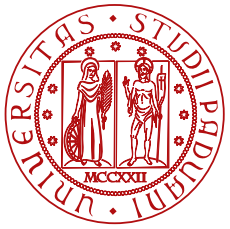

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di laurea in Ingegneria Informatica

Realizzazione di una connessione sicura tra servizi cloud e sensori IoT per il monitoraggio della qualità delle acque costiere e fluviali

Laureando

Alessandro Canova

Matricola 1225232

Relatore

Prof. Filippo Campagnaro

Università di Padova

Anno Accademico
2022/2023

Alla mia famiglia
e ai miei amici

Sommario

L'Internet of Things (IoT), ovvero l'internet delle cose, è un acronimo che fa riferimento all'estensione e all'applicazione di Internet al mondo degli oggetti, che vengono così abilitati a comunicare con altri oggetti connessi in rete, con l'obiettivo di fornire servizi aggiuntivi agli utenti. L'utilizzo dell'IoT può essere esteso a moltissimi campi come l'agricoltura 4.0 o le Smart Cities, ma una delle sue applicazioni più importanti riguarda il monitoraggio e il contrasto dei cambiamenti climatici. Negli ultimi anni si sente sempre più spesso parlare di essi e di come stiano impattando sulla nostra vita. Ogni anno la temperatura media della Terra aumenta, causando gravi danni ambientali e, in alcuni casi, irreversibili.

La città di Venezia e tutto il suo comune, in quanto area costiera, è particolarmente vulnerabile agli impatti del cambiamento climatico. I rischi attribuiti alle inondazioni così come i rischi derivanti dall'occorrenza di interventi estremi sono previsti in aumento, comportando così conseguenze negative sulla popolazione, la sua salute e in generale la sicurezza nel medio-lungo termine della città. È necessario, inoltre, capire come i cambiamenti climatici impattino sulla laguna e sulla qualità delle sue acque [29].

Il progetto su cui si basa questa tesi è stato messo in essere per monitorare la qualità di bacini idrici dolci e salati attraverso l'utilizzo dell'IoT. L'impiego di sensori posti sott'acqua, la raccolta di informazioni e l'elaborazione finale di quest'ultime aiuterà a capire l'evoluzione che il nostro territorio sta subendo e a studiare soluzioni per gestire questi cambiamenti.

In particolare, si andrà ad esplorare il mondo dei sensori Long Range Wide Area Network (LoRaWAN) e a gestire ed implementare la connessione di due servizi molto diffusi nel mondo IoT, ovvero "Amazon Web Services (AWS) IoT Core" e "The Things Network (TTN)". Si analizzerà come implementare una connessione sicura tra i due servizi in maniera tale da poter salvare i dati provenienti dai sensori all'interno di un database per la loro eventuale elaborazione. Si studieranno nel dettaglio i due servizi, i due protocolli principali che verranno utilizzati e tutti i procedimenti necessari per implementare la connessione.

In conclusione, si analizzeranno i costi che comporta la soluzione implementata a confronto con una soluzione alternativa, e verrà presentata un'analisi sulla sicurezza che le due soluzioni offrono.

Indice

Elenco delle figure

Elenco delle tabelle

Acronimi

AES-128 Advanced Encryption Standard - 128. 37, 38

API Application Programming Interface. 4, 16–18, 33, 34, 38

AppKey Application Key. 37, 38

AppSKey Application Session Key. 37, 38

ARN Amazon Resource Names. 20

AWS Amazon Web Services. 2–5, 8, 9, 14, 16–18, 20–25, 27–30, 32–34, 36–39

CSS Chirp Spread Spectrum. 11, 13

GUI Graphical User Interface. 14

HTTPS Hypertext Transfer Protocol over Secure Socket Layer. 14

IAM Identity Access Manager. 4, 18, 20, 27, 37

IBM International Business Machines Corporation. 10

IoT Internet of Things. 2–5, 8, 9, 13, 14, 16, 17, 20–25, 27, 28, 30, 33, 34, 36–39

IP Internet Protocol. 10, 11

JSON JavaScript Object Notation. 23

LoRa Long Range. 11, 13

LoRaWAN Long Range Wide Area Network. 2, 3, 5, 8, 10, 11, 13–15, 22, 30, 32–34, 37, 39

LPWAN Low Power Wide Area Network. 13

MAC Media Access Control. 13

MQTT MQ Telemetry Transport. 3–5, 8, 10–12, 14, 20–24, 26, 30, 31, 33, 34, 39

MQTT-SN MQTT for Sensor Networks. 10, 11

NwkSKey Network Session Key. 37, 38

SQL Structured Query Language. 16, 26

TCP Transmission Control Protocol. 10, 11

TLS Transport Layer Security. 11, 36, 37

TTI The Things Industries. 15

TTN The Things Network. 2, 3, 5, 15–18, 20, 21, 26, 28–30, 33, 34, 37–39

TTS The Things Stack. 15

UDP User Datagram Protocol. 11

URL Uniform Resource Locator. 17

WSS WebSocket Secure. 14

Capitolo 1

Introduzione

Il monitoraggio dei bacini idrici è diventato sempre più importante per capire come l'impatto del cambiamento climatico stia intervenendo su di essi. Il controllo manuale di questi ultimi, però, richiederebbe risorse umane e temporali molto onerose e poco pratiche. Inoltre sarebbe impossibile ottenere delle misurazioni continue nel tempo, non si avrebbe libertà nel decidere le cadenze delle misurazioni e la loro elaborazione sarebbe onerosa. Grazie all'utilizzo dell'Internet of Things (IoT) applicato al mondo della sensoristica, tutti questi problemi possono essere risolti. Utilizzando sensori connessi in rete, in grado di scambiare informazioni tra di loro e con l'esterno, si possono ottenere dati in tempo reale in qualsiasi momento, su richiesta o a cadenze regolari e modificabili. Inoltre, con l'utilizzo di database e di sistemi di instradamento delle informazioni, possiamo salvare dati ma soprattutto elaborarli e analizzarli con algoritmi di Machine Learning per studiare e prevedere i fenomeni atmosferici in maniera efficiente e accurata.

Attraverso l'utilizzo del servizio "The Things Network" in questa tesi siamo andati a creare una rete di sensori Long Range Wide Area Network (LoRaWAN) in grado di leggere valori di varie grandezze fisiche utili allo studio e di comunicarli in rete. Utilizzando poi "Amazon Web Services (AWS) Internet of Things (IoT) Core" siamo andati a leggere questi valori comunicando con il precedente servizio grazie al protocollo MQ Telemetry Transport (MQTT) e a salvarli all'interno di un database. Durante lo studio vedremo vari protocolli e servizi che utilizzeremo per il nostro caso di studio.

Vedremo anche un'analisi dettagliata dei costi che questa soluzione comporterebbe, prendendo in considerazione vari numeri di sensori, e quanto sia conveniente o meno rispetto ad altre soluzioni alternative. Andremo anche a considerare la componente di sicurezza e ad analizzare pro e contro delle varie opzioni anche sotto questo punto di vista.

La tesi è strutturata come segue: il Capitolo 2 introduce i due protocolli di comunicazione utilizzati in questa tesi: MQTT e Long Range Wide Area Network (LoRaWAN), analizzandone caratteristiche e funzionamento. Inoltre,

vengono introdotti i due servizi presi in esame in questa tesi ossia “Amazon Web Services (AWS) IoT Core” e “The Things Network”.

Il Capitolo 3 illustra passo passo come i due servizi sopracitati possano interagire e come si realizza nella pratica una connessione sicura tra essi.

I Capitoli 4 e 5 si occupano di analizzare le differenze tra i due servizi rispettivamente dal punto di vista economico e della sicurezza.

Infine, il Capitolo 6 conclude la tesi.

Capitolo 2

Componenti base del sistema

2.1 Il protocollo MQTT

MQTT [9] è un protocollo di messaggistica leggero, machine-to-machine di tipo publish-subscribe. È stato progettato per l'utilizzo su dispositivi di ridotte dimensioni e con poca potenza ed è in grado di comunicare in reti con basse velocità. È stato inventato nel 1999 da Andy Stanford-Clark, un dipendente dell'International Business Machines Corporation (IBM). L'obiettivo era quello di creare un protocollo che consumasse poca energia, che fosse leggero e utilizzasse poca banda, in quanto doveva essere utilizzato per le trasmissioni satellitari, all'epoca molto costose.

La sigla "MQ" presente nel nome originariamente sta per "Message Queuing", anche se in realtà il protocollo funziona tramite tecnologia "publish-and-subscribe". Da ciò è nata una lunga controversia durante le varie versioni del protocollo, che ha portato nel 2013 a definire che la sigla "MQTT" non ha alcun significato, ma è semplicemente il nome del protocollo [4].

Ne esiste anche una versione più leggera, chiamata MQTT for Sensor Networks (MQTT-SN), rivolta a dispositivi embedded alimentati a batteria su reti non Transmission Control Protocol (TCP)/Internet Protocol (IP), come Zigbee e LoRaWAN.

2.1.1 Funzionamento

MQTT definisce due tipi di entità: un broker e un numero di client.

Un MQTT broker è un server che riceve tutti i messaggi dai vari client e li instrada verso i destinatari.

Un MQTT client è un dispositivo di qualsiasi tipo (da un microcontrollore a un server) che si connette ad un MQTT Broker.

Le informazioni sono organizzate in una gerarchia di argomenti, detti anche topic. Quando un sensore o un device dedito a fornire informazioni ha un messaggio da inviare, invia il dato al broker. Quest'ultimo lo inoltra a tutti i client che si sono iscritti a quello specifico argomento. Se il broker riceve un messaggio di un argomento alla quale nessun client è iscritto lo scarta, a meno che non sia specificato dal mittente che si tratta di un messaggio detto "retained message". Un retained message è un normale messaggio MQTT con il flag "retained" settato con il valore 1. Il broker in questo caso salva sempre un solo "retained message", l'ultimo, per ogni topic e questo verrà inviato a ogni client appena si iscriverà al topic corrispondente.

MQTT lavora appoggiandosi sul protocollo di trasporto TCP, mentre la sua variante MQTT-SN utilizza il protocollo User Datagram Protocol (UDP). Per quanto riguarda la sicurezza del protocollo, non è previsto alcun sistema di criptazione dei dati, se non utilizzando il protocollo Transport Layer Security (TLS) per criptare e proteggere i dati durante il trasporto.

MQTT di default lavora sulla porta non criptata 1883, mentre se si utilizza TLS la porta è la 8883 [28].

2.1.2 Tipi di messaggi MQTT

MQTT utilizza i seguenti messaggi.

- **Connect**: aspetta che la connessione con il broker venga stabilita e crea un link tra i due nodi.
- **Disconnect**: aspetta che il client finisca tutte le operazioni che sta effettuando e disconnette la sessione TCP/IP.
- **Publish**: pubblicazione del messaggio da parte del client verso il broker o da parte del broker verso il client.
- **Subscribe**: richiesta da parte del client di un messaggio appartenente a uno specifico topic.

Sono presenti, inoltre, altri tipi di messaggi, in tutto quattordici, riguardanti messaggi di acknowledgment per ogni operazione disponibile. Un esempio di utilizzo dei messaggi e di comunicazione MQTT è visibile a Figura 2.1 [9].

2.2 Il protocollo LoRaWAN

LoRa

Long Range (LoRa) è una tecnologia di modulazione di frequenza a lunga gittata e a basso impatto energetico che usa Chirp Spread Spectrum (CSS) [14], una tecnica a spettro espanso che utilizza impulsi chirp modulati in frequenza lineare a banda larga per codificare le informazioni [8].

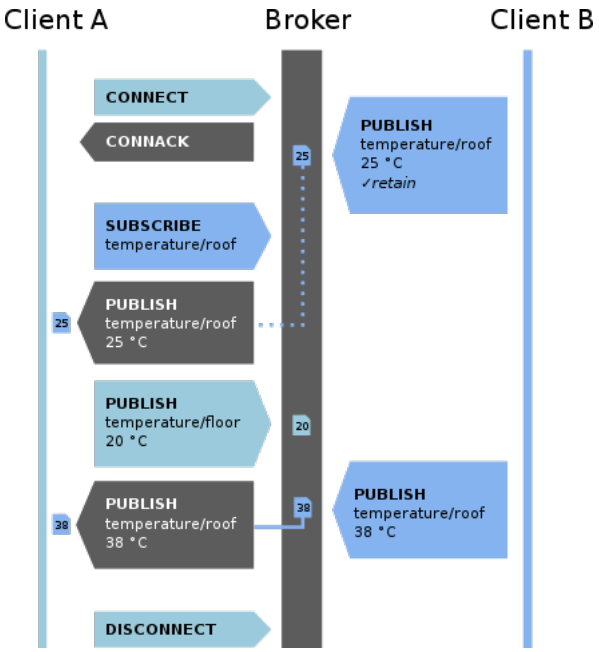

Figura 2.1: Esempio di comunicazione con protocollo MQTT tra due client e il Broker [6].

È stata inventata dalla compagnia Cycleo di Grenoble, in Francia, e ora è proprietà dell'azienda Semtech, unico produttore di moduli LoRa [32]. Funziona tramite l'utilizzo di frequenze sub-gigahertz libere come 433MHz, 868MHz in Europa, 915MHz in Nord America, e altre diverse frequenze in altre parti del mondo. LoRa consente trasmissioni a lungo raggio, raggiungendo anche oltre 10 chilometri in situazioni ideali di campo aperto e 4 chilometri in zone urbane. Utilizza tecniche di forward error correction coding per migliorare la resistenza alle interferenze. Può raggiungere velocità di trasmissioni che vanno da 0.3 Kbit/s fino a 27 Kbit/s a seconda delle distanze e del mezzo trasmissivo [1].

Spreading Factors

La tecnologia CSS su cui si basa LoRa utilizza i chirps (detti anche simboli) per trasmettere i dati. Un chirp è un segnale radio nel quale la frequenza può variare linearmente nel tempo, crescendo o decrescendo, [31]. Uno fattore di diffusione (spreading Factor) più basso implica segnali chirp più veloci e quindi una velocità di trasmissione più alta, ma anche una minore portata del segnale. Cambiando il fattore di diffusione e, di conseguenza, la frequenza dei chirps si può guadagnare o perdere in termini di bit rate e gittata del segnale.

La rete utilizza anche fattori di diffusione per controllare la congestione. Essi, infatti, sono ortogonali, quindi i segnali modulati con diversi fattori di diffusione e trasmessi contemporaneamente sullo stesso canale di frequenza non interferiscono tra loro, [11].

LoRaWAN

Definendone solo il livello fisico, LoRa ha bisogno di essere supportata da protocolli di livello superiore che gestiscono le caratteristiche di rete, come LoRaWAN.

LoRaWAN è un protocollo di comunicazione che fornisce funzionalità di livello Media Access Control (MAC) e di livello di rete per la gestione di comunicazioni tra gateway Low Power Wide Area Network (LPWAN) e dispositivi end-node. Si appoggia al livello fisico LoRa ed è responsabile della gestione delle frequenze di comunicazione, della velocità di trasmissione dei dati e dell'alimentazione dei dispositivi. I dispositivi comunicano in maniera asincrona, ovvero non necessitano di stabilire una connessione con i gateway, ma comunicano appena hanno dati da inviare.

Il suo utilizzo in campo IoT è molto diffuso in quanto, grazie all'uso di basse frequenze e della comunicazione asincrona, è in grado di raggiungere elevate distanze di comunicazione con consumi di energia irrisori. Infatti, per trasmettere un messaggio, il protocollo LoRaWAN consuma al massimo 25mW di potenza. I dispositivi LoRaWAN si possono distinguere in tre categorie, dette classi [9]:

- Bidirectional end devices (Classe A): solitamente dispositivi a bassa potenze e basso consumo. Ogni trasmissione in uplink è seguita da due piccole finestre in downlink. Non è possibile iniziare una comunicazione con una finestra di downlink;

- Bidirectional end devices with scheduled receive slots (Classe B): in aggiunta alle finestre in downlink della classe A, possono aprire ulteriori finestre di ricezione;

- Bidirectional end devices with maximum receive slots (Classe C): solitamente dispositivi ad alimentazione continua e con bassa latenza di comunicazione. Caratterizzati da una finestra di ricezione costantemente aperta tranne in fase di trasmissione uplink.

2.3 AWS IoT Core e The Things Network

Il primo scopo di questa tesi è quello di creare una connessione tra due servizi, "AWS IoT Core" e "The Things Network". Prima di vedere nel dettaglio come implementare questo collegamento ed analizzarne le funzionalità, è importante conoscere cosa sono questi due servizi, i loro punti di forza e le loro caratteristiche principali.

2.3.1 AWS IoT Core

Amazon Web Services (AWS) è la piattaforma cloud più completa e utilizzata nel mondo, offre più di 200 servizi completi da data center a livello globale. È una piattaforma adatta a tutti, dalle start-up in crescita alle grandi aziende, e permette di includere servizi e funzionalità specifiche per ogni esigenza [3].

"AWS IoT Core" fa parte della gamma di servizi offerti da AWS. Fornisce i servizi cloud che collegano i dispositivi IoT ad altri dispositivi e ai servizi cloud di AWS. Fornisce, inoltre, software per dispositivi che possono permettere di integrarli ad AWS e renderli accessibili agli altri servizi [2].

"AWS IoT Core" consente di selezionare le tecnologie più appropriate e aggiornate per ogni soluzione, supportando i seguenti protocolli [2]:

- MQTT.
- MQTT su WebSocket Secure (WSS).
- Hypertext Transfer Protocol over Secure Socket Layer (HTTPS).
- Long Range Wide Area Network (LoRaWAN).

La piattaforma fornisce un'interfaccia utente grafica (Graphical User Interface (GUI)) tramite la quale è possibile configurare e gestire oggetti, certificati, regole, processi, policy e altri elementi e soluzioni IoT.

2.3.2 The Things Network

"The Things Network" offre un insieme di strumenti e una rete globale per creare applicazioni IoT con la massima sicurezza e pronta scalabilità [26].

Il servizio The Things Stack è un server di rete LoRaWAN, componente fondamentale per qualsiasi soluzione LoRaWAN. Viene utilizzato da migliaia di

aziende e sviluppatori in tutto il mondo, gestisce in modo sicuro applicazioni, end devices e gateways ed è gestito da The Things Industries. The Things Stack è costituito da un'architettura che garantisce la sicurezza, il routing dei dati e l'ottimizzazione dei consumi energetici degli end devices LoRaWAN. È un server di rete completo e open source progettato per vari scenari di installazione, che supporta tutte le versioni del protocollo LoRaWAN esistenti, le modalità operativa A, B e C e tutti i parametri regionali rilasciati dalla LoRa Alliance. Nel sito e nel forum di questi servizi, e inoltre anche in questa tesi, si utilizzano spesso abbreviazioni per indicare i tre nomi: The Things Network (TTN), The Things Stack (TTS) e The Things Industries (TTI). Vediamo cosa vogliono dire e nel dettaglio di cosa si occupano [27].

The Things Stack (TTS)

TTS è l'abbreviazione di "The Things Stack", ovvero il network di rete per LoRaWAN. Attualmente è alla versione 3 e viene anche informalmente chiamato "V3".

The Things Network (TTN)

TTN è l'abbreviazione di "The Things Network", ovvero un ecosistema globale dedicato all'Internet of Things in grado di creare reti, dispositivi e applicativi per LoRaWAN.

The Things Industries (TTI)

TTI è l'abbreviazione di "The Things Industries", ovvero la compagnia responsabile dello sviluppo di TTS e di scrivere la documentazione. The Things Industries offre, inoltre, soluzioni cloud hosted per reti LoRaWAN private con funzionalità aziendali aggiuntive e supporto premium per clienti aziendali. Queste soluzioni personalizzate non sono presenti nel sito, ma è necessario contattare il team di TTI per conoscerle.

Dunque, in questa tesi quando utilizzerò la sigla TTN sarà per indicare il servizio "The Things Network".

Capitolo 3

Infrastruttura per la raccolta dati

In questa sezione vedremo come si implementa il collegamento tra Amazon AWS IoT Core e "The Things Network", e come è possibile creare un Database nel quale salvare i dati inviati dai sensori. La prima cosa da fare è collegare AWS e TTN in modo che possano scambiarsi i messaggi. Successivamente creeremo la tabella nella quale salvare i messaggi e implementeremo la query Structured Query Language (SQL) in grado di effettuare questa operazione.

3.1 Collegamento tra AWS IoT Core e TTN

Per far in modo che i messaggi provenienti dai sensori vengano scambiati con "AWS IoT Core" dobbiamo creare un collegamento tra i due servizi. Fortunatamente, TTN offre una guida apposita per eseguire questa operazione. Dobbiamo però prima munirci di vari elementi che ci serviranno durante l'operazione.

Tramite l'interfaccia presente in Figura 3.1 possiamo vedere e gestire l'applicazione creata su "The Things Network". In questa pagina ci salviamo il campo "Application-id", che identifica la nostra applicazione. Il secondo elemento di cui abbiamo bisogno è una chiave Application Programming Interface (API) che servirà per autorizzare lo scambio di messaggi tra i due servizi. Per creare questa chiave dobbiamo cliccare su "API Keys" e "Add API Key". In questa sezione ci viene richiesto il nome della chiave e quali autorizzazioni specifiche vogliamo fornire ad essa. Nel nostro caso non possiamo dare tutte le autorizzazioni, ma solamente quelle indicate nella Figura 3.2. Le autorizzazioni riguardano la possibilità di vedere e creare device nell'applicazione, vedere e modificare le chiavi dell'applicazione, modificare impostazioni di base dell'applicazione, e soprattutto la possibilità di creare e visualizzare il flusso di messaggi riguardati l'applicazione.

Una volta creata la nuova chiave ci viene fornito un codice, la API Key, che dobbiamo salvare e mettere da parte per i passaggi successivi.

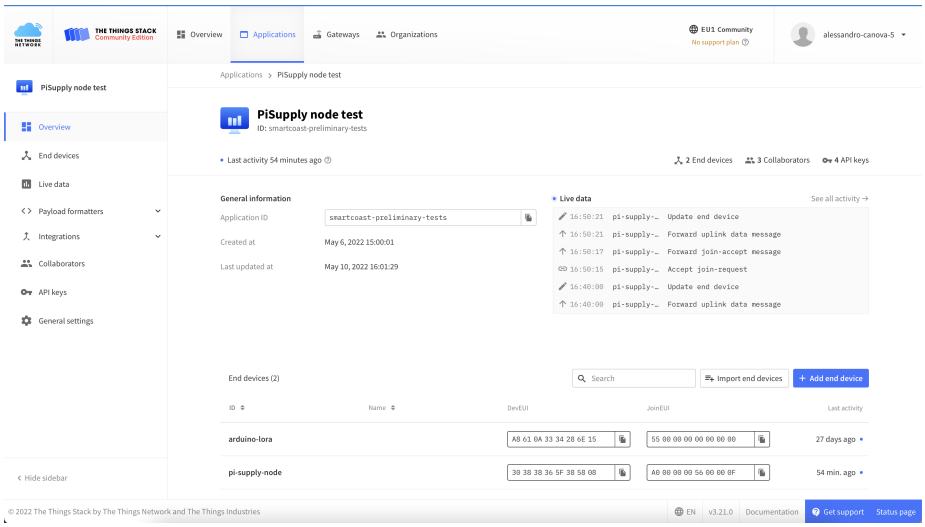

Figura 3.1: Homepage di “The Things Network” [13].

Successivamente, andiamo su “Profilo” e poi su “console”. Selezioniamo “Europe 1” e nella schermata risultante scorriamo fino a fondo pagina. Troveremo varie voci che indicano lo stato dei server di TTN. A noi interessa salvare l’indirizzo riportato sotto i server. Questo Uniform Resource Locator (URL) indica l’indirizzo del cluster dove è salvata la nostra applicazione.

Infine, tornando della pagina indicata in Figura 3.1 e cliccando su “End devices” troveremo l’elenco dei dispositivi collegati alla nostra applicazione di TTN. In questa tesi lavoreremo con un sensore di prova, chiamato “pi-supply-node”. Selezionando questo dispositivo visualizzeremo sulla pagina Web tutti i dettagli a suo riguardo. Di questi campi ci annotiamo solamente il campo “End device ID” che, in questo caso, corrisponde proprio a “pi-supply-node”. Ci servirà per identificare il dispositivo su “AWS IoT Core”.

A questo punto disponiamo di tutti gli elementi necessari per connettere “The Things Network” con “AWS IoT Core”.

Per effettuare questo passaggio ci dirigiamo nel menù a sinistra, clicchiamo su “Integrations” e su “AWS IoT”. Nella pagina che si apre sono presenti due pulsanti, uno dei quali denominato “Deployment Guide”. Cliccandolo veniamo reindirizzati a una pagina facente parte della documentazione di “The Things Network”. Scorrendo in basso, dopo la guida per l’API Key, troviamo la scritta “Deploy AWS CloudFormation Template”. Quello che permette di fare questa sezione è creare un template che serve ad “AWS IoT Core” per predisporre il database al momento della creazione. In questa sezione selezioniamo la nostra zona di competenza, ovvero “Europe, Ireland (eu-west-1)” e clicchiamo sul tab “Community”. È importante selezionare la regione e modalità di funzionamento corretta altrimenti l’integrazione non andrà a buon fine. Fatto questo, cliccando il pulsante “Deploy for The Things Network (eu-west-1)” veniamo reindirizzati

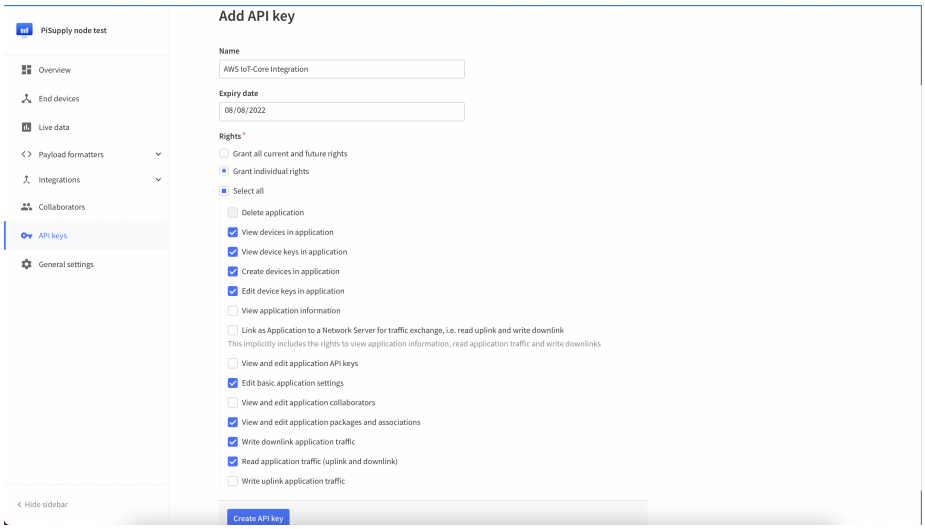

Figura 3.2: Autorizzazioni necessarie per la nuova API Key [13].

in una pagina di AWS CloudFormation dove dobbiamo creare uno stack.

CloudFormation è un servizio che aiuta a modellare e configurare le risorse AWS così da poter dedicare meno tempo alla gestione di tali risorse e concentrarsi invece sulle applicazioni eseguite in AWS [19].

Uno stack è una raccolta di risorse AWS gestibile come un'unità singola. In altre parole, è possibile creare, aggiornare o eliminare una raccolta di risorse mediante la creazione, l'aggiornamento e l'eliminazione di uno stack. Tutte le risorse di uno stack sono definite dal modello AWS CloudFormation dello stack. Ad esempio, uno stack può includere tutte le risorse necessarie per l'esecuzione di un'applicazione Web, ad esempio un server Web, un database e le regole di rete. Se l'applicazione Web non è più necessaria, è possibile eliminare lo stack e conseguentemente verranno eliminate anche tutte le relative risorse correlate [24].

Dunque, nella pagina in cui ci troviamo inseriamo il nome dello stack e vari dati che abbiamo preso precedentemente da TTN, ovvero "Cluster Address", "Application ID", "Application API Key" la chiave creata su "The Things Network", e infine specifichiamo il campo "Thing Type Name" che indica il nome che identificherà di che tipo saranno i dispositivi che si collegheranno ad AWS tramite questo stack. Questa schermata è visibile in Figura 3.3.

Dobbiamo poi spuntare la casella relativa al concedere a CloudFormation l'autorizzazione per creare risorse in AWS Identity Access Manager (IAM).

AWS IAM è un servizio Web che consente di controllare in modo sicuro l'accesso alle risorse AWS. Si utilizza IAM per controllare chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) per l'utilizzo di certe risorse [15].

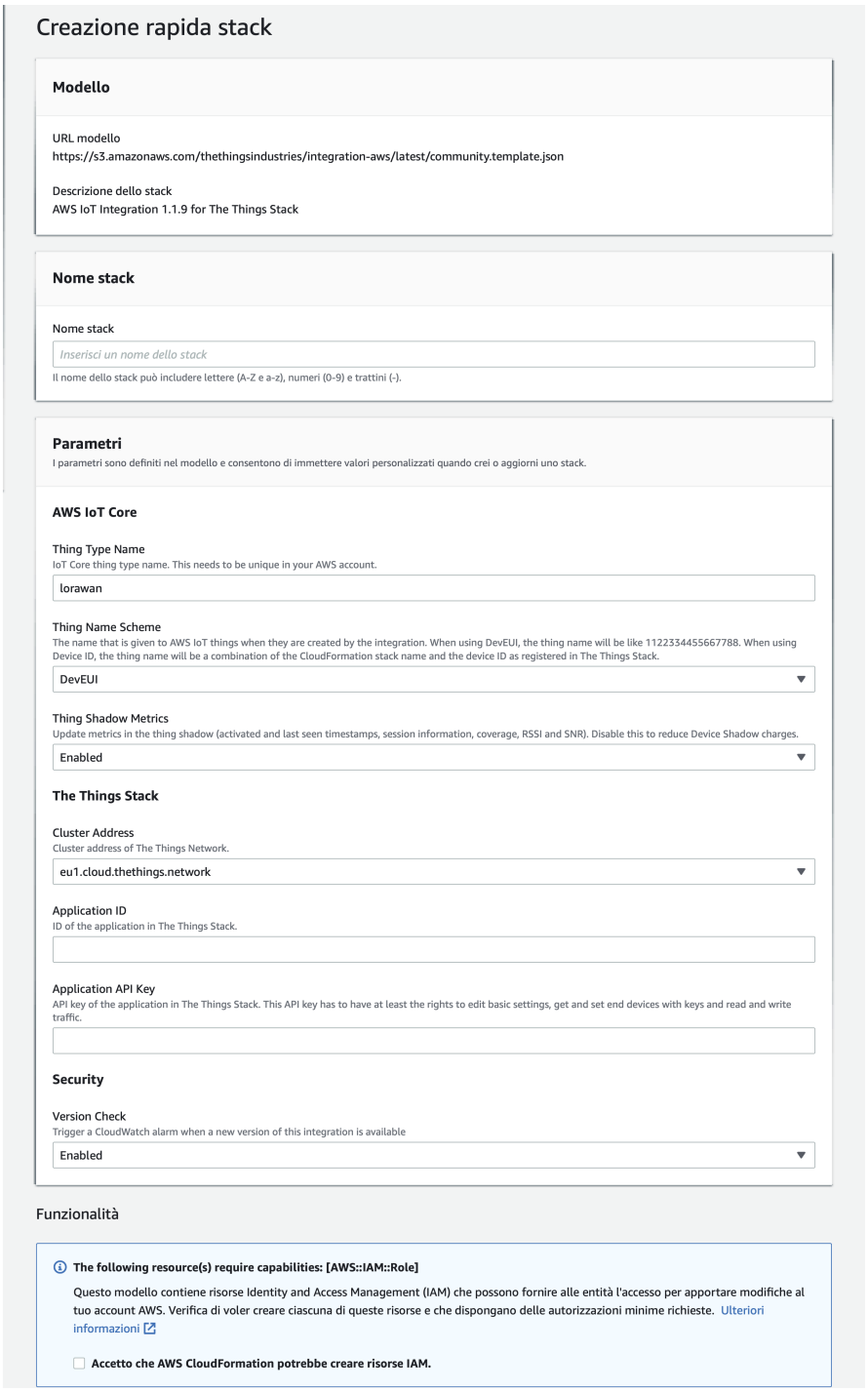

Figura 3.3: Pagina di configurazione di uno stack [16].

CloudFormation necessita di avere l'autorizzazione a modificare le regole in IAM per fare in modo che possa dare certe autorizzazioni ai dispositivi che si collegheranno tramite TTN ad "AWS IoT Core".

Una volta cliccato il pulsante "Crea stack" bisogna attendere qualche minuto perché AWS elabori la creazione. Una volta completata l'operazione possiamo verificare che tutto sia andato a buon fine dirigendoci nella pagina relativa agli stack di CloudFormation e nella riga relativa allo stack appena creato dovremmo trovare la scritta "CREATE_COMPLETE", come possiamo osservare a Figura 3.4.

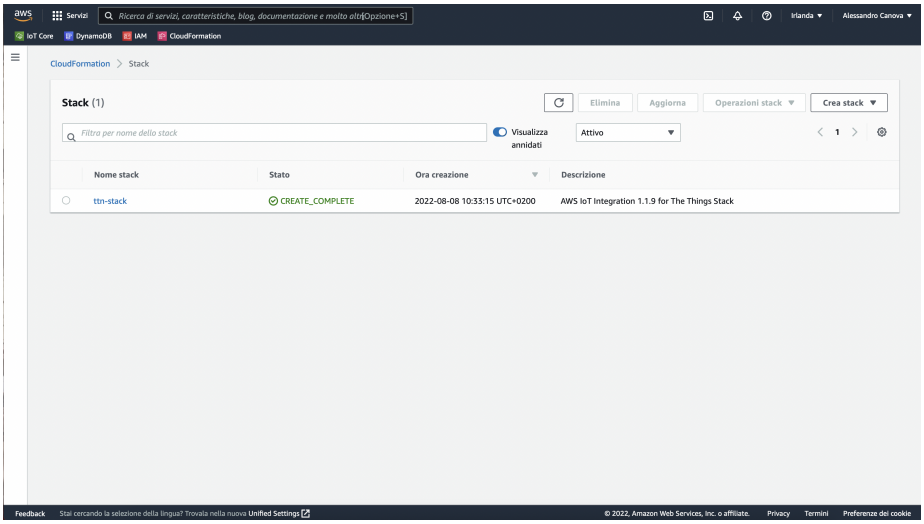

Figura 3.4: Conferma dell'avvenuta creazione dello stack [16].

Per verificare che il collegamento tra i due servizi sia avvenuto con successo, possiamo cliccare sul nome dello stack e aprire la scheda "Output" nella pagina risultante. Qui troveremo un attributo chiamato "CrossAccountRoleArn" che come valore avrà una stringa di caratteri che iniziano con il prefisso Amazon Resource Names (ARN) ("arn:"). Dirigendoci su "The Things Network", nella pagina raffigurata in Figura 3.1, cliccando su "Integrations" e "AWS IoT" troveremo tre campi, rispettivamente "Region", che corrisponderà alla regione selezionata in fase di configurazione del template, "CloudFormation Stack name", corrispondente al nome dello stack che abbiamo assegnato precedentemente, e infine il campo "Role ARN", corrispondente all'attributo "CrossAccountRoleArn" del nostro stack su AWS.

Se tutti i valori corrispondono i due servizi sono collegati tra loro e possono scambiarsi informazioni.

Il prossimo passo è quello di testare il funzionamento del servizio attraverso un client di messaggistica MQTT integrato in AWS e creare un database con una tabella in grado di ospitare i valori provenienti dai messaggi dei dispositivi.

3.2 Messaggistica MQTT e integrazione dei sensori in AWS

Una volta implementata la connessione tra i due servizi siamo in grado di leggere i messaggi che i sensori inviano a TTN anche su "AWS IoT Core". Per fare ciò necessitiamo di uno strumento di IoT Core chiamato "oggetto".

Un oggetto è una rappresentazione di un'entità logica o un dispositivo specifico. Può trattarsi di un dispositivo fisico o un sensore, ad esempio una lampadina o un interruttore su un muro. Può anche trattarsi di un'entità logica, ad esempio un'istanza di un'applicazione, o un'entità fisica che non si connette ad AWS IoT ma che è correlata ad altri dispositivi che si connettono, ad esempio un'auto con sensori per il motore o un pannello di controllo [21].

Ogni oggetto che andremo a creare corrisponderà a un sensore che invia dati a TTN. In realtà, grazie all'integrazione creata con questo servizio, la creazione degli oggetti nel nostro caso avverrà automaticamente ogni volta che un nuovo dispositivo o sensore invierà dei messaggi.

Perciò, presupponendo che il nostro sensore "pi-supply-node" stia già comunicando dei dati, dirigendoci dalla Home di IoT Core su "Tutti i dispositivi" e "Oggetti" troveremo già l'oggetto corrispondente al sensore "pi-supply-node". In fase di configurazione dello stack, visibile a Figura 3.3, abbiamo deciso di salvare ogni oggetto con il nome corrispondente al "DevEUI" del sensore. Avendo fatto ciò, l'oggetto creatosi in IoT Core avrà il nome "303838365F385808", come si vede in Figura 3.5, corrispondente al codice "DevEUI" del sensore "pi-supply-node" in TTN, visibile in Figura 3.1.

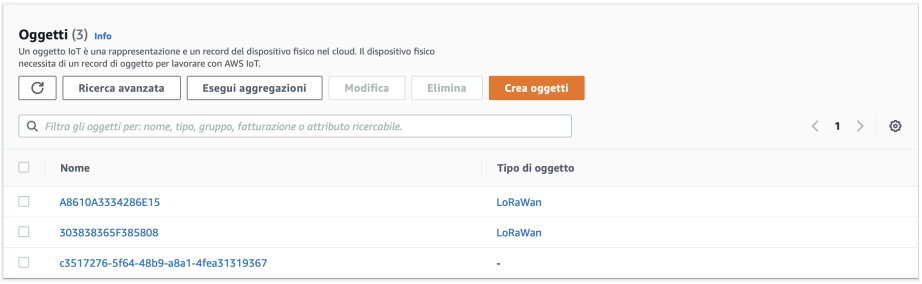

Figura 3.5: Elenco degli oggetti creati in "AWS IoT Core" [16].

Una volta creato l'oggetto siamo ora in grado di leggere i messaggi che il sensore genera tramite AWS IoT Core. Per fare ciò, la piattaforma mette a disposizione un client di messaggistica MQTT, protocollo che abbiamo visto precedentemente.

Cliccando sul nome del sensore da testare si apre una pagina relativa ai dettagli di quest'ultimo. Recandoci nel tab "Attività" possiamo visualizzare un elenco di messaggi più recenti ricevuti dall'oggetto in questione, ed è presente un pulsante "Client di test MQTT", come vediamo in Figura 3.6, che ci permette di

aprire una console per testare se AWS IoT Core riceve effettivamente i messaggi MQTT.

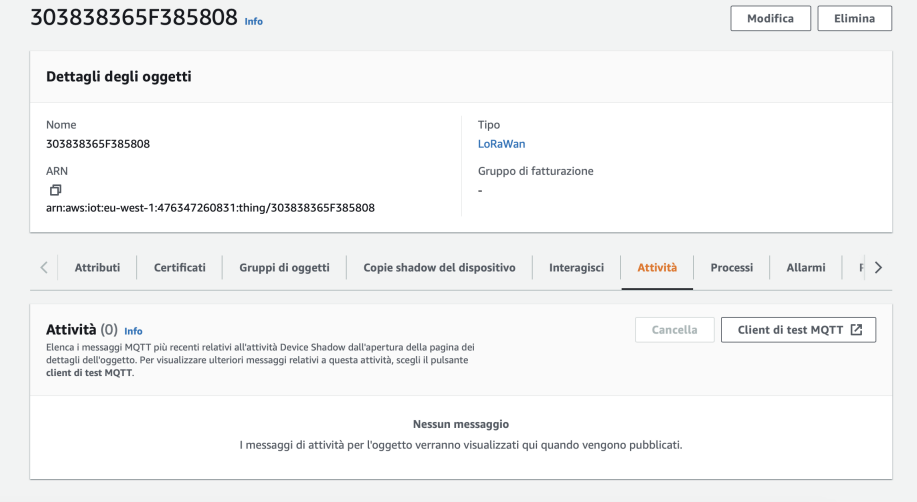

Figura 3.6: Dettaglio attività di un oggetto in "AWS IoT Core" [16].

3.2.1 Console MQTT

La console MQTT permette di testare il corretto funzionamento della messaggistica MQTT di ogni oggetto presente in AWS IoT Core. Come possiamo vedere in Figura 3.7, l'interfaccia è molto semplice e intuitiva, e con il solo specificare di un topic, possiamo andare a visualizzare il messaggio che il sensore in questione ha inviato.

Come visto in precedenza, un topic indica un argomento, ovvero possiamo specificare la famiglia di oggetti dei quali vogliamo testarne il funzionamento, il singolo oggetto oppure una specifica funzionalità di uno specifico oggetto. Per costruire un topic basta seguire la seguente regola:

<nome_tipo_oggetto>/<nome_oggetto>/<tipo_messaggio>

Per visualizzare invece delle macro categorie, basta indicare la categoria desiderata e sostituire con "#" il restante.

Per esempio, se volessimo visualizzare tutti i messaggi che inviano tutti i sensori facenti parte del tipo "LoRaWAN", basterebbe indicare come argomento "lorawan/#" e cliccare su "Sottoscrivi". In questo modo potremo visualizzare tutti i messaggi che vengono inviati da qualsiasi sensore della famiglia "lorawan".

Per specificare invece che vogliamo visualizzare tutti i messaggi di un singolo sensore basta indicare, oltre al tipo, anche il nome del sensore, sottoscrivendo l'argomento "lorawan/303838365F385808/#". In questo modo visualizzeremo tutti i messaggi relativi solo all'oggetto "303838365F385808".

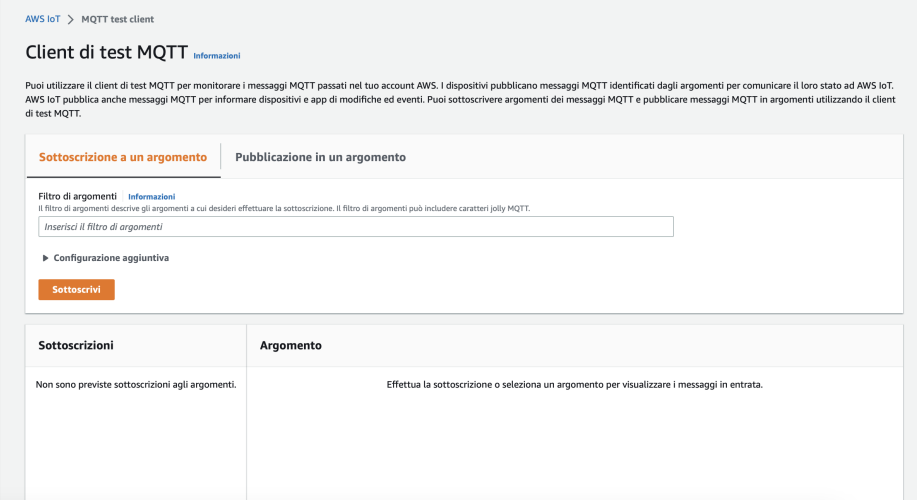

Figura 3.7: Interfaccia del Client MQTT di “AWS IoT Core” [16].

Quindi nel nostro caso sostituiremo il campo “<nome_tipo_oggetto>” con il nome inserito nel campo “Thing Type Name” in fase di creazione dello stack in Figura 3.3 e il campo “<nome_oggetto>” con il nome dell’oggetto desiderato.

I messaggi ai quali siamo interessati per questo progetto sono gli “uplink”, ovvero i messaggi contenenti i dati inviati dai sensori. Dunque, per testare la ricezioni di questi dobbiamo sottoscrivere il topic

lorawan/303838365F385808/uplink

Ovviamente, per avere i dati da sensori diversi basta cambiare il nome dell’oggetto e sottoscrivere un altro topic. Fatto questo, quando il sensore invierà un messaggio potremo visualizzarlo, come vediamo in Figura 3.8.

Il messaggio è in formato JavaScript Object Notation (JSON). Per poter estrarre un’informazione da questo messaggio, quello che ci basterà fare sarà seguire l’incapsulamento dato dalle parentesi graffe e accedere ai vari campi separandoli da un punto “.”. Per esempio, se volessimo ricavare il campo “Application_id” dal messaggio di Figura 3.8 basterebbe scrivere:

end_device_ids.application_ids.application_id

Questo ci tornerà utile in seguito, quando dovremo ricavare i dati dai messaggi per inserirli all’interno del database.

3.3 Creazione del Database e inserimento dei dati

Abbiamo visto come, una volta implementata la connessione tra i due servizi, sia possibile leggere tramite il client di messaggistica MQTT i messaggi inviati

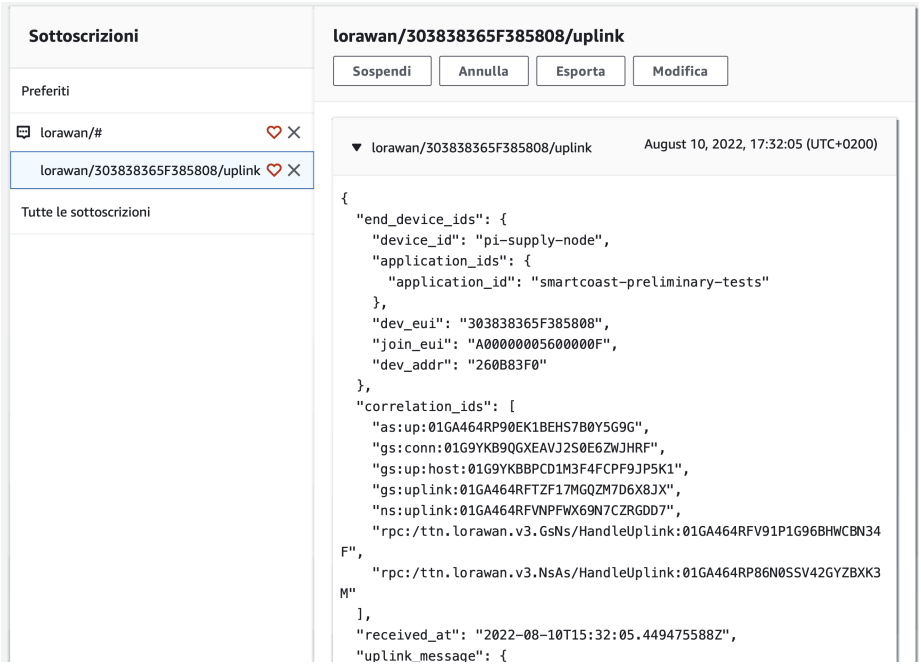

Figura 3.8: Messaggio MQTT proveniente dal sensore pi-supply-node [16].

dai sensori.

A questo punto dobbiamo creare i presupposti perché questi messaggi siano appositamente filtrati per far in modo di estrarne solo le componenti utili, e salvarli in una tabella di un database.

Per effettuare questa operazione entra in gioco un'opzione di "AWS IoT Core" chiamata "Regola".

Le regole [23] indicano ad AWS IoT Core cosa fare quando riceve messaggi dai dispositivi. Esse estraggono i dati dai messaggi, valutano le espressioni utilizzando i dati dei messaggi e richiamano una o più operazioni quando vengono soddisfatte le condizioni della regola. È possibile usare le regole a supporto di attività come:

- aumento o filtro dei dati dei messaggi ricevuti da un dispositivo;
- scrittura dei dati dei messaggi in un database Amazon DynamoDB;
- salvataggio dei dati dei messaggi in un bucket Amazon S3;

Dunque il primo passo è quello di creare un database e successivamente definire una regola che instradi i dati al suo interno.

3.3.1 Creazione del Database

Esistono vari modi per creare un database in AWS. Il più semplice è tramite l'interfaccia grafica messa a disposizione dal servizio integrato chiamato "DynamoDB".

DynamoDB è un servizio di database NoSQL veloce e flessibile, pensato per tutte le applicazioni che richiedono una latenza costante inferiore a una decina di millisecondi su qualsiasi scala. Il suo modello di dati flessibile e le sue prestazioni affidabili rendono DynamoDB perfetto per dispositivi mobili, Web, videogiochi, tecnologie pubblicitarie, Internet of Things e altre applicazioni [17].

Ci dirigiamo, dunque, nella pagina principale del servizio e creiamo una nuova tabella tramite il pulsante "Tabelle" e "crea tabella". Nella nuova schermata ci verranno richiesti tre campi, ovvero il nome della tabella, la chiave di partizione e la chiave di ordinamento.

La chiave di partizione fa parte della chiave primaria e serve per recuperare gli elementi della tabella e per allocare i dati per scalabilità e disponibilità. La chiave di ordinamento, invece, consente di ordinare tutti gli elementi che condividono la stessa chiave di partizione [18].

Nel nostro caso utilizziamo "device_id" come chiave di partizione, e "timestamp" come chiave di ordinamento, in modo che i messaggi salvati verranno suddivisi per sensore e i campi relativi a ogni sensore saranno ordinati per "timestamp", ovvero per orario e data di ricezione.

Una volta cliccato sul tasto "crea tabella" dobbiamo aspettare alcuni secondi che AWS impiega per creare la tabella in modo sicuro e a questo punto potremo visualizzarla nel menù "Tabelle" di DynamoDB, come vediamo in Figura 3.9.

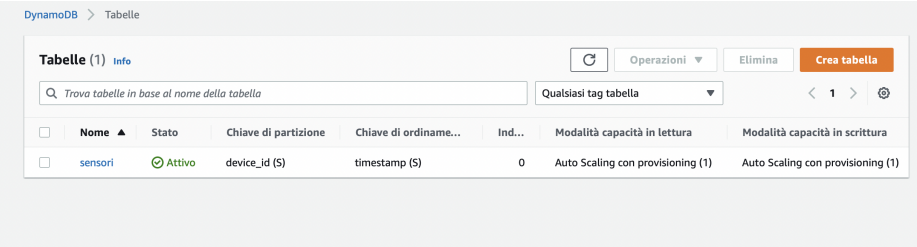

Figura 3.9: Elenco delle tabelle create in DynamoDB [16].

3.3.2 Creazione della regola e instradamento dei messaggi

Una volta creata la tabella nel database non ci resta altro da fare che indirizzare i messaggi che riceviamo dai sensori al suo interno. Le regole possono essere utilizzate a supporto di attività come invio di allarmi, invio di messaggi per IoT Analitycs e instradare messaggi in una o più tabelle DynamoDB.

Dirigendoci, dunque, nella homepage di IoT Core clicchiamo su "Instradamento dei messaggi" e su "Regole". Qui troveremo una lista di regole preesistenti che AWS crea, ad esempio, quando si realizza un nuovo stack o quando

si collega un servizio esterno, come abbiamo fatto noi con TTN. Sono esattamente queste regole che permettono di leggere i messaggi dei sensori o di registrare nuovi oggetti quando nuovi sensori vengono collegati a TTN.

Nella fase di creazione della nuova regola, dopo aver inserito un nome e una breve descrizione, ci viene richiesto di inserire la query SQL. Una query SQL è un comando che serve ad interrogare un database. In questo caso, invece, utilizzeremo una query per estrarre le informazioni dal messaggio ricevuto dal sensore, utilizzando il metodo citato in precedenza.

Nel nostro caso di studio abbiamo bisogno di estrarre il campo "device_id" per la chiave di partizione della tabella, e il campo "timestamp" per la chiave di ordinamento. Inoltre, abbiamo deciso di salvare tre dati che il sensore "pi-supply-node" ci fornisce, ovvero altitudine, latitudine e longitudine. Dunque, nel campo "SELECT" specifichiamo i dati da selezionare dal messaggio, separati da virgole. Si noti come dopo ogni campo sia presente la parola chiave *"as"*, che in Figura 3.10 viene anche evidenziata, seguita da un termine. Questo comando delle query SQL, chiamato *"rename"*, si utilizza per fornire alla tabella una migliore visualizzazione. Infatti, ogni campo dell'attributo "SELECT" indica una colonna della tabella che prenderà lo stesso nome indicato nella query. Con il comando *"as"* possiamo fornire un nome personalizzato alla colonna, ovviamente attenendoci al contesto. Nel campo "FROM", poi, inseriamo il topic MQTT dal quale vogliamo leggere i dati, ovvero <nome_tipo_oggetto>/<nome_oggetto>/uplink. Possiamo, infine, visualizzare la query completa in Figura 3.10.

Proseguendo, ci viene richiesto di specificare quale azione dovrà compiere la regola una volta attivata. Per l'inserimento dei dati nel database ci sono due

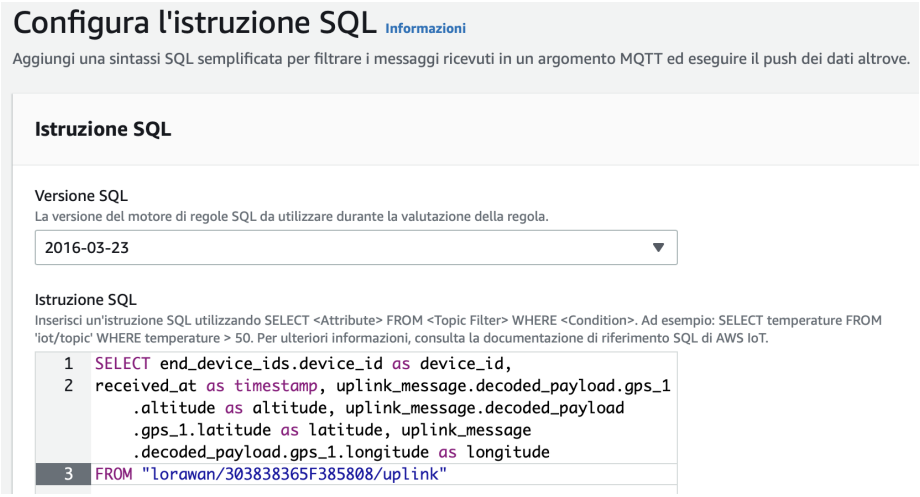

Figura 3.10: Dettaglio della query completa per l'estrazione delle informazioni dal messaggio [16].

opzioni, una chiamata “DynamoDB” e una “DynamoDBv2”.

La prima inserisce un record la prima volta che viene rilevata una nuova chiave non pre-esistente e successivamente, ogni volta che verrà letto un messaggio con quella chiave, aggiornerà il record presente con i nuovi dati. Dunque la nostra applicazione inserirebbe un solo dato per ogni sensore, aggiornandolo ogni qualvolta verrà ricevuto un messaggio dal sensore stesso.

La seconda alternativa, invece, crea un nuovo record per ogni messaggio che riceve da qualsiasi sensore, ed è esattamente ciò che vogliamo per il nostro caso di studi.

Dunque, selezioniamo “DynamoDBv2”, diamo un nome al nuovo ruolo IAM che creiamo al momento, e confermiamo poi la creazione della regola.

Ora, ogni volta che un sensore invierà un messaggio, esso verrà ricevuto in “AWS IoT Core”, ne verranno estratti i dati che verranno poi salvati all’interno del database.

Per visualizzare gli elementi, ci rechiamo in DynamoDB alla voce “Tabelle” e “Esplora elementi”. Nel caso in cui alcuni sensori abbiano inviato dei messaggi, dovremmo poter visualizzare i loro dati in questa schermata. Un esempio è visibile in Figura 3.11.

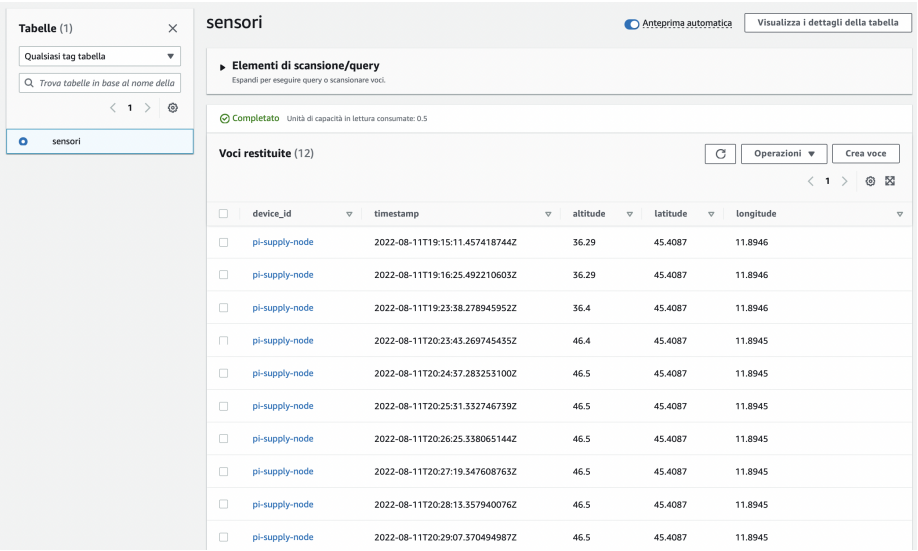

Figura 3.11: Esempio di dati inseriti nella tabella dopo la creazione della regola [16].

Capitolo 4

Analisi dei costi

Nel capitolo precedente abbiamo visto i passaggi per implementare la connessione tra il servizio "AWS IoT Core" e "The Things Network". Questo è stato fatto perchè inizialmente, per questo progetto, era stato previsto l'utilizzo del solo TTN, ma ci si è poi resi conto che era necessaria l'integrazione anche di AWS per lo storage dei dati e per le sue molte altre funzionalità.

È però inoltre possibile bypassare TTN collegando direttamente i sensori ad AWS ed evitando l'uso di entrambi. La domanda alla quale cercheremo di rispondere in questo capitolo è se, dal punto di vista economico, valga la pena mantenere il sistema appena visto oppure cambiare filosofia e approccio, andando dunque a considerare solo "AWS IoT Core". Andremo, quindi, ad analizzare i costi di entrambe le soluzioni.

I due servizi seguono una filosofia diversa per quanto riguarda il costo di messaggistica. La scelta di usare direttamente "AWS IoT Core" oppure appoggiarsi prima a "The Things Network" varia in relazione a quanti dispositivi compongono la nostra rete.

Per questo è stata effettuata un'analisi dettagliata dei costi che comporterebbero le due soluzioni, analizzando vari casi differenti per numero di dispositivi.

4.1 The Things Network

"The Things Network" prevede diversi piani in base alle esigenze dell'utente [25]. Il piano base, chiamato "The Things Stack Discovery", è gratuito ma prevede una limitazione di 10 device, 1 applicazione e 1 gateway collegati e la possibilità di un'interfaccia Cloud. Il secondo piano, chiamato "The Things Stack", invece, è suddiviso in quattro sezioni.

La prima si chiama "Cloud" e permette una gestione e hosting da parte di TTN, un multi-region deployment in caso di necessità legate alla bassa latenza e il suo costo varia in base al numero di device collegati.

La seconda si chiama "AWS Launcher" ed è semplicemente l'integrazione con il servizio di AWS CloudFormation.

Tabella 4.1: Costi dei vari piani di “The Things Network”

N° sensori	1000	1500	2500	3500	4500	5500	10000	12000
Cloud	2280 €	4200 €	5700 €	7200 €	8400 €	9900 €	12600 €	14400 €
AWS	-	-	-	-	-	-	-	-
Enterprise e Docker	5400 €	7128 €	8478 €	9828 €	10908 €	12258 €	14688 €	16308 €
Enterprise e AWS	10800 €	12528 €	13878 €	15228 €	16308 €	17658 €	20088 €	21708 €

La terza opzione, denominata “Enterprise”, è la più completa e permette di ospitare la piattaforma nei propri server, collegamenti con Docker e AWS CloudFormation e include il piano standard di assistenza.

L’ultima opzione chiamata “Dedicated Cloud” non specifica le funzionalità e tanto meno i costi, ma per conoscerli è necessario contattare il team di TTN.

La relazione prezzo-numero di sensori per questi servizi non è specificata e varia in base alla quantità di dispositivi e dall’opzione scelta, dunque nella Tabella 4.1 abbiamo preso in considerazione vari esempi e riportato i vari costi per ogni opzione.

I vari pacchetti, fatta eccezione di quello “Enterprise”, non sono compresi di assistenza, mentre quest’ultimo dispone dell’assistenza Standard inclusa. È possibile, però, acquistare dei pacchetti di assistenza aggiuntivi. Per l’opzione “Cloud” e l’opzione “AWS Launcher” è possibile scegliere un’assistenza Standard che comprende:

- tempo massimo di risposta: 8h;
- portale dedicato al cliente;
- massimo 2 ticket al mese;
- massimo 2 utenti possono aprire dei ticket;
- contatti solo tramite mail o tramite il portale.

In alternativa è disponibile anche l’assistenza Priority che include:

- tempo massimo di risposta: 4h;
- contatto anche tramite via telefonica;
- ticket illimitati;
- massimo 6 utenti possono aprire ticket;
- supporto di debugging.

Il piano Standard costa 2400€/anno, mentre quello Priority costa 7200€/anno. Per il piano “Enterprise” il pacchetto di assistenza Priority costa 3000€/anno. Per il piano “Dedicated Cloud”, invece, non è possibile sapere se l’assistenza è compresa e, in caso contrario, quanto costerebbe.

Osservando i servizi che ci offrono i vari pacchetti, quello più indicato per il nostro caso di studio è il pacchetto “Cloud”, perciò andiamo ad analizzare più approfonditamente il suo costo, senza contare l’assistenza.

Tabella 4.2: Costi del piano "Cloud" di "The Things Network" in base al numero di dispositivi

10-1000	1001-1500	1501-2500	2501-3500	3501-4500	4501-5500	5501-7000
2280 €	4200 €	5700 €	7200 €	8400 €	9900 €	11100 €
7001-8500	8501-10500	10501-13000	13001-15500	15501-18000	18001-20500	20501-25500
12600 €	14400 €	15900 €	17100 €	18300 €	20700 €	22920 €
25501-30500	30501-40500	40501-50500	50501-60500	60501-73000	73001-85500	85501 - inf
24960 €	26760 €	30360 €	33660 €	37560 €	40860 €	44160 €

Come detto in precedenza, "The Things Network" non offre un costo strettamente correlato al numero di sensori ma effettua degli aumenti di prezzo non regolari a cadenze non regolari. Dunque, per capire esattamente quanto potrebbe costare il servizio, riportiamo nella Tabella 4.2 il costo relativo a ogni scaglione, fino al numero limite di 100'000 dispositivi.

4.2 AWS IoT Core

Il servizio "AWS IoT Core" si basa sul numero di dispositivi collegati e sul numero e tipo di azioni che ogni dispositivo esegue. I costi vengono calcolati in base a diversi parametri e in base alla regione di appartenenza. Per il nostro caso di studio abbiamo scelto la regione "eu-west-1", corrispondente alla zona dell'Irlanda, dunque prenderemo in considerazione questa zona. La scelta di quest'ultima è dettata dal fatto che il servizio di messaggistica LoRaWAN è disponibile solamente in questa zona.

Per effettuare un'analisi dei costi che "AWS IoT Core" potrebbe richiedere è presente una pagina apposita del sito [22]. Dirigendoci in questa pagina ci viene ricordato che il servizio prevede un piano gratuito di 12 mesi. Questo, però, contiene alcune limitazioni incompatibili con il nostro progetto. Inoltre, trattandosi di un progetto a lungo termine, non prenderemo in considerazione questo piano.

Nel caso generale, dunque, ogni servizio ha un prezzo specifico e dettagliato. Vanno però distinte due tipologie di connessione: MQTT e LoRaWAN. Esse infatti comportano dei costi diversi anche in relazione al fatto che richiedono servizi diversi per funzionare. Nel caso in cui volessimo collegare i due servizi e utilizzare entrambi, per la parte riguardante AWS andrebbe preso in considerazione il caso di messaggistica MQTT, mentre in caso di collegamento diretto dei sensori ad AWS va presa in considerazione la messaggistica LoRaWAN. Vediamo nel dettaglio entrambi i casi.

4.2.1 Caso MQTT

Per quanto riguarda una connessione con il servizio TTN, i messaggi che riceveremo dal servizio stesso utilizzeranno il protocollo MQTT. Dunque vediamo nel dettaglio i costi. Essi dipendono da vari fattori, e quelli che consideriamo nel nostro caso sono i seguenti:

- tempo di connettività;
- messaggistica MQTT;
- motore delle regole.

Vediamo nel dettaglio i costi di ogni componente per la nostra regione.

Tempo di connettività

Per ogni sensore collegato con protocollo MQTT, il costo è di 0,08 USD per milione di minuti di connessione. Questo vuol dire che mantenere un dispositivo connesso 24 ore al giorno, 7 giorni su 7, comporta un costo di 0,042128 USD all'anno.

Messaggistica MQTT

Per quanto riguarda la messaggistica MQTT i prezzi cambiano in base al numero di messaggi in un anno:

- fino a 1 Miliardo di messaggi: 1,00 USD per milione di messaggi;
- i successivi 4 Miliardi di messaggi: 0,80 USD per milione di messaggi;
- oltre i 5 Miliardi di messaggi: 0,70 USD per milione di messaggi.

La massima dimensione di ogni messaggio è di 128KB e i costi vengono calcolati in incrementi di 5KB. Per esempio se vogliamo trasmettere un pacchetto da 8KB verrà conteggiato come 2 messaggi.

Motore delle regole

Il prezzo delle regole viene calcolato in base al numero di regole attivate e al numero di azioni eseguite da ogni regola. Queste due operazioni hanno il seguente costo:

- regole attivate: 0,15 USD per milione di regole attivate;
- azioni eseguite: 0,15 USD per milione di azioni eseguite.

Dunque per ogni dispositivo, in questo caso abbiamo i seguenti costi:

- connettività: n x 0,043128 USD;
- messaggi: 0,000001 USD x 720 messaggi/mese x 12 mesi x n = n x 0,00864 USD;
- regole: 2 x 0,15 USD/1000000 regole x 12mesi x 720 regole/mese x n = n x 0,002592 USD.

Quindi: (0,043128 x 0,00864 x 0,002592) USD x n = n x 0,05336 USD.

4.2.2 Caso LoRaWAN

Per quanto riguarda il caso in cui volessimo connettere direttamente i dispositivi LoRaWAN ad AWS dobbiamo considerare il caso di messaggistica LoRaWAN. In questo caso i costi da considerare sono solo due:

- messaggistica LoRaWAN;
- motore delle regole.

Non viene contato il tempo di connettività perché la messaggistica LoRaWAN, come visto in precedenza, non prevede una connessione continua al servizio. Vediamo quindi nel dettaglio i costi di ogni componente.

Messaggistica LoRaWAN

Per quanto riguarda la messaggistica LoRaWAN i prezzi cambiano in base al numero di messaggi in un anno:

- fino a 1 Miliardo di messaggi: 2,30 USD per milione di messaggi;
- i successivi 4 Miliardi di messaggi: 1,50 USD per milione di messaggi;
- oltre i 5 Miliardi di messaggi: 1,20 USD per milione di messaggi.

Non sono previsti limiti per le dimensioni dei messaggi, se non quelli imposti dal protocollo.

Motore delle regole

Il prezzo delle regole è lo stesso del precedente caso, dunque:

- regole attivate: 0,15 USD per milione di regole attivate;
- azioni eseguite: 0,15 USD per milione di azioni eseguite.

Dunque, per ogni dispositivo che comunicherà con il servizio AWS si avranno i seguenti costi:

- messaggi: 0,0000023 USD x 720 messaggi/mese x 12 mesi x n = n x 0,019872 USD;
- regole: 2 x 0,15 USD/1000000 regole x 12 mesi x 720 regole/mese = n x 0,002592 USD.

Quindi: (0,019872 x 0,002592) x n = n x 0,022464 USD.

Tabella 4.3: Costi di "AWS IoT Core" con messaggistica MQTT, di "The Things Network" e dei due servizi combinati

No. sensori	AWS IoT Core con MQTT	The Things Network	AWS + TTN
1	$ 0,05	$ 0,00	$ 0,05
10	$ 0,53	$ 0,00	$ 0,53
50	$ 2,67	$ 2280,00	$ 2282,67
100	$ 5,34	$ 4200,00	$ 4205,34
500	$ 26,68	$ 4200,00	$ 4226,68
1000	$ 53,36	$ 5700,00	$ 5753,36
10000	$ 533,60	$ 12600,00	$ 13133,60
30000	$ 1600,80	$ 22920,00	$ 24520,80
40000	$ 2134,40	$ 26760,00	$ 28894,40
50000	$ 2668,00	$ 30360,00	$ 33028,00
70000	$ 3735,20	$ 37560,00	$ 41295,20
90000	$ 4802,40	$ 44160,00	$ 48962,40
91257	$ 4869,47	$ 44160,00	$ 49029,47
100000	$ 5336,00	$ 44160,00	$ 49496,00

4.2.3 Confronto e conclusioni

In questo modo abbiamo ottenuto due coefficienti moltiplicativi che indicativamente ci daranno il costo annuale relativo al numero di dispositivi da utilizzare.

In aggiunta, se consideriamo il caso di TTN insieme a una messaggistica MQTT di AWS, dobbiamo considerare i due costi combinati, in quanto servono entrambi i servizi.

Prendendo un certo numero di sensori come prova possiamo creare le Tabelle 4.3 e 4.4 e ricavarne i grafici di Figura 4.1 e 4.2.

Possiamo vedere dai grafici di Figura 4.1 e 4.2 che l'integrazione diretta dei sensori in "AWS IoT Core" risulta la scelta di gran lunga migliore sotto il punto di vista economico. Anche se la messaggistica LoRaWAN comporta un costo maggiore, il fatto di non dover contare la connettività e, soprattutto, non doversi appoggiare a un servizio molto oneroso come "The Things Network" permette di risparmiare una grande quantità di denaro.

Una possibile ulteriore alternativa potrebbe essere quella di creare diversi account su TTN, ognuno dei quali contenente al massimo 10 sensori, 1 applicazione e 1 gateway, in maniera tale da rientrare nel piano gratuito del servizio, e successivamente collegare ogni singola applicazione ad "AWS IoT Core" e utilizzare la messaggistica MQTT. Questa soluzione, però, oltre ad avere un costo di base più alto in quanto il servizio MQTT, come abbiamo visto, costa di più, comporta una difficoltà di gestione maggiore in quanto necessitiamo di un numero elevato di account e credenziali da gestire. Inoltre, se volessimo collegare un gran numero di applicazioni ad "AWS IoT Core" entrerebbe in gioco un ulteriore costo aggiuntivo. Infatti, per gestire le chiavi API necessarie al collegamento dei due servizi, AWS utilizza un suo applicativo chiamato "AWS Secrets Manager".

Tabella 4.4: Confronto tra il costo del servizio “AWS IoT Core” con messaggistica LoRaWAN e il costo di AWS e TTN combinati con utilizzo di messaggistica MQTT

No. sensori	AWS IoT Core con LoRaWAN	The Things Network	AWS + TTN
1	$ 0,02	$ 0,00	$ 0,05
10	$ 0,22	$ 0,00	$ 0,53
50	$ 1,12	$ 2280,00	$ 2282,67
100	$ 2,25	$ 4200,00	$ 4205,34
500	$ 11,23	$ 4200,00	$ 4226,68
1000	$ 22,46	$ 5700,00	$ 5753,36
10000	$ 224,64	$ 12600,00	$ 13133,60
30000	$ 673,92	$ 22920,00	$ 24520,80
40000	$ 898,56	$ 26760,00	$ 28894,40
50000	$ 1123,20	$ 30360,00	$ 33028,00
70000	$ 1572,48	$ 37560,00	$ 41295,20
90000	$ 2021,76	$ 44160,00	$ 48962,40
91257	$ 2050,00	$ 44160,00	$ 49029,47
100000	$ 2246,40	$ 44160,00	$ 49496,00

AWS Secrets Manager è un servizio che si occupa di archiviare, ruotare, monitorare e controllare l’accesso ai segreti, ad esempio le credenziali del database, le chiavi API e i token OAuh [20].

Durante lo svolgimento del progetto abbiamo visto come sia necessario creare una API Key per collegare TTN con “AWS IoT Core”. Questa chiave viene salvata su AWS utilizzando questo servizio. Il costo di “AWS Secrets Manager” è di 0,40USD/mese (4,80USD/anno) per ogni segreto custodito.

Nel nostro caso di studio, avendo al massimo una sola chiave custodita, il costo è stato trascurato. Invece, nel caso in cui collegassimo un numero elevato di applicativi TTN dovremmo generare una chiave diversa per ogni applicativo, andando ad aumentare ulteriormente il costo di gestione.

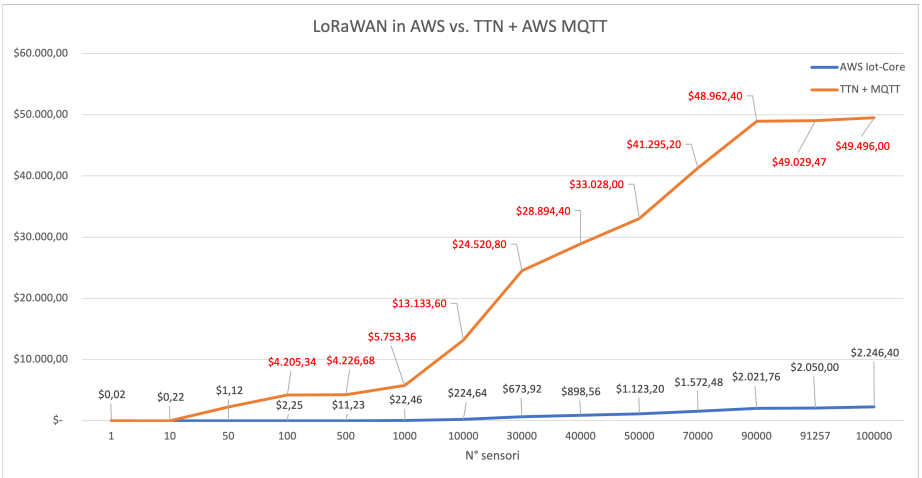

Figura 4.1: Analisi sui costi dei due servizi.

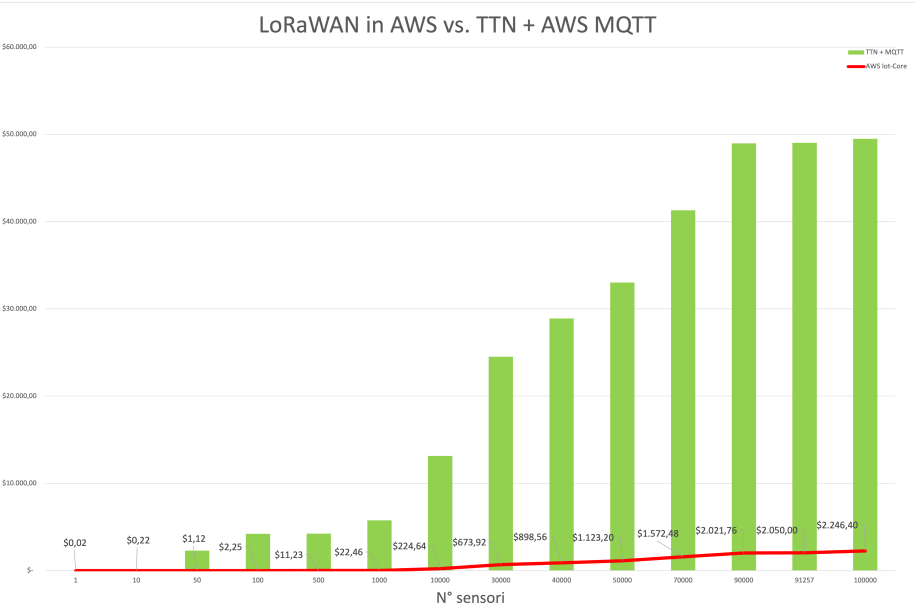

Figura 4.2: Analisi sui costi dei due servizi.

Capitolo 5

Analisi sulla sicurezza

L'Internet of Things (IoT) sta rivoluzionando il modo in cui interagiamo con il mondo fisico, consentendo a dispositivi e sensori di comunicare tra loro e con il cloud per fornire dati in tempo reale e fornire un supporto prezioso all'uomo nel prendere decisioni. Tuttavia, questa connettività continua presenta anche una serie di rischi per la sicurezza come l'accesso non autorizzato ai dati, la compromissione dei dispositivi e la manipolazione delle comunicazioni.

In questo capitolo esploreremo i due servizi utilizzati durante questo progetto, "AWS IoT Core" e "The Things Network", analizzando le misure di sicurezza che offrono per proteggere la comunicazione tra dispositivi e la rete. Esamineremo in dettaglio i protocolli di sicurezza, i meccanismi di autenticazione e autorizzazione, la crittografia dei dati e la gestione delle chiavi. Inoltre verificheremo come viene gestita la rilevazione delle minacce e la conformità.

Confronteremo, infine, i due servizi valutando le loro differenze in termini di sicurezza e di gestione delle risorse. Alla fine di questo capitolo, saremo in grado di valutare quale dei due servizi sia il più adatto in termini di sicurezza al nostro caso di studi.

5.1 Sicurezza in AWS IoT Core

Le strategia utilizzate da AWS per garantire la sicurezza dei dati e degli utenti includono l'autenticazione, l'autorizzazione, la sicurezza fisica, il controllo degli accessi e la crittografia delle comunicazioni. AWS supporta diversi protocolli di sicurezza noti e famosi per la loro robustezza e per il loro vasto utilizzo. Tutto il traffico da e verso i server di AWS IoT Core viene protetto tramite l'uso del protocollo TLS, un protocollo di sicurezza che sfrutta una combinazione di crittografia simmetrica e asimmetrica per garantire la protezione dei dati.

AWS utilizza anche un sistema di autenticazione basato su certificati per verificare l'identità dei dispositivi e delle applicazioni che accedono ai dati. Infatti, ogni dispositivo che si connette ad AWS IoT Core deve possedere un certificato di sicurezza, chiamato certificato X.509 [7], che viene utilizzato per autenticare

Figura 5.1: Gestione della sicurezza da parte di AWS IoT Core tramite l'utilizzo di ruoli e policies con IAM [5].

il dispositivo. Questo certificato viene rilasciato nel momento in cui si registra un nuovo dispositivo e il suo trasporto nella rete viene protetto dal protocollo TLS. Solo quando un dispositivo è in possesso di questo certificato può connettersi e autenticarsi con la piattaforma AWS IoT Core e una volta stabilita la connessione può iniziare a trasmettere e ricevere dati in modo sicuro e protetto. La custodia e la gestione di tale certificato è riservata all'utente [5].

AWS IoT Core integra anche un meccanismo di autorizzazione basato su regole per controllare l'accesso ai dati. Le regole di autorizzazione sono utilizzate per consentire o negare l'accesso a specifici dati in base all'identità del dispositivo o all'autorizzazione dell'utente. Ciò significa che solo gli utenti o i dispositivi autorizzati possono accedere a determinati dati, riducendo così il rischio di accessi non autorizzati. Questo meccanismo, visibile in Figura 5.1 è gestito da IAM, acronimo di "Identity and Access Management", una sezione di AWS dedicata alla gestione della parte di autenticazione e accesso al mondo cloud di AWS.

5.2 Sicurezza in The Things Network

The Things Network è una rete globale di dispositivi interconnessi tramite il protocollo LoRaWAN. L'uso di una rete così vasta può presentare problemi di sicurezza dei dati e degli utenti. Per ovviare a questi problemi, The Things Network utilizza un meccanismo di sicurezza basato sulla crittografia dei dati per proteggerli durante la trasmissione nella rete. TTN si basa sulla sicurezza fornita dai protocolli LoRaWAN v1.0.x e v1.1 [12]. Il meccanismo generale che governa la sicurezza in LoRaWAN prevede l'utilizzo di tre chiavi: Network Session Key (NwkSKey), Application Session Key (AppSKey) e Application Key (AppKey). Tutte e tre hanno una lunghezza di 128 bit e sono generate utilizzando l'algoritmo Advanced Encryption Standard - 128 (AES-128) [10].

L'algoritmo AES-128 è uno tra i più sicuri e diffusi algoritmi di cifratura in circolazione al momento. È basato sulla cifratura a blocchi a chiave simmetrica. È un algoritmo veloce sia se implementato via software che in hardware, è relativamente facile da implementare, richiede poca memoria e offre un ottimo livello di protezione, tanto da essere utilizzato anche dal governo degli Stati Uniti [30]. Quando un dispositivo accede alla rete vengono create una AppSKey e una NwkSKey. La prima rimane privata mentre la seconda viene condivisa in rete.

La NwkSKey è utilizzata per l'interazione tra un nodo e il server per validare l'integrità dei messaggi.

L'AppSKey è utilizzata, invece, per crittografare e decrittografare il payload dei messaggi. Queste due chiavi sono uniche per ogni dispositivo e per ogni sessione. Questo vuol dire che per ogni sessione che si genera con ogni dispositivo, verranno create delle chiavi diverse.

L'AppKey è conosciuta solo dal dispositivo e dall'applicazione in The Things Network. Questa chiave viene utilizzata per generare all'occorrenza le chiavi NwkSKey e AppSKey. [2]

L'uso di queste chiavi garantisce un buon livello di protezione dei dati durante le comunicazioni e un accesso controllato e autenticato alla rete di dispositivi.

5.3 Sicurezza del collegamento tra i due servizi

L'utilizzo di entrambi i servizi prevede l'interazione e lo scambio di dati tra di essi. Questa procedura potrebbe comportare un rischio per la sicurezza di entrambi i sistemi. Per ovviare a questo i due sistemi comunicano solo se la connessione garantisce un meccanismo di autenticazione e autorizzazione. Abbiamo visto in precedenza nel Capitolo 5 come durante la procedura di integrazione del servizio The Things Network in AWS IoT Core sia necessario creare una chiave API che identifica univocamente l'applicazione TTN, e che durante la sua creazione sia necessario specificare quali autorizzazioni si vogliono concedere ad AWS IoT Core. Inoltre, la chiave API Key verrà salvata in AWS utilizzando il servizio "AWS Secrets Manager" che garantisce il salvataggio in sicurezza. In questo modo possiamo assicurare uno scambio sicuro e integro dei dati tra i due servizi.

Capitolo 6

Conclusioni e sviluppi futuri

Lo scopo di questo elaborato era quello di dimostrare come effettuare un collegamento tra i due servizi in modo da riuscire a salvare i dati dei sensori registrati su TTN in un database presente su AWS. Sappiamo però che possiamo anche registrare direttamente i sensori al servizio IoT Core di AWS in modo da bypassare la procedura di collegamento e l'utilizzo di TTN. Effettuando, dunque, un'analisi approfondita del costo dei due servizi combinati a confronto di una soluzione con l'utilizzo del solo AWS si evidenzia come quest'ultima sia di molto più economica in quanto il servizio TTN richiede dei costi molto elevati. Inoltre, il servizio di sola messaggistica LoRaWAN per AWS è più economico del servizio di messaggistica MQTT in quanto non prevede costi per la connessone, non richiesta dal protocollo.

Per quanto visto in termini di sicurezza non c'è grande differenza tra i due servizi. AWS utilizza un sistema diverso di gestione della sicurezza in aggiunge dei livelli di protezione che riguardano principalmente la gestione delle policy e dei ruoli degli utenti. Un'integrazione tra i due servizi permette di sfruttare i benefici di entrambi i servizi, ma considerando che sostanzialmente si equivalgono, la mia opinione è che in termini di sicurezza AWS è preferibile.

Dunque in caso di ampliamento della flotta di sensori, a mio avviso la scelta migliore è quella di utilizzare la sola piattaforma AWS IoT Core insieme a DynamoDB. Bisogna considerare anche che AWS al suo interno offre una vasta gamma di servizi non citati in questa tesi che permettono di effettuare studi approfonditi sui dati, utilizzo di Machine Learning e Deep Learning per algoritmi predittivi, la possibilità di collegare servizi di terze parti per la visualizzazione dei dati e molto altro. Un futuro sviluppo di questo progetto sarà il collegamento dei sensori ad AWS tramite protocollo LoRaWAN e la gestione dei dati esclusivamente attraverso questa piattaforma.

Bibliografia

[1] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the limits of lorawan. *IEEE Communications Magazine*, 55(9):34–40, 2017.

[2] Amazon. AWS IoT Core Developer Guide. "https://docs.aws.amazon.com/pdfs/iot/latest/developerguide/iot-dg.pdf". [Online].

[3] Amazon. Cloud computing con AWS. "https://aws.amazon.com/it/what-is-aws/". [Online].

[4] Amazon. Cos'è MQTT? "https://aws.amazon.com/it/what-is/mqtt/". [Online].

[5] Amazon. AWS IoT Security. "https://docs.aws.amazon.com/iot/latest/developerguide/iot-security.html", 2023. [Online].

[6] Simon A. Eugster. Exchanged packets of an MQTT connection with QoS = 0. "https://en.wikipedia.org/wiki/MQTT". [Online].

[7] Network Working Group. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. "https://www.rfc-editor.org/rfc/rfc5280", May 2008. [Online].

[8] IEEE. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs): Amendment 1: Add Alternate PHYs. *IEEE Std 802.15.4a-2007 (Amendment to IEEE Std 802.15.4-2006)*, pages 1–210, 2007.

[9] Milan Milenkovic. *Internet of Things: Concepts and System Design.* Springer, 8 edition, 2020.

[10] The Things Network. Security. "https://www.thethingsnetwork.org/docs/lorawan/security/". [Online].

[11] The Things Network. Spreading Factors. "https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/". [Online].

[12] The Things Network. The Things Certified Security. "https://www.thethingsnetwork.org/docs/lorawan/the-things-certified-security". [Online].

[13] The Things Network. The Things Network. "https://eu1.cloud.thethings.network/console/". [Online].

[14] Semtech. LoRa Modulation Basics. "https://web.archive.org/web/20190718200516/https://www.semtech.com/uploads/documents/an1200.22.pdf". [Online].

[15] Amazon Web Services. AWS Identity and Access Management Guida per l'utente. "https://docs.aws.amazon.com/it_it/IAM/latest/UserGuide/iam-ug.pdf". [Online].

[16] Amazon Web Services. AWS IoT. "https://eu-west-1.console.aws.amazon.com/iot/home". [Online].

[17] Amazon Web Services. Che cos'è Amazon DynamoDB? "https://docs.aws.amazon.com/it_it/amazondynamodb/latest/developerguide/Introduction.html". [Online].

[18] Amazon Web Services. Componenti principali di Amazon DynamoDB. "https://docs.aws.amazon.com/it_it/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html". [Online].

[19] Amazon Web Services. Cos'è AWS CloudFormation? "https://docs.aws.amazon.com/it_it/AWSCloudFormation/latest/UserGuide/Welcome.html". [Online].

[20] Amazon Web Services. Cos'è AWS Secrets Manager? "https://docs.aws.amazon.com/it_it/secretsmanager/latest/userguide/intro.html". [Online].

[21] Amazon Web Services. Gestione di dispositivi con AWS IoT. "https://docs.aws.amazon.com/it_it/iot/latest/developerguide/iot-thing-management.html". [Online].

[22] Amazon Web Services. Prezzi di AWS IoT Core. "https://aws.amazon.com/it/iot-core/pricing/". [Online].

[23] Amazon Web Services. Regole per AWS IoT. "https://docs.aws.amazon.com/it_it/iot/latest/developerguide/iot-rules.html?icmpid=docs_iot_hp_act". [Online].

[24] Amazon Web Services. Utilizzo degli Stack. "https://docs.aws.amazon.com/it_it/AWSCloudFormation/latest/UserGuide/stacks.html". [Online].

[25] The Things Stack. Plans. " https://www.thethingsindustries.com/stack/plans/". [Online].

[26] The Things Stack. The Things Network. "https://www.thethingsindustries.com/docs/getting-started/ttn". [Online].

[27] The Things Stack. What Is The Things Stack? "https://www.thethingsindustries.com/docs/getting-started/what-is-tts". [Online].

[28] OASIS Standard. MQTT Version 5.0. "https://aws.amazon.com/it/what-is/mqtt", 07 March 2019. [Online].

[29] Fondazione Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC). ANALISI DEL RISCHIO. I cambiamenti climatici in sei città italiane. "https://www.cmcc.it/it/report-venezia". [Online].

[30] Wikipedia. Advanced Encryption Standard. "https://it.wikipedia.org/wiki/Advanced_Encryption_Standard". [Online].

[31] Wikipedia. Chirp. "https://it.wikipedia.org/wiki/Chirp". [Online].

[32] Wikipedia. LoRa. "https://en.wikipedia.org/wiki/LoRa". [Online].

Ringraziamenti

Innanzitutto, un grande ringraziamento va al mio relatore, Professor Campagnaro Filippo, che mi ha guidato con molta pazienza durante questo lungo anno nella realizzazione e stesura di questa tesi.

Ringrazio infinitamente la mia famiglia, che mi ha sempre motivato e sostenuto nei momenti più difficili, e i miei più cari amici che hanno condiviso con me gioie e dolori di questo percorso.

Ringrazio, inoltre, tutte le persone che hanno fatto parte di questo percorso e che mi hanno spinto a raggiungere questo traguardo.