

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:

```
oxfordosx:~/camunda$ ./start.sh
Setting JAVA property to /usr/lib/jvm/java-11-openjdk-amd64/bin/java
REST API enabled
WebApps enabled
Swagger UI enabled
classpath: ./internal/webapps/../internal/rest/./internal/swaggerui./configuration/userlib/./configuration/keystore/
[camunda@oxfordosx ~]$
```

Spring-Boot: (V2.4.3)
Camunda Platform: (v7.15.0)

```
2021-04-14 20:37:26.902 INFO 7116 --- [           main] org.camunda.bpm.run.CamundaBpmRun : Starting CamundaBpmRun V7.15.0 using Java 11.0.10 on oxsoa with PID 7116 (/home/oxsoa/camunda/internal/camunda-bpm-run-core.jar started by oxsoa in /home/oxsoa/camunda)
```

```
2021-04-14 20:37:29.433 INFO 7116 --- [           main] org.camunda.bpm.engine.cfg.BpmnRun : No active profile set, falling back to default profiles: default
```

```
2021-04-14 20:37:29.433 INFO 7116 --- [           main] org.camunda.bpm.engine.cfg.BpmnRun : Tomcat initialized with port(s): 8080 (http)
```

```
2021-04-14 20:37:29.449 INFO 7116 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
```

```
2021-04-14 20:37:29.549 INFO 7116 --- [           main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.43]
```

```
2021-04-14 20:37:29.549 INFO 7116 --- [           main] w.s.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
```

```
2021-04-14 20:37:29.549 INFO 7116 --- [           main] Root WebApplicationContext : Initialization completed in 2461 ms
```

```
2021-04-14 20:37:29.642 INFO 7116 --- [           main] o.c.b.s.b.s.c.CamundaJerseyResourceConfig : Configuring camunda rest api
```

```
2021-04-14 20:37:29.642 INFO 7116 --- [           main] o.c.b.s.b.s.c.CamundaJerseyResourceConfig : Linking camunda rest api
```

```
2021-04-14 20:37:30.103 INFO 7116 --- [           main] org.camunda.bpm.spring.boot : STARTER-50040 Setting up jobExecutor with corePoolSize=3, maxPoolSize=10
```

```
2021-04-14 20:37:30.103 INFO 7116 --- [           main] org.camunda.bpm.spring.boot : Initializing ExecutorService 'camundaTaskExecutor'
```

```
2021-04-14 20:37:30.199 INFO 7116 --- [           main] org.camunda.bpm.engine.cfg : ENGINE-12003 Plugin 'CompositeProcessEnginePlugin[genericPropertiesConfiguration, camundaDeploymentConfiguration, camundaProcessEngineConfiguration, camundaDataSourceConfiguration, camundaJobConfiguration, camundaHistoryConfiguration, camundaMetricsConfiguration, camundaAuthorizationConfiguration, CreateAdminUserConfiguration[adminUser=adminUserProperty[id=demo, firstName=demo, lastName=demo, email=demo@localhost, password='*****']], failedJobConfiguration, eventPublisherPlugin, SpringBootSpinProcessingEnginePlugin]' activated on process engine 'default'
```

```
2021-04-14 20:37:30.199 INFO 7116 --- [           main] org.camunda.bpm.engine.cfg : STARTER-50021 Auto-Deploying resources: []
```

```
2021-04-14 20:37:30.199 INFO 7116 --- [           main] o.c.b.s.b.s.event.EventPublisherPlugin : EVENTING-001: Initialized Camunda Spring Boot Eventing Engine Plugin.
```

```
2021-04-14 20:37:30.199 INFO 7116 --- [           main] o.c.b.s.b.s.event.EventPublisherPlugin : EVENTING-003: Task events will be published as Spring Events.
```

```
2021-04-14 20:37:30.199 INFO 7116 --- [           main] o.c.b.s.b.s.event.EventPublisherPlugin : EVENTING-005: Execution events will be published as Spring Events.
```

```
2021-04-14 20:37:30.209 INFO 7116 --- [           main] o.c.b.s.b.s.event.EventPublisherPlugin : EVENTING-007: History events will be published as Spring events.
```

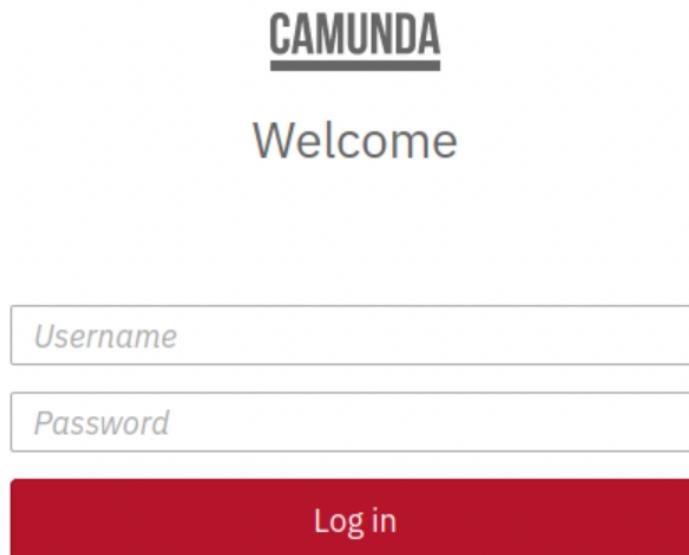
```
2021-04-14 20:37:30.212 INFO 7116 --- [           main] org.camunda.spin : SPIN-01010 Discovered Spin data format provider: org.camunda.spin.impl.json.jackson.ObjectMapper[domainFormatProvider[name = application/json]]
```

```
2021-04-14 20:37:30.631 INFO 7116 --- [           main] org.camunda.spin : SPIN-01010 Discovered Spin data format provider: org.camunda.spin.impl.xml.dom.FormatDomXmlDataFormatProvider[name = application/xml]
```

```
2021-04-14 20:37:30.650 INFO 7116 --- [           main] org.camunda.spin : SPIN-01009 Discovered Spin data format: org.camunda.spin.impl.xml.dom.FormatDomXmlDataFormat[name = application/xml]
```

```
2021-04-14 20:37:30.650 INFO 7116 --- [           main] org.camunda.spin : SPIN-01009 Discovered Spin data format: org.camunda.spin.impl.json.jackson.FormatJsonDataFormat[name = application/json]
```

4. Now open <http://localhost:8080>



5. Login as demo/demo

Telemetry Settings

To enhance user experience, Camunda Platform Runtime Engine can integrate with Camunda Services GmbH, which requires external network requests. Please choose from the setting below.

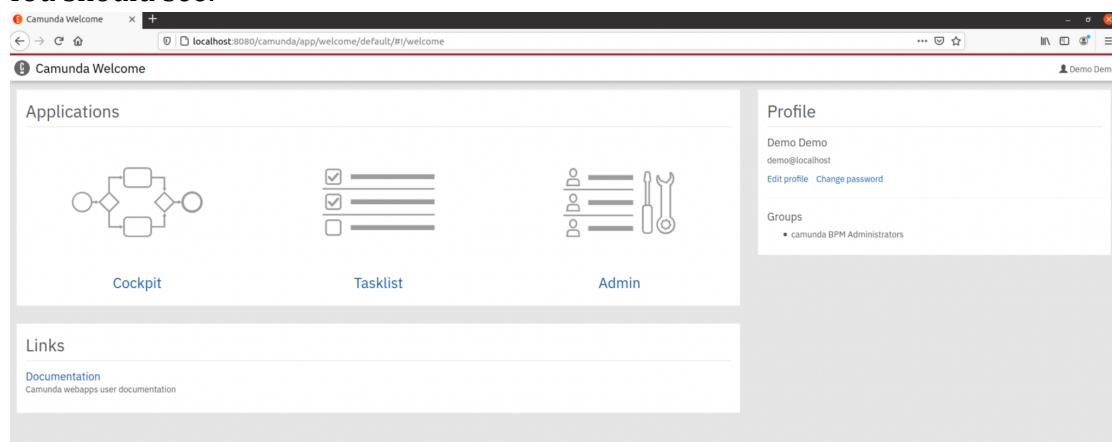
Enable Usage Statistics
Allow Camunda Platform Runtime Engine to send anonymous usage statistics. We use this information to provide you a stable and improved product experience in the environment you are using. This allows Camunda to collect information about the product version and technical environment you are using, and how you are using it.

We respect your privacy. None of your personal information or stored data will be submitted. To learn more, read our [documentation](#) or view our [privacy policy](#).

[Cancel](#) [Save](#)

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
```

Privacy Preferences

To enhance user experience, Camunda Modeler can integrate with 3rd party services, which requires external network requests. Please choose from the settings below.

Enable Error Reports
Allow Camunda Modeler to send error reports containing stack traces and unhandled exceptions.

Enable Usage Statistics
Allow Camunda Modeler to send pseudonymised usage statistics.

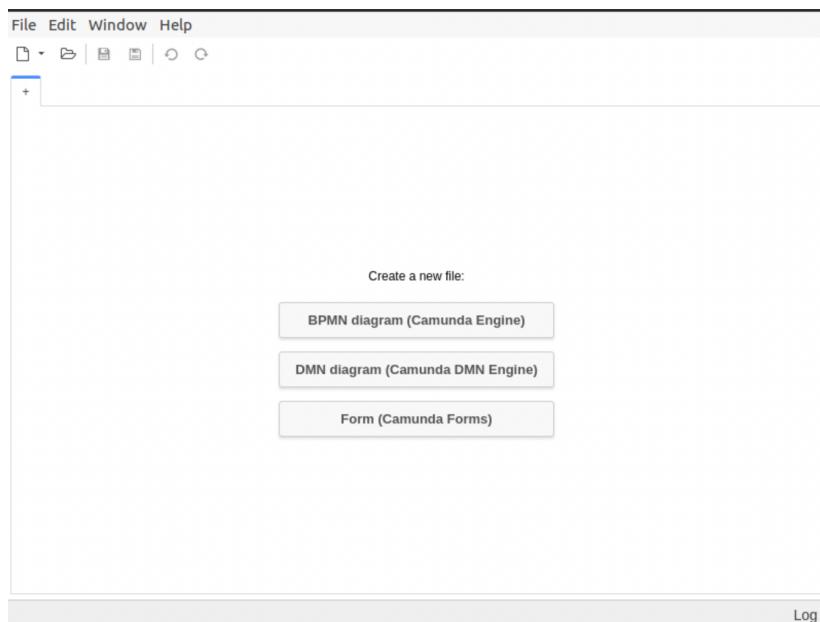
Enable Update Checks
Allow Camunda Modeler to periodically check for new updates.

With any of these options, none of your personal information or stored data will be submitted.
Learn more: [Camunda Privacy Policy](#)

Save

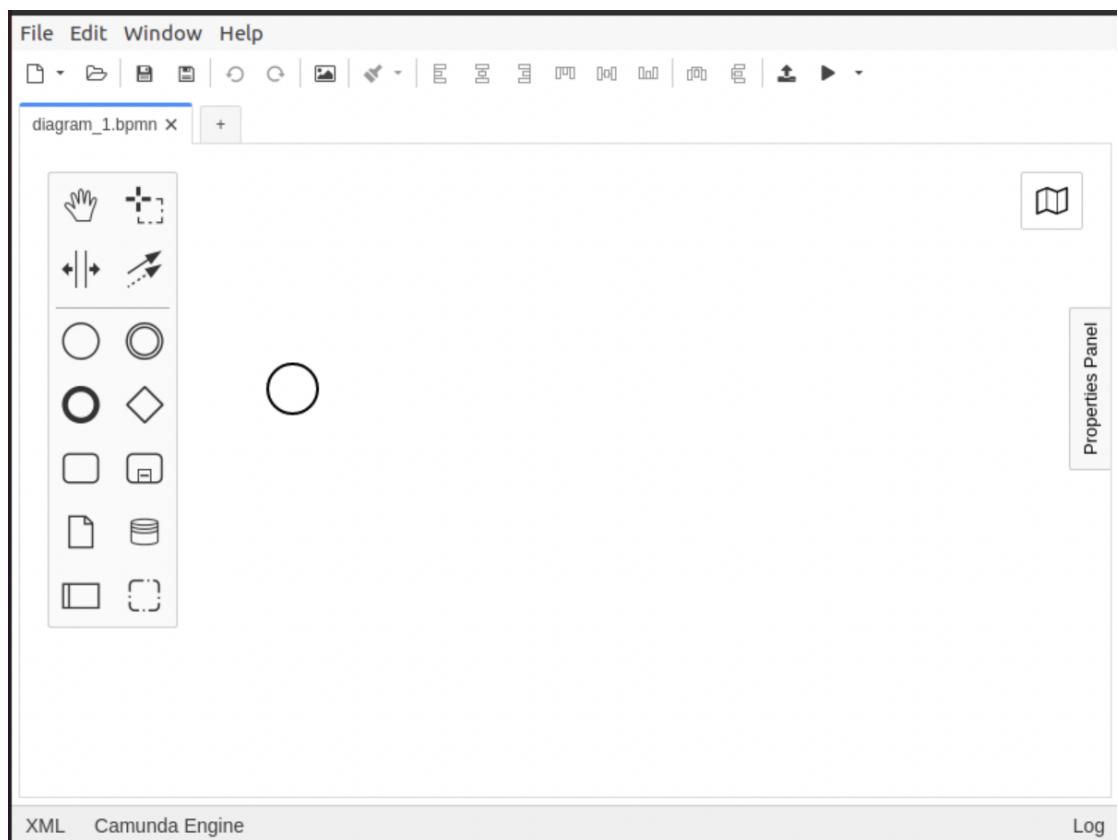
9. Click Save



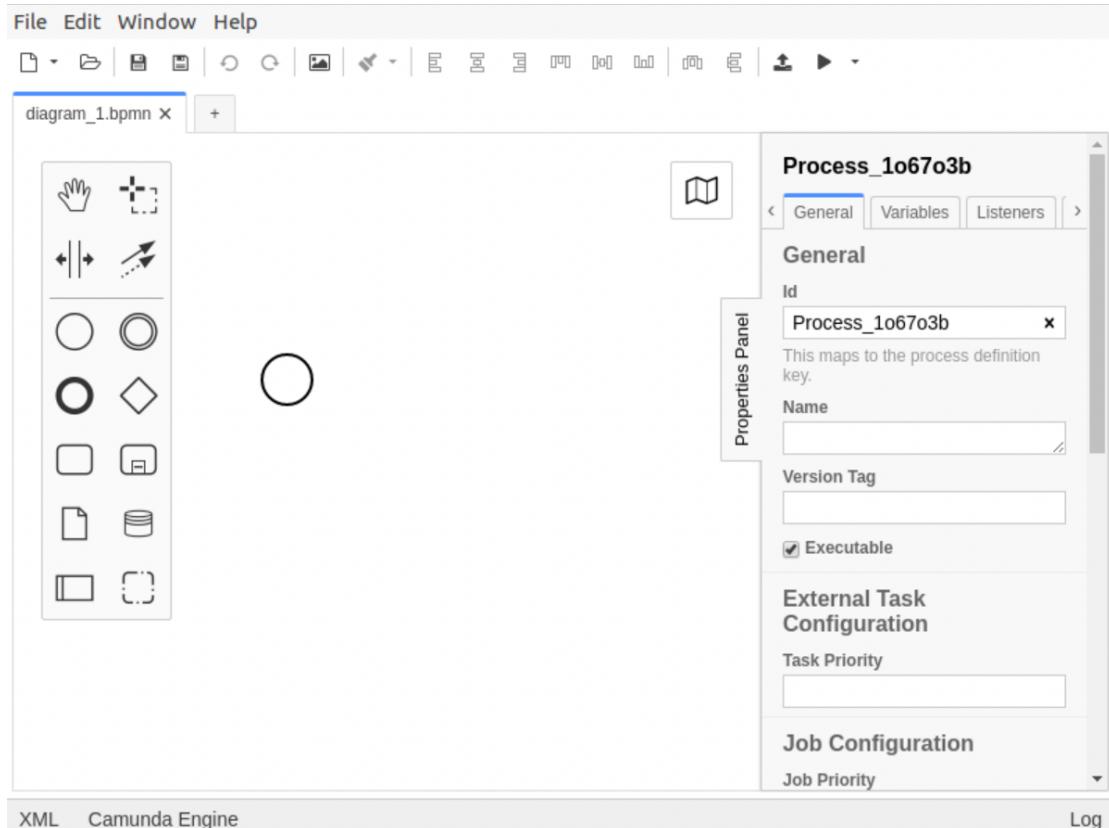


10. Click on Create a BPMN diagram (Camunda Engine).

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:

ApproveOrder

General Listeners Extensions

General

Id: ApproveOrder

This maps to the process definition key.

Name: ApproveOrder

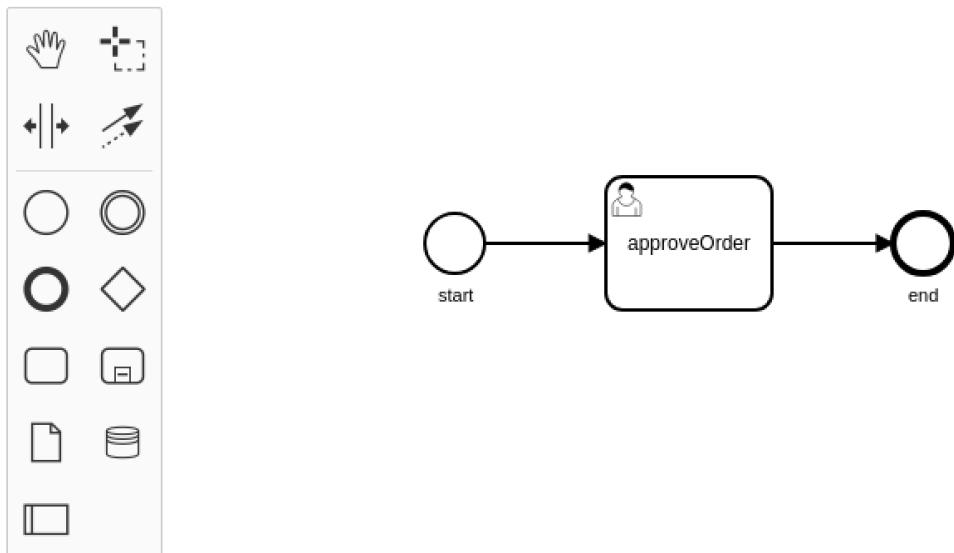
Version Tag: 1.0.0

Executable

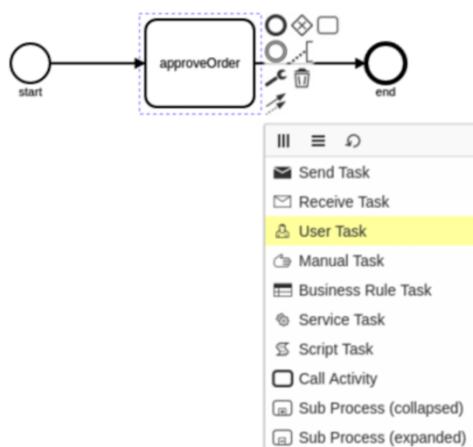
External Task Configuration

Task Priority: (empty field)

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

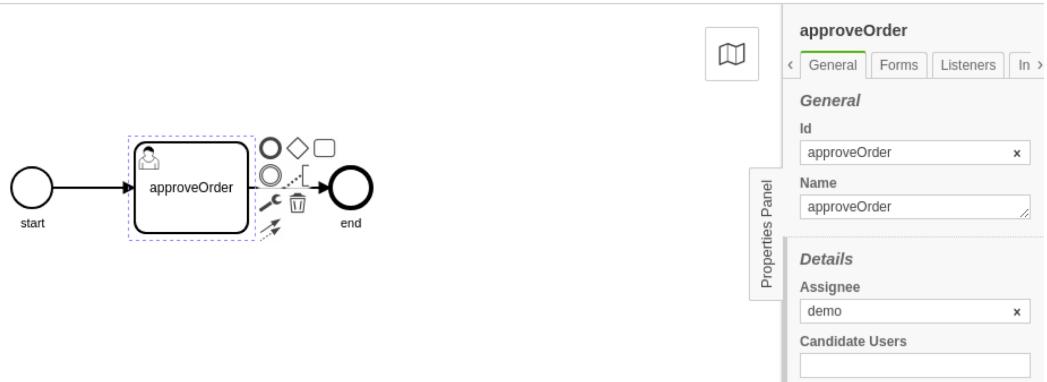


15. In order to make the task into a User Task (with the little “man” icon), click on the spanner/wrench and choose User Task.



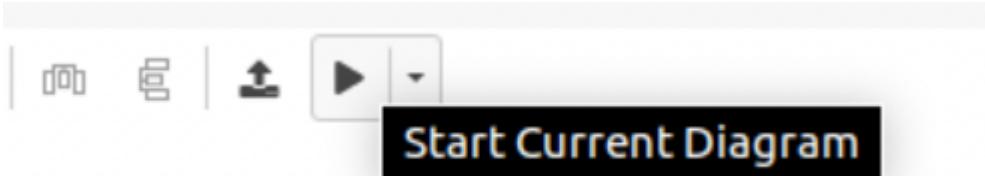
16. Edit the properties for the Approve Order Task:

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:

Start Process Instance - Step 1 of 2

Specify deployment details to deploy this diagram to Camunda.

Deployment Name: **approveorder**

Tenant ID: *Optional*

Endpoint Configuration

REST Endpoint: <http://localhost:8080/engine-rest>

Cancel **Next**

Click **Next**

Enter a test key:

Start Process Instance X

Enter details to start a process instance on the Camunda Engine. Alternatively, you can start a process instance [via a Rest Client](#).

Details

Business Key

Cancel Start

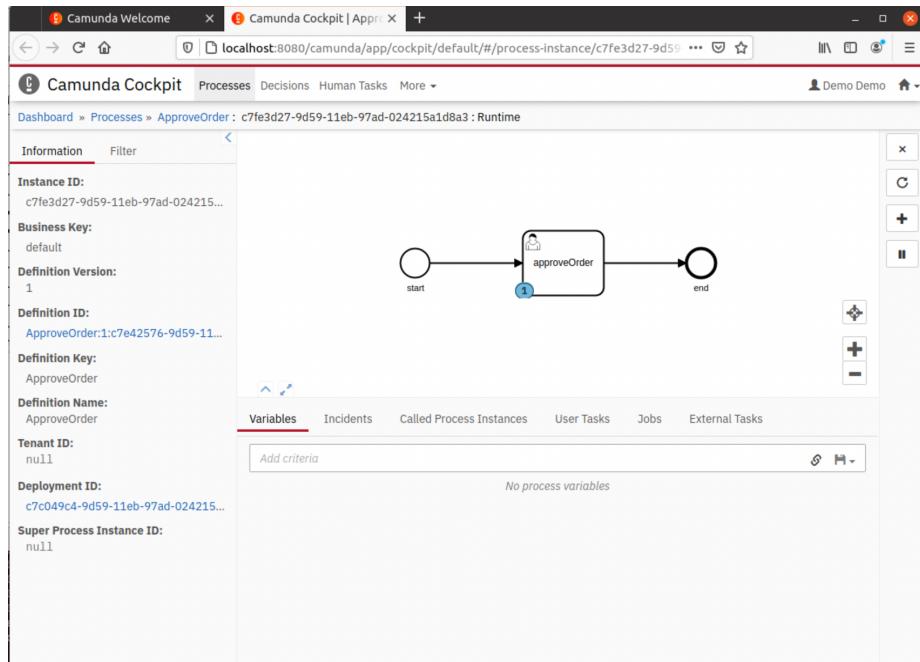
Process instance started X
successfully

[Open in Camunda Cockpit](#) 

Click on the link.

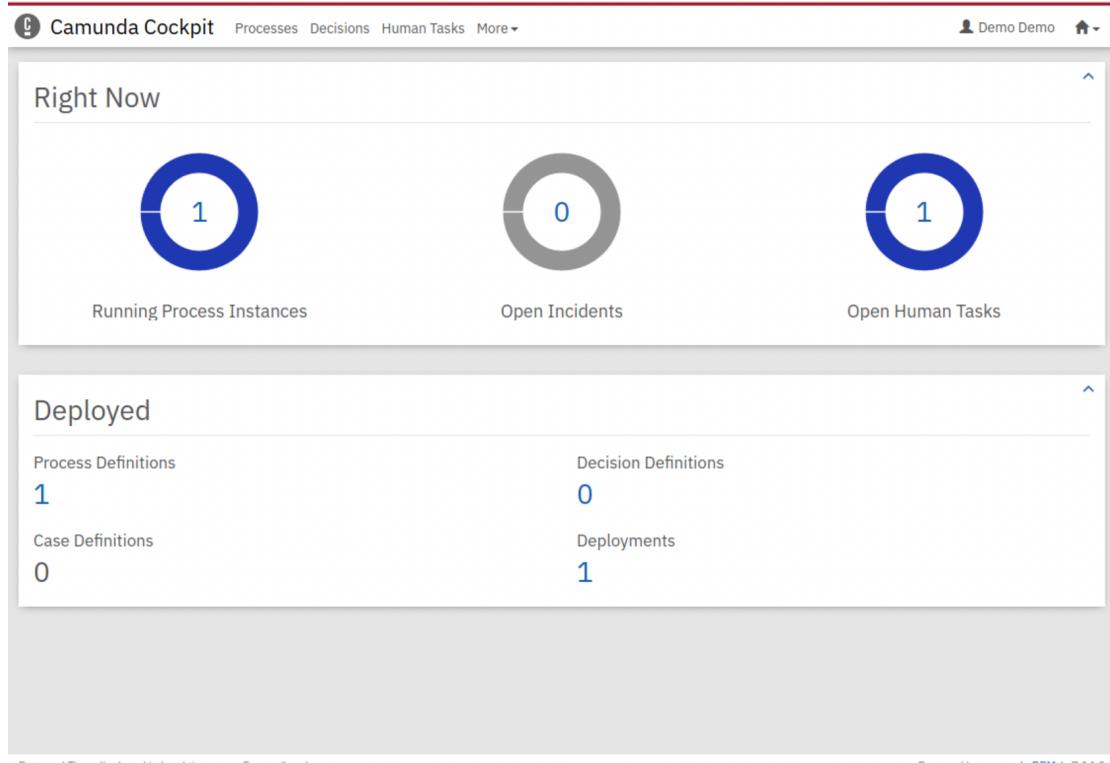


You should see the same process in the runtime now:



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:

The screenshot shows the Camunda Tasklist interface. On the left, there's a sidebar with a 'Create a filter +' button and a '+ Add a simple filter' link. In the center, there's a 'Created' dropdown menu set to 'Created'. Below it is a 'Filter Tasks' input field containing 'Filter Tasks' with a count of '1'. A message says 'Select a task in the list.' There are also 'Keyboard Shortcuts', 'Create task', 'Start process', and user profile links at the top right.

23. Click **Add a simple filter**

The screenshot shows the Camunda Tasklist interface after adding a filter. The sidebar now shows 'All Tasks (1)'. The central area displays a single task named 'approveOrder' with the sub-task 'ApproveOrder'. The task was created '2 minutes ago' by 'Demo Demo' with a priority of '50'. The 'Created' dropdown still shows 'Created'. The 'Filter Tasks' input field now shows '1' and has a 'Delete' icon. The message 'Select a task in the list.' remains.

24. Click on the “approveOrder” task in the Created column

The screenshot shows the Camunda Tasklist interface with the 'approveOrder' task selected. The task details are shown on the right: 'ApproveOrder (v. 1.0.0)', 'Demo Demo' assigned, created '3 minutes ago', priority '50', and a business key 'test-key'. There are tabs for 'Form' (selected), 'History', 'Diagram', and 'Description'. A note says 'You can set variables, using a generic form, by clicking the "Add a variable" link below.' Buttons for 'Set follow-up...', 'Set due date', 'Add groups', and 'Demo Demo' are visible. A 'Complete' button is at the bottom right. The sidebar shows 'All Tasks (1)'.



25. Click on the **Diagram** tab

You will see the process you designed, now with the current step highlighted:

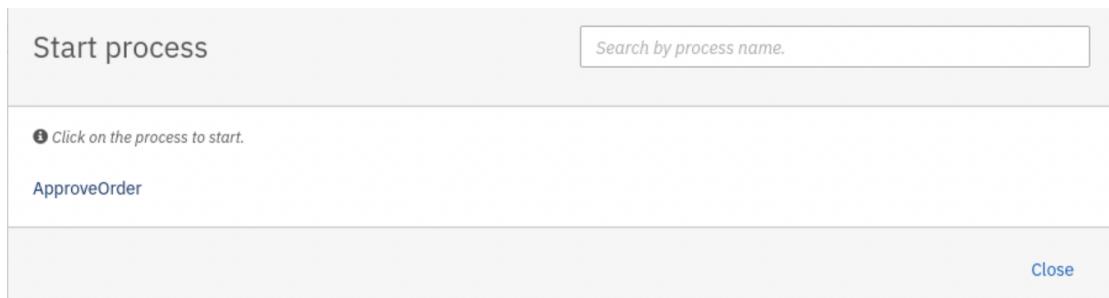


26. Go back to the **Form** tab and click Complete. This will “complete” this instance of the process.

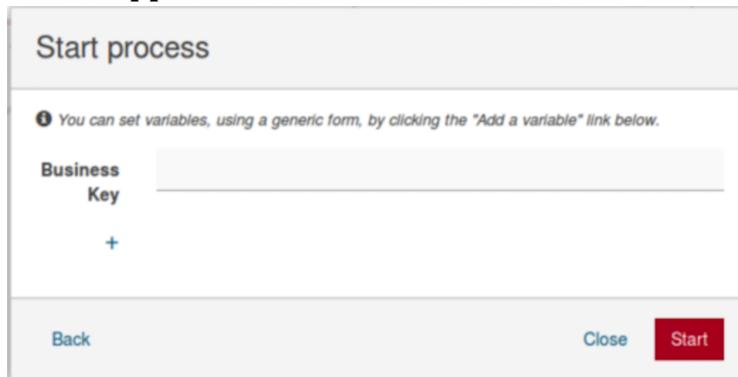
27. This process works, but let's be honest, it is almost pointless.

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**

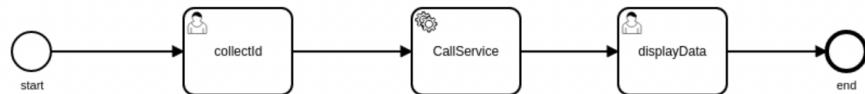
32. Do you remember where you set the assignee for the User Task in the BPMN process to **demo**? Well this has just happened and you are logged in as demo and hence the portal has popped up a message saying you've

been assigned a task to work on:

You are assigned to the following tasks in the same process : X
approveOrder

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:

collectID

General Forms Listeners Input/Output Extentions

General

Id
collectID

This maps to the task definition key.

Name
collectId

Details

Assignee
demo

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

The screenshot shows a configuration interface for a 'collectID' element. At the top, there are tabs: General, Forms (which is selected), Listeners, Input/Output, and Extent. In the 'Forms' section, there is a 'Form Key' input field and a 'Form Fields' list. The 'orderid' field is listed in this list. Below this, there is a detailed 'Form Field' configuration panel with the following fields:

- ID (process variable name): orderid
- Type: string
- Label: orderid
- Default Value: (empty)

38. Configure the CallService task like this:

The screenshot shows a configuration interface for a 'CallService' task. At the top, there are tabs: General (selected), Listeners, Input/Output, and Field Injections. In the 'General' section, there are fields for Id (set to 'CallService') and Name (set to 'CallService'). In the 'Details' section, there are fields for Implementation (set to 'External') and Topic (set to 'getPurchase'). In the 'External Task Configuration' section, there is a Task Priority input field. In the 'Asynchronous Continuations' section, there are two checkboxes: 'Asynchronous Before' and 'Asynchronous After', neither of which is checked. In the 'Documentation' section, there is an Element Documentation input field.



39. Configure the displayData task like this:

displayData

< General Forms Listeners Input/Output Extent >

General

Id
 x

This maps to the task definition key.

Name
 /

Details

Assignee
 x

Candidate Users

Candidate Groups

Due Date

40. And create form fields:

displayData

< General Forms Listeners Input/Output Extent >

Forms

Form Key

Form Fields
 quantity poNumber date error

Form Field

ID (process variable name)
 x

Type

Label
 x

Default Value

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:

The screenshot shows a browser window with the address bar set to 'localhost:8000/purchase'. Below the address bar, there are tabs for 'JSON', 'Raw Data', and 'Headers'. Underneath these tabs are buttons for 'Save', 'Copy', 'Collapse All', 'Expand All', and 'Filter JSON'. A dropdown menu is open, showing '0:' and the value 'href: "0f450c16-31bd-499a-96a2-1e789d0f0aef"'.

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**:

The screenshot shows the Camunda Modeler Tasklist. There is one task listed: 'collectId' assigned to 'ApproveOrder'. The task was created 'a few seconds ago'. The due date is listed as '50'. The task is currently in the 'Demo' state.

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

The screenshot shows the Camunda Modeler Tasklist with the 'collectId' task selected. The 'orderid' field is filled with the value '0f450c16-31bd-499a-96a2-1e789d0f0aef'. The 'Form' tab is active. At the bottom right, there are 'Save' and 'Complete' buttons.

48. Click **Complete**

49. Refresh the Tasks

The screenshot shows the Camunda Modeler Tasklist after completing the 'collectId' task. There is one task listed: 'displayData' assigned to 'ApproveOrder'. The task was created 'a few seconds ago'. The due date is listed as '50'. The task is currently in the 'Demo' state.

50. Click on displayData

The screenshot shows a process instance titled "displayData". The top navigation bar includes "Filter Tasks", a user icon, and a dropdown menu. Below the title, there's a summary row with "ApproveOrder (v. 1.0.0)" and a blue link. To the right are buttons for "Set follow-up ...", "Set due date", "Add groups", and a user profile. The main area contains four form fields: "quantity" (value: 5), "poNumber" (value: PO879888), "date" (value: 2021-04-14), and "error" (value: none). At the bottom right are "Save" and "Complete" buttons.

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.

