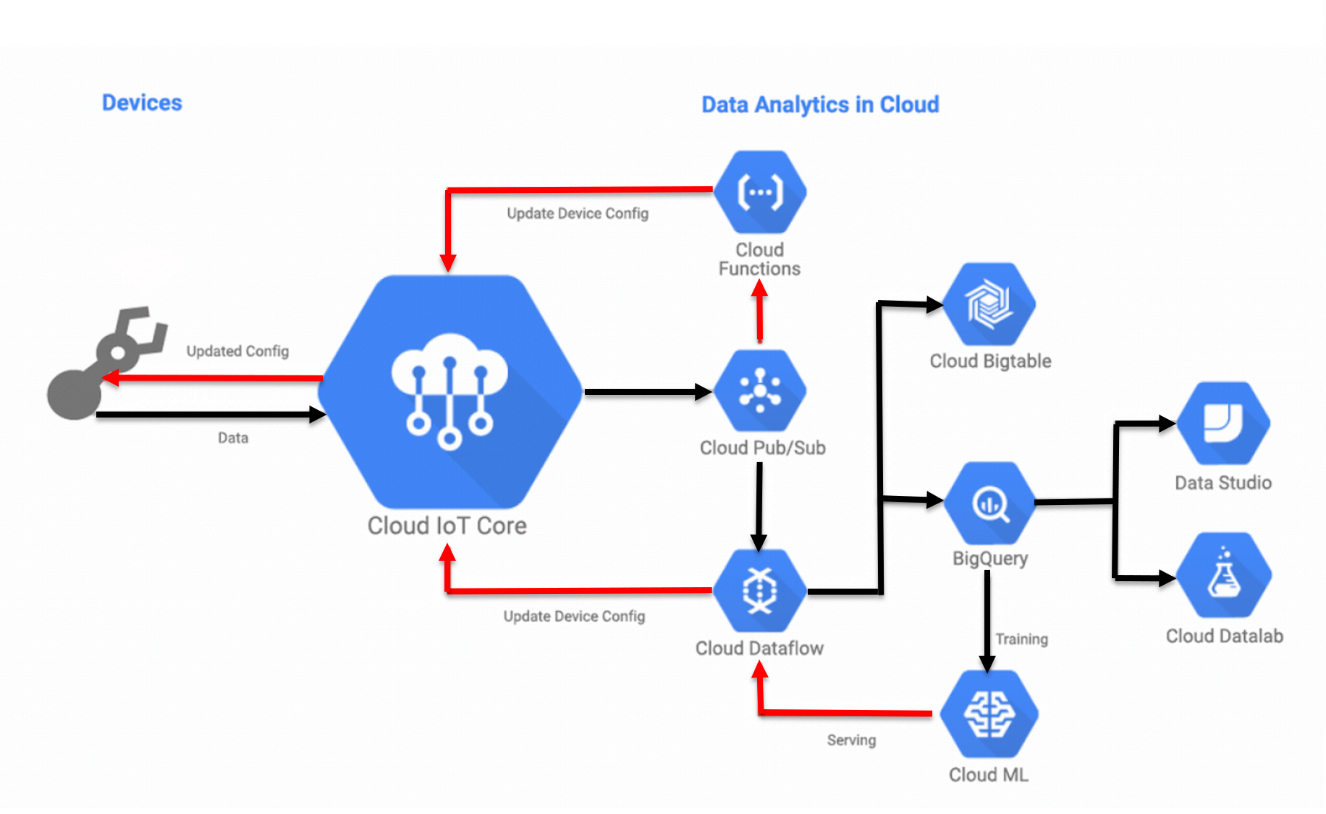
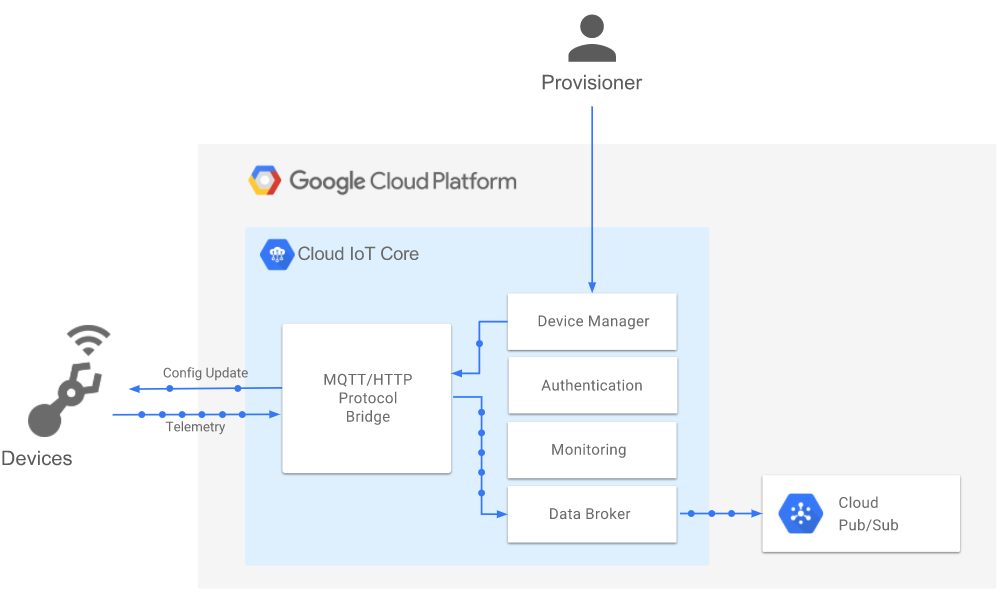
**Phần 1: Giới thiệu về Google IoT Core**

**I. Tổng quan về Google Cloud Platform và Google IoT Core**

- Dịch vụ **Google IoT Core** là 1 dịch vụ con nằm trong **Google Cloud Platform** (GCP). Đây là 1 dịch vụ hữu ích cho phép chúng ta tạo kết nối hai chiều an toàn giữa các thiết bị IoT với nền tảng đám mây của Google thông qua các phương thức bảo mật tối ưu. Với nền tảng mà google cung cấp, các nhà phát triển không cần phải bận tâm về cơ sở hạ tầng, thiết lập bảo mật, trích xuất dữ liệu hay phân tích dữ liệu từ các thiết bị IoT sao cho hiệu quả. Google Cloud Platform cung cấp cho chúng ta rất nhiều dịch vụ con với các chức năng khác nhau như Cloud BigQuery, AutoML, DataLab,... Và **Google IoT Core** chính là cầu nối quan trọng để các thiết bị IoT của chúng ta có thể dễ dàng tương tác với các dịch vụ đó.

*Ví dụ về một mô hình mẫu giữa Cloud IoT Core với các dịch vụ khác của GCP*

**II. Google IoT Core là gì?**

****

* Google IoT Core là một **dịch vụ quản lý** với 2 thành phần chính là **Device Management** và **Communication Broker.**
  + **Device Management:** Các thiết bị kết nối tới Google IoT Core sẽ được xem như là một “*device*” bên trong Google IoT Core. Gộp tất cả các “*device*” lại thành một nhóm, chúng ta có “*registry*” là ô chứa đại diện cho tất cả các “*device*” đó. Việc khởi tạo và quản lý “*device*” với “*registry*” sẽ phụ thuộc vào cách tổ chức của chúng ta.
  + **Communication Broker:** Sau khi thực hiện thành công việc kết nối thiết bị IoT với Google IoT Core. Bây giờ, nếu muốn dữ liệu thu thập từ các thiết bị này gửi lên Google IoT Core được thì chúng ta phải sử dụng một trong hai giao thức đó là **MQTT** và **HTTPS**. Sau đó, Data Brokersẽ chuyển tiếp luồng dữ liệu này tới một dịch vụ khác để lưu trữ có tên gọi là Cloud Pub/Sub.

**III. Thiết lập GCP để liên kết với tài khoản google**

* Để bắt đầu với Google IoT Core, trước tiên bạn cần phải có tài khoản google. Các bạn có thể đăng ký tại đây:

<https://accounts.google.com/SignUp?hl=vn>

* Sau khi đăng ký tài khoản google xong, bạn có thể đăng nhập và chuyển hướng tới giao diện Console của Google Cloud Platform:

[https://console.cloud.google.com](https://console.cloud.google.com/)

* Để bắt đầu, chúng ta khởi tạo một project trong GCP. Sau đó, chúng ta liên kết tài khoản google với dịch vụ **Billing**. (Lưu ý để đăng ký thành công bạn cần phải có thông tin thẻ Visa hoặc MasterCard, các bạn có thể đăng ký mở thẻ ở các ngân hàng trong nước).
* Sau khi liên kết thành công dịch vụ **Billing** với tài khoản google của bạn. Lúc này chúng ta sẽ điều hướng trang console chuyển tới dịch vụ Google IoT Core và chọn **Enable** để kích hoạt dịch vụ này.
* Ở phần này, chúng ta đã đi qua tổng quan về kiến trúc của Google IoT Core cũng như lợi ích to lớn mà dịch vụ này đem lại. Trong phần tới, nhóm TAPIT IoTs sẽ đi sâu hơn vào lý thuyết cũng như thực hành để các bạn có thể nắm rõ và vận dụng công nghệ này thành thạo hơn. Chúc các bạn thành công!

**Phần 2: Hướng dẫn tạo Public/ Private key và Json Web Token**

**I. Vì sao phải tạo Public/ Private Key và Json Web Token (JWT) ?**

* Google IoT Core sử dụng hai chuẩn bảo mật là **TLS1.2** cho tầng *transport* và **JWT** cho tầng *application*. Điều này có nghĩa chúng ta sẽ cần phải tạo ra cặp Public key - Private key cho thiết bị IoT và Cloud để có thể mã hóa thông tin liên lạc theo chuẩn TLS đưa ra. Đồng thời thông qua JWT ta sẽ bảo đảm an toàn hơn cho việc trao đổi cặp key này giữa các thiết bị IoT với Cloud.

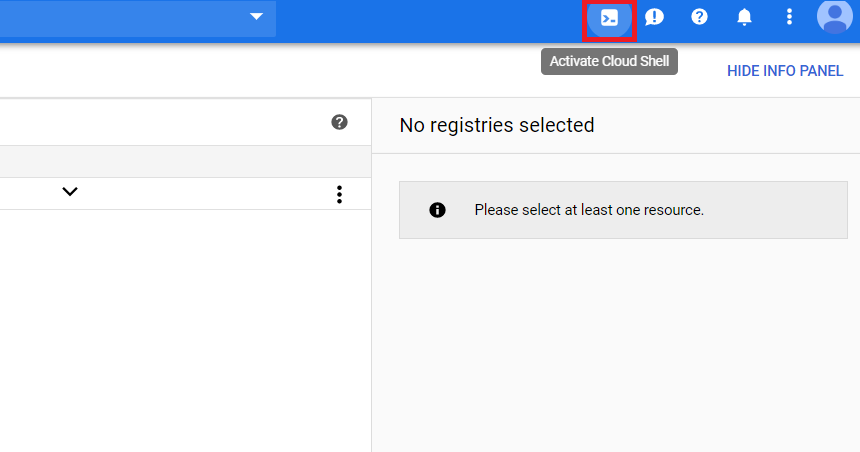
+ Thiết bị IoT sử dụng ***Private key*** để tạo phần **signature** cho **JWT**. Mã JWT này sẽ được chuyển tới Google IoT Core để *chứng minh danh tính* cho thiết bị IoT này.

+ Google IoT Core sử dụng ***Public key*** để *xác nhận danh tính* của thiết bị này.

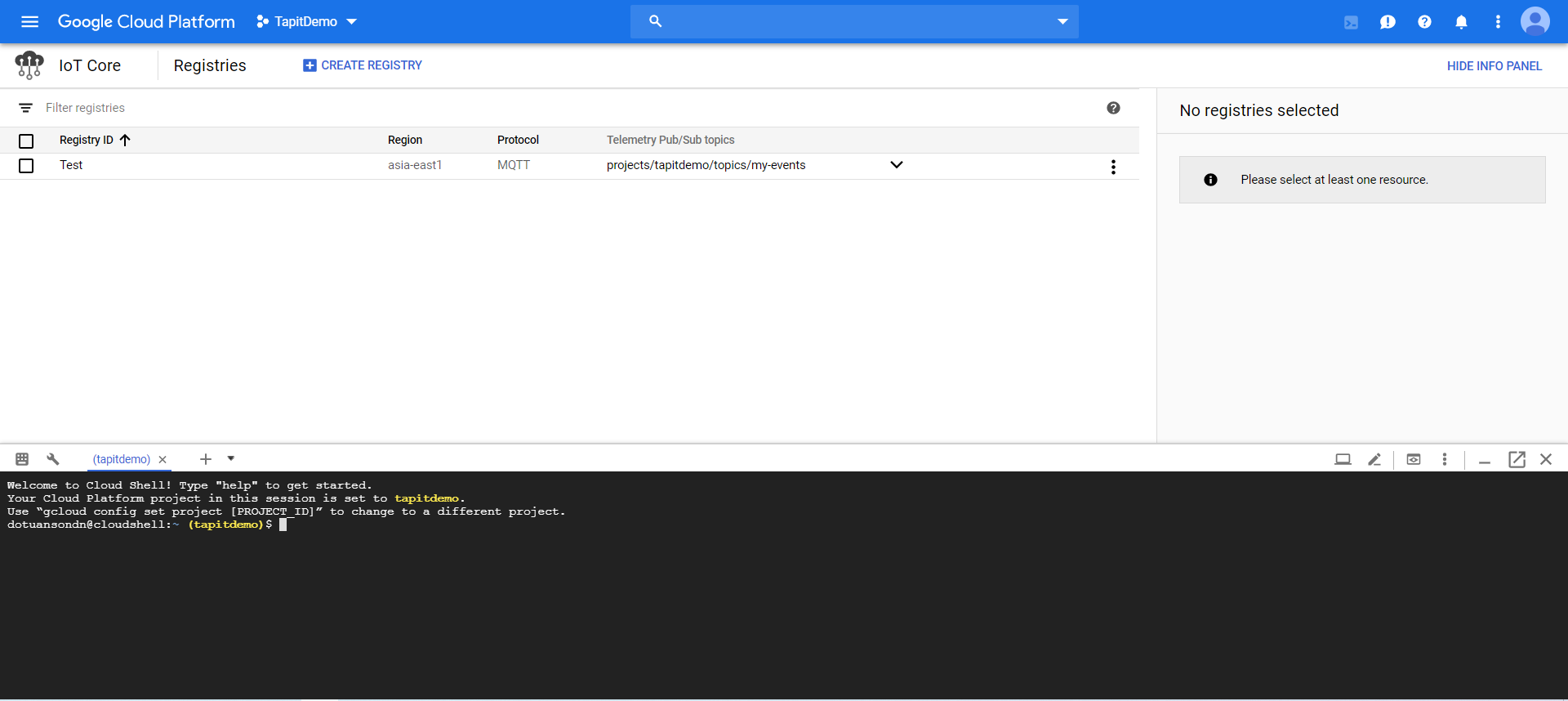
**II. Hướng dẫn tạo Public/ Private Key**

* Google IoT Core hỗ trợ cho chúng ta hai thuật toán để sinh cặp public private key là RSA và Elliptic Curve.
* Để tạo cặp key này chúng ta sẽ sử dụng công cụ hỗ trợ dòng lệnh gcloud mà GCP đã cung cấp cho chúng ta.

+ Tại giao diện Console, ở thanh công cụ phía trên ta nhấn vào biểu tượng có tên **Activate Cloud Shell** như hình bên dưới.



+ Giao diện Console sẽ trả về cho chúng ta cửa sổ dòng lệnh gcloud để tương tác với các nguồn tài nguyên trong GCP.



+ Tạo cặp key sử dụng thuật toán RSA:

openssl genpkey -algorithm RSA -out rsa\_private.pem -pkeyopt rsa\_keygen\_bits:2048

openssl rsa -in rsa\_private.pem -pubout -out rsa\_public.pem

* Các dòng lệnh này sẽ tạo cho chúng ta 2 file:
* *rsa\_private.pem*: Chứa **private key**, được lưu trữ tại thiết bị IoT và được mã hóa bởi JWT.
* *rsa\_public.pem*: Chứa **public key,** được lưu trữ tại dịch vụ Google IoT Core để xác nhận mã JWT từ thiết bị IoT gửi lên.

+ Tạo cặp key sử dụng thuật toán Eliptic Curve

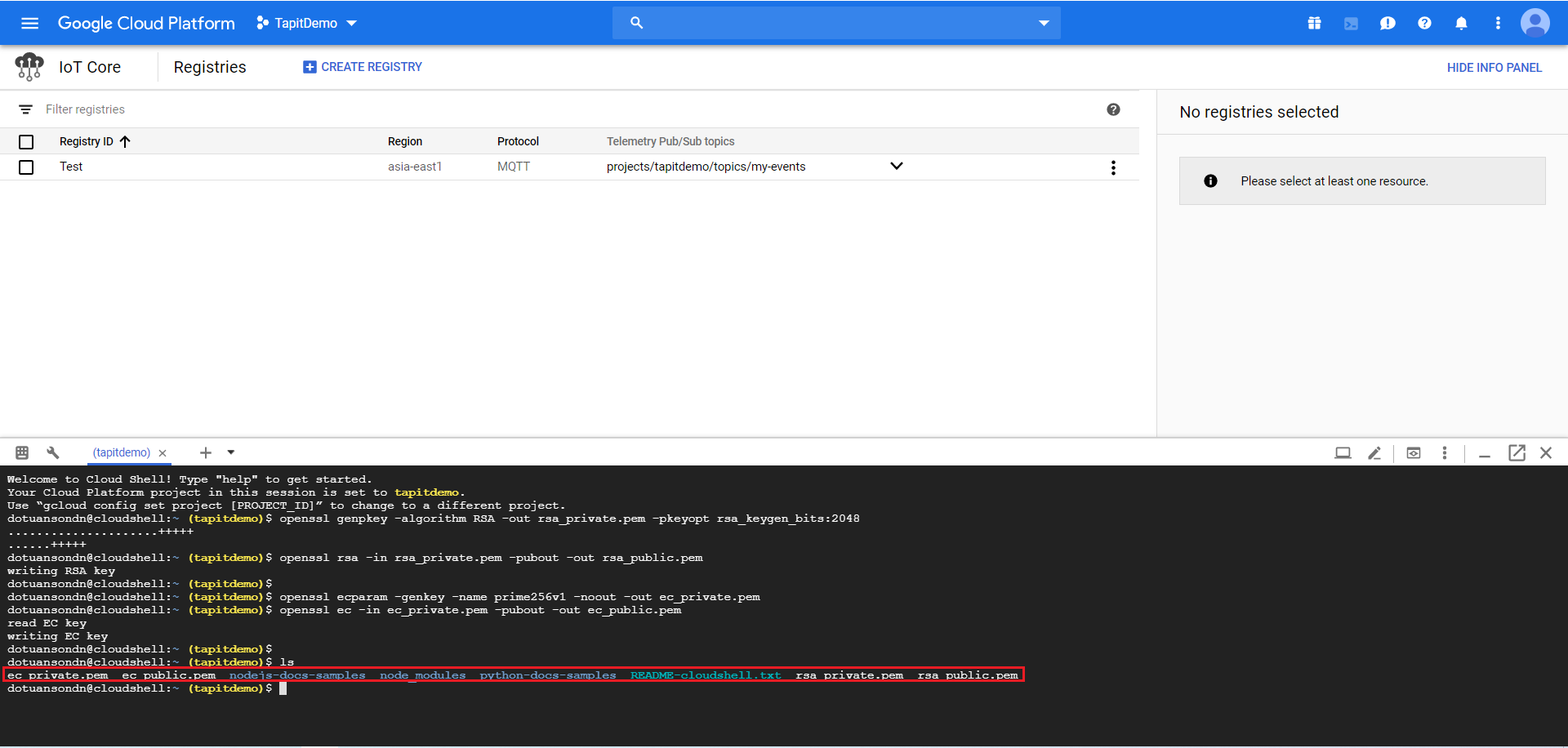
openssl ecparam -genkey -name prime256v1 -noout -out ec\_private.pem

openssl ec -in ec\_private.pem -pubout -out ec\_public.pem

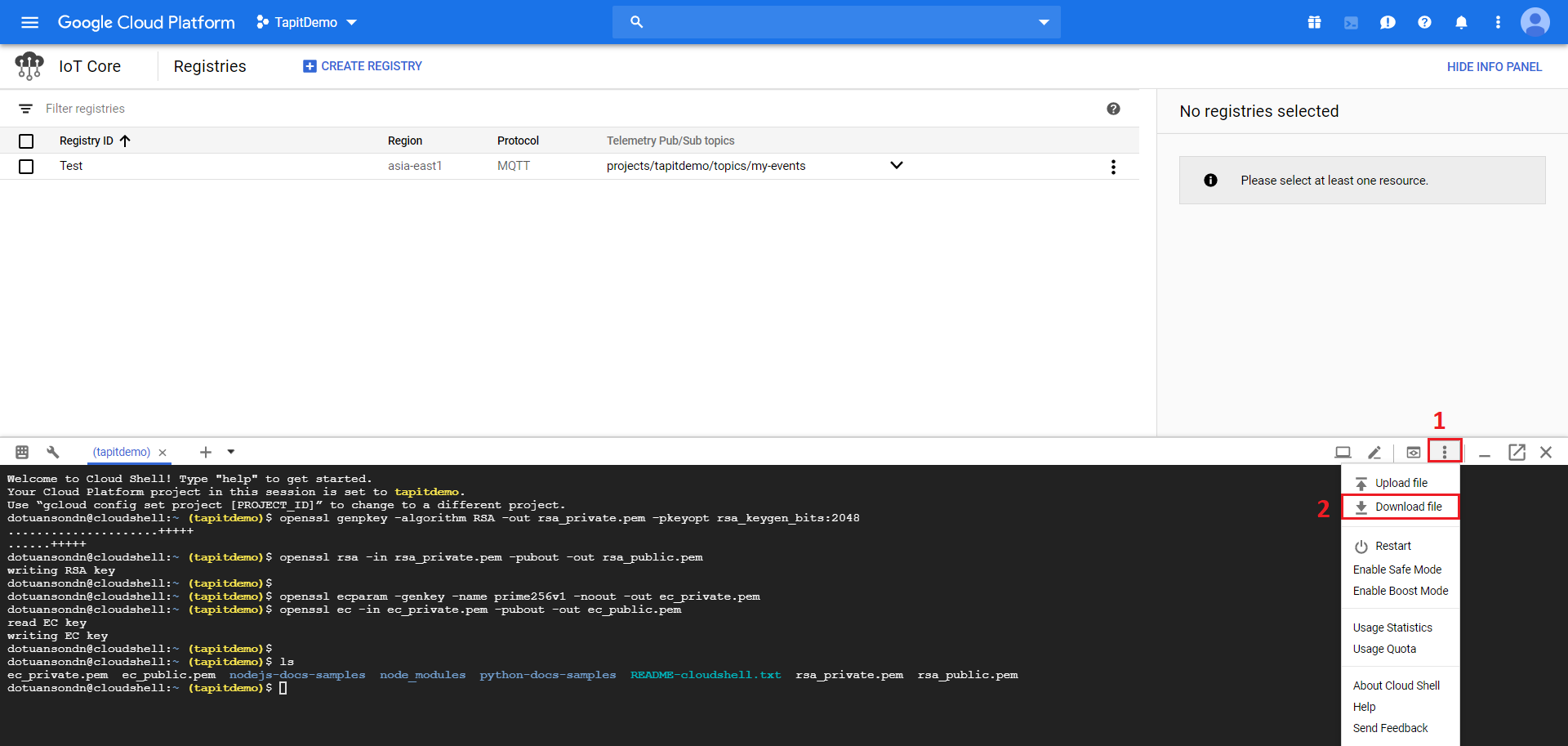
* Các dòng lệnh này sẽ tạo cho chúng ta 2 file:
* *ec\_private.pem*: Chứa **private key**, được lưu trữ tại thiết bị IoT và được mã hóa bởi JWT.
* *ec\_public.pem*: Chứa **public key,** được lưu trữ tại dịch vụ Google IoT Core để xác nhận mã JWT từ thiết bị IoT gửi lên.

*\* Ngoài ra Google IoT Core còn hỗ trợ cho chúng ta thêm cách tạo các cặp key sử dụng thuật toán RSA và EC sở hữu chứng chỉ X.509.*

* Để xem các file chúng ta vừa khởi tạo, tại cửa sổ dòng lệnh gcloud, chúng ta điền lệnh *ls.*



* Để tải các file này về máy, tại cửa sổ dòng lệnh gcloud, chúng ta thao tác như hình bên dưới, sau đó điền tên file (bao gồm cả đuôi file) mà chúng muốn tải về máy. VD: rsa\_public.pem, ec\_private.pem,...



- Ở đây mình tải 2 file rsa\_private.pem và rsa\_public.pem về máy để tiện cho việc đọc file và tạo chuỗi JWT sau này. Ngoài ra, các bạn còn có thể đọc trực tiếp nội dung file trên cửa sổ gcloud bằng lệnh *cat.* VD: cat rsa\_public.pem

**III. Hướng dẫn tạo Json Web Token (JWT)**

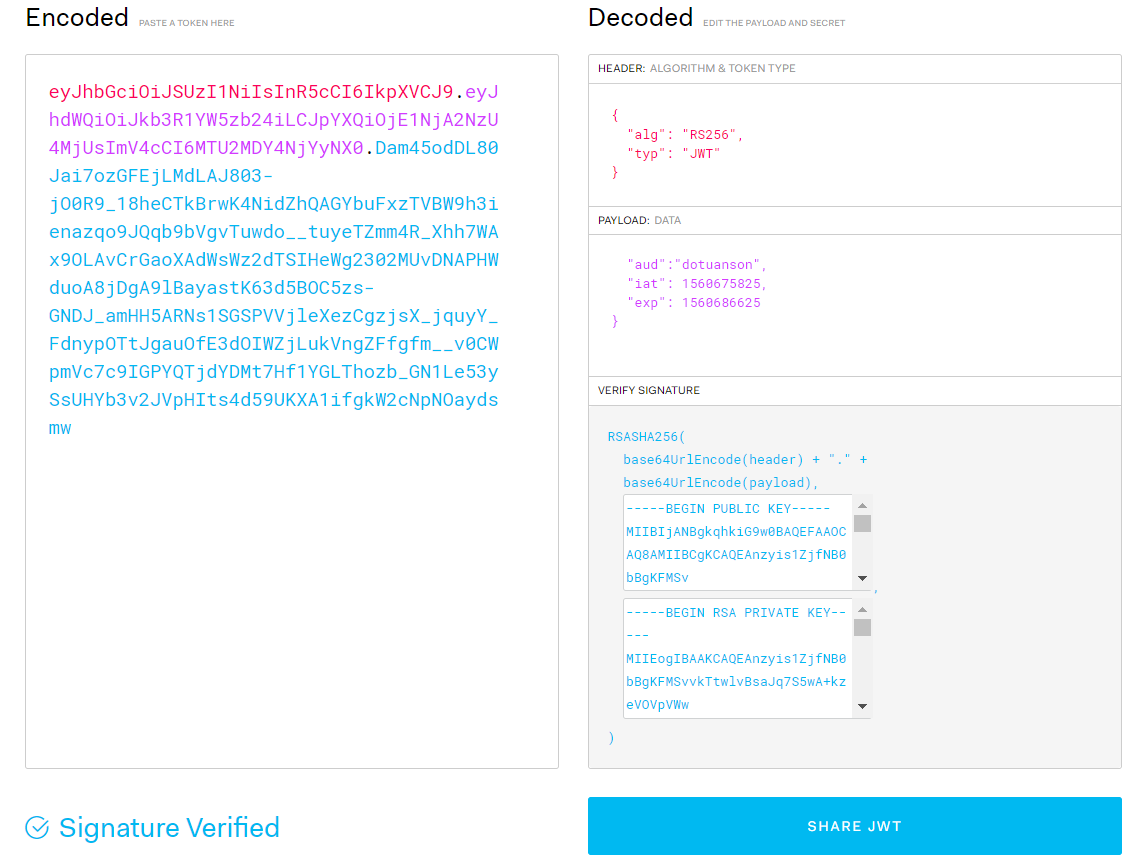
* Để tạo JWT thì chúng ta có thể sử dụng các thư viện có sẵn do Google cung cấp với những ngôn ngữ phổ biến như C++, Java, Python,... Ở đây thì mình sẽ không sử dụng các thư viện có sẵn này mà thay vào đó, mình sẽ tự tạo một chuỗi JWT thông qua tool online trên trang jwt.io, việc này sẽ giúp chúng ta nắm rõ hơn cách để tạo ra một chuỗi JWT thì cần những thông số gì.
* Để bắt đầu với JWT, chúng ta sẽ đi vào định nghĩa của nó:

JWT là là một đối tượng JSON được định nghĩa như một đoạn mã an toàn để trình bày một tập hợp thông tin giữa hai bên. Đoạn mã token bao gồm header, payload và signature.

* Nói cách khác, JWT là một chuỗi có định dạng:

*Header.Payload.Signature*

* Để có thể tạo ra một chuỗi JWT hợp lệ do Google IoT Core quy định, trong phần Decoded chúng ta khai báo các thông số tương ứng với 3 phần Header, Payload, Signature như sau.



*Khởi tạo chuỗi JWT sử dụng thuật toán RS256 thông qua jwt.io*

+ *Heade*r: Đây là thành phần tiêu đề với trường “alg” là thuật toán mã hóa và trường “typ” xác định đối tượng là JWT.

+ *Payload*: Đây là thành phần nội dung chứa thông tin mà chúng ta muốn thiết bị IoT trao đổi với Cloud. Google IoT Core yêu cầu chúng ta phải đặt vào phần này những trường sau:

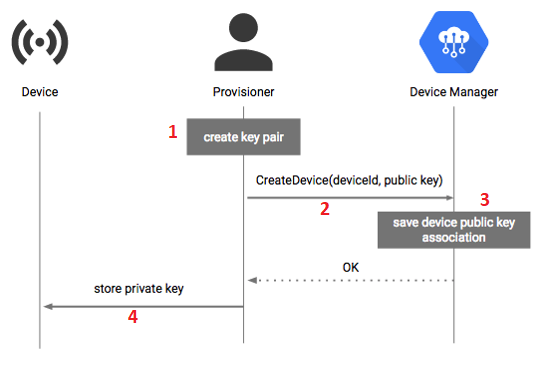
* “aud” (Audience): Project-ID của project mà chúng ta đang làm việc.
* “iat” (Issued At): Thời điểm chuỗi Json được tạo ra (Sử dụng định dạng timestamp).
* “exp” (Expiration): Thời điểm chuỗi Json hết hạn (Sử dụng định dạng timestamp).

+ Signature: Đây là thành phần chữ ký được tạo ra bằng cách kết hợp 2 phần (Header + payload), rồi mã hóa bằng một giải thuật encode nào đó, ở đây mình dùng RS256 nên sẽ phải sử dụng cặp public/ private key. Các bạn nhớ điền vào public/private key tương ứng đã tạo ở phần I.

* Sau khi điền hết các thông số trên vào phần **Decoded**. Ở phần **Encoded**, chúng ta đã có một chuỗi JWT hoàn chỉnh. Chúng ta sẽ lưu chuỗi token này vào trong máy để sau này dễ dàng sử dụng.
* Như vậy là chúng ta đã có thể tạo được một cặp Public/ Private key và một chuỗi JWT đầy đủ cho một thiết bị muốn kết nối tới Google IoT Core. Ở bài viết tiếp theo nhóm TAPIT IoTs sẽ hướng dẫn các bạn cách thiết lập cặp Public/ Private key và JWT đồng thời demo tổng quan một client muốn kết nối tới Google IoT Core sẽ trải qua những bước như thế nào. Chúc các bạn thành công!

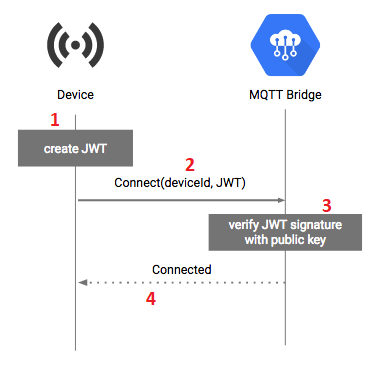
**Phần 3: Hướng dẫn thiết lập Public/ Private key và JWT**

**I. Quá trình cung cấp Public/ Private key cho thiết bị và cloud**



1. Đầu tiên chúng ta (**Provisioner**) đã tạo được cặp key (Public Key - Private Key) như ở phần trước.
2. Tiếp theo chúng ta sẽ tạo các *device* **(deviceID*)*** sử dụng **Cloud Platform Console**.
3. Sau đó chúng ta sẽ liên kết **Public Key** với các *device* nằm trong dịch vụ Google IoT Core.
4. Cuối cùng chúng ta sẽ lưu trữ **Private Key** vào thiết bị IoT tương ứng. Thiết bị sẽ trao đổi **Private Key** thông qua **JWT**.

**II. Quá trình cung cấp và xác thực chuỗi JWT cho thiết bị IoT thông qua giao thức MQTT**

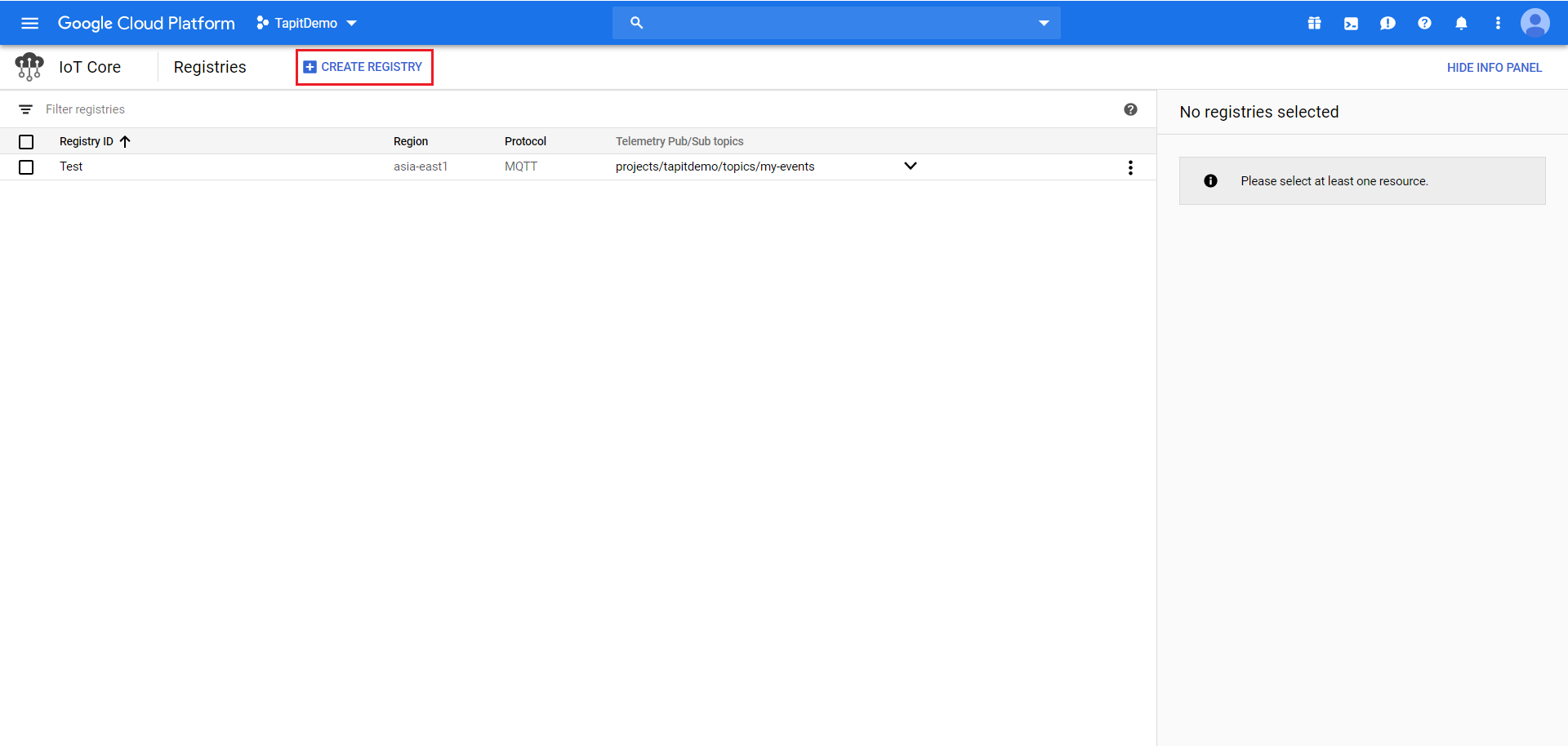


1. Thiết bị IoT phải tạo sẵn 1 chuỗi JWT. Chuỗi JWT này sẽ kết hợp với **Private Key** có **Public Key** tương ứng chứa trên IoT Core.
2. Khi kết nối tới Google IoT Core thông qua giao thức MQTT, thiết bị IoT sẽ chỉ định JWT là *password* trong gói tin **MQTT CONNECT**. Phần *username* có thể bỏ qua, tuy nhiên với một số thư viện MQTT client nó bắt buộc phải có trường username nếu không sẽ không gửi trường password. Vì thế để cho kết quả tốt nhất, chúng ta nên đặt vào trường *user name* một giá trị tùy ý.
3. Giao thức MQTT sẽ xác nhận JWT dựa trên Public Key mà chúng ta đã lưu trữ trên IoT Core.
4. Sau khi xác nhận chuỗi JWT thì giao thức MQTT sẽ được thiết lập giữa thiết bị IoT với Cloud. Giao thức MQTT sẽ đóng khi chuỗi JWT hết hiệu lực. Thời gian hiệu lực của 1 chuỗi JWT mà Google IoT Core cho phép là **1h -> 24h.**

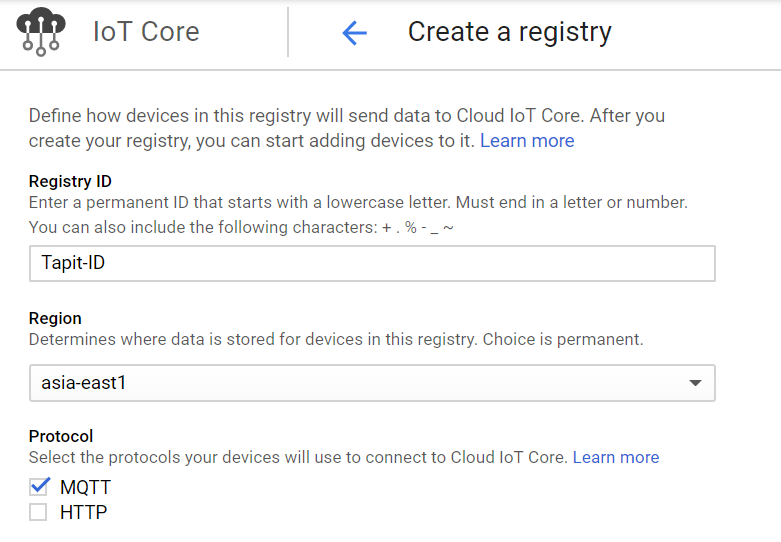
\* Mục đích của việc tạo thêm lớp bảo mật JWT này là làm cho cầu giao thức MQTT có thể tự động làm mới một phiên làm việc bằng cách thay thế một chuỗi JWT khác (thay đổi 2 trường “iat” và “exp” => thay đổi chuỗi JWT), phục vụ cho trường hợp dù hacker đã lấy được Password thì khi token hết hạn, hacker vẫn không thể nào thực hiện việc kết nối được.

**III. Hướng dẫn đăng ký thiết bị với registry và liên kết Public key với device**

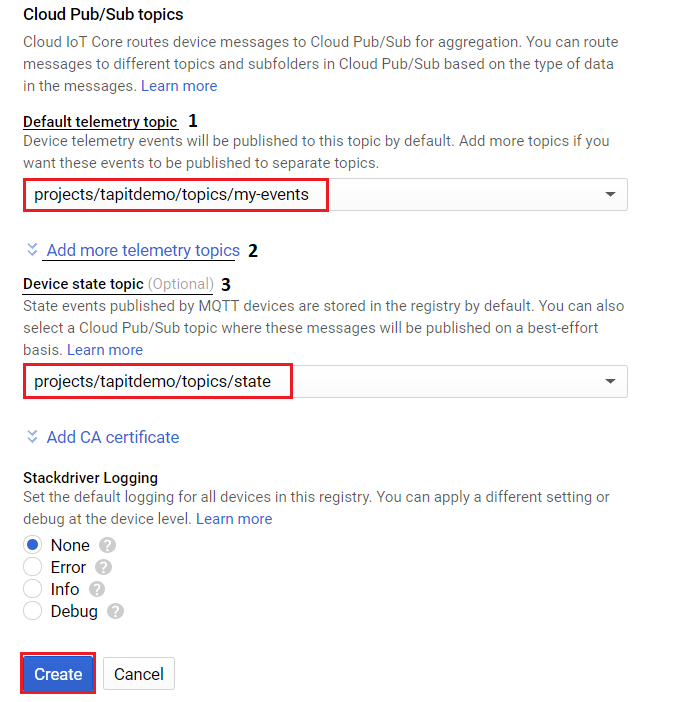
* Các thiết bị IoT kết nối đến dịch vụ Google IoT Core được xem là các “device” trong “registry”. Registry là nơi cho phép chúng ta lưu trữ các thiết bị IoT có những đặc điểm giống nhau cùng kết nối tới Google IoT Core.
* Để đơn giản hóa thì chúng ta có thể xem *project* như một ngôi nhà, bên trong ngôi nhà lại chia ra các căn phòng như phòng ngủ, phòng khách,... tương ứng với các phòng đó chính là các *registry*. Và những thiết bị gói gọn trong các căn phòng đó sẽ được xem là các *device*.
* Để bắt đầu, tại giao diện console của GCP các bạn chọn ***CREATE REGISTRY.***



* Chúng ta có thể tạo 1 Registry ID bất kỳ (NodeMCU, MyHome,..) nhưng không được trùng với bất kỳ registry nào khác nằm bên trong project mà chúng ta đang làm việc, ở đây mình ví dụ đặt tên là Tapit-ID. Tại phần *region* mình sẽ chọn *asia-east1* để tối ưu độ trễ của đường truyền dữ liệu. Còn *protocol* mình sẽ làm việc với giao thức MQTT (Đây là giao thức phổ biến trong việc tương tác machine-to-machine).



* Mỗi registry mà chúng ta đăng ký, Google IoT Core bắt buộc chúng ta phải liên kết tới một hay nhiều topics nằm trong một dịch vụ khác có tên là Cloud Pub/Sub (Mình sẽ nói về dịch vụ này ở bài viết sau). Tổ chức của Google IoT Core chính là khi một gói tin gửi từ thiết bị IoT lên Google IoT Core, Google IoT Core sẽ không lưu trữ gói tin đó mà nó sẽ chuyển tiếp gói tin đó tới một dịch vụ khác tên là Cloud Pub/Sub.
* Vậy nên nhiệm vụ của chúng ta khi tạo một registry bất kỳ là phải xác định trước gói tin đó sẽ được chuyển tiếp tới những topic nào nằm trong dịch vụ Cloud Pub/Sub.
* Ở đây, Google IoT Core đã định nghĩa sẵn cho chúng ta 3 loại topic nằm trong cloud Pub/Sub mà dựa vào đó chúng ta sẽ chia luồng dữ liệu sao cho hợp lý:



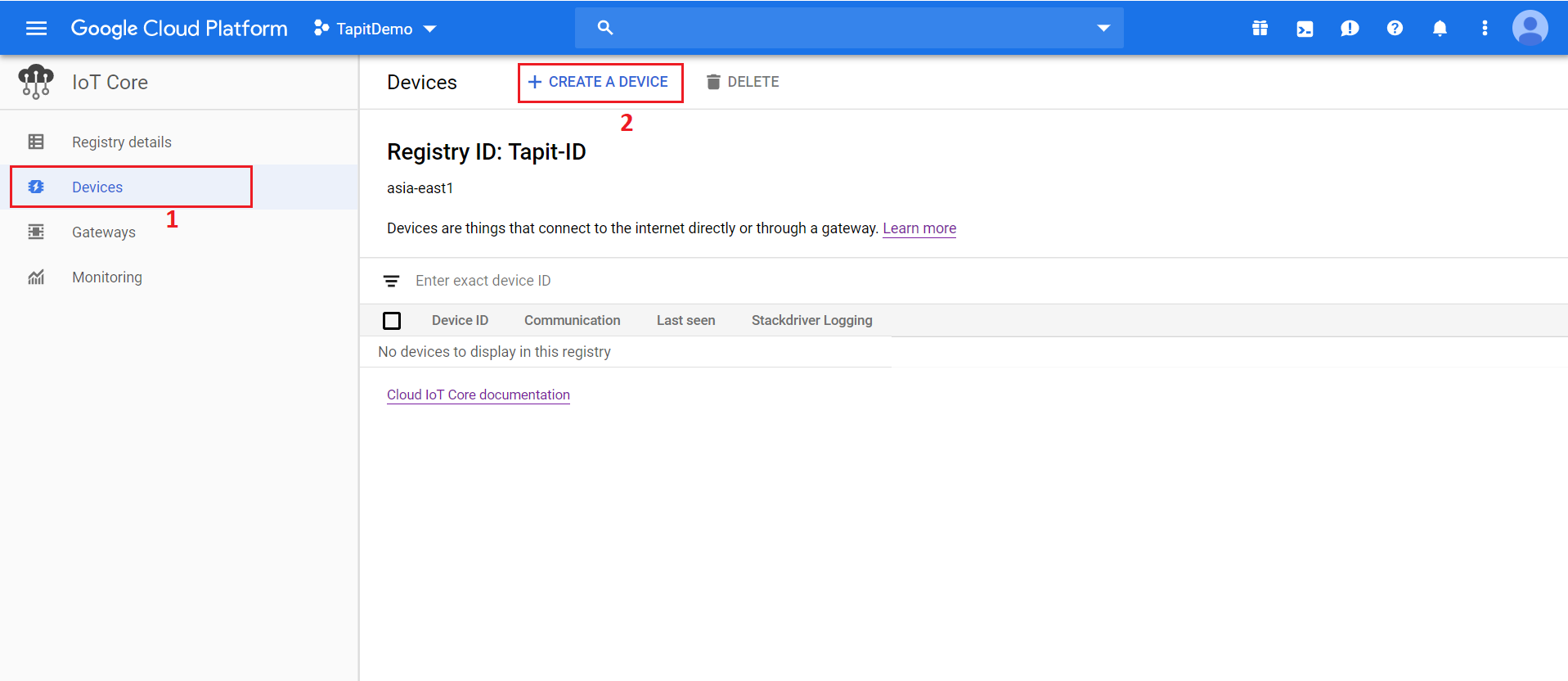
+ ***Default telemetry***: Topic này **bắt buộc** phải có khi khởi tạo 1 registry. Mục đích của nó là để publish các gói tin *telemetry event* không có subfolder hoặc không khớp với bất kỳ subfolder nào mà chúng ta tạo trong registry.

+ ***Device telemetry***: Topic này **không bắt buộc** phải có khi khởi tạo 1 registry. Tuy nhiên nếu chúng ta muốn khởi tạo thì các bạn có thể nhấn vào *Add more telemetry topics*. Mục đích của việc này là chúng ta có thể phân luồng dữ liệu riêng biệt từ một thiết bị. Lưu ý khi tạo một telemetry topic thì chúng ta bắt buộc phải chỉ định cho telemetry topic đó một subfolder cụ thể. Khi publish dữ liệu tới subfolder, dữ liệu sẽ chuyển tới topic tương ứng với subfolder đó. Bạn có thể chỉ định 1 topic trong Cloud Pub/Sub liên kết với nhiều subfolder, tuy nhiên ngược lại thì không đúng. Mỗi subfolder bắt buộc phải là **duy nhất trong một registry** và một subfolder không thể nằm trong nhiều topic khác nhau.

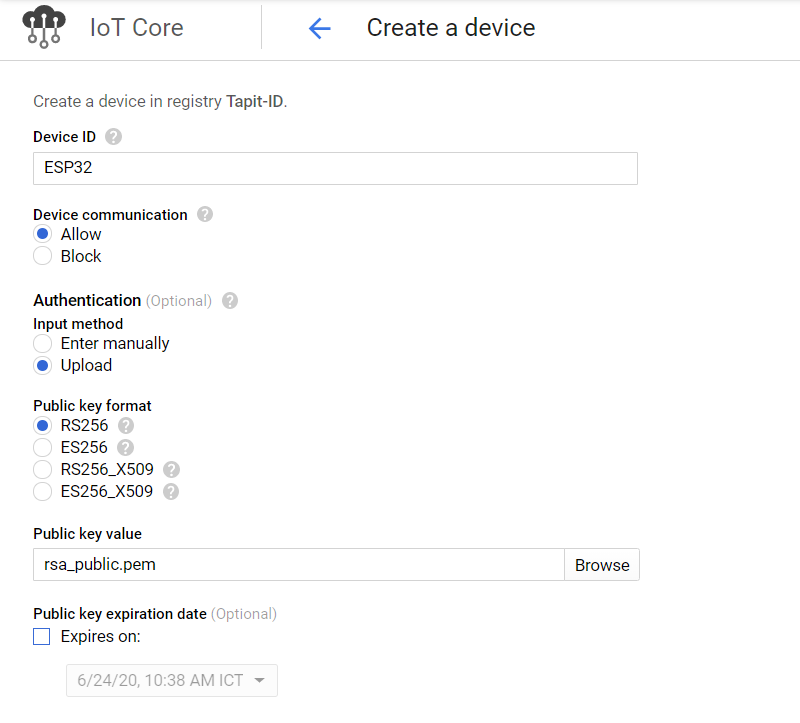
+ ***Device state***: Topic này **không bắt buộ**c phải có khi khởi tạo 1 registry. Nó có thể giống như một telemetry topic bình thường, hoặc có thể chỉ dùng cho việc publish dữ liệu trạng thái của thiết bị IoT.

\* Lợi ích của subFolder: Giả sử bạn mong muốn thiết bị của bạn publish nhiều loại dữ liệu khác nhau lên cloud như là nhiệt độ môi trường, độ ẩm hoặc dữ liệu logging,... Bằng cách hướng các luồng dữ liệu này vào từng subFolder riêng lúc cấu hình cho thiết bị của bạn, bạn có thể loại bỏ nhu cầu không cần phải tách dữ liệu thành các mục khác nhau sau khi dữ liệu đã ở trong Cloud Pub/Sub.

* Ở đây mình demo 2 topic là **default telemtry topic**với path *projects/tapitdemo/topics/my-events* và **device state topic**với path *projects/tapitdemo/topics/state*. Sau đó nhấn chọn **CREATE***.*
* Sau khi khởi tạo Registry xong, ở thanh menu registry của giao diện Console, các bạn chọn **Devices***,* sau đó các bạn nhấn chọn **CREATE A DEVICE***.*



* Tại giao diện đăng ký, các bạn có thể đặt tên bất kỳ cho Device ID nhưng lưu ý không được trùng tên với các Device ID khác trong registry, ở đây mình đặt tên Device ID là “ESP32”.



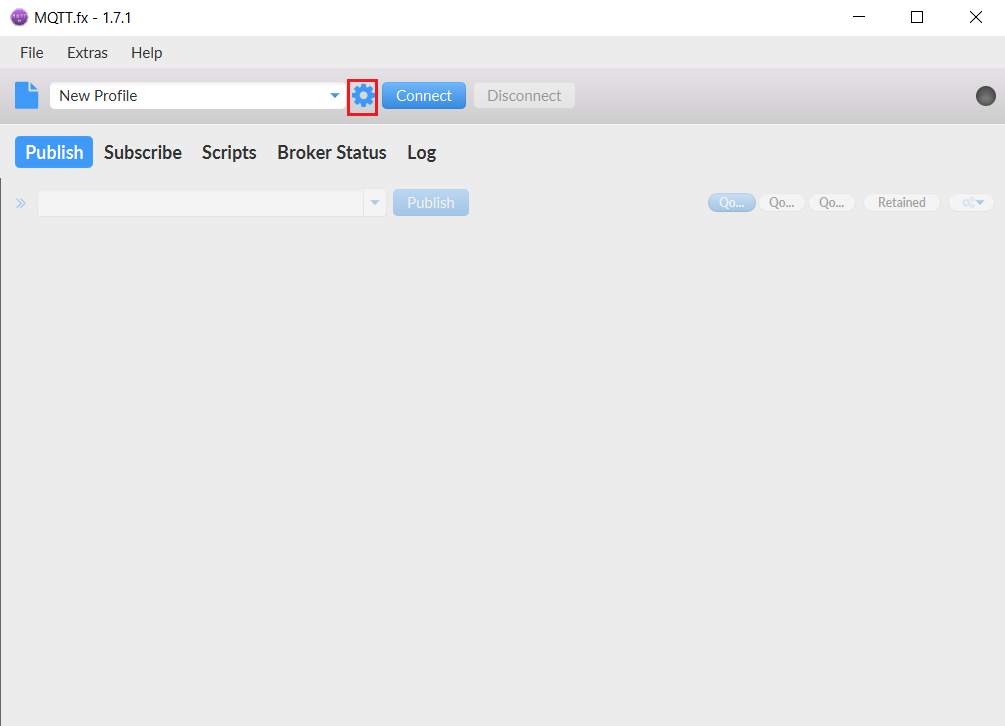
* Ở phần **Authentication**, các bạn có thể chọn Input method là Enter manually hoặc Upload. Ở đây mình chọn là Upload vì file Public Key mình đã tải về máy rồi nên mình chỉ cần tải lại lên Google IoT Core là ok. Lưu ý phần *Public key format* các bạn chọn mục tương ứng với thuật toán đã khởi tạo Public key. Các thông số còn lại chúng ta không cần quan tâm, các bạn cuộn xuống dưới rồi nhấn **CREATE**.

**IV. Hướng dẫn cung cấp chuỗi JWT cho một MQTT Client cụ thể**

- Ở phần trước chúng ta đã biết cách làm sao để tạo cặp Private/ Public Key, JWT và làm thế nào để có thể kết nối từ thiết bị IoT tới Cloud. Phần này chúng ta sẽ đi sâu hơn vào một trường hợp cụ thể để có thể hiểu rõ hơn về việc kết nối giữa thiết bị IoT với Cloud. Ở đây mình sẽ sử dụng công cụ MQTT.fx như là một MQTT Client.

- Đầu tiên các bạn tải file cài đặt MQTT.fx về máy, sau khi thực hiện xong các bước cài đặt, chúng ta sẽ đi vào bước thiết lập cấu hình tham số sao cho MQTT.fx có thể kết nối được với cloud.

- Tại giao diện MQTT.fx, các bạn nhấn vào biểu tượng răng cưa (biểu tượng setting).



* Ở thanh điều hướng bên trái, các bạn chọn New Profile và bắt đầu việc cấu hình.

+ ***Profile Name***: Các bạn có thể đặt bất kỳ tên gì cho Profile Name, ở đây mình đặt là GCP.

+ ***Profile Type***: Các bạn có thể chọn mục Google Cloud IoT cho việc cài đặt dễ dàng hơn, tuy nhiên nó sẽ bỏ qua bước khởi tạo JWT và Client ID. Vì mình muốn trực quan hơn cho mọi người nên mình sẽ demo mục MQTT Broker.

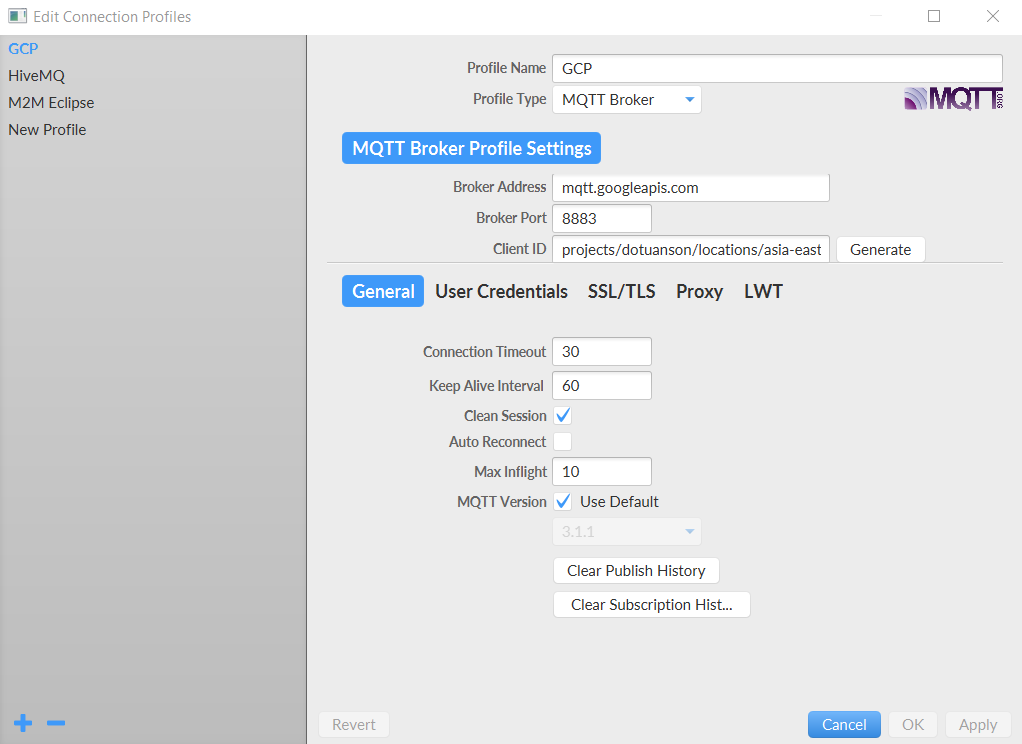
+ ***Broker Address***: mqtt.googleapis.com

+ ***Broker Port***: 8883 hoặc 443.

+ ***Client ID***: Các bạn thay thế các thông số bôi đỏ dưới đây, bốn thông số này có thể tìm thấy trên Google IoT Core sau khi đã thiết lập device và registry như ở các bài viết trước.

projects/**PROJECT-ID**/locations/**REGION**/registries/**REGISTRY-ID**/devices/**DEVICE-ID**

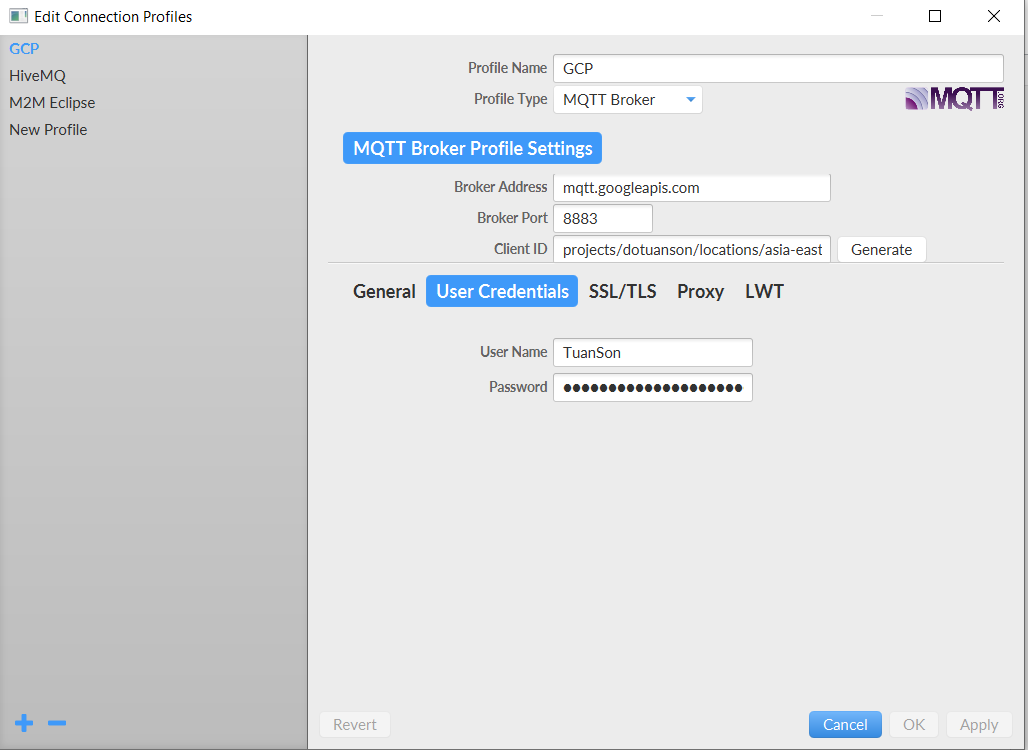
* Các thông số mục **General** chúng ta tạm không quan tâm, chúng ta sẽ chuyển hướng sang mục khác có tên là **User Credentials.**



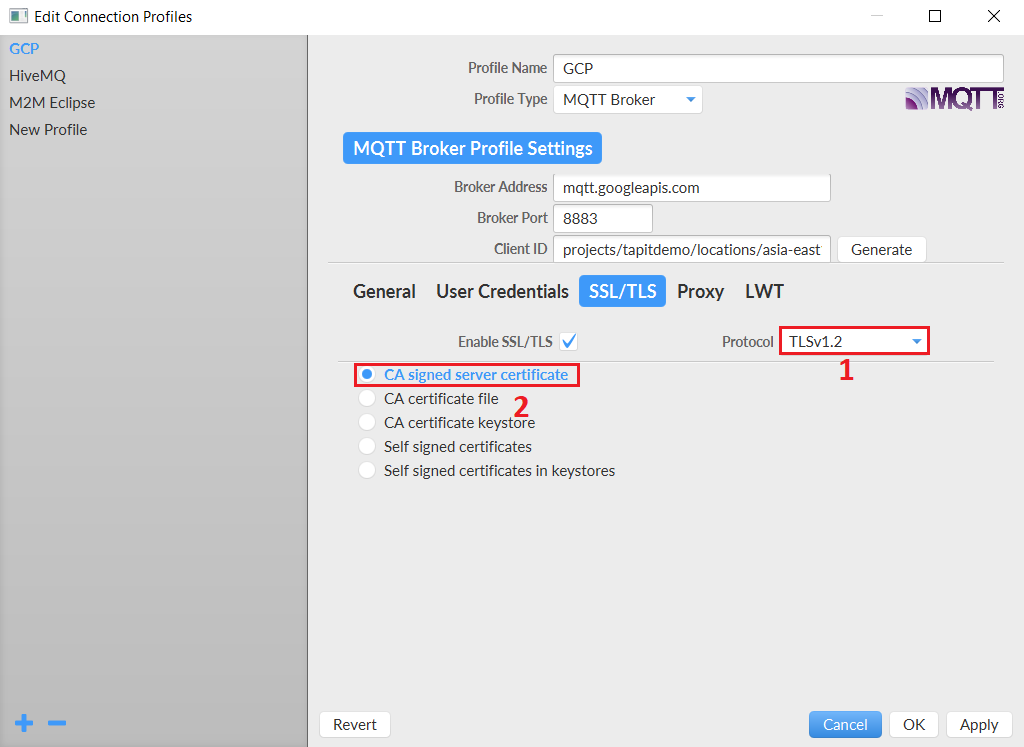
* Ở mục **User Credentials**, chúng ta có thực hiện cấu hình như sau:

+ ***User Name***: Các bạn có thể đặt tên bất kỳ, tại vì Google IoT Core không quan tâm trường này. Tuy nhiên, các bạn lưu ý **không được bỏ trống** mục này vì MQTT.fx quy định không cho phép gửi trường Password nếu như không có trường User Name, dẫn đến việc thực hiện kết nối bị lỗi.

+ ***Password***: Copy chuỗi JWT và paste vào mục này. Lưu ý chuỗi JWT phải nằm trong khoản thời gian “iat” và “exp” (Trong trường payload) thì Google IoT Core mới chấp nhận chuỗi JWT này.



* Cuối cùng chúng ta chuyển hướng sang mục **SSL/TLS**, hiện tại Google IoT Core chỉ hỗ trợ chuẩn **TLS1.2** nên ở mục **Protocol** chúng ta chỉ có thể chọn **TLSv1.2**. Tiếp theo ta **chọn CA signed server certificate**, đây là chứng chỉ tự server Google cấp cho chúng ta và Google sẽ tự xác thực chứng chỉ này mà không thông qua bên thứ ba nào.



* Sau khi đã cấu hình xong, chúng ta nhấn **OK**. Bây giờ thì chúng ta đã hoàn toàn có thể thực hiện việc kết nối tới Cloud thông qua giao thức MQTT. Để kết nối MQTT.fx tới Google IoT Core, các bạn nhấn chọn nút **CONNECT**.
* Như vậy là phần này mình đã giới thiệu cho các bạn cách thức để cấu hình cũng như thiết lập một client cụ thể khi muốn kết nối client đó tới Google IoT Core thông qua giao thức MQTT. Trong phần tới, chúng ta sẽ đi sâu vào cách thức hoạt động cũng như sử dụng hiệu quả việc tương tác giữa các thiết bị IoT với Google IoT Core.

**Phần 4: Tương tác với Google IoT Core thông qua giao thức MQTT và thao tác với các topic**

**I. Tương tác giữa MQTT Client với Google IoT Core**

* Để thấy được sự tương tác giữa các thiết bị IoT với Google IoT Core hoạt động như thế nào thông qua giao thức MQTT. Chúng ta sẽ sử dụng một MQTT Client (Ở đây ta dùng MQTT.fx) để thực hiện việc kết nối và trao đổi dữ liệu với Cloud. Việc trao đổi dữ liệu này sẽ được thực hiện trung gian thông qua các topic.
* Đầu tiên, chúng ta sẽ phân loại các kiểu dữ liệu do Google định nghĩa khi truyền từ thiết bị lên Cloud và từ Cloud truyền xuống thiết bị:

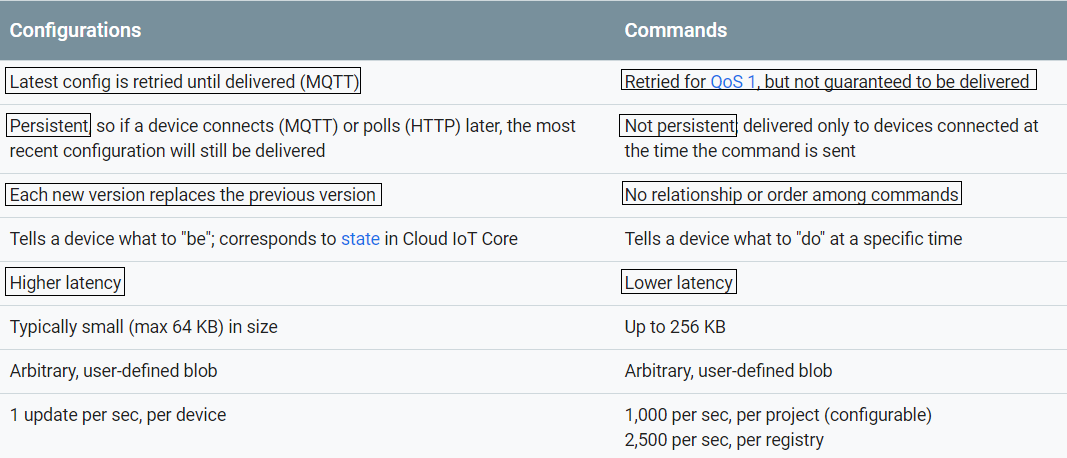
+ Thiết bị -> Cloud:

* *Telemetry*: Các dữ liệu về **môi trường xung quanh** mà thiết bị đo được như nhiệt độ, độ ẩm,...
* *State*: Các dữ liệu phản ánh **trạng thái hiện tại của bản thân thiết bị** như pin, ram trống, nhiệt độ chip,...

+ Cloud -> Thiết bị:

* *Configuration*: Các dữ liệu dùng để **cấu hình thiết bị** (Dựa vào dữ liệu state mà thiết bị báo cáo lên Cloud).
* *Command*: Các dữ liệu dùng để **ra lệnh** cho thiết bị thực hiện (Dựa vào dữ liệu telemetry mà thiết bị báo cáo lên Cloud).

\* So sánh giữa *Configurations và Commands*



* Từ các kiểu dữ liệu mà Google chia ra, chúng ta sẽ làm việc với 4 loại topics có path bắt đầu bằng /devices/DEVICE-ID/:

+ Các topic từ thiết bị -> Cloud:

* + - /devices/DEVICE-ID/events/{subFolders}
    - /devices/DEVICE-ID/state

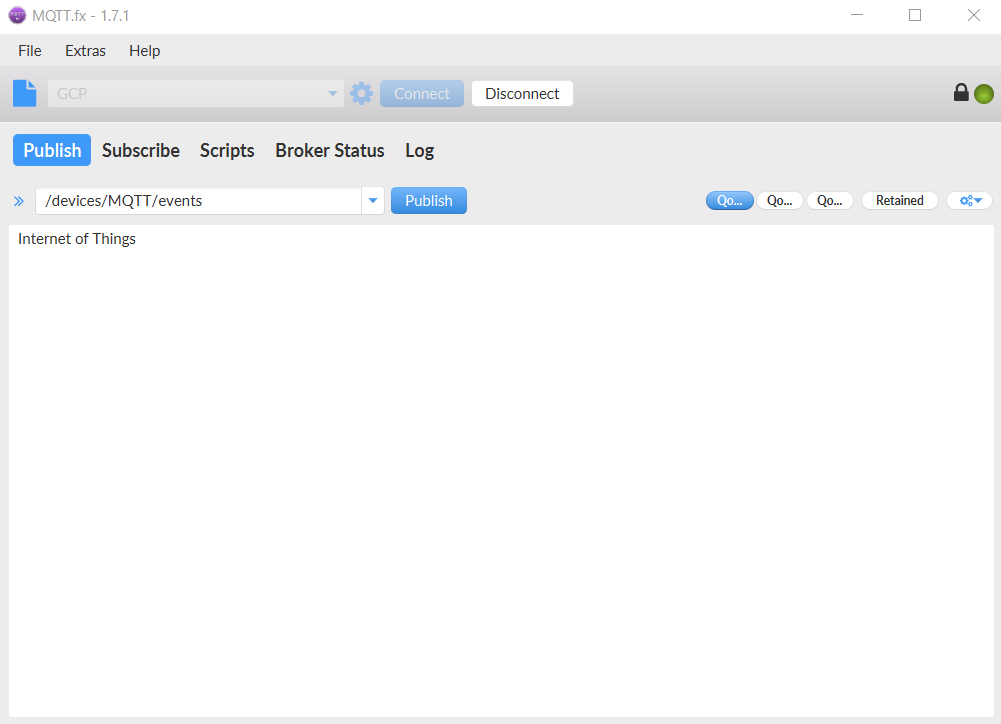
+ Các topic từ Cloud -> Thiết bị:

* + - /devices/DEVICE-ID/config
    - /devices/DEVICE-ID/commands/#
* Ở thiết bị IoT, Google IoT Core chỉ cho phép chúng ta **publish** dữ liệu vào 2 topics là */devices/DEVICE-ID/events/{subFolders}* và */devices/DEVICE-ID/state*, nếu chúng ta publish dữ liệu ngoài 2 topics này thì giao thức MQTT sẽ ngay lập tức ngắt kết nối.
* Còn về **subscribe** thì Google IoT Core cũng chỉ cho phép chúng ta subscribe vào 2 topics là */devices/DEVICE-ID/config* và */devices/DEVICE-ID/commands/#* (bắt buộc phải có wildcard #). Bằng việc subscribe vào wildcard topic này, thiết bị IoT không chỉ nhận dữ liệu từ topic */devices/DEVICE-ID/commands* mà nó còn nhận được cả từ subfolder */devices/DEVICE-ID/commands/{subFolder}*. Lý do cho việc này là tại vì hiện tại Google IoT Core chưa hỗ trợ cho thiết bị có thể subscribe vào 1 subFolder cụ thể nào cả.

**II. Thao tác với các topic trong Google IoT Core**

**1. Publish message từ thiết bị IoT lên Cloud**

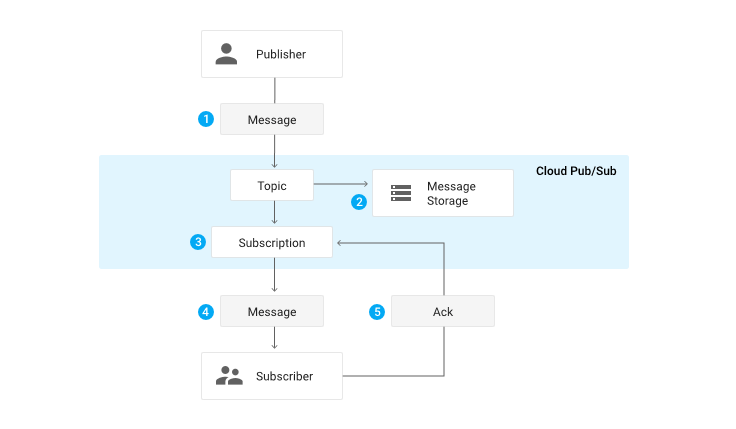
* Chúng ta sẽ publish 1 gói tin bất kỳ tới topic */devices/DEVICE-ID/events/{subFolder}* hoặctopic */devices/DEVICE-ID/state:*



* Sau khi nhấn chọn **Publish** thì gói tin sẽ thông qua giao thức MQTT mà gửi tới topic */devices/DEVICE-ID/events* nằm trong dịch vụ Google IoT Core. Tuy nhiên như mình đã nói ở bài viết trước trước, gói tin sẽ không được lưu trữ ở dịch vụ Google IoT Core, mà nó sẽ tự động chuyển tiếp và lưu trữ ở dịch vụ khác có tên là Cloud Pub/Sub. Các gói tin này sẽ lưu trữ trong Cloud Pub/Sub với topic có định dạng:

*project/PROJECT-ID/topics/{Topic you defined in registry}*

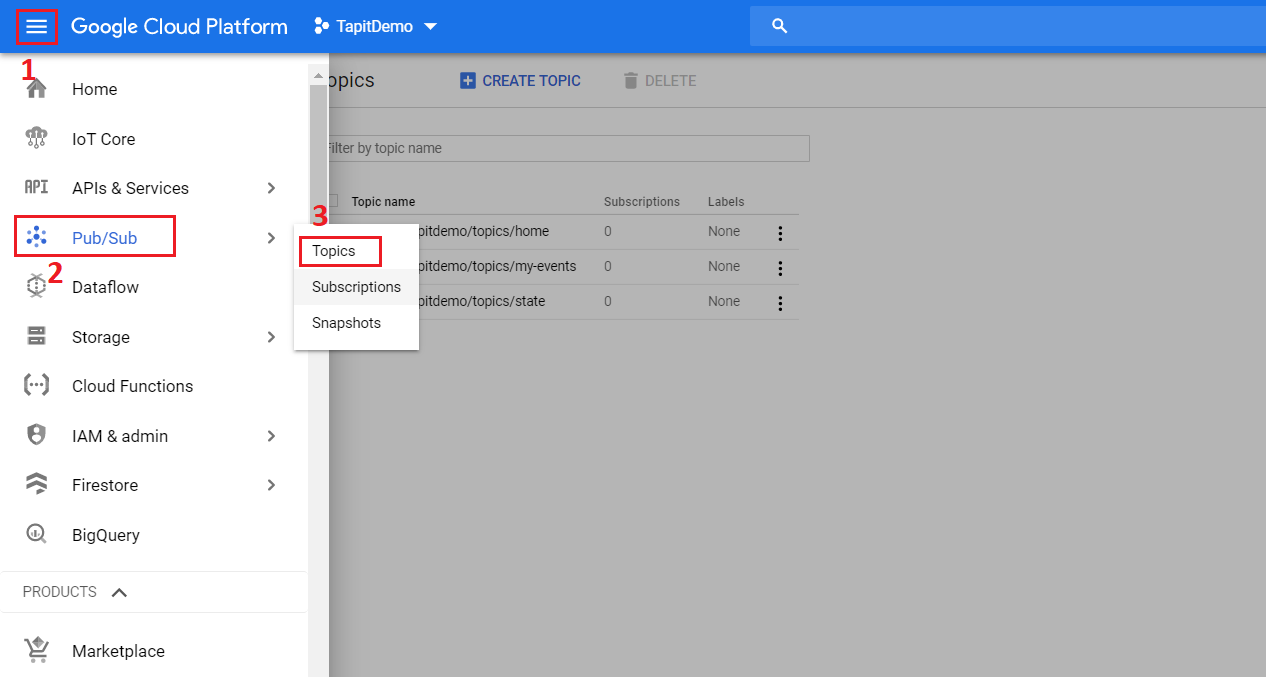
* Tiếp theo, để có thể quan sát các gói tin mà thiết bị IoT gửi lên Cloud, chúng ta có các cách sau:
* Sử dụng gcloud command-line được tích hợp sẵn có trong [Cloud SDK](https://cloud.google.com/sdk/) của Google.
* Sử dụng các [thư viện client](https://cloud.google.com/apis/docs/cloud-client-libraries) được Google hỗ trợ như (java/python/nodejs/golang)
* Sử dụng **Console** của Cloud Pub/Sub để tạo **Topic** và **Subscription**
* Để đơn giản, phần này mình sẽ sử dụng giao diện **Console** của Cloud Pub/Sub để demo cho các bạn xem các gói tin từ thiết bị gửi lên Cloud (Cụ thể thiết bị ở đây là phần mềm MQTT.fx).
* Vì mặc định khi chúng ta khởi tạo một registry trong Google IoT Core thì Google bắt buộc nó phải liên kết với một hay nhiều topic khác trong Cloud Pub/Sub, vậy nên thực tế chúng ta đã tạo ra được các topic trong Cloud Pub/Sub trong khi thao tác với Google IoT Core. Việc của chúng ta bây giờ là tạo **Subscription** cho các topics này. **Subscription** chính là tên đại diện cho các dòng tin nhắn từ một hoặc nhiều topic cụ thể nào đó.
* Vì bài viết này chỉ giới hạn trong nền tảng dịch vụ Google IoT Core, thế nên mình sẽ không trình bày quá sâu về Cloud Pub/Sub. Các bạn có thể tham khảo các bài viết trên mạng hoặc document của Google về Cloud Pub/Sub. Ở đây mình xin tóm tắt tổng quan về Cloud Pub/Sub để các bạn dễ hình dung.

****

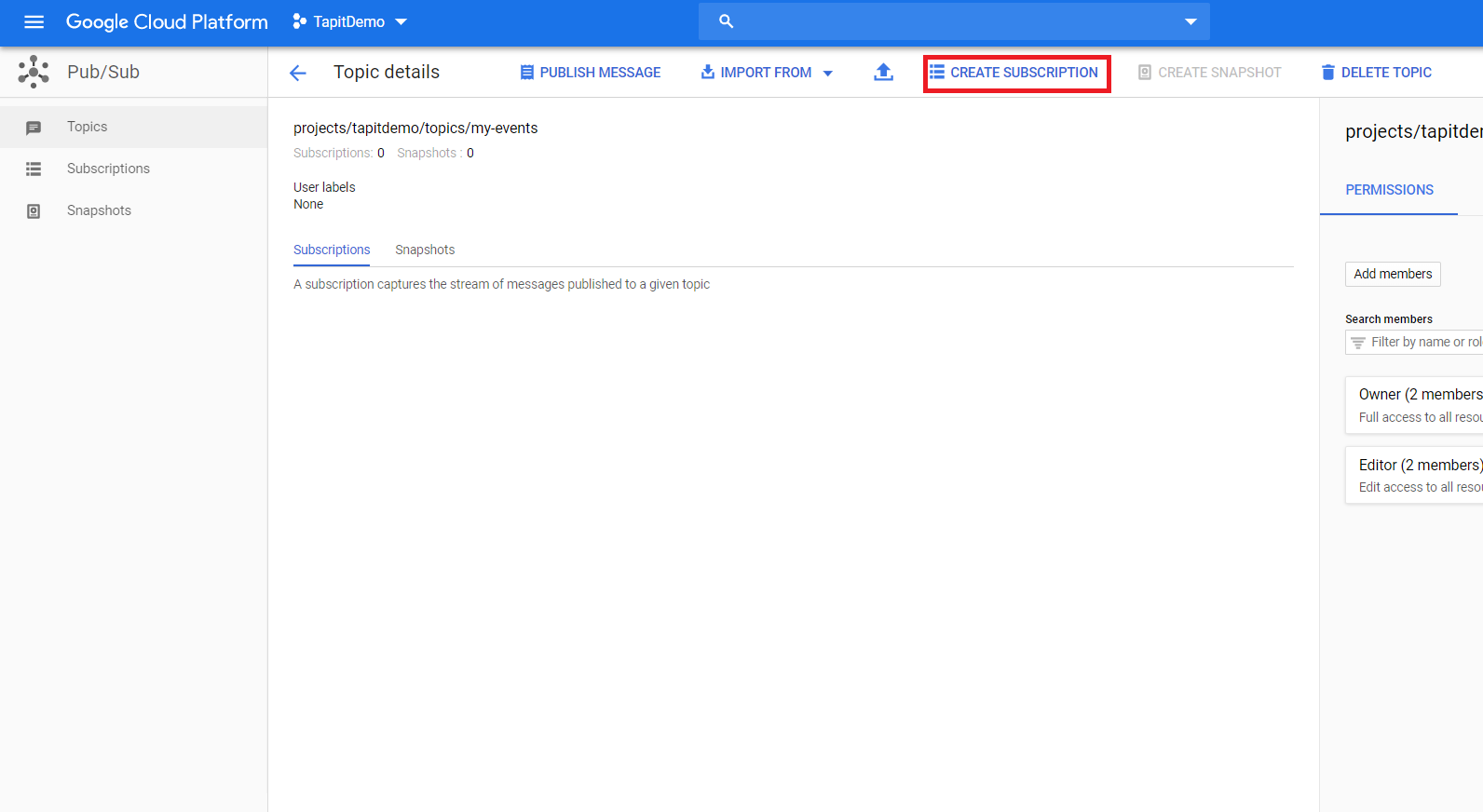
*Luồng xử lý message trong Cloud Pub/Sub*

1. **Pulisher**tạo **Topic**trêndịch vụ **Cloud Pub/Sub** và gửi messsagelên **Topic**.
2. **Message**sẽ được lưu trữ tại **Message Storage**cho đến khi tin nhắn được chuyển đến cho **subscribers**và nhận được phản hồi **(ACK)** từ **subscribers.**
3. **Cloud Pub/Sub**  ẽ chuyển tiếp messagetừ **topic**đến hàng đợi của tất cả **subscriptions**của nó. Mỗi **subscriptions**sau khi được tin nhắn, nó sẽ đẩy (**push**) **message**đó về cho **subscriber**. Hoặc **subscriber**có thể tự kéo (**pull**) **message**từ một **subscription**nào đó.
4. **Subscriber**nhận được messageđang chờ xử lý từ **subscription**.   
   **Subscriber**sẽ xác nhận (**ACK**) từng tin nhắn từ **subscription**.
5. Khi **subscription**nhận được tin nhắn xác nhận (**ACK**) của **message**đó thì thì **message**đó sẽ được xóa khỏi hàng đợi của **subscription.**

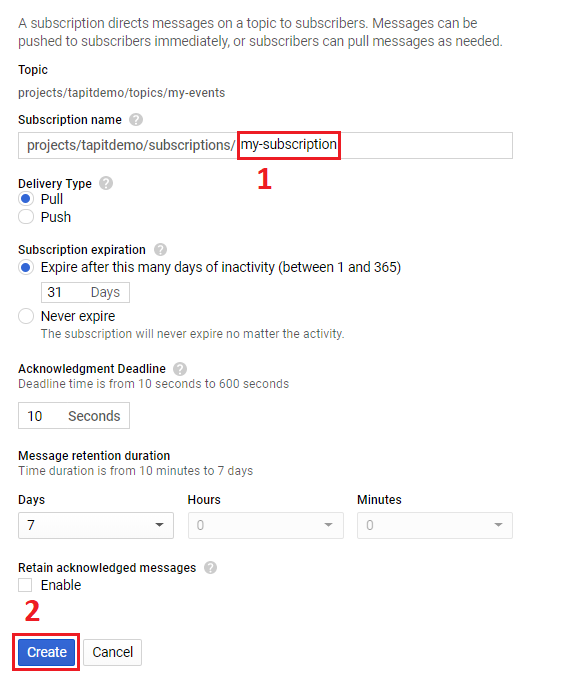
* Sau khi tổng quan về mô hình của Cloud Pub/Sub, các bạn có thể dễ dàng thấy rằng để đọc được các gói tin trong các topic, các bạn cần tạo cho topic đó một hoặc nhiều subscription.
* Tại giao diện Console, các bạn chuyển hướng tới dịch vụ Cloud Pub/Sub như sau:



* Giao diện Console sẽ hiển thị các topic name. Ở đây trên giao diện của các bạn chính là tên các topic đã tạo trong khi thao tác với Google IoT Core khi đăng ký một registry.
* Ở đây mình có một topic tên là *projects/tapitdemo/topics/my-events* mà mình đã tạo ở phần trước, đây chính là topic trong Cloud Pub/Sub lưu trữ các dữ liệu *telemetry event* từ thiết bị IoT gửi tới, chúng ta sẽ nhấn chọn topic này. Ngay lập tức trang console sẽ trả cho ta thông tin về topic, tiếp tục các bạn chọn **CREATE SUBSCRIPTION** như hình dưới:



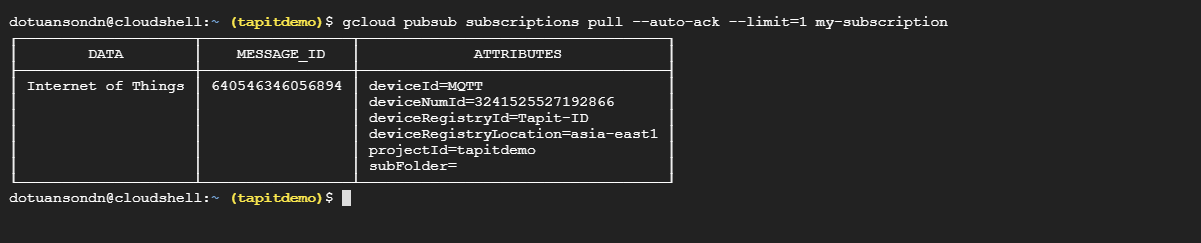
* Các bạn có thể đặt bất kỳ tên gì ở trường **Subscription Name**, ở đây mình đặt là *my-subscription*. Các thông số khác các bạn không cần quan tâm, chúng ta sẽ cuộn xuống và nhấn chọn **CREATE**.



* Sau khi đã khởi tạo xong subscription cho topic mà chúng ta muốn quan sát message. Các bạn mở **gcloud command-line** của google ra, sau đó các bạn điền command-line này vào gcloud:

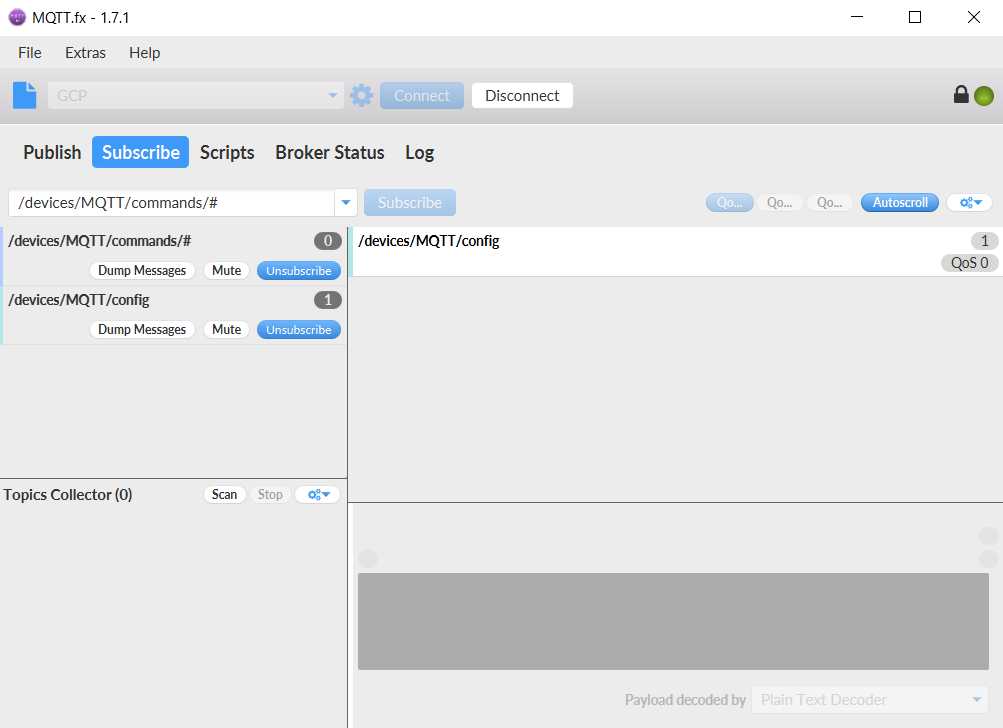
*gcloud pubsub subscriptions pull --auto-ack --limit=1 my-subscription*

* Dòng lệnh này sẽ trả về cho chúng ta gói tin “Internet of Things” nằm trong *my-subscription* với các **ATTRIBUTES** và **MESSAGE\_ID** như sau:

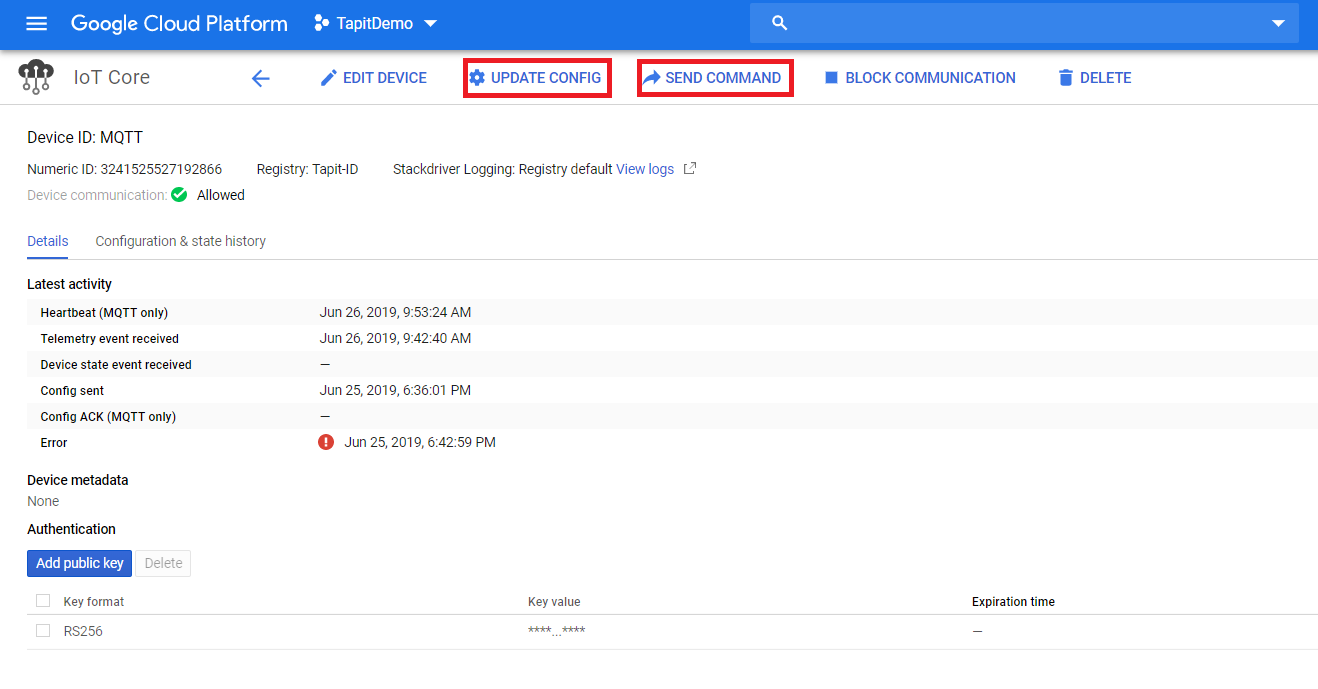


**2. Subscribe message từ Cloud xuống thiết bị IoT**

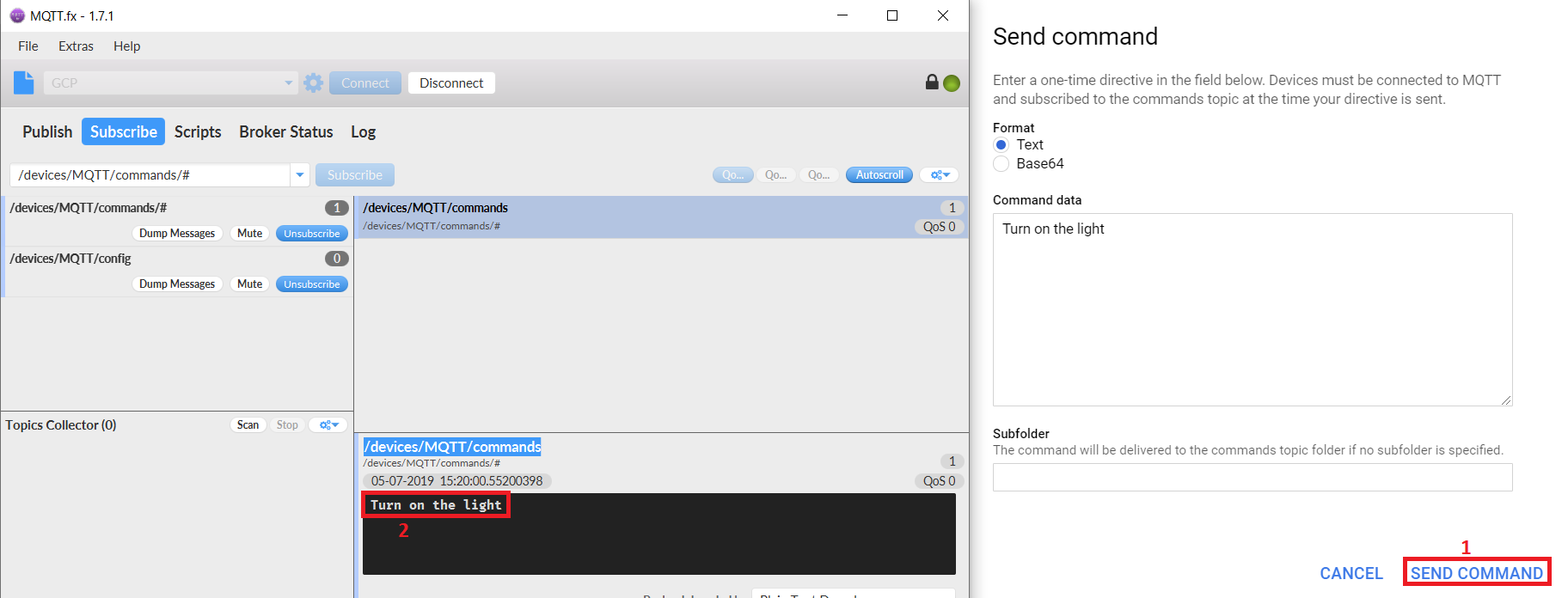
* Google IoT Core cho phép MQTT Client subscribe 2 topics:



* Để có thể gửi dữ liệu command hoặc config từ cloud xuống thiết bị của bạn, chúng ta có thể dùng **Console**, **gcloud** hoặc thông qua **API** mà Google cung cấp. Để trực quan, mình sẽ sử dụng **Console** của Google.
* Đầu tiên chúng ta vào GCP Console, ở thanh điều hướng bên trái, chúng ta chọn dịch vụ **IoT Core**. Chọn **registry ID** mà chúng ta muốn thao tác. Ở thanh registry menu bên trái, chúng ta nhấn **Devices** và chọn **device ID** mà chúng ta muốn gửi command hoặc config.
* Để gửi command hoặc config, chúng ta chọn các mục tương ứng như hình sau:



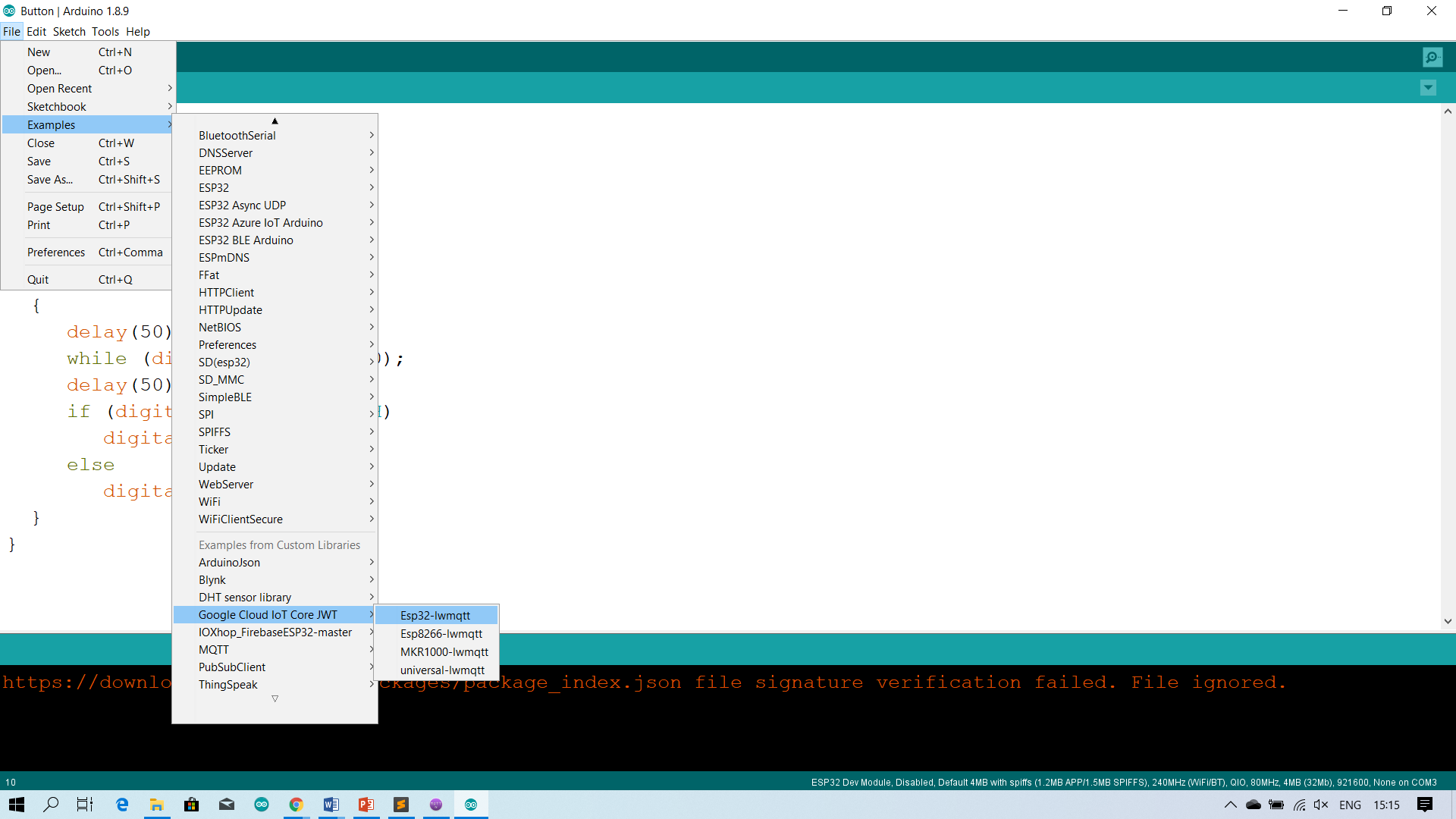
* Ở đây mình ví dụ sau khi MQTT.fx đã subscribe vào topic có path */devices/MQTT/commands/#* thì khi ta **SEND COMMAND** gói tin “Turn on the light” từ Google IoT Core xuống cho MQTT.fx, chúng ta sẽ nhận được chính xác gói tin đó ở MQTT.fx



* Từ phần 1 tới giờ chúng ta đã đi tìm hiểu về các khái niệm cũng như làm việc với các thiết bị thông qua giao thức MQTT nhưng chỉ dừng lại ở mức demo một tool MQTT client. Trong phần tiếp theo, nhóm TAPIT IoTs sẽ hướng dẫn các bạn cách kết nối một MCU là ESP32 với Google IoT Core xem có gì khó khăn không nhé.

**Phần 5: Sử dụng ESP32 thực hiện việc kết nối tới Google IoT Core thông qua giao thức MQTT**

* Để kết nối ESP32 với Google IoT Core, chúng ta sử dụng thư viện “Google Cloud IoT Core JWT”, đây là thư viện chuẩn do lập trình viên của google cung cấp, chúng ta có thể thoải mái sử dụng mà không lo về lỗi.
* Link tải: <https://github.com/GoogleCloudPlatform/google-cloud-iot-arduino>
* Sau khi tải thư viện về và cài đặt, chúng ta sẽ sử dụng example để thực hiện việc cấu hình
* Các bạn chọn **File** -> **Examples** -> **Google Cloud IoT Core JWT** -> **Esp32-lwmqtt**



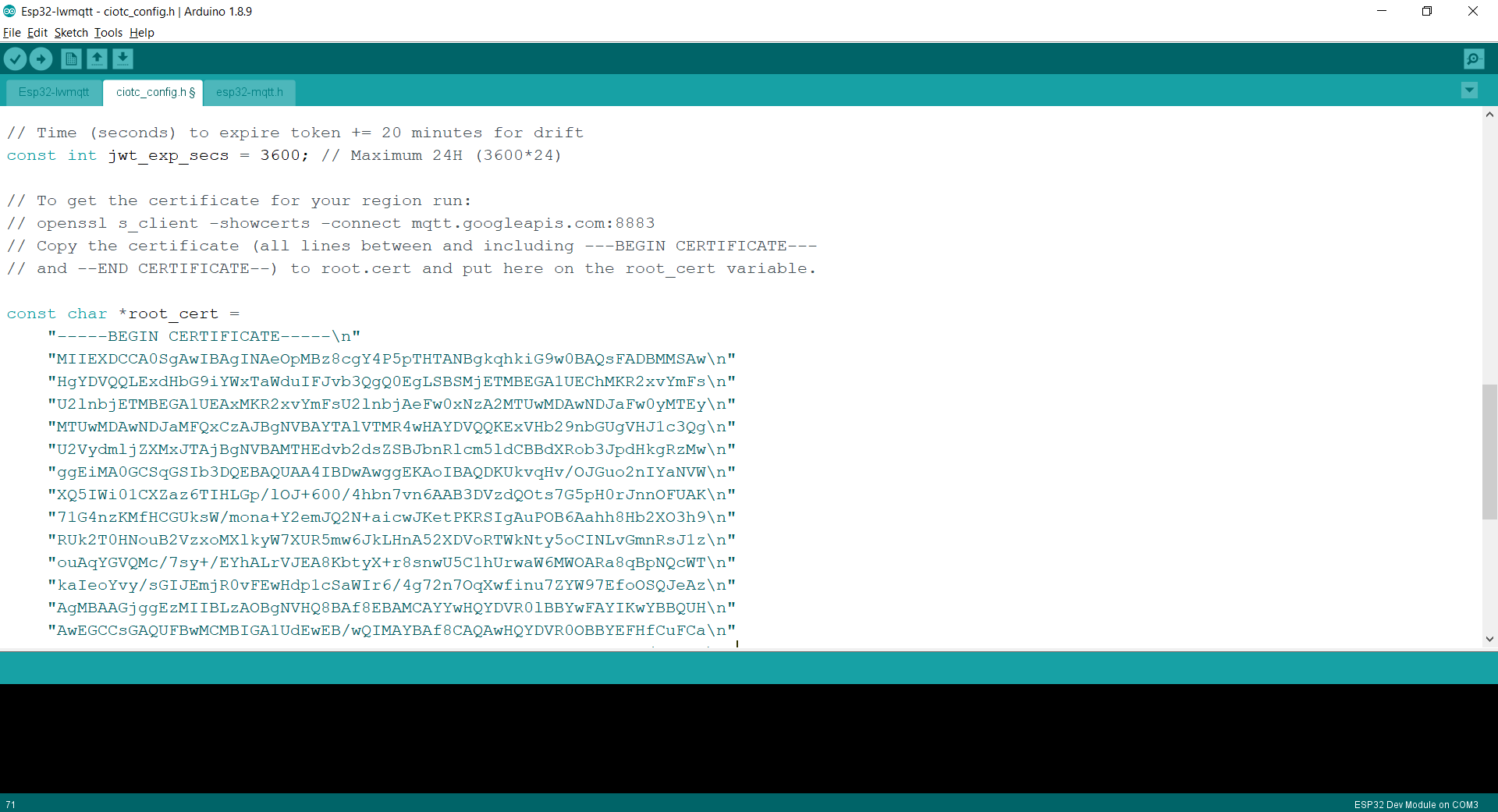
* Example đã cho chúng ta sẵn 3 file, chúng ta chỉ cần quan tâm tới file *ciotc\_config.h*. Mọi thông số mà chúng ta thay đổi trong file này sẽ giúp ESP32 kết nối được với Cloud.
* Trước tiên chúng ta sẽ thay đổi 6 thông số như hình bên dưới, các bạn lưu ý ở phần này mình sử dụng một thiết bị khác (**ESP32**) để kết nối tới Google IoT Core nên mình sẽ tạo thêm một device nữa ở trong registry.



* Sau đó chúng ta thay đổi thông số của biến *private\_key\_str* dựa vào file private key. Việc sinh ra chuỗi kí tự hex như hình bên dưới thì Google đã command-line hướng dẫn cho chúng ta (Lưu ý google sử dụng thuật toán sinh khóa EC để tạo).



* Tiếp theo ta có thể thay đổi biến *jwt\_exp\_secs*, đây là biến xác định thời gian có hiệu lực của 1 chuỗi JWT (Việc sinh chuỗi JWT thì google đã viết sẵn cho chúng ta).
* Cuối cùng là thay đổi thông số của biến *root\_cert*. Việc sinh ra chuỗi kí tự này thì Google đã command-line hướng dẫn cho chúng ta như hình bên dưới.

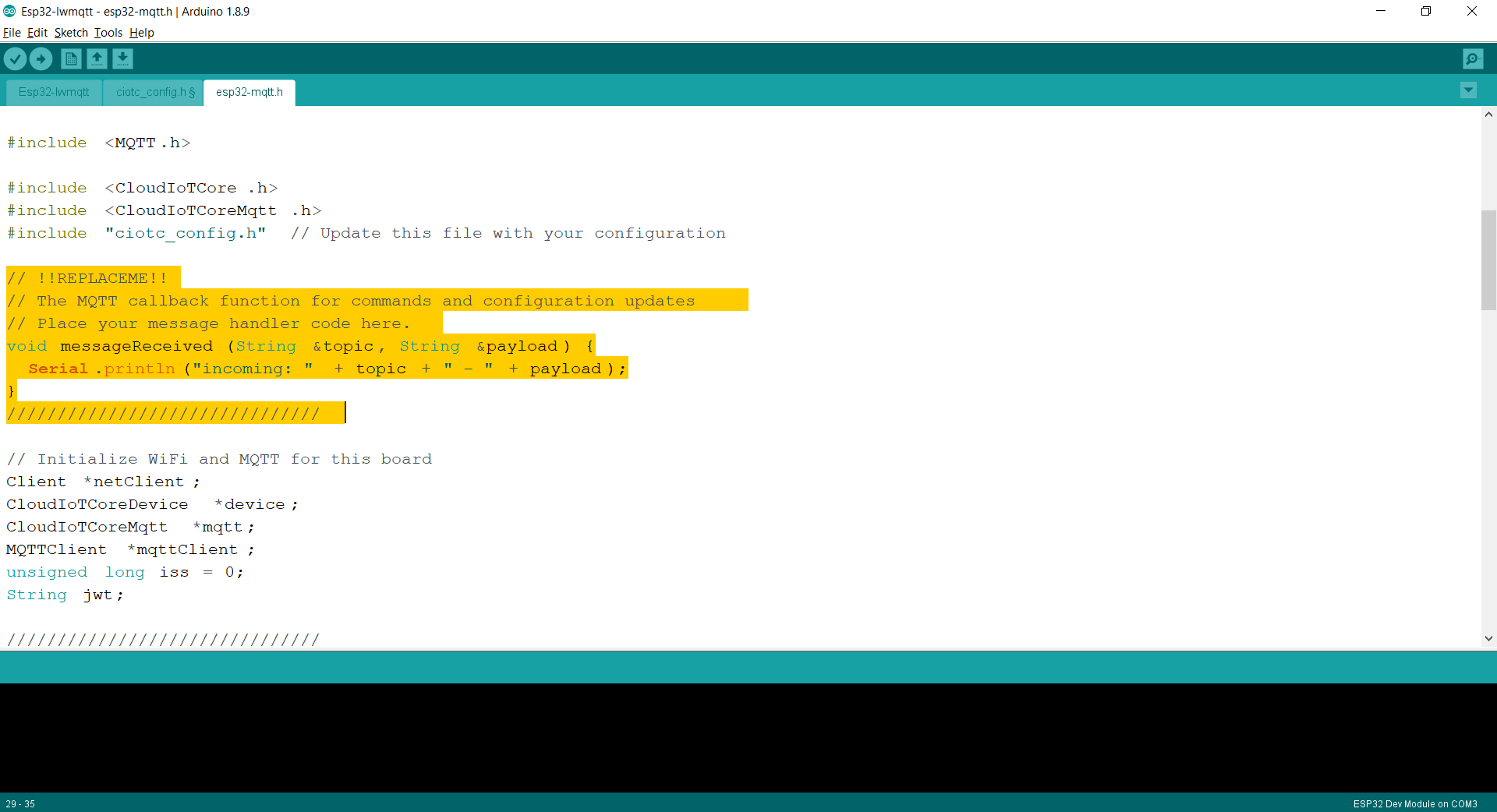


Sau khi đã cấu hình cho file ciotc\_config.h xong. Chúng ta đã có thể thực hiện việc kết nối ESP32 tới Google IoT Core.

* Để Publish dữ liệu từ ESP32 -> Cloud ta chỉnh sửa code ở các dòng này (Nằm trong file Esp32-lwmqtt)



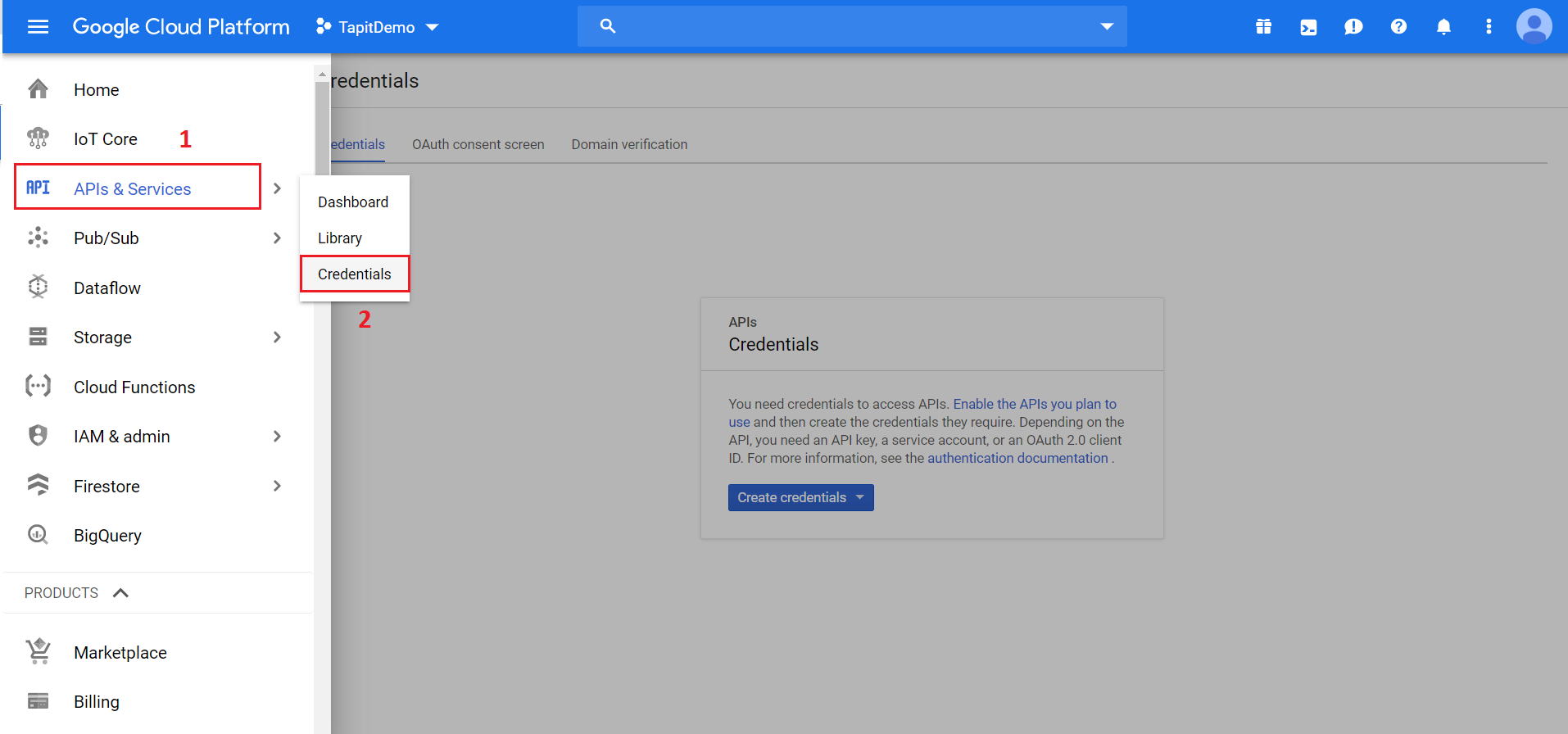
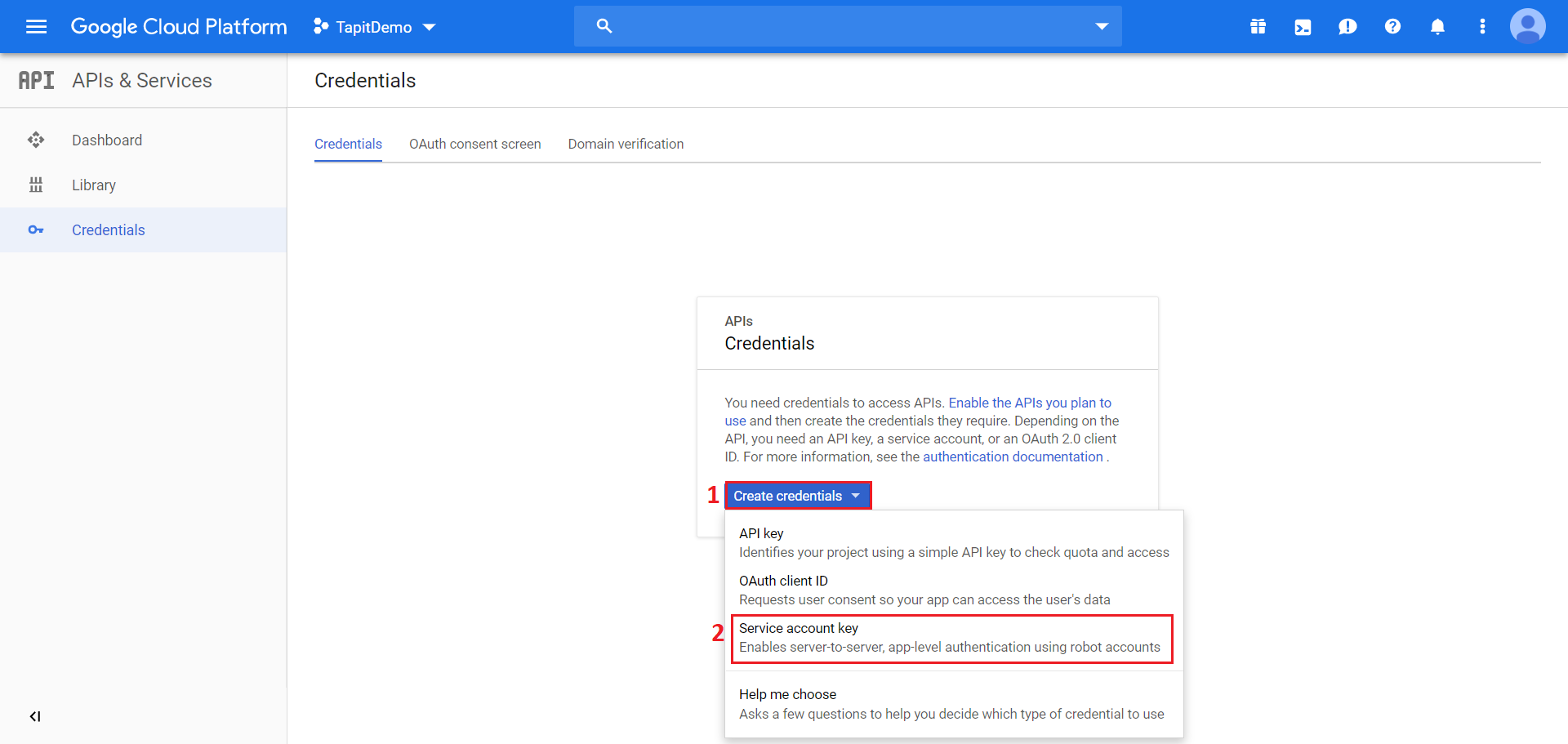
* Để Subscribe dữ liệu từ Cloud -> ESP32 chúng ta chỉnh sửa code bên trong hàm messageReceived() (Nằm trong file esp32-mqtt.h).

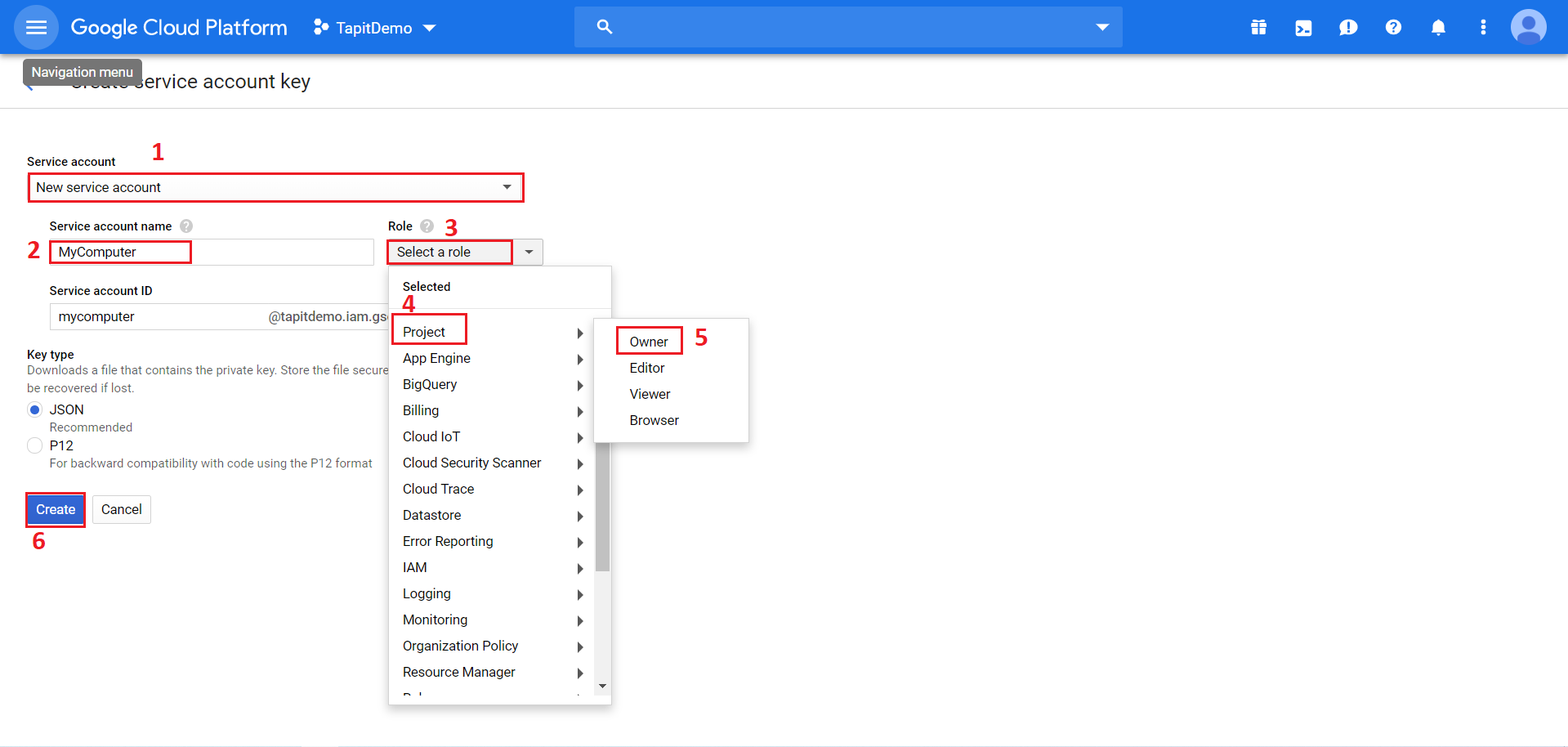


* Trên đây mình đã giới thiệu cho các bạn cách để kết nối ESP32 tới Google IoT Core. Tương tự các bạn có thể mở rộng cho dòng MCU khác cũng nổi tiếng không kém chính là ESP8266. Ở phần tiếp theo, mình sẽ hướng dẫn các bạn cách để lấy dữ liệu thu thập được từ ESP32, ESP8266 hay bất cứ thiết bị IoT nào về local server của bạn mà không cần phải thông qua dịch vụ GCP nào khác. Hẹn các bạn ở bài viết sau!

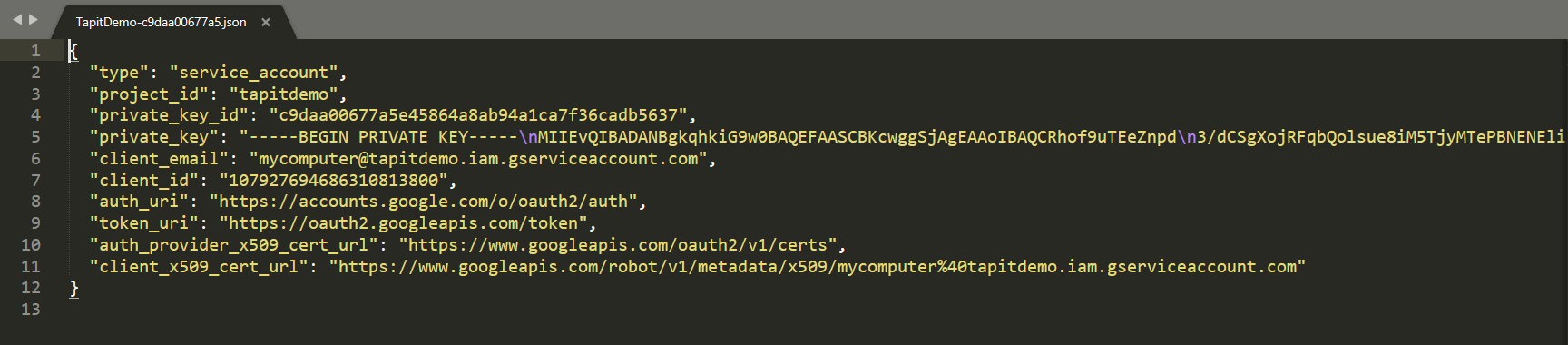
**Phần 6: Quan sát messages từ thiết bị IoT gửi lên Cloud thông qua Python**

**I/ Thiết lập chứng chỉ cho ứng dụng Python**

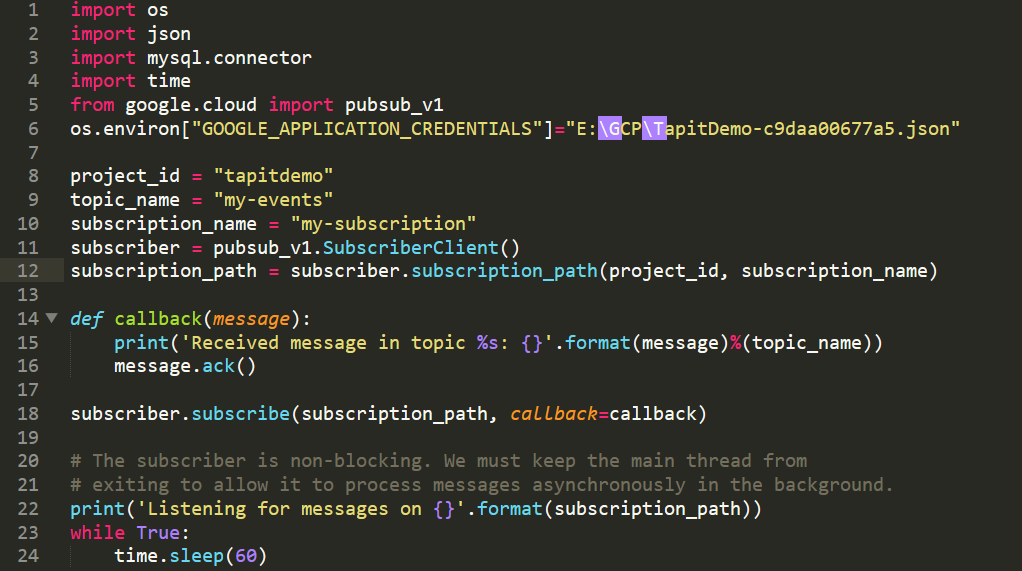
* Cloud Pub/Sub là 1 dịch vụ con nằm trong GCP. Mục đích của nó là dùng để thu thập dữ liệu, còn việc phân tích và xử lý thì dữ liệu này sẽ chuyển sang một dịch vụ khác để thực hiện. Như mình đã trình bày ở những bài viết trước thì các dữ liệu từ thiết bị IoT gửi lên Cloud sẽ thông qua Google IoT Core mà chuyển tới Cloud Pub/Sub, và nhiệm vụ của chúng ta chính là đọc dữ liệu từ dịch vụ Cloud Pub/Sub chứ không phải Google IoT Core.
* Ở phần trước mình có trình bày cách lấy dữ liệu từ Cloud Pub/Sub thông qua gcloud. Tuy nhiên nếu chúng ta xây dựng 1 ứng dụng cụ thể thì việc sử dụng command-line thực sự không hiệu quả lắm. Vậy nên phần này chúng ta sẽ tìm hiểu xem cách lấy dữ liệu từ Cloud Pub/Sub trong ngôn ngữ lập trình python để có thể dễ dàng tích hợp vào ứng dụng của mình.
* Để sử dụng bất kỳ phần mềm thứ 3 nào muốn lấy tài nguyên từ GCP, trước tiên chúng ta phải tạo và cung cấp cho phần mềm đó một chứng chỉ xác thực từ GCP.
* Đầu tiên các bạn vào giao diện Console của GCP, ở thanh điều hướng bên trái, các bạn chọn dịch vụ **APIs & Services** -> **Credentials**.
* 
* Tiếp theo các bạn chọn **Create credentials** -> **Service account key**
* 
* Cuối cùng các bạn chọn như hình bên dưới, mục **service account name** các bạn có thể đặt tên bất kỳ. Sau khi nhấn **Create** thì tự động một file json sẽ được tải về máy bạn.



* Đây là ví dụ về nội dung bên trong file json của mình sau khi tải về.



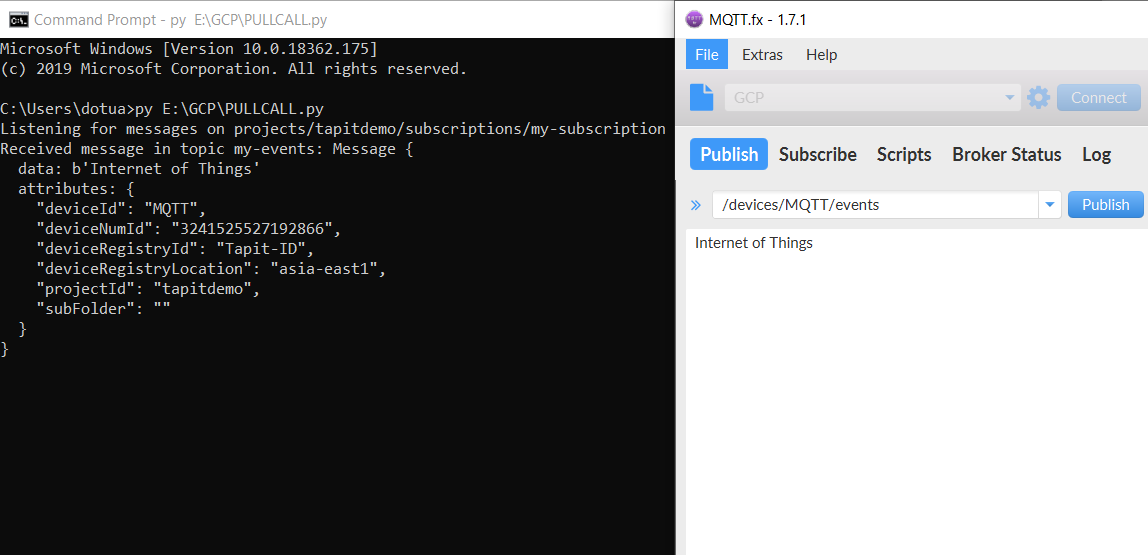
* **II/ Cung cấp chứng chỉ và mô phỏng ứng dụng Python**
* Bây giờ chúng ta sẽ viết chương trình tự động thực hiện lệnh **Pull Call** (kéo dữ liệu về máy) mỗi khi có một gói tin gửi đến Cloud Pub/Sub và tự động gửi lại tin nhắn ACK báo cho Cloud Pub/Sub đã nhận được message.
* Source code chương trình:



* Link tải: <https://1drv.ms/f/s!ArTWbJWNaWLJyD7vvl5_VSQfux2l>

\* Để không gây ra lỗi cho đoạn code trên, các bạn lưu ý download và import các thư viện mà mình đã khai báo ở trên, sau đó thay đổi các thông số *project\_id*, *topic\_name*, *subscription\_name* sao cho phù hợp. Lưu ý chỉ định đường dẫn *os.environ[“GOOGLE\_APPLICATION\_CREDENTIALS”]* tới file json bạn lưu trong máy.

* Để chạy source code trên, chúng ta mở **CMD** của **Windows** lên, gõ “py + [Đường dẫn tới file python]”.
* Ở đây chúng ta Publish gói tin “Internet of Things” từ MQTT.fx lên Cloud, và ứng dụng Python của chúng ta nhận thấy đang có một gói tin gửi tới Cloud Pub/Sub nên nó sẽ tự động thực hiện lệnh Pull Call để lấy dữ liệu về. Dữ liệu nhận được sẽ có định dạng như hình sau:



**Phần 7: Gửi dữ liệu cho thiết bị IoT thông qua HTTP Cloud Function**

**I. Cloud Function là gì?**

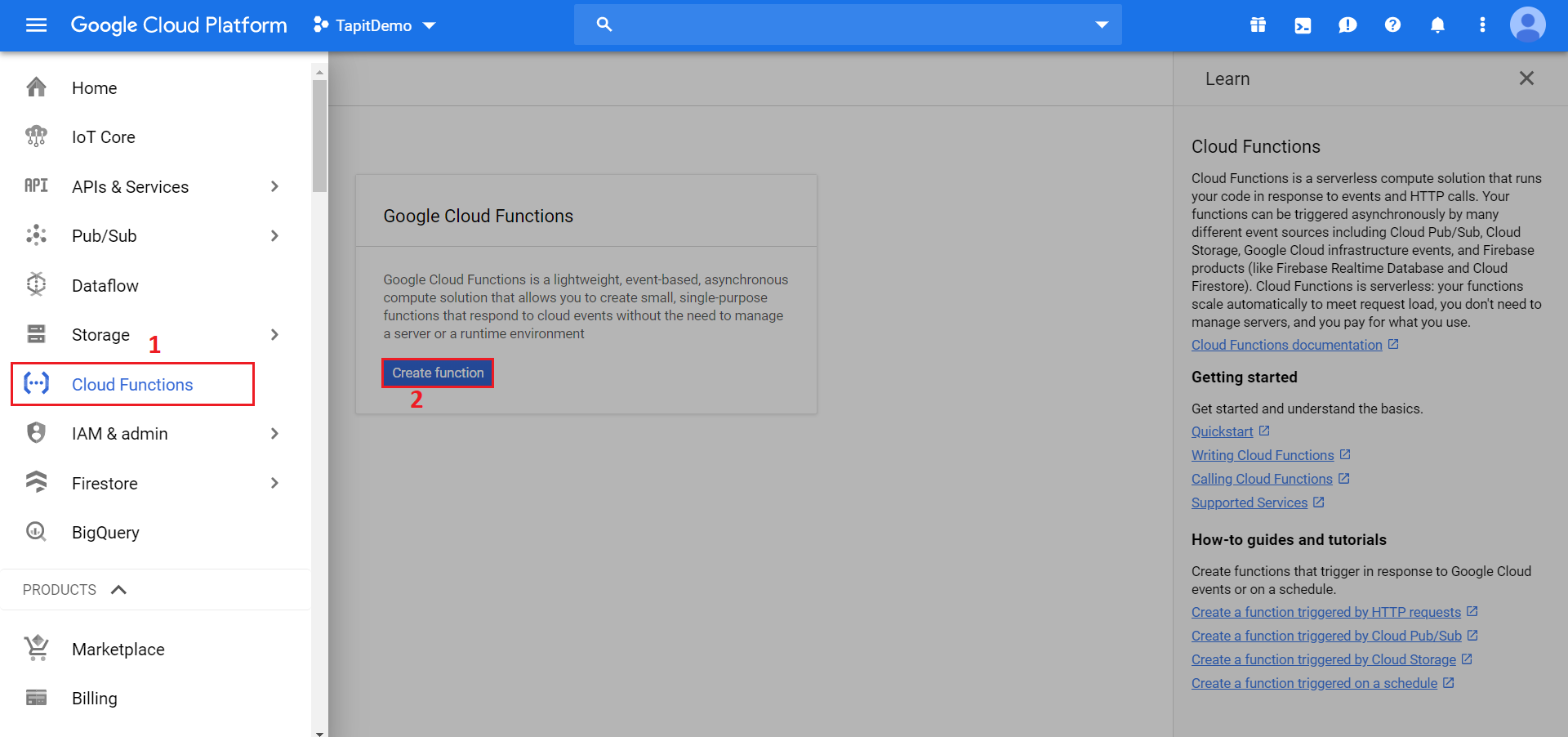
* Cloud function là nơi chúng ta viết chương trình và quản lý code. Mục đích của dịch vụ này là dùng để phản ứng với các sự kiện. Khi có sự kiện, Cloud Function sẽ được kích hoạt để thực hiện các hàm bên trong chương trình của chúng ta.
* Dựa vào các sự kiện kích hoạt, Cloud function chia ra 2 loại:

+ ***Background***: Các sự kiện thay đổi bên trong GCP (Ví dụ có dữ liệu đẩy vào Cloud Pub/Sub hoặc một file ảnh vừa được upload lên dịch vụ BigQuery,..).

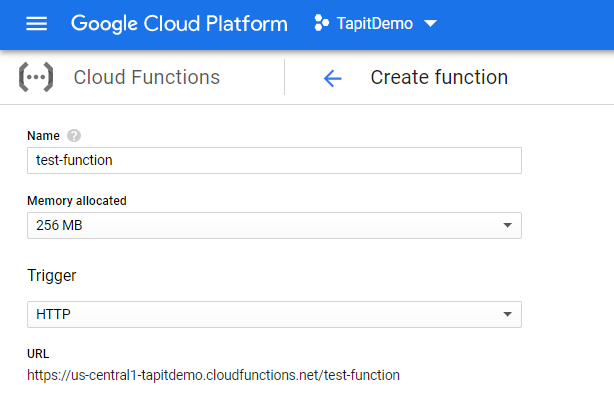
+ ***HTTP***: Một HTTP request gửi lên server.

**II. Sử dụng HTTP Cloud Function để gửi dữ liệu cho thiết bị IoT**

* Google bắt buộc mỗi function chúng ta tạo ra phải có chứng chỉ xác thực do google cung cấp, việc tạo chứng chỉ xác thực mình đã trình bày ở phần trước. Các bạn có thể xài chung một chứng chỉ cho tất cả các function của bạn.
* Tại giao diện console, chúng ta chuyển hướng tới dịch vụ **Cloud Functions** -> **Create Function**.



* Chúng ta có thể đặt tên tùy ý ở mục Name, ở đây mình đặt là *test-function*. Mục **memory allocated** các bạn có thể chọn dung lượng tùy ý, ở đây mình chọn 256MB. Vì chúng ta mong muốn function của mình sẽ được kích hoạt mỗi khi có một HTTP request gửi tới, nên mình sẽ chọn **Trigger** là **HTTP**. Cloud Function sẽ cung cấp cho chúng ta URL để gửi request kích hoạt function đó.



* Các bạn cuộn xuống dưới sẽ thấy các mục cấu hình và cài đặt cho source code của bạn. Ở đây mình có viết sẵn chương trình sẽ được kích hoạt khi có một HTTP request, sau khi nhận được request này thì chương trình sẽ thực thi hàm updateDevice() bên trong chương trình.
* Các bạn có thể tham khảo code tại đây (bao gồm cả file index.js và package.json):

<https://1drv.ms/u/s!ArTWbJWNaWLJyDwm7Bq6yNUp5OYr?e=fCpzdl>



* Tiếp theo các bạn nhấn **Create** và đợi khoảng 5-10 phút để thực hiện việc cài đặt.
* Sau khi source code của bạn được triển khai trên Cloud Function. Thông qua HTTP phương thức GET, các bạn có thể gửi tin nhắn và cấu hình xuống thiết bị IoT tương ứng thông qua URL do google cung cấp với các tham số đại diện là *message*, *deviceID* và *which*.

<https://asia-east2-dotuanson.cloudfunctions.net/test-function?message=aaa&deviceID=ESP32&which=config>

+ message: Tin nhắn gửi tới thiết bị

+ deviceID: Device ID của thiết bị, xác định tin nhắn gửi tới thiết bị mong muốn.

+ which: Có 2 loại dữ liệu là *config* và *commands*, xác định định dạng ta muốn gửi dữ liệu.

\* Các bạn có thể kiểm tra dữ liệu đã được gửi xuống thiết bị chưa bằng cách cho MQTT.fx hoặc ESP32 subscribe vào các topic tương ứng.

- Hẹn gặp lại các bạn ở những bài hướng dẫn tiếp theo của nhóm TAPIT IoTs.

**KẾT THÚC**