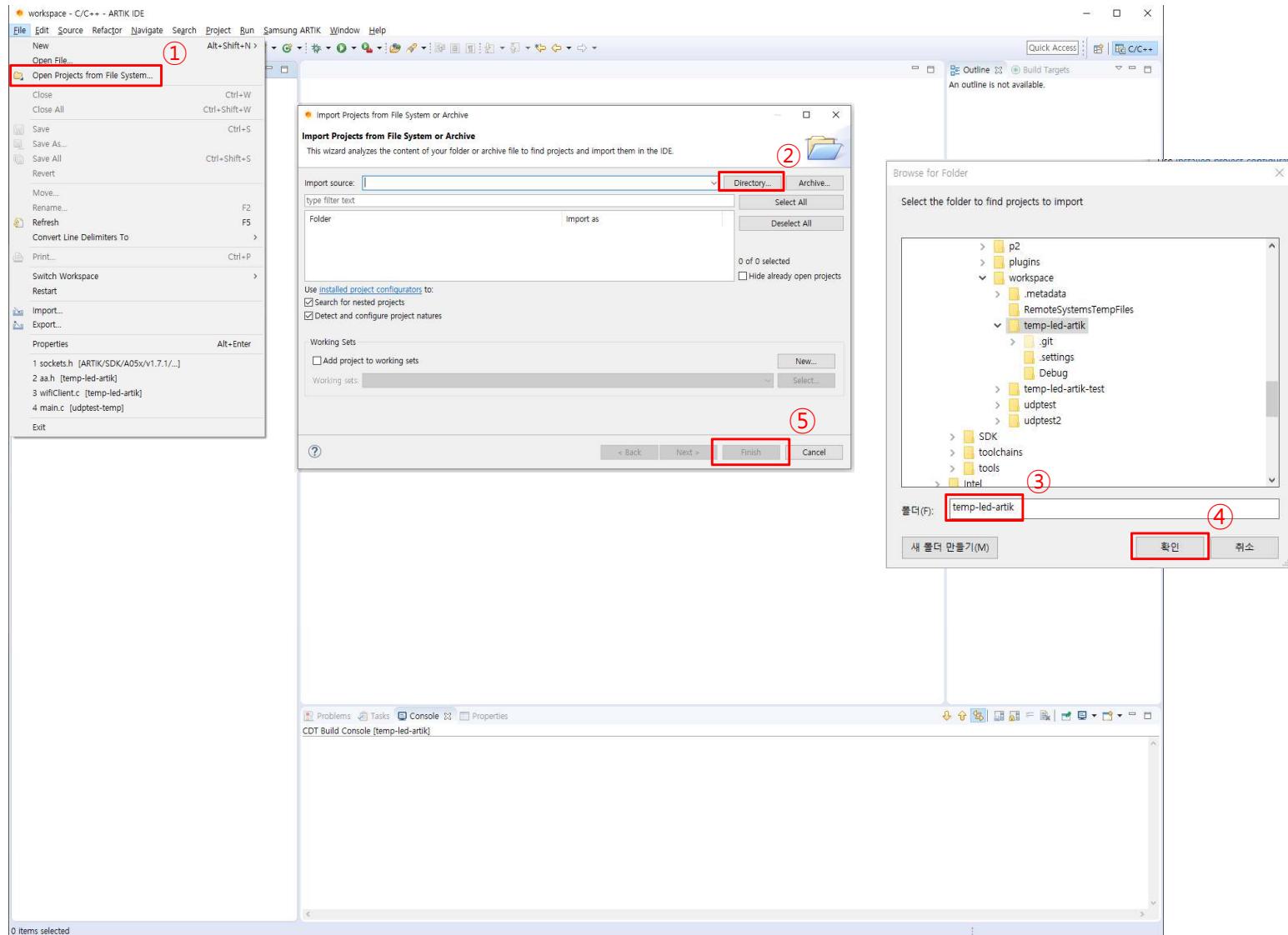
❖ ARTIK 053 세팅 - 아틱 소스코드 다운로드

<https://github.com/dslab-lws/temp-led-artik>



디바이스 만들기 -ARTIK 053

❖ ARTIK 053 세팅 - 프로젝트 불러오기



디바이스 만들기 -ARTIK 053

❖ ARTIK 053 세팅 - main.c 수정

```
#include "A053BasicKit.h"

#define BACKLOG 10
#define BUFSIZE 256
#define MAX_DATA_SIZE 256
#define PORT 10000
#define IP_addr "000.000.000.000" //Gateway IP Address
#define device_id "temp-led-artik" //Device ID

int sockfd;
int ret,len;
struct sockaddr_in servAddr;
char sendBuffer[BUFSIZE], recvBuffer[BUFSIZE];
int recvLen, servLen;
char message[100] = {0,};
```

Gateway IP 입력 (모르면 라즈베리파이에서 ifconfig)

디바이스 만들기 -ARTIK 053

- ❖ ARTIK 053 세팅 - A053BasicKit.h 수정

```
// WIFI
#define STATE_DISCONNECTED      0
#define STATE_CONNECTED         1
#define SLSI_WIFI_SECURITY_OPEN "open"
#define SLSI_WIFI_SECURITY_WPA2_AES "wpa2_aes"
#define NET_DEVNAME "wl1"
#define SSID "WIFI SSID" //FIX
#define PSK "WIFI PASSWORD" //FIX
```

와이파이 이름과 비밀번호 입력

디바이스 만들기 -ARTIK 053

❖ ARTIK 053 세팅 - 프로젝트 빌드 및 에러확인

① 프로젝트 빌드

```

#include "A053BasicKit.h"

#define BACKLOG 10
#define BUFSIZE 256
#define MAX_DATA_SIZE 256
#define PORT 10000
#define IP_addr "192.168.0.48" //Gateway IP Address
#define device_id "temp-led-artik" //Device ID

int sockfd;
int ret,len;
struct sockaddr_in servAddr;
char sendBuffer[BUFSIZE], recvBuffer[BUFSIZE];
int recvlen, servlen;
char message[100] = {0,};

void RecvMessage(int flag){
    if((recvLen=recvfrom(sockfd, recvBuffer, BUFSIZE-1, Flag, (struct sockaddr*)&servAddr,(socklen_t*)&servAddr)) > 0) {
        recvBuffer[recvlen] = '\0';
        printf("RecvMsgId: %s\n", recvBuffer);
        if (strcmp(recvBuffer,"LEDON") == 0 ){
            gpio_write(47,1);
            printf("*****LED Is On*****\n");
        }
        if (strcmp(recvBuffer,"LEDOFF") == 0 ){
            gpio_write(47,0);
            printf("*****LED Is Off*****\n");
        }
    }
}

void MakeMessage(char *action, char *data){
    int i;
    for(i = 0 ; i < 100 ; i++){
        message[i] = 0;
    }

    int len_id = strlen(device_id);
    int len_ac = strlen(action);
    int len_da = strlen(data);

    char msg_device[] = "\\"device\\" : \";
    char msg_action[] = "\\", \"action\":=\"";
    char msg_data[] = "\", \"data\":\"";
    char msg_finish[] = "\")";

    strcpy(message,msg_device);
    strcat(message,device_id);
    strcat(message,msg_action);
    strcat(message,data);
    strcat(message,msg_finish);

    if(len_da != 1)
        message[len_da] = '\0';
}

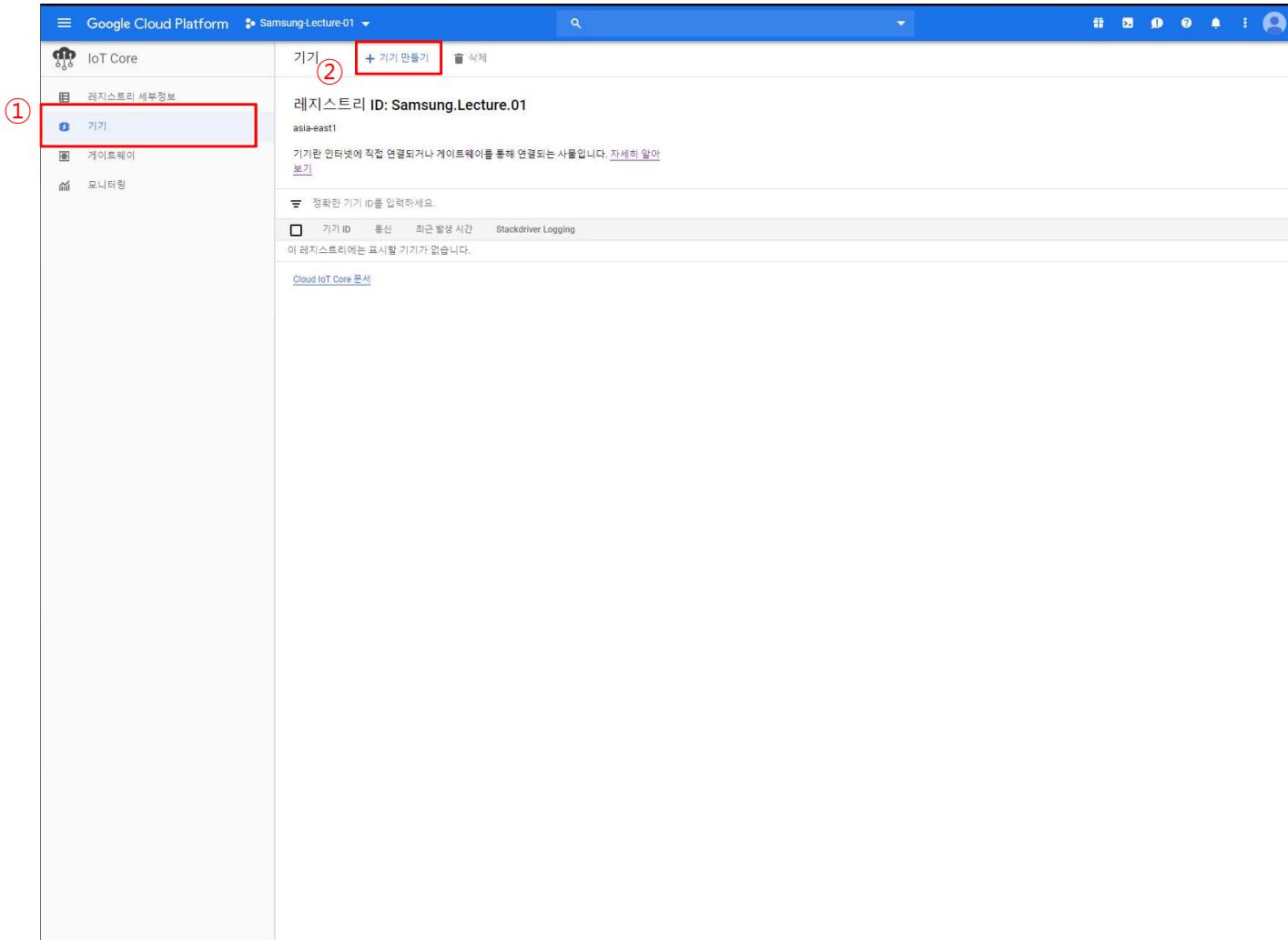
```

12:06:03 Build Finished (took 2s.474ms)

② 빌드 성공 여부 확인

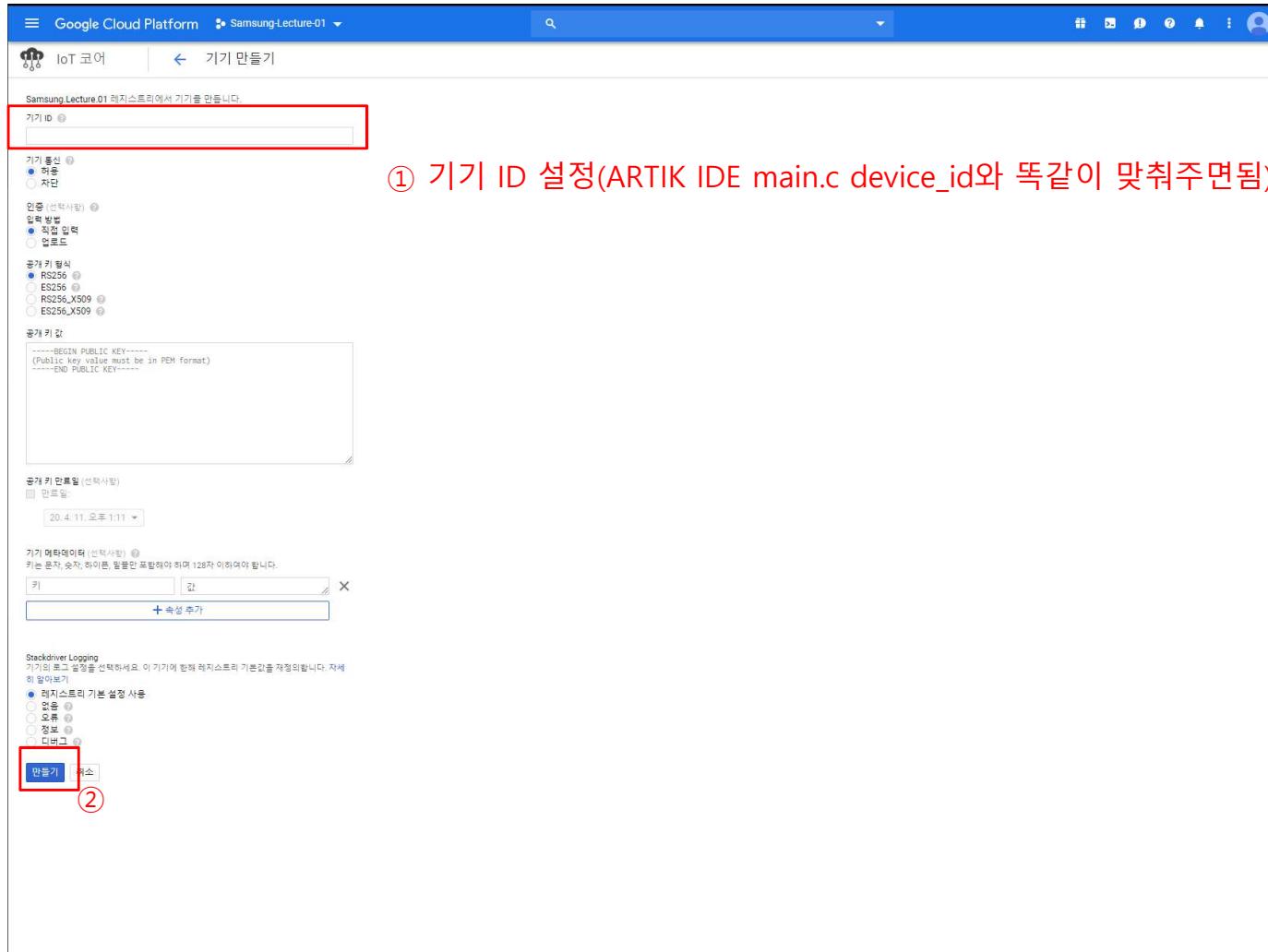
디바이스 만들기 -ARTIK 053

❖ IoT 코어 기기 등록 - 기기 만들기 클릭



디바이스 만들기 -ARTIK 053

❖ IoT 코어 기기 등록 - 디바이스 ID 설정



디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - 게이트웨이 ID 클릭

The screenshot shows the Google Cloud Platform IoT Core interface. The left sidebar has three main categories: '레지스트리 세부정보' (Registry Details), '기기' (Devices), and '게이트웨이' (Gateways). The '게이트웨이' category is highlighted with a red box and a circled '①'. The main content area displays a single gateway entry:

게이트웨이 ID	통신	최근 발생 시간	Stackdriver Logging
gateway01	허용됨	2019. 4. 10. 오후 7:44:42	레지스트리 기본값

A red box highlights the 'gateway01' entry, and a circled '②' points to it.

디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - 기기결합 클릭

The screenshot shows the Google Cloud Platform IoT Core interface for a gateway named 'gateway01'. The '기기 결합' (Device Connection) tab is selected. It displays a timeline of recent events:

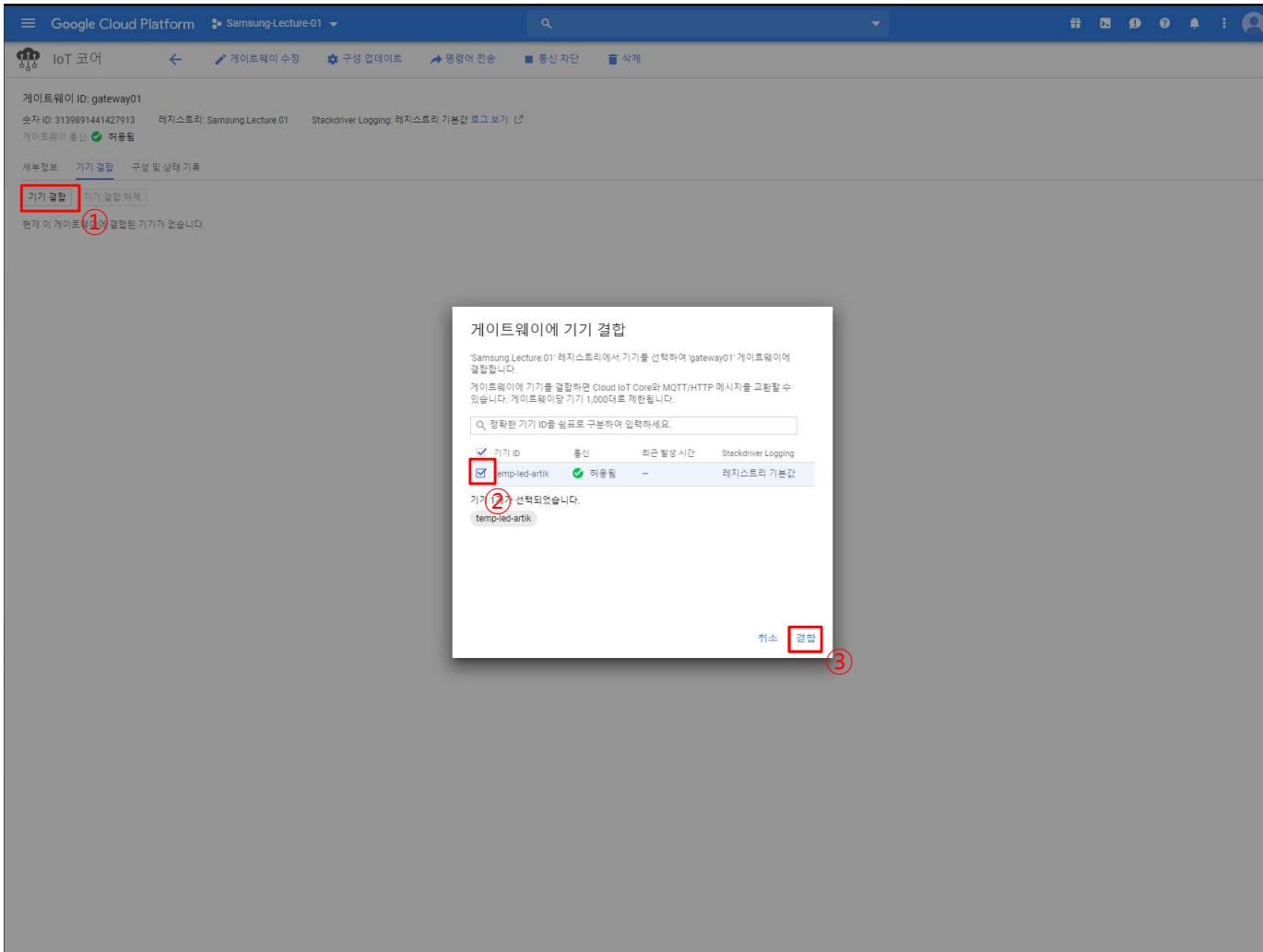
날짜	내용
2019. 4. 10. 오후 7:44:42	하트비트(MQTT란 해당)
—	접속 이벤트 수신
2019. 4. 10. 오후 1:35:43	게이트웨이 상태 이벤트 수신
2019. 4. 10. 오후 1:35:43	구성 전송
2019. 4. 10. 오후 7:45:00	구성 ACK(MQTT란 해당)
2019. 4. 10. 오후 7:45:00	오류

Below the timeline, there are sections for '송인 방법' (Delivery Method) and '설정' (Configuration). Under '설정', there is a '공개 키 추가' (Add Public Key) button. A table shows key details:

키 타입	키 값	만료 시간
키 활성	*****	—
RS256	*****	—

디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - Device ID 선택 및 결합



디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - Flash Project 후 연결확인

The screenshot displays two terminal windows side-by-side.

Top Terminal (Raspberry Pi - RaspberryPi-Gateway):

```

pi@raspberrypi: ~ /RaspberryPi-Gateway
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 6 for response ( "device": "temp-led-artik", "command": "event", "status": "ok" )
on_publish, userdata None, mid 6
pending response count 0
[Thu Apr 11 13:25:18 2019]: From Address 192.168.0.32:49154 receive data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 7 for response ( "device": "temp-led-artik", "command": "event", "status": "ok" )
on_publish, userdata None, mid 7
pending response count 0
[Thu Apr 11 13:25:22 2019]: From Address 192.168.0.32:49154 receive data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 8 for response ( "device": "temp-led-artik", "command": "event", "status": "ok" )
on_publish, userdata None, mid 8
pending response count 0
[Thu Apr 11 13:25:26 2019]: From Address 192.168.0.32:49154 receive data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 9 for response ( "device": "temp-led-artik", "command": "event", "status": "ok" )
on_publish, userdata None, mid 9
pending response count 0
[Thu Apr 11 13:25:30 2019]: From Address 192.168.0.32:49154 receive data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 10 for response ( "device": "temp-led-artik", "command": "event", "status": "ok" )
on_publish, userdata None, mid 10
pending response count 0
[Thu Apr 11 13:25:35 2019]: From Address 192.168.0.32:49154 receive data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 11 for response ( "device": "temp-led-artik", "command": "event", "status": "ok" )
on_publish, userdata None, mid 11
pending response count 0
[Thu Apr 11 13:25:39 2019]: From Address 192.168.0.32:49154 receive data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 12 for response ( "device": "temp-led-artik", "command": "event", "status": "ok" )
on_publish, userdata None, mid 12
pending response count 0

```

Bottom Terminal (ARTIK 053 - Putty):

```

COM10 - Putty
Autoboot in 50 milliseconds
QSPI flash initialized
# Starting application at 0x940C8020 ...
s5_stlflash_init: FLASH Quad Enabled
I2C_uicoreregister: Registering /dev/i2c-0
I2C_uicoreregister: Registering /dev/i2c-1
System Information:
Version: 1.0
    Commit: 412e99dfe7f02237d9463b59fb292a0ffcaacda
    Build User: ARTIK8Samsung
    Build Time: 2018-07-17 15:48:39
    System Time: 01 Jan 2010, 00:00:00 [s] UTC Hardware RTC Support
TASH>>>Starting supplicants in foreground...
Connect to Wi-Fi success
Get IP address
IP address: 192.168.0.32
Create the socket...
Set up address...
send data: ( "device": "temp-led-artik", "action": "detach" )
Received: ( "device": "temp-led-artik", "command": "detach", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "attach" )
Received: ( "device": "temp-led-artik", "command": "attach", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "subscribe" )
Received: ( "device": "temp-led-artik", "command": "subscribe", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )
send data: ( "device": "temp-led-artik", "action": "event", "data": "temp=25" )
Received: ( "device": "temp-led-artik", "command": "event", "status": "ok" )

```

디바이스 만들기 -ARTIK 053

❖ 디바이스 구성 업데이트 테스트 - Device ID 클릭

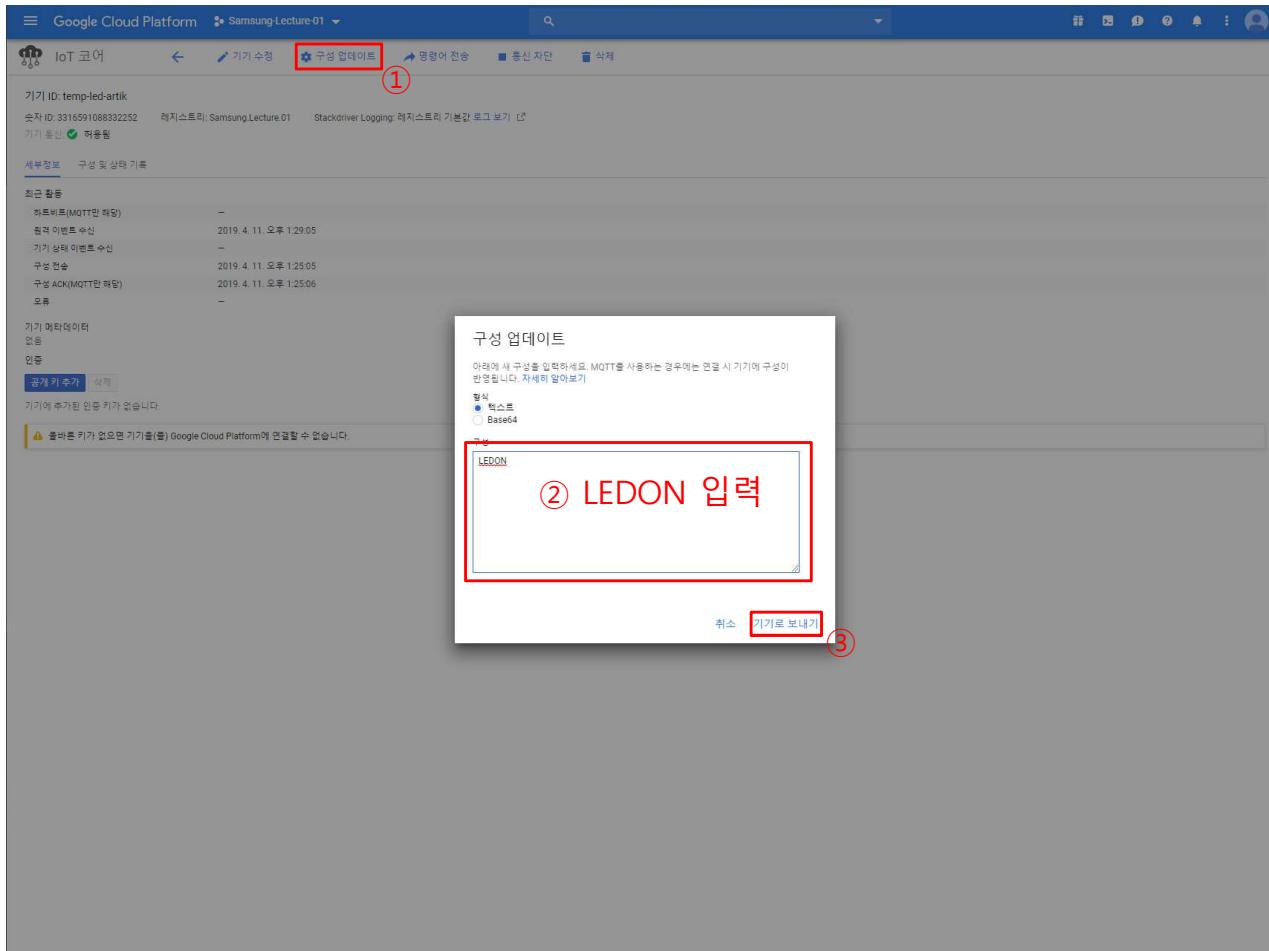
The screenshot shows the Google Cloud Platform IoT Core console. The left sidebar has options: 레지스트리 세부정보, 기기 (selected), 게이트웨이, and 모니터링. The main area displays a registry entry for 'Samsung.Lecture.01' in 'asia-east1'. It shows a table with one row:

기기 ID	종신	최근 발생 시간	Stackdriver Logging
temp-led-artik	허용됨	2019. 4. 11. 오후 1:28:03	레지스트리 기본값

A red box highlights the 'temp-led-artik' entry in the table. At the bottom of the page is a URL: <https://console.cloud.google.com/iot/locations/asia-east1/registries/Samsung.Lecture.01/devices/temp-led-artik?authuser=1&project=samsung-lecture-01>.

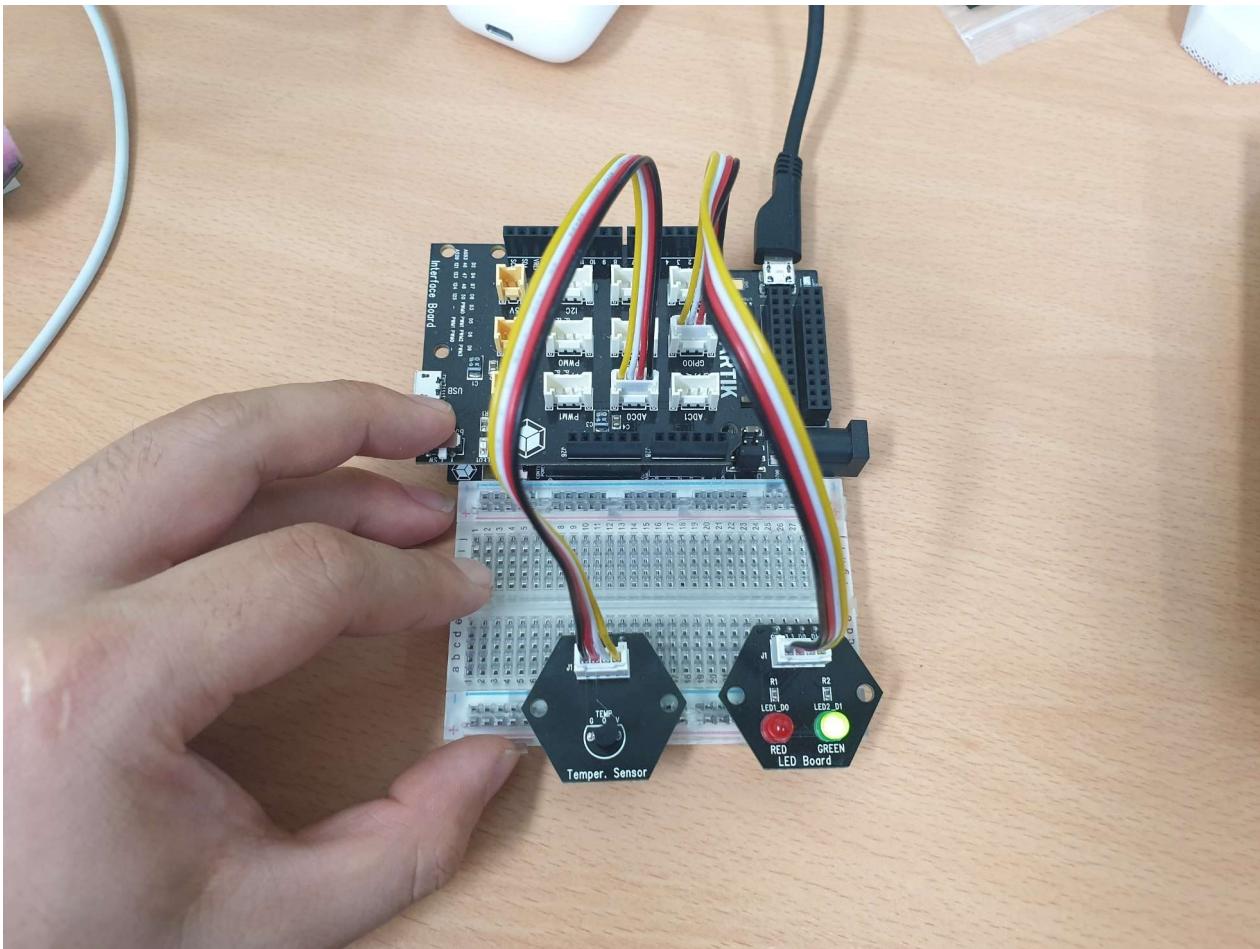
디바이스 만들기 -ARTIK 053

❖ 디바이스 구성 업데이트 테스트 - LEDON을 기기로 보내기

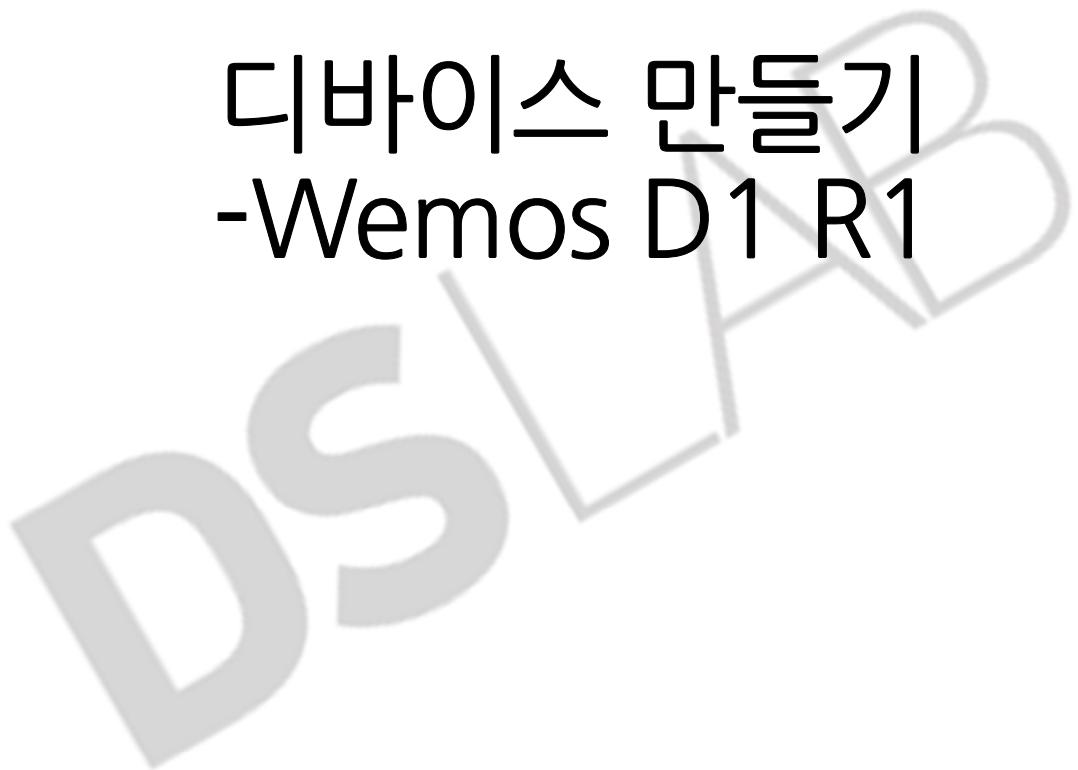


디바이스 만들기 -ARTIK 053

- ❖ 디바이스 구성 업데이트 테스트 - D4번 LED가 켜지는지 확인

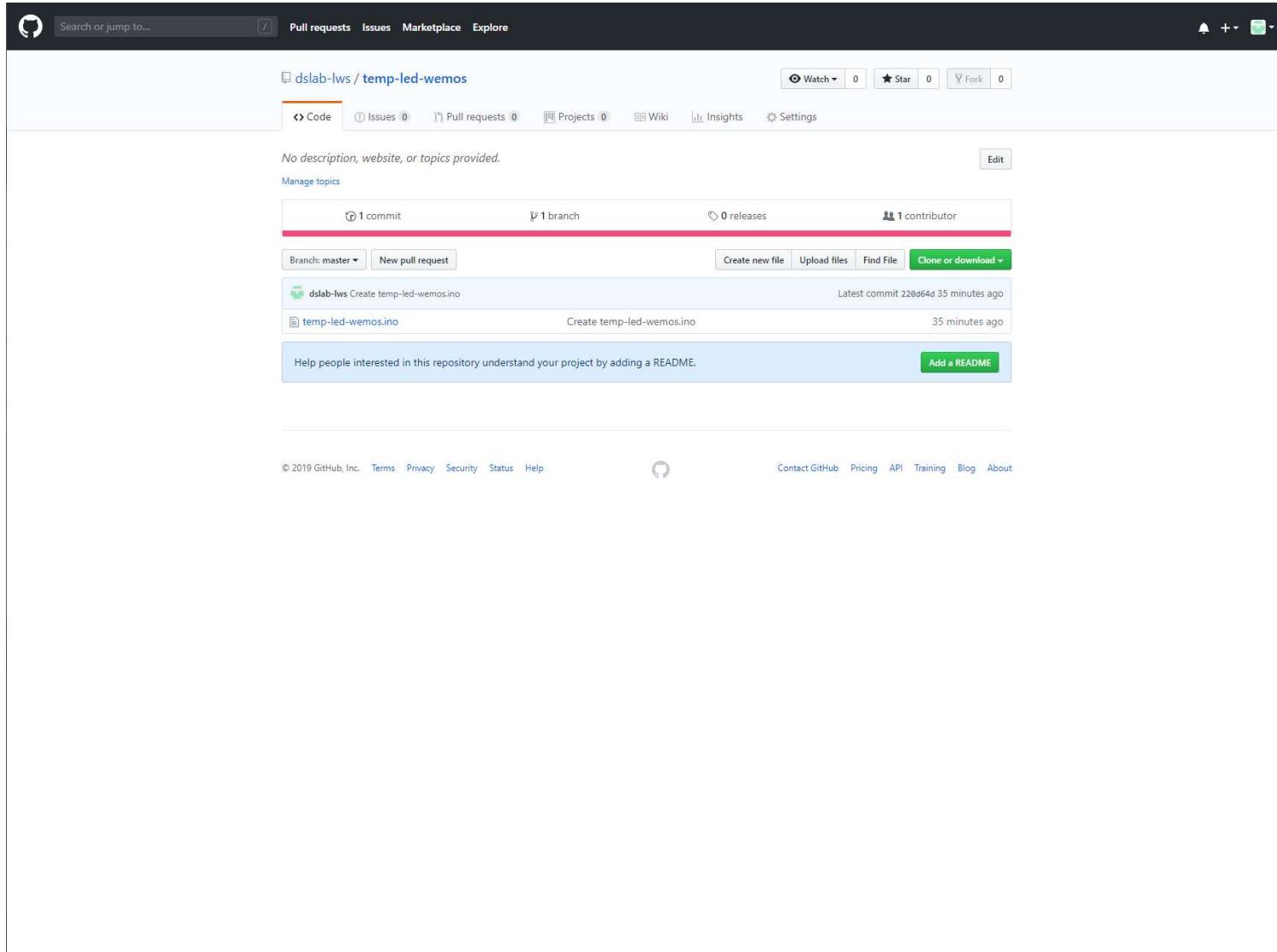


디바이스 만들기 -Wemos D1 R1



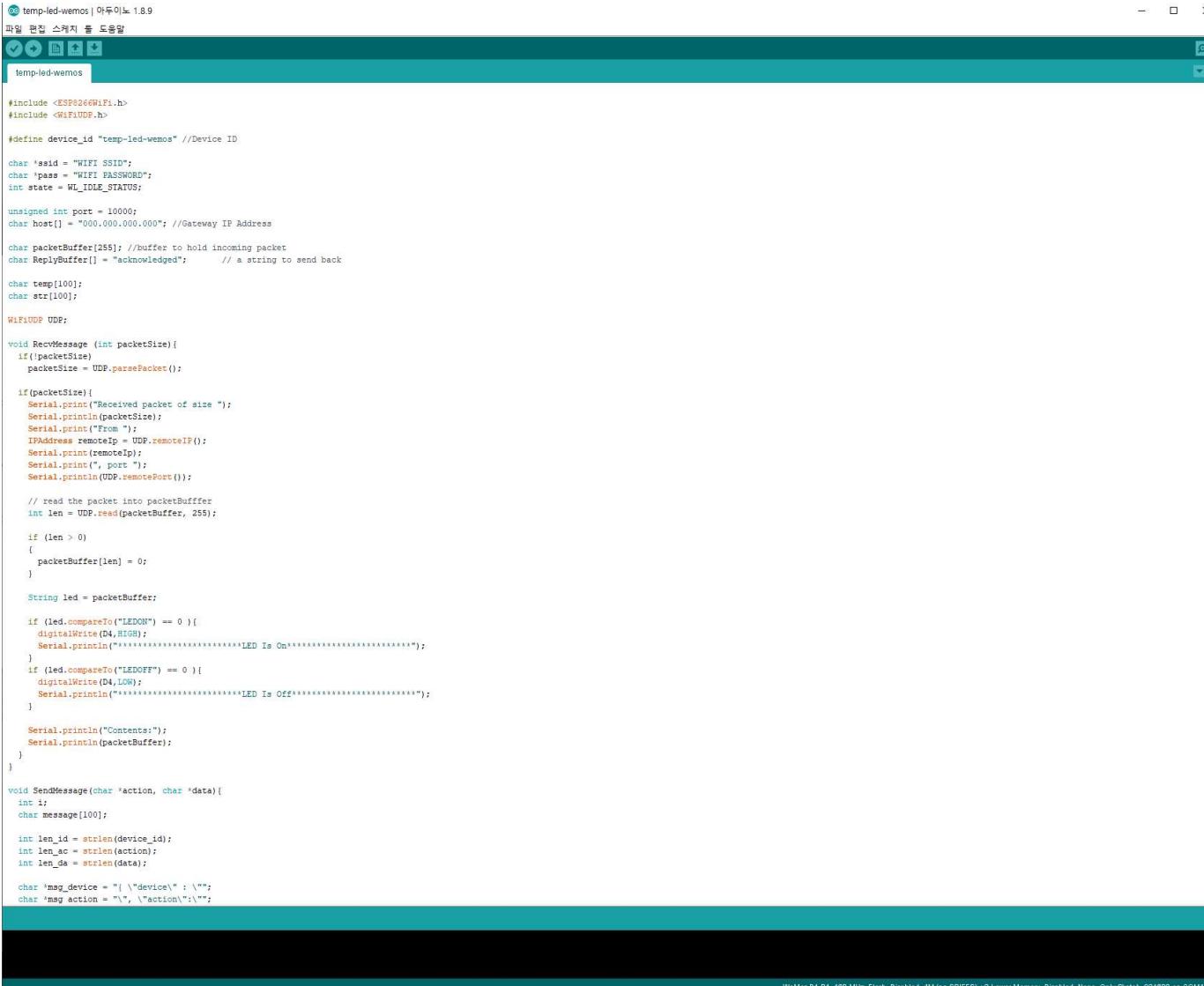
디바이스 만들기 - Wemos D1 R1

- ❖ Wemos D1 R1 세팅 - Wemos 소스코드 다운로드 <https://github.com/dslab-lws/temp-led-wemos>



디바이스 만들기 - Wemos D1 R1

❖ Wemos D1 R1 세팅 - 소스코드 열기



The screenshot shows the Arduino IDE interface with the sketch titled "temp-led-wemos". The code implements a simple WiFi server that listens on port 10000. It reads incoming UDP packets, checks if they contain "LEDON" or "LEDOFF", and then turns the LED on or off accordingly. The sketch also includes a function to send messages back to the client.

```
#include <ESP8266WiFi.h>
#include <WiFiUDP.h>

#define device_id "temp-led-wemos" //Device ID

char *ssid = "WIFI SSID";
char *pass = "WIFI PASSWORD";
int state = WL_IDLE_STATUS;

unsigned int port = 10000;
char host[] = "000.000.000.000"; //Gateway IP Address

char packetBuffer[255]; //buffer to hold incoming packet
char ReplyBuffer[] = "acknowledged"; // a string to send back

char temp[100];
char str[100];

WiFiUDP UDP;

void RecvMessage (int packetSize){
if(packetSize)
    packetSize = UDP.parsePacket();

if(packetSize){
    Serial.print("Received packet of size ");
    Serial.println(packetSize);
    Serial.print("From ");
    IPAddress remoteIp = UDP.remoteIP();
    Serial.print(remoteIp);
    Serial.print(", port ");
    Serial.println(UDP.remotePort());

    // read the packet into packetBuffer
    int len = UDP.read(packetBuffer, 255);

    if (len > 0)
    {
        packetBuffer[len] = 0;
    }

    String led = packetBuffer;
    if (led.compareTo("LEDON") == 0 ){
        digitalWrite(D4,HIGH);
        Serial.println("*****LED Is On*****");
    }
    if (led.compareTo("LEDOFF") == 0 ){
        digitalWrite(D4,LOW);
        Serial.println("*****LED Is Off*****");
    }

    Serial.println("Contents:");
    Serial.println(packetBuffer);
}
}

void SendMessage(char *action, char *data){
int i;
char message[100];

int len_id = strlen(device_id);
int len_ac = strlen(action);
int len_da = strlen(data);

char *mag_device = "|\"device\" : \"";
char *mag_action = "\", \"action\":\"";
```

디바이스 만들기 - Wemos D1 R1

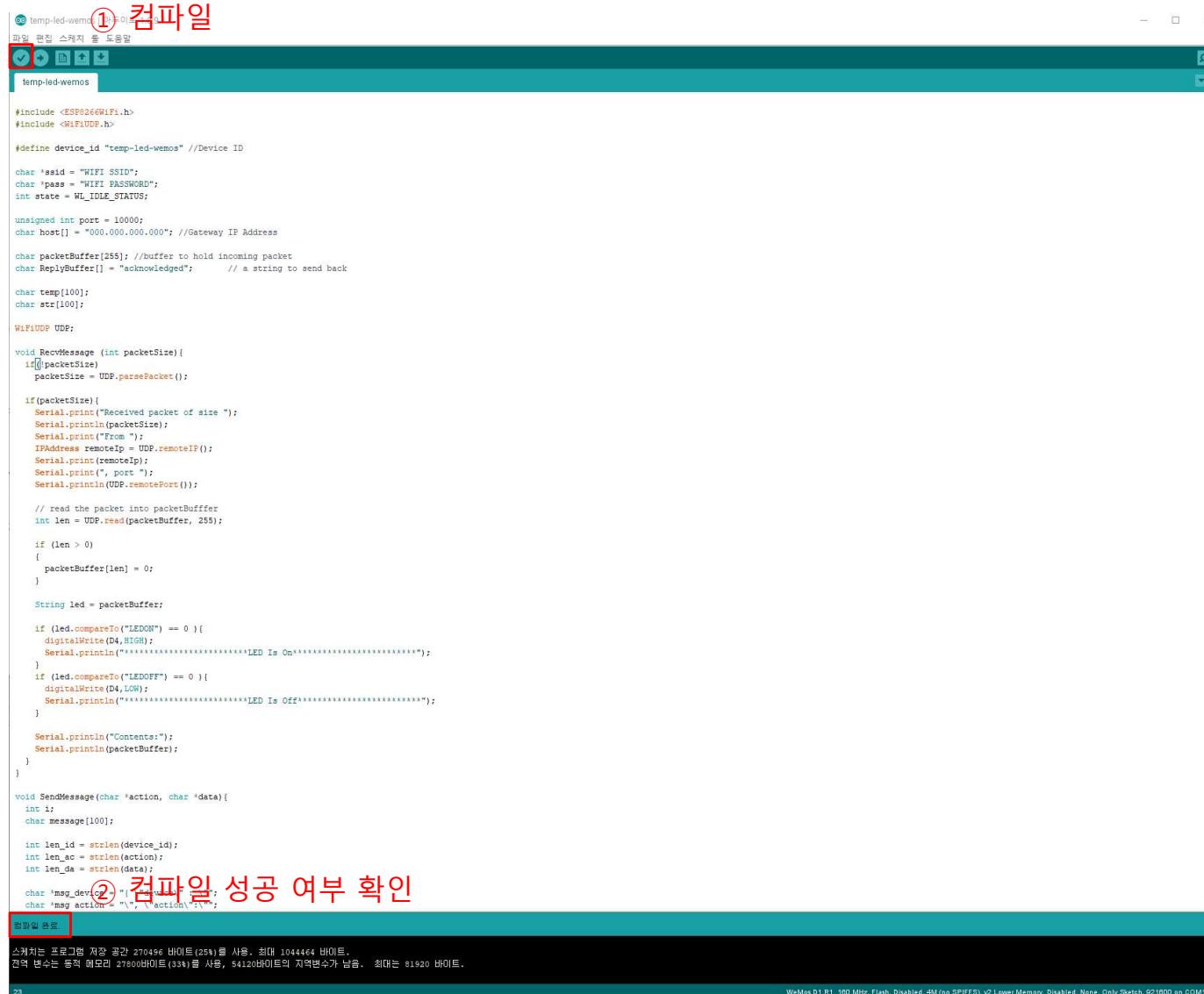
❖ Wemos D1 R1 세팅 - 소스코드 수정

```
#define device_id "temp-led-wemos" //Device ID  
  
char *ssid = "WIFI SSID";  
char *pass = "WIFI PASSWORD";  
int state = WL_IDLE_STATUS; 와이파이 이름과 비밀번호 입력  
  
unsigned int port = 10000; Gateway IP 입력  
char host[] = "000.000.000.000"; //Gateway IP Address
```

디바이스 만들기 - Wemos D1 R1

❖ Wemos D1 R1 세팅 - 소스코드 컴파일 및 에러확인

① 컴파일



```
#include <ESP8266WiFi.h>
#include <WiFiUDP.h>

#define device_id "temp-led-wemos" //Device ID

char *ssid = "WIFI SSID";
char *pass = "WIFI PASSWORD";
int state = WL_IDLE_STATUS;

unsigned int port = 10000;
char host[] = "0.0.0.0.0.0.0.0"; //Gateway IP Address

char packetBuffer[255]; //buffer to hold incoming packet
char ReplyBuffer[] = "acknowledged"; // a string to send back

char temp[100];
char str[100];

WiFiUDP UDP;

void RecvMessage (int packetSize){
    if(packetSize)
        packetSize = UDP.parsePacket();

    if(packetSize){
        Serial.print("Received packet of size ");
        Serial.println(packetSize);
        Serial.print("From ");
        IPAddress remoteIp = UDP.remoteIP();
        Serial.print(remoteIp);
        Serial.print(", port ");
        Serial.println(UDP.remotePort());

        // read the packet into packetBuffer
        int len = UDP.read(packetBuffer, 255);

        if (len > 0)
        {
            packetBuffer[len] = 0;
        }

        String led = packetBuffer;
        if (led.compareTo("LEDON") == 0 ){
            digitalWrite(D4,HIGH);
            Serial.println("*****LED Is On*****");
        }
        if (led.compareTo("LEDOFF") == 0 ){
            digitalWrite(D4,LOW);
            Serial.println("*****LED Is Off*****");
        }

        Serial.println("Contents:");
        Serial.println(packetBuffer);
    }
}

void SendMessage(char *action, char *data){
    int i;
    char message[100];

    int len_id = strlen(device_id);
    int len_ac = strlen(action);
    int len_da = strlen(data);

    char *msg_device;
    char *msg_action;
    char *msg_content;

    msg_device = device_id;
    msg_action = action;
    msg_content = data;
}
```

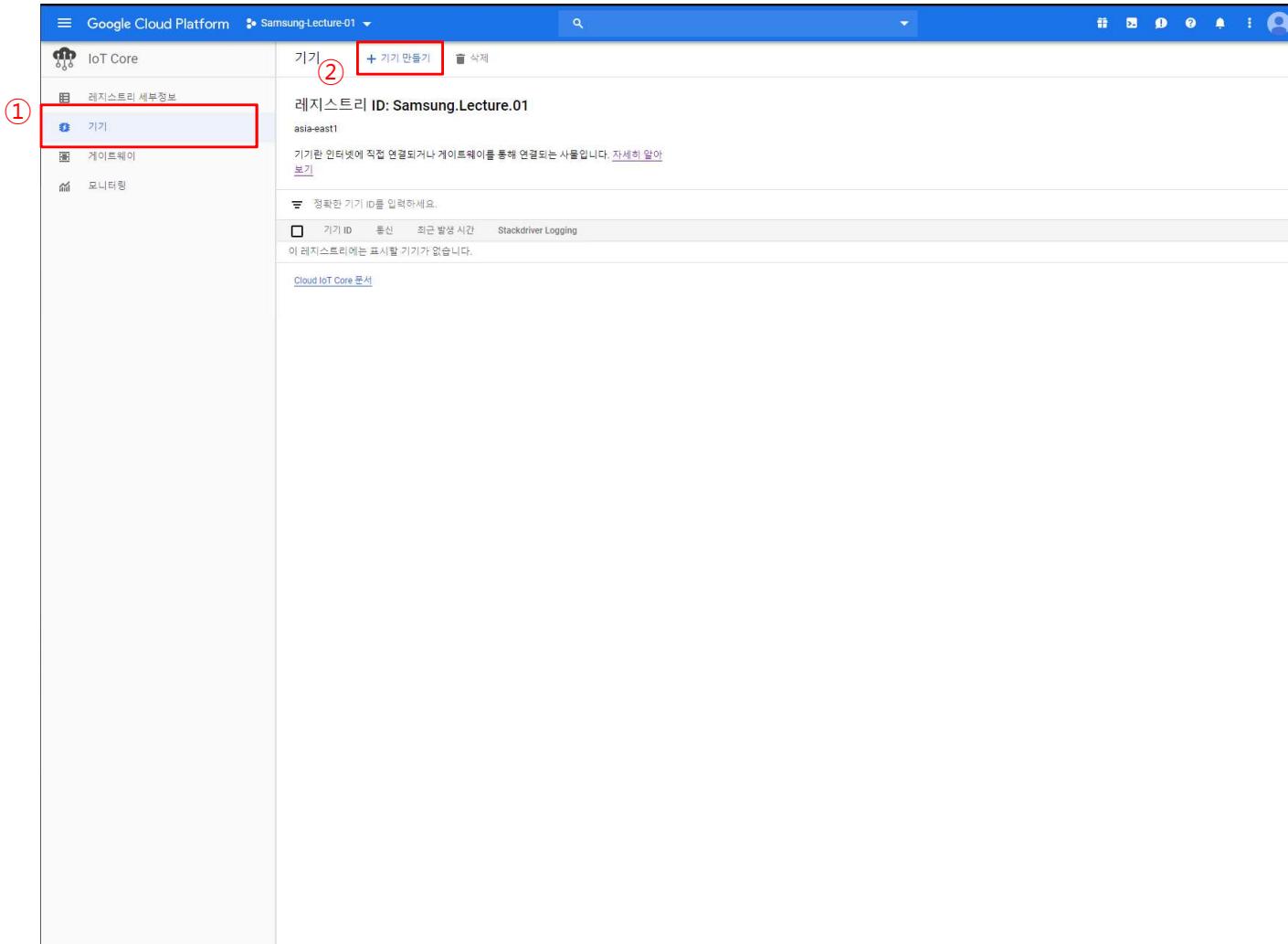
② 컴파일 성공 여부 확인

Sketch file successfully compiled.

Sketch is 270496 bytes (25%) and uses at most 1044464 bytes.
Memory usage is 27800 bytes (33%) and uses at most 54120 bytes of external memory. Max memory usage is 81920 bytes.

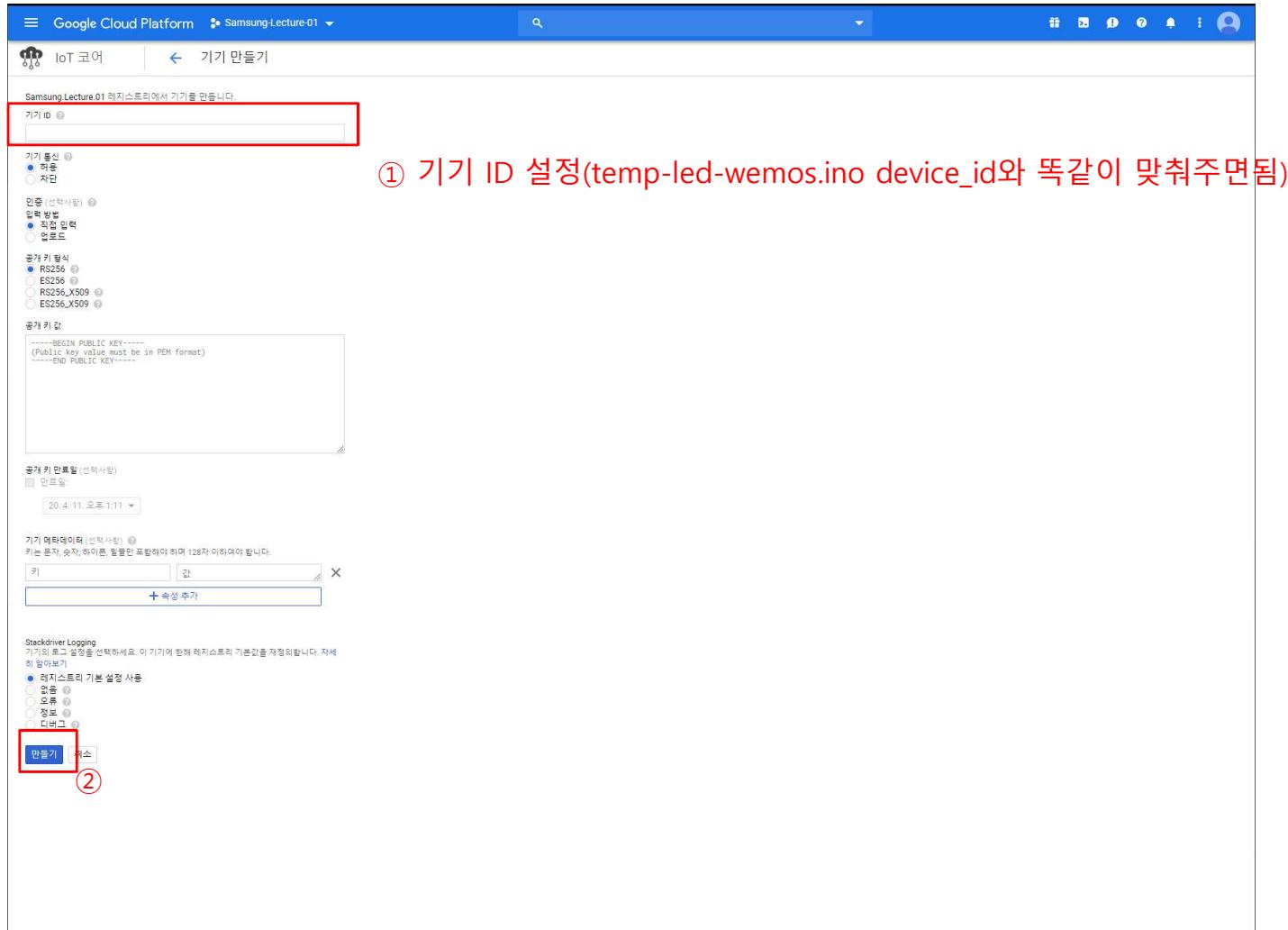
디바이스 만들기 - Wemos D1 R1

❖ IoT 코어 기기 등록 - 기기 만들기 클릭



디바이스 만들기 - Wemos D1 R1

❖ IoT 코어 기기 등록 - 디바이스 ID 설정



디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - 게이트웨이 ID 클릭

The screenshot shows the Google Cloud Platform IoT Core interface. The left sidebar has 'IoT Core' selected, with '게이트웨이' (Gateway) highlighted by a red box and circled with a red number ①. The main pane displays a list of devices under the heading '레지스트리 ID: Samsung.Lecture.01' and 'asia-east1'. A single device, 'gateway01', is listed with its status as '허용됨' (Allowed). The row for 'gateway01' is also highlighted with a red box and circled with a red number ②.

디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - 기기결합 클릭

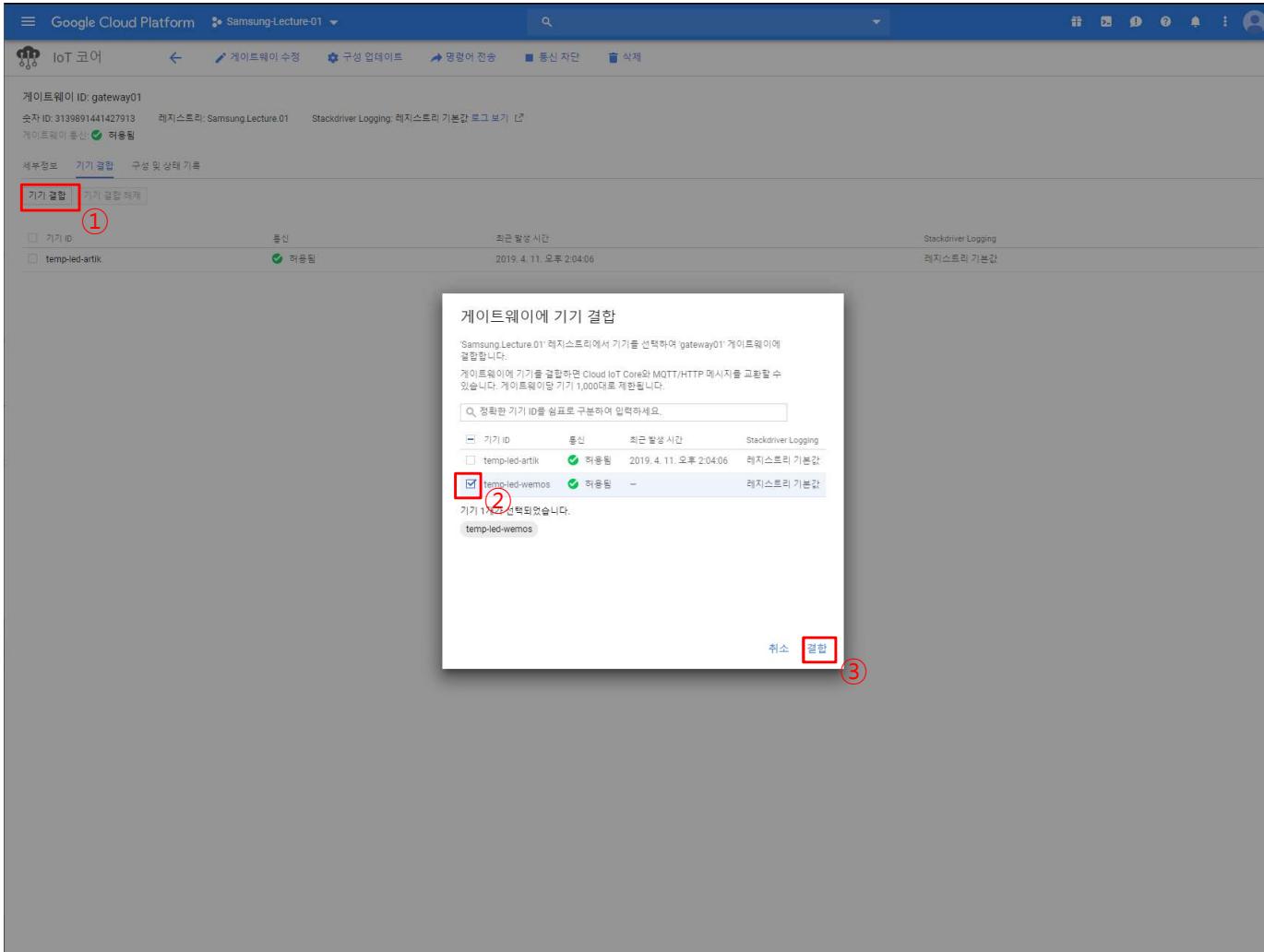
The screenshot shows the Google Cloud Platform IoT Core interface for a gateway named 'gateway01'. The '기기 결합' (Device Connection) tab is selected. The log table displays the following data:

시작일자	제작자	내용	날짜
2019. 4. 10.	IoT 코어	하트비트(MQTT란 해당)	2019. 4. 10. 오후 7:44:42
		접속 이벤트 수신	-
		게이트웨이 상태 이벤트 수신	-
		구성 전송	2019. 4. 10. 오후 1:35:43
		구성 ACK(MQTT란 해당)	2019. 4. 10. 오후 1:35:43
		오류	① 2019. 4. 10. 오후 7:45:00

Below the log table, there are sections for '승인 방법' (Approval Method) and '공개 키 추가' (Add Public Key). The '승인 방법' section includes options for '연결만' (Connection only), '게이트웨이 미터데이터' (Gateway metering data), '값만' (Values only), and '인증' (Authentication). The '공개 키 추가' section contains fields for '키 활성' (Key active), '키 값' (Key value), and '만료 시간' (Expiration time), with 'RS256' selected as the key type.

디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - Device ID 선택 및 결합



디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - Upload Project 후 연결확인

The screenshot displays two terminal windows. The top window shows the Raspberry Pi Gateway (RaspberryPi-Gateway) receiving and publishing MQTT messages. The bottom window shows the Wemos D1 R1 device (COM12) receiving and publishing MQTT messages.

RaspberryPi-Gateway Terminal Output:

```
pi@raspberrypi:~/RaspberryPi-Gateway
Save mid 4 for response ["device": temp-led-wemos, "command": "subscribe", "status": "ok"]
...
Received message '' on topic '/devices/temp-led-wemos/config' with Qos 1
Received message '' on topic '/devices/temp-led-wemos/config' with Qos 1
[Thu Apr 11 14:25:02 2019]: From Address 192.168.0.36:10000 receive data: { "device" : "temp-led-wemos", "action":"event", "data":"temp=23" }
Sending telemetry event for device temp-led-wemos
Publishing message to topic /devices/temp-led-wemos/events with payload 'temp=23'
Save mid 5 for response ["device": temp-led-wemos, "command": "event", "status": "ok"]
...
on publish, userdata None, mid 5
pending response count 0
[Thu Apr 11 14:25:06 2019]: From Address 192.168.0.36:10000 receive data: { "device" : "temp-led-wemos", "action":"event", "data":"temp=23" }
Sending telemetry event for device temp-led-wemos
Publishing message to topic /devices/temp-led-wemos/events with payload 'temp=23'
Save mid 6 for response ["device": temp-led-wemos, "command": "event", "status": "ok"]
...
on publish, userdata None, mid 6
pending response count 0
[Thu Apr 11 14:25:10 2019]: From Address 192.168.0.36:10000 receive data: { "device" : "temp-led-wemos", "action":"event", "data":"temp=23" }
Sending telemetry event for device temp-led-wemos
Publishing message to topic /devices/temp-led-wemos/events with payload 'temp=23'
Save mid 7 for response ["device": temp-led-wemos, "command": "event", "status": "ok"]
...
on publish, userdata None, mid 7
pending response count 0
[Thu Apr 11 14:25:14 2019]: From Address 192.168.0.36:10000 receive data: { "device" : "temp-led-wemos", "action":"event", "data":"temp=24" }
Sending telemetry event for device temp-led-wemos
Publishing message to topic /devices/temp-led-wemos/events with payload 'temp=24'
Save mid 8 for response ["device": temp-led-wemos, "command": "event", "status": "ok"]
...
on publish, userdata None, mid 8
pending response count 0
[Thu Apr 11 14:25:18 2019]: From Address 192.168.0.36:10000 receive data: { "device" : "temp-led-wemos", "action":"event", "data":"temp=23" }
Sending telemetry event for device temp-led-wemos
Publishing message to topic /devices/temp-led-wemos/events with payload 'temp=23'
Save mid 9 for response ["device": temp-led-wemos, "command": "event", "status": "ok"]
...
on publish, userdata None, mid 9
pending response count 0
[Thu Apr 11 14:25:23 2019]: From Address 192.168.0.36:10000 receive data: { "device" : "temp-led-wemos", "action":"event", "data":"temp=23" }
Sending telemetry event for device temp-led-wemos
Publishing message to topic /devices/temp-led-wemos/events with payload 'temp=23'
Save mid 10 for response ["device": temp-led-wemos, "command": "event", "status": "ok"]
...
on publish, userdata None, mid 10
pending response count 0
```

Wemos D1 R1 Terminal Output:

```
From 192.168.0.21, port 10000
Content:
{"device": temp-led-wemos, "command": "attach", "status": "ok"}
Send message : {"device": "temp-led-wemos", "action": "subscribe", "data": ""}

Received packet of size 69
From 192.168.0.21 port 10000
Content:
{"device": temp-led-wemos, "command": "subscribe", "status": "ok"}
Send message : {"device": "temp-led-wemos", "action": "event", "data": "temp=23" }

Received packet of size 64
From 192.168.0.21 port 10000
Content:
{"device": temp-led-wemos, "command": "event", "status": "ok"}
Send message : {"device": "temp-led-wemos", "action": "event", "data": "temp=23" }

Received packet of size 64
From 192.168.0.21 port 10000
Content:
{"device": temp-led-wemos, "command": "event", "status": "ok"}
Send message : {"device": "temp-led-wemos", "action": "event", "data": "temp=24" }

Received packet of size 64
From 192.168.0.21 port 10000
Content:
{"device": temp-led-wemos, "command": "event", "status": "ok"}
Send message : {"device": "temp-led-wemos", "action": "event", "data": "temp=24" }

Received packet of size 64
From 192.168.0.21 port 10000
Content:
{"device": temp-led-wemos, "command": "event", "status": "ok"}
Send message : {"device": "temp-led-wemos", "action": "event", "data": "temp=23" }

Received packet of size 64
From 192.168.0.21 port 10000
Content:
{"device": temp-led-wemos, "command": "event", "status": "ok"}
Send message : {"device": "temp-led-wemos", "action": "event", "data": "temp=23" }
```

디바이스 만들기 - Wemos D1 R1

❖ 디바이스 구성 업데이트 테스트 - Device ID 클릭

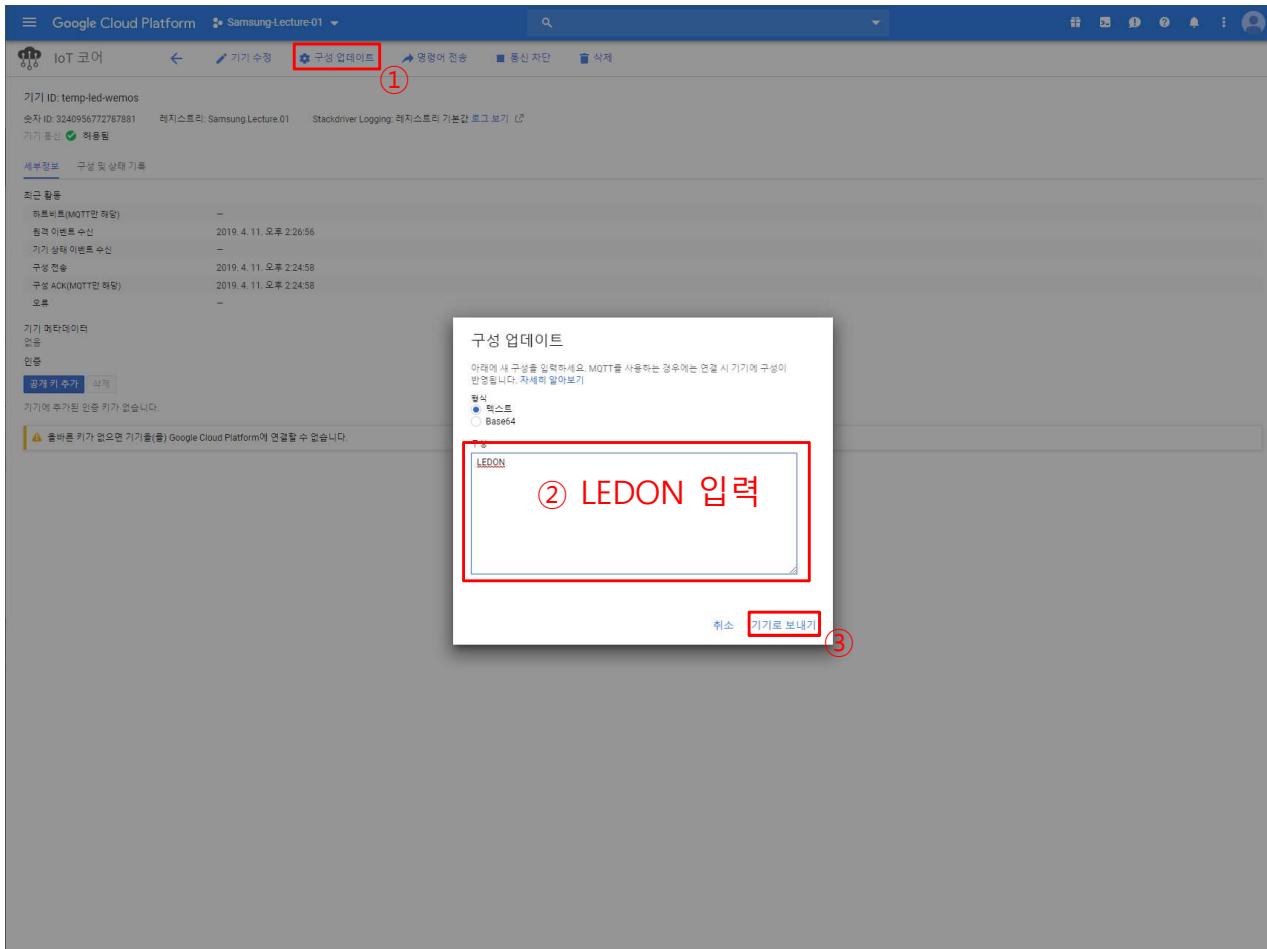
The screenshot shows the Google Cloud Platform IoT Core console for the project "Samsung-Lecture-01". The left sidebar lists "IoT Core", "레지스트리 세부정보" (Registry Details), "기기" (Devices), "케이트웨이" (Gateway), and "모니터링" (Monitoring). The main pane displays the "Devices" section with the heading "레지스트리 ID: Samsung.Lecture.01" and location "asia-east1". It includes a note about connecting via WiFi or Ethernet. Below is a table titled "정확한 기기 ID를 입력하세요" (Enter the exact device ID) with two rows:

기기 ID	통신	최근 발생 시간	Stackdriver Logging
temp-led-artik	✓ 허용됨	2019. 4. 11. 오후 2:26:06	레지스트리 기본값
temp-led-wemos	✓ 허용됨	2019. 4. 11. 오후 2:25:56	레지스트리 기본값

At the bottom of the table, there is a link "Cloud IoT Core 문서".

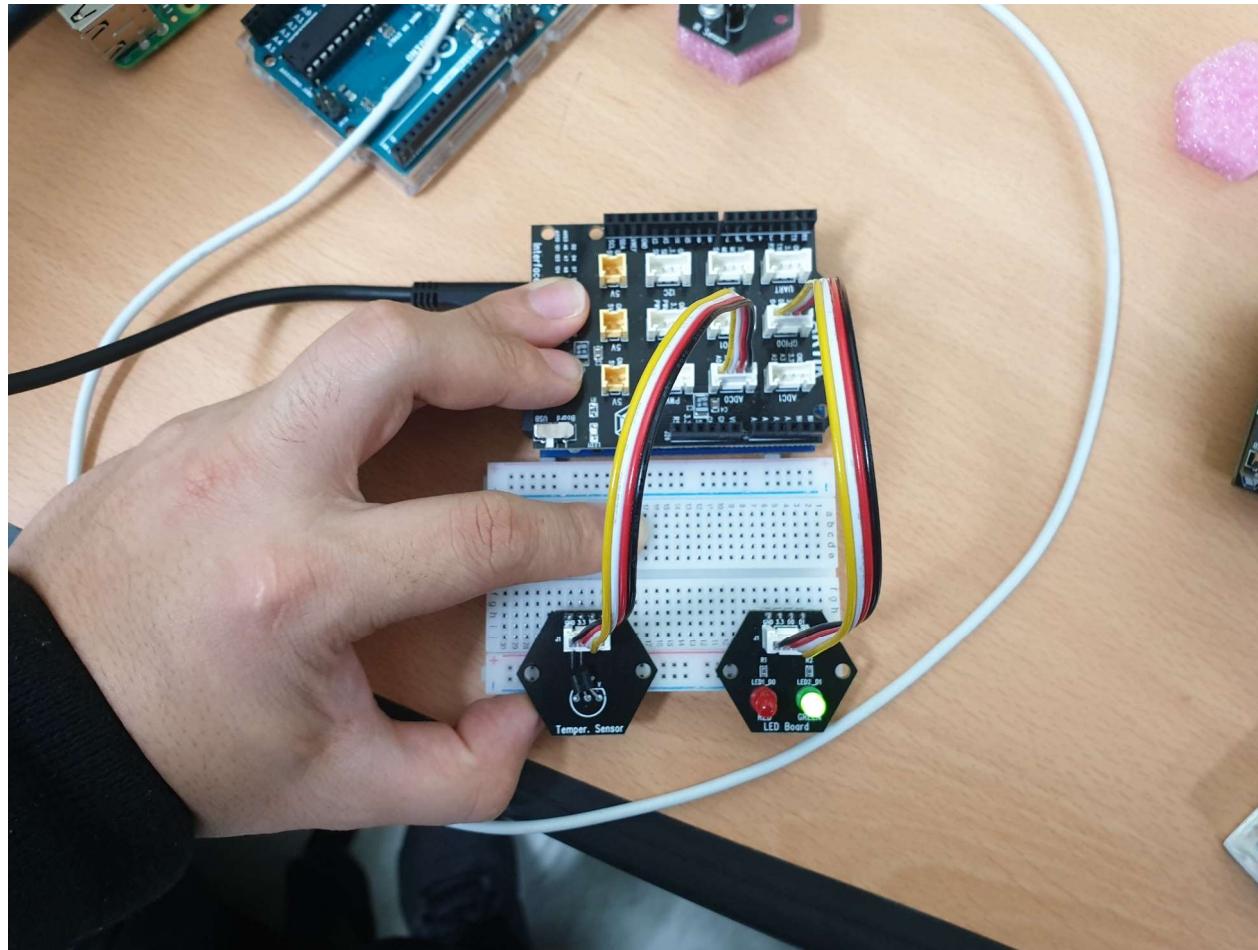
디바이스 만들기 - Wemos D1 R1

❖ 디바이스 구성 업데이트 테스트 - LEDON을 기기로 보내기



디바이스 만들기 - Wemos D1 R1

- ❖ 디바이스 구성 업데이트 테스트 - D4번 LED가 켜지는지 확인



OAuth 2.0 Access Token 받기



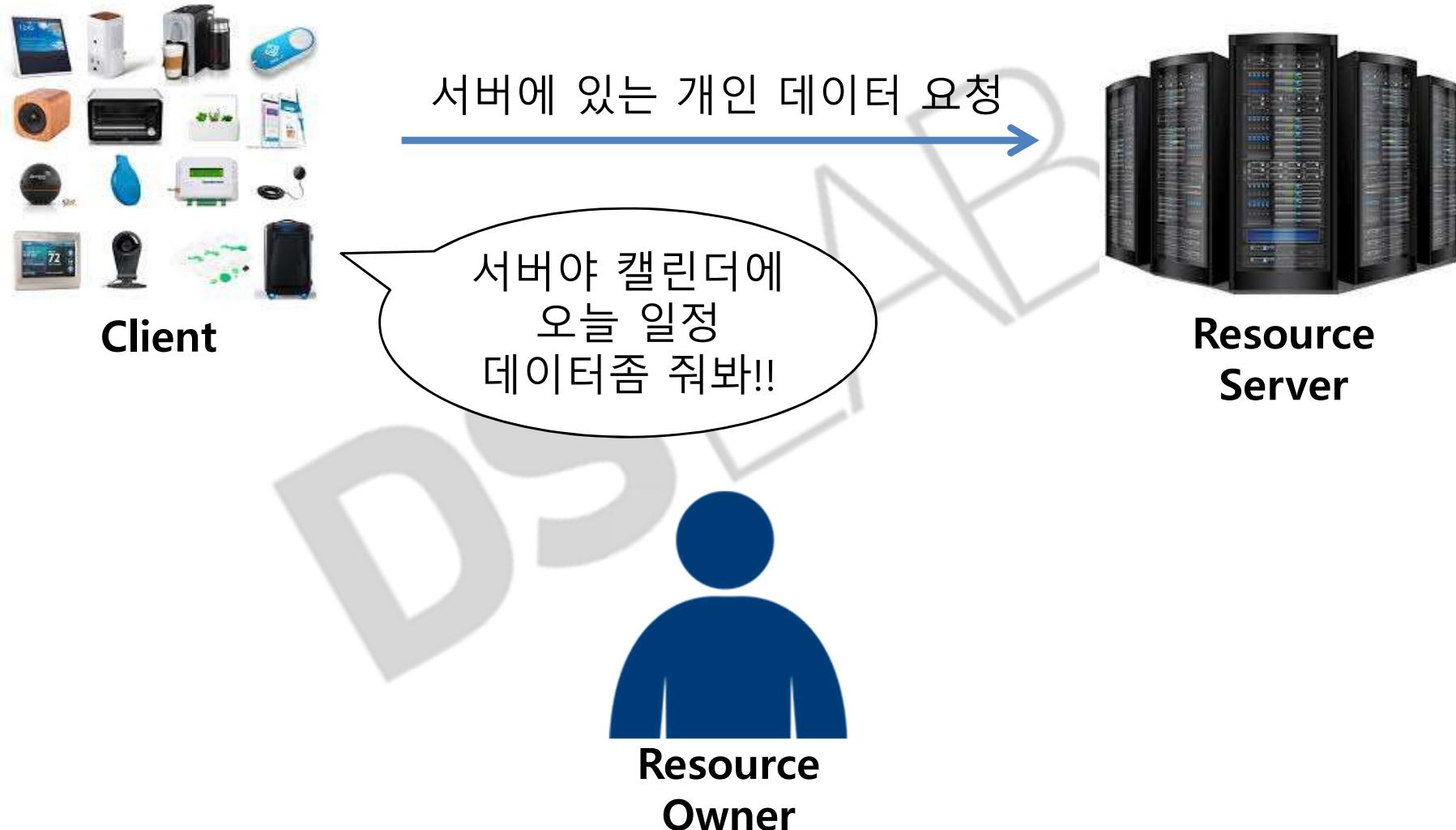
OAuth 2.0 Access Token 받기

❖ OAuth 2.0이란?

- 대부분의 서비스는 인증(Authentication)과 리소스에 대한 권한부여(Authorization) 기능이 필요
- 인증과 권한부여 기능은 다양한 방법으로 제공되고 있는데 대표적인 방법으로 OAuth를 사용
- Facebook, Google, Twitter 등 대형 서비스에서 널리 사용되고 있기 때문에 많은 개발자에게 친숙
- 리소스 서버(Facebook, Google, Twitter 등)와 클라이언트(IOT장치, 웹, 앱 등) 사이에 인증을 완료하면 서버는 권한부여의 결과로써 access token을 전송하고, 클라이언트는 access token을 이용해서 접근 및 서비스를 요청, 서버는 access token 기반으로 서비스와 권한을 확인하여 접근을 허용할지 말지를 결정하고, 결과 데이터를 클라이언트에게 보내줌

OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 이유



OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 이유



Client

사용자한테
물어볼게!!



Resource
Server

동의 및 로그인 요청



Resource
Owner

데이터 넘겨줘도
괜찮아??

OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 이유



OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 이유



Client

데이터
넘겨주래!!

개인 데이터 전송



Resource
Server

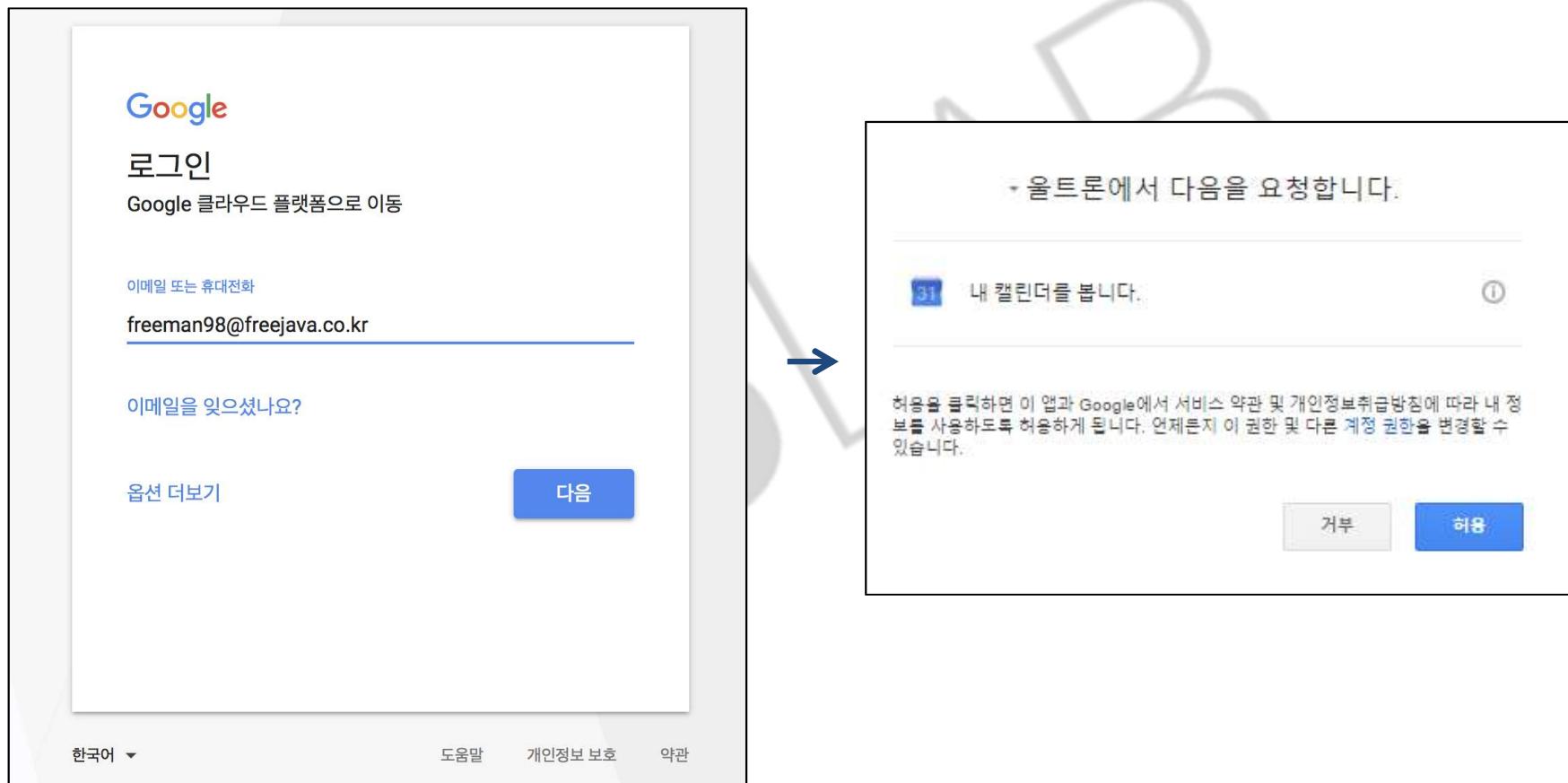


Resource
Owner

OAuth 2.0 Access Token 받기

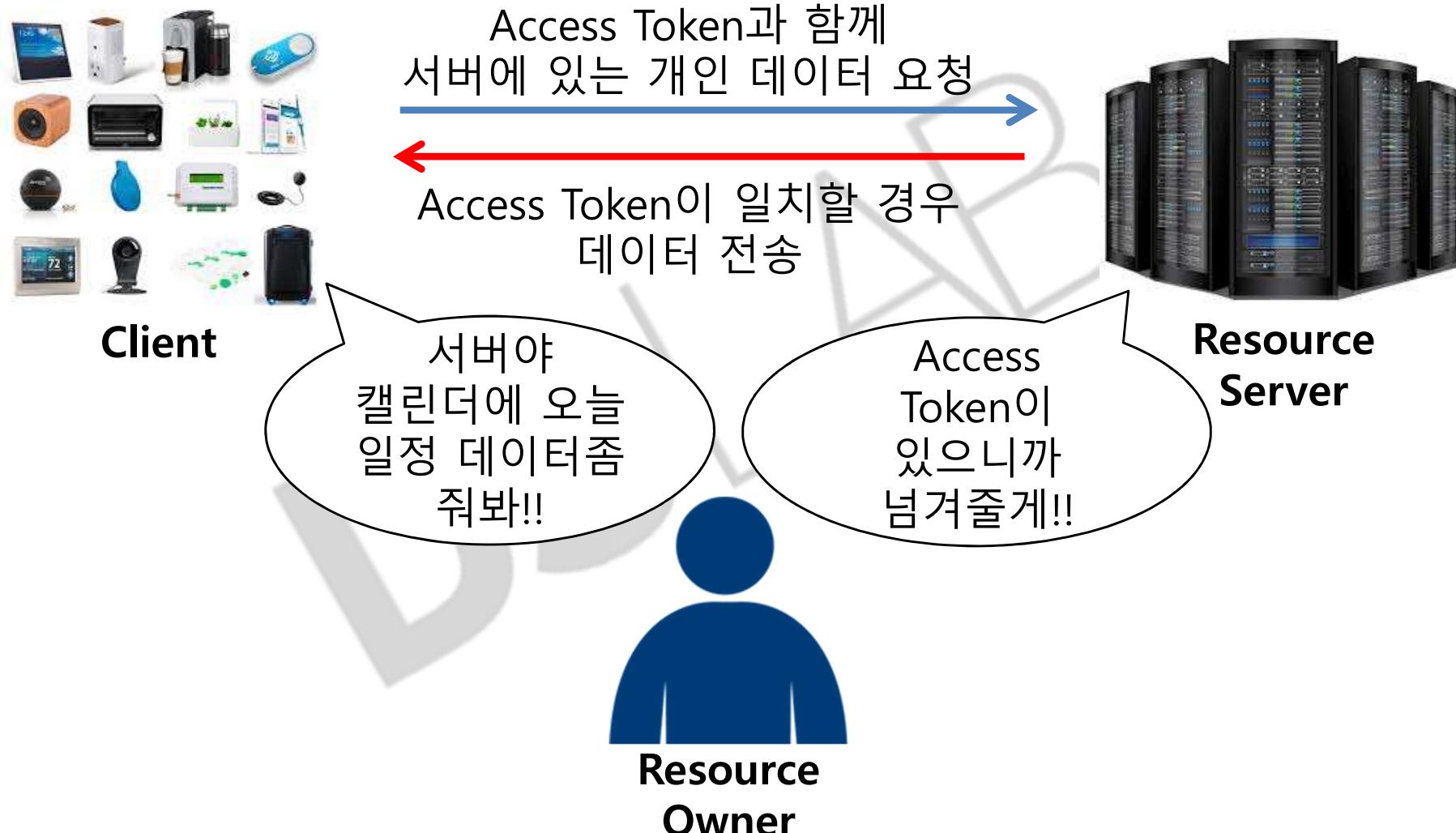
❖ Access Token을 받는 이유

Access Token이 없으면 클라이언트가 데이터를 갱신할 때마다 동의 및 로그인을 해 줘야함



OAuth 2.0 Access Token 받기

❖ Access Token을 받는 이유



OAuth 2.0 Access Token 받기

❖ Access Token을 받는 과정

1. 리소스서버에서 Client ID 및 Client Secret을 발급
2. 사용자에게 클라이언트 ID 사용 동의 요청
3. 사용자가 동의를 할 경우 Access Token을 발급받기 위한 Code가 생성
4. Code와 Client ID 및 Client Secret을 가지고 Access Token 및 Refresh Token을 발급
5. Access Token은 발급받으면 1시간 후에 영구적으로 삭제되기 때문에 Refresh Token을 사용하여 Access Token을 갱신

OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 과정

① 클라이언트 ID 생성



Client



Resource
Server



Resource
Owner

OAuth 2.0 Access Token 받기

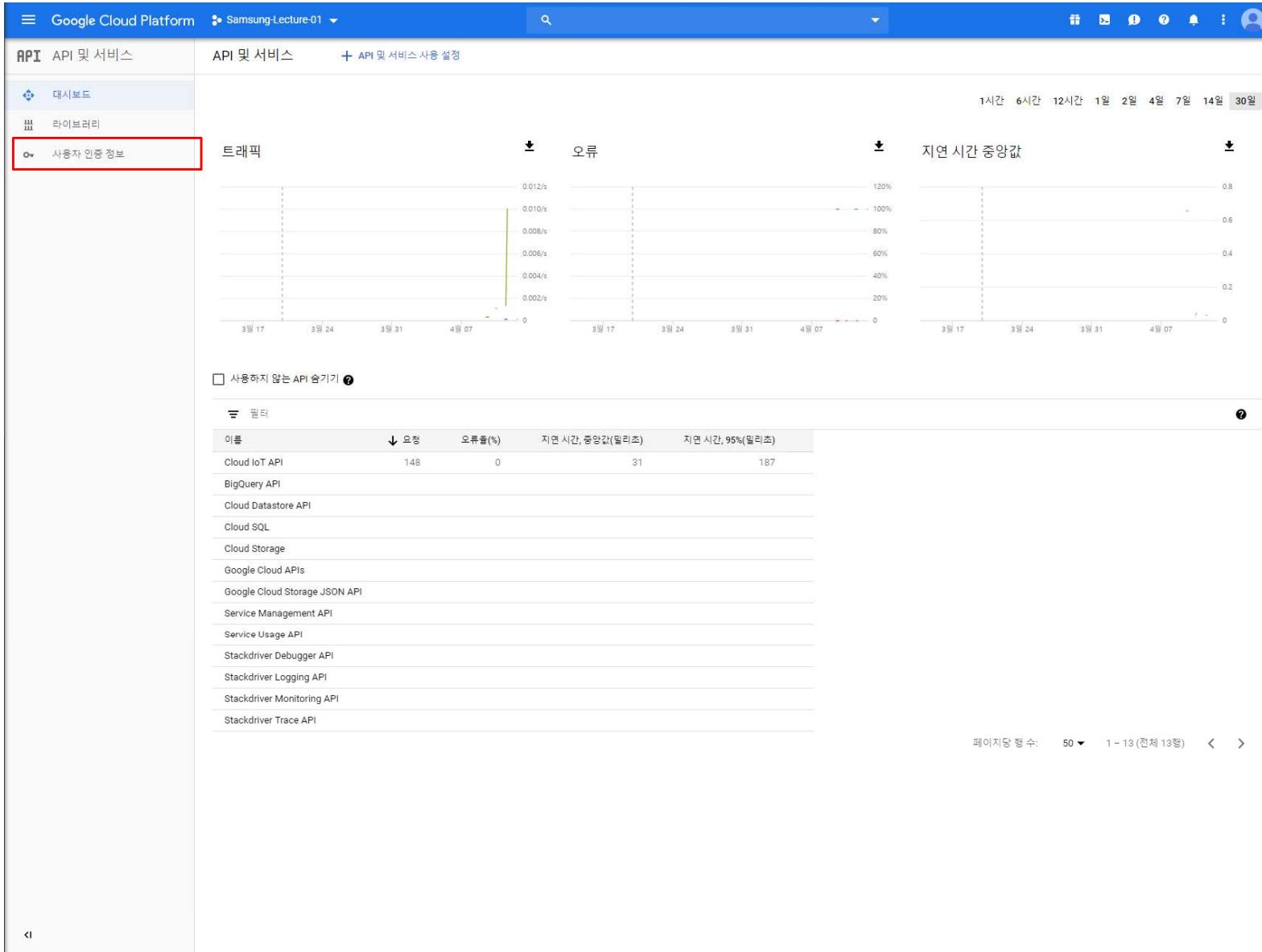
❖ 클라이언트 ID 생성 - API 및 서비스 클릭

The screenshot shows the Google Cloud Platform (GCP) dashboard for the project 'Samsung-Lecture-01'. The left sidebar is collapsed, and the main content area is divided into several sections:

- 프로젝트 정보**: Shows the project name 'Samsung-Lecture-01', project ID 'samsung-lecture-01', and project number '437995137963'. A red box highlights the 'API & Services' link in the sidebar.
- API API**: Shows the status of the API quota for the current month. It indicates a usage of 0.6 out of 1.0, with a note: '선택한 기간에는 데이터가 없습니다.' (No data available for the selected period).
- 리소스**: A section stating '이 프로젝트에는 리소스가 없습니다.'
- 추적**: A section stating '지난 7일 동안에는 추적 데이터가 없습니다.'
- 시작하기**: A list of quick start guides:
 - RPC API 탐색 및 사용 설정
 - 사전 제작된 솔루션 배포
 - 실령 중인 애플리케이션에 동적 로그 기록 추가
 - 오류 보고를 통한 오류 모니터링
 - Hello World 앱 배포
 - VM 빠르게 시작하기
 - Cloud Storage 버킷 생성
 - Cloud 할수 만들기
 - Cloud SDK 설치
- Google Cloud Platform 상태**: Shows '모든 서비스 정상' and a link to the 'Cloud 상태 대시보드로 이동'.
- 결제**: Shows '예상 요금' (USD \$0.00) for the period '2019. 4. 1. ~ 2019. 4. 12.'
- 오류 보고**: A note: '오류가 감지되지 않았습니다. Error Reporting을 설정하셨나요?' and a link to '오류 보고 설정 방법 알아보기'.
- 뉴스**: A news feed with articles like 'Day 3 at Next '19: A look back at an amazing week' (11시간 전), 'Acer introduces the Chromebook 714 and 715, built for frontline workers' (14시간 전), and 'Chrome Browser Cloud Management: Unified browser management made easy' (15시간 전). A link to '전체 뉴스 읽기' is also present.
- 문서**: Links to 'Compute Engine 알아보기', 'Cloud Storage 알아보기', and 'App Engine 알아보기'.

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 사용자 인증정보 클릭



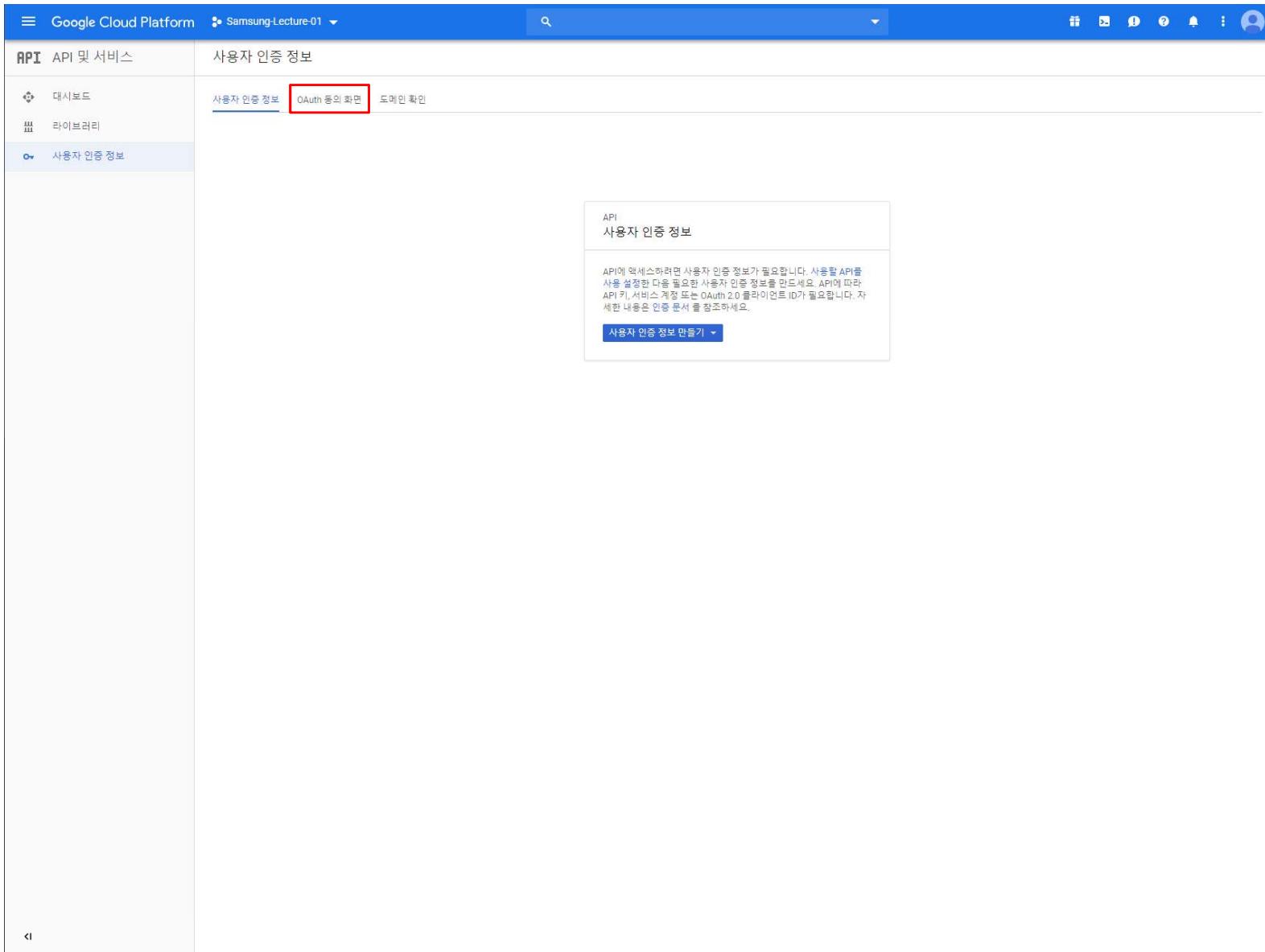
The screenshot shows the Google Cloud Platform API & Services dashboard. The left sidebar has 'API 및 서비스' selected. Under '사용자 인증 정보', there is a red box highlighting the '사용자 인증 정보' link. The main area displays three line charts: 'Traffic' (요청), 'Latency' (오류), and 'Regional Time Average' (지역 시간 중앙값). Below the charts is a checkbox for '사용하지 않는 API 숨기기'. A 'Filter' section lists various APIs with their request counts and latencies. At the bottom, there are pagination controls.

이름	요청	오류율(%)	지연 시간, 중앙값(밀리초)	지연 시간, 95%(밀리초)
Cloud IoT API	148	0	31	187
BigQuery API				
Cloud Datastore API				
Cloud SQL				
Cloud Storage				
Google Cloud APIs				
Google Cloud Storage JSON API				
Service Management API				
Service Usage API				
Stackdriver Debugger API				
Stackdriver Logging API				
Stackdriver Monitoring API				
Stackdriver Trace API				

페이지당 행 수: 50 ▾ 1 ~ 13 (전체 13행) < >

OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 생성 - OAuth 동의화면 클릭



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 어플리케이션 이름 설정 및 저장

The screenshot shows the Google Cloud Platform interface for managing APIs and services. The user is configuring an OAuth consent screen for an application named "EXLecture".

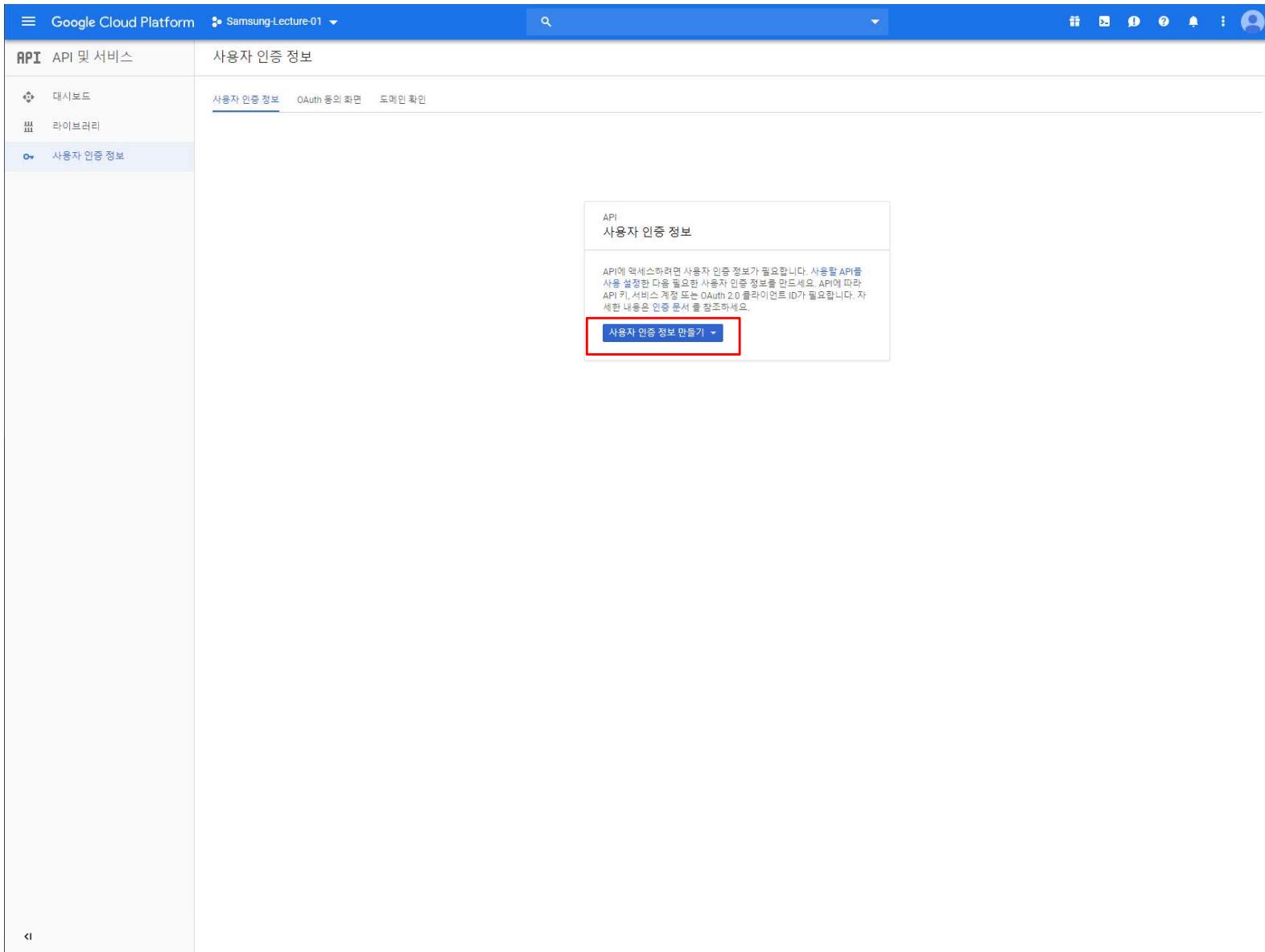
OAuth consent screen configuration:

- Application Name:** EXLecture
- Description:** 사용자는 인증 전에 이 동의 화면을 통해 비공개 데이터에 대한 액세스 권한을 부여 할지 선택할 수 있으며 서비스 약관 및 개인정보처리방침 링크도 제공됩니다. 이 페이지에서 프로젝트에 속한 모든 어플리케이션의 동의 화면을 구성합니다.
- Logo:** A placeholder image with a question mark.
- Authorized redirect URIs:** dslab.lws@gmail.com
- Scopes:** email, profile, openid
- Additional Hosted Domains:** example.com
- OAuth Scopes:** https:// 또는 http://

Save button: The "저장" (Save) button at the bottom left is highlighted with a red box.

OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 생성 - 사용자 인증 정보 만들기 클릭



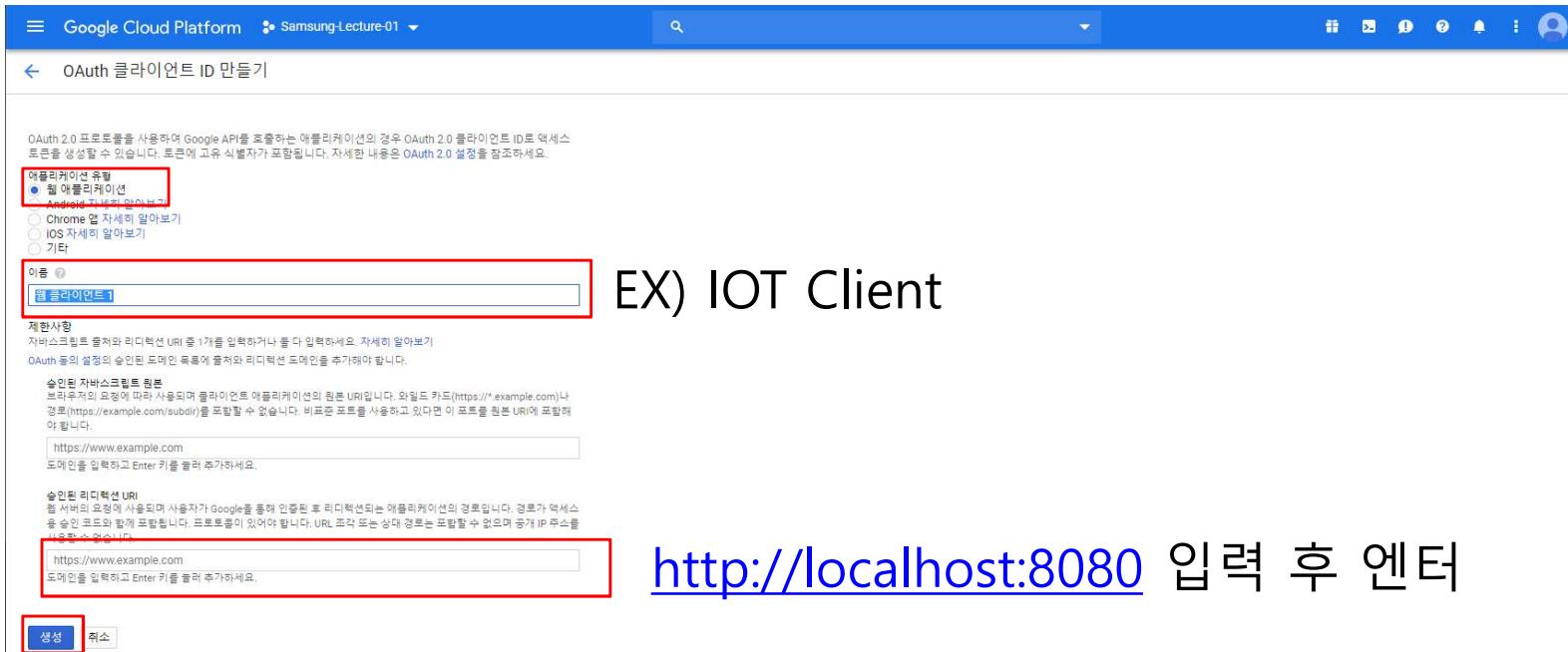
OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 생성 - 사용자 인증 정보 만들기 - OAuth 클라이언트 ID 클릭

The screenshot shows the Google Cloud Platform interface for managing APIs and services. The left sidebar is titled 'API & 서비스' (API & Services) and lists '대시보드' (Dashboard), '라이브러리' (Library), and '사용자 인증 정보' (User Authentication). The main content area is titled '사용자 인증 정보' (User Authentication) and contains three tabs: '사용자 인증 정보' (selected), 'OAuth 동의 화면' (OAuth Consent Screen), and '도메인 확인' (Domain Verification). A central modal window is open, titled 'API 사용자 인증 정보' (API User Authentication Information). It contains instructions about creating an OAuth 2.0 client ID and includes a red box highlighting the '사용자 인증 정보 만들기' (Create User Authentication Information) button. Another red box highlights the 'OAuth 클라이언트 ID' (OAuth Client ID) section, which states: '업에서 사용자 데이터에 액세스할 수 있도록 사용자 동의를 요청합니다.' (Asks for user consent to access user data). Below this are sections for 'API 키' (API Key), '서비스 계정 키' (Service Account Key), and '사용자 인증 정보 선택 도움말' (Help for selecting user authentication information).

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 클라이언트 이름 및 리디렉션 URI 설정



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 클라이언트 ID 및 클라이언트 Secret 생성확인

The screenshot shows the Google Cloud Platform interface for managing OAuth 2.0 clients. On the left, a sidebar lists 'API & Services' with 'API 및 서비스' selected. Under 'OAuth 2.0 클라이언트 ID' (Client ID), there is one entry: 'IOT Client' (Name: IOT Client, Created: 2019. 4. 12., Client ID: 437995137963-jpt8ecsnk7si3mghf2h0e58nfj4tfo.apps.googleusercontent.com). A modal window titled 'OAuth 클라이언트' (OAuth Client) is displayed, containing the Client ID and Client Secret. The Client Secret is highlighted with a red box. A red box also highlights the text 'Client Secret이 노출되지 않도록 조심해야함' (Be careful not to expose the Client Secret).

Client Secret이 노출되지 않도록 조심해야함

OAuth 2.0 Access Token 받기

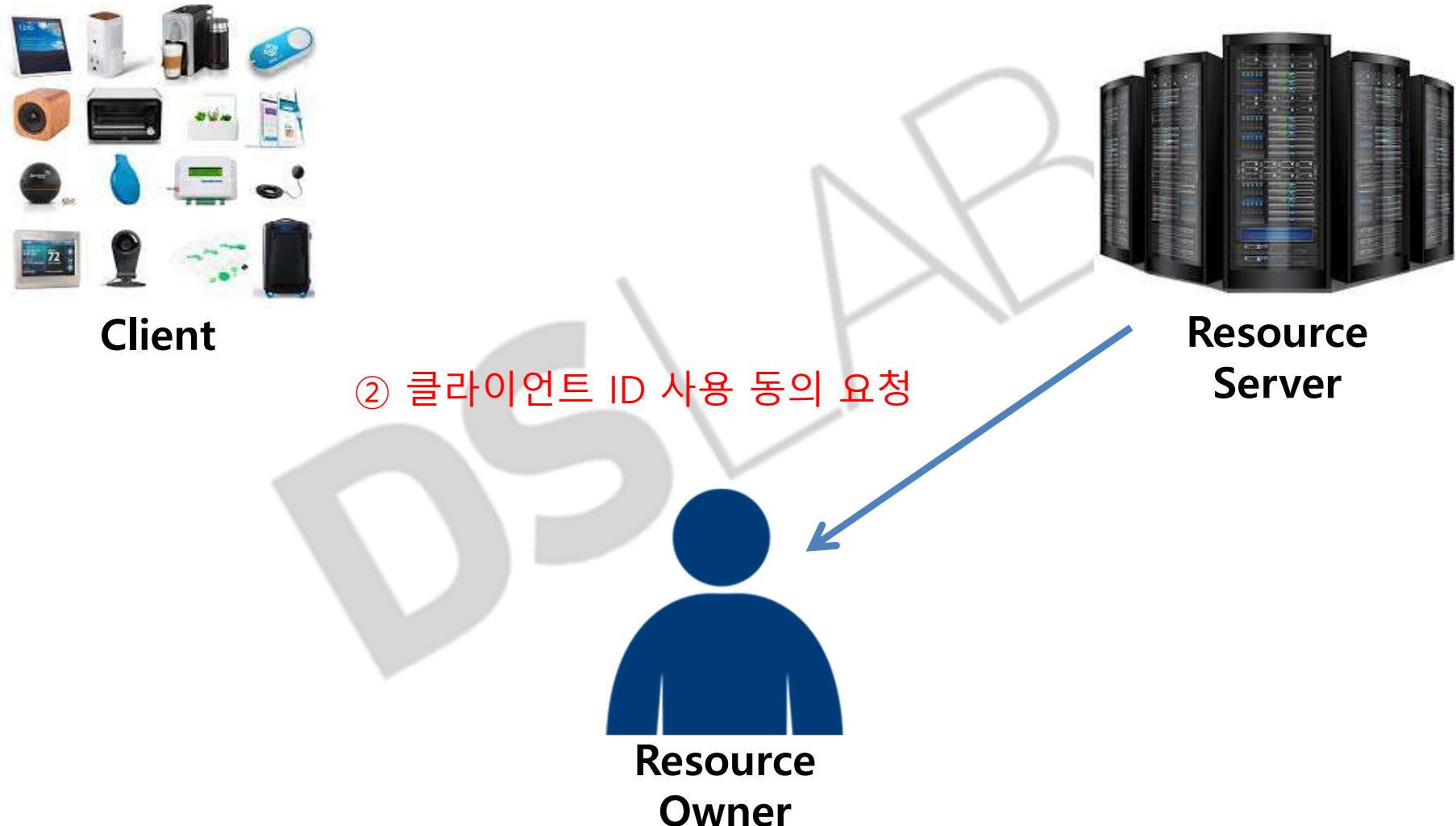
❖ 클라이언트 ID 생성 - Json파일 다운로드

The screenshot shows the Google Cloud Platform API & Services page under the 'API' tab. In the 'OAuth 2.0 클라이언트 ID' section, a client named 'IOT Client' is listed with the ID '437995137963-jpt8ecsnk7si3mghf2h0e58nij4tfo.apps.googleusercontent.com'. A red box highlights the 'Download' button next to the client ID. Below this, a file explorer window titled '다른 이름으로 저장' (Save As) is open, showing a list of files in the '내 PC > 문서' folder. A red box highlights the '파일 이름(N): client_secret_437995137963-jpt8ecsnk7si3mghf2h0e58nij4tfo.apps.googleusercontent.com.json' field. A large red box at the bottom right of the file explorer window contains the text 'Client_Secret.json으로 저장' (Save as Client_Secret.json).

Client_Secret.json으로 저장

OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 과정



OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 요청 - 동의 요청 Url 만들기

구글 계정에 인증 요청을 위한 Url
https://accounts.google.com/o/oauth2/v2/auth? 권한을 얻으려고하는 API주소
scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudiot&
access_type=offline&
include_granted_scopes=true&
state=state_parameter_passthrough_value&
redirect_uri=http%3A%2F%2Flocalhost%3A8080&
response_type=code&

client_id=**client_id**
Client ID

OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 요청 - Scope 주소란?

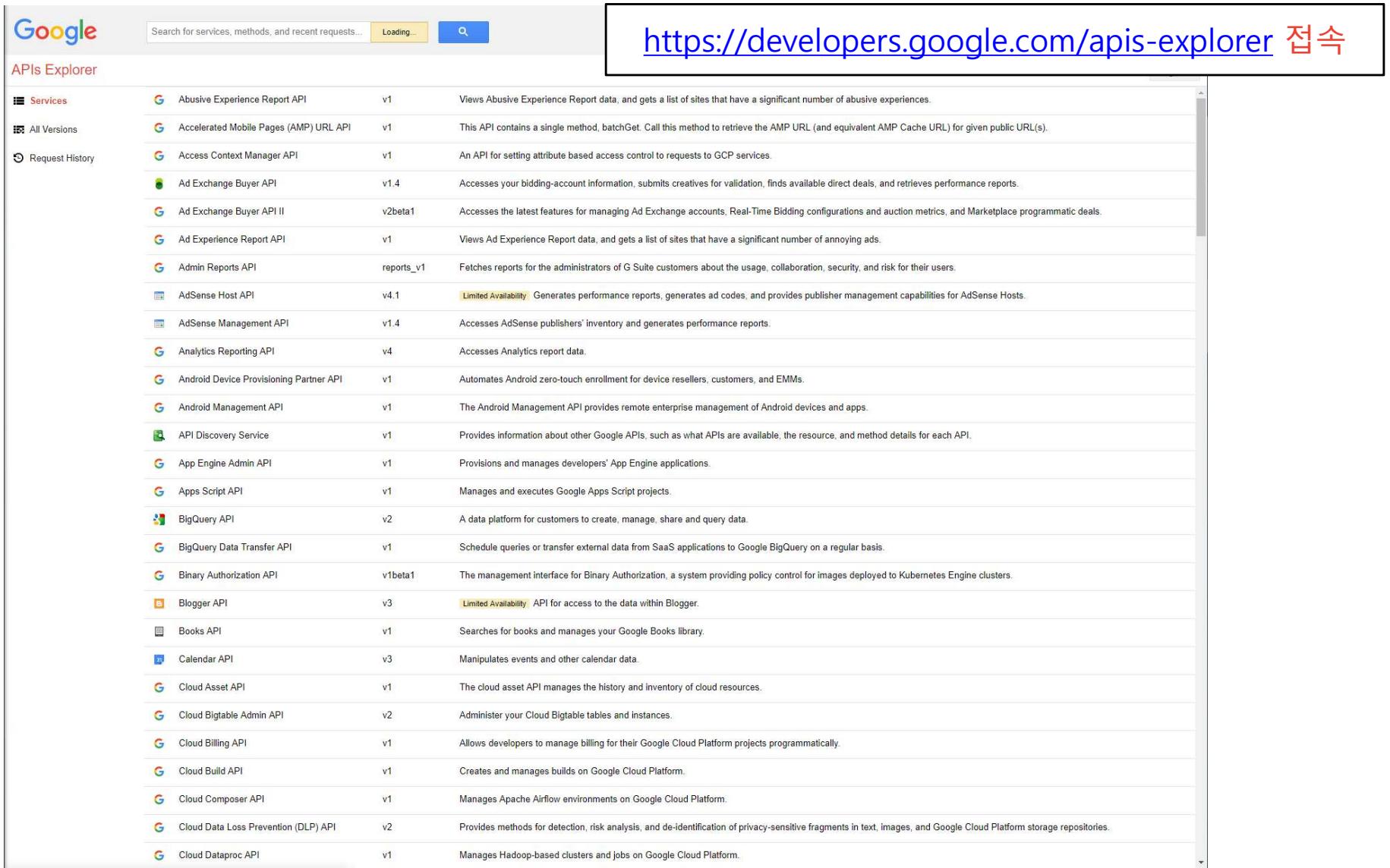
[https://accounts.google.com/o/oauth2/v2/auth?](https://accounts.google.com/o/oauth2/v2/auth?scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudiot&access_type=offline&include_granted_scopes=true&state=state_parameter_value&redirect_uri=http%3A%2F%2Flocalhost%3A8080&response_type=code&client_id=client_id)

권한을 얻으려고하는 API주소



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - API Explorer 접속



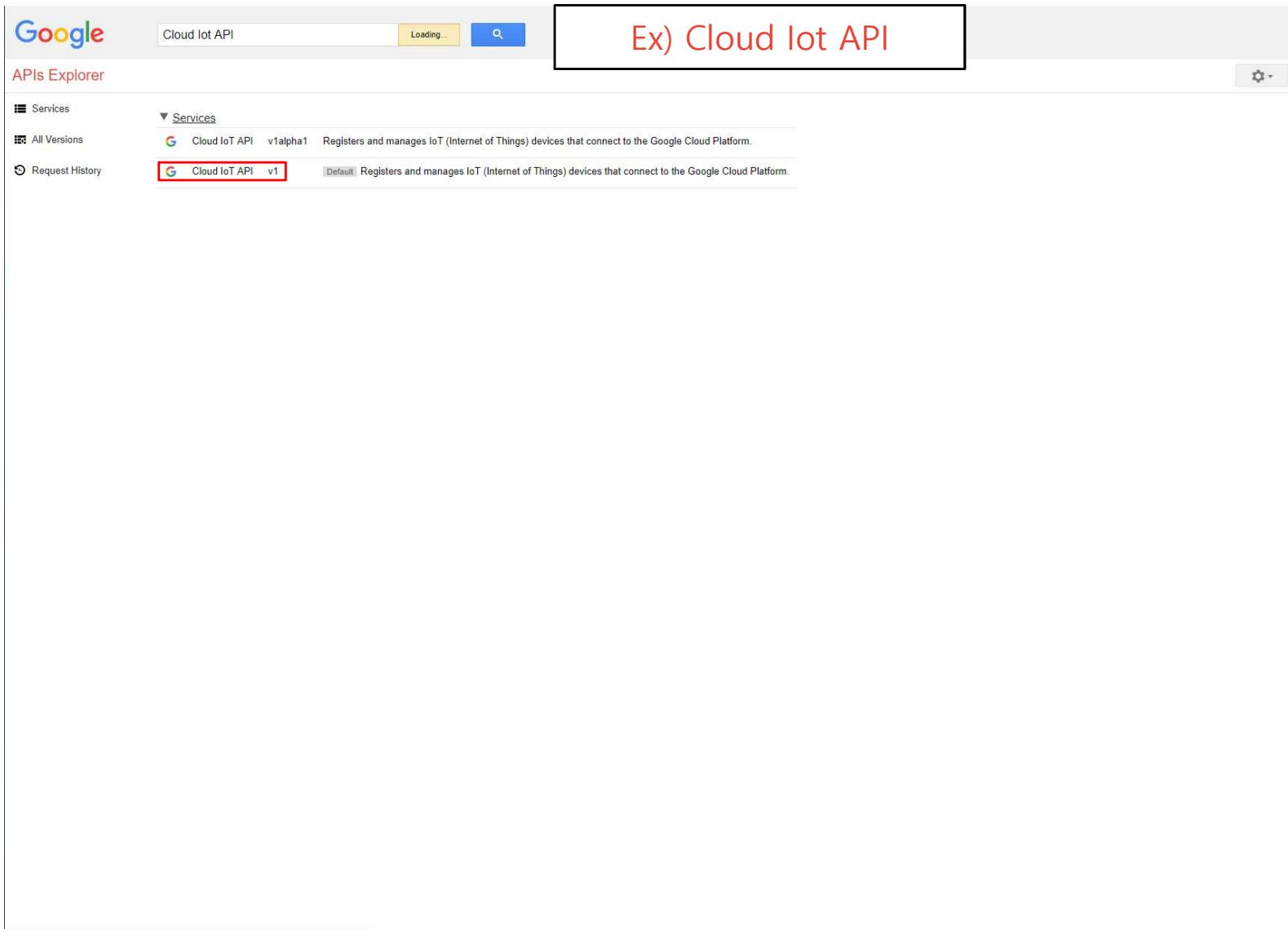
The screenshot shows the Google APIs Explorer interface. At the top, there is a search bar with placeholder text "Search for services, methods, and recent requests..." and a "Loading..." status indicator. To the right of the search bar is a magnifying glass icon. Below the search bar, the title "APIs Explorer" is displayed. On the left side, there is a sidebar with three main items: "Services" (selected), "All Versions", and "Request History". The main content area lists various Google APIs with their versions and brief descriptions. A specific API entry, "Cloud Data Loss Prevention (DLP) API", is highlighted with a red rectangular box around its URL. The URL is also displayed prominently in red text on the right side of the screenshot.

[https://developers.google.com/apis-explorer 접속](https://developers.google.com/apis-explorer)

API	Version	Description
Abusive Experience Report API	v1	Views Abusive Experience Report data, and gets a list of sites that have a significant number of abusive experiences.
Accelerated Mobile Pages (AMP) URL API	v1	This API contains a single method, batchGet. Call this method to retrieve the AMP URL (and equivalent AMP Cache URL) for given public URL(s).
Access Context Manager API	v1	An API for setting attribute based access control to requests to GCP services.
Ad Exchange Buyer API	v1.4	Accesses your bidding-account information, submits creatives for validation, finds available direct deals, and retrieves performance reports.
Ad Exchange Buyer API II	v2beta1	Accesses the latest features for managing Ad Exchange accounts, Real-Time Bidding configurations and auction metrics, and Marketplace programmatic deals.
Ad Experience Report API	v1	Views Ad Experience Report data, and gets a list of sites that have a significant number of annoying ads.
Admin Reports API	reports_v1	Fetches reports for the administrators of G Suite customers about the usage, collaboration, security, and risk for their users.
AdSense Host API	v4.1	Limited Availability Generates performance reports, generates ad codes, and provides publisher management capabilities for AdSense Hosts.
AdSense Management API	v1.4	Accesses AdSense publishers' inventory and generates performance reports.
Analytics Reporting API	v4	Accesses Analytics report data.
Android Device Provisioning Partner API	v1	Automates Android zero-touch enrollment for device resellers, customers, and EMMs.
Android Management API	v1	The Android Management API provides remote enterprise management of Android devices and apps.
API Discovery Service	v1	Provides information about other Google APIs, such as what APIs are available, the resource, and method details for each API.
App Engine Admin API	v1	Provisions and manages developers' App Engine applications.
Apps Script API	v1	Manages and executes Google Apps Script projects.
BigQuery API	v2	A data platform for customers to create, manage, share and query data.
BigQuery Data Transfer API	v1	Schedule queries or transfer external data from SaaS applications to Google BigQuery on a regular basis.
Binary Authorization API	v1beta1	The management interface for Binary Authorization, a system providing policy control for images deployed to Kubernetes Engine clusters.
Blogger API	v3	Limited Availability API for access to the data within Blogger.
Books API	v1	Searches for books and manages your Google Books library.
Calendar API	v3	Manipulates events and other calendar data.
Cloud Asset API	v1	The cloud asset API manages the history and inventory of cloud resources.
Cloud Bigtable Admin API	v2	Administer your Cloud Bigtable tables and instances.
Cloud Billing API	v1	Allows developers to manage billing for their Google Cloud Platform projects programmatically.
Cloud Build API	v1	Creates and manages builds on Google Cloud Platform.
Cloud Composer API	v1	Manages Apache Airflow environments on Google Cloud Platform.
Cloud Data Loss Prevention (DLP) API	v2	Provides methods for detection, risk analysis, and de-identification of privacy-sensitive fragments in text, images, and Google Cloud Platform storage repositories.
Cloud Dataproc API	v1	Manages Hadoop-based clusters and jobs on Google Cloud Platform.

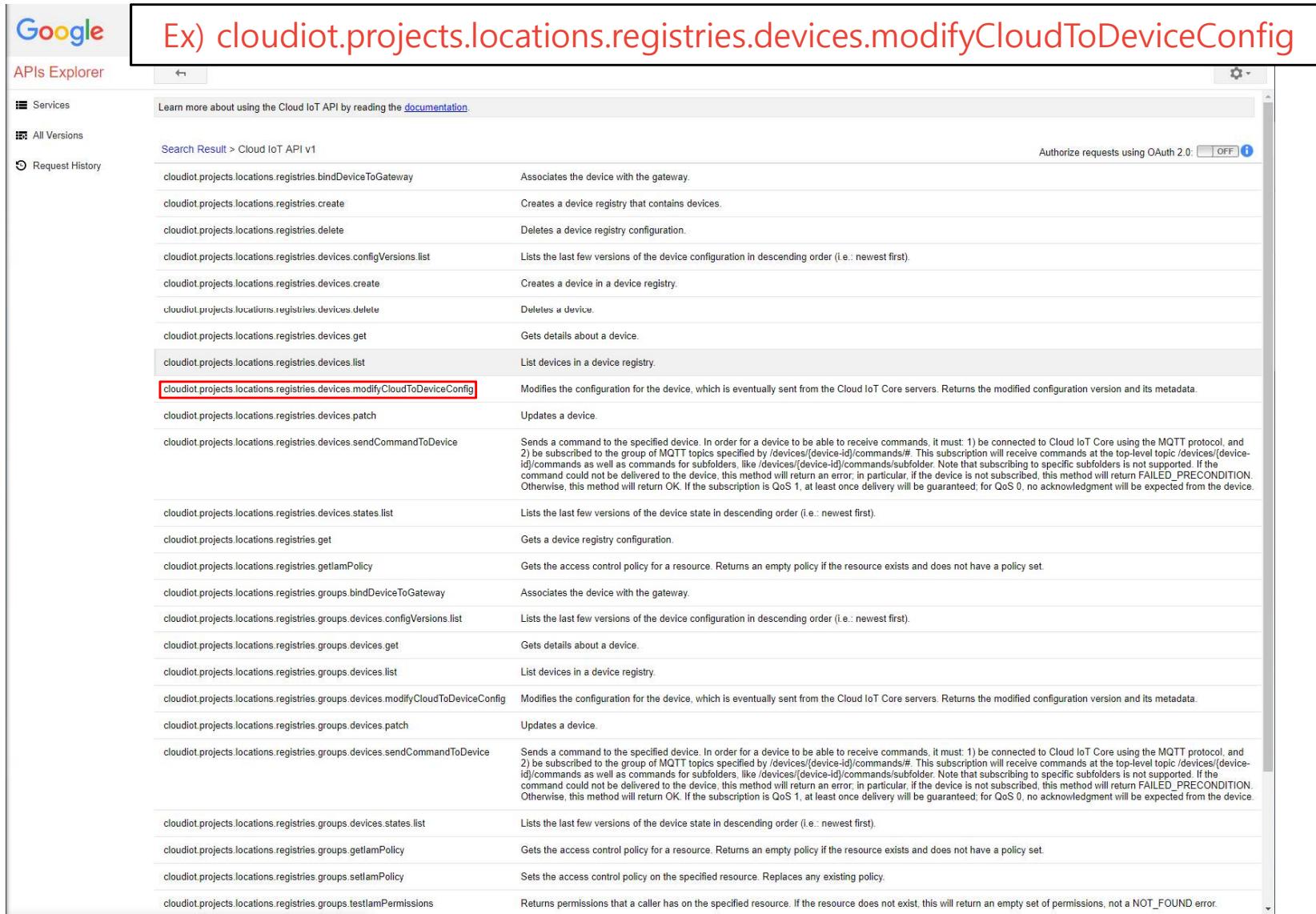
OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 요청 - 사용할 API 검색



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - 사용할 API 선택

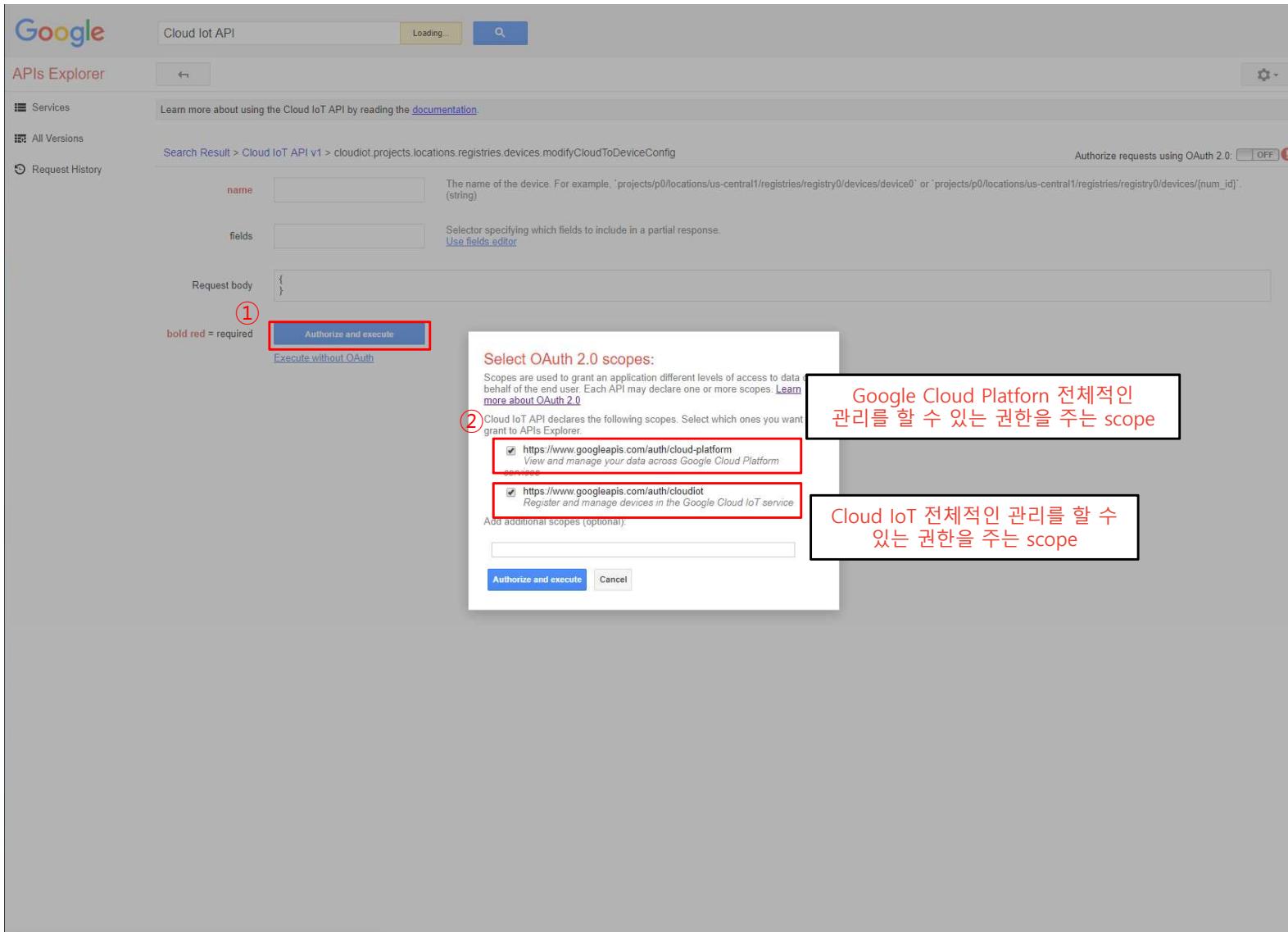


Ex) `clouidot.projects.locations.registries.devices.modifyCloudToDeviceConfig`

The screenshot shows the Google APIs Explorer interface for the Cloud IoT API v1. The search results list various API endpoints. One endpoint, `clouidot.projects.locations.registries.devices.modifyCloudToDeviceConfig`, is highlighted with a red box. The description for this endpoint states: "Modifies the configuration for the device, which is eventually sent from the Cloud IoT Core servers. Returns the modified configuration version and its metadata." Other endpoints listed include `list`, `get`, `patch`, `sendCommandToDevice`, `states.list`, `getIamPolicy`, `bindDeviceToGateway`, `configVersions.list`, `devices.get`, `groups.list`, `modifyCloudToDeviceConfig`, `patch`, `sendCommandToDevice`, `states.list`, `getIamPolicy`, `setIamPolicy`, and `testIamPermissions`.

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - 사용할 Scope 주소 복사



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - Scope 주소 인코딩

The screenshot shows a web-based URL Decoder/Encoder tool interface. At the top left, there is a red box labeled ① containing the URL <https://www.googleapis.com/auth/cloudiot>. To the right of the URL input field is a red box labeled ② containing the text "Url 인코딩사이트(<https://meyerweb.com/eric/tools/dencoder/>) 접속". Below the URL input field, there is a red box labeled "사용할 scope 주소 붙여넣기". In the bottom left corner of the tool's interface, there is another red box labeled ② containing the word "Encoder". The tool has a "Decode" button and an "Encoder" button, with the "Encoder" button being highlighted. Below the buttons, there is a list of instructions:

- Input a string of text and encode or decode it as you like.
- Handy for turning encoded JavaScript URLs from complete gibberish into readable gibberish.
- If you'd like to have the URL Decoder/Encoder for offline use, just view source and save to your hard drive.

At the bottom of the tool's interface, there is a note: "The URL Decoder/Encoder is licensed under a Creative Commons Attribution-ShareAlike 2.0 License." and a Creative Commons license logo.

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - 인코딩된 scope 주소 입력

URL Decoder/Encoder

`https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudiot`

Decode Encode

- Input a string of text and encode or decode it as you like.
- Handy for turning encoded JavaScript URLs from complete gibberish into readable gibberish.
- If you'd like to have the URL Decoder/Encoder for offline use, just view source and save to your hard drive.

The URL Decoder/Encoder is licensed under a Creative Commons Attribution-ShareAlike 2.0 License.

This tool is provided without warranty, guarantee, or much in the way of explanation. Note that use of this tool may or may not crash your browser, lock up your machine, erase your hard drive, or e-mail those naughty pictures you hid in the Utilities folder to your mother. Don't blame me if anything bad happens to you, because it's actually the aliens' fault. The code expressed herein is solely that of the author, and he's none too swift with the JavaScript, if you know what we mean, so it's likely to cause giggles fits in anyone who knows what they're doing. Not a flying toy. Thank you for playing. Insert coin to continue.

SOME RIGHTS RESERVED
Creative Commons

scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudiot

OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 요청 - Redirect URI

<https://accounts.google.com/o/oauth2/v2/auth?>

scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudiot&

access_type=offline&

include_granted_scopes=true&

state=state_parameter_value&

redirect_uri=http%3A%2F%2Flocalhost%3A8080&

response_type=code&

client_id=client_id

Code를 Redirect 받을 주소(Client ID를 만들때 입력한 주소)

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - Redirect URI 인코딩

URL Decoder/Encoder

http://localhost:8080

CODE값을 받아올 서버의 URI를 입력

http://localhost:8080 현재 사용자 PC의 8080번 포트 서버

Decode Encode

- Input a string of text and encode or decode it as you like.
- Handy for turning encoded JavaScript URLs from complete gibberish into readable gibberish.
- If you'd like to have the URL Decoder/Encoder for offline use, just view source and save to your hard drive.

The URL Decoder/Encoder is licensed under a Creative Commons Attribution-ShareAlike 2.0 License.

This tool is provided without warranty, guarantee, or much in the way of explanation. Note that use of this tool may or may not crash your browser, lock up your machine, erase your hard drive, or e-mail those naughty pictures you hid in the Utilities folder to your mother. Don't blame me if anything bad happens to you, because it's actually the aliens' fault. The code expressed herein is solely that of the author, and he's none too swift with the JavaScript, if you know what we mean, so it's likely to cause giggles fits in anyone who knows what they're doing. Not a flying toy. Thank you for playing. Insert coin to continue.

※ Client ID 생성시 승인 URI 목록에 추가해야함
※ 존재하지 않는 서버를 입력할 경우 연결이 되지 않지만 코드값은 URL에서 받아올 수 있음

redirect_uri=http%3A%2F%2Flocalhost%3A8080

OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 요청 - 예시

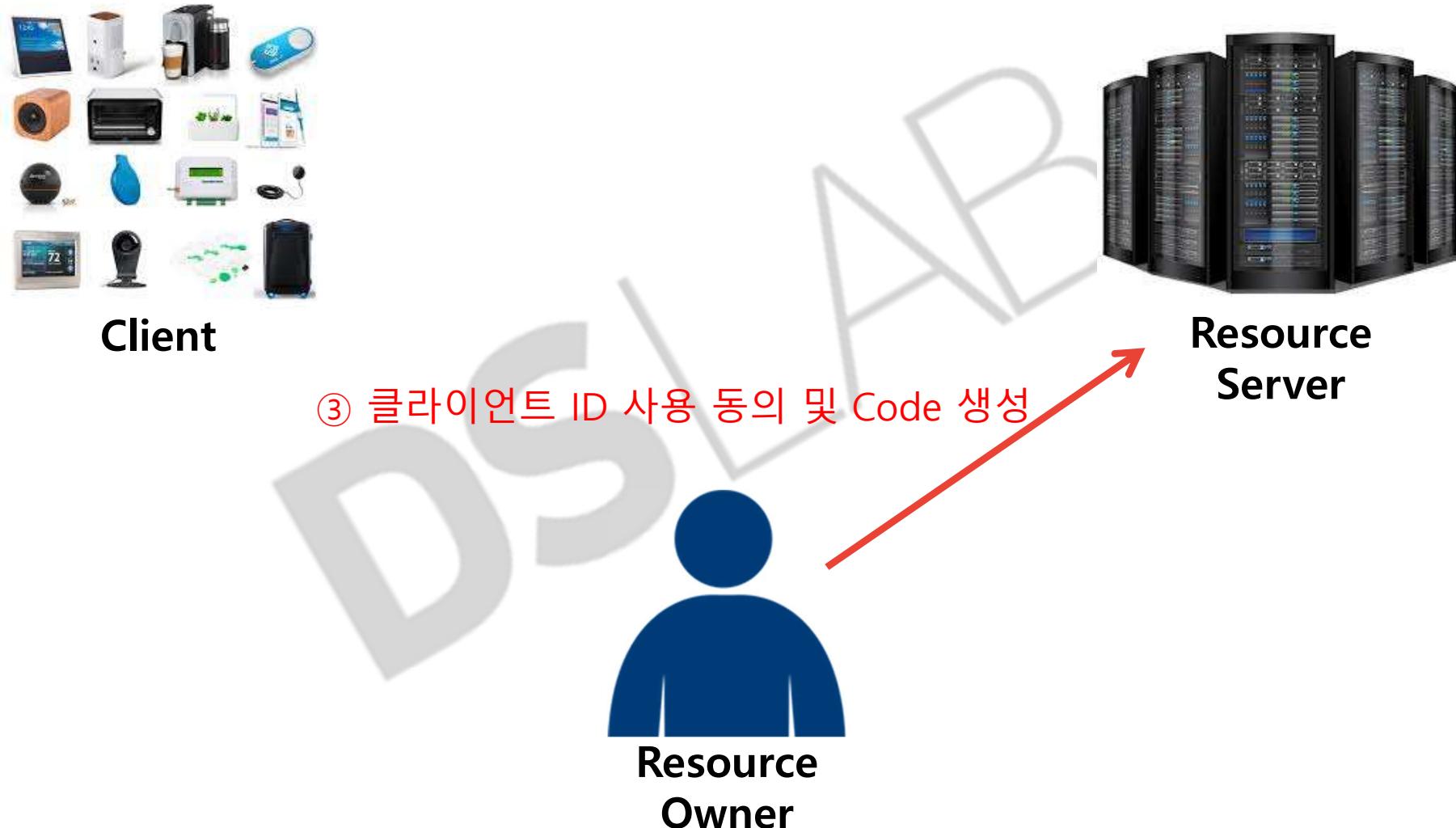
https://accounts.google.com/o/oauth2/v2/auth?scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudiot&access_type=offline&include_granted_scopes=true&state=state_parameter_passthrough_value&redirect_uri=http%3A%2F%2Flocalhost%3A8080&response_type=code&client_id=클라이언트 ID

EX)

https://accounts.google.com/o/oauth2/v2/auth?scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudiot&access_type=offline&include_granted_scopes=true&state=state_parameter_passthrough_value&redirect_uri=http%3A%2F%2Flocalhost%3A8080&response_type=code&client_id=437995137963-jpt8ecsnsk7sl3mghf2h0e58nfjl4tfo.apps.googleusercontent.com

OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 과정



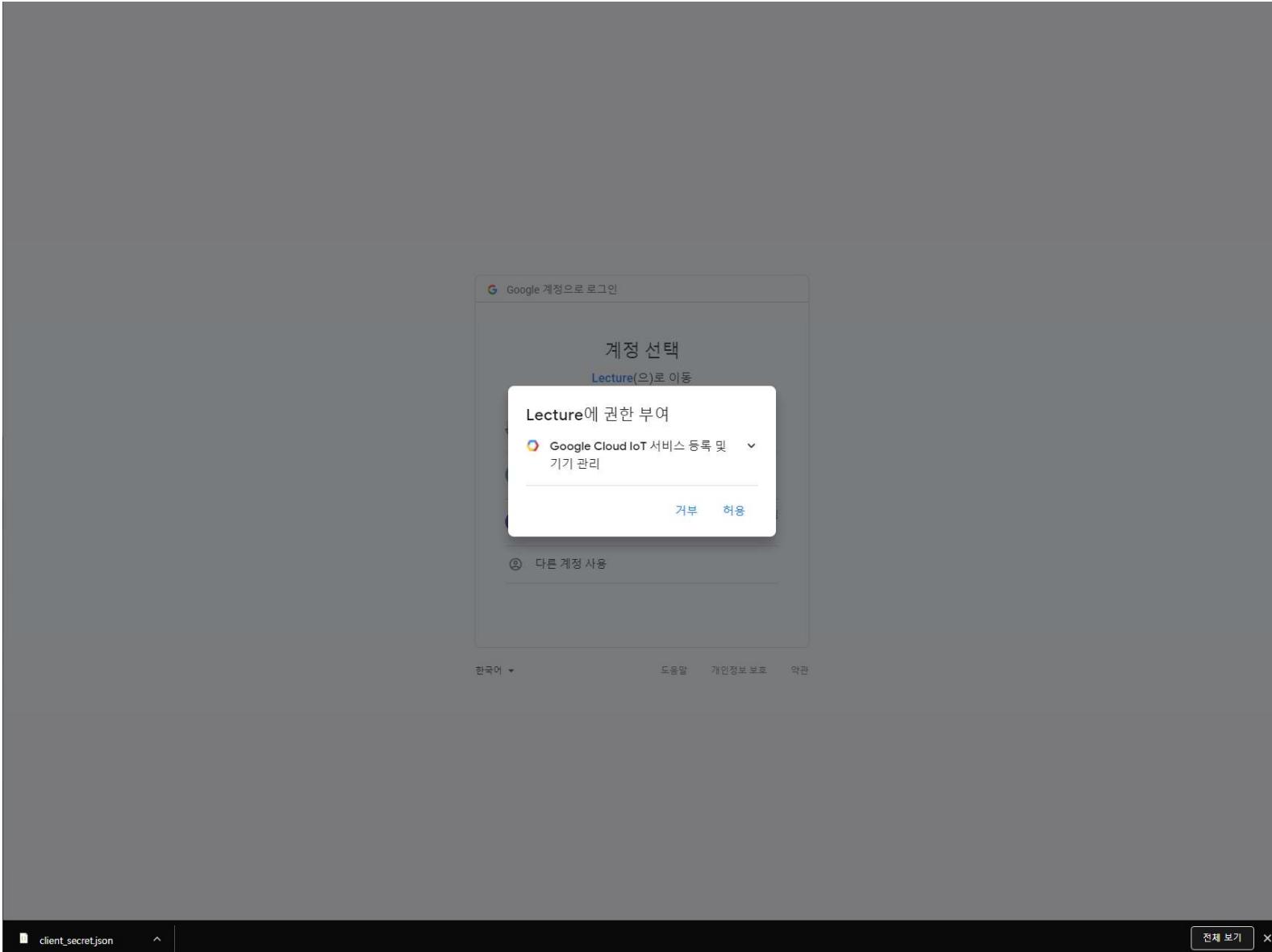
OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 및 Code 생성 - ID 및 비밀번호 입력



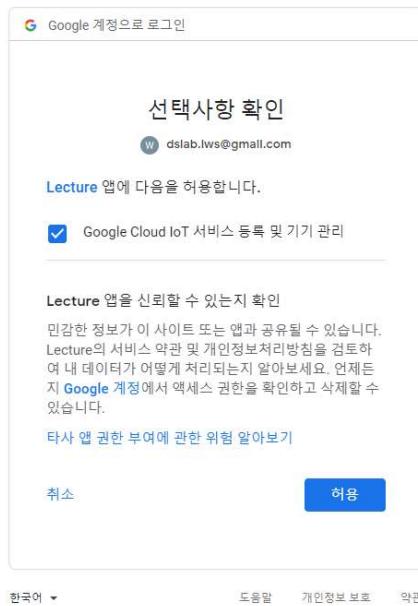
OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 및 Code 생성 - 허용



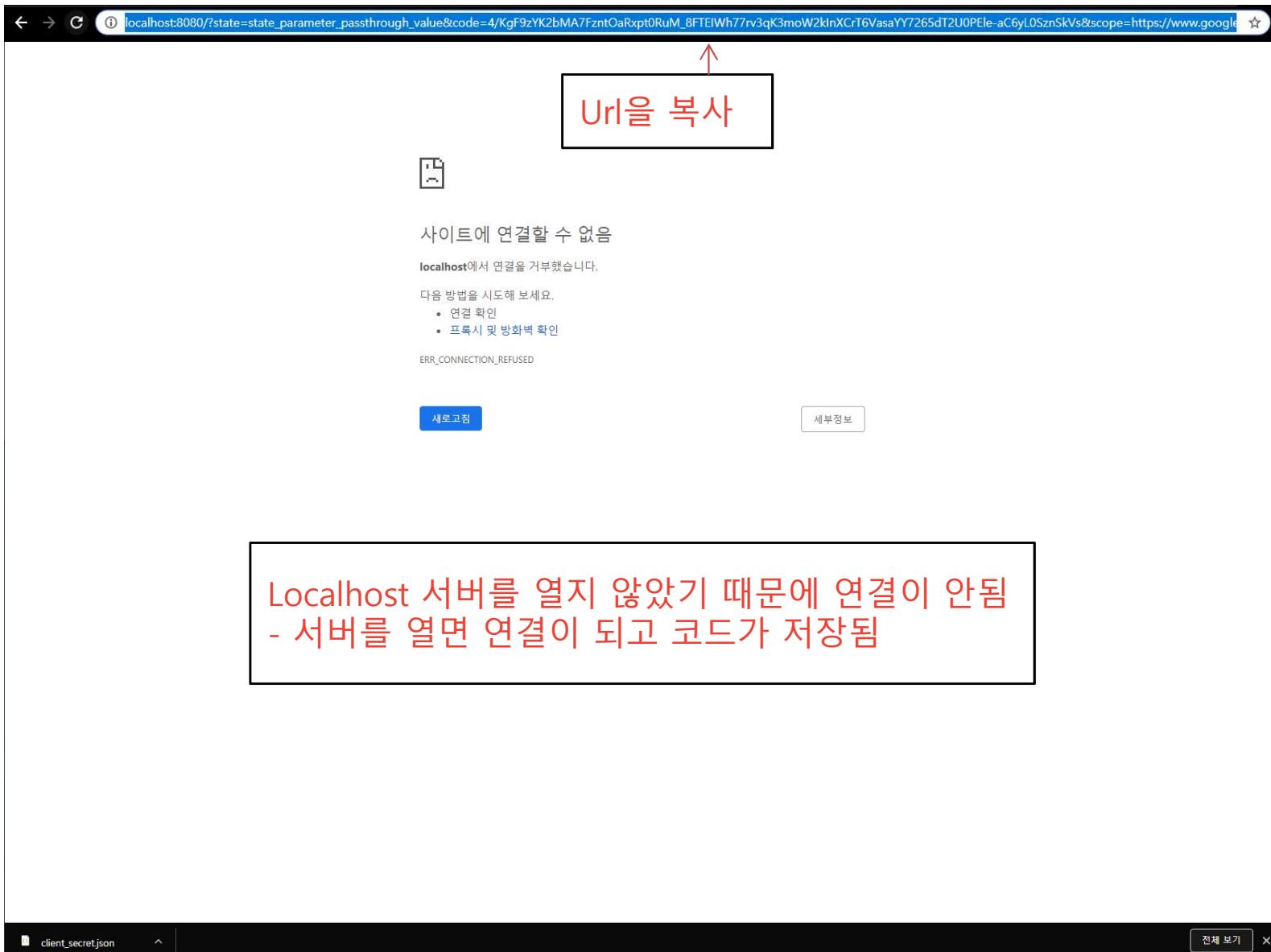
OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 및 Code 생성 - 허용



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 및 Code 생성 - code 생성



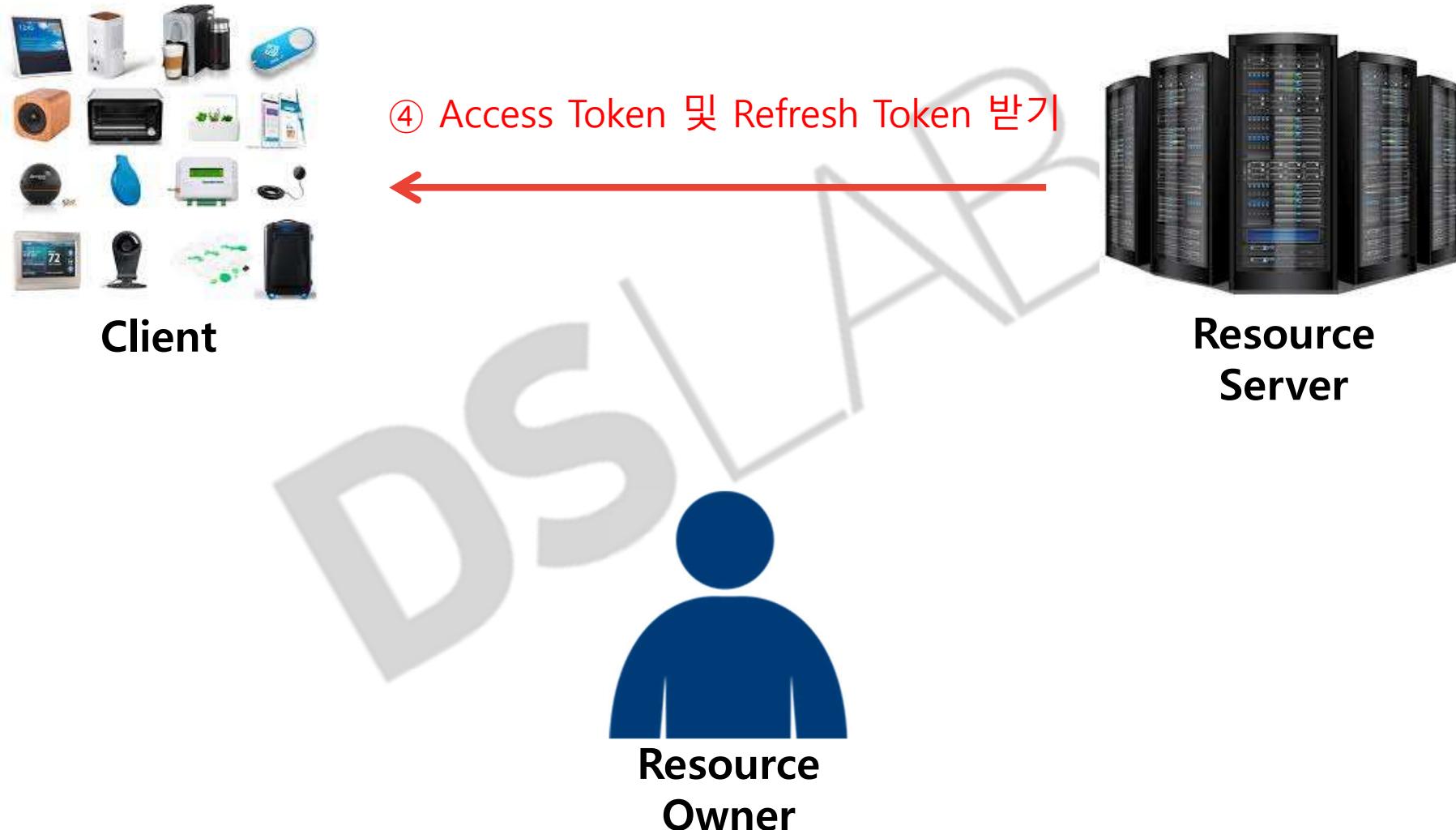
OAuth 2.0 Access Token 받기

- ❖ 클라이언트 ID 사용 동의 및 Code 생성 - Url에 들어있는 코드값 가져오기

http://localhost:8080/?state=state_parameter_passthrough_value&code=4%2FLQGcqlBaUa-4DZHlzFFgt3eAQ11_Z9UEIPfZj7dQ-N_7WrOOpHjLBDgXBDnwERe4TgWoY9yvDx3NS4yUKEfjZ4s&scope=https://www.googleapis.com/auth/cloudiot

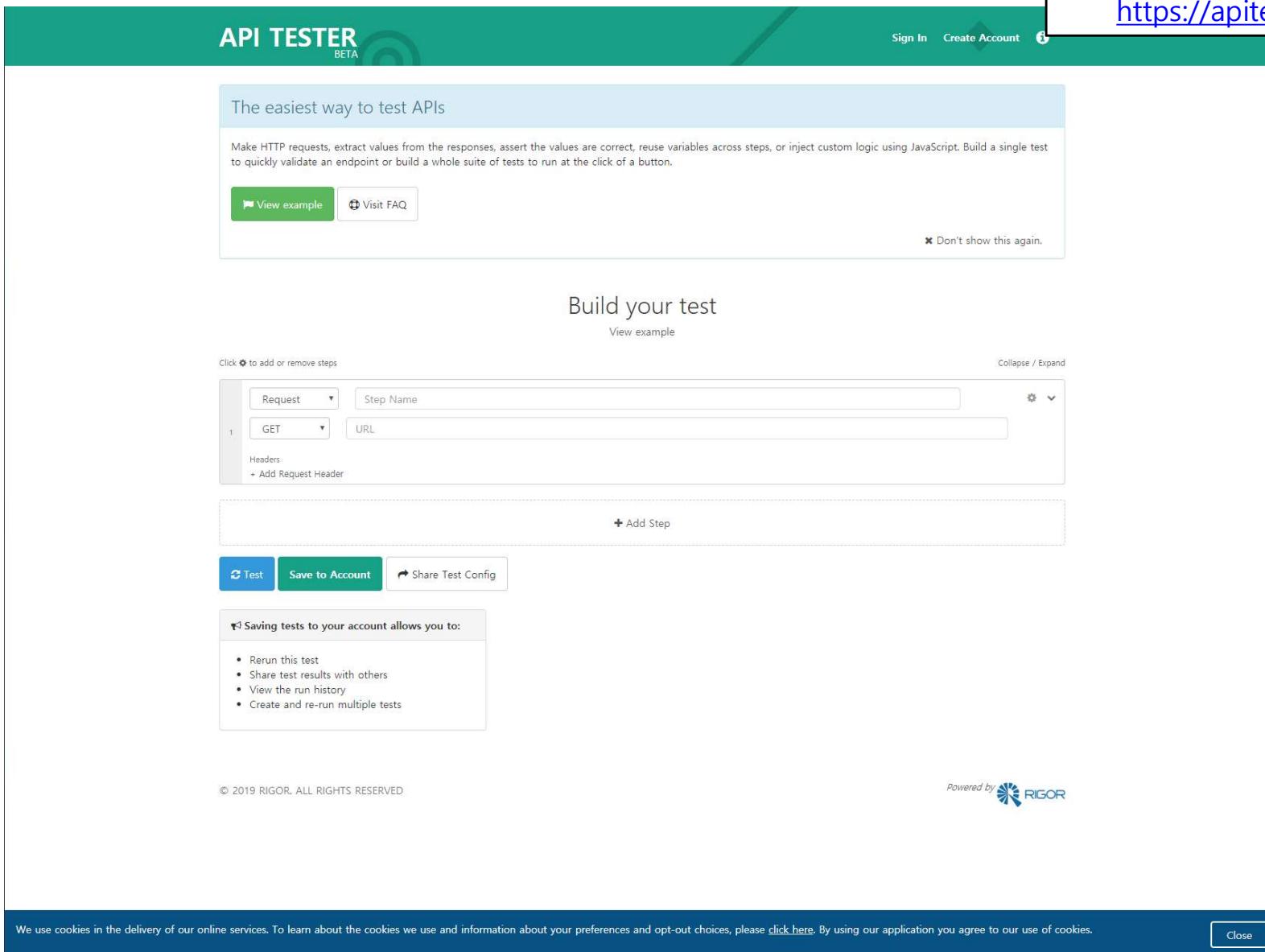
OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 과정



OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - API Tester 접속



The screenshot shows the API Tester BETA interface. At the top, there's a navigation bar with 'API TESTER BETA' on the left, 'Sign In' and 'Create Account' on the right, and a URL field containing <https://apitester.com/>. Below the navigation is a banner with the text 'The easiest way to test APIs' and a subtext about making HTTP requests and validating endpoints. It includes 'View example' and 'Visit FAQ' buttons, and a 'Don't show this again.' link.

The main area is titled 'Build your test'. It features a 'Request' configuration panel with dropdowns for 'Method' (GET) and 'URL', and a 'Headers' section with a '+ Add Request Header' button. Below this is a large dashed box labeled '+ Add Step'. At the bottom of this panel are 'Test', 'Save to Account', and 'Share Test Config' buttons. A note above the buttons says 'Saving tests to your account allows you to:' followed by a list: 'Rerun this test', 'Share test results with others', 'View the run history', and 'Create and re-run multiple tests'. The footer contains copyright information ('© 2019 RIGOR. ALL RIGHTS RESERVED'), a 'Powered by RIGOR' logo, and a cookie policy notice at the bottom.

OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - Access Token 요청

The screenshot shows the API Tester BETA interface with the following details:

- Request Type:** POST (highlighted with a red box)
- Endpoint:** https://accounts.google.com/o/oauth2/token (highlighted with a red box)
- Post Data:** code=4%2FLQ9QH5v_Rz1Nf8kRik10dSykYY5JnDzIE43gI5g7vs0XTksawXbiBBAt0Zz4SP1DR7HARH4UxGEYn0xfM5df0&client_id=437995137963-jpt0ecsnk7s13mghf2h0e58nfj14tfo.apps.googleusercontent.com&client_secret=[REDACTED]&redirect_uri=http://localhost:8080&grant_type=authorization_code (highlighted with a red box)
- Headers:** Content-Type: application/x-www-form-urlencoded (highlighted with a red box)
- Buttons:** Test (highlighted with a red box), Save to Account, Share Test
- Text Labels:** code=복사한 코드&client_id=부여된 client ID&client_secret= client secret&redirect_uri=http://localhost:8080&grant_type=authorizatio n_code (highlighted with a red box)
- Bottom Text:** Saving tests to your account allows you to:
 - Rerun this test
 - Share test results with others
 - View the run history
 - Create and re-run multiple tests
- Bottom Footer:** We use cookies in the delivery of our online services. To learn about the cookies we use and information about your preferences and opt-out choices, please [click here](#). By using our application you agree to our use of cookies.

OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - Refresh Token 복사

성공시 화면

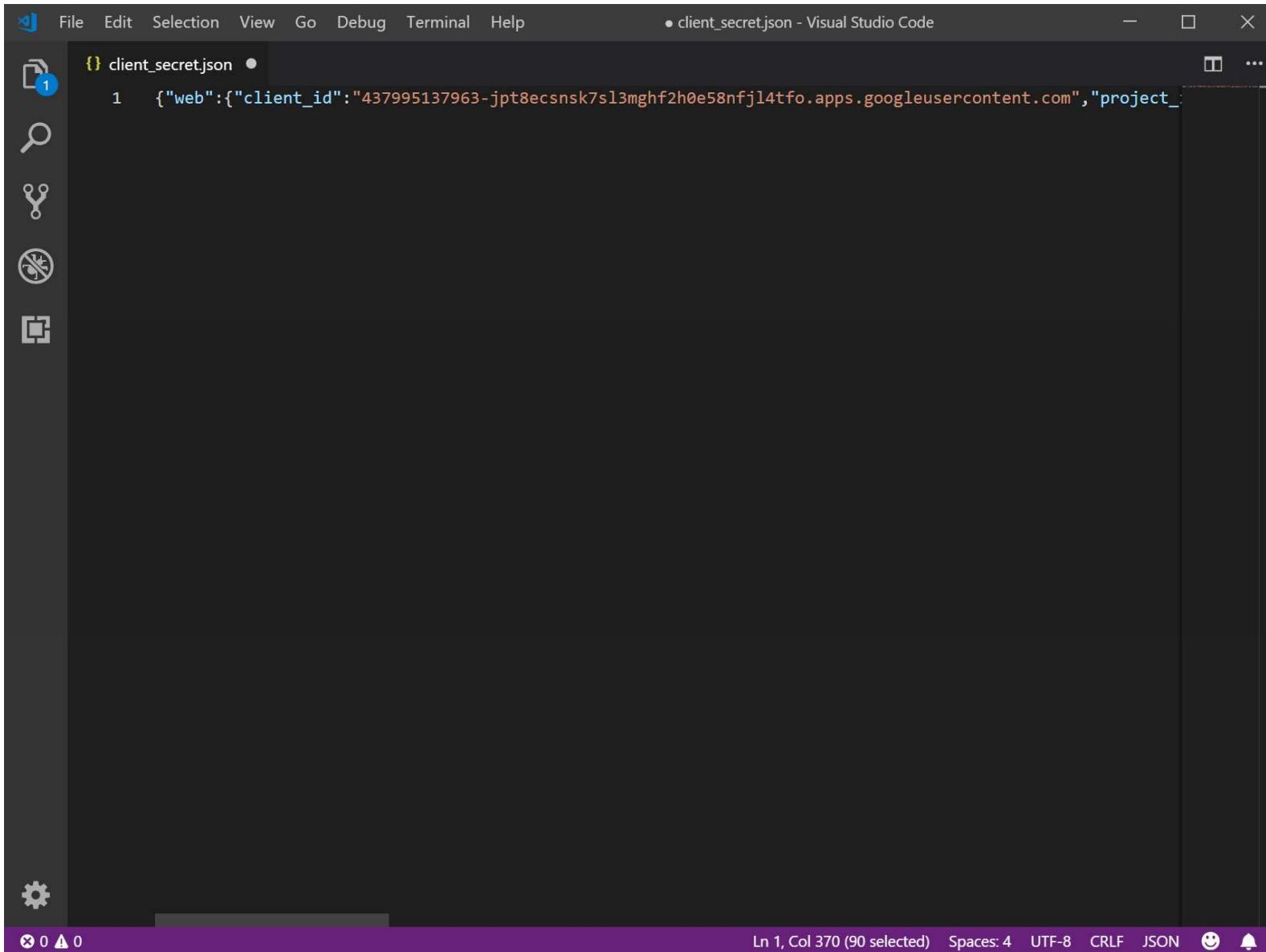
```
[{"access_token": "ya29.Glv..._xUTdCL75S9am0VFbKF9", "expires_in": 3600, "refresh_token": "...", "scope": "https://www.googleapis.com/auth/cloudiot", "token_type": "Bearer"}]
```

We use cookies in the delivery of our onl... Close

Name	Before	After
{{headers.Accept-Ranges}}	changed	--
{{headers.Alt-Svc}}	changed	--

OAuth 2.0 Access Token 받기

- ❖ Access Token 및 Refresh Token 받기 - client_secret.json 파일 열기



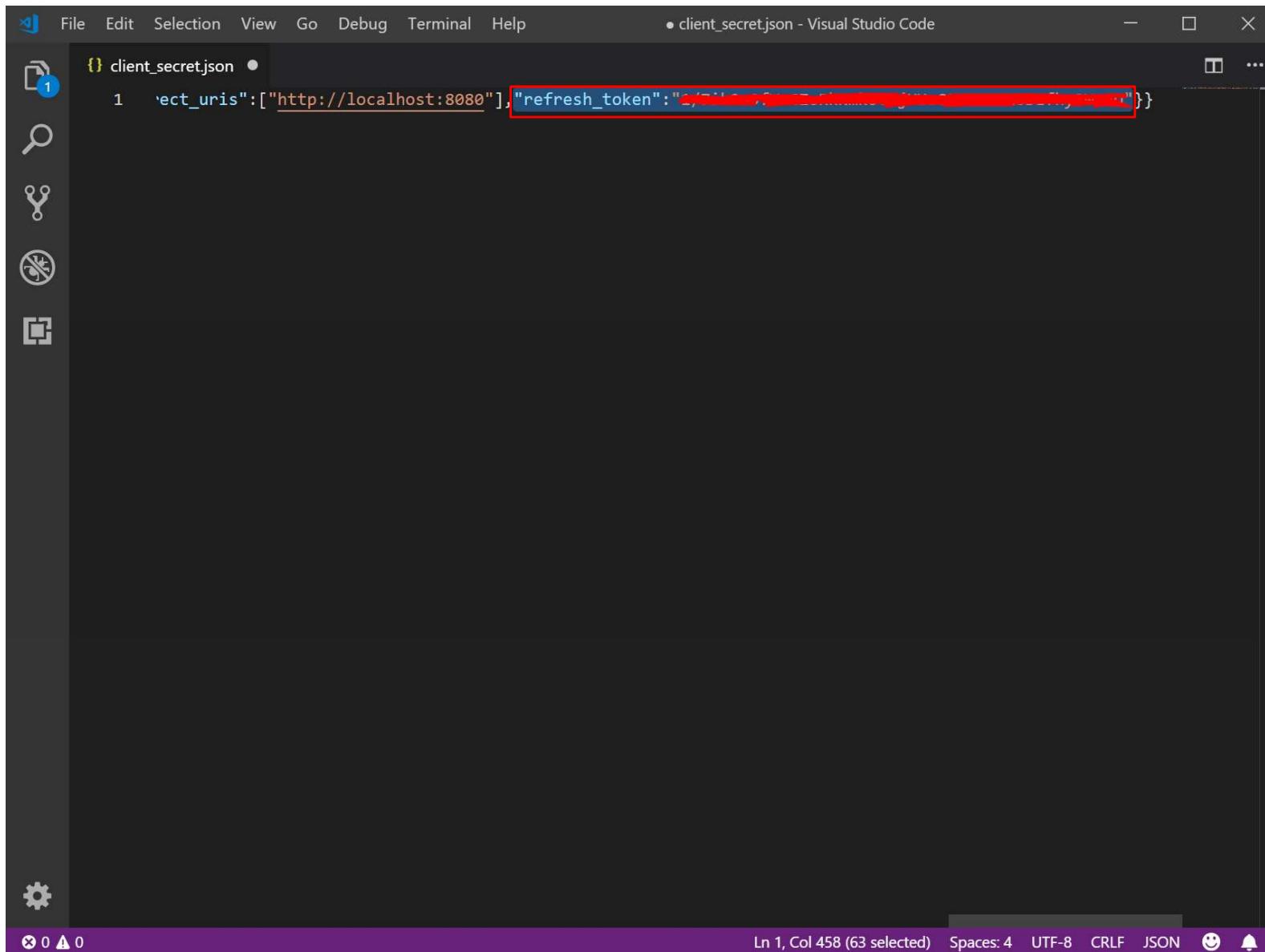
A screenshot of the Visual Studio Code interface. The title bar shows "client_secret.json - Visual Studio Code". The left sidebar has a file icon with a blue circle containing the number "1", indicating one file is open. The main editor area displays the following JSON code:

```
{ "web": { "client_id": "437995137963-jpt8ecsnk7sl3mghf2h0e58nfjl4tfo.apps.googleusercontent.com", "project_id": "437995137963", "auth_uri": "https://accounts.google.com/o/oauth2/auth", "token_uri": "https://oauth2.googleapis.com/token", "client_secret": "GOkDyLJXWzvPQHgjwCqUOOGV", "redirect_uris": [ "http://localhost:8080/callback" ] } }
```

The status bar at the bottom shows "Ln 1, Col 370 (90 selected) Spaces: 4 UTF-8 CRLF JSON". There are also icons for a smiley face and a bell.

OAuth 2.0 Access Token 받기

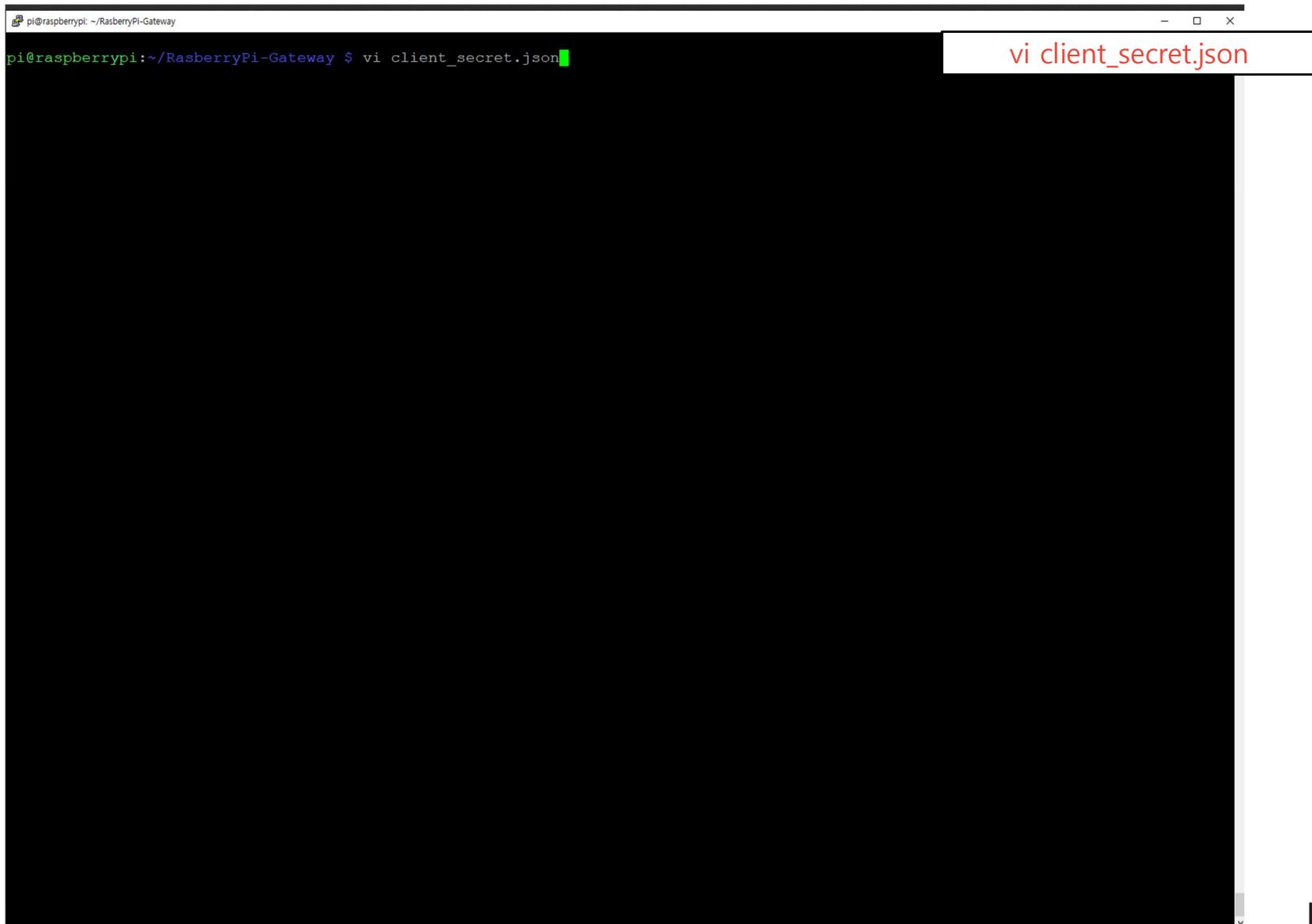
❖ Access Token 및 Refresh Token 받기 - Refresh Token 붙여넣기



```
{ "client_uris": ["http://localhost:8080"], "refresh_token": "...REDACTED..." }
```

OAuth 2.0 Access Token 받기

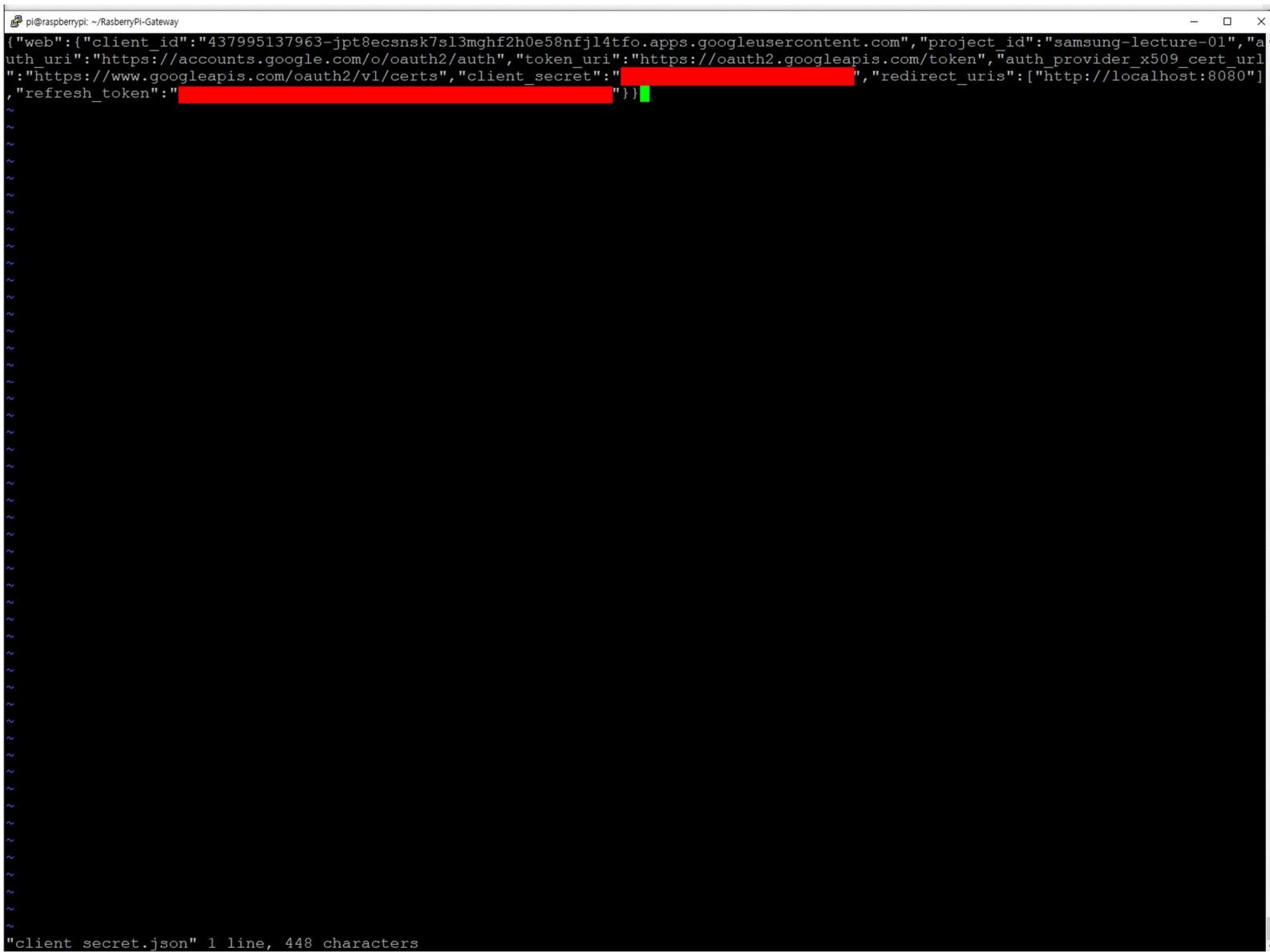
- ❖ Access Token 및 Refresh Token 받기 - 라즈베리파이 파일 생성



A screenshot of a terminal window on a Raspberry Pi. The window title is "pi@raspberrypi: ~/RaspberryPi-Gateway". The command "vi client_secret.json" is being typed into the terminal. A red callout box highlights the text "vi client_secret.json".

OAuth 2.0 Access Token 받기

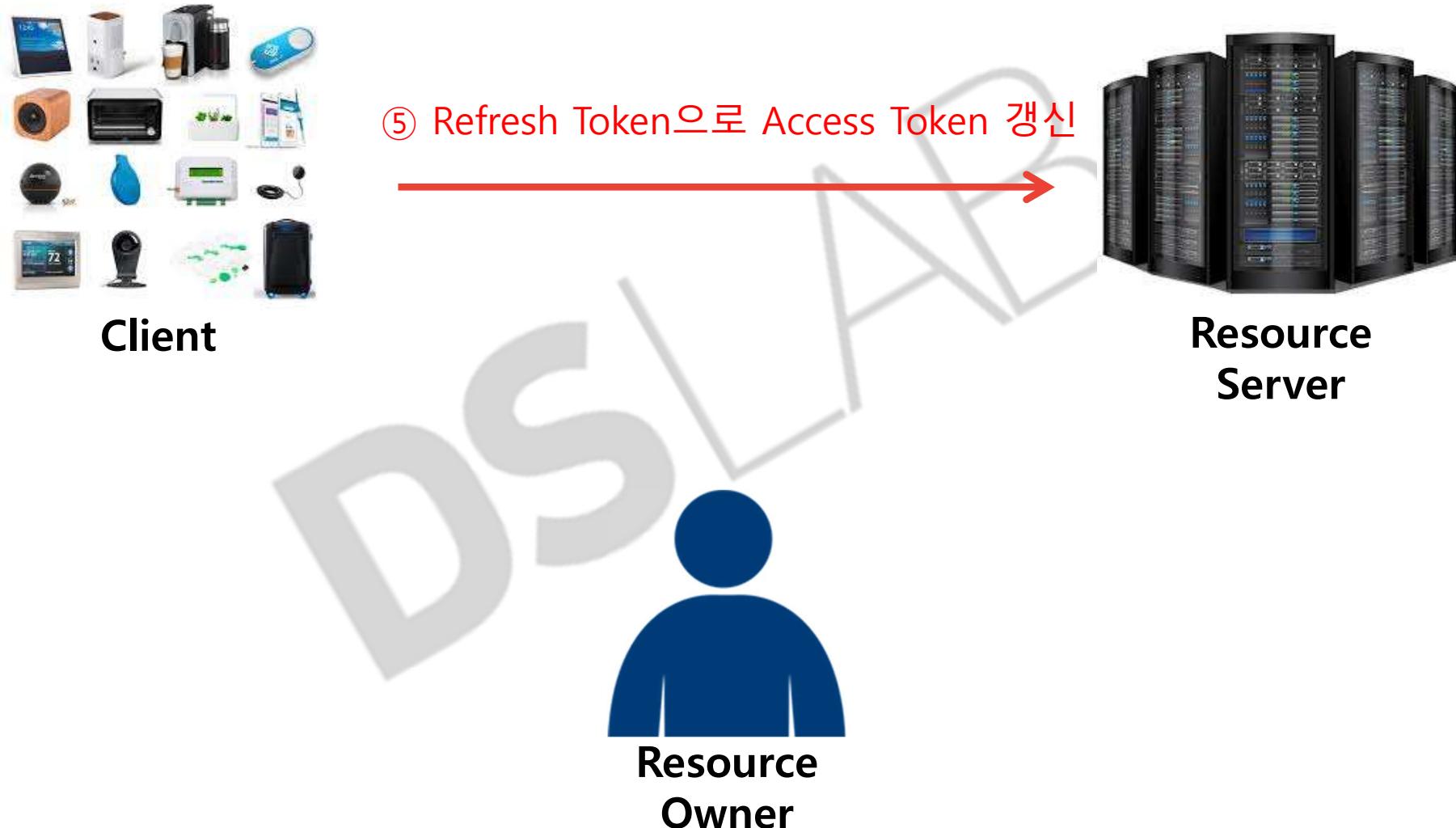
❖ Access Token 및 Refresh Token 받기 - 붙여넣기



```
pi@raspberrypi: ~/RaspberryPi-Gateway
{
  "web": {
    "client_id": "437995137963-jpt8ecsnk7s13mghf2h0e58nfjl4tfo.apps.googleusercontent.com",
    "project_id": "samsung-lecture-01",
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
    "client_secret": "REDACTED",
    "redirect_uris": [
      "http://localhost:8080"
    ],
    "refresh_token": "REDACTED"
  }
}
"client secret.json" 1 line, 448 characters
```

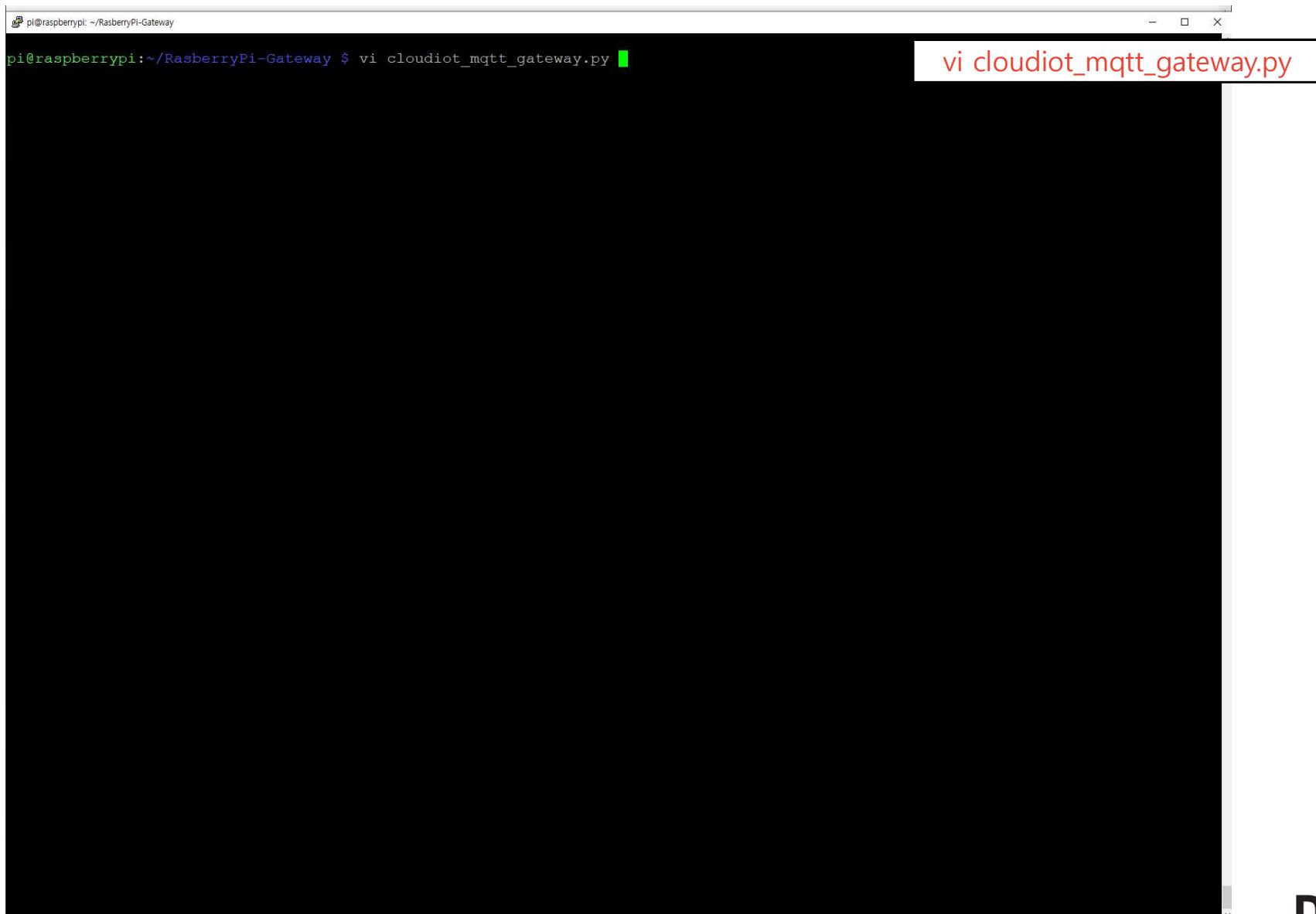
OAuth 2.0 Access Token 받기

- ❖ Access Token을 받는 과정



OAuth 2.0 Access Token 받기

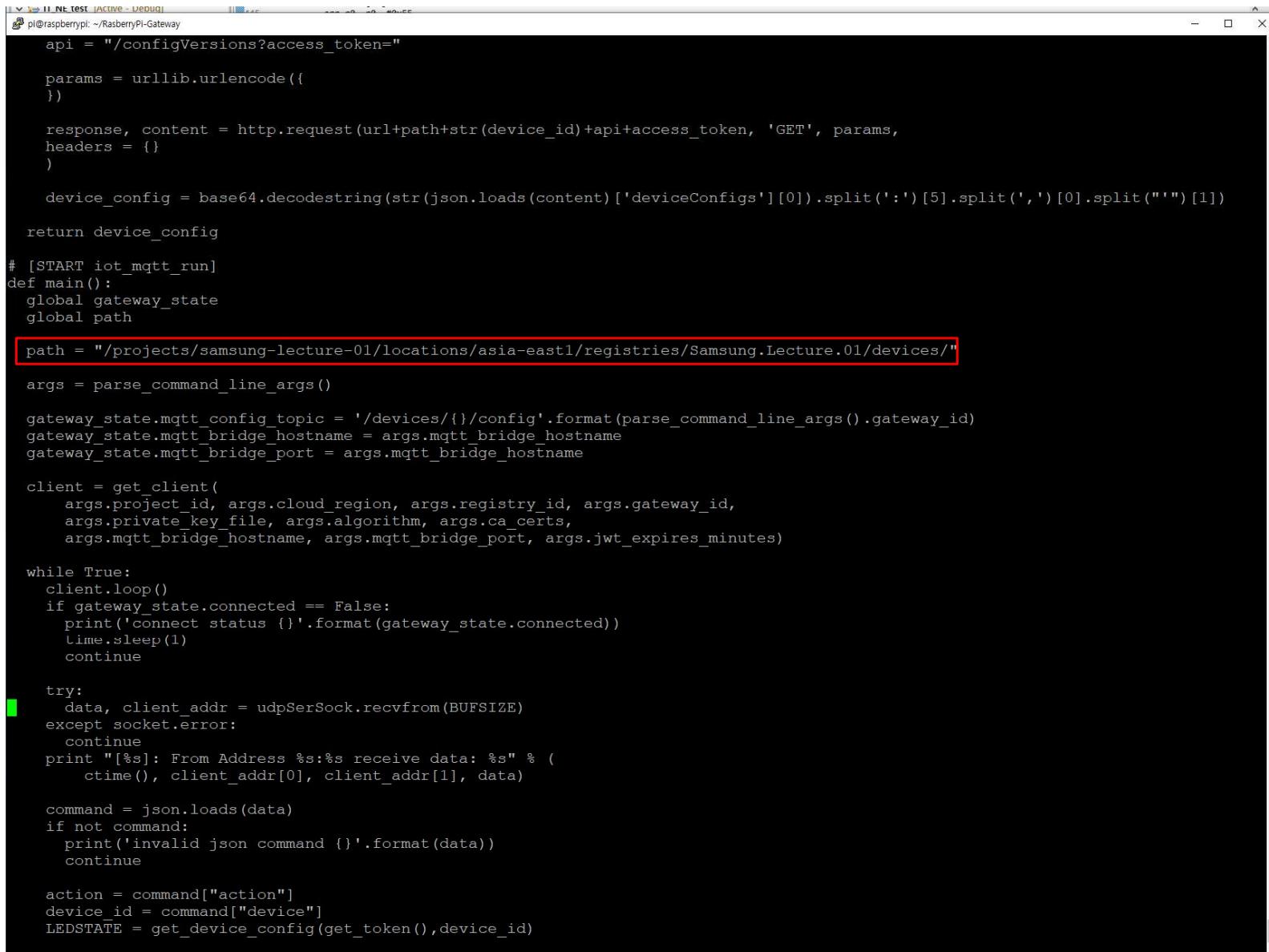
- ❖ Refresh Token으로 Access Token 갱신 - 파이썬 코드 경로 입력



A screenshot of a terminal window titled "pi@raspberrypi: ~/RaspberryPi-Gateway". The command "vi cloudiot_mqtt_gateway.py" is being typed into the terminal. A red box highlights the command "vi cloudiot_mqtt_gateway.py".

OAuth 2.0 Access Token 받기

- ❖ Refresh Token으로 Access Token 갱신 - main함수에 path 수정



```
pi@raspberrypi: ~$ cat RaspberryPi-Gateway.py
api = "/configVersions?access_token="

params = urllib.urlencode({})

response, content = http.request(url+path+str(device_id)+api+access_token, 'GET', params,
headers = {})

device_config = base64.decodestring(str(json.loads(content) ['deviceConfigs'][0]).split(':')[5].split(',') [0].split("') [1])

return device_config

# [START iot_mqtt_run]
def main():
    global gateway_state
    global path

    path = "/projects/samsung-lecture-01/locations/asia-east1/registries/Samsung.Lecture.01/devices/"

    args = parse_command_line_args()

    gateway_state.mqtt_config_topic = '/devices/{}/config'.format(parse_command_line_args().gateway_id)
    gateway_state.mqtt_bridge_hostname = args.mqtt_bridge_hostname
    gateway_state.mqtt_bridge_port = args.mqtt_bridge_hostname

    client = get_client(
        args.project_id, args.cloud_region, args.registry_id, args.gateway_id,
        args.private_key_file, args.algorithm, args.ca_certs,
        args.mqtt_bridge_hostname, args.mqtt_bridge_port, args.jwt_expires_minutes)

    while True:
        client.loop()
        if gateway_state.connected == False:
            print('connect status {}'.format(gateway_state.connected))
            time.sleep(1)
            continue

        try:
            data, client_addr = udpSerSock.recvfrom(BUFSIZE)
        except socket.error:
            continue
        print "[%s]: From Address %s:%s receive data: %s" % (
            ctime(), client_addr[0], client_addr[1], data)

        command = json.loads(data)
        if not command:
            print('invalid json command {}'.format(data))
            continue

        action = command["action"]
        device_id = command["device"]
        LEDSTATE = get_device_config(get_token(), device_id)
```

OAuth 2.0 Access Token 받기

❖ Refresh Token으로 Access Token 갱신 - 게이트웨이 및 디바이스 실행

The screenshot shows two terminal windows. The top window is titled 'pi@raspberrypi: ~/RaspberryPi-Gateway' and the bottom window is titled 'COM10 - PUTTY'. Both windows display MQTT message exchange logs.

Top Window (Gateway Log):

```
pi@raspberrypi: ~/RaspberryPi-Gateway
25
on_publish, userdata None, mid 11
pending response count 0
[Mon Apr 15 16:17:49 2019]: From Address 192.168.0.32:49154 receive data: { "device" : "temp-led-artik", "data": "temp=25" }
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 12 for response { "device": temp-led-artik, "command": "event", "status" : "ok" }
25
on_publish, userdata None, mid 12
pending response count 0
[Mon Apr 15 16:17:53 2019]: From Address 192.168.0.32:49154 receive data: { "device" : "temp-led-artik", "action":"event", "data": "temp=25" }
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 13 for response { "device": temp-led-artik, "command": "event", "status" : "ok" }
25
on_publish, userdata None, mid 13
pending response count 0
[Mon Apr 15 16:17:58 2019]: From Address 192.168.0.32:49154 receive data: { "device" : "temp-led-artik", "action":"event", "data": "temp=25" }
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=25'
Save mid 14 for response { "device": temp-led-artik, "command": "event", "status" : "ok" }
25
on_publish, userdata None, mid 14
pending response count 0
```

Bottom Window (Device Log):

```
Recevied: { "device": temp-led-artik, "command": "attach", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"subscribe" }
Recevied: { "device": temp-led-artik, "command": "subscribe", "status" : "ok" }
Recevied: LEDOFF
*****LED Is Off*****
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action": "event", "data": "temp=25" }
Recevied: { "device": temp-led-artik, "command": "event", "status" : "ok" }
```

A red box highlights the text 'source run-gateway' in the top window's log.

OAuth 2.0 Access Token 받기

❖ Refresh Token으로 Access Token 갱신 - 디바이스 테스트

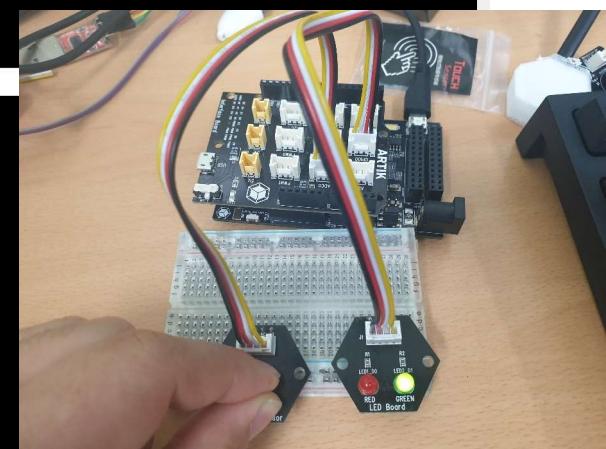
```
pi@raspberrypi: ~/RaspberryPi-Gateway
32
on_publish, userdata None, mid 34
pending response count 0
[Mon Apr 15 16:19:41 2019]: From Address 192.168.0.32:49154 receive data: { "device" : "temp-led-artik", "action":"event", "data":"
temp=32" }
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=32'
Save mid 35 for response { "device": temp-led-artik, "command": "event", "status" : "ok" }

32
on_publish, userdata None, mid 35
pending response count 0
[Mon Apr 15 16:19:45 2019]: From Address 192.168.0.32:49154 receive data: { "device" : "temp-led-artik", "action":"event", "data":"
temp=32" }
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=32'
Save mid 36 for response { "device": temp-led-artik, "command": "event", "status" : "ok" }

32
on_publish, userdata None, mid 36
pending response count 0
[Mon Apr 15 16:19:50 2019]: From Address 192.168.0.32:49154 receive data: { "device" : "temp-led-artik", "action":"event", "data":"
temp=32" }
Sending telemetry event for device temp-led-artik
Publishing message to topic /devices/temp-led-artik/events with payload 'temp=32'
Save mid 37 for response { "device": temp-led-artik, "command": "event", "status" : "ok" }

32
on_publish, userdata None, mid 37
pending response count 0
pi@raspberrypi: ~
```

30도가 넘어가면 LED가 켜지는지 확인



```
COM10 - PUTTY
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=29" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=28" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=30" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=31" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
Recevied: LEDON
*****LED Is On*****
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
send_data: { "device" : "temp-led-artik", "action":"event", "data":"temp=32" }
Recevied: { "device": "temp-led-artik", "command": "event", "status" : "ok" }
```

Thank you

