Working with Node-red Nodes

Working with Function node:

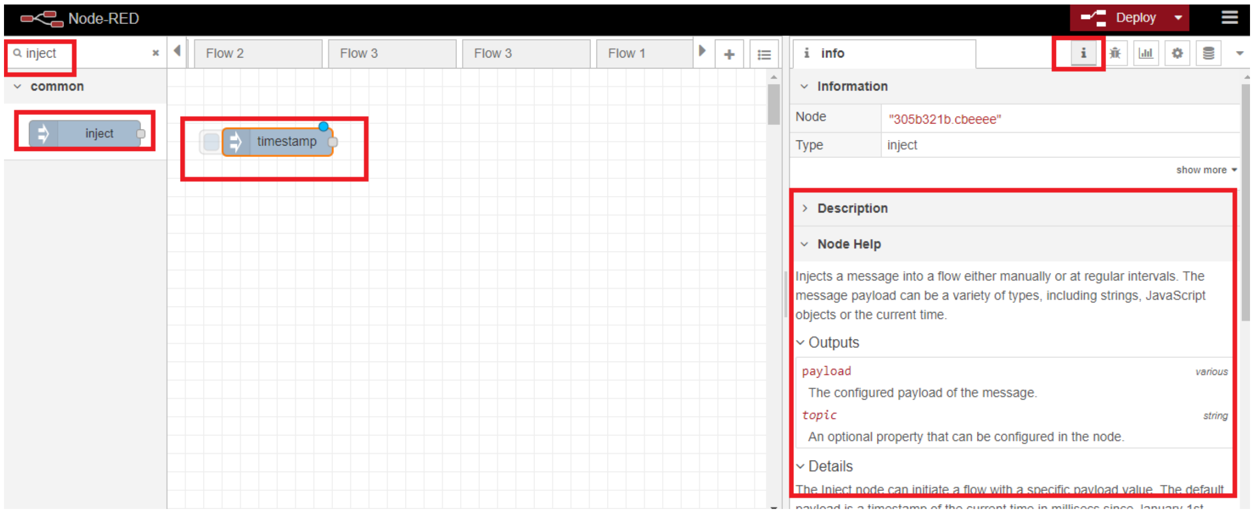
In this activity introduces the Node-RED editor and creates a flow the demonstrates the Inject, Debug and Function nodes.

Inject node:

The Inject node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.

Drag one onto the workspace from the palette.

Select the newly added Inject node to see information about its properties and a description of what it does in the Information sidebar pane.



Node	"305b321b.cbeeee"
Type	inject

Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time.

payload *various*
The configured payload of the message.
topic *string*
An optional property that can be configured in the node.

The Inject node can initiate a flow with a specific payload value. The default

Add Debug Node:

The Debug node causes any message to be displayed in the **Debug sidebar**. By default, it just displays the payload of the message, but it is possible to display the entire message object.

Wire the two together:

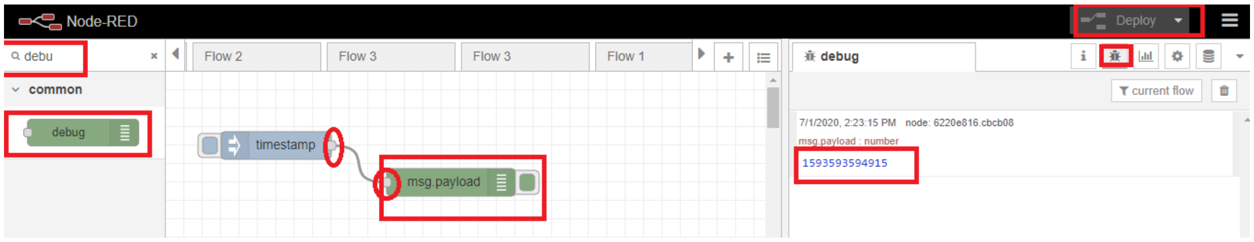
Connect the Inject and Debug nodes together by dragging between the output port of one to the input port of the other.

Deploy

At this point, the nodes only exist in the editor and must be deployed to the server.

Click the Deploy button.

With the Debug sidebar tab selected, click the Inject button. You should see numbers appear in the sidebar. By default, the Inject node uses the number of milliseconds since January 1st, 1970 as its payload.



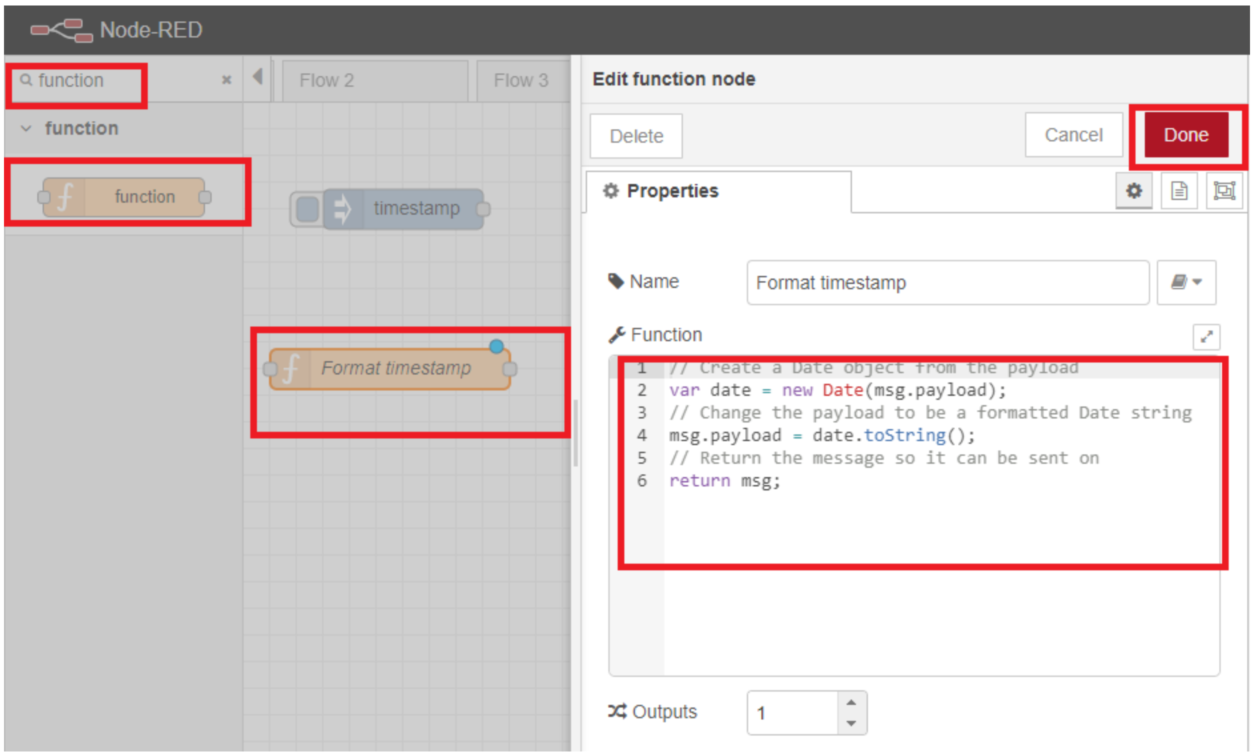
```
7/1/2020, 2:23:15 PM   node: 6220e816.cbcb08
msg.payload : number
1593593594915
```

Add Function Node:

The Function node allows you to pass each message though a JavaScript function.
Delete the existing wire (select it and press delete on the keyboard).
Wire a Function node in between the Inject and Debug nodes.
Double-click on the Function node to bring up the edit dialog. Copy the following code into the function field:

```
// Create a Date object from the payload
var date = new Date(msg.payload);
// Change the payload to be a formatted Date string
msg.payload = date.toString();
// Return the message so it can be sent on
return msg;
```

Click **Done** to close the edit dialog.

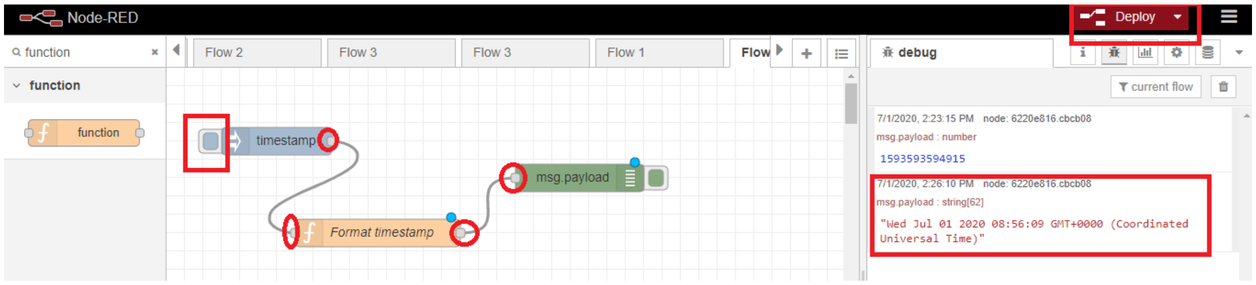


Name: Format timestamp

```
// Create a Date object from the payload
var date = new Date(msg.payload);
// Change the payload to be a formatted Date string
msg.payload = date.toString();
// Return the message so it can be sent on
return msg;
```

Outputs: 1

Click the **deploy** button. Now when you click the Inject button, the messages in the sidebar will now be formatted is readable timestamps.



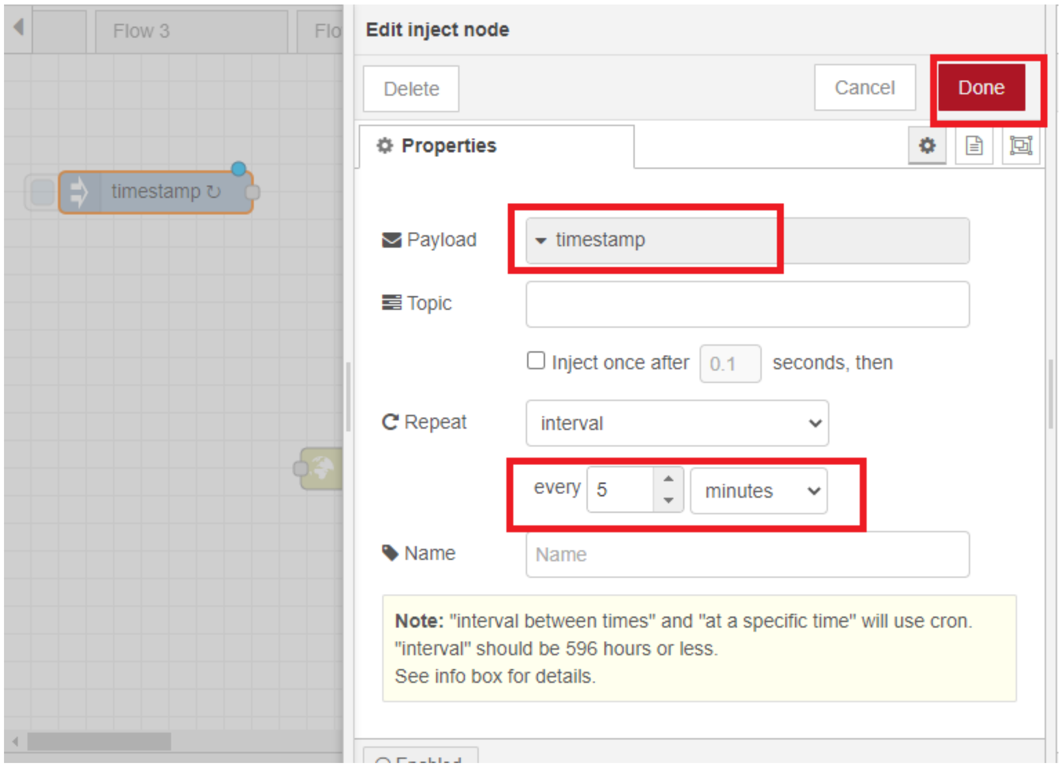
```
7/1/2020, 2:23:15 PM   node: 6220e816.cbcb08
msg.payload : number
1593593594915
7/1/2020, 2:26:10 PM   node: 6220e816.cbcb08
msg.payload : string[62]
"Wed Jul 01 2020 08:56:09 GMT+0000 (Coordinated
Universal Time)"
```

Working with Apis:
This flow is automatically triggered every 5 minutes and retrieves data from a url. It parses the data and displays in the Debug sidebar. It also checks the magnitude value in the data and branches the flow for any messages with a magnitude greater than, or equal to, 7. The payloads of such messages are modified and displayed in the Debug sidebar.
The flow will:

- Retrieve information from a website at a regular interval
- Convert that information into a useful form
- Display the result in the Debug sidebar

Add an Inject node:

In the previous Task, the Inject node was used to trigger the flow when its button was clicked. For this Activity, the Inject node will be configured to trigger the flow at a regular interval.
Drag an Inject node onto the workspace from the palette.
Double click the node to bring up the edit dialog. Set the repeat interval to every 5 minutes.
Click Done to close the dialog.



Payload: timestamp
Topic:
Inject once after 0.1 seconds, then
Repeat: interval
every 5 minutes
Name: Name

Note: "interval between times" and "at a specific time" will use cron.
"interval" should be 596 hours or less.
See info box for details.

Add an HTTP Request node:

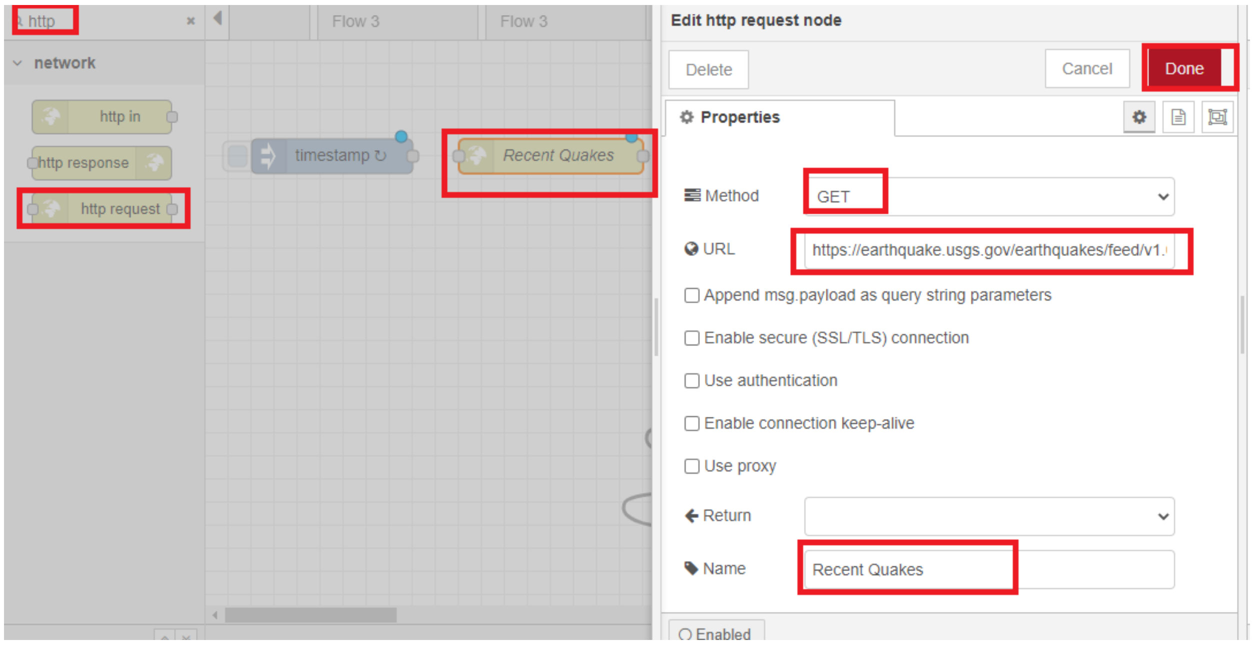
The HTTP Request node can be used to retrieve a web-page when triggered.
After adding one to the workspace, edit it to set the **URL** property to:

```
https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_month.csv
```

h**ttps://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_month.csv**

Then click Done to close the dialog.

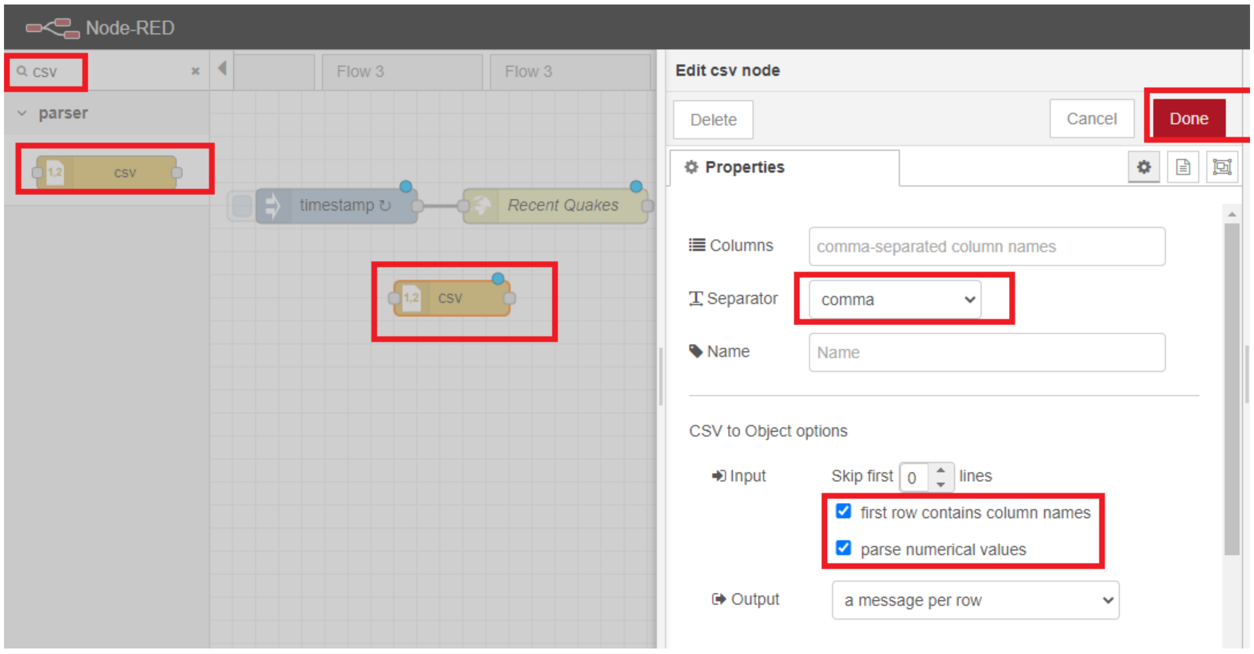
This URL is a feed of significant earthquakes in the last month from the US Geological Survey web site. The site offers a number of other options that you may want to play around with after completing this Guided lab.



Method: GET
URL: https://earthquake.usgs.gov/earthquakes/feed/v1.
☐ Append msg.payload as query string parameters
☐ Enable secure (SSL/TLS) connection
☐ Use authentication
☐ Enable connection keep-alive
☐ Use proxy
Return:
Name: Recent Quakes

Add a CSV node:

Add a CSV node and edit its properties. Enable option for ‘First row contains column names’.
Then click Done to close.



Columns: comma-separated column names
Separator: comma
Name: Name

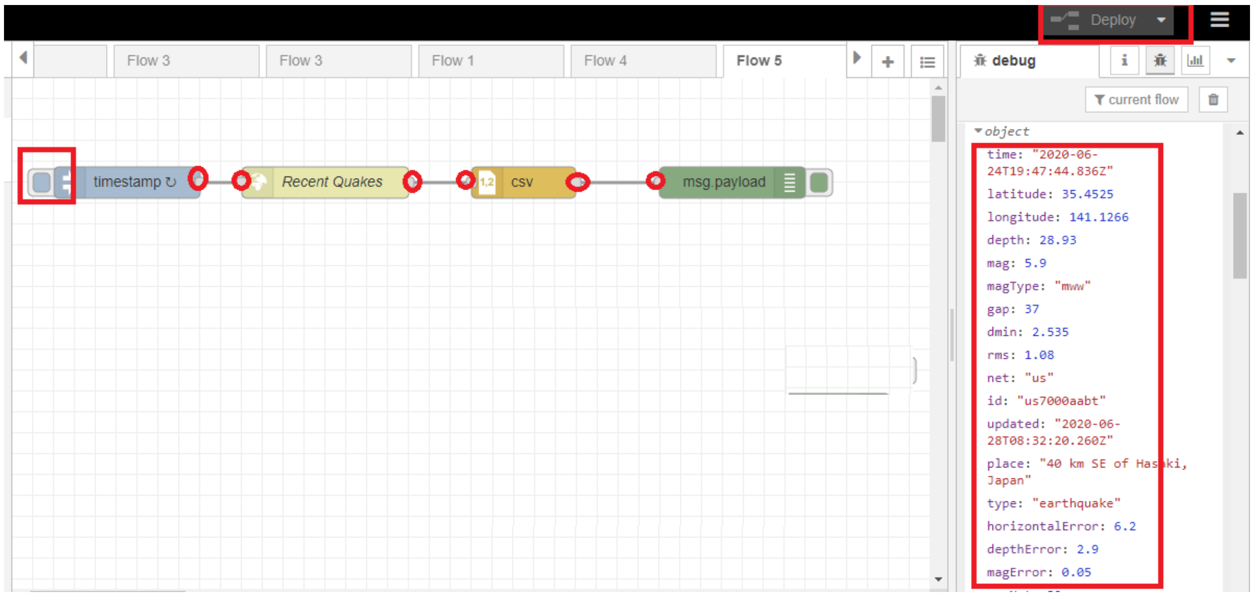
CSV to Object options

Input: Skip first 0 lines
☑ first row contains column names
☑ parse numerical values

Output: a message per row

Add a Debug node
Add a Debug node to the output.
Wire them all together
Add wires connecting:

- The Inject node output to the HTTP Request node input.
- The HTTP Request node output to the CSV node input.
- The CSV node output to the Debug node input.



```
▾object
time: "2020-06-
24T19:47:44.836Z"
latitude: 35.4525
longitude: 141.1266
depth: 28.93
mag: 5.9
magType: "mww"
gap: 37
dmin: 2.535
rms: 1.08
net: "us"
id: "us7000aabt"
updated: "2020-06-
28T08:32:20.260Z"
place: "40 km SE of Hasaki,
Japan"
type: "earthquake"
horizontalError: 6.2
depthError: 2.9
magError: 0.05
```

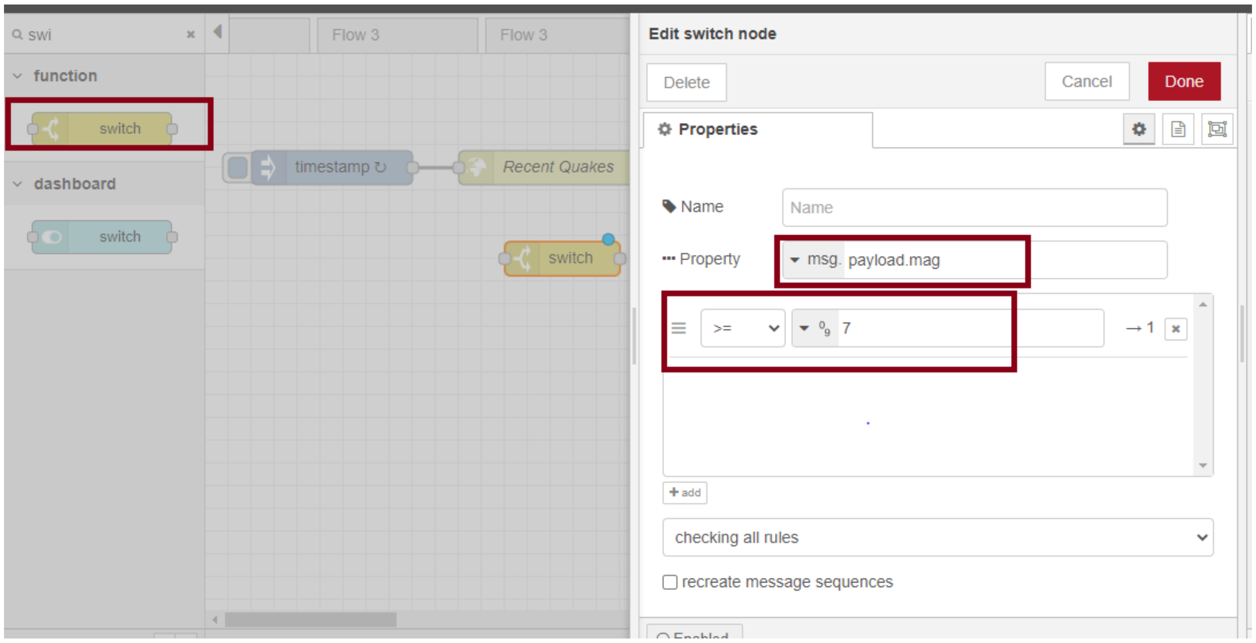
Click on inject node to see the output Deploy the flow to the runtime by clicking the Deploy button. With the Debug sidebar tab open click the Inject button. You should see a list of entries with some contents that look like:

```
msg.payload : Object
{"time":"2017-11-
19T15:09:03.120Z","latitude":-21.5167,"longitude":168.5426,"depth":14.19,"mag":6.6,"magType":"mww"
,"gap":21,"dmin":0.478,"rms":0.86,"net":"us","id":"us2000brgk","updated":"2017-11-
19T17:10:58.449Z","place":"68km E of Tadine, New
Caledonia","type":"earthquake","horizontalError":6.2,"depthError":2.8,"magError":0.037,"magNst":72
,"status":"reviewed","locationSource":"us","magSource":"us"}
```

Add a Switch Node:

Add a Switch node to the workspace. Edit its properties and configure it to check the property msg.payload.mag (of magnitude of earth quake coming from csv output), with a test of >= change it to test on a number and the value 7. Click Done to close.

Add a second wire from the CSV node to this Switch node.



Name: Name

Property: msg. payload.mag

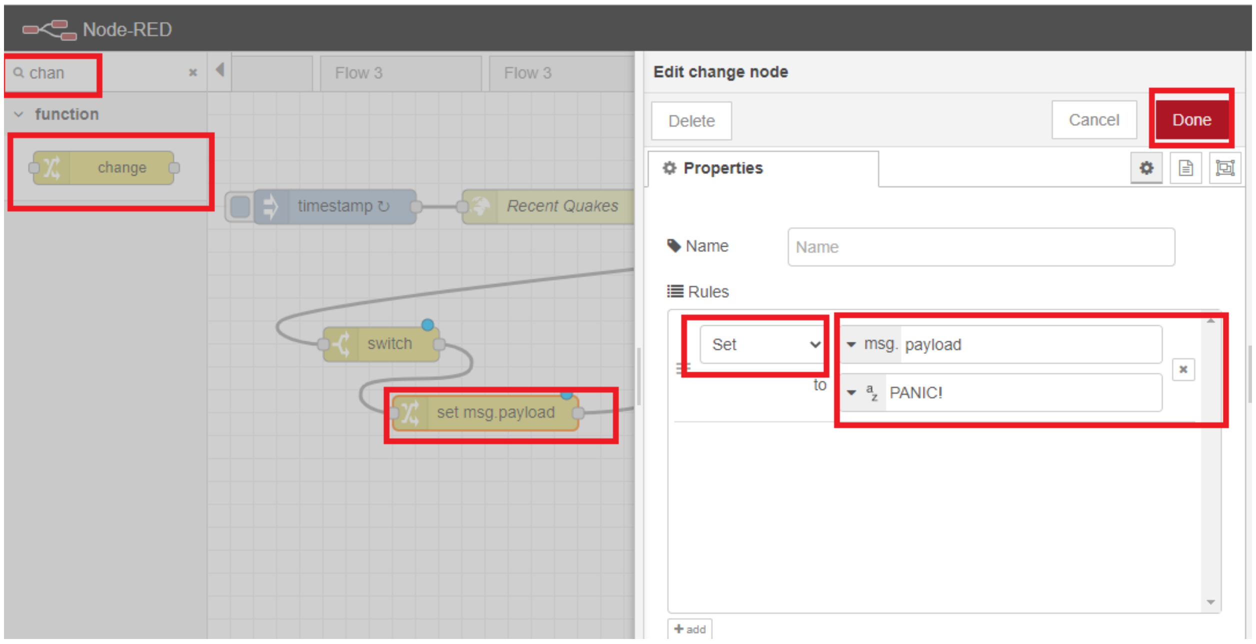
>= 9 7 → 1

checking all rules

recreate message sequences

Add a Change node

Add a Change node, wired to the output of the Switch node. Configure it to set **msg.payload** to the string **PANIC!.**



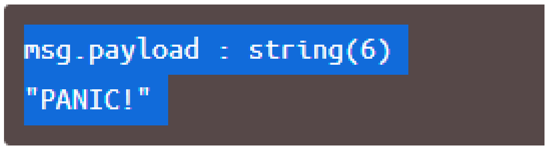
Name: Name

Set msg. payload
to PANIC!

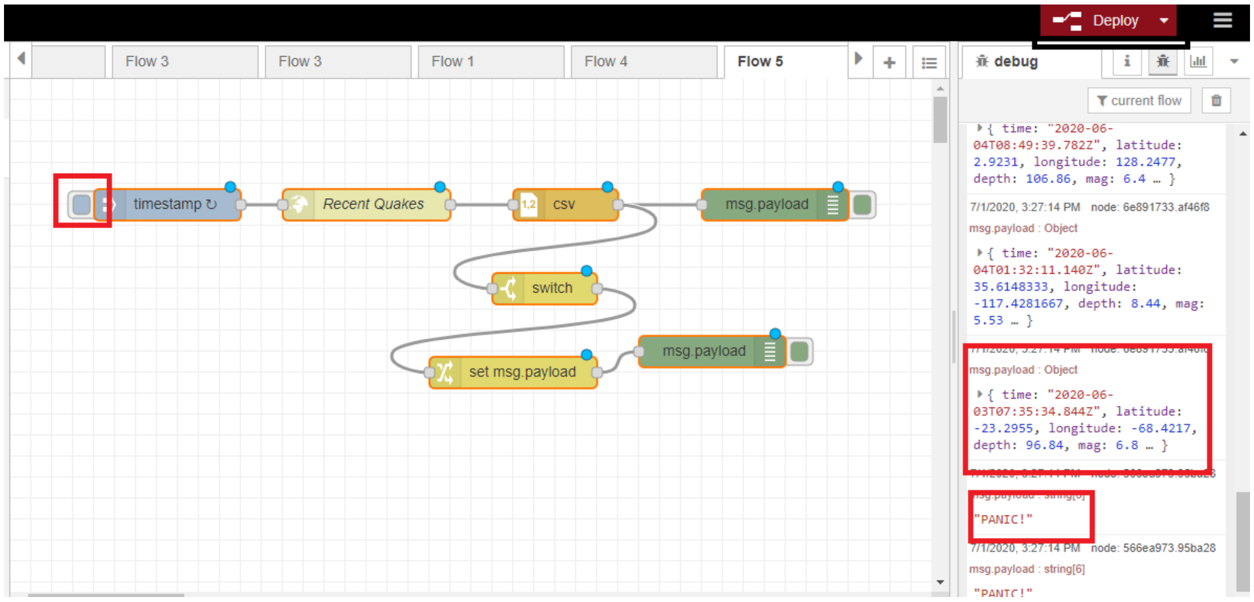
Add a Debug node

Wire up all the Nodes
Click on deploy and Click inject node

You can now click on the little arrow to the left of each property to expand them and examine the contents
If there were any quakes with a magnitude greater than 7 you will also see debug messages like:



```
msg.payload : string(6)
"PANIC!"
```



```
{ time: "2020-06-04T08:49:39.782Z", latitude: 2.9231, longitude: 128.2477, depth: 106.86, mag: 6.4 … }
7/1/2020, 3:27:14 PM   node: 6e891733.af46f8
msg.payload : Object
{ time: "2020-06-04T01:32:11.140Z", latitude: 35.6148333, longitude: -117.4281667, depth: 8.44, mag: 5.53 … }
msg.payload : Object
{ time: "2020-06-03T07:35:34.844Z", latitude: -23.2955, longitude: -68.4217, depth: 96.84, mag: 6.8 … }
"PANIC!"
7/1/2020, 3:27:14 PM   node: 566ea973.95ba28
msg.payload : string[6]
"PANIC!"
```

You could change the switch value of 7 to a smaller one to test your program. Remember to click on deploy after the change.

For info click the below:
https://nodered.org/docs/user-guide/nodes#template

https://nodered.org/docs/user-guide/nodes#template