

iPaaS Workshop

Lab Instructions



Table of Contents

Introduction	4
Overview	4
1. Validate Your Environment	5
1.1 Sign up for Cloud Trial Environment	5
1.2 Sign in to TIBCO Cloud.....	7
1.2.1 Getting Ready.....	7
1.2.2 How to Do It	7
2. Design an API and Create a Mock Application	10
2.1 Design an API	10
2.1.1 Getting Ready.....	10
2.1.2 How to Do It	11
2.2 Create a Mock App.....	15
2.2.1 Getting Ready.....	15
2.2.2 How to Do It	16
2.3 Import API Specs	17
2.3.1 Getting Ready.....	17
2.3.2 How to Do It	18
2.4 Additional Reading.....	22
3. Build and Deploy Your API	23
3.1 BusinessWorks Apps in TIBCO Cloud Integration	24
3.1.1 How to Do it: Import the Business Works Application via EAR import.....	24
3.2 Flogo Apps in TIBCO Cloud Integration	28
3.2.1 Getting Ready.....	28
3.2.2 How to Do It: Create the Skeleton Flogo App from the API Specification	28
3.2.3 How to Do It: Implement the Flogo App	30
3.2.4 Push the Integration App to TIBCO Cloud and Test It.....	39
3.3 Additional Reading.....	43
Extra LAB: Business Events Simple Decision Service	44
What You Will Learn	44
Getting Ready	44
Creating a Base TCE Project	45
Implementing a Simple decision service	47

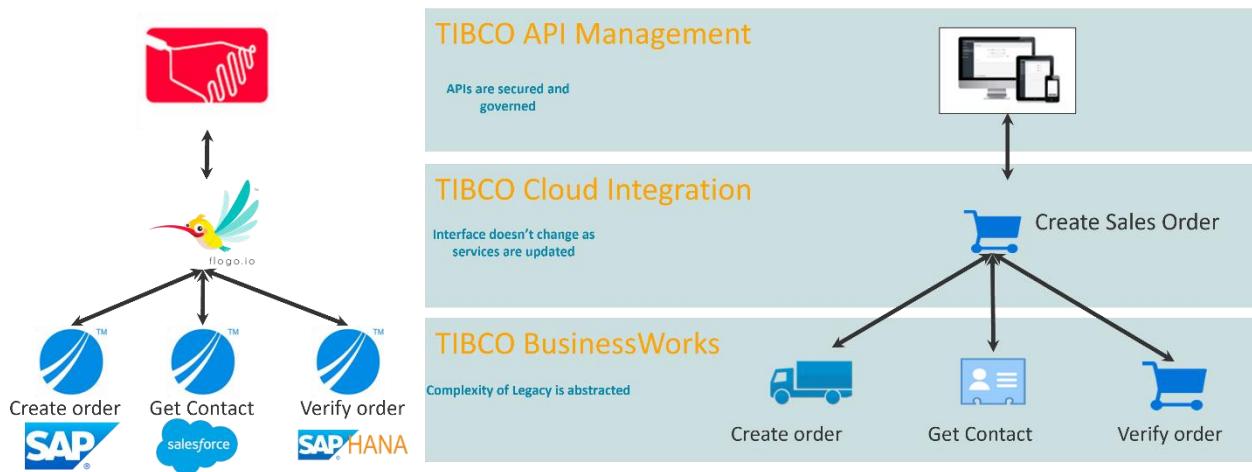
Deploying the Decision Service	50
Testing the Decision Service	52
Summary.....	53
Extra Lab: BusinessWorks Apps in TIBCO Cloud Integration	54
1 Getting Ready	54
2 How to Do It: Connect Business Studio to TIBCO Cloud	54
3 How to Do It: Import a Project in to Business Studio	56
4 How to Do It: Push the Project to TIBCO Cloud and Test	57

Introduction

In this lab exercise you will learn the Tibco Cloud Integration capabilities. You will define new API's in the API management by an API-Led approach. Then you will implement this API using the Integration capabilities. For the connection to the Business Applications, you will deploy existing on-premise Business Works integrations into the Tibco Cloud.

Overview

The use case that you will build is a sales order processing scenario.



Via this step-by-step lab instruction, you will learn the Tibco Cloud Integration service and experience the low code development capabilities of our platform.

1. Validate Your Environment

In this section, you'll validate your lab environment, which consists of a TIBCO Cloud Integration trial environment.

1.1 Sign up for Cloud Trial Environment

You need access to TIBCO Cloud Events subscription. If you do not have one, you can request for free trial via the below link.

TIBCO Cloud™: <https://account.cloud.tibco.com/signup/tci>

Experience the full power of
TIBCO CLOUD™ Integration

iPaaS Trial to Accelerate Your Integration Process

TIBCO Cloud™ Integration accelerates the integration process by empowering more people in your business to connect your information assets together no matter where they are hosted. Offering a high degree of flexibility and choice, our enterprise iPaaS makes connecting everything in your business faster and easier. Take advantage of this FREE iPaaS trial offer today!

Empower integration specialists, developers, and non-technical integrators with user experiences tailored to their skill levels.

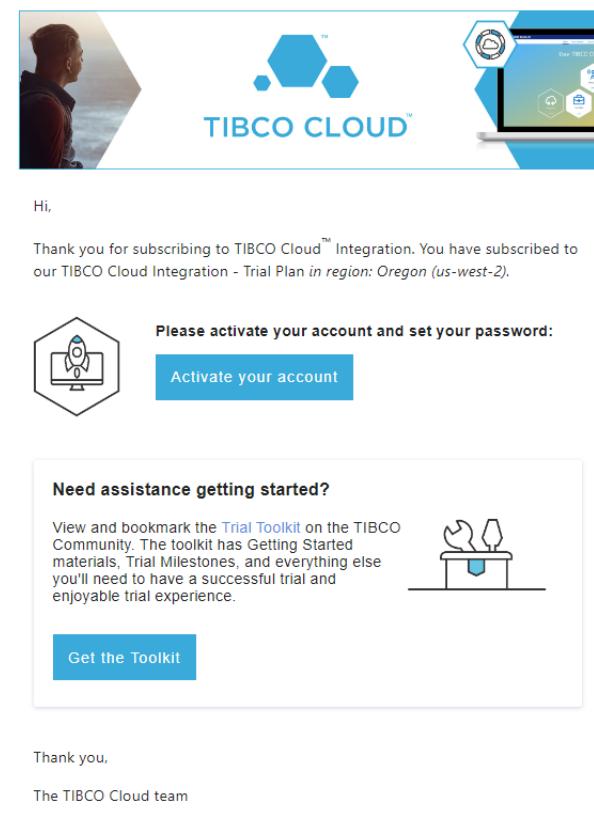
Choose where to host your integration environment - either on AWS or Microsoft Azure.

Address any integration use case using a full-featured API-led and Event-driven integration solution.

Connect quickly and confidently to any endpoint or technology using hundreds of pre-built connectors. Find all available connectors here.

- ❖ choose the “**AWS United States**” as a region and click “**Start Free Trial**”

- ❖ Wait for the Welcome email to activate your trial account.



- ❖ Click the “Activate your Account” button and set your password in the form.

Create your password

.....

Password policy ▾

- ✓ Must have minimum 8 characters.
- ✓ Include atleast three of following
 - English uppercase characters (A through Z)
 - English lowercase characters (a through z)
 - Non-alphanumeric characters (such as !, @, #, \$, %, ^, &, * etc)
 - Numbers (0 through 9)
- ✓ Password mismatch

Confirm password

.....

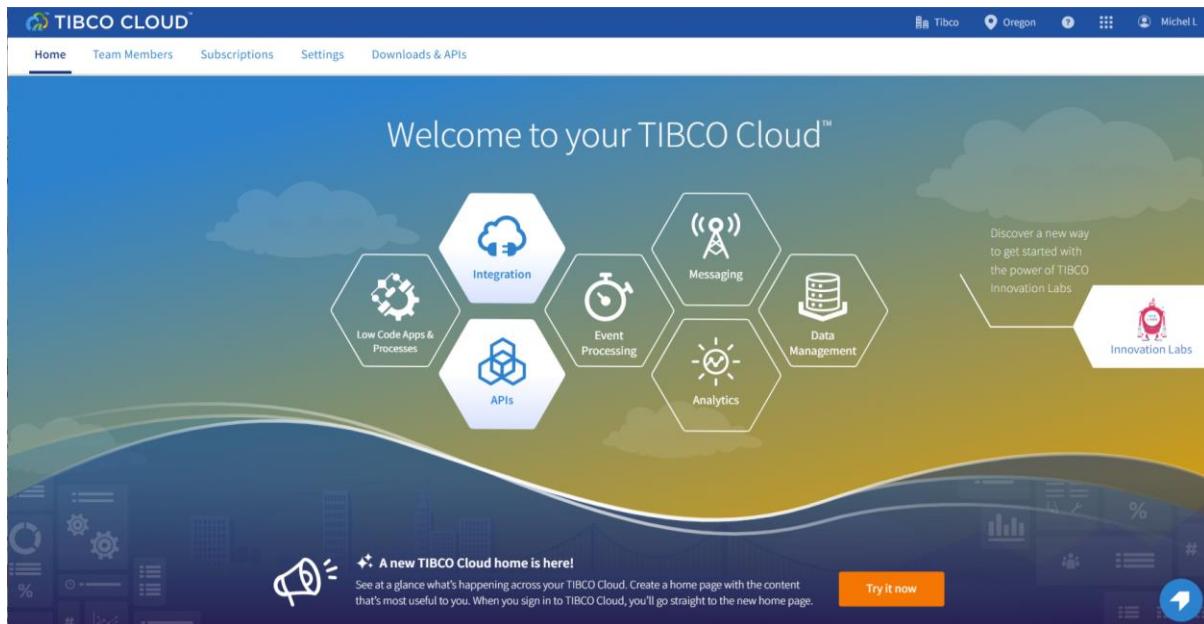
I have read and accept the [End User License Agreement](#), [Privacy Policy](#) and [Terms of Services](#).

ACTIVATE

1.2 Sign in to TIBCO Cloud

1.2.1 Getting Ready

This lab assumes you have started a trial of TIBCO Cloud™ Integration (If you haven't, [click here](#) and follow the instructions below)



1.2.2 How to Do It

1. Browse to <https://cloud.tibco.com/>.

A screenshot of the TIBCO Cloud landing page. The top half features a dark background with a network of glowing nodes and connections, overlaid with text: "The power of TIBCO, in the cloud" and a bulleted list: "• Unified Experience", "• Security by Design", and "• Enterprise Grade". Below this are "SIGN IN" and "LEARN MORE" buttons. The bottom half has a white background with the heading "TIBCO Cloud: Connected Intelligence in Action" and a subtext: "TIBCO Cloud™ is the digital platform that runs and adapts your connected business". It features three large overlapping circles labeled "CONNECT" (blue), "UNIFY" (green), and "PREDICT" (orange) with dashed arrows between them.

2. Click **SIGN IN** (at the right top) and select the **United States – West** Region

Select the region for your TIBCO Cloud™ Account subscription

United States - West (Oregon)	powered by 	Continue
United States - East (North Virginia)	powered by 	Continue
Europe (Ireland)	powered by 	Continue
Australia (Sydney)	powered by 	Continue
United States (Washington)	 Microsoft Azure	Continue

3. Fill out the relevant account details (**Email Address** and **Password**) in the following screen.



Sign into TIBCO® Account
To continue to [TIBCO Cloud](#)

Email Address

Password

[TIBCO LOGIN](#)

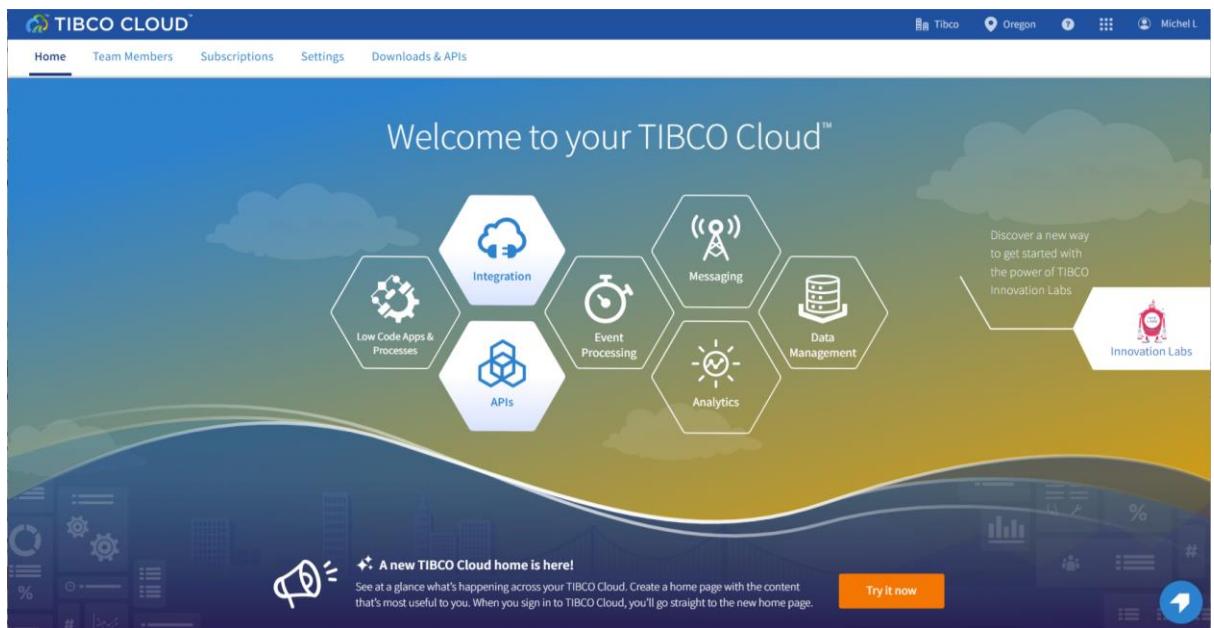
Need help with Login?

OR

 [Use your corporate account](#)

 [Sign in with Google](#)

- After this, you should see a landing page which looks similar to this:



2. Design an API and Create a Mock Application

API Modeler is a web-based tool that provides you with the capability to graphically create and model a REST API.

With API Modeler, you can import REST API specifications either in a YAML or JSON file for further editing, or model your own REST API step-by-step in a visual interface. After modelling an API, you can choose to mock it and see the API in action, or directly implement it.

In this section, you'll design an API using API Modeler and generate a mock application based on it. Furthermore, you'll import several API specs.

- Design an API
- Create a Mock App
- Import API Specs

2.1 Design an API

2.1.1 Getting Ready

In this lab, you'll create an API using the API Modeler. To navigate from the landing page to the API Modeler, do the following:

1. Navigate from the TIBCO Cloud landing page to Cloud Integration by clicking the **APIs** hexagon.
2. Click on the **API Model and Mock** link.
3. After this, your screen should look similar to this:

The screenshot shows the TIBCO CLOUD Integration dashboard. On the left, there's a sidebar titled 'All API specs (1)' under 'GROUPS (1)'. A single entry named 'Default (1)' is listed. The main content area is titled 'All API specs' and contains a table with one row. The table columns are 'API name', 'Version', 'Description', 'Group', and 'Last modified'. The row shows 'TIBCO Cloud Integration PetStor...', '1.1', 'To get you started with the API Modeling and Mock capabilities of TIB...', 'Default', and '21 Sep 2022 1:57 pm'. At the top right of the main area are buttons for 'Search...', 'Create', and other actions. A blue circular arrow icon is in the bottom right corner.

2.1.2 How to Do It

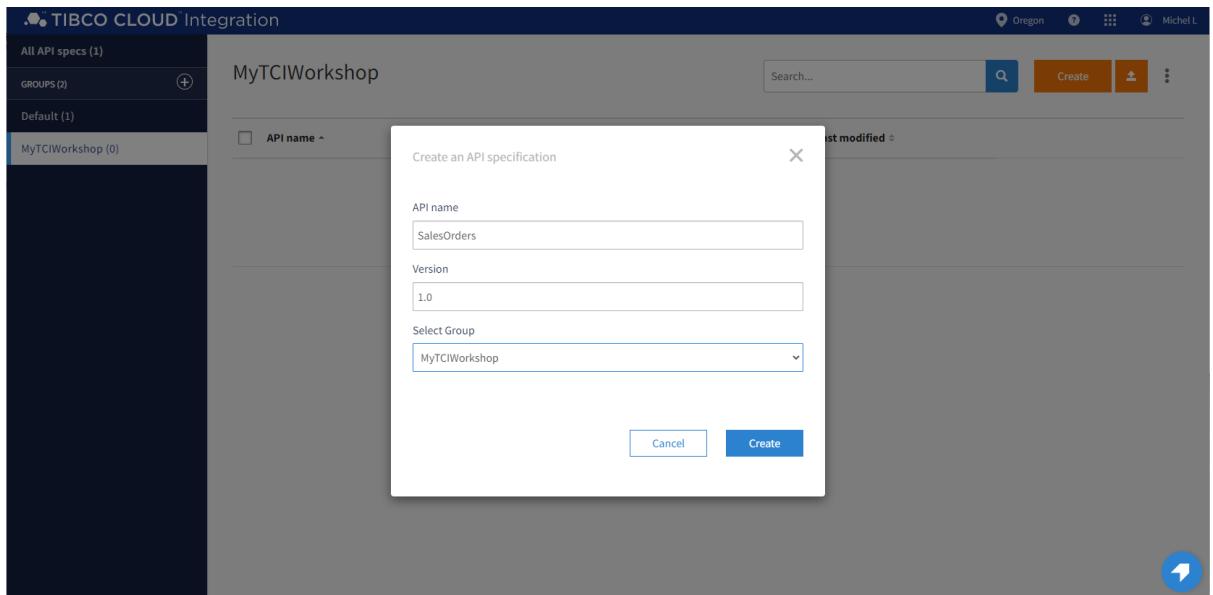
1. Create a group (this is optional) by clicking the + sign to the right from **GROUP** and give it a name e.g. **MyTCIWorkshop**.

A modal dialog box titled 'Create group' is displayed. It has a 'New group name' input field containing 'MyTCIWorkshop'. Below the input field is a 'Create group' button. The background of the dialog is semi-transparent, showing the 'GROUPS (1)' section of the main interface.

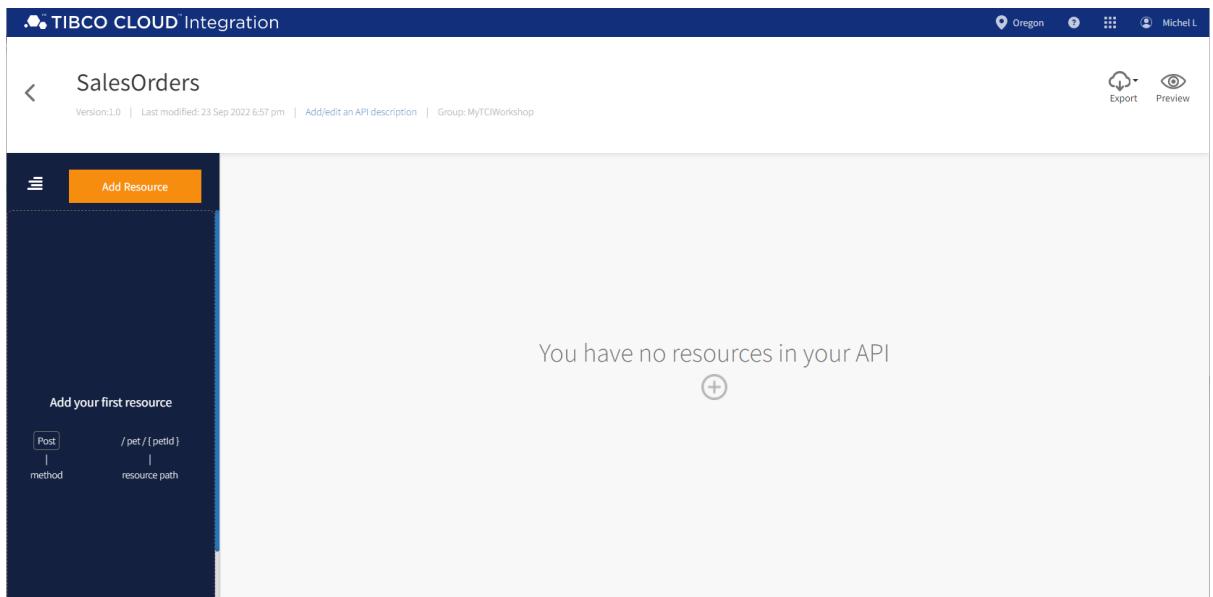
2. Create an API spec by clicking on **MyTCIWorkshop**, and click on the orange **Create** button.
3. In the form, use the following values:

Field	Value
API name	SalesOrders
Version	1.0
Select Group	MyTCIWorkshop

4. It should look something like this:



5. Click the blue **Create** button
6. Add a resource to the API spec by clicking on the orange **Add Resource** button in the next **SalesOrders** screen.



7. In the form, use the following values:

Field	Value
Resource path	/sync/{orderId}
Method	POST

8. The form should look something like this:

The dialog box has a title 'Add a resource and methods' and a close button 'X'. It contains a text input field 'Enter a new resource path (begin with /)' with the value '/sync/{orderId}'. Below it is a section for adding methods with buttons for 'GET', 'POST', 'PUT', 'PATCH', and 'DELETE'. At the bottom are 'Cancel' and 'Save' buttons.

9. Click on the blue **Save** button.

10. Edit the success response of the **POST** method by first clicking on the white **Response** button in the next **POST /sync/{orderId}** screen, and then clicking on the **Success response (200/OK)** card.

The screenshot shows the 'SalesOrders' API endpoint. On the left, there's a sidebar with 'Add Resource' and a list item for '/sync/{orderId}' with a 'POST' method selected. The main panel shows the 'POST /sync/{orderId}' operation. It includes fields for 'Summary: +ADD', 'Description: +ADD', and 'ID: postSync_orderId'. Under 'Response Content Type', 'application/json' is selected. In the 'Responses' section, a '200/OK' card is labeled 'Success response', and an empty dashed box is labeled 'Add'. Navigation tabs at the top include 'API Specs' (which is active), 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. Top right icons include 'Oregon', 'Boaz G', 'Export', and 'Preview'.

11. In the form, click on the **Generate Schema from Sample Data** link.

The dialog box for 'Edit Response' has fields for 'http status code' (set to 200) and 'Description' (set to 'Success response'). Below these are sections for 'Headers' (empty) and 'Schema' (radio buttons for 'String', 'Reference', and 'None', with 'None' selected). A red box highlights the 'Generate Schema from Sample Data' button at the bottom right.

Then on the orange **Continue** button.

Copy the below sample json in the **Provide Sample Data** field:

```
{  
  "_response_string": "Order has been created with SalesDocumentID 00000013572"  
}
```

The form should look something like this:

The screenshot shows a configuration interface for a service operation. At the top, there's an 'Edit Response' section with fields for 'http status code' (set to 200) and 'Description' (set to 'Success response'). Below this is a 'Headers' section with a note: 'Your operation has no header objects'. The main area is titled 'Provide Sample Data' and contains a code editor with the following JSON:

```
1 {  
2   "_response_string": "Order has been created with SalesDocumentID 00000013572"  
3 }
```

To the right of the code editor are two buttons: 'Generate Schema' (orange) and 'Input Schema' (green). At the bottom right of the interface are 'Cancel' and 'Save' buttons.

Click on the orange **Generate Schema** button, and replace **GiveNewSchemaNameHere** by **salesOrderSyncResponse** in the next form.
The form should look something like this:

http status code

200

Description

Success response

Headers +

Your operation has no header objects

Schema String Reference None

Generate Schema from Sample Data

`salesOrderSyncResponse`

```

1: {
2:   "type": "object",
3:   "properties": {
4:     "_response_string": {
5:       "type": "string",
6:       "default": "Order has been created with SalesDocumentID 00000013572"
7:     }
8:   }
9: }

```

Cancel Save

Click on the blue **Save** button.

12. Click on the **Preview** icon, and your screen should look similar to this:

. TIBCO CLOUD Integration

Apps API Specs Connections Extensions VPN Connections Downloads

Oregon

SalesOrders Preview

See more

Search resources

default

POST /sync/{orderId}

Response Class (Status 200)
Success response

Model Example Value

```

{
  "application/json": {
    "_response_string": "Order has been created with SalesDocumentID 00000013572"
  }
}

```

Response Content Type application/json

Parameter	Value	Description	Parameter Type	Data Type
orderId			path	string

[BASE URL: , API VERSION: 1.0]

2.2 Create a Mock App

2.2.1 Getting Ready

In this lab you'll create a mock application based on the API specification created in the previous lab. In order to do this:

1. Navigate to the API specifications by clicking on the **API Specs** menu item.

- Select the group you've created the **SalesOrders** API specification in, e.g. **MyTCIWorkshop**.
- Your screen should look similar to:

The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'TIBCO CLOUD' logo, location 'Oregon', and various icons. The main menu has tabs for 'Apps', 'API Specs' (which is selected), 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. On the left, a sidebar titled 'All API specs' shows a 'GROUP' section with 'Default' and 'MyTCIWorkshop' selected. The main content area is titled 'MyTCIWorkshop' and lists the 'SalesOrders' API specification. The table columns are 'API name', 'Version', 'Description', and 'Last modified'. A single row for 'SalesOrders' is shown with version 1.0 and last modified on 08 May 2019 at 07:41 pm Central European Summer Time. A search bar and a 'Create' button are at the top right of the content area.

2.2.2 How to Do It

- Hover over the **SalesOrders** record to the right of the preview icon, and select **Create Mock app** from the menu:

This screenshot is similar to the previous one, showing the 'MyTCIWorkshop' group in the API Specs section. The 'SalesOrders' entry is highlighted. A context menu is open to the right of the entry, listing options: 'Create Mock app' (with a circled 'i' icon), 'Generate Node.js code', 'Create Flugo app', 'Clone', 'Edit', and 'Remove'. The 'Create Mock app' option is likely the one being referred to in the instructions.

- Create a mock app by clicking on the **Create** button in next form:

The dialog box is titled 'Create Mock app'. It contains a text input field labeled 'Give your new Mock app a name' with the value 'salesorders_1_0_mock_app'. At the bottom are two buttons: 'Cancel' and 'Create'.

- Once the mock app is running, hover over the **Endpoint** link, and select **View and Test** from the menu:

The screenshot shows the TIBCO CLOUD Integration interface. At the top, there's a navigation bar with the TIBCO logo, location (Oregon), user (Michel L.), and a three-dot menu. Below the bar, the application title is 'Salesorders_1_0_mock_app', with status 'Running' and a 'Stop' button. A 'Logs' tab is selected. In the center, there's a section for 'Simple Mock Responses' with a 'POST /sync/{orderId}' endpoint. A 'Success response' is shown as a JSON object: { "application/json": { "_response_string": "Order has been created with SalesDocumentID 00000013572" } }. To the right, there are buttons for 'View and Test' (which is highlighted in blue), 'Copy URL', and 'Publish to API Management'. A message at the bottom right says 'No updates to push'.

- Test the mock app by filling out a value in the **orderId** field, and clicking on the **Try it out!** button:

The screenshot shows the 'SalesOrders' API tester interface. On the left, there's a sidebar with a search bar, 'Collapse all | Expand all', and a 'default' section with a 'POST /sync/{orderId}' button. The main area shows a 'Response Class (Status 200)' section with a 'Success response' JSON object. Below it, there's a 'Response Content Type' dropdown set to 'application/json', a 'Parameters' table with an 'orderId' entry, and a 'Request URL' input field containing 'https://64286c3e561fcb85707011eabb-integration.cloud.tibcoapps.com:443/xrzzwskx3lcyaizkn445'. At the bottom, there's a 'Response Body' section with the same JSON response as the 'Success response'.

2.3 Import API Specs

2.3.1 Getting Ready

In this lab you'll import additional API specifications. In order to do this:

1. Navigate to the API specifications by clicking on the **API Specs** menu item.
2. Select the group you've created the **SalesOrders** API specification in, e.g. **MyTCIWorkshop**.
3. Your screen should look similar to:

The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'Oregon' and other icons. The main menu has tabs for 'Apps', 'API Specs' (which is selected), 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. On the left, a sidebar titled 'All API specs' shows a 'GROUP' section with 'Default' and 'MyTCIWorkshop' selected. The main content area is titled 'MyTCIWorkshop' and lists one API specification: 'SalesOrders' (Version 1.0, Last modified 08 May 2019 07:41 pm Central European Summer Time). A search bar and a 'Create' button are at the top right of the list table.

2.3.2 How to Do It

1. Click on the **Import** button to import the API specifications:

This screenshot is identical to the previous one, but the 'Create' button has been replaced by an 'Import' button, which is highlighted with a red circle. The rest of the interface and data remain the same.

2. Select **Import from filesystem**,

This screenshot shows the 'Import' button with a context menu open. The menu items are 'Import from filesystem' (highlighted in blue), 'Import from URL', and 'Import from Github'. The rest of the interface is consistent with the previous screenshots.

and select the following files from the github [/api_specs/](#) directory:

- **SAPHanaPurchaseOrders.json**
- **SAPOrders.json**
- **sfContacts.json**

3. Once these files have been uploaded, your screen should look similar to this:

The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'TIBCO CLOUD Integration', 'Oregon' location, and various icons. Below the navigation is a sub-menu with 'All API specs', 'GROUP', and 'Default'. A search bar and a 'Create' button are also present. The main content area is titled 'MyTCIWorkshop' and lists five APIs: SalesOrders, SAPHanaPurchaseOrders, SAPOrders, and sfContacts, all version 1.0 and last modified on 08 May 2019 at 08:04 pm Central European Summer Time.

- For the SAPHanaPurchaseOrders we also create a Mock application as we did in 2.2.2.

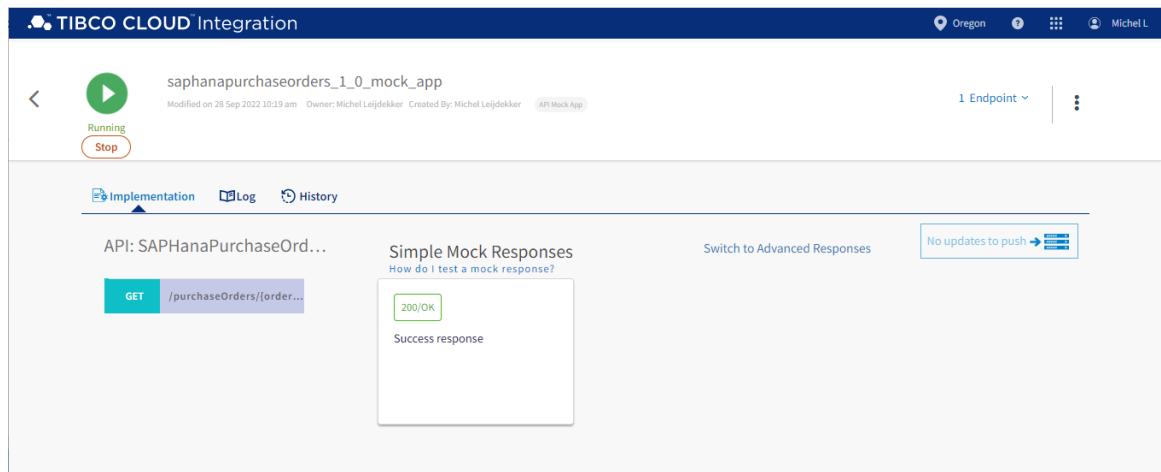
- Hover over the **SAPHanaPurchaseOrders** record to the right of the preview icon, and select **Create Mock app** from the menu:

This screenshot shows the same interface as above, but with a context menu open over the SAPHanaPurchaseOrders row. The menu includes options like 'Create Mock app' (which is highlighted), 'Generate Node.js code', 'Clone', 'Edit', and 'Remove'.

- Create a mock app by clicking on the **Create** button in next form:

The dialog box is titled 'Create Mock app'. It contains a text input field labeled 'Give your new Mock app a name' with the value 'saphanapurchaseorders_1_0_mock_app'. At the bottom are 'Cancel' and 'Create' buttons.

- Click on the **Simple Mock Responses** Tile to specify the response message that the mock service will return.



In the Edit Mock response wizard, copy and paste the following .json snippet

```
{
  "d": {
    "CreationDate": "/Date(1472774400000)/",
    "CreatedByUser": "CB998000027",
    "DistributionChannel": "10",
    "OverallSDProcessStatus": "20",
    "OverallTotalDeliveryStatus": "B",
    "SalesDistrict": " ",
    "SoldToParty": "17100001",
    "TotalNetAmount": "353.50",
    "TransactionCurrency": "USD",
    "SalesOrderType": "OR",
    "CustomerPurchaseOrderDate": "/Date(1472774400000)/",
    "OverallSDDocumentRejectionSts": "A",
    "ShippingCondition": "01",
    "IncotermsTransferLocation": "Palo Alto",
    "IncotermsVersion": " ",
    "SalesOrderDate": "/Date(1472774400000)/",
    "HeaderBillingBlockReason": " ",
    "SalesGroup": " ",
    "SalesOrder": "2",
    "ShippingType": " ",
    "IncotermsLocation2": " ",
    "DeliveryBlockReason": " ",
    "IncotermsLocation1": "Palo Alto",
    "AssignmentReference": " ",
    "OrganizationDivision": "00",
    "PurchaseOrderByCustomer": "John::Fisher::jfisher@acme.com",
    "SalesOrganization": "1710"
  }
}
```

```

        "IncotermsClassification": "123",
        "SalesOffice": " ",
        "TotalCreditCheckStatus": " ",
        "CustomerPurchaseOrderType": "123",
        "PricingDate": "/Date(1472774400000)/",
        "CustomerPaymentTerms": "0004",
        "LastChangeDate": "/Date(1472774400000)/",
        "SDDocumentReason": " ",
        "RequestedDeliveryDate": "/Date(1472774400000)/",
        "PaymentMethod": " ",
        "LastChangeDateTime": "/Date(1472774400000)/"
    }
}

```

- Click “**Save**”

200/OK

```

1 {
2   "d": {
3     "CreationDate": "/Date(1472774400000)/",
4     "CreatedByUser": "CB9980000027",
5     "DistributionChannel": "10",
6     "OverallISDProcessStatus": "20",
7     "OverallTotalDeliveryStatus": "B",
8     "SalesDistrict": " ",
9     "SoldToParty": "17100001",
10    "TotalNetAmount": "353,50".
11

```

Generate response from schema Cancel Save

- After the save action click **Update Mock app** in the top right corner.
- Once the mock app is running again, hover over the **Endpoint** link, and select **View and Test** from the menu:

TIBCO CLOUD® Integration

saphanapurchaseorders_1_0_mock_app

Running

Implementation Log History

Simple Mock Responses

How do I test a mock response?

GET /purchaseOrders/{order...}

200/OK

Success response

Switch to Advanced Responses

No updates to push

- Verify if the mock service response is as expected.

- If everything is fine, we can change the endpoint visibility to refer to the Tibco Cloud Mesh. This will make sure the endpoint is only available as a private endpoint. To do this, click the < icon to navigate back to the API overview screen. From the menu on the top right corner, select the option **“Set endpoints private to my Tibco Cloud”** and click **Update**.

2.4 Additional Reading

- [How to Model Your API with TIBCO Cloud Integration](#)
- [TIBCO Cloud Integration - API Modeler](#)
- [My first API with API Modeler](#)
- [TIBCO Cloud Integration - API Mock App](#)
- [Testing APIs with Mock apps](#)
- [Meter Data Service API Mock for TIBCO Cloud Integration](#)

3. Build and Deploy Your API

In this section you have two options:

1. You will import the exported Business Works applications into the TIBCO Cloud Business Works runtime environment.
2. You'll push an app implemented in BusinessWorks from Business Studio to TIBCO Cloud. The lab instructions for this part has been added as an additional lab, since it will require the install of a local Business Works Studio environment.

Functionally, the app implements a system API that gets purchase orders from SAP Hana.

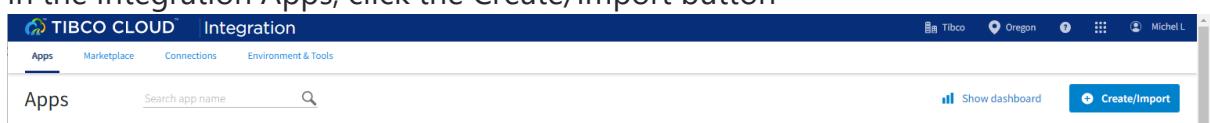
After that you'll create a Flogo app that implements a process API that creates sales orders, by first invoking the system API that gets a purchase order from SAP Hana, then invoking a system API that gets contact details from Salesforce, and finally creates a sales order by invoking a system API for SAP ERP.

- BusinessWorks Apps in TIBCO Cloud Integration
 - Option 1: Import the Business Works Application via EAR import.
 - Option 2: Connect Business Studio to TIBCO Cloud
 - Import a Project in to Business Studio
 - Push the Project to TIBCO Cloud and Test It
- Flogo Apps in TIBCO Cloud Integration
 - Create the Skeleton Flogo App from the API Specification
 - Implement the Flogo App
 - Push the Flogo App to TIBCO Cloud and Test It

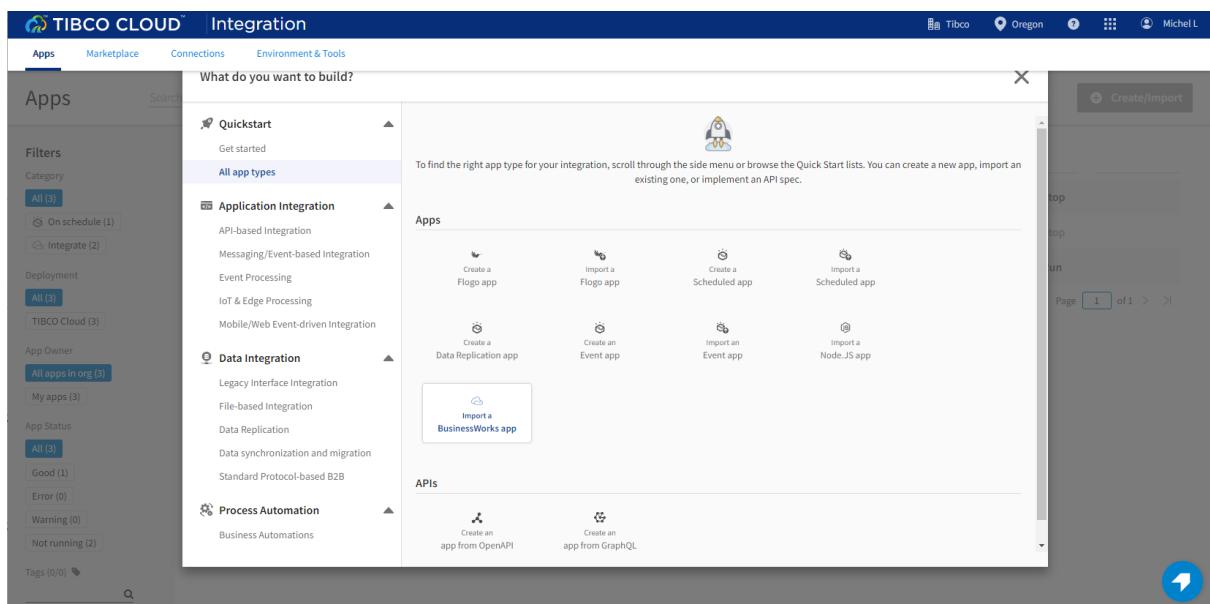
3.1 BusinessWorks Apps in TIBCO Cloud Integration

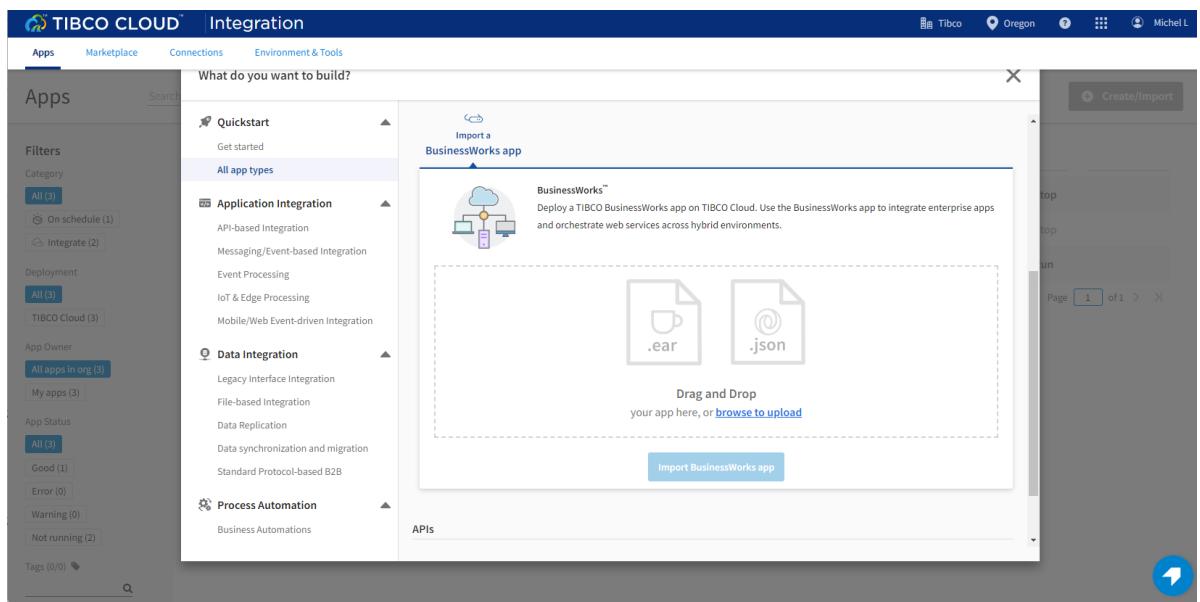
3.1.1 How to Do it: Import the Business Works Application via EAR import

1. We will import the EAR export files and the corresponding manifest files into the Tibco Cloud Integration.
2. In the Integration Apps, click the Create/Import button



3. Then select the “**All app types**” menu option from the wizard and select the “**Import a BusinessWorks app**”

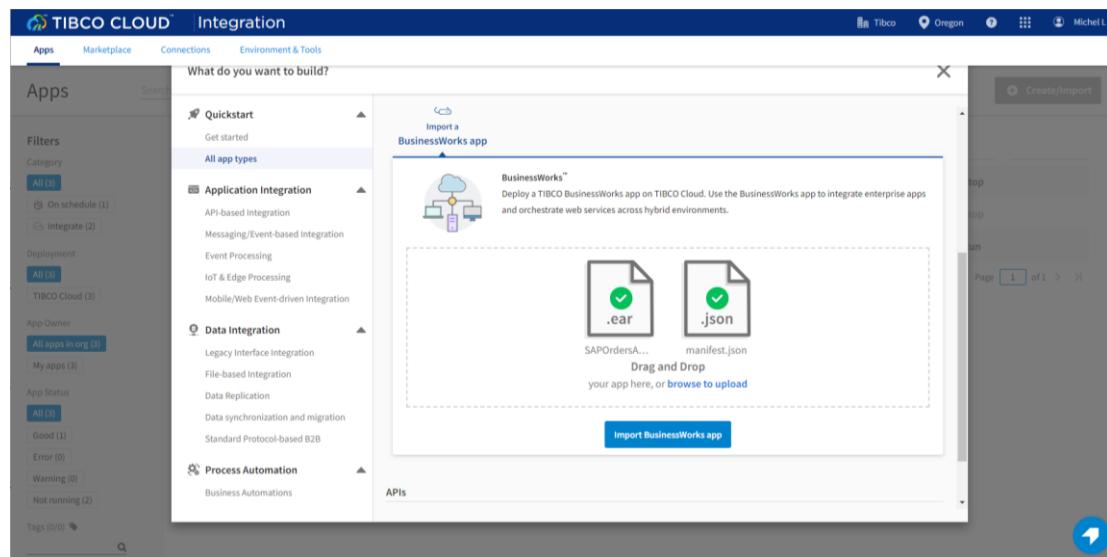




4. Select the following EAR- and manifest files. Files can be found on the GitHub, by clicking in the links below.

Service	Files
Contact Service	SalesForceAPI 1.0.0.ear + manifest.json
Orders Service	SAPOrdersAPI 1.0.0.ear + manifest.json
Sales Order Service (stub via API Mock)	SAPHanaAPI 1.0.0.ear + manifest.json

Important Note: The trial subscription only allows to run two apps at the same time. Therefor we will use an API Mock to simulate 1 of the 3 required Business Works solutions.



- After successful installation of the 3 Business Works applications, your screen should look similar to the following:

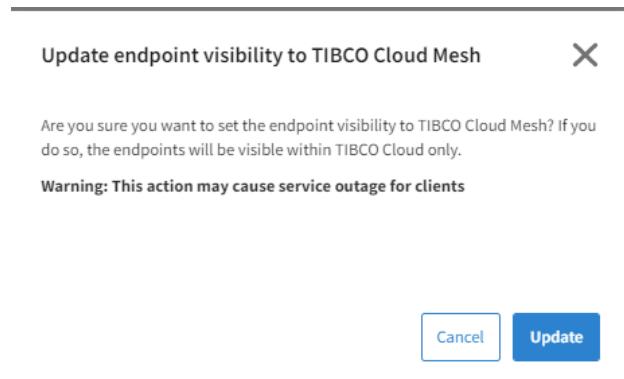
Category	Name	Deployment	Last Modified	Last Started	Instances	Status
Integrate	SAPOrdersAPI	TIBCO Cloud	24 Sep 2022 10:29 am	23 Sep 2022 2:30 pm	1	Running Stop
Integrate	SalesForceStub	TIBCO Cloud	23 Sep 2022 1:43 pm	23 Sep 2022 1:25 pm	1	Running Stop
Integrate	SAPHana_SalesOrder_Stub	TIBCO Cloud	23 Sep 2022 2:00 pm		0	Stopped Run

- When the Business Works integrations are running, we need to set the endpoint visibility to limit it from public to Tibco Cloud Mesh.

The Tibco Cloud mesh is an internal API Library within the Tibco Cloud. This prevents API calls to be called from a public internet. In our integration we only make calls to other Tibco Cloud components, therefor we select the Tibco Cloud Mesh as the API endpoint option.

On each of the Business Works Applications, select the option to update the endpoint visibility. Click on the three dots menu in the right top corner of the application details screen and select the "Set Endpoint Visibility".

Click **Update** to confirm the action.



3.2 Flogo Apps in TIBCO Cloud Integration

In this lab you'll implement a process API that creates sales orders, by first invoking the system API that gets a purchase order from SAP Hana, then invoking a system API that gets contact details from Salesforce, and finally creates a sales order by invoking a system API for SAP ERP.

You will first create a "skeleton" Flogo app from the **SalesOrders** API specification. Then you will implement the flow with the 3 system API calls, to finally push the Flogo app and test it.

3.2.1 Getting Ready

To be able to create the "skeleton" Flogo app from the **SalesOrders** API specification, do the following:

1. Navigate to the Integration Apps specifications by clicking on the **Integration** hexacon in the main menu.
2. Your screen should look similar to:

The screenshot shows the TIBCO CLOUD Integration Apps interface. On the left, there is a sidebar with filters for Category (All 3, Integrate 3), Deployment (All 3, TIBCO Cloud 3), App Owner (All apps in org 3, My apps 3), App Status (All 3, Good 2, Error 0, Warning 0, Not running 1), and Tags (0/0). The main area displays a table of apps with columns: Category, Name, Deployment, Last Modified, Last Started, Instances, and Status. The table contains three rows:

Category	Name	Deployment	Last Modified	Last Started	Instances	Status
Integrate	SAPOrdersAPI	TIBCO Cloud	24 Sep 2022 10:29 am	23 Sep 2022 2:30 pm	1	Running Stop
Integrate	SalesForceStub	TIBCO Cloud	23 Sep 2022 1:43 pm	23 Sep 2022 1:25 pm	1	Running Stop
Integrate	SAPHana_SalesOrder_Stub	TIBCO Cloud	23 Sep 2022 2:00 pm		0	Stopped Run

At the bottom right of the table, there is a pagination indicator showing "Page 1 of 1".

3.2.2 How to Do It: Create the Skeleton Flogo App from the API Specification

To create a "skeleton" Flogo app from on the **SalesOrders** API specification, do the following:

1. Click on the blue Create/Import button, this will open the following wizard

- Select the API based Integration option from the left menu and select your SalesOrders API specification.

The screenshot shows the TIBCO Cloud Integration platform. On the left, there's a sidebar with various filters and categories. The main area is focused on 'Application Integration' under 'API-based Integration'. A modal window titled 'Flogo' is open, asking to 'Create an app from an OpenAPI Specification using an existing file stored in API Modeler or by uploading an Open API Specification file.' It shows a list of APIs, including 'SalesOrders', which is highlighted. At the bottom of the modal is a button labeled 'Import OpenAPI spec'.

- Click on the **Import OpenAPI spec** button, to generate the skeleton. After generation, your screen should look like this.

The screenshot shows the generated integration skeleton for 'New_Flogo_App_0'. The 'Flows' tab is active, showing a single node named 'ReceiveHTTP...' with the action 'postSync_orderid_POST'. The interface includes tabs for 'Endpoints', 'Monitoring', 'Environment controls', 'Logs', 'History', and 'Execution History'. At the top right, there are buttons for 'Properties', 'Schemas', 'Validate', and 'Push'. The overall layout is clean and modern, typical of a cloud-based integration tool.

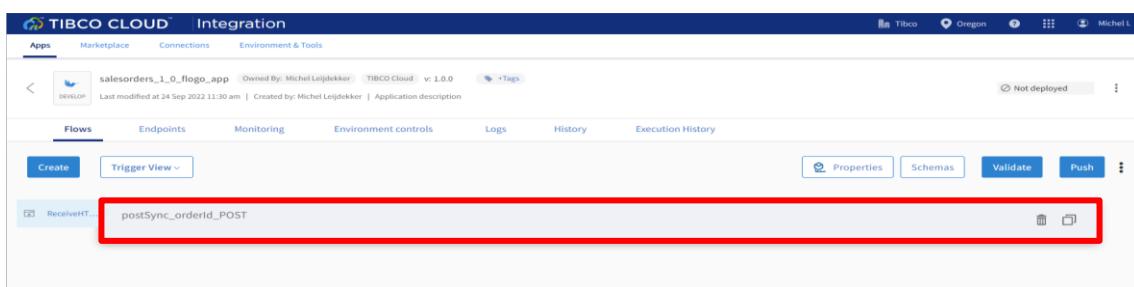
- Click on the "New_Flogo_App_0" name of the integration and rename it to "**salesorders_1_0_flogo_app**".

3.2.3 How to Do It: Implement the Flogo App

In this lab, you'll create a process API that creates sales orders by making 3 system API calls. The address of first system API call is the URL of the BusinessWorks app you have pushed and tested in the previous lab. The addresses of the other API calls are from apps we have already deployed, and their URLs are specified below.

To implement the process API, do the following:

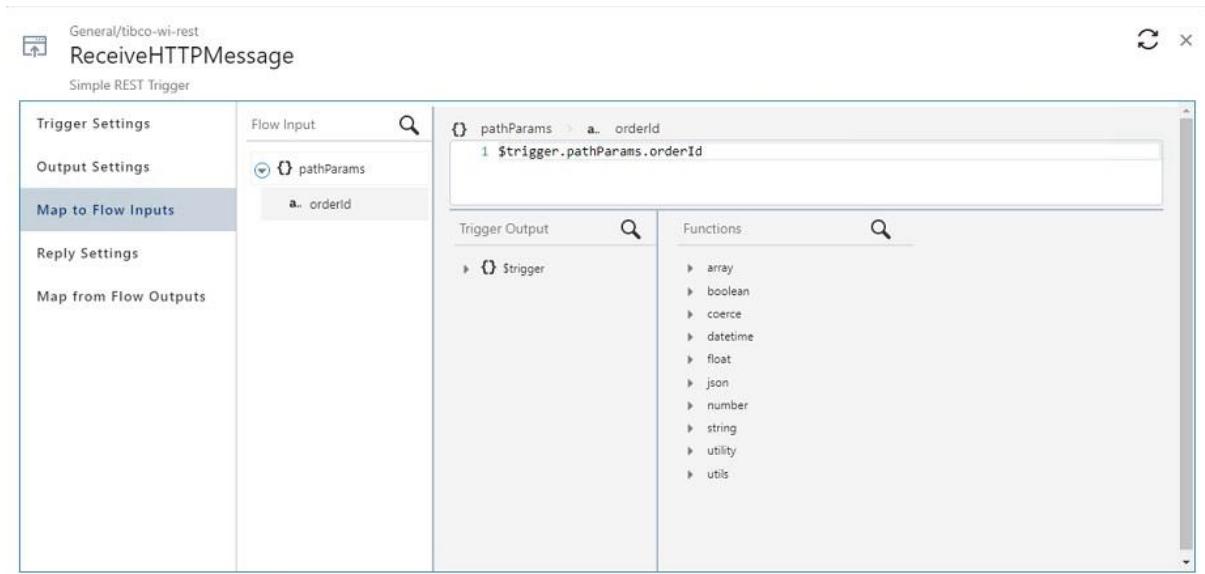
1. Navigate from the app **salesorders_1_0_flogo_app** to the skeleton flow **postSync_orderId_POST**.



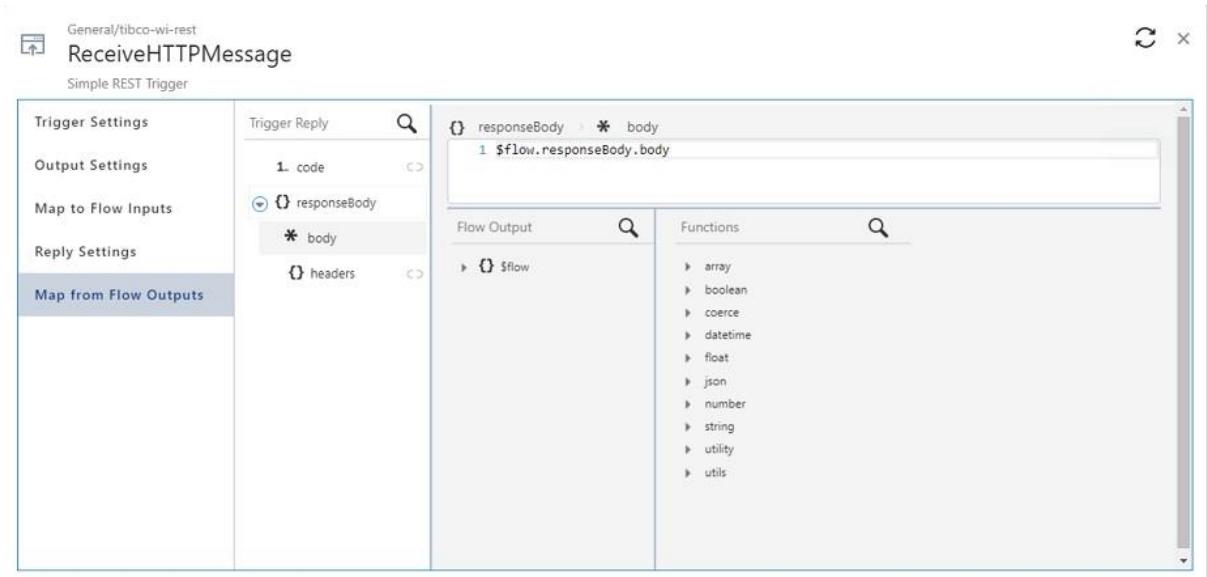
2. To enable data to flow between the trigger and the flow, you first need to map the trigger output to the flow input, and flow data to the trigger reply.

Click on the icon to open the configuration of the **ReceiveHTTPMessage** trigger.

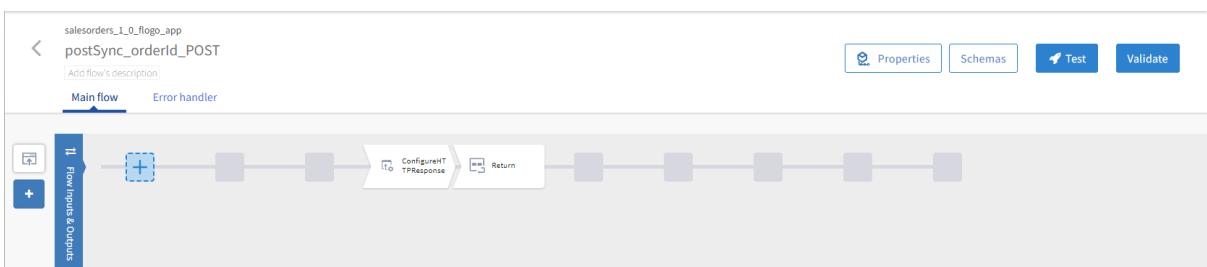
3. Check the auto mapped mappings from the trigger output to the flow input as follows:



- Also check the mapping of the flow data to the trigger reply as follows:

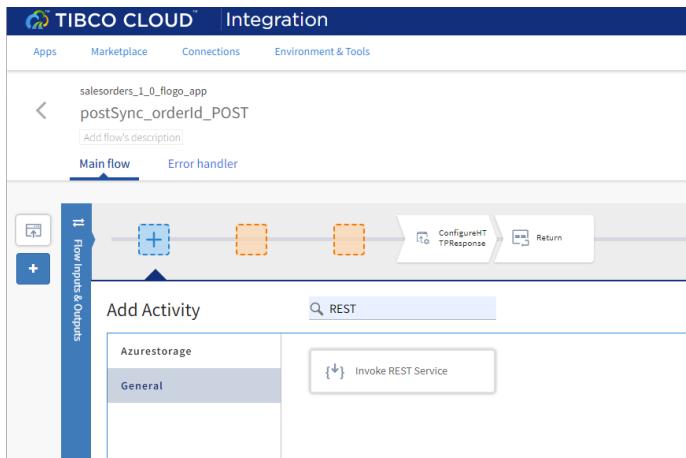


- Close the trigger configuration screen, and move the **ConfigureHTTPResponse** and **Return** tiles at least 3 positions to the right.



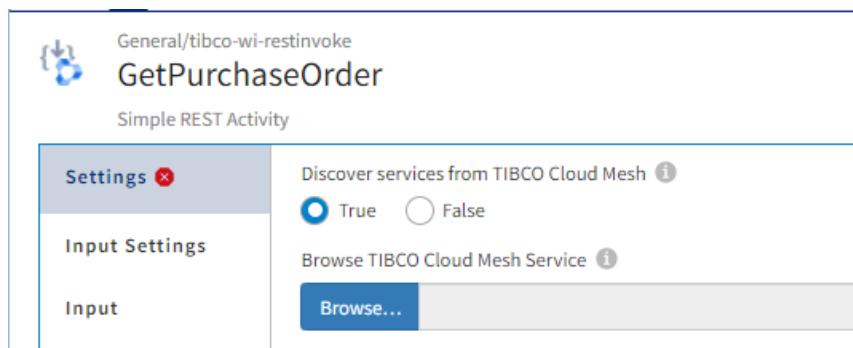
Creating the Activities

- Create a first activity, by clicking on the left-most + and navigate through **General > Invoke REST Service**.

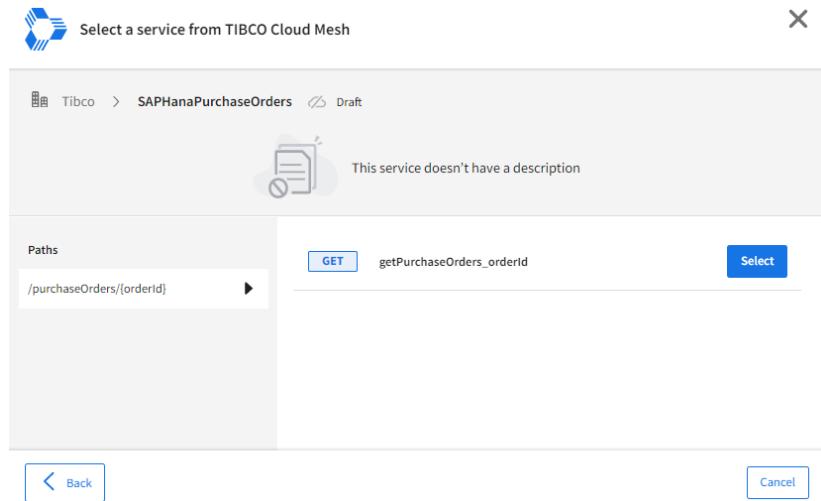


7. Configure this activity as follows:

- Give it a name: **GetPurchaseOrder**.
- The SalesOrder Service that connects to SAP Hana, we will retrieve from the Tibco Cloud Mesh. Select the **True** button for Discovering Services from the Tibco Cloud Mesh. Click **Browse..**

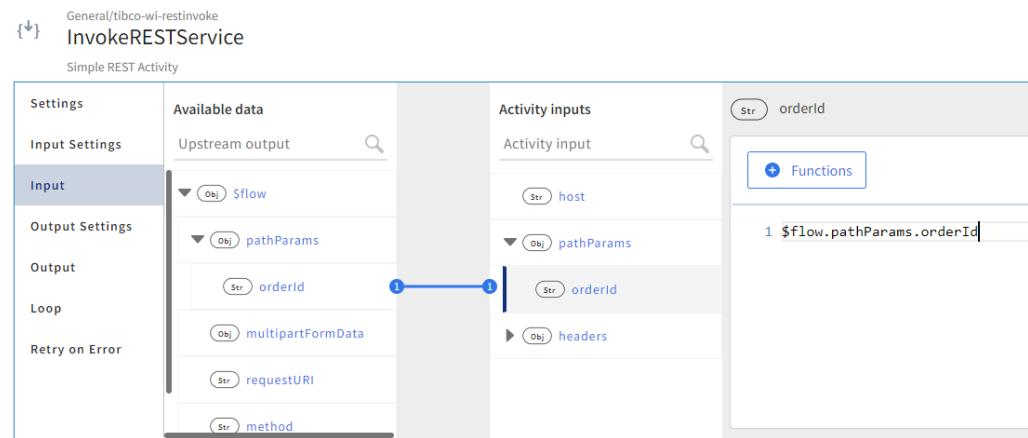


- Click on the **SAPHanaPurchaseOrders** API Definition from the list. In the next wizard screen **select** the operation.

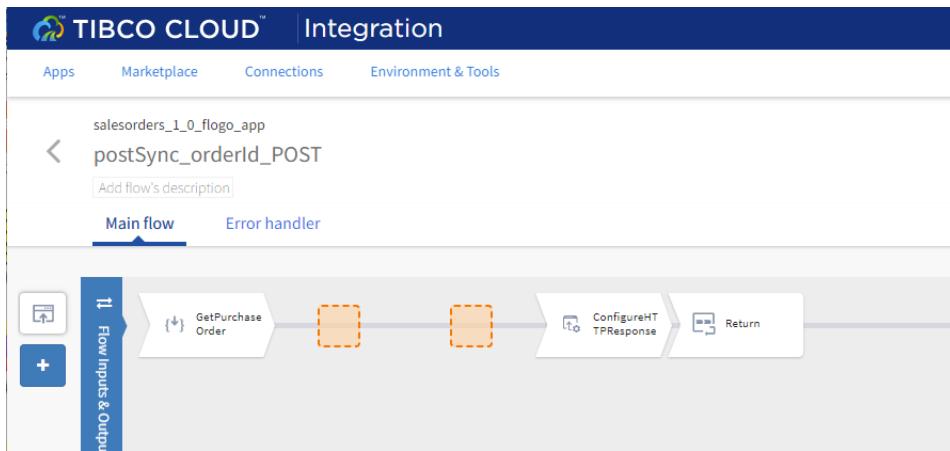


- In the **Input** section, create the following mapping:

Activity Input	Upstream Output
pathParams.orderId	\$flow.pathParams.orderId

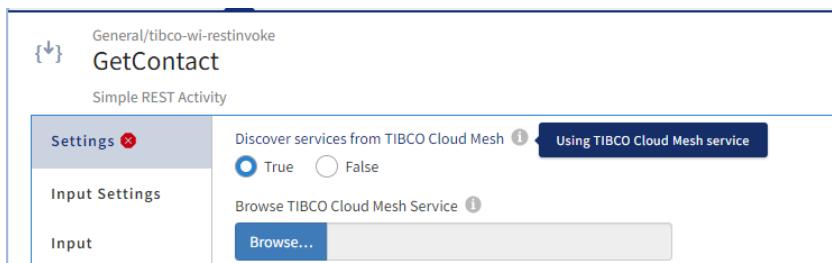


- Close the activity configuration screen, the integration should now look like the picture below.

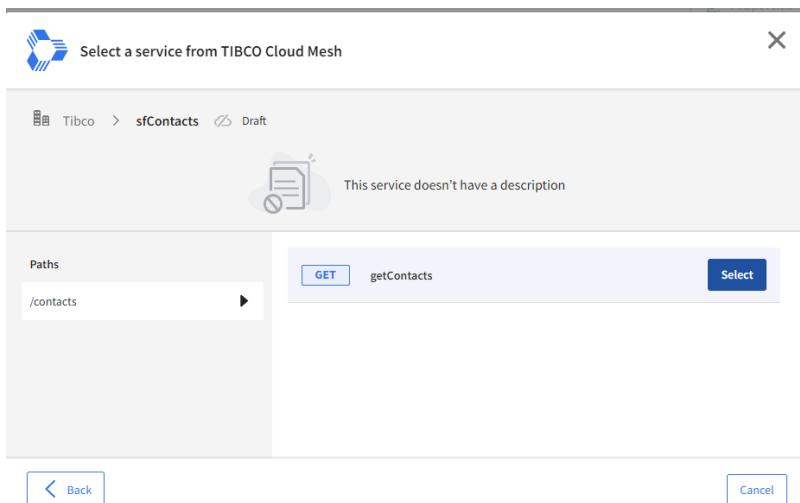


9. Now create a second activity, by clicking on the next + and navigate through **General > Invoke REST Service**. Configure this activity as follows:

- Give it a name: **GetContacts**.
- The Contact Service that connects to SalesForce, we will retrieve from the Tibco Cloud Mesh. Select the **True** button for Discovering Services from the Tibco Cloud Mesh. Click **Browse..**



- Click on the **sfContacts** API Definition from the list. In the next wizard screen select the operation.

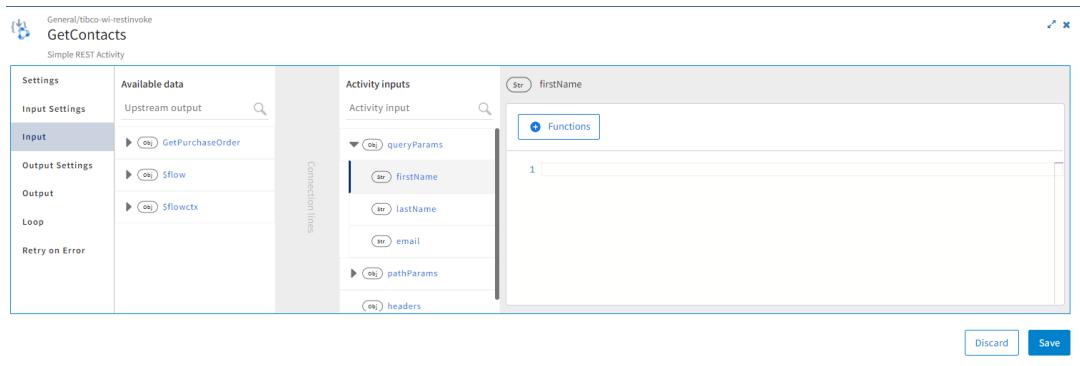


- In the **Input Settings** section, check the imported query parameters.
- In the **Input** section, create the mapping.

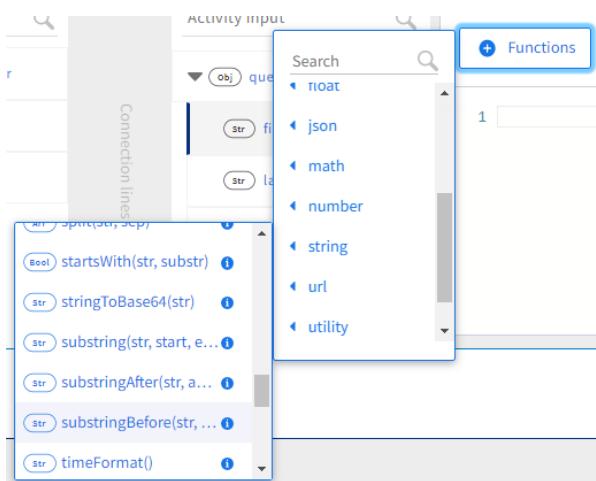
Activity Input	Upstream Output
<code>queryParams</code>	<code>string.substringBefore(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":")</code>
<code>queryParams</code>	<code>string.substringBefore(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")</code>
<code>queryParams</code>	<code>string.substringAfter(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")</code>

- The first mapping we will create manually using the Function wizard.

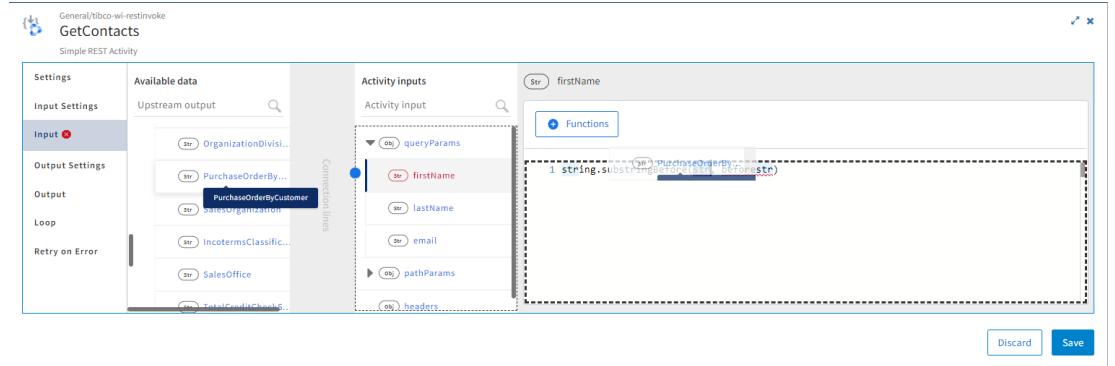
1. Select the *Activity Input : queryParams.firstName* and click on the Functions button.



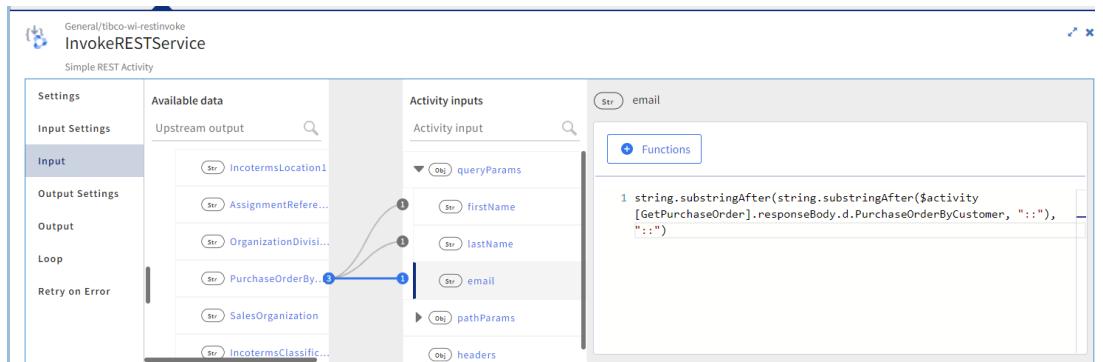
2. In the Functions dropdown Search in the string functions for the **substringBefore** function and select this to drop it to the mapping area.



- Set the first string **str** parameter to the PurchaseOrderByCustomer field from the responsebody of the GetPurchaseOrder Activity.



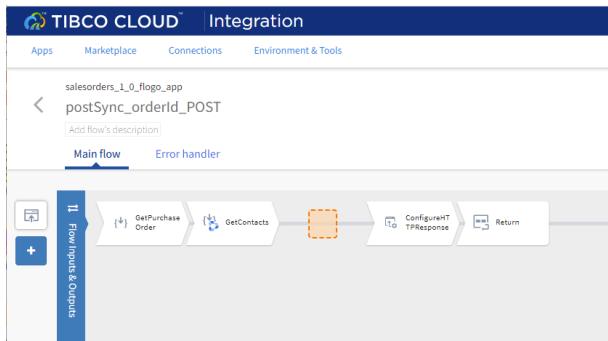
- Set the second string **beforestr** to ":", by typing into the mapping.



- Now create the other two mappings for the other two fields, by copy and pasting from this table.

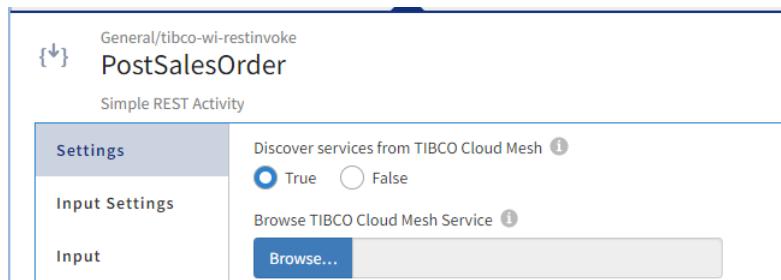
Activity Input	Upstream Output
queryParams	string.substringBefore(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")
.lastName	string.substringBefore(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")
queryParams	string.substringAfter(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")
.email	string.substringAfter(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")

- Close the activity configuration screen, the integration flow now should look like this

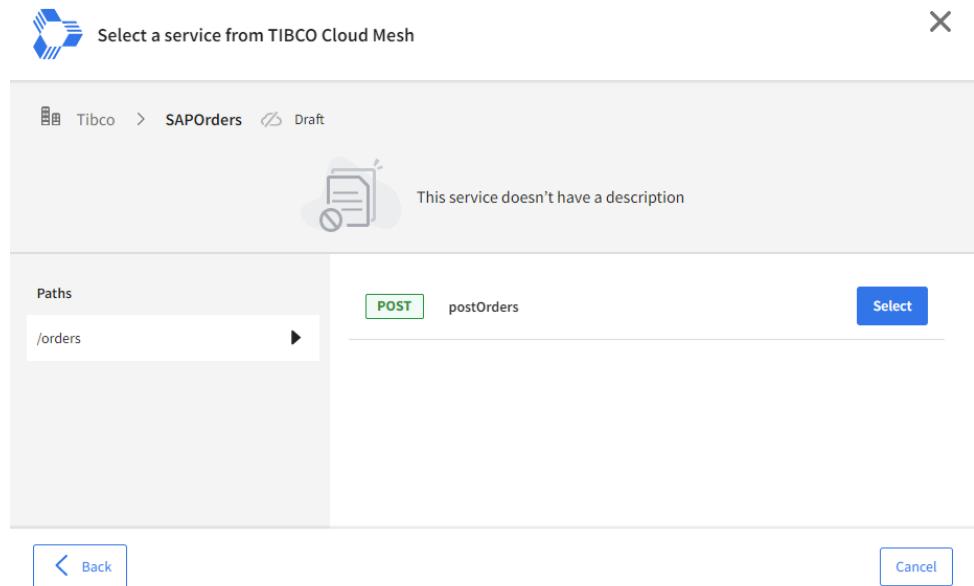


11. Now we create the third activity, which corresponds to the SAP Order Creation Service. Click on the next + and navigate through **General > Invoke REST Service**. Configure this activity as follows:

- Give it a name: **PostSalesOrder**.
- The Sales Order Create Service that connects to SAP, we will retrieve from the Tibco Cloud Mesh. Select the **True** button for Discovering Services from the Tibco Cloud Mesh. Click **Browse..**



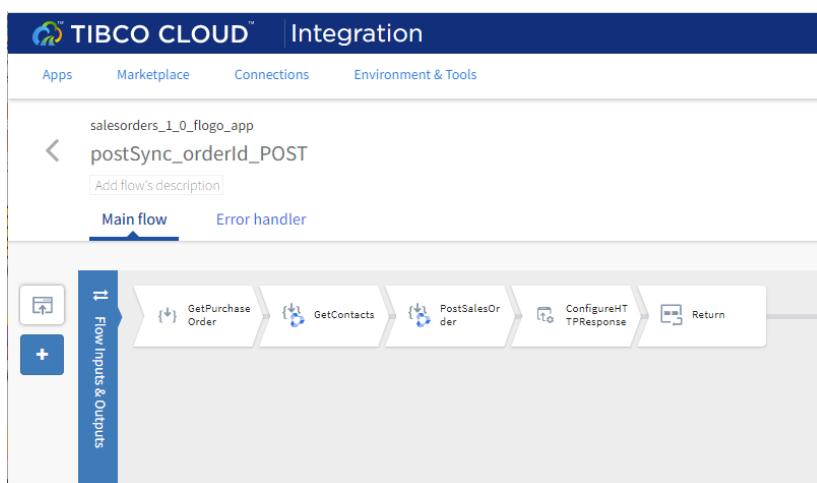
- Now select the SAPOrders from the list and select the postOrders Operation.



- In the **Input** section, create the following mapping:

Activity Input	Upstream Output
<code>body.Email</code>	<code>\$activity[GetContacts].responseCodes["200"].email</code>
<code>body.firstName</code>	<code>\$activity[GetContacts].responseCodes["200"].firstName</code>
<code>body.item</code>	<code>"1"</code>
<code>body.lastName</code>	<code>\$activity[GetContacts].responseCodes["200"].lastName</code>
<code>body.phone</code>	<code>\$activity[GetContacts].responseCodes["200"].mobile</code>
<code>body.price</code>	<code>\$activity[GetPurchaseOrder].responseCodes["200"].d.TotalNetAmount</code>

12. Close the activity configuration screen. The flow now looks similar to the following picture.



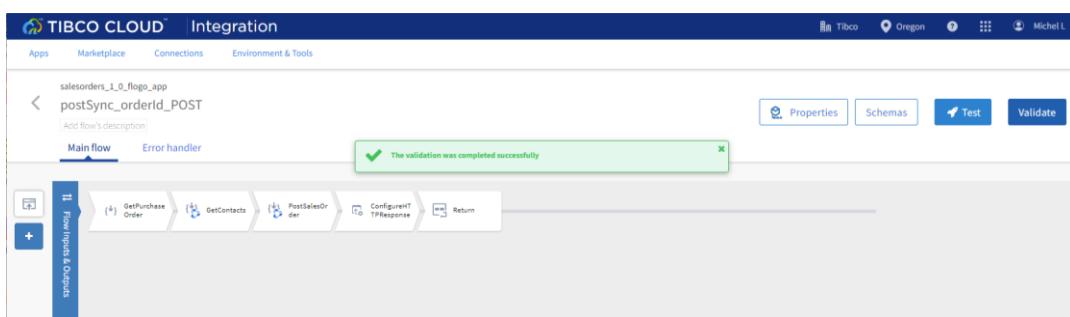
13. Now configure the **ConfigureHTTPResponse** activity as follows:

- In the **Input** section, create the following mapping:

Activity Input	Upstream Output
code	200
Input.body._response_string	\$activity[PostSalesOrder].responseCodes["200"]._response_string

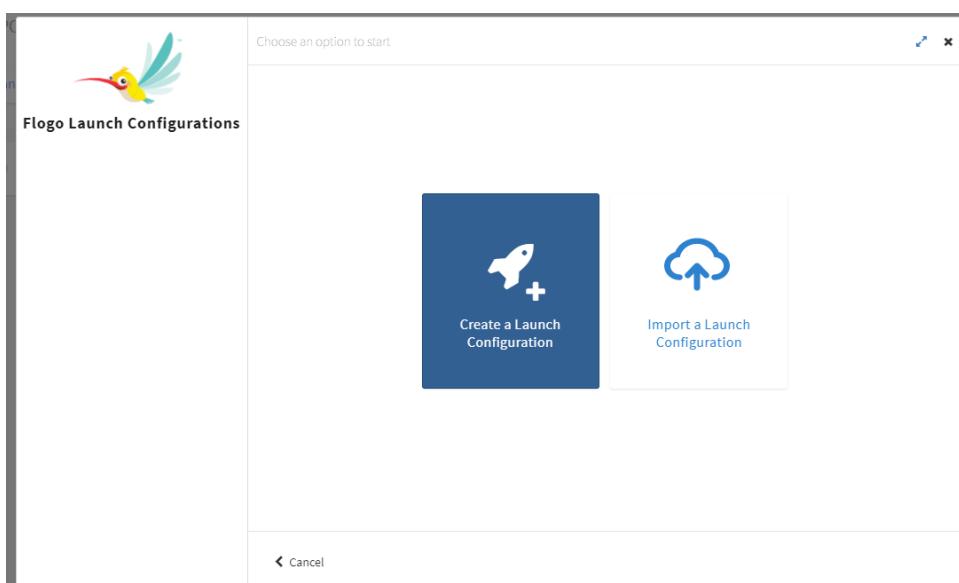
14. Close the activity configuration screen.

15. Now Validate the Integration, the result should look something like:

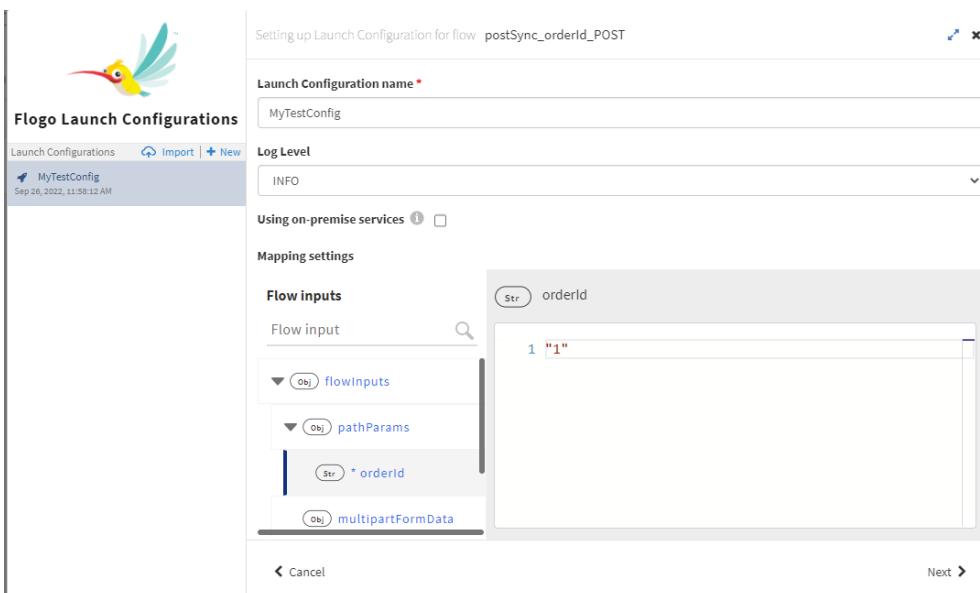


3.2.4 Push the Integration App to TIBCO Cloud and Test It

1. Click on the Blue **Test** Button to start the Test configuration wizard. First you need to create a Launch Configuration for this test.



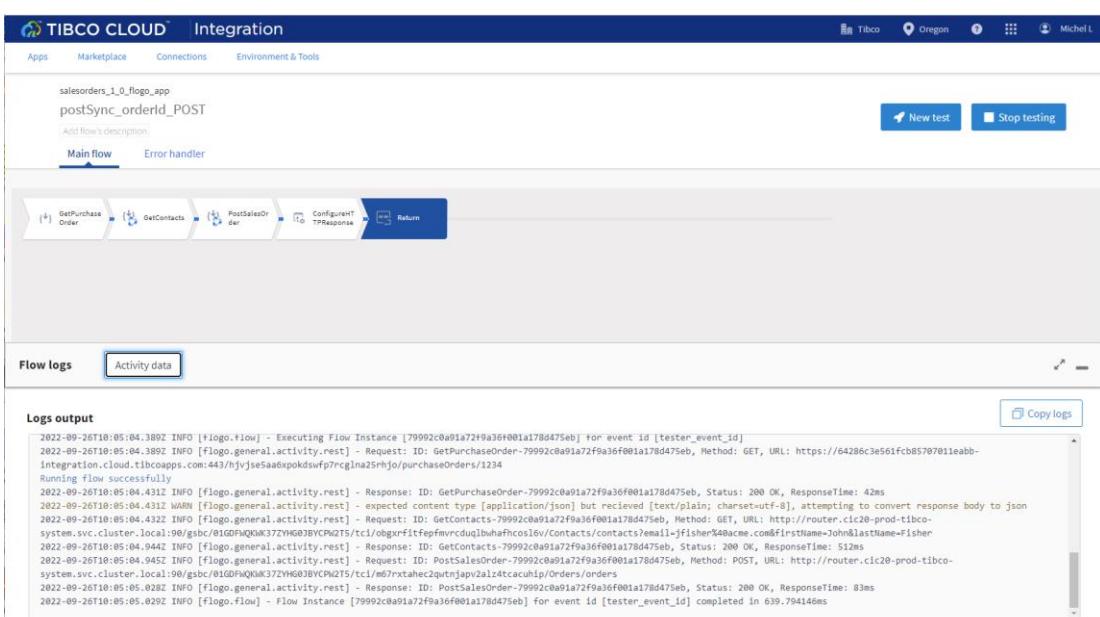
- Click on the corresponding tile and give your configuration a name. Create a mapping for the integration input parameter **orderId** as shown below.



3. Click the **Next** button in the right lower corner and check the summary. Now click the Run button in the bottom right corner to start the test run.



4. Now the integration flow test log will open and show the status of the test run.



5. You can introspect individual service and API calls on input and output result, by just clicking on the flow diagram and select the activity you want to inspect.

The screenshot shows the TIBCO CLOUD Integration interface. At the top, there's a navigation bar with 'TIBCO CLOUD' logo, 'Integration', 'Apps', 'Marketplace', 'Connections', 'Environment & Tools', and user information ('Tibco', 'Oregon', 'Michel L'). Below the navigation is a search bar with 'salesorders_1_0_flogo_app' and a 'postSync_orderId_POST' entry. A 'Main flow' tab is selected, showing a sequence of activities: 'GetPurchaseOrder' (with a warning icon), 'GetContacts' (selected, highlighted in blue), 'PostSalesOrder', 'ConfigureTTPResponse', and 'Return'. To the right of the flow are 'New test' and 'Stop testing' buttons. Below the flow is a 'Activity data' section with tabs for 'Activity logs' (selected) and 'Flow logs'. Under 'Activity logs' for the 'GetContacts' activity, there's a 'Simple REST Activity' section with a 'Run test from this activity' button. On the left, there's a code editor showing the 'Inputs' and 'Outputs' for the 'GetContacts' activity. The 'Inputs' code is:

```

1 {
2   "queryParams": {
3     "email": "jfisher@acme.com",
4     "firstName": "John",
5     "lastName": "Fisher"
6   },
7   "pathParams": {},
8   "headers": {}
9 }

```

The 'Outputs' code is:

```

1 {
2   "statusCode": 200,
3   "responseCodes": {
4     "200": {
5       "email": "jfisher@acme.com",
6       "firstName": "John",
7       "lastName": "Fisher",
8       "mailingCity": "Philadelphia",
9       "mailingCode": "19150",
10      "mailingState": "PA",
11      "mailingStreet": "431 Chestnut Street",
12      "mobile": "1231231234"
13    }
14  }
15 }

```

To push the Integration app to the runtime environment and test it, do the following:

1. Click the **Push app** button.

The screenshot shows the deployment interface. At the top, there's a header with 'Tibco', 'Oregon', and user information ('Michel L'). Below the header, there's a summary row with '0 Instances' and a 'Deploying' status indicator. At the bottom, there are several buttons: 'Properties', 'Schemas', 'Validate', 'Push' (which is highlighted in blue), and a more options button '...'. The 'Push' button has a tooltip 'Push to runtime environment'.

2. Now we have to scale the application to start an instance of the integration

The screenshot shows the application management interface. It displays basic info about the app: 'salesorders_1_0_flogo_app', 'Owned By: Michel Leijdekker', 'TIBCO Cloud v: 1.0.0', 'Draft', 'Public', '+Tags'. It also shows the last modified date ('Last modified at 26 Sep 2022 11:48 am') and creation details ('Created by: Michel Leijdekker | Application description'). At the bottom right, there's a 'Scale' button with a value of '1' and a status of 'Stopped'.

3. Once the **salesorders_1_0_flogo_app** app is running, click on the app and select the **Endpoint** tab. Select **Test** from the possible actions

- Click on the green POST operation to open the API details.

- Test the **salesorders_1_0_flogo_app** app by clicking on the **Try it out!** button. Fill out a value in the **orderId** field, and click on the blue Execute bar. Now watch the result.

The screenshot shows the Tibco Business Works API endpoint configuration interface. The 'Endpoints' tab is selected. A parameter 'orderid' is defined as a required string path parameter with value '1'. Below the parameters are 'Execute' and 'Clear' buttons. The 'Responses' section shows a curl command, request URL, and server response body.

You now have successfully build the Sales Orders integration, that uses Tibco Business Works integrations, API definitions and Mock Applications.

3.3 Additional Reading

BusinessWorks (Cloud Integration)

- [TIBCO Cloud Integration: Getting Started with BusinessWorks Applications](#)
- [TIBCO Business Studio™ - Cloud Edition](#)
- [TIBCO ActiveMatrix BusinessWorks™ Plug-ins](#)
- [TIBCO® Cloud Integration Frequently Asked Questions](#)

Flogo (Cloud Integration)

- [TIBCO Cloud Integration: Getting Started with Flogo](#)
- [TIBCO Flogo® apps](#)
- [TIBCO Flogo® Connectors](#)

Project Flogo

- [Project Flogo™ Community Wiki](#)
- [Project Flogo](#)
- [Project Flogo Documentation](#)
- [Project Flogo Github Repos](#)

Extra LAB: Business Events Simple Decision Service

Simple Decision Services (SDS) is a capability of **TIBCO Cloud Events. (TCE)**

SDS is an easy and simplified approach to create stateless decision tables. It has browser-based authoring experience without any complexity.

Business (or non-technical) users can now easily define their business logic in the form of simple conditions and actions in an excel like format called as a "Decision Service". These services can be exposed as RESTful APIs for internal and external consumption.

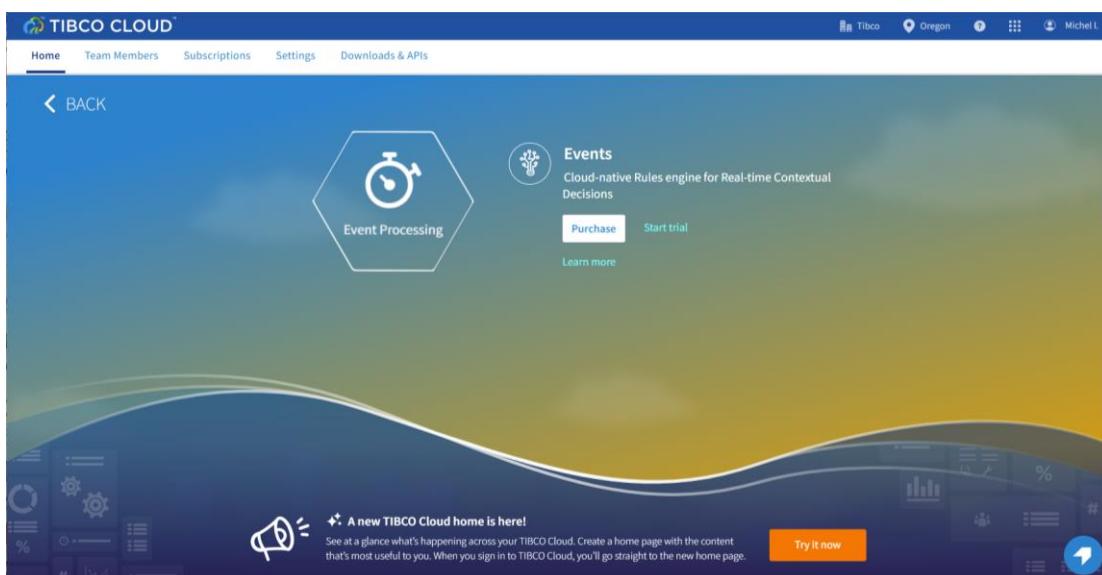
This walkthrough provides a step-by-step journey to create a Simple Decision Service. You can have one or more Simple Decision Services as part of a single TCE application.

What You Will Learn

- Understanding the concept of Simple Decision Service(s).
- How to create, deploy and test a Simple Decision Service(s).

Getting Ready

1. Extend your trial with a Business Events trial. In the Home screen, navigate to the Event Processing hexacon and Start trial.



- Follow the necessary registration steps. After registration, the Cloud Event environment opens in the window. Close this window and return to the Home menu. This should now show the Event Processing as opened.



Creating a Base TCE Project

- Login to TIBCO Cloud Events and select the Org you want to use.
- Click the Push App button (on the top right side).



Figure 1: Push App Button

- On clicking the Push App button, select the Decision Service tab in the wizard shown below.
- Select the desired Application Size, No.of Instances and the Name of the Application before pushing the application using the Push button.

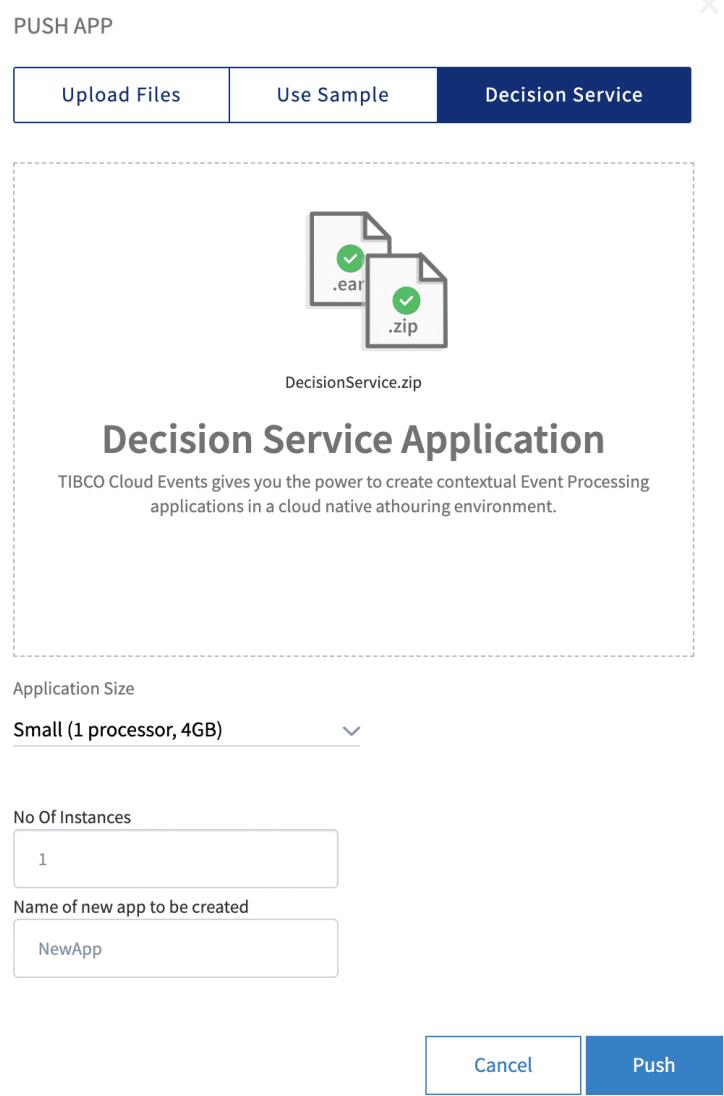


Figure 2: SDS - Push Application

- Once you Push the application, it takes a few seconds to Scale and come to the Running state. The Application itself is in the Draft stage scaled to the number of instances mentioned during deployment.



Figure 3: Application Push Stages



Figure 4: Application Deployment

- Now the base TCE project to create a SDS is available and deployed. Let's proceed ahead to implement a SDS now.

Implementing a Simple decision service

- Leverage TIBCO Cloud Events WebStudio to implement a Simple Decision Service.
- Click the WebStudio button (on the top right) to begin the implementation steps.

Figure 5: TIBCO Cloud Events Home - WebStudio option

- Within the WebStudio, click + sign on the Project Explorer tab (on the left) and then "Create a new decision service" option to add a new implementation of SDS.

Figure 6: Create a new Decision Service option

4. Provide the required input details as prompted by the wizard. Project name from the dropdown list are the ones already deployed as part of the previous step. If the project is not previously checked out, it will be checked out during the SDS creation process.
5. Enter the input and output fields in the following wizards. The inputs and outputs are defined in the form of a simple name and data type format.

Figure 7: Create Decision Service wizard

Figure 8: Input fields of SDS

The screenshot shows the 'DECISION SERVICE' configuration page. At the top, there are tabs for 'Decision Service Information', 'Decision Service Input', and 'Decision Service Output'. The 'Decision Service Output' tab is active, indicated by a blue circle with a number '3'. A sub-header says 'The final step is to define the output fields of your Decision Service'. Below this, a section titled 'Output Fields' contains a table with one row: 'Result' (Field Name), 'String' (Field Type), and an empty 'Description' field. To the right, a 'Field Result' panel asks if the field will contain a list of values, with a note that no value restrictions have been defined.

Figure 9: Output fields of SDS

6. Click Finish once the input and output fields are defined to create the default implementation of a SDS.
7. Click on Add row button to add the desired conditions (inputs) and actions (outputs).

The screenshot shows the 'NewAppSDS_defaultImpl' editor. The title bar indicates the artifact type is 'Decision Service Implementation'. The main area displays a 'Decision Table' with three rows and two columns. The columns are labeled 'DT' and 'CONDITION | ACTION'. The rows are numbered 1, 2, and 3. Row 1: 'input.Marks (int)' < 40, 'Fail'. Row 2: '>40', 'Pass'. Row 3: '=40', 'Pass'. On the right side, there are columns for 'Priority' (values 5, 5, 5) and 'Columns' (with a 'Filters' button). A toolbar at the top provides save, commit, undo, redo, and validate options.

Figure 10: Decision Service Implementation

Once the rows are defined, click Save button - this will enable the Commit option to commit the changes for an Approval process.

- ❖ Commit the changes with appropriate message for the Admin user to approve the changes. The changes need to be committed and approved before they can be deployed to the TCE application.

Deploying the Decision Service

1. After Commit action, the changes are available for Admin Approval process.

The screenshot shows the TIBCO Cloud Events WebStudio interface. At the top, there's a navigation bar with links like 'Events', 'WebStudio', 'Resources', and 'Dashboard'. Below the navigation is a header bar with 'TIBCO CLOUD' logo, 'Events WebStudio', and user information ('Oregon', 'Arpita T'). The main content area is divided into sections:

- Recent User Activity:** Shows a log of recent commits. One entry for 'NewApp' by 'Tirodkar, Arpita' is highlighted with a yellow background, indicating it has been committed. Buttons for 'Reject' and 'Approve' are visible at the bottom of this card.
- Project Snapshots:** Displays project details for 'NewApp', 'NewTest', 'AlertProcessing', and 'CreditCardApplication'. Each project card includes metrics like 'Unresolved worklist item(s)', 'Deployable artifact(s)', and file counts.
- Deployment Metrics:** Global counts for 'total unresolved item(s)' (3) and 'total deployable artifact(s)' (14).

Figure 11: Deployment Dashboard

2. Admin user can click on the eye button on the left side under the Recent User Activity log to Approve/Reject the committed changes.
3. Once the changes are approved, under the Project Snapshots section, left click on the Option menu (3 dots icon on the right side specific to your Project) and select "Generate Deployable" as shown below.

The screenshot shows the TIBCO CLOUD Events WebStudio dashboard. At the top, there's a search bar with the placeholder "Welcome back, ArpitA! What would you like to do today?". To the right, there are two circular status indicators: one red with '2' labeled "total unresolved item(s)" and one blue with '15' labeled "total deployable artifact(s)". Below the search bar are three main sections: "Recent User Activity", "Project Snapshots", and "AlertProcessing". The "Recent User Activity" section shows a list of recent commits, including one from "Tirodkar, ArpitA" at 2:04 PM. The "Project Snapshots" section shows two projects: "NewApp" and "NewTest". "NewApp" has 1 unresolved worklist item and 1 deployable artifact. "NewTest" has 1 unresolved worklist item and 15 deployable artifacts. The "AlertProcessing" section shows no commits found for this project.

Figure 12: Generate Deployable

4. This step generates a deployable (an EAR file) for the project and hot deploys the changes to the running engine. No need to redeploy the changes.
5. Click on "Back to Home Page" option on the top right to view the endpoints of SDS.

The screenshot shows the TIBCO CLOUD Events WebStudio dashboard. A yellow warning box appears in the center with the message: "Deployable generated successfully and hot deployed to running engine. [Note: If the services in this project have changed since the last deployment, a manual re-scale of the application is required for these changes to take effect]". Below the message, there are three main sections: "Recent User Activity", "Project Snapshots", and "AlertProcessing". The "Project Snapshots" section shows the "NewApp" project again, indicating it has been successfully deployed.

Figure 13: Home page - view Endpoints option

The next section describes the steps for Testing the Decision Service.

Testing the Decision Service

Once the deployable is generated and deployed, the decision service can be tested using the exposed REST API Endpoints.

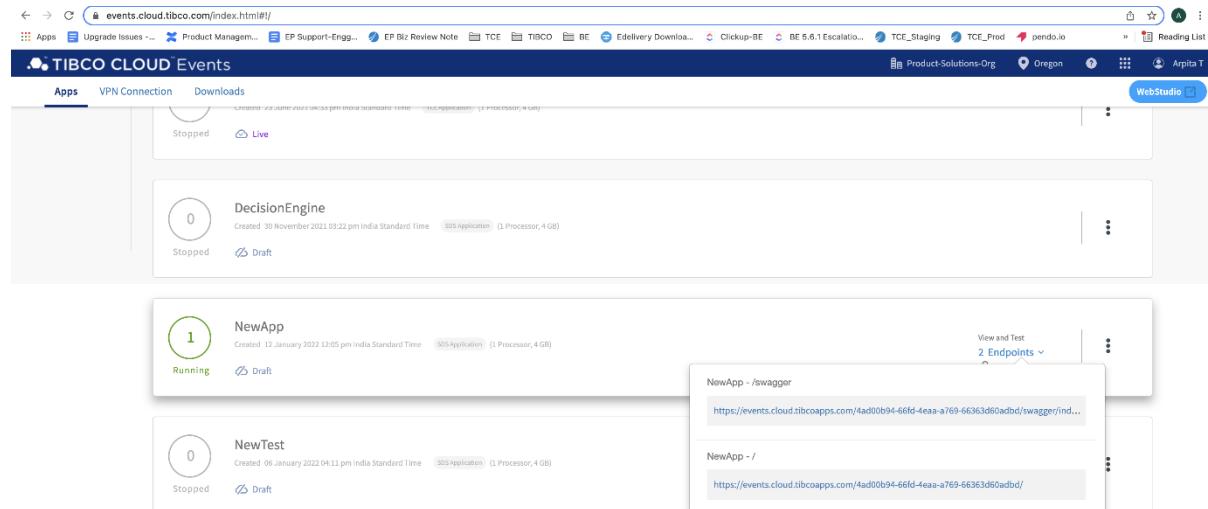


Figure 14: REST Endpoints

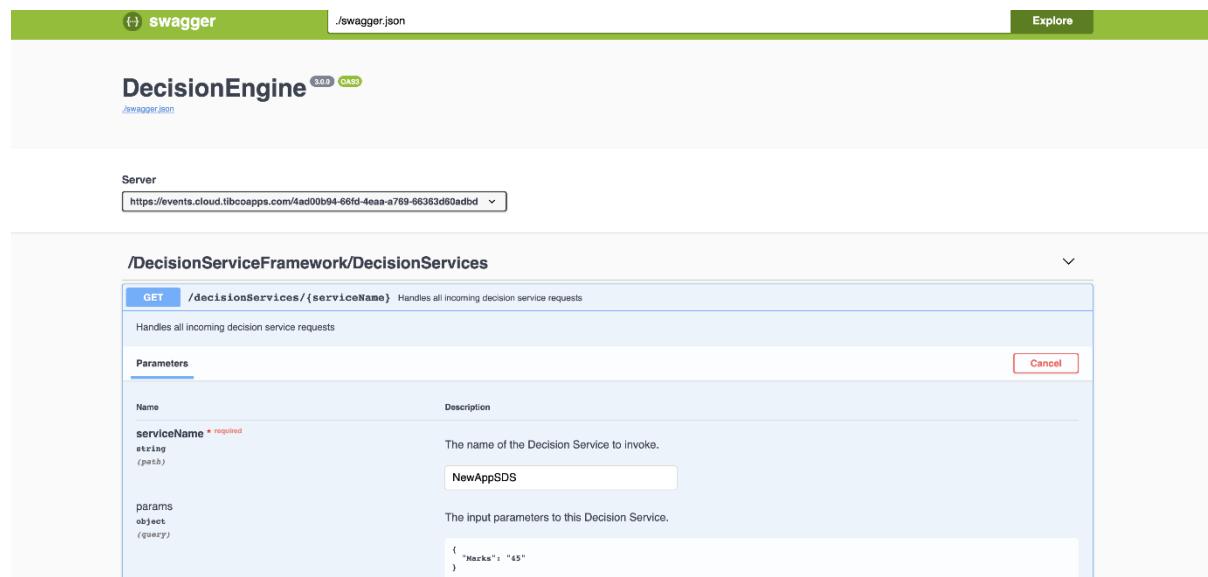


Figure 15: Swagger Client

Note:

- The serviceName is case-sensitive and expects the name as entered while creating the decision service in the WebStudio.

- The input and output are both in the JSON format.

Server response

Code	Details						
200	<p>Response body</p> <pre>{ "attributes": { "id": 6, "type": "www.tibco.com/be/ontology/DecisionServiceFramework/Events/ServiceResponse" }, "message": "The service matched 1 row(s)", "payload": { "response": { "attributes": { "type": "www.tibco.com/be/ontology/DecisionServiceFramework/Events/ServiceResponse" }, "outputs": [{ "attributes": { "id": 2, "type": "www.tibco.com/be/ontology/DecisionServices/NowAppSDS/NowAppSDSResponse" }, "Result": "Pass" }] } } }</pre> <p>Download</p> <p>Response headers</p> <pre>access-control-allow-origin: * cache-control: max-age=0, no-cache, no-store, must-revalidate content-length: 103 content-type: application/json; charset=utf-8 date: Thu, 13 Jan 2022 11:33:23 GMT expires: - pragma: no-cache strict-transport-security: max-age=31536000; includeSubDomains; preload x-content-type-options: nosniff x-frame-options: sameorigin x-xss-protection: 1; mode=block</pre> <p>Responses</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>200 success</td> <td>No links</td> </tr> </tbody> </table>	Code	Description	Links	200	200 success	No links
Code	Description	Links					
200	200 success	No links					

Figure 16: Swagger Output

Summary

- As you've noticed, creating a Simple Decision Service via TIBCO Cloud Events is straightforward and provides a strong rules and decisioning capability via REST APIs which can be easily integrated with any other systems or applications within an organization.
- For any feedback, queries or additional information, please reach out to <mailto:support@tibco.com>

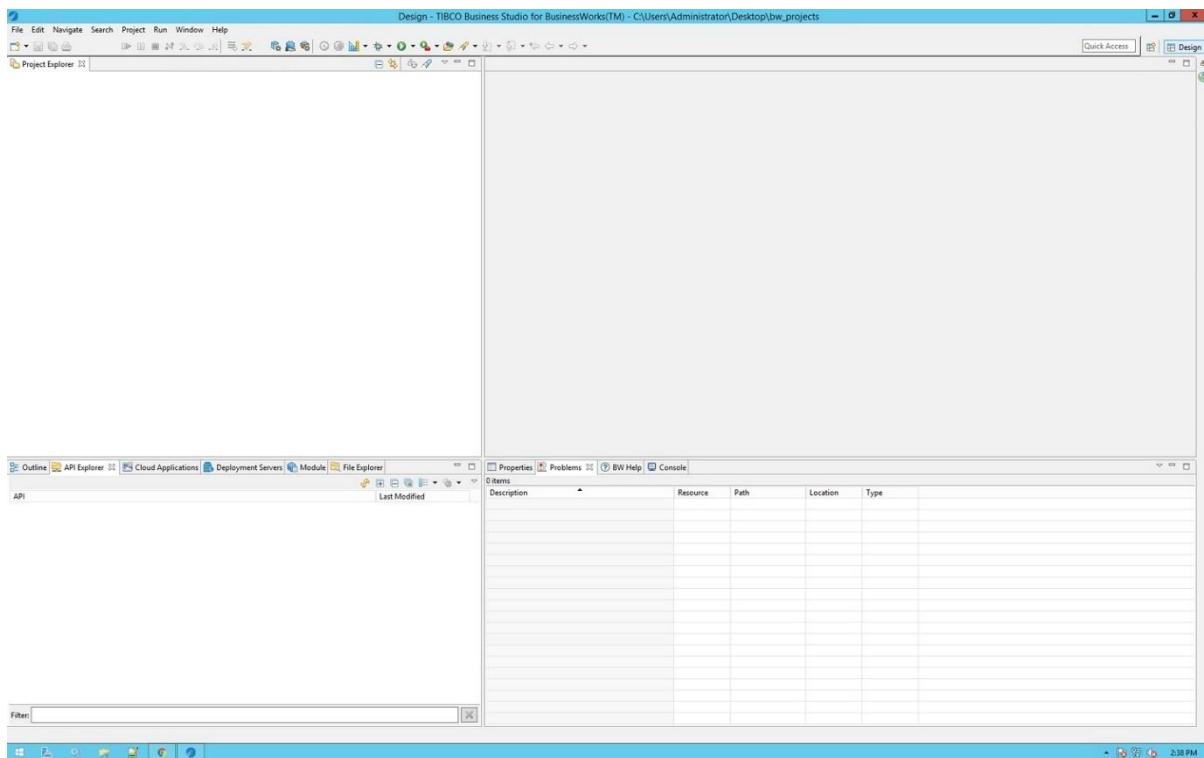
Extra Lab: BusinessWorks Apps in TIBCO Cloud Integration

In this lab, you'll first connect TIBCO Business Studio™ for BusinessWorks from your workstation to TIBCO Cloud. You'll then import a BusinessWorks project into Business Studio that implements a system API that gets purchase orders from SAP Hana. Finally, you'll push the BusinessWorks app to TIBCO Cloud and test it.

1 Getting Ready

To get ready, open TIBCO Business Studio™ for BusinessWorks.

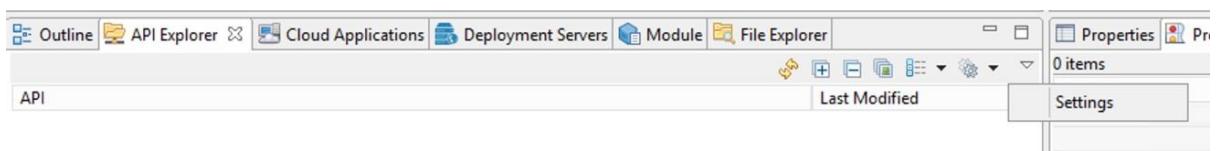
Assuming you start with a clean environment, your screen should look similar to this:



2 How to Do It: Connect Business Studio to TIBCO Cloud

To setup a connection from TIBCO Business Studio™ for BusinessWorks to TIBCO Cloud that enables you to push an app to the cloud directly from TIBCO Business Studio, do the following:

1. Go to API Explorer, click the down arrow in the upper right corner, and click Settings.



2. In the **Configure API Settings** form, click the **New** button. Fill out the **URL** and your TIBCO Cloud Integration username and password in the new form.

Field	Value
URL	<code>https://integration.cloud.tibco.com:443</code>
Username	<code>your@email.com</code>
Password	<code>YoUrPassWoRd</code>

3. Your screen should look like this:

The screenshot shows the 'Edit' dialog box for API Registry client configuration. The title bar says 'Edit'. The main area is titled 'Edit' and contains the sub-instruction 'Edit API Registry client configuration.'.

Fields in the dialog:

- Name: Cloud
- Type: Cloud Local Folder
- URL: https://integration.cloud.tibco.com:443
- Basic
- Authentication:
 - Oauth2 URL: https://sso-ext.tibco.com/as/token.oauth2
 - Username: @
 - Password: [REDACTED]
- Buttons at the bottom: ? (help), Finish, Cancel.

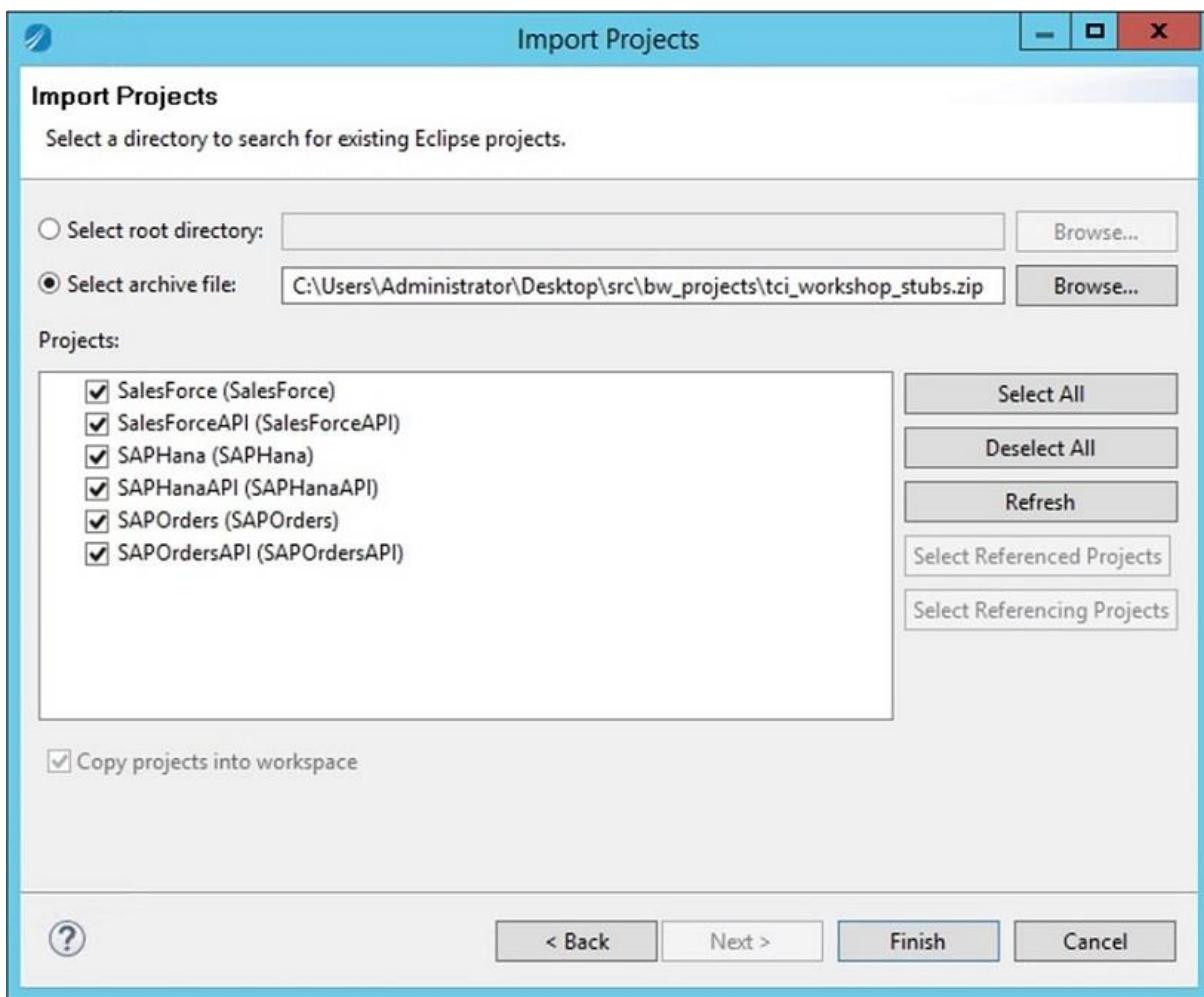
4. Click the **Finish** button when you done done, and click the **Close** button in the **Configure API Settings** form. The API Explorer should look like this:



3 How to Do It: Import a Project in to Business Studio

To import a BusinessWorks project into Business Studio that implements a system API, do the following:

1. From the menu, select **File > Import**. In the new form, select **General > Existing Studio Projects into Workspace**.
2. Select the [tci_workshop_stubs.zip](#) file from the `src/bw_projects` directory. Your screen should look like this:



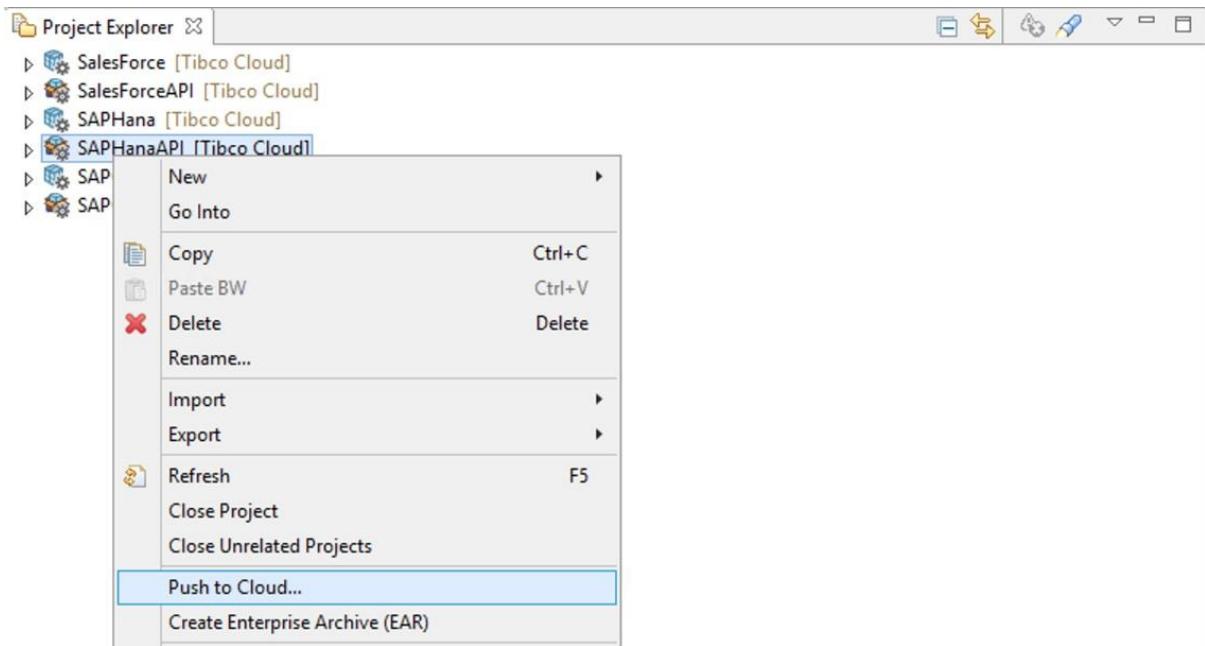
3. Click the **Finish** button. The Project Explorer should look like this:



4 How to Do It: Push the Project to TIBCO Cloud and Test

To push the SAP Hana system API app to TIBCO Cloud, do the following:

1. In the Project Explorer, select **SAPHanaAPI > Push to Cloud ...:**



2. Navigate to TIBCO Cloud, and once the app has been successfully pushed, your screen should look like this:

3. Once the **SAPHanaAPI** app is running, hover over the **Endpoint** link, and select **View and Test** from the menu
4. Test the **SAPHanaAPI** app by filling out a value in the **orderId** field, and clicking on the **Try it out!** button:

The screenshot shows the TIBCO CLOUD Integration platform. At the top, there's a navigation bar with tabs for 'Apps', 'API Specs', 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. The 'API Specs' tab is currently selected. Below the navigation bar, the page title is 'SAPHanaPurchaseOrders' with a 'Tester' button. A search bar labeled 'Search resources' is present. On the left, there's a sidebar with a tree view showing 'default' and 'purchaseOrders/{orderId}'. The main content area displays an 'Example Value' for a GET request to '/purchaseOrders/{orderId}'. The response body is a JSON object with fields like 'AssignmentReference', 'CompletedDeliveryIsDefined', 'CreatedByUser', 'CreationDate', 'CustomerAddress', 'CustomerPurchaseOrderDate', 'CustomerPurchaseOrderType', and 'DeliveryBlockReason'. Below the example value, there's a table for parameters:

Parameter	Value	Description	Parameter Type	Data Type
orderId	123		path	string

A 'Try it out!' button is located at the bottom of the parameters section.

5. Copy the URL, which has a pattern

like <https://integration.cloud.tibcoapps.com:443/xxxxx/PurchaseOrders/purchaseOrders/{orderId}>, and store it somewhere. You will need it when you implement the process API in the next lab.

