Lab 2: Introduction to the Node-Red

Exercise structure

1. Introduction

1.1 Node-Red: Main features

1.2 Node-Red arquitecture

1.3 What can we use Node-Red for? -> TinyGs

1.4 Installation

1.5 Main nodes and basic programming

- Inject-Debug
- Change
- Switch
- Http request

2. Configuration and Security

2.1.HTTPs and Password configuration

2.2. Username/password based authentication

2.3. Setting a default user

3. Node-Red Dashboard

3.1 Real time dashboard

3.2 Advanced dashboard

4. Report requirements

1.Introduction

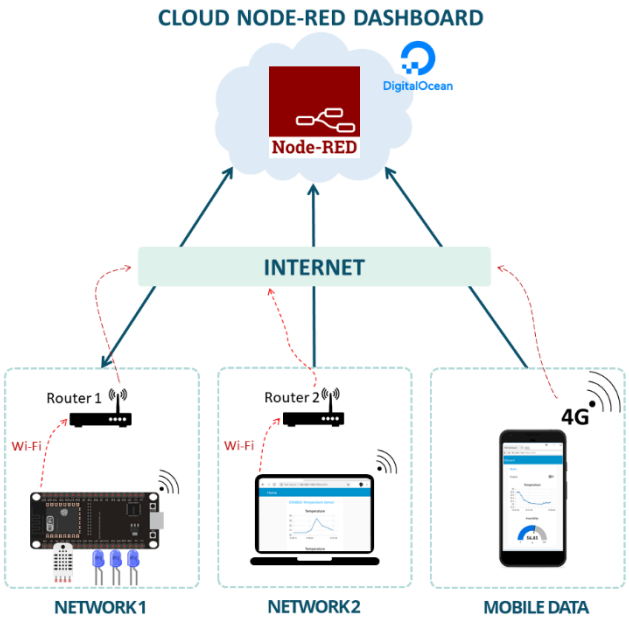
1.1 Node-Red: Main features

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. Node-Red allows the user to visualize relationships and functions by coding very few lines of code. Node-RED is a browser-based flow editor where you can add or remove nodes and connect them to communicate with each other. The idea of Node-RED is to be 'low-code'.

Node-RED

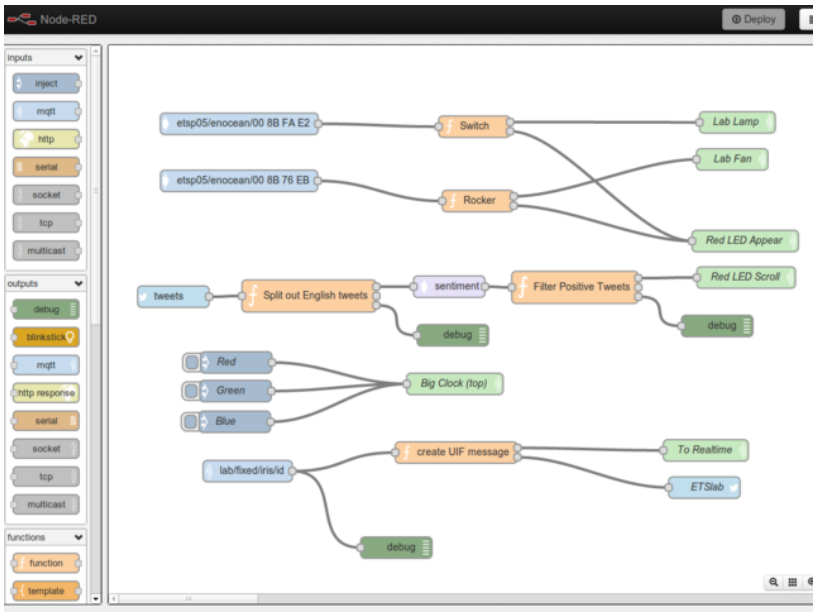
Node-Red makes connecting hardware devices, APIs, and online services easier than ever.

Node-red has become the open-source standard for the management and processing of data in real time, managing to simplify the processes between producers and consumers of information. Its strong point is simplicity. It allows us to use complex technologies without drilling down to the smallest detail in all of them. We stay in an initial layer where we focus on essentials and put aside what is not practical.



Node-RED is built from NodeJS and the D3.js JavaScript library. NodeJS provides enough power to make Node-RED reliable and scalable. NodeJS is very powerful software that enables server-side JavaScript programming.

The minimum structure is the nodes. These are dragged through the graphical interface and allow us to do a specific task. Receive an HTTP call, MQTT message, or push-button activation. All these nodes are organized into flows or flows that group nodes that connect to each other. All in a visual way, without hardly having to program.



Node-RED is a flow engine with an IoT approach, which allows to graphically define service flows through standard protocols such as REST, MQTT, WebSocket, AMQP, and offering integration with third-party APIs, such as Twitter, Facebook, etc.

One of the most notable characteristics of Node-RED is the simplicity with which new nodes can be created and installed, in the following link we have all the necessary documentation:

- http://nodered.org/docs/creating-nodes/
- Node-RED Library: https://flows.nodered.org

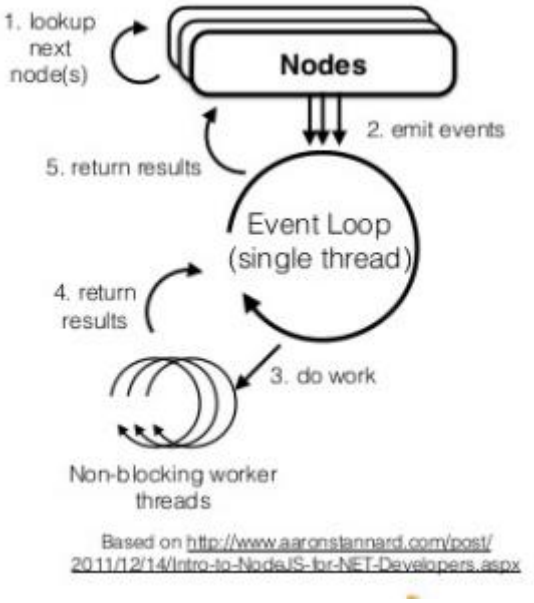
Flows programmed in Node-RED are stored internally in JSON format and are portable between different Node-RED installations, as long as the destination Node-RED has the nodes used in the flow installed.

Thus, a Node-RED flow consists of a file that looks like this:

```
[
    {"id":"3c95cae6.8db046","type":"tab","label":"Flow 1"},
    {"id":"c15bbf51.850d6","type":"tab","label":"Flow 3"},
    {"id":"e74c826.0407f8","type":"ssap-process-request","z":"3c95cae6.8db046","name":"","direccion":"","tipomensaje":"","ontology":"","kp":"","instanciakp"
    {"id":"3aed593c.96d696","type":"debug","z":"3c95cae6.8db046","name":"","active":true,"console":"false","complete":"payload","x":401,"y":169,"wires":[]},
    {"id":"2bc04eaa.34fc92","type":"inject","z":"c15bbf51.850d6","name":"","topic":"","payload":"","payloadType":"date","repeat":"","crontab":"","once":fals
    {"id":"6d4232b6.c9ee5c","type":"debug","z":"c15bbf51.850d6","name":"","active":true,"console":"false","complete":"false","x":457,"y":164,"wires":[]}
]
```

1.2 Node-Red architecture

- Single flow stored per instance in the file system
- On deploy, JSON flow is parsed, nodes instantiated and wired up directly.
- Each node has a connection to downstream nodes. Node.js EventEmitter API to manage listeners.
- By calling EventEmitter API, rather than downstream node directly, uses Node.js' single threaded Event Loop.
- Worker threads are created for long running async operations.
- All tabs are part of a single flow.

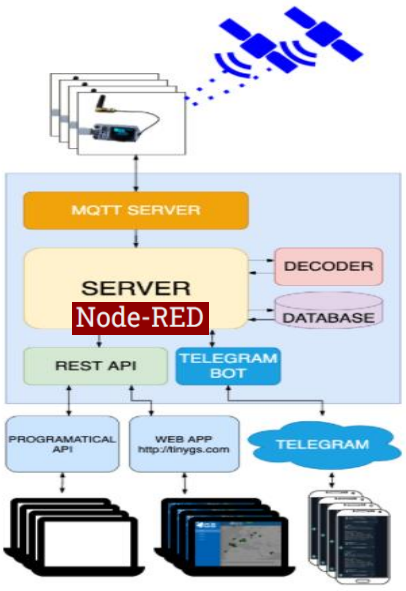


1.3 What can we use Node-Red for?

An example: TinyGs

TinyGS is an open network of Ground Stations distributed around the world to receive and operate LoRa satellites, weather probes and other flying objects, using cheap and versatile modules.

TinyGS web: https://tinygs.com/
GitHub: https://github.com/G4lile0/tinyGS



1.4 Installation

Node-RED can be installed in a multitude of Operating Systems and Hardware, the main requirement is that Node.js can be installed: https://nodejs.org/

Supported versions of node.js: https://nodered.org/docs/faq/node-versions

You can find installation/"getting started" guide for different devices:

- Node-RED installation: https://nodered.org/docs/getting-started/
- Docker: https://nodered.org/docs/getting-started/docker
- Local**:** https://nodered.org/docs/getting-started/local
- Windows: https://nodered.org/docs/getting-started/windows
- Raspberry Pi: https://nodered.org/docs/getting-started/raspberrypi
- AWS: https://nodered.org/docs/getting-started/aws
- Microsoft Azure: https://nodered.org/docs/getting-started/azure

1.5 Main nodes and basic programming

Inject

The Inject node can be used to manual trigger a flow by clicking the node's button within the editor. It can also be used to automatically trigger flows at regular intervals.

The message sent by the Inject node can have its payload and topic properties set.

The payload can be set to a variety of different types:

- A flow or global context property value
- A String, number, boolean, Buffer or Object
- A Timestamp in milliseconds since January 1st, 1970

Debug

The Debug node can be used to display messages in the Debug sidebar within the editor.

The sidebar provides a structured view of the messages it is sent, making it easier to explore the message.

Alongside each message, the debug sidebar includes information about the time the message was received and which Debug node sent it. Clicking on the source node id will reveal that node within the workspace.

The button on the node can be used to enable or disable its output. It is recommended to disable or remove any Debug nodes that are not being used.

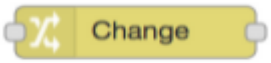
The node can also be configured to send all messages to the runtime log, or to send short (32 characters) to the status text under the debug node.

Exercise 1

Create a node-red flow that prints “Hello World” in the debug sidebar**.**

Change

The Change node can be used to modify a message’s properties and set context properties without having to resort to a Function node.



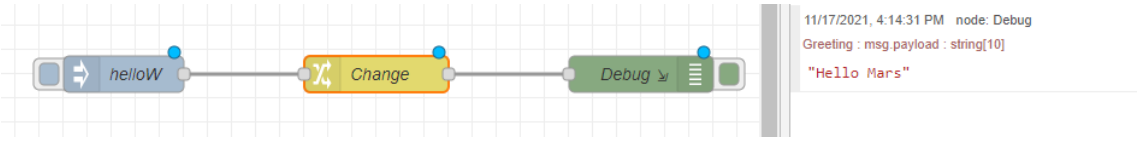
Each node can be configured with multiple operations that are applied in order. The available operations are:

- Set - set a property. The value can be a variety of different types, or can be taken from an existing message or context property.
- Change - search and replace parts of a message property.
- Move - move or rename a property.
- Delete - delete a property.

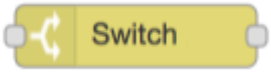
When setting a property, the value can also be the result of a JSONata expression. JSONata is a declarative query and transformation language for JSON data.

Exercise 2

Modify previous node-red flow, using a change node to replace “World” by “Mars”.



Switch



The Switch node allows messages to be routed to different branches of a flow by evaluating a set of rules against each message.

The node is configured with the property to test - which can be either a message property or a context property.

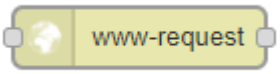
There are four types of rule:

- Value rules are evaluated against the configured property

- Sequence rules can be used on message sequences, such as those generated by the Split node
- A JSONata Expression can be provided that will be evaluated against the whole message and will match if the expression returns a true value.
- An Otherwise rule can be used to match if none of the preceding rules have matched.

The node will route a message to all outputs corresponding to matching rules. But it can also be configured to stop evaluating rules when it finds one that matches.

Http request

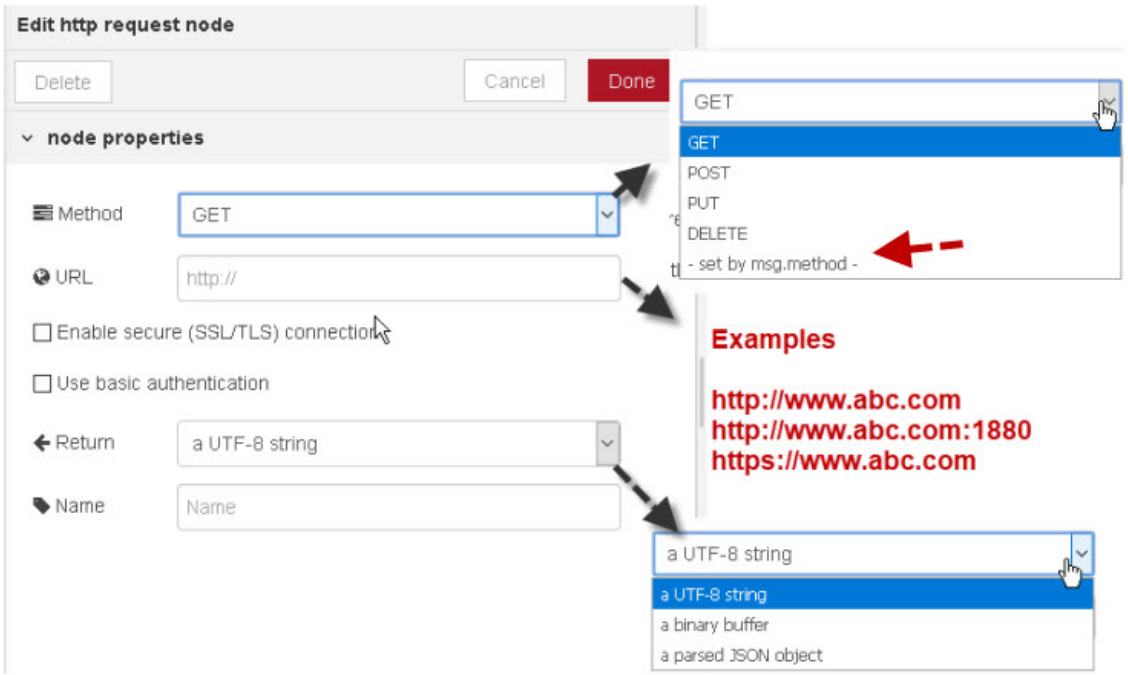


The http request node can be used for:

- Retrieving web pages from a website
- Making API Request
- Sending and receiving JSON data to a website or API.

The node will send a request and receive the response.

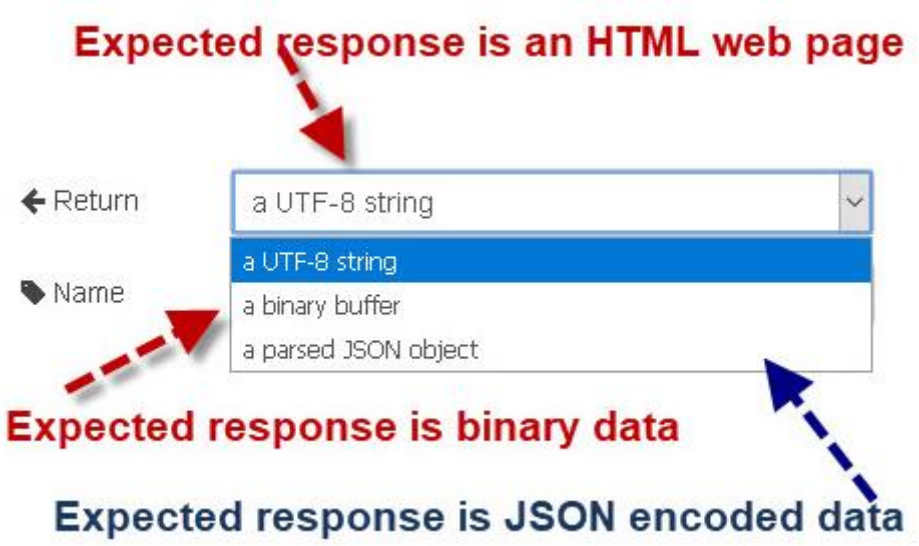
The node handles both the request and response.



The settings include:

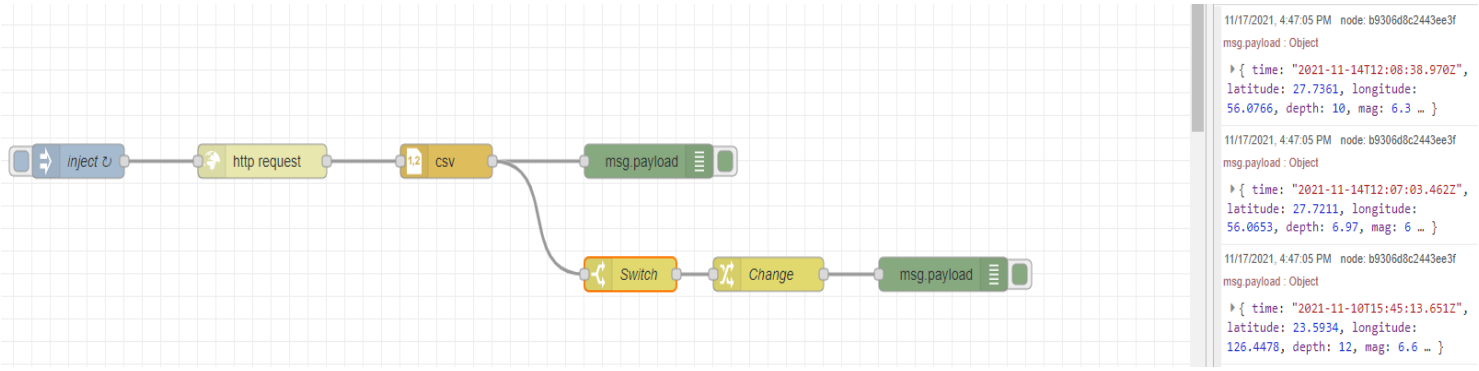
- The request method -The node supports the GET,POST,PUT and DELETE methods.
- The main method is GET which is used for getting a web page.
- The URL of the resource.
- Authentication -If the website requires authentication then basic authentication can be done here.

- SSL– A check box for forcing the connection to use SSL
- Response:



Exercise 3

Create a new Flow using the following nodes:



1. Add an Inject Node: In this case, the Inject node will be configured to trigger the flow at a regular interval. Set the repeat interval to every 5 minutes.
2. Add an HTTP Request node and set the URL property to:

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

   This URL is a feed of significant earthquakes in the last month from the US Geological Survey web site (https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php).

3. Add a CSV node and edit its properties: Enable option for 'First row contains column names'.
4. Add a Debug node to the output.
5. Add a Switch node to the workspace: Edit its properties and configure it to check the property `msg.payload.mag` with a test of `>=` change it to test on a number and the value (check different values!).
6. Add a second wire from the CSV node to this Switch node.
7. Add a Change node, wired to the output of the Switch node. Configure it to set `msg.payload` to the string `PANIC!`.
8. Wire a new Debug node to the output of the Change node

2. Configuration and Security

The Node-RED editor is not protected by default: anyone who can access your IP address can access the editor and implement changes.

2.1. User and HTTPs configuration

To enable access to the Node-RED Editor over HTTPS, rather than the default HTTP, you can use the https configuration option in your settings file. (Check here where to find it: https://nodered.org/docs/user-guide/runtime/settings-file).

The https option can be either a static set of configuration options, or, since Node-RED 1.1.0, a function that returns the options. The full set of options are documented here: https://nodejs.org/api/tls.html#tls_tls_createsecurecontext_options.

As a minimum, the options should include:

- key - Private key in PEM format, provided as a String or Buffer
- cert - Cert chain in PEM format, provided as a String or Buffer

For a guide on how to generate certificates, you can follow: https://it.knightnet.org.uk/kb/nr-qa/https-valid-certificates/

The default Node-RED settings file includes a **https section** that can be used to load the certificates from local files.

```
https: {
    key: require("fs").readFileSync('privkey.pem'),
    cert: require("fs").readFileSync('cert.pem')
},
```

If the https property is a function, it can be used to return the options object. The function can optionally return a Promise that will resolve to the options object, allowing it to complete asynchronously.

```
https: function() {
    return new Promise((resolve, reject) => {
        var key, cert;
        // Do some work to obtain valid certificates
        // ...
        resolve({
            key: key
            cert: ccert
        })
    });
}
```

2.2. Username/password based authentication

To enable user authentication on the Editor and Admin API, uncomment the adminAuth property in your settings file:

```
adminAuth: {
```

```
    type: "credentials",
    users: [
        {
            username: "admin",
            password:
"$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.",
            permissions: "*"
        },
        {
            username: "george",
            password:
"$2b$08$wuAqPiKJlVN27eF5qJp.RuQYuy6ZYONW7a/UWYxDTtwKFCdB8F19y",
            permissions: "read"
        }
    ]
}
```

The users property is an array of user objects. This allows you to define multiple users, each of whom can have different permissions.

This example configuration above defines two users. One called admin who has permission to do everything within the editor and has a password of password. The other called george who is given read-only access.

Note that the passwords are securely hashed using the bcrypt algorithm. If you are using Node-RED 1.1.0 or later, you can use the command:

```
node-red admin hash-pw
```

2.3. Setting a default user

The example configuration above will prevent anyone from accessing the editor unless they log in.

In some cases, it is desirable to allow everyone some level of access. Typically, this will be giving read-only access to the editor. To do this, the default property can be added to the adminAuth setting to define the default user:

```
adminAuth: {
    type: "credentials",
    users: [ /* list of users */ ],
    default: {
        permissions: "read"
    }
}
```

For further information or configuration related to older versions check the full configuration path: Securing Node-RED : Node-RED (nodered.org)

3. Node-Red Dashboard

The objective of this series of exercises is to explore part of the potential of Node-Red. For this, we are going to work with real data obtained online and dashboards to study and visualize the data collected.

In order to carry out the practice, several additional nodes must be downloaded. To download new nodes, follow:

≡ > *Manage palette* > *install* > *node-name*

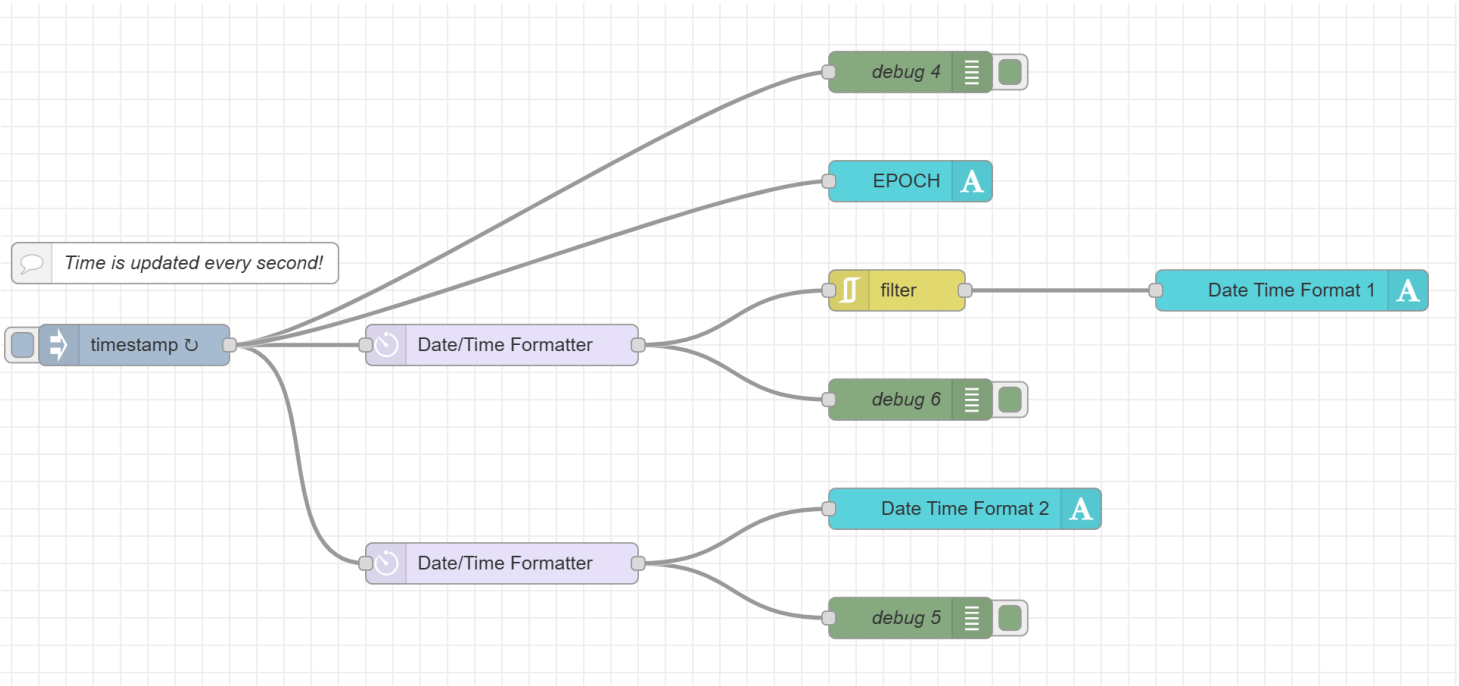
3.1 Real time dashboard

In this first exercise, we are going to work on creating a simple dashboard to display the time and date in 3 different formats.

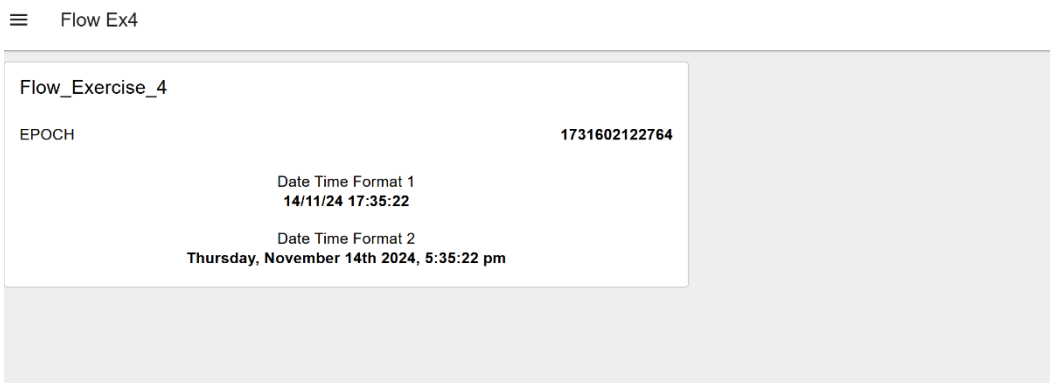
Exercise 4

Build a dashboard in Node-Red where the current date and time is displayed. The dashboard should display epoch, the date and time in different formats (see screenshot with result visualisation). The timestamp (epoch time – millisecs since January 1st, 1970) should be updated every second.

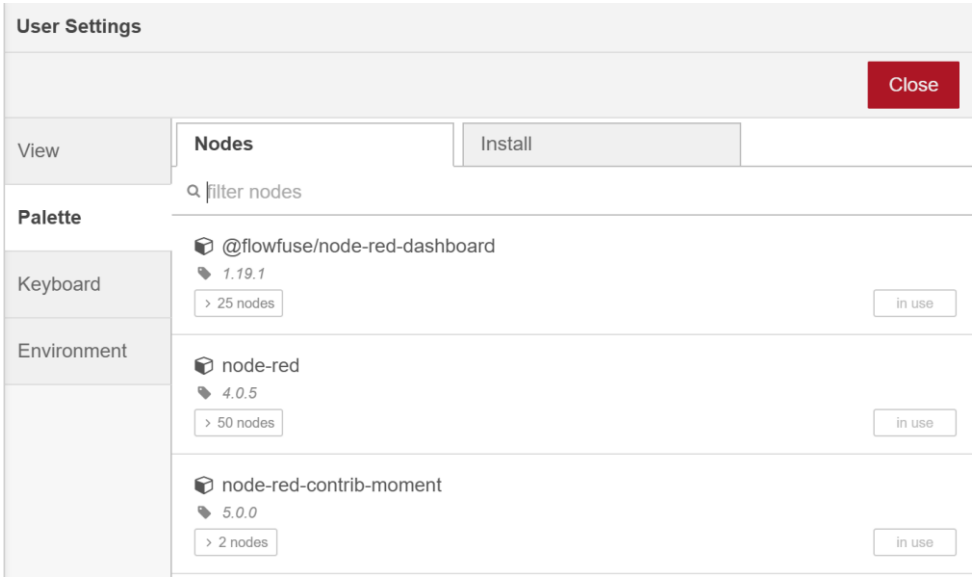
The resulting flow should look like this:



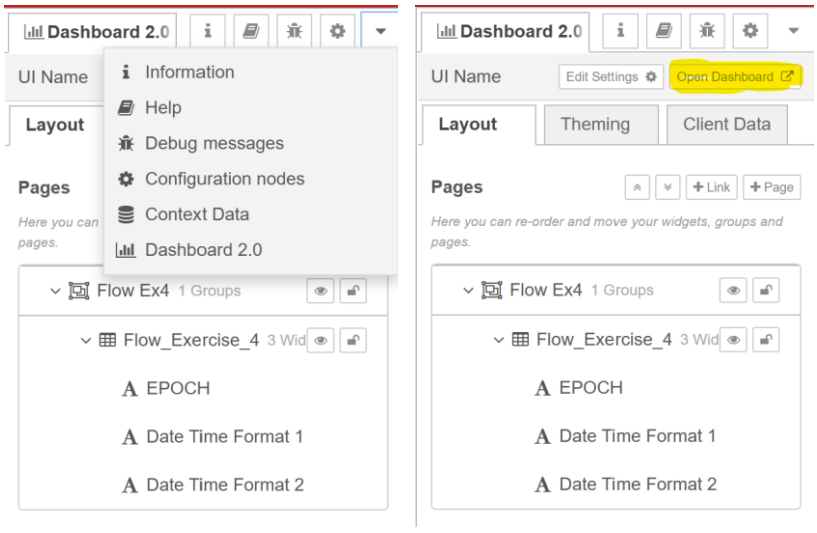
The final result on the dashboard should look something like this:



You will need the following Nodes:



Once the flow is configured and implemented, we need to visualize the dashboard. To do this, follow the steps that appear in the following images.



3.2 Advanced dashboard

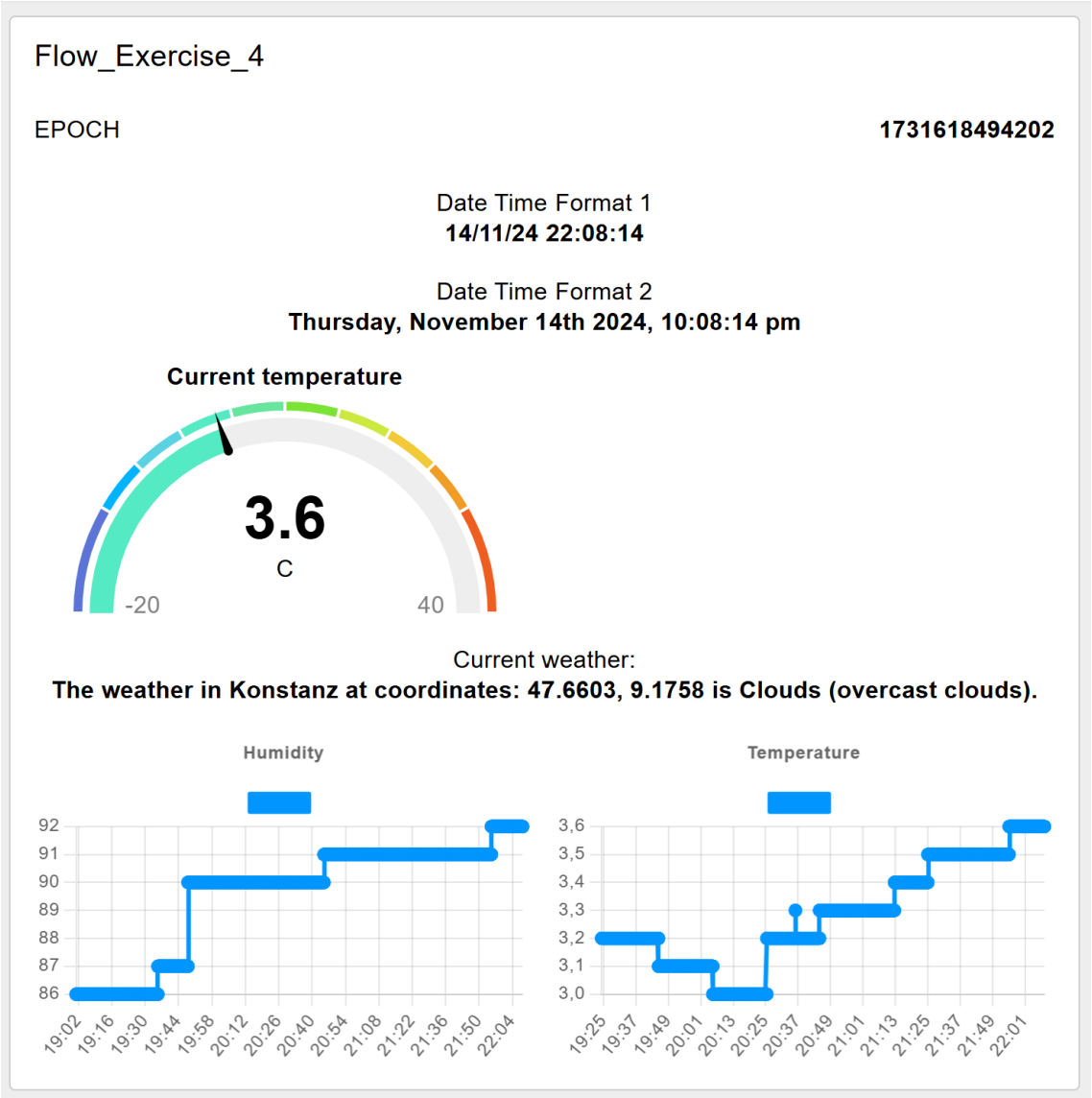
The aim of this exercise is to work with information obtained from online services and to visualise it in different ways.

Exercise 5

Add current weather information to your dashboard:

- **Current temperature as a gauge (incl. several segments)**
- **Description of the current weather**
- **Chart visualising the changes in humidity**
- **Chart visualising the changes in temperature (both charts should be visualized in the same line as on the example visualisation).**

Your dashboard should be similar to this visualisation:



4. Report requirements

Lab 2 should be carried out individually by each student. For this reason, the entire report is to be written by each person on an individual basis. On the day of delivery, please sit in groups and each group member will explain 1 or 2 exercises on their own laptop.

Reports should include a short description of each exercise and following screenshots (for each of 5 exercises!):

- visualisation of the node structure/flow (including debug sidebar)
- configuration of relevant nodes (no more than 2 nodes for exercises 1-3, no more than 4 nodes for exercises 4, 5). Do not repeat the same nodes in screenshots
- dashboard (if applicable).