TIBCO™

# iPaaS Workshop

## Lab Instructions

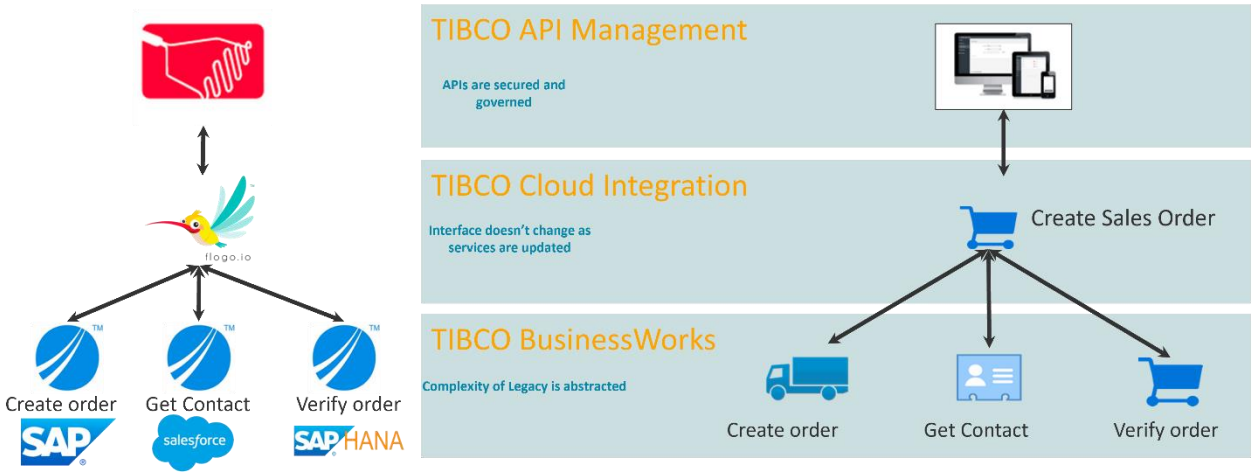
# Table of Contents

# 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.




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

TIBCO CLOUD™

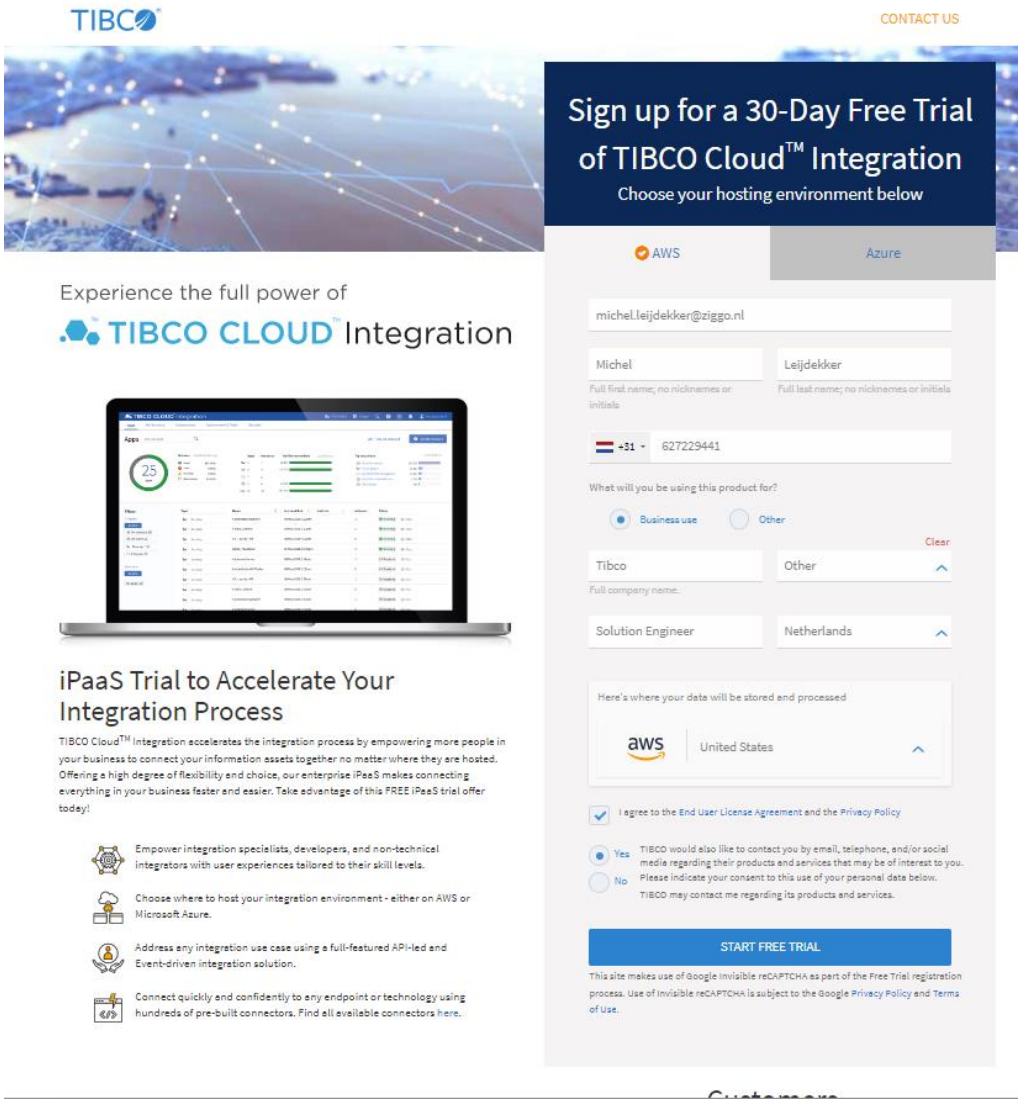
# 1. Validate Your Environment

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.

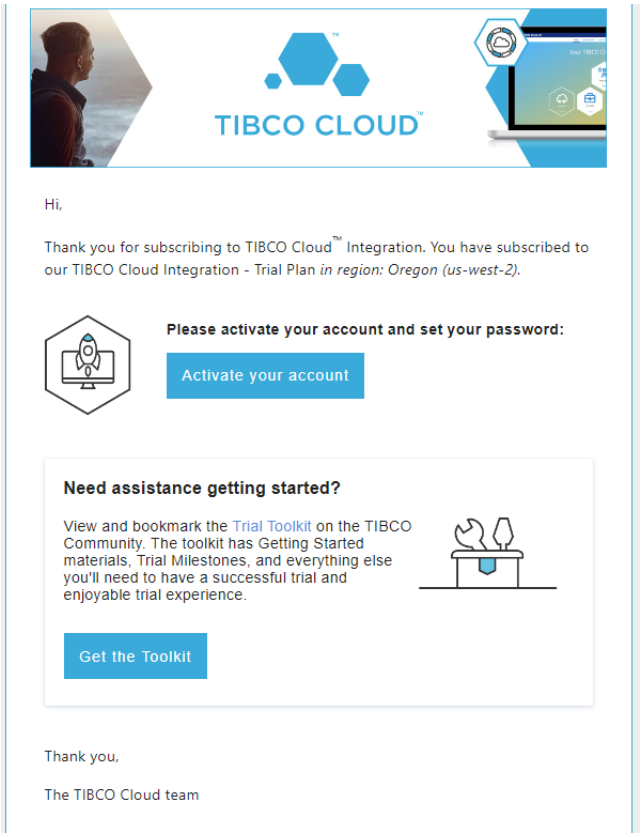
TIBCO Cloud™: [https://account.cloud.tibco.com/signup/tci](https://account.cloud.tibco.com/signup/tci)



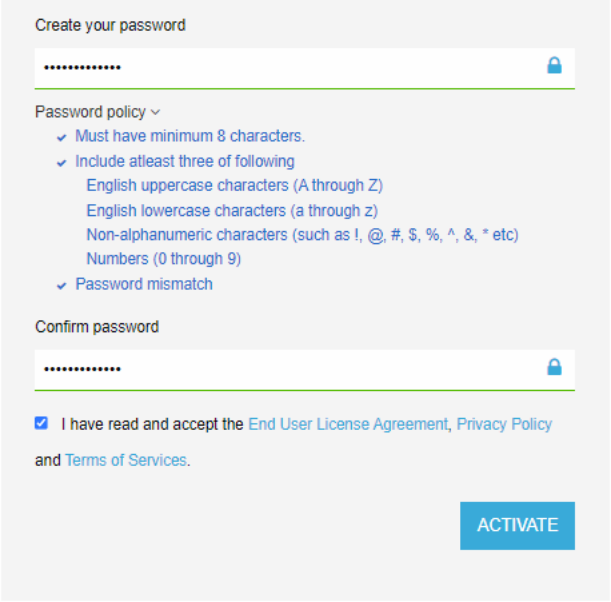

❖ choose the **"AWS United States"** as a region and click "**Start Free Trial**"

TIBCO CLOUD™

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



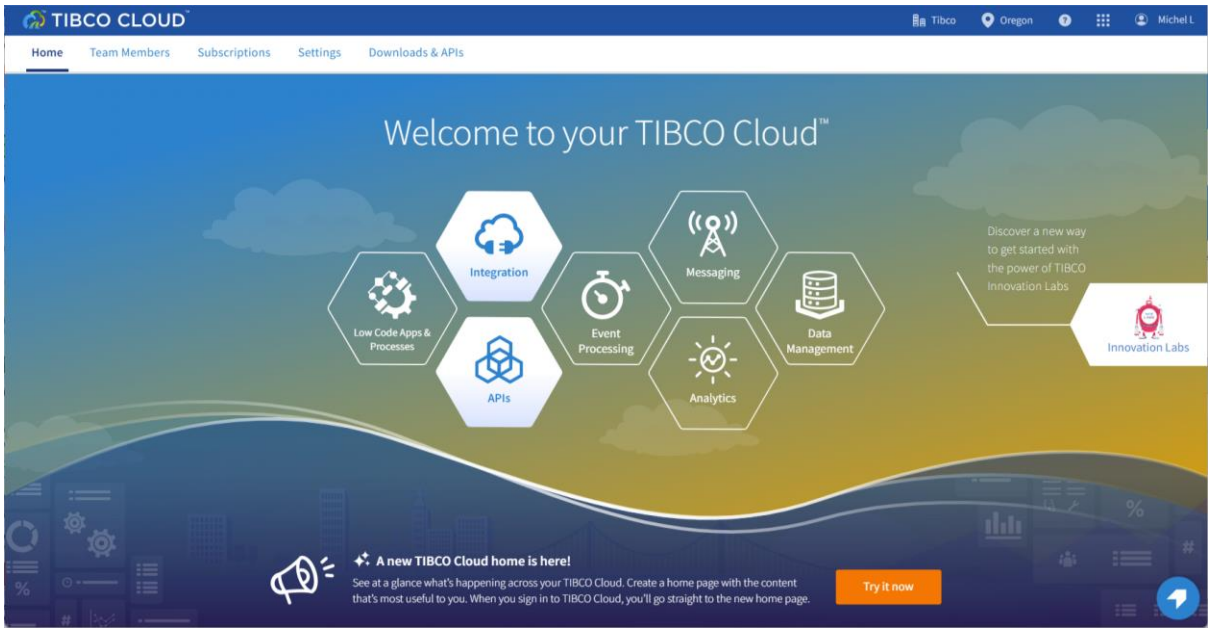

❖ Click the "Activate your Account" button and set your password in the form.



# 1.2 Sign in to TIBCO Cloud

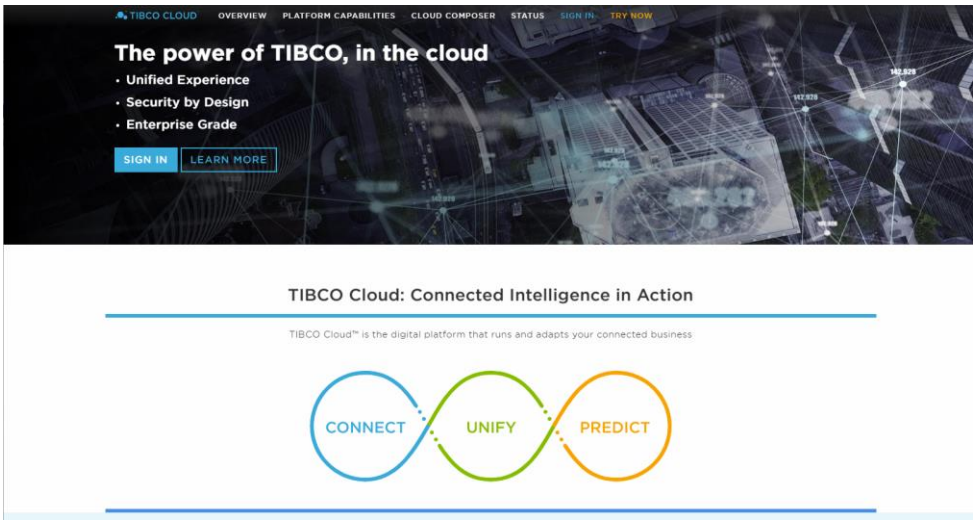
## 1.2.1 Getting Ready

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

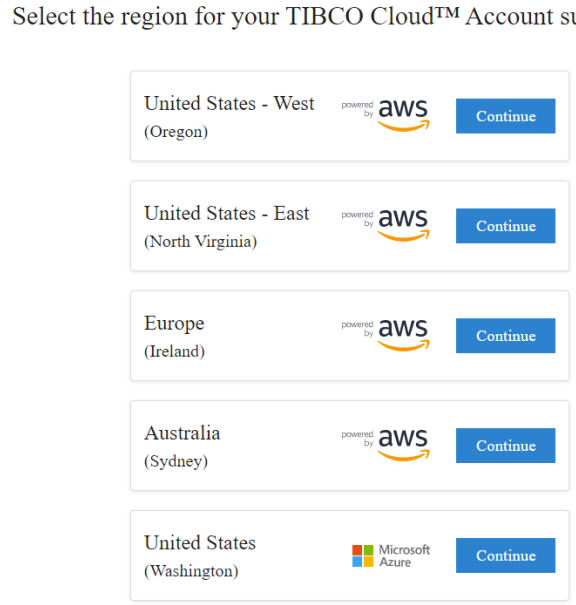



## 1.2.2 How to Do It

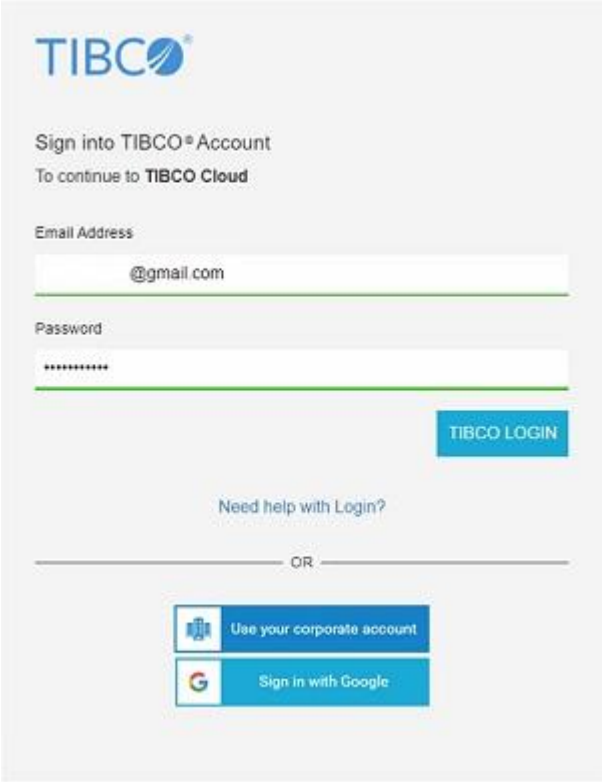
1. Browse to https://cloud.tibco.com/.



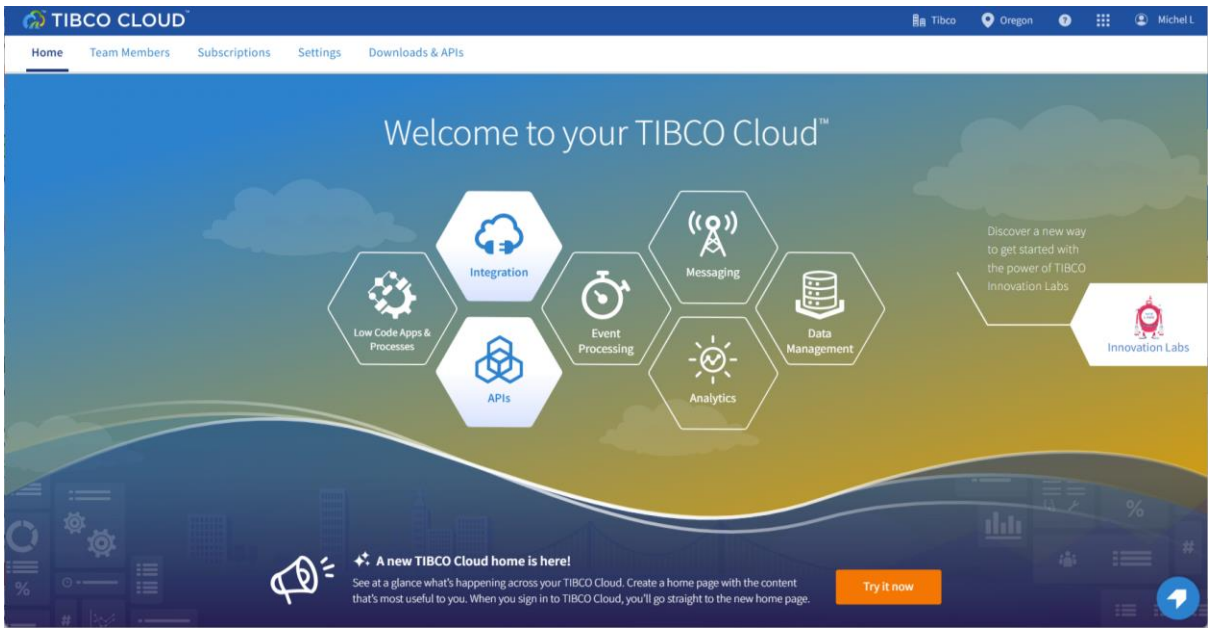

2. Click **SIGN IN** (at the right top) and select the **United States – West** Region




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



4. 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.

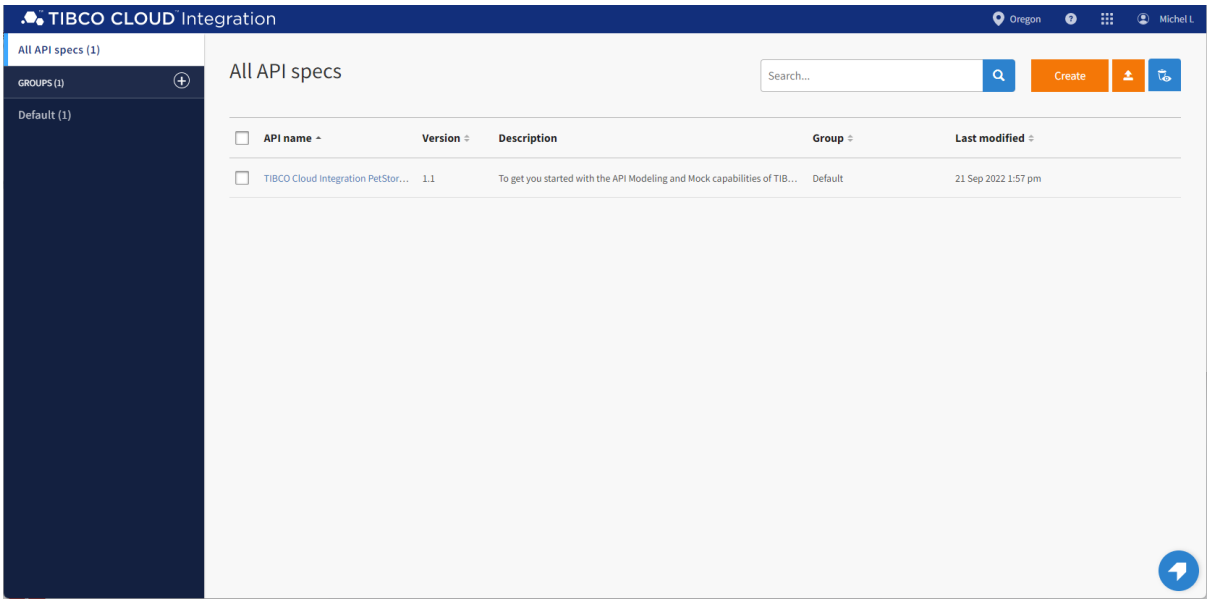
- Design an API
- Create a Mock App
- Import API Specs

## 2.1 Design an API

### 2.1.1 Getting Ready

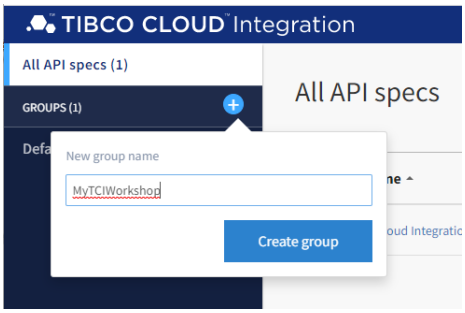
In this lab, you'll create an API using the API Modeler. To navigate from the landing page to the API Modeler, do the following:

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:




## 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**.

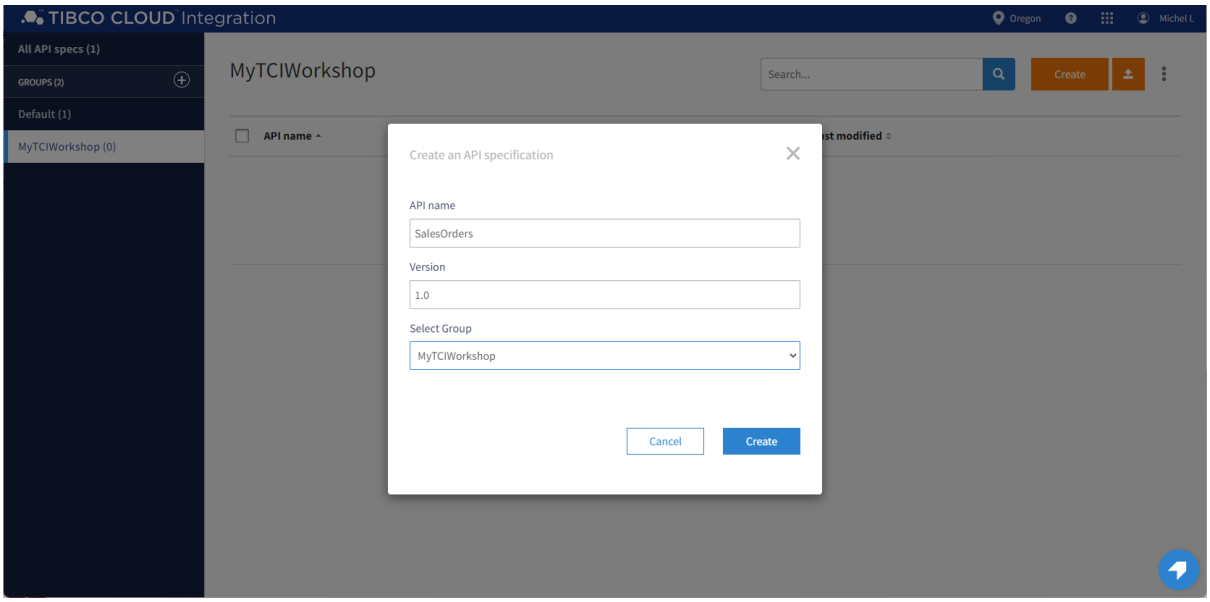



2. Create an API spec by clicking on **MyTCIWorkshop**, and click on the orange **Create** button.

3. In the form, use the following values:

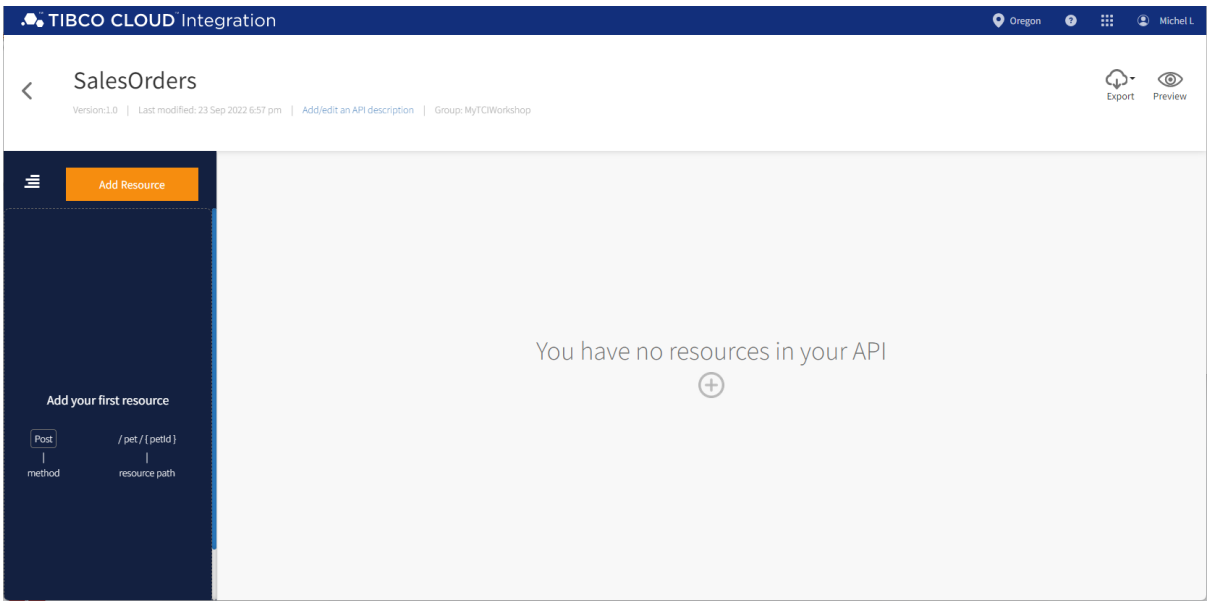
| Field | Value |
|---|---|
| **API name** | SalesOrders |
| **Version** | 1.0 |
| **Select Group** | MyTCIWorkshop |

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.

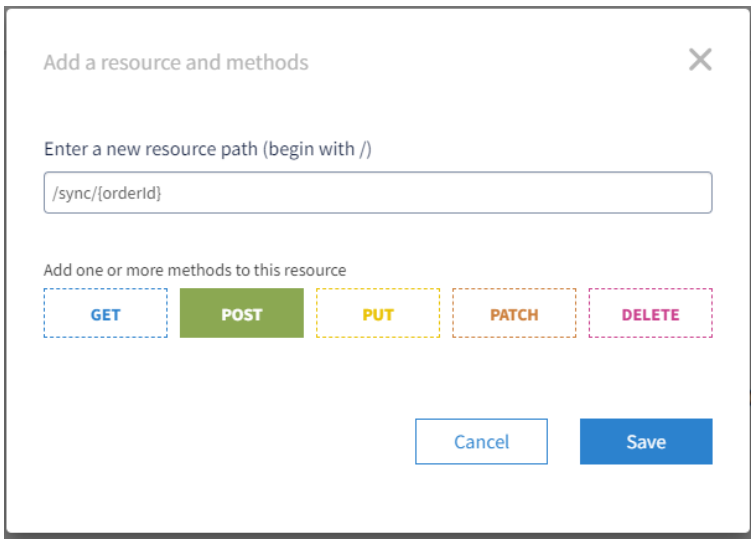



7. In the form, use the following values:

| Field | Value |
|---|---|
| **Resource path** | /sync/{orderId} |
| **Method** | POST |

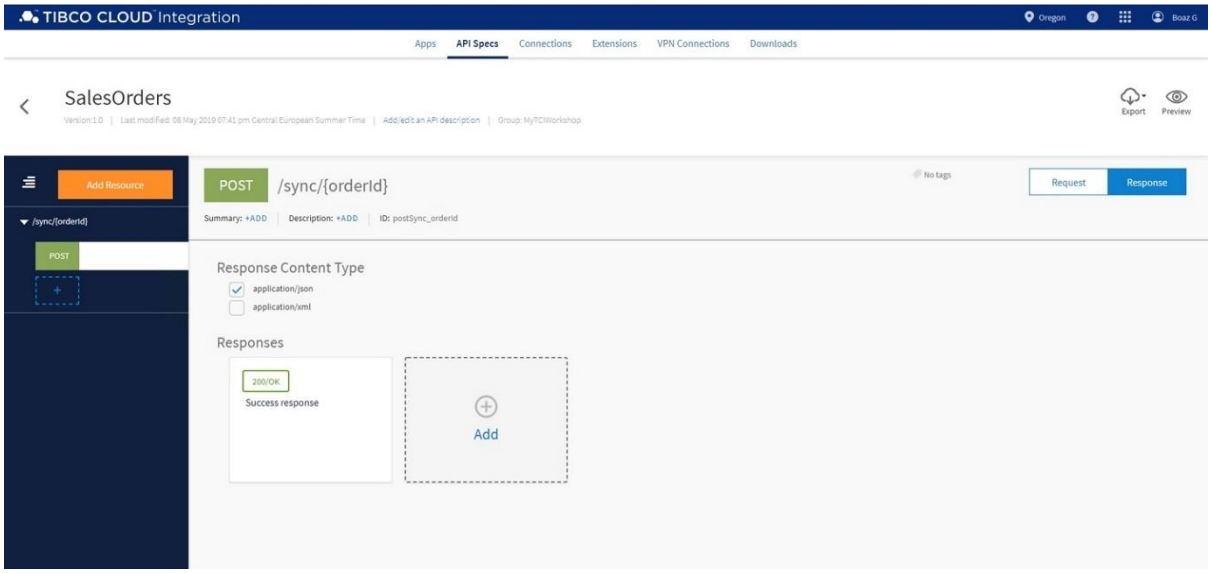
TIBCO CLOUD™

8. The form should look something like this:

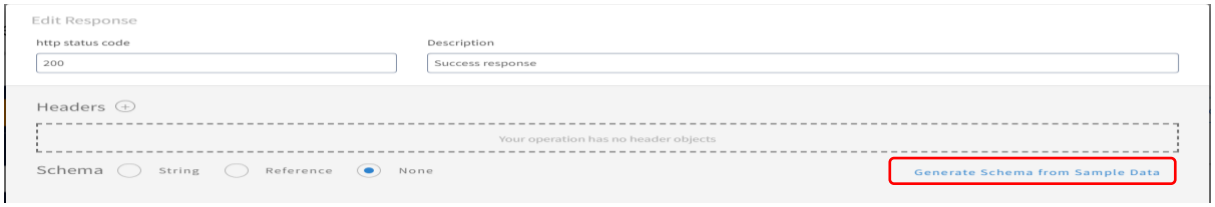



9. Click on the blue **Save** button.

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.




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

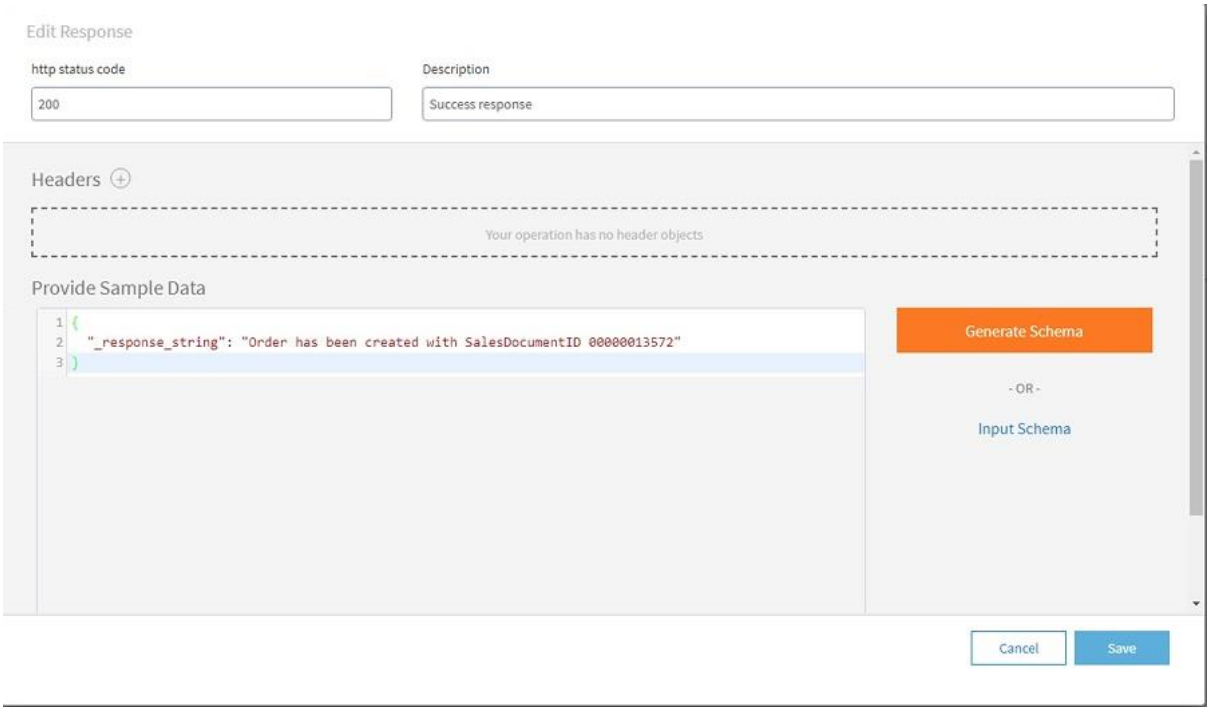


Then on the orange **Continue** button.

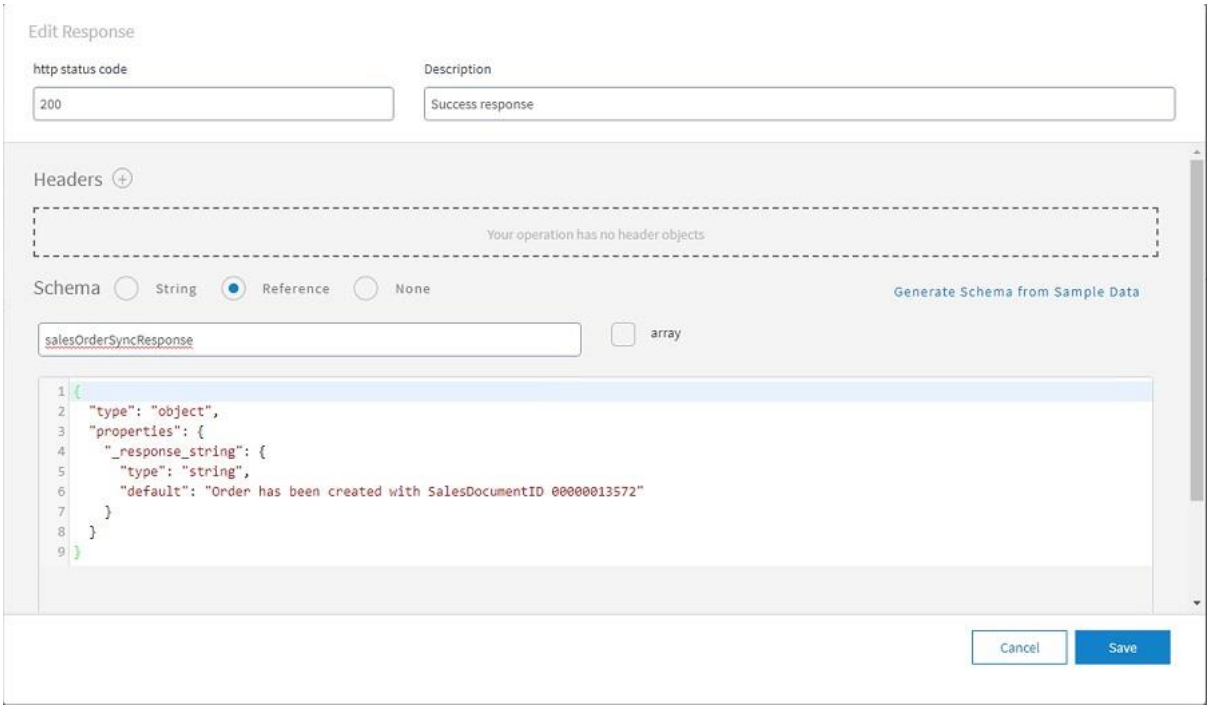
Copy the below sample json in the **Provide Sample Data** field:

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

The form should look something like this:

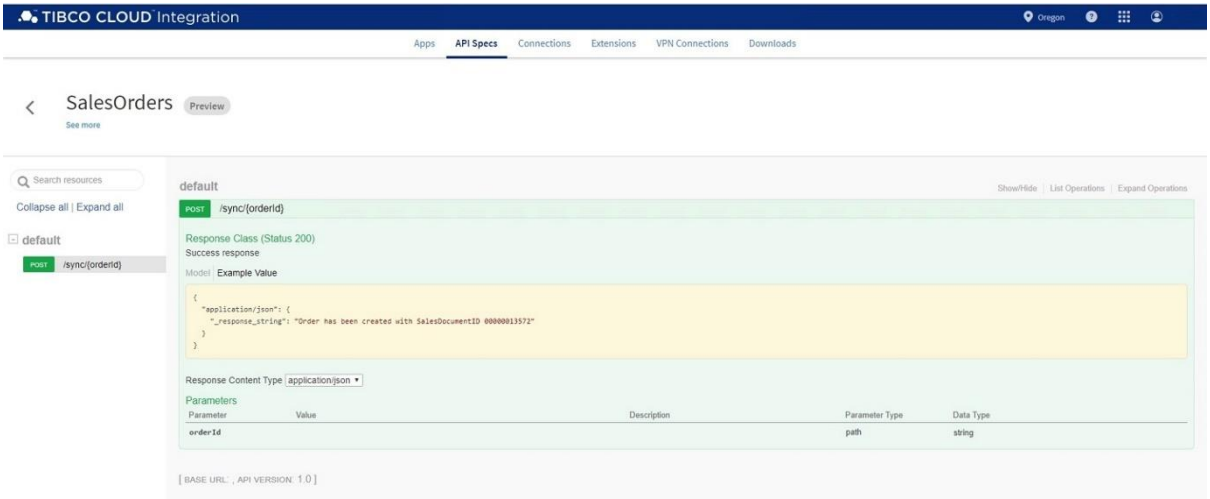



Click on the orange **Generate Schema** button, and replace **GiveNewSchemaNameHere** by **salesOrderSyncResponse** in the next form.
The form should look something like this:




Click on the blue **Save** button.

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




## 2.2 Create a Mock App

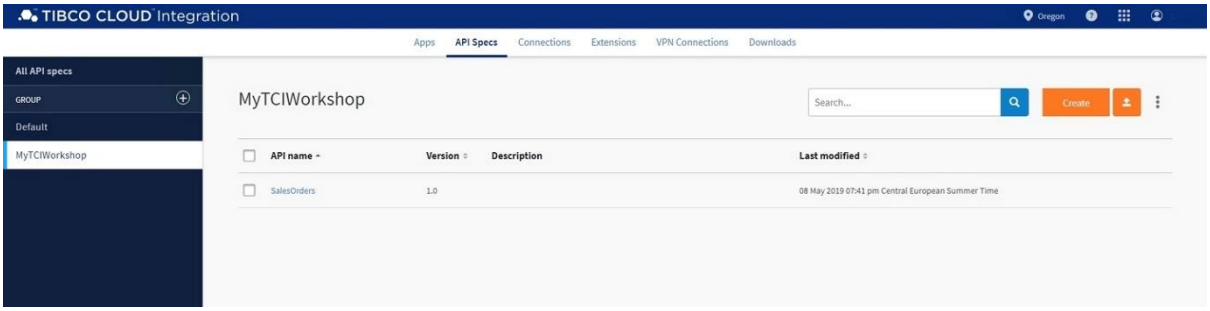
### 2.2.1 Getting Ready

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.

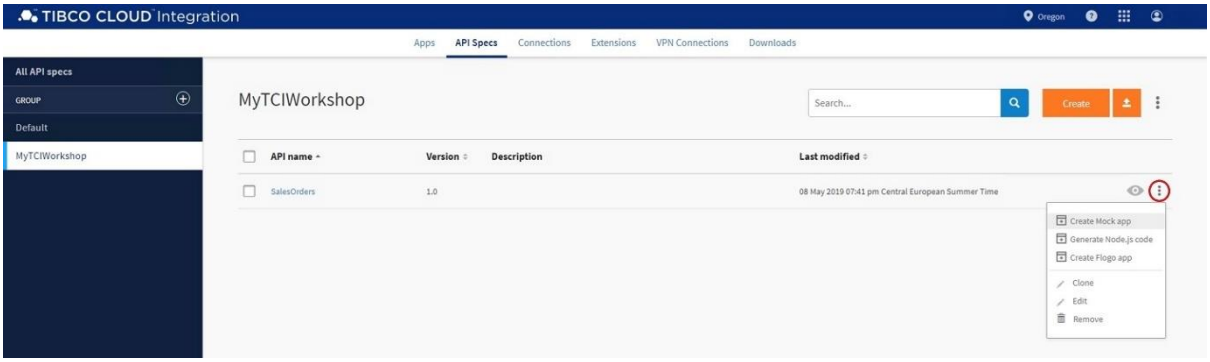
2. Select the group you've created the **SalesOrders** API specification in, e.g. **MyTCIWorkshop**.

3. Your screen should look similar to:

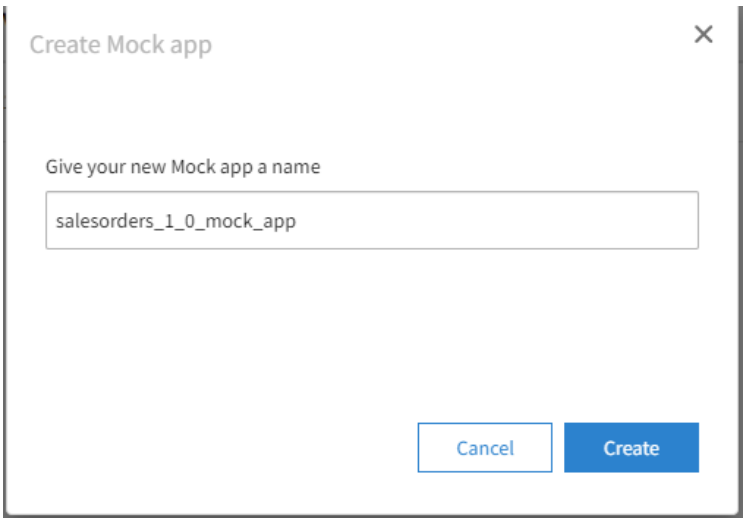



## 2.2.2 How to Do It

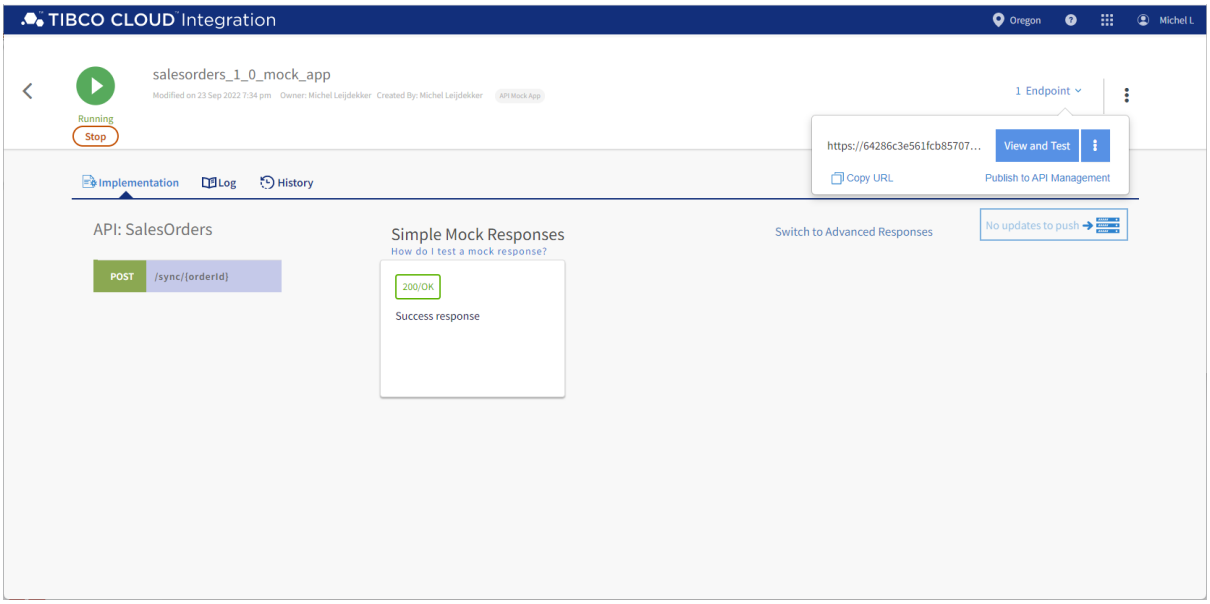
1. Hover over the **SalesOrders** record to the right of the preview icon, and select **Create Mock app** from the menu:



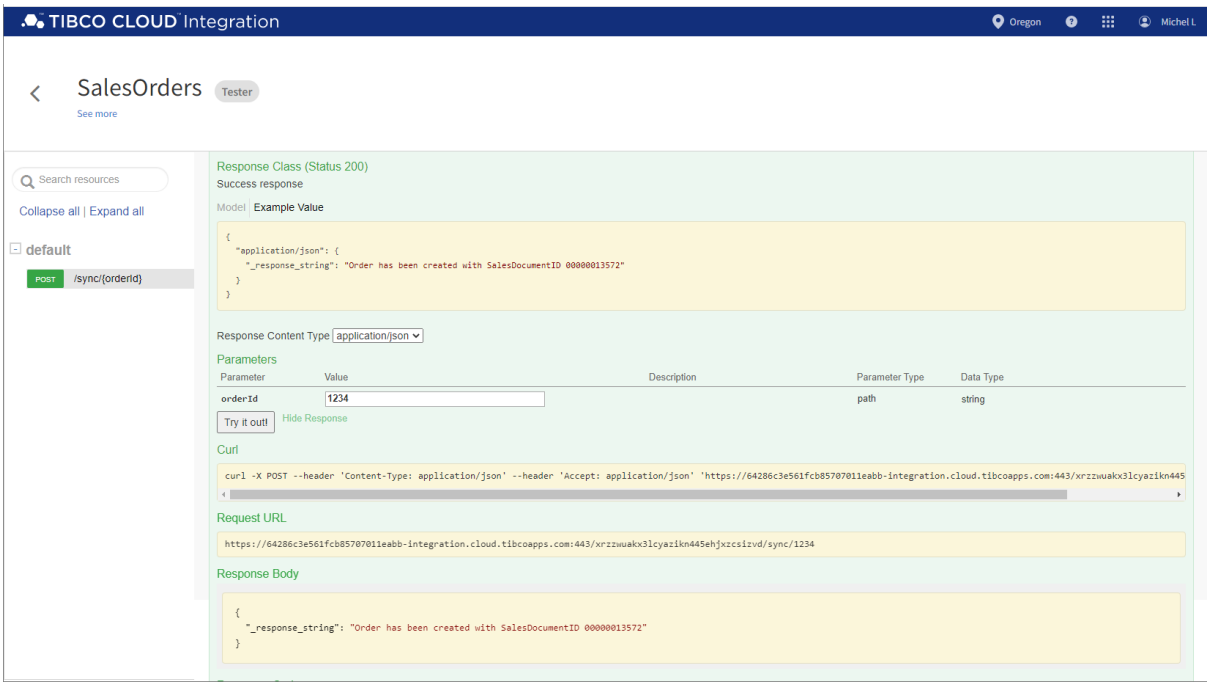

2. Create a mock app by clicking on the **Create** button in next form:



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




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

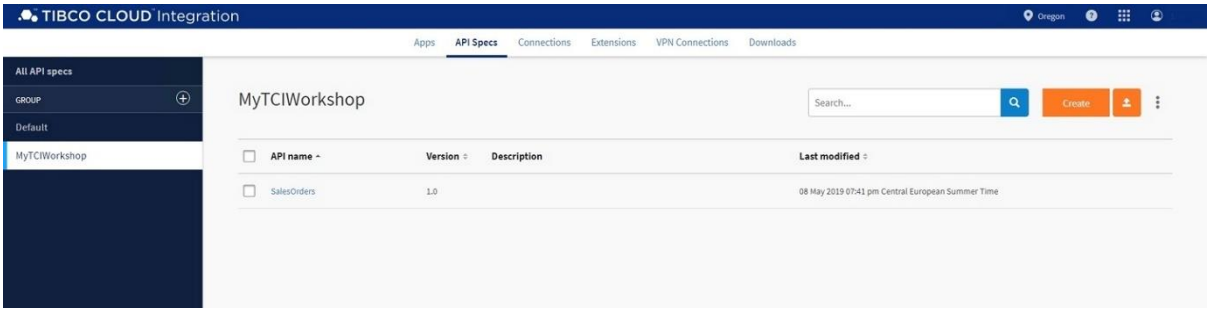



# 2.3 Import API Specs

## 2.3.1 Getting Ready

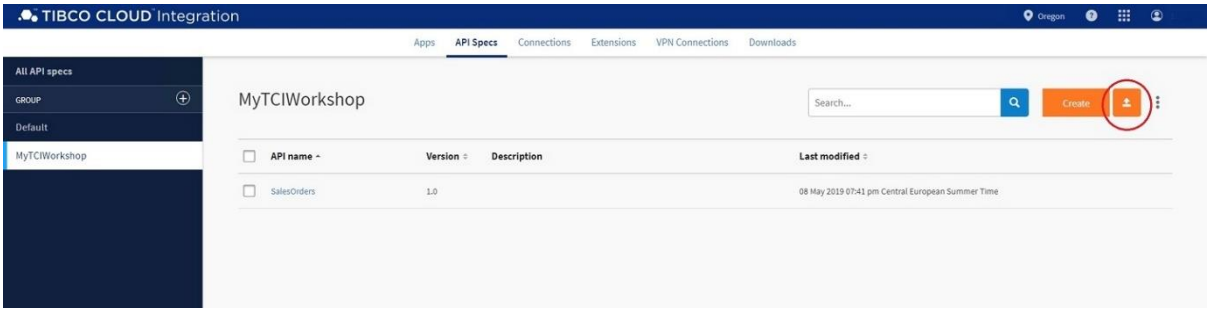
In this lab you'll import additional API specifications. In order to do this:

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:

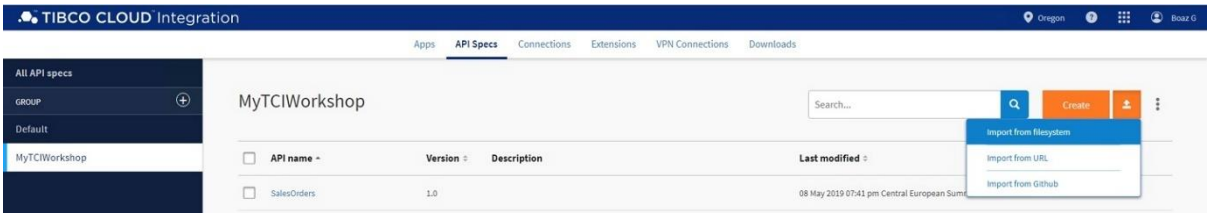



## 2.3.2 How to Do It

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



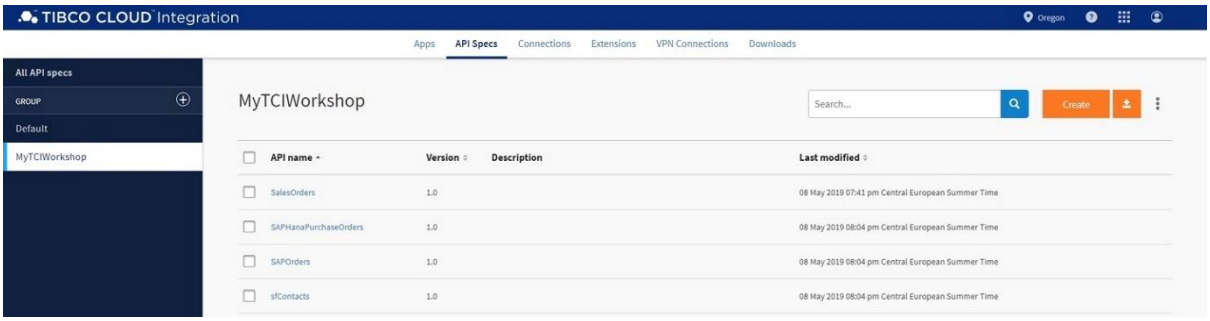

2. Select **Import from filesystem**,




and select the following files from the **API Specs** directory:

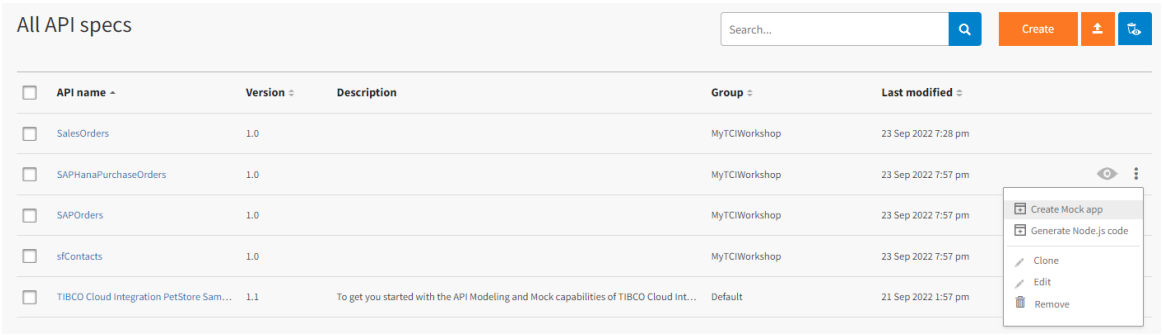
- **SAPHanaPurchaseOrders.json**
- **SAPOrders.json**
- **sfContacts.json**

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

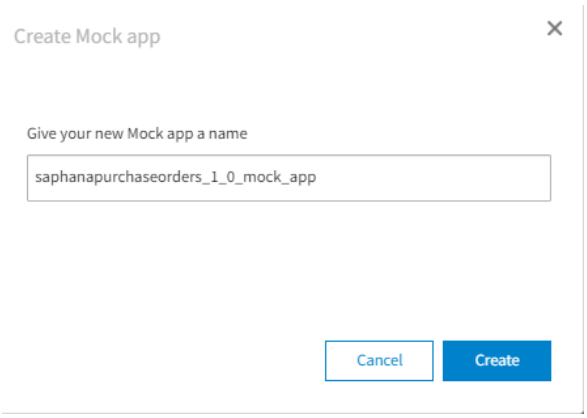



4. For the SAPHanaPurchaseOrders we also create a Mock application as we did in 2.2.2.

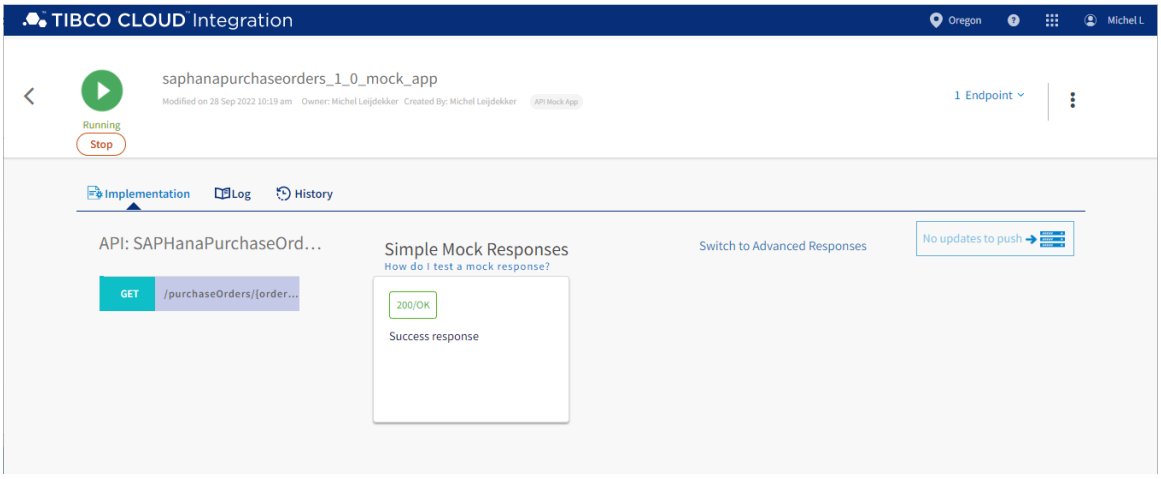
   - Hover over the **SAPHanaPurchaseOrders** record to the right of the preview icon, and select **Create Mock app** from the menu:




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




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

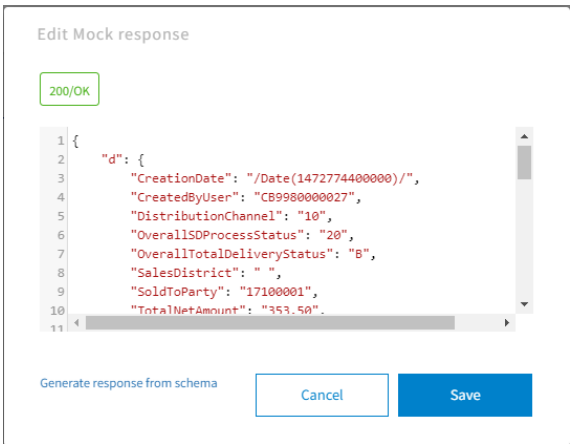



In the Edit Mock reponse wizard, copy and paste the following .json snippet

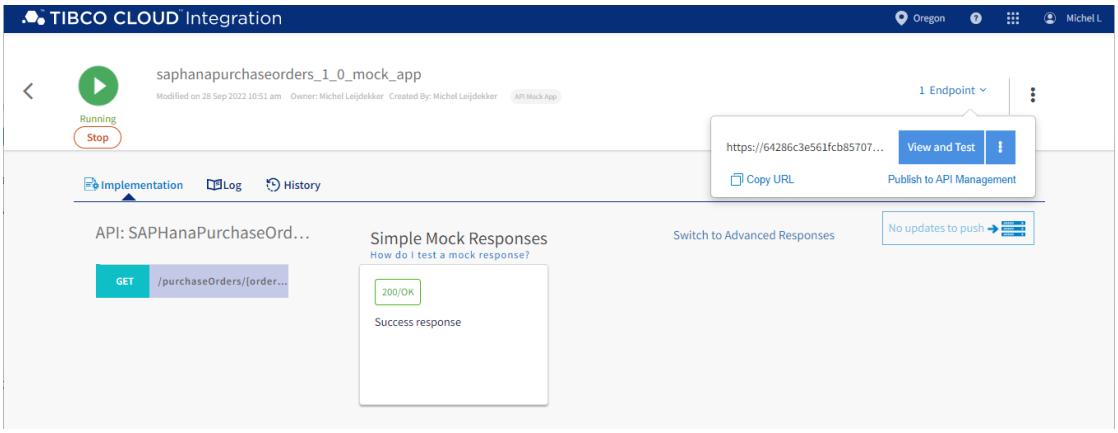
```
{
    "d": {
        "CreationDate": "/Date(1472774400000)/",
        "CreatedByUser": "CB9980000027",
        "DistributionChannel": "10",
        "OverallSDProcessStatus": "20",
        "OverallTotalDeliveryStatus": "B",
        "SalesDistrict": " ",
        "SoldToParty": "17100001",
        "TotalNetAmount": "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)/"
    }
}
```

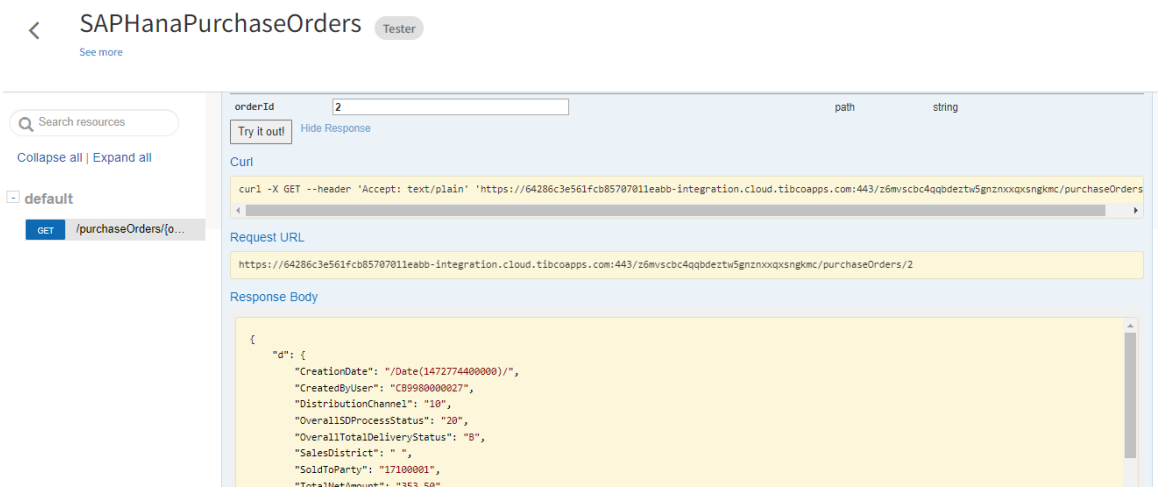

- Click **"Save"**



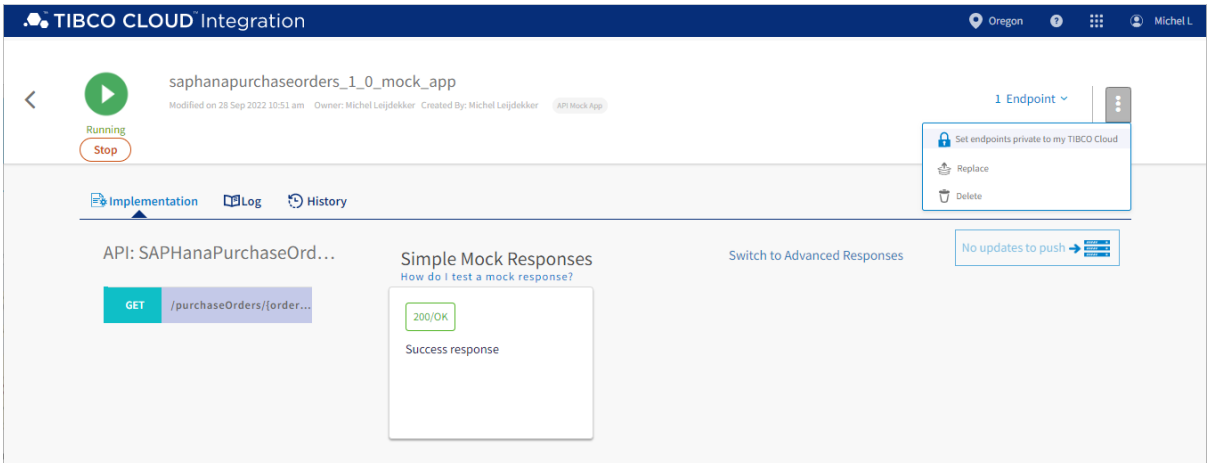

- After the save action click **Update Mock app** in the top right corner.
- Once the mock app is running again, hover over the **Endpoint** link, and select **View and Test** from the menu:




- 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

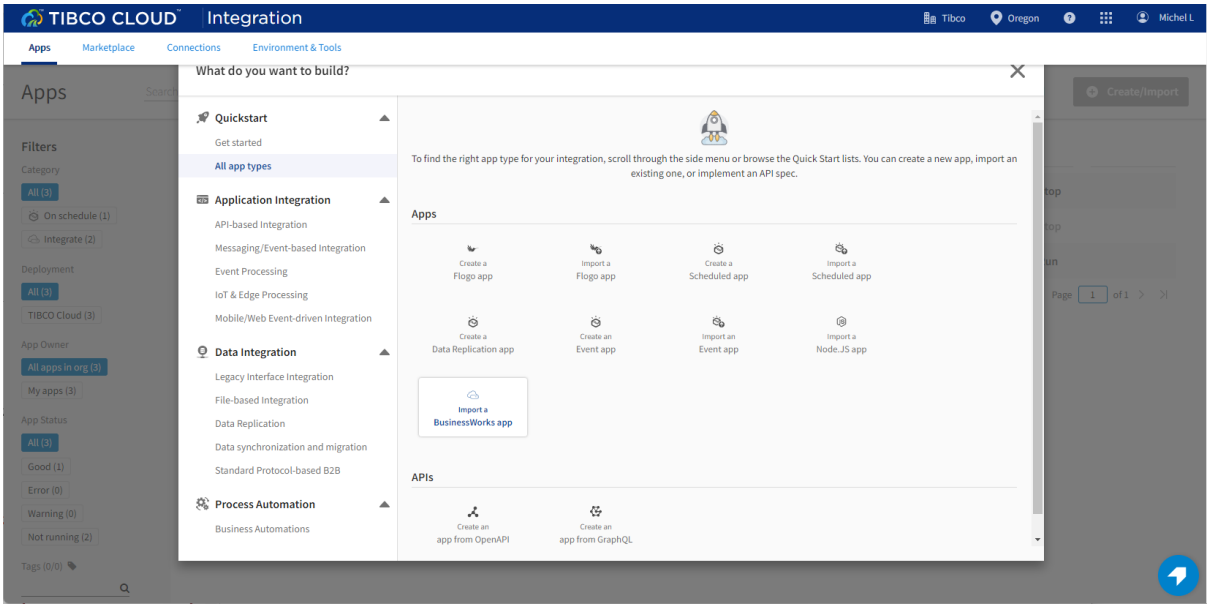
## 3.1 BusinessWorks Apps in TIBCO Cloud Integration

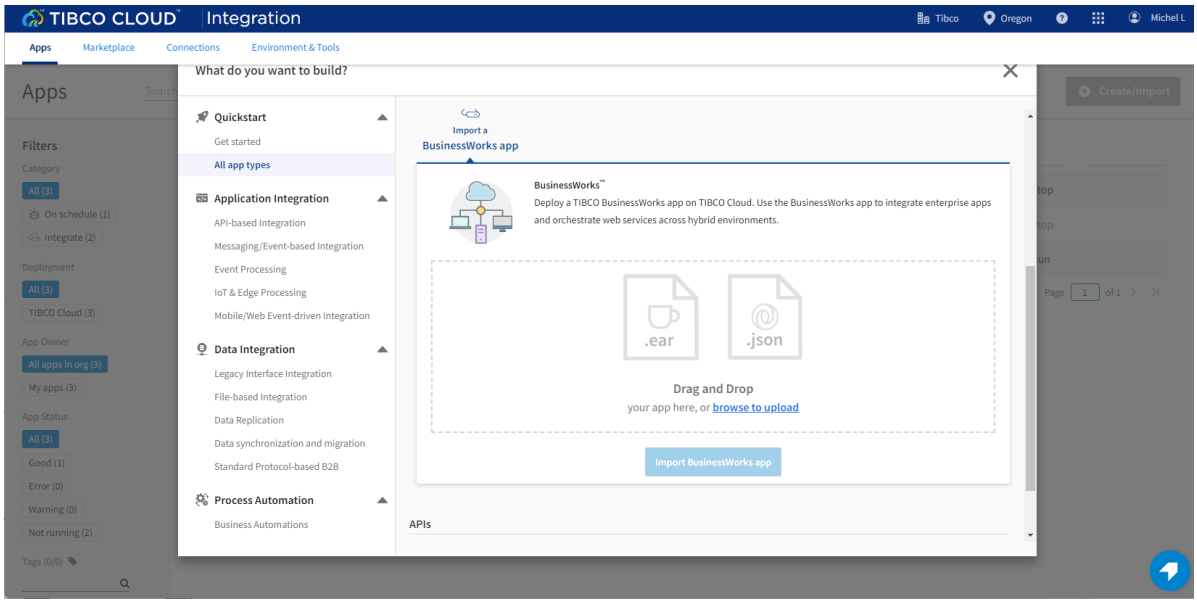
### 3.1.1 How to Do it: Import the Business Works Application via EAR import

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

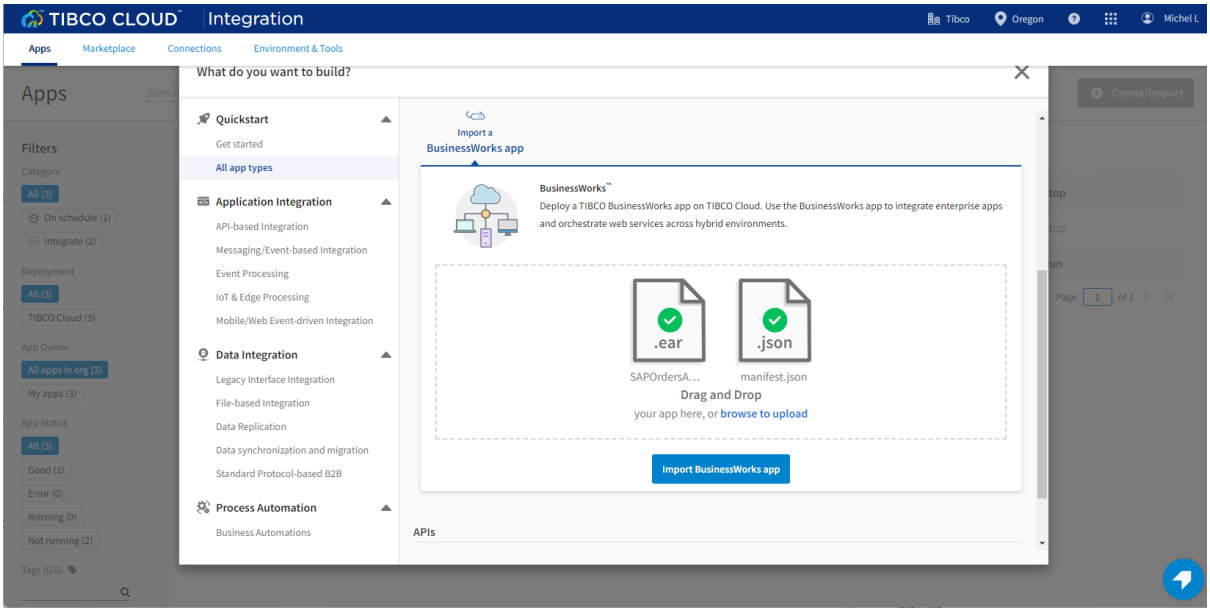

3. Then select the "**All app types**" menu option from the wizard and select the "**Import a BusinessWorks app**"



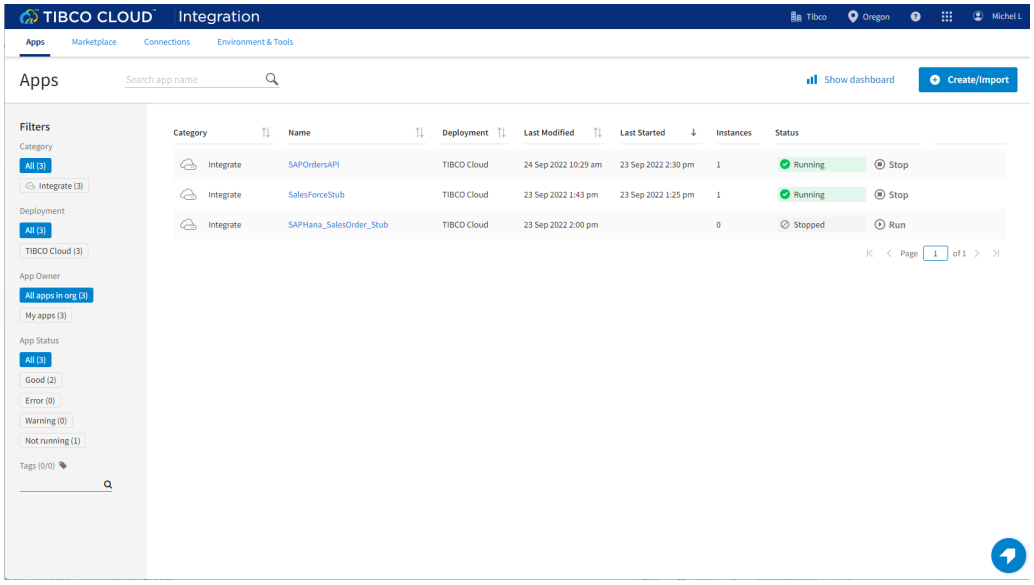



4. Select the following EAR- and manifest files.

| Service | Files |
|---|---|
| Contact Service | SalesForceAPI_1.0.0.ear + manifest.json |
| Orders Service | SAPOrdersAPI_1.0.0.ear + manifest.json |
| Sales Order Service | SAPHanaAPI_1.0.0.ear + manifest.json |




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




TIBCO CLOUD™

**Important Note:** The trial subscription only allows to run two apps at the same time. Therefor we will use API Mocks to simulate 2 of the 3 required Business Works solutions.

## 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.

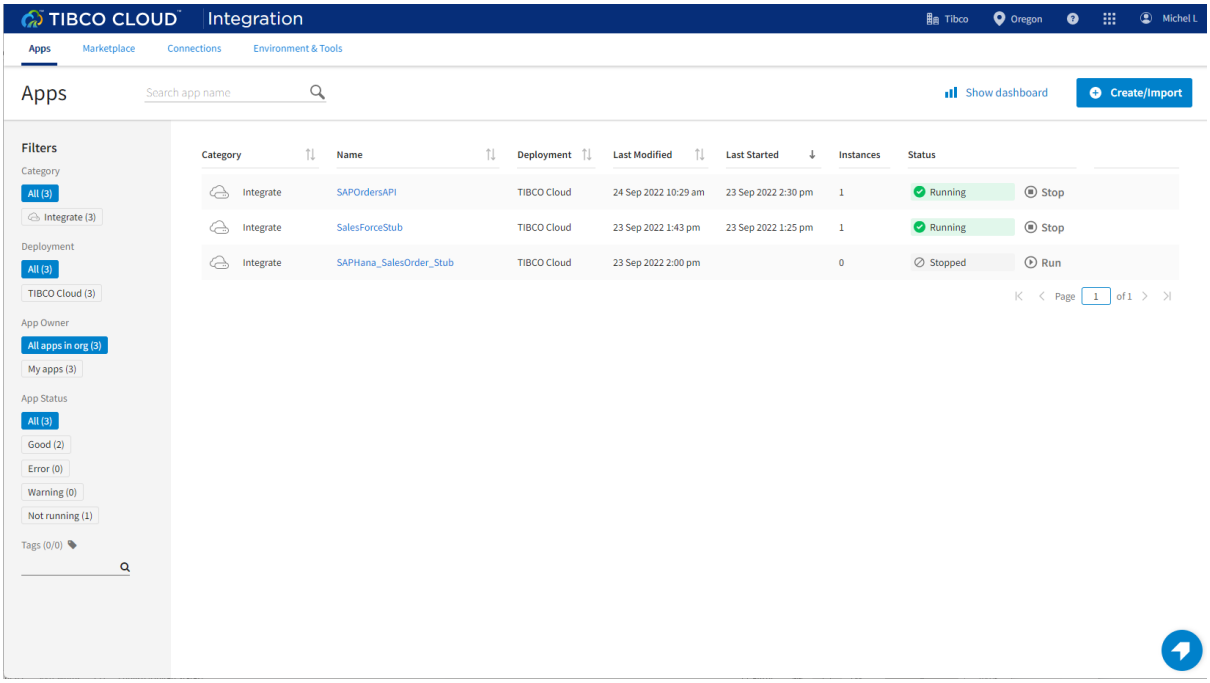
You will first create a "skeleton" Flogo app from the **SalesOrders** API specification. Then you will implement the flow with the 3 system API calls, to finally push the Flogo app and test it.

### 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:

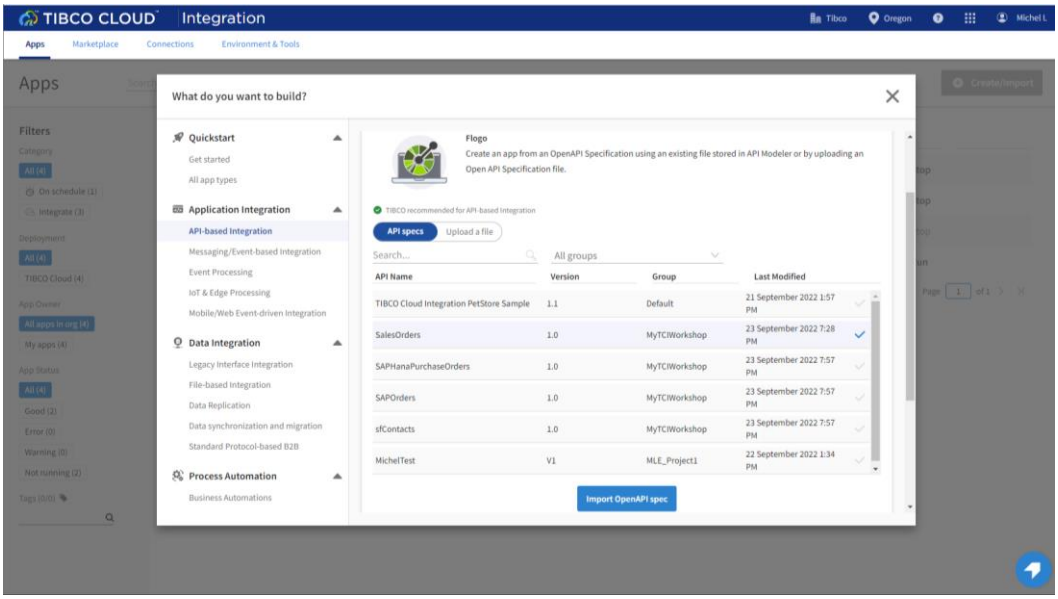



### 3.2.2 How to Do It: Create the Skeleton Flogo App from the API Specification

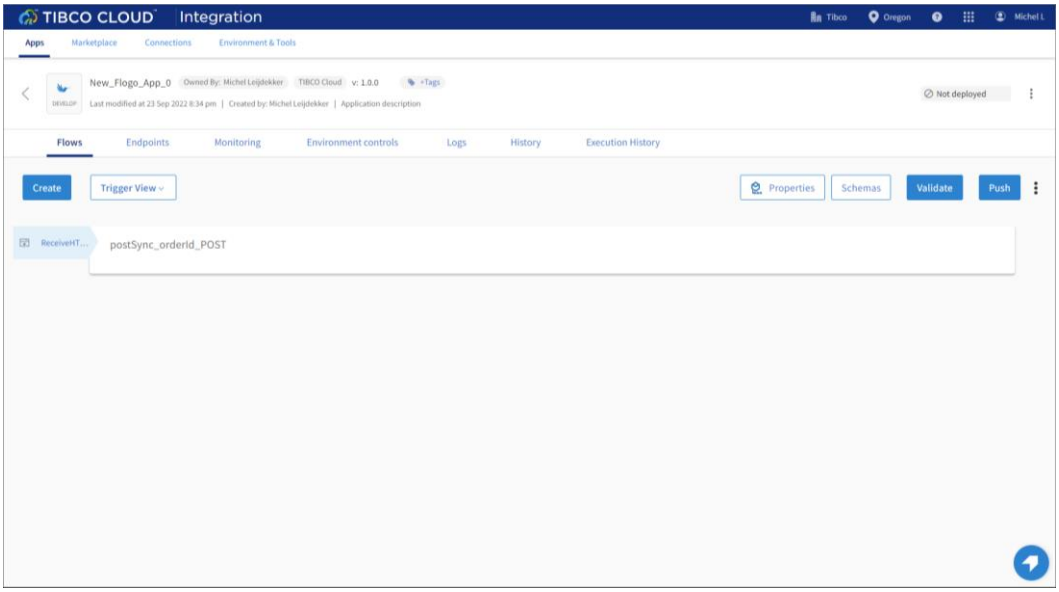
To create a "skeleton" Flogo app from on the **SalesOrders** API specification, do the following:

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

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




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



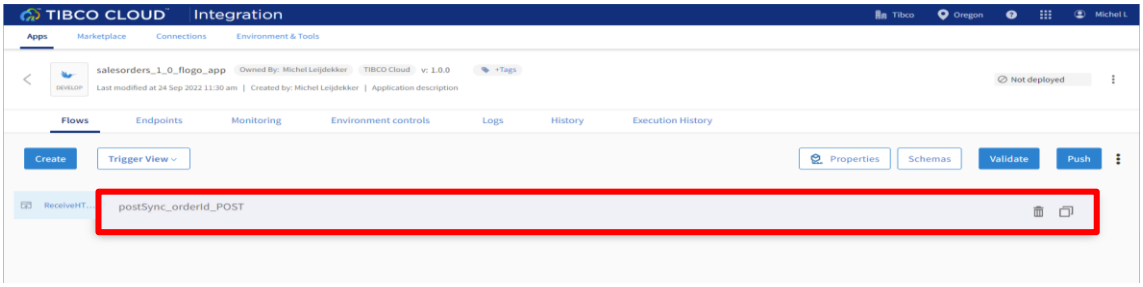

4. Click on the "New_Flogo_App_0" name of the integration and rename it to "**salesorders_1_0_flogo_app**".

## 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 adresses 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:

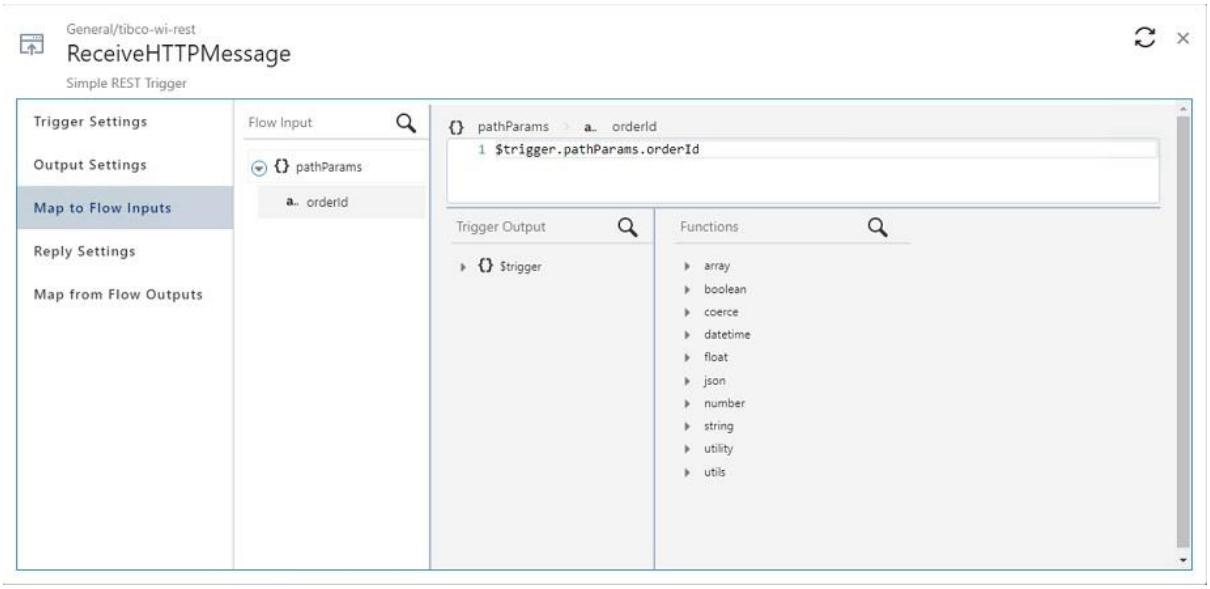
1. Navigate from the app **salesorders_1_0_flogo_app** to the skeleton flow **postSync_orderId_POST**.



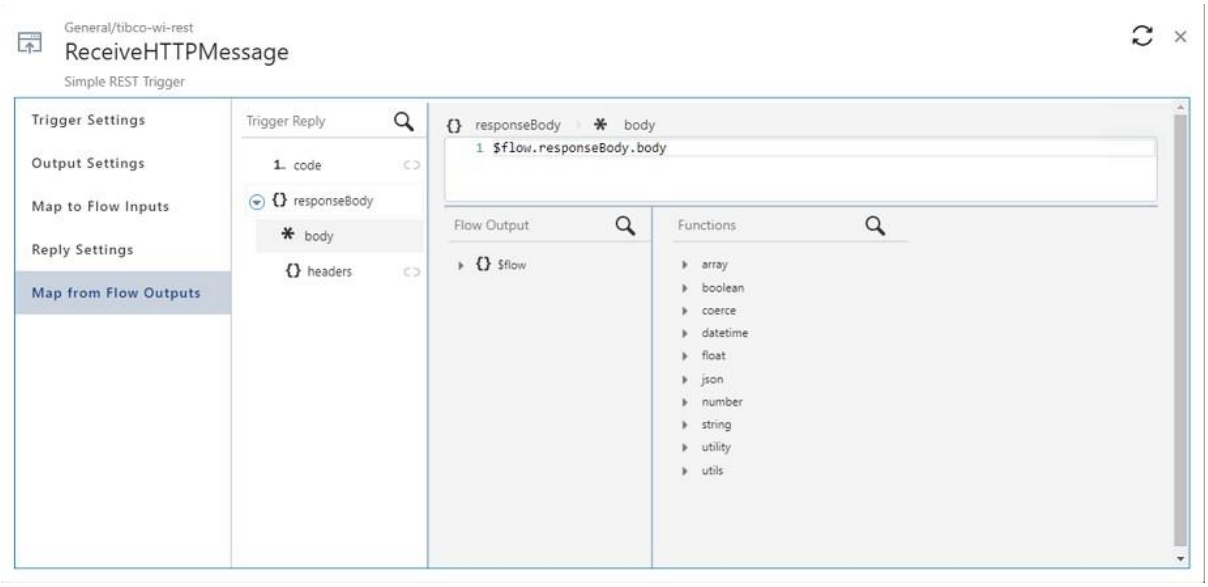

2. To enable data to flow between the trigger and the flow, you first need to map the trigger output to the flow input, and flow data to the trigger reply.

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

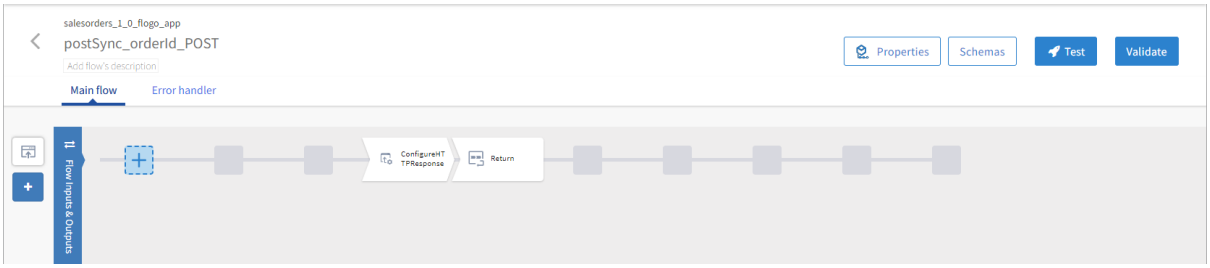
3. Check the auto mapped mappings from the trigger output to the flow input as follows:



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

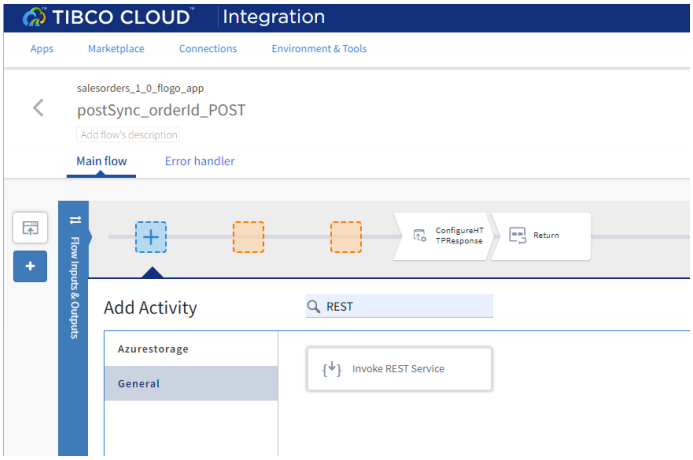



5. Close the trigger configuration screen, and move the **ConfigureHTTPResponse** and **Return** tiles at least 3 positions to the right.




### Creating the Activities

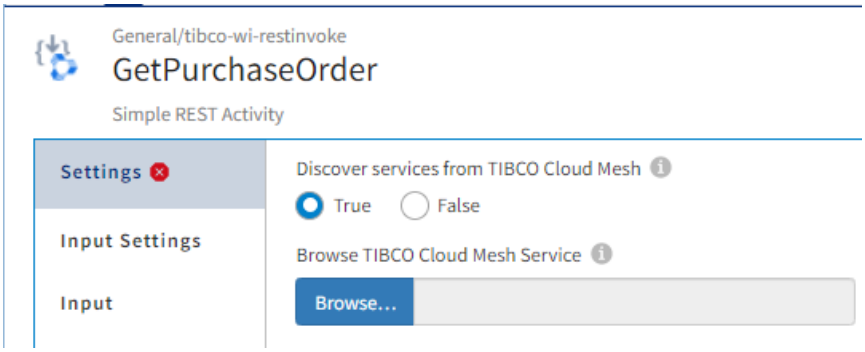
6. Create a first activity, by clicking on the left-most + and navigate through **General > Invoke REST Service**.



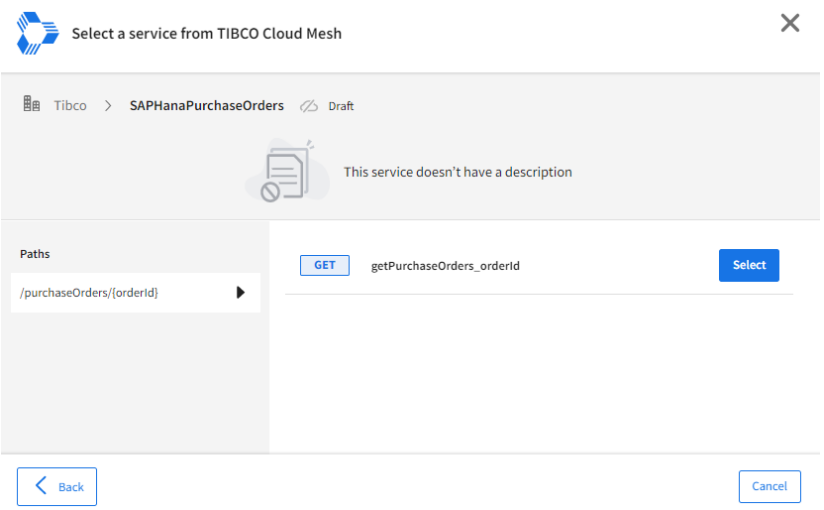

7. Configure this activity as follows:

   - 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..**

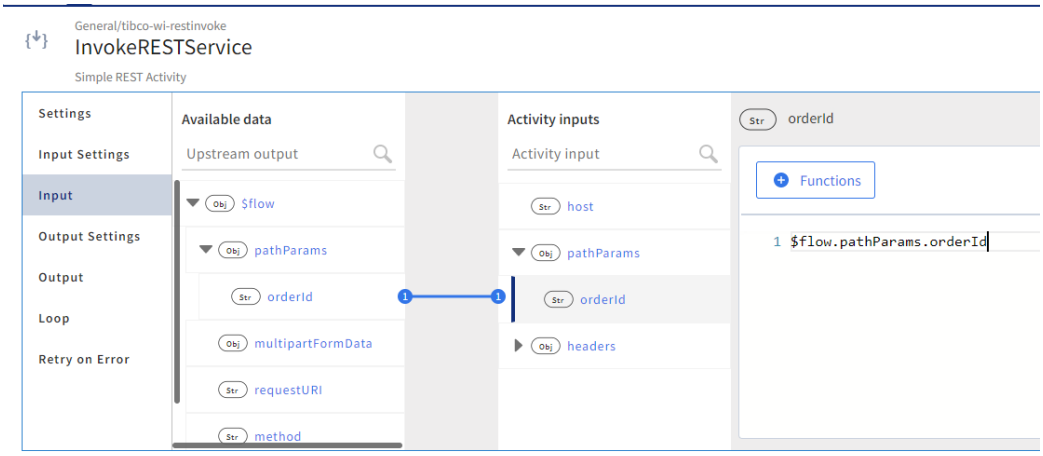



   - Click on the **SAPHanaPurchaseOrders** API Definition from the list. In the next wizard screen **select** the operation.

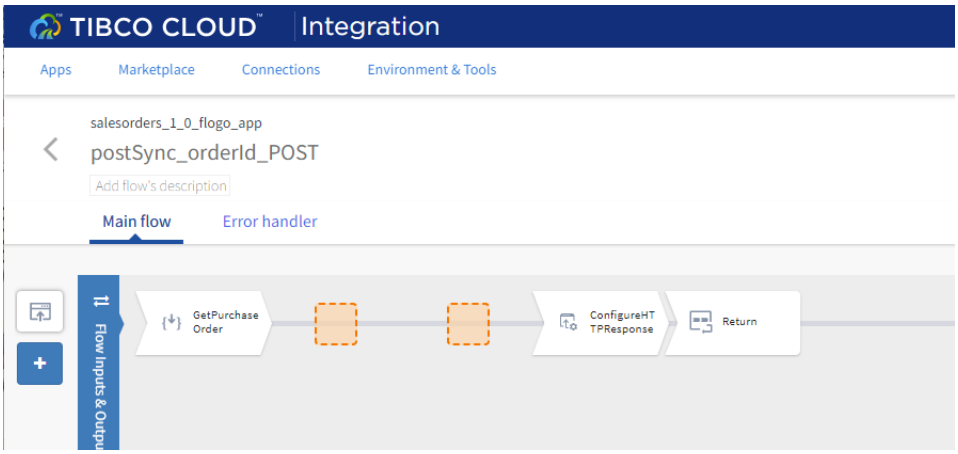



- In the **Input** section, create the following mapping:

| Activity Input | Upstream Output |
|---|---|
| `pathParams.orderId` | `$flow.pathParams.orderId` |



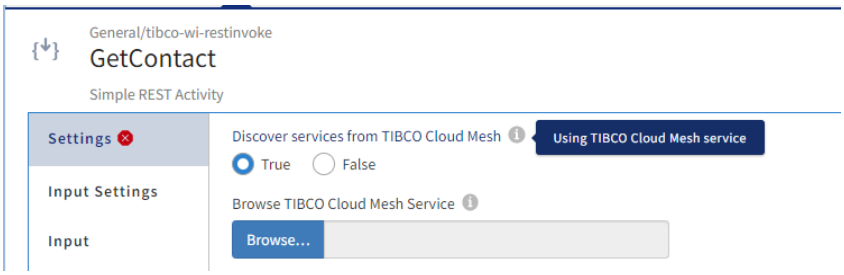

8. Close the activity configuration screen, the integration should now look like the picture below.



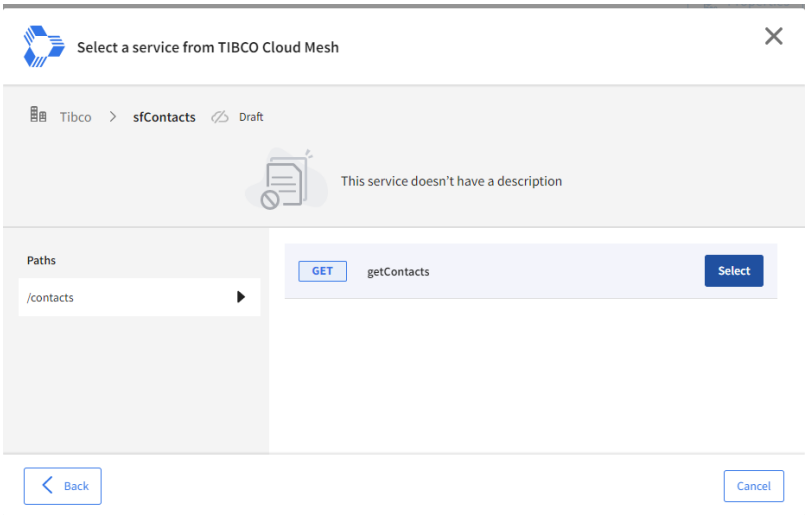

9. Now create a second activity, by clicking on the next + and navigate through **General > Invoke REST Service**. Configure this activity as follows:

   - Give it a name: **GetContact**.

   - 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.

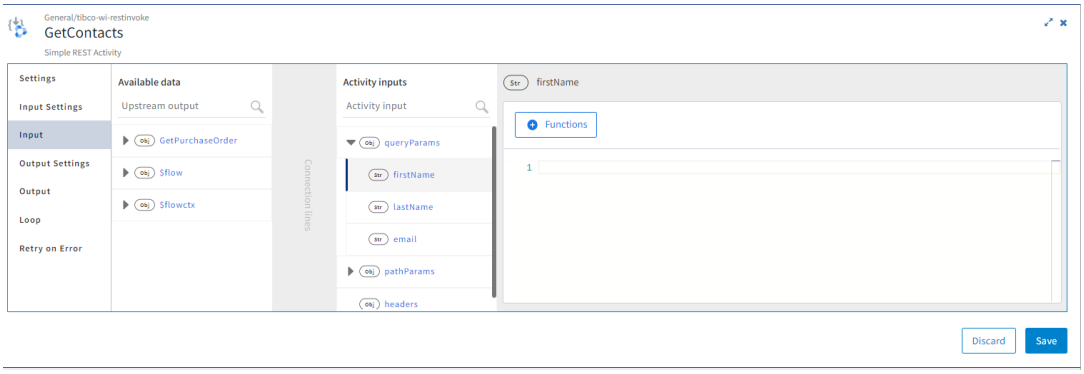


- In the **Input Settings** section, check the imported query parameters.
- In the **Input** section, create the mapping.

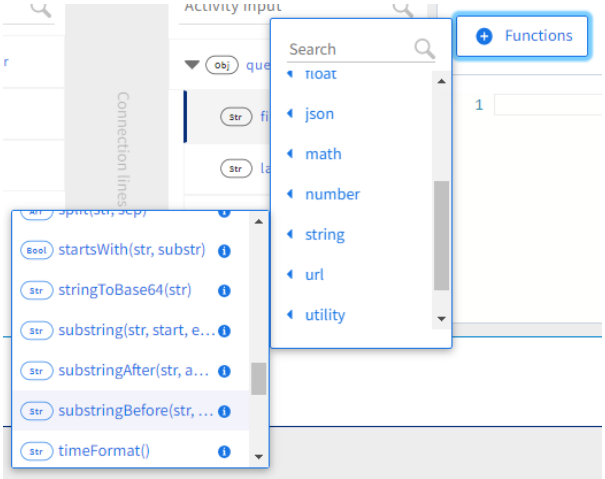
| Activity Input | Upstream Output |
|---|---|
| **queryParams.firstName** | `string.substringBefore($activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, "::")` |
| **queryParams.lastName** | `string.substringBefore(string.substringAfter($activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer,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

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

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

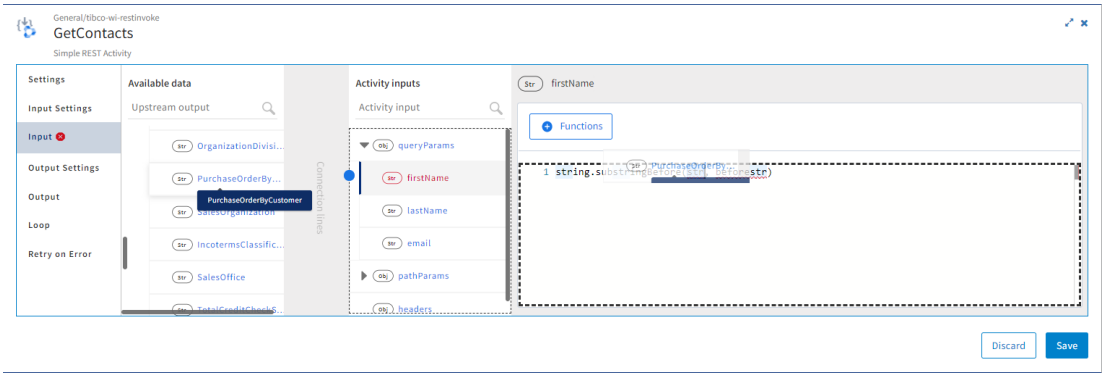



2. In the Functions dropdown Search in the string functions for the **substringBefore** function and select this to drop it to the mapping area.

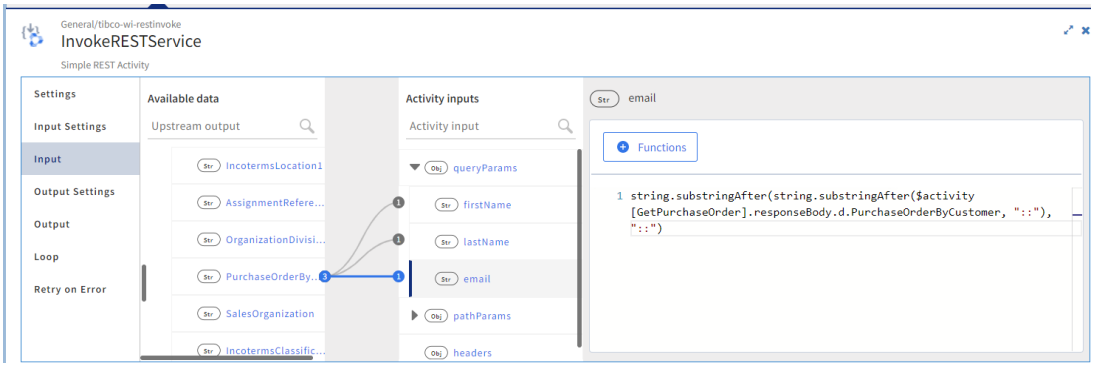



TIBCO CLOUD™

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



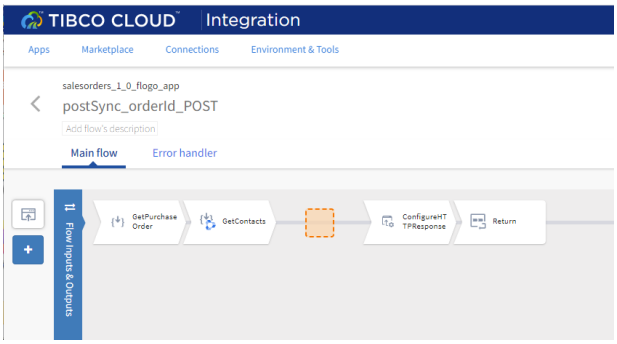

4. Set the second string ***beforestr*** to **"::",** by typing into the mapping.




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

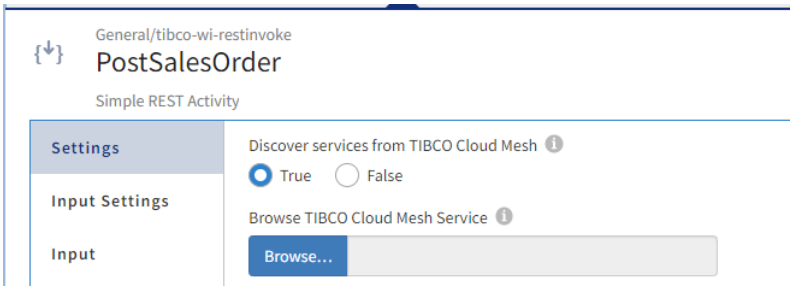
| Activity Input | Upstream Output |
|---|---|
| **queryParams .lastName** | `string.substringBefore(string.substringAfter($activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, "::"), "::")` |
| **queryParams .email** | `string.substringAfter(string.substringAfter($activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer,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

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

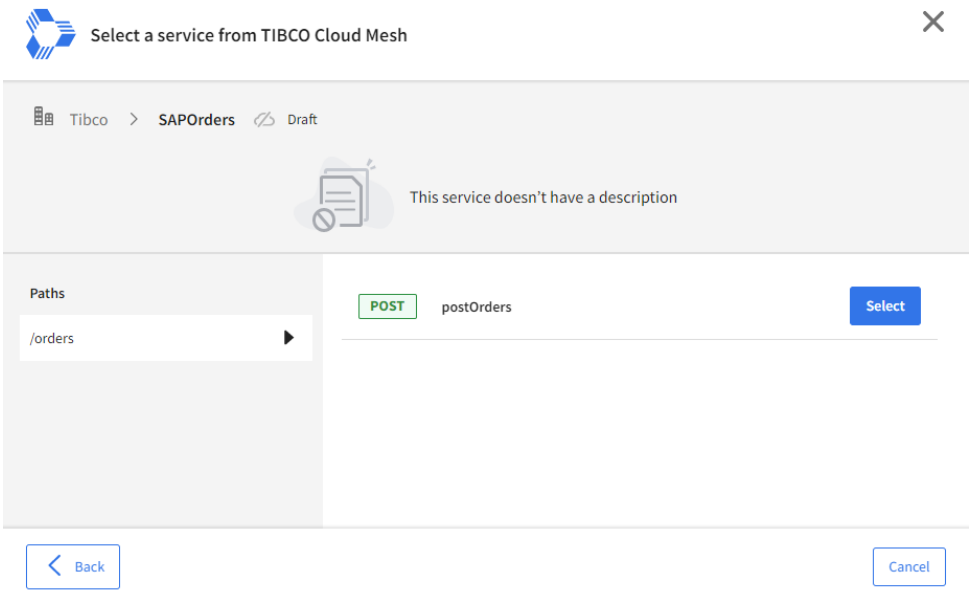



11. Now we create the third activity, which corresponds to the SAP Order Creation Service. Click on the next + and navigate through **General > Invoke REST Service**. Configure this activity as follows:

    - 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..**



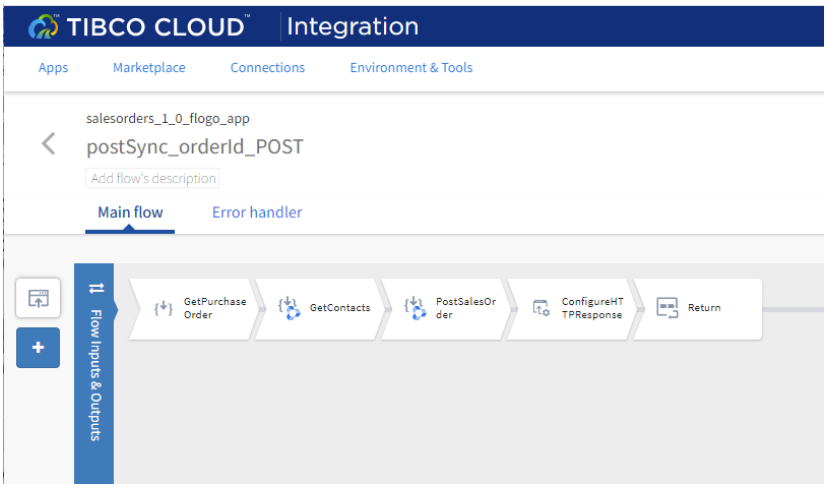

    - Now select the SAPOrders from the list and select the postOrders Operation.




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

| Activity Input | Upstream Output |
|---|---|
| **body.Email** | `$activity[GetContacts].responseCodes["200"].email` |
| **body.firstName** | `$activity[GetContacts].responseCodes["200"].firstName` |
| **body.item** | `"1"` |
| **body.lastName** | `$activity[GetContacts].responseCodes["200"].lastName` |
| **body.phone** | `$activity[GetContacts].responseCodes["200"].mobile` |
| **body.price** | `$activity[GetPurchaseOrder].responseCodes["200"].d.TotalNetAmount` |

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



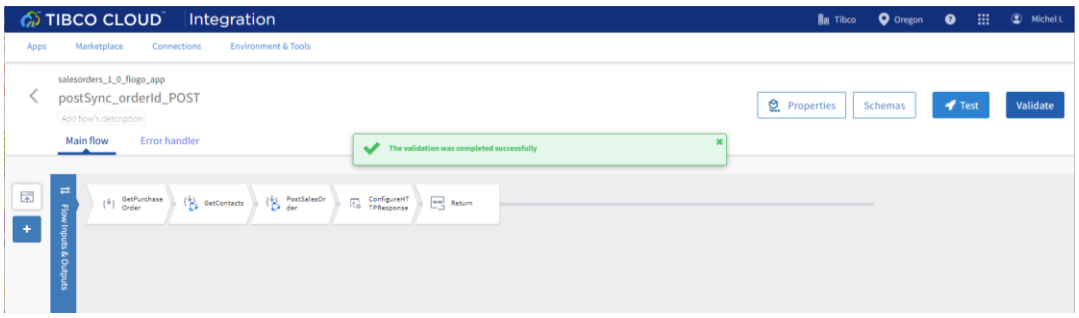
13. Now configure the **ConfigureHTTPResponse** activity as follows:

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

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

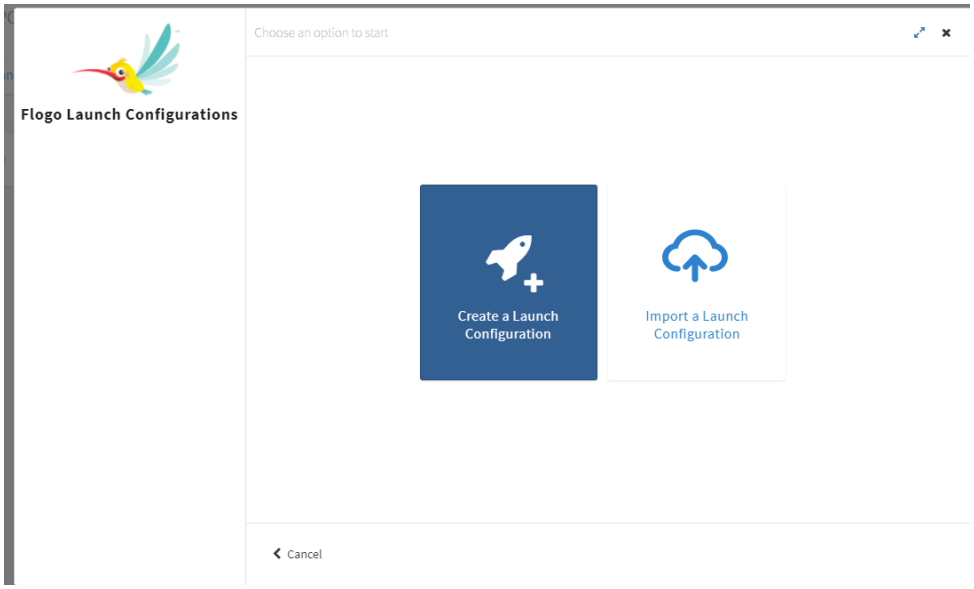
14. Close the activity configuration screen.

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

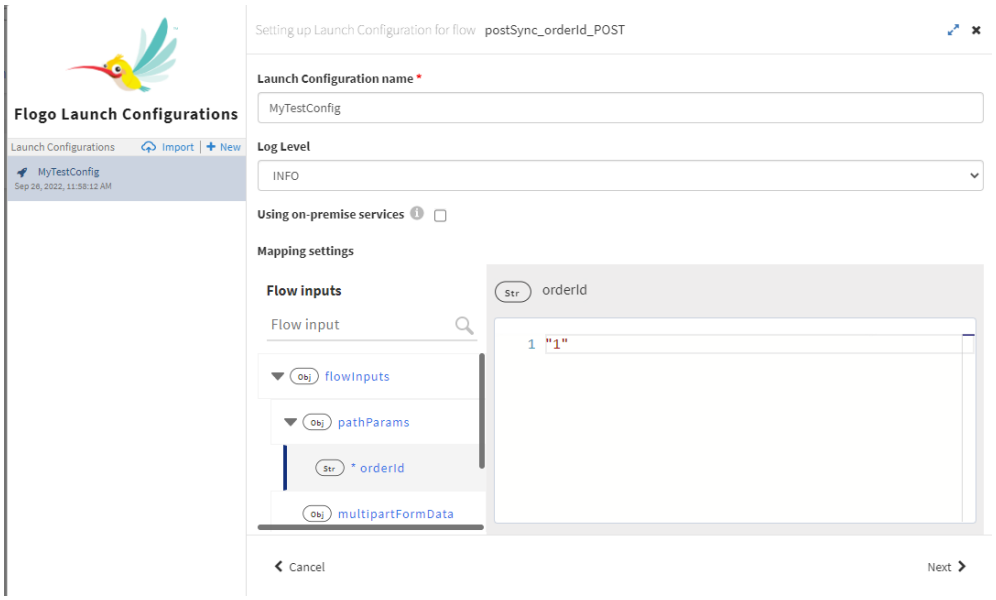



## 3.2.4 Push the Integration App to TIBCO Cloud and Test It

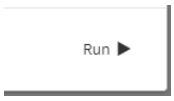
1. Click on the Blue **Test** Button to start the Test configuration wizard. First you need to create a Launch Configuration for this test.



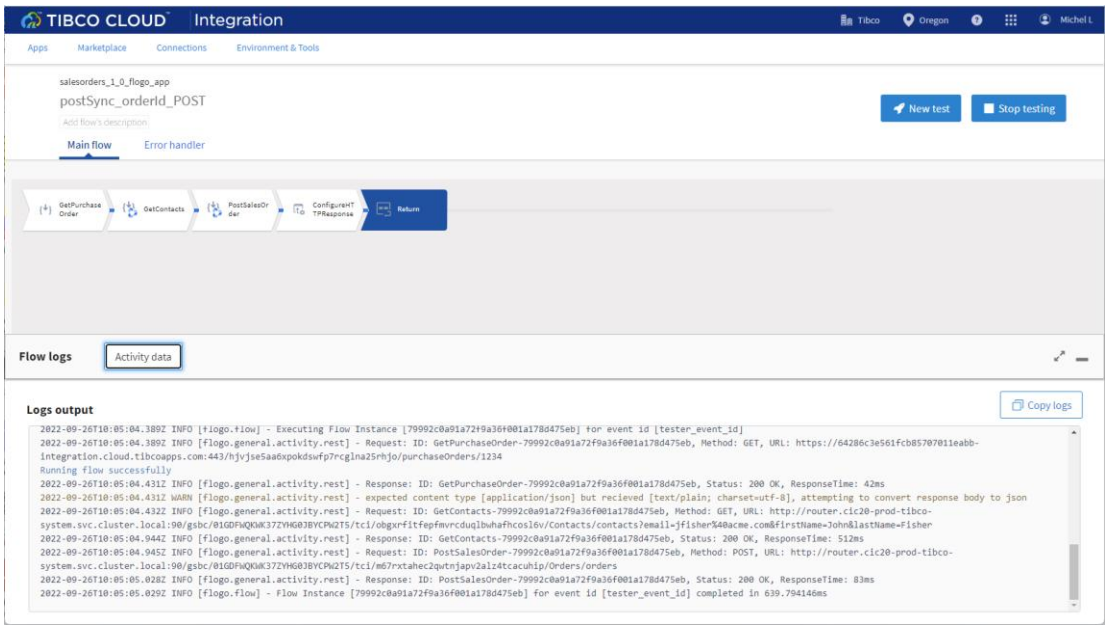
2. Click on the corresponding tile and give your configuration a name. Create a mapping for the integration input parameter **orderId** as shown below.



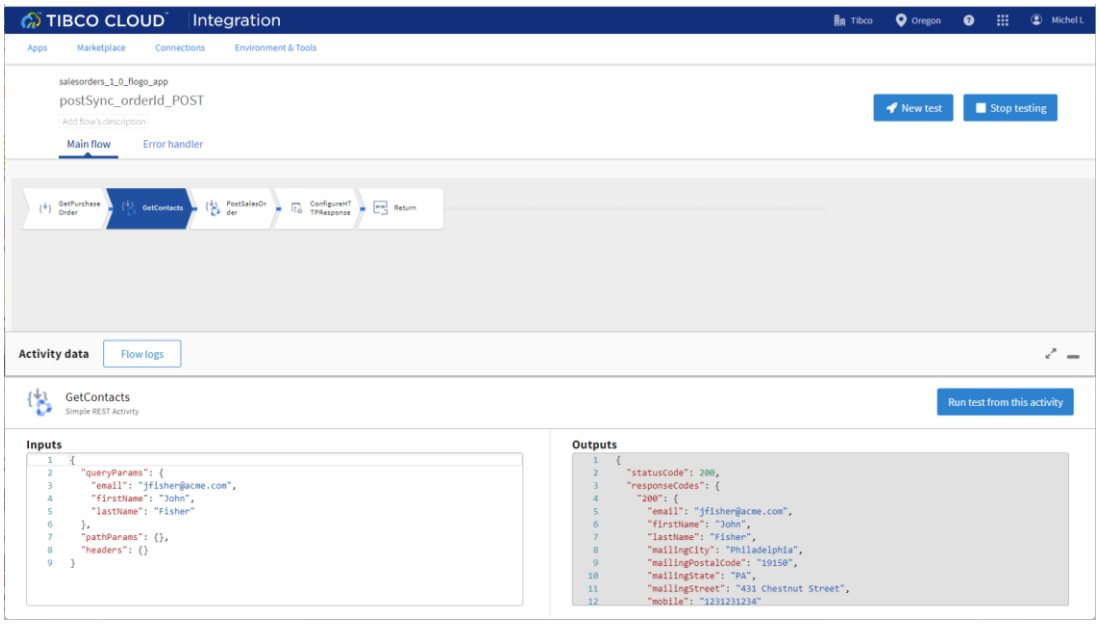

3. Click the **Next** button in the right lower corner and check the summary. Now click the Run button in the bottom right corner to start the test run.




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

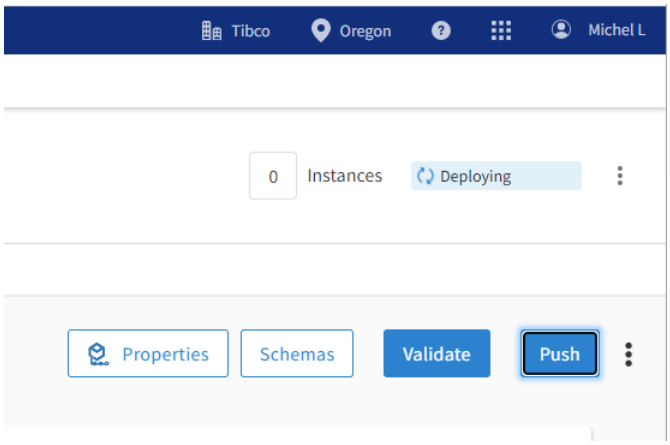


5. You can introspect individual service and API calls on input and output result, by just clicking on the flow diagram and select the activity you want to inspect.

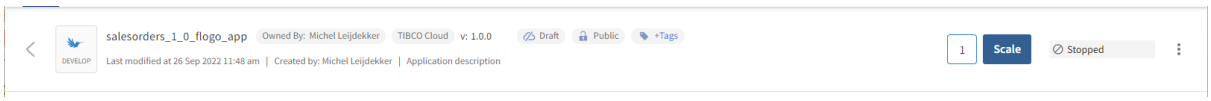



To push the Integration app to the runtime environment and test it, do the following:

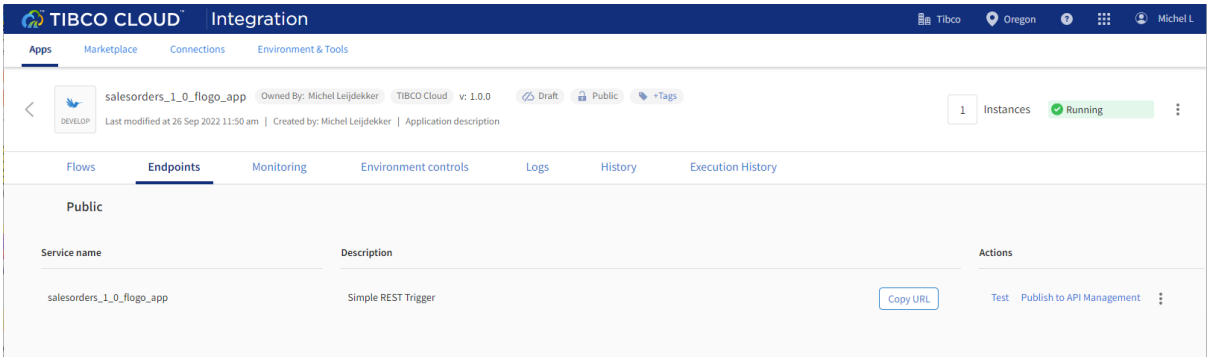
1. Click the **Push app** button.



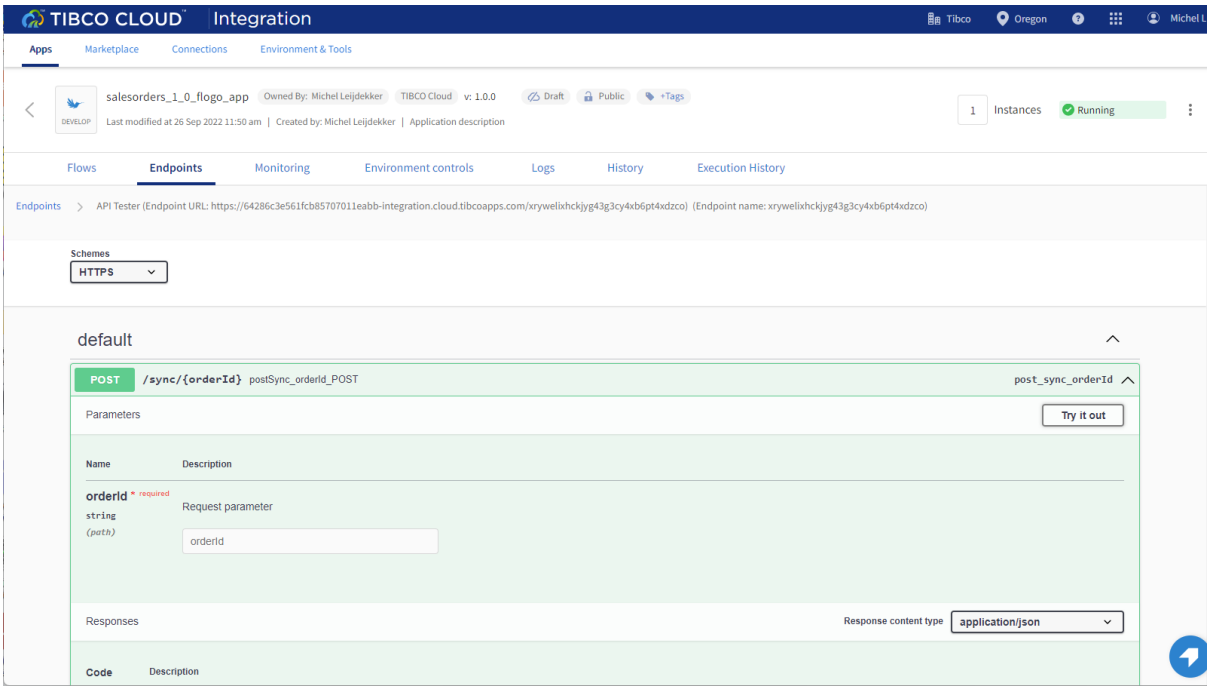

2. Now we have to scale the application to start an instance of the integration



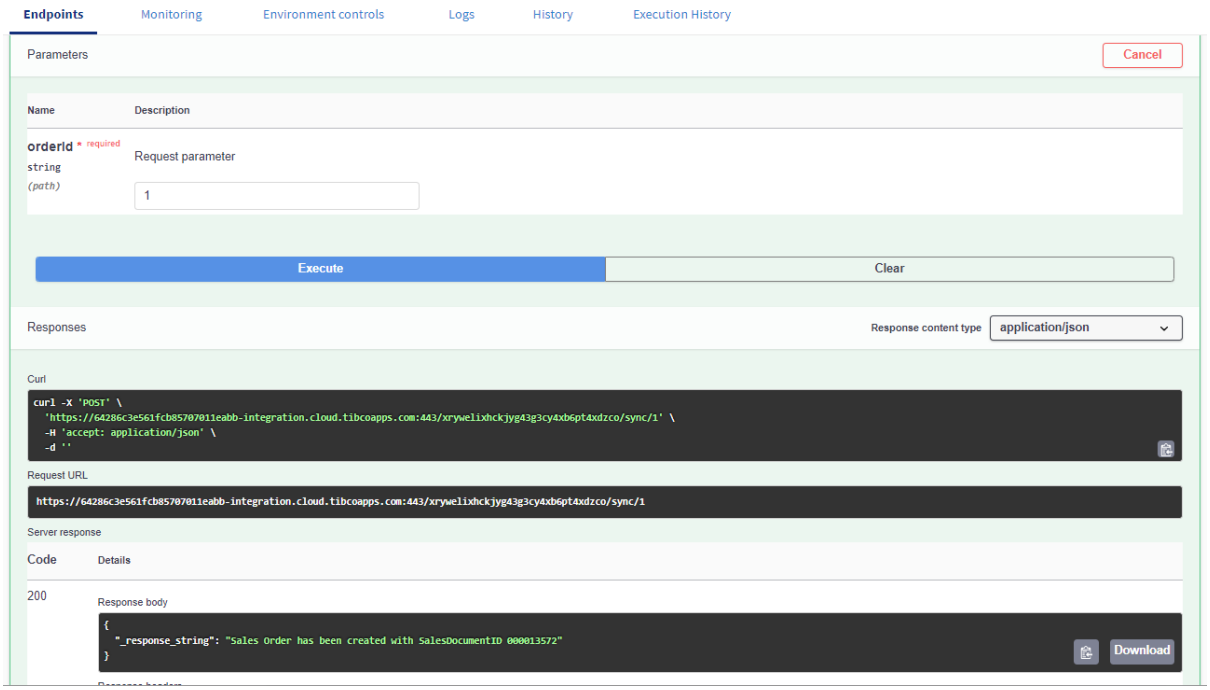

3. Once the **salesorders_1_0_flogo_app** app is running, click on the app and select the **Endpoint** tab. Select **Test** from the possible actions




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




5. 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.




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.

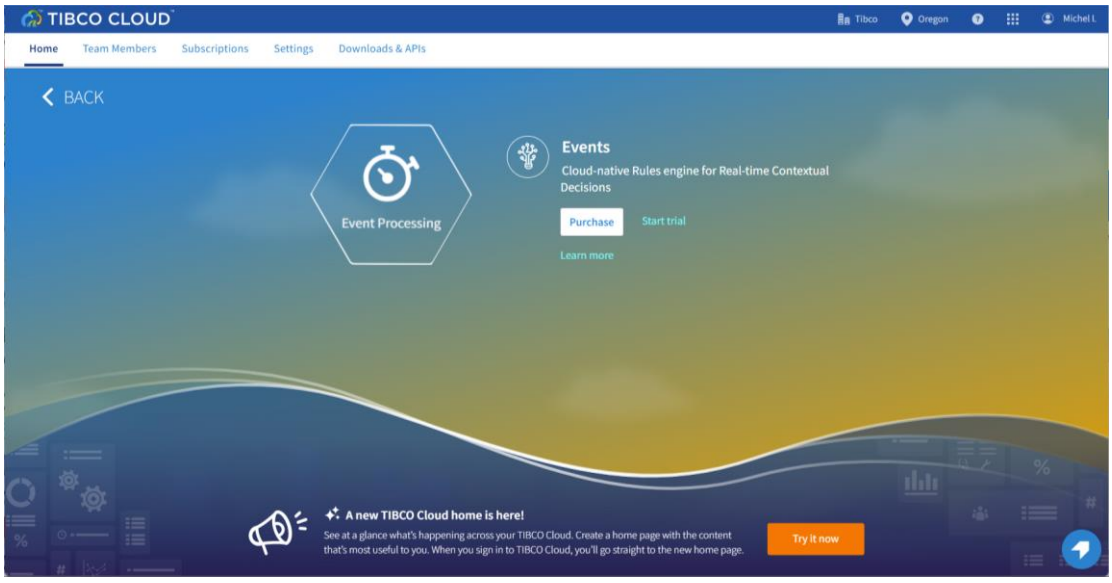
This walkthrough provides a step-by-step journey to create a Simple Decision Service.  You can have one or more Simple Decision Services as part of a single TCE application.

## What You Will Learn

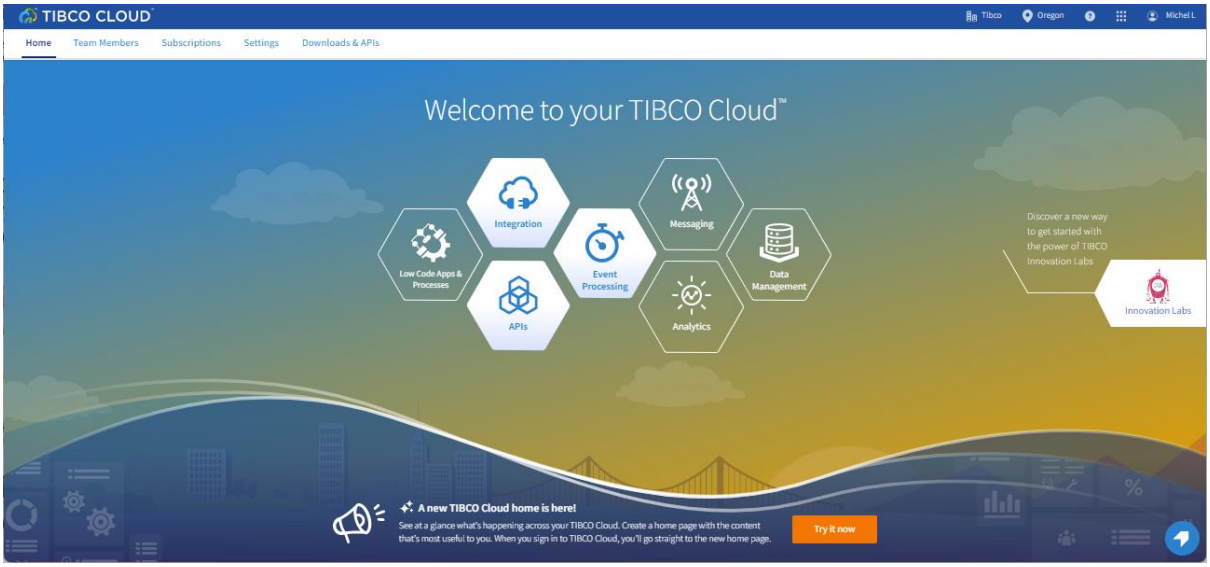
- Understanding the concept of Simple Decision Service(s).
- How to create, deploy and test a Simple Decision Service(s).

## Getting Ready

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



2. 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

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

*Figure 1: Push App Button*

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

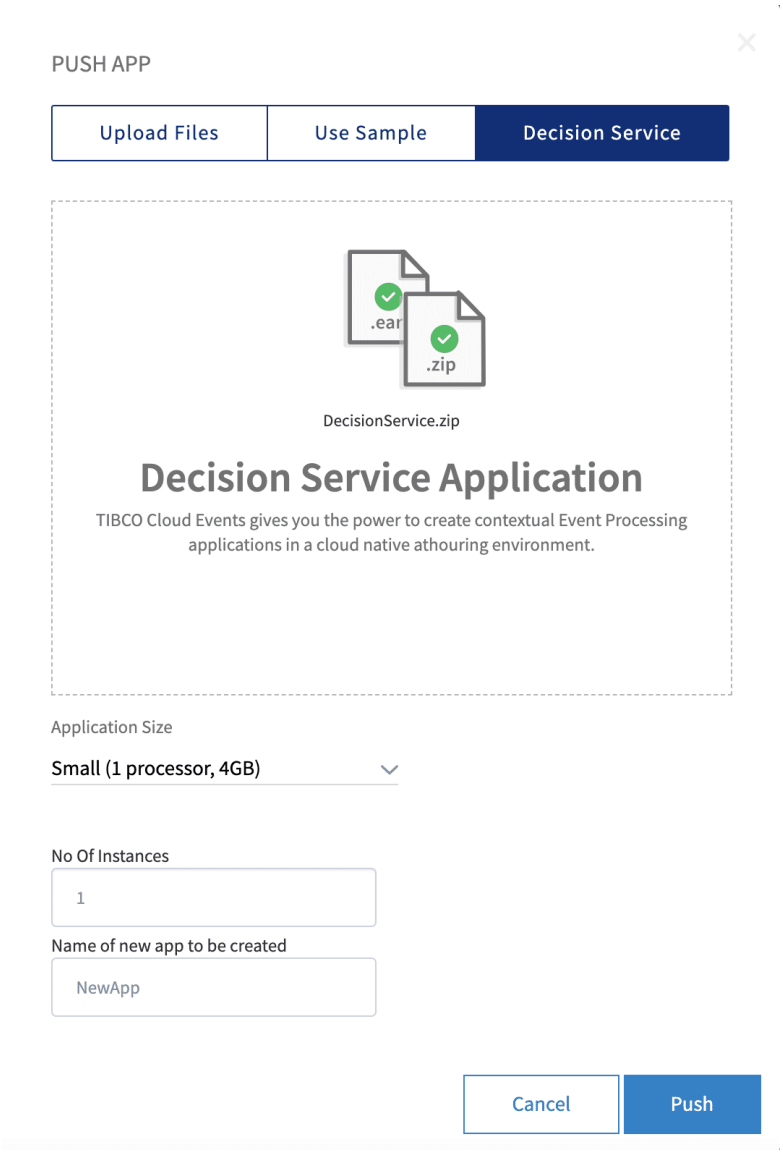



*Figure 2: SDS - Push Application*

5. 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.

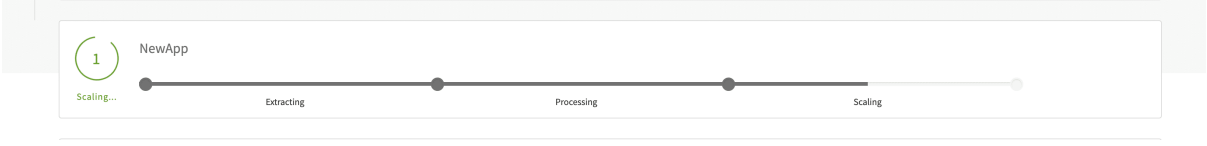



*Figure 3: Application Push Stages*


*Figure 4: Application Deployment*

6. 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

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

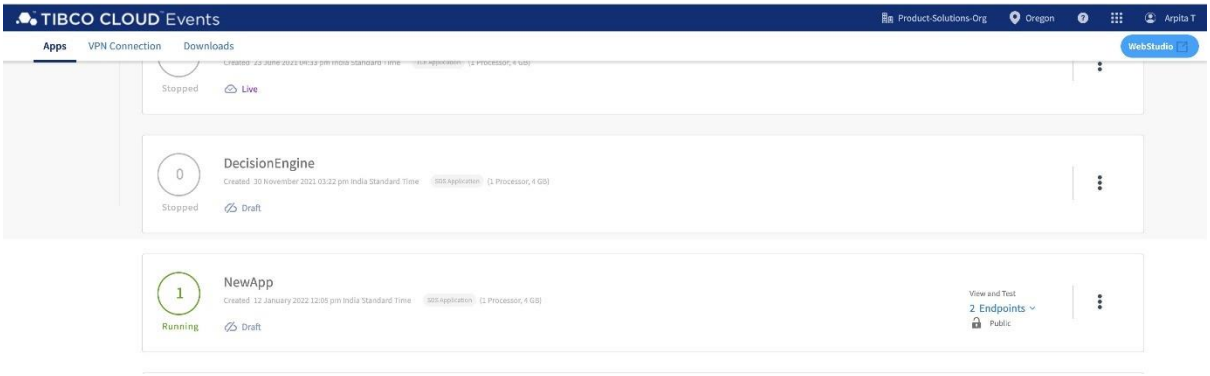



*Figure 5: TIBCO Cloud Events Home - WebStudio option*

3. 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.

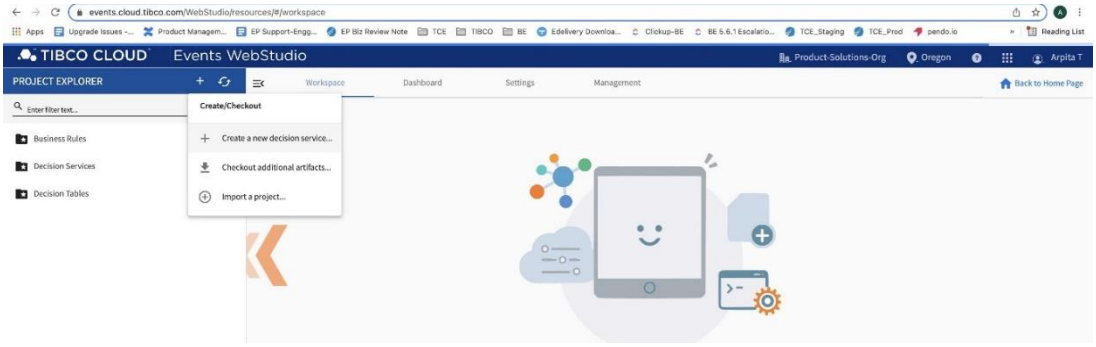



*Figure 6: Create a new Decision Service option*

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.

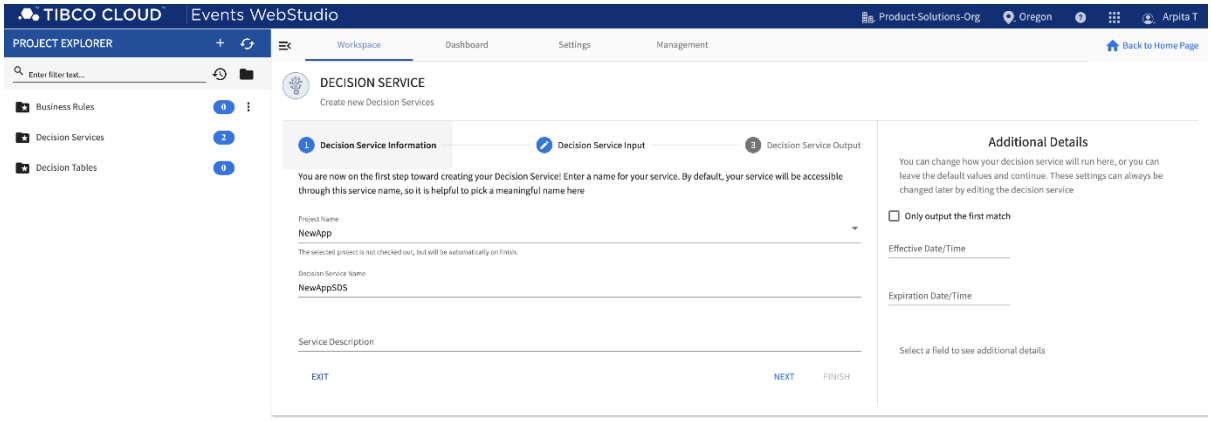



*Figure 7: Create Decision Service wizard*

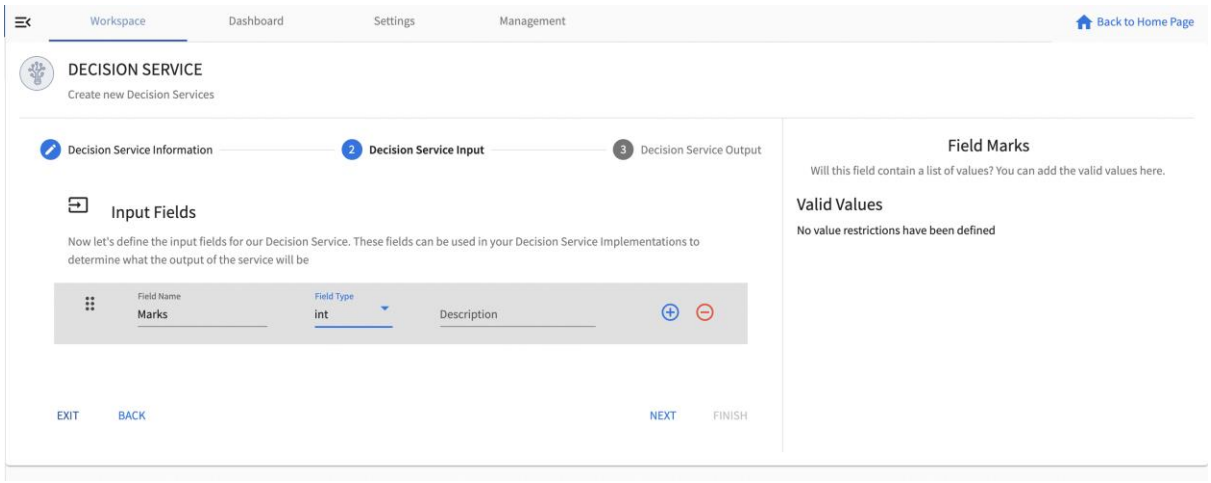



*Figure 8: Input fields of SDS*

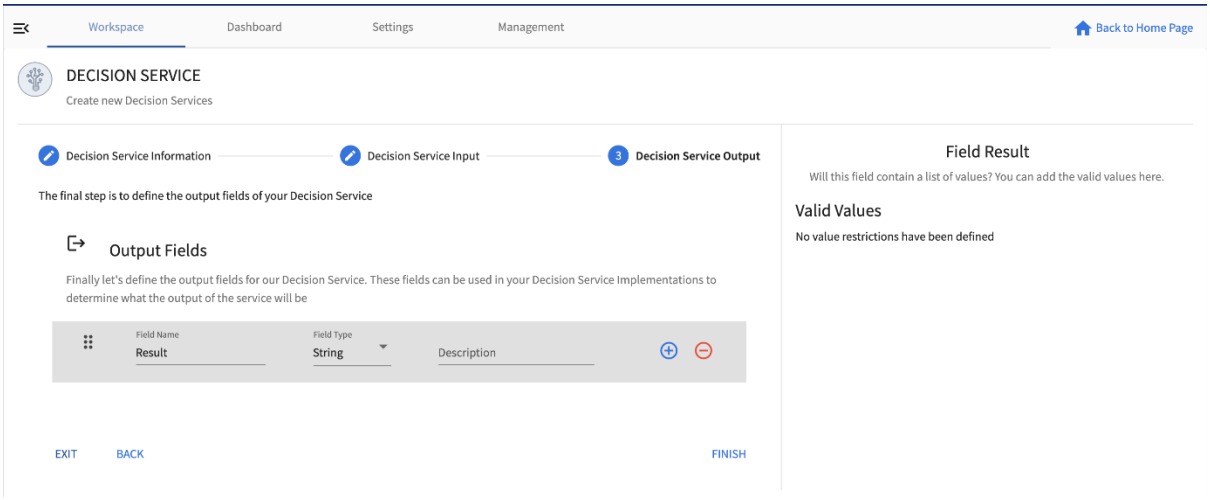



*Figure 9: Output fields of SDS*

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).

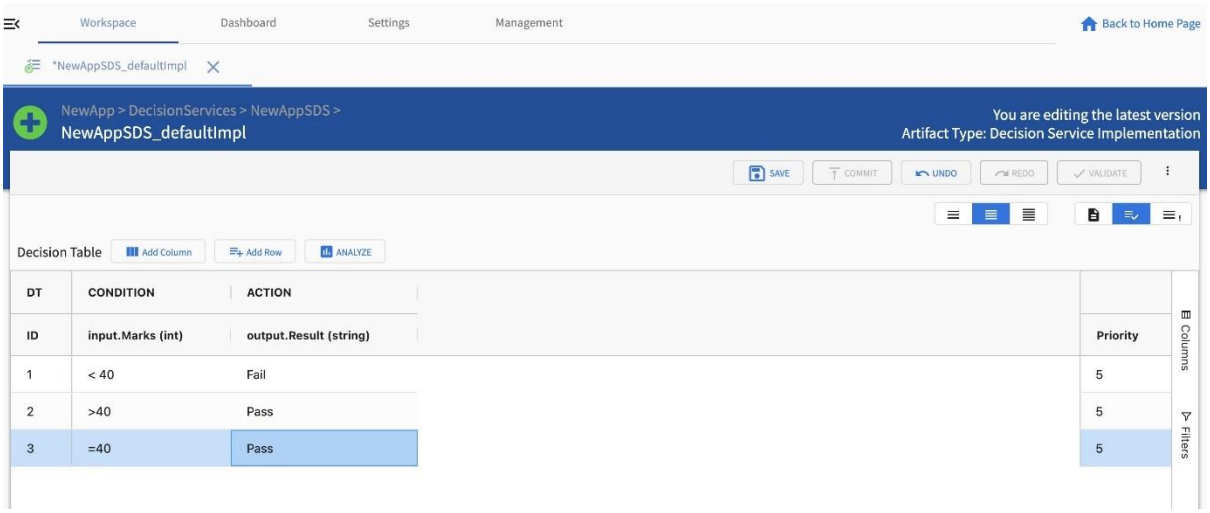



*Figure 10: Decision Service Implementation*

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.

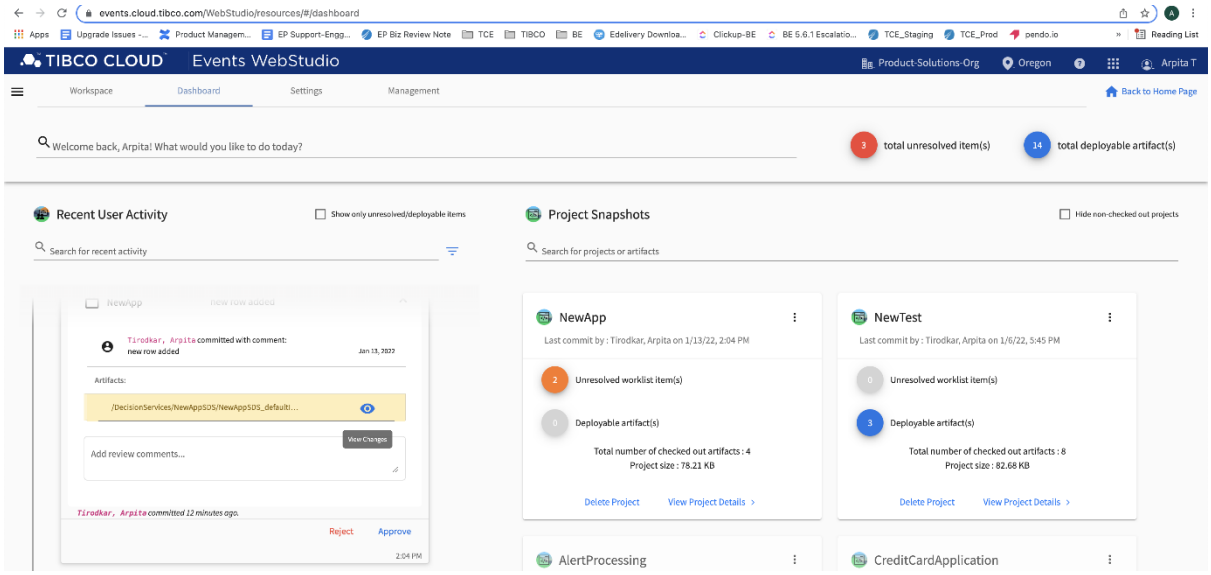



*Figure 11: Deployment Dashboard*

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.

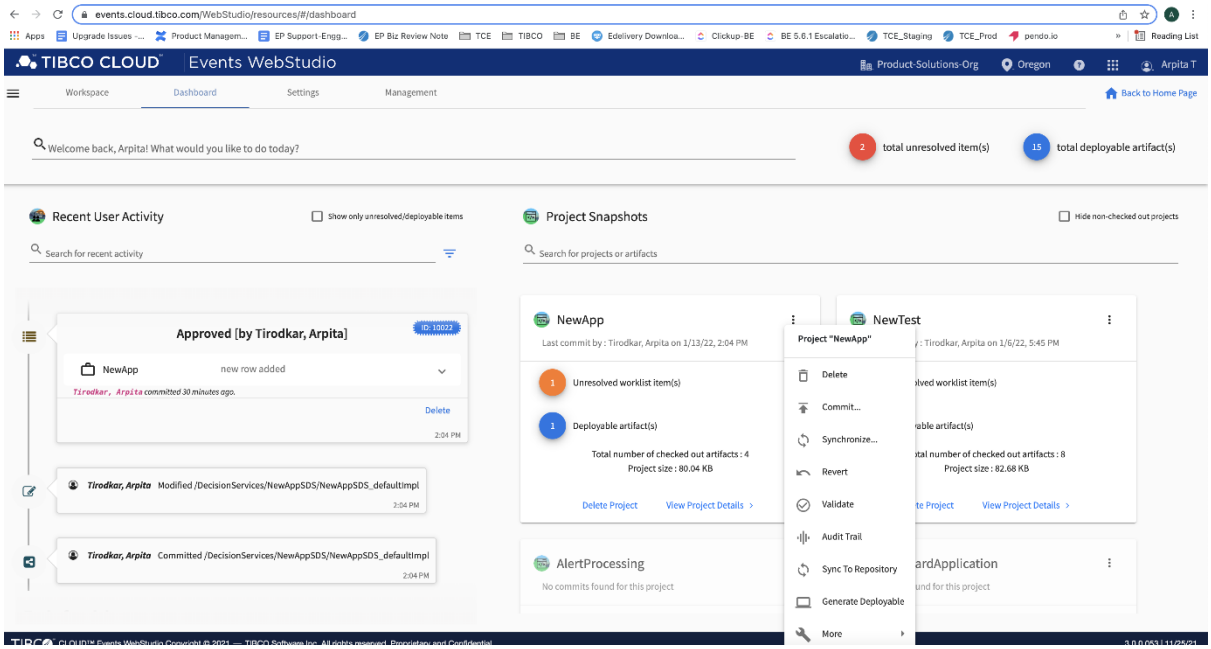



*Figure 12: Generate Deployable*

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.

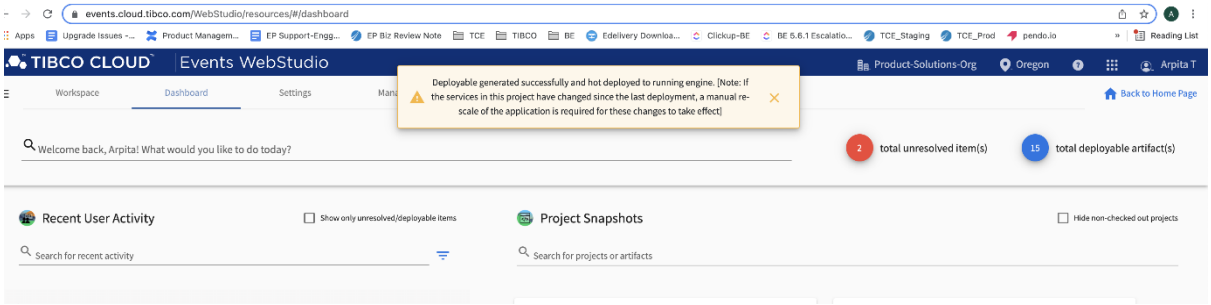



*Figure 13: Home page - view Endpoints option*

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.

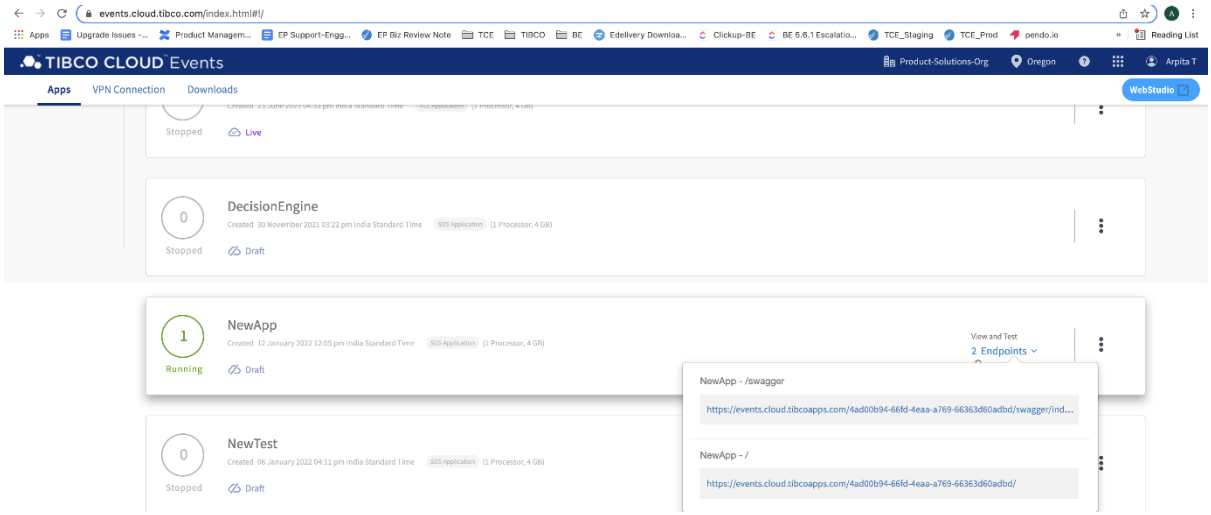



*Figure 14: REST Endpoints*

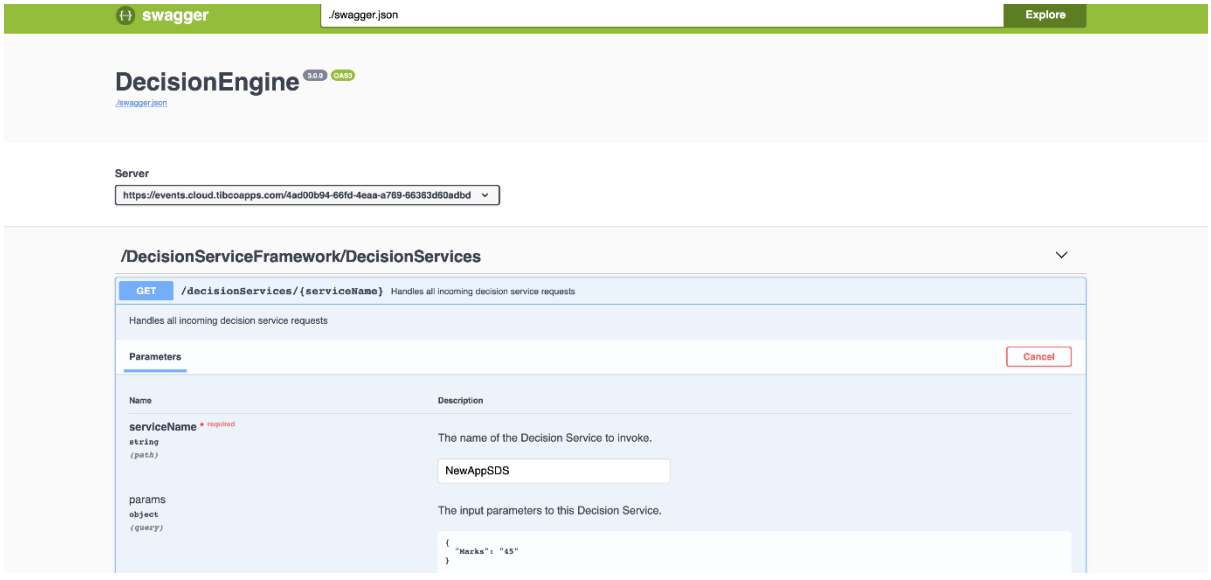



*Figure 15: Swagger Client*

**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.

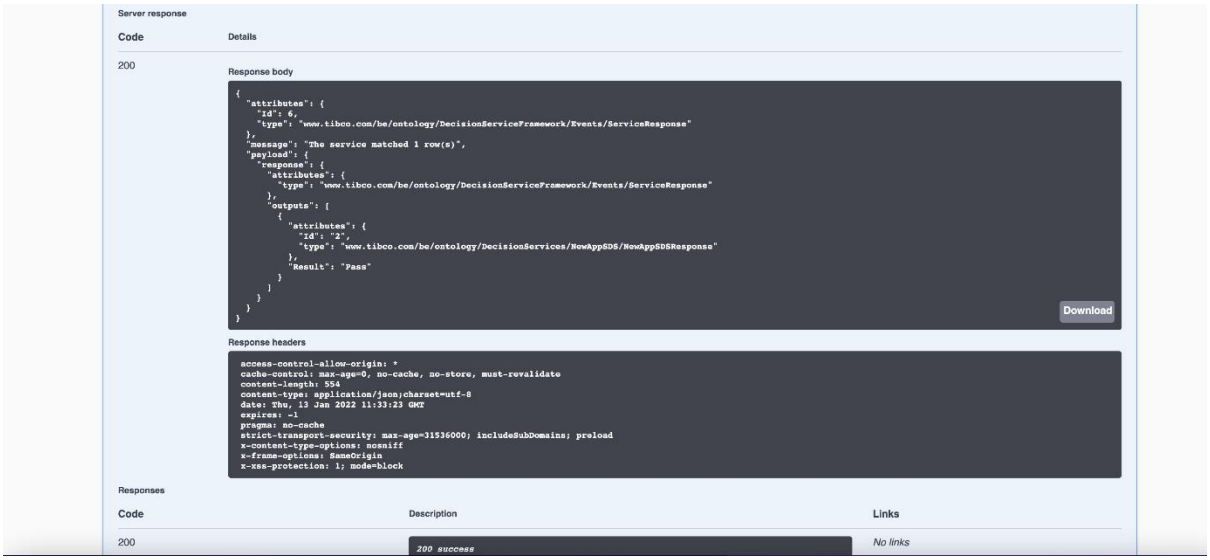



*Figure 16: Swagger Output*

# 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.

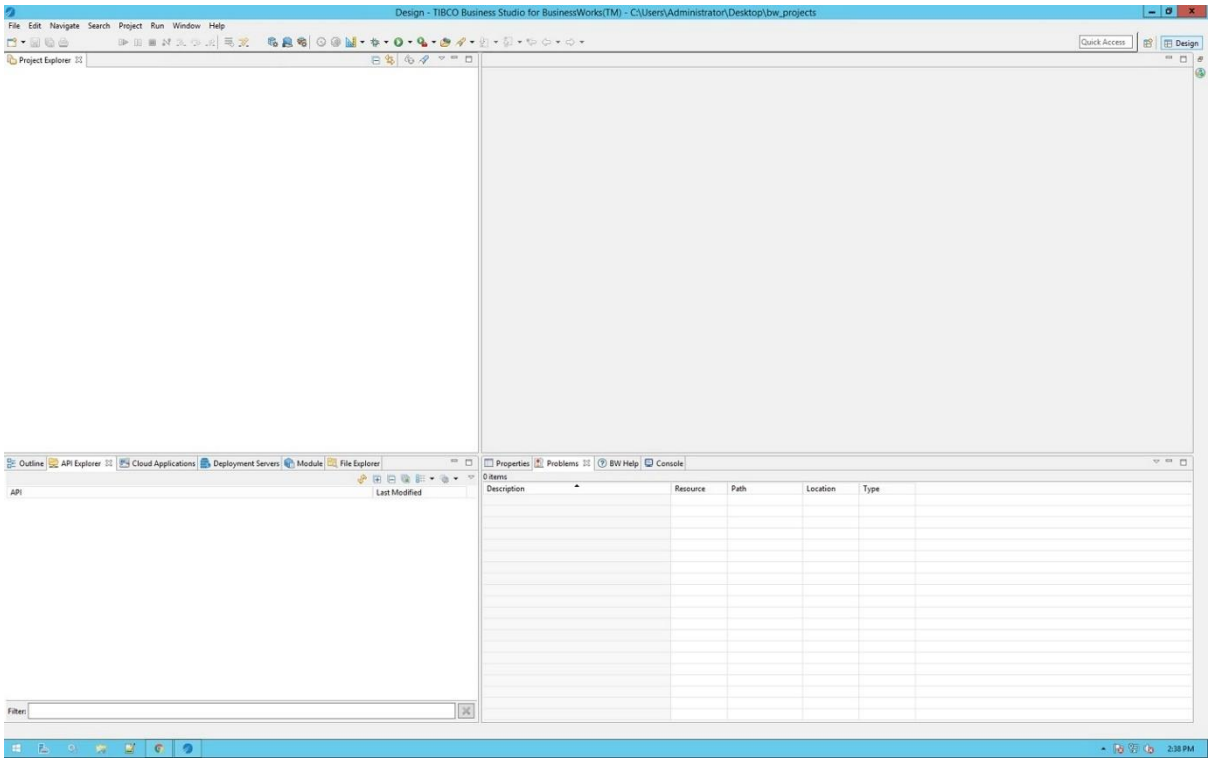
- For any feedback, queries or additional information, please reach out to [mailto:support@tibco.com](mailto:support@tibco.com)

# 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

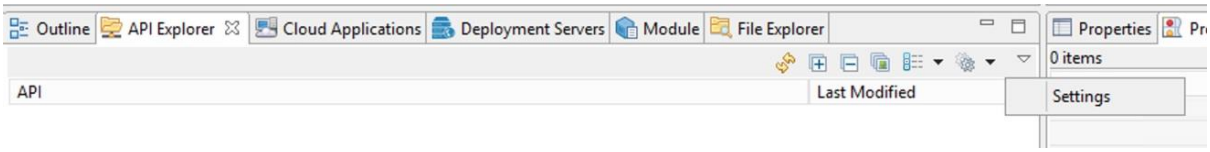
To get ready, open TIBCO Business Studio™ for BusinessWorks.
Assuming you start with a clean enviroment, your screen should look similar to this:




## 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:

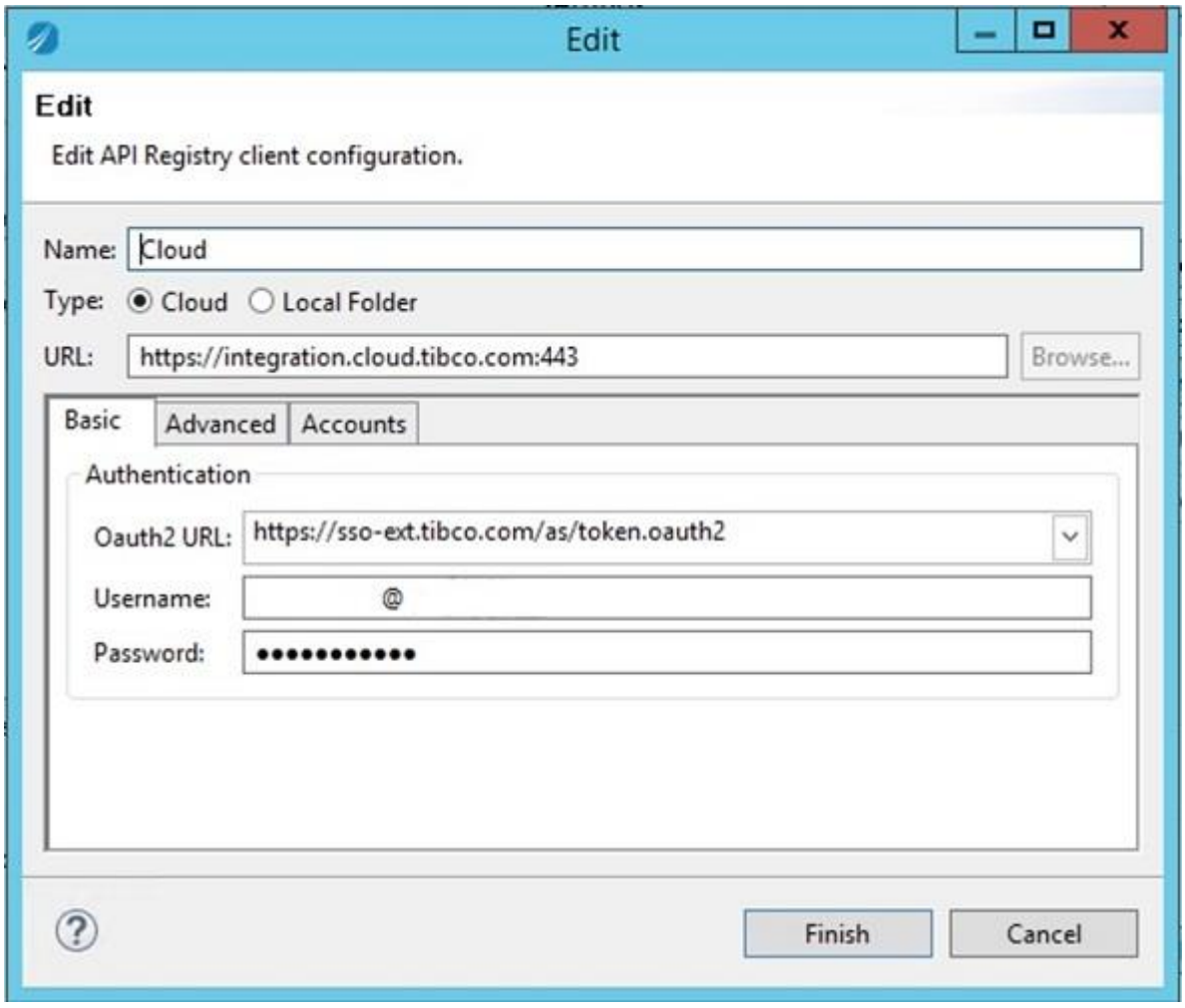
1. Go to API Explorer, click the down arrow in the upper right corner, and click Settings.



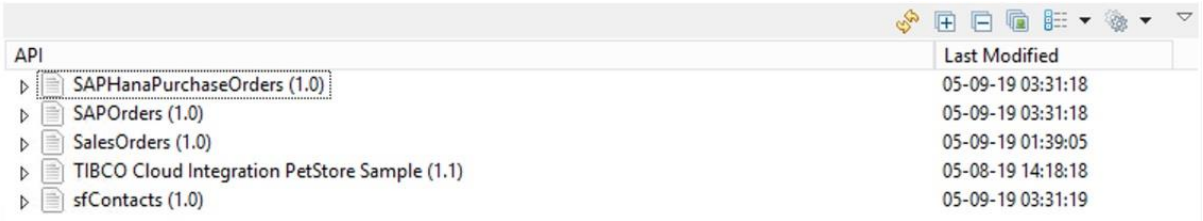

2. In the **Configure API Settings** form, click the **New** button. Fill out the **URL** and your TIBCO Cloud Integration username and password in the new form.

| Field | Value |
|---|---|
| **URL** | `https://integration.cloud.tibco.com:443` |
| **Username** | `your@email.com` |
| **Password** | `YoUrPassWoRd` |

3. Your screen should look like this:



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

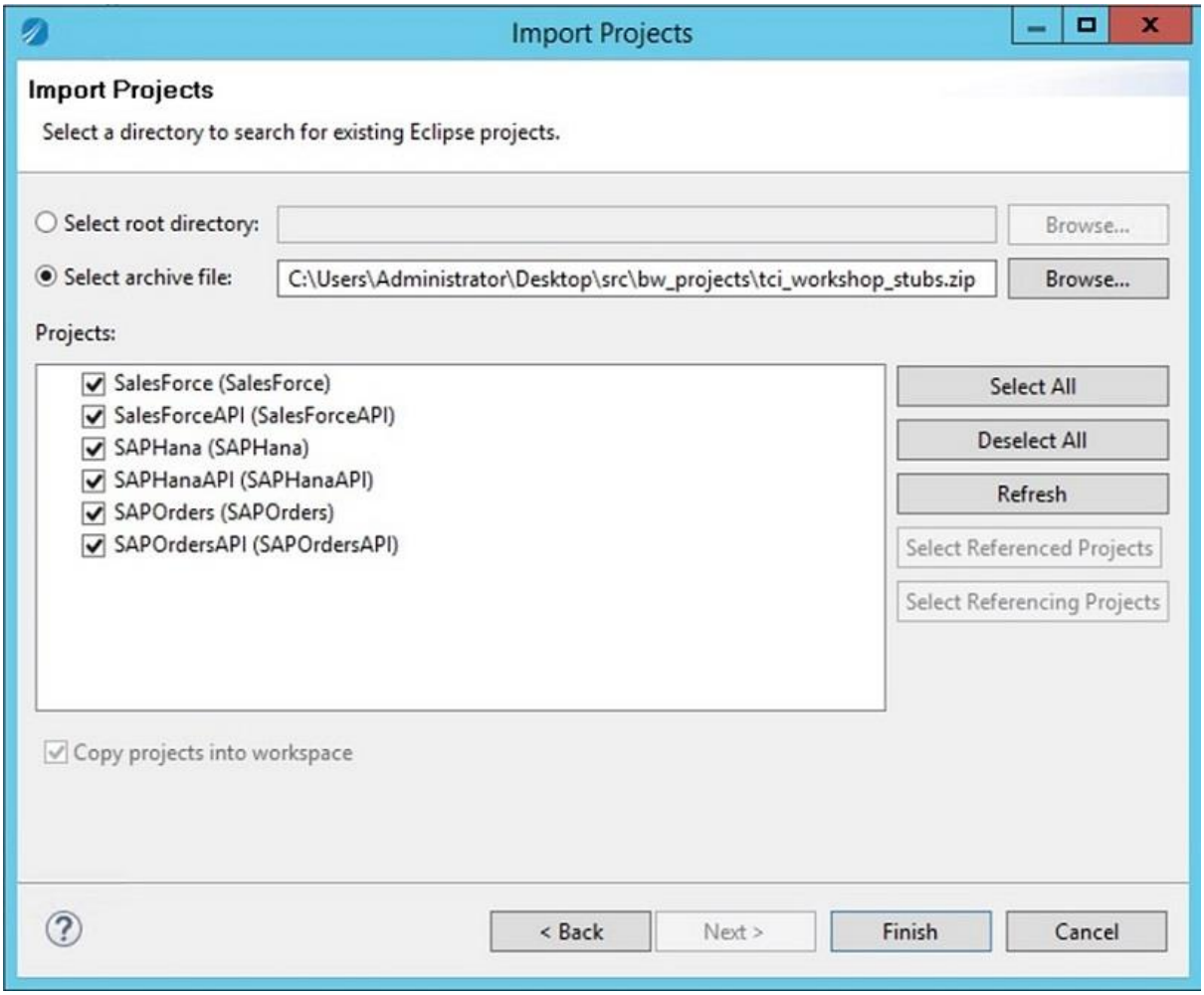



# 3 How to Do It: Import a Project in to Business Studio

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:

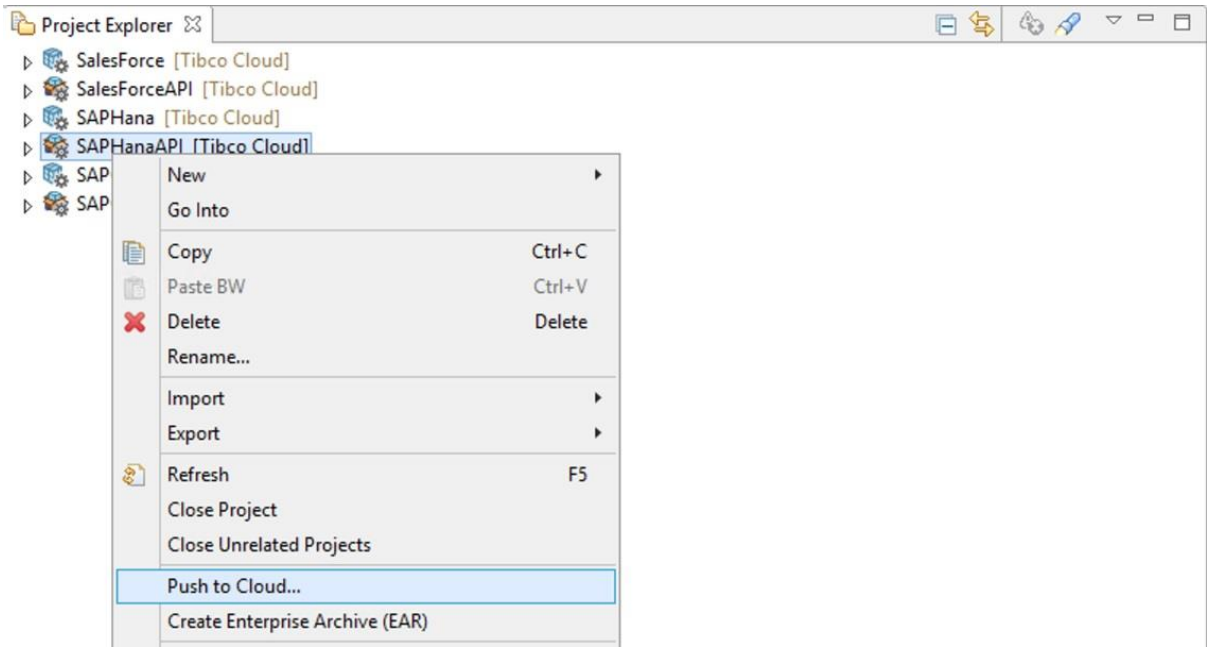



3. Click the **Finish** button. The Project Explorer should look like this:

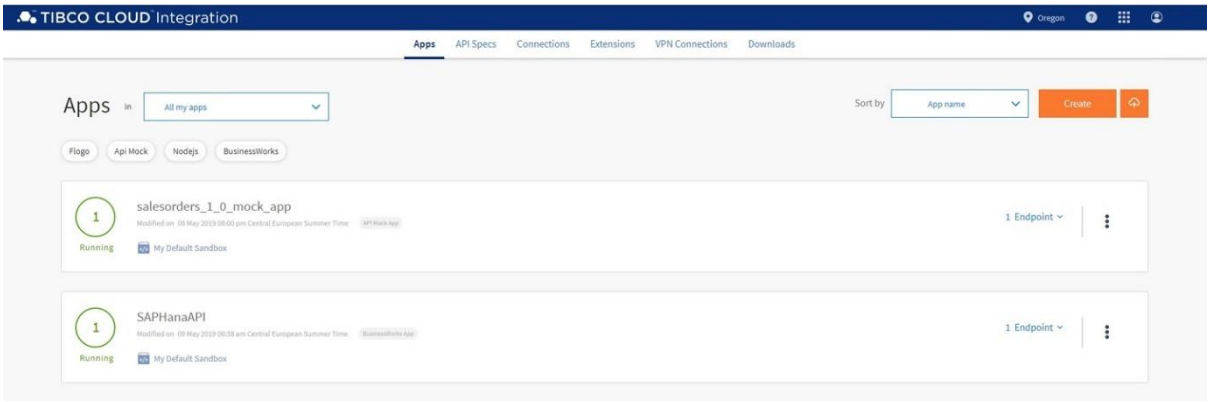

## 4 How to Do It: Push the Project to TIBCO Cloud and Test

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

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

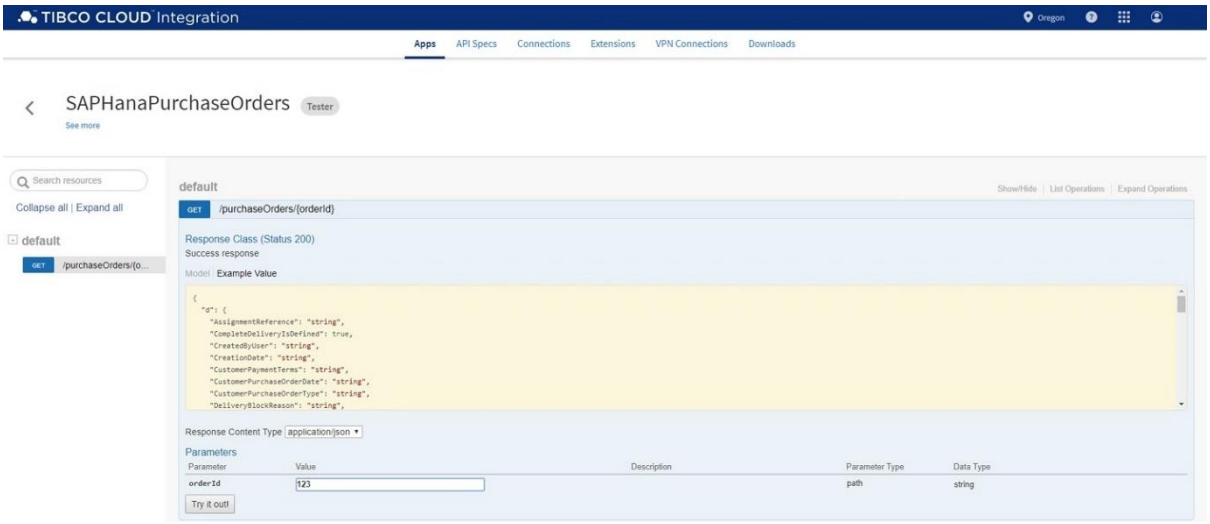



2. Navigate to TIBCO Cloud, and once the app has been successfully pushed, your screen should look like this:




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:




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.