



# Leveraging Power Apps for SAP on Azure Workshop

Hands-on Lab Step-by-Step

March 2022

# Table of Contents

<b>Lab Overview and Pre-requisites.....</b>	<b>4</b>
<i>Abstract and Learning Objectives.....</i>	4
<i>Lab structure and Learning Objectives.....</i>	5
<b>Pre-requisites .....</b>	<b>6</b>
<i>Task 1: Check your user to access SAP .....</i>	6
<i>Task 2: Open Power Apps Studio.....</i>	6
<b>Connect to SAP from Power Platform.....</b>	<b>7</b>
<b>Exercise 1: Define the connection .....</b>	<b>7</b>
<i>Task 1: Create Custom Connector.....</i>	7
<i>Task 2: Test your Connector.....</i>	10
<b>Exercise 2: Create Power App with SAP Data.....</b>	<b>13</b>
<i>Task 1: Create new app.....</i>	13
<i>Task 2: Connect to data to display a list of Sales Orders from SAP.....</i>	21
<i>Task 3: Create a gallery to display a list of Sales Orders from SAP .....</i>	23
<b>Exercise 3: Add a new screen to show products from SAP.....</b>	<b>50</b>
<i>Task 1: Add a new screen .....</i>	50
<i>Task 2: Set up the navigation between screens.....</i>	51
<i>Task 3: Add a gallery for Products to the SAP Products Screen .....</i>	54
<i>Task 4: Add a Search bar for the Products .....</i>	62
<i>Task 5: Add a combo box and label for selecting Customer Name .....</i>	65
<i>Task 6: Add a date picker box and label for selecting the Delivery Date.....</i>	70
<b>Exercise 4: Add Power Automate flows.....</b>	<b>74</b>
<i>Task 1: Create Power Automate Flow to add new Sales Order in SAP.....</i>	74
<i>Task 2: Create Power Automate Flow to update price.....</i>	<b>Error! Bookmark not defined.</b>
<i>Task 3: Use flow in Power App to update price .....</i>	<b>Error! Bookmark not defined.</b>
<i>Task 4: Test your App.....</i>	<b>Error! Bookmark not defined.</b>
<b>Exercise 5: Import Power App into Teams. .....</b>	<b>90</b>

## Leveraging Power Apps with SAP on Azure Workshop

<i>Task 1: Final Save and Publish App.....</i>	91
<i>Task 2: Add app to Teams.....</i>	91
<b>Appendix.....</b>	<b>94</b>
<i>How to create a Trial Environment?.....</i>	94
<b>Copyright .....</b>	<b>99</b>

# Lab Overview and Pre-requisites

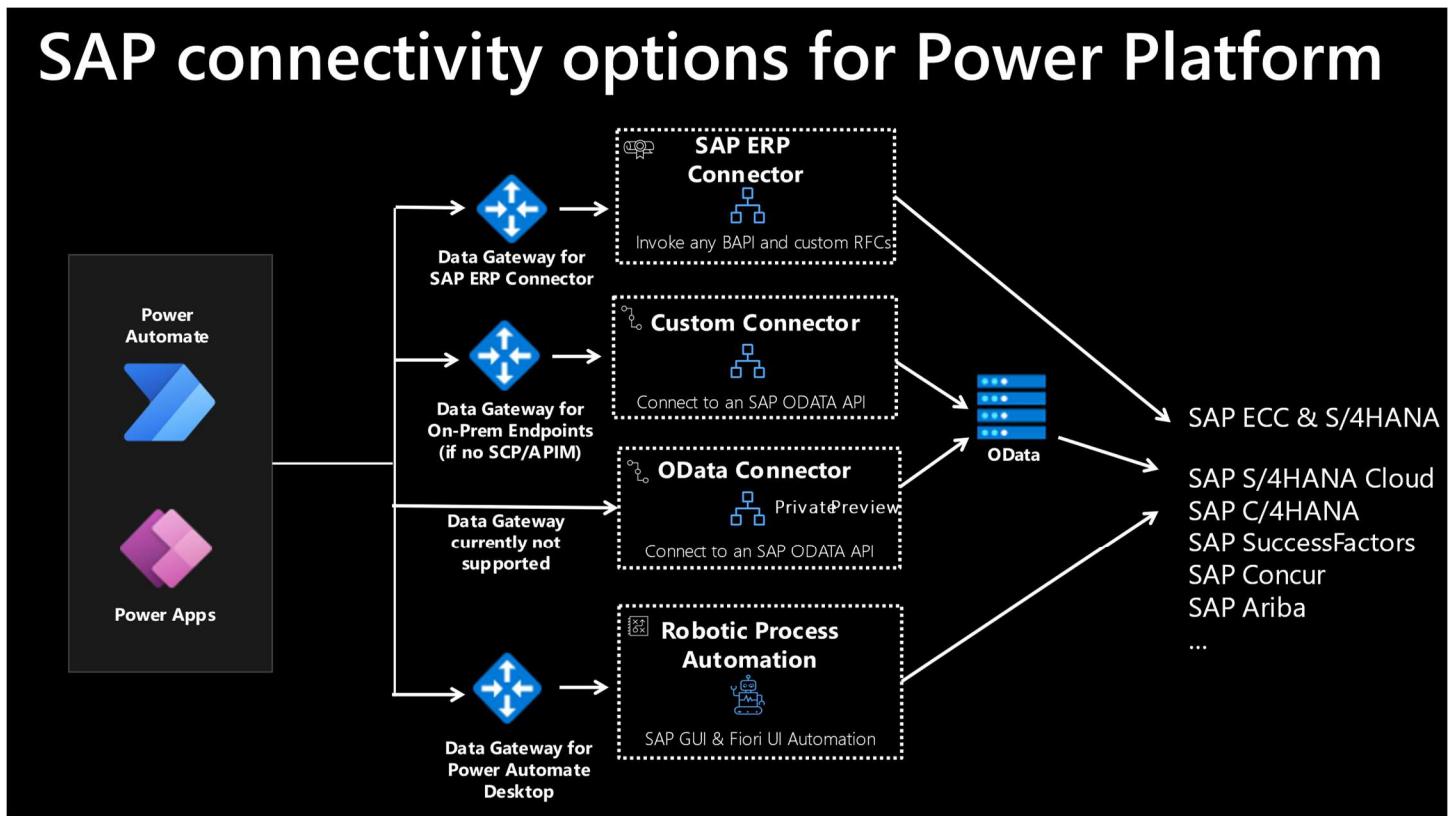
## Abstract and Learning Objectives

For Power Platform connected to SAP scenarios you can choose between different technical options:

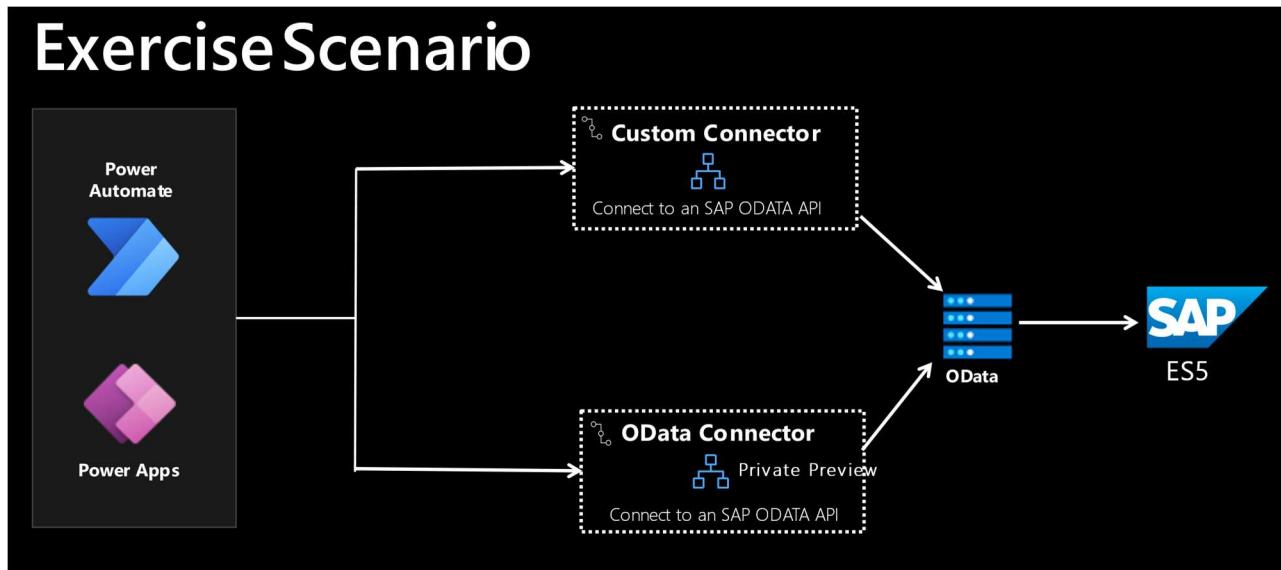
- OData connectivity through a Custom Connector or the new OData connector in private preview
- BAPI/RFC connectivity through the SAP ERP Connector
- RPA capabilities with Power Automate through the SAP GUI

Which option to use really depends on the company integration strategy, the version of the SAP system, if the system is API enabled and so on.

If the SAP system is hosted in a private cloud/data centre the On-premise data gateway needs to be installed and configured for the three options.



For this lab we will use OData connectivity, probably the most popular way of connecting to SAP systems as many customers will have an API strategy in place. We will be connecting to an SAP system available on the Internet called ES5. **This system is used for testing and training purposes only.**



The scenario that we will be creating is a canvas app that displays Sales Orders and also allows you to create a new Sales Order that is sent to SAP via Power Automate.

## Lab structure and Learning Objectives

The lab is divided into pre-requisites and exercises.

# Pre-requisites

Before working through the exercises, you need to complete the following tasks.

## Task 1: Check your user to access SAP

1. You will need an SAP user ID and password for the ES5 system available from SAP. This system is used for testing and learning purposes only. A user is available just for this week for the exercises in this hands-on lab and we will share with you in the lab session in the chat window.
2. Optionally you could sign up and get your own user in ES5. Follow this tutorial [Create an Account on the SAP Gateway Demo System | Tutorials for SAP Developers](#)
3. Test your access by logging on to <https://sapes5.sapdevcenter.com/>

## Task 2: Open Power Apps Studio

1. If you have a demo tenant available where Custom Connectors can be created use that. If not, create a trial as described in the [Appendix](#) of this document

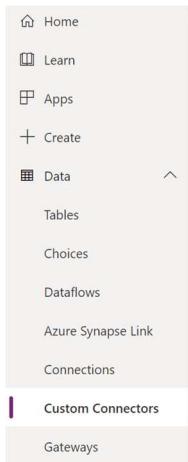
# Connect to SAP from Power Platform

## Exercise 1: Define the connection

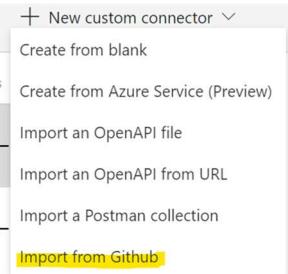
In this exercise, you will create the custom connector to call an SAP API on the ES5 system available from SAP in internet.

### Task 1: Create Custom Connector

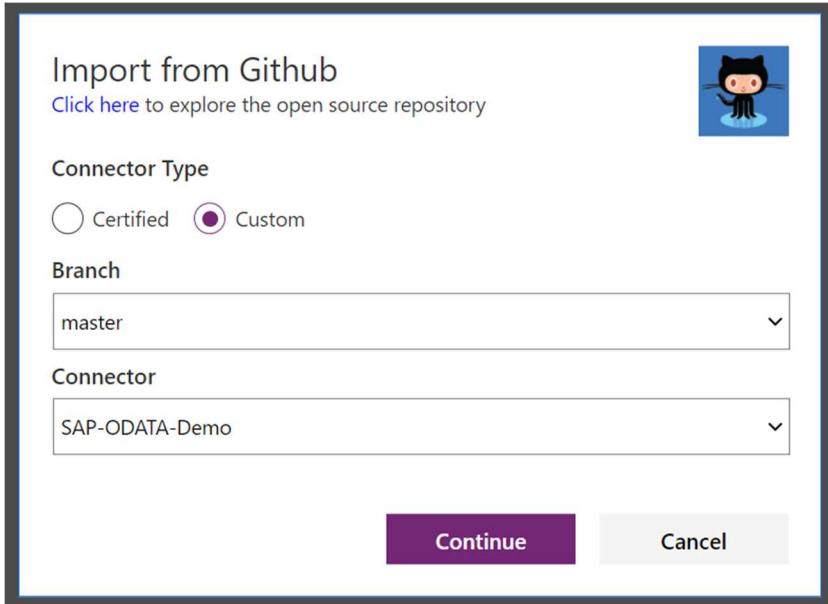
1. Open Power Apps [Power Apps maker portal](#) and navigate to Data->Custom Connectors on the right panel



1. From the New custom connector on the right choose Import from Github



2. Choose Connector Type- Custom, Branch: master and Connector SAP-ODATA-Demo and choose Continue



3. Change the name of your connector to SAP-ODATA-<yourusername>  
Optionally change the icon or icon background color.

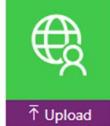
← Connector Name **SAP-ODATA-MARYSOL...**

1. General > 2. Security > 3. Definition > 4. Code (Preview) > 5. Test       Swagger Editor       Create connector     

**General information**

Add an icon and short description to your custom connector. Your host and base URL will be automatically generated from the swagger file.

**General information**

  Upload connector icon  
Supported file formats are PNG and JPG. (< 1MB)

Icon background color

Description

Connect via on-premises data gateway [Learn more](#)

Scheme \*  
 HTTPS  HTTP

Host \*

Base URL

[Security →](#)

The host for ES5 has been filled out already by the template.

#### 4. Click **Security**

The screenshot shows the '2. Security' step of the connector creation process. At the top, the breadcrumb navigation shows: Connector Name > SAP-ODATA-MARYSOL > 2. Security > 3. Definition > 4. Code (Preview) > 5. Test. Below the navigation, there are tabs for Swagger Editor, Create connector, and Cancel.

**General information**

Add an icon and short description to your custom connector. Your host and base URL will be automatically generated from the swagger file.

**Icon**: A placeholder icon showing a globe and a user profile.

**Upload connector icon**: A button to upload a connector icon (PNG or JPG, less than 1MB).

**Icon background color**: A color picker input field.

**Description**: A text area containing: "SAP ODATA Demo using a custom connector. This connects to the sample ODATA service provided by SAP. This connector is provided as a sample as-is, for illustrative purpose only."

**Connect via on-premises data gateway**: An unchecked checkbox with a link to learn more.

**Scheme \***: A radio button selected for "HTTPS".

**Host \***: The value "sapess5.sapdevcenter.com" is entered.

**Base URL**: The value "/" is entered.

**Security →**: A link to the next step.

#### 5. Leave it as it is. Our new connector will be using Basic Authentication. Click **Definition**

The screenshot shows the '3. Definition' step of the connector creation process. At the top, the breadcrumb navigation shows: Connector Name > SAP-ODATA-MARYSOL > 2. Security > 3. Definition > 4. Code (Preview) > 5. Test. Below the navigation, there are tabs for Swagger Editor, Create connector, and Cancel.

**Authentication type**

Choose what authentication is implemented by your API \*: Basic authentication.

**Basic authentication**

Users will have to provide a valid user name and password before using this API.

Parameter label *	Parameter name
username	username
password	password

**Do NOT enter secrets here. These fields are used to configure display names for connections.**

**General** < **Definition →**

## Leveraging Power Apps with SAP on Azure Workshop

- In the Definition tab you can see all the actions available from the API. You can choose one from the left and look at the General, Request and Response to understand how the action looks like.

The screenshot shows the 'Definition' tab of a Power Apps connector named 'SAP-ODATA-MARYSQL'. On the left, there are two expandable sections: 'Actions (14)' and 'References (20)'. The 'Actions (14)' section is expanded, listing numbered actions from 1 to 14. Action 1, 'ListProductSets...', is selected and its details are shown on the right. The 'General' tab is active, displaying the action's summary ('List product sets'), description ('Fetch the list of products'), operation ID ('ListProductSets'), and visibility settings ('none'). Below the General tab is the 'Request' tab, which includes fields for Verb (set to GET), URL (https://sapess5.sapdevcenter.com/sap/opu/odata/wbep/GWSAMPLE\_BASIC/ProductSet), Path (empty), Query parameters (empty), and Headers (x-csrf-token). A 'New action' button is located at the bottom of the actions list.

- When ready Click **Create Connector**

The screenshot shows the 'Create connector' button highlighted in yellow on the top right of the Power Apps connector creation interface. The interface includes tabs for General, Security, Definition, Code (Preview), and Test. The 'Definition' tab is active. The 'Actions (14)' section is expanded, showing the same list of actions as the previous screenshot. The 'General' tab is active, displaying the action's summary ('List product sets').

- Your connector has been created and saved.

## Task 2: Test your Connector

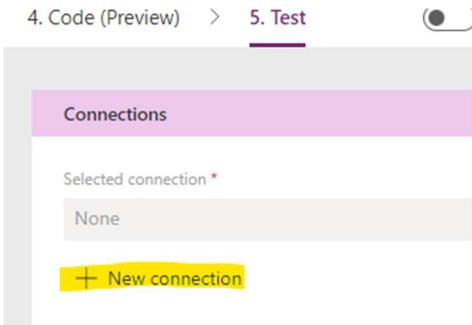
In this task, you will test your newly created connector.

- Now that your connector has been created choose the **Test** tab

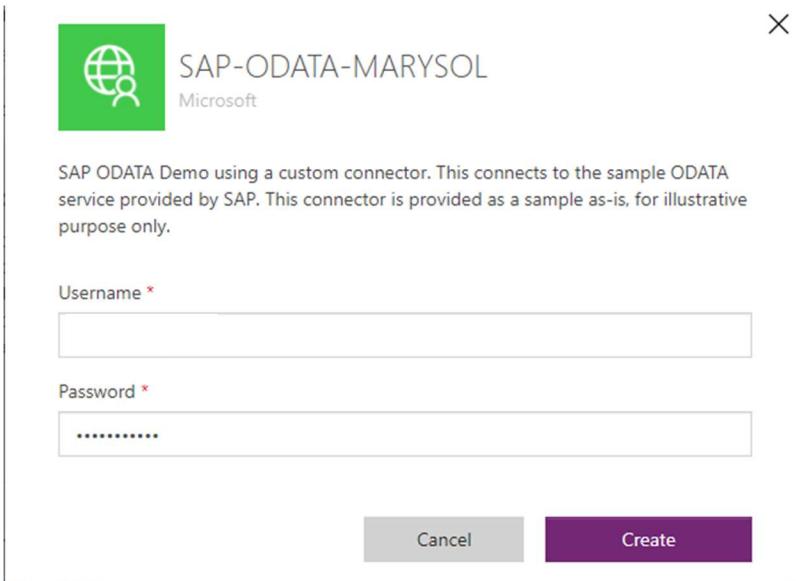
The screenshot shows the 'Test' tab highlighted in yellow on the top right of the Power Apps connector test interface. The interface includes tabs for General, Security, Definition, Code (Preview), and Test. The 'Test' tab is active. The 'Actions (14)' section is expanded, showing the same list of actions as the previous screenshots. The 'General' tab is active, displaying the action's summary ('List product sets').

3. Before we can test the connector, we need to create a Connection. For this you will require your ES5 user id and password. The password is in the chat window. If you cannot see the password, please send a message.

4. Choose **New Connection**



5. In the pop up enter your P-number and password for ES5 and choose **Create**



When the connection is created it you may have to navigate back to Data-> Custom Connectors and find your connector. Connectors should be in alphabetical order. Choose the pencil to go back to Edit mode.



Choose the Test tab again. If you don't see your connection, click the refresh button in the Connections area



6. Now you should be ready to Test.
7. You can test some of the operations. For example, choose ListProductSets from the Operations list and enter 5 in the \$top parameter and click **Test Operation**.

In the Response area you should see the Top 5 Products data coming back from the SAP system.

**Operations (14)**

These are the operations defined by your custom connector. This includes actions and triggers.

- 1 ListProductSets
- 2 AddProduct
- 3 GetProduct
- 4 UpdateProduct
- 5 ListSalesOrders
- 6 GetSalesOrder
- 7 ListBusinessPartner
- 8 GetBusinessPartner
- 9 ListProductTypeC...
- 10 ListProductCateg...
- 11 ListCurrencyCode
- 12 ListMeasureUnit
- 13 ListWeightUnit
- 14 ListLengthUnit

**ListProductSets**

\$format \*  
json

\$filter

\$top  
5

\$skip

\$select

x-csrf-token

**Test operation**

**Request**      **Response**

**Status**  
(200)

**Headers**

```
"sap-perf-fesrec": "261777.000000",
"sap-processing-info": "ODataBEP=.crp=.RAL=.st=X,MedCacheHub=Table,MedCacheBEP=SHN",
"sap-server": "true",
"x-ms-apihub-cached-response": "false",
"x-ms-apihub-obo": "true"
```

**Body**

```
{
  "id": "https://sapess5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_BASIC/Product",
  "uri": "https://sapess5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_BASIC/Product",
  "type": "GWSAMPLE_BASIC.Product",
  "etag": "W/\\"datetime'2022-02-10T02%3A15%3A16.7282130\\\"",
},
"ProductID": "0003350",
"TypeCode": "PR",
"Category": "Accessories",
"Name": "MEDICAL PROBE",
"NameLanguage": "EN",
"Description": "To Be Filled In Later",
"DescriptionLanguage": "EN",
"SupplierID": "0100000047",
"SupplierName": "Becker Berlin",
```

- If you want to do another test choose the GetProduct operation and enter HT-1000 as the id and click Test Operation.

The screenshot shows the Power Apps Maker portal interface. On the left, a sidebar lists 14 operations under 'Operations (14)'. The 'GetProduct' operation is selected and highlighted with a green checkmark. The main area displays the 'GetProduct' configuration screen. It includes fields for 'id' (set to 'HT-1000'), 'x-csrf-token' (empty), and '\$format' (set to 'json'). A 'Test operation' button is visible. Below this, the 'Response' tab is selected, showing the API request status (200) and headers, which include SAP-specific metadata like 'sap-perr-resrec' and 'sap-processing-info'. The 'Body' section shows the JSON response for the product 'HT-1000', detailing its type ('GWSAMPLE\_BASIC.Product'), etag ('W/"datetime2022-02-10T00%3A01%3A31.000000"'), product ID ('HT-1000'), type code ('PR'), category ('Notebooks'), name ('Notebook Basic 15'), name language ('EN'), description ('Notebook Basic 15 with 2.80 GHz quad core, 15" LCD, 4 GB DDR3 RAM, 500 C'), description language ('EN'), supplier ID ('010000046'), supplier name ('SAP'), tax tariff code (1), and measure unit ('EA').

CONGRATULATIONS!! You have created your custom connector calling an SAP API of the ES5 system by using a Github Template. For more information about the template go to [GitHub](#)

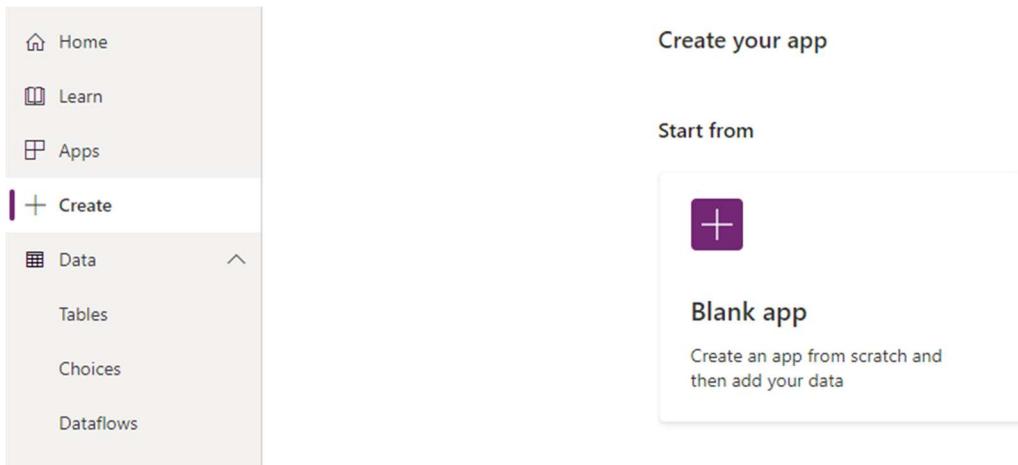
## Exercise 2: Create Power App with SAP Data.

In this exercise, you will create a simple power app to interact with SAP through your new custom connector created in exercise 1.

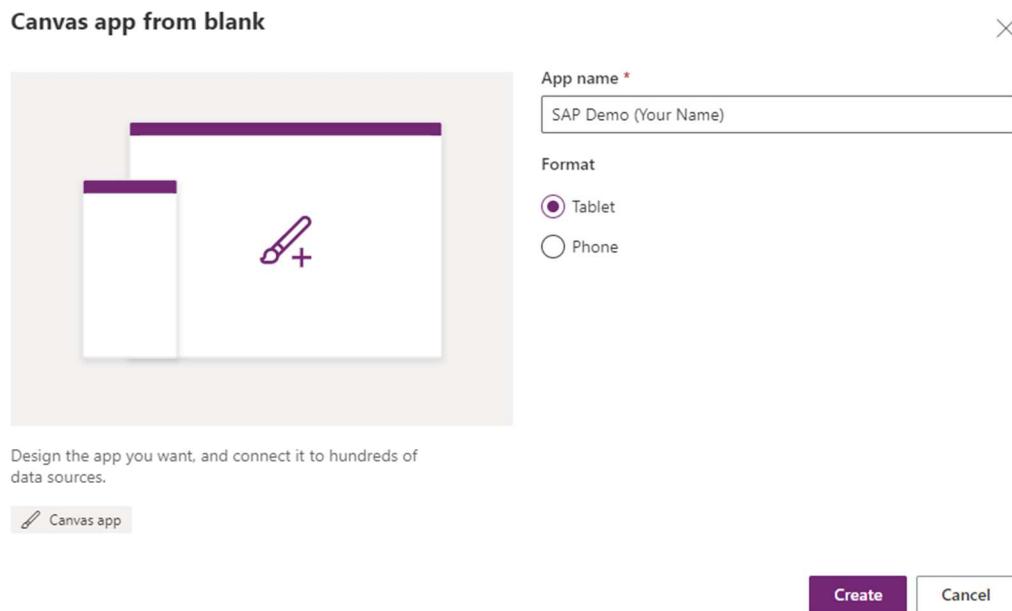
### Task 1: Create new app

2. Navigate to [Power Apps Maker portal](#) and make sure you are in the correct environment. Choose **Create** and **Start from Blank App**.

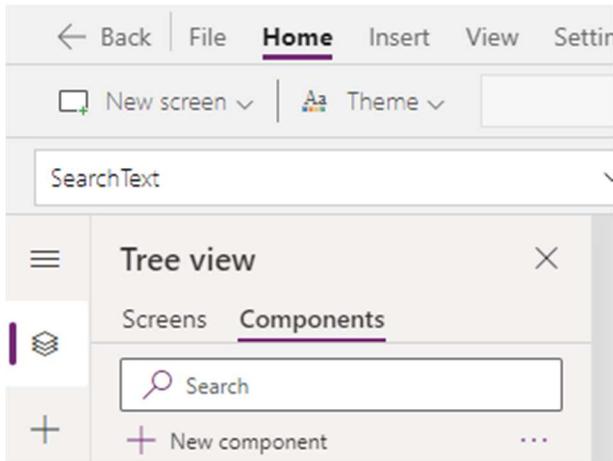
## Leveraging Power Apps with SAP on Azure Workshop



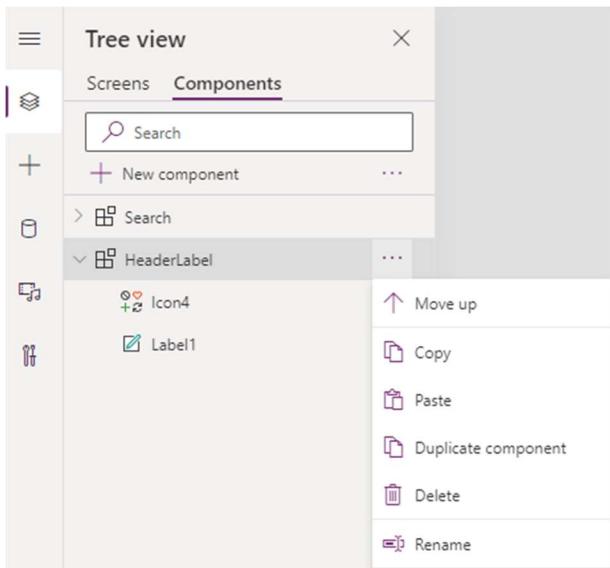
3. Choose Blank Canvas App and give a name to your app for example SAP Demo <yourname>, select Tablet as the format then click on **Create**.



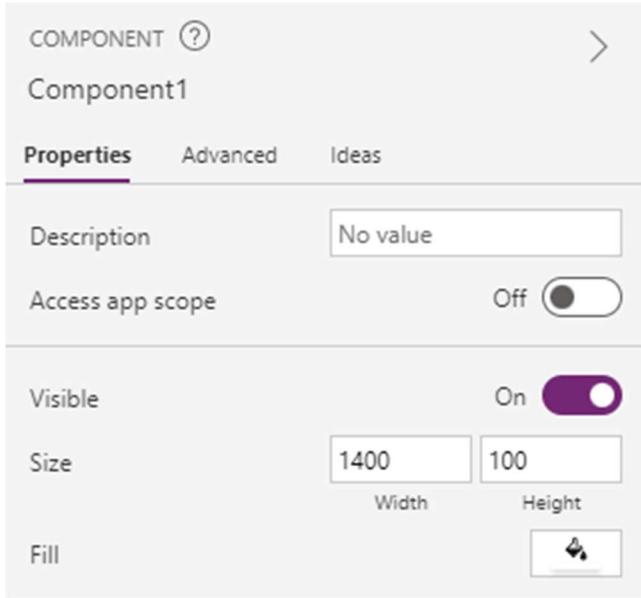
4. We are going to create the Header Label for the first screen that we will reuse, so we will create it as a Component. In the left-hand navigation pane, click on **Components** in the **Tree View**.



5. Click on **+ New Component**. Rename the Component to **HeaderLabel** by clicking on the "..." and selecting **Rename**.



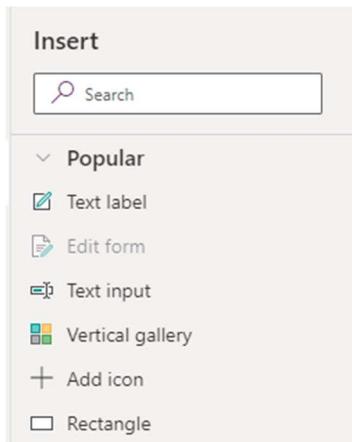
6. On the right-hand pane under **Properties**, change the **Size** to: Width – 1400 and Height – 100. This will stretch it across the entire screen and an appropriate height.



7. On the Canvas, click on **Add an item from the Insert Pane**.

Add an item from the Insert pane or connect to data

8. In the **Insert** Pane, select **Text Label**.

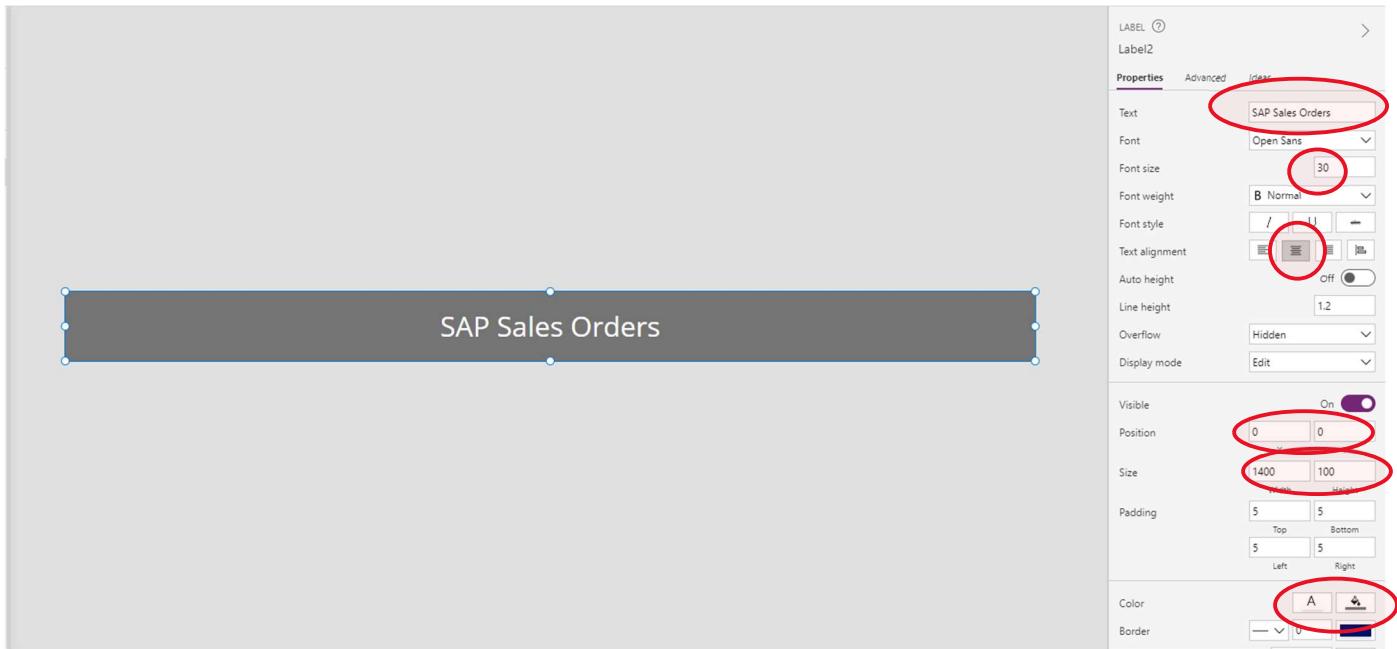


9. In the right-hand pane under **Properties**:

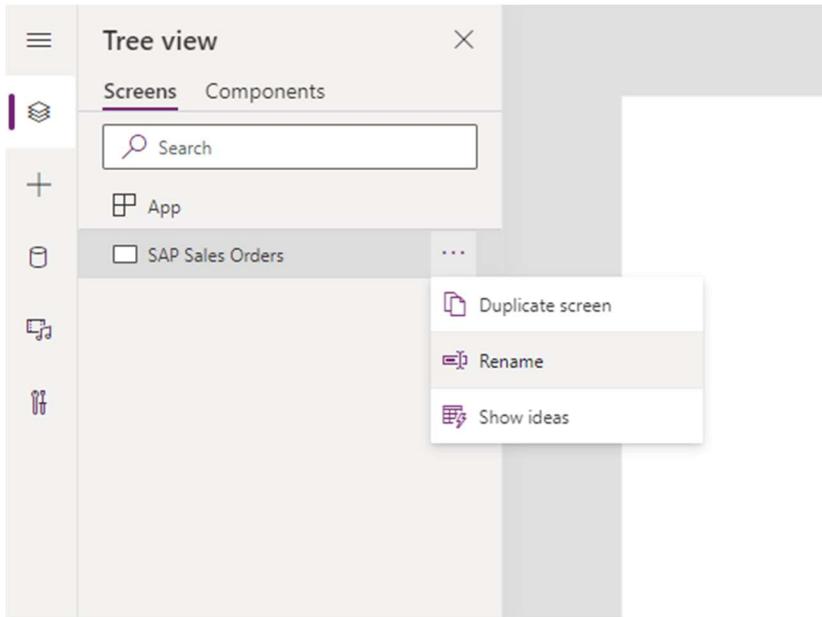
- Change the Text to – SAP Sales Orders
- Increase Font Size to 30
- Change Text Alignment to Center
- Change Position to X – 0, Y – 0
- Set the size to: Width – 1400 and Height – 100.
- Change the text Color to White.

- Change the Fill color to match your theme.

Close the **Insert Screen** panel. Now this component is available to use on any of your screens.

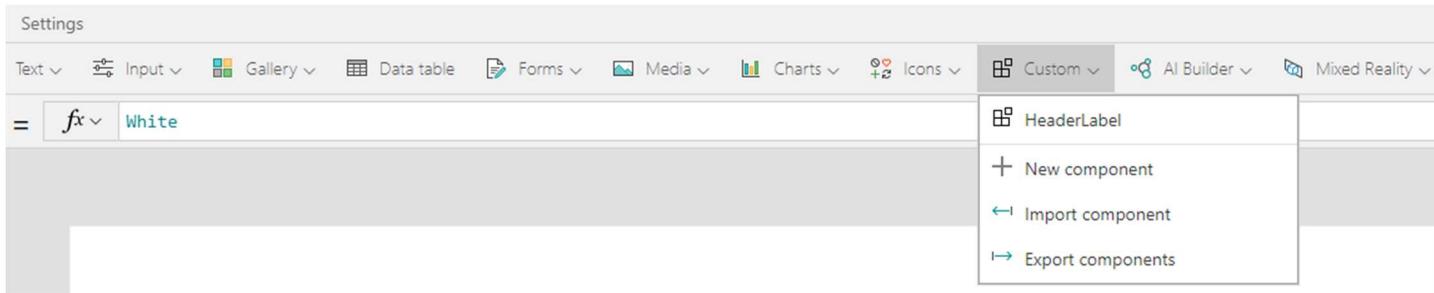


10. Now we will rename the screen. On the left-hand pane, click on **Tree View** and select **Screens** rather than **Components**. Select **Screen1**. Click “...” next to **Screen1** (or right click Screen1) and select the **Rename** option. Change the name to **SAP Sales Orders**.

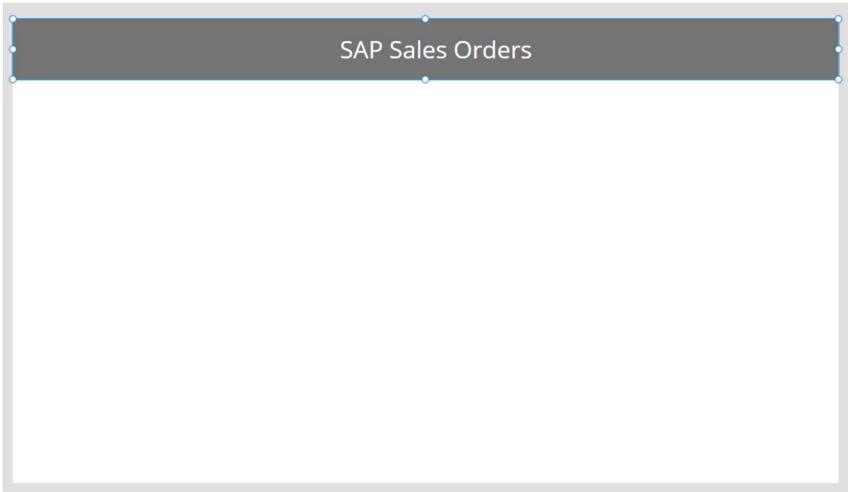


**Tip:** It is a good practice to rename screens and controls as you create them, so they are easier to locate as you work with formulas that reference different controls. In this lab, you will be prompted to rename screens and some of the controls. For the others, you may rename them as you please on your own. It is important that you rename screens as prompted in this lab as future steps may rely on specific screen names.

11. Now let's use our new component. In the ribbon, click on **Custom** and select the **HeaderLabel**.



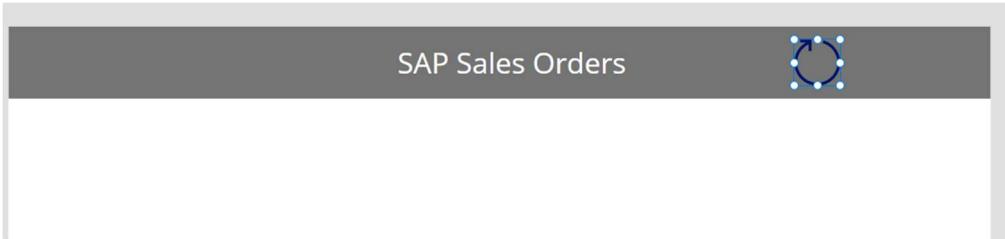
Your screen should now look like this:



12. Add an Icon to refresh the Sales Orders from the SAP demo system. In the ribbon, click **Insert** and then **Icons**. Scroll down and select the **Reload** icon.

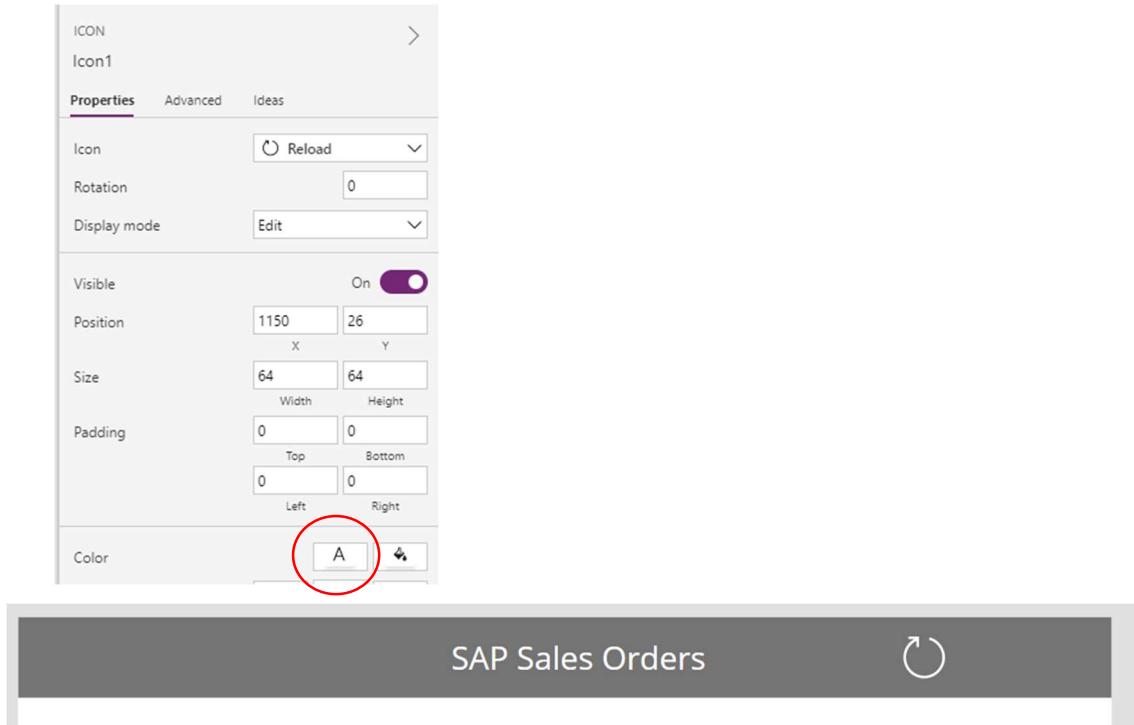


13. Drag into the right-hand side of the label and resize to fit in the label.

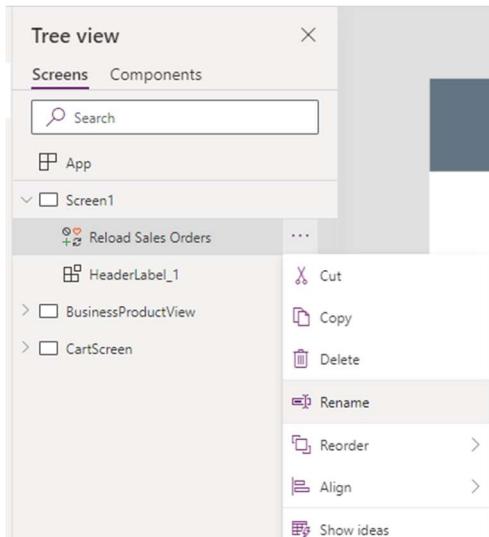


14. In the **Properties** tab on the change the text **color** if required.

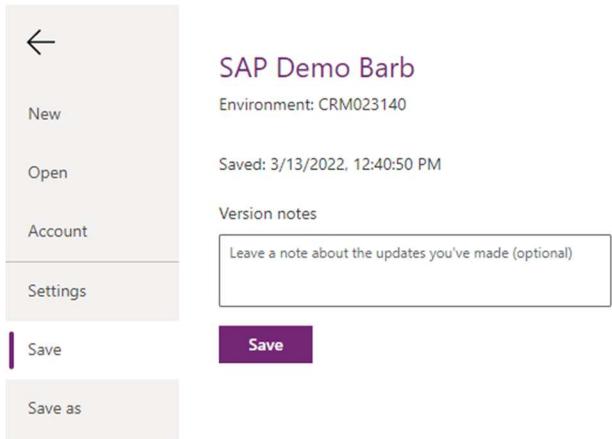
## Leveraging Power Apps with SAP on Azure Workshop



15. Rename the icon. Click on **Tree View** and select the icon. Click on "...." and select **Rename**. Rename the icon to Reload Sales Order.



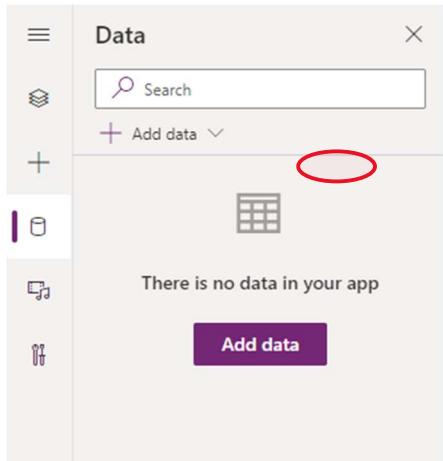
16. Save your app. In the ribbon, click on **File** and then **Save**.



Click on the **back arrow** to go back to editing the app.

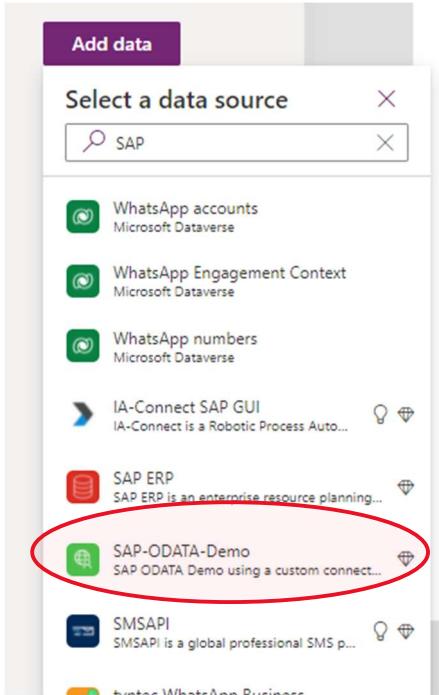
## Task 2: Connect to data to display a list of Sales Orders from SAP

1. In the left panel, choose **Data** and click on **Add Data**

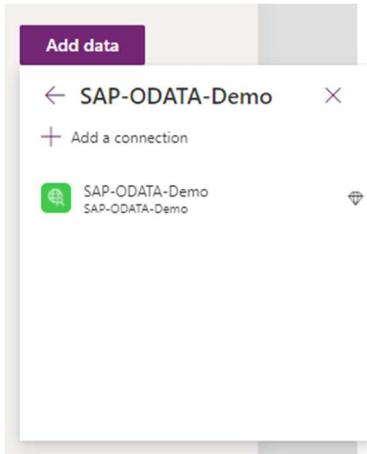


2. In the Search bar, type in SAP to find your custom connector.

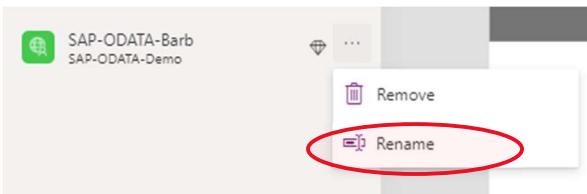
## Leveraging Power Apps with SAP on Azure Workshop



3. Click on the Connector and then select the connector to add the connection.



4. Rename the Connector by clicking on the ellipse "... " and select **Rename**. Rename to something specific to you – SAP-ODATA-<Your Name>. Now you are ready to use the connector.



5. Select the **reload icon**  and in the **OnSelect** formula bar, copy and paste section below with your specific connector name.

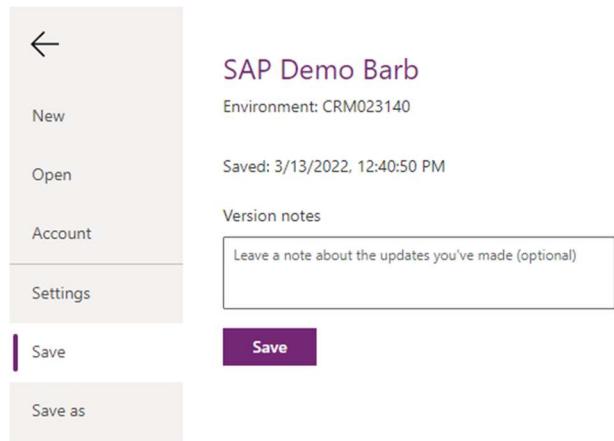
**TIP:** It may be easier to paste this into notepad first and update the connector name and then paste that into formula bar. If you are getting red squiggly lines double check you have all the punctuation marks in the right spot. You can start from scratch and use the Intellisense options to complete filling out the formulas.

```
ClearCollect(zSalesOrders,'SAP-ODATA-<your name>'.ListSalesOrders({'$top':15}).d.results)
```

This function will collect the top 15 Sales Orders from ES5 in a collection called zSalesOrders.



6. Save your app. In the ribbon, click on **File** and then **Save**.



Click on the **back arrow** to go back to editing the app.

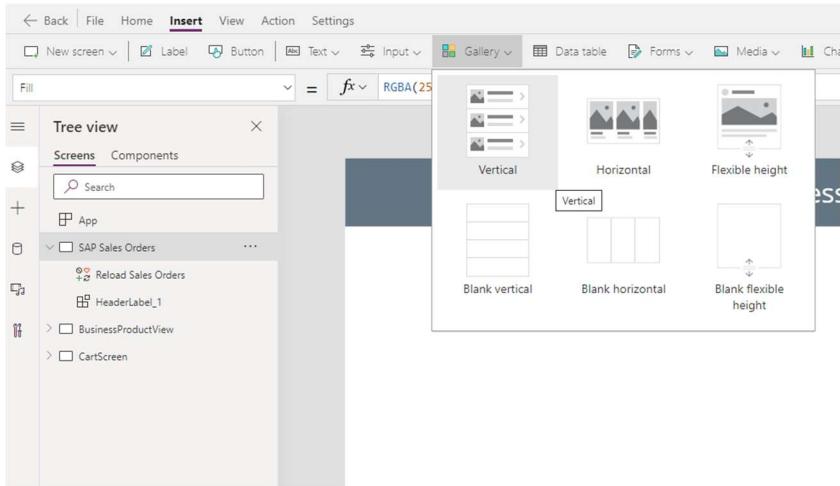
## Task 3: Create a gallery to display a list of Sales Orders from SAP

We are now going to add a gallery. The first option is to add gallery with an existing template. I would recommend this for people who are just starting out with Power Apps. It is the easier route as Power Apps has a gallery has a template for different layouts with the fields and you can add additional fields from a drop-down list. The second option will be more complex as we are starting with a blank gallery and a container. Using containers is best practice if you want your Power App to be responsive. The container will shrink or expand depending on the amount of screen available. I would recommend this option if you have some experience with Power Apps and want to learn more about containers.

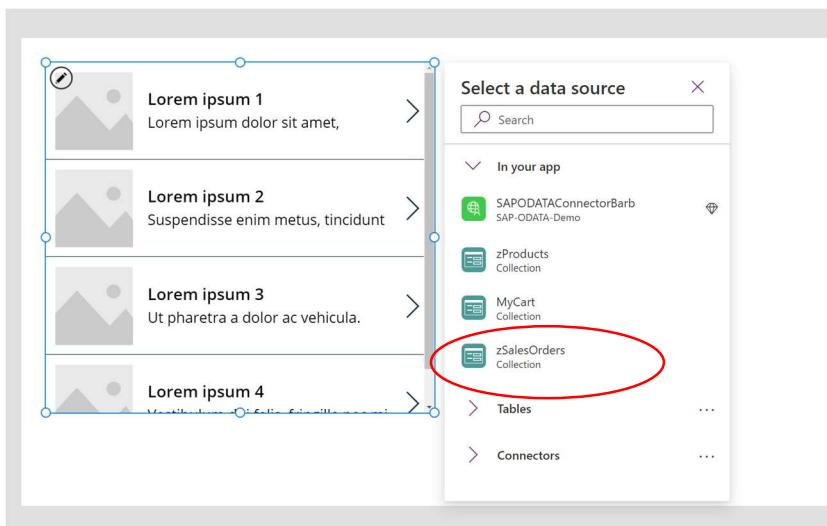
[Option # 1: Adding a gallery via a template to display Sales Orders.](#)

## Leveraging Power Apps with SAP on Azure Workshop

1. Add a new gallery to display the collection of Sales Orders. In the left-hand pane, select **Tree View** and click on **SAP Sales Orders** screen. In the Ribbon, select **Insert** and then click on **Gallery**. Select the **Vertical** gallery.

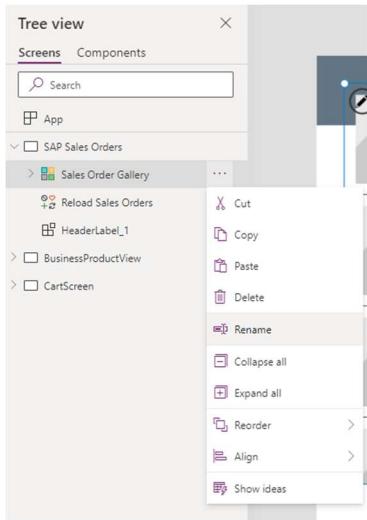


2. Select a data source. Click on the **zSalesOrders Collection** for your data source.

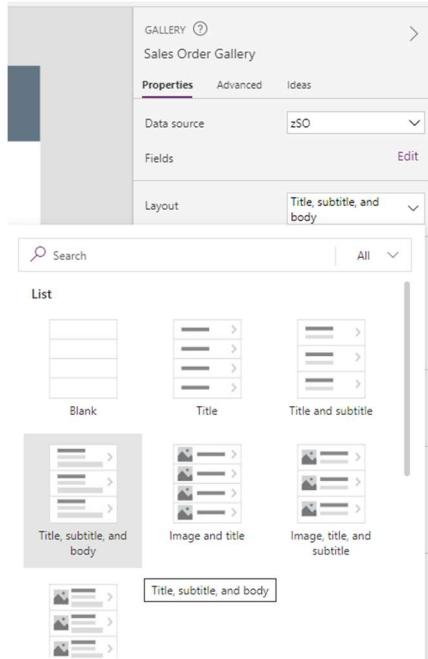


3. Rename the Gallery. Click on **Gallery 1** and then the "...". Click on **Rename** and rename the gallery to **Sales Order Gallery**.

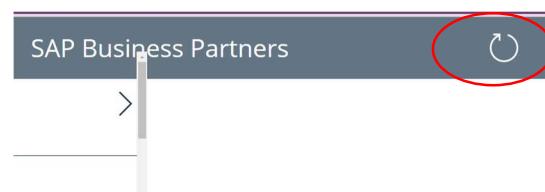
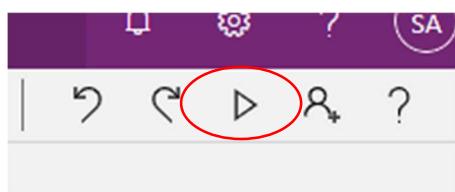
## Leveraging Power Apps with SAP on Azure Workshop



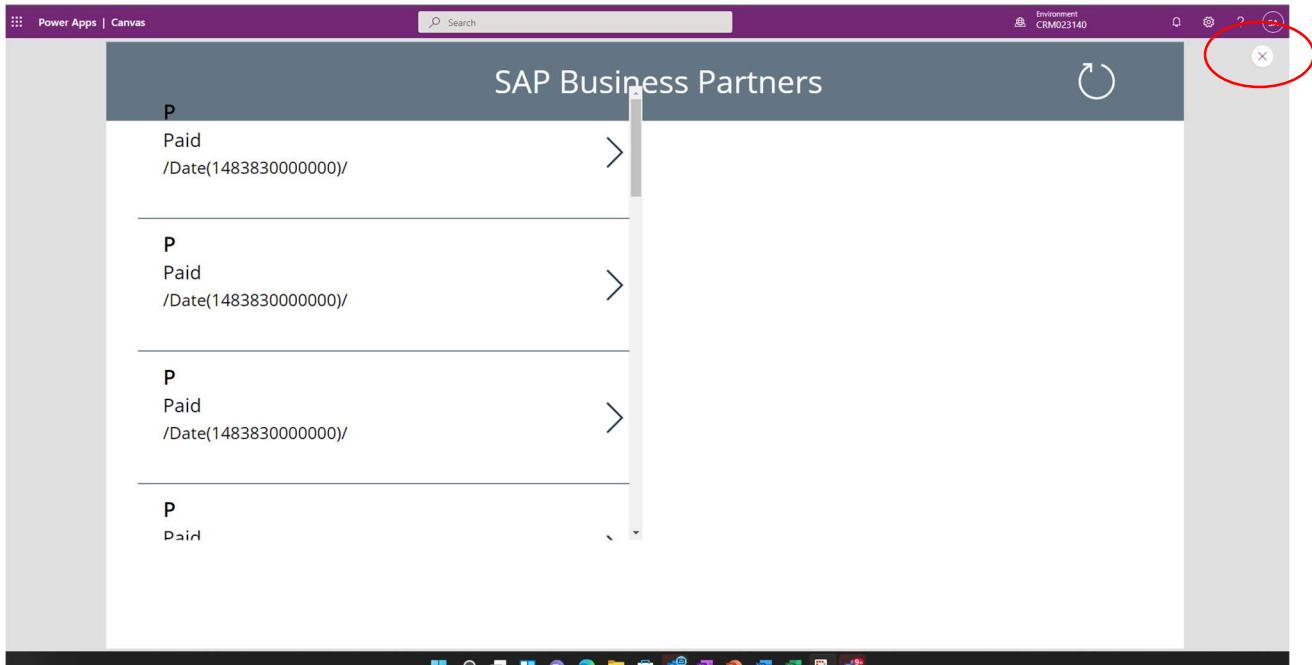
- Now we need to change the layout and map the correct fields to our gallery. On the right-hand pane under **Properties**, click on **Layout**. Since we don't have an image for sales orders, select **Title, subtitle, and body**.



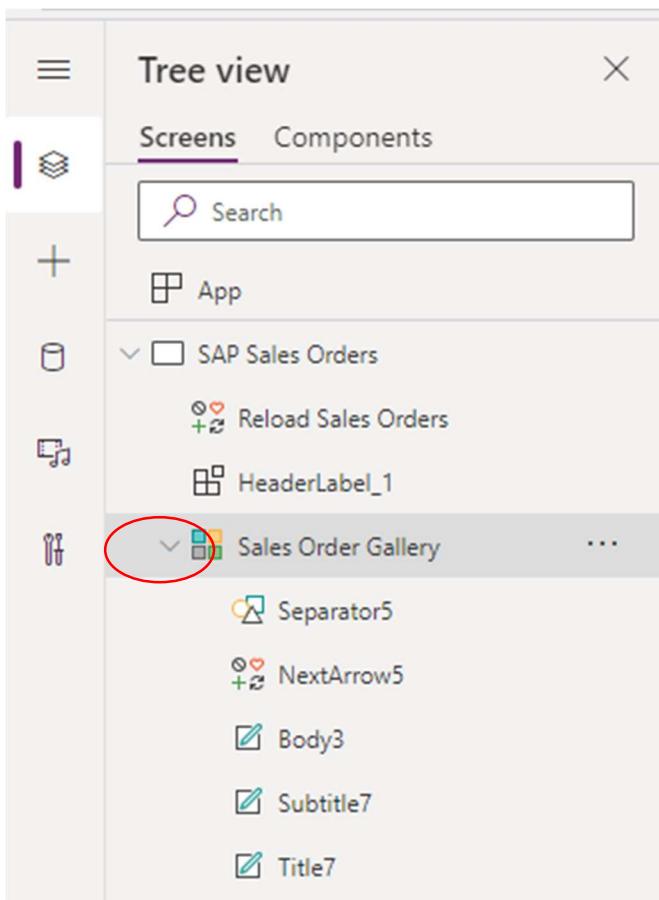
- You may not have any data showing up yet in your table. Click on the **Play** button in the ribbon and then click on the **Reload Icon** to load the data.



Your screen should now show some data. Close the app so we can edit again by clicking on the **X** in the upper right hand corner.



6. Rename and then map the correct fields in the gallery. In **Tree view**, click on the next to **SAP Sales Gallery** to expand it.

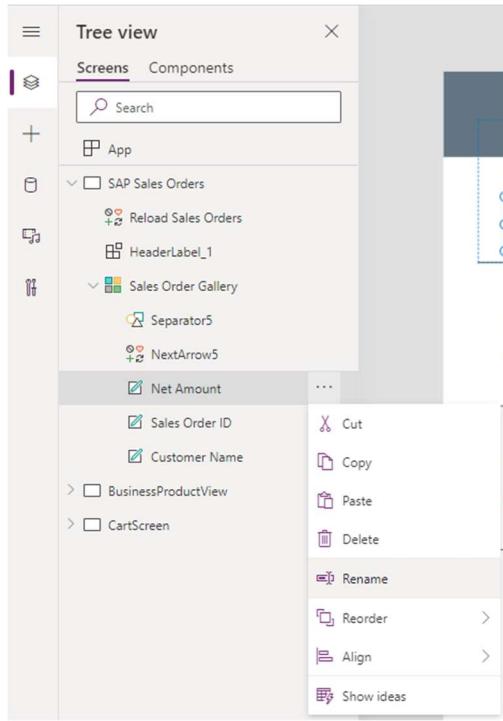


7. Rename the Body, Subtitle and Title fields. Click on the “...” next to each field and select **Rename**. Rename each field as listed below:

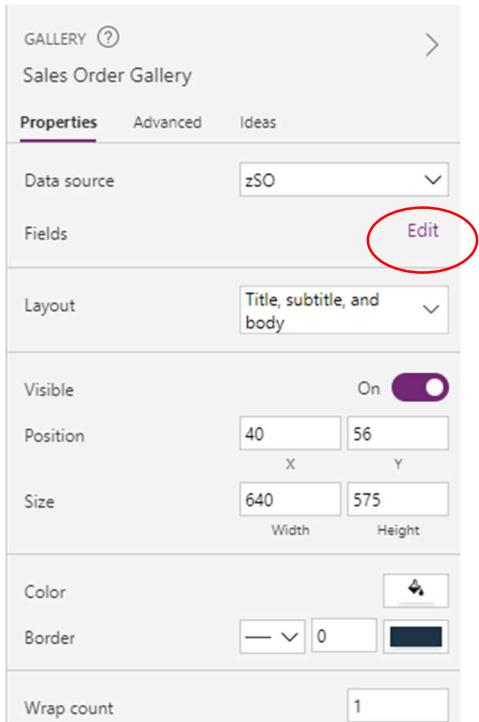
**Body** – Net Amount

**Subtitle** – Sales Order ID

**Title** – Customer Name



8. Change the fields shown in the gallery. With the **Sales Order Gallery** still selected, on the right-hand pane under **Properties**, click on **Edit** for **Fields**.



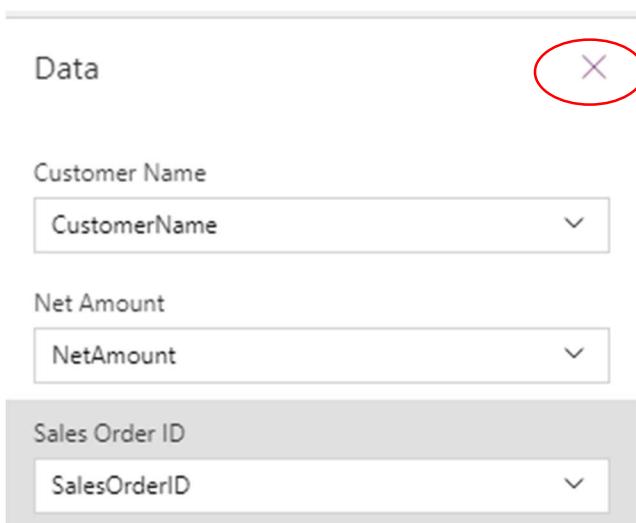
9. Using the down arrow, change the fields to the following:

Customer Name – **CustomerName**

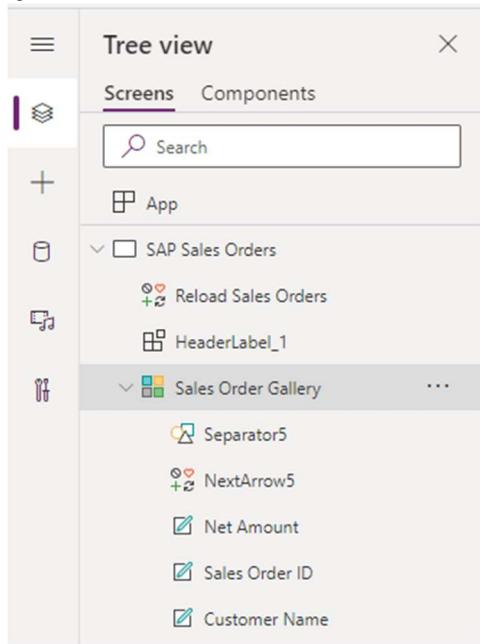
Net Amount – **NetAmount**

Sales Order ID - **SalesOrderID**

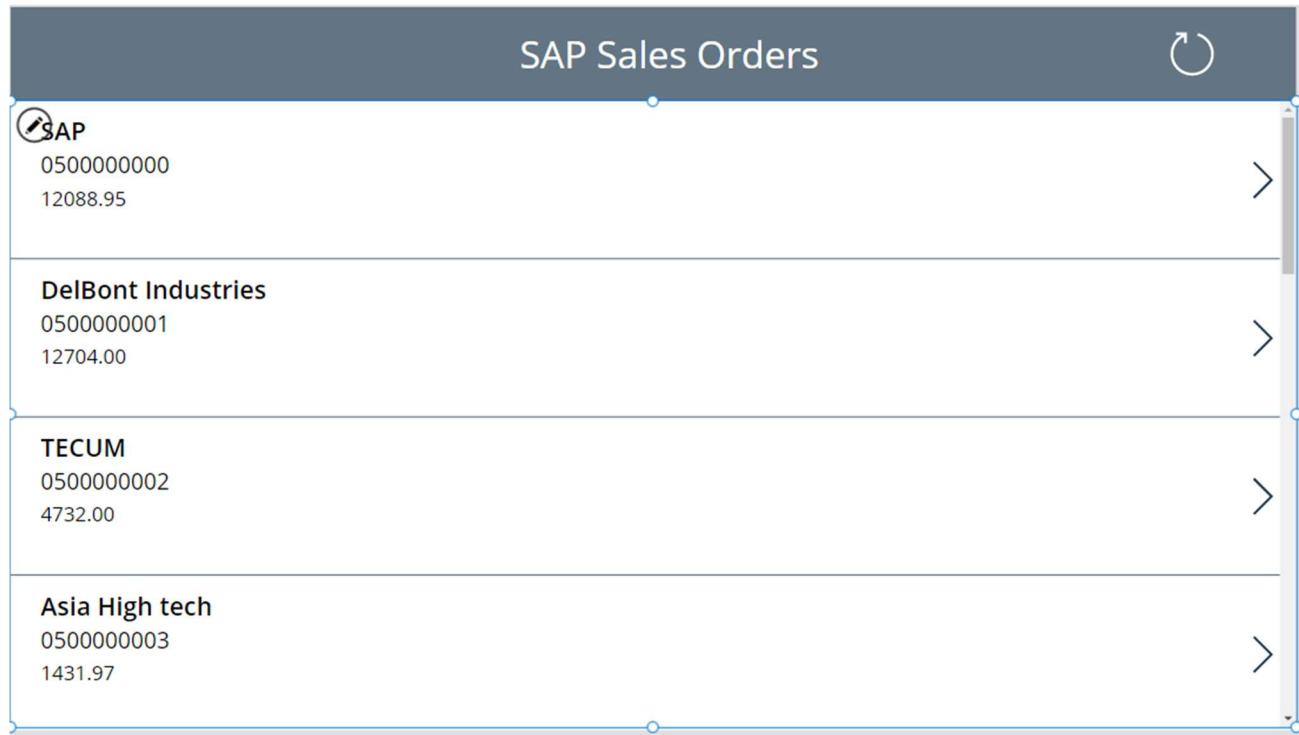
Close the data field by clicking on the **X** in the upper right corner.



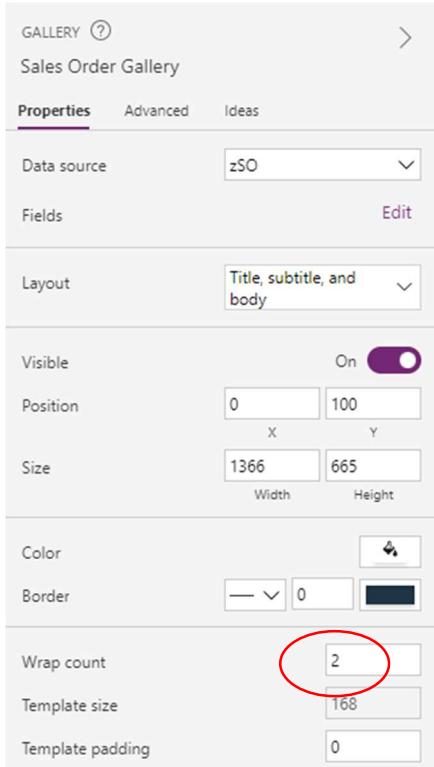
10. Move the gallery and change to 2 columns. In the left-hand navigation pane in **Tree view**, ensure that that **Sales Order Gallery** is selected.



11. Drag the gallery down under the Header Label and stretch it across the entire canvas. Your canvas should look like the screenshot below.



12. Change to 2 columns of data. In the right-hand pane under **Properties**, change the **Wrap count** to **2**.



Your screen should now look like the image below.



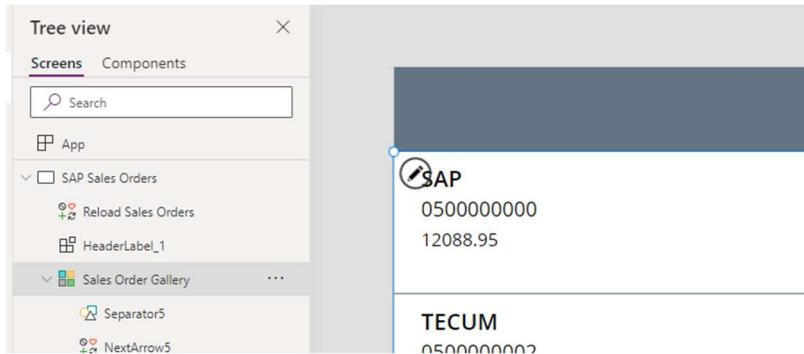
### Tips on working with galleries:

Galleries provide a powerful way to visualize tabular data in Power Apps. It is important to become familiar with customizing a gallery. Key components of a gallery: the gallery control, the template cell (first cell), and controls within the template cell.

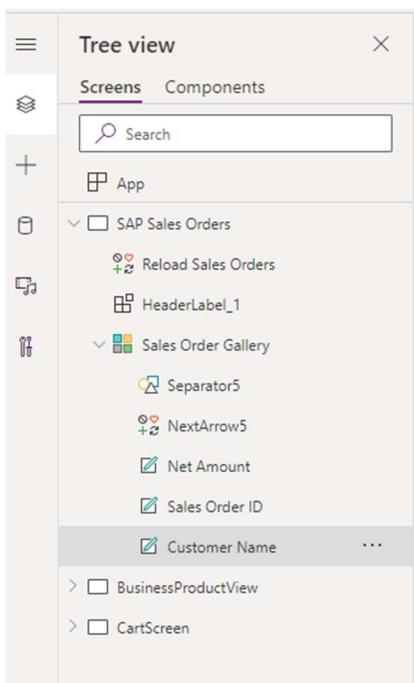
To select the entire gallery – click on the gallery in the tree view on the left or click on the second or third cell. Clicking any cell that is not the first cell of the gallery will select the entire gallery. Now you can specify properties

that apply to the entire gallery, such as the **Items** property which is the data source, the gallery fill color, borders, etc. To customize how each item is displayed in the gallery, you will customize the template cell. Select the template by clicking in the first cell of the gallery or click on the pencil icon in the top left corner when the entire gallery is selected. You can now add, remove and customize the controls within the template cell. These changes will then repeat across each item or row in the table.

13. Adding a description in front of the fields in the gallery. We will now add a description in front of our fields. Select the **template cell** by clicking in the **first cell of the gallery** or click on the **pencil icon** in the top left corner when the entire gallery is selected.

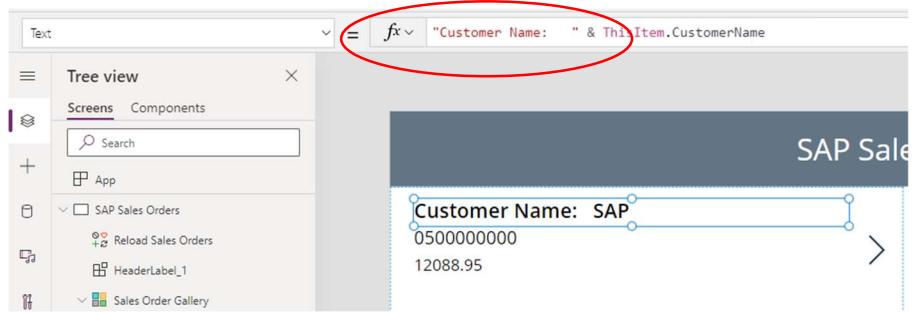


14. In the left-hand pane under **Tree View**, select the **Customer Name** label.



15. Edit the Text Formula bar. In the ribbon, the **Text formula bar** is currently displaying the customer name from the SAP Data source. Replace the text below in the formula bar to add a description for the field.

**"Customer Name: " & ThisItem.CustomerName**



The “&” sign combines both the text between the quotes and the data from data source.

- Follow the same steps and replace the text in the **Text Formula bar** with the following for each label:

Net Amount: **Text(Value(ThisItem.NetAmount),"#,###.##")**

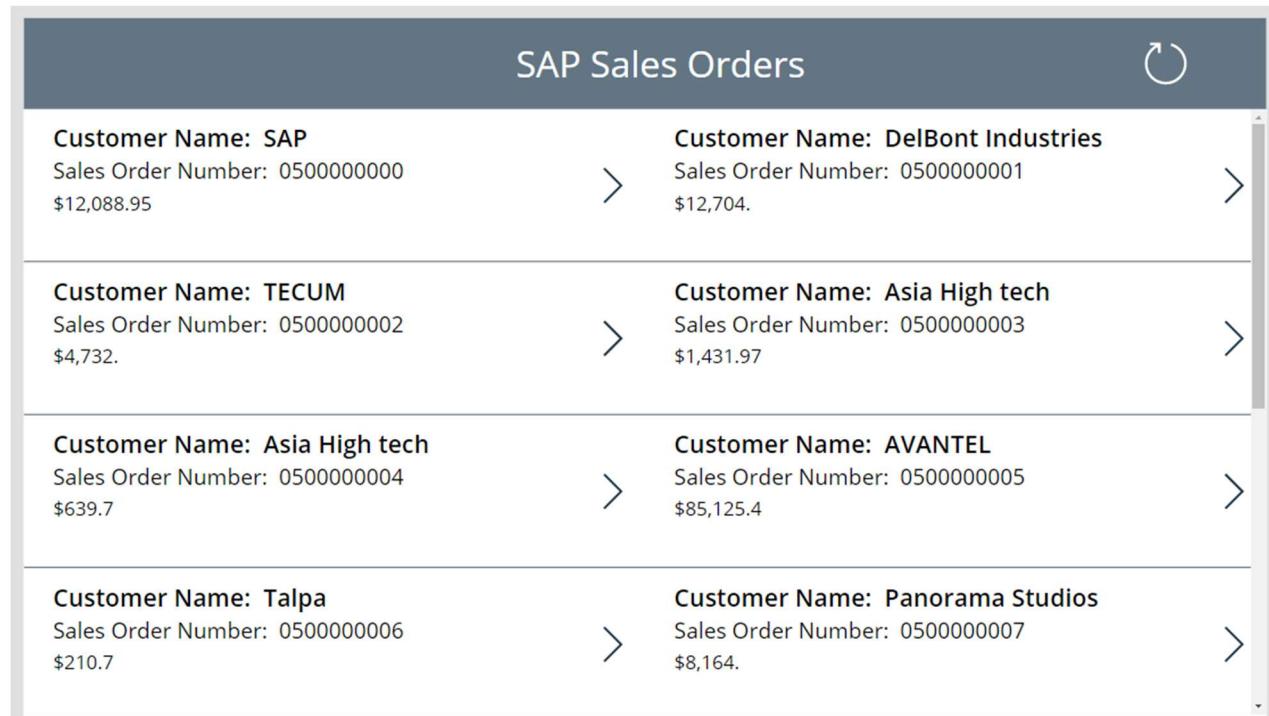


**Note:** Since SAP returns the NetAmount as a string, we need to use the Value function to convert to a Number to get the right format.

Sales Order ID: **"Sales Order Number: " & ThisItem.SalesOrderID**



Your screen should now look like this



17. Save your app. In the ribbon, click on **File** and **Save**. Click on the **back arrow** to return to editing.

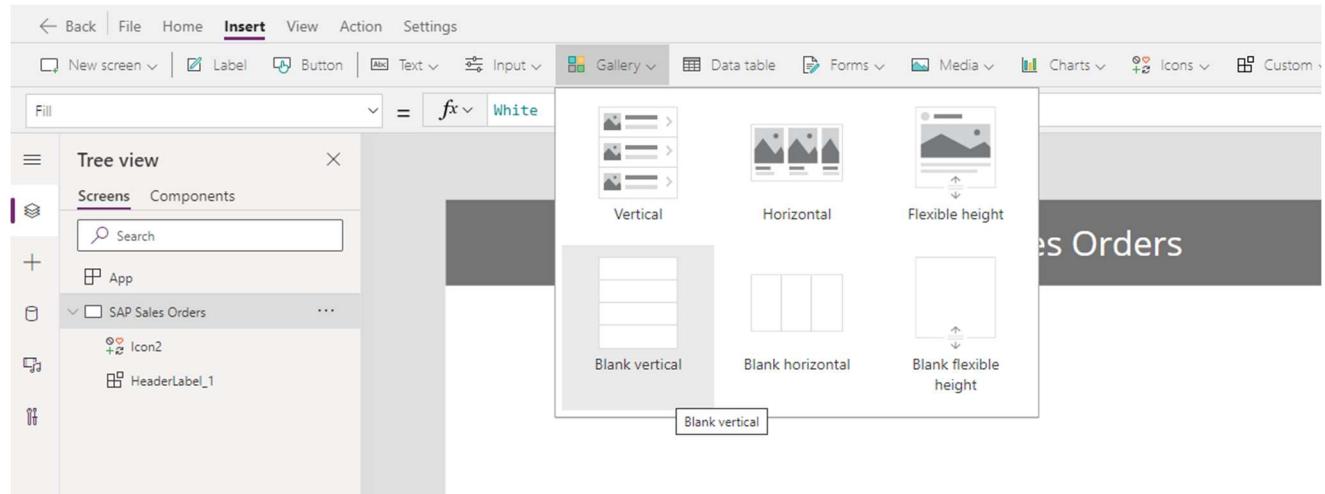
Currently, the Next Arrow icon  does not do anything on this screen. As an option, you could have it navigate to a new screen that would show the order details or you could remove it from the screen by deleting the icon in Tree View.

For this lab, we will be adding a button at the top of the screen to create a new Sales Order in the next exercise.

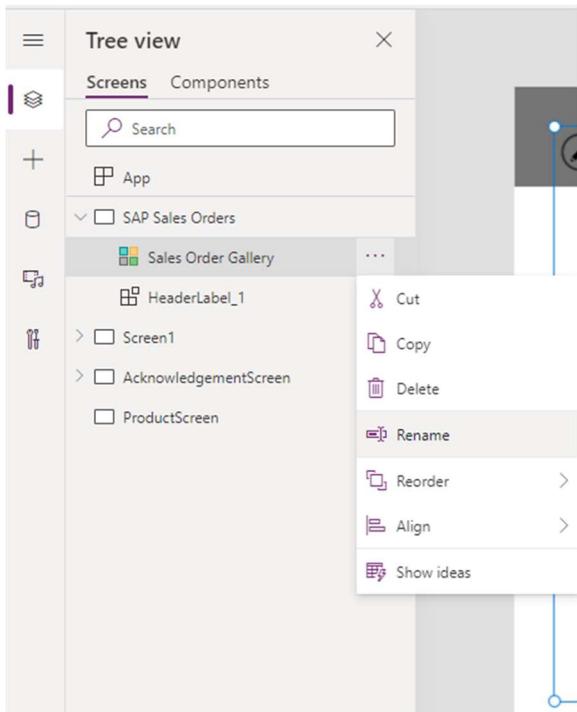
**Congratulations!** You have created an app that displays SAP Sales Orders! You can proceed to Exercise 3 for building the next screen.

### Option # 2: Adding a blank gallery and using a container to display Sales Orders.

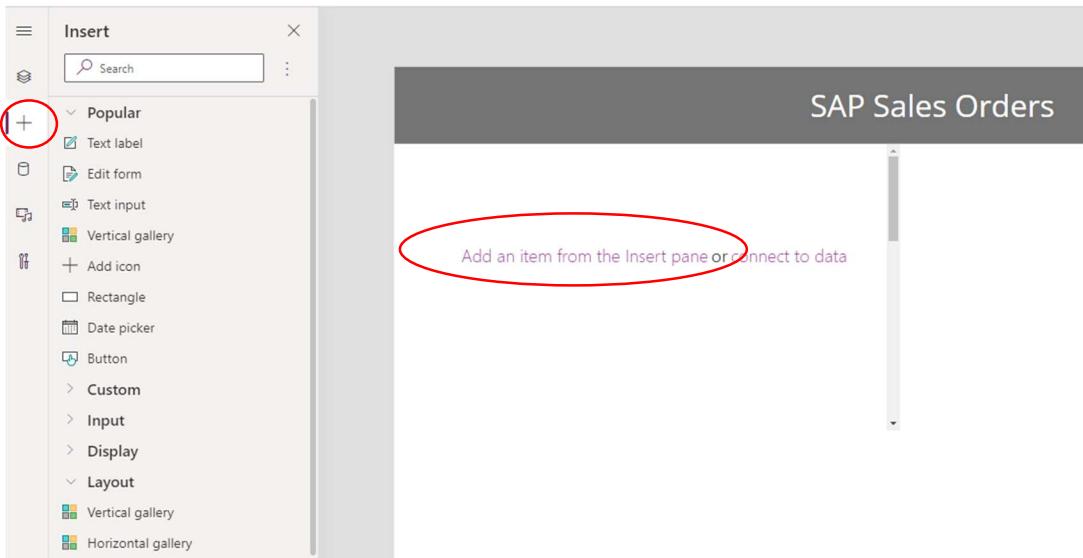
1. Add a new gallery to display the collection of Sales Orders. In the left-hand pane, select **Tree View** and click on **SAP Sales Orders** screen. In the Ribbon, select **Insert** and then click on **Gallery**. Select the **Blank Vertical** gallery.



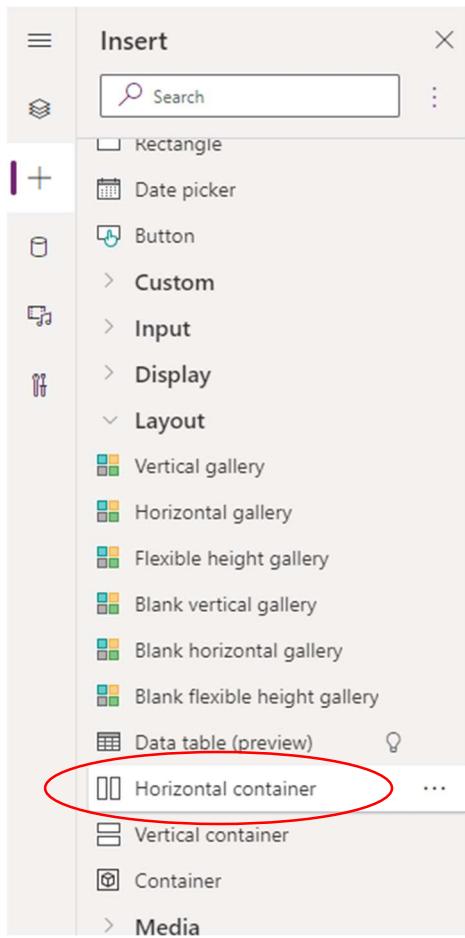
2. Rename the Gallery. In the left-hand pane, select the **Tree View** and click on the newly inserted **Gallery**. Click on the “...” and select **Rename**. Rename the gallery to **Sales Order Gallery**.



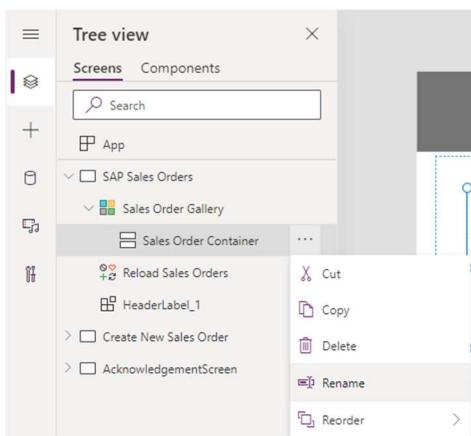
3. Insert Container. In the text in the gallery, click on **Add an item from the Insert pane**. Or you can click on **Insert** in the left-hand navigation pane. Confirm you have the **Sales Order Gallery** selected before you click on the **Insert** tab or the container will be outside the gallery.



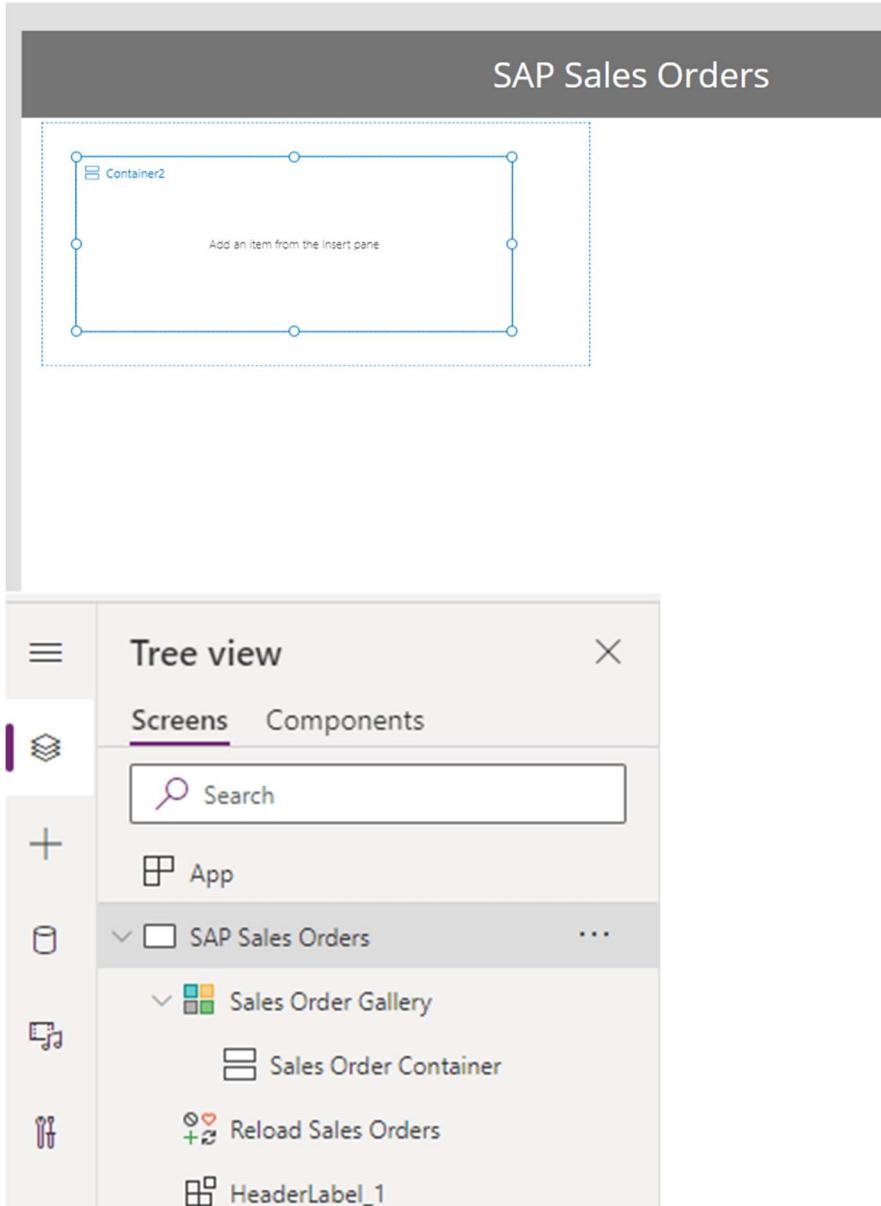
4. Select a Horizontal Container. Scroll down and expand the **Layout** section. Select **Horizontal Container**.



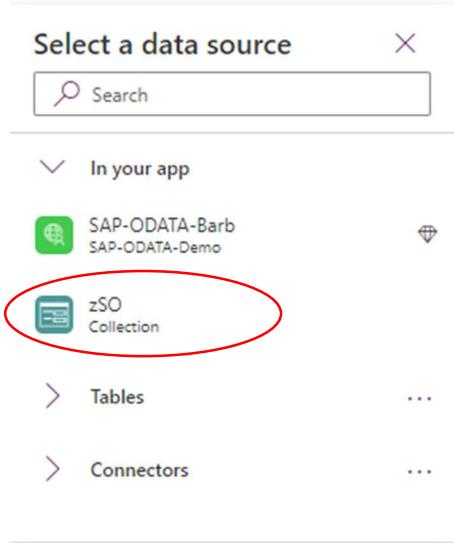
5. Rename the Container. In the left-hand navigation pane in **Tree View**, select the **Container**. Click on the "...", select **Rename**, and rename to **Sales Order Container**.



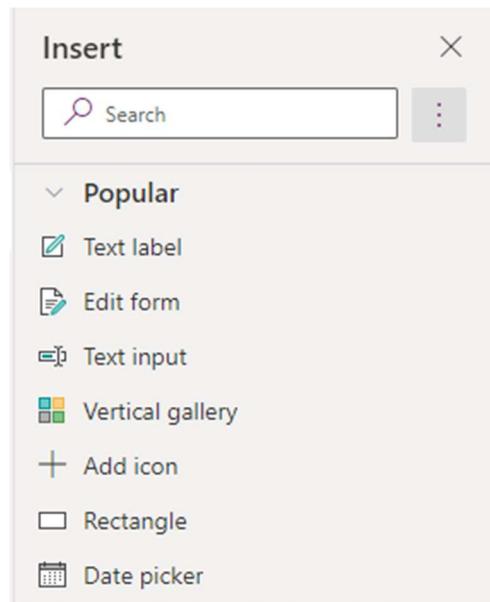
6. Confirm the Container is inside the gallery by looking at the canvas or verifying in the **Tree View**.



7. Connect to data. In the left-hand pane under **Tree view**, select the **Sales Order Gallery**. Select **zSO Collection** as the data source.



8. Add text labels for the data. In the left-hand navigation, select **Tree view** and select the **Sales Order Container**. Click on the **Insert** tab.
9. From the **Insert pane**, click on **Text Label** (5) times to insert five **Text Labels** into the Sales Order Container.

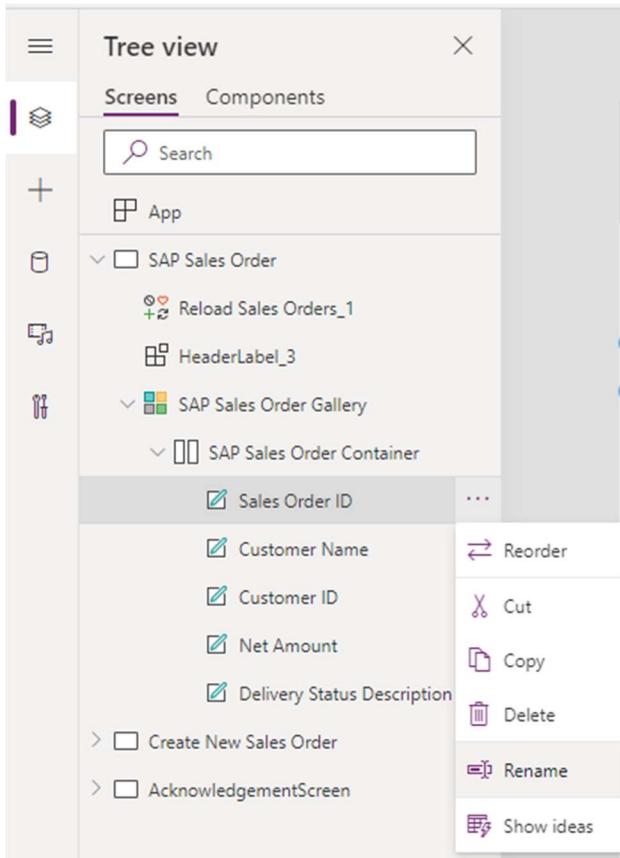


10. Rename the Text Labels. In the left-hand navigation pane in **Tree view**, select the **Sales Order Container**. Click on the "..." and select **Rename**. Rename the ( 5 ) **Text Labels** as listed below by

Label 5 – Sales Order ID  
Label 4 – Company  
Label 3 – Customer ID

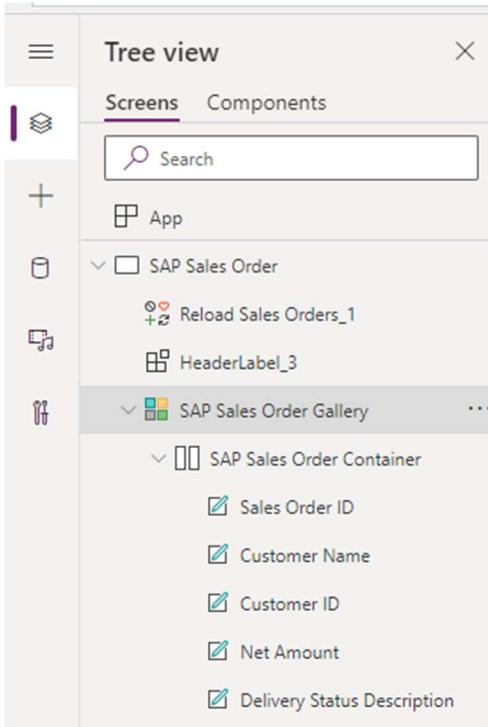
Label 2 – Net Amount

Label 1 – Delivery Status

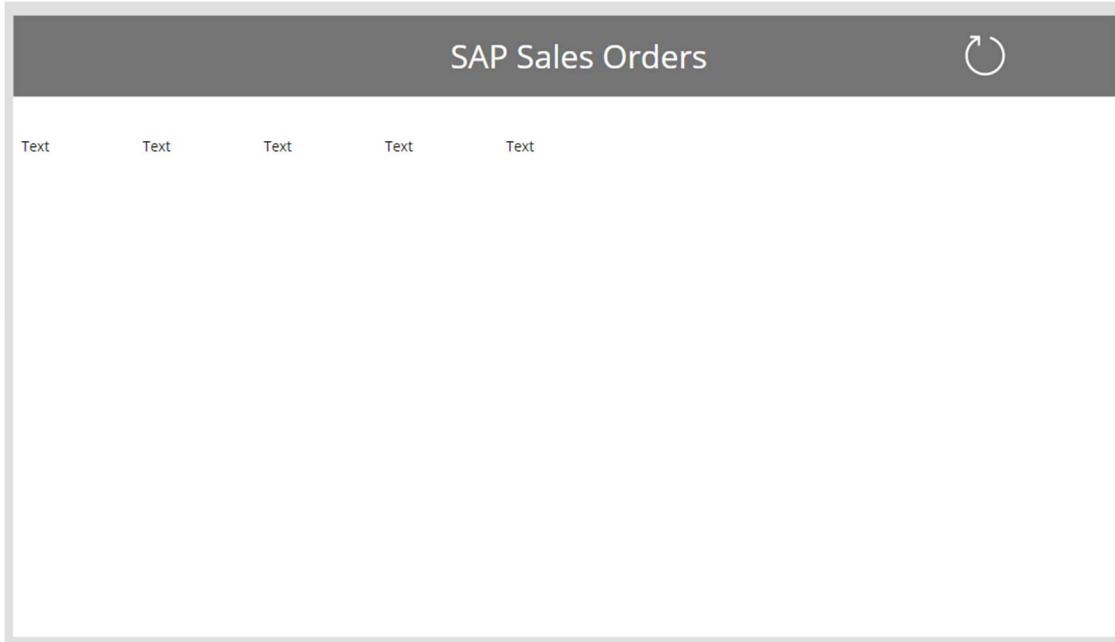


**TIP:** When working with a canvas that has many items, use the Tree View to select the Gallery or any other items that may be hard to select on the canvas

11. Position the gallery. In the left-hand pane in **Tree View**, select the **Sales Order** gallery.



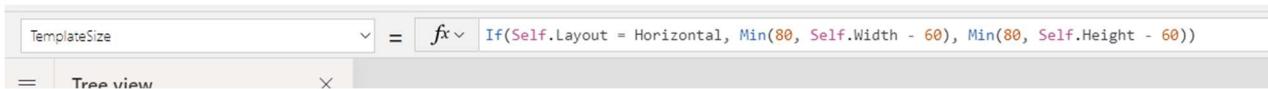
12. Drag the gallery down under the **HeaderLabel** and stretch width across the entire canvas. Keep room under the **HeaderLabel** for more description labels for each column.



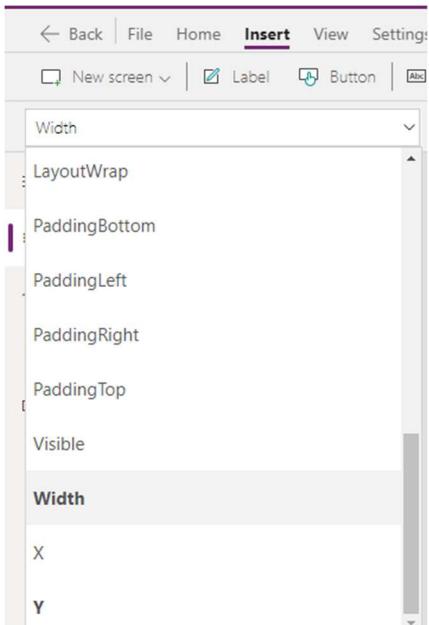
13. Set the Height of the Gallery Template. Keep the **Sales Order Gallery** selected. In the **formula bar**, use the drop down to select **Template Size**. Cut and paste below for the formula to change the Width to 80.

```
If(Self.Layout = Layout.Horizontal, Min(80, Self.Width - 60), Min(80, Self.Height - 60))
```

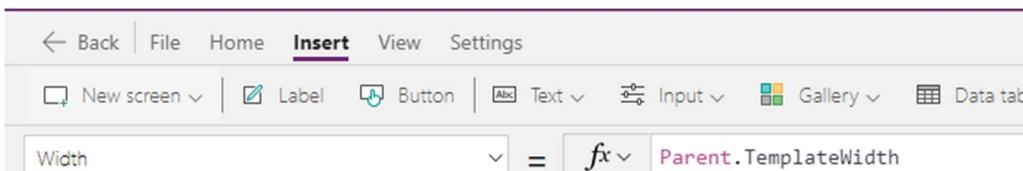
## Leveraging Power Apps with SAP on Azure Workshop



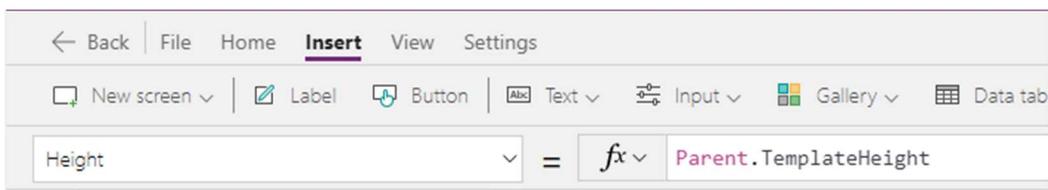
14. Set Width of the Container. Now we want to set the width of the container to be matched to the gallery. In the left-hand navigation pane under **Tree view**, select the **Sales Order Container**. On the ribbon on the **formula bar**, click on the drop down and scroll down to **Width**.



Set the formula to **Parent.TemplateWidth**

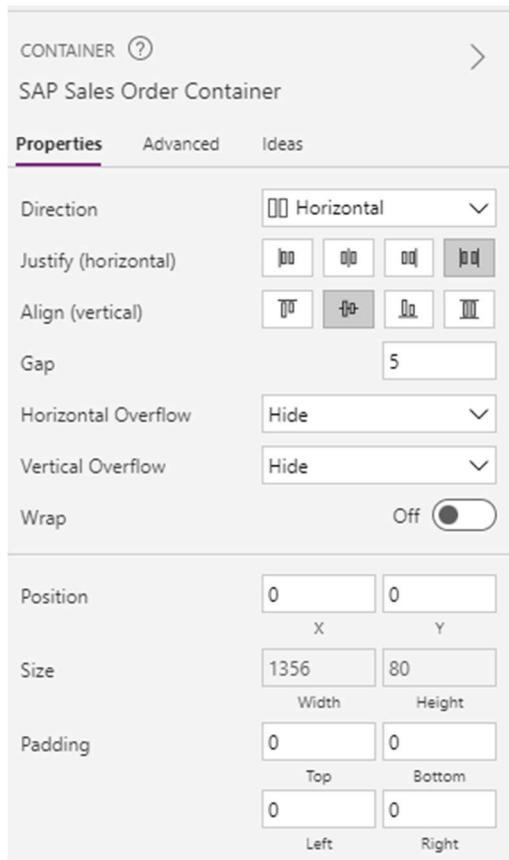


15. Set the Height of the Container. In the same area, select **Height** in the drop down and set the **formula** to **Parent.TemplateHeight**.



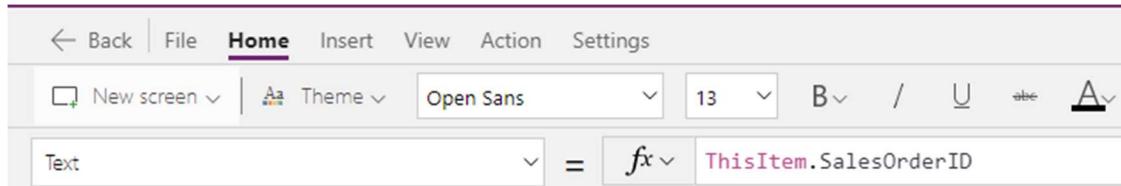
16. Move the Text Labels. Now we want to spread the text labels across the container. Keep the **Sales Order Container** still selected. In the right-hand pane under **Properties**, select **Space Between** under **Justify (horizontal)**.
17. Additional **Properties** for the **Sales Order Container**.
- For **Align (vertical)**, select the second option – **Center**.

- To have a small space between each column, change **Gap** to **5**.
- For **position**, select **0** for **X** and **0** for **Y**.



18. Connect the Text fields to data from the SAP Data source. Now we will bind the text labels to specific fields from the SAP Data source. In the left-hand pane in **Tree view**, select the **Sales Order ID text box** in the **SAP Sales Order Container**. In the ribbon, cut and paste the text below in the **Text formula box**.

**ThisItem.SalesOrderID**



Follow the same steps for the rest of the Text labels.

Customer Name Text Formula – **ThisItem.CustomerName**

Customer ID Text Formula – **ThisItem.CustomerID**

Net Amount Text Formula – **Text(Value(ThisItem.NetAmount), "#,###.##")**

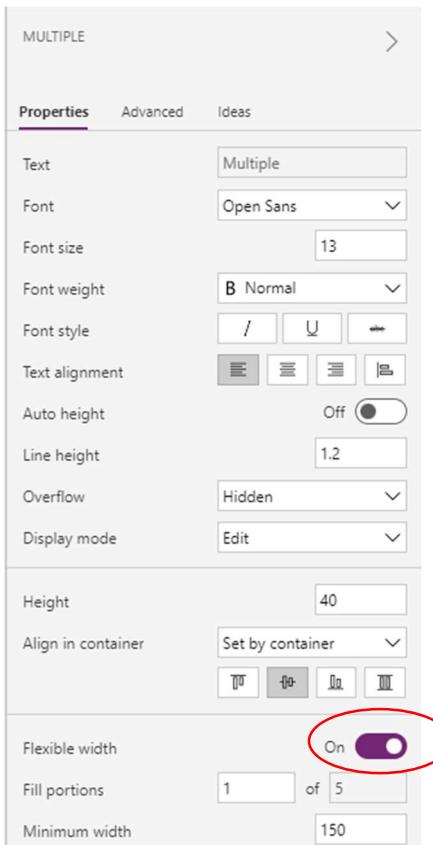
**Note:** Since SAP returns the NetAmount as a String, we need to use the Value function to convert the String to a Number to get the right format.

Delivery Status Description Text Formula – **ThisItem.DeliveryStatusDescription**

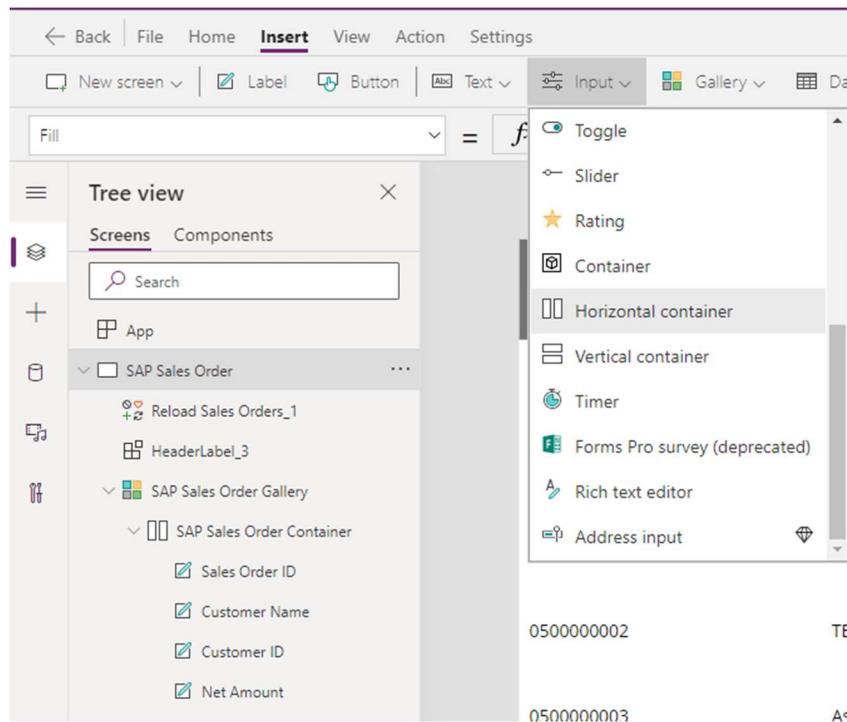
Your screen should now look like the screenshot below.

SAP Sales Orders				
0500000000	SAP	0100000000	\$12,088.95	Delivered
0500000001	DelBont Industries	0100000002	\$12,704.	Delivered
0500000002	TECUM	0100000005	\$4,732.	Delivered
0500000003	Asia High tech	0100000006	\$1,431.97	Delivered
0500000004	Asia High tech	0100000006	\$639.7	Delivered
0500000005	AVANTEL	0100000008	\$85,125.4	Delivered
0500000006	Talpa	0100000003	\$210.7	Delivered

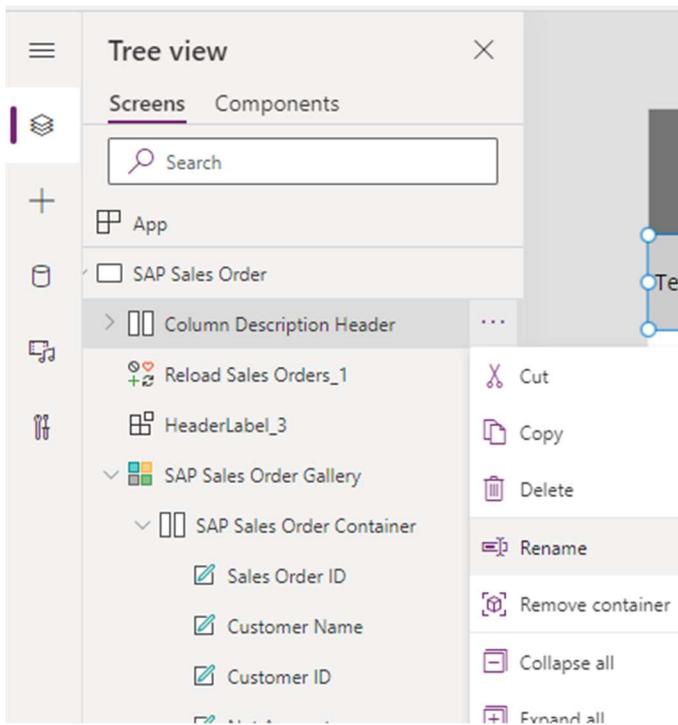
19. Change the properties of the width to be flexible and fill out the entire length of the screen. Select **all** the Text Labels in the **SAP Sales Order Container** by holding down the **Ctrl key** and selecting each one. In the right-hand pane under **Properties**, turn on the **Flexible width** property.



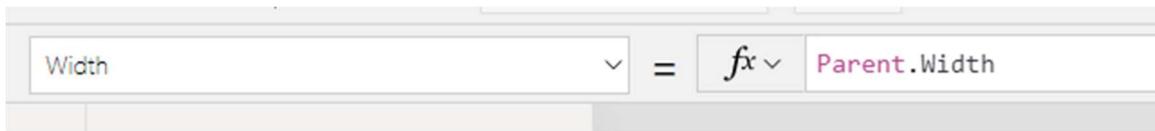
20. Create Description Labels for above the SAP Sales Order Gallery. Now we will add another container and text labels above the gallery to describe the content in each column. In the left-hand navigation pane, select the **SAP Sales Order screen**. In the ribbon, click on **Insert** and then **Input**. Scroll down and select **Horizontal Container**.



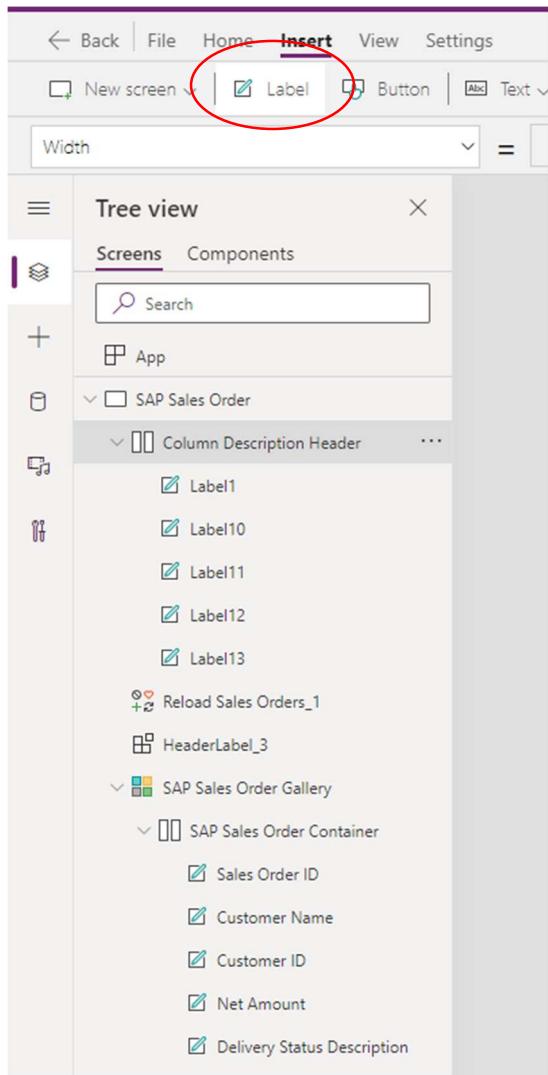
21. Rename the container. In the left-hand navigation pane in **Tree view**, select the **new container**. Click on the “...” and select **Rename**. Rename the container to **Column Description Header**.



22. Position the container. You can drag the container under the Header Label and stretch it out to meet the top of the gallery. In the **ribbon**, select the **Width** and change the formula to **Parent.Width**.



23. Add text labels for column descriptions. In the left-hand navigation pane in **Tree view**, select the **Column Description Header**. In the ribbon, select the **Insert** tab. Click on **Label** (5) times to insert (5) text labels.



24. Rename the Text Labels. Select each label and click on the “....”. Click on **Rename** and rename as listed below.

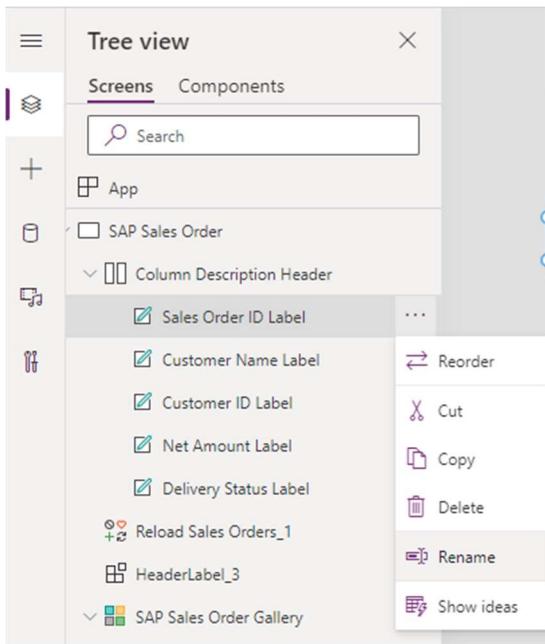
Label 1: **Sales Order ID Label**

Label 2: **Customer Name Label**

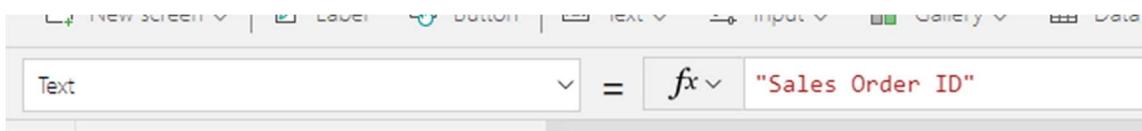
Label 3: **Customer ID Label**

Label 4: **Net Amount Label**

Label 5: **Delivery Status Label**



25. Change the text displayed for the column headings. In the left-hand navigation pane in **Tree view**, select the **Sales Order ID Label** in the **Column Description Header container**. In the ribbon, change the **Text formula** to – **“Sales Order ID”**



Using the same steps above, select the other text labels and change the Text formula as listed below:

Customer Name Label – **“Customer Name”**

Customer ID Label – **“Customer ID”**

Net Amount Label – **“Net Amount”**

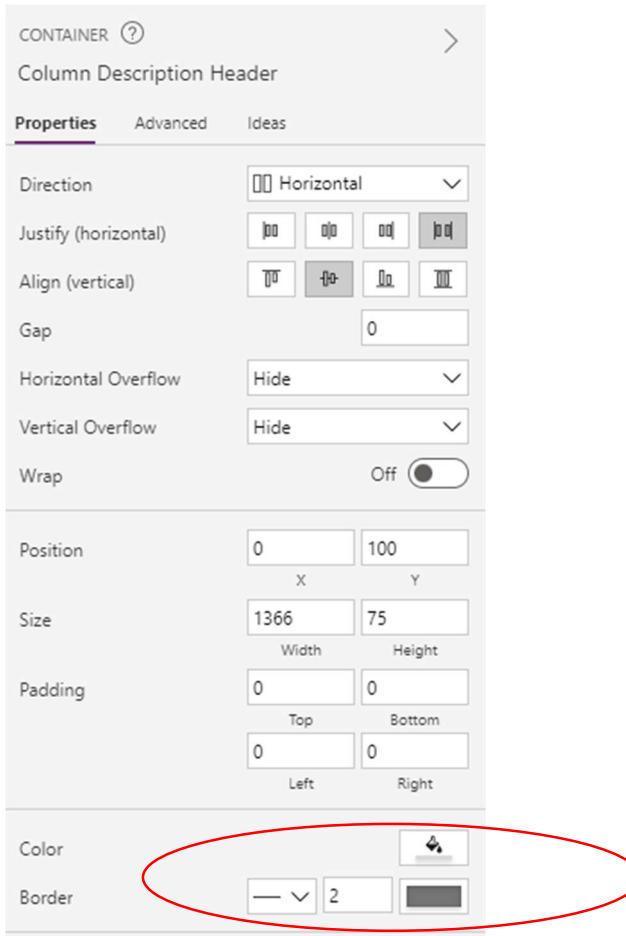
Delivery Status Label – **“Delivery Status”**

Your screen should now look like the screen shot below.

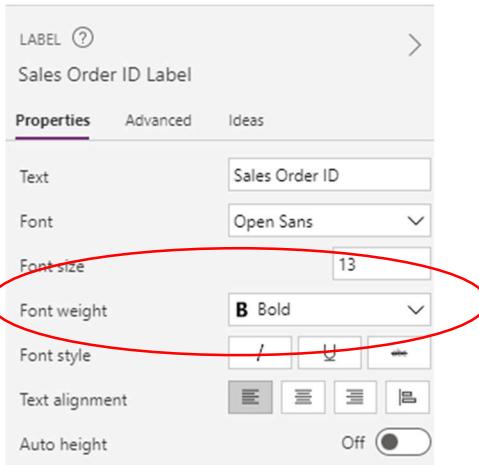
SAP Sales Orders				
Sales Order ID	Customer Name	Customer ID	Net Amount	Delivery Status
0500000000	SAP	0100000000	\$12,088.95	Delivered
0500000001	DelBont Industries	0100000002	\$12,704.	Delivered
0500000002	TECUM	0100000005	\$4,732.	Delivered
0500000003	Asia High tech	0100000006	\$1,431.97	Delivered
0500000004	Asia High tech	0100000006	\$639.7	Delivered
0500000005	AVANTEL	0100000008	\$85,125.4	Delivered
0500000006	Talpa	0100000003	\$210.7	Delivered

26. Add Fill into container and modify text style. To set the head label apart from the gallery, we can add a fill and change the font style. In the left-hand navigation pane in **Tree view**, select the **Column Description Header**. In the right-hand pane under Properties change the following.

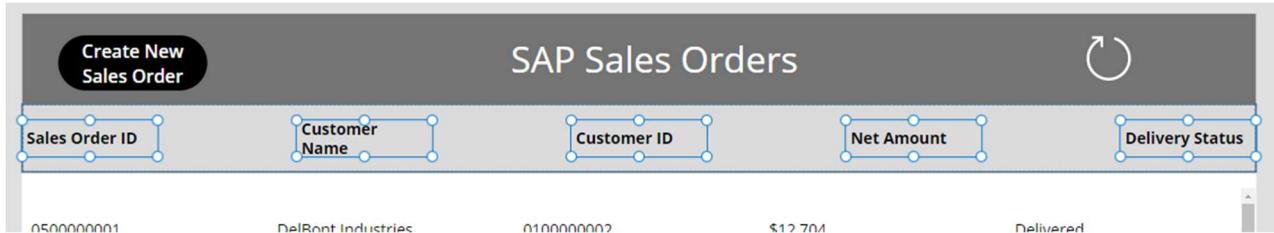
- Change **Color** to **Light Grey**
- Change **Border** size to **2** and **color** to the same **dark grey** as the main header label.



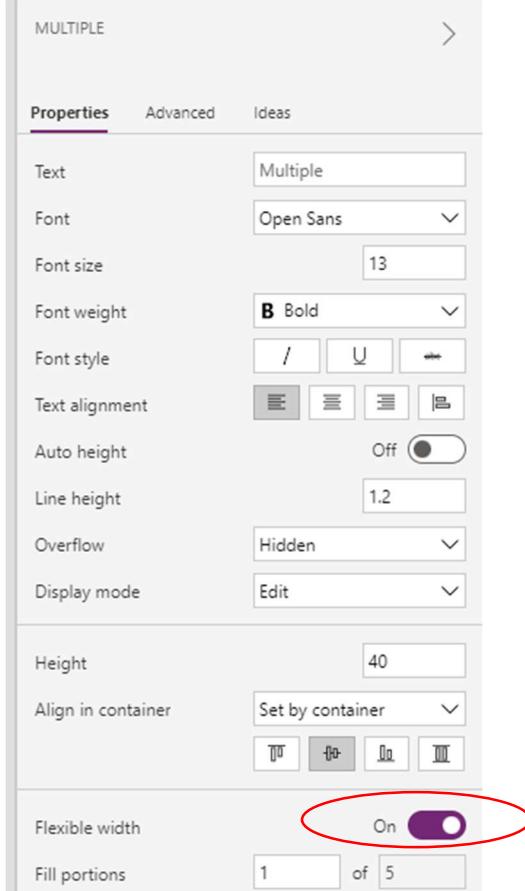
Change the font of the text to bold. In the left-hand navigation pane under **Tree view**, select the **Sales Order ID Label**. In the right-hand pane under **Properties**, change **Font Weight** to **Bold**. Repeat these steps for the (4) other text labels in the **Column Description Header** container.



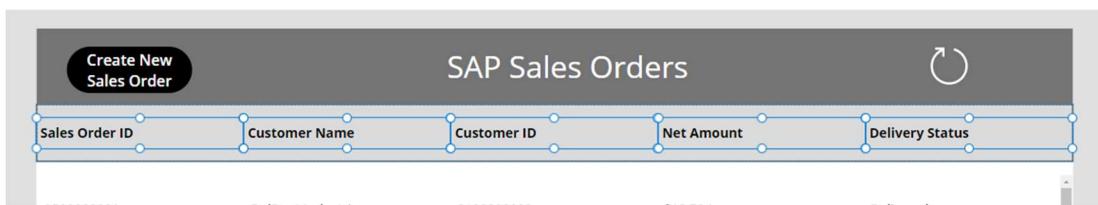
27. Change the properties of the width to be flexible and fill out the entire length of the screen. Select all the **Text Labels** in the container by holding down the **Ctrl key** and **selecting each one**.



In the right-hand pane under **Properties**, turn on the **Flexible Width** option.



Now all the text labels should be stretched across the length of the screen.



28. Save your app. In the ribbon, click on **File** and then **Save**. Click on the **back arrow** to return to editing.

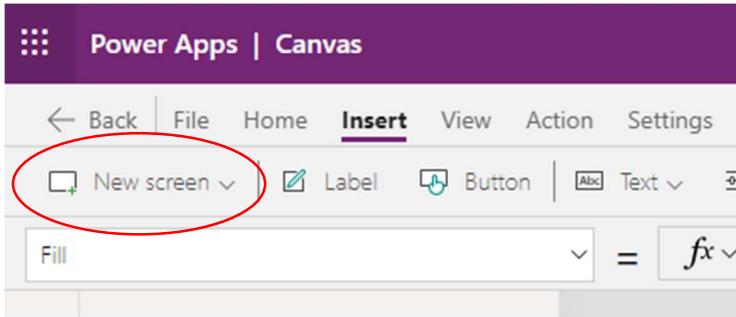
**CONGRATULATIONS!!** You created an app that shows a list of Sales Orders from an SAP system.

# Exercise 3: Add a new screen to show products from SAP.

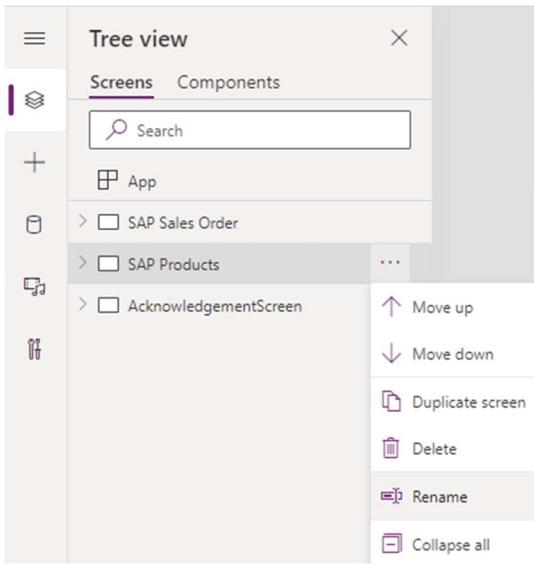
In this exercise, you will add a new screen to show the products from the SAP Demo system.

## Task 1: Add a new screen

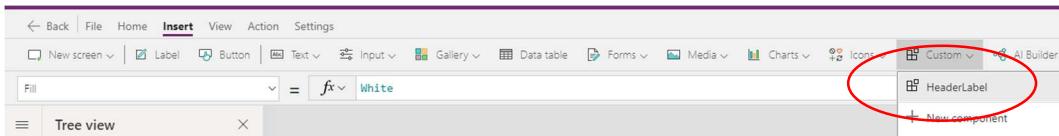
1. Add a new screen. In the **ribbon**, click on **Insert** and then **New Screen**.



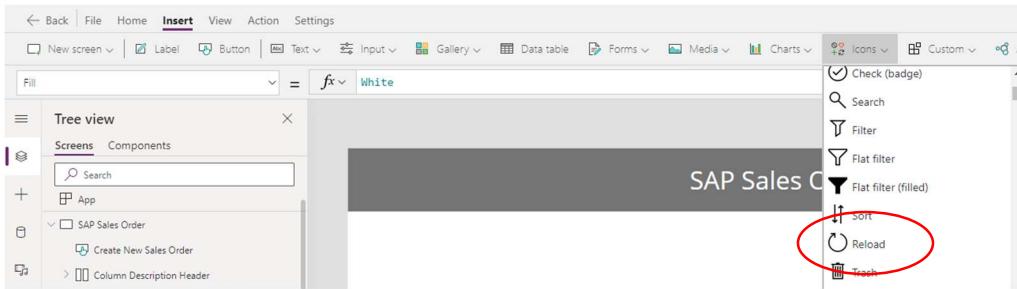
2. Rename the screen. In the left-hand navigation pane in **Tree view**, select the new screen. Click on the “...” and select **Rename**. Rename the screen **SAP Products**.



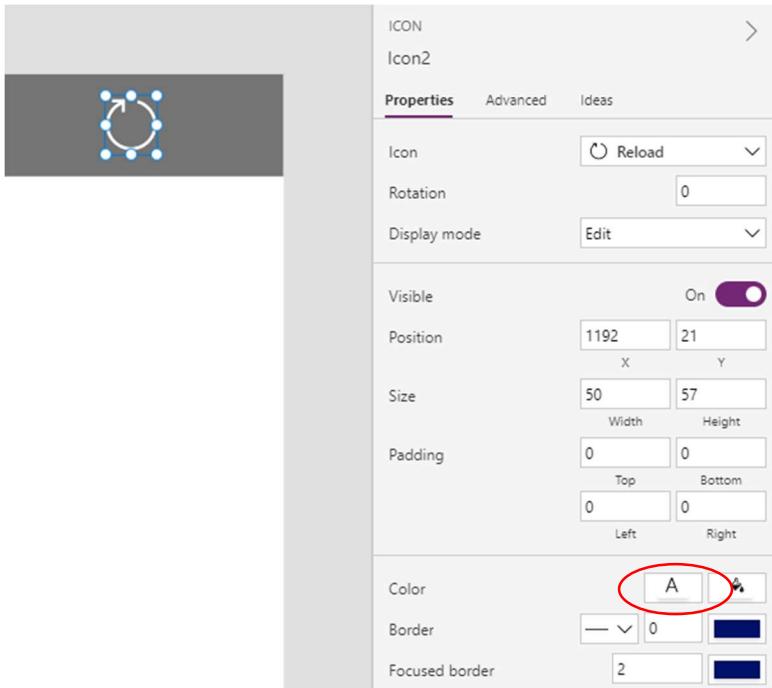
3. Insert Header. In the **ribbon**, click on **Insert** and the **Custom**. Select the **HeaderLabel**.



4. Add an icon to connect and re-load the product data. In the **ribbon** select **Insert** and then **Icons**. Scroll down and select the **Reload Icon**.



- Move and change the Icon color. **Drap and drop the Reload icon** to the right side of the header screen and resize if needed. In the right-hand pane under **Properties**, change the **Color** to **White**.



**TIP:** When you are moving items around, you can see the guide lines to help you position your item just like in PowerPoint

- Connect to a collection of products from the SAP datasource. We will set up a new collection called zProducts and grab the first 30 products for the collection for this demo. In the **ribbon** for the **OnSelect** property, replace the formula with the following, updating the highlighted area with your connection name.

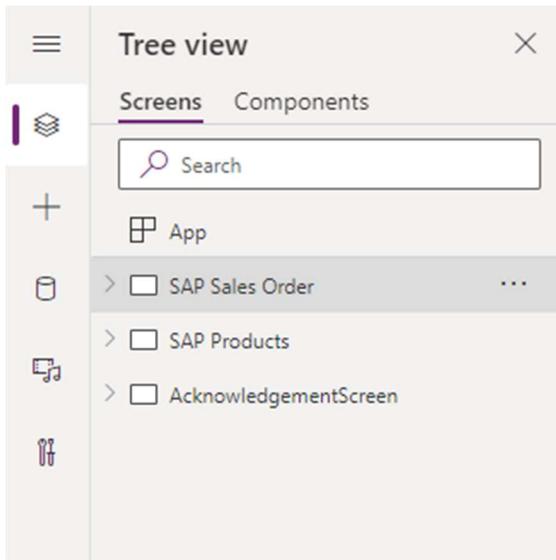
**ClearCollect(zProducts, 'SAP-ODATA-YOURNAME'.ListProductSets({'\$top':30}).d.results)**



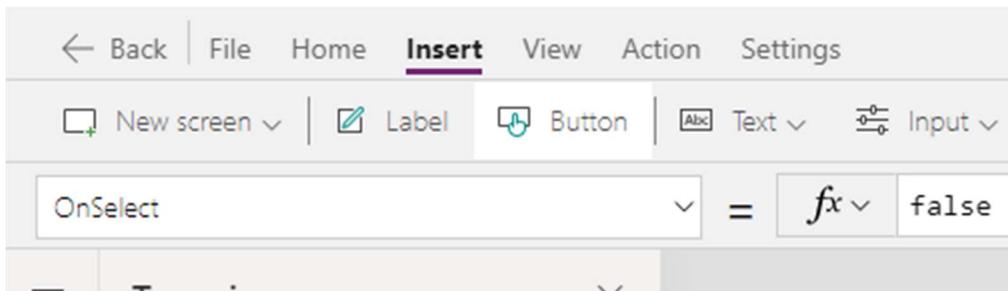
## Task 2: Set up the navigation between screens

## Leveraging Power Apps with SAP on Azure Workshop

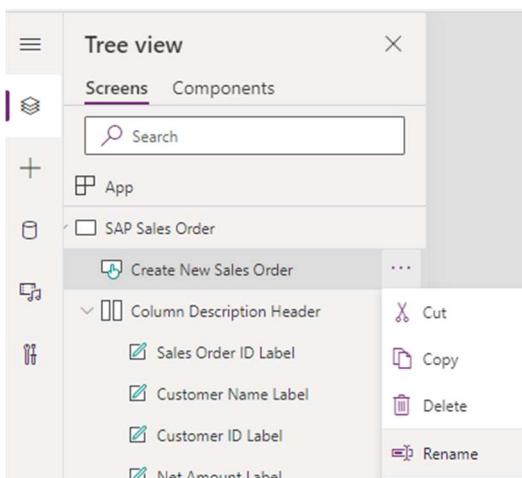
1. Add navigation to the new SAP Products screen. We need to add a button from our SAP Sales Order screen that will navigate to our SAP Products screen. In the left-hand navigation pane in **Tree view**, select the **SAP Sales Order Screen**.



2. Insert a button. In the ribbon, select **Insert** and then click on **Button**.

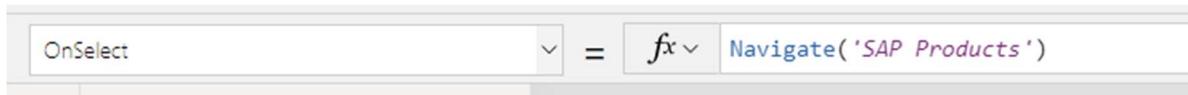


3. Rename button. In the left-hand navigation in **Tree view**, select the newly inserted **Button** on the **SAP Sales Order screen**. Click on the “...” and select **Rename**. Rename button to **Create New Sales Order**.

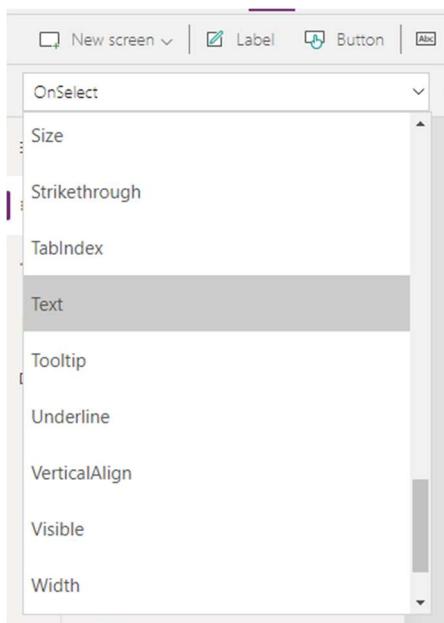


- Set up navigation from SAP Sales Order screen to SAP Products screen. In the left-hand navigation under **Tree view**, select the **Create New Sales Order button** on the **SAP Sales Order screen**. In the OnSelect formula bar in the ribbon, replace the text in the formula bar to:

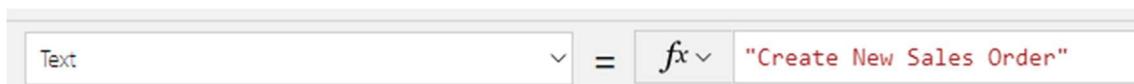
**Navigate('SAP Products')**



- Change the text in the button on the screen. With the **Create New Sales Order** button still selected, click on the **drop down in the ribbon** and scroll down to select the **Text** field.



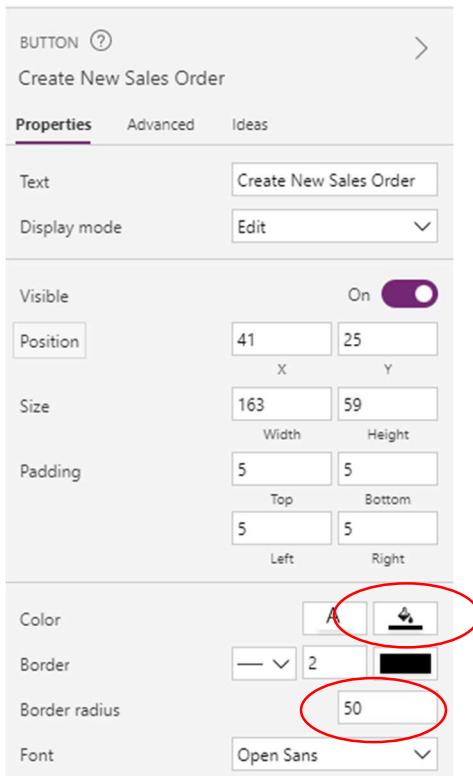
- Change the text in the formula bar from "Button" to "**Create New Sales Order**".



- Change the button properties. We can make the button larger, change the fill color and radius of the button to make it look more modern. With the **Create New Sales Order button** still selected, **drag the corners** of the button to make it larger so you can see all the text.



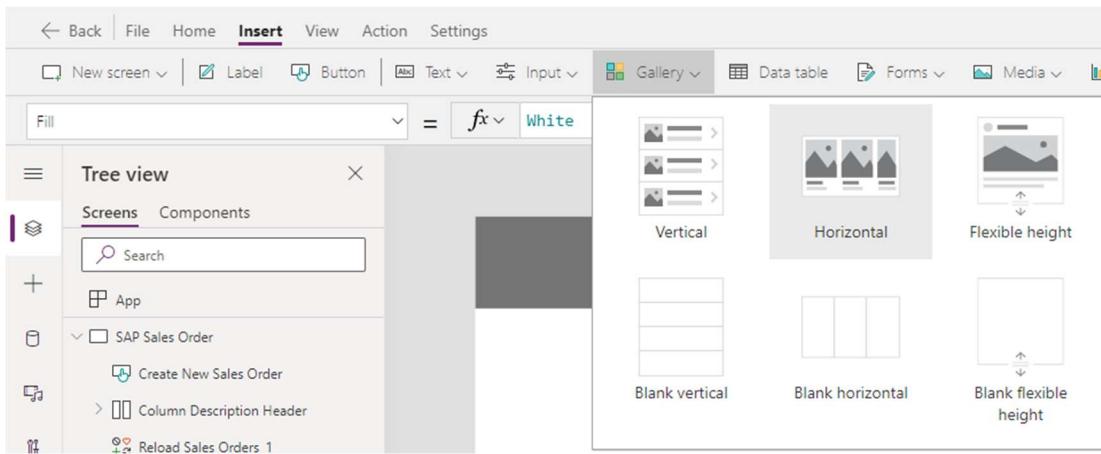
8. In the right-hand pane under **Properties**, change the **Color** to **Black** and the **Border Radius** to **50**.



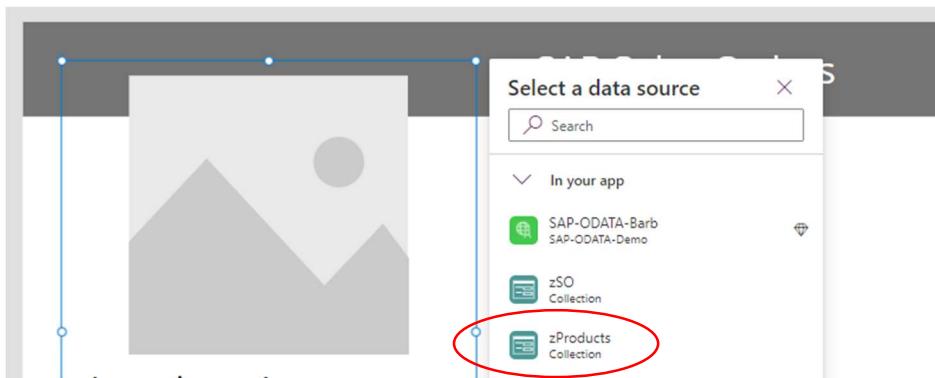
9. Test the navigation. Now we want to test the navigation from the SAP Sales Order Screen to the SAP Products screen. In the **ribbon**, click on the **Play button**. When the application opens, click on the **Create New Sales Order** button and ensure you navigate to the **SAP Products screen** (which just has the header screen right now).
10. Save your app. In the ribbon, click on the **File** and then **Save**. Click on the **back arrow** to continue to edit your app.

## Task 3: Add a gallery for Products to the SAP Products Screen

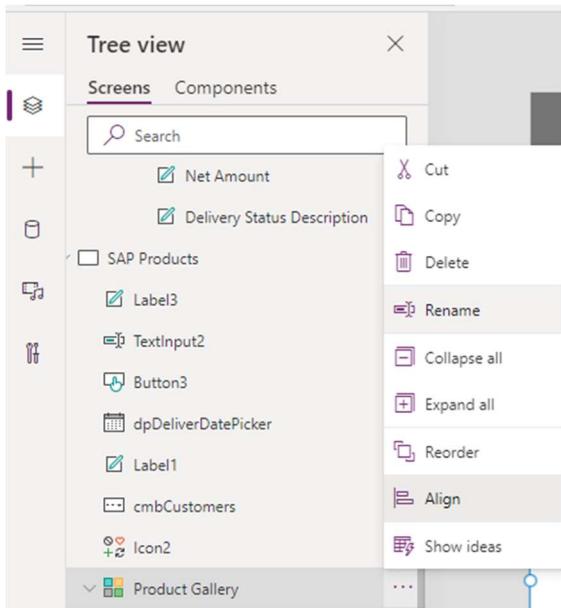
1. We will add a gallery to display products that can be added to sales order. In the left-hand pane in the **Tree view**, select the **SAP Products screen**. In the ribbon, select **Insert** and then **Gallery**. Select the **Horizontal Gallery**.



2. Select the data source. Select the **zProducts Collection** for your data source.

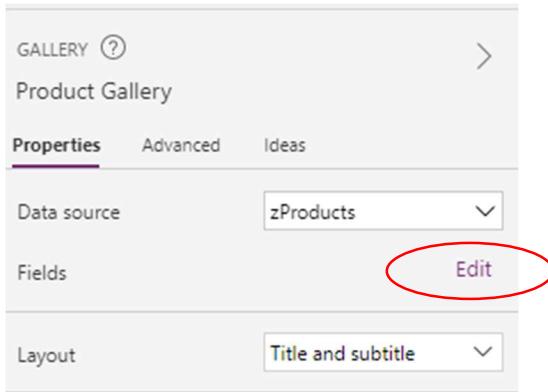


3. Rename the gallery. In the left-hand navigation in **Tree view**, select the new gallery. Click on the “...” and select **Rename**. Rename the gallery **Product Gallery**.



## Leveraging Power Apps with SAP on Azure Workshop

4. Connect the fields in the gallery to the SAP data source. We need to map the correct fields from the SAP data source to the gallery. Ensure the **Product Gallery** still selected. In the right-hand pane under **Properties**, click on **Edit** across from the **Fields** property.



5. Map the fields. Select the **Image field** and cut and paste the information below in the **formula bar**.

`Concatenate("https://sapes5.sapdevcenter.com/sap/public/bc/NWDEMO_MODEL/IMAGES/",ThisItem.ProductID,".jpg")`



Follow the same steps for the Subtitle and Title field:

Subtitle – **Price: "&Text(Value(ThisItem.Price),"\$#,###")**

**Note:** Since SAP returns the NetAmount as a String, we need to use the Value function to convert the String to a Number to get the right format.

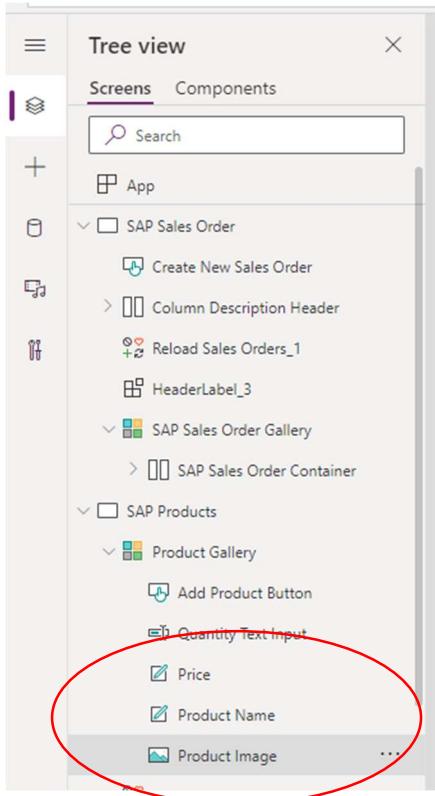
Title – **ThisItem.Name**

6. Rename the Product Gallery fields. In the left-hand navigation pane under **Tree view**, select the **Image field** under the **Product Gallery**. Click on the "... " and then **Rename**. Rename it **Product Image**.

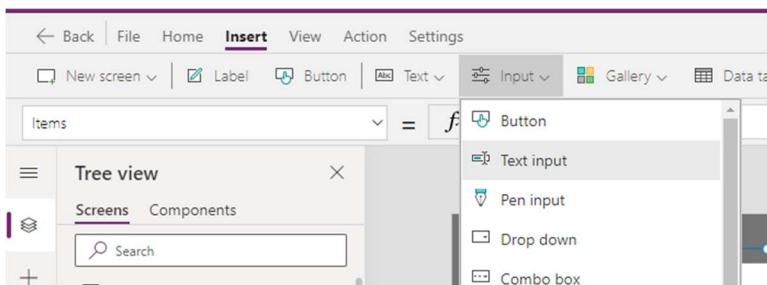
Follow the same steps and rename the other fields in the Product Gallery as listed below:

Subtitle – **Price**

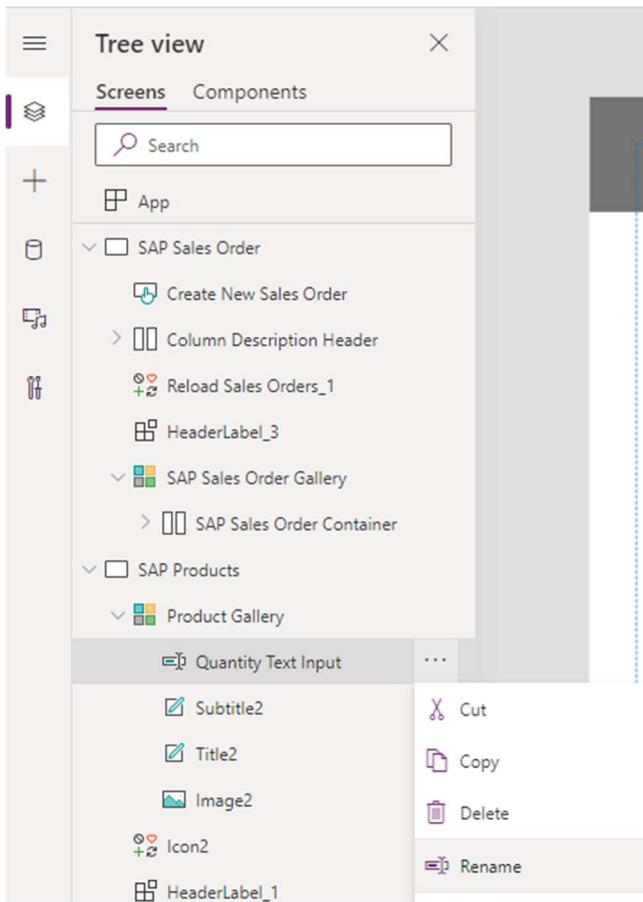
Title – **Product Name**



7. Insert a text input field to indicate how many items you want to purchase. With the **Product Gallery** selected, click the **pencil icon** in the upper right-hand corner to edit the gallery template. The first box of the gallery should be selected. In the **ribbon**, select **Insert** and then **Input** and **Text input**.

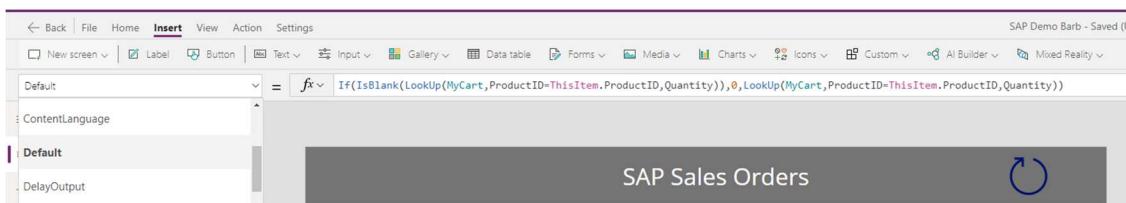


8. Rename the Text input. In the left-hand navigation under **Tree View**, select the **Text input** and then click on "...". Click on **Rename** and rename to **Quantity Text Input**.

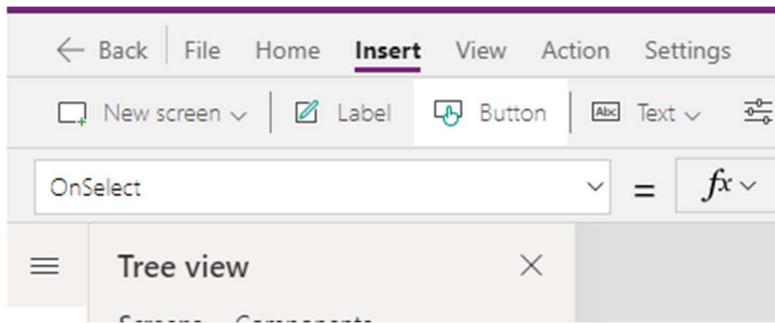


9. Change the default properties of the Text input to show a 0. With the **Text input** still selected, click on the **drop down in the ribbon** for the property and scroll down to select **Default**. Cut and paste the expression below into the formula field

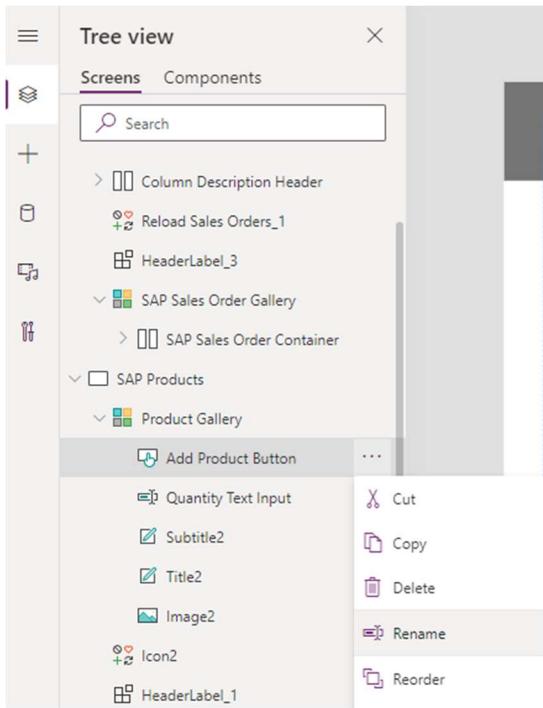
```
If(IsBlank(LookUp(MyCart, ProductID=ThisItem.ProductID, Quantity)),0,LookUp(MyCart, ProductID=ThisItem.ProductID, Quantity))
```



10. Insert a button to add the product to our sales order. In the left-hand navigation in **Tree View**, select the **Product Gallery**. Click on the **pencil icon** to edit the **gallery template**. Click on **Insert** and then **Button**.



11. Rename the button. In the left-hand navigation under **Tree View**, select the **Button** and then click on "...". Click on **Rename** and rename to **Add Product Button**.



12. Set the property of the button to move the item and quantity to the cart that will be passed to SAP with the Power Automate Flow we will build in a later step. With the button still selected, replace the formula for the **OnSelect** property to the following:

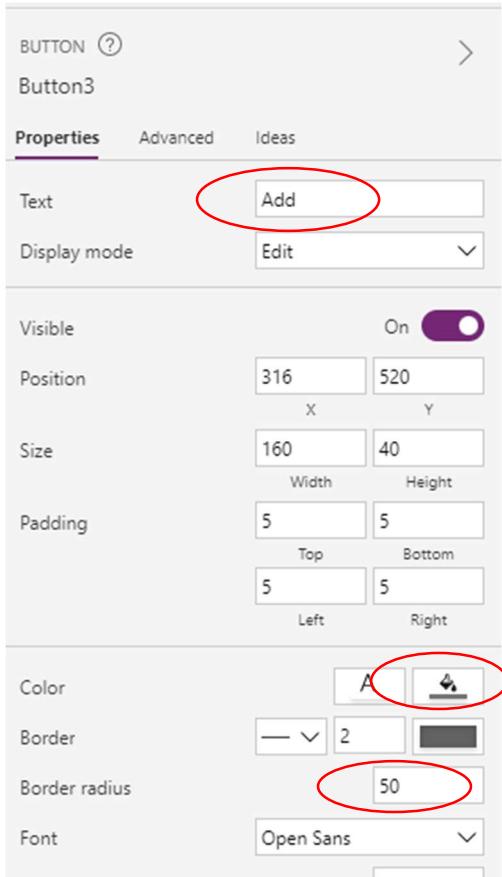
```
Collect(MyCart,{ProductID:ThisItem.ProductID,ProductName:ThisItem.Name,ProductDescription:ThisItem.Description,Quantity:Value('Quantity Text Input'.Text)});Reset('Quantity Text Input')
```

13. Change properties of the button. We will change several properties of the button. Ensure the button is still selected. In the right-hand pane under **Properties**, change the following:

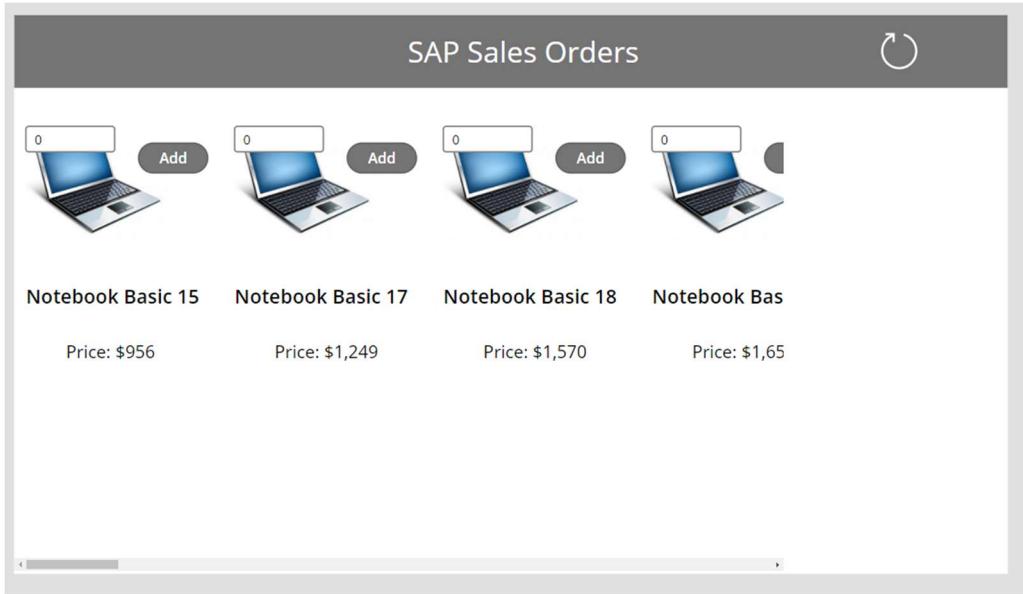
**Text** – change to Add

**Color** – change fill to a dark grey

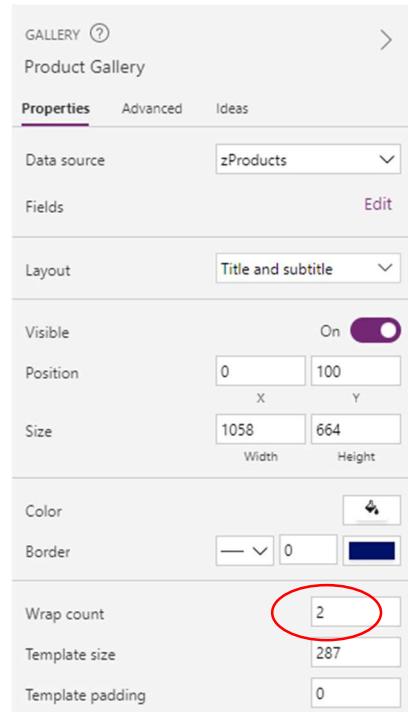
**Border Radius** – change to 50



14. Move the Product Gallery. In the left-hand navigation pane under Tree view, select the Product Gallery. Move the gallery under the header label and stretch to the bottom of the screen. Stretch the gallery  $\frac{3}{4}$  of the way across the canvas leaving room on the side for a search bar and to select the customer and delivery date.



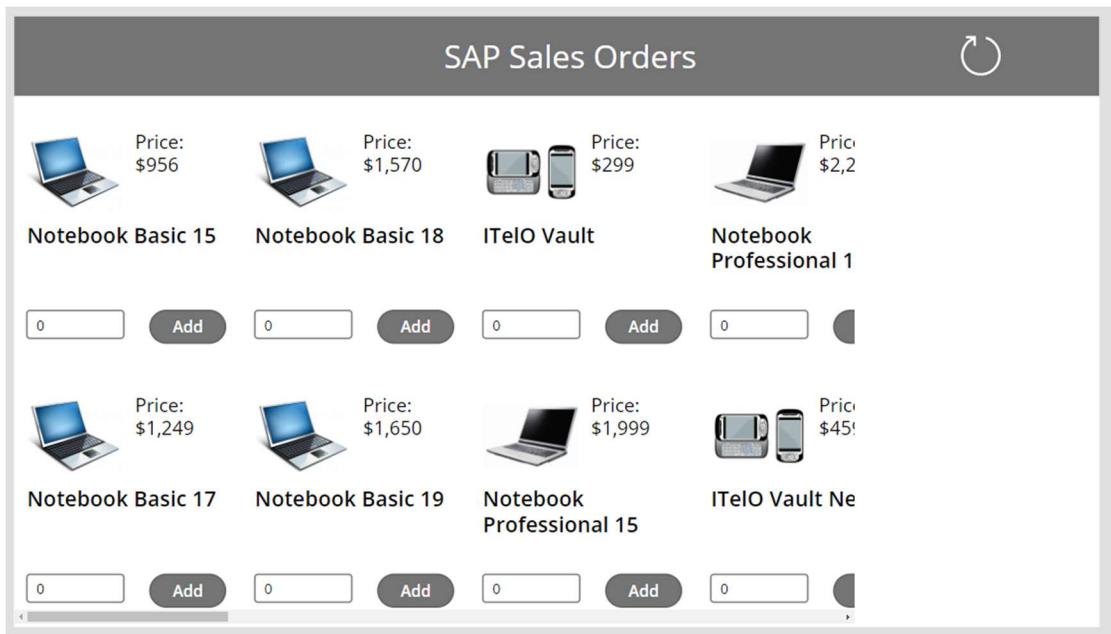
15. Change the gallery to have 2 rows. Ensure the Product Gallery is still selected. In the right-hand pane under Properties, change the Wrap to 2.



16. Move the fields around in the Product Gallery template. In the left-hand navigation pane under **Tree view**, select the **Product Gallery**. Click on the **pencil icon** to modify the gallery template.

- Drag and drop the **Quantity Text Input** and **Add Product Button** to the bottom of the gallery template. Reduce the size of the fields if necessary.
- Select the **Product Image** and reduce the size so there is room for the price next to it.
- Drag and drop the **Price** field next to the image. Stretch the box out so "Price" is on 1 line and the actual price is on the 2<sup>nd</sup> line.
- **Product Name** should be under the picture.

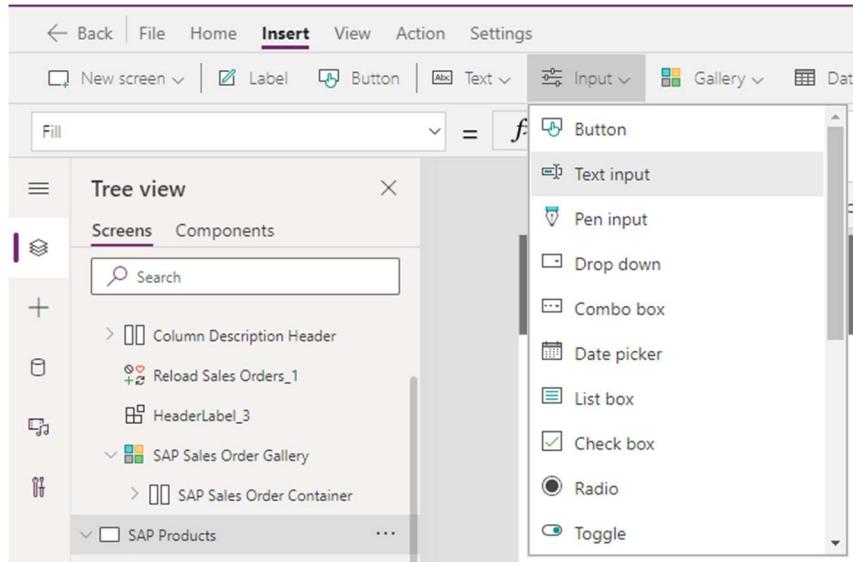
**TIP:** Use the Tree view to select the items in the gallery template.



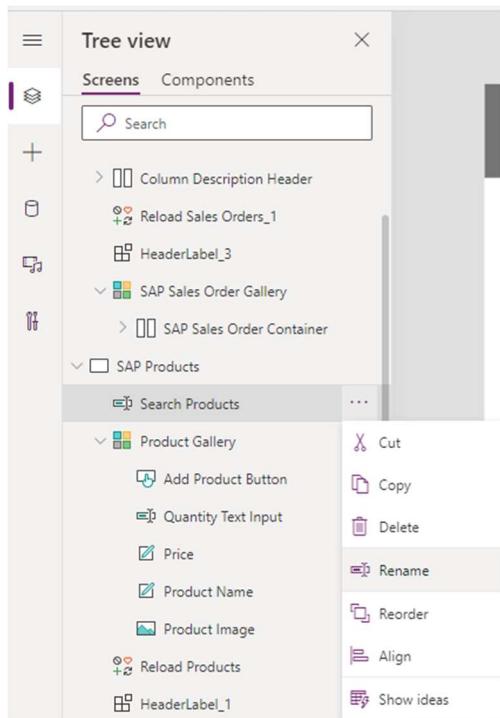
17. Save your app. In the ribbon, click on the **File** and then **Save**. Click on the **back arrow** to continue to edit your app.

## Task 4: Add a Search bar for the Products

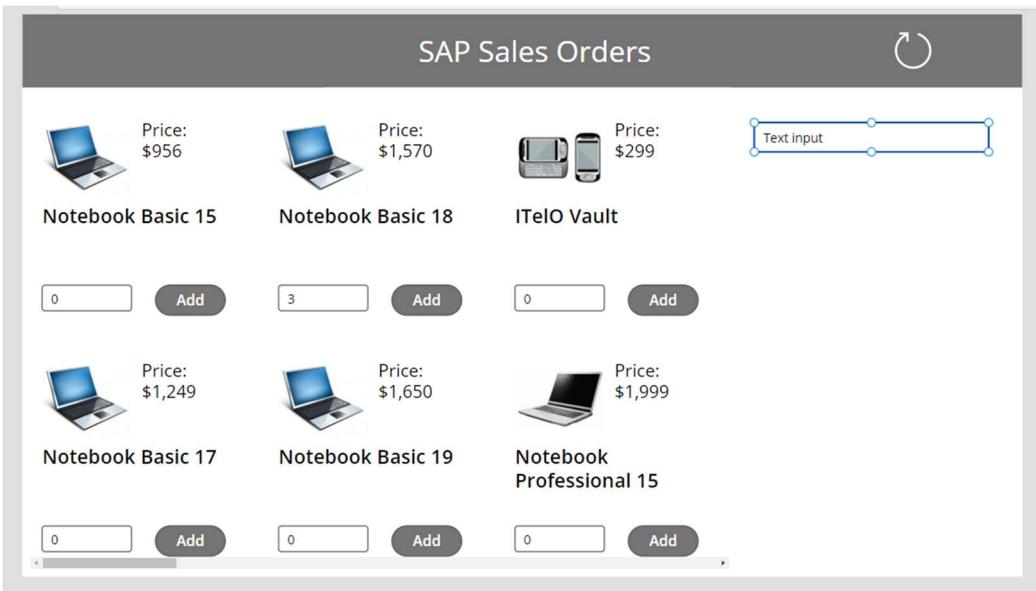
1. Add a search bar to search for products. In the left-hand navigation under **Tree view**, select the **SAP Products screen**. In the **ribbon**, click on **Insert**, then **Input** and finally **Text input**.



2. Rename the text input box. With the **Text input box** still selected in the **Tree view**, Click on the “...” and then **Rename**. Rename to **Search Products**.



3. Move the Search Products box. Drag and drop the Search Products box under the Header Label on the far side of the canvas.



4. Update the Product Gallery Items Property. We need to change the Items property for the Product Gallery to only show the product names from the search field. In the left-hand navigation under **Tree View**, select the **Product Gallery**. In the ribbon under the **Items property**, change the **formula** to the following:

**Search(zProducts,'Search Products'.Text,"ProductID","Name")**

**TIP:** If your Product Gallery is blank. Play the app and search for a product ( mouse, notebook ) and the collection should refresh.

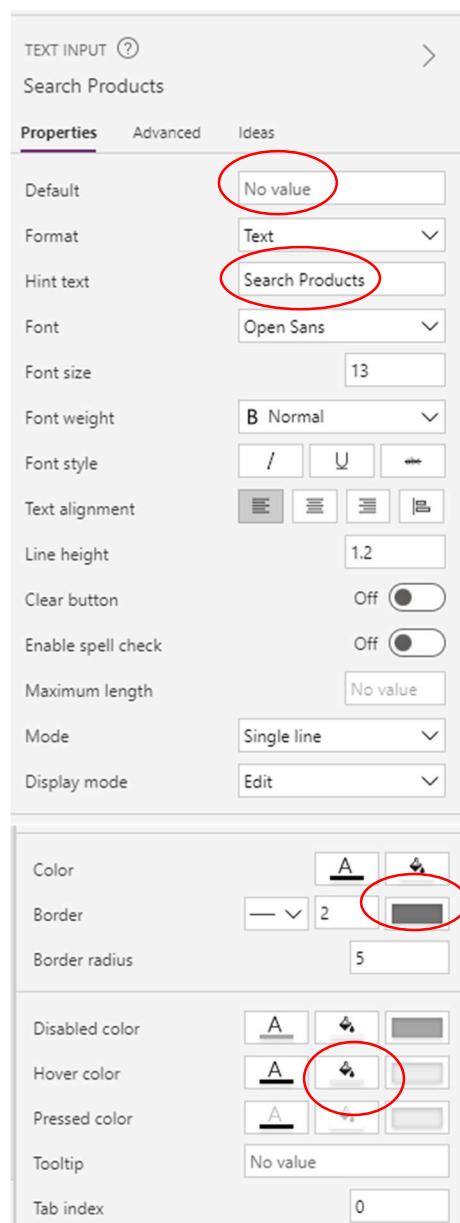
5. Update more properties for the Search box. We change the default to blank, add a hint text for the search box and change the border color to match our color scheme. In the left-hand navigation under Tree view, select the Search Products box. In the right-hand pane under Properties, change the following:

Default – delete all text and have blank.

Hint Text – **Search Products**

Border – change to dark grey

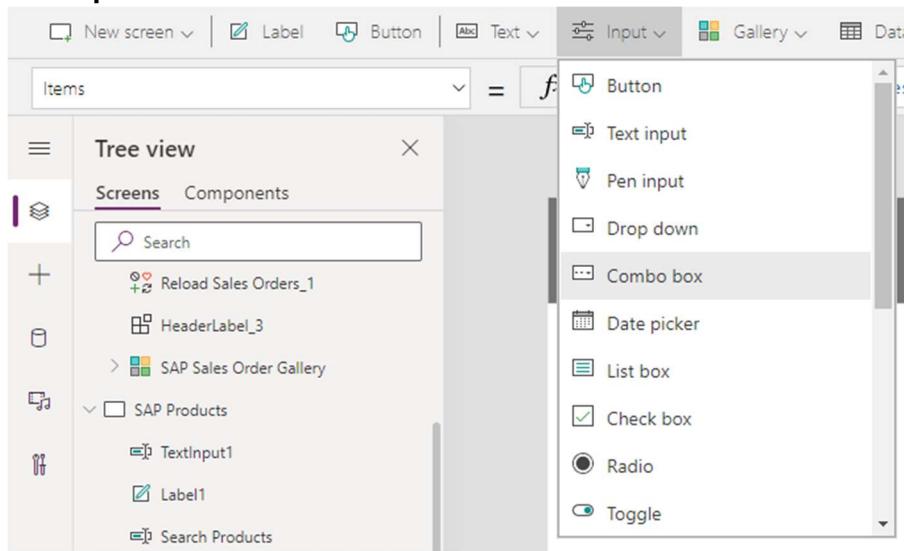
Hoover color – change to a very light grey



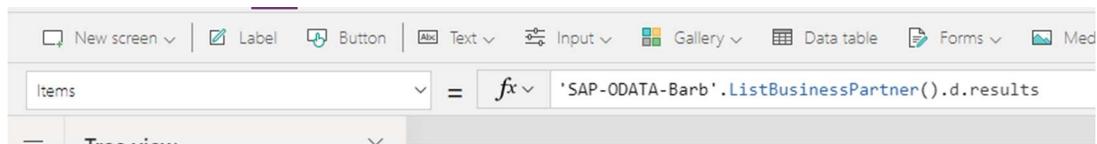
6. Save your app. In the ribbon, click on the **File** and then **Save**. Click on the **back arrow** to continue to edit your app.

## Task 5: Add a combo box and label for selecting Customer Name

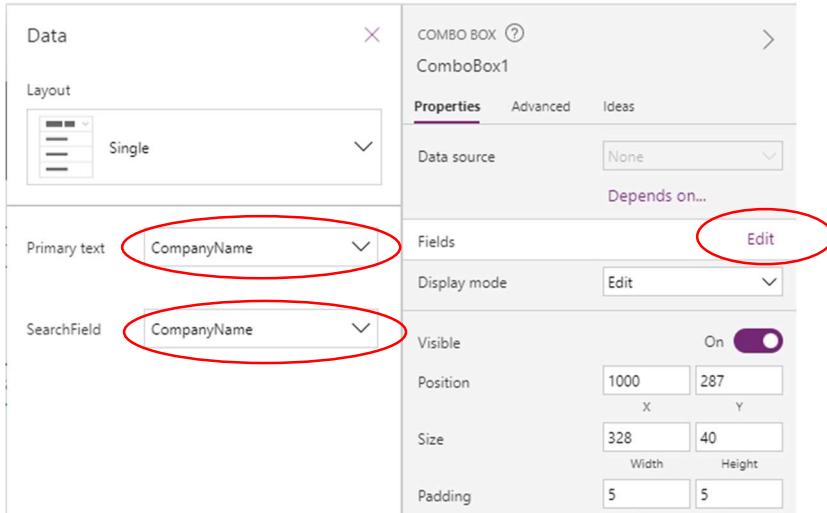
1. We need to add a combo box for selecting the Customer Name for the Sales Order and a label for a description. In the left-hand navigation under **Tree View**, select the **SAP Products screen**. In the **ribbon**, click on **Insert** and then **Input** and then select **Combo Box** to a box to select the customer name.



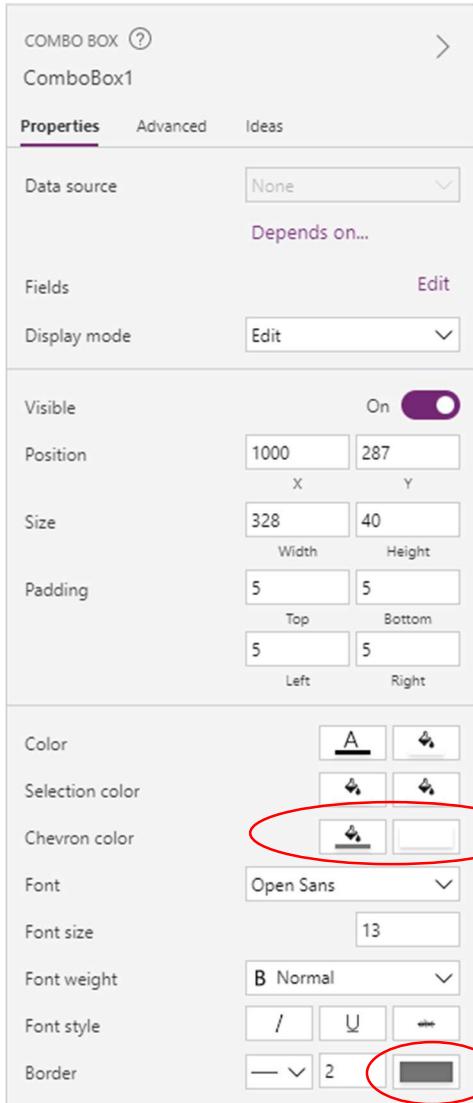
2. Set the Items property for the combo box. Close out of the data source selection. In the ribbon under the **Items** property, change the formula to '**'SAP-ODATA-YourName'.ListBusinessPartner().d.results**'. Change the highlighted text to match your connection.



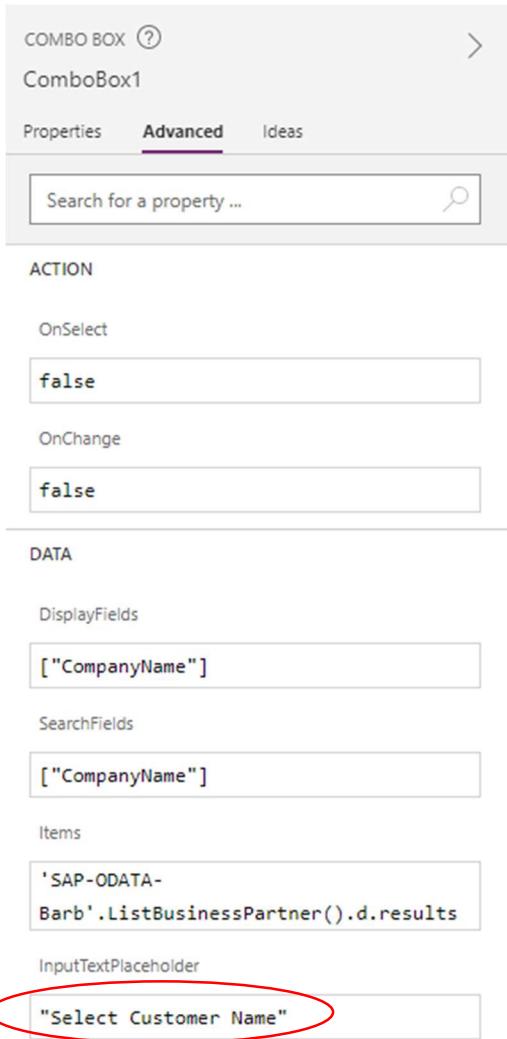
3. Change field to show Customer Name. Currently if you run the app, it will show the customer ID rather than name. So we need to change to Customer Name. In the left-hand pane under **Properties**, click on **Edit** across from **Fields**. Using the drop down, change both **Primary text** and **Search Field** to **CompanyName**.



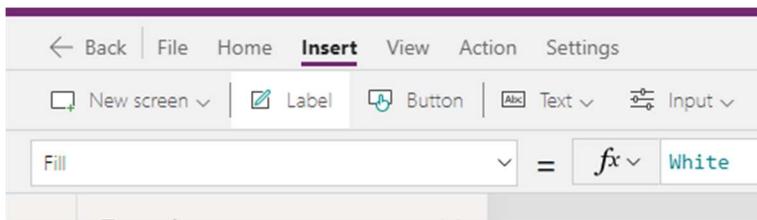
4. Change more properties for combo box. We can change the border and chevron color of the combo box to match our color theme. Still on the Properties tab, change the chevron fill color to dark grey and change the chevron background color to white. Finally, change the border color to dark grey.



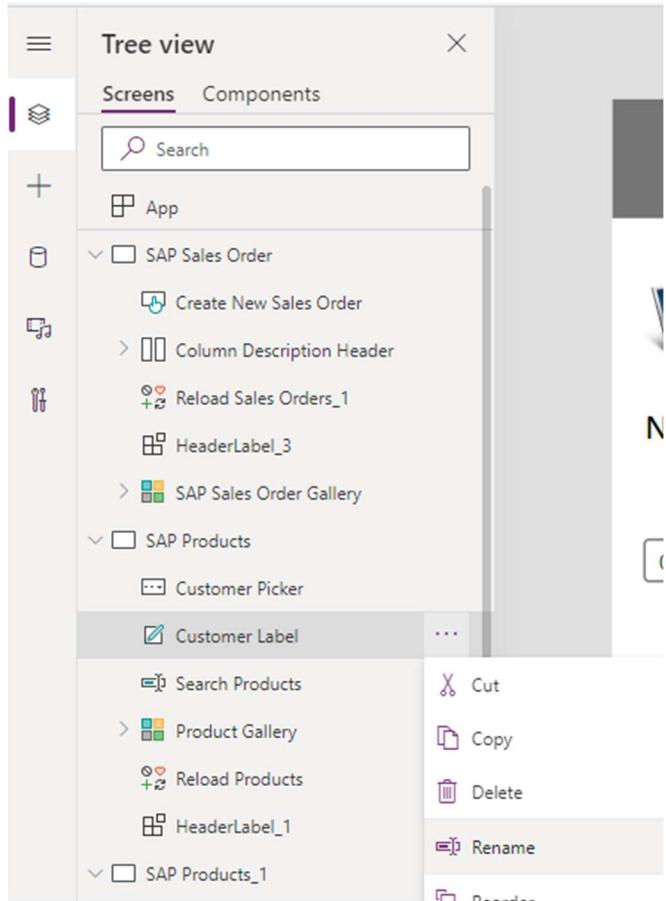
5. Change the default text in the combo box. We can change the text that is displayed in the box from Find Items to Select Customer Name. In the left-hand pane click on **Advanced**. Change **InputTextPlaceholder** to “**Select Customer Name**”.



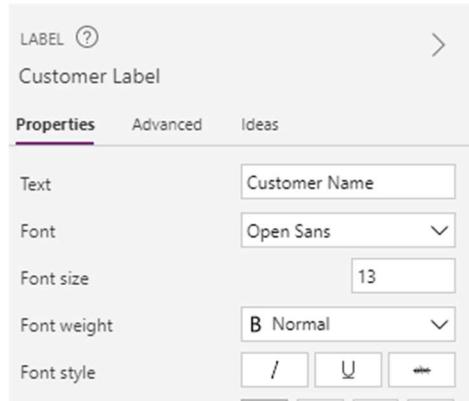
6. Insert a label field for a description. Still on the **Insert** tab, click on **Input** and then **Label** to insert a label describe the combo box.



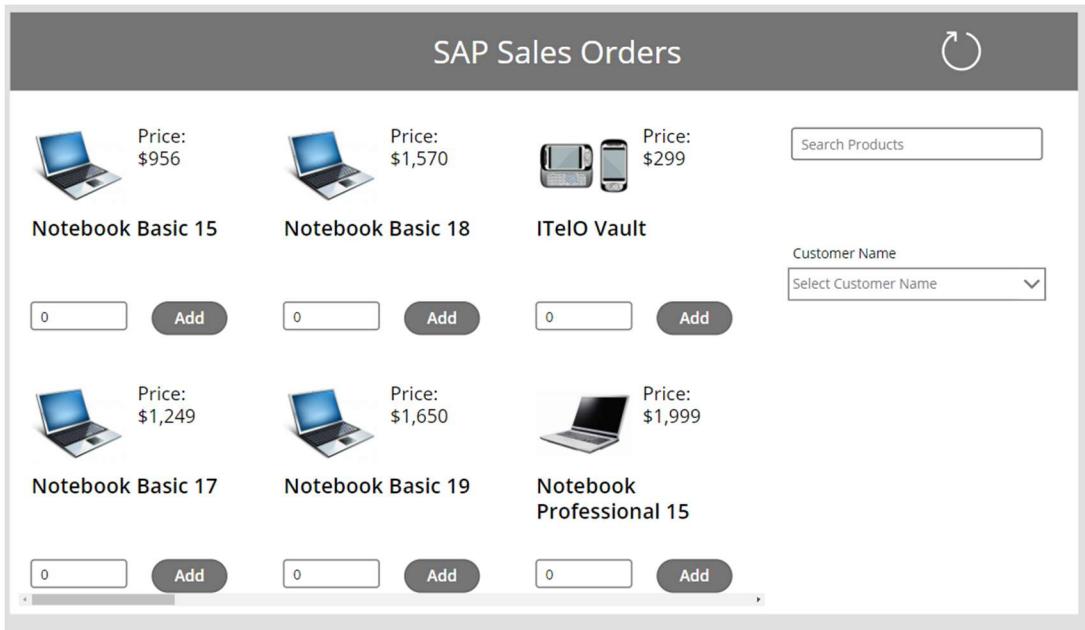
7. Rename the combo box and label. In the left-hand navigation under **Tree view**, select the **Combo box** on the **SAP Products screen**. Click on the “...” and select **Rename**. Rename to **Customer Picker**. Follow the same steps and rename the label to **Customer Label**.



8. Change the text on the screen for the Customer Label. In the left-hand navigation under **Tree View**, select the **Customer Label**. In the right-hand pane under **Properties**, change the **Text** to **Customer Name**.



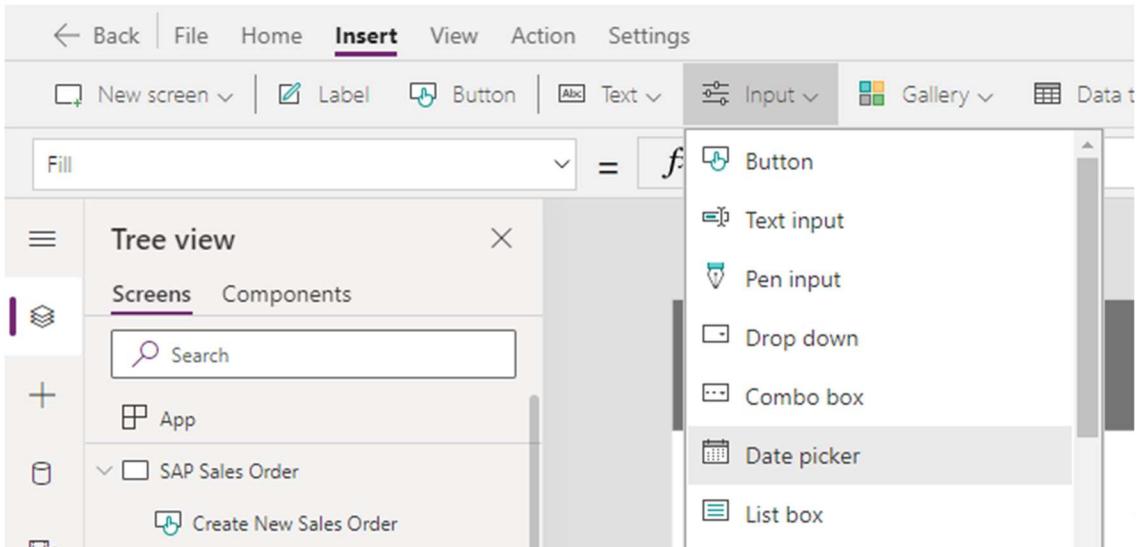
9. Position the label and combo box. **Drag and drop** the **Customer Label** and **Customer Picker** **combo box** underneath the **Search box**.



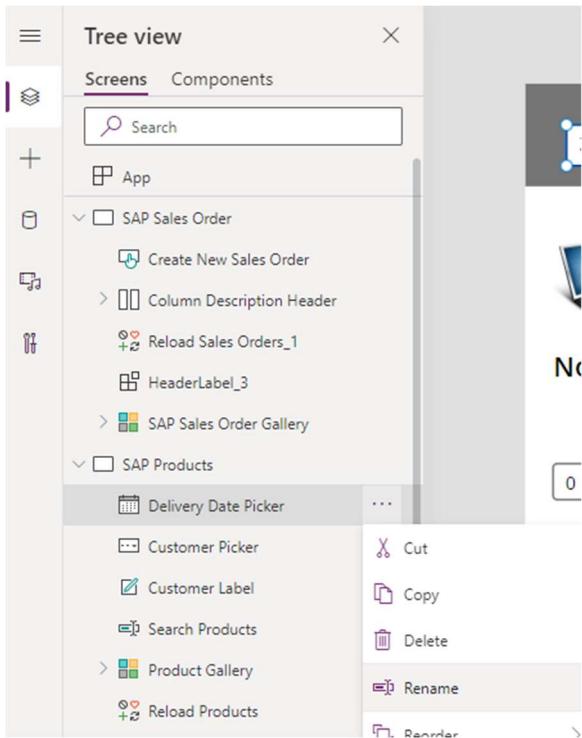
10. Save your app. In the ribbon, click on the **File** and then **Save**. Click on the **back arrow** to continue to edit your app.

## Task 6: Add a date picker box and label for selecting the Delivery Date.

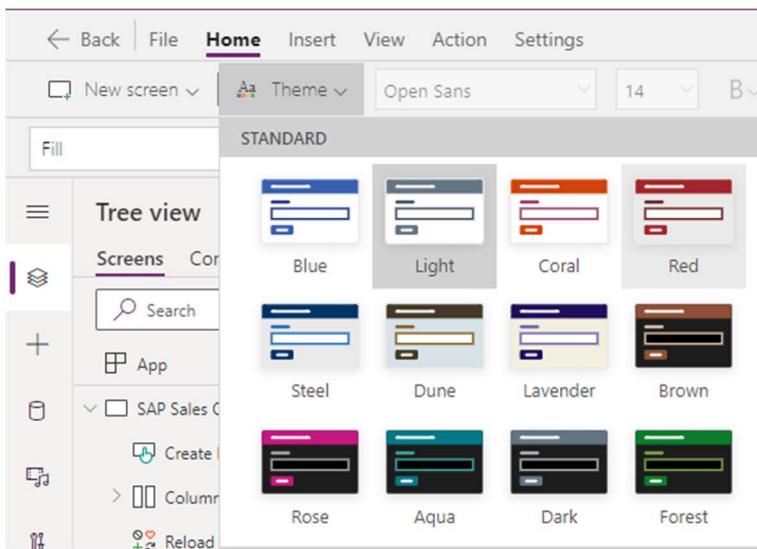
1. Add a date picker box. In the left-hand navigation under **Tree view**, select the **SAP Products Screen**. In the ribbon, select **Insert**, then **Input** and then **Date Picker**.



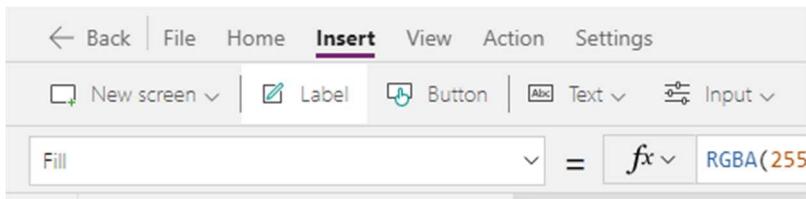
2. Rename the date picker box. In the left-hand navigation under **Tree view**, select the **DatePicker** on the **SAP Products screen**. Click on the “...” and select **Rename**. Rename to **Delivery Date Picker**.



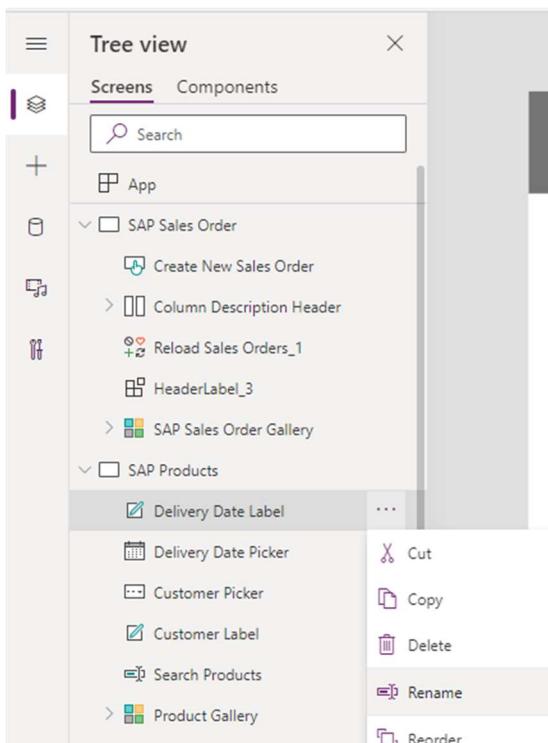
3. Change the properties of the date picker. Change the color scheme of the date picker box to match our color theme. There is currently no way to change the calendar icon on the date picker. We can change the border under properties. You can change the Theme and that will change the calendar icon. In the ribbon, click on **Home**. Change the **Theme** to **Light**. This will change the border and calendar icon to a similar color.



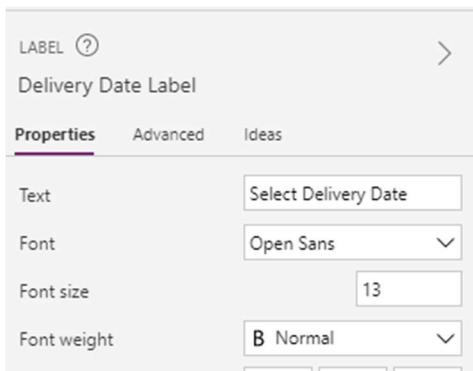
4. Insert a label to describe the date picker. In the left-hand navigation pane under **Tree view**, select the **SAP Products screen**. In the ribbon, select **Insert** and then **Label**.



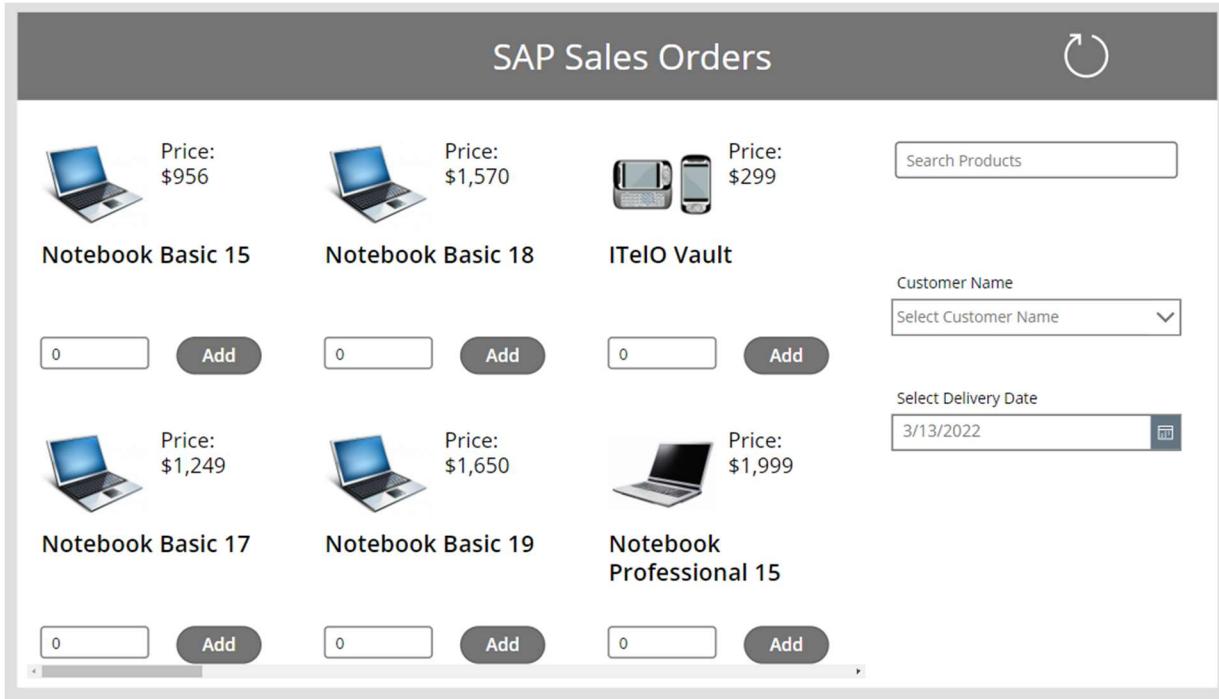
5. Rename the label. In the left-hand navigation under **Tree view**, select the **Label** on the **SAP Products** screen. Click on the “...” and select **Rename**. Rename to **Delivery Date Label**.



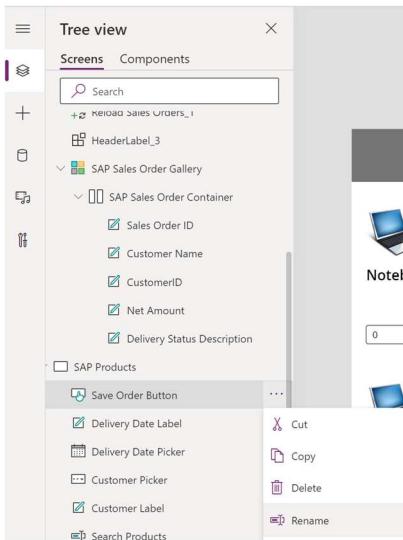
6. Change the properties of the label. Change the words that are displayed in the label. Keep the **Delivery Date Label** selected. In the right-hand pane under **Properties**, change the **Text** to **Select Delivery Date**.



7. Move the label and date picker box. **Drag and drop** the **Delivery Data Label** and **Delivery Date Picker** under the Customer Name combo box. Using the corners of the **Delivery Data Label**, stretch it out so there is only 1 line of text.



- Save your app. In the ribbon, click on the **File** and then **Save**. Click on the **back arrow** to continue to edit your app.

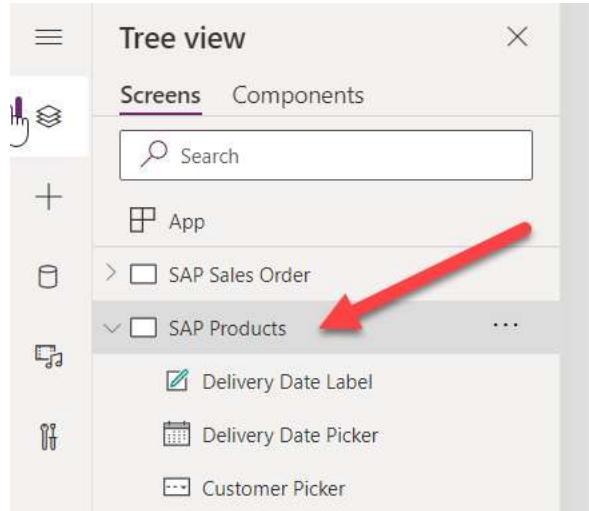


You are now ready to create Power Automate Flows!

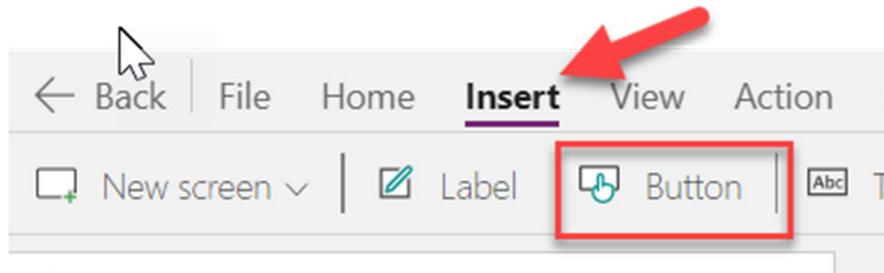
## Exercise 4: Add Power Automate flows

### Task 1: Create a button on the SAP Products screen

1. Click to select the **SAP Products screen**



2. Insert a **button**



3. Locate in the lower-right section and change the button text to: Save Order



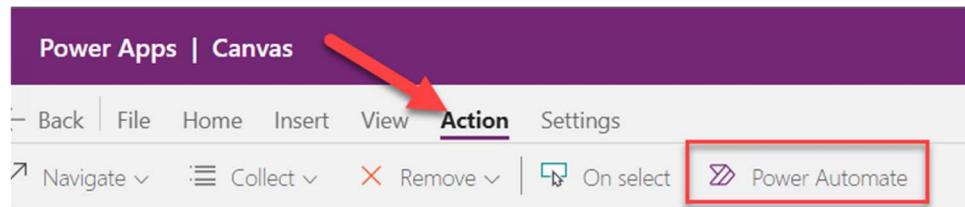
4. Rename the Button. In the left-hand navigation in **Tree view**, select the button. Click on the ‘...’ and click on **Rename**. Rename to **Save Order Button**.

## Task 2: Create Power Automate Flow to add new Sales Order in SAP

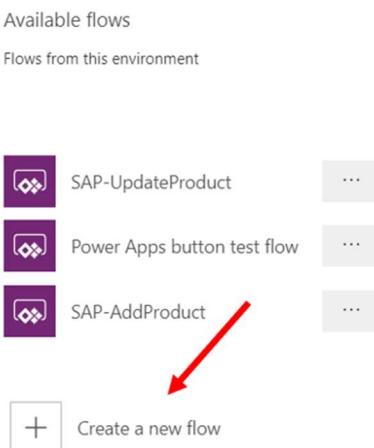
1. Select your **Save Order** Flow button



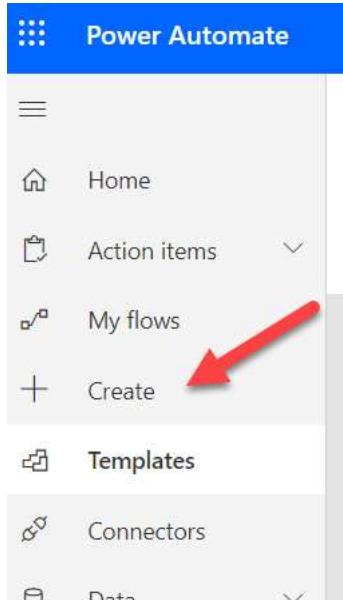
2. Select the **Action** menu item and click **Power Automate**



3. Click to **create a new flow**



4. Click + **Create** from the left navigation



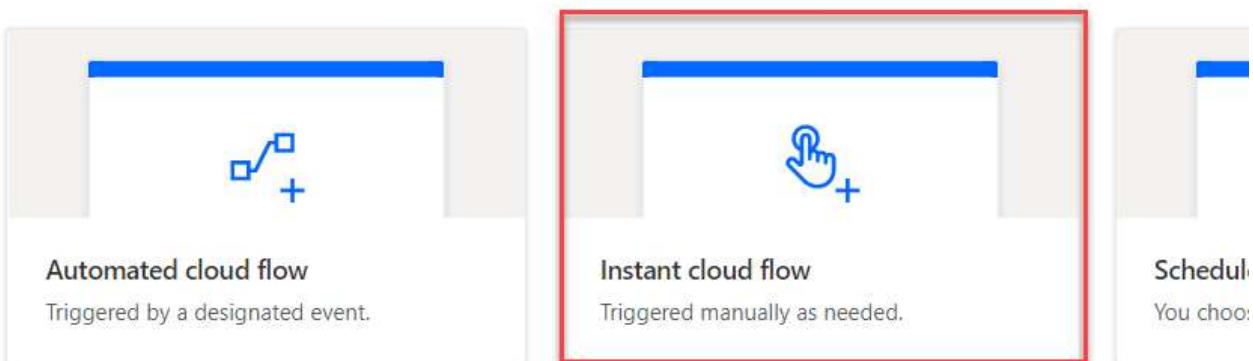
5. Append the following the url: **?addConnectorHideKey=odataconnector** - hit **enter**

A screenshot of a web browser window titled 'Power Automate'. The address bar shows the URL <https://us.flow.microsoft.com/manage/environments/9cc11098-36dd-465b-a86d-7e136f926267/create?addConnectorHideKey=odataconnector>. A red box highlights the query string part of the URL.

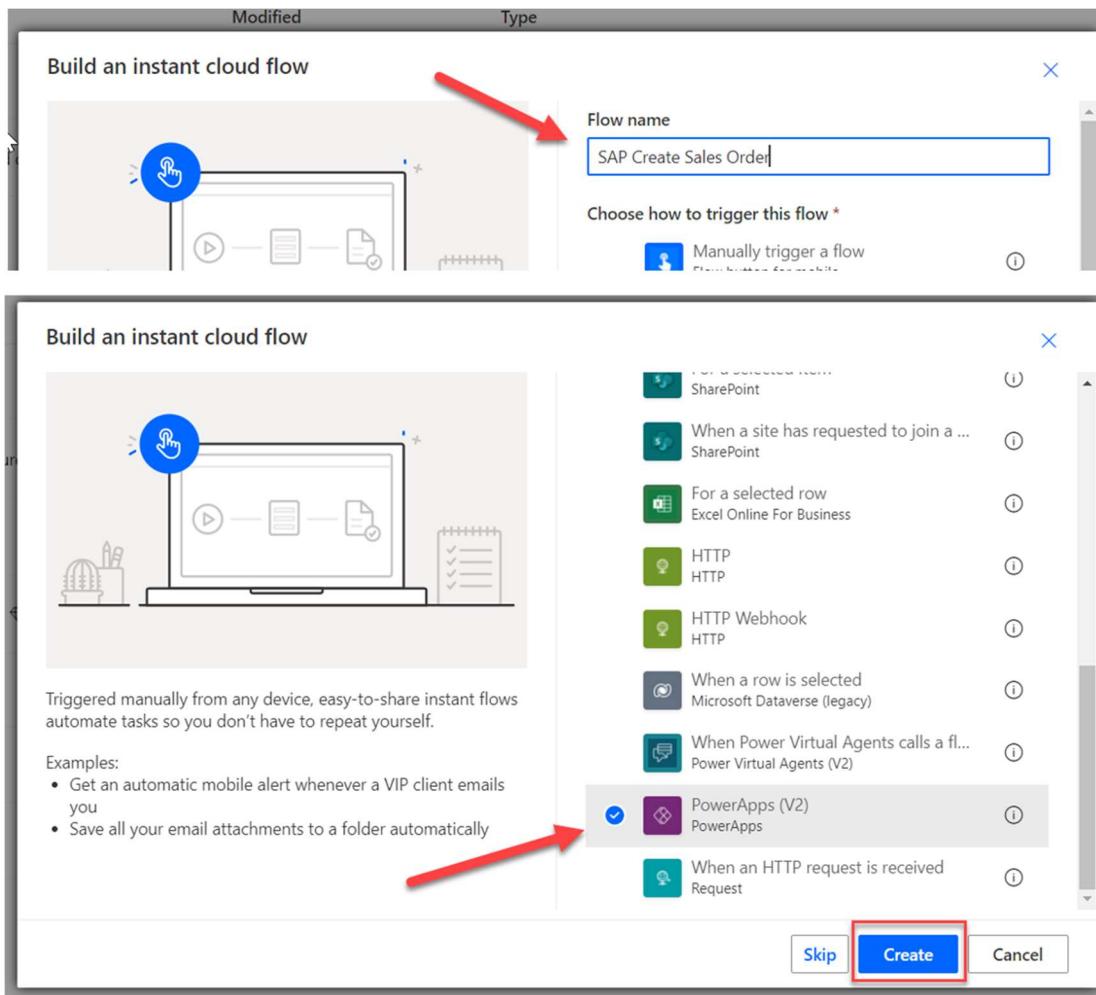
6. In the refreshed page select new **Instant Cloud Flow**

### Three ways to make a flow

Start from blank ⓘ



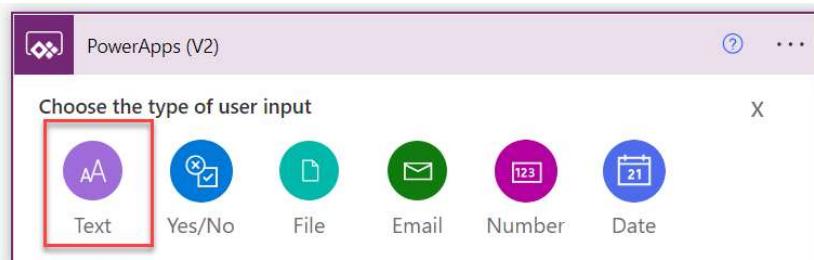
7. Name the Flow: **SAP Create Sales Order**, scroll down to **PowerApps (V2)**, and click **Create**



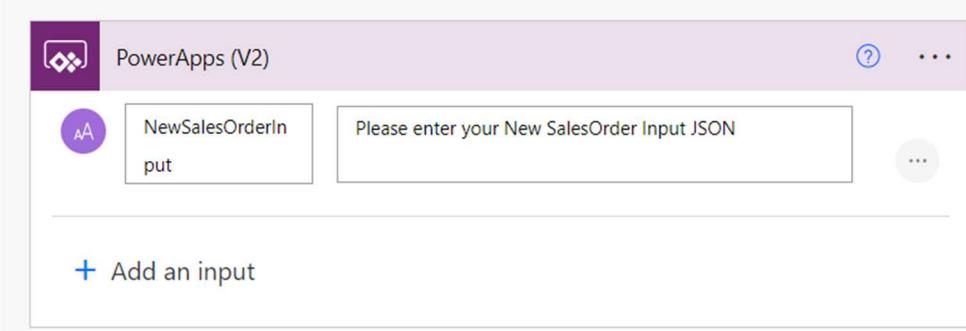
8. Click to expand PowerApps (V2) and click **+Add an input**



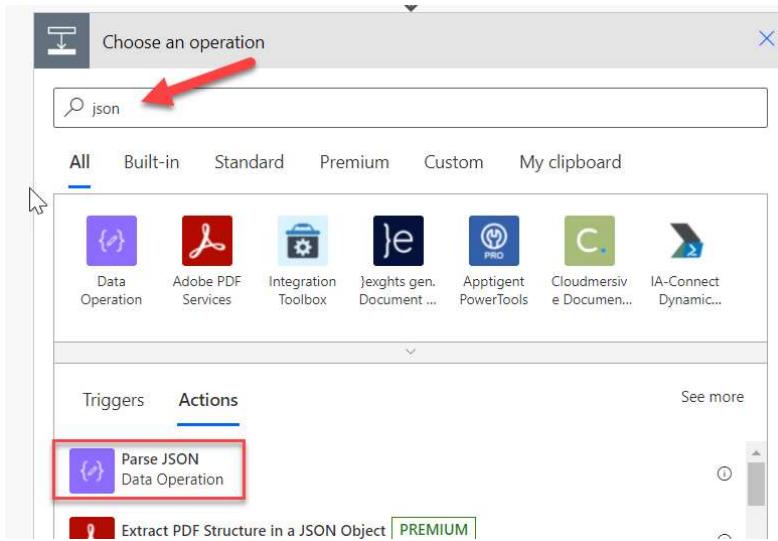
9. Select **Text**



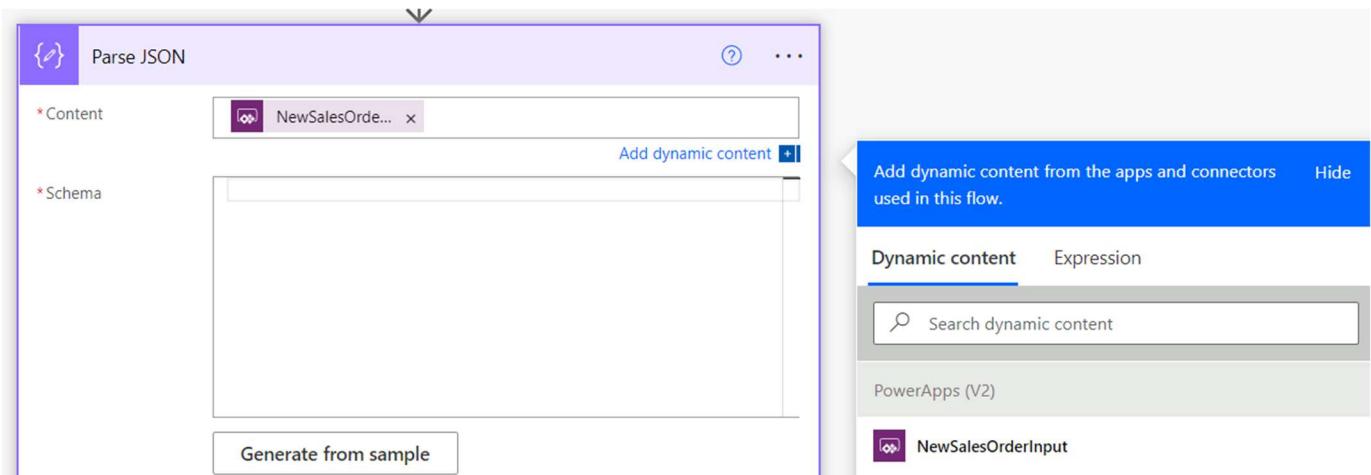
10. Name the input: **NewSalesOrderInput** and enter: **Please enter your New SalesOrder Input**



11. Click **+ New Step**. Search for **json** and select **Parse JSON**



12. In Content select **NewSalesOrderInput** from Dynamic Content

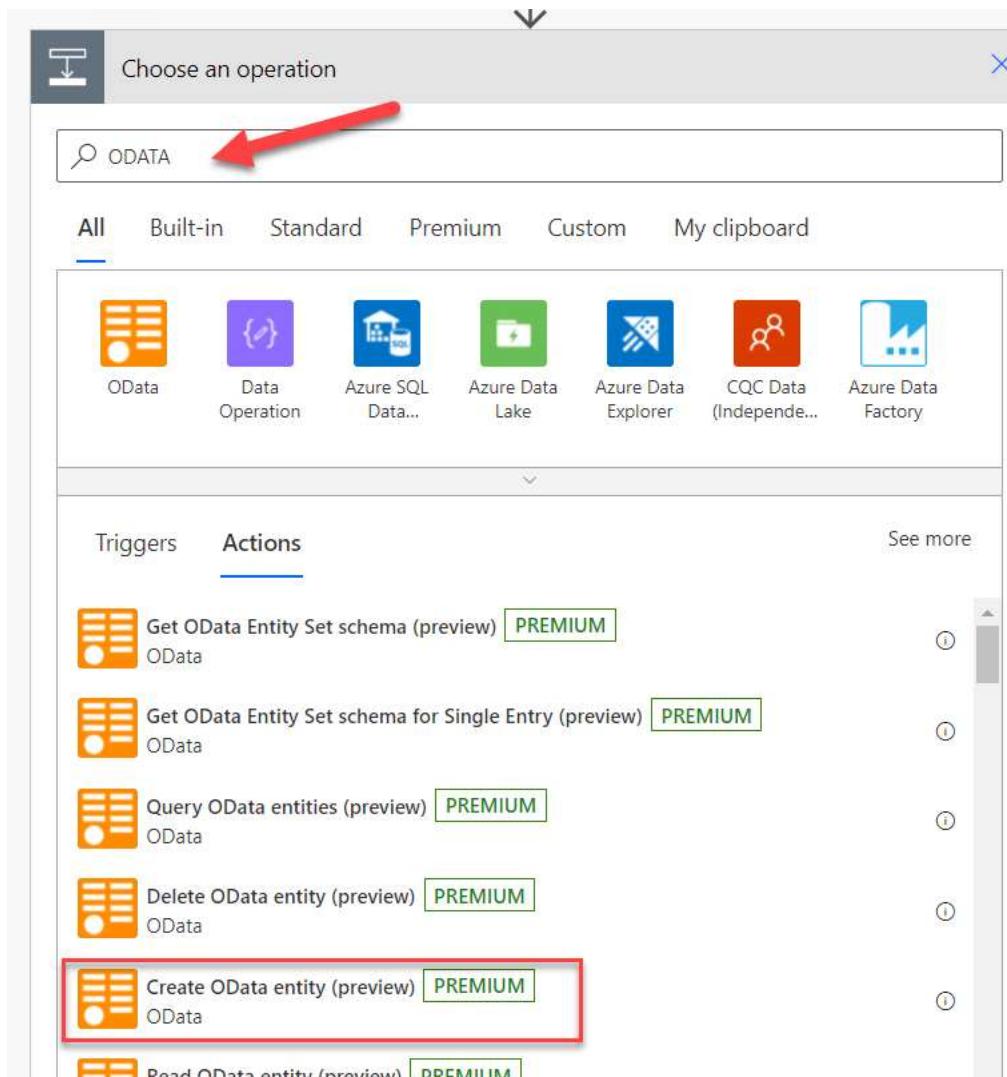


13. Paste the following Schema

```
{  
    "type": "object",  
    "properties": {  
        "CurrencyCode": {  
            "type": "string"  
        },  
        "CustomerID": {  
            "type": "string"  
        },  
        "DeliveryDate": {  
            "type": "string"  
        },  
        "SalesOrderID": {  
            "type": "string"  
        },  
        "SalesOrderNote": {  
            "type": "string"  
        },  
        "_SalesOrderLineItems": {  
            "type": "array",  
            "items": {  
                "type": "object",  
                "properties": {  
                    "ProductID": {  
                        "type": "string"  
                    },  
                    "Quantity": {  
                        "type": "integer"  
                    }  
                },  
                "required": [  
                    "ProductID",  
                    "Quantity"  
                ]  
            }  
        }  
    }  
}
```



14. Click + New Step and search for **OData**, select **Create OData entity (preview)**



15. Populate the following

- OData service url: [https://sapdev5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE\\_BASIC/](https://sapdev5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_BASIC/)
- OData Entity name: SalesOrderSet
- Dynamic value: SalesOrderID
- Dynamic value: SalesOrderNote
- Dynamic Value: CustomerID
- Dynamic Value: CusrrencyCode

Create OData entity (Preview)

\* OData Service Uri: https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE\_BASIC/

\* OData Entity name: SalesOrderSet

SalesOrderSet  
SalesOrderID: SalesOrderID x

SalesOrderSet Note: SalesOrderNote x

SalesOrderSet  
CustomerID: CustomerID x

SalesOrderSet  
CurrencyCode: CurrencyCode x

Add dynamic content

+ New step Save

Dynamic content Expressions

Search dynamic content

Parse JSON

CurrencyCode

CustomerID

16. Click the ellipsis of the Create OData entity and select: **Rename**

Create OData entity (Preview) ...

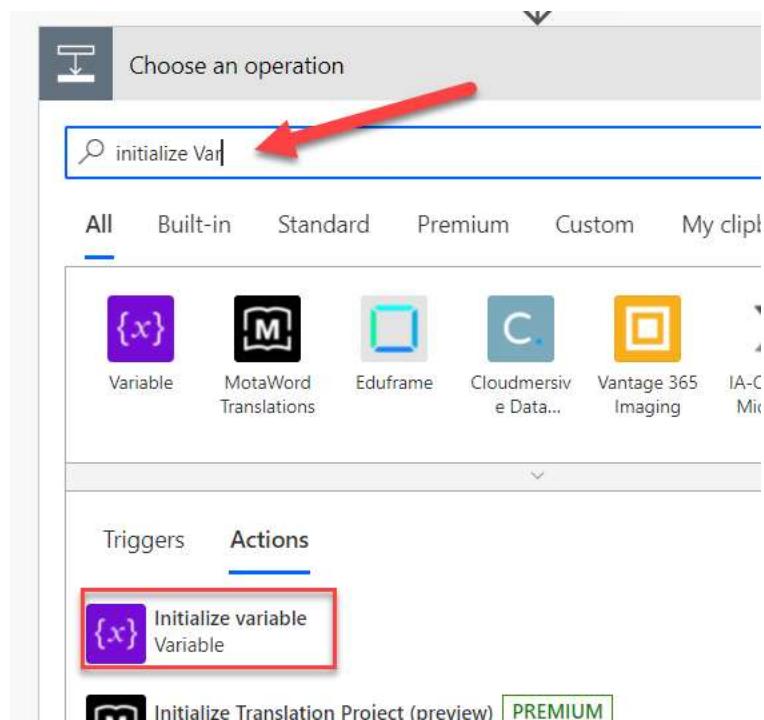
Copy to my clipboard (Preview)

Rename

17. Rename to Create OData Entity – SalesOrderSet

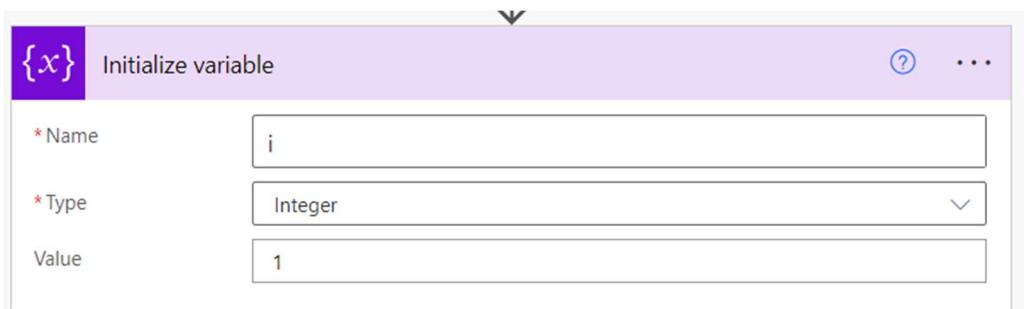
Create OData entity - SalesOrderSet (Preview)

18. Click + New Step and search for **Initialize variable**

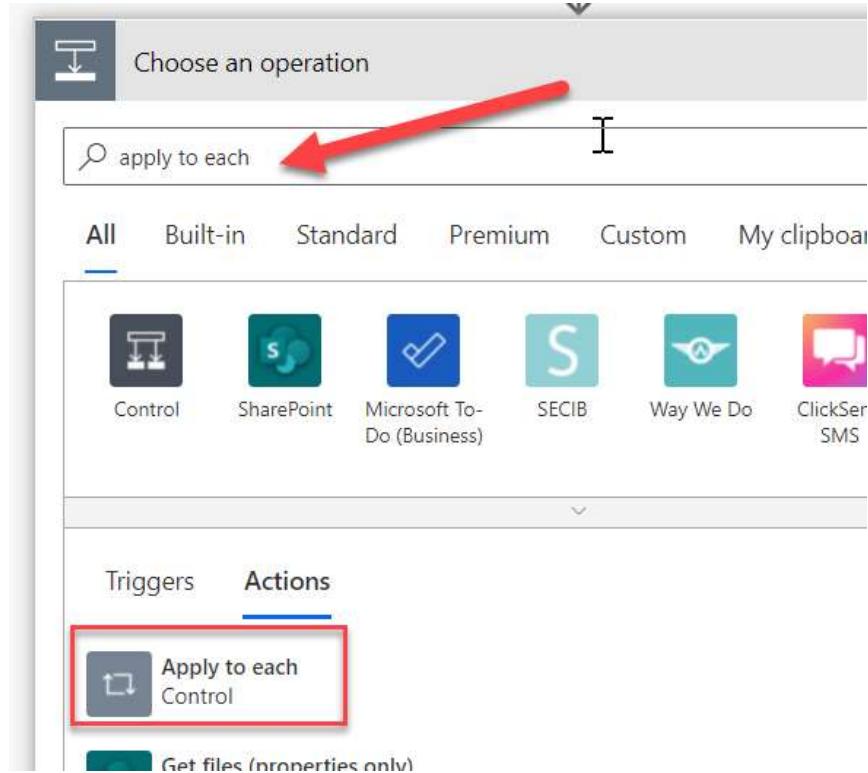


19. Populate the following:

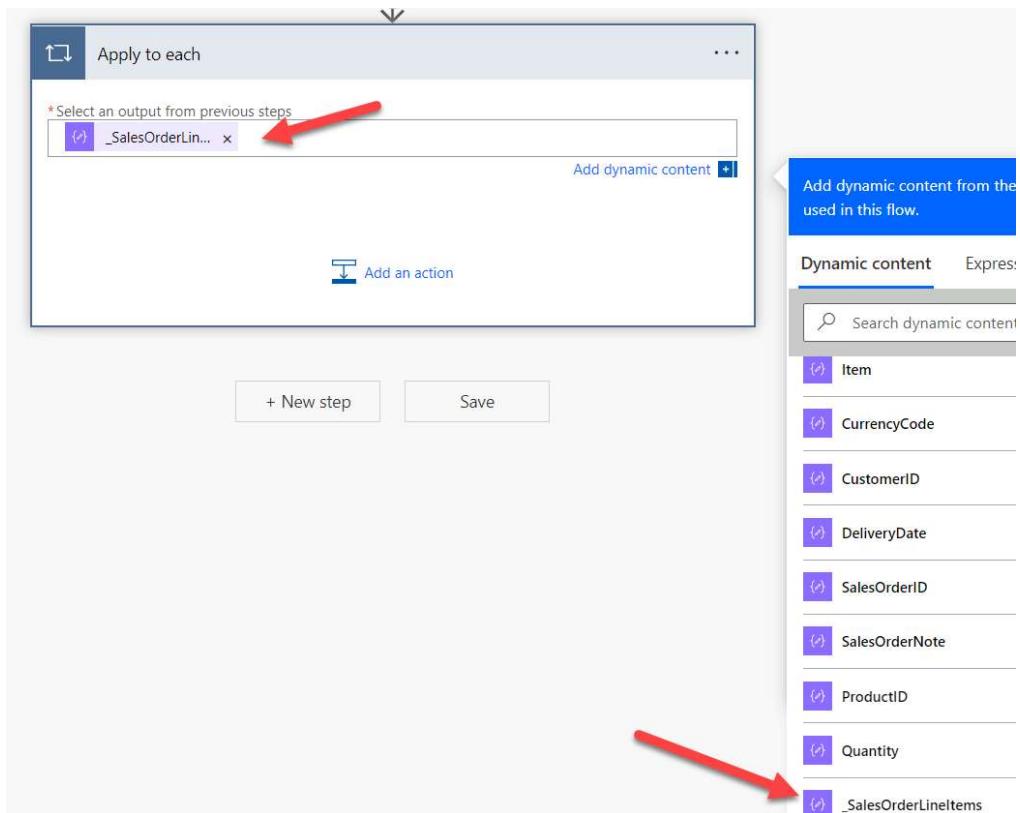
- Name: i
- Type: Integer
- Value: 1



20. Click + New Step, search and select **Apply to Each**

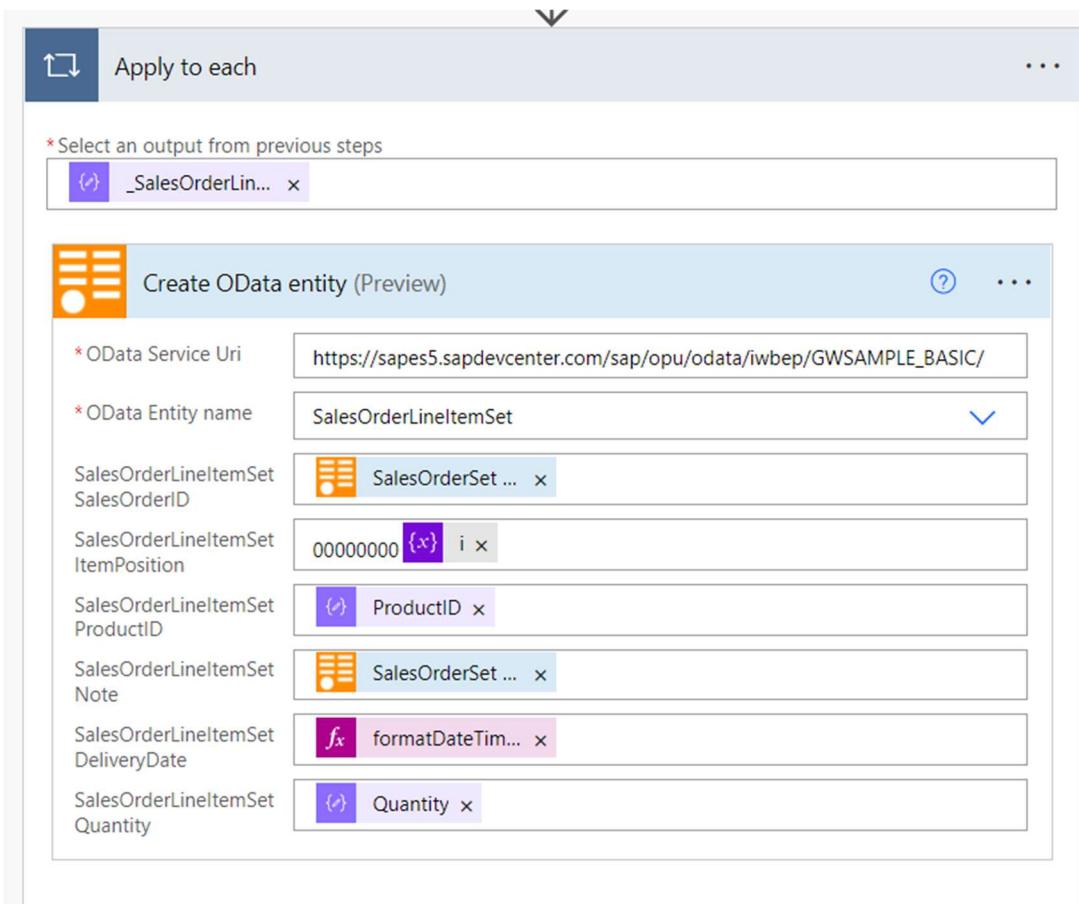


21. Select to add from Dynamic content: \_SalesOrderLineItems

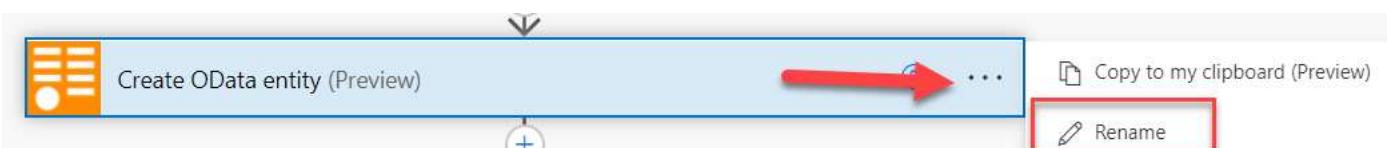


22. Inside the **Apply To Each** control, Click **Add an Action**, search for **OData – Create OData Entity** again. Populate as follows

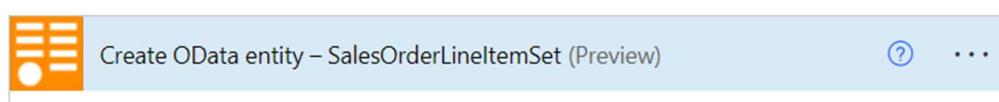
- a. OData Service Uri: [https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE\\_BASIC/](https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_BASIC/)
- b. OData Entity Name: SalesOrderLineItemSet
- c. Dynamic value: SalesOrderSet SalesOrderID
- d. SalesOrderLineItemSet ItemPosition: type: 00000000 and click to add {x} I variable
- e. SalesOrderLineItemSet ProductID: ProductID
- f. SalesOrderLineItemSet Note: SalesOrderSet Note
- g. SalesOrderLineItemSet DeliveryDate: click on Expression, type or paste the following and click ok  
formatDateTime(body('Parse\_JSON')?['DeliveryDate'])
- h. SalesOrderLineItemSet Quantity: Quantity



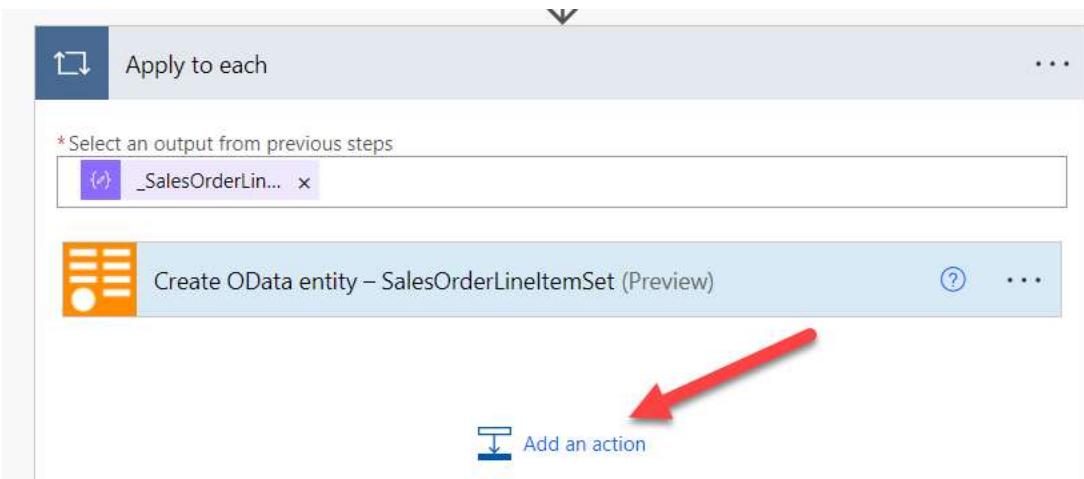
23. Click the ellipsis of the Create OData entity and select: **Rename**



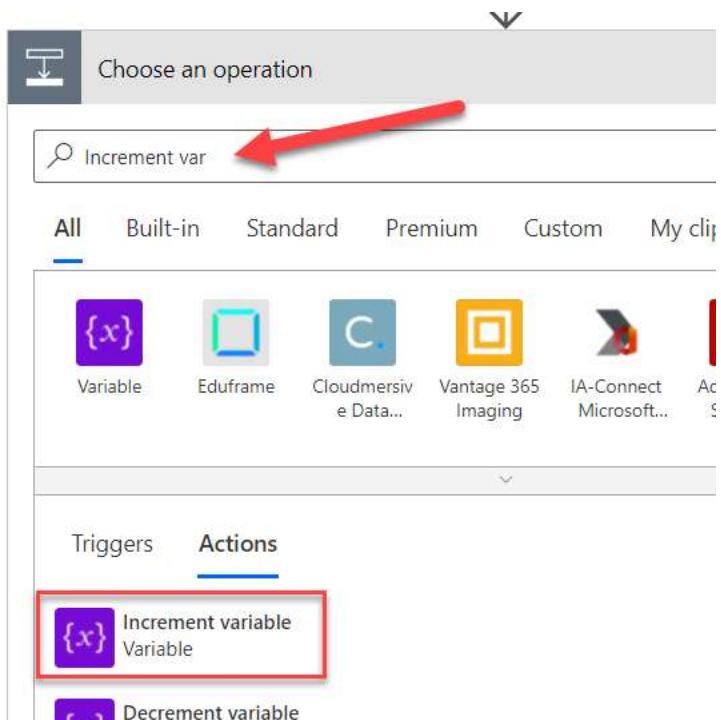
24. Rename to: **Create OData entity – SalesOrderLineItemSet**



25. Still *inside* the **Apply to Each** step, click **Add an Action**

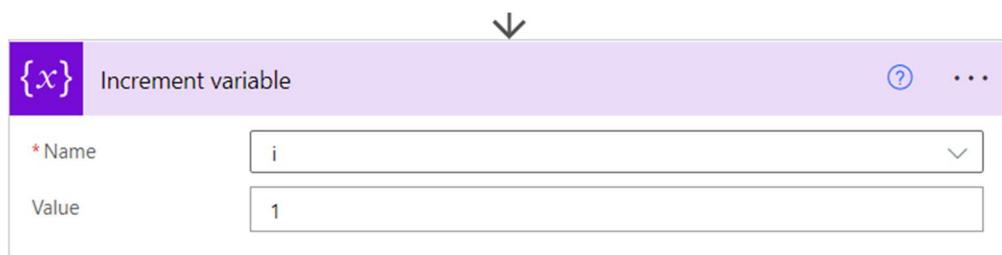


26. Search for and Select Increment Variable

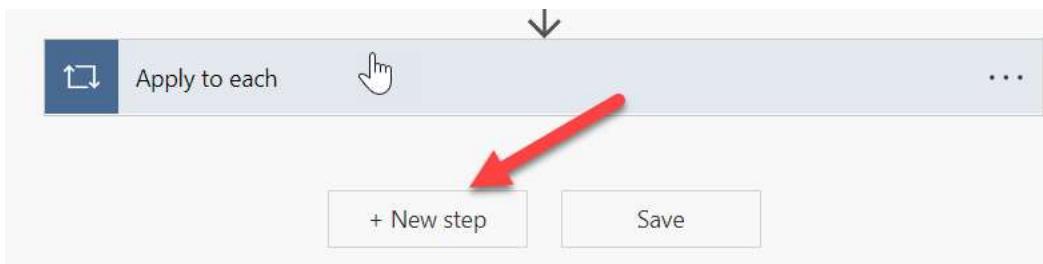


27. Populate

- Name: i
- Value: 1



28. Outside the **Apply to Each step**, click **+ New Step**



29. Search for **OData** and Read **OData Entity**

Choose an operation

odata

All Built-in Standard Premium Custom My clipboard

OData Data Operation Azure SQL Data... Azure Data Lake Azure Data Explorer CQC Data (Independ...

Triggers Actions

Get OData Entity Set schema (preview) PREMIUM  
OData

Get OData Entity Set schema for Single Entry (preview) PREMIUM  
OData

Query OData entities (preview) PREMIUM  
OData

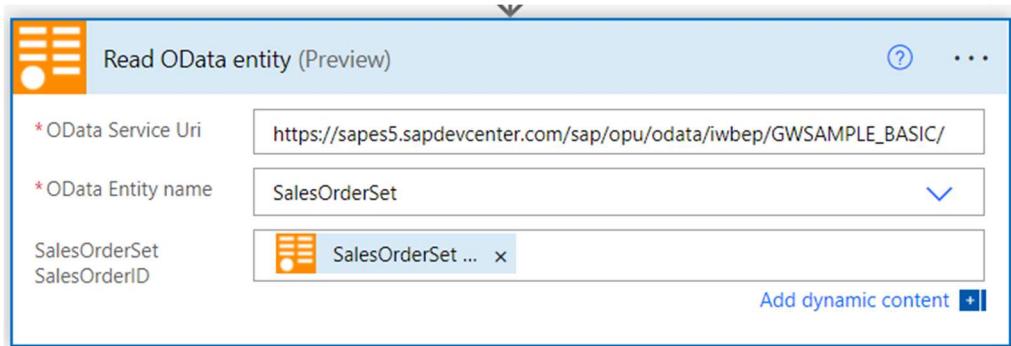
Delete OData entity (preview) PREMIUM  
OData

Create OData entity (preview) PREMIUM  
OData

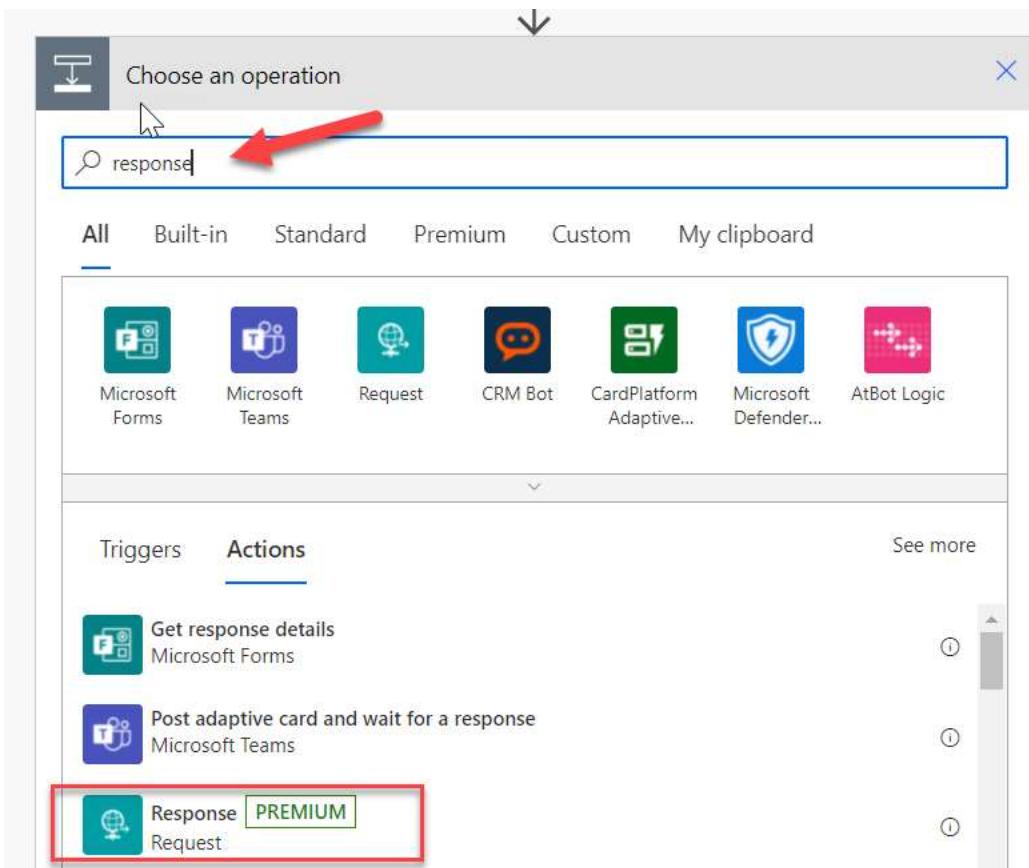
Read OData entity (preview) PREMIUM  
OData

30. Populate the following:

- OData Service Uri: [https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE\\_BASIC/](https://sapes5.sapdevcenter.com/sap/opu/odata/iwbep/GWSAMPLE_BASIC/)
- OData Entity Name: SalesOrderSet
- SalesOrderSet SalesOrderID: SalesOrderSet SalesOrderID



31. Click to + New Step, search for and select **Response**

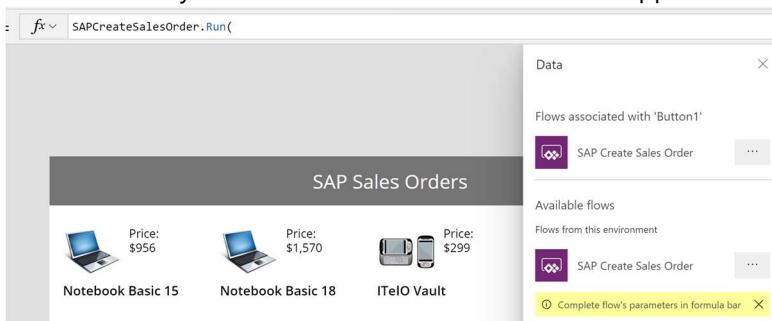


32. Populate the following:

- Status code: 200
- Body: SalesOrderSet

The screenshot shows the Microsoft Flow editor interface. On the left, there's a 'Response' step configuration screen. It has fields for 'Status Code' (set to 200), 'Headers' (with 'Enter key' and 'Enter value' fields), and 'Body' (containing 'SalesOrderSet' which is highlighted with a red box). Below these are 'Add dynamic content' and 'Show advanced options' buttons. At the bottom are '+ New step' and 'Save' buttons. On the right, a sidebar titled 'Dynamic content' is open, showing a search bar and a list of available entities: SalesOrderSet, Note, CustomerID, CurrencyCode, and SalesOrderSet again. The second 'SalesOrderSet' item is highlighted with a red box.

33. Click **Save**.
34. Click back on your **Make.PowerApps.com editor tab** where you were editing the canvas app.
35. Click on the new Flow you created to add it into the Canvas App and associate with that button.



36. In the **OnSelect** action of the **Save Order Button** paste the code below:

```
//Create variable to hold data of SalesOrderSet and SalesOrdersLineItemSet from Canvas App
```

```
UpdateContext({_SalesOrderHeader:{SalesOrderID:"0500000000",SalesOrderNote:"",CustomerID:'Customer Picker'.Selected.BusinessPartnerID,CurrencyCode:"USD",DeliveryDate:'Delivery Date Picker'.SelectedDate,_SalesOrderLineItems>ShowColumns(MyCart,"ProductID","Quantity")}});
```

```
//Create JSON string to pass data of SalesOrderSet and SalesOrdersLineItemSet
```

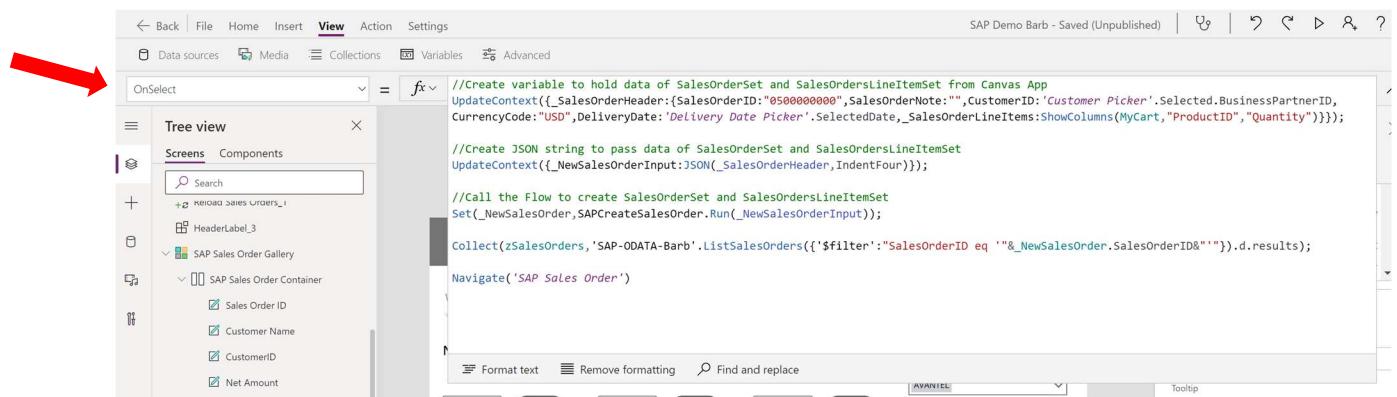
```
UpdateContext({_NewSalesOrderInput:JSON(_SalesOrderHeader,JSONFormat.IndentFour)});
```

```
//Call the Flow to create SalesOrderSet and SalesOrdersLineItemSet
```

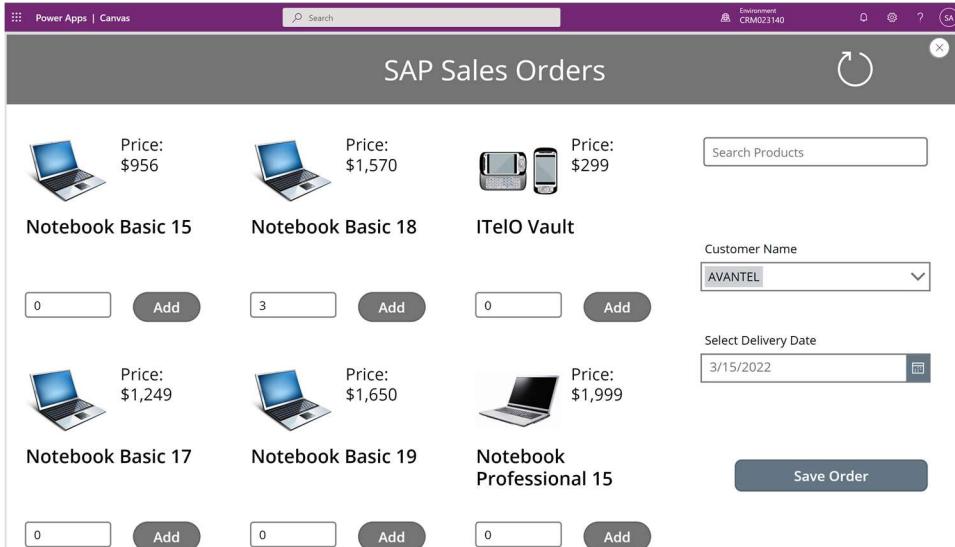
```
Set(_NewSalesOrder,SAPCreateSalesOrder.Run(_NewSalesOrderInput));
```

```
Collect(zSalesOrders,'SAP-ODATA-Barb'.ListSalesOrders({'$filter':'SalesOrderID eq "'&_NewSalesOrder.SalesOrderID&""'}).d.results);
```

### Navigate('SAP Sales Order')



37. Save your app. In the ribbon, click on the **File** and then **Save**. Click on the **back arrow** to continue to edit your app.
38. Test your app by selecting some products, a customer name and a delivery date. Click on the Save Order button and you should see dots across the top of the screen indicating the Power Automate Flow is working. You will return to the Sales Order screen when it is completed. You may need to refresh to see your new Sales Order.



The screenshot shows a table titled "SAP Sales Orders" with a "Create New Sales Order" button at the top left. The table has columns: Sales Order ID, Customer Name, Customer ID, Net Amount, and Delivery Status. The data is as follows:

Sales Order ID	Customer Name	Customer ID	Net Amount	Delivery Status
0500000000	SAP	0100000000	\$21,737.	Initial
0500000001	DelBont Industries	0100000002	\$12,271.	Initial
0500000002	New Line Design	0100000012	\$.	Initial
0500000003	AVANTEL	0100000008	\$9,420.	Initial

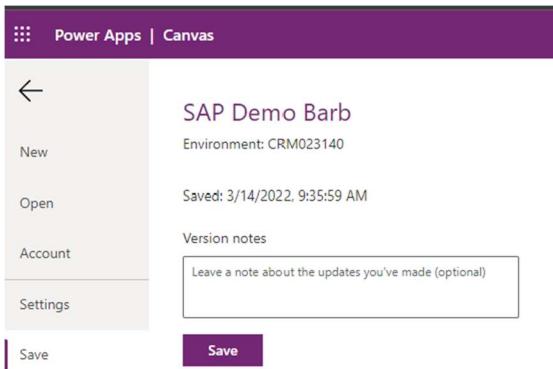


## Exercise 5: Import Power App into Teams.

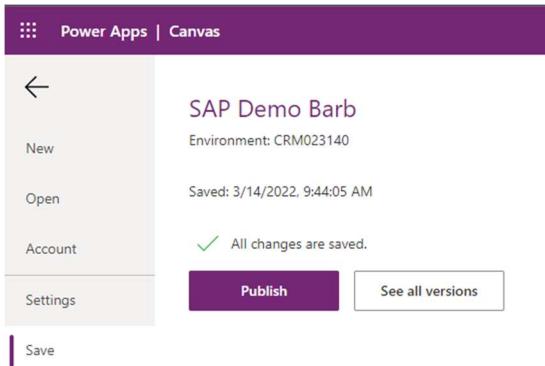
You can use Teams as the platform for all your applications. We will import the Power App we have created into Teams so people don't have to leave Teams to create a new Sales Order in SAP.

## Task 1: Final Save and Publish App

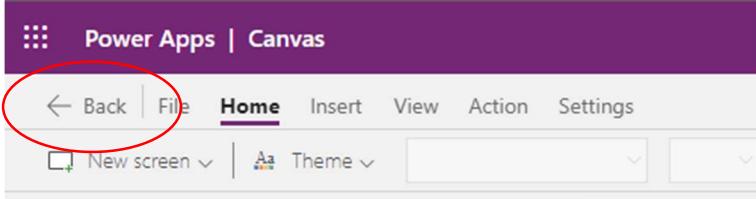
1. Save your app and publish. In the ribbon, click on **File** and then **Save**.



2. Publish your app. When the app has completed saving you will have the option to publish. This actions makes the new version available to others. Click on **Publish**.



3. Return to Power Apps home screen. Click on the **Back arrow** to return to the Power Apps home screen.



## Task 2: Add app to Teams

4. Add app to Teams. Select your app and click on the "...". Click on **Add to Teams**.

## Leveraging Power Apps with SAP on Azure Workshop

### Apps

Apps Component libraries (preview)

Name	Modified
SAP Demo Barb	5 min ago
SAP Products Sales Order App - Barb	
SAP Products Sales Order App	
Barb-SAPDemoNonResponsive	
Customer Service admin center (preview)	
SAP EPM Products Demo Starter	
Unified Service Desk Administrator	
Social Selling Assistant	
Omnichannel for Customer Service	
Customer Service workspace	1 mo ago

A context menu is open for the "SAP Demo Barb" app, showing options: Edit, Play, Share, Export package, Add to Teams (which is circled in red), Monitor, Analytics (preview), Settings, Delete, and Details.

5. Review app details. Review your app details and then click on **Add to Teams**.

Add to Teams

Review app details  
App details can't be changed in Teams once added.

SAP Demo Barb

Description  
Your app doesn't have a description, optionally add one in app settings.

Edit details

Advanced settings ▾

Add to Teams Download app

The "Add to Teams" button is circled in red.

6. Use Teams web client. You will see a pop-up message that the application is trying to open Teams. This will open the Teams desktop app. Since we want to use our Teams trial environment, we will stay with the web client. Click on **Cancel** to allow open the web client.

%2Fba1cabe6-dfd2-4334-96c0-0dcdf86e18e5%3F-templateInstanceId%3De56e2a9f-4a

This site is trying to open Microsoft Teams.  
https://teams.microsoft.com wants to open this application.

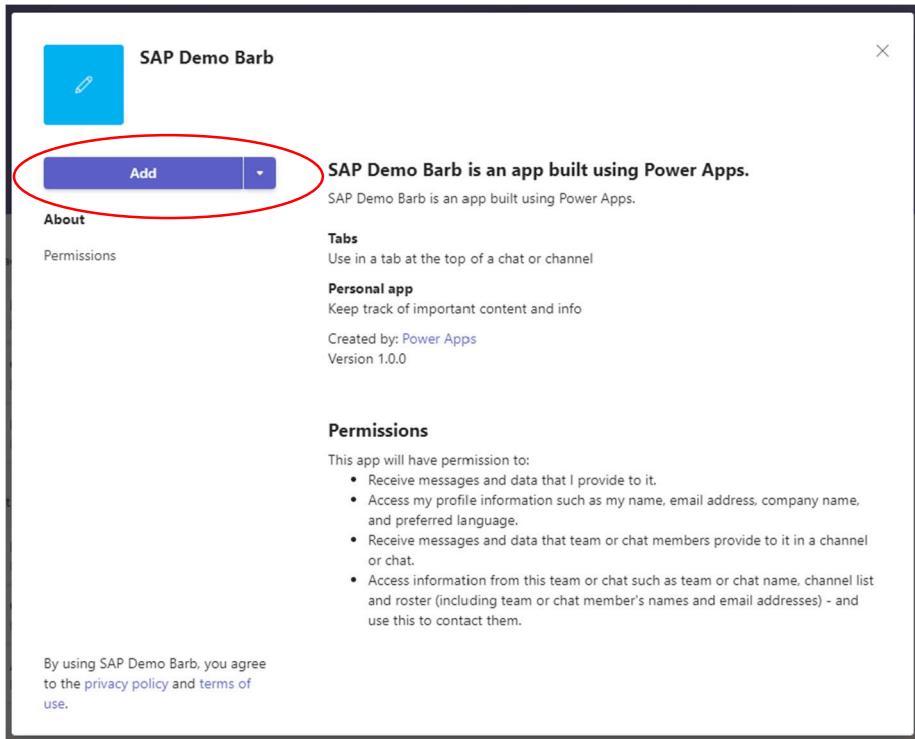
Always allow teams.microsoft.com to open links of this type in the associated app

Open Cancel

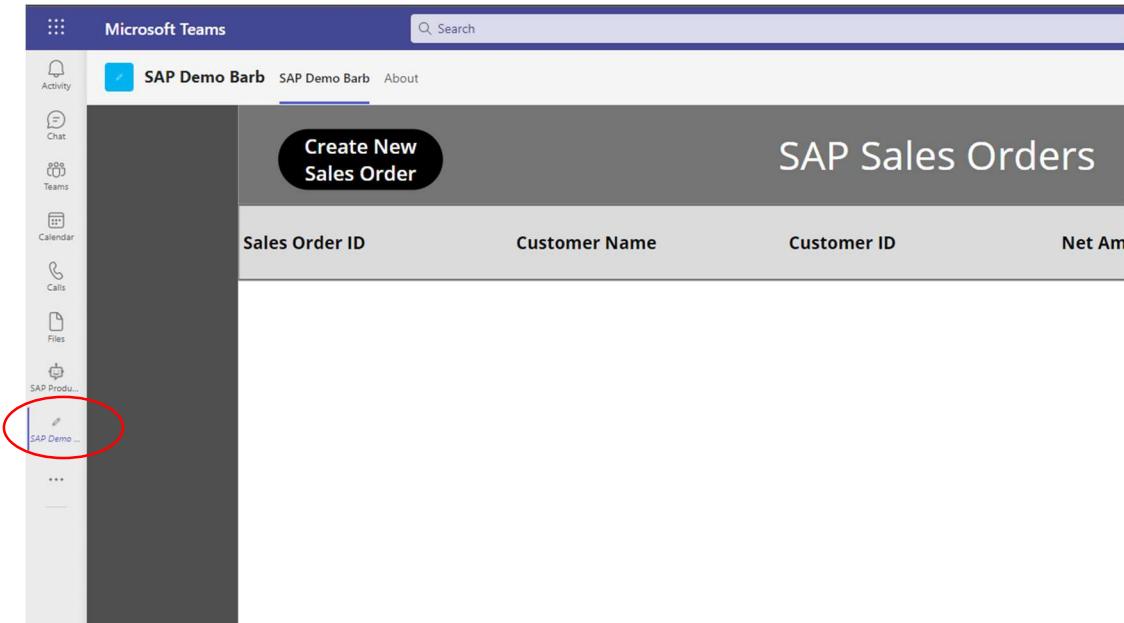
The "Cancel" button is circled in red.

## Leveraging Power Apps with SAP on Azure Workshop

7. Add app to Teams. You will see a pop-up with all the app details. Click on **Add** to add the app into Teams.



8. App is now open and ready for use and has been added to the left-hand navigation pane as an app.



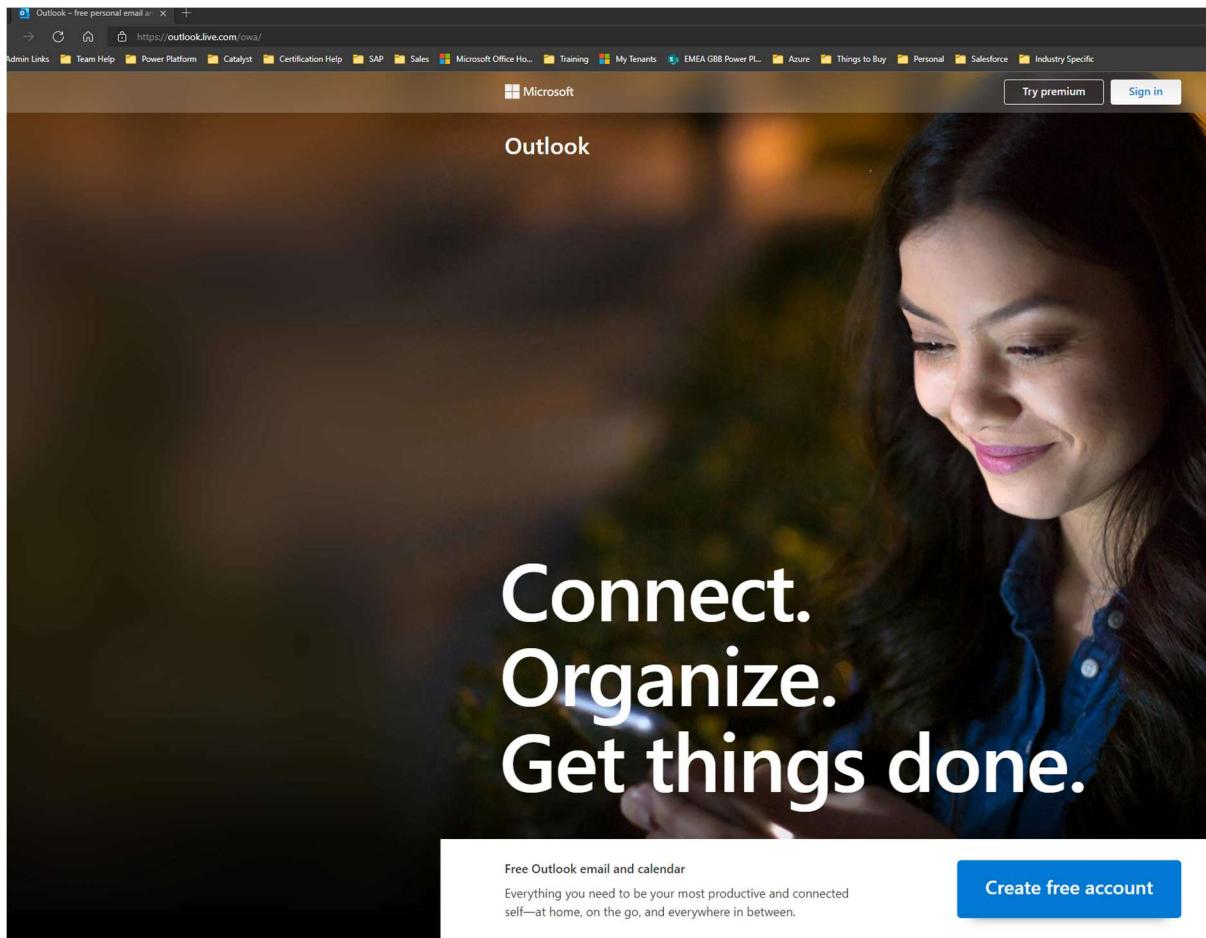
**Congratulations!** You have completed the lab!

# Appendix

## How to create a Trial Environment?

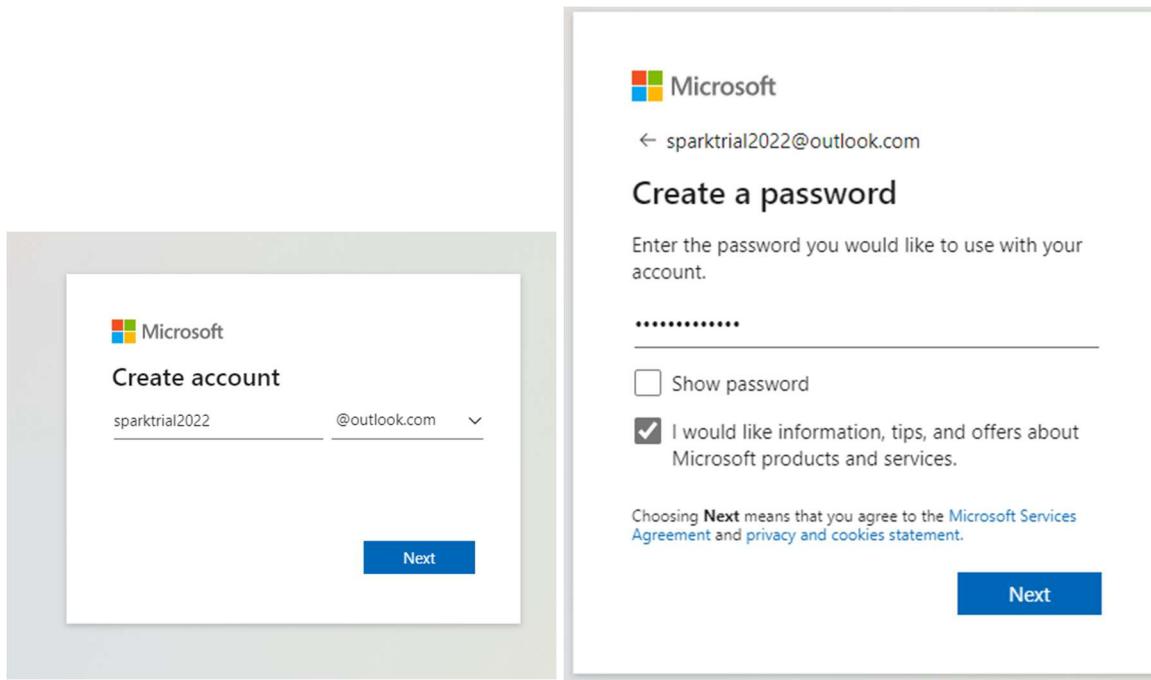
1. Create outlook account

Go to <http://outlook.com/>



Choose Create free account

2. Sign up for the account



3. Create Office 365 trial tenant. Go to [Office 365 E5 Trial](#) and click in Free Trial



## You've selected Office 365 E5 Trial

### 1 Let's get you started

Enter your work or school email address, we'll check if you need to create a new account for Office 365 E5 Trial.

Email

sparktrial2022@outlook.com

Next

### 2 Tell us about yourself

### 3 How you'll sign in

### 4 Confirmation details

### What is Office 365 E5 Trial?

Fully installed Office apps for PC and Mac



### Premium services



### Other benefits

- Unlimited personal cloud storage with qualifying plans
- Email hosting with 100 GB mailbox
- Online & desktop versions of Office applications
- Free FastTrack deployment support with 150+ seats

### Trial details

25 users allowed in 1-month free trial

Click Next and then Set Up account

Answer the questionnaire.

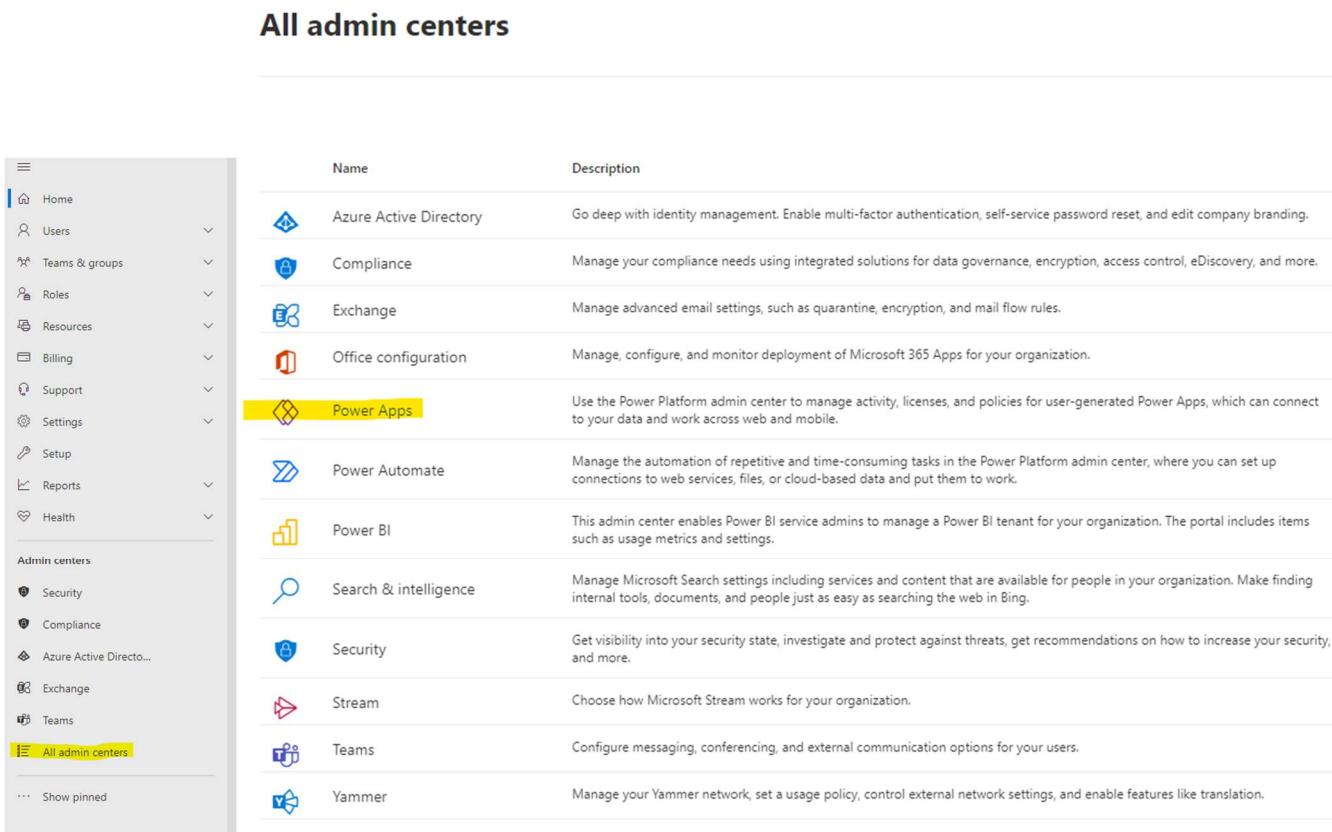
Make note of your admin account ie MarysolMantilla@Spark2022791.onmicrosoft.com

Click Get started

## Leveraging Power Apps with SAP on Azure Workshop

### 4. Exit the Initial setup

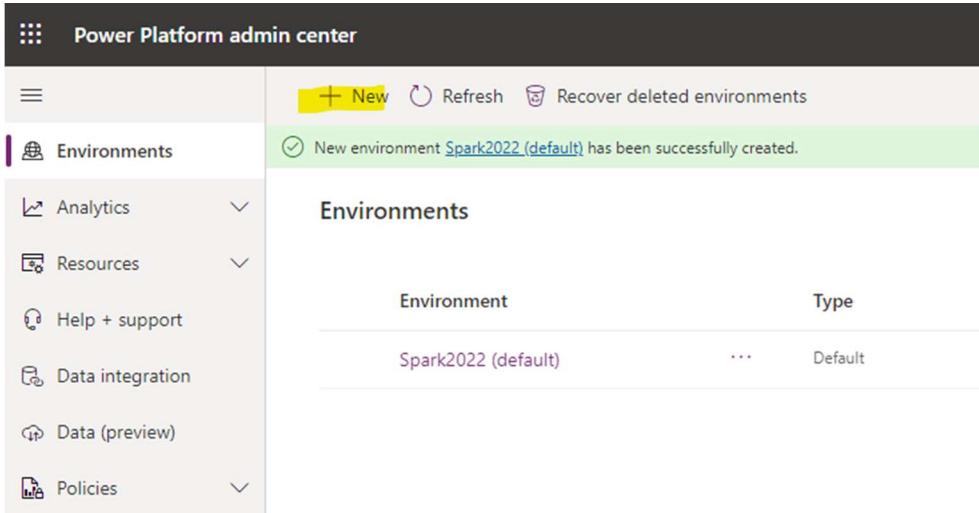
On the panel on the left choose All admin centers



The screenshot shows the Microsoft 365 Admin Center interface. On the left, there's a navigation sidebar with various categories like Home, Users, Teams & groups, Roles, Resources, Billing, Support, Settings, Setup, Reports, and Health. Below these are sections for Admin centers: Security, Compliance, Azure Active Directo..., Exchange, Teams, and 'All admin centers', which is highlighted with a yellow box. The main content area is titled 'All admin centers' and lists several admin centers with their names and descriptions:

Name	Description
Azure Active Directory	Go deep with identity management. Enable multi-factor authentication, self-service password reset, and edit company branding.
Compliance	Manage your compliance needs using integrated solutions for data governance, encryption, access control, eDiscovery, and more.
Exchange	Manage advanced email settings, such as quarantine, encryption, and mail flow rules.
Office configuration	Manage, configure, and monitor deployment of Microsoft 365 Apps for your organization.
<b>Power Apps</b>	Use the Power Platform admin center to manage activity, licenses, and policies for user-generated Power Apps, which can connect to your data and work across web and mobile.
Power Automate	Manage the automation of repetitive and time-consuming tasks in the Power Platform admin center, where you can set up connections to web services, files, or cloud-based data and put them to work.
Power BI	This admin center enables Power BI service admins to manage a Power BI tenant for your organization. The portal includes items such as usage metrics and settings.
Search & intelligence	Manage Microsoft Search settings including services and content that are available for people in your organization. Make finding internal tools, documents, and people just as easy as searching the web in Bing.
Security	Get visibility into your security state, investigate and protect against threats, get recommendations on how to increase your security, and more.
Stream	Choose how Microsoft Stream works for your organization.
Teams	Configure messaging, conferencing, and external communication options for your users.
Yammer	Manage your Yammer network, set a usage policy, control external network settings, and enable features like translation.

### 5. Click New



The screenshot shows the Power Platform admin center. The left sidebar has sections for Environments, Analytics, Resources, Help + support, Data integration, Data (preview), and Policies. The main area is titled 'Environments' and shows a table of environments:

Environment	Type
Spark2022 (default)	Default

A green success message at the top right says 'New environment Spark2022 (default) has been successfully created.'

### 6. Enter the details as below for the new environment

## Leveraging Power Apps with SAP on Azure Workshop

New environment X

(i) This operation is subject to [capacity constraints](#)

Name \*

Region \*  
 ▼  
A local region can provide quicker data access

Type (i) \*  
 ▼

Purpose

Create a database for this environment? (i)  
 Yes

Make sure you Create a database for this environment

And choose Next

**Note:** When saving if you get an error. In a separate tab navigate to [Business Apps | Microsoft Power Apps](#) and choose Start Free. Add your phone number and Get started.

Go back and save the environment

Power Platform admin center

+ New ↻ Refresh ↻ Recover deleted environments

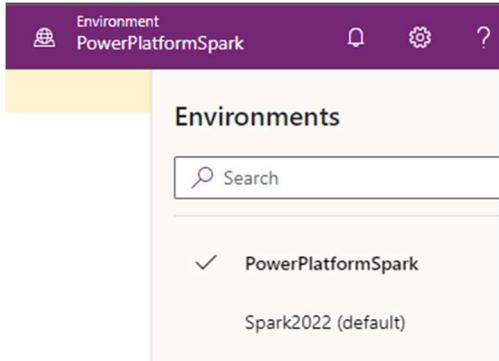
(i) Environments ✓ New environment [PowerPlatformSpark](#) has been successfully created.

(i) Analytics (i) Resources (i) Help + support (i) Data integration (i) Data (preview)

Environment	Type	State
PowerPlatformSpark	... Trial (29 days remaining)	Ready
Spark2022 (default)	... Default	Ready

7. Go to <http://make.powerapps.com/> and change the environment on the left to your new PowerPlatformSpark

## Leveraging Power Apps with SAP on Azure Workshop



A screenshot of the Microsoft Power Platform environment list. The top navigation bar shows 'Environment' and 'PowerPlatformSpark'. Below the header, there's a search bar with a magnifying glass icon and the word 'Search'. A list of environments is displayed, with 'PowerPlatformSpark' checked and 'Spark2022 (default)' listed below it.

Environment
✓ PowerPlatformSpark
Spark2022 (default)

Use this new environment for your exercises.

# Copyright

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2022 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.