Exercise 16

Creating and executing a BPMN flow

Prior Knowledge
BPMN Lecture

Objectives
Understand the basics of the BPMN specification, and be able to create and execute a business process using the BPMN tooling using Camunda. Deploy the BPMN into the Camunda runtime and be able to track instances etc.

Software Requirements
- Camunda BPMN Modeler 4.6.0
- Camunda BPMN runtime 7.14.0

Steps.

1. Make sure nothing is running on port 8080 by killing any old servers.

   (techie hint:
   `sudo lsof -n -P -i | grep 8080`
)

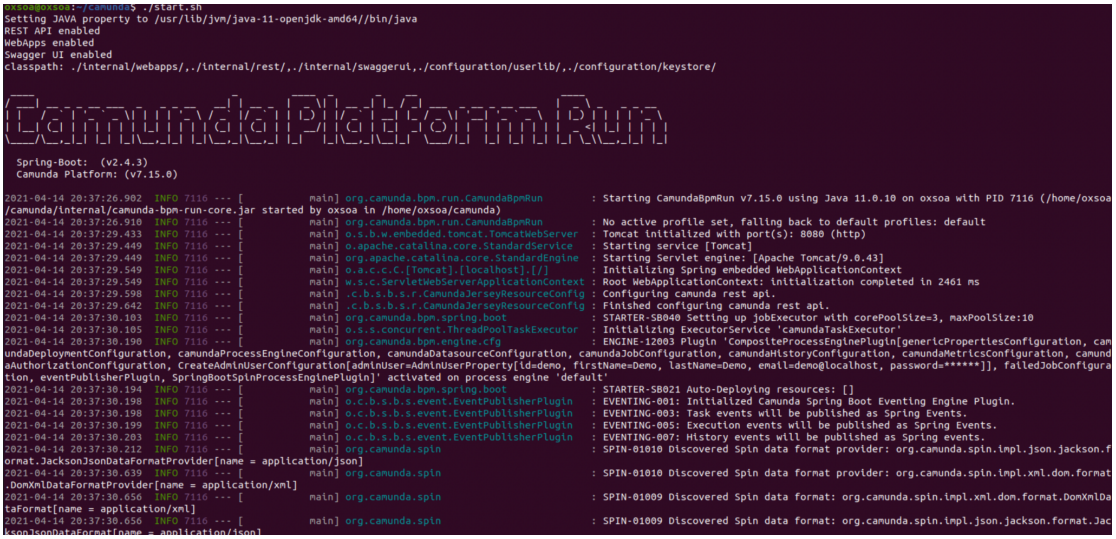
2. Let's get our project initiated.

   First let's install the Camunda Run runtime:

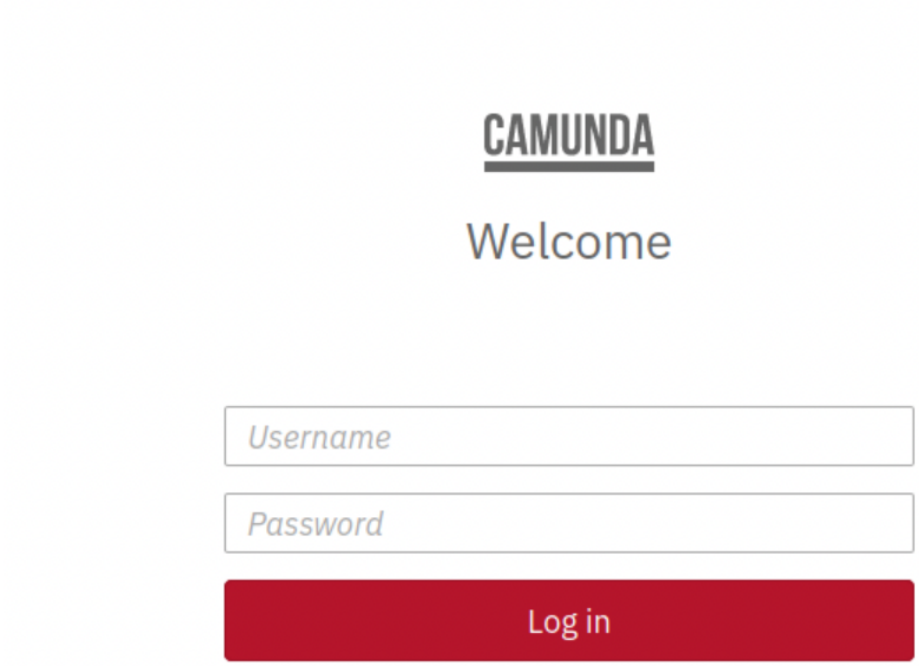
   ```
   cd ~/Downloads
   wget
   https://downloads.camunda.cloud/release/camunda-bpm/run/7.15/camunda-bpm-run-7.15.0.zip
   wget
   https://downloads.camunda.cloud/release/camunda-modeler/4.7.0/camunda-modeler-4.7.0-linux-x64.tar.gz

   cd ~
   mkdir ~/camunda
   cd camunda
   unzip ~/Downloads/camunda-bpm-run-7.15.0.zip
   ./start.sh
   ```

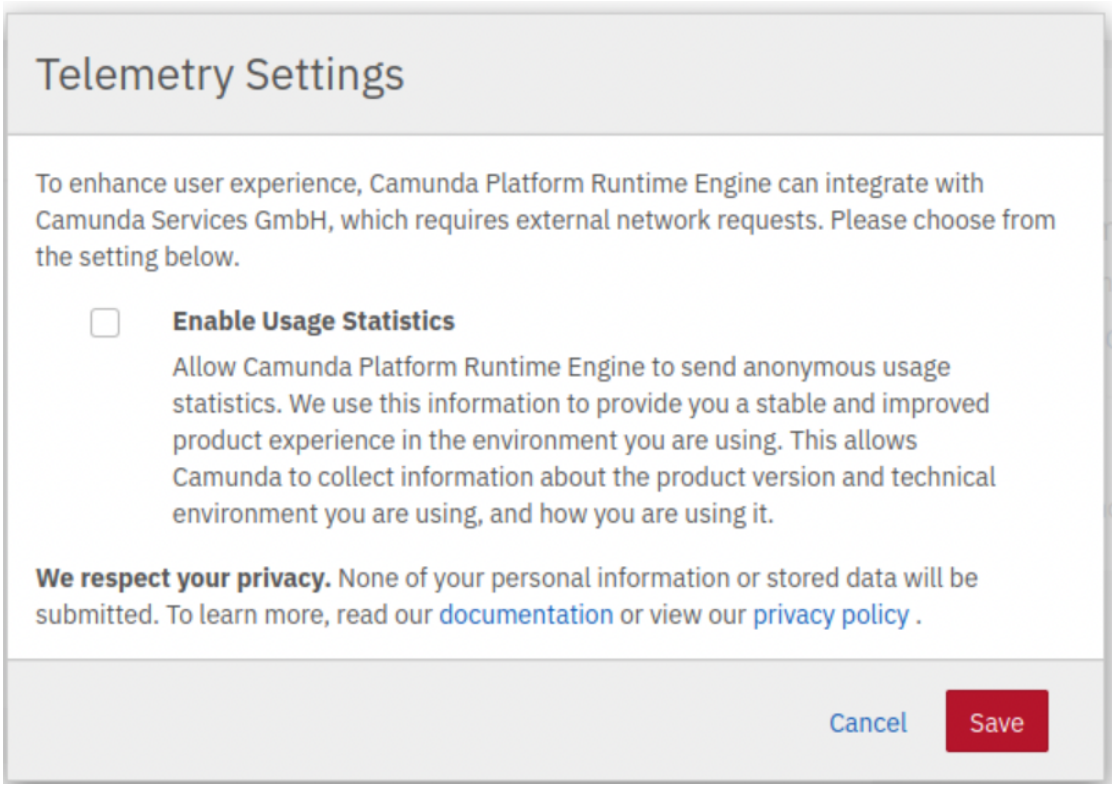
3. You should see something like:



4. Now open http://localhost:8080

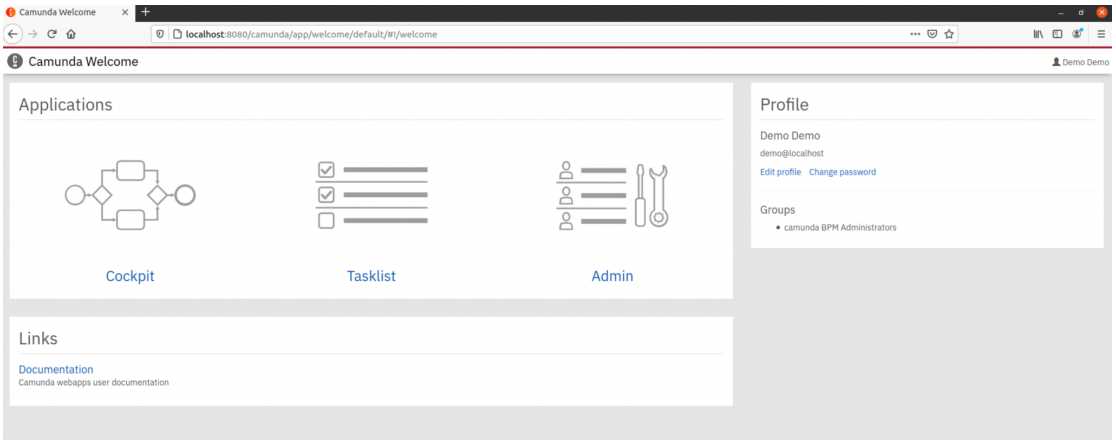


5. Login as demo/demo



Save

6. You should see:



We will come back to look at these in a minute.

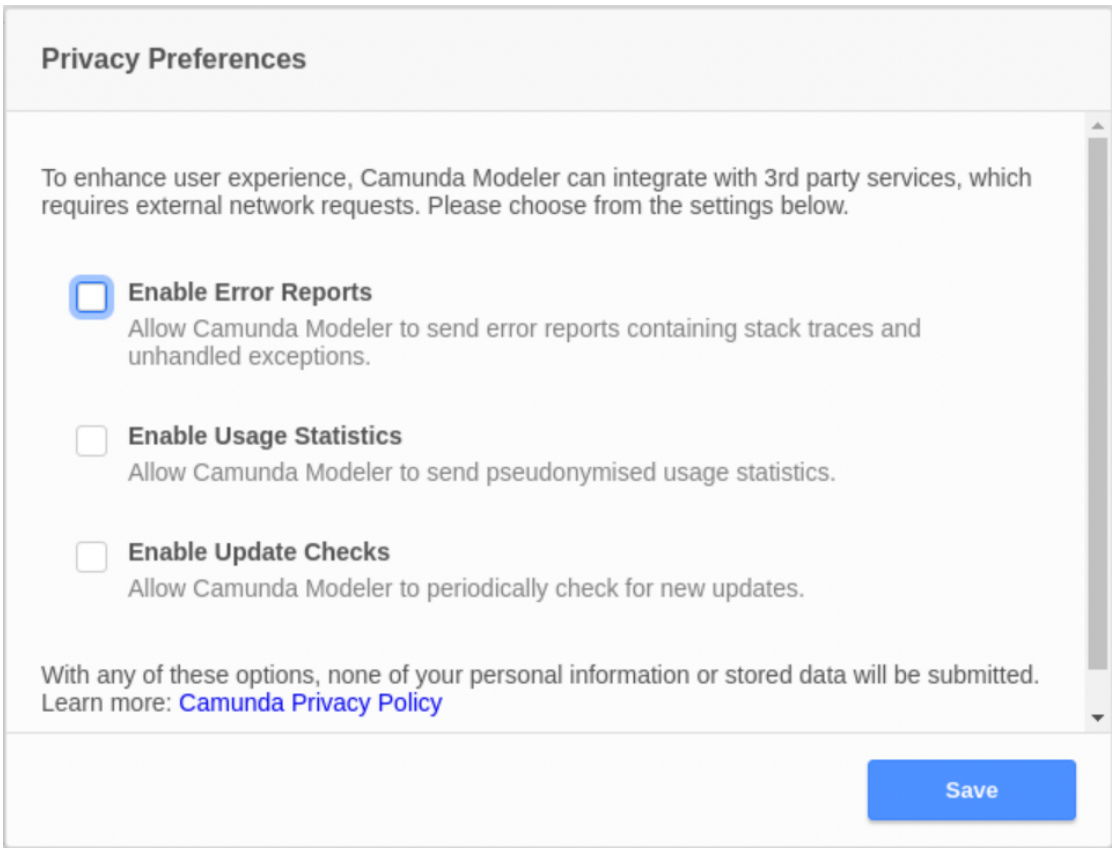
7. Start a new terminal window.

   Make a place for our BPMN files:

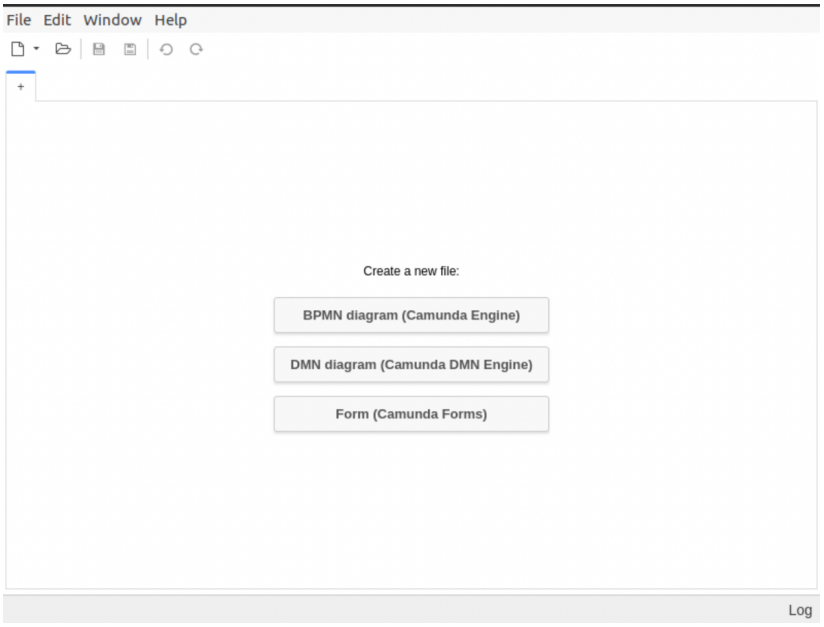
   ```
   mkdir ~/bpmn
   ```

8. Now let's untar and start up our modeling tool:

```
cd ~
tar xvzf
~/Downloads/camunda-modeler-4.7.0-linux-x64.tar.gz
cd camunda-modeler-4.7.0-linux-x64/
./camunda-modeler
```

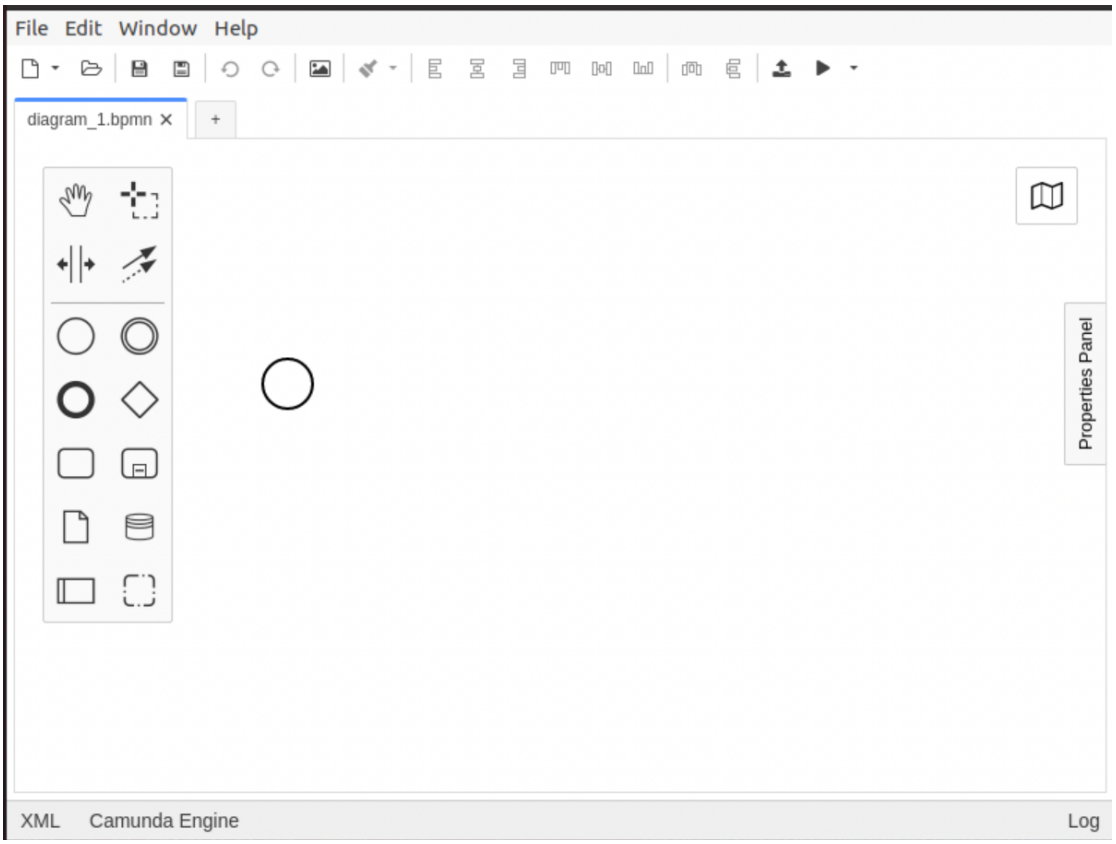


9. Click Save

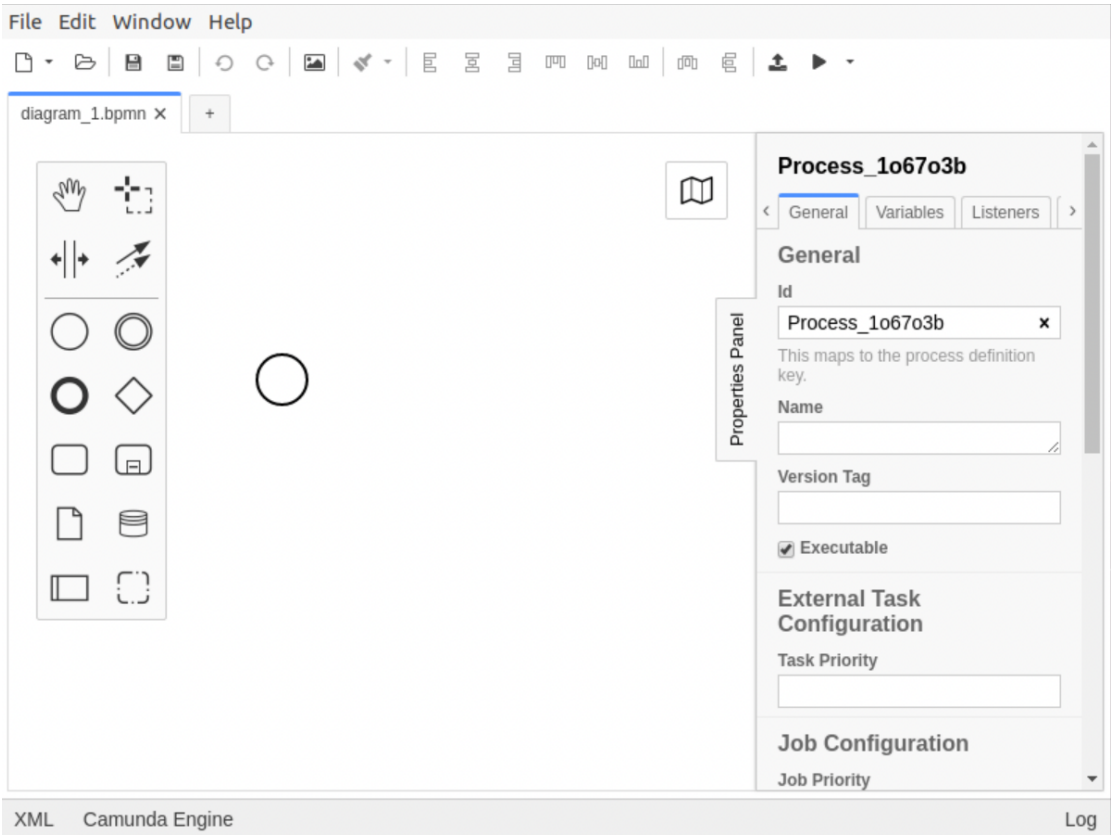


10. Click on Create a BPMN diagram (Camunda Engine).

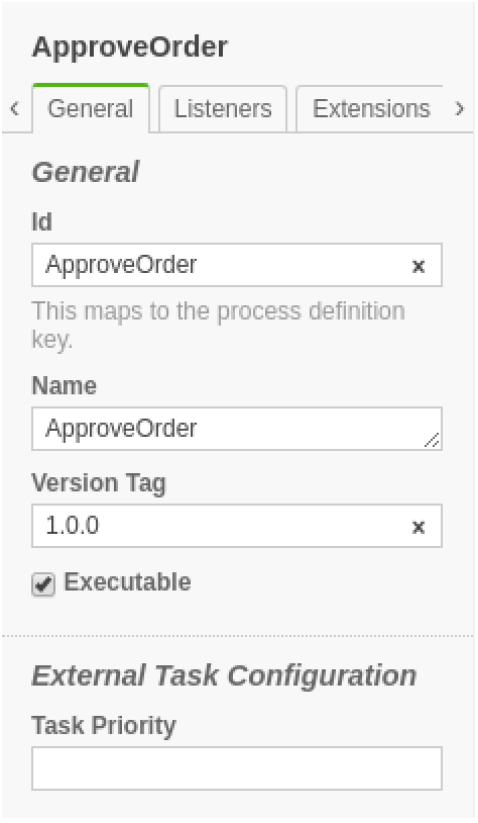
11. You should see:



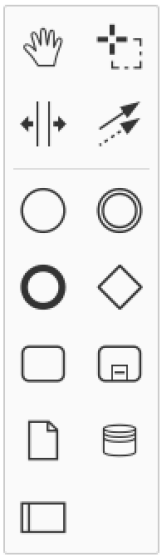
12. Make sure the properties panel is expanded.



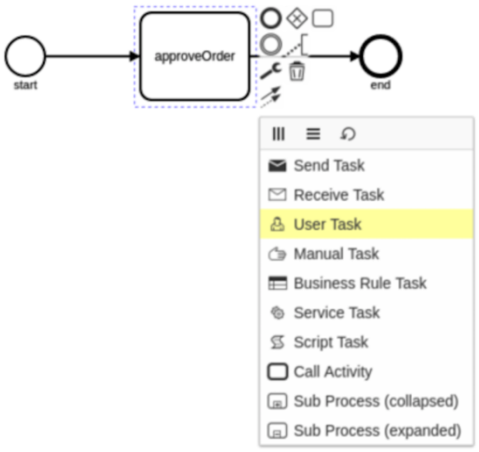
13. Make sure the **Executable** tag is ticked, and change the process id and name to be **ApproveOrder.** Change the version number to be 1.0.0:



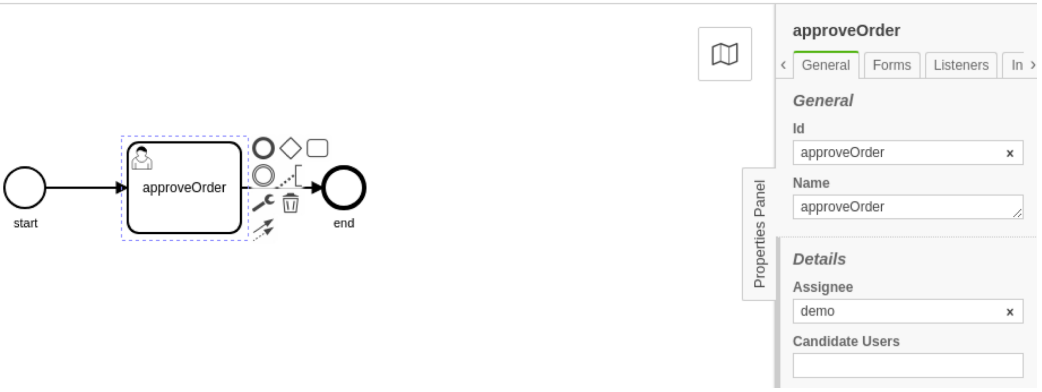
14. Use the tool to draw a simple process like this. Make the **id** of each object match the name (e.g start/start).



15. In order to make the task into a User Task (with the little "man" icon), click on the spanner/wrench and choose User Task.

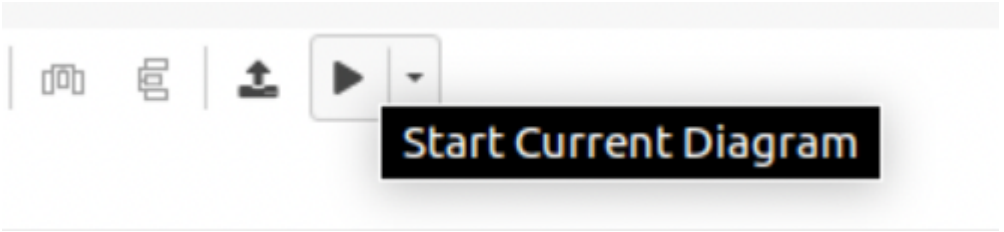


16. Edit the properties for the Approve Order Task:
    Id: **approveOrder**
    Name: **approveOrder**
    Assignee: **demo**



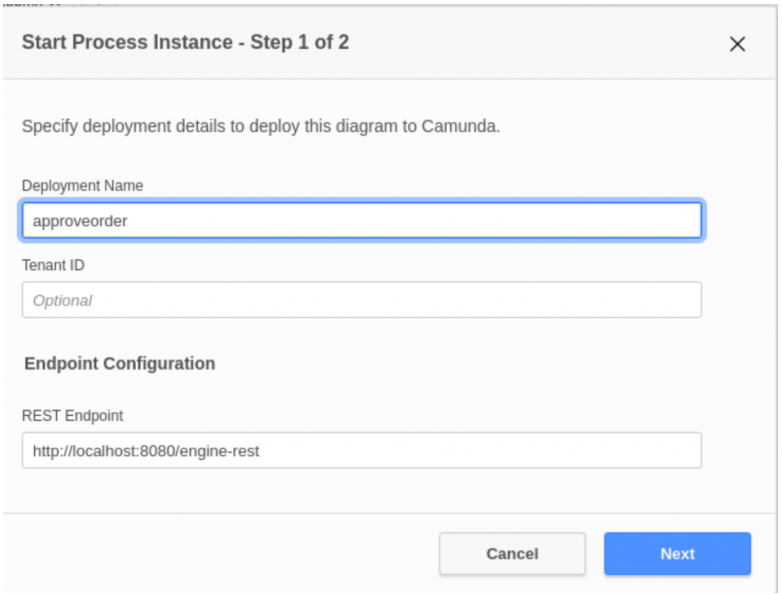
17. Save the file as ~/bpmn/approveorder.bpmn

18. Now you can test your process:



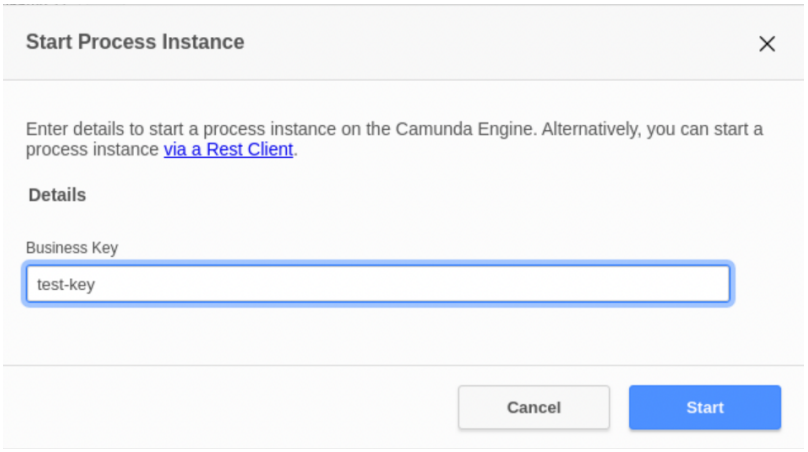
Click on the Play icon.

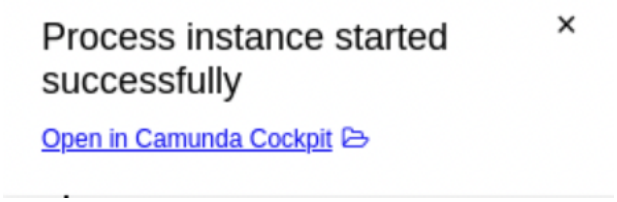
You should see:



Click **Next**

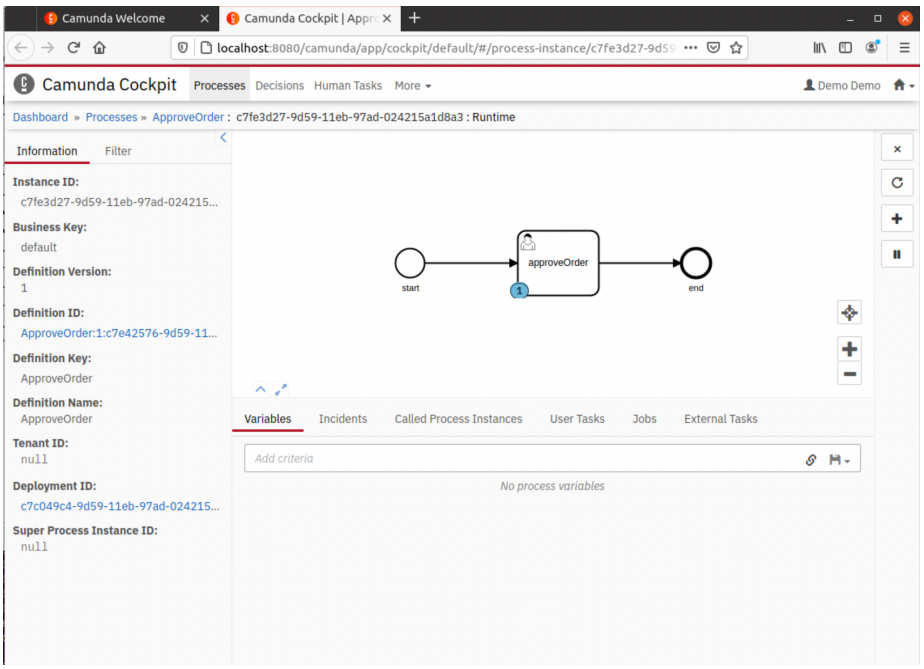
Enter a test key:





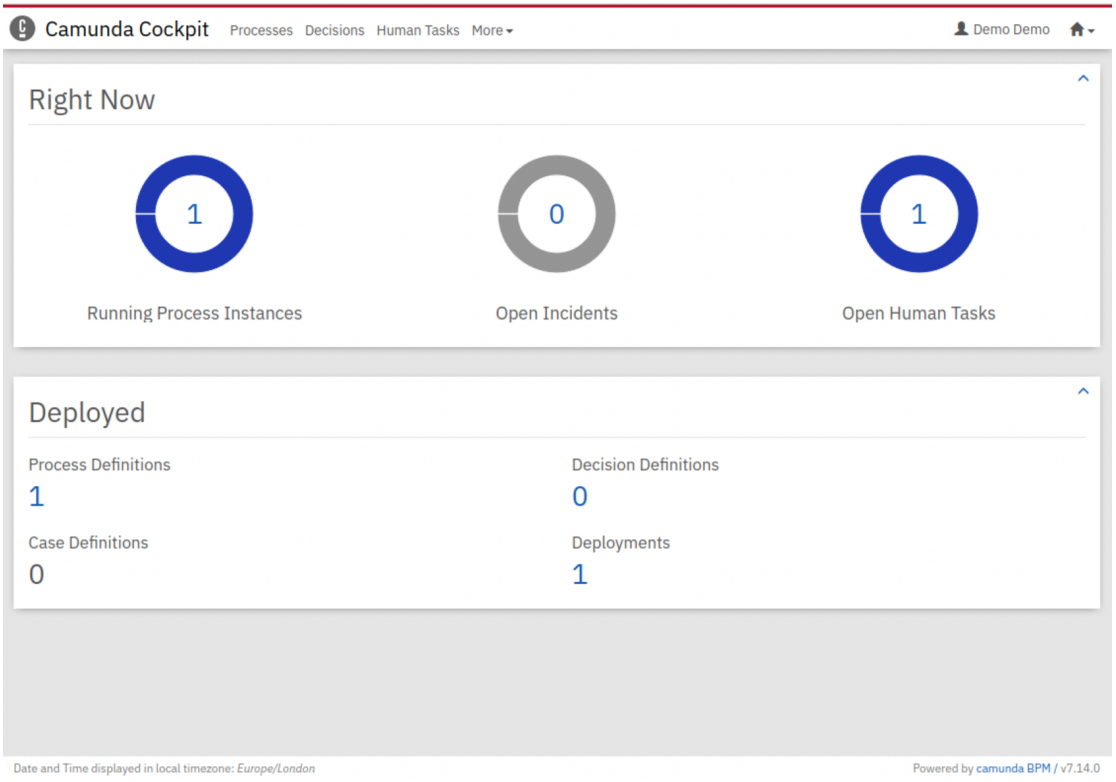
Click on the link.

You should see the same process in the runtime now:



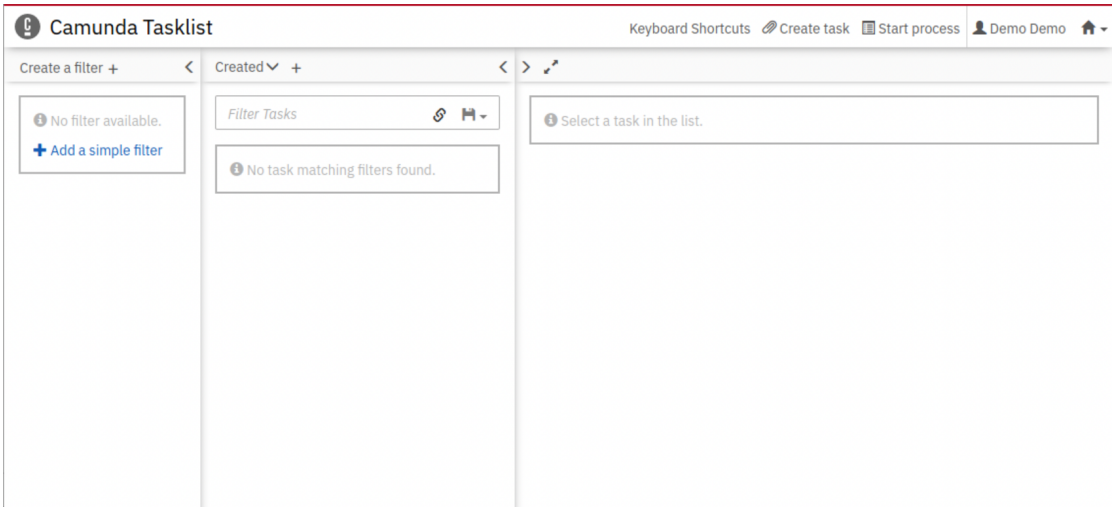
19. Click on the "**Camunda Cockpit**" words in the corner:

20. You should see something similar to this:

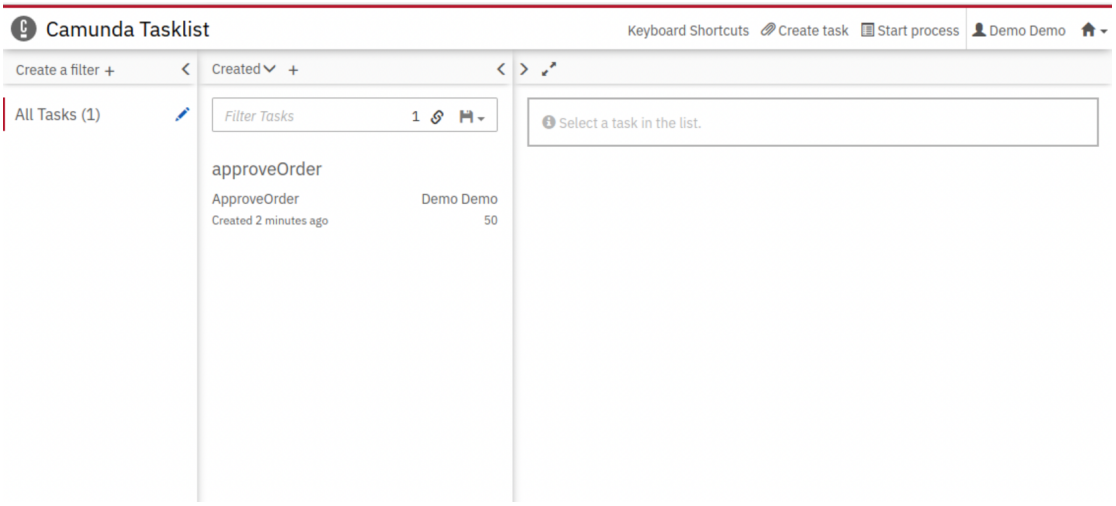


21. Select the little Home icon in the corner and choose **Tasklist**

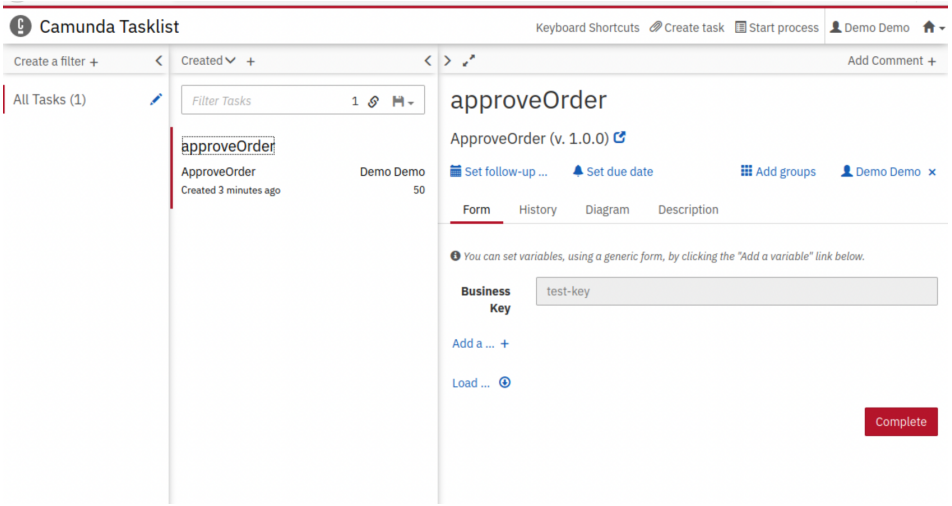
22. Now you should see something like this:



23. Click **Add a simple filter**



24. Click on the "approveOrder" task in the Created column



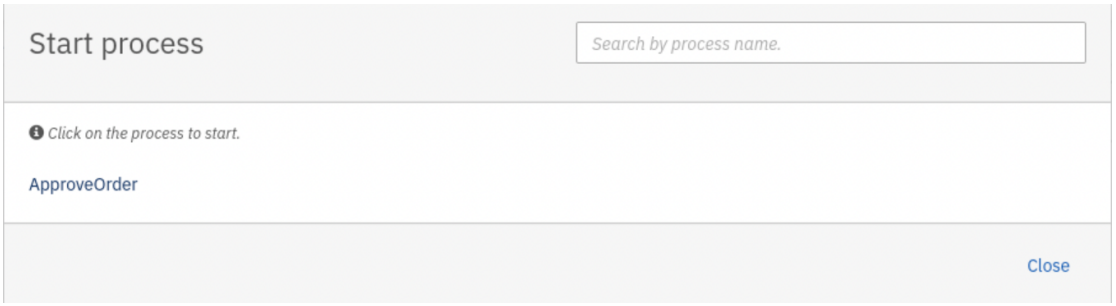
25. Click on the **Diagram** tab
    You will see the process you designed, now with the current step highlighted:

26. Go back to the **Form** tab and click Complete. This will "complete" this instance of the process.

27. This process works, but let's be honest, it is almost pointless.

28. We can repeat this from the UI as well as initiating this from the tool.

29. Click on **Start Process**

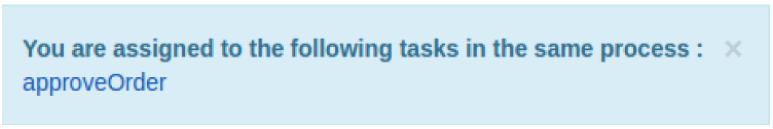


30. Choose **ApproveOrder**

31. Enter anything you like in Business Key and then click **Start**

32. Do you remember where you set the assignee for the User Task in the BPMN process to **demo**? Well this has just happened and you are logged in as demo and hence the portal has popped up a message saying you've

been assigned a task to work on:



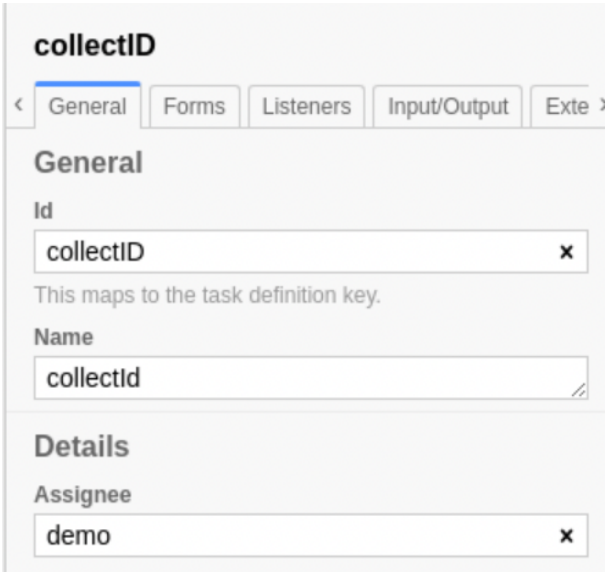
Repeat to approve this order as well.

33. To make the process more interesting we need to tie it in with our existing order processing system. A simple way to do that would be to take an order ID as input, and then gather information about that order to put in front of the approver.

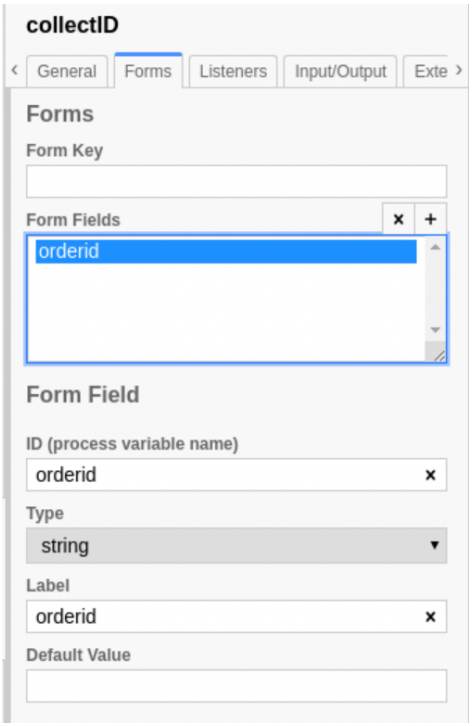
34. Firstly, let's create a more complex process. This will have three tasks. The first will collect an orderID (basically the UUID from the purchase service). The second will call out to Purchase to get info. The third will review that information with the user.

35. Delete the previous diagram elements and create these three. The first and last task are **User Tasks,** and the middle one is a **Service Task.**

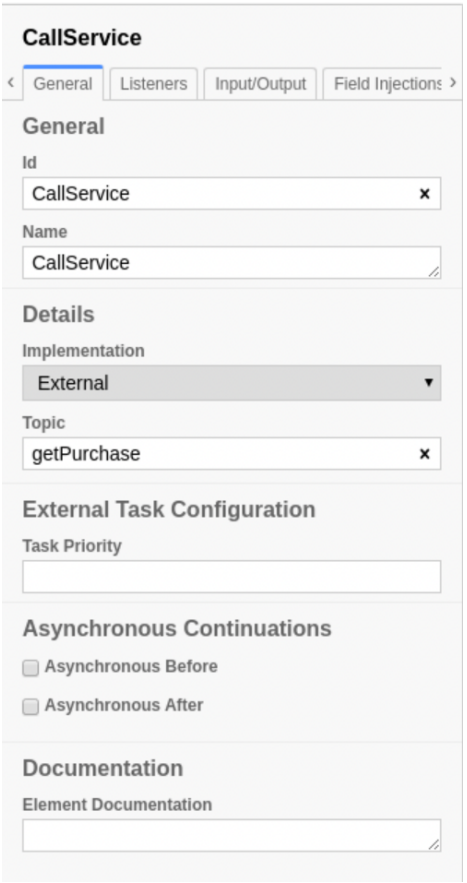
36. Configure the the collectID User Task like this:



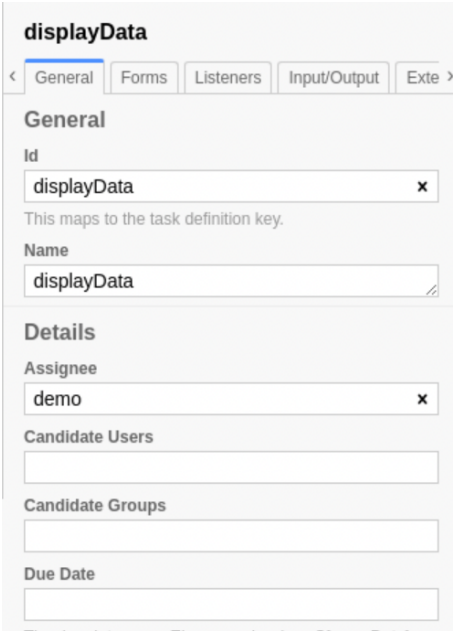
37. Add a Form Field called **orderid** (this name matters!)



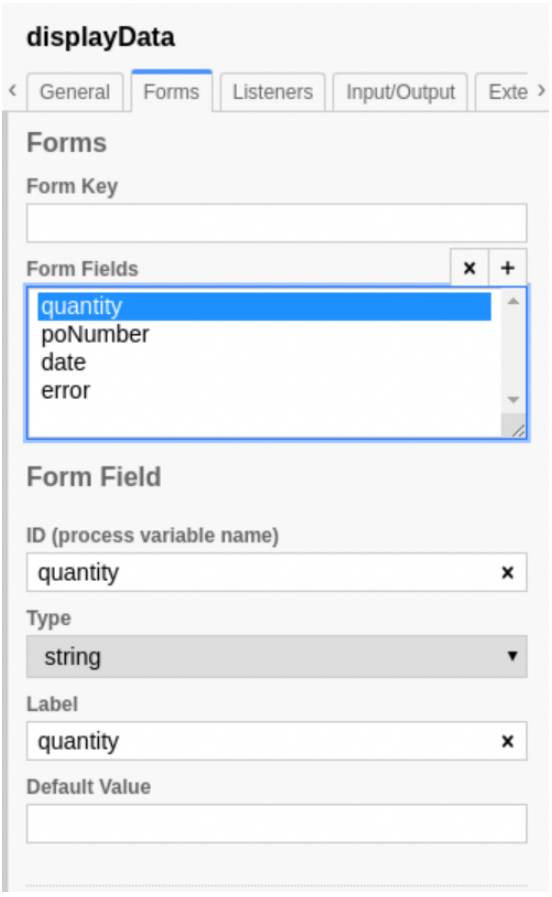
38. Configure the CallService task like this:



39. Configure the displayData task like this:



40. And create form fields:



Each form field is a string, with the ID and label the same. The names and cases matter!

41. Now we should be ready to run this. **Save the BPMN.**

42. We need to handle the external service. This is done with a node.js service that polls Camunda via the REST interface for work, does the work, and then returns the result.

In a new terminal:

```
cd ~
git clone https://github.com/pzfreo/camunda-node.git
cd camunda-node
yarn install
node index.js
```

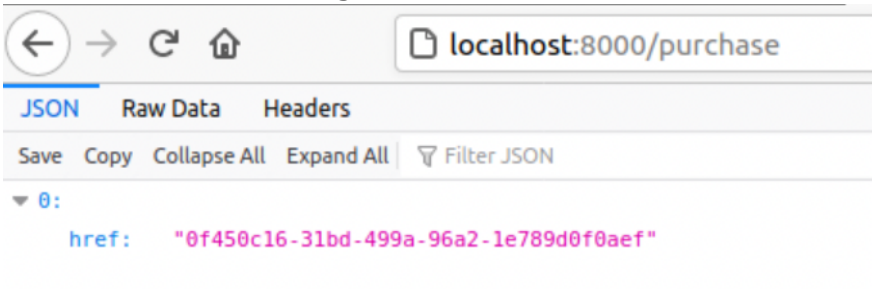
The code is here:

Users > paul > repos > camunda-node > JS index.js > client.subscribe("getPurchase") callback

```
1   const { Client, logger, Variables } = require("camunda-external-task-client-js");
2   const axios = require("axios");
3
4   const config = { baseUrl: "http://localhost:8080/engine-rest", use: logger };
5
6   // create a Client instance with custom configuration
7   const client = new Client(config);
8
9   client.subscribe("getPurchase", async function({ task, taskService }) {
10    const orderid = task.variables.get("orderid");
11    const processVariables = new Variables();
12    const localVariables = new Variables();
13    try {
14      // call the purchase service using orderid
15      const response = await axios.get(`http://localhost:8000/purchase/${orderid}`);
16      if (response.status == 200) {
17          console.log("found");
18          console.log(response.data);
19          processVariables.set("poNumber", response.data.poNumber);
20          processVariables.set("quantity", response.data.quantity);
21          processVariables.set("customerNumber", response.data.customerNumber);
22          processVariables.set("lineItem", response.data.lineItem);
23          processVariables.set("paymentReference", response.data.paymentReference);
24          processVariables.set("date", response.data.date);
25          processVariables.set("error", "none");
26      }
27      else {
28          processVariables.set("error", "orderid not found");
29      }
30    } catch (error) {
31          processVariables.set("error", "server not available");
32    }
33
34    await taskService.complete(task, processVariables, localVariables);
35  });
```

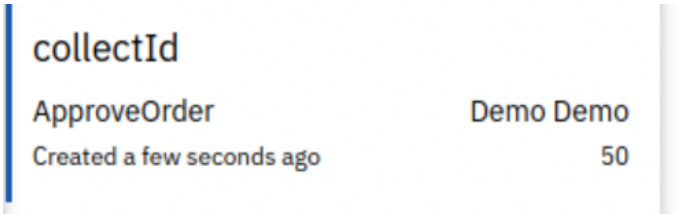
43. Make sure that your backend service from previous exercises is running on port 8000

44. Browse to http://localhost:8000/purchase
    You should see something like:

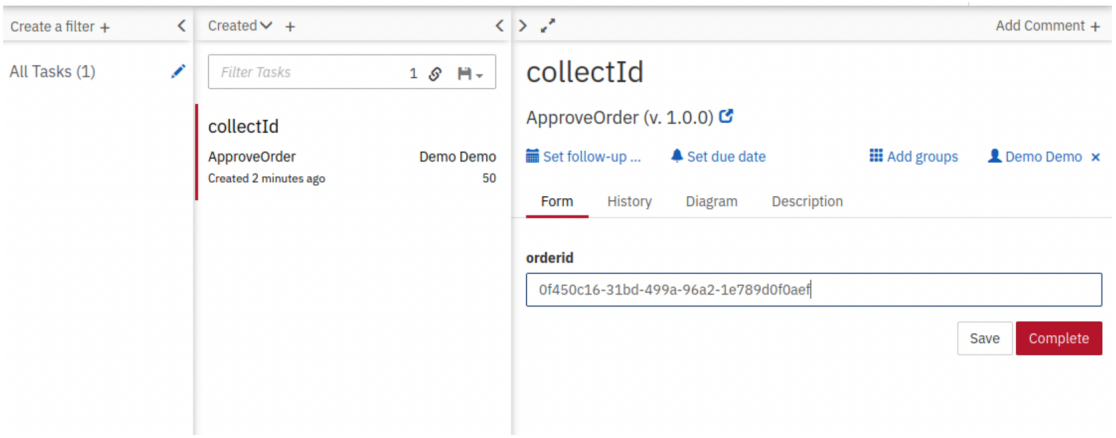


45. Keep this window open as you are going to need that order uuid.

46. Use the play button on the Camunda Modeler to resubmit your BPMN process. Go to the **Tasklist:**

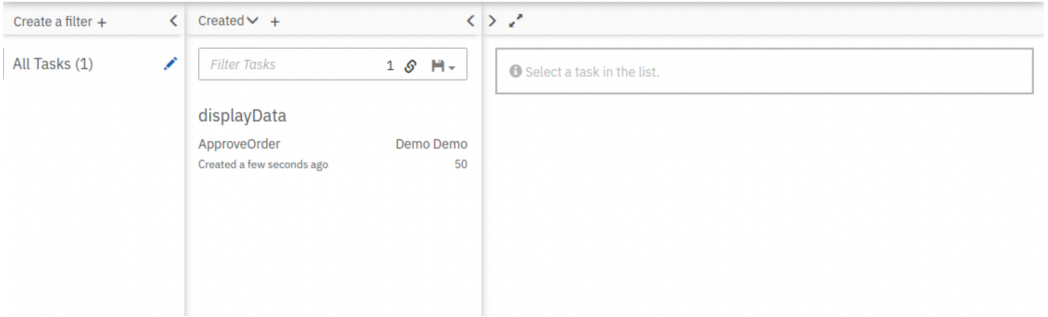


47. Click on the task, and paste in the orderid:

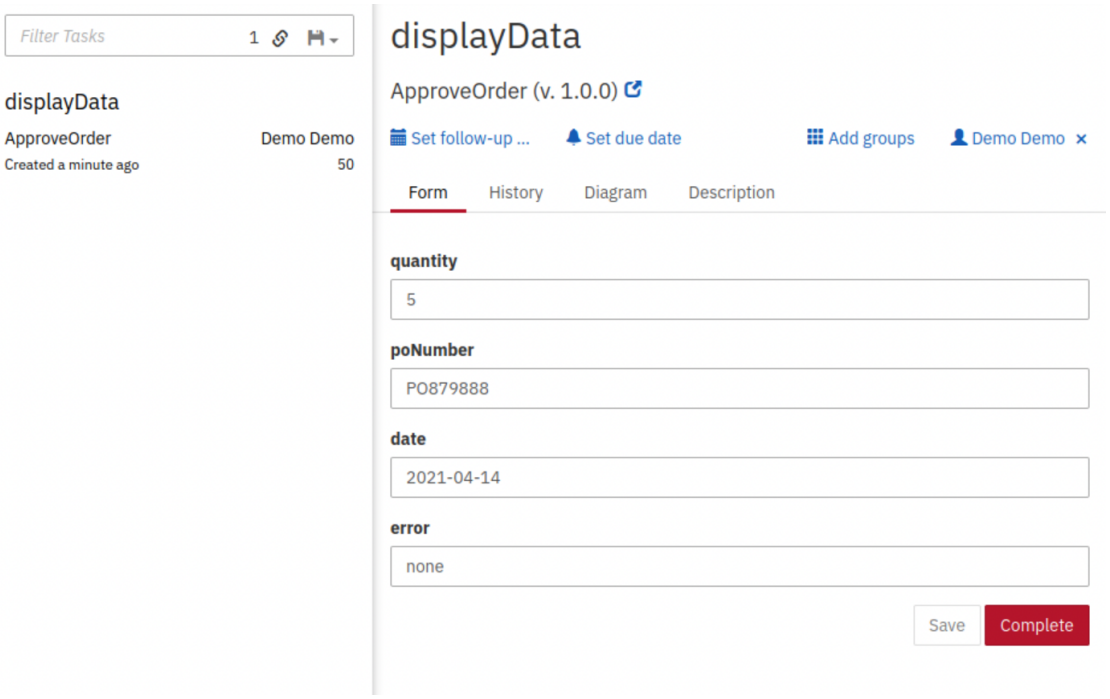


48. Click **Complete**

49. Refresh the Tasks



50. Click on displayData



51. Click Complete to approve this purchase.

52. We've seen how the process can get user data, call out to external services, and review. There are many more features of BPMN - this is really just a tiny preview.

53. Check out the Cockpit and the Admin windows (from the little House icon). There are a lot of features in the package that you would need for a real process management scenario (e.g. adding users, creating tenants, adding new approvers to existing processes, checking on the state of processes, etc).

54. That's all.