Una panoramica completa dell'IoT su AWS

19 Gennaio 2022 - 8 min. read

Amazon CloudWatch | Amazon DynamoDB | Amazon S3 | AWS CloudFormation | AWS IoT Core | AWS Lambda | Internet of Things (IoT) | Lambda

Negli ultimi anni abbiamo assistito a una crescita costante della popolarità di dispositivi connessi ad internet, portando alla nascita di ecosistemi IoT (Internet of Things). Tra le applicazioni IoT nel settore industriale o retail troviamo la telemetria, il controllo remoto, l'inventario, i sistemi di monitoraggio biomedico, i dispositivi indossabili per il fitness, i gadget connessi per gli assistenti vocali e molto altro. Tutti questi oggetti devono essere accessibili da Internet per permetterci di monitorarne costantemente lo stato, ricevere notifiche o controllarli a distanza.

In questo articolo tratteremo l'argomento in modo piuttosto informativo e discorsivo, partendo dalle aree più generiche fino ai classici problemi che ci si trova ad affrontare sviluppando questo tipo di applicazioni, alla descrizione dei protocolli e, infine, una panoramica dei servizi che AWS offre per la realizzazione di soluzioni IoT.

Nei capitoli seguenti esamineremo la creazione di un ecosistema IoT, i punti chiave da considerare e quali strumenti possono aiutarci.

La larghezza di banda e il consumo di dati

Supponiamo di voler realizzare una produzione di massa di dispositivi connessi. Dobbiamo considerare che gli utenti potrebbero posizionarli ovunque, la quantità di dati trasferiti potrebbe variare e la comunicazione potrebbe dover essere bidirezionale. Potremmo avere dispositivi collocati in una stanza con accesso a una connessione

Internet stabile senza vincoli di larghezza di banda e altri che operano all'aperto collegati tramite una rete mobile. Queste limitazioni ci costringono a scegliere un protocollo leggero ed elastico adatto a varie condizioni di rete e posizionamento dei dispositivi.

Sicurezza

La sicurezza della nostra infrastruttura è da considerarsi cruciale: ogni dato inviato dai nostri dispositivi deve essere crittografato sia in transito che a riposo. Anche se il traffico non contiene informazioni sensibili secondo le leggi applicabili, non vogliamo che persone o sistemi non autorizzati scoprano cosa viene inviato o ottengano l'accesso alle informazioni sull'infrastruttura sottostante. Inoltre, i dati a volte possono essere difficili da classificare e le informazioni sensibili potrebbero non essere evidenti all'inizio, pertanto l'approccio più sicuro consiste nel considerare tutto il traffico e i dati archiviati come delicati, applicando quindi un livello di sicurezza ragionevole a tutti i livelli. Dobbiamo proteggere il tunnel di connessione perché la situazione peggiore che potrebbe verificarsi è l'esposizione di backdoor che potrebbero potenzialmente consentire a chiunque l'accesso ai nostri dispositivi.

La nostra soluzione deve essere scalabile e basata su eventi

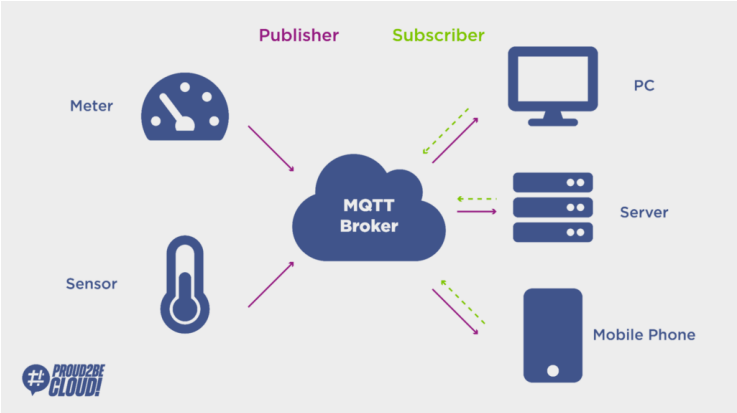
La tipica applicazione IoT consiste in un insieme di azioni che vengono eseguite lato server quando i client inviano messaggi, inclusa l'elaborazione e l'invio di risposte o comandi/notifiche ai dispositivi. La frequenza dei messaggi può variare a seconda dell'applicazione, con picchi di lavoro alternati a periodi di inattività, quindi la nostra architettura deve essere scalabile ed elastica. Nella maggior parte dei casi, il software deve reagire agli eventi. Quando un client notifica un cambio di stato al server, viene creato un nuovo evento e diverse aree dell'applicazione back-end possono agire di conseguenza. Vogliamo inoltre che la nostra architettura si adatti al numero di dispositivi connessi in modo da essere sufficientemente elastica per far fronte a un carico di lavoro crescente.

L'avvento del protocollo MQTT

MQTT sta per Message Queuing Telemetry Transport ed è un protocollo di messaggistica creato per soddisfare gli scopi sopra citati. Di solito si appoggia al protocollo TCP/IP e definisce due tipi di entità di rete:

- Il broker di messaggi, ovvero il server che gestisce i messaggi ricevuti dai client.
- I client, ovvero i dispositivi che inviano informazioni al broker.

Il broker di messaggi offre diversi topic, che consistono in spazi utili per organizzare le informazioni pubblicate dai client, e tiene traccia degli stati di connessione, delle credenziali di sicurezza e dei certificati di tutti i dispositivi. Se la comunicazione deve essere bidirezionale, i client possono anche sottoscriversi a un topic specifico (o più di uno se necessario) per ricevere i messaggi pubblicati dal broker di messaggi.



Esistono tre tipi di messaggi che i client possono inviare al broker di messaggi:

- Connect - crea un tunnel di connessione che il client utilizzerà per inviare le informazioni.
- Disconnect - Chiude il tunnel di connessione.
- Publish - invia un messaggio tramite il tunnel di connessione ad un topic che verrà gestito dal broker di messaggi.

È possibile inoltre configurare la qualità del servizio, che indica i parametri usati per caratterizzare la qualità del servizio offerto dal broker. Questi sono i possibili valori che puoi configurare:

- 0 - al massimo una volta. I messaggi che non possono essere recapitati correttamente andranno persi.
- 1- almeno una volta. Il back-end potrebbe ricevere duplicati di messaggi su un determinato topic.
- 2 - esattamente una volta. Viene garantita la consegna univoca del messaggio.

Poiché la sicurezza è un punto cruciale delle infrastrutture IoT, il protocollo MQTT può proteggere le informazioni trasferite utilizzando TLS e autenticare i client utilizzando

protocolli di autenticazione moderni, come OAuth.

Distribuzione dell'ecosistema IoT nel cloud

Come promesso all'inizio dell'articolo, analizzeremo i servizi di AWS che possono essere utilizzati per sviluppare applicazioni IoT nel cloud. Ma niente paura, seguiranno altri articoli in cui esploreremo nel dettaglio i servizi e i principali pattern architettonici che possiamo utilizzare!

IoT Core

AWS offre un'ampia varietà di servizi che possono aiutarci a costruire una solida soluzione IoT e il più famoso è IoT Core.

IoT Core ti consente di gestire le connessioni MQTT tra i tuoi dispositivi e il cloud in un ambiente completamente serverless. Pagherai solo per le risorse che utilizzerai e si scalerà automaticamente in base alle tue esigenze.

Per creare una nuova connessione è necessario prima configurare un dispositivo su IoT Core. La creazione del dispositivo è semplice. Tutto ciò di cui hai bisogno è un nome per la tua risorsa e configurare i certificati utilizzati per creare una connessione crittografata. Puoi anche organizzare il dispositivo IoT in gruppi distinti, assegnare tag per facilitare il filtraggio e inserirli nel proprio gruppo di fatturazione per allocare i costi.

I certificati sono facili da creare: IoT Core fornirà le chiavi private e pubbliche con pochi click (è vivamente consigliato archiviarle in un bucket S3 crittografato poiché non saranno accessibili dopo il processo di creazione dell'oggetto). Se desideri fornire i tuoi certificati, IoT Core ti consente di caricarli ed associarli al dispositivo.

L'ultima cosa da fare prima di iniziare ad utilizzare il tuo nuovo dispositivo è applicare una policy.

Le policy definiscono cosa può fare il tuo dispositivo. Ad esempio, potresti avere oggetti che possono pubblicare e sottoscriversi ad un sottoinsieme specifico di topic, altri che possono solo ricevere notifiche senza poter inviare messaggi, ecc. Queste

policy di sicurezza possono essere modificate in seguito e la modifica avrà effetto immediato.

Device shadow

IoT Core fornisce inoltre una funzionalità chiamata *shadow*. Lo shadow è un documento JSON che contiene lo stato corrente di uno specifico dispositivo. I tuoi dispositivi possono aggiornarlo semplicemente pubblicando un messaggio sull'endpoint del proprio topic. Un oggetto IoT può avere più shadow con nome e solo una senza nome. Il contenuto degli shadow è accessibile dall'applicazione anche quando il relativo dispositivo è offline e viene inoltrato al dispositivo quando questo si riconnette.

Gli oggetti IoT possono trarre vantaggio dallo shadow poiché è un modo per archiviare lo stato e la configurazione in uno spazio cloud centralizzato accessibile solo da esso e dalla tua applicazione.

IoT Rules

A fronte degli aggiornamenti sui topic, potresti voler scatenare eventi che li gestiscano. Questo può essere fatto con le IoT Rules. Ogni regola ha un nome, un'istruzione simile a SQL che consente di filtrare i messaggi e selezionare un sottoinsieme di campi e un elenco di azioni da intraprendere. È possibile associare diversi servizi AWS alle tue regole IoT; Ecco alcuni esempi:

- Funzioni Lambda per applicare la tua logica di business.
- SNS per inviare i messaggi a più destinatari.
- SQS per disaccoppiare la tua logica.
- S3 e DynamoDB per archiviare le informazioni in una memoria persistente per un utilizzo successivo.
- Kinesis Firehose per elaborare più messaggi in batch.

Quest'ultimo servizio è utile anche quando vuoi ridurre il numero di chiamate ed i costi di S3: è consigliato evitare l'esecuzione di una chiamata PutObject verso S3 per ogni aggiornamento inviato dai tuoi dispositivi perché si rischia di incorrere in un aumento inaspettato dei costi del servizio. Con Kinesis Firehose è possibile archiviare più messaggi in un singolo oggetto S3 con una singola chiamata API.

Monitoraggio dei dispositivi

La console IoT Core offre un modo semplice e rapido per monitorare lo stato generale dei tuoi dispositivi. È possibile consultare direttamente dalla homepage del servizio il numero di dispositivi attualmente connessi, il numero di messaggi pubblicati e le regole eseguite. Inoltre IoT Core può essere configurato per pubblicare i log delle attività e metriche su Amazon CloudWatch.

Un esempio

Supponiamo di voler costruire un ecosistema IoT con diversi sensori che raccolgono informazioni meteorologiche. Ogni sensore è un dispositivo IoT Core con i propri certificati e invia periodicamente un aggiornamento in formato JSON contenente le informazioni di campionamento a uno o più topic MQTT. Ecco cosa vogliamo fare:

1. Automatizzare la creazione di nuovi oggetti IoT con una funzione Lambda che configura il nuovo dispositivo su IoT Core e archivia i certificati in un bucket S3 crittografato.
2. Archiviare tutti i messaggi ricevuti in un bucket su S3. Non sappiamo ancora se avremo bisogno della cronologia dei campionamenti per un dispositivo, quindi è buona norma conservare questi dati in un servizio di archiviazione economico e scalabile come S3.
3. Salvare i messaggi su DynamoDB (o Timestream) per recuperarli rapidamente. Vogliamo mostrare alcuni grafici su un'applicazione Web che risiede nel relativo bucket S3 protetto da Amazon CloudFront e la velocità di recupero dei dati è un punto cruciale.
4. Gestire gli eventi di connessione e disconnessione con le notifiche e-mail. Possiamo farlo facilmente con una IoT Rule che invoca una funzione Lambda a fronte di messaggi automatici pubblicati sui topic di connessione e disconnessione. Potremmo anche pubblicare una metrica personalizzata su CloudWatch per tenerne traccia.

Questa possibile architettura soddisfa tutte le nostre esigenze: è interamente serverless, il che significa che può scalare in base alle nostre esigenze; è conveniente perché paghiamo solo per le risorse che stiamo utilizzando; è elastica poiché può crescere in termini di funzionalità in quanto ogni aspetto della nostra applicazione è disaccoppiato dagli altri; le connessioni tra client e IoT Core sono crittografate, quindi la nostra infrastruttura è sicura.

Conclusioni

Abbiamo esplorato le principali aree di interesse relative al mondo IoT; abbiamo introdotto il protocollo MQTT ed il servizio AWS IoT Core con le sue funzionalità principali utili a costruire applicazioni interamente cloud-native

Restate sintonizzati per la serie di futuri articoli in cui approfondiremo le problematiche legate al mondo IoT e di cui questo è solo il capitolo introduttivo!

Mattia Costamagna

Ingegnere DevOps e sviluppatore cloud-native @ beSharp. Adoro passare il mio tempo libero a leggere romanzi e ascoltare musica rock e blues degli anni '70. Sempre alla ricerca di nuove tecnologie e framework da testare e utilizzare. La birra artigianale è il mio carburante!
