AWS IoT Core Device Management in azione

18 Marzo 2022 - 8 min. read

AWS IoT Core *Internet of Things (IoT)*

Gli ecosistemi IoT sono costituiti da molteplici dispositivi connessi a un back-end centralizzato che gestisce le connessioni e i messaggi in entrata e in uscita. Nella fase di progettazione della nostra infrastruttura dobbiamo considerare come organizzare le nostre risorse in un modo che ci permetta di ricercarle rapidamente. In caso contrario, rischieremmo di operare in un ambiente caotico, difficile da gestire e quasi impossibile da debuggare.

La gestione dei dispositivi consiste nella definizione di metodologie per filtrare rapidamente i nostri oggetti, individuare facilmente quali non stanno operando come dovrebbero e disporre di un'archiviazione organizzata e scalabile per certificati e messaggi.

Questo articolo conclude la trilogia della nostra panoramica IoT: abbiamo già visto cosa si dovrebbe considerare quando si inizia a sviluppare un progetto IoT e come sfruttare la gestione delle regole AWS per attivare altri servizi in un'applicazione basata su eventi. Vediamo ora come organizzare i dispositivi nel miglior modo possibile!

Le questioni principali che dobbiamo considerare

Automazione

Attività come la creazione, l'eliminazione e il rinnovo dei certificati del dispositivo, richiedono, in genere, più azioni. Automatizzare questo tipo di operazioni consente di eliminare la probabilità di errore. Su AWS è possibile gestire queste automazioni tramite delle funzioni Lambda che possono essere attivate come eventi da varie altre parti della nostra applicazione, o, nel caso di automazioni più complesse, attraverso l'utilizzo di Step Function.

Sicurezza

La sicurezza è probabilmente l'aspetto più cruciale delle applicazioni IoT: come discusso in precedenza, è fondamentale evitare che persone e sistemi non autorizzati possano accedere ai nostri dispositivi. I certificati devono essere salvati in uno spazio di archiviazione cifrato con accesso limitato e replicati in più availability zone e, se possibile, in aree geografiche diverse per ridurre la possibilità di perdita di dati. Amazon S3 è il servizio perfetto per questo scopo: offre encryption at rest (gestita con AWS Key Management Service (KMS), replica su un altro bucket in un'altra AWS Region, policy di sicurezza e integrazione IAM per consentire l'accesso solo alle persone e alle applicazioni che si devono interfacciare con esso.

Scalabilità

Qual è il numero atteso di dispositivi connessi al nostro ecosistema? La nostra applicazione è B2B o B2C? Queste sono alcune delle domande che è utile porci per definire una strategia di organizzazione dei dispositivi e delle informazioni che ci aiuti a trovare ciò che stiamo cercando, anche se la quantità di dati filtrabili è enorme.

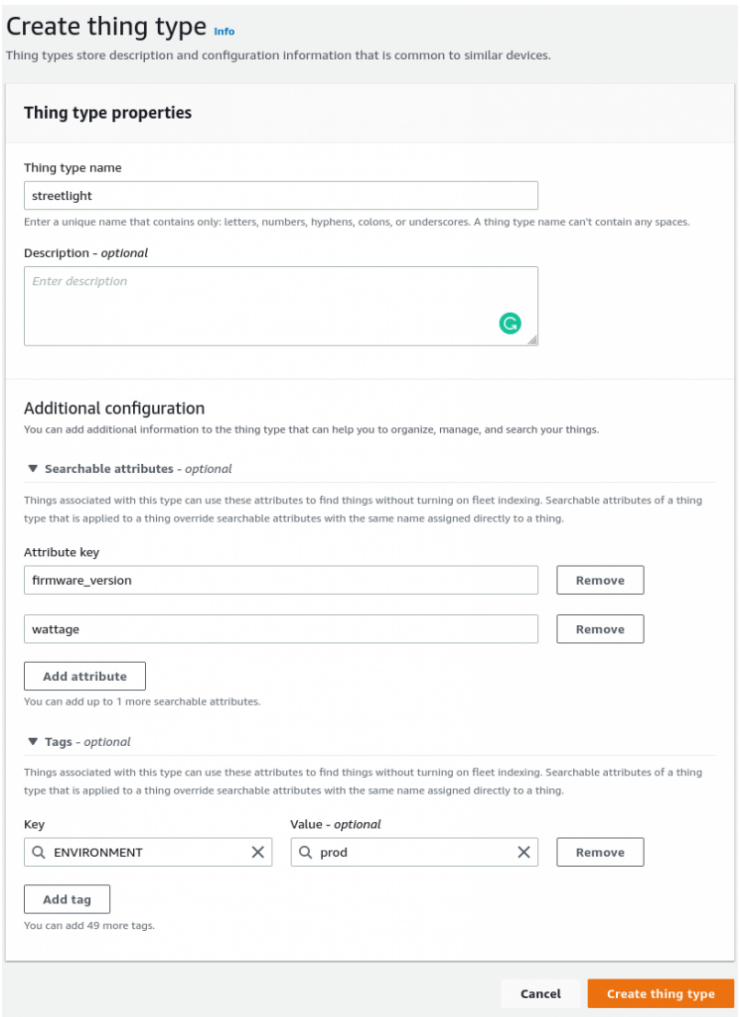
Scalabilità implica anche la capacità del nostro sistema di crescere automaticamente in base al traffico. Le architetture serverless sono progettate per soddisfare questa necessità, quindi ogni volta che aggiungiamo nuove funzionalità, dovremmo cercare una soluzione che non richieda il provisioning di risorse statiche.

AWS IoT Device Management

IoT Device Management è una delle funzionalità più utili di AWS IoT Core. Offre molti strumenti per l'organizzazione della tua flotta di dispositivi in una pagina web centralizzata. Vediamo alcune di queste caratteristiche.

Tipi di oggetti

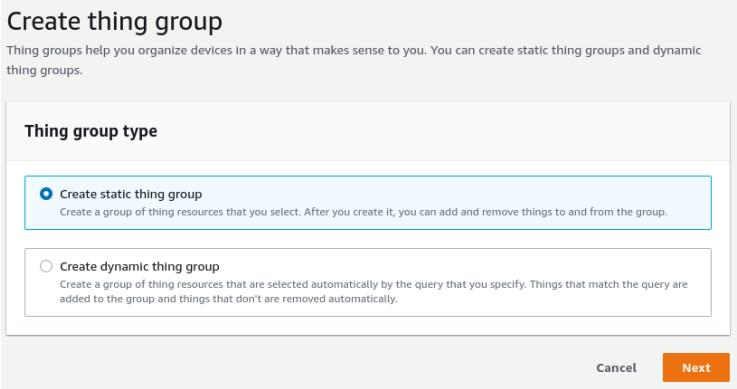
I tipi di oggetti ti consentono di definire una descrizione generale dei tuoi dispositivi e un insieme di attributi per ogni oggetto ad esso associato. Ad un dispositivo IoT può essere associato un unico tipo. I tipi di oggetti sono immutabili, quindi non è possibile aggiungere un attributo al tuo tipo dopo averlo creato. Per fare ciò, è necessario deprecare il tipo e crearne uno nuovo. Non è possibile associare un oggetto a un tipo deprecato. I tipi di oggetti possono essere visti come uno scheletro per la configurazione dei nostri dispositivi: se un dispositivo è di tipo "lampione", avrà gli attributi "potenza" e "versione_firmware".



Gruppi

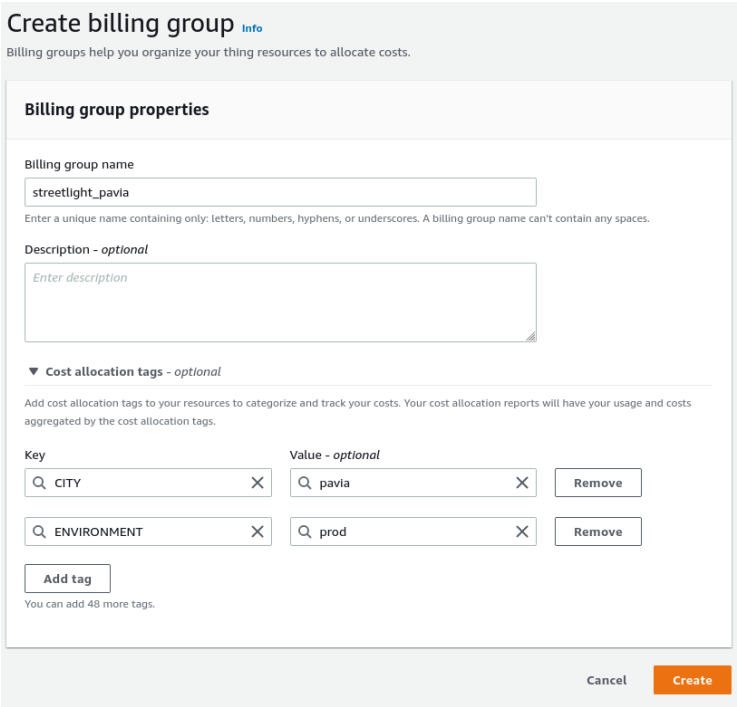
I gruppi ci aiutano a classificare i dispositivi in un modo che abbia senso per noi. Possono definire anche degli attributi, ma a differenza dei tipi, bisogna definre sia il nome dell'attributo che il suo valore e queste informazioni saranno condivise tra tutti i dispositivo che fanno parte di questo gruppo. I gruppi possono essere statici o dinamici. I gruppi statici sono, in un certo senso, simili ai tipi, tranne per il fatto che possono essere annidati gerarchicamente. Sono utili anche se è necessario collegare le stesse policy a più dispositivi, ma si desidera astrarre questa gestione ad un livello superiore. I gruppi dinamici, invece, ci aiutano a raggruppare i dispositivi in modo

assertivo sulla base di condizioni applicate ad una query. Non è possibile associare policy a gruppi dinamici o annidarli l'uno nell'altro, ma è si possono sfruttare se è necessario effattuare delle azioni su un gruppo di dispositivi che soddisfano determinati criteri.



Gruppi di fatturazione

I gruppi di fatturazione sono utili per classificare e tenere traccia dell'utilizzo dei costi. Con i gruppi di fatturazione, è possibile associare dei tag ai nostri dispositivi e utilizzarli per allocare e monitorare i costi. Quando vengono applicati dei tag ai gruppi di fatturazione, AWS genera un report di allocazione dei costi con l'utilizzo e i costi aggregati dai tag in modo da poter determinare come il budget è distribuito nella flotta di dispositivi.



Use Case

Con questi strumenti, possiamo organizzare i nostri dispositivi in maniera granulare. Immaginiamo di avere un'applicazione che gestisce una flotta di lampioni connessi:

come dicevamo, ad ogni dispositivo è associato un *tipo* che definisce un insieme di attributi che ogni dispositivo deve avere (la versione del firmware installato, la potenza assorbita, ecc. .), un gruppo di oggetti statici con una policy associata che definisce l'insieme di azioni consentite, un gruppo dinamico che contiene tutti i dispositivi che hanno installata una vecchia versione del firmware (permettendoci quindi di usare questo gruppo come destinazione per un job di aggiornamento del firmware), e infine un gruppo di fatturazione per ogni città in cui stiamo posizionando i dispositivi (in modo da poter monitorare meglio i costi).

Job

I Job IoT definiscono un insieme di operazioni che vogliamo inviare a una destinazione che può essere un dispositivo specifico o un gruppo di dispositivi. Possiamo definire le operazioni sotto forma di un documento JSON con codifica UTF-8 che contiene uno o più URL utilizzati dai dispositivi per scaricare aggiornamenti e altri dati. I job sono utili per gestire operazioni occasionali come l'aggiornamento del firmware del dispositivo, il download della configurazione locale, il rinnovo dei certificati, ecc.

Ecco un esempio di job che permette ai dispositivi di scaricare un file da un bucket S3, eseguirlo e infine eseguire un riavvio del sistema:

```
{
    "version": "1.0",
    "steps": [
        {
            "action": {
                "name": "Download-File",
                "type": "runHandler",
                "input": {
                    "handler": "download-file.sh",
                    "args": [
                        "${aws:iot:parameter:downloadUrl}",
                        "${aws:iot:parameter:filePath}"
                    ],
                    "path": "${aws:iot:parameter:pathToHandler}"
                },
                "runAsUser": "${aws:iot:parameter:runAsUser}"
```

```
            }
        },
        {
            "action": {
                "name": "Install-Application",
                "type": "runHandler",
                "input": {
                    "handler": "exec-script.sh",
                    "args": [
                        "${aws:iot:parameter:filePath}"
                    ],
                    "path": "${aws:iot:parameter:pathToHandler}"
                },
                "runAsUser": "${aws:iot:parameter:runAsUser}"
            }
        },
        {
            "action": {
                "name": "Reboot",
                "type": "runHandler",
                "input": {
                    "handler": "reboot.sh",
                    "path": "${aws:iot:parameter:pathToHandler}"
                },
                "runAsUser": "${aws:iot:parameter:runAsUser}"
            }
        }
    ]
}
```

Tunnel

Spesso, in un'infrastruttura IoT, è necessario accedere ai propri dispositivi, soprattutto per effettuare debug. In molti casi non è semplice connettersi direttamente al dispositivo, soprattutto quando installato dietro un firewall che blocca gli accessi dall'esterno. I tunnel IoT permettono proprio di ovviare questi problemi di

raggiungibilità. Questa funzione ci consente di stabilire comunicazioni protette bidirezionali con qualsiasi dispositivo e può farci risparmiare tempo e denaro che altrimenti dovremmo allocare per inviare un tecnico sul posto per indagare sul problema. Per creare un tunnel, tutto ciò che serve è configurare quale servizio si desidera utilizzare per stabilire la connessione (ad esempio SSH), il dispositivo a cui ci si vuole connettere e la durata della vita del tunnel. Al termine del processo di creazione, verranno forniti due token di accesso necessari per avviare un proxy locale (è possibile trovare trovare il codice sorgente qui https://github.com/aws-samples/aws-iot-securetunneling-localproxy) sul tuo computer. Dopo aver avviato il proxy locale, sarà possibile accedere via SSH al dispositivo. Il tunnel si chiuderà automaticamente allo scadere del timeout indicato.

Gestione di certificati e policy

La gestione dei certificati è la base per un'infrastruttura ben progettata. AWS IoT fornisce una pagina centralizzata nella console per controllare i certificati dei tuoi dispositivi, ma dovrebbe essere utilizzata più per il monitoraggio che per la gestione. I certificati vengono visualizzati in una visualizzazione tabellare paginata ed è possibile selezionare ciascuna voce per visualizzarne i dettagli. Non è consigliato creare certificati direttamente da questa pagina (a meno che non tu non stia solo testando una nuova soluzione o creando una POC) perché la pulizia sarà impegnativa poiché molte risorse sono collegate tra loro e si potrebbe rischiare di lasciare risorse inutilizzate creando un ambiente caotico. La soluzione ottimale prevede la gestione dei certificati attraverso un processo codificato (Lambda è l'ideale per questo compito). È fondamentale tenere a mente che i certificati sono scaricabili solo in fase di creazione e non sarà recuperarli in seguito, quindi è bene assicurarsi di salvarli in un archivio crittografato persistente per un uso successivo! Un'altra best practice consiste nel creare dispositivi con certificati dedicati per aggiungere un livello di sicurezza alla tua infrastruttura: se uno dei tuoi certificati è compromesso, non esporrai l'accesso a tutti i dispositivi e potrai disabilitare ed eliminare rapidamente il certificato danneggiato.

Le policy definiscono ciò che i nostri dispositivi saranno in grado di fare. È possibile associare una policy ad un certificato o ad un gruppo di oggetti per riutilizzare la stessa policy per più dispositivi. Quando si definiscono delle policy è necessario seguire i consigli di sicurezza standard per la concessione dell'accesso con privilegi minimi alle proprie risorse. In caso contrario si rischia di avere comportamenti

imprevisti ed esporre l'accesso a parti dell'applicazione che devono rimanere nascoste ed inaccessibili dall'esterno.

Per concludere

AWS IoT offre molte funzionalità centralizzate per organizzare le tue risorse senza crearle da zero. Una solida gestione dei dispositivi è la base per un'applicazione affidabile, quindi è consigliato di dedicare un po' di tempo alla progettazione di questa parte nel miglior modo possibile.

Concludiamo così la nostra mini-serie dedicata all'IoT in cui abbiamo affrontato alcuni degli aspetti più critici da considerare durante lo sviluppo diprogetti IoT su AWS.

Volete sapere di più su questi aspetti o affrontarne di nuovi? Fatecelo sapere nei commenti!

Ci vediamo tra 14 giorni su Proud2beCloud per un nuovo articolo!

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!
