# B: Example Node-RED Client

**Prerequisites:**

- Installing the Java Runtime Environment
- Installing Ignition
- Installing the following MQTT Modules
  - MQTT Distributor
    - v3.1.0 or greater if using Ignition 7.9.X
  - MQTT Engine
    - v3.1.0 or greater if using Ignition 7.9.X
- Downloading the Sparkplug Sample Code onto a development system

**Overview:**

Sparkplug is an open source project developed by Cirrus Link Solutions which shows how devices or projects can be enabled to communicate with MQTT Engine and Ignition. This example will show how data can be published via MQTT from an emulated device running on a development machine. In addition, it will show how devices or projects can be controlled by writing to tags in Ignition. It will also show the caveats associated with establishing /ending an MQTT session and ensuring that the tag values in Ignition are valid.

**Example JavaScript Client:**

This tutorial assumes:

- Ignition is running and in active trial mode or using a purchased license.
- MQTT Distributor is installed and running, using the default configuration, and in active trial mode or using a purchased license.
- MQTT Engine is installed and running, using the default configuration, and in active trial mode or using a purchased license.
- The Node.js JavaScript runtime is installed.
- The Node-RED tool is installed.

With the standalone Sparkplug example downloaded onto your development machine, change into the directory:

```
sparkplug_b/stand_alone_examples/nodered
```

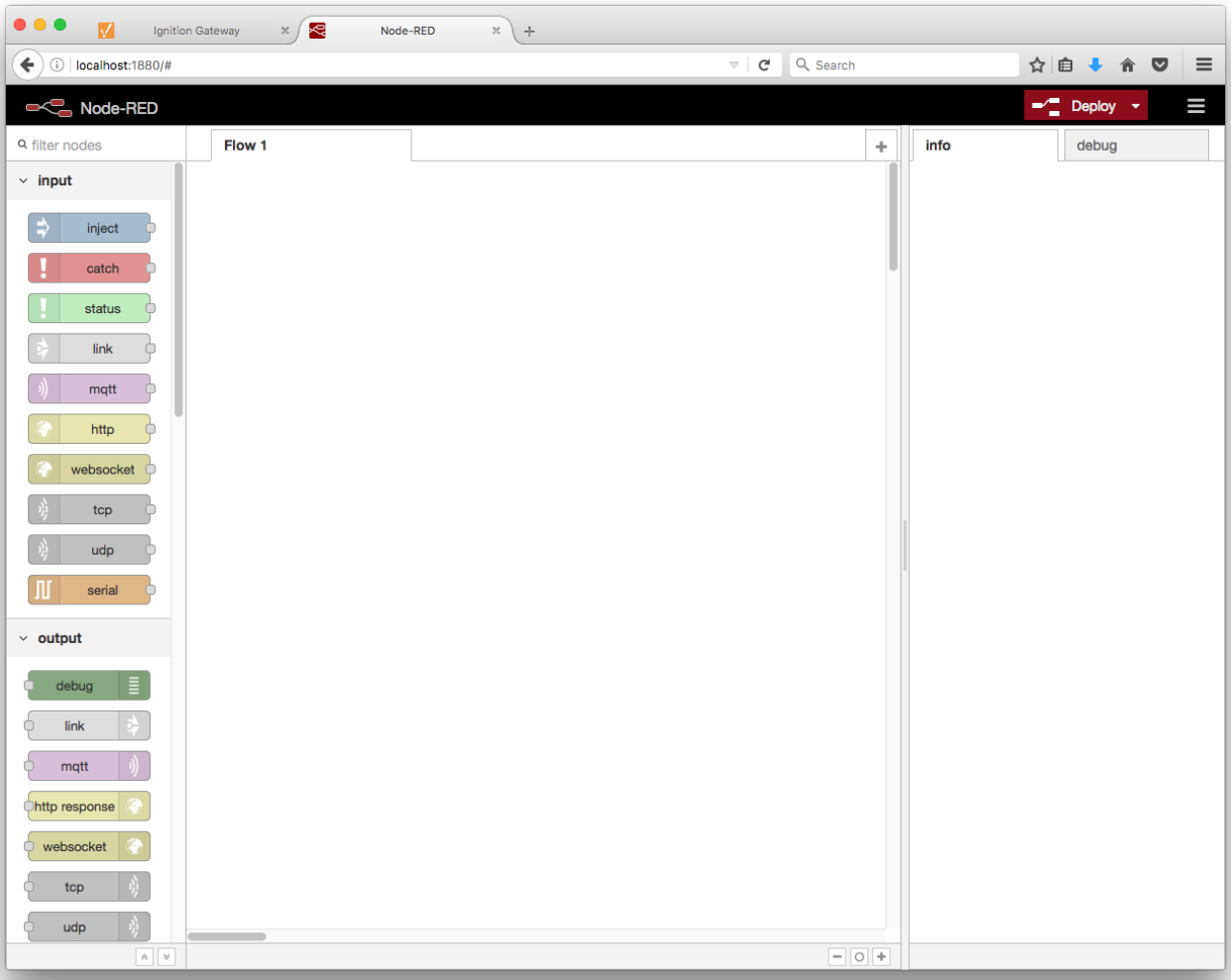
Before running the application we need to install the node-red-contrib-sparkplug library using the Node Package Manager (npm). Issue the following command:

```
npm install -g node-red-contrib-sparkplug
```

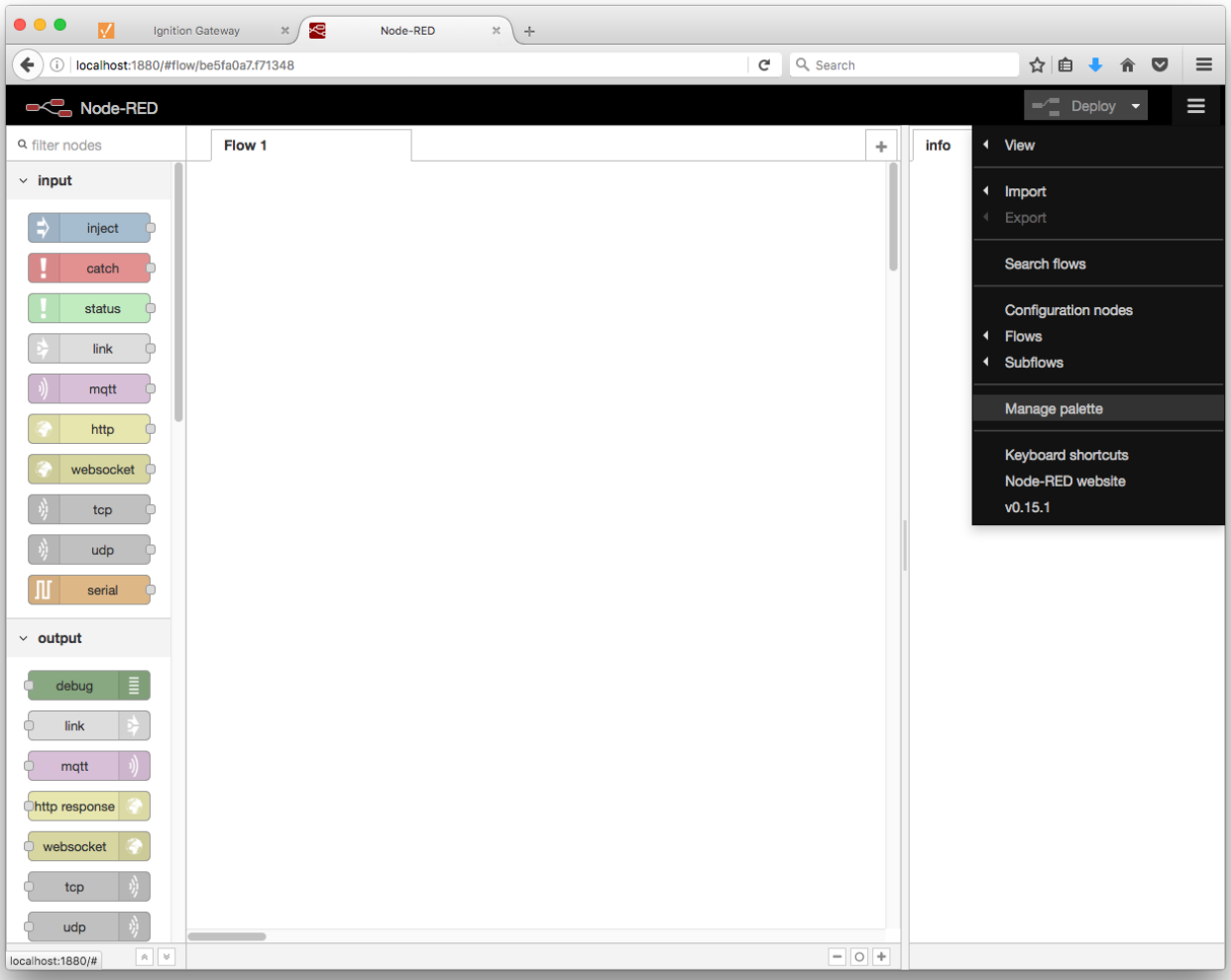
Now start the Node-RED application with the following command:

```
node-red -v
```

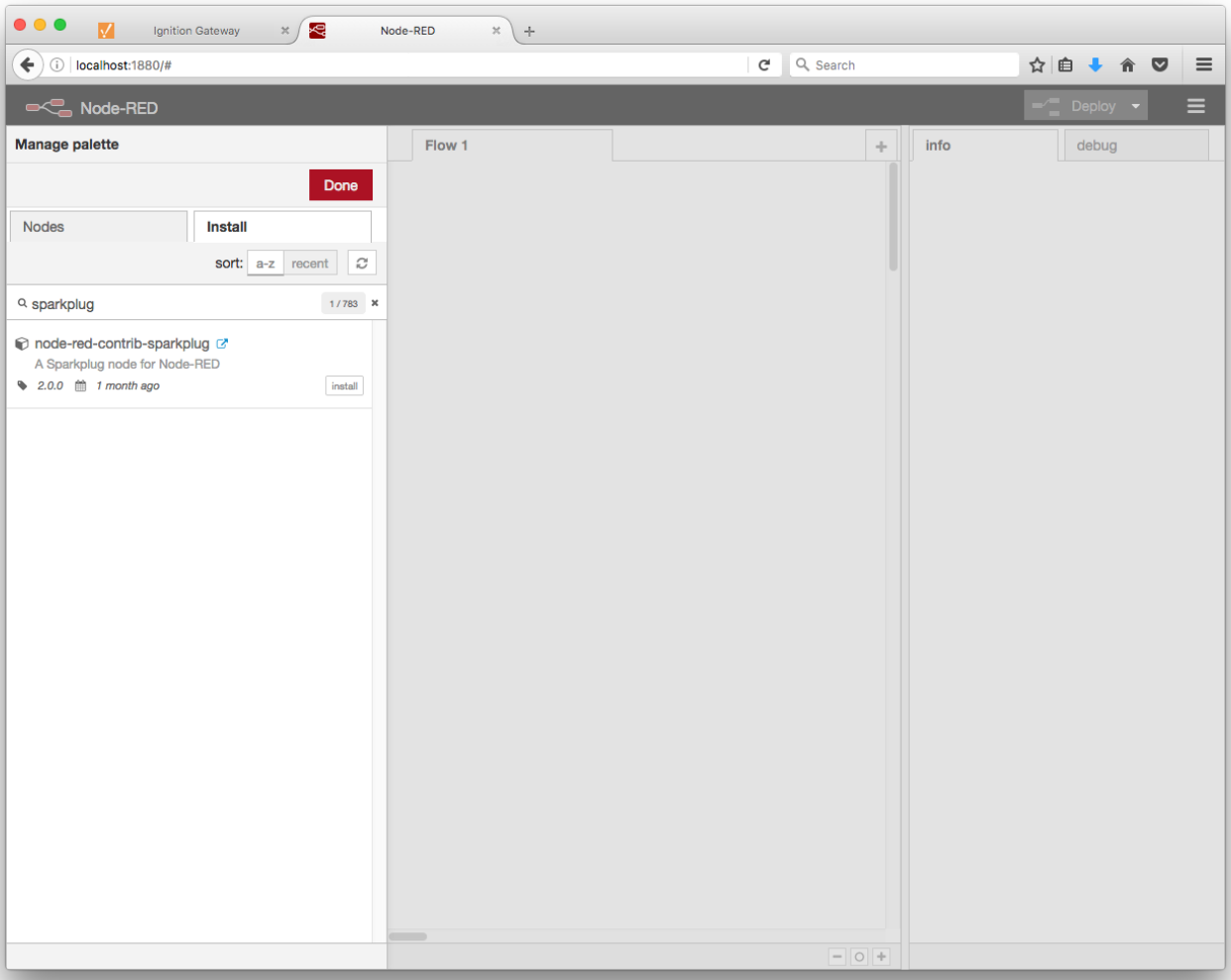
Now you can open a browser to http://localhost:1880/ to view the Node-RED visual tool.




We need to install the node-red-contrib-sparkplug library module into Node-RED. We can do this using the "Manage palette" option in the upper right dropdown menu.



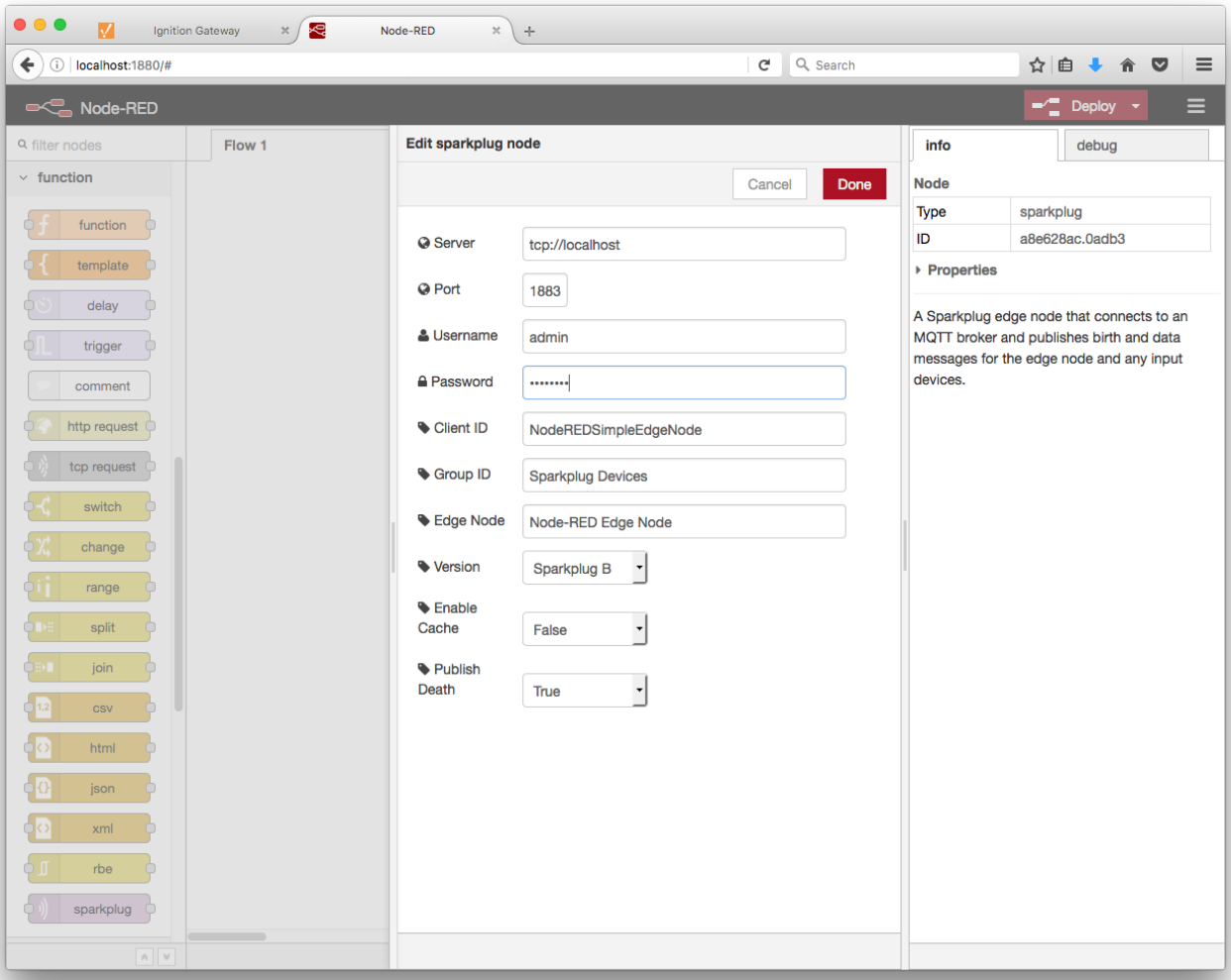

Once "Manage palette" has been selected a new menu will appear on the left.  Click the "Install" tab and search for "sparkplug".




You will see the node-red-contrib-sparkplug node appear in the search results. Click the install button for this node to install it into Node-RED. Click the Done button once it has successfully installed.

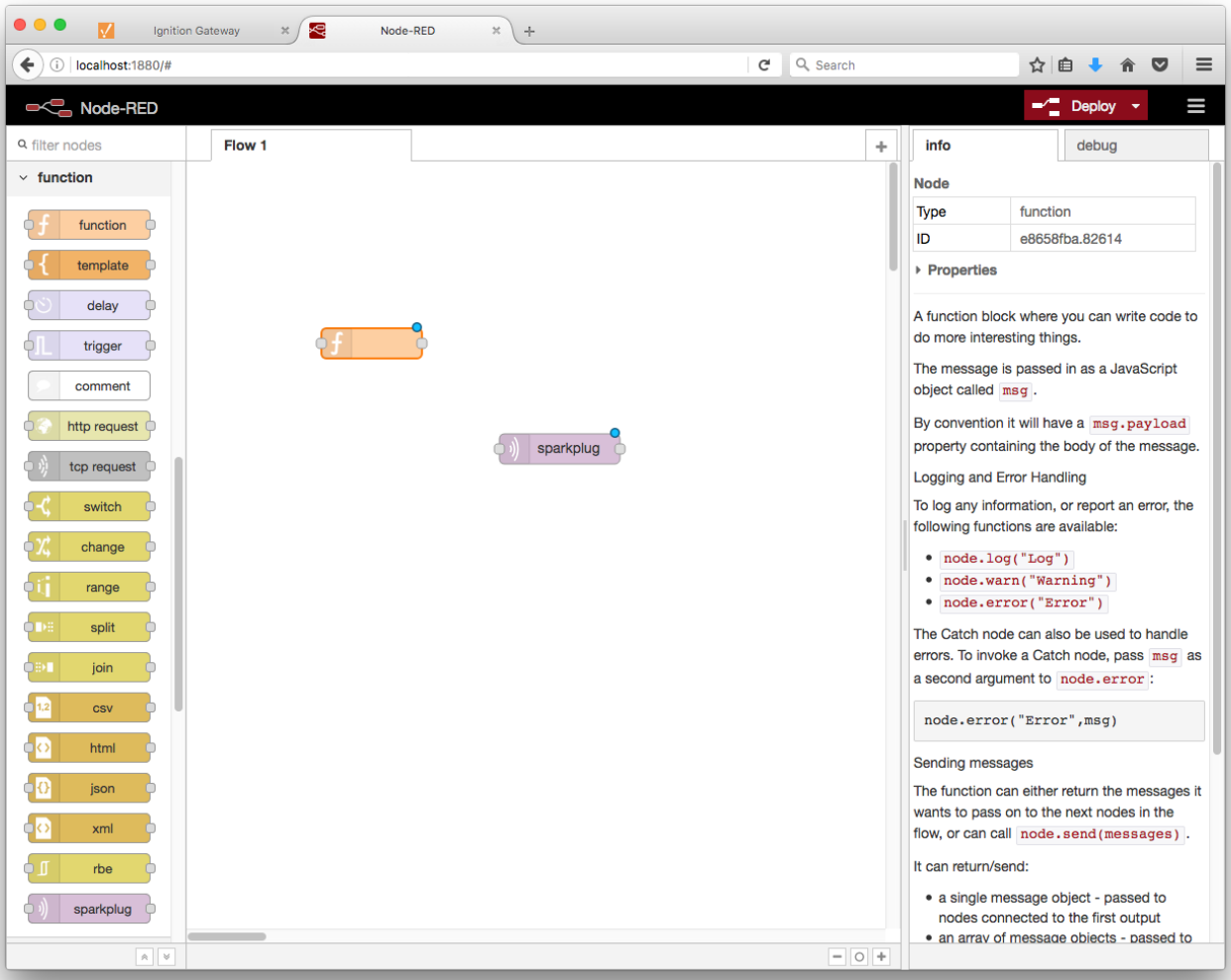
In the lower left, under the function section, there is a now a Sparkplug node. Click and drag the Sparkplug node into the flow diagram. This node will represent a Sparkplug Edge Node. It will establish and maintain a connect with the MQTT Server, publish NBIRTH messages, handle any received NCMD messages, publish DDATA and DBIRTH messages from connected device nodes, as well as send and received DCMD messages to the device nodes.

Double click the node to bring up the screen to edit the sparkplug node's properties.




Enter the MQTT Server URL and port number to connect to along with the username and password (the default for MQTT Distributor is admin /changeme). The remaining properties can be left as the defaults. Click "Ok".

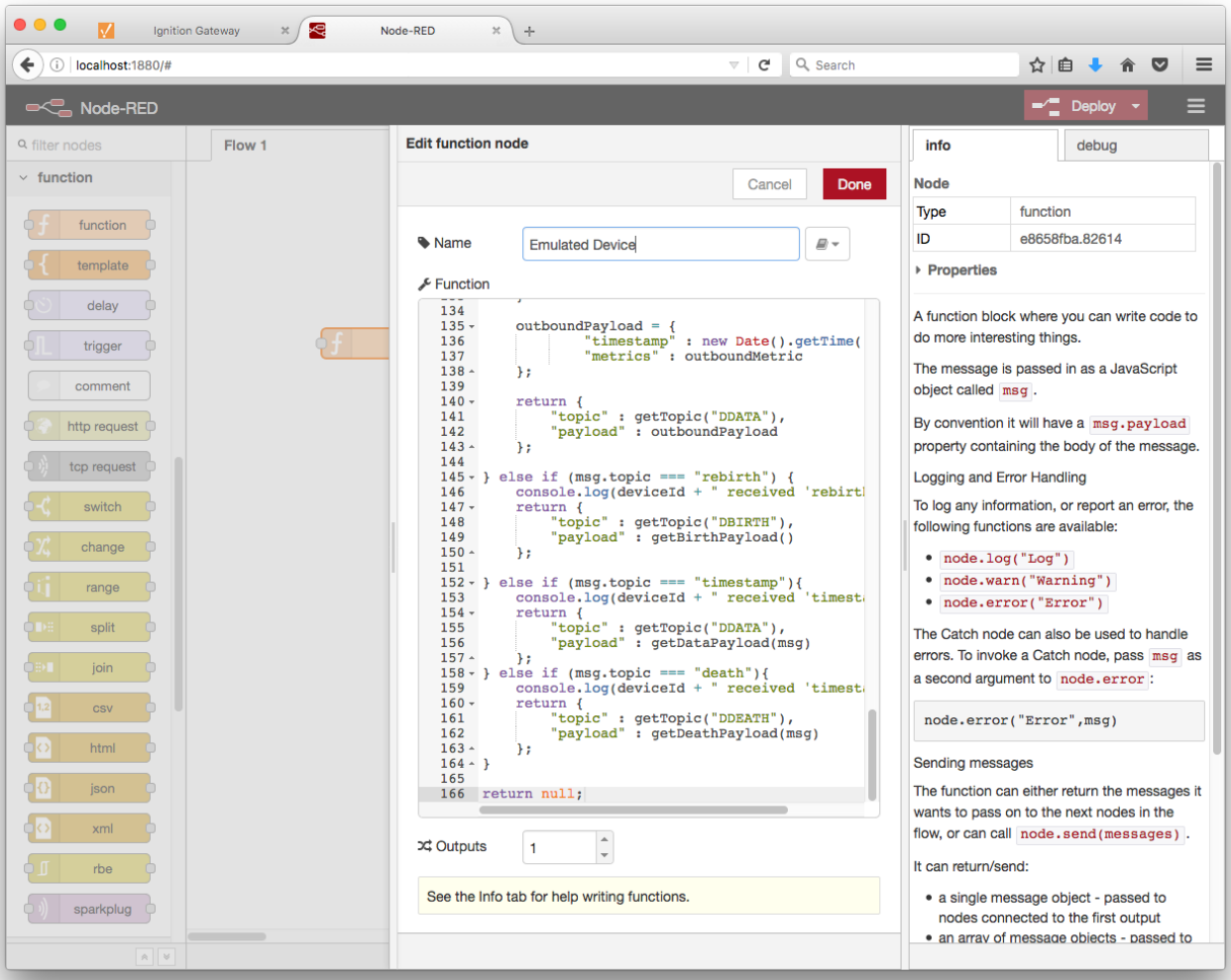
Now click and drag a function node onto the flow diagram to the left of the sparkplug node.



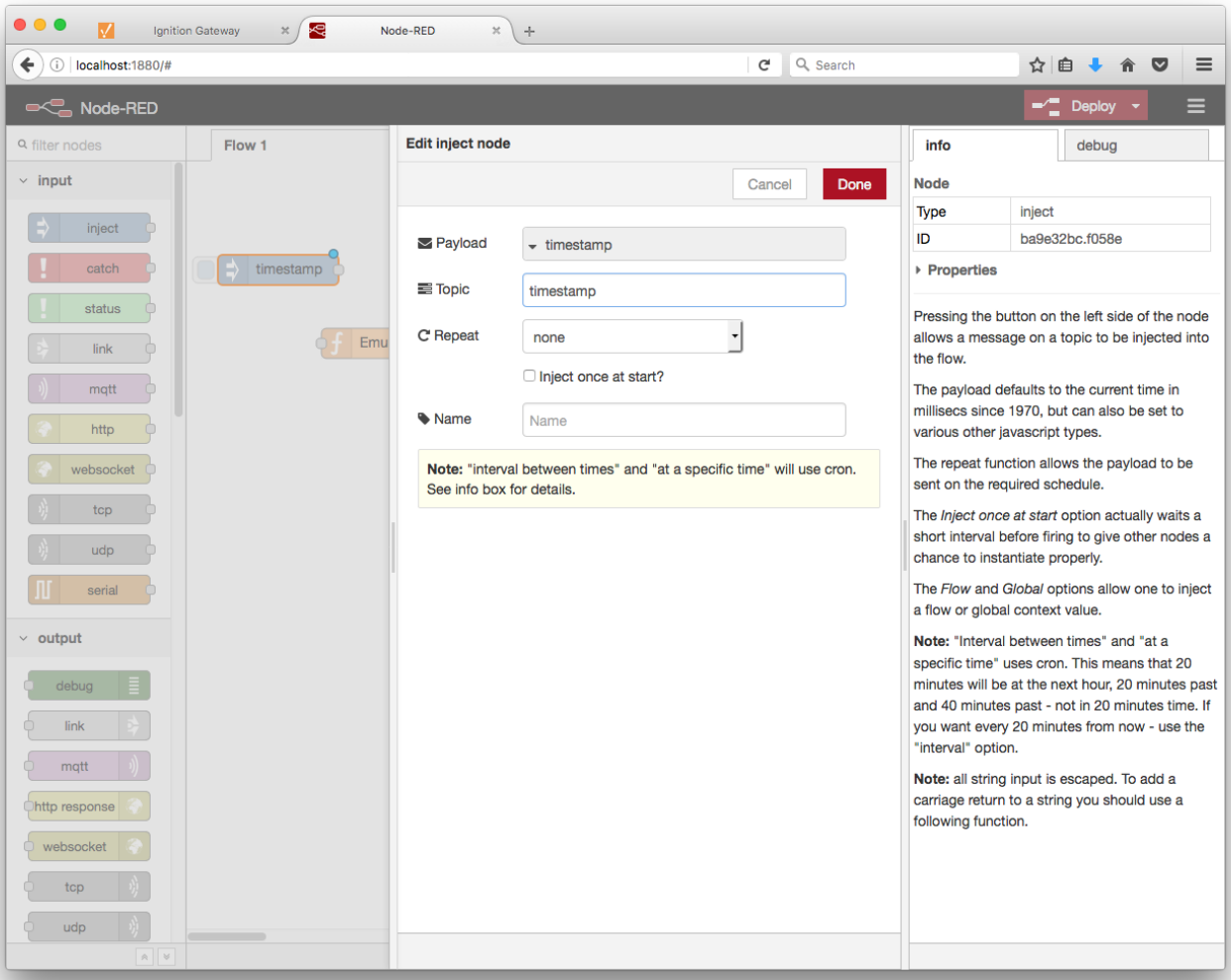

Use a text editor of your choice to copy the contents of the following JavaScript file to the clipboard:

sparkplug_b/stand_alone_examples/nodered/emulated-device.js

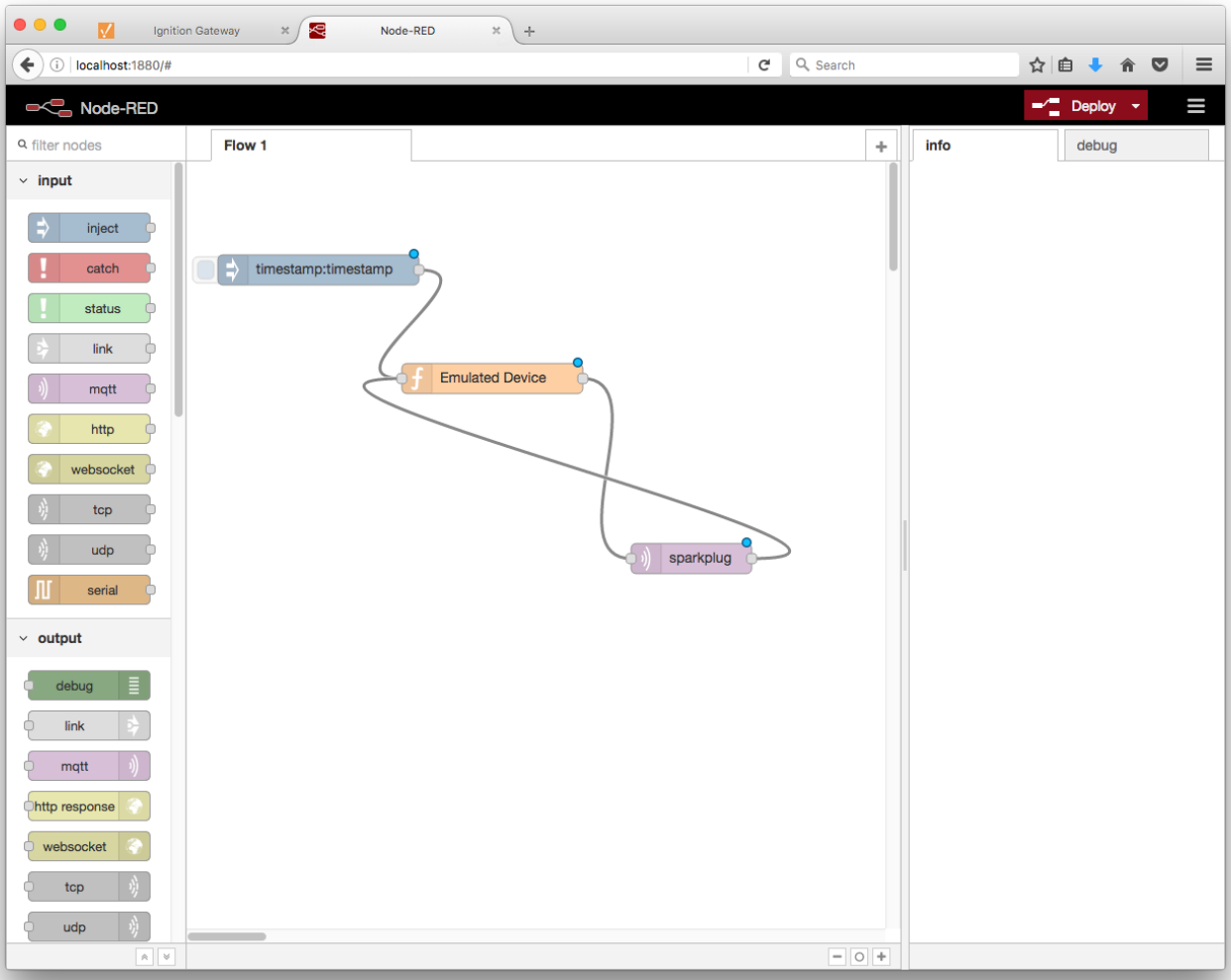
then double click the function node in the flow diagram in Node-RED to bring up the the editor for the function node properties. This node will be emulating a device by generating random data points, and sending "DBIRTH", "DDEATH", and "DDATA" messages to the Sparkplug Edge Node as well as respond to "command" and "rebirth" messages sent from the Sparkplug Edge Node . Give the function node a name and paste javascript (that was copied to the clipboard) into the "Function" editor, overwriting what was there by default. Click "Ok".




We now need a way of triggering a DDATA publish even from the device. Click and drag an input node to the flow diagram, to the left of the Emulated Device node. Double click the inject node to edit it's properties. Set the topic property to "timestamp" and leave the rest as defaults. This will allow us the ability to manually trigger a publish of device data.

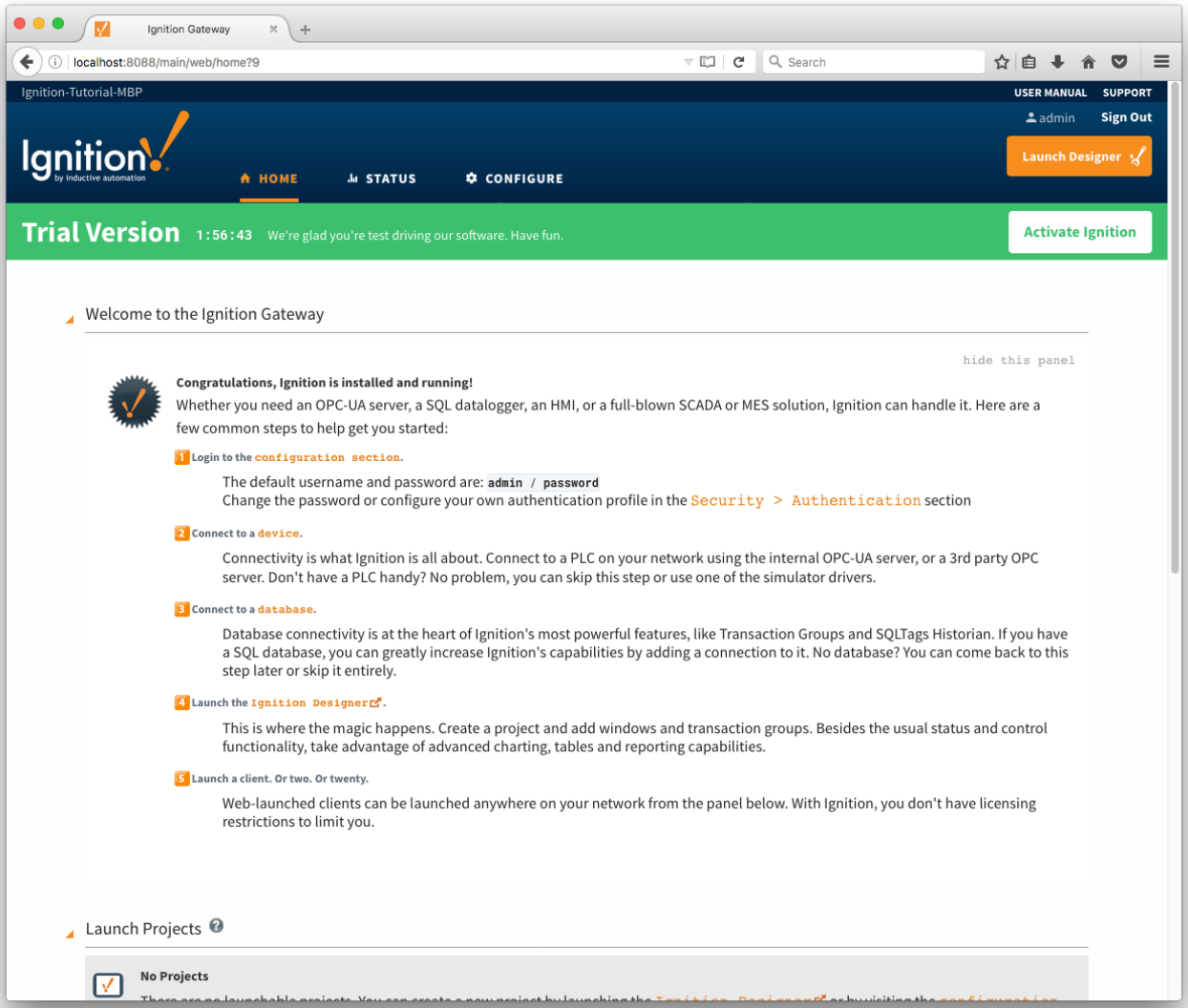



Now we can wire the nodes together. Connect the output of the timestamp node to the input of the Emulated Device node. Connect the output of the Emulated Device node to the input of the Sparkplug Edge Node. Finally Connect the output of the Sparkplug Edge Node to the input of the Emultated Device node. This creates a way for the device to both publish messages and receive messages from the Sparkplug Edge Node.



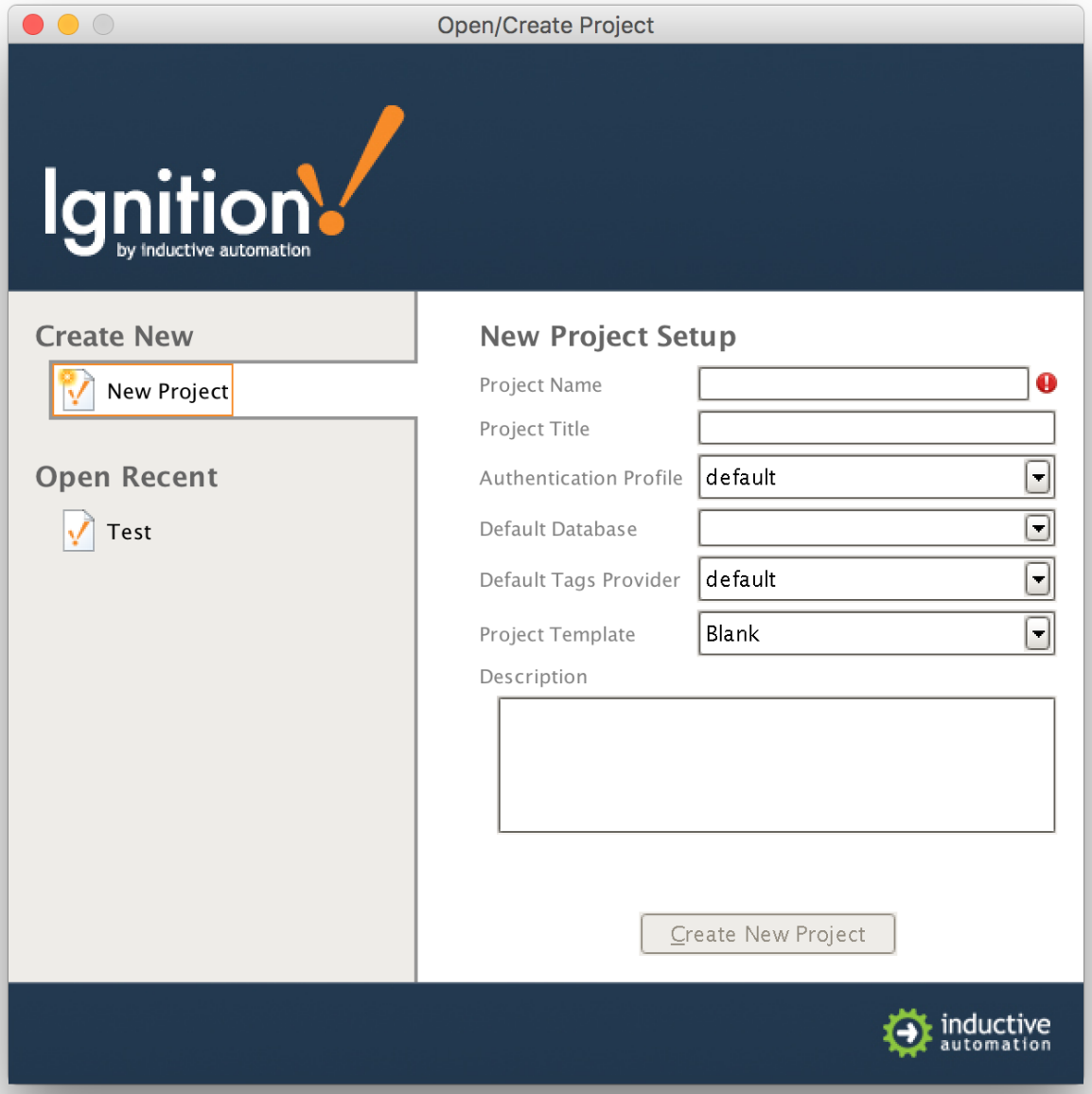

Now it's time to click Deploy in the top right corner of the Node-RED tool. You can monitor the command line window where you started Node-RED and see log messages indicating that the Node-RED Edge Node's client has connected, subscribed to control and command topics, published a NBIRTH message, and emitted a "rebirth" event to the Emulated Device. The Emulated Device then published a DBIRTH message with a payload containing all data points/values that the device will report.

This can be verified via Ignition Designer. Using a Web Browser, browse to the Ignition Gateway on your Ignition Gateway. If it is running on your development machine, that is: http://localhost:8088. You should see this:

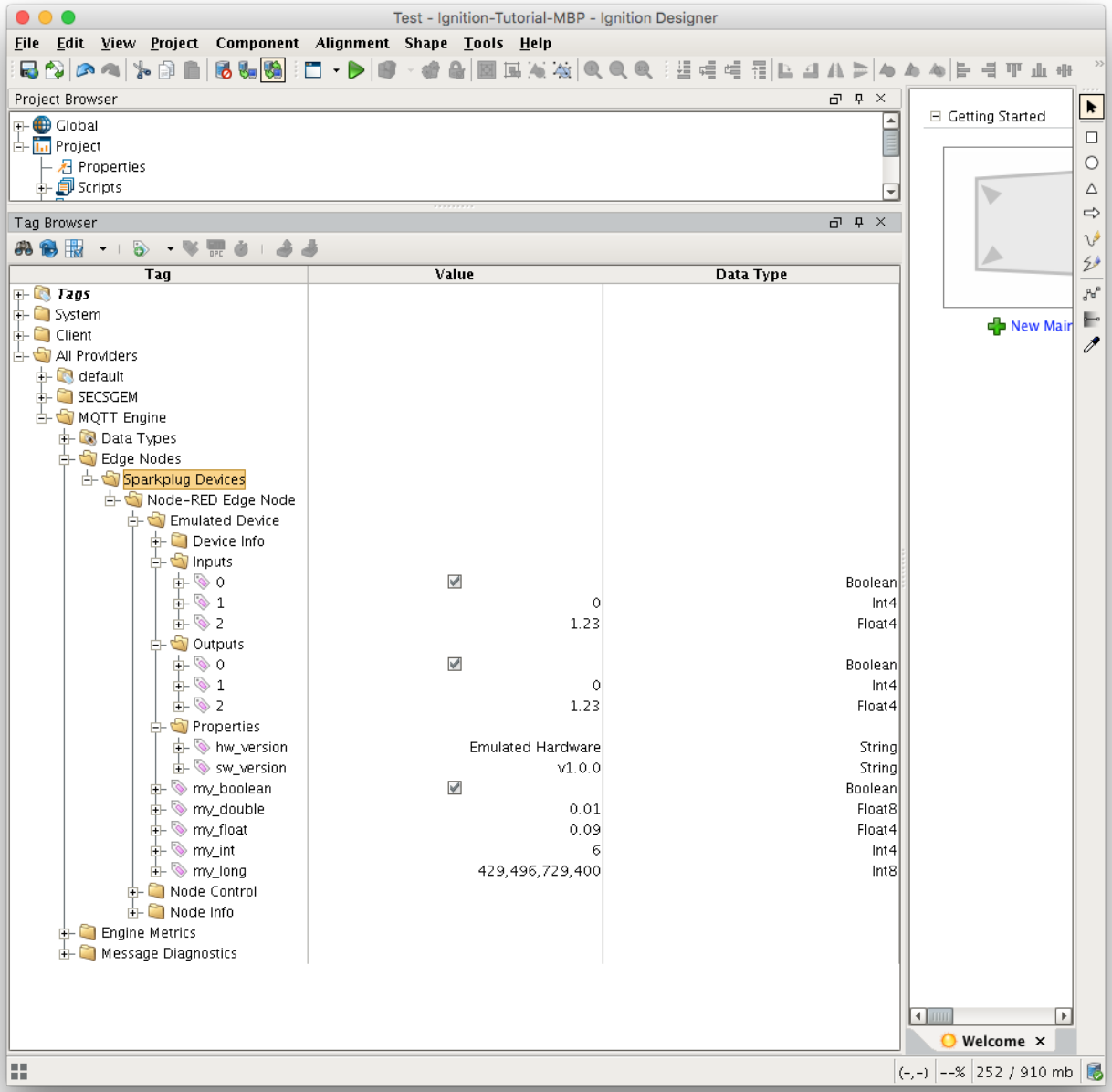



Near the upper right corner, click 'Launch Designer'. This will open the following window after downloading the .jnlp file and executing it. Note the default username/password is admin/password. Type those into the appropriate fields and click 'Login'.


This will bring you to a new Window where you can select an Ignition Project or create a new one.  Create a new project by giving it a name and clicking 'Create New Project'.

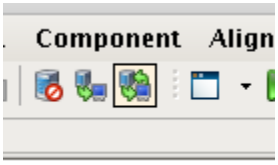



Now you should be in designer. In the left hand side of the main window is a 'Tag Browser' window. In it, expand 'All Providers -> MQTT Engine -> Edge Nodes -> Sparkplug Devices -> Node-RED Edge Node -> Emulated Device'. You should see the following




You will see that MQTT Engine saw a new device attach to the MQTT Server and publish a Birth Certificate. As a result, MQTT Engine created the Ignition Tags shown above. These are also dynamically updated as the values change. You can also write to the outputs after you Enable Device Writes from Ignition. This can be done by putting designer into read/write mode.

Do so by clicking this button in the menu to enable read/write mode:




Then you can change any of the Tag values in the Outputs folder here:


You should see the output value change as well as one of the inputs.  In the Emulated Device sample code the outputs are virtually tied to the inputs.  So, when modifying an output value, this causes an MQTT message to be sent from MQTT Engine, to the virtual device, which catches the message and virtually writes the values, then publishes a MQTT message back to MQTT Engine where the two values are updated.

Note: If you are not seeing the output and input values update to reflect the change you have made, make sure MQTT Engine is not configured to block outbound device tag writes as described here.