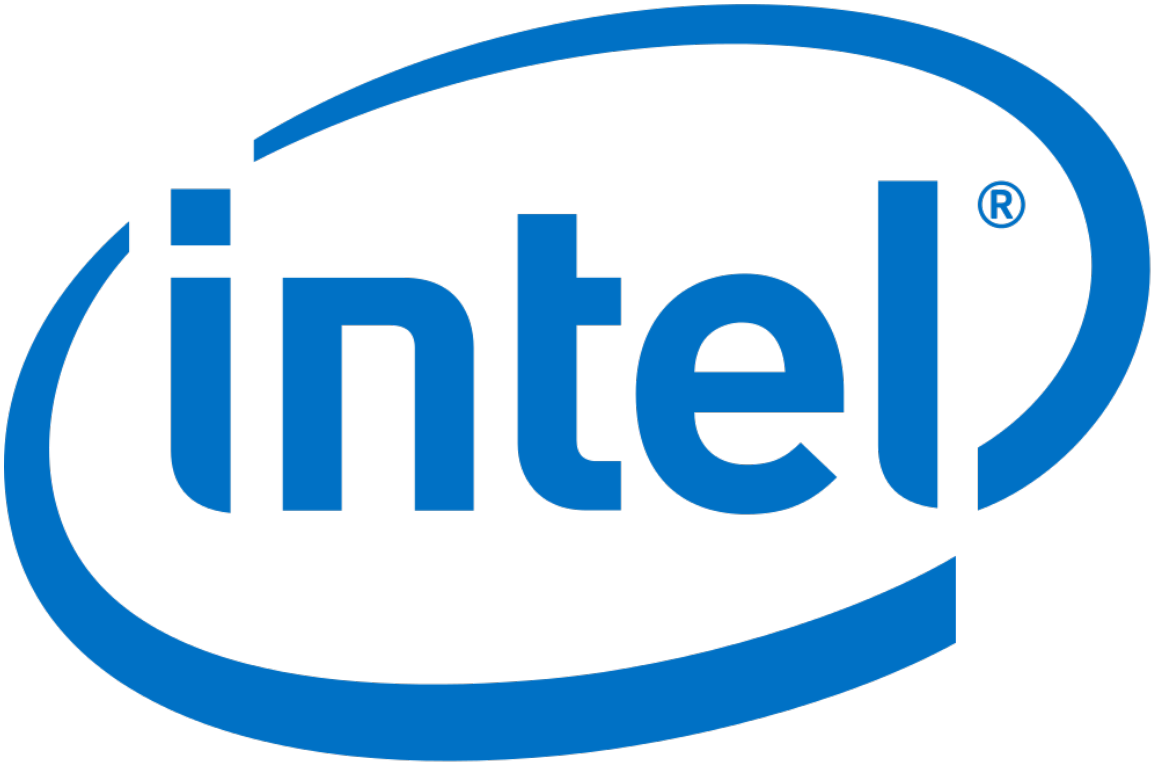
Intel IoT Gateway / Arduino 101 with AWS IoT

Hands-On Workshop

Authors:
Martin Kronberg | martin.kronberg@intel.com | @martinkronberg
Priyanka Bagade | priyanka.bagade@intel.com | @bagadepriyanka



Topics covered: Intel IoT Gateway, Grove sensors, Node-RED, and AWS Internet of Things Platform

In this lab, we will unbox and set up an Intel IoT Gateway and the Arduino 101 board (with a Grove Starter kit)
a. Using Node-RED, running on the Intel NUC Gateway… ADD DESCRIPTION OF FINAL WORKSHOP MATERIAL

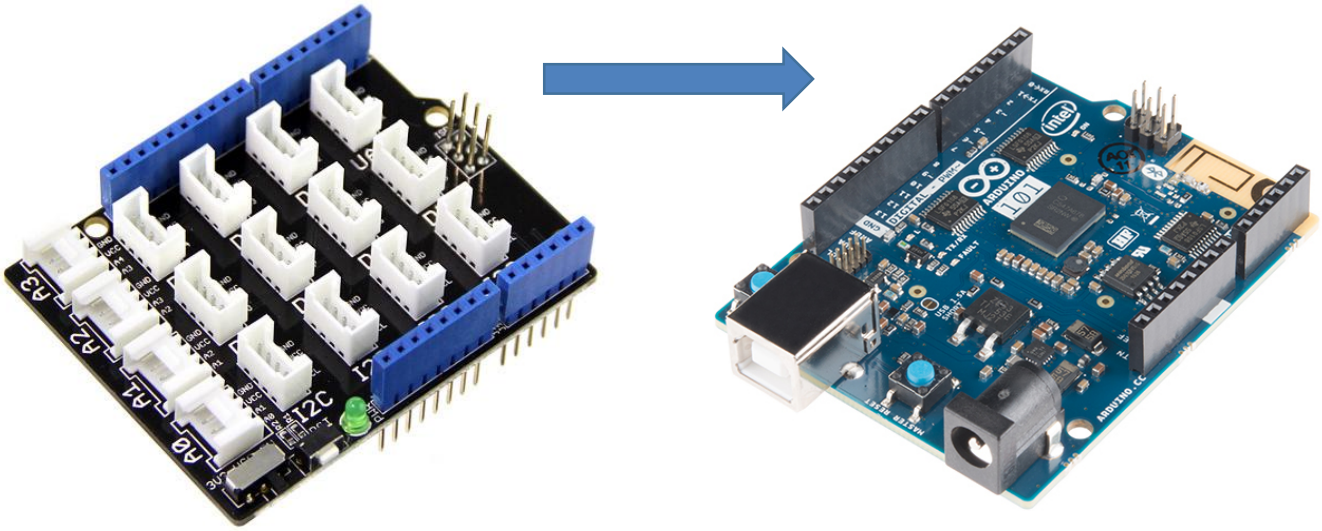
Getting Started with Grove IoT Commercial Developer Kit

Step 1 - Unboxing Grove IoT Commercial Developer Kit and Arduino 101

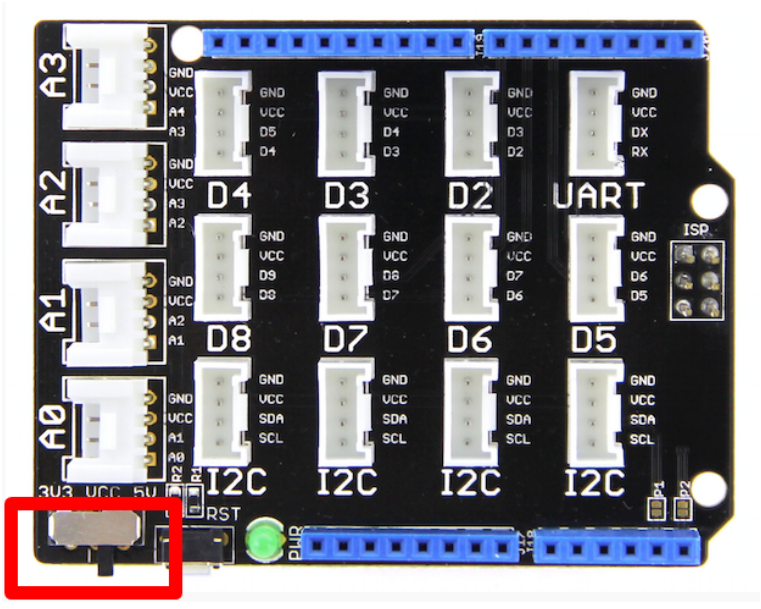
Arduino 101 is not included in Grove IoT Commercial Development Kit and will be given separately.

Step 2 – Connect Base shield to Arduino 101

Base shield will be included in the Grove starter kit. Attach the Grove base shield to the Arduino 101.

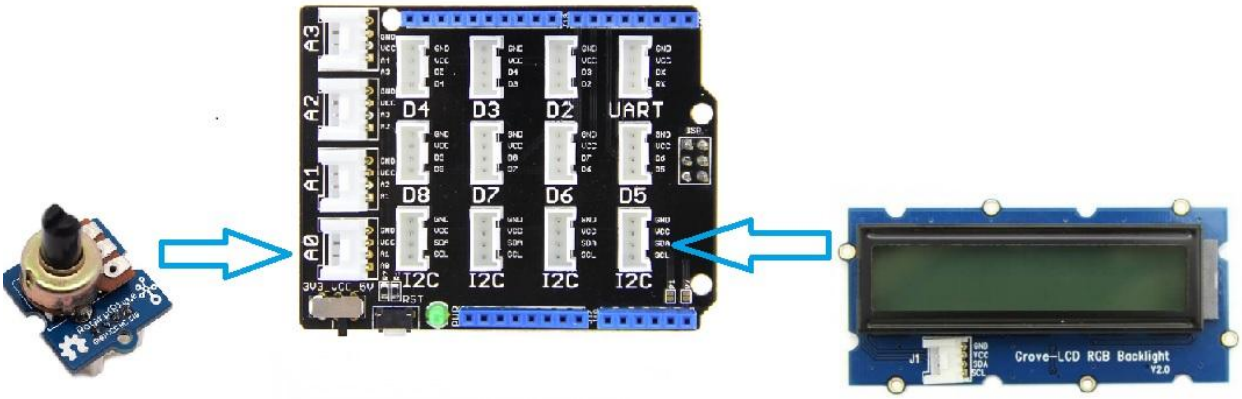


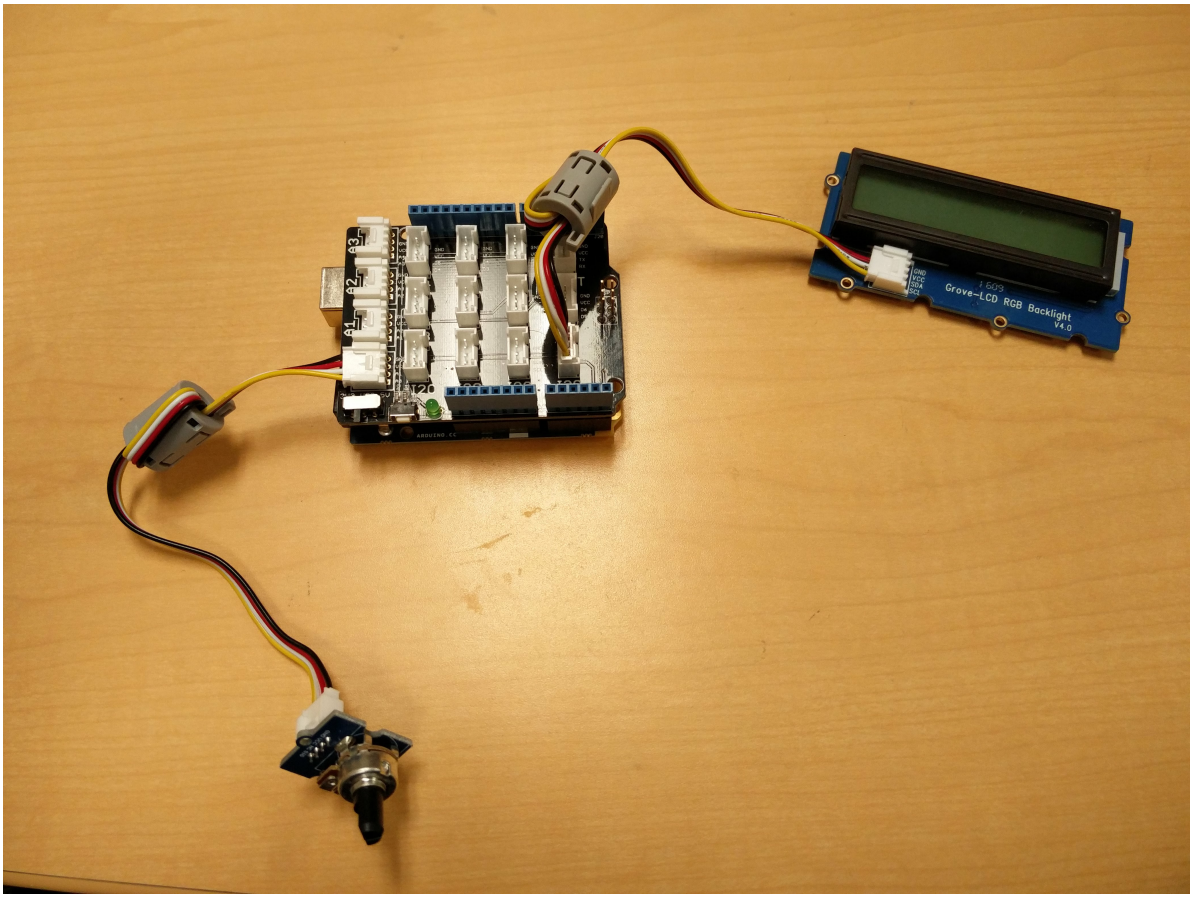
Step 3 – Move Base Shield switch towards 5 Volt



Step 4 – Connect LCD and Rotary Angle to Arduino 101

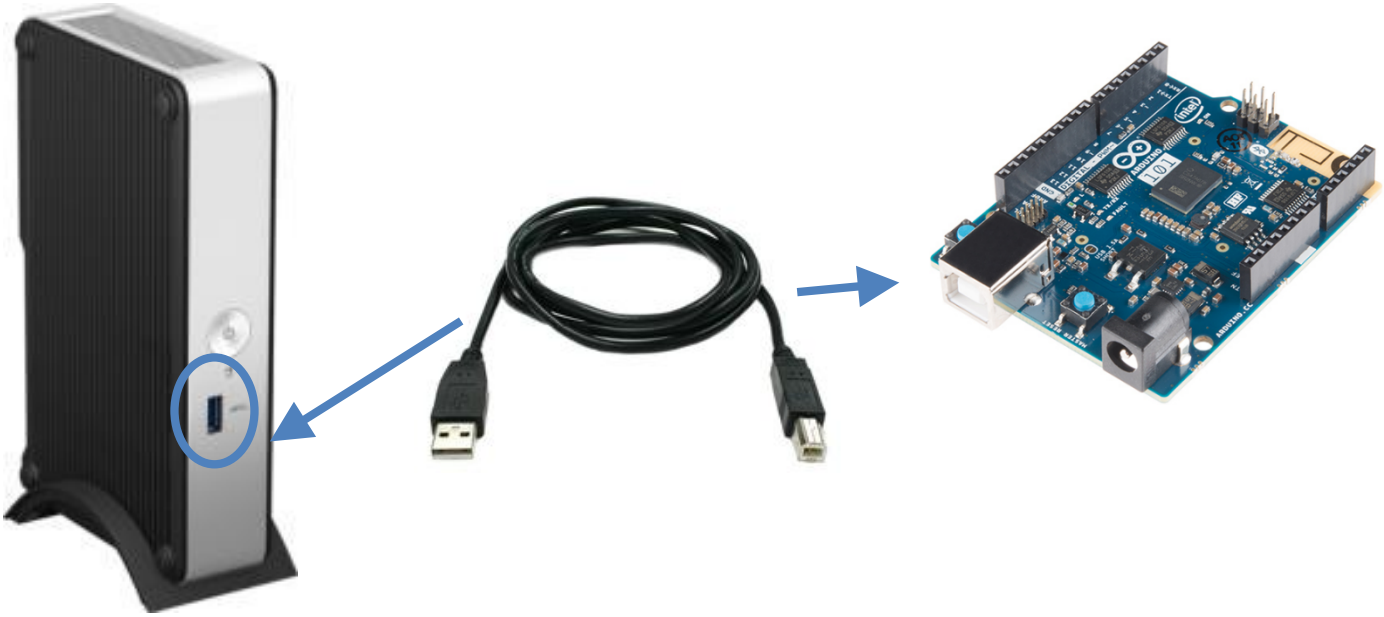
Connect LCD to any I2C socket and rotary angle to A0 socket with a grove connector cable as shown below, the cables are in the green box.



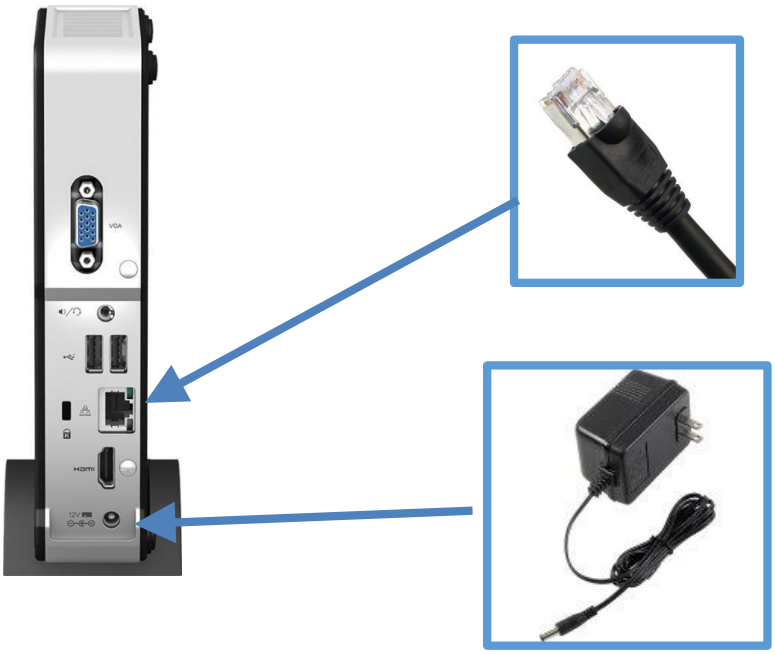


Step 5 – Connect Arduino 101 and Intel IoT Gateway via USB A-B Cable

Now connect Arduino 101 with Intel IoT Gateway via USB A-B cable.



Step 6 – Connect Intel IoT gateway with Ethernet cable and Power adapter



Step 7 – Press Power button to start Gateway

Press power button to start gateway. After around 2 minutes, it will display the IP address of Gateway on the LCD. **Note down the IP address.**

Step 8 – Open your browser and go to IP address to access the IoT Developer Hub

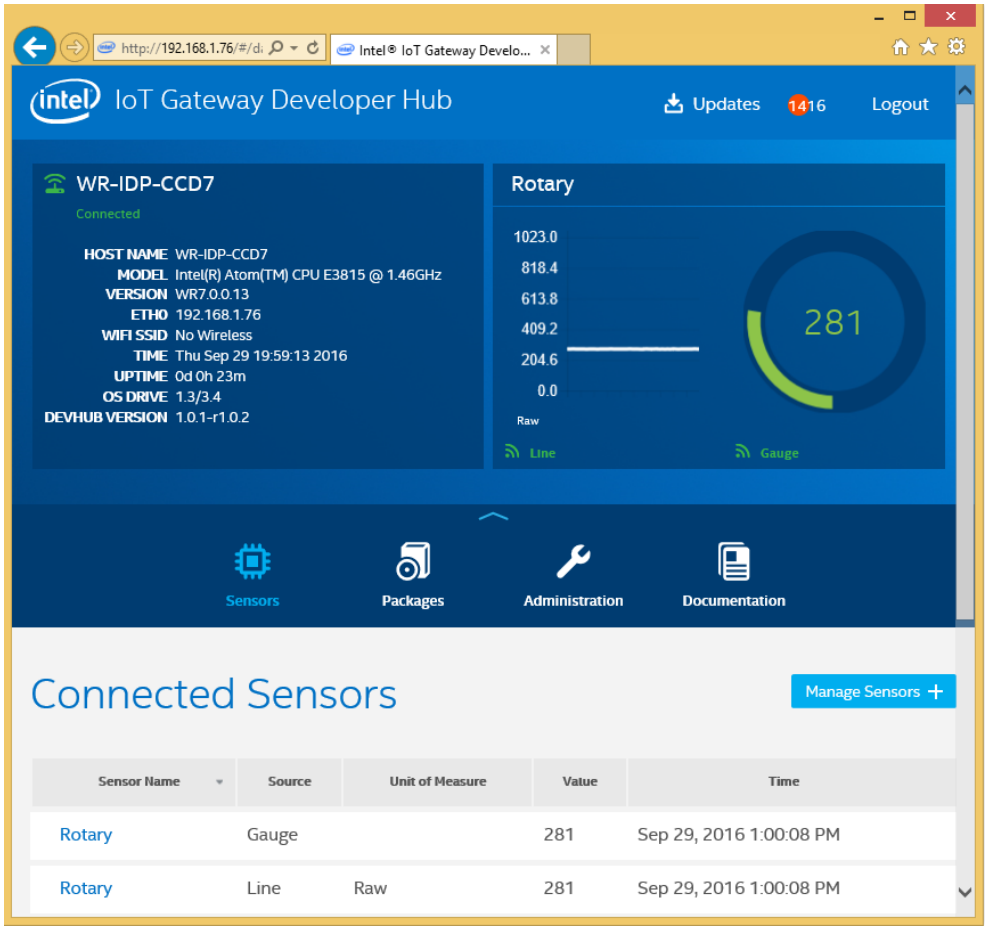
- Login to the Intel® IoT Gateway Developer Hub using root for both the username and the password.
- Use the Intel® IoT Gateway Developer Hub to explore and extend the basic capabilities of the Gateway.

Intel NUC Developer Hub Overview

The Developer Hub is a front end interface for the Gateway. It has 5 main functions:

- Display sensor data and basic Gateway information on a configurable dashboard
- Access the Node-RED development environment
- Add repositories, install and upgrade packages
- Administer the Gateway – update OS, configure network setting
- Access documentation

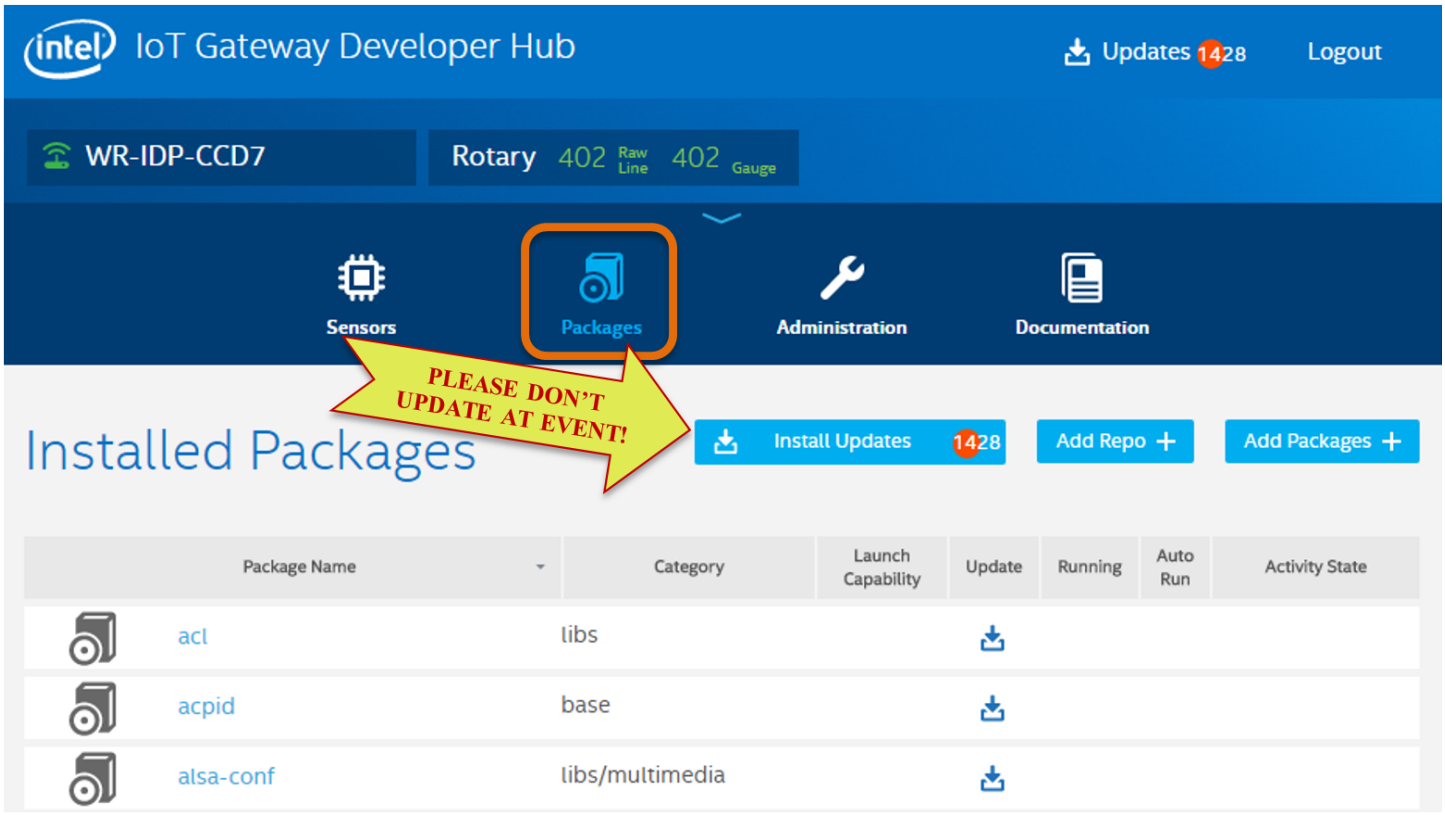
Dashboard



Dashboard with Rotary Angle Sensor Displayed

By default, the Gateway is programmed to display the value of the rotary angle sensor (knob) on the dashboard. Twist the knob to see the gauge move.

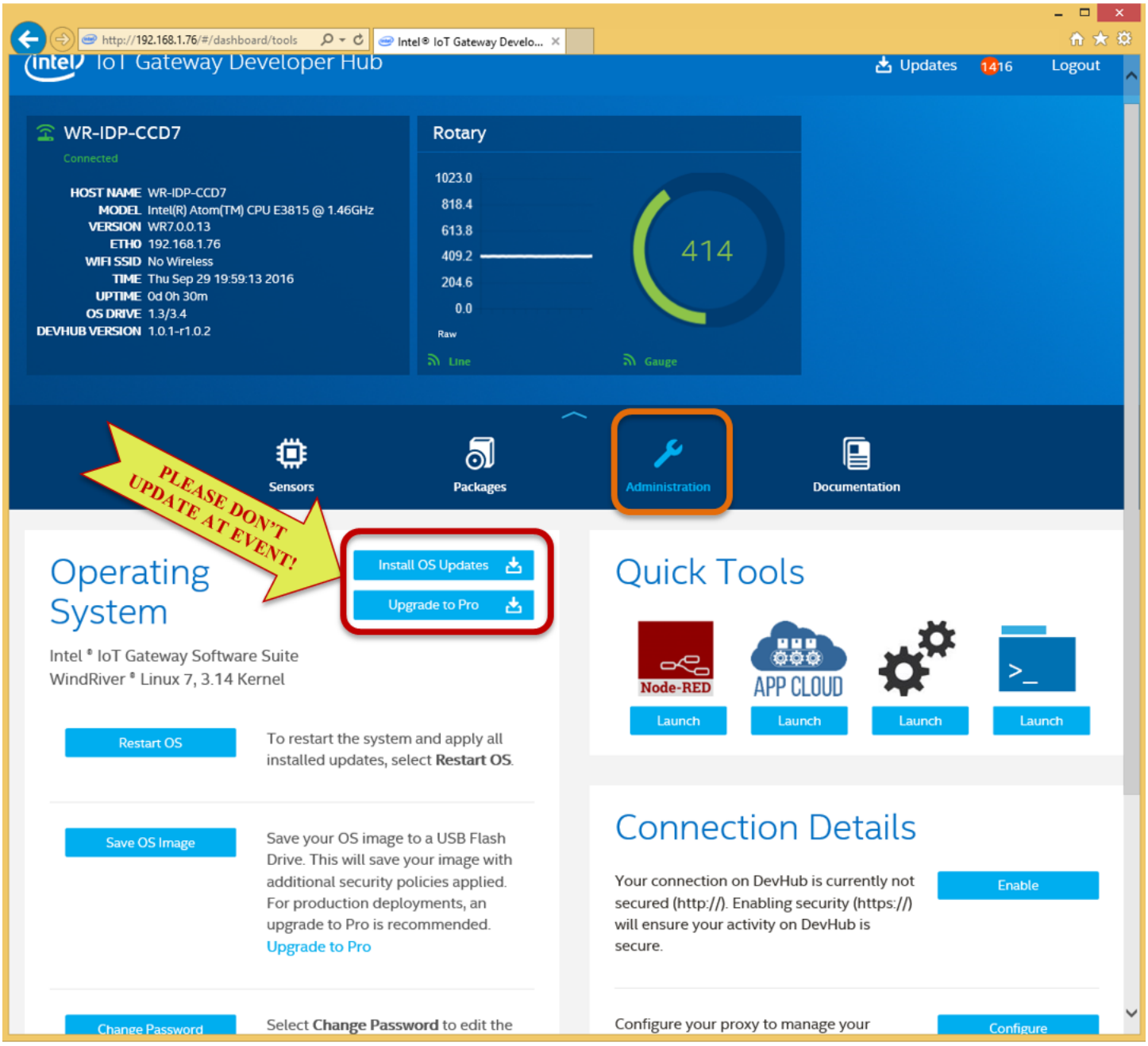
Package Manager



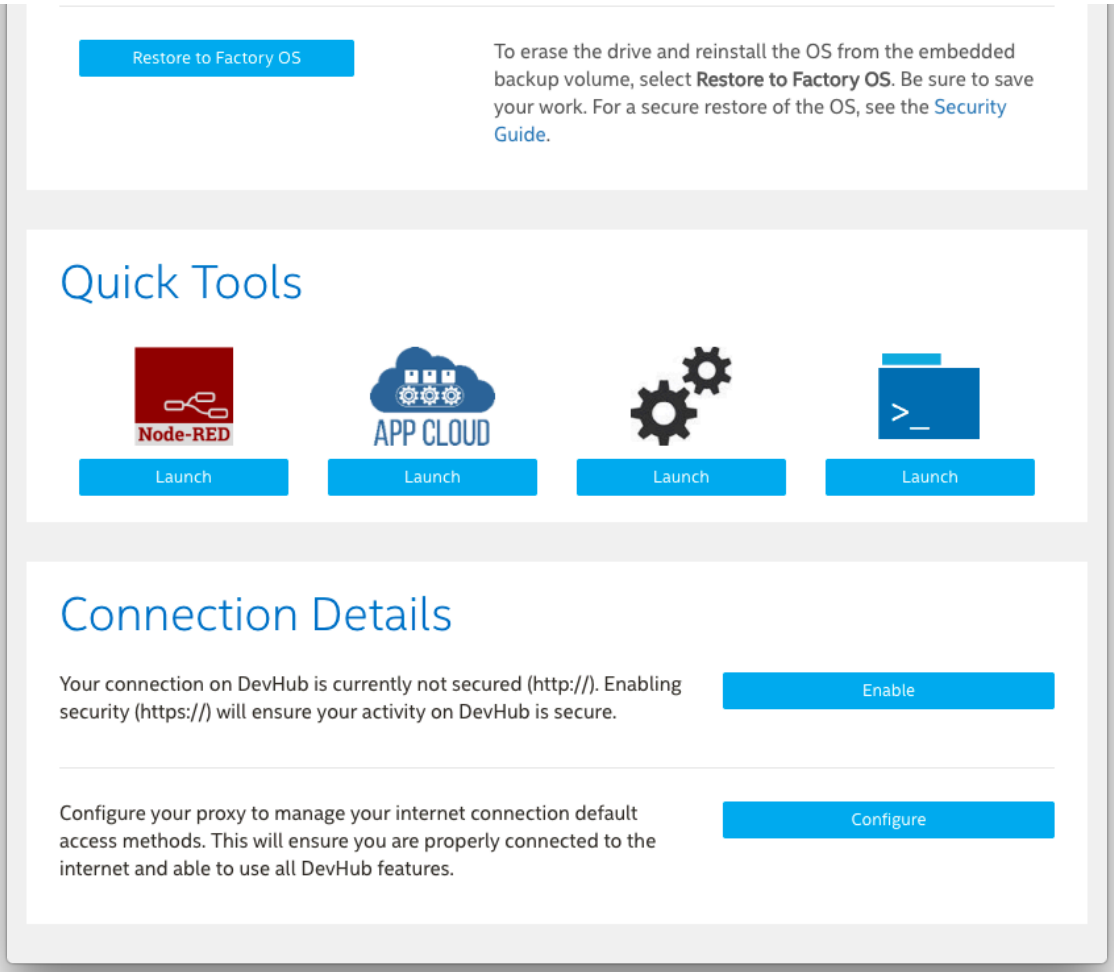
View of Packages screen

On the package manager interface, you will most likely see an option to “Install Updates”.
Do Not Install Updates!
This process can take a significant amount of time and will use up a lot of bandwidth.

Administration

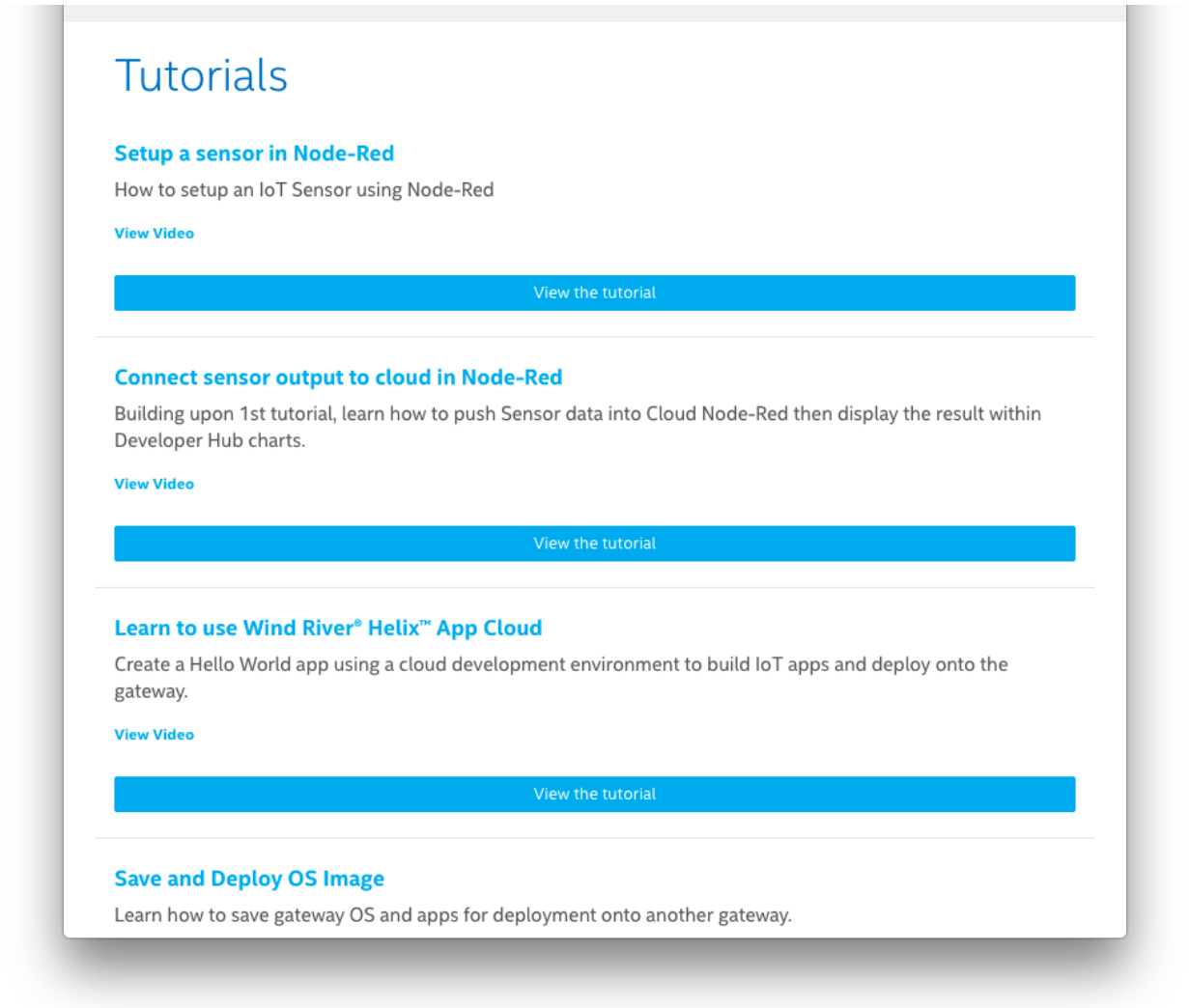


In the Administration interface you will see options to Update the OS, Upgrade to Pro, as well as others.
Please do not select any of these options!



Further down on the page you will see links to some quick tools. Feel free to explore these, however:
Please do not alter any settings in the configuration tool (two gears)!

Documentation



The Documentation tab will give you links to some of the documentation for using the Gateway.

Connect to your Gateway with a terminal client

In order to perform advanced configuration of the Gateway either a monitor and keyboard, or a Secure Shell (SSH) connection is required. On OSX and Linux there are default programs that can do this - Screen and SSH respectively. However, on Windows no default exists, however Putty is light weight and easy to install and can be used to SSH into the Gateway.

For Windows Users:

- Visit the PuTTY download page.
- Under the "For Windows on Intel x86" heading, click on the "putty.exe" link to download the latest release version to your computer.

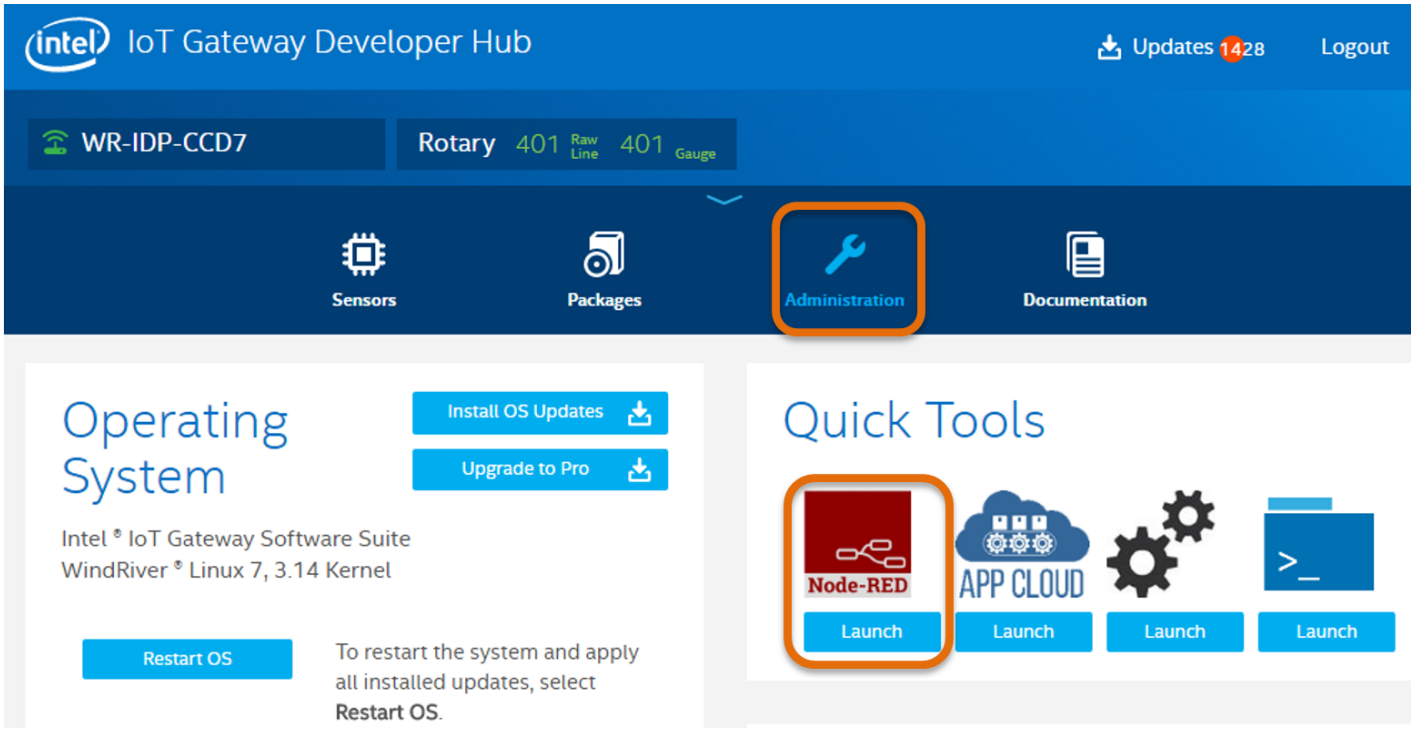
- Double-click putty.exe on your computer to launch PuTTY.
- Enter IP address of the Gateway
- You can login with the username **root** and the password **root**.

For Mace and Linux Users:

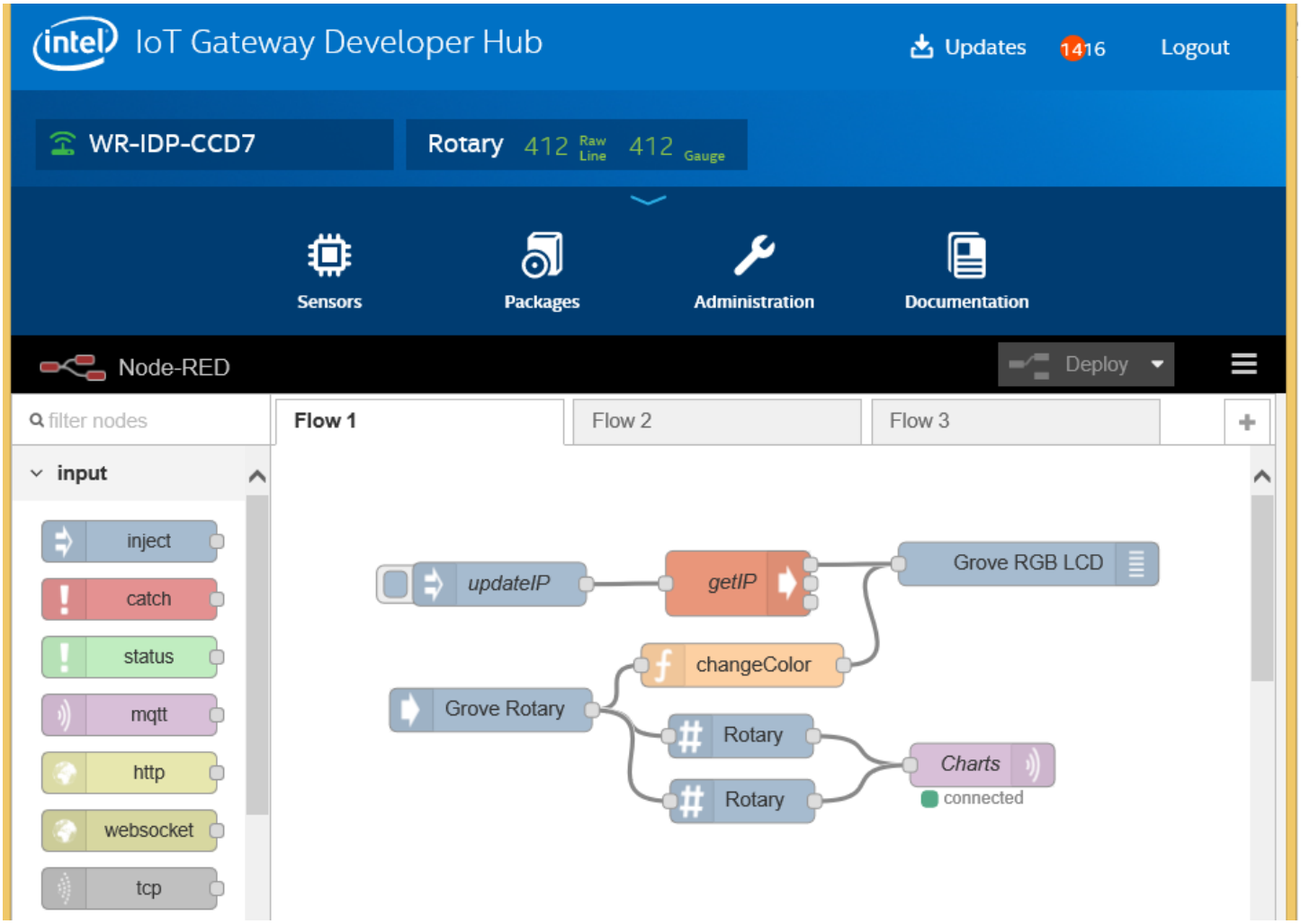
- Open a Terminal
- Type $ *ssh root@<<IP Address>>. Replace <<IP Address>> with the IP address of your gateway.*
- Enter ***root*** as password

Starting Node-RED for Blinky LED

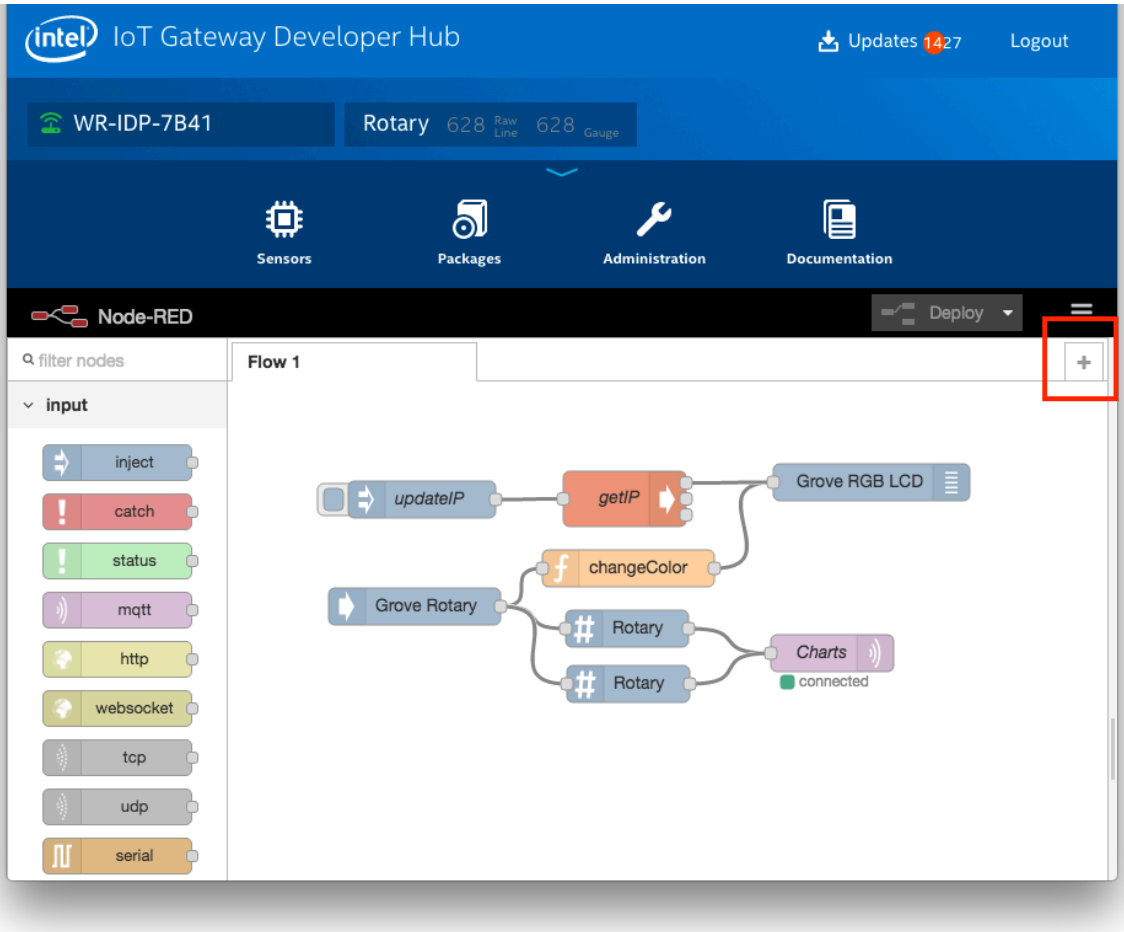
- Go to Development Hub by entering the IP Address assigned to your Gateway (Eg. http://192.168.1.1) in your browser.
- To load the Node-RED interface, click **Administration** on the navigation ribbon. Click **Launch** under the Node-RED icon.



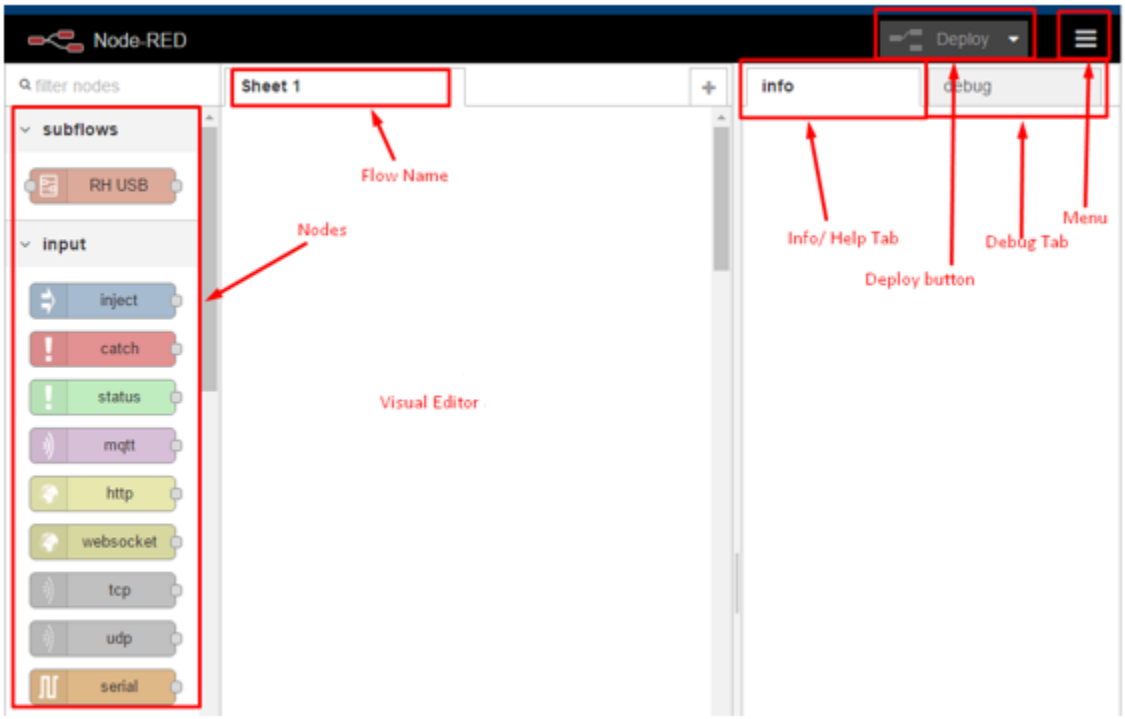
- Node-Red is a visual tool for writing the Internet of Things (IoT).
- Enter username and password as **root** and **root** respectively.



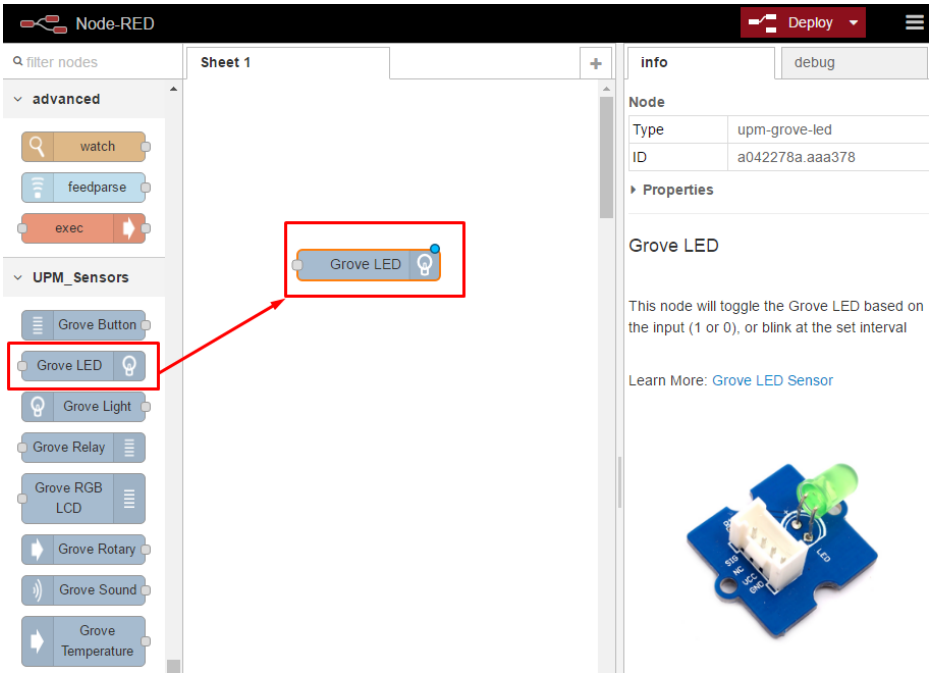
You will see the default application that sends the IP address to the LCD screen and the rotary angle sensor data to the chart on the dashboard. Leave this flow as is for now.



Create a new flow to program the rest of the exercises. Click the + button in the upper right corner of the Node-RED interface.

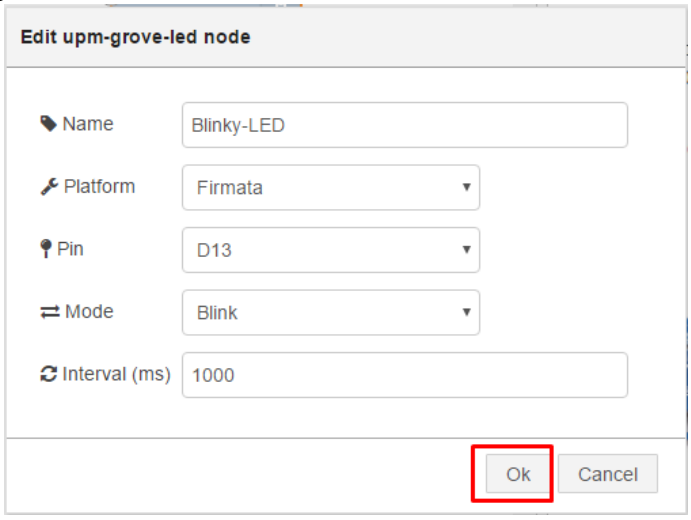


- Deploy Blinky LED Node. Go to Nodes, at the bottom of page there will be group on Nodes called UPM Sensors. Drag Grove LED from Nodes to Visual Editor

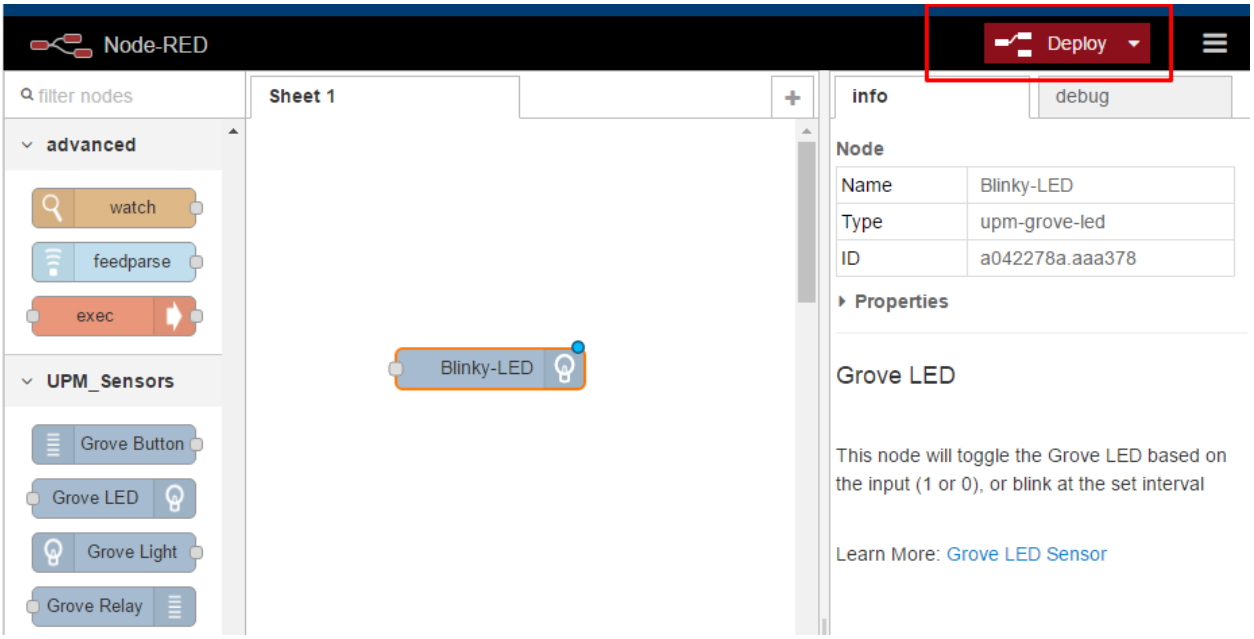


- Double Click on Grove LED node on visual Editor. It will open a dialog box to edit properties of node.
  - ➢ Name – Blinky LED
  - ➢ Platform – Firmata
  - ➢ Pin – D13
  - ➢ Mode- Blink
  - ➢ Interval(ms) – 1000

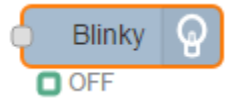
  Then Click OK to save changes.



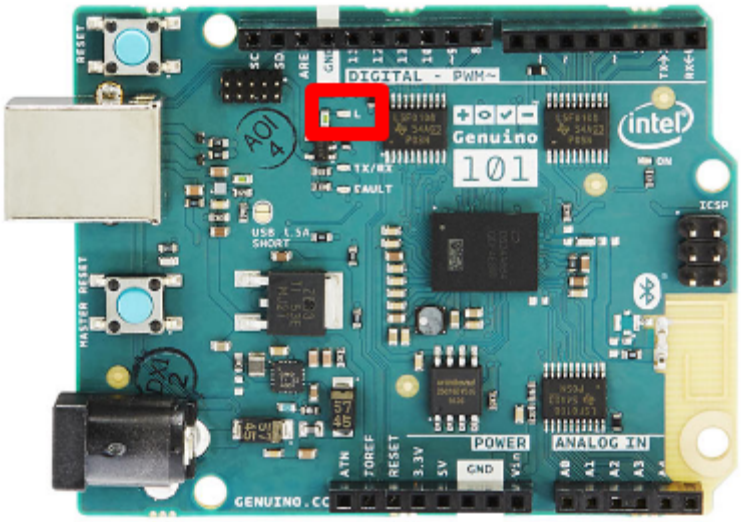
- Click the Deploy button on the top of menu bar to deploy the Node-RED flow.



- The Arduino 101 LED on board LED will start blinking



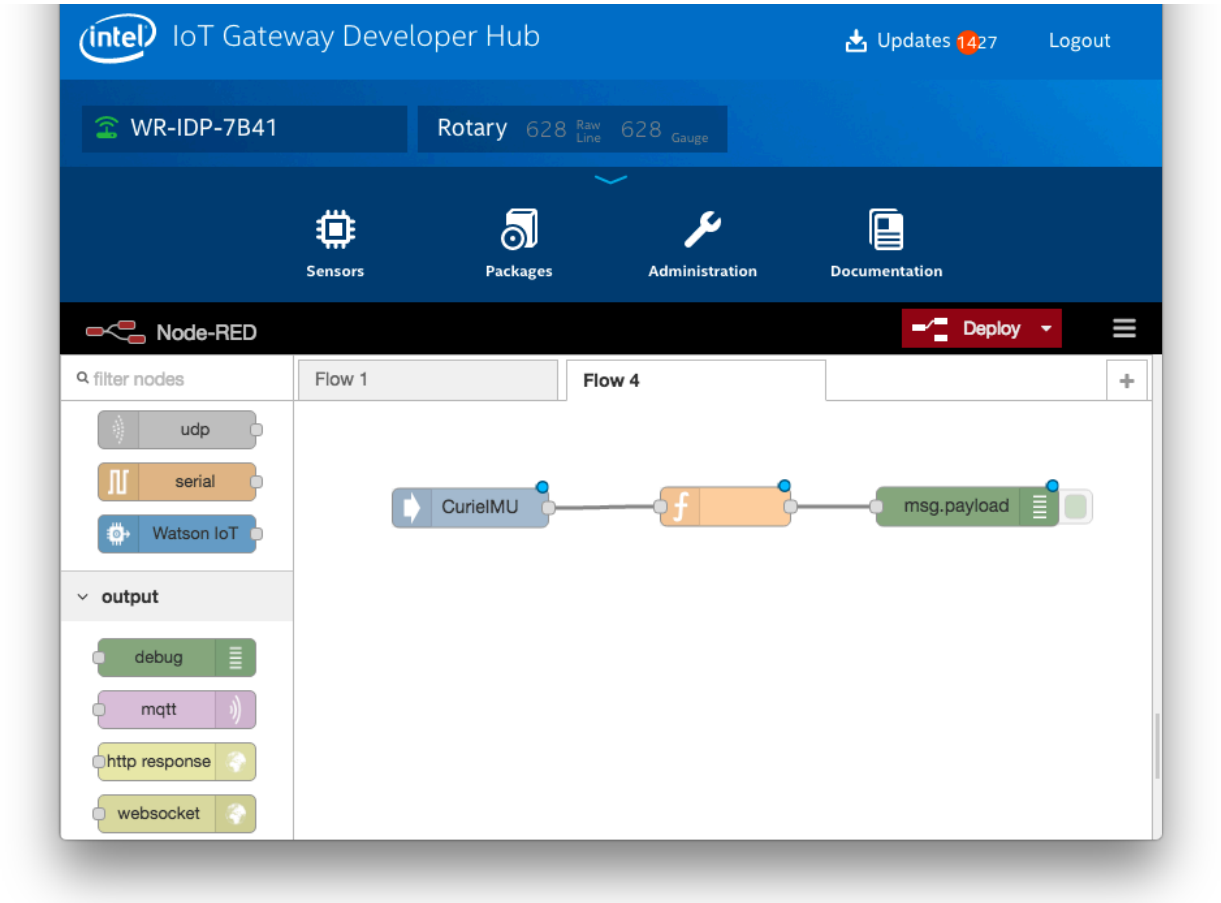
Note: With the Grove Base Shield attached to the Arduino, the LED will be difficult to see. You will need to peek under the shield to see the LED on the Arduino 101 blinking on and off.



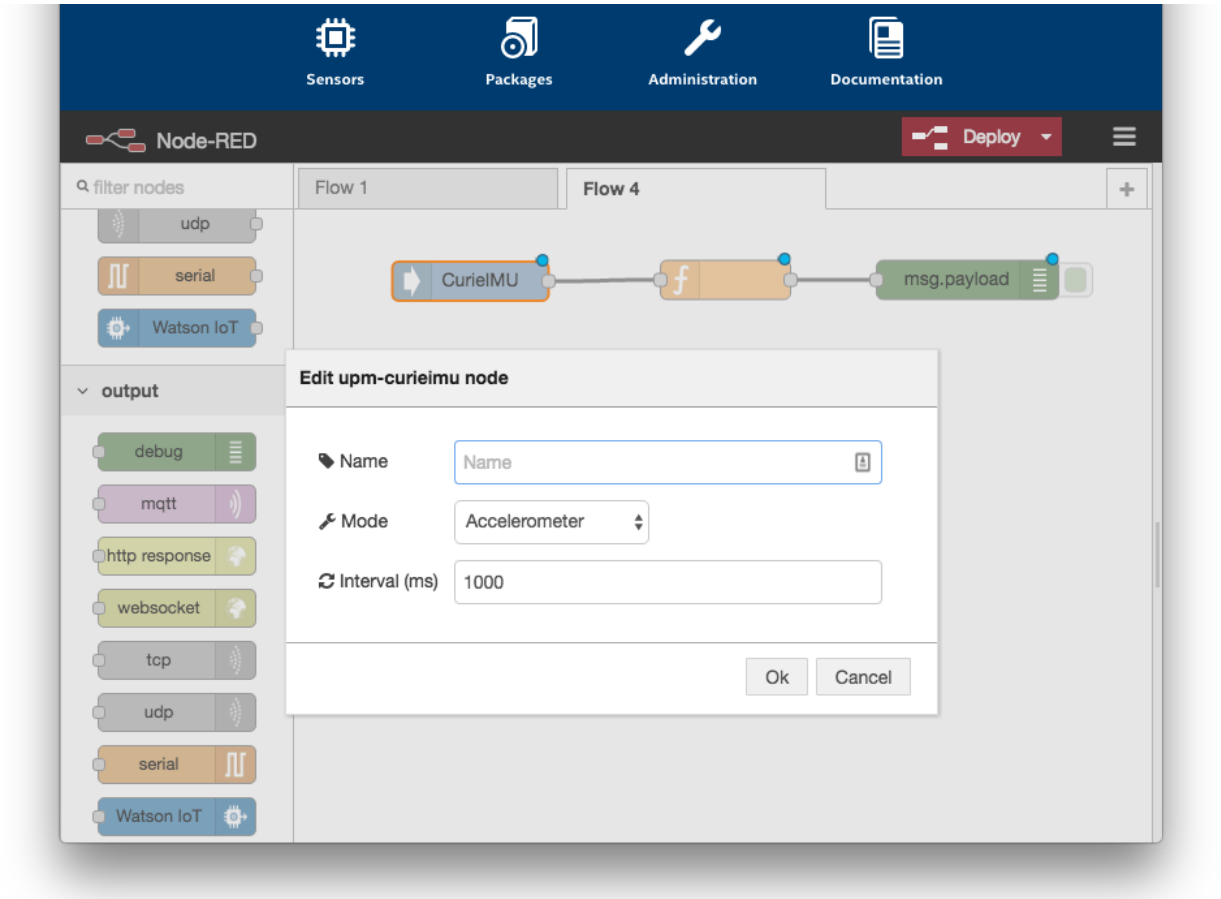
Note: When connecting the Intel Iot Gateway to an Arduino 101, the gateway uses a subplatform called Firmata to communicate GPIO requests to the Arduino 101. Firmata is a generic protocol for communicating with microcontrollers from software on a host computer. It is intended to work with any host computer software package. Basically, this firmware establishes a protocol for talking to the Arduino from the host software.

Adding Sensors and Processing to the Gateway

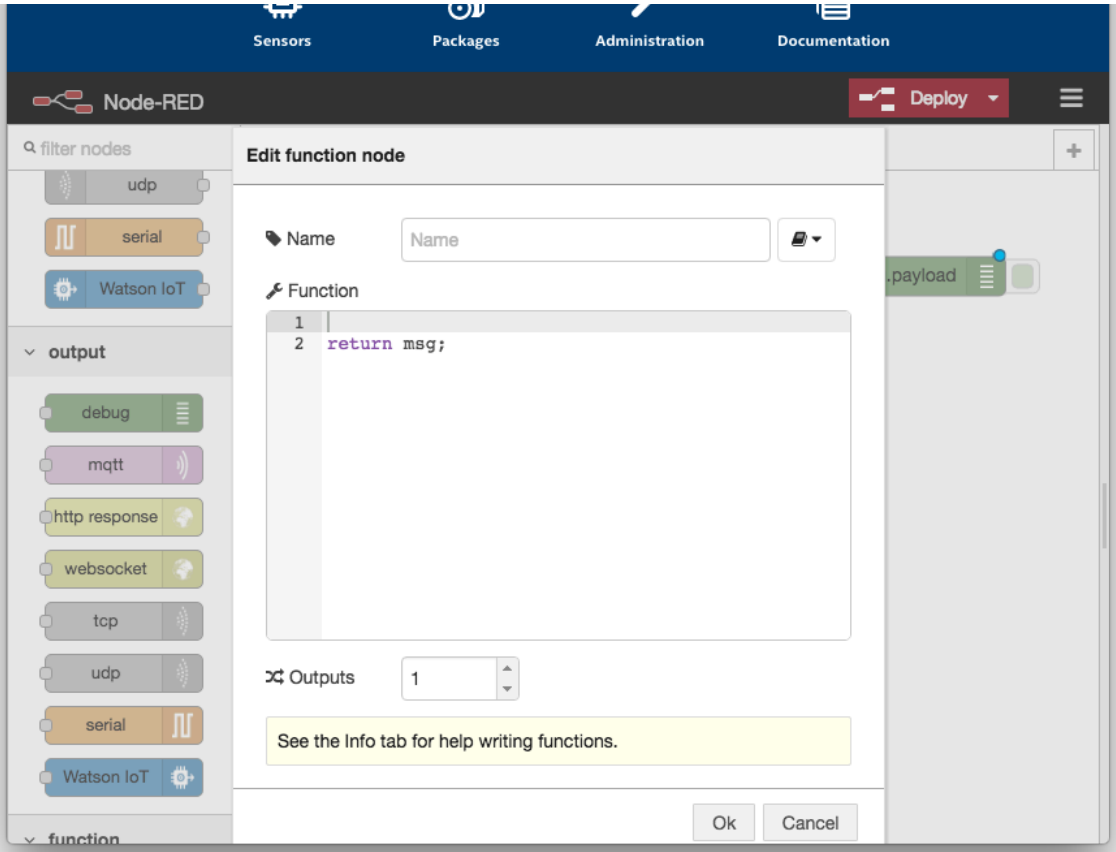
Now that you have a basic understanding of how to interact with the Gateway let's move on to adding some more sensor data and processing to the gateway.



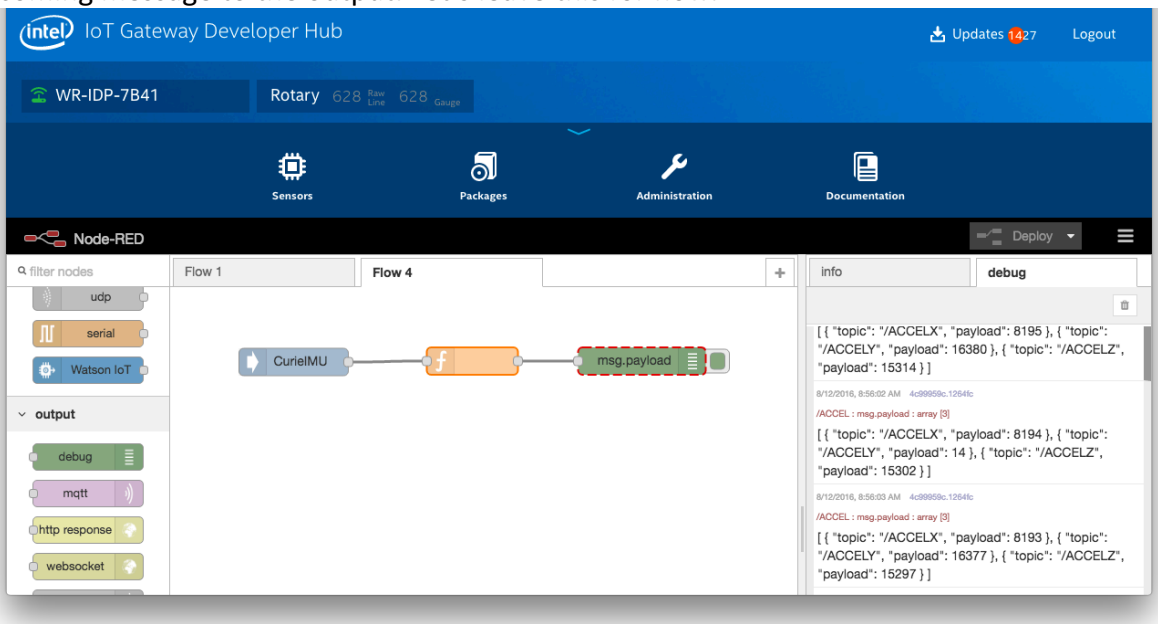
- From **UPM Sensors** add **CurieIMU** node
- From **function** add a **function** node
- From **output** add a **debug** node
- Wire the nodes together



If you double click the **CurieIMU** node you can bring up the configuration interface. Here you can name the node, change the mode from Accelerometer to Gyroscope and set the intervals at which the node will send data. Leave these as defaults for now.



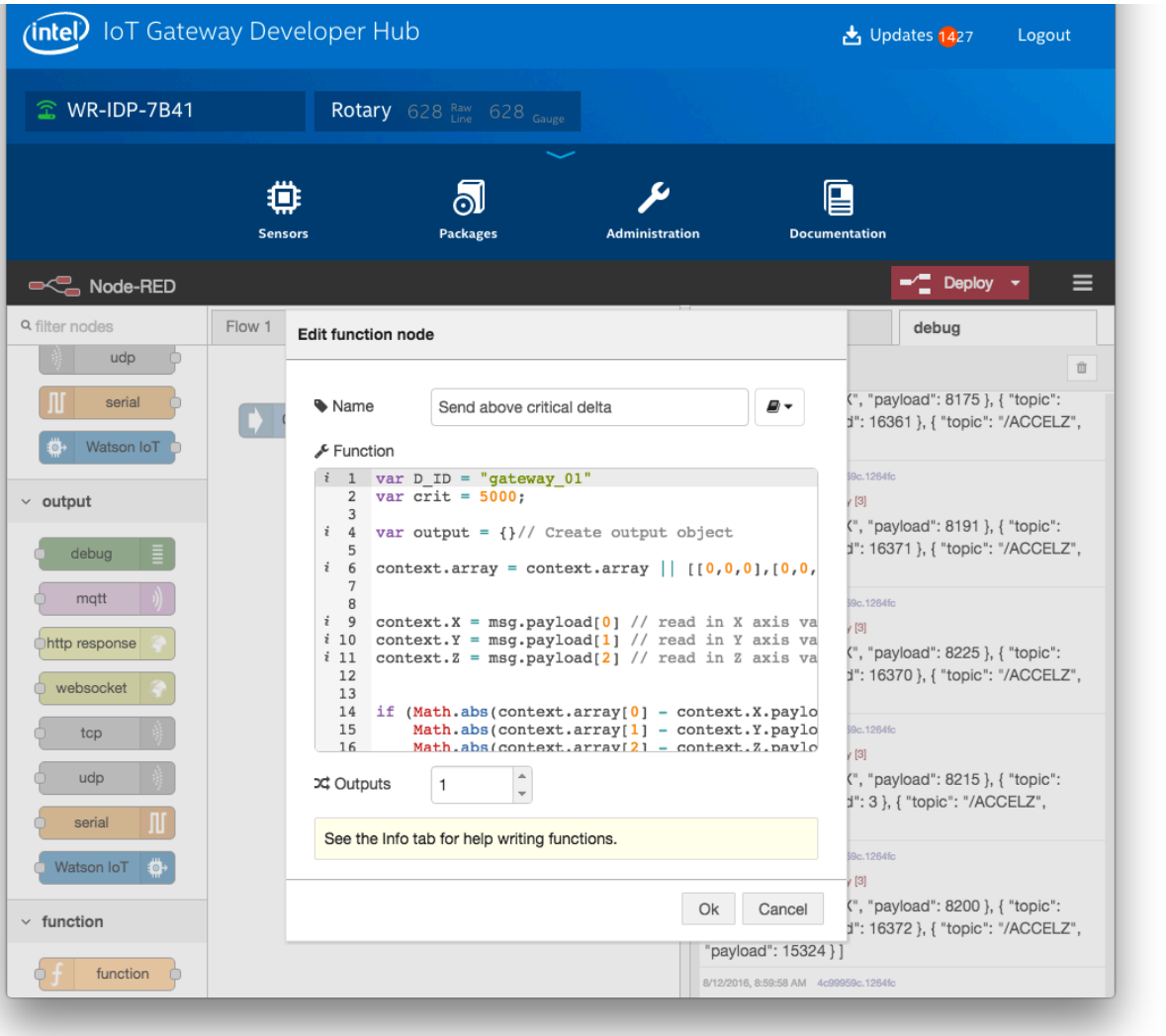
If you double click the function node you will bring up the function interface. Here you can enter JavaScript code that will be executed when the node gets input. By default, it only has “return msg;” which will simply pass the incoming message to the output. Let’s leave this for now.



Hit deploy to run the flow. Open the debug tab to see the output from the function node.
Note: you may have to drag from the right to open the debug tab.

As you can see the CurieIMU reads out the accelerometer data once a second and passes it to the debug output.

If we are sending data up to a cloud service, we probably want to limit it to significant data. Let's add a filter to the function node so that it only passes the accelerometer data to the debug tab if the values between reading exceed some critical value – i.e. the accelerometer is bumped.



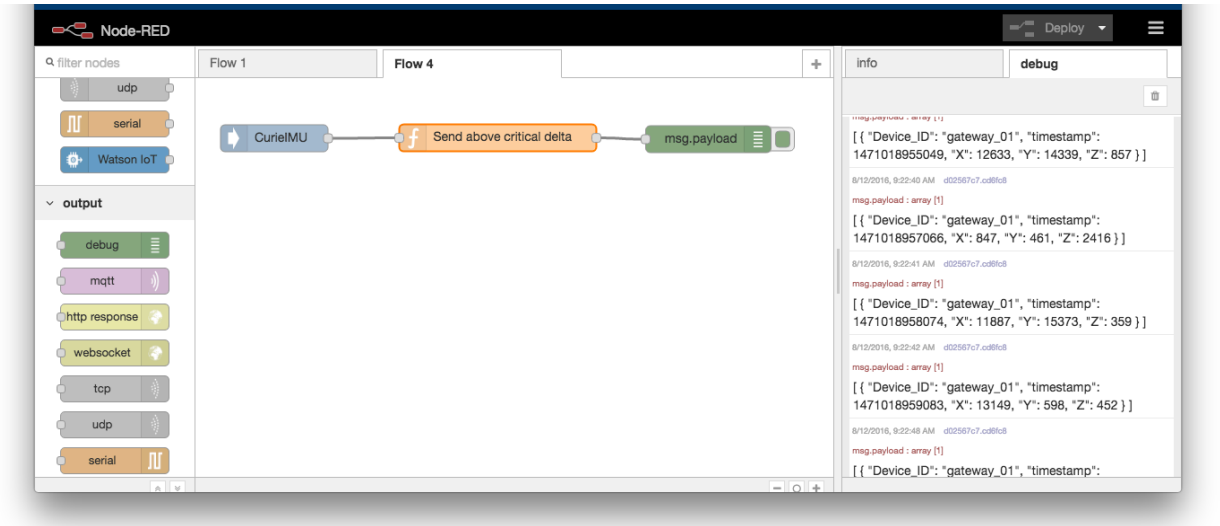
Open the function node again and enter / paste the following code:

```
var D_ID = "gateway_01"
var crit = 5000;

var output = {}
context.array = context.array || [0,0,0]

context.X = msg.payload[0]
context.Y = msg.payload[1]
context.Z = msg.payload[2]

if (Math.abs(context.array[0] - context.X.payload) > crit |
    Math.abs(context.array[1] - context.Y.payload) > crit |
    Math.abs(context.array[2] - context.Z.payload) > crit)
    {
        output.payload = [
            {
                d: {
                    "Device_ID": D_ID,
                    "timestamp" : Date.now(),
                    "X" : context.X.payload,
                    "Y" : context.Y.payload,
                    "Z" : context.Z.payload
                }
            }
        ]
        context.array = [context.X.payload, context.Y.payload, context.
Z.payload]
        return output;
    }
    else {
        context.array = [context.X.payload, context.Y.payload, context.
Z.payload]
```

```
    }
```

Note: The context object is specific to Node-RED. Any variables inside the node are lost every time the code is run (whenever the node gets a message input). In order to preserve the variables across iterations the context object is used. There are also contexts for the entire flow and even across flows. Think of them as global variables. For more info see: http://nodered.org/docs/creating-nodes/context

The code does a few things. First it parses the incoming message into X, Y, and Z accelerometer values. Then it compares the incoming values against the last measured values which are stored in a an array ([0,0,0] at the start). If the delta between values is above the critical value, 5000 in our case, it sends the accelerometer data to the output. The code then overwrites the array with the incoming values to prepare for the next iteration. The code also adds a Device Id and timestamp to the output.



Hit deploy to run the code. The debug output should only display the accelerometer data when the Arduino 101 board registers movement.

Share Node-RED flows across workspaces

You can export and import entire flows (nodes and connections) out of and into your workspace.

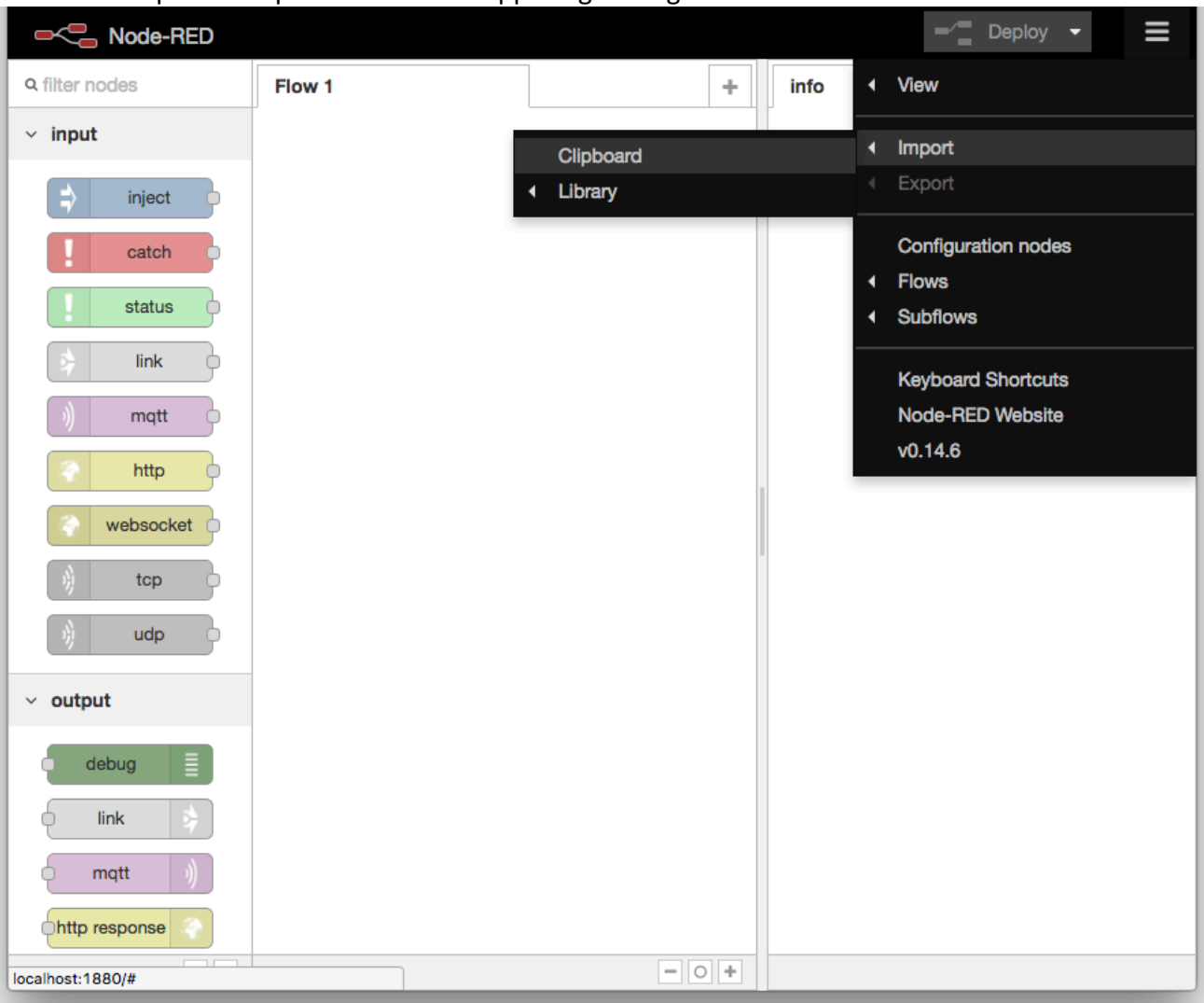
Importing Flows:

To import the above flow go to:

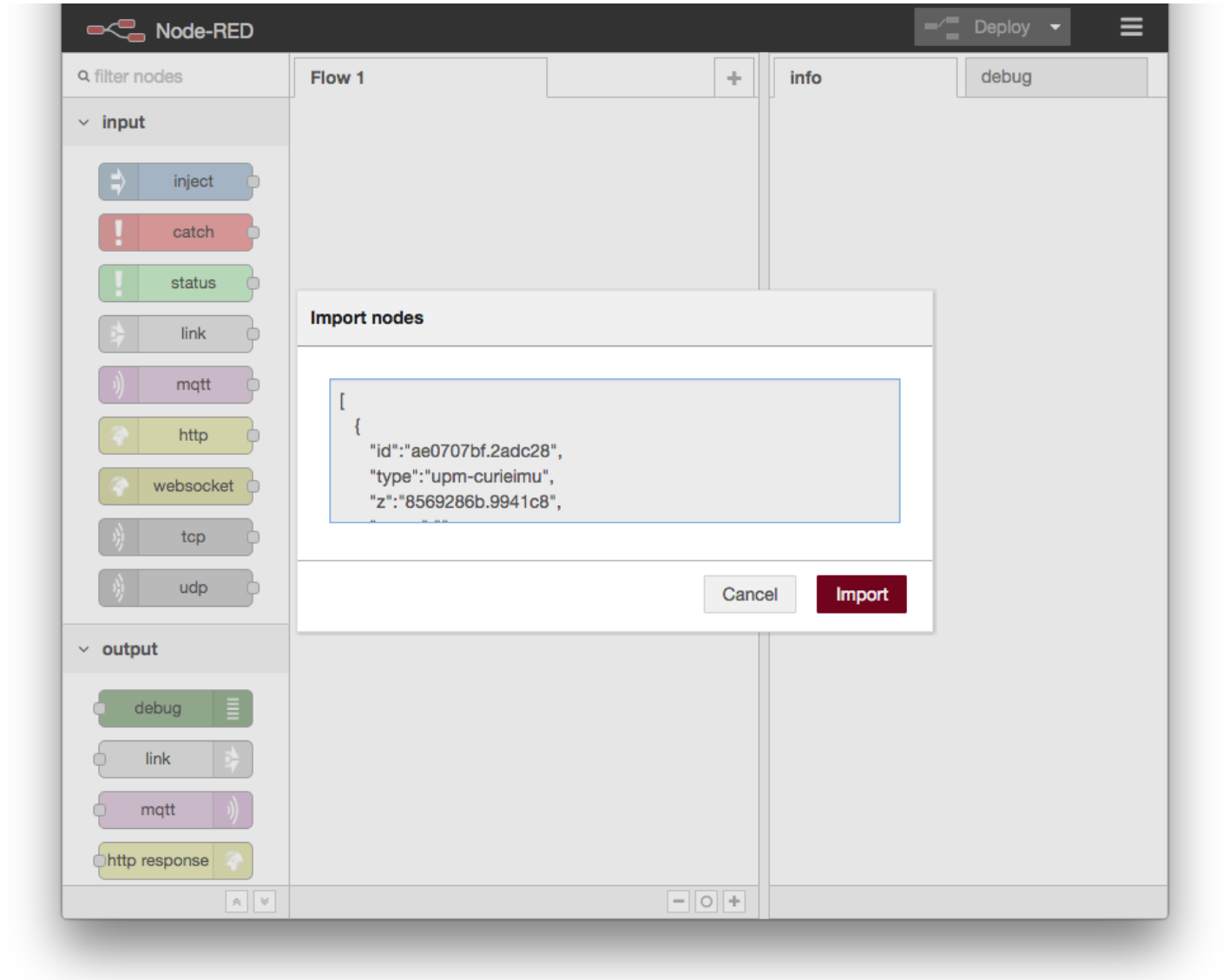
https://github.com/martinkronberg/Intel-AWS-IoT-Workshop/blob/master/read-accel.json

Copy the JSON code to your clipboard (ctrl-c)

Navigate to the Import -> Clipboard from the upper right burger:



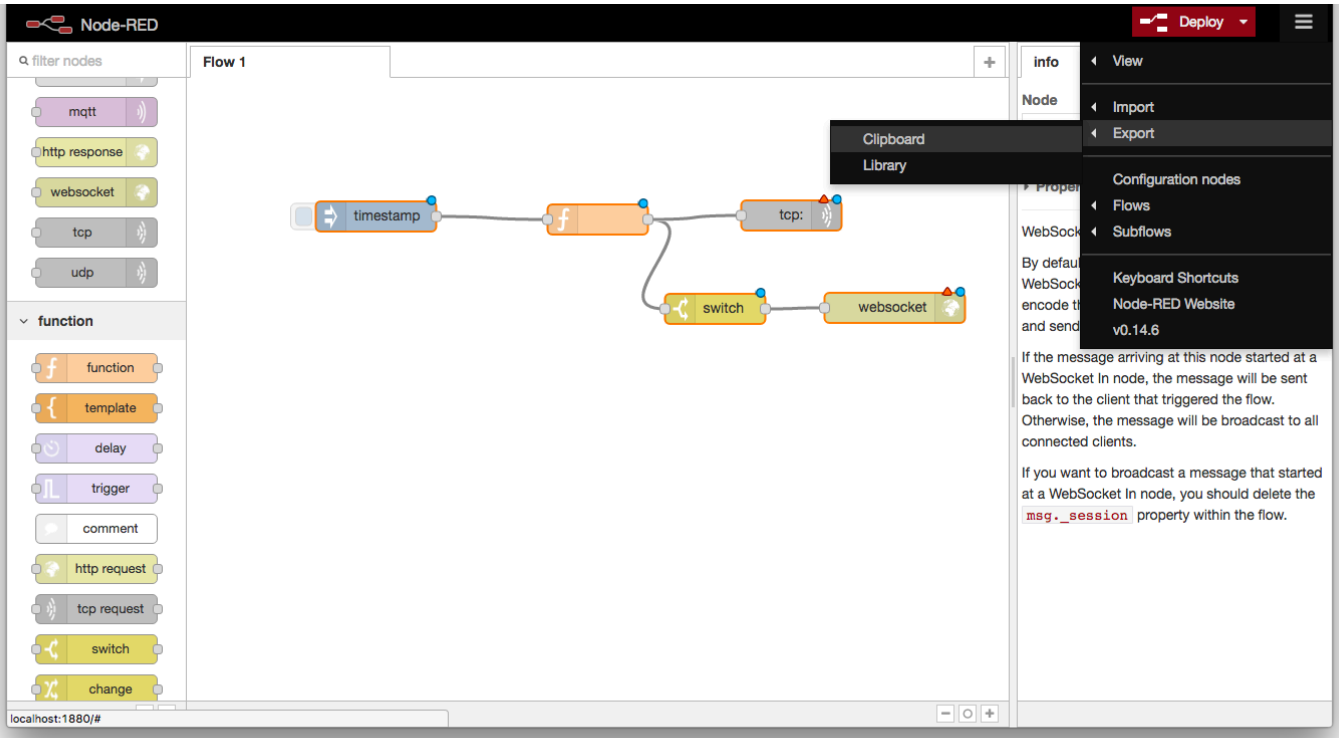
Paste the JSON from your clipboard (ctrl-v) and clip import:



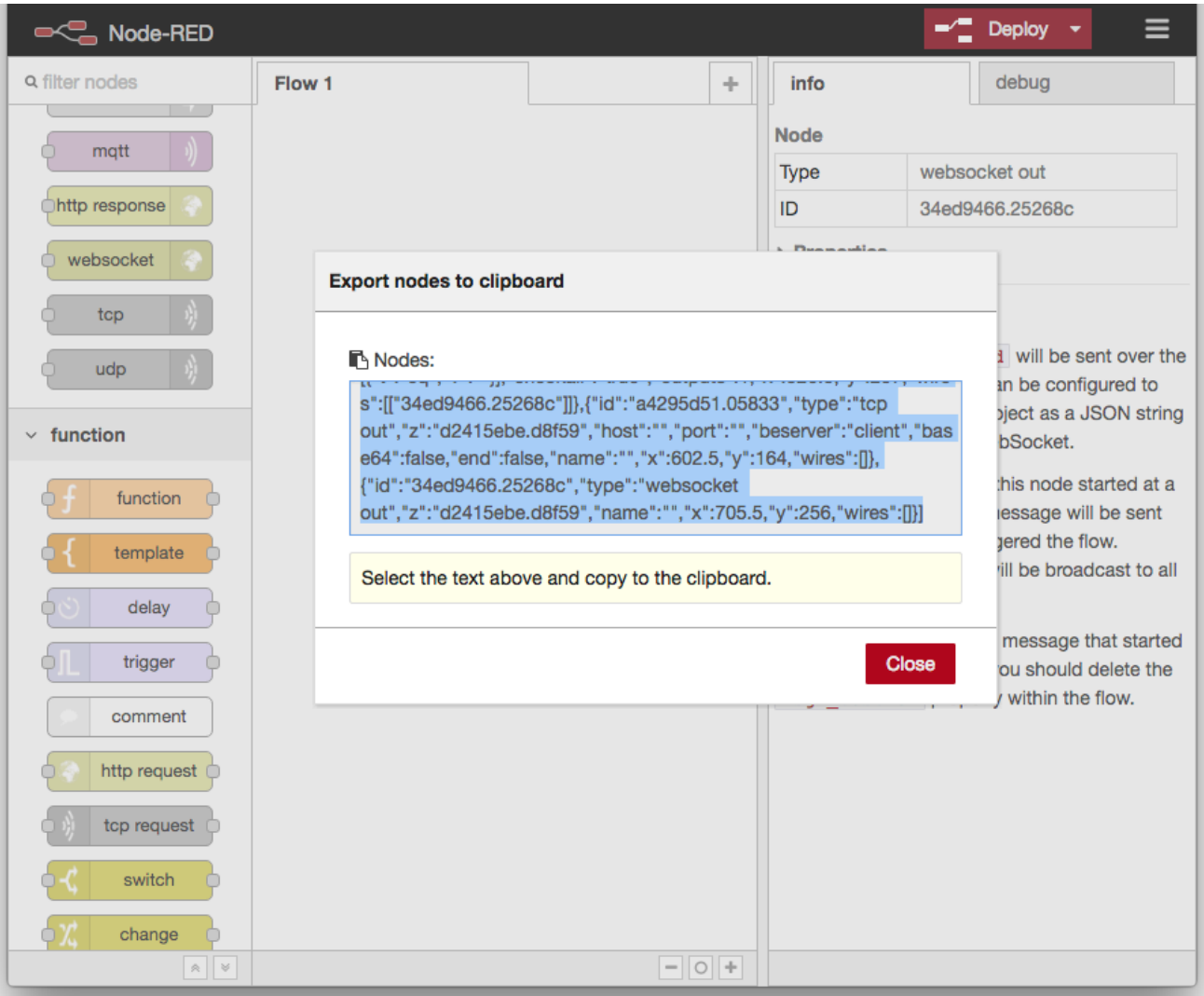
You will now have the entire flow inside your workspace. Follow the same procedure for importing anymore code you come across in this workshop.

Exporting Flows

In order to export a flow select the nodes you wish to export, navigate to Export -> Clipboard from the burger



Copy the text in the window that opens. You can now paste this somewhere and share it with others. I would recommend running it through a json beautifier like: https://jsonformatter.curiousconcept.com/ first if you want it to be readable.



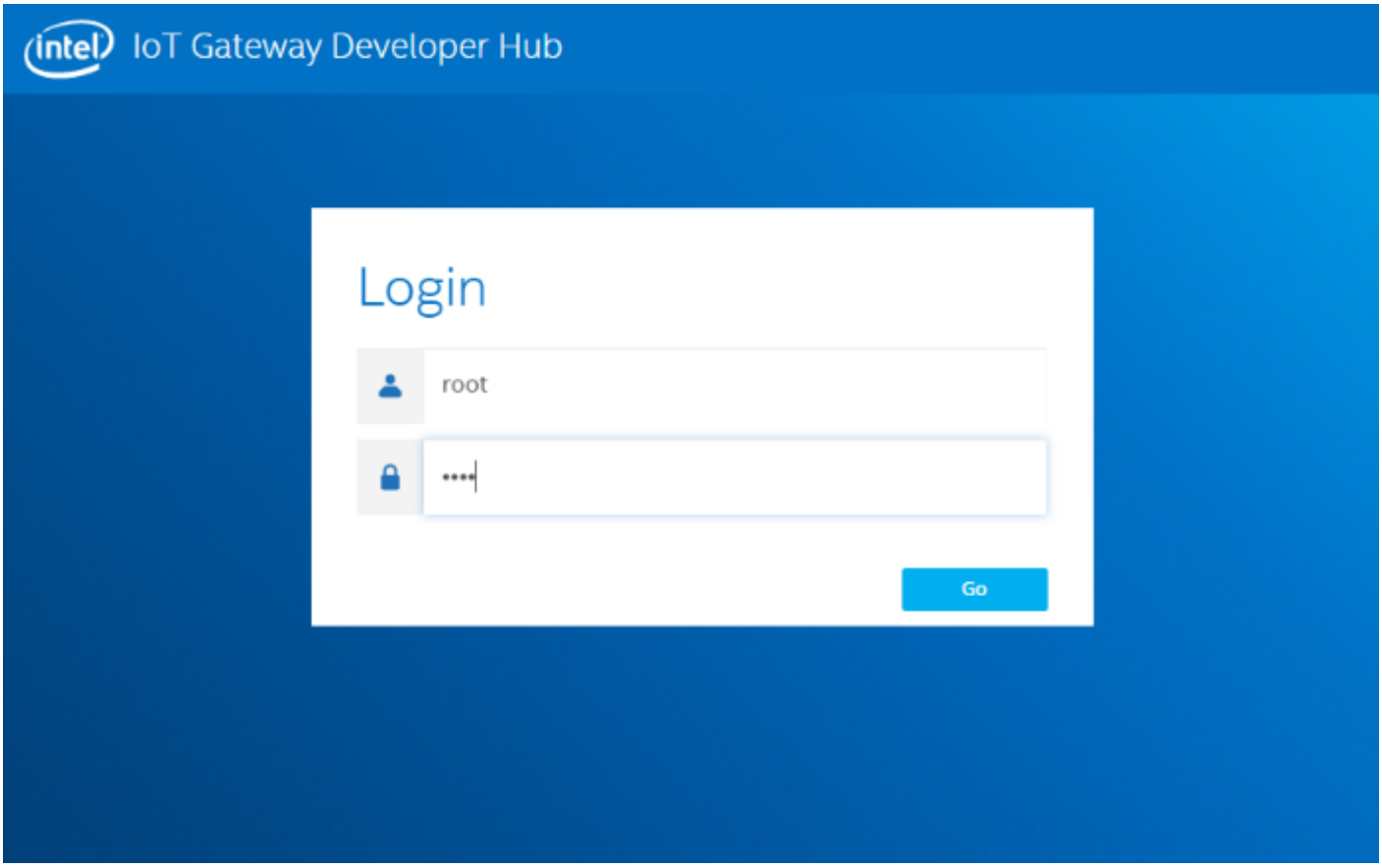
Add AWS IoT Nodes and OS Monitoring

1. Access the console on your gateway using either a monitor or keyboard connected directly, or SSH (recommended).

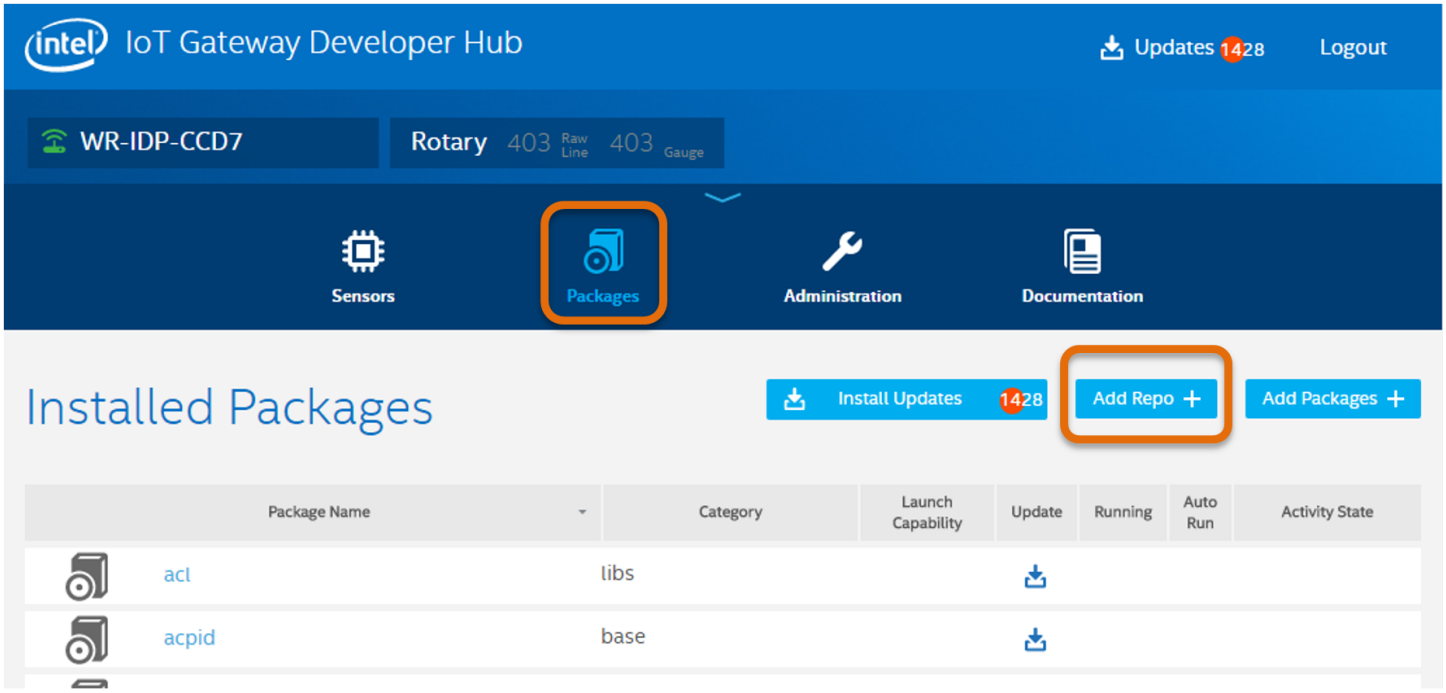
2. Add the GPG key for the cloud repository using the following command:
rpm --import http://iotdk.intel.com/misc/iot_pub.key

3. On your development device (e.g., laptop) open a web browser and load the IoT Gateway Developer Hub interface by entering the IP address of your gateway in the address bar.
Tip: You can find your gateway's IP address using the ifconfig command.

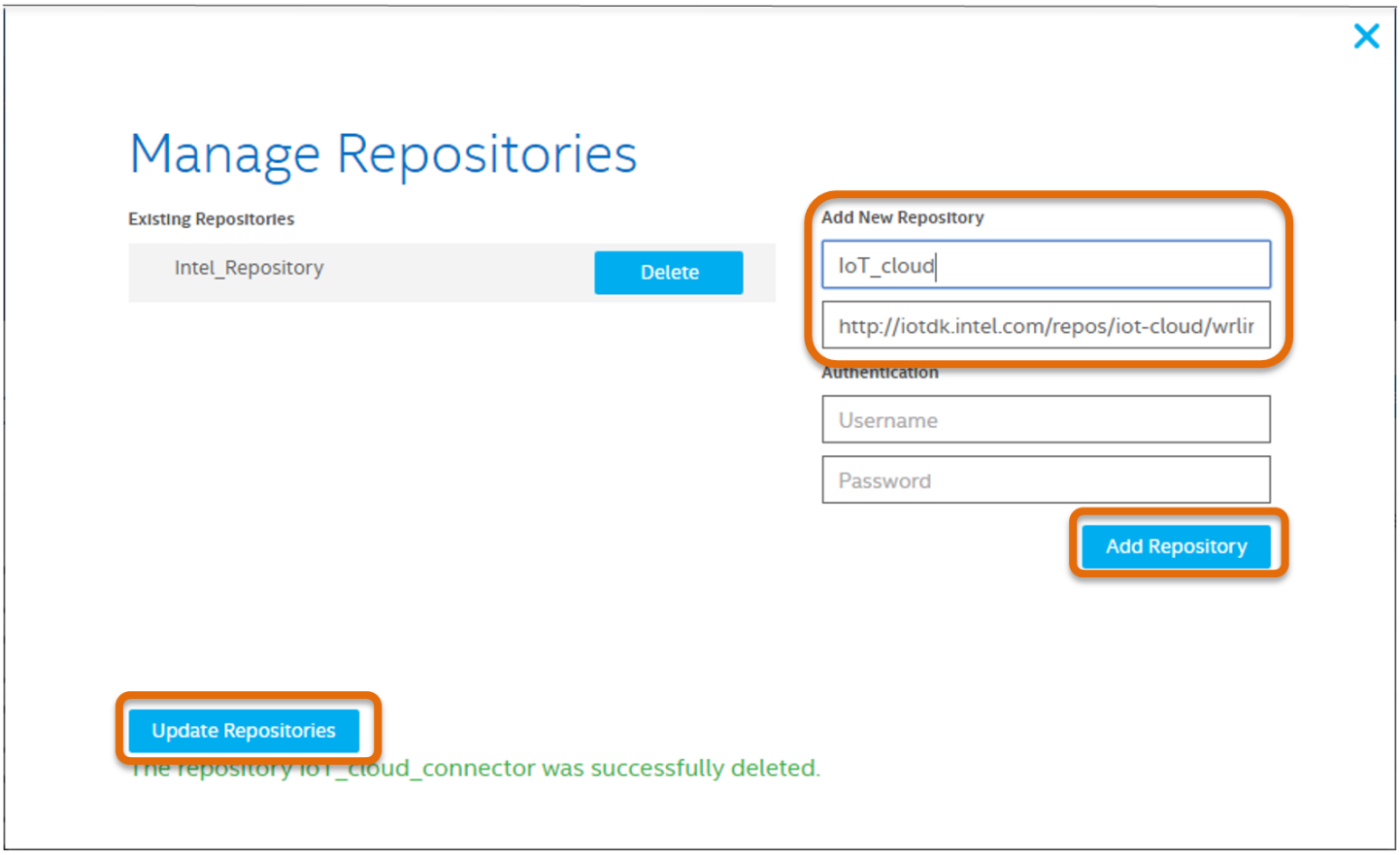
4. Login to the IoT Gateway Developer Hub interface using your credentials. The default login and password are both **root**.



5. Add the **IoT Cloud** repository.



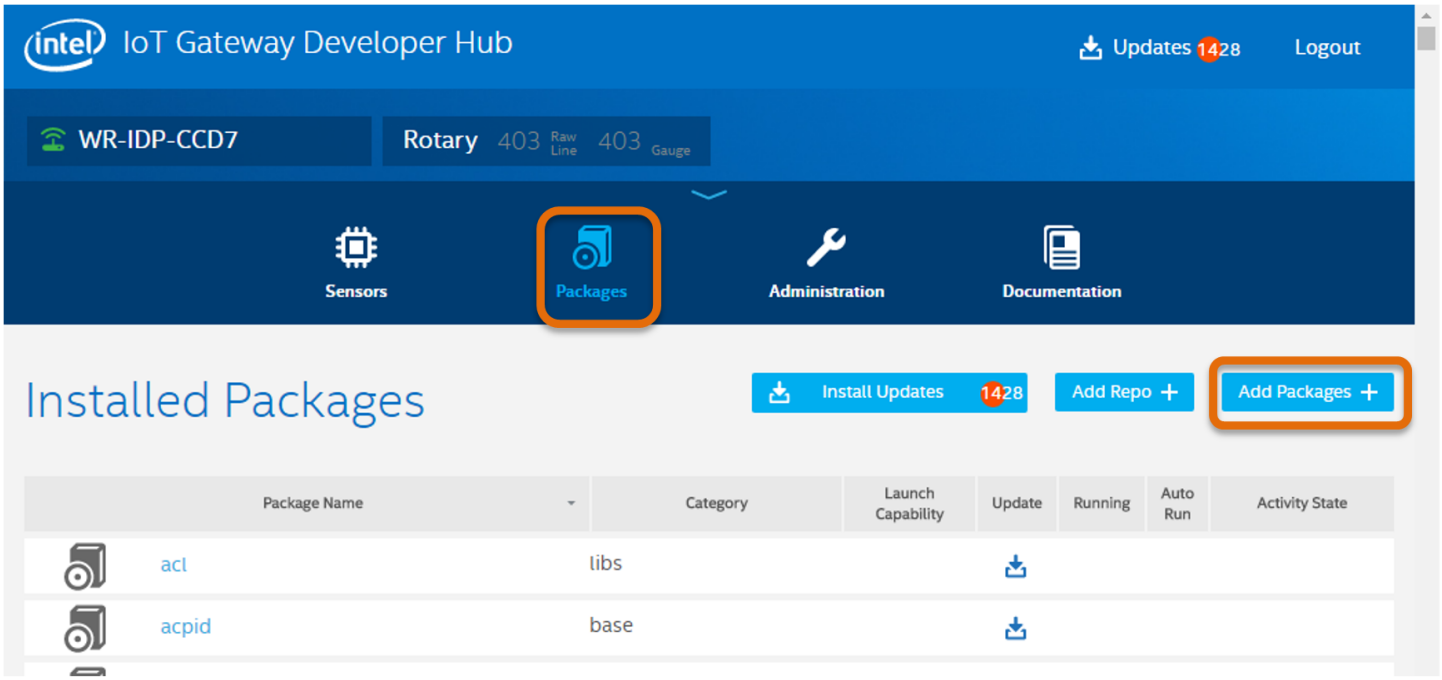
6. Go to the **Packages** section and click the **Add Repo +** button.



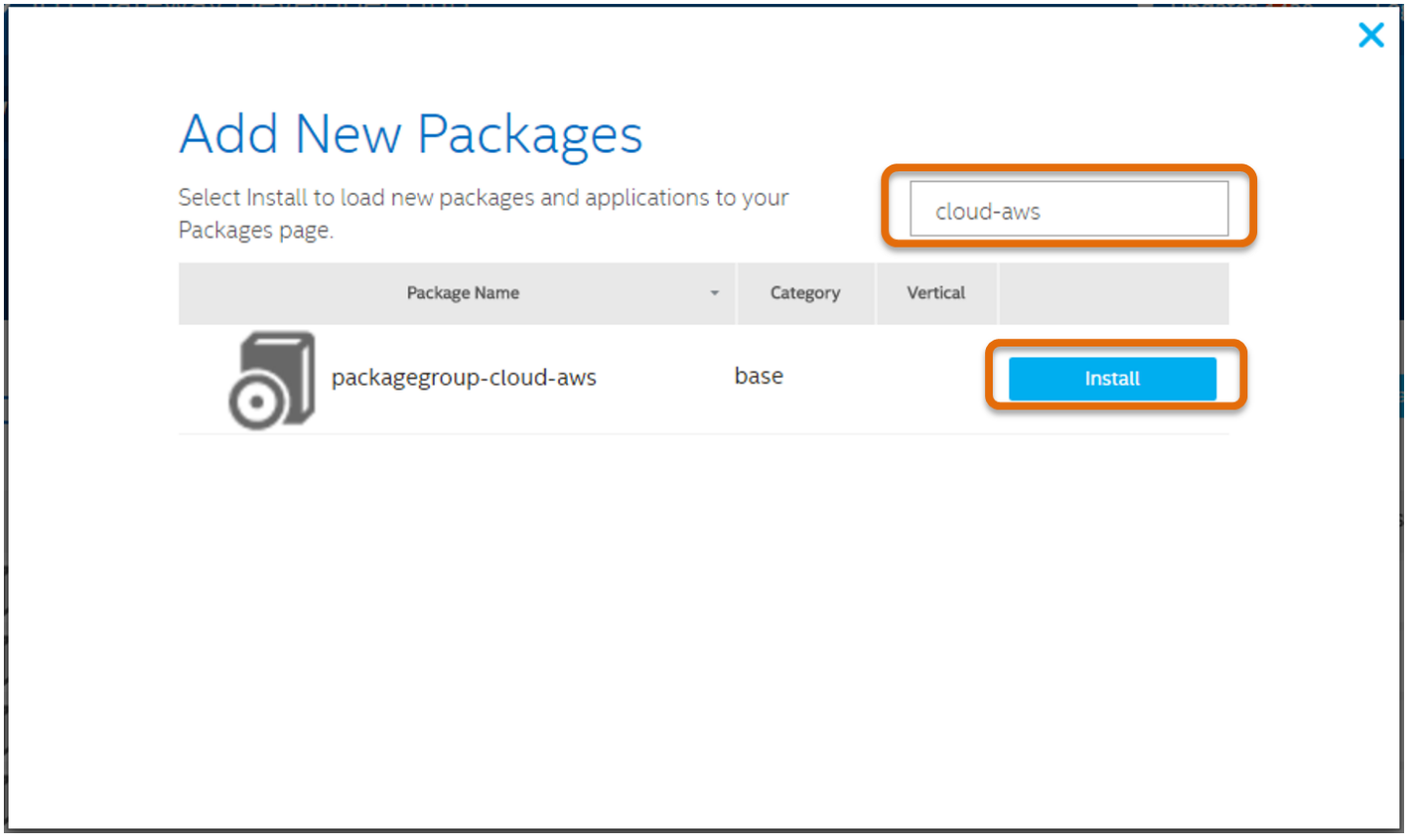
7. Populate the fields with the following information and click **Add Repository**:

Name: IoT_Cloud
URL: http://iotdk.intel.com/repos/iot-cloud/wrlinux7/rcpl13
8. Finally, click the **Update Repositories** button to update the package list.

Adding AWS* support to your Intel® IoT Gateway



1. Click the **Add Packages +** button to bring up the list of packages you can install.



Search for **cloud-aws** using the search box at the top of the package window. Click the **Install** button next to the **packagegroup-cloud-aws** entry.

2. Back in your ssh connection, run the following command to restart the Node-RED environment on the NUC. This is necessary because the package that we just installed updated the resources available to Node-RED so it needs to be re-initialized:

```
$systemctl restart node-red-experience
```

What Next?

To explore some advanced projects that you can build with the gateway make sure you have the Intel XDK IoT Edition up and running and check out:

Smart Home Path to Product:

https://software.intel.com/en-us/articles/iot-path-to-product-how-to-build-the-smart-home-prototype

Smart Vending Machine Path to Product:

https://software.intel.com/en-us/articles/iot-path-to-product-how-to-build-an-intelligent-vending-machine

Connected Transportation Path to Product:

https://software.intel.com/en-us/articles/iot-path-to-product-how-to-build-a-connected-transportation-solution

A collection of How-To IoT Projects:

https://software.intel.com/en-us/blogs/2015/11/04/announcing-18-new-how-to-intel-iot-code-samples