**NODERED**

Node-RED is a low-code programming tool for event driven applications. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

**Installing Node-RED and generating a flow using UI:**

1. We can install Node-RED using the command “sudo npm install -g --unsafe-perm node-red”.
2. After installing the Node-RED we must start the server by using the command “node-red”
3. Then we must go to “[**http://localhost:1880**](http://localhost:1880/)” to access the UI.

Ref:” <https://www.npmjs.com/package/node-red>”

Table

Description automatically generated

This is the UI of the Node-RED server.

1.We can select the components from the left

2.Drag them into the white space.

3.configure the components.

4.click on Deploy on the top right corner which deploys the connected components into the flow.

**Generating a sample flow of adding two numbers:**

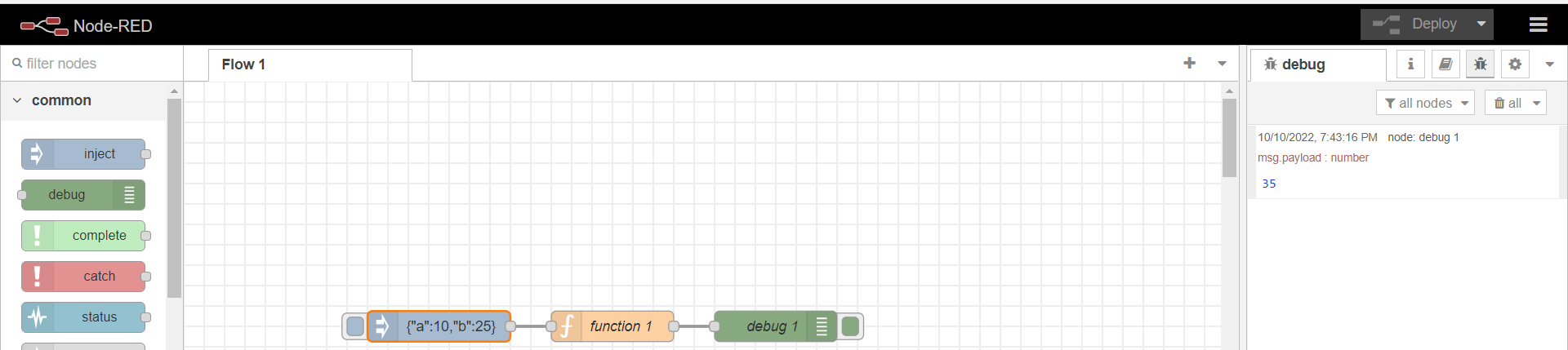
**Chart

Description automatically generated**

We created 3 components

1. Inject- This component takes an input of JSON object where we are passing 2 numbers with their respective keys.
2. Function- which takes the values from inject and we can write java script code which can manipulate the input data and return message payload.
3. Debug- It will display the values which are coming from the function.

After creating the components and connecting them, we must click on the deploy button on the top right corner so that we can use the flow. Blue circle above each component represents that they are not deployed into the flow.



Deploy button will send a post request to the “<http://localhost:1880/flows>” endpoint so that they will be deployed into the flow, and we can start sending inputs using inject component.

**Exporting/Importing a flow:**

Chart

Description automatically generated with low confidence

We can also export or import a flow which creates a JSON object for that connected components or we can create for entire flow. We can select the components which we want to export before going to export option. Selection will create a orange outline around the components.

Graphical user interface, application

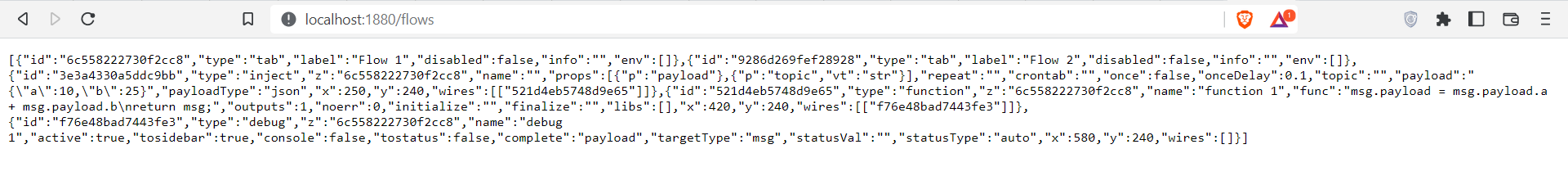
Description automatically generated

We can also select the components after clicking on the EXPORT option where we can select components, flow, all flows.

Graphical user interface

Description automatically generated

On the JSON tab the JSON object will be created so that which can be copied or downloaded to create components using the JSON object so that we don’t have to drag and configure the components again.



We can also use this URL <http://localhost:1880/flows> to see all the flows present in our server.

**Generating a sample flow of adding two numbers without UI:**

We observed the JSON object which is created when we are using export option and observed some common fields for each component.

**"id" - Id of the component(it is a randomly generated ID)**

**"type" - Type of that component(inject, function(component name)..etc)**

**"z" - flow ID**

**"name" - It will be displayed above the component(not mandatory)**

**"x": 320,(This is the HORIZONTAL position of the component in the window)**

**"y": 320,(This is the VERTICAL position of the component in the window)**

**"wires": [] - it is an array where we can give the IDs of the components so that they will be connected.**

**Example: "wires": [**

**[**

**"e1e62ee9814f407e",**

**"e25e51296f168f0c"**

**]**

**]**

We created a JSON object from scratch by providing the fields which are necessary to create a flow of adding two numbers.

[

**This is the JSON object of the flow**

{

"id": "abcd12354",

"type": "tab",

"label": "Flow 2",

"disabled": false,

},

**This is the JSON object of inject component**

{

"id": "o160995",

"type": "inject",

"z": "abcd12354",

"props": [

{

"p": "payload"

},

{

"p": "topic",

"vt": "str"

}

],

"repeat": "",

"once": false,

"onceDelay": 0.1,

"payload": "{\"a\":10,\"b\":25}",

"payloadType": "json",

"x": 250,

"y": 240,

"wires": [

[

"o160996"

]

]

},

**This is the JSON object of the function**

{

"id": "o160996",

"type": "function",

"z": "abcd12354",

"name": "function 1",

"func": "msg.payload = msg.payload.a + msg.payload.b\nreturn msg;",

"x": 420,

"y": 240,

"wires": [

[

"o160997"

]

]

},

**This is the JSON object of the debug/output.**

{

"id": "o160997",

"type": "debug",

"z": "abcd12354",

"name": "debug 1",

"active": true,

"complete": "payload",

"targetType": "msg",

"statusType": "auto",

"x": 580,

"y": 240,

"wires": []

}

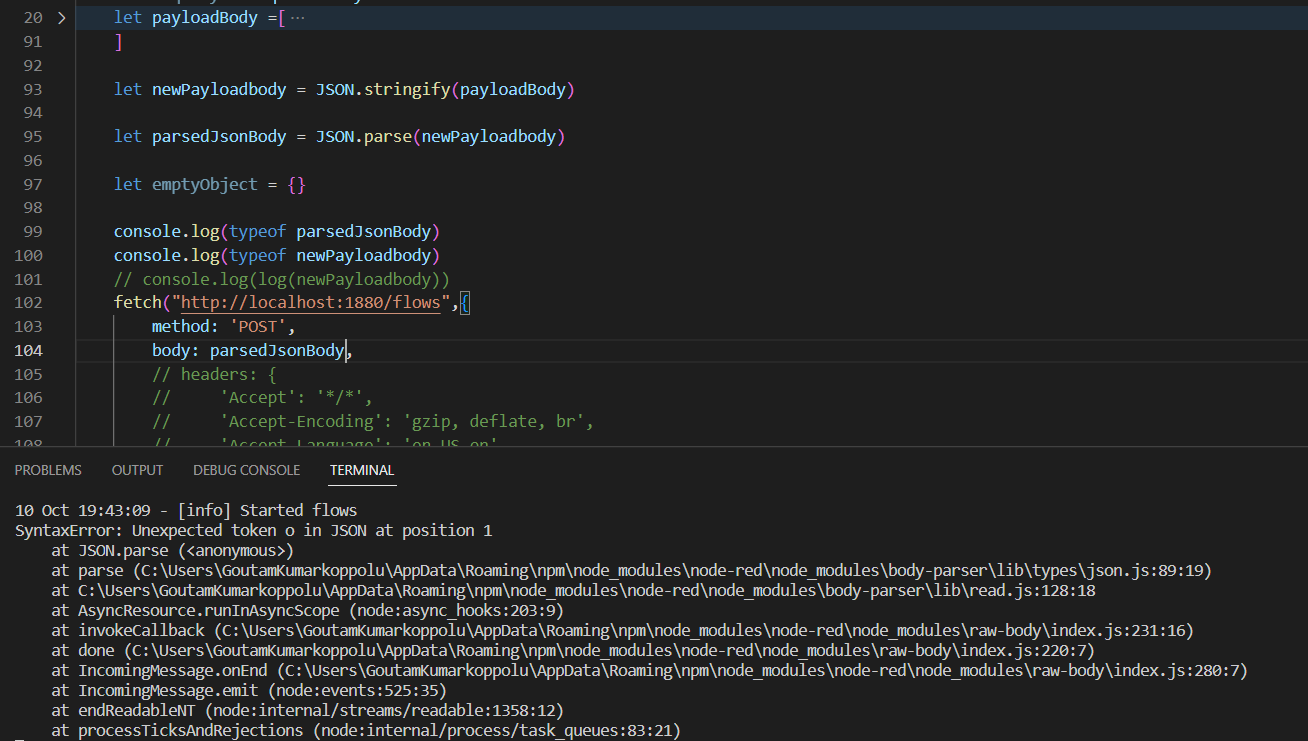
]

We created this JSON object which creates a flow to add two numbers with minimal attributes so that the functionality of the flow won’t get effected.

API URL: <http://localhost:1880/flows>

We tried hitting the above URL using post method by sending a string converted JSON object.

1. Above endpoint doesn’t accepts input in the form of JSON object.



1. We will encounter the error “SyntaxError: Unexpected token o in JSON at position 1” when we try to send response in JSON object format.

Ref: <https://www.positioniseverything.net/unexpected-token-o-in-json-at-position-1/>

1. We are sending the input in the below format which was accepted by the API.

[{"id":"abcd12354","type":"tab","label":"Flow 2","disabled":false},{"id":"o160995","type":"inject","z":"abcd12354","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","once":false,"onceDelay":0.1,"payload":"{\"a\":10,\"b\":25}","payloadType":"json","x":250,"y":240,"wires":[["o160996"]]},{"id":"o160996","type":"function","z":"abcd12354","name":"function 1","func":"msg.payload = msg.payload.a + msg.payload.b\nreturn msg;","x":420,"y":240,"wires":[["o160997"]]},{"id":"o160997","type":"debug","z":"abcd12354","name":"debug 1","active":true,"complete":"payload","targetType":"msg","statusType":"auto","x":580,"y":240,"wires":[]}]

**Issues when sending through code:**

**1.We must review changes in the UI**

Graphical user interface

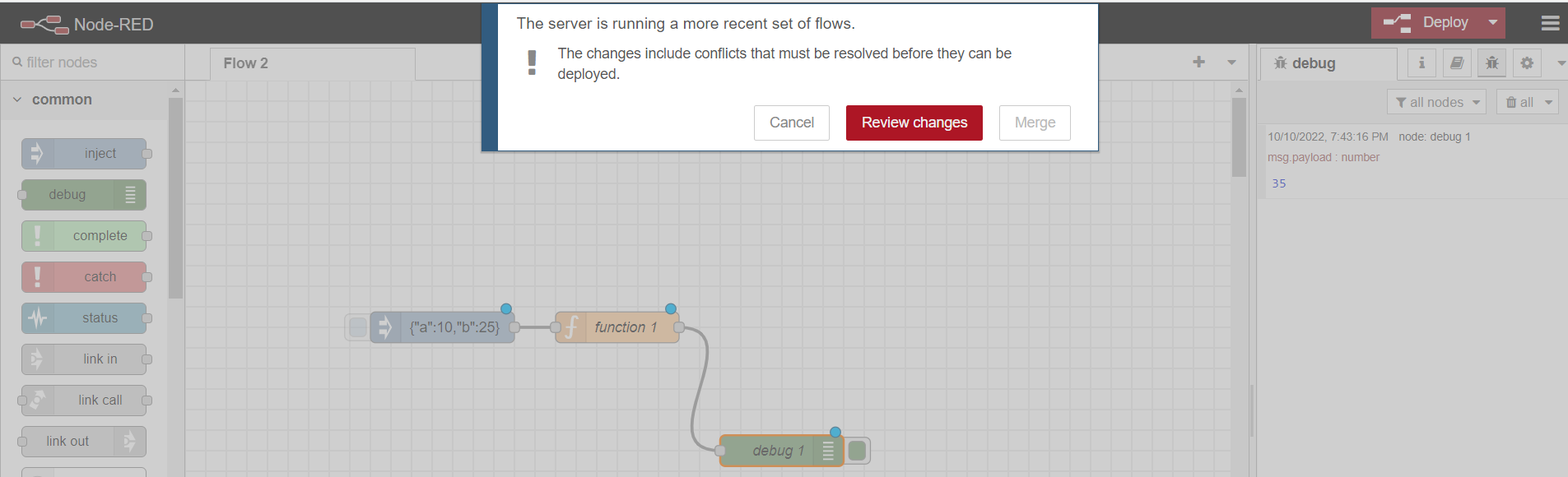
Description automatically generated with low confidence

When we send the JSON object using our code, it creates a popup in the UI to create the flow by reviewing changes.

A picture containing graphical user interface

Description automatically generated

If there are no conflicts, Merge option will be available if there are no conflicts.



If there are any conflicts, we must resolve using review changes later we can merge them.

**2.Deploy option is not available until we change something in the UI for any component**

Chart

Description automatically generated with medium confidence

Once review changes are completed and merge was successful, components were just placed and wired with other components in the UI. We must deploy them first if we must run them. Deploy option is not available until we make some changes in the components

Diagram

Description automatically generated

We have changed the position of debug component vertically. After doing that change, we can see the deploy option in top right corner which was not available in the first place.

Components wont work until they were deployed. We can see blue circle above each component which was a sign that they aren’t deployed.

**3.Previous connected components gets deleted from the flow if we deploy new components with different functionality**



We can’t add more than one functionality of connected components to a flow using code. When we try to create another JSON object with different functionality using code, previous connected components will get deleted from the flow.

Chart

Description automatically generated with medium confidence

But we can create different connected components performing different functionalities on the same flow using UI. Every functionality was different, and no functionality was getting effected by other.

**Conclusion:** When we are using Node-RED without UI we must go to the UI once again to review changes and merge once we send the object from code. When a new flow is sent through code, all the existing flows were deleted. But in UI we can add as many flows and as many components as we want which are independent and not affected by other flows or components.