Global Headquarters
3307 Hillview Avenue
Palo Alto, CA 94304
Tel: +1 650-846-1000
Toll Free: 1 800-420-8450
Fax: +1 650-846-1005
www.tibco.com

TIBCO Cloud Integration Workshop

How to Create Process API's using TIBCO Flogo

Introduction **2**

Workshop Flow **2**

Product Overview **3**
- TIBCO Cloud Integration 3
- API Management 5

Before you start **6**
- Creating a TIBCO Cloud Integration Trial Account 6
- Creating a TIBCO Mashery Trial Account 8
- Downloading Postman 9
- Accessing TCI 9

API Modelling **10**
- Using Groups 10
- Creating an API Specification 11
- Adding a Resource to the API 12
- Configuring the API Resource's POST Request 13
- Configuring the Resource's POST Response 16
- Creating an Application from the API Specification 19
- Mock Application 19

Creating a Web Based Flogo App **21**
- Adding a getTransactionID Activity 26
- Adding a getCustomerInfo Activity 27
- Adding a LogMessage Activity 30
- Adding a getProductInfo Activity 31
- Adding a putOrder Activity 32
- Complete the HTTP Response via the Return activity 33
- Testing the Flogo App 35
- Pushing and Testing the App 37

Creating an API from your Application **39**
- Pushing Flogo App as an API 39
- Verifying the API Deployment 43
- Generating a New Application 45

Testing the API **48**

Summary **49**

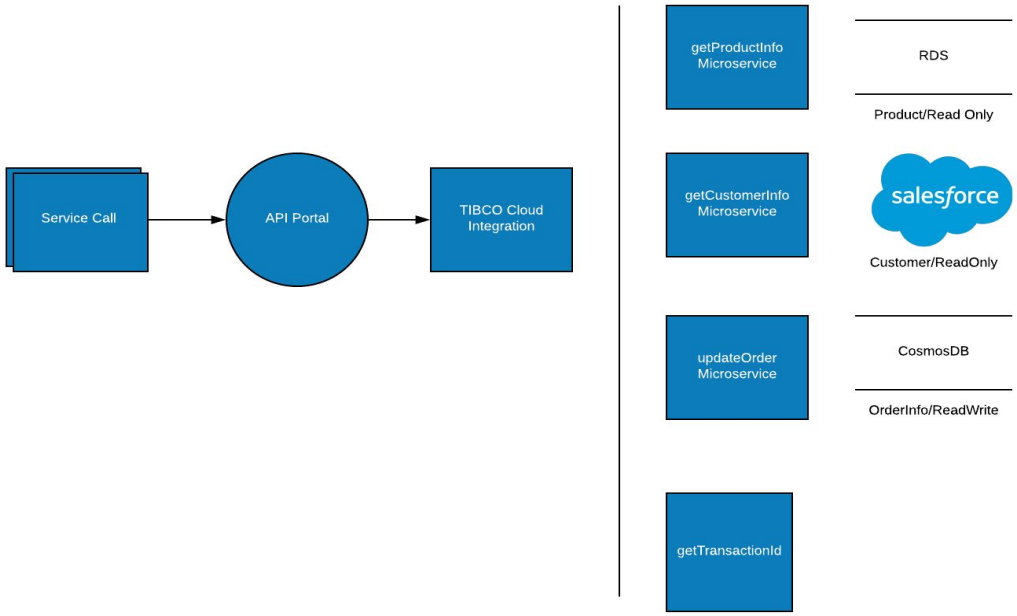
Introduction

This workshop is a hands-on experience which will allow the user to create an API Led integration, starting from API Specification, implementing the spec as a microservice, all the way to deployment and management of the API. This workshop will enable the user to integrate an order system getting product information from a database running on AWS RDS, retrieve customer information from Salesforce, and updating the completed order into CosmosDB running on Azure.

Workshop Flow

The workshop will consist of the following:

1. Create an API Specification
2. Create a mock application to allow development of mobile and web based applications
3. Create an application to consume 4 exposed Microservices
4. Expose the final API for consumption



Product Overview

TIBCO Cloud Integration

CONNECTOR MARKETPLACE

TIBCO Cloud Integration offers connectors for hundreds of applications, databases, protocols, and formats. Simply enter your credentials, and the connector discovers metadata, including custom objects and fields, and provides operations such as query, lookup, create, update/upsert, and delete. It offers connectors in many ecosystems, including:

- CRM SYSTEMS such as Microsoft Dynamics 365/CRM, Salesforce, and SugarCRM
- MARKETING AUTOMATION SYSTEMS like HubSpot, IBM Silverpop, Marketo, Oracle Eloqua, Pardot, and Salesforce Marketing Cloud
- ERP SYSTEMS including Microsoft Dynamics 365/GP/NAV, NetSuite, and SAP

For the current list of connectors, go to: https://www.tibco.com/connected

RISK-FREE DATA MIGRATION AND REPLICATION

- Data Migration: Extract and migrate your data simply and efficiently. Copy data from an old application to a new one or load production data to a test instance.

- Data Replication: Use a simple wizard to copy, archive, and analyze your business critical data. Processes are completely automated including scheduling options and auto-detection of changes. You own your data, with no manual exports or imports required.

ENTERPRISE-READY

More complex integration flows are handled through rich visual tooling that implements choreography, transformations, and routing options. You have the option to work in a familiar Eclipse-based IDE using an intuitive, full-featured drag-and-drop palette of integration functions. Using the same environment, multi-operation services can be examined at a glance, designs can be debugged and modified without stopping the debugger, and execution can be monitored. Hand coding is possible, but strictly optional.

API Management

The cloud-native API platform you can deploy anywhere, and manage APIs from everywhere
Your digital business requires a market-leading API platform to enable API-led innovation and agility, new business models, and enterprise-scale security to protect your assets. APIs are connective tissue of every digital business platform, from powering digital marketplaces to seamless API-led connectivity of legacy, cloud, and data from the edge. As enterprises increasingly adopt cloud-native tools for more speed, agility, and scale, your API program must similarly evolve. cloud-native API programs can supercharge key transformation enablers like microservices, containers, and serverless compute.

TIBCO Cloud™ Mashery® delivers market-leading full lifecycle API management capabilities for enterprises adopting cloud-native development and deployment practices, such as DevOps, Microservices, and Containers. Its rich set of capabilities includes API creation, productization, security, and analytics of your API program and community of developers. Now, the cloud-native enterprise has a platform to power their digitally-disruptive initiatives. TIBCO Cloud Mashery: The cloud-native API platform you can deploy anywhere, and manage APIs from everywhere.

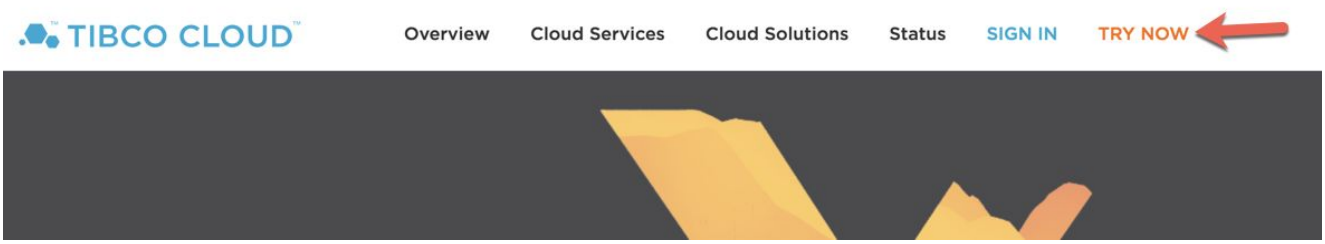
Before you start

Before we begin the workshop you will need the following:

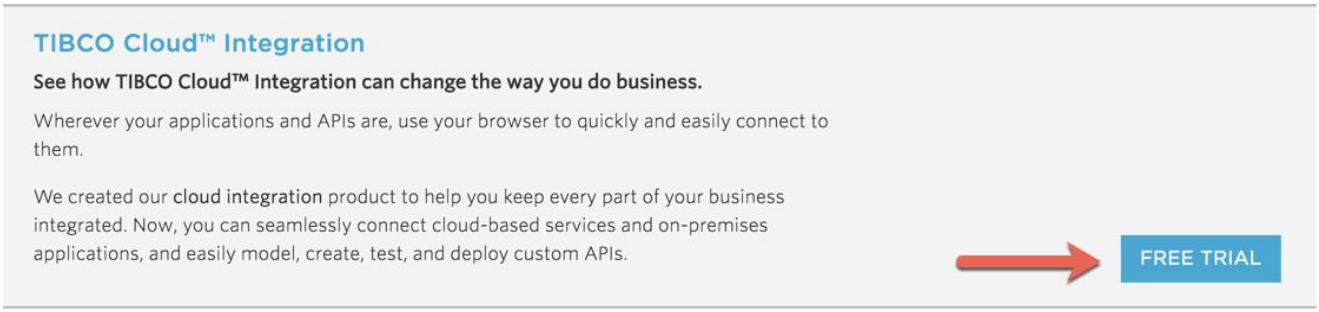
1. Trial subscription to TIBCO Cloud Integration
2. Trial subscription to TIBCO Cloud Mashery
3. Download an API Tester. In this case we will use Postman

Creating a TIBCO Cloud Integration Trial Account

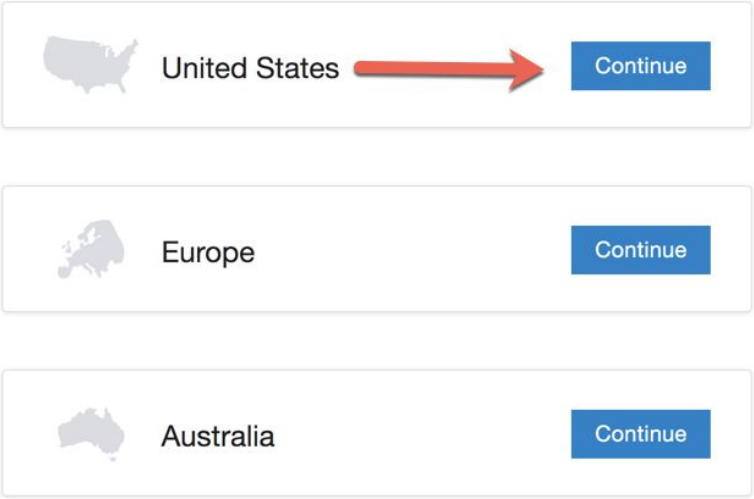
In order to create a TIBCO Cloud Integration Trial Account you will need to navigate to https://cloud.tibco.com



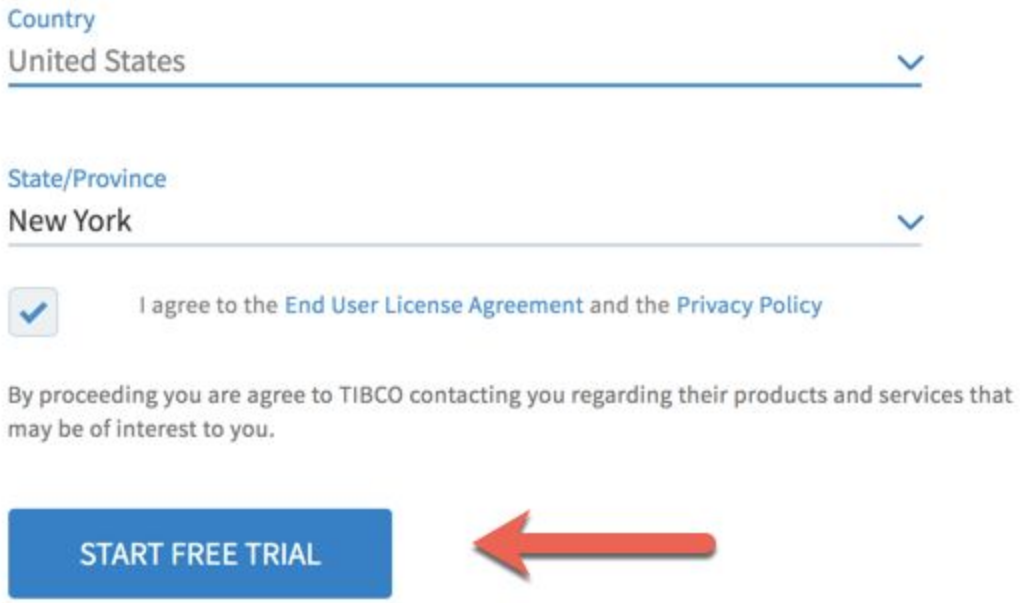
Click on TRY NOW



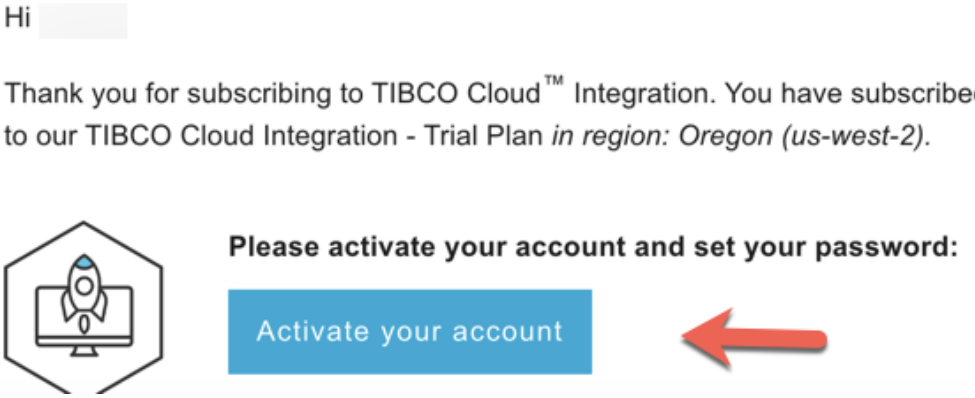
Click on FREE TRIAL



Select the appropriate region in this case United States



Enter your information and click on Start Free Trial



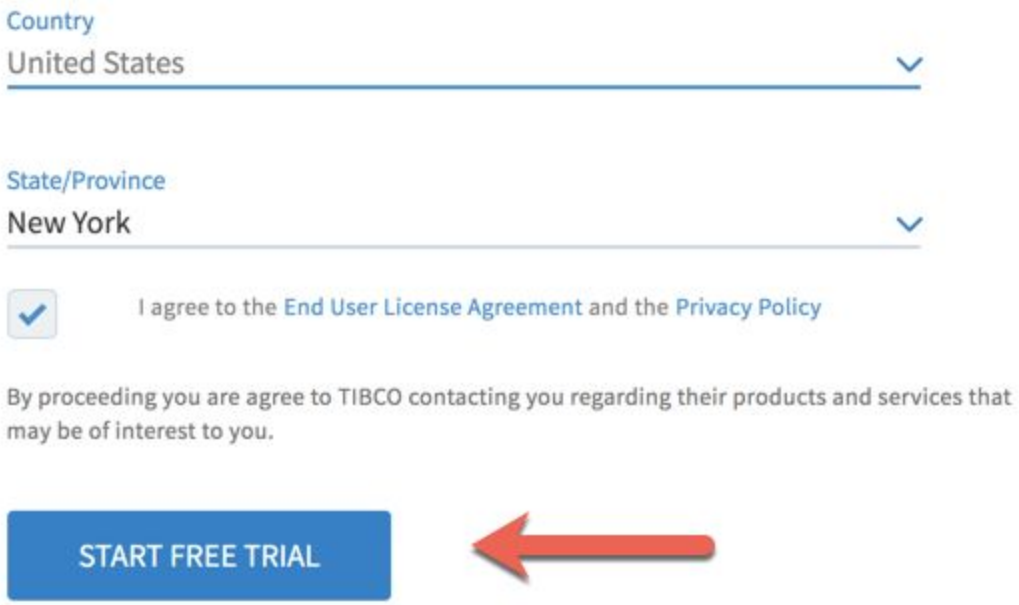
Lastly check your email and activate your new TCI Trial Account

Creating a TIBCO Mashery Trial Account

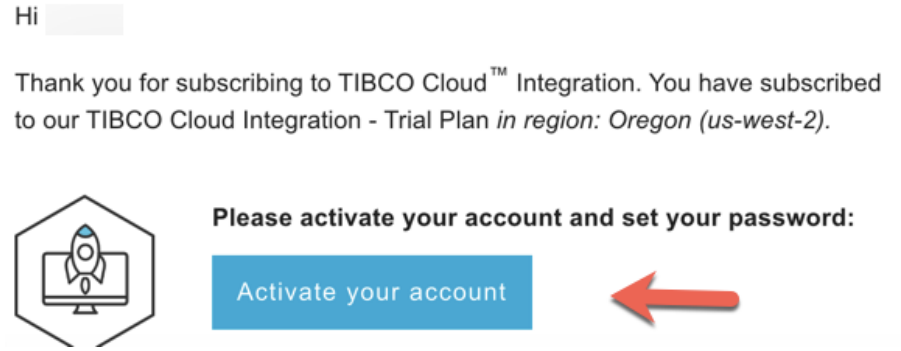
For the second part of this workshop you will also be publishing your API to TIBCO Mashery. For that you will need to create a TIBCO Mashery Trial Account:

Once again navigate to https://cloud.tibco.com

Click on FREE TRIAL



Enter the appropriate Information and click on START FREE TRIAL



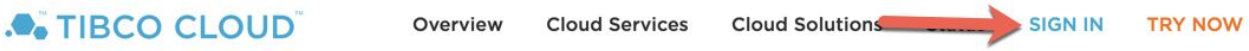
Remember to verify your account.

Downloading Postman

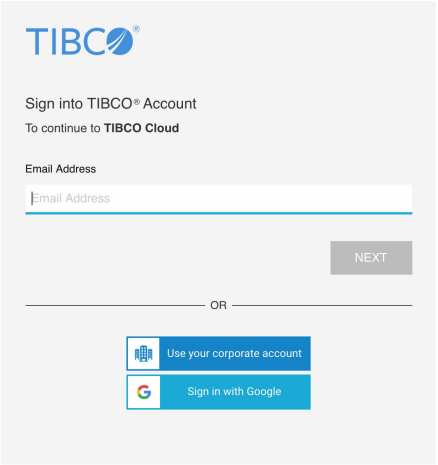
In order to test our completed API's we will be using Postman. Download Postman from the following URL: https://www.getpostman.com/downloads/

Accessing TCI

Navigate to https://cloud.tibco.com



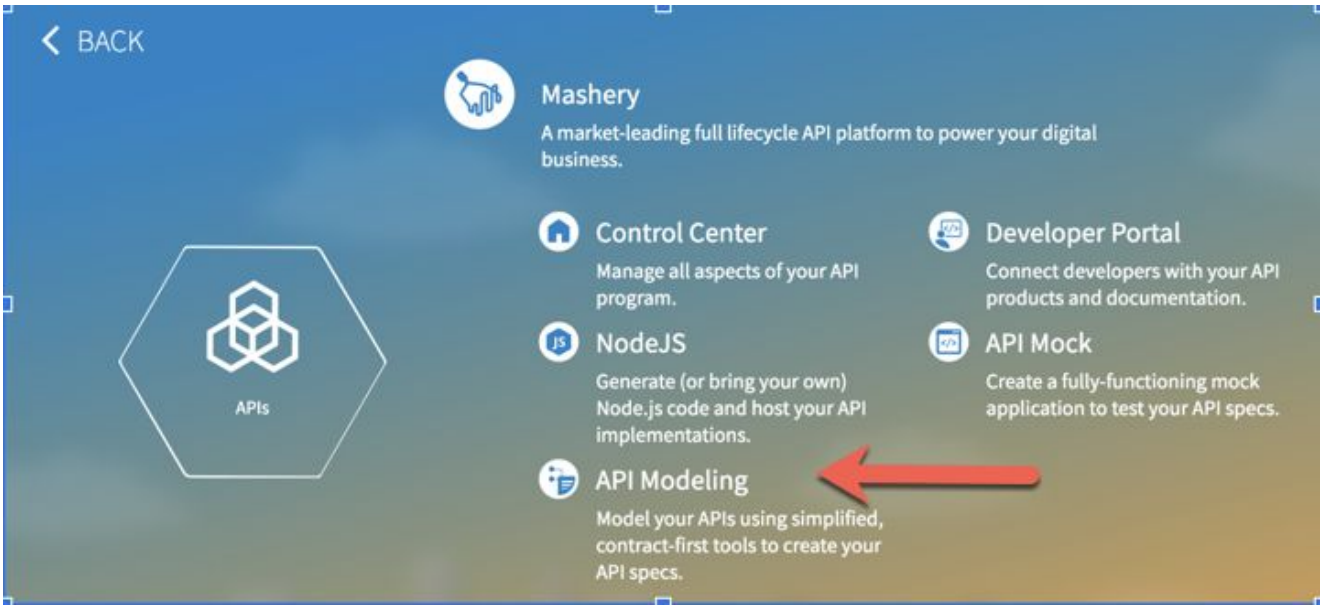
Click on SIGN IN



You should now see the capabilities you have access to (Unless you also registered for additional capabilities).

API Modelling

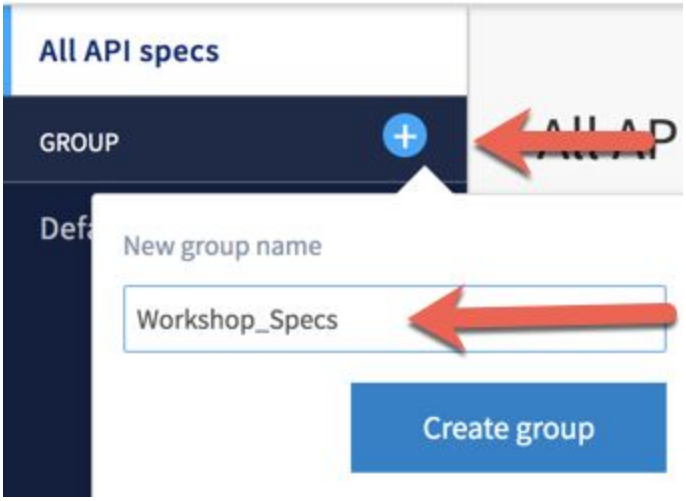
We first need to create the specifications for our API, so we will access the “APIs” tile from the main menu, and then select “API Modeling”.



Using Groups

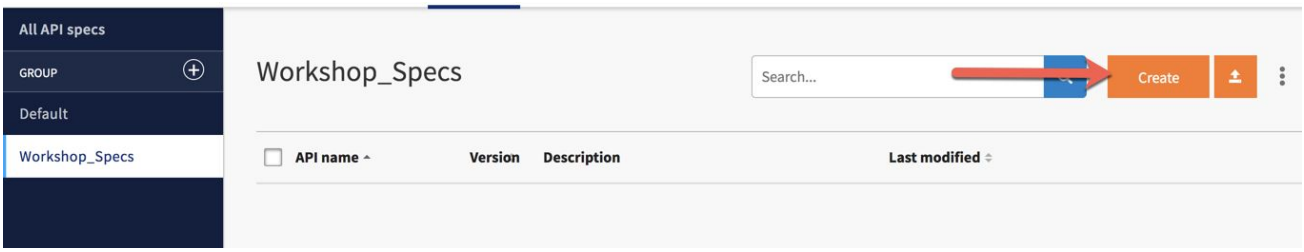
TCI is a collaborative environment where multiple users can create, manage, and review API specifications within the team. Users can create groups to logically group API specifications, for example for a specific application, projet, or user.

Create a new group for the workshop called Workshop_Specs as shown below:

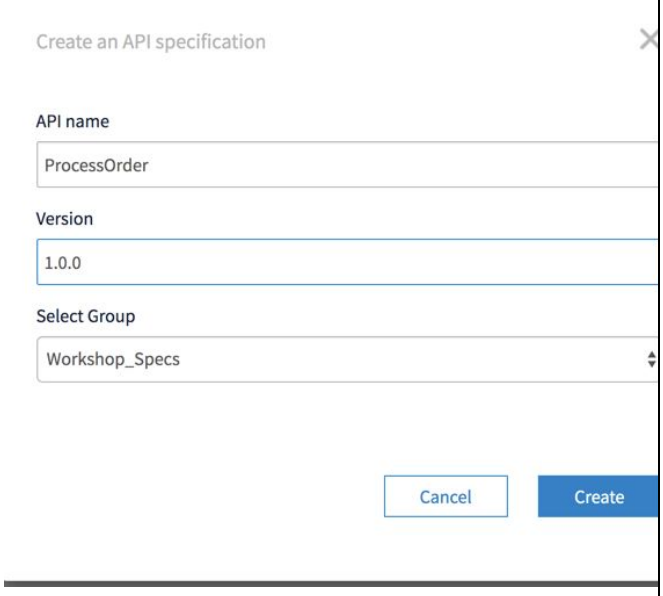
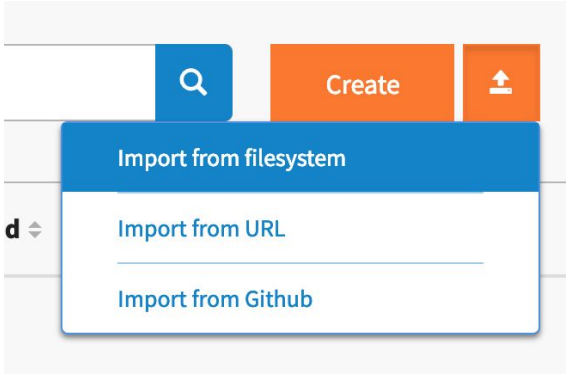


Creating an API Specification

Click on the group you created, and then click the “Create” button to create a new API specification to begin our development



For this workshop please create a new “ProcessOrder” API specification. You can use version 1.0.0, and make sure to select your group as illustrated on the left-end side below :

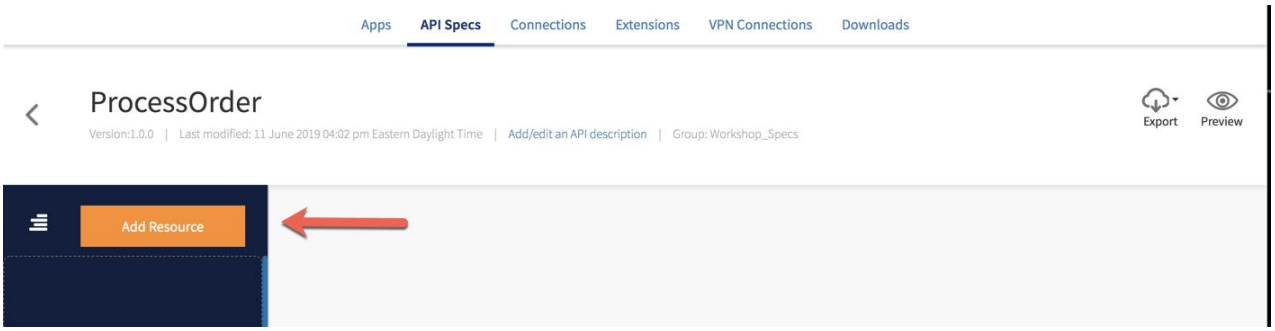
Either Create an API Spec from scratch	OR Import an existing Spec
	

Note: TCI also supports exporting/importing API specifications. If an API of same name and version exists within the team (not just within the group you created), it will fail to import or create because API Specifications are meant to be team developed and tested.

Adding a Resource to the API

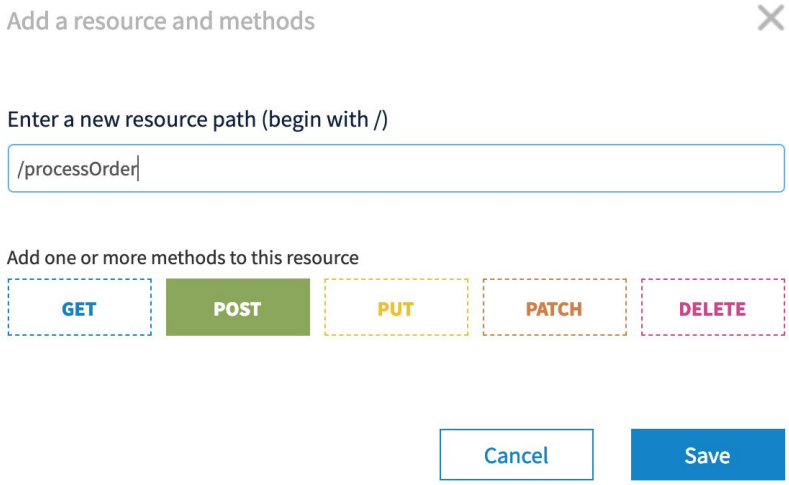
The next step is to add a resource to the API specification, and defining its associated methods.

To create a new resource, click on the “Add Resource” button on the left as illustrated below:



When creating a resource path we need to specify the resource path.
For this workshop, enter “/processOrder”.

We must then select which method to implement from GET, POST, PUT, PATCH, and DELETE.
In this workshop we will implement the POST method, please click on “POST”

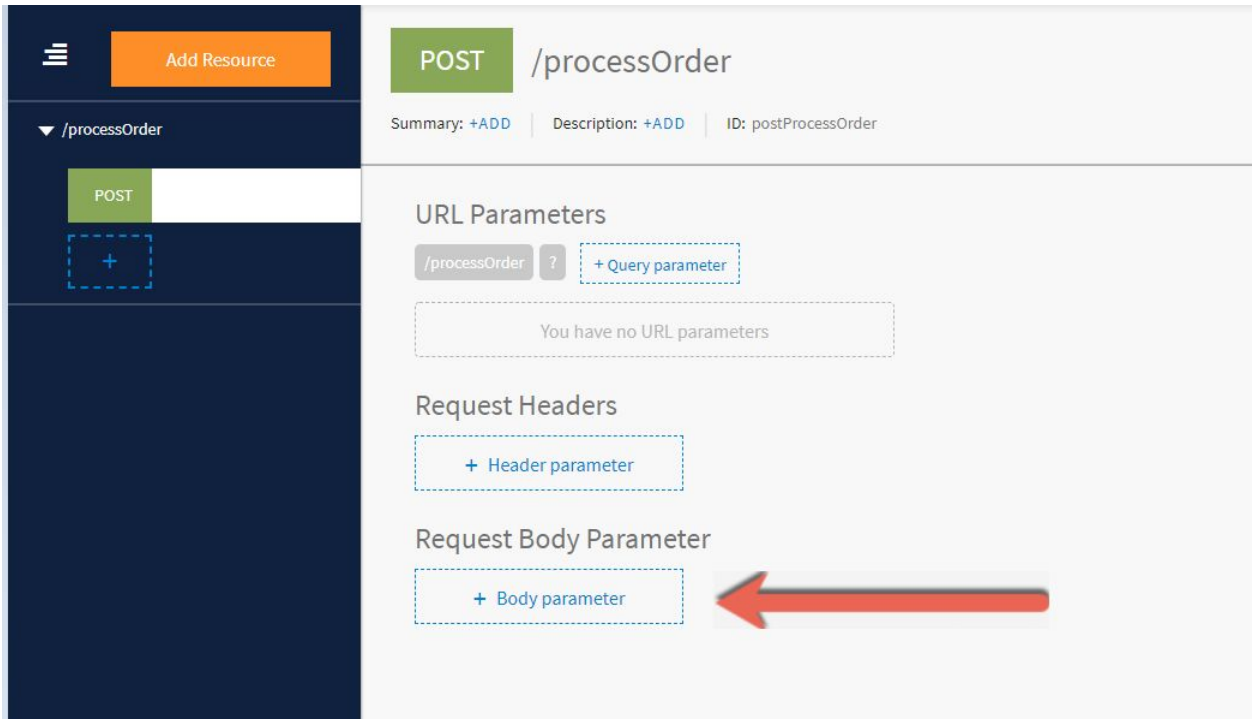


Configuring the API Resource's POST Request

As best practice for API development you should add a summary and a description (by clicking the "+Add" buttons in the header). This is optional for this workshop.

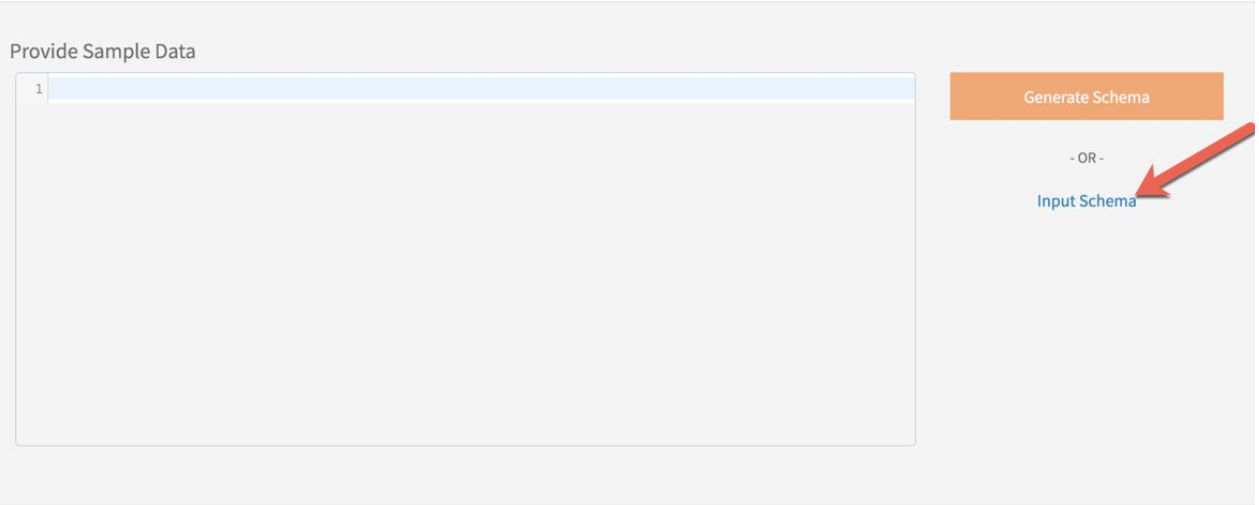
To configure the POST Request, make sure the "Request" button is highlighted as illustrated below.

To configure the request body parameters, please click on "+ Body parameter":



Next we will generate a request schema from a sample request.

Click on Input Schema



Please enter OrderInput for a name and paste the following below:

```
{
   "type": "object",
   "properties": {
      "CustomerID": {
         "type": "string",
         "default": "2"
      },
      "ProductID": {
         "type": "string",
         "default": "1"
      },
      "Qty": {
         "type": "string",
         "default": "2"
      }
   }
}
```

OrderInput

array

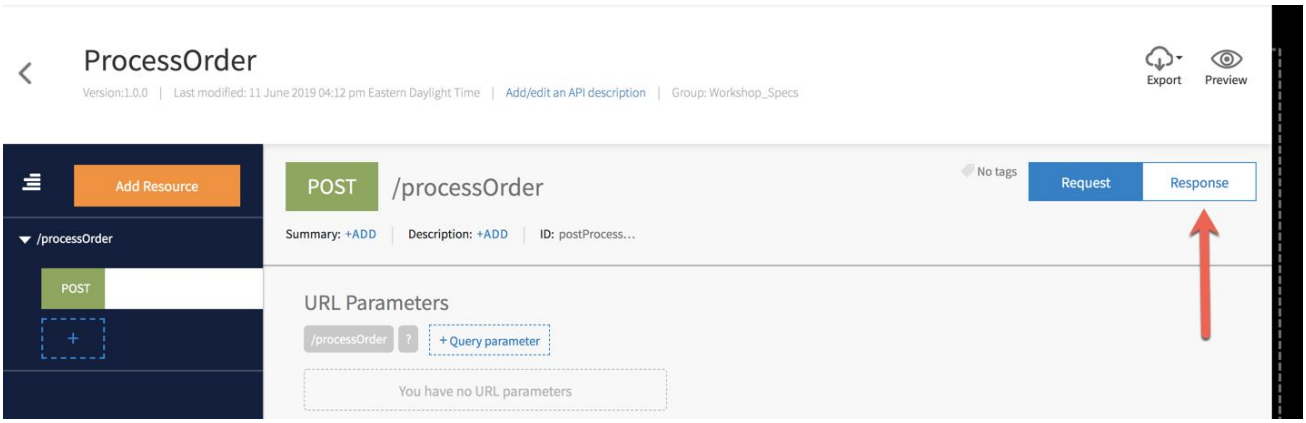
```
1  {
2     "type": "object",
3     "properties": {
4        "CustomerID": {
5           "type": "string",
6           "default": "2"
7        },
8        "ProductID": {
9           "type": "string",
10          "default": "1"
11       },
12       "Qty": {
13          "type": "string",
14          "default": "2"
15       }
16    }
17 }
18
```

Cancel Save

Configuring the Resource's POST Response

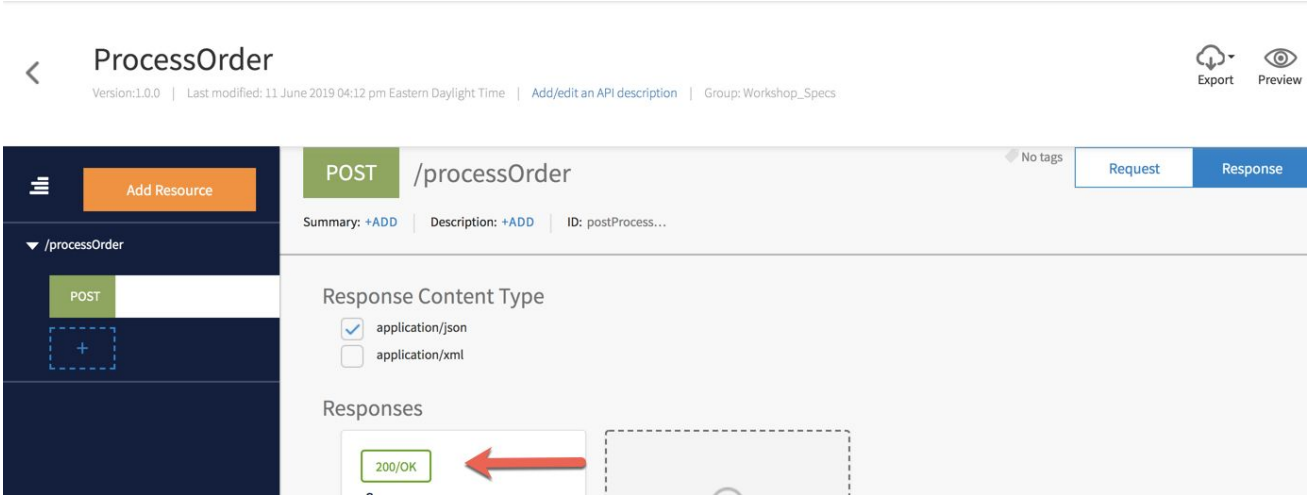
Next we will configure the POST method's response to define from an API specification perspective what the user should be receiving as a response when invoking the POST operation of the 'processOrder' resource.

First make sure you highlight the "Response" button:

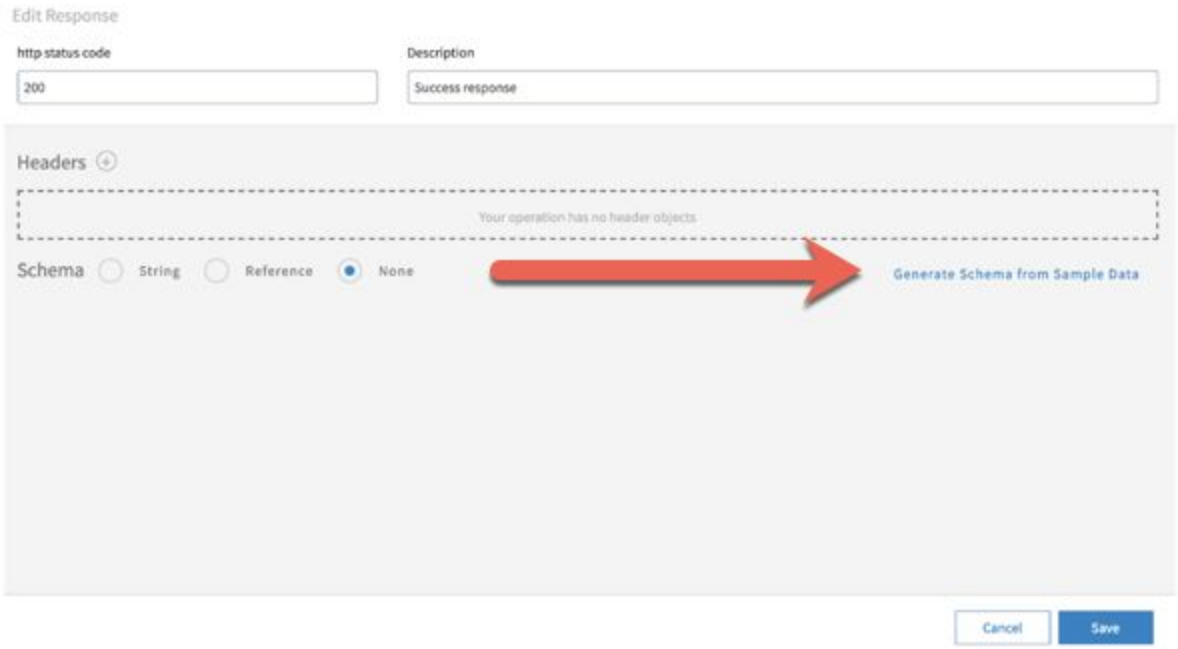


As you can see below TCI automatically created the 200/OK Success Response, and you can now specify the schema for the response.

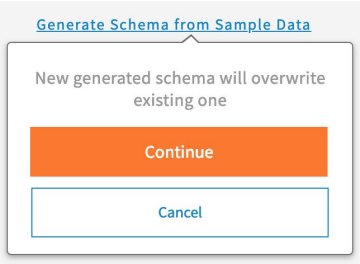
Click on the pre-generated 200/OK response.



Next click on "Generate Schema from Sample Data" (note that it can also be created manually) :



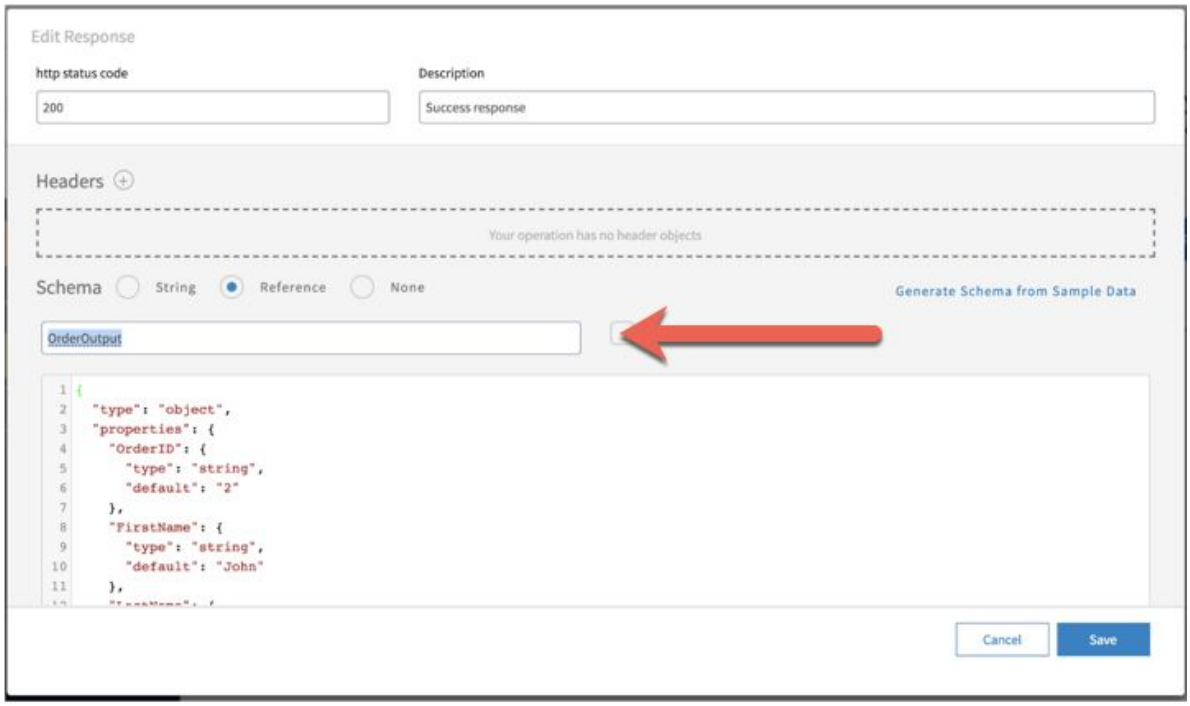
You should get a message as illustrated below, simply select “Continue” :



This time we will use a sample payload rather than a full schema. Copy the sample data below, and click on Generate Schema.

```
{
  "City": "Anywhere",
  "Company": "ACME Corporation",
  "Country": "USA",
  "FirstName": "John",
  "LastName": "Smith",
  "OrderID": "71",
  "ProductName": "MacBook Pro",
  "State": "NY",
  "StreetAddress": "234 Main Street",
  "Zip": "11111"
}
```

As we did earlier for the request, make sure to name the generated schema for reusability (ex. OrderOutput):

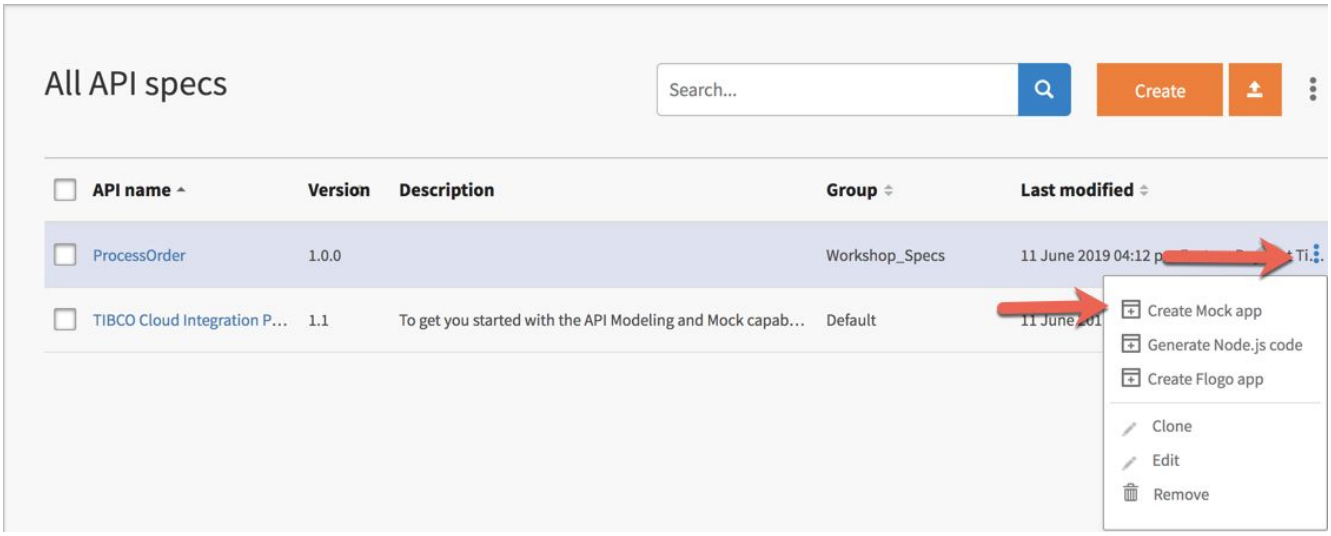


Creating an Application from the API Specification

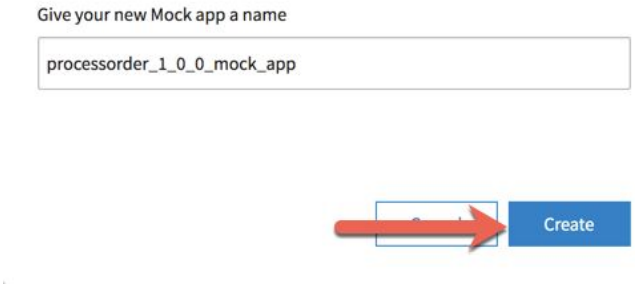
Once the API Specification is created, we can create an application to implement it. In this workshop we will create a Mock Application to allow our developers to develop against our API while we implement the full back end services using a Web Based editor to create a Flogo Application.

Mock Application

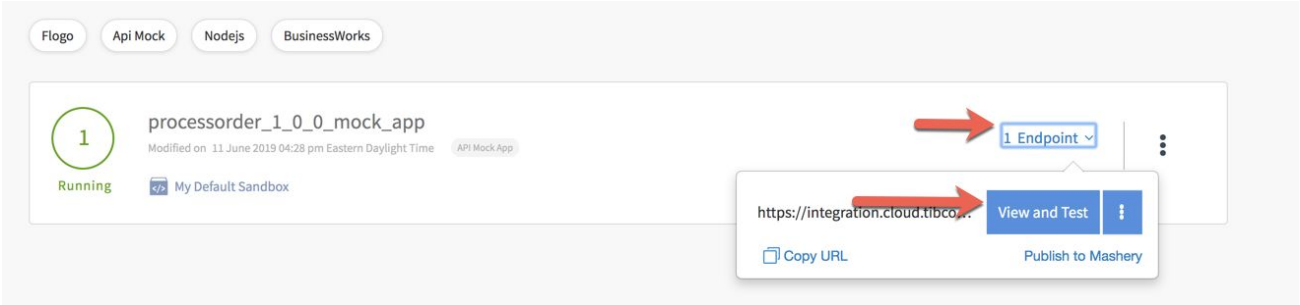
In order to create our Mock Application we will simply click on the 3 bars to the right of the API Specification and select Create Mock Application as shown below:



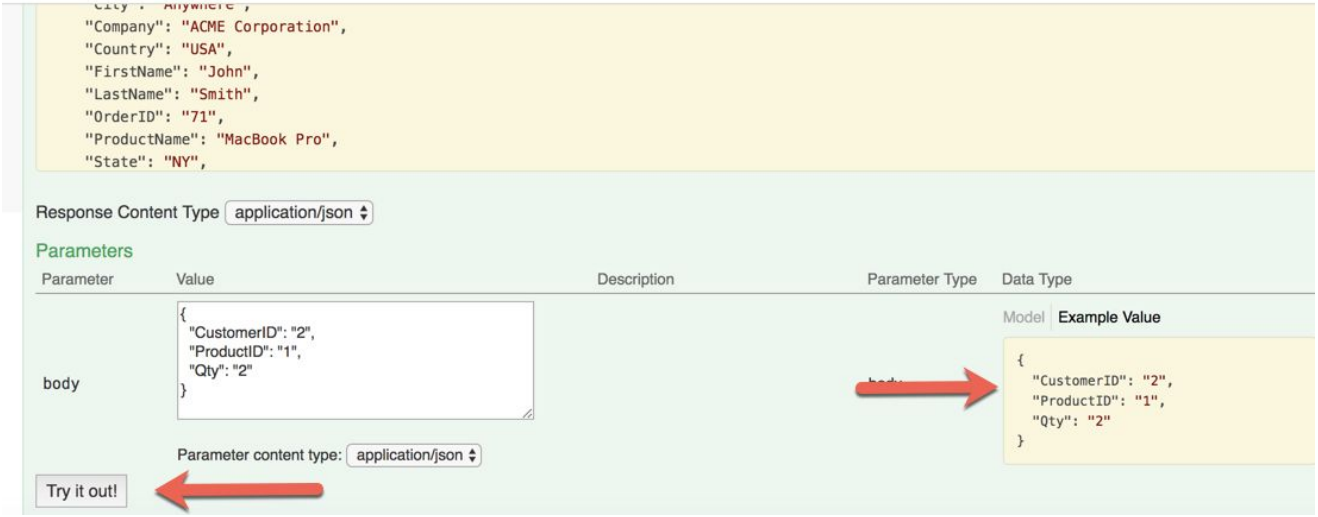
Accept the defaults as shown below and click create:



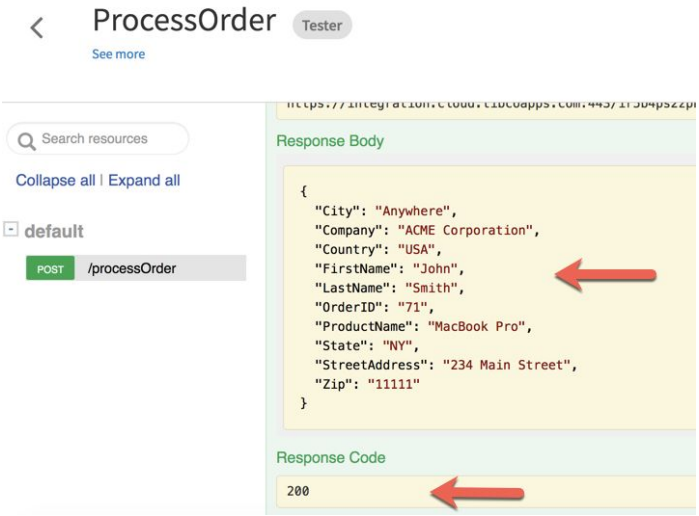
Once the application has been created and is running you can click on the endpoint and test it as shown below:



Scroll down to the sample payload and click try it out:



Notice the successful return code and sample data. Your Mock app is ready for testing by your developers:

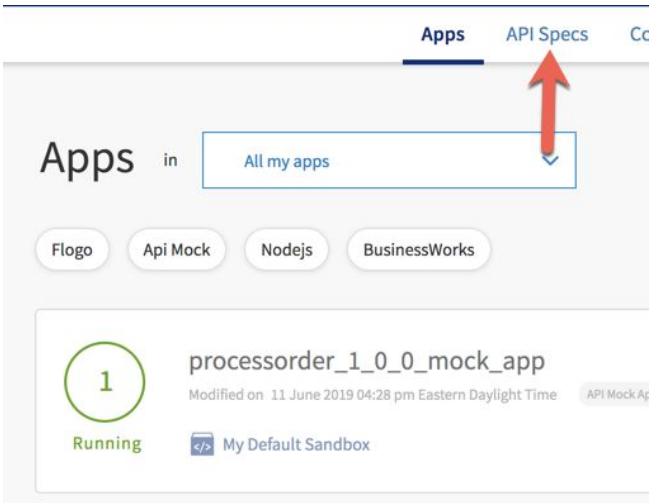


STOP AND WAIT FOR THE INSTRUCTOR BEFORE PROCEEDING

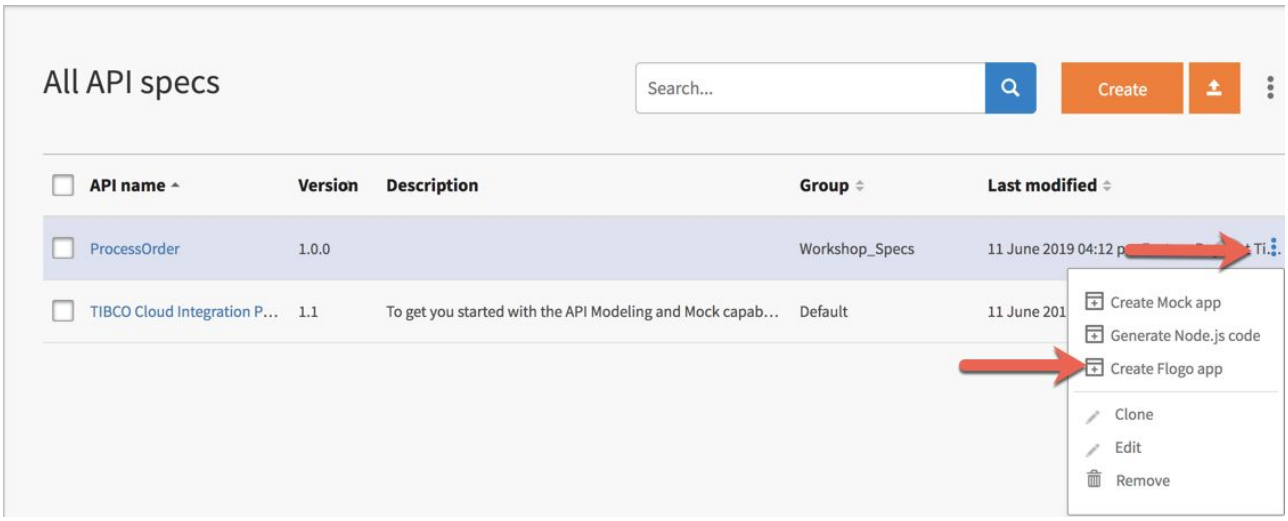
Creating a Web Based Flogo App

Now that we have completed our Mock Application and our developers are coding and testing against it, it is now time to complete the back-end application. In order to do that we will create a web based Flogo Application.

In order to return to the API Specifications Page click on API Specs as shown below:

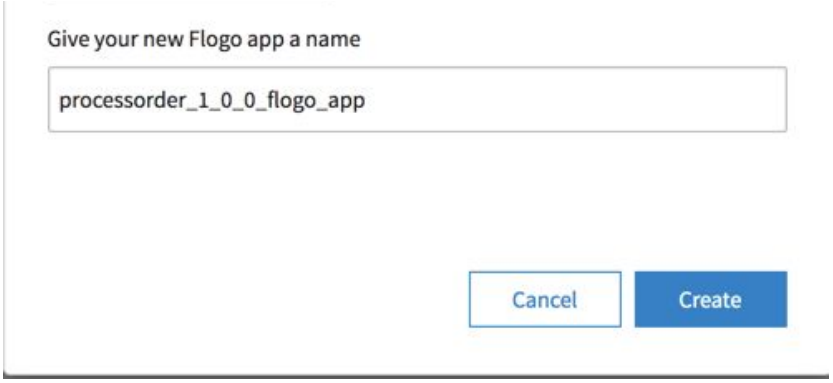


From the main API Specs page, under the group you created, you should see your API specification. When you highlight your application you will see three vertical dots on the right, you can click on the dots to select "Create Flogo app", as shown below .



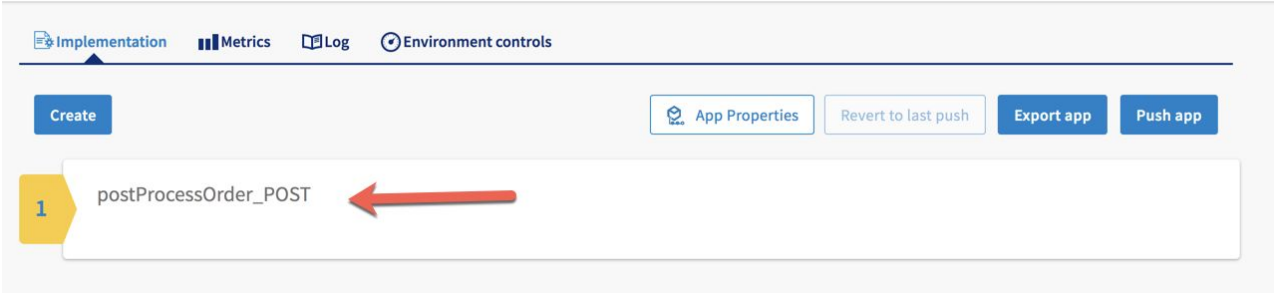
For this workshop we will use the Default information provided. You can also rename your application, or keep the name that was automatically generated.

Finally click on “Create” to automatically generate a Flogo app for your API specification.

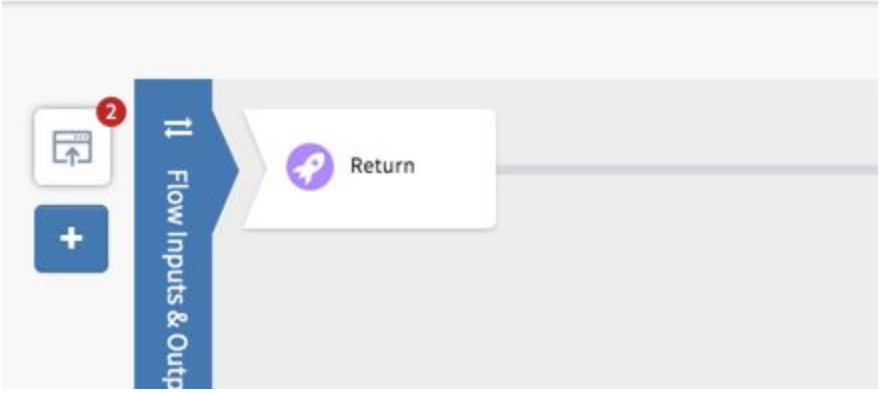


As illustrated below, TCI automatically created a flow for the POST method you designed. We can now configure this flow, and add additional steps as required.

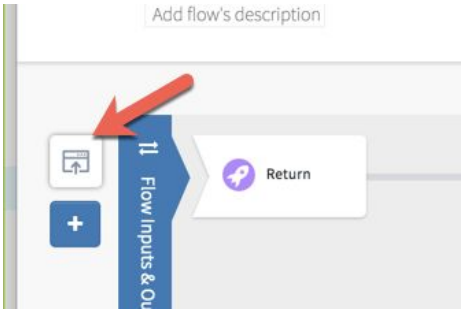
Click on the flow to edit it :



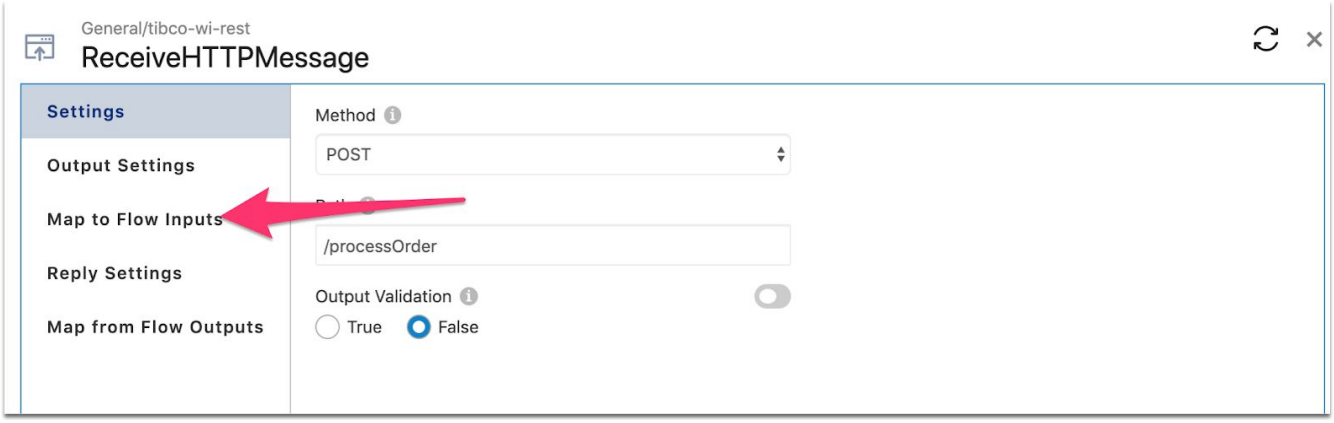
See a trigger and an activity have been created for you:



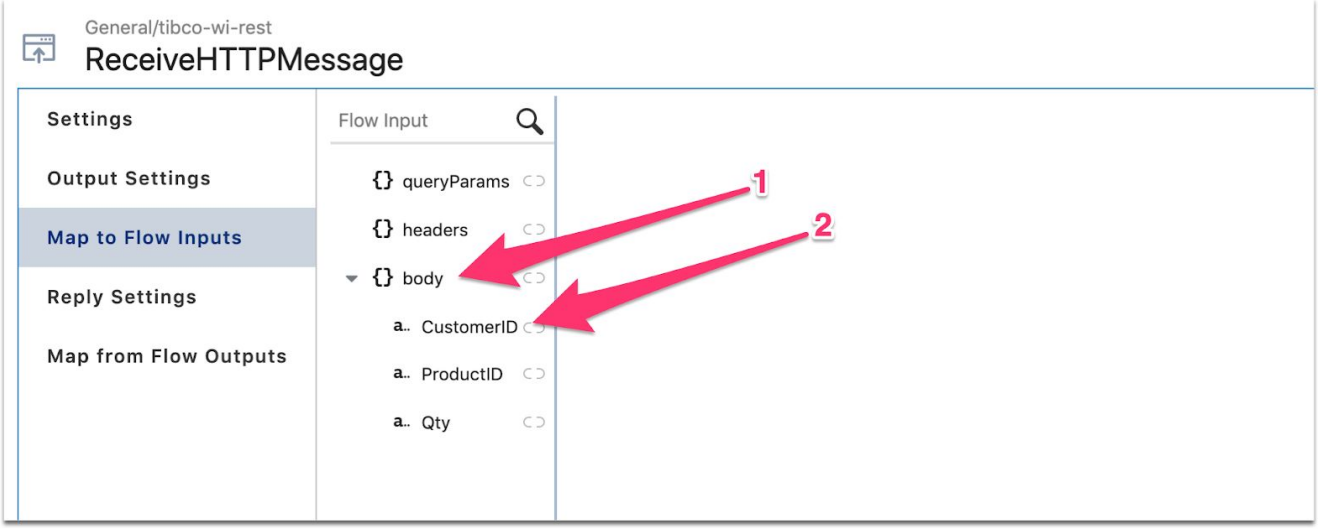
We will now need to map the inputs to the outputs. First, Click on the 'Trigger' Icon as shown below:



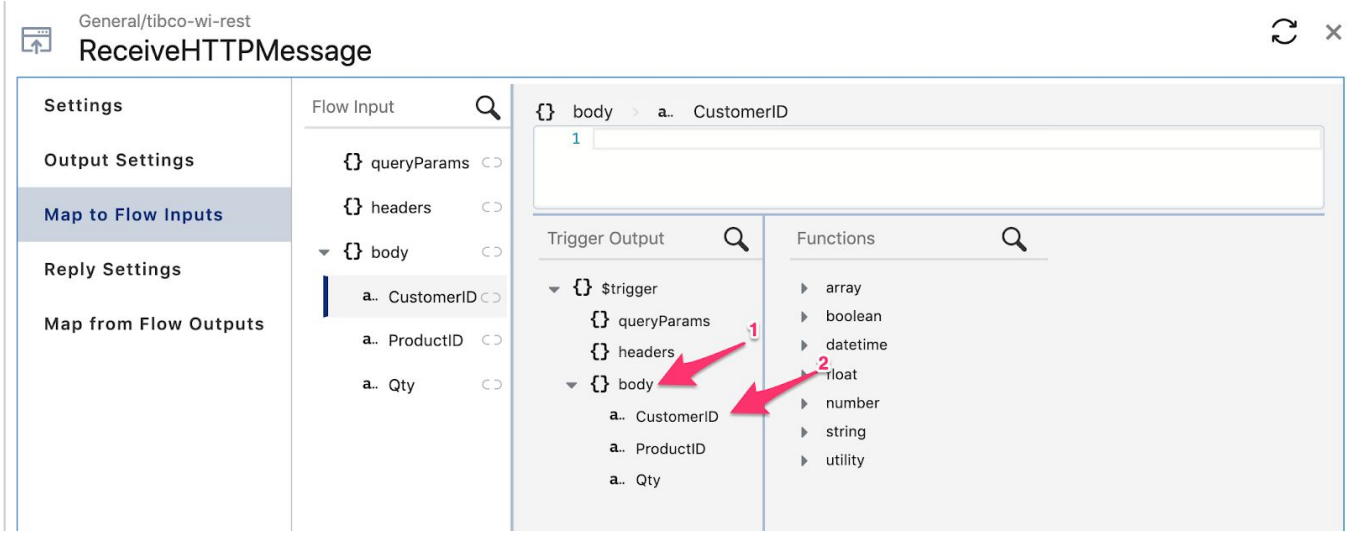
The 'ReceiveHTTPMessage' window opens up and select the 'Map to Flow Inputs'



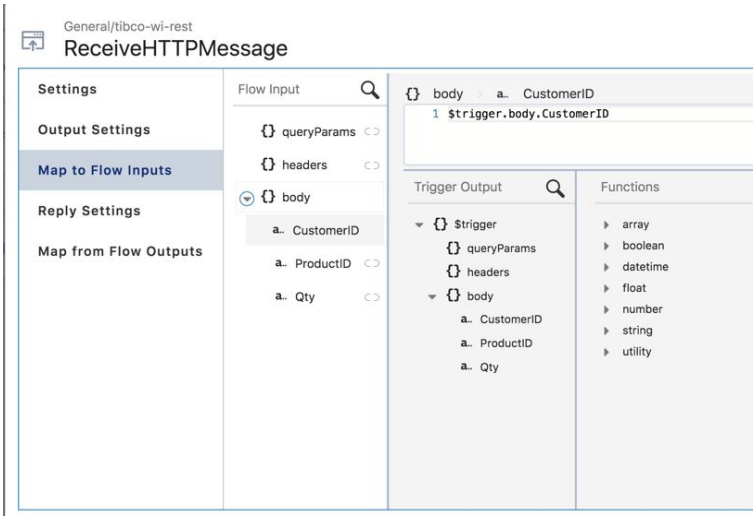
Once 'Map to Flow Inputs' is selected, click on 'body' and then 'CustomerID' in the **'Flow Input'** section



In the next step, we will being the mapping by clicking on the **'Trigger Output'** section

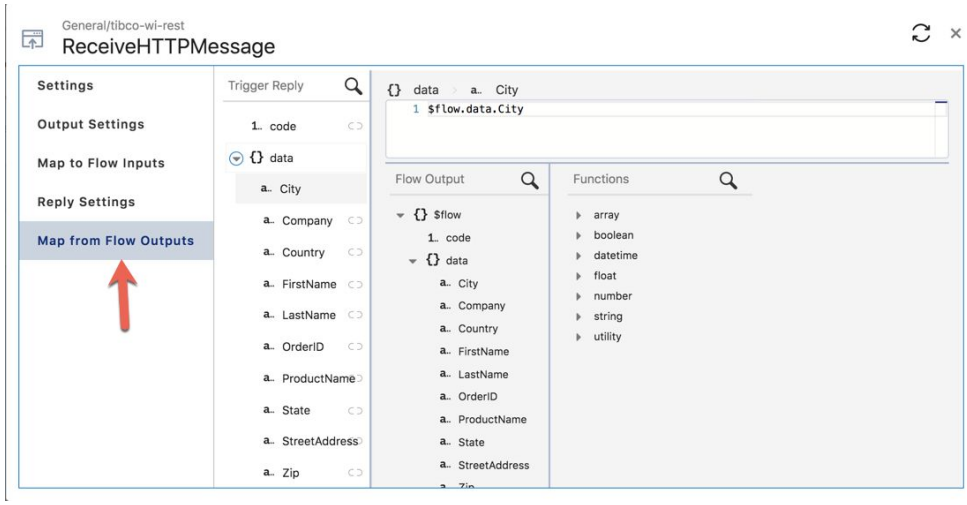


Note that you can simply click on the elements or type them out as shown below:



CustomerID: $trigger.body.CustomerID
ProductID: $trigger.body.ProductID
Qty: $trigger.body.Qty

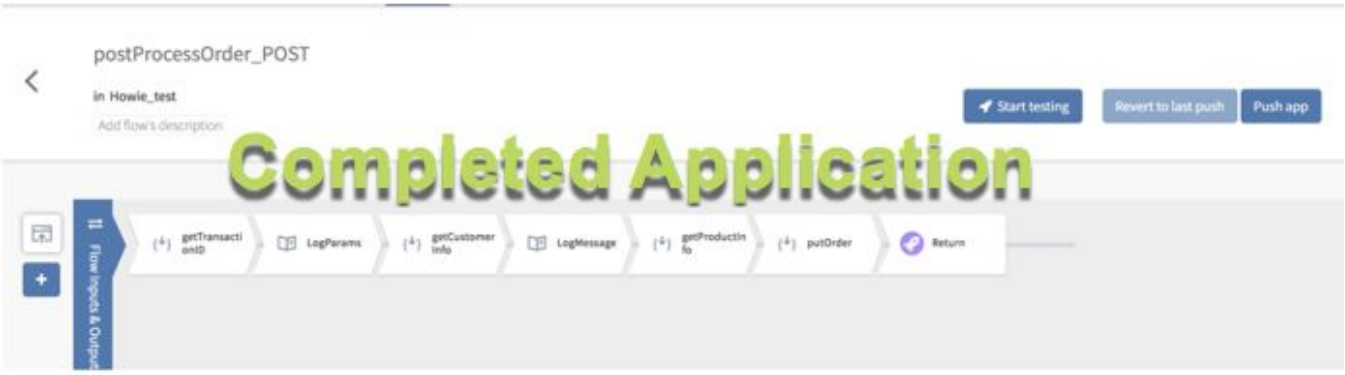
Lastly we will map the following to Map to Flow Outputs. Note you can simply click on the elements or type them out as shown below:



$flow.data.City
$flow.data.Company
$flow.data.Country
$flow.data.FirstName
$flow.data.LastName
$flow.data.OrderID
$flow.data.ProductName
$flow.data.State
$flow.data.StreetAddress
$flow.data.Zip

You can use a simple click and drag to move the activities within the flow, this will allow you to add new activities in order to build a complete end-to-end process. For this workshop we will be calling existing RESTful API to get a new transaction ID, extract customer and product information, log a message into the application log, and implement the response sent back to the user.

When we are finished the completed process orchestration will look similar to the one below.

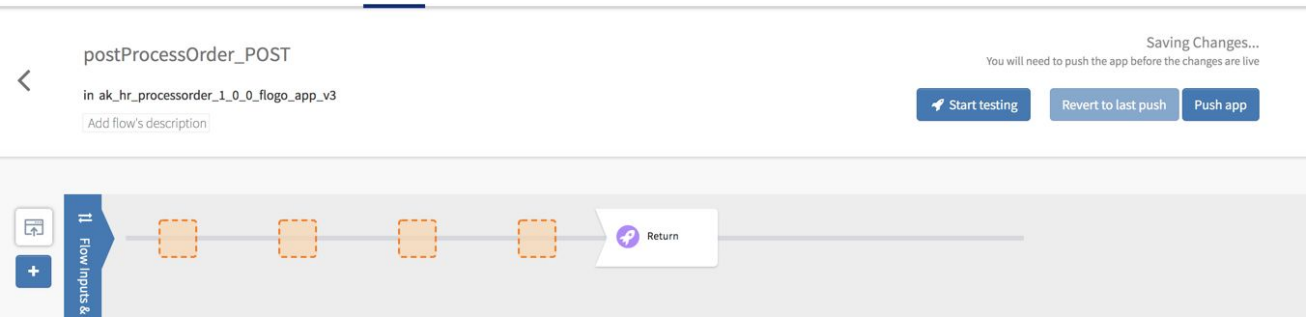


Let's **start by dragging and moving the 'Return' activity** out by 4 or 5 spaces to allow inserts of the new activities.

Adding a getTransactionID Activity

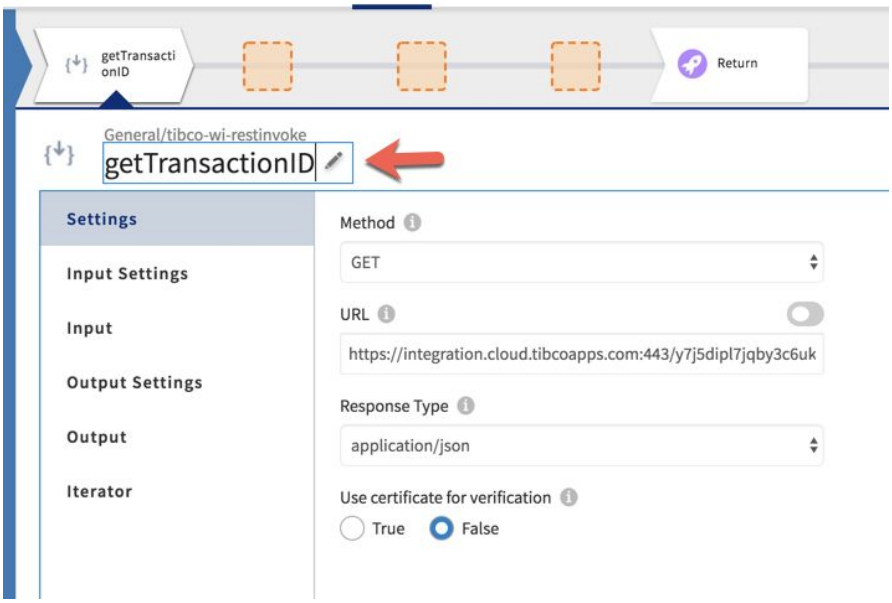
Now, let's start configuring the process API to achieve an orchestration of services

Click on the ReplyToHTTPMessage activity, and drag it a few of places to the right as illustrated below :



Note: Everything you do here is automatically saved.

Then click on the first orange box, find the "General" options, and select the 'Invoke REST Service' activity.
Also rename that activity as "getTransactionID" by clicking on the pencil icon:



Ensure the following in the Configuration tab:

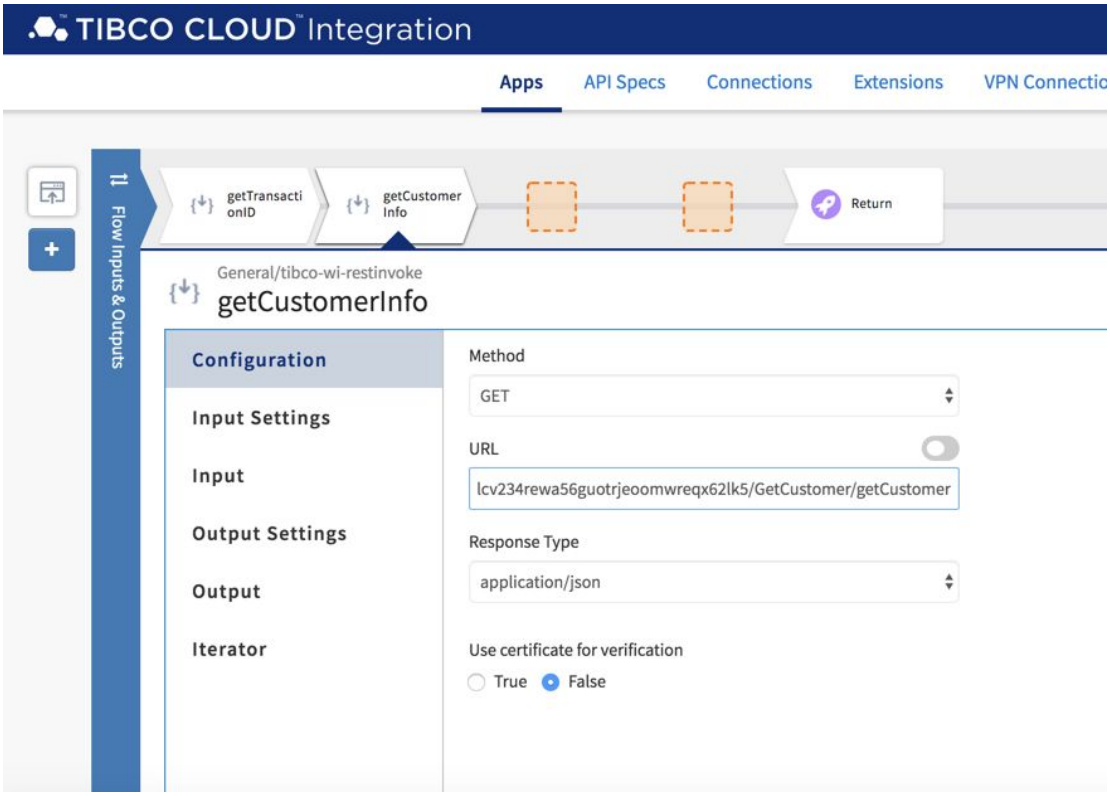
1. The selected method should be GET
2. In the URL field, provide the URL below of the System API that returns a TransactionID.

https://integration.cloud.tibcoapps.com:443/y7j5dipl7jqby3c6ukfjhzdh536sdmmi/GetTransactionID/getTransactionID

Next under the Output Settings tab, we will copy the sample response schema below. This will allow us to map the data from the response to other activities (i.e in this case we will receive a Transaction ID).

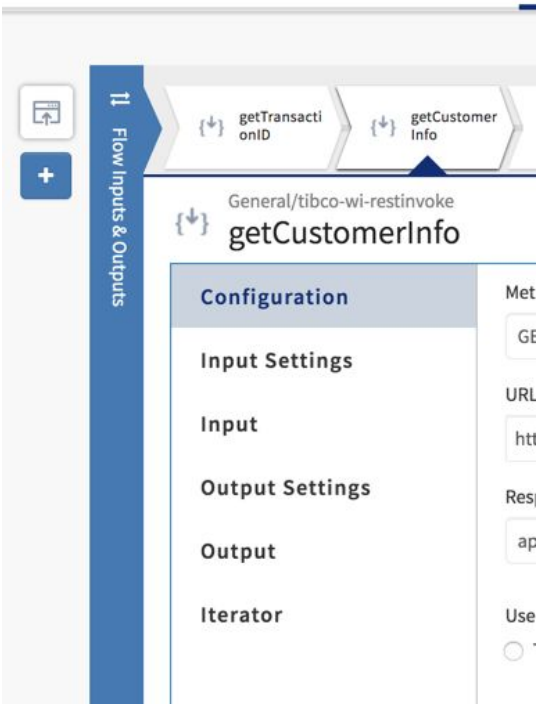
```
{
   "type": "object",
   "properties": {
      "transactionID": {
         "type": "string"
      }
   }
}
```

Adding a getCustomerInfo Activity

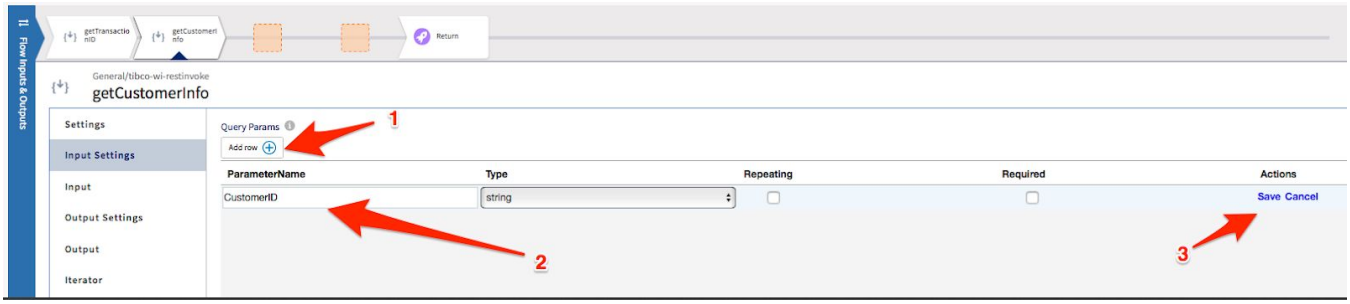


Repeat the process to add a “getCustomerInfo” activity, also as “Invoke REST Service”.

1. Ensure the method is set to GET
2. In the URL field enter the URL below:
   https://integration.cloud.tibcoapps.com:443/lcv234rewa56guotrjeoomwreqx62lk5/GetCustomer/getCustomer

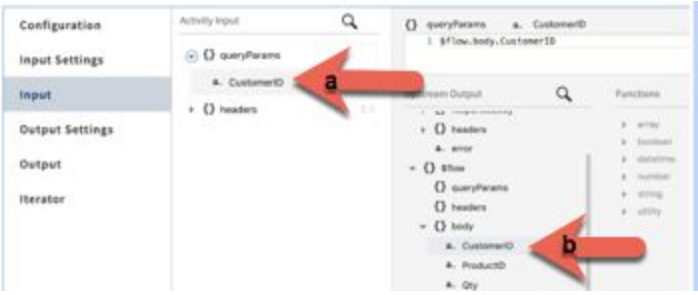


This REST service call requires a parameter, which will customer ID received from the request. To specify a parameter, go to the InputSettings tab, click on **"Add row" under Query Params**, and create a "**CustomerID**" parameter as shown below, then click **Save**.



We will then map this parameter to the customer ID we received in the HTTP request. To do this, access the Input tab, and then :

A. Click on CustomerID under QueryParams
B. Click on CustomerID under Flow-->Body

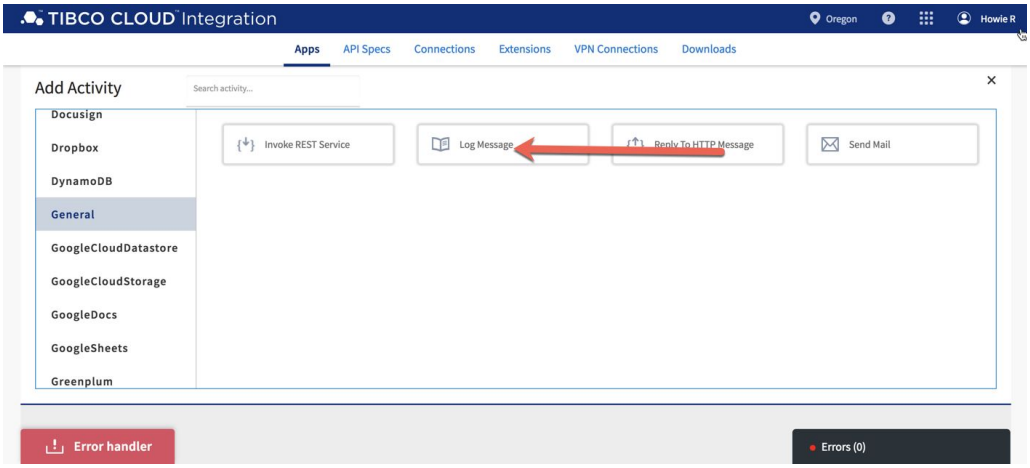


Finally under the Output Settings tab, we will copy the sample response schema below:

```
{
   "type":"object",
   "properties":{
      "Address":{
         "type":"string"
      },
      "City":{
         "type":"string"
      },
      "Company":{
         "type":"string"
      },
      "Country":{
         "type":"string"
      },
      "CustomerID":{
         "type":"string"
      },
      "FirstName":{
         "type":"string"
      },
      "LastName":{
         "type":"string"
      },
      "State":{
         "type":"string"
      },
      "Zip":{
         "type":"string"
      }
   }
}
```

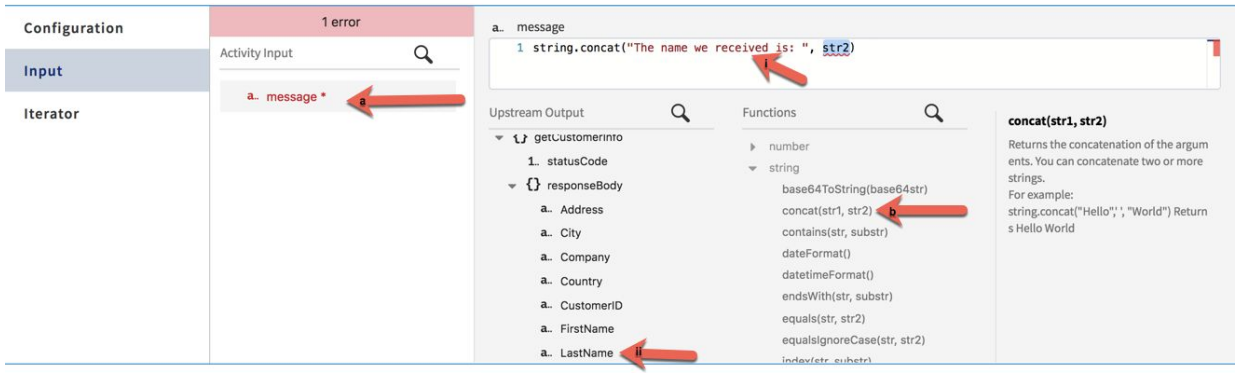
Adding a LogMessage Activity

Add a new activity to the flow, this time using the “Log Message” activity type under the General section. This will allow you to create an entry on the application log when running the application.



Now configure the log as follows:

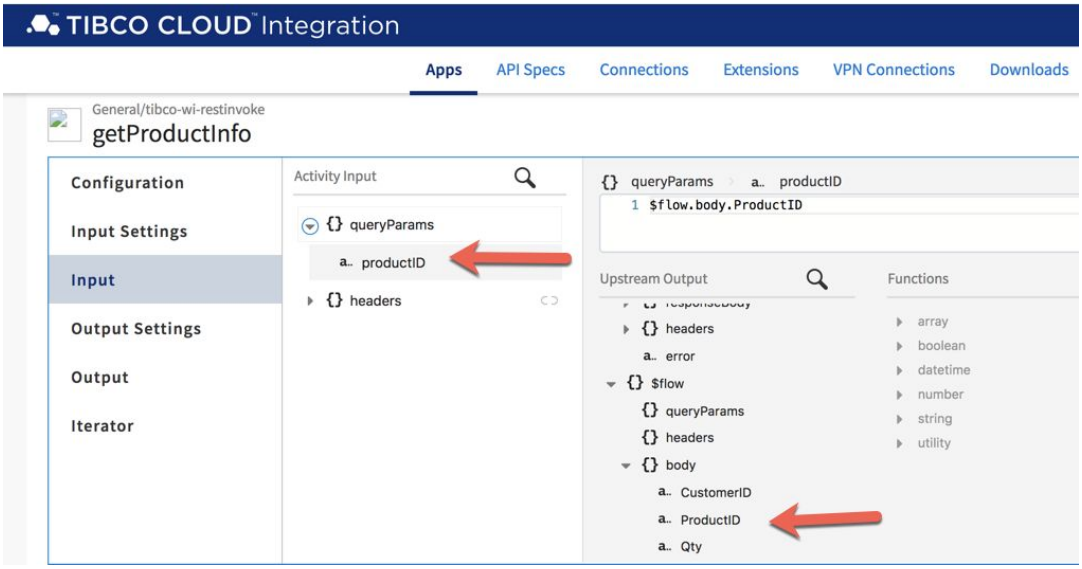
1. Access the Input tab
2. We will be using a function to create the content of the log. There are many built-in functions in Flogo including number, string, date, and other operations. In this case will be using a concat function to combine two strings. **Under “Functions”** access the “string” functions, and select **concat(str1, str2)**.
    a. Replace str1 with “The name we received is: “
    b. Highlight str2 and click on getCustomerInfo-->responseBody-->LastName



Adding a getProductInfo Activity

Add another InvokeRESTService activity which will be used to extract product information using the ProductID parameter in the request. Make sure that :

1. Method is set to GET
2. The URL is: https://integration.cloud.tibcoapps.com:443/ojvrinaw3vuvknh5mjcebeoqtpw5tvex/GetProductInfo/getProductInfo
3. Add a productID query parameter
4. Map the productID from the request to the productID parameter in the Input tab :
    a. Under queryParams click on productID
    b. On the right panel click on $flow-->body-->productID



Finally under the Output Settings tab, we will copy the sample response schema below.

```
{
   "type":"object",
   "properties":{
      "productID":{
         "type":"string"
      },
      "productDesc":{
         "type":"string"
      },
      "price":{
         "type":"string"
      },
      "availableQty":{
         "type":"string"
```

```
            },
            "productName":{
               "type":"string"
            }
         }
      }
```

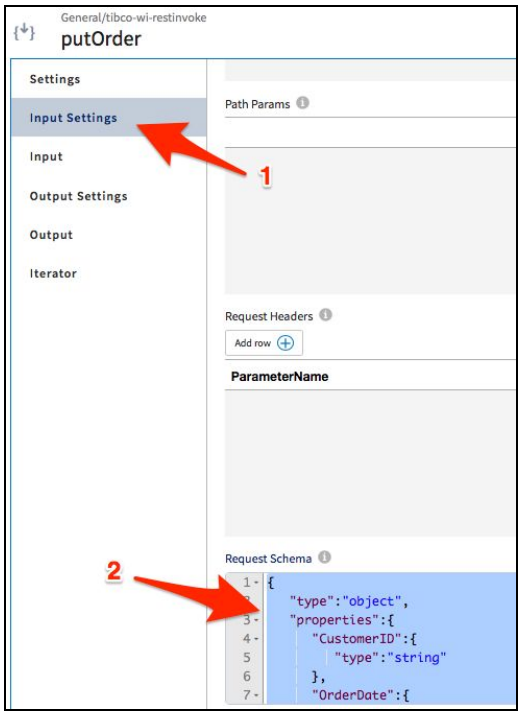
Adding a putOrder Activity

Next we will create one final activity to push the order to the Order Fulfillment system. This will be done using a POST operation in a InvokeRESTService activity type.

A. Make sure that the Method type is POST (not a GET like our previous calls)
B. Use the following URL :
   https://integration.cloud.tibcoapps.com:443/iejak6hhxpp2wkug5g62wv5hqpd4jyjv/GetOrder/putOrder

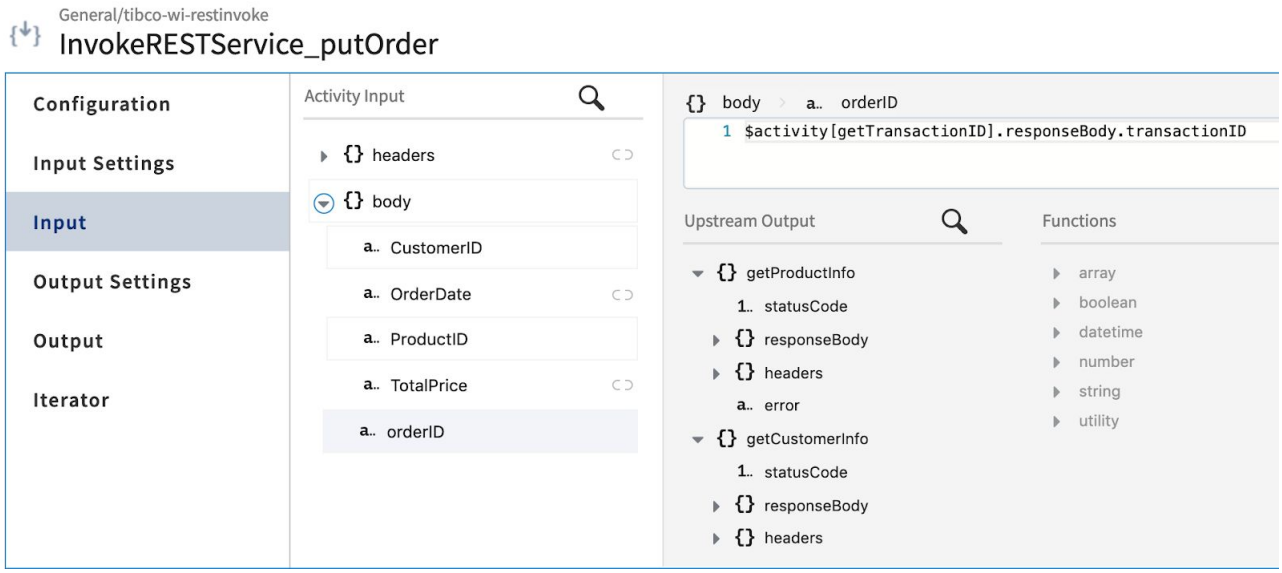
Next we will need to define the request body. Under the Input Settings tab, copy the sample request schema below in the “Request Schema” setting :

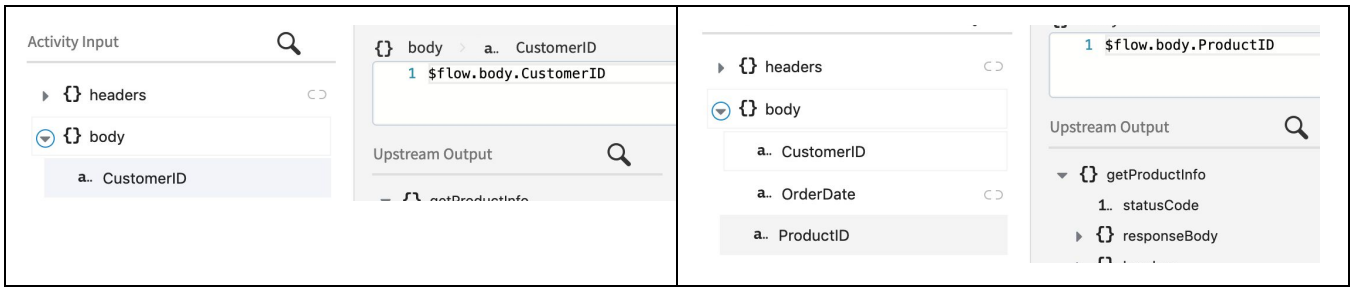
```
{
   "type":"object",
   "properties":{
      "CustomerID":{
         "type":"string"
      },
      "OrderDate":{
         "type":"string"
      },
      "ProductID":{
         "type":"string"
      },
      "TotalPrice":{
         "type":"string"
      },
      "orderID":{
         "type":"string"
      }
   }
}
```



Then we need to map the content of the fields in the request to the data from our process flow. Access the Input tab, extend the “body” section, and map the fields as follows :

- body.CustomerID → $flow.body.CustomerID
- body.ProductID → $flow.body.ProductID
- body.orderID → $activity[getTransactionID].responseBody.transactionID



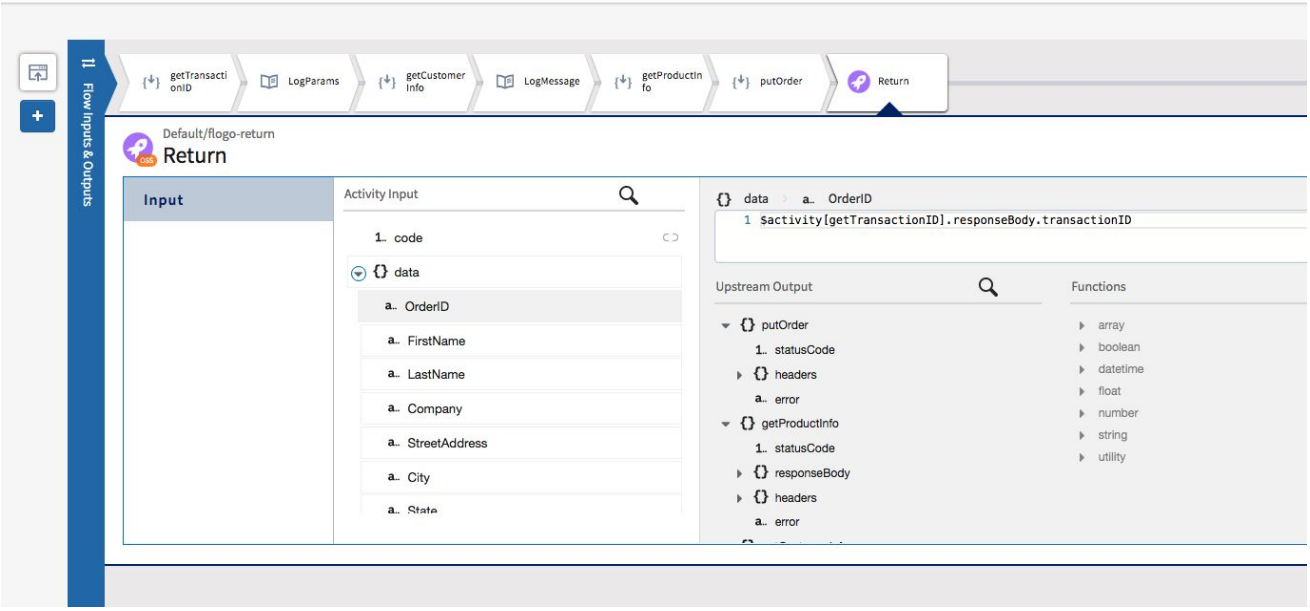


Complete the HTTP Response via the Return activity

The last activity in our process flow is the Return activity, where we will configure what gets returned to the user. We will therefore map the fields in the Return (as defined in our API specification) to the data in our process flow.

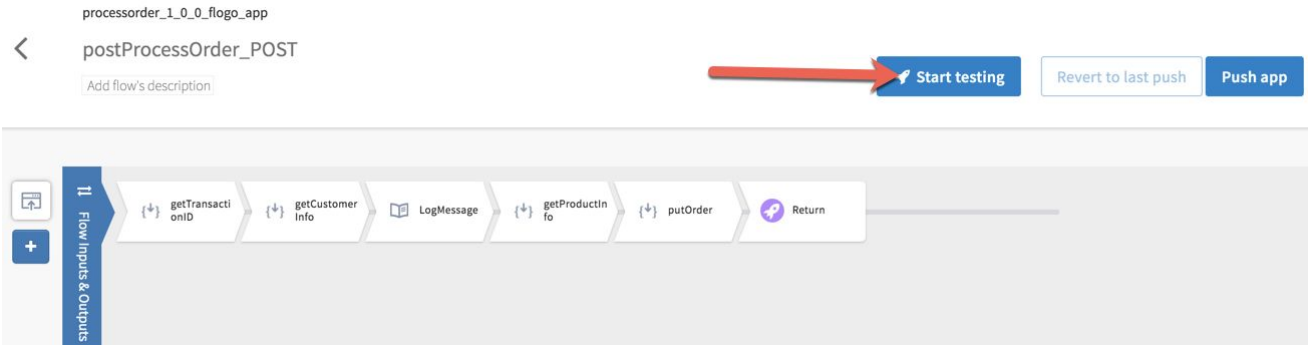
Access the Input tab, extend the “data” section, and map the fields as follows :

- data.OrderID → $activity[getTransactionID].responseBody.transactionID
- data.FirstName → $activity[getCustomerInfo].responseBody.FirstName
- data.LastName → $activity[getCustomerInfo].responseBody.LastName
- data.Company → $activity[getCustomerInfo].responseBody.Company
- data.StreetAddress → $activity[getCustomerInfo].responseBody.Address
- data.City → $activity[getCustomerInfo].responseBody.City
- data.State → $activity[getCustomerInfo].responseBody.State
- data.Zip → $activity[getCustomerInfo].responseBody.Zip
- data.Country → $activity[getCustomerInfo].responseBody.Country
- data.ProductName → $activity[getProductInfo].responseBody.productName

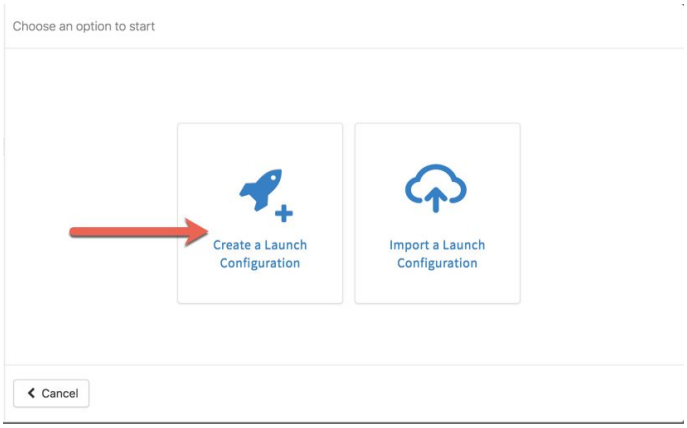


Testing the Flogo App

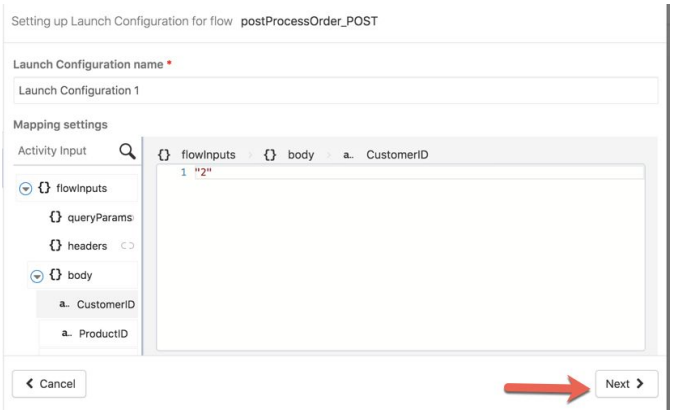
Before we deploy our App we can test it. In order to do that click on the Start testing button as shown below:



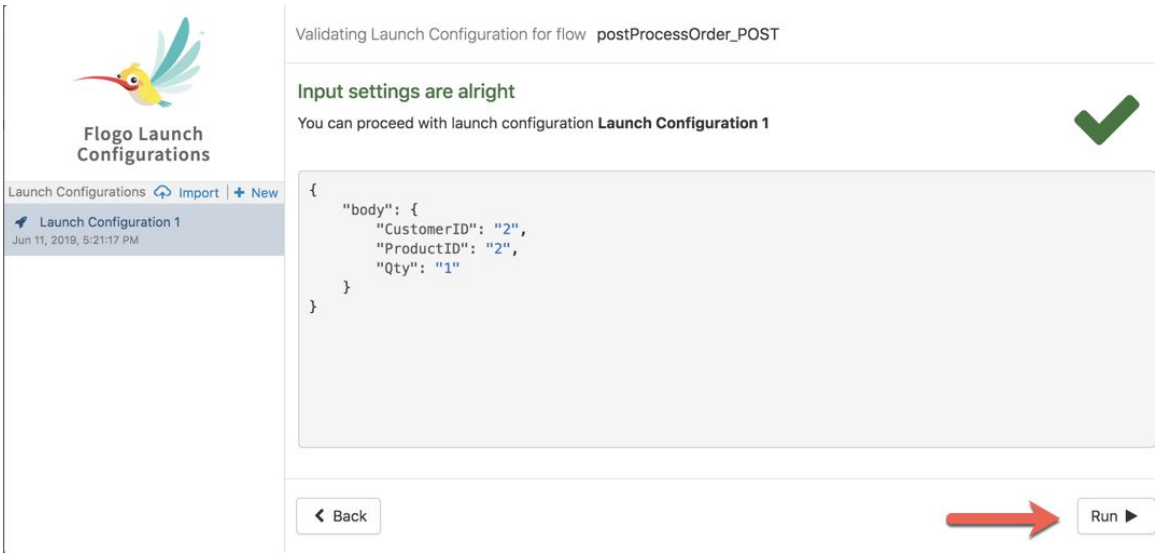
Click on Create a Launch Configuration:



Enter a value of “2” for **CustomerID**, **ProductID**, and **Qty** respectively and click Next:



Finally click Run to start testing:



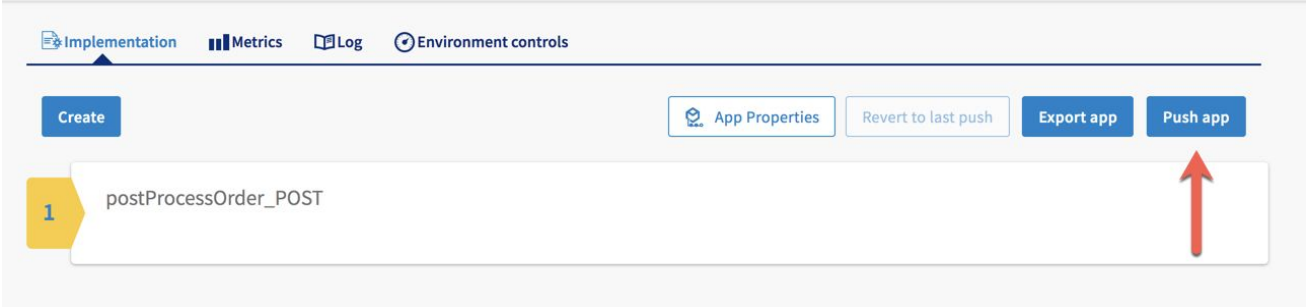
A successful call should show the following at the bottom of your log:

```
2019-06-11T21:41:34.775Z INFO [flogo.flow] - Instance [959d3124a6c2fc4cf0cbea70fe51383c] Done
Flow execution successful
{
"data": {
"City": "Anywhere",
"Company": "ACME Corporation",
"Country": "USA",
"FirstName": "John",
"LastName": "Smith",
"OrderID": "79",
"ProductName": "Magic Mouse",
"State": "NY",
"StreetAddress": "234 Main Street",
"Zip": "11111"
}
}
```

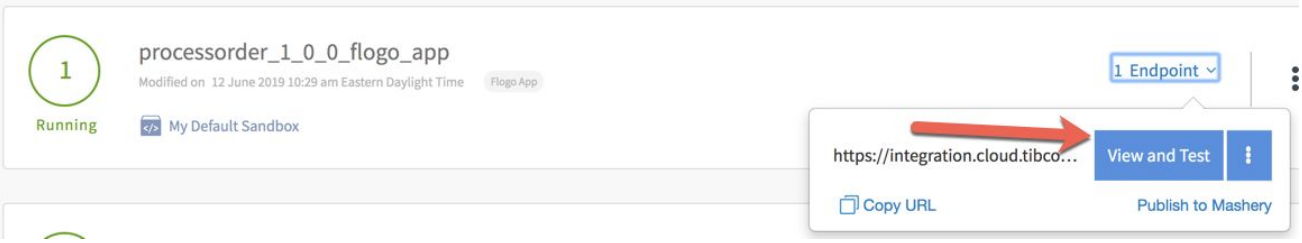
Click **Stop** testing, and **Cancel** to clear the Configuration Tester.

Pushing and Testing the App

Once the configuration is complete, click on the “Push App” button on the upper right corner. This will enable the application, and set it in a running state.



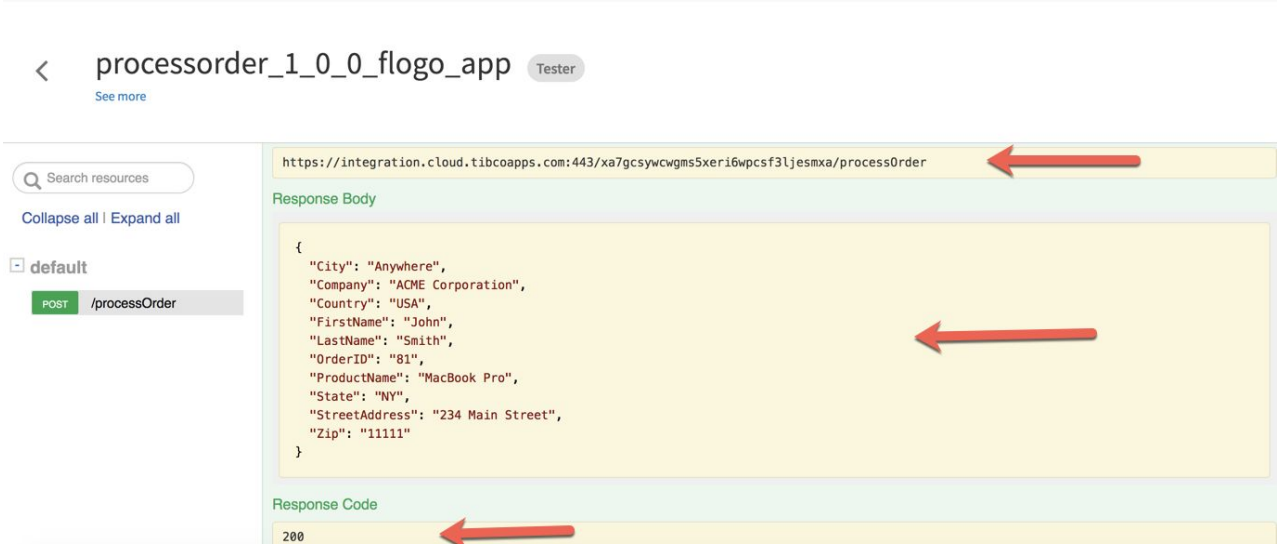
To view and access the API endpoints, click on “1 Endpoint”, and then the “View and Test” button .



TIBCO provides a full Swagger testing framework which is important for documentation and testing. You can now enter a sample JSON sample to test your POST operation, you can use the default values by clicking on the sample payload as shown below.

Now you will see the results and notice 3 important aspects:

1. The actual URL called for this service
2. The return payload from the application
3. The response code provided.



We are now ready to expose the application as an API.

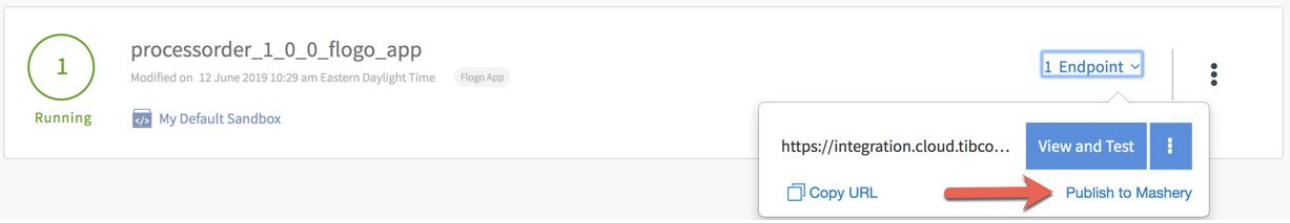
STOP AND WAIT FOR THE INSTRUCTOR BEFORE PROCEEDING

Creating an API from your Application

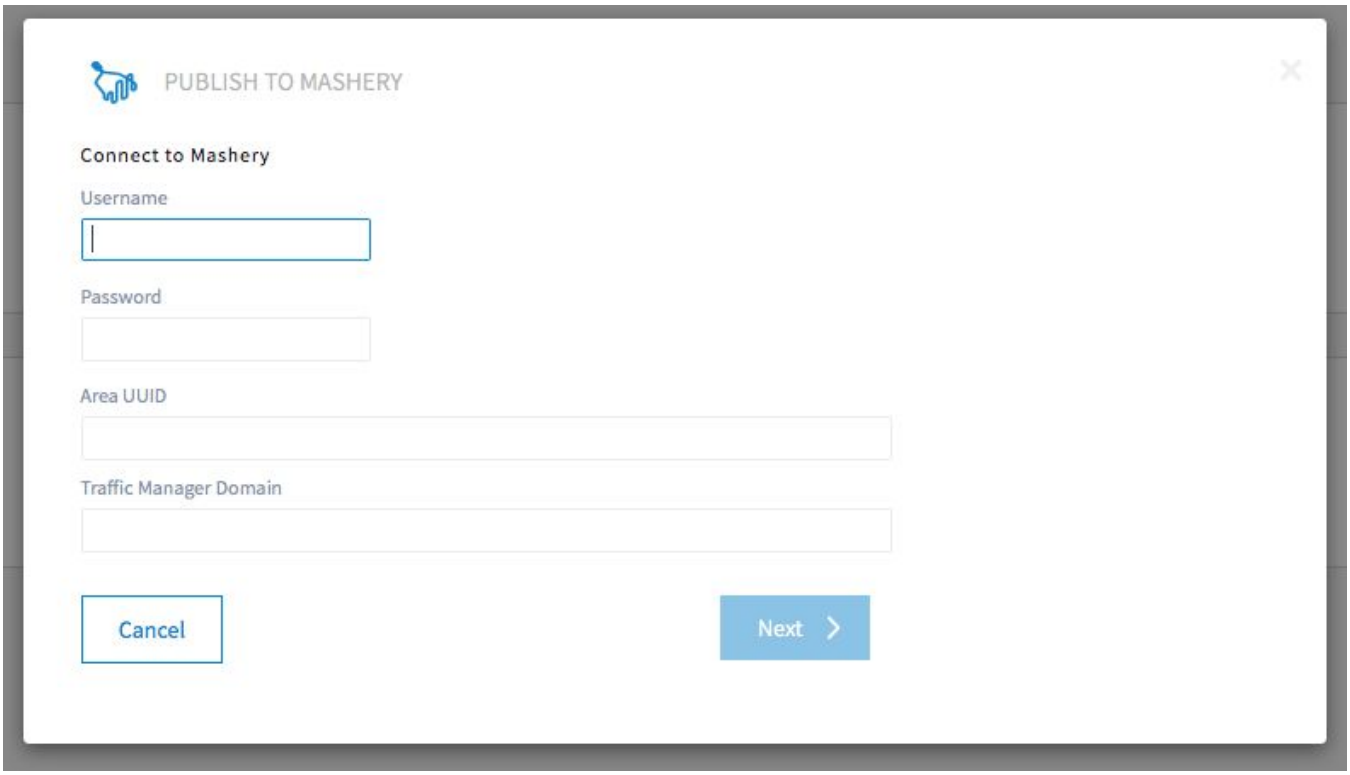
Now that we have created an application to process orders we can now expose this as an API with little effort. We will now push our Flogo App to Mashery which is the TCI API Manager.

Pushing Flogo App as an API

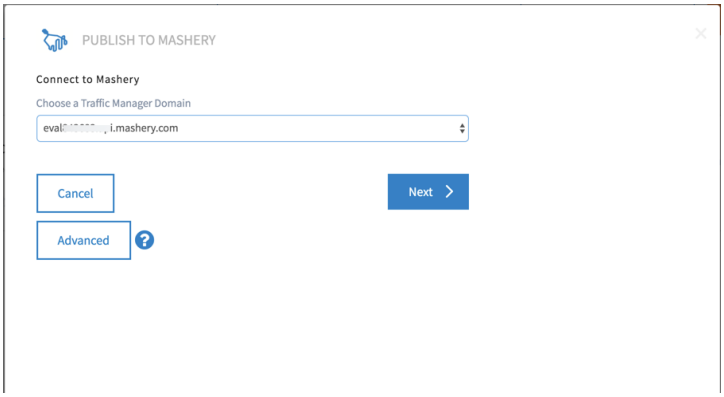
From the Flogo App that was created click on the Publish to Mashery button as shown below:



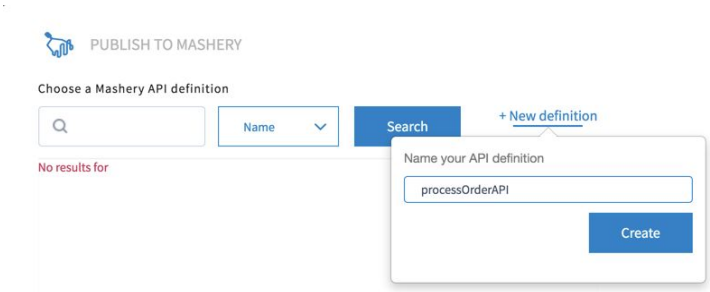
If you get this image, your API Trial is not complete. Please



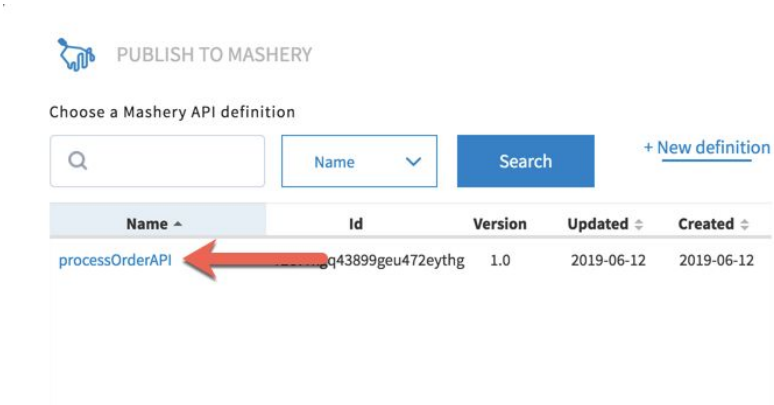
Under the Traffic Manager Domain drop down **select the Eval traffic manager that has been created for your trial** and click Next:



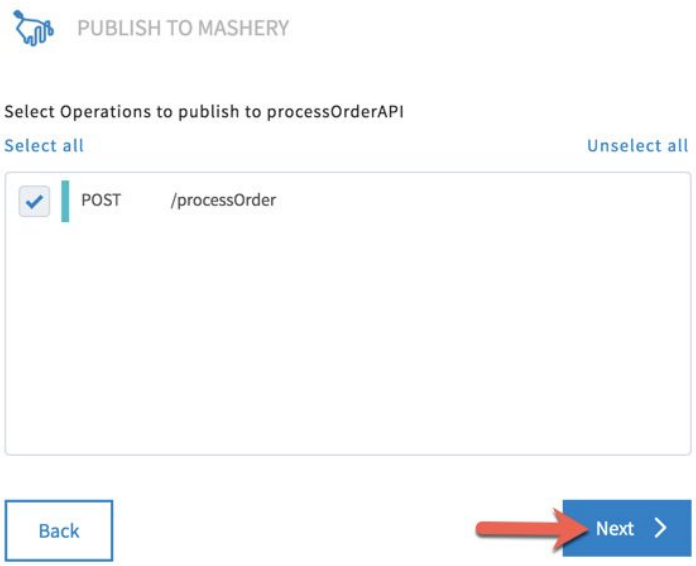
Next click on New Definition. This will create an API Definition for us. Name it processOrderAPI and click Create:



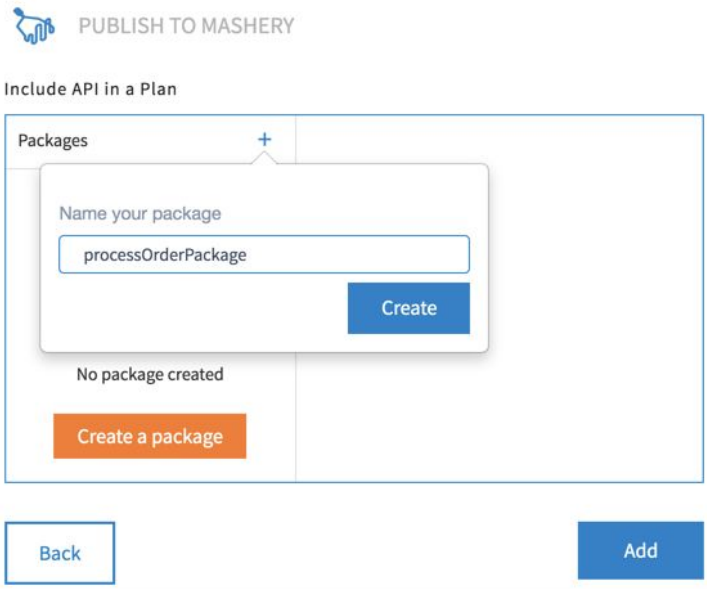
Click on processOrderAPI:



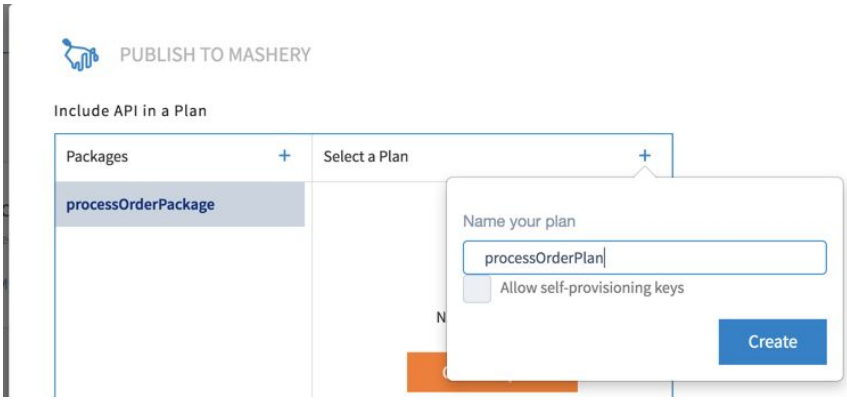
Ensure the processOrder operation is selected and click Next:



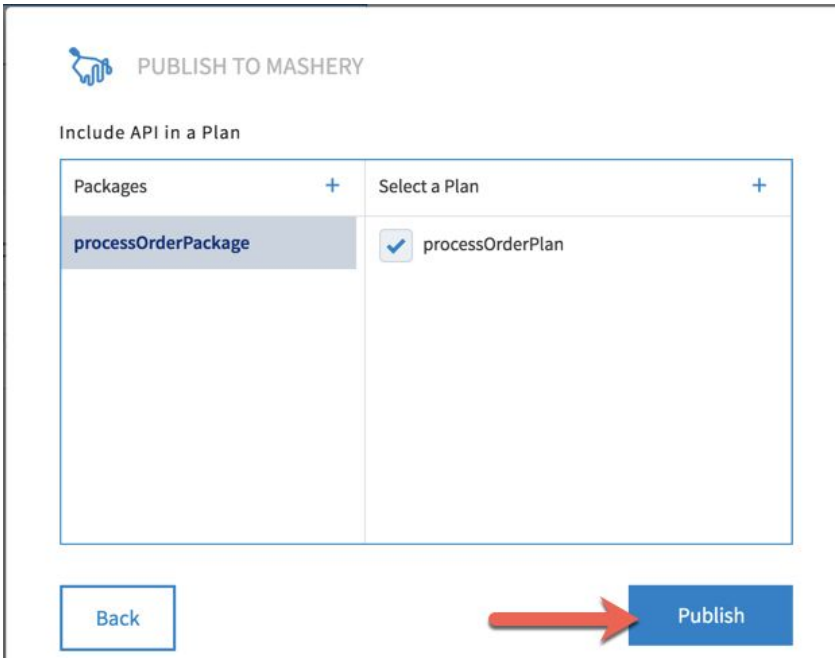
API's are provided through Packages which contain 1 or more plans. We will now create both a package and a plan. First lets create our package by clicking Create a package as shown below and name it processOrderPackage:



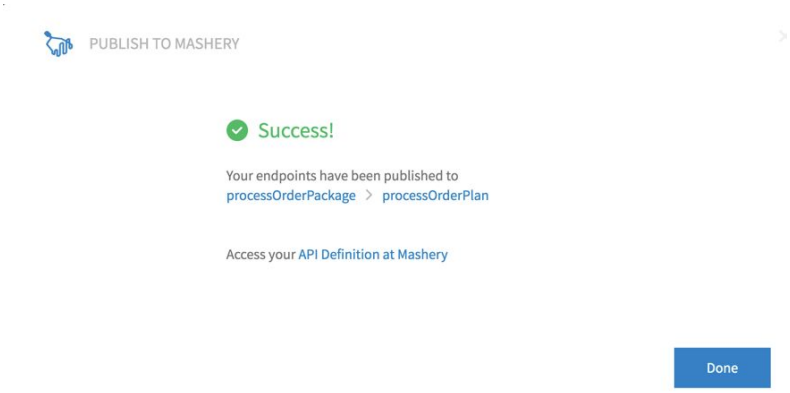
Next we will add a plan as shown. Ensure the Allow self-provisioning keys box is checked and click Create:



Finally click Publish as shown below:



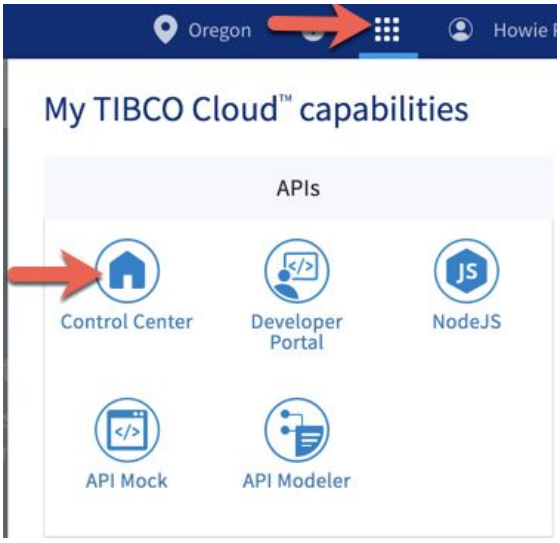
Once the API has been published a message similar to below will be shown:



Click on **'Done'** and exit.

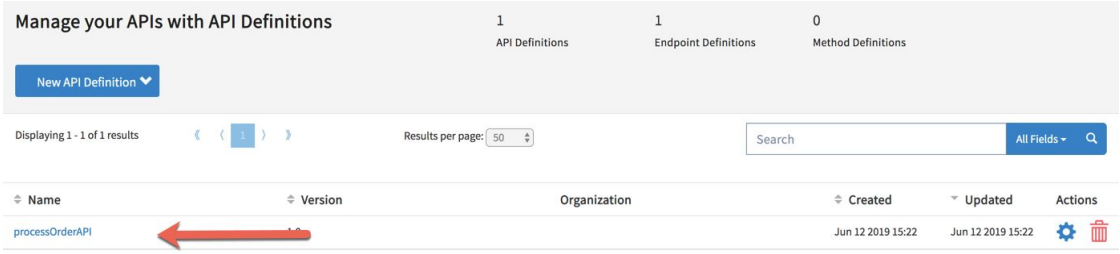
Verifying the API Deployment

Now that we have deployed the API lets take a look at what we did. From the TCI Capabilities Icon select Control Center as shown below:

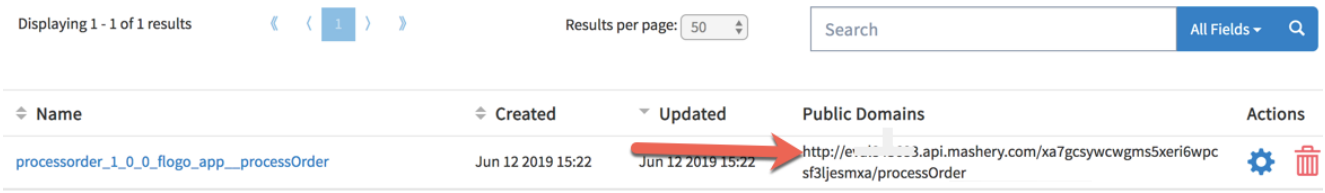


From the Control Center Select Design and then API's from the drop down menu as shown below:

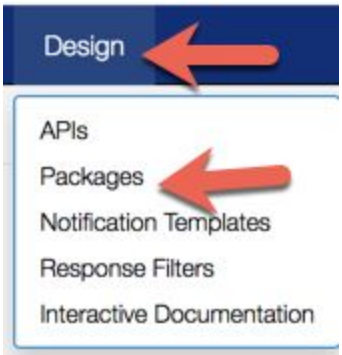
Now we can see the API that we deployed earlier. Click on the processOrderAPI to see details about it:



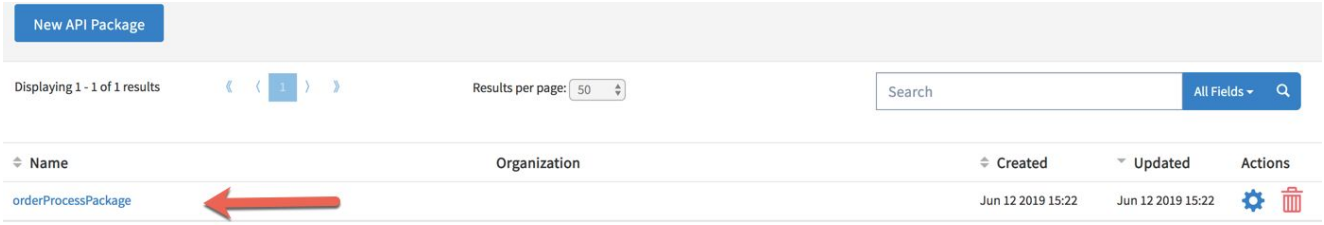
Now we see details about the endpoint and how to access it. This is the public URL that is provided for the API. Copy the public endpoint from your screen and paste it into a text file as we will be using this later:



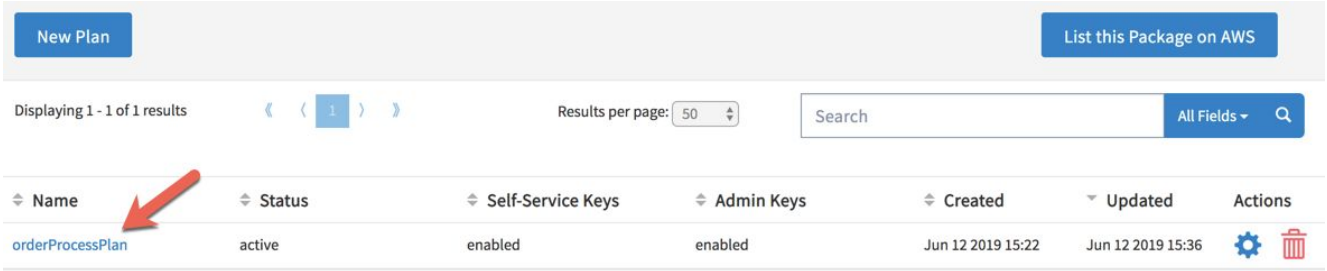
Now click on Packages from the Design menu as shown below to look at the Package that was created:



Click on orderProcessPackage to view the details of the package:



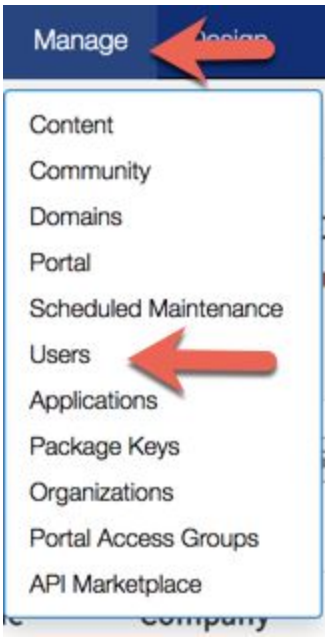
You will now see the plan that was created within the package. Click on the plan and look at the settings within it. Do not make any changes at this point.



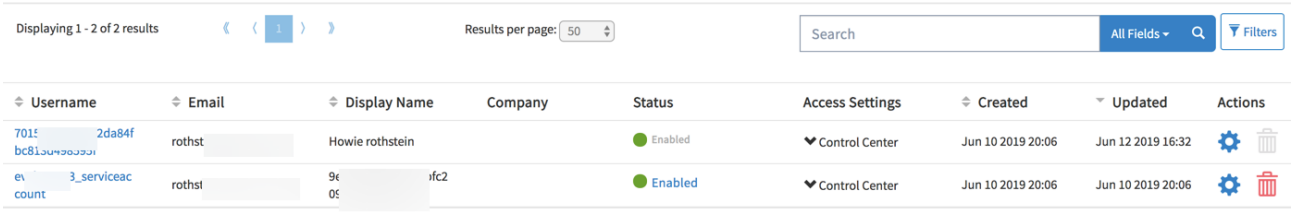
Generating a New Application

While we work with the workshop we will need to know what the Mashery UserID is. This ID is created for you when you sign up for the TCI Trial. In order to find your UserID simply do the following:

From the Control Center Click Manage and then Users as shown below:

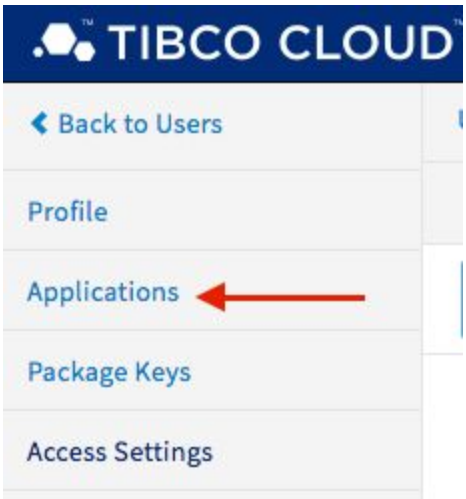


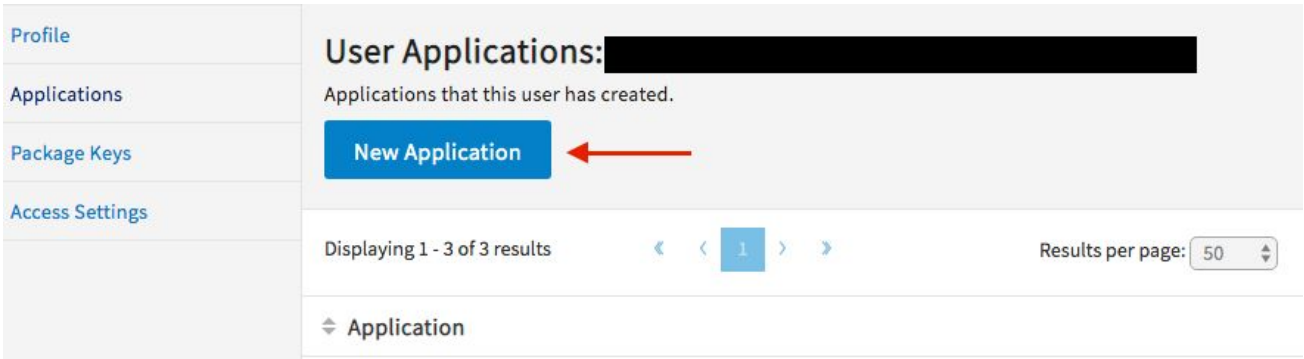
You will then see two UserID's similar to below. One is a normal UserID while the other is a Service Account. Click on the link for the UserID (NOT the Service Account). See the sample below:



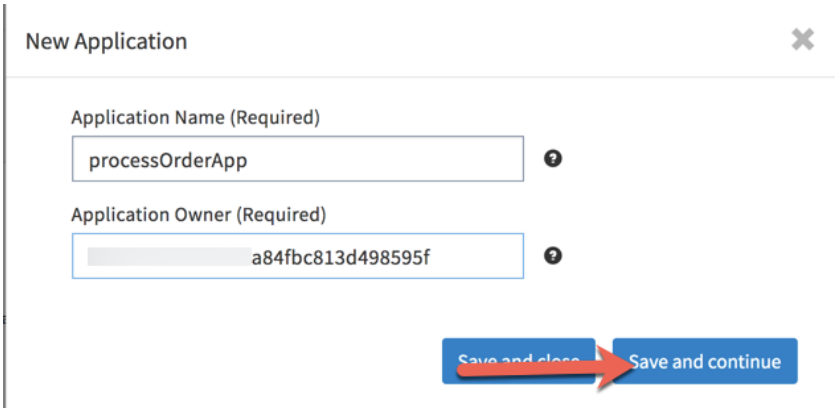
We will now generate an Application to use our Package. This can be done from the developer portal but since we are in the Control Center we will leverage it to create the application and its assorted API Key.

Click on Applications and then New Application as shown below:

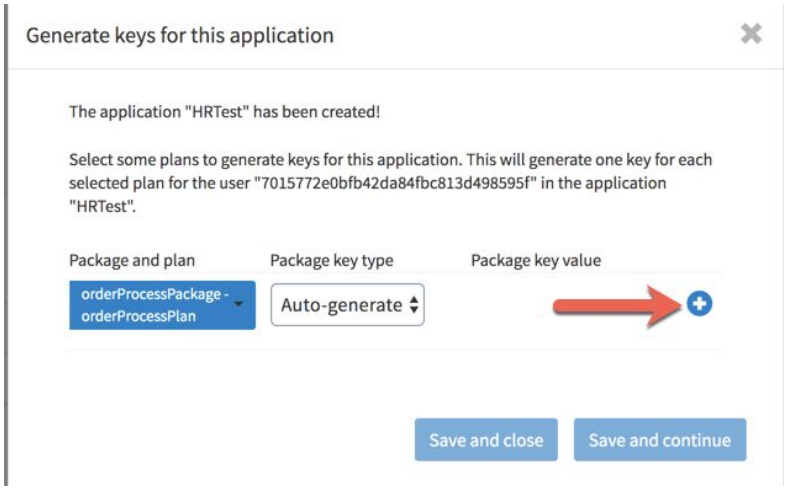




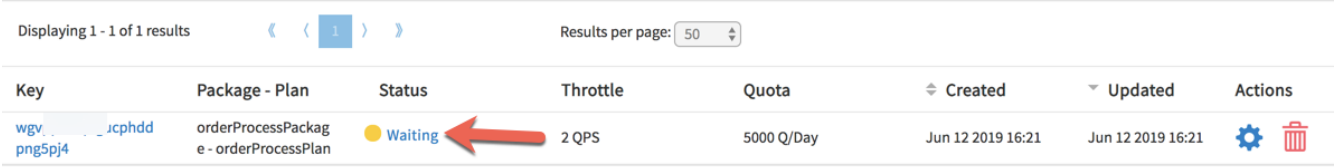
Enter the Application Name orderProcessApp and your UserID that was obtained earlier and click Save and Continue:



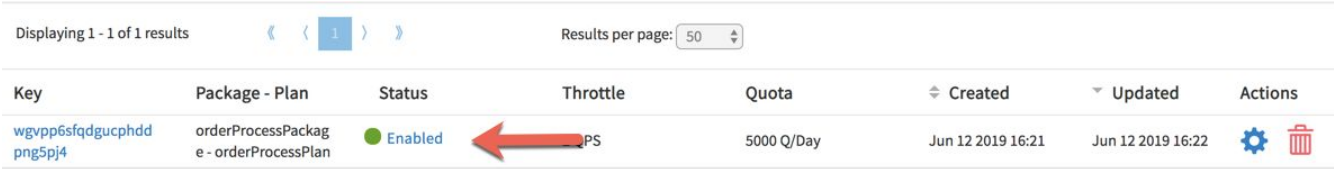
Select processOrderPackage and then orderProcessPlan and then click the + to enable the Save and Continue button.



Once the Plan is successfully added click Save and Close. **You should see a key for your application. Copy this key and paste it to a text file for use later:**



Click on the waiting link to enable the key:



We are now ready to test our Application.

STOP AND WAIT FOR THE INSTRUCTOR BEFORE PROCEEDING

Testing the API

We are now ready to test our API. You may now launch Postman that we downloaded before we started. We will need 2 pieces of information we collected earlier:

1. The API endpoint created
2. The API Key created for the endpoint

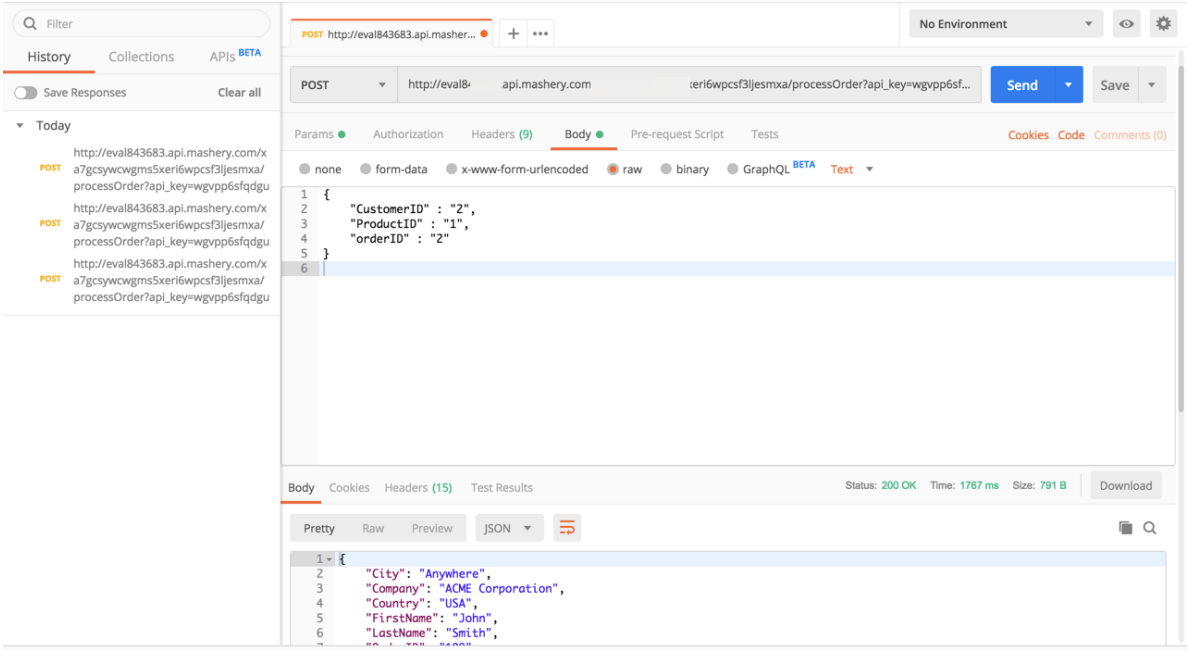
The URL will look like this: <API Endpoint>**?api_key=**<API Key>

The Method should be set to POST

The body should contain the following JSON payload:

```
{
        "CustomerID" : "2",
        "ProductID" : "1",
        "orderID" : "2"
}
```

A sample is shown below:



Summary

We have now completed our workshop. Let us review what we did:

1. Created an API Specification
2. Create a Mock Application
3. Created a Web Based Flogo App
4. Tested the Application
5. Created an API from the Application
6. Tested Connectivity to the API via Postman

We have barely scraped the surface of what is capable by leveraging the power of TIBCO Cloud Integration.

Where will your digital transformation take you? The sky (or the cloud) is the limit.