Creating Custom Activity in Journey Builder

Custom Activity: Custom Activities allows us to perform non-native actions, tailor the journey to the needs of anything we require and make direct connections to our system to send or retrieve data, or to make decisions based on that specific data.

Custom Journey Builder activities provide businesses with a powerful tool to extend the functionality of Journey Builder and automate complex workflows in a personalized manner, ultimately leading to better customer engagement and satisfaction.

Journey Builder in Marketing cloud provides certain functionalities or activities which supports solutions to most of the industries but not all. In that case, Marketing cloud allows to build the complex solution outside of ecosystem and can be interacted with Journey Builder to achieve the desired business use cases.

Creation of a Custom Activity involves the following components:

- **index.html**: Used to load the screen used to configure the activity.
- **config.json**: Main configuration file that defines the metadata and other options about the activity.
- **activity.js**: File with main functionality for index.html.
- **postmonger.js**: JS plugin to make it easier to interact between iframes in two different domains.
- **icon.png**

index.html:

index.html > html > body

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Custom Activity for Journey Builder</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>

    <h3>Select an option:</h3>
    <select id="dropdown">
        <option value="option1">Code 1</option>
        <option value="option2">Code 2</option>
        <option value="option3">Code 3</option>
    </select>

    <div>
        <button id="submitBtn">Submit</button>
        <button id="cancelBtn">Cancel</button>
    </div>

    <script src="postmonger.js"></script>
    <script src="activity.js"></script>

</body>
</html>

```

activity.js:

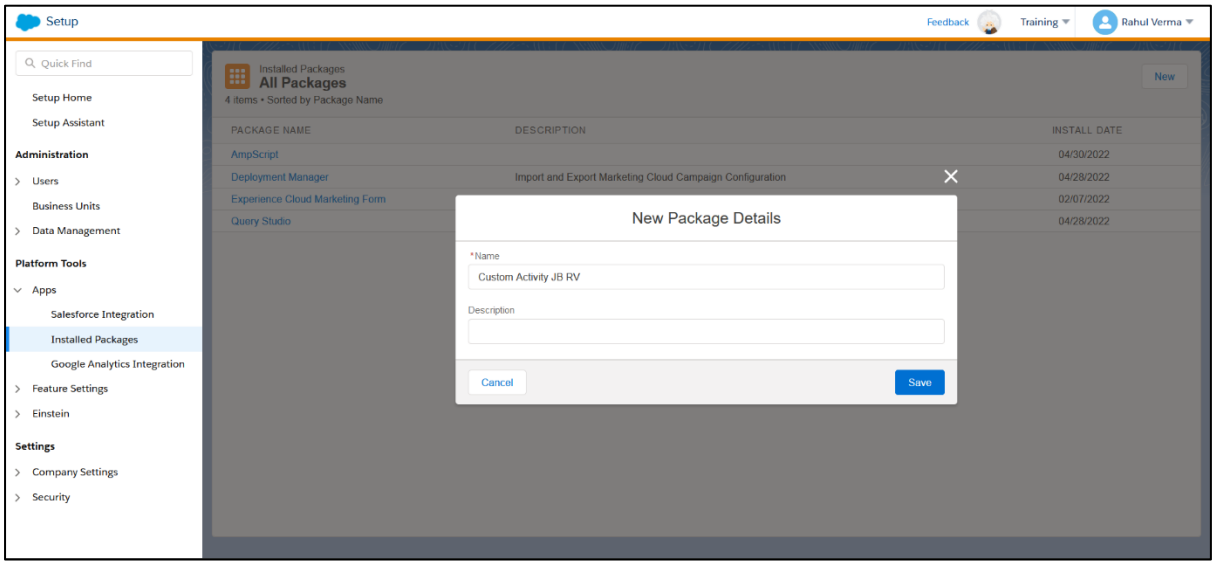
```
JS activity.js > ...
1   var connection = new Postmonger.Session();
2
3   connection.trigger('ready');
4
5   connection.on('initActivity', function(data) {
6       document.getElementById('configuration').value = JSON.stringify(data, null, 2);
7   });
8
9   connection.on('clickedNext', function() {
10      var configuration = JSON.parse(document.getElementById('configuration').value);
11      connection.trigger('updateActivity', configuration);
12  });
13
14  document.getElementById('submitBtn').addEventListener('click', function() {
15      var configuration = JSON.parse(document.getElementById('configuration').value);
16      connection.trigger('updateActivity', configuration);
17  });
18
19  document.getElementById('cancelBtn').addEventListener('click', function() {
20      connection.trigger('destroyActivity');
21  });
22
```

config.json:

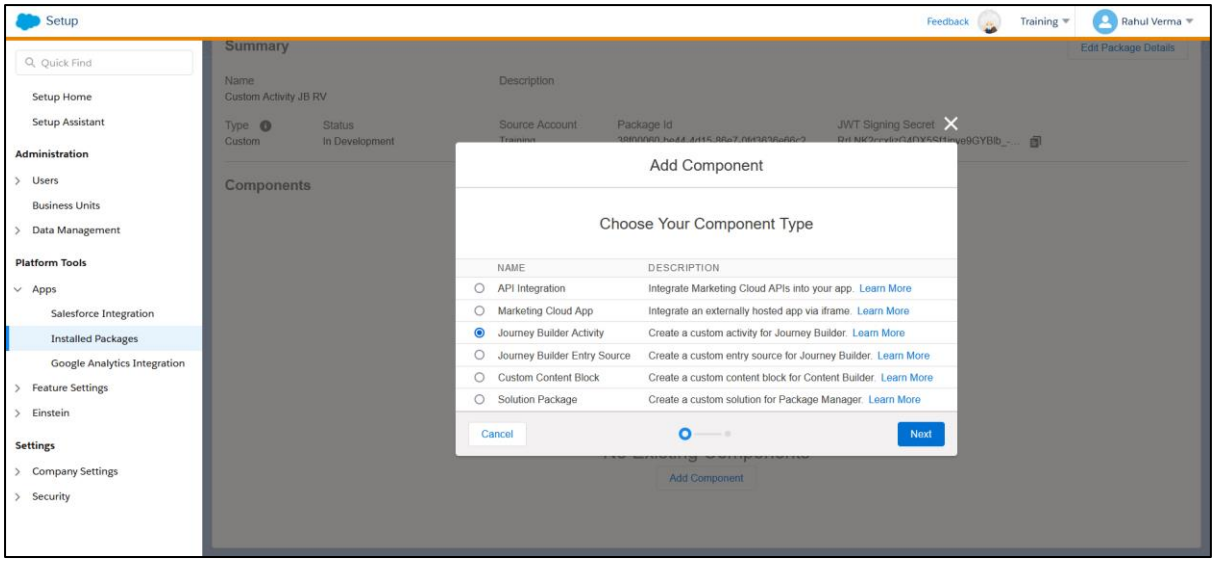
```
1
2   {
3       "workflowApiVersion": "1.1",
4       "metaData": {
5           "icon": "icon.png",
6           "category": "message"
7       },
8       "type": "REST",
9       "lang": {
10          "en-US": {
11              "name": "Custom Activity for Journey Builder",
12              "description": "A custom activity using REST API."
13          }
14      },
15      "arguments": {
16          "execute": {
17              "inArguments": [],
18              "outArguments": []
19          }
20      },
21      "userInterfaces": {
22          "configInspector": {
23              "size": "medium"
24          }
25      },
26      "configurationArguments": {
27          "execute": {
28              "url": "https://customactivity-jb-sfmc.onrender.com/URI/for/your/activity/execute"
29          },
30          "publish": {
31              "url": "https://customactivity-jb-sfmc.onrender.com/URI/for/your/activity/publish"
32          }
33      },
34      "schema": {
35          "arguments": {
36              "execute": {
37                  "inArguments": [],
38                  "outArguments": []
39              }
40          }
41      }
42  }
43
```

Setting up Custom Activity Package in SFMC involves the following steps:

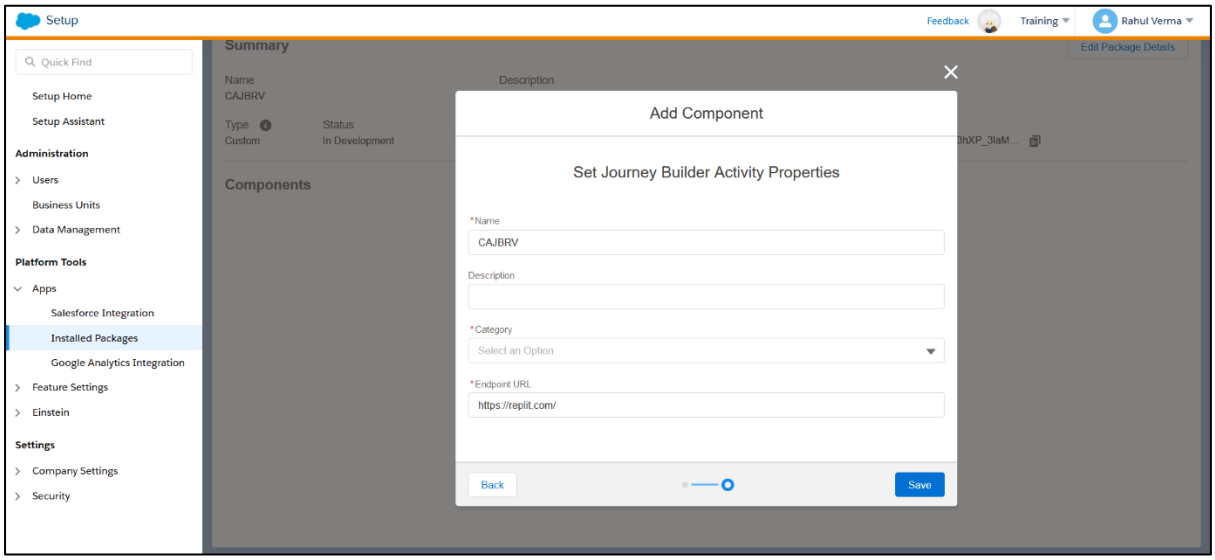
Step 1: In the first step, on the SFMC website we go to the top right corner and click on the “Name”. Then we click on “Setup”, which will open the setup page. On the left side of the screen, we go under the “Platform Tools section” and click on “Installed Packages” where in we get to see all the installed packages on the account. We go to the top right corner and click on “New”; we will get a pop-up as shown below, here we will enter the name we want and the description if required.



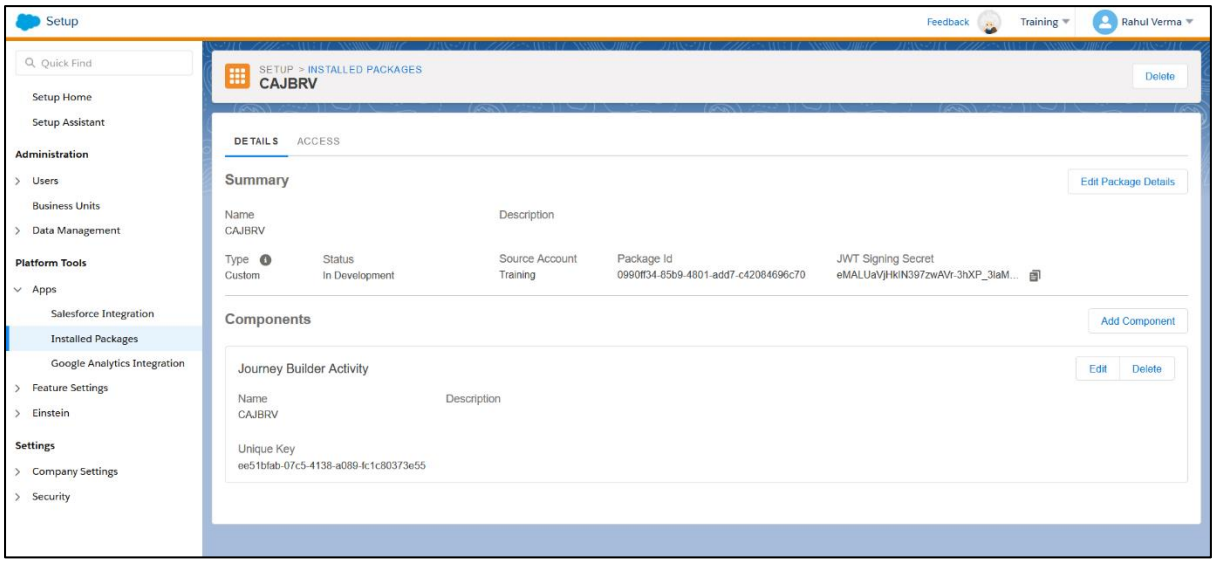
Step 2: After it is created successfully, we can add components as shown at the bottom of the page. We then click on “Add Component”, after which we get a pop-up as shown below. Here we select “Journey Builder Activity” as we are creating a custom Journey Builder activity and then we click “Next”.



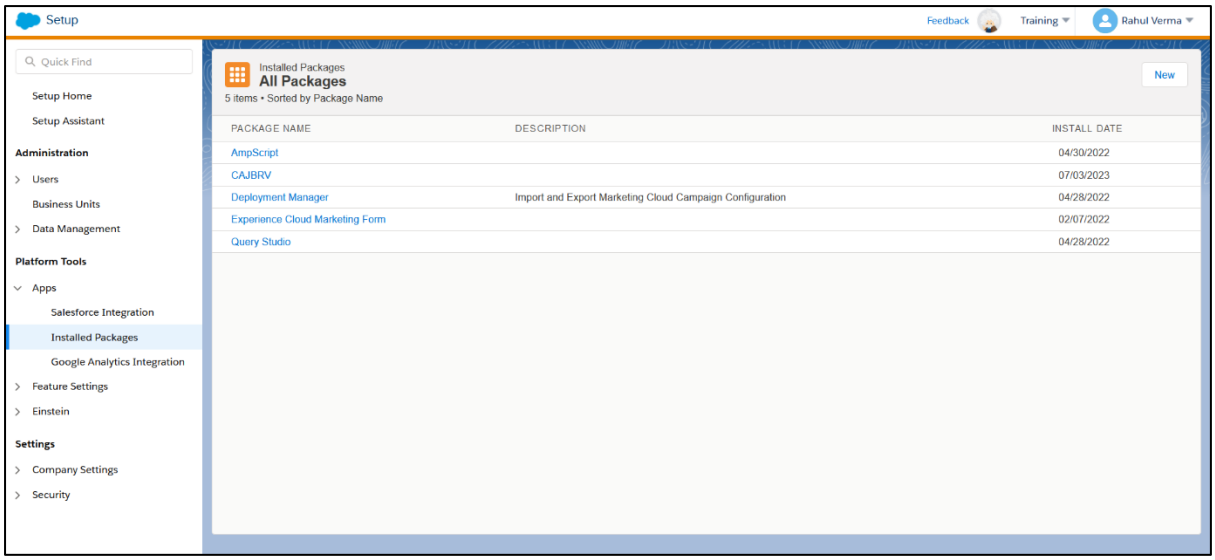
Step 3: In this set we get to set the Custom Journey Builder Activity Properties, we will enter the "Name", "Category", "Endpoint URL" and "Description" if required. We then click "Save".



Step 4: Here in this step, we see all the details we entered in the "Components" section of the page.

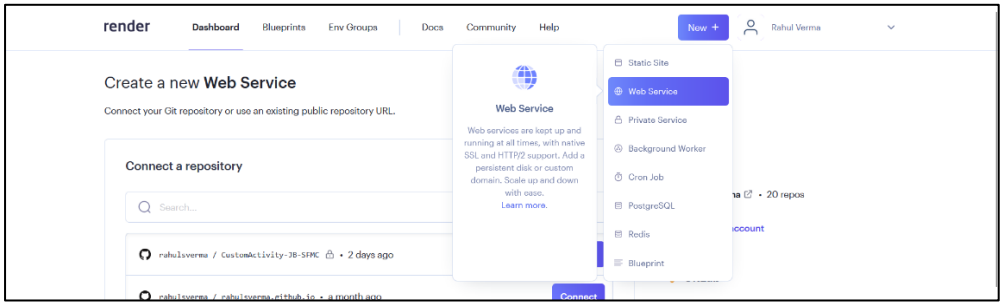


Step 5: Final step shows the Package we have created in the “Installed Packages” section.



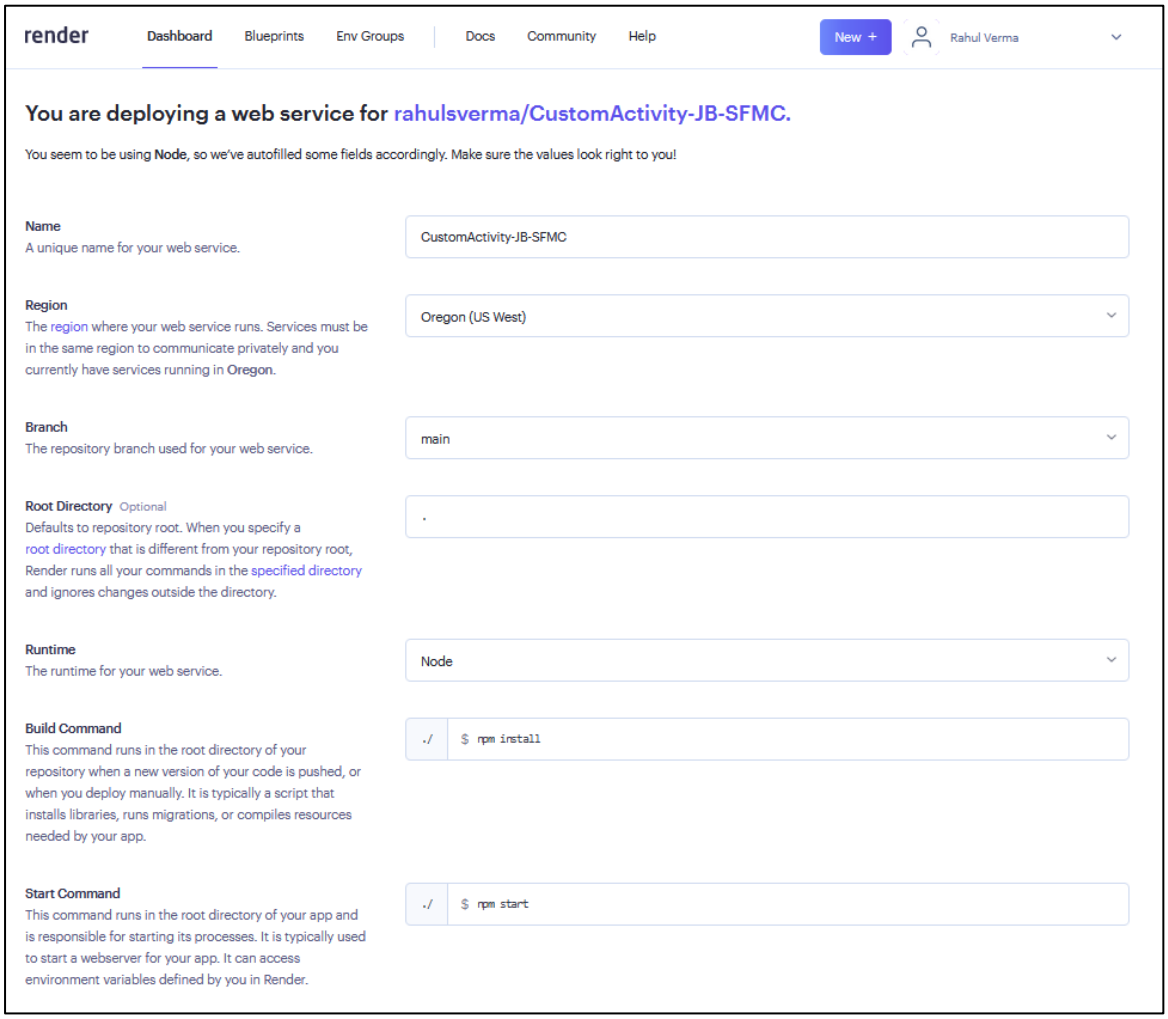
We have used render.com to host the project, for hosting we will follow the following process:

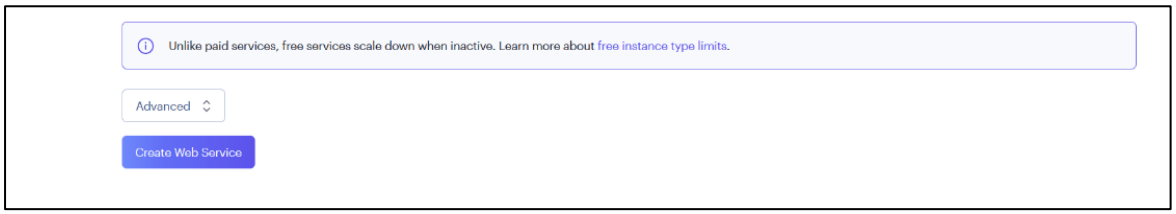
Step 1: The first step is to go to https://render.com/, sign up and navigate to the Dashboard. On the top-right corner of the page we can see a “New” button as shown below. After we click on this we get a drop-down menu in which we will select “Web Service”.



Step 2: In the second step, we are redirected to a new page where we need to connect render.com to the Github repository where we have pushed all the data and the code. After connecting to the Github account, we can connect to the specific repository as shown below.

Step 3: After connecting the Github repository we will be redirected to a new page where we will provide all the required settings before the page is deployed. Setting we need to focus on is the Name which should be unique, after that the Region (closer is better), Root Directory should be ".", Runtime is "Node", Build Command should be "npm install" and the Start Command is "npm start". We select the Instance Type "Free" and click on "Create Web Service" as shown below.





Step 4: After this the process is started and the deployment is in progress, we can see the service is live once the page is successfully hosted.