
AWS IoT

Manuel du développeur



AWS IoT: Manuel du développeur

Copyright © 2017 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Qu'est-ce que AWS IoT ?	1
Composants AWS IoT	1
Comment démarrer avec AWS IoT	2
Accéder à AWS IoT	2
Services connexes	2
Fonctionnement d'AWS IoT	2
Démarrage rapide du bouton AWS IoT	4
Démarrage avec le bouton AWS IoT à l'aide de la console AWS Lambda	5
Etapes suivantes	17
Démarrage d'AWS IoT	18
Connectez-vous à la console AWS IoT	19
Créer un dispositif dans le Thing Registry	20
Créer et activer un certificat d'appareil	22
Créer une stratégie AWS IoT	25
Attacher une stratégie AWS IoT à un certificat d'appareil	28
Attacher un objet à un certificat	30
Configurer votre dispositif	34
Bouton AWS IoT	34
Afficher les messages MQTT du dispositif avec le client MQTT AWS IoT	35
Configurer et tester des règles	39
Créer une rubrique SNS	40
S'abonner à une rubrique Amazon SNS	41
Créer une règle	42
Tester la règle Amazon SNS	48
Etapes suivantes	51
Didacticiels concernant les règles AWS IoT	53
Création d'une règle DynamoDB	54
Création d'une règle Lambda	67
Créer la fonction Lambda	68
Test de votre fonction Lambda	78
Création d'une règle Lambda	80
Tester votre règle Lambda	84
Gestion des objets avec AWS IoT	88
Gestion des objets avec le Thing Registry	88
Créer un objet	88
Liste des objets	89
Recherche d'objets	89
Mettre à jour un objet	90
Supprimer un objet	90
Attacher un mandataire à un objet	90
Détacher un mandataire d'un objet	90
Types d'objets	90
Créer un type d'objet	91
Liste des types d'objets	91
Décrire un type d'objet	91
Associer un type d'objet à un objet	92
Rendre obsolète un type d'objet	92
Supprimer un type d'objet	93
Sécurité et identité	94
Authentification dans AWS IoT	96
Certificats X.509	96
Utilisateurs, groupes et rôles IAM	103
Identités Amazon Cognito	103
Autorisation	104
Stratégies AWS IoT	104

Sécurité du transport	112
Prise en charge des suites de chiffrement TLS	112
Agent de messages	114
Protocoles	114
MQTT	114
HTTP	115
MQTT via le protocole WebSocket	115
Rubriques	118
Rubriques réservées	118
Événements du cycle de vie	118
Stratégie requise pour recevoir des événements de cycle de vie	119
Événements de connexion/déconnexion	119
Événements d'abonnement/désabonnement	120
Règles	121
Attribuer à AWS IoT l'accès requis	122
Transmettre les autorisations de rôle	123
Création d'une règle AWS IoT	123
Affichage des règles	125
Versions de SQL	125
Nouveautés de la version 2016-03-23-beta du moteur de règles SQL	126
Résolution des problèmes d'une règle	127
Suppression d'une règle	127
Actions de règle AWS IoT	127
Action d'alarme CloudWatch	128
Action de métrique CloudWatch	128
Action DynamoDB	129
L'action Amazon ES	130
Action Kinesis	131
Action lambda	131
Action d'S3	132
Action d'SNS	133
Action Firehose	133
Action d'SQS	134
Action de republication	134
Référence d'SQL AWS IoT	135
Expressions	135
Clause SELECT	136
Clause FROM	137
Clause WHERE	137
Fonctions	138
Extensions JSON	142
Modèles de substitution	142
Device Shadows	144
Flux de données Device Shadows	144
Documents Device Shadows	148
Propriétés du document	148
Versioning d'un thing shadow	149
Jeton client	149
Exemple de document	149
Sections vides	149
Tableaux	150
Utilisation de Device Shadows	150
Prise en charge du protocole	150
Mise à jour d'une thing shadow	151
Récupération d'un document Thing Shadow	151
Suppression des données	152
Suppression d'un thing shadow	153
État Delta	153

Observation des changements d'état	153
Ordre des messages	154
Supprimer des messages Device Shadows	155
API RESTful	155
GetThingShadow	155
UpdateThingShadow	156
DeleteThingShadow	157
Rubriques de publication/abonnement MQTT	157
/update	158
/update/accepted	158
/update/documents	159
/update/rejected	159
/update/delta	159
/get	160
/get/accepted	160
/get/rejected	161
/supprimer	161
/delete/accepted	161
/delete/rejected	162
Syntaxe du document	162
Documents d'état de la demande	162
Documents d'état de la réponse	162
Documents de réponse d'erreur	163
Messages d'erreur	163
SDK AWS IoT	165
SDK pour les appareils	166
Ensemble de fonctions du SDK pour les appareils	166
Plateformes prises en charge	166
SDK pour les appareils pour C AWS IoT	167
Prérequis	167
Connexion à Raspberry Pi	168
SDK pour les appareils pour Javascript AWS IoT	186
Prérequis	187
Connexion à Raspberry Pi	187
Supervision	211
Outils de supervision	212
Outils automatiques	212
Outils manuels	212
Supervision avec Amazon CloudWatch	213
Métriques et dimensions	213
Utilisation de métriques AWS IoT	214
Création d'alarmes CloudWatch	214
Journalisation des appels d'API AWS IoT avec AWS CloudTrail	216
Informations relatives à AWS IoT dans CloudTrail	216
Présentation des entrées des fichiers journaux AWS IoT	217
Dépannage	218
Diagnostic des problèmes de connectivité	218
Authentification	218
Autorisation	218
Configuration de CloudWatch Logs	218
Configuration d'un rôle IAM pour la journalisation	219
Format d'entrée de journal CloudWatch	219
Journalisation des événements et codes d'erreur	220
Diagnostic des problèmes de règles	223
Diagnostic des problèmes de Thing Shadows	223
Limites AWS IoT	225
Limites de l'agent de messages	225
Limites Device Shadow	227

Limites de sécurité et d'identité	228
Limites de limitation	228
Limites de moteur de règles AWS IoT	229

Qu'est-ce que AWS IoT ?

AWS IoT permet une communication sécurisée directionnelle entre les objets connectés à Internet (par exemple, capteurs, actionneurs, périphériques intégrés ou appareils intelligents) et le cloud AWS. Cela vous permet de collecter les données de télémétrie de plusieurs périphériques et de stocker et d'analyser ces données. Vous pouvez également créer des applications qui permettent à vos utilisateurs de contrôler ces appareils à partir de leurs téléphones ou tablettes.

Composants AWS IoT

AWS IoT se compose des éléments suivants :

- Agent de messages— Offre un mécanisme sécurisé pour que les objets et les applications — publient et reçoivent des messages les uns des autres. Vous pouvez utiliser le protocole MQTT directement ou MQTT sur WebSockets effectuer des publications et vous abonner. Vous pouvez utiliser l'interface HTTP REST pour la publication.
- Moteur de règles— Offre des fonctions de traitement des messages et d'intégration avec d'autres services AWS. Vous pouvez utiliser un langage SQL pour sélectionner des données à partir des charges utiles de messages, traiter les données et envoyer les données à d'autres services, par exemple Amazon S3, Amazon DynamoDB, et AWS Lambda. Vous pouvez également utiliser l'agent de messages pour re publier des messages à d'autres abonnés.
- registre d'objets— Parfois appelé Registre des appareils. Organise les ressources associées à chaque objet. Vous enregistrez vos objets et associez jusqu'à trois attributs personnalisés à chaque objet. Vous pouvez également associer des certificats et des ID de client MQTT à chaque objet pour améliorer votre capacité à gérer et dépanner vos objets.
- Service de Thing Shadows— Fournit des représentations permanentes de vos objets dans le cloud AWS. Vous pouvez publier des informations d'état mises à jour dans un fichier thing shadow et votre objet peut synchroniser son état lorsqu'il se connecte. Vos objets peuvent également publier leur état actuel dans fichier thing shadow utilisé par les applications ou les appareils.
- Thing shadow— Parfois appelé device shadow. Document JSON utilisé pour stocker et récupérer des informations d'état actualisées concernant un objet (appareil, applications etc.).
- Passerelle pour les appareils— Permet aux appareils de communiquer de manière efficace et en toute sécurité avec AWS IoT.
- Service de sécurité et d'identité— Partage la responsabilité de la sécurité dans le cloud AWS. Vos objets doivent protéger leurs informations d'identification pour envoyer des données en toute sécurité à l'agent de messages. L'agent de messages et les règles de message utilisent les

fonctions de sécurité AWS pour envoyer des données en toute sécurité vers des appareils ou autres services AWS.

Comment démarrer avec AWS IoT

- Pour plus d'informations sur AWS IoT, consultez [Fonctionnement d'AWS IoT \(p. 2\)](#).
- Pour savoir comment connecter un objet à AWS IoT, consultez [Démarrage d'AWS IoT \(p. 18\)](#).

Accéder à AWS IoT

AWS IoT offre les interfaces suivantes afin de créer et d'interagir avec vos objets :

- AWS Command Line Interface (AWS CLI)— Exécute des commandes pour AWS IoT sur Windows, Mac et Linux. Pour démarrer, consultez [AWS Command Line Interface Guide de l'utilisateur](#). Pour plus d'informations sur les commandes pour AWS IoT, consultez `iot` dans le manuel AWS Command Line Interface Reference.
- Kits de développement logiciel (SDK) AWS— Crée vos applications IoT à l'aide d'API spécifique au langage. Pour plus d'informations, consultez [Kits de développement logiciel et outils AWS](#).
- API AWS IoT— Crée vos applications IoT à l'aide de requêtes HTTP ou HTTPS. Pour plus d'informations sur les actions d'API pour AWS IoT, consultez [Actions](#) dans le manuel Référence d'API AWS IoT.
- Thing SDK for C AWS IoT— Crée des applications IoT pour les objets à ressources limitées, tels que les microcontrôleurs.

Services connexes

AWS IoT s'intègre directement avec les services AWS suivants :

- Amazon Simple Storage Service— Fournit un stockage évolutif dans le cloud AWS. Pour plus d'informations, consultez [Amazon S3](#).
- Amazon DynamoDB— Fournit des bases de données NoSQL gérées. Pour plus d'informations, consultez [Amazon DynamoDB](#).
- Amazon Kinesis— Permet de traiter en temps réel des données diffusées en continu à très grande échelle. Pour plus d'informations, consultez [Amazon Kinesis](#).
- AWS Lambda— Exécute votre code sur des serveurs virtuels à partir de Amazon EC2 en réponse à des événements. Pour plus d'informations, consultez [AWS Lambda](#).
- Amazon Simple Notification Service— Envoie ou reçoit des notifications. Pour plus d'informations, consultez [Amazon SNS](#).
- Amazon Simple Queue Service— Stocke les données dans une file d'attente récupérée par les applications. Pour plus d'informations, consultez [Amazon SQS](#).

Fonctionnement d'AWS IoT

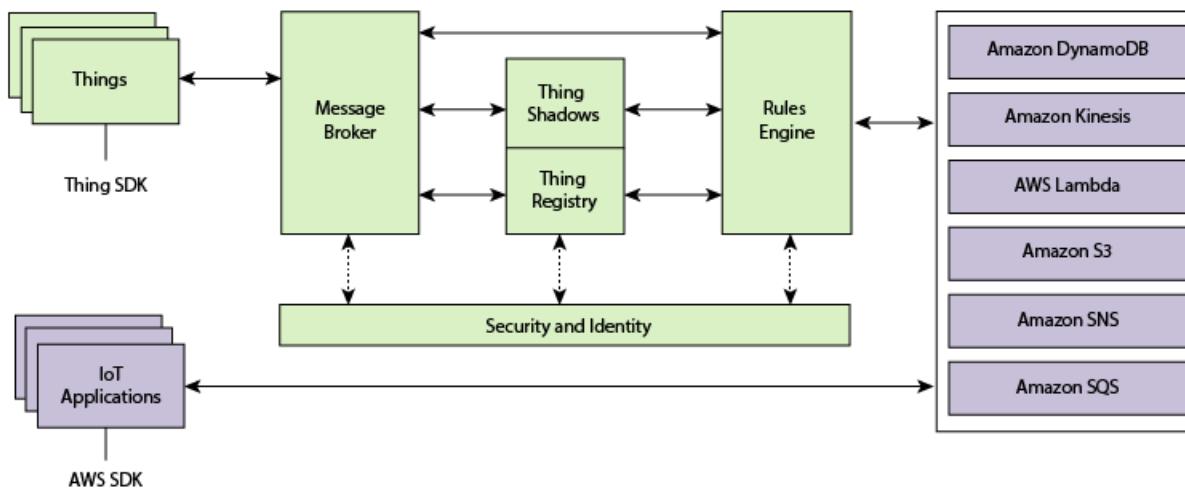
AWS IoT permet à des objets connectés à Internet de se connecter au Cloud AWS et aux applications dans le cloud d'interagir avec les objets connectés à Internet. Les applications IoT courantes recueillent et traitent des données de télémétrie de dispositifs ou permettant aux utilisateurs de contrôler un appareil à distance.

Les objets indiquent leur état en publiant des messages, au format JSON, dans des rubriques MQTT. Chaque rubrique MQTT possède un nom hiérarchique qui identifie l'objet dont l'état est mis à jour. Lorsqu'un message est publié dans une rubrique MQTT, le message est envoyé à l'agent de messages AWS IoT MQTT qui est responsable de l'envoi de tous les messages publiés dans une rubrique MQTT à tous les clients abonnés à cette rubrique.

La communication entre un objet et AWS IoT est protégé par l'utilisation de certificats X.509. AWS IoT peut générer un certificat ou vous pouvez utiliser le vôtre. Dans les deux cas, le certificat doit être enregistré et activé avec AWS IoT. Une fois que le certificat a été enregistré et activé, il doit être copié dans votre objet. Quand votre objet communique avec AWS IoT, il présente le certificat à AWS IoT sous la forme d'informations d'identification.

Nous recommandons que toutes les objets qui se connectent à AWS IoT possèdent une entrée dans le registre d'objets. Le registre d'objets stocke des informations concernant un objet et les certificats utilisés par l'objet pour sécuriser les communications avec AWS IoT.

Vous pouvez créer des règles qui définissent une ou plusieurs actions à exécuter en fonction des données d'un message. Vous pouvez, par exemple, insérer, mettre à jour ou interroger une table DynamoDB ou appeler une fonction Lambda. Les règles utilisent des expressions pour filtrer les messages. Lorsqu'une règle correspond à un message, le moteur de règles appelle l'action en utilisant les propriétés sélectionnées. Vous pouvez utiliser toutes les propriétés JSON dans un message ou uniquement les propriétés dont vous avez besoin. Les règles contiennent également un rôle IAM qui accorde une autorisation AWS IoT aux ressources AWS utilisée afin d'effectuer l'action.



Chaque objet possède un fichier thing shadow qui stocke et récupère les informations d'état. Chaque élément des informations d'état a deux entrées : le dernier état signalé par l'objet et l'état souhaité demandé par une application. Une application peut demander les informations d'état actuel d'un objet. Le fichier shadow répond à la demande en fournissant un document JSON contenant les informations d'état (signalées et souhaitées), les métadonnées et un numéro de version. Une application peut contrôler un objet en demandant une modification de son état. Le fichier shadow accepte la demande de modification de l'état, met à jour les informations d'état et envoie un message pour indiquer que les informations d'état ont été mises à jour. L'objet reçoit le message, change son état, puis indique son nouvel état.

Démarrage rapide du bouton AWS IoT

Utilisez l'Assistant de bouton AWS IoT de la console AWS Lambda afin de configurer rapidement et facilement votre bouton AWS IoT. La console AWS Lambda contient un plan qui permet d'automatiser le processus de configuration de votre bouton AWS IoT en :

- Créeant et activant un certificat X.509 et d'une clé privée pour l'authentification auprès de AWS IoT.
- Vous guidant dans la configuration de votre bouton AWS IoT afin de vous connecter à votre réseau Wi-Fi.
- Vous guidant lors du processus de copie de votre certificat et de la clé privée dans votre bouton AWS IoT.
- Créeant et attachant au certificat une stratégie AWS IoT qui accorde au bouton l'autorisation d'effectuer des appels à AWS IoT.
- Créeant une règle AWS IoT qui appelle une fonction Lambda lorsque vous appuyez sur votre bouton AWS IoT.
- Créeant un rôle IAM et une stratégie qui permet à la fonction Lambda d'envoyer des e-mails à l'aide de Amazon SNS.
- Créeant une fonction Amazon SNS qui envoie un message électronique à l'adresse spécifiée dans le code de la fonction Lambda.



Si vous n'avez pas de bouton, vous pouvez en acheter un [ici](#). Pour plus d'informations sur AWS IoT, consultez [Qu'est-ce que AWS IoT ? \(p. 1\)](#).

Rubriques

- [Démarrage avec le bouton AWS IoT à l'aide de la console AWS Lambda \(p. 5\)](#)
- [Etapes suivantes \(p. 17\)](#)

Démarrage avec le bouton AWS IoT à l'aide de la console AWS Lambda

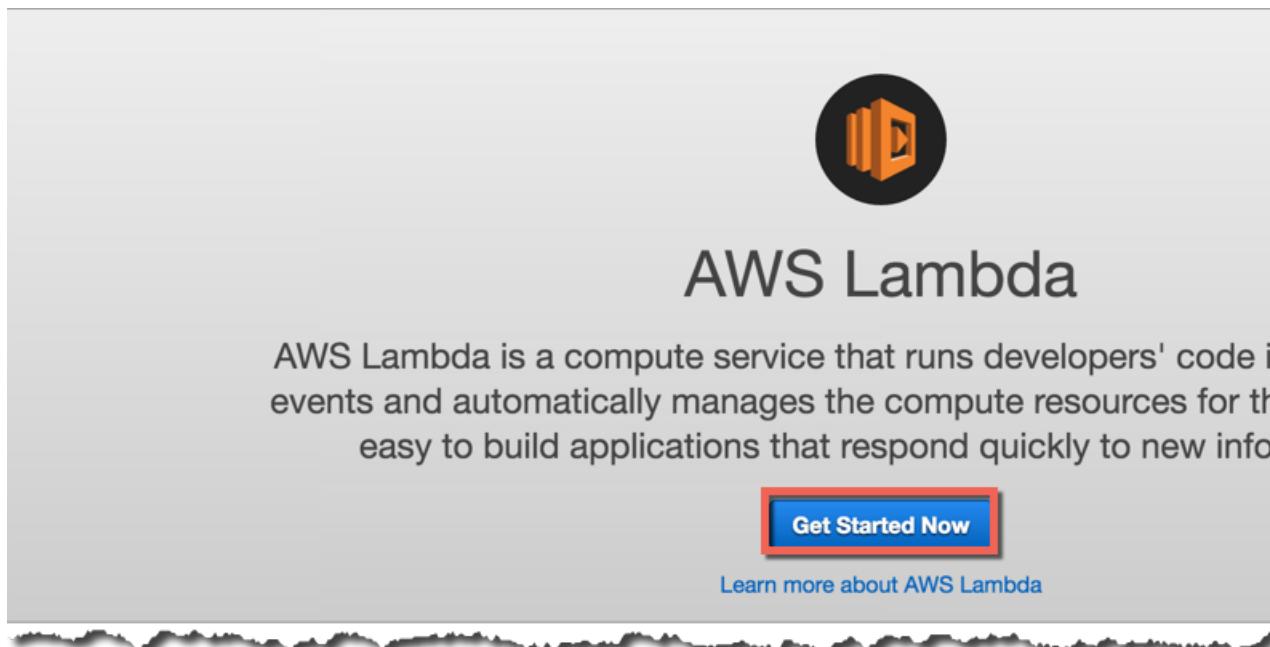
L'Assistant de bouton AWS IoT étant un plan Lambda, vous devez vous connecter à la console AWS Lambda pour pouvoir l'utiliser. Si vous ne possédez pas de compte AWS, vous pouvez en créer un en procédant comme suit.

Pour créer un compte AWS

1. Ouvrez la [page d'accueil d'AWS](#) et sélectionnez Créez un compte AWS..
2. Suivez les instructions en ligne. Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisissez un code PIN sur le clavier numérique du téléphone.

Pour configurer le bouton AWS IoT

1. Connectez-vous à AWS Management Console et ouvrez la [console AWS Lambda](#).
2. Si c'est la première fois que vous utilisez la console Lambda, vous verrez la page suivante. Cliquez sur le bouton Pour commencer .



Si vous avez déjà utilisé la console Lambda, vous verrez la page suivante. Cliquez sur le bouton Crée une fonction Lambda .

The screenshot shows the AWS Lambda Functions page. On the left, there's a sidebar with "AWS Lambda" at the top, followed by "Dashboard BETA" and "Functions". The main area has a breadcrumb "Lambda > Functions". A message says "You have 32 Lambda function(s) using 1.6 MB of code storage. Choose any Lambda function to take up to 60 seconds to appear.". There's a red-bordered "Create a Lambda function" button. Below it is a table with columns "Function name" and "Description". Two functions are listed: "myButtonFunction" (description: "An AWS Lambda function that sends an email when the click of an IoT button.") and "michgreFunction" (description: "A starter AWS Lambda function").

	Function name	Description
<input type="radio"/>	myButtonFunction	An AWS Lambda function that sends an email when the click of an IoT button.
<input type="radio"/>	michgreFunction	A starter AWS Lambda function.

3. Sur la page Plan, dans le menu déroulant Runtime , sélectionnez Node.js 4.3. Dans la zone de texte de filtre, tapez **button**. Pour choisir le plan iot-button-email , double-cliquez dessus ou cliquez sur le bouton Suivant .

Lambda > New function

Select blueprint

Configure triggers

Configure function

Review

Select blueprint

Blueprints are sample configurations of event sources and Lambda functions and customize as needed, or skip this step if you want to author a Lambda function otherwise noted, blueprints are licensed under [CC0](#).

Welcome to AWS Lambda! You can get started on creating your first Lambda function.

Node.js 4.3

button

iot-button-email

An AWS Lambda function that sends an email on the click of an IoT button.

nodejs · iot · button

Feedback English

© 2008 - 2016, Amazon Web Services, Inc. et/ou ses filiales. Tous droits réservés.

- Sur la page Configurer les déclencheurs Plan, dans le menu déroulant Type d'IoT , sélectionnez Bouton IoT.

Saisissez le numéro de série de votre dispositif. Vous trouverez le numéro de série du dispositif (DSN) au dos du bouton.

Choisissez Générer des certificats et des clés.



Note

Vous avez besoin de générer un certificat et une clé privée une seule fois. Vous pouvez ensuite accéder à <http://192.168.0.1/index.html> dans un navigateur pour configurer votre bouton.

Lambda > New function using blueprint iot-button-email

Select blueprint

Configure triggers

Configure function

Review

Configure triggers

Configure an optional trigger to automatically invoke your function.



Warning: Altering the description or SQL statement of an existing rule

IoT Type

IoT Button

Device Serial Number

Generate certificate and private key

Utilisez ces liens sur la page pour télécharger le certificat de l'appareil et la clé privée.

[Generate certificate and keys](#)



We have created the necessary AWS IoT resources (thing, policy, certificate, private key). The remaining resources (rule and function) will be created after your function is created.

Download these resources by clicking the links below. (NOTE: If you are using Internet Explorer or Safari, right click save the files.)

- a. [Your certificate PEM](#)
- b. [Your private key](#)

To configure the AWS IoT Button to use your Wi-Fi and these resources to connect to AWS securely, follow these steps:

1. Place the button into configuration mode by pressing the button down for 5 seconds until it flashes blue.
2. Connect your computer to the button's Wi-Fi network SSID "Button ConfigureMe - FFD", using "5364XVRB" (last 8 digits of the serial number) as the WPA2-PSK password.
3. Click [here](#) (opens in new tab) and use the following information to fill out the form:
 - a. Enter your local network's Wi-Fi SSID and password.
 - b. Select the certificate and private key files that you just downloaded above.
 - c. Your endpoint subdomain is **a182jd32qs965e**.
 - d. Your endpoint region is **us-east-1**.
 - e. Check the box to agree to the terms and conditions.
 - f. Click "configure".
4. Re-connect to your original Wi-Fi network.

The button should stop blinking blue and you will see a white blinking light followed by a green solid light. Your button is now connected to the internet and AWS! Continue creating your function, and your button will be connected to it automatically.



La page inclut également des instructions pour configurer votre bouton AWS IoT. À l'étape 3, vous allez cliquer un lien pour ouvrir une page web qui vous permet de connecter le bouton AWS IoT à votre réseau. Sous Configuration Wi-Fi, tapez l'ID (SSID) et le mot de passe de votre réseau Wi-Fi. Sous Configuration AWS IoT, sélectionnez le certificat et la clé privée que vous avez téléchargé plus tôt. Cela permet de les copier dans votre bouton AWS IoT. Activez la case à cocher pour accepter les conditions générales relatives au bouton AWS IoT, puis cliquez sur le bouton Configuration .

Enter the value for any field that you wish to change.

Wi-Fi Configuration:

SSID	<input type="text" value="Guest"/> 
Security	<input checked="" type="checkbox"/> Open Network(No Password)
Password	<input type="text" value="None (unsecured)"/>

AWS IoT Configuration:

Certificate	<input type="button" value="Browse..."/> certificate.pem
Private Key	<input type="button" value="Browse..."/> private.key
Endpoint Subdomain	<input type="text" value="A3T2RR9XSNT91O"/>
Endpoint Region	<input type="text" value="us-west-2"/> 
Final Endpoint	<input type="text" value="REDACTED.iot.us-west-2.amazonaws.com"/> 

By clicking this box, you agree to the [AWS IoT Button Terms and Conditions](#).

Une page de confirmation de configuration apparaît.

Button Config

Thank you for config

If you are unable to use your device, please

- 
5. Fermez l'onglet Configuration et retournez à la page de la console AWS Lambda. Choisissez Activer le déclencheur, puis sélectionnez Suivant.

Sur la page Configurer une fonction , tapez un nom pour votre fonction. La description, le runtime et le code de fonction Lambda sont déjà renseignés.

Lambda > New function using blueprint iot-button-email

Select blueprint

Configure triggers

Configure function

Review

Configure function

A Lambda function consists of

Lambda function code

Provide the code for your function. You can add dependencies and libraries, you can upload your own code or use a pre-existing blueprint.

We have restored the code from the previous step.

```
1  /**
2   * This is a sample Lambda function for an AWS IoT button.
3   * It sends an email when the button is pressed.
4   * To the top of the file, add the required imports:
5   *
6   * Follow the steps below to update the code:
7   *
8   * 1. Update the recipient email address.
9   * 2. Enter a subject for the email.
```

Dans le code de fonctionLambda , remplacez l'exemple d'adresse e-mail par votre propre adresse e-mail.

```
1  /**
2   * This is a sample Lambda function that sends an
3   * button. It creates a SNS topic, subscribes an e-
4   * to the topic and publishes to the topic.
5   *
6   * Follow these steps to complete the configuration:
7   *
8   * 1. Update the EMAIL variable with your email address.
9   * 2. Enter a name for your execution role in the
10  *    Your function's execution role needs specific permissions
11  *    to send an email. We have pre-selected the "Amazon SNS
12  *    policy template that will automatically add the required
13  */
14
15 const EMAIL = 'my_email@example.com'; // TODO change this to your email
16
17 const AWS = require('aws-sdk');
18 const SNS = new AWS.SNS({ apiVersion: '2010-03-31' });
19
20 function findExistingSubscription(topicArn, nextToken) {
21     const params = {
22         TopicArn: topicArn,
23         NextToken: nextToken || null,
24     };
25     SNS.listSubscriptionsByTopic(params, (err, data) => {
26         if (err) {
27             console.error(`Error listing subscriptions for topic ${topicArn}: ${err}`);
28         } else {
29             const subscriptions = data.Subscriptions;
30             if (subscriptions.length > 0) {
31                 const subscriptionArn = subscriptions[0].SubscriptionArn;
32                 return { subscriptionArn, token: subscriptions[0].Token };
33             }
34         }
35     });
36 }
37
38 exports.handler = async (event) => {
39     const { topicArn } = event;
40
41     const existingSubscription = await findExistingSubscription(topicArn);
42
43     if (!existingSubscription) {
44         const params = {
45             TopicArn: topicArn,
46             Protocol: 'email',
47             Endpoint: EMAIL,
48         };
49         SNS.subscribe(params, (err, data) => {
50             if (err) {
51                 console.error(`Error subscribing to topic ${topicArn}: ${err}`);
52             } else {
53                 const subscriptionArn = data.SubscribeResponse.SubscriptionArn;
54                 return { subscriptionArn };
55             }
56         });
57     } else {
58         return { subscriptionArn: existingSubscription.subscriptionArn };
59     }
60 }
```

Dans la section Gestionnaire de la fonction Lambda et rôle , dans le menu déroulant Rôle , sélectionnez Crée un rôle à partir de modèles. Tapez un nom de rôle pour le rôle.

Lambda function handler and role

Handler*

index.handler

Role*

Create new role from

Lambda will automatically
Lambda permissions (logg
VPC permissions will also

Role name

myIoTButtonRole

Policy templates

 AWS IoT Button p

Dans le bas de la page, cliquez sur le bouton Suivant.

Passez en revue les paramètres de la fonction Lambda, puis choisissez Créeer la fonction.

[Lambda](#) > New function using blueprint iot-button-email

[Select blueprint](#)

[Configure triggers](#)

[Configure function](#)

Review

Review

Please review your Lambda function configuration to complete the setup process.

Triggers

Lambda function

Vous devez voir une page qui confirme la création de votre fonction Lambda :

The screenshot shows the AWS Lambda function configuration interface. At the top, there's a navigation bar with icons for AWS, Services, and Edit. Below it, the path 'Lambda > Functions > myButtonFunction' is shown. Underneath the path are three buttons: 'Qualifiers', 'Test' (which is highlighted in blue), and 'Actions'. A large green box contains the message 'Congratulations! Your Lambda function "myButtonFunction" has been created.' Below this, there are tabs for 'Code', 'Configuration', 'Triggers' (which is also highlighted in orange), and 'Monitoring'. Under the 'Triggers' tab, there's a section for 'AWS IoT: iotbutton_G030JF055364XVRB'. It shows the ARN: arn:aws:iot:us-east-1:...:rule/iotbutton_... and a 'Rule Description: Event source for your IoT Button to Lambda function'. At the bottom of this section is a blue button labeled '+ Add trigger'.

6. Pour tester votre fonction Lambda, cliquez sur le bouton Test . Après environ une minute, vous devriez recevoir un message électronique avec pour objet Notification d'AWS - Confirmation d'abonnement . Cliquez sur le lien dans l'e-mail afin de confirmer l'abonnement à une rubrique SNS créée par la fonction Lambda. Lorsque AWS IoT reçoit un message de votre bouton, il envoie un message à Amazon SNS. La fonction Lambda crée un abonnement à la rubrique Amazon SNS à l'aide de l'adresse e-mail que vous avez ajoutée dans le code.

Lorsque Amazon SNS reçoit un message concernant cette rubrique Amazon SNS, il le transfère à l'adresse e-mail de l'abonnement.

Appuyez sur le bouton pour envoyer un message à AWS IoT. Le message déclenchera votre règle Lambda, puis votre fonction Lambda sera appelée. La fonction Lambda vérifie si votre rubrique SNS existe. La fonction Lambda envoie ensuite le contenu du message à la rubrique Amazon SNS. Amazon SNS transmettra ensuite le message à l'adresse e-mail spécifiée dans le code de fonction Lambda.

Etapes suivantes

Pour en savoir plus sur le plan Lambda utilisé pour configurer votre bouton, consultez la rubrique [Démarrage d'AWS IoT](#).

Démarrage d'AWS IoT

Cette section vous guidera dans la création des ressources requises pour envoyer, recevoir et traiter les messages MQTT provenant de dispositifs utilisant AWS IoT. Vous aurez besoin d'un ordinateur avec un accès Wi-Fi pour suivre ce didacticiel. Si vous avez acheté un bouton AWS IoT (illustré ici), vous pouvez cliquer sur ce bouton pour suivre ce didacticiel.



Si vous n'avez pas de bouton, vous pouvez en acheter un [ici](#) ou vous pouvez utiliser le client MQTT dans la console AWS IoT pour suivre ce didacticiel. Pour plus d'informations sur AWS IoT, consultez [Qu'est-ce que AWS IoT ? \(p. 1\)](#).



Note

Ce didacticiel utilise Amazon SNS, qui n'est pas disponible dans toutes les régions. Lorsque vous créez des ressources AWS pour ce didacticiel, prenez soin de vous connecter à la région USA Est (Virginie du Nord). Pour plus d'informations sur les régions AWS, consultez [Régions et points de terminaison](#).

Rubriques

- [Connectez-vous à la console AWS IoT \(p. 19\)](#)

- Créez un dispositif dans le Thing Registry (p. 20)
- Créez et activez un certificat d'appareil (p. 22)
- Créez une stratégie AWS IoT (p. 25)
- Attachez une stratégie AWS IoT à un certificat d'appareil (p. 28)
- Attachez un objet à un certificat (p. 30)
- Configurer votre dispositif (p. 34)
- Afficher les messages MQTT du dispositif avec le client MQTT AWS IoT (p. 35)
- Configurer et tester des règles (p. 39)
- Etapes suivantes (p. 51)

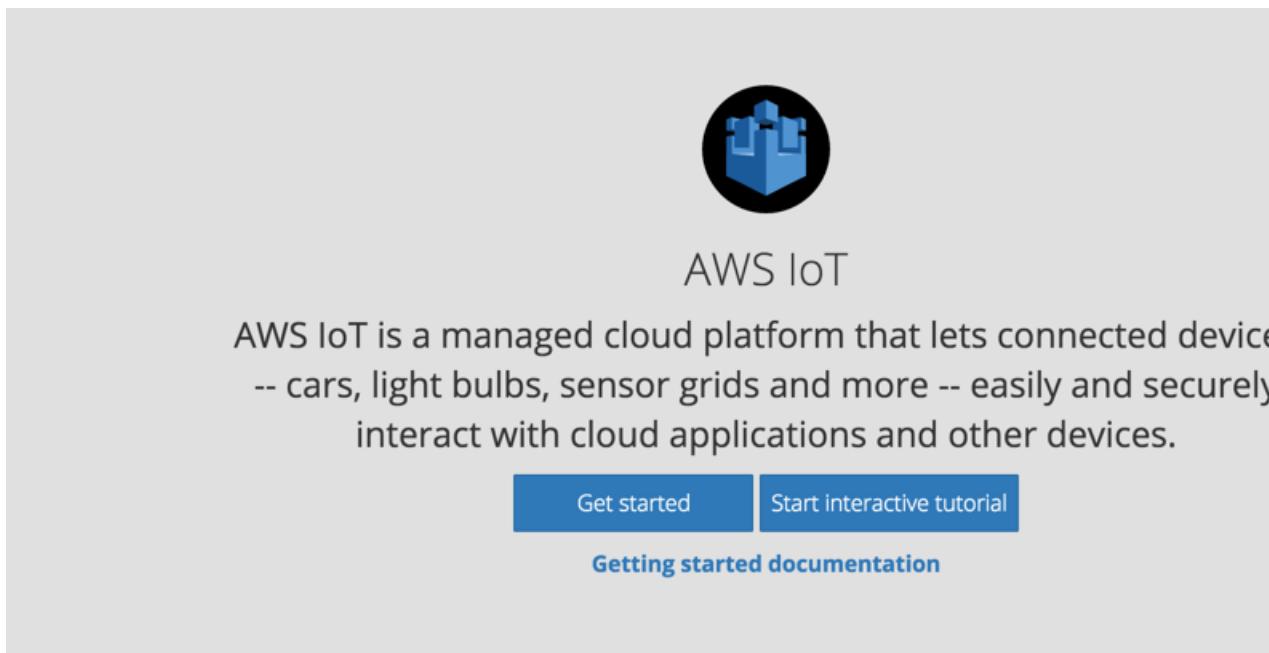
Connectez-vous à la console AWS IoT

Si vous ne possédez pas de compte AWS, créez-en un.

1. Ouvrez la [page d'accueil d'AWS](#) et sélectionnez Créez un compte AWS..
2. Suivez les instructions en ligne. Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisissez un code PIN sur le clavier numérique du téléphone.
3. Connectez-vous à AWS Management Console et ouvrez la [Console IoT AWS](#).
4. Sur la Bienvenue , sélectionnez Démarrer avec AWS IoT.



5. S'il s'agit de votre première utilisation de la console AWS IoT, deux options se présentent à vous : Démarrer and Démarrer le didacticiel interactif. Choisissez Mise en route.



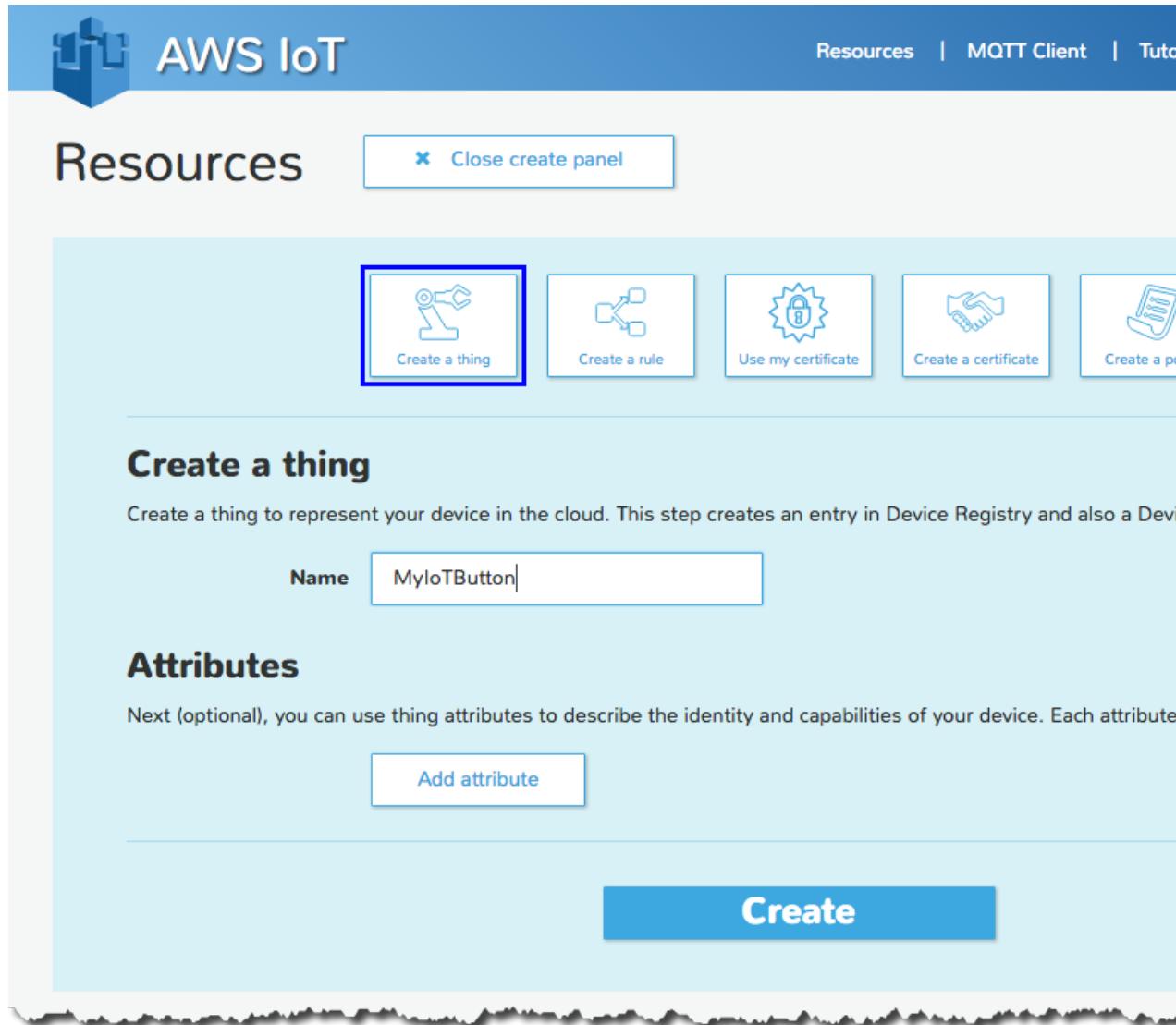
6. Sur la page Ressources , si vous ne voyez pas une bannière bleue avec les boutons Crée un objet, Crée une règle, Crée un certificat, and Crée une stratégie , sélectionnez Crée une ressource.

The image shows the "Resources" section of the AWS IoT console. At the top left is the AWS IoT logo. To its right are links for "Resources" and "Tutorial". Below the logo, the word "Resources" is displayed next to a blue "Create a resource" button, which is highlighted with a red rectangular box. To the left of the "Create a resource" button is a search bar with the placeholder text "Filter by resource names or by resource type (below)". Below the search bar are four status indicators: "All" (highlighted), "0/0 things", "0/0 rules", "0/0 certificates", and "0/0 policies". On the right side of the screen are "Select all" and navigation buttons for "First" and "Previous".

Créer un dispositif dans le Thing Registry

Pour connecter un dispositif à AWS IoT, nous recommandons de créer d'abord un dispositif dans le Thing Registry. Ce registre permet de conserver un registre de tous les dispositifs connectés à votre compte AWS IoT.

1. Choisissez Créez un objet et tapez un nom pour votre dispositif. Vous pouvez également sélectionner Ajouter un attribut pour fournir des informations concernant votre dispositif (par exemple, son numéro de série, fabricant, etc.). Choisissez Créez pour ajouter votre dispositif au Thing Registry.



2. Choisissez Afficher l'objet pour afficher les informations concernant votre dispositif.

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below it, the main title "Resources" is displayed. To the right of the title is a button labeled "Close create panel". On the left side of the main area, there's a large button labeled "Create a thing" with a robotic arm icon, which is highlighted with a blue border. Other buttons include "Create a rule" (with a gear and link icon), "Use my certificate" (with a lock icon), "Create a certificate" (with a handshake icon), and "Create a policy" (with a document icon). A message below these buttons says "Your new thing has been created. Click 'View thing' to continue." Another message below it says "From there, you can connect a device to this thing, or add a rule to take actions when your thing publishes a message." At the bottom of the main area is a large blue button labeled "View thing". Below this, there's a search/filter bar with the placeholder "Filter by resource names or by resource type (below)" and a "Select all" checkbox. Underneath the search bar, there are summary statistics: "All 1/1 things 0/0 rules 0/0 CAs 0/0 certificates 0/0 policies". On the far right, there are "First" and "Last" navigation buttons. The main content area displays a single resource card for "MyIoTButton". The card has a blue border and contains the name "MyIoTButton" at the top, followed by two small icons: a robotic arm and a square.

Créer et activer un certificat d'appareil

La communication entre votre bouton AWS IoT et AWS IoT est protégé par l'utilisation de certificats X.509. AWS IoT peut générer un certificat à votre place ou vous pouvez utiliser votre propre certificat X.509. Ce guide suppose que AWS IoT génère le certificat X.509 à votre place. Les certificats doivent être activés avant utilisation.

1. Dans la section Créez un certificat , sélectionnez Créez un certificat avec 1-Click.

The screenshot shows the AWS IoT Resources page. At the top, there's a blue header bar with the AWS IoT logo and a 'Resources' link. Below the header, the word 'Resources' is displayed in large, bold, dark gray letters. To its right is a button labeled 'Close create panel'. Underneath, there are four buttons: 'Create a thing' (with a wrench icon), 'Create a rule' (with a network icon), 'Use my certificate' (with a lock icon), and 'Create a certificate' (with a handshake icon). The 'Create a certificate' button is highlighted with a blue border. A large, semi-transparent overlay box covers the middle portion of the page. Inside this box, the title 'Create a certificate' is shown in bold black font. Below it is a subtitle: 'Create a certificate to authenticate your device's connection to AWS IoT.' A note follows: 'You can generate a certificate with 1-click (recommended), or you can upload your own certificate signing request'. Two buttons are present: 'Create with CSR' (with a file icon) and '1-Click certificate create' (with a download icon). Below these buttons is a message: 'You don't have any CAs registered yet. Register your CA certificate'. The bottom of the overlay has a decorative wavy pattern.

2. Sur la page Ressources , cliquez sur les liens Télécharger la clé privée et Télécharger le certificat , puis enregistrez la clé privée et le certificat sur votre ordinateur.

The screenshot shows the AWS IoT Resources page. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below it, the main title "Resources" is displayed. To the right of the title is a "Close create panel" button. Underneath the title, there are five action buttons: "Create a thing" (robot icon), "Create a rule" (gear and chain icon), "Use my certificate" (padlock icon), "Create a certificate" (handshake icon, which is highlighted with a blue border), and "Create a policy" (document icon). A message below these buttons states: "Your new certificate has been created. You can attach a certificate to a thing so it can connect to AWS IoT and attach permissions." It also says: "Please download these files and save them in a safe place. Certificates can be retrieved at any time, but the private key will not be retrievable after closing this form." Below this message, there's a list of download links: "Download public key", "Download private key", and "Download certificate".

Filter by resource names or by resource type (below)

All 1/1 things 0/0 rules 0/0 CAs 1/1 certificates
0/0 policies

MyIoTButton

INACTIVE

Select all

First

3. Activez la case à cocher sur le certificat et, dans le menu Actions , sélectionnez Activation.

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "Resources". Below it, a white panel contains five buttons: "Create a thing", "Create a rule", "Use my certificate", "Create a certificate" (which is highlighted with a blue border), and "Create a policy". A message below these buttons says: "Your new certificate has been created. You can attach a certificate to a thing so it can connect to AWS IoT and attach permissions." It also instructs the user to download the public key, private key, and certificate. At the bottom, there's a table-like view of resources. The first row shows filters: "All" (selected), "1/1 things", "0/0 rules", "0/0 CAs", "1/1 certificates", and "0/0 policies". The second row lists two items: "INACTIVE" (with a handshake icon and a checked checkbox) and "MyIoTButton" (with a wrench and gear icon). The "MyIoTButton" item is also highlighted with a blue border.

Créer une stratégie AWS IoT

Les certificats X.509 sont utilisés pour authentifier votre bouton AWS IoT. Les stratégies AWS IoT sont utilisées pour autoriser votre bouton à effectuer des opérations AWS IoT, telles que l'abonnement ou la publication dans des rubriques MQTT. Votre bouton présentera son certificat lors de l'envoi de messages à AWS IoT. Pour permettre à votre bouton d'effectuer des opérations AWS IoT, vous devez créer une stratégie AWS IoT et l'attacher au certificat de votre dispositif.

1. Dans la console AWS IoT, si vous ne voyez pas le volet Créer , sélectionnez Crée une ressource.

2. Choisissez Créer une stratégie.
3. Dans la section Crée une stratégie , tapez un nom pour la stratégie. Dans le menu Action , sélectionnez iot:Publish. Dans le champ Ressource , tapez l'ARN de votre bouton AWS IoT, puis activez la case à cocher Autoriser . Cela permet à votre bouton de publier des messages dans AWS IoT.



Note

L'ARN de votre AWS IoT bouton a la forme suivante :

```
arn:aws:iot:<your-region>:<your-aws-account>:topic/iotbutton/<your-button-serial-number>
```

Exemples :

```
arn:aws:iot:us-east-1:123456789012:topic/iotbutton/G030JF055364XVRB
```

Le numéro de série figure dans le bas de votre bouton.

Choisissez Ajouter une instruction, puis sélectionnez Créer.

The screenshot shows the AWS IoT Resources page. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the main title "Resources" is displayed in large, dark text. To the right of the title is a button labeled "Close create panel" with a small "X" icon. The main content area is titled "Create a policy" in bold. Below this, a sub-instruction says "Create a policy to define a set of authorized actions. You can au...". On the right side of the policy creation panel, there are two columns: "Name" and "Action". The "Name" column contains the text "Mylo" in an orange box. The "Action" column contains the text "iot:Pu..." in an orange box. Below these columns is a blue button with the text "Add" in white. The bottom of the page features a decorative wavy footer.

AWS IoT

Resources

[Close create panel](#)

Create a policy

Create a policy to define a set of authorized actions. You can au...

Name	Action
Mylo	iot:Pu...

Add

Pour plus d'informations sur les stratégies AWS IoT, consultez [Gestion des stratégies AWS IoT](#).

Attacher une stratégie AWS IoT à un certificat d'appareil

Maintenant que vous avez créé une stratégie, vous devez l'attacher au certificat de votre appareil. Attacher une stratégie AWS IoT à un certificat donne à l'appareil les autorisations spécifiées dans la stratégie.

1. À partir de la console AWS IoT, sélectionnez le certificat de votre appareil, puis dans le menu Actions , sélectionnez Attacher une stratégie.

AWS IoT

Resources

[✖ Close create panel](#)

Your new policy has been created. You can view your policy and edit it.

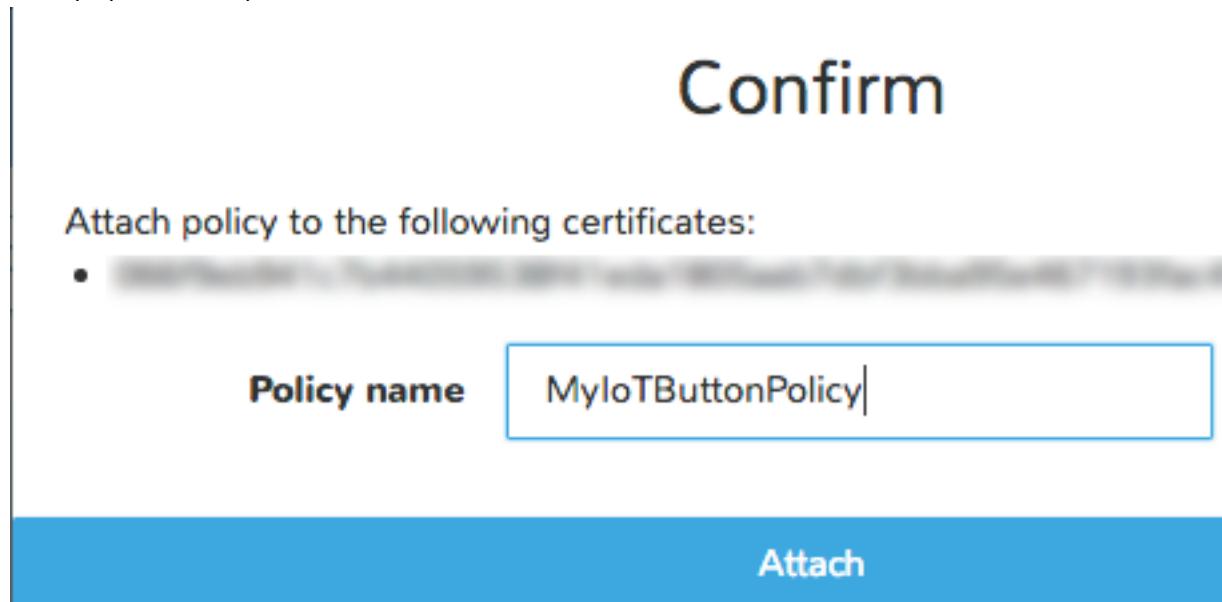
Filter by resource names or by resource type (below)

All 0/1 things 0/0 rules 0/0 CAs **1/1 certificates** 0/2 policies

ACTIVE	

29

2. Dans la boîte de dialogue Confirmer , tapez le nom de la stratégie AWS IoT que vous avez créée à l'étape précédente, puis sélectionnez Attacher.



Attacher un objet à un certificat

Pour attacher un certificat à un dispositif dans le Thing Registry :

1. Dans la console AWS IoT, sélectionnez le certificat que vous voulez attacher et, dans le menu Actions , sélectionnez Attacher un objet.

The screenshot shows the AWS IoT Resources page. At the top, there is a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the page title "Resources" is displayed in large, bold, dark text. To the right of the title is a button labeled "+ Create a resource". Underneath the title, there is a search/filter bar with a magnifying glass icon and the placeholder text "Filter by resource names or by resource type (below)". Below the search bar, there are five categories with counts: "All" (selected), "1/1 things", "1/1 rules", "0/0 CAs", and "1/1 certificates". Under "things", there is also a count of "0/0 policies". The main content area displays two resource cards. The first card, titled "MyIoTButton", has a "Edit" icon and a "Delete" icon at the bottom. The second card, which appears to be a certificate, has a "View Details" icon and a "Checkmark" icon at the bottom. The certificate card also shows a long string of characters: "3b95b9d18852d75d6 d3f1d405d284e53f3 1a9f3c67f9bcfd5af...". The word "ACTIVE" is displayed in blue text below the certificate ID.

2. Dans la boîte de dialogue Confirmer , tapez le nom de l'objet à attacher, puis cliquez sur Attacher.

Confirm

Attach the following certificates to a thing:

- [REDACTED]

Thing name

MyIoTButton

Attach

3. Pour vérifier que l'objet est attaché, double-cliquez sur le certificat. La stratégie et l'objet doivent apparaître dans le volet de détails.

AWS IoT

Resources

[Close create panel](#)

[Create a thing](#) [Create a rule](#) [Use my certificate](#)

Your new policy has been created. You can view your policy and edit it at any time.

[View policy](#)

Filter by resource names or by resource type (below)

All 1/1 things 1/1 rules 0/0 CAs 1/1 certificates
1/1 policies

MyloTButtonPolicy	MyloTButton	3b95b9d1 75d6d3f1 84e53f31 ACTIVE

33

Configurer votre dispositif

La configuration de votre dispositif lui permet de se connecter à votre réseau Wi-Fi. Votre dispositif doit être connecté à votre réseau Wi-Fi pour permettre l'installation du certificat d'appareil et pour permettre à votre dispositif d'envoyer des messages à AWS IoT. Tous les dispositifs doivent posséder un certificat d'appareil afin de communiquer avec AWS IoT.

Bouton AWS IoT

Pour configurer votre bouton AWS IoT :

Allumez votre appareil

1. Retirez le bouton AWS IoT de son emballage, puis appuyez dessus et maintenez-le appuyé pendant 15 secondes, jusqu'à ce qu'une lumière bleue clignotante s'affiche.
2. Le bouton fait office de point d'accès Wi-Fi ; lorsque votre ordinateur recherchera des réseaux Wi-Fi, il en trouvera un nommé Bouton ConfigureMe - XXXX où XXXX est une chaîne de trois caractères générée par le bouton. Utilisez votre ordinateur pour vous connecter au point d'accès du bouton Wi-Fi.
3. La première fois que vous vous connecterez au point d'accès du bouton Wi-Fi, vous serez invité à saisir le mot de passe WPA2-PSK. Saisissez les 8 derniers caractères du numéro de série de l'appareil. Vous trouverez le numéro de série au dos de l'appareil, comme illustré dans l'image suivante :



Copier votre certificat d'appareil sur votre bouton AWS IoT

Pour vous connecter à AWS IoT, vous devez copier votre certificat d'appareil dans le bouton AWS IoT.

1. Dans un navigateur, ouvrez l'adresse <http://192.168.0.1/index.html>.
2. Remplissez le formulaire de configuration.
 1. Saisissez le SSID et le mot de passe Wi-Fi.
 2. Recherchez et sélectionnez votre certificat et votre clé privée.
 3. Trouver votre point de terminaison personnalisé dans la [console AWS IoT](#). Votre point de terminaison aura l'aspect suivant :

ABCDEFG1234567.iot.us-east-1.amazonaws.com

où ABCDEFG1234567 est le sous-domaine et us-east-1 la région.

4. Sur la page Bouton ConfigureMe , tapez le sous-domaine, puis sélectionnez la région qui correspond à la région de votre point de terminaison AWS IoT.
5. Sélectionnez le Conditions générales . Vos paramètres doivent maintenant ressembler à ceux-ci :

Enter the value for any field that you wish to change.

Wi-Fi Configuration:

SSID	Guest
Security	<input checked="" type="checkbox"/> Open Network(No Password)
Password	None (unsecured)

AWS IoT Configuration:

Certificate	<input type="button" value="Choose File"/> MyIoTButtonCert.pem
Private Key	<input type="button" value="Choose File"/> MyIoTButton...ateKey.pem
Endpoint Subdomain	AMUN9F6MTZ77O
Endpoint Region	<input type="button"/>
Final Endpoint	AMUN9F6MTZ77O.iot.us-east-1.amazonaws.com

By clicking this box, you agree to the [AWS IoT Button Terms and Conditions](#).

-
6. Votre bouton doit maintenant se connecter à votre Wi-Fi.

Afficher les messages MQTT du dispositif avec le client MQTT AWS IoT

Vous pouvez utiliser le client MQTT AWS IoT afin de mieux comprendre les messages MQTT envoyés par un dispositif.

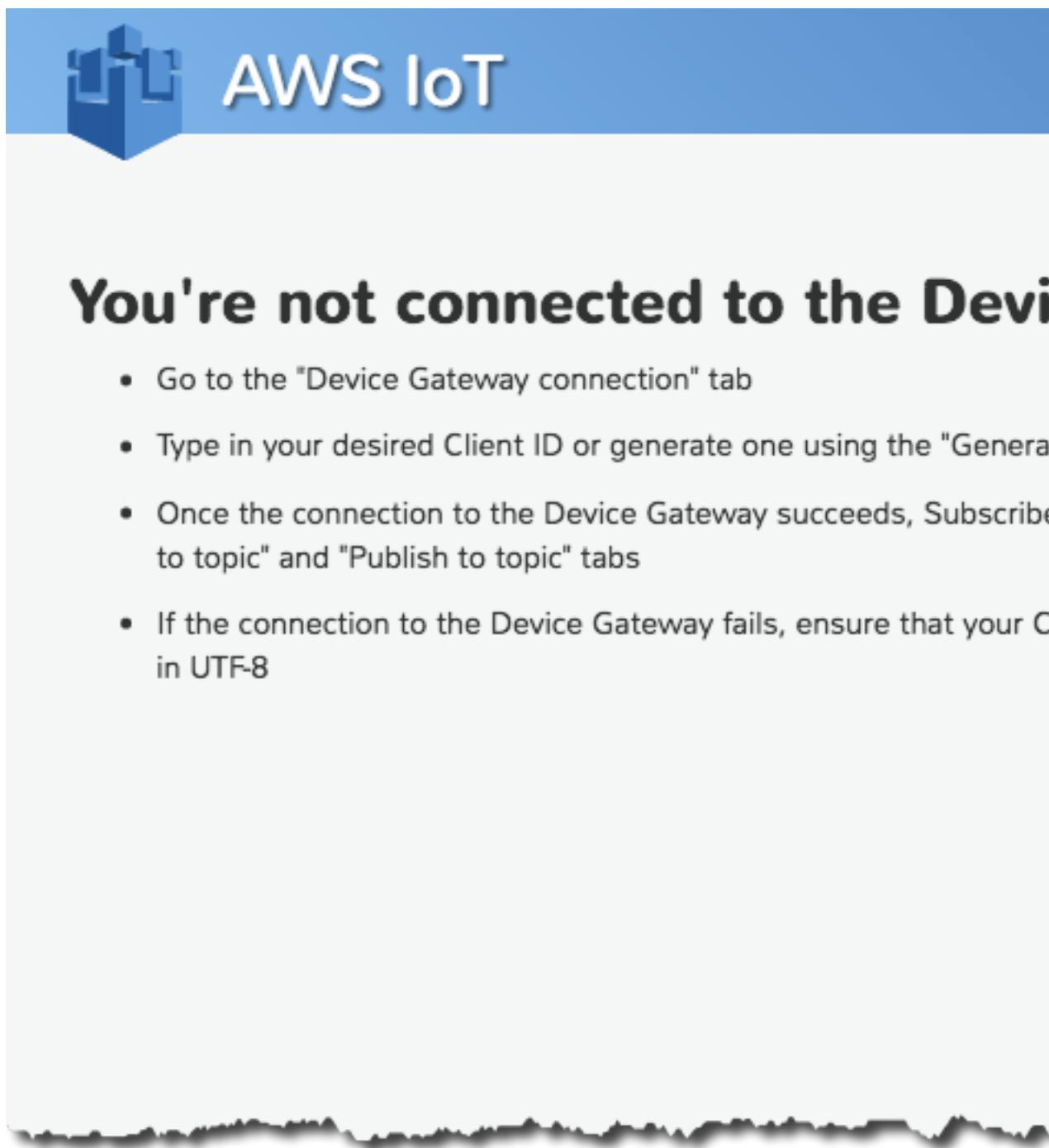
Les appareils publient des messages MQTT dans des rubriques. Vous pouvez utiliser le client MQTT AWS IoT pour vous abonner à ces rubriques afin de voir le contenu de ces messages.

Pour afficher les messages MQTT :

1. Dans la console AWS IoT, sélectionnez Client MQTT.

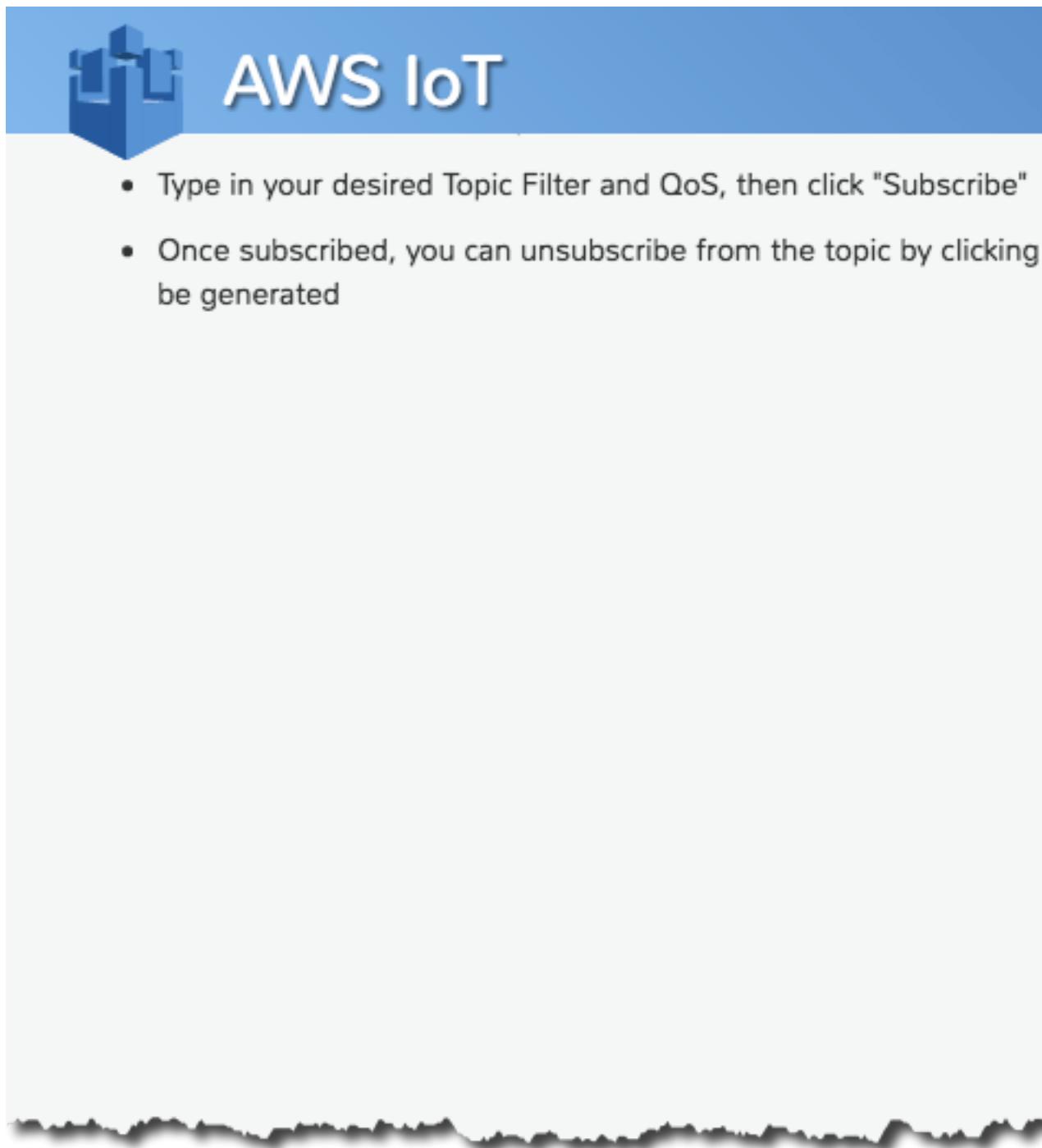


2. Tapez un ID de client ou sélectionnez Générer des ID de client, puis sélectionnez Connexion.



- Go to the "Device Gateway connection" tab
- Type in your desired Client ID or generate one using the "Generate" button
- Once the connection to the Device Gateway succeeds, Subscribe to topic and "Publish to topic" tabs
- If the connection to the Device Gateway fails, ensure that your Client ID and Password are correctly entered and that your password is encoded in UTF-8

-
3. Abonnez-vous à la rubrique dans laquelle votre objet publie. Dans le cas du bouton AWS IoT, vous pouvez vous abonner à `iotbutton/+`. Choisissez S'abonner à la rubrique et dans la rubrique abonnement, tapez `iotbutton/+`, puis sélectionnez S'abonner.



4. Appuyez sur votre bouton AWS IoT, puis affichez le message dans le client MQTT AWS IoT.



Configurer et tester des règles

Le moteur de règles AWS IoT écoute les messages entrants MQTT qui correspondent à une règle. Lorsqu'un message correspondant est reçu, la règle effectue une action avec les données contenues dans le message MQTT (par exemple, écrire les données dans un compartiment Amazon S3, appeler

une fonction Lambda ou envoyer un message à une rubrique Amazon SNS). Au cours de cette étape, vous allez créer et configurer une règle pour envoyer les données reçues à partir d'un dispositif à une rubrique Amazon SNS. Plus précisément, vous allez :

- Créer une rubrique Amazon SNS.
- Vous abonner à la rubrique Amazon SNS à l'aide d'un numéro de téléphone portable.
- Créer une règle qui envoie un message à la rubrique Amazon SNS lorsqu'un message est reçu à partir de votre appareil.
- Tester la règle à l'aide de votre bouton AWS IoT ou d'un client MQTT.

Dans le coin supérieur droit de cette page se trouve une liste déroulante Filtrer la vue . Vous pouvez appuyer sur le bouton AWS IoT pour voir les instructions de test de votre règle avec le bouton AWS IoT ou le client MQTT pour voir les instructions de test de votre règle avec le client MQTT AWS IoT.

Créer une rubrique SNS

Vous allez utiliser la console Amazon SNS pour créer une rubrique Amazon SNS.



Note

Amazon SNS n'est pas disponible dans toutes les régions AWS.

1. Ouvrez <https://console.aws.amazon.com/sns/v2/home>.
2. Dans le volet gauche, sélectionnez Rubriques, puis dans le volet droit, Créez une rubrique.
3. Saisissez un nom de rubrique et un nom complet, puis cliquez sur Créez la rubrique.

Create new topic

A topic name will be used to create a permanent unique identifier called a

Topic name

MyIoTButtonSNSTopic

Display name

IoT Button

4. Notez l'ARN de la rubrique que vous venez de créer.

Topics

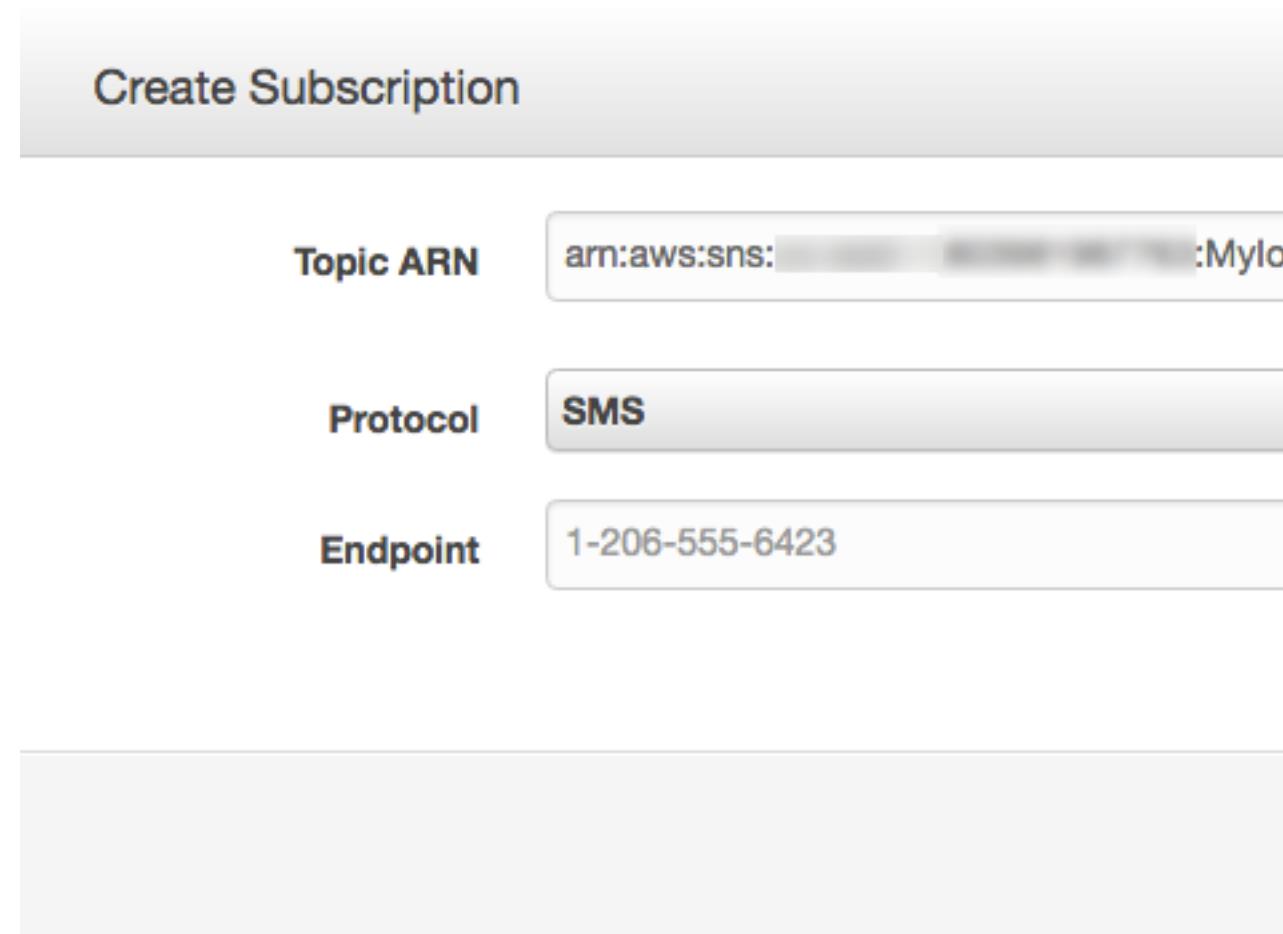
The screenshot shows the AWS SNS Topics console. At the top, there are three buttons: 'Publish to topic' (blue), 'Create new topic' (grey), and 'Actions ▾'. Below these is a search bar labeled 'Filter' containing the text 'MyIoTButtonSNSTopic'. A table follows, with columns 'Name' and 'ARN'. There is one row visible, showing a checkbox next to 'MyIoTButtonSNSTopic' and its ARN starting with 'arn:aws:sns:'.

	Name	ARN
<input type="checkbox"/>	MyIoTButtonSNSTopic	arn:aws:sns: [REDACTED] : [REDACTED]

S'abonner à une rubrique Amazon SNS

Pour recevoir des messages SMS sur votre téléphone portable, vous devez vous abonner à la rubrique Amazon SNS.

1. Dans le menu Actions de la console Amazon SNS, sélectionnez S'abonner à la rubrique.
2. Dans le menu Protocole , sélectionnez SMS.



3. Dans Point de terminaison, tapez le numéro d'un téléphone permettant d'envoyer des SMS, puis cliquez sur Créer un abonnement.



Note

Saisissez le numéro de téléphone à l'aide de chiffres et de tirets uniquement.

Vous recevrez un texto confirmant que vous avez créé l'abonnement.

Créer une règle

Les règles AWS IoT consistent en un filtre de rubrique, une action de règle et, dans la plupart des cas, un rôle IAM. Les messages publiés dans des rubriques qui correspondent au filtre de rubrique déclenchent la règle. L'action de règle définit les mesures à prendre lorsque la règle est déclenchée. Le rôle IAM contient une ou plusieurs stratégies IAM qui déterminent à quels services AWS la règle a accès. Vous pouvez créer plusieurs règles qui écoutent une seule rubrique. De même, vous pouvez créer une seule règle qui est déclenchée par plusieurs rubriques. Le moteur de règles AWS IoT traite en continu les messages publiés dans des rubriques qui correspondent aux filtres de rubrique définis dans les règles.

Dans cet exemple, vous allez créer une règle qui utilise Amazon SNS pour envoyer une notification par SMS à un numéro de téléphone portable.

1. Dans la console AWS IoT, sélectionnez Créer une règle.

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the word "Resources" is displayed in large, dark font. In the top right corner of the main area, there's a button labeled "Close create panel" with a small "X" icon. In the center of the main area, there are two buttons: "Create a thing" (with a robot icon) and "Create a rule" (with a network icon). The "Create a rule" button is highlighted with a thick blue border. Below these buttons, there's a section titled "Create a rule" with a sub-instruction: "Create a rule to evaluate inbound messages published into AW endpoint such as a DynamoDB table." Further down, there's a form for creating a rule, with fields for "Name" (containing "mySNSRule") and "Description" (containing "A simple SNS rule").

2. Sur la page Crer une rgle , dans Nom, tapez un nom pour votre rgle.
3. Dans Description, tapez une description pour la rgle.
4. Dans Attribut, type *. Cela spcifie que vous souhaitez envoyer le message MQTT entier qui a dclench la rgle.

5. Le moteur de règles utilise le filtre de rubriques pour déterminer quelles règles déclencher lors de la réception d'un message MQTT. Dans Filtre de rubriques, type `iotbutton/votre bouton-DSN`. Si vous n'utilisez pas un bouton AWS IoT, tapez `my/topic`.



Note

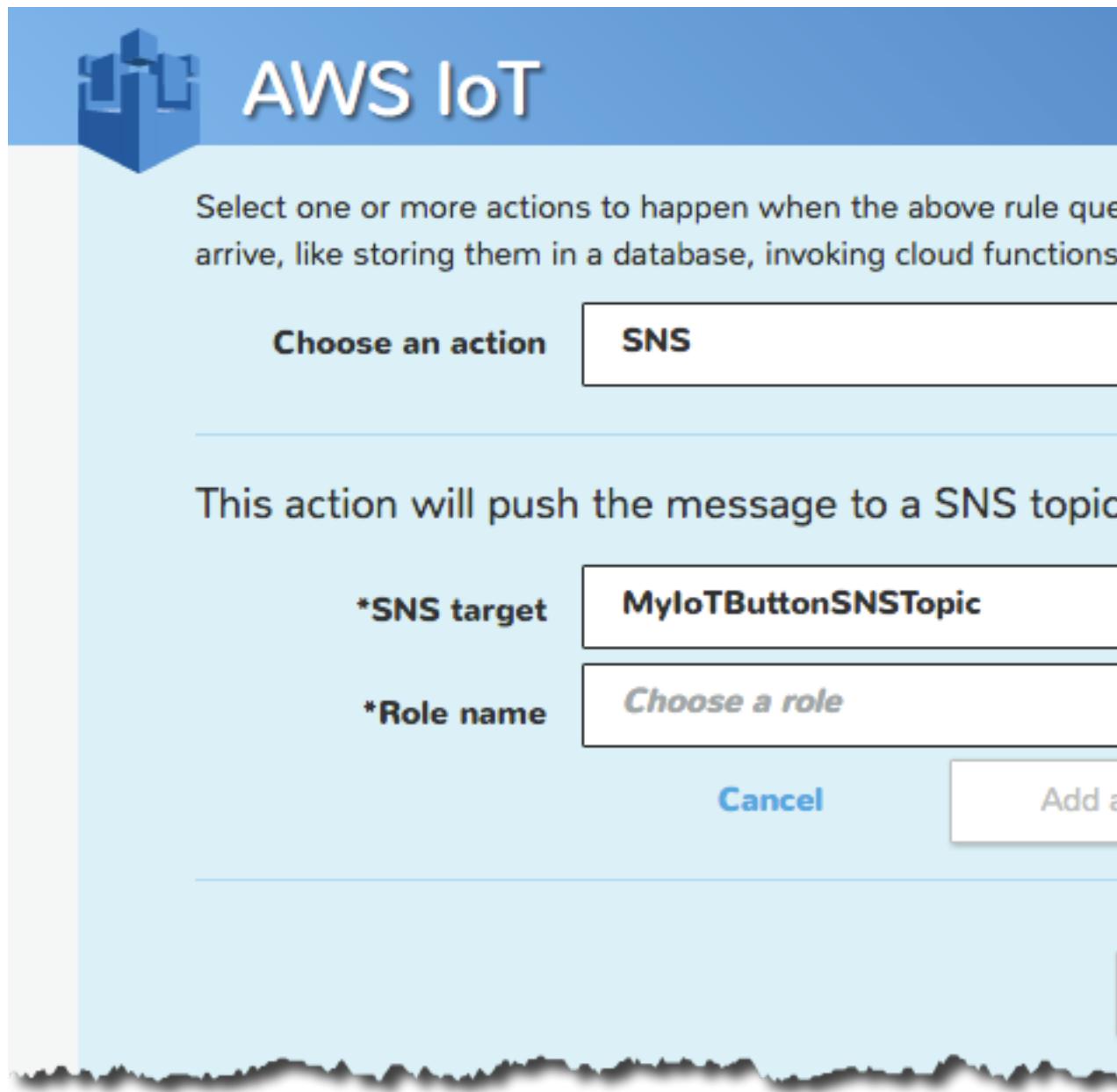
Vous trouverez le DSN de votre bouton imprimé au bas de l'appareil.

6. Quitter Condition blank.

The screenshot shows a note about finding the DSN on the device, then a 'Condition blank' section with a large input field containing the rule query statement. The query statement includes a SELECT clause, an attribute of '*', a topic filter of 'my/topic', and a condition of 'e.g. temperature > 75'.

Rule query statement	SELECT * FROM 'my/topic'
Attribute	*
Topic filter	my/topic
Condition	e.g. temperature > 75

7. Dans le menu Choisir une action , sélectionnez Envoyer un message sous la forme de notification push (SNS).
8. Dans la liste déroulante Cible SNS sélectionnez la rubrique Amazon SNS que vous avez créé précédemment.



9. Vous devez maintenant donner à AWS IoT l'autorisation de publier dans la rubrique Amazon SNS pour votre compte lorsque la règle est déclenchée. Cliquez sur le lien [Créer un rôle](#). Cela ouvre une page web dans la console IAM.
10. Acceptez les valeurs par défaut, puis sélectionnez Autoriser.

AWS IoT is requesting permission to use resources in your account

Click allow to give AWS IoT write access to resources in your account.

▼ Hide Details

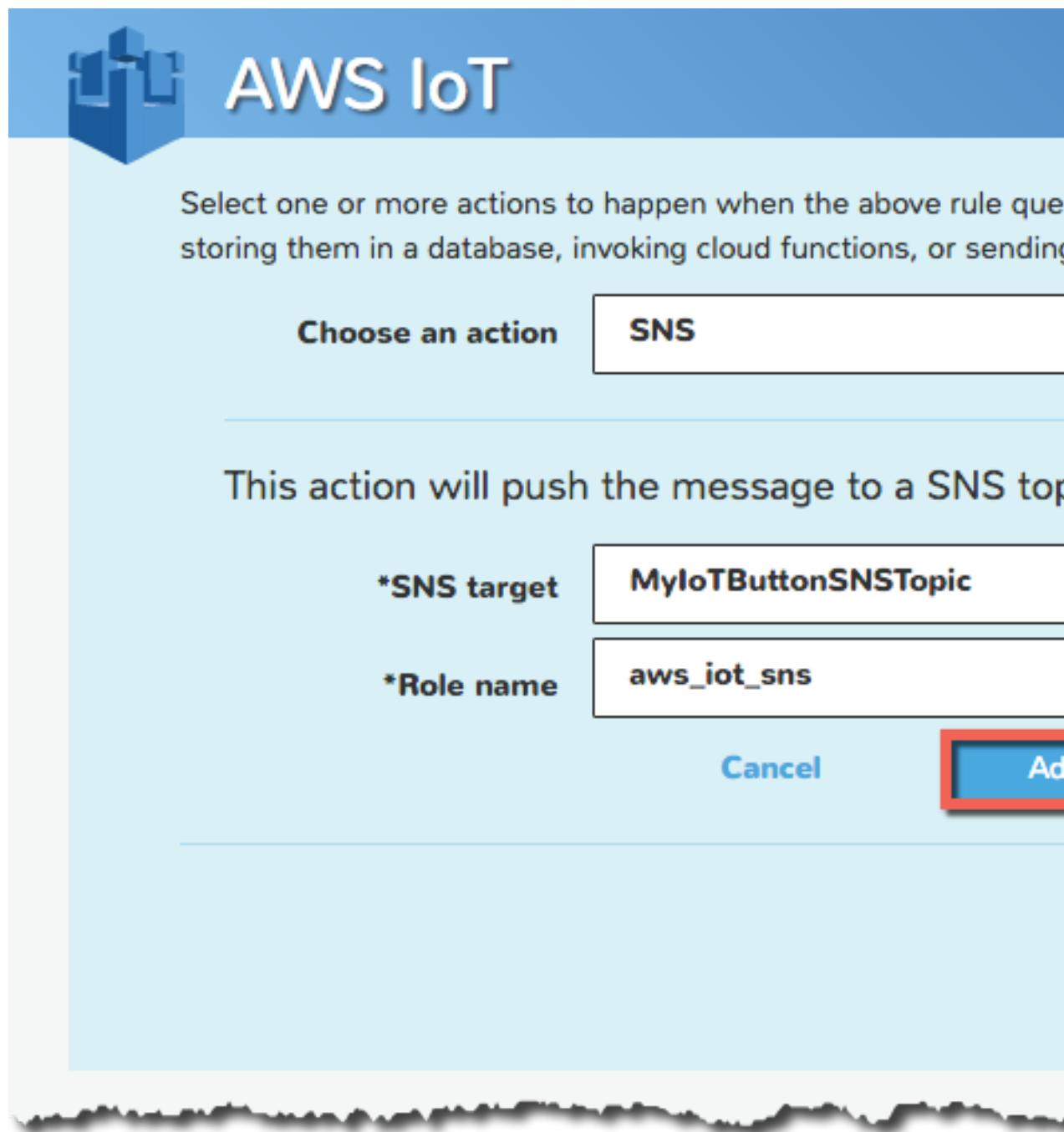
Role Summary 

Provides write access to AWS Services and Resources

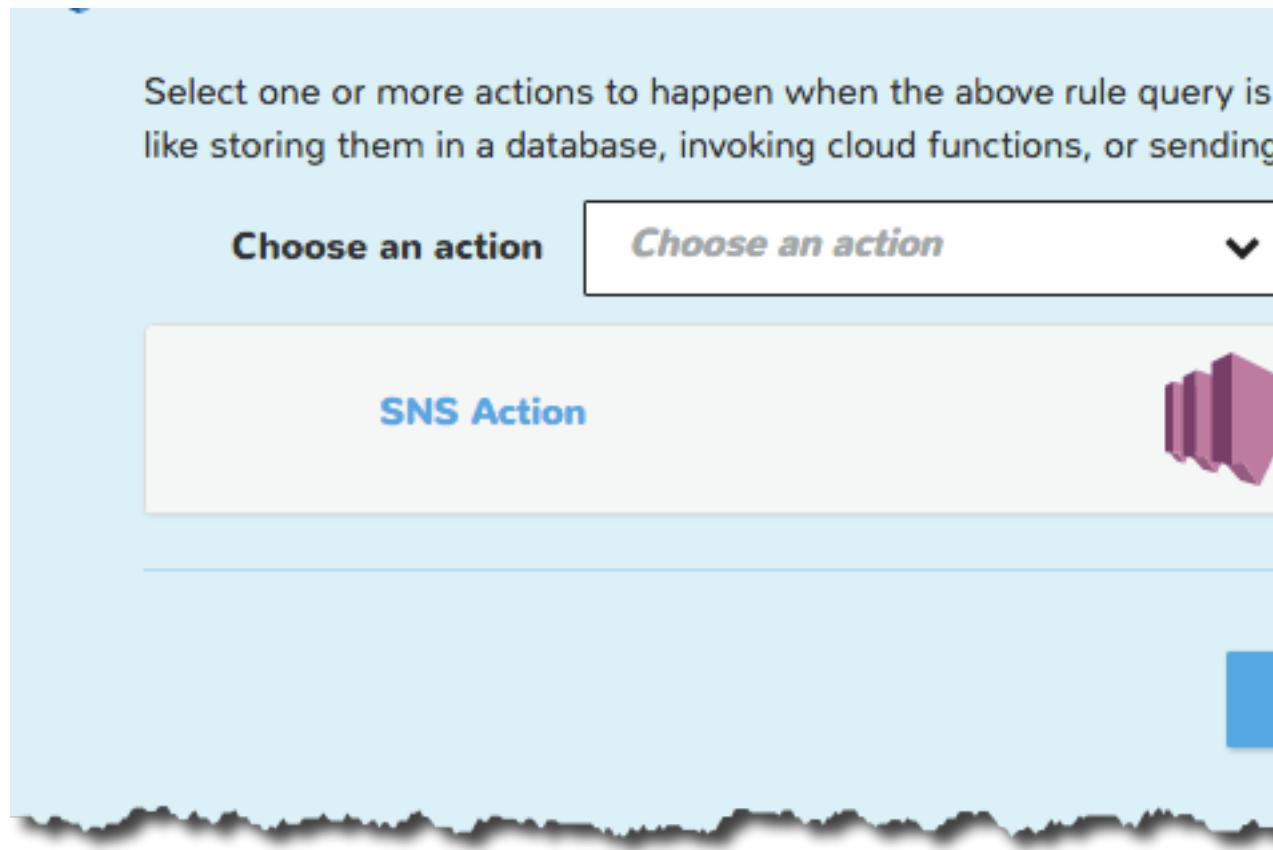
aws_iot_sns

Create a new Role Policy

-
11. Choisissez Ajouter une action. Cela ajoute l'action à la règle.



12. Choisissez Créer pour créer la règle.



Pour plus d'informations sur la création de règles, consultez [Règles AWS IoT](#).

Tester la règle Amazon SNS

Vous pouvez tester votre règle en appuyant sur un bouton AWS IoT ou avec le client MQTT AWS IoT.

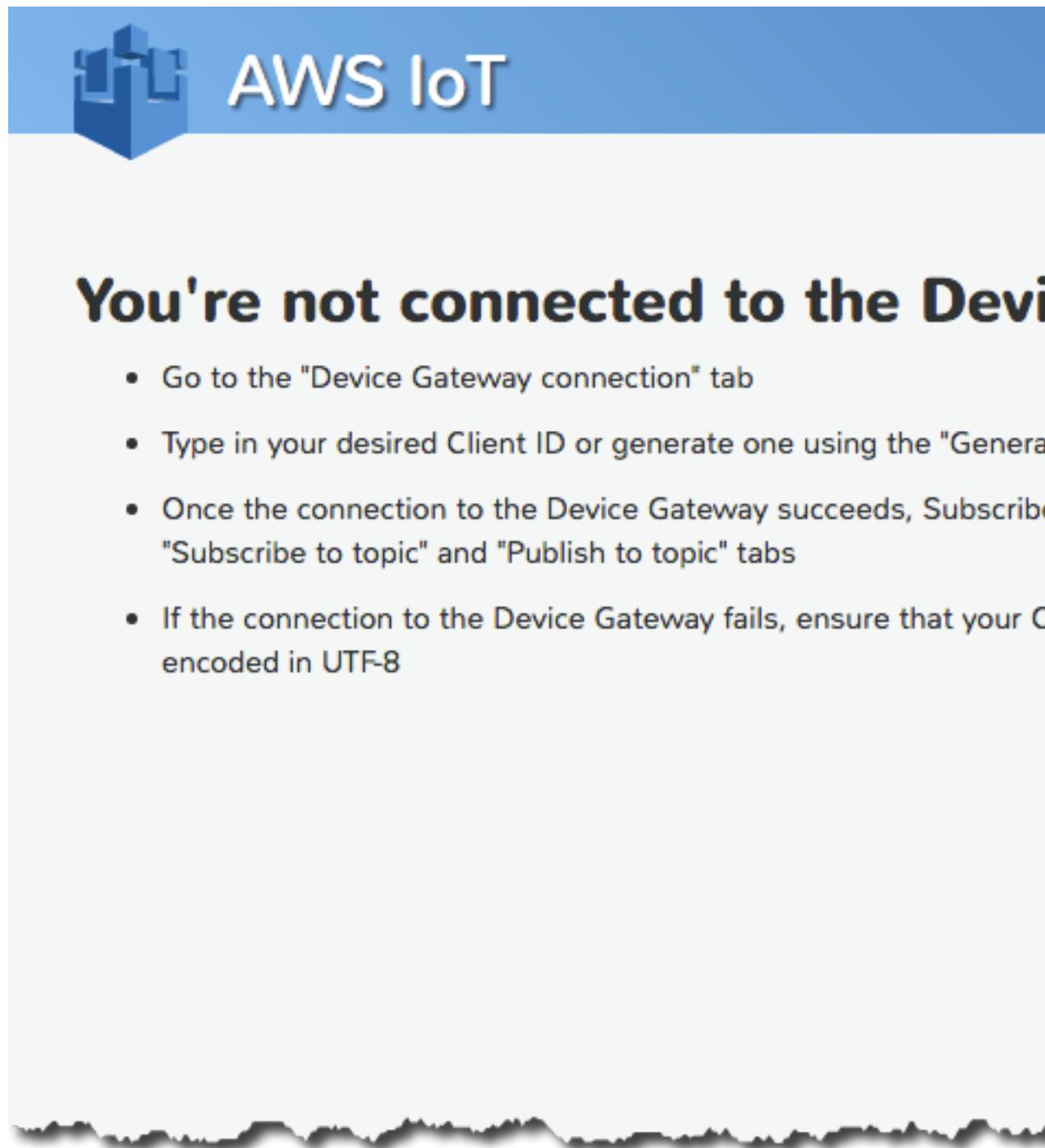
Bouton AWS IoT

Appuyez sur le bouton. Vous devriez recevoir un texto indiquant les frais facturés sur votre dispositif.

Client MQTT AWS IoT

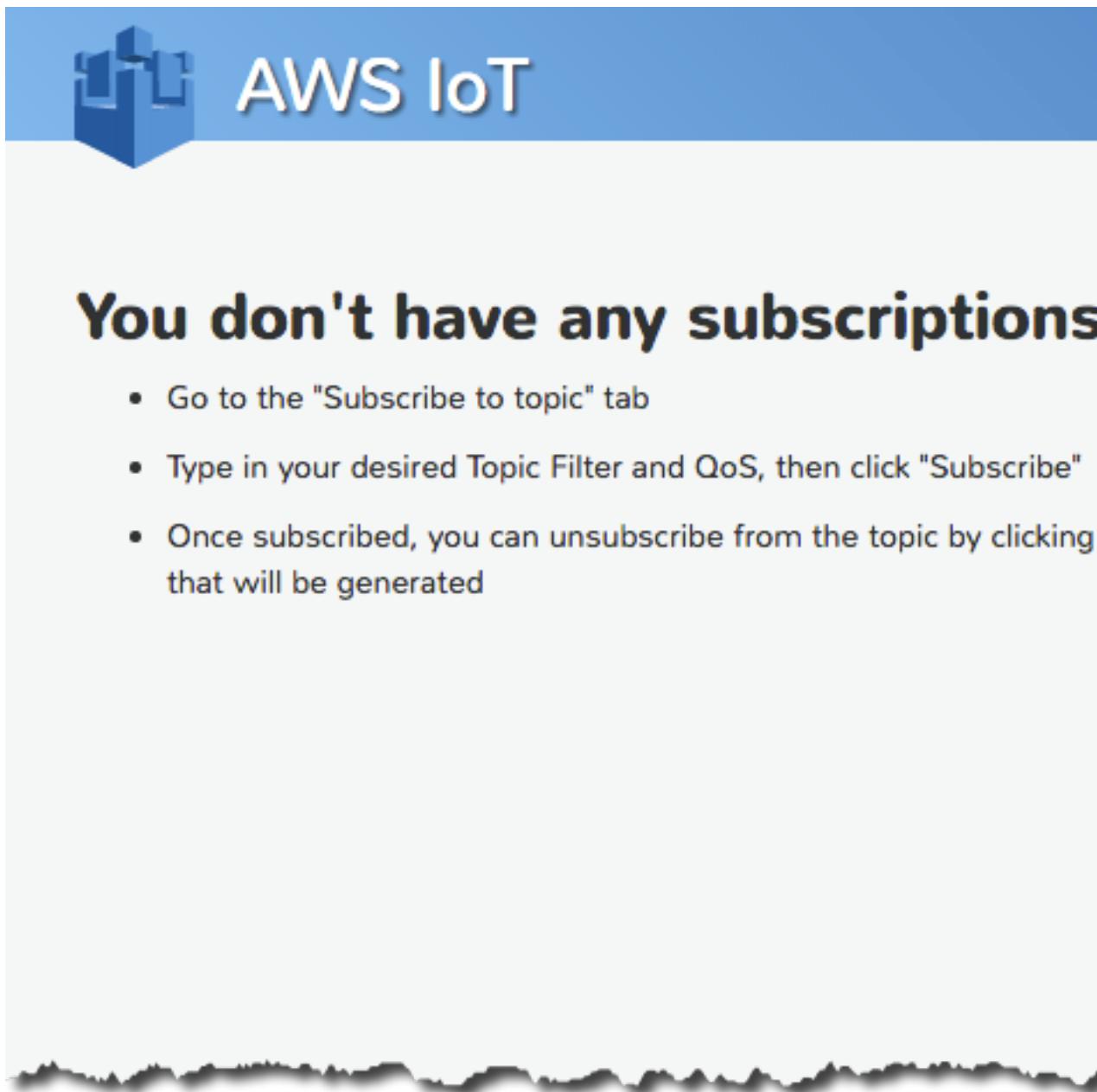
Pour tester votre règle avec le client MQTT AWS IoT :

1. Dans la [console AWS IoT](#), sélectionnez client MQTT.
2. Choisissez Générer des ID de client, puis sélectionnez Connexion.



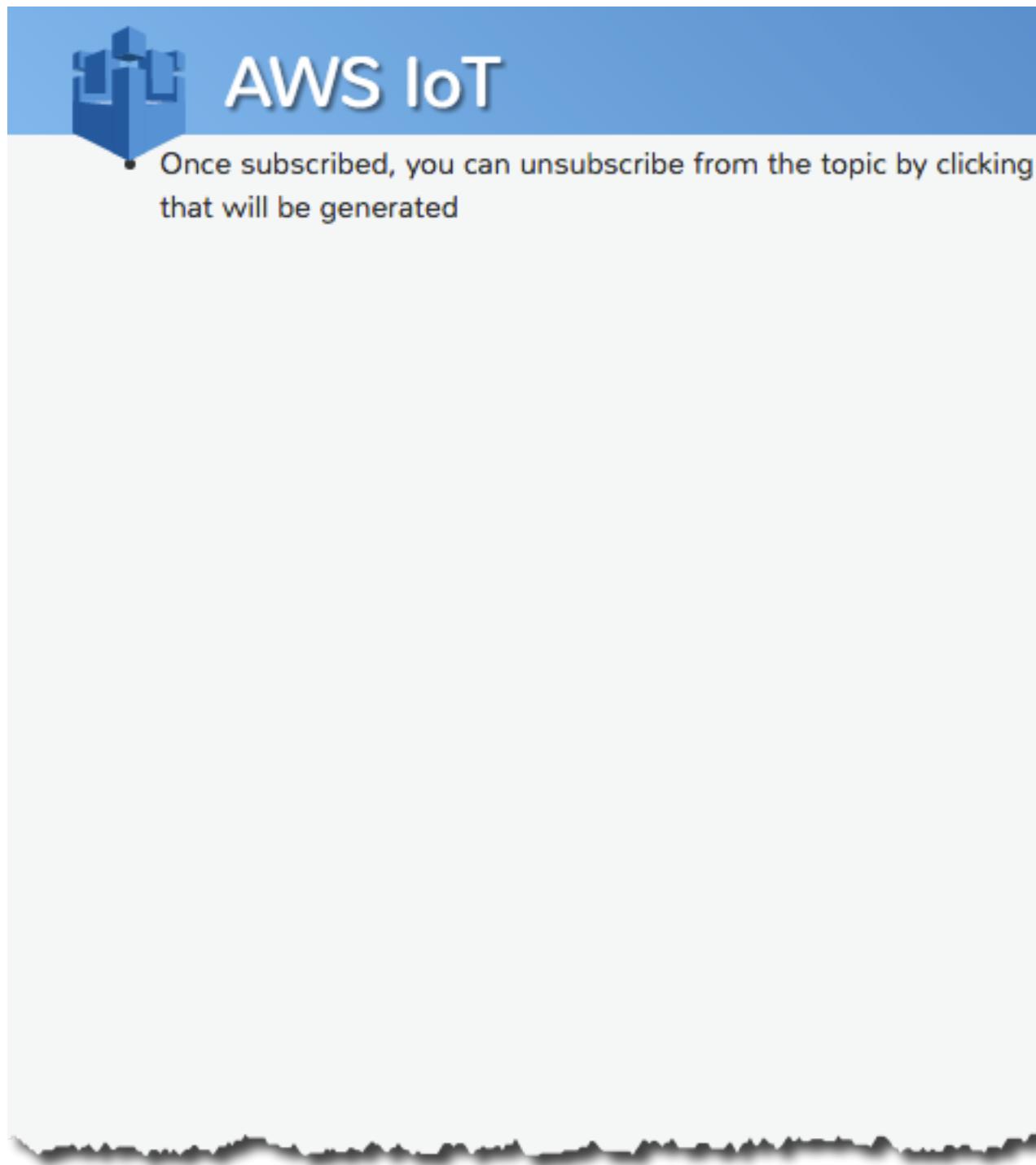
- Go to the "Device Gateway connection" tab
- Type in your desired Client ID or generate one using the "Generate" button
- Once the connection to the Device Gateway succeeds, Subscribes to the "Subscribe to topic" and "Publish to topic" tabs
- If the connection to the Device Gateway fails, ensure that your Client ID is encoded in UTF-8

3. Sur la page du client MQTT, sélectionnez Publier dans la rubrique.



4. Dans le champ Publier une rubrique , tapez `my/topic`.
5. Dans Charge utile, tapez le code JSON suivant :

```
{ "message": "Bonjour à tous depuis AWS IoT !" }
```



6. Choisissez Publier. Vous devriez recevoir un message Amazon SNS sur votre téléphone portable.

Etapes suivantes

Vous trouverez plus d'informations sur les règles AWS IoT grâce aux liens suivants :

- Didacticiels concernant les règles AWS IoT (p. 53)
- Règles AWS IoT (p. 121)

Didacticiels concernant les règles AWS IoT

Ce guide contient des didacticiels qui vous guident dans la création et les tests de règles AWS IoT. Si vous n'avez pas terminé le [Didacticiel de démarrage AWS IoT \(p. 18\)](#), nous vous conseillons de le faire avant toute chose. Celui-ci explique comment créer un compte AWS et connecter votre dispositif à AWS IoT.

Une règle AWS IoT se compose d'une instruction SQL SELECT, un filtre de rubriques et une action de règle. Les dispositifs envoient des informations à AWS IoT en publiant des messages dans des rubriques MQTT. L'instruction SQL SELECT permet d'extraire des données d'un message MQTT entrant. Le filtre de la rubrique d'une règle AWS IoT spécifie une ou plusieurs rubriques MQTT. La règle est déclenchée lorsqu'un message MQTT est reçu dans une rubrique qui correspond au filtre de rubriques. Les actions de règle permettent de prendre les informations extraites d'un message MQTT et de les envoyer à un autre service AWS. Les actions de règle sont définies pour les services AWS, comme Amazon DynamoDB, AWS Lambda, Amazon SNS et Amazon S3. En utilisant une règle Lambda, vous pouvez appeler d'autres AWS ou des services Web tiers. Pour obtenir la liste complète des actions de règle, consultez [Actions de règle AWS IoT \(p. 127\)](#).

Les didacticiels dans ce guide supposent que vous utilisez le bouton AWS IoT et que vous utiliserez `iotbutton/+` comme filtre de rubriques dans les règles. Si vous n'avez pas de bouton AWS IoT que vous souhaitez en acheter un, consultez la rubrique [acheter un bouton AWS IoT](#).

Le bouton AWS IoT envoie une charge utile JSON semblable à ceci :

```
{ "serialNumber" : "ABCDEFG12345", "batteryVoltage" : "2000mV", "clickType" :  
"SINGLE" }
```

Vous pouvez émuler le bouton AWS IoT en utilisant un client MQTT, tel que le client MQTT AWS IoT dans la [console AWS IoT](#). Pour émuler le bouton AWS IoT, publiez un message similaire sur la rubrique `iotbutton/ABCDEFG12345`. Le nombre après la / est arbitraire. Il sera utilisé comme numéro de série du bouton.

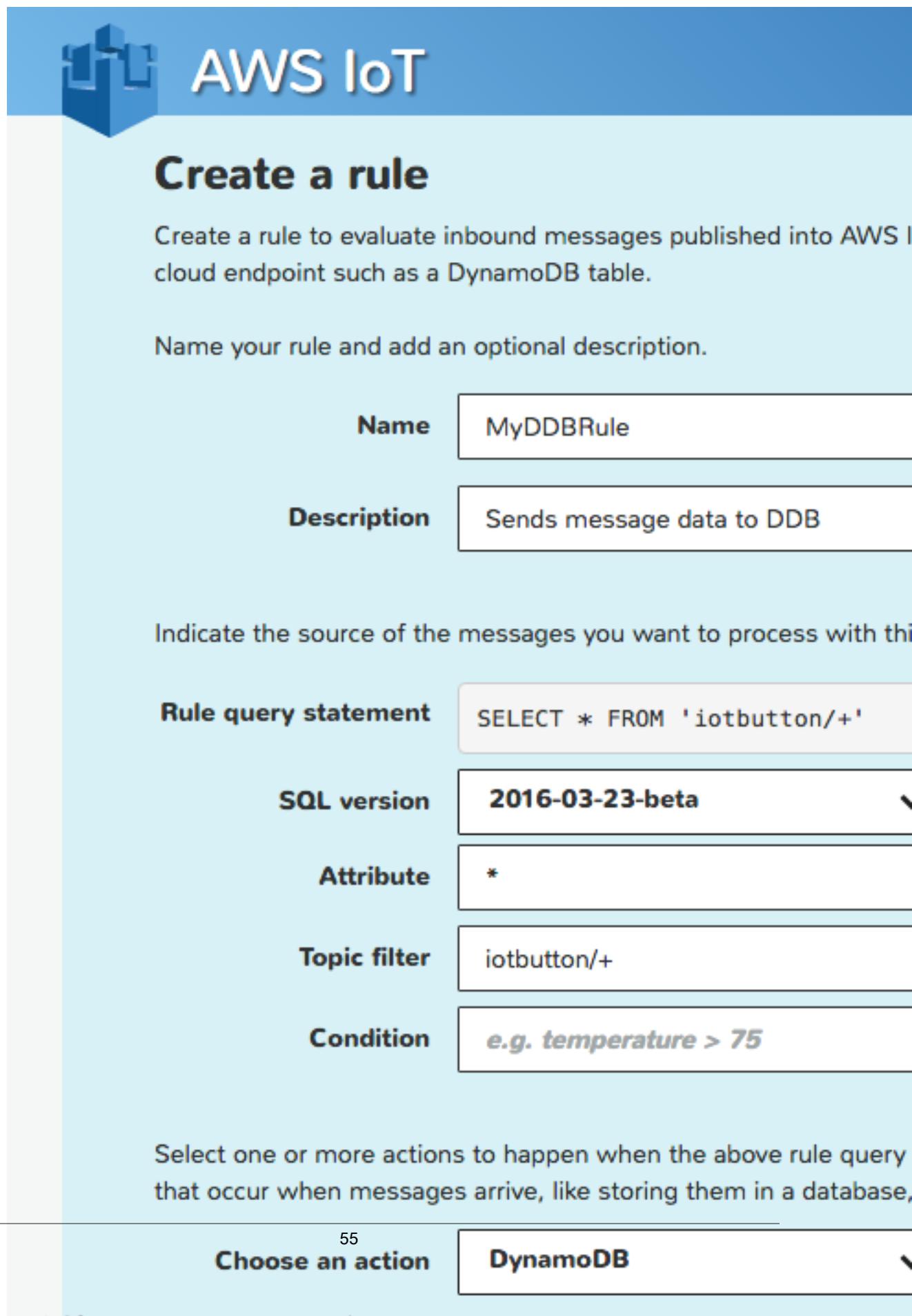
Vous pouvez utiliser votre propre dispositif, mais vous devez savoir dans quelle rubrique MQTT votre dispositif publie pour pouvoir le spécifier comme filtre de rubrique dans la règle.

Pour plus d'informations, consultez [Règles AWS IoT \(p. 121\)](#).

Création d'une règle DynamoDB

Les règles DynamoDB permettent de prendre des informations d'un message MQTT entrant et de les écrire dans une table DynamoDB.

Pour créer une règle DynamoDB :



The image shows the AWS IoT 'Create a rule' interface. At the top, there's a blue header bar with the AWS IoT logo and the title 'Create a rule'. Below the header, there's a large text area with the heading 'Create a rule' and a sub-instruction: 'Create a rule to evaluate inbound messages published into AWS IoT cloud endpoint such as a DynamoDB table.' Further down, there's a section for naming the rule with the placeholder 'Name' and 'MyDDBRule'. There's also a 'Description' field with the placeholder 'Sends message data to DDB'. On the left side, there's a sidebar with sections for 'Rule query statement', 'SQL version', 'Attribute', 'Topic filter', and 'Condition'. The 'Rule query statement' section contains the SQL query 'SELECT * FROM 'iotbutton/+'' and a dropdown menu set to '2016-03-23-beta'. The 'Attribute' field has an asterisk (*) placeholder. The 'Topic filter' field contains 'iotbutton/+'. The 'Condition' field contains the placeholder 'e.g. temperature > 75'. At the bottom, there's a section for selecting actions with the heading 'Choose an action' and a button labeled 'DynamoDB'.

Create a rule

Create a rule to evaluate inbound messages published into AWS IoT cloud endpoint such as a DynamoDB table.

Name your rule and add an optional description.

Name	MyDDBRule
Description	Sends message data to DDB

Indicate the source of the messages you want to process with this rule.

Rule query statement	SELECT * FROM 'iotbutton/+'
SQL version	2016-03-23-beta
Attribute	*
Topic filter	iotbutton/+
Condition	e.g. temperature > 75

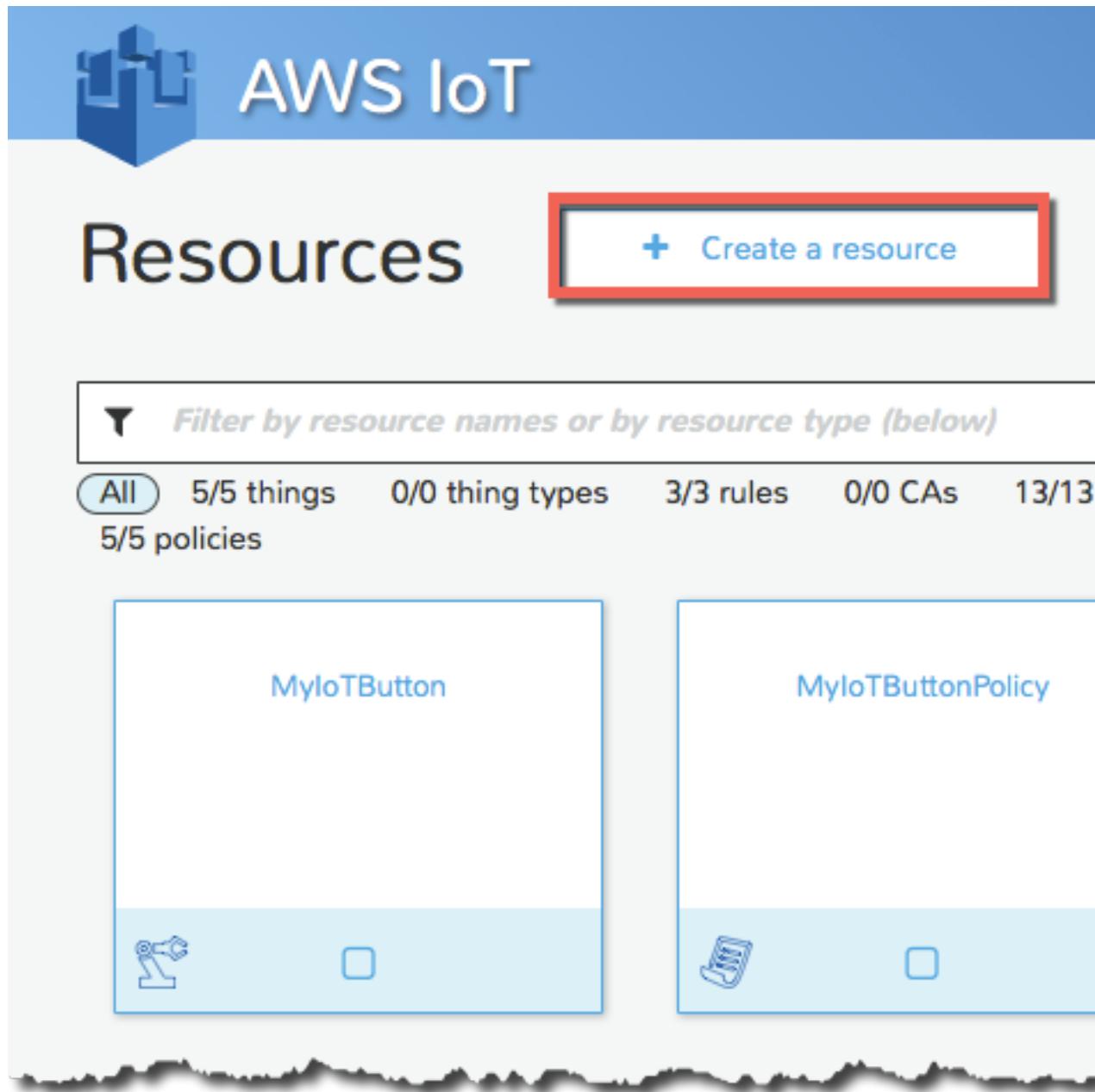
Select one or more actions to happen when the above rule query that occur when messages arrive, like storing them in a database,

55

Choose an action

DynamoDB

1. Dans la console AWS IoT, sélectionnez Créer une ressource.



2. Choisissez Créer une règle.

The screenshot shows the AWS IoT Resources page. At the top, there is a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the main title "Resources" is displayed in large, bold, dark blue text. In the top right corner of the main area, there is a button labeled "Close create panel" with a small "X" icon. Below the title, there are two large, light blue rectangular boxes. The first box contains a wrench icon and the text "Create a thing". The second box contains a wrench and screwdriver icon and the text "Create a thing type". Underneath these boxes, there is a search/filter bar with a magnifying glass icon and the placeholder text "Filter by resource names or by resource type (below)". Below the search bar, there are several status indicators: "All" (highlighted in a blue oval), "5/5 things", "0/0 thing types", "3/3 rules", "0/0 CAs", and "13/13 policies". To the right of these indicators, there are two resource cards. The first card, titled "MyIoTButton", contains a wrench icon and a blue "Create" button. The second card, titled "MyIoTButtonPolicy", contains a smartphone icon and a blue "Create" button.

3. Sur la page Crée une règle :

Tapez un nom de règle et une description dans Nom and Description.

Le Instruction de requête de règle sera renseigné automatiquement lorsque vous saisissez des données dans les champs en dessous.

Dans Attribut, type *. Cela détermine quelle partie du message entrant sera envoyée à l'action de règle. Utiliser * envoie l'intégralité du message.

Dans Filtre de rubriques, type `iotbutton/+`. Si vous utilisez un dispositif différent, tapez un filtre de rubriques qui correspondre à la rubrique MQTT dans laquelle votre dispositif effectue les publications.

De Choisir une action, sélectionnez Insérer un message dans une table de base de données (DynamoDB).

4. La page Créer une règle s'ouvre. En regard de la liste déroulante Nom de la table , sélectionnez Créer une ressource. Cela ouvre la console DynamoDB où vous pouvez créer une table DynamoDB.

The screenshot shows the AWS IoT Rule Configuration interface. At the top, there's a blue header bar with the AWS IoT logo and the title "AWS IoT". Below the header, a large blue sidebar on the left contains the text "Indicate the source of the messages you want to process with". To the right of the sidebar, there are five configuration fields:

- Rule query statement:** SELECT * FROM 'iotbutton/+'
- SQL version:** 2016-03-23-beta
- Attribute:** *
- Topic filter:** iotbutton/+
- Condition:** e.g. temperature > 75

Below these fields, another section titled "Select one or more actions to happen when the above rule que..." is visible. This section includes a "Choose an action" dropdown which has "DynamoDB" selected. A note below states "This action will insert the message into a DynamoDB table".

At the bottom of the configuration area, there are two buttons: "Cancel" and "Add action".

5. Choisissez Créer une table.

The screenshot shows the AWS DynamoDB console. On the left, a sidebar menu lists 'Dashboard', 'Tables', and 'Reserved capacity', with 'Tables' being the active tab. The main content area has a title 'Create table' and a large blue button labeled 'Create table'. Below this, there's a section titled 'Recent alerts' stating 'No CloudWatch alarms have been triggered'. A summary table follows, showing 'Total capacity for US East (N. Virginia)':

	Total capacity
Provisioned read capacity	401
Provisioned write capacity	342
Reserved read capacity	0
Reserved write capacity	0

Finally, a 'Service health' section shows a green checkmark next to 'Amazon DynamoDB (N. Virginia)' with the status 'Current Status'.

6. Dans Nom de la table, tapez un nom pour la table. Les clés de partition et de tri sont combinées pour créer une clé primaire pour votre table DynamoDB. Dans Clé de partition, tapez `SerialNumber`, puis sélectionnez Ajouter une clé de tri. Dans Clé de tri, tapez `ClickType`. La clé de partition et la clé de tri doivent être de type String.

Votre écran doit maintenant avoir l'aspect suivant :

Create DynamoDB table

DynamoDB is a schema-less database that primary key is made up of one or two attributes sort data within each partition.

Table name*

Primary key* Partition key

Add sort key

Table settings

Default settings provide the fastest way to get started. You can change these settings now or after your table has been created.

Use default settings

- No secondary indexes
- Provisioned throughput
- Basic alarm monitoring ("dynamodb:TableSizeBytes")

Additional charges may apply if you exceed the AWS Free Usage Tier. Advanced alarm settings are available in the CloudWatch Metrics section.

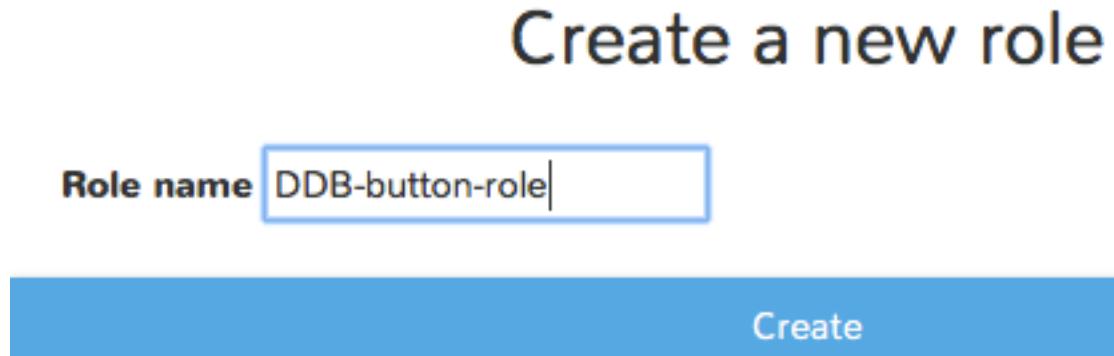
7. Choisissez Créer. La création de la table DynamoDB prend quelques secondes. Fermez l'onglet de navigateur qui contient la console DynamoDB. Si vous ne fermez pas l'onglet, votre table DynamoDB n'apparaît pas dans la liste déroulante Nom de la table de la console AWS IoT. Dans la console AWS IoT, sélectionnez votre nouvelle table.
8. Dans Valeur de la clé de hachage, tapez `${serialNumber}` . Cela indique à la règle qu'elle doit prendre la valeur de l'attribut `serialNumber` du message MQTT et l'écrire dans la colonne `SerialNumber` de la table DynamoDB. Dans Valeur de clé de plage, tapez `${clickType}` . Cela écrit la valeur de l'attribut `clickType` dans la colonne `ClickType` . Laissez le champ charge utile vide. Par défaut, l'ensemble du message sera écrit dans une colonne de la table nommée Charge utile. Sélectionner Créer un rôle.

The screenshot shows the AWS IoT Rule Editor interface. At the top, there's a blue header bar with the AWS IoT logo and the text "AWS IoT". Below the header, a navigation bar has "Choose an action" on the left and "DynamoDB" on the right. A horizontal line separates this from the main content area.

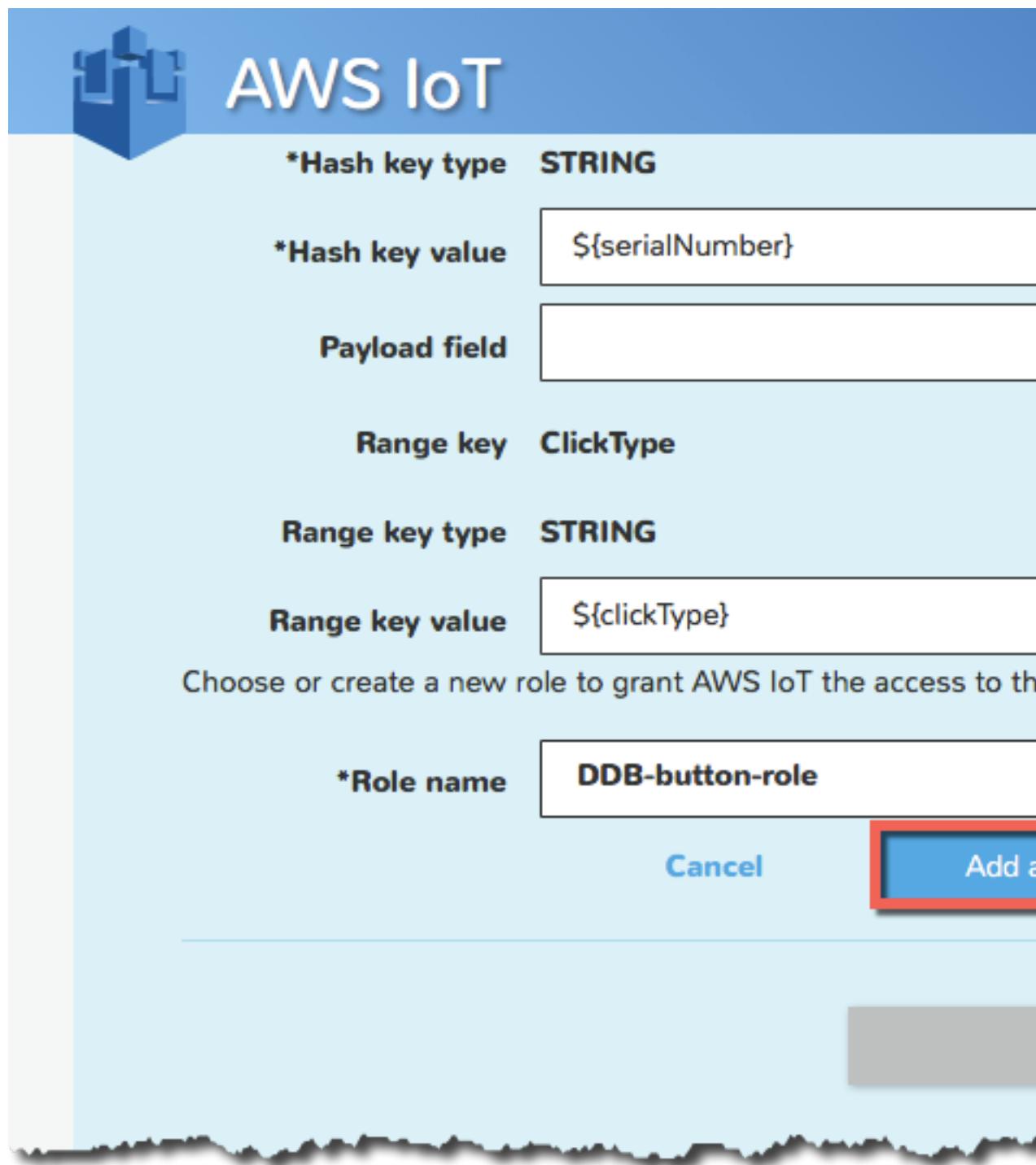
The main content area contains the following information:

- This will insert the message into a DynamoDB table:** This is a descriptive text above the configuration fields.
- *Table name:** **IoTButtonTable** (highlighted in red)
- *Hash key:** **SerialNumber**
- *Hash key type:** **STRING**
- *Hash key value:** **\${serialNumber}**
- Payload field:** (empty input field)
- Range key:** **clickType**
- Range key type:** **STRING**
- Range key value:** **\${clickType}**
- Choose or create a new role to grant AWS IoT the access to the table:** This is a descriptive text below the role selection field.
- *Role name:** **Choose a role** (highlighted in red)
- Cancel** (blue button)
- Add a condition** (button with a plus sign)

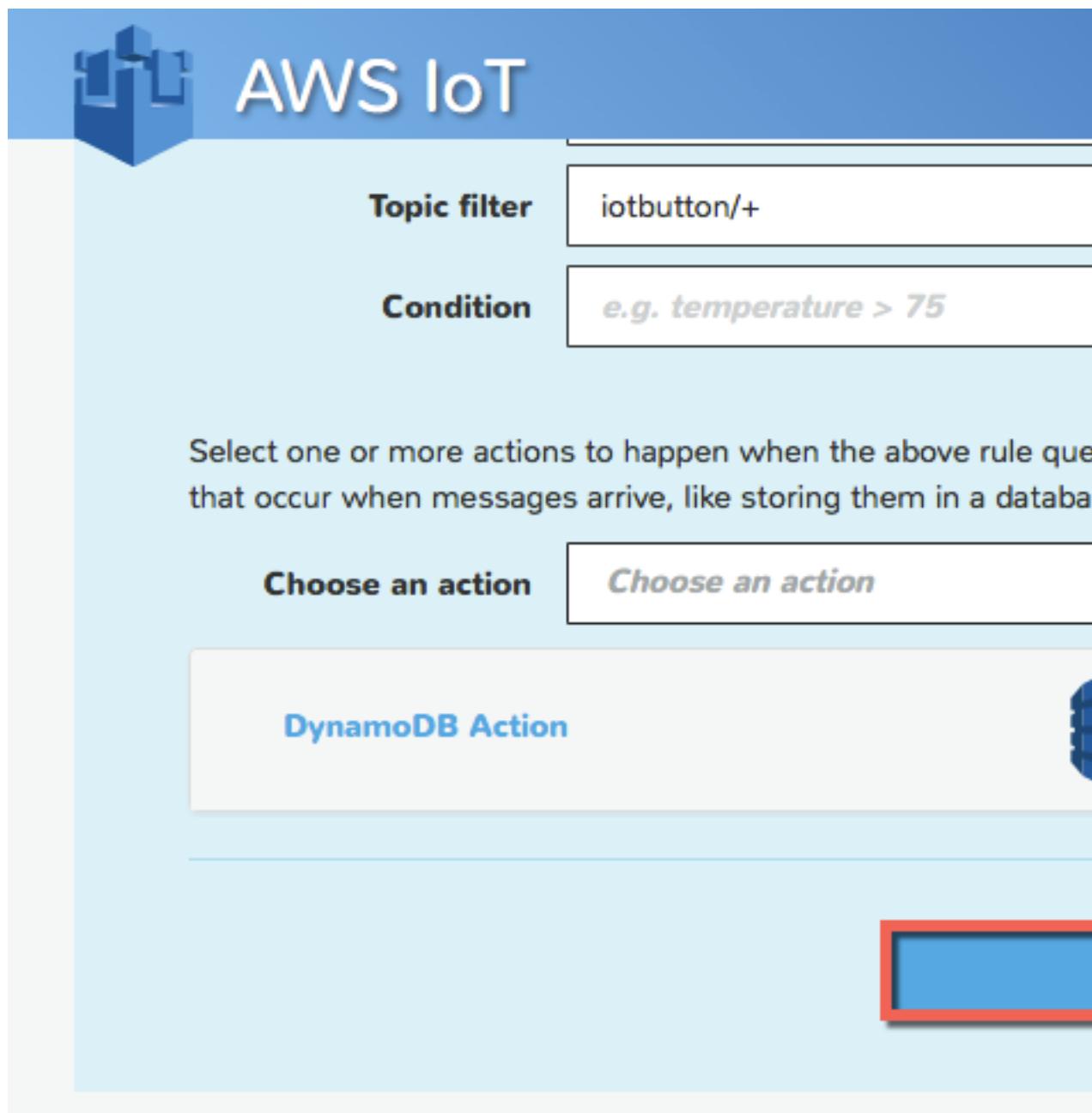
9. Tapez un nom de rôle unique dans la boîte de dialogue Créeer un rôle et cliquez sur le bouton Créeer .



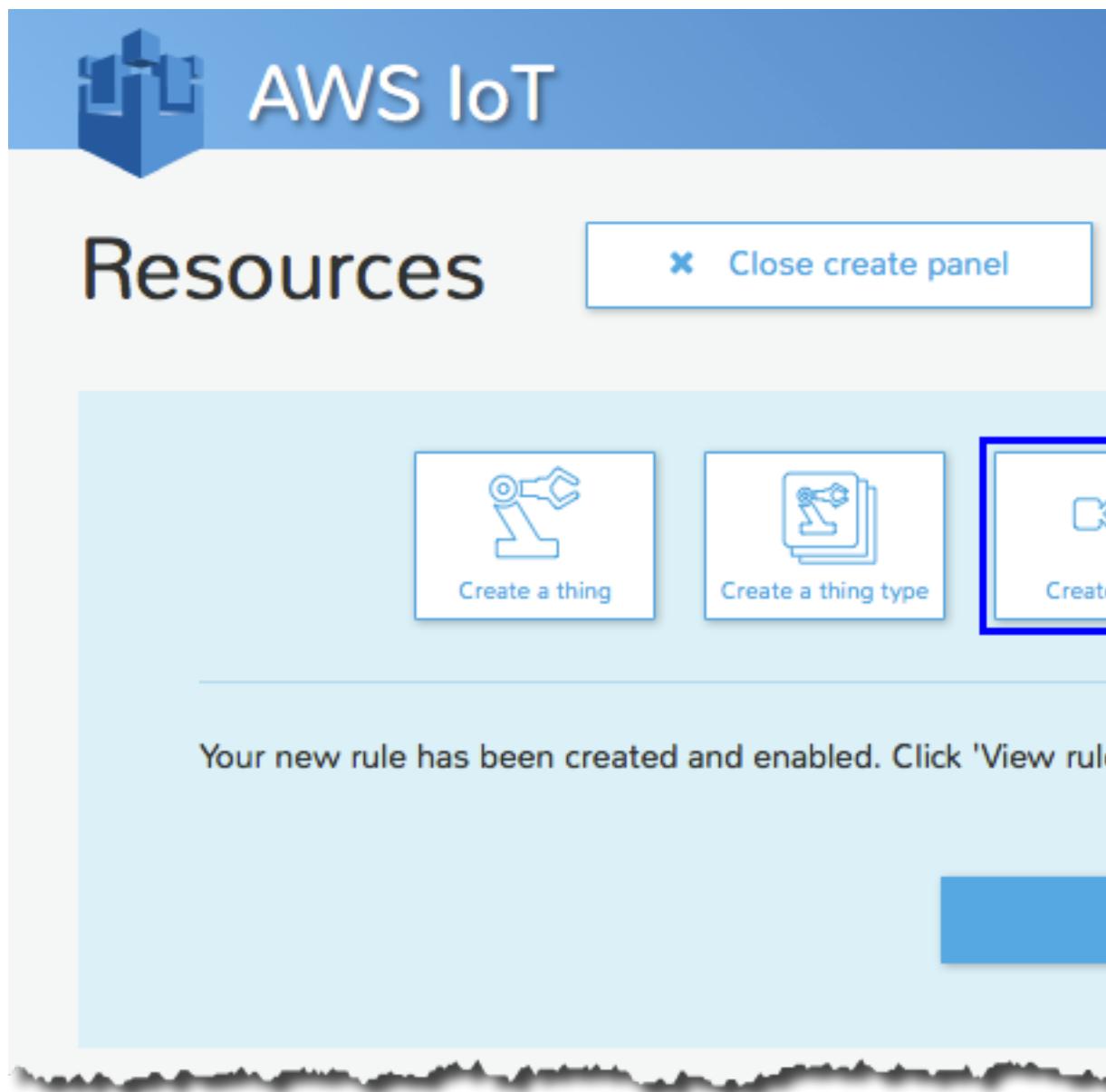
10. Choisissez Ajouter une action pour ajouter l'action à la règle.



11. Choisissez Créer pour créer la règle.



12. Un message de confirmation s'affiche sur l'écran indiquant que la règle a été créée.



13. Testez la règle en cliquant sur votre bouton AWS IoT configuré ou en utilisant un client MQTT pour publier un message dans une rubrique correspondant au filtre de rubriques de votre règle.

Création d'une règle Lambda

Vous pouvez définir une règle qui appelle une fonction Lambda en transmettant des données du message MQTT qui a déclenché la règle. Cela permet de traiter le message entrant, puis d'appeler un autre service AWS ou tiers.

Ce didacticiel suppose que vous avez terminé le [Didacticiel de démarrage AWS IoT \(p. 18\)](#) dans lequel vous avez créé une rubrique Amazon SNS à laquelle vous vous êtes abonné à l'aide de votre numéro de téléphone portable. Vous allez créer une fonction Lambda qui publie un message dans la rubrique Amazon SNS que vous avez créée dans le [Didacticiel de démarrage AWS IoT \(p. 18\)](#). Vous

allez également créer une règle Lambda qui appelle la fonction Lambda en transmettant des données depuis le message MQTT qui a déclenché la règle.

Ce didacticiel suppose également que vous utilisez un bouton AWS IoT pour déclencher la règle Lambda. Si vous n'avez pas de bouton AWS IoT, vous pouvez en acheter un [ici](#) ou utiliser un client MQTT pour envoyer un message MQTT qui va déclencher la règle.

Créer la fonction Lambda

Pour créer la fonction Lambda :

1. Dans la console AWS Lambda, sélectionnez Crée une fonction Lambda.

The screenshot shows the AWS Lambda Functions page. At the top, there is a header with the text "Lambda > Functions". Below the header, a message says "You have 32 Lambda function(s) using 1.6 MB of code storage. Choose a name for your function (it can take up to 60 seconds to appear)". There are two buttons: "Create a Lambda function" (highlighted with a red box) and "Actions ▾". Below these buttons are two filter icons: a grid and a funnel, followed by a "Filter" input field. The main area displays a table of Lambda functions. The columns are "Function name" and "Description". The table contains the following data:

Function name	Description
myIoTButtonFunction	A starter AWS Lambda function
myLambaTest	Demonstrates how to invoke Lambda functions from AWS IoT Buttons
michgreFunction	A starter AWS Lambda function
MyCodeCommitFunction	
Awesome_CFN_Function_for_CodePipeline	Tim's sample Lambda function
ParseWiki	An Amazon Lambda function

2. Pour le filtre, tapez `hello-world`, puis sélectionnez le plan `hello-world`.

Lambda > New function

Step 1: Select blueprint

Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. You can use them as-is, copy and customize as needed, or skip this step if you want to author a Lambda function from scratch. Note that, unless otherwise noted, blueprints are licensed under CC0.

The screenshot shows the 'Select blueprint' step of the AWS Lambda 'New function' wizard. At the top, there's a search bar with a funnel icon labeled 'hello-world' and a 'All languages' dropdown. Below the search bar, several blueprint cards are displayed. The first card, 'hello-world', is highlighted with a red border. It contains the text 'A starter AWS Lambda function.' and 'nodejs' at the bottom. To its right is another card for 'hello-world-p' (partially visible) with 'python2.7' at the bottom. The background features a decorative wavy pattern at the bottom.

3. Sur la page Configurer une fonction , saisissez un nom et une description pour la fonction Lambda. Dans Runtime, sélectionnez Node.js 4.3.

Lambda > New function using blueprint hello-world

Step 1: Select blueprint

Step 2: Configure function

Step 3: Review

Configure function

A Lambda function consists of the custom code you want to execute.

Name* myIoTButtonFunc

Description Sends a message

Runtime* Node.js 4.3

- Faites défiler vers le bas jusqu'à la section Code de fonction Lambda de la page. Remplacez le code existant par le suivant :

```
console.log('Loading function'); // Charge le SDK AWS var AWS =  
require("aws-sdk"); // Configure le code à appeler lorsque la fonction  
Lambda est appelée exports.handler = (event, context, callback) =>  
{ // Charge le message transmis dans la fonction Lambda dans un objet  
JSON var eventText = JSON.stringify(event, null, 2); // Enregistre  
un message dans la console, vous pouvez afficher ce texte dans  
l'onglet Surveillance de la console Lambda ou de la console CloudWatch  
Logs console.log("Received event:", eventText); // Crée une chaîne pour  
l'extraction du type de clic et du numéro de série du message envoyé  
par le bouton AWS IoT var messageText = "Reçu " + event.clickType + "  
message de l'ID de bouton : " + event.serialNumber; // Écrit la chaîne  
dans la console console.log("Message à envoyer : " + messageText); //  
Crée un objet SNS var sns = new AWS.SNS(); // Renseigne les paramètres  
de l'opération de publication // - Message : texte du message à  
envoyer // - TopicArn : ARN de la rubrique Amazon SNS dans laquelle vous  
voulez effectuer la publication var params = { Message: messageText,
```

```
TopicArn: "arn:aws:sns:us-east-1:123456789012:MyIoTButtonSNSTopic" };  
sns.publish(params, context.done); };
```

5. Faites défiler vers le bas jusqu'à la section Gestionnaire de la fonction Lambda et rôle de la page. Pour Rôle, sélectionnez Rôle de base de l'exécution. La console IAM s'ouvre, qui vous permet de créer un rôle IAM que Lambda peut assumer lors de l'exécution de la fonction Lambda.

Pour modifier la stratégie de rôle afin de lui accorder l'autorisation de publier dans votre rubrique Amazon SNS :

1. Choisissez Afficher le document de stratégie.

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permission

▼ Hide Details

Role Summary

Role Lambda execution role permissions

Description

IAM Role lambda_basic_execution 

Policy Name Create a new Role Policy 

▶ **View Policy Document** 



Choisissez Modification pour modifier la stratégie du rôle.

▼ Hide Details

Role Summary ⓘ

Role Description Lambda execution role permissions

IAM Role lambda_basic_execution

Policy Name Create a new Role Policy

▼ Hide Policy Document

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "logs:CreateLogGroup",  
        "logs:CreateLogStream",  
        "logs:PutLogEvents"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

2. Remplacer le document de stratégie par les éléments suivants :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
  "Action": [ "logs>CreateLogGroup", "logs>CreateLogStream",  
            "logs>PutLogEvents" ], "Resource": "arn:aws:logs:*:*:*" },  
  { "Effect": "Allow", "Action": [ "sns>Publish" ], "Resource":  
    "arn:aws:sns:us-east-1:123456789012:MyIoTButtonSNSTopic" } ] }
```

Ce document de stratégie ajoute les autorisations de publication dans votre rubrique Amazon SNS.



Note

Cet exemple utilise un numéro de compte AWS fictif dans la ressource ARN.
Assurez-vous d'utiliser l'ARN pour votre rubrique Amazon SNS.

6. Choisissez Autoriser.

▼ Hide Details

Role Summary ?

Role Description Lambda execution role permissions

IAM Role lambda_basic_execution ▾

Policy Name Create a new Role Policy ▾

▼ Hide Policy Document

```
{  
    "Effect": "Allow",  
    "Action": [  
        "sns:Publish"  
    ],  
    "Resource": "arn:aws:sns:us-east-  
1:123456789012:MyIoTButtonSNSTopic"  
}  
]  
}
```

7. Conservez les valeur par défaut des Paramètres avancés , puis sélectionnez Suivant

Advanced settings

These settings allow you to control the code execution performance and cost. Selecting memory (or selecting memory) or changing the timeout may impact your function's performance and cost.

Memory (MB)*

128

Timeout*

0

min

3

All AWS Lambda functions run securely inside a default system-managed VPC. You can also run them within your own private VPC or within resources, such as databases, within your custom VPC. [Learn more about Lambda VPC support](#). To run your function in a VPC, select the appropriate permissions to configure VPC. Select "Basic with VPC" if you want to run your function in the default VPC.

VPC

No VPC

* These fields are required.

8. Sur la page Révision , sélectionnez Créer la fonction.

Review

Please review your Lambda function details. You can go back to edit changes or complete the setup process.

Lambda function

Name myIoTButtonFunction

Description Sends a message to SNS topic

Runtime Node.js 4.3

Handler index.handler

Role lambda_basic_execution

Memory (MB) 128

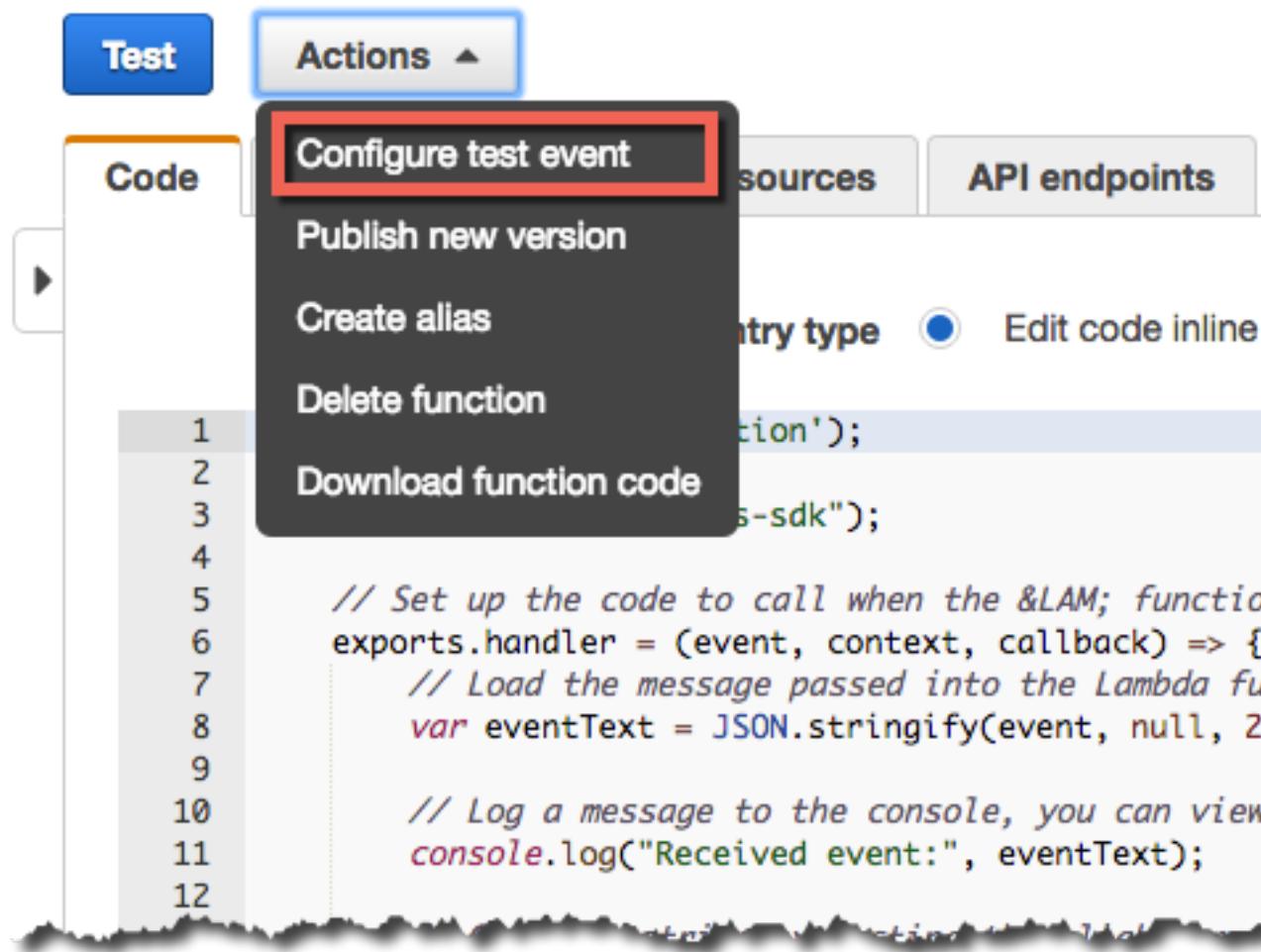
Timeout 3

Test de votre fonction Lambda

Pour tester la fonction Lambda :

1. Dans le menu Actions , choisissez Configurer un événement de test.

Lambda > Functions > myIoTButtonFunction



2. Copiez et collez le code JSON suivant dans la page Événement de test en entrée , puis sélectionnez Enregistrer et tester.

Input test event

It looks like you have not configured a test event for this function yet. Use this window to define the input to your function with (please remember that this will actually execute the code!).

Configure test event in the Actions list. Note that changes to the event template will affect the test event.

Sample event template

Hello World

```
1  {
2    "serialNumber": "ABCDEFG12345",
3    "clickType": "SINGLE",
4    "batteryVoltage": "2000 mV"
5 }
```

3. Dans la console AWS Lambda, sélectionnez l'onglet Surveillance , puis faites défiler vers le bas de l'écran. La section Sortie de journal affiche la sortie que la fonction Lambda a écrit dans la console.

Log output

The area below shows the logging calls in your code. These correspond to the group corresponding to this Lambda function. [Click here](#) to view the CloudWatch Logs for this function.

```
| START RequestId: c4b5b4d1-1631-11e6-b78f-0d4d596724ad Version: 1
| 2016-05-09T22:02:49.501Z           c4b5b4d1-1631-11e6-b78f-0d4d596724ad
|   "serialNumber": "ABCDEFG12345",
|   "clickType": "SINGLE",
|   "batteryVoltage": "2000 mV"
|
| }
| 2016-05-09T22:02:49.501Z           c4b5b4d1-1631-11e6-b78f-0d4d596724ad
| END RequestId: c4b5b4d1-1631-11e6-b78f-0d4d596724ad
| REPORT RequestId: c4b5b4d1-1631-11e6-b78f-0d4d596724ad Duration: 10ms
```

Création d'une règle Lambda

Maintenant que vous avez créé une fonction Lambda, vous pouvez créer une règle qui appelle la fonction Lambda.

1. Dans la [console AWS IoT](#), sélectionnez Créez une ressource.
2. Choisissez Créez une règle.
3. Tapez un nom et une description pour la règle.

The screenshot shows the AWS IoT Resources page. At the top, there is a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the main title "Resources" is displayed. In the top right corner of the main area, there is a button labeled "Close create panel" with a small "X" icon. On the right side of the main area, there are two buttons: "Create a thing" (with a wrench icon) and "Create a rule" (with a gear and connection icon). The "Create a rule" button is highlighted with a blue border. Below these buttons, the section title "Create a rule" is centered. A descriptive text follows: "Create a rule to evaluate inbound messages published into AWS IoT Core, such as a DynamoDB table." Further down, there is a form for defining a new rule:

Name	MyLambdaRule
Description	Invokes a Lambda function & calls

Create a rule

Create a rule to evaluate inbound messages published into AWS IoT Core, such as a DynamoDB table.

Name your rule and add an optional description.

Name	MyLambdaRule
Description	Invokes a Lambda function & calls

4. Saisissez les paramètres suivants pour la règle :

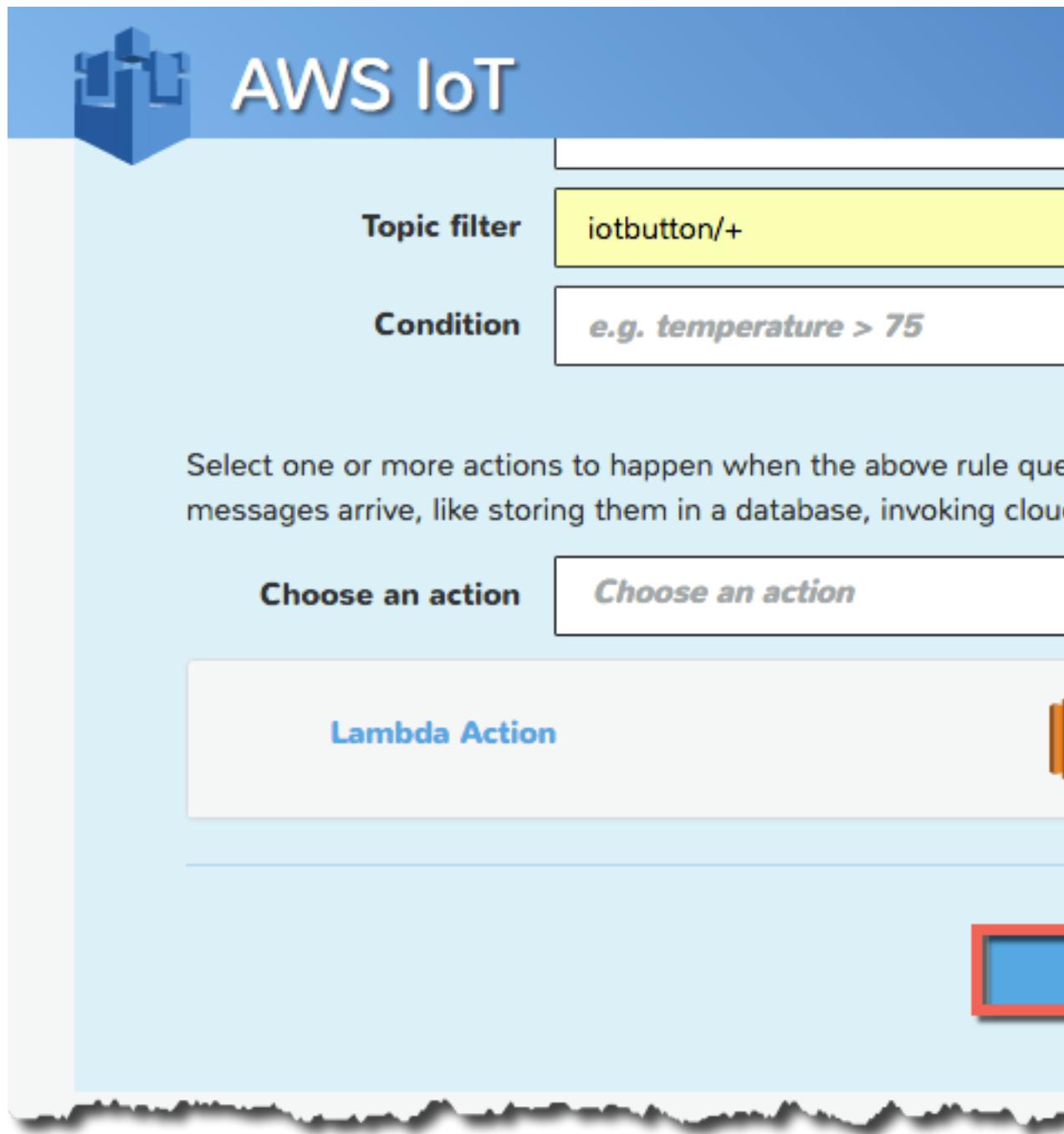
Indicate the source of the messages you want to process with this rule.

Rule query statement	SELECT * FROM 'iotbutton/+'
SQL version	2016-03-23-beta
Attribute	*
Topic filter	iotbutton/+
Condition	e.g. temperature > 75

5. Dans Choisir une action, sélectionnez Insérer ce message dans une fonction de code et l'exécuter (Lambda).
6. De Nom de la fonction, choisissez le nom de votre fonction Lambda, puis sélectionnez Ajouter une action.

The screenshot shows the AWS IoT Rule Creation interface. At the top, there's a blue header with the AWS IoT logo and the word "AWS IoT". Below it, there are two input fields: "Topic filter" containing "iotbutton/+/" and "Condition" containing "e.g. temperature > 75". A large text area below these says: "Select one or more actions to happen when the above rule queue messages arrive, like storing them in a database, invoking cloud functions, or sending notifications." Underneath, there's a section titled "Choose an action" with a "Lambda" button. This leads to another step where it says "This action will invoke a Lambda function with the following settings". It shows an input field for "*Function name" with "myIoTButtonFunction" entered, and two buttons at the bottom: "Cancel" and "Add action" (which is highlighted with a red border).

7. Choisissez Crée pour créer votre fonction Lambda.



Tester votre règle Lambda

Ce didacticiel suppose que vous avez terminé le [Didacticiel de démarrage AWS IoT \(p. 18\)](#), qui couvre :

- Configuration d'un bouton AWS IoT.
- Création d'une rubrique Amazon SNS et abonnement à cette rubrique avec un numéro de téléphone portable.

Maintenant que votre bouton est configuré et connecté au Wi-Fi et que vous avez configuré une rubrique Amazon SNS, vous pouvez cliquer sur le bouton pour tester votre règle Lambda. Vous devriez recevoir un texto sur votre téléphone qui contient le numéro de série de votre bouton, le type de pression (SIMPLE ou DOUBLE) et le voltage de la batterie.

Le message doit avoir l'aspect suivant :

```
IOT BUTTON> { "serialNumber" : "ABCDEFG12345", "clickType" : "SINGLE",  
"batteryVoltage" : "2000 mV" }
```

Si vous n'avez pas de bouton, vous pouvez en acheter un [ici](#) ou utiliser le client MQTT AWS IoT à la place.

1. Dans la [console AWS IoT](#), sélectionnez Client MQTT.



2. Tapez un ID de client ou sélectionnez Générer des ID de client, puis sélectionnez Connexion.



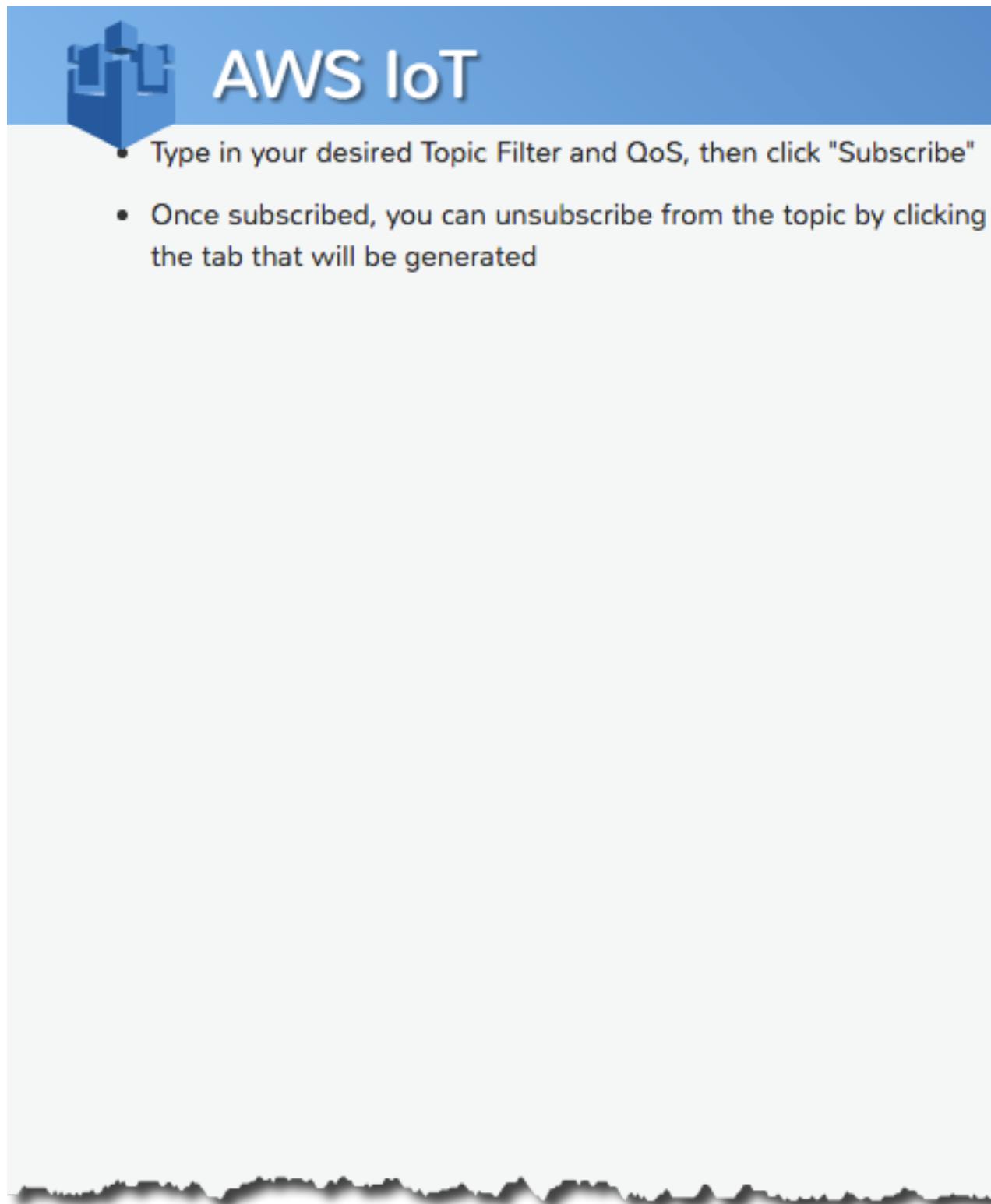
AWS IoT

You're not connected to the Device Gateway

- Go to the "Device Gateway connection" tab
- Type in your desired Client ID or generate one using the "Generate" button
- Once the connection to the Device Gateway succeeds, Subscribe to topic" and "Publish to topic" tabs
- If the connection to the Device Gateway fails, ensure that your Client ID is correctly typed and that your message is encoded in UTF-8

3. Choisissez Publier dans la rubrique.
4. Dans Publier une rubrique, type **iotbutton/ABCDEFG12345**.
5. Dans Charge utile, tapez le code JSON suivant, puis sélectionnez Publier.

```
{ "serialNumber" : "ABCDEFG12345" , "clickType" : "SINGLE" ,  
"batteryVoltage" : "2000 mV" }
```



Vous devriez recevoir un message sur votre téléphone portable.

Gestion des objets avec AWS IoT

AWS IoT fournit un service Thing Registry qui vous aide à gérer vos objets. Un objet est une représentation d'un dispositif spécifique ou d'une entité logique. Il peut s'agir d'un dispositif physique ou d'un capteur (par exemple, une ampoule ou un interrupteur sur un mur). Il peut également s'agir d'une entité logique, comme une instance d'une application ou une entité physique qui ne se connecte pas à AWS IoT, mais qui est associée à d'autres dispositifs qui se connectent (par exemple, une voiture dotée de capteurs de moteur ou d'un ordinateur de bord).

Les informations concernant un objet sont stockées dans le thing registry en tant que données JSON. Voici un exemple d'objet :

```
{ "version": 3, "thingName": "MyLightBulb", "defaultClientId": "MyLightBulb",  
"thingTypeName": "LightBulb", "attributes": { "model": "123", "wattage":  
"75" } }
```

Les objets sont identifiés par un nom. Ils peuvent également avoir des attributs, qui sont des paires nom-valeur, que vous pouvez utiliser pour stocker des informations concernant l'objet, comme son numéro de série ou le fabricant.

Un cas d'utilisation de dispositif classique consisterait à utiliser le nom d'un objet en tant qu'ID de client MQTT par défaut. Bien que nous n'appliquions pas de mapping entre le nom de registre d'un objet et son utilisation d'ID de client MQTT, de certificats ou d'état de thing shadow, nous vous recommandons de choisir un nom d'objet et de l'utiliser comme ID de client MQTT pour le thing registry et le service Thing Shadows. Cela permet d'organiser votre parc IoT plus facilement, sans perdre la souplesse du modèle de certificat d'appareil ou du thing shadow sous-jacent.

Vous n'avez pas besoin de créer un objet dans le thing registry pour le connecter à AWS IoT. Ajouter vos objets au thing registry permet de les gérer et de les rechercher plus facilement.

Gestion des objets avec le Thing Registry

Vous utilisez la console AWS IoT ou AWS CLI pour interagir avec le registre. Les sections suivantes montrent comment utiliser l'interface de ligne de commande pour qu'elle fonctionne avec le thing registry.

Créer un objet

La commande suivante montre comment utiliser AWS IoT `create-thing` Commande CLI pour créer un objet :

```
$ aws iot create-thing --thing-name "MyLightBulb" --attribute-payload  
"{"attributes": {"wattage": "75", "model": "123"} }"
```

Le `create-thing` L'API affiche le nom et l'ARN de votre nouvel objet :

```
{ "thingArn": "arn:aws:iot:us-east-1:803981987763:thing/MyLightBulb",  
"thingName": "MyLightBulb" }
```

Liste des objets

Vous pouvez utiliser l'API `list-things` pour répertorier tous les objets de votre compte :

```
$ aws iot list-things { "things": [ { "attributes": { "model": "123",  
"wattage": "75" }, "version": 1, "thingName": "MyLightBulb" },  
{ "attributes": { "numOfStates": "3" }, "version": 11, "thingName":  
"MyWallSwitch" } ] }
```

Recherche d'objets

Vous pouvez utiliser la Réseau de Distribution `describe-thing` API pour répertorier les informations concernant un objet spécifique :

```
$ aws iot describe-thing --thing-name "MyLightBulb" { "version":  
3, "thingName": "MyLightBulb", "defaultClientId": "MyLightBulb",  
"thingTypeName": "StopLight", "attributes": { "model": "123", "wattage":  
"75" } }
```

Vous pouvez rechercher tous les objets associés à un nom de type d'objet grâce à l'API `list-things` :

```
$ aws iot list-things --thing-type-name "LightBulb"
```

```
{ "things": [ { "thingTypeName": "LightBulb", "attributes": { "model":  
"123", "wattage": "75" }, "version": 1, "thingName": "MyRGBLight" },  
{ "thingTypeName": "LightBulb", "attributes": { "model": "123", "wattage":  
"75" }, "version": 1, "thingName": "MySecondLightBulb" } ] }
```

Vous pouvez aussi rechercher tous les objets qui ont un attribut avec une valeur spécifique grâce à l'API `list-things` :

```
$ aws iot list-things --attribute-name "wattage" --attribute-value "75"
```

```
{ "things": [ { "thingTypeName": "StopLight", "attributes": { "model":  
"123", "wattage": "75" }, "version": 3, "thingName": "MyLightBulb" },  
{ "thingTypeName": "LightBulb", "attributes": { "model": "123", "wattage":  
"75" }, "version": 1, "thingName": "MyRGBLight" }, { "thingTypeName":  
"LightBulb", "attributes": { "model": "123", "wattage": "75" }, "version":  
1, "thingName": "MySecondLightBulb" } ] }
```

Mettre à jour un objet

Vous pouvez utiliser l'API `update-thing` pour mettre à jour un objet :

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload
"{"attributes": {"wattage": "150", "model": "456"} }"
```

La commande `update-thing` ne génère pas de sortie. Vous pouvez utiliser l'API `describe-thing` pour voir le résultat :

```
$ aws iot describe-thing --thing-name "MyLightBulb" { "attributes": {
    "model": "456", "wattage": "150" }, "version": 2, "thingName": "MyLightBulb" }
```

Supprimer un objet

Vous pouvez utiliser l'API `delete-thing` pour supprimer un objet :

```
$ aws iot delete-thing --thing-name "MyThing"
```

Attacher un mandataire à un objet

Un dispositif physique doit posséder un certificat X.509 pour pouvoir communiquer avec AWS IoT. Vous pouvez associer le certificat de votre dispositif à l'objet dans le thing registry qui représente votre appareil. Pour attacher un certificat à votre objet, utilisez l'API `attach-thing-principal` API:

```
$ aws iot attach-thing-principal --thing-name "MyLightBulb"
--principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739ale94ee4162977b02b12842847"
```

La commande `attach-thing-principal` ne génère pas de sortie.

Détacher un mandataire d'un objet

Vous pouvez utiliser la Réseau de Distribution `detach-thing-principal` API pour détacher un certificat d'un objet :

```
$ aws iot detach-thing-principal --thing-name "MyLightBulb"
--principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739ale94ee4162977b02b12842847"
```

La commande `detach-thing-principal` ne génère pas de sortie.

Types d'objets

Les types d'objets permettent de stocker des informations de configuration et de description communes à tous les objets qui associés au même type d'objet. Cela simplifie la gestion des objets dans le thing registry. Par exemple, vous pouvez définir un type d'objet ampoule, LightBulb. Tous les

objets associés au type d'objet LightBulb partagent un ensemble d'attributs : numéro de série, fabricant et puissance. Lorsque vous créez un objet de type LightBulb (ou que vous modifiez le type d'un objet existant en LightBulb), vous pouvez spécifier des valeurs pour chacun des attributs définis dans le type d'objet LightBulb.

Bien que les types d'objet soient facultatifs, leur utilisation permet de mieux découvrir les objets.

- Les objets peuvent posséder jusqu'à 50 attributs.
- Les objets sans type d'objet peuvent posséder jusqu'à trois attributs.
- Un objet ne peut être associé qu'à un seul type d'objet.
- Il n'y a aucune limite au nombre de types d'objets que vous pouvez créer dans votre compte.

Les types d'objets sont immuables. Vous ne pouvez pas modifier un nom de type d'objet après sa création. Vous pouvez rendre obsolète un type d'objet à tout moment pour empêcher de nouveaux objets d'y être associés. Vous pouvez aussi supprimer les types d'objets qui n'ont aucun objet associé.

Créer un type d'objet

Vous pouvez utiliser l'API `create-thing-type` pour créer un type d'objet :

```
$ aws iot create-thing-type --thing-type-name "LightBulb" --  
thing-type-properties "thingTypeDescription=light bulb type,  
searchableAttributes=wattage,model"
```

La commande `create-thing-type` renvoie une réponse qui contient le type d'objet et son ARN :

```
{ "thingTypeName": "LightBulb", "thingTypeArn": "arn:aws:iot:us-  
west-2:803981987763:thingtype/LightBulb" }
```

Liste des types d'objets

Vous pouvez utiliser l'API `list-thing-types` pour répertorier des types d'objets :

```
$ aws iot list-thing-types
```

La commande `list-thing-types` renvoie une liste des types d'objets définis dans votre compte AWS :

```
{ "thingTypes": [ { "thingTypeName": "LightBulb", "thingTypeProperties":  
{ "deprecated": false, "creationDate": 1468423800950,  
"searchableAttributes": [ "wattage", "model" ], "thingTypeDescription":  
"light bulb type" } } ] }
```

Décrire un type d'objet

Vous pouvez utiliser l'API `describe-thing-type` pour obtenir des informations sur un type d'objet :

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

L'API `describe-thing-type` répond avec des informations sur le type spécifié :

```
{ "thingTypeName": "LightBulb", "thingTypeProperties": { "deprecated": false, "creationDate": 1468423800950, "searchableAttributes": [ "wattage", "model" ], "thingTypeDescription": "light bulb type" } }
```

Associer un type d'objet à un objet

Vous pouvez utiliser l'API `create-thing` pour spécifier un type d'objet lorsque vous créez un objet :

```
$ aws iot create-thing --thing-name "MySecondLightBulb" --thing-type-name "LightBulb" --attribute-payload "{\"attributes\": {\"wattage\": \"75\", \"model\": \"123\"}}"
```

Vous pouvez utiliser l'API `update-thing` à tout moment pour modifier le type d'objet associé à un objet :

```
$ aws iot update-thing --thing-name "MyLightBulb" --thing-type-name "StopLight" --attribute-payload "{\"attributes\": {\"wattage\": \"75\", \"model\": \"123\"}}"
```

Vous pouvez également utiliser l'API `update-thing` pour dissocier un objet d'un type d'objet.

Rendre obsolète un type d'objet

Les types d'objets sont immuables. Il est impossible de les modifier une fois qu'ils sont définis. Vous pouvez, toutefois, rendre obsolète un type d'objet pour empêcher les utilisateurs de lui associer de nouveaux objets. Tous les objets existants associés au type d'objet restent inchangés.

Pour rendre obsolète un type d'objet, utilisez l'API `deprecate-thing-type` :

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

Vous pouvez voir le résultat en utilisant l'API `escribe-thing-type` :

```
$ aws iot describe-thing --thing-type-name "StopLight":
```

```
{ "thingTypeName": "StopLight", "thingTypeProperties": { "deprecated": true, "creationDate": 1468425854308, "searchableAttributes": [ "wattage", "numOfLights", "model" ], "thingTypeDescription": "traffic light type", "deprecationDate": 1468446026349 } }
```

Rendre obsolète un type d'objet est une opération réversible. Vous pouvez annuler une obsolescence en utilisant l'indicateur `--undo-deprecate` avec la commande CLI `deprecate-thing-type` :

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

Vous pouvez voir le résultat en utilisant l'API `deprecate-thing-type` :

```
$ aws iot deprecate-thing-type --thing-type-name "StopLight":
```

```
{ "thingTypeName": "StopLight", "thingTypeProperties": { "deprecated": false, "creationDate": 1468425854308, "searchableAttributes": [ "wattage", "numOfLights", "model" ], "thingTypeDescription": "traffic light type" } }
```

Supprimer un type d'objet

Vous pouvez supprimer des types d'objet uniquement une fois qu'ils sont obsolètes. Pour supprimer un type d'objet, utilisez l'API `delete-thing-type` :

```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

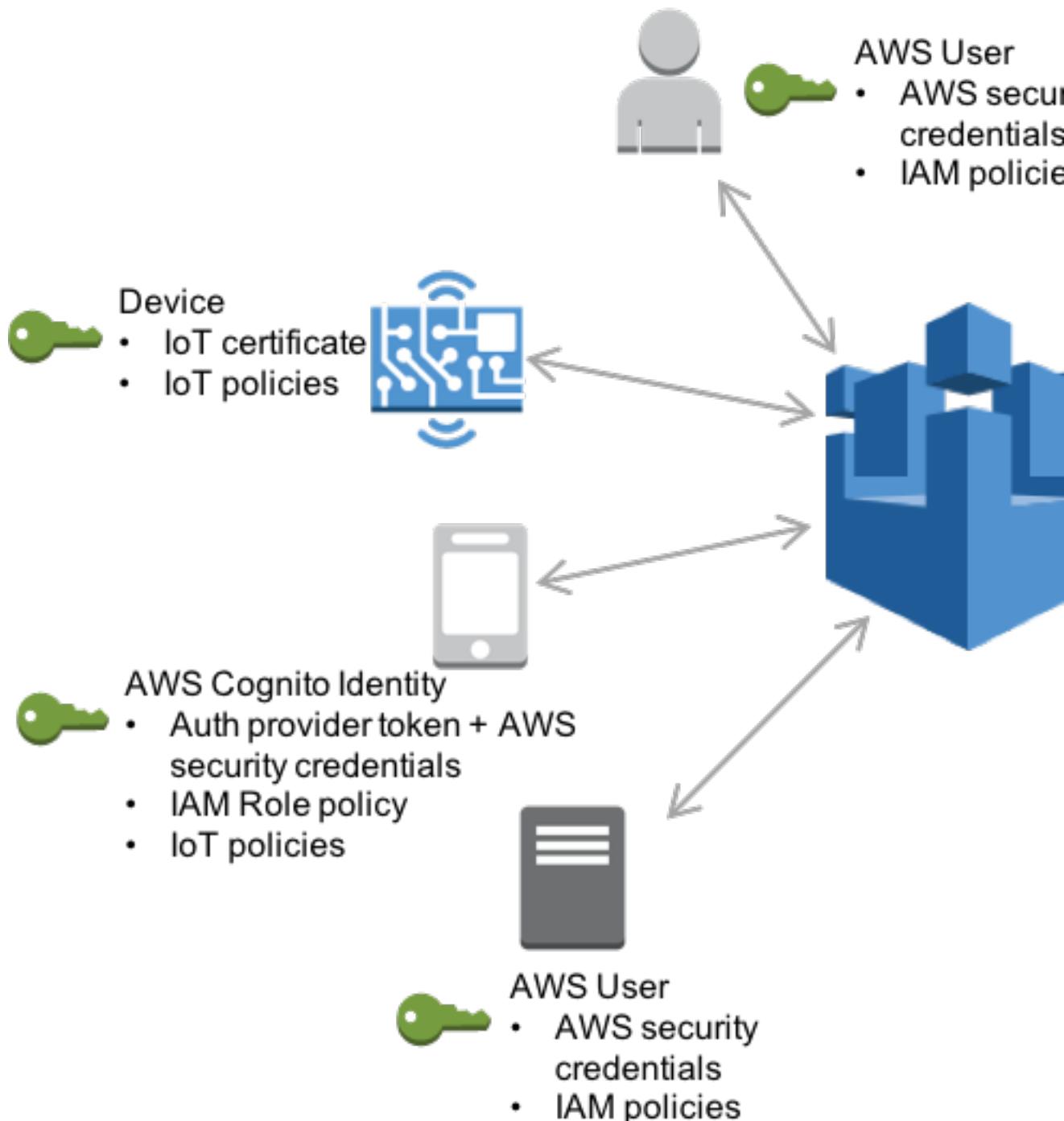


Note

Vous devez attendre 5 minutes après avoir rendu obsolète un type d'objet avant de le supprimer.

Sécurité et identité pour AWS IoT

Chaque dispositif connecté doit disposer d'informations d'identification pour accéder à l'agent de messages ou au service Thing Shadows. Tout le trafic vers et depuis AWS IoT doit être chiffré sur le protocole Transport Layer Security (TLS). Les informations d'identification du dispositif doivent être conservées dans un endroit sûr, afin d'envoyer des données en toute sécurité à l'agent de messages. Une fois que les données atteignent l'agent de messages, les mécanismes de sécurité du Cloud AWS protègent les données lors de leur transfert entre AWS IoT et d'autres dispositifs ou services AWS.



- Vous êtes responsable de la gestion des informations d'identification de dispositif (certificats X.509, informations d'identification AWS) sur vos dispositifs ainsi que des stratégies dans AWS IoT. Vous êtes responsable de l'affectation d'identités uniques à chaque dispositif et de la gestion des autorisations pour un dispositif ou un groupe de dispositifs.

- Les dispositifs se connectent à l'aide de l'identité de votre choix (certificats X.509, utilisateurs et groupes IAM ou identités Amazon Cognito) via une connexion sécurisée selon le modèle de connexion AWS IoT.
- L'agent de messages AWS IoT authentifie et autorise toutes les actions au sein de votre compte. Il est responsable de l'authentification de vos dispositifs, de l'intégration sécurisée des données de dispositif et du respect des autorisations d'accès que vous placez sur les dispositifs à l'aide de stratégies.
- Le moteur de stratégies de AWS IoT transfère les données du dispositif vers d'autres dispositifs et d'autres services AWS selon les règles que vous définissez. Il est responsable de l'exploitation des systèmes de gestion de l'accès AWS pour transférer les données en toute sécurité vers leur destination finale.

Authentification dans AWS IoT

AWS IoT prend en charge trois types de principes d'identité pour l'authentification :

- Certificats X.509
- utilisateurs, groupes et rôles IAM
- Identités Amazon Cognito

Chaque type d'identité prend en charge différents cas d'utilisation pour accéder à l'agent de messages AWS IoT et au service Thing Shadows.

Le type d'identité que vous utilisez dépend de votre choix de protocole d'application. Si vous utilisez HTTP, utilisez des identités IAM (utilisateurs, groupes, rôles) ou Amazon Cognito. Si vous utilisez MQTT, utilisez des certificats X.509.

Certificats X.509

Les certificats X.509 sont des certificats numériques qui font appel à la norme d'infrastructure de clé publique X.509 pour associer une clé publique à une identité contenue dans un certificat. Les certificats X.509 sont émis par une entité de confiance appelée autorité de certification (CA). Celle-ci gère un ou plusieurs certificats spéciaux appelés certificats d'autorité de certification, qu'elle utilise pour émettre des certificats X.509. Seule l'autorité du certificat a accès aux certificats d'autorité de certification.

AWS IoT prend en charge les algorithmes de signature de certificat suivants :

- SHA256WITHRSA
- SHA384WITHRSA
- SHA384WITHRSA
- SHA512WITHRSA
- RSASSAPSS
- DSA_WITH_SHA256
- ECDSA-WITH-SHA256
- ECDSA-WITH-SHA384
- ECDSA-WITH-SHA512

Les certificats X.509 offrent plusieurs avantages par rapport à d'autres mécanismes d'identification et d'authentification. Les certificats X.509 activent des clés asymétriques à utiliser avec les dispositifs.

Vos processus de fabrication et dispositifs peuvent contrôler les clés. Vous n'avez pas besoin de vous appuyer sur AWS pour générer des informations d'identification de sécurité. Cela signifie que vous pouvez graver les clés privées dans un stockage sécurisé sur un dispositif sans jamais autoriser le matériel cryptographique stratégique à quitter le dispositif. Les certificats offrent une authentification du client plus fiable via d'autres systèmes, tels que le nom d'utilisateur et le mot de passe ou les jetons de porteur, car la clé secrète ne quitte jamais le dispositif.

AWS IoT authentifie les certificats grâce au mode d'authentification de client du protocole TLS. TLS est disponible dans de nombreux langages de programmation et systèmes d'exploitation ; il est couramment utilisé pour le chiffrement des données. Dans l'authentification de client TLS, AWS IoT demande un certificat X.509 de client et valide l'état et le compte AWS du certificat par rapport à un registre de certificats. Il teste ensuite le client pour vérifier la propriété de la clé privée qui correspond à la clé publique contenue dans le certificat.

Pour utiliser les certificats AWS IoT, les clients doivent prendre en charge tous les éléments suivants dans leur implémentation TLS :

- TLS 1.2.
- Validation de la signature de certificat RSA SHA-256.
- Une des suites de chiffrement de la section de prise en charge de suites de chiffrement TLS.

Certificats X.509 et AWS IoT

AWS IoT peut utiliser des certificats générés par AWS IoT ou des certificats signés par un certificat d'autorité de certification pour l'authentification du dispositif. Les certificats générés par AWS IoT n'expireront pas. La date et l'heure d'expiration des certificats signés par un certificat d'autorité de certification sont définies au moment de la création du certificat.

Pour utiliser un certificat qui n'est pas créé par AWS IoT, vous devez enregistrer un certificat d'autorité de certification. Tous les certificats de dispositif doivent être signées par le certificat d'autorité de certification que vous enregistrez.

Vous pouvez utiliser la console AWS IoT ou l'interface de ligne de commande pour créer et gérer des certificats. Les opérations suivantes sont disponibles :

- Créer et enregistrer un certificat AWS IoT.
- Enregistrer un certificat d'autorité de certification.
- Enregistrer un certificat d'appareil.
- Activer ou désactiver un certificat d'appareil.
- Révoquer un certificat d'appareil.
- Transfert un certificat d'appareil à un autre compte AWS.
- Répertorier tous les certificats d'autorité de certification enregistrés dans votre compte AWS.
- Répertorier tous les certificats d'appareil enregistrés dans votre compte AWS.

Pour plus d'informations sur les commandes CLI à utiliser pour effectuer ces opérations, consultez la page [Référence de l'interface de ligne de commande AWS IoT](#).

Authentification du serveur

Les certificats d'appareil permettent à AWS IoT d'authentifier les dispositifs. Afin de vous assurer que votre appareil communique avec AWS IoT et pas un autre serveur se faisant passer pour AWS IoT, copiez le [certificat d'autorité de certification racine](#) sur votre dispositif. Dans le code de votre dispositif, faites référence au certificat d'autorité de certification racine lors de la connexion à AWS IoT. Pour plus d'informations, consultez la [SDK pour les appareils AWS IoT \(p. 166\)](#).



Note

Vous ne pouvez pas utiliser votre propre certificat d'autorité de certification pour authentifier le serveur AWS IoT, uniquement le certificat d'autorité de certification racine VeriSign.

Créer et enregistrer un certificat d'appareil AWS IoT

Vous pouvez utiliser la console AWS IoT ou la CLI AWS IoT pour créer un certificat AWS IoT.

Pour créer un certificat (console)

Vous pouvez utiliser l'API [UpdateCertificate](#) pour révoquer un certificat à tout moment. Pour plus d'informations sur la gestion des certificats d'appareil, consultez [AWS Command Line Interface Guide de l'utilisateur](#).

1. Connectez-vous à AWS Management Console et ouvrez la [console AWS IoT](https://console.aws.amazon.com/iot) à <https://console.aws.amazon.com/iot>.
2. Choisissez Créer une ressource, puis sélectionnez Créer un certificat.
3. Choisissez Créer un certificat avec 1-Click. Vous pouvez également générer une demande de signature de certificat (CSR). Cliquez sur le bouton Créer avec CSR .
4. Utilisez les liens vers la clé publique, la clé privée et le certificat pour les télécharger dans un emplacement sécurisé.
5. Le certificat nouvellement créé s'affiche comme étant INACTIVE. Sélectionnez-le et, dans la liste déroulante Actions , sélectionnez Activation.

Pour créer un certificat (CLI)

La CLI AWS IoT fournit deux commandes pour créer des certificats :

- [create-keys-and-certificate](#)

Le [CreateKeysAndCertificate](#) L'API crée une clé privée, une clé publique et un certificat X.509.

- [create-certificate-from-csr](#)

Le [CreateCertificateFromCSR](#) L'API crée un certificat avec une CSR.

Utiliser votre propre certificat

Pour utiliser vos propres certificats X.509, vous devez enregistrer un certificat d'autorité de certification avec AWS IoT. Le certificat d'autorité de certification (CA) peut ensuite servir pour signer les certificats d'appareil. Vous pouvez enregistrer jusqu'à 10 certificats CA avec le même champ d'objet et la même clé publique par compte AWS. Cela vous permet d'avoir plusieurs autorités de certification qui signent les certificats de votre appareil.



Note

Les certificats d'appareil doivent être signés par le certificat CA enregistré. Il est fréquent qu'un certificat CA soit utilisé pour créer un certificat CA intermédiaire. Si vous utilisez un certificat intermédiaire pour signer les certificats de votre appareil, vous devez enregistrer le certificat CA intermédiaire. Vous devez utiliser le certificat CA racine AWS IoT lors de la connexion à AWS IoT, même si vous enregistrez votre propre certificat CA racine. Le certificat CA racine AWS IoT est utilisé par un appareil afin de vérifier l'identité des serveurs AWS IoT.

Table des matières

- [Enregistrer votre certificat CA \(p. 99\)](#)
- [Création d'un certificat d'appareil \(p. 100\)](#)
- [Enregistrement d'un certificat d'appareil \(p. 101\)](#)
- [Enregistrement manuel des certificats d'appareil \(p. 101\)](#)
- [Utilisation de l'enregistrement automatique/juste-à-temps pour les certificats d'appareil \(p. 101\)](#)
- [Désactiver le certificat CA \(p. 102\)](#)
- [Révoquer le certificat de l'appareil \(p. 102\)](#)

Si vous n'avez pas de certificat CA, vous pouvez en créer un grâce aux outils [OpenSSL](#) .

Pour créer un certificat CA

1. Générez une paire de clés.

```
openssl genrsa -out rootCA.key 2048
```

2. Utilisez la clé privée de la paire de clés pour générer un certificat CA.

```
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
```

Enregistrer votre certificat CA

Pour enregistrer votre certificat, vous devez obtenir un code d'enregistrement auprès de AWS IoT, signer un certificat de vérification de clé privée avec votre certificat CA, puis transmettre votre certificat CA et un certificat de vérification de clé privée à la commande CLI `register-ca-certificate`. Commande CLI. Le champ `Nom commun` du certificat de vérification de clé privée doit être défini comme étant le code d'enregistrement généré par la commande CLI `get-registration-code`. Un code d'enregistrement unique est généré par le compte AWS. Vous pouvez utiliser la commande `register-ca-certificate` ou la console AWS IoT pour enregistrer les certificats CA.

Pour enregistrer un certificat d'autorité de certification

1. Obtenez un code d'enregistrement auprès de AWS IoT. Ce code sera utilisé en tant que `Nom commun` du certificat de vérification de clé privée.

```
aws iot get-registration-code
```

2. Générer une paire de clés pour le certificat de vérification de clé privée.

```
openssl genrsa -out privateKeyVerificationCert.key 2048
```

3. Créer une CSR pour le certificat de vérification de clé privée, en définissant le champ `Nom commun` du certificat comme étant votre code d'enregistrement.

```
openssl req -new -key privateKeyVerificationCert.key -out privateKeyVerificationCert.csr
```

Vous êtes invité à saisir plus d'informations, y compris le certificat `Nom commun`.

```
Nom du pays (code à 2 lettres) [AU] : Nom de l'état ou de la province  
(nom complet) [] : Nom de la localité (par ex, ville) [] : Nom de
```

l'organisation (par ex, entreprise) [] : Nom de l'unité d'organisation
(par ex, section) [] : Nom commun (par ex, serveur FQDN ou VOTRE nom)
[] : Adresse électronique XXXXXXXXXXXXMONCODEENREGISTREMENTXXXXXX [] :

4. Utiliser la CSR pour créer un certificat de vérification de clé privée.

```
openssl x509 -req -in privateKeyVerificationCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out privateKeyVerificationCert.crt -days 500 -sha256
```

5. Enregistrez le certificat CA avec AWS IoT, en transmettant le certificat CA et le certificat de vérification de clé privée à la commande CLI register-ca-certificate .

```
aws iot register-ca-certificate --ca-certificate file://rootCA.pem -- verification-cert file://privateKeyVerificationCert.crt
```

6. Activez le certificat CA à l'aide la commande CLI update-certificate .

```
aws iot update-ca-certificate --certificate-id xxxxxxxxxxxx --new-status ACTIVE
```

Création d'un certificat d'appareil

Vous pouvez utiliser un certificat de CA enregistré auprès de AWS IoT pour créer un certificat d'appareil. Le certificat d'appareil doit être enregistré auprès de AWS IoT avant d'être utilisé.

Pour créer un certificat d'appareil

1. Générez une paire de clés.

```
openssl genrsa -out deviceCert.key
```

2. Créez une CSR pour le certificat d'appareil.

```
openssl req -new -key deviceCert.key -out deviceCert.csr
```

Vous serez invité à saisir des informations supplémentaires, comme illustré ici.

Nom du pays (code à 2 lettres) [AU] : Nom de l'état ou de la province
(nom complet) [] : Nom de la localité (par ex, ville) [] : Nom de l'organisation (par ex, entreprise) [] : Nom de l'unité d'organisation (par ex, section) [] : Nom commun (par ex, serveur FQDN ou VOTRE nom) [] : Adresse électronique [] :

3. Créez un certificat d'appareil à partir de la CSR.

```
openssl x509 -req -in deviceCert.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out deviceCert.crt -days 500 -sha256
```



Note

Vous devez utiliser un certificat CA enregistré auprès de AWS IoT pour créer des certificats d'appareil. Si vous avez plusieurs certificats CA (avec le même champ objet et la même clé publique) enregistrés dans votre compte AWS, vous devez spécifier le

certificat CA utilisé pour créer le certificat d'appareil lors de l'enregistrement de votre certificat d'appareil.

4. Enregistrer un certificat d'appareil

```
aws iot register-certificate --certificate file://deviceCert.crt --  
caCertificate file://caCert.crt
```

5. Activez le certificat d'appareil à l'aide de la commande CLI `update-certificate`.

```
aws iot update-ca-certificate --certificate-id xxxxxxxxxxxx --new-status  
ACTIVE
```

Enregistrement d'un certificat d'appareil

Vous devez utiliser un certificat CA enregistré auprès de AWS IoT pour signer des certificats d'appareil. Si vous avez plusieurs certificats CA (avec le même champ objet et la même clé publique) enregistrés dans votre compte AWS, vous devez spécifier le certificat CA utilisé pour signer le certificat d'appareil lors de l'enregistrement de votre certificat d'appareil. Vous pouvez enregistrer manuellement chaque certificat ou utiliser l'enregistrement automatique qui permet aux appareils d'enregistrer leur certificat lorsqu'ils se connectent à AWS IoT pour la première fois.

Enregistrement manuel des certificats d'appareil

Utilisez la commande CLI suivante pour enregistrer un certificat d'appareil :

```
aws iot register-certificate --certificate file://deviceCert.crt --  
caCertificate file://caCert.crt
```

Utilisation de l'enregistrement automatique/juste-à-temps pour les certificats d'appareil

Vous pouvez également faire en sorte que vos certificats d'appareil soient enregistrés automatiquement lorsque les appareils se connectent à AWS IoT pour la première fois. Pour ce faire, vous devez activer l'enregistrement automatique pour votre certificat CA. Cela enregistre automatiquement tout certificat d'appareil signé par votre certificat CA lorsqu'il se connecte à AWS IoT.

Activer l'enregistrement automatique

Utilisez l'API `update-ca-certificate` pour définir les certificats CA `auto-registration-status` sur `ENABLE`:

```
$ aws iot update-ca-certificate --certificate-id caCertificateId --new-auto-  
registration-status ENABLE
```

Vous pouvez également définir `auto-registration-status` sur `ENABLE` lorsque vous enregistrez votre certificat CA à l'aide de l'API `register-ca-certificate` :

```
aws iot register-ca-certificate --ca-certificate file://rootCA.pem  
--verification-cert file://privateKeyVerificationCert.crt --permet  
l'enregistrement automatique
```

Lorsqu'un dispositif tente de se connecter à AWS IoT, dans le cadre de la liaison TLS, il doit présenter un certificat CA enregistré et un certificat d'appareil. AWS IoT reconnaît le certificat d'autorité de certification comme étant un certificat CA enregistré et enregistre automatiquement le certificat de l'appareil et défini son état à la valeur `PENDING_ACTIVATION`. Cela signifie que le certificat de l'appareil a été enregistré automatiquement et qu'il est en attente d'activation. Un certificat doit se

trouver dans l'état ACTIVE pour être utilisé pour la connexion à AWS IoT. Lorsque AWS IoT enregistre automatiquement un certificat ou lorsqu'un certificat en état PENDING_ACTIVATION se connecte, AWS IoT publie un message dans la rubrique MQTT suivante :

```
$aws/events/certificates/registered/caCertificateId
```

Où *caCertificateId* est l'ID de l'autorité de certification ayant émis le certificat de l'appareil.

Le message publié sur cette rubrique a la structure suivante :

```
{ "certificateId": "certificateID", "caCertificateId": "caCertificateId",  
"timestamp": timestamp, "certificateStatus": "PENDING_ACTIVATION",  
"awsAccountId": "awsAccountId", "certificateRegistrationTimestamp":  
"certificateRegistrationTimestamp" }
```

Vous pouvez créer une règle qui écoute cette rubrique et effectue certaines actions supplémentaires. Nous vous recommandons de créer une règle Lambda qui vérifie que le certificat de l'appareil ne figure pas sur une liste de révocation de certificat (CRL), qui active le certificat et crée et attache une stratégie au certificat. La stratégie détermine à quelles ressources l'appareil est en mesure d'accéder. Pour obtenir un exemple de création d'une règle Lambda qui écoute la rubrique \$aws/events/certificates/registered/*caCertificateId* et réalise ces actions, consultez la page [Enregistrement juste-à-temps](#).

Désactiver le certificat CA

Lorsque vous tentez d'enregistrer un certificat d'appareil, AWS vérifie si le certificat CA associé est ACTIVE. Si le certificat CA est INACTIVE, AWS IoT n'autorise pas son enregistrement. En marquant le certificat comme étant INACTIVE, vous empêchez les nouveaux certificats d'appareil émis par l'autorité de certification compromise d'être enregistré dans votre compte. Vous pouvez désactiver le certificat CA à l'aide de l'API update-ca-certificate API:

```
$ aws iot update-ca-certificate --certificate-id certificateId --nouvel état  
INACTIVE
```



Note

Tous les certificats d'appareil qui ont été signés par le certificat CA compromis continueront de fonctionner jusqu'à ce que vous révoquiez explicitement le certificat de l'appareil.

Utilisez l'API ListCertificatesByCA pour obtenir une liste de tous les certificats d'appareils enregistrés qui ont été signés par l'autorité de certification compromise. Pour chaque certificat d'appareil signé par le certificat CA compromis, utilisez l'API UpdateCertificate pour révoquer le certificat de l'appareil, afin qu'il ne soit pas utilisé.

Révoquer le certificat de l'appareil

Si vous détectez toute activité douteuse provenant d'un certificat d'appareil enregistré, vous pouvez le révoquer en utilisant l'API update-certificate :

```
$ aws iot update-certificate --certificate-id certificateId  
--new-status REVOKED
```

Si une erreur ou exception se produit lors de l'enregistrement automatique des certificats d'appareil, INACTIVE envoie les événements/messages correspondants à votre CloudWatch Logs. Pour plus d'informations sur la configuration de CloudWatch Logs pour votre compte, consultez la [documentation CloudWatch Logs](#).

Utilisateurs, groupes et rôles IAM

Les utilisateurs, groupes et rôles IAM constituent le mécanisme standard pour la gestion des identités et authentification dans AWS. Comme avec n'importe quel autre service AWS, vous pouvez les utiliser pour vous connecter à aux interfaces HTTP de AWS IoT à l'aide du SDK et de l'AWS CLI.

Les rôles IAM constituent également la base de la sécurité AWS IoT dans le cloud. Les rôles permettent à AWS IoT d'émettre des appels à d'autres ressources AWS de votre compte en votre nom. Si vous souhaitez qu'un dispositif publie son état dans une table DynamoDB, par exemple, les rôles IAM autorisent AWS IoT à effectuer le travail en toute sécurité. Pour plus d'informations, consultez [Rôles IAM](#).

Pour les connexions à l'agent de messages, AWS IoT authentifie les utilisateurs, groupes et rôles IAM à l'aide du processus Signature Version 4. Pour plus d'informations sur l'authentification des informations d'identification de sécurité AWS, consultez [Signature des requêtes d'API AWS](#).

Lorsque vous utilisez Signature Version 4 d' AWS avec AWS IoT, les clients doivent prendre en charge les éléments suivants dans leur implémentation de TLS :

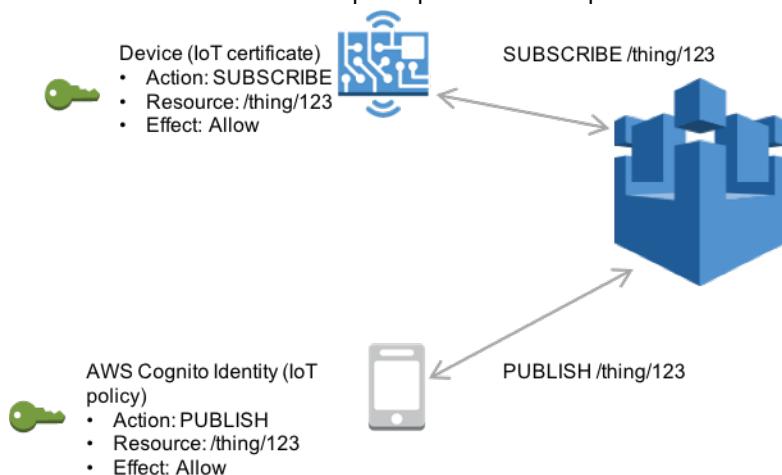
- TLS 1.2, TLS 1.1, TLS 1.0.
- Validation de la signature de certificat RSA SHA-256.
- Une des suites de chiffrement de la section de prise en charge de suites de chiffrement TLS.

Pour de plus amples informations, consultez le [IAM Guide de l'utilisateur](#).

Identités Amazon Cognito

Une identité Amazon Cognito vous permet d'utiliser votre propre fournisseur d'identité ou de mettre à profit d'autres fournisseurs d'identité plébiscités, comme Login with Amazon, Facebook ou Google. Vous échangez un jeton provenant de votre fournisseur d'identité contre les informations d'identification de sécurité AWS. Les informations d'identification représentent un rôle IAM, elles peuvent être utilisées avec AWS IoT.

AWS IoT étend Amazon Cognito et permet l'attachement de stratégies aux identités Amazon Cognito. Vous pouvez attacher une stratégie à une identité Amazon Cognito et donner des autorisations précises d'accès à un utilisateur de votre application AWS IoT. Cela peut servir pour attribuer des autorisations entre des clients spécifiques et leurs dispositifs.



Pour plus d'informations, consultez [Identité Amazon Cognito](#).

Autorisation

La communication avec Amazon Cognito suit le principe du moindre privilège. Une identité peut réaliser des opérations Amazon Cognito uniquement si vous accordez les autorisations appropriées. Vous créez des stratégies AWS IoT et IAM pour donner des autorisations à des identités authentifiées dans AWS IoT.

Les stratégies accordent des autorisations aux clients AWS IoT quel que soit le mécanisme d'authentification qu'ils utilisent pour se connecter à AWS IoT. Pour contrôler les ressources auxquelles un dispositif peut accéder, attachez une ou plusieurs stratégies AWS IoT au certificat associé au dispositif. Pour contrôler les ressources auxquelles une application Web ou mobile peut accéder, attachez une ou plusieurs stratégies AWS IoT au pool d'identités Amazon Cognito associé à l'application. Les stratégies AWS IoT contrôlent l'accès aux ressources AWS IoT (rubriques MQTT, appareils, thing shadows, etc.). Les stratégies IAM contrôlent l'accès aux autres services AWS et sont attachées aux utilisateurs, groupes et rôles IAM.

L'autorisation basée sur la stratégie est un outil puissant. Il vous donne un contrôle complet sur les sujets et les filtres de rubriques de votre compte AWS. Par exemple, imaginons un dispositif se connectant à AWS IoT avec un certificat. Vous pouvez ouvrir l'accès à toutes les rubriques ou limiter l'accès à une seule rubrique. Ce dernier exemple permet d'affecter une rubrique par appareil. Par exemple, le dispositif portant l'ID 123ABC peut s'abonner à /device/123ABC et vous pouvez accorder aux autres identités l'autorisation de s'abonner à cette rubrique, ce qui ouvre effectivement un canal de communication sur cet appareil.

Stratégies AWS IoT

Les stratégies AWS IoT sont des documents JSON. Elles suivent les mêmes conventions que les stratégies IAM. Pour plus d'informations, consultez [Présentation des stratégies IAM](#).

Une stratégie AWS IoT ressemble à ce qui suit :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/bar"] }, { "Effect": "Allow", "Action": ["iot:Connect"], "Resource": ["*"] } ] }
```

Cette stratégie autorise la personne habilitée à se connecter et publier les messages dans AWS IoT.

Gestion des stratégies AWS IoT

AWS IoT prend en charge les stratégies nommées de sorte que plusieurs identités puissent référencer le même document de stratégie. Les stratégies nommées comptent plusieurs versions afin de faciliter leur restauration.

Actions de stratégie AWS IoT

Les actions suivantes sont disponibles pour utilisation avec AWS IoT :

iot:Publish

L'autorisation de publier est vérifiée chaque fois qu'une demande de publication est envoyée à l'agent. L'action de publication est utilisée pour permettre aux clients de publier des modèles de rubriques spécifiques.

iot:Subscribe

L'autorisation de s'abonner est vérifiée chaque fois qu'une demande d'abonnement est envoyée à l'agent. L'action d'abonnement est utilisée pour permettre aux clients de s'abonner aux rubriques qui correspondent à des modèles de rubriques spécifiques.

iot:Receive

L'autorisation de réception est vérifiée chaque fois qu'un message est remis à un client. L'autorisation de réception étant vérifiée à chaque remise, elle peut être utilisée pour révoquer les autorisations pour les clients abonnés à une rubrique.

iot:Connect

L'autorisation de connexion est vérifiée chaque fois qu'une demande de connexion est envoyée à l'agent. L'agent de messages n'autorise pas deux clients avec le même ID client à rester connectés en même temps. Lorsque le second client se connecte, l'agent le détecte et déconnecte un des clients. L'autorisation de connexion peut être utilisée pour s'assurer que seuls les clients autorisés peuvent se connecter à l'aide d'un ID de client spécifique.

iot:UpdateThingShadow

L'autorisation UpdateThingShadow est vérifiée chaque fois qu'une demande est faite pour mettre à jour l'état d'un document thing shadow.

iot:GetThingShadow

L'autorisation GetThingShadow est vérifiée chaque fois qu'une demande est faite pour obtenir l'état d'un document thing shadow.

iot>DeleteThingShadow

L'autorisation DeleteThingShadow est vérifiée chaque fois qu'une demande est faite de supprimer le document thing shadow.

Ressources d'action

Le tableau suivant présente la ressource à spécifier pour chaque type d'action :

Action	Ressource
iot:DeleteThingShadow	ARN d'objet
iot:Connect	ARN d'ID client
iot:Publish	ARN de rubrique
iot:Subscribe	ARN de filtre de rubriques
iot:Receive	ARN de rubrique
iot:UpdateThingShadow	ARN d'objet
iot:GetThingShadow	ARN d'objet

Variables de stratégie AWS IoT

AWS IoT définit deux variables de stratégie qui peuvent être utilisées dans les stratégies AWS IoT : `iot:ClientId` and `aws:SourceIp`. Lorsqu'une stratégie est évaluée, les variables sont remplacées par les valeurs réelles. `iot:ClientId` est remplacée par l'ID du client qui a envoyé un message MQTT ou HTTP. `aws:SourceIp` est remplacée par l'adresse IP à l'origine du message.

La stratégie AWS IoT suivante illustre l'utilisation de variables de stratégie :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["iot:Connect"],
      "Resource": [ "arn:aws:iot:us-east-1:123451234510:client/${iot:ClientId}" ]
    },
    {
      "Effect": "Allow",
      "Action": ["iot:Publish"],
      "Resource": [ "arn:aws:iot:us-east-1:123451234510:topic/foo/bar/${iot:ClientId}" ]
    }
  ]
}
```

Lorsque vous utilisez des variables de stratégie telles que \${iot:ClientId}, vous pouvez ouvrir par inadvertance l'accès aux rubriques que vous ne souhaitez pas rendre accessibles. Par exemple, si vous utilisez une stratégie qui utilise \${iot:ClientId} pour spécifier un filtre de rubrique :

```
{ "Effect": "Allow", "Action": ["iot:Subscribe"], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/${iot:ClientId}/bar" ] }
```

Un client peut se connecter à l'aide de + comme ID de client. Cela permettrait à l'utilisateur de s'abonner à n'importe quelle rubrique correspondant à foo/+bar. Pour assurer une protection contre ces failles de sécurité, utilisez l'action de stratégie iot:Connect afin de contrôler les ID client qui peuvent se connecter. Exemples :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Connect"], "Resource": [ "arn:aws:iot:us-east-1:123456789012:client/clientid1" ] } ] }
```

Cette stratégie autorise uniquement les clients dont l'ID client est clientid1 à se connecter.

Exemples de stratégies

Les stratégies AWS IoT sont spécifiées dans un document JSON. Ce sont les composants d'une stratégie AWS IoT :

Version

Doit avoir la valeur « 2012-10-17 ».

Effect

Doit avoir la valeur « Autoriser » ou « Refuser ».

Action

Doit avoir la valeur « iot:<nom de l'opération> » où <nom de l'opération> a l'une des valeurs suivantes :

« iot:Publish » - publication MQTT.

« iot:Subscribe » - abonnement MQTT.

« iot:UpdateThingShadow » - mettre à jour un thing shadow.

« iot:GetThingShadow » - récupère un thing shadow.

« iot>DeleteThingShadow » - supprime un thing shadow.

Ressource

Doit avoir l'une des valeurs suivantes :

Client - arn:aws:iot :<région>:<accountId>:client/<clientId>

Rubrique ARN - arn:aws:iot :<région>:<accountId>:topic/<topicName>

Filtre de rubrique ARN - arn:aws:iot :<région>:<accountId>:topicfilter/<topicFilter>

Exemples de stratégies de connexion

La stratégie suivante autorise un ensemble d'ID client à se connecter :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:client/clientid1", "arn:aws:iot:us-east-1:123456789012:client/clientid2", 
```

```
"arn:aws:iot:us-east-1:123456789012:client/clientid3" ] }, { "Effect": "Allow", "Action": [ "iot:Publish", "iot:Subscribe", "iot:Receive" ], "Resource": [ "*" ] } }
```

La stratégie suivante empêche un ensemble d'ID client de se connecter :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Deny", "Action": [ "iot:Connect" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:client/clientid1", "arn:aws:iot:us-east-1:123456789012:client/clientid2" ] }, { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] } ] }
```

La stratégie suivante autorise le titulaire du certificat à s'abonner à un filtre de rubrique avec n'importe quel ID client `foo/*`:

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Subscribe" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/*" ] } ] }
```

Exemples de stratégie de publication/abonnement

La stratégie que vous utilisez dépend de la façon dont vous vous connectez à AWS IoT. Vous pouvez vous connecter à AWS IoT via un client MQTT, HTTP ou WebSocket. Lorsque vous vous connectez à un client MQTT, vous allez vous authentifier avec un certificat X.509. Lorsque vous vous connectez via HTTP ou le protocole WebSocket, vous vous authentifiez avec signature version 4 et Amazon Cognito.

Stratégies pour les clients MQTT

La stratégie suivante autorise le titulaire du certificat à publier dans toutes les rubriques et à s'abonner à tous les filtres de rubrique du compte AWS avec n'importe quel ID client :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:publish" ], "Resource": [ "*" ] } ] }
```

La stratégie suivante autorise le détenteur du certificat à publier dans toutes les rubriques du compte AWS avec n'importe quel ID client :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Publish", "iot:Connect" ], "Resource": [ "*" ] } ] }
```

La stratégie suivante autorise le détenteur du certificat à publier avec n'importe quel ID client dans les rubriques `foo/bar` and `foo/baz` :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Publish" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo/bar", "arn:aws:iot:us-east-1:123456789012:topic/foo/baz" ] } ] }
```

La stratégie suivante empêche le titulaire du certificat de publier avec n'importe quel ID client dans la rubrique `foo/bar` :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Deny", "Action": [ "iot:Publish" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo/bar" ] } ] }
```

```
[ "iot:Publish" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo/bar" ] } }
```

La stratégie suivante autorise le titulaire du certificat à s'abonner à un filtre de rubrique avec n'importe quel ID client `foo/+/bar`:

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Subscribe" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/+/bar" ] } ] }
```

La stratégie suivante autorise le détenteur du certificat à publier avec n'importe quel ID client dans la rubrique `foo` et vous abonner au filtre de rubrique `foo/bar/*`:

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Publish" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo" ] }, { "Effect": "Allow", "Action": [ "iot:Subscribe" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar/*" ] } ] }
```

La stratégie suivante autorise le détenteur du certificat à publier avec n'importe quel ID client dans la rubrique `foo` et empêche le titulaire du certificat de publier avec n'importe quel ID client dans la rubrique `bar`:

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Publish" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo" ] }, { "Effect": "Deny", "Action": [ "iot:Publish" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/bar" ] } ] }
```

La stratégie suivante autorise le titulaire du certificat à s'abonner à un filtre de rubrique avec n'importe quel ID client `foo/bar`:

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Subscribe" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar" ] } ] }
```

La stratégie suivante autorise le détenteur du certificat à publier avec n'importe quel ID client dans la rubrique `arn:aws:iot:us-east-1:123456789012:topic/iotmonitor/provisioning/8050373158915119971` et autorise le titulaire du certificat à s'abonner au filtre de rubrique avec n'importe quel ID client `arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/provisioning/8050373158915119971`:

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Publish", "iot:Receive" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/iotmonitor/provisioning/8050373158915119971" ] }, { "Effect": "Allow", "Action": [ "iot:Subscribe" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topicfilter/iotmonitor/provisioning/8050373158915119971" ] } ] }
```

Stratégies pour les clients HTTP et WebSocket

Pour les opérations suivantes, AWS IoT utilise des stratégies attachées à des identités Amazon Cognito (via l'API `AttachPrincipalPolicy`) pour définir les autorisations attachées au pool d'identités

Amazon Cognito avec les identités authentifiées. Cela signifie qu'une identité Amazon Cognito nécessite des autorisations à partir de la stratégie de rôle attachée au pool et de la stratégie attachée à l'identité Amazon Cognito via AWS IoT `AttachPrincipalPolicy` API.

- `iot:Connect`
- `iot:Publish`
- `iot:Subscribe`
- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot>DeleteThingShadow`



Note

Pour d'autres opérations AWS IoT ou des identités non authentifiées, AWS IoT ne définit pas les autorisations attachées au rôle de pool d'identités Amazon Cognito. Pour les identités authentifiées et non authentifiées, c'est la stratégie la plus permissive que nous vous recommandons d'attacher au rôle de pool Amazon Cognito.

Pour autoriser les identités Amazon Cognito non authentifiés à publier des messages sur HTTP dans n'importe quelle rubrique, attachez la stratégie suivante au rôle de pool d'identités Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Connect", "iot:Publish", "iot:Subscribe", "iot:Receive", "iot:GetThingShadow", "iot:UpdateThingShadow", "iot>DeleteThingShadow" ], "Resource": ["*"] } ] }
```

Pour autoriser les identités Amazon Cognito non authentifiés à publier des messages MQTT via HTTP dans n'importe quelle rubrique de votre compte, attachez la stratégie suivante pour le rôle de pool d'identités Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["*"] } ] }
```



Note

Cet exemple est donné à titre d'illustration uniquement. À moins que votre service en ait absolument besoin, nous recommandons d'utiliser une stratégie plus restrictive, qui ne permet pas aux identités Amazon Cognito non authentifiés de publier dans une rubrique quelconque.

Pour autoriser les identités Amazon Cognito non authentifiés à publier des messages MQTT sur HTTP dans la rubrique `topic1` de votre compte, attachez la stratégie suivante à votre rôle de pool d'identités Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"] } ] }
```

Pour qu'une identité Amazon Cognito authentifiée publie des messages MQTT sur HTTP dans la rubrique `topic1` de votre compte AWS, vous devez spécifier deux stratégies, comme indiqué ici. La première stratégie doit être attaché à un rôle de pool d'identités Amazon Cognito et autoriser les identités de ce pool d'effectuer un appel de publication. La deuxième stratégie est attachée à un

utilisateur Amazon Cognito à l'aide de l'API AWS IoT [AttachPrincipalPolicy](#), elle permet à l'utilisateur Amazon Cognito spécifié d'accéder à la rubrique `topic1`.

Stratégie de pool d'identités Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"] } ] }
```

Stratégie d'utilisateur Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"] } ] }
```

De même, la stratégie d'exemple suivante permet à l'utilisateur Amazon Cognito de publier des messages MQTT sur HTTP dans les rubriques `topic1` et `topic2`. Deux stratégies sont requises. La première donne au rôle de pool d'identités Amazon Cognito la possibilité d'effectuer l'appel de publication. La deuxième stratégie donne à l'utilisateur Amazon Cognito accès aux rubriques `topic1` and `topic2`.

Stratégie de pool d'identités Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["*"] } ] }
```

Stratégie d'utilisateur Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/topic1", "arn:aws:iot:us-east-1:123456789012:topic/topic2" ] } ] }
```

Les stratégies suivantes autorisent plusieurs utilisateurs Amazon Cognito à publier une rubrique. Deux stratégies par identité Amazon Cognito sont requises. La première donne au rôle de pool d'identités Amazon Cognito la possibilité d'effectuer l'appel de publication. Les deuxième et troisième donnent aux utilisateurs Amazon Cognito accès aux rubriques `topic1` and `topic2`, respectivement.

Stratégie de pool d'identités Amazon Cognito :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["*"] } ] }
```

Stratégie d'utilisateur Amazon Cognito 1 :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic1"] } ] }
```

Stratégie d'utilisateur Amazon Cognito 2 :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/topic2"] } ] }
```

Exemples de stratégies de réception

La stratégie suivante empêche le titulaire du certificat de recevoir des messages à partir d'une rubrique avec n'importe quel ID client :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Deny", "Action": [ "iot:Receive" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo/restricted" ] }, { "Effect": "Allow", "Action": [ "iot: *" ], "Resource": [ "*" ] } ] }
```

La stratégie suivante autorise le titulaire du certificat à s'abonner et à recevoir des messages dans une rubrique avec n'importe quel ID client :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Subscribe" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topicfilter/foo/bar" ] }, { "Effect": "Allow", "Action": [ "iot:Receive" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo/bar" ] } ] }
```

Stratégies IoT IAM

AWS IoT propose un ensemble de templates de stratégie IAM que vous pouvez utiliser telles quelles ou comme point de départ pour la création de stratégies IAM personnalisées. Ces templates permettent d'accéder à des opérations de configuration et sur les données. Les opérations de configuration vous permettent de créer des objets, des certificats, des stratégies et des règles. Les opérations sur les données envoient des données via les protocoles MQTT ou HTTP. Le tableau suivant décrit ces templates.

Template de stratégie	Description
AWSIoTLogging	Permet à l'identité associée de configurer la journalisation CloudWatch. Cette stratégie est attachée à votre rôle de journalisation CloudWatch.
AWSIoTConfigAccess	Autorise l'identité associée à accéder à toutes les opérations de configuration AWS IoT.
AWSIoTConfigReadOnlyAccess	Autorise l'identité associée à appeler les opérations de configuration en lecture seule.
AWSIoTDataAccess	Autorise à l'identité associée un accès complet à toutes les opérations sur les données AWS IoT. Les opérations sur les données envoient des données via les protocoles MQTT ou HTTP. Lorsque MQTT via le protocole WebSocket est utilisé, seules les stratégies stockées dans IAM sont appliquées à la connexion WebSocket.
AWSIoTFullAccess	Autorise aux identités associées l'accès complet à toutes les opérations de configuration et sur les données AWS IoT.
AWSIoTRuleActions	Autorise l'identité associée à accéder à tous les services AWS pris en charge dans les actions de règle AWS IoT.

Accès entre comptes

AWS IoT permet d'autoriser un utilisateur habilité à publier ou à s'abonner à une rubrique définie dans un compte AWS qui n'est pas sa propriété. Vous configurez l'accès entre comptes en créant une stratégie IAM et un rôle IAM, puis en attachant la stratégie au rôle.

Tout d'abord, créez une stratégie IAM comme vous le feriez pour d'autres utilisateurs et certificats de votre compte AWS. Par exemple, la stratégie suivante accorde des autorisations de connexion et de publication à la rubrique /foo/bar .

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Connect" ], "Resource": [ "*" ] }, { "Effect": "Allow", "Action": [ "iot:Publish" ], "Resource": [ "arn:aws:iot:us-east-1:123456789012:topic/foo/bar" ] } ] }
```

Ensuite, suivez les étapes de [Création d'un rôle pour un utilisateur IAM](#). Saisissez l'ID du compte AWS avec lequel vous souhaitez partager l'accès. Puis, dans la dernière étape, attachez la stratégie que vous venez de créer au rôle. Si, ultérieurement, vous avez besoin de modifier l'ID de compte AWS auquel vous accordez l'accès, vous pouvez utiliser le format de stratégie d'approbation suivant à cet effet.

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": "arn:aws:iam:us-east-1:111111111111:user/MyUser" }, "Action": "sts:AssumeRole" } ] }
```

Sécurité du transport

L'agent de messages AWS IoT et le service Thing Shadows chiffrent toutes les communications avec TLS. TLS est utilisé afin de garantir la confidentialité des protocoles d'application (MQTT, HTTP) pris en charge par AWS IoT. TLS est disponible dans un certain nombre de langages de programmation et de systèmes d'exploitation.

Pour MQTT, TLS chiffre la connexion entre le dispositif et l'agent. L'authentification de client TLS est utilisée par AWS IoT pour identifier les dispositifs. Pour HTTP, TLS chiffre la connexion entre le dispositif et l'agent. L'authentification est déléguée à AWS Signature Version 4.

Prise en charge des suites de chiffrement TLS

AWS IoT prend en charge les suites de chiffrement suivantes :

- ECDHE-ECDSA-AES128-GCM-SHA256 (recommandée)
- ECDHE-RSA-AES128-GCM-SHA256 (recommandée)
- ECDHE-ECDSA-AES128-SHA256
- ECDHE-RSA-AES128-SHA256
- ECDHE-ECDSA-AES128-SHA
- ECDHE-RSA-AES128-SHA
- ECDHE-ECDSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA

- ECDHE-ECDSA-AES256-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA
- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA

Agent de messages pour AWS IoT

L'agent de messages AWS IoT est un service d'agent de publication/abonnement qui permet l'envoi et la réception de messages vers et depuis AWS IoT. Lors de la communication avec AWS IoT, un client envoie un message adressé à une rubrique comme « Sensor/temp/room1 ». L'agent de messages, envoie à son tour le message à tous les clients qui se sont abonnés pour recevoir des messages concernant cette rubrique. L'acte d'envoi du message s'appelle publication. L'acte d'inscription pour recevoir des messages pour un filtre de rubrique s'appelle abonnement.

L'espace de noms de rubrique est isolé pour chaque paire de compte et région AWS. Par exemple, la rubrique `Sensor/temp/room1` d'un compte AWS est indépendante de la rubrique « `Sensor/temp/room1` » d'un autre compte AWS. C'est également vrai des régions. La rubrique « `Sensor/temp/room1` » du même compte AWS dans us-east-1 est indépendante de la même rubrique dans us-west-2. AWS IoT ne prend pas en charge l'envoi et la réception de messages dans les comptes et les régions AWS.

L'agent de messages conserve une liste de toutes les sessions et les abonnements des clients pour chaque session. Lorsqu'un message est publié dans une rubrique, l'agent recherche les sessions abonnées à cette rubrique. Il transmet ensuite le message de publication à toutes les sessions ayant un client actuellement connecté.

Protocoles

L'agent de messages prend en charge l'utilisation du protocole MQTT pour la publication et l'abonnement et le protocole HTTPS pour la publication. Les deux protocoles sont pris en charge via IP version 4 et IP version 6. L'agent de messages prend également en charge MQTT via le protocole WebSocket.

MQTT

MQTT est un protocole de messagerie léger largement utilisé, destiné aux dispositifs limités. Pour plus d'informations, consultez la section concernant [MQTT](#). Bien que l'implémentation de l'agent de messages AWS IoT soit basée sur MQTT version 3.1.1, il s'éloigne de la spécification comme suit :

- Dans AWS IoT, l'abonnement à une rubrique avec une qualité de Service (QoS) 0 signifie qu'un message est remis zéro fois ou plus. Un message peut être remis plusieurs fois. Les messages

remis plusieurs fois peuvent être envoyés avec un ID de paquet différent. Dans ce cas, l'indicateur DUP n'est pas défini.

- AWS IoT, ne prend pas en charge la publication et l'abonnement avec QoS 2. L'agent de messages AWS IoT, n'envoie pas de PUBACK ou SUBACK lorsque QoS 2 est demandée.
- Les niveaux de qualité de service pour la publication et l'abonnement à une rubrique n'ont aucun lien entre eux. Un client peut s'abonner à une rubrique avec QoS1 alors qu'un autre peut publier dans la même rubrique avec QoS0.
- Lorsqu'il répond à une demande de connexion, l'agent de messages envoie un message CONNACK. Ce message contient un indicateur précisant si la connexion reprend une session précédente. La valeur de cet indicateur peut être incorrecte si deux clients MQTT se connectent avec le même ID de client simultanément.
- Lorsqu'un client s'abonne à une rubrique, il peut y avoir un délai entre le moment où l'agent de messages envoie un SUBACK et le moment où le client commence à recevoir de nouveaux messages correspondants.
- La spécification MQTT fournit une clause permettant à un éditeur de demander à l'agent de conserver le dernier message envoyé à une rubrique et de l'envoyer à tous les abonnés à venir de cette rubrique. AWS IoT ne prend pas en charge les messages conservés. Si une demande est faite de conserver les messages, la connexion est interrompue.
- L'agent de messages utilise l'ID de client pour identifier chaque client. L'ID de client est transmis depuis le client à l'agent de messages dans le cadre de la charge utile MQTT. Deux clients possédant le même ID de client ne sont pas autorisés à être connectés simultanément à l'agent de messages. Lorsqu'un client se connecte à l'agent de messages à l'aide d'un ID de client qu'un autre client utilise, un message CONNACK est envoyé aux deux clients et le client connecté est déconnecté.
- L'agent de messages ne pas prend en charge les sessions permanentes (session valide définie à 0). Toutes les sessions sont supposées valides et les messages ne sont pas stockés dans les sessions. Si un client MQTT envoie un message avec l'attribut de session valide défini à la valeur false, il est déconnecté.
- À de rares occasions, l'agent de messages peut renvoyer le même message de publication logique avec un ID de paquet différent.
- L'agent de messages ne garantit pas l'ordre dans lequel les messages et les ACK sont reçus.

HTTP

L'agent de messages prend en charge les clients qui se connectent avec le protocole HTTP à l'aide d'une API REST. Les clients peuvent publier en envoyant un message POST à `<AWS IoT Endpoint>/topics/<url_nom_rubrique_codé>?qos=1`.

MQTT via le protocole WebSocket

AWS IoT prend en charge MQTT sur le protocole [WebSocket](#) pour permettre aux applications basées sur un navigateur et à distance d'envoyer et de recevoir des données à partir d'appareils connectés à AWS IoT en utilisant les informations d'identification AWS. Les informations d'identification AWS sont spécifiées avec AWS signature version 4. Pour plus d'informations, consultez [AWS Signature Version 4](#). Le support WebSocket est disponible sur le port tcp:443, qui permet aux messages de passer à travers la plupart des pare-feux et proxys Web.

Une connexion WebSocket est lancée sur un client en envoyant une demande HTTP GET. L'URL que vous utilisez a la forme suivante :

```
wss://<endpoint>.iot.<region>.amazonaws.com/mqtt
```

wss

Spécifie le protocole WebSocket.

endpoint

Le point de terminaison AWS IoT spécifique de votre compte AWS. Vous pouvez utiliser la commande de CLI AWS IoT, [describe-endpoint](#) pour trouver ce point de terminaison.

région

Région AWS de votre compte AWS.

mqtt

Spécifie que vous enverrez des messages MQTT via le protocole WebSocket.

Lorsque le serveur répond, le client envoie une demande de mise à niveau pour indiquer au serveur qu'il va communiquer avec le protocole WebSocket. Une fois que le serveur reconnaît la demande de mise à niveau, toutes les communications sont effectuées à l'aide du protocole WebSocket. L'implémentation de WebSocket que vous utilisez fait office de protocole de transport. Les données que vous envoyez via le protocole WebSocket sont des messages MQTT.

Utilisation du protocole WebSocket dans une application Web

L'implémentation de WebSocket fournie par la plupart des navigateurs Web n'autorise pas la modification des en-têtes HTTP, vous devez ajouter les informations de signature version 4 à la chaîne de requête. Pour plus d'informations, consultez [Ajout d'informations de signature à la chaîne de requête](#).

Le code JavaScript suivant définit certaines fonctions d'utilitaire utilisées pour générer une demande signature version 4.

```
/** * utilities to do sigv4 * @class SigV4Utils */ function
SigV4Utils(){} SigV4Utils.sign = function(key, msg){ var hash =
CryptoJS.HmacSHA256(msg, key); return hash.toString(CryptoJS.enc.Hex); };
SigV4Utils.sha256 = function(msg) { var hash = CryptoJS.SHA256(msg);
return hash.toString(CryptoJS.enc.Hex); }; SigV4Utils.getSignatureKey
= function(key, dateStamp, regionName, serviceName) { var
kDate = CryptoJS.HmacSHA256(dateStamp, 'AWS4' + key); var
kRegion = CryptoJS.HmacSHA256(regionName, kDate); var kService
= CryptoJS.HmacSHA256(serviceName, kRegion); var kSigning =
CryptoJS.HmacSHA256('aws4_request', kService); return kSigning; };
```

Pour créer une demande Signature Version 4

1. Créer une demande canonique pour Signature Version 4.

Le code JavaScript suivant crée une demande canonique :

```
var time = moment.utc(); var dateStamp = time.format('YYYYMMDD');
var amzdate = dateStamp + 'T' + time.format('HHmmss') + 'Z'; var
service = 'iotdevicegateway'; var region = this.options.regionName;
var secretKey = this.options.secretKey; var accessKey =
this.options.accessKey; var algorithm = 'AWS4-HMAC-SHA256';
var method = 'GET'; var canonicalUri = '/mqtt'; var host =
this.options.endpoint; var credentialScope = dateStamp + '/' + region
+ '/' + service + '/' + 'aws4_request'; var canonicalQueryString
= 'X-Amz-Algorithm=AWS4-HMAC-SHA256'; canonicalQueryString +=
'&X-Amz-Credential=' + encodeURIComponent(accessKey + '/' +
credentialScope); canonicalQueryString += '&X-Amz-Date=' + amzdate;
canonicalQueryString += '&X-Amz-SignedHeaders=host'; var canonicalHeaders
= 'host:' + host + '\n'; var payloadHash = SigV4Utils.sha256('');
```

```
var canonicalRequest = method + '\n' + canonicalUri + '\n' +  
canonicalQueryString + '\n' + canonicalHeaders + '\nhost\n' +  
payloadHash; console.log('canonicalRequest ' + canonicalRequest);
```

- Créez une chaîne de connexion, générez une clé de signature et signez la chaîne.

Prenez l'URL canonique que vous avez créé à l'étape précédente et assemblez-la en une chaîne de connexion. Pour ce faire, créez une chaîne composée de l'algorithme de hachage, la date, la portée des informations d'identification et le SHA de la demande canonique. Ensuite, générez la clé de signature et signez la chaîne, comme illustré dans le code JavaScript suivant.

```
var stringToSign = algorithm + '\n' + amzdate + '\n' + credentialScope  
+ '\n' + SigV4Utils.sha256(canonicalRequest); var signingKey =  
SigV4Utils.getSignatureKey(secretKey, dateStamp, region, service); var  
signature = SigV4Utils.sign(signingKey, stringToSign);
```

- Ajoutez les informations de signature à la demande.

Le code JavaScript suivant montre comment ajouter les informations de signature à la chaîne de requête.

```
canonicalQueryString += '&X-Amz-Signature=' + signature; var  
requestUrl = 'wss://' + host + canonicalUri + '?' + canonicalQueryString;
```

- Si vous disposez d'informations d'identification de session (à partir d'un serveur STS, AssumeRole ou Amazon Cognito), ajoutez le jeton de session à la fin de la chaîne d'URL après la signature :

```
requestUrl += "&X-Amz-Security-Token=" + encodeURIComponent(sessionToken);
```

- Ouvrez le WebSocket.

Le code JavaScript suivant montre comment créer un client Paho MQTT et appeler CONNECT à AWS IoT. L'argument `endpoint` correspond à votre point de terminaison spécifiques du compte AWS. Le `clientId` est un identificateur de texte qui est unique parmi tous les clients connectés en même temps à votre compte AWS.

```
var client = new Paho.MQTT.Client(requestUrl, clientId); var  
connectOptions = { onSuccess: function(){ // connect succeeded }, useSSL:  
true, timeout: 3, mqttVersion: 4, onFailure: function() { // connect  
failed } }; client.connect(connectOptions);
```

Utilisation du protocole WebSocket dans une application mobile

Nous vous conseillons d'utiliser l'un des SDK d'appareil AWS IoT pour connecter votre dispositif à AWS IoT quand vous établissez une connexion WebSocket. Les SDK d'appareil AWS IoT suivant prennent en charge les connexions MQTT-basées sur WebSocket à AWS IoT :

- [node.js](#)
- [iOS](#)

- [Android](#)

Vous trouverez ici une implémentation de référence pour la connexion d'une application Web à AWS IoT via MQTT sur le protocole WebSocket : [Exemple de WebSocket Labs AWS](#).

Si vous utilisez un langage de programmation ou de script qui n'est pas pris en charge actuellement, toute bibliothèque WebSocket existante peut être utilisée tant que la demande de mise à niveau WebSocket initiale (HTTP POST) est connectée à l'aide d'AWS Signature Version 4. Certains clients MQTT, tels que [Eclipse Paho for JavaScript](#), prennent en charge le protocole WebSocket en mode natif.

Rubriques

L'agent de messages utilise des rubriques pour acheminer les messages des clients publiant aux clients abonnées. La barre oblique (/) est utilisée pour séparer la hiérarchie des rubriques. Les tableaux suivants répertorient les caractères génériques pouvant être utilisés dans le filtre de rubriques lorsque vous vous abonnez.

Caractères génériques de rubrique

Caractère générique	Description
#	Doit être le dernier caractère dans la rubrique à laquelle vous vous abonnez. Fonctionne comme un caractère générique correspondant à l'arborescence actuelle et à toutes les sous-arborescences. Par exemple, un abonnement à Sensor/# reçoit des messages publiés dans Sensor/, Sensor/temp, Sensor/temp/room1, mais pas les messages publiés dans « Sensor ».
+	Apparie exactement un élément dans la hiérarchie de rubriques. Par exemple, un abonnement à Sensor/+/room1 reçoit des messages publiés dans Sensor/temp/room1, Sensor/moisture/room1, etc.

Rubriques réservées

Toutes les rubriques commençant par « \$ » sont considérées comme réservées et ne sont pas prises en charge pour la publication et l'abonnement, sauf si elles sont utilisées avec le service Thing Shadows. Pour plus d'informations, consultez [Thing Shadows](#).

Evènements du cycle de vie

AWS IoT publie des événements de cycle de vie dans les rubriques MQTT présentées dans les sections suivantes. Ces messages vous permettent d'être informé des événements de cycle de vie de l'agent de messages.



Note

Des messages de cycle de vie peuvent être envoyés dans le désordre et vous pouvez recevoir des messages en double.

Stratégie requise pour recevoir des événements de cycle de vie

Voici un exemple de stratégie requise pour recevoir des événements de cycle de vie :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
  "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource":  
  ["arn:aws:iot:région:compte:topicfilter/$aws/events/*" ] } ] }
```

Événements de connexion/déconnexion

AWS IoT publie un message dans les rubriques MQTT suivantes lorsqu'un client se connecte ou se déconnecte :

```
$aws/events/presence/connected/clientId
```

ou

```
$aws/events/presence/disconnected/clientId
```

Où **clientId** est l'ID du client MQTT qui se connecte ou se déconnecte de l'agent de messages AWS IoT.

Le message publié sur cette rubrique a la structure suivante :

```
{ "clientId": "a1b2c3d4e5f6a7b8c9d0elf2a3b4c5d6", "timestamp":  
1460065214626, "eventType": "connected", "sessionIdentifier":  
"00000000-0000-0000-0000-000000000000", "principalIdentifier":  
"000000000000/ABCDEFGHIJKLMNPQRSTUVWXYZ:some-user/ABCDEFGHIJKLMNPQRSTUVWXYZ:some-  
user" }
```

Voici une liste d'éléments JSON qui sont contenus dans les messages de connexion/déconnexion publiés dans la rubrique `$aws/events/presence/connected/clientId`.

clientId

ID du client qui se connecte ou se déconnecte.



Note

Les ID des clients qui contiennent des # ou + ne reçoivent pas les événements de cycle de vie.

eventType

Type d'événement. Les valeurs valides sont `connected` ou `disconnected`.

principalIdentifier

Informations d'identification utilisées pour l'authentification. Pour les certificats d'authentification mutuelle TLS, il s'agit de l'ID du certificat. Pour les autres connexions, ce sont les informations d'identification IAM.

sessionIdentifier

Identifiant globalement unique dans AWS IoT qui existe pendant la durée de vie de la session.

timestamp

Approximation du moment où l'événement s'est produit, exprimée en millisecondes en heure Unix. La précision de l'horodatage est de +/- 2 minutes.

Événements d'abonnement/désabonnement

AWS IoT publie un message dans la rubrique MQTT suivante lorsqu'un client s'abonne ou se désabonne à une rubrique MQTT :

```
$aws/events/subscriptions/subscribed/clientId
```

ou

```
$aws/events/subscriptions/unsubscribed/clientId
```

Où *clientId* est l'ID du client MQTT qui se connecte à l'agent de messages AWS IoT.

Le message publié sur cette rubrique a la structure suivante :

```
{ "clientId": "186b5", "timestamp": 1460065214626, "eventType": "subscribed" | "unsubscribed", "sessionIdentifier": "00000000-0000-0000-0000-000000000000", "principalIdentifier": "000000000000/ABCDEFGHIJKLMNPQRSTUVWXYZ:some-user/ABCDEFGHIJKLMNPQRSTUVWXYZ:some-user" "topics" : [ "foo/bar", "device/data", "dog/cat" ] }
```

Voici une liste d'éléments JSON qui sont contenus dans les messages d'abonnement/désabonnement publiés dans les rubriques `$aws/events/subscriptions/subscribed/clientId` and `$aws/events/subscriptions/unsubscribed/clientId`.

clientId

ID du client qui s'abonne ou se désabonne.



Note

Les ID des clients qui contiennent des # ou + ne reçoivent pas les événements de cycle de vie.

eventType

Type d'événement. Les valeurs valides sont `subscribed` ou `unsubscribed`.

principalIdentifier

Informations d'identification utilisées pour l'authentification. Pour les certificats d'authentification mutuelle TLS, il s'agit de l'ID du certificat. Pour les autres connexions, ce sont les informations d'identification IAM.

sessionIdentifier

Identifiant globalement unique dans AWS IoT qui existe pendant la durée de vie de la session.

timestamp

Approximation du moment où l'événement s'est produit, exprimée en millisecondes en heure Unix.

La précision de l'horodatage est de +/- 2 minutes.

topics

Tableau des sujets MQTT auquel le client s'est abonné.



Note

Les messages de cycle de vie peuvent être envoyés dans le désordre. Vous pouvez recevoir des messages en double.

Règles pour AWS IoT

Les règles offrent à vos dispositifs la possibilité d'interagir avec les services AWS. Les règles sont analysées et les actions exécutées en fonction du flux de rubrique MQTT. Vous pouvez utiliser des règles pour prendre en charge des tâches telles que celles-ci :

- Augmenter ou filtrer les données reçues d'un dispositif.
- Écrire des données provenant d'un dispositif dans une base de données Amazon DynamoDB.
- Enregistrer un fichier dans Amazon S3.
- Envoyer une notification push à tous les utilisateurs à l'aide de Amazon SNS.
- Publier des données dans une file d'attente Amazon SQS.
- Appeler une fonction Lambda pour extraire des données.
- Traiter les messages provenant d'un grand nombre de dispositifs à l'aide de Amazon Kinesis.
- Envoyer des données à Amazon Elasticsearch Service.
- Saisir une métrique CloudWatch.
- Modifier une alarme CloudWatch.
- Envoyer les données à partir d'un message MQTT à Amazon Machine Learning pour faire des prédictions basées sur un template Amazon ML.

Pour que AWS IoT puisse effectuer ces actions, vous devez lui accorder l'autorisation d'accéder à vos ressources AWS en votre nom. Lorsque les actions sont effectuées, vous payez les frais standards pour les services AWS que vous utilisez.

Table des matières

- [Attribuer à AWS IoT l'accès requis \(p. 122\)](#)
- [Transmettre les autorisations de rôle \(p. 123\)](#)
- [Création d'une règle AWS IoT \(p. 123\)](#)
- [Affichage des règles \(p. 125\)](#)
- [Versions de SQL \(p. 125\)](#)
- [Résolution des problèmes d'une règle \(p. 127\)](#)
- [Suppression d'une règle \(p. 127\)](#)
- [Actions de règle AWS IoT \(p. 127\)](#)

- Référence d'SQL AWS IoT (p. 135)

Attribuer à AWS IoT l'accès requis

Les rôles IAM permettent de contrôler les ressources AWS à laquelle chaque règle a accès. Avant de créer une règle, vous devez créer un rôle IAM avec une stratégie qui autorise l'accès aux ressources AWS requises. AWS IoT assume ce rôle lors de l'exécution d'une règle.

Pour créer un rôle IAM (AWS CLI)

1. Enregistrez le document de stratégie d'approbation suivant, qui accorde à AWS IoT l'autorisation d'assumer le rôle, dans un fichier appelé iot-role-trust.json :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "iot.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }
```

Utilisez la commande [create-role](#) pour créer un rôle IAM en spécifiant le fichier iot-role-trust.json :

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document file://iot-role-trust.json
```

Le résultat de cette commande va ressembler à ce qui suit :

```
{ "Role": { "AssumeRolePolicyDocument": "url-encoded-json", "RoleId": "AKIAIOSFODNN7EXAMPLE", "CreateDate": "2015-09-30T18:43:32.821Z", "RoleName": "my-iot-role", "Path": "/", "Arn": "arn:aws:iam::123456789012:role/my-iot-role" } }
```

2. Enregistrez le code JSON suivant dans un fichier nommé iot-policy.json.

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "dynamodb:*", "Resource": "*" } ] }
```

Ce JSON est un exemple de document de stratégie qui accorde à AWS IoT un accès administrateur à DynamoDB.

Utilisez la commande [create-policy](#) pour accorder à AWS IoT l'accès à vos ressources AWS en assumant le rôle, en transmettant le fichier iot-policy.json :

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy-document.json
```

Pour plus d'informations sur la façon d'accorder l'accès aux services AWS dans les stratégies pour AWS IoT, consultez la page [Création d'une règle AWS IoT \(p. 123\)](#).

Le résultat de la commande [create-policy](#) contient l'ARN de la stratégie. Vous devez attacher la stratégie à un rôle.

```
{ "Policy": { "PolicyName": "my-iot-policy", "CreateDate": "2015-09-30T19:31:18.620Z", "AttachmentCount": 0, "IsAttachable": true, "PolicyId": "ZXR6A36LTYANPAI7NJ5UV", "DefaultVersionId": "v1", "Path": "/", "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy", "UpdateDate": "2015-09-30T19:31:18.620Z" } }
```

3. Utilisez la commande [attach-role-policy](#) pour attacher votre stratégie à votre rôle :

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn  
"arn:aws:iam::123456789012:policy/my-iot-policy"
```

Transmettre les autorisations de rôle

Lorsque vous créez ou remplacez une règle, vous devez transmettre un rôle qui contrôle les ressources AWS à laquelle la règle a accès. Le rôle doit être défini dans le même compte AWS en tant que règle. Le moteur de règles AWS IoT vérifie que vous détenez l'autorisation `iam:PassRole` de transmettre le rôle à l'API `create-topic-rule`. Pour vous assurer que vous bénéficiez de cet accès, vous devez créer une stratégie qui accorde cet accès et l'attacher à votre utilisateur IAM. La stratégie suivante montre comment accorder l'autorisation `iam:PassRole` à un rôle.

```
{ "Version": "2012-10-17", "Statement": [ { "Sid": "Stmt1",  
"Effect": "Allow", "Action": [ "iam:PassRole" ], "Resource":  
[ "arn:aws:iam::123456789012:role/myRole" ] } ] }
```

Dans cet exemple de stratégie, l'autorisation `iam:PassRole` est accordée pour le rôle `myRole`. Le rôle a été spécifié à l'aide de son ARN. Vous devez également attacher cette stratégie à votre utilisateur IAM ou au rôle auquel appartient votre utilisateur. Pour plus d'informations sur l'utilisation de ces stratégies, consultez [Utilisation des stratégies gérées](#).



Note

Les fonctions Lambda utilisent des stratégies basées sur les ressources, où la stratégie est attachée directement à la fonction Lambda même. Lorsque vous créez une règle qui appelle une fonction Lambda, vous ne transmettez pas un rôle, donc l'utilisateur qui crée la règle n'a pas besoin de l'autorisation `iam:PassRole`. Pour plus d'informations sur l'autorisation de fonctions Lambda, consultez la rubrique [Accord d'autorisations à l'aide d'une stratégie de ressources](#).

Création d'une règle AWS IoT

Vous configurez les règles d'acheminement des données provenant de vos objets connectés. Règles se composent de ce qui suit :

Nom de la règle

 Nom de la règle.

Description facultative

 Description textuelle de la règle.

Instruction SQL

 Syntaxe SQL simplifiée pour filtrer les messages reçus dans une rubrique MQTT et pousser les données ailleurs. Pour plus d'informations, consultez [Référence d'SQL AWS IoT \(p. 135\)](#).

Version de SQL

 Version du moteur de règles SQL à utiliser lors de l'évaluation de la règle. Même si cette propriété est facultative, nous vous recommandons vivement de préciser la version de SQL. Si cette propriété n'est pas définie, la valeur par défaut, 2015-10-08, sera utilisée.

Une ou plusieurs actions

 Actions effectuées par AWS IoT lors de l'exécution de la règle. Par exemple, vous pouvez insérer des données dans une table DynamoDB, écrire des données dans un compartiment Amazon S3, publier dans une rubrique Amazon SNS ou appeler une fonction Lambda.

Lorsque vous créez une règle, soyez conscient de la quantité de données que vous publiez dans des rubriques. Si vous créez des règles qui incluent un modèle de rubrique de caractère générique, elles peuvent correspondre à un pourcentage élevé de vos messages, et vous devrez peut-être augmenter la capacité des ressources AWS utilisées par les actions cibles. En outre, si vous créez une règle de republication qui inclut un modèle de rubrique de caractère générique, vous pouvez vous retrouver avec une règle circulaire qui tourne en boucle à l'infini.



Note

La création et la mise à jour de règles sont des actions de niveau administrateur. Tout utilisateur détenant des autorisations de création ou de mise à jour de règles pourra accéder aux données traitées par les règles.

Pour créer une règle (AWS CLI)

Utilisez la commande [create-topic-rule](#) pour créer une règle :

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://my-rule.json
```

Voici un exemple de fichier de charge utile avec une règle qui insère tous les messages envoyés à la rubrique iot/test dans la table DynamoDB spécifiée. L'instruction SQL filtre les messages et le rôle ARN accorde à AWS IoT autorisation d'écrire dans la table DynamoDB.

```
{ "sql": "SELECT * FROM 'iot/test'", "ruleDisabled": false, "awsIotSqlVersion": "2016-03-23-beta", "actions": [ { "dynamoDB": { "tableName": "my-dynamodb-table", "roleArn": "arn:aws:iam::123456789012:role/my-iot-role", "hashKeyField": "topic", "hashKeyValue": "${topic(2)}", "rangeKeyField": "timestamp", "rangeKeyValue": "${timestamp()}" } ] }
```

Voici un exemple de fichier de charge utile avec une règle qui insère tous les messages envoyés à la rubrique iot/test dans le compartiment S3 spécifié. L'instruction SQL filtre les messages et le rôle ARN accorde à AWS IoT autorisation d'écrire dans le compartiment Amazon S3.

```
{ "rule": { "awsIotSqlVersion": "2016-03-23-beta", "sql": "SELECT * FROM 'iot/test'", "ruleDisabled": false, "actions": [ { "s3": { "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3", "bucketName": "my-bucket", "key": "myS3Key" } ] }, "ruleName": "MyS3Rule" } }
```

Voici un exemple de fichier de charge utile avec une règle qui publie des données dans Amazon ES :

```
{ "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'", "ruleDisabled": false, "awsIotSqlVersion": "2016-03-23-beta", "actions": [ { "elasticsearch": { "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es", "endpoint": "https://my-endpoint", "index": "my-index", "type": "my-type", "id": "${newuuid()}" } ] }
```

Voici un exemple de fichier de charge utile avec une règle qui appelle une fonction Lambda :

```
{ "sql": "expression", "ruleDisabled": false, "awsIotSqlVersion": "2016-03-23-beta", "actions": [ { "lambda": { "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function" } } ] }
```

Voici un exemple de fichier de charge utile avec une règle qui publie dans une rubrique Amazon SNS :

```
{ "sql": "expression", "ruleDisabled": false, "awsIotSqlVersion": "2016-03-23-beta", "actions": [ { "sns": { "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic", "roleArn": "arn:aws:iam::123456789012:role/my-iot-role" } } ] }
```

Voici un exemple de fichier de charge utile avec une règle qui permet de republier dans une autre rubrique MQTT :

```
{ "sql": "expression", "ruleDisabled": false, "awsIotSqlVersion": "2016-03-23-beta", "actions": [ { "republish": { "topic": "my-mqtt-topic", "roleArn": "arn:aws:iam::123456789012:role/my-iot-role" } } ] }
```

Voici un exemple de fichier de charge utile avec une règle qui publie des données dans un flux Amazon Kinesis Firehose :

```
{ "sql": "SELECT * FROM 'my-topic'", "ruleDisabled": false, "awsIotSqlVersion": "2016-03-23-beta", "actions": [ { "firehose": { "roleArn": "arn:aws:iam::123456789012:role/my-iot-role", "deliveryStreamName": "my-stream-name" } } ] }
```

Voici un exemple de fichier de charge utile avec une règle qui utilise la fonction machinelearning_predict de Amazon Machine Learning pour republier dans une rubrique, si les données dans la charge utile MQTT sont classées comme un 1.

```
{ "sql": "SELECT * FROM 'iot/test' where machinelearning_predict('my-model', 'arn:aws:iam::123456789012:role/my-iot-aml-role', *).predictedLabel=1", "ruleDisabled": false, "awsIotSqlVersion": "2016-03-23-beta", "actions": [ { "republish": { "roleArn": "arn:aws:iam::123456789012:role/my-iot-role", "topic": "my-mqtt-topic" } } ] }
```

Affichage des règles

Utilisez la commande `list-topic-rules` pour répertorier vos règles :

```
aws iot list-topic-rules
```

Utilisez la commande `get-topic-rule` pour obtenir des informations sur une règle :

```
aws iot get-topic-rule --rule-name my-rule
```

Versions de SQL

Le moteur de règles AWS IoT utilise une syntaxe de type SQL pour sélectionner les données dans des messages MQTT. Les instructions SQL sont interprétées d'après une version SQL spécifiée avec la propriété `awsIotSqlVersion` dans un document JSON qui décrit la règle. Pour plus d'informations sur la structure des documents de règle JSON, consultez la page [Création d'une règle \(p. 123\)](#). La propriété `awsIotSqlVersion` permet de spécifier la version du moteur de règles SQL AWS IoT que vous voulez utiliser. Lorsqu'une nouvelle version est déployée, vous pouvez continuer à utiliser

une version antérieure ou modifier votre règle pour utiliser la nouvelle version. Vos règles actuelles continueront à utiliser la version avec laquelle elles ont été créées.

L'exemple de code JSON suivant montre comment spécifier la version SQL à l'aide la propriété awsIoTSqlVersion :

```
{ "sql": "expression", "ruleDisabled": false, "awsIoTSqlVersion": "2016-03-23-beta", "actions": [ { "republish": { "topic": "my-mqtt-topic", "roleArn": "arn:aws:iam::123456789012:role/my-iot-role" } } ] }
```

Les versions prises en charge sont :

- 2015-10-08, la version SQL d'origine créée le 8 août 2015 (2015-10-08).
- 2016-03-23-beta, la version de SQL créée le 23 mars 2016 (2016-03-23).
- version bêta, la version bêta de SQM la plus récente. L'utilisation de cette version peut entraîner des modifications de décomposition dans vos règles.

Nouveautés de la version 2016-03-23-beta du moteur de règles SQL

- Correctifs pour sélectionner les objets JSON imbriqués.
- Correctifs pour les requêtes de tableaux.
- Prise en charge des requêtes inter-objet.
- Prise en charge de la génération en sortie d'un tableau comme un objet de niveau supérieur.
- Ajoute la fonction encode(valeur, encodingScheme), qui peut être appliquée sur les données aux formats JSON et autres.

Requêtes inter-objet

Cette fonction permet d'exécuter une requête pour un attribut d'un objet JSON. Par exemple, avec le message MQTT suivant :

```
{ "e": [ { "n": "temperature", "u": "Cel", "t": 1234, "v": 22.5 }, { "n": "light", "u": "lm", "t": 1235, "v": 135 }, { "n": "acidity", "u": "pH", "t": 1235, "v": 7 } ] }
```

Et la règle suivante :

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

La règle génère la sortie suivante :

```
{"temperature": [{"v": 22.5}]}
```

Avec le même message MQTT et une règle un peu plus compliquée telle que :

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'),1).v as temperature FROM 'topic'
```

La règle génère la sortie suivante :

```
{ "temperature":22.5 }
```

Générer un tableau en sortie comme un objet de niveau supérieur

Cette fonction permet à une règle de retourner un tableau comme un objet de niveau supérieur. Par exemple, avec le message MQTT suivant :

```
{ "a": { "b": "c" }, "arr":[1,2,3,4] }
```

Et la règle suivante :

```
SELECT VALUE arr FROM 'topic'
```

La règle génère la sortie suivante :

```
[1,2,3,4]
```

Encoder la fonction

Encode la charge utile, qui peut être constituée de données non JSON, dans sa représentation de chaîne basée sur le schéma de codage spécifié.

Résolution des problèmes d'une règle

Si vous rencontrez un problème avec vos règles, vous devez activer CloudWatch Logs. En analysant vos journaux, vous pouvez déterminer si le problème relève de l'autorisation ou si, par exemple, une condition de clause WHERE n'a pas été respectée. Pour plus d'informations, consultez [Résolution des problèmes de AWS IoT \(p. 218\)](#).

Suppression d'une règle

Lorsque vous avez terminé avec une règle, vous pouvez la supprimer.

Pour supprimer une règle (AWS CLI)

Utilisez la commande `delete-topic-rule` pour supprimer une règle :

```
aws iot delete-topic-rule --rule-name my-rule
```

Actions de règle AWS IoT

Les actions de règle AWS IoT sont utilisées pour spécifier que faire lorsqu'une règle est déclenchée. Vous pouvez définir les actions pour écrire des données dans une base de données DynamoDB ou un flux Amazon Kinesis ,ou pour appeler une fonction Lambda et plus encore. Les actions suivantes sont prises en charge :

- `cloudwatchAlarm` pour modifier une alarme CloudWatch.
- `cloudwatchMetric` pour saisir une métrique CloudWatch.
- `dynamoDB` pour écrire des données dans une base de données DynamoDB.
- `elasticsearch` pour écrire des données dans un domaine Amazon Elasticsearch Service.
- `kinesis` pour écrire des données dans un flux Amazon Kinesis.
- `lambda` pour appeler une fonction Lambda.
- `s3` pour écrire des données dans un compartiment Amazon S3.
- `sns` pour écrire des données dans une notification push.
- `firehose` pour écrire des données dans un flux Amazon Kinesis Firehose.
- `sqs` pour écrire des données dans une file d'attente SQS.
- `republish` pour republier le message dans une autre rubrique MQTT.

Les sections suivantes décrivent chaque action en détail.

Action d'alarme CloudWatch

L'action d'alarme CloudWatch permet de modifier l'état d'alarme de CloudWatch. Vous pouvez spécifier la raison du changement d'état et la valeur dans cet appel. Lorsque vous créez une règle AWS IoT avec une alarme d'action CloudWatch, vous devez spécifier les informations suivantes :

`roleArn`

Rôle IAM qui autorise l'accès à l'alarme CloudWatch.

`alarmName`

Nom de l'alarme CloudWatch.

`stateReason`

Raisons de la modification de l'alarme.

`stateValue`

Valeur de l'état de l'alarme. Les valeurs acceptables sont `OK`, `ALARME`, `INSUFFICIENT_DATA`.



Note

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation `cloudwatch:SetAlarmState`.

L'exemple JSON suivant montre comment définir une action d'alarme CloudWatch dans une règle AWS IoT :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "cloudwatchAlarm": { "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw", "alarmName": "IotAlarm", "stateReason": "Temperature stabilized.", "stateValue": "OK" } } ] } }
```

Pour plus d'informations, consultez [Alarmes CloudWatch](#).

Action de métrique CloudWatch

L'action de métrique CloudWatch permet de saisir une métrique CloudWatch. Vous pouvez spécifier le namespace, le nom, la valeur, l'unité et l'horodatage de la métrique. Lorsque vous créez une règle AWS IoT avec une alarme de métrique CloudWatch, vous devez spécifier les informations suivantes :

`roleArn`

Rôle IAM qui autorise l'accès à l'alarme CloudWatch.

metricNamespace
nom du namespace de la métrique CloudWatch.

metricName
Nom de la métrique CloudWatch.

metricValue
Valeur de la métrique CloudWatch.

metricUnit
Unité de métrique pris en charge par CloudWatch.

metricTimestamp
Horodatage Unix facultatif.



Note

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation `cloudwatch:PutMetricData`.

L'exemple JSON suivant montre comment définir une action de métrique CloudWatch dans une règle AWS IoT :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "cloudwatchMetric": { "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw", "metricNamespace": "IoTNamespace", "metricName": "IoTMetric", "metricValue": "1", "metricUnit": "Count", "metricTimestamp": "1456821314" } ] } }
```

Pour plus d'informations, consultez [Métriques CloudWatch](#).

Action DynamoDB

L'action `dynamoDB` permet d'écrire tout ou partie d'un message MQTT dans une table DynamoDB. Lorsque vous créez une règle DynamoDB, vous devez spécifier les informations suivantes :

hashKeyType
Type de données de la clé de hachage (également appelée clé de partition). Les valeurs valides sont : "CHAÎNE" ou "NUMBER".

hashKeyField
Nom de la clé de hachage (également appelée clé de partition).

hashKeyValue
Valeur de la clé de hachage.

rangeKeyType
Facultatif. Type de données de la clé de plage (également appelée clé de tri). Les valeurs valides sont : "CHAÎNE" ou "NUMBER".

rangeKeyField
Facultatif. Nom de la clé de plage (également appelée clé de tri).

rangeKeyValue
Facultatif. Valeur de la clé de plage.

fonctionnement
Facultatif. Type d'opération à effectuer. Elle suit le template de substitution et peut donc être `${fonctionnement}`, mais la substitution doit avoir pour résultat l'un des éléments suivants : INSERT, UPDATE, OU DELETE.

payloadField
Facultatif. Nom du champ où la charge utile sera écrite. Si cette valeur n'est pas spécifiée, la charge utile est renseignée dans le champ `payload`.

table

Nom de la table DynamoDB.

roleARN

Rôle IAM qui autorise l'accès à la table DynamoDB. Au minimum, le rôle doit autoriser l'action IAM dynamoDB:PutItem .

Les données écrites dans la table DynamoDB sont le résultat de l'instruction SQL de la règle. Les champs *hashKeyValue* et *rangeKeyValue* sont généralement composés d'expressions régulières (par exemple, « \${topic()} » ou « \${timestamp()} »).



Note

Les données non-JSON sont écrites dans DynamoDB en tant que données binaires. La console DynamoDB présente les données sous la forme de texte codé en Base64.

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation dynamodb:PutItem .

L'exemple JSON suivant montre comment définir une action `dynamoDB` dans une règle AWS IoT :

```
{ "rule": { "ruleDisabled": false, "sql": "SELECT * AS message FROM 'some/topic'", "description": "A test Dynamo DB rule", "actions": [ { "dynamoDB": { "hashKeyField": "key", "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB", "tableName": "my_ddb_table", "hashKeyValue": "${topic()}", "rangeKeyValue": "${timestamp()}" } } ] } }
```

Pour plus d'informations, consultez la [Guide de démarrage de Amazon DynamoDB](#).

L'action Amazon ES

Le `elasticsearch` permet d'écrire des données à partir de messages MQTT dans un domaine Amazon Elasticsearch Service. Les données d'Amazon ES peuvent ensuite être interrogées et visualisées à l'aide d'outils tels que Kibana. Lorsque vous créez une règle AWS IoT avec une action `elasticsearch`, vous devez spécifier les informations suivantes :

endpoint

Point de terminaison de votre domaine Amazon ES.

index

Index Amazon ES dans lequel vous souhaitez stocker vos données.

type

Type de document que vous stockez.

id

Identifiant unique de chaque document.



Note

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation es:ESHttpPut .

L'exemple JSON suivant montre comment définir une action `elasticsearch` dans une règle AWS IoT :

```
{ "rule": { "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'", "ruleDisabled": false, "actions": [ { "elasticsearch": {} } ] } }
```

```
{ "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es",
  "endpoint": "https://my-endpoint", "index": "my-index", "type": "my-type",
  "id": "${newuuid()}" } ] }
```

Pour plus d'informations, consultez la [Manuel du développeur Amazon ES](#).

Action Kinesis

L'action `kinesis` permet d'écrire des données à partir de messages MQTT dans un flux Amazon Kinesis. Lorsque vous créez une règle AWS IoT avec une action `kinesis`, vous devez spécifier les informations suivantes :

`stream`

Flux dans lequel Amazon Kinesis écrire les données.

`partitionKey`

Clé de partition utilisée pour déterminer dans quelle partition les données sont écrites. La clé de partition est généralement composée d'une expression (par exemple, « \${topic()} » ou « \${timestamp()} »).



Note

Assurez-vous que la stratégie associée à la règle détient l'autorisation `Kinesis:PutRecord`.

L'exemple JSON suivant montre comment définir une action `kinesis` dans une règle AWS IoT :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false,
  "actions": [ { "kinesis": { "roleArn": "arn:aws:iam::123456789012:role/
aws_iot_kinesis", "streamName": "my_kinesis_stream", "partitionKey":
"${topic()}" } } ], } }
```

Pour plus d'informations, consultez la [Manuel du développeur Amazon Kinesis](#).

Action lambda

Une action `lambda` appelle une fonction Lambda, en transmettant le message MQTT qui a déclenché la règle. Pour que AWS IoT appelle une fonction Lambda, vous devez configurer une stratégie accordant l'autorisation `lambda:InvokeFunction` à AWS IoT. Les fonctions Lambda utilisent des stratégies basées sur les ressources, vous devez attacher la stratégie à la fonction Lambda même. Utilisez la commande CLI suivante pour attacher une stratégie accordant l'autorisation `lambda:InvokeFunction` :

```
aws lambda add-permission --function-name "function_name" --region
  "region" --principal iot.amazonaws.com --source-arn arn:aws:iot:us-
  east-1:account_id:rule/rule_name --source-account "account_id" --statement-id
  "unique_id" --action "lambda:InvokeFunction"
```

Les paramètres suivants sont ceux de la commande `add-permission` :

`--function-name`

Nom de la fonction Lambda dont vous mettez à jour la stratégie de ressource en ajoutant une nouvelle autorisation.

`--region`

Région AWS de votre compte.

--principal

Mandataire qui obtient l'autorisation. Il doit être `iot.amazonaws.com` pour octroyer à AWS IoT l'autorisation d'appeler une fonction Lambda.

--source-arn

ARN de la règle. Vous pouvez utiliser la commande de CLI `get-topic-rule` pour obtenir l'ARN d'une règle.

--source-account

Compte AWS où la règle est définie.

--statement-id

Identifiant unique de l'instruction.

--action

Action Lambda que vous voulez autoriser dans cette instruction. Dans ce cas, nous voulons autoriser AWS IoT à appeler une fonction Lambda, afin de préciser `lambda:InvokeFunction`.

Pour plus d'informations, consultez [Modèle d'autorisation Lambda](#).

Lorsque vous créez une règle avec une action `lambda`, vous devez spécifier la fonction Lambda à appeler lorsque la règle est déclenchée.

L'exemple JSON suivant présente une règle qui appelle une fonction Lambda :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "lambda": { "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:myLambdaFunction" } } ] } }
```

Pour plus d'informations, consultez la [Manuel du développeur AWS Lambda](#).

Action d'S3

Une action `s3` écrit les données provenant du message MQTT qui déclenche la règle dans un compartiment Amazon S3. Lorsque vous créez une règle AWS IoT avec une action `s3`, vous devez spécifier les informations suivantes :

bucket

Compartiment Amazon S3 dans lequel écrire les données.

key

Chemin d'accès au fichier dans lequel les données sont écrites. Par exemple, si la valeur de ce paramètre est « `${topic()}/${timestamp()}` », la rubrique à laquelle le message a été envoyée est « `this/is/my/topic` », et l'horodatage en cours est 1460685389, les données seront écrit dans un fichier appelé « `1460685389` » dans le dossier « `this/is/my/topic` » dans Amazon S3.



Note

L'utilisation d'une clé statique entraîne le remplacement d'un seul fichier dans Amazon S3 pour chaque appel de la règle. Les cas d'utilisation les plus courants sont l'utilisation d'un horodatage de message ou d'un autre identifiant de message unique, de sorte qu'un nouveau fichier sera enregistré dans Amazon S3 pour chaque message reçu.

roleArn

Rôle IAM qui autorise l'accès au compartiment Amazon S3.



Note

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation `s3:PutObject`.

L'exemple JSON suivant montre comment définir une action `s3` dans une règle AWS IoT :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "s3": { "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3", "bucketName": "my-bucket", "key": "${topic()}/ ${timestamp()}" } } ] } }
```

Pour plus d'informations, consultez la [Guide du développeur Amazon S3](#).

Action d'SNS

Une action `sns` envoie les données du message MQTT qui déclenche la règle comme une notification push SNS. Lorsque vous créez une règle avec une action `sns`, vous devez spécifier les informations suivantes :

messageFormat

Format du message. Les valeurs acceptées sont « JSON » et « RAW ». La valeur par défaut de l'attribut est « RAW ». SNS utilise ce paramètre pour déterminer si la charge utile doit être analysée et les parties pertinentes de la charge utile spécifiques à la plateforme doivent être extraites.

roleArn

Rôle IAM qui autorise l'accès à l'option SNS.

targetArn

Rubrique SNS ou dispositif individuel auxquels les notifications push seront envoyées.



Note

Vérifiez que la stratégie associée à la règle détient l'autorisation `sns:Publish`.

L'exemple JSON suivant montre comment définir une action `sns` dans une règle AWS IoT :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "sns": { "targetArn": "arn:aws:sns:us-east-1:123456789012:my sns topic", "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns" } } ] } }
```

Pour plus d'informations, consultez la [Guide du développeur Amazon SNS](#).

Action Firehose

Une action `firehose` envoie les données à partir d'un message MQTT qui déclenche la règle à un flux Firehose. Lorsque vous créez une règle avec une action `firehose`, vous devez spécifier les informations suivantes :

deliveryStreamName

Flux Firehose dans lequel écrire les données du message.

roleArn

Rôle IAM qui autorise l'accès à Firehose.

separator

Séparateur de caractères qui sera utilisé pour séparer les enregistrements écrits dans le flux firehose. Les valeurs valides sont : « \n » (nouvelle ligne), « \t » (tabulation), « \r\n » (nouvelle ligne Windows), « , » (virgule).



Note

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation `Firehose:PutRecord`.

L'exemple JSON suivant montre comment créer une règle AWS IoT avec une action `firehose` :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "firehose": { "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose", "deliveryStreamName": "my_firehose_stream" } } ] } }
```

Pour plus d'informations, consultez la [Guide du développeur Firehose](#).

Action d'SQS

A `sqs` envoie les données à partir d'un message MQTT qui a déclenché la règle dans une file d'attente SQS. Lorsque vous créez une règle avec une action `sqs`, vous devez spécifier les informations suivantes :

`queueUrl`

URL de la file d'attente SQS dans laquelle écrire les données.

`useBase64`

Définissez cette option à la valeur `true` si vous souhaitez que les données du message MQTT soient codées en Base64 avant d'écrire dans la file d'attente SQS. Sinon, attribuez-lui la valeur `false`.

`roleArn`

Rôle IAM qui autorise l'accès à la file d'attente SQS.



Note

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation `sqs:SendMessage`.

L'exemple JSON suivant montre comment créer une règle AWS IoT avec une action `sqs` :

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "sqs": { "queueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/my_sqs_queue", "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs", "useBase64": false } } ] } }
```

Pour plus d'informations, consultez la [Guide du développeur Amazon SQS](#).

Action de republication

L'action `republish` permet de republier le message qui a déclenché le rôle dans une autre rubrique MQTT. Lorsque vous créez une règle avec une action `republish`, vous devez spécifier les informations suivantes :

`topic`

Rubrique MQTT dans laquelle republier le message.

`roleArn`

Rôle IAM qui permet de publier dans la rubrique MQTT.



Note

Vérifiez que le rôle associé à la règle a une stratégie accordant l'autorisation iot:Publish .

```
{ "rule": { "sql": "SELECT * FROM 'some/topic'", "ruleDisabled": false, "actions": [ { "republish": { "topic": "another/topic", "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish" } } ] } }
```

Référence d'sQL AWS IoT

Cette référence se concentre sur les différences entre le SQL ANSI SQL et le SQL AWS IoT. Si vous n'êtes pas familiarisé avec le SQL ANSI, consultez [le didacticiel SQL W3Schools](#).

Toutes les règles comprennent une instruction SQL qui se compose d'une clause SELECT et d'une clause WHERE facultative. La clause SELECT permet d'extraire un ou plusieurs objets JSON ou les attributs de la charge du message MQTT. La clause WHERE permet de filtrer les objets JSON ou les attributs extraits par la clause SELECT.

Toutes les données traitées par une requête SELECT sont supposés être au format JSON, à moins que certaines fonctions soient utilisées dans la requête. Si les données sont au format JSON, elles peuvent contenir un objet racine unique ou un objet racine unique avec des objets imbriqués. Pour plus d'informations sur les fonctions applicables aux données non JSON, consultez la page [Fonctions SQL \(p. 138\)](#).

Comme avec le SQL ANSI, les espaces blancs n'ont pas d'importance et les mots clés ne sont pas sensibles à la casse. Les chaînes et les propriétés JSON sont sensibles à la casse. Dans nos exemples, tous les mots-clés sont en capitale, ce qui est une pratique courante en SQL.

Expressions

Les clauses Select, FROM et WHERE sont toutes composées d'expressions régulières. Les expressions suivantes sont autorisées dans AWS IoT.

Jeton	Sens	exemple
=	Égalité, comparaison	Couleur = « rouge »
<>	Différence, comparaison	Couleur <> « rouge »
AND	ET logique	Couleur = « rouge » AND sirène = « oui »
OU	OU logique	Couleur = « rouge » OR sirène = « oui »
()	Parenthèse, regroupement	Couleur = « rouge » AND (sirène = « oui » OR isTest)
\$	Signe du dollar, utilisé dans les noms de rubrique réservées. Lorsque vous spécifiez un	"\$\$aws/things/mything/shadow/update/accepted"

Jeton	Sens	exemple
	nom de rubrique réservée dans un template de substitution, le « \$» doit être placé dans une séquence d'échappement avec un autre « \$»	
+	Addition, calcul	4 + 5
-	Soustraction, calcul	5-4
/	Division, calcul	20 / 4
*	Multiplication, calcul	5 * 4
%	Modulo, calcul	20 % 6
<	Inférieur à, comparaison	5 < 6
<=	Inférieur ou égal à, comparaison	5 <= 6
>	Supérieur à, comparaison	6 > 5
>=	Supérieur ou égal à, comparaison	6 >= 5
Appel de fonction	Appel d'une fonction SQL	clientId()
Expression d'extension JSON	Expression qui sélectionne une valeur spécifique dans un document JSON.	state.desired.color
CASE ... WHEN ... THEN ... ELSE ... END	Instruction Case	CASE location WHEN 'home' THEN 'off' WHEN 'work' THEN 'on' ELSE 'silent' END

Clause SELECT

La clause SELECT AWS IoT est essentiellement la même que la clause SELECT SQL ANSI, avec quelques différences mineures. Le mot clé « AS » est requis avec les clauses SELECT AWS IoT, sauf si vous utilisez les extensions SON (la syntaxe '.'), par exemple :

```
SELECT state.temperature FROM 'mydevices/device1'
```

Pour plus d'informations concernant les extensions JSON, consultez [Extensions JSON \(p. 142\)](#).

À la différence de l'interrogation des bases de données relationnelles, vous pouvez utiliser `SELECT * sans risque de récupérer trop d'informations.` Les informations retournées par la clause `SELECT * AS some_name` contiennent uniquement la charge utile JSON du message MQTT. Le message peut être retourné sans analyse, ce qui en fin de compte améliore les performances de la requête. Les requêtes `SELECT` pour un document JSON d'objet unique prennent la forme : `SELECT <object> AS <object-name>`. L'`<objet>` peut être n'importe quel objet ou nom d'attribut dans le document JSON. `<object-name>` fournit un nom pour le résultat de la requête `SELECT`. Les requêtes `SELECT` pour des documents JSON avec des objets imbriqués prennent la forme : `SELECT <object>, <object> correspondant à l'objet racine, tout niveau d'objets imbriqués et un attribut facultatif.` Exemples :

- `object`
- `object.nestedObject`
- `object.nestedObject1.nestedObject2`
- `object.nestedObject1.nestedObject2.attribute`

Vous pouvez spécifier autant de couches d'objets imbriqués qu'il y en a dans le document JSON.

Clause FROM

En SQL ANSI, vous sélectionnez des données dans des tables. En SQL AWS IoT, vous sélectionnez des données dans des propriétés JSON des messages MQTT.

La source de données, qui est la rubrique à laquelle les messages MQTT sont envoyés, a été spécifiée à l'aide de la fonction `MQTT()`, comme illustré dans cet exemple :

```
SELECT * FROM mqtt('com.example/sensors/+')
```

Toutefois, la clause `FROM` suppose que vous interrogez un message MQTT, vous pouvez ignorer l'appel à la fonction `mqtt()` et spécifier uniquement le nom de la rubrique, comme illustré dans l'exemple suivant :

```
SELECT * FROM 'com.example/sensors'
```

Clause WHERE

La clause `WHERE` filtre les données du message retourné par la clause `SELECT` selon les valeurs d'attributs JSON. Par exemple, supposons que vous avez un filtre rubrique à partir d'une rubrique MQTT, `iot/thing/#`. Voici un exemple JSON de charge utile qui peut être publié par un dispositif :

```
{ "deviceid" : "iot123", "temp" : 54.98, "humidity" : 32.43, "coords" : { "latitude" : 47.615694, "longitude" : -122.3359976 } }
```

Vous pouvez utiliser l'instruction SQL dans votre règle pour interroger la rubrique `iot/thing/#` et extraire les données des capteurs lorsque la valeur du champ `temp` est supérieure à 50.

```
SELECT * FROM 'iot/thing/#' WHERE temp > 50
```

Fonctions

Vous pouvez utiliser les fonctions intégrées suivantes dans les clauses SELECT ou WHERE de vos expressions SQL.

Fonction	Description	Version de SQL
abs(number)	Retourne la valeur absolue.	2015-10-08 et ultérieure.
accountId()	Retourne l'ID de compte du client MQTT envoyant le message, ou une valeur non définie si le message n'est pas arrivé via MQTT.	2015-10-08 et ultérieure.
asin(number)	Renvoie l'arc sinus.	2015-10-08 et ultérieure.
atan(number)	Renvoie l'arc tangent.	2015-10-08 et ultérieure.
bitand(number1, number2)	Renvoie le résultat d'une opération AND au niveau du bit.	2015-10-08 et ultérieure.
cast(value as type)	<p>Convertit la valeur du type de données spécifié. Les types de données pris en charge sont :</p> <ul style="list-style-type: none"> double Convertit la valeur en un nombre double. float Convertit la valeur en un nombre double. int Convertit la valeur en un nombre entier. integer Convertit la valeur en un nombre entier. ntext Convertit la valeur en une chaîne. num Convertit la valeur en un nombre double. number Convertit la valeur en un nombre double. nvarchar Convertit la valeur en une chaîne. chaîne Convertit la valeur en une chaîne. text Convertit la valeur en une chaîne. varchar Convertit la valeur en une chaîne. 	2015-10-08 et ultérieure.

Fonction	Description	Version de SQL
ceil(number)	Renvoie le résultat de l'arrondi à l'entier supérieur le plus proche.	2015-10-08 et ultérieure.
chr(number)	Renvoie le caractère ASCII représenté par number.	2015-10-08 et ultérieure.
clientId()	Retourne l'ID de du client MQTT envoyant le message, ou une valeur non définie si le message n'est pas arrivé via MQTT.	2015-10-08 et ultérieure.
concat(string1, string2)	Renvoie la concaténation de deux chaînes.	2015-10-08 et ultérieure.
cos(number)	Renvoie le cosinus.	2015-10-08 et ultérieure.
cosh(number)	Renvoie le cosinus hyperbolique.	2015-10-08 et ultérieure.
encode(valeur, encodingScheme)	Encode la valeur d'après le schéma d'encodage fourni. La fonction renvoie une représentation de chaîne de la charge utile codée.	2016-03-23-bêta et ultérieure.
endswith(input, suffix)	Renvoie true si input se termine par suffix.	2015-10-08 et ultérieure.
exp(number)	Renvoie e à la puissance du nombre spécifié.	2015-10-08 et ultérieure.
floor(number)	Renvoie le résultat de l'arrondi à l'entier inférieur le plus proche.	2015-10-08 et ultérieure.
get_thing_shadow(thingName, roleArn)	Renvoie le thing shadow de l'objet spécifié. thingName Nom de l'objet dont vous souhaitez récupérer l'état. roleArn Rôle avec une autorisation iot:GetThingShadow .	2016-03-23-bêta et ultérieure.
ln(number)	Renvoie le logarithme naturel.	2015-10-08 et ultérieure.
log(n, m)	Renvoie le logarithme de n base m.	2015-10-08 et ultérieure.
lower(chaîne)	Renvoie le résultat de la conversion de tous les caractères en minuscules.	2015-10-08 et ultérieure.
lpad(chaîne, n)	Ajoute n espaces à gauche de la chaîne.	2015-10-08 et ultérieure.
ltrim(chaîne)	Supprime tous les espaces blancs à gauche de la chaîne.	2015-10-08 et ultérieure.
machinelearning_predict(model, roleArn, record)	Effectue une prévision par rapport au modèle spécifié, à l'ARN du rôle et à l'enregistrement.	2015-10-08 et ultérieure.
md2(chaîne)	Renvoie la valeur de hachage MD2.	2015-10-08 et ultérieure.
md5(chaîne)	Renvoie la valeur de hachage MD5.	2015-10-08 et ultérieure.
mod(m, n)	Renvoie le reste de m divisé par n.	2015-10-08 et ultérieure.

Fonction	Description	Version de SQL
nanvl(valeur, default)	Renvoie valeur si elle n'est pas nulle, default dans le cas contraire.	2015-10-08 et ultérieure.
power(m, n)	Renvoie m élevé à la puissance n.	2015-10-08 et ultérieure.
remainder(m, n)	Renvoie le reste de m divisé par n.	2015-10-08 et ultérieure.
replace(source, substring, replacement)	Renvoie source avec toutes les occurrences de substring remplacées par replacement.	2015-10-08 et ultérieure.
round(number, precision)	Renvoie le résultat de l'arrondi number sur precision décimales. Si precision est égale à 0, la fonction arrondit au nombre entier le plus proche.	2015-10-08 et ultérieure.
rpad(chaîne, n)	Ajoute n espaces à droite de la chaîne.	2015-10-08 et ultérieure.
rtrim(chaîne)	Supprime tous les espaces blancs à droite de la chaîne.	2015-10-08 et ultérieure.
sign(number)	Renvoie une valeur indiquant le signe d'un nombre. If number < 0, then -1. Else, if number = 0, then 0. Else, if number > 0, then 1.	2015-10-08 et ultérieure.
sin(number)	Renvoie le sinus.	2015-10-08 et ultérieure.
sinh(number)	Renvoie le sinus hyperbolique.	2015-10-08 et ultérieure.
sqrt(number)	Renvoie la racine carrée.	2015-10-08 et ultérieure.
startswith(input, prefix)	Renvoie true si input commence par prefix.	2015-10-08 et ultérieure.
tan(number)	Renvoie la tangente.	2015-10-08 et ultérieure.
tanh(number)	Renvoie la tangente hyperbolique.	2015-10-08 et ultérieure.
traceId()	Retourne l'ID de la trace du message MQTT, ou une valeur non définie si le message n'est pas arrivé via MQTT.	2015-10-08 et ultérieure.
topic(number)	Renvoie le segment de rubrique spécifié. Par exemple, si la rubrique est foo/bar, topic() renvoie « foo/bar », topic(1) renvoie « foo » et topic(2) « bar ».	2015-10-08 et ultérieure.
trunc(number, precision)	Renvoie le résultat de la troncation de number sur precision décimales.	2015-10-08 et ultérieure.
upper(chaîne)	Renvoie le résultat de la conversion de tous les caractères en majuscules.	2015-10-08 et ultérieure.
sha1(chaîne)	Renvoie la valeur de hachage SHA-1.	2015-10-08 et ultérieure.
sha224(chaîne)	Renvoie la valeur de hachage SHA-224.	2015-10-08 et ultérieure.
sha256(chaîne)	Renvoie la valeur de hachage SHA-256.	2015-10-08 et ultérieure.
sha512(chaîne)	Renvoie la valeur de hachage SHA-512.	2015-10-08 et ultérieure.

Fonction	Description	Version de SQL
rand()	Renvoie un nombre aléatoire entre 0 et 1.	2015-10-08 et ultérieure.
newuuid()	Retourne un UUID aléatoire de 16 octets.	2015-10-08 et ultérieure.
timestamp()	Renvoie l'horodatage Unix actuel, tel qu'observé par le serveur actuel.	2015-10-08 et ultérieure.

Faire des prévisions avec Amazon Machine Learning dans une règle AWS IoT

Utilisez la commande `machinelearning_predict` pour faire des prévisions en utilisant les données d'un message MQTT basé sur un modèle Amazon ML. Les paramètres de la fonction `machinelearning_predict` sont les suivants :

`modelId`

ID du modèle par rapport auquel exécuter la prévision. Le point de terminaison en temps réel du modèle doit être activé.

`roleArn`

Rôle IAM qui dispose d'une stratégie avec les autorisations `machinelearning_predict` et `machinelearning:GetMLModel` et autorise l'accès au modèle par rapport auquel la prévision est réalisée.

`record`

Données à transmettre dans l'API de prévision Amazon Machine Learning. Elles doivent être représentées sous la forme d'un objet JSON à couche unique. Si l'enregistrement est un objet JSON multiniveau, il est mis à plat en sérialisant ses valeurs. Par exemple, le code JSON suivant :

```
{ "key1": { "innerKey1": "value1" }, "key2": 0 }
```

deviendrait :

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0 }
```

La fonction renvoie un objet JSON dans les champs suivants :

`predictedLabel`

Classification de l'entrée basée sur le modèle.

`détails`

Contient les attributs suivants :

`PredictiveModelType`

Type de modèle. Les valeurs valides sont REGRESSION, BINARY, MULTICLASS.

`Algorithme`

Algorithme utilisé par Amazon Machine Learning pour faire des prévisions. La valeur doit être SGD.

`predictedScores`

Contient le score de classification brut correspondant à chaque étiquette.

`predictedValue`

Valeur prévue par Amazon Machine Learning.

Encode la charge utile avant un autre traitement

Utilisez la fonction `encode` pour coder la charge utile, qui peut être constituée de données non JSON, dans sa représentation de chaîne basée sur le schéma de codage.

valeur

Une des expressions valides, telles que définies dans la [Syntaxe SQL \(p. 135\)](#). En outre, vous pouvez spécifier `*` pour encoder la charge utile dans son ensemble, qu'elle soit ou non au format JSON. Si vous fournissez une expression, le résultat de l'évaluation sera converti en une chaîne avant d'être codé.

encodingScheme

Chaîne littérale qui représente le schéma de codage à utiliser. Actuellement, seul le codage « `base64` » est pris en charge.

Extensions JSON

Vous pouvez utiliser les extensions suivantes de la syntaxe SQL ANSI pour faciliter l'utilisation d'objets JSON imbriqués.

Opérateur `"."`

Cet opérateur accède aux membres dans les objets et les fonctions JSON imbriqués de la même manière qu'au SQL ANSI et à JavaScript.

Opérateur `*`

Fonctionne de la même manière que le caractère générique `*` dans le SQL ANSI. Il est utilisé dans la clause `SELECT` uniquement et crée un nouvel objet JSON contenant les données du message. Si la charge utile du message n'est pas au format JSON, `*` renvoie la charge utile du message entier sous la forme d'octets bruts.

Appliquer une fonction à une valeur d'attribut

Voici un exemple JSON de charge utile qui peut être publié par un dispositif :

```
{ "deviceid" : "iot123", "temp" : 54.98, "humidity" : 32.43, "coords" :  
  { "latitude" : 47.615694, "longitude" : -122.3359976 } }
```

L'exemple suivant applique une fonction à une valeur d'attribut dans une charge utile JSON :

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

Le résultat de cette requête est l'objet JSON suivant :

```
{ "temp": 54.98, "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1" }
```

Modèles de substitution

Vous pouvez utiliser un modèle de substitution pour alimenter les données JSON renvoyées lorsqu'une règle est déclenchée et que AWS IoT exécute une action. La syntaxe d'un modèle de substitution est `$(expression)`, où `expression` peut être toute expression prise en charge par AWS IoT dans des clauses `SELECT` ou `WHERE`. Pour plus d'informations concernant les expressions prises en charge, consultez la page [Expressions \(p. 135\)](#).

Les modèles de substitution apparaissent dans la clause SELECT au sein d'une règle, par exemple :

```
{ "sql": "SELECT *, topic() AS topic FROM 'my/iot/topic'", "ruleDisabled": false, "actions": [ { "republish": { "topic": "${topic()}", "roleArn": "arn:aws:iam::123456789012:role/my-iot-role" } } ] }
```

Si cette règle est déclenchée par le code JSON suivant :

```
{ "deviceid" : "iot123", "temp" : 54.98, "humidity" : 32.43, "coords" : { "latitude" : 47.615694, "longitude" : -122.3359976 }
```

Voici le résultat de la règle :

```
{ "coords":{ "longitude":-122.3359976, "latitude":47.615694 }, "humidity":32.43, "temp":54.98, "deviceid":"iot123", "topic":"my/iot/topic" }
```

Device Shadows pour AWS IoT

Un thing shadow (parfois appelé device shadow) est un document JSON utilisé pour stocker et récupérer des informations d'état actualisées concernant un objet (appareil, applications etc.). Le service chose Shadows conserve un thing shadow pour chaque objet que vous vous connectez à AWS IoT. Vous pouvez utiliser des thing shadows pour obtenir et définir l'état d'un objet sur MQTT ou HTTP, que l'objet soit connecté ou non à Internet. Chaque thing shadow est identifié par son nom.

Table des matières

- [Flux de données Device Shadows \(p. 144\)](#)
- [Documents Device Shadows \(p. 148\)](#)
- [Utilisation de Device Shadows \(p. 150\)](#)
- [API RESTful de Device Shadow \(p. 155\)](#)
- [Rubriques Device Shadow MQTT \(p. 157\)](#)
- [Syntaxe d'un document Device Shadow \(p. 162\)](#)
- [Messages d'erreur Thing Shadow \(p. 163\)](#)

Flux de données Device Shadows

Le service Device Shadows agit en tant qu'intermédiaire, ce qui permet aux dispositifs et aux applications de récupérer et de mettre à jour les thing shadows.

Afin d'illustrer la façon dont les dispositifs et applications communiquent avec le service Thing Shadows, cette section vous guide dans l'utilisation du client MQTT AWS IoT et de l'AWS CLI pour simuler la communication entre une lampe connectée à internet, une application et le service Thing Shadows.

Le service Thing Shadows utilise diverses rubriques MQTT pour faciliter la communication entre les applications et les dispositifs. Pour voir comment cela fonctionne, utilisez le client MQTT AWS IoT pour vous abonner aux rubriques MQTT suivantes avec une qualité de service 1 :

\$aws/things/myLightBulb/shadow/update/accepted

Le service Thing Shadows envoie des messages à cette rubrique lors de la réussite d'une mise à jour dans un thing shadow.

\$aws/things/myLightBulb/shadow/update/rejected

Le service Thing Shadows envoie des messages à cette rubrique lors du rejet d'une mise à jour dans un thing shadow.

`$aws/things/myLightBulb/shadow/update/delta`

Le service Thing Shadows envoie des messages à cette rubrique lorsqu'un écart est constaté entre les sections déclarées et les sections souhaitées d'un thing shadow.

`$aws/things/myLightBulb/shadow/get/accepted`

Le service Thing Shadows envoie des messages à cette rubrique lorsqu'une demande de thing shadow aboutit.

`$aws/things/myLightBulb/shadow/get/rejected`

Le service Thing Shadows envoie des messages à cette rubrique lorsqu'une demande de thing shadow est rejetée.

`$aws/things/myLightBulb/shadow/delete/accepted`

Le service Thing Shadows envoie des messages à cette rubrique lorsqu'un thing shadow est supprimé.

`$aws/things/myLightBulb/shadow/delete/rejected`

Le service Thing Shadows envoie des messages à cette rubrique lorsqu'une demande de suppression d'un thing shadow est rejetée.

Pour en savoir plus sur toutes les rubriques MQTT utilisées par le service Thing Shadows, consultez la page [Rubriques Device Shadow MQTT \(p. 157\)](#).



Note

Nous vous recommandons de vous abonner aux rubriques `.../rejected` pour voir les éventuelles erreurs envoyées par le service Thing Shadows.

Lorsque l'ampoule est en ligne, elle envoie son état actuel au service Thing Shadows en envoyant un message MQTT à la rubrique `$aws/things/myLightBulb/shadow/update`.

Pour simuler cela, utilisez le client MQTT AWS IoT pour publier le message suivant dans la rubrique `$aws/things/myLightbulb/shadow/update` :

```
{ "state": { "reported": { "color": "red" } } }
```

Le service Thing Shadows répond en envoyant le message suivant à la rubrique `$aws/things/myLightBulb/shadow/update/accepted` :

```
{ "messageNumber": 4, "payload": { "state": { "reported": { "color": "red" } }, "metadata": { "reported": { "color": { "timestamp": 1469564492 } } }, "version": 1, "timestamp": 1469564492 }, "qos": 0, "timestamp": 1469564492848, "topic": "$aws/things/myLightBulb/shadow/update/accepted" }
```

Ce message indique que le service Thing Shadows a reçu la demande de mise à jour et a mis à jour le thing shadow. Si le thing shadow n'existe pas, il est créé. S'il existe, il est mis à jour avec les données contenues dans le message. Si vous ne voyez pas de message publié dans `$aws/things/myLightBulb/shadow/update/accepted`, vérifiez l'abonnement à `$aws/things/myLightBulb/shadow/update/rejected` pour y rechercher d'éventuels messages d'erreur.

Une application qui interagit avec la lampe s'ouvre en ligne et demande l'état actuel de l'ampoule. Elle envoie un message vide à la rubrique `$aws/things/myLightBulb/shadow/get`. Pour simuler cela, utilisez le client MQTT AWS IoT pour publier un message vide ("") dans la rubrique `$aws/things/myLightBulb/shadow/get`.

Le service Thing Shadows répond en publant le thing shadow demandé dans la rubrique `$aws/things/myLightBulb/shadow/get/accepted` :

```
{ "messageNumber": 1, "payload": { "state": { "reported": { "color": "red" } }, "metadata": { "reported": { "color": { "timestamp": 1469564492 } } }, "version": 1, "timestamp": 1469564571 }, "qos": 0, "timestamp": 1469564571533, "topic": "$aws/things/myLightBulb/shadow/get/accepted" }
```

Si vous ne voyez pas de message dans la rubrique \$aws/things/myLightBulb/shadow/get/accepted , vérifiez la rubrique \$aws/things/myLightBulb/shadow/get/rejected pour rechercher d'éventuels messages d'erreur.

L'application affiche ces informations à l'attention de l'utilisateur et l'utilisateur demande une modification de couleur de l'ampoule (du rouge au vert). Pour ce faire, l'application publie un message dans la rubrique \$aws/things/myLightBulb/shadow/update :

```
{ "state": { "desired": { "color": "green" } } }
```

Pour simuler cela, utilisez le client MQTT AWS IoT pour publier le message précédent dans la rubrique \$aws/things/myLightBulb/shadow/update .

Le service Thing Shadows répond en envoyant un message à la rubrique \$aws/things/myLightBulb/shadow/update/accepted :

```
{ "messageNumber": 5, "payload": { "state": { "desired": { "color": "green" } }, "metadata": { "desired": { "color": { "timestamp": 1469564658 } } }, "version": 2, "timestamp": 1469564658 }, "qos": 0, "timestamp": 1469564658286, "topic": "$aws/things/myLightBulb/shadow/update/accepted" }
```

et à la rubrique \$aws/things/myLightBulb/shadow/update/delta :

```
{ "messageNumber": 1, "payload": { "version": 2, "timestamp": 1469564658, "state": { "color": "green" }, "metadata": { "color": { "timestamp": 1469564658 } } }, "qos": 0, "timestamp": 1469564658309, "topic": "$aws/things/myLightBulb/shadow/update/delta" }
```

L'ampoule est abonnée à la rubrique \$aws/things/myLightBulb/shadow/update/delta , elle reçoit donc le message, change de couleur et publie son nouvel état. Pour simuler cela, utilisez le client MQTT AWS IoT pour publier le message suivant dans la rubrique \$aws/things/myLightbulb/shadow/update pour mettre à jour l'état du thing shadow :

```
{ "state": { "reported": { "color": "green" }, "desired": null } }
```

En réponse, le service Thing Shadows envoie un message à la rubrique \$aws/things/myLightBulb/shadow/update/accepted :

```
{ "messageNumber": 6, "payload": { "state": { "reported": { "color": "green" }, "desired": null }, "metadata": { "reported": { "color": { "timestamp": 1469564801 } } }, "desired": { "timestamp": 1469564801 }, "version": 3, "timestamp": 1469564801 }, "qos": 0, "timestamp": 1469564801673, "topic": "$aws/things/myLightBulb/shadow/update/accepted" }
```

L'application demande l'état actuel du service Thing Shadows et affiche les données d'état les plus récentes. Pour simuler cela, exécutez la commande suivante :

```
aws iot-data get-thing-shadow --thing-name "myLightBulb" "output.txt" && cat "output.txt"
```



Note

Sous Windows, omettez `&& cat "output.txt"`, qui affiche le contenu du fichier `output.txt` dans la console. Vous pouvez ouvrir le fichier dans le Bloc-notes ou utiliser tout éditeur de texte de votre choix pour afficher le contenu du thing shadow.

Le service Thing Shadows renvoie le document thing shadow :

```
{ "state":{ "reported":{ "color":"green" } }, "metadata":  
{ "reported":{ "color":{ "timestamp":1469564801 } } }, "version":3,  
"timestamp":1469564864}
```

Si vous souhaitez déterminer si un dispositif est actuellement connecté, incluez un paramètre de connexion dans le thing shadow et utilisez un message MQTT Last Will and Testament (LWT) qui définira la valeur du paramètre de connexion à `false` si un dispositif est déconnecté en raison d'une erreur.



Note

Actuellement, les messages LWT envoyés aux rubriques réservées AWS IoT (rubriques qui commencent par \$) sont ignorés. Pour contourner ce problème, enregistrez un message LWT dans une rubrique non réservée et créez une règle qui permet de republier le message dans la rubrique réservée. L'exemple suivant montre comment créer une règle de republication qui écoute des messages à partir de la rubrique `my/things/myLightBulb/update` et les republie dans `$aws/things/myLightBulb/shadow/update`.

```
{ "rule": { "ruleDisabled": false, "sql": "SELECT * FROM 'my/things/myLightBulb/update'", "description": "Turn my/things/ into $aws/things/", "actions": [ { "republish": { "topic": "$aws/things/myLightBulb/shadow/update", "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish" } } ] } }
```

Lorsqu'un dispositif se connecte, il enregistre un LWT qui définit le paramètre de connexion à la valeur `false`:

```
{ "reported": { "connected":"false" } }
```

Il publie également un message dans sa rubrique de mise à jour (`$aws/things/myLightBulb/shadow/update`), en définissant son état connecté à la valeur `true`:

```
{ "reported": { "connected":"true" } }
```

Lorsque le dispositif se déconnecte normalement, il publie un message dans sa rubrique de mise à jour et définit son état connecté à la valeur `false`:

```
{ "reported": { "connected":"false" } }
```

Si l'appareil se déconnecte en raison d'une erreur, son message LWT est publié automatiquement dans la rubrique de mise à jour.

Pour supprimer le thing shadow, publiez un message vide dans la rubrique `$aws/things/myLightBulb/shadow/delete`. AWS IoT répondra en publiant un message dans la rubrique `$aws/things/myLightBulb/shadow/delete/accepted` :

```
{ "messageNumber": 2, "payload": { "version": 3, "timestamp": 1469564968 },  
  "qos": 0, "timestamp": 1469564968492, "topic": "$aws/things/myLightBulb/  
shadow/delete/accepted" }
```

Documents Device Shadows

Le service Thing Shadows respecte toutes les règles de la spécification JSON. Valeurs, objets et tableaux sont stockés dans le document thing shadow.

Table des matières

- [Propriétés du document \(p. 148\)](#)
- [Versioning d'un thing shadow \(p. 149\)](#)
- [Jeton client \(p. 149\)](#)
- [Exemple de document \(p. 149\)](#)
- [Sections vides \(p. 149\)](#)
- [Tableaux \(p. 150\)](#)

Propriétés du document

Un document thing shadow possède les propriétés suivantes :

`state`

`desired`

État souhaité de l'objet. Les applications peuvent écrire dans cette partie du document pour mettre à jour l'état d'un objet sans avoir à se connecter directement à l'objet.

`reported`

L'état déclaré de l'objet. Les objets écrivent dans cette partie du document pour signaler leur nouvel état. Les applications lisent cette partie du document pour déterminer l'état d'un objet.

`metadata`

Informations sur les données stockées dans la section `state` du document. Il s'agit des horodatages, en heure Unix, de chaque attribut de la section `state`, ce qui vous permet de déterminer s'ils ont été mis à jour.

`timestamp`

Indique si le message a été transmis par AWS IoT. En utilisant l'horodatage dans le message et les horodatages des attributs dans la section souhaitée ou déclarée, un objet peut déterminer l'âge d'un élément mis à jour, même s'il ne dispose pas d'une horloge interne.

`clientToken`

Chaîne unique du dispositif qui permet d'associer les réponses à des demandes dans un environnement MQTT.

`version`

Version du document. Chaque fois que le document est mis à jour, ce numéro de version est incrémenté. Permet de s'assurer que la version du document en cours de mise à jour est la plus récente.

Pour plus d'informations, consultez [Syntaxe d'un document Device Shadow \(p. 162\)](#).

Versioning d'un thing shadow

Le service Thing Shadows prend en charge la gestion du versioning sur chaque message de mise à jour (demande et réponse), ce qui signifie qu'à chaque mise à jour d'un thing shadow, la version du document JSON est incrémentée. Cela permet de garantir deux choses :

- Un client peut recevoir une erreur s'il tente de remplacer un shadow par un numéro de version plus ancien. Le client est informé qu'il doit effectuer une resynchronisation avant de mettre à jour un thing shadow.
- Un client peut décider de ne pas agir sur un message reçu si le message est d'une version inférieure à la version stockée par le client.

Dans certains cas, un client peut contourner la correspondance de version en ne soumettant pas une version.

Jeton client

Vous pouvez utiliser un jeton client avec la messagerie MQTT pour vérifier si une demande et une réponse contiennent le même jeton client. Cela garantit que la réponse et la demande sont associées.

Exemple de document

Voici un exemple de document thing shadow :

```
{ "state" : { "desired" : { "color" : "RED", "sequence" : [ "RED", "GREEN", "BLUE" ] }, "reported" : { "color" : "GREEN" } }, "metadata" : { "desired" : { "color" : { "timestamp" : 12345 }, "sequence" : { "timestamp" : 12345 } }, "reported" : { "color" : { "timestamp" : 12345 } } }, "version" : 10, "clientToken" : "UniqueClientToken", "timestamp": 123456789 }
```

Sections vides

Un document thing shadow contient une section souhaitée uniquement s'il comporte un état souhaité. Par exemple, le document suivant est un document d'état valide avec sans section souhaitée :

```
{ "reported" : { "temp": 55 } }
```

La section déclarée peut également être vide :

```
{ "desired" : { "color" : "RED" } }
```

Si une mise à jour entraîne la nullité des sections souhaitées ou déclarées, la section est supprimée du document. Pour supprimer la section souhaitée d'un document (en réponse, par exemple, à la mise à jour par un dispositif de son état), définissez la section souhaitée à la valeur null:

```
{ "state": { "reported": { "color": "red" }, "desired": null } }
```

Il est également possible qu'un document thing shadow ne contienne pas de section souhaitée ou déclarée. Dans ce cas, le document thing shadow est vide. Par exemple, le document suivant est valide :

```
{ }
```

Tableaux

Les thing shadows prennent en charge les tableaux, mais les traitent comme des valeurs normales, dans la mesure où une mise à jour d'un tableau remplace l'intégralité du tableau. Il n'est pas possible de mettre à jour une partie d'un tableau.

État initial :

```
{ "desired" : { "colors" : [ "RED", "GREEN", "BLUE" ] } }
```

Mise à jour:

```
{ "desired" : { "colors" : [ "RED" ] } }
```

État final :

```
{ "desired" : { "colors" : [ "RED" ] } }
```

Les tableaux ne peuvent pas avoir de valeurs null. Par exemple, le tableau suivant n'est pas valide et sera rejeté.

```
{ "desired" : { "colors" : [ null, "RED", "GREEN" ] } }
```

Utilisation de Device Shadows

AWS IoT fournit trois méthodes à utiliser avec thing shadows :

MISE A JOUR

Crée un thing shadow s'il n'existe pas, ou met à jour le contenu d'un thing shadow avec les données fournies dans la demande. Les données sont stockées avec les informations d'horodatage pour indiquer la dernière date de mise à jour. Des messages sont envoyés à tous les abonnés indiquant l'écart entre l'état déclaré et l'état souhaité (delta). Les objets ou applications qui reçoivent un message peuvent effectuer une action en fonction de l'écart entre les états déclarés et souhaités. Par exemple, un dispositif peut mettre à jour son état à l'état souhaité, ou une application peut mettre à jour son interface utilisateur pour afficher le changement d'état de l'appareil.

GET

Récupère le dernier état stocké dans le thing shadow (par exemple, au moment du démarrage d'un dispositif pour récupérer la configuration et le dernier état de l'opération). Cette méthode retourne le document JSON complet, métadonnées incluses.

DELETE

Supprime un thing shadow, avec tout son contenu. Cela supprime le document JSON du magasin de données. Vous ne pouvez pas restaurer un thing shadow que vous avez supprimé, mais vous pouvez créer un nouveau thing shadow portant le même nom.

Prise en charge du protocole

Ces méthodes sont pris en charge via [MQTT](#) et une API RESTful sur HTTPS. Etant donné que MQTT est un modèle de communication de publication et d'abonnement, AWS IoT implémente un ensemble

de rubriques réservées. Les objets ou applications s'abonnent à ces rubriques avant de publier dans une rubrique de demande afin d'implémenter un comportement de demande-réponse. Pour plus d'informations, consultez [Rubriques Device Shadow MQTT \(p. 157\)](#) and [API RESTful de Device Shadow \(p. 155\)](#).

Mise à jour d'une thing shadow

Vous pouvez mettre à jour un thing shadow en utilisant [UpdateThingShadow \(p. 156\)](#) l'API RESTful ou en publiant dans la [/update \(p. 158\)](#) rubrique. Les mises à jour concernent uniquement les champs spécifiés dans la demande.

État initial :

```
{ "state": { "reported": { "color": { "r": 255, "g": 255, "b": 0 } } } }
```

Envoi d'un message de mise à jour :

```
{ "state": { "desired": { "color": { "r": 10 }, "engine": "ON" } } }
```

Le dispositif reçoit l'état souhaité dans la rubrique /update/delta qui est déclenchée par le message previous/update, puis exécute les modifications souhaitées. Lorsqu'il a terminé, le dispositif doit confirmer son état mis à jour via la section déclarée du document JSON thing shadow.

État final :

```
{ "state": { "reported": { "color": { "r": 10, "g": 255, "b": 0 }, "engine": "ON" } } }
```

Récupération d'un document Thing Shadow

Vous pouvez récupérer un thing shadow en utilisant [GetThingShadow \(p. 155\)](#) l'API RESTful ou en vous inscrivant et en publiant dans la [/get \(p. 160\)](#) rubrique. Cela récupère l'ensemble du document, plus le delta entre les états souhaités et déclarés.

Exemple de document :

```
{ "state": { "desired": { "lights": { "color": "RED", "engine": "ON" }, "reported": { "lights": { "color": "GREEN", "engine": "ON" } }, "metadata": { "desired": { "lights": { "color": { "timestamp": 123456 }, "engine": { "timestamp": 123456 } } }, "reported": { "lights": { "color": { "timestamp": 789012 }, "engine": { "timestamp": 789012 } } }, "version": 10, "timestamp": 123456789 } } }
```

Réponse:

```
{ "state": { "desired": { "lights": { "color": "RED", "engine": "ON" }, "reported": { "lights": { "color": "GREEN", "engine": "ON" }, "delta": { "lights": { "color": "RED" } } }, "metadata": { "desired": { "lights": { "color": { "timestamp": 123456 }, "engine": { "timestamp": 123456 } } }, "reported": { "lights": { "color": { "timestamp": 789012 }, "engine": { "timestamp": 789012 } } }, "delta": { "lights": { "color": { "timestamp": 123456 } } }, "version": 10, "timestamp": 123456789 } } }
```

Verrouillage optimiste

Vous pouvez utiliser la version du document d'état pour vous assurer que vous mettez à jour la version la plus récente d'un document thing shadow. Lorsque vous fournissez une version dans une demande de mise à jour, le service rejette la demande avec un code de réponse de conflit HTTP 409 si la version actuelle du document d'état ne correspond pas à la version fournie.

Exemples :

Document initial :

```
{ "state" : { "desired" : { "colors" : [ "RED", "GREEN", "BLUE" ] } },  
  "version" : 10 }
```

Mise à jour : (la version ne correspond pas ; la demande est rejetée)

```
{ "state": { "desired": { "colors": [ "BLUE" ] } }, "version": 9 }
```

Résultat:

```
Conflit 409
```

Mise à jour : (la version correspond ; cette demande est acceptée)

```
{ "state": { "desired": { "colors": [ "BLUE" ] } }, "version": 10 }
```

État final

```
{ "state": { "desired": { "colors": [ "BLUE" ] } }, "version": 11 }
```

Suppression des données

Vous pouvez supprimer les données d'un thing shadow en publiant dans la rubrique [/update \(p. 158\)](#), en définissant les champs à supprimer à la valeur null. Tout champ avec une valeur null est supprimé du document.

État initial :

```
{ "state": { "desired" : { "lights": { "color": "RED" }, "engine" : "ON" },  
  "reported" : { "lights" : { "color": "GREEN" }, "engine" : "OFF" } } }
```

Envoi d'un message de mise à jour :

```
{ "state": { "desired": null, "reported": { "engine": null } } }
```

État final :

```
{ "state": { "reported" : { "lights" : { "color" : "GREEN" } } } }
```

Vous pouvez supprimer toutes les données à partir d'un thing shadow en définissant son état à la valeur null. Par exemple, l'envoi du message suivant supprime toutes les données d'état, mais le thing shadow reste.

```
{ "state": null }
```

Le thing shadow existe toujours, même si son état est null. La version du thing shadow sera incrémentée lors de la mise à jour suivante.

Suppression d'un thing shadow

Vous pouvez supprimer un document thing shadow en utilisant la rubrique [DeleteThingShadow \(p. 157\)](#) l'API RESTful ou en publiant dans la [/supprimer \(p. 161\)](#).

État initial :

```
{ "state": { "desired": { "lights": { "color": "RED", "engine": "ON" } }, "reported": { "lights": { "color": "GREEN", "engine": "OFF" } } }
```

Un message est envoyé à la rubrique /delete.

État final :

```
HTTP 404 - Ressource introuvable
```

État Delta

L'état Delta est un type virtuel d'état qui contient l'écart entre les états souhaités et déclarés. Champs de la section souhaitée qui ne sont pas dans la section déclarée sont inclus dans le delta. Champs de la section déclarée qui ne sont pas dans la section souhaitée ne sont pas inclus dans le delta. Le delta contient des métadonnées et ses valeurs sont égales aux métadonnées contenues dans le champ souhaité. Exemples :

```
{ "state": { "desired": { "color": "RED", "state": "STOP" }, "reported": { "color": "GREEN", "engine": "ON" }, "delta": { "color": "RED", "state": "STOP" }, "metadata": { "desired": { "color": { "timestamp": 12345 } }, "state": { "timestamp": 12345 }, "reported": { "color": { "timestamp": 12345 } }, "engine": { "timestamp": 12345 } }, "delta": { "color": { "timestamp": 12345 } }, "state": { "timestamp": 12345 } }, "version": 17, "timestamp": 123456789 }
```

Lorsque des objets imbriqués diffèrent, le delta contient le chemin d'accès à la racine.

```
{ "state": { "desired": { "lights": { "color": { "r": 255, "g": 255, "b": 255 } } }, "reported": { "lights": { "color": { "r": 255, "g": 0, "b": 255 } } }, "delta": { "lights": { "color": { "g": 255 } } }, "version": 18, "timestamp": 123456789 }
```

Le service Thing Shadows calcule le delta par itération sur chaque champ de l'état souhaité en le comparant à l'état déclaré.

Les tableaux sont traités comme des valeurs. Si un tableau de la section souhaitée ne correspond pas au tableau de la section déclarée, l'intégralité du tableau souhaité est copiée dans le delta.

Observation des changements d'état

Lorsqu'un thing shadow est mis à jour, les messages sont publiés dans deux rubriques MQTT :

- \$aws/things/*thing-name*/shadow/update/accepted
- \$aws/things/*thing-name*/shadow/update/delta

Le message envoyé à la rubrique /update/delta est destiné à l'objet dont l'état est mis à jour. Ce message contient uniquement l'écart entre les sections souhaitées et déclarées du document thing shadow. À la réception de ce message, l'objet décide s'il convient d'effectuer le changement demandé. Si l'état de l'objet est modifié, il publie son nouvel état dans la rubrique \$aws/things/thing-name/shadow/update.

Les dispositifs et les applications peuvent s'abonner à l'un de ces rubriques pour être informés du changement d'état du document.

Voici un exemple de ce flux :

1. Le dispositif déclare son état.
2. Le système met à jour le document d'état dans son magasin de données persistantes.
3. Le système publie un message delta, qui contient uniquement le delta et vise les dispositifs abonnés. Les dispositifs doivent s'abonner à cette rubrique pour recevoir des mises à jour.
4. Le thing shadow publie un message accepté, qui contient l'ensemble du document reçu, métadonnées comprises. Les applications doivent s'abonner à cette rubrique pour recevoir des mises à jour.

Ordre des messages

Il n'existe aucune garantie que les messages émanant du service AWS IoT parviendront au dispositif dans un ordre spécifique.

Document d'état initial :

```
{ "state" : { "reported" : { "color" : "blue" } }, "version" : 10,  
"timestamp": 123456777 }
```

Mise à jour 1 :

```
{ "state": { "desired" : { "color" : "RED" } }, "version": 10, "timestamp":  
123456777 }
```

Mise à jour 2 :

```
{ "state": { "desired" : { "color" : "GREEN" } }, "version": 11 ,  
"timestamp": 123456778 }
```

Document d'état final :

```
{ "state": { "reported": { "color" : "GREEN" } }, "version": 12, "timestamp":  
123456779 }
```

Il en résulte deux messages delta :

```
{ "state": { "color": "RED" }, "version": 11, "timestamp": 123456778 }
```

```
{ "state": { "color" : "GREEN" }, "version": 12, "timestamp": 123456779 }
```

Le dispositif peut recevoir ces messages dans le désordre. Etant donné que l'état de ces messages est cumulé, un dispositif peut ignorer en toute sécurité tous les messages qui contiennent un numéro de version plus ancien que celui qu'il suit. Si le dispositif reçoit le delta pour la version 12 avant la version 11, il peut en toute sécurité ignorer le message concernant la version 11.

Supprimer des messages Device Shadows

Pour réduire la taille des messages thing shadow envoyés à votre dispositif, définissez une règle qui sélectionne uniquement les champs dont votre dispositif a besoin et republiez le message dans une rubrique MQTT que votre dispositif écoute.

La règle est spécifiée dans JSON et doit ressembler à ceci :

```
{ "sql": "SELECT state, version FROM '$aws/things/+shadow/update/delta'", "ruleDisabled": false, "actions": [ { "republish": { "topic": "${topic(2)}/delta", "roleArn": "arn:aws:iam::123456789012:role/my-iot-role" } } ] }
```

L'instruction select détermine quels champs du message seront republiés dans la rubrique spécifiée. Un caractère générique « + » est utilisé pour appeler toutes les noms de thing shadow. La règle spécifie que tous les messages correspondants doivent être republiés dans la rubrique spécifiée. Dans ce cas, la fonction « topic() » est utilisée pour indiquer la rubrique dans laquelle republier. topic(2) prend la valeur du nom de l'objet dans la rubrique d'origine. Pour plus d'informations sur la création de règles, consultez [Règles](#).

API RESTful de Device Shadow

Un thing shadow expose l'URI suivante pour mettre à jour les informations d'état :

```
https://endpoint/things/thingName/shadow
```

Le point de terminaison est spécifique à votre compte AWS. Pour récupérer votre point de terminaison, utilisez la commande [describe-endpoint](#). Le format du point de terminaison est le suivant :

```
identifiant.iot.région.amazonaws.com
```

Actions API

- [GetThingShadow](#) (p. 155)
- [UpdateThingShadow](#) (p. 156)
- [DeleteThingShadow](#) (p. 157)

GetThingShadow

Obtient le thing shadow de l'objet spécifié.

Le document d'état de réponse comprend le delta entre les états souhaité et déclaré.

Requête

La demande comprend les en-têtes HTTP standard, plus l'URI suivante :

```
HTTP GET https://endpoint/things/thingName/shadow
```

Réponse

En cas de réussite, la réponse comprend les en-têtes HTTP standard, plus le code et le corps suivants :

```
HTTP 200 BODY: response state document
```

Pour plus d'informations, consultez [Exemple de document d'état de réponse \(p. 162\)](#).

Autorisation

La récupération d'un thing shadow nécessite une stratégie qui permet au mandataire de réaliser l'action `iot:GetThingShadow`. Le service Thing Shadows accepte deux formes d'authentification : signature version 4 avec les informations d'identification IAM ou l'authentification mutuelle TLS avec un certificat client.

Voici un exemple de stratégie qui permet à un mandataire de récupérer un thing shadow :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "iot:GetThingShadow", "Resource": ["arn:aws:iot:région:compte:thing/thing"] } ] }
```

UpdateThingShadow

Met à jour le thing shadow de l'objet spécifié.

Les mises à jour concernent uniquement les champs spécifiés dans le document d'état de la demande. Tout champ avec une valeur `null` est supprimé du thing shadow.

Requête

La demande comprend les en-têtes HTTP standard, plus l'URI et le corps suivants :

```
HTTP POST https://endpoint/things/thingName/shadow BODY: request state document
```

Pour plus d'informations, consultez [Exemple de document d'état de la demande \(p. 162\)](#).

Réponse

En cas de réussite, la réponse comprend les en-têtes HTTP standard, plus le code et le corps suivants :

```
HTTP 200 BODY: response state document
```

Pour plus d'informations, consultez [Exemple de document d'état de réponse \(p. 162\)](#).

Autorisation

La mise à jour d'un thing shadow nécessite une stratégie qui permet au mandataire de réaliser l'action `iot:UpdateThingShadow`. Le service Thing Shadows accepte deux formes d'authentification :

signature version 4 avec les informations d'identification IAM ou l'authentification mutuelle TLS avec un certificat client.

Voici un exemple de stratégie qui permet à un mandataire de mettre à jour un thing shadow :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "iot:UpdateThingShadow", "Resource": ["arn:aws:iot:région:compte:thing/thing"] } ] }
```

DeleteThingShadow

Supprime le thing shadow de l'objet spécifié.

Requête

La demande comprend les en-têtes HTTP standard, plus l'URI suivante :

```
HTTP DELETE https://endpoint/things/thingName/shadow
```

Réponse

En cas de réussite, la réponse comprend les en-têtes HTTP standard, plus le code et le corps suivants :

```
HTTP 200 BODY: Document d'état de la réponse vide
```

Autorisation

La suppression d'un thing shadow nécessite une stratégie qui permet au mandataire de réaliser l'action iot:DeleteThingShadow . Le service Thing Shadows accepte deux formes d'authentification : signature version 4 avec les informations d'identification IAM ou l'authentification mutuelle TLS avec un certificat client.

Voici un exemple de stratégie qui permet au mandataire de supprimer un thing shadow :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "iot:DeleteThingShadow", "Resource": ["arn:aws:iot:région:compte:thing/thing"] } ] }
```

Rubriques Device Shadow MQTT

Le service Thing Shadows utilise des rubriques réservées MQTT pour permettre à des applications et des objets d'obtenir, mettre à jour ou supprimer les informations d'état d'un objet (thing shadow). Les noms de ces rubriques commencent par \$aws/things/**thingName**/shadow. La publication et l'abonnement à des rubriques thing shadow nécessite une autorisation basée sur les rubriques. AWS IoT se réserve le droit d'ajouter de nouvelles rubriques à la structure de rubriques existante. C'est pourquoi nous vous recommandons d'éviter abonnements de caractère générique aux rubriques shadow. Par exemple, évitez de vous abonner à des filtres de rubrique tels que \$aws/things/**thingName**/shadow/# car le nombre de rubriques qui correspondent à ce filtre peut augmenter lorsque AWS IoT introduit de nouvelles rubriques shadow.

Voici les rubriques MQTT utilisés pour interagir avec les thing shadows.

Rubriques

- [/update \(p. 158\)](#)
- [/update/accepted \(p. 158\)](#)
- [/update/documents \(p. 159\)](#)
- [/update/rejected \(p. 159\)](#)
- [/update/delta \(p. 159\)](#)
- [/get \(p. 160\)](#)
- [/get/accepted \(p. 160\)](#)
- [/get/rejected \(p. 161\)](#)
- [/supprimer \(p. 161\)](#)
- [/delete/accepted \(p. 161\)](#)
- [/delete/rejected \(p. 162\)](#)

/update

Un objet publie un document d'état de la demande dans cette rubrique pour mettre à jour le thing shadow :

```
$aws/things/thingName/shadow/update
```

AWS IoT répond en publant dans [/update/accepted \(p. 158\)](#) ou [/update/rejected \(p. 159\)](#).

Pour plus d'informations, consultez [Documents d'état de la demande \(p. 162\)](#).

Exemple de stratégie

Voici un exemple de stratégie :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": ["iot:Publish"], "Resource": ["arn:aws:iot:région:compte:topic/$aws/things/thingName/shadow/update"] } ] }
```

/update/accepted

AWS IoT publie un document d'état de réponse dans cette rubrique lorsqu'il accepte une modification du thing shadow :

```
$aws/things/thingName/shadow/update/accepted
```

Pour plus d'informations, consultez [Documents d'état de la réponse \(p. 162\)](#).

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource": ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/update/accepted"] } ] }
```

/update/documents

AWS IoT publie un document d'état dans cette rubrique chaque fois qu'une mise à jour du shadow est effectué avec succès :

```
$aws/things/thingName/shadow/update/documents
```

Le document JSON contient deux nœuds principaux « previous » et « current ». Le nœud « previous » contient le contenu du document thing shadow complet avant la mise à jour alors que « current » contient le document thing shadow après la mise à jour. Lorsque le device shadow est mis à jour (créé) pour la première fois, le nœud « previous » contient « null ».

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
  "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource":  
    ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/update/  
      documents" ] } ] }
```

/update/rejected

AWS IoT publie un document de réponse à l'erreur dans cette rubrique lorsqu'il rejette une modification du thing shadow :

```
$aws/things/thingName/shadow/update/rejected
```

Pour plus d'informations, consultez [Documents de réponse d'erreur \(p. 163\)](#).

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
  "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource":  
    ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/update/  
      rejected" ] } ] }
```

/update/delta

AWS IoT publie un document d'état de la réponse dans cette rubrique lorsqu'il accepte une modification du thing shadow et que le document d'état de la demande contient des valeurs différentes pour les états souhaités et déclarés :

```
$aws/things/thingName/shadow/update/delta
```

Pour plus d'informations, consultez [Documents d'état de la réponse \(p. 162\)](#).

Détails de la publication

- Un message publié dans update/delta comprend uniquement les attributs souhaités qui diffèrent entre les sections déclarées et les sections souhaitées. Il contient tous ces attributs,

indépendamment de s'ils étaient contenus dans le message de mise à jour actuel ou s'ils étaient déjà stockés dans AWS IoT. Attributs qui ne diffèrent pas entre les sections déclarées et souhaitées ne sont pas compris.

- Si un attribut figure dans la section déclarée, mais qu'il n'a aucun équivalent dans la section souhaitée, il n'est pas inclus.
- Si un attribut figure dans la section souhaitée, mais qu'il n'a aucun équivalent dans la section déclarée, il n'est pas inclus.
- Si un attribut est supprimé de la section déclarée, mais qu'il existe toujours dans la section souhaitée, il est inclus.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
  "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource":  
  ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/update/  
  delta" ] } ] }
```

/get

Un objet publie dans cette rubrique pour obtenir le thing shadow :

```
$aws/things/thingName/shadow/get
```

AWS IoT répond en publiant dans [/get/accepted](#) (p. 160) ou [/get/rejected](#) (p. 161).

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action":  
  ["iot:Publish"], "Resource": [ "arn:aws:iot:région:compte:topic/$aws/  
  things/thingName/shadow/get" ] } ] }
```

/get/accepted

AWS IoT publie un document d'état de réponse dans cette rubrique lors du retour au thing shadow :

```
$aws/things/thingName/shadow/get/accepted
```

Pour plus d'informations, consultez [Documents d'état de la réponse](#) (p. 162).

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
  "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource":  
  ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/get/  
  accepted" ] } ] }
```

/get/rejected

AWS IoT publie un document de réponse d'erreur dans cette rubrique lorsqu'il ne peut pas retourner le thing shadow :

```
$aws/things/thingName/shadow/get/rejected
```

Pour plus d'informations, consultez [Documents de réponse d'erreur \(p. 163\)](#).

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource": ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/get/rejected"] } ] }
```

/supprimer

Un objet publie un document dans cette rubrique pour supprimer un thing shadow :

```
$aws/things/thingName/shadow/delete
```

Pour supprimer un thing shadow, envoyez un message à la rubrique de suppression. Le contenu du message est ignoré.

AWS IoT répond en publiant dans [/delete/accepted \(p. 161\)](#) ou [/delete/rejected \(p. 162\)](#).

Exemple de stratégie

Voici un exemple de stratégie :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource": ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/delete"] } ] }
```

/delete/accepted

AWS IoT publie un message dans cette rubrique lors de la suppression d'un thing shadow :

```
$aws/things/thingName/shadow/delete/accepted
```

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource": ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/delete/accepted"] } ] }
```

/delete/rejected

AWS IoT publie un document de réponse d'erreur dans cette rubrique lorsqu'il ne peut pas supprimer le thing shadow :

```
$aws/things/thingName/shadow/delete/rejected
```

Pour plus d'informations, consultez [Documents de réponse d'erreur \(p. 163\)](#).

Exemple de stratégie

Voici un exemple de document de stratégie requise :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow",  
  "Action": [ "iot:Subscribe", "iot:Receive" ], "Resource":  
    ["arn:aws:iot:région:compte:topicfilter/$aws/things/thingName/shadow/delete/  
      rejected" ] } ] }
```

Syntaxe d'un document Device Shadow

Le service Thing Shadows utilise les documents suivants dans les opérations UPDATE, GET et DELETE avec l'[API RESTful \(p. 155\)](#) ou [les messages de publication/abonnement MQTT \(p. 157\)](#). Pour plus d'informations, consultez [Documents Device Shadows \(p. 148\)](#).

Exemples

- [Documents d'état de la demande \(p. 162\)](#)
- [Documents d'état de la réponse \(p. 162\)](#)
- [Documents de réponse d'erreur \(p. 163\)](#)

Documents d'état de la demande

Les documents d'état de la demande ont le format suivant :

```
{ "state": { "desired": { "attribute1": integer2, "attribute2":  
  "string2", ... "attributeN": boolean2  
    }, "reported": { "attribute1": integer1, "attribute2": "string1", ...  
  "attributeN": boolean1  
    } } "clientToken": "token", "Version": Version  
}
```

- state — Les mises à jour affectent uniquement les champs spécifiés.
- clientToken — Permet de vérifier que la demande et la réponse contiennent le même jeton client.
- Version — Le service Thing Shadows traite la mise à jour uniquement si la version spécifiée correspond à la dernière version qu'il a.

Documents d'état de la réponse

Les documents d'état de la réponse ont le format suivant :

```
{ "state": { "desired": { "attribute1": integer2, "attribute2": "string2", ... "attributeN": boolean2 }, "reported": { "attribute1": integer1, "attribute2": "string1", ... "attributeN": boolean1 }, "delta": { "attribute3": integerX, "attribute5": "stringY" } }, "metadata": { "desired": { "attribute1": { "timestamp": timestamp }, "attribute2": { "timestamp": timestamp }, ... "attributeN": { "timestamp": timestamp } }, "reported": { "attribute1": { "timestamp": timestamp }, "attribute2": { "timestamp": timestamp }, ... "attributeN": { "timestamp": timestamp } } }, "timestamp": timestamp, "clientToken": "token", "Version": Version }
```

- state
 - reported — Présent uniquement si un objet a déclaré des données dans la section `reported` et qu'il contient uniquement les champs qui se trouvaient dans le document d'état de la demande.
 - desired — Présent uniquement si un objet a déclaré des données dans la section `desired` et qu'il contient uniquement les champs qui se trouvaient dans le document d'état de la demande.
- metadata — Contient les horodatages pour chaque attribut des sections `desired` et `reported` afin que vous puissiez déterminer si l'état a été mis à jour.
- timestamp — Date et heure Unix auxquelles la réponse a été générée par AWS IoT.
- clientToken — Présent uniquement si un jeton client a été utilisé lors de la publication d'un code JSON valide dans la rubrique `/update`.
- Version — Version actuelle du document du thing shadow partagés dans AWS IoT. Elle est augmentée d'une unité par rapport à la version précédente du document.

Documents de réponse d'erreur

Les documents de réponse d'erreur ont le format suivant :

```
{ "code": error-code, "message": "error-message", "timestamp": timestamp, "clientToken": "token" }
```

- code — Code de réponse HTTP qui indique le type d'erreur.
- message — Message texte qui fournit des informations supplémentaires.
- timestamp — Date et heure auxquelles la réponse a été générée par AWS IoT.
- clientToken — Présent uniquement si un jeton client a été utilisé lors de la publication d'un code JSON valide dans la rubrique `/update`.

Pour plus d'informations, consultez [Messages d'erreur Thing Shadow \(p. 163\)](#).

Messages d'erreur Thing Shadow

Le service Thing Shadows publie un message dans la rubrique d'erreur (via MQTT) en cas d'échec d'une tentative de modifier le document d'état. Ce message est émis uniquement en réponse à une publication dans l'une des rubriques \$aws réservées. Si le client met à jour le document à l'aide de l'API REST, il reçoit le code d'erreur HTTP dans le cadre de la réponse, et aucun message d'erreur MQTT n'est émis.

Code d'erreur HTTP	Messages d'erreur
400 (Requête erronée)	<ul style="list-style-type: none"> • JSON non valide • Nœud requis manquant : état • Le nœud d'état doit être un objet • Le nœud souhaitée doit être un objet • Le nœuds déclaré doit être un objet • Version non valide • ClientToken non valide • Le code JSON contient trop de niveaux d'imbrication ; le maximum est 6 • L'état contient un nœud non valide
401 (Accès non autorisé)	<ul style="list-style-type: none"> • Non autorisé
403 (Accès interdit)	<ul style="list-style-type: none"> • Accès interdit
404 (Introuvable)	<ul style="list-style-type: none"> • Objet non trouvé
409 (Conflit)	<ul style="list-style-type: none"> • Conflit de versions
413 (Charge utile trop importante)	<ul style="list-style-type: none"> • La charge utile dépasse la taille maximale autorisée
415 (Type de support non pris en charge)	<ul style="list-style-type: none"> • Codage documenté non pris en charge ; l'encodage pris en charge est UTF-8
429 (Nombre de requêtes trop élevé)	<ul style="list-style-type: none"> • Le service Thing Shadow génère ce message d'erreur lorsqu'il y a plus de 10 demandes en vol.
500 (Erreur interne du serveur)	<ul style="list-style-type: none"> • Échec du service interne

SDK AWS IoT

Table des matières

- [SDK pour les appareils AWS IoT \(p. 166\)](#)
- [Démarrer AWS IoT sur Raspberry Pi et le SDK pour les appareils pour C IoT AWS \(p. 167\)](#)
- [Démarrer avec AWS IoT AWS sur Raspberry Pi et le SDK pour les appareils pour Javascript AWS IoT \(p. 186\)](#)

AWS IoT fournit un certain nombre de façons d'interagir avec la plateforme de service, via les protocoles [MQTT](#) et [HTTP RESTful](#), ainsi que diverses infrastructures de langages différents. Selon votre utilisation de l'application, vous pouvez trouver un ou plusieurs kits de développement logiciel à utiliser.

- **SDK AWS pour la configuration :** Toutes les API de configuration sont disponibles dans le cadre des SDK AWS standard dans tous les langages de SDK AWS pris en charge sous le namespace du paquetage `iot`. Il existe des fonctions pour créer et gérer :
 - Les identités et l'autorisation (y compris les certificats et les stratégies).
 - Les règles et les actions dans le moteur de règles, les points de terminaison, les rôles et les niveaux de journalisation.
 - Les noms et attributs pour les appareils dans le registre d'objets.
- **SDK AWS pour publier des messages et utiliser des thing shadows :** AWS IoT permet la prise en charge des éléments suivants :
 - Les applications utilisant des utilisateurs/rôles IAM(directement ou par le biais de Amazon Cognito).
 - Le protocole HTTP pour publier les messages directement à l'agent de messages à l'aide de HTTP POST.
 - La capacité d'obtenir, mettre à jour et supprimer les états des fichiers device shadow.

Ces API sont disponibles dans tous les langages pris en charge par les SDK AWS, sous le namespace de paquetage `iot-data`.

- **SDK appareil :** AWS IoT prend en charge les appareils et applications qui utilisent l'authentification de certificat et le protocole MQTT (généralement, des appareils limités en mémoire) afin de publier des messages et de s'abonner à l'agent de messages. Le SDK C peut être compilé sur des distributions Linux à l'aide d'OpenSSL ou des bibliothèques TLS mbed et de la bibliothèque MQTT incluse. Les clients peuvent également utiliser ce SDK pour les plates-formes intégrées supplémentaires et intégrer des bibliothèques TLS et MQTT personnalisées, le cas échéant. Une version spéciale du SDK C est fournie pour Arduino Yún. Le SDK appareil est disponible pour JavaScript dans l'écosystème du paquetage Node.js.

SDK pour les appareils AWS IoT

Le SDK pour les appareils AWS IoT est un kit de développement logiciel en langage C développé pour les appareils ou les systèmes sur puce (SoC) limités avec 256 Ko ou plus de mémoire disponible. Le SDK pour appareils simplifie le processus de connexion de périphériques intégrés à la plateforme AWS IoT en implémentant les exigences de sécurité pour la connexion à l'agent du service AWS IoT et pour fournir les fonctions de publication et souscription de base ainsi que l'accès aux fichiers device shadows via MQTT. Cela élimine la nécessité d'implémenter la prise en charge d'une API HTTP RESTful. Le SDK pour les appareils a été conçu et testé pour fonctionner avec Linux intégré et diverses normes du secteur, les systèmes d'exploitation en temps réel ainsi que les TLS 1.2 tiers et les implémentations de bibliothèques cryptographiques pour prendre en charge les normes de sécurité élevées de l'agent de plateforme HTTP AWS IoT.

Ensemble de fonctions du SDK pour les appareils

Le SDK pour les appareils offre la fonctionnalité suivante sur les plateformes :

- Configuration des informations d'identification de sécurité et des objets de communication d'authentification mutuelle TLS 1.2 avec l'agent de messages AWS IoT dans les régions prises en charge.
- Établit et gère la connexion avec l'agent de messages AWS IoT (via MQTT), y compris la configuration du délai d'attente.
- Fournit une enveloppe adaptée aux implémentations de client MQTT courantes pour publier des données et s'abonner aux rubriques sur MQTT.
- Prend en charge la mise à jour, la récupération et la suppression des device shadows, y compris le versioning des device shadows.

Plateformes prises en charge

Le SDK pour les appareils est disponible pour la création de prototypes de kits de démarrage rapide et pour les kits d'évaluation des principaux fabricants de semi-conducteurs pour le développement de produit. Voici les plateformes prises en charge :

- Arduino Yún

Arduino Yún dispose d'un processeur de communication puissant qui permet d'utiliser des certificats TLS 1.2 et clients pour connecter la carte Arduino Yún à AWS IoT. Le SDK contient des instructions et des scripts d'installation qui préparent votre carte Arduino Yún pour la connexion à AWS IoT. Vous pouvez désormais réaliser davantage de choses avec votre croquis Arduino en accédant aux fonctions Cloud d'AWS via AWS IoT.

Pour plus d'informations, consultez:

[Manuel du développeur et référence d'API](#)

[Code source](#)

[Télécharger le SDK pour Arduino AWS IoT](#)

- SDK C pour les plates-formes intégrées (Linux, système d'exploitation en temps réel)

Le SDK C a été développé dans le respect de la norme du langage C99. Le SDK fournit des exemples d'applications et des instructions de démarrage destinés à améliorer la mise en place et à connecter rapidement votre dispositif à AWS IoT. Le SDK C comprend également un manuel de portage qui permet de modifier les fonctions d'enveloppe adaptées du SDK pour prendre en charge vos types de données spécifiques à la plateforme, les horloges du système d'exploitation,

l'implémentation de TLS embarqués, de sorte que le code de votre application puisse rester inchangé lorsque vous passez d'un environnement de création de prototype rapide sous Linux à un micro contrôleur plus limité et économique.

Pour plus d'informations, consultez:

[Téléchargez le SDK pour les appareils créé pour le package OpenSSL](#)

[Téléchargez le SDK pour les appareils créé pour le package mbedTLS](#)

[Manuel du développeur](#)

[Manuel de portage](#)

[Référence d'API AWS IoT](#)

[Code source](#)

- [Package d'exécution JavaScript](#)

Le SDK pour JavaScript AWS IoT a été développé pour la création de prototype rapide sur des plateformes intégrées plus puissantes, qui prennent en charge l'écosystème de package Node.js. Vous pouvez utiliser le gestionnaire de package NPM pour installer le SDK pour JavaScript AWS IoT. Les exemples d'application et d'instructions présenteront les packages supplémentaires nécessaires pour exécuter votre application JavaScript et la connecter rapidement à AWS IoT.

Pour plus d'informations, consultez:

[Télécharger le SDK pour JavaScript AWS IoT](#)

[Source du SDK pour JavaScript AWS IoT](#)

[Manuel du développeur et référence d'API](#)

Le SDK AWS IoT peut être emballé pour s'exécuter dans un navigateur à l'aide de [browserify](#). Il comprend des scripts d'assistance et des exemples de code d'application destinés à vous aider à commencer à écrire des applications de navigateur utilisant AWS IoT. Pour plus d'informations, consultez <https://github.com/aws/aws-iot-device-sdk-js#browser>

Démarrer AWS IoT sur Raspberry Pi et le SDK pour les appareils pour C AWS IoT

Ce manuel contient des instructions pour connecter votre Raspberry Pi à la plateforme AWS IoT et le configurer pour l'utiliser avec le SDK pour les appareils pour C AWS IoT. Après avoir suivi les étapes de ce manuel, vous pourrez vous connecter à la plateforme AWS IoT et exécuter des exemples d'applications inclus dans le SDK pour les appareils pour C AWS IoT.

Prérequis

- Carte Raspberry Pi entièrement configurée avec accès Internet

Pour plus d'informations sur la configuration de Raspberry Pi, consultez le [Guide de démarrage Raspberry Pi](#).

- Navigateur Chrome ou Firefox (Iceweasel)

Pour plus d'informations sur l'installation d'Iceweasel, consultez [les instructions contenues dans le Wiki Linux intégré](#).

Dans ce manuel, le matériel et les logiciels suivants sont utilisés :

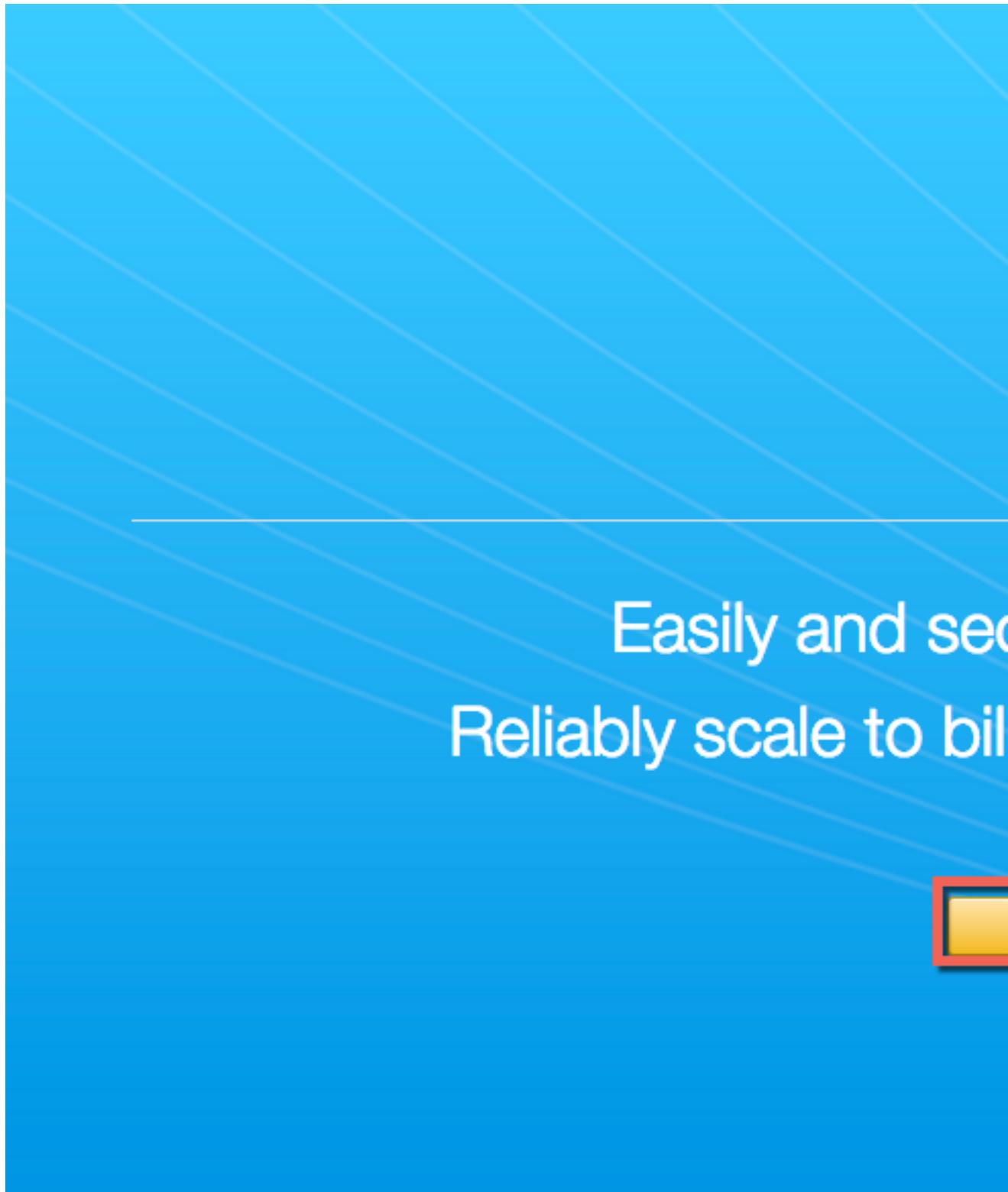
- [Raspberry Pi 2 Modèle B](#)
- [Raspbian Wheezy](#)
- [Navigateur iceweasel](#)

Connexion à Raspberry Pi

Connectez-vous à la console AWS IoT

Mettez le Raspberry Pi sous tension et vérifiez que vous avez une connexion Internet.

Connectez-vous à AWS Management Console et ouvrez la console AWS IoT à partir de l'adresse <https://aws.amazon.com/iot>. Sur la page d'accueil, sélectionnez Démarrer avec AWS IoT.



S'il s'agit de votre première utilisation de la console AWS IoT, vous verrez deux boutons : Mise en route and Démarrer le didacticiel interactif:



AWS I

AWS IoT is a managed cloud platform
-- cars, light bulbs, sensor grids and more -- can
interact with cloud applications.

[Get started](#)

St

[Getting started documentation](#)

Choisissez Mise en route. La page suivante doit apparaître.

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the word "Resources" is displayed in large grey text. To the right of "Resources" is a button with a close icon and the text "Close create panel". Underneath this, there are two main creation buttons: "Create a thing" (represented by a wrench icon) and "Create a rule" (represented by a network icon). Below these buttons is a filter section with a magnifying glass icon and the text "Filter by resource names or by resource type (below)". Under the filter, there are four categories: "All" (selected), "0/0 things", "0/0 rules", "0/0 certificates", and "0/0 policies".

Si vous ne voyez pas la bannière bleue avec les boutons Crée un objet, Crée une règle, Crée un certificat, and Crée une stratégie , cliquez sur le bouton Crée une ressource :

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the word "Resources" is displayed. To the right of "Resources", there's a button labeled "+ Create a resource" which is highlighted with a red rectangular border. Underneath "Resources", there's a search bar with a magnifying glass icon and the placeholder text "Filter by resource names or by resource type (below)". Below the search bar, there are four categories: "All" (which is selected and highlighted with a blue oval), "0/0 things", "0/0 rules", "0/0 certificates", and "0/0 policies".

Créer et attacher un objet (dispositif)

Un objet représente un dispositif dont le statut ou les données sont stockés dans le cloud AWS IoT. Le service Thing Shadow conserve un fichier thing shadow pour chaque dispositif connecté à AWS IoT. Les thing shadows permettent d'accéder aux données d'état des objets et de les modifier.

Choisissez Créer un objet, tapez un nom pour l'objet, puis sélectionnez Créer:

The screenshot shows the AWS IoT Resources page. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the main title "Resources" is displayed. To the right of the title is a button labeled "Close create panel" with a small "X" icon. Further down, there are two main buttons: "Create a thing" (with a robotic arm icon) and "Create a rule" (with a network icon). The "Create a thing" button is highlighted with a blue border. Below these buttons, the section title "Create a thing" is shown in large bold letters. A descriptive text follows: "Create a thing to represent your device in the cloud. This step creates a resource in the AWS IoT service that you can use to publish sensor data or receive commands from the cloud." Under this text, there's a form field with the label "Name" and the value "MyIoTButton". At the bottom of the page, the section title "Attributes" is shown in large bold letters, followed by the text "Next (optional), you can use thing attributes to describe the identity of your device." There's also a "Add attribute" button.

Create a thing

Create a thing to represent your device in the cloud. This step creates a resource in the AWS IoT service that you can use to publish sensor data or receive commands from the cloud.

Name

Attributes

Next (optional), you can use thing attributes to describe the identity of your device.

Add attribute

Outre un message de confirmation, le bouton Afficher l'objet s'affiche :



Filter by resource names or by resource type (below)

All 1/1 things 0/0 rules 0/0 certificates 0/0 policies

MyNewThing

Choisissez Afficher l'objet pour présenter les informations concernant votre objet :

The screenshot shows the AWS IoT Resources page. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the main title "Resources" is displayed. To the right of the title is a button labeled "Close create panel" with a small "X" icon. In the center of the page, there are three main buttons: "Create a thing" (with a wrench icon), "Create a rule" (with a gear icon), and "Use my device" (with a gear icon). The "Create a thing" button is highlighted with a thick blue border. Below these buttons, there's a message: "Your new thing has been created. Click 'View thing' to continue. From there, you can connect a device to this thing, or add a rule to it." To the right of this message is a large blue "View" button. The background of the main content area is light blue.



Filter by resource names or by resource type (below)

All

1/1 things

0/0 rules

0/0 CAs

0/0 certificates

0/0 policies

MyIoTButton

Cliquez sur le bouton Connexion d'un appareil pour télécharger une paire de clés et un certificat générée par AWS IoT :



AWS IoT

Resources

Close create panel



Create a thing



Create a rule



Create a certificate

Your new thing has been created. Click 'View thing' to continue.

From there, you can connect a device to this thing, or add a rule when your thing publishes a message.

View thing



Filter by resource names or by resource type (below)

All

1/1 things

0/0 rules

0/0 certificates

0/0 policies

Select all

First

MyNewThing

Sur la page Connexion d'un appareil , sélectionnez le SDK à utiliser, puis Générer un certificat et une stratégie:



AWS IoT

Connect a device

Connect your device to one of our many supported SDKs.

Embedded C **NodeJS**

Arduino Yún

First, you will follow the steps to authenticate (what you do).

You can generate a device policy at any time.

Un certificat X.509 et une paire de clés sont générés ; activez le certificat X.509, puis créez une stratégie AWS IoT et attachez-la au certificat.

La page suivante s'affiche :



Connect a device

Connect your device to one of our many supported SDKs.

Embedded C **NodeJS**

Arduino Yún

Please download
any time, but

- [Download](#)
- [Download](#)
- [Download](#)

Créez un répertoire de travail appelé `deviceSDK` où stocker vos fichiers. Cliquez sur les liens pour télécharger vos clés publiques et privées et les certificats et enregistrez-les dans le répertoire `deviceSDK`.

Choisissez Confirmer et démarrer la connexion. La page suivante s'affiche :



Connect a device

Connect your device to one of our many supported SDKs.

Embedded C NodeJS

Arduino Yún

AWS IoT

Download on GitHub

- [OpenSSL](#)
- [mbed-TLS](#)

Set up the SDK

Add in the following code:

```
// Get from GitHub
// =====
#define AWS_IOT_MQTT_HOST "a391onaws.com"
#define AWS_IOT_MQTT_PORT 8883
#define AWS_IOT_MQTT_KEEPALIVE 60
#define AWS_IOT_MQTT_CACERT "/path/to/cacert.pem"
#define AWS_IOT_MQTT_CERT "/path/to/cert.pem"
#define AWS_IOT_MQTT_KEY "/path/to/key.pem"
// =====
```

Start one of the examples and observe the shadow. Only one update is sent at the same time. If you change something (and click "Update")

Il existe deux versions du SDK pour les appareils pour C AWS IoT : OpenSSL et mbed TLS. Cliquez sur le lien [OpenSSL](#). Cela va télécharger le SDK pour les appareils pour C AWS IoT dans une archive (`linux_mqtt_openssl-latest.tar`). Enregistrez-le dans votre répertoire `deviceSDK`. Dans une fenêtre du terminal, tapez la commande suivante pour extraire l'archive dans votre répertoire `deviceSDK` :

```
« tar -xvf linux_mqtt_openssl-latest.tar »
```

Configurer le SDK pour les appareils pour C AWS IoT pour l'environnement d'exécution C

Pour pouvoir utiliser le SDK pour les appareils pour C AWS IoT, vous devez installer la bibliothèque OpenSSL sur Raspberry Pi. Dans une fenêtre du terminal, exécutez `sudo apt-get install libssl-dev`.

Configuration de l'exemple d'application

Le SDK pour les appareils pour C AWS IoT contient des exemples d'applications que vous pouvez essayer. Pour plus de simplicité, nous allons exécuter `subscribe_publish_sample`. Copiez votre certificat et la clé privée dans le répertoire `deviceSDK/certs`. Télécharger un certificat d'autorité de certification racine [ici](#). Copiez le texte du certificat d'autorisation racine depuis le navigateur, collez-le dans un fichier, puis copiez-le dans le répertoire `deviceSDK/certs`.

Accédez au répertoire `deviceSDK/sample_apps/subscribe_publish_sample`. Vous devez configurer vos propres point de terminaison, clé privée et certificat. Si vous avez accès à une machine sur laquelle AWS CLI est installé, vous pouvez utiliser la commande `aws iot describe-endpoint` pour trouver l'URL de votre point de terminaison personnel. Sinon, dans la console AWS IoT, double cliquez sur `MyNewThing`, puis cherchez point de terminaison de l'API REST. Copiez tout après « `https://` » notamment « `.com` » :

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the word "Resources" is displayed in large grey text. To the right of "Resources" is a button with a plus sign and the text "Create a resource". Underneath the main title, there's a search bar with a magnifying glass icon and the placeholder text "Filter by resource names or by resource type (below)". Below the search bar, there are four categories with counts: "All" (1/1 things), "0/0 rules", "1/1 certificates", and "1/1 policies". To the right of these counts is a "Select all" button. At the bottom of the visible area, there's a "First" label.

Ouvrez le fichier `aws_iot_config.h` et mettez à jour les valeurs des éléments suivants :

`AWS_IOT_MQTT_HOST`

Votre point de terminaison personnel.

`AWS_IOT_MY_THING_NAME`

Le nom de votre objet.

`AWS_IOT_ROOT_CA_FILENAME`

Votre certificat d'autorité de certification racine.

`AWS_IOT_CERTIFICATE_FILENAME`

Votre certificat.

`AWS_IOT_PRIVATE_KEY_FILENAME`

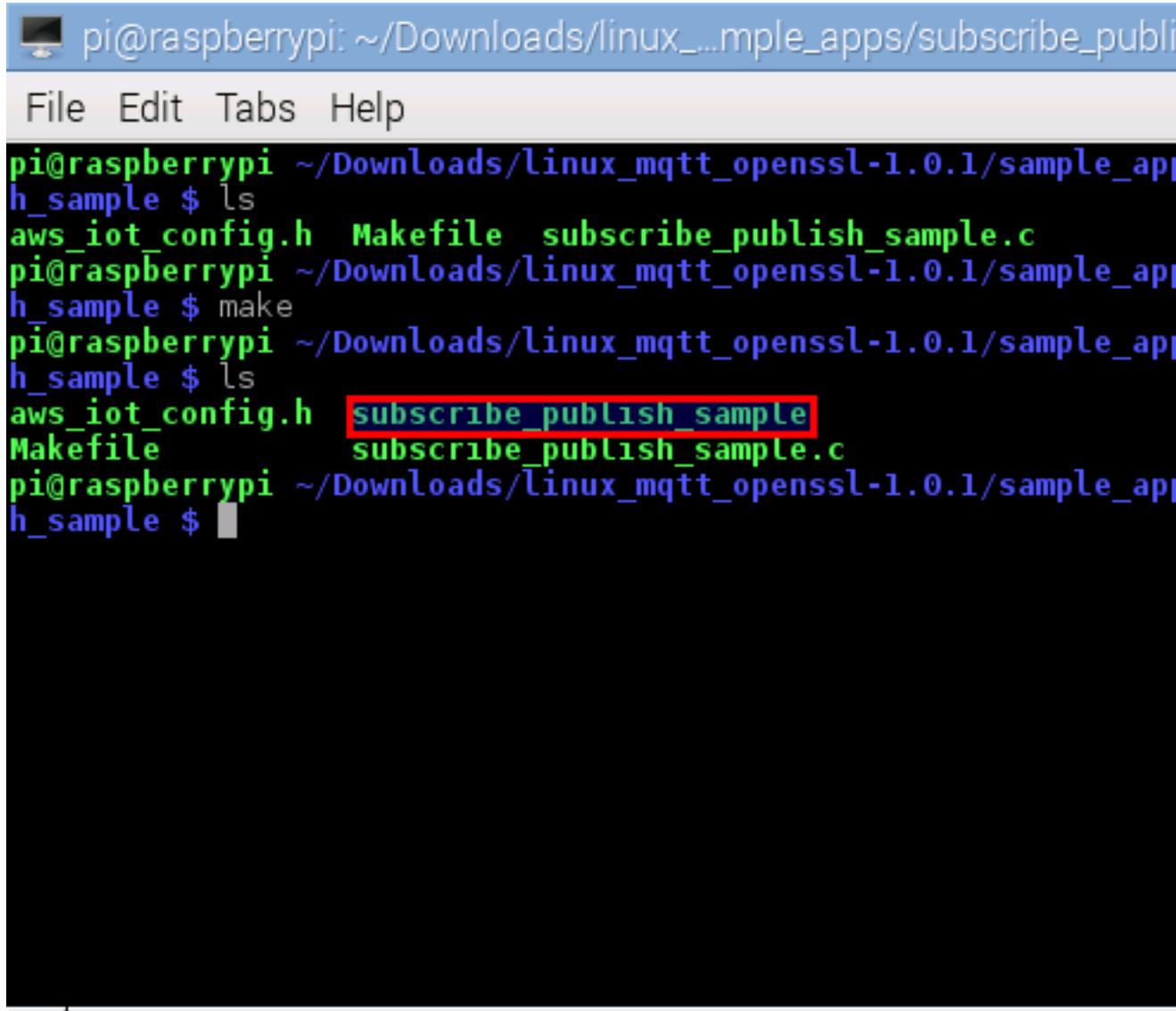
Votre clé privée.

Exécuter les exemples d'applications

Compilez `subscribe_publish_sample_app` avec le makefile inclus.

```
make -f Makefile
```

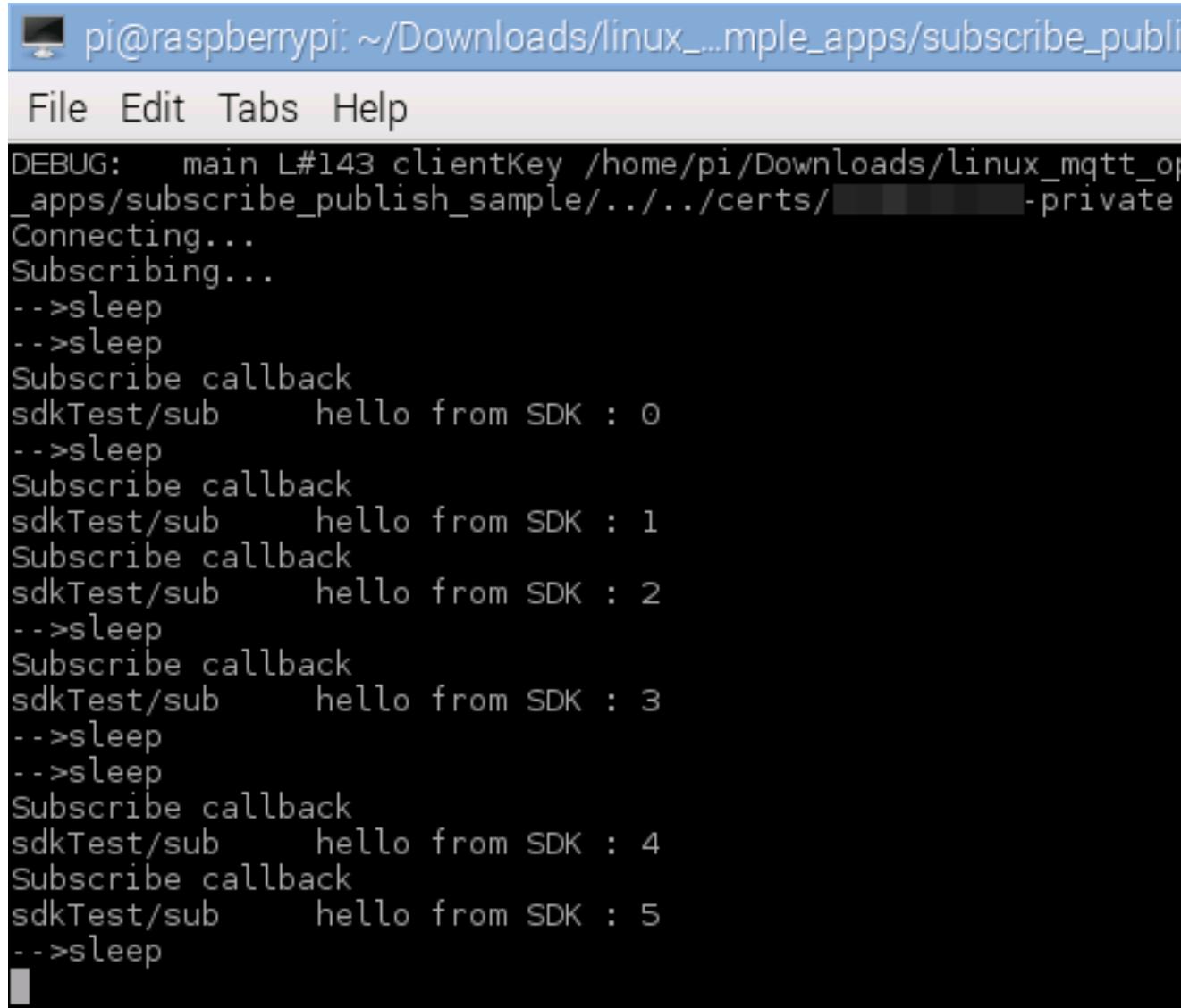
Cela génère un fichier exécutable.



The screenshot shows a terminal window titled "pi@raspberrypi: ~/Downloads/linux_mqtt_openssl-1.0.1/sample_app_h_sample \$". The window contains the following command-line session:

```
pi@raspberrypi ~$ ls
aws_iot_config.h Makefile subscribe_publish_sample.c
pi@raspberrypi ~$ make
pi@raspberrypi ~$ ls
aws_iot_config.h subscribe_publish_sample.c
pi@raspberrypi ~$
```

Maintenant, exéutez la commande `subscribe_publish_sample_app`. Vous devez visualiser des résultats similaires à ce qui suit :



pi@raspberrypi: ~/Downloads/linux_sample_apps/subscribe_publi

File Edit Tabs Help

```
DEBUG: main L#143 clientKey /home/pi/Downloads/linux_mqtt_op
      _apps/subscribe_publish_sample/....certs/ -private
      Connecting...
      Subscribing...
      -->sleep
      -->sleep
      Subscribe callback
      sdkTest/sub    hello from SDK : 0
      -->sleep
      Subscribe callback
      sdkTest/sub    hello from SDK : 1
      Subscribe callback
      sdkTest/sub    hello from SDK : 2
      -->sleep
      Subscribe callback
      sdkTest/sub    hello from SDK : 3
      -->sleep
      -->sleep
      Subscribe callback
      sdkTest/sub    hello from SDK : 4
      Subscribe callback
      sdkTest/sub    hello from SDK : 5
      -->sleep
```

Raspberry Pi est maintenant connecté à AWS IoT via le SDK pour les appareils pour C AWS IoT.

Démarrer avec AWS IoT sur Raspberry Pi et le SDK pour les appareils pour Javascript AWS IoT

Ce manuel fournit des instructions pour connecter Raspberry Pi à la plateforme AWS IoT et pour le configurer de sorte à l'utiliser avec le SDK pour les appareils pour Javascript AWS IoT. Après avoir suivi les étapes de ce manuel, vous serez en mesure de vous connecter à la plateforme AWS IoT et d'exécuter des exemples d'applications contenus dans le SDK.

Prérequis

- Carte Raspberry Pi entièrement configurée avec accès Internet

Pour plus d'informations sur la configuration de Raspberry Pi, consultez le [Guide de démarrage Raspberry Pi](#).

- Navigateur Chrome ou Firefox (Iceweasel)

Pour plus d'informations sur l'installation d'Iceweasel, consultez [les instructions contenues dans le Wiki Linux intégré](#).

Dans ce manuel, le matériel et les logiciels suivants sont utilisés :

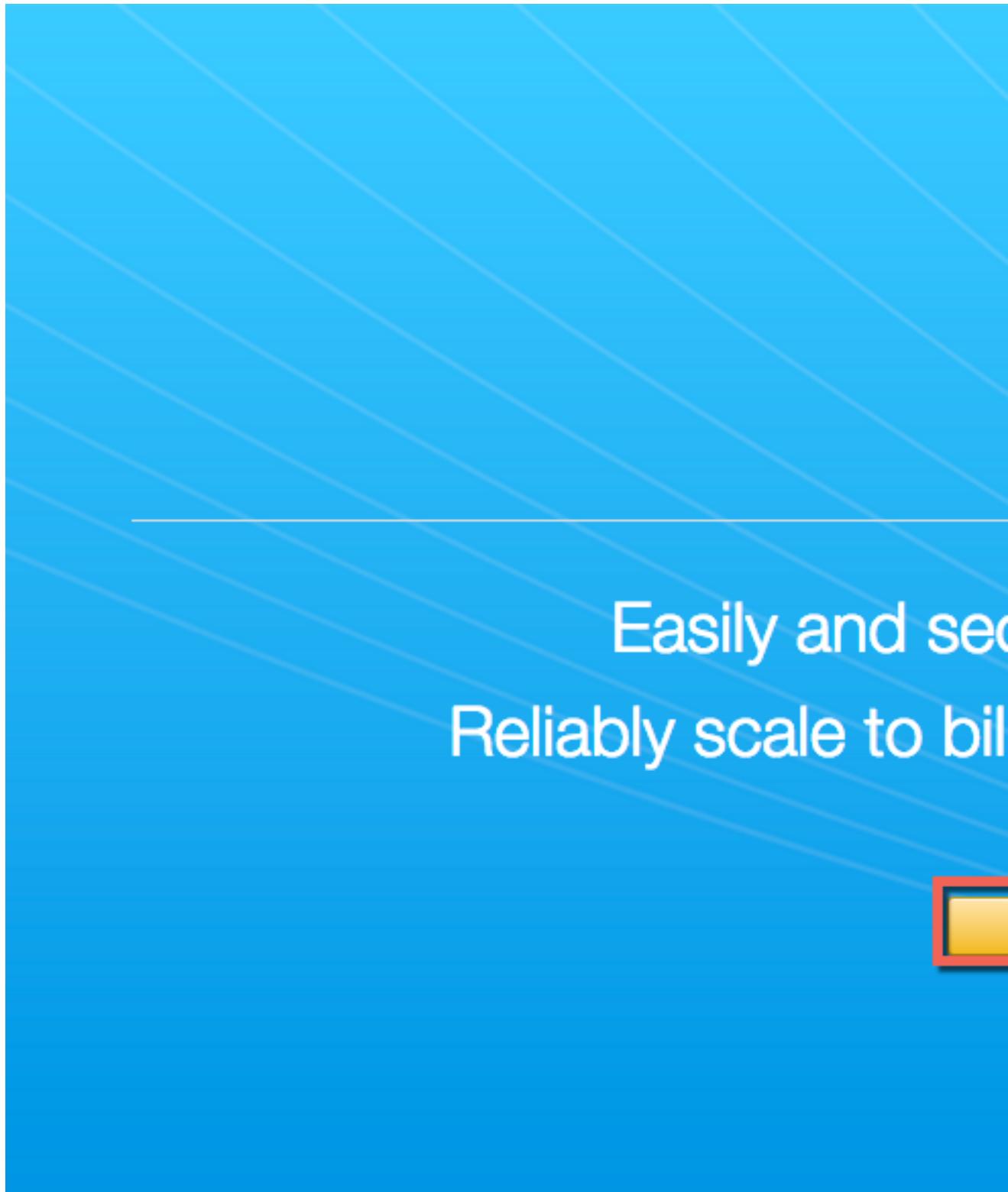
- [Raspberry Pi 2 Modèle B](#)
- [Raspbian Jessie](#)
- [Navigateur iceweasel](#)

Connexion à Raspberry Pi

Connectez-vous à la console AWS IoT

Mettez le Raspberry Pi sous tension et vérifiez que vous avez une connexion Internet.

Connectez-vous à AWS Management Console et ouvrez la console AWS IoT à partir de l'adresse <https://aws.amazon.com/iot>. Sur la page d'accueil, sélectionnez Démarrer avec AWS IoT:



S'il s'agit de votre première utilisation de la console AWS IoT, vous verrez deux boutons : Mise en route and Démarrer le didacticiel interactif:



AWS I

AWS IoT is a managed cloud platform
-- cars, light bulbs, sensor grids and more -- can
interact with cloud applications.

[Get started](#)

St

[Getting started documentation](#)

Choisissez Mise en route. La page suivante doit apparaître.

The screenshot shows the AWS IoT Resources page. At the top, there is a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the page title "Resources" is displayed, along with a "Close create panel" button. The main content area has a light blue background. It features two large buttons for creating resources: "Create a thing" (with a wrench icon) and "Create a rule" (with a network icon). Below these buttons, there is a filter section with a magnifying glass icon and the text "Filter by resource names or by resource type (below)". Underneath the filter, there is a row of four buttons: "All" (selected), "0/0 things", "0/0 rules", "0/0 certificates", and "0/0 policies".

Si vous ne voyez pas la bannière bleue avec les boutons Crée un objet, Crée une règle, Crée un certificat, and Crée une stratégie , cliquez sur le bouton Crée une ressource :

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the word "Resources" is displayed. To the right of "Resources", there's a button labeled "+ Create a resource" which is highlighted with a red rectangular border. Underneath the "Resources" heading, there's a search/filter bar with a magnifying glass icon and the placeholder text "Filter by resource names or by resource type (below)". Below the search bar, there are four categories: "All" (which is selected and highlighted with a blue oval), "0/0 things", "0/0 rules", "0/0 certificates", and "0/0 policies".

Créer et attacher un objet (dispositif)

Un objet représente un dispositif dont le statut ou les données sont stockés dans le cloud AWS IoT. Le service Thing Shadow conserve un fichier thing shadow pour chaque dispositif connecté à AWS IoT. Les thing shadows permettent d'accéder aux données d'état des objets et de les modifier.

Choisissez Créer un objet, tapez un nom pour l'objet, puis sélectionnez Créer:

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the main title "Resources" is displayed. In the top right corner of the main area, there's a button labeled "Close create panel" with a small "X" icon. On the right side of the main area, there are two buttons: "Create a thing" (with a robotic arm icon) and "Create a rule" (with a gear and chain icon). The "Create a thing" button is highlighted with a blue border. Below these buttons, the section title "Create a thing" is shown in large bold letters. A descriptive text follows: "Create a thing to represent your device in the cloud. This step creates a resource in the AWS IoT service that you can use to publish sensor data or receive commands from the cloud." Under this text, there's a form field with the label "Name" and the value "MyIoTButton". Further down, the section title "Attributes" is shown in large bold letters, followed by a descriptive text: "Next (optional), you can use thing attributes to describe the identity of your device." A "Add attribute" button is located below this text.

Resources

[Close create panel](#)

[Create a thing](#) [Create a rule](#)

Create a thing

Create a thing to represent your device in the cloud. This step creates a resource in the AWS IoT service that you can use to publish sensor data or receive commands from the cloud.

Name

Attributes

Next (optional), you can use thing attributes to describe the identity of your device.

Add attribute

Outre un message de confirmation, le bouton Afficher l'objet s'affiche :



Filter by resource names or by resource type (below)

All 1/1 things 0/0 rules 0/0 certificates 0/0 policies

MyNewThing

Choisissez Afficher l'objet pour présenter les informations concernant votre objet :

The screenshot shows the AWS IoT Resources page. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below the header, the word "Resources" is displayed in large, bold, dark blue text. To the right of "Resources", there's a button with a close icon and the text "Close create panel". In the center of the page, there are three main buttons: "Create a thing" (with a wrench icon), "Create a rule" (with a gear icon), and "Use my thing" (with a gear icon). The "Create a thing" button is highlighted with a thick blue border. Below these buttons, there's a message: "Your new thing has been created. Click 'View thing' to continue. From there, you can connect a device to this thing, or add a rule to it." To the right of this message is a large blue "View" button. The background of the main content area is light blue.



Filter by resource names or by resource type (below)

All

1/1 things

0/0 rules

0/0 CAs

0/0 certificates

0/0 policies

MyIoTButton

Cliquez sur le bouton Connexion d'un appareil pour télécharger une paire de clés et un certificat générée par AWS IoT :



AWS IoT

Resources

[✖ Close create panel](#)



Create a thing



Create a rule



Create a certificate

Your new thing has been created. Click 'View thing' to continue.

From there, you can connect a device to this thing, or add a rule when your thing publishes a message.

[View thing](#)



Filter by resource names or by resource type (below)

All

1/1 things

0/0 rules

0/0 certificates

0/0 policies

Select all

First

MyNewThing

Sur la Connexion d'un appareil , sélectionnez le SDK à utiliser, puisGénérer un certificat et une stratégie:



AWS IoT

Connect a device

Connect your device to one of our many supported SDKs.

Embedded C **NodeJS**

Arduino Yún

First, you will follow the steps to authenticate (what you do).

You can generate a device policy at any time.

Un certificat X.509 et une paire de clés sont générés ; activez le certificat X.509, puis créez une stratégie AWS IoT et attachez-la au certificat.

La page suivante s'affiche :



Connect a device

Connect your device to one of our many supported SDKs.

Embedded C **NodeJS**

Arduino Yún

Please download
any time, but

- [Download](#)
- [Download](#)
- [Download](#)

Créez un répertoire de travail appelé `deviceSDK` où stocker vos fichiers. Cliquez sur les liens pour télécharger vos clés publiques et privées et les certificats, puis enregistrez-les dans le répertoire `deviceSDK`.

Choisissez Confirmer et démarrer la connexion. La page suivante s'affiche :



Connect a device

Connect your device to one of our many supported SDKs.

Embedded C NodeJS

Arduino Yún

AWS IoT

Download on GitHub

- [OpenSSL](#)
- [mbed-TLS](#)

Set up the SDK

Add in the following code:

```
// Get from GitHub
// =====
#define AWS_IOT_MQTT_HOST "a391onaws.com"
#define AWS_IOT_MQTT_PORT 8883
#define AWS_IOT_MQTT_KEEPALIVE 60
#define AWS_IOT_MQTT_CACERT "/path/to/cacert.pem"
#define AWS_IOT_MQTT_CERT "/path/to/cert.pem"
#define AWS_IOT_MQTT_KEY "/path/to/key.pem"
// =====
```

Start one of the examples and observe the state of the shadow. Only one update is sent at the same time. If you send multiple updates at the same time (and client-side shadow synchronization is disabled), the client will ignore the second update.

Configurer le SDK pour les appareils AWS IoT pour l'environnement d'exécution Javascript

Pour pouvoir utiliser le SDK pour les appareils pour JavaScript AWS IoT, vous devez installer le nœud et l'outil de développement NGP sur votre Raspberry Pi. Ces packages ne sont pas installés par défaut.



Note

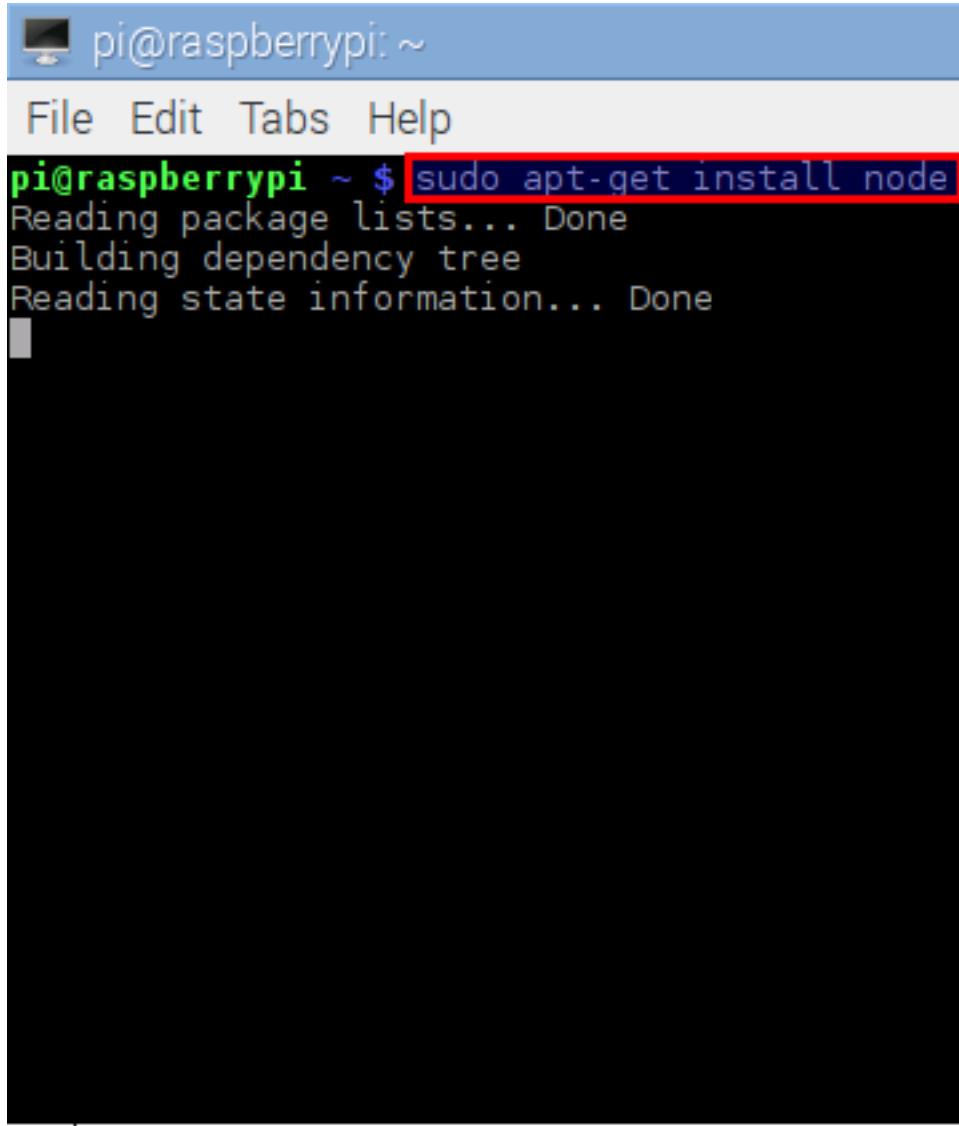
Avant de poursuivre, vous souhaiterez peut-être configurer le mapping du clavier de votre Raspberry Pi. Pour plus d'informations, consultez [Configurer le mapping du clavier Raspberry Pi](#).

Pour ajouter le référentiel nœud, ouvrez un terminal et exécutez la commande suivante :

```
curl -sLS https://apt.adafruit.com/add | sudo bash
```

The screenshot shows a terminal window titled "pi@raspberrypi: ~". The window has a blue header bar with the title and a white body. At the top, there is a menu bar with "File", "Edit", "Tabs", and "Help". Below the menu, the command "curl -sLS https://apt.adafruit.com/add | sudo bash" is typed into the terminal. The command is highlighted with a red rectangle. The rest of the terminal window is blank, showing a black background.

Pour installer le noeud, exécutez la commande `sudo apt-get install node`. Vous devez visualiser des résultats similaires à ce qui suit :



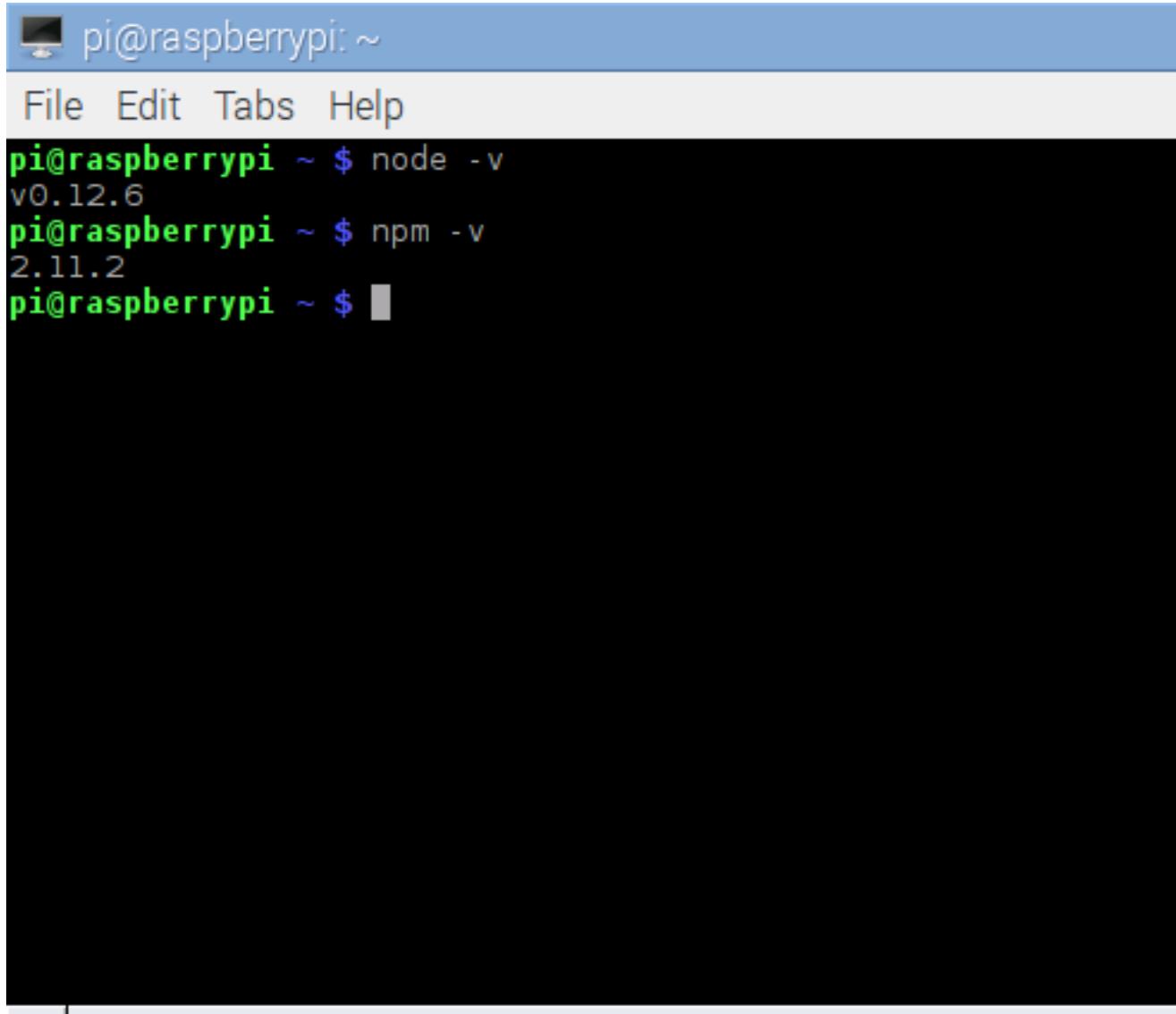
The screenshot shows a terminal window titled "pi@raspberrypi: ~". The window has a menu bar with "File", "Edit", "Tabs", and "Help". Below the menu, the command `pi@raspberrypi ~ $ sudo apt-get install node` is entered and highlighted with a red box. The terminal output shows the process of installing Node.js:

```
pi@raspberrypi ~ $ sudo apt-get install node
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Pour installer NGP, exécutez la commande `sudo apt-get install npm`. Vous devez visualiser des résultats similaires à ce qui suit :

The screenshot shows a terminal window with a blue header bar. The header bar displays a monitor icon, the text 'pi@raspberrypi: ~', and a menu bar with 'File Edit Tabs Help'. Below the header, the terminal prompt 'pi@raspberrypi ~ \$' is followed by the command 'sudo apt-get install npm'. The output of the command shows 'Reading package lists... Done' and 'Building dependency tree... 75%'. The rest of the terminal window is blank black space.

Pour vérifier l'installation du nœud et de NGP, exéutez la commande `node -v` and `npm -v`. Vous devez visualiser des résultats similaires à ce qui suit :



The screenshot shows a terminal window titled "pi@raspberrypi: ~". The window includes a menu bar with "File", "Edit", "Tabs", and "Help". Below the menu, the command-line interface displays the following output:

```
pi@raspberrypi ~ $ node -v
v0.12.6
pi@raspberrypi ~ $ npm -v
2.11.2
pi@raspberrypi ~ $ █
```

Installez le SDK pour les appareils pour Javascript AWS IoT

Vous allez maintenant porter le SDK dans Raspberry Pi. Sur la [SDK pour les appareils AWS IoT](#) , cliquez sur les liens Obtenez le code source en cliquant sur le lien GitHub :

≡ Menu



AWS re:Invent Announcements

PRODUCTS

- AWS IoT >
- How It Works >
- Pricing >
- Getting Started >
- FAQs >

AWS IoT

The AWS IoT hardware development gateway and

The AWS IoT guide with s IoT products

SDKs

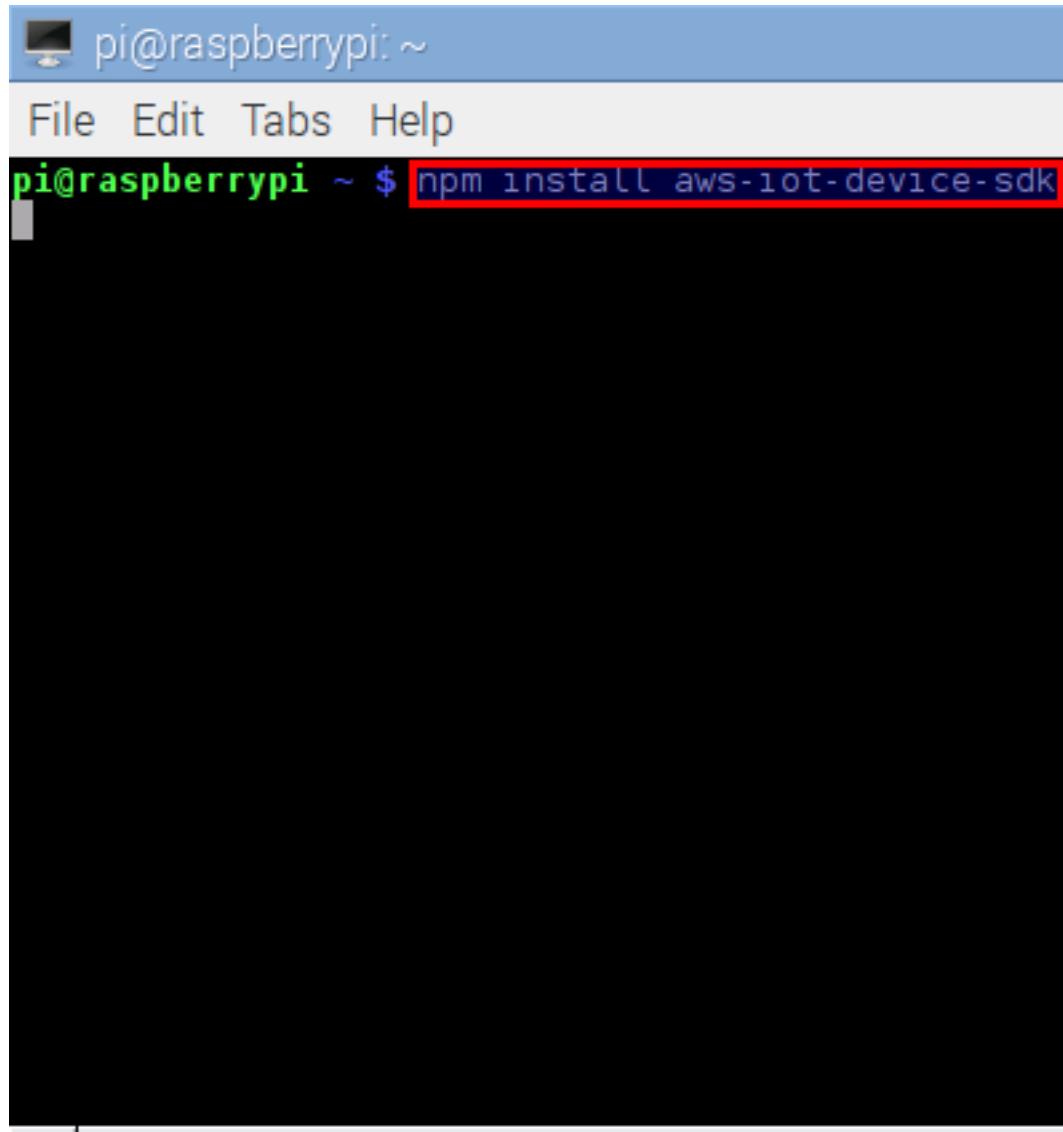
Embedded

Get source in

Developer Gui

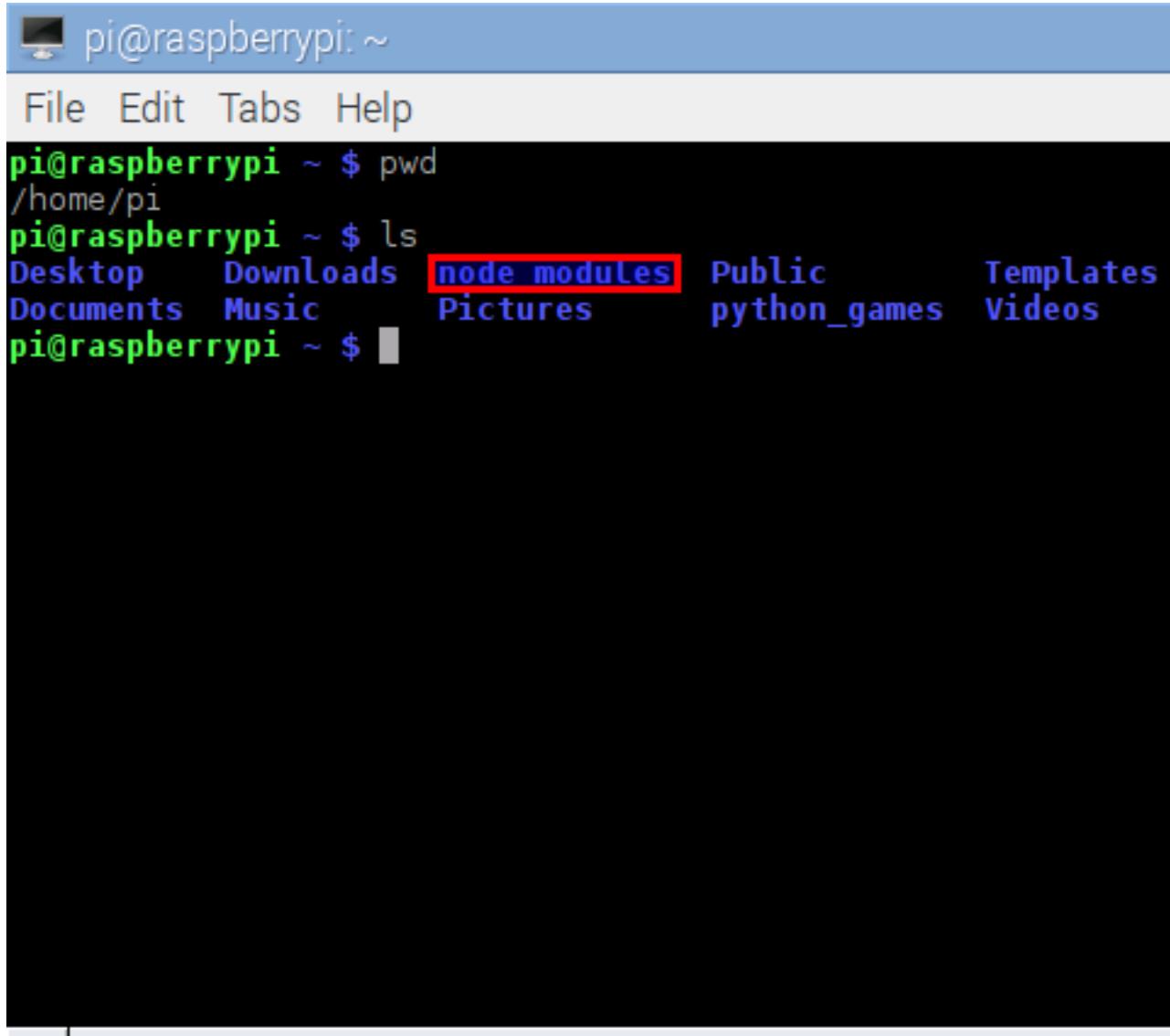
Porting Guide

Ouvrez une fenêtre de la console. Pour simplifier les choses, nous utiliserons NGP pour installer le SDK à partir du référentiel NGP :



The screenshot shows a terminal window titled "pi@raspberrypi: ~". The window has a blue header bar with the title and a menu bar below it labeled "File Edit Tabs Help". The main area of the terminal is black, and the command "npm install aws-iot-device-sdk" is visible at the top, highlighted with a red rectangle. The rest of the terminal window is blank.

Une fois l'installation terminée, vous devriez trouver le module installé dans le répertoire ~ (/home/pi est la valeur par défaut) :



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi: ~ $ pwd
/home/pi
pi@raspberrypi: ~ $ ls
Desktop Downloads node modules Public Templates
Documents Music Pictures python_games Videos
pi@raspberrypi: ~ $
```

Se préparer à exécuter les exemples d'applications

Le SDK pour les appareils pour Javascript AWS IoT contient des exemples d'applications que vous pouvez essayer. Pour les exécuter, vous devez configurer vos certificats et la clé privée.

Tapez `cd ~` pour accéder à votre répertoire de base. Créez un répertoire dans lequel votre certificat, la clé privée et le certificat d'autorité de certification seront stockés. Nommez ce répertoire `certs`.

Copiez votre certificat et la clé privée dans le répertoire. Téléchargez un certificat d'autorité de certification racine à partir de [ici](#). Copiez le texte du depuis le navigateur, collez-le dans un fichier, puis copiez-le dans le répertoire `certs`.

Vous devez configurer vos propres point de terminaison, clé privée et certificat. Si vous avez accès à une machine sur laquelle AWS CLI est installé, vous pouvez utiliser la commande `aws iot describe-endpoint` pour trouver l'URL de votre point de terminaison personnel. Sinon, dans la console AWS IoT, double cliquez sur MyNewThing, puis cherchez point de terminaison de l'API REST. Copiez tout après « `https://` » notamment « `.com` » :

The screenshot shows the AWS IoT Resources interface. At the top, there's a blue header bar with the AWS IoT logo and the word "AWS IoT". Below it, the main title "Resources" is displayed next to a "Create a resource" button with a plus sign icon. A search bar with a magnifying glass icon and the placeholder text "Filter by resource names or by resource type (below)" is present. Below the search bar, there are four categories with counts: "All" (1/1 things), "0/0 rules", "1/1 certificates", and "1/1 policies". To the right of these counts, there's a "Select all" checkbox. The bottom right corner of the interface shows the text "First".

Par défaut, les fichiers doivent être nommés comme suit :

- votre clé privée : `private.pem.key`
- votre certificat : `certificate.pem.crt`
- le certificat d'autorité de certification racine : `root-CA.crt`

Vous pouvez modifier le fichier `cmdline.js` de sorte à changer les noms par défaut utilisés pour chaque exemple.

```
default: { region: 'us-east-1', clientId: clientIdDefault, privateKey:  
'private.pem.key', clientCert: 'certificate.pem.crt', caCert: 'root-CA.crt',  
testMode: 1, reconnectPeriod: 3 * 1000, /* milliseconds */ delay: 4 * 1000 /  
* milliseconds * };
```

Exécuter les exemples d'applications

Vous pouvez maintenant exécuter les exemples grâce au nœud `examples/<ExempleDeVotreChoix>.js -f <emplacement certificats>` (en supposant que vous êtes sous `node_modules/aws-iot-device-sdk/`). Dans ce cas, l'emplacement des certificats doit être `~/certs/`. Vous pouvez spécifier l'emplacement des certificats et votre propre adresse d'hôte grâce aux options de ligne de commande. Pour plus d'informations, consultez [Certificats](#).

Raspberry Pi est maintenant connecté à AWS IoT via le SDK pour les appareils pour Javascript AWS IoT.

Supervision de AWS IoT

La supervision constitue une part importante de la gestion de la fiabilité, de la disponibilité et des performances de vos instances AWS IoT et de vos solutions AWS. Vous devez recueillir les données de supervision de toutes les parties de votre solution AWS de telle sorte que vous puissiez déboguer plus facilement une éventuelle défaillance à plusieurs points. Avant de commencer la supervision de AWS IoT, vous devez créer un plan de supervision qui contient les réponses aux questions suivantes :

- Quels sont les objectifs de la supervision ?
- Quelles sont les ressources à surveiller ?
- A quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de supervision utiliser ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

L'étape suivante consiste à établir une référence de performances normale de AWS IoT dans votre environnement, en mesurant la performance à divers moments et dans diverses conditions de charge. Comme vous surveillez AWS IoT, conservez les données de supervision historiques afin de les comparer aux données de performances actuelles, d'identifier les modèles de fonctionnement normal et les anomalies de performances et de concevoir des méthodes pour résoudre les problèmes.

Par exemple, si vous utilisez Amazon EC2, vous pouvez superviser l'utilisation de l'UC, les E/S disque et l'utilisation réseau de vos instances. Lorsque les performances se trouvent en dehors de votre référence établie, il se peut que vous ayez besoin de reconfigurer l'instance ou de l'optimiser pour réduire l'utilisation de l'UC, améliorer les E/S disque ou réduire le trafic réseau.

Pour établir une référence, vous devez, au moins, superviser les éléments suivants :

- PublishIn.Success
- PublishOut.Success
- Subscribe.Success
- Ping.Success
- Connect.Success
- GetThingShadow.Accepted
- UpdateThingShadow.Accepted
- DeleteThingShadow.Accepted
- RulesExecuted

Rubriques

- [Outils de supervision \(p. 212\)](#)
- [Supervision avec Amazon CloudWatch \(p. 213\)](#)
- [Journalisation des appels d'API AWS IoT avec AWS CloudTrail \(p. 216\)](#)

Outils de supervision

AWS fournit différents outils que vous pouvez utiliser pour surveiller AWS IoT. Vous pouvez configurer certains outils pour qu'ils effectuent la supervision automatiquement, tandis que d'autres nécessitent une intervention manuelle. Nous vous recommandons d'automatiser le plus possible les tâches de supervision.

Outils de supervision automatique

Vous pouvez utiliser les outils de supervision automatique suivants pour surveiller AWS IoT et être informé en cas de problème :

- Alarmes Amazon CloudWatch – surveillez une seule métrique sur une durée définie et exécutez une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de durées. L'action est une notification envoyée à une rubrique Amazon Simple Notification Service (Amazon SNS) ou une stratégie Auto Scaling. Les alarmes CloudWatch n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier : l'état doit avoir changé et été maintenu pendant un certain nombre de périodes. Pour plus d'informations, consultez [Supervision avec Amazon CloudWatch \(p. 213\)](#).
- Amazon CloudWatch Logs – supervisez vos fichiers journaux, stockez-les et accédez-y à partir de AWS CloudTrail ou d'autres sources. Pour plus d'informations, consultez la section [Gestion des fichiers journaux](#) dans le manuel &guide-cw-dev;
- Amazon CloudWatch Events – mettez en correspondance les événements et transmettez-les à un ou plusieurs flux ou fonctions cibles pour apporter des modifications, capturer des informations d'état et prendre des mesures correctives. Pour plus d'informations, consultez la section [Utilisation d'événements](#) dans le manuel Guide de l'utilisateur Amazon CloudWatch.
- Gestion des journaux AWS CloudTrail – partagez des fichiers journaux entre les comptes, surveillez les fichiers journaux CloudTrail en temps réel en les envoyant à CloudWatch Logs, écrivez des applications de traitement des journaux en Java et assurez-vous que vos fichiers journaux n'ont pas changé après leur livraison par CloudTrail. Pour plus d'informations, consultez la section [Utilisation des fichiers journaux CloudTrail](#) dans le manuel AWS CloudTrail User Guide.

Outils de supervision manuelle

La supervision de AWS IoT implique également de surveiller manuellement les éléments que les alarmes CloudWatch ne couvrent pas. Les tableaux de bord des consoles AWS IoT, CloudWatch et AWSfournissent un aperçu de l'état de votre environnement AWS. Nous recommandons de vérifier également les fichiers-journaux sur AWS IoT.

- Le tableau de bord AWS IoT présente :
 - Certificats CA
 - Certificats
 - Stratégies
 - Règles
 - Objets
- La page d'accueil CloudWatch présente :
 - Alarmes et statuts en cours

- Graphiques des alarmes et des ressources
- Statut d'intégrité du service

De plus, vous pouvez utiliser CloudWatch pour effectuer les tâches suivantes :

- Créer [des tableaux de bord personnalisés](#) pour surveiller les services de votre choix
- Données de métriques de graphiques pour résoudre les problèmes et découvrir les tendances
- Rechercher et parcourir toutes les métriques des ressources AWS
- Créer et modifier des alarmes pour être informé des problèmes

Supervision avec Amazon CloudWatch

Vous pouvez surveiller AWS IoT grâce à CloudWatch, qui recueille et traite les données brutes de AWS IoT en métriques lisibles et disponibles presque en temps réel. Ces statistiques sont enregistrées pour une durée de deux semaines et, par conséquent, vous pouvez accéder aux informations historiques et acquérir un meilleur point de vue de la façon dont votre service ou application web s'exécute. Par défaut, les données des métriques AWS IoT sont automatiquement envoyées à CloudWatch toutes les minutes. Pour plus d'informations, consultez [Que sont Amazon CloudWatch, Amazon CloudWatch Events et Amazon CloudWatch Logs ?](#) dans le manuel Guide de l'utilisateur Amazon CloudWatch.

Rubriques

- [Métriques et dimensions AWS IoT \(p. 213\)](#)
- [Comment utiliser les métriques AWS IoT ? \(p. 214\)](#)
- [Création d'alarmes CloudWatch pour surveiller AWS IoT \(p. 214\)](#)

Métriques et dimensions AWS IoT

Lorsque vous interagissez avec AWS IoT, il envoie les métriques et les dimensions suivantes à CloudWatch toutes les minutes. Vous pouvez utiliser les procédures suivantes pour afficher les métriques de AWS IoT.

Pour afficher les métriques grâce à la console CloudWatch

Les métriques sont d'abord regroupées par namespace de service, puis par les différentes combinaisons de dimension au sein de chaque namespace.

1. Ouvrez la console CloudWatch à l'adresse <https://console.aws.amazon.com/cloudwatch/>.
2. Dans le volet de navigation, choisissez Métriques.
3. Dans le volet Métriques CloudWatch par catégorie , sous la catégorie de métriques de AWS IoT, sélectionnez une catégorie de mesures, puis, dans le volet supérieur, faites défiler vers le bas pour afficher la liste complète des métriques.

Pour afficher les métriques grâce à AWS CLI

- A partir d'une invite de commande, utilisez la commande suivante :

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch affiche les mesures suivantes pour AWS IoT :
&cloudwatch_viewing_aws_iot ;

Comment utiliser les métriques AWS IoT ?

Les métriques présentées par AWS IoT fournissent des informations qui permettent divers types d'analyses. Les cas d'utilisation suivants sont basés sur un scénario où vous avez dix objets qui se connectent à Internet une fois par jour. Chaque jour :

- Dix objets se connectent à AWS IoT en même temps.
- Chaque objet s'abonne à un filtre de rubrique, puis attend une heure avant de déconnecter. Au cours de cette période, les objets communiquent entre eux pour en savoir plus sur l'état du monde.
- Chaque objet publie sa perception, d'après les données qu'il vient de détecter avec `UpdateThingShadow`.
- Chaque objet se déconnecte de AWS IoT.

Voici quelques suggestions pour vous aider à démarrer, qui ne forment pas une liste exhaustive.

- [Comment puis-je être informé si mes objets ne se connectent pas chaque jour ? \(p. 214\)](#)
- [Comment puis-je être informé si mes objets ne publient pas de données chaque jour ? \(p. 215\)](#)
- [Comment puis-je être informé si les mises à jour de mon thing shadow sont rejetées chaque jour ? \(p. 215\)](#)

Création d'alarmes CloudWatch pour surveiller AWS IoT

Vous pouvez créer une alarme CloudWatch qui envoie un message Amazon SNS lorsque l'alarme change d'état. Un alarme surveille une seule métrique sur une durée que vous définissez et exécute une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de durées. L'action est une notification envoyée à une rubrique Amazon SNS ou une stratégie Auto Scaling. Les alarmes appellent les actions pour les modifications d'état soutenues uniquement. Les alarmes CloudWatch n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier : l'état doit avoir changé et été maintenu pendant un certain nombre de durées.

Comment puis-je être informé si mes objets ne se connectent pas chaque jour ?

1. Créer une rubrique Amazon SNS `arn:aws:sns:us-east-1:123456789012:things-not-connecting-successfully`.

Pour plus d'informations, consultez [Configurer Amazon Simple Notification Service](#).

2. Créez l'alarme.

```
Prompt>aws cloudwatch put-metric-alarm \ --alarm-name ConnectSuccessAlarm
\ --alarm-description "Alarm when my Things don't connect successfully"
\ --namespace AWS/IoT \ --metric-name Connect.Success \ --dimensions
Name=Protocol,Value=MQTT \ --statistic Sum \ --threshold 10 \ --
comparison-operator LessThanThreshold \ --period 86400 \ --unit
Count \ --evaluation-periods 1 \ --alarm-actions arn:aws:sns:us-
east-1:1234567890:things-not-connecting-successfully
```

```
Prompt>aws cloudwatch put-metric-alarm \ --alarm-name ConnectSuccessAlarm
\ --alarm-description "Alarm when my Things don't connect successfully"
\ --namespace AWS/IoT \ --metric-name Connect.Success \ --dimensions
```

```
Name=Protocol,Value=MQTT \ --statistic Sum \ --threshold 10 \ --
comparison-operator LessThanThreshold \ --period 86400 \ --unit
Count \ --evaluation-periods 1 \ --alarm-actions arn:aws:sns:us-
east-1:1234567890:things-not-connecting-successfully
```

3. Testez l'alarme.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --
state-reason "initializing" --state-value OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --
state-reason "initializing" --state-value ALARM
```

Comment puis-je être informé si mes objets ne publient pas de données chaque jour ?

1. Créer une rubrique Amazon SNS arn:aws:sns:us-east-1:123456789012:things-not-publishing-data.
Pour plus d'informations, consultez [Configurer Amazon Simple Notification Service](#).
2. Créez l'alarme.

```
Prompt>aws cloudwatch put-metric-alarm \ --alarm-name
PublishInSuccessAlarm\ --alarm-description "Alarme quand mes objets
ne publient pas leurs données" \ --namespace AWS/IoT \ --metric-name
PublishIn.Success \ --dimensions Name=Protocol,Value=MQTT \ --statistic
Sum \ --threshold 10 \ --comparison-operator LessThanThreshold \ --
period 86400 \ --unit Count \ --evaluation-periods 1 \ --alarm-actions
arn:aws:sns:us-east-1:1234567890:things-not-publishing-data
```

3. Testez l'alarme.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --
state-reason "initializing" --state-value OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --
state-reason "initializing" --state-value ALARM
```

Comment puis-je être informé si les mises à jour de mon thing shadow sont rejetées chaque jour ?

1. Créer une rubrique Amazon SNS, arn:aws:sns:us-east-1:1234567890:things-shadow-updates-rejected.
Pour plus d'informations, consultez [Configurer Amazon Simple Notification Service](#).
2. Créez l'alarme.

```
Prompt>aws cloudwatch put-metric-alarm \ --alarm-name
UpdateThingShadowSuccessAlarm \ --alarm-description "Alarme
quand mes mises à jour Things Shadow sont rejetées" \ --namespace
```

```
AWS/IoT \ --metric-name UpdateThingShadow.Success \ --dimensions  
Name=Protocol,Value=MQTT \ --statistic Sum \ --threshold 10 \ --  
comparison-operator LessThanThreshold \ --period 86400 \ --unit  
Count \ --evaluation-periods 1 \ --alarm-actions arn:aws:sns:us-  
east-1:1234567890:things-shadow-updates-rejected
```

3. Testez l'alarme.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name  
UpdateThingShadowSuccessAlarm --state-reason "initializing" --state-value  
OK
```

```
Prompt>aws cloudwatch set-alarm-state --alarm-name  
UpdateThingShadowSuccessAlarm --state-reason "initializing" --state-value  
ALARM
```

Journalisation des appels d'API AWS IoT avec AWS CloudTrail

AWS IoT est intégré dans CloudTrail, un service qui enregistre tous les appels APIAWS IoT et transmet les fichiers-journaux dans un compartiment Amazon S3 que vous spécifiez. CloudTrail enregistre les appels d'API provenant de la console AWS IoT ou de votre code aux API AWS IoT. Les informations collectées par CloudTrail vous permettent de déterminer quelle demande a été envoyée à AWS IoT, l'adresse IP source à partir de laquelle la demande a été effectuée, qui a effectué la demande, quand, etc.

Pour en savoir plus sur CloudTrail, y compris la façon de le configurer et de l'activer, consultez le manuel [AWS CloudTrail User Guide](#).

Informations relatives à AWS IoT dans CloudTrail

Lorsque la journalisation CloudTrailest activée dans votre compte AWS, les appels d'API passés aux actions AWS IoT sont suivis dans les fichiers journaux CloudTrail, où ils sont écrits avec d'autres enregistrements de service AWS. CloudTrail détermine quand créer un fichier et écrire dedans en fonction d'une période et d'une taille de fichier.

Toutes les actions AWS IoT sont consignées par CloudTrail et sont documentées dans la [Référence d'API AWS IoT](#). Par exemple, les appels aux sections CreateThing, ListThingset ListTopicRules génèrent des entrées dans les fichiers journaux de CloudTrail.

Chaque entrée du journal contient des informations sur la personne qui a générée la demande. Les informations d'identité utilisateur dans l'entrée de journal permettent déterminer les éléments suivants :

- Si la demande a été effectuée avec les informations d'identification utilisateur racine ou IAM.
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré.
- Si la demande a été effectuée par un autre service AWS.

Pour plus d'informations, consultez la [Élément userIdentity CloudTrail](#).

Vous pouvez stocker vos fichiers journaux dans votre compartiment Amazon S3 aussi longtemps que vous le souhaitez, mais vous pouvez également définir des règles de cycle de vie Amazon S3 pour

archiver ou supprimer automatiquement les fichiers journaux. Par défaut, vos fichiers journaux sont chiffrés avec le chiffrement côté serveur (SSE) d'Amazon S3.

Si vous souhaitez être averti de la remise des fichiers-journaux, vous pouvez configurer CloudTrail pour qu'il publie des notifications Amazon SNS lorsque de nouveaux fichiers journaux sont remis. Pour plus d'informations, consultez [Configuration des Notifications de Amazon SNS pour CloudTrail](#).

Vous pouvez également regrouper des fichiers journaux AWS IoT provenant de plusieurs régions AWS et de plusieurs comptes AWS dans un compartiment Amazon S3 unique.

Pour plus d'informations, consultez [Recevoir les fichiers journaux CloudTrail de plusieurs régions](#) et [Recevoir les fichiers journaux CloudTrail de plusieurs comptes](#).

Présentation des entrées des fichiers journaux AWS IoT

Les fichiers journaux CloudTrail peuvent contenir une ou plusieurs des entrées de journal. Chaque entrée répertorie plusieurs événements au format JSON. Une entrée de journal représente une demande individuelle à partir d'une source quelconque et comprend des informations sur l'action demandée, sur tous les paramètres, les paramètres de la demande, etc. Les entrées de journal ne sont pas des séries ordonnées retracant les appels aux API publics, elles ne suivent aucun ordre précis.

L'exemple suivant montre une entrée de journal CloudTrail qui illustre l'action `AttachPrincipalPolicy`.

```
{ "timestamp": "1460159496", "AdditionalEventData": "", "Annotation": "",  
  "ApiVersion": "", "ErrorCode": "", "ErrorMessage": "", "EventID": "8bff4fed-  
c229-4d2d-8264-4ab28a487505", "EventName": "AttachPrincipalPolicy",  
  "EventTime": "2016-04-08T23:51:36Z", "EventType": "AwsApiCall",  
  "ReadOnly": "", "RecipientAccountList": "", "RequestID": "d4875df2-fde4-11e5-  
b829-23bf9b56cbcd", "RequestParamters": { "principal": "arn:aws:iot:us-  
east-1:123456789012:cert/528ce36e8047f6a75ee51ab7beddb4eb268ad41d2ea881a10b67e8e76924d894",  
    "policyName": "ExamplePolicyForIoT" }, "Resources": "", "ResponseElements": "",  
  "SourceIpAddress": "52.90.213.26", "UserAgent": "aws-internal/3",  
  "UserIdentity": { "type": "AssumedRole", "principalId": "AKIAI44QH8DHBEXAMPLE",  
    "arn": "arn:aws:sts::12345678912:assumed-role/iotmonitor-  
us-east-1-beta-InstanceRole-1C5T1YCYMHPYT/i-35d0a4b6",  
    "accountId": "222222222222", "accessKeyId": "access-key-id",  
    "sessionContext": { "attributes": { "mfaAuthenticated": "false",  
      "creationDate": "Fri Apr 08 23:51:10 UTC 2016" }, "sessionIssuer":  
      { "type": "Role", "principalId": "AKIAI44QH8DHBEXAMPLE",  
        "arn": "arn:aws:iam::123456789012:role/executionServiceEC2Role/iotmonitor-  
us-east-1-beta-InstanceRole-1C5T1YCYMHPYT", "accountId": "222222222222",  
        "userName": "iotmonitor-us-east-1-InstanceRole-1C5T1YCYMHPYT" } },  
    "invokedBy": { "serviceAccountId": "111111111111" } }, "VpcEndpointId": "" }
```

Résolution des problèmes de AWS IoT

Les informations suivantes peuvent vous aider à résoudre les problèmes courants dans AWS IoT.

Tâches

- [Diagnostic des problèmes de connectivité \(p. 218\)](#)
- [Configuration de CloudWatch Logs \(p. 218\)](#)
- [Diagnostic des problèmes de règles \(p. 223\)](#)
- [Diagnostic des problèmes de Thing Shadows \(p. 223\)](#)

Diagnostic des problèmes de connectivité

Authentification

Comment mes dispositifs authentifient-ils des points de terminaison AWS IoT ?

Ajoutez le certificat d'autorité de certification (CA) AWS IoT au référentiel d'approbations de votre client. Vous pouvez télécharger le certificat CA [ici](#).

Comment puis-je valider un certificat correctement configuré ?

Utilisez la commande OpenSSL s_client pour tester une connexion à un point de terminaison AWS IoT :

```
openssl s_client -connect custom_endpoint.iot.us-east-1.amazonaws.com:8443  
-CAfile CA.pem -cert cert.pem -key privateKey.pem
```

Autorisation

J'ai reçu un réponse PUBNACK ou SUBNACK de l'agent. Que puis-je faire ?

Assurez-vous qu'il y ait une stratégie attachée au certificat que vous utilisez pour appeler AWS IoT. Toutes les opérations de publication/abonnement sont rejetées par défaut.

Configuration de CloudWatch Logs

Puisque les messages provenant de vos dispositifs sont transmis via l'agent de messages et le moteur de règles, AWS IoT envoie des événements d'avancement concernant chaque message. Vous pouvez

vous inscrire pour voir ces événements dans CloudWatch Logs. Pour plus d'informations, consultez [CloudWatch Logs](#).



Note

Avant d'activer la journalisation AWS IoT, assurez-vous de comprendre les autorisations d'accès à CloudWatch Logs de votre compte AWS. Les utilisateurs détenant un accès à CloudWatch Logs pourront consulter les informations de débogage à partir de vos dispositifs.

Configuration d'un rôle IAM pour la journalisation

Utilisez la console IAM pour créer un rôle de journalisation.

Créer un rôle IAM pour la journalisation

Les documents de stratégie suivants fournissent la stratégie de rôle et la stratégie d'approbation qui permettent à AWS IoT d'envoyer des journaux à CloudWatch en votre nom.

Stratégie de rôle :

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "logs>CreateLogGroup", "logs>CreateLogStream", "logs>PutLogEvents", "logs>PutMetricFilter", "logs>PutRetentionPolicy" ], "Resource": [ "*" ] } ] }
```

Stratégie d'approbation :

```
{ "Version": "2012-10-17", "Statement": [ { "Sid": "", "Effect": "Allow", "Principal": { "Service": "iot.amazonaws.com" }, "Action": "sts>AssumeRole" } ] }
```

Enregistrer le rôle de journalisation auprès de AWS IoT

Utilisez la console AWS IoT ou la commande CLI suivante pour vous inscrire au rôle de journalisation auprès de AWS IoT.

```
aws iot setLoggingOptions --logging-options-payload  
roleArn="arn:aws:iam::<notre-numéro-compte-aws>:role/  
IoTLoggingRole",logLevel="INFO"
```

Le niveau de journalisation peut être DEBUG, INFO, ERROR ou DISABLED :

- DÉBOGAGE fournit les informations les plus détaillées concernant l'activité de AWS IoT.
- INFO fournit une synthèse de la plupart des actions. C'est suffisant pour la plupart des utilisateurs.
- ERREUR fournit uniquement les cas d'erreur.
- DÉSACTIVÉE supprime la journalisation, mais conserve votre rôle de journalisation.

Format d'entrée de journal CloudWatch

Chaque entrée du journal contient les informations suivantes :

Événement

Décris les actions qui ont lieu dans AWS IoT.

TimeStamp

Heure à laquelle le journal a été généré.

TraceId

Identifiant généré de façon aléatoire pour une demande entrante qui peut être utilisé pour filtrer tous les journaux correspondants à un message entrant.

PrincipalId

Empreinte de certificat ou nom d'un objet, selon le point de terminaison (MQTT ou HTTP) ayant reçu la demande d'un dispositif.

LogLevel

Niveau de journalisation. Peut être DEBUG, INFO, ERROR ou WARN.

Nom de la rubrique

Nom de rubrique MQTT, qui est ajouté à une entrée lorsqu'un message MQTT de publication ou d'abonnement est reçu.

ClientId

ID du client qui a envoyé un message MQTT.

ThingId

Identifiant d'objet ajouté à une entrée lorsqu'une demande est envoyée à un point de terminaison HTTP pour mettre à jour ou supprimer l'état de l'objet.

RuleId

Identifiant de la règle, qui contient l'ID d'une règle lorsque celle-ci est déclenchée.

Niveau de journalisation

Le niveau de journalisation spécifie les types de journaux qui seront générés.

DEBUG

Informations qui peuvent être utiles lors du débogage d'un problème.

Les journaux comprendra des informations DEBUG, INFO, ERROR et WARN.

ERROR

Toute erreur qui entraîne l'échec d'une opération.

Les journaux contiendront uniquement des informations ERROR.

INFO

Informations générales sur le flux des objets.

Les journaux contiendront des informations INFO, ERROR et WARN.

WARN

Tout ce qui peut éventuellement entraîner des incohérences dans le système, mais qui n'entraîne pas nécessairement l'échec de l'opération.

Les journaux contiendront des informations ERROR et WARN.

Journalisation des événements et codes d'erreur

Cette section présente les événements de journalisation et les codes d'erreur envoyés par AWS IoT.

Identité et sécurité

Nom de l'opération/événement	Description
Réussite de l'authentification	Certificat authentifié avec succès.
Échec de l'authentification	L'authentification d'un certificat a échoué.

Identités et des codes d'erreur de sécurité

Code d'erreur	Description de l'erreur
401	Non autorisé

Agent de messages

Nom de l'opération/événement	Description
Publication MQTT	Nombre de publications MQTT reçues.
Abonnement MQTT	Nombre d'abonnements MQTT reçus.
Connexion MQTT	Nombre de connexions MQTT reçues.
Déconnexion MQTT	Nombre de déconnexions MQTT reçues.
HTTP/1.1 POST	Nombre de messages HTTP/1.1 POST reçus.
HTTP/1.1 GET	Nombre de messages HTTP/1.1 GET reçus.
Méthode HTTP/1.1 non prises en charge	Utilisé lorsqu'un message contient une erreur de syntaxe ou que l'action (HTTP PUT/DELETE/) est interdite.
Message HTTP incorrect	La connexion a été interrompue en raison d'un message HTTP incorrect.
Message MQTT incorrect	La connexion a été interrompue en raison d'un message MQTT incorrect.
Échec d'autorisation	Ce client a tenté publier dans une rubrique ou de s'y abonner sans autorisation.
Le package dépasse la taille de charge utile maximale	Ce client a tenté de publier une charge utile qui dépasse la limite supérieure de l'agent de messages.

Codes d'erreur de l'agent de messages

Code d'erreur	Description de l'erreur
400	Demande erronée
401	Accès non autorisé
403	Accès interdit
503	Service non disponible

Événements du moteur de règles

Nom de l'opération/événement	Description
MessageReceived	Demande de rubrique reçue.
DynamoActionSuccess	Enregistrement DynamoDB placé avec succès.

Nom de l'opération/événement	Description
DynamoActionFailure	Échec de placement de l'enregistrement DynamoDB.
KinesisActionSuccess	Message Amazon Kinesis publié avec succès.
KinesisActionFailure	Échec de publication d'un message Amazon Kinesis.
LambdaActionSuccess	Fonction Lambda appelée avec succès.
LambdaActionFailure	Échec d'appel de la fonction Lambda.
RepublishActionSuccess	Message republié.
MessageReceived	Demande de rubrique reçue.
RepublishActionFailure	Échec de republication du message.
S3ActionSuccess	Objet Amazon S3 placé avec succès.
S3ActionFailure	Échec de placement de l'objet Amazon S3.
SNSActionSuccess	Publication réussie dans la rubrique Amazon SNS.
SNSActionFailure	Échec de publication dans la rubrique Amazon SNS.
SQSActionSuccess	Message envoyé avec succès à Amazon SQS.
SQSActionFailure	Échec d'envoi du message à Amazon SQS.

Événements Thing Shadow

Nom de l'opération/événement	Description
UpdateThingState	L'état d'un objet est mis à jour via HTTP ou MQTT.
DeleteThing	Un objet est supprimé.

Codes d'erreur Thing Shadow

Code d'erreur	Description de l'erreur
400	Requête erronée.
401	Accès non autorisé.
403	Accès interdit.
404	Introuvable.
409	Conflit.
413	Demande trop longue.
422	Impossible de traiter la demande.

Code d'erreur	Description de l'erreur
429	Nombre de requêtes trop élevé.
500	Erreur interne.
503	Service non disponible.

Diagnostic des problèmes de règles

CloudWatch Logs est idéal pour résoudre les problèmes que vous pouvez rencontrer avec les règles. Lorsque vous activez CloudWatch Logs pour AWS IoT, vous obtenez une meilleure visibilité des règles qui sont déclenchées, ainsi que de leur réussite ou échec. Vous obtenez également des informations concernant la correspondance ou non des conditions de clause WHERE.

Le problème le plus fréquent est celui de l'autorisation. Dans ce cas, les journaux vous indiquent que votre rôle n'est pas autorisé à effectuer des opérations AssumeRole sur la ressource.

Pour consulter les journaux CloudWatch (console)

1. Dans AWS Management Console, accédez à la console CloudWatch.
2. Choisissez Journaux, puis sélectionnez le plan AWSIoTLogs Groupe de journaux dans la liste.
3. Sur la Flux de la page AWSIoTLogs qui s'affiche, vous trouverez un flux de journal pour chacun de vos mandataires (certificat X.509, utilisateur IAM ou identité Amazon Cognito) qui a passé un appel dans AWS IoT avec votre compte.

Pour plus d'informations, consultez [CloudWatch Logs](#).

Les services externes sont contrôlés par l'utilisateur final. Avant l'exécution d'une règle, vérifiez que les services externes sont configurés avec suffisamment de débit et d'unités de capacité.

Diagnostic des problèmes de Thing Shadows

Diagnostic de Thing Shadows

Problème	Consignes pour la résolution des problèmes
Un document thing shadow est rejeté avec l'erreur « Document JSON non valide ».	Si vous ne connaissez pas JSON, modifiez les exemples fournis dans ce manuel pour les adapter à votre utilisation. Pour plus d'informations, consultez Syntaxe de document Thing Shadow .
J'ai envoyé un code JSON correct, mais aucune ou seulement quelques parties sont stockées dans le document Thing Shadow.	Vérifiez que vous avez suivi les consignes de formatage JSON. Seuls les champs JSON dans les sections souhaitées et déclarées seront stockés. Le contenu JSON (même s'il est formaté correctement) à l'extérieur de ces sections sera ignoré.
J'ai reçu un message d'erreur indiquant que le thing shadow dépasse la taille autorisée.	Le thing shadow prend en charge seulement 8 Ko de données. Essayez de raccourcir les

Problème	Consignes pour la résolution des problèmes
	noms de champs au sein de votre document JSON ou créez simplement des thing shadows supplémentaires. Un dispositif peut avoir un nombre illimité de thing shadows. La seule condition est que le nom de chose soit unique à votre compte.
Lorsque je reçois une thing shadow, il fait plus de 8 Ko. Comment est-ce possible ?	À la réception, le AWS IoT service ajoute des métadonnées au thing shadow. Le service inclut ces données dans sa réponse, mais elles ne comptent pas dans la limite de 8 Ko. Seules les données d'état souhaité et déclaré au sein du document d'état envoyé au thing shadow comptent pour le calcul de la limite.
Ma demande a été rejetée car la version était incorrecte. Que dois-je faire ?	Exécutez une opération GET pour effectuer une synchronisation avec la dernière version du document d'état. Lorsque vous utilisez MQTT, abonnez-vous à la rubrique/update/accepted pour recevoir des notifications concernant des changements d'état et recevoir la dernière version du document JSON.
L'horodatage est décalé de quelques secondes.	L'horodatage des champs individuels et de l'ensemble du document JSON est mis à jour lorsque le document est reçu par le service AWS IoT ou lorsque le document d'état est publié dans le message /update/accepted and le message /update/delta. Les messages peuvent être retardés sur le réseau, ce qui peut entraîner un décalage de l'horodatage de quelques secondes.
Mon dispositif peut publier et s'abonner aux rubriques thing shadow correspondantes, mais lorsque je tente de mettre à jour le document thing shadow via l'API HTTP REST, un message HTTP 403 s'ouvre.	Vérifiez que vous avez créé des stratégies dans IAM pour autoriser l'accès à ces rubriques et pour l'action (UPDATE/GET/DELETE) correspondant aux informations d'identification que vous utilisez. Les stratégies et les stratégies de certificat IAM sont indépendantes.
Autres problèmes.	Le service Thing Shadows enregistre des erreurs dans CloudWatch Logs. Pour identifier les problèmes d'appareils et de configuration, activez CloudWatch Logs et consultez les journaux d'informations de débogage.

Limites AWS IoT

Le tableau suivant décrit les limites de AWS IoT.

Limites de l'agent de messages

Taille de l'ID du client	128 octets de caractères codés UTF-8.
Inactivité de la connexion (keep-alive)	<p>Par défaut, une connexion de client MQTT est déconnectée après 30 minutes d'inactivité. Lorsque le client envoie un message PUBLISH, SUBSCRIBE, PING ou PUBACK, l'horloge d'inactivité est réinitialisée.</p> <p>Un client peut demander un intervalle keep-alive plus court en spécifiant une valeur keep-alive entre 5 et 1 200 secondes dans le message MQTT CONNECT envoyé au serveur. Si une valeur keep-alive est spécifiée, le serveur déconnectera le client s'il ne reçoit pas de message PUBLISH, SUBSCRIBE, PINGREQ ou PUBACK dans un délai correspondant à 1,5 fois l'intervalle demandé. L'horloge keep-alive démarre une fois que l'expéditeur a envoyé un CONNACK.</p> <p>Si un client envoie une valeur keep-alive de zéro, le comportement keep-alive par défaut reste en place.</p> <p>Si un client demande une valeur keep-alive inférieure à 5 secondes, le serveur traite le client comme s'il avait demandé un intervalle keep-alive de 5 secondes.</p> <p>L'horloge keep-alive commence immédiatement, lorsque le serveur retourne un CONNACK au client. Un bref délai peut survenir entre l'envoi d'un message CONNECT par le client et le début du comportement keep-alive.</p>

Nombre maximal de barres oblique dans la rubrique et le filtre de rubriques	Une rubrique fournie lors de la publication d'un message ou un filtre de rubriques fourni lors de l'abonnement ne peut pas avoir plus de huit barres obliques (/).
Nombre maximal de messages entrants non reconnus	L'agent de messages autorise 100 messages non reconnus en cours par client. (Cette limite est appliquée sur tous les messages qui requièrent ACK.) Lorsque cette limite est atteinte, aucun nouveau message n'est accepté provenant de ce client jusqu'à ce qu'un ACK soit renvoyé par le serveur.
Nombre maximal de messages sortants non reconnus	L'agent de messages autorise seulement 100 messages non reconnus en cours par client. (Cette limite est appliquée sur tous les messages qui requièrent ACK.) Lorsque cette limite est atteinte, aucun nouveau message n'est envoyé au client jusqu'à ce que le client reconnaît les messages en cours.
Intervalle maximal de nouvelle tentative de remise de messages avec une qualité de service 1	Si un client connecté n'est pas en mesure de recevoir un ACK sur un message de qualité de service 1 pendant une heure, l'agent de messages supprime le message. Le client peut ne pas être en mesure de recevoir le message s'il comporte 100 messages en cours, il devient engorgé en raison de charges utiles importantes, ou d'autres erreurs.
Nombre maximal d'abonnements par appel d'abonnement	Un seul appel d'abonnement est limité à demander un maximum de huit abonnements.
Taille de message	Les charges utiles pour tous les messages de publication sont limitées à 128 Ko. Le service AWS IoT rejette les messages dont la taille est supérieure à cette valeur.
Préfixe d'ID de client limité	« \$» est réservé aux ID de client générés en interne.
Préfixe de rubrique limitée	Les rubriques commençant par « \$» sont considérées comme réservées et ne sont pas prises en charge pour la publication et l'abonnement, sauf si elles sont utilisées avec le service Thing Shadows.
Abonnements par session	L'agent de messages limite chaque session du client à s'abonner à un maximum de 50 abonnements. Une demande d'abonnement qui pousse le nombre total d'abonnements au-delà de 50 entraînera une déconnexion.
Taille du nom de l'objet	128 octets de caractères codés UTF-8. Cette limite s'applique aux services Thing Registry et Thing Shadow.

Débit par connexion	AWS IoT limite le débit du trafic entrant et sortant taux sur chaque connexion au client à 512 Ko/s. Les données envoyées ou reçues à un débit plus élevé sont restreintes à ce débit.
Taille de la rubrique	La rubrique transmise à l'agent de messages lors de la publication d'un message ne peut pas dépasser 256 octets de caractères codés UTF-8.
Durée de la connexion WebSocket	<p>Les connexions WebSocket sont limitées à 24 heures. Si la limite est dépassée, la connexion WebSocket sera automatiquement fermée lors d'une tentative d'envoi d'un message par le client ou le serveur. Si vous avez besoin de conserver une connexion WebSocket active pendant plus de 24 heures, il vous suffit fermer et rouvrir la connexion WebSocket du côté client avant expiration du délai.</p> <p>AWS IoT prend en charge les valeurs keep-alive spécifiées dans les messages MQTT CONNECT. Lorsqu'un client spécifie une valeur keep-alive, le client indique au serveur qu'il doit déconnecter le client et transmettre les messages last-will associés à la session MQTT si le serveur n'a pas reçu de message (PUBLISH, SUBSCRIBE, PUBACK, PINGREQ) au cours de la période keep-alive multipliée par 1,5. AWS IoT prend en charge les valeurs keep-alive entre 5 secondes et 20 minutes. Si un client ne demande aucune valeur keep-alive (s'il définit le champ à la valeur 0 dans le message MQTT CONNECT), le serveur définit la valeur keep-alive à 20 minutes, ce qui correspond à la durée maximale d'inactivité pris en charge par AWS IoT en 30 minutes. La plupart des clients MQTT (y compris les clients SDK AWS) prennent en charge les valeurs keep-alive en envoyant une PINGREQ si la période keep-alive expire sans transmission d'autre message par le client.</p>

Limites Device Shadow

Profondeur maximale des documents JSON d'état d'appareil	<p>Le nombre maximal de niveaux dans la section « souhaité » ou « déclaré » du document JSON d'état d'appareil est 5. Exemples :</p> <pre>"desired": { "one": { "two": { "three": { "four": { "five": {} } } } } }</pre>
Nombre maximal de messages en cours, non reconnus	Le service Thing Shadows prend en charge jusqu'à 10 messages en cours non reconnus. Lorsque cette limite est atteinte, toutes les

	nouvelles demandes shadow seront rejetées avec un code d'erreur 429.
Nombre maximal d'objets JSON par compte AWS	Il n'existe aucune limite au nombre d'objets JSON par compte AWS.
Taille maximale d'un document d'état JSON	La taille maximale d'un document JSON d'état est de 8 Ko.
Taille maximale d'un nom d'objet	La taille maximale d'un nom d'objet est de 128 octets de caractères codés UTF-8.
Durée de vie d'un thing shadow	Un thing shadow est supprimé par AWS IoT s'il n'a pas été mis à jour ou récupéré depuis plus d'un an.

Limites de sécurité et d'identité

Nombre maximal de stratégies pouvant être attachées à un certificat.	10
Nombre maximal de versions de stratégies nommées.	5
Taille maximale du document de stratégie.	2 048 caractères (espaces non compris).
Nombre maximal de certificats d'appareils qui peuvent être enregistrés par seconde.	15

Limites de limitation

Le tableau suivant répertorie les limites de limitation pour l'API AWS IoT :

API	Transactions par seconde
AcceptCertificateTransfer	10
AttachThingPrincipal	15
CancelCertificateTransfer	10
CreateCertificateFromCsr	15
CreatePolicy	10
CreatePolicyVersion	10
CreateThing	15
DeleteCertificate	10
DeleteCACertificate	10
DeletePolicy	10
DeletePolicyVersion	10

API	Transactions par seconde
DeleteThing	10
DescribeCertificate	10
DescribeCACertificate	10
DescribeThing	10
DetachThingPrincipal	10
DetachPrincipalPolicy	15
DeleteRegistrationCode	10
GetPolicy	10
GetPolicyVersion	15
GetRegistrationCode	10
ListCertificates	10
ListCertificatesByCA	10
ListPolicies	10
ListPolicyVersions	10
ListPrincipalPolicies	15
ListPrincipalThings	10
ListThings	10
ListThingPrincipals	10
RegisterCertificate	10
RegisterCACertificate	10
RejectCertificateTransfer	10
SetDefaultPolicyVersion	10
TransferCertificate	10
UpdateCertificate	10
UpdateCACertificate	10
UpdateThing	10

Limites de moteur de règles AWS IoT

Nombre maximal de règles par compte AWS.	1 000
Nombre d'actions par règle.	Un maximum de 10 actions peut être défini par règle.

Taille de la règle

Jusqu'à 256 Ko de caractères codés UTF-8
(espaces compris).