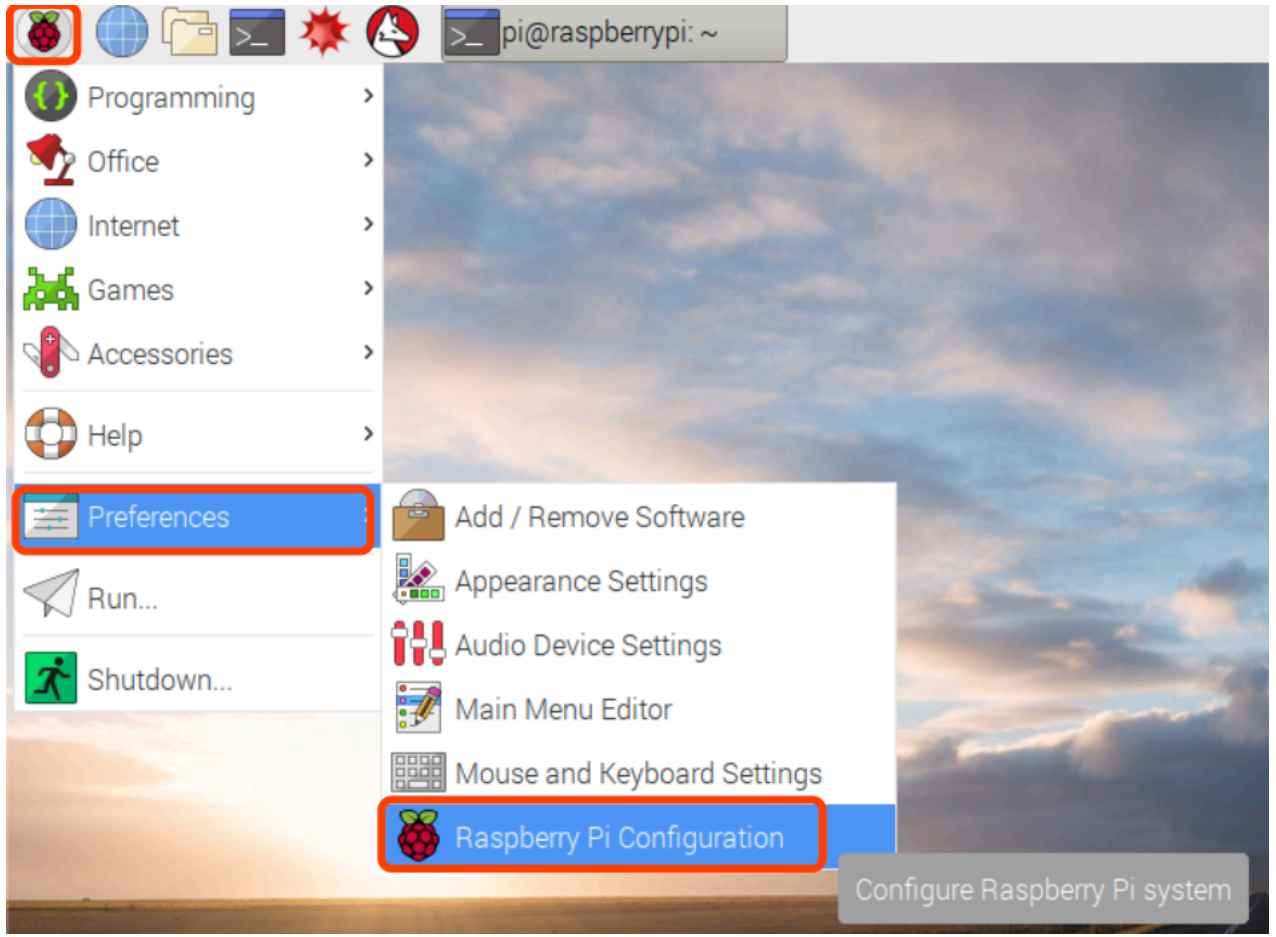
Lab 1 - Node-RED Application

Locale and Keyboard

Open System Configuration

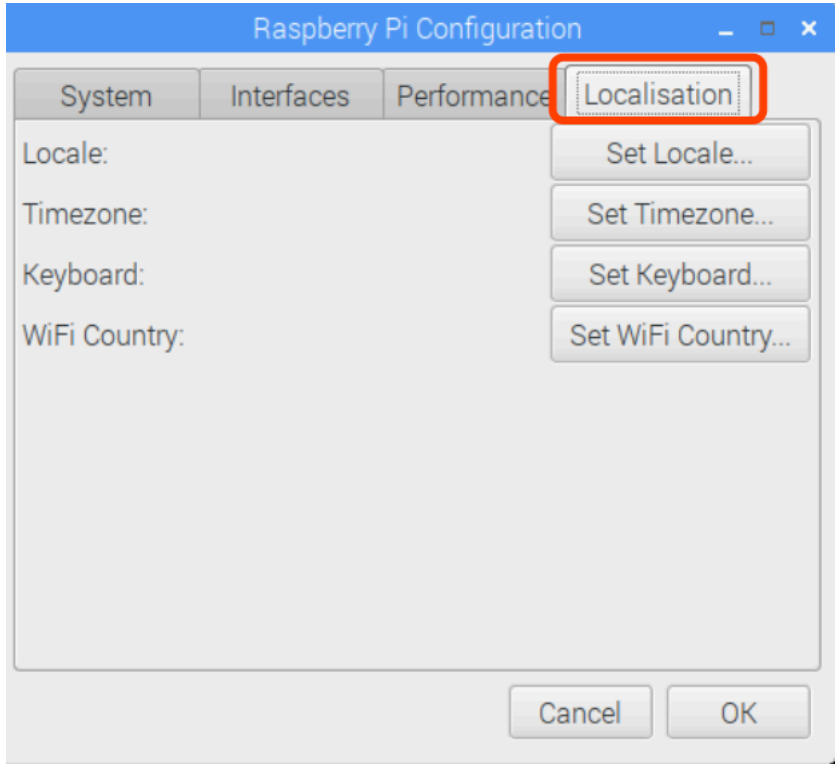
- Application Menu -> Preferences -> Raspberry Pi Configuration



Locale and Keyboard

Localisation Configuration

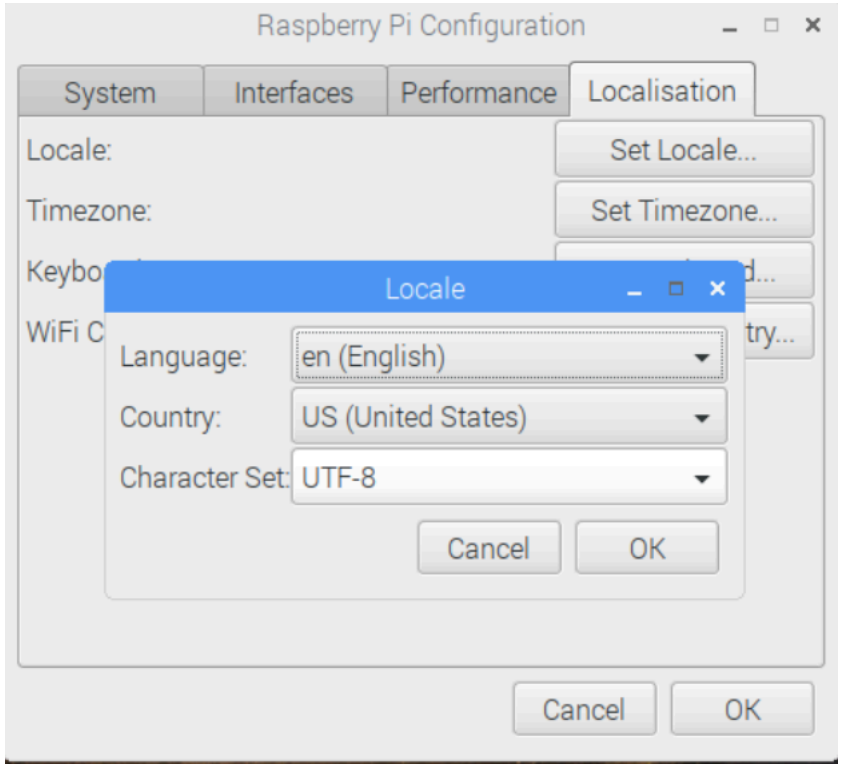
- Switch to Localisation tab
- Select appropriate option for
  - Locale
  - Timezone
  - Keyboard
  - WiFi Country (necessary if you are not able to connect to WiFi)



Locale and Keyboard

Localisation Configuration

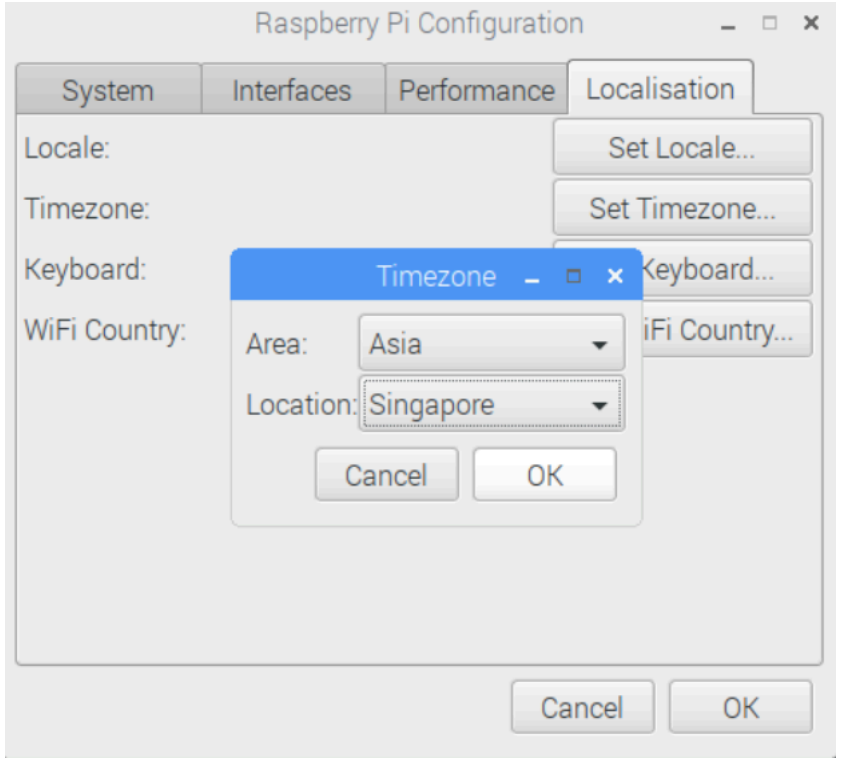
- Switch to Localisation tab
- Select appropriate option for
  - **Locale**
  - Timezone
  - Keyboard
  - WiFi Country (necessary if you are not able to connect to WiFi)



Locale and Keyboard

Localisation Configuration

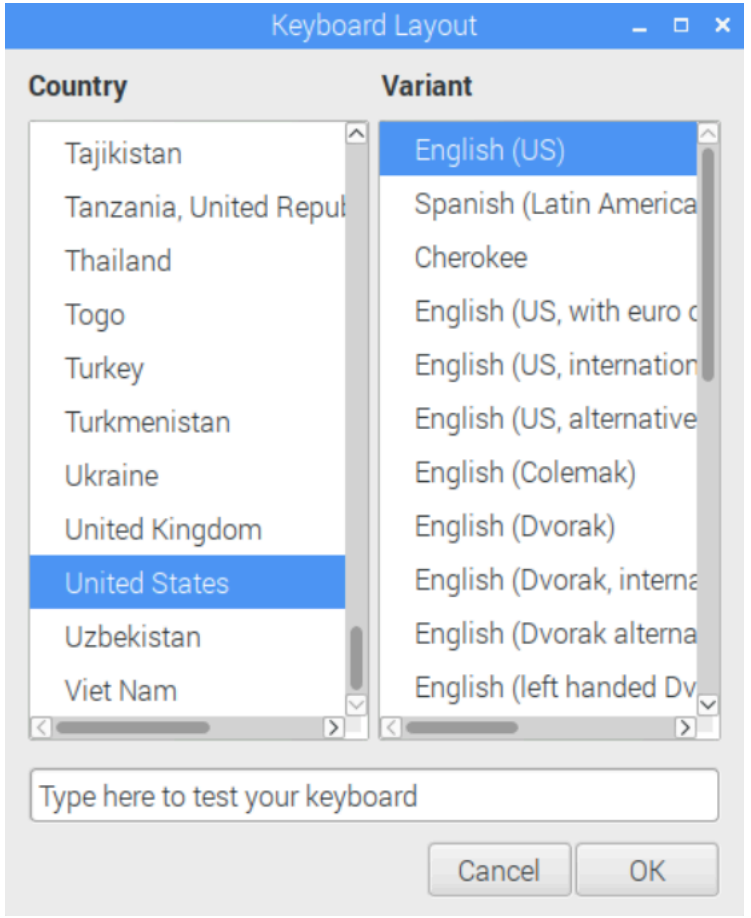
- Switch to Localisation tab
- Select appropriate option for
  - Locale
  - **Timezone**
  - Keyboard
  - WiFi Country (necessary if you are not able to connect to WiFi)



Locale and Keyboard

Localisation Configuration

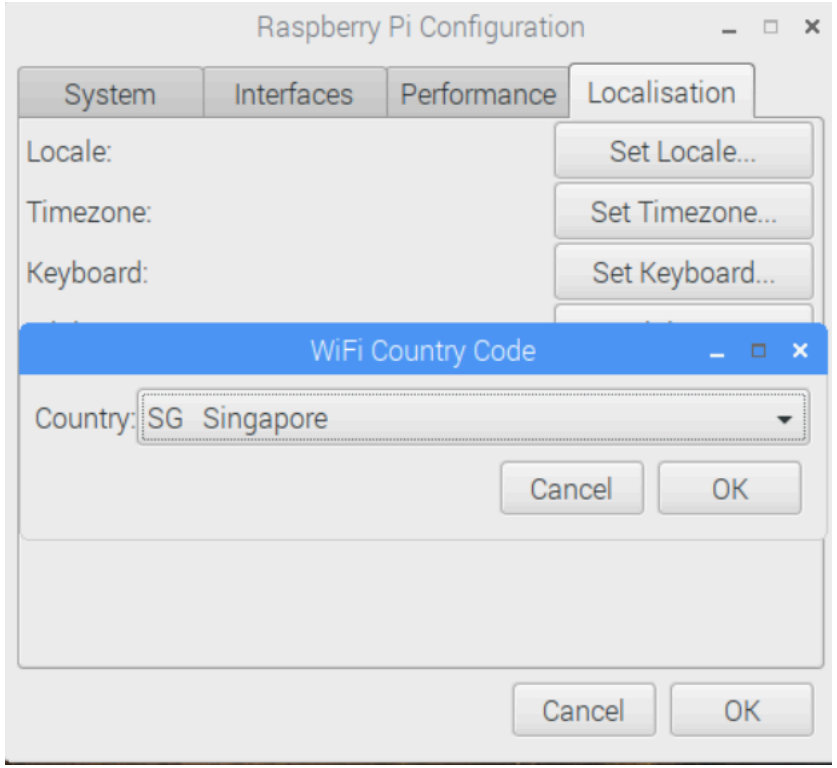
- Switch to Localisation tab
- Select appropriate option for
  - Locale
  - Timezone
  - **Keyboard**
  - WiFi Country (necessary if you are not able to connect to WiFi)



Locale and Keyboard

Localisation Configuration

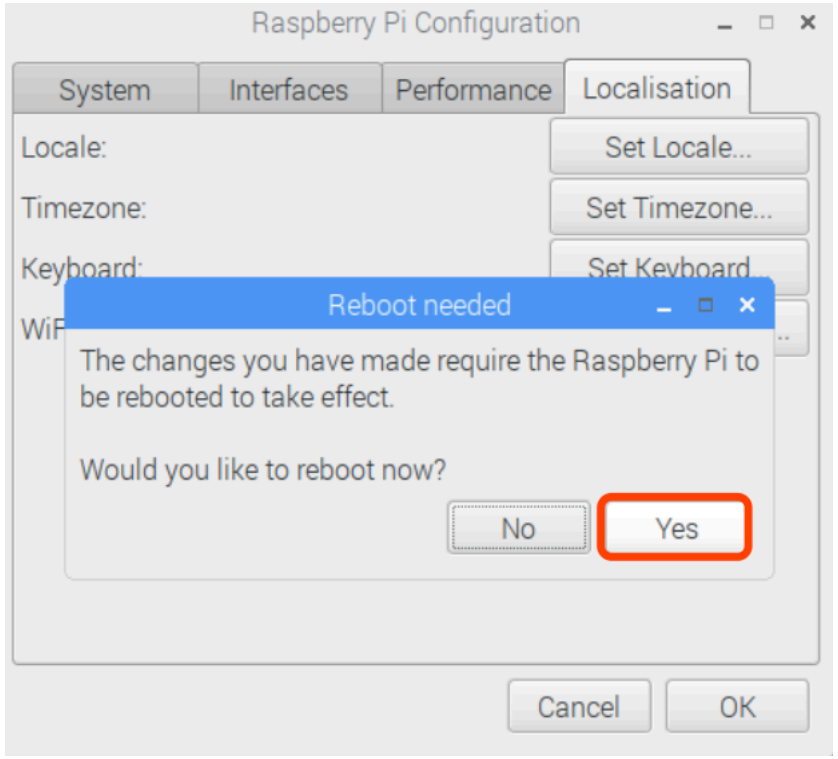
- Switch to Localisation tab
- Select appropriate option for
  - Locale
  - Timezone
  - Keyboard
  - **WiFi Country** (necessary if you are not able to connect to WiFi)



Locale and Keyboard

Localisation Configuration

- Save and reboot



Install Node.js and NPM

Install Node Version Manager or nvm

- Open command prompt on Pi

- Run the following commands:
  - `curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.8/install.sh | bash`
  - `source ~/.bashrc`
  - `nvm`

Install Nodejs version 10

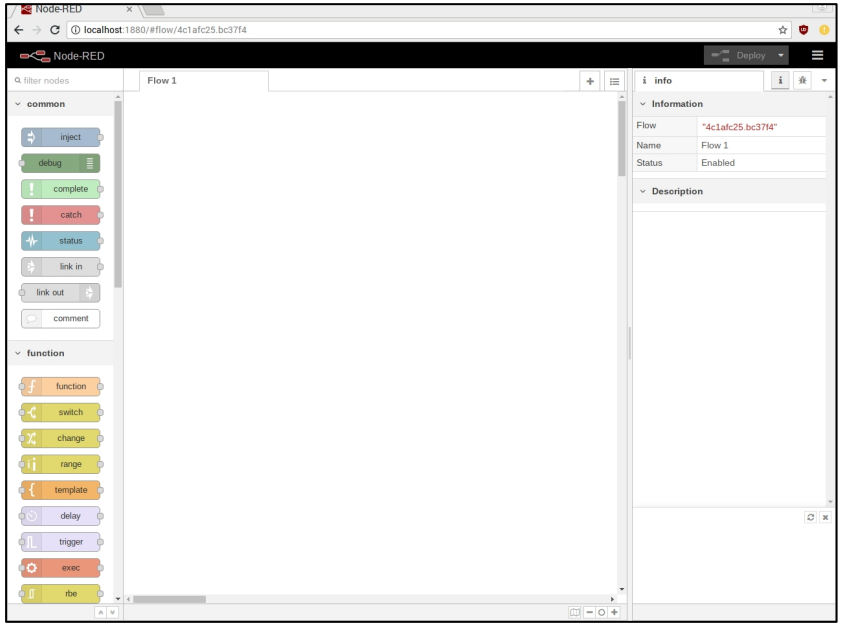
- Run the following commands:
  - `nvm install 10.18.1`
  - `node -v`
  - `npm -v`

`https://github.com/genie-training/IoT-Training`

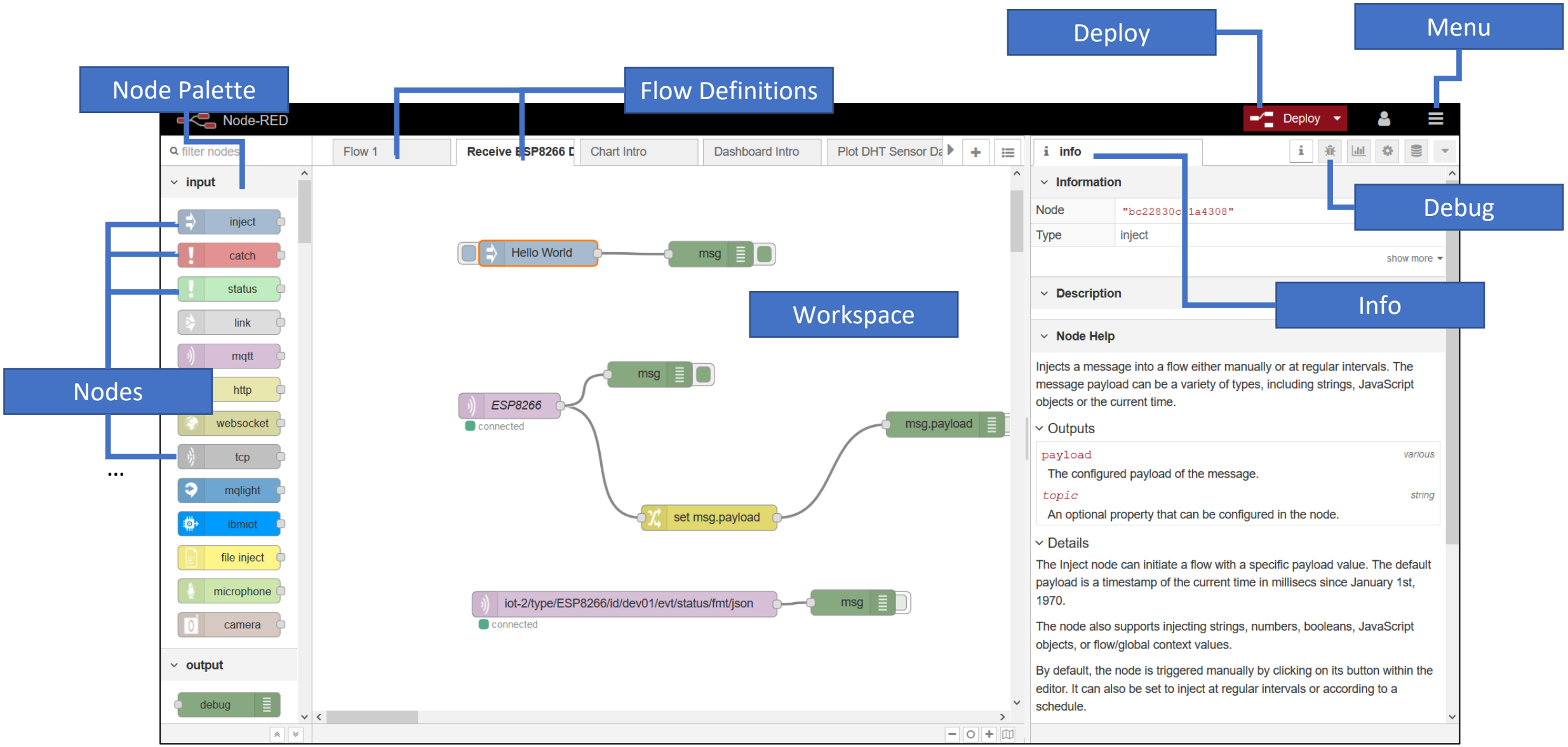
Open Node-RED on Pi

Prerequisite: Raspberry Pi has been properly setup with Node-RED, NodeJS and NPM

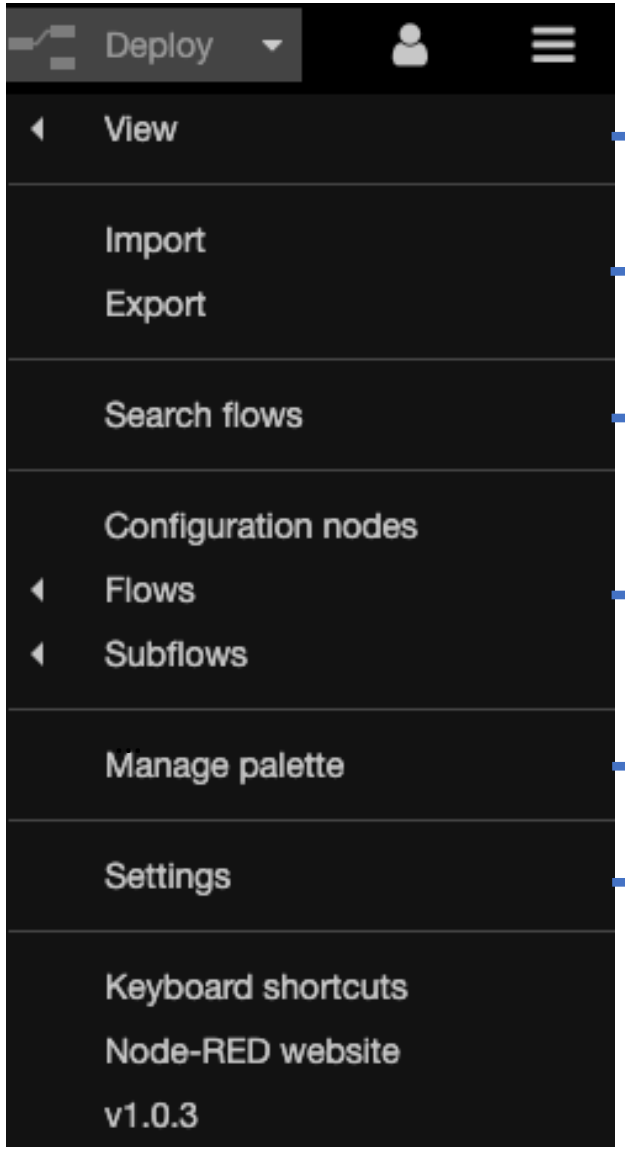
1. Open command prompt on Pi 
2. Install latest Node-RED by typing `npm install -g -unsafe-perm node-red`; check logs if successful
3. Start Node-RED by typing `node-red`
4. To stop, press Ctrl-C then type `node-red-stop`
5. Open browser and go to http://localhost:1880 to view



Node-RED UI



Node-RED UI



Display/Hide sections on the webpage, such as node palette, sidebar...

Import/Export the flows

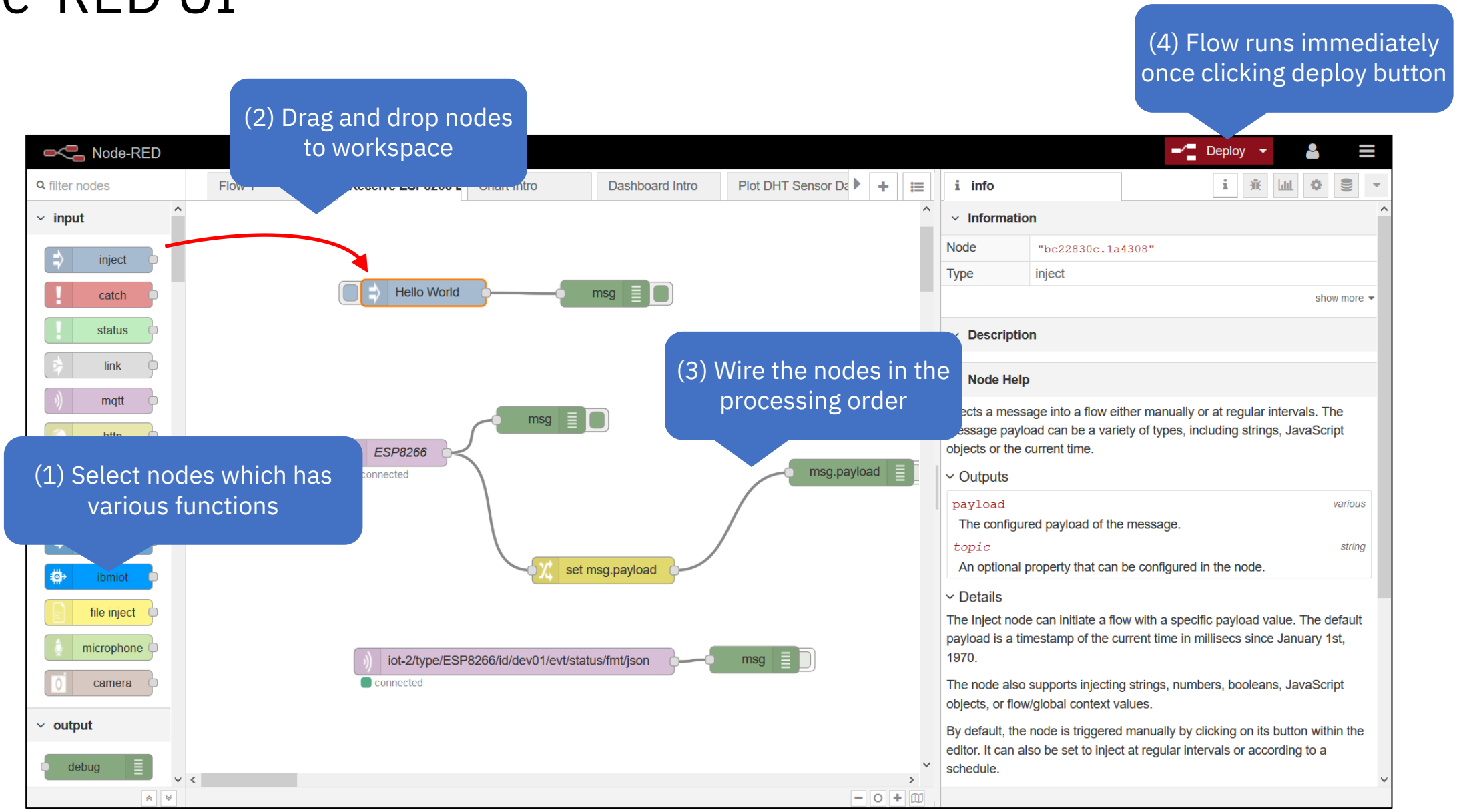
Search nodes/flows by keywords

Create/modify flows and subflows

Node module management, enable/disable nodes

General settings for view, keyboard and palette

Node-RED UI

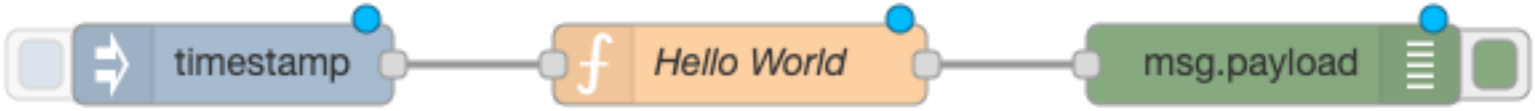


Download Flow Files

There are some flow files will be used in this lab. Before we start, please follow the steps to download:

1. Open the terminal in you Raspberry Pi.
2. Type command `cd ~/Desktop` to direct to your desktop.
3. Type command `git clone https://github.com/genie-training/IoT-Training.git` to download.
4. After downloading complete, check the IoT-Training on your desktop.

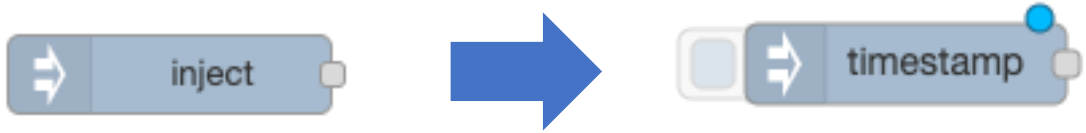
Building your first flow: Hello World



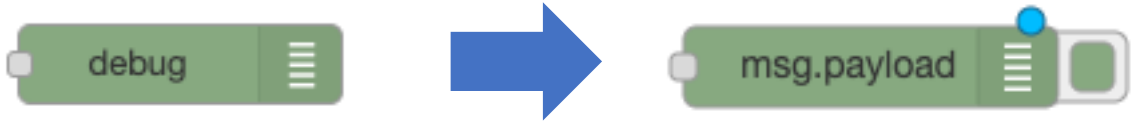
When you click on the Inject Node, it sends the event through the flow – triggering the function node and sending the result to the Debug node.

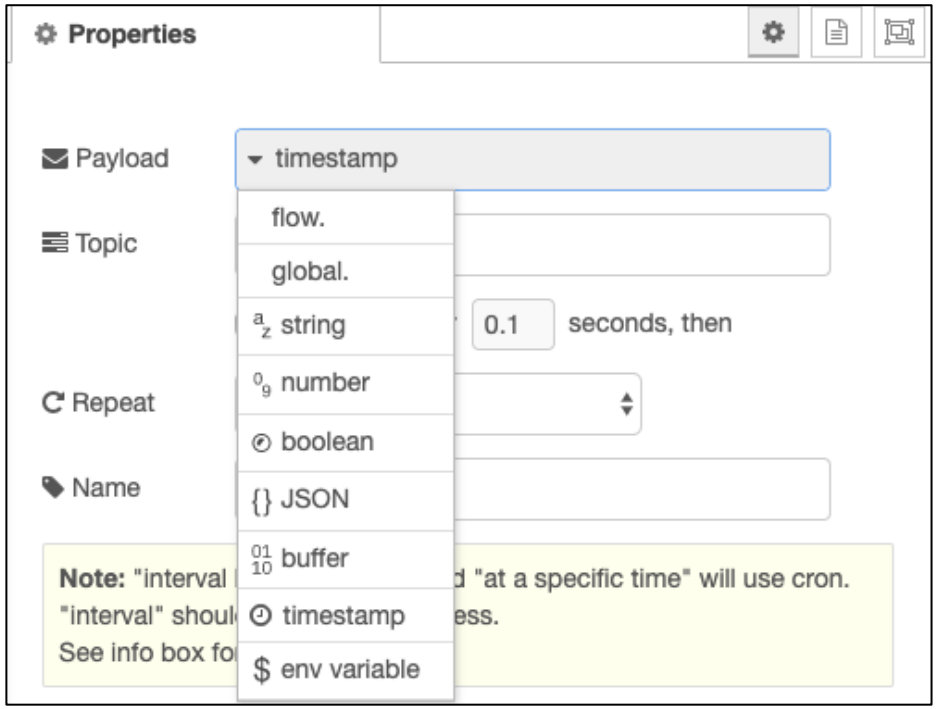
Selecting Nodes

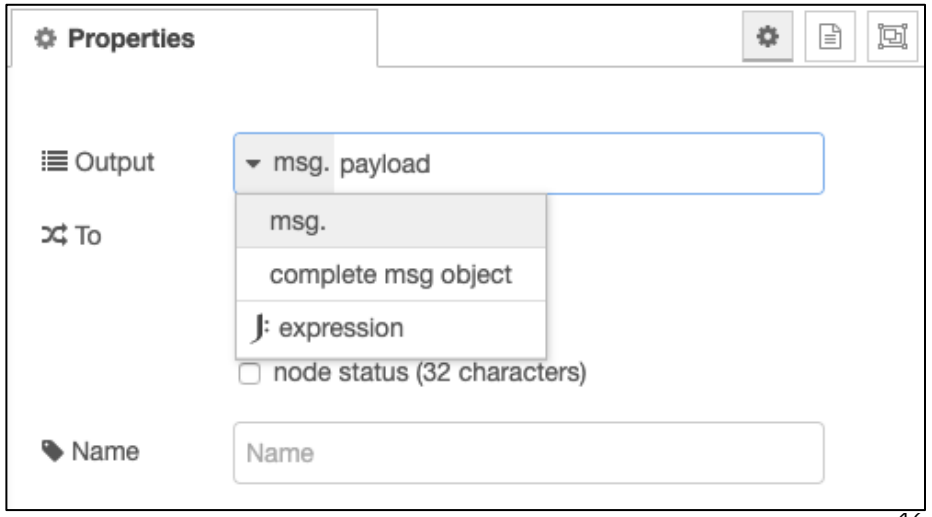
- Find the Inject Node under the input section
- Drag and drop the selected node to the workspace



- Repeat the steps to add the Debug Node

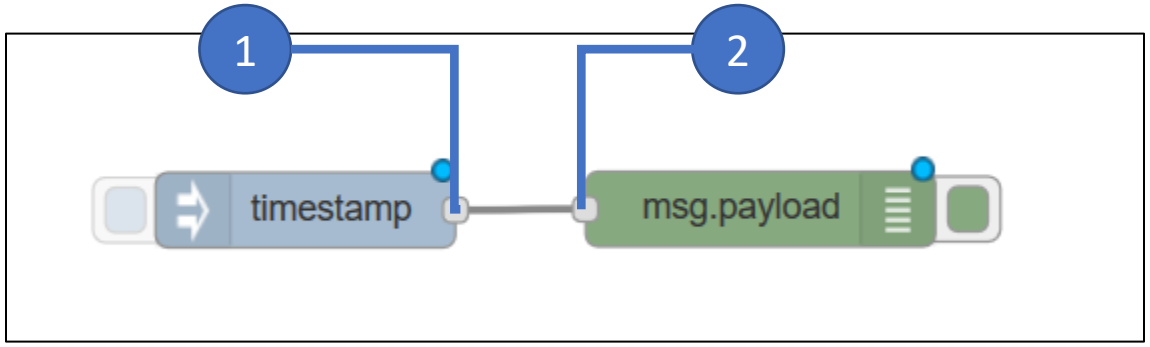






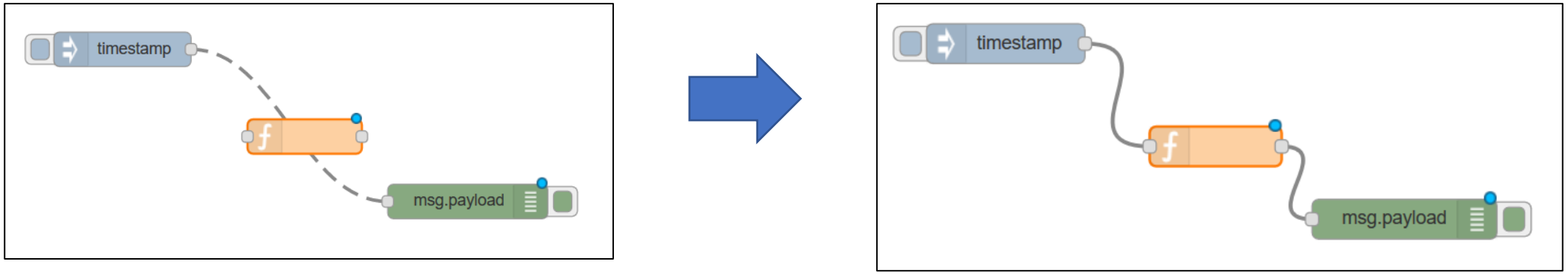
Connecting Nodes

- Click on point 1 and drag it to point 2



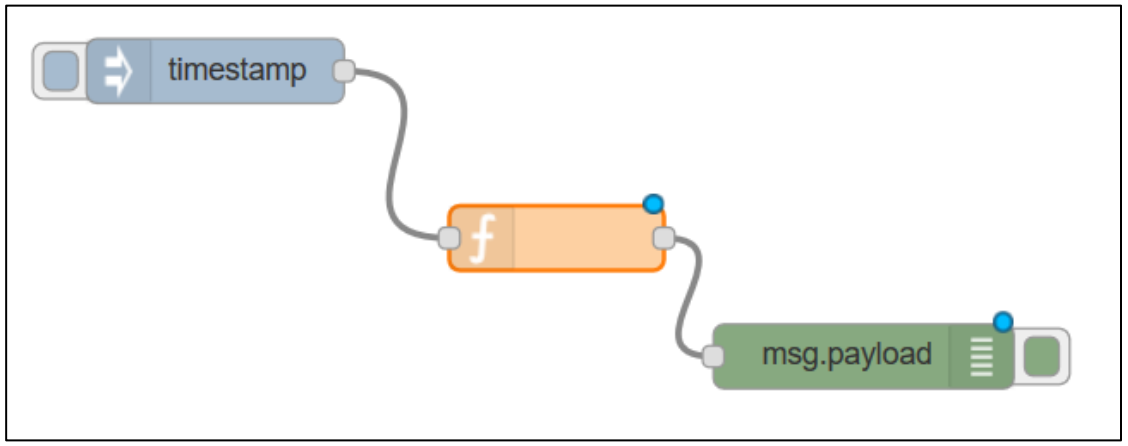
***Inject node – timestamp by default*
***Debug node – msg.payload by default*

- Drag a function node in between two nodes
- Once the line changes to a dashed line, release the node

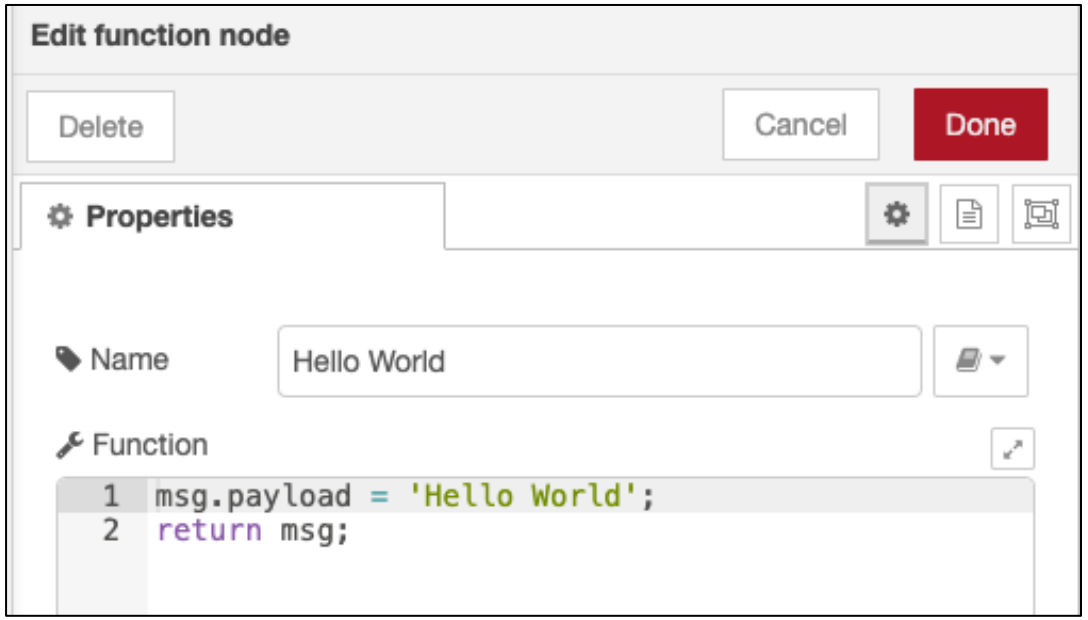


Updating output by Function

- Double click function node



- Update the name and the function code as:



Deployment and Testing

1. Click Deploy button

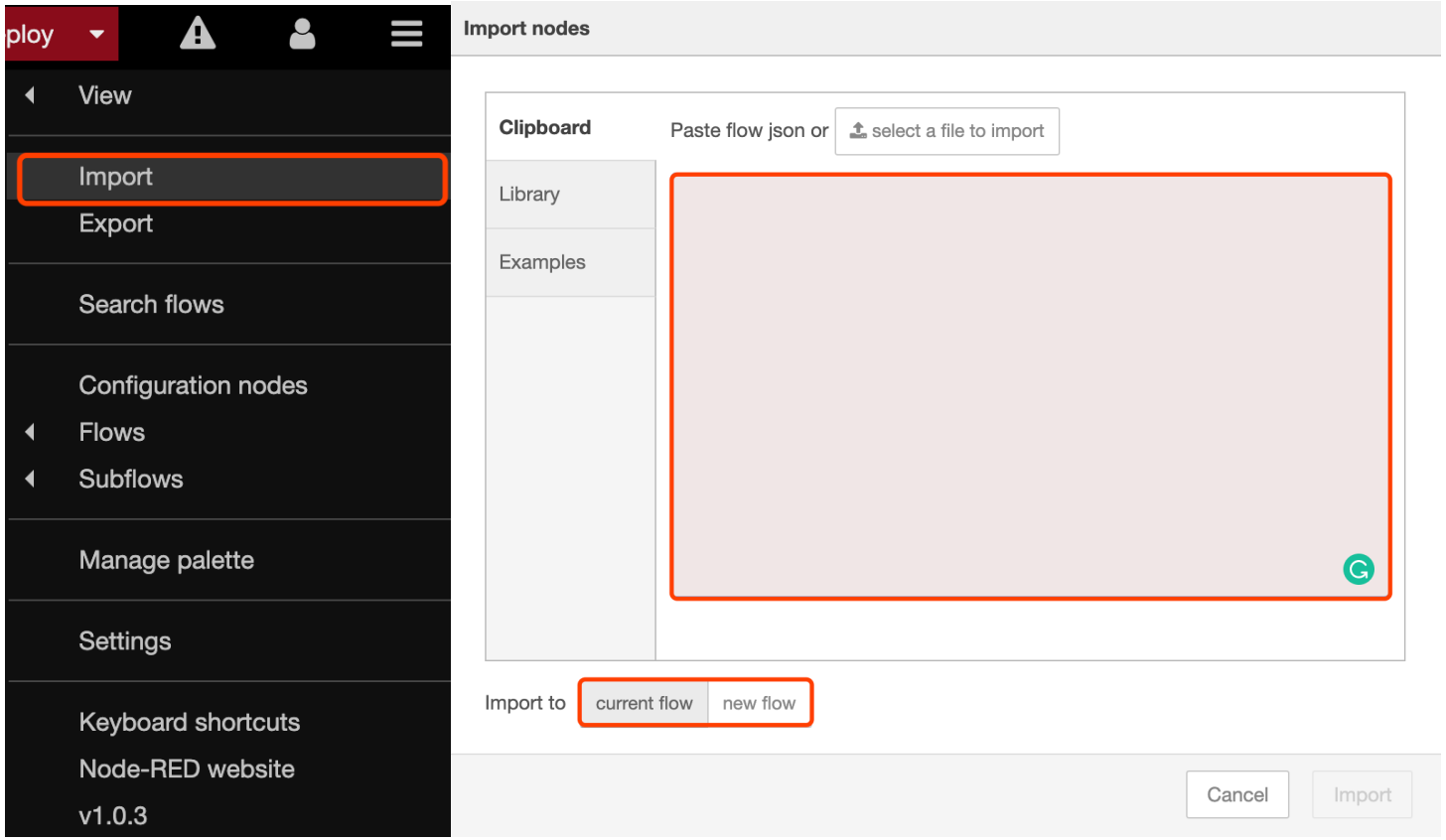
2. Trigger the inject node

3. Check the output in the debug panel

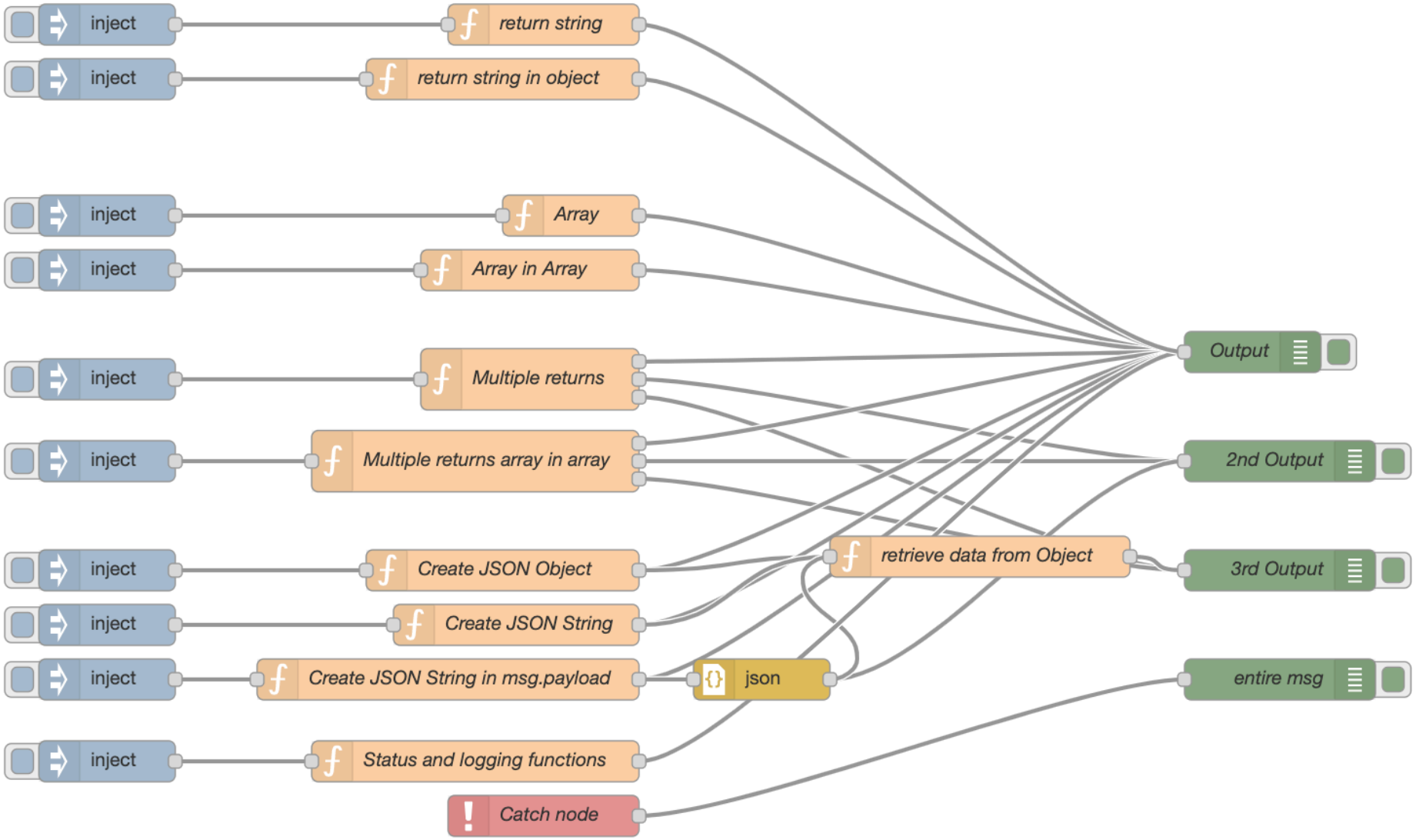
Import Flow

1. Open the Manage Palette under Menu
2. Select Import
3. Select the json file in

   Desktop/IoT-Training/Lab 1 – Node-REED Application/ Exercise 2 – Function Node.json
4. Select import to new flow
5. Click Import button
6. After the flow is loaded, click Deploy button to apply the changes

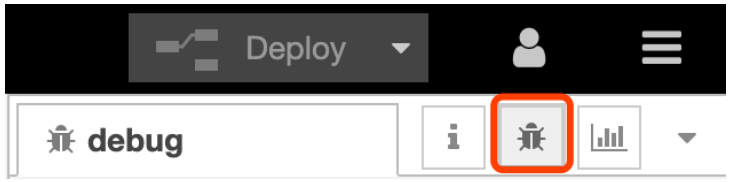


Function Node & Output

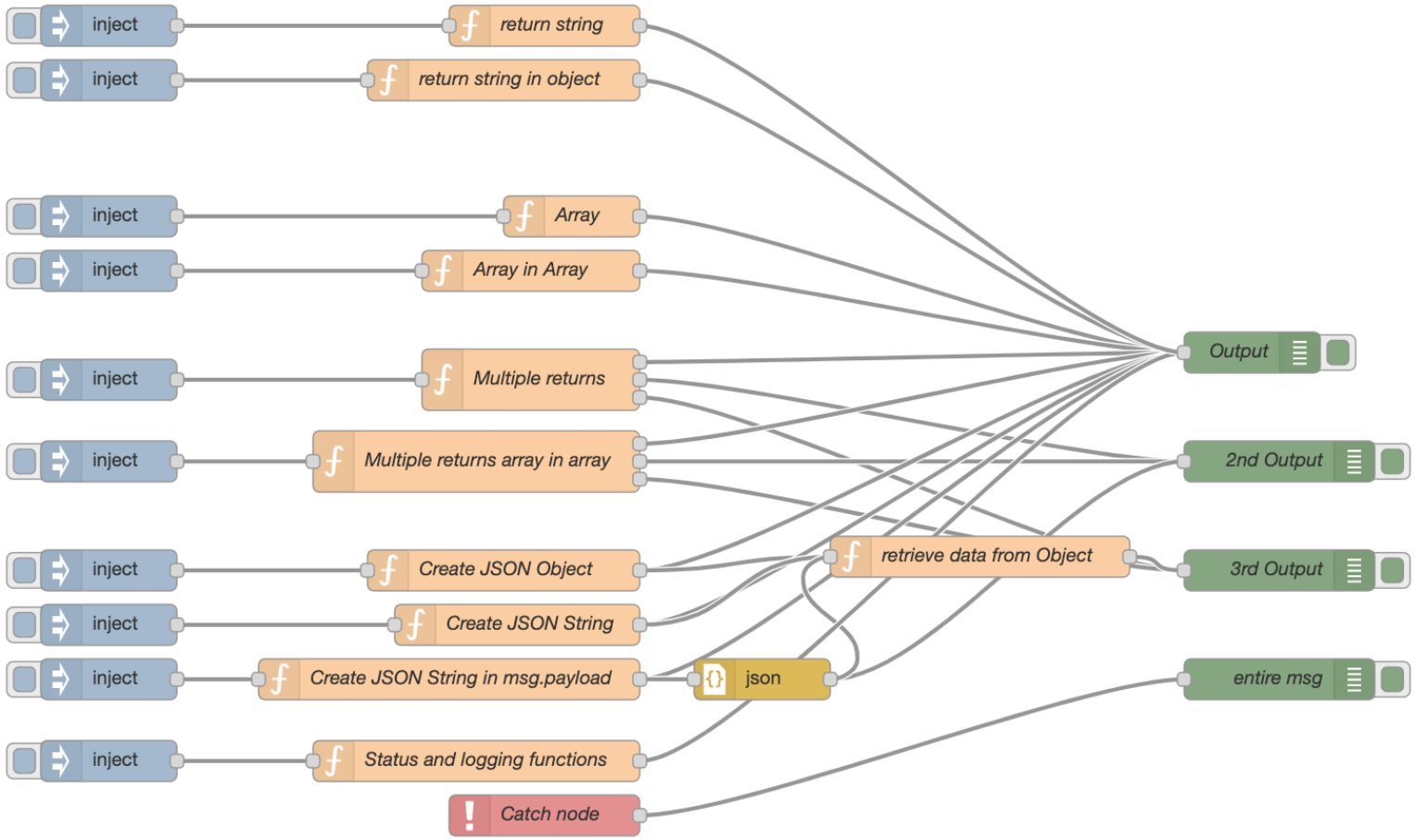


Function Node & Output

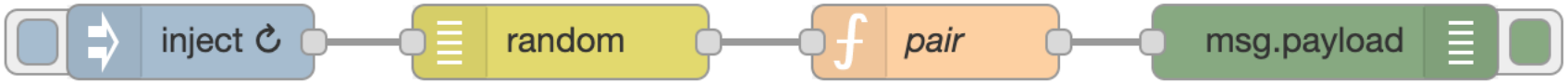
1. Open the Debug panel



2. Try to trigger each inject node and check the Output



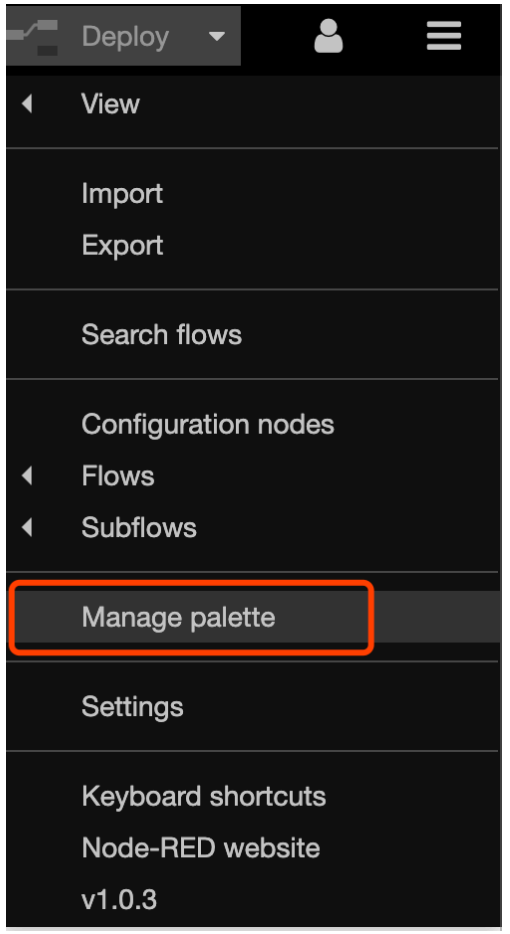
Pre-defined Function Node - Random



The inject Node is set to repeat triggering every 5 seconds, it sends the event through the flow – triggering the random node to generate a random number between 0 and 10 every 5 seconds. The generated number is sent to the *pair* function node to generate the output message based on the number value.

Import a node – Random Node

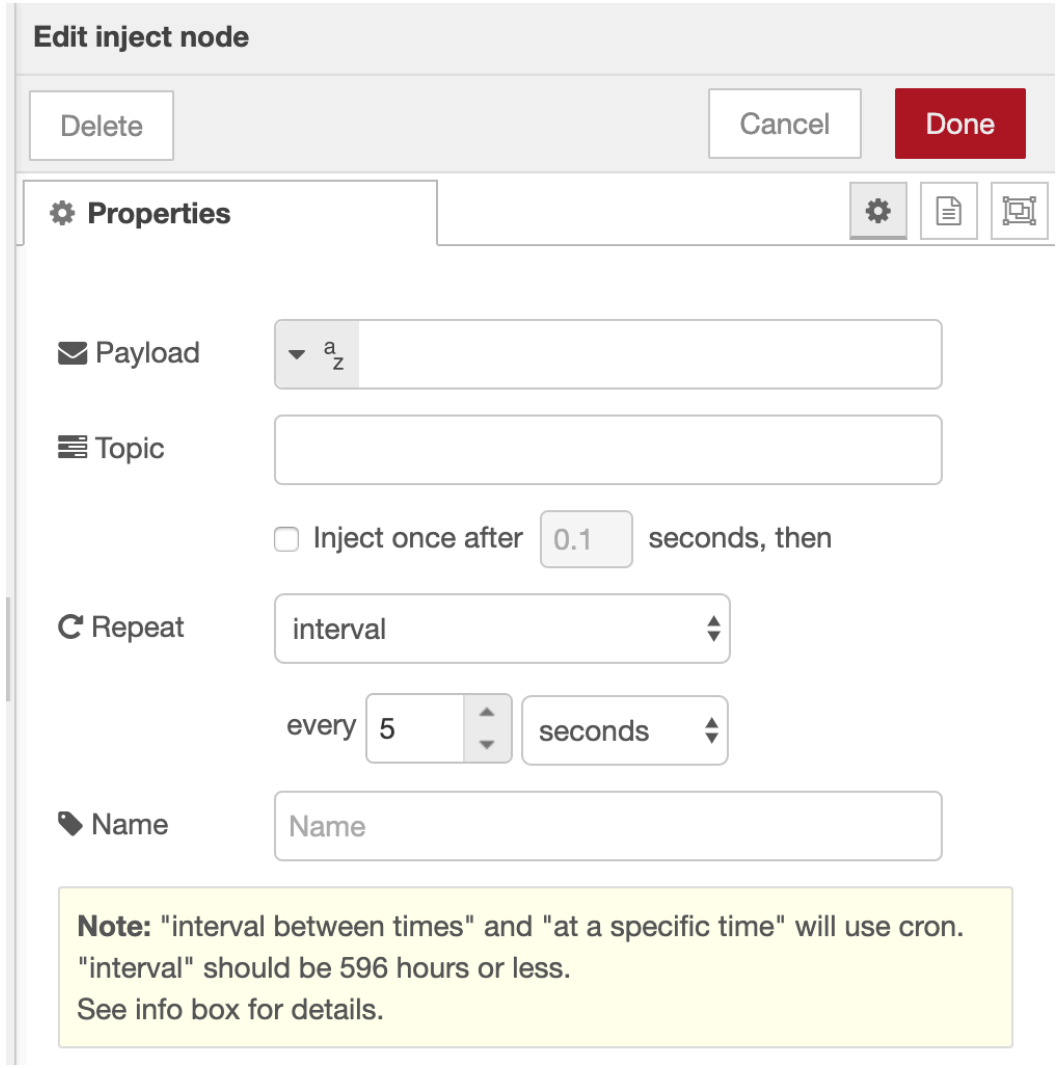
1. Open the Manage Palette under Menu
2. Select Install tab
3. Search for “random”
4. Install “node-red-node-random”
5. Confirm if prompted



***To find more modules, go to https://flows.nodered.org/search?type=node*

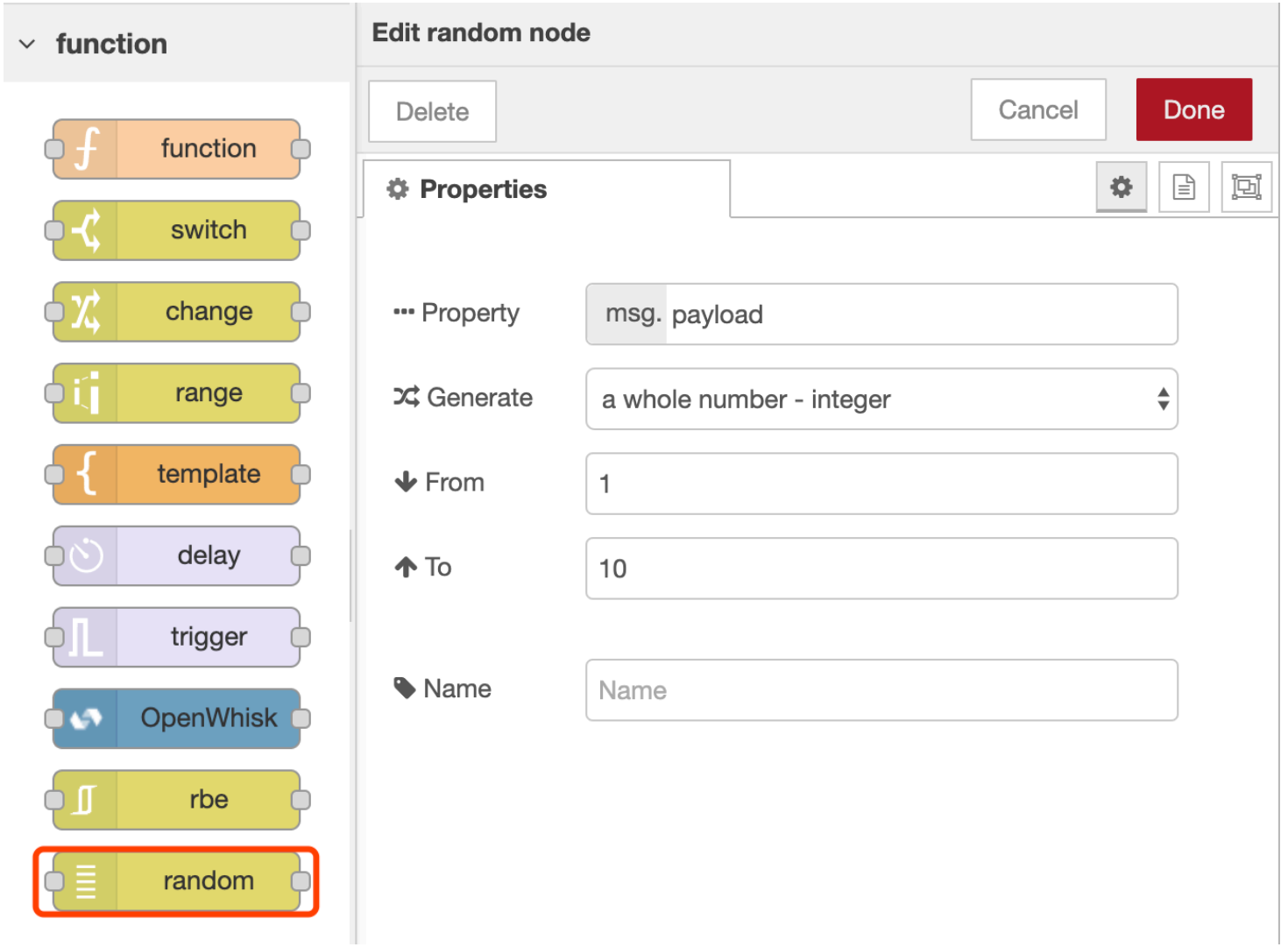
Pre-defined Function Node - Random

1. Get the inject node from the node palette
2. Set to repeat every 5 seconds
3. Click Done to save the changes



Pre-defined Function Node - Random

1. Get the random node from the node palette (Under function section)
2. Set to generate integer number
3. Set the range as from 1 to 10
4. Click Done to save the changes



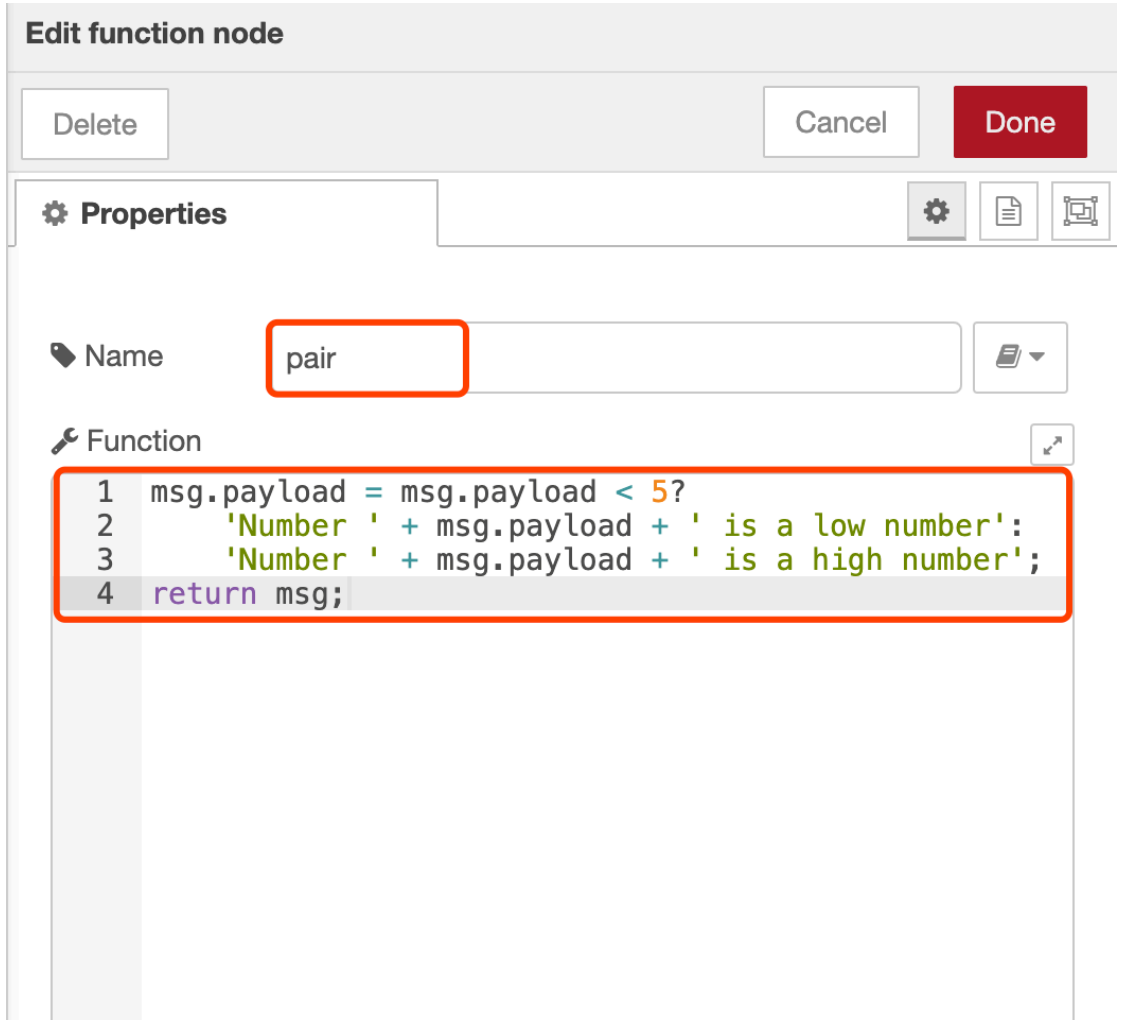
Pre-defined Function Node - Random

1. Get the function node from the node palette (Under function section)
2. Name this function as pair
3. Define the function as shown on the picture. If the number is less than 5, set the msg as

   'Number X is a low number',

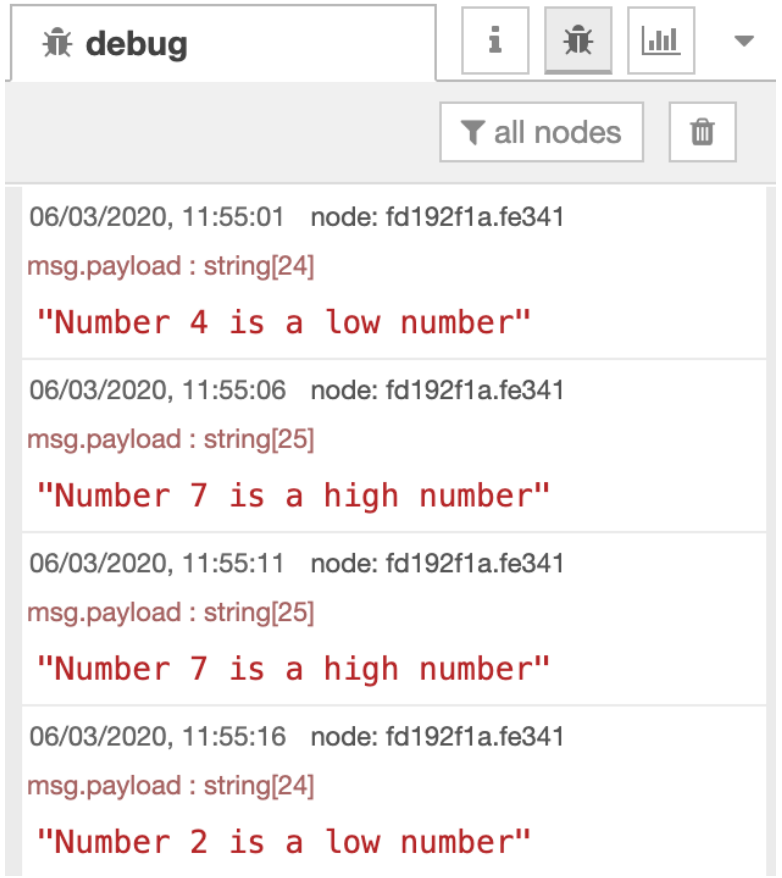
   otherwise

   'Number X is a high number'
4. Click Done to save the changes

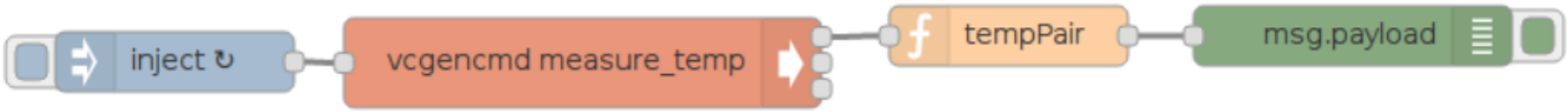


Pre-defined Function Node - Random

1. Get the debug node from the node palette
2. Keep the properties of this node as default value
3. Click Deploy to deploy this flow
4. Check the output in the debug panel



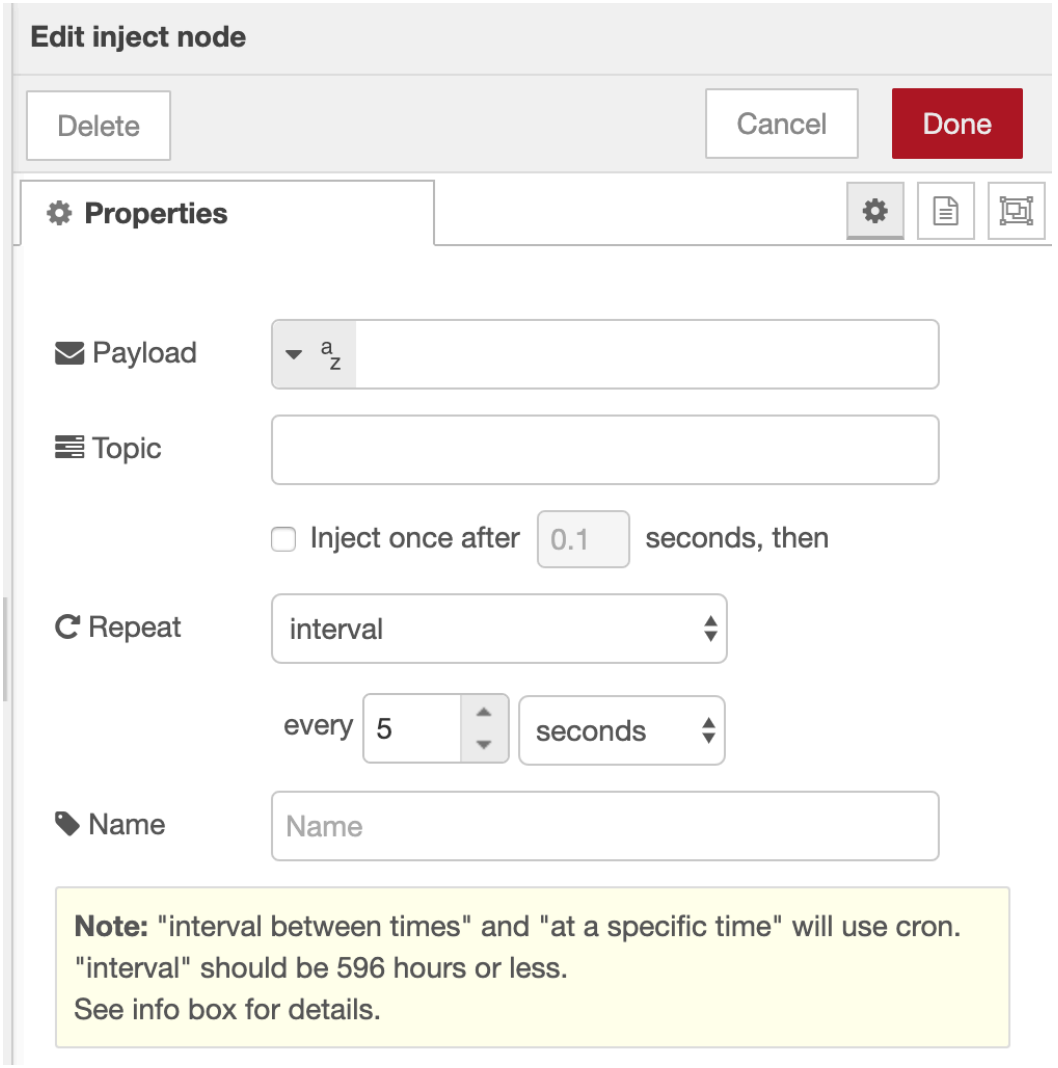
Exec Node



In this practice, you'll use NodeRED as the runtime to create a flow to get the CPU temperature of your Raspberry Pi using the `vcgencmd` and publish the temperature in the appropriate format.

Exec Node

1. Get the inject node from the node palette
2. Set to repeat every 5 seconds
3. Click Done to save the changes



Exec Node

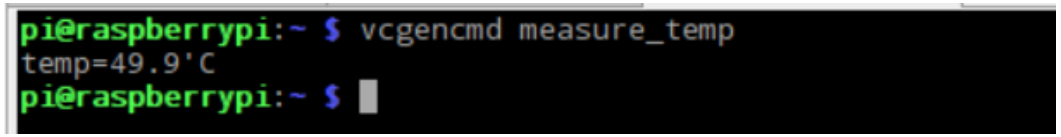
1. Get the exec node from the node palette

   (Under function section) 

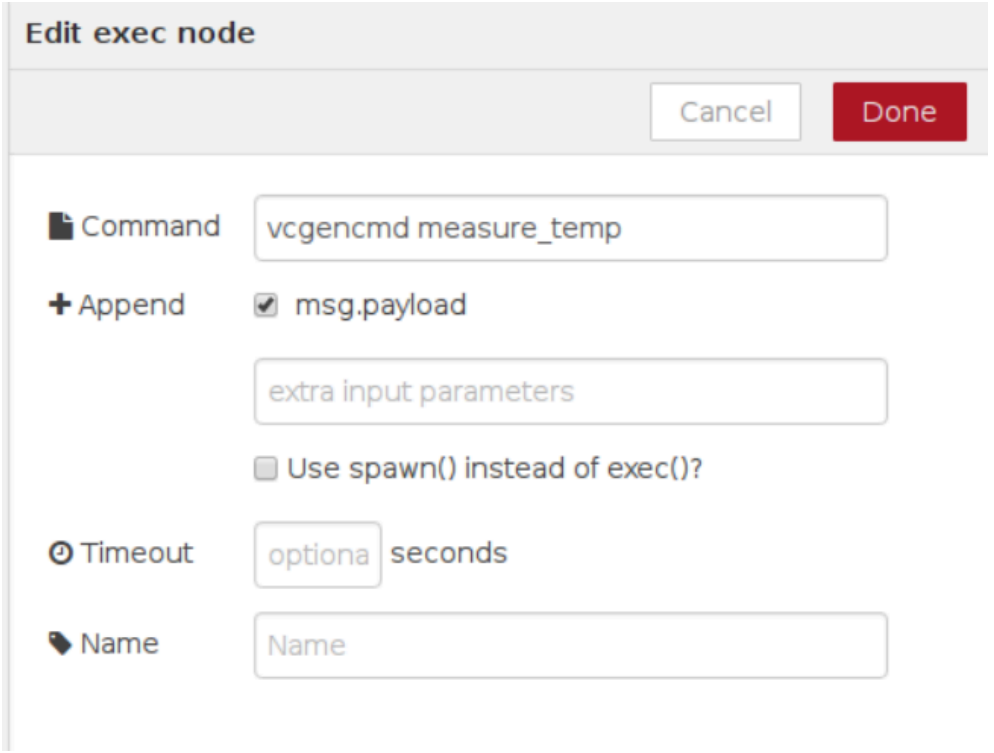
2. Add command

   `vcgencmd measure_temp`

   which also can be run in the terminal

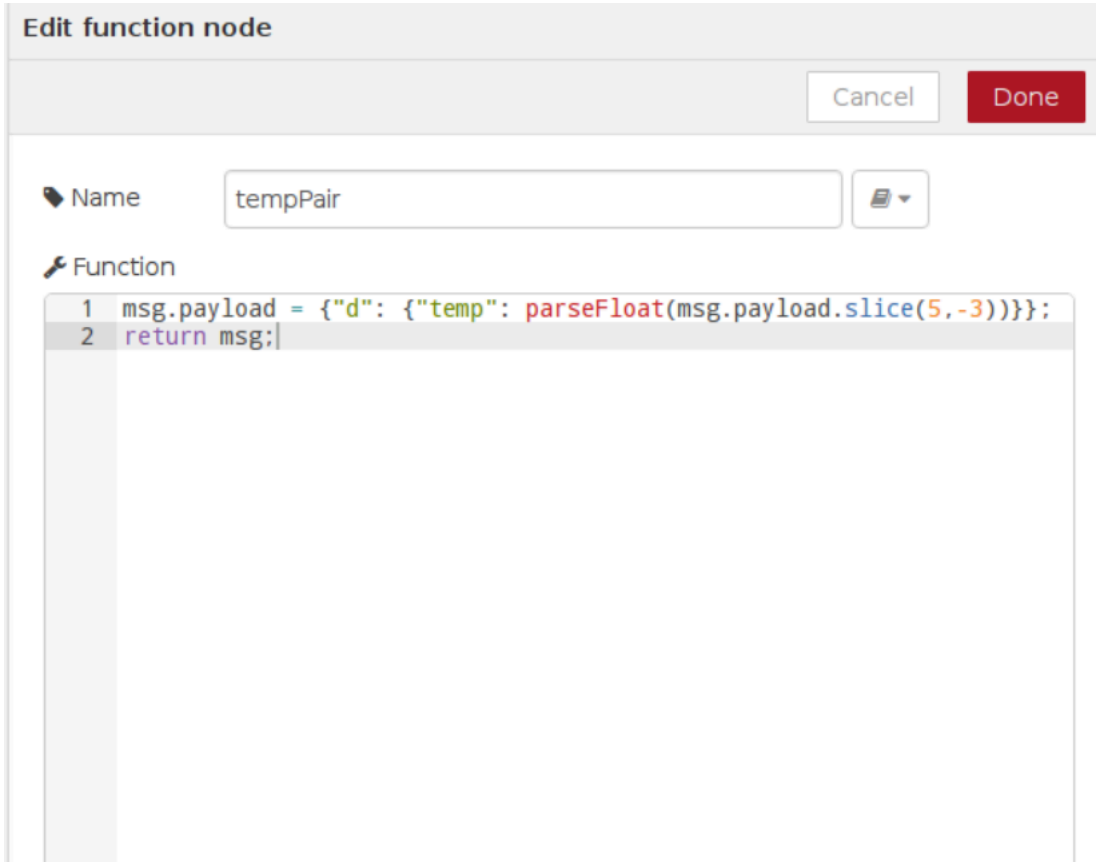


3. Click Done to save the changes



Exec Node

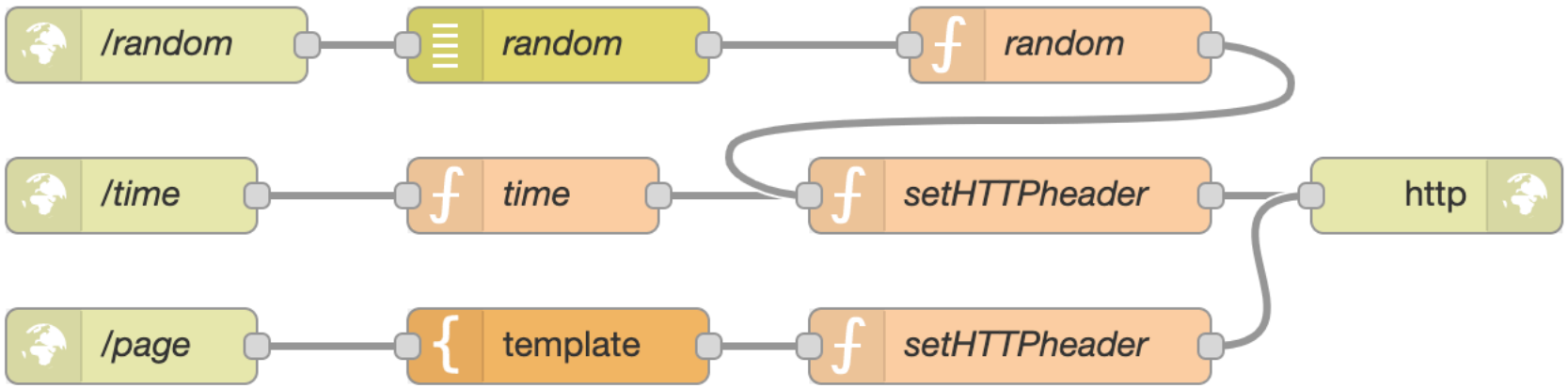
1. Get the function node from the node palette (Under function section)
2. Name this function as tempPair
3. Define the function as shown on the picture. Using parseFloat() function to parse the value returned from the vcgencmd: <CPU temperature> to float type instead of a string in the object.
4. Click Done to save the changes



Exec Node

1. Get the debug node from the node palette
2. Keep the properties of this node as default value
3. Click Deploy to deploy this flow
4. Check the output in the debug panel

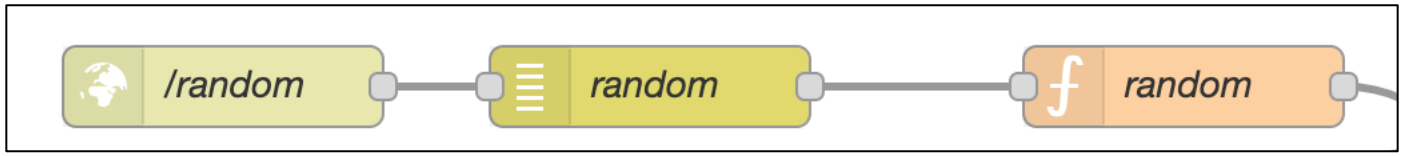
Http Node



In this practice, we'll create an application that will accept:

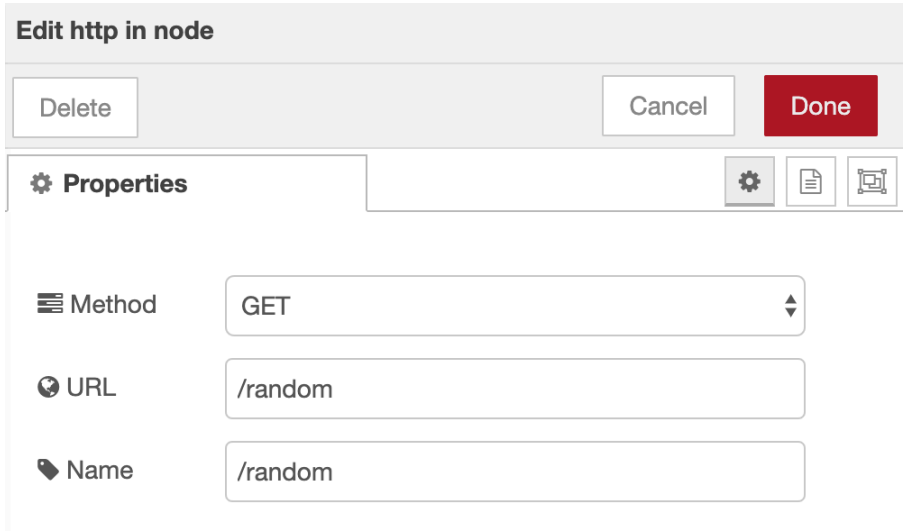
- an http GET request to http://localhost:1880/time that returns a JSON document containing the current time on your RPI
- an http GET request to http://localhost:1880/random that returns a JSON document containing a random number
- an http GET request to http://localhost:1880/page that returns an http page that shows:
    1. when the last time was requested
    2. when the last random number was requested
    3. the value of the last random number generated

Http Node



Http Request Node

1. Get the http input node from the node palette (Under input section)
2. Set URL and Name as below:

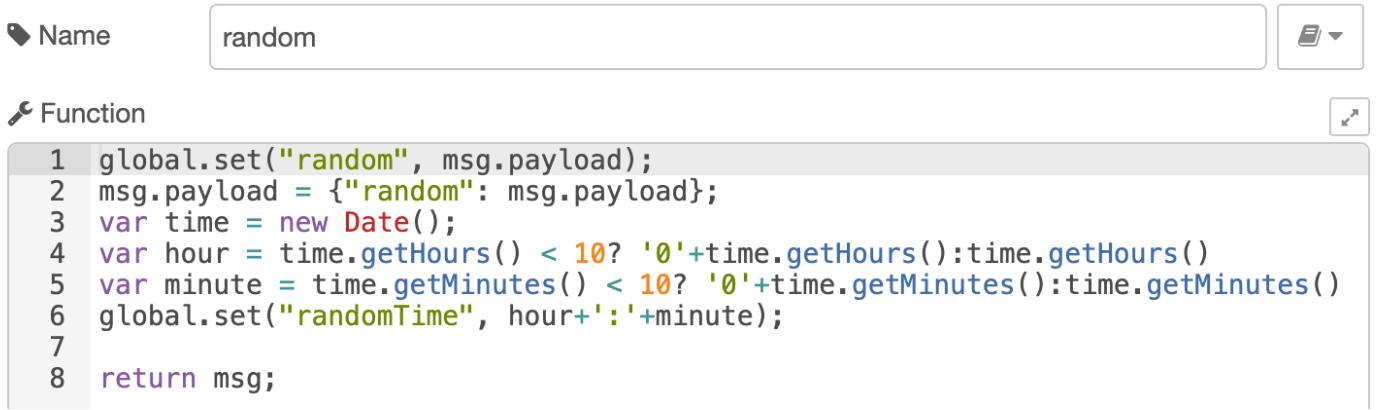


Random Node

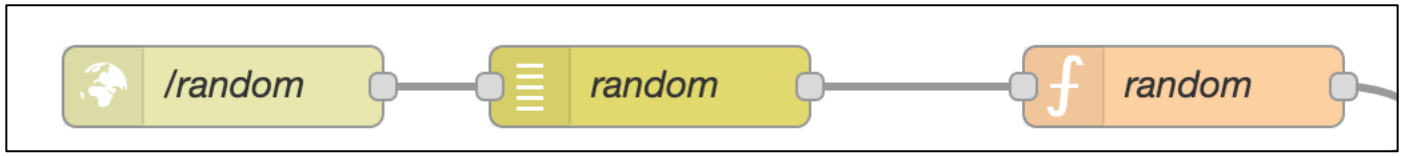
1. Get Random node and set to generate integer from 1 to 10

Function Node

1. Get the function node and define as:



Http Node



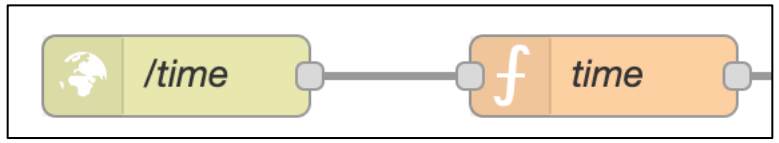
Function Node

Get the function node and define as:

```
global.set("random", msg.payload);
msg.payload = {"random": msg.payload};
var time = new Date();

var hour = time.getHours() < 10?
'0'+time.getHours():time.getHours()

var minute = time.getMinutes() < 10?
'0'+time.getMinutes():time.getMinutes()

global.set("randomTime", hour+':'+minute);
return msg;
```

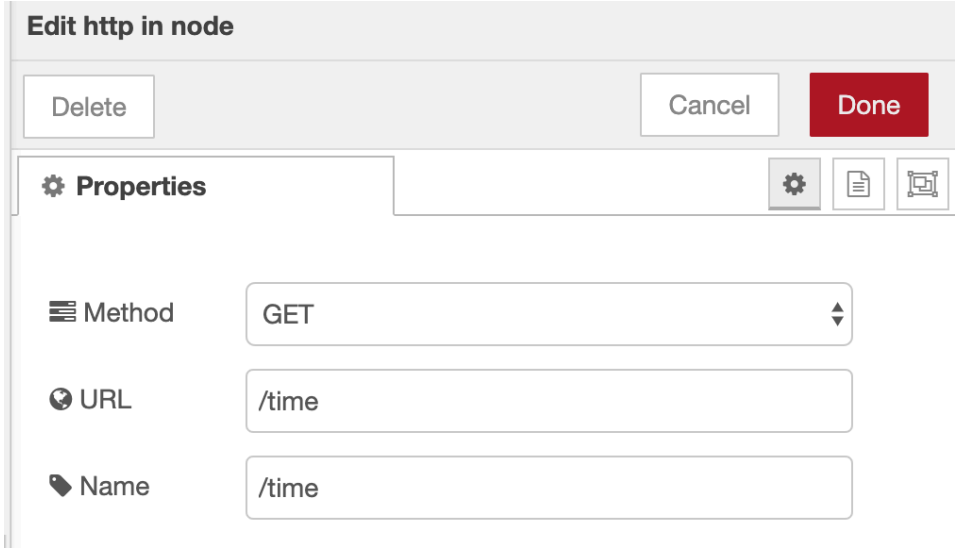
1. Using `global.set()` function to set global variables which can be used in other flows
2. Create time object by `new Date()` to fetch current time.
3. Formatting the time for displaying.
   - `timeObject.getHours()`
   - `timeObject.getMinutes()`

Http Node



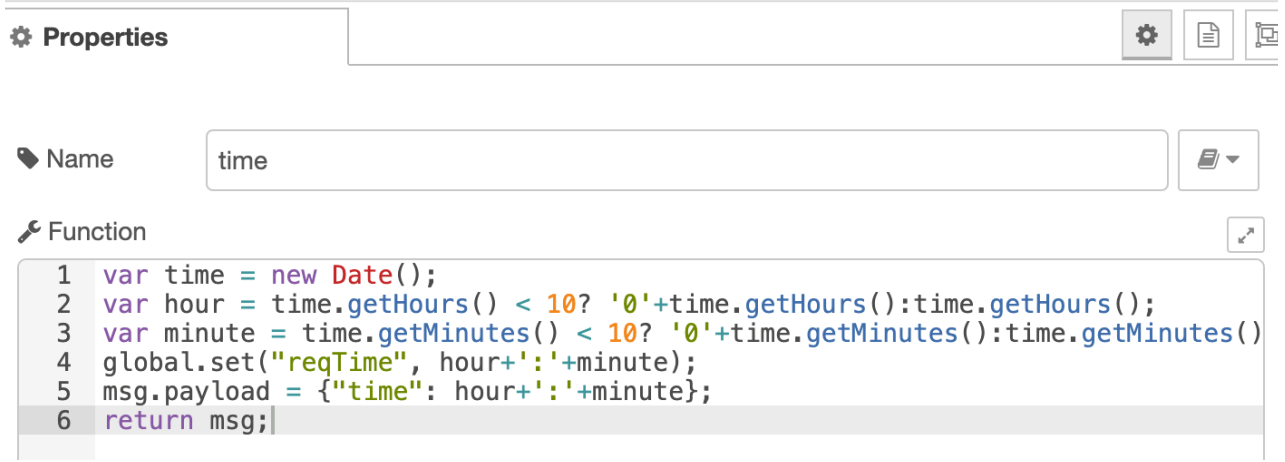
Http Request Node

1. Get the http input node from the node palette (Under input section)
2. Set URL and Name as below:



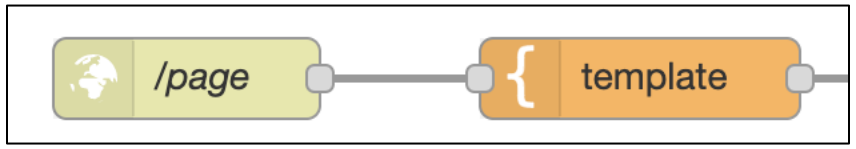
Function Node

Get the function node and define as:



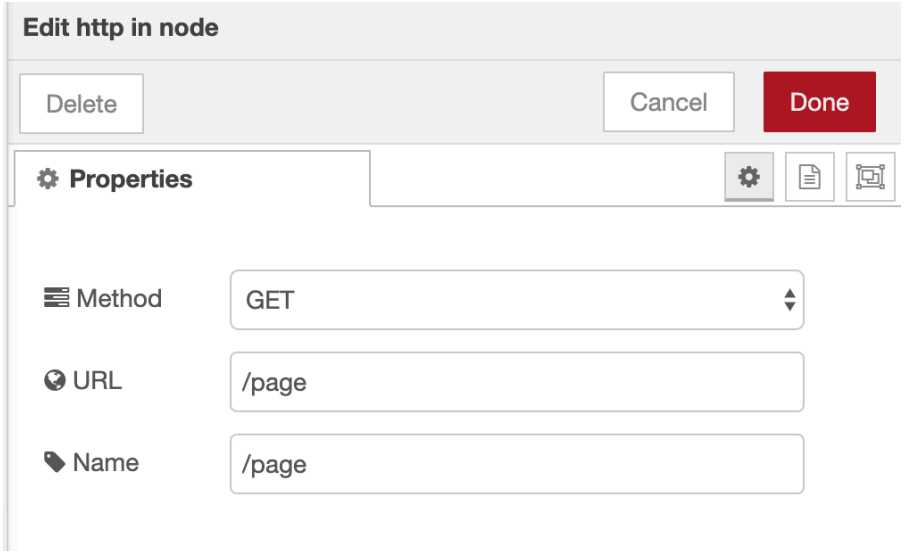

```
var time = new Date();
var hour = time.getHours() < 10? '0'+time.getHours():time.getHours();
var minute = time.getMinutes() < 10? '0'+time.getMinutes():time.getMinutes();
global.set("reqTime", hour+':'+minute);
msg.payload = {"time": hour+':'+minute};
return msg;
```

Http Node



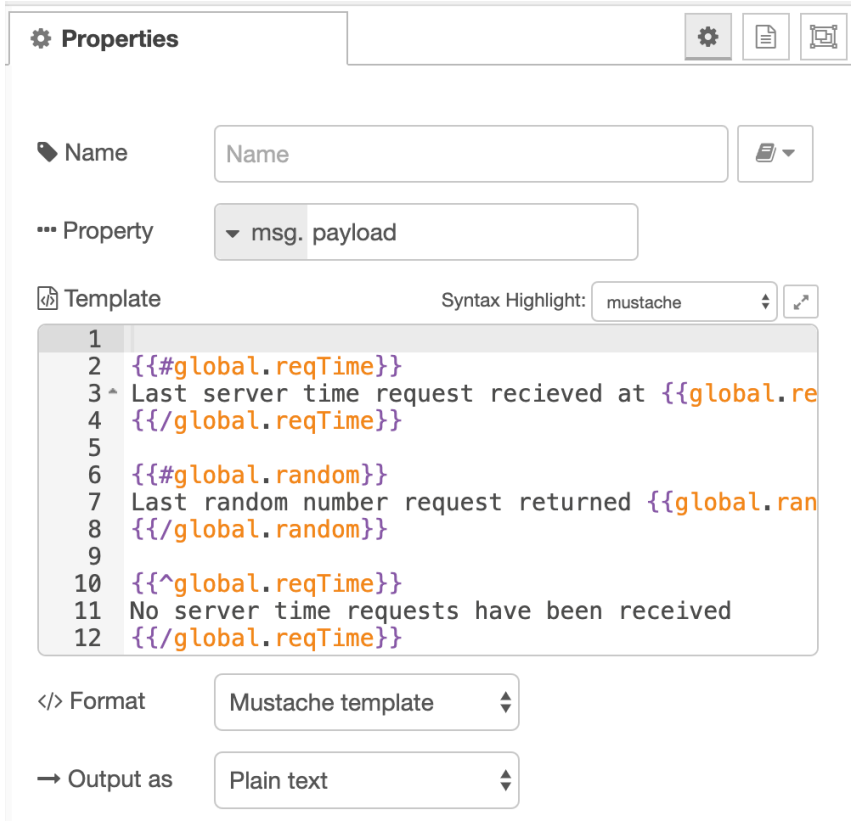
Http Request Node

1. Get the http input node from the node palette (Under input section)
2. Set URL and Name as below:

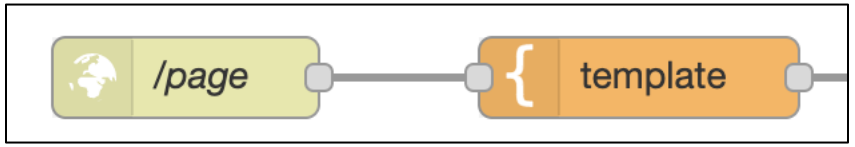


Template Node

Get the template node and edit as:



Http Node

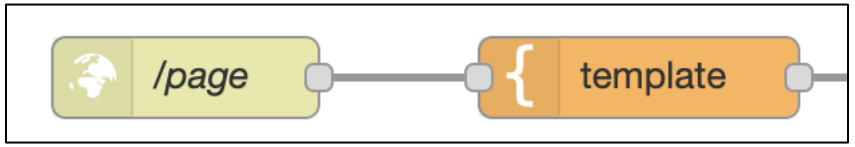


The template node can be used to generate text using a message's properties to fill out a template. It uses the Mustache templating language to generate the result.

Mustache Syntax

- Mustache can be used for HTML, config files, source code - anything.
- It works by expanding tags in a template using values provided in a hash or object.
- It is "logic-less" because there are no if statements, else clauses, or for loops.
  - There are only tags.
  - Some tags are replaced with a value, some nothing, and others a series of values.

Http Node



The template node can be used to generate text using a message's properties to fill out a template.
It uses the Mustache templating language to generate the result.

Mustache Syntax

Variables

- {{***name***}}

  will find the ***name*** key in the current context.

  If not found, nothing will be rendered.

- {{{***name***}}}

  return unescaped HTML.

- {{***& name***}}

  Return unescaped variable.

Sections

- {{***#person***}} section content {{***/person***}}

  The behavior of the section determined by its value.

  - *False Values or Empty Lists* - not be displayed
  - *Non-Empty Lists* - displayed once for each item in the list
  - *Non-False Values but not a list* - used as the context

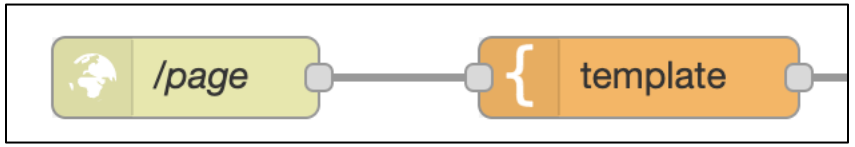
Inverted Sections

- {{***^person***}} section content {{***/person***}}

  This section will be rendered if the key doesn't exist,

  is false, or is an empty list.

Http Node



The template node can be used to generate text using a message's properties to fill out a template. It uses the Mustache templating language to generate the result.

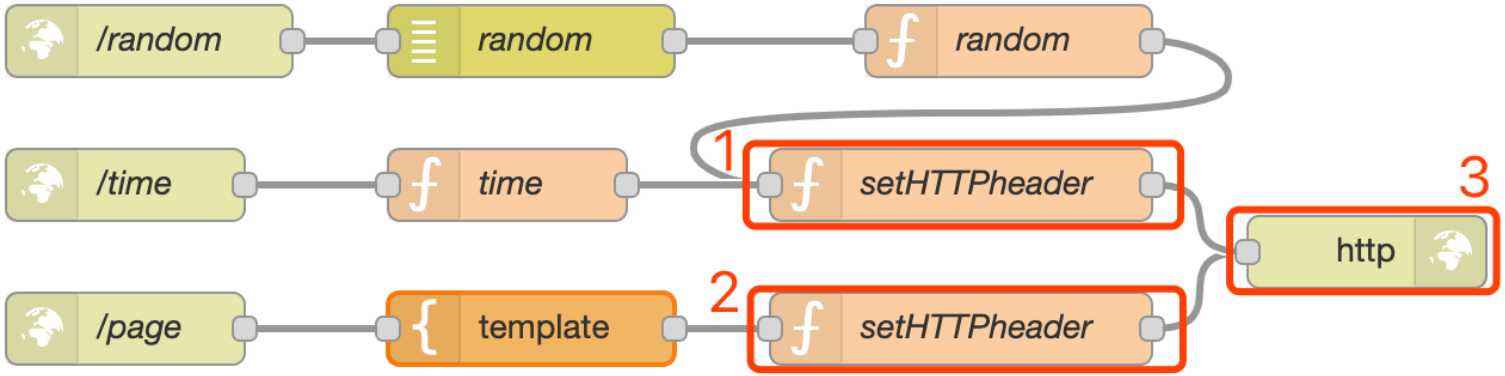
Mustache Syntax

Edit template node > **Text editor**

Cancel | Done

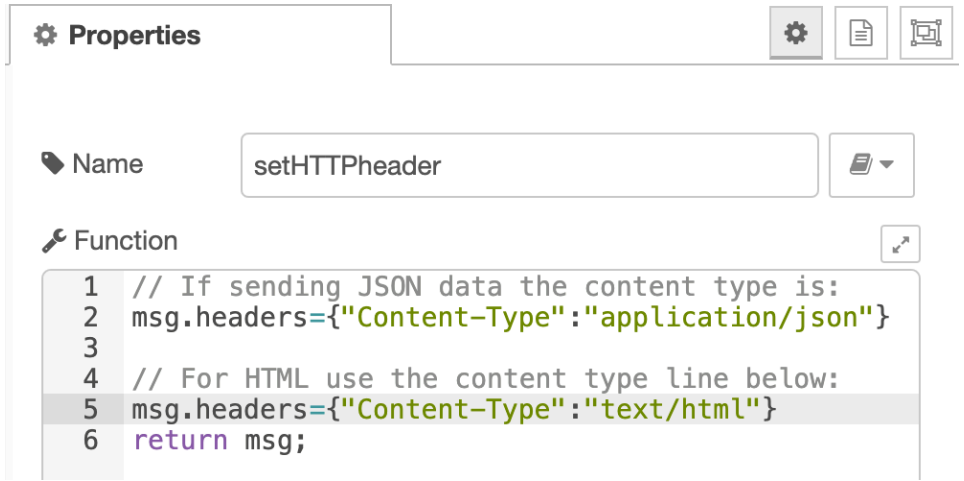
```
1
2  {{#global.reqTime}}
3  Last server time request recieved at {{global.reqTime}} </br>
4  {{/global.reqTime}}
5
6  {{#global.random}}
7  Last random number request returned {{global.random}}, which was received at {{global.randomTime}}
8  {{/global.random}}
9
10 {{^global.reqTime}}
11 No server time requests have been received
12 {{/global.reqTime}}
13
14 {{^global.random}}
15 No requests for random numbers have been received
16 {{/global.random}}
```

Http Node



Set Http Header

Get the function node and define as:



Http Response Node

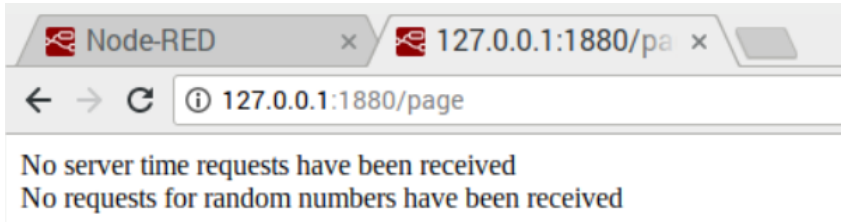
Get the http response node and keep it as default.

Make sure all nodes are connected correctly then Let's Deploy

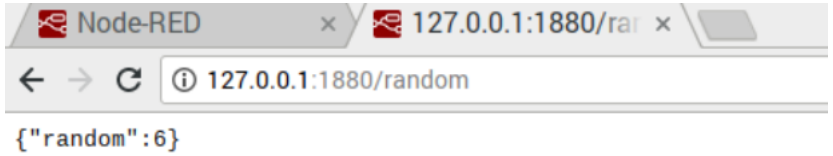
Http Node

Once finished these flows, open the URLs below in web browser to check the results:

1. http://localhost:1880/page (Check this page first to see the result when there's no stored global variables)

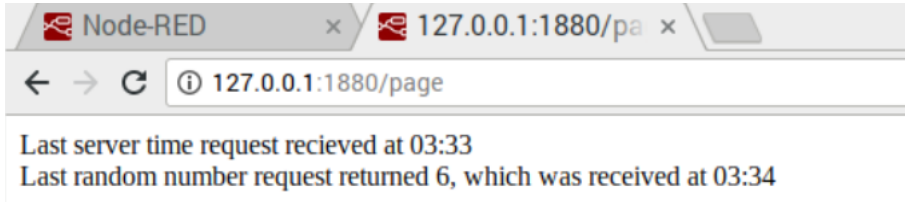


2. http://localhost:1880/random



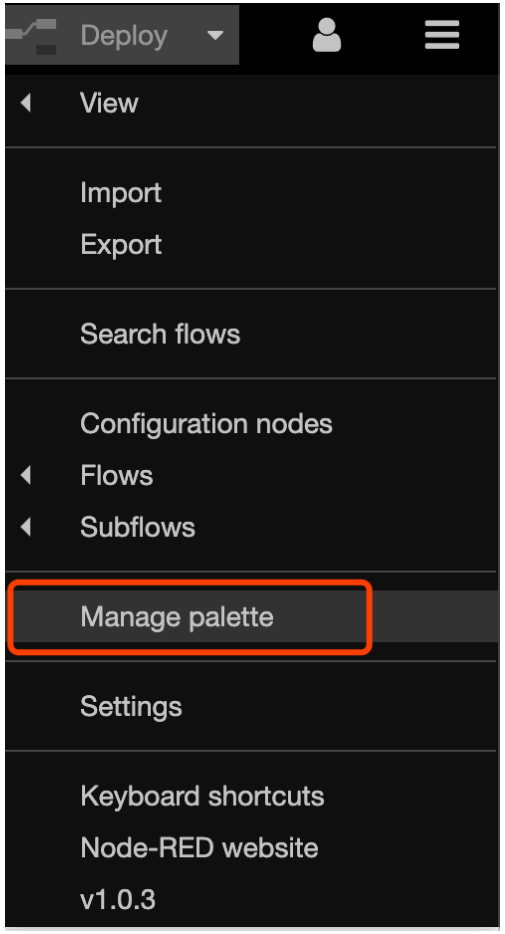
3. http://localhost:1880/time

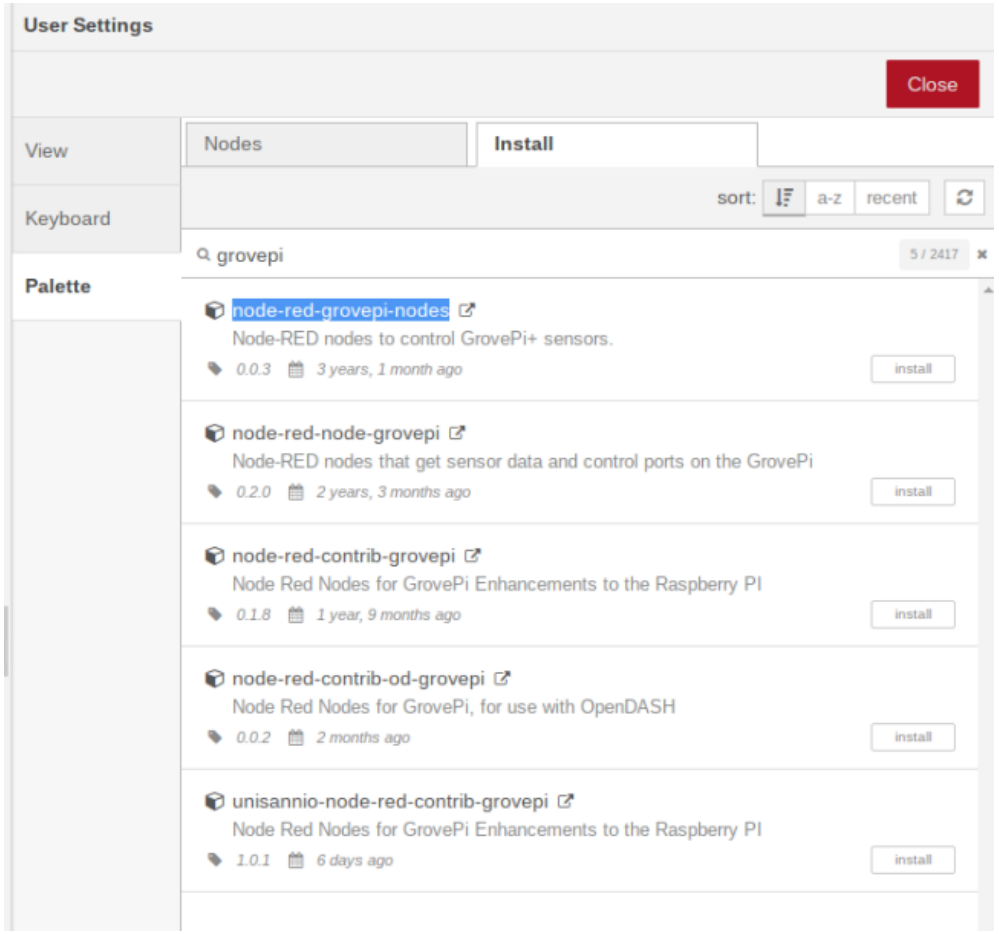
4. http://localhost:1880/page



Import a module – GrovePi Nodes

1. Open the Manage Palette under Menu
2. Select Install tab
3. Search for “grovepi”
4. Install “node-red-grovepi-nodes”
5. Confirm if prompted

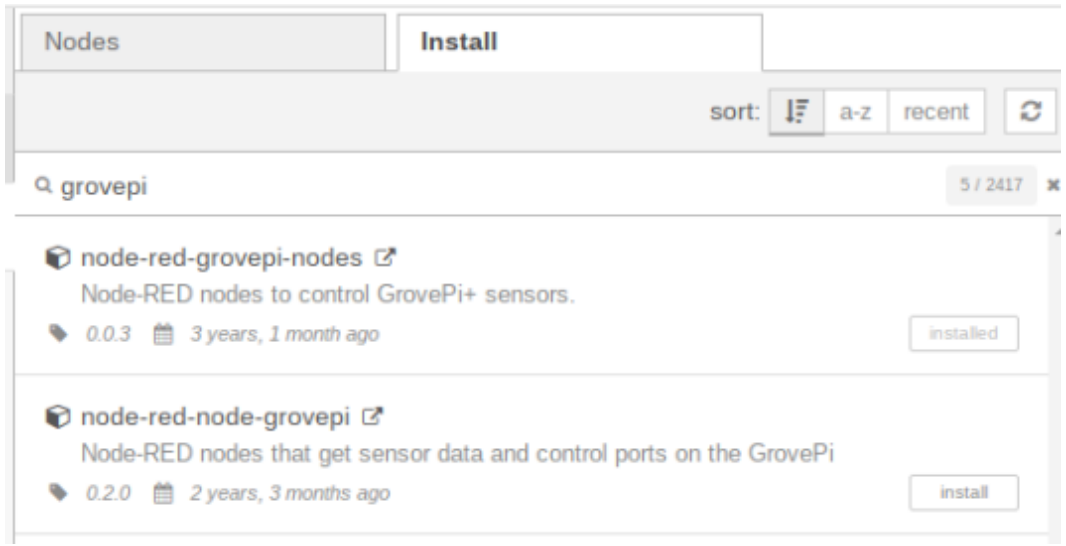


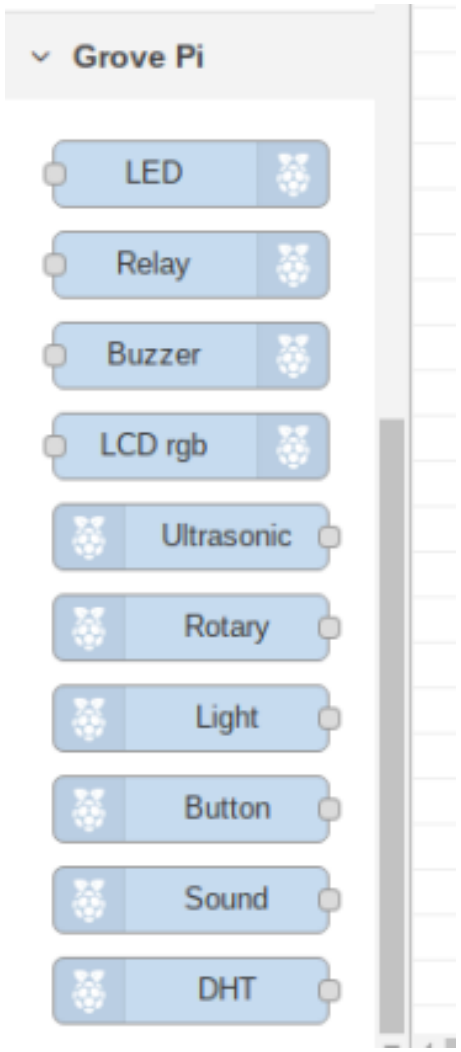


***To find more modules, go to https://flows.nodered.org/search?type=node*

Import a module – GrovePi Nodes

1. Check if Installed
2. Close settings menu
3. On the nodes palette, check if the GrovePi nodes are available






IBM®