

使用 **AWS Lambda** 和 **Amazon DynamoDB** 实现 **Serverless** 的 **AWS IoT** 后台

Phodal Huang

October 24, 2017

目录

| | |
|-------------------------|----|
| 步骤 1: 先决条件 | 3 |
| 步骤 2: 构建后台 | 4 |
| 步骤 3: 激活和注册逻辑 | 10 |
| 步骤 4: 清理 | 14 |
| 步骤 5: 结论 | 14 |

玩点什么: <https://www.wandianshenme.com>

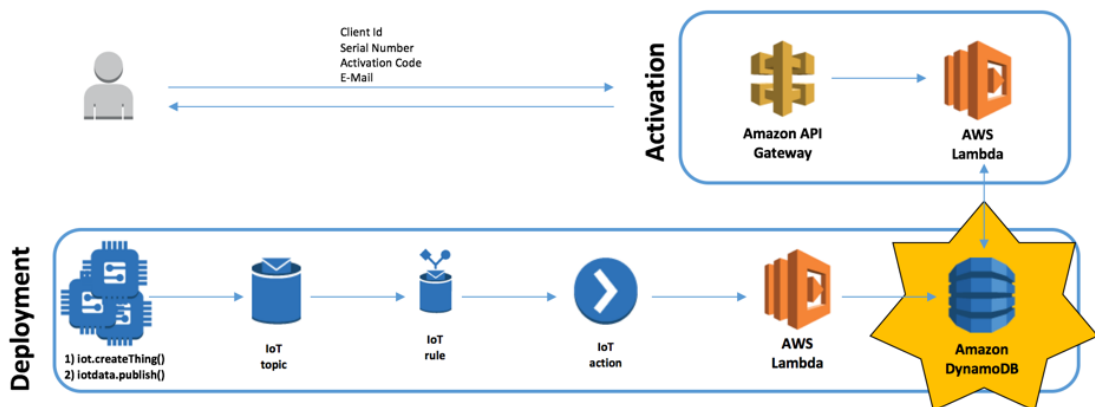
原文链接:<https://www.wandianshenme.com/play/aws-lambda-iot-rules-dynamodb-build-serverless>

您的 IoT 设备是否可以扩展到数百台或数千台设备? 检索多个设备的详细信息有什么挑战吗? AWS IoT 提供了一个连接这些设备的平台, 并为您的物联网工作负载构建可扩展的解决方案。

开箱即用的 AWS IoT 控制台, 为您提供自己的可搜索设备注册表, 可以访问设备状态和有关设备 shadows 的信息。您可以使用 AWS Lambda 和 Amazon DynamoDB 来增强和定制服务, 以使用可自定义的设备数据库构建 Serverless 物联网后端, 可用于存储有关设备的有用信息, 如果需要的话, 还可以帮助跟踪哪些设备使用激活码进行激活。

您可以使用 DynamoDB 来扩展 AWS IoT 内部设备注册表, 以帮助管理设备队列, 以及存储有关每个设备的特定附加数据。Lambda 提供了 AWS IoT 和 DynamoDB 之间的链接, 允许您添加、更新和查询新的设备数据库后端。

在本文中, 您将学习如何使用 AWS IoT rules 来使用 Lambda 来触发特定的设备注册逻辑, 以将数据存储在 DynamoDB 表。然后, 使用第二个 Lambda 函数在数据库中搜索特定设备序列号、随机生成的激活码以用于激活设备, 并在同一个表中注册设备所有者的电子邮件。完成后, 您将拥有完整功能的 Serverless IoT 后台, 可让您专注于自己的 IoT 解决方案和逻辑, 而不是管理基础设施。



步骤 1: 先决条件

在创建和部署此框架之前, 必须具有以下内容:

- 一个 AWS 帐户
- 具有创建 AWS 资源 (AWS IoT 事件和规则, Lambda 函数, DynamoDB 表, IAM 策略和角色等) 的 IAM 用户
- JavaScript 和安装在本地的基于 JavaScript 的 AWS SDK 用于测试部署

步骤 2: 构建后台

在这篇文章中,我假设你对所涉及的服务有一些基本的了解。如果没有,您可以查看相关的文档:

- [Creating Lambda functions](#)
- [Creating DynamoDB tables](#)
- [Overview of AWS IoT](#)
- [Configuring AWS IoT rules to trigger newly-created Lambda functions](#)

对于这个用例,假设你有一个名为“myThing”的设备。这些设备可以是任何东西:智能灯泡,智能中枢(**smart hub**),互联网连接的机器人,音乐播放器,智能恒温器,或具有特定传感器的任何可以使用 **AWS IoT** 进行管理的设备。

创建 **myThing** 设备时,您需要在数据库中提供一些特定信息,即:

- **Client ID**
- **Serial number**
- **Activation code**
- **Activation status**
- **Device name**
- **Device type**
- **Owner email**
- **AWS IoT endpoint**

以下是一个包的 (**payload**) 示例,其中包含要发送到特定 **MQTT** 主题的单个 **myThing** 设备的详细信息,这将触发 **IoT** 规则。数据是 **AWS IoT** 可以理解的格式,很好的旧 **JSON**。例如:

```
1 {
2   "clientId": "ID-91B2F06B3F05",
3   "serialNumber": "SN-D7F3C8947867",
4   "activationCode": "AC-9BE75CD0F1543D44C9AB",
5   "activated": "false",
6   "device": "myThing1",
7   "type": "MySmartIoTDevice",
8   "email": "not@registered.yet",
9   "endpoint": "<endpoint prefix>.iot.<region>.amazonaws.com"
10 }
```

然后 **rule** (规则) 调用您现在创建的第一个 **Lambda** 函数。打开 [Lambda 控制台](#), 选择创建一个 **Lambda** 函数, 然后按照步骤。以下是代码:

```
1 console.log('Loading function');
2 var AWS = require('aws-sdk');
3 var dynamo = new AWS.DynamoDB.DocumentClient();
4 var table = "iotCatalog";
5
6 exports.handler = function(event, context) {
7     //console.log('Received event:', JSON.stringify(event, null, 2));
8     var params = {
9         TableName:table,
10        Item:{
11            "serialNumber": event.serialNumber,
12            "clientId": event.clientId,
13            "device": event.device,
14            "endpoint": event.endpoint,
15            "type": event.type,
16            "certificateId": event.certificateId,
17            "activationCode": event.activationCode,
18            "activated": event.activated,
19            "email": event.email
20        }
21    };
22
23    console.log("Adding a new IoT device...");
24    dynamo.put(params, function(err, data) {
25        if (err) {
26            console.error("Unable to add device. Error JSON:",
27                JSON.stringify(err, null, 2));
28            context.fail();
29        } else {
30            console.log("Added device:", JSON.stringify(data, null, 2));
31            context.succeed();
32        }
33    });
34 }
```

该函数将根据事件（如前面提供的 JSON 数据），将一个项目添加到名为 `iotCatalog` 的 DynamoDB 数据库。您现在需要创建数据库，并确保 Lambda 函数具有将项目添加到 DynamoDB 表的权限，可以通过使用适当的执行角色（[execution role](#)）进行配置。

打开 [DynamoDB 控制台](#)，选择创建表并按照步骤。对于此表，请使用以下详细信息：

Create DynamoDB table

Tutorial



DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

| | | |
|--------------|--|--------|
| Table name* | <input type="text" value="iotCatalog"/> | |
| Primary key* | Partition key | |
| | <input type="text" value="serialNumber"/> | String |
| | <input checked="" type="checkbox"/> Add sort key | |
| | <input type="text" value="clientId"/> | String |

Table settings

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

Use default settings

序列号用于唯一标识您的设备；例如，如果它是具有连接到其的不同客户端设备的智能中心，则使用客户端 ID 作为排序键。

后台已经准备好了！你只需要使新的资源来一起工作；为此，您可以配置一个 IoT 规则。

在 AWS IoT 控制台上，选择创建资源并创建规则，并使用以下设置将规则指向新创建的 Lambda 函数，也称为 `iotCatalog`。

Rule query statement

SQL version

Attribute

Topic filter

Condition

Select one or more actions to happen when the above rule query is matched by messages arriving on the topic, like storing them in a database, invoking cloud function, or sending them to a third party.

Choose an action

This will invoke a Lambda function with the message set in the payload of the message as the input to the function for you.

***Function name**

创建规则后，AWS IoT 会在后台添加权限，以便每当将消息发布到名为 **registration** 的 MQTT 主题时触发 **Lambda** 功能。您可以使用以下 **Node.js** 部署代码进行测试：

```

1 var AWS = require('aws-sdk');
2 AWS.config.region = 'ap-northeast-1';
3
4 var crypto = require('crypto');
5 var endpoint = "<endpoint prefix>.iot.<region>.amazonaws.com";
6 var iot = new AWS.Iot();
7 var iotdata = new AWS.IotData({endpoint: endpoint});
8 var topic = "registration";
9 var type = "MySmartIoTDevice"
10
11 //Create 50 AWS IoT Things
12 for(var i = 1; i < 51; i++) {
13   var serialNumber =
14     "SN-"+crypto.randomBytes(Math.ceil(12/2)).toString('hex').slice(0,15).toUpperCase();
15   var clientId =
16     "ID-"+crypto.randomBytes(Math.ceil(12/2)).toString('hex').slice(0,12).toUpperCase();

```

```
15 var activationCode =
    "AC-"+crypto.randomBytes(Math.ceil(20/2)).toString('hex').slice(0,20).toUpperCase();
16 var thing = "myThing"+i.toString(); var thingParams = {
17   thingName: thing
18 };
19
20 iot.createThing(thingParams).on('success', function(response) {
21   //Thing Created!
22 }).on('error', function(response) {
23   console.log(response);
24 }).send();
25
26 //Publish JSON to Registration Topic
27
28 var registrationData = '{\n  "serialNumber": \''+serialNumber+'\n',\n
    "clientId": \''+clientId+'\n',\n  "device": \''+thing+'\n',\n
    "endpoint": \''+endpoint+'\n',\n  "type": \''+type+'\n',\n
    "activationCode": \''+activationCode+'\n',\n  "activated":
    "false",\n  "email": "not@registered.yet" \n}';
29
30 var registrationParams = {
31   topic: topic,
32   payload: registrationData,
33   qos: 0
34 };
35
36 iotdata.publish(registrationParams, function(err, data) {
37   if (err) console.log(err, err.stack); // an error occurred
38   // else Published Successfully!
39 });
40 setTimeout(function() {}, 50);
41 }
42
43 //Checking all devices were created
44
45 iot.listThings().on('success', function(response) {
```

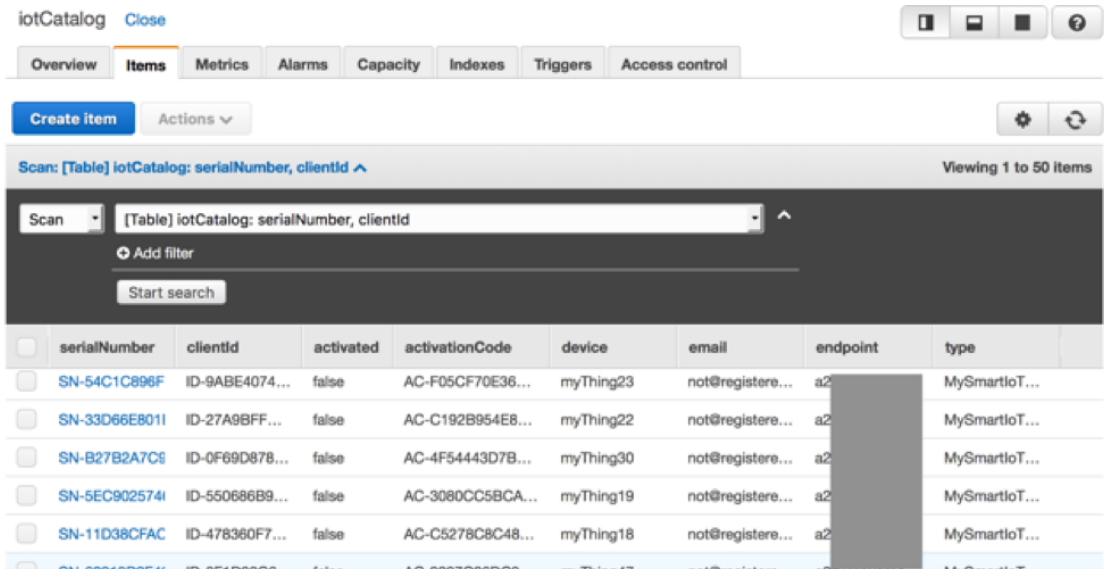


```
46 var things = response.data.things;
47 var myThings = []; for(var i = 0; i < things.length; i++) {
48     if (things[i].thingName.includes("myThing")){
49         myThings[i]=things[i].thingName;
50     }
51 }
52
53 if (myThings.length = 50){
54     console.log("myThing1 to 50 created and registered!");
55 }
56 }).on('error', function(response) {
57     console.log(response);
58 }).send();
59
60 console.log("Registration data on the way to Lambda and DynamoDB");
```

以上代码用于在 AWS IoT 中创建了 50 个 IoT 事件，并为每个设备生成随机客户端 ID，序列号和激活码。然后，它将设备数据作为 JSON 包发布到 IoT 主题，这反过来又触发了 Lambda 函数：

```
> node iot2lambda.js
Registration data on the way to Lambda and DynamoDB
myThing1 to 50 created and registered!
```

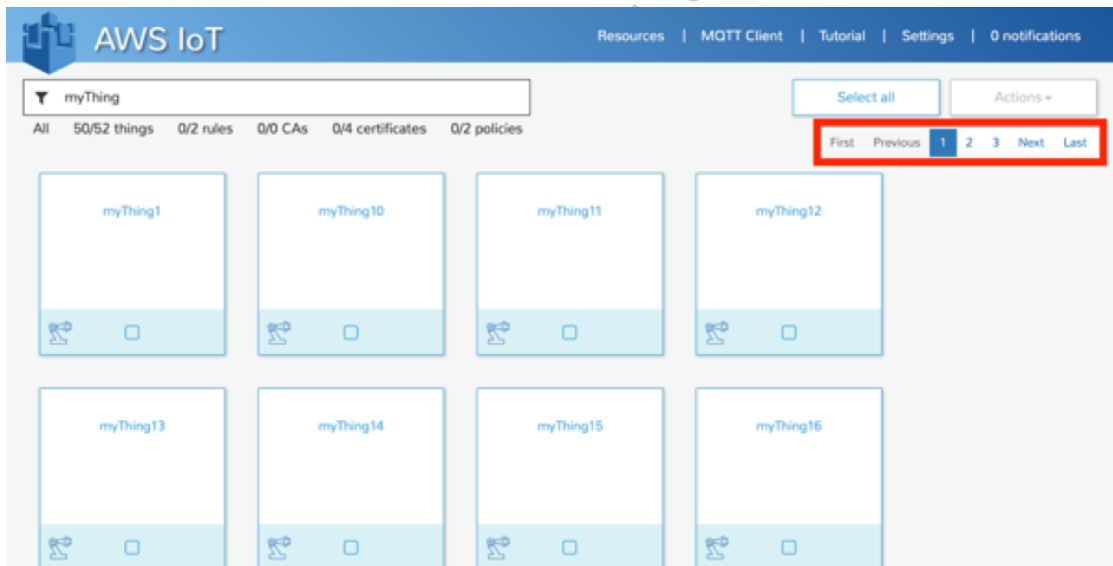
这里是！该功能由您的 IoT 规则成功触发，并创建了您所需的所有自定义信息的 IoT 设备数据库。您可以查询数据库，以便查找您的内容以及与其相关的任何其他详细信息。



The screenshot shows the AWS IoT Catalog console. At the top, there are tabs for Overview, Items, Metrics, Alarms, Capacity, Indexes, Triggers, and Access control. Below the tabs, there is a 'Create item' button and an 'Actions' dropdown. The main content area displays a table of IoT things with the following columns: serialNumber, clientId, activated, activationCode, device, email, endpoint, and type. The table contains several rows of data, including items like 'myThing23', 'myThing22', 'myThing30', 'myThing19', and 'myThing18'. A search bar is visible at the top of the table, and a 'Start search' button is located below it.

| serialNumber | clientId | activated | activationCode | device | email | endpoint | type |
|---------------|----------------|-----------|------------------|-----------|-----------------|----------|---------------|
| SN-54C1C896F | ID-9ABE4074... | false | AC-F05CF70E36... | myThing23 | not@registre... | a2 | MySmartioT... |
| SN-33D66E801I | ID-27A9BFF... | false | AC-C192B954E8... | myThing22 | not@registre... | a2 | MySmartioT... |
| SN-B27B2A7C6 | ID-0F69D878... | false | AC-4F54443D7B... | myThing30 | not@registre... | a2 | MySmartioT... |
| SN-5EC902574I | ID-550686B9... | false | AC-3080CC5BCA... | myThing19 | not@registre... | a2 | MySmartioT... |
| SN-11D38CFAC | ID-478360F7... | false | AC-C5278C8C48... | myThing18 | not@registre... | a2 | MySmartioT... |

在 AWS IoT 控制台中，新创建的 things 也可以在 thing 注册表中使用。



现在，您可以创建证书、策略，将其附加到每个“myThing” AWS IoT Thing，然后在配置物理设备时安装每个证书。

步骤 3：激活和注册逻辑

但是，你还没有完成... 如果要使用预先生成的激活码激活现场设备，以及注册激活设备的人的电子邮件详细信息怎么办？

您需要一个第二个 Lambda 函数，与第一个函数（Basic with DynamoDB）具有相同的执行角色。以下是代码：

```
1 console.log('Loading function');
```

```
2
3 var AWS = require('aws-sdk');var dynamo = new AWS.DynamoDB.DocumentClient();
4 var table = "iotCatalog";
5
6 exports.handler = function(event, context) {
7     //console.log('Received event:', JSON.stringify(event, null, 2));
8
9     var params = {
10         TableName:table,
11         Key:{
12             "serialNumber": event.serialNumber,
13             "clientId": event.clientId,
14         }
15     };
16
17     console.log("Gettings IoT device details...");
18     dynamo.get(params, function(err, data) {
19         if (err) {
20             console.error("Unable to get device details. Error JSON:",
21                 JSON.stringify(err, null, 2));
22             context.fail();
23         } else {
24             console.log("Device data:", JSON.stringify(data, null, 2));
25             console.log(data.Item.activationCode);
26             if (data.Item.activationCode == event.activationCode){
27                 console.log("Valid Activation Code! Proceed to register owner
28                     e-mail and update activation status");
29                 var params = {
30                     TableName:table,
31                     Key:{
32                         "serialNumber": event.serialNumber,
33                         "clientId": event.clientId,
34                     },
35                     UpdateExpression: "set email = :val1, activated = :val2",
36                     ExpressionAttributeValues:{
37                         ":val1": event.email,
```

```
36         ":val2": "true"
37     },
38     ReturnValues: "UPDATED\_NEW"
39 };
40 dynamo.update(params, function(err, data) {
41     if (err) {
42         console.error("Unable to update item. Error JSON:",
43             JSON.stringify(err, null, 2));
44         context.fail();
45     } else {
46         console.log("Device now active!", JSON.stringify(data,
47             null, 2));
48         context.succeed("Device now active! Your e-mail is now
49             registered as device owner, thank you for
50             activating your Smart IoT Device!");
51     }
52 });
53 }
```

该函数仅需要使用较早的数据的一小部分:

```
1 {
2     "clientId": "ID-91B2F06B3F05",
3     "serialNumber": "SN-D7F3C8947867",
4     "activationCode": "AC-9BE75CD0F1543D44C9AB",
5     "email": "verified@registered.iot"
6 }
```

Lambda 使用散列和范围键 (**serialNumber** 和 **clientId**) 查询数据库, 并将数据库当前预生成的激活代码与设备所有者提供的代码、电子邮件地址进行比较。如果激活代码与数据库中的激活码匹配, 则相应地在 **DynamoDB** 中更新激活状态和电子邮件详细信息。如果没有, 用户将收到一条错误消息, 并指出激活码无效。

您可以使用 **Amazon API Gateway** 将其转换为 **API**。为了做到这一点, 转到 **Lambda**

函数并添加 API 节点，如下所示：

Add API endpoint ✕

Warning: Your API endpoint will be publicly available and can be invoked by all users. ✕

Configure your Lambda function to be invoked on requests made to an API endpoint.

API endpoint type i

API name i

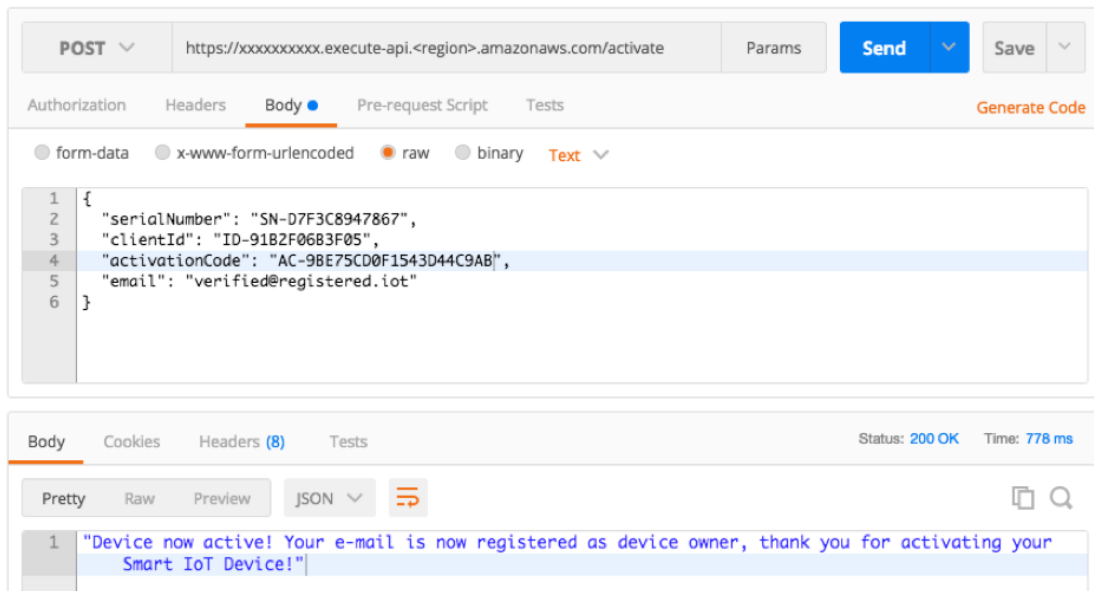
Resource name i

Method i

Deployment stage i

Security i

现在，使用 Postman 这样的工具来测试对新创建的 API 节点的访问。



POST Params

Authorization Headers **Body** Pre-request Script Tests Generate Code

form-data x-www-form-urlencoded raw binary

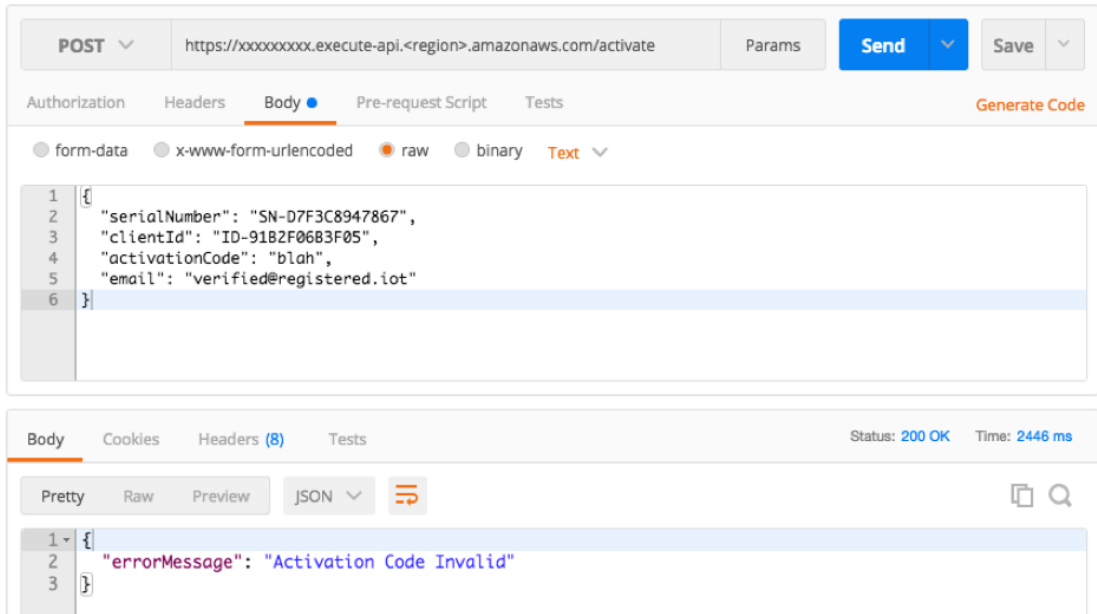
```
1 {
2   "serialNumber": "SN-D7F3C8947867",
3   "clientId": "ID-91B2F06B3F05",
4   "activationCode": "AC-9BE75CD0F1543D44C9AB",
5   "email": "verified@registered.iot"
6 }
```

Body Cookies Headers (8) Tests Status: 200 OK Time: 778 ms

Pretty Raw Preview

```
1 "Device now active! Your e-mail is now registered as device owner, thank you for activating your Smart IoT Device!"
```

如果提供了无效的激活码，则请求者会相应地获取错误消息。



返回数据库，您可以确认记录已经按需要更新了。



步骤 4: 清理

完成本教程后，删除所有新创建的资源（IoT，Lambda 函数和 DynamoDB 表）。或者，您可以保留 Lambda 功能代码以备将来参考，因为除非调用函数，否则不会产生费用。

步骤 5: 结论

如你所见，通过利用 AWS IoT 规则引擎的强大功能，您可以利用与 AWS Lambda 的无缝集成，以创建由 Amazon DynamoDB 提供的灵活且可扩展的 IoT 后端，可用于管理日益增长的物联网“舰队”。

您还可以配置激活 API，以利用新创建的后端。并激活设备，以及从设备所有者注册电子邮件联系人详细信息；该信息可用于与您的用户联系，了解关于新产品或新版本的 IoT 产品的营销活动或简报。

原文链接: <https://aws.amazon.com/cn/blogs/compute/implementing-a-serverless-aws-iot-backend-with-aws-lambda-and-amazon-dynamodb/>

原文链接: <https://www.wandianshenme.com/play/aws-lambda-iot-rules-dynamodb-build-serverles>

玩点什么: <https://www.wandianshenme.com>