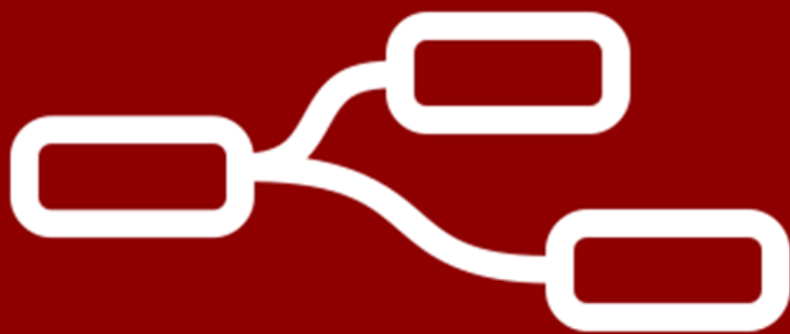


Snap4City IoT App



Corso di: Big Data Architectures

Prof. Paolo Nesi

Dep DINFO, University of Florence

Via S. Marta 3, 50139, Firenze, Italy

DISIT Lab, Sistemi Distribuiti e Tecnologie Internet

<http://www.disit.dinfo.unifi.it/> , <https://www.disit.org>

paolo.nesi@unifi.it <http://www.disit.dinfo.unifi.it/nesi>

Node-RED

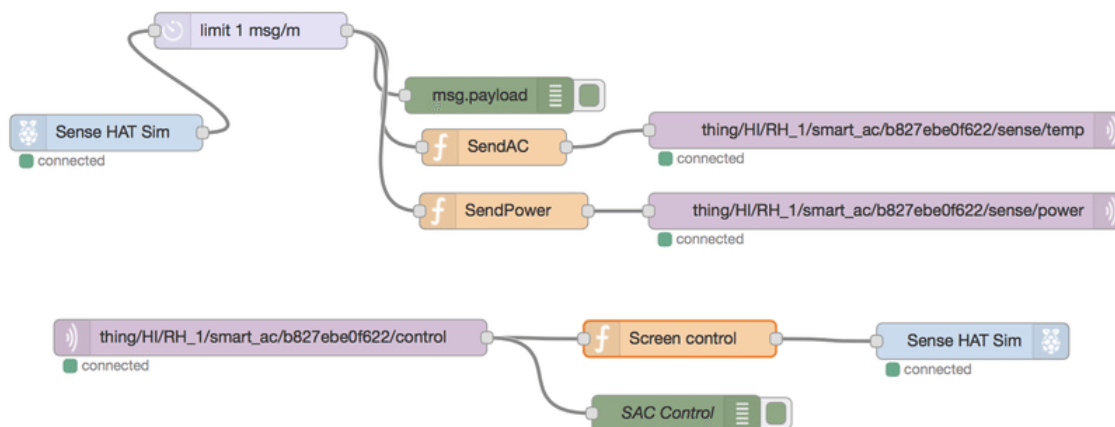
BIG DATA ARCHITECTURES

a.a. 2023/2024



Node-RED

- Node-RED (<https://nodered.org/>) is a programming tool for **wiring together** hardware devices, APIs and online services to create apps
- It provides a **browser-based editor** that makes it easy to wire together flows using the wide range of nodes...
- ...and then **deploy** and **run** the app in a single-click



Node-RED in Snap4City

- Node-RED is integrated into the Snap4City platform

The screenshot displays the Snap4City Node-RED interface. On the left, a sidebar shows the user profile for 'User: fanfa, Org: DISIT' and a list of dashboards and processing logics. The main workspace, titled 'mytest', contains a flow diagram with several nodes: 'http request', 'timestamp', 'function', 'switch', 'Save on CountRooms', and 'msg payload status'. The flow starts with an 'http request' node, followed by a 'timestamp' node, then a 'function' node. The output of the function node goes to a 'switch' node, which branches into two paths. One path goes to 'Save on CountRooms', followed by another 'function' node, and then 'msg payload status'. The other path goes to a 'Value of my CountRooms' node, followed by a 'function' node, a 'switch' node, and finally 'msg payload status'. A second 'timestamp' node is also present in the flow. The right sidebar shows a list of flows and a detailed view of a selected 'function' node with ID 'd8054298.42bb'.

Node-RED in Snap4City

- Node-RED is integrated into the Snap4City platform

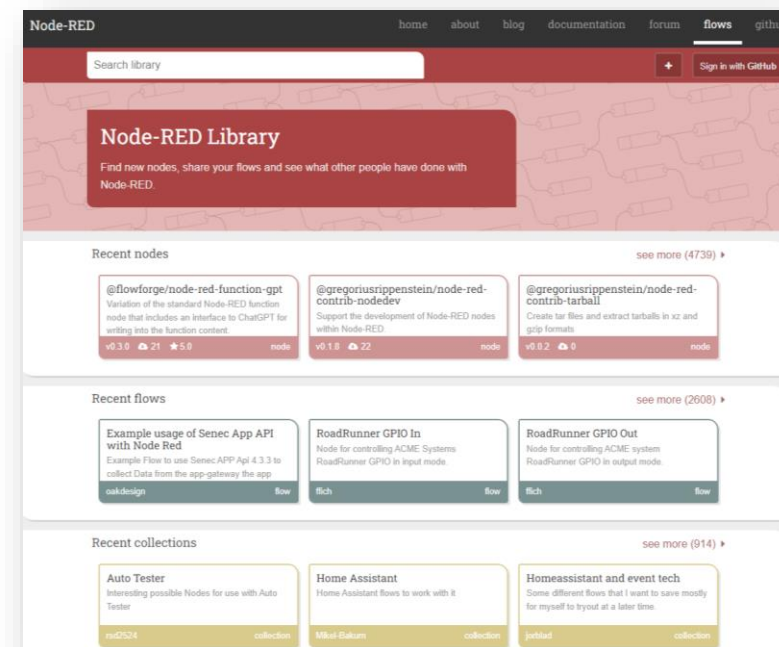
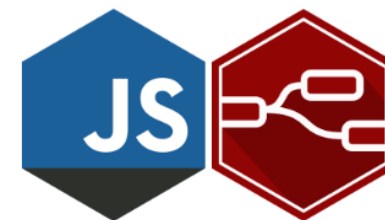
The image shows a screenshot of the Snap4City Node-RED interface. The main workspace displays a Node-RED flow editor with several nodes connected in a sequence. The nodes include 'timestamp', 'function', 'switch', 'Save on CountRooms', 'msg payload status', and 'Value of my CountRooms'. The interface is titled 'mytest' and includes a 'Deploy' button. On the left, a sidebar menu is visible, listing various components and services. A red dashed box highlights the 'Processing Logics / IOT App' menu item, which is also highlighted in blue. The sidebar menu includes items like 'My Snap4City.org', 'Tour Again', 'Extra Dashboard Widgets', 'Data Management, HLT', 'Knowledge and Maps', 'Processing Logics / IOT App', 'MicroServices for Proc.Logic/IOT Apps', 'MicroServices from DataAnalytic', 'IOT MicroServices for Final Users', 'IOT MicroServices for Developers', 'DOC: Processing Logic/IOT App', 'How to Develop Proc.Logic / Iot Apps', 'Create A MicroService from RestCall', 'Entity Directory and Devices', 'Resource Manager', 'Development Tools', 'Management', 'Decision Support Systems', 'Deploy and Installation', 'Help and Contacts', 'Documentation and Articles', 'My Profile', and 'Km4City portal'.

Node-RED in Snap4City

- Node-RED is integrated into the Snap4City platform
- It can be used to
 - Real-time data ingestion through IoT Orion broker
 - Data manipulation and conversion
 - Execute analytic processes (even with specific scheduling or after receiving events)
 - Perform business-logic operation on data
 - Create and control dashboard widgets
 - ...

Nodes

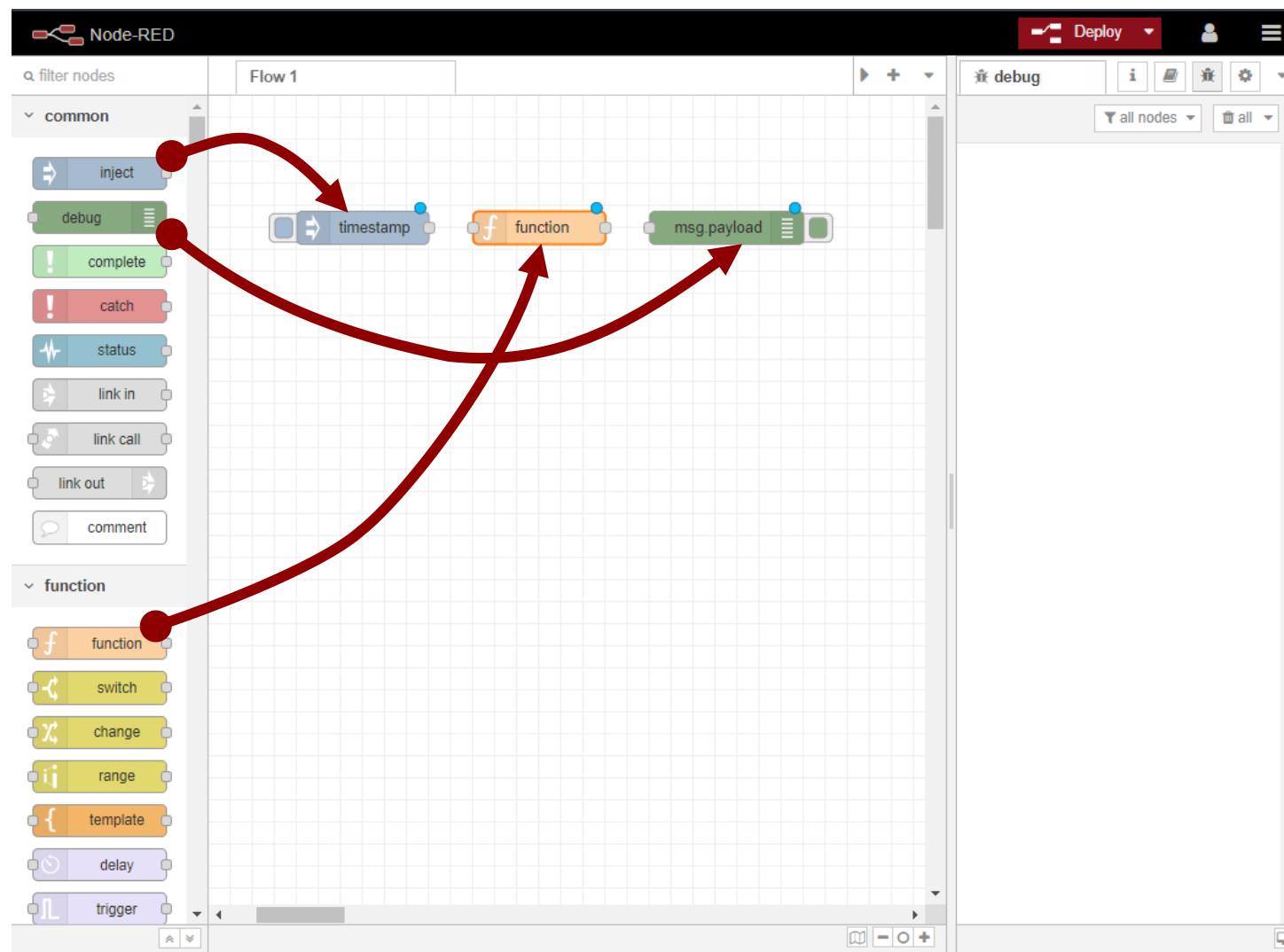
- Nodes are the building-blocks used to create flows
- Nodes can be of three main types
 - Input nodes (e.g., the **inject**)
 - Output nodes (e.g., the **debug**)
 - Input/output nodes (e.g., the **function**)
- The basic installation is provided with a minimum set of nodes
- Many other blocks can be easily added loading them from a large library made available by the JS Foundation



Nodes and flows

- In the Node-RED editor each node can be dragged into the main window
- Different nodes can then be linked to build flows
- Then each node in the flow can communicate with its neighbors by passing messages that are simple JavaScript objects that can have any set of properties

Nodes and flows



Nodes and flows

The image shows a screenshot of the Node-RED web interface. On the left, a palette of nodes is visible, with the 'inject' node highlighted. The main workspace shows a flow with one 'inject' node. A configuration dialog for the 'inject node' is open, displaying the following settings:

- Name:** Name
- msg. payload:** timestamp
- msg. topic:** (empty)
- Inject once after:** 0.1 seconds, then
- Repeat:** none
- Enabled:** (checked)

The 'msg. payload' dropdown menu is open, showing a list of options: flow., global., string, number, boolean, JSON, buffer, timestamp, expression, env variable, and msg..

Nodes and flows



Node-RED interface showing the 'Edit function node' dialog. The dialog includes a 'Name' field, tabs for 'Setup', 'On Start', 'On Message', and 'On Stop', and a code editor. The code editor contains the following JavaScript code:

```
1 let data = msg.payload;
2 data = 'Now is: ' + Date(data);
3 msg.payload = data;
4 return msg;
```

The left sidebar shows a list of nodes under the 'function' category, including 'function', 'switch', 'change', 'range', 'template', 'delay', and 'trigger'. The top right of the interface shows a 'Deploy' button and a 'debug' console.

Nodes and flows

The screenshot displays the Node-RED web interface. On the left, a sidebar shows a list of nodes under 'common' and 'function' categories. A 'debug' node is highlighted in the sidebar. The main workspace shows a flow with a 'debug' node and a 'timestamp' node. The 'Edit debug node' configuration panel is open, showing the following settings:

- Output:** msg.payload
- To:** debug window, system console, node status (32 characters)
- Name:** Name

At the bottom of the configuration panel, there is an 'Enabled' checkbox which is currently checked. To the right of the configuration panel, a 'debug' window is visible, showing a list of nodes and a 'all nodes' dropdown menu.

Nodes and flows

The image shows a screenshot of the Node-RED web interface. The main workspace displays a flow named "Flow 1" with three nodes connected in a sequence: a "timestamp" node, a "function" node, and a "msg.payload" node. Red circles highlight the connection points between the "timestamp" and "function" nodes, and between the "function" and "msg.payload" nodes. On the left, a sidebar lists various node categories: "common" (including inject, debug, complete, catch, status, link in, link call, link out, comment) and "function" (including function, switch, change, range, template, delay, trigger). The top right of the interface features a "Deploy" button and a "debug" panel.

Nodes and flows

The screenshot displays the Node-RED web interface. On the left, a sidebar contains a search bar and two categories of nodes: 'common' and 'function'. The 'common' category includes nodes like inject, debug, complete, catch, status, link in, link call, link out, and comment. The 'function' category includes function, switch, change, range, template, delay, and trigger. The main workspace, titled 'Flow 1', shows a horizontal flow of three nodes: 'timestamp', 'function', and 'msg.payload'. Each node is connected to the next by a line, and the connections are highlighted with red circles. A red hand icon points to the 'Deploy' button in the top right corner of the interface. The interface also features a top navigation bar with a 'Deploy' dropdown menu and a bottom status bar with zoom controls.

Nodes and flows

The screenshot displays the Node-RED web interface. On the left, a sidebar lists various nodes under 'common' and 'function' categories. The main workspace, titled 'Flow 1', contains a sequence of three nodes: a blue 'timestamp' node, an orange 'function' node, and a green 'msg.payload' node. A red hand icon is positioned over the 'timestamp' node. On the right, a 'debug' console shows a log entry for the 'msg.payload' node, indicating a string of 70 characters. The log text is: "Now is: Fri Nov 24 2023 16:31:43 GMT+0000 (Coordinated Universal Time)".

Nodes

The image displays a collection of nodes organized into eight categories, each with a dropdown arrow on the left:

- common**: inject, debug, complete, catch, status, link in, link out, comment
- location**: turf, worldmap, worldmap in, tracks, convex hull, **input** (amqp in, amqp2 in, stomp in), **output** (amqp out, amqp2 out, stomp out)
- function**: function, switch, change, range, template, delay, trigger, exec, zip, md5, soap request, string, xml converter, random, rbe
- network**: mqtt in, mqtt out, http in, http response, http request, websocket in, websocket out, tcp in, tcp out, tcp request, udp in, udp out
- social**: email, twitter in, email, twitter out, **advanced** (feedparser), **NGSI** (NGSI Entity, NGSI v2ToLD), **lwm2m** (lwm2m client in, lwm2m client out)
- dashboard**: button, dropdown, switch, slider, numeric, text input, date picker, colour picker, form, text, gauge, chart, audio out, notification, ui control, template
- sequence**: split, join, sort, batch, **parser** (csv, html, json, xml, yaml, base64, msgpack)
- storage**: file, file in, watch, ftp in, mysql, tail, **time** (sunrise)

Nodes – Snap4City libraries

The image displays a collection of Snap4City library nodes, organized into several categories:

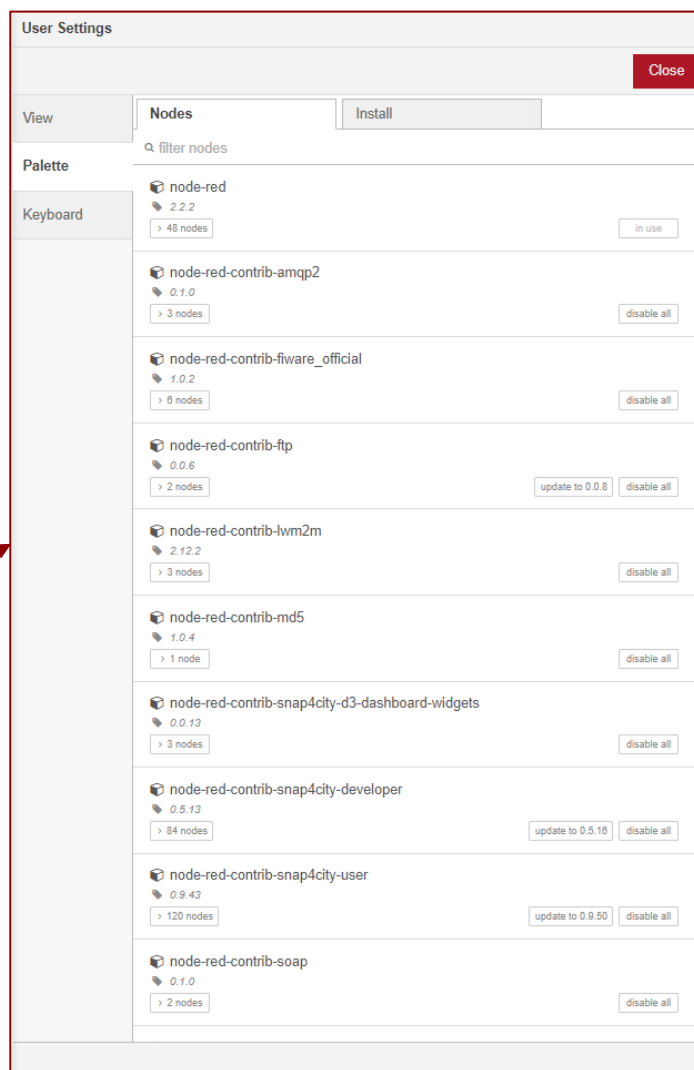
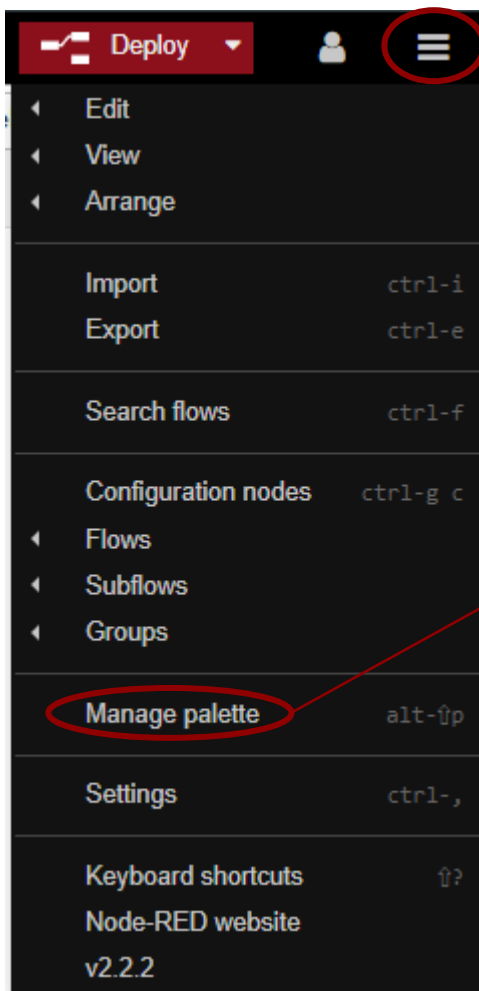
- S4CSearchDev:** Includes nodes for service search, event search, and full text search with various filters like 'near gps position' or 'within wkt area'.
- S4CDataAnalytic:** Contains nodes for descriptive statistics, trend plots, time series predictions, machine learning predictions, anomaly detection, plumber data analytic, and python data analytic.
- S4CMapping:** Features nodes for service info mapped, mapping, and set mapping.
- S4CManagement:** Includes nodes for checking job status (exists, in standby mode, shutdown, started), pausing/resuming jobs and triggers, and getting currently executing jobs.
- S4CUtility:** Contains nodes for service info dev and distance from coordinates.
- S4CSearch:** Offers nodes for service search near marker, within circle, along path, and event search within polygon.
- S4CBigData:** Includes nodes for datagate insert, search, create, and portia crawler.
- S4CIOTApp:** Features nodes for iotapp restart, upgrade, and ownership.
- S4CData:** Contains nodes for getting my data, delegator, delegated, and activity.

Nodes – Snap4City libraries

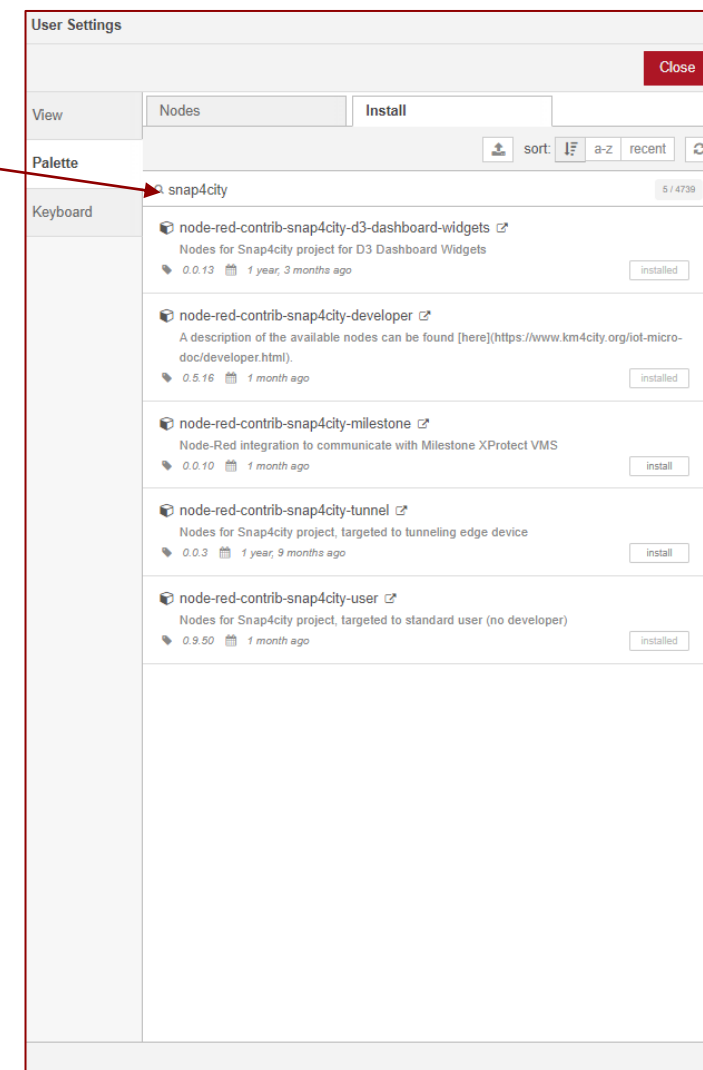
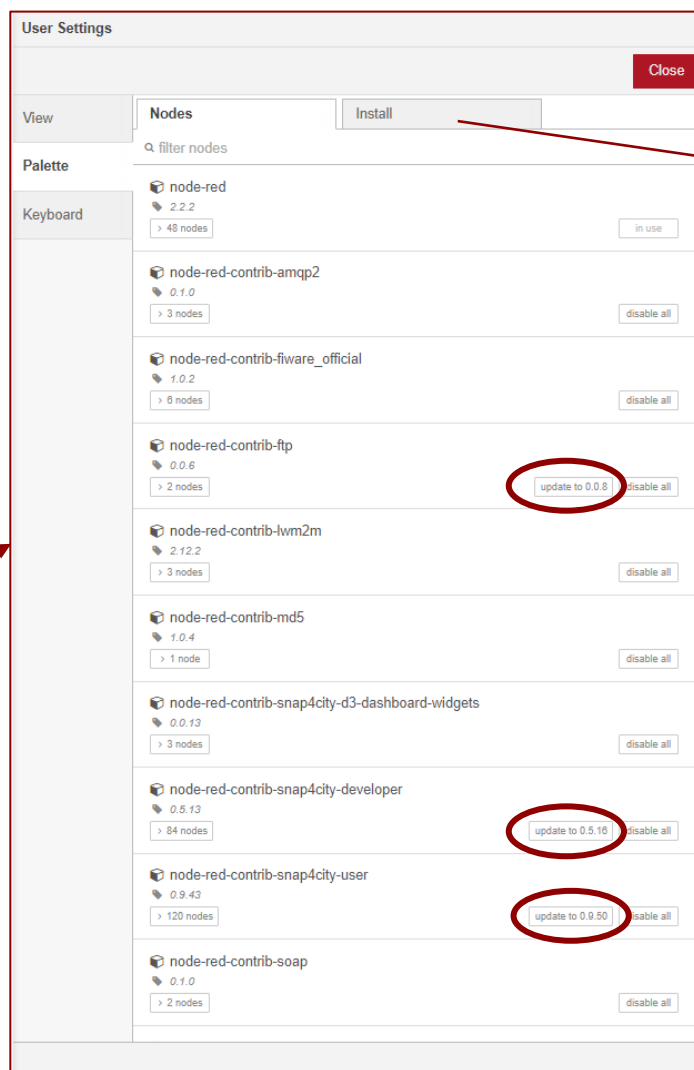
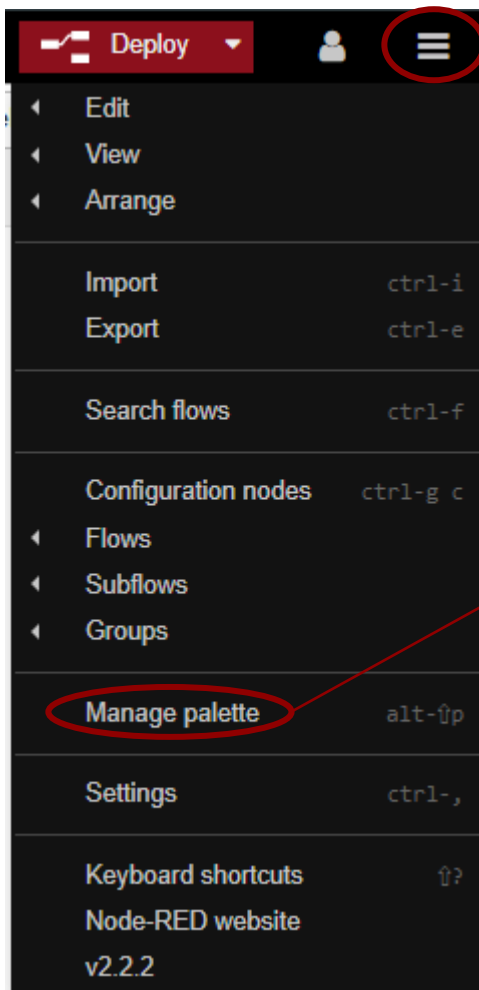
The image displays a comprehensive set of Snap4City library nodes, organized into several categories:

- S4CDashboard:** Includes nodes for data retrieval (e.g., 'get other activity on my data', 'save my data'), visualization (e.g., 'table - content', 'calendar', 'speak - synthesis'), and interaction (e.g., 'impulse - button', 'numeric - keyboard', 'switch - button').
- S4CKPIData:** Focuses on data management and retrieval (e.g., 'get my kpdata', 'get delegated kpdata', 'get public kpdata').
- S4CIOt:** Contains IoT-related nodes such as 'iotdirectory new device from model', 'delegate my device', and various 'fiware orion' API nodes.
- S4CLogDev:** Includes 'event log' and 'S4CView' nodes for displaying micro web apps and iframes.
- S4CSocial:** Features nodes for social media integration, like 'twitter last channel' and 'twitter last tweet'.
- S4CSigfox:** Includes 'sigfox device filter' and 'sigfox' nodes.
- S4CWhatif:** Contains 'get my scenarios' and 'save a scenario' nodes.
- UserCreated:** Lists user-generated nodes like 'Twitter Herit Data Sentiment Analysis Channel' and 'Twitter Herit Data Sentiment Analysis Search'.
- NGSi:** Includes nodes for 'NGSi Entity', 'NGSi Dataset', 'NGSi Update', 'NGSi Subscription', and 'NGSi v2ToLD'.
- social:** Contains nodes for 'email', 'twitter', and 'triplesToVirtuo'.
- subflows:** Includes 'triplesToVirtuo'.
- location:** Features nodes for 'utm', 'turf', 'worldmap', 'worldmap in', 'tracks', and 'convex hull'.
- Advanced FTP:** Includes 'Advanced FTP' and 'Advanced FTP Logger'.
- lwm2m:** Includes 'lwm2m client' nodes.

Manage palette



Manage palette – update & install



Import & export flows

Deploy

- Edit
- View
- Arrange
- Import** ctrl-i
- Export** ctrl-e
- Search flows ctrl-f
- Configuration nodes ctrl-g c
- Flows
- Subflows
- Groups
- Manage palette alt-⌘p
- Settings ctrl-,
- Keyboard shortcuts ↑?
- Node-RED website
- v2.2.2

Export nodes

Export selected nodes current flow all flows

Clipboard Export nodes JSON

Save On S4C

Local

```
[{"id":"3bda05975c47fbd8","type":"inject","z":"a37dc6863c2ce9e3","name":"","props":[{"p":"payload"}, {"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"oncedelay":0.1,"topic":"","payload":"","payloadType":"date","x":340,"y":120,"wires":[]}]
```

compact formatted

Cancel Download Copy to clipboard

Import & export flows

A screenshot of the Node-RED application's main menu. The menu is dark-themed and lists various actions. The 'Import' and 'Export' options are circled in red. A red arrow points from the 'Export' option to the 'Export nodes' dialog box shown in the next image.

- Deploy
- Edit
- View
- Arrange
- Import** (ctrl-I)
- Export** (ctrl-E)
- Search flows (ctrl-F)
- Configuration nodes (ctrl-G C)
- Flows
- Subflows
- Groups
- Manage palette (alt-UP)
- Settings (ctrl-,)
- Keyboard shortcuts (↑?)
- Node-RED website
- v2.2.2

The 'Export nodes' dialog box is shown. It has three tabs: 'selected nodes', 'current flow', and 'all flows'. The 'selected nodes' tab is active. Below the tabs, there are three sections: 'Clipboard', 'Export nodes', and 'JSON'. The 'Clipboard' section has 'Save On S4C' and 'Local' options. The 'Export nodes' section contains a text area with JSON data. The 'JSON' section is empty. A 'Cancel' button is at the bottom right.

```
[{"id": "3bda05975c47fbd", "type": "inject", "z": "a37dc6863c2ce9e3", "name": "", "props": [{"p": "payload"}, {"p": "topic", "vt": "str"}, {"p": "repeat": ""}, {"p": "cronTab": ""}, {"p": "once": false}, {"p": "oncedelay": 0.1}, {"p": "topic": ""}, {"p": "payload": ""}], "wires": [[]]}
```

The 'Import nodes' dialog box is shown. It has two tabs: 'Clipboard' and 'Paste flow json or select a file to import'. The 'Clipboard' tab is active. Below the tabs, there are four sections: 'S4C Flows', 'S4C Nodes', 'S4C DA Nodes', and 'Local'. The 'Examples' section is also visible. At the bottom, there are 'Import to' options: 'current flow' and 'new flow'. 'Cancel' and 'Import' buttons are at the bottom right.

Working with context

- Node-RED provides a way to store information that can be **shared between different nodes without using the messages** that pass through a flow. This is called **context**
- There are three different context scope:
 - Node - only visible to the node that set the value
 - Flow - visible to all nodes on the same flow (or tab in the editor)
 - Global - visible to all nodes
- For example, in the function node, we can use
 - `flow.set("count", 123);` - to set the context name "count"
 - `var myCount = flow.get("count");` - to read the context