

Seminario 4: Node-RED

Sistemas Distribuidos

Gabriel Guerrero Contreras

Departamento de Ingeniería Informática
Universidad de Cádiz



Indice

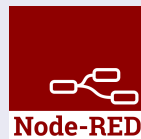
- 1 Introducción
- 2 Conceptos Básicos
- 3 Creando Flujos
- 4 Sockets y HTTP
- 5 Extensiones
- 6 Bibliografía

Sección 1 | Introducción

Introducción

Node-RED

- Entorno gráfico de desarrollo para construir aplicaciones Javascript
- Funciona sobre Node.js
- Open Source y diseñado por IBM
- Node-RED 1.0 fue publicado en 2019
- Su principal objetivo es simplificar el desarrollo de aplicaciones orientadas al manejo de **eventos asíncronos**



Instalación

Instalación en Linux:

- 1 Descargamos la información de todas las fuentes configuradas:

```
$ sudo apt-get update
```

- 2 Instalamos el sistema de gestión de paquetes por defecto para Node.js:

```
$ sudo apt-get install npm
```

- 3 Instalamos Node-RED:

```
$ sudo npm install -g -unsafe-perm node-red
```

La opción `-g` añade el comando `node-red` al PATH

- 4 Comprobamos Node-RED se ha instalado correctamente:

```
$ node-red -help
```

Nos debe aparecer la ayuda del comando `node-red` junto a la versión

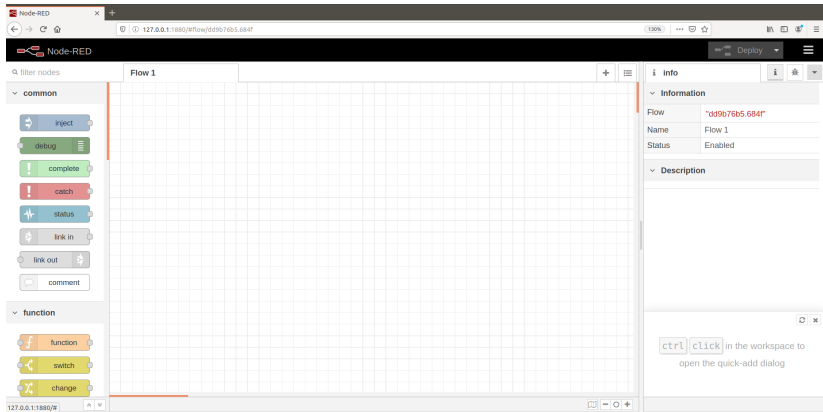
Instalación en Windows: <https://nodered.org/docs/getting-started/windows>

Lanzando Node-RED Localmente

- 1 Desde la terminal ejecutamos el comando:
`$ node-red`
- 2 Entre la información mostrada debe aparecer la dirección en la que está corriendo:
`[info] Server now running at http://127.0.0.1:1880/`
- 3 Accedemos a la dirección indicada a través del navegador web.

Lanzando Node-RED Localmente

Nos aparecerá la Interfaz de Usuario de Administración de Node-RED



Sección 2 | Conceptos Básicos

Conceptos Básicos

Node

- Unidad básico de construcción
- Bloque de software que procesa **mensajes**
- Puede tener **entradas** y **salidas** que permiten el paso de mensajes entre nodos
- Una entrada puede aceptar conexiones de múltiples nodos y una salida puede, también, dar salida a múltiples nodos

Conceptos Básicos

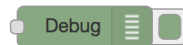
Inject Node

- Nodo de salida
- Producen mensajes sin necesidad de entrada



Debug Node

- Nodo de entrada
- Se utiliza para mostrar mensajes de depuración en el editor



Conceptos Básicos

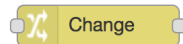
Function Node

- Nodo de entrada/salida
- Permite ejecutar código **JavaScript** sobre los mensajes que pasan por él



Change Node

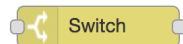
- Nodo de entrada/salida
- Puede utilizarse para modificar las propiedades de un mensaje y establecer propiedades de contexto sin tener que recurrir a un **Function Node**



Conceptos Básicos

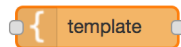
Switch Node

- Nodo de entrada/salida
- Permite enrutar los mensajes a diferentes ramas evaluando un conjunto de reglas en cada mensaje



Template Node

- Nodo de entrada/salida
- Usado para generar texto mediante una plantilla usando las propiedades de un mensaje
- Utiliza el lenguaje de plantillas Mustache

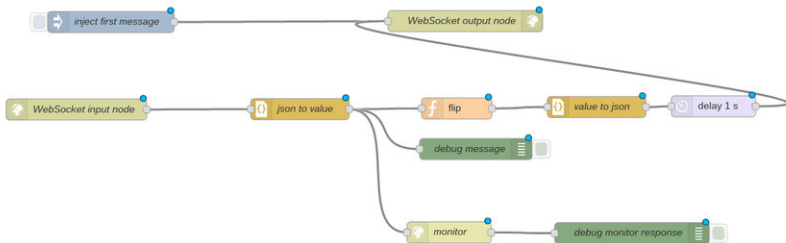


Mustache: <https://mustache.github.io/mustache.5.html>

Conceptos Básicos

Flow

- Una colección de nodos **conectados** entre sí y funcionando como una aplicación o programa
- Cada flujo puede tener un nombre y una descripción que se muestra en la barra lateral de información
- Todos los nodos de un flujo pueden acceder al mismo **contexto** de flujo



Conceptos Básicos

Message

- Es la información que se pasa entre los nodos de un flujo
- Son simples **JavaScript Objects** que pueden tener cualquier conjunto de propiedades (**payload**)
- En el editor aparecerán como **msg**
- El valor de una propiedad puede ser cualquier tipo de JavaScript válido, como:

Boolean (**true**, **false**), Number (**5**, **23.56**), String (**“Hello”**), Array (**[1,2,3,4]**), Object (**{“a”:1, “b”:2}**) o Null

- Cuidado, se usa el formato JSON, JSON \neq Javascript Object

Wire

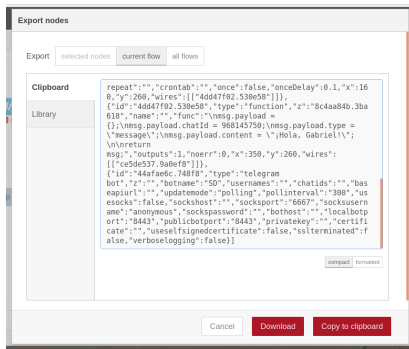
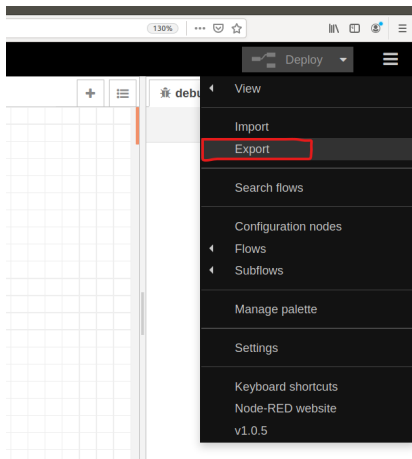
- Conectan los nodos y representan cómo los mensajes pasan a través del flujo

Conceptos Básicos

Context

- Utilizado para almacenar información que puede ser compartida entre diferentes nodos sin usar los mensajes que pasan a través de un flujo
- Alcance:
 - **Node**: solo visible para el nodo que almacena el valor
 - **Flow**: visible para todos los nodos que comparten el mismo flujo
 - **Global**: visible a todos los nodos
- Por defecto, el contexto se almacena solo en memoria, es decir, su contenido se borra cada vez que Node-RED se reinicia

Exportar Proyecto



Sección 3 | Creando Flujos

Hola Mundo

The screenshot displays the Node-RED web interface in a browser window. The address bar shows the URL `127.0.0.1:1880/flows/dd9b76b5.6b4f`. The interface is titled "Node-RED" and includes a "Deploy" button in the top right corner.

The main workspace, labeled "Flow 1", contains a single flow with two nodes connected by a wire:

- An **inject** node (blue) with the text "Hola Mundo" and a dropdown menu.
- A **msg.payload** node (green) with a dropdown menu.

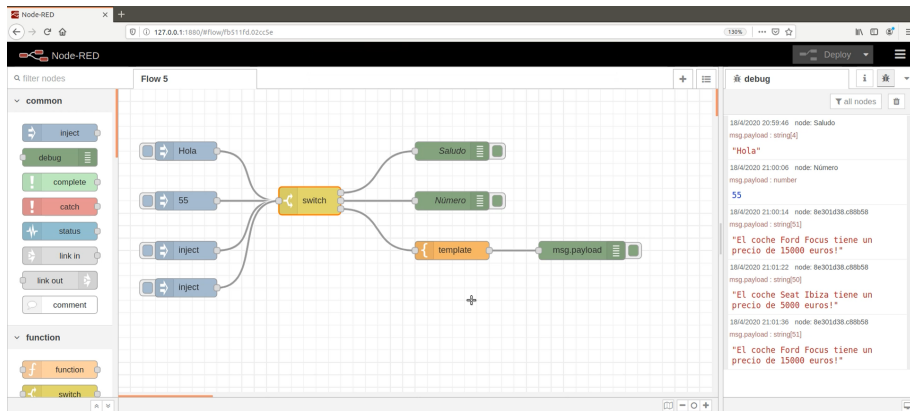
The left sidebar shows the "filter nodes" search bar and two categories of nodes:

- common**: inject, debug, complete, catch, status, link in, link out, comment.
- function**: function, switch.

The right sidebar shows the **debug** console, which displays a log of the "Hola Mundo" message being sent. The log entries show the timestamp, node ID, and the message payload:

```
18/4/2020 20:17:18 node: 8232033e.b9aca  
msg.payload : string[10]  
"Hola Mundo"  
18/4/2020 20:17:20 node: 8232033e.b9aca  
msg.payload : string[10]  
"Hola Mundo"  
18/4/2020 20:17:22 node: 8232033e.b9aca  
msg.payload : string[10]  
"Hola Mundo"  
18/4/2020 20:17:24 node: 8232033e.b9aca  
msg.payload : string[10]  
"Hola Mundo"  
18/4/2020 20:17:26 node: 8232033e.b9aca  
msg.payload : string[10]  
"Hola Mundo"  
18/4/2020 20:17:28 node: 8232033e.b9aca  
msg.payload : string[10]  
"Hola Mundo"
```

Switch & Template Nodes



Function Node

The screenshot displays the Node-RED web interface in a browser. The main workspace shows a flow diagram with the following components:

- Flow 6:** Contains an `inject` node, a `debug` node, a `complete` node, a `catch` node, a `status` node, a `link in` node, a `link out` node, and a `comment` node.
- Flow 7:** Contains a `function` node and a `switch` node.
- Flow 8:** Contains a `function` node and a `switch` node.

The flow diagram shows a sequence of nodes: `Dólar` and `Euros` (inject nodes) feed into a `Cambia divisa` (function node). The output of `Cambia divisa` is split into two paths: one leading to `Plantilla Euros` and `Plantilla Dólares` (template nodes), and another leading to a `msg.payload` (output node). The `Plantilla Euros` and `Plantilla Dólares` nodes feed into a `msg.payload` (output node).

The debug console on the right shows the following output:

```
divisa: "euro"
cantidad: 4.5

18/4/2020 22:51:44 node: 61bd3d37.5d2e9c
msg.payload : Object
  divisa: "euro"
  cantidad: 4.5
  cambio: object
    divisa: "dólar"
    cantidad: 4.905

18/4/2020 22:51:44 node: 90a7cb88.5f3d88
msg.payload : string[15]
"4.5$ son 4.905€"

18/4/2020 22:52:33 node: 90a7cb88.5f3d88
msg.payload : string[27]
"4.5€ son 4.1400000000000001$"

18/4/2020 22:52:36 node: 90a7cb88.5f3d88
msg.payload : string[15]
"4.5$ son 4.905€"
```

Context

The screenshot shows the Node-RED web interface in a browser. The main workspace displays a flow with a **timestamp** node connected to **Función 1**, which is then connected to **Función 2**. The left sidebar contains a palette with nodes categorized under **common** and **function**. The right sidebar shows the **debug** console with a log of messages and their payloads.

Edit function node

Delete Cancel Done

Properties

Name: Función 2

Function

```
1 var i = global.get('contador') || 0;
2 i++;
3 msg.payload = "F2: Contador = " + i;
4 global.set('contador',i);
5 return msg;
```

Outputs: 1

Enabled

debug

all nodes

Log entries:

- "F1: Contador = 1"
- 19/4/2020 0:14:54 node: fbacb324.8eac3
msg.payload: string[16]
"F2: Contador = 2"
- 19/4/2020 0:14:58 node: fbacb324.8eac3
msg.payload: string[16]
"F2: Contador = 3"
- 19/4/2020 0:14:58 node: fbacb324.8eac3
msg.payload: string[16]
"F1: Contador = 3"
- 19/4/2020 0:15:01 node: 931f76cc.aa01e8
msg.payload: string[16]
"F3: Contador = 5"
- 19/4/2020 0:15:10 node: fbacb324.8eac3
msg.payload: string[16]
"F1: Contador = 6"
- 19/4/2020 0:15:10 node: fbacb324.8eac3
msg.payload: string[16]
"F2: Contador = 7"

Sección 4 | Sockets y HTTP

UDP

The screenshot displays the Node-RED web interface in a browser. The main workspace shows a flow named "Flow 1" with the following components:

- An **udp 20001** node (grey) that receives incoming UDP packets.
- A **function** node (orange) that processes the incoming data. It has three output connections to **msg** nodes: **msg.port**, **msg.ip**, and **Cliente**.
- An **udp :** node (grey) that sends the processed data out over the network.

The left sidebar shows the "common" and "function" node categories. The right sidebar contains a "debug" console showing the following log entries:

```
19/4/2020 14:02:48 node: Cliente  
msg.payload : buffer[73]  
▼ buffer[73] | string  
Holy cow, Rick! I didn't know hanging out with you was making me smarter!  
  
19/4/2020 14:05:10 node: Cliente  
msg.payload : string[73]  
"Holy cow, Rick! I didn't know hanging out with you was making me smarter!"  
  
19/4/2020 14:05:11 node: ca9dad43.d3d5f8  
msg.ip : string[9]  
"127.0.0.1"  
  
19/4/2020 14:05:11 node: 9d338ff.03f328  
msg.port : number  
51129
```

GET Request

The screenshot displays the Node-RED web interface in a browser. The address bar shows the URL `127.0.0.1:1880/#flow/6229679a.372a56`. The interface includes a sidebar with various node types, a workspace with three flows (Flow 2, Flow 1, and Flow 3), and a debug console on the right.

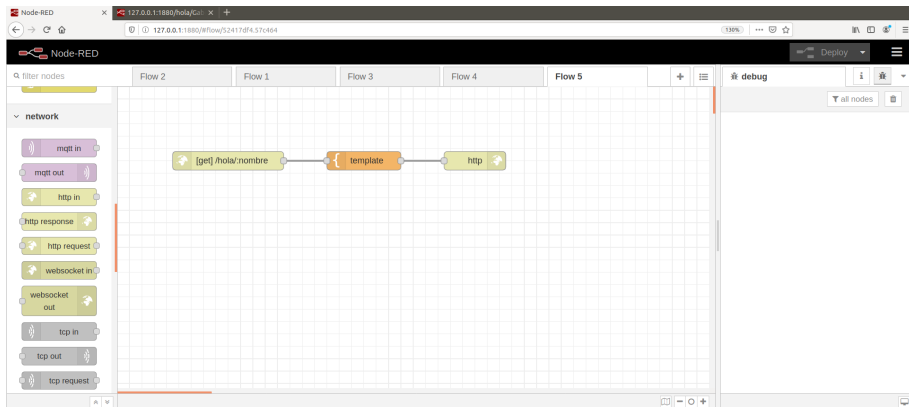
The flow in Flow 3 consists of the following nodes:

- inject** node (blue) connected to an **http request** node (yellow).
- The **http request** node is connected to a **msg.payload** node (green).
- The **msg.payload** node is connected to a **template** node (orange).
- The **template** node is connected to a **Convertir K a °** node (yellow).

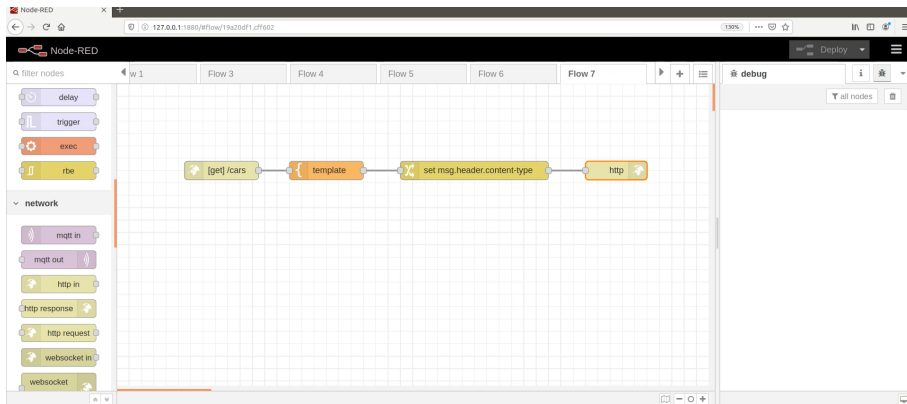
The debug console on the right shows the following messages:

```
19/4/2020 15:30:06 node: de5b08ce.d4d7b6  
msg.payload : Object  
> { coord: object, weather:  
  array[1], base: "stations", main:  
    object, visibility: 10000 - }  
19/4/2020 15:30:06 node: de5b08ce.d4d7b6  
msg.payload : string[32]  
"En Cadiz hay 21.410000000000025"  
19/4/2020 15:30:30 node: de5b08ce.d4d7b6  
msg.payload : Object  
> { coord: object, weather:  
  array[1], base: "stations", main:  
    object, visibility: 10000 - }  
19/4/2020 15:30:30 node: de5b08ce.d4d7b6  
msg.payload : string[34]  
"En Seville hay  
23.3799999999999995"
```


Endpoint GET + HTML Response



Endpoint GET + JSON Response



Endpoint POST + JSON Response

The screenshot displays the Node-RED web interface in a browser. The address bar shows the URL `127.0.0.1:1880/#flow/a85b8515.cb3d28`. The interface includes a left sidebar with a node palette, a central workspace, and a right sidebar with a debug console.

Node Palette:

- filter nodes
- trigger
- exec
- rbe
- network**
- mqtt in
- mqtt out
- http in
- http response
- http request
- websocket in
- websocket out
- tcp in

Flow 10:

```
graph LR; A["[post] /hello"] --> B["f"]; B --> C["http"];
```

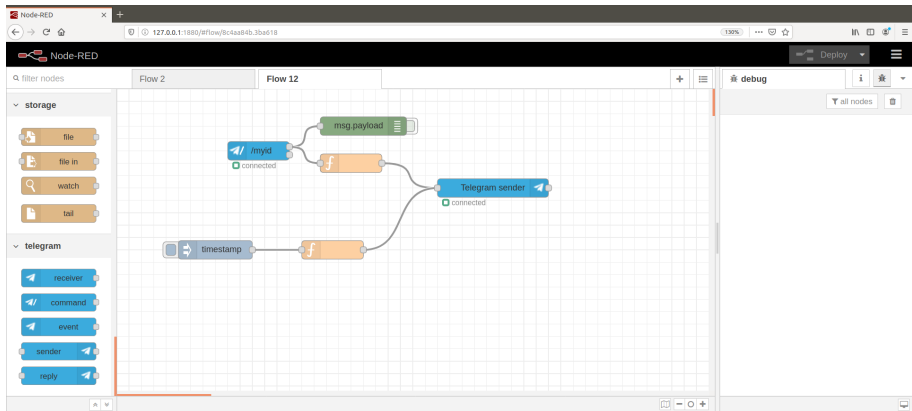
The flow consists of three nodes connected in sequence: a `[post] /hello` node (green), a function node `f` (orange), and an `http` node (green).

Right Sidebar:

- debug console (active)
- all nodes
- Deploy button

Sección 5 | Extensiones

Telegram



Sección 6 | Bibliografía

Bibliografía

- <https://nodered.org/docs/>
- <https://flows.nodered.org/node/node-red-contrib-telegrambot>