Build a Node-RED App

That reveals Real-time Earthquakes

Prepared by: Armen Pischdotchian

Overview

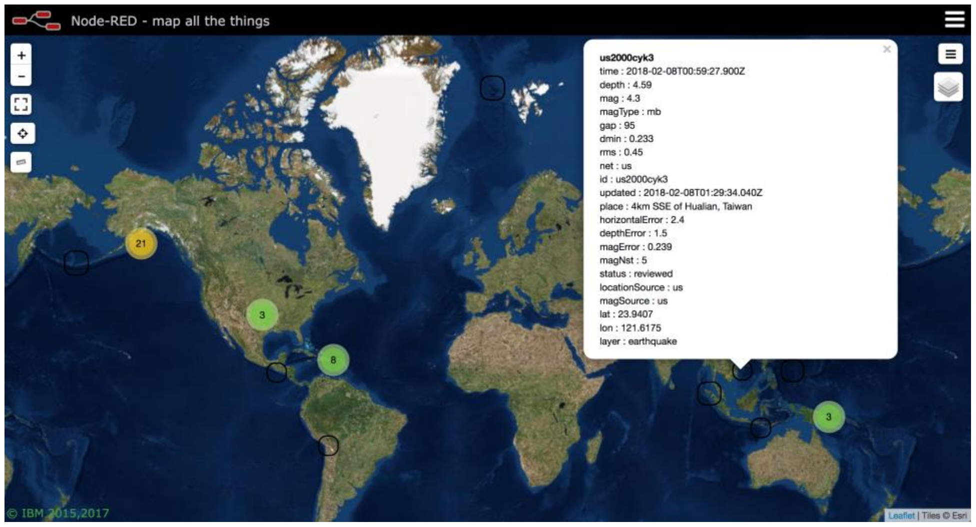
Node-RED is a visual tool for wiring the Internet of Things. It can also be used for other types of applications to quickly assemble flows of services. Node-RED is available as open source and has been implemented by the IBM Emerging Technology organization. Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime in a single-click. While Node-Red is based on Node.js, JavaScript functions can be created within the editor using a rich text editor. A built-in library allows you to save useful functions, templates or flows for re-use.

You can also deploy it as a stand-alone Node.js application. Node-RED is not just used for IoT applications, but it is a generic event-processing engine. For example, you can use it to listen to events from http, web sockets, TCP, Twitter and more and store this data in databases without having to program much if at all. You can also use it for example to implement simple REST APIs.

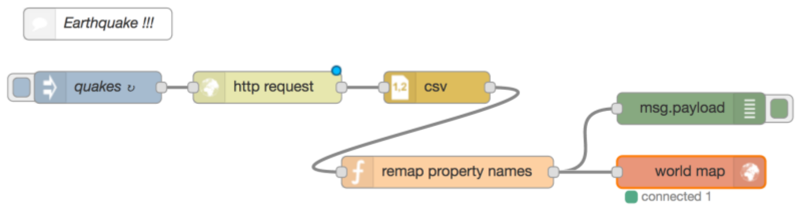
About your application

In this exercise, we will show how to generate an ESRI global map depicting active Earthquake location, in real time from csv data gathered from USGS government resources. The flow will present a separate tab in your browser by pressing `ctrl-shift-m`.

Use the flow below as a guideline as you drag and drop nodes onto the canvas. Connect the nodes at the end and always refer to the flow depicted below for logical placement of the nodes onto the canvas. You will be building the app from scratch using Node-RED as depicted in the image below:



```
us2000cyk3
time : 2018-02-08T00:59:27.900Z
depth : 4.59
mag : 4.3
magType : mb
gap : 95
dmin : 0.233
rms : 0.45
net : us
id : us2000cyk3
updated : 2018-02-08T01:29:34.040Z
place : 4km SSE of Hualian, Taiwan
horizontalError : 2.4
depthError : 1.5
magError : 0.239
magNst : 5
status : reviewed
locationSource : us
magSource : us
lat : 23.9407
lon : 121.6175
layer : earthquake
```



Build a Node-RED app that reveals live Earthquakes

Let’s begin by creating a Node-RED boilerplate app in IBM Cloud. This lab assumes that you have an IBM Cloud account and that you can sign in.

At the core of Node-RED is Node.js, which is a JavaScript runtime that boasts the largest ecosystem of open source components

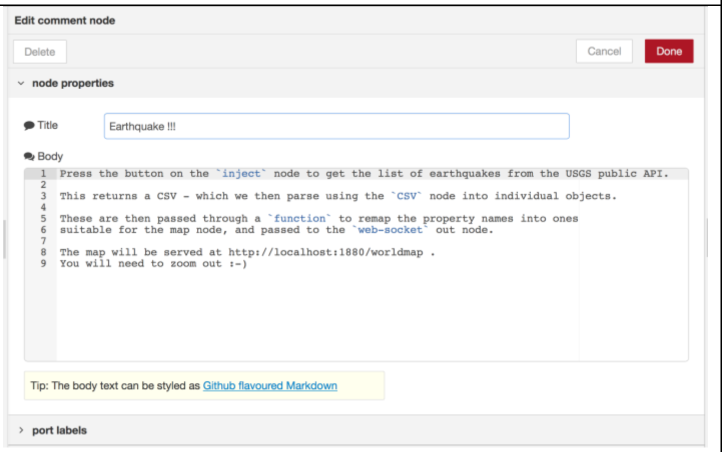
Installing on Mac	Installing on PC

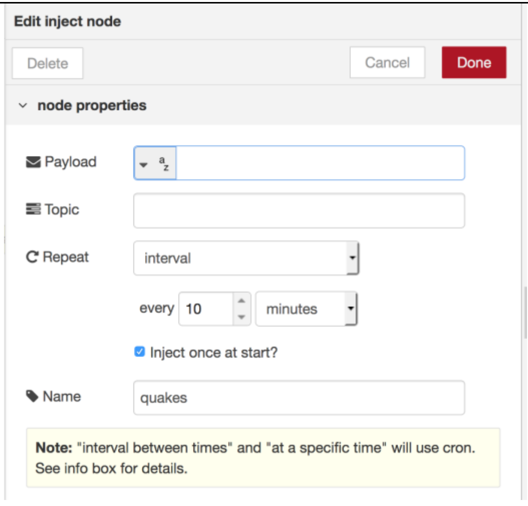
1. Download and Install npm: https://nodejs.org/en/#home-downloadhead
 	
2. Accept the defaults when installing.
 	
3. Open a terminal on your Mac.
 	
4. Enter the following command to install node-RED:
 	
`sudo npm install -g --unsafe-perm node-red`
 	
5. You are now ready to invoke Node-RED. At the terminal prompt, type:
`$ node-red`
 	
6. Enter the following URL (or per returned command) http://127.0.0.1:1880/
	1. Download the latest 10.x LTS version of Node.js from the official Node.js home page.
	
	2. Run the downloaded MSI file. Installing Node.js requires local administrator rights; if you are not a local administrator, you will be prompted for an administrator password on install.
	
	3. Accept the defaults when installing.
	
	4. After installation completes, close any open command prompts and re-open to ensure new environment variables are picked up.
	
	5. Install Node-RED:
	
	`npm install -g --unsafe-perm node-red`
	
	6. At the Command prompt Type:
	`node-red`
	
	7. Enter the following URL (or per returned command) http://127.0.0.1:1880/

Populate the Node-RED canvas

The remaining steps that pertain to building your node-RED canvas are outlined in the table below.

Steps	Example screen capture

1. Drag and drop a **Comment** node. Keep an eye on the completed flow in the previous page as you add nodes. It can help you identify the nodes easily and roughly where to place them on the canvas.
2. For Title specify: Earthquake
3. In the Body copy/paste this:
`Press the button on the `inject` node to get the list of earthquakes from the USGS public API.`
`This returns a CSV - which we then parse using the `CSV` node into individual objects.`
`These are then passed through a `function` to remap the property names into ones suitable for the map node, and passed to the `web-socket` out node.`
`The map will be served at http://localhost:1880/worldmap`
`You will need to zoom out :-)`
	
	Edit comment node
	Delete · Cancel · Done
	node properties
	Title: Earthquake !!!
	Body:
	Press the button on the \`inject\` node to get the list of earthquakes from the USGS public API.
	
	This returns a CSV - which we then parse using the \`CSV\` node into individual objects.
	
	These are then passed through a \`function\` to remap the property names into ones
	suitable for the map node, and passed to the \`web-socket\` out node.
	
	The map will be served at http://localhost:1880/worldmap .
	You will need to zoom out :-)
	Tip: The body text can be styled as Github flavoured Markdown
	port labels

4. Drag and drop an **inject** node. 5. For the Repeat field, select **interval**. 6. Set the interval to every **10 minutes**. 7. Check the Inject once at Start? check box. 8. Click **Done**.	 **Edit inject node** Delete · Cancel · Done node properties Payload: Topic: Repeat: interval every 10 minutes ✓ Inject once at start? Name: quakes **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.

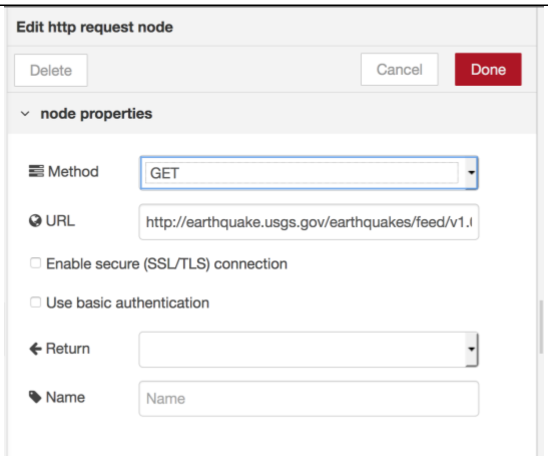
9. Drag and drop a **http request** node.
10. For Method, select **GET** (default) method.
11. For URL copy/paste this path (one line)

```
http://earthquake.usgs.gov/earthquakes/feed/v1.0/summa
ry/2.5_day.csv
```

12. Click **Done**.

You can obtain other data sets from this website:

https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php



Edit http request node

Method: GET

URL: http://earthquake.usgs.gov/earthquakes/feed/v1.(

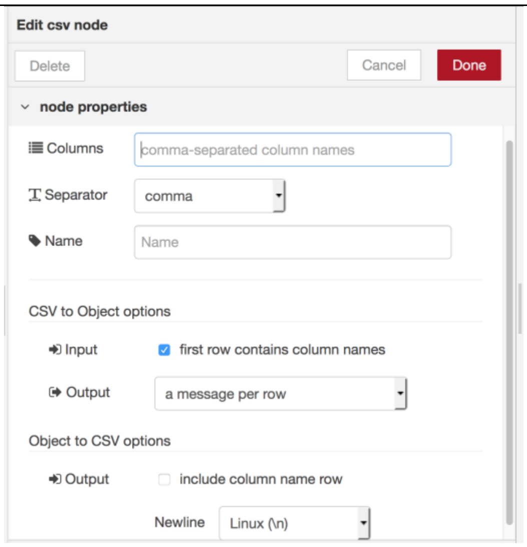
☐ Enable secure (SSL/TLS) connection

☐ Use basic authentication

Return:

Name: Name

13. Drag and drop the **csv** node.
14. For the Input field check the **first row contains names.**
15. Leave all else per default and click **Done**.



Edit csv node

Columns: comma-separated column names

Separator: comma

Name: Name

CSV to Object options

Input: ☑ first row contains column names

Output: a message per row

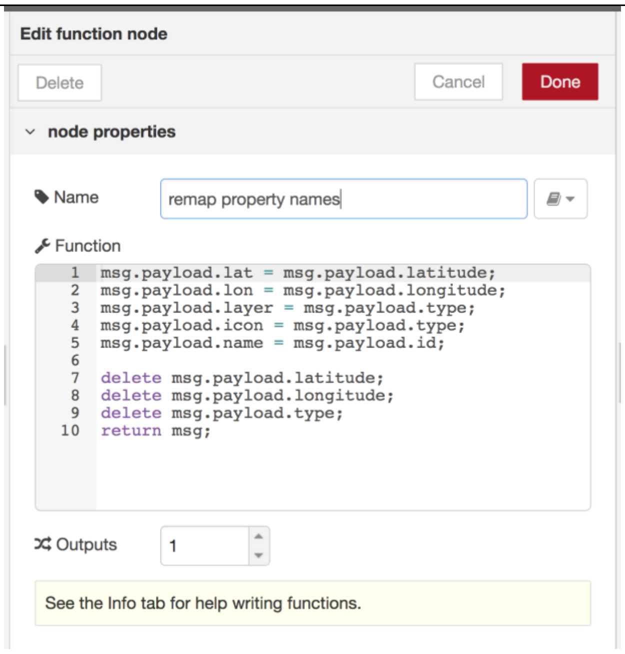
Object to CSV options

Output: ☐ include column name row

Newline: Linux (\n)

16. Drag and Drop a **function** node to extract the query parameter from the web request and assign it to the payload to be provided as input to the Visual Recognition node.
17. Specify a name, for example: **remap property names**.
18. Copy and paste the following *replacing* everything that you see in the box:

```
msg.payload.lat = msg.payload.latitude;

msg.payload.lon = msg.payload.longitude;

msg.payload.layer = msg.payload.type;

msg.payload.icon = msg.payload.type;

msg.payload.name = msg.payload.id;


delete msg.payload.latitude;

delete msg.payload.longitude;

delete msg.payload.type;

return msg;
```



Edit function node

Name: remap property names

Function

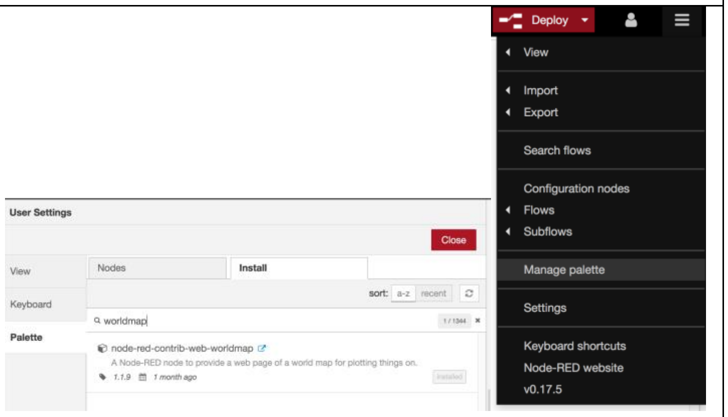
```
msg.payload.lat = msg.payload.latitude;
msg.payload.lon = msg.payload.longitude;
msg.payload.layer = msg.payload.type;
msg.payload.icon = msg.payload.type;
msg.payload.name = msg.payload.id;

delete msg.payload.latitude;
delete msg.payload.longitude;
delete msg.payload.type;
return msg;
```

Outputs: 1

See the Info tab for help writing functions.

Before you proceed any further, you need to install the worldmap node. Not something that appears by default. This is a good opportunity to explore the numerous open-sourced nodes that the community has developed for Node-RED. Continue with the steps below to install the worldmap node.

19. From the hamburger icon to the top left select **Manage palate**.
20. Click the **Install** tab.
21. Search for **worldmap**.
22. Install the node.
23. Click **Close**.



User Settings

Close

View | Keyboard | Palette

Nodes | Install

sort: a-z recent

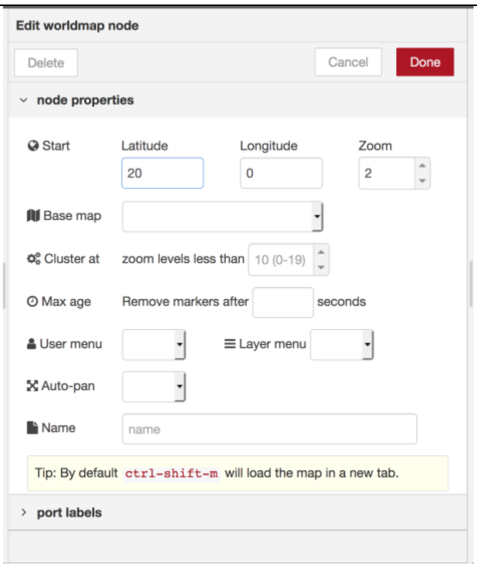
worldmap 1 / 1344

node-red-contrib-web-worldmap

A Node-RED node to provide a web page of a world map for plotting things on.

1.1.9 1 month ago installed

24. Drag and drop a **worldmap** node (not the worldmap in).
25. For Latitude select **20**.
26. For Longitude select **0**. These values are just starting points or place holders.
27. For Zoom, select **2**.
28. Leave all else empty (default) and click **Done**.



Edit worldmap node

Delete Cancel Done

node properties

Start — Latitude: 20 — Longitude: 0 — Zoom: 2

Base map

Cluster at: zoom levels less than 10 (0-19)

Max age: Remove markers after seconds

User menu — Layer menu

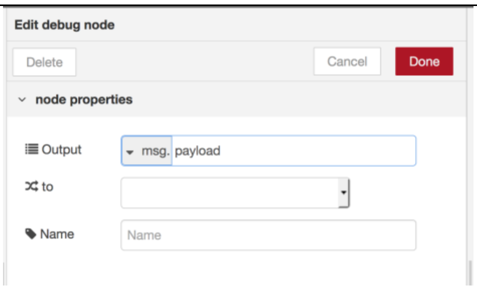
Auto-pan

Name: name

Tip: By default `ctrl-shift-m` will load the map in a new tab.

port labels

29. Drag and drop a **debug** node. No need to make any changes here. Incidentally, you can add debug nodes to any of your nodes and activate and deactivate them as you try to debug your more complex flows. This is akin to inserting a stepper in your code to see just which function calls what method and so forth.
30. Click **Done**.



Edit debug node

Delete Cancel Done

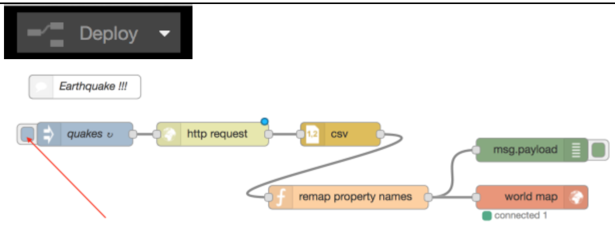
node properties

Output: msg. payload

to

Name: Name

31. Click **Deploy**.
32. Press `ctrl-shift-m`. This is the default key stroke to invoke another tab that has the map of the world.
33. Click the button to the left of the inject node to activate the inject node. Have fun with the map in the adjacent tab in your browser; try the ESRI view!


Deploy
Earthquake !!!
quakes
http request
csv
remap property names
msg.payload
world map
connected 1