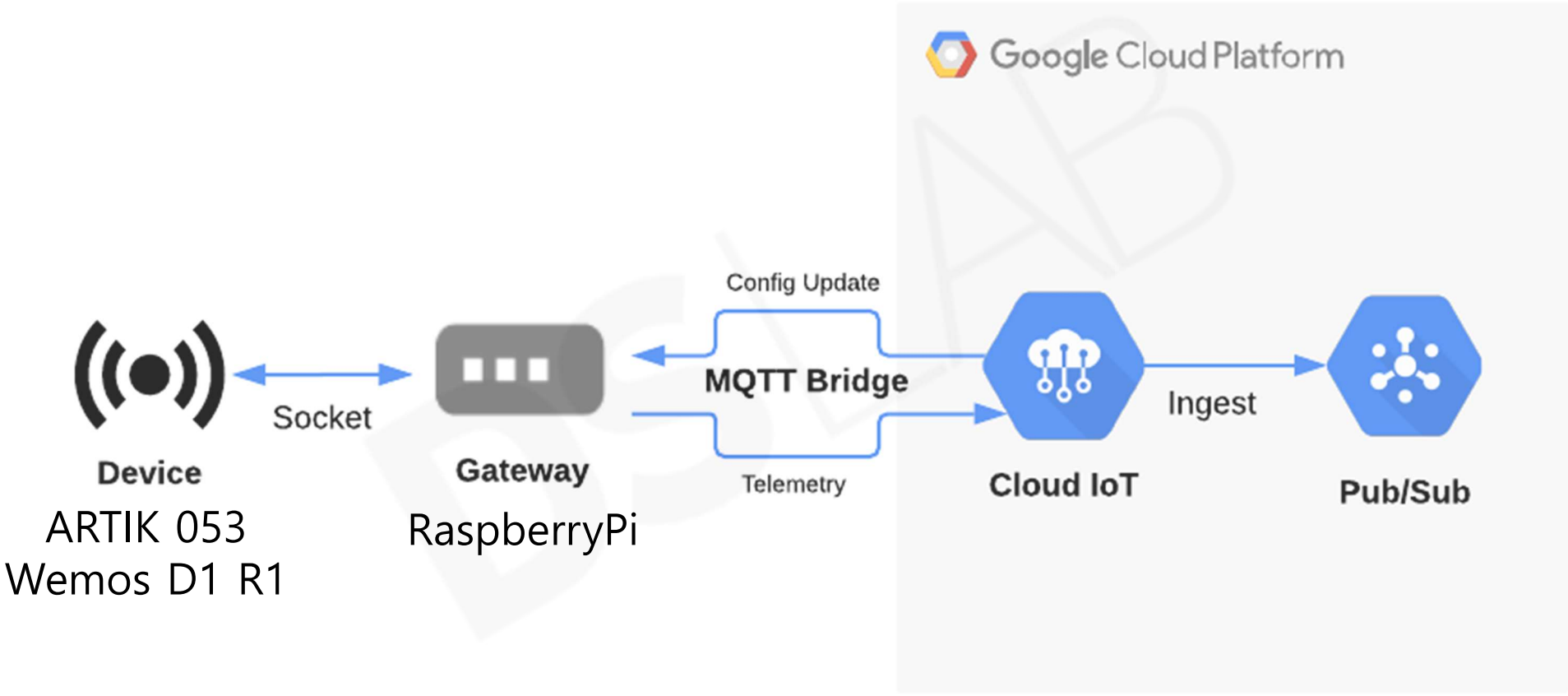
IoT 프로젝트 특별과정

– Part 5. Google IOT Core –

Contents

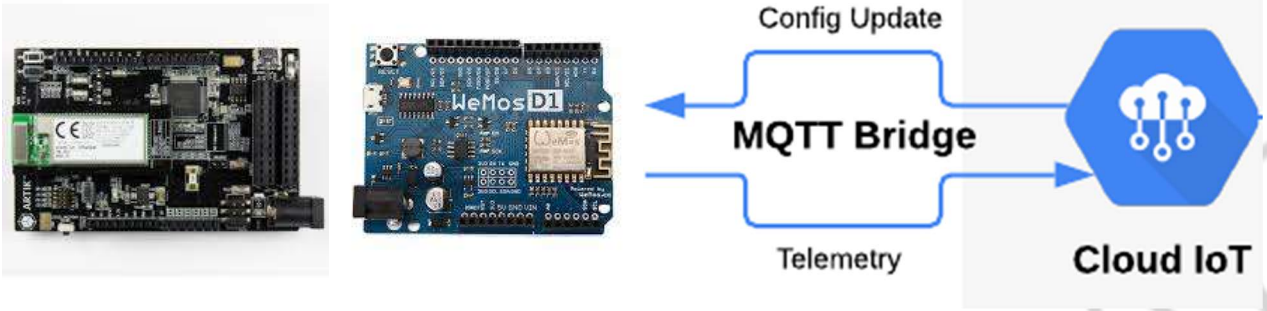
Google IOT Core 전체적 흐름도



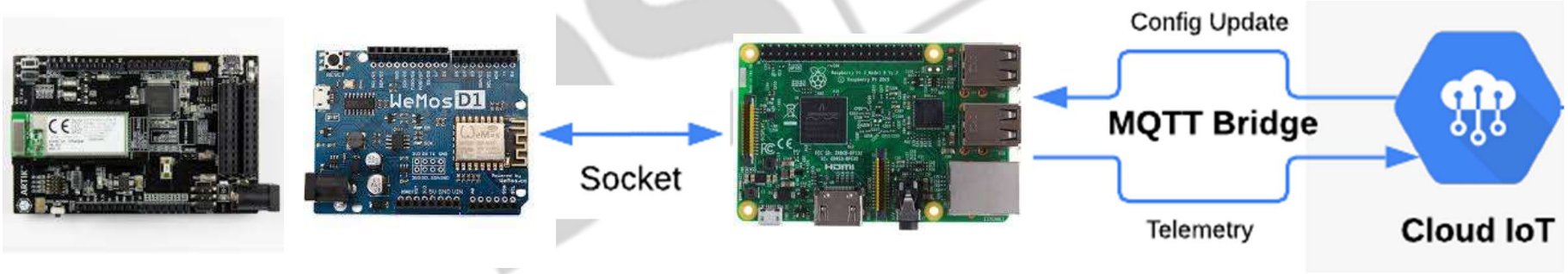
RaspberryPi Gateway 구축하기

RaspberryPi Gateway 구축하기

❖ Gateway를 구축하는 이유



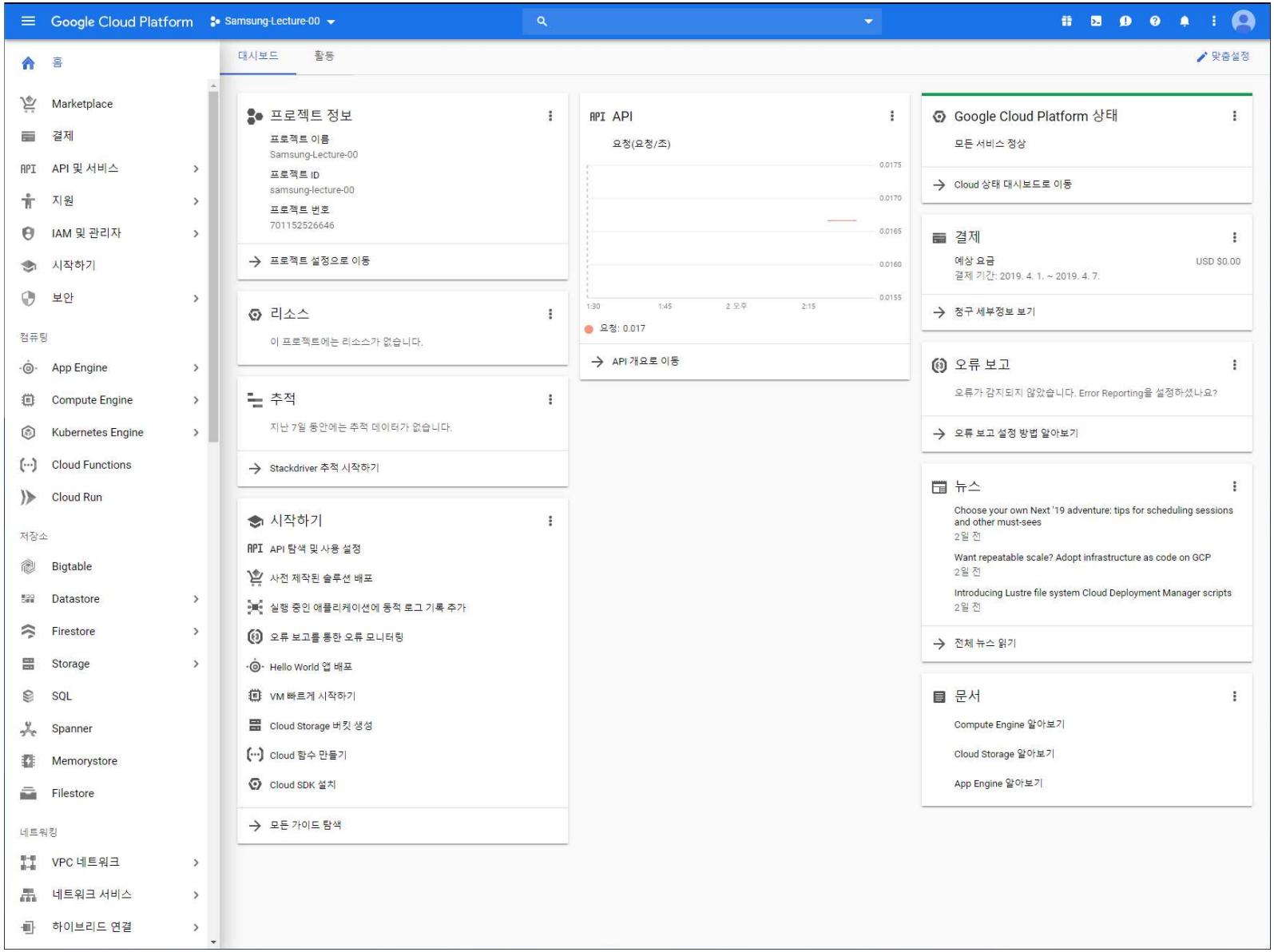
- 각각의 디바이스 키값을 발급 및 관리
- 멀티프로세싱으로 인한 성능 저하
- C언어 기반



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

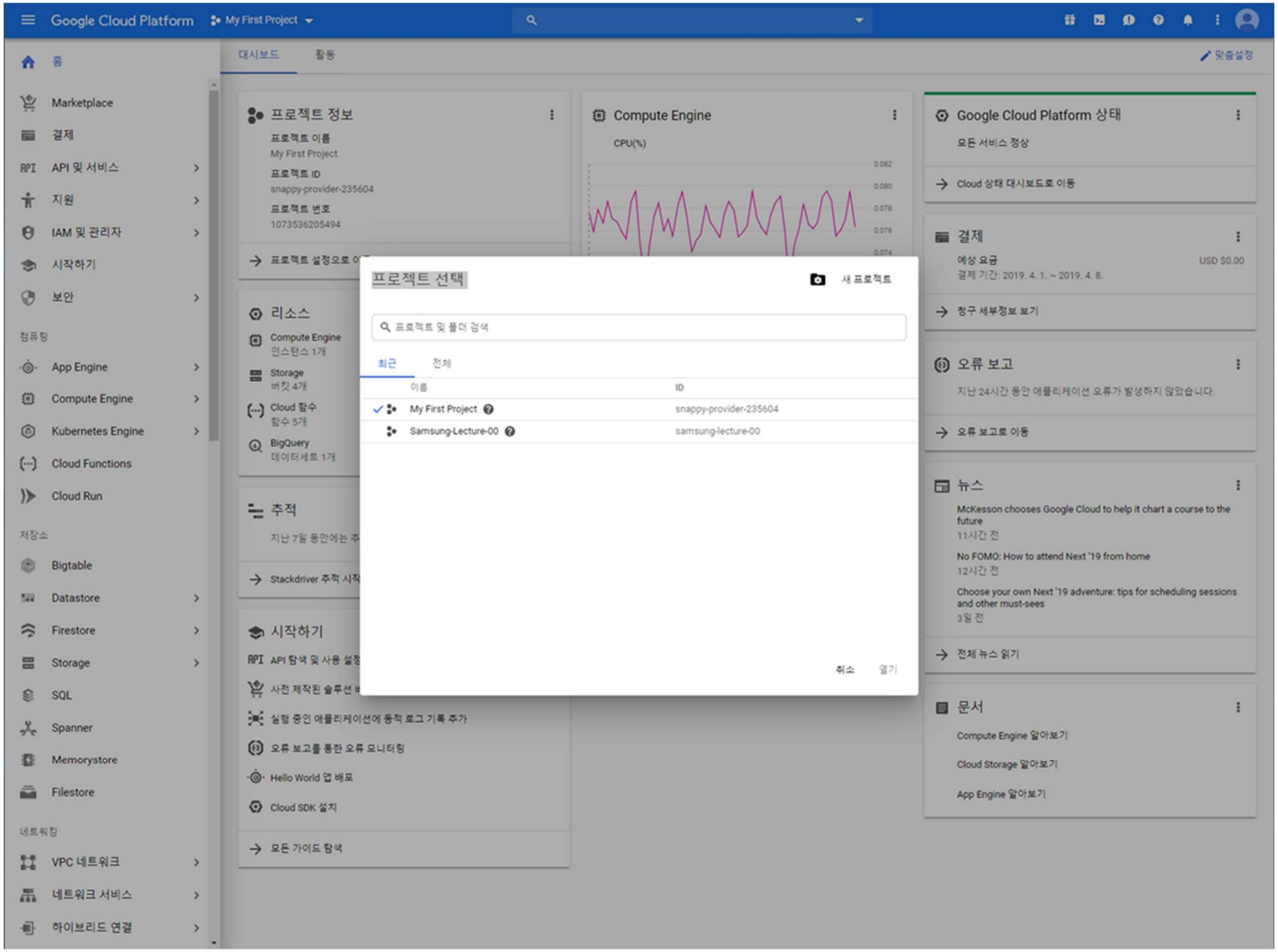
RaspberryPi Gateway 구축하기

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



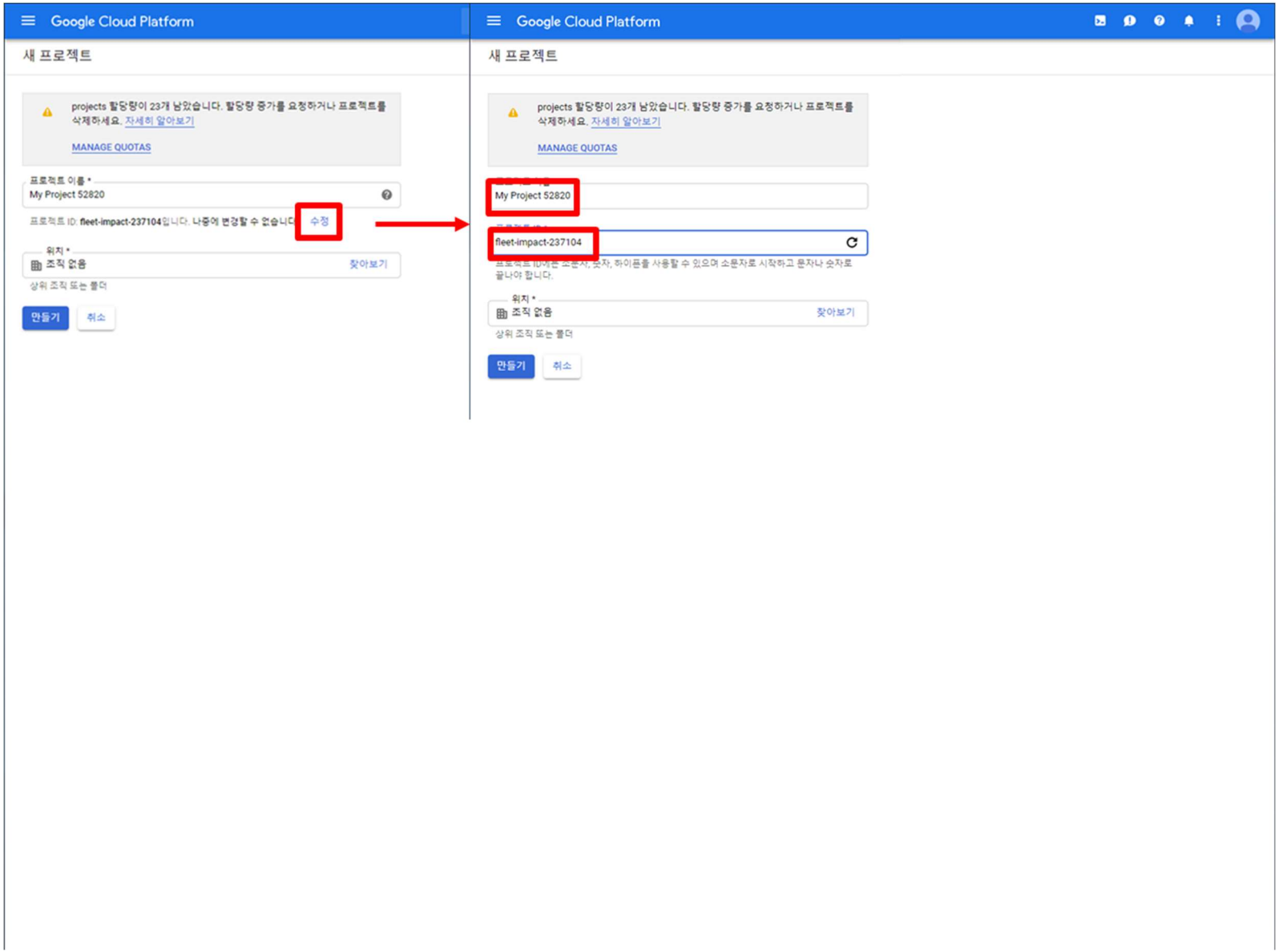
RaspberryPi Gateway 구축하기

❖ 프로젝트 생성 - 프로젝트 선택 클릭 및 새 프로젝트



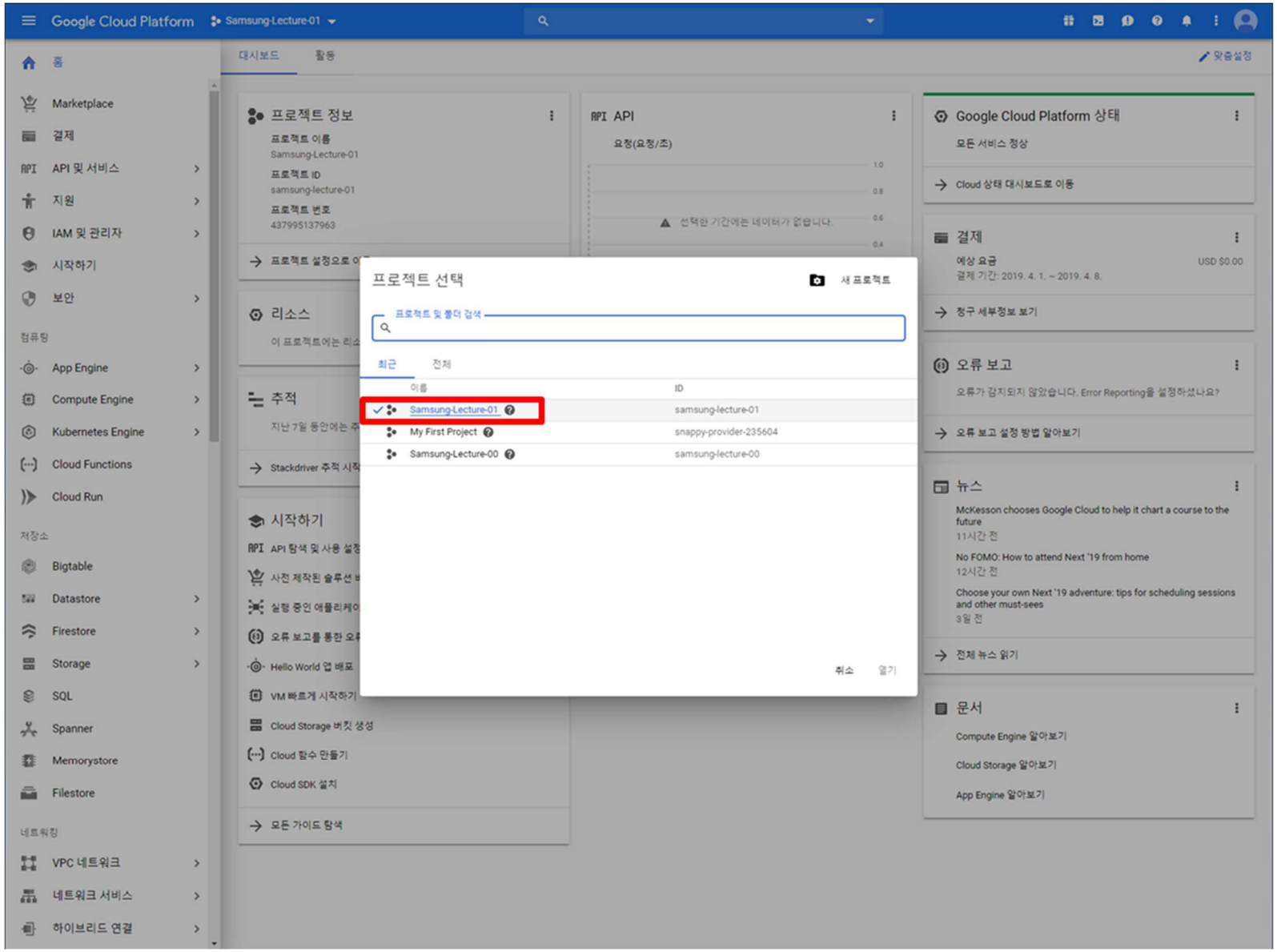
RaspberryPi Gateway 구축하기

❖ 프로젝트 생성 - 프로젝트 이름 및 프로젝트 ID 설정



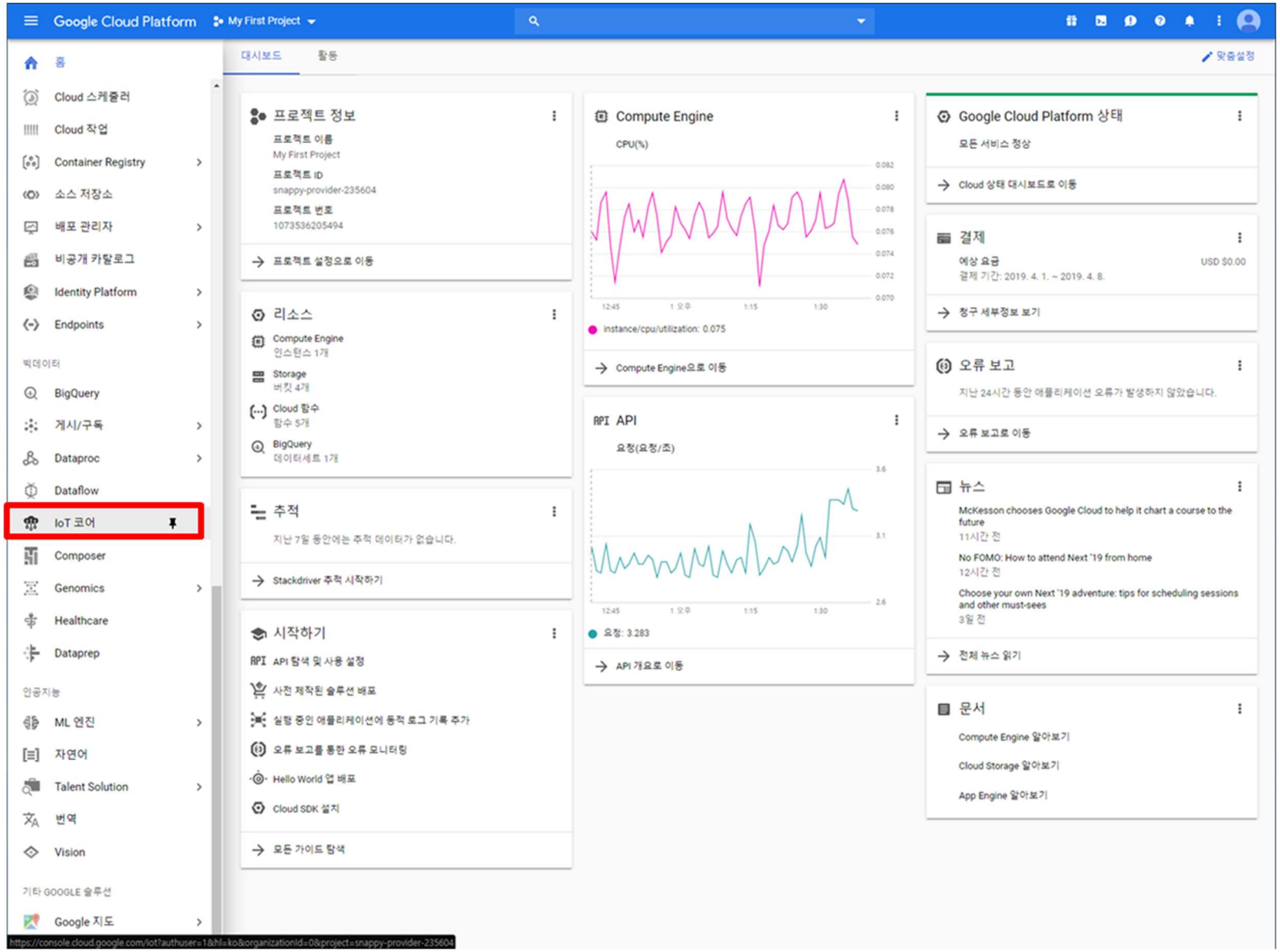
RaspberryPi Gateway 구축하기

❖ 프로젝트 생성 - 프로젝트 선택



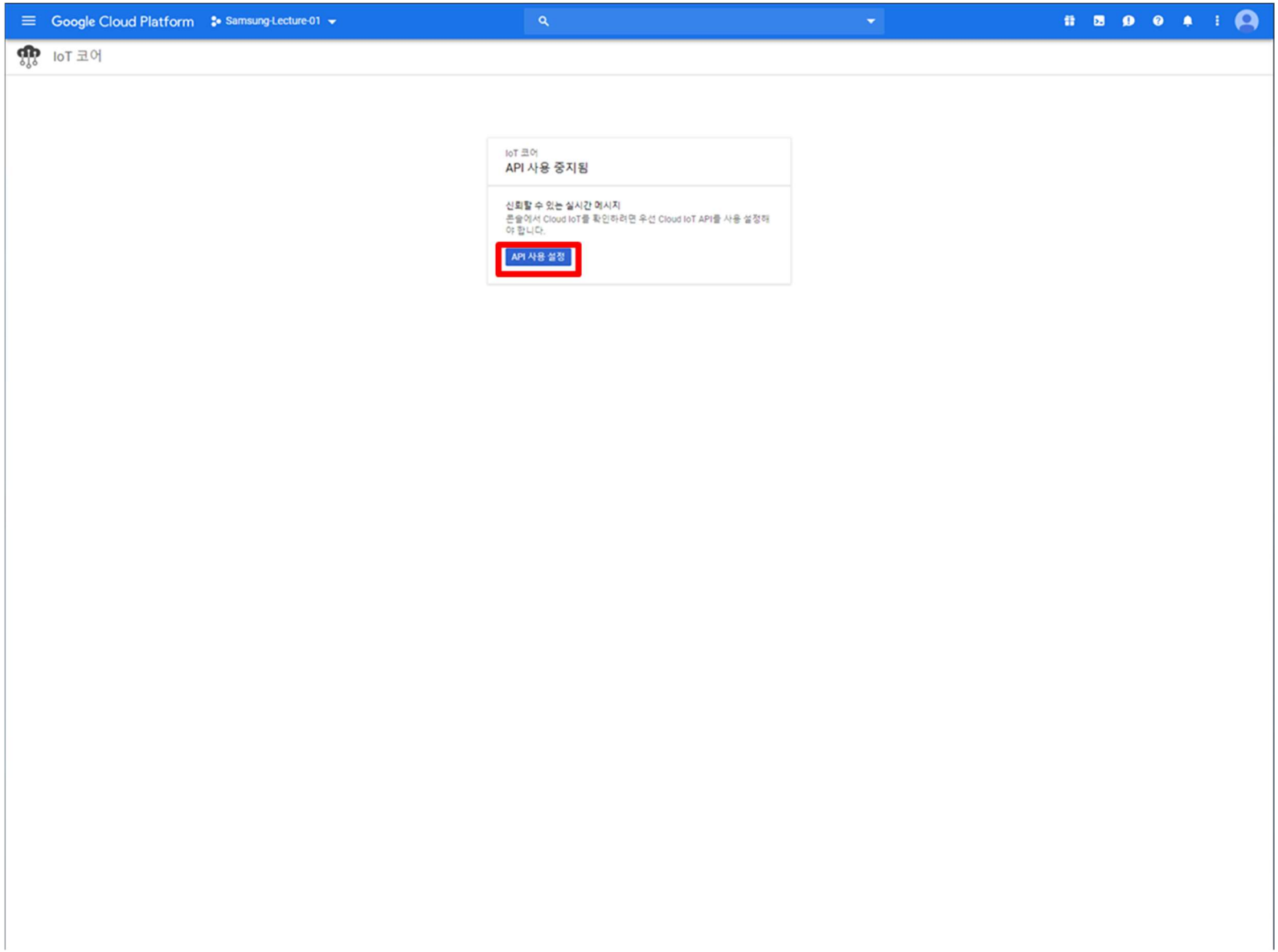
RaspberryPi Gateway 구축하기

❖ 레지스트리 생성 - IoT 코어 클릭



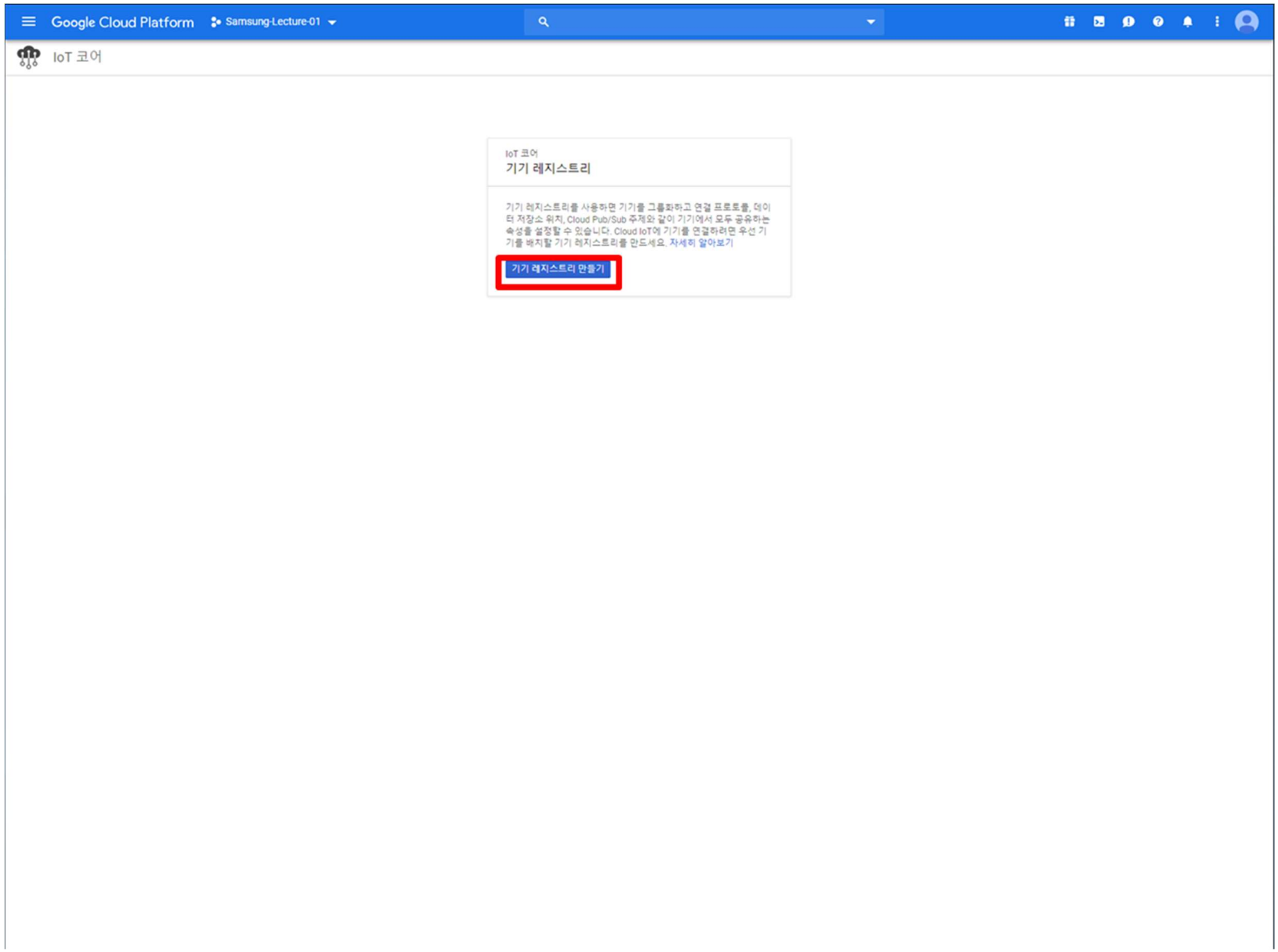
RaspberryPi Gateway 구축하기

❖ 레지스트리 생성 - API 사용 설정



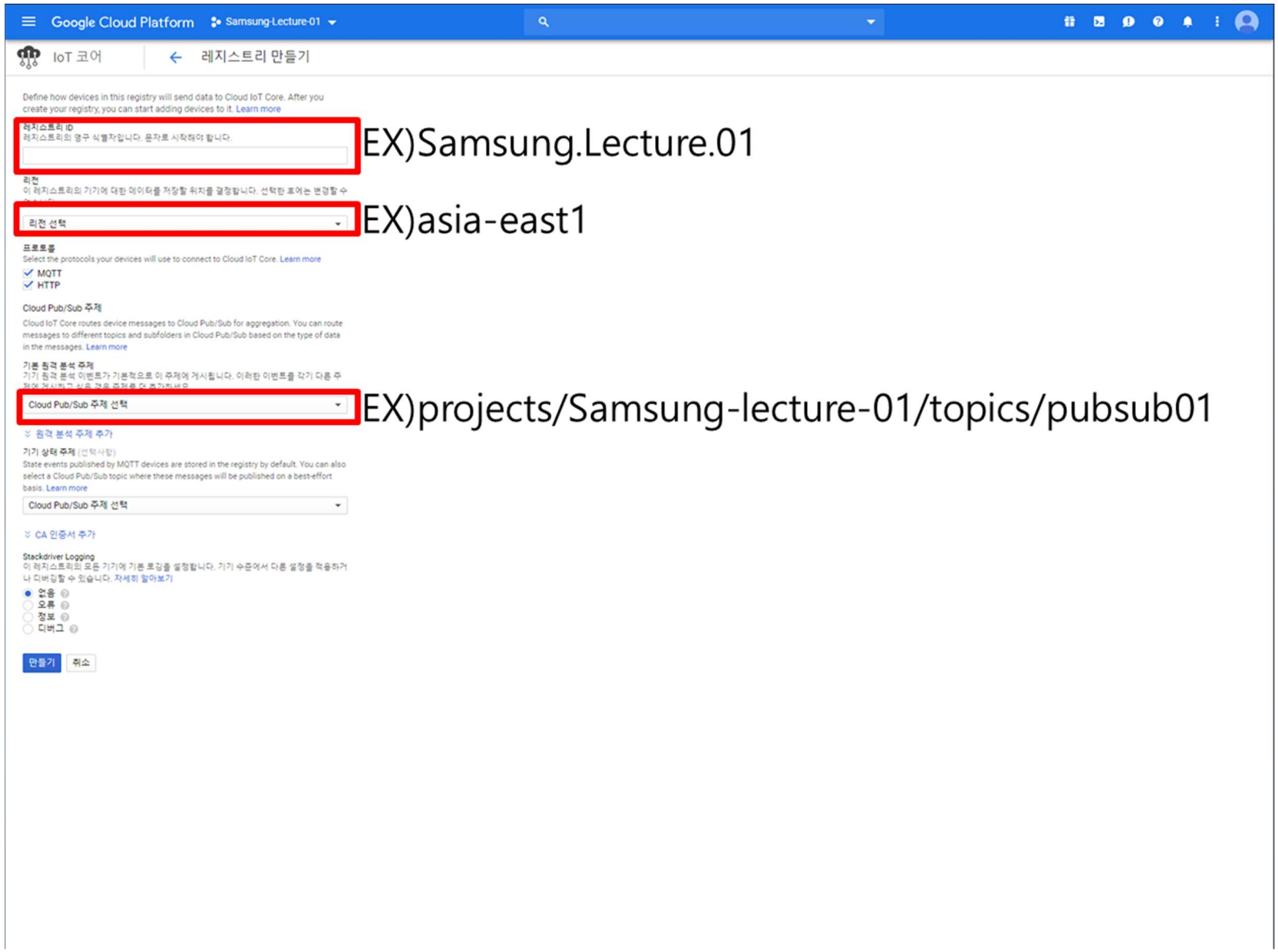
RaspberryPi Gateway 구축하기

❖ 레지스트리 생성 - 레지스트리 만들기 클릭



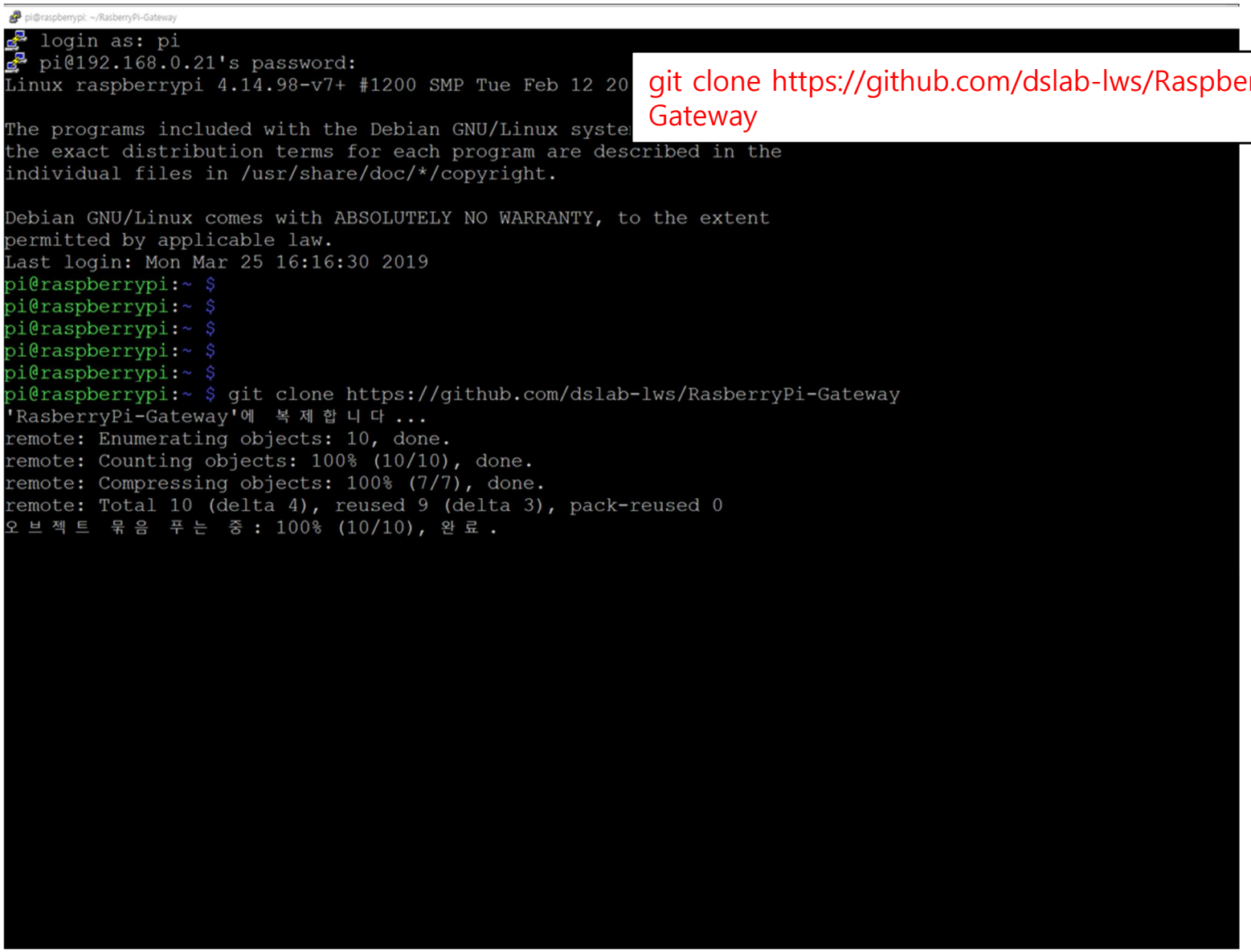
RaspberryPi Gateway 구축하기

❖ 레지스트리 생성 - ID 및 Region, Topic 설정



RaspberryPi Gateway 구축하기

❖ 라즈베리파이 세팅 - 게이트웨이 코드 다운로드



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

RaspberryPi Gateway 구축하기

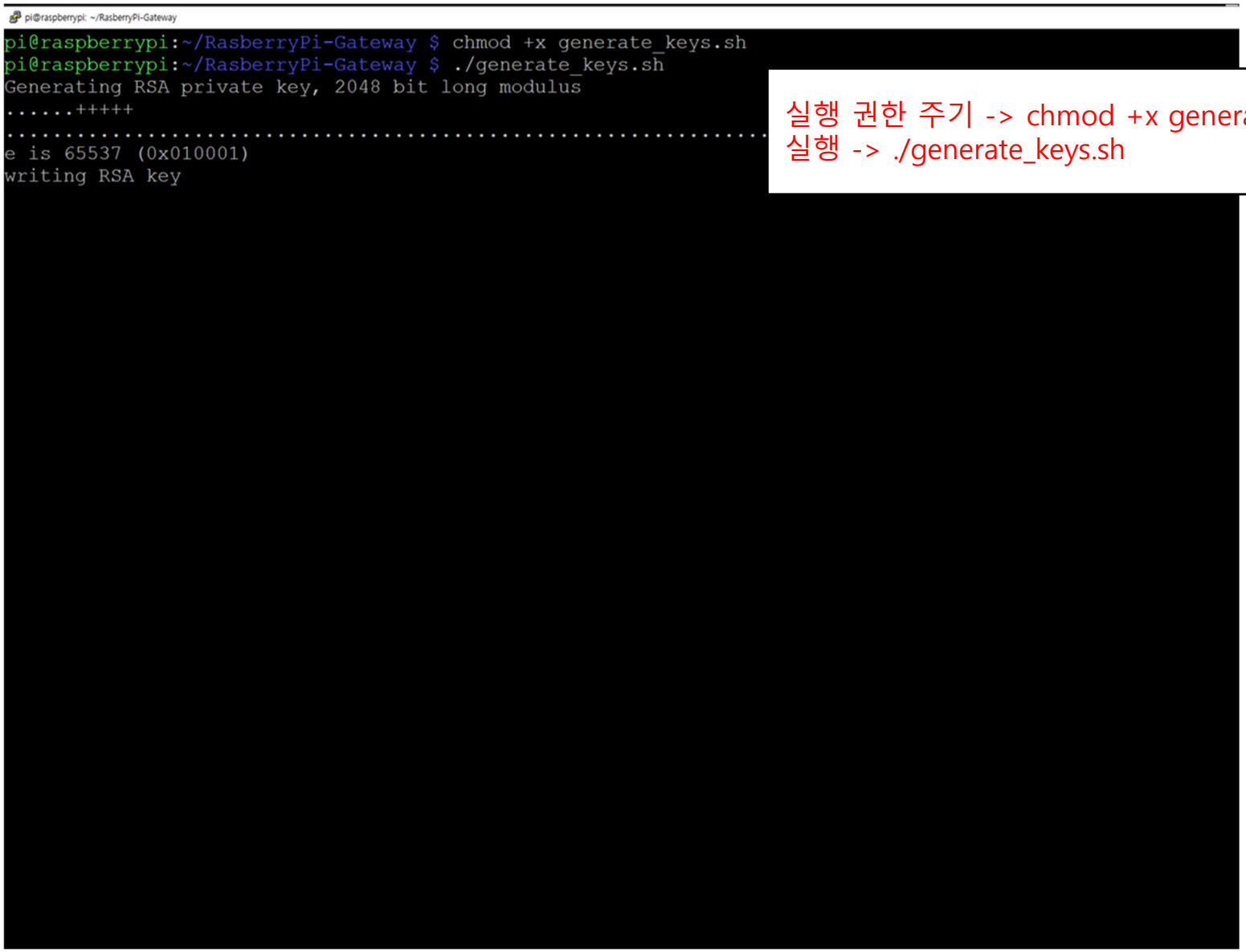
❖ 라즈베리파이 세팅 - 디렉토리 변경

```
pi@raspberrypi: ~/RasberryPi-Gateway
login as: pi
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/RasberryPi-Gateway
'RasberryPi-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.
pi@raspberrypi:~ $ cd RasberryPi-Gateway/
pi@raspberrypi:~/RasberryPi-Gateway $
```

cd RaspberryPi-Gateway

RaspberryPi Gateway 구축하기

❖ 라즈베리파이 세팅 - 키 생성



RaspberryPi Gateway 구축하기

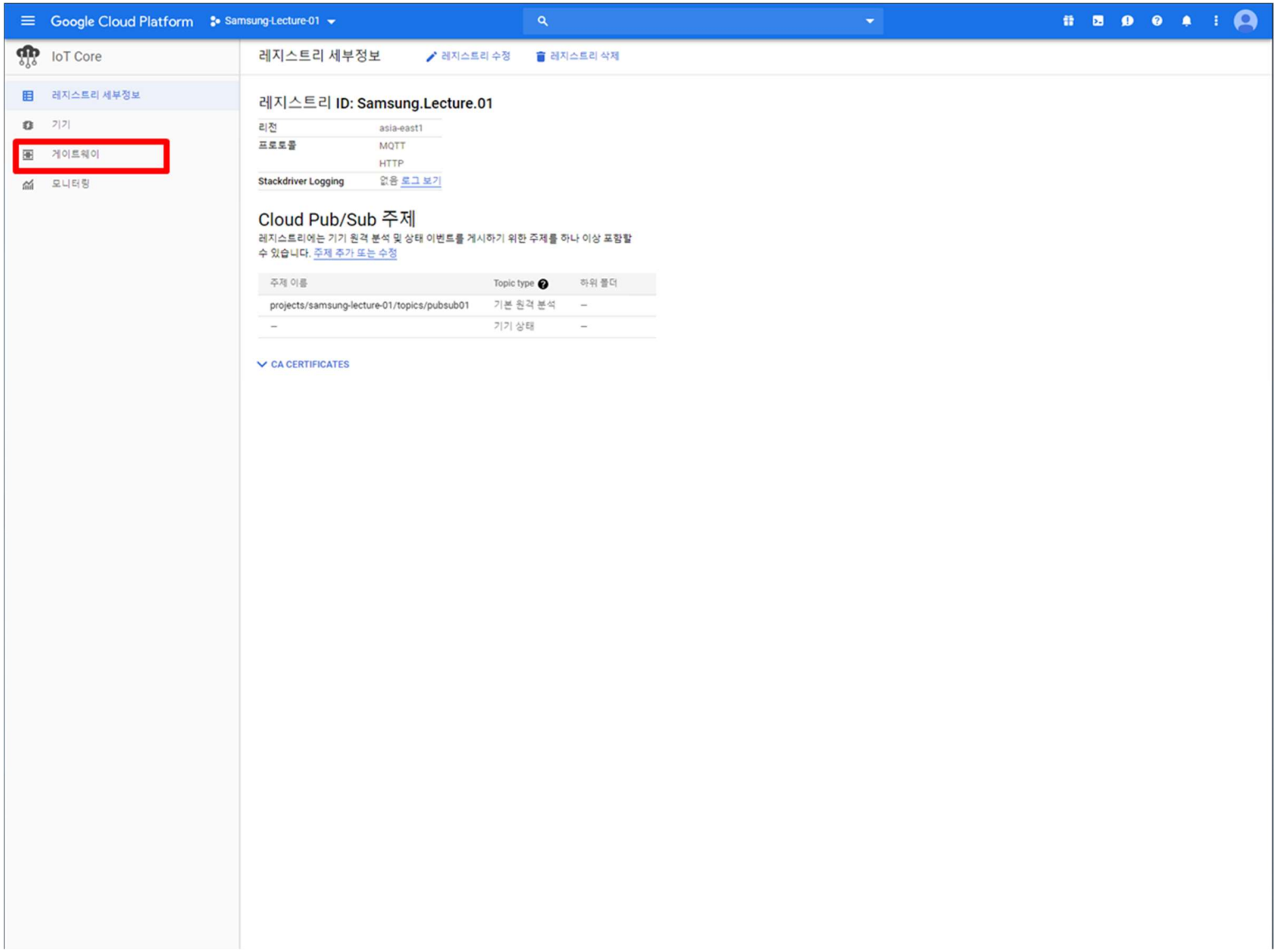
❖ 라즈베리파이 세팅 - 생성된 키 파일 열기 및 키 복사

vi rsa_public.pem
복사 후 :q + Enter

DSLAB

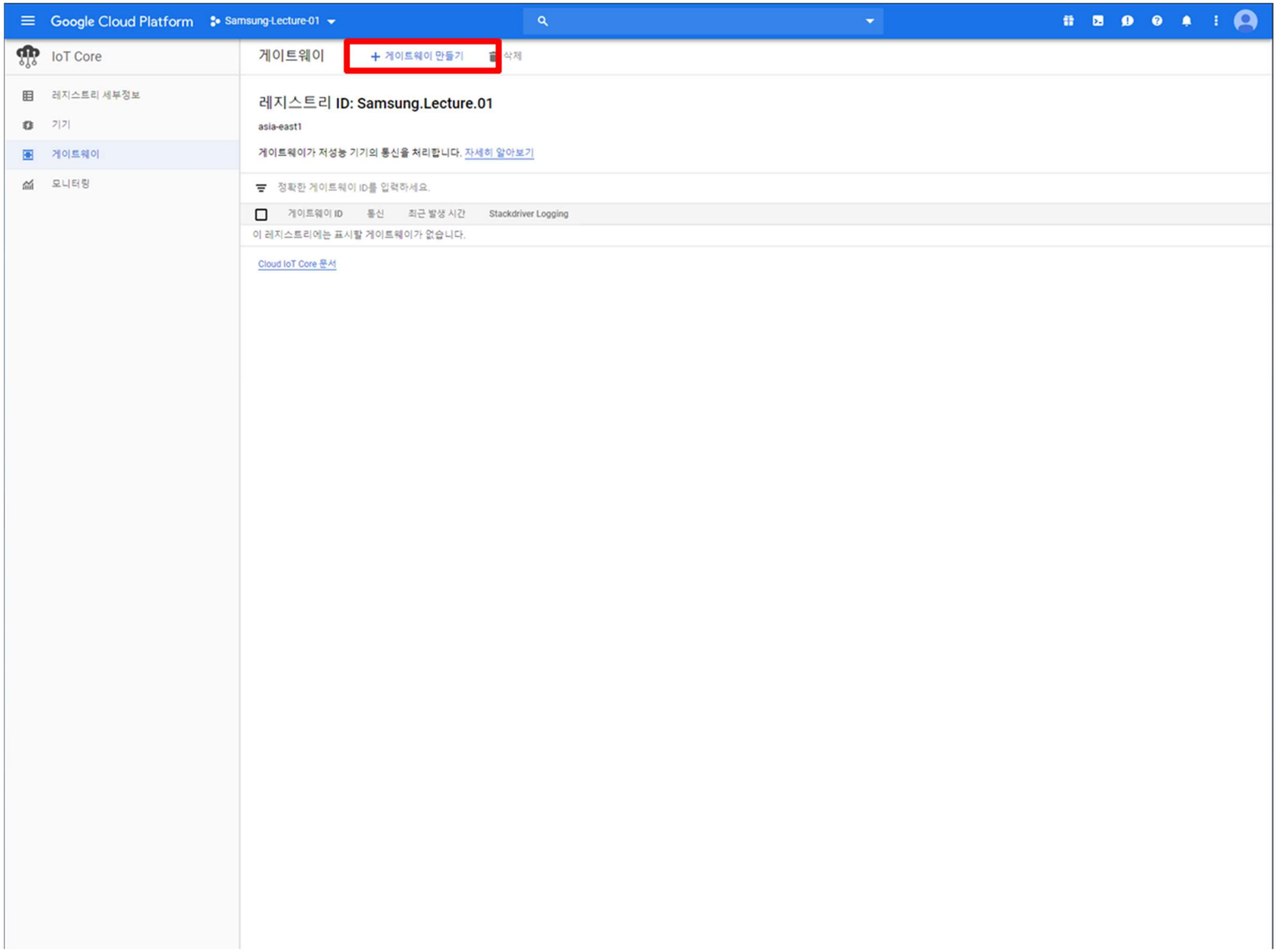
RaspberryPi Gateway 구축하기

❖ 게이트웨이 생성 - 게이트웨이 클릭



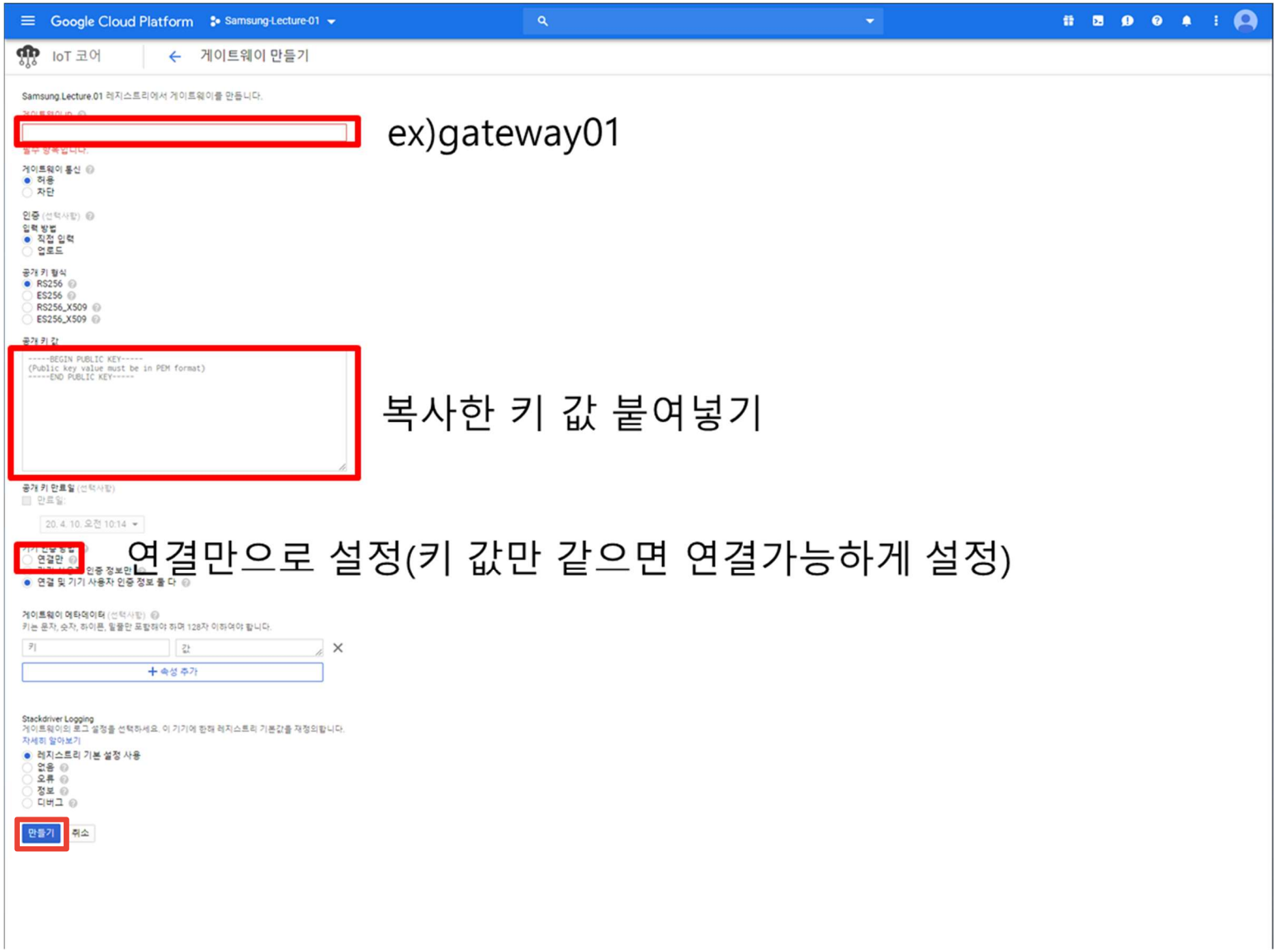
RaspberryPi Gateway 구축하기

❖ 게이트웨이 생성 - 게이트웨이 만들기 클릭



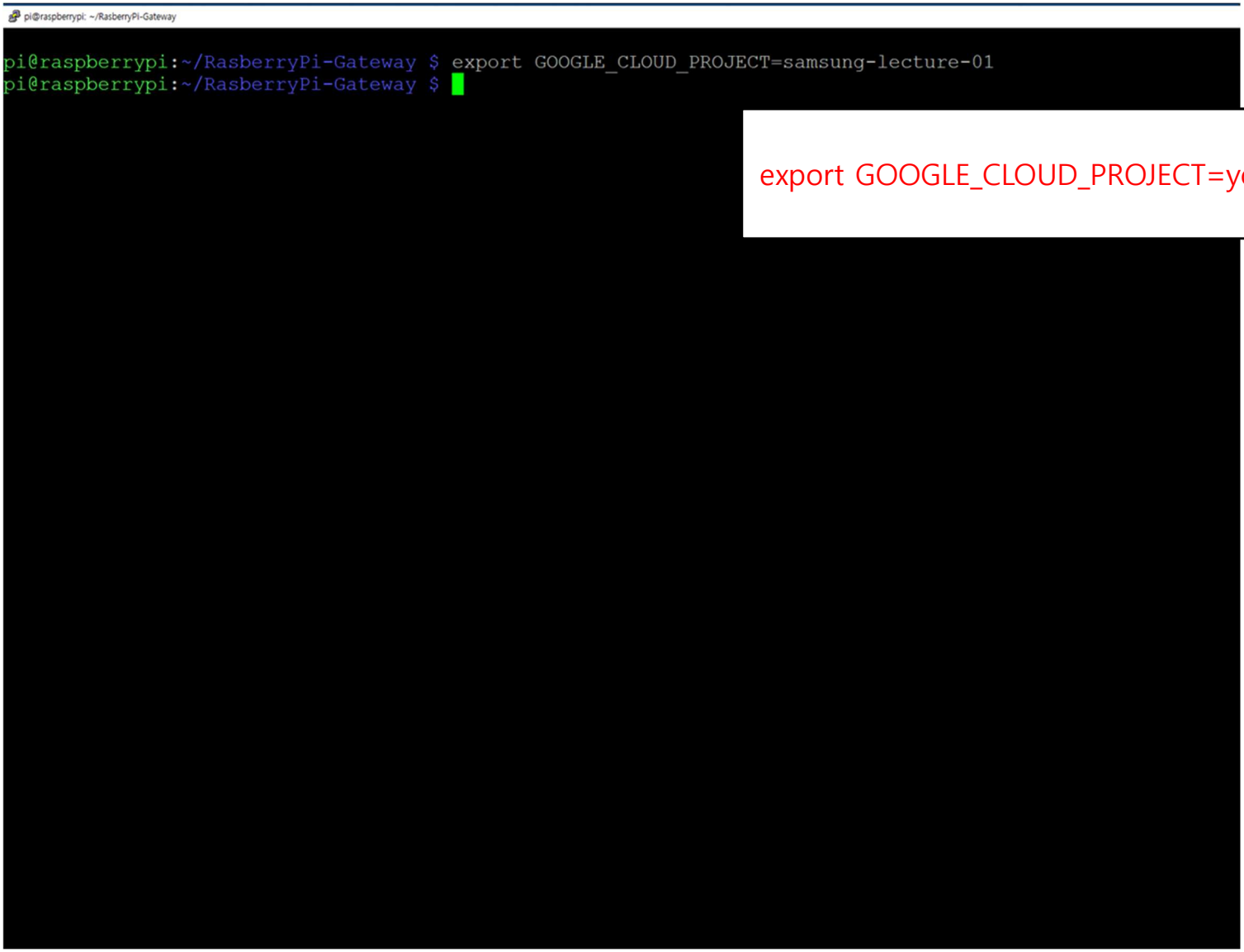
RaspberryPi Gateway 구축하기

❖ 게이트웨이 생성 - Gateway ID 및 키값 설정



RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - Project ID 전역변수 설정



export GOOGLE_CLOUD_PROJECT=your-project-id

RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - run-gateway 파일 열기 및 수정

vi run-gateway

pi@raspberrypi: ~/RasberryPi-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

pip install -r requirements-gateway.txt

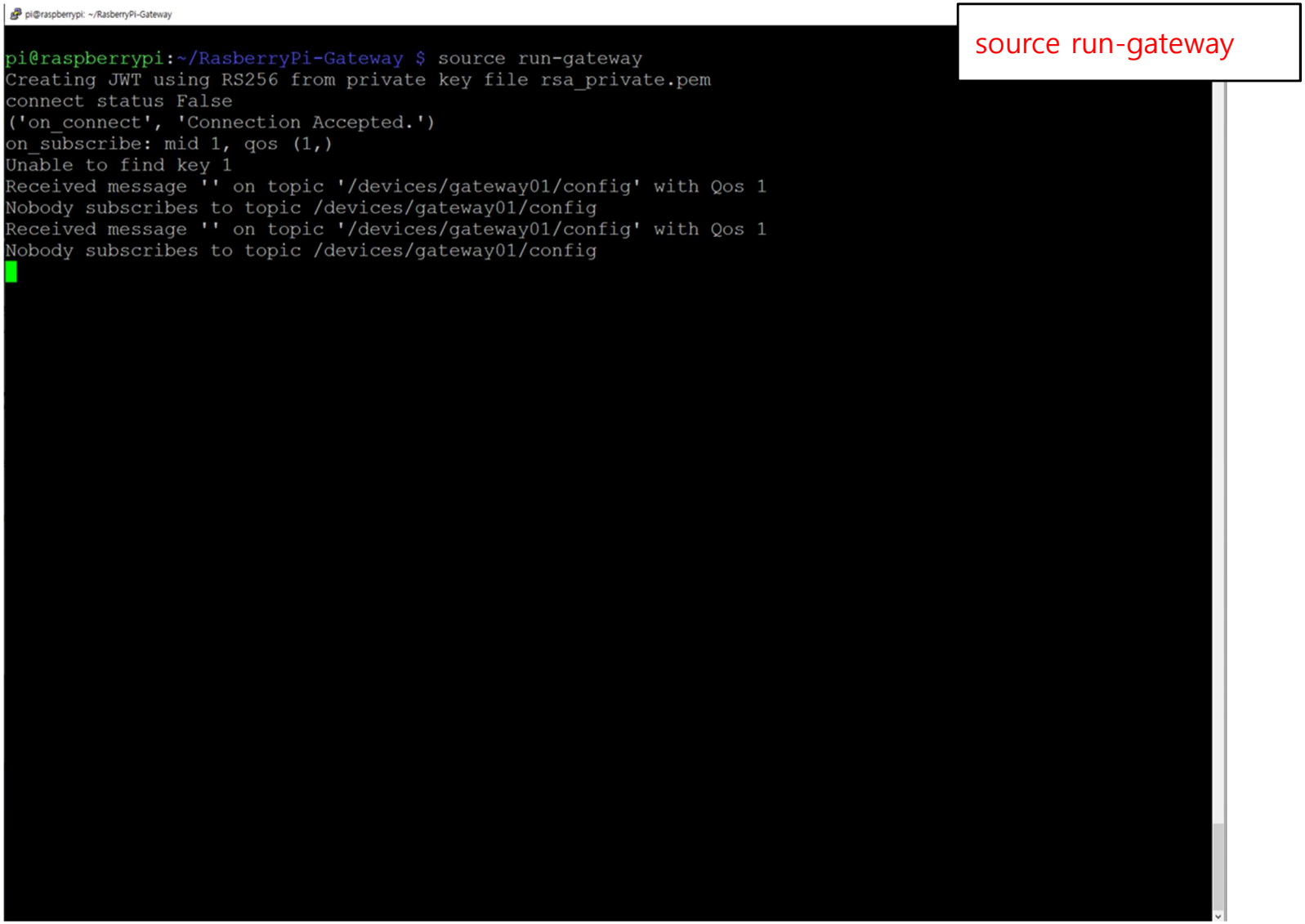
```
pi@raspberrypi:~/RasberryPi-Gateway $ pip install -r requirements-gateway.txt
Collecting cryptography==2.4.1 (from -r requirements-gateway.txt (line 1))
Collecting paho-mqtt==1.4.0 (from -r requirements-gateway.txt (line 2))
Collecting pyjwt==1.6.4 (from -r requirements-gateway.txt (line 3))
  Using 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/e56e6b4bbac7c4a06de1c50de6fe1ef3810018ae11732a50f15f62c7d050/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 션  오 류
pi@raspberrypi:~/RasberryPi-Gateway $
```

RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - Google CA 루트인증서 받아오기

RaspberryPi Gateway 구축하기

❖ 게이트웨이 접속 - 게이트웨이 실행

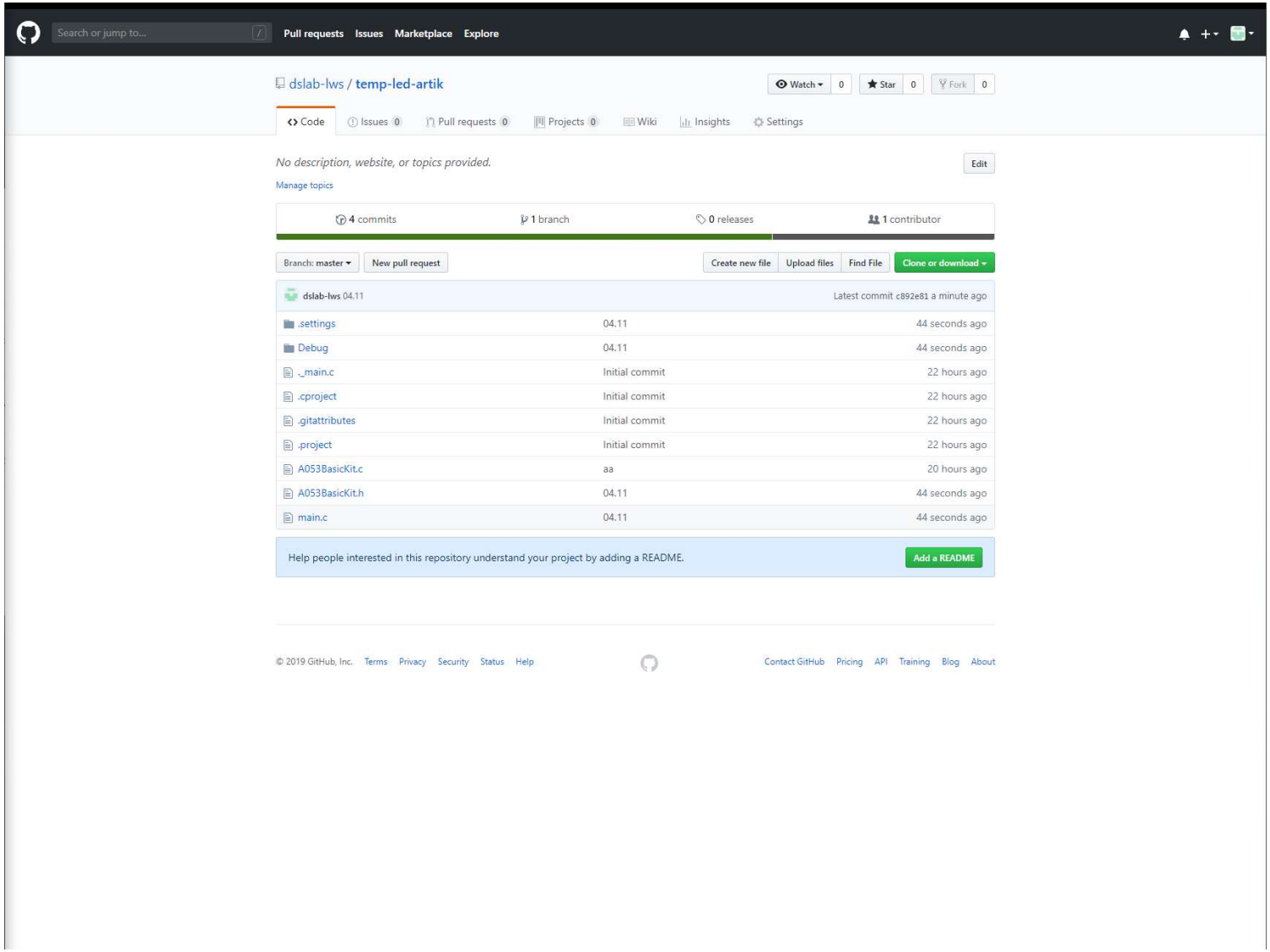


디바이스 만들기
-ARTIK 053

디바이스 만들기 -ARTIK 053

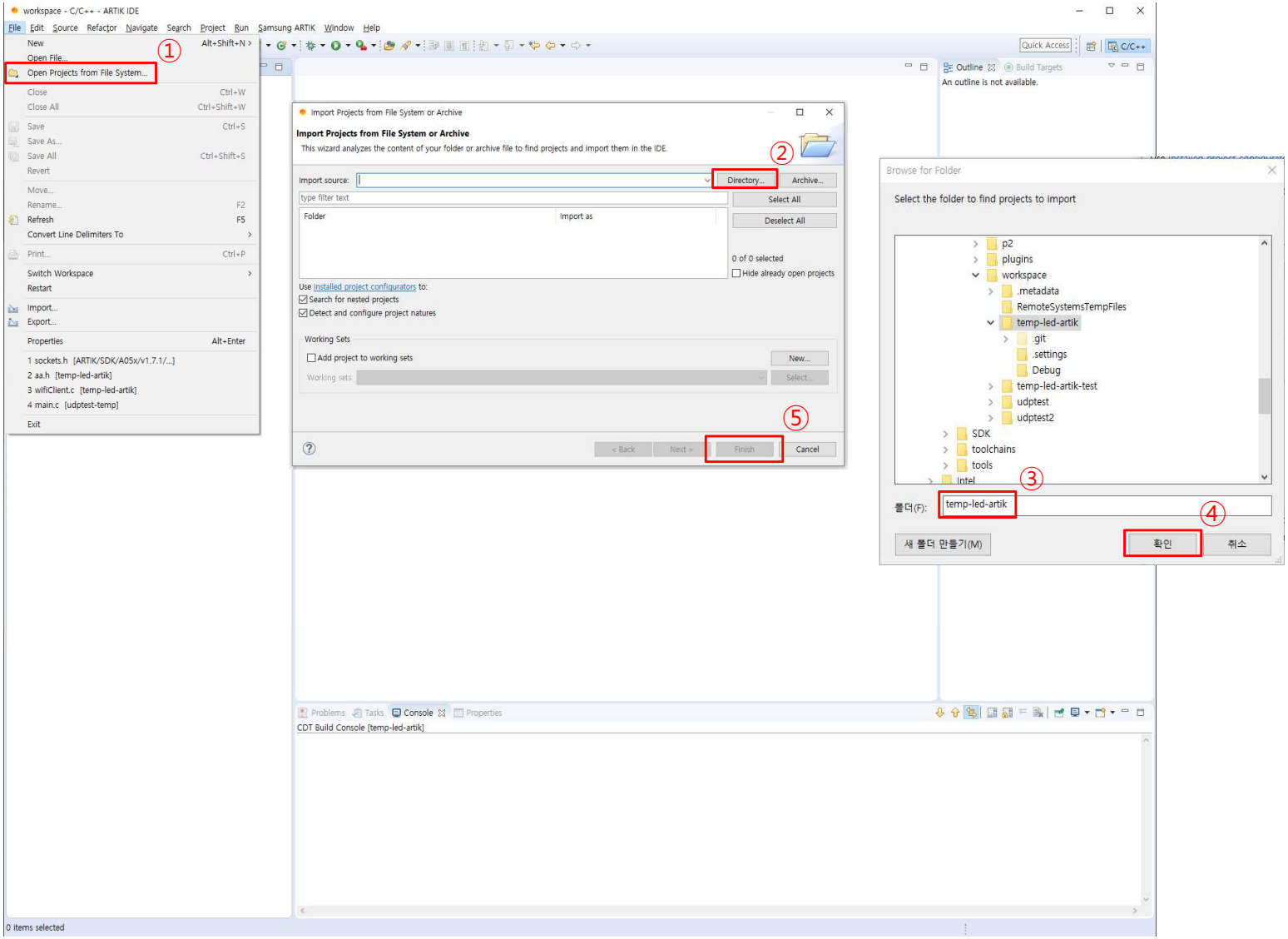
❖ ARTIK 053 세팅 - 아틱 소스코드 다운로드

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

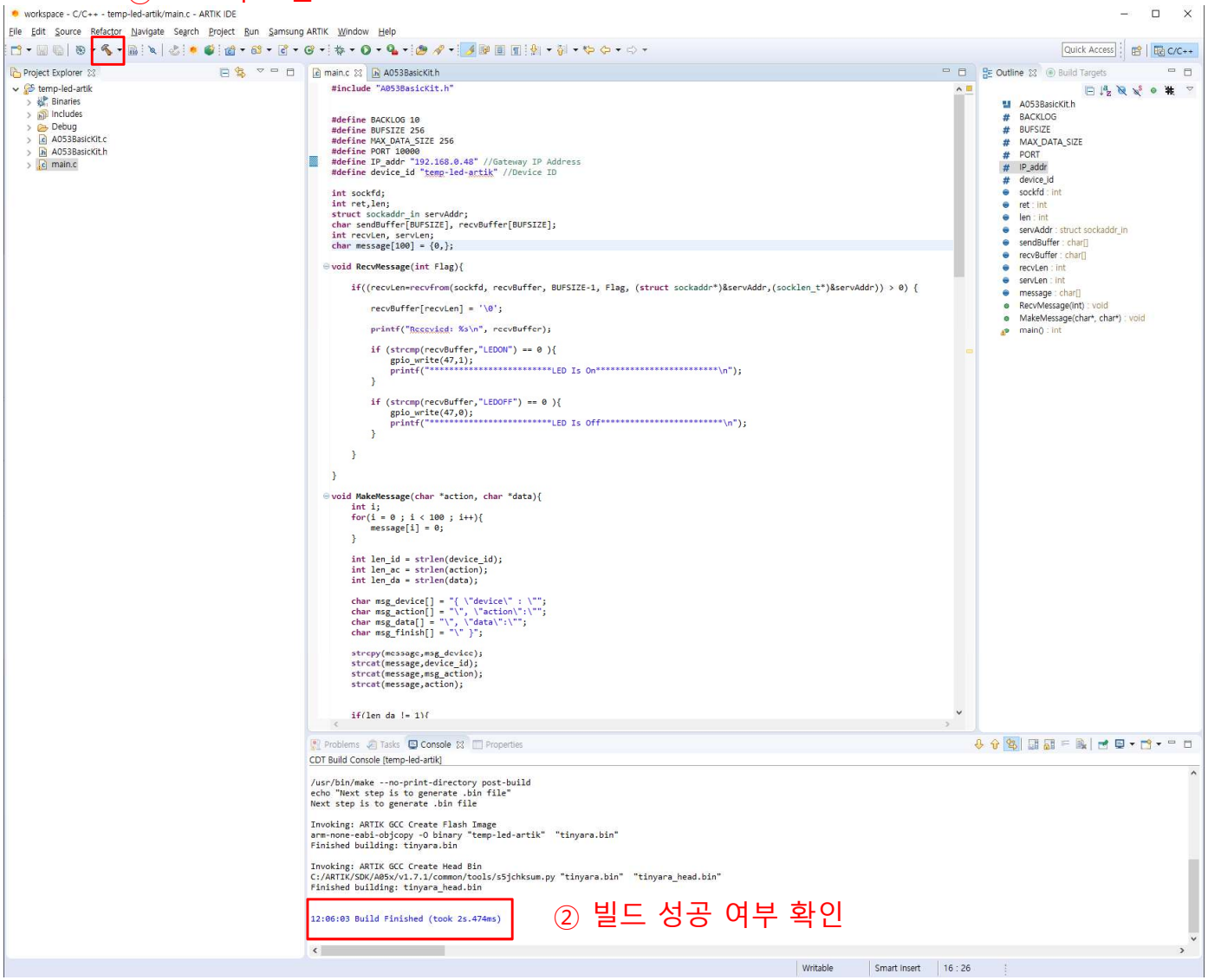
```
// WIFi
#define STATE_DISCONNECTED      0
#define STATE_CONNECTED         1
#define SLSI_WIFI_SECURITY_OPEN    "open"
#define SLSI_WIFI_SECURITY_WPA2_AES "wpa2_aes"
#define NET_DEVNAME "wl1"
#define SSID  "WIFI SSID" //FIX
#define PSK   "WIFI PASSWORD" //FIX
```

와이파이 아름과 비밀번호 입력

디바이스 만들기 -ARTIK 053

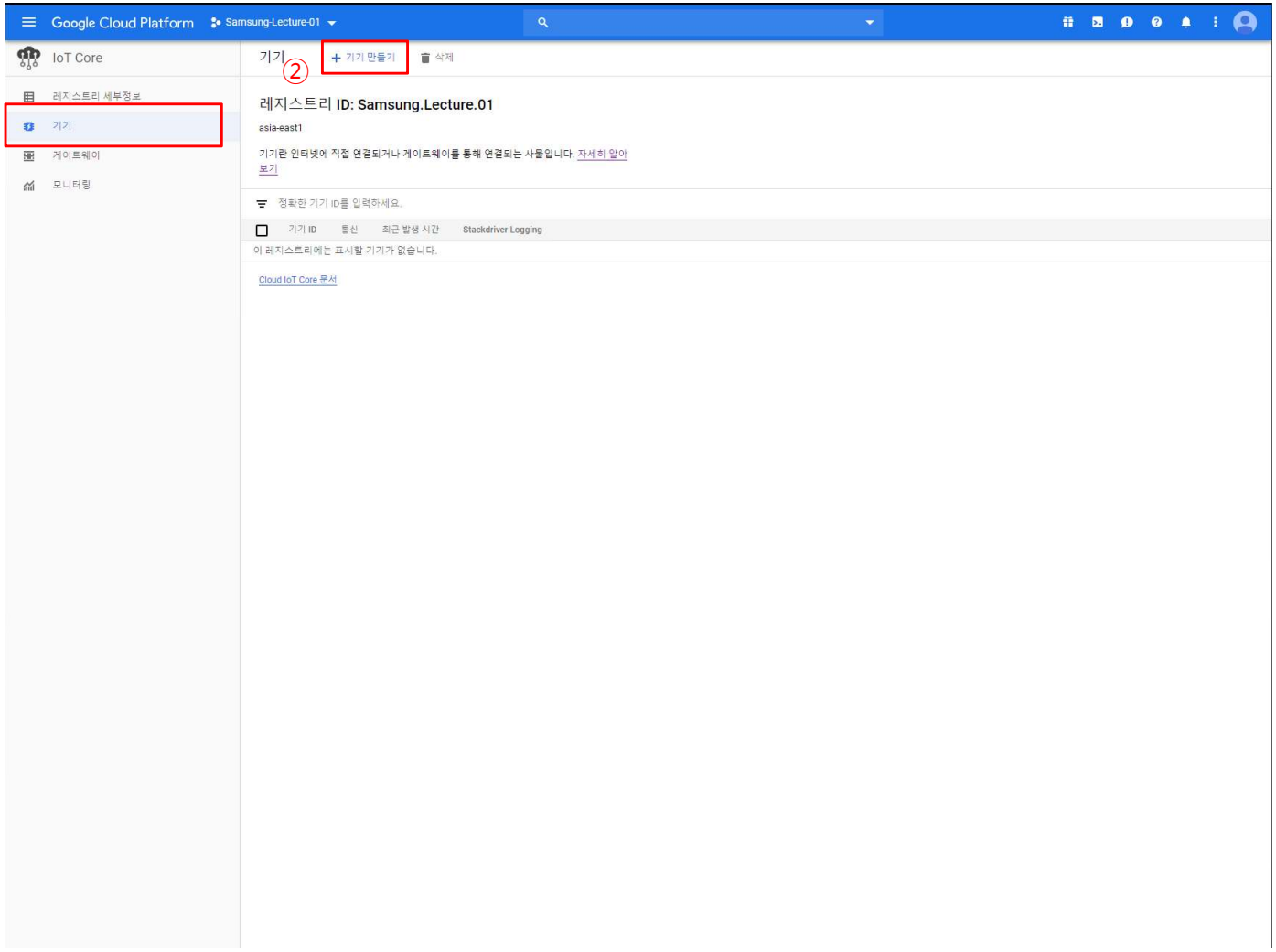
❖ ARTIK 053 세팅 - 프로젝트 빌드 및 에러확인

① 프로젝트 빌드



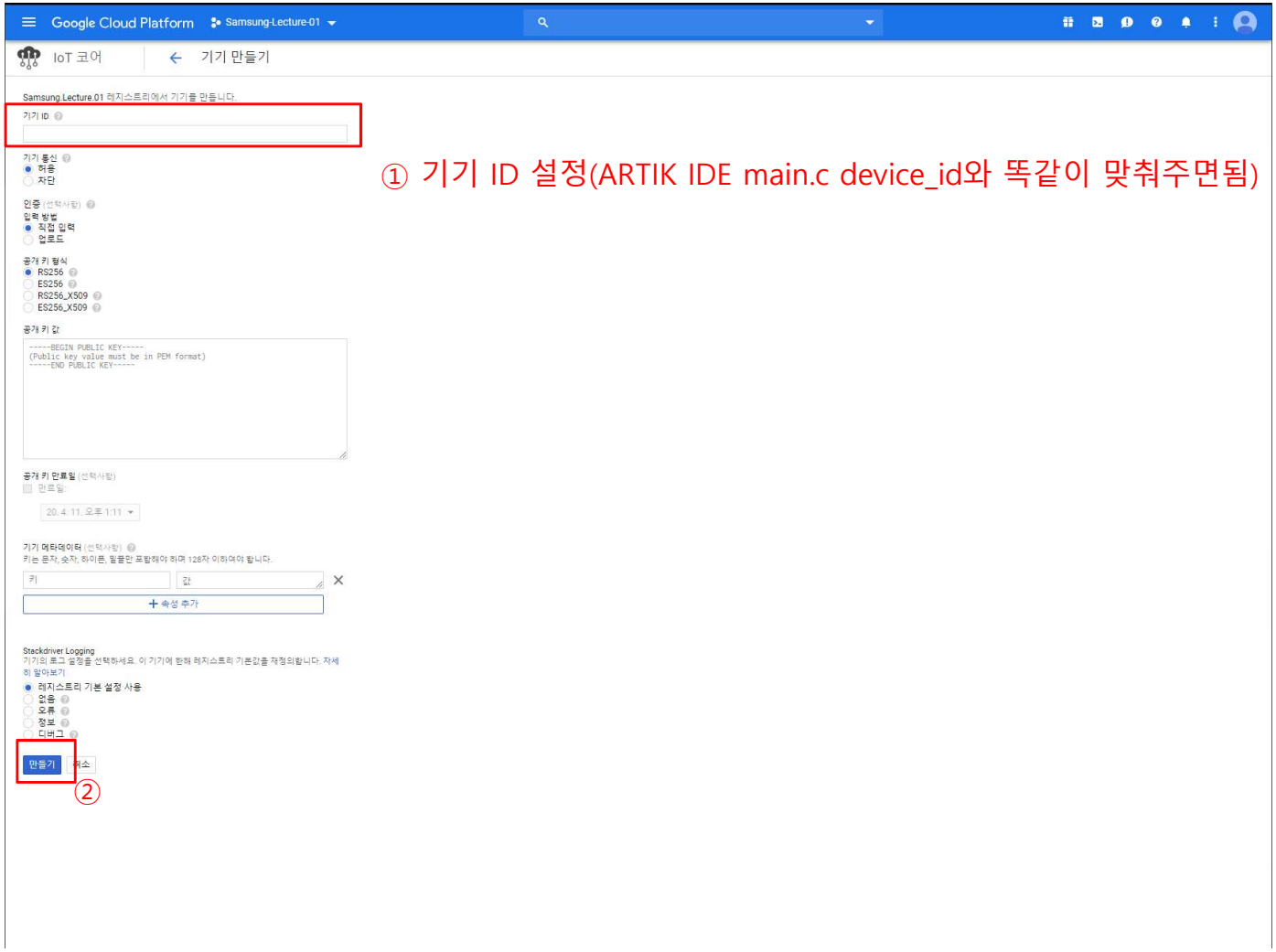
디바이스 만들기 -ARTIK 053

❖ IOT 코어 기기 등록 - 기기 만들기 클릭



디바이스 만들기 -ARTIK 053

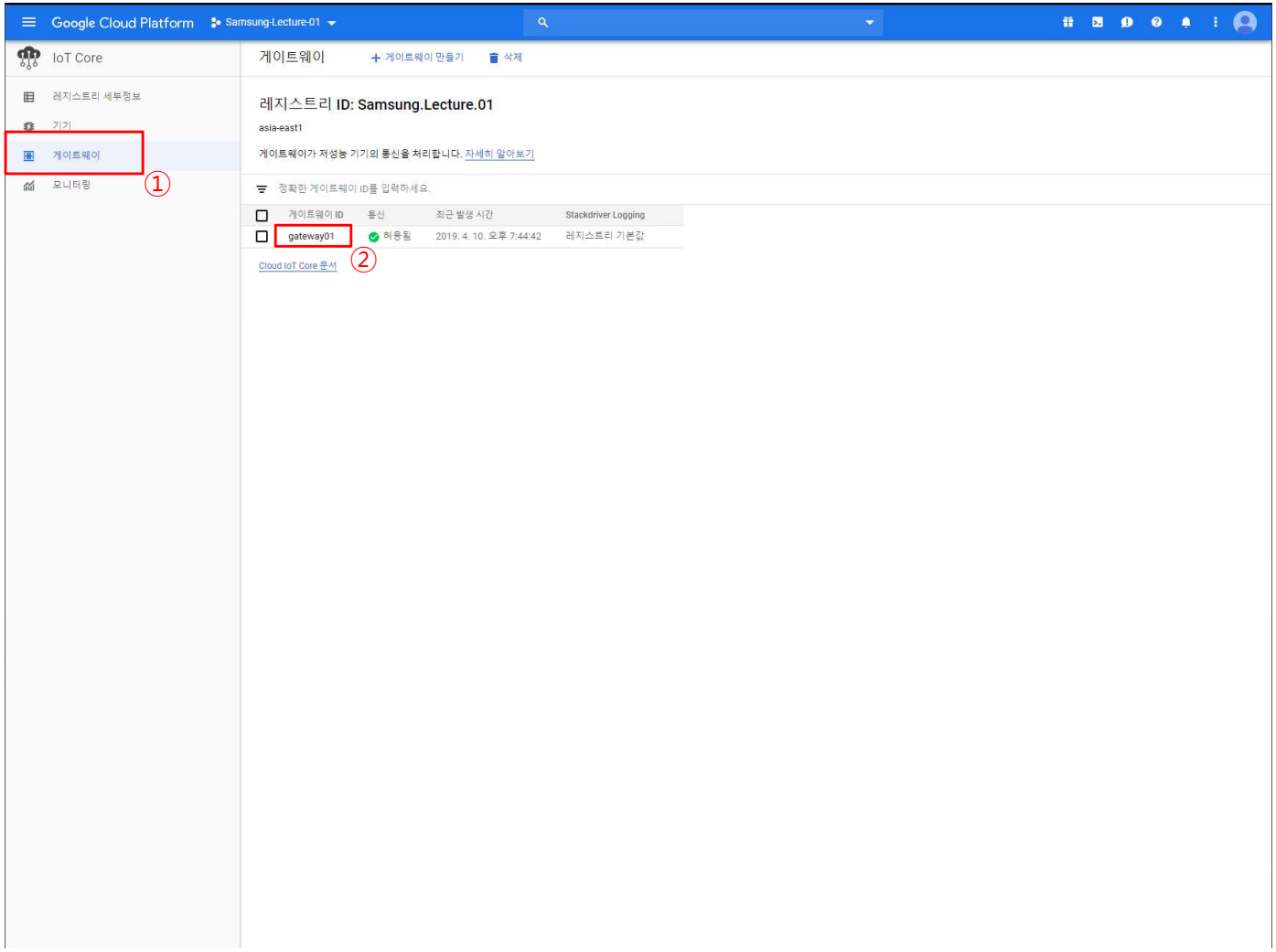
❖ IOT 코어 기기 등록 - 디바이스 ID 설정



① 기기 ID 설정(ARTIK IDE main.c device_id와 똑같이 맞춰주면됨)

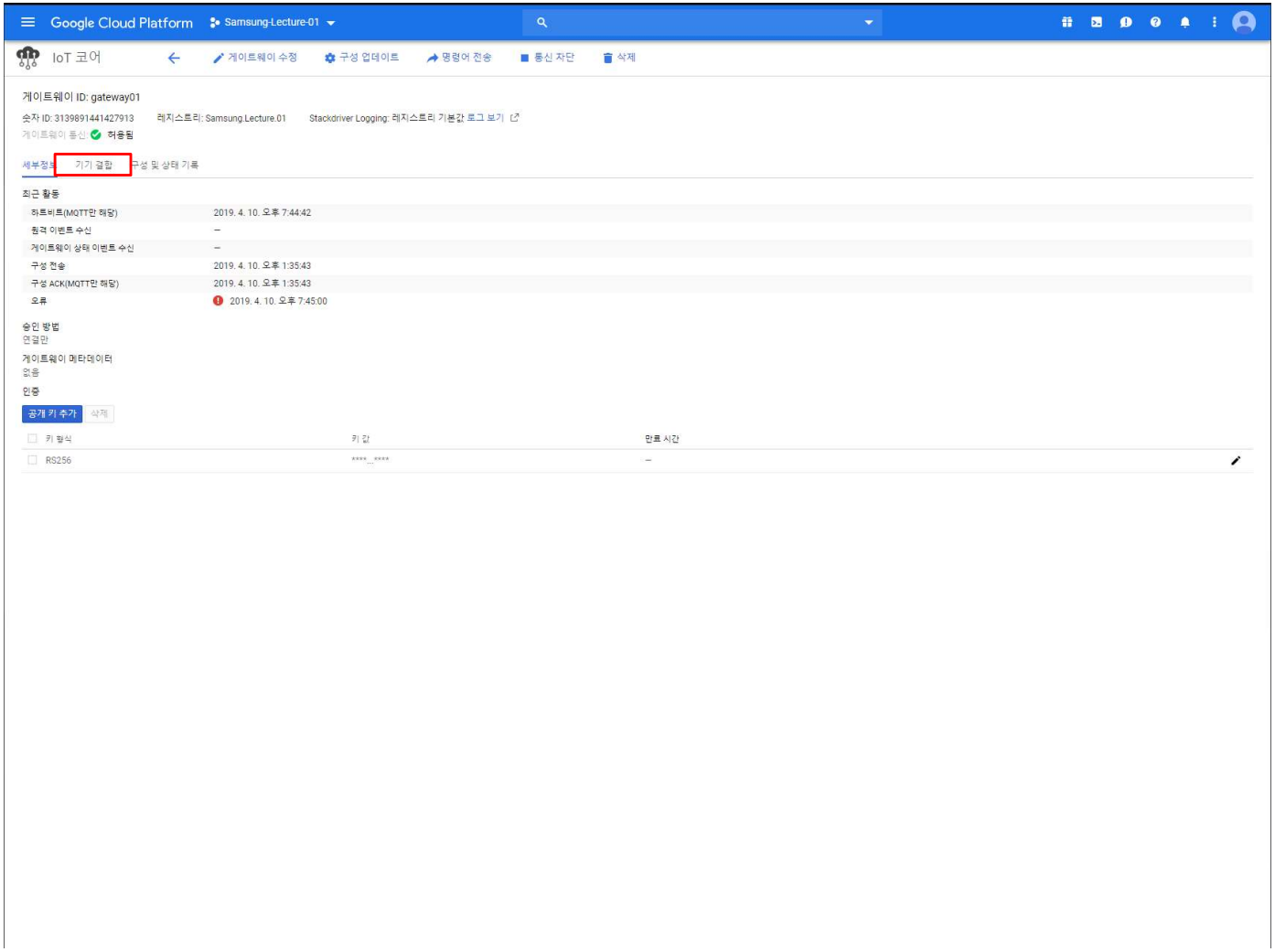
디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - 게이트웨이 ID 클릭



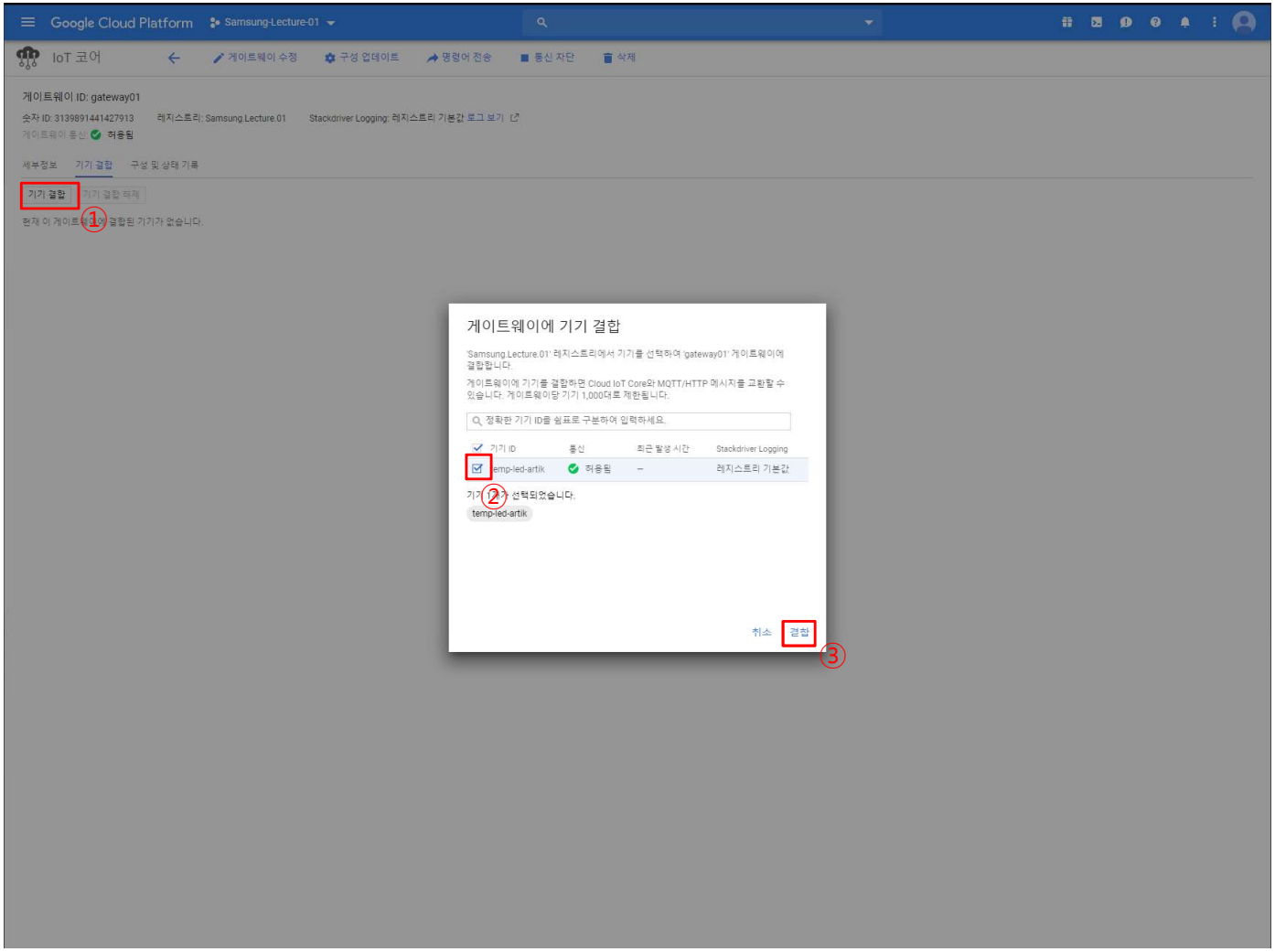
디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - 기기결합 클릭



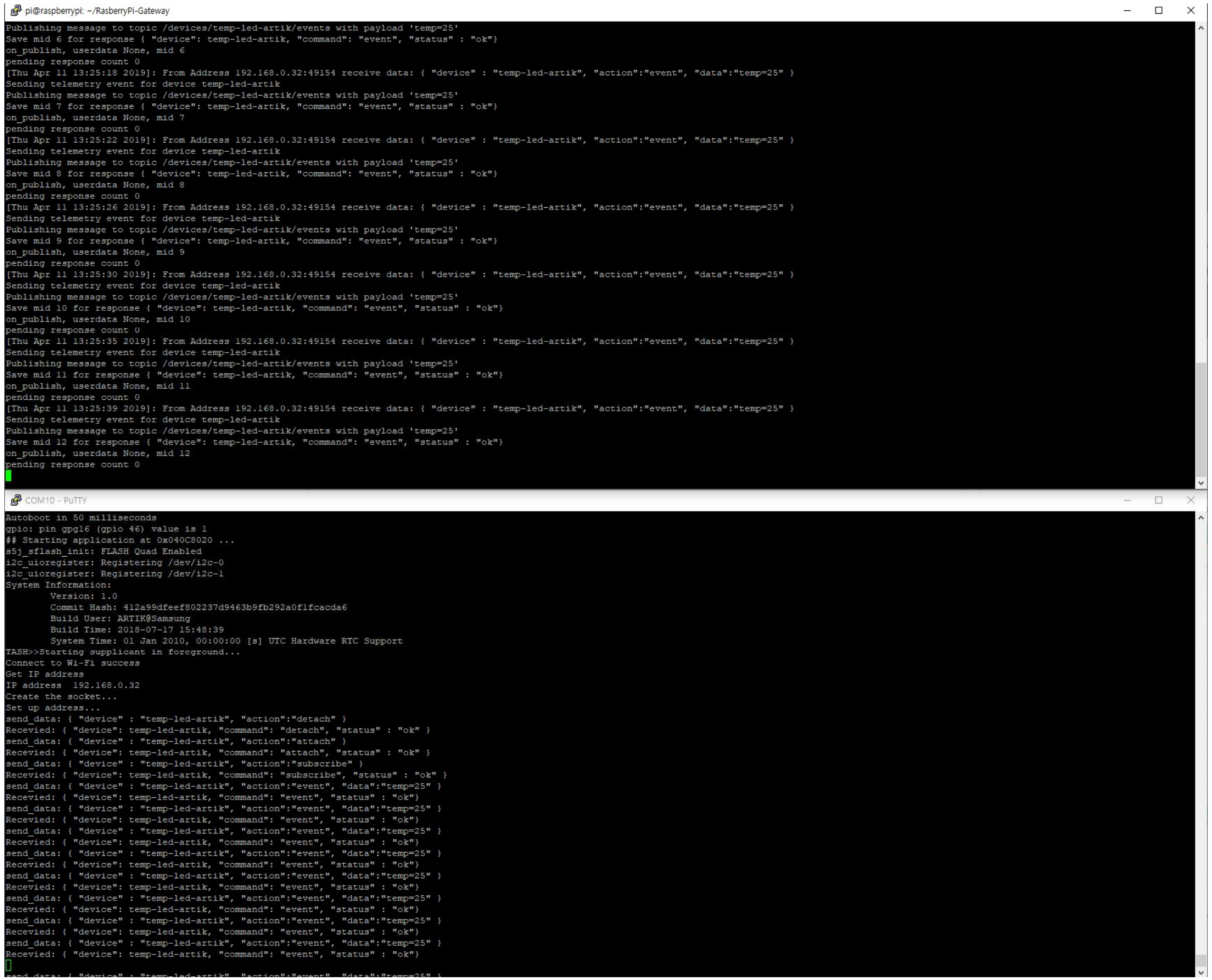
디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - Device ID 선택 및 결합



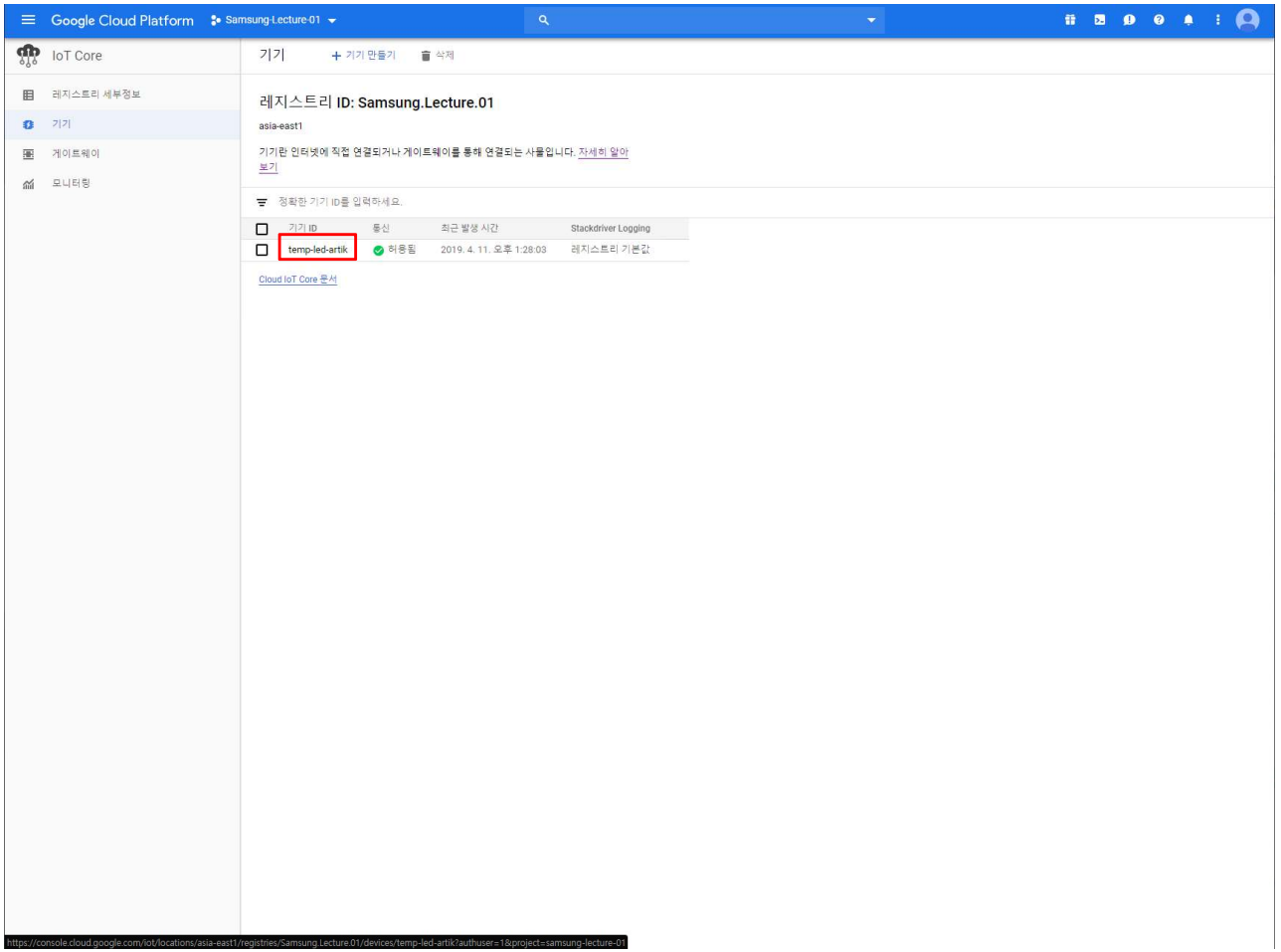
디바이스 만들기 -ARTIK 053

❖ 게이트웨이와 디바이스 결합 - Flash Project 후 연결확인



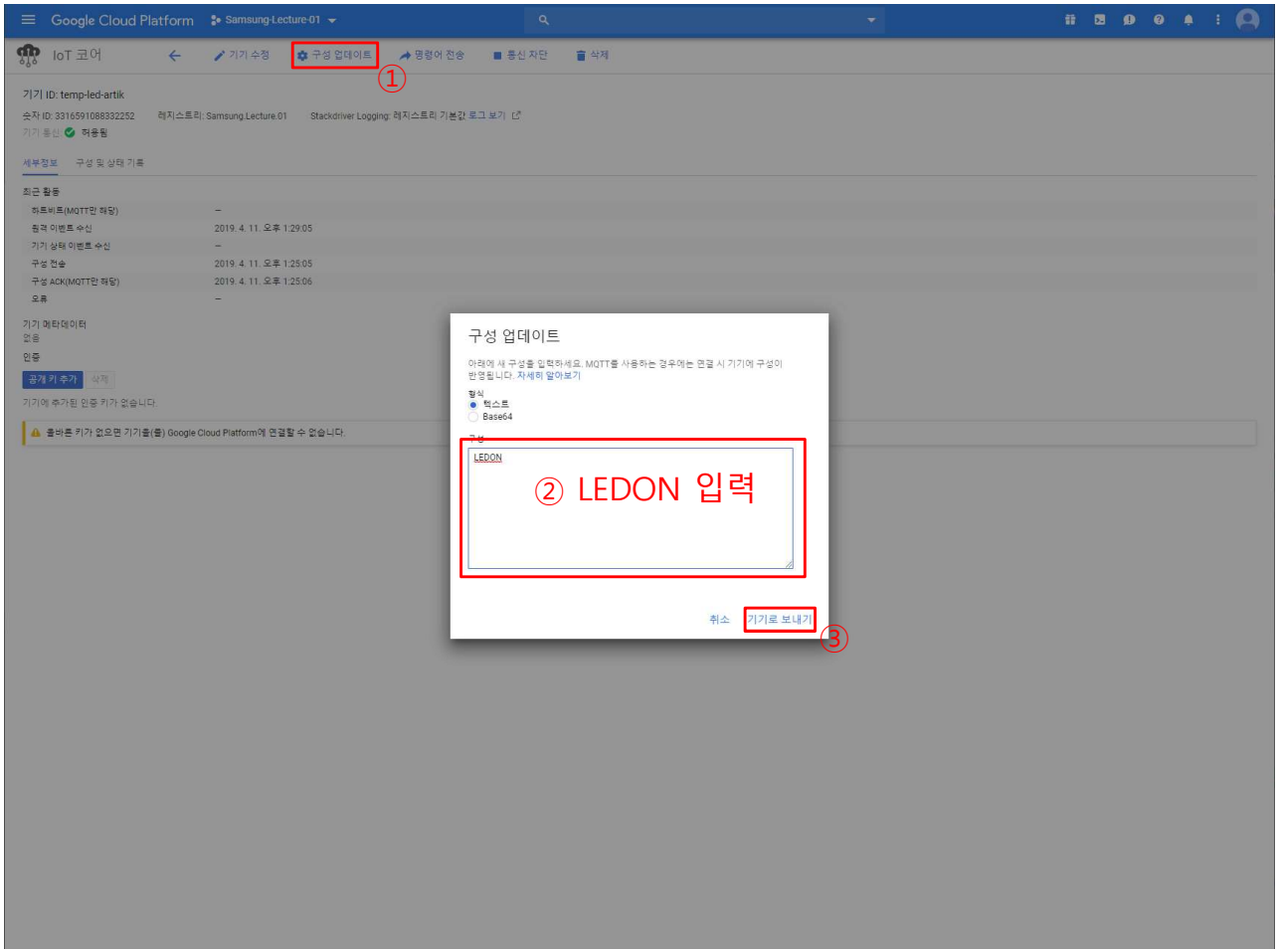
디바이스 만들기 -ARTIK 053

❖ 디바이스 구성 업데이트 테스트 - Device ID 클릭



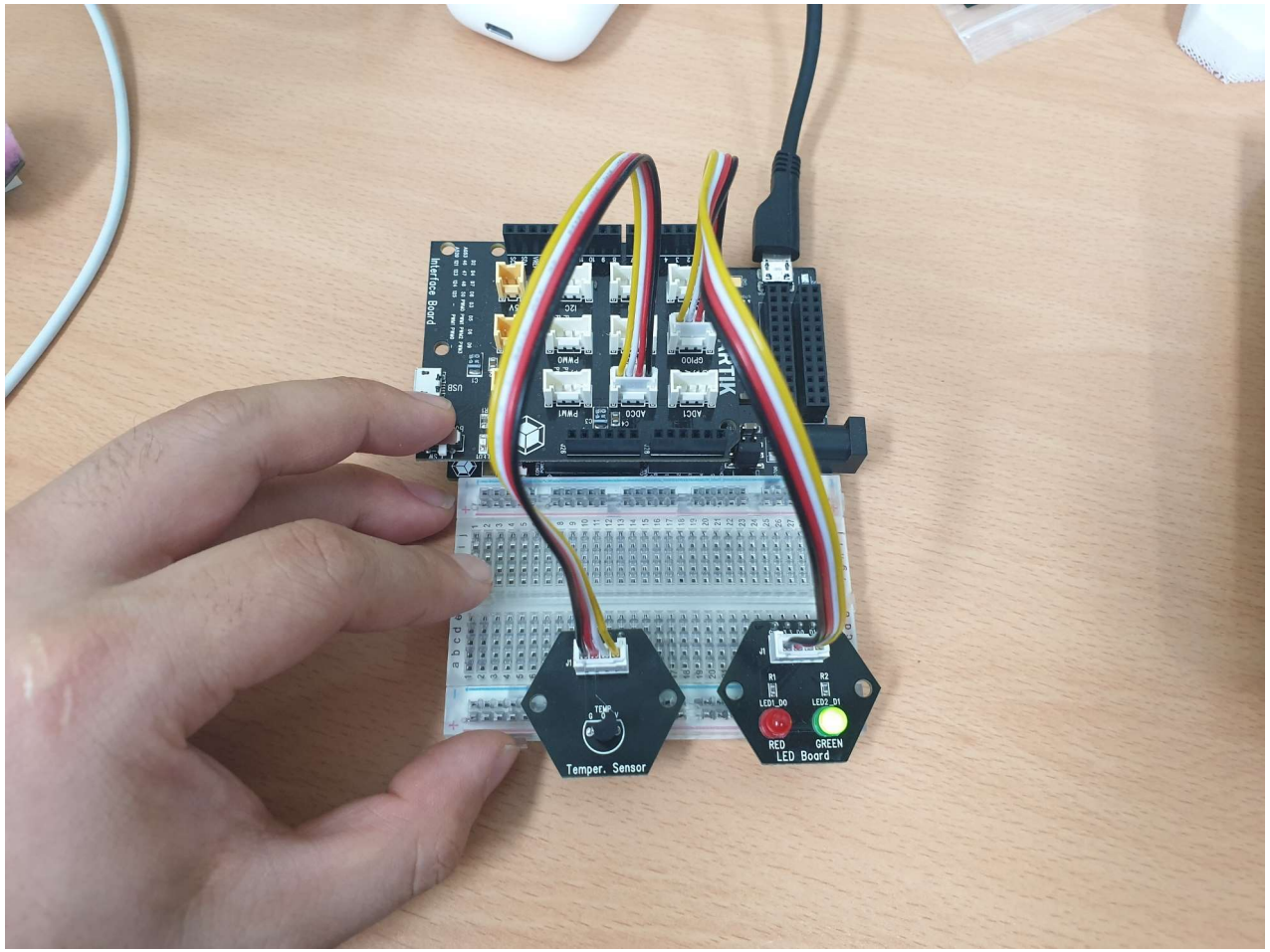
디바이스 만들기 -ARTIK 053

❖ 디바이스 구성 업데이트 테스트 - LEDON을 기기로 보내기



디바이스 만들기 -ARTIK 053

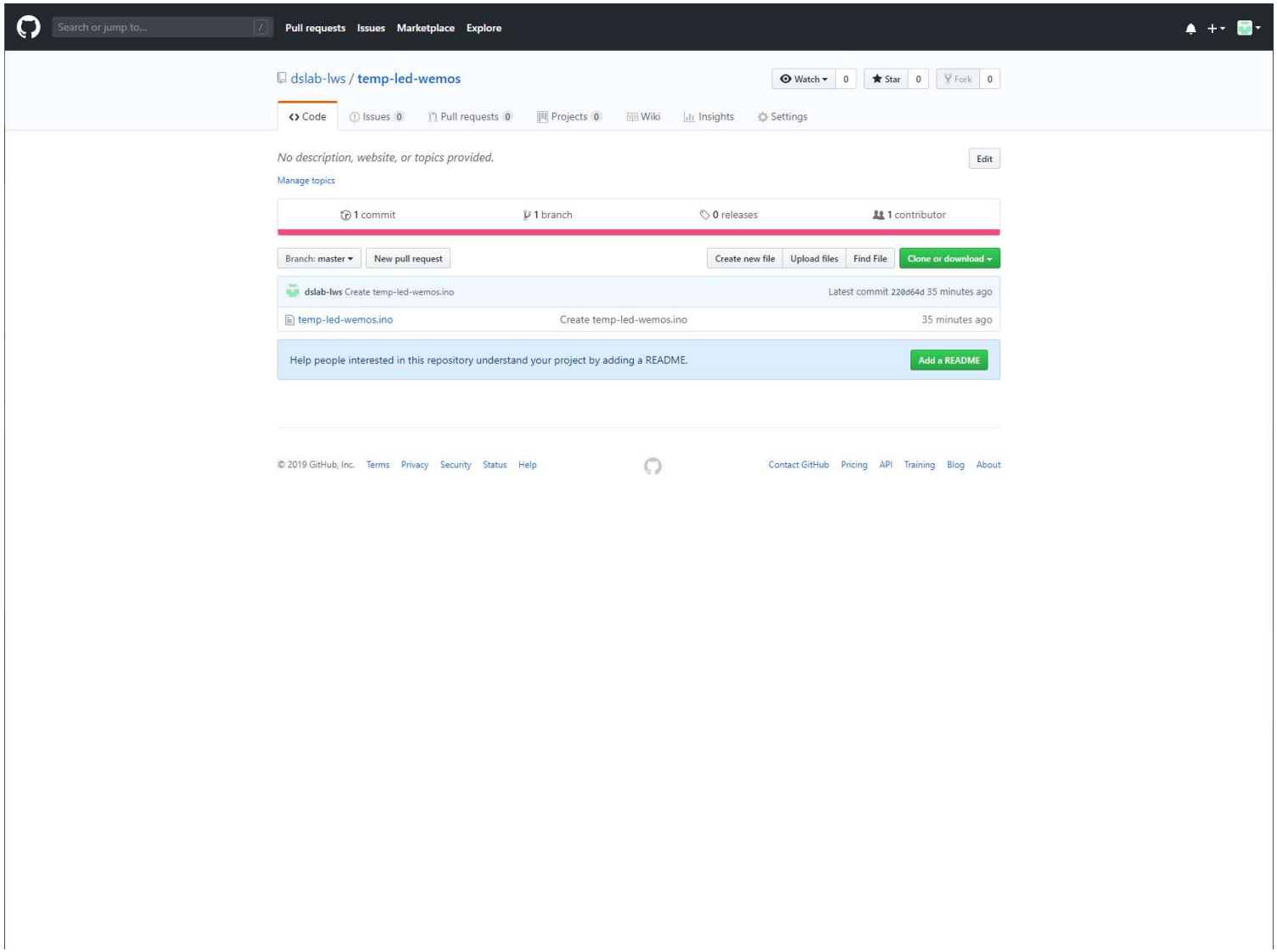
❖ 디바이스 구성 업데이트 테스트 - D4번 LED가 켜지는지 확인



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

temp-led-wemos | 아두이노 1.8.9

파일 편집 스케치 툴 도움말

temp-led-wemos

```
#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 *msg_device = "{ \"device\" : \"";
  char *msg_action = "\", \"action\":\"";
```

WeMos D1 R1, 160 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 921600 on COM12

디바이스 만들기 - Wemos D1 R1

❖ Wemos D1 R1 세팅 - 소스코드 수정

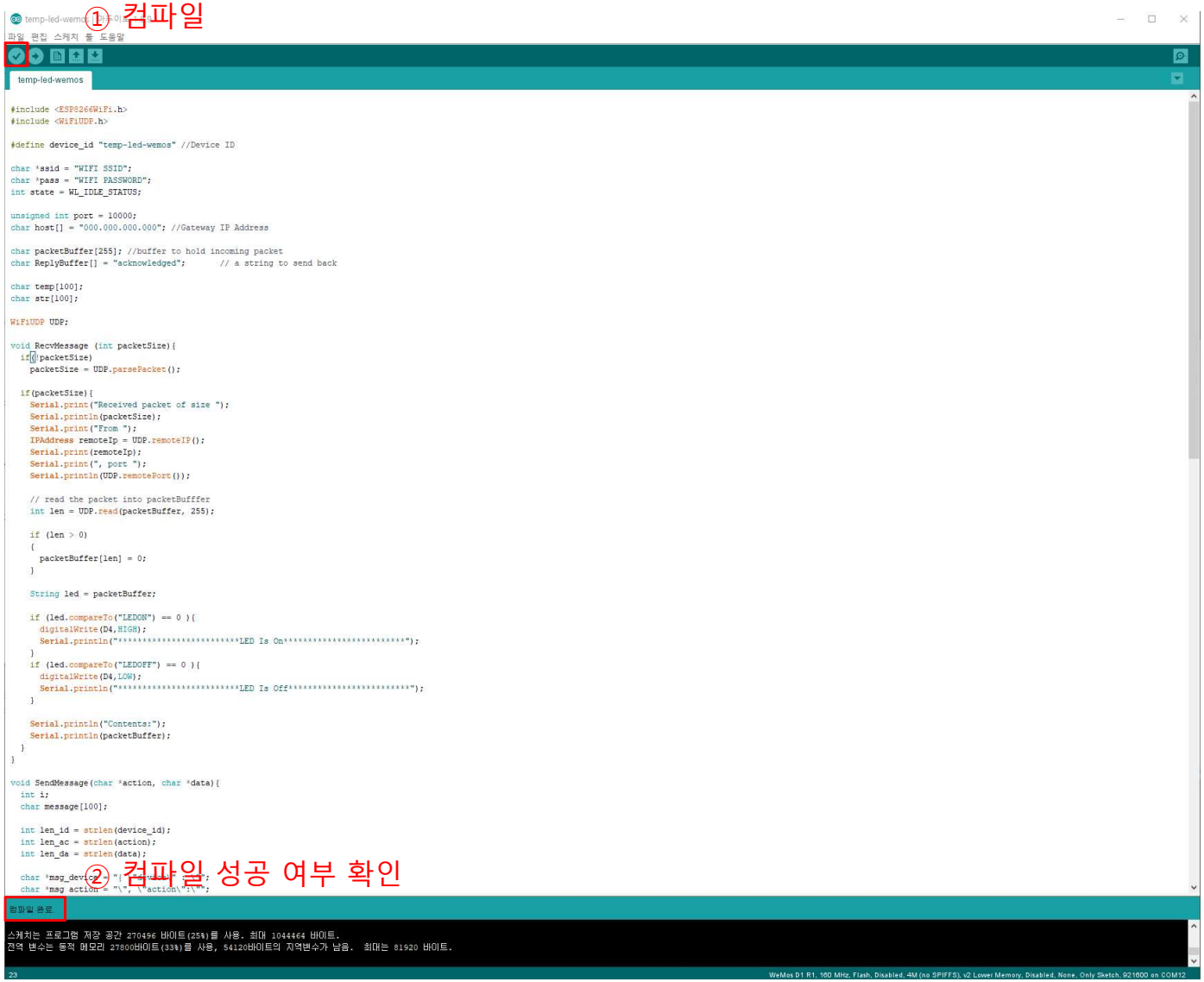
```
#define device_id "temp-led-wemos" //Device ID

char *ssid = "WIFI SSID";
char *pass = "WIFI PASSWORD";
int state = WL_IDLE_STATUS;
unsigned int port = 10000;
char host[] = "000.000.000.000"; //Gateway IP Address
```

와이파이 이름과 비밀번호 입력

Gateway IP 입력

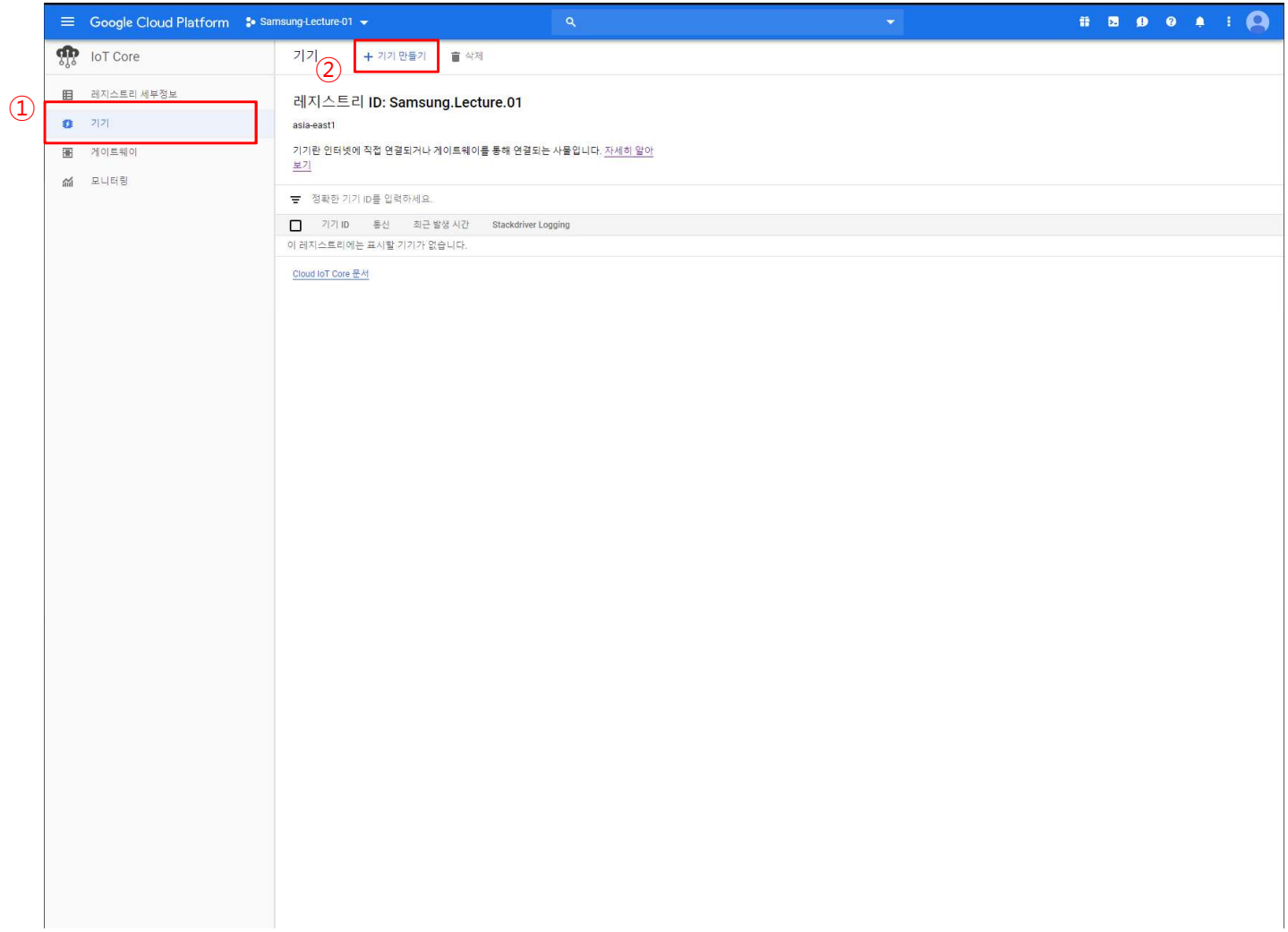
디바이스 만들기 - Wemos D1 R1

❖ Wemos D1 R1 세팅 - 소스코드 컴파일 및 에러확인



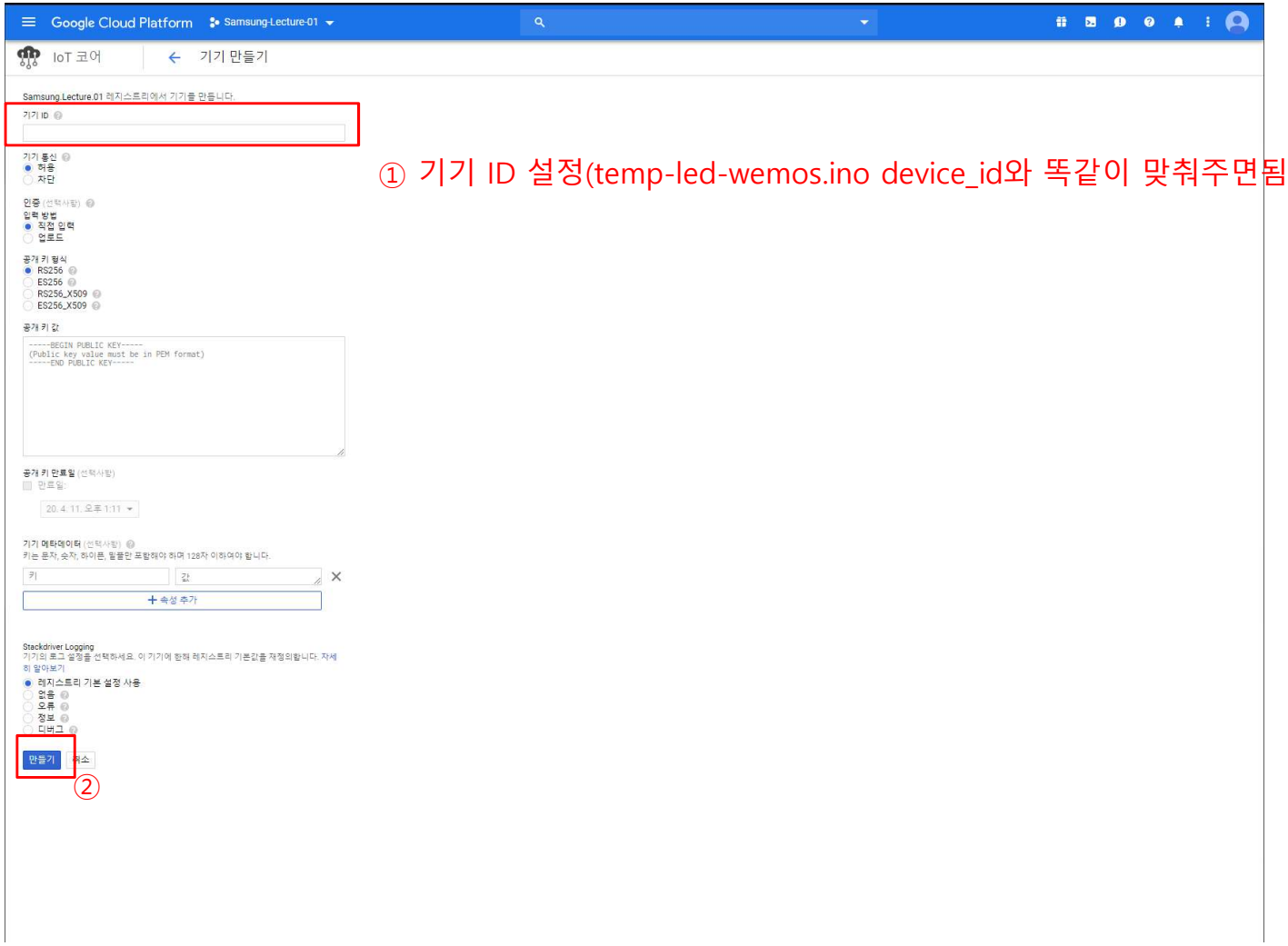
디바이스 만들기 - Wemos D1 R1

❖ IOT 코어 기기 등록 - 기기 만들기 클릭



디바이스 만들기 - Wemos D1 R1

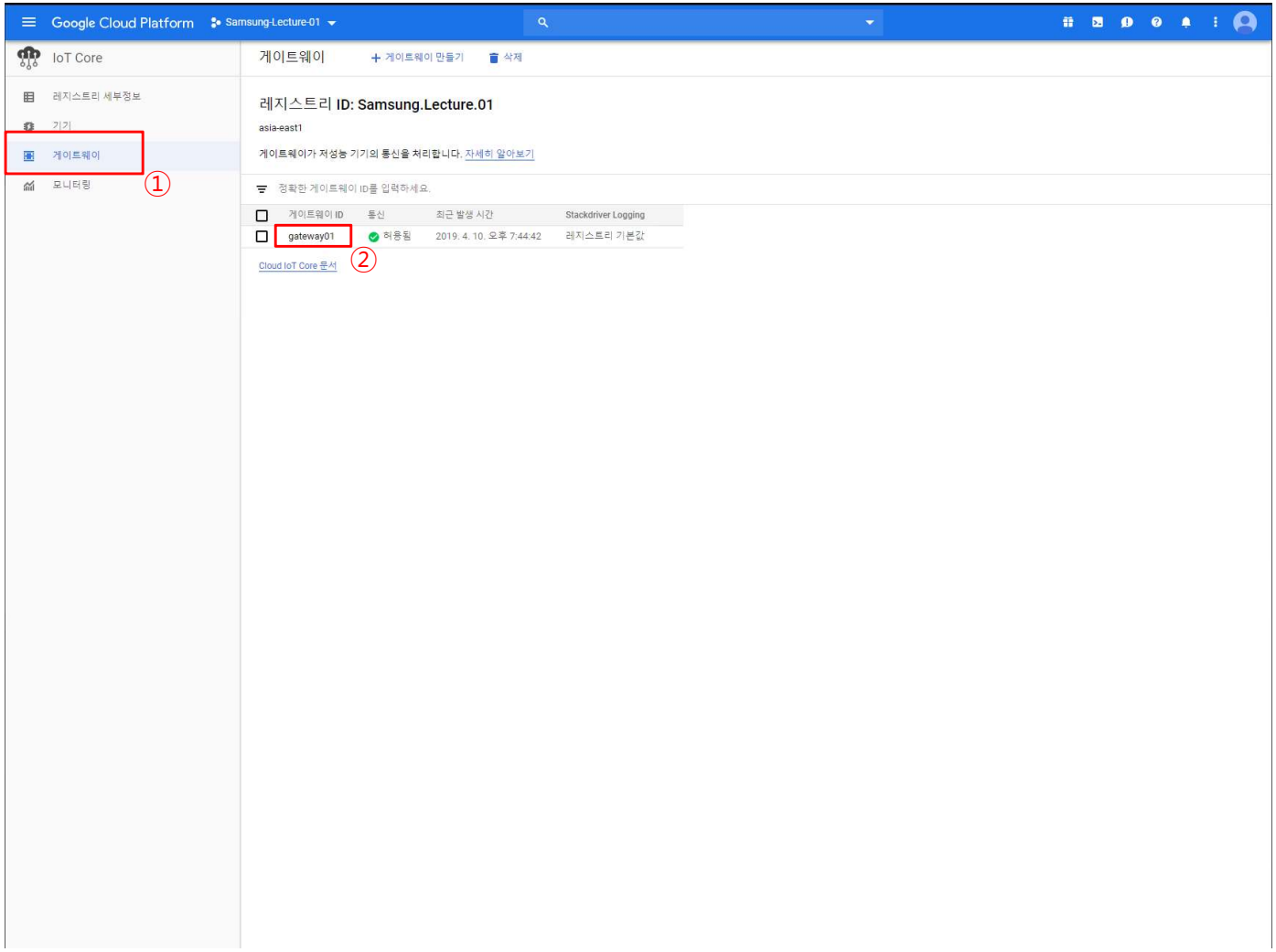
❖ IOT 코어 기기 등록 - 디바이스 ID 설정



① 기기 ID 설정(temp-led-wemos.ino device_id와 똑같이 맞춰주면됨)

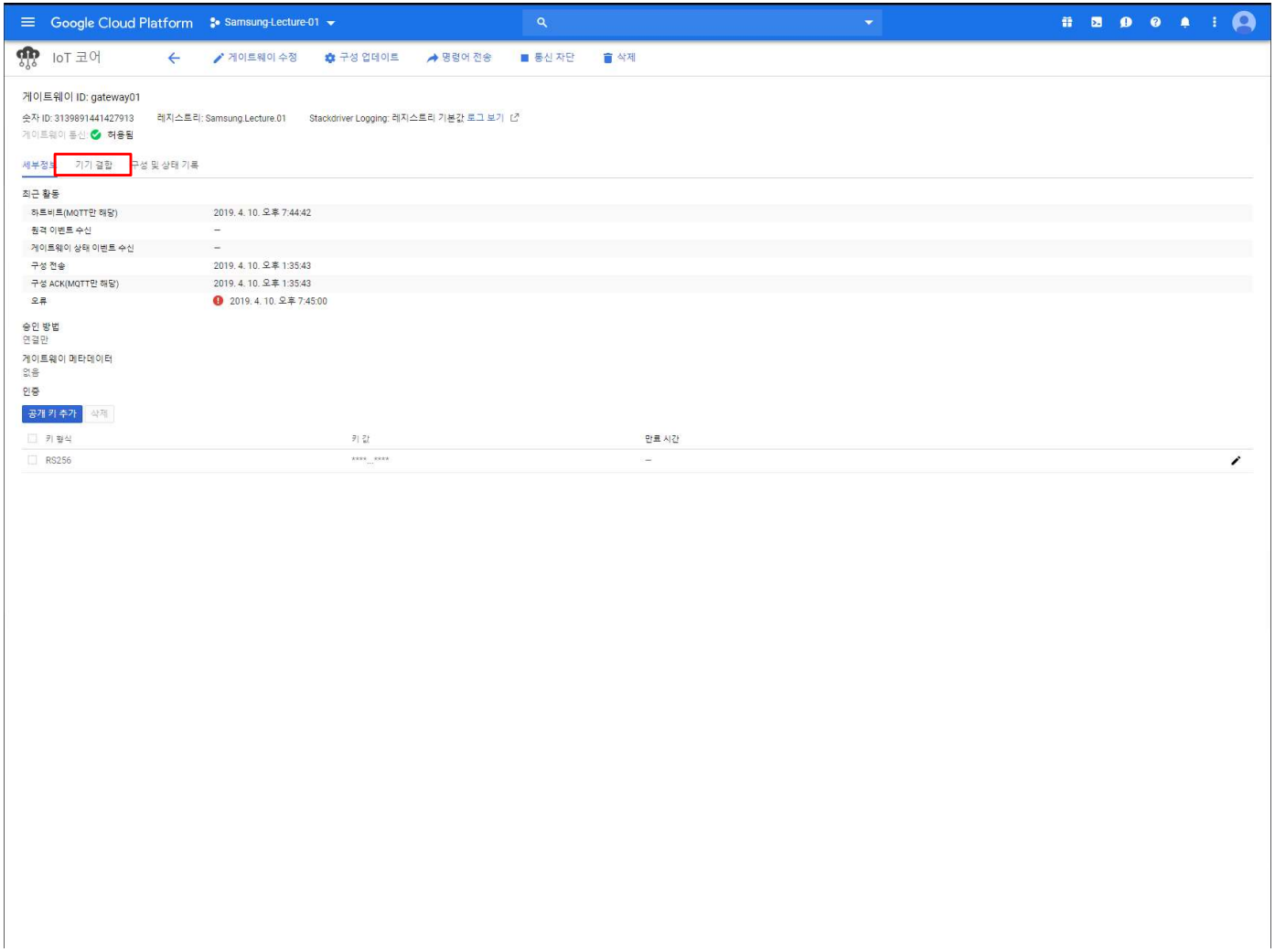
디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - 게이트웨이 ID 클릭



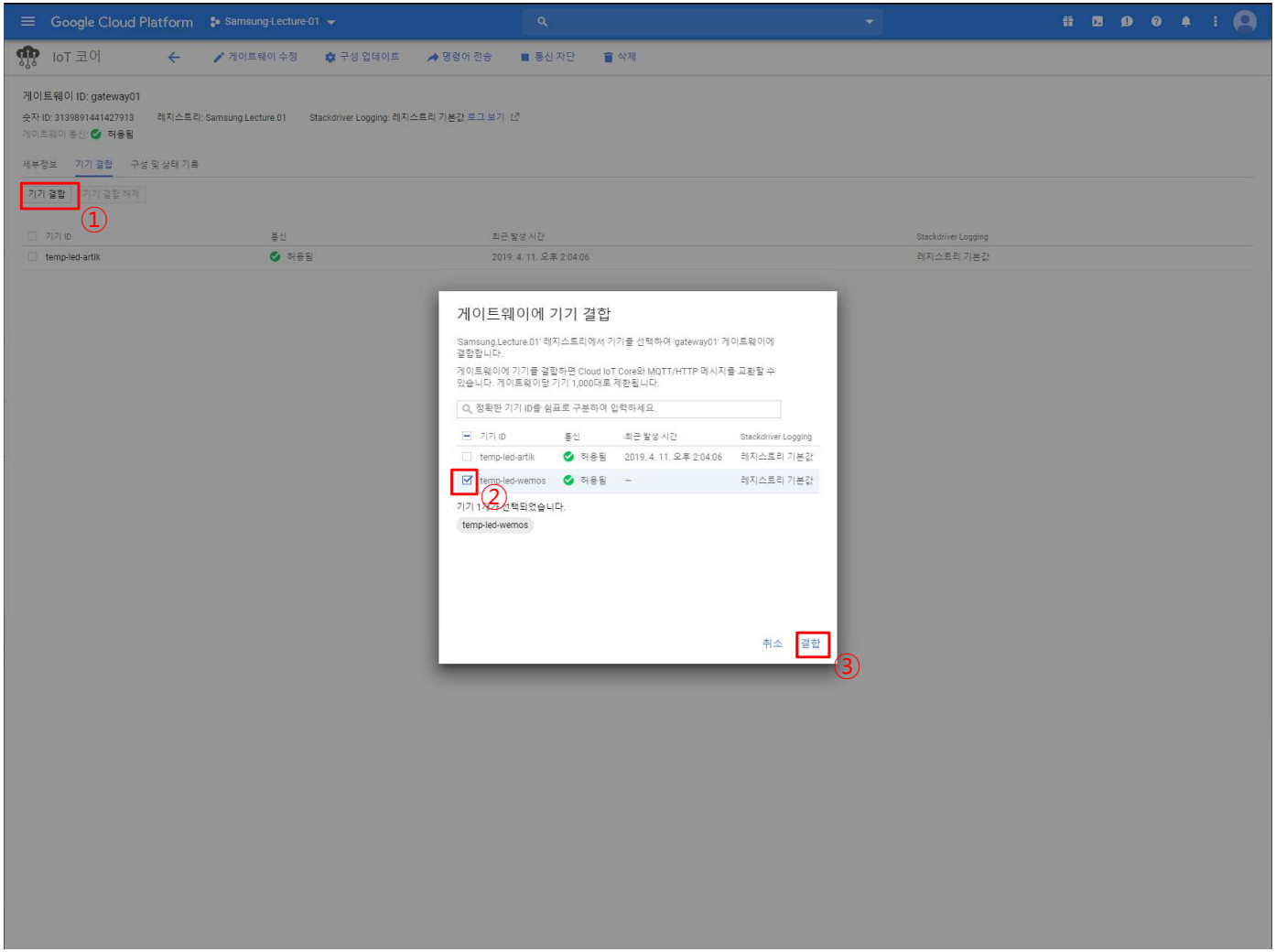
디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - 기기결합 클릭



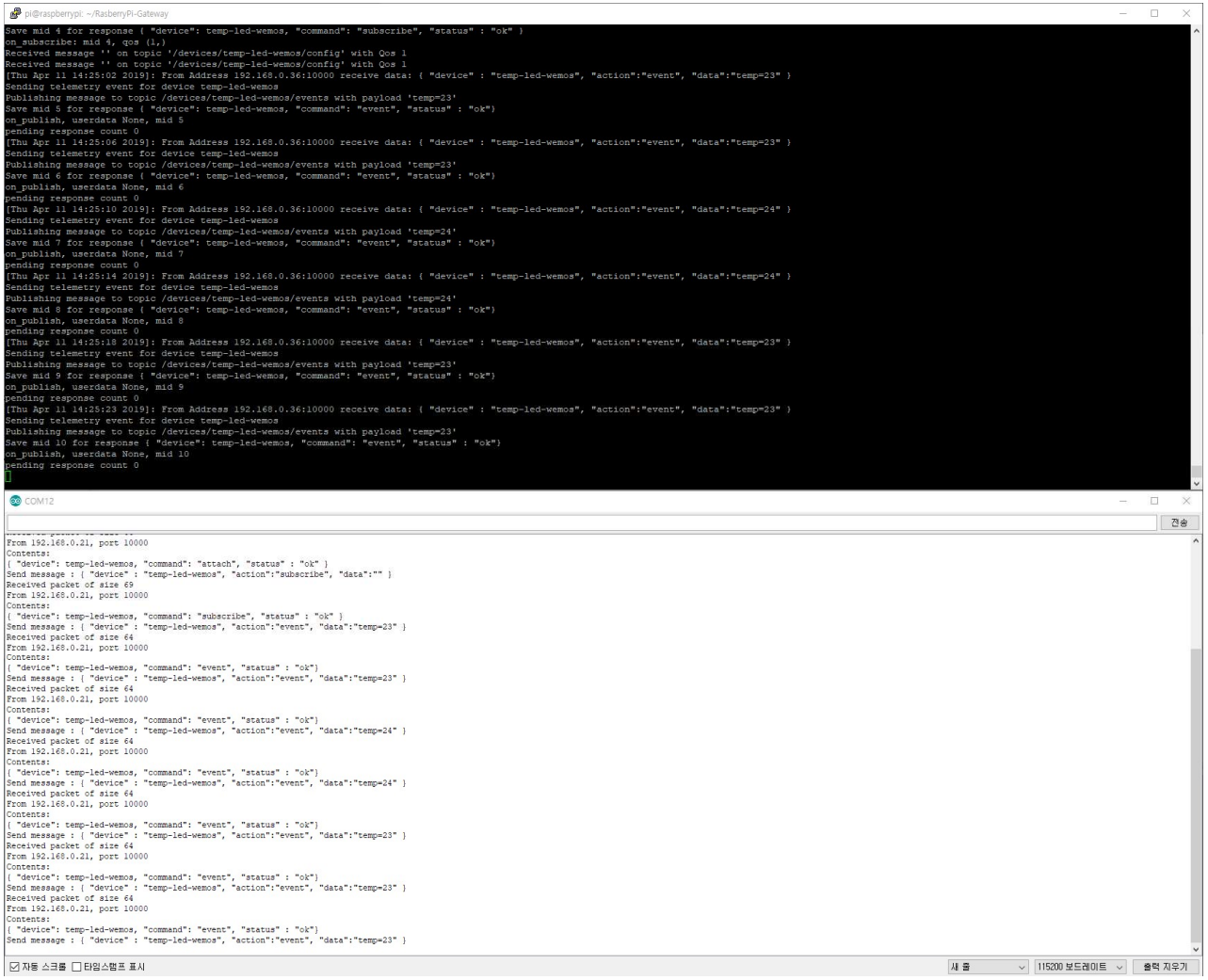
디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - Device ID 선택 및 결합



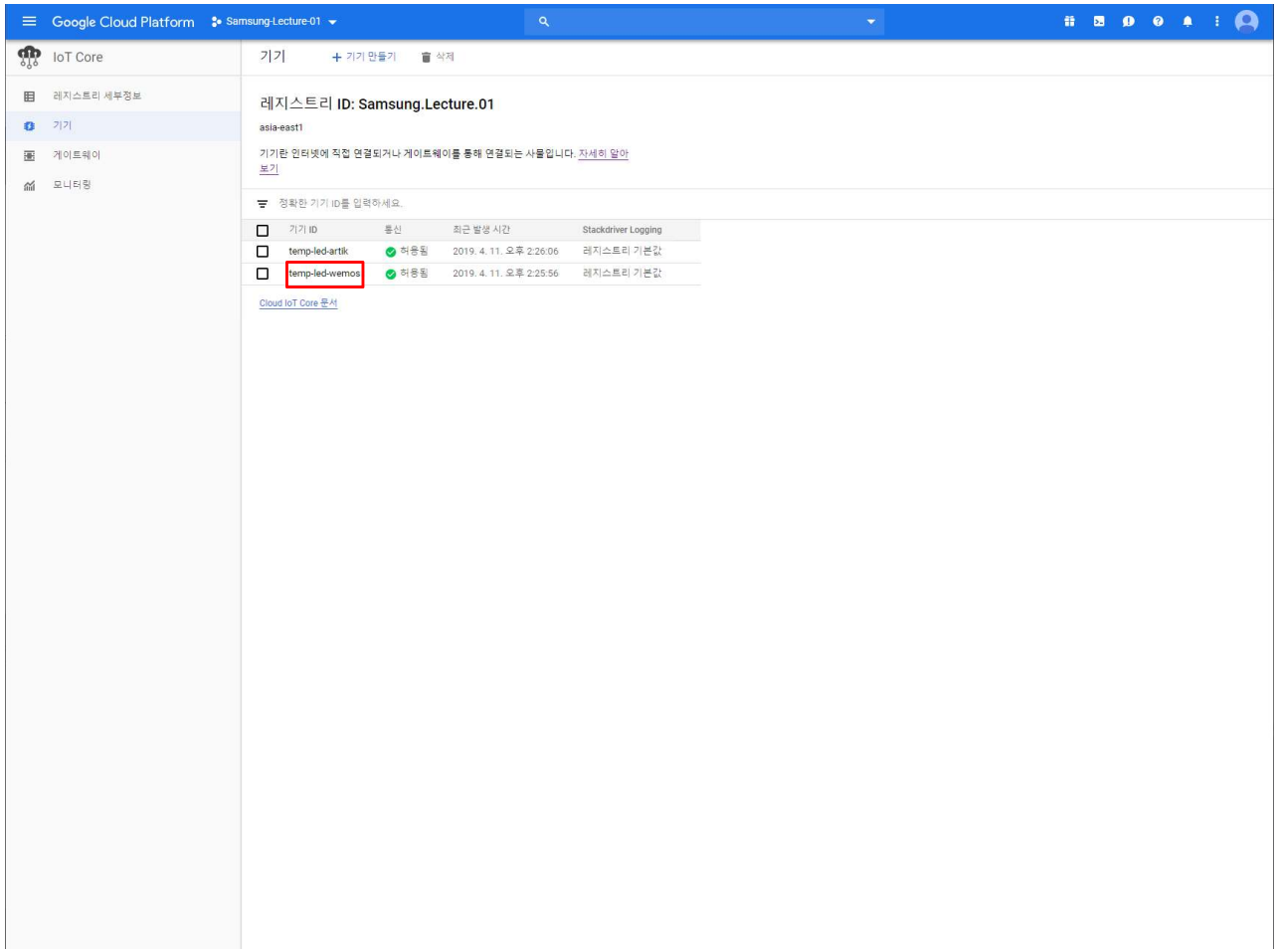
디바이스 만들기 - Wemos D1 R1

❖ 게이트웨이와 디바이스 결합 - Upload Project 후 연결확인



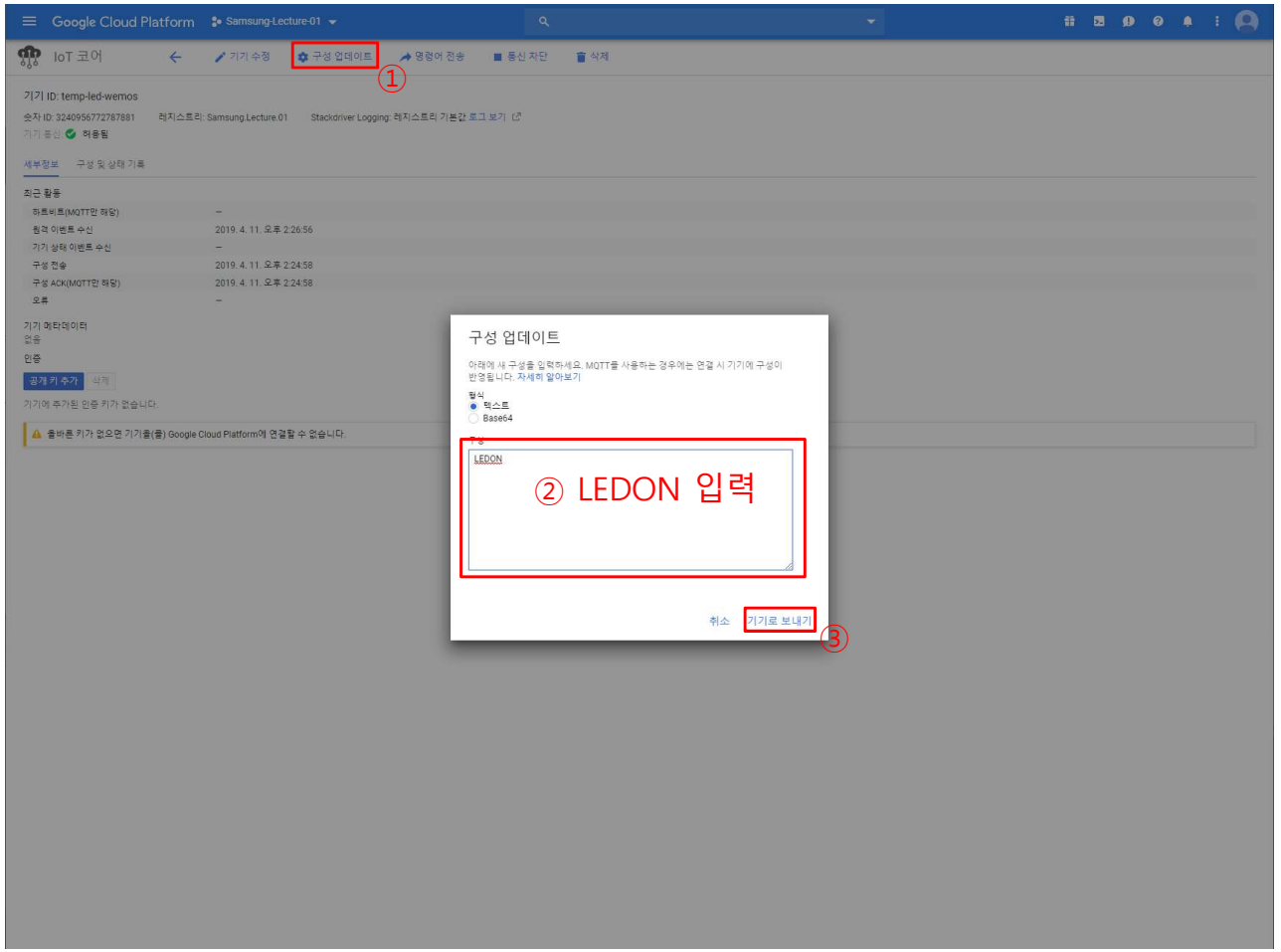
디바이스 만들기 - Wemos D1 R1

❖ 디바이스 구성 업데이트 테스트 - Device ID 클릭



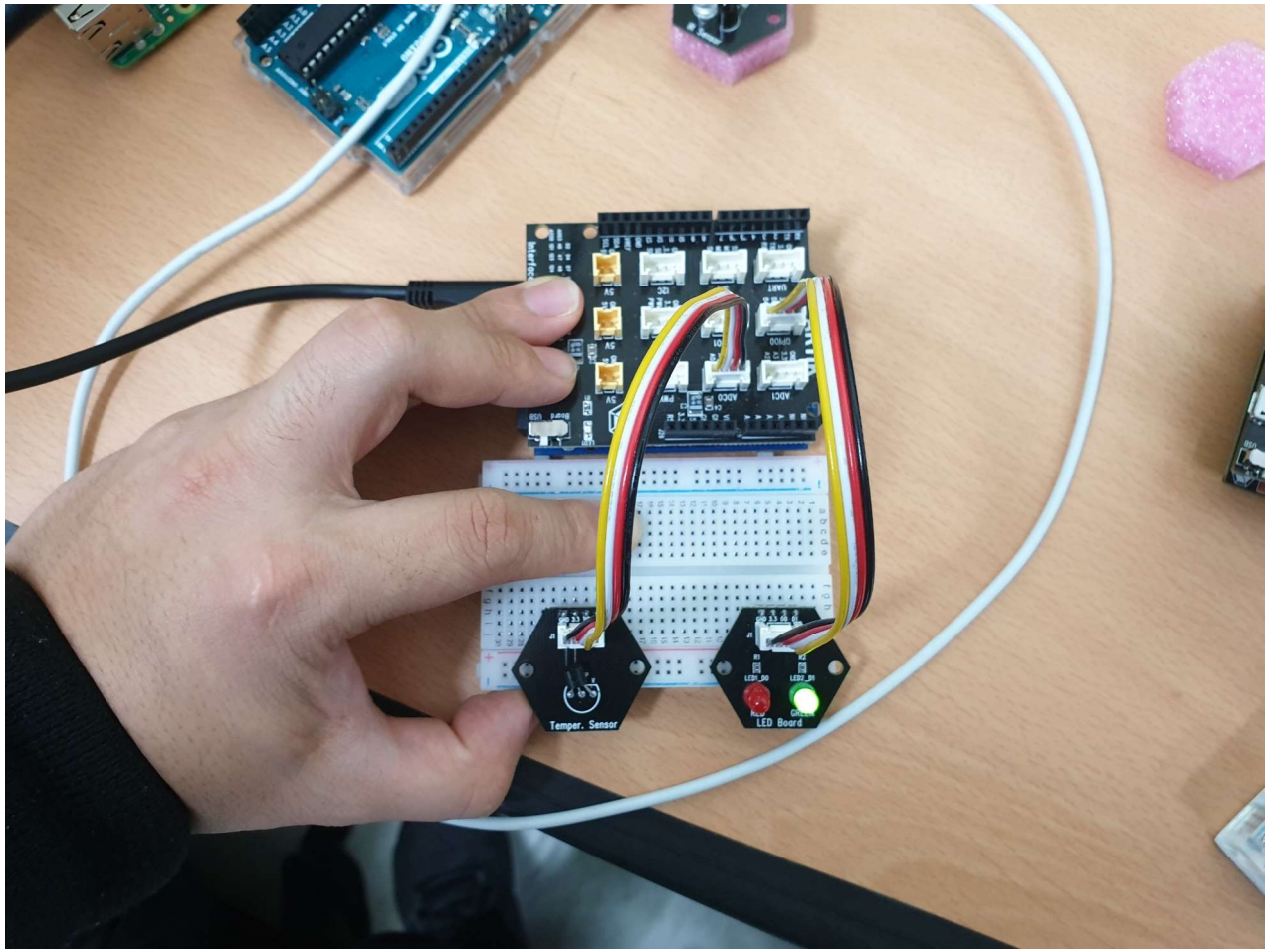
디바이스 만들기 - Wemos D1 R1

❖ 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

Client

서버에 있는 개인 데이터 요청

Resource Server

서버야 캘린더에
오늘 일정
데이터좀 줘봐!!

Resource Owner

OAuth 2.0 Access Token 받기

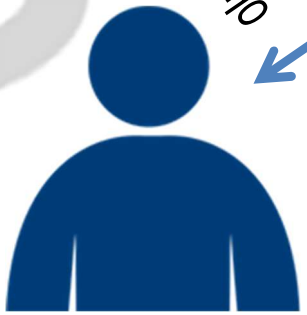
❖ Access Token을 받는 이유

Client

사용자한테
물어볼게!!

Resource Server

동의 및 로그인 요청



Resource Owner

데이터 넘겨줘도
괜찮아??

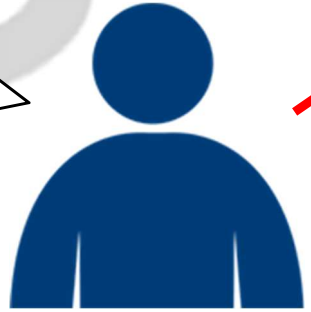
OAuth 2.0 Access Token 받기

❖ Access Token을 받는 이유

Client

Resource Server

데이터 넘겨줘!!



Resource Owner

동의 및 로그인

OAuth 2.0 Access Token 받기

❖ Access Token을 받는 이유

Client

데이터
넘겨주래!!

개인 데이터 전송

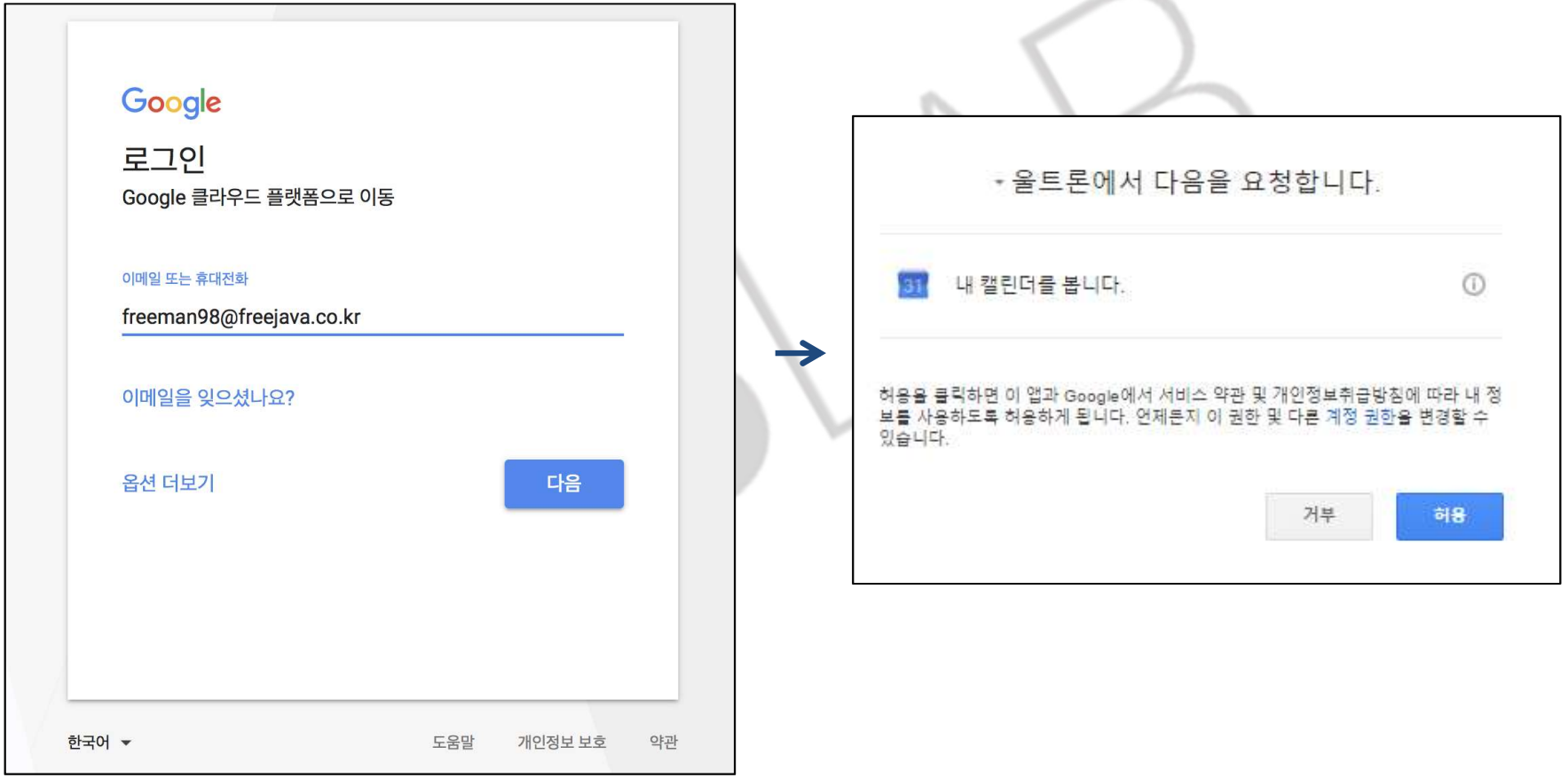
Resource Server

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

Client

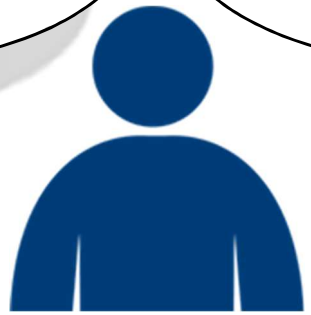
Access Token과 함께
서버에 있는 개인 데이터 요청

Access Token이 일치할 경우
데이터 전송

Resource Server

서버야
캘린더에 오늘
일정 데이터좀
줘봐!!

Access
Token이
있으니까
넘겨줄게!!



Resource Owner

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

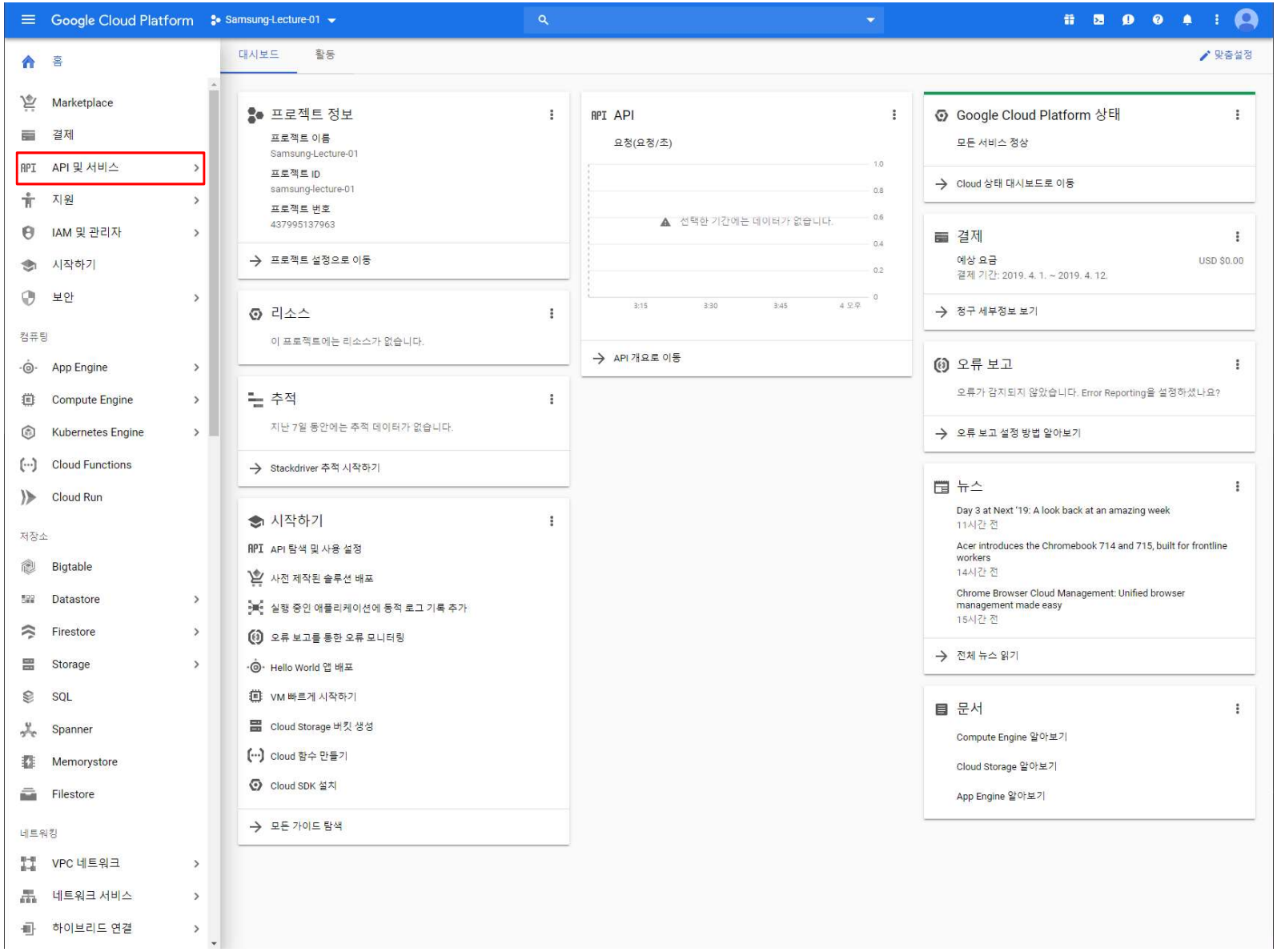
Client

Resource Server

Resource Owner

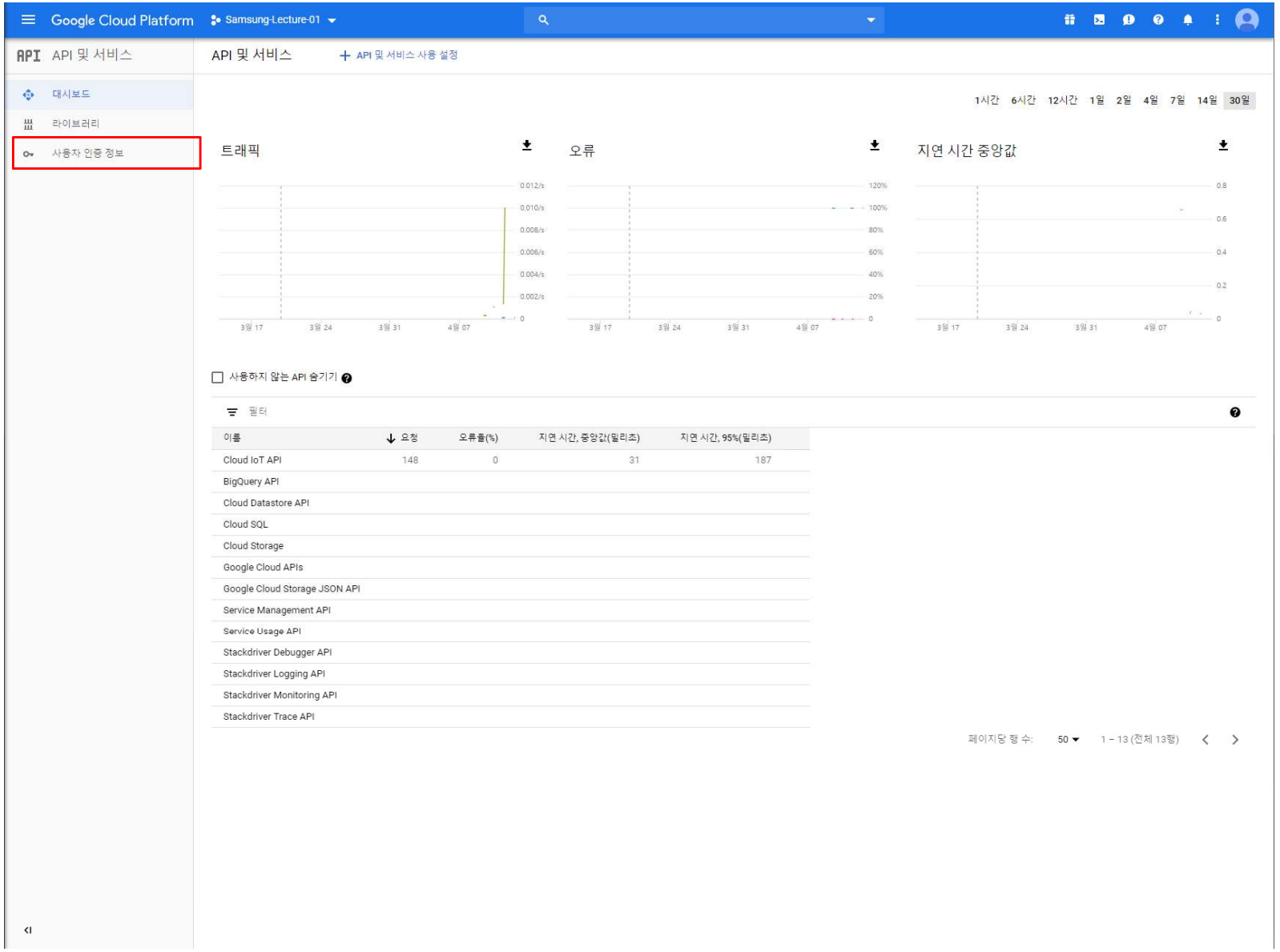
OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - API 및 서비스 클릭



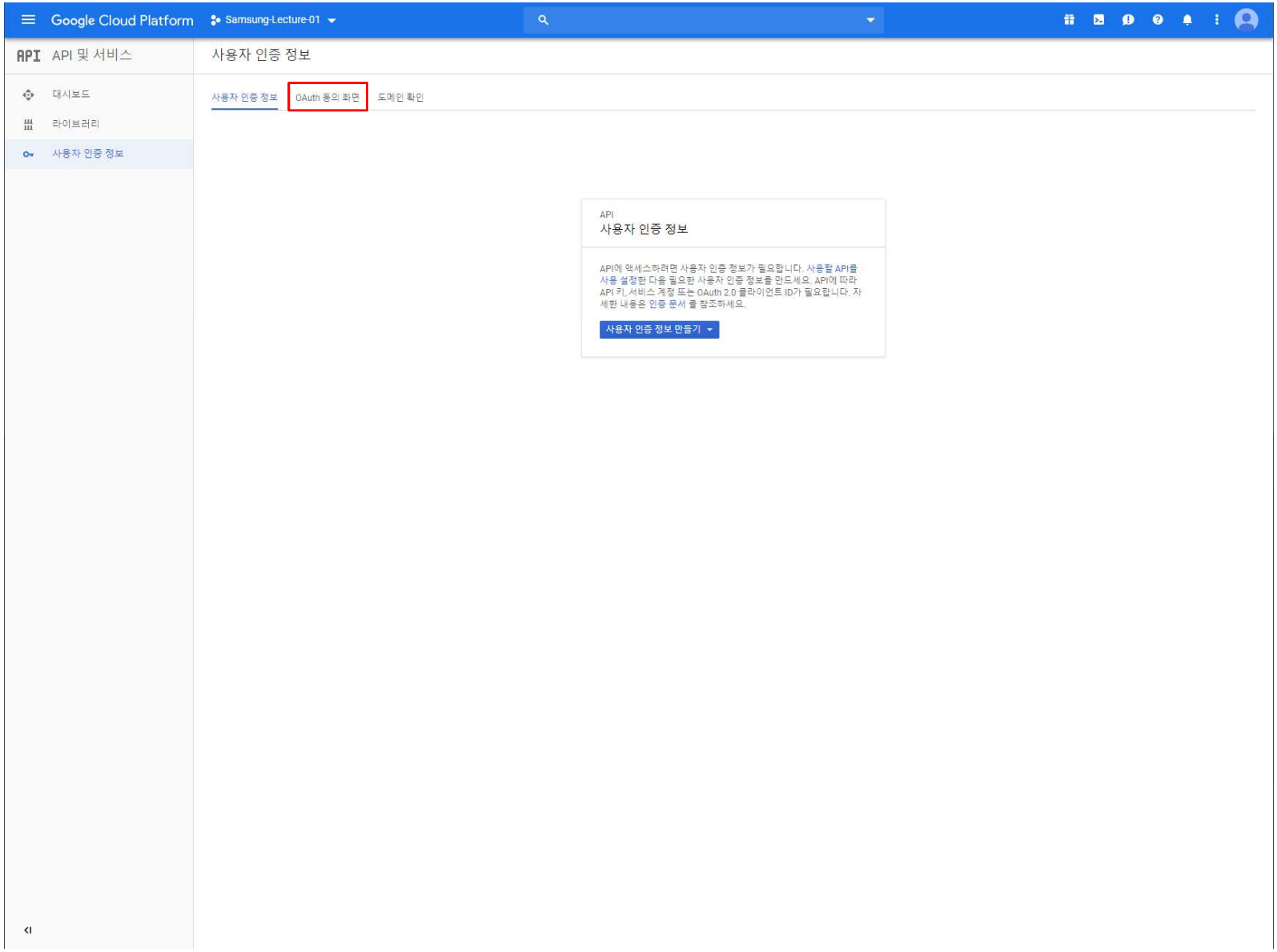
OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 사용자 인증정보 클릭



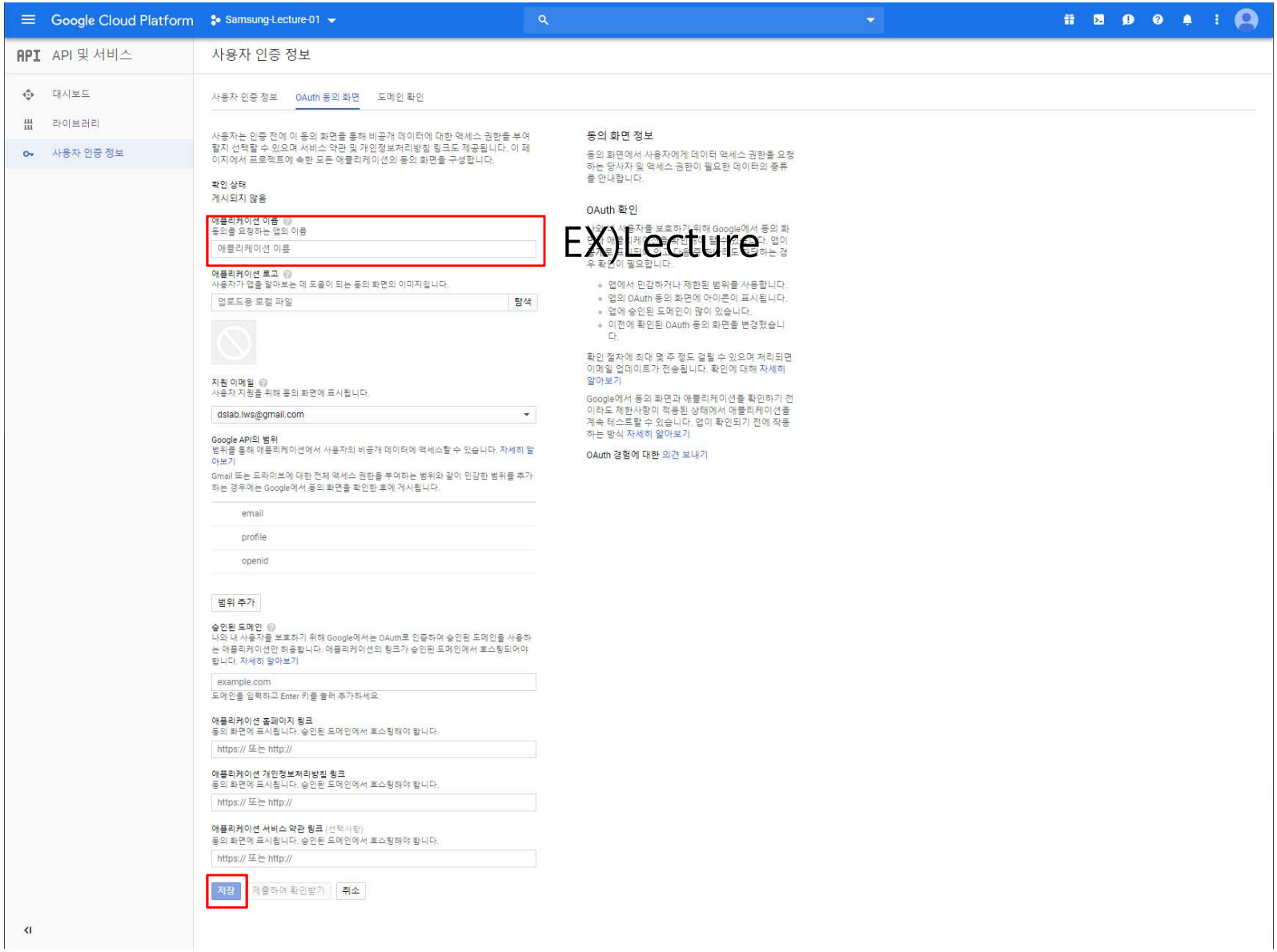
OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - OAuth 동의화면 클릭



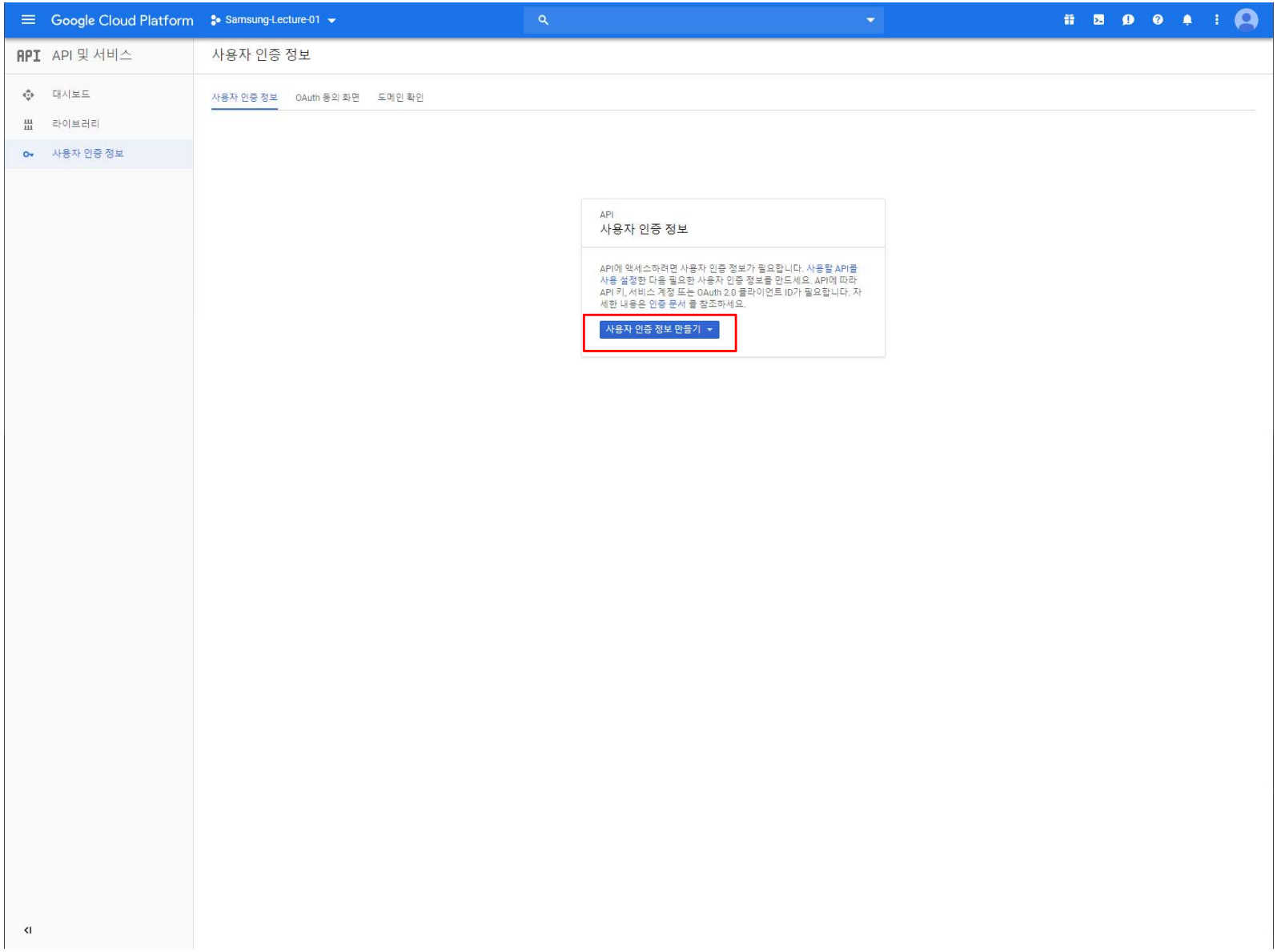
OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 어플리케이션 이름 설정 및 저장



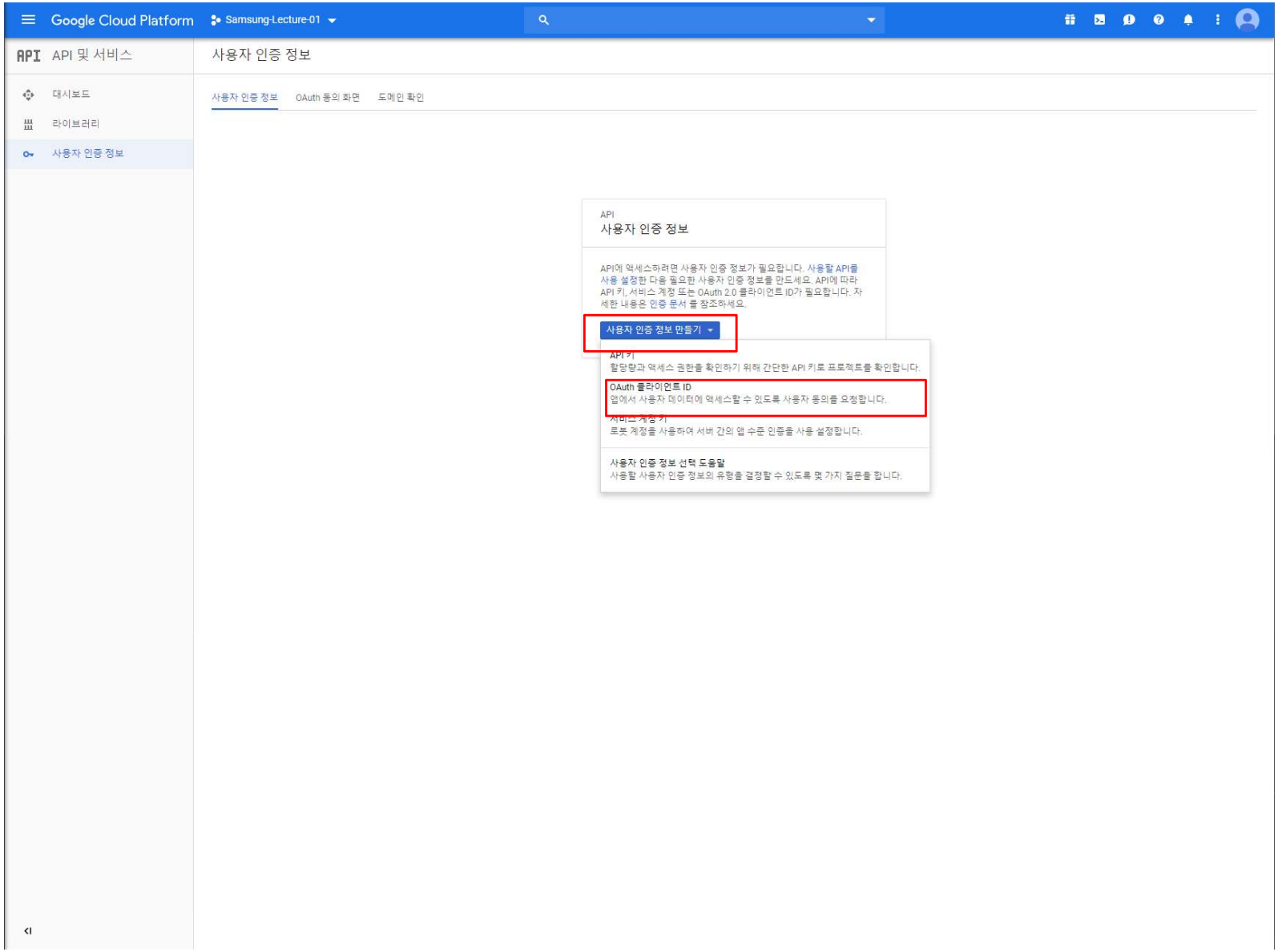
OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 사용자 인증 정보 만들기 클릭



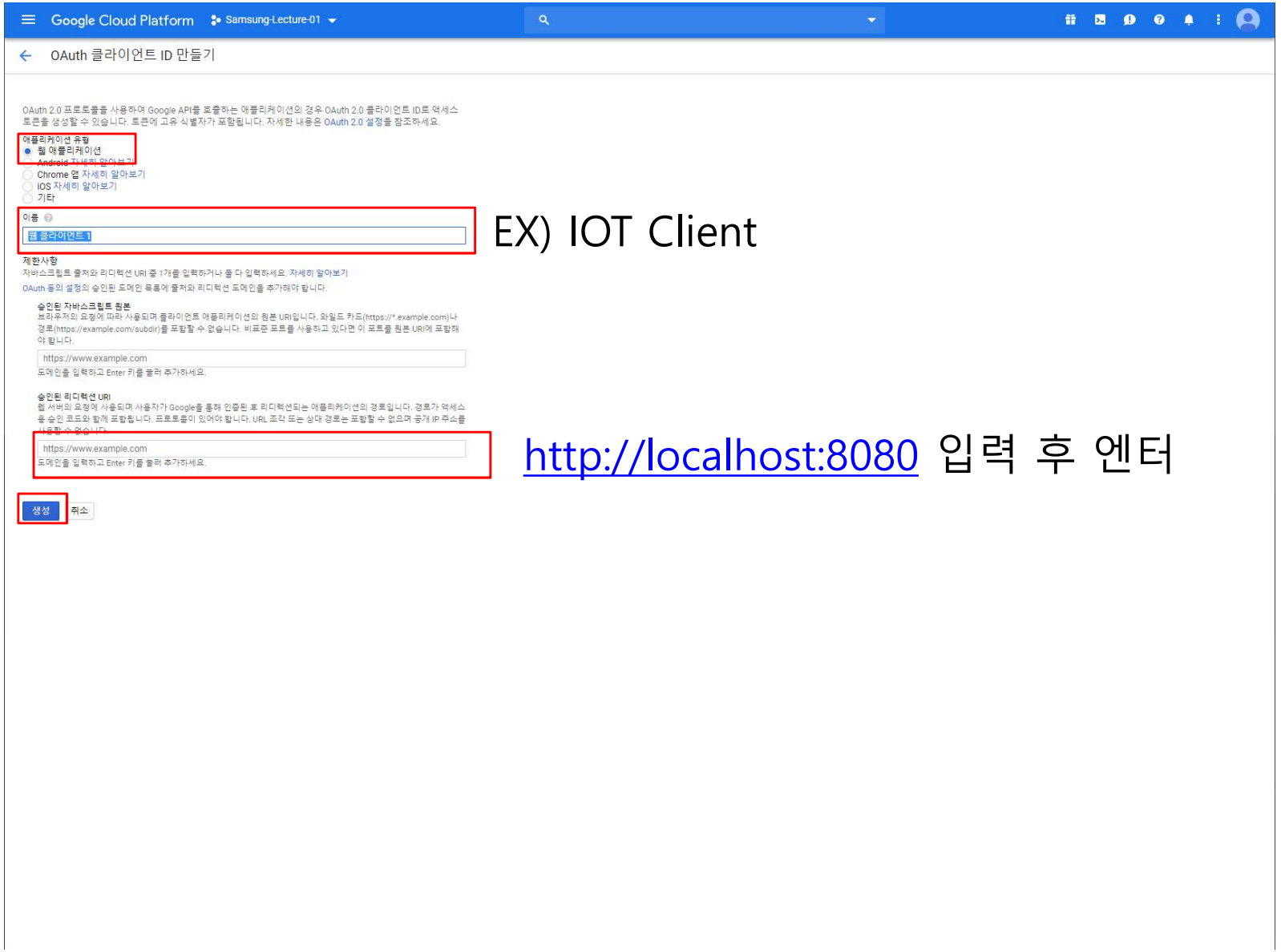
OAuth 2.0 Access Token 받기

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



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - 클라이언트 이름 및 리디렉션 URI설정

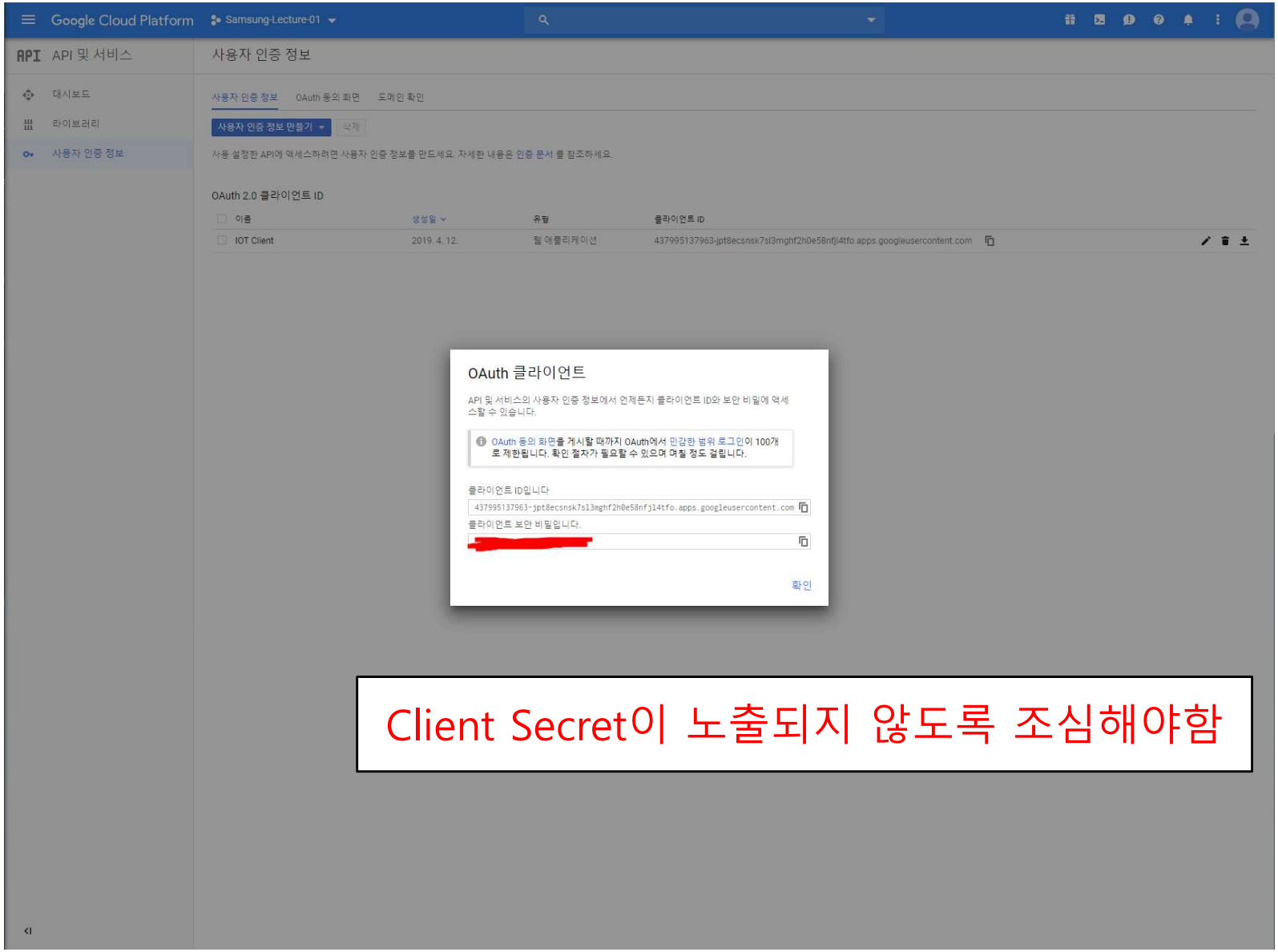


EX) IOT Client

http://localhost:8080 입력 후 엔터

OAuth 2.0 Access Token 받기

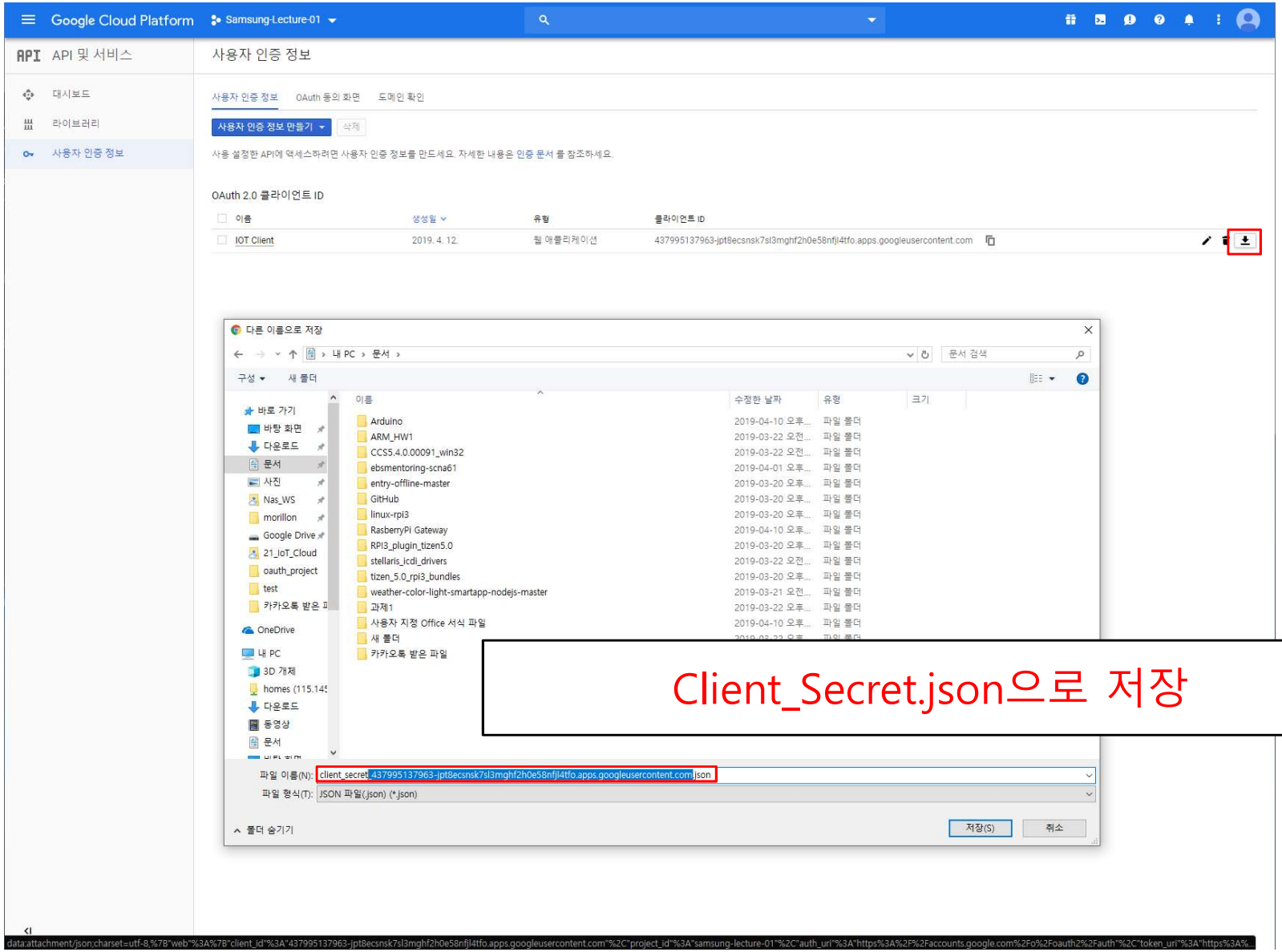
❖ 클라이언트 ID 생성 - 클라이언트 ID 및 클라이언트 Secret 생성확인



Client Secret이 노출되지 않도록 조심해야함

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 생성 - Json파일 다운로드



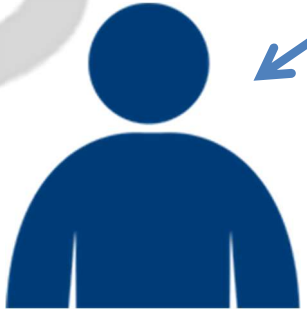
OAuth 2.0 Access Token 받기

❖ Access Token을 받는 과정

Client

Resource Server

② 클라이언트 ID 사용 동의 요청



Resource Owner

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&

Code를 Redirect 받을 주소(Client ID를 만들때 입력한 주소)

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?

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&

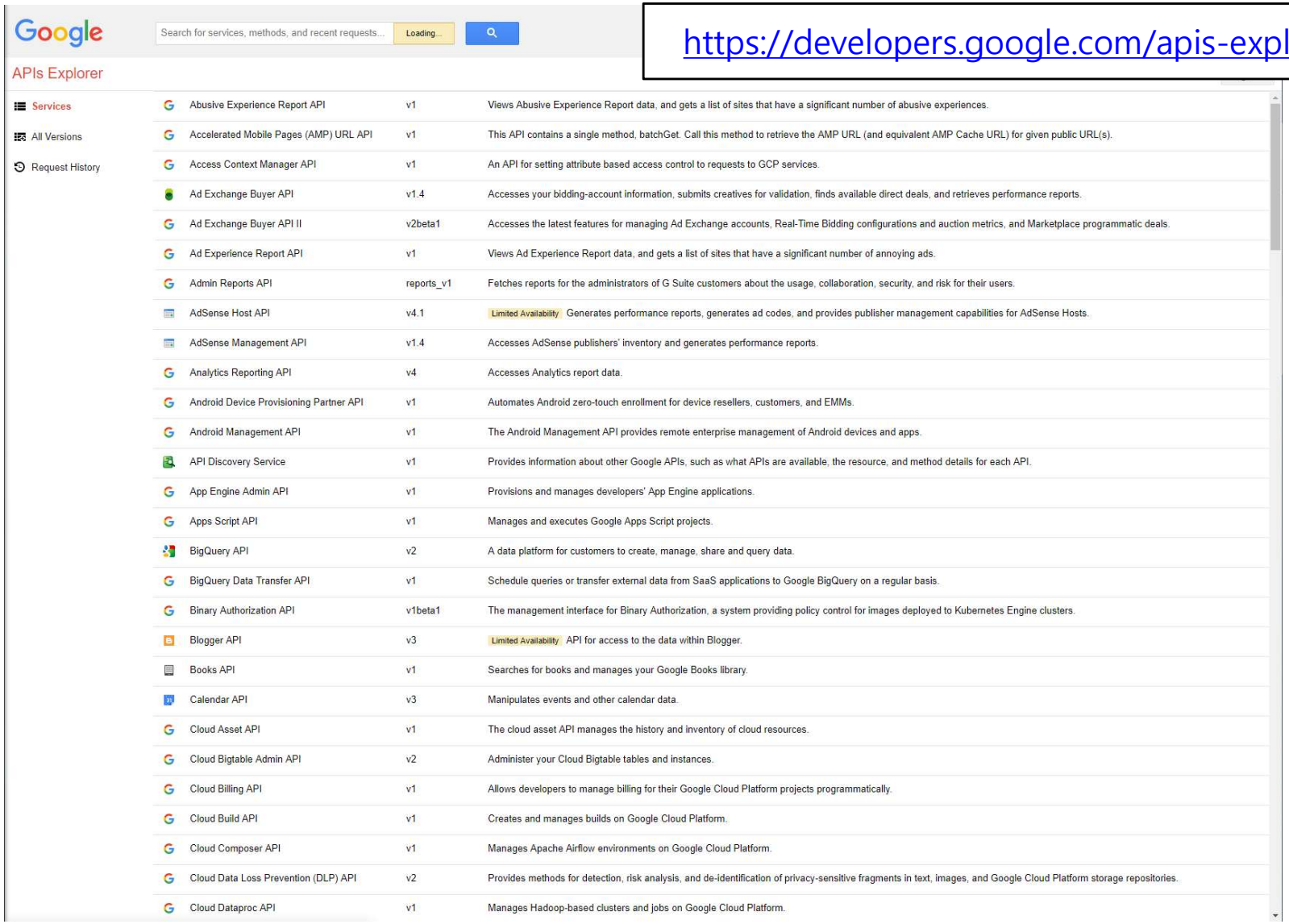
response_type=code&

client_id=**client_id**

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - API Explorer 접속

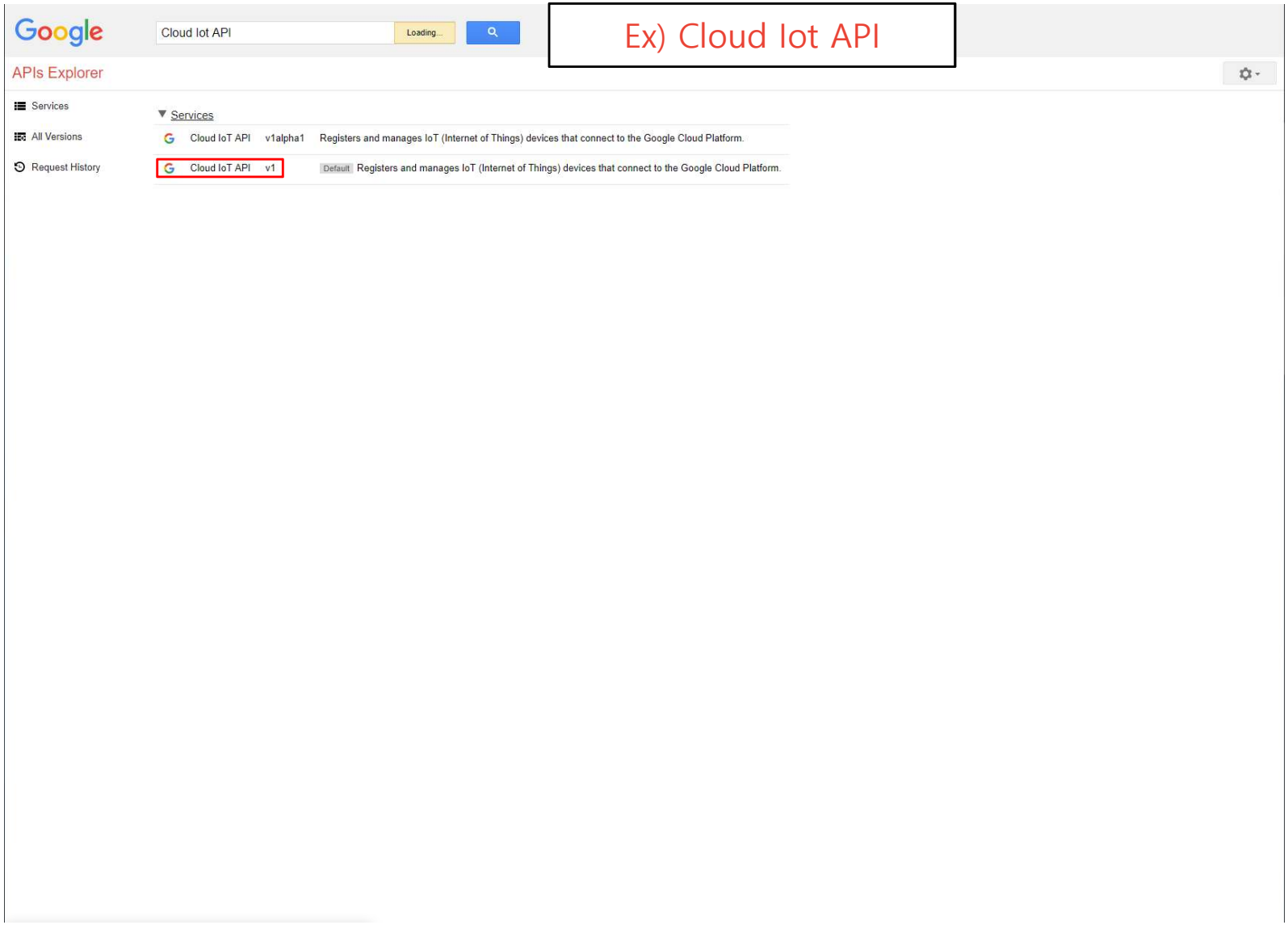
https://developers.google.com/apis-explorer 접속



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) cloudiot.projects.locations.registries.devices.modifyCloudToDeviceConfig

Google APIs Explorer

Services
All Versions
Request History

Learn more about using the Cloud IoT API by reading the documentation.

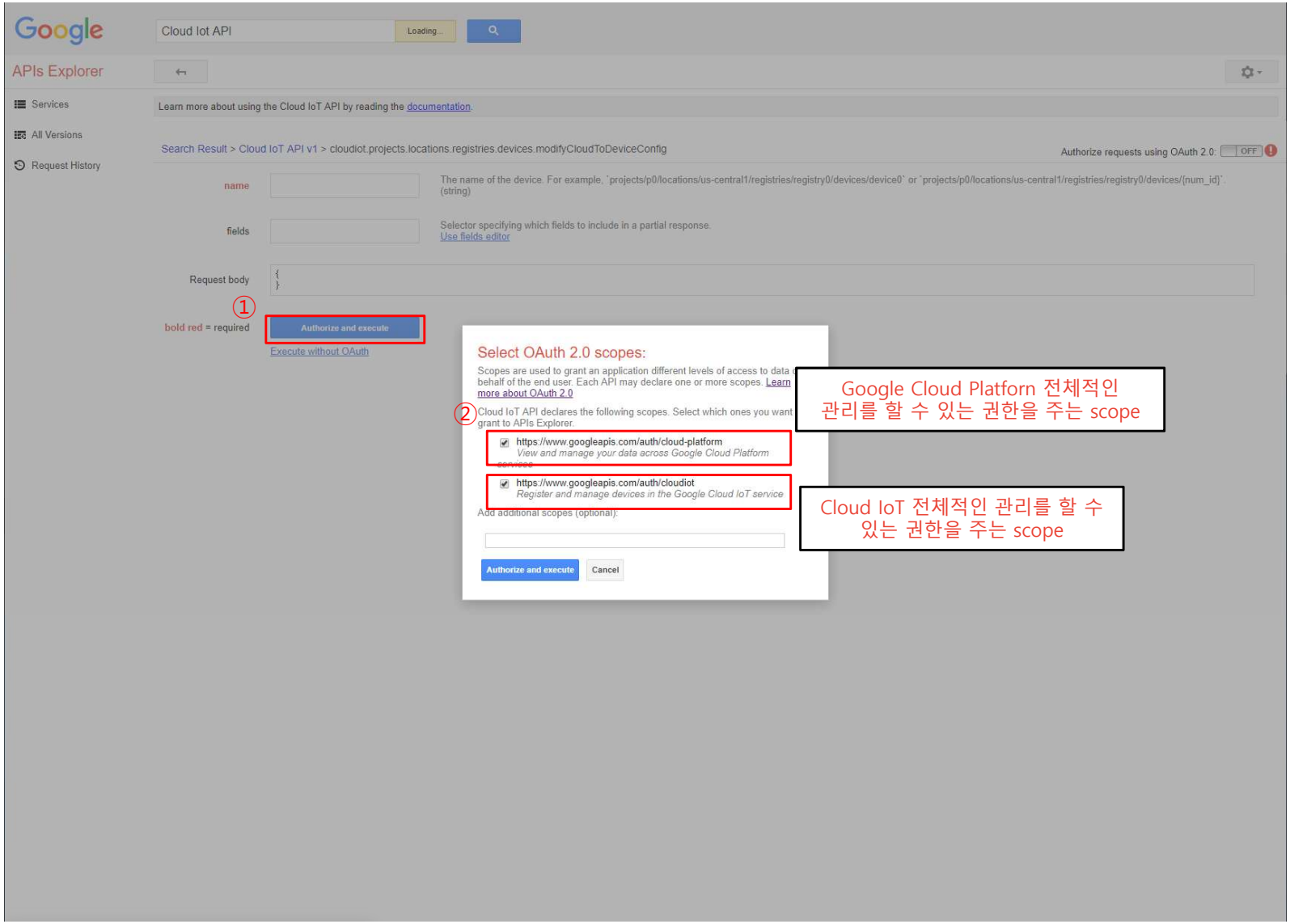
Search Result > Cloud IoT API v1

Authorize requests using OAuth 2.0: OFF

cloudiot.projects.locations.registries.bindDeviceToGateway	Associates the device with the gateway.
cloudiot.projects.locations.registries.create	Creates a device registry that contains devices.
cloudiot.projects.locations.registries.delete	Deletes a device registry configuration.
cloudiot.projects.locations.registries.devices.configVersions.list	Lists the last few versions of the device configuration in descending order (i.e.: newest first).
cloudiot.projects.locations.registries.devices.create	Creates a device in a device registry.
cloudiot.projects.locations.registries.devices.delete	Deletes a device.
cloudiot.projects.locations.registries.devices.get	Gets details about a device.
cloudiot.projects.locations.registries.devices.list	List devices in a device registry.
cloudiot.projects.locations.registries.devices.modifyCloudToDeviceConfig	Modifies the configuration for the device, which is eventually sent from the Cloud IoT Core servers. Returns the modified configuration version and its metadata.
cloudiot.projects.locations.registries.devices.patch	Updates a device.
cloudiot.projects.locations.registries.devices.sendCommandToDevice	Sends a command to the specified device. In order for a device to be able to receive commands, it must: 1) be connected to Cloud IoT Core using the MQTT protocol, and 2) be subscribed to the group of MQTT topics specified by /devices/{device-id}/commands/#. This subscription will receive commands at the top-level topic /devices/{device-id}/commands as well as commands for subfolders, like /devices/{device-id}/commands/subfolder. Note that subscribing to specific subfolders is not supported. If the command could not be delivered to the device, this method will return an error; in particular, if the device is not subscribed, this method will return FAILED_PRECONDITION. Otherwise, this method will return OK. If the subscription is QoS 1, at least once delivery will be guaranteed; for QoS 0, no acknowledgment will be expected from the device.
cloudiot.projects.locations.registries.devices.states.list	Lists the last few versions of the device state in descending order (i.e.: newest first).
cloudiot.projects.locations.registries.get	Gets a device registry configuration.
cloudiot.projects.locations.registries.getIamPolicy	Gets the access control policy for a resource. Returns an empty policy if the resource exists and does not have a policy set.
cloudiot.projects.locations.registries.groups.bindDeviceToGateway	Associates the device with the gateway.
cloudiot.projects.locations.registries.groups.devices.configVersions.list	Lists the last few versions of the device configuration in descending order (i.e.: newest first).
cloudiot.projects.locations.registries.groups.devices.get	Gets details about a device.
cloudiot.projects.locations.registries.groups.devices.list	List devices in a device registry.
cloudiot.projects.locations.registries.groups.devices.modifyCloudToDeviceConfig	Modifies the configuration for the device, which is eventually sent from the Cloud IoT Core servers. Returns the modified configuration version and its metadata.
cloudiot.projects.locations.registries.groups.devices.patch	Updates a device.
cloudiot.projects.locations.registries.groups.devices.sendCommandToDevice	Sends a command to the specified device. In order for a device to be able to receive commands, it must: 1) be connected to Cloud IoT Core using the MQTT protocol, and 2) be subscribed to the group of MQTT topics specified by /devices/{device-id}/commands/#. This subscription will receive commands at the top-level topic /devices/{device-id}/commands as well as commands for subfolders, like /devices/{device-id}/commands/subfolder. Note that subscribing to specific subfolders is not supported. If the command could not be delivered to the device, this method will return an error; in particular, if the device is not subscribed, this method will return FAILED_PRECONDITION. Otherwise, this method will return OK. If the subscription is QoS 1, at least once delivery will be guaranteed; for QoS 0, no acknowledgment will be expected from the device.
cloudiot.projects.locations.registries.groups.devices.states.list	Lists the last few versions of the device state in descending order (i.e.: newest first).
cloudiot.projects.locations.registries.groups.getIamPolicy	Gets the access control policy for a resource. Returns an empty policy if the resource exists and does not have a policy set.
cloudiot.projects.locations.registries.groups.setIamPolicy	Sets the access control policy on the specified resource. Replaces any existing policy.
cloudiot.projects.locations.registries.groups.testIamPermissions	Returns permissions that a caller has on the specified resource. If the resource does not exist, this will return an empty set of permissions, not a NOT_FOUND error.

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

Url 인코딩사이트(https://meyerweb.com/eric/tools/dencoder/)접속

사용할 scope 주소 붙여넣기

OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 요청 - 인코딩된 scope 주소 입력

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_passthrough_value&

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

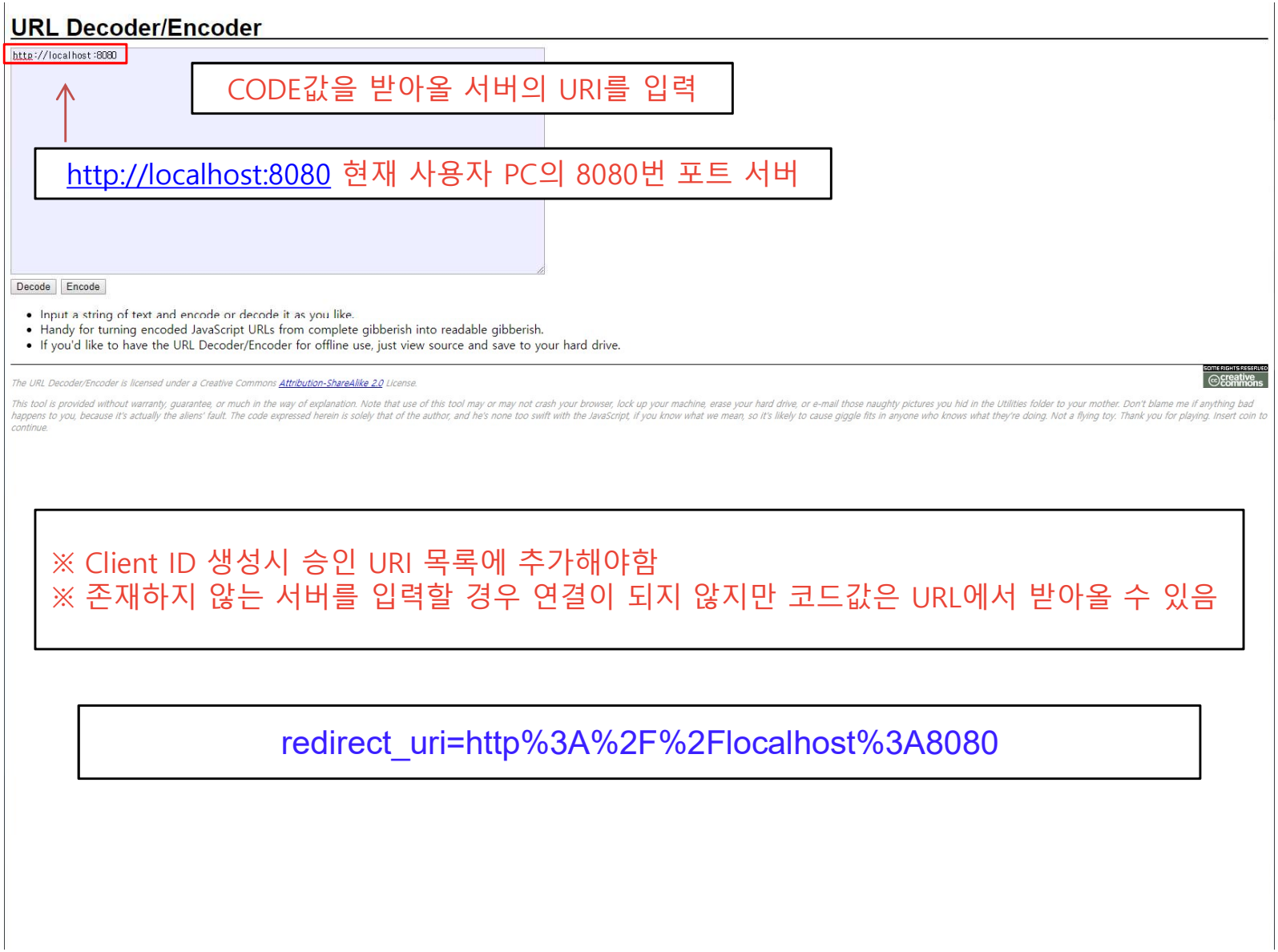
response_type=code&

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



※ 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

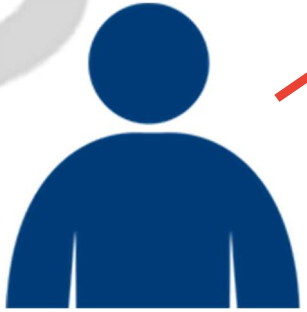
OAuth 2.0 Access Token 받기

❖ Access Token을 받는 과정

Client

Resource Server

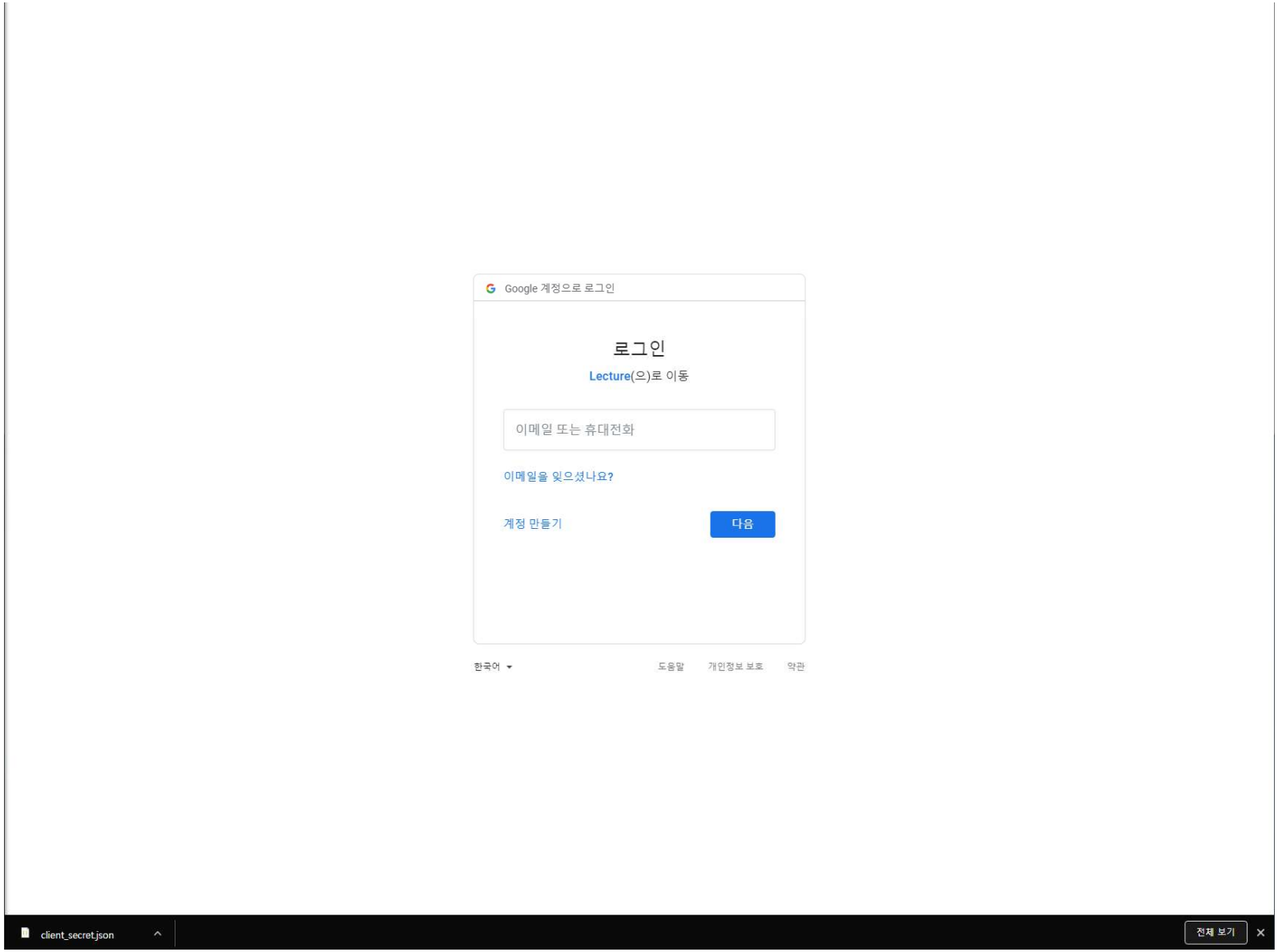
③ 클라이언트 ID 사용 동의 및 Code 생성



Resource Owner

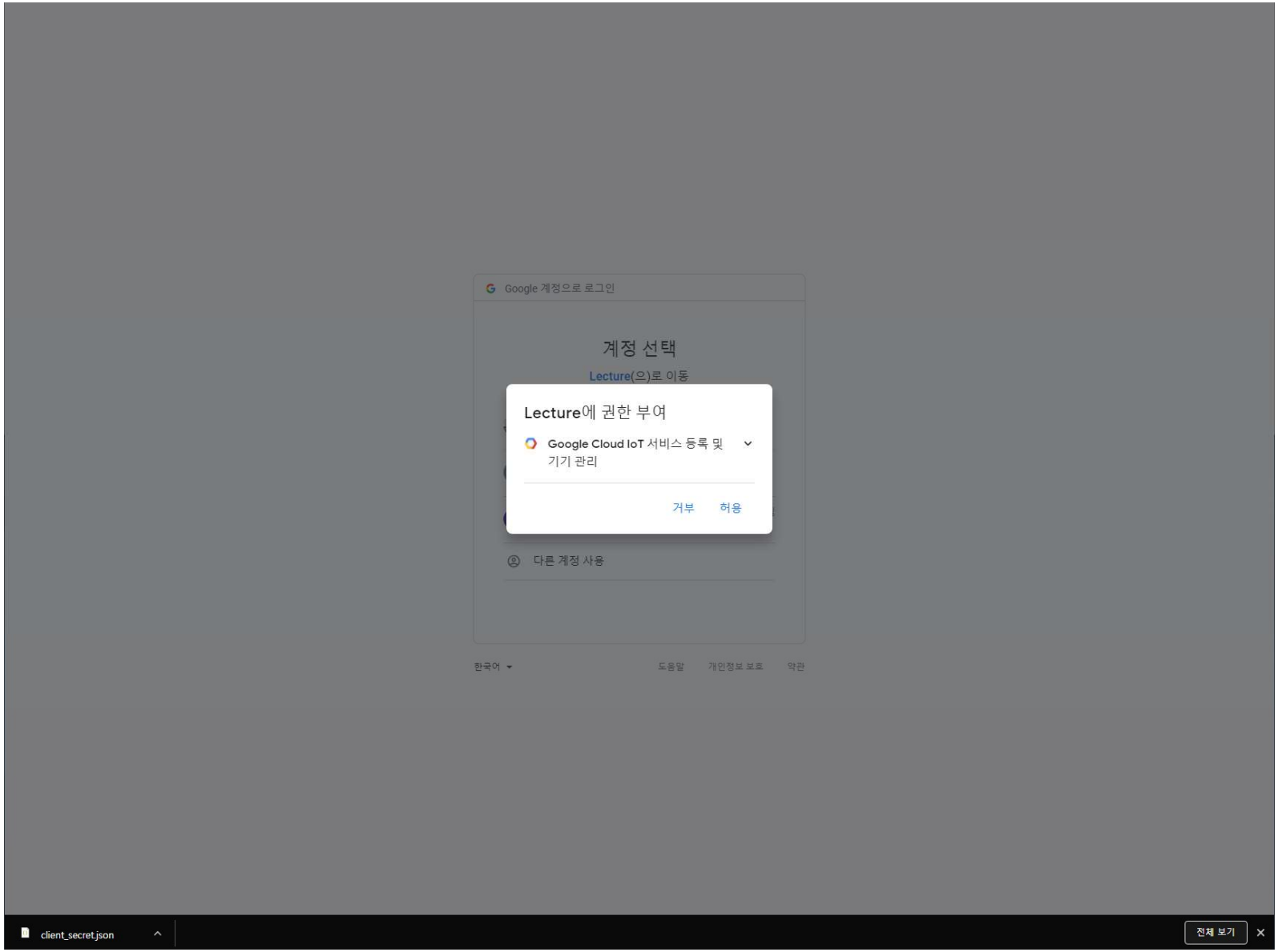
OAuth 2.0 Access Token 받기

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



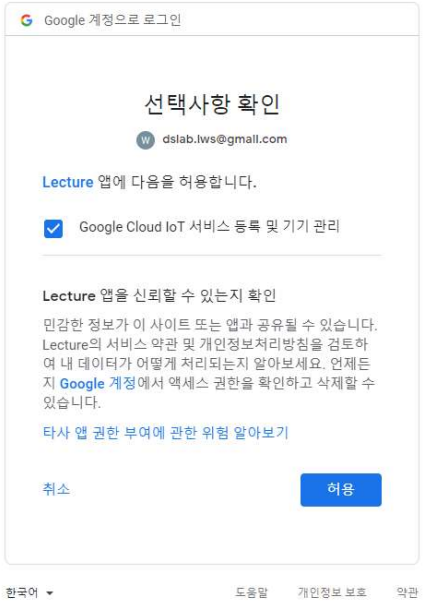
OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 및 Code 생성 - 허용



OAuth 2.0 Access Token 받기

❖ 클라이언트 ID 사용 동의 및 Code 생성 - 허용

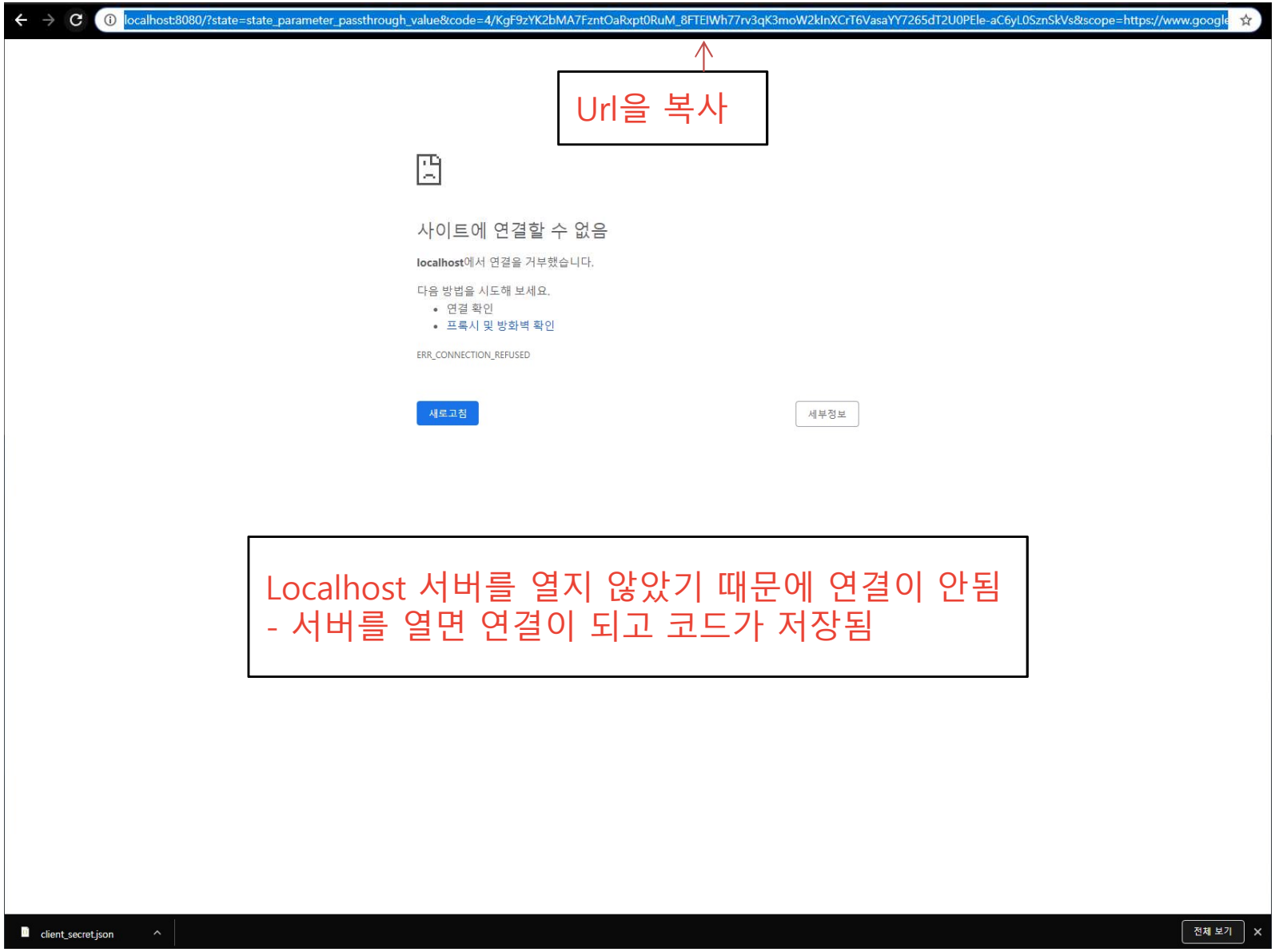


client_secret.json

조대표님
수정이 필요하겠네요. 역시 한...

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-4DZHIzFFgt3eAQ11_Z9UElPfZj7dQ-N_7WrOOpHjLBDgXBDnwERe4TgWoY9yvDx3NS4yUKEfjZ4s&scope=https://www.googleapis.com/auth/cloudiot

OAuth 2.0 Access Token 받기

❖ Access Token을 받는 과정

Client

④ Access Token 및 Refresh Token 받기

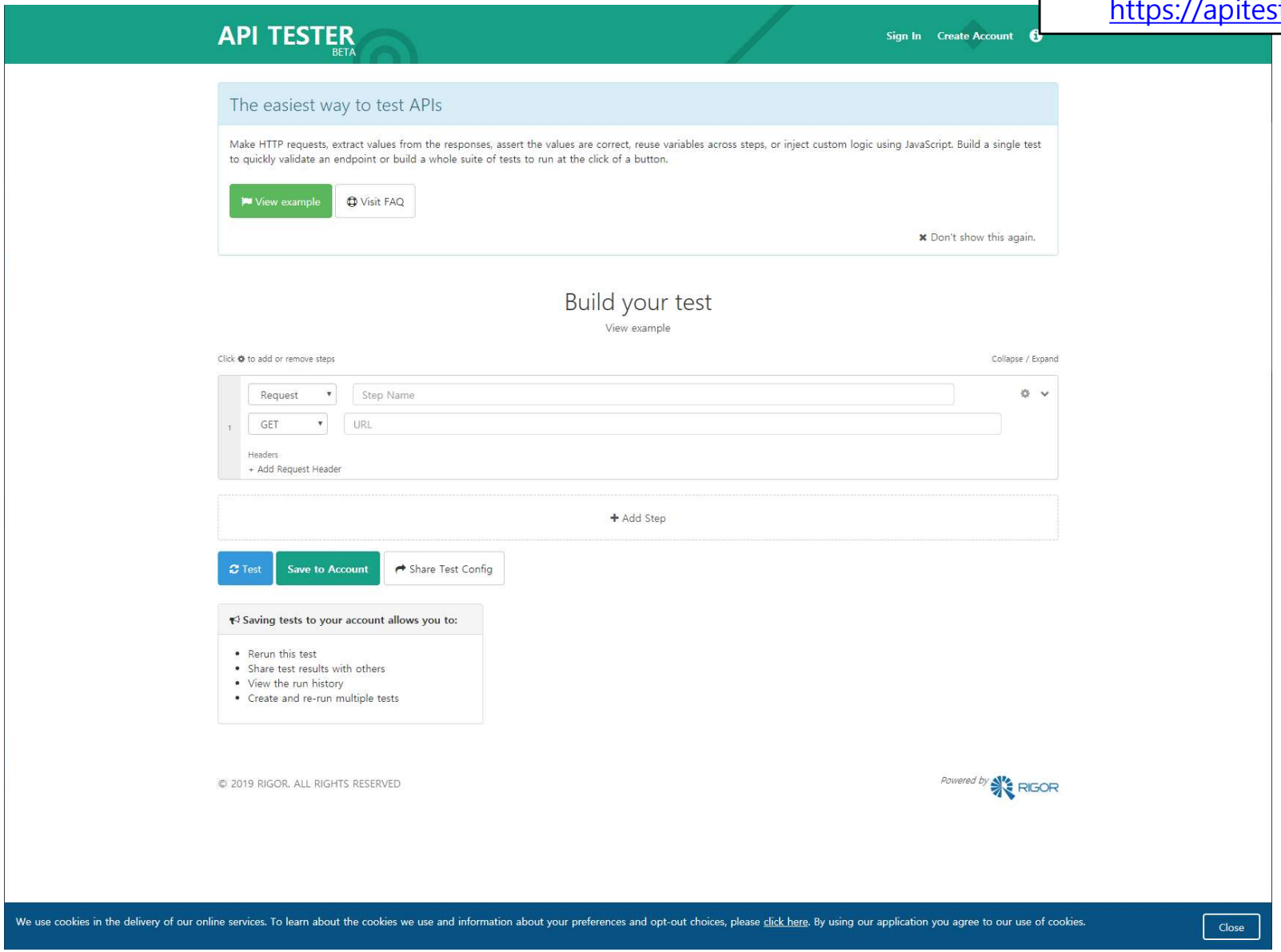
Resource Server

Resource Owner

OAuth 2.0 Access Token 받기

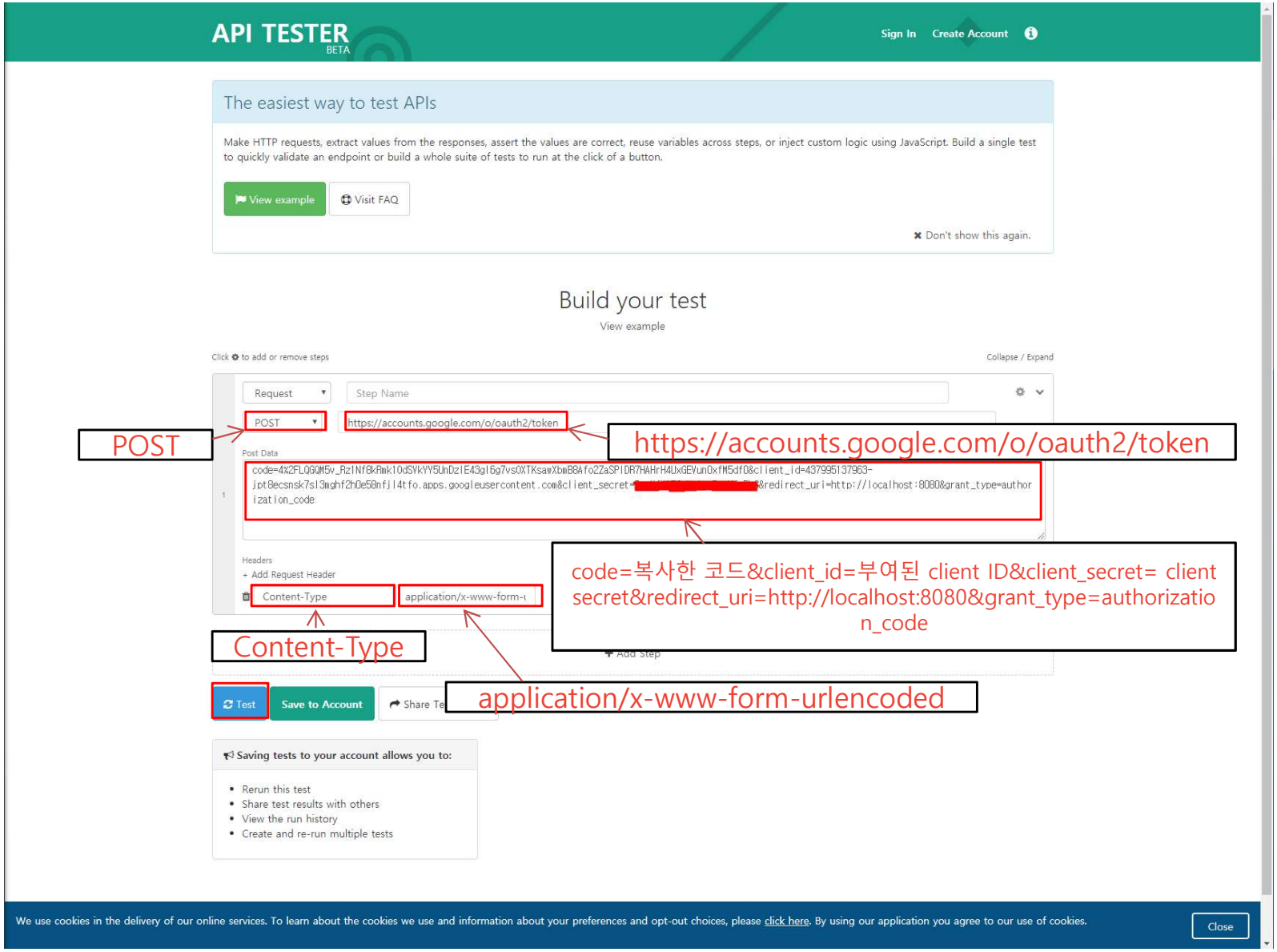
❖ Access Token 및 Refresh Token 받기 - API Tester 접속

https://apitester.com/



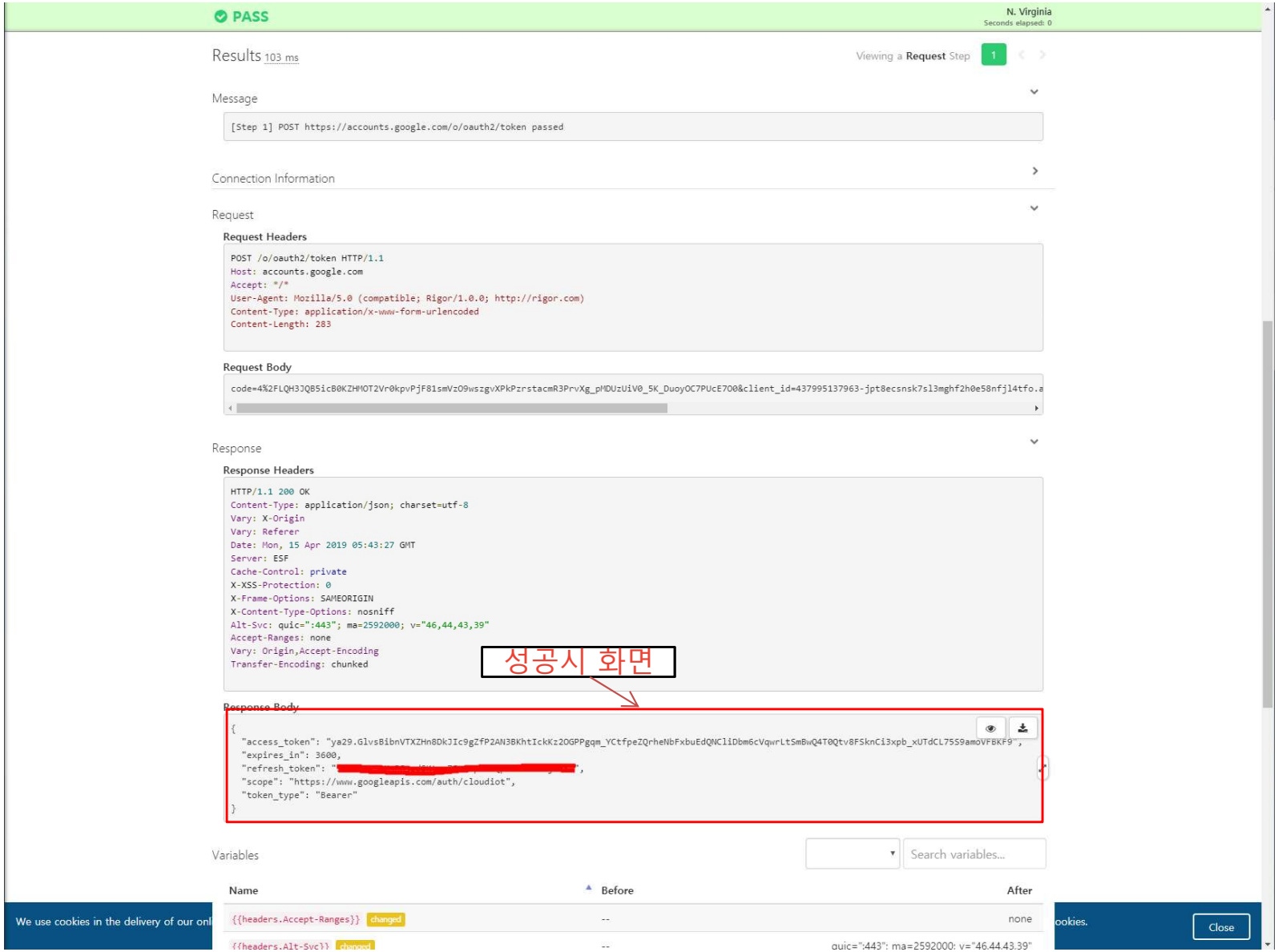
OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - Access Token 요청



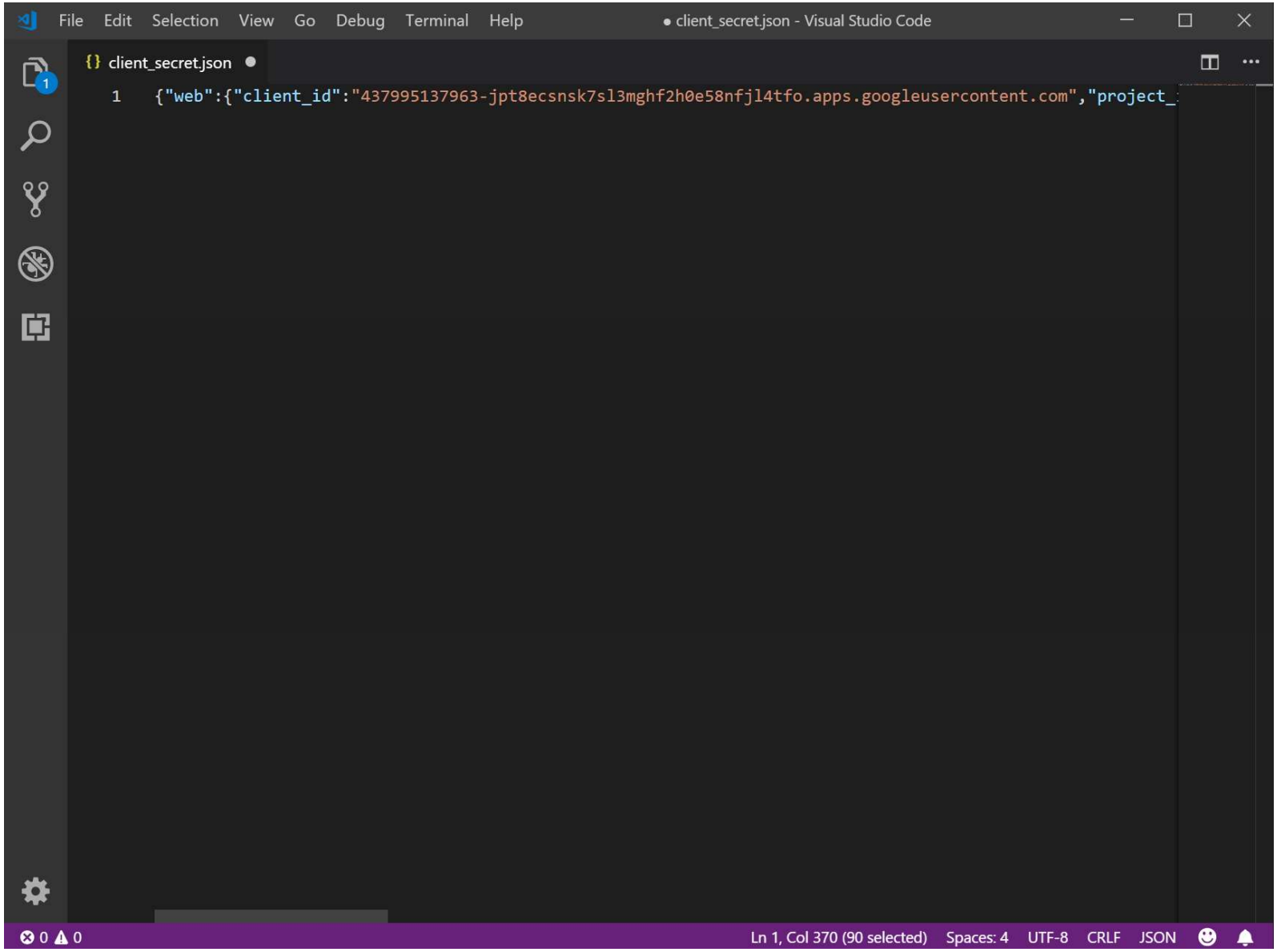
OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - Refresh Token 복사



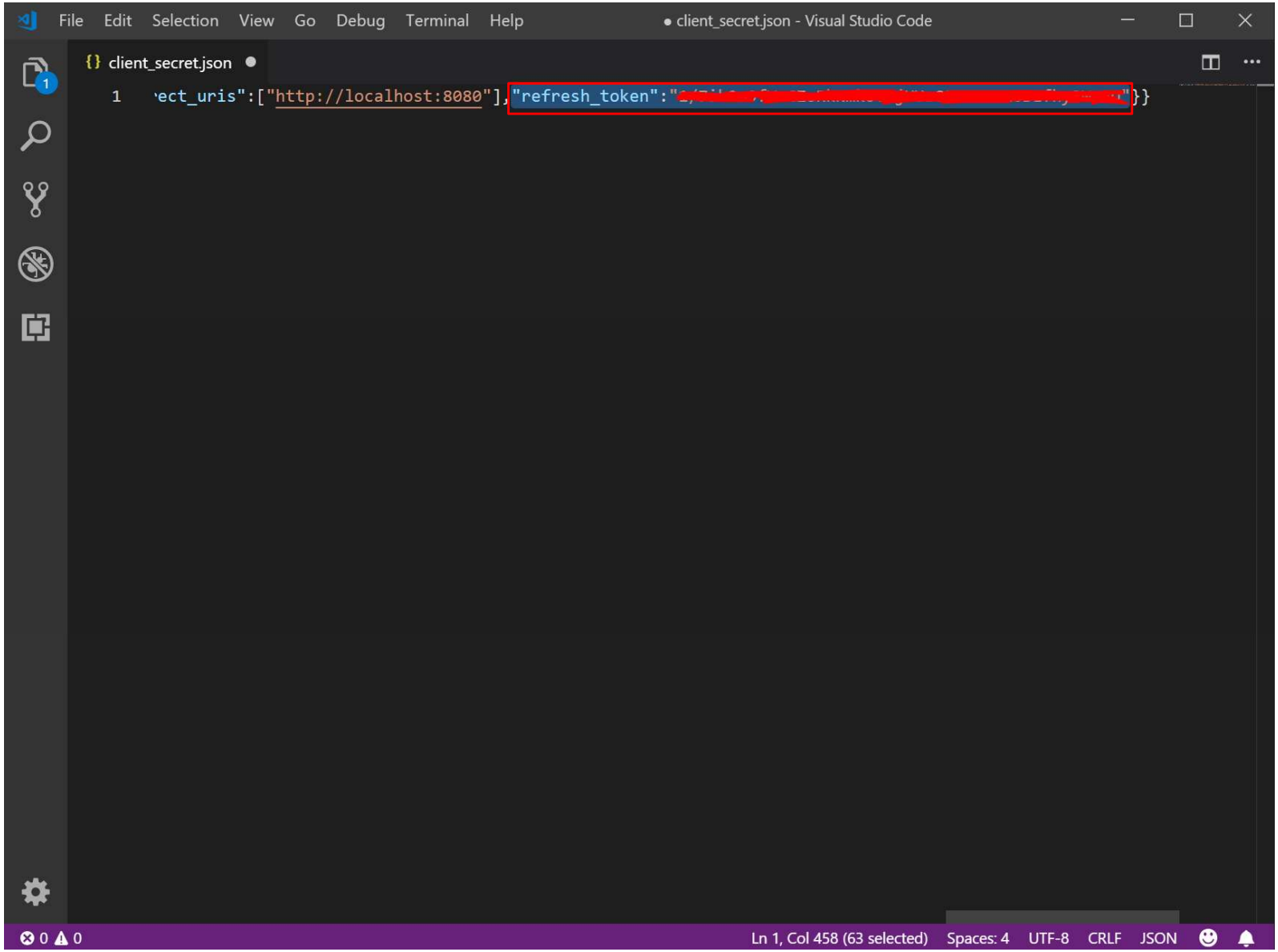
OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - client_secret.json파일 열기



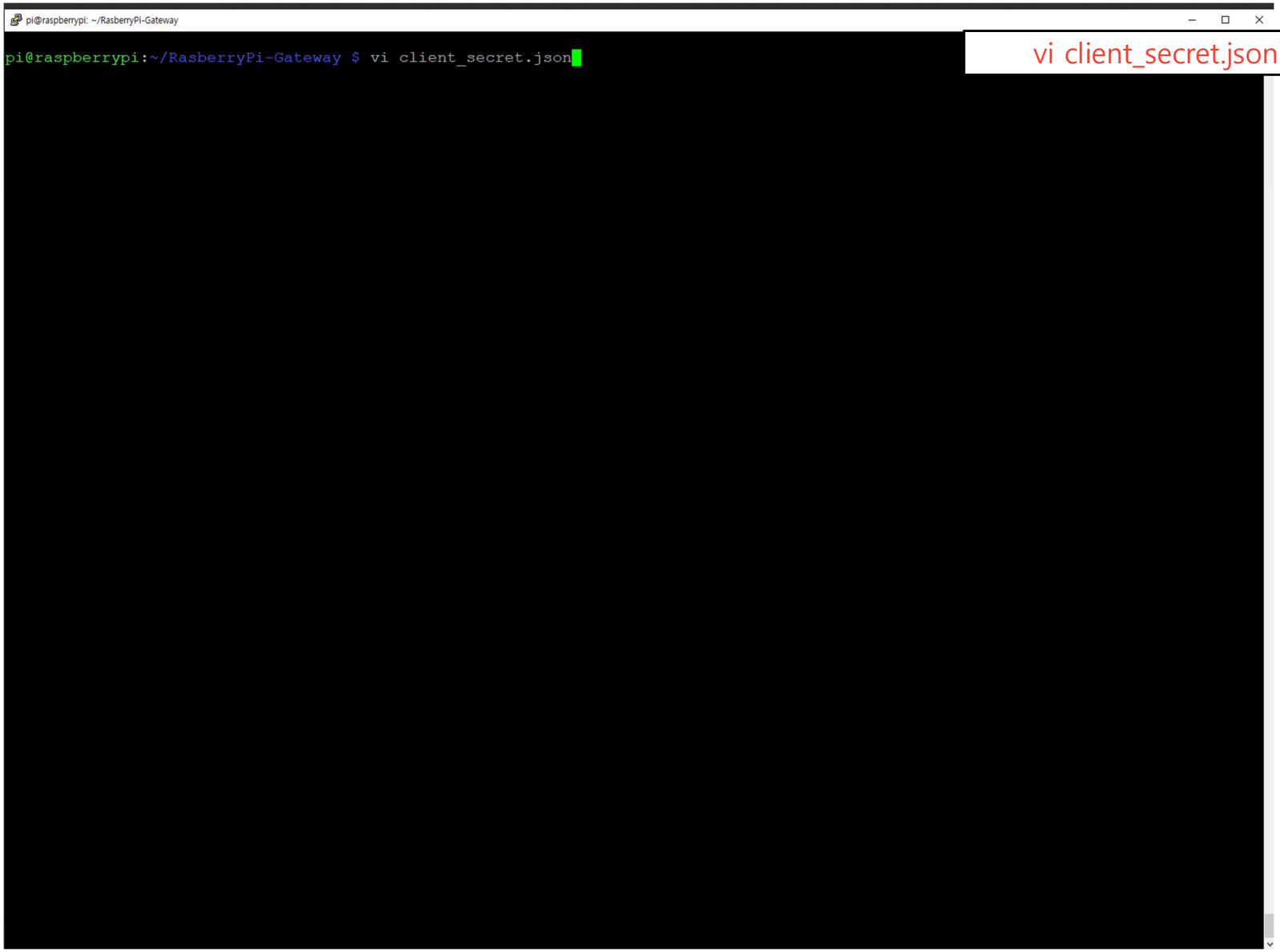
OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - Refresh Token 붙여넣기



OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - 라즈베리파이 파일 생성



OAuth 2.0 Access Token 받기

❖ Access Token 및 Refresh Token 받기 - 붙여넣기

OAuth 2.0 Access Token 받기

❖ Access Token을 받는 과정

Client

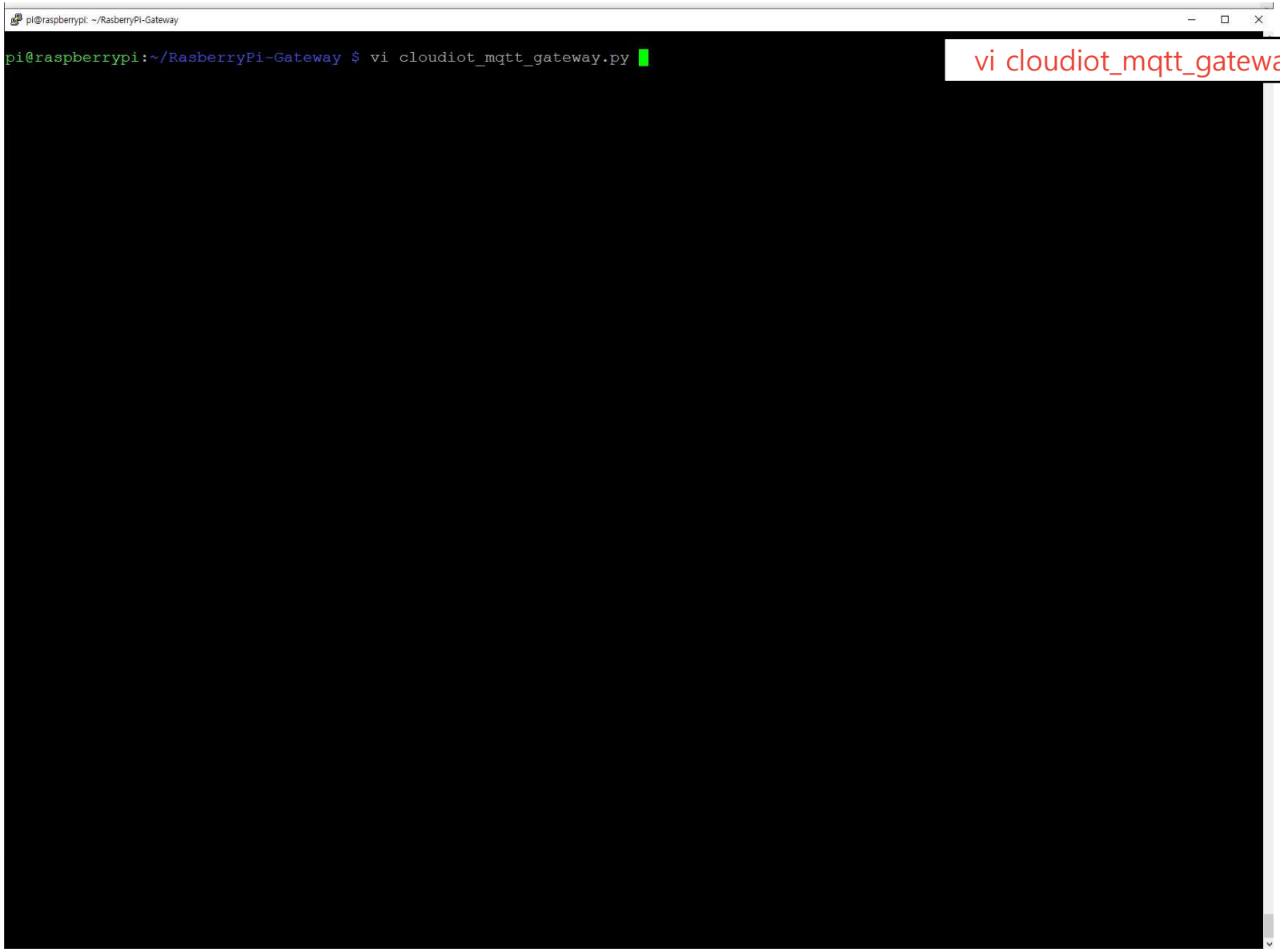
⑤ Refresh Token으로 Access Token 갱신

Resource Server

Resource Owner

OAuth 2.0 Access Token 받기

❖ Refresh Token으로 Access Token 갱신 - 파이썬 코드 경로 입력



OAuth 2.0 Access Token 받기

❖ Refresh Token으로 Access Token 갱신 - main함수에 path 수정

pi@raspberrypi: ~/RasberryPi-Gateway

```
    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

pi@raspberrypi: ~/RasberryPi-Gateway

source run-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-a
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
```

COM10 - PuTTY

```
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"}
```

OAuth 2.0 Access Token 받기

❖ Refresh Token으로 Access Token 갱신 - 디바이스 테스트

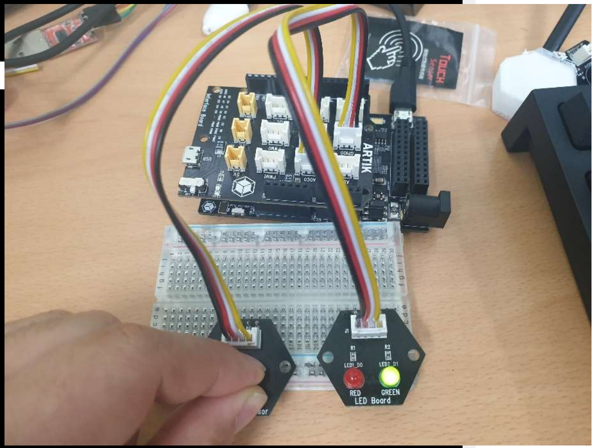
30도가 넘어가면 LED가 켜지는지 확인

pi@raspberrypi: ~/RasberryPi-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
```

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"}
```



Thank you