*TI Designs：TIDC-01002*

# 低于 *1GHz* 嵌入式传感器到云工业物联网 *(IoT)* 网关参考设计



## 说明

TIDC-01002 设计展示了如何通过低于 1GHz 的远距离无线网络将传感器连接到云，适合工业环境，例如楼宇控制和资产跟踪。此 TI 设计搭载了 TI SimpleLink™CC3220 处理器和 SimpleLink 超低功耗 (ULP) 低于 1GHz CC13x0 和 CC13x2 器件。此参考设计预集成了 SimpleLink CC13xx 软件开发套件 (SDK) 和 SimpleLink CC3220 SDK（两者均为 TI SimpleLink MCU 平台的组成部分）组成部分中的 TI 15.4-Stack 软件，从而在 TI 低功耗有线和无线 MCU 中实现了一致的软件体验。

## 资源

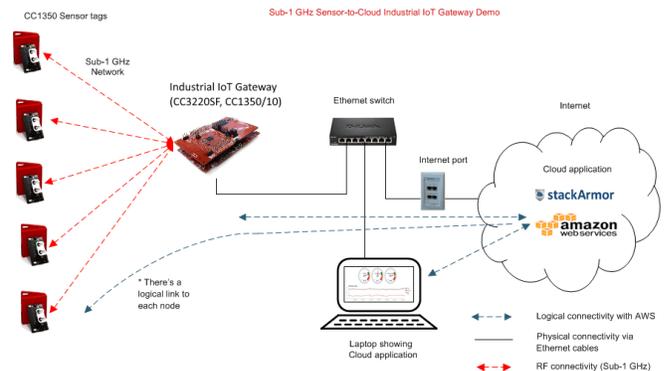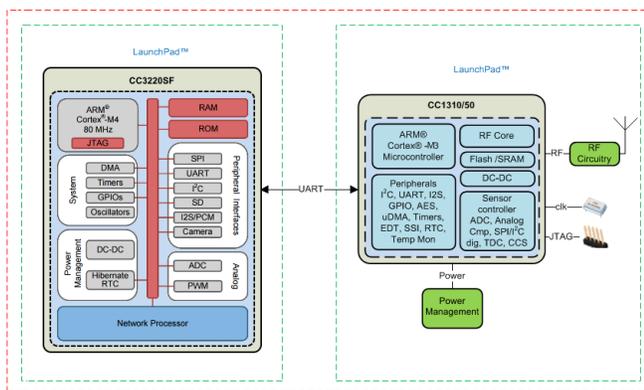| | |
|---|---|
| TIDC-01002 | 设计文件夹 |
| TIDC-01002 | 示例库页面 |
| CC1352 | 产品文件夹 |
| CC1310 | 产品文件夹 |
| CC1350 | 产品文件夹 |
| CC3220 | 产品文件夹 |



咨询我们的 E2E 专家

## 特性

- 大型网络到云连接可实现远距离应用，支持长达 1km 的视线
- 使用 TI 15.4-Stack，有助于设计人员实现符合 IEEE 802.15.4e/g 规范的系统
- 基于经 TI 测试的硬件设计，具有开箱即用的现成演示软件，可缩短上市时间
- 基于可移植操作系统接口 (POSIX) 进行实施，可在 TI 互联网连接的微控制器 (MCU) 之间实现轻松移植
- 支持星形网络
- ULP 传感器节点

## 应用

- 楼宇安全网关
- 门窗传感器网络
- HVAC 网关
- 资产管理和跟踪



该 TI 参考设计末尾的重要声明表述了授权使用、知识产权问题和其他重要的免责声明和信息。

# 1 System Description

The TIDC-01002 provides a reference for creating an industrial, Internet of things (IoT) gateway that is capable of connecting a network of wireless sensors to an enterprise cloud provider. In this reference design, a long-range, low-power wireless network, made up of Sub-1 GHz CC13x0 or CC13x2 devices (both are supported) that run the TI 15.4-Stack-based application, can be connected to multiple *cloud service* providers, such as IBM Watson IoT®, AWS IoT, and so on. An online dashboard is provided, which allows users visualize the real-time sensor data as well as send actuation commands from anywhere in the world using an Internet-connected device with a web browser.

This reference design provides a list of suggested hardware, schematics, and foundational software to quickly begin IoT product development. The design also provides the ability to visualize the data inside a local network without connecting to a *cloud service*. The software design is created to be flexible, to enable other *cloud service* providers of choice.

This reference design enables IoT in numerous applications, such as building security gateways, door and window sensor networks, asset management and tracking, and other IoT-enabled home and industrial automation applications.

The connection between the wireless sensor network and the cloud is made possible by the TI SimpleLink CC3220 device on the CC3220SF LaunchPad™ development platform. On one side, the CC3220 is connected to a Sub-1 GHz device acting as the central node in the wireless network, and on the other side, the device is connected to a cloud service such as IBM Watson IoT or AWS IoT using Wi-Fi®. These two connections allow the CC3220 device to act as a gateway to get the sensor messages from the Sub-1 GHz wireless network to the cloud and to get the actuation requests from the cloud dashboard sent back to the Sub-1 GHz wireless network.

Due to the long-range and low-power capabilities of the Sub-1 GHz sensors, this reference design may be useful for any application that would benefit from distributed sensing. This reference design provides an example that gives the ability to visualize or actuate tens or hundreds of sensors while only needing one gateway device, the SimpleLink CC3220, to be connected to the Internet.

## 2        System Overview

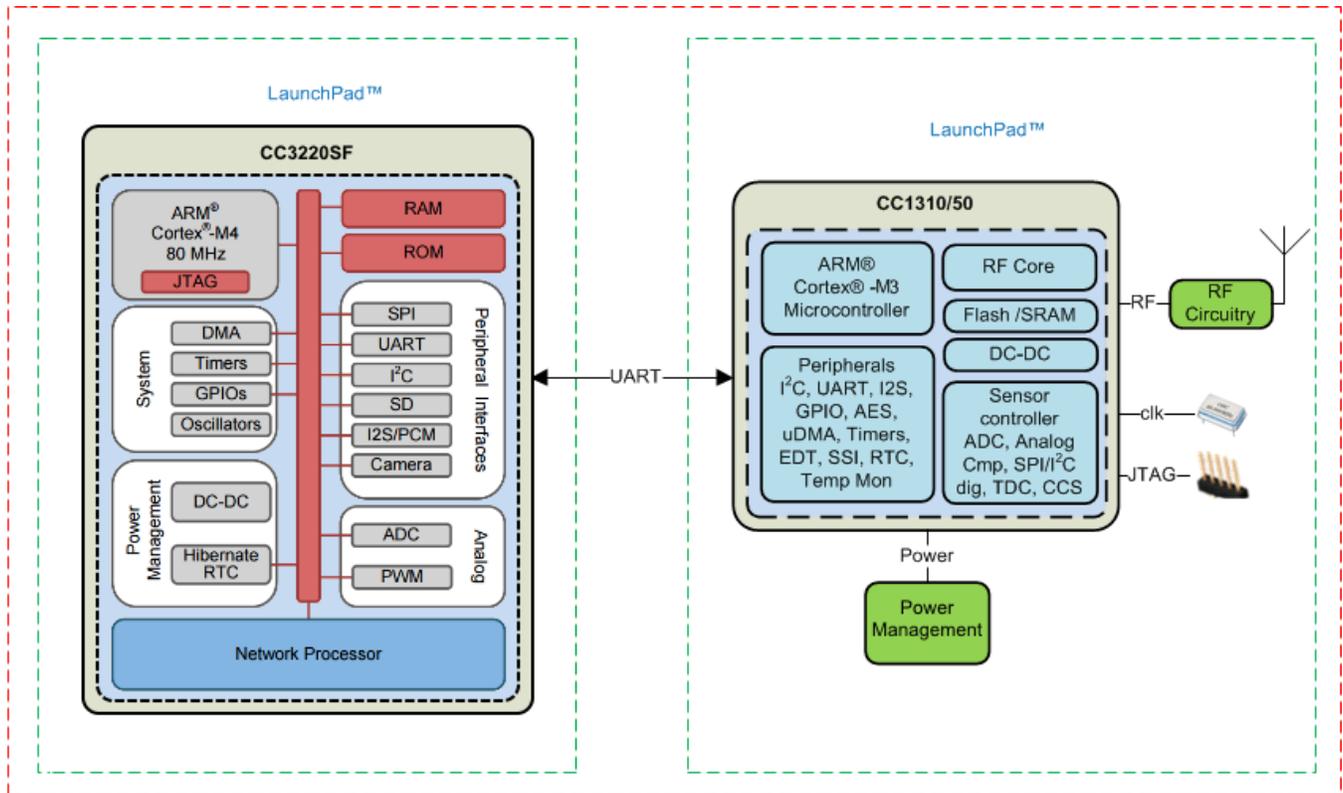### *2.1      Block Diagram*



图 **1. Block Diagram of IoT Gateway Reference Design**
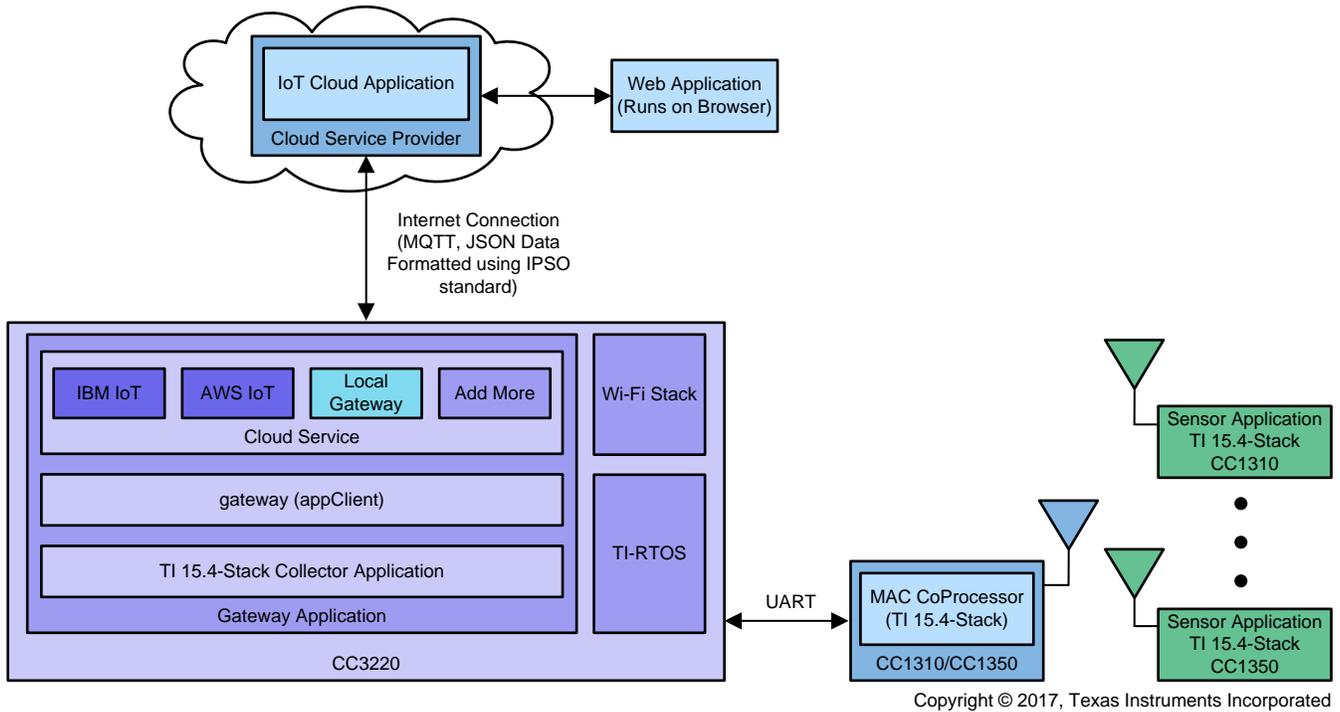
### 2.1.1 Software Block Diagram



图 **2. Software Block Diagram of TI 15.4-Stack Sensor-to-Cloud Reference Design**

The following is a high-level description of each module in the software block diagram:

- User interface application: This application presents the network information and device information and provides ability to control network behavior to the end user.

- IoT cloud application: This application runs on the cloud server, which communicates with the IoT gateway application. The interface of the *cloud service* task with the cloud server is described in 节 2.3.1.

- IoT gateway application: This application runs on the SimpleLink CC3220. The application interfaces on one side with the *cloud service* task to enable cloud connectivity and on the other side to the *collector* task to interface with the TI 15.4-Stack based network. The interface between the IoT gateway and the *cloud service* is described in 节 2.3.1.

  - *Cloud service* task: This task provides the *cloud service* provider specific functionality. Users can take the current interface, which is designed as an extensible framework, and quickly modify the interface to add their own functionality for their end product development.

  - *Gateway*: This application component interfaces with the *collector* task through a POSIX message queue interface to enable connection with the TI-15.4 Stack network.

- *TI 15.4-Stack collector* task: This task implements the functionality that starts the network, allows new devices to join the network, configures the joining devices on how often to report the sensor data, configures how often to poll for buffered messages in case of non-beacon and frequency-hopping mode of network operation for sleepy network devices, and tracks connected devices to determine if they are active or inactive on the network. This determination is achieved by the collector periodically sending tracking request messages and awaiting corresponding tracking response messages. The *collector* task also implements components that talk to the *gateway* module. The communication is

implemented through POSIX-based message queues.

- *MAC CoP* application: The MAC coprocessor application runs on the CC13x0 or CC13x2 LaunchPad, which provides a UART-based interface from TI 15.4-Stack to the IoT gateway application.
- CC13x0 or CC13x2 LaunchPad Sensor End Node: The sensor example application from TI 15.4-Stack and runs on the CC13x0 or CC13x2 LaunchPad.

## 2.2 *Highlighted Products*

This section highlights key hardware devices and software components used in the reference design.

### 2.2.1 SimpleLink™ CC13x0 and CC13x2

The CC13xx is a member of the SimpleLink family of cost-effective, ULP, 2.4-GHz and Sub-1 GHz RF devices. In addition to flexible low-power modes, very-low active RF and MCU current consumption provide excellent battery lifetime and allow long-range operation on small, coin-cell batteries and in energy-harvesting applications.

The CC13x2R devices combine a flexible, very low-power RF transceiver with a powerful 48-MHz Arm® Cortex®-M4F CPU in a platform supporting multiple physical layers and RF standards. A dedicated Radio Controller (Arm® Cortex®-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus ensuring ultra-low power and great flexibility. The low power consumption of the CC1352R device does not come at the expense of RF performance; the CC1352R device has excellent sensitivity and robustness (selectivity and blocking) performance.

The CC1350 is the first device in the CC13xx and CC26xx family of cost-effective, ULP wireless MCUs capable of handling both Sub-1 GHz and 2.4-GHz RF frequencies. The CC1350 device combines a flexible, very-low-power RF transceiver with a powerful, 48-MHz, Cortex®-M3 MCU in a platform supporting multiple physical layers and RF standards. A dedicated radio controller (Cortex-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus, ensuring ULP and flexibility to handle both Sub-1 GHz protocols and 2.4-GHz protocols (for example, *Bluetooth®* low energy). This enables the combination of a Sub-1 GHz communication stack that offers the best possible RF range together with a connection to a Bluetooth low energy smartphone that enables a great user experience through a phone application. The Sub-1 GHz-only devices in this family are the CC1310 and the CC1312.
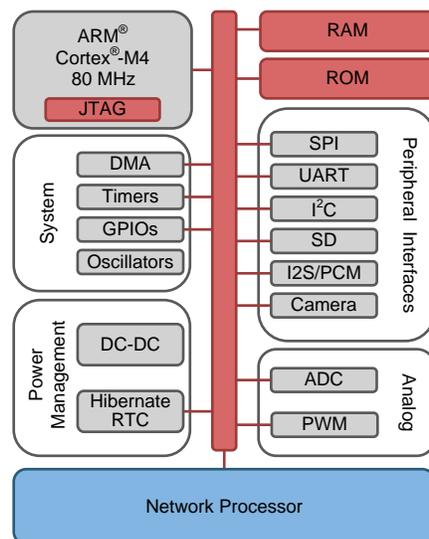
### 2.2.2 SimpleLink™ CC3220

The CC3220x device is part of the SimpleLink MCU platform, which consists of Wi-Fi, low energy, Sub-1 GHz and host MCUs, which all share a common, easy-to-use development environment with a single core SDK and rich tool set. A one-time integration of the SimpleLink platform enables the user to add any combination of the portfolio's devices into their design, which allows 100% code reuse when the design requirements change. For more information, visit *SimpleLink Solutions* overview.

Created for the IoT, the SimpleLink CC3220x device family from Texas Instruments is a single-chip solution that integrates two physically separated, on-chip MCUs. One of the MCUs is an application processor— an ARM® Cortex®-M4 with a user-dedicated 256KB of RAM and an optional 1MB of XIP flash. The other MCU is a network processor in charge of running all Wi-Fi and Internet logical layers. This ROM-based subsystem includes an 802.11b/g/n radio, baseband, and MAC with a powerful crypto engine for fast, secure Internet connections with 256-bit encryption.

The CC3220x wireless MCU family is part of the second generation of TI's Internet-on-a-chip™ family. This generation introduces new features and capabilities that further simplify the connectivity of things to the Internet. The new capabilities including the following:

- IPv6

- Enhanced Wi-Fi provisioning

- Enhanced power consumption

- Enhanced file system security (supported only by the CC3220S and CC3220SF devices)

- Wi-Fi AP connection with up to four stations

- More concurrently opened BSD sockets; up to 16 BSD sockets, of which six are secure

- HTTPS support

- RESTful API support

- Asymmetric keys crypto library

The CC3220x wireless MCU family supports the following modes: station, AP, and Wi-Fi Direct®. The device also supports both WPA2-Personal and WPA2-Enterprise security modes. This subsystem includes embedded TCP/IP and TLS/SSL stacks, HTTP server, and multiple Internet protocols. The device supports a variety of Wi-Fi provisioning methods including HTTP based on AP mode, SmartConfig™ technology, and WPS2.0.



Copyright © 2017, Texas Instruments Incorporated

图 **3. CC3220 Block Diagram**

### 2.2.3 TI 15.4-Stack

TI 15.4-Stack is an IEEE802.15.4e/g-based software stack part of the SimpleLink CC13x0 SDK supporting a star network topology for Sub-1 GHz applications. TI 15.4-Stack software runs on the SimpleLink Sub-1 GHz CC13x0 or CC13x2 wireless MCU from TI. The TI 15-4 Stack offers several key benefits, such as longer range in FCC band and better protection against in-band interference by implementing frequency hopping. The SDK also offers customers an accelerated time to market by providing a complete end-to-end, node-to-gateway reference design. TI 15.4-Stack is supported on the industry's lowest-power SimpleLink Sub-1 GHz wireless MCU platform.

This release is available royalty-free to customers using TI's CC13x0 or CC13x2 wireless MCU and also runs on TI's SimpleLink Sub-1 GHz CC13x0 or CC13x2 wireless MCU LaunchPad development kit. This release is available royalty-free to customers using TI's CC13x0 or CC13x2 wireless MCU and also runs on the SimpleLink Sub-1 GHz CC13x0 or CC13x2 wireless MCU LaunchPad development kit from TI.

Features:

- IEEE 802.15.4e/g standards-based software stack

- Frequency hopping

- Medium access with CSMA/CA

- Built-in acknowledgment and retry

- Network and device management (joining, commissioning, service discovery)

- Security feature through AES 128-bit encryption and integrity check

- Supported on SimpleLink Sub-1 GHz CC1310 wireless MCU

- Star topology: point-to-point, one-to-many, and data concentrator

- Synchronous (beacon) and asynchronous (non-beacon) modes

- Designed for 915-MHz FCC, 863-MHz ETSI, and 433-MHz China bands

- SimpleLink long range mode for all supported frequency bands

- Support for SimpleLink CC1190

- Bluetooth low energy beacon advertisement support

- Sensor-to-web example application

- Easy application development guided through sample applications showcasing the stack configuration and APIs

- Coprocessor mode for adding connectivity to any MCU or MPU with Linux® host middleware and console application

For more details and to get the TI 15.4-Stack software, download the SimpleLink CC13x0 SDK, which includes the TI 15.4-Stack.

### 2.2.4 SimpleLink™ Wi-Fi CC3220 SDK

The SimpleLink Wi-Fi CC3220 SDK contains drivers for the CC3220 programmable MCU, over 30 sample applications, and related documentation. The SDK also contains the flash programmer, a command line tool for flashing software, configuring network and software parameters (SSID, access point channel, network profile, and so on), system files, and user files (certificates, web pages, and so on). This SDK can be used with TI's SimpleLink Wi-Fi CC3220 LaunchPad development kits.

Features:

- Internet-on-a-chip sample applications:
  - Email from SimpleLink Wi-Fi
  - Information center: Get time and weather from the Internet
  - https server: Host a secure web page on SimpleLink Wi-Fi
  - XMPP: IM chat client
  - Serial interface
- Wi-Fi sample applications:

- – Easy Wi-Fi configuration
- – Station, AP modes
- – TCP/UDP
- – Security—Enterprise and personal, TLS/SSL
- – Power management—Deep sleep, hibernate
- MCU peripheral sample applications:
  - – Including parallel camera, I2S audio, ADC, I²C, PWMs, JTAG Flashing, and more

## 2.3 System Design Theory

### 2.3.1 TI IoT Gateway-to-Cloud Service Interface

The purpose of this section is to provide a description of the message types and expected data flows that will be shared between the TI IoT gateway and an IoT cloud server. The interface is designed to be flexible to support multiple cloud vendors. For this purpose, the Sub-1 GHz wireless network and node information will be exchanged between the gateway and the cloud using the long-established JavaScript object notation (JSON) format. Additionally, IPSO alliance smart object definitions will be used to define sensors (and their data) that are connected to each node in the wireless networks.

#### 2.3.1.1 Message Types

To fully specify the Sub-1 GHz wireless network information, as well as the Sub-1 GHz sensors and their data, two distinct message types have been defined for the IoT gateway to update the cloud. In order to allow the cloud to send messages back to the TI IoT gateway, two additional message types are defined that allow the cloud to update the wireless network state and send actuation messages to specific devices in the network.

##### 2.3.1.1.1 Network Information Message Type (From TI IoT Gateway to the Cloud)

This message type presents information about the wireless network, its current state, and a list of devices that are connected to the network. As described later in this design guide, this will be the first message type sent after the network is initialized.This message type contains all the information necessary to prepare for receiving sensor data from devices. This message type contains the following fields:

- name: begins as the short address of the network but allows for the cloud to provide a more specific name
- channels: list of channels that the wireless network is operating on
- pan_id: the 16-bit PAN identifier of the network
- short_addr: the 16-bit short address of the PAN-coordinator
- ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device
- security_enabled: yes, if security enabled; no, otherwise
- mode: network operation mode (beacon, non-beacon, frequency hopping)
- state: PAN-coordinator state values (waiting, starting, restoring, started, open, closed)
- devices: list of wireless nodes in the network
  - – name: begins as the short address of the device but allows cloud to update

– short_addr: the 16-bit short address of the PAN-coordinator

– ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device

– topic: the topic that the device will send its sensor data updates to

– object_list: list of IPSO alliance smart objects (sensors) attached to this device

   • oid: object ID which specifies the sensor type in the IPSO standard

   • iid: list of instance IDs for the current object (can be multiple same type sensors)

### 2.3.1.1.2    Device Information Message Type (From TI IoT Gateway to the Cloud)

This message type provides information about the wireless device as well as the latest data for all of the sensors connected to the device. This message type will be sent when a device reports sensor data or switches between an active or inactive state. The following fields are contained in this message type:

• active: whether or not the wireless node is active

• ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device

• rssi: received signal strength indicator of the last message received

• smart_objects: list of the IPSO alliance smart objects connected to this wireless device

   – object ID description: type of sensor (as defined in the IPSO standard); can be multiple types of sensors connected to each device

   – instance ID: the instance ID for the parent object type; can be multiple sensors of the same type

   – resource ID description list: sensor data name value pairs (for example, *sensorValue*: 32.5, *units:* Celsius, and so forth); these resources match what is specified for the given object ID in the IPSO standard

### 2.3.1.1.3    Update Network State Message Type (From Cloud to TI IoT Gateway)

In the current implementation of the TI IoT gateway, this message type is intended to be able to open or close the wireless network to new devices joining. The cloud's front end user interface can allow a user to click a button to open or close the network and then generate this message type and send it to the TI IoT gateway. The gateway will then notify the network on whether it needs to open or close to new device joins. This message type only includes the desired state of the network and should be sent to the same topic that the cloud is receiving the network information messages from. The following field is all that is required:

• state: should be set to either *open* or *closed*

### 2.3.1.1.4    Device Actuation Message Type (From Cloud to TI IoT Gateway)

This message type is added to allow the cloud to send actuation messages to specific devices in the wireless network. The current implementation only supports toggling an LED on the wireless device's board. The device actuation message should be sent to the topic of the device as given in the devices list of the network information message. The following field is the only requirement for this message:

• toggleLED: should be set to *true*

### 2.3.1.2    Data Flows

#### 2.3.1.2.1    Network Information Sent to the Cloud

The following bulleted items are the list of events that can occur on the TI IoT gateway that will cause a network information message type to be sent to the cloud. A description is given with each event and the end of this section describes the expected behavior from the cloud upon receipt of this type of message.

*   **Network Startup**

    This is the initial event in the TI IoT gateway. The TI IoT gateway will aggregate the information about the wireless network as well as the list of connected devices and their sensor types. The TI IoT gateway will then make a connection to the cloud and send the aggregated data contained in the network information message type.

*   **Network Information Update**

    This event can occur if any of the information about the wireless network changes. For example, if the network operation mode of the wireless network was changed, the TI IoT gateway would once again aggregate all the information needed (network information and device list) and send the network information message type to the cloud.

*   **Network State Change**

    This event occurs if the state of the wireless network changes. For example, if the network state changes from open to closed the TI IoT gateway will send a network information message type to the cloud.

*   **Device Joins the Wireless Network**

    When a new device joins the network, after the network is up and running, this event will occur. In this case, the TI IoT gateway will add the new device and its information to the devices list within the network information message type and then send the updated information to the cloud.

*   **Expected Cloud Behavior**

    It is expected that the cloud will be prepared for the network startup event and will be able to receive the network information message type (using a wildcard and then filtering or by having prior knowledge about the destination or topic of the message). Once the cloud receives the network information message, the wireless network information (PANID, security, mode, and so on) can be displayed to users and the device list information (topic, object list, and so on) can be used to prepare itself to receive and display device and sensor data.

#### 2.3.1.2.2    Device Information Sent to the Cloud

The following bulleted items are the list of events that will cause the TI IoT gateway to send a device information message type to the cloud. A description is given with each event and the end of this section describes the expected behavior from the cloud upon receipt of this type of message.

*   **Device Becomes Inactive**

    This event occurs when the TI IoT gateway detects that one of the devices in the connected devices list has stopped sending sensor data updates. The TI IoT gateway will update the *active* field and send a Device Information Message Type to the cloud for the inactive device.

*   **Device Reports Sensor Data**

    Each time a sensor on a connected device reports sensor data this event occurs. The TI IoT gateway updates the IPSO Alliance Smart Object list in the device for each sensor and then sends a device

information message type to the cloud.

- **Expected Cloud Behavior**

    It is expected that the cloud will be listening on each topic given in the connected devices list from the network information message. When one of the two events occur in this section, the TI IoT gateway will send the device information message to the topic (corresponding to the device being update) that the cloud should be listening on or subscribed to. When the device information message arrives at the cloud, the cloud should display the latest device information and sensor data to users.

### 2.3.1.2.3    Update Network State Message Sent to the TI IoT Gateway

This message is used to open or close the wireless network to new devices joining. This message should be an option provided to users in the front end user interface that the cloud presents. When the user decides to update the network state, the cloud should send an update network state message type to the TI IoT gateway on the same topic that the network information messages are arriving on.

- **Expected TI IoT Gateway Behavior**

    The TI IoT gateway will receive the Update Network State message and will generate the correct command (either open or close) to the wireless network. This message should in turn cause a network state change event (from 7.2.1 above) that will send a network information message back to the cloud which can confirm the successful completion of the Update Network State command.

### 2.3.1.2.4    Device Actuation Message Sent to the TI IoT Gateway

This method is used to toggle the LED on the board of the connected devices. This message is meant to be a proof-of-concept on the current device setup and will change for customer use-case specific actuations. A toggle LED button for each device will be provided to users of the cloud's front end interface. When the toggle LED button is clicked the cloud should send a device actuation message to the TI IoT gateway on the same topic that the device information messages are arriving on.

- **Expected TI IoT Gateway Behavior**

    The TI IoT gateway will generate a toggle LED command and send it to the device corresponding to the topic that the device actuation message was received on. This will cause the LED to toggle. Because the state of the LED is not captured in the device information message type, there will be no feedback to the cloud that the LED actually toggled.

## 3 Hardware, Software, Testing Requirements, and Test Results

### 3.1 Required Hardware and Software

The CC3220SF plus CC13x0/CC13x2 sensor-to-cloud reference design helps developers create ULP, long-range, star-topology network solutions. The sensor-to-cloud reference design includes the Gateway example application running on the CC3220SF, MAC CoProcessor (CoP) application running on the CC13x0 or CC13x2, in addition to sensor node applications. The CC3220SF Gateway example application interfaces over UART with a CC13x0 or CC13x2 LaunchPad, which acts as a MAC CoP. The Gateway example application implements a IEEE 802.15.4 full-function device, which performs the functions of a network PAN coordinator (starting a network and permitting devices to join this network) and also provides an interface for monitoring and collecting sensor data from one or more sensor devices.

The Gateway example application provides an IEEE 802.15.4 network to IP bridge and is a great starting point to create IoT applications based on TI 15.4-Stack.

#### 3.1.1 Hardware

- 2× CC13x0 or 2× CC13x2 LaunchPad development kits
- 1× CC3220SF LaunchPad development kit
- USB cables
- Wi-Fi access point with Internet access

#### 3.1.2 Software

- CC3220-SensorToCloud SW
- CC3220 SDK v1.50.00.06
- SimpleLink CC13x0 SDKor SimpleLink CC13x2 SDK
- UniFlash v4.1.1.1250 or later
- Tera Term or any other equivalent terminal program
- Cloud Foundry CLI (for IBM Cloud only)
- (Optional) SimpleLink Starter Pro IOS® app or SimpleLink Wi-Fi Starter Pro Android™ app (downloaded from the app store on smartphones or tablets)

## 3.2 Testing and Results

This section describes the hardware and software used for running the tests and the results obtained.

### 3.2.1 Test Setup

During the development process of this reference design, the full hardware and software portions described in earlier sections were used for testing. Multiple CC13x0 and CC13x2 sensor nodes and a CC3220SF LaunchPad (connected to a CC1310 coprocessor) were used to verify the IoT gateway functionality with the IBM cloud and AWS IoT cloud services. The test results of this reference design can be visualized by the IoT dashboard shown in 节 3.2.2.

#### 3.2.1.1 Running the Out-of-Box Examples

This section provides detailed instructions to assist developers set up and understand the principles behind the out-of-box examples.

Some of the topics covered in this section are:
1. Programming the CC13xx LaunchPad development kits
2. Programming the gateway example application on the CC3220SF LaunchPad
3. Connecting the CC13x0 *MAC-CoP* LaunchPad with the CC3220SF LaunchPad
4. Setting up and configuring the cloud service
5. Running and using the example

For steps on how to setup and run the *IBM gateway example* please follow the instructions in 节 3.2.1.1.1 and to setup and run the *AWS IoT gateway example* please follow the instructions in 节 3.2.1.1.2.

> 注**:**     This guide can be performed using either CC13x0 or CC13x2 LaunchPad development kits for the MAC CoP. The training material is based on CC1350, and the same procedures apply when using either a CC1310 or CC13x2 devices.

#### 3.2.1.1.1 Running IBM Gateway Example

1. Label one CC1350 LaunchPad as *Sensor* and the other as *MAC-CoP*. These labels will be referred to throughout this guide. It is recommended to use non-permanent marking for this (for example, sticky notes) as these labels may only be relevant for this specific example.
2. Program the *MAC-CoP* and the *Sensor* LaunchPad by following the steps in 节 3.2.1.1.3.
3. Program the CC3220SF LaunchPad by following the steps in 节 3.2.1.1.4.
4. Connect the MAC-CoP LaunchPad with the CC3220SF LaunchPad by following the steps in 节 3.2.1.1.5.
5. Set up an account for IBM Cloud by following the instructions in 节 3.2.1.1.6.
6. Set up the Watson IoT platform service as in 节 3.2.1.1.7.
7. Follow 节 3.2.1.1.8 to set up Node.js and cloud foundry app.
8. After all of these steps have been completed, go to 节 3.2.1.1.9, and follow the steps to connect the gateway to the Internet and run the example.

### 3.2.1.1.2 *Running AWS IoT Example*

Before getting started with the AWS IoT example please request the AWS IoT demo configuration from StackArmor by filling out the following form.

1. Label one CC1350 LaunchPad as Sensor and the other as MAC-CoP. These labels will be referred to throughout this guide. It is recommended to use non-permanent marking for this (for example, sticky notes), as these labels may only be relevant for this specific example.
2. Program the *MAC-CoP* and the Sensor *LaunchPad™* by following the steps in 节 3.2.1.1.3.
3. Import and build the AWS IoT example by following the steps in 节 3.2.1.1.4.2.
4. Program the CC3220SF LaunchPad by following the steps in 节 3.2.1.1.4.4.
5. Connect the MAC-CoP LaunchPad with the CC3220SF LaunchPad by following the steps in 节 3.2.1.1.5.
6. After all of these steps have been completed, go to 节 3.2.1.1.9, and follow the steps on how to connect the gateway to the Internet and run the example.

### 3.2.1.1.3 *Programming CC13x0 LaunchPad™*

1. It is assumed that all the required software has already been installed. If not, install the required software节 3.1.2 now.
2. Connect the *CC13x0* LaunchPad to the PC.
3. Open UniFlash.
4. Select LAUNCHXL-CC1350 as show in 图 4, and click on the *Start* button.
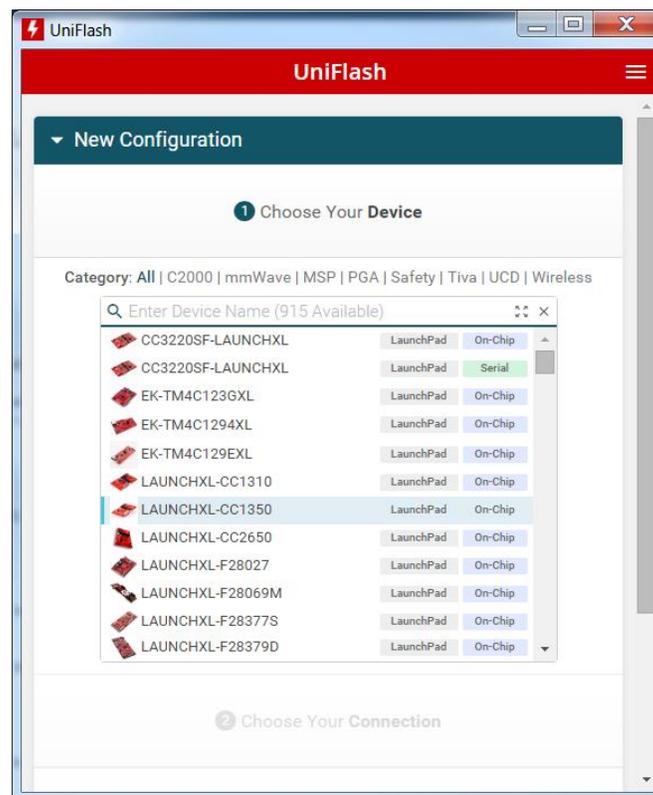


**图 4. Choose Device: UniFlash**

5.  Make sure the *Program* tab is selected on the left, and click the *Browse* button to select the desired image for the CC13xx LaunchPad located in <S2C_Repo_Directory>\tidc01002\prebuilt\.
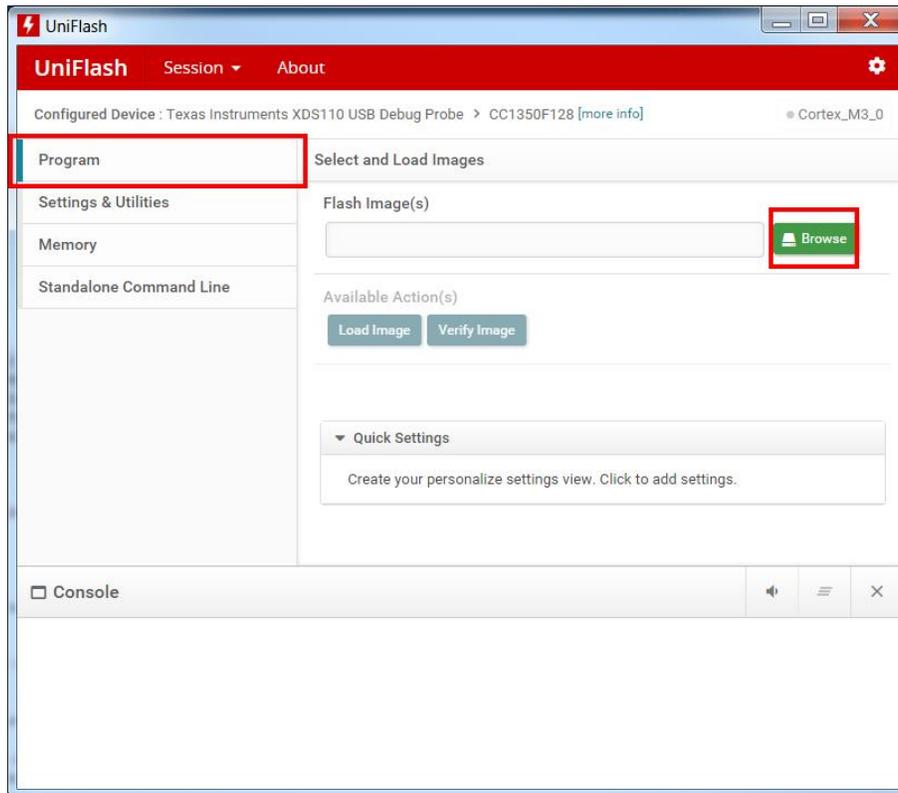


图 **5. Browse for Firmware Image**

6. After selecting the desired image, click on the *Load Image* button to flash the CC1350 LaunchPad.
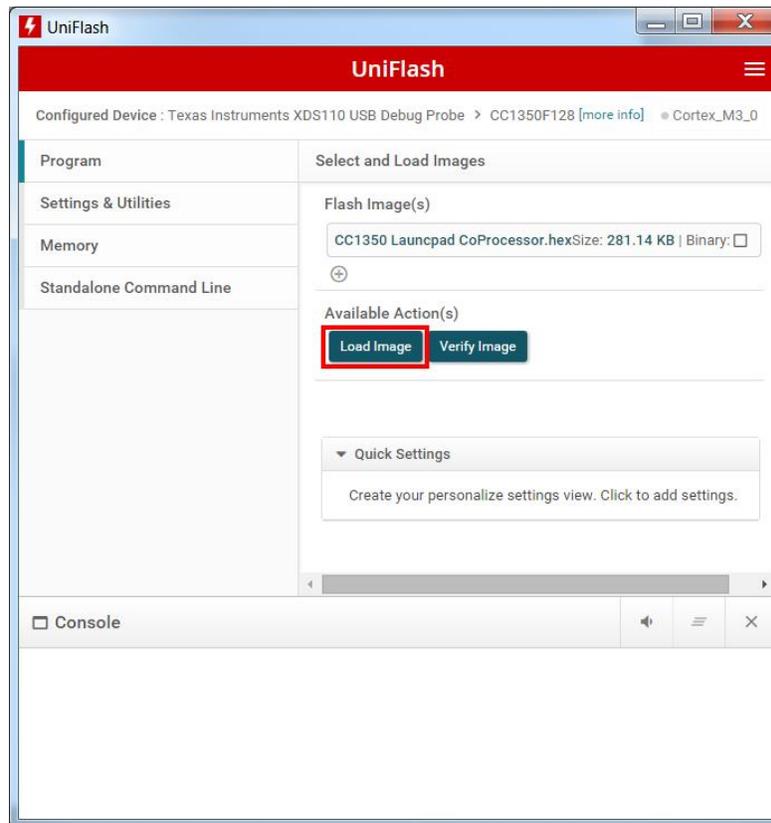


图 **6. Load Image**

7. If loading the image was successful, a success message should show in the console as shown in 图 7.
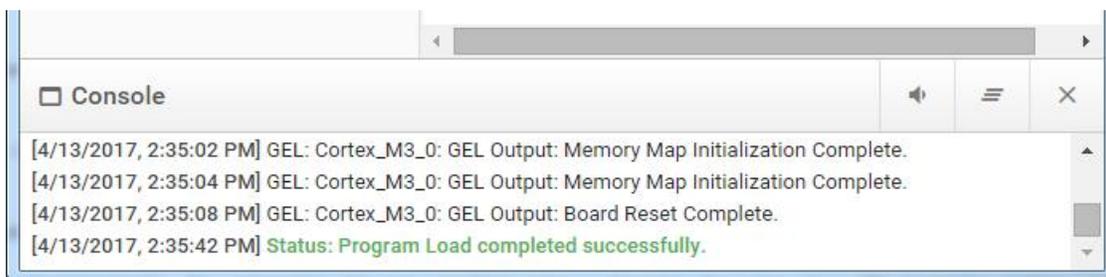


图 **7. Successful Load**

### 3.2.1.1.4    *Programming the CC3220SF LaunchPad™*

This section describes two ways of programming the CC3220SF LaunchPad. 节 3.2.1.1.4.1 explains the process of programming the CC3220SF LaunchPad by importing a preconfigured Image Creator project to UniFlash. 节 3.2.1.1.4.4 shows how to create an Image Creator project from scratch to program the CC3220SF with a binary generated from Code Composer Studio™ (CCS).

### 3.2.1.1.4.1  *Programming a Preconfigured Image Creator Project*

Note that only the IBM cloud service is provided in the preconfigured Image Creator project since the AWS IoT example needs extra configuration that need to be added and compiled in the source code.

1. Open UniFlash.
2. On the *Choose your Device* section select *CC3220SF-LAUNCHXL*, and make sure to select the *Serial* option and not *On-Chip* as shown in 图 8. Click on the *Start Image Creator* button.
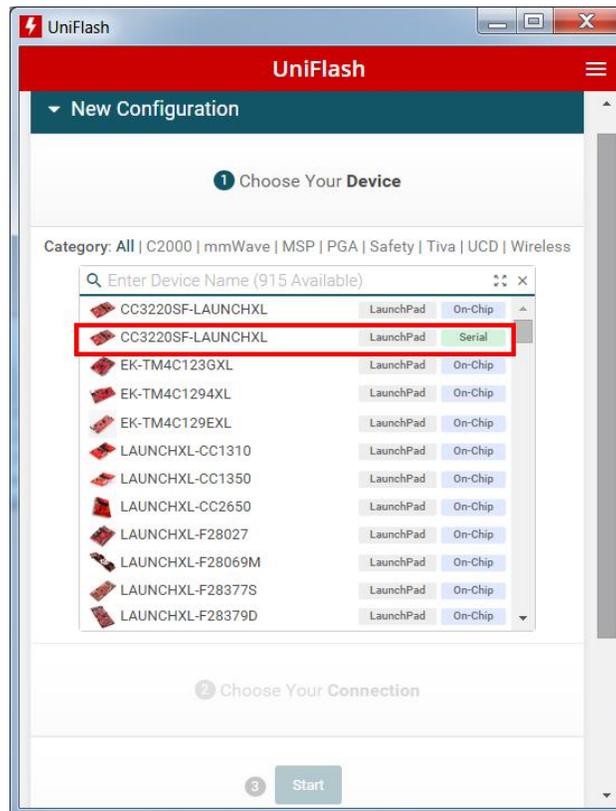


**图 8. Choose Device: CC3220SF**

3. After starting Image Creator, click on the *Manage Projects* button as shown in 图 9.
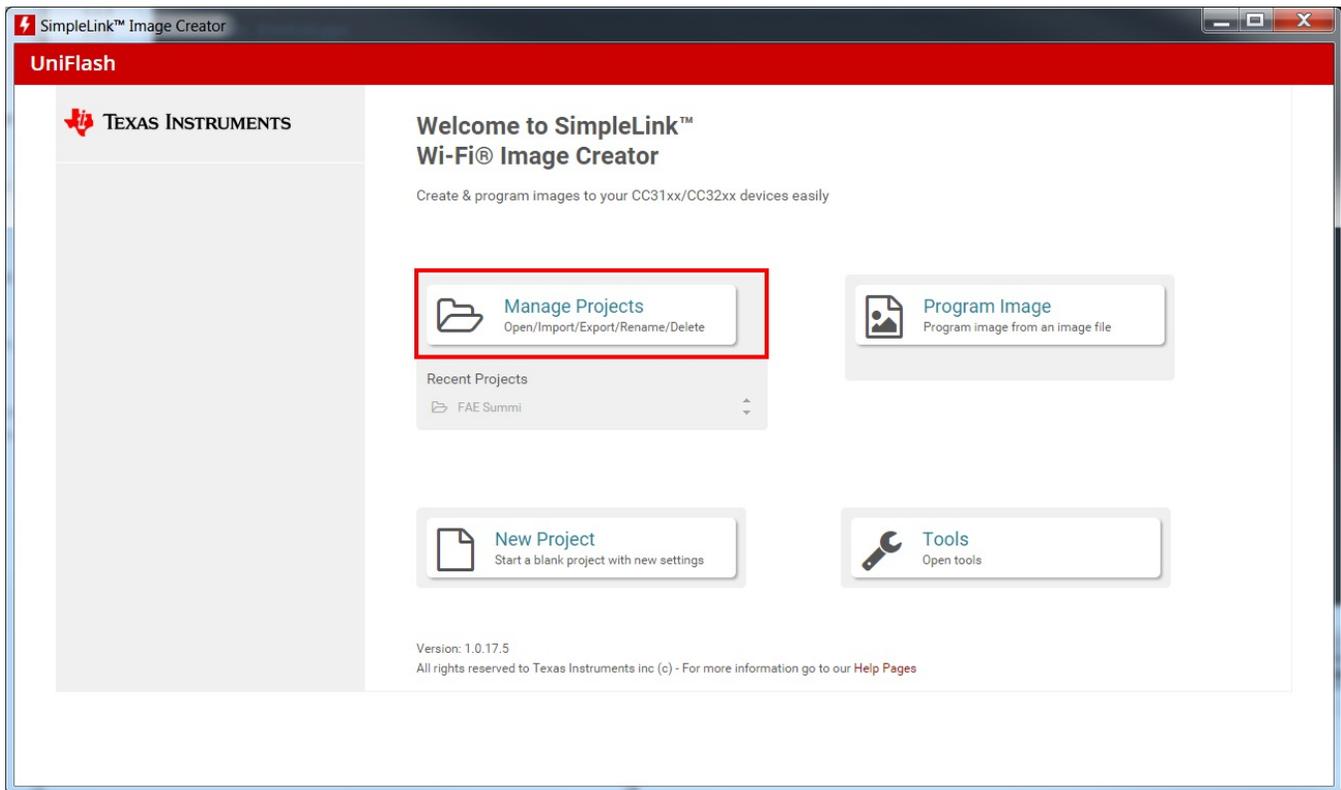


图 **9. Manage Project**

4. Click on the *Import Project from ZIP file* button, and select the zip folder
   *C:\<S2C_Repo_Directory>\tidc01002\prebuilt\CC3220SF_LaunchXL\Uniflash_CC3220ImageCreatorPr oject.zip*.
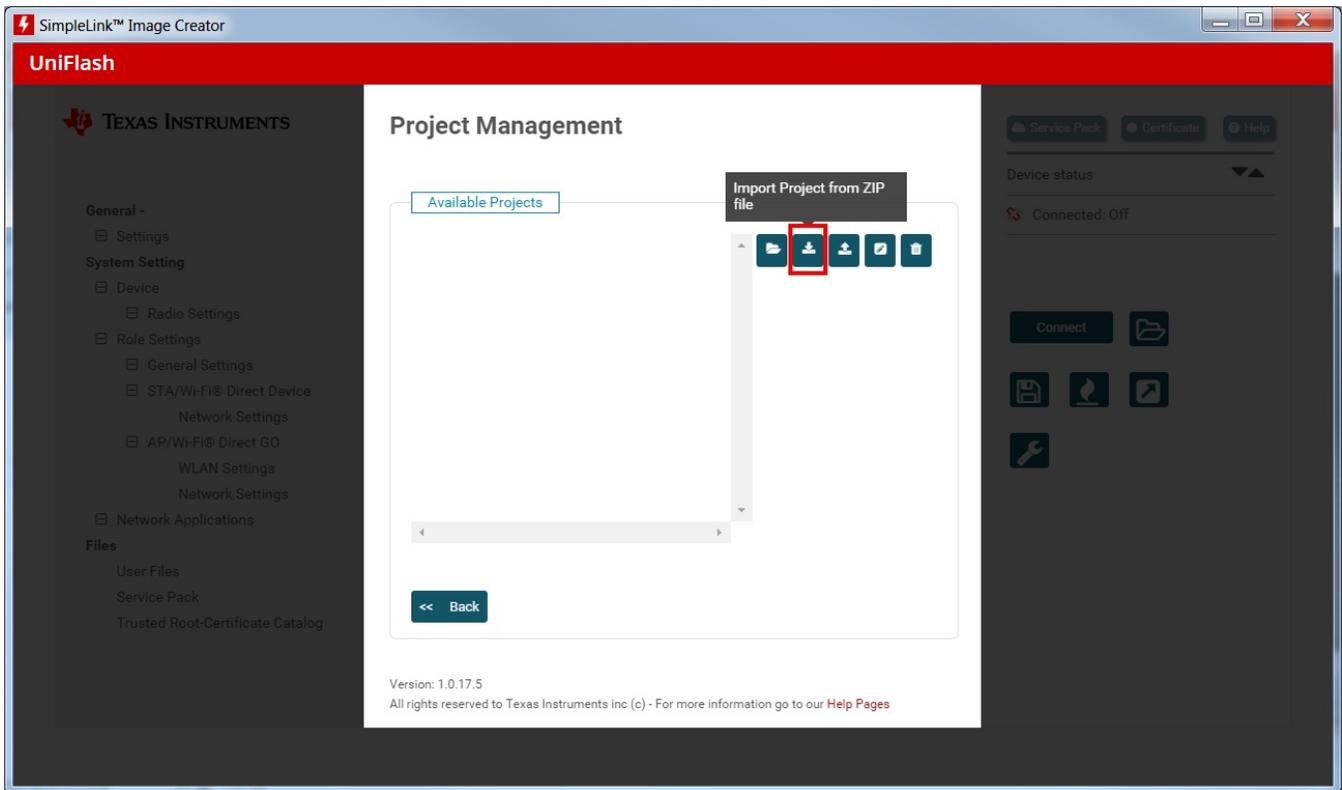


图 **10. Import Project**

5. Open the *CC3220SF_154StackGtway* project.
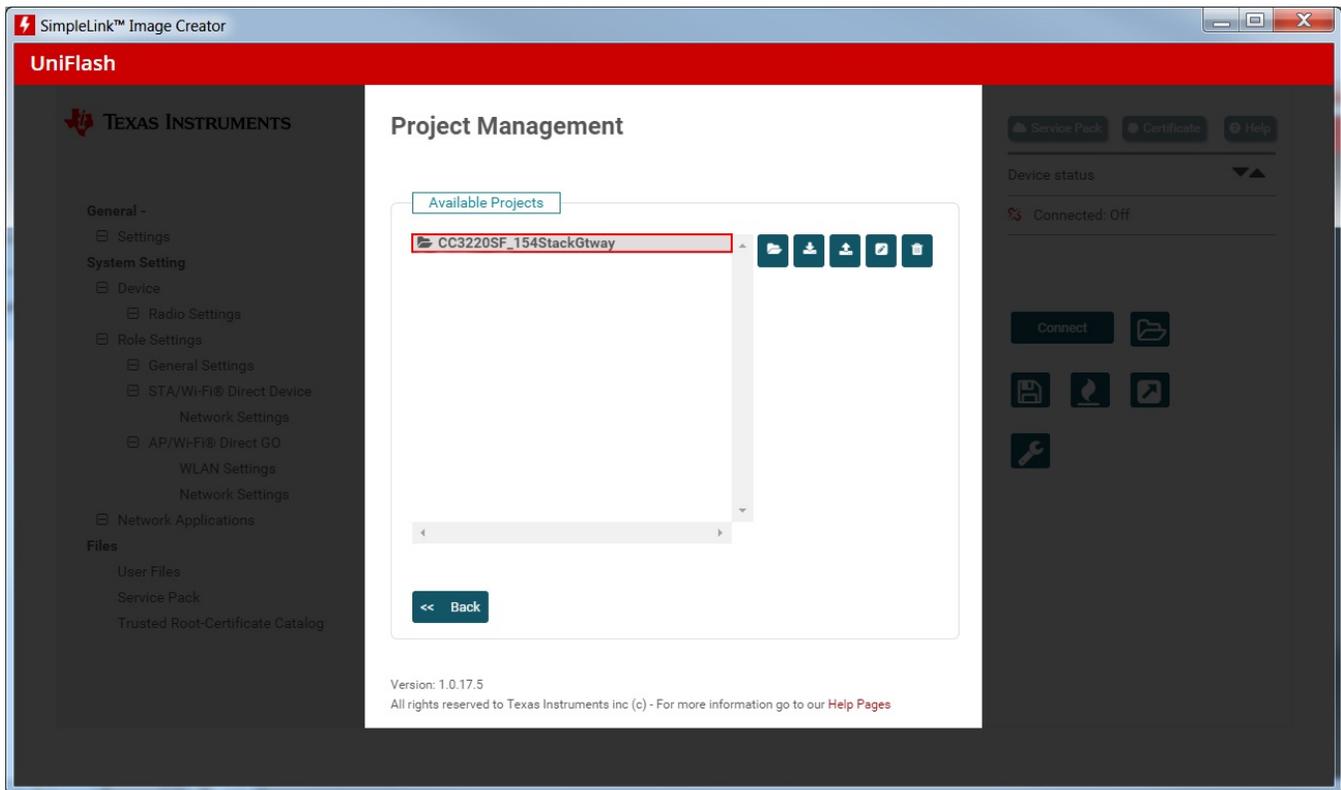


图 **11. Open Project**

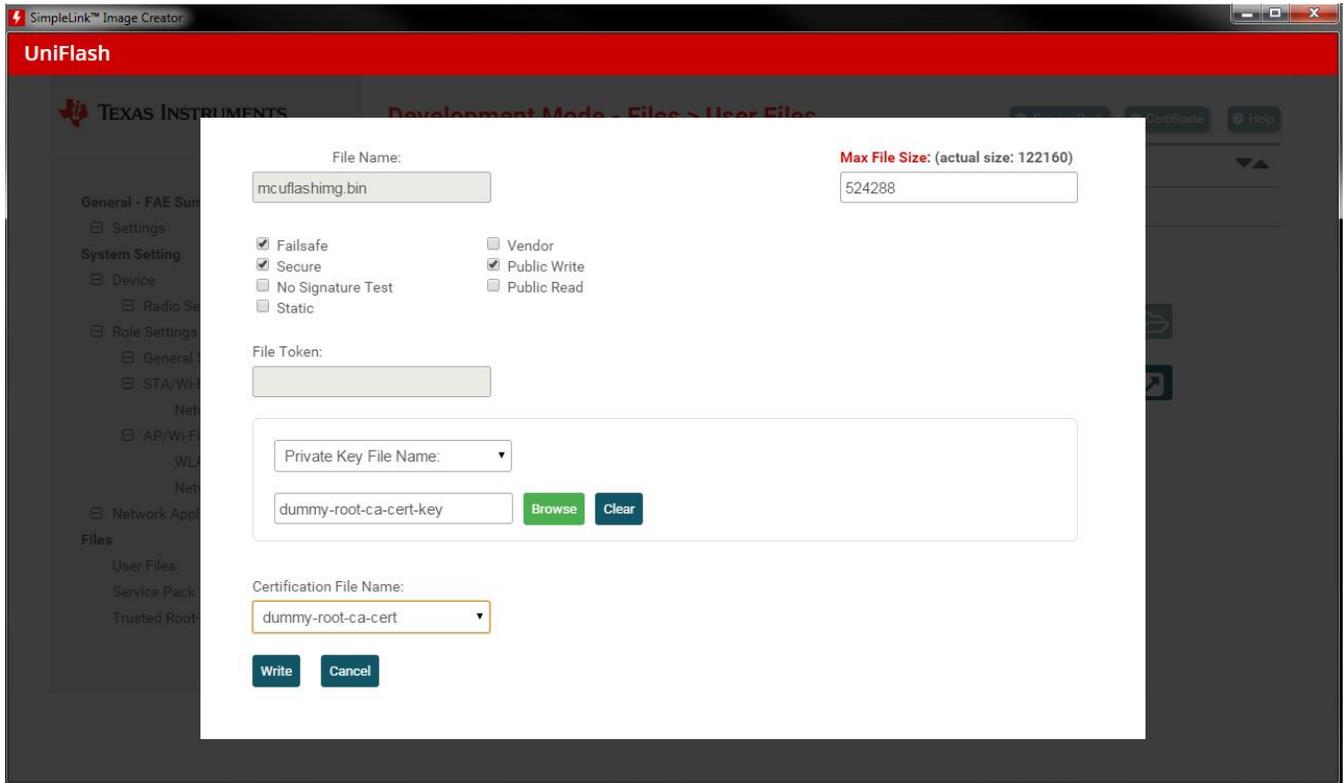6.  Connect the device to the PC through a USB cable, and press the *Connect* button found on the bottom-right corner. Once the device is connected, select the *Generate Image* button underneath the *Disconnect* button, and select *Program Image (Create & Program)*.
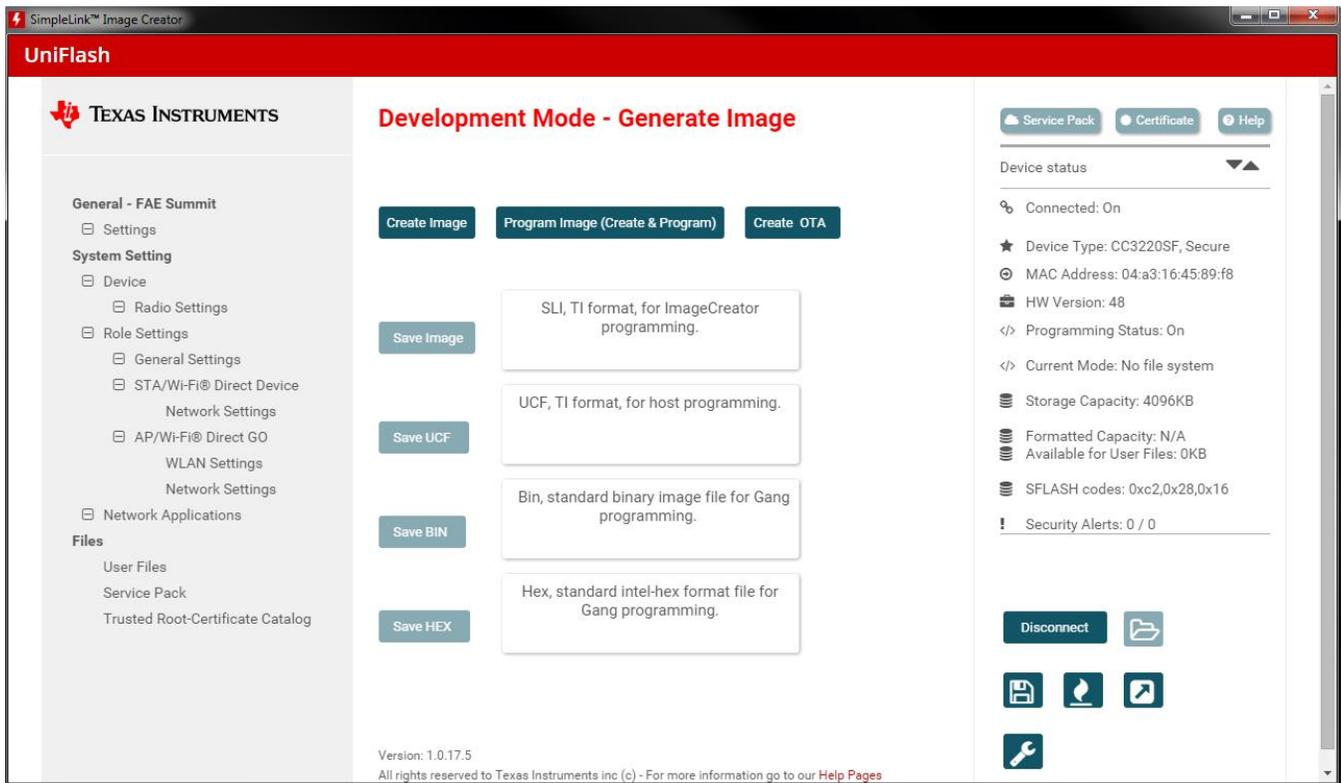


图 **12. Generate Image**

### 3.2.1.1.4.2 *Importing Examples to Code Composer Studio (CCS)*

1. Open CCS version 7.3.
2. It is assumed that the CC3220 SDK has already been installed in a directory referred to as *CC32XX_SDK_INSTALL_DIR*. If not, install the CC3220 SDK now as this example requires it.
3. Import the Gateway project by going to *File → Import → C/C++ → CCS Projects*, and click on *Next*.
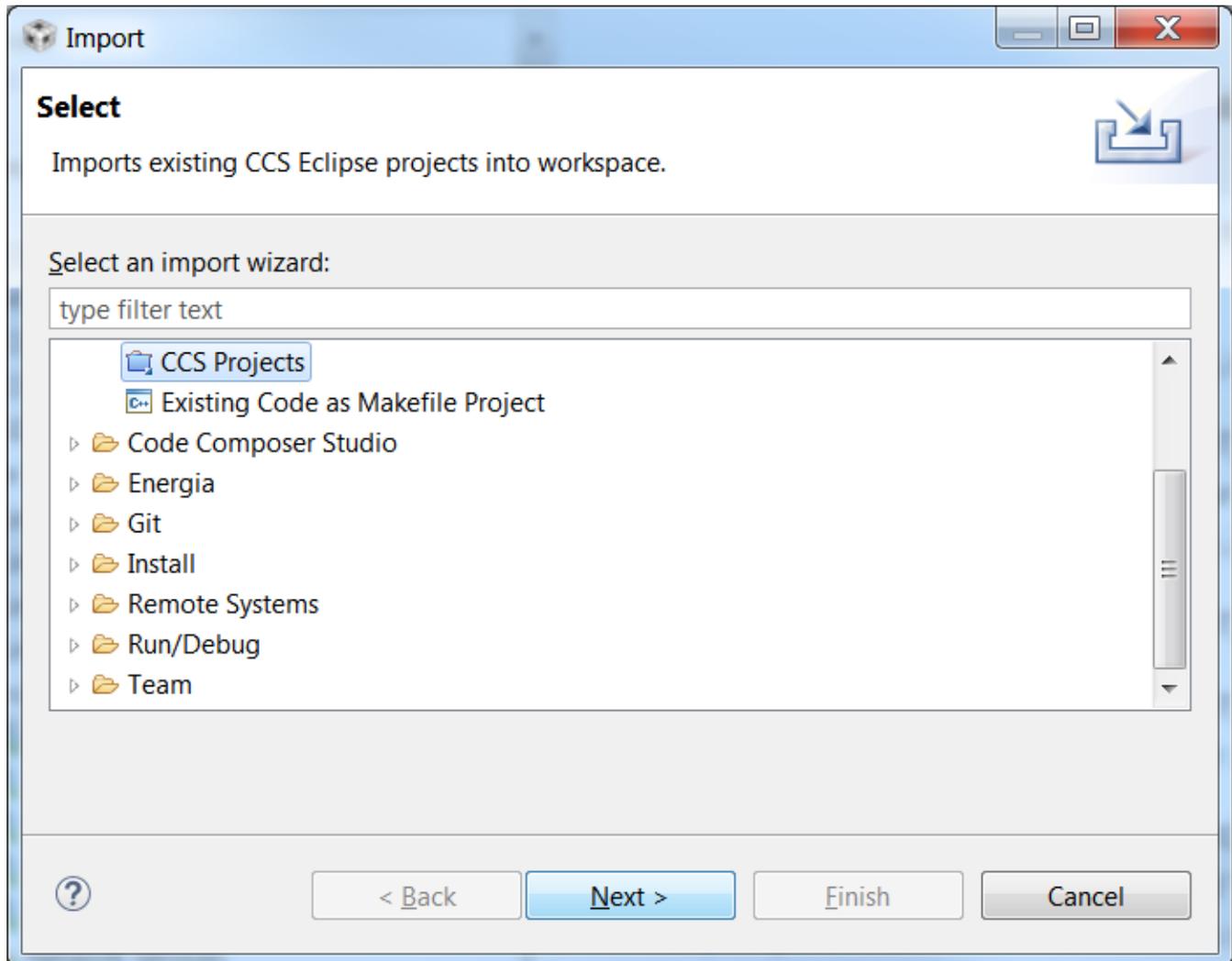


图 **13. Select Project to Import**

4.  Click on *Browse...* and navigate to *<S2C_Repo_Dir>\tidc01002\examples\cc3220sf_gateway_app*.
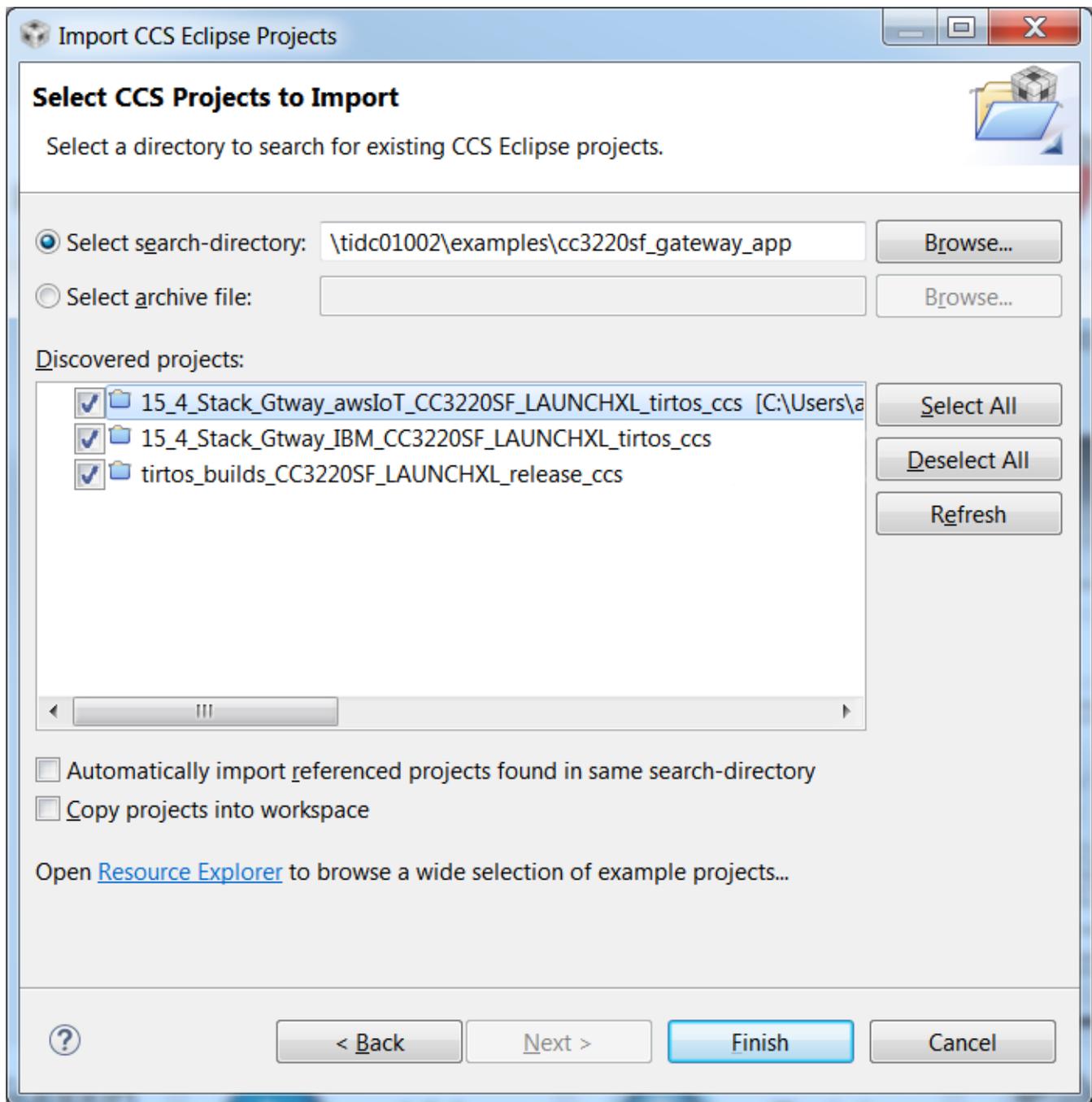


图 **14. Select Projects to Import**

5.  Select the desired examples to be imported and click on *Finish*.

6.  *(AWS Example Only)* Request AWS IoT demo configuration from StackArmor by filling out the following form.

7.  *(AWS Example Only)* Update the AWS configuration file:

    1.  After receiving the certificates and configuration information from StackArmos open the aws_iot_config.h file, found in CloudService/AWS.

    2.  Set the value of AWS_IOT_MQTT_HOST to the URL provided by StackArmor, it should look something like this "https://<random-string>.iot.us-east-1.amazonaws.com "

3. Modify AWS_IOT_MQTT_CLIENT_ID to a unique name for the device.

4. Replace the value of AWS_IOT_MY_THING_NAME with the extended address of the MAC-Cop LaunchPad.

5. Open the file certs.h, found in the CloudService/AWS directory.

6. Search for "USER STEP" and update the CA root certificate string, the client certificate string, and the client (private) key string.

8. Compile the example by clicking on the Build button (                                          ).

9. If the CC3220SF LaunchPad is already in developer mode, the example can be run and debugged

directly from CCS by clicking on the debug button (                                          ).

10. If the CC3220SF Launchpad is not in developer mode, follow 节 3.2.1.1.4.4 to flash and run the example.

### 3.2.1.1.4.3  Compiling the Mac-CoP from Source

These are optional steps if there is a need to re-compile the Mac-CoP firmware for the gateway as opposed to using the prebuilt CoP firmware.

When re-compiling the Mac CoP firmware, the pin configuration for the UART needs to be modified since the default Mac CoP example in the CC13xx SDK uses IOID 3 and IOID 2 pins for UART and the gateway uses IOID 11 and IOID 9. See steps below to modify the pin configuration.

1. Import the **coprocessor** example from the CC13xx SDK to CCS

2. Open the file CC13X0_LAUNCHXL.h and modify the two lines shown below.

```
/* UART Board */
#define Board_UART_RX              IOID_9          /* RXD  */
#define Board_UART_TX              IOID_11         /* TXD  */
```

3. Now rebuild the example and flash it into the CC13xx LaunchPad

### 3.2.1.1.4.4  Creating an Image Creator Project in UniFlash

The following steps will allow the user to customize the example and use the new, updated files instead of the pre-build ones.

1. Open UniFlash.

2. On the *Choose your Device* section, select *CC3220SF-LAUNCHXL*. Make sure the *Serial* option is selected and not *On-Chip* as shown in 图 15. Click on the *Start Image Creator* button.
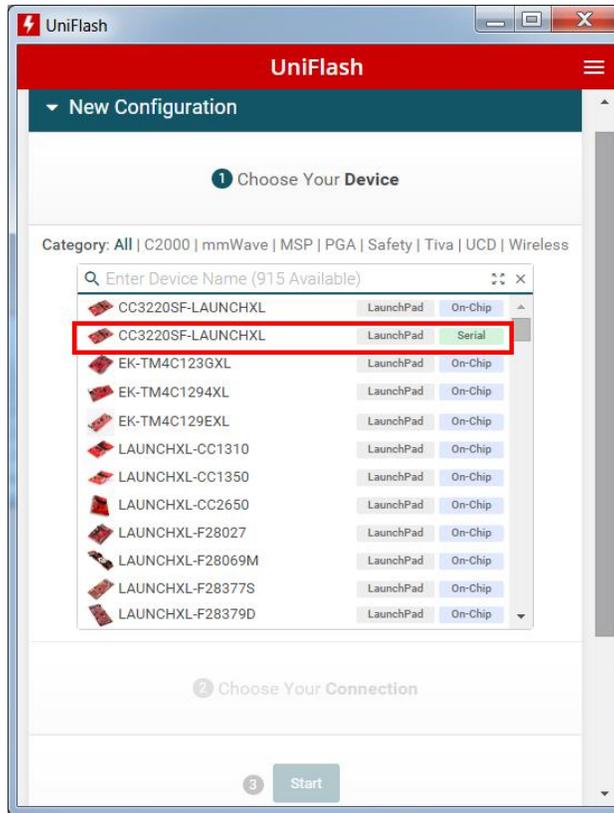
图 **15. Select CC3220 Device**

3. After starting Image Creator, click on the *New Project* button as in 图 16.
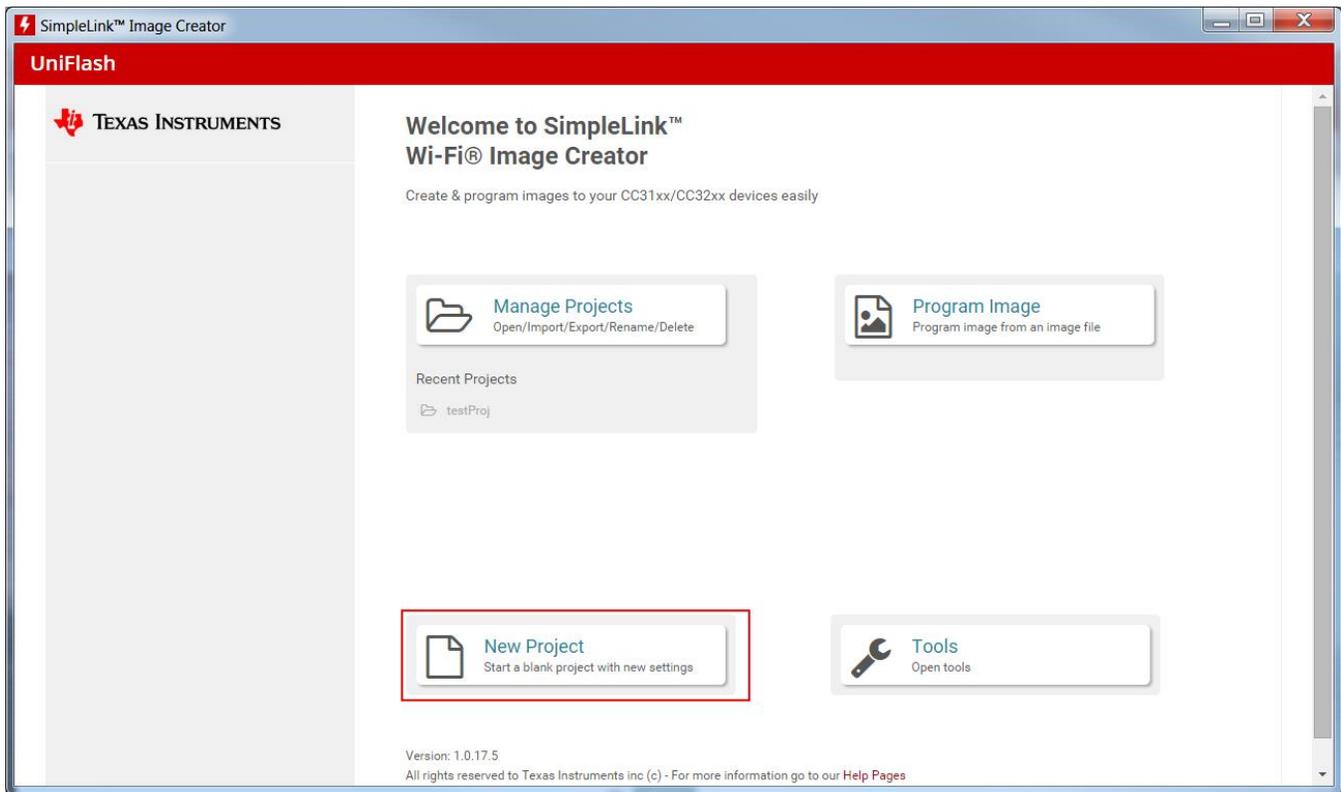


图 **16. New Project**

4. Enter a project name, select *CC3220SF* in the *Device Type* drop-down menu, make sure device mode is in *Develop*, and click on the *Create Project* button.
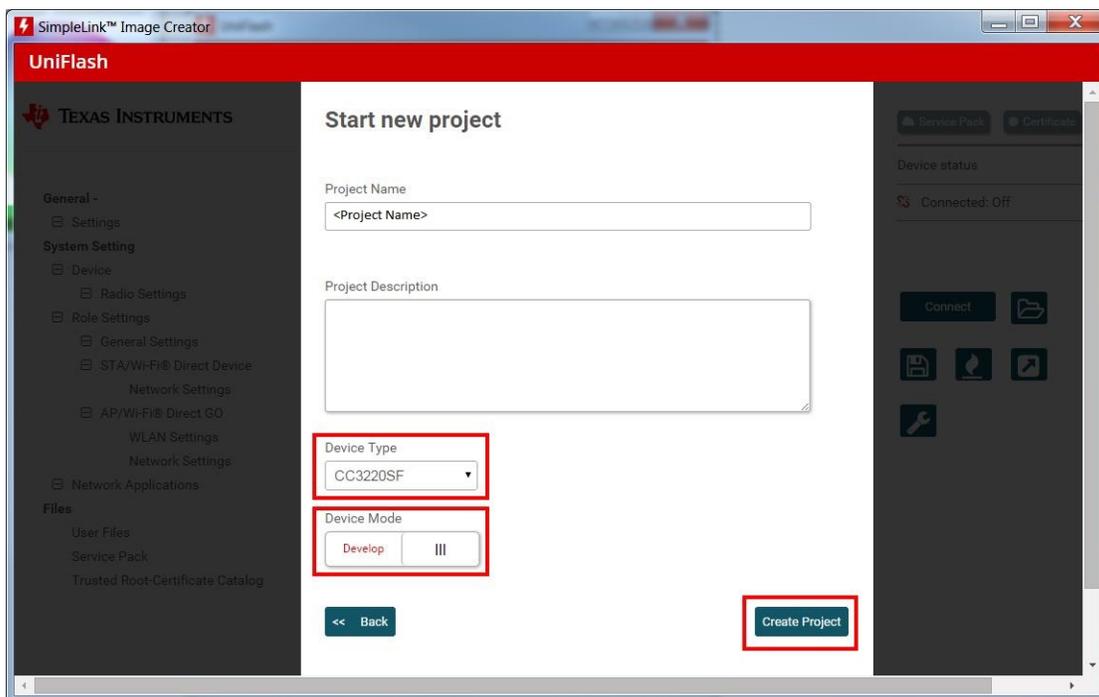


图 **17. Starting a New Project**

5.  Select *Trusted Root-Certificate Catalog* in the bottom-left corner and uncheck the *Use default Trusted Root-Certificate Catalog* box. Include the *Source File* (*certcatalogPlayGround20160911.lst*) and *Signature Source File* (*certcatalogPlayGround20160911.lst.signed.bin*) found in *C:\ti\simplelink_cc32xx_sdk_1_50_00_06\tools\cc32xx_tools\certificate-catalog*.
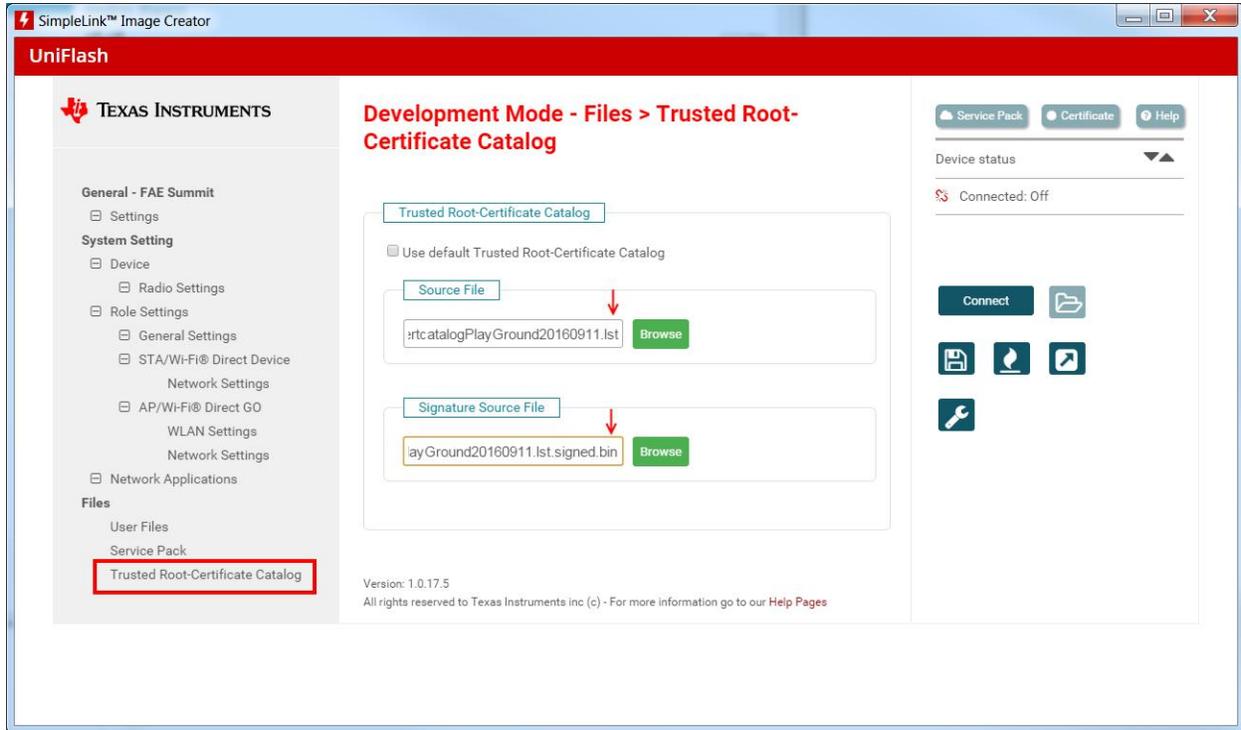


图 **18. Trusted Root Certificates**

6. Select *Service Pack* in the bottom-left corner and include the service pack bin
   (*sp_3.3.0.0_2.0.0.0_2.2.0.4.bin*) found in
   *C:\ti\simplelink_cc32xx_sdk_1_50_00_06\tools\cc32xx_tools\servicepack-cc3x20.*
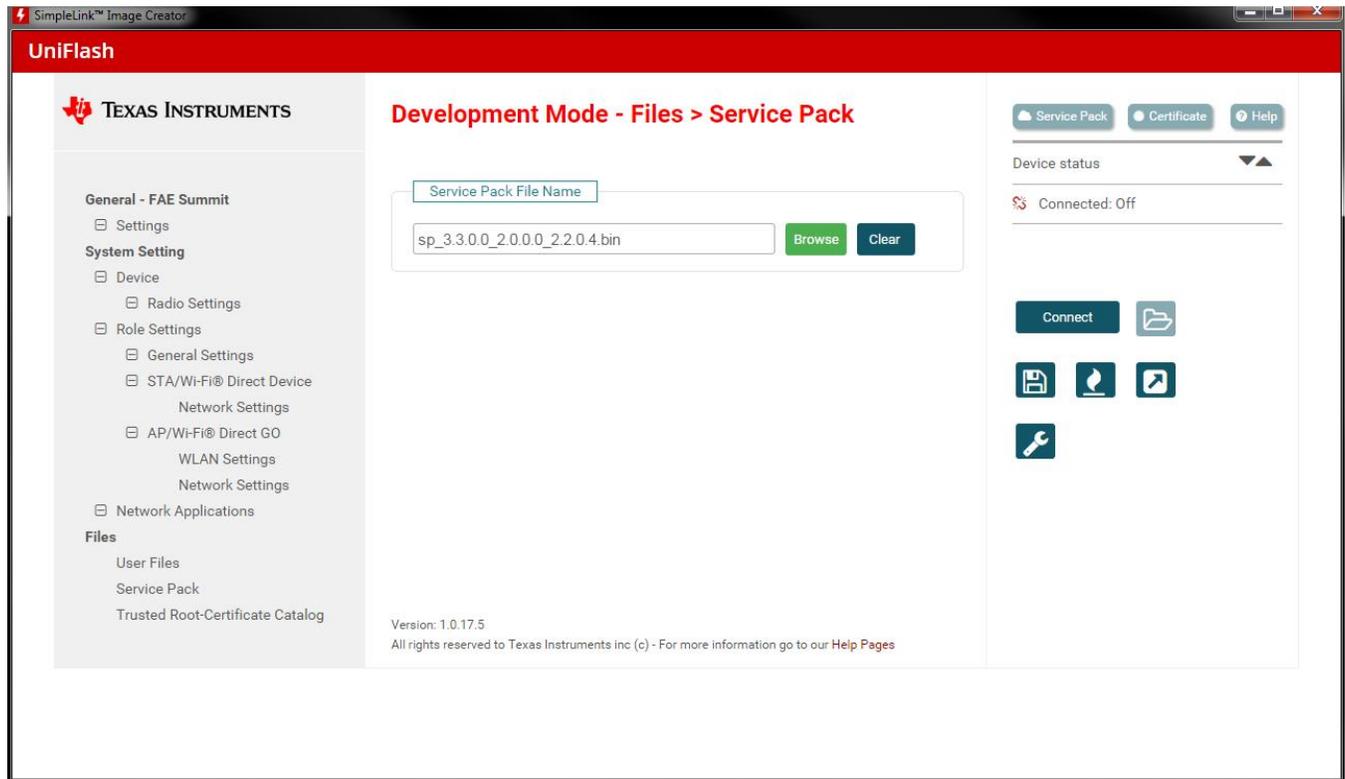


图 **19. Service Pack**

7. Select *User Files* and include the *dummy-root-ca-cert* and *dummy-root-ca-cert-key* files by clicking on the *Add File* icon. These files can be found in *C:\ti\simplelink_cc32xx_sdk_1_50_00_06\tools\cc32xx_tools\certificate-playground.Create* a folder named *www* by clicking on the *New Folder* icon. This folder will contain all the web server required files.

---

注**:**        When adding the files do not select any of the options in the pop-up window—just click the *Write* button.

---
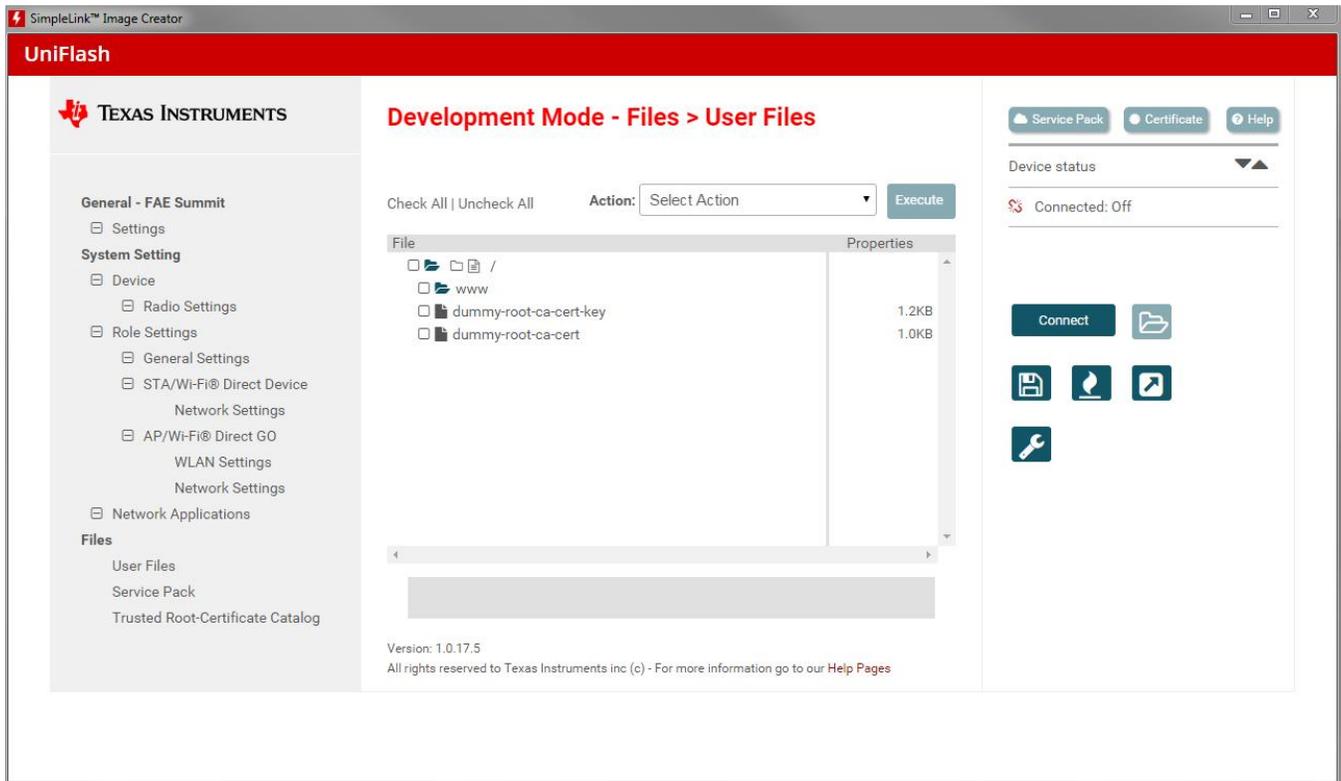


图 **20. User Files**

8. Go to the following directory: *C:\<S2C Repo Directory>\tidc01002\src\www*, and recreate the same folder structure in the *www* folder in the devices *User Files*.

9. After creating all the folders in the www folder, transfer all the files in *C:\<S2C Repo Directory>\tidc01002\src\www* to the *www* folder of the device.
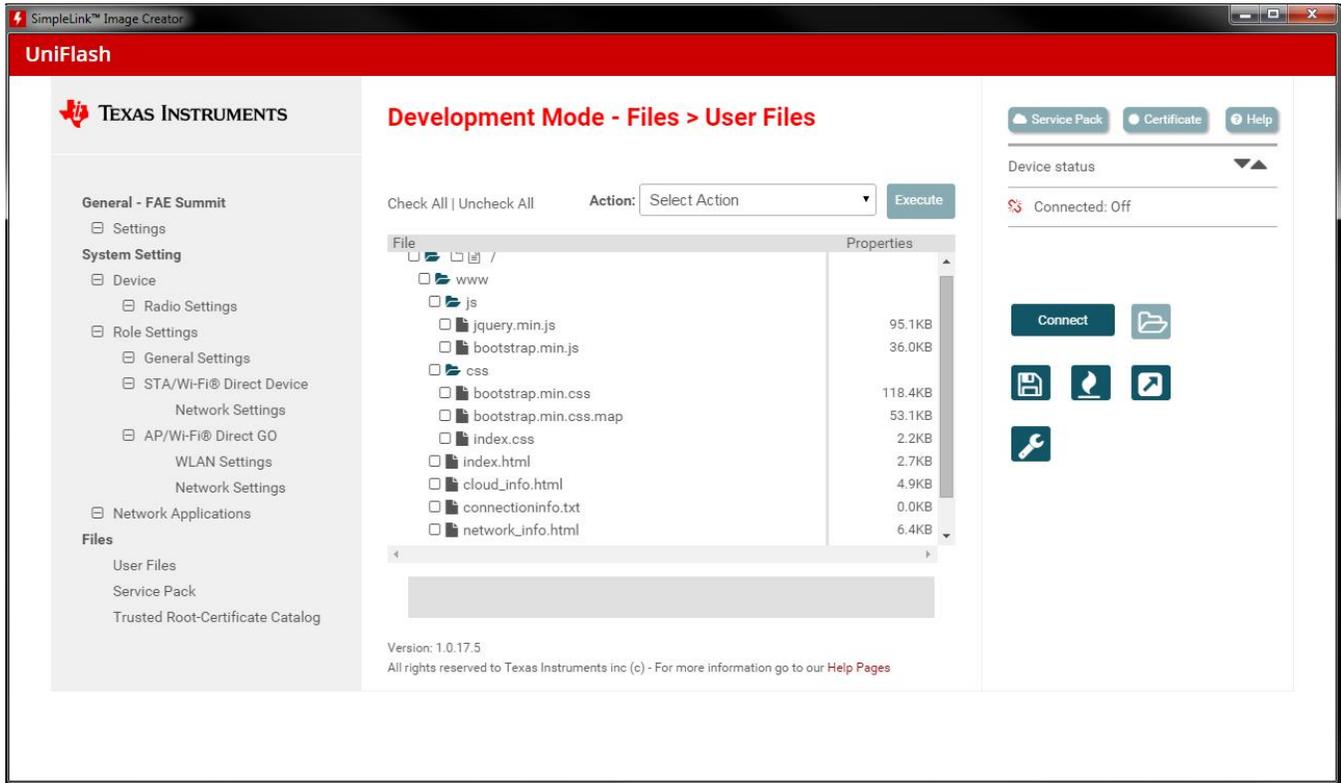


图 **21. www Folder**

10.  On the drop-down box in the top-right corner, select *Select MCU Image*, and press *Browse*.
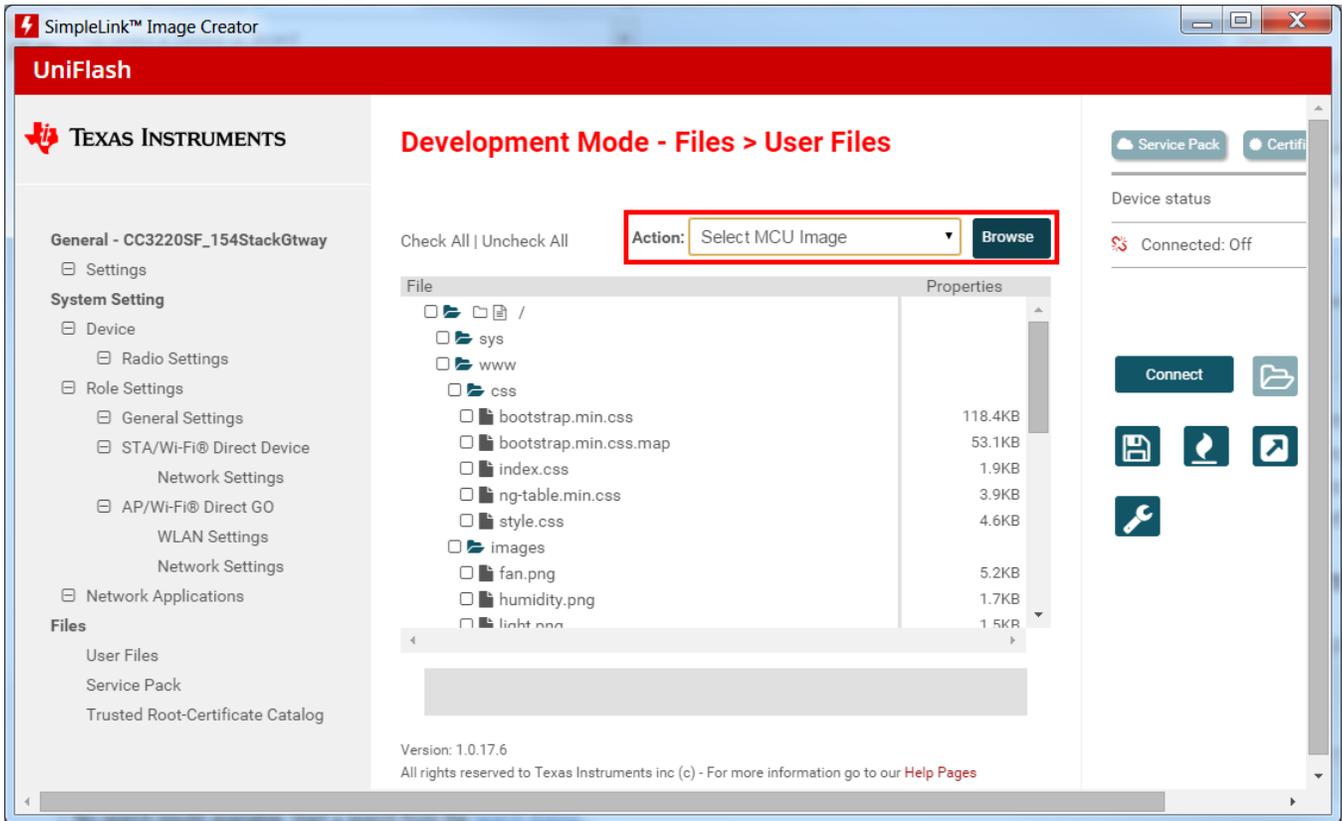


图 **22. Select MCU Image**

11. Navigate to the examples directory (*<S2C Repo Directory>\tidc01002\examples\cc3220sf_gateway_app*). Go into the directory of the example to be programmed in the CC3220sf and then go to the Debug folder. If the example has been successfully compiled in CCS then the Debug folder should have the files shown in 图 23. Select the *.bin* file.
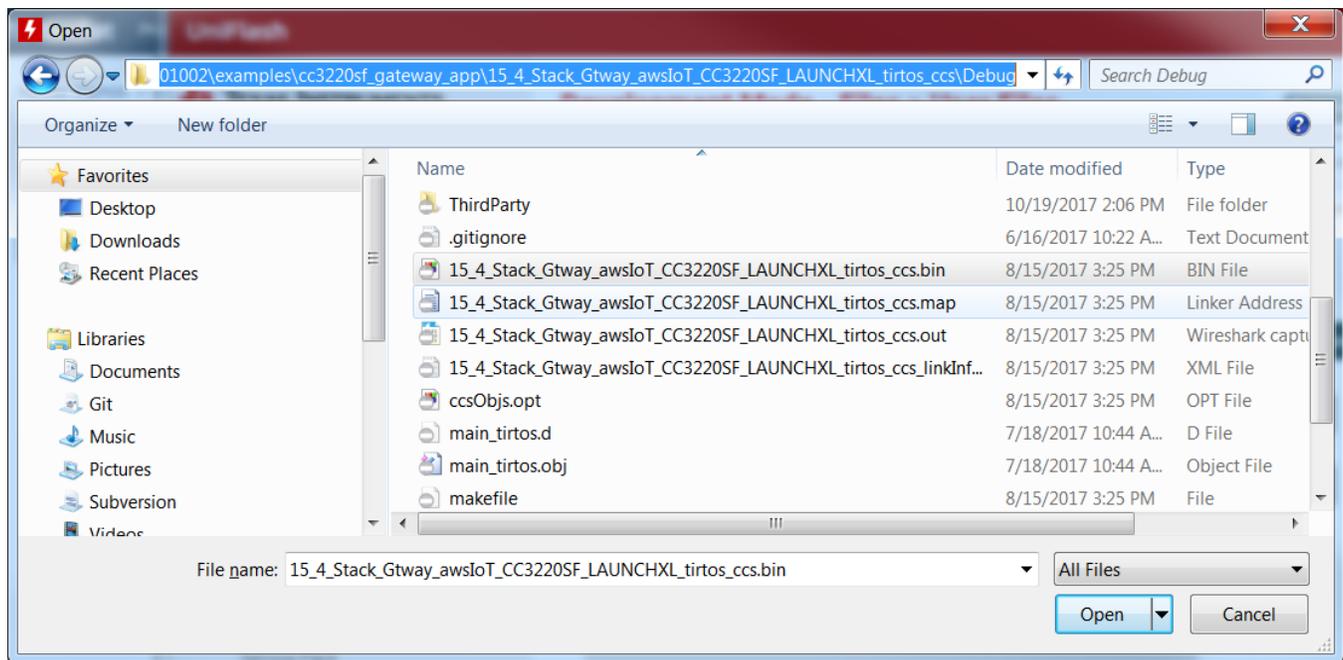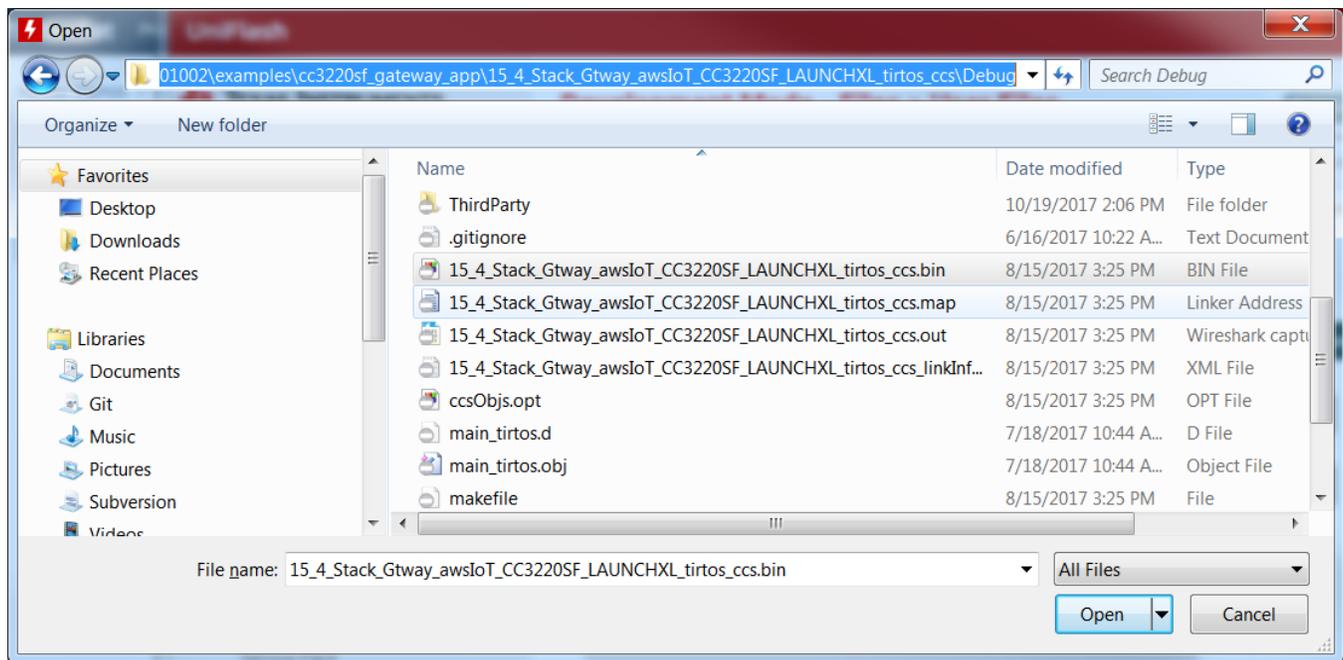


图 **23. Select Desired Example Binary**

12. On the next menu select *Private Key Name:* and include the dummy-root-ca-cert-key. On the *Certification File Name:* select *dummy-root-ca-cert* from the list.



图 **24. Select MCU Image**

13. Connect the device to the PC through a USB cable, and press the *Connect* button found on the bottom-right corner. Once the device is connected, select the *Generate Image* button underneath the *Disconnect* button. Select *Program Image (Create & Program)*.
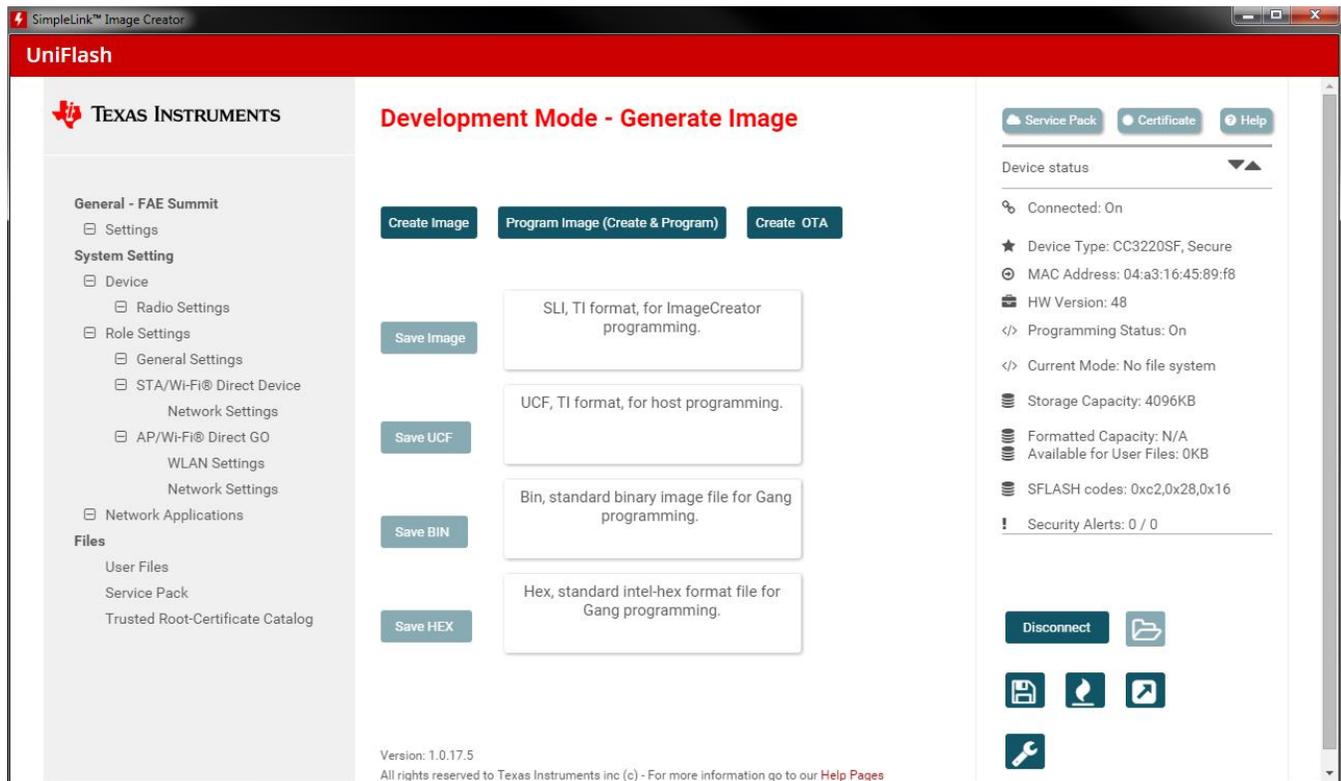


图 25. Generate Image

### 3.2.1.1.5 *Connecting the MAC-CoP and the CC3220SF LaunchPad™*

The following procedure applies for both CC13x0 and CC13x2.

1. Remove all jumpers at the center of the *MAC-Cop LaunchPad* except the *Reset* jumper highlighted in *green* . Also move the *VSENSE* jumperhighlighted in *blue* in 图 26 to the Extern Pwr position.
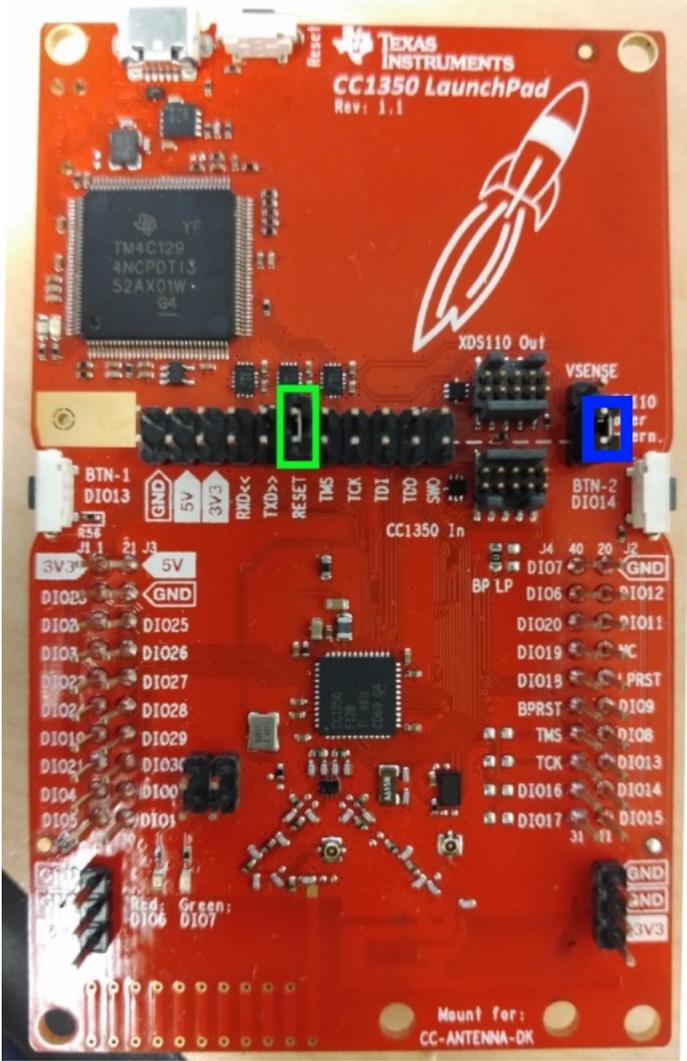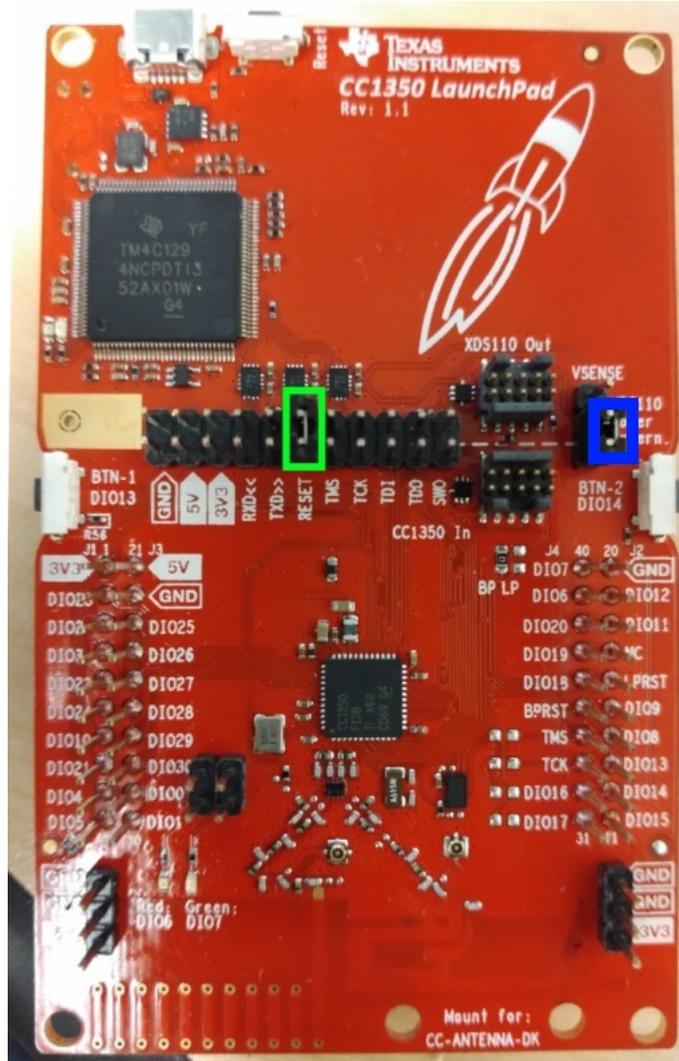


**图 26. CC1350 LaunchPad™**

2. Stack both LaunchPad development kits on top of each other as shown in 图 27.
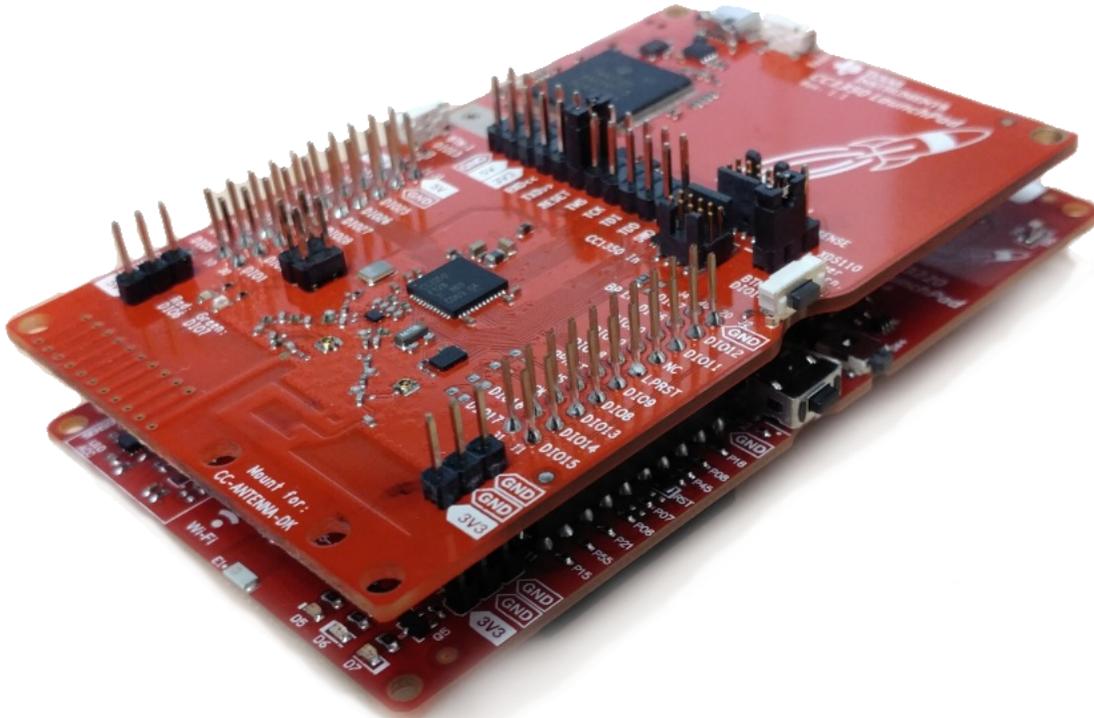


**图 27. Stacked CC1350 and CC3220SF LaunchPad™ Development Kits**

3. Connect a USB cable only to the CC3220 LaunchPad, and plug it in the PC.

4. Open a serial console (such as, PuTTy or Tera Term), select the COM port associated to the CC3220SF LaunchPad, and use the configuration below:

   Baud Rate: 115200
   Data: 8bit
   Parity: none
   Stop: 1bit
   Flow Control: none

5. Press the Reset button on the CC13x0 or CC13x2 LaunchPad attached to the CC3220SF, if the setup was successful, some debug logs should be displayed on the terminal..

### 3.2.1.1.6    *Open and Configure an IBM Cloud™ Account*

1. It is assumed that all the required software has already been installed. If not, install the required software now.

2. Go to IBM Cloud, click sign up if to create an account or log in if you already have an account.

3. Once logged in, click on the *Catalog* tab located on the top-right corner.
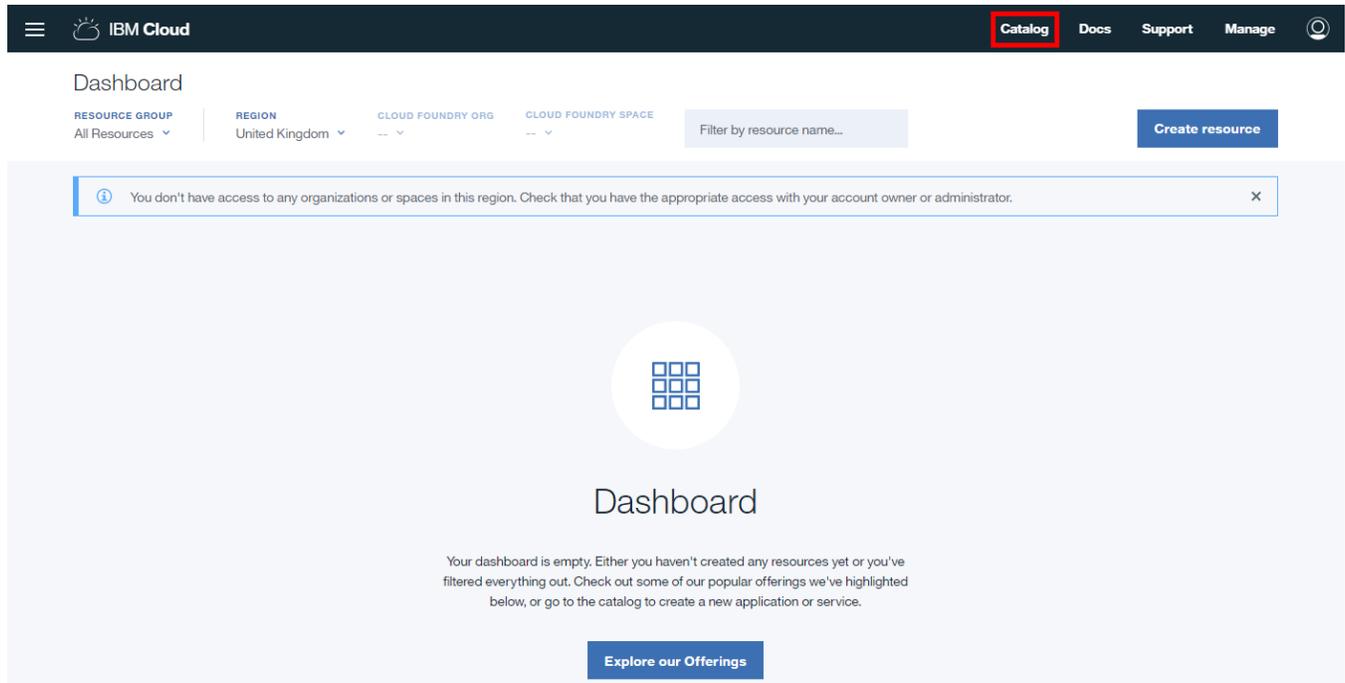


图 **28. IBM® Cloud**

4.  After selecting Catalog, click on *Cloud Foundry Apps* on the left menu.
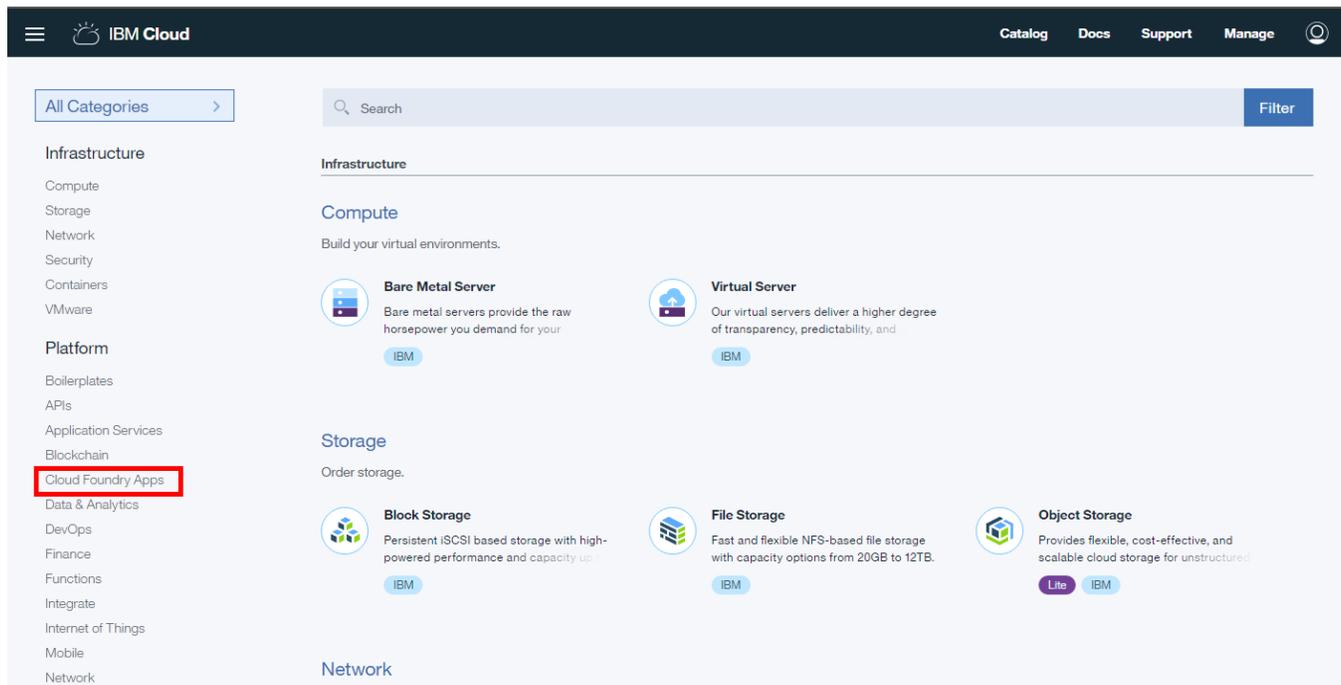


**图 29. IBM Cloud™ Catalog**

5.  Choose *SDK for Node.js* from the options provided.
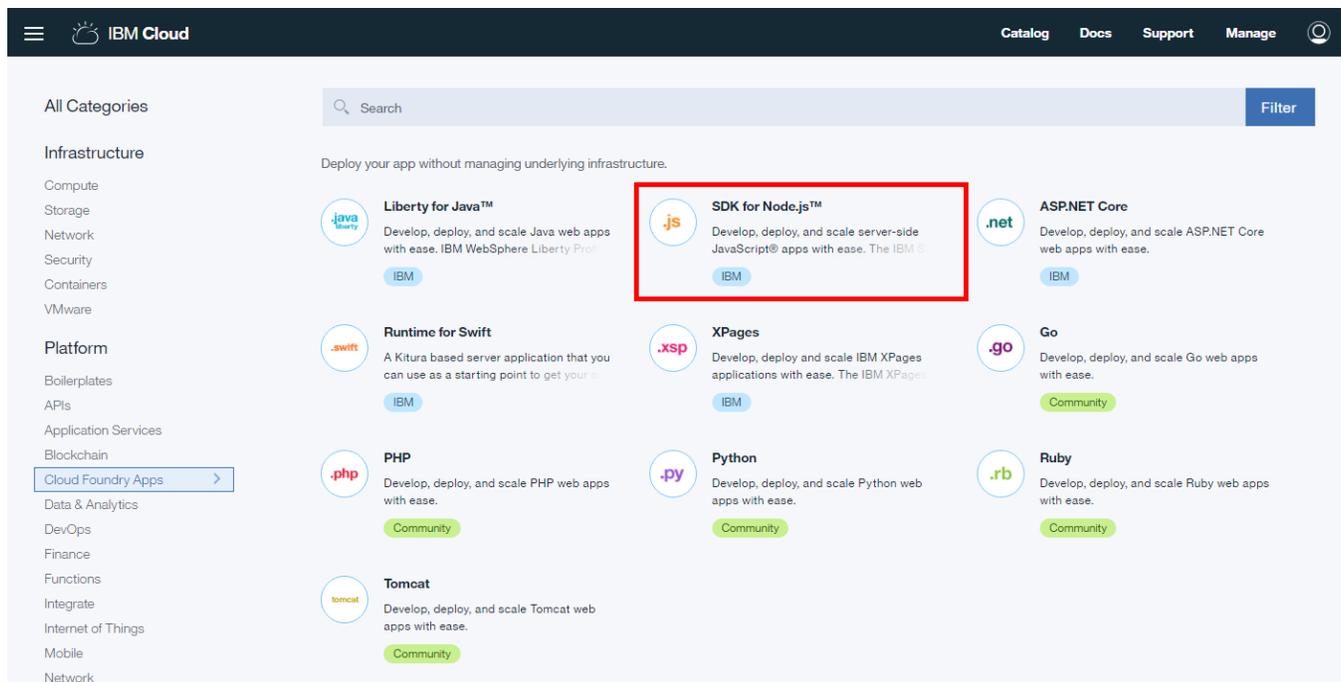


**图 30. Create Node.js Application**

6. Enter a name for the new Node.js application under the *App name* text box, and press the *Create* button on the bottom-right corner.
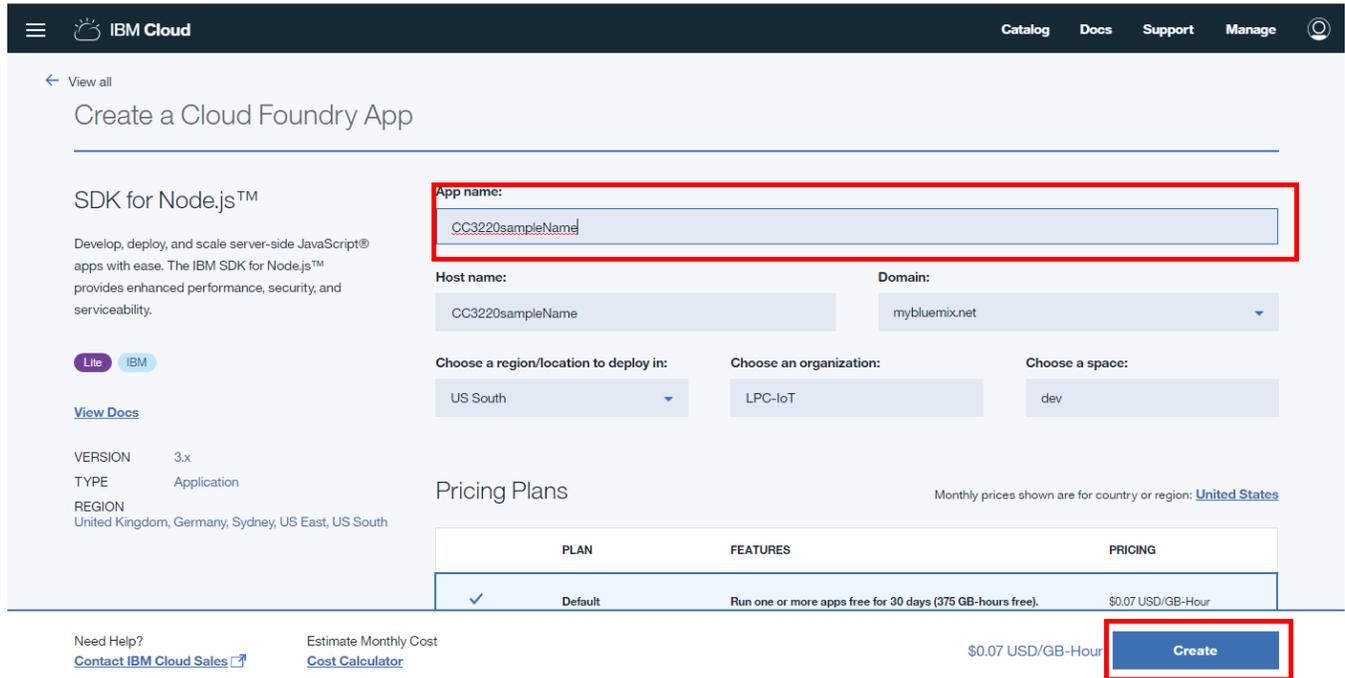


图 **31. Name and Create Application**

7. Once the app has been created, click on the *Catalog*tab located on the top-right corner.
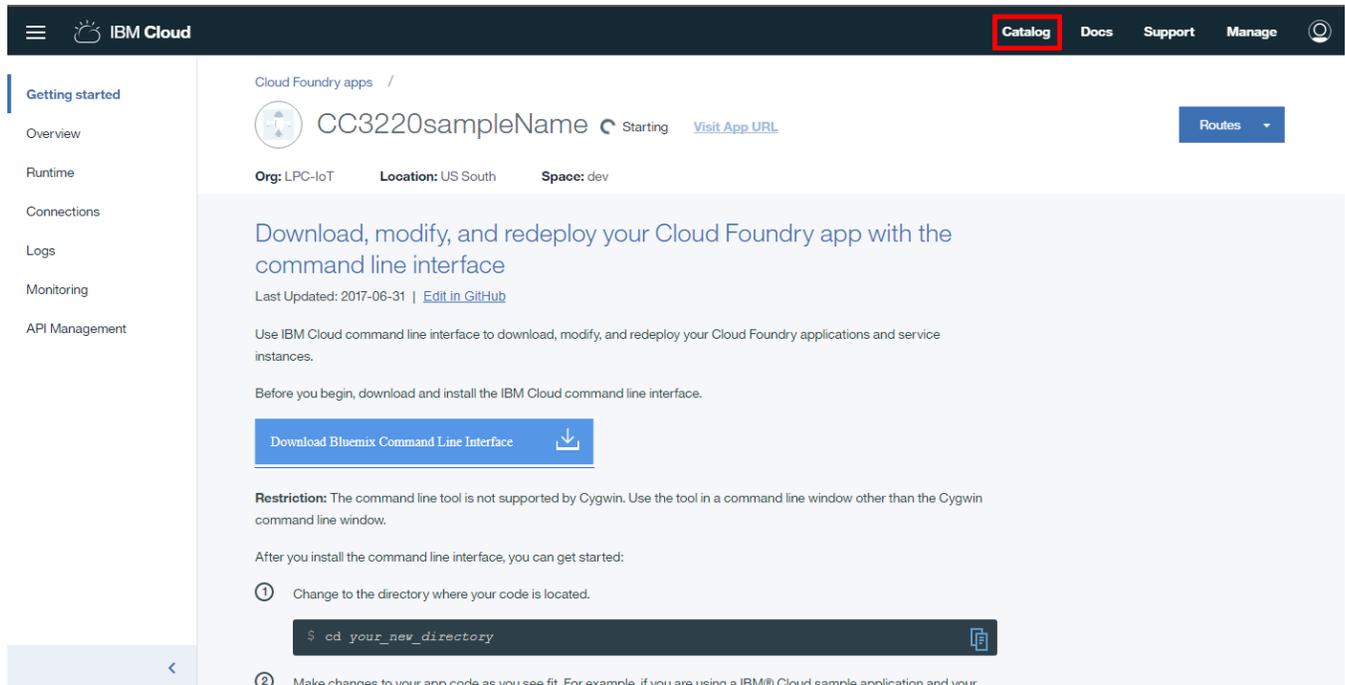


图 **32. Go to Catalog**

8.  After selecting the Catalog, click on *Internet of Things* under *Services* on the left menu
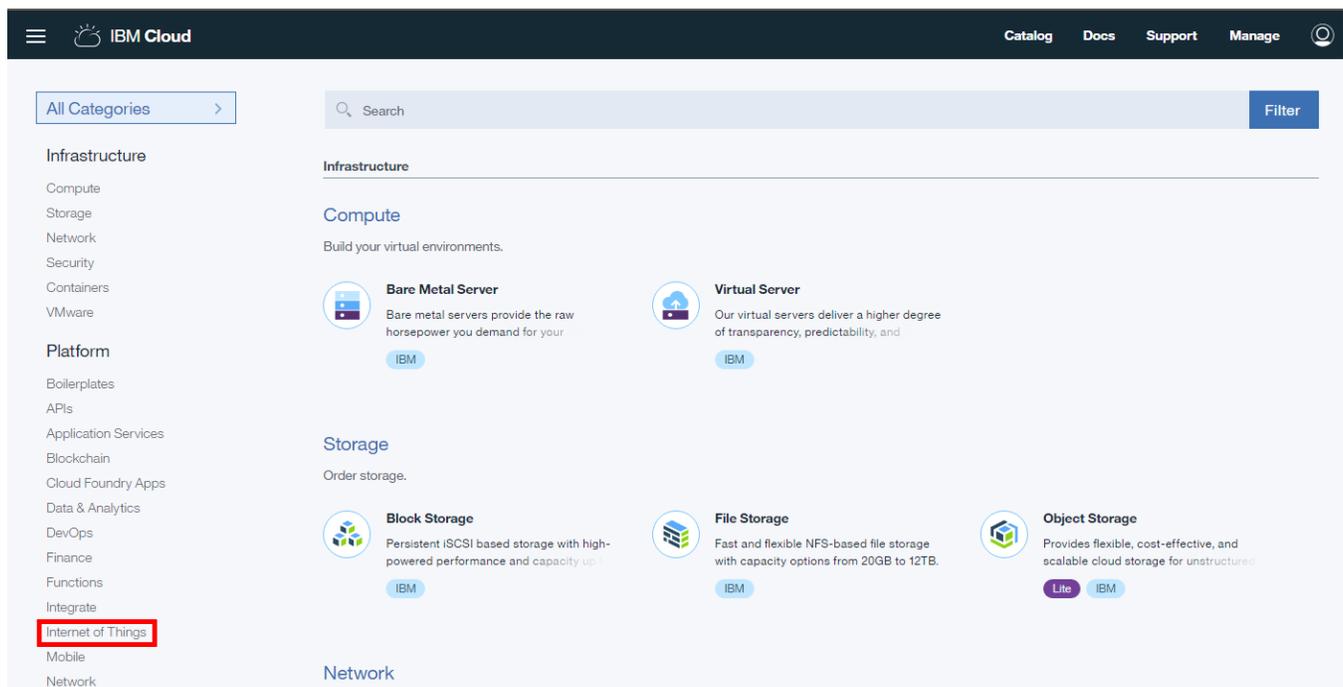


图 **33. IBM Cloud™ Catalog**

9.  Choose *Internet of Things Platform* from the options provided.
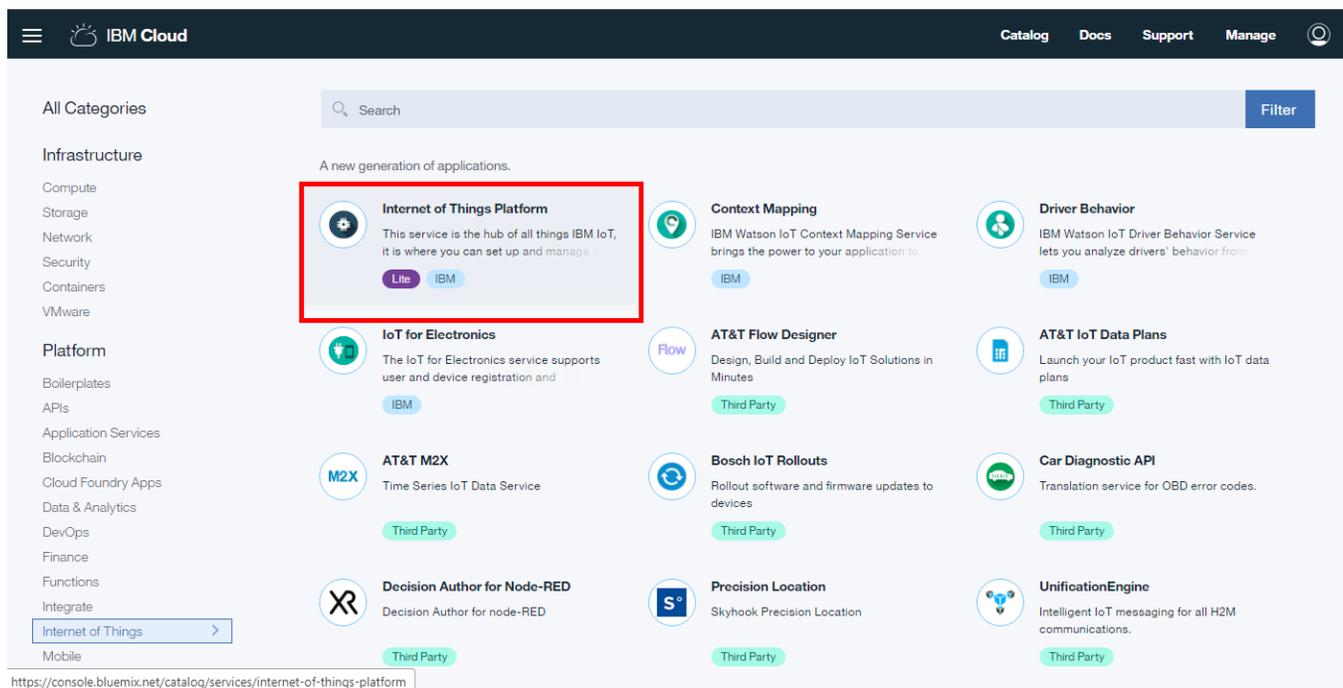


图 **34. IoT Service Options**

10. Give the new platform a service name. Click the *Create* button on the bottom-right corner.
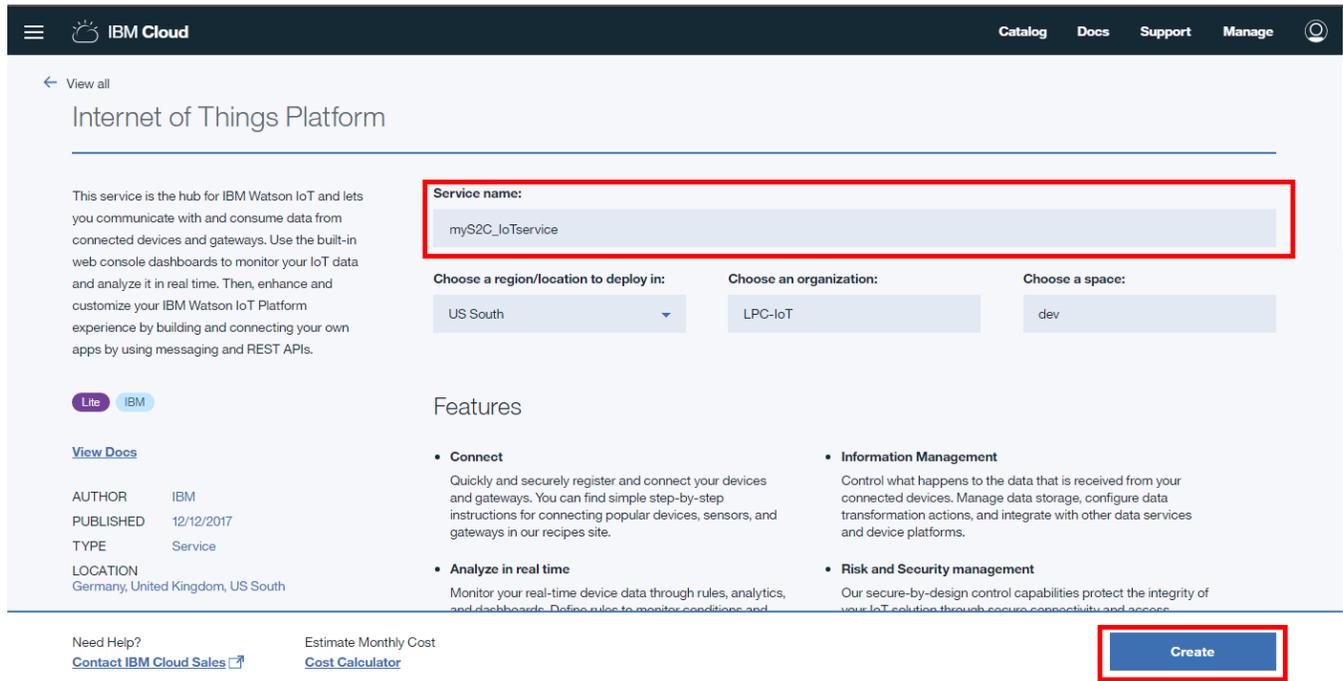


图 **35. Create IoT Platform**

11. Click on the Launch button as shown below.
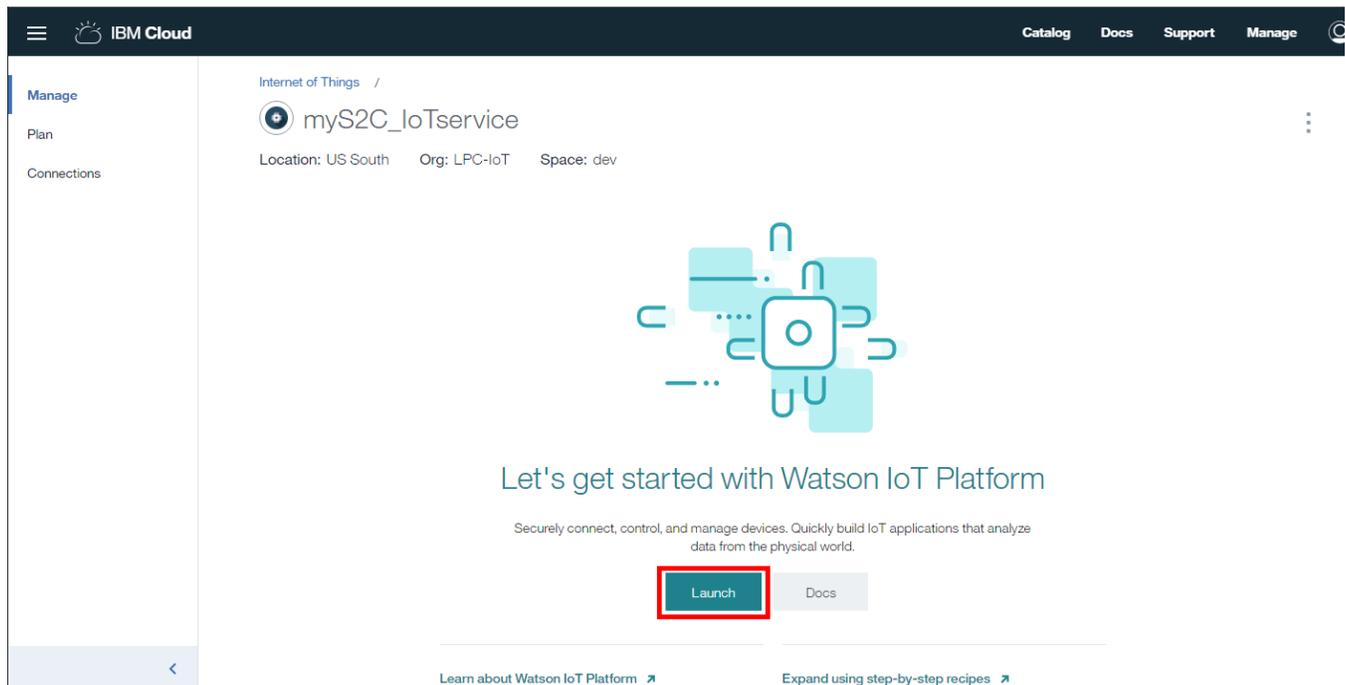


图 **36. Launch Service Screen**

### *3.2.1.1.7 Set Up Watson IoT™ Platform Service*

1. Once on the IBM Watson IoT Platform, go to Devices on the navigation bar on the left as shown in 图 37.
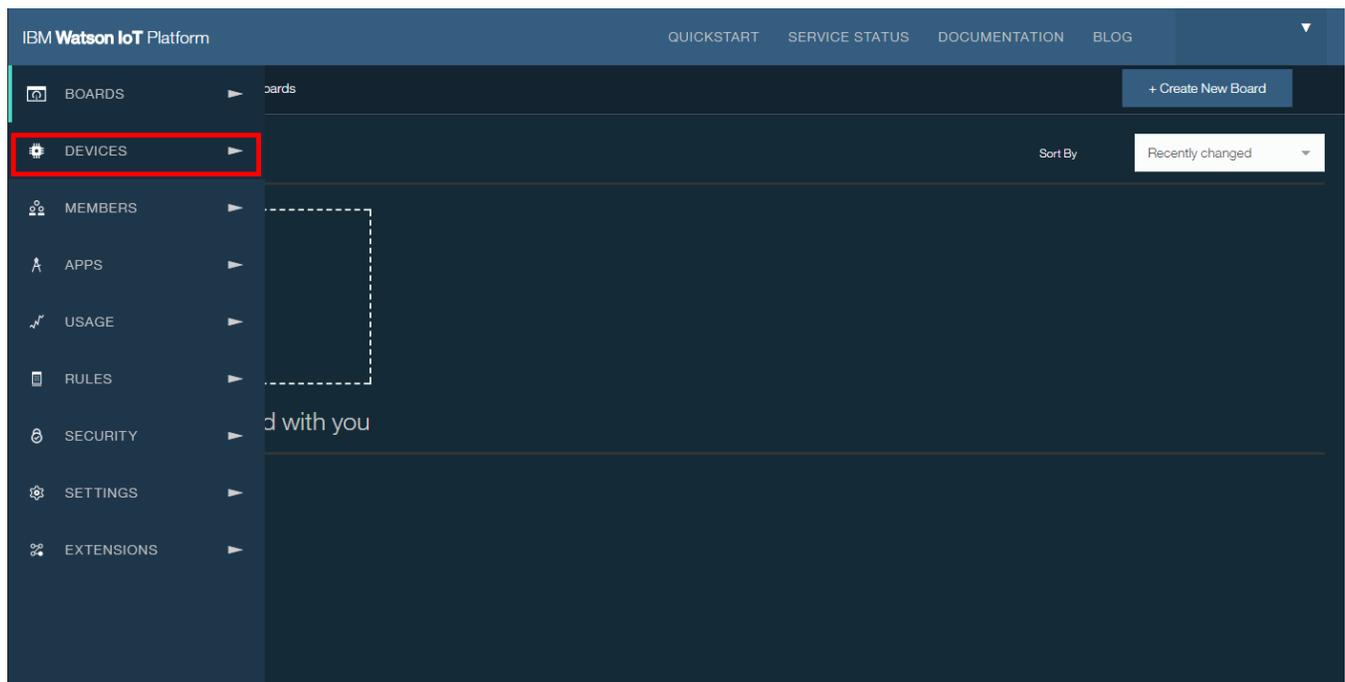


图 **37. Go to Dashboard**

2. Go to *Device Types* and Select *Add Device Type* on the top-right corner.
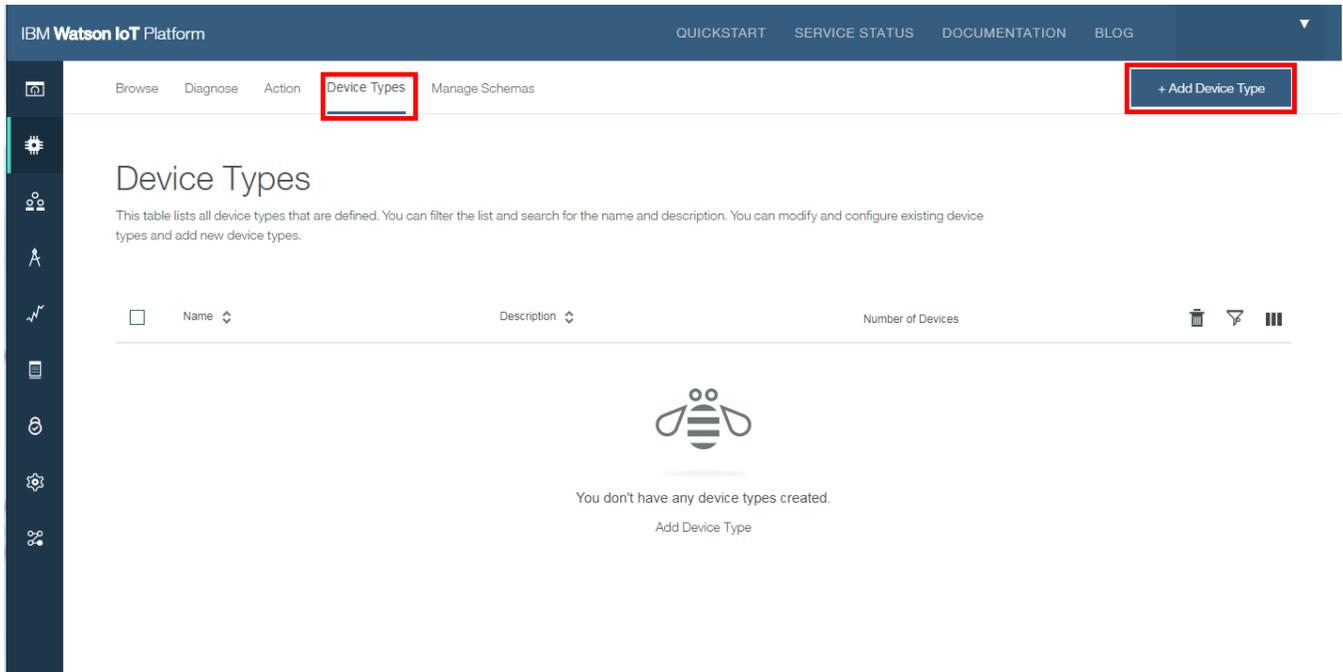
图 **38. Add Device Type**

3. Select *Gateway under Type and enter "gateway" for the name and click on Next* .
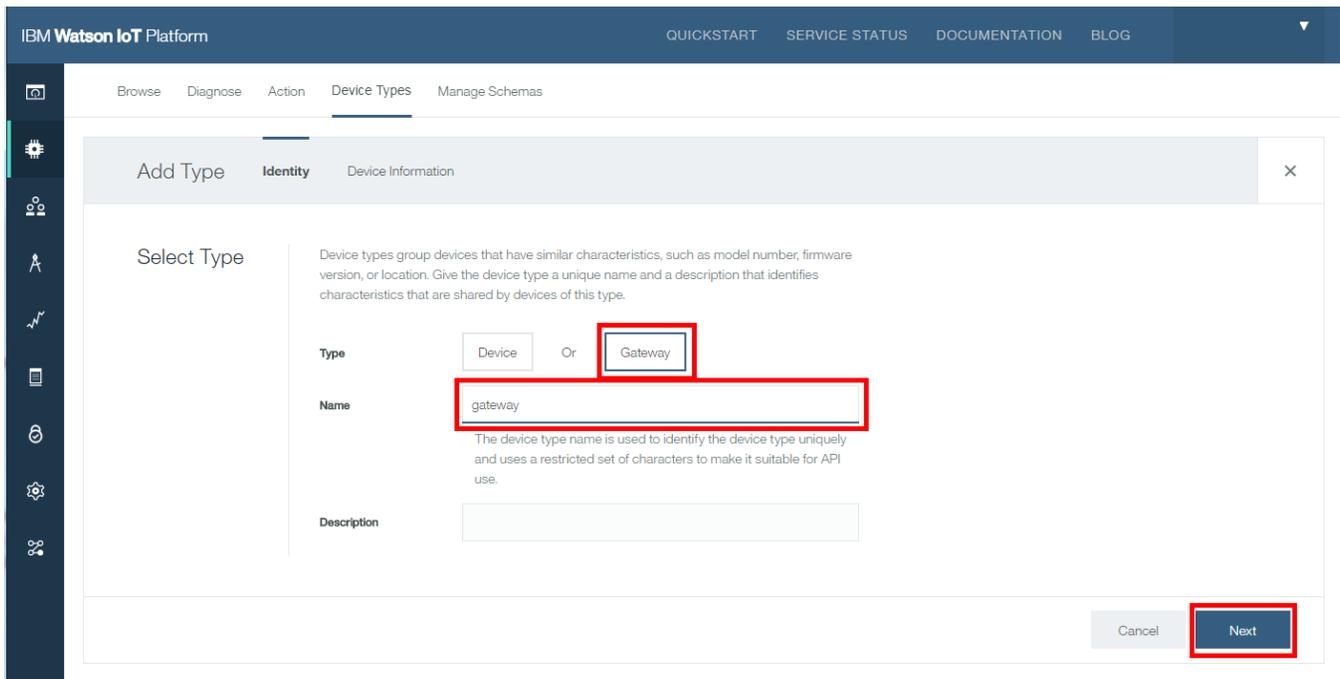


图 **39. Create Gateway Type**

4. Click on Done.
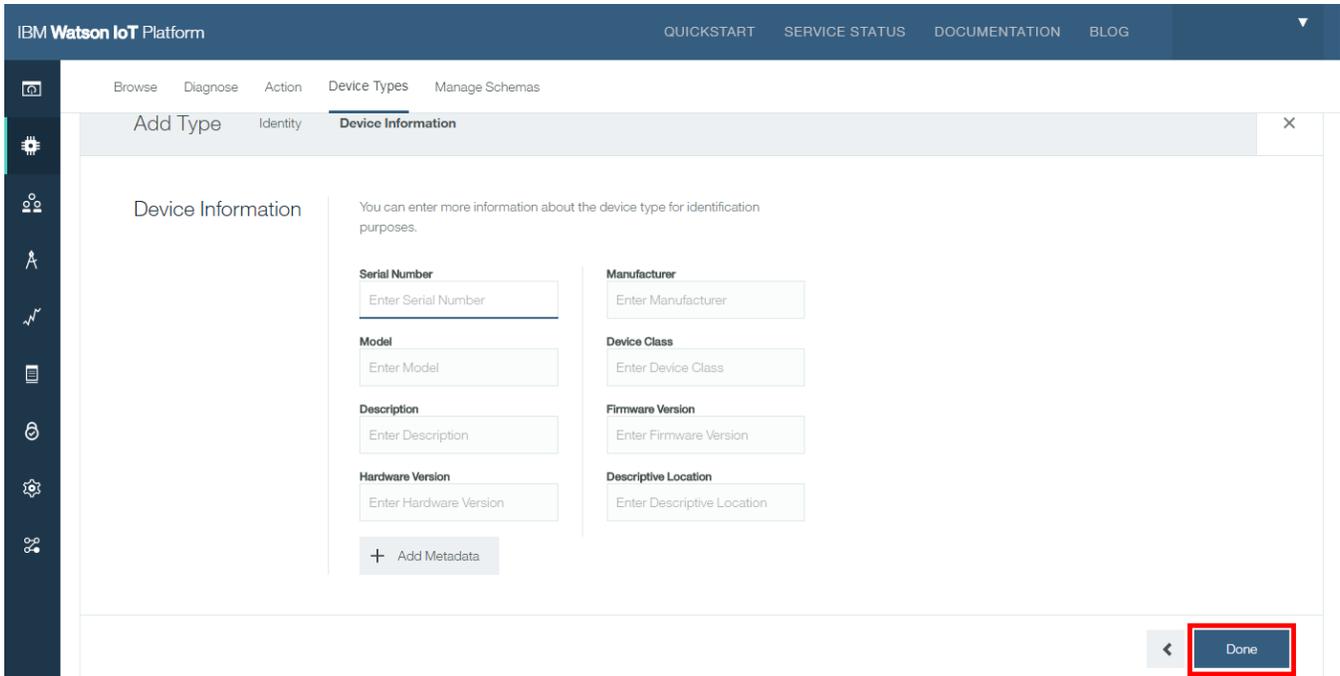


图 **40. Gateway Type Device Information**

5. Go to the Browse tab and click on Add Device.
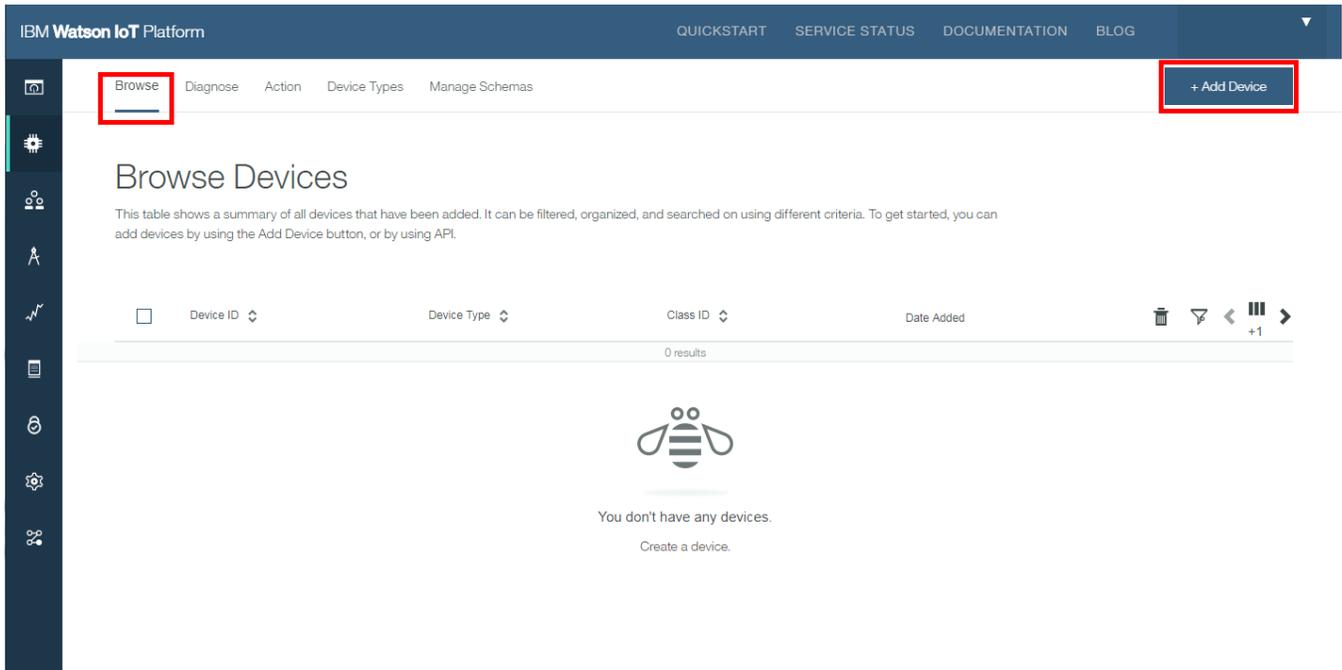


**图 41. Browsing Devices**

6. Select gateway as the device type and enter a device ID.
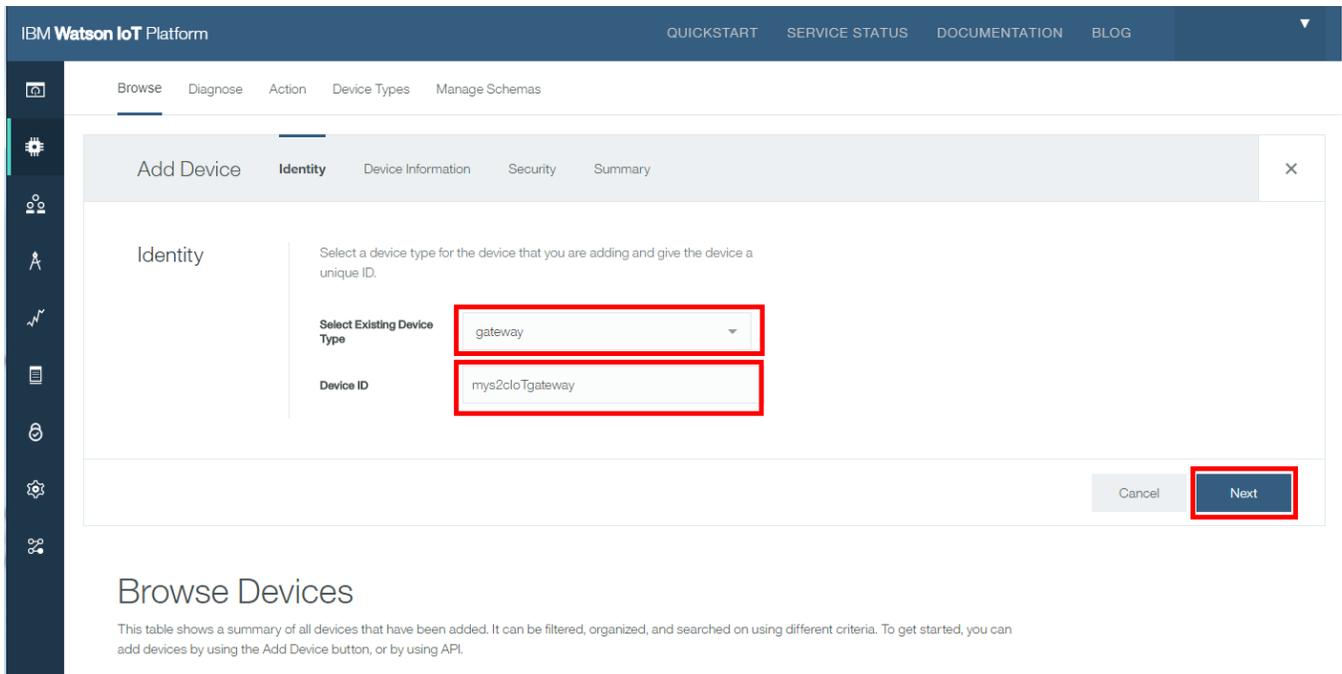


**图 42. Add Device Identity**

7.  Skip the Device Information and click next.
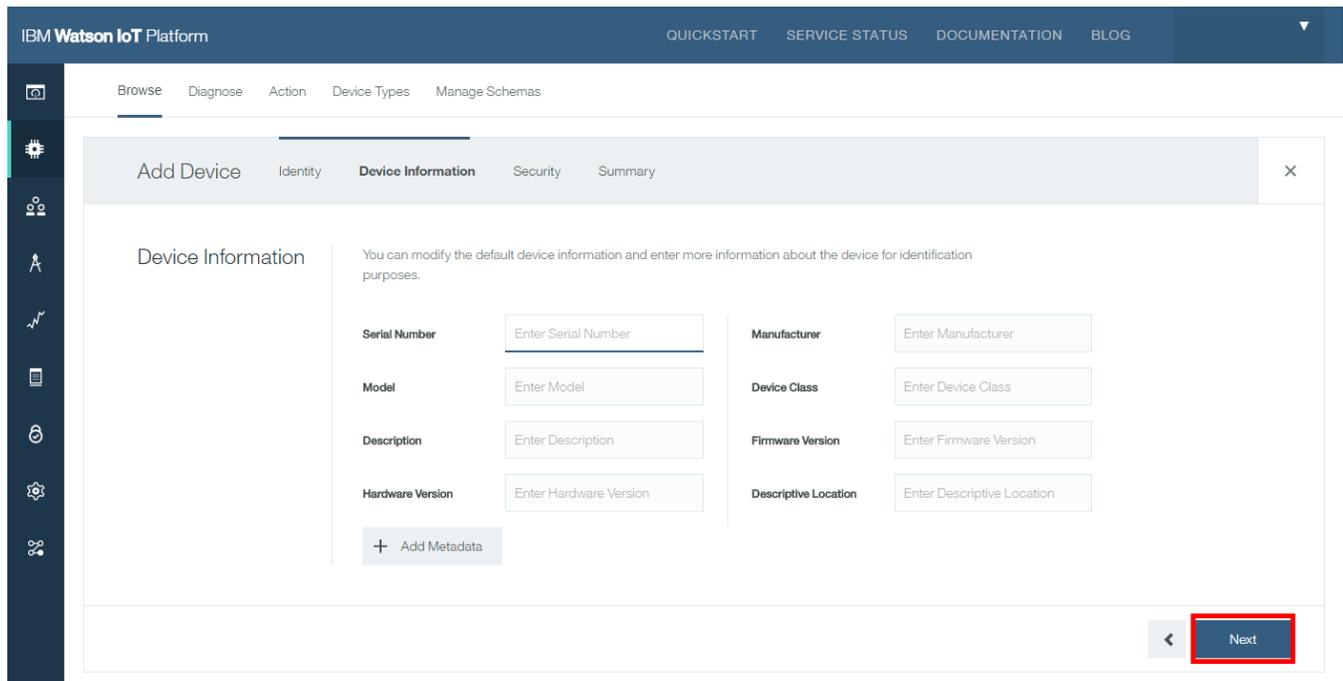


图 **43. Device ID**

8.  On the *Security* tab, fill out the *token* field. Make note of this token, as this will be used for authenticating the device to the cloud. Click *Next*.
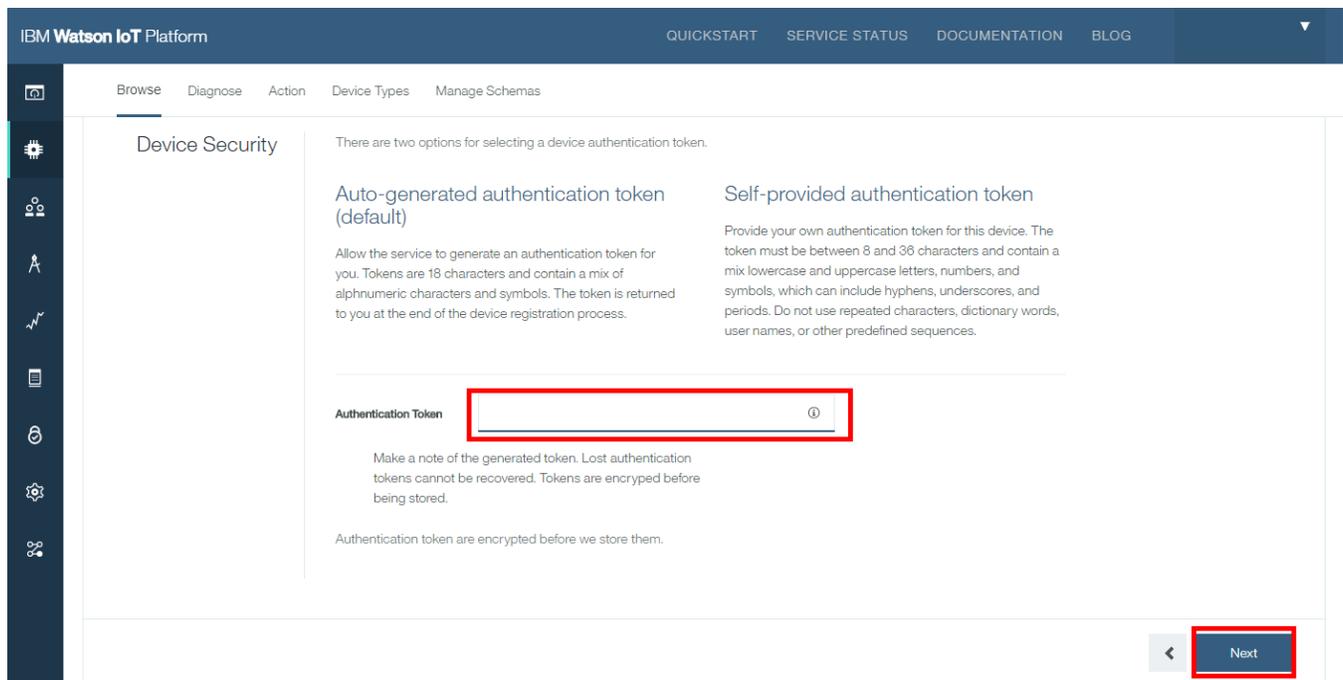


图 **44. Authentication Token**

9. Keep clicking *Next* until a summary of the device credentials and information shows. Take a screenshot of this page, as this will be the last time the *Authentication Token* is visible. Click on Done.
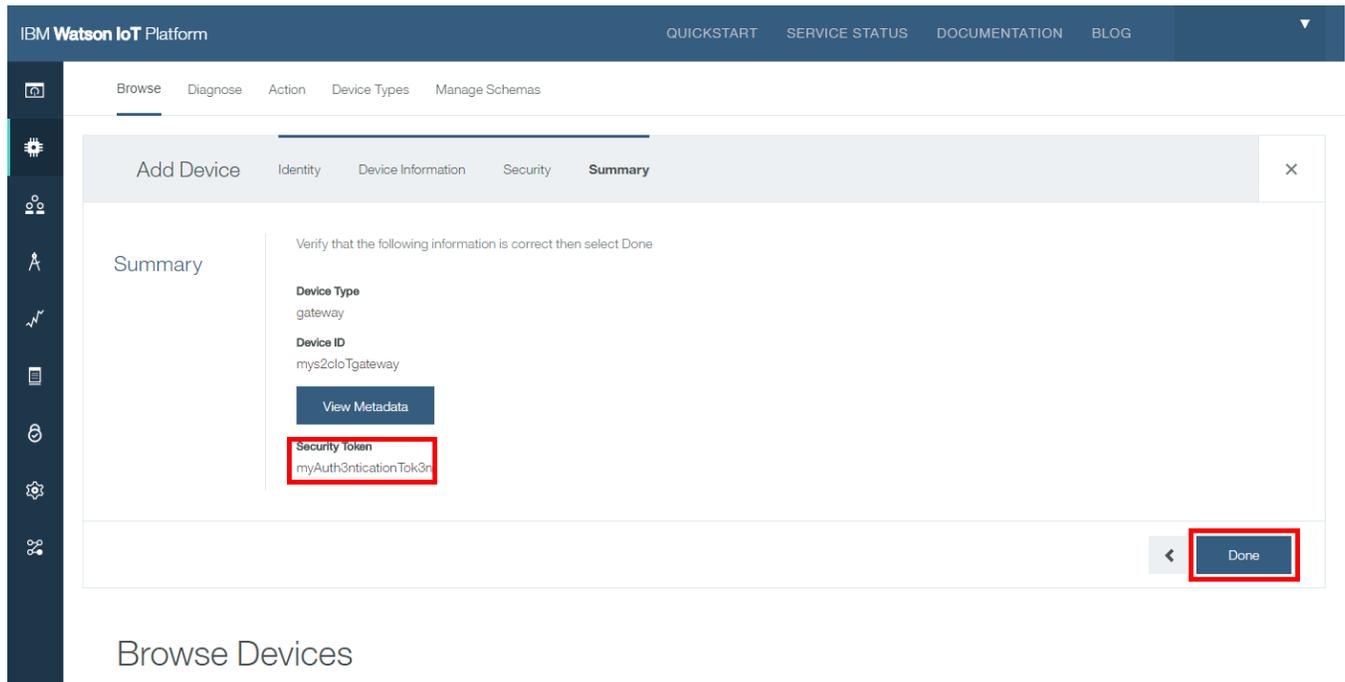


图 **45. Adding Device Summary**

10. After adding the new device, go to security on the menu on the left as shown below.



图 **46. Security Menu**

*Hardware, Software, Testing Requirements, and Test Results*
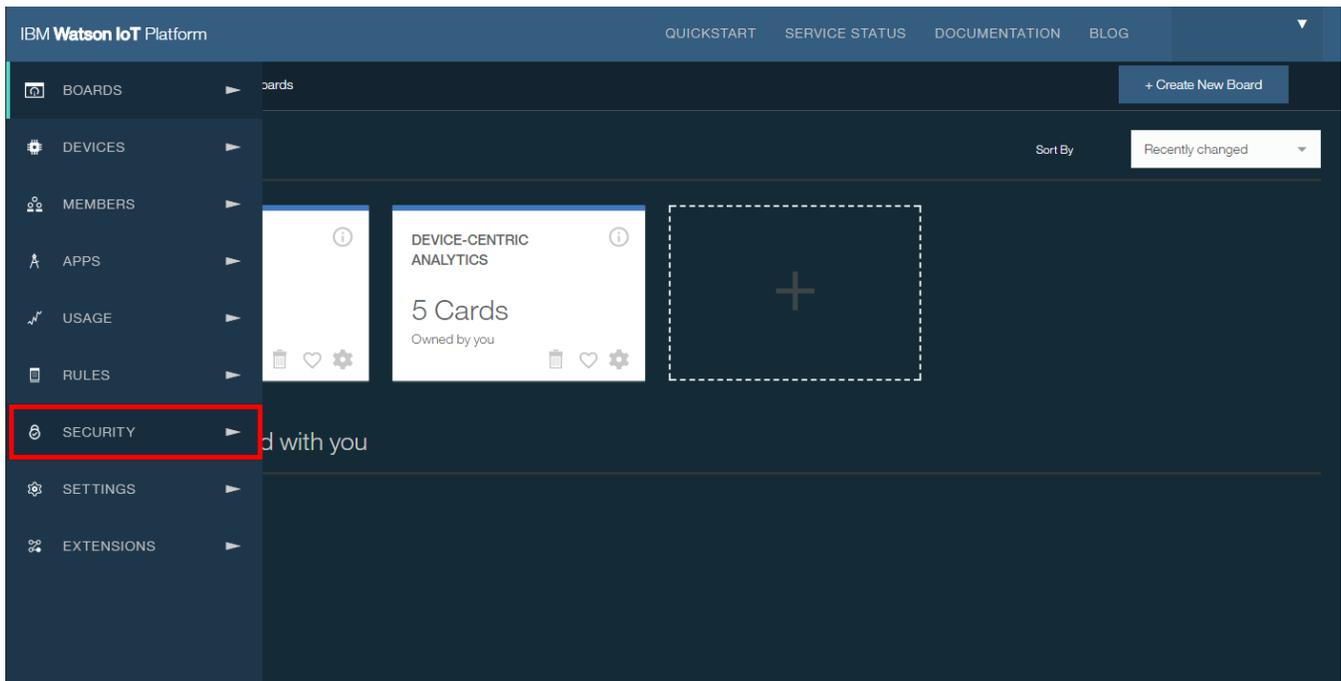
11. Click on the edit button for Connection Security.
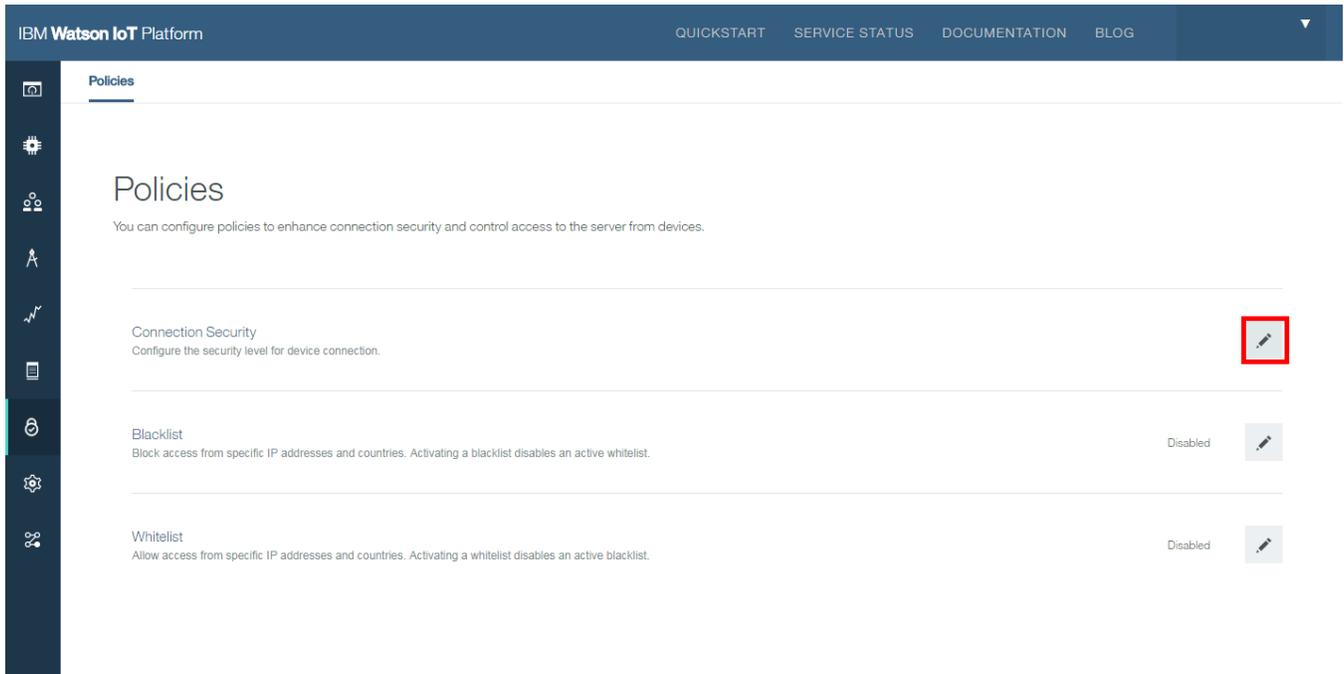


图 **47. Modifying Connection Security**

*低于 1GHz 嵌入式传感器到云工业物联网 (IoT) 网关参考设计* 49

12. Click on the drop down menu under security level and select TLS Optional, and then save the new configuration.
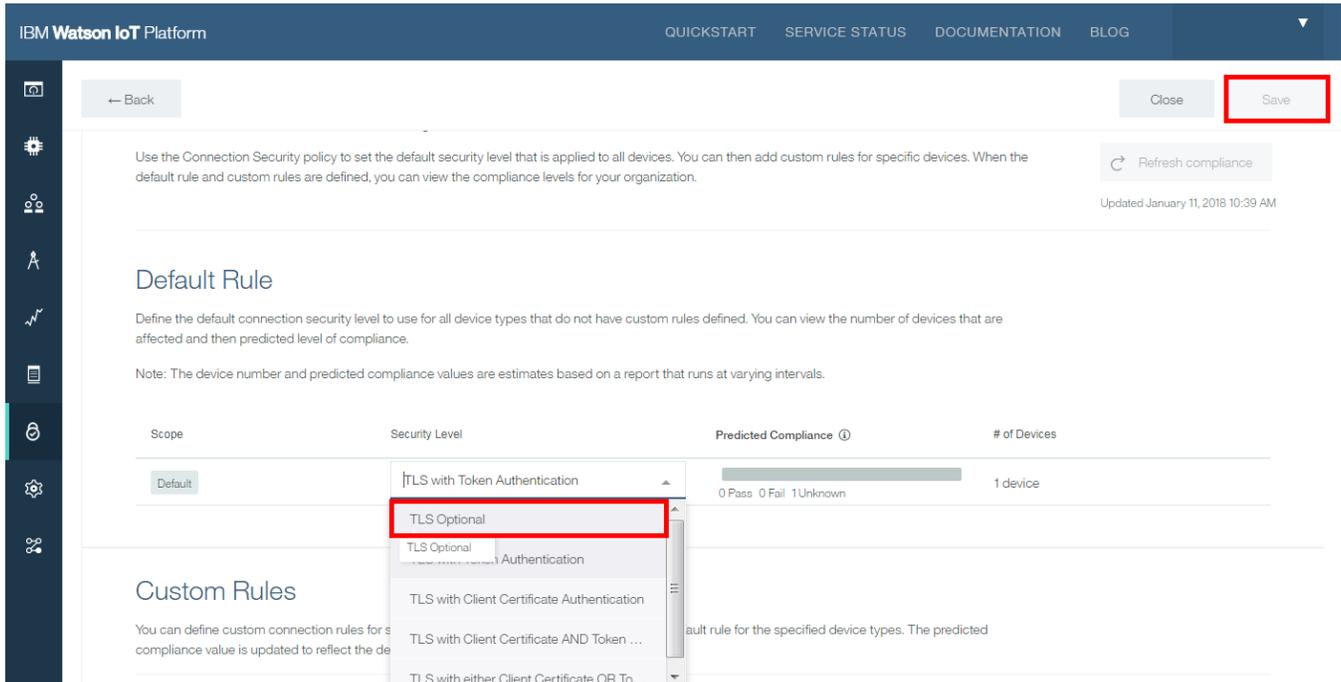


图 **48. Save New Security Rule**

#### *3.2.1.1.8　Set Up Node.js Cloud Foundry App*

1. Locate *C:\<S2C Repo Directory>\tidc01002\examples* .
2. Open the *ibm_cloud_application* folder, and open the *Manifest.yml* with a text editor. Replace the name and *services* fields with the *name* of the cloud foundry app and Watson IoT platform.
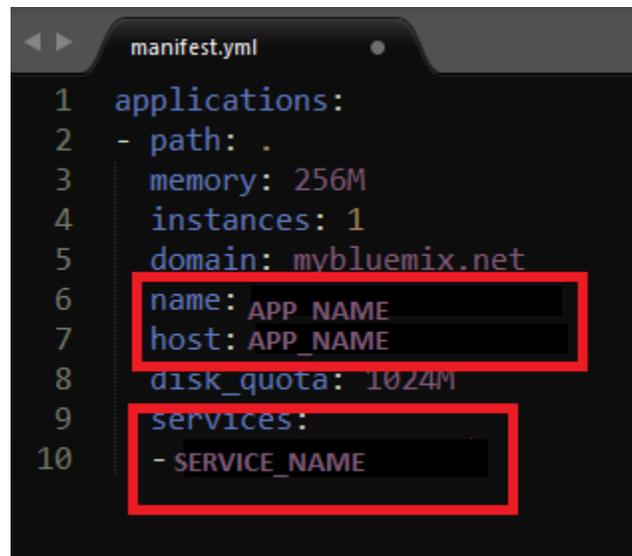


图 **49. Manifest**

3. Open a command console and navigate to the IBM-Cloud-Dashboard folder (*cd C:\<S2C Repo Directory>\tidc01002\examples\ibm_cloud_application*).

4. Type in **cf api https://api.ng.bluemix.net**

5. Log in to the created account: **cf login**

6. Push the code to the IBM cloud foundry app: **cf push**

7. Go back to the IBM Cloud *Dashboard* by clicking *IBM Cloud* on the top left.
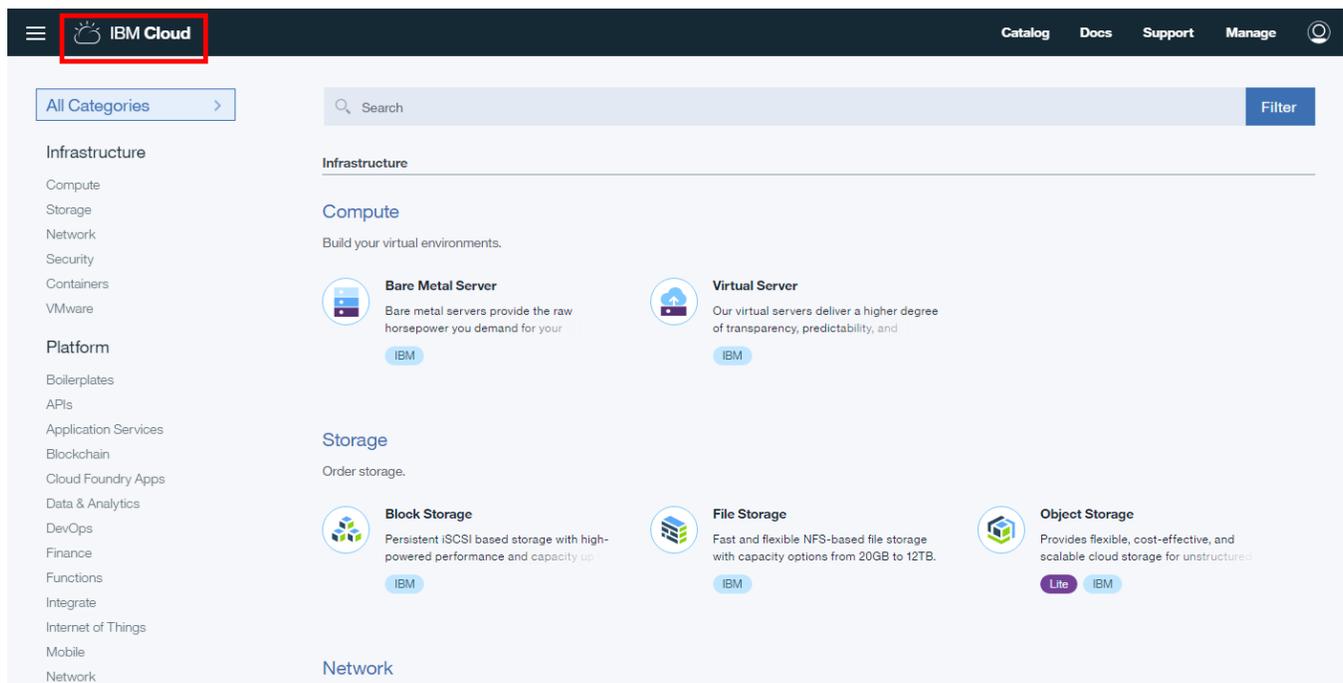


图 **50. Dashboard**

8. The cloud foundry apps and IoT services created on the previous on the dashboard will be visible. Click on the cloud foundry app.
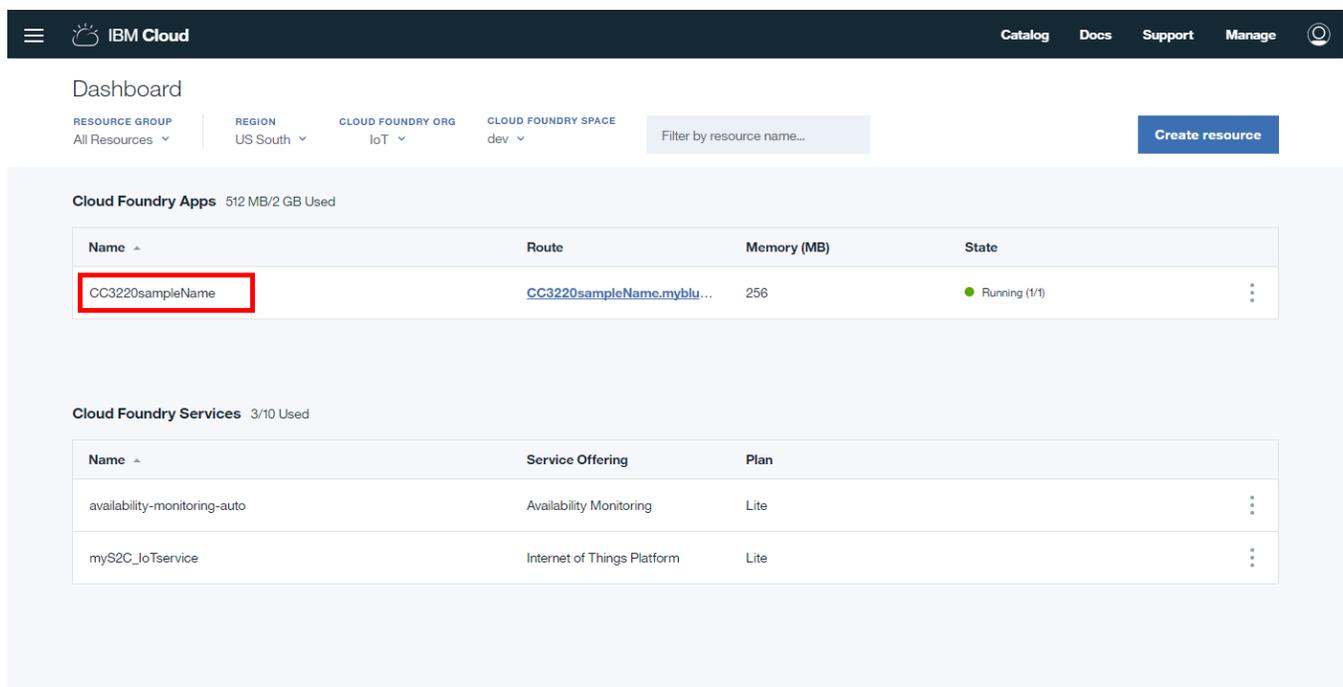


图 **51. Cloud Foundry App**

9. Click on the *Connections* tab to the left.
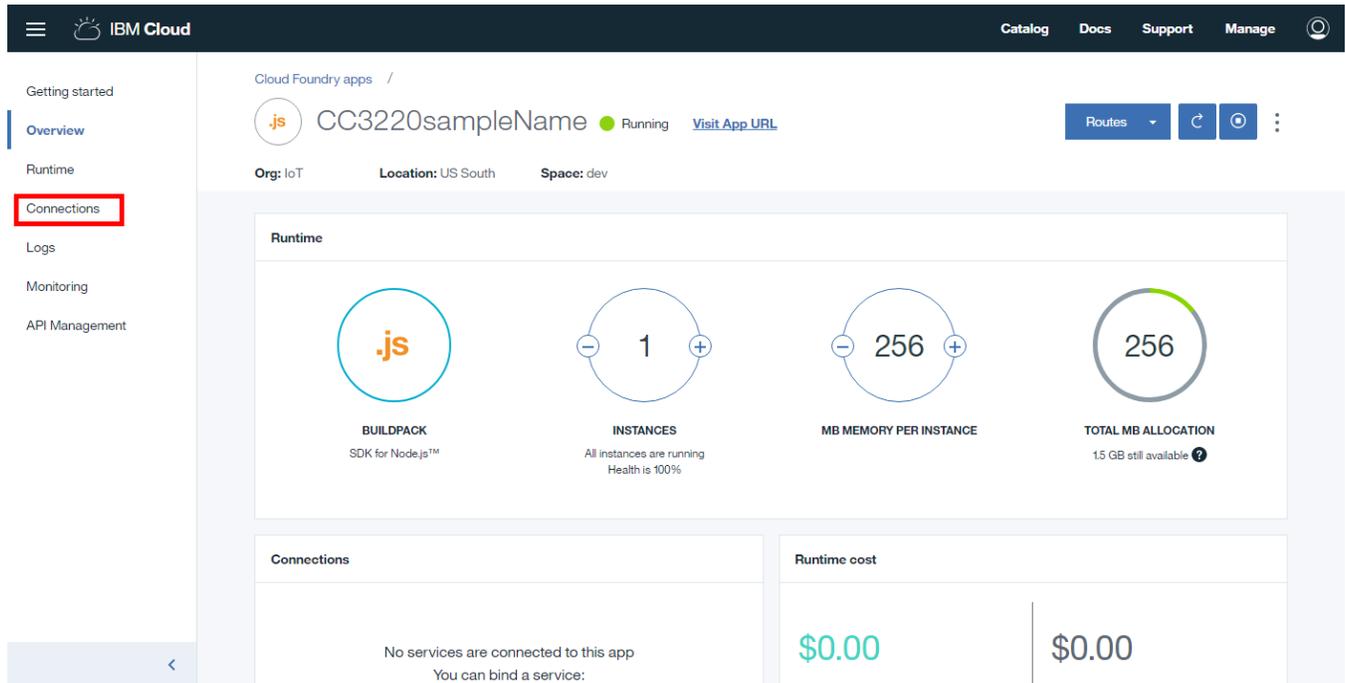


图 **52. Connections**

10. Click on the *View credentials* button. This page shows the information required to establish a connection between the cloud front end and the back end server. Screenshot or save the information for later.
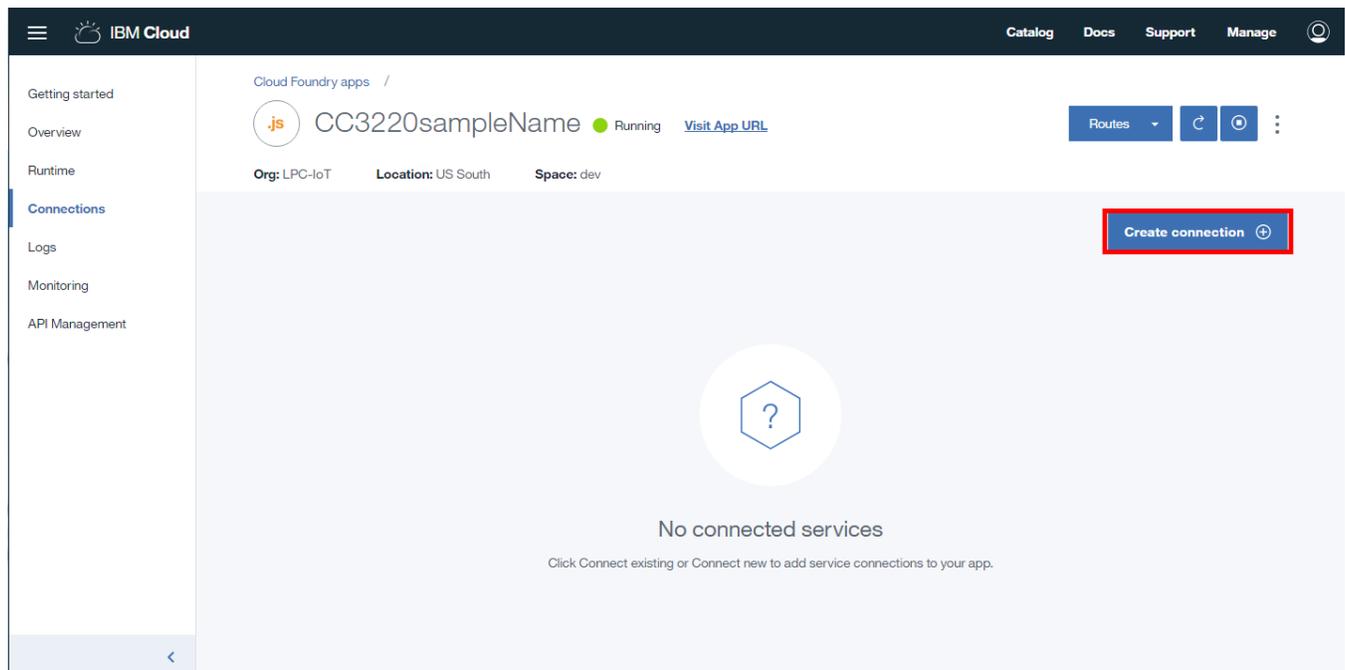


图 **53. View Credentials**

11. Select the service previously created in this guide and click the connect button.
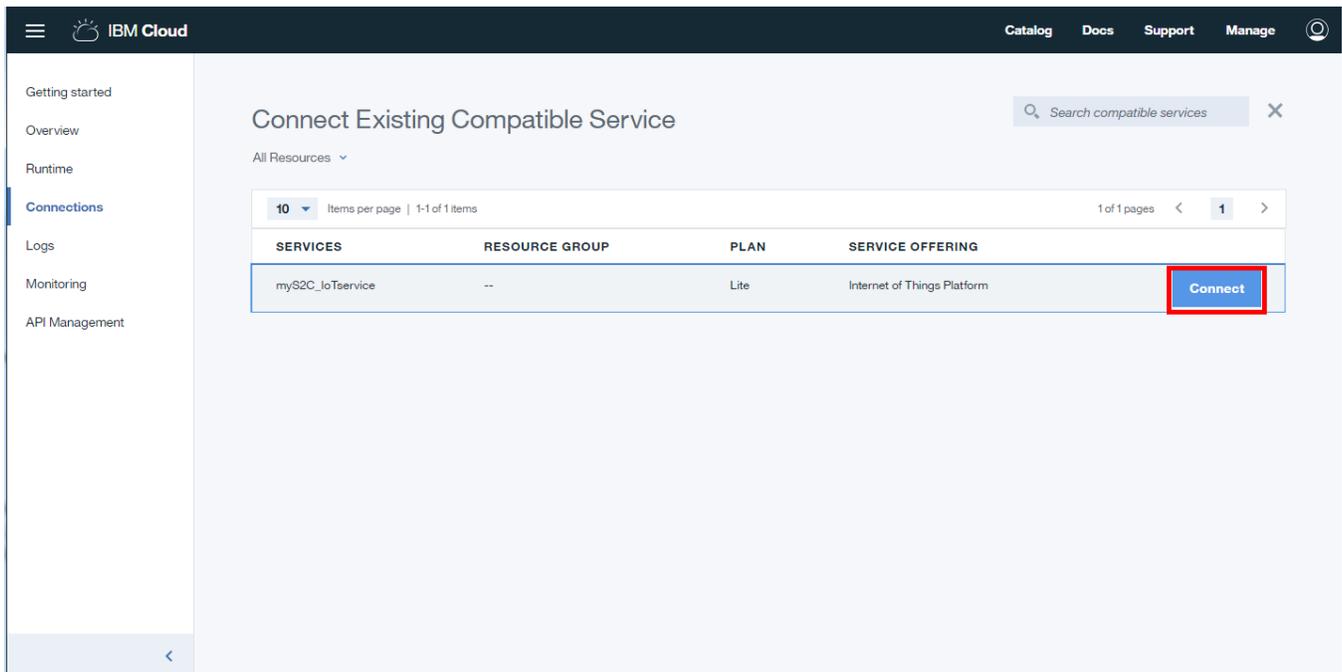


图 **54. Connect Service**

12. Restage the app.
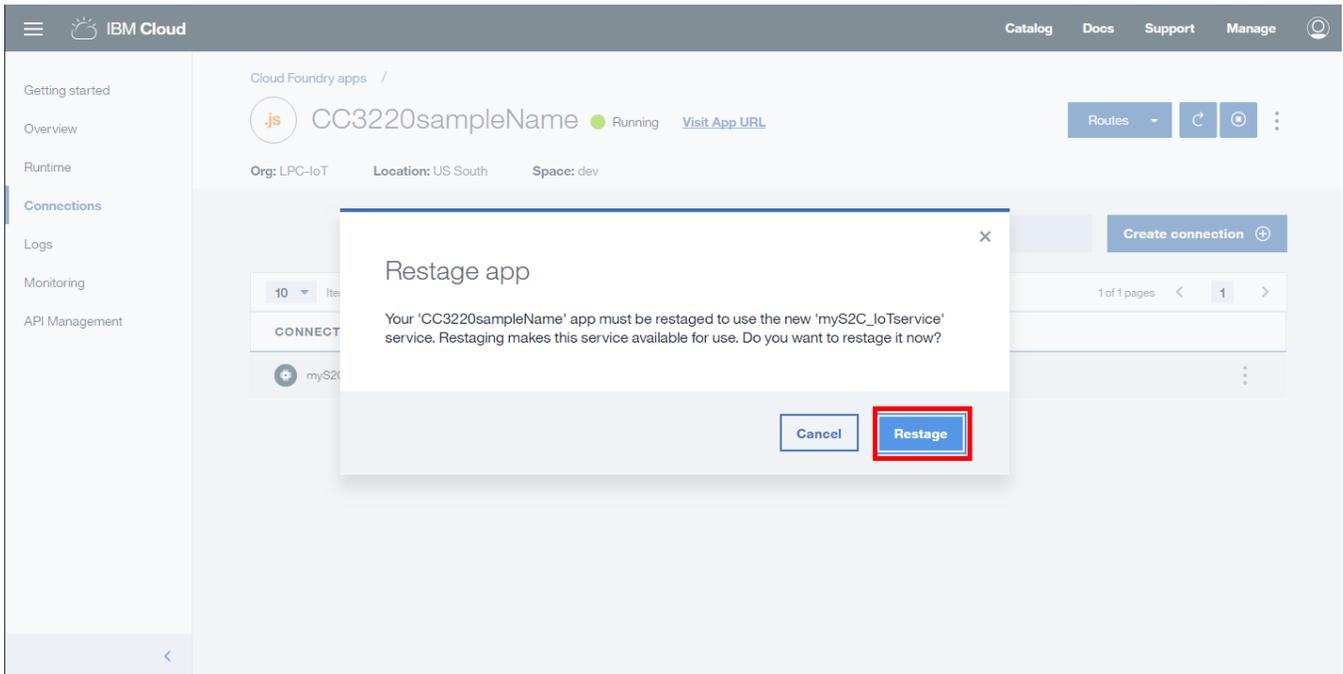


图 55. Restage the App

13. Click on the right side of the service and select View Credentials.
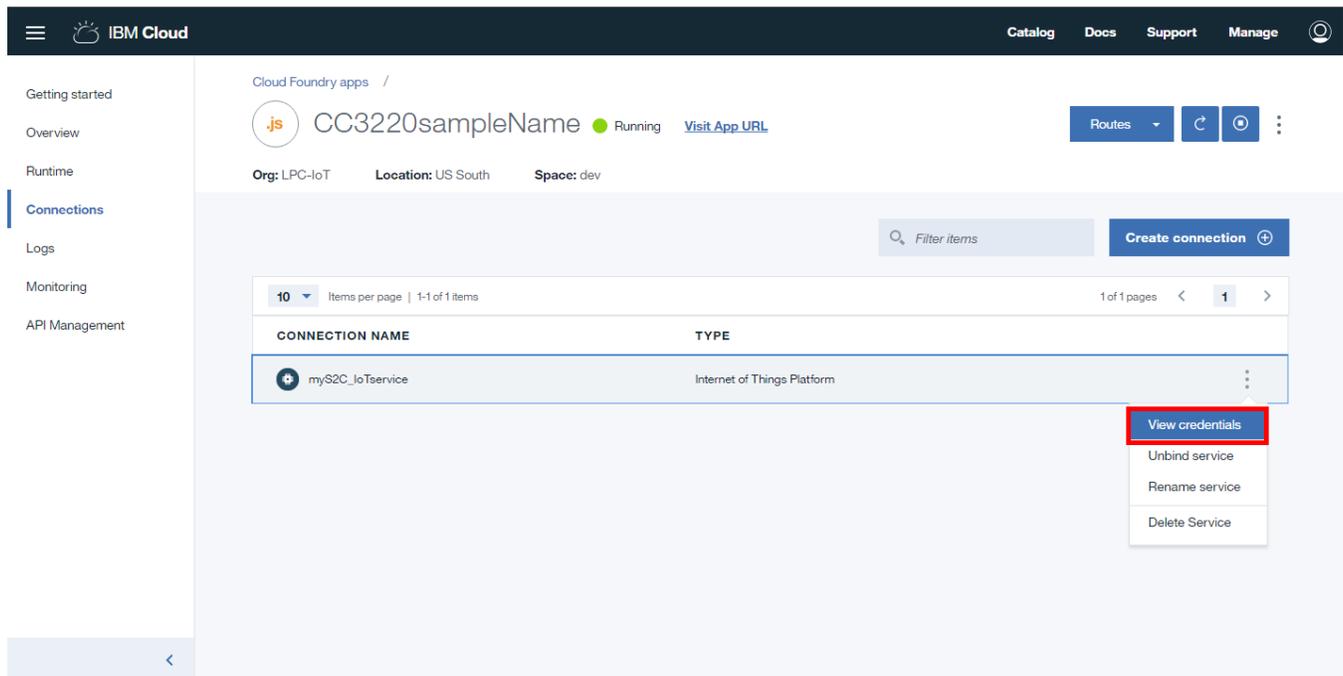


图 **56. View Service Credentials**

14. Take note of the Service credentials by clicking on the copy button and paste them into a notepad since they will be used in the next steps.
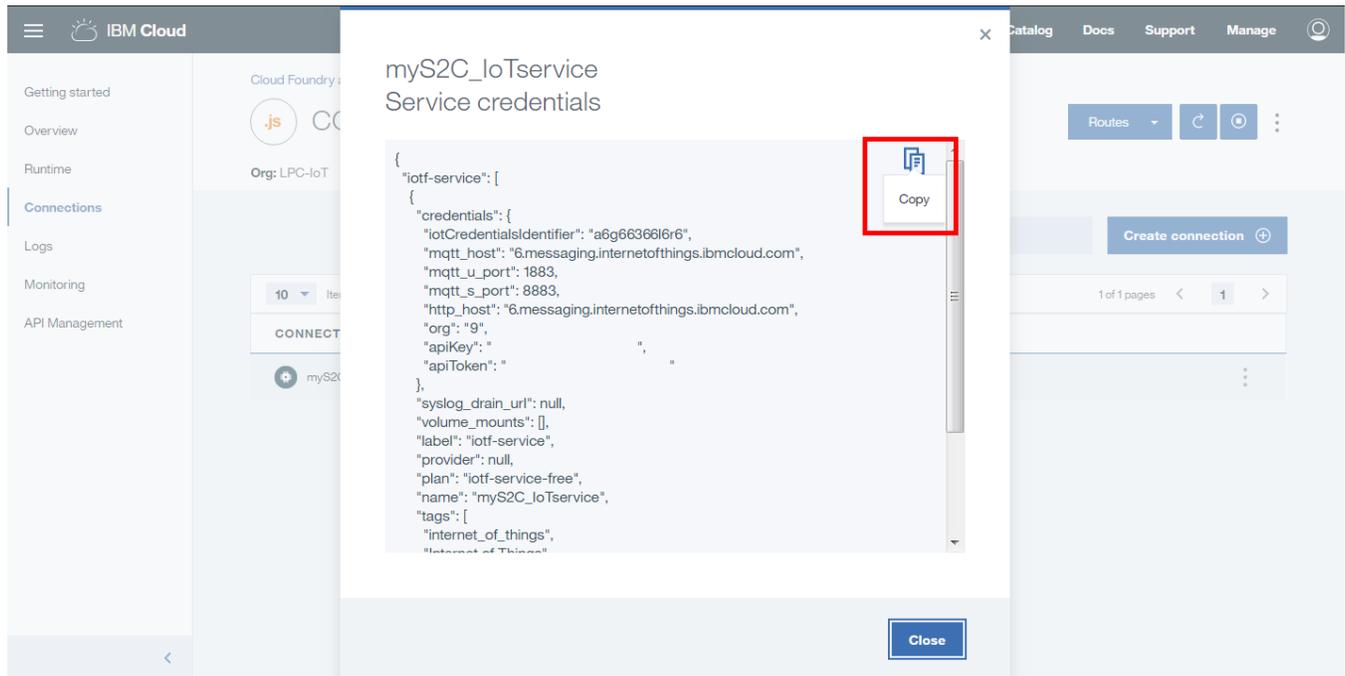


图 **57. Copy Service credentials**

15. Now go back to the Dashboard and open the webpagefor the Cloud Foundry App.The link can be found next to the cloud foundry app name.
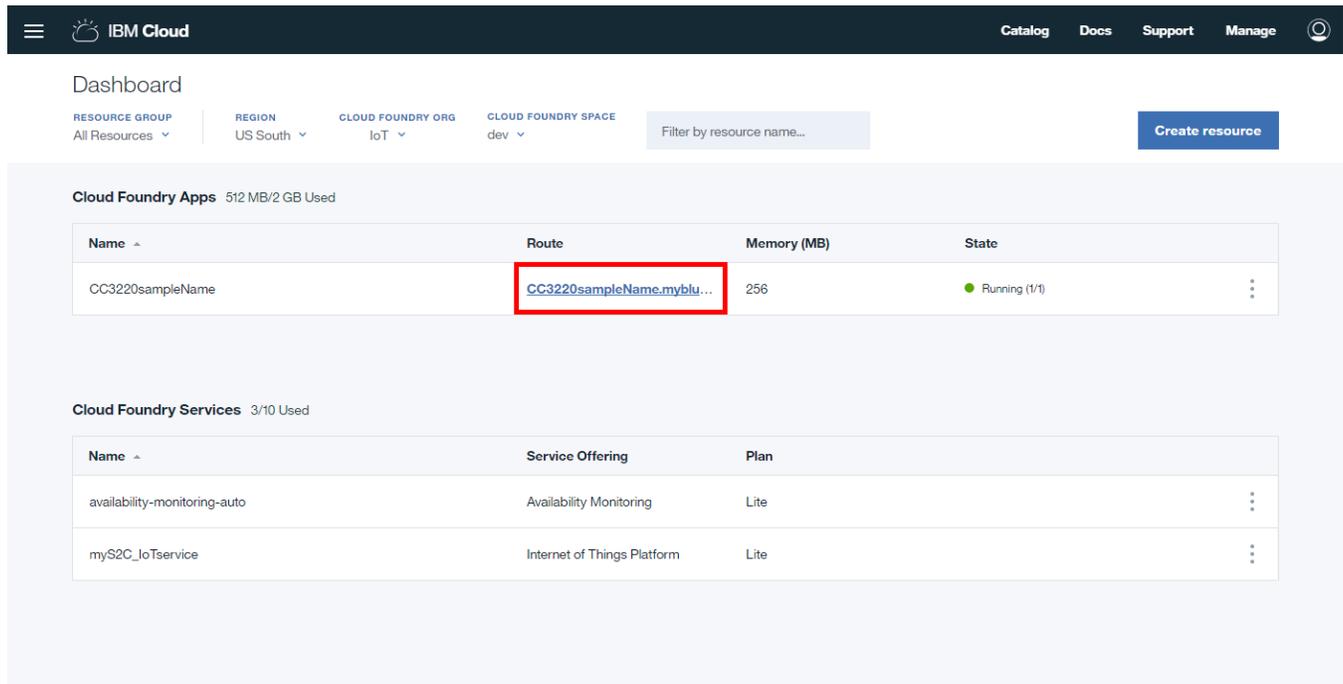


图 58. IBM® Web Page Link

16. If everything setup correctly, the dashboard will be visible. At this point, open the configuration menu located at the top.
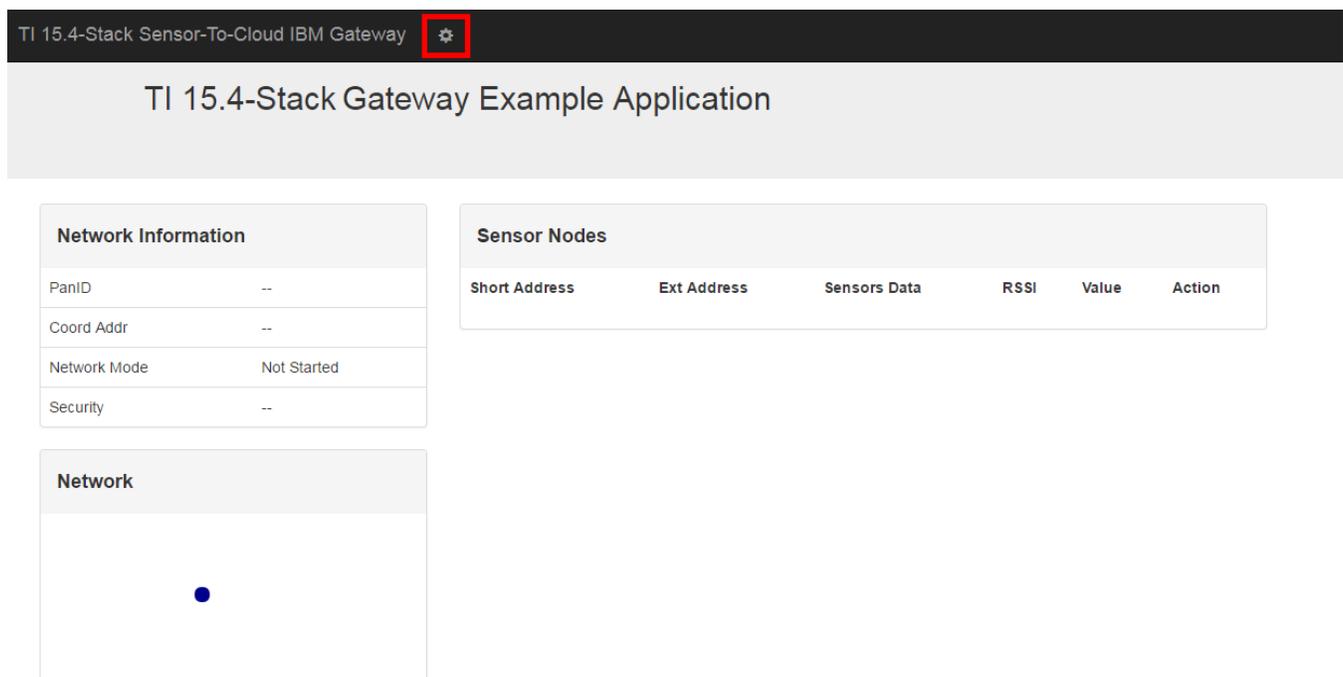


图 59. Sensor2Cloud Front End

17. A form will pop up. Use the information saved in step 15 to fill out the form. For the *Device Type* and *Device ID*, use the information entered in 图 60. Save the changes, and close when done.



图 **60. IBM® IoT Credentials**

### 3.2.1.1.9    Run and Use the Gateway

Before getting started with the instructions on how to run the gateway, make sure that the CC3220SF is plugged into the PC and that a serial console has been oppened on the serial port assigned to the CC3220SF LaunchPad.

There are two ways to get the S2C Gateway up an running. The first method is described in 节 3.2.1.1.9.1, which explains how to provision the CC3220SF LaunchPad to a WiFi Network from the Simple Link Starter Pro App. The second method is explained in 节 3.2.1.1.9.2; this method uses the built-in, local provisioning web page.

### 3.2.1.1.9.1   Using the SimpleLink™ Starter Pro App

This section assumes that either the IOS or the Android app is already installed on the user's mobile phone. If not, install the app now.

1.  Launch the SimpleLink Starter Pro App from the phone.
2.  If the device is not found automatically by the app, go to *Device to configure*, and tap on *search for your device.*



图 **61. Configuration Page**

3.  Wait for the app to find the device to connect. The name should be something like *mysimplelink-XXXX.*

4. Select the device to connect, and tap *OK*.
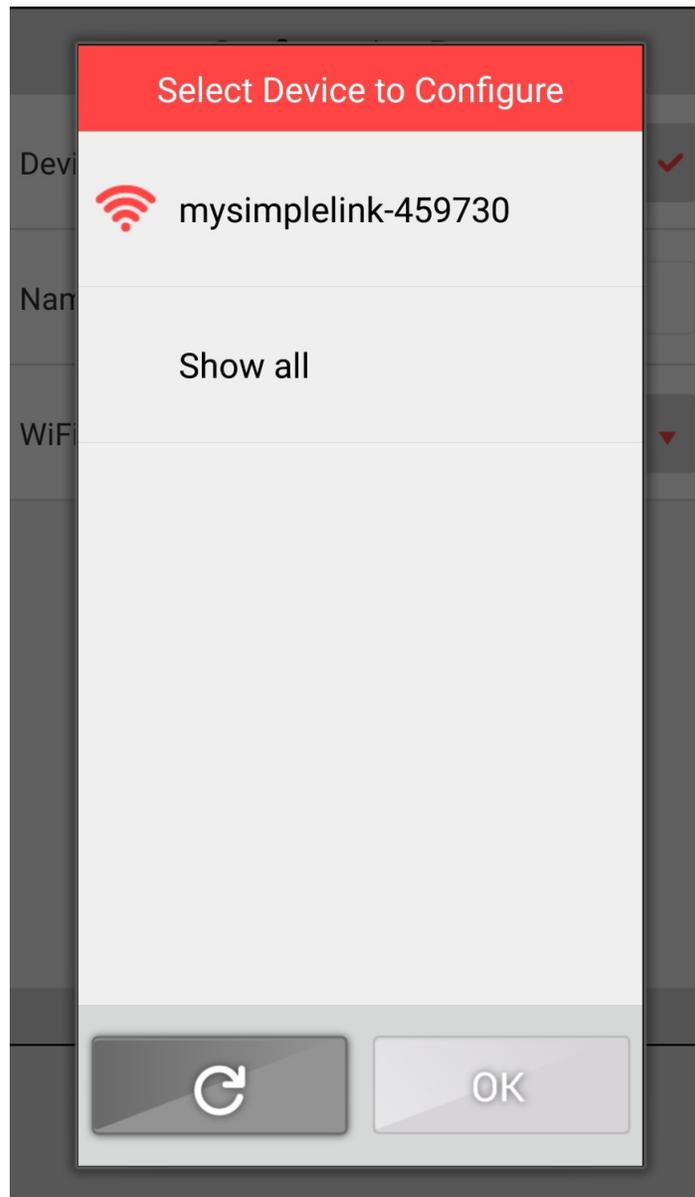


图 **62. Select Device to Configure**

5. Select the desired Wi-Fi network to connect with the CC3220SF LaunchPad. Enter the password, tap *OK*, and then tap *Start Configuration*.
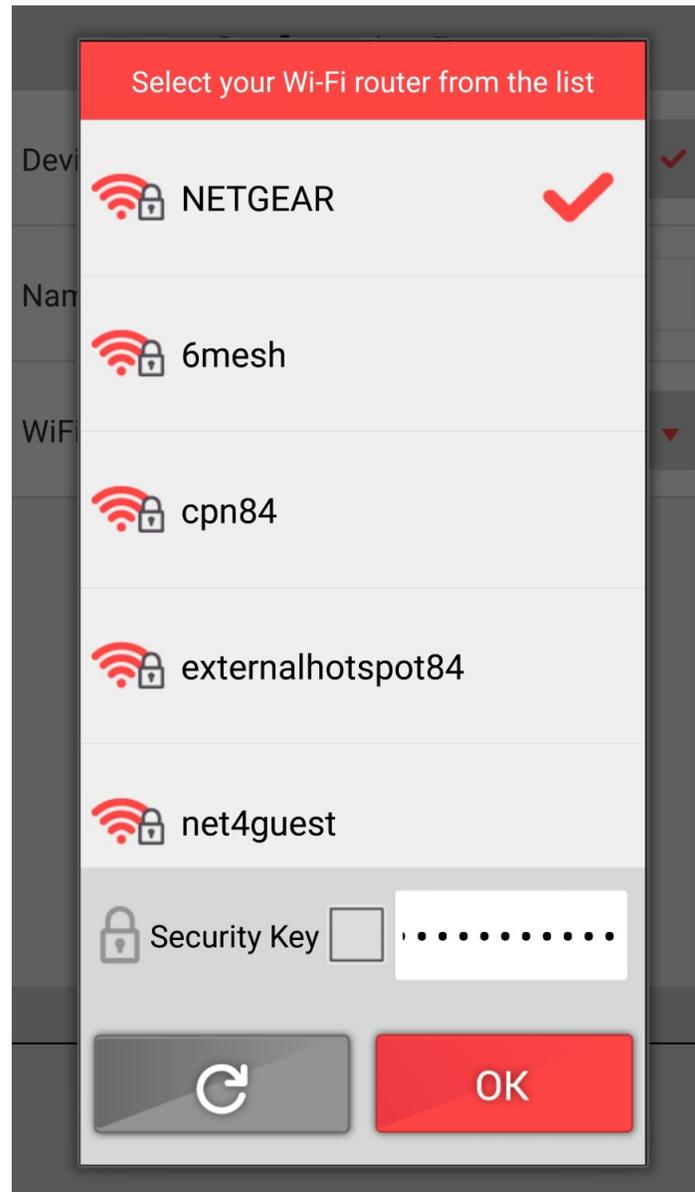


图 **63. Select Wi-Fi Router**

6. After the configuration is done, make sure the phone is connected to the same Wi-Fi network that the CC3220SF is connected.

7. Once connected, select the device from the device list.
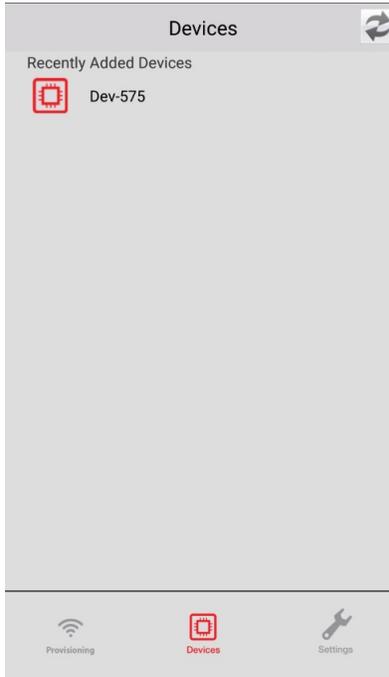


图 **64. Devices**

8. This will open a web page hosted by the device. If the device has already been provisioned to a Wi-Fi network then click on Sensor Dashboard, otherwise click on Configure WiFi Network and enter the WiFi access point credentials.
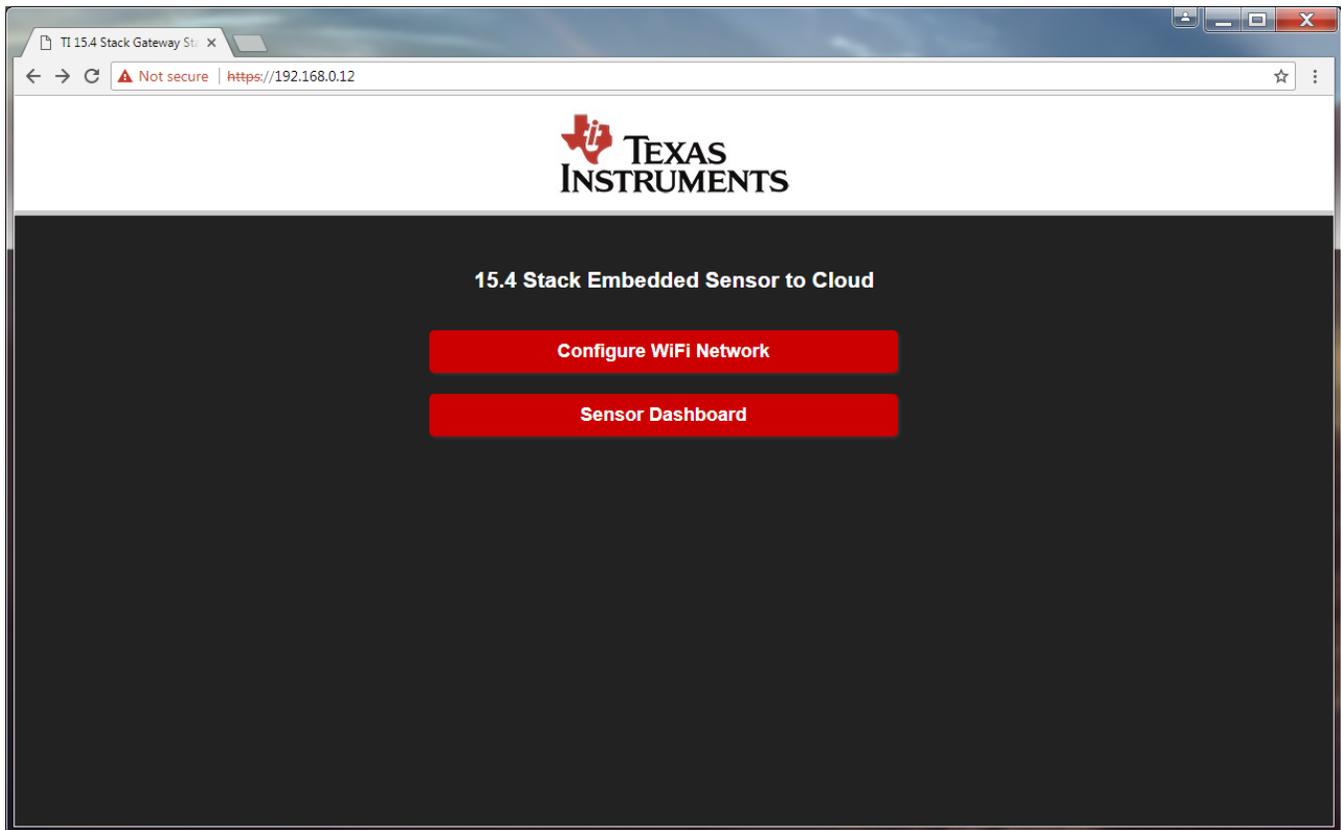


图 **65. Local Web Server Start Page**

---

注**:**      If the browser gives a warning about security certificates, ignore it, and continue to the web page.

---

9.  *(AWS IoT Example Only)* If running the AWS Example and the gateway is connected to the Internet then got to the front end website provided by StackArmor, it should look something like this *http://iotdash.stackbuilder.us/#/dashboard/home?net=AWS_IOT_MY_THING_NAME* where *AWS_IOT_MY_THING_NAME* should match the thing name in aws_iot_config.h

    NOTE: if running the IBM Example then continue with the steps below.
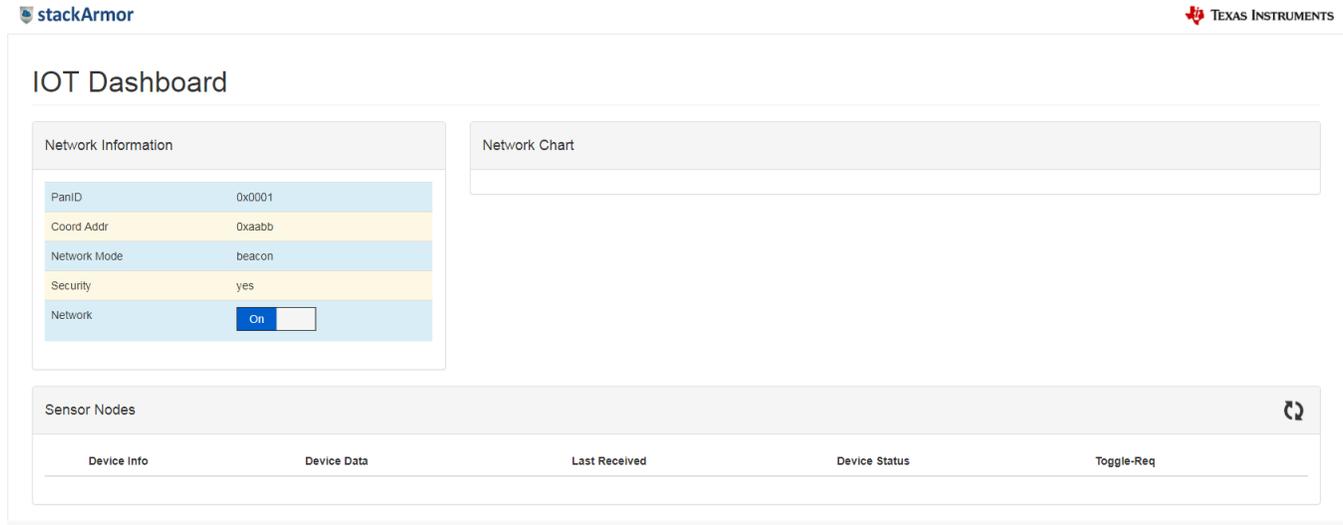


图 **66. AWS IoT Dashboard**

10. *(IBM Example Only)* On the start page click on the Sensor Dashboard button. Once on the dashboard click on the settings button to enter the IBM Cloud configuration information.
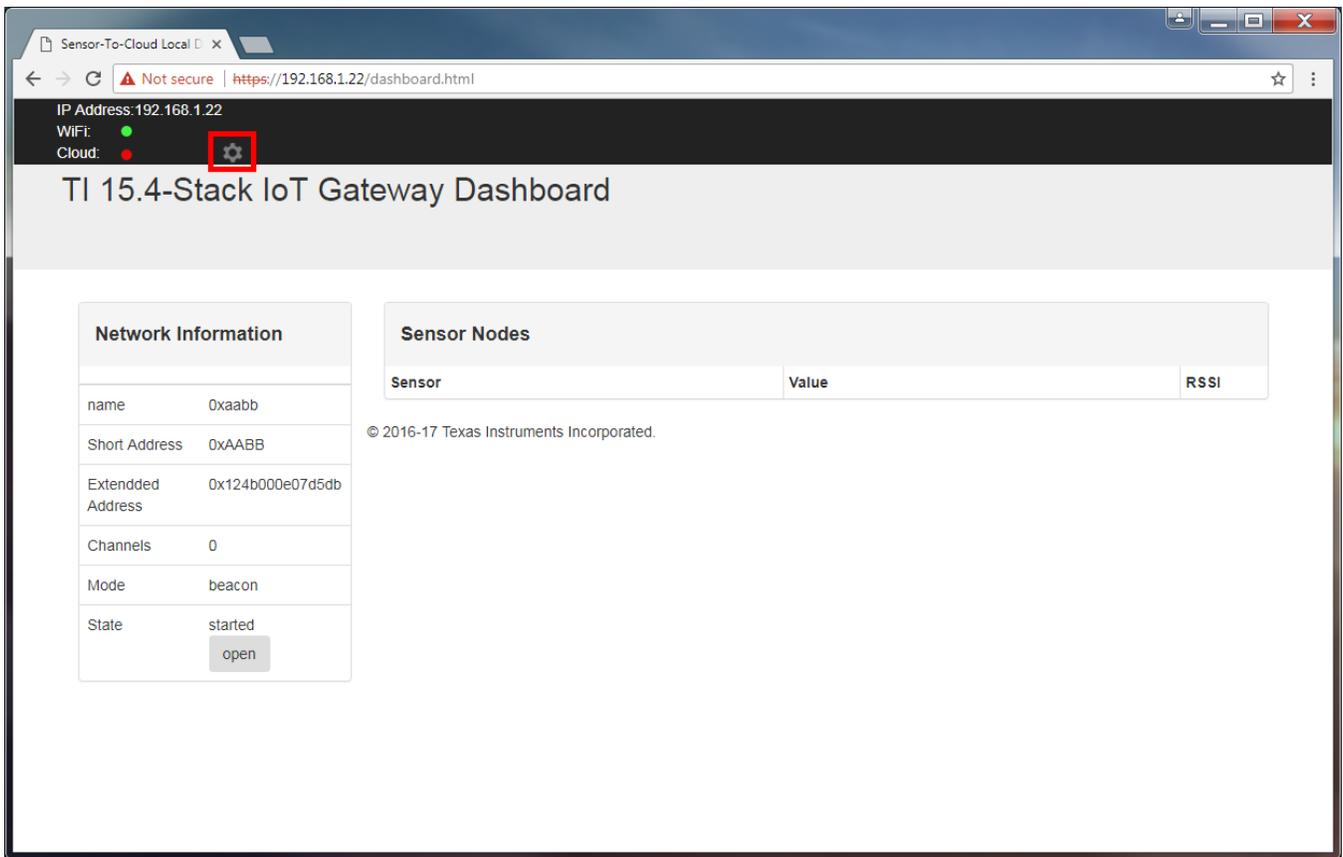
图 **67. Local Dashboard**

11.  *(IBM Example Only)* A form will pop up, fill out the form with the IBM Cloud account information. When complete, click *Save Changes* and then *Close*. If the Gateway was successful on connecting to the IBM Cloud then the Red dot next to "Cloud" on the dashboard should turn green.
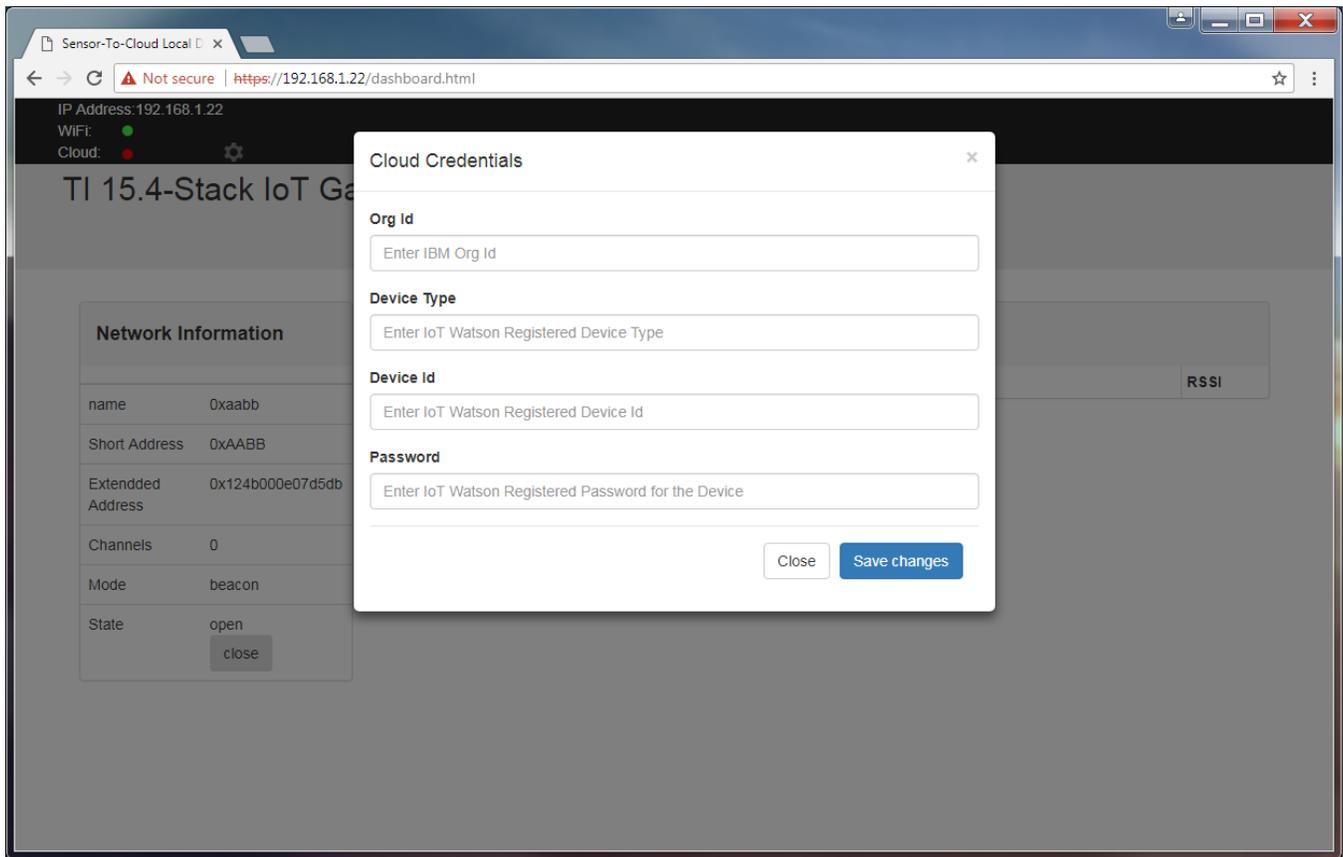


图 **68. Cloud Credentials Form**

12.  *(IBM Example Only)* Go to the IBM Cloud Foundry App URL provided on the IBM Cloud account, which looks something like *APP_NAME.mybluemix.net*.

13.  Click the *open* button on the dashboard to allow sensors to join the network.
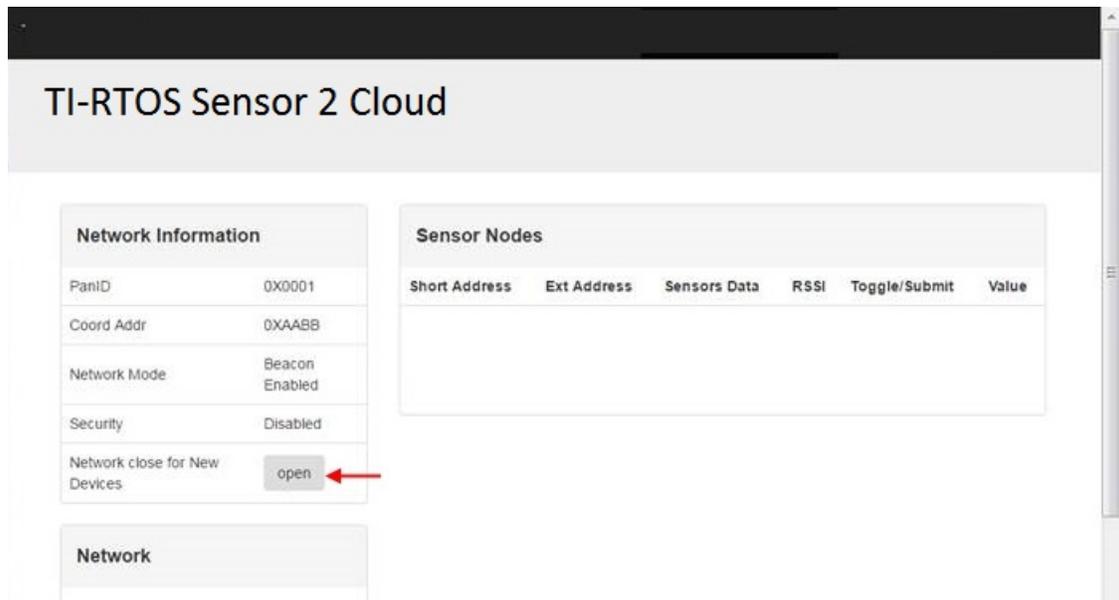
图 **69. IBM Cloud Dashboard Open Button**
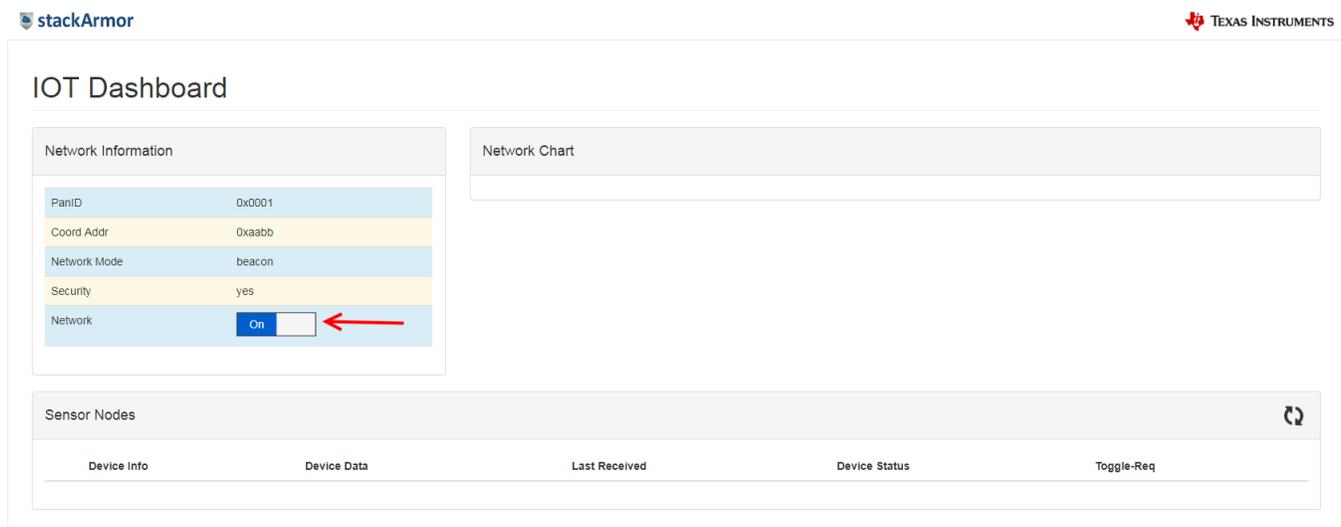


图 **70. AWS Cloud Dashboard Open Button**

14. Apply power to the LaunchPad labeled *Sensor.*

15. Now the sensor should automatically start looking for a network. If paired with the network successfully, the *Sensor* board can be viewed and controlled from the web browser.

注： If the device is not visible in the web browser, the device is most likely connected to another network. To solve this error, complete a factory reset on the sensor by pressing the reset button while holding the right button (BTN-2), and try again.
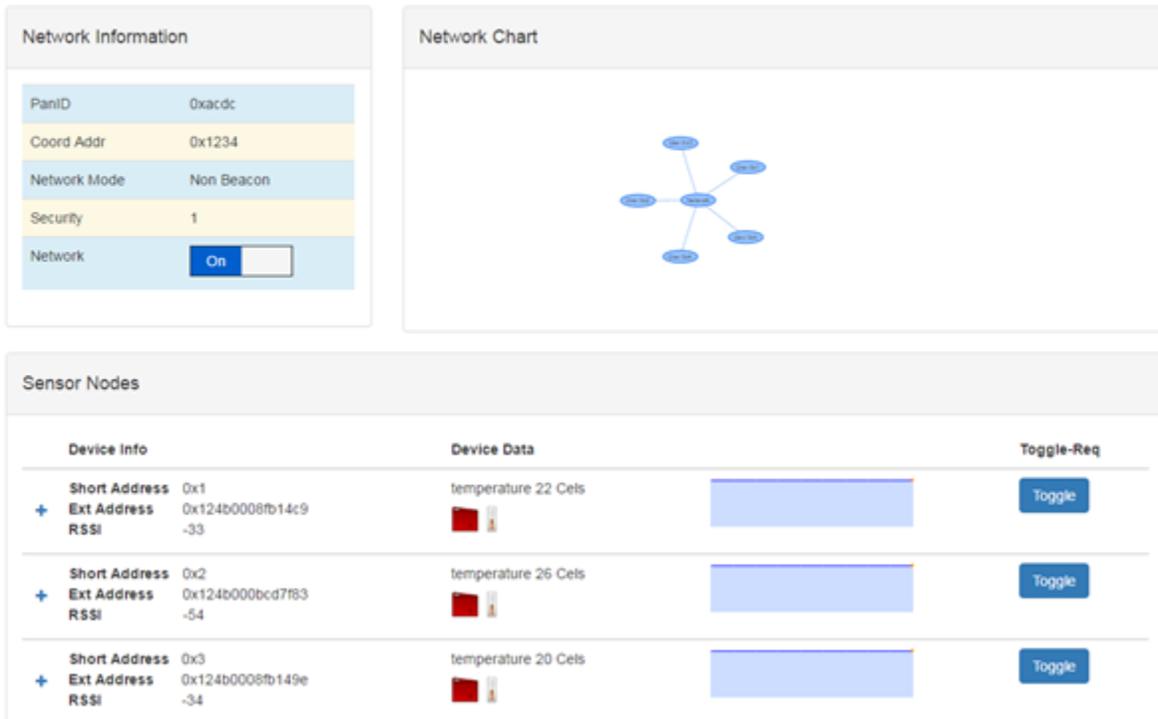
图 71. AWS Front End After Devices Joined
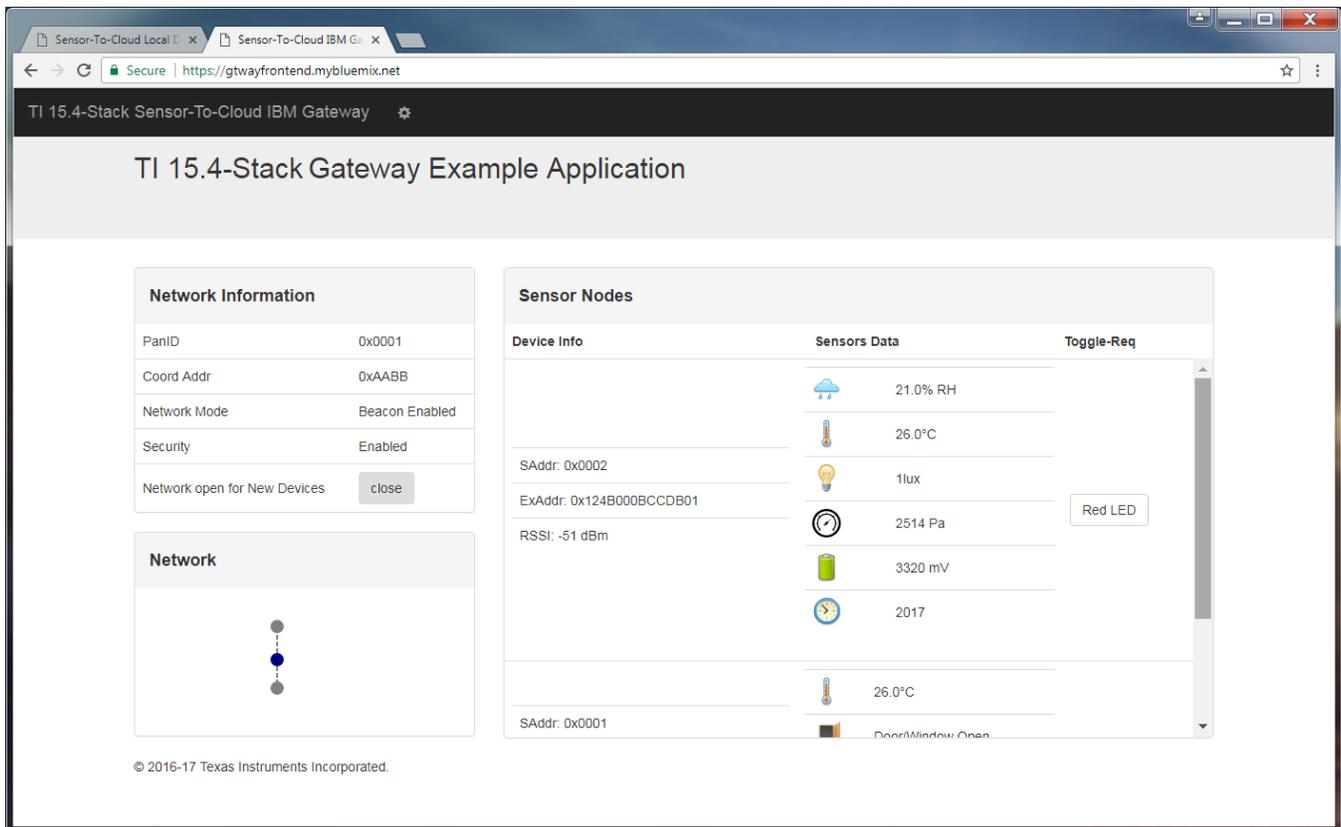
图 **72. Sensor2Cloud Front End**

#### 3.2.1.1.9.2 *Using the Local Provisioning Web Page*

---

注：        This is an alternate option if using the SimpleLink App is not desired.

---

1. Make sure the CC3220SF LaunchPad is powered on and that a serial console, like Tera Term, is opened to see the console output of the device.
2. On the PC search for Wi-Fi networks, and connect to the one broadcasted by the LaunchPad, which should look something like *mysimplelink-XXXX*.

图 **73. Connect to LaunchPad™**

3. Open a browser window, and navigate to *mysimplelink.net* this should open the start page shown below.
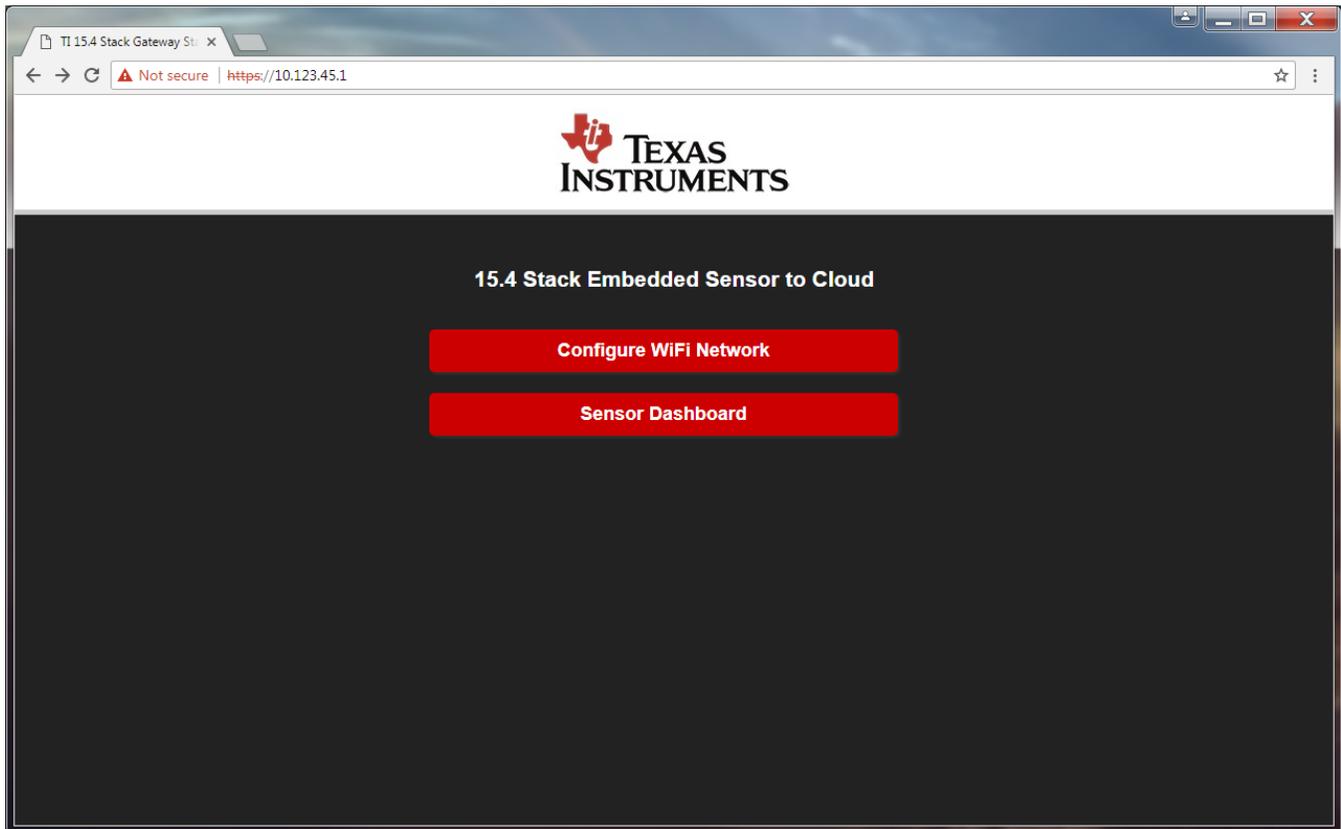


**图 74. mysimplelink.net Local Home Page**

4. Press the *Configure WiFi Network* button to go to the *Network Configuration* page.

Note: Before entering the network information make sure to have a serial terminal open for the CC3220 gateway so you can record the IP address given to the gateway.
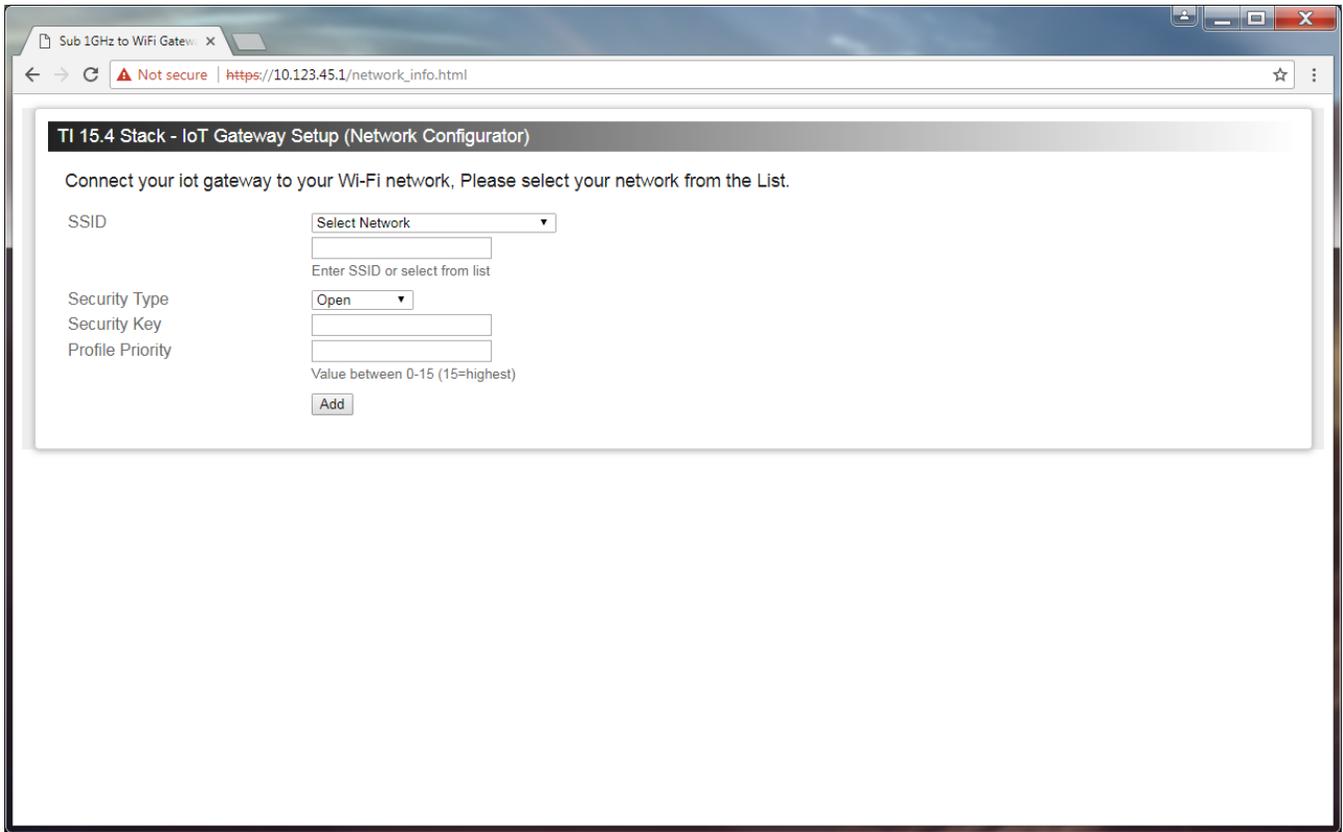
图 **75. Network Configuration Page**

5. Enter the required information for the desired Wi-Fi access point, and click on the *Add* button. A pop-up window with instructions like 图 76 will appear.
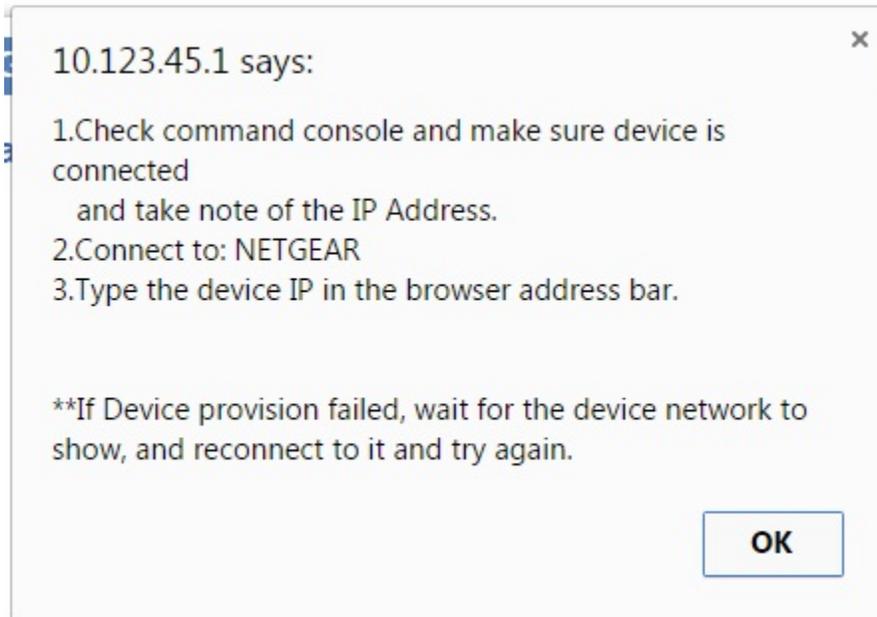


**图 76. Pop-up Message**

6. Go to the serial terminal and make sure to save the new IP address assigned to the CC3220 gateway.
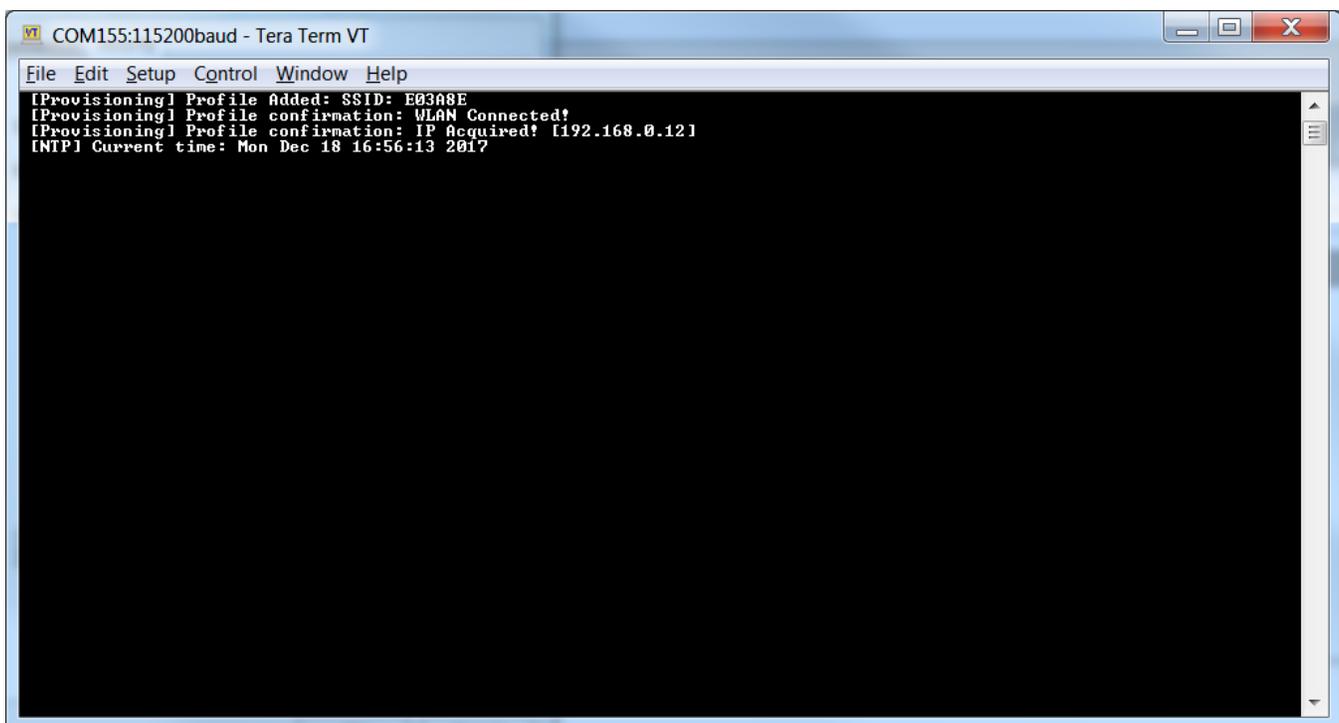


**图 77. IP Address displayed on Serial Terminal**

7. Using a computer or a mobile device connect to the same WiFi access point as the CC3220 gateway, open an internet browser and go to the IP addess assigned to the gateway, in this case it would be https://192.168.0.12 this should open the start page shown below.
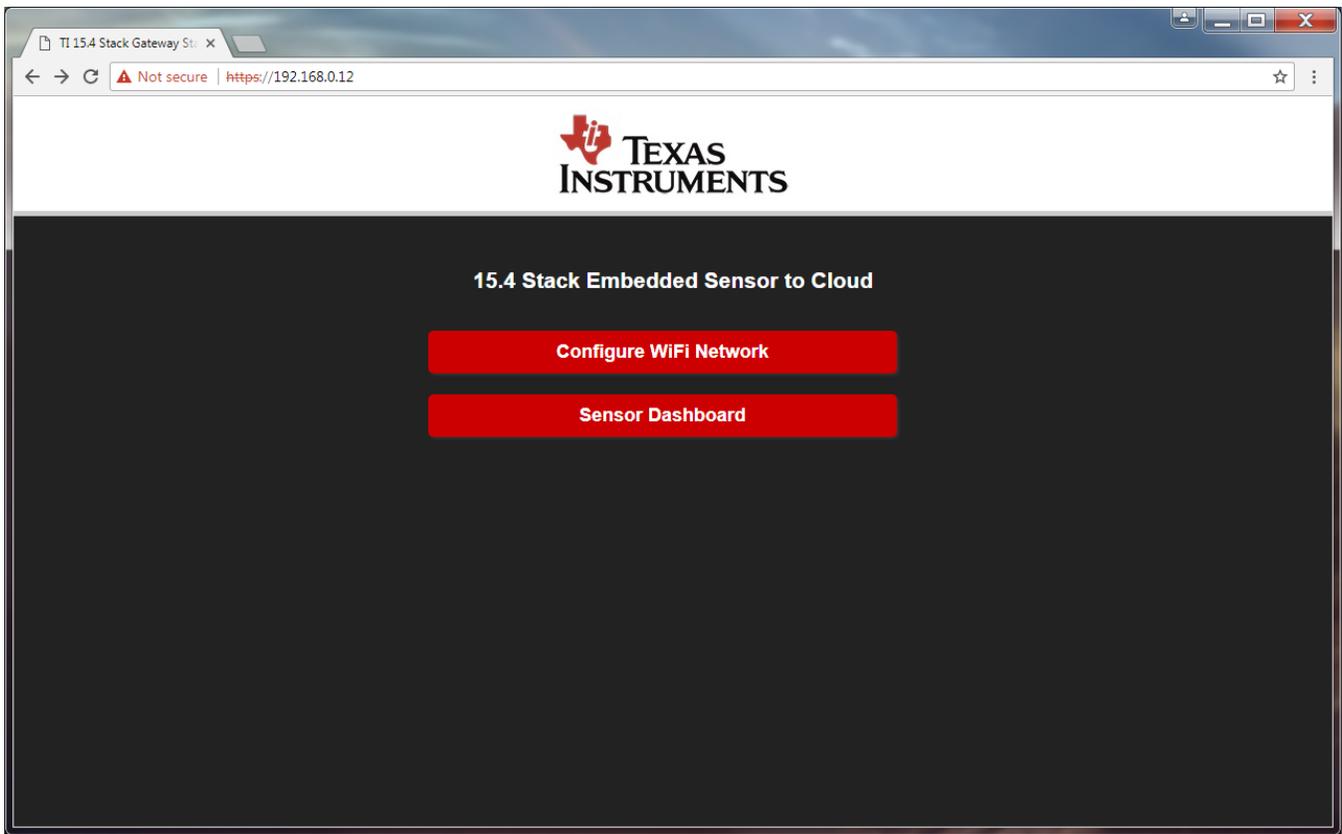
图 **78. Start Page**

8.  Go to the section *Using SimpleLink Starter Pro App* at the beginning of 节 3.2.1.1.9, and follow the instructions starting from step 8.

### 3.2.2   Test Results

#### *3.2.2.1   IoT Dashboard*

图 79 shows is an example of the IoT dashboard displayed on the web interface. Note that the current network information is shown. The network chart displays the number of connected devices, and the sensor nodes section shows the device and current sensor information for all the devices in the network.
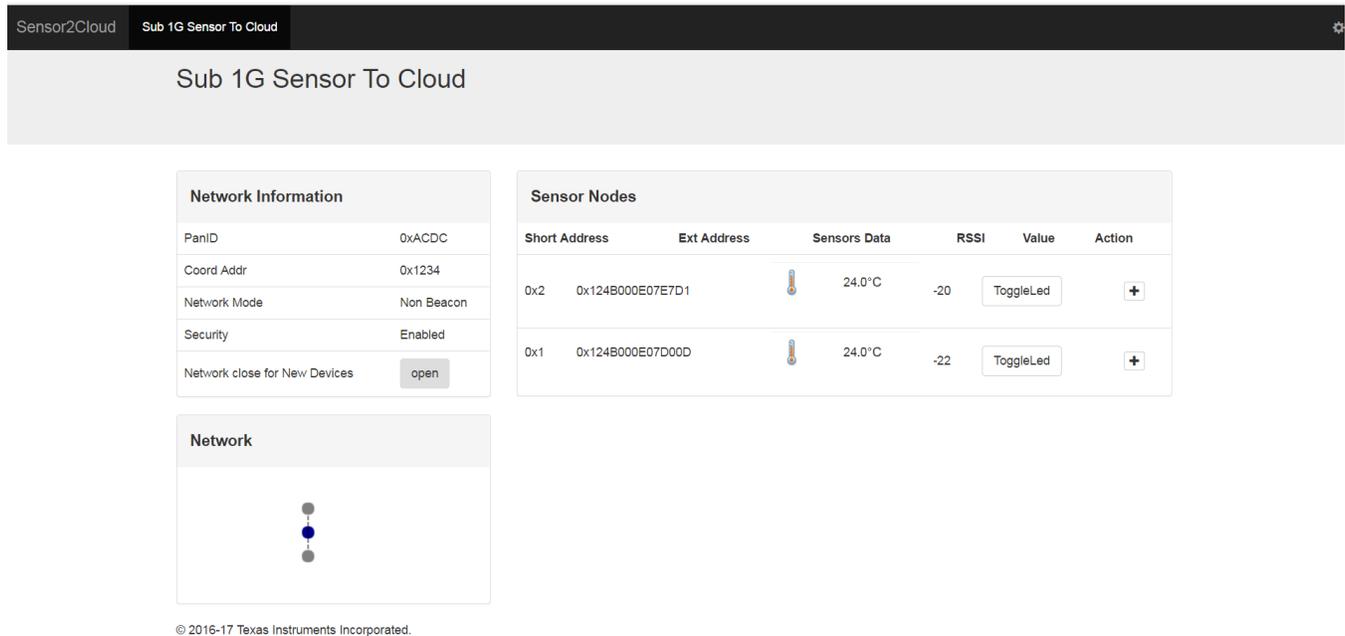


图 **79. IBM Sensor-to-Cloud Dashboard Results**

# 4 Design Files

This reference design showcases the connectivity between CC3220SF and CC13x0 devices. The CC3220SF acts as a gateway processor and the CC13x0 as communication node. The CC3220SF LaunchPad is used as a platform for gateway processor, and the CC13x0-based LaunchPad acts as communication node. The recommended schematics for this reference design uses the schematics of the CC3220SF LaunchPad and CC13x0 LaunchPad and interface the two devices using the UART lines. This IoT gateway reference design only uses one UART port. In addition, bootloader backdoor pins are described in the CC2538/CC26xx Serial Bootloader Interface application report. These pins can be connected to upgrade the firmware on the CC13x0 using the serial ROM bootloader on the CC13x0 devices.

## 4.1 Schematics

To download the schematics for this reference design, see the following links:

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

## 4.2 Bill of Materials

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

## 4.3 PCB Layout Recommendations

For layout prints, Altium project files, Gerber files, and assembly drawings, see the following links:

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

# 5 Software Files

To download the software files, see the link at https://git.ti.com/tidc01002/tidc01002.

# 6 Related Documentation

1. Texas Instruments, SimpleLink TI 15.4-Stack IEEE 802.15.4e/g Standard Based Star Networking Software Development Kit, Tools Folder
2. Texas Instruments, SimpleLink CC3220 SDK, Tools Folder
3. Texas Instruments, TI 15.4-Stack Wiki, Wiki Page
4. Texas Instruments, *TI 15.4-Stack Embedded Developers Guide*

## 6.1 商标

LaunchPad, Internet-on-a-chip, SmartConfig, Code Composer Studio are trademarks of Texas Instruments.
SimpleLink is a trademark of Texas Instruments Incorporated.
Cortex is a registered trademark of ARM Limited.
ARM, Cortex are registered trademarks of Arm Limited.
Bluetooth is a registered trademark of Bluetooth SIG, Incorporated.

IOS is a registered trademark of Cisco.
Android is a trademark of Google Inc.
IBM Watson IoT is a registered trademark of International Business Machines Corporation.
Linux is a registered trademark of Linus Torvalds.
Wi-Fi, Wi-Fi Direct are registered trademarks of Wi-Fi Alliance.
All other trademarks are the property of their respective owners.

# 7 About the Authors

**HECTOR RAMOS** is a software applications engineer at Texas Instruments, where he has been working at since 2014 and is specialized in low-power wireless protocols. Hector earned his bachelors of science in electrical engineering and computer science from University of California Berkeley.

**ANDRES BLANCO** is an applications engineer at Texas Instruments, where he is responsible for supporting customers designing low power wireless systems. Andres earned his bachelor of science in computer engineering from University of Puerto Rico in Mayagüez, PR.

# 修订历史记录

注：之前版本的页码可能与当前版本有所不同。

## 有关 TI 设计信息和资源的重要通知

德州仪器 (TI) 公司提供的技术、应用或其他设计建议、服务或信息，包括但不限于与评估模块有关的参考设计和材料（总称"TI 资源"），旨在帮助设计人员开发整合了 TI 产品的 应用；如果您（个人，或如果是代表贵公司，则为贵公司）以任何方式下载、访问或使用了任何特定的 TI 资源，即表示贵方同意仅为该等目标，按照本通知的条款进行使用。

TI 所提供的 TI 资源，并未扩大或以其他方式修改 TI 对 TI 产品的公开适用的质保及质保免责声明；也未导致 TI 承担任何额外的义务或责任。TI 有权对其 TI 资源进行纠正、增强、改进和其他修改。

您理解并同意，在设计应用时应自行实施独立的分析、评价和 判断，且应全权负责并确保 应用的安全性，以及您的 应用（包括应用中使用的所有 TI 产品））应符合所有适用的法律法规及其他相关要求。你就您的 应用声明，您具备制订和实施下列保障措施所需的一切必要专业知识，能够 (1) 预见故障的危险后果，(2) 监视故障及其后果，以及 (3) 降低可能导致危险的故障几率并采取适当措施。您同意，在使用或分发包含 TI 产品的任何 应用前，您将彻底测试该等 应用 和该等应用所用 TI 产品的 功能而设计。除特定 TI 资源的公开文档中明确列出的测试外，TI 未进行任何其他测试。

您只有在为开发包含该等 TI 资源所列 TI 产品的 应用时，才被授权使用、复制和修改任何相关单项 TI 资源。但并未依据禁止反言原则或其他法理授予您任何TI知识产权的任何其他明示或默示的许可，也未授予您 TI 或第三方的任何技术或知识产权的许可，该等产权包括但不限于任何专利权、版权、屏蔽作品权或与使用TI产品或服务的任何整合、机器制作、流程相关的其他知识产权。涉及或参考了第三方产品或服务的信息不构成使用此类产品或服务的许可或与其相关的保证或认可。使用 TI 资源可能需要您向第三方获得对该等第三方专利或其他知识产权的许可。

TI 资源系"按原样"提供。TI 兹免除对 TI 资源及其使用作出所有其他明确或默认的保证或陈述，包括但不限于对准确性或完整性、产权保证、无屡发故障保证，以及适销性、适合特定用途和不侵犯任何第三方知识产权的任何默认保证。

TI 不负责任何申索，包括但不限于因组合产品所致或与之有关的申索，也不为您辩护或赔偿，即使该等产品组合已列于 TI 资源或其他地方。对因 TI 资源或其使用引起或与之有关的任何实际的、直接的、特殊的、附带的、间接的、惩罚性的、偶发的、从属或惩戒性损害赔偿，不管 TI 是否获悉可能会产生上述损害赔偿，TI 概不负责。

您同意向 TI 及其代表全额赔偿因您不遵守本通知条款和条件而引起的任何损害、费用、损失和/或责任。

本通知适用于 TI 资源。另有其他条款适用于某些类型的材料、TI 产品和服务的使用和采购。这些条款包括但不限于适用于 TI 的半导体产品 (http://www.ti.com/sc/docs/stdterms.htm)、评估模块和样品 (http://www.ti.com/sc/docs/sampterms.htm) 的标准条款。