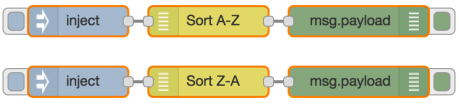
# Custom Nodes in Node-RED

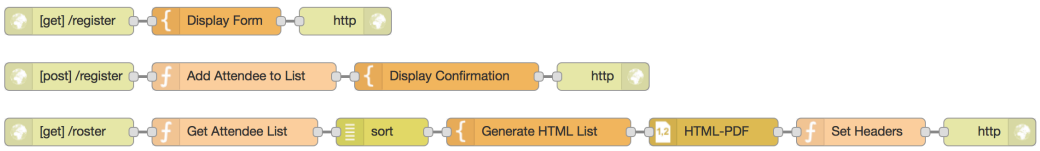
## Hands-On Lab

**Author: JeanCarl Bisson | jbisson@us.ibm.com | @dothewww**

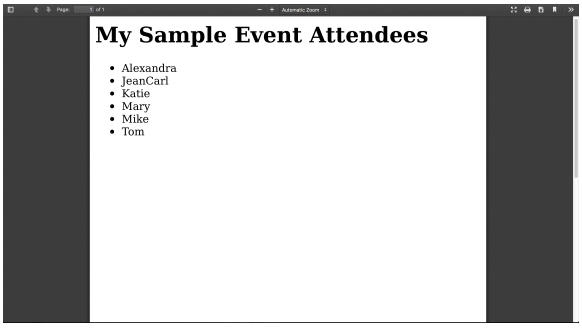




*Create a custom node that sorts an array in alphabetical order.*




*Create a registration form, save names into a global context array, and display the list in alphabetical order in a PDF file.*

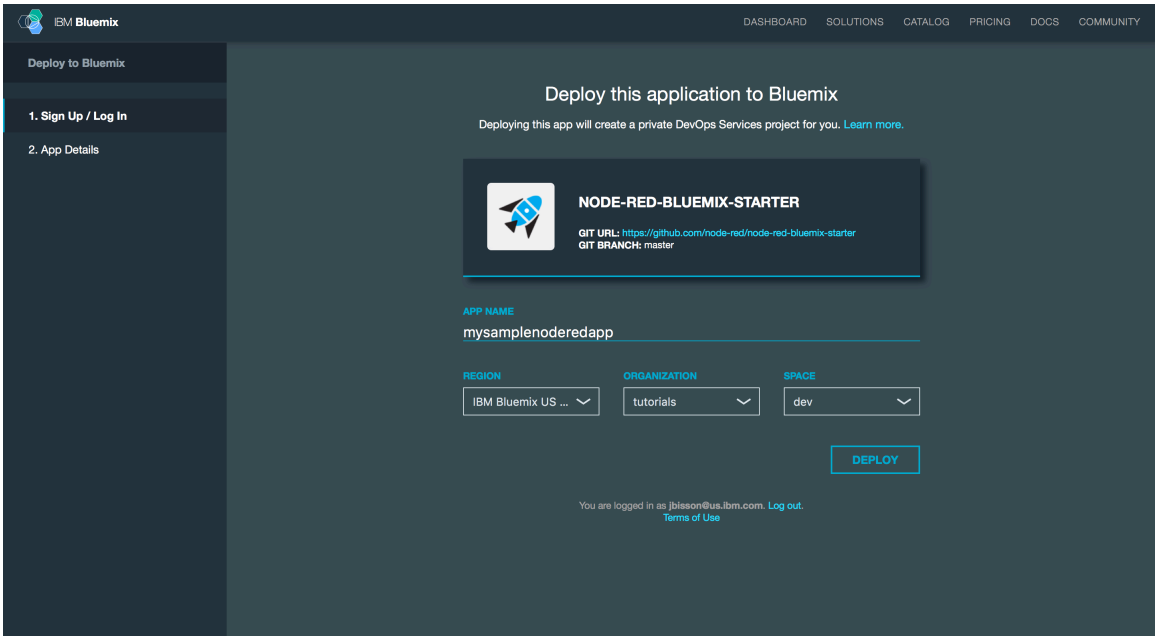




A digital copy of this lab and completed flows can be found at:
`http://ibm.biz/node-red-custom-nodes`

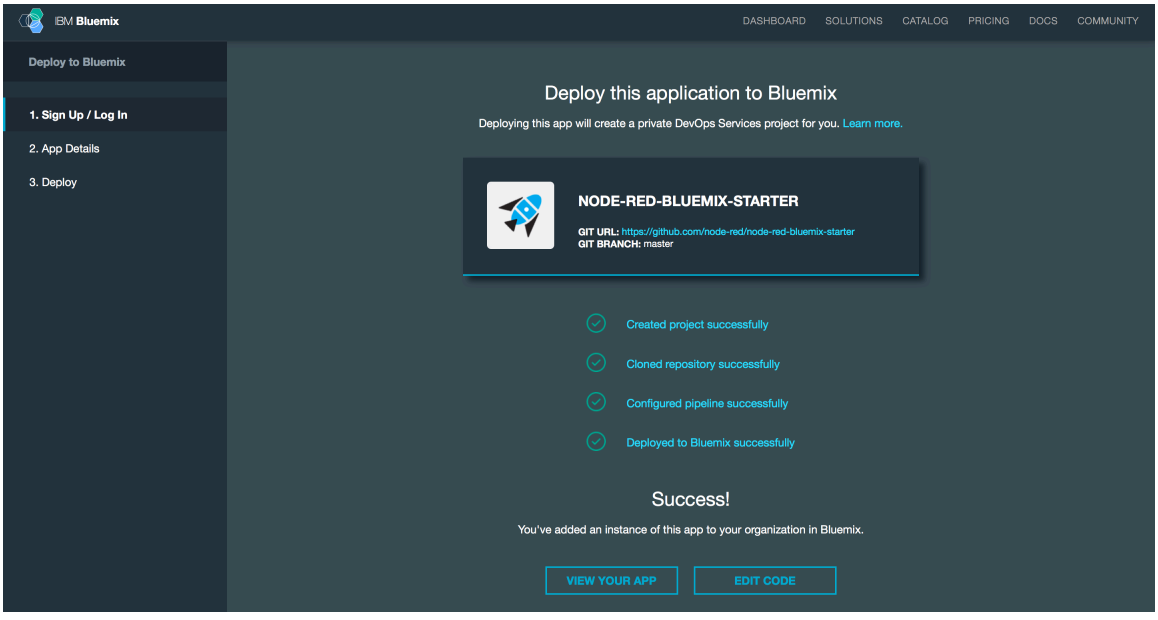
# Deploy Node-RED Application to IBM Bluemix

In this section, we will deploy a Node-RED application from a Git repo. Note that this is a different method from deploying Node-RED from the Node-RED Starter boilerplate application from the IBM Bluemix catalog. By deploying it from a Git repo, we will be able to modify the Node.js code and, when there is new code committed, the IBM DevOps pipeline will deploy the changes to the application instance in IBM Bluemix.

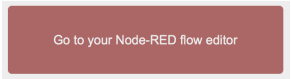
1. Begin by deploying the IBM Bluemix Node-RED application from a GitHub repo by visiting `http://ibm.biz/node-red-starter`
2. If you are prompted to login in, sign in with your IBM Bluemix account credentials. Select a region, organization, and space where this application should be deployed to. Click on **Deploy**.



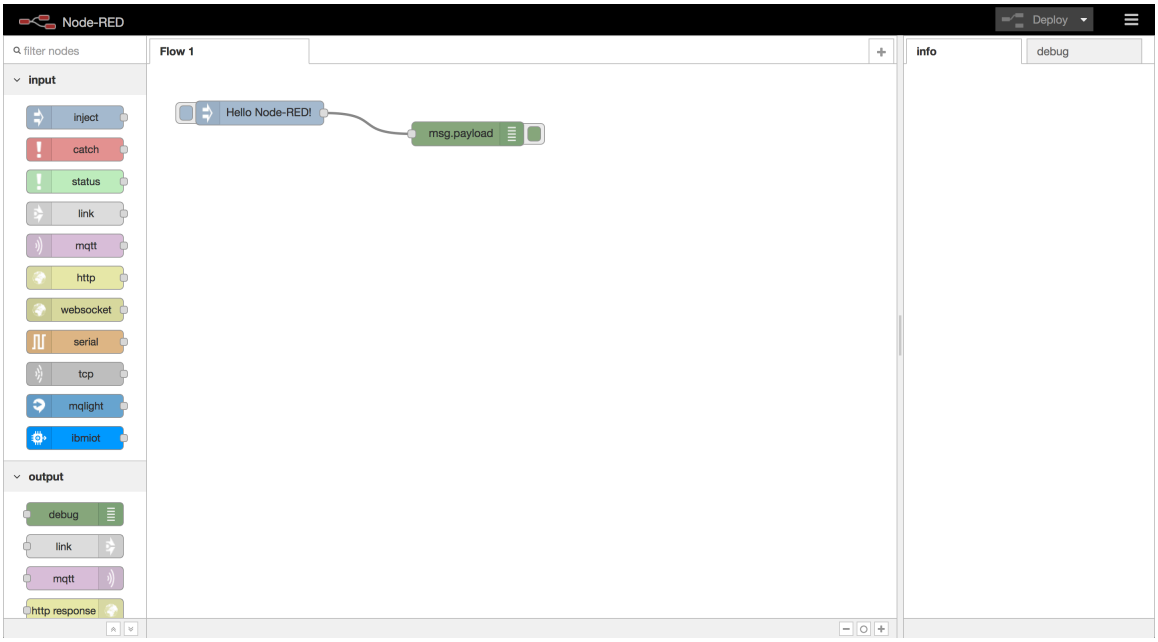

3. IBM DevOps will create a Git project in your Jazz Hub account, clone the GitHub repo, configure the IBM Bluemix DevOps pipeline, and deploy the code to a new application instance. When completed, click on the **View Your App** link.



4. Click on the big red button labeled **Go to Node-RED editor**.

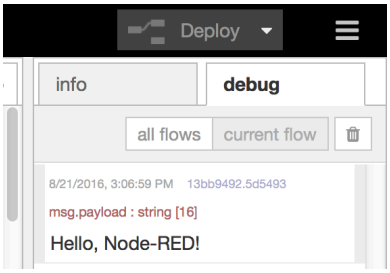



5. This is the default Node-RED application. On the left side are nodes that can be dragged into the middle pane, called a canvas, and connected together via wires connecting the grey knobs, to form what are called flows. In the right sidebar, info/documentation about the selected node is displayed. Select the **debug** tab in the right-hand pane.



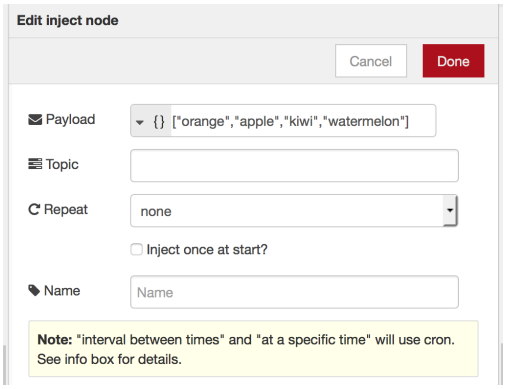

6. Click on the blue tab on the left side of the 

node. The contents of the message payload is displayed in the debug pane.




7. Let's modify the existing flow so it injects a JSON array into the flow. Double-click on the 

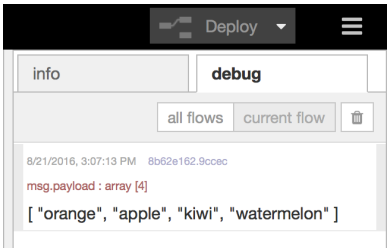
node and modify it as shown below.



8. Click on the 

button in the top-right corner to save and deploy your changes.

9. Click on blue tab on the left side of the inject node. This will inject the contents of the payload, a JSON array, into the flow. This message is then passed to the debug node, which displays the contents of the payload in the debug pane.




10. Let's imagine, while prototyping our application, we need to sort this list in alphabetical order. Add a 

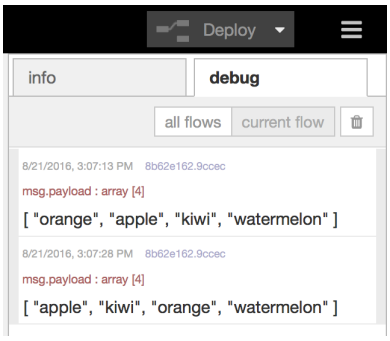
node with the JavaScript code shown below.

```
1.  msg.payload.sort();
2.  return msg;
```


11. Connect the nodes as shown below.


12. Click on the 

button in the top-right corner to save and deploy your changes.
13. Click on blue tab on the left side of the inject node. Again, the list is injected into the flow. This time, the function node sorts the list alphabetically and the debug node displays the contents in the debug pane.

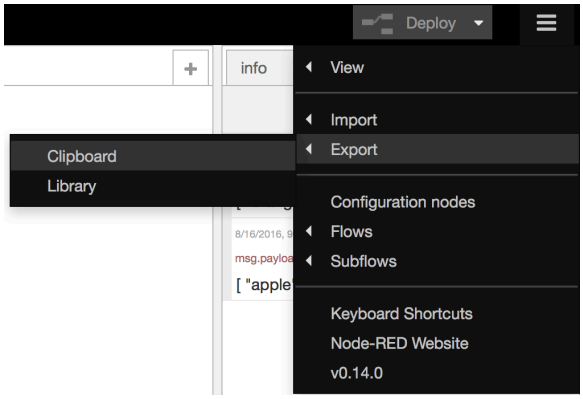


# Importing and Exporting Flows

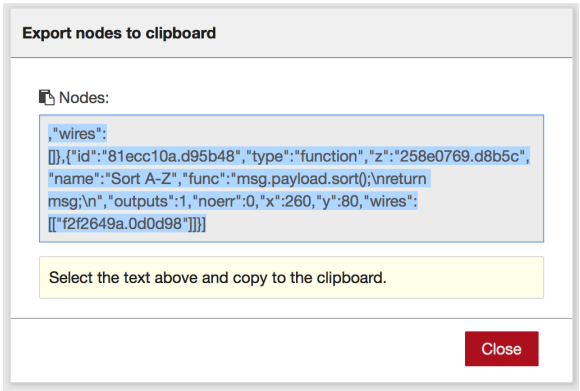
We're so excited about the flow we've created that we want to share the code with another developer. In this section, we will export the flow so that we can share it.

1. Drag a rectangle around the three nodes to select the flow.


2. Click on the menu in the top-right corner. Select **Export** > **Clipboard**.

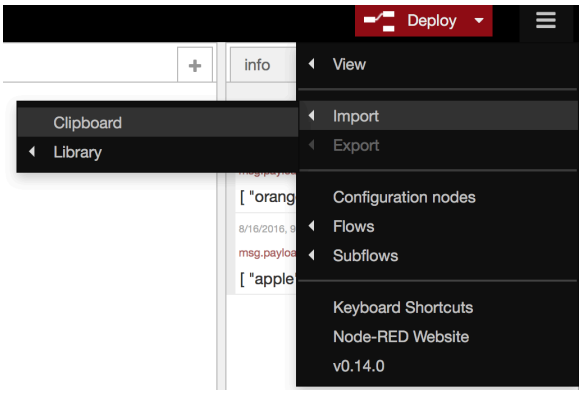



3. A dialog appears with JSON code. Copy this code. This is a JSON representation of the flow that can be shared with another developer, perhaps in a GitHub Gist and then imported into another Node-RED application.

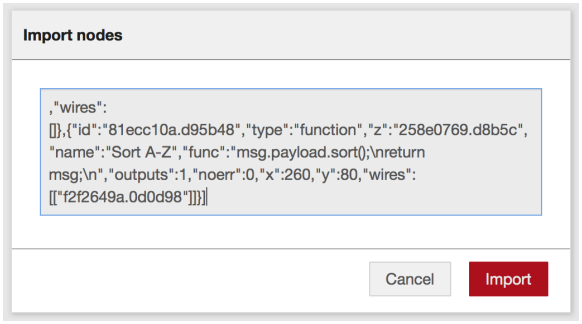



4. Now let's see how to import this JSON into a Node-RED application. Copy the code. Click **Close**. Delete the flow. The canvas should now be blank.

5. Click on the menu in the top-right corner. Select **Import** > **Clipboard**.




6. Paste the JSON code into the textbox. Click on **Import**.

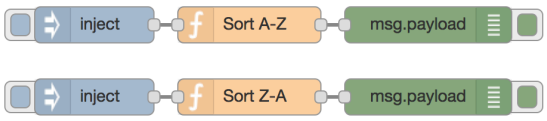



7. Click on the 

button in the top-right corner to save and deploy your changes. You can find flows that other developers in the Node-RED have contributed at `http://www.nodered.org`.

# Create a Custom Node

Let's imagine our client has come to us and has requested that we add the option to also sort the array in reverse order. We could copy the flow and modify the second JavaScript function as shown below.

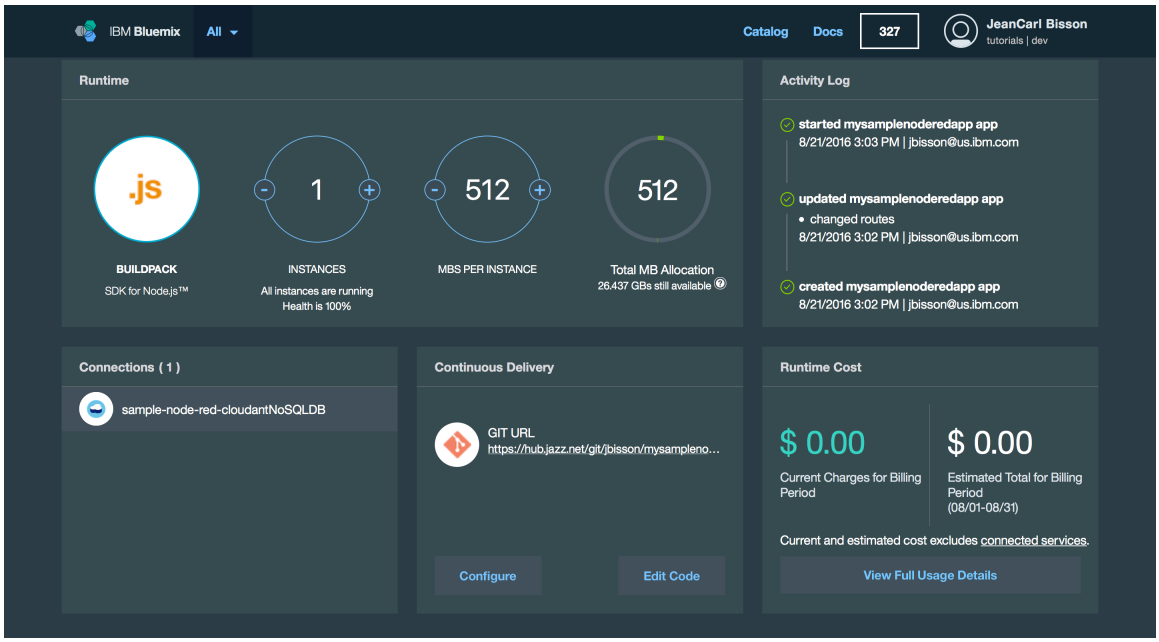




While this is a quick way to add this feature, it also starts to touch on a good coding practice called D.R.Y. (Don't Repeat Yourself). Our Node-RED application can quickly become cluttered if we continue to add new sorting options to copies of this function node, changing little details but keeping the main functionality the same. In this section, we will create a custom node that can be reused.

Underneath the hood, a node consists of a pair of files: a JavaScript file that defines what the node does, and a HTML file that defines the node's properties, edit dialog and help text.

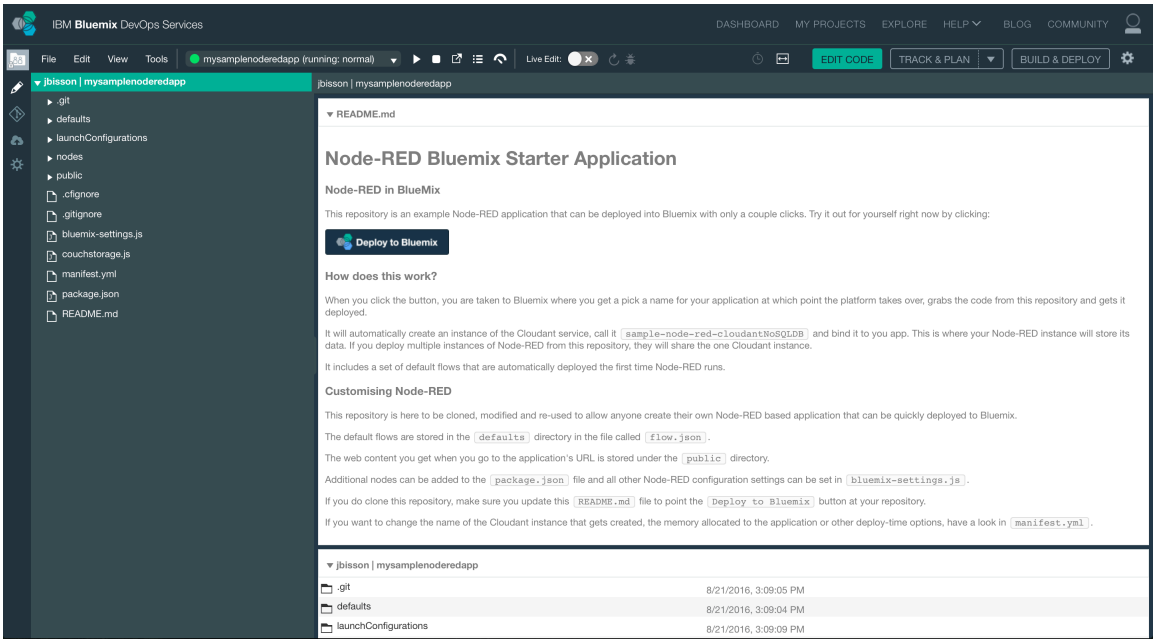
1. Return to the application overview in your IBM Bluemix dashboard. Click on the **Edit Code** button under the Continuous Delivery section.



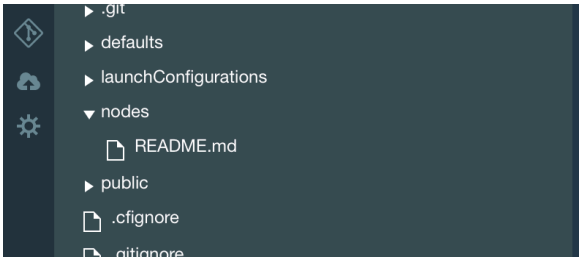

On the first visit, IBM Bluemix DevOps Services will clone the repo and create a new workspace for you. When completed, you will be taken to an online editor that looks similar to the one shown below.

IBM Bluemix DevOps Services provides a number of services and tools that make it easy to work with code in a team environment that is Git version-controlled. You can also specify a pipeline of actions that should be taken when new code is pushed to the Git repo. For example, you can run tests against new changes before deploying them to a production environment to avoid breaking an application.

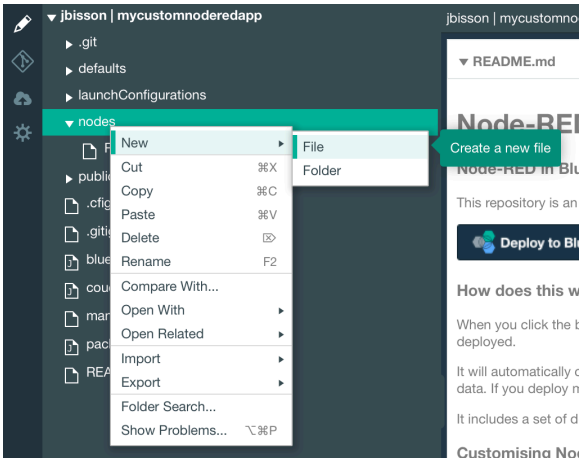
For our application, we'll use the default Build and Deploy stages for Node.js.



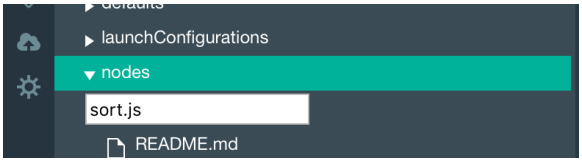

2. Click on the **nodes** directory to expand the directory tree. It has one file, a README.md file. This directory is where you place node JavaScript and HTML files that the Node-RED application should include in the palette.




3. Right click on the nodes directory to display a context menu. Select **New** > **File**.



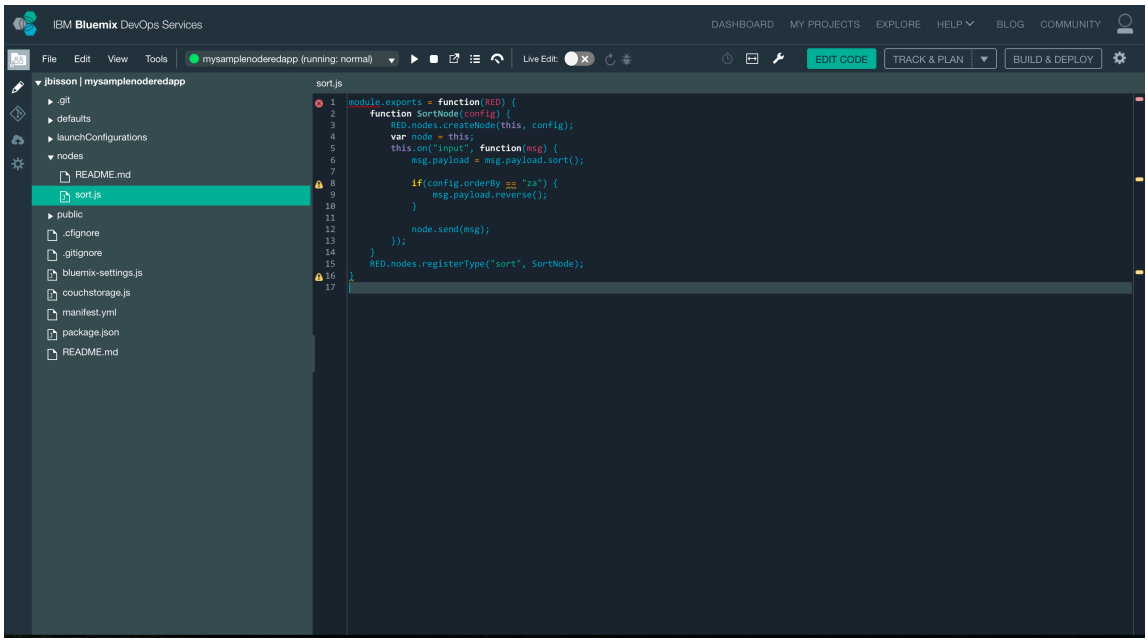

4. Enter the filename `sort.js`. Press Enter when finished. You have created a new file in your workspace.



5. Enter the following code into the main code window on the right. This code is also available in the file `1-sort.js` at `http://ibm.biz/BdrgRx`

```
module.exports = function(RED) {
    function SortNode(config) {
        RED.nodes.createNode(this, config);
        var node = this;
        this.on("input", function(msg) {
            msg.payload = msg.payload.sort();

            if(config.orderBy == "za") {
               msg.payload.reverse();
            }

            node.send(msg);
        });
    }
    RED.nodes.registerType("sort", SortNode);
}
```

Get the code:
`ibm.biz/BdrgRx`

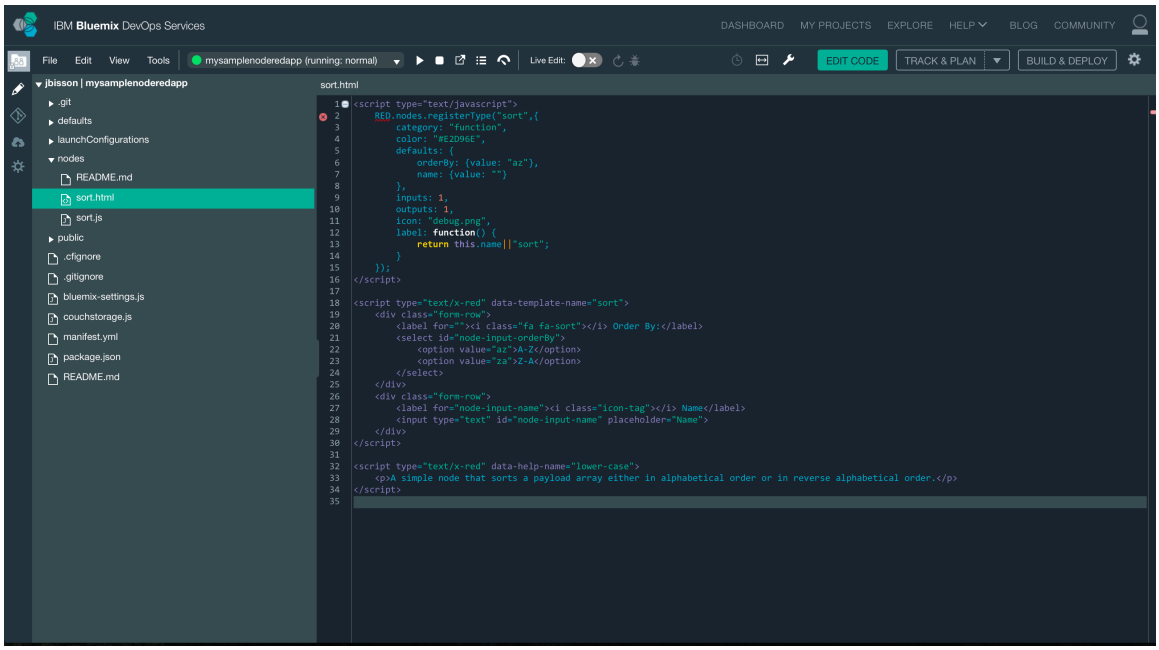



6. Create another file named `sort.html` with the code shown below. This code is also available in the file `2-sort.html` at `http://ibm.biz/BdrgRF`

```
<script type="text/javascript">
    RED.nodes.registerType("sort",{
        category: "function",
        color: "#E2D96E",
        defaults: {
            orderBy: {value: "az"},
            name: {value: ""}
        },
        inputs: 1,
        outputs: 1,
        icon: "debug.png",
        label: function() {
            return this.name||"sort";
        }
    });
</script>

<script type="text/x-red" data-template-name="sort">
    <div class="form-row">
        <label for=""><i class="fa fa-sort"></i> Order By:</label>
        <select id="node-input-orderBy">
            <option value="az">A-Z</option>
            <option value="za">Z-A</option>
        </select>
    </div>
```

Get the code:
`ibm.biz/BdrgRF`

```
26.      <div class="form-row">
27.          <label for="node-input-name"><i class="icon-tag"></i> Name</label>
28.          <input type="text" id="node-input-name" placeholder="Name">
29.      </div>
30.  </script>
31.
32.  <script type="text/x-red" data-help-name="lower-case">
33.      <p>A simple node that sorts a payload array either in alphabetical order or in
     reverse alphabetical order.</p>
34.  </script>
```



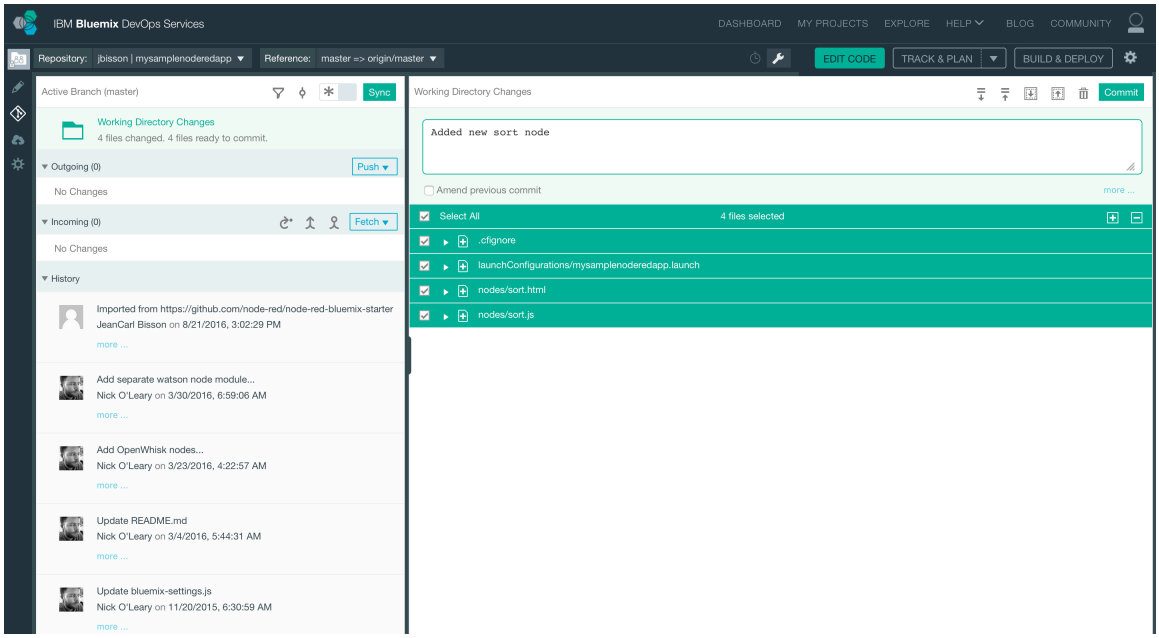

7. Let's take a moment and understand some things about the code in this file.

```
 1.  <script type="text/javascript">
 2.      RED.nodes.registerType("sort",{                    Registers the node type sort

 3.          category: "function",                          Specifies to categorize this under function
                                                            category

 4.          color: "#E2D96E",                              Changes the HTML background color of the node

 5.          defaults: {                                    Defines the default values of the properties
 6.              orderBy: {value: "az"},                    if they aren't defined.
 7.              name: {value: ""}
 8.          },
 9.          inputs: 1,                                     Specifies how many inputs this node has

10.          outputs: 1,                                    Specifies how many outputs this node has

11.          icon: "debug.png",                             Specifies an icon for this node

12.          label: function() {                            Specifies a callback function that returns the
13.              return this.name||"sort";                  label displayed on the node
14.          }
15.      });
16.  </script>
17.
18.  <script type="text/x-red" data-template-name="sort">   Template that defines the fields that can be
19.      <div class="form-row">                             configured in the edit dialog
20.          <label for=""><i class="fa fa-sort"></i> Order
     By:</label>
21.          <select id="node-input-orderBy">
22.              <option value="az">A-Z</option>
23.              <option value="za">Z-A</option>
24.          </select>
25.      </div>
26.      <div class="form-row">
27.          <label for="node-input-name"><i class="icon-
     tag"></i> Name</label>
```

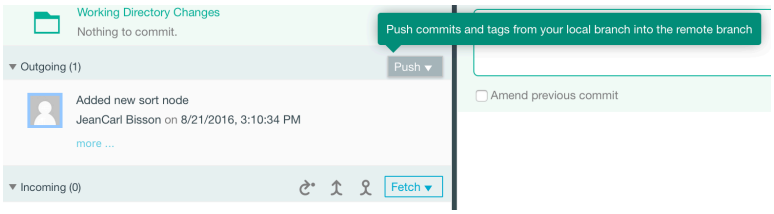
```
28.          <input type="text" id="node-input-name"
     placeholder="Name">
29.      </div>
30.  </script>
31.
32.  <script type="text/x-red" data-help-name="lower-case">     Template that defines the help contents that
33.      <p>A simple node that sorts a payload array either     are displayed in the info tab when the node is
     in alphabetical order or in reverse alphabetical          selected in the palette or canvas.
     order.</p>
34.  </script>
```

8. Switch to the Git view by clicking on the Git icon in the vertical toolbar in the left sidebar.

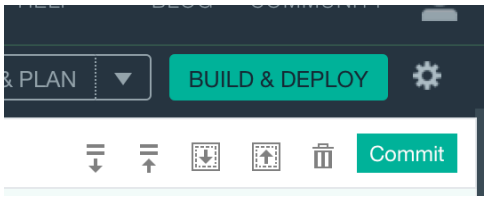

9. You'll see the uncommitted files we just created. Enter a commit message and click on **Commit**.



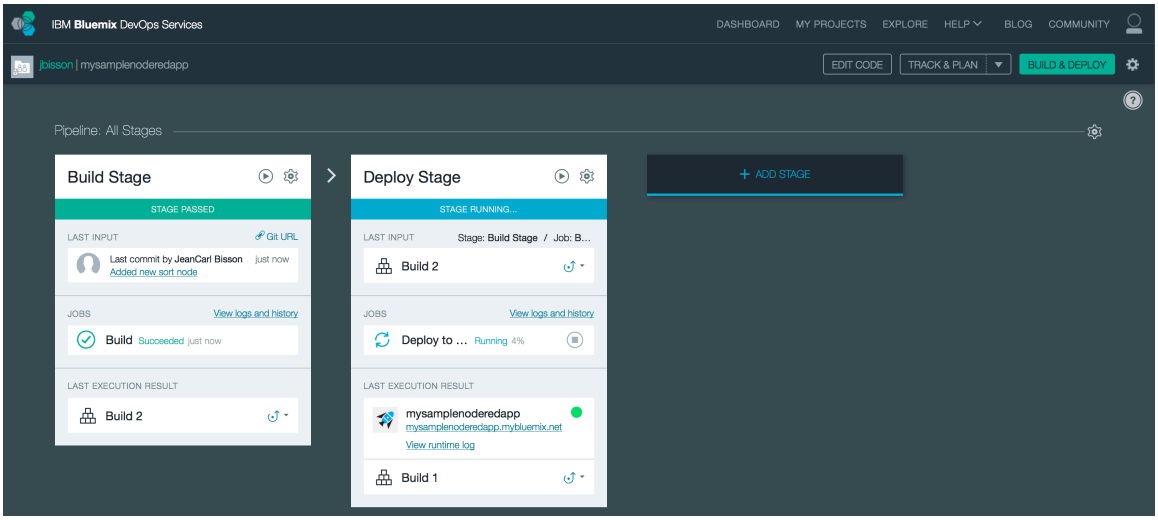

10. An entry will be added under the Outgoing section containing the commit. Click on **Push** to push the changes to the remote branch.



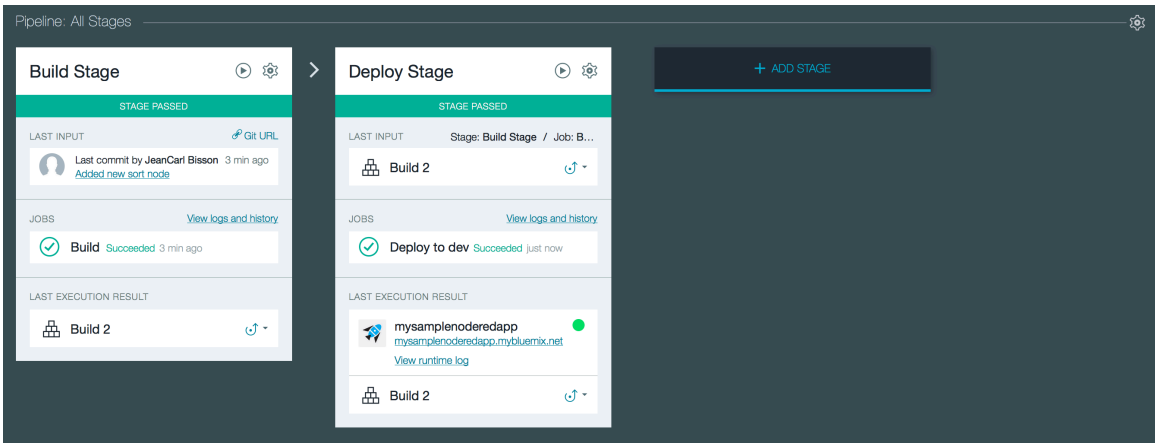
11. Click on the **Build & Deploy** button in the top-right corner.



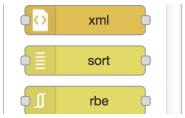

12. When new code is committed to this branch, the IBM Bluemix DevOps Services pipeline deploys the changes to the IBM Bluemix application that is linked. The Build Stage packages the code from your project along with the Node.js buildpack. The Deploy Stage deploys the result to the IBM Bluemix platform. You can click on the gears to view the settings. During the Deploy Stage, click on the **View logs and history** link to see what is happening behind the scenes.

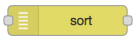



13. When the application has finished being deployed, both stages should be green.

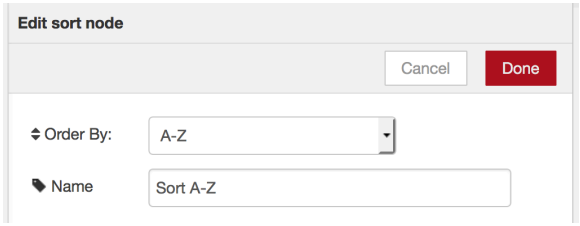



14. Return to your Node-RED application and refresh the editor. Scroll down to the section labeled **function**. A sort node should be displayed in the list.

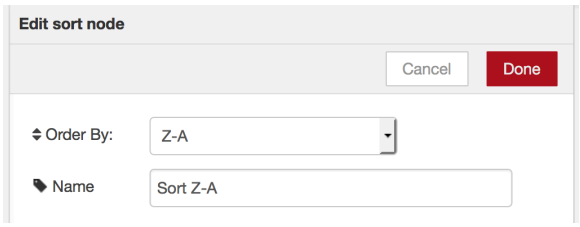


15. Replace the JavaScript function with the 

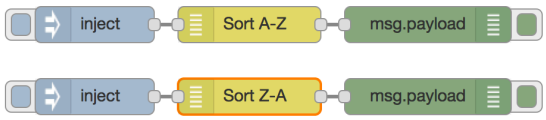
node. In the first flow, use the default **Order By** value of `A-Z`.




16. In the second flow, use the **Order By** value of `Z-A`.




17. The two flows are shown below.




18. Click on the 

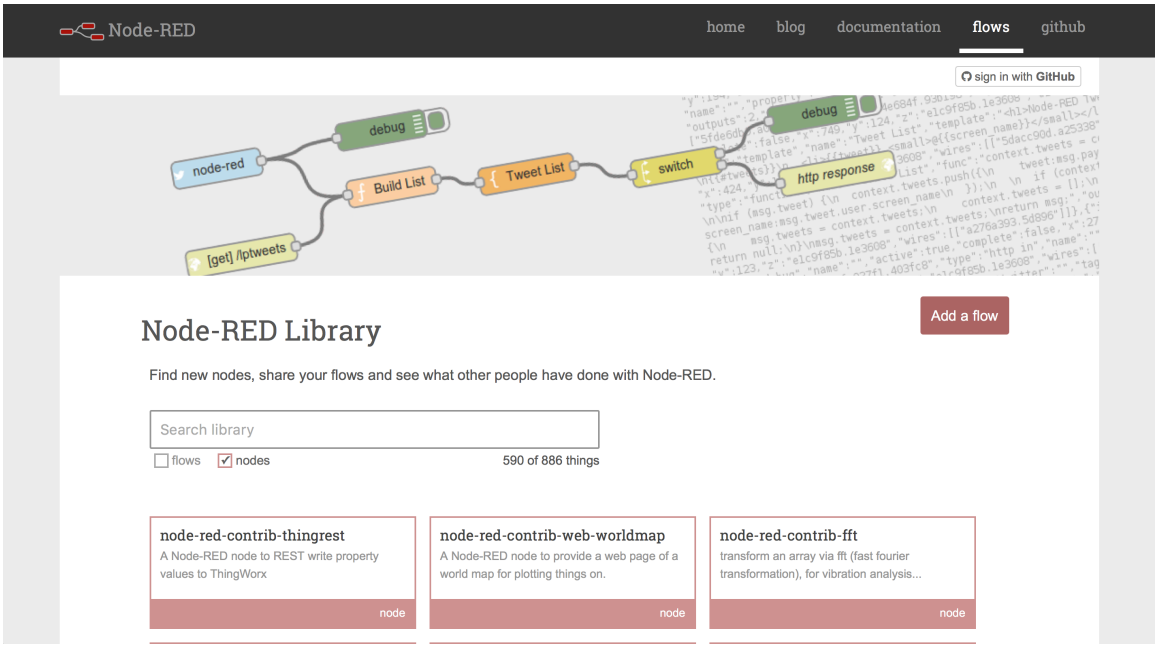
button in the top-right corner to save and deploy your changes.
19. When the inject tab for the first flow (sorting in alphabetical order) is triggered, the debug tab displays the array sorted in alphabetical order.


20. When the inject tab for the second flow (sorting in reverse alphabetical order) is triggered, the debug tab displays the array sorted in reverse alphabetical order.

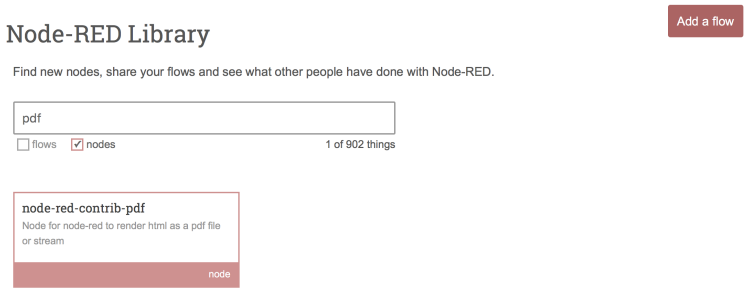

You have created your first node in Node-RED and deployed it to IBM Bluemix. What other sorting options could you add to your node? Return to the IBM Bluemix DevOps Services editor and modify the code, commit the changes, push the code to the remote branch, and wait for the changes to be deployed to the IBM Bluemix application.

# Installing a Custom Node from NPM

Since Node-RED is an open-sourced community project, Node-RED has an active community of node contributors. You can find community created nodes at http://flows.nodered.org/ .



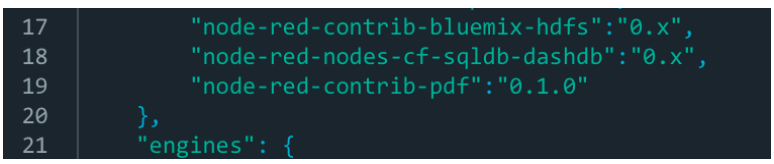

In this section, let's imagine we need to generate a PDF file. Search the library for `pdf`, with the filter checkbox selected for `nodes`. The search results include `node-red-contrib-pdf`, a node someone has added to the NPM package manager.




1. Note the name (`node-red-contrib-pdf`) and version (`0.1.0`) of the NPM package on the details page.

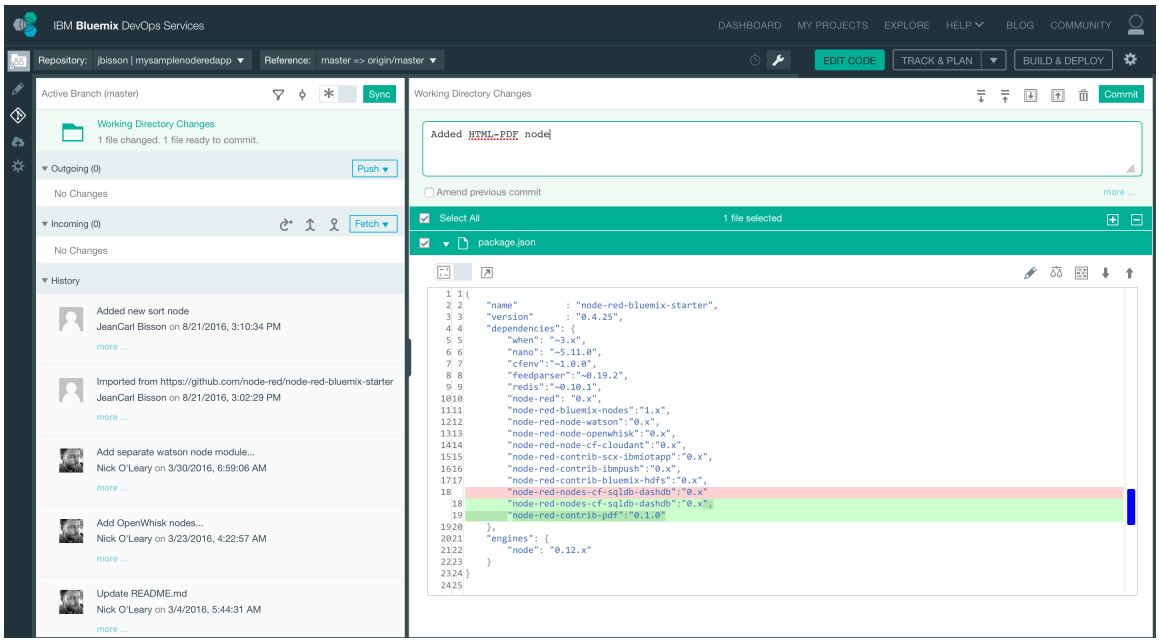

2. Returning to the IBM Bluemix DevOps Services editor, open up the `package.json` file and add the following line in the dependencies property.

```
"node-red-contrib-pdf":"0.1.0"
```

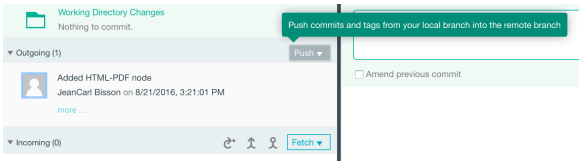


3. Switch to the Git view by clicking on the Git icon in the vertical toolbar in the left sidebar.

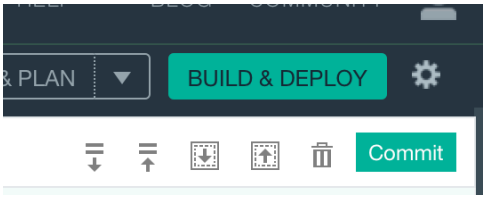

4. You'll see the uncommitted change we just made. Enter a commit message and click on **Commit**.



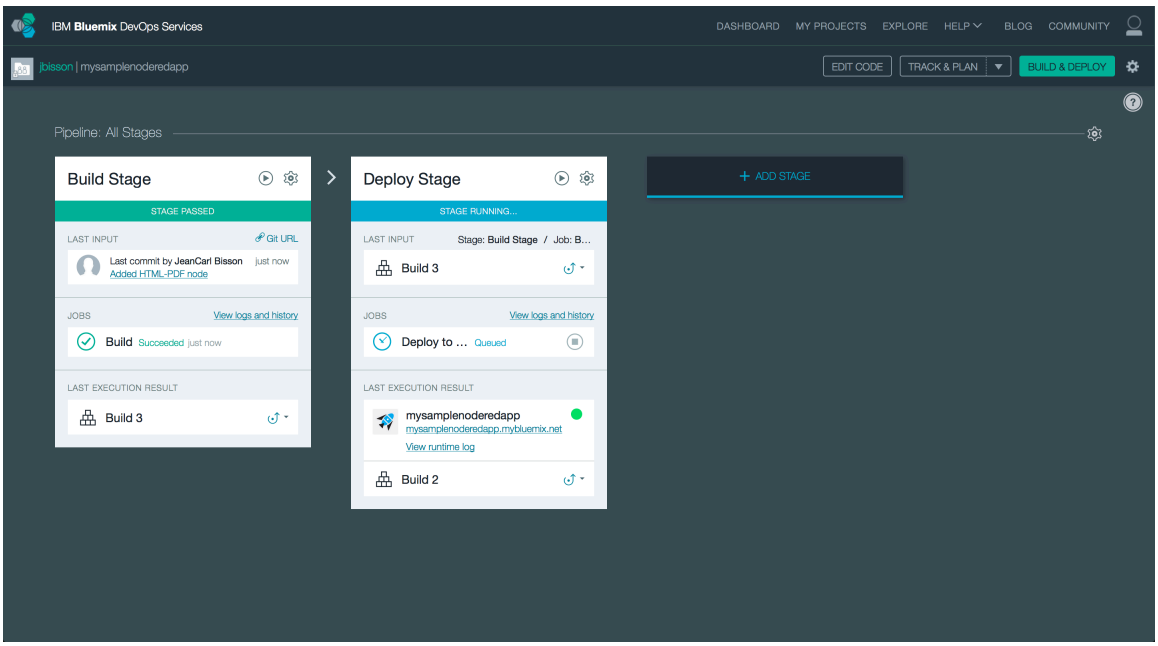

5. An entry will be added under the Outgoing section containing the commit. Click on **Push** to push the changes to the remote branch.



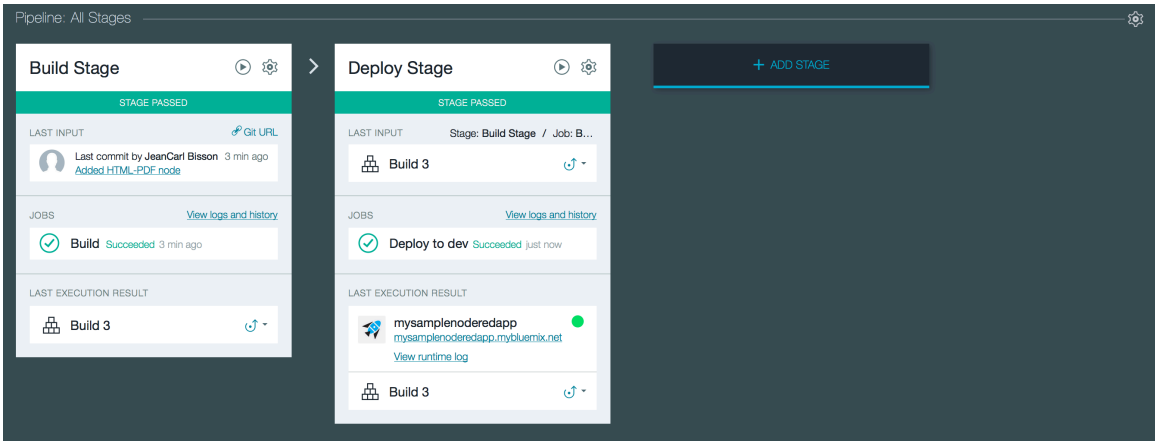

6. Click on the **Build & Deploy** button in the top-right corner.



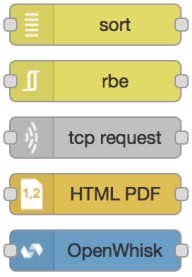
7. When new code is committed to this Git repo, the DevOps pipeline deploys the changes to the IBM Bluemix application that is linked. During the Deploy Stage, click on the **View logs and history** link to see what is happening behind the scenes.




8. When the application has finished being deployed, both stages should be green.



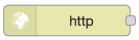

9. Return to your Node-RED application and refresh the editor. Scroll down to the section labeled function. A HTML PDF node should be displayed in the list.



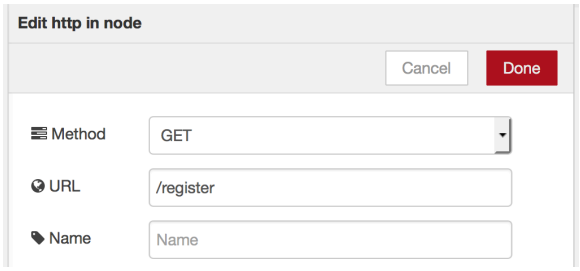

We'll use this node in the last section of this lab to generate a PDF.

# Global Context

Node-RED offers an in-memory global storage option called the global context. This can be useful when creating a variable that can be used in multiple flows and function nodes in an application and that persists for the lifecycle of the Node-RED application. In this section, we'll create a registration form to track people signing in at an event.

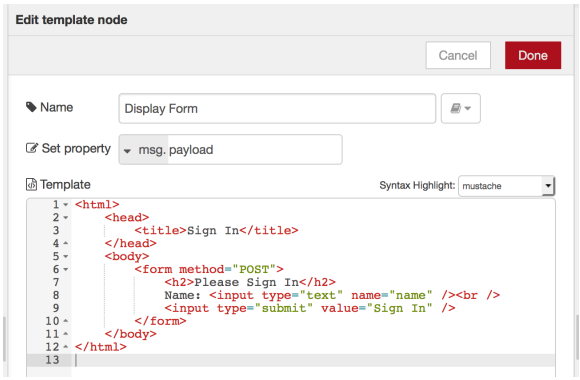
1. Add a 

node as shown below to expose this flow under the URL endpoint `/register`

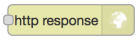



2. Add a 

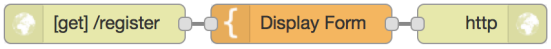
node as shown below. This code is available in the `code/3-display-form.html` file.

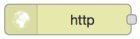



Get the code:
`ibm.biz/BdrgRX`

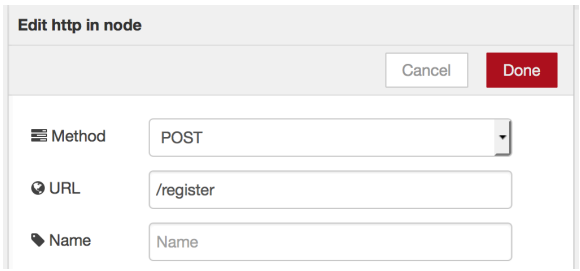
3. Add a 

node.
4. Connect the nodes together as shown below.



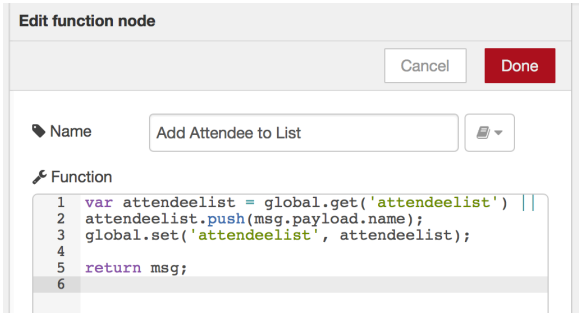

5. This will expose the `/register` endpoint to display the registration form. Let's create a second flow to handle the name when the form is submitted. Add a 

node as shown below.



6. Add a 

node as shown below. This code is available in the `code/4-add-attendee-to-list.js` file.

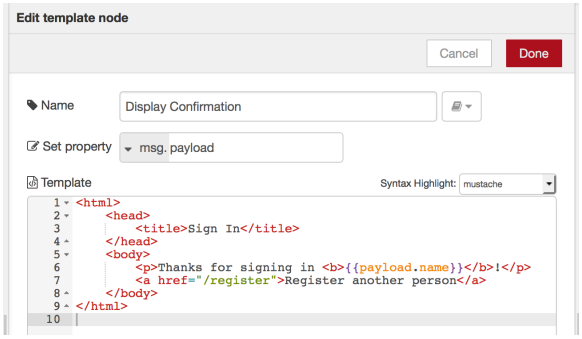



Get the code:
`ibm.biz/BdrgRi`

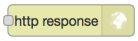
Using the get and set functions on the global object, we get the attendee list array (line #1), push the new name onto the array (line #2), and then set the attendee list back to the global object (line #3).

7. Add a 

node as shown below. This template creates a simple confirmation webpage displayed after the form is submitted. This code is available in the `code/5-display-confirmation.html` file.




Get the code:
`ibm.biz/BdrgRj`

8. Add a 

node.
9. Connect the nodes together as shown below.


10. Click on the red 

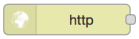
button in the top-right corner to save and deploy your changes.
11. Open a browser tab and visit your application's URL, appended with `/register`.

`http://myapp.mybluemix.net/register`

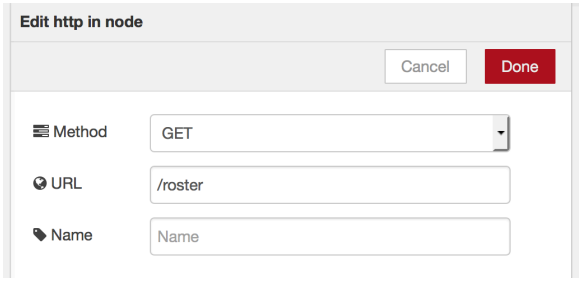

12. Enter a name and click **Sign In**. The following page shows the confirmation.

# Generate a Sorted PDF List

In this final section, we'll connect together the nodes we added to Node-RED to generate a PDF list of attendees in alphabetical order.

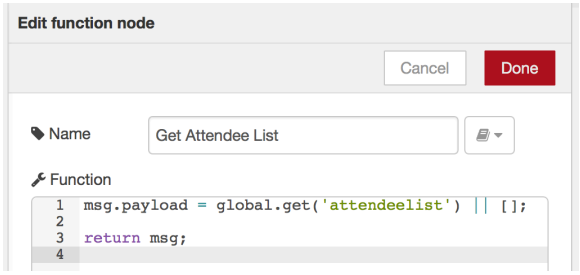
1. Add a 

node as shown below to expose the endpoint `/roster`.

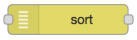



2. Add a 

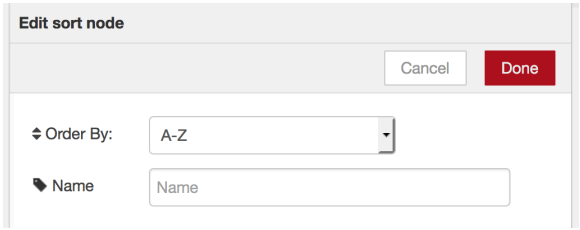
node as shown below. This will get the attendee list from the global context, or return an empty array if the global object doesn't have an attendee list. This code is available in the `code/ 6-get-attendee-list.js` file.




Get the code:
`ibm.biz/BdrgRZ`

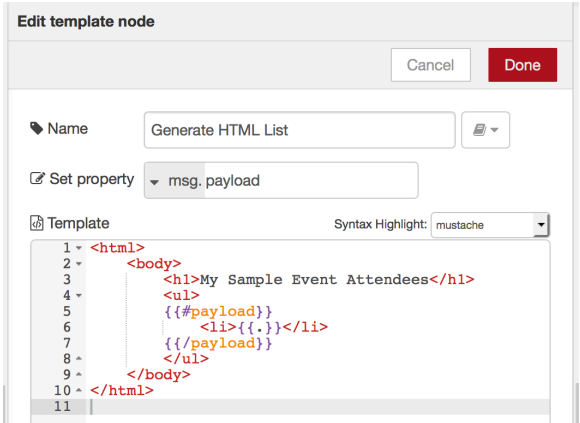
3. Add a 

node as shown below. This will sort the list of attendees in alphabetical order.




4. Add a 

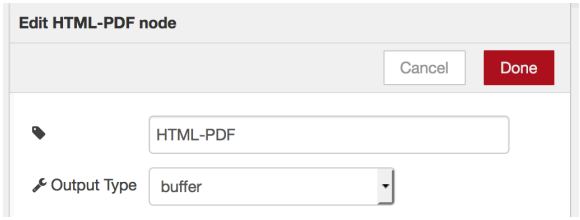
node as shown below. This template will iterate through the list of attendees and create a simple bulleted list. This code is available in the `code/7-generate-html-list.html` file.




Get the code:
`ibm.biz/BdrgRY`

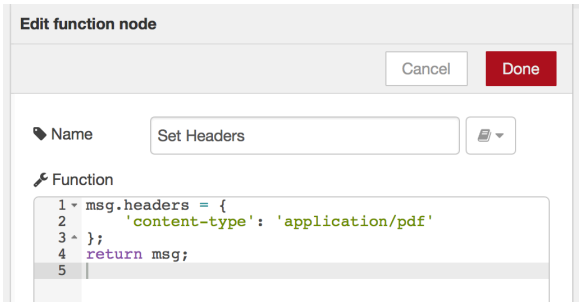
5. Add a  node as shown below.



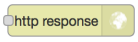



6. Add a  node as shown below. This function will output the proper HTTP response header to instruct the browser to display this content as a PDF. This code is available in the `code/8-set-headers.js` file.

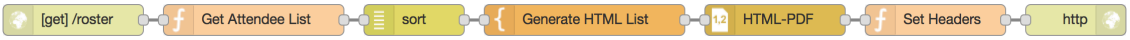





Get the code:
`ibm.biz/BdrgR2`

7. Add a  node. Connect the nodes together as shown below.

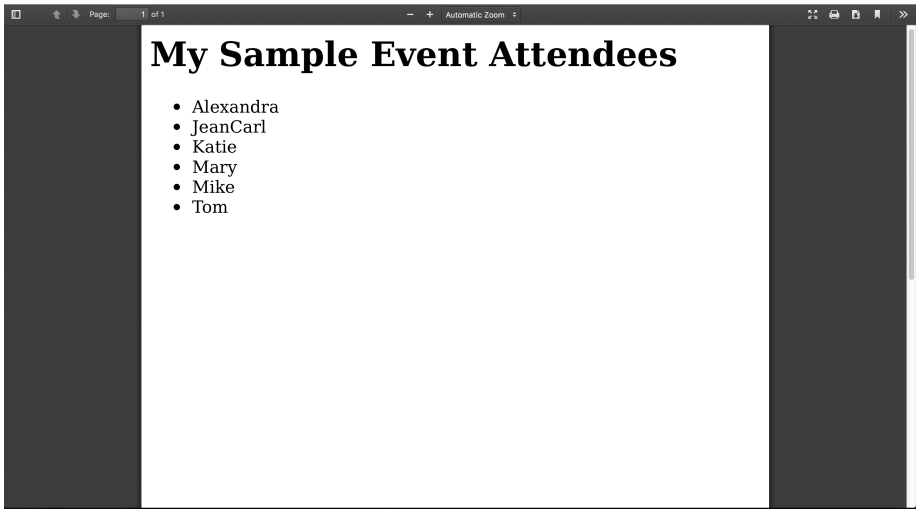





8. Click on the  button in the top-right corner to save and deploy your changes.
9. Open a browser tab and visit your application's URL, appended with `/roster`.



`http://myapp.mybluemix.net/roster`




In summary, we created a simple application to capture names, save them in our Node-RED application, retrieve the list of names, sort the list using our own sort node, and create a basic PDF using a node built by the Node-RED community.

But what happens if our application restarts? The list would likely be lost. Using a more persistent storage solution, such as a Cloudant NoSQL database, you might save the list to a JSON document and retrieve it when appending names to the list or displaying the list. This exercise is left up to the developer to implement. A solution is available at `ibm.biz/Bdrgb3`.