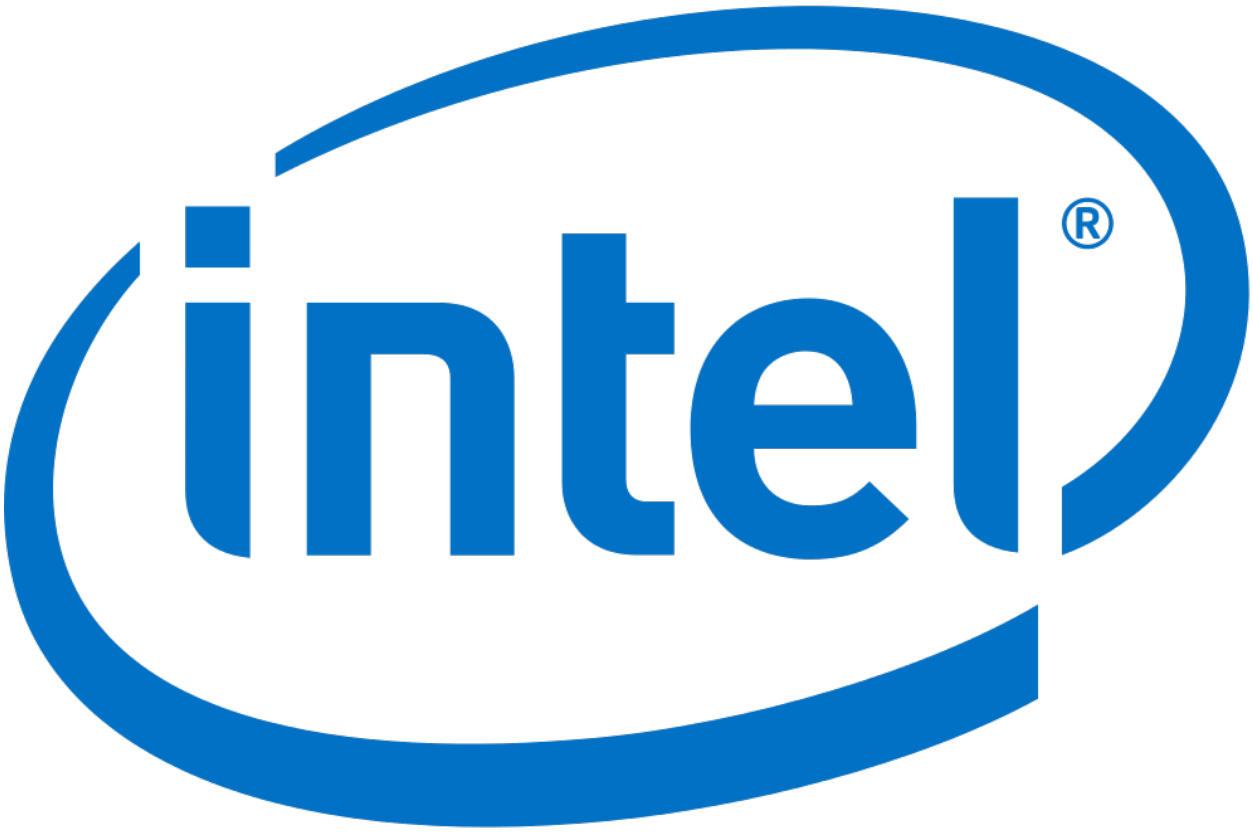
Intel IoT Gateway / Arduino 101 with Watson IoT

Hands-On Workshop

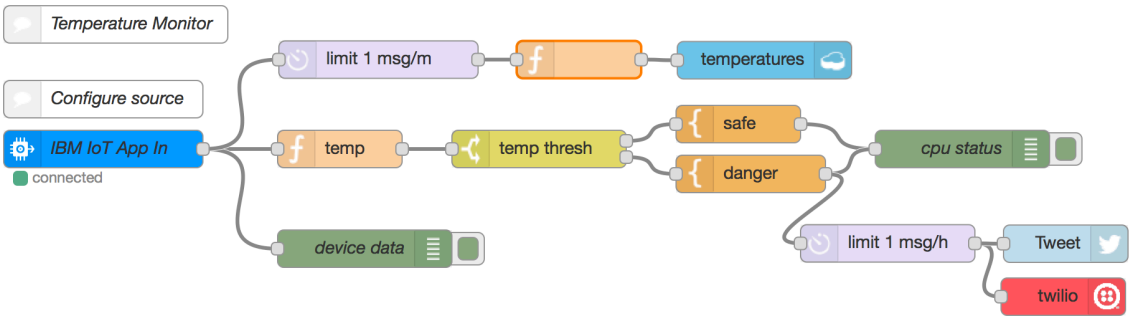
Authors:

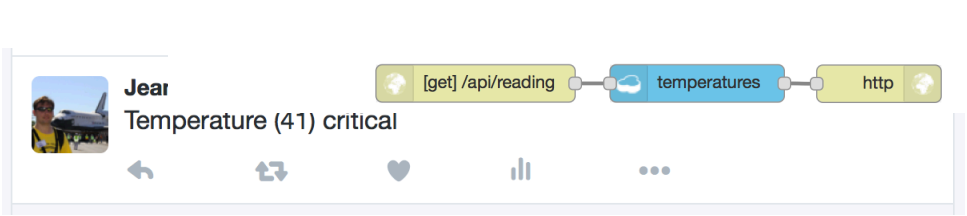
Martin Kronberg	martin.kronberg@intel.com	@martinkronberg
Vaghesh Patel	vaghesh.patel@intel.com	@vaghesh
JeanCarl Bisson	jbisson@us.ibm.com	@dothewww
John Walicki	walicki@us.ibm.com	@johnwalicki



Topics covered: Intel IoT Gateway, Grove sensors, Node-RED, Watson Internet of Things Platform, Cloudant NoSQL Database, Twitter, Twilio, and Weather Company Data for IBM Bluemix.

In this lab, we will unbox and set up an Intel IoT Gateway and the Arduino 101 board (with a Grove Starter kit) along with several services available in IBM Bluemix to monitor the temperature and alert maintenance of a high temperature. Using Node-RED, running on the Intel NUC Gateway, the application will read the temperature value from a Grove temperature sensor. If the temperature rises, a LED light is turned on. The temperature will be sent via Watson's Internet of Things Platform service to a Node-RED application hosted on IBM Bluemix. If the temperature reaches 40°C, we'll send a tweet via Twitter and a text message via Twilio. We will retrieve outside weather data and display it on a LCD screen connected to the Arduino 101. We'll store the temperature in a Cloudant NoSQL database so we can track patterns. Finally, we'll create an HTTP endpoint that exposes the historical temperatures that third-party applications could consume and perform analysis or other fun stuff.





[{"_id":"bf263d883e2946bccf426eff410d76cb","_rev":"1-b91cdf81c2ea48efaf8bdeff95213691","time":1470947934516,"temp":29},
{"_id":"bf263d883e2946bccf426eff41258098","_rev":"1-a8a28b2abd28dfc8e1d50cee12fa0cc6","time":1470947995007,"temp":29},
{"_id":"d346612f6fabfd5359759c7d889098e4","_rev":"1-d15dbcef08e6f098664e85248c493054","time":1470948055512,"temp":29},
{"_id":"d346612f6fabfd5359759c7d88a71d25","_rev":"1-0358ac7df1770c79c6acaf102bcbb24e","time":1470948116011,"temp":29}]

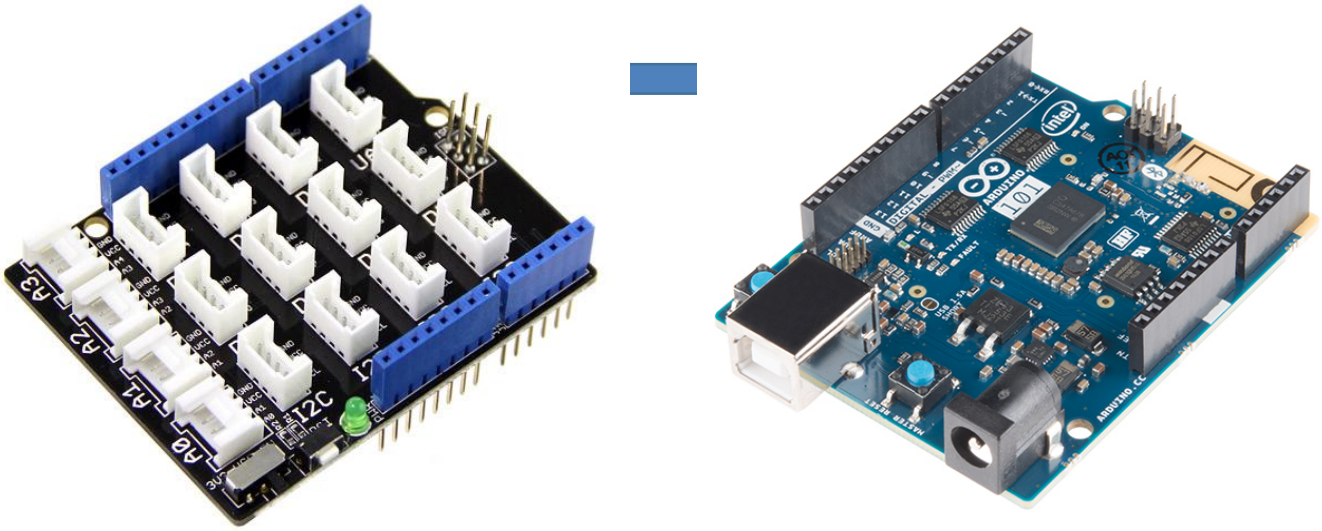
Getting Started with Grove IoT Commercial Developer Kit

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

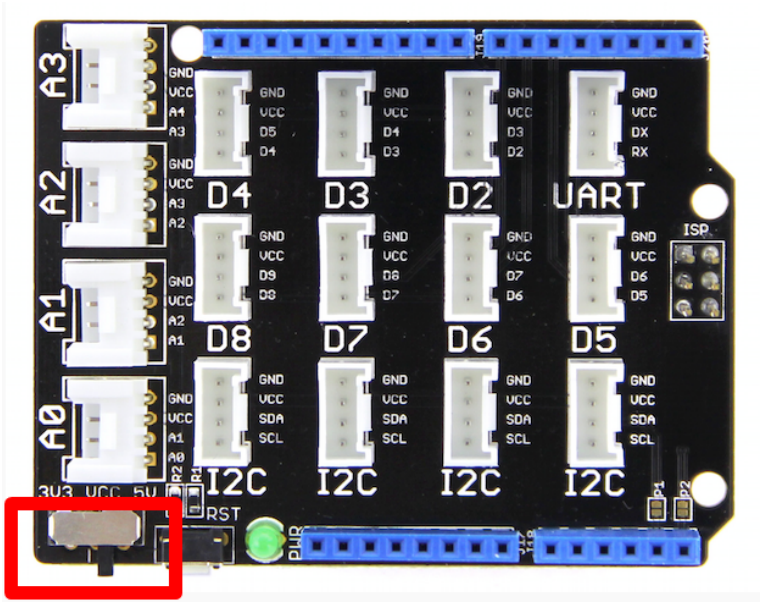
Arduino 101 is not included in Grove IoT Commercial Development Kit and will be given separately.

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.

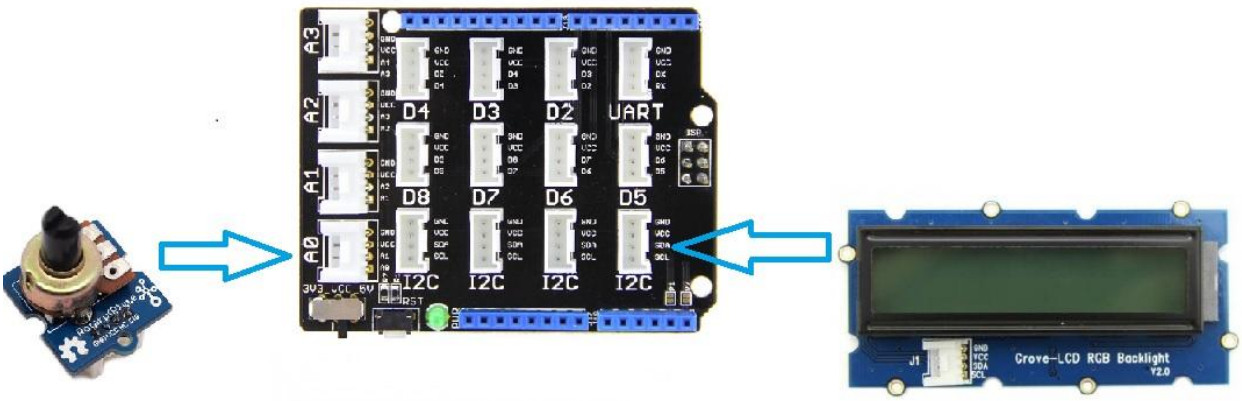


Step 3 – Move Base Shield switch towards 5 Volt



Step 4 – Connect LCD and Rotary Angle to Arduino 101

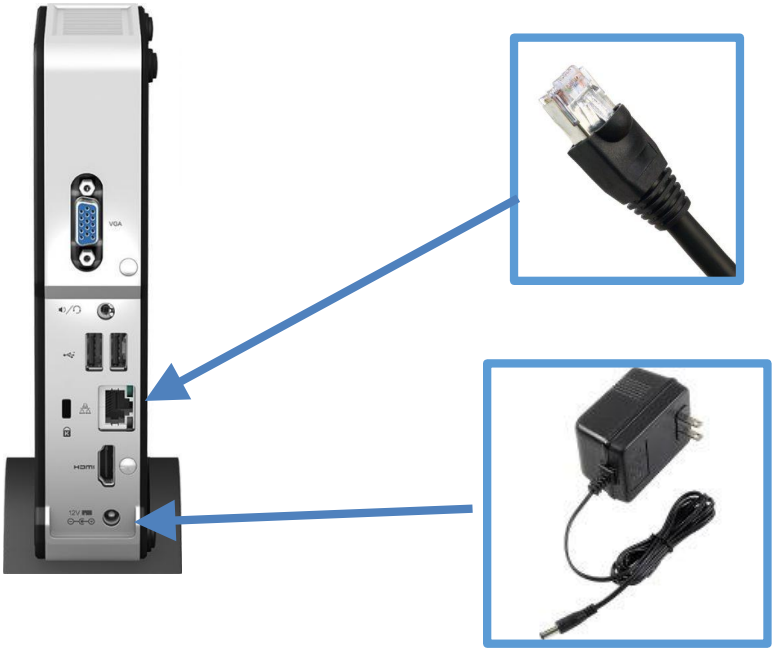
Connect LCD to any I2C pin of base shield and rotary angle to A0 pin of base shield with the help of a USB cable as shown in below image.



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

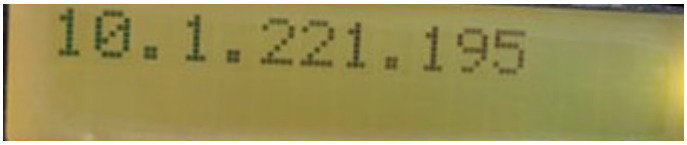
Now connect Arduino 101 with Intel IoT Gateway via USB A-B cable.

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

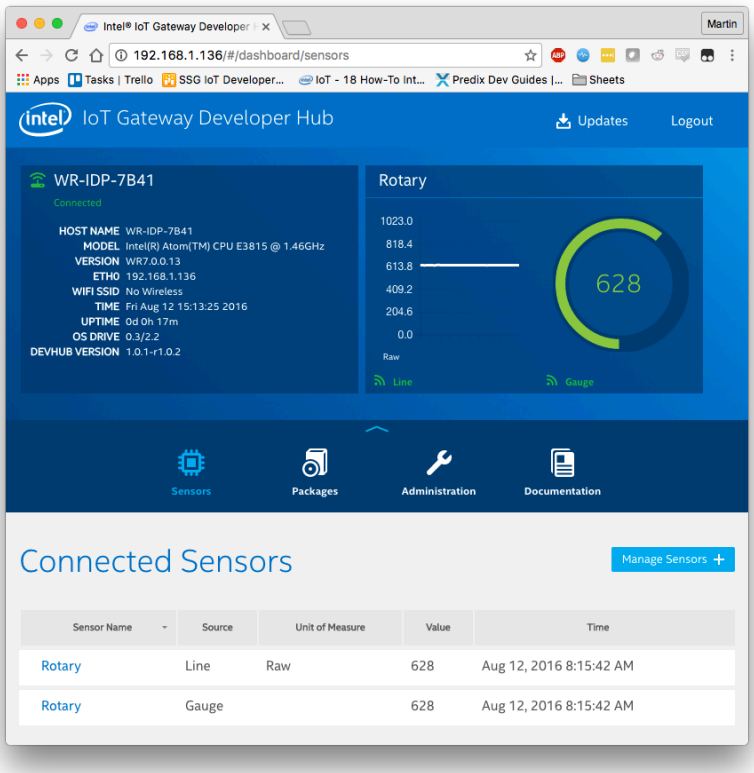
- Login to the Intel® IoT Gateway Developer Hub using root for both the username and the password.
- Use the Intel® IoT Gateway Developer Hub to explore and extend the basic capabilities of the Gateway.

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

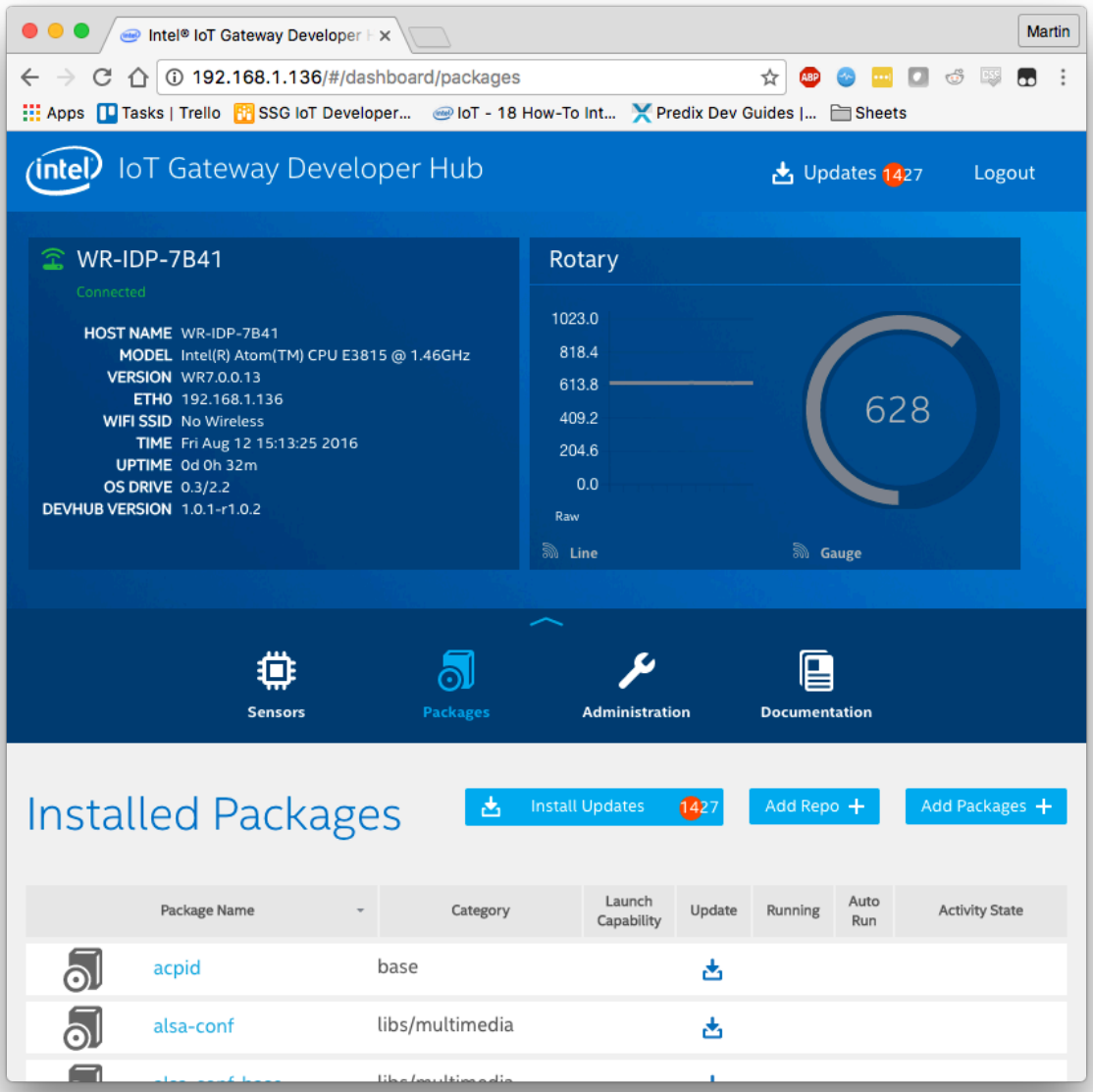
Dashboard



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.

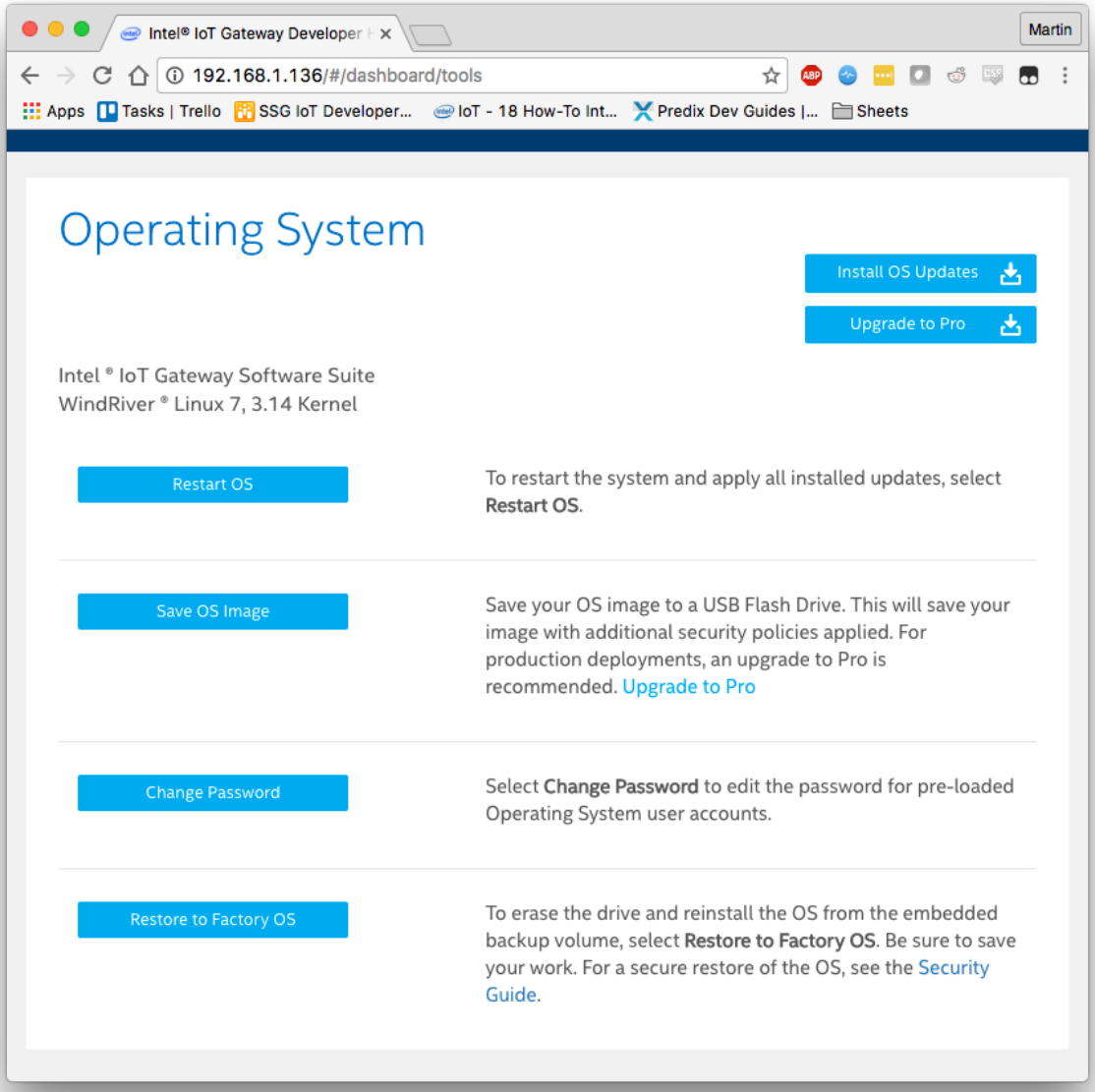
Package Manager



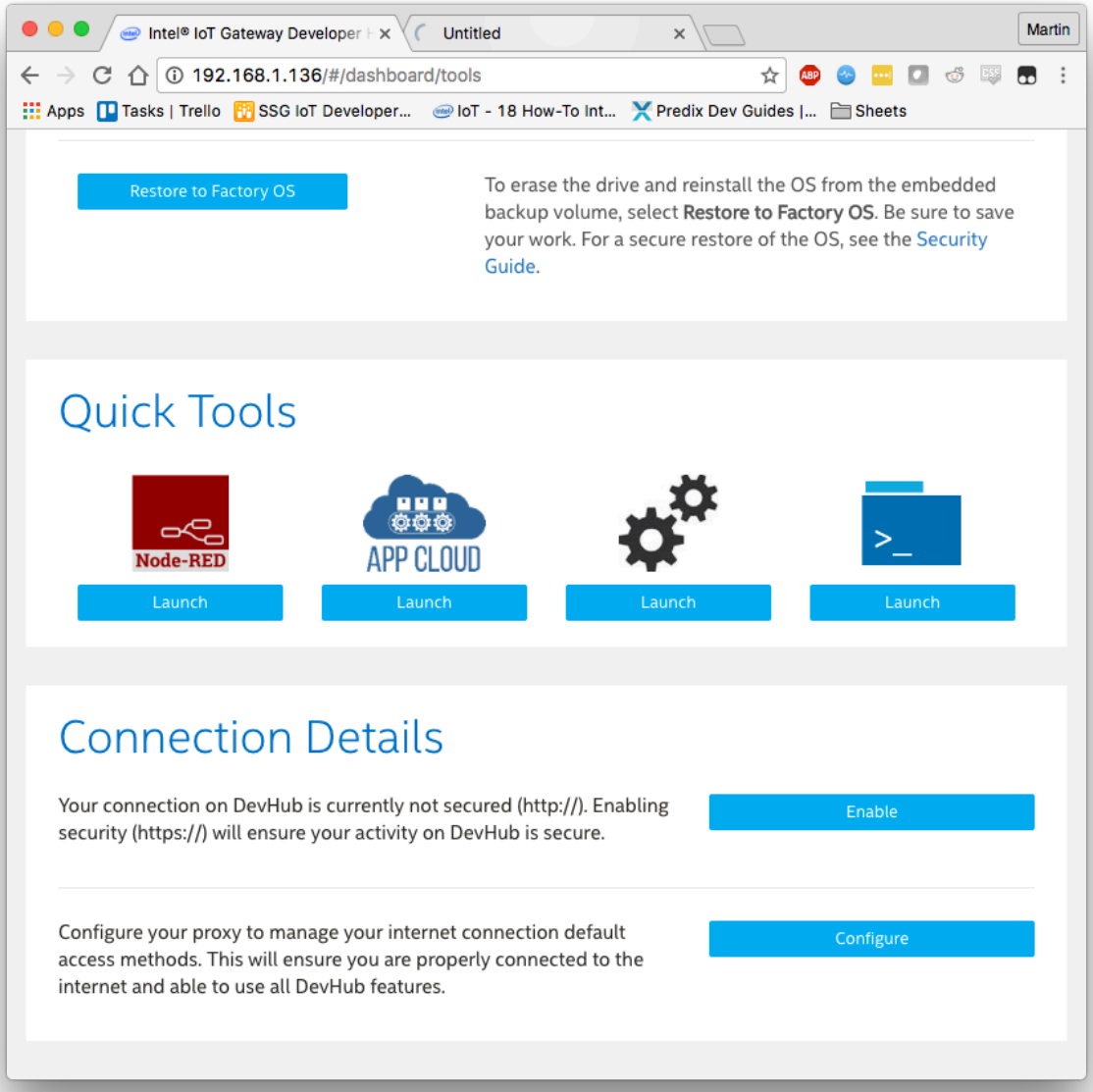
View of Packages screen

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

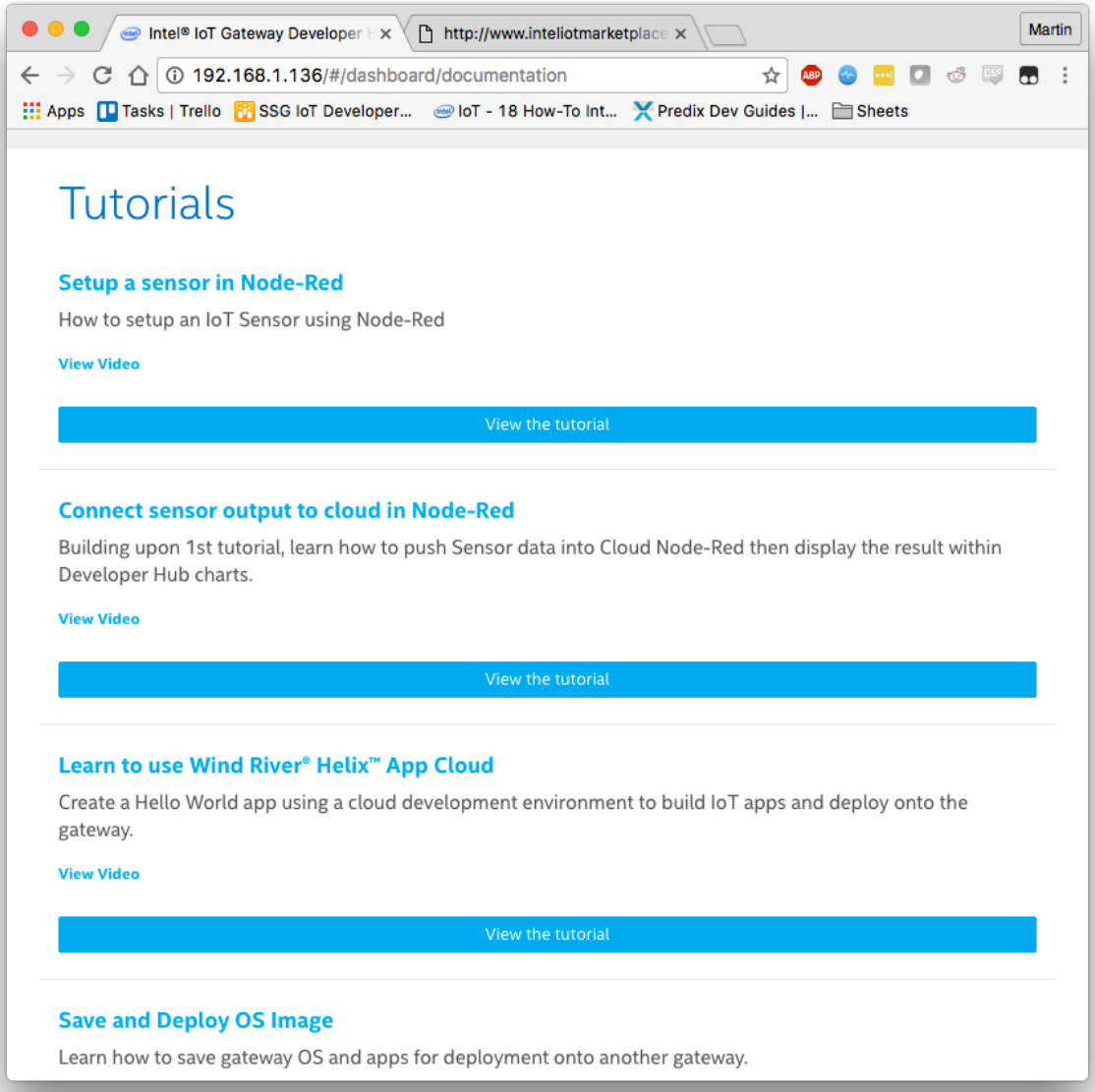


In the Administration interface you will see options to Update the OS, Upgrade to Pro, as well as others.
Please do not select any of these options!



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.

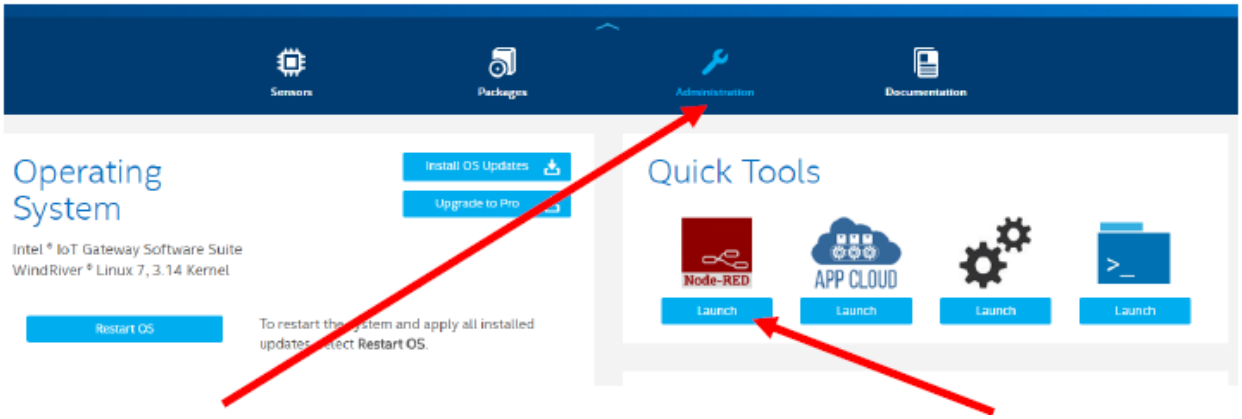
- Double-click putty.exe on your computer to launch PuTTY.
- Enter IP address of the Gateway
- You can login with the username **root** and the password **root**.

For Mace and Linux Users:

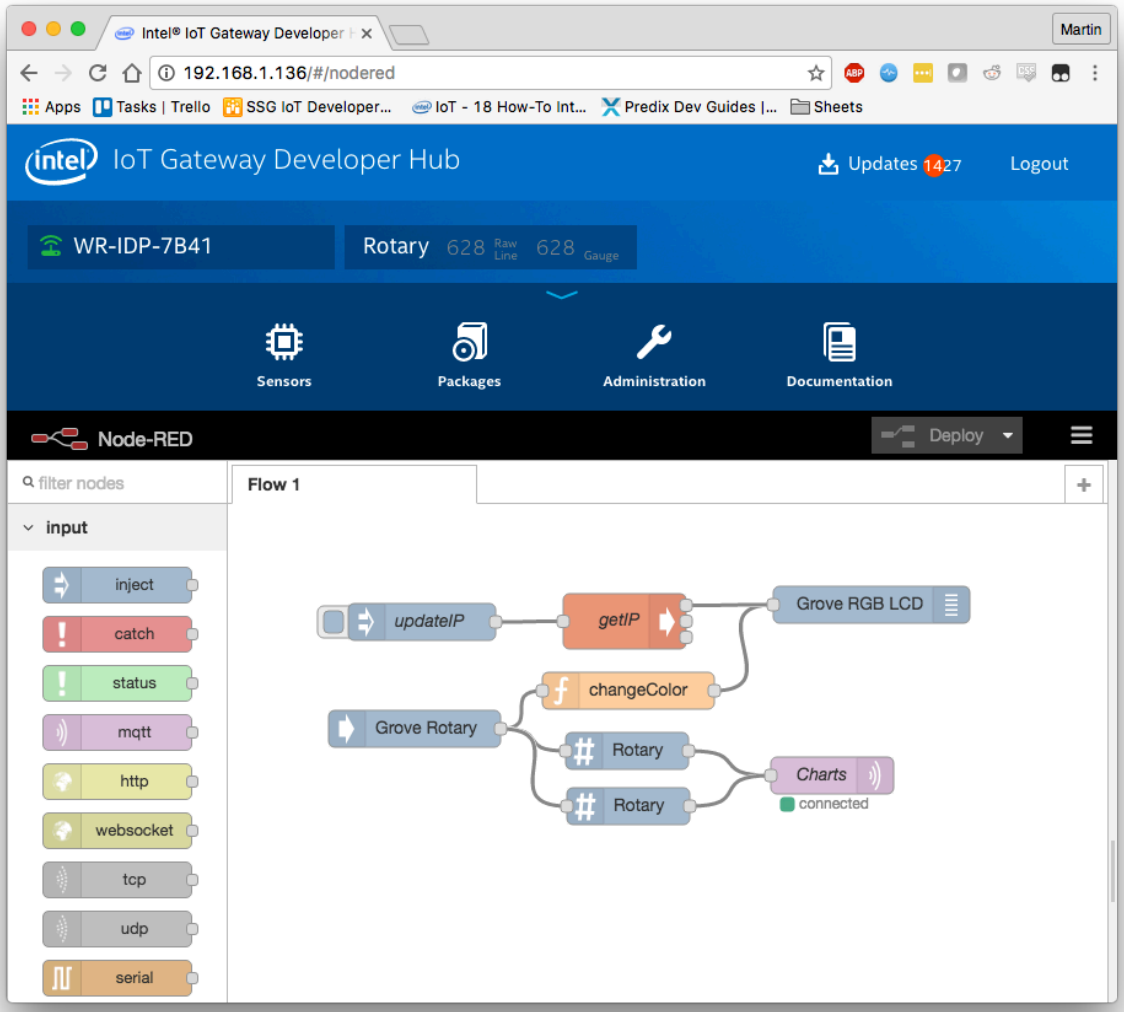
- Open a Terminal
- Type $ *ssh root@<<IP Address>>. Replace <<IP Address>> with the IP address of your gateway.*
- Enter ***root*** as password

Starting Node-RED for Blinky LED

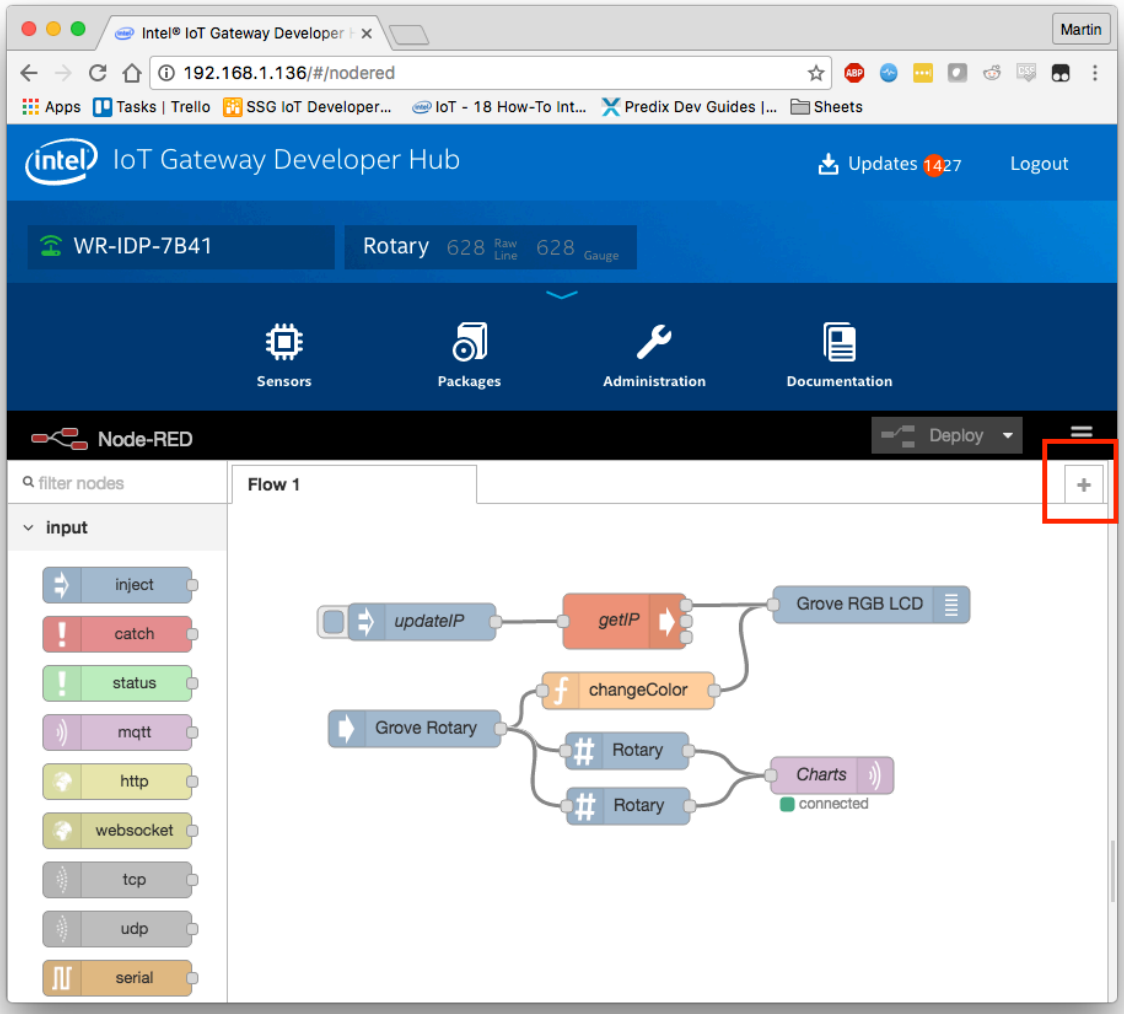
- Go to Development Hub by entering the IP Address assigned to your Gateway (Eg. http://192.168.1.1) in your browser.
- To load the Node-RED interface, click **Administration** on the navigation ribbon. Click **Launch** under the Node-RED icon.



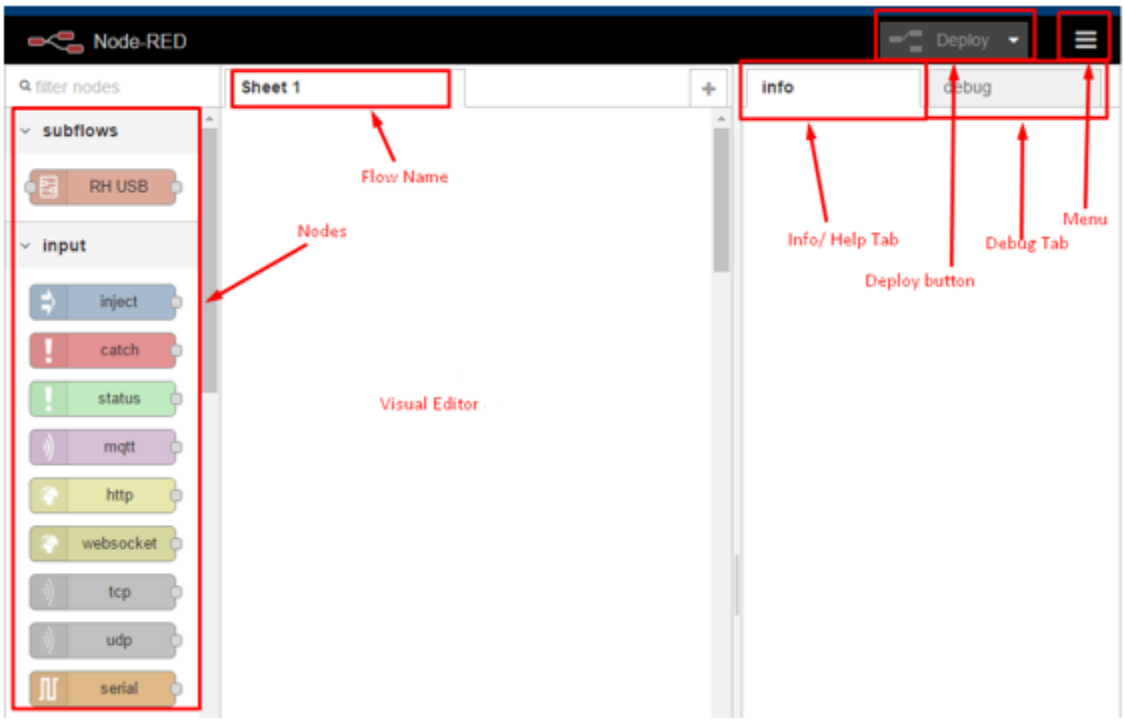
- Node-Red is a visual tool for writing the Internet of Things (IoT).
- Enter username and password as **root** and **root** respectively.



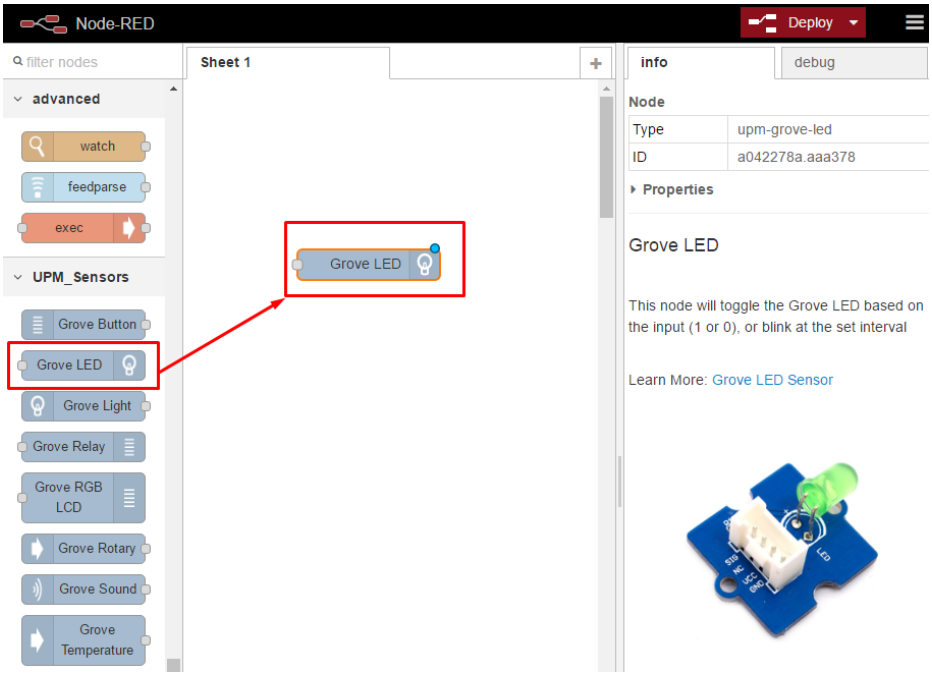
You will see the default application that sends the IP address to the LCD screen and the rotary angle sensor data to the chart on the dashboard. Leave this flow as is for now.



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

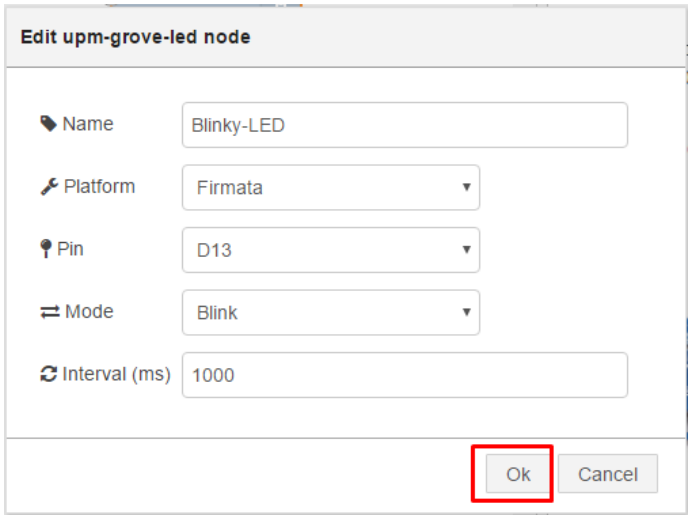


- Deploy Blinky LED Node. Go to Nodes, at the bottom of page there will be group on Nodes called UPM Sensors. Drag Grove LED from Nodes to Visual Editor

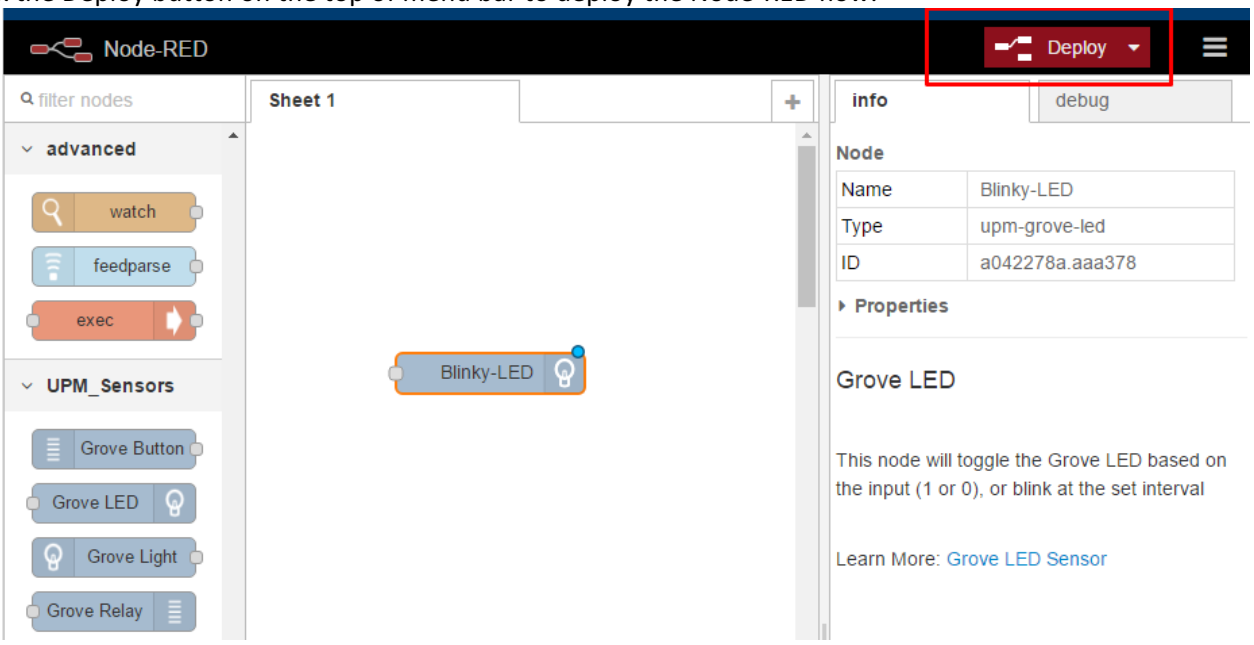


- Double Click on Grove LED node on visual Editor. It will open a dialog box to edit properties of node.
  - ➢ Name – Blinky LED
  - ➢ Platform – Firmata
  - ➢ Pin – D13
  - ➢ Mode- Blink
  - ➢ Interval(ms) – 1000

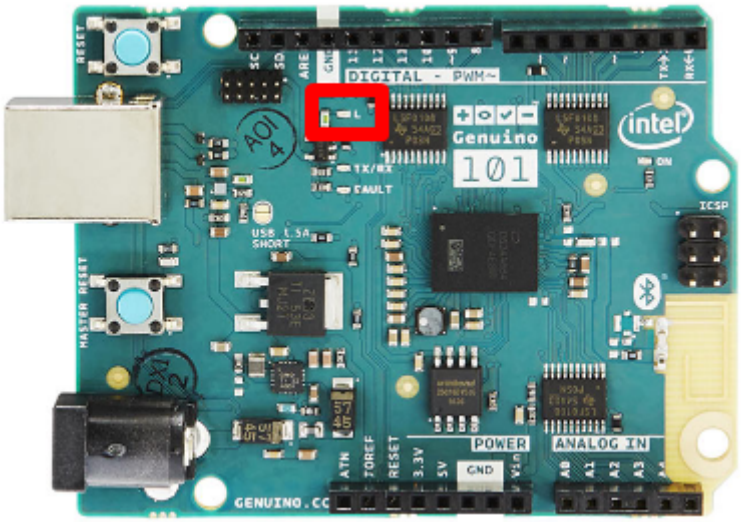
Then Click OK to save changes.



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

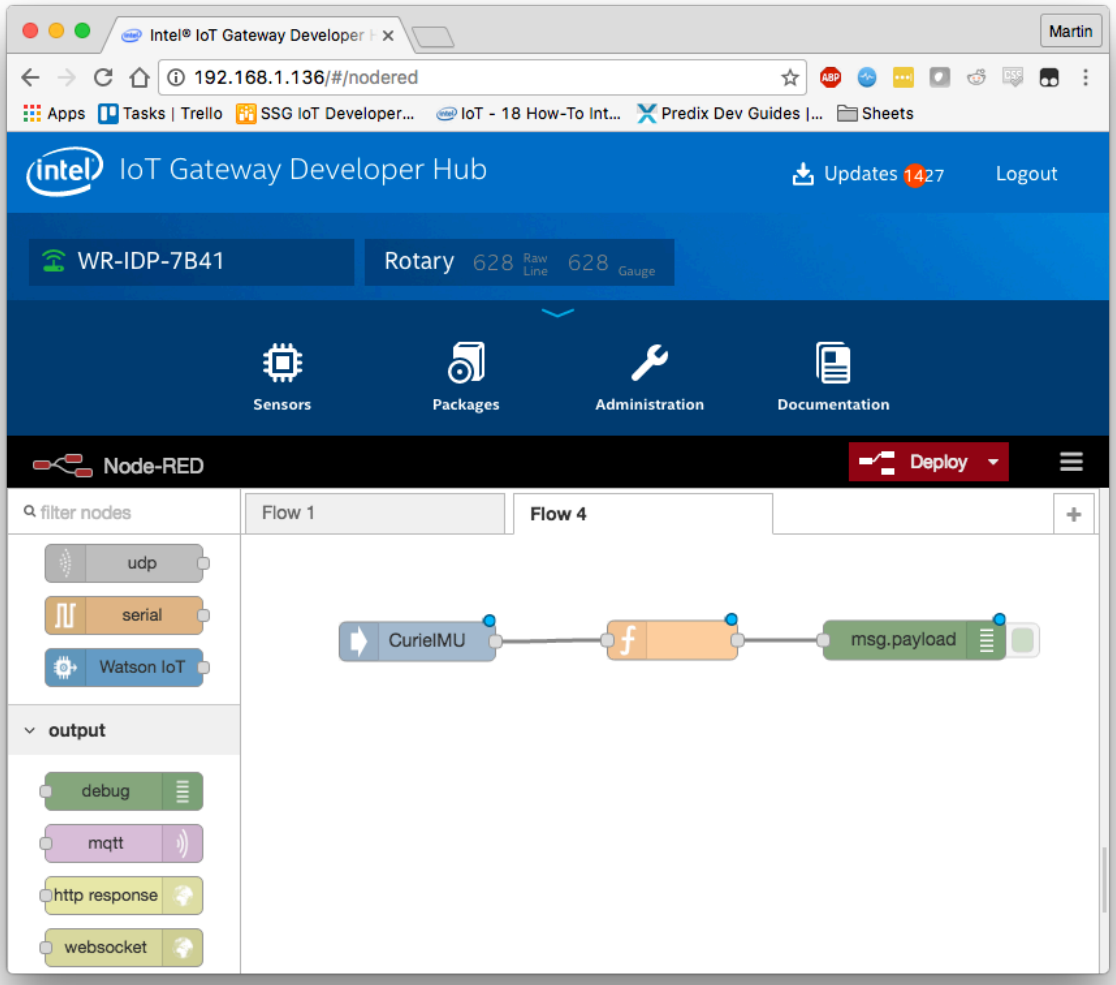


- The Arduino 101 LED on board LED will start blinking

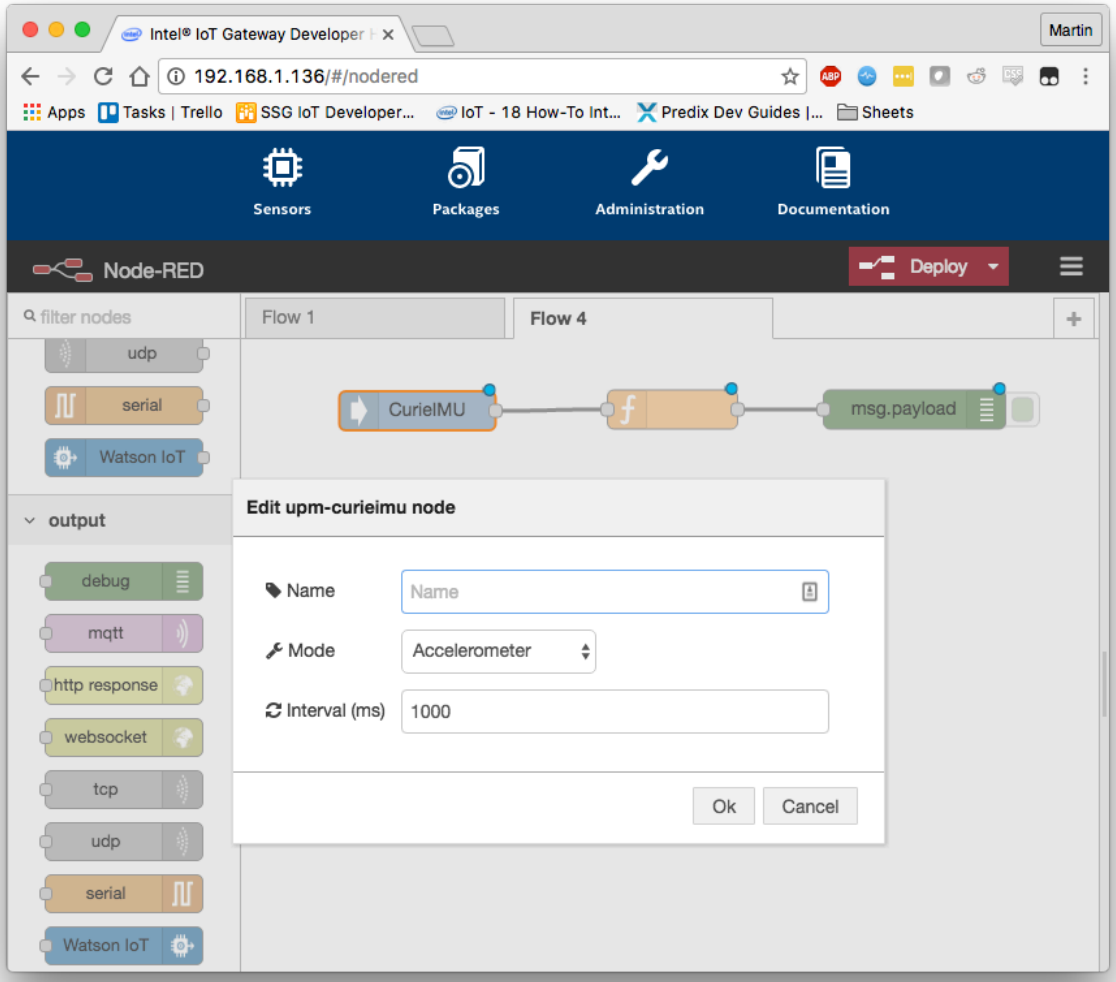


Adding Sensors and Processing to the Gateway

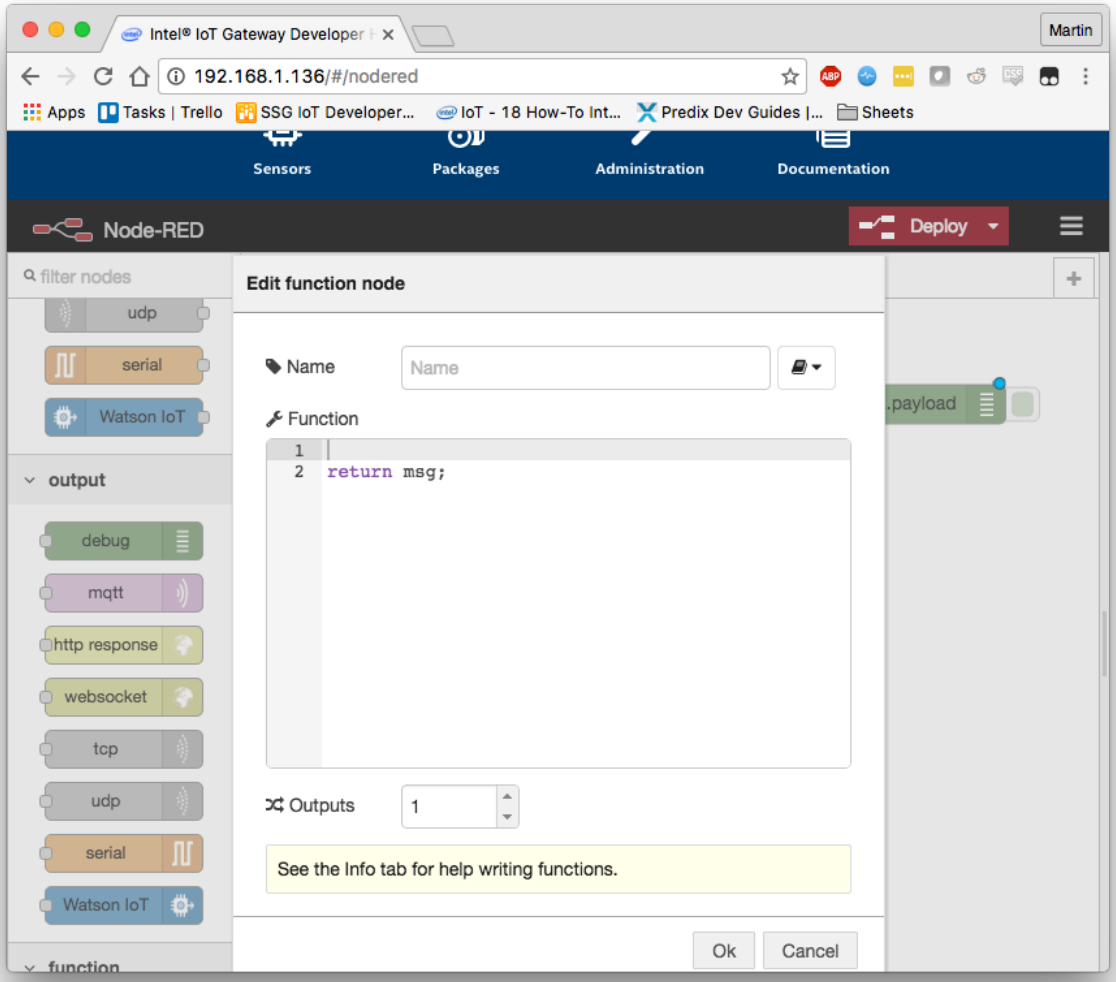
Now that you have a basic understanding of how to interact with the Gateway let's move on to adding some more sensor data and processing to the gateway.



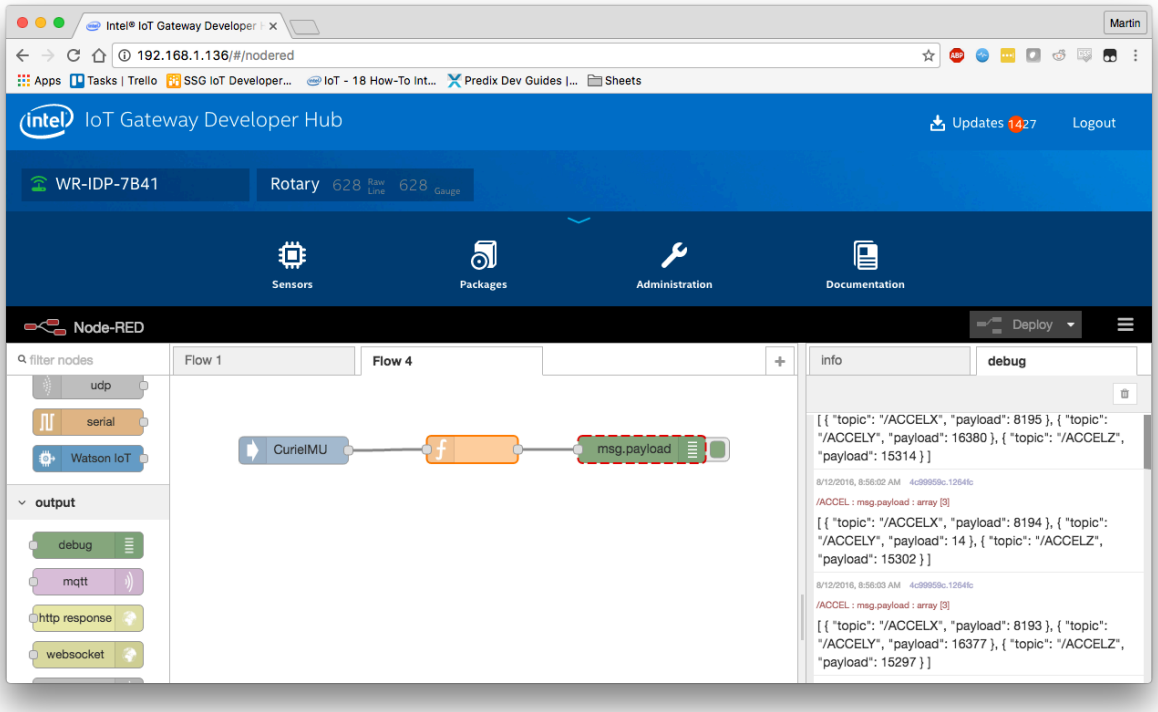
- From **UPM Sensors** add **CurieIMU** node
- From **function** add a **function** node
- From **output** add a **debug** node
- Wire the nodes together



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.



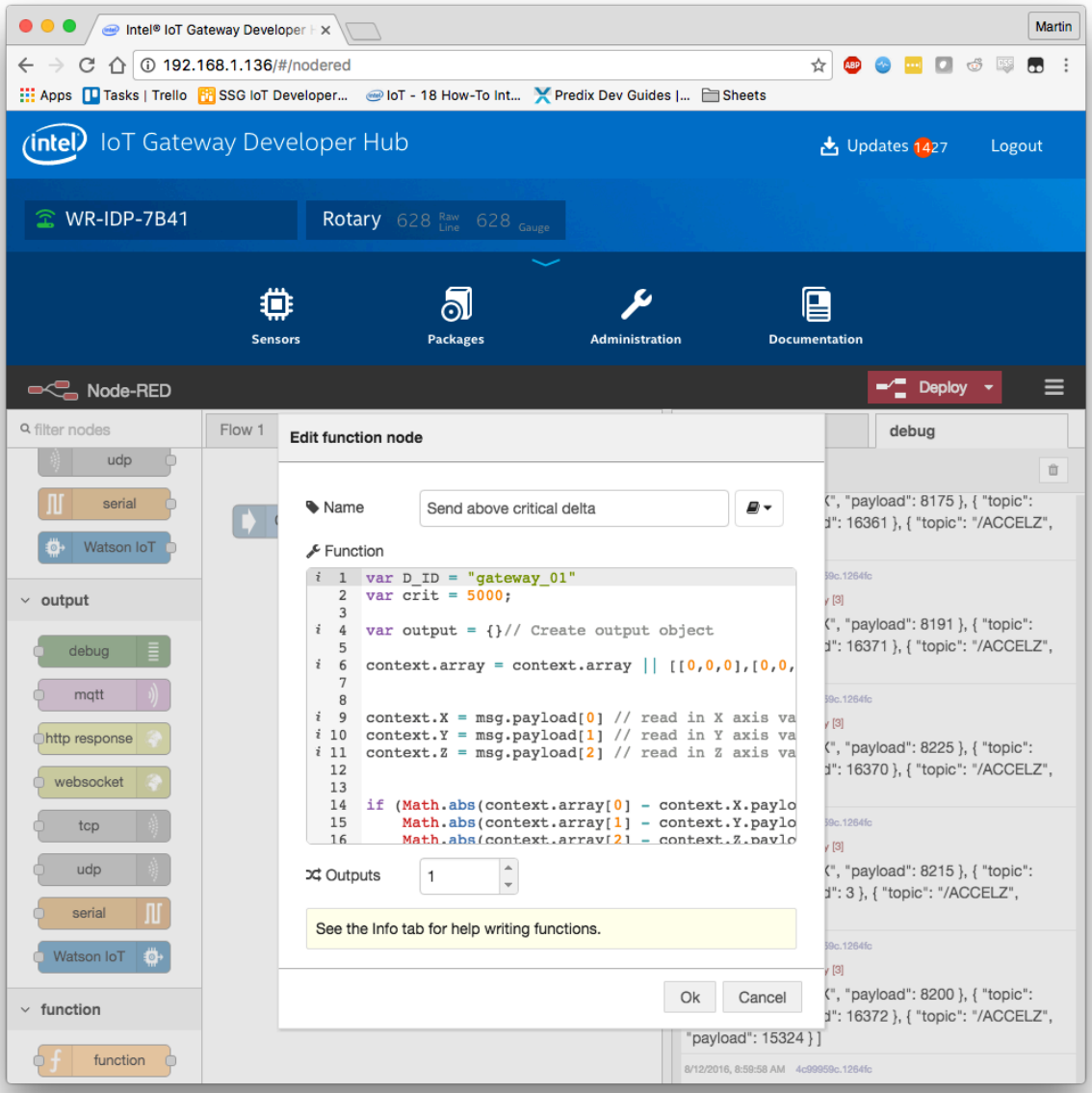
If you double click the function node you will bring up the function interface. Here you can enter JavaScript code that will be executed when the node gets input. By default, it only has “return msg;” which will simply pass the incoming message to the output. Let’s leave this for now.



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.

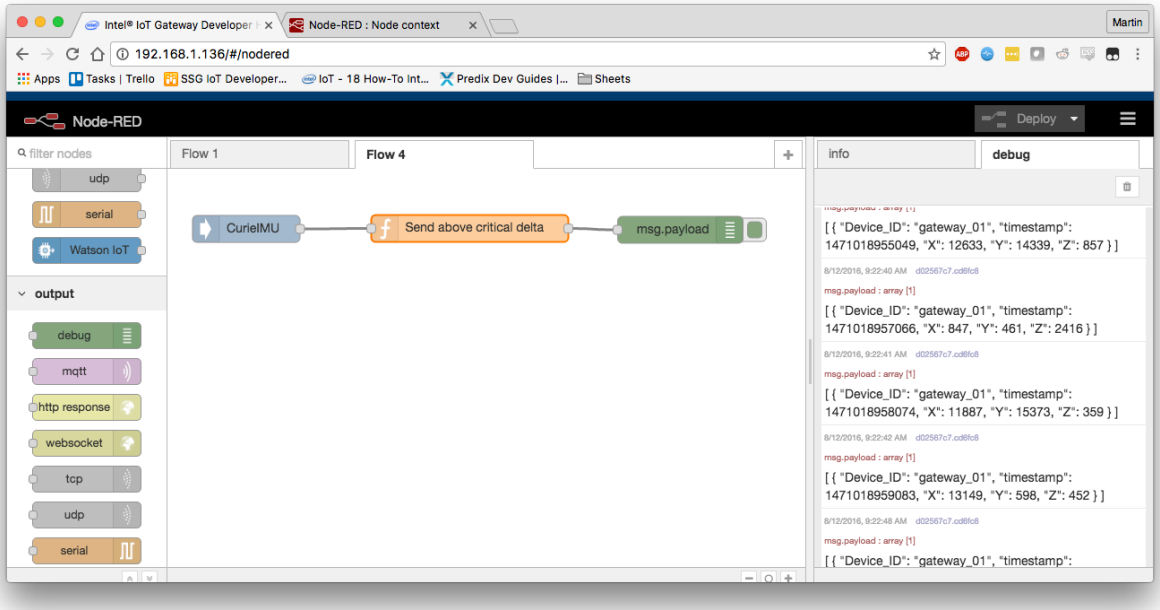


Open the function node again and enter / paste the following code - which is also available at : https://ibm.biz/BdrgCd

```
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.

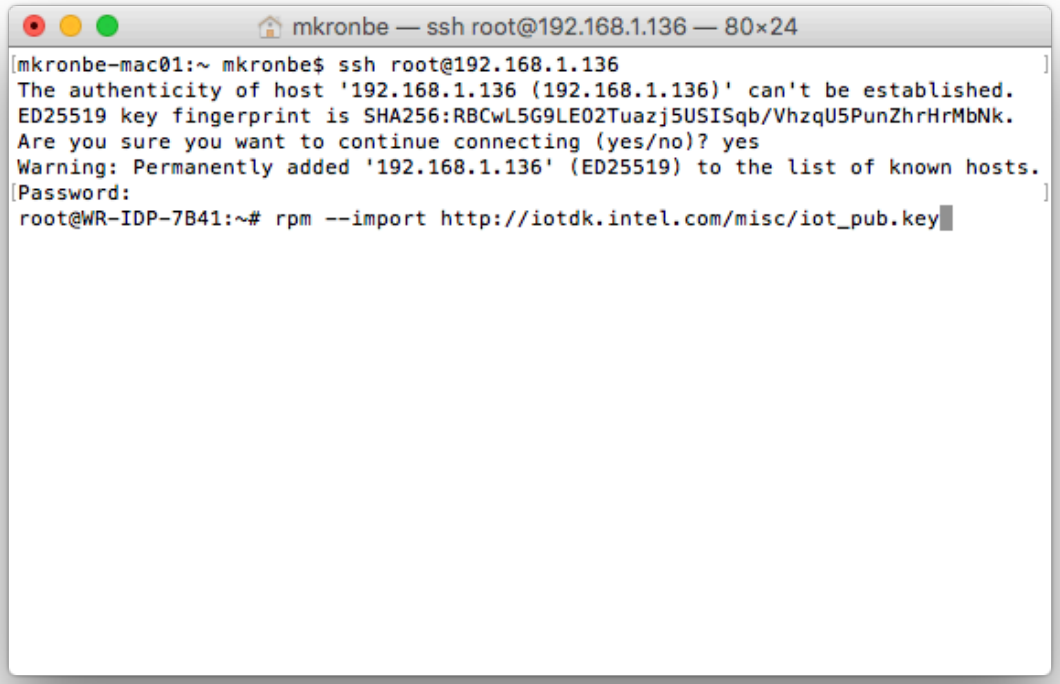


Hit deploy to run the code. The debug output should only display the accelerometer data when the Arduino 101 board registers movement.

Add Watson IoT Nodes and OS Monitoring

Next we will add some custom nodes in order to connect to Watson IoT as well as monitor memory load.

Open a command line interface to the Gateway. You can do this using the SSH interface of your choice or with the built in command line on the Gateway. **Note:** The built in command line interface is functional, however not recommended for extended use as it is missing many ease of life features.



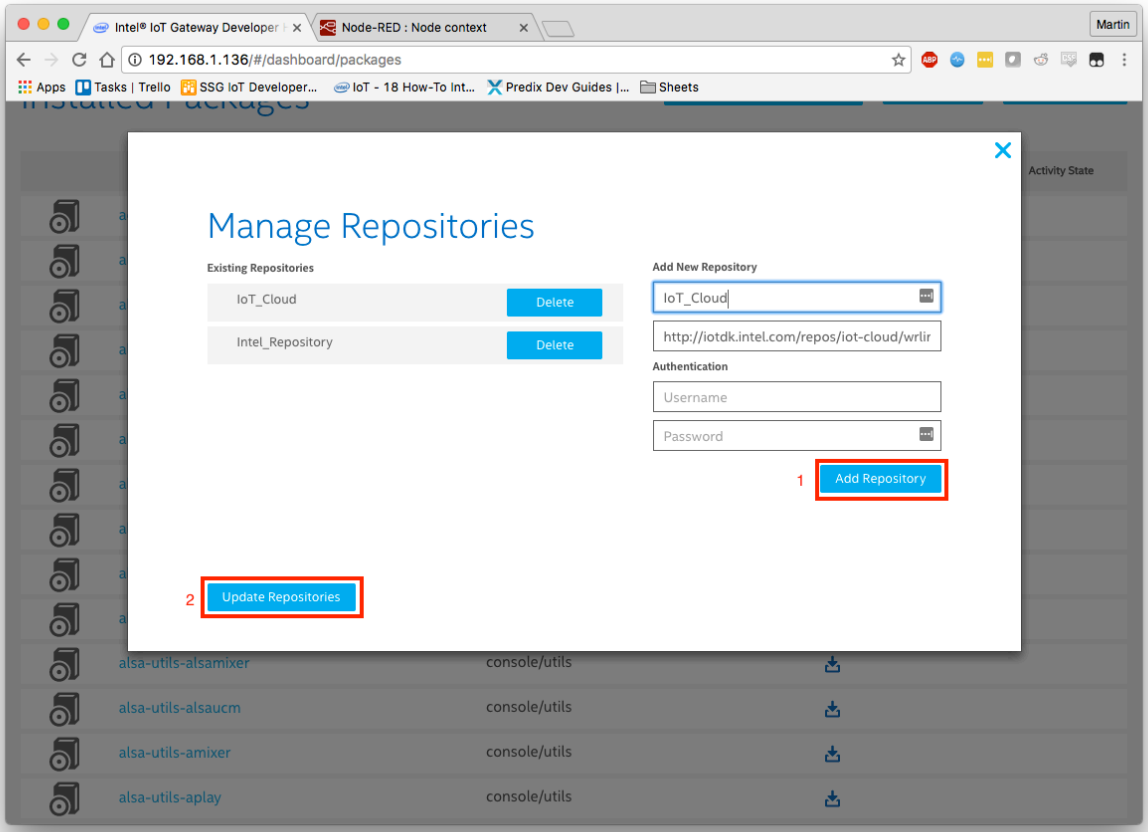
SSH into the ip address displayed on the LCD as root. The password is root. Once you are connected to the command line enter the following to add the repository key to rpm:

rpm –-import http://iotdk.intel.com/misc/iot_pub.key

Also install the Node-RED OS monitoring nodes with:

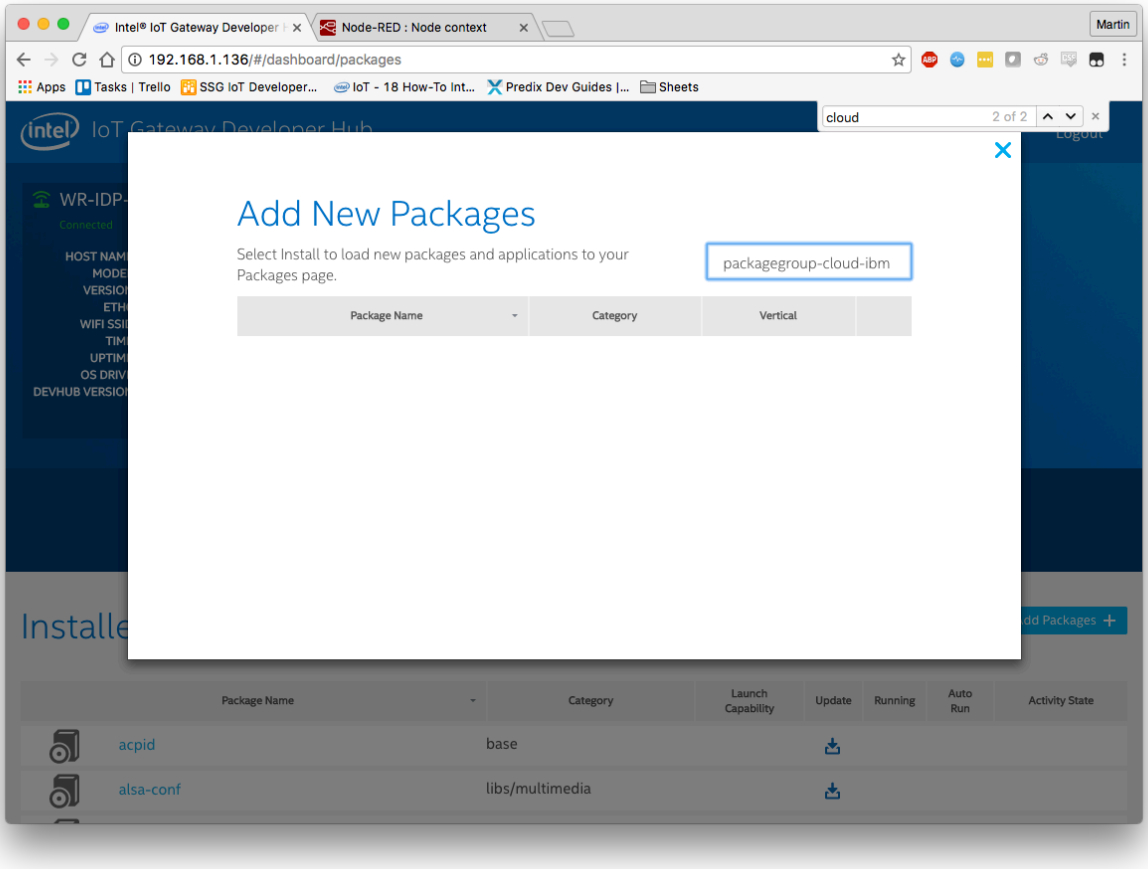
npm install node-red-contrib-os –g

The install process will throw some warnings; you can ignore these.

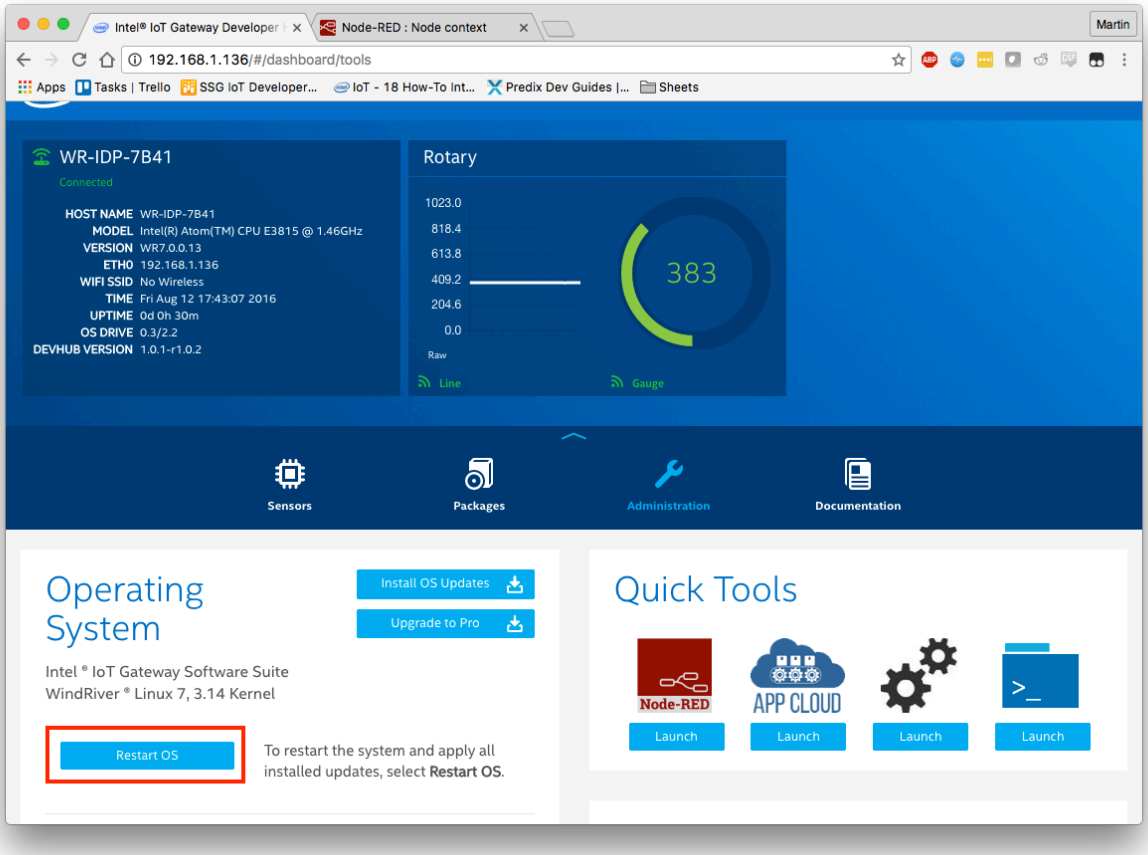


Navigate back to the repository manager on the Gateway.

- Select “Add Repo”
- In the Name Field enter “IoT_Cloud”
- In the URL Field enter http://iotdk.intel.com/repos/iot-cloud/wrlinux7/rcpl13/
- Click “Add Repository“
- After it is finished adding click “Update Repositories”
- Close the manage repository window

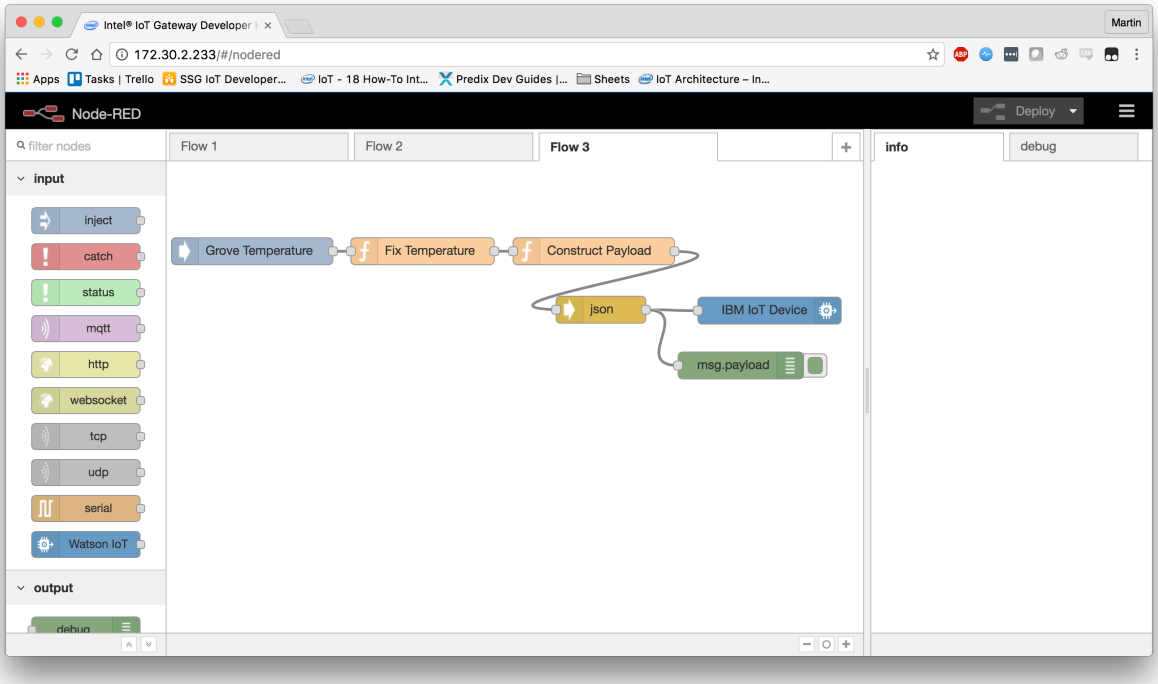


- Open “Add Packages”
- Search for “packagegroup-cloud-ibm”
- Click “Install” next to the package name
- Close the Add New Package window



Restart OS from the Administration tab

Create Temperature Flow



The above flow is available at http://bit.ly/2bONsqO

Once the Gateway reboots go back to your Node-RED, create a new flow and add the following nodes:

- From **UPM_Sensors** add a **Grove Temperature Sensor** node
- From **function** add 2 **function** nodes
- From **function** add a **json** node
- From **output** add a **Watson IoT** node
- Wire the nodes as shown above

In the “Fix Temperature” function add:

```
msg.payload = msg.payload * 3.3 / 5;
return msg;
```

This will allow the correct temperature to be read at a 5V board voltage (the sensor functions at 3.3 by default)

In the “Construct Payload” function add:

```
return {
   payload: {
         d: {
            temp: msg.payload
         }
   }
};
```

This will format the temperature to allow Watson IoT to ingest the data easier.

Once you have configured the IBM IoT Device nodes your flow will send local temperature data to IBM Watson IoT cloud services.

Gateway Security

McAfee Embedded Control

The Intel IoT Gateway provides a McAfee layer that let you configure McAfee embedded products for the Wind River Linux target platform. McAfee Embedded Control (MEC) provides the following capabilities in Wind River Linux platforms:

- Code and application protection, which only lets whitelisted programs (binary executables, scripts) run. Any program that does not appear in the whitelist cannot run. This stops malicious programs from installing and functioning on the system.
- Tamper proofing for whitelisted programs so the files cannot be modified on the disk, directories or volumes
- Dynamic whitelisting, which eliminates the need to manually maintain the inventory list of authorized applications. This features let you manage and update whitelisted files.

In this lab, you will perform the following tasks:

- Explore how MEC manages the inventory of executables, configurations, operation modes and logging.
- Enable McAfee embedded control
- Observe how the MEC code and application protection features work
- Verify the MEC write/read protection feature
- Use MEC update mode

1. Exploring McAfee Embedded Control

In this section you will explore how McAfee embedded control integrates into Wind River Linux and how MEC manages the inventory of executables, configurations, operating modes and logging.

1. On the Intel Iot Gateway, open a terminal window and execute the following command to confirm that the MEC RPM is installed:

```
# rpm -qa | egrep solidcore
```

   You should see: solidcores3-6.6.0_145-r0.0.intel_baytrail_64

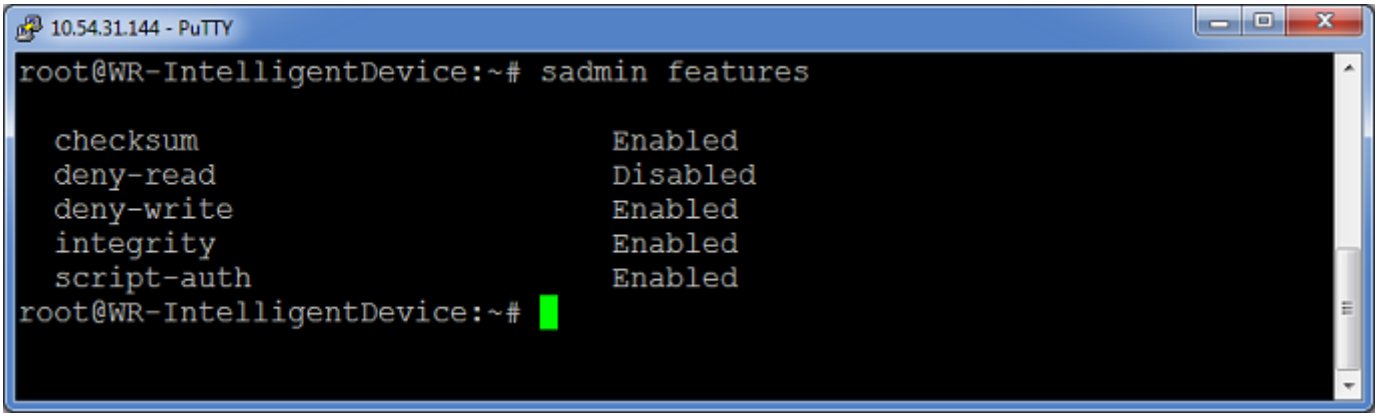
2. Execute the following command to display the help menu:

```
# sadmin help
```

```
root@WR-IntelligentDevice:~# sadmin help
Copyright 2008-2015 McAfee, Inc. All Rights Reserved.
Usage: sadmin <COMMAND> [options] [arguments]

Sadmin is the command line interface to administer McAfee Solidifier.

aef                         Modify or display advanced exclude filter rules.
begin-update (bu)           Begin update window to allow updates to the system
disable                     Disable McAfee Solidifier control on next reboot
enable                      Enable McAfee Solidifier control on next reboot
end-update (eu)             End update window
help                        Display help for basic commands
help-advanced               Display help for advanced commands
license                     Configure McAfee Solidifier licenses
monitor (mon)               Modify or display the monitoring rules
passwd                      Set or unset a password for the actionable commands
solidify (so)               Solidify the system
status                      Display status of McAfee Solidifier
trusted                     Modify or display the rules for trusted paths
unsolidify (unso)           Unsolidify the specified file, directory or volume
updaters                    Add, list or remove authorized updaters
version                     Display version of McAfee Solidifier

Type 'sadmin help <COMMAND>' for detailed help on a specific command.
root@WR-IntelligentDevice:~#
```

3. Execute the following command to review the list of all application control features and their status (enabled or disabled).

```
# sadmin features
```



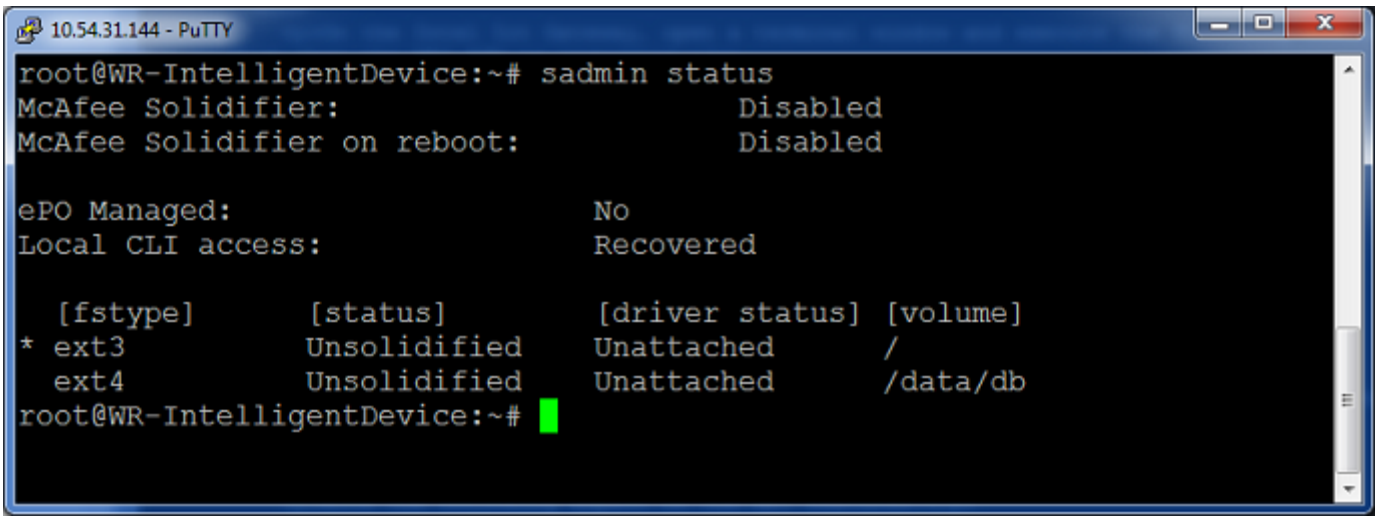
Note the following aspect of the MEC features:

- The feature **script-auth** is like deny-exec, but for scripts - only whitelisted scripts file can execute.
- The feature **deny-write** provides tamper-proofing to protect data file (for example configuration file). Unlike the deny-exec and script-auth features (which rely on a whitelist), the deny-write feature is rules-based. The MEC configuration file (solidcore.conf) records the rules.

- The feature **deny-read** provides tamper-proofing to prevent reading of critical files. The feature **deny-read** is alos rule based (like deny-write) - The MEC configuration file (solidcore.conf) records the rules. The feature is disabled by default.
- The feature **integrity** protects MEC data and file from modification , renaming or deletion.

4. As the MEC administrators, execute the following command to check the status of the McAfee embedded control on your target:

```
# sadmin status
```



Observe that the status is Unsolidified.

The following table describes the fields and their meaning:

Field	Description
McAfee Solidifier	specifies the operation mode of the application control
McAfee Solidifier on reboot	specifies the operation mode of the application control after a system restart
ePO Managed	displays the connectivity status of application control with McAfee ePO. In a standalone configuration, this status is No.
Local CLI Access	displays the statue (lockdown or recovered) of the local CLI. In standalone configuration, this status is Recovered.
fstype	displays the supported file systems for a volume
status	displays the current whitelist status for all the supported

	volumes on a system. If a volume name is specified, only the whitelist status for the volume displays.
driver status	displays whether the application control driver is loaded on a volume. If the driver is loaded, the status is attached; otherwise the status is unattached.
volume	displays the volume name

5. Execute the following command to display the log file:

```
# cat /usr/local/mcafee/solidcore/log/solidcore.log
```

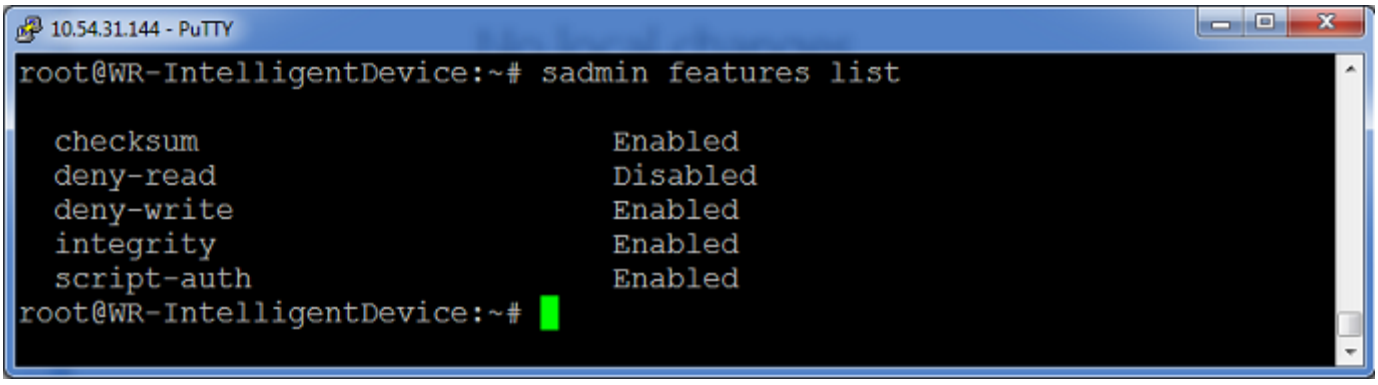
6. Execute the following command to display the configuration file:

```
# cat /etc/mcafee/solidcore/solidcore.conf | more
```

Observe that the file includes the following rules and configurations:

- the run-time mode
- the run-time mode on next reboot
- the license
- the features installed
- the features enabled
- write protect, read protect, and monitoring rules
- the installation directory
- the log file directory

```
# sadmin features list
```



2. Enabling McAfee Embedded Control

In this section you will generate an initial whitelist, enable MEC, and restart the McAfee solidifier service.

Dynamic whitelisting is a core feature of MEC. The whitelist (or inventory) enumerates the set of programs (called authorized or solidified program code) that are allowed to run on the host computer. Any programs not

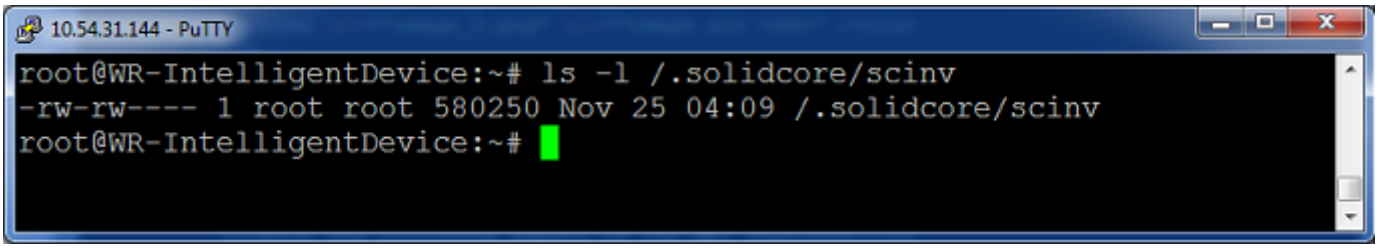
in the whitelist are considered unauthorized - they cannot execute, and MEC logs their failed attempt to execute. Whitelisting does not change the files listed in the inventory.

1. As the MEC administrator, execute the following command to create the initial whitelist.

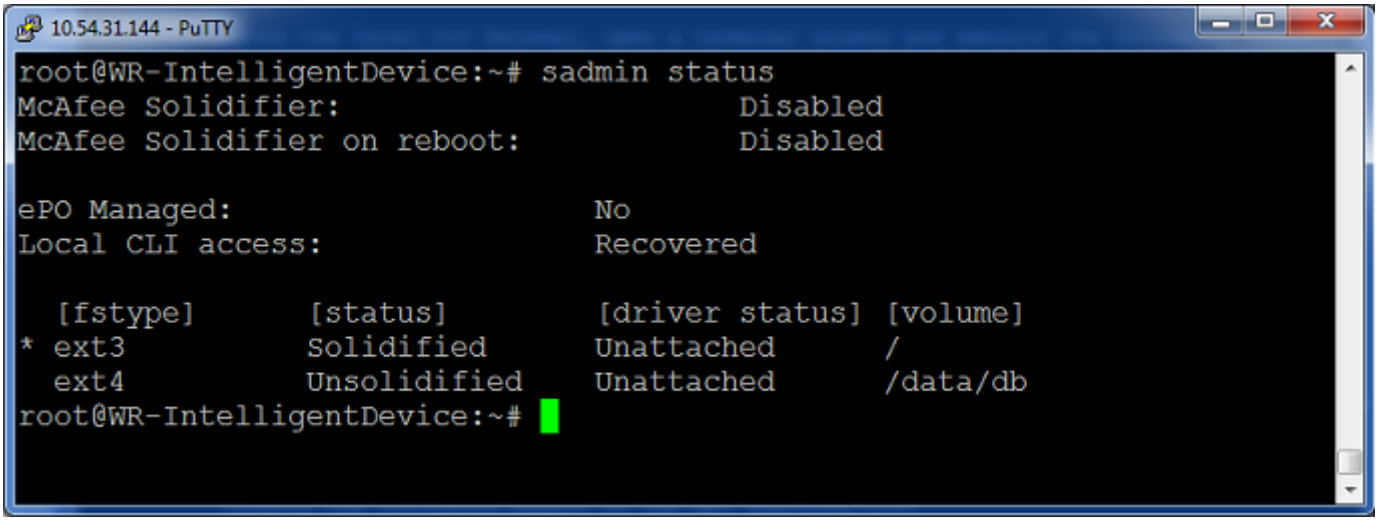
```
# sadmin solidify /bin /boot /etc /lib /lib64 /opt /root
/sbin /usr /www
```

   This command may take a few minutes to complete.

2. As the MEC administrator, execute the following command to confirm that MEC created the initial whitelist. The application control component stores a whitelist for each volume at volume**/.solidcore/scinv**



3. Execute the following command to check the status of McAfee embedded control.

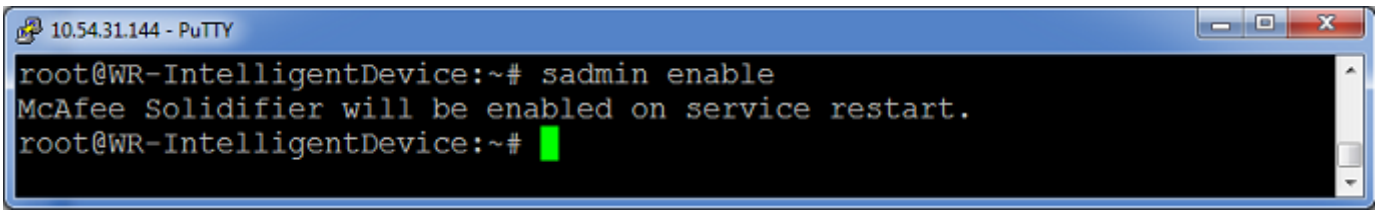


   Observe that the status of the ext3 drive has changed to Solidified.

4. As the MEC administrator, execute the following command to enable MEC.

```
# sadmin enable
```

   McAfee Solidifier will be enabled on service restart.

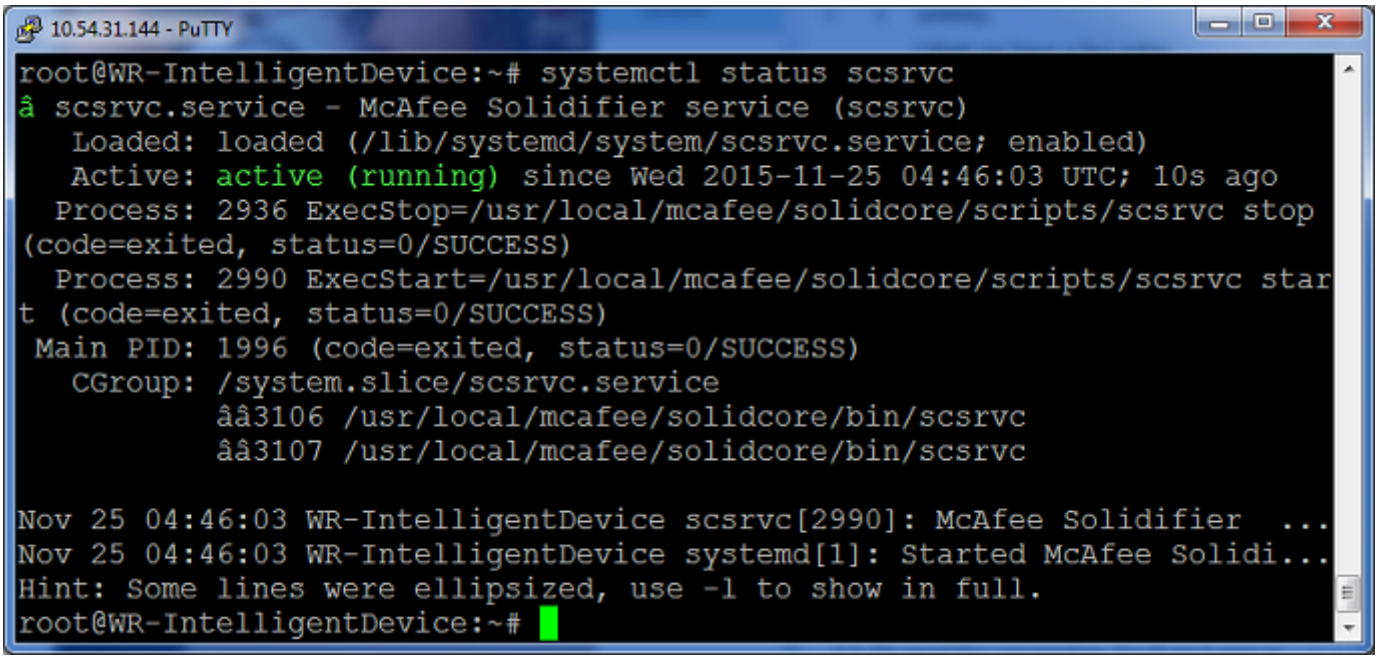


5. As the MEC admin, execute this command to restart the McAfee service

```
# systemctl restart scsrvc
```

6. Execute the following command to verify that MEC is enabled.

```
# systemctl status scsrvc
```



3. Verifying MEC Code and Application Protection

In this section you will create a script and observe how MEC code and application protection works.

MEC only lets whitelisted programs (binaries, executables, or scripts) run - any program not in the whitelist cannot run.

1. Create the script by typing

```
# vi /root/test_script.sh
```

Type "i" to enter insert mode and then type

```
#!/bin/bash
echo "Hello, World"
```

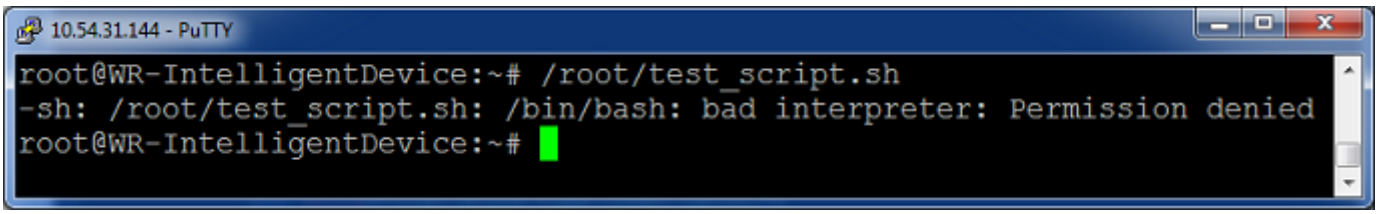
Then press "ESC" and type ":wq" This will save and quit the vi editor.

2. Give the script execute permissions

```
$ chmod +x /root/test_script.sh
```

Then try to execute the script.

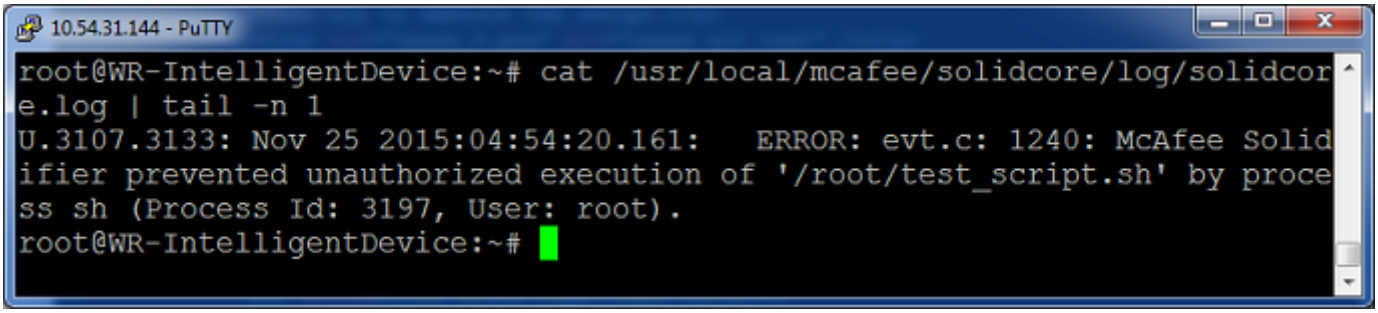
```
$ /root/test_script.sh
```



Note the response Permission denied - MEC did not allow the script to run.

3. Review the log by typing

```
cat /usr/local/mcafee/solidcore/log/solidcore.log | tail -n 1
```



This response indicates that MEC prevented the script from executing.

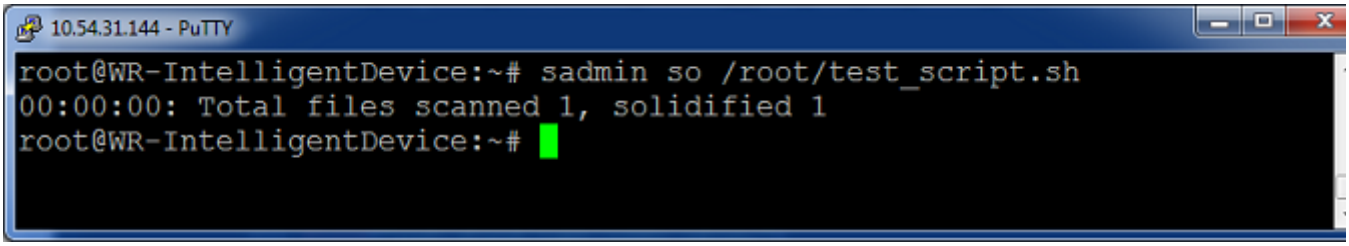
1. Check if the script is on the whitelist

```
sadmin list-solidified | grep /root/test_script.sh
```

Not out displays - this confirms that test_script.sh is not on the whitelist.

2. Add test_script.sh to the whitelist

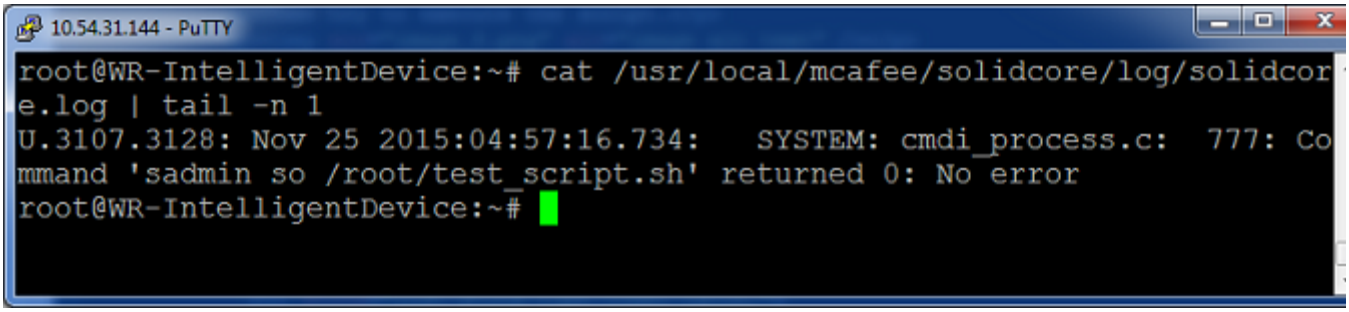
```
sadmin so /root/test_script.sh
```



3. Execute the command again.

```
/root/test_script.sh
```

4. Check the MEC log. The script was allowed to execute.



4. Verifying MEC Write/Read Protection

In this section you will implement write and read protection to the file /root/test_script.sh

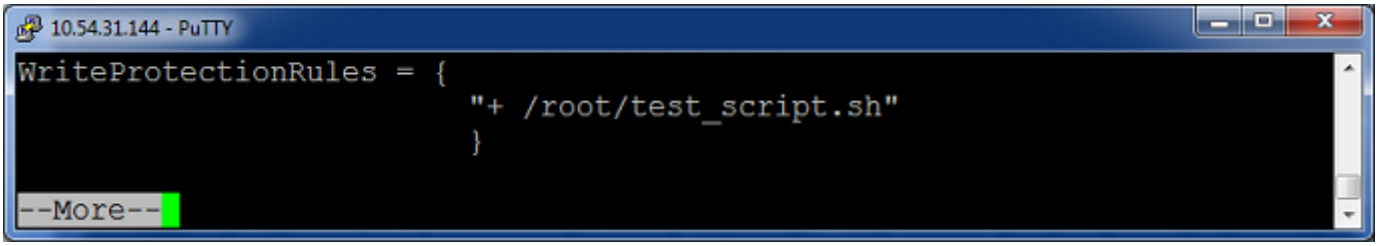
MEC not only protects executable files, but also data files (including configuration files). The MEC write/read protection feature provides tamper-proofing for all kinds of files. Unlike the code and application protection feature is rule-based, with the rules recorded in the MEC configuration file).

1. Execute the following command to set write protection for the file.

```
# sadmin wp -i /root/test_script.sh
```

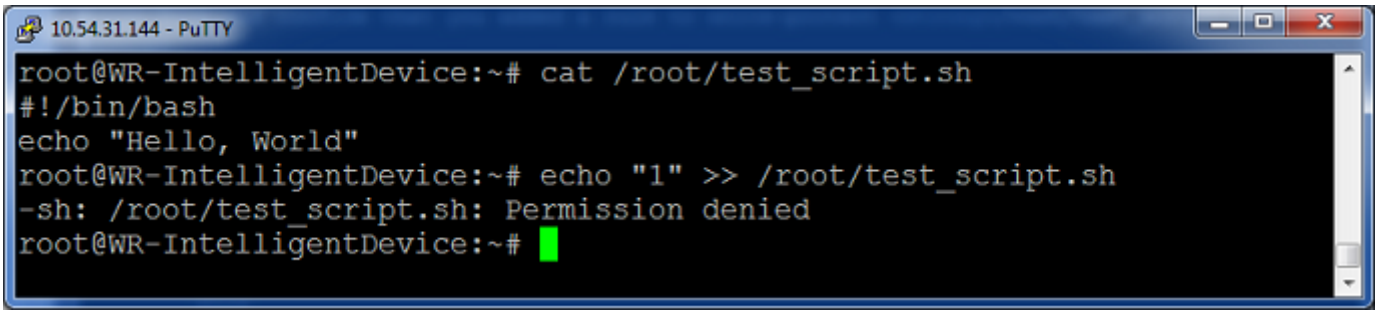
2. As the MEC administrator, execute the following command to review the MEC configuration file and confirm that you added a rule to write-protect **/root/test_script.sh**

```
# cat /etc/mcafee/solidcore/solidcore.conf | more
```



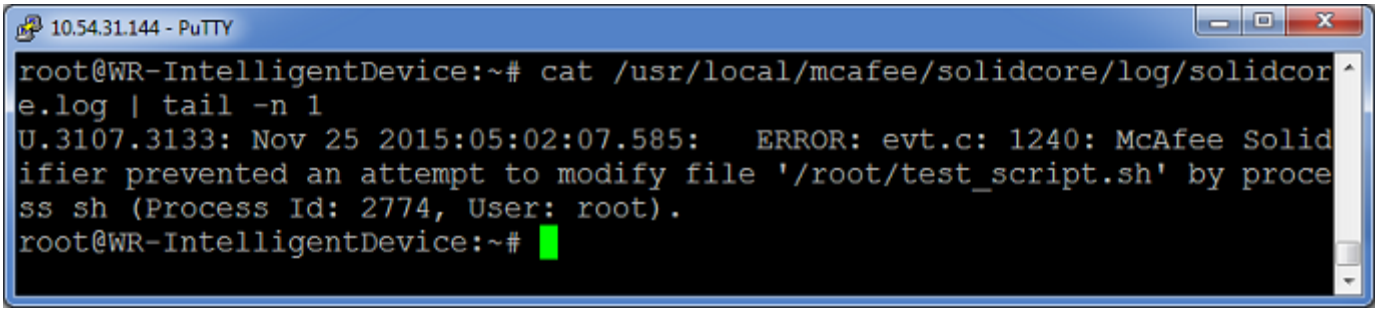
3. Now try to modify the file by executing the following command.

```
# touch /root/test_script.sh
```



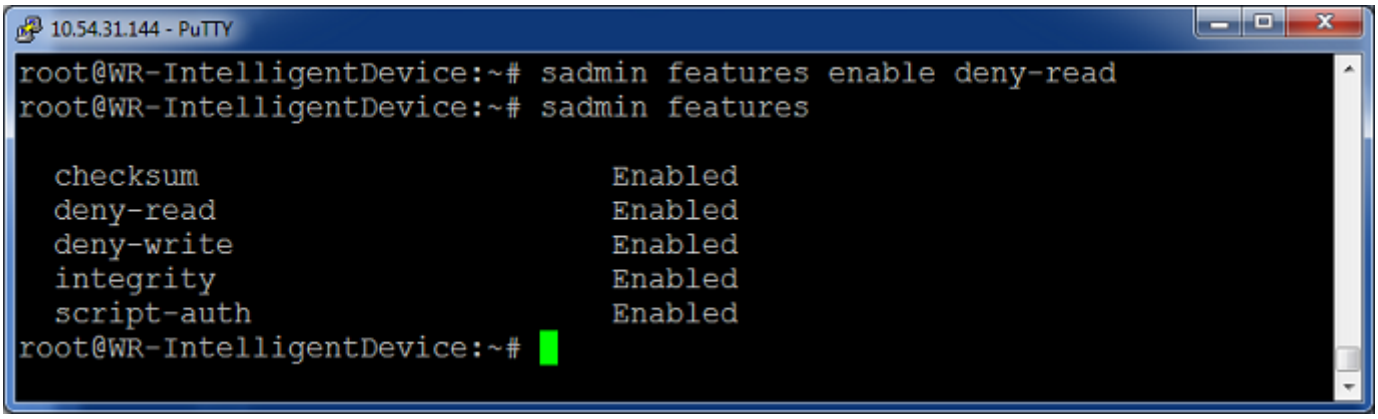
4. Execute the following command to review the messages in the log file solidcore.log so you can determine the reason for the Permission denied.

```
cat /usr/local/mcafee/solidcore/log/solidcore.log | tail -n 1
```



5. Execute the following command to enable deny-read and then review the status of the MEC features.

```
sadmin features enable deny-read
```



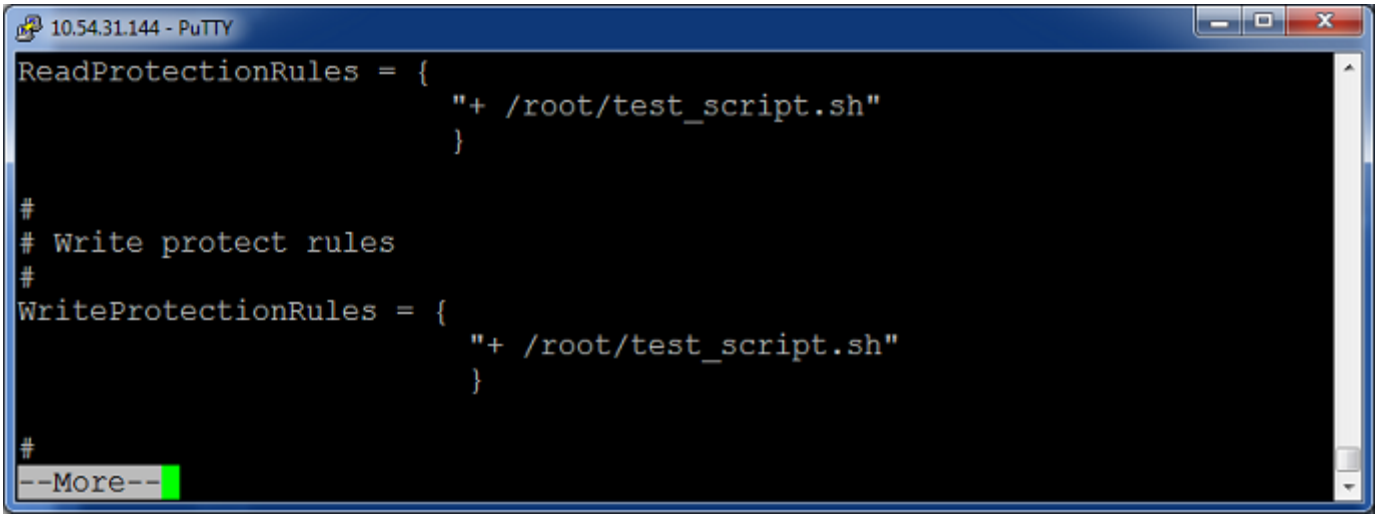
The response indicates that the deny-read feature.

6. Execute the following command to set read protection for the file.

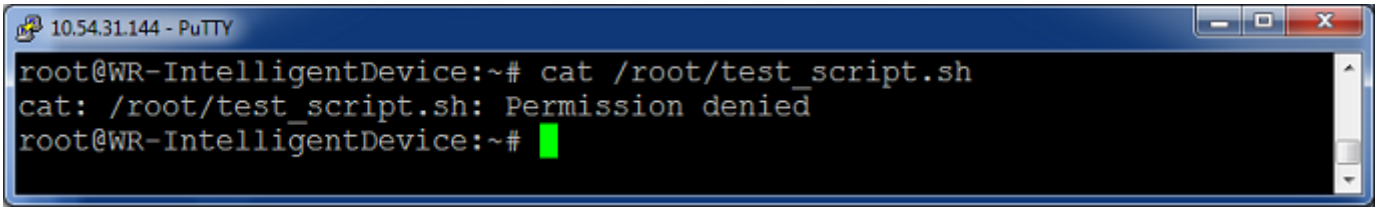
```
# sadmin rp -i /root/test_script.sh
```

7. Execute the following command to review the MEC configuration file and confirm that you added read protection for the file /root/test_setup.sh

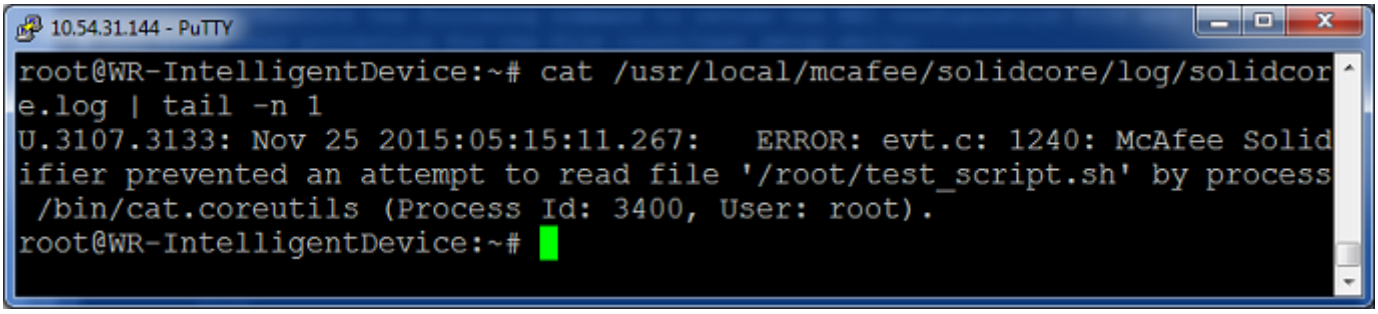
```
# cat /etc/mcafee/solidcore/solidcore.conf | more
```



8. Now attempt to read the file by printing to the console.



9. Execute the following command to review the messages in the log file solidcore.log so you can determine the reason for the **Permission denied error**.

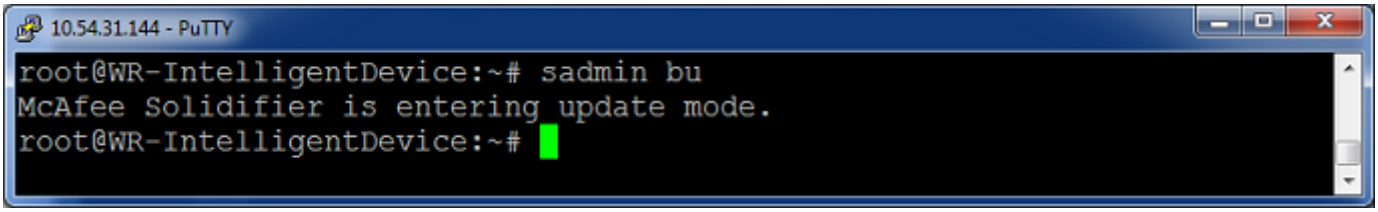


5. Using the Update Mode

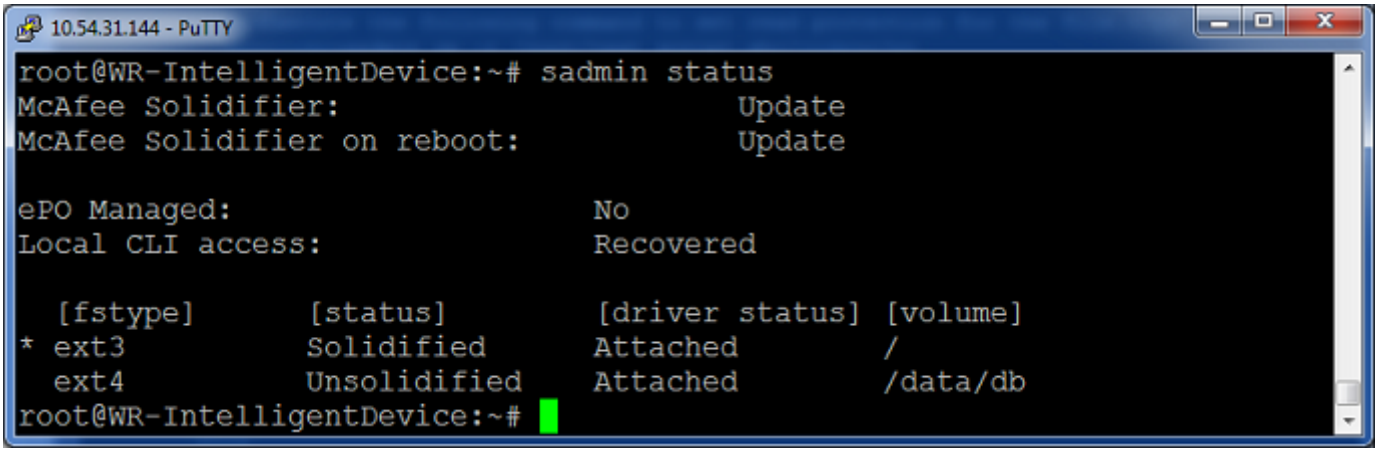
In this section you will use MEC update mode to update the file /root/test_script.sh which is write and read protected.

When MEC is in update mode it allows all changes on a protected system. You can use update mode to complete maintenance tasks, such as installing patches or upgrading software and data files.

1. Type this command to cause MEC to go into update mode.

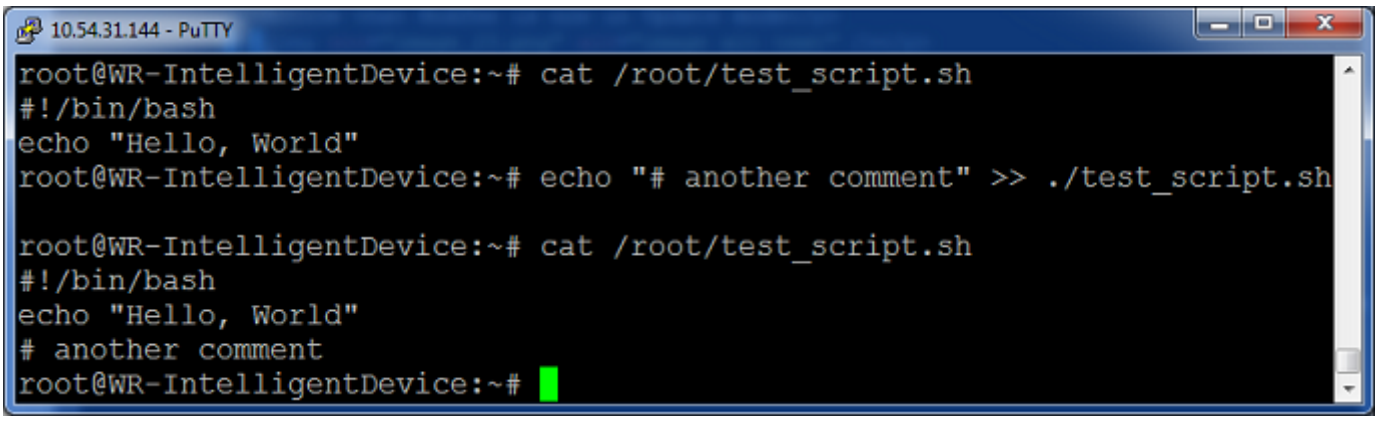


Notice that McAfee is now in Update mode

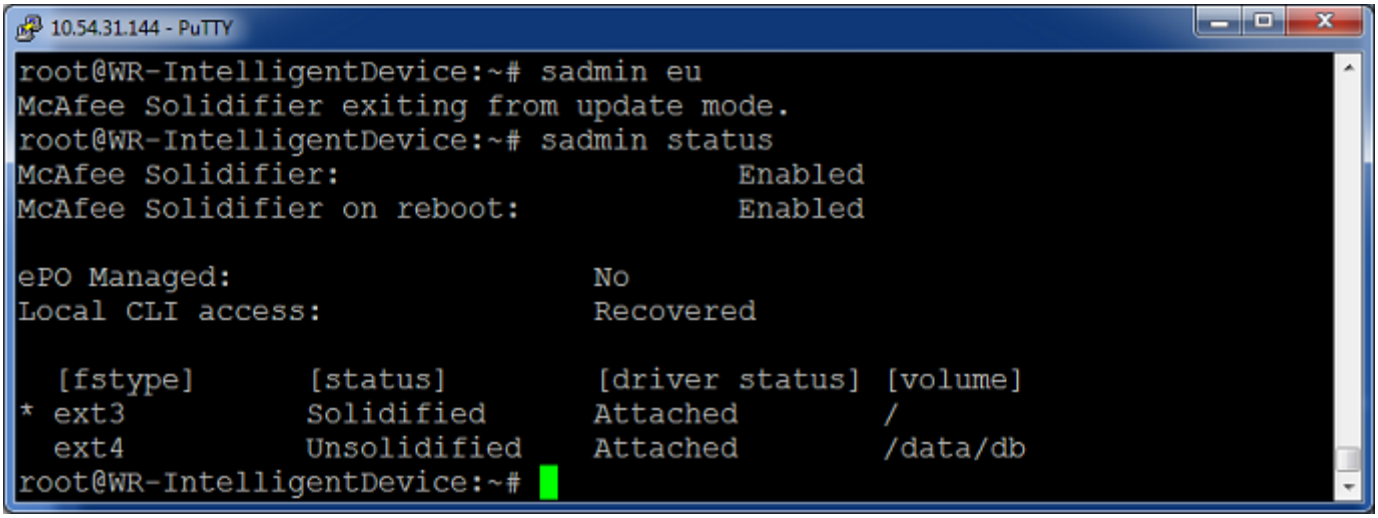


2. Execute the following command to display then update the contents of the file

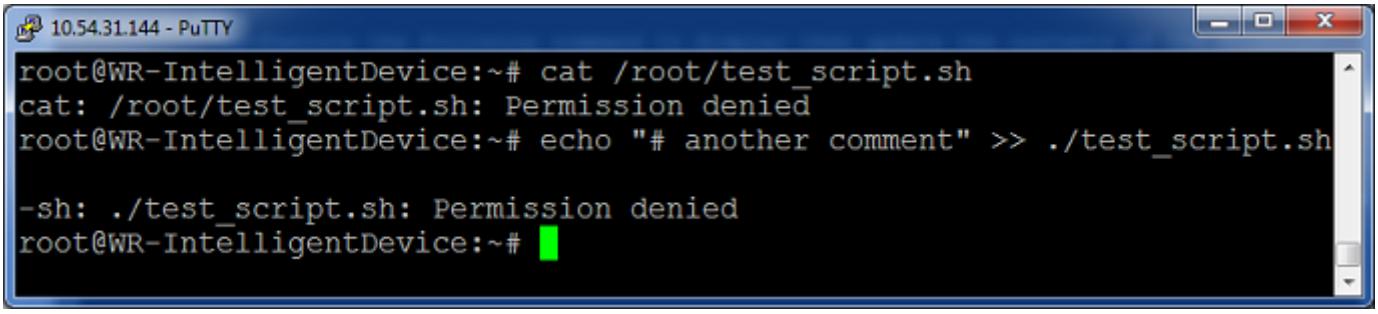
```
# cat /root/test_script.sh
# echo "# another comment " >> ./test_script.sh
# cat /root/test_script.sh
```



3. Now that you have modified /root/test_script.sh, execute the following command to change the MEC operational mode from update to enable and verify the new status.



4. Execute the following command to test that update mode is closed and write/read protection is working.



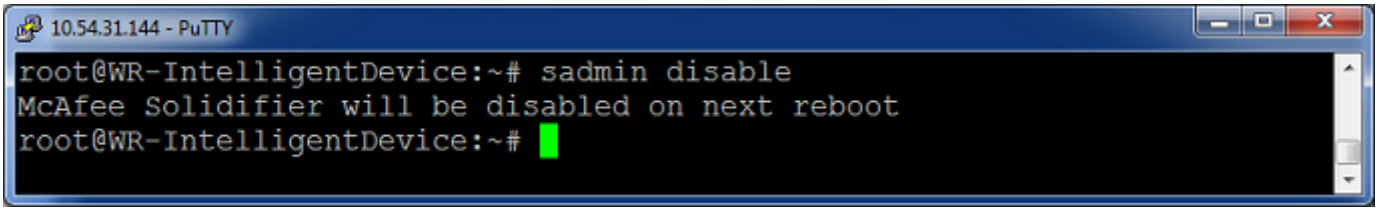
The **Permission denied** response confirm that you cannot read or change the file and that MEC protection is working.

Disable Application Control

In this section, you will disable the MEC feature to return your target to the standard configuration.

1. As the MEC administrator, execute the following command to switch EC to disabled mode to deactivate the features of MEC application control.

```
sadmin disable
```



Gateway Security

Role Based Security and Memory Protection with GR Security

Lab Overview

GRSecurity is a set of software patches released under the GNU GPL for the Linux kernel to enhance the system security. It allows the system administrator to, among other things, define a minimum privilege policy for the system, in which every process and user have only the lowest privileges necessary to function. GRSecurity has 2 primary features, the RBAC (Rules Based Access Control) and the PaX.

gradm, the administration utility for the role-based access control system, is a powerful tool that parses your ACLs (Access Control Lists), performs the enforcement of a secure base policy, optimizes the ACLs, as well as handles parsing of the learning logs, merges them with your ACL set and outputs the final ACLs.

Disable McAfee Embedded Control

By design, **McAfee Embedded Control** will protect your system from changes. So let's turn it off now.

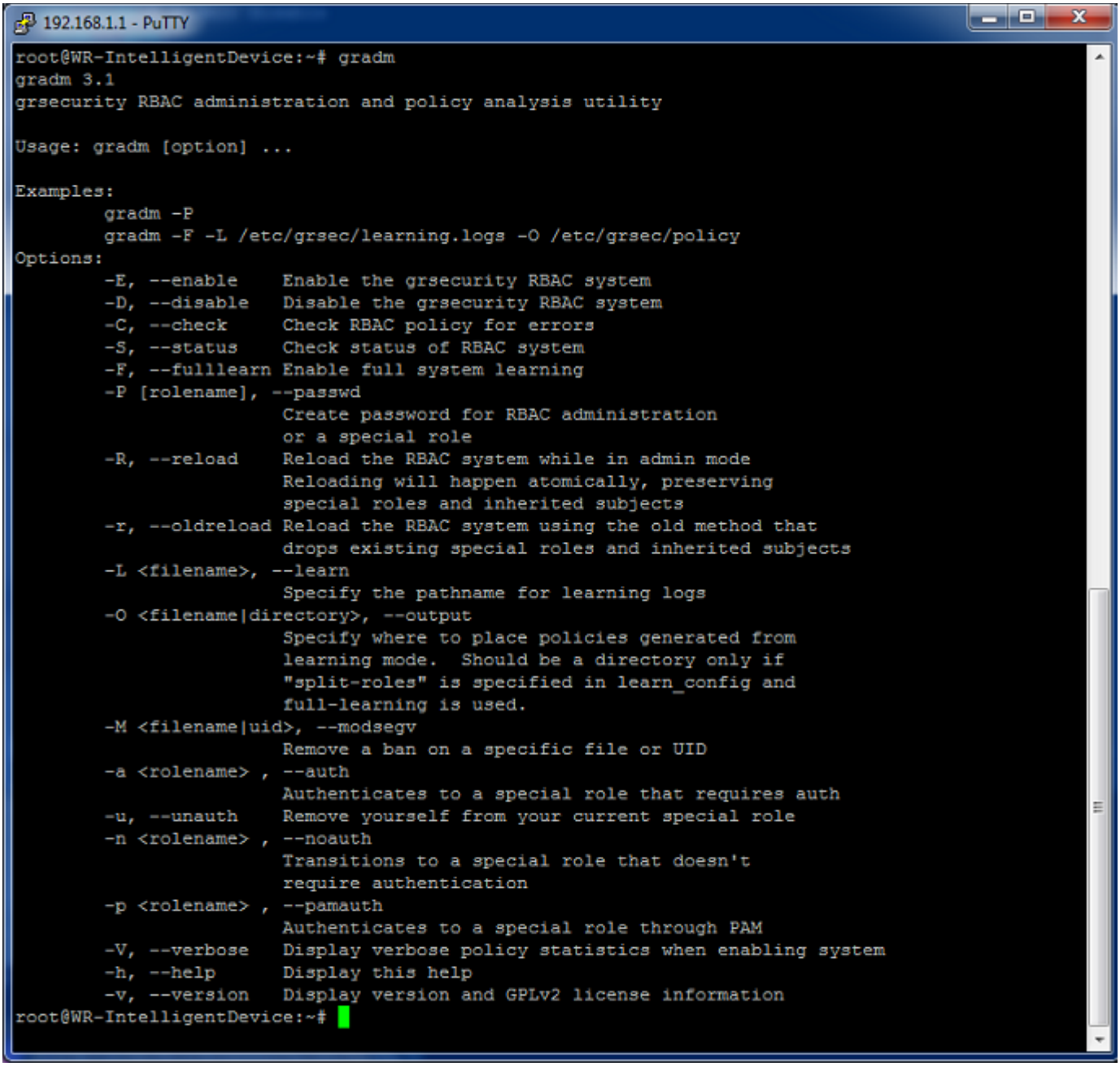
```
# sadmin disable
```

The gradm command

gradm is the grsecurity RBAC administration and policy analysis utility. It allows you to configure and administrate the grsecurity system.

To display all available command-line switches, run gradm --help.

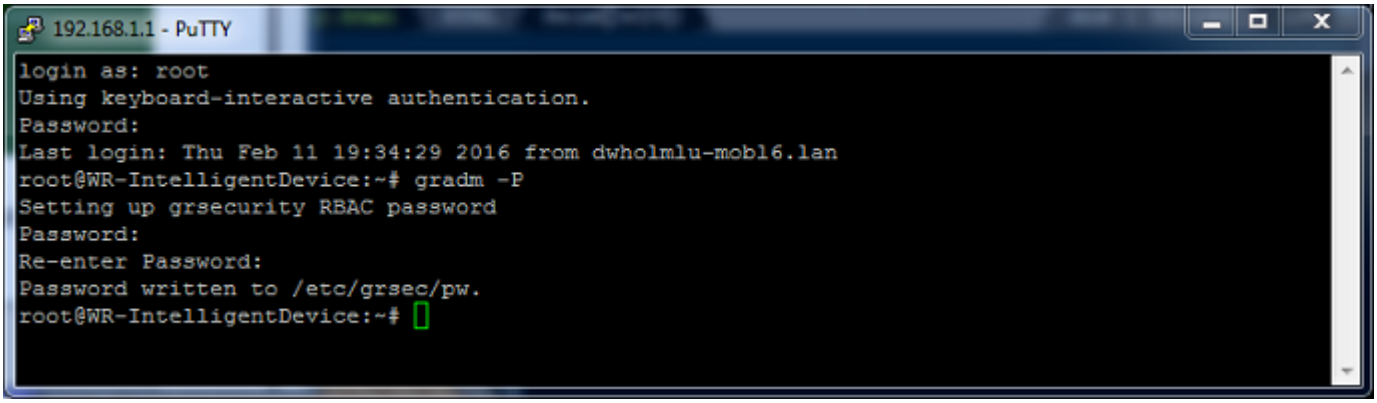
```
# gradm
```



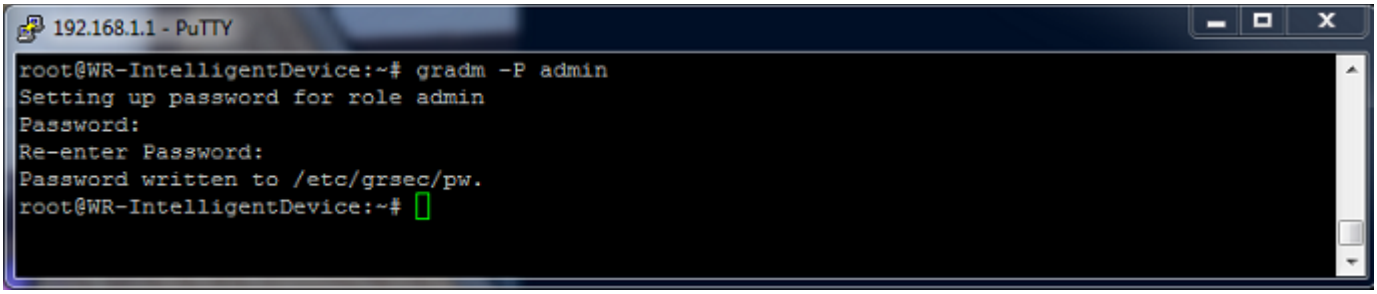
Setting up a Role-Based Access

Here we will create three roles **root**, **admin**, **shutdown**. Each of these roles will have unique abilities that will be constrained by the by RBAC. You can use any password that you like for each role.

```
# gradm -P
```




```
# gradm -P admin
```




```
# gradm -P shutdown
```

Enable Learning Mode

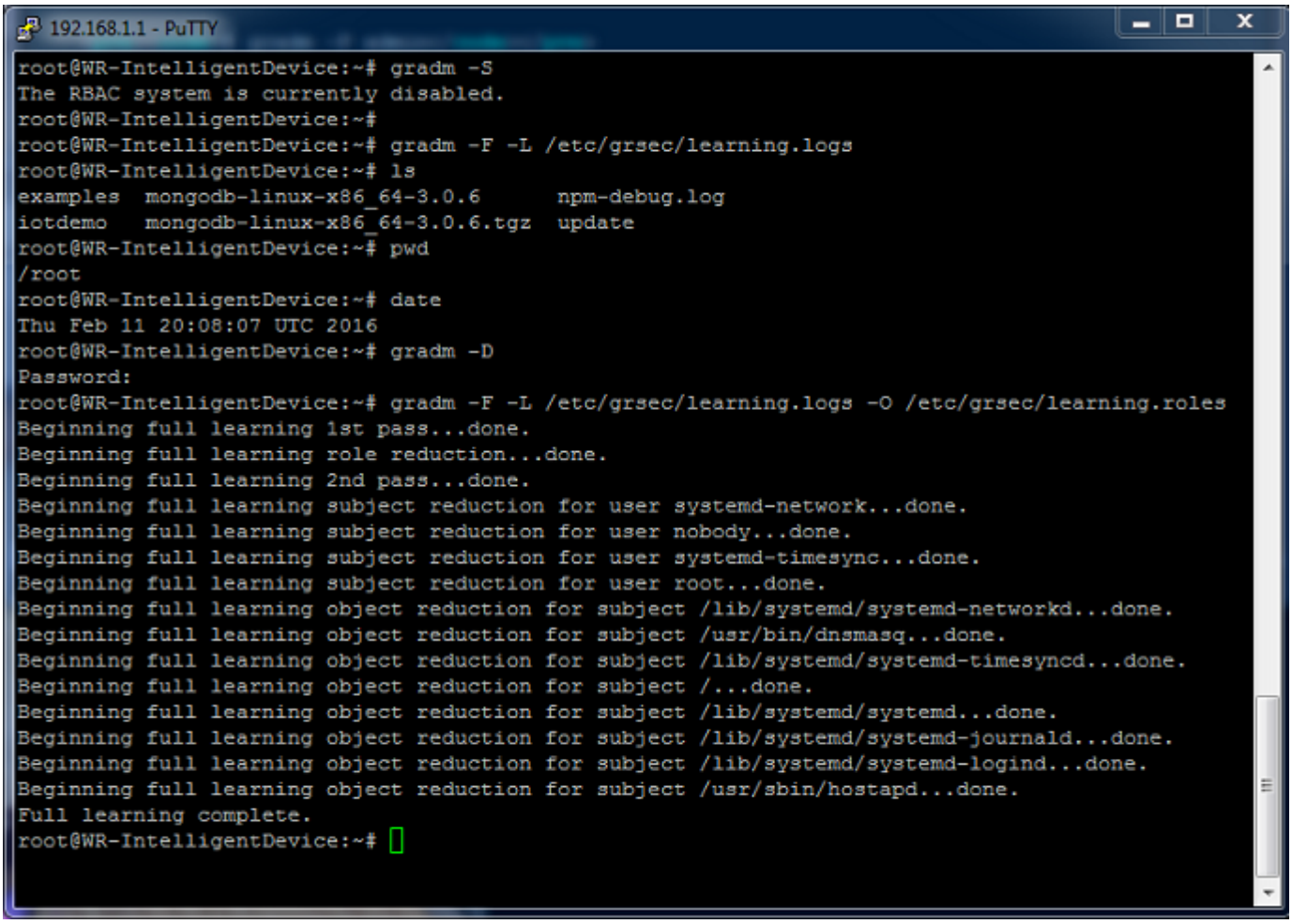
Learning Mode for Role based access – RBAC requires policy to be created to ensure that system is aware of known execution, this can be done by enabling GR-SEC in learning mode or manually updating the policy file.

Grsecurity's learning mode can be used on a per-subject or per-role basis, as well as system-wide. When using the learning mode on a single process or role, the rest of the system remains protected as defined by the policy. The learning mode can learn all things that the RBAC system supports: files, capabilities, resources, what IP addresses make use of each role, and socket usage. The learning system performs intelligent reduction of filesystem and network access to reduce policy size, increase readability, and reduce the amount of manual tweaking needed later. Furthermore, the learning system enforces a secure base that is configurable. Once it learning mode GR-sec will log all the activities of system.

To enable full system learning, run gradm as root with the following options:

Please run few commands. In actual environment learning mode is usually 24-48 hours.

```
# gradm -S
# gradm -F -L /etc/grsec/learning.logs
# ls
# pwd
# date
# gradm -D
# gradm -F -L /etc/grsec/learning.logs -O /etc/grsec/learning.roles
```



You've now created a set of commands that are allowed by grsecurity. Let's now enabled grsecurity and test to see if the learned commands are allowed and all other commands are disallowed.

Enabling Grsecurity with Learned Rules

First backup the current policy file.

```
# cp /etc/grsec/policy /etc/grsec/policy.bak
# cp /etc/grsec/learning.roles /etc/grsec/policy
```

Now enable grsecurity.

```
# gradm -E
```

Test the Learned Rules

```
# ls
# pwd
# cat
# ifconfig
# date
# gradm -D
```

Gateway Security

Node-RED with HTTPS

HTTPS provides SSL encyption for all data passed between client and server. Below you will learn how to secure your Node-RED server, and all web requests to it, with HTTPS. More info can be found at https://nodejs.org/api/https.html

Generate Certificates

First you will generate a private key and certificate using openssl.

Make a serial connection to your Edison and enter in the following commands:

```
$ cd ~

$ openssl req -newkey rsa:2048 -nodes -keyout privatekey.pem -x509 -day
s 365 -out certificate.pem
```

Answer the prompts with your info, or hit enter to leave as blank.

you now have a private key and certificate in your home directory.

Edit Node-RED Settings

In order to enable HTTPS on your Node-RED server you must edit the settings file.

Edit settings.js in user Node-RED user directory:

```
$ vi ~/.node-red/settings.js
```

- Hit "a" to enable editing
- On line 19 remove the "//" to uncomment the line.
- On lines 107-110 remove the "//" at the start of each line to uncomment the lines.
- Hit Esc, type :wq, and hit enter to save and exit vi.

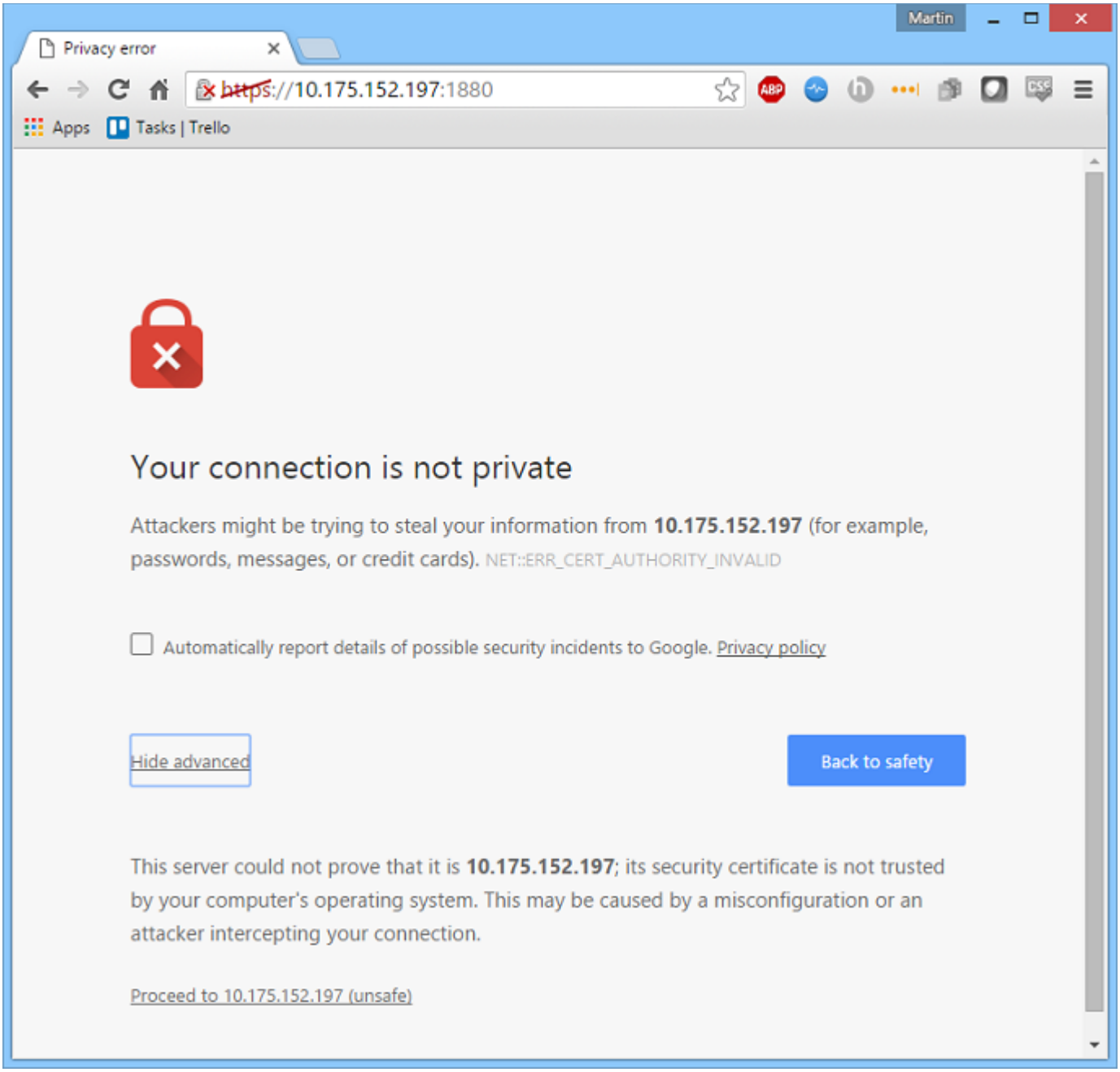
```
    //         password: "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxW
    //         permissions: "*"
    //     }]
    //},

    // To password protect the node-defined HTTP endpoints (httpNodeRoot), or
    // the static content (httpStatic), the following properties can be used.
    // The pass field is a bcrypt hash of the password.
    // See http://nodered.org/docs/security.html#generating-the-password-hash
    //httpNodeAuth: {user:"user",pass:"$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6
    //httpStaticAuth: {user:"user",pass:"$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNb

    // The following property can be used to enable HTTPS
    // See http://nodejs.org/api/https.html#https_https_createserver_options_req
    // for details on its contents.
    // See the comment at the top of this file on how to load the `fs` module us
    // this setting.
    //
    https: {
        key: fs.readFileSync('privatekey.pem'),
        cert: fs.readFileSync('certificate.pem')
    },

:wq
```

Run Node-RED server

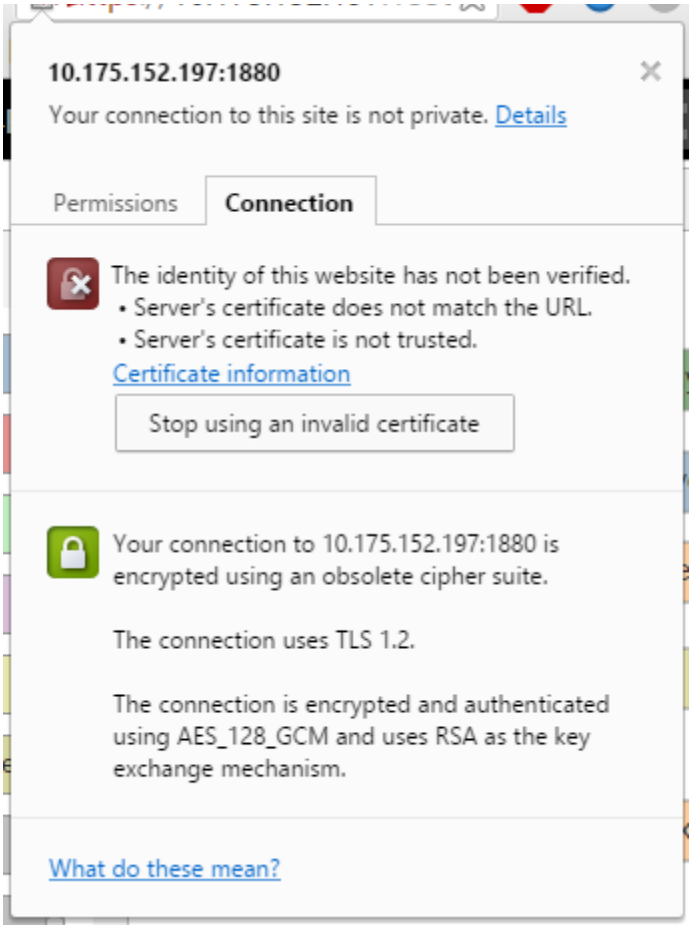
Launch Node-RED with:

```
$ cd ~

$ node-red
```

Navigate to the Node-RED server in your browser at https://edison-ip:1880 , note that you must replace edison-ip with the actual ip address of your Edison.



Since the SSL certifiacte you are using is self signed your browser will throw a warning. Click advanced and select "Proceed to xx.xx.xx (unsafe)"



Your Node-RED server now uses TLS 1.2 to secure the connection.

You should now have a Node-RED HTTPS server running.

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:

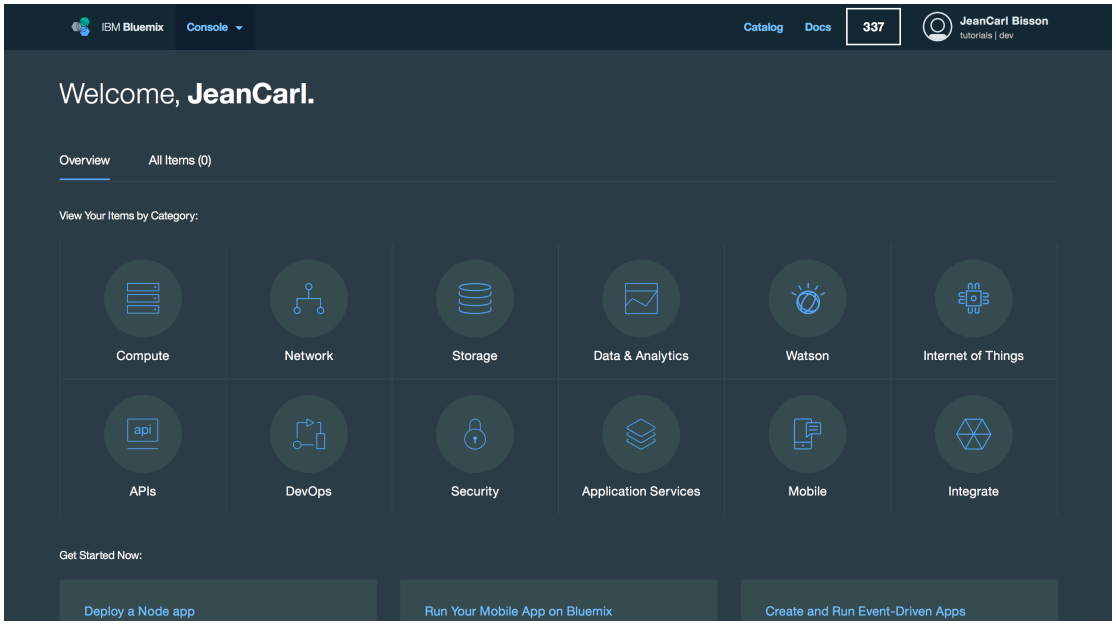
https://software.intel.com/en-us/blogs/2015/11/04/announcing-18-new-how-to-intel-iot-code-samples

Please wait for IBM to finish their presentation before moving on!

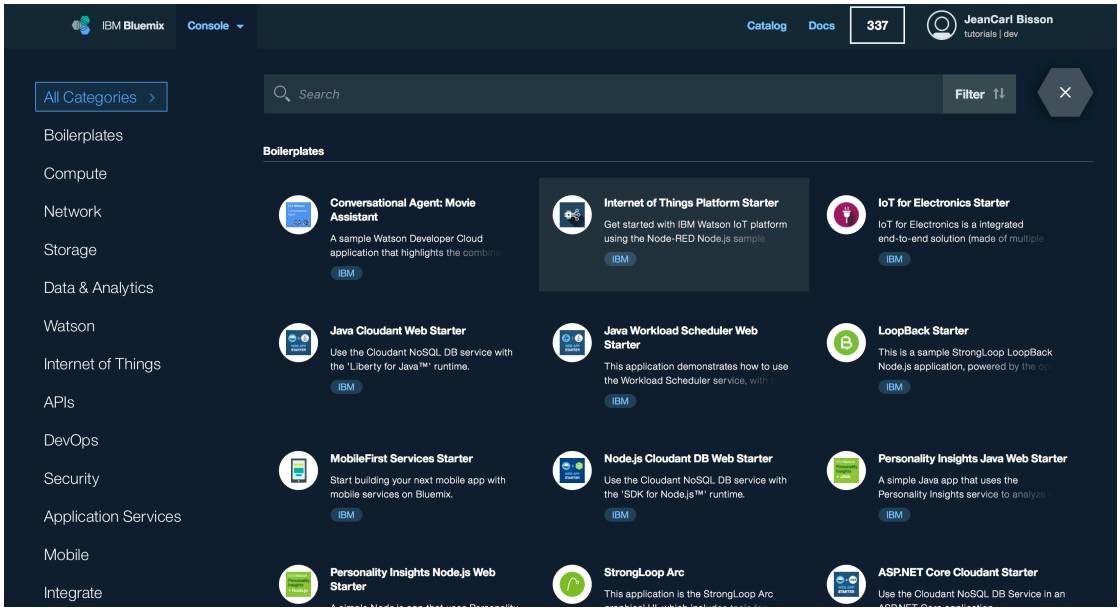
Creating an IoT application in IBM Bluemix

In this section, we will create an Internet of Things boilerplate application in IBM Bluemix and use it to receive temperature values from a simulated IoT temperature sensor.

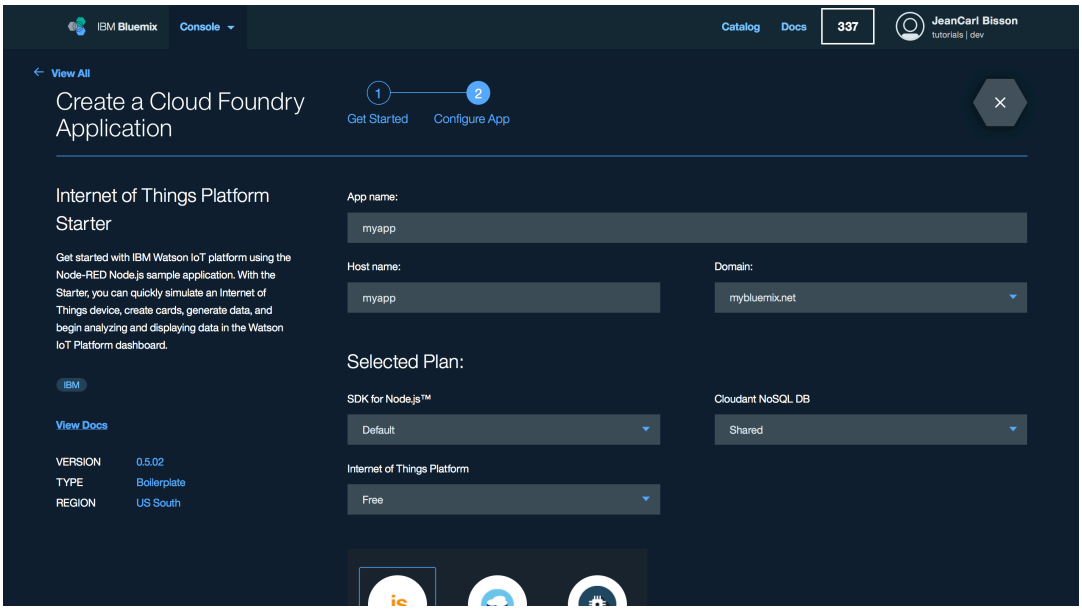
1. Open a web browser and go to your IBM Bluemix dashboard, http://bluemix.net. This is the IBM Bluemix dashboard where you have access to creating Cloud Foundry apps, IBM Containers, Virtual Machines, and a variety of Services. We are going to create a Cloud Foundry application today. To access the catalog, click on the **All Items tab** and then the blue **+** hexagon on the top right.



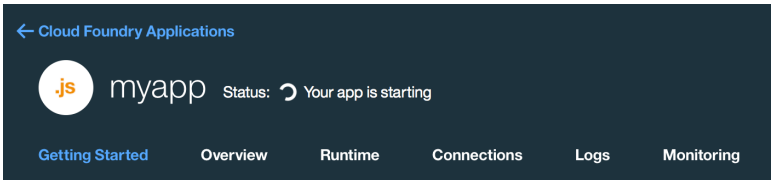
2. IBM Bluemix offers a handful of boilerplate applications that you can create and get started quickly. The Internet of Things Starter boilerplate includes Node-RED, a Cloudant NoSQL database, and the SDK for Node.js. Under Boilerplates, select the **Internet of Things Platform Starter** boilerplate tile.



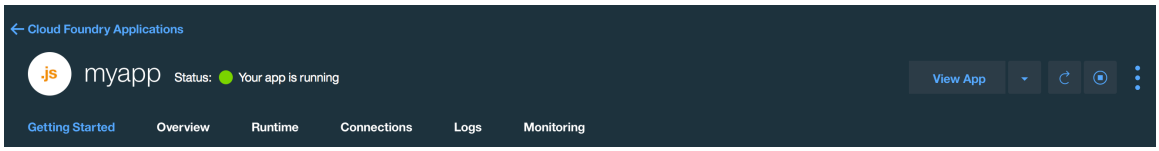
3. Pick a unique name for your application. If you choose ***myapp***, your application will be located at http://***myapp***.mybluemix.net. There can only be one "***myapp***" application registered in IBM Bluemix. You can try adding your initials in front of the host if the host you choose is already taken by someone else. Click on **Create** to create the application instance.



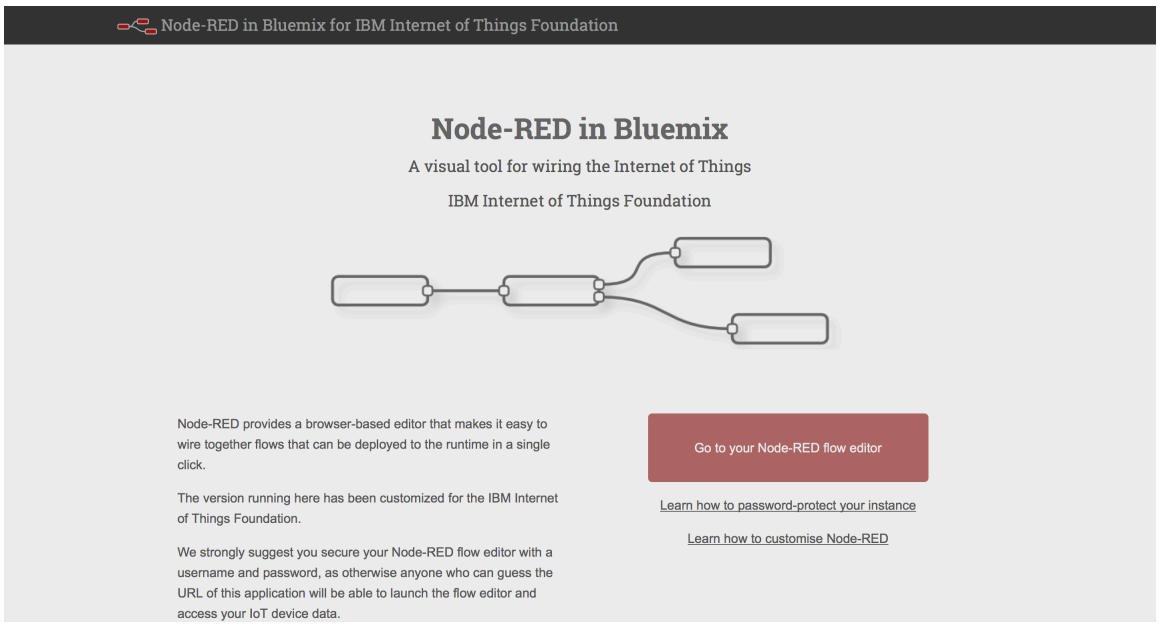
4. IBM Bluemix will create an application in your account based on the services in the boilerplate. This is called staging an application. It can take a few minutes for this process to complete. While you wait, you can click on the **Logs** tab and see activity logs from the platform and Node.js runtime.



5. When the application is running, click on the **View App** button to open the application in a new browser tab.

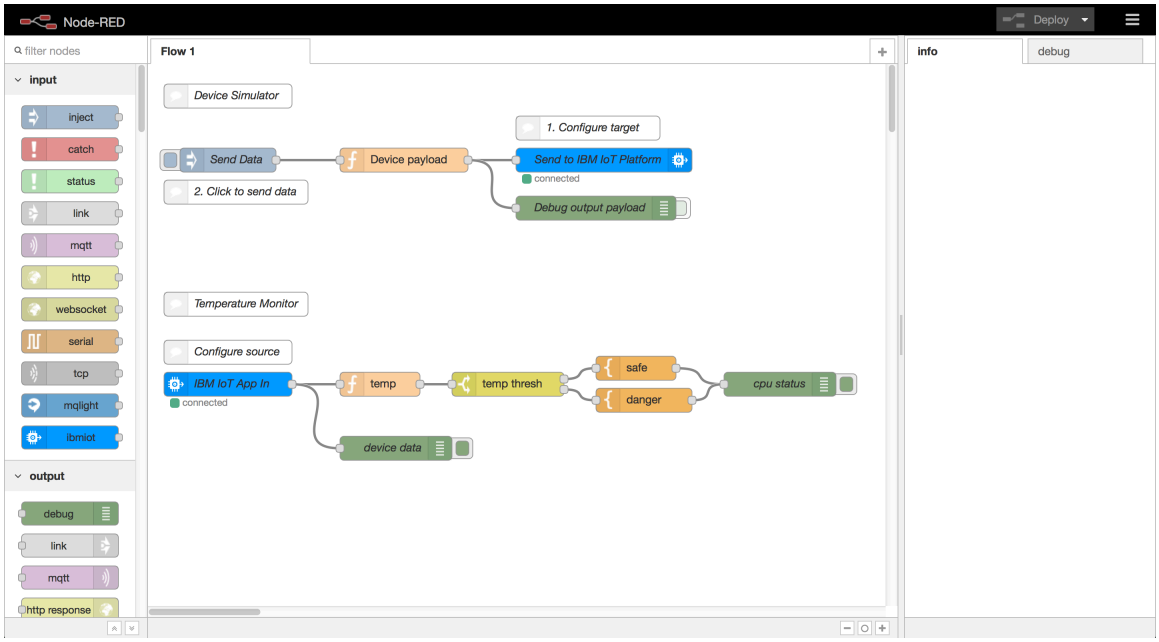


6. This is Node-RED start page. Node-RED is an open-sourced Node.js application that provides a visual editor that makes it easy to wire together flows. Click on the red button **Go to your Node-RED flow editor** to launch the editor.

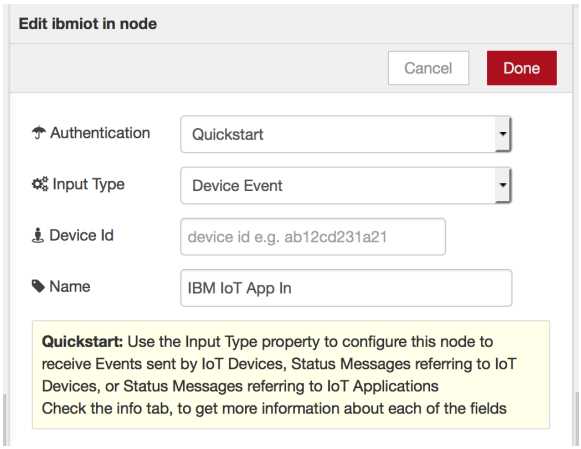


7. On the left side is a palette of nodes. Each node performs a defined function, configurable by dialog windows or by modifying the msg input. You can click on a node in the palette and find out what it does in the info tab on the right side of the screen. Drag and drop nodes and connect them together in the middle pane, called a canvas. Double click on nodes in the canvas to customize settings used by the node. Connect two or more nodes via the grey knobs on the left and right sides of the node, constructing what is called a flow. A flow is executed from left to right and is completed when the last node is reached. A flow can split off into two or more flows, for example, with a switch node. Two flows can also share a node or a flow, reusing the same logic in multiple scenarios by connecting their grey output knobs to the shared node's left grey input knob.

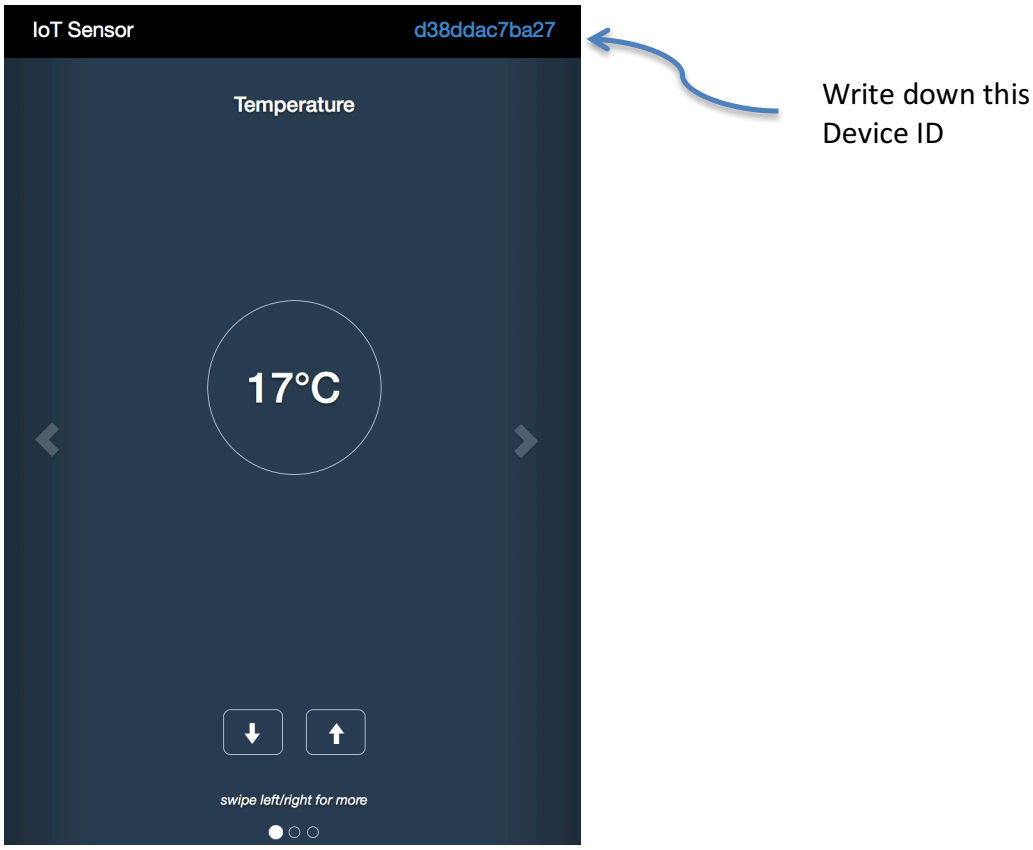
   Let's begin by configuring the Watson IoT App In node. Double click on the Watson IoT App In node.



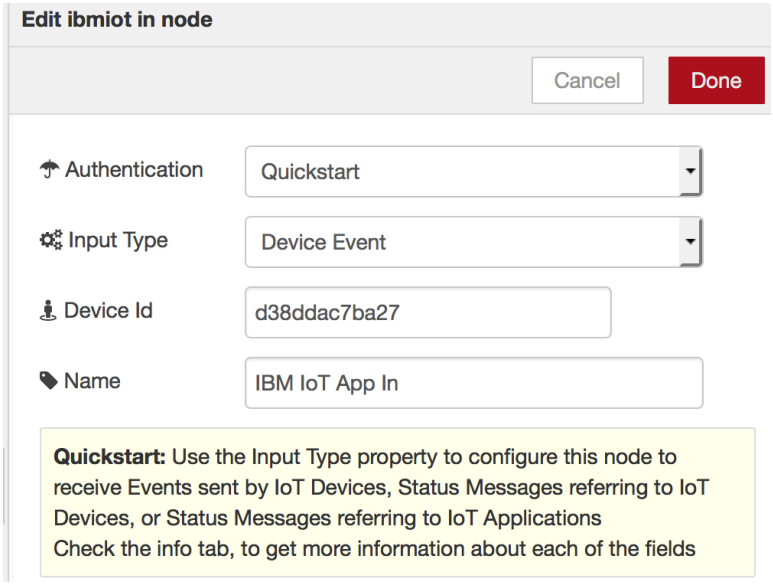
Nodes can be customized in different ways depending on what they do. The IBM IoT node begins a flow when a message is received from a specific device via the Watson IoT Platform. We need a Device Id of a device that is connected to the IoT Platform. We could use a real device, or we can simulate devices.



8. To get a device ID from a simulated device, visit http://ibm.biz/iotsensor. In the upper right is an alphanumeric value. This simulated temperature sensor sends the temperature to the Watson Internet of Things Platform service using this device ID.



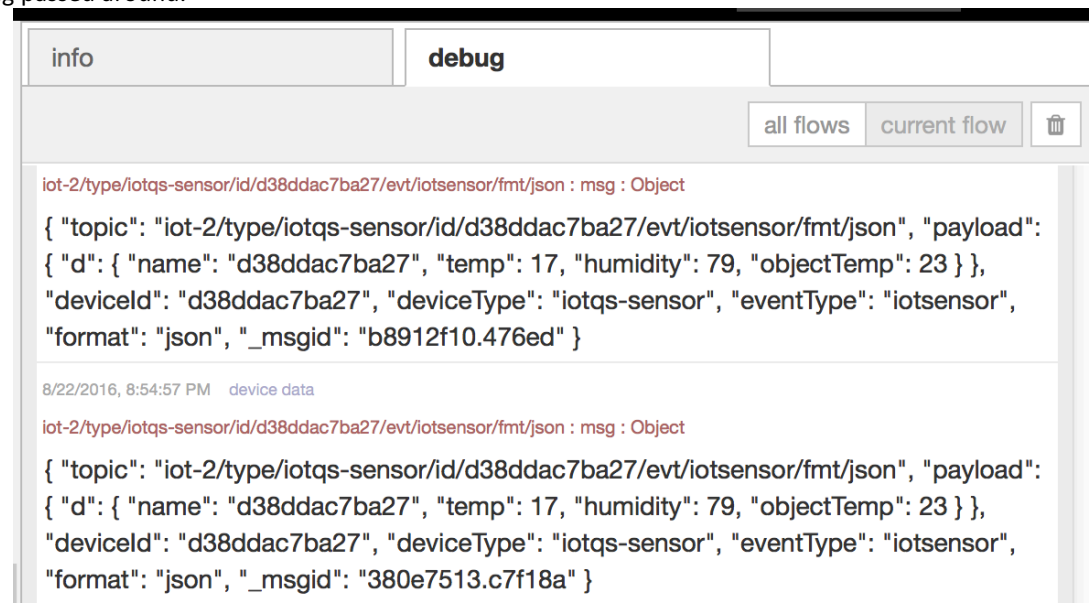
9. Copy this alphanumeric device ID into the Device ID field in the Node-RED application as shown below. Click **Done**.

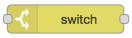


10. Click on the 

button in the top right of the screen to save and deploy your changes.

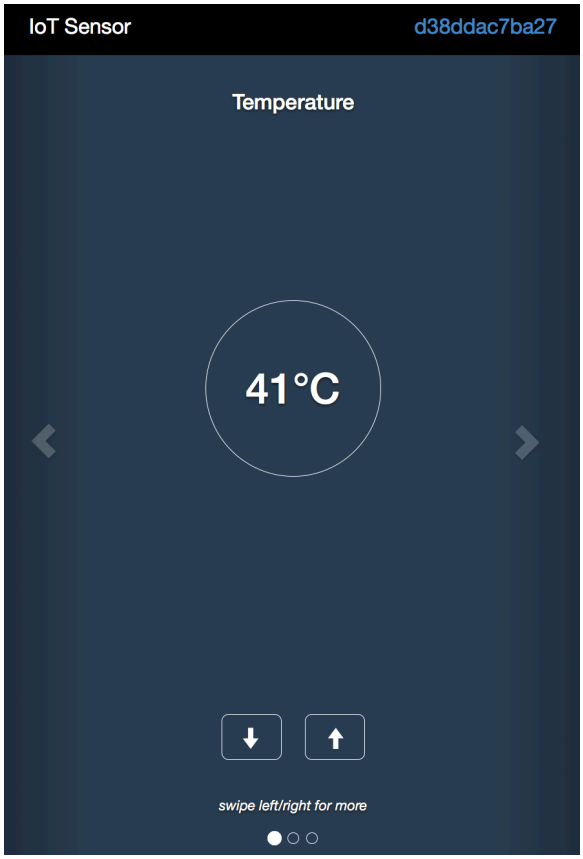
11. Click on the debug tab in the right-hand pane. Every second, the simulator emits a device event to the Watson IoT platform with temperature and other data. The Node-RED application subscribes to these events and the IBM IoT in node triggers the flow with the temperature data in the message. When the debug nodes are processed, contents of the message object are in the debug tab. Adding debug nodes can be helpful when something doesn't work right and you want to see the values being passed around.



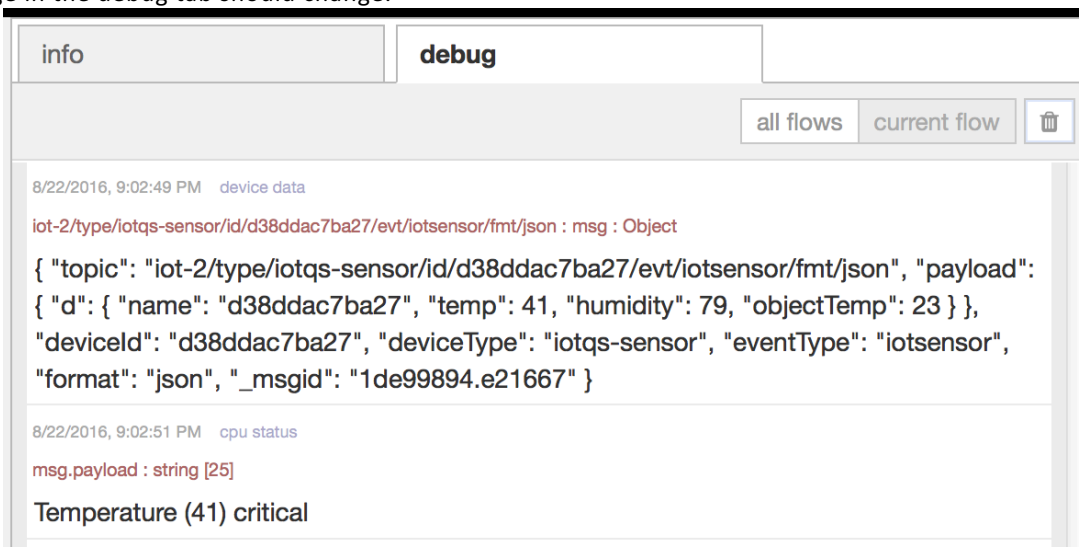
12. The yellow node is called a 

node. You can program logic using a switch node and split a flow into two or more flows based on a property's value. In this example, if the temperature is less than or equal to 40°C, it is considered "safe" and continues with the flow to the template labeled safe. If the temperature is greater than 40°C, it is considered "danger[ous]" and continues with the flow to the template labeled danger.

13. To trigger the flow labeled danger, increment the temperature using the up arrow on the simulated temperature gauge.



The message in the debug tab should change.



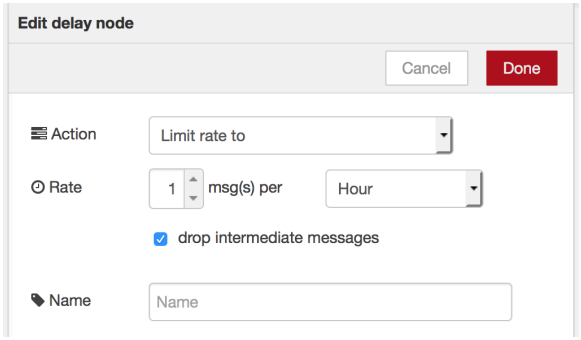
Connect to Twitter and Tweet High Temperature

In this section, we will connect a Twitter account and use the Twitter account to tweet when the temperature from the temperature sensor is “dangerous”. This section is optional and may be skipped.

1. Sign up for a Twitter account at http://twitter.com . If you already have a Twitter account, proceed to step 2.

2. Add a 

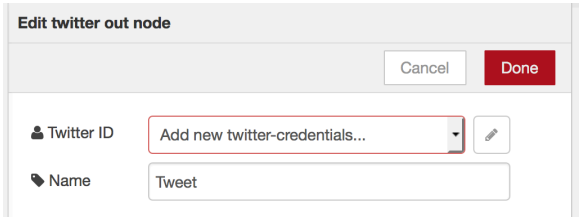
node as shown below.

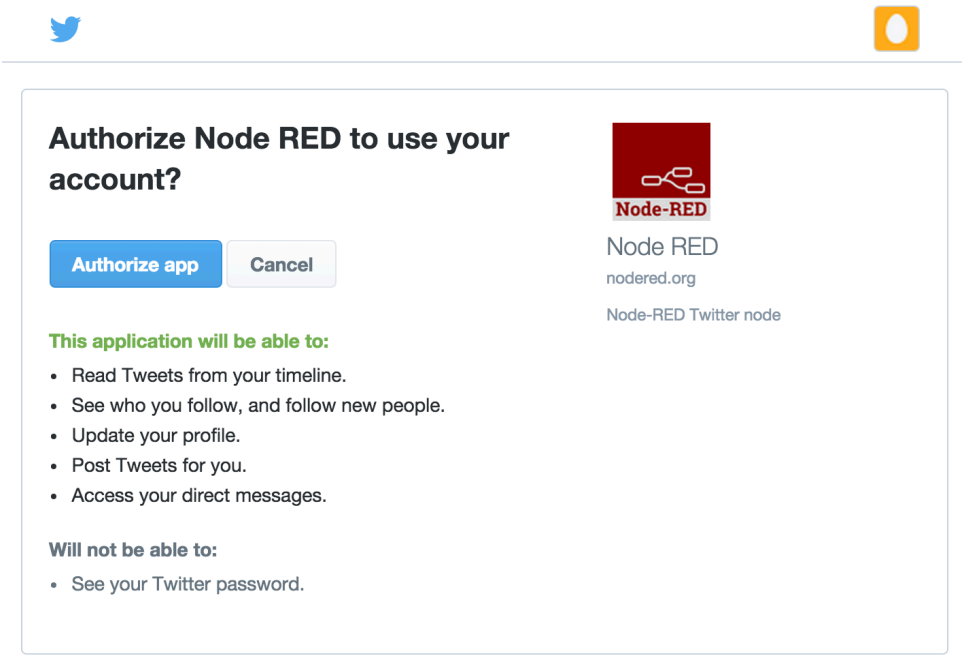


The delay node limits how often the flow is run. Since the temperature is dangerous every second, without this node, a tweet would be sent every second. With this node limiting messages to once an hour, the Twitter node will send a tweet once an hour, dropping any additional messages during the hour timeframe.

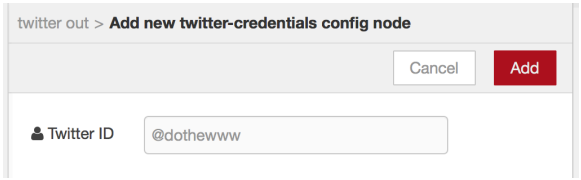
3. Add a 

node. Click on the pencil button and authenticate with Twitter. The account you sign in with will be used to send tweets.

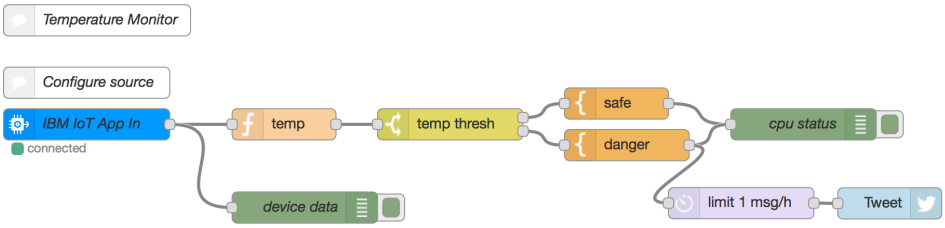




Return back to the node configuration. The settings for the Twitter node should have your username set. Click **Add**.



4. Connect the nodes as shown below.



Get the code:
ibm.biz/BdresH

5. Click on 

to save and deploy the changes.

6. Since the temperature is above 40°C, the switch statement will continue to the “danger[ous]” function, compose a message, and pass it to the Twitter node. The Twitter node uses this message as the content for the tweet.

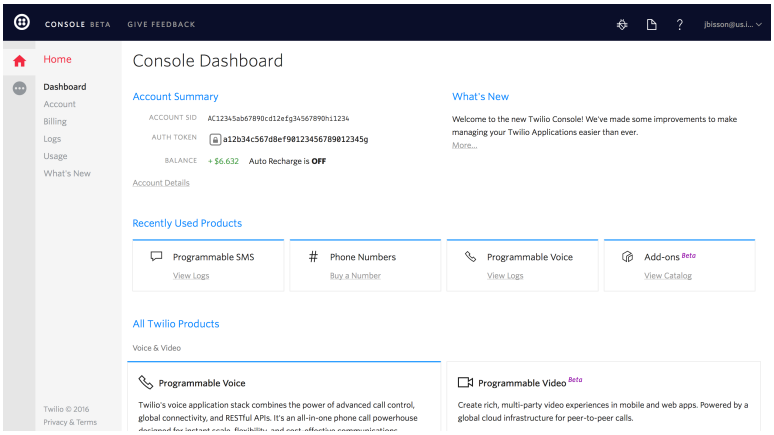
7. Visit the Twitter timeline for the user you authenticated with and verify the message has been tweeted.

Connect to Twilio and Text High Temperature

In this section, we will connect a Twilio phone number to the application and send a text message notification when the temperature is at least 40°C. This section is optional and may be skipped.

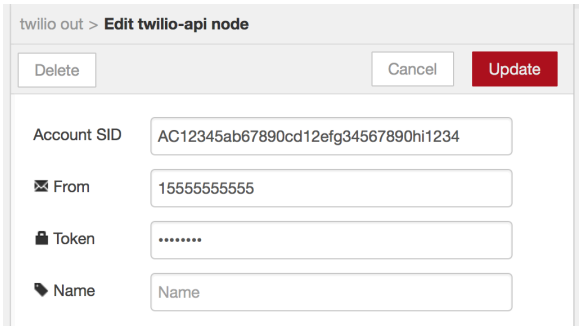
1. Sign up for a Twilio account at http://twilio.com . If you already have a Twilio account, sign in. Create a Twilio phone number that will be used in step #3.

2. In the Console Dashboard, click on the lock icon next to Auth Token. Copy the Account SID and Auth Token.

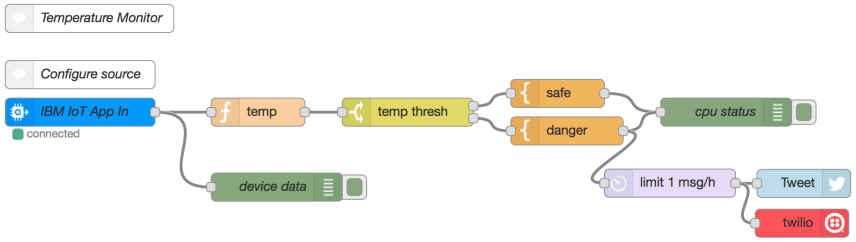


3. Add a 

node. Click on the Pencil to provide your **Account SID** and **Auth Token** from step #2, and **From** phone number using the Twilio phone number from step #1. Fill in the **SMS to** textbox with a phone number that will be texted to.



4. Connect the nodes together as shown.



Get the code:
ibm.biz/Bdress

5. Click on 

to save and deploy the changes. You should receive a text message shortly.

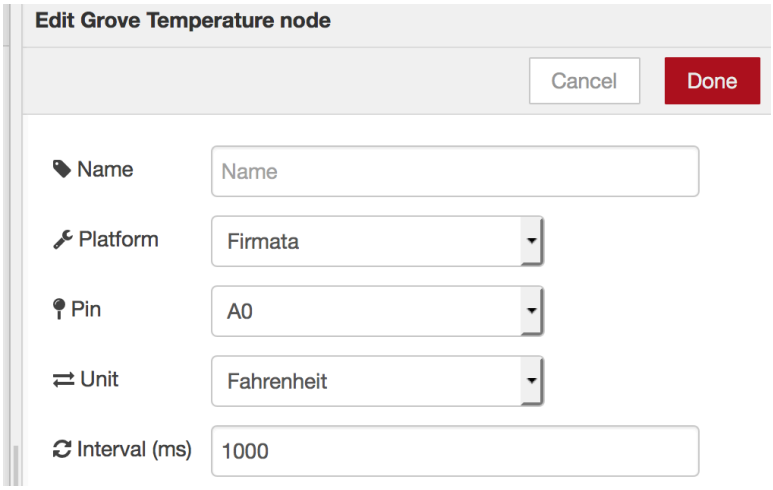
Create a Node-RED flow on Intel NUC

In the first section we used a simulated device to get started with the Internet of Things Platform. When you have simulated what your device will do, then you can invest in making the hardware and hopefully reduce the cost incurred along the way. The Intel NUC board has made prototyping IoT devices easy. The Grove Starter Kit comes with a variety of sensors that can be connected to the Intel NUC and Arduino 101 and, along with custom logic, can bring developing hardware solutions down to a level anyone can build quickly and easily.

In this section, we will connect to the Node-RED application running on the Intel Edison board, capture the value of a real temperature sensor, control an LCD screen, control a LED light, and send the temperature to the Watson Internet of Things Platform.

1. To get started, drag a 

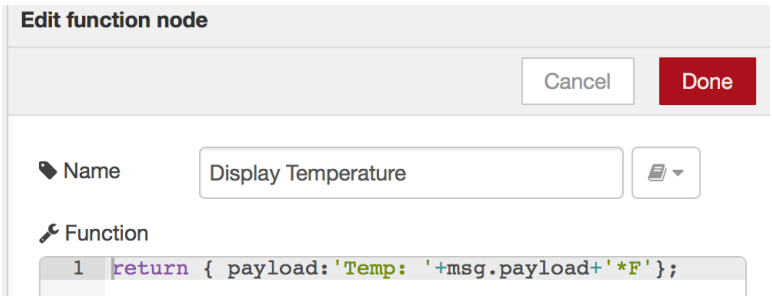
Grove Temperature node onto the canvas. Double click on the Grove Temperature node and customize as shown below.



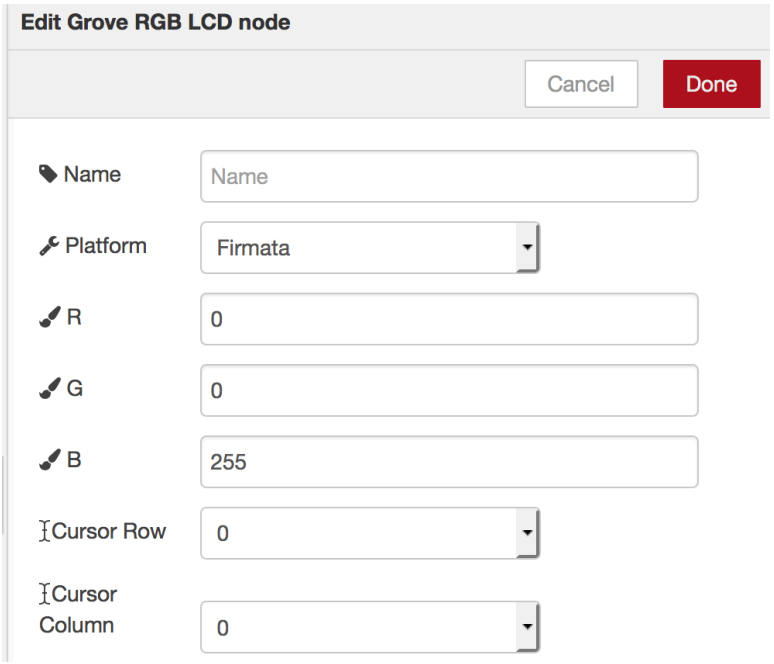
This will configure the temperature node to listen on Port A0, and return Fahrenheit temperature value.

2. Add a 

function node as shown below.



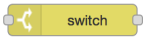
3. Add a  Grove RGB LCD node as shown below.



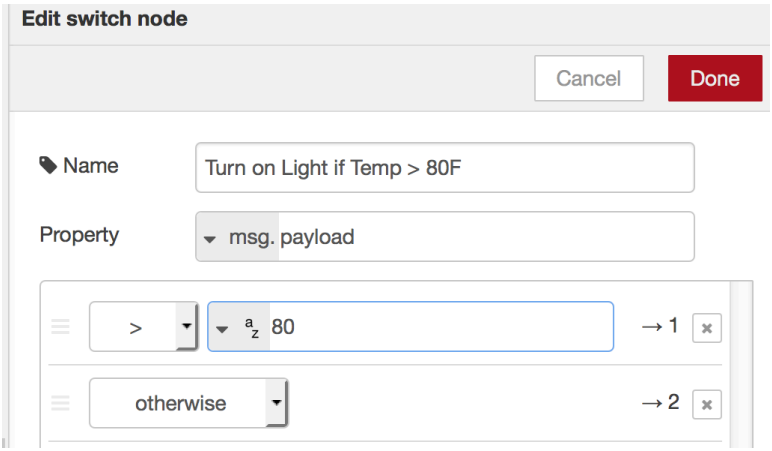
This will display the temperature value on the LCD connected to the I2C port.

4. Connect the nodes together as shown below.

5. Connect a Grove LCD screen to the I2C port on the Grove shield connected to Arduino.
6. Click on the  button in the top right of the screen to save and deploy your changes. The LCD screen should display the temperature in Fahrenheit. The temperature value screen will update once every second when the flow is activated by the inject node.

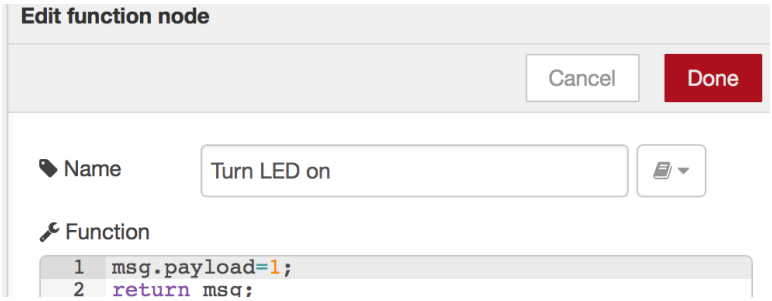
7. Next, we will activate a LED light when the temperature exceeds 80°F. Add a 

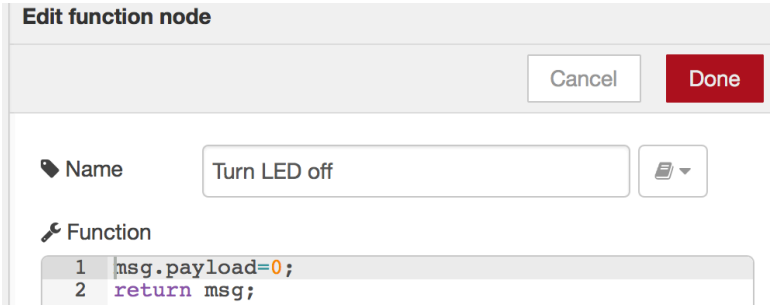
node and connect it to the temperature node as shown below.



8. Add two 

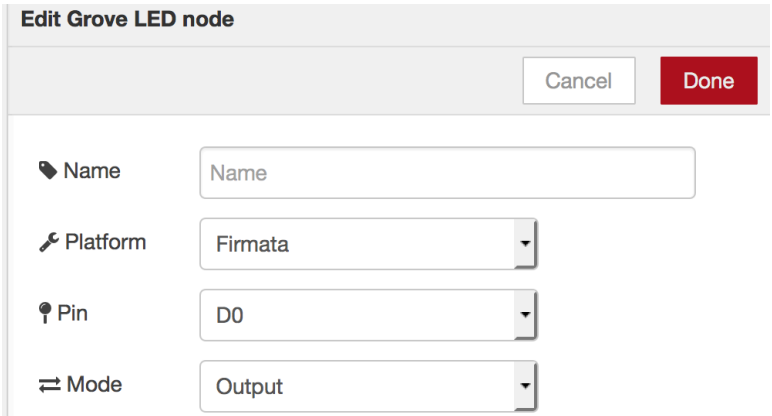
nodes, one to turn the LED on, and one to turn the LED off as shown below.



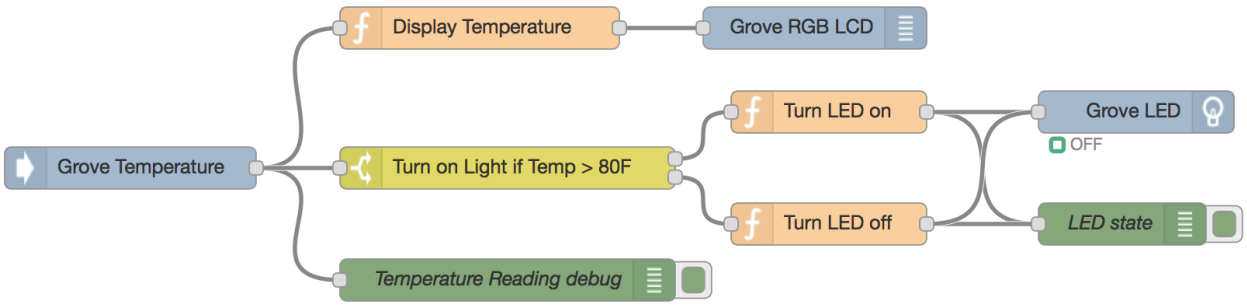


9. Add a 

Grove LED node as shown below. This will control the LED connected to port D0.



10. Connect the nodes together as shown below.



This flow is available at https://ibm.biz/BdrgCw

11. Connect a LED light to the D6 port on the Grove base shield connected to the Intel Edison.

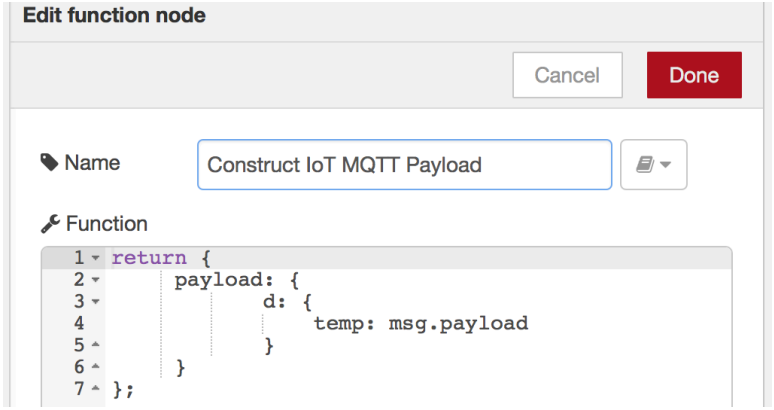
12. Click on the Deploy button in the top right of the screen to save and deploy your changes. When the temperature reaches above 80°F, the LED light will turn on. Otherwise, the LED light will turn off.

Sending Temperature Sensor to IoT Platform

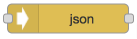
In this section, we will connect the Node-RED application running on the Intel NUC, Arduino 101 and Grove Shield to the Watson Internet of Things Platform service. We will send the temperature value from the Grove temperature sensor as device events so that the Node-RED application in IBM Bluemix can react to high temperatures.

1. Add a 

node as shown below



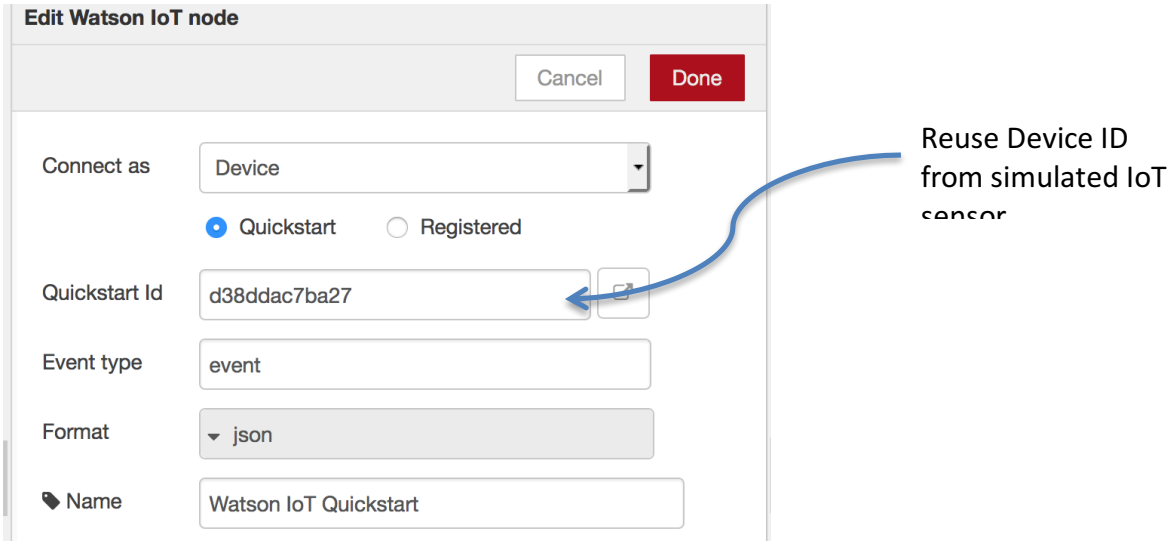
2. This JavaScript function constructs and returns a JSON object. The JavaScript object has a property named **payload**, which has property named **d**, which contains a property named **temp**. We use the Celsius value from the temperature sensor and assign it to the property **temp**.

3. Add a 

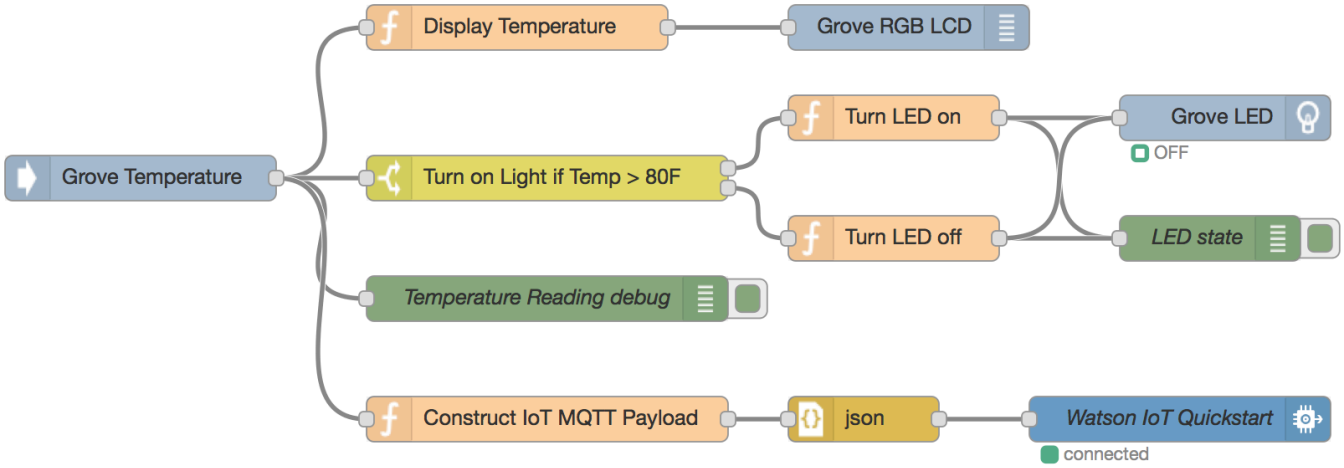
node. This node does one of two things. When passed a JSON string, it will attempt to parse it into a JSON object. When passed a JavaScript object, it returns a JSON string. Since we're passing in a JavaScript object, it will return a JSON string.

4. Add a 

node as shown below. You can reuse the device Id generated by the simulated temperature sensor, or, change the device ID here and in the Watson IoT in node of the Node-RED application in the cloud. **If you reuse the device ID, close the simulated temperature sensor window to stop the simulated temperature values from conflicting with the real temperature data.**



5. Connect the nodes together as shown below.



This flow is available at https://ibm.biz/BdrgCw

6. Click on  to save and deploy your changes.

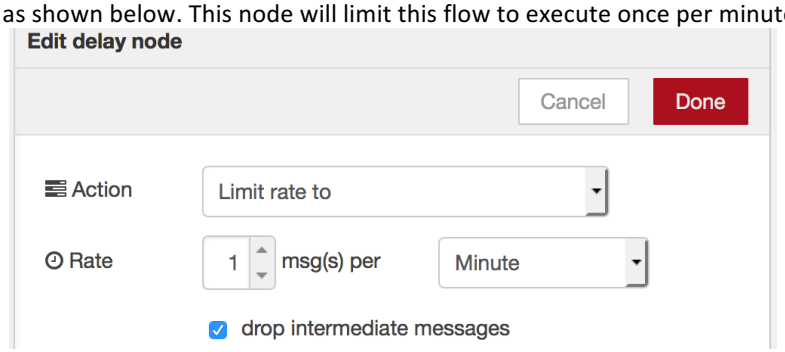
7. Return to the Node-RED application in IBM Bluemix. You should see the real temperature being reported.

Store Temperature Into Cloudant NoSQL Database

This section will show how to add a Cloudant NoSQL database to the Node-RED application and store temperatures reported (one per minute). This functionality can be useful to run historical analysis (outside the scope of this lab) or find patterns over time. This section is optional and can be skipped. However, it is a prerequisite for the **Retrieve Temperatures From Cloudant NoSQL Database** section.

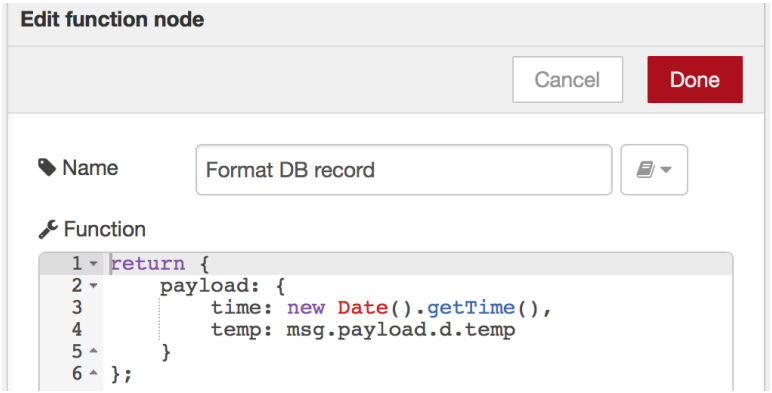
1. Add a 

node as shown below. This node will limit this flow to execute once per minute.



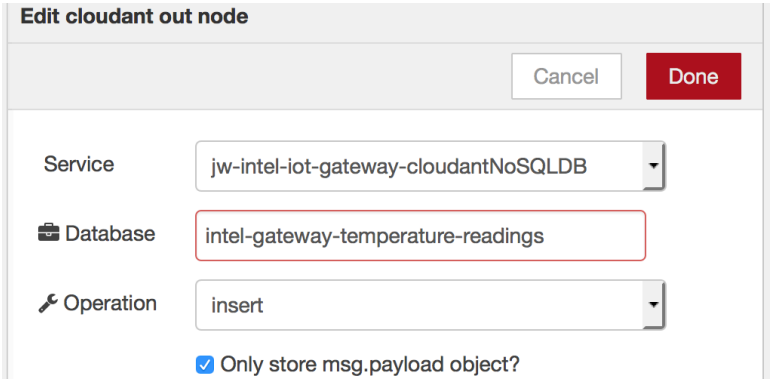
2. Add a 

node as shown below.

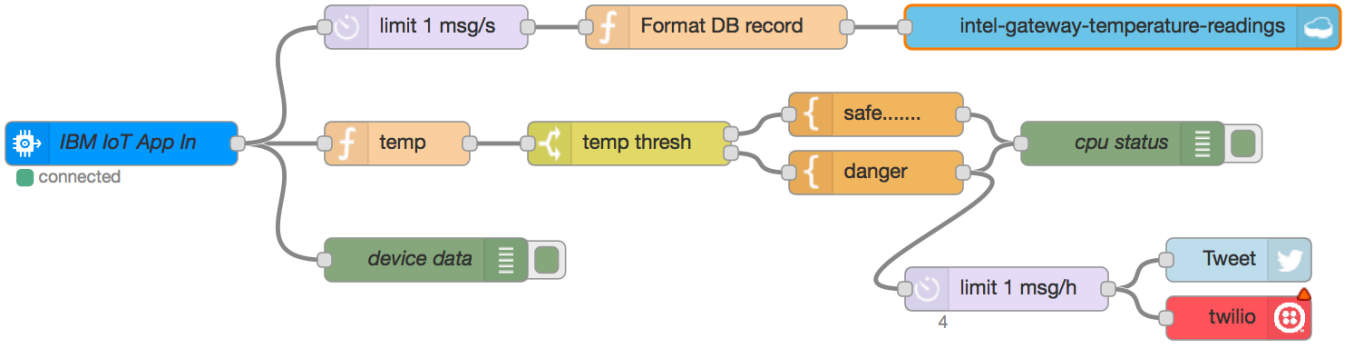


3. Add a 

node as shown below.



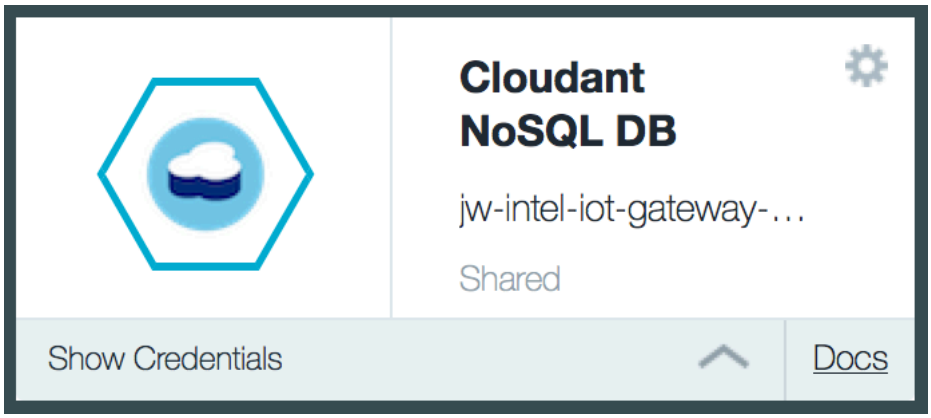
4. Connect the nodes together as shown below.



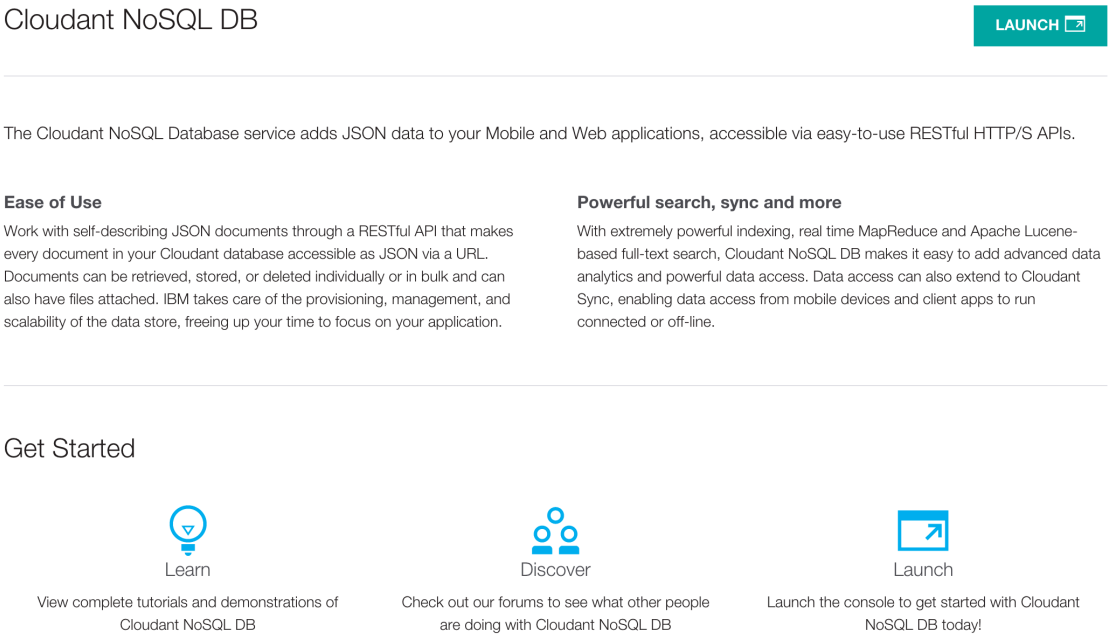
This flow is available at https://ibm.biz/BdrgQS

5. Click on  to save and deploy your changes.

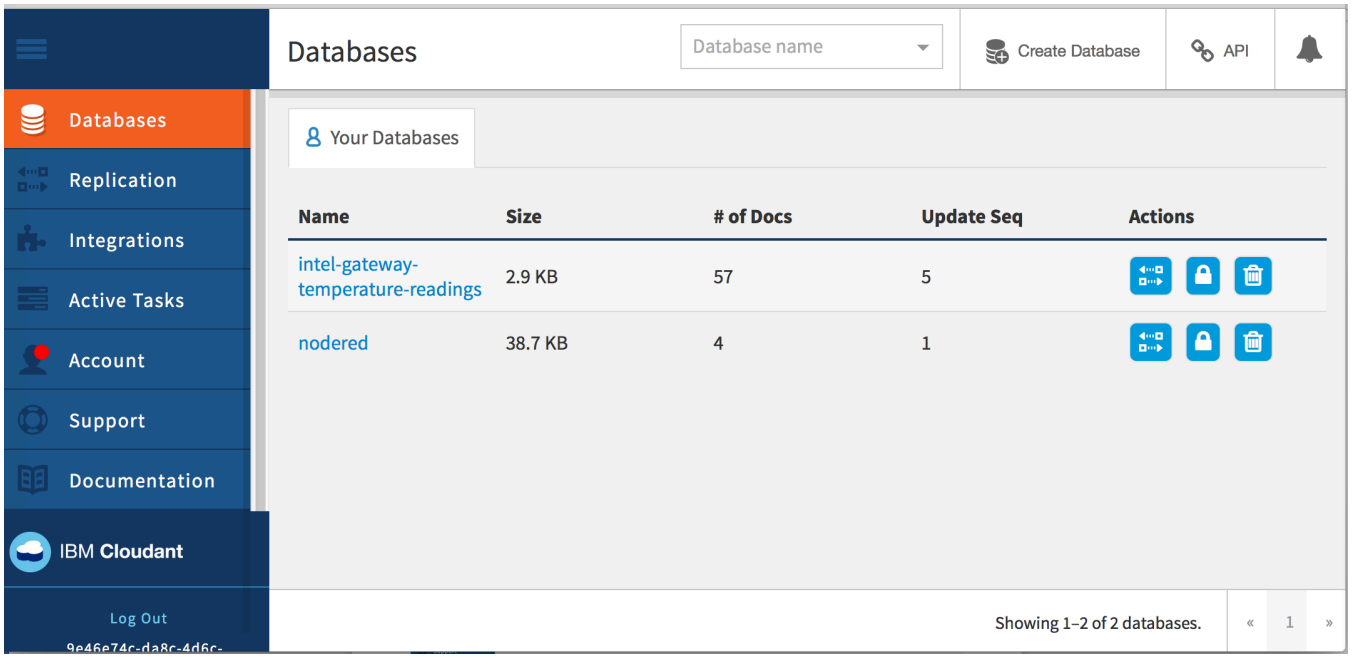
6. Go back to the IBM Bluemix dashboard and the **Services** tab. Click on the **Cloudant NoSQL DB** service tile.



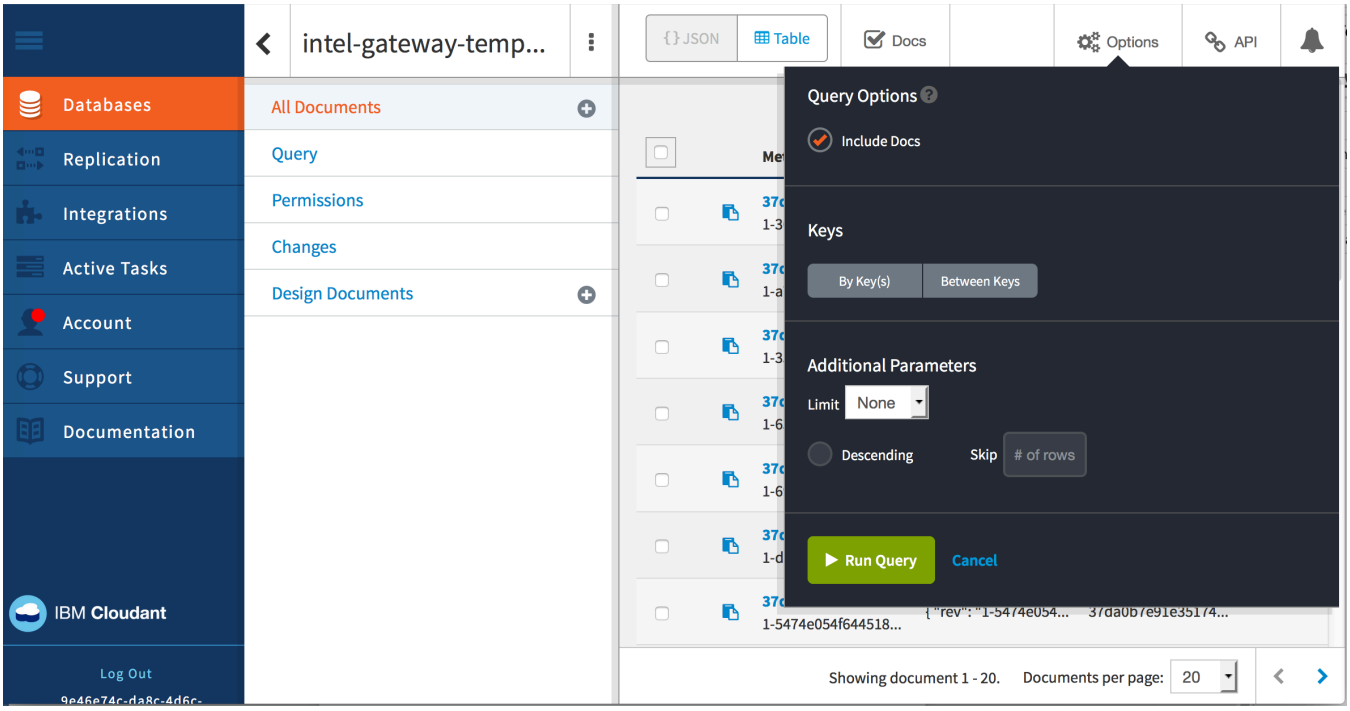
7. Click on the green **Launch** button.



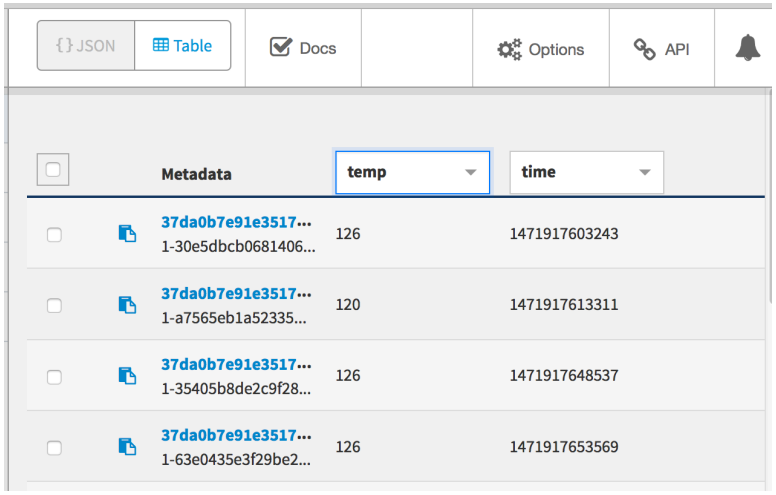
8. This is the Cloudant NoSQL database dashboard. A list of databases is displayed. The database named intel-gateway-temperature-readings contains documents representing each temperature event that has been stored by the Node-RED application. Click on the database named **intel-gateway-temperature-readings.**



9. To see the expanded view of the documents, click on **Query Options**, check the box next to **Include Docs**, and click on the green **Run Query** button.



10. Each box represents one document (in our case one temperature event) that contains the payload (time and temperature) we stored earlier.



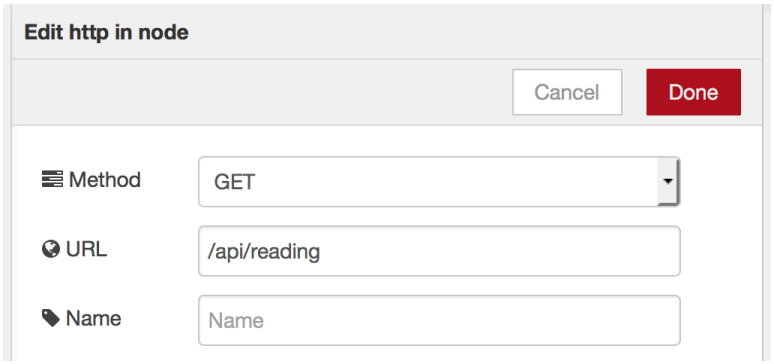
Retrieve Temperatures From Cloudant NoSQL DB

This section shows how to retrieve temperature data from a Cloudant NoSQL database and exposes it as a HTTP endpoint. This functionality can be useful to run historical analysis (outside of the scope of this lab) or find usage patterns over time. This section is optional and can be skipped. Completion of the section titled **Store Temperature Into Cloudant NoSQL Database** is required before beginning this section.

Now that we have a Cloudant NoSQL database containing reported temperatures, let's expose the data as an HTTP endpoint.

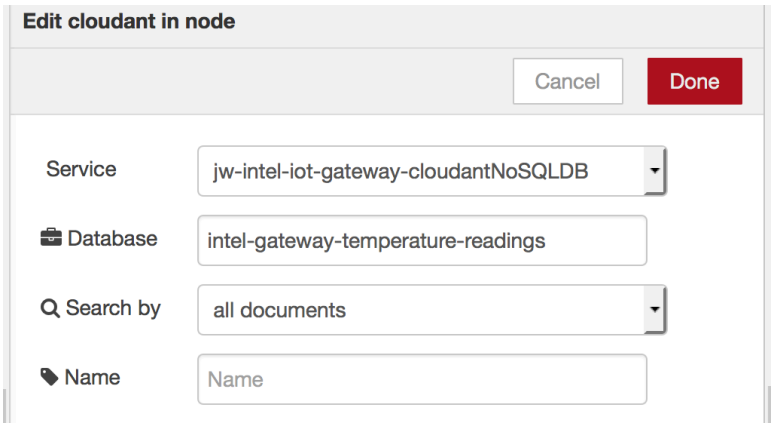
1. Add a 

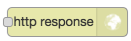
node as shown below.



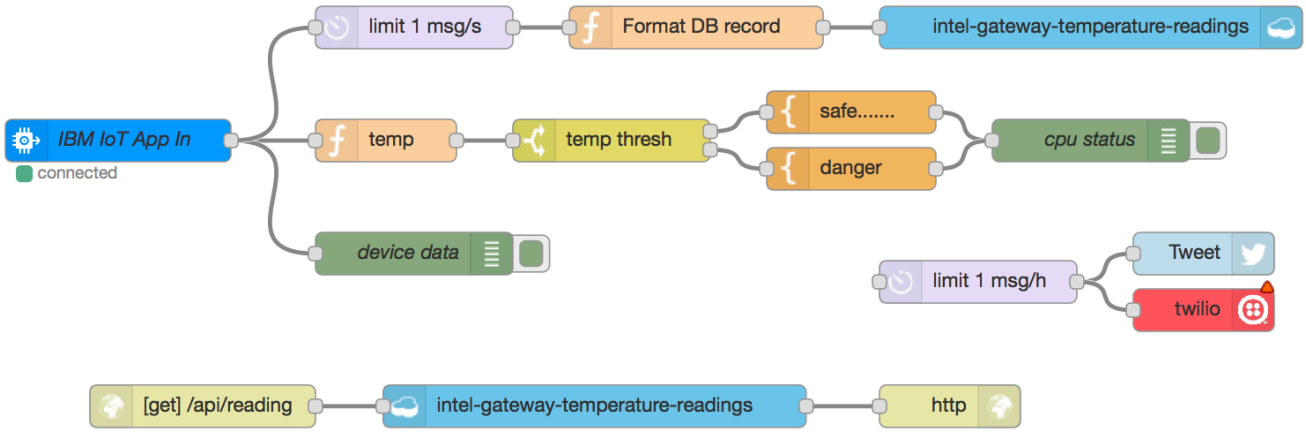
2. Add a 

node as show below.



3. Finally, add a 

node. Connect the nodes together as shown below. Download this flow: https://ibm.biz/BdrgQS



4. Click on  to save and deploy your changes.

5. Open a browser tab and visit your application's URL, appended by /api/reading. If you chose ***myapp*** when setting up your application, the URL would be:

   https://***myapp***.mybluemix.net/api/reading

   You should see data returned similar to the following:


```
[{"_id":"0bf60630164bd19c0bc74bef6e037da6","_rev":"1-cad3644fdbe75376b9998c6be5280fe2","time":1471922146114,"temp":129},
{"_id":"0bf60630164bd19c0bc74bef6e070b6a","_rev":"1-ae2aa69dd2fb9d4ab8ae48582f206a66","time":1471922161179,"temp":129},
{"_id":"0bf60630164bd19c0bc74bef6e0d72df","_rev":"1-c909094db72089d0da08f9eb97e2fe87","time":1471922186297,"temp":129},
{"_id":"0bf60630164bd19c0bc74bef6e11d7c9","_rev":"1-4da325037071066aae783cf27f99920a","time":1471922201380,"temp":129},
{"_id":"0bf60630164bd19c0bc74bef6e12e917","_rev":"1-f1b4e045bbb1d895beb146c50fba000d","time":1471922206405,"temp":124},
{"_id":"0bf60630164bd19c0bc74bef6e15a4dc","_rev":"1-532d6200c5c4c5be3fcb3d0a6b1367ec","time":1471922216456,"temp":129},
```

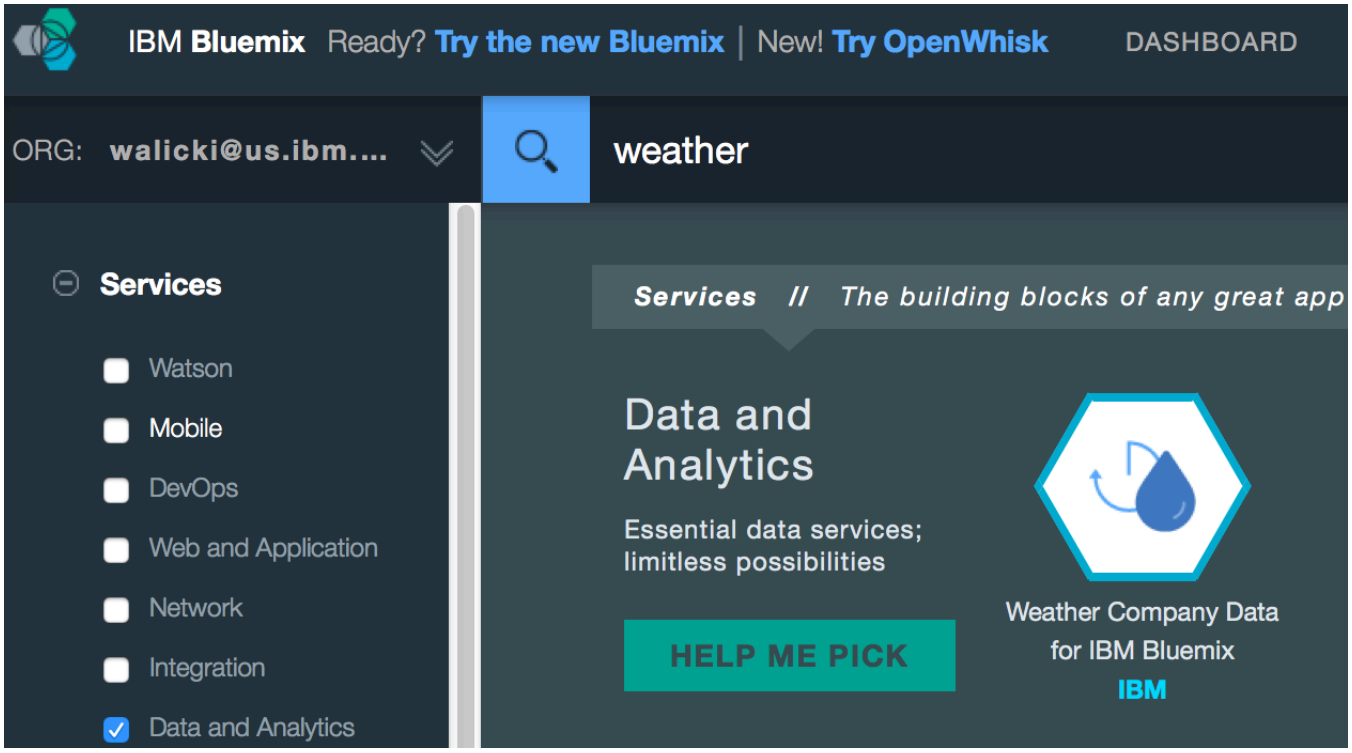

   The temperature values are returned in a JSON array. You can use this data in web applications or analytical tools. As you inject more data in the IoT example in Node-RED, this dataset will expand to include that data.

Send Alerts from Watson IoT to the Intel Gateway

In this section, we'll extend the temperature sensor LCD to include the outside temperature. The outside temperature will be retrieved via the Weather Company Data for IBM Bluemix service.

1. Click on the **+ Add a Service or API** card in the application overview for your Node-RED application in IBM Bluemix.

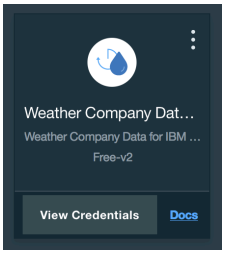
2. Select the **Weather Company Data for IBM Bluemix** under the Data & Analytics category.



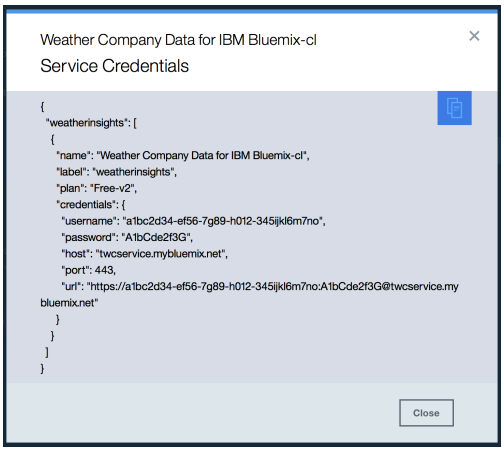
3. Click **Create** to add the service to your application.

4. When prompted to restage the application, click **Restage** to restart the application and update the environment with the credentials to the Weather service.

5. When the application has restarted, a third service tile will appear in the Connections tab. Click on **View Credentials** in the Weather Company Data service tile to show the service credentials.

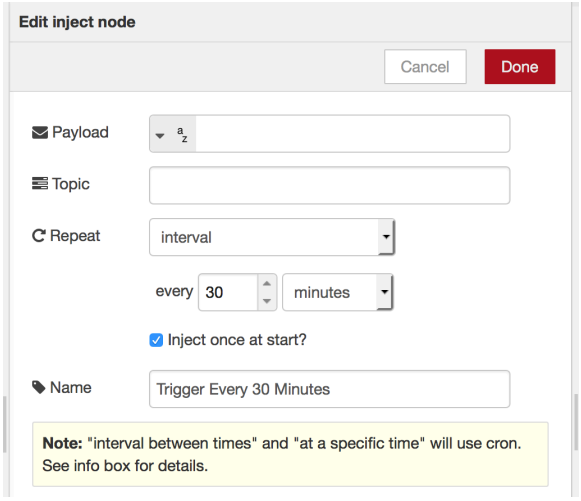


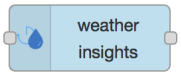
6. Copy the username and password values and save them for use in step #8.



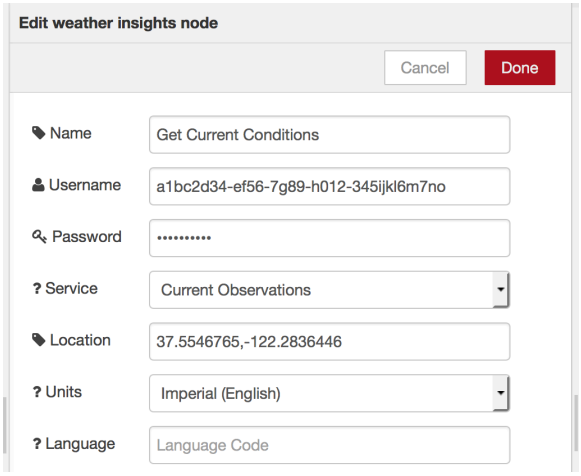
7. Add a 

node as shown below. This will trigger the flow once at startup, and then every 30 minutes.



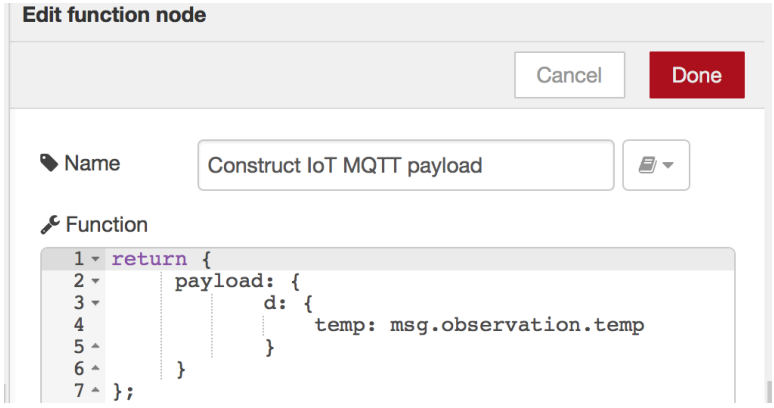
8. Add a 

node as shown below. Use the Weather Company Data API credentials from step #6.



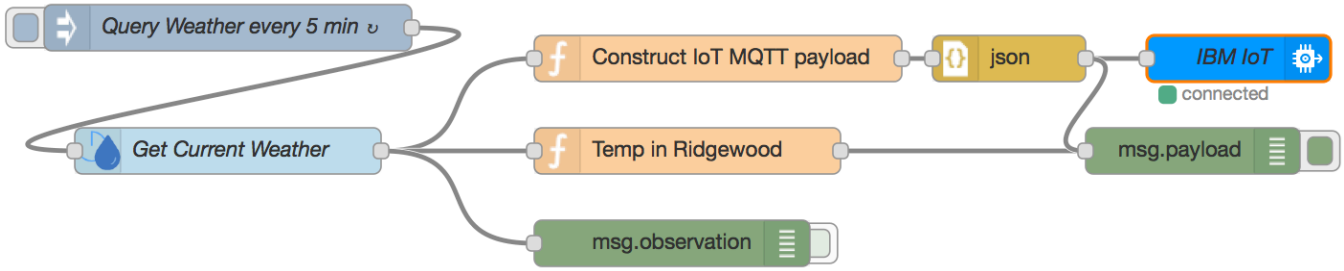
9. Add a 

node as shown below.



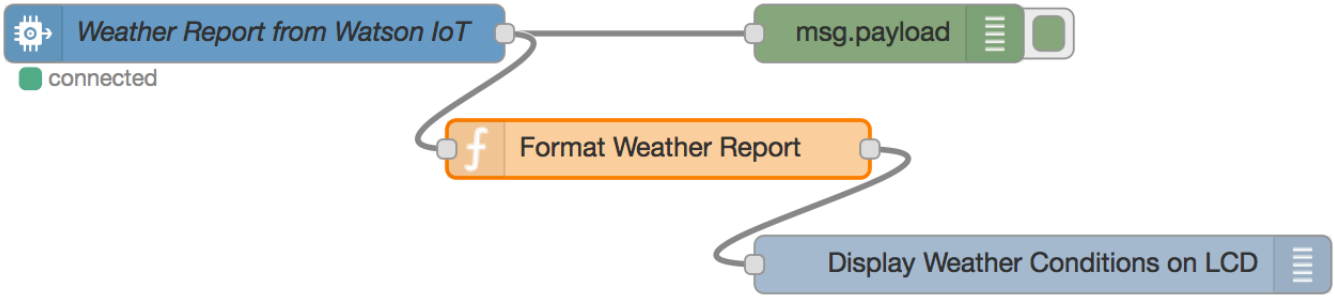
This will format the temperature in a MQTT message payload, for use in step #11. If the weather condition changes every 5 minutes, the value will be sent to the Intel NUC for display on the LCD.

10. On IBM Bluemix, Connect the nodes together as shown below.



This flow is available at https://ibm.biz/BdrgQL

11. On the Intel NUC, Connect the nodes together as shown.



This flow is available at https://ibm.biz/BdrgQg

12. Click on 

to save and deploy your changes. The LCD should display the outside temperature and will update the temperature from the Weather service every 5 minutes.