Microsoft Flow in a Day Workshop

Microsoft Flow in a Day

Hands-on labs & Hackathon

Written by Serge Luca for Microsoft

Last update : 5/28/2019

Content

Microsoft Flow in a Day step-by-step hands-on labs & Hackathon 1

Lab 1. Building a time tracking flow 2
Tasks: 2
Optional exercise: interacting with your flow on your mobile 11

Lab 2. Approvals (Part 1) - Travel Approval 16
Tasks: 16

Lab 3. Flow Notifications and Conditions 29
Tasks: 29

Lab 4. Flow control, variables, expressions 32
Tasks: 32

Lab 5. Dynamically add a person's manager as an approver (Approval Part 2) 58
Tasks: 58

Lab 6. Task overdue 61
Tasks : 61

Lab 7. Creating a Business Process Flow 68
Tasks: 68

Lab 8. Call an external API from flow with the HTTP action 90
Tasks: 90

Lab 9. Create a Flow/PowerApps Custom Connector 96
Tasks: 96

Lab 10. (Advanced) Implementing the flow controller pattern 109
Tasks : 109

Hackathon 128

Theme 1. Timeout approval and escalation. 128

Theme 2. Team bots -Request Teams channels 129

Theme 3. Hot Dog, Not Hot dog 129

Theme 4. Twitter Sentiment Analysis 130

Theme 5. Intelligent Customer Service 130

Microsoft Flow in a Day step-by-step hands-on labs & Hackathon

- Labs and demos just require a browser.
- Office 365 E3 tenant (trial)
- Dynamics 365 tenant (trial) for the LUIS intelligent Customer Service Hackathon.

.

Lab 1. Building a time tracking flow

Learning objectives: Building a flow, hello world, button, time.

Duration: 30 minutes

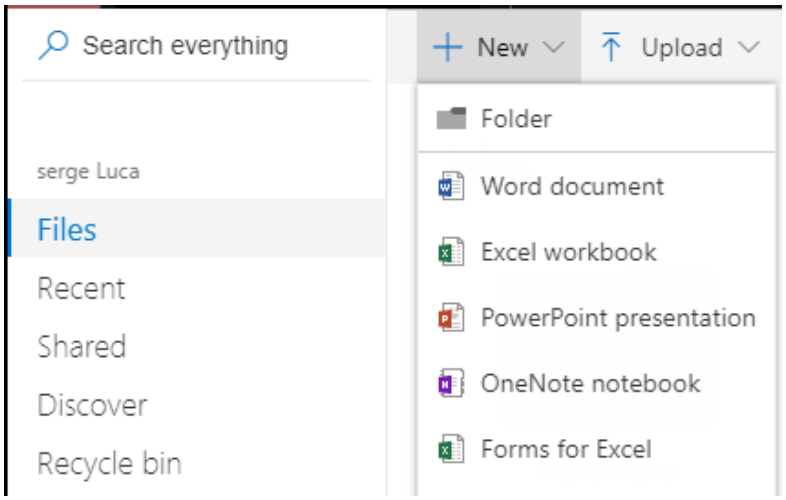
Scenario: When a user pushes a button, the current time and the current location will be saved in an Excel document stored in OneDrive for Business.

Perquisites: Ensure the timesheet.xls in your OneDrive

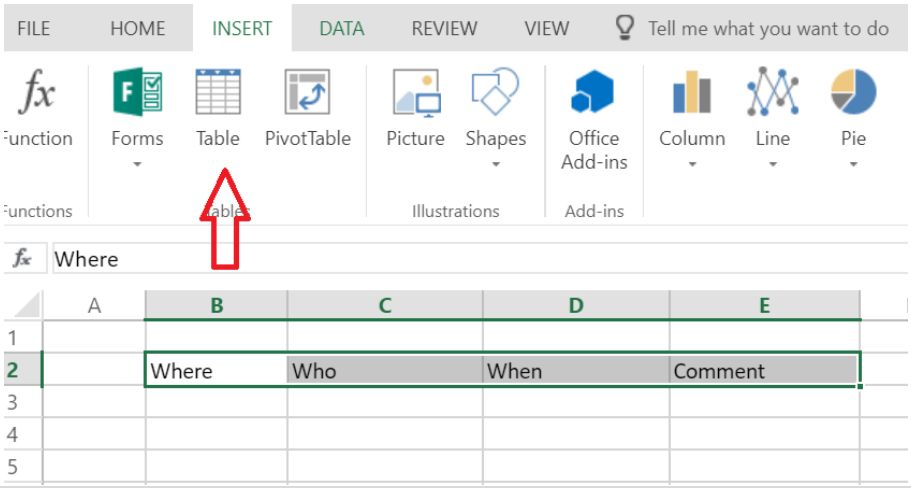
Build this Flow from Blank by following the below steps!

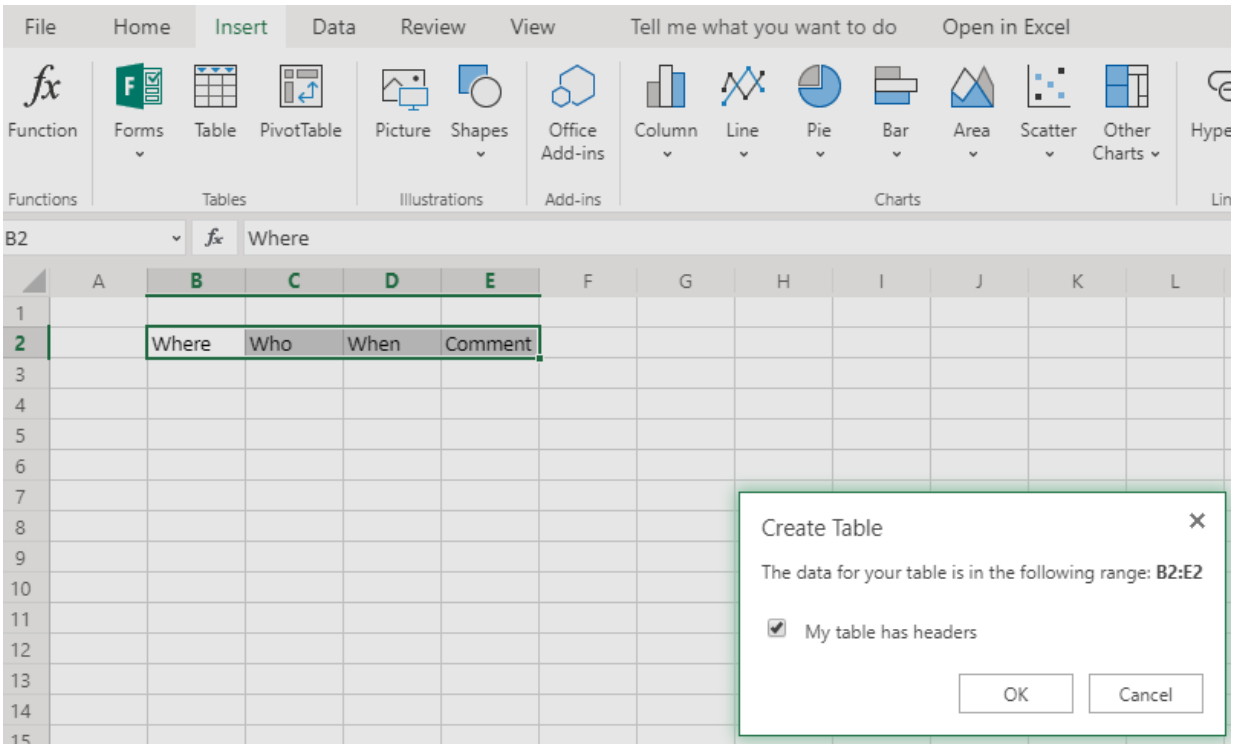
Tasks:

1. Go to your One Drive for Business and create a new Excel workbook:

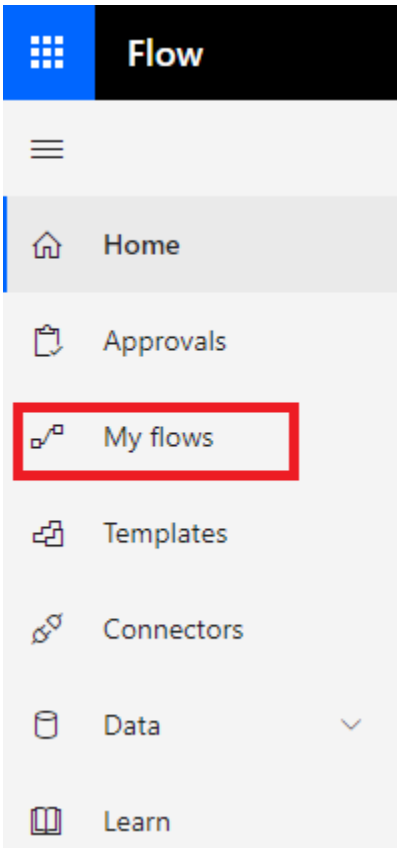


2. Create 4 columns (**where**, **who**, **when, comment**) and format these as a table (check the box, has headers):

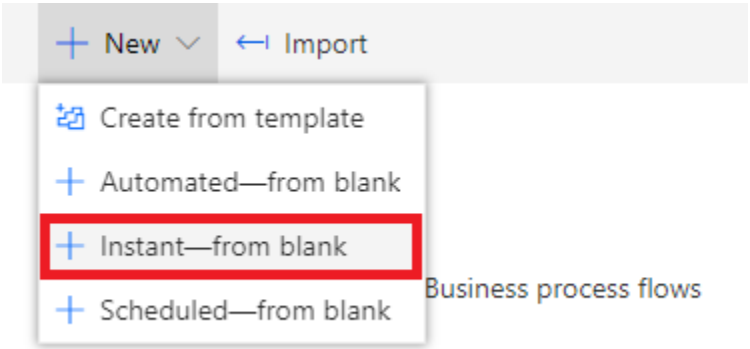




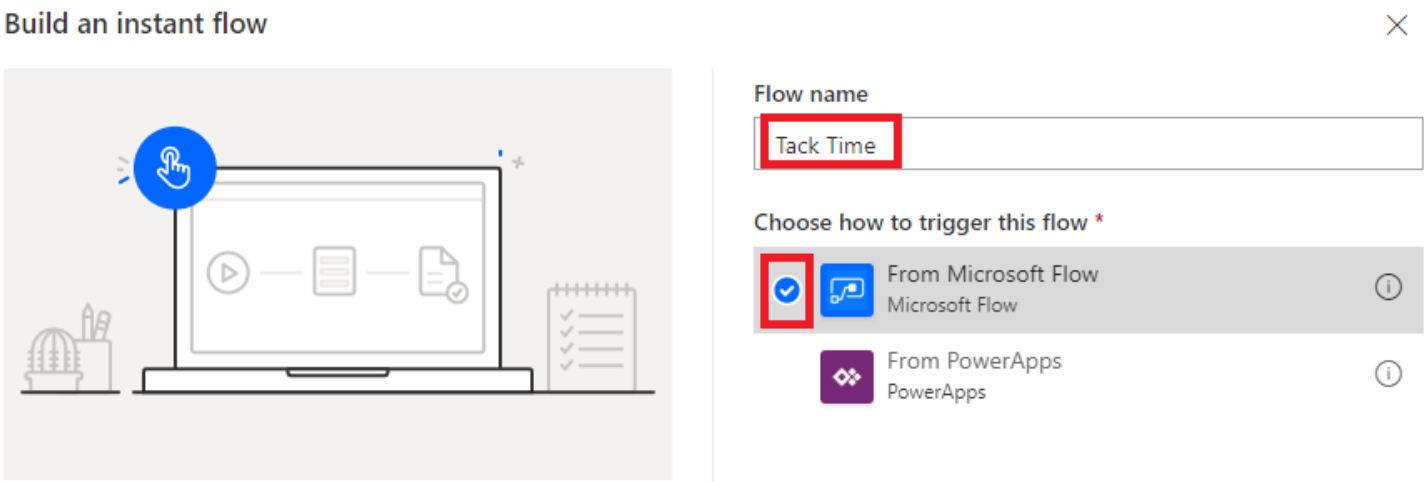
3. Save the file (Save as - Rename) with the name **Timesheet.xlsx**.
4. Let's create the **Track Time** flow: go to flow.microsoft.com and if requested, sign-in.
5. Go to **My flows**



6. **New-Instant from blank**:

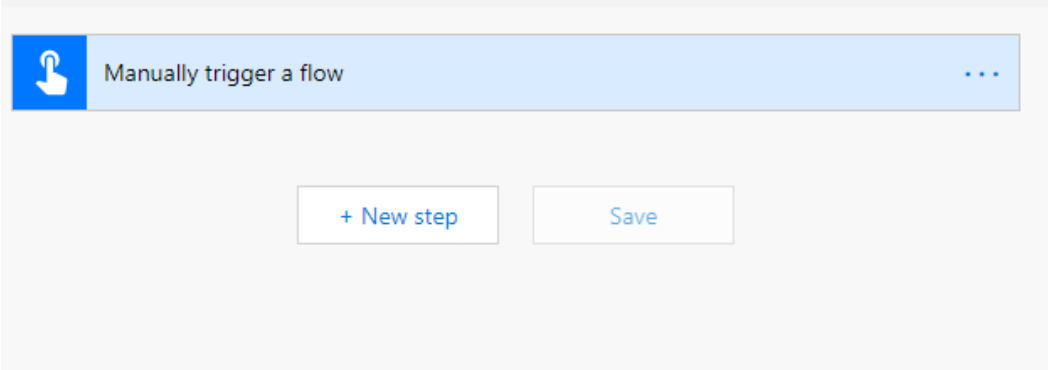


7. In the next window name your flow **Track Time** and select the trigger "From Microsoft Flow**":**



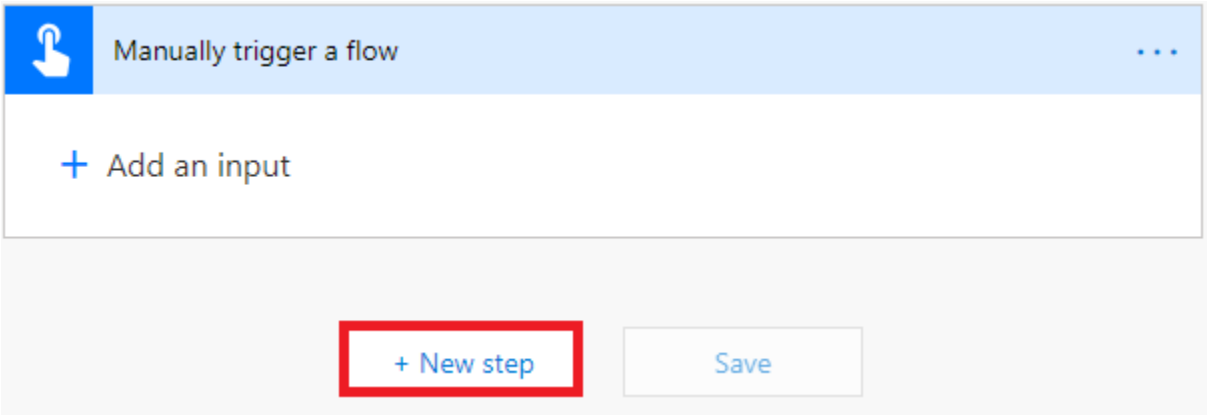
8. Click Create.

   A flow will be generated with the trigger "Manually trigger a flow":

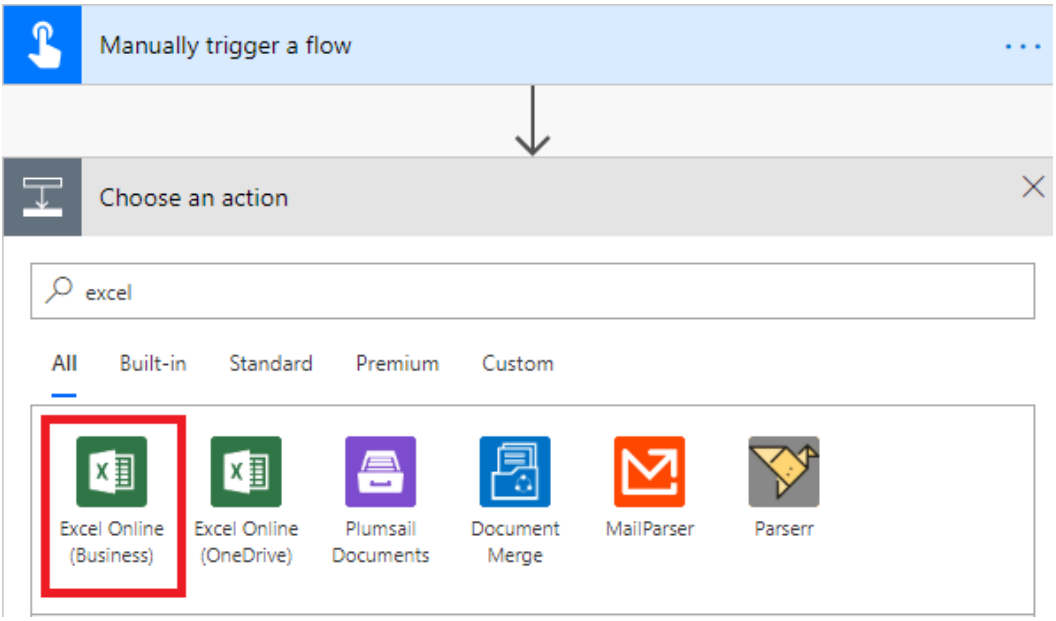


   Note: This trigger will start the flow by pressing a button. The button can also be visible in the flow mobile app, including in the home screen of a smartphone. If needed, a form can show up when the user pushes the button. But in any case, some default information like the current user coordinates and current time are captured automatically by flow when the button is pressed.

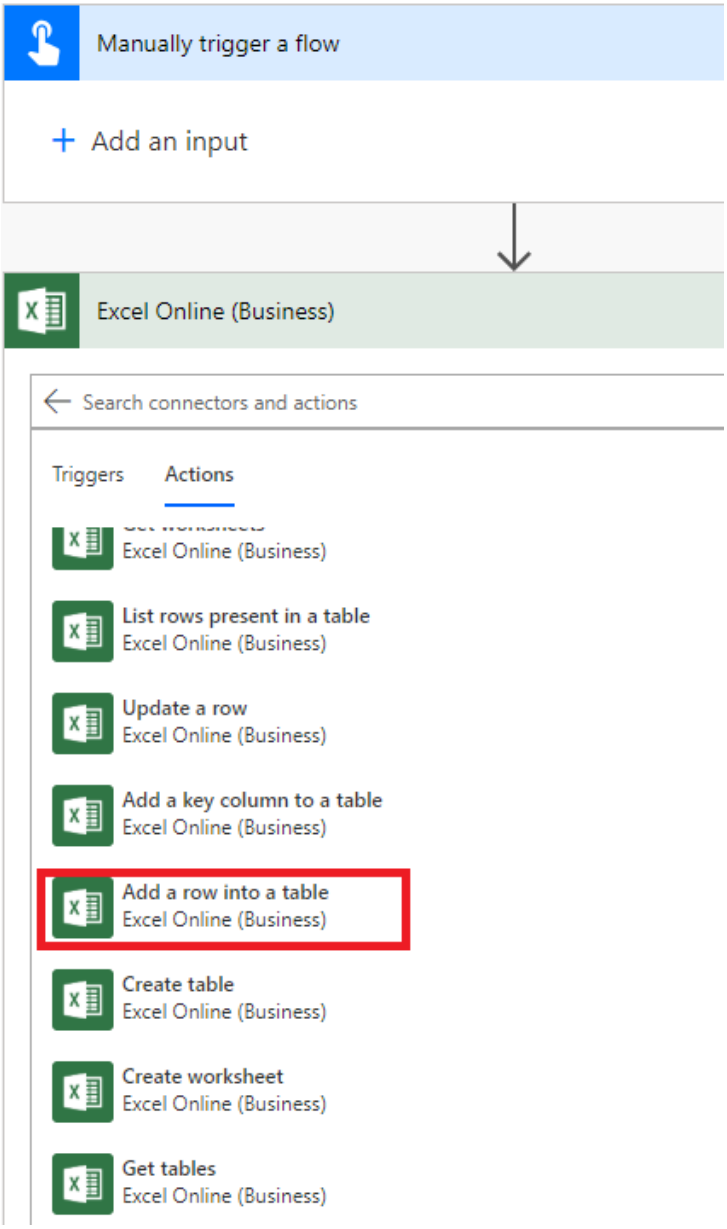
9. Click on **New step:**



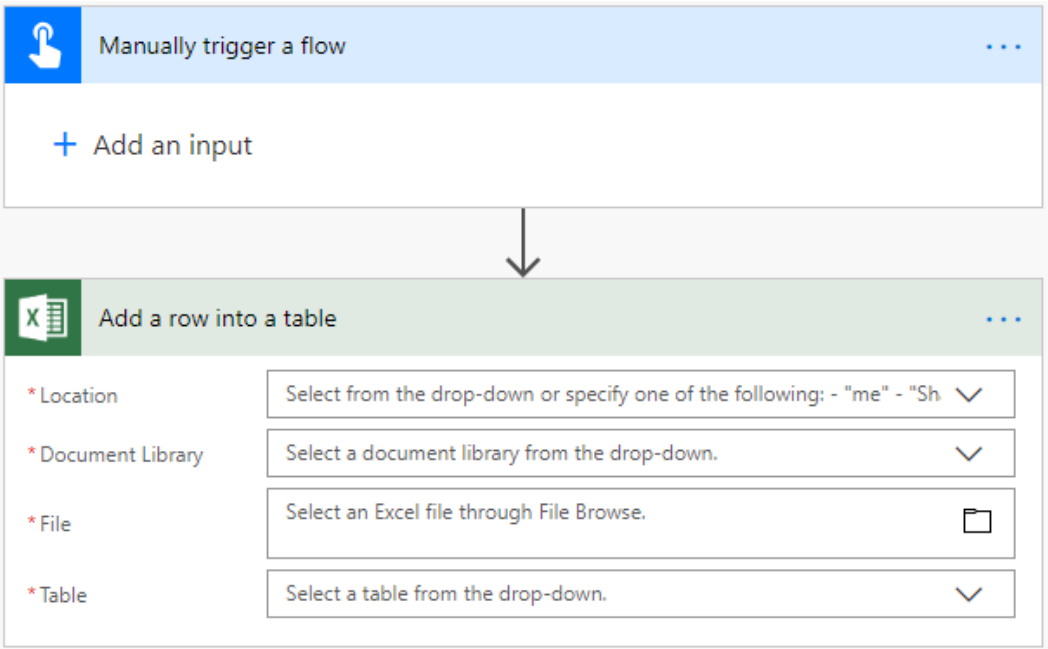
10. and then in **Choose an action**, type Excel to find the Excel Online (Business) connector (not Excel Online OneDrive; be careful here many users select the wrong connector); select it :



11. Add the **Excel Online (Business) Connector** and select the associated action **Excel Online (Business) Add a row into a table** action.



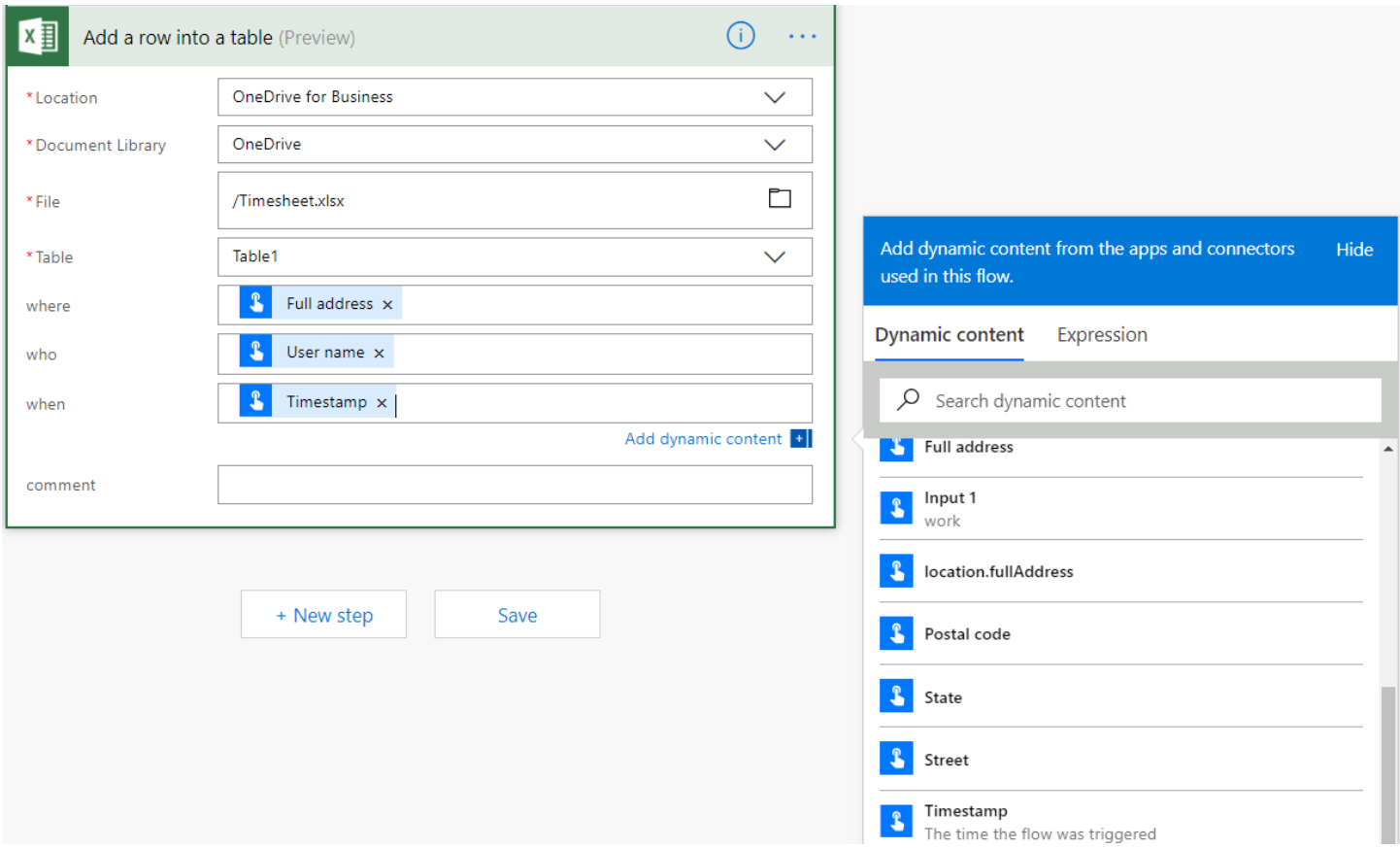
12. You should have this:



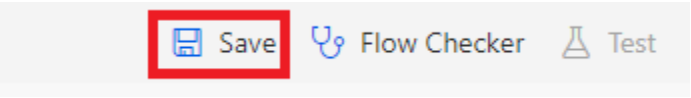
13. You need to configure this action; add a **Location, Document Library, File** and **Table** based upon the location of your spreadsheet in OneDrive for Business**.**

    What we want to achieve is adding the current user address location and time stamp into the spreadsheet. Doing this will create a connection with the current user account.

14. Once your table is launched you will see additional fields appear (column names) then fill in the action properties with **DYNAMIC CONTENT** (select them with the dropdown) with the following values:



15. Save the flow (the Save menu in the upper right corner)



16. Run the flow by going to the Flow page and by clicking on the three dots next to **More** then click **Run now** (we can also use the **Test** button).

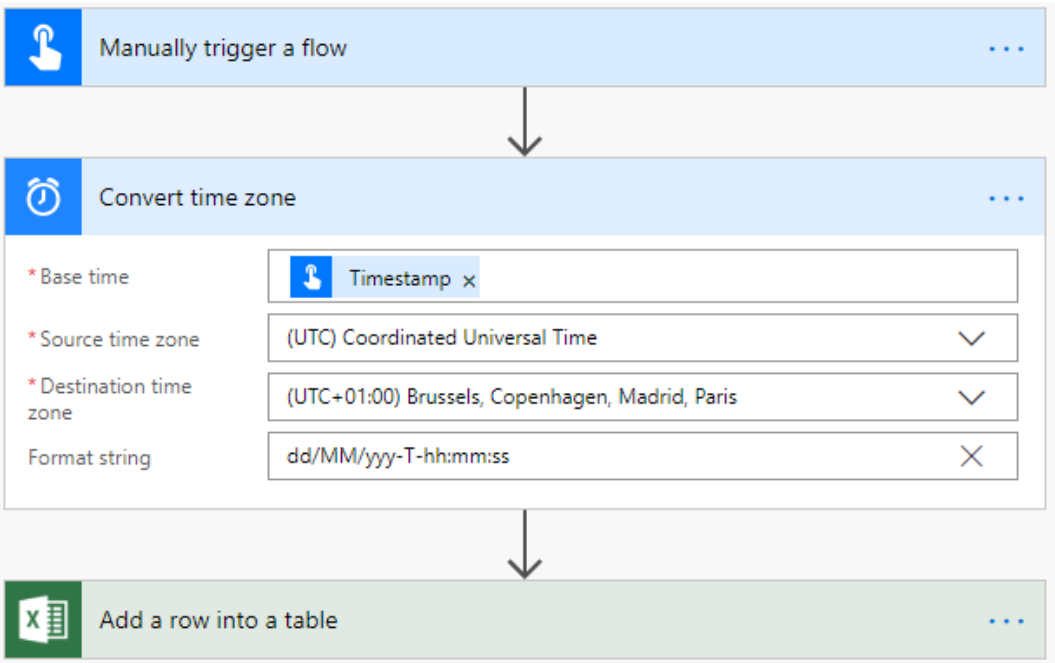
    Note: You may be prompted to consent to providing your location. Click **Allow** to do so.

17. Make sure the flow ran successfully and check your Excel sheet. You should see all the requested information

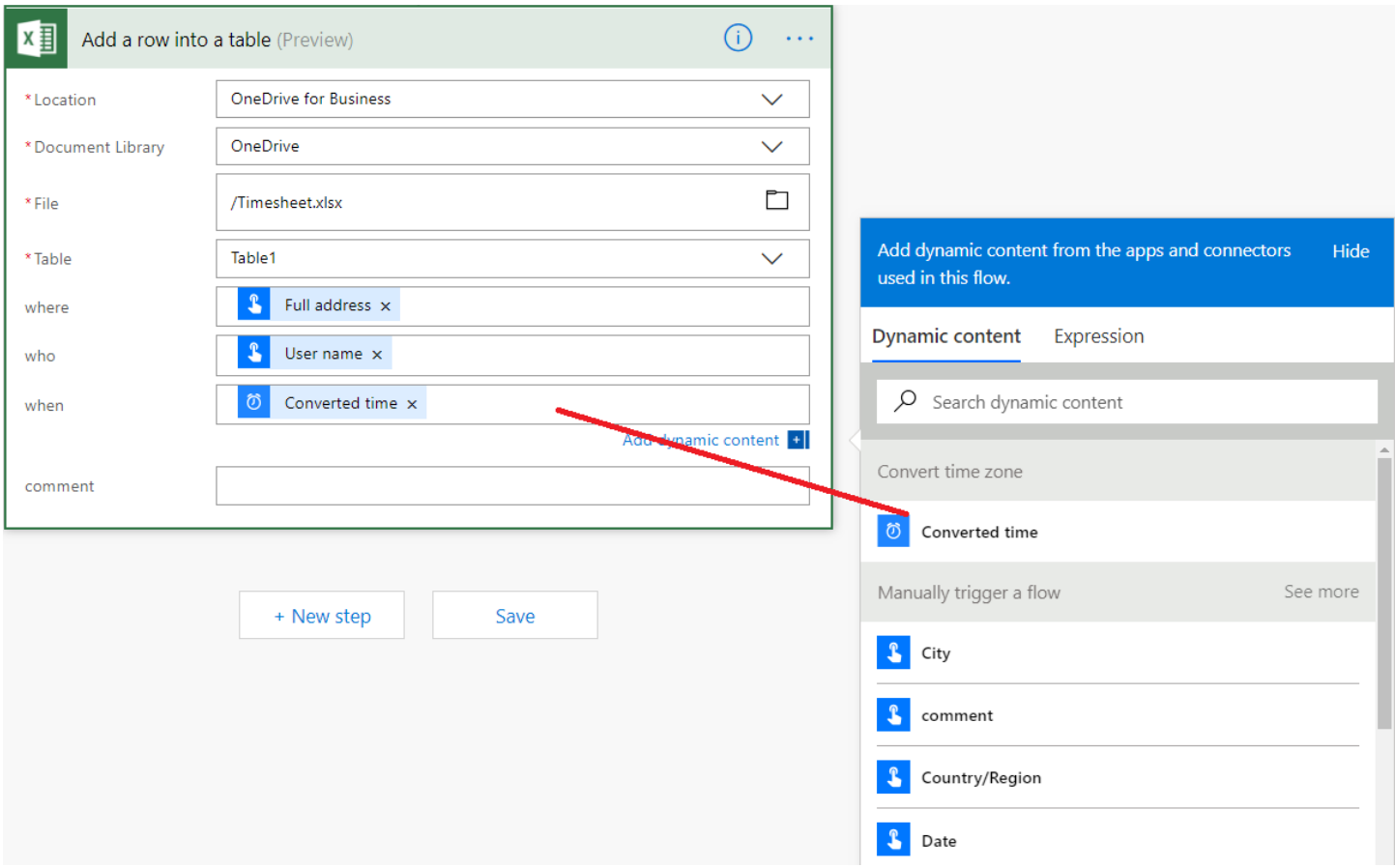
    Note: You might have to refresh the spreadsheet to see the updated values.

where	who	when
Jean en Pierre Carsoellaan 243, 1180 Ukkel, Belgium	Paul Pieter	2018-06-28T07:17:51.35413Z

18. By default, Microsoft Flow uses UTC as its default time zone. Let's display a more user-friendly date/time that is in your time zone.
19. Edit the flow and add a **Convert time zone** action between your button trigger and excel action (search for 'convert' to find it).
20. Select a **Source time zone** and **Destination time zone** based upon your current time zone. In addition, we can use several predefined Date & Time formats, but you can also define your own custom format. Use **custom** and provide your format string **dd/MM/yyyy-T-hh:mm:ss** as illustrated in the next picture:



21. Update the **Add a row into a table** action with the **Converted time** value:



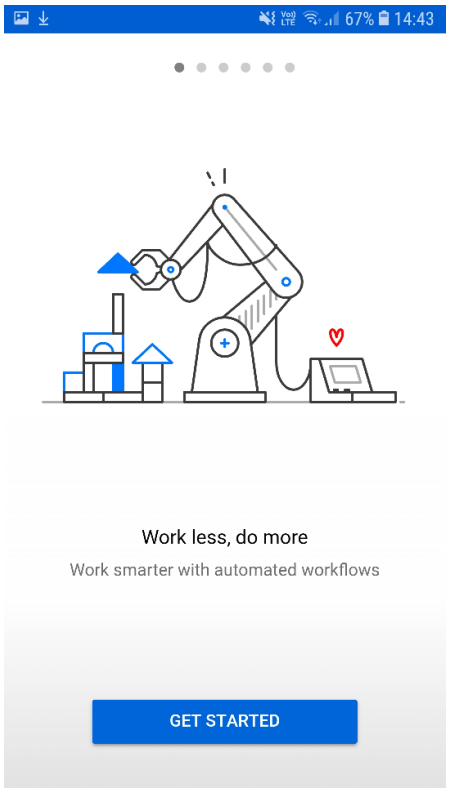
22. Save the flow, run it again and check your Excel sheet:

where	who	when
Jean en Pierre Carsoellaan 243, 1180 Ukkel, Belgium	Paul Pieter	2018-06-28T07:17:51.35413Z
Jean en Pierre Carsoellaan 243, 1180 Ukkel, Belgium	Paul Pieter	28/06/2018-T-09:42:51

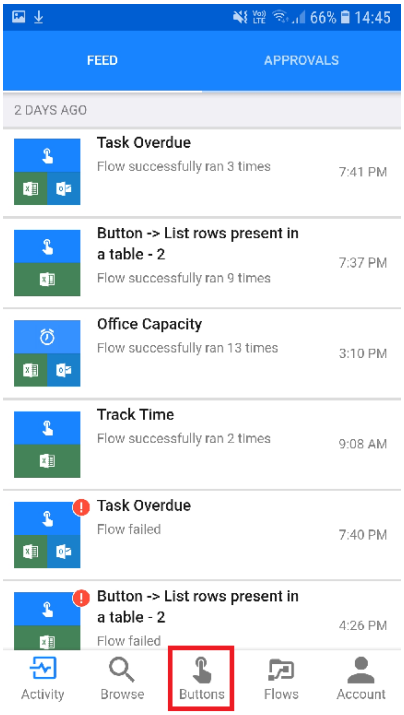
Optional exercise: interacting with your flow on your mobile

Note: this lab has been tested on an Android phone, the look and feel might be a little bit different on iOS.

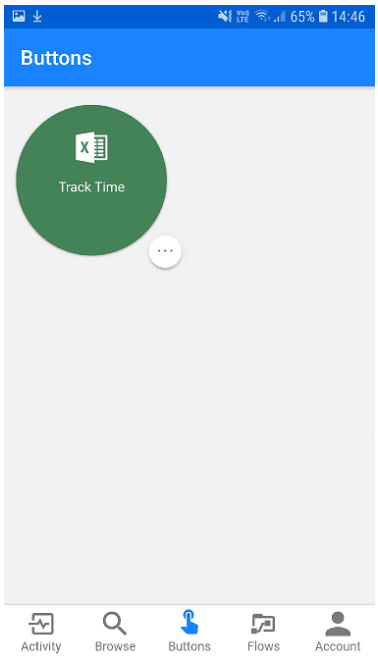
1. Download the **Microsoft Flow** mobile application from the store (Android/iOS).
2. In the Flow mobile app, click on **GET STARTED**



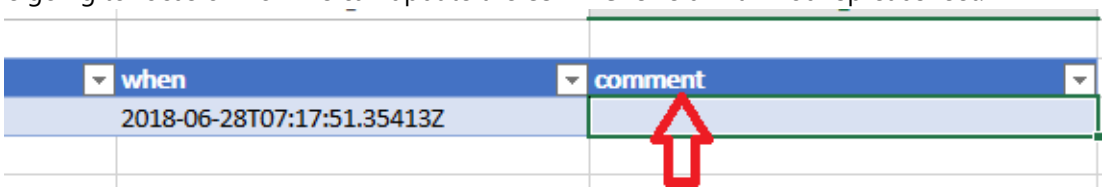
3. Sign in with your account.
4. Click **Buttons**:



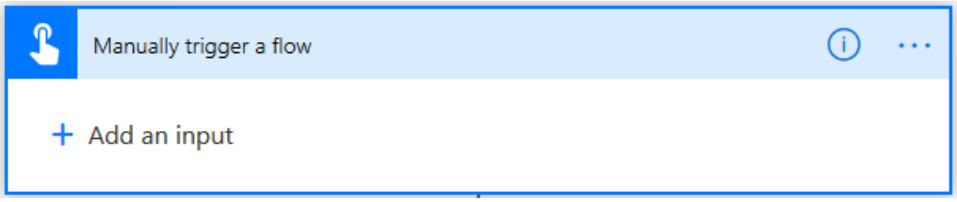
5. Your **Track Time button** should be displayed, and you can go ahead and click the button to execute the flow:



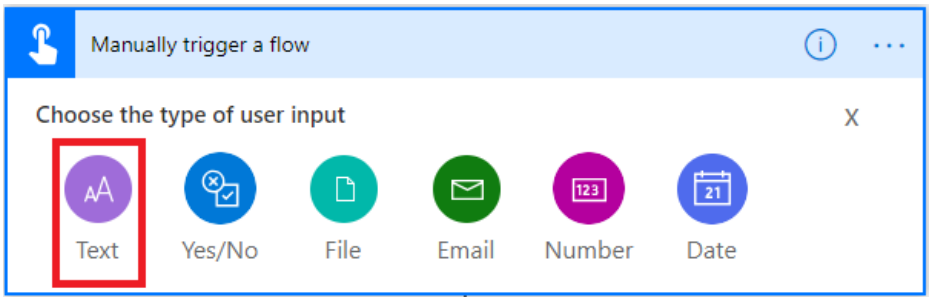
6. Now we are going to focus on how we can update the **comment** field within our spreadsheet:



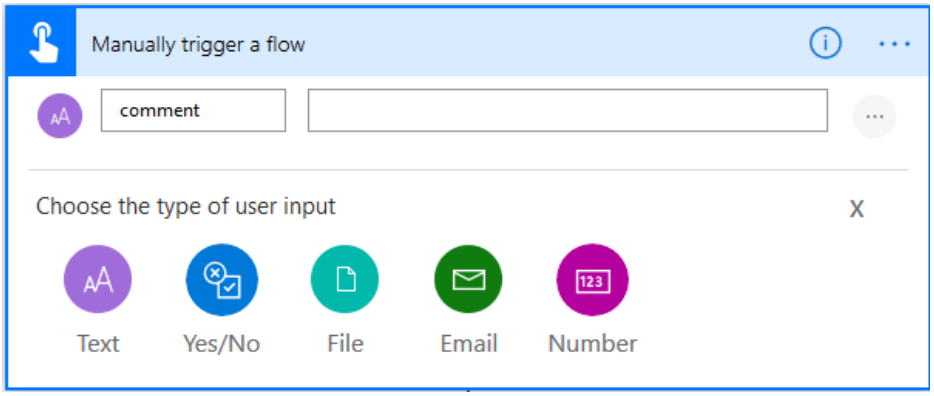
7. Edit the flow, click on the flow trigger (**Manually trigger a flow**) and click on **Add an input:**



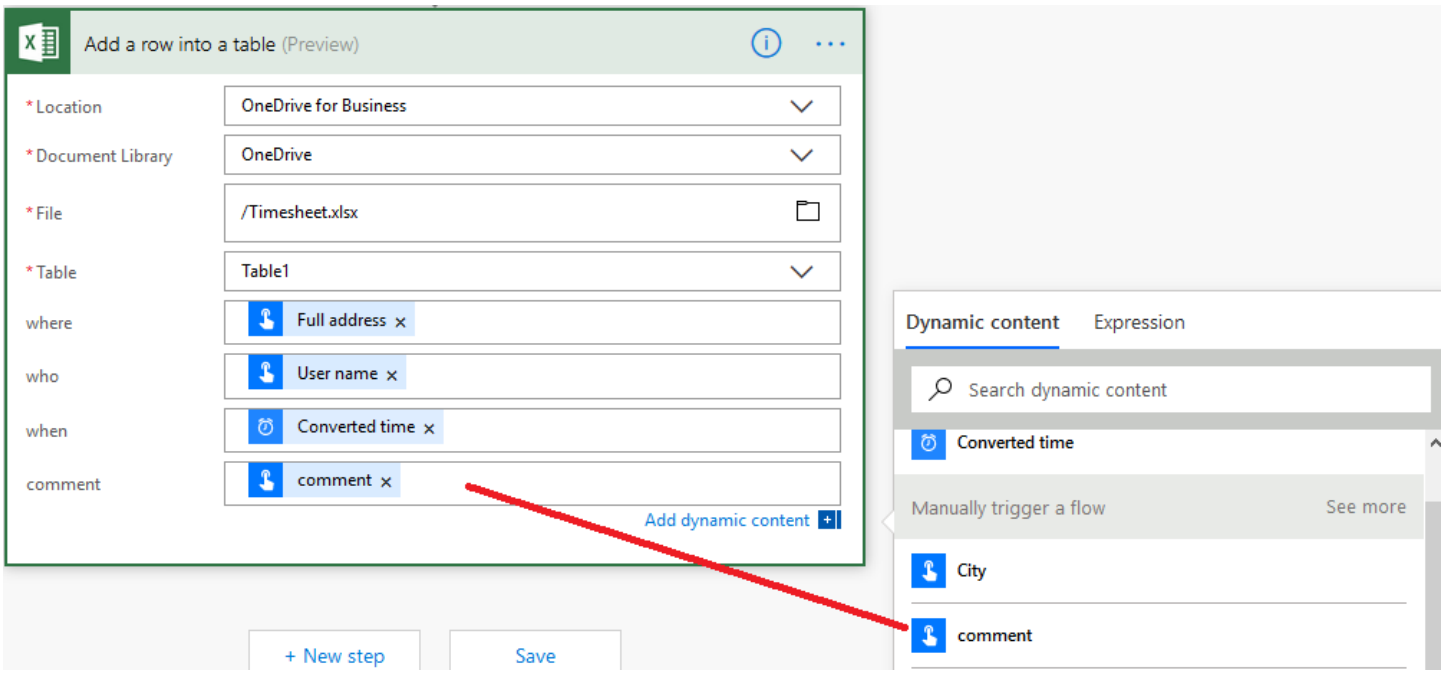
8. There are many options to provide greater interaction with users when they click on a Flow button. Select the **Text option** to add a text field:



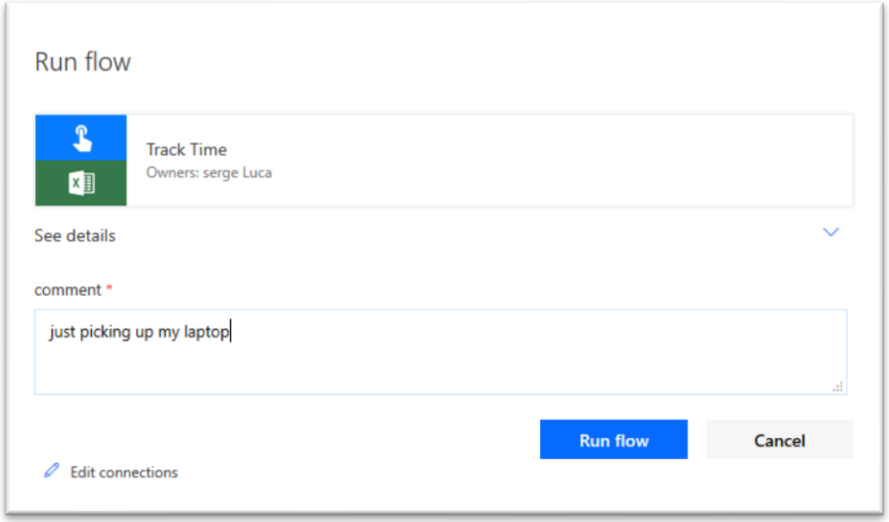
9. Replace **Input** text value with **comment** and provide a single space as default value:



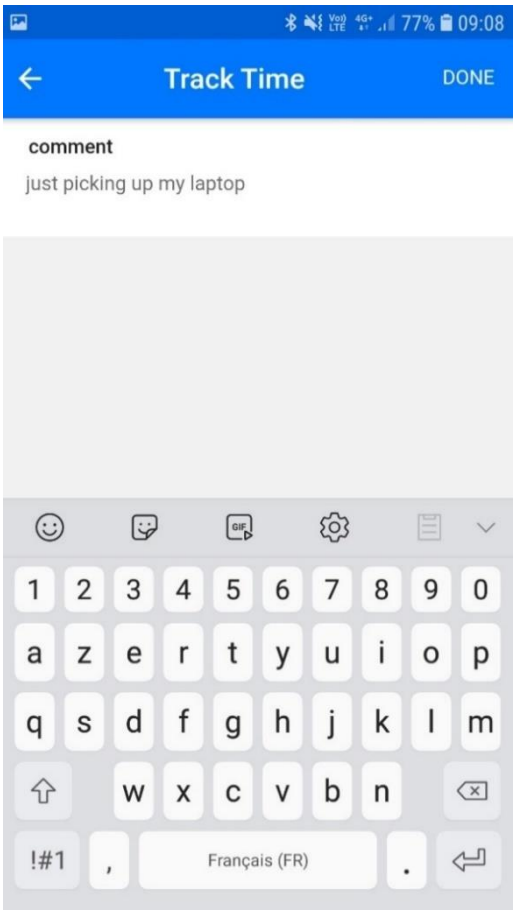
10. Click the action **Add row into a table**. We are going to store the comment field into the spreadsheet's **comment** column:



11. Save the flow and return to the Flow details page. Run the flow by clicking on **Run now** from **...More** menu. If the flow requests your location, accept the option, and fill in the form by providing a comment:



12. If you run it from your mobile, you will see something like this after pushing the button:



13. Check the spreadsheet and you should see your new comment:

where	who	when	comment
Jean en Pierre Carsoellaan 243, 1180 Ukkel, Belgium	Paul Pieter	2018-06-28T07:17:51.35413Z	
Jean en Pierre Carsoellaan 133, 1180 Ukkel, Belgium	Paul Pieter	29/06/2018-T-08:52:42	just picking up my laptop

Lab 2. Approvals (Part 1) - Travel Approval

Learning objectives: Approvals and conditions.

Duration: 20 minutes.

Scenario: A user stores his/her travel information in a SharePoint list named Travels. When a new travel request is created, a flow will be triggered and will ask a manager to Approve/Reject.

Prerequisites: Each student must have a dedicated custom SharePoint list named Travels_<name>. The list must have 3 fields: Title, Amount (currency), Status (single line of text).

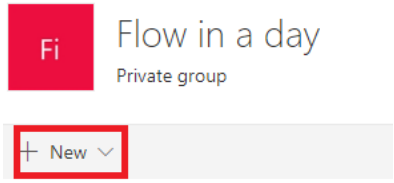
If you want to take a look at the solution: select the **Submit a travel request for approval** template.

If you want to start the lab from scratch:

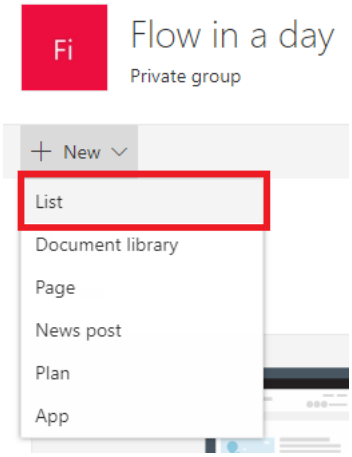
Tasks:

1. You will have to create your own custom SharePoint list. Go to your SharePoint site. The site url is:**<tenant URL>sites/flowinaday.**
2. Create a new list:

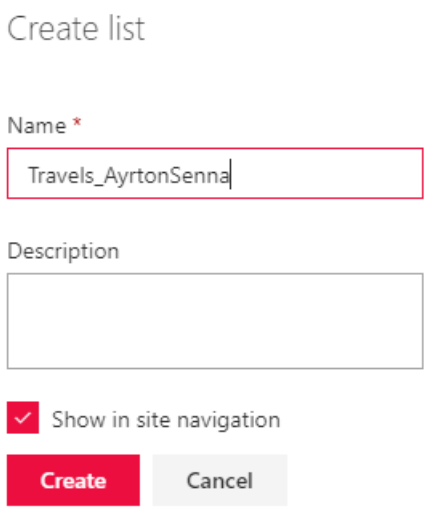
   a. Click **New**:

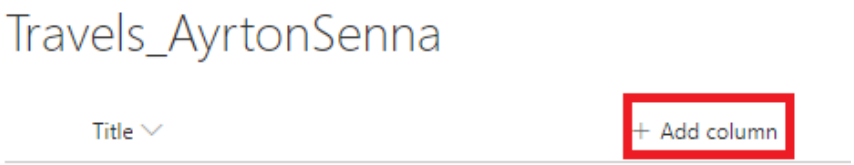
   b. Click **List**:

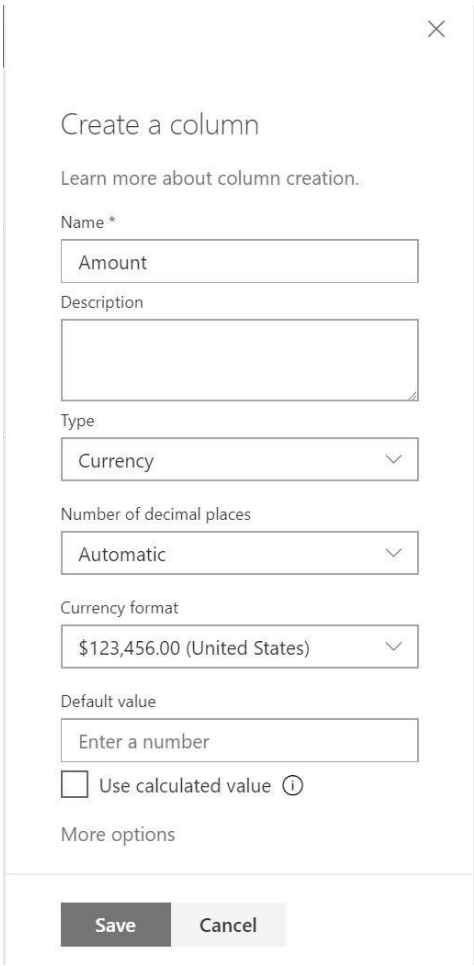
3. Provide a list name and make sure the list is unique, like **Travels_<YourFirstnameYourLastname>:**



4. Click **Create** and the list will be generated. In the list add a new column:



5. Select the type **Currency** and provide the following information:



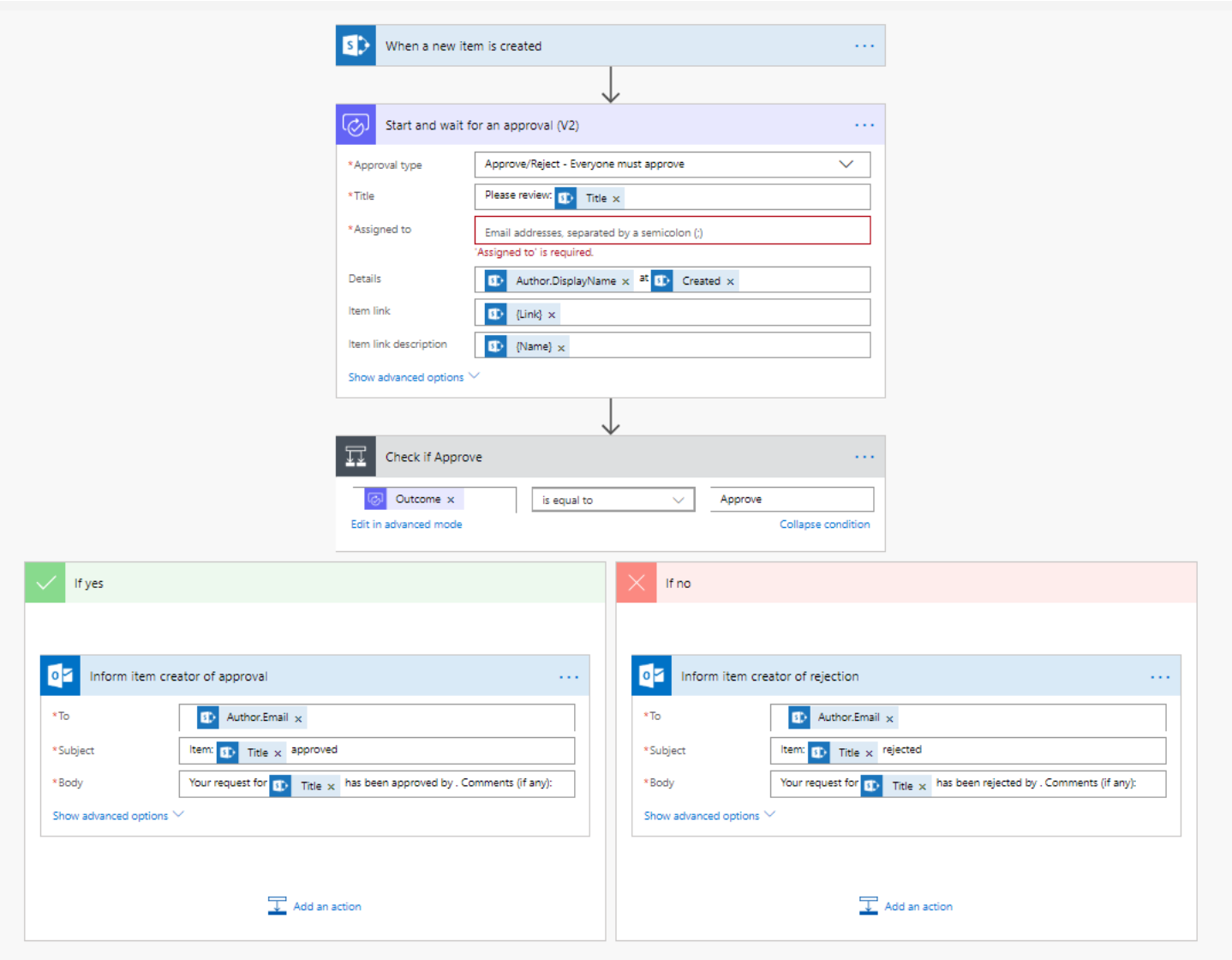
6. Click **Save**.

7. Now create a **Single line of text** column named **Status**.

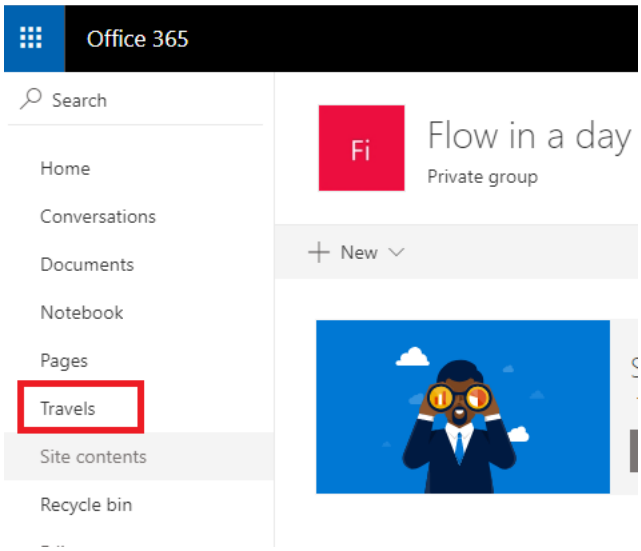
8. Create a flow from the template **Start Approval when an item is added.**

   Note: If you have not connected to Office 365, SharePoint or Approvals before, you will need to provide your credentials to create connections to these services.

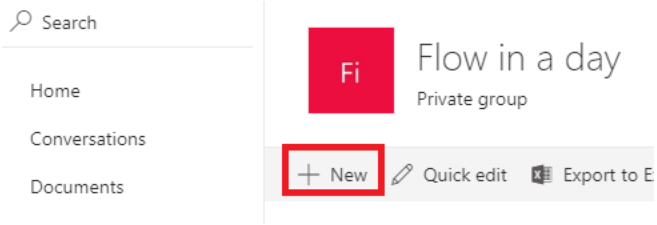
9. Once you have clicked Continue from the **Start approval when a new item is added**, the flow will look like this:



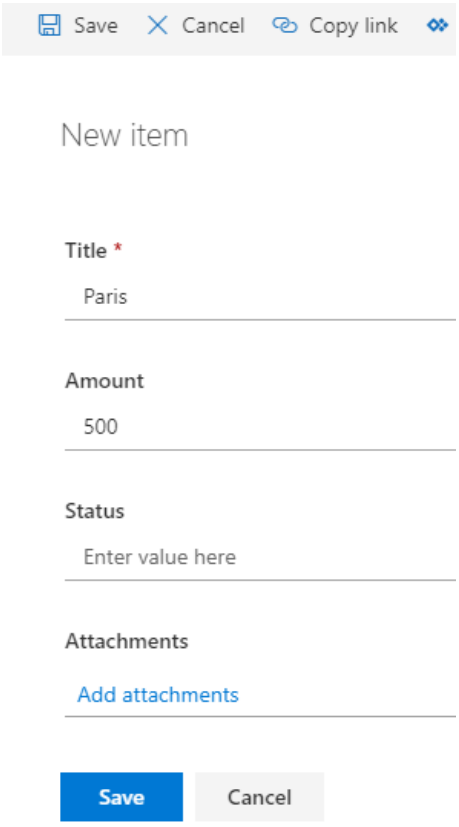
10. In the trigger, set your SharePoint site url and retrieve the list name, **Travels**<**FirstnameLastname**>.
11. In the **Assign To**, set to the email address that is used to access Microsoft Flow. (in a real world scenario, the flow will retrieve the current user flow manager and we can define it in the Start an approval action, as we will do it in a further exercise).
12. You won't need to change the **If yes** and **If no** conditions.
13. Name the flow **Approve travel** and save it.
14. Add a new travel request in the SharePoint Travels list; this should trigger the flow:
    a. Click on the **list name** on the left side menu:



b. Click on New:

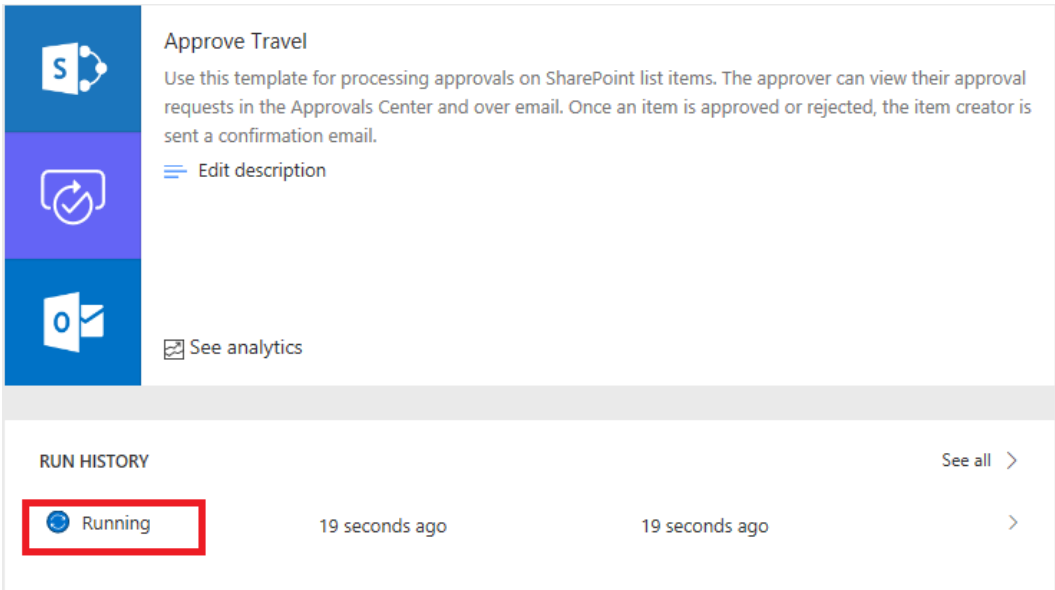


15. Fill in the form like this (keep the Status field empty):

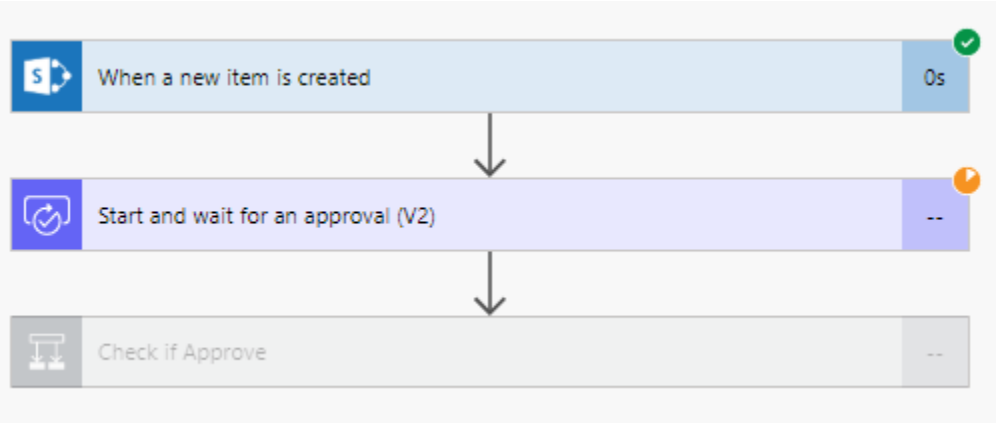


16. Save the new record, the flow will trigger automatically.

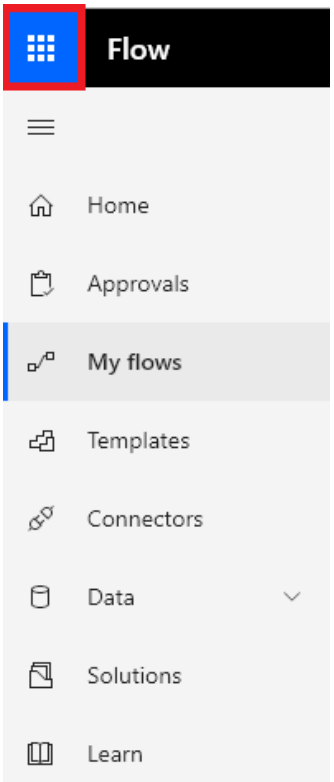
17. If you check the flow status, it should be **running**:



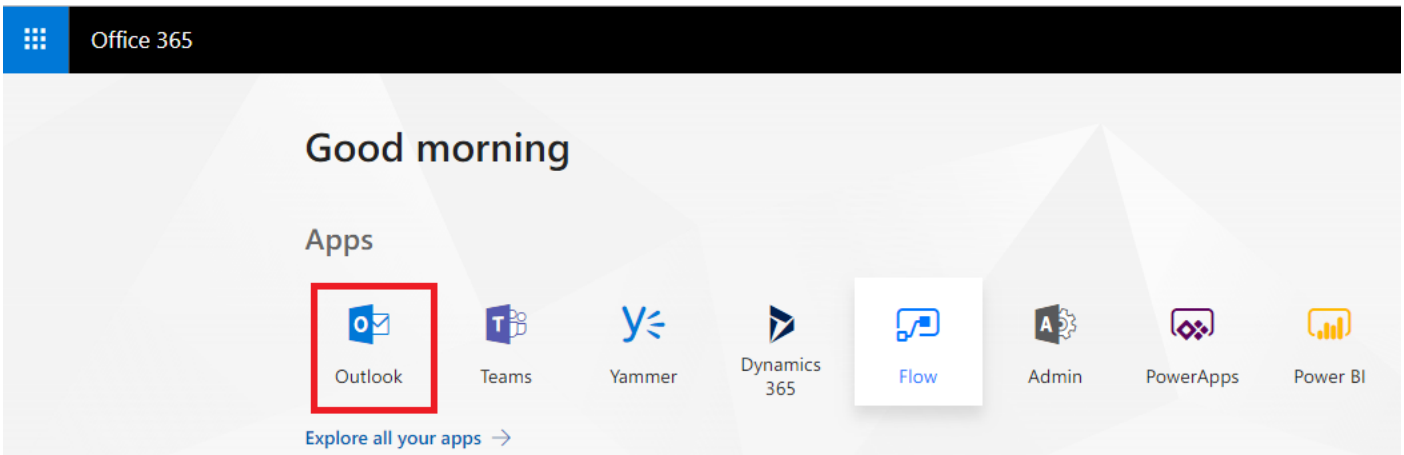
18. If you click the flow can you see where this flow instance is. Click Running and you will see this:



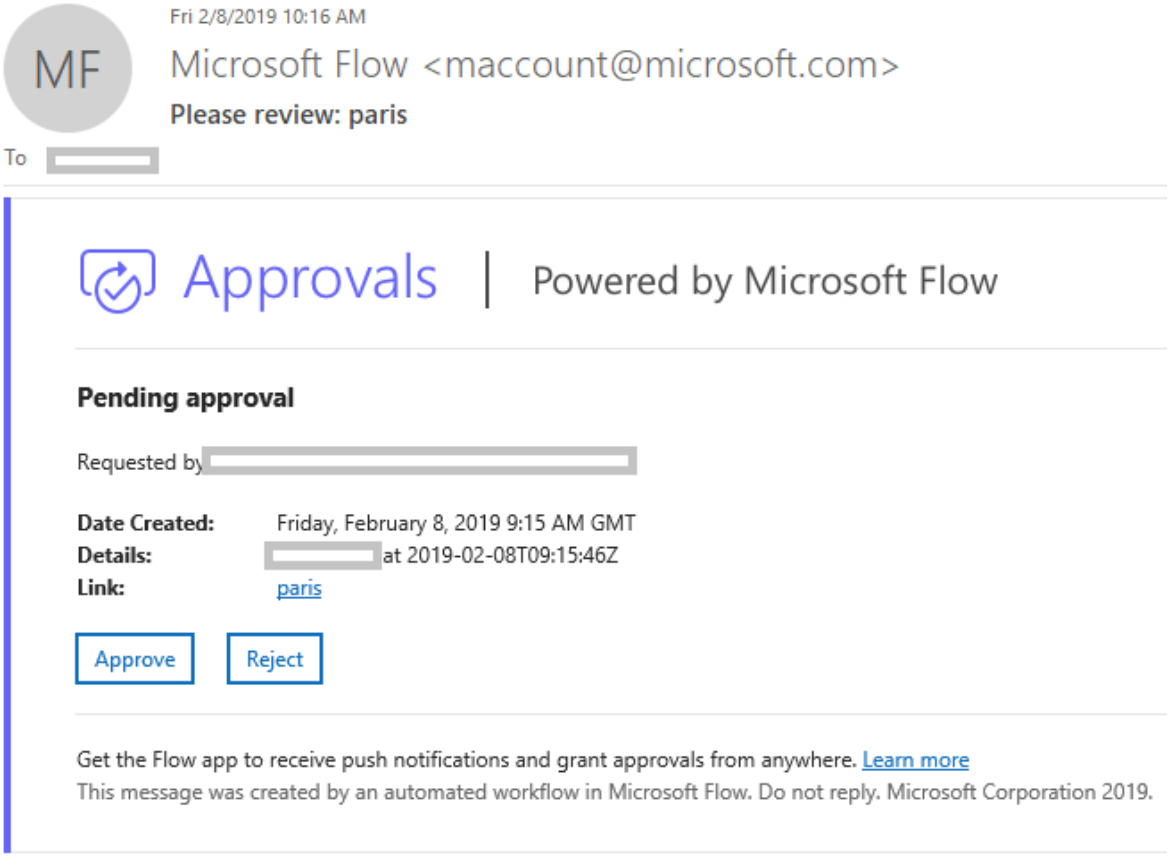
19. The flow is waiting for an approval. Approvals send an e-mail to the person(s) in charge of the approval.
20. Check your e-mail to display your approval e-mail:
    a. To check your e-mails right click on the app launcher Flow button to open a link in a new tab



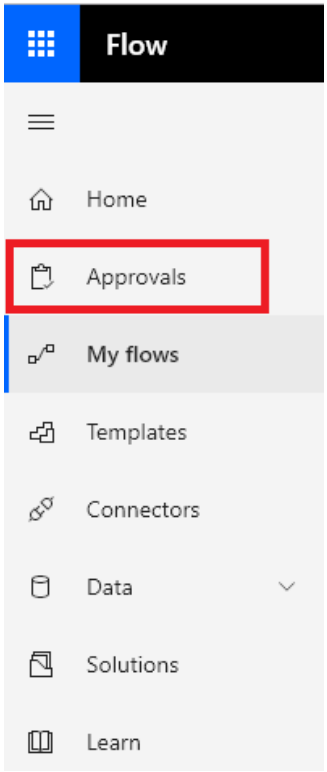
    b. In the list of Apps, click **Outlook**:

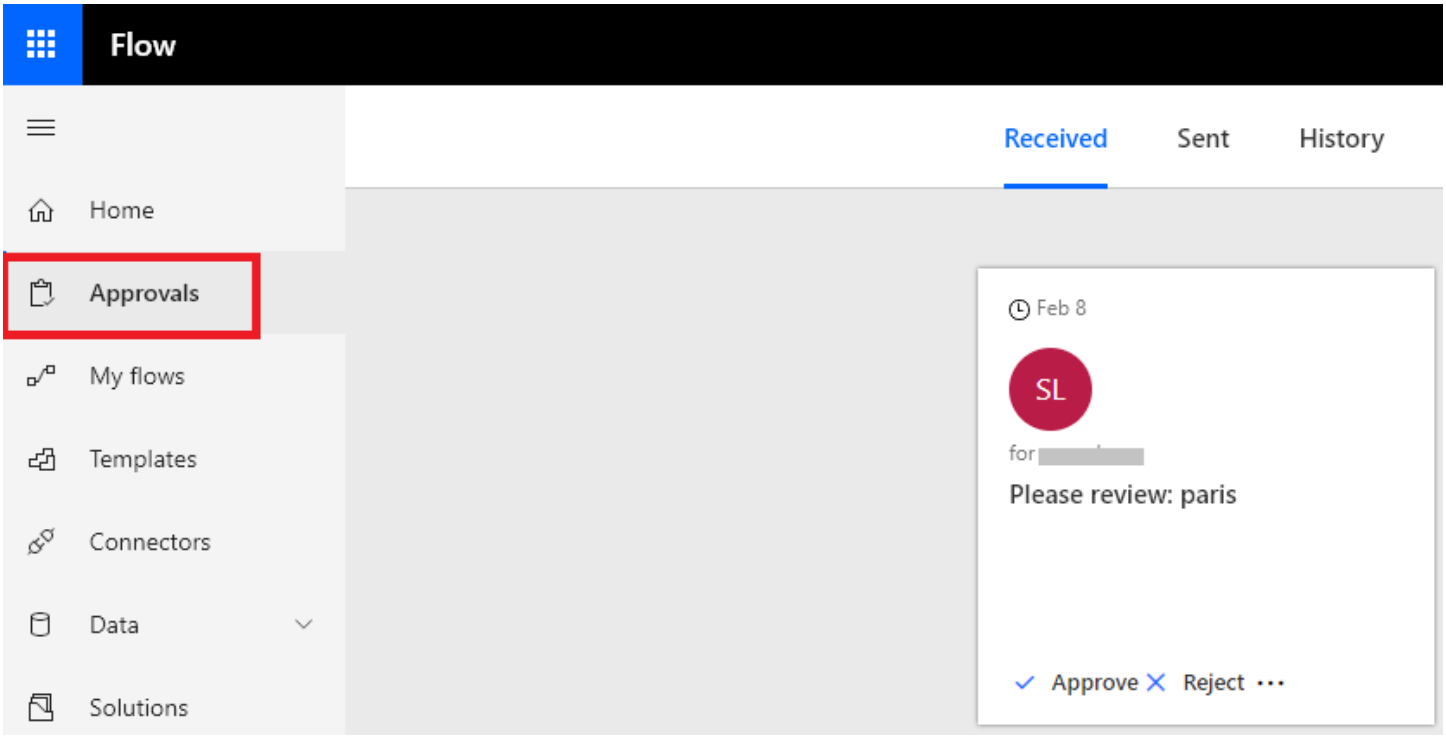


c. In the Focused link or in the Other link you should see your Flow approval e-mail:

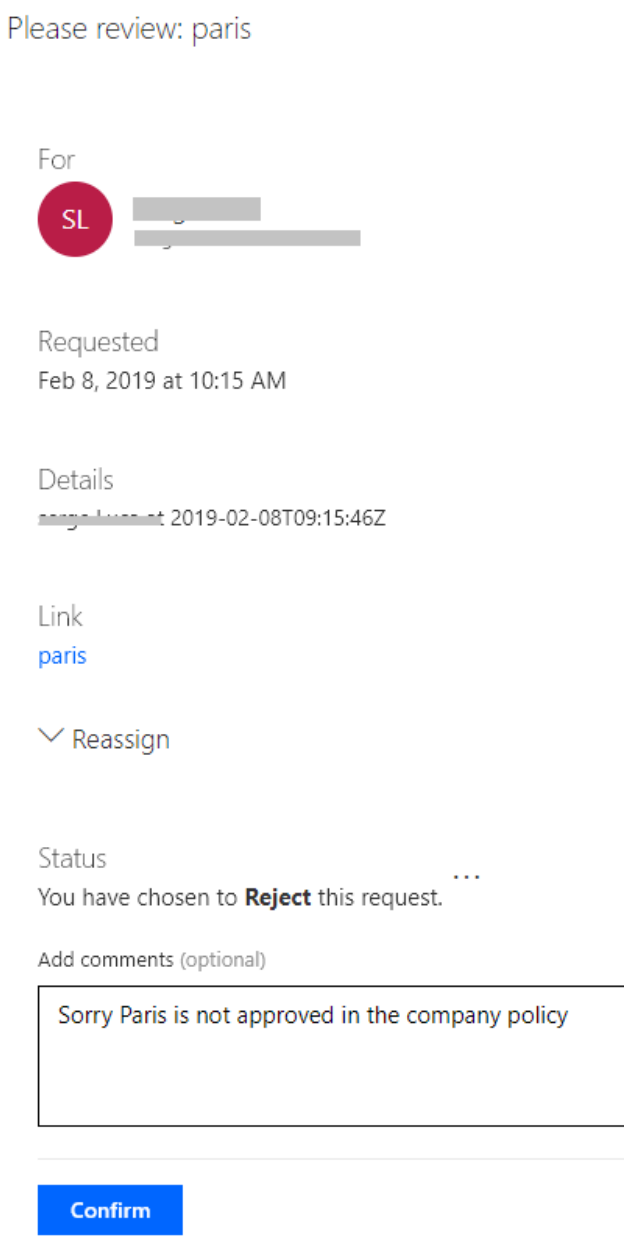


21. Approvers can also find the pending approvals in the **Approvals center**:



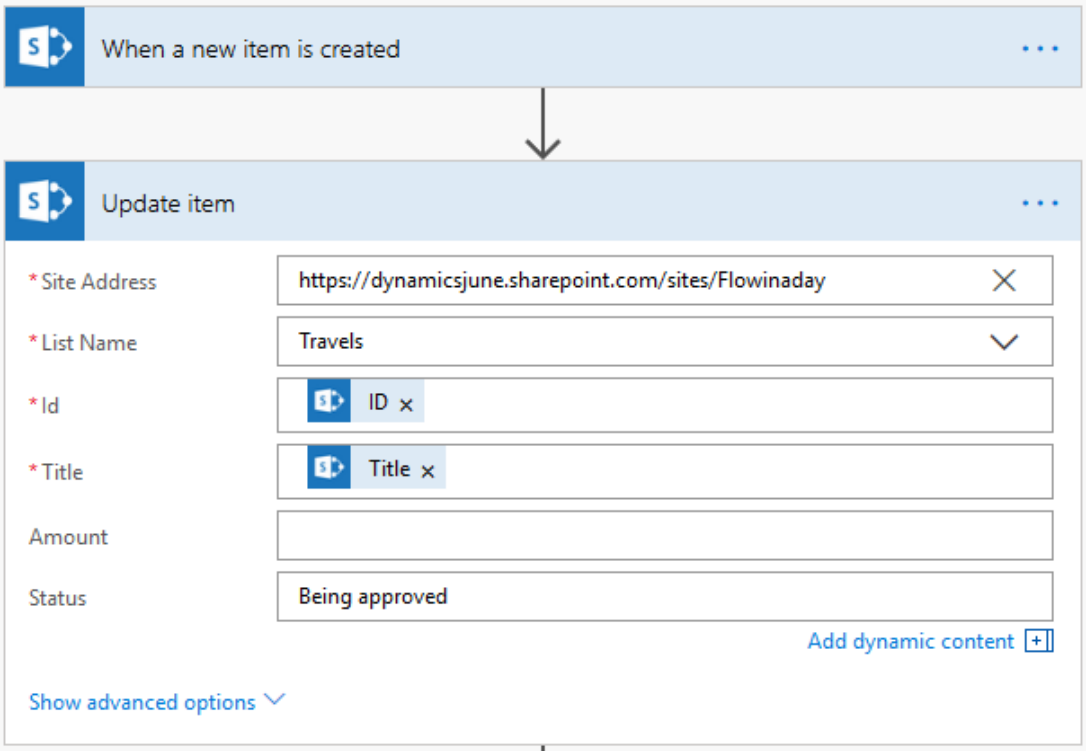


22. Click on **Reject** and a new page where can add additional information will show up and the flow will go on…

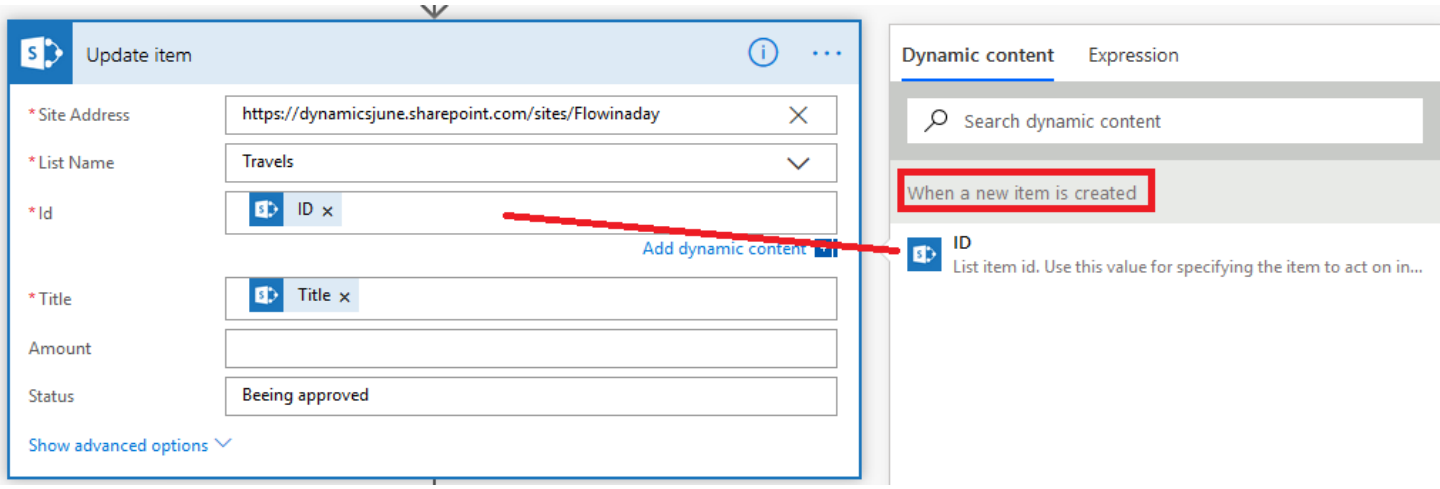


23. The flow will send you a notification by e-mail:

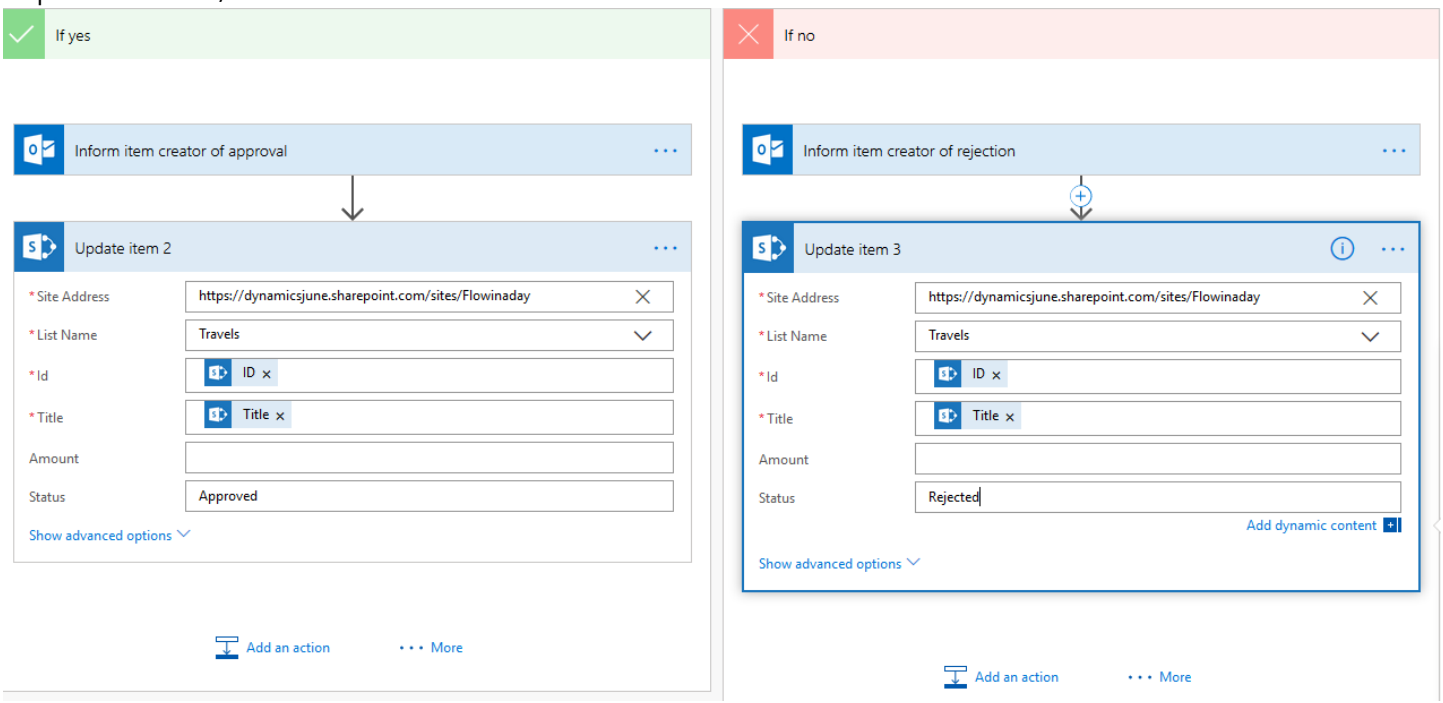
24. We will extend the flow by adding the text **Being approved** when the flow is waiting and **Approved** or **Rejected** when the approval is completed.
25. Click **Edit** in your flow details page
26. After the trigger, add a new **SharePoint Update Item** action and fill it with the following values:



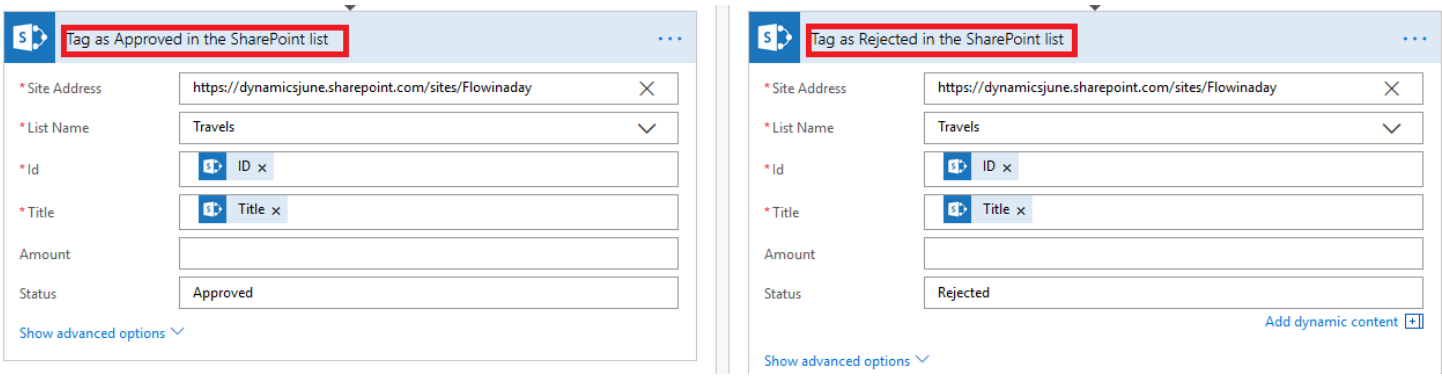
27. The **ID** Dynamic content comes from the trigger and identifies the corresponding SharePoint list item. **Title** also comes from our Trigger and we will provide a value of **Being approved** in the **Status** field:



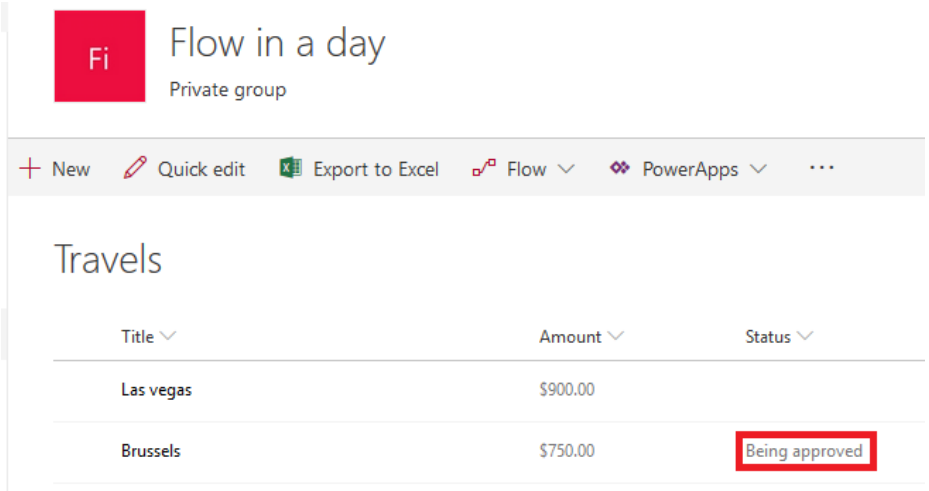
28. Provide similar updates in the **If** yes and **If** no branches, but type **Approved** or **Rejected** in the **Update Item** steps for the If Yes/ If No sides of the condition:



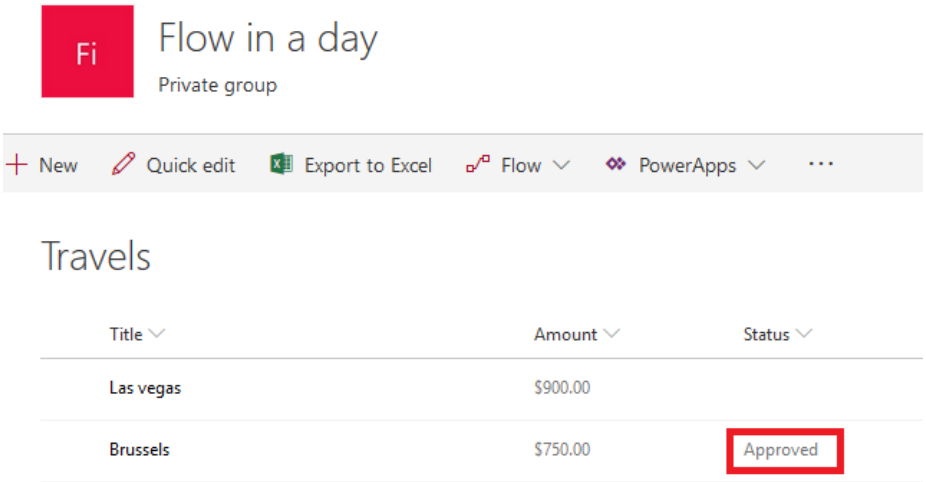
29. As a good practice, rename the actions with a meaningful name:



30. Add a new travel expense in the SharePoint list and check its status in the SharePoint list (it might take a few seconds before the flow starts):



31. Approve and check again the **Status** in the list, once the approval is completed:



Optional exercise if time permits: If amount is smaller than 500 $, the expense will automatically be approved, otherwise it will have to be Approved.

Lab 3. Flow Notifications and Conditions

Learning objectives: Notifications and conditions

Duration: 20 minutes

Scenario: You've set an Out of Office notification in your e-mail box. In the Out of Office message you will mention that if the message is urgent, people can send an e-mail containing just one exclamation point '!' . You also want to get a mobile notification when such message shows-up.

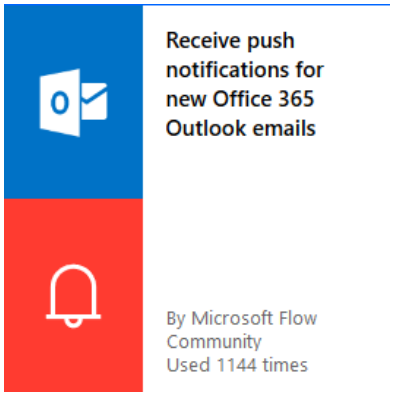
If you want to take a look at the solution: navigate to the **Important e-mails notification template.**

If you want to start the lab from scratch:

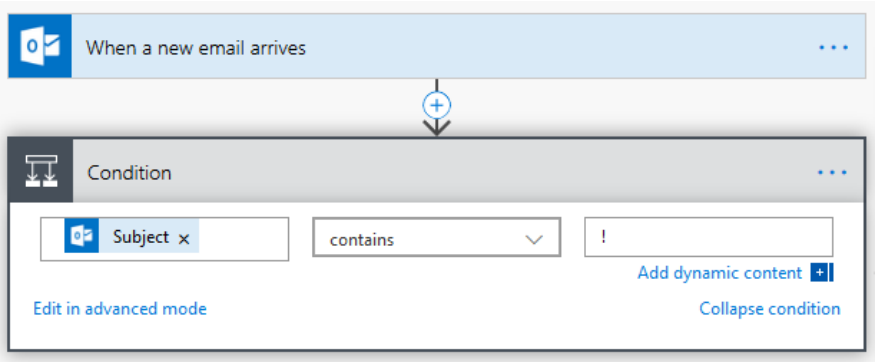
Tasks:

1. In your smartphone, install the Microsoft Flow application and enable notifications for it.
2. Create a new flow from the template **Receive push notifications for new Office 365 Outlook emails.**

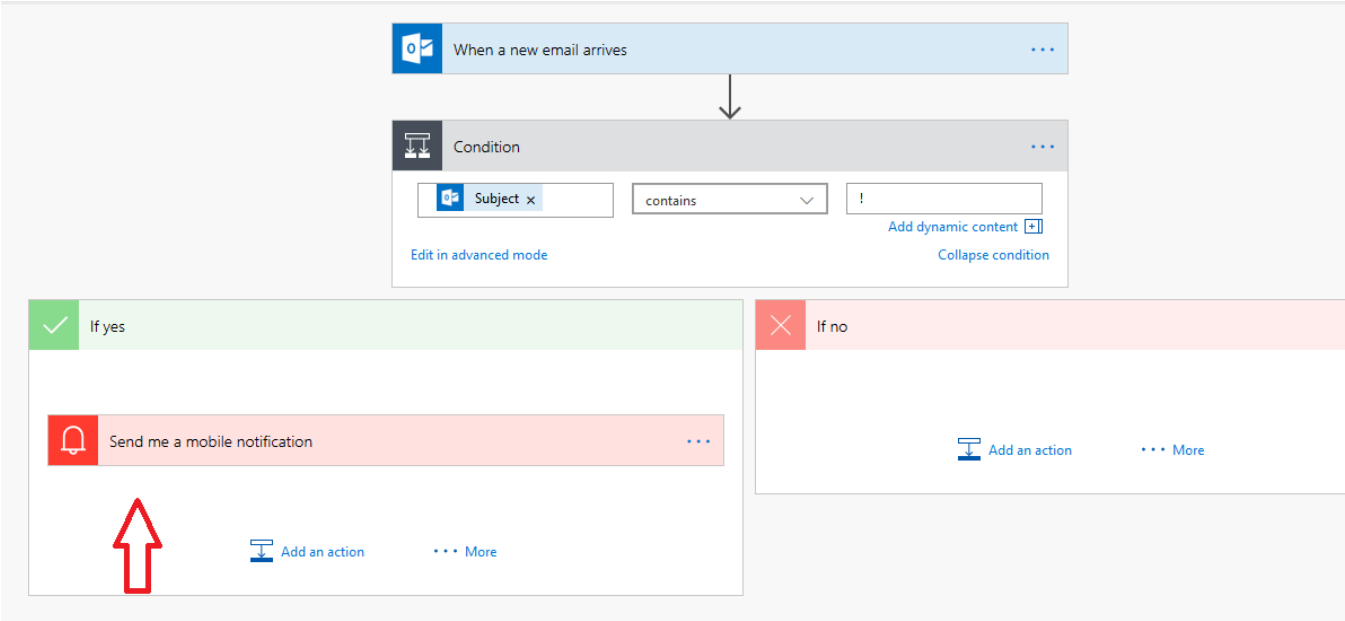
   Note: If you have not created connections for Notifications, you will need to do so in order to provision the template.



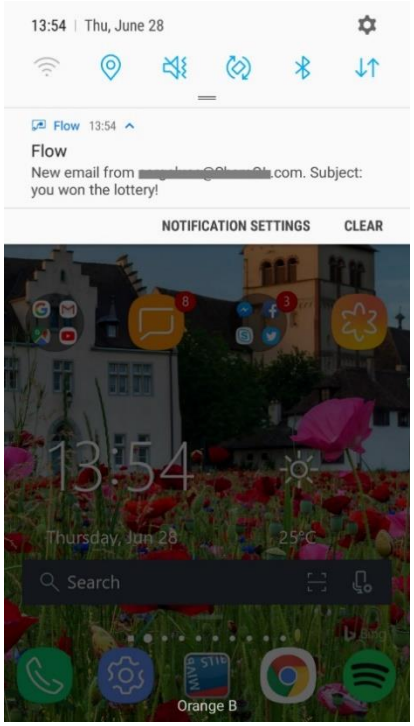
3. Save the flow as **Important e-mails notifications**.
4. Add a condition to check if the mail subject contains **!**.



5. Update the **If yes** condition by moving the notification action into it:



6. Test the flow by sending an e-mail to yourself with the subject **You won the lottery!**
7. You should get a phone notification which looks like this (in Android)
   Note: When using your flow mobile application, you may be prompted to enable notifications.



Lab 4. Flow control, variables, expressions

Learning objectives: Flow control, expressions, variables, using Date/Time

Duration: 50 minutes.

Scenario: We have a list of offices in an Excel sheet. Create a flow that will send a report describing this list of offices, including the biggest office.

If you want to take a look at the solution: Navigate to the **Find the office with the largest capacity** template.

If you want to start the lab from scratch:

Tasks:

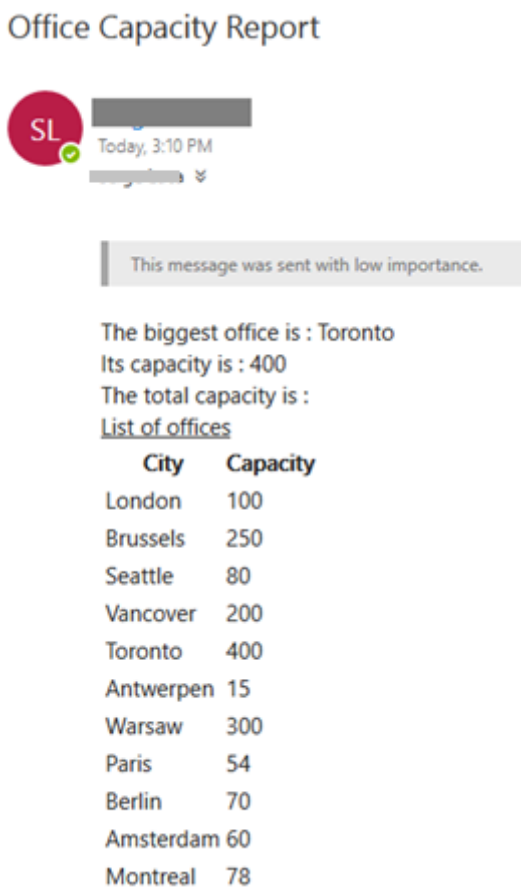
1. In your One Drive for Business, create an **Excel workbook** named **Offices.xlsx**.
2. Add two columns like below with all of the cities and capacities then format as a table with headers:

Contoso Offices

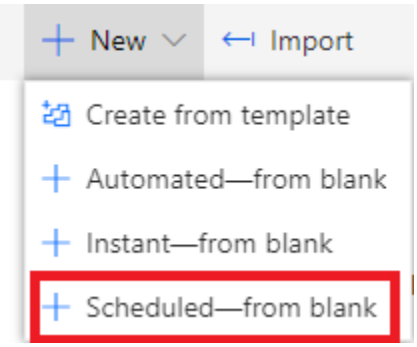
city	Capacity
London	100
Brussels	250
Seattle	80
Vancover	200
Toronto	400
Antwerpen	15
Warsaw	300
Paris	54
Berlin	70
Amsterdam	60
Montreal	78

Amsterdam

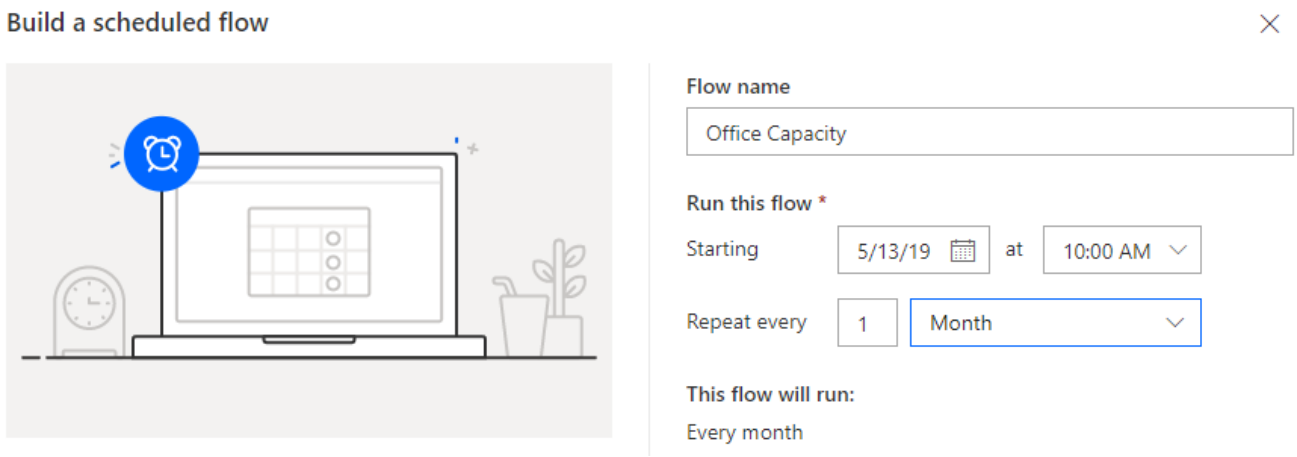
Note: This document contains the list of offices of Contoso Corp. Each office has a limited number of seats. Every month a report describing the list of offices and the total number of seats is sent to the management (in this case the management is…yourself). The e-mail should look like this:



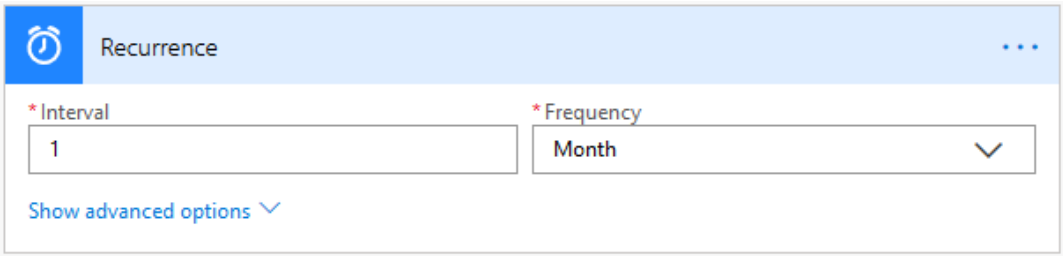
3. To create a flow that will generated this report, go to the [Microsoft Flow portal] and **Create a new flow (Scheduled –from blank)from blank**:



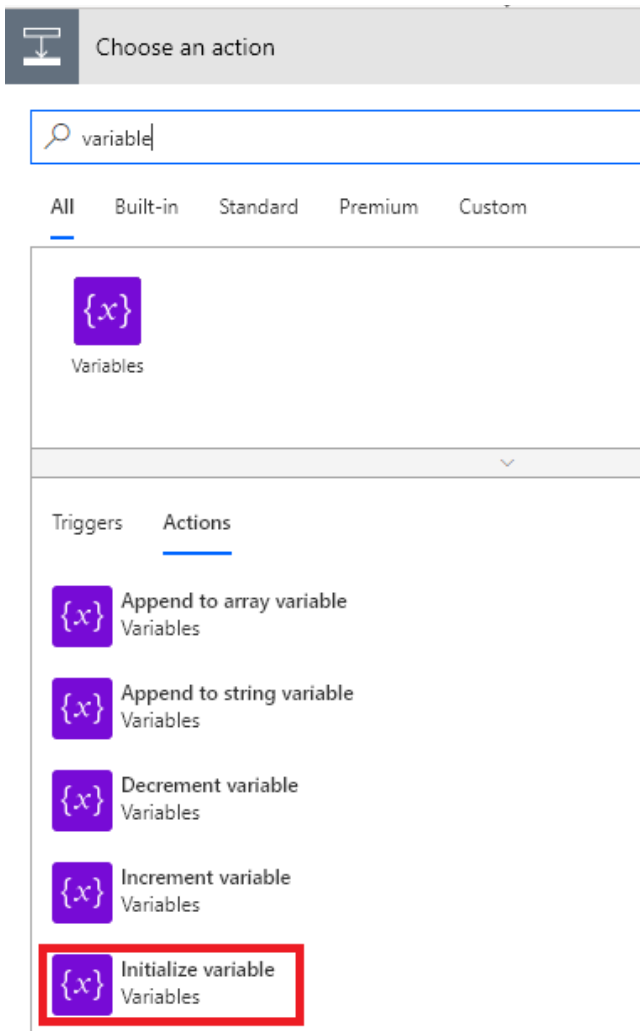
4. The next screen provides the flow name, the flow frequency and click Create:



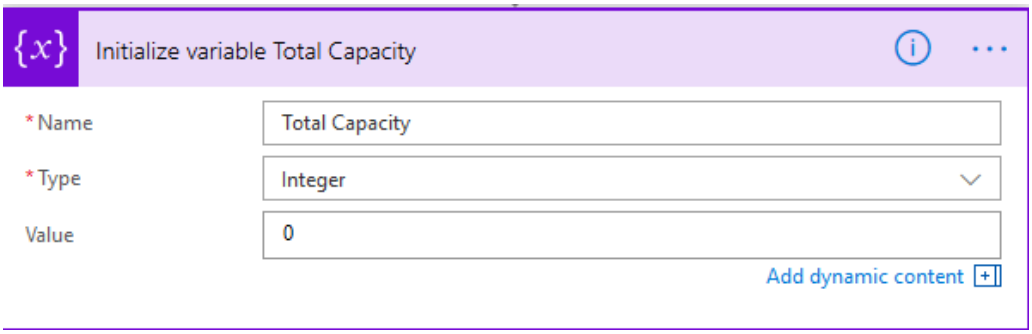
The following flow will be generated:



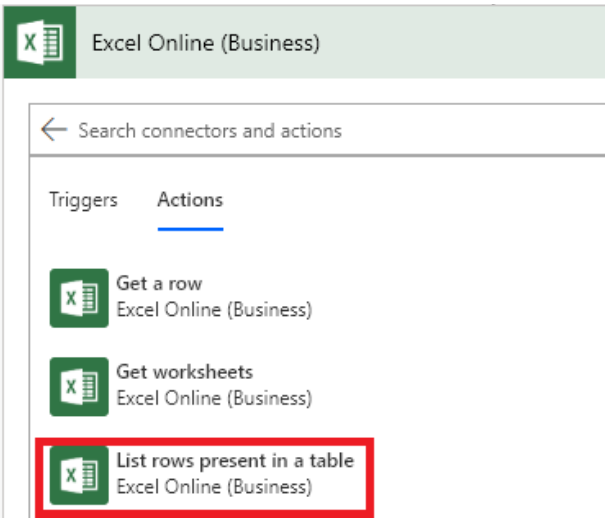
5. The first challenge will be to define the **Total Capacity**. As such, define a variable by adding the **Variable – Initialize variable** action:



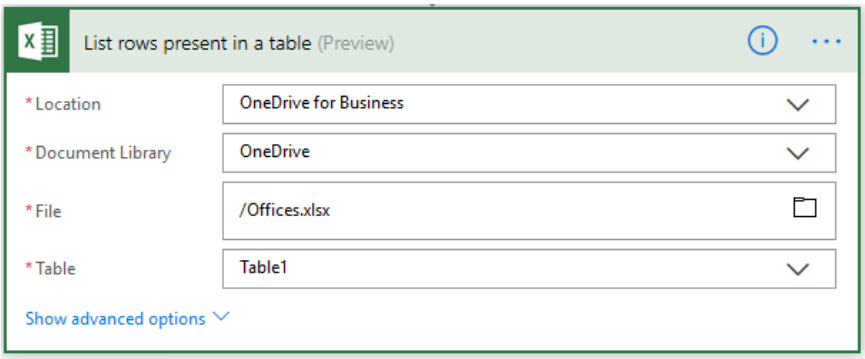
6. Rename this action to be **Initialize variable Total Capacity**, set the variable Name **Total Capacity,** and select **Integer** as the type and an initial **Value** of **0**:



7. Name the flow **Office Capacity** and **Save** it.
8. In the next steps we will update the flow to make it loop through all offices, retrieve their capacity and **increment the variable Global Capacity** to get the total capacity.
9. To retrieve the list of offices, add an action and type **Excel** in the box, then select the action **Excel Online (Business) - List rows present in a table**:

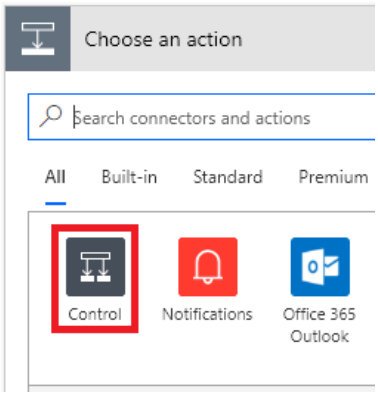


10. Set its properties according to the next picture:

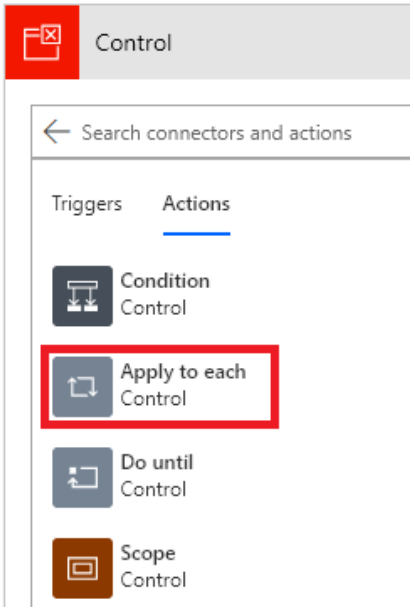


14. Add an **Apply to each** (from the Control connector) after the Excel action:

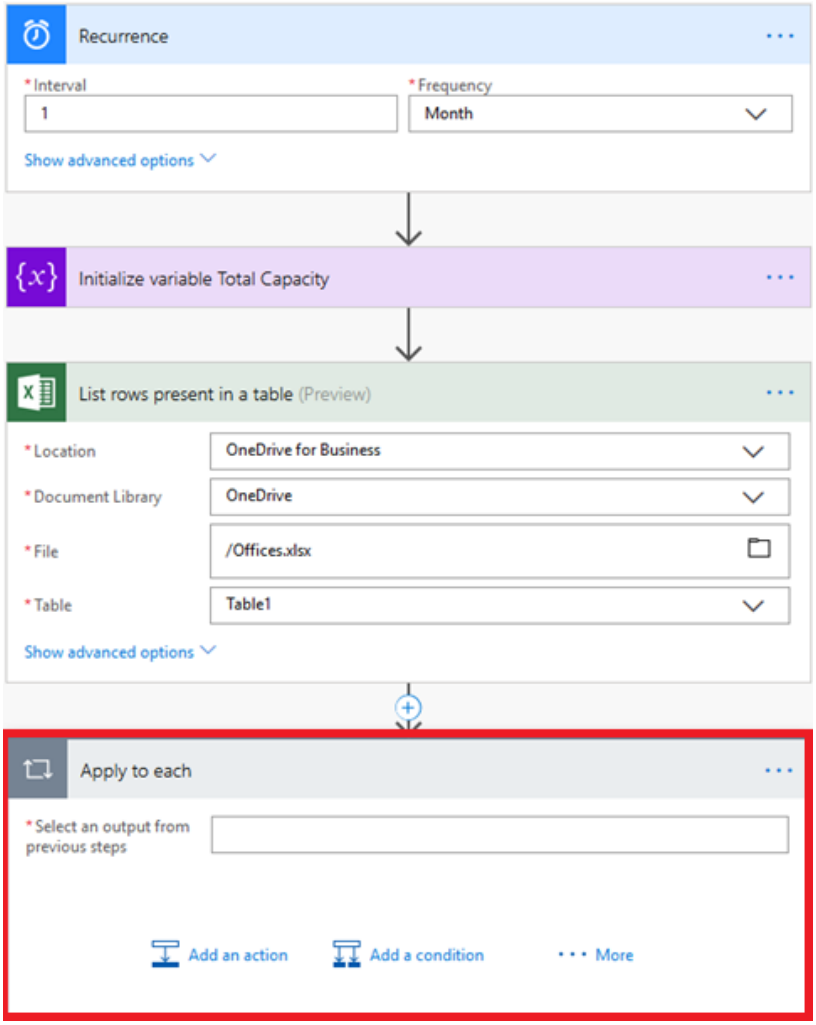
a. Click Control



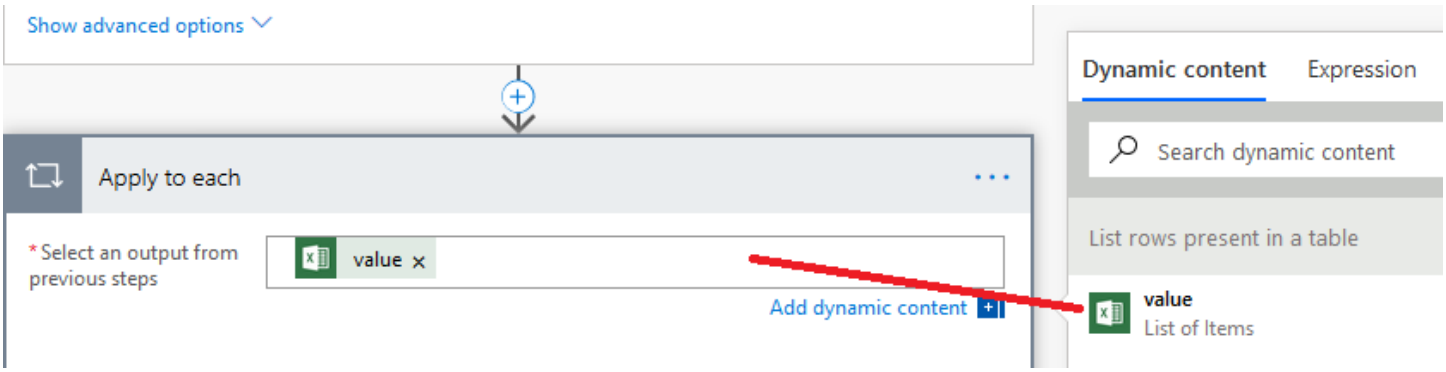
b. Click **Apply to each :**



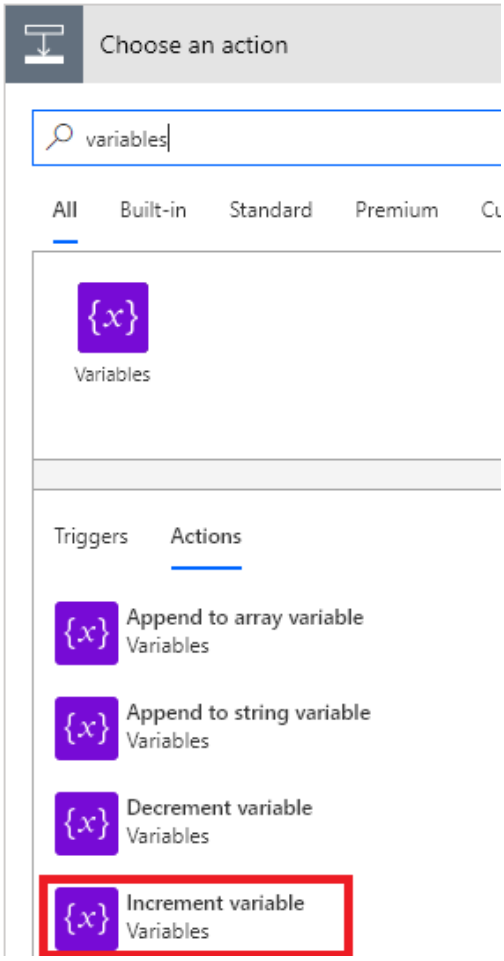
15. Your flow should now look like this:



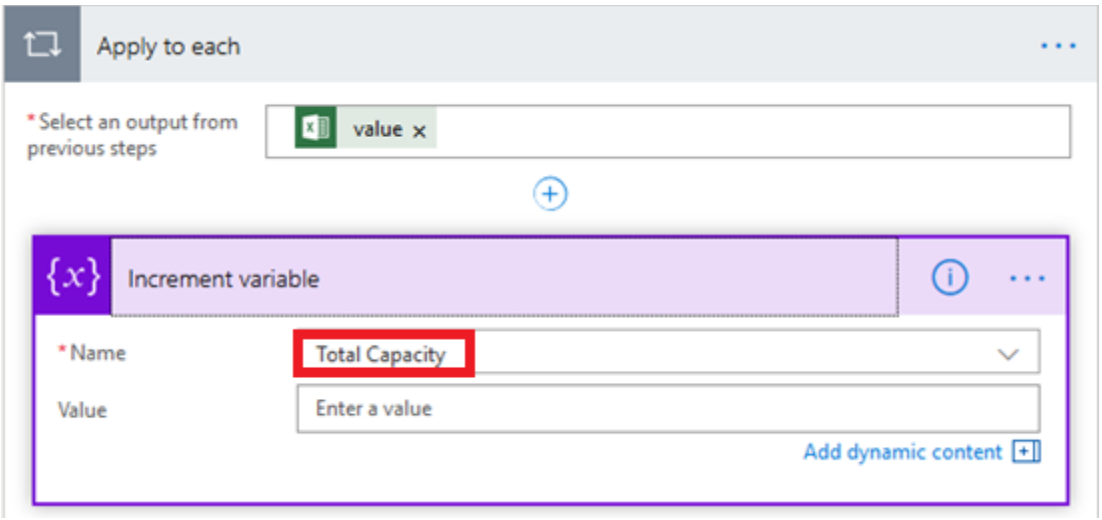
11. The **Apply to each** action expects a list of values, use the Dynamic content **value** of the **List rows present in a table** action:



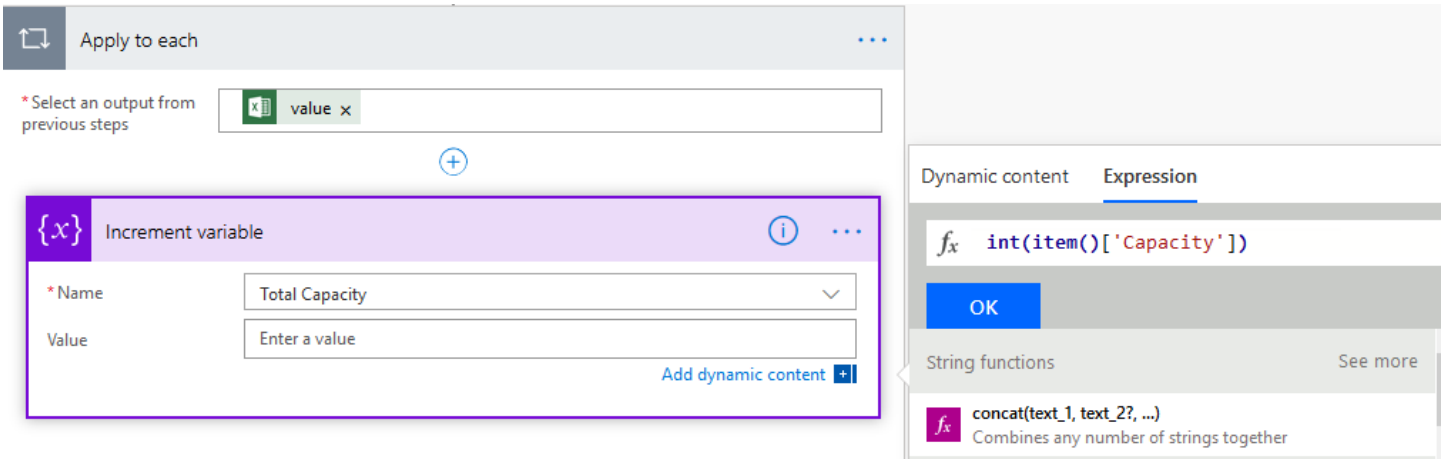
12. In the **Apply to each** action, click **Add an action**. We will compute the total capacity, so add the action **Variables – Increment variable** action:



13. In the name property of the **Increment variable** action, select **Total Capacity:**



14. How can we retrieve the current office Capacity? Currently, we need to use an expression. Click on the **Expression** tab next to Dynamic content and in the fx textbox, type **int(item()['Capacity'])** and **click OK**:

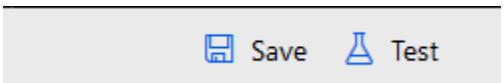


Note:

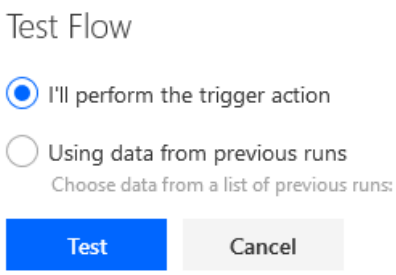
The **item()** expression retrieves the current record information in the current loop and **['Capacity']** provides the field name to retrieve. **Item()['Capacity']** returns a string, to transform a string to a integer (because we need to increment it) we can use the **int()** expression.

There are many other expressions available in flow, we encourage you to read the Flow documentation related to expressions after doing the labs. You can start from here https://flow.microsoft.com/en-us/blog/use-expressions-in-actions/.

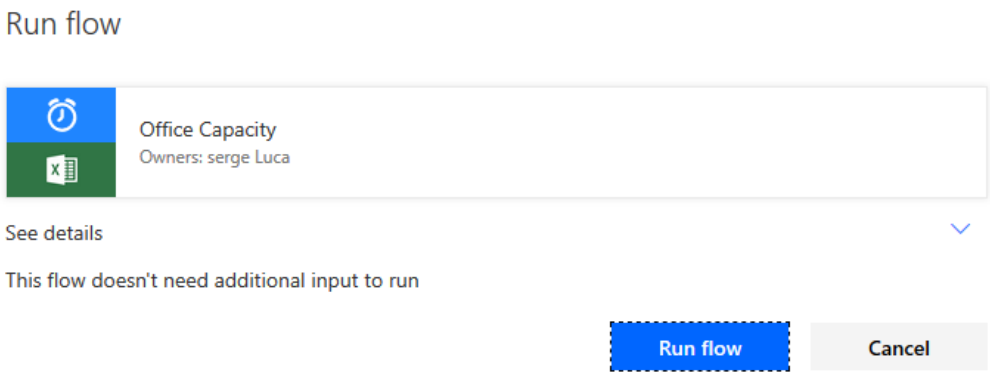
15. Let's test our flow, but we don't want to wait one month before the flow starts. Instead, we are going to use the **Test** button, we can manually start the flow on demand (in test mode). This is convenient for testing and debugging purposes. Test your flow, by clicking the **Test** button.



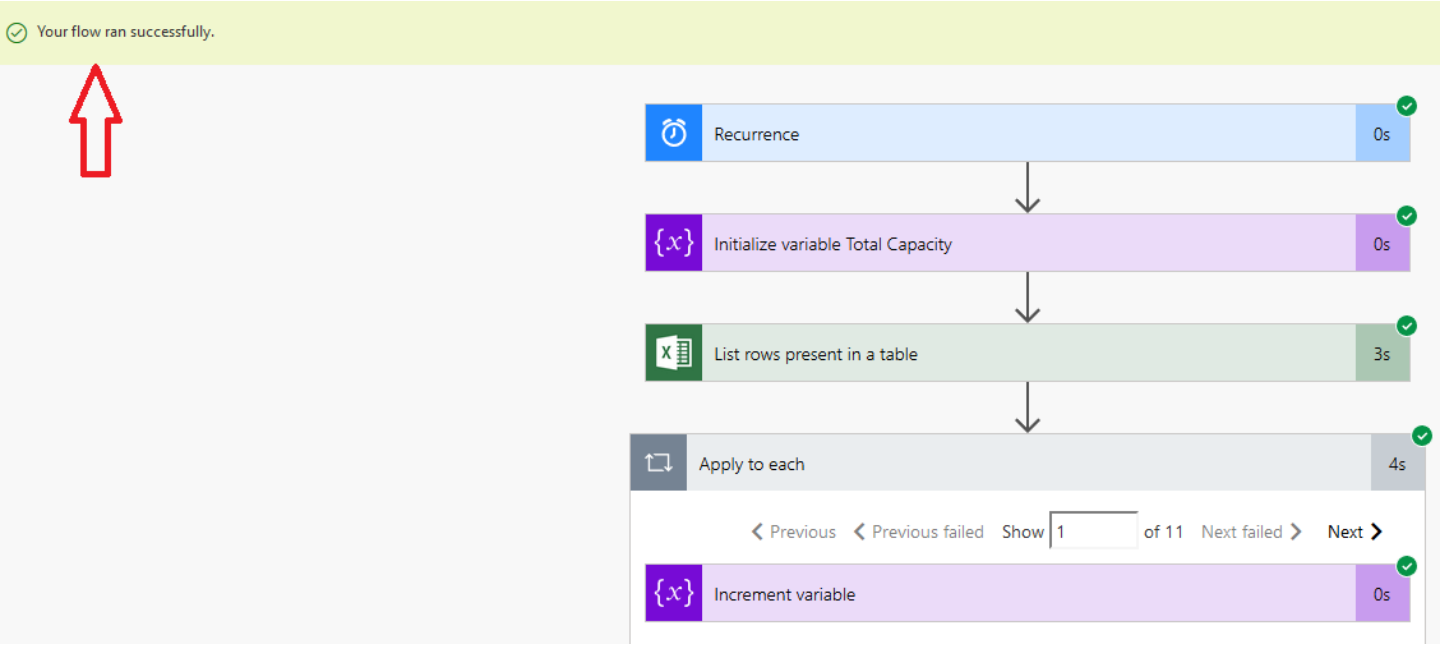
16. Select **I'll perform the trigger action**:



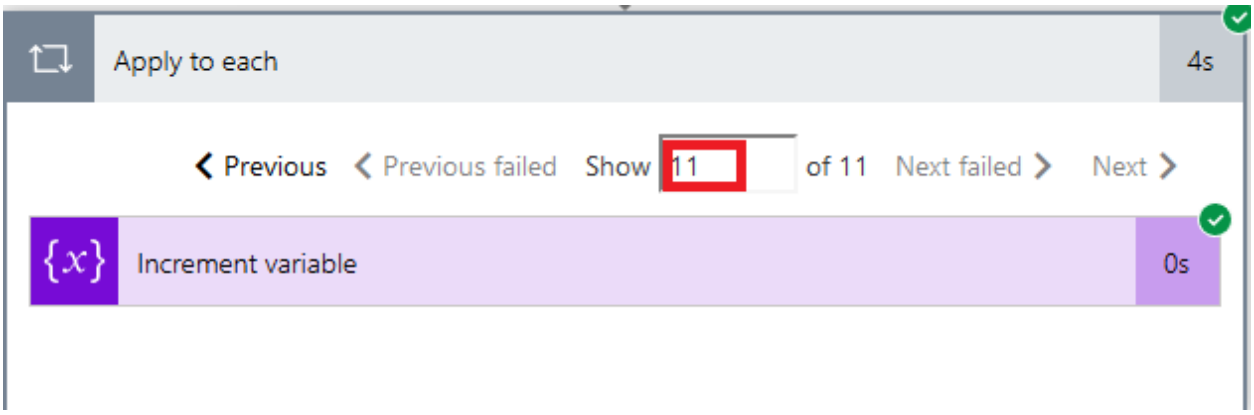
17. Click on **Run flow**:



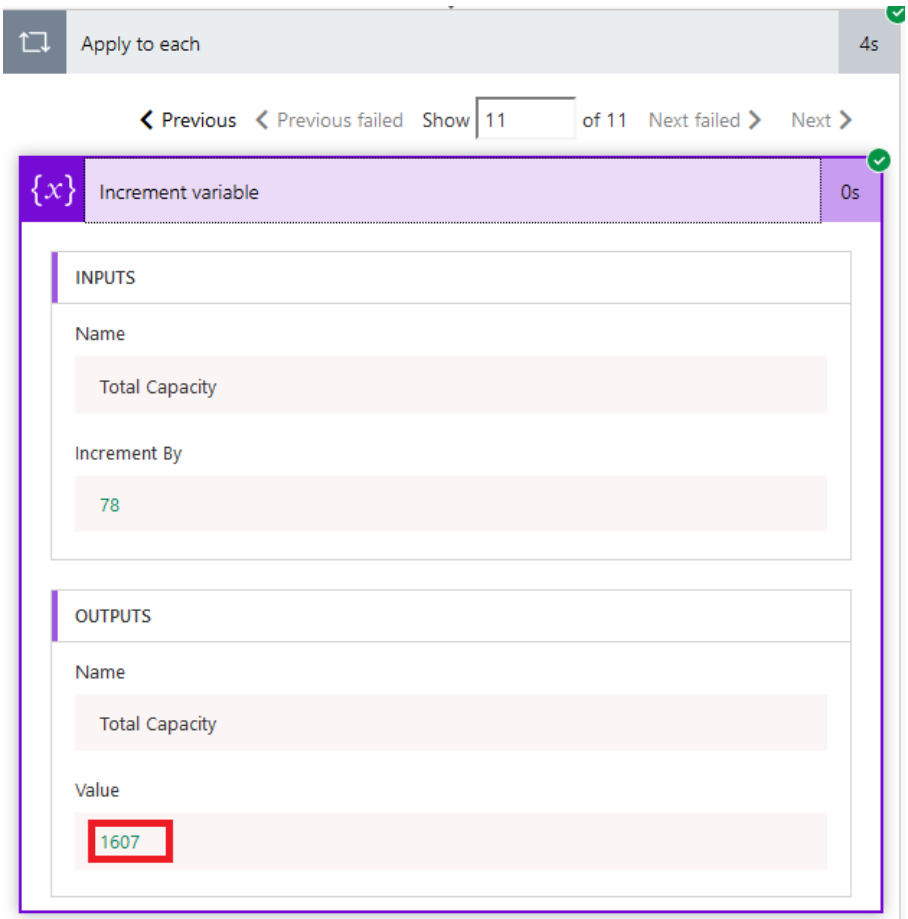
18. Wait until you get the message: **Your flow ran successfully.**



19. How can we check the **Total Capacity** value? We can examine the value of **Total Capacity** for each step. For example, in our case, we will check its value in step 11 (last step): so type **11** In the **Show** textbox:

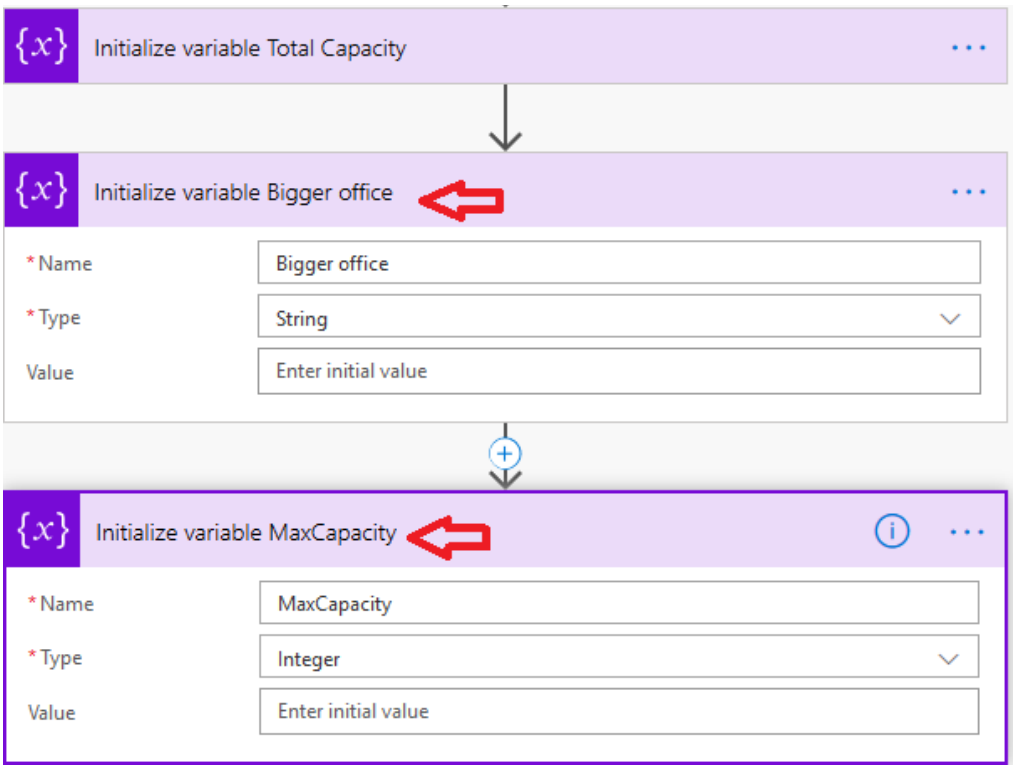


20. Click on **Increment variable** action. Once clicked, you should get a value of **1607** if you use the values defined in the beginning of the lab:



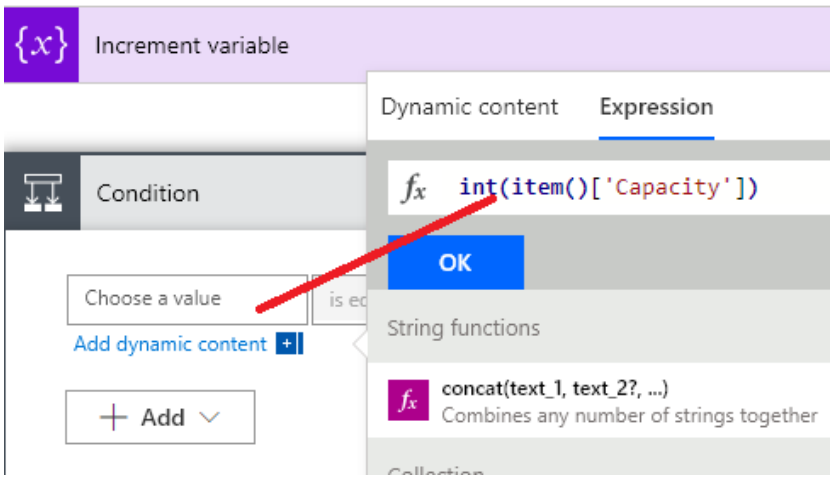
21. Let's define 2 new variables underneath total capacity:

- **Bigger Office** (type string)
- **MaxCapacity** (type integer)

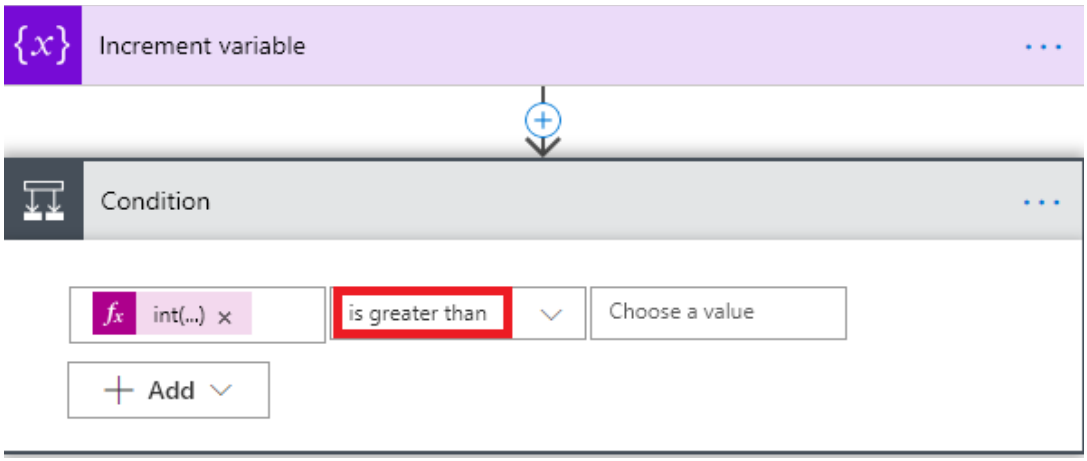


22. **Add a Condition** (from) the Control connector) in the **Apply to each** action:

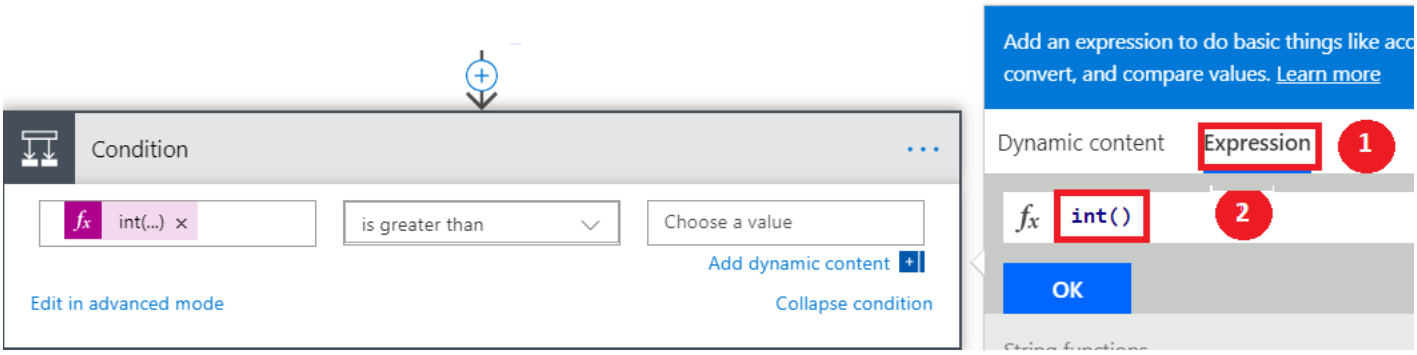
23. The goal is to compare 2 numbers and to select the larger one. In order to do so, we need to transform our capacity values into integers. In the left side of the condition, click **Choose a value** and click on **Expression**. As we already did it before, type **int(item()['Capacity'])** as illustrated below:



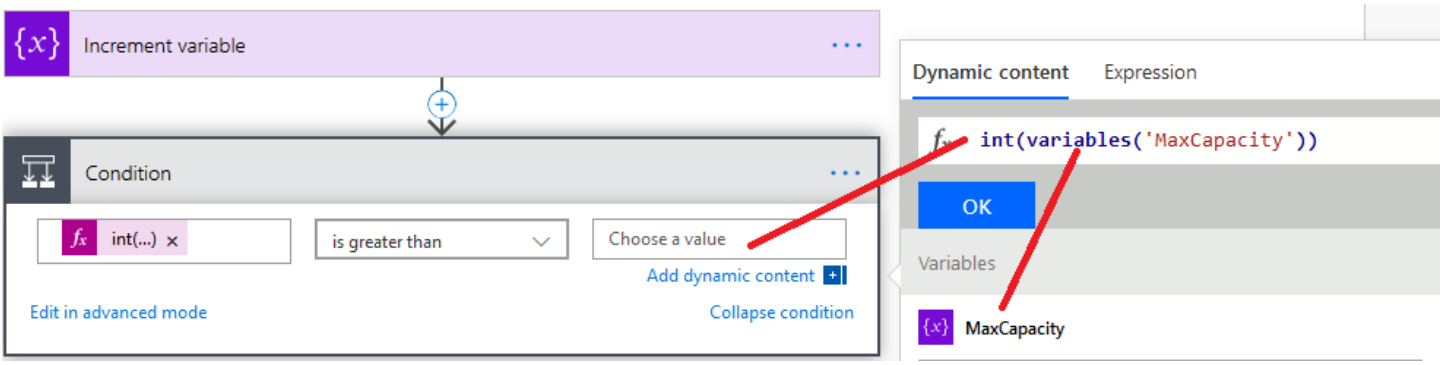
24. Click **Ok.**
25. Select the comparison operator **is greater than**:



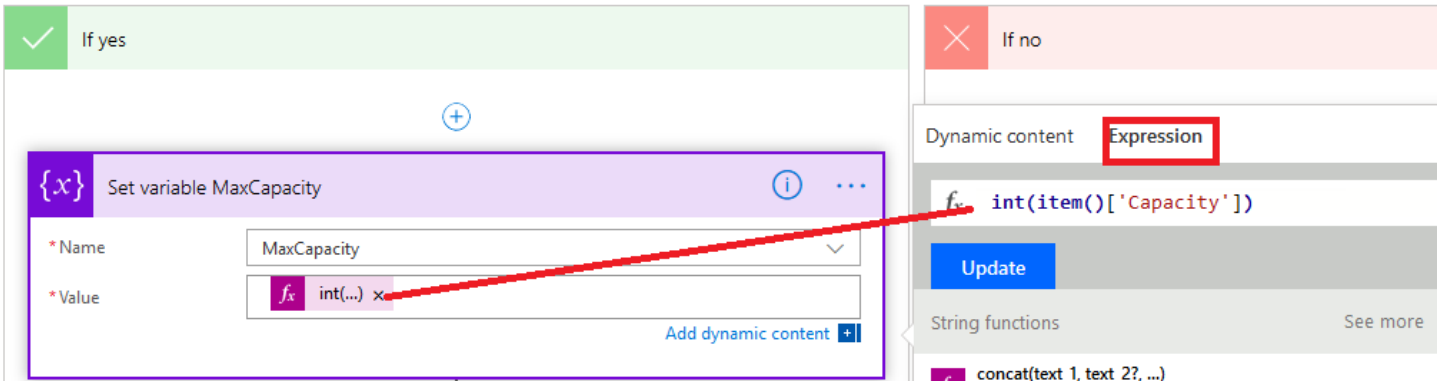
26. In the **Choose a value** textbox, we will include an expression much like we did before by using the **int()** expression, but with a small variation. Click choose a value, click on **Expression** and type **int()**



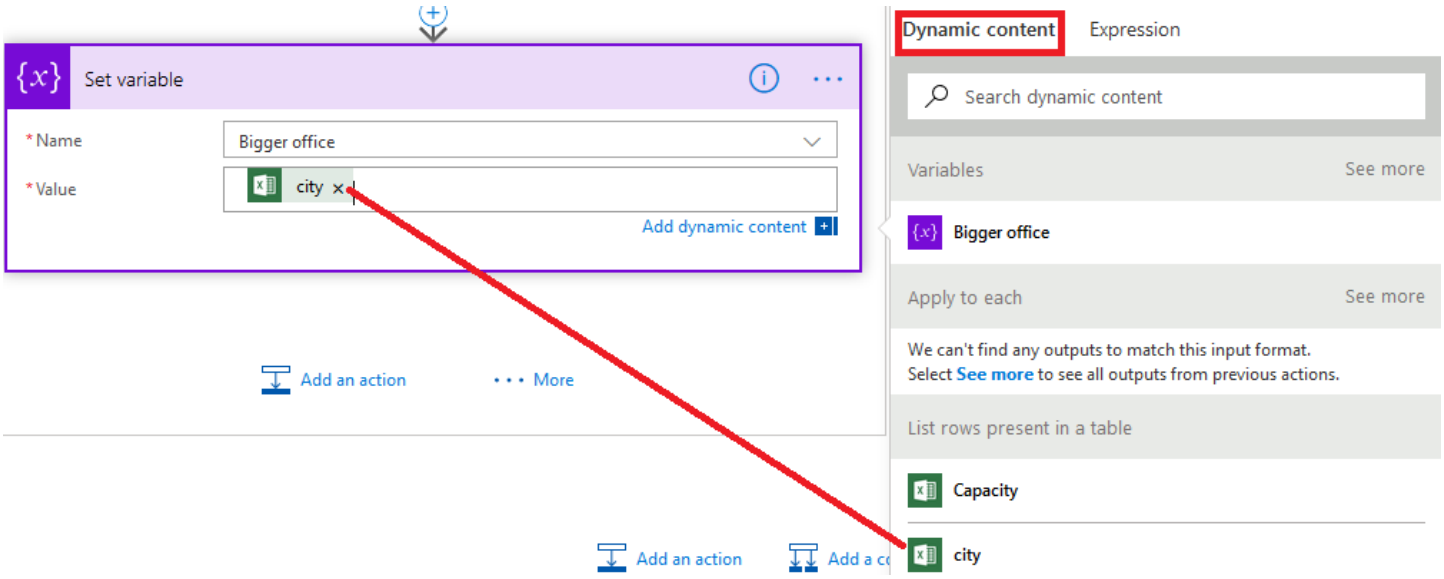
27. Move the cursor within the **int()** parentheses.
28. Click **Dynamic content**, select the **MaxCapacity** variable: the editor will automatically generate the expression. Click **Ok** and **Save** the flow.



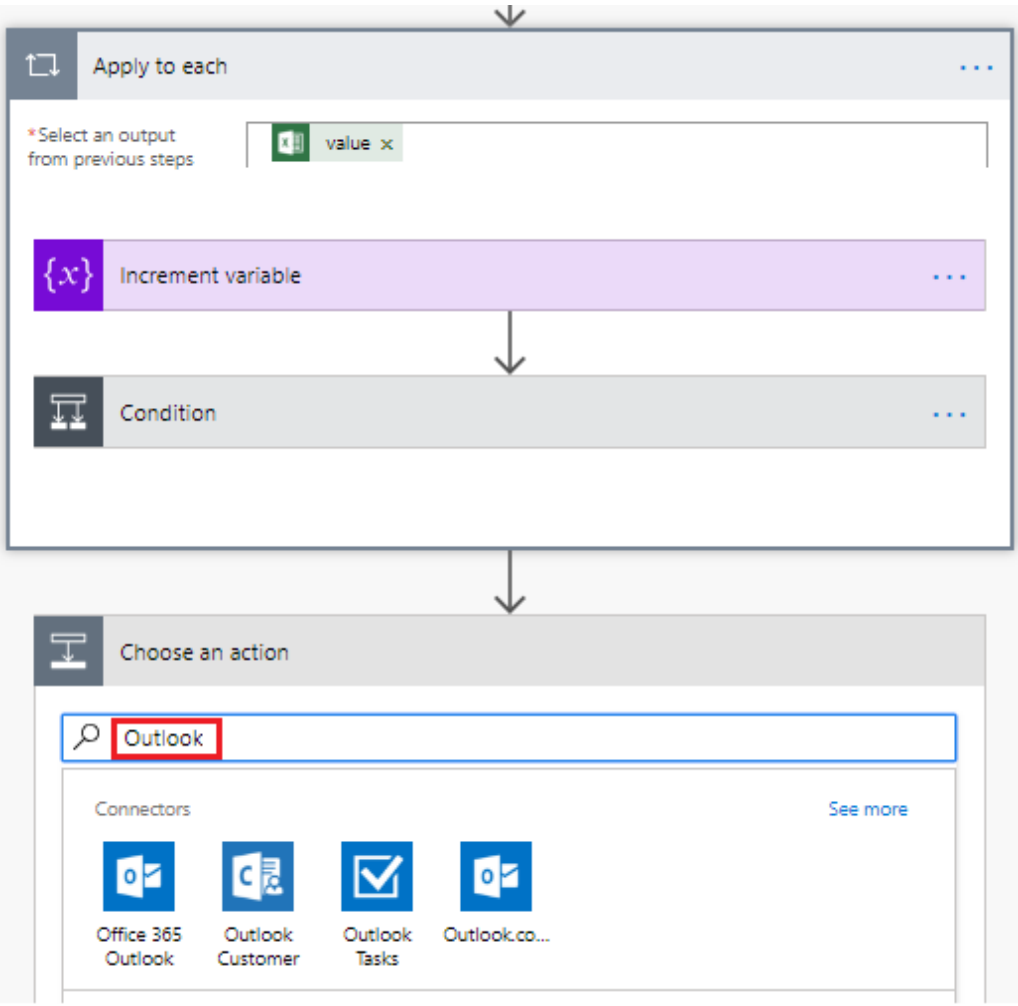
29. Now, in the left **If yes** branch, add a new action **Variables – Set variable** for our **MaxCapacity** variable and in the **Expression** panel type
int(item()['Capacity']) as illustrated in the next picture.



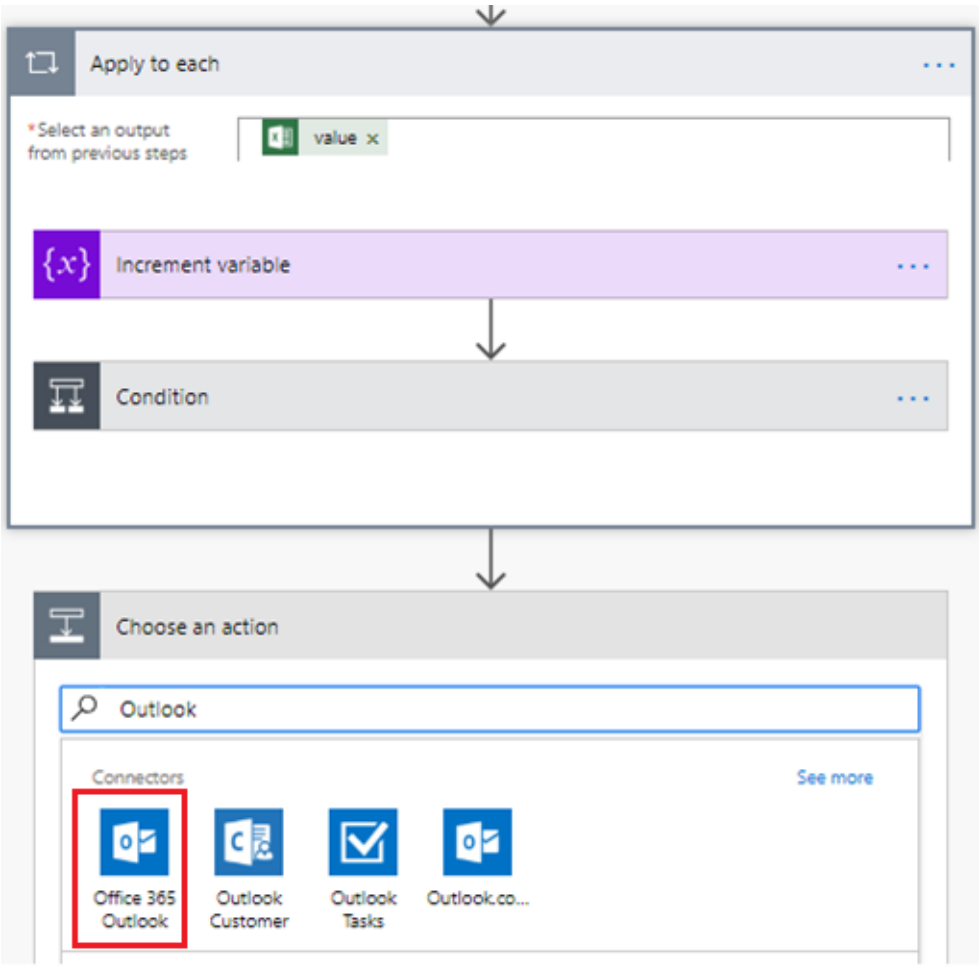
30. Rename the action **Set variable MaxCapacity.**
31. In the same left branch of the condition add **another set variable action** and select the variable **Bigger office** and assign it a value of **city**.



32. Save and test the flow to figure out which city has the bigger capacity (Toronto in our case). You can debug the flow or add a notification (or send an e-mail to yourself).
33. Next, let's send an e-mail by adding an **Outlook 365 Outlook - Send an e mail** action **after the Apply to each:**
    a. Find the action by typing Outlook:



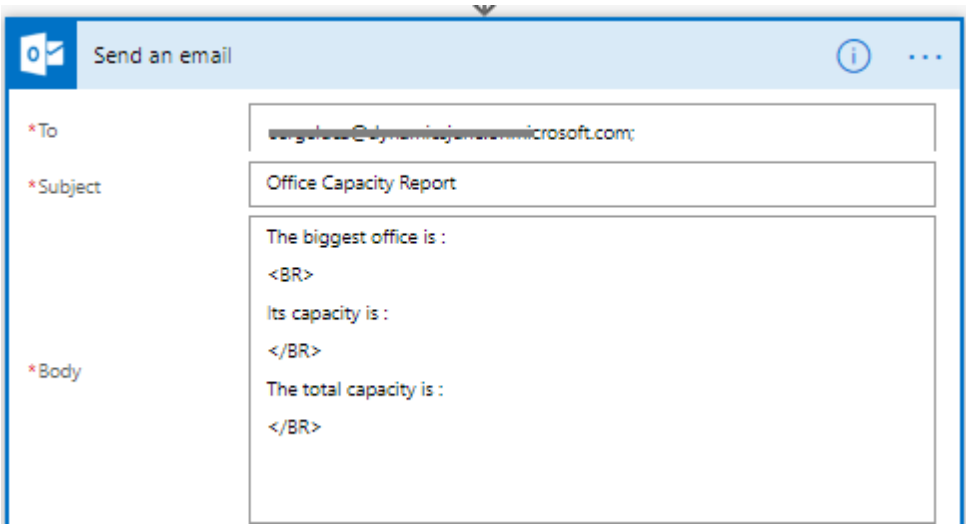
b. In the Connectors list click **Office 365 Outlook**:



c. Select the action **Office 365 Outlook – Send an email**:

d. Fill-in the Send an email action with the following values
   i. In the To field provide your e-mail address
   ii. In the Subject, type "Office Capacity Report"

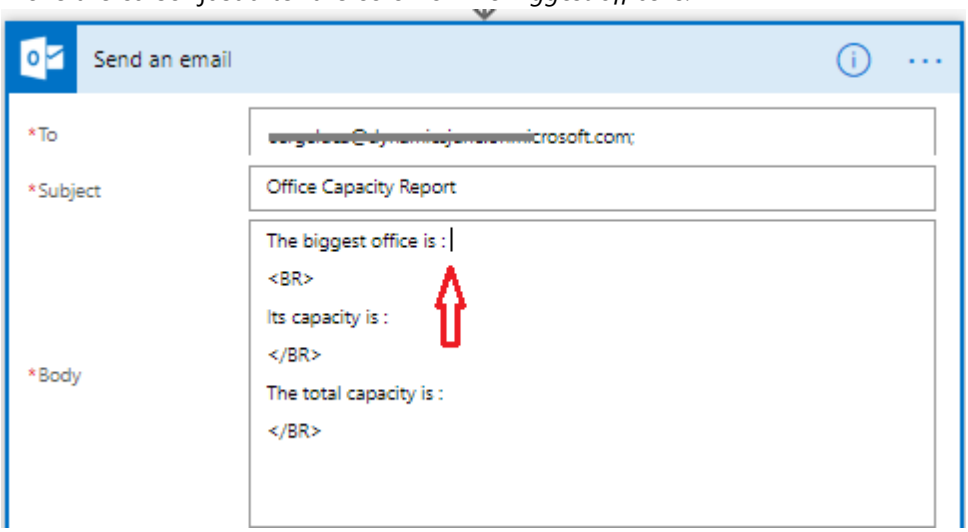
iii. In the Body type the following text:



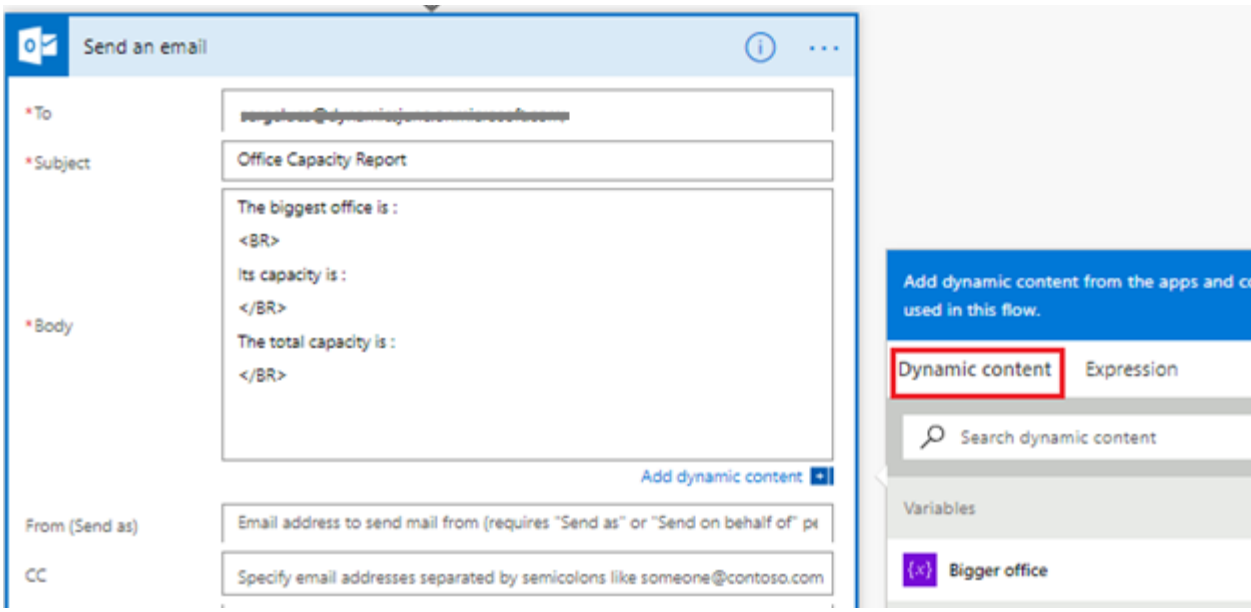
You will notice that we are using HTML <BR> tags directly in the body to generate line breaks in the mail message.

iv. We will now add our variables value directly in the Body

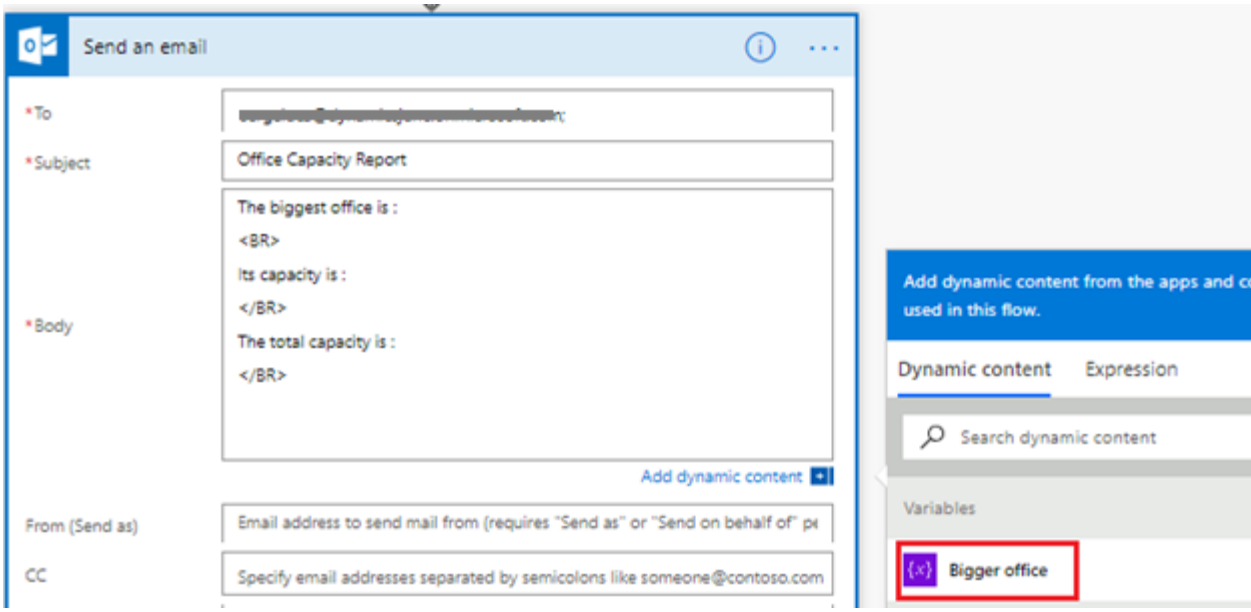
v. Move the cursor just after the colon of *The Biggest office is:*



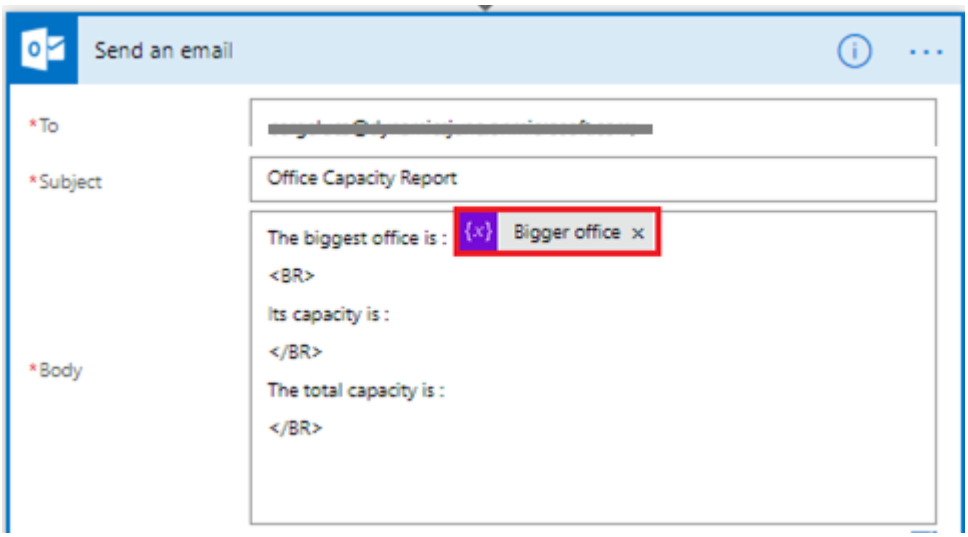
vi. The Dynamic content panel should show up:



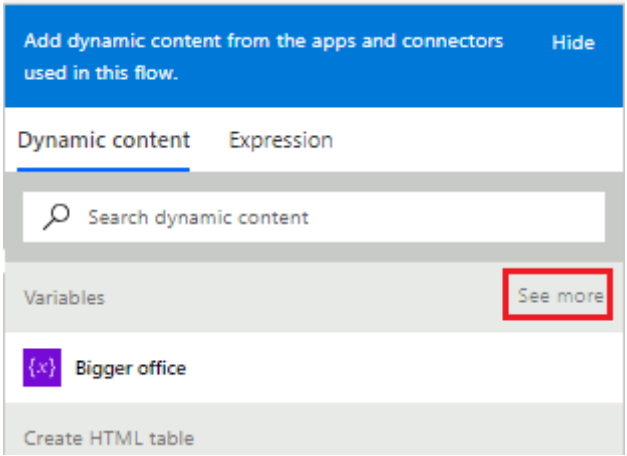
vii. In the Variables section select "Bigger office":



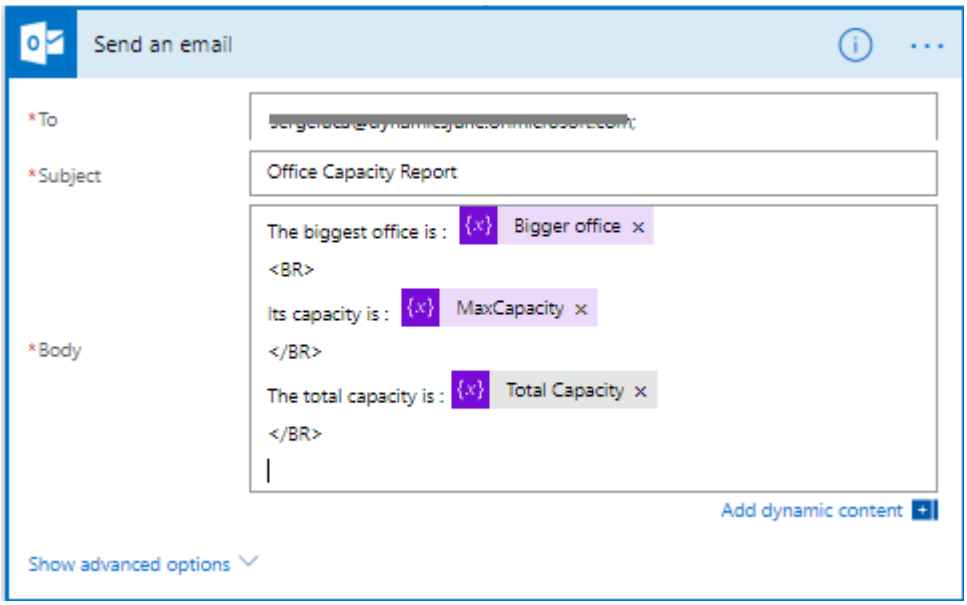
viii. The variable name **Bigger office** should now be visible in the body:



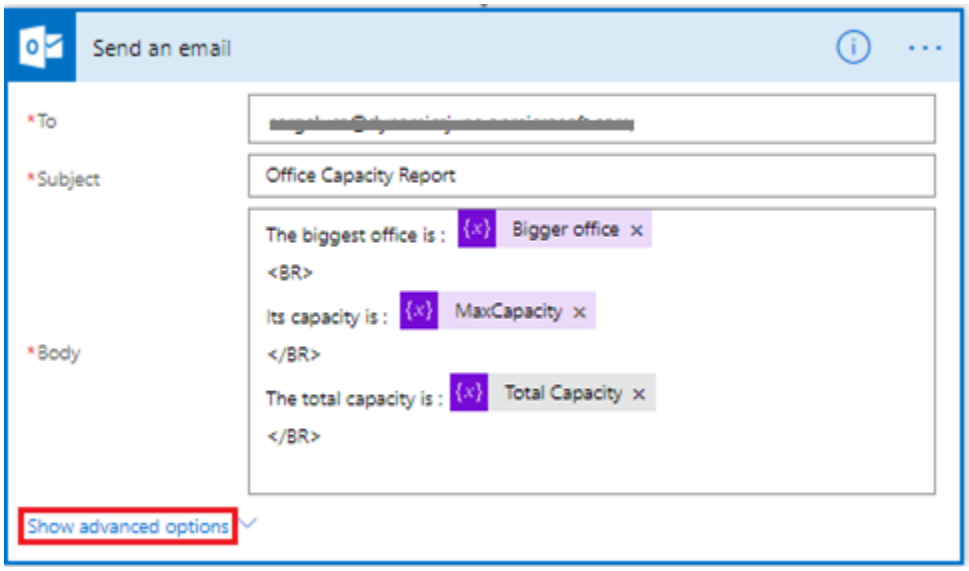
ix. Proceed the same way with the other variables **MaxCapacity** and **Total Capacity**; if you don't see these variables click **See more** in the variables section:



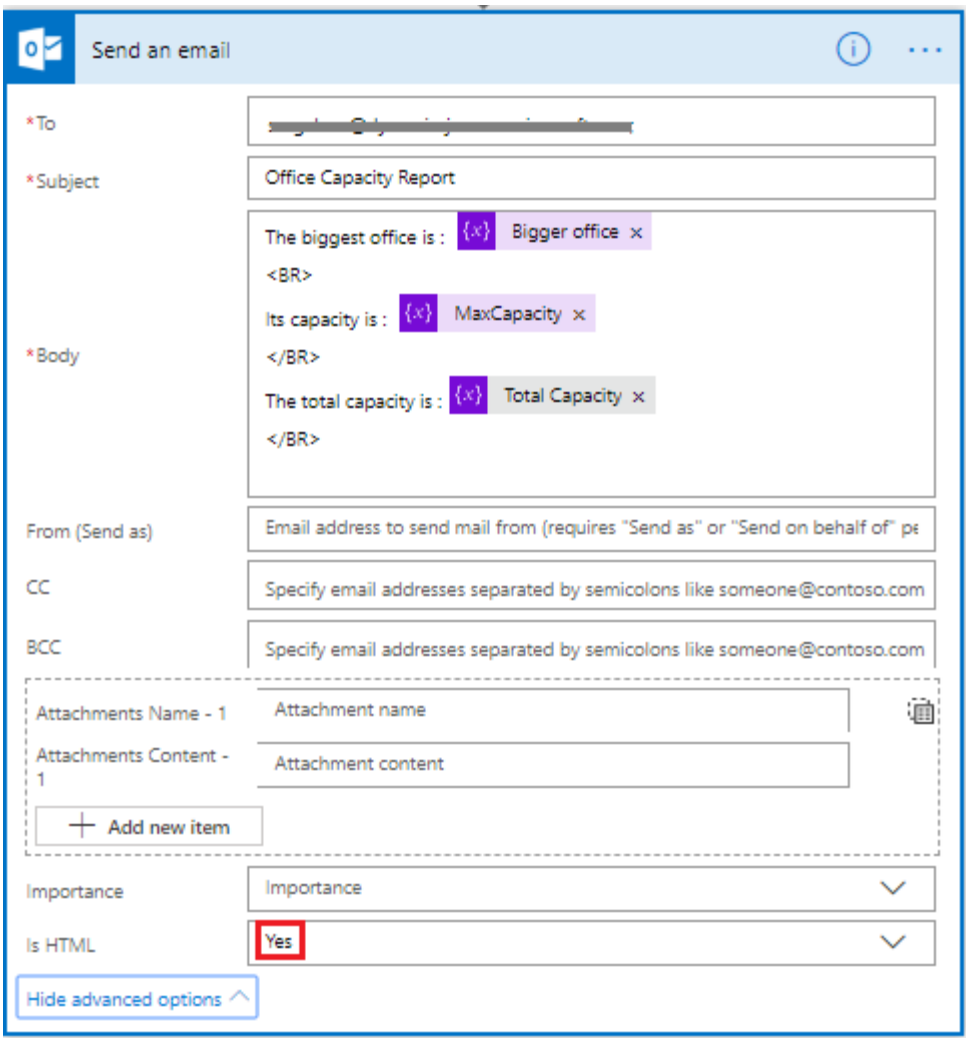
x. Eventually the e-mail body should look like this:



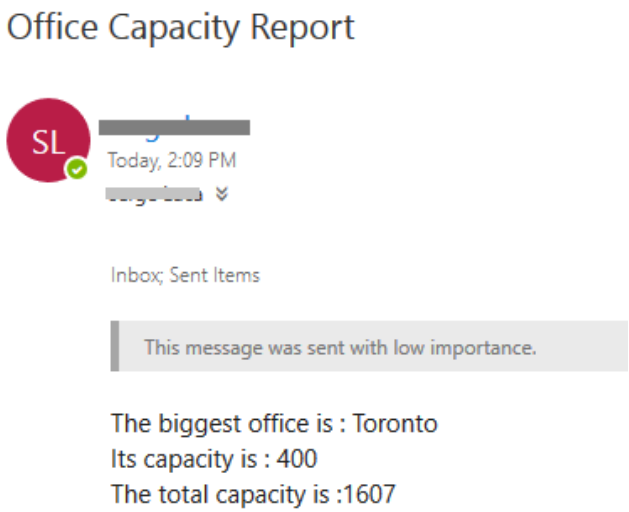
xi. Turn HTML on in the e-mail body by clicking on the **Show advanced options** link:



xii. Switch **is HTML** to **Yes**:

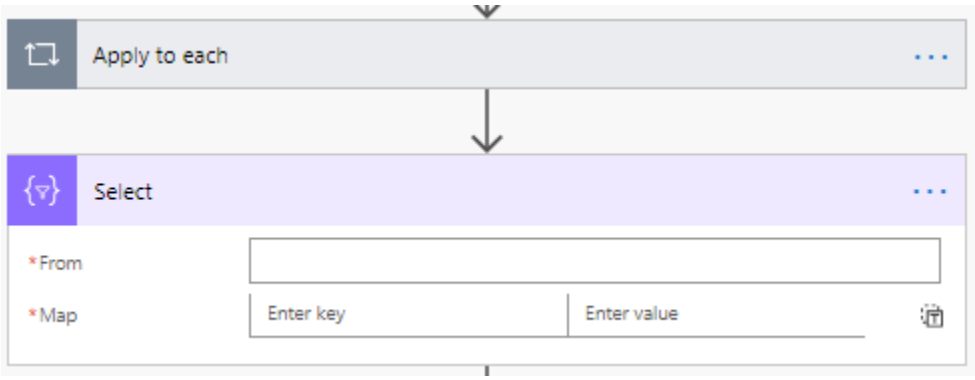


34. Save your flow and test it.
35. Check your e-mail, you should receive something like this:

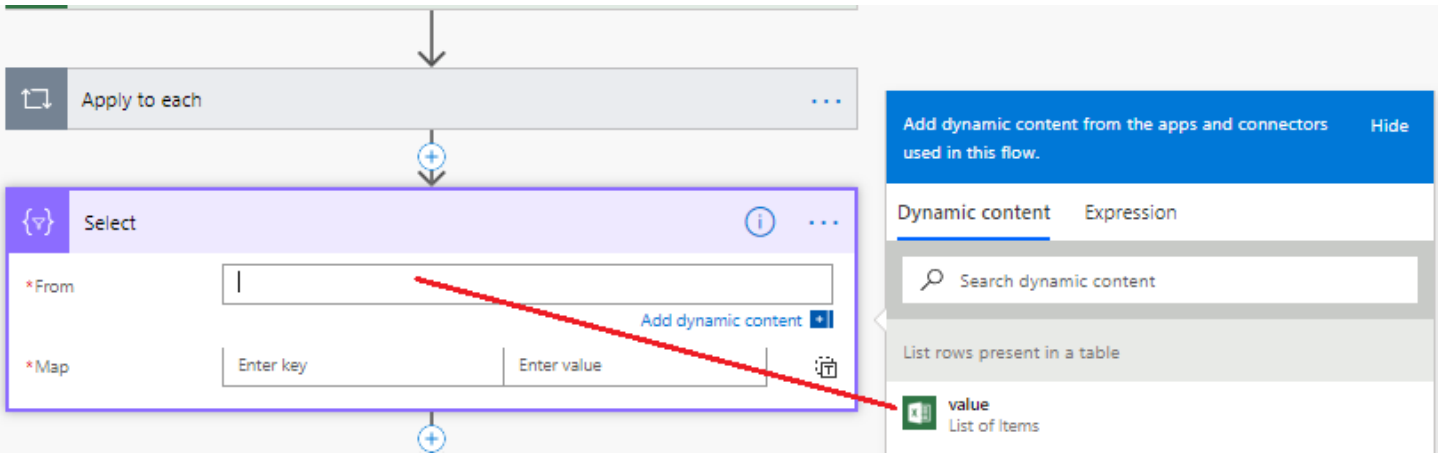


36. In the next steps we will display the list of offices, so we will have to define a list formatting logic and create an HTML table based on this logic.

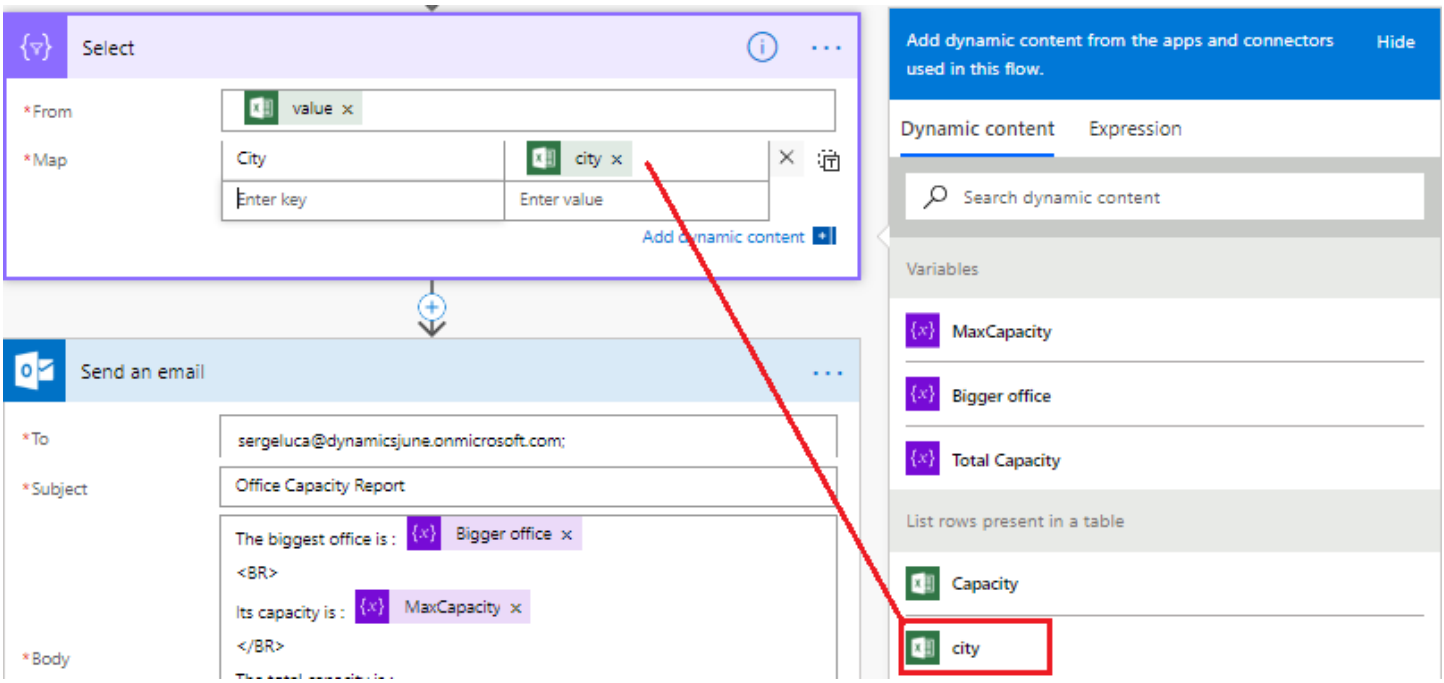
37. Let's define the list formatting logic. Before the **Send an email action** add a **Data Operations – Select** action:



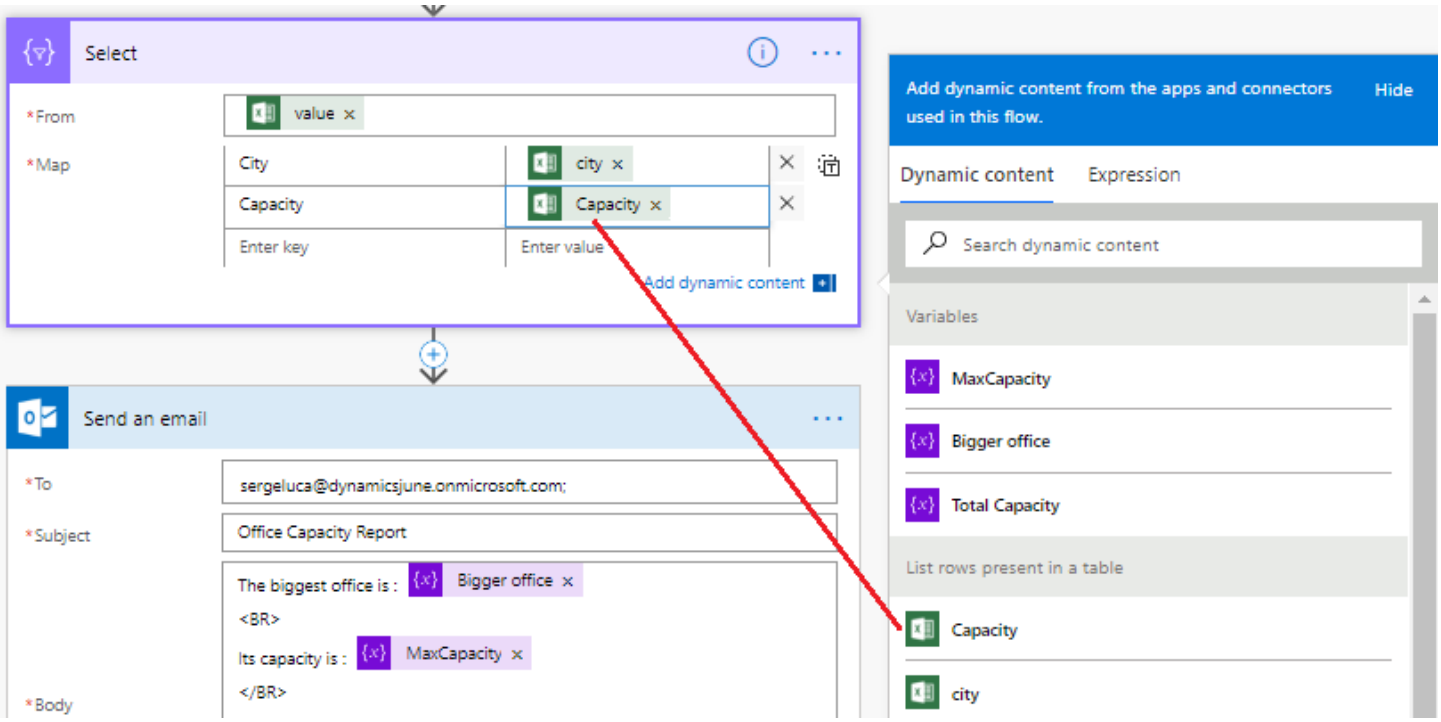
38. Move the cursor in the From field and in the **value** Dynamic content associated with **the List rows present in a table** action:



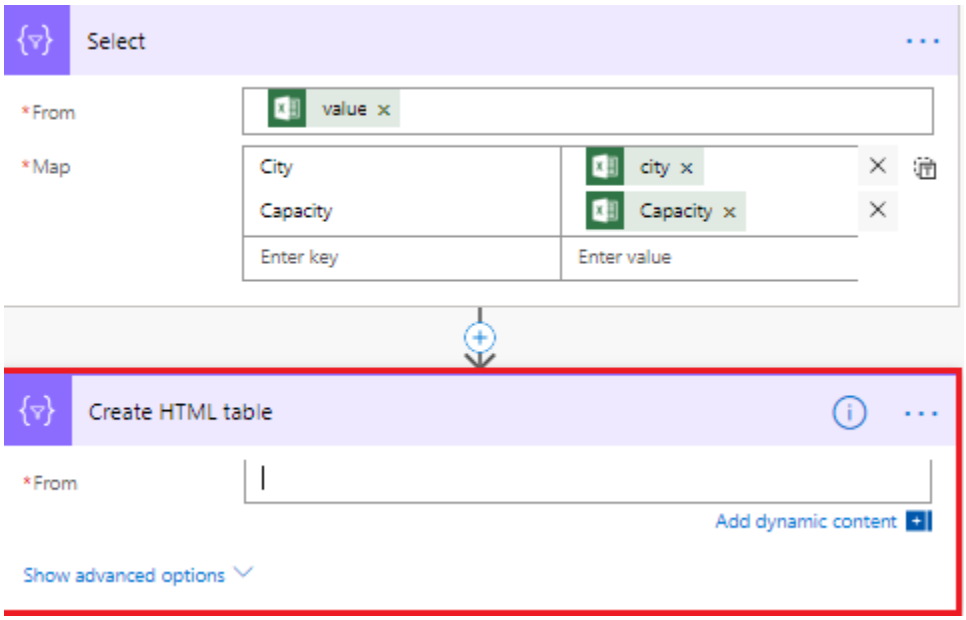
39. In the map field, add the following values: the key field should be **City** and the value should be the **city** value in the **Dynamic content**:



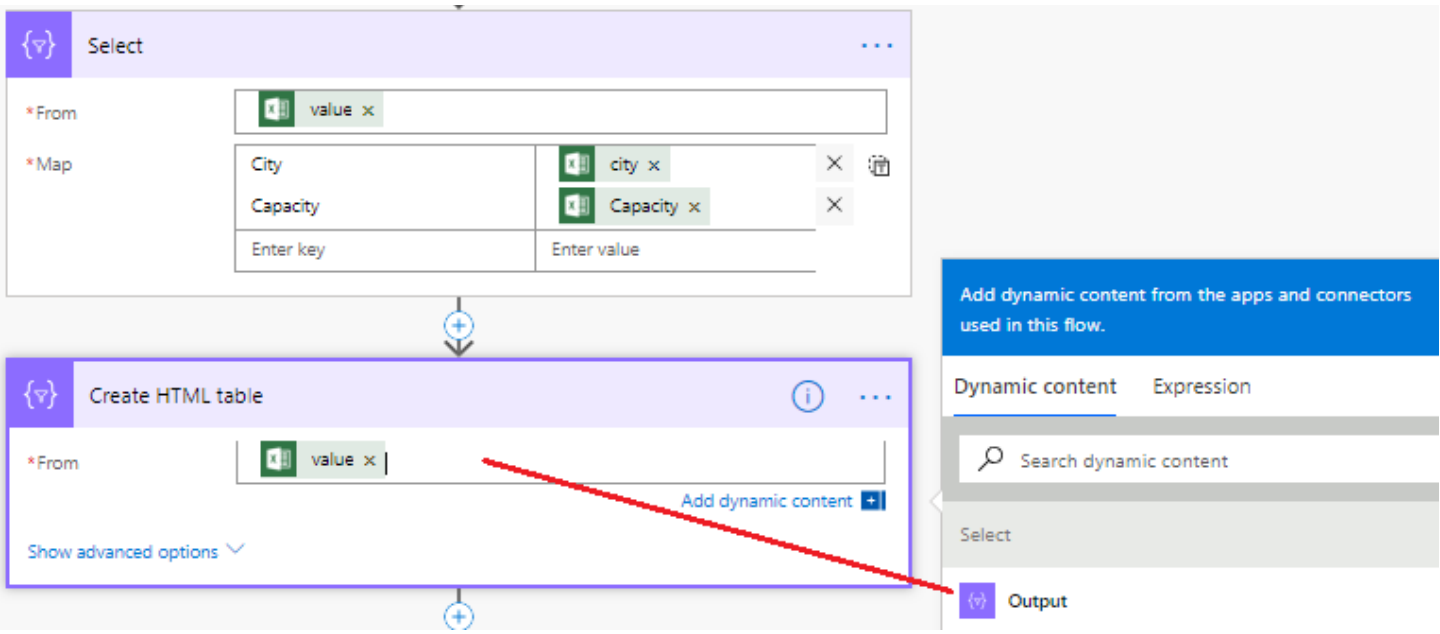
40. Add another map field for **Capacity**:



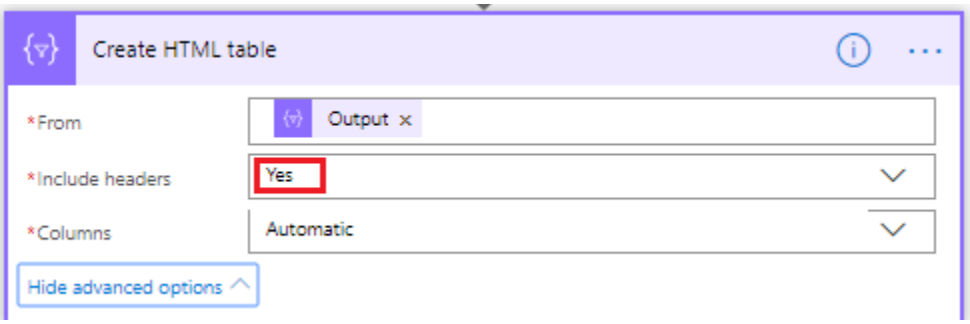
41. Just **after the Select action** add a **Data Operations - Create HTML table** action**:**



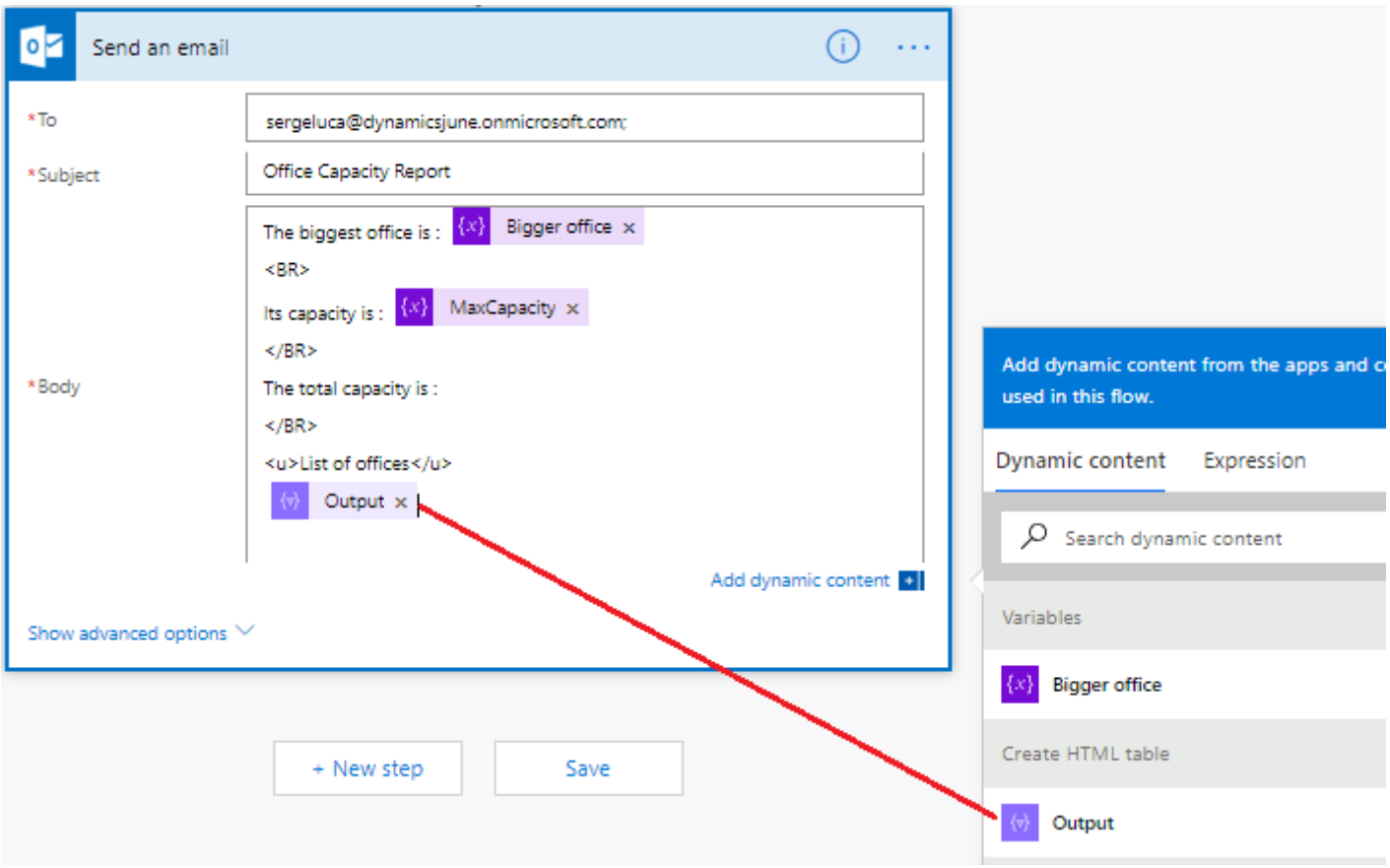
42. Move the cursor to the From field to show the **Dynamic content** panel and click the **Output** value of the **Select** action:



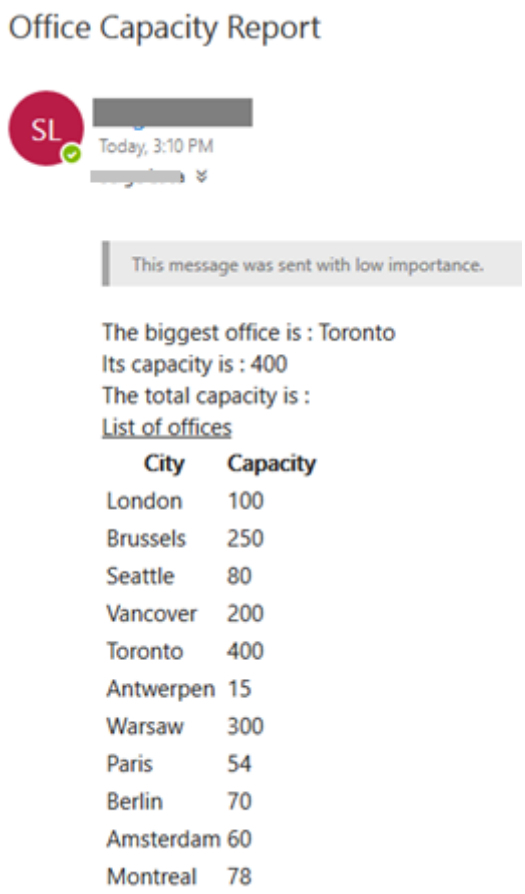
43. Click Show advanced options and set **Include headers** to **Yes**:



44. Go back to the **Send an email** action and update the **Body** text box to include the Create HTML Output value :

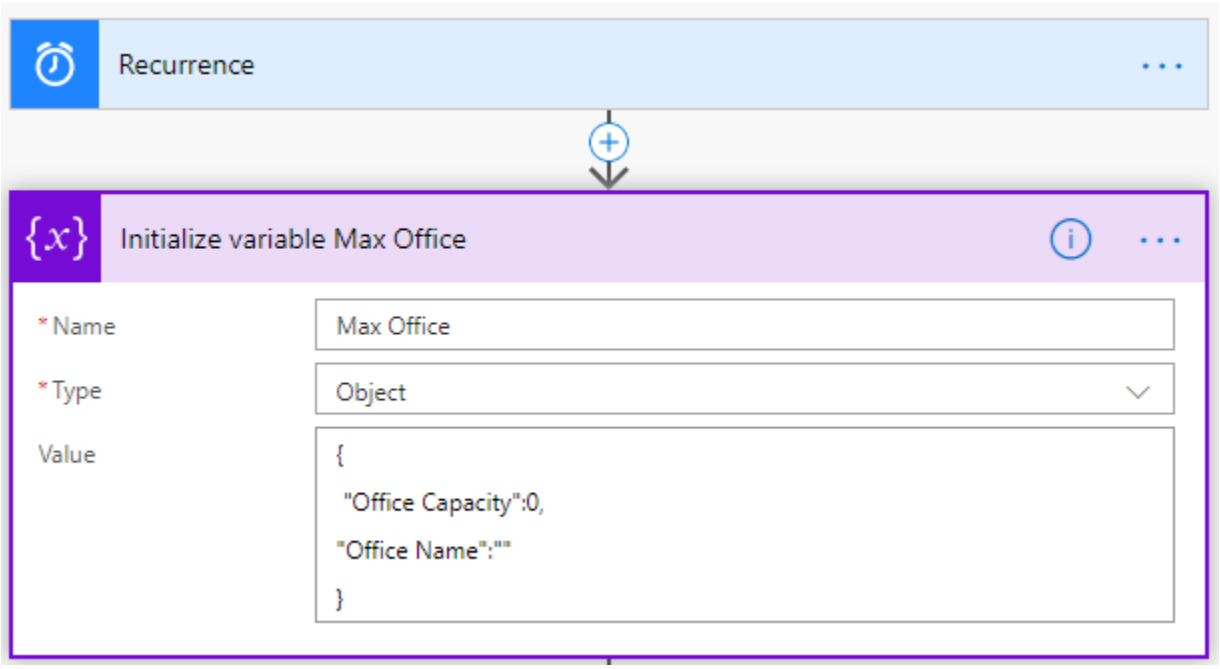


45. Test your flow and check your e-mail:

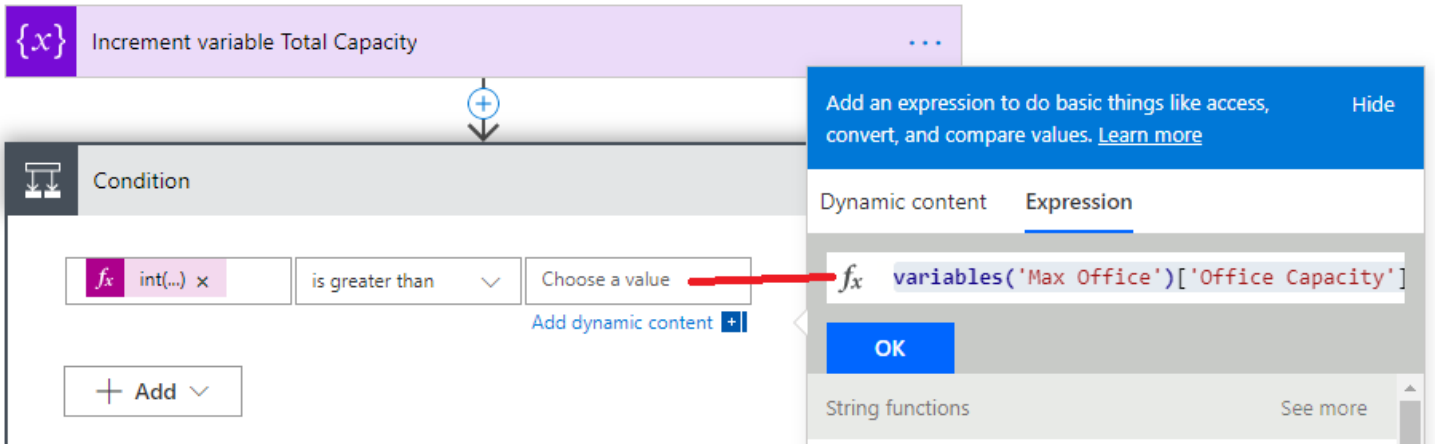


Optional exercise if time permits: use an object instead of dedicated variables.

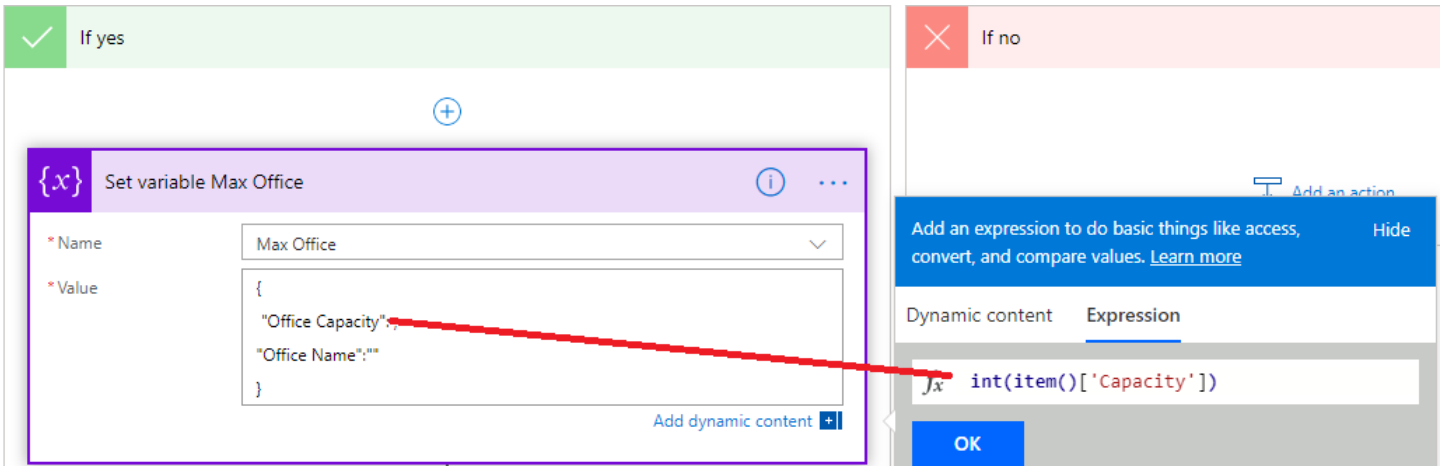
46. Create a variable **Max Office** as an object containing the following JSON data:

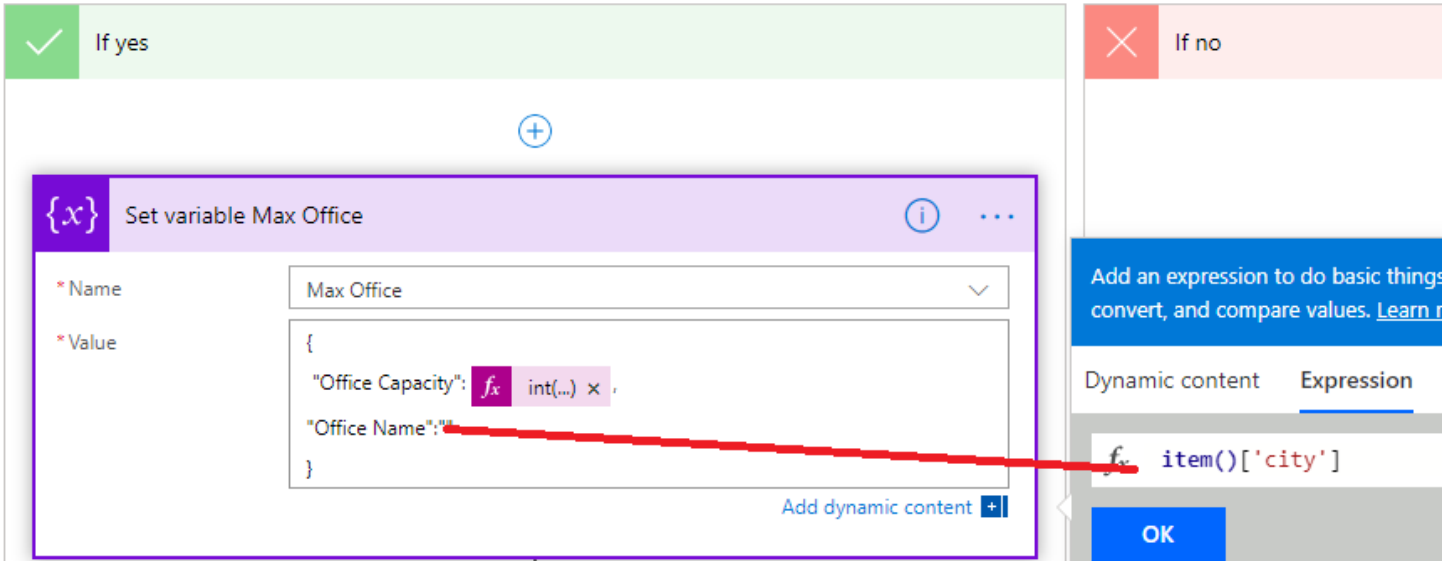


47. In the condition, use this new variable content:



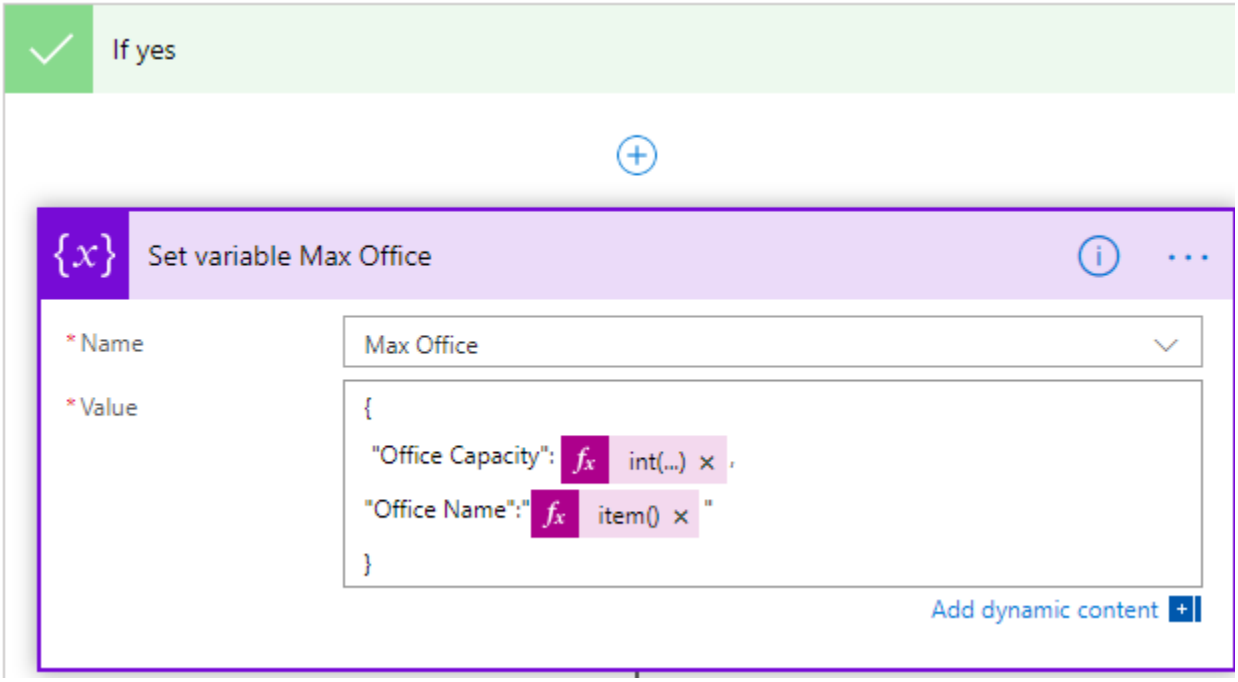
48. In the condition, create a Set variable that will update **Max Office** :





49. Remove the actions **Set Variable MaxCapacity** and set variable Bigger office previously defined

You should get something like this in the If yes branch:



50. Update the send an e-mail action with the new variable.

Lab 5. Dynamically add a person's manager as an approver (Approval Part 2)

Duration: 15 minutes.

Scenario: In this lab we will explore how a flow can dynamically assign an approval task to the current user's manager

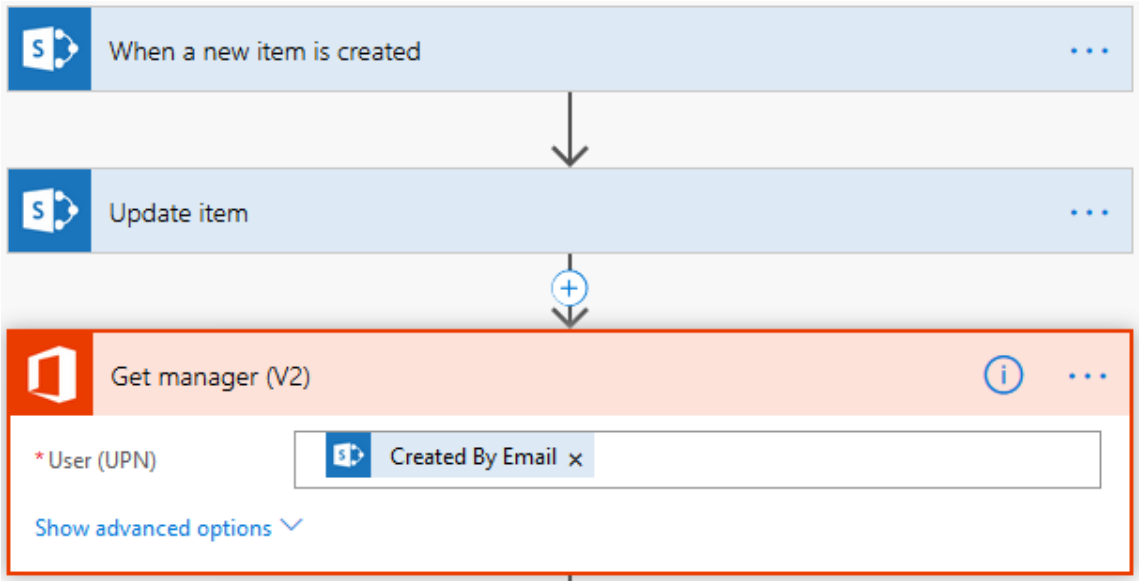
Prerequisites: The lab Approval (part 1) must be completed.

If you want to take a look at the solution: select the **Get my manager to approve a travel request** template.

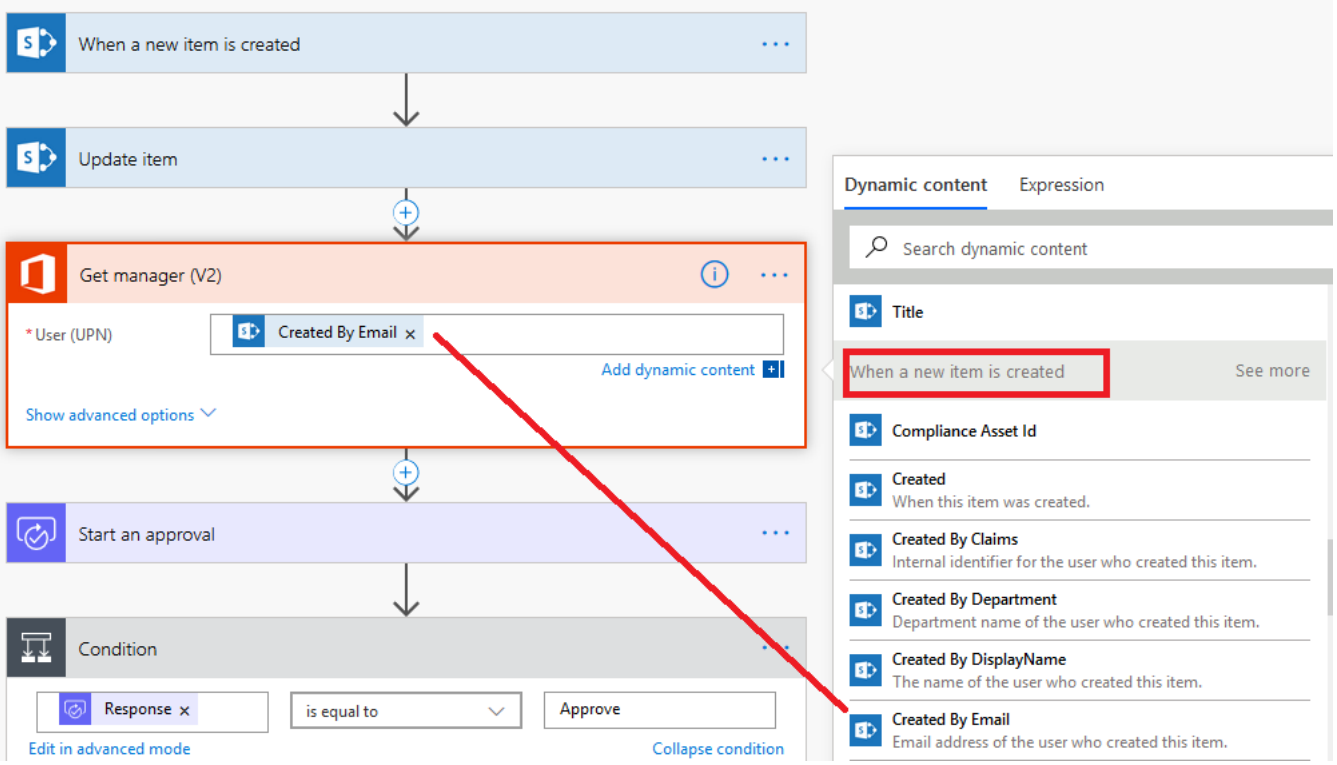
If you want to start the lab from scratch:

Tasks:

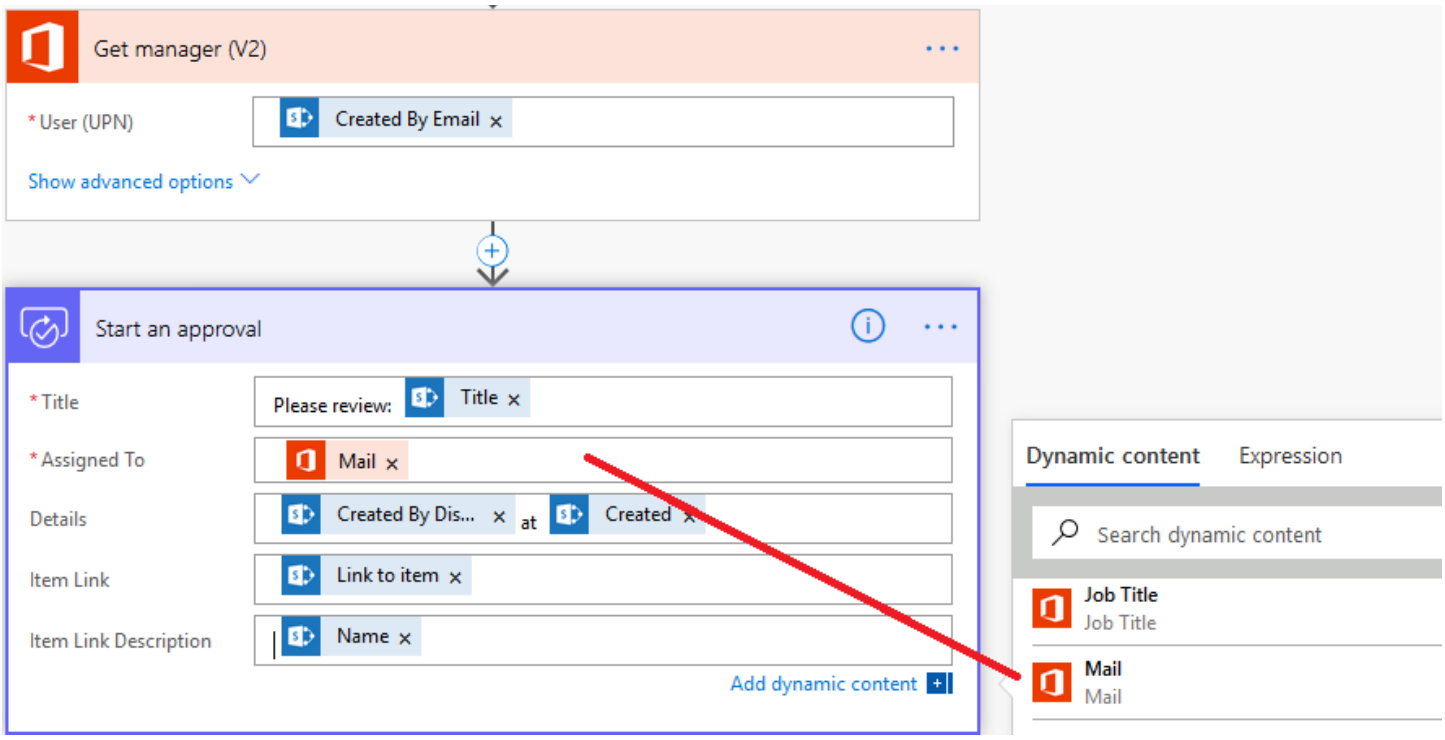
1. Go back to the previous expense approval flow, edit the flow and add a **Get manager (V2)** action just after the **Update item** action:



2. Set the **User (UPN)** to the **Created By Email** dynamic property of the **When a new item is created** action.



3. Update the **Start an approval** action **Assigned To** property with the **Mail** dynamic property of the **Get manager (V2)** action.



Note: Ensure the current user has manager defined in Azure Active directory.

4. Save and test the flow.
5. In order to approve the request, log into Microsoft Flow with the manager account to approve/reject your request from the Microsoft Flow approval center.

Lab 6. Task overdue

Learning objective: Controls, manipulation Excel, conditions, Date & time, expressions

Duration: 20 minutes

Scenario: We have an Excel document with a set of tasks, where some of these tasks are overdue. You will create a flow that will find all overdue tasks and will send a report of these tasks.

If you want to take a look at the solution: select the **Check task deadlines and send an email for overdue tasks** template.

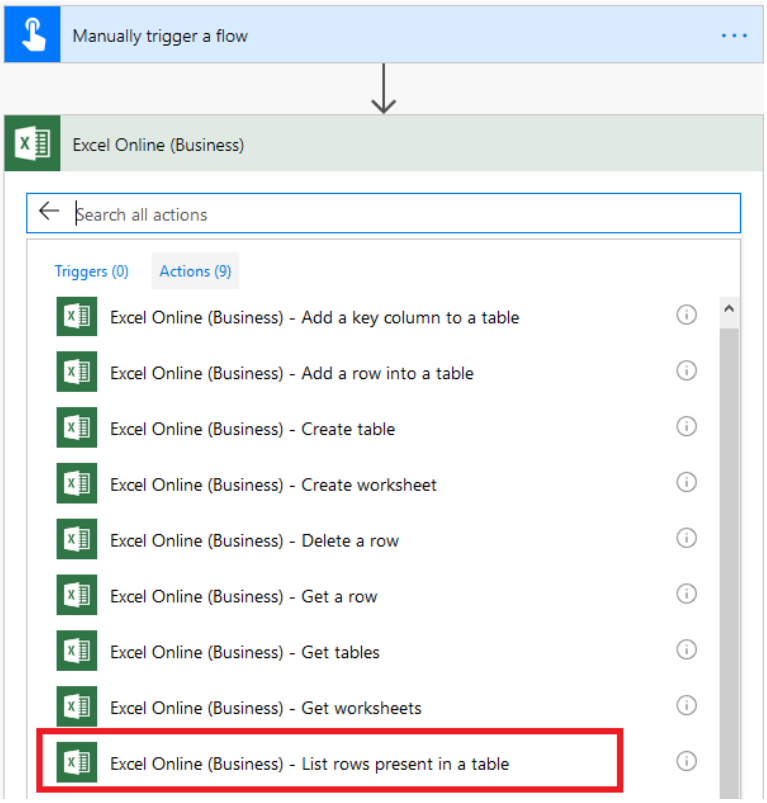
If you want to start the lab from scratch:

Tasks :

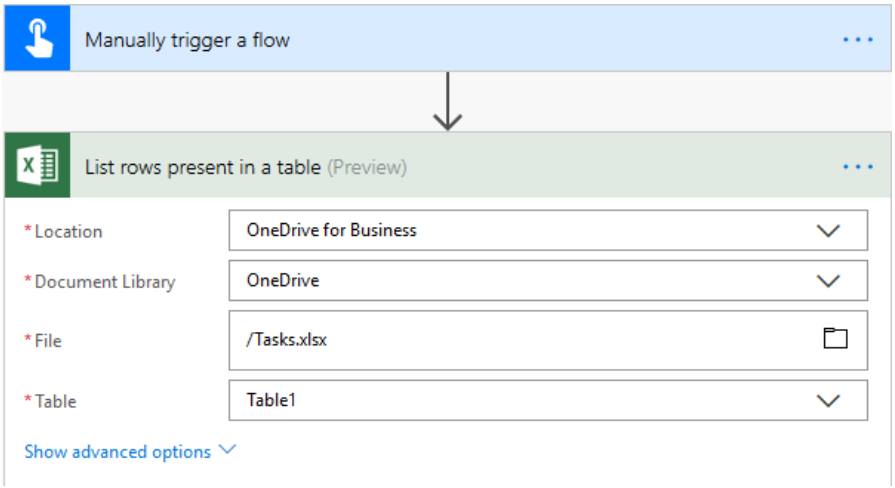
1. Create a new Excel file in your OneDrive for Business that looks like this (use the same columns). Before adding data in the Deadline column, make sure it is in **Text** format.

Task	Status	in charge	Deadline
Feed the cat	Started	user1@dynamicsjuly.onmicrosoft.com	1/1/2018
Call jon	Not Started	user1@dynamicsjuly.onmicrosoft.com	2/4/2018
Patch sql server	Started	user1@dynamicsjuly.onmicrosoft.com	7/6/2018
Call mum	Started	user1@dynamicsjuly.onmicrosoft.com	6/7/2018
Buy fruits	Started	user1@dynamicsjuly.onmicrosoft.com	8/8/2018
Call Kent	Not Started	user1@dynamicsjuly.onmicrosoft.com	8/8/2018
Buy a Porsche	Done	user1@dynamicsjuly.onmicrosoft.com	1/9/2018

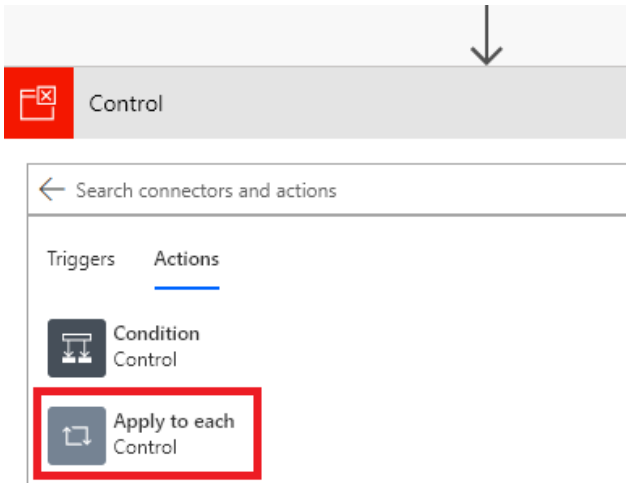
2. Name it **Tasks.xlsx**
3. Change the value of the **in charge column** to your e-mail address and adjust some deadline value.
4. We want to write a flow that will loop through all tasks and that will check if the task is overdue. To do so, **Create a new Instant flow from blank** and use a **Microsoft Flow as a trigger.** Name it **my Overdue Tasks.**
5. The flow needs to connect to the **Tasks.xlsx** file, so add an **Excel Online (Business) – List present in a table** action (Not OneDrive !!!) as illustrated below:



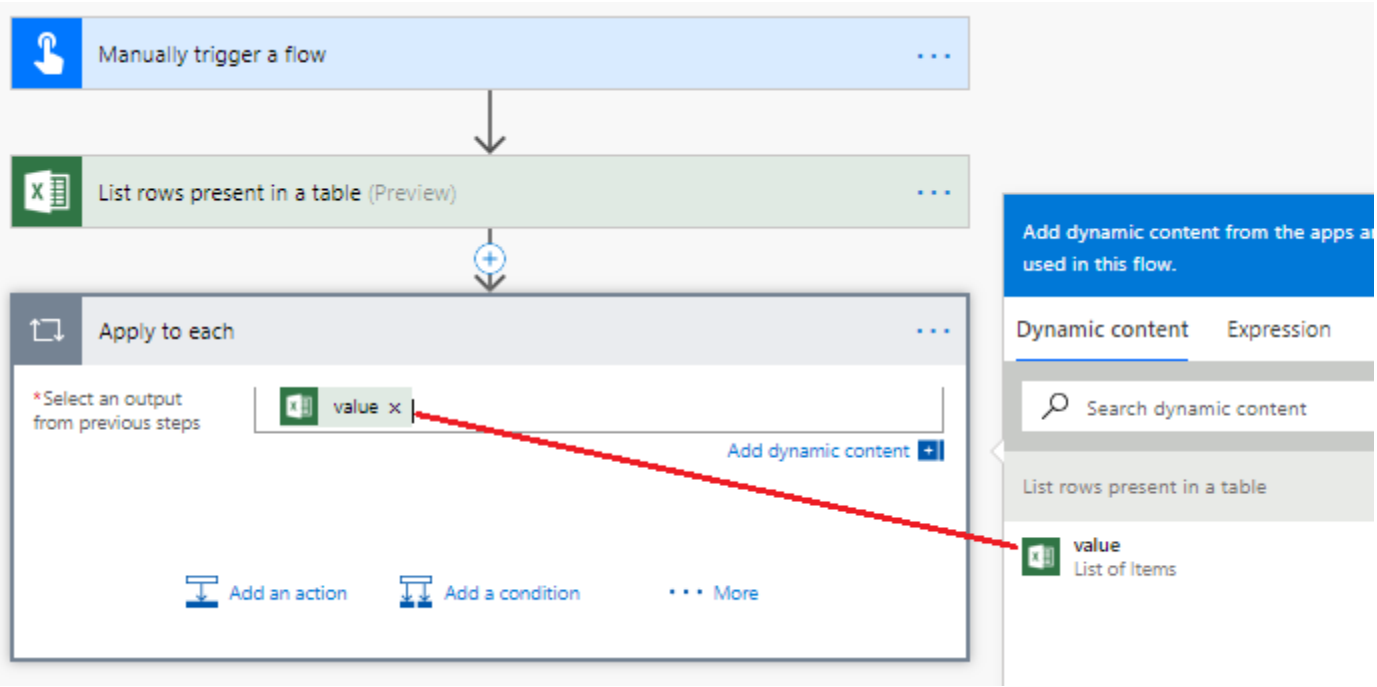
6. Set the action properties as illustrated in the next picture:



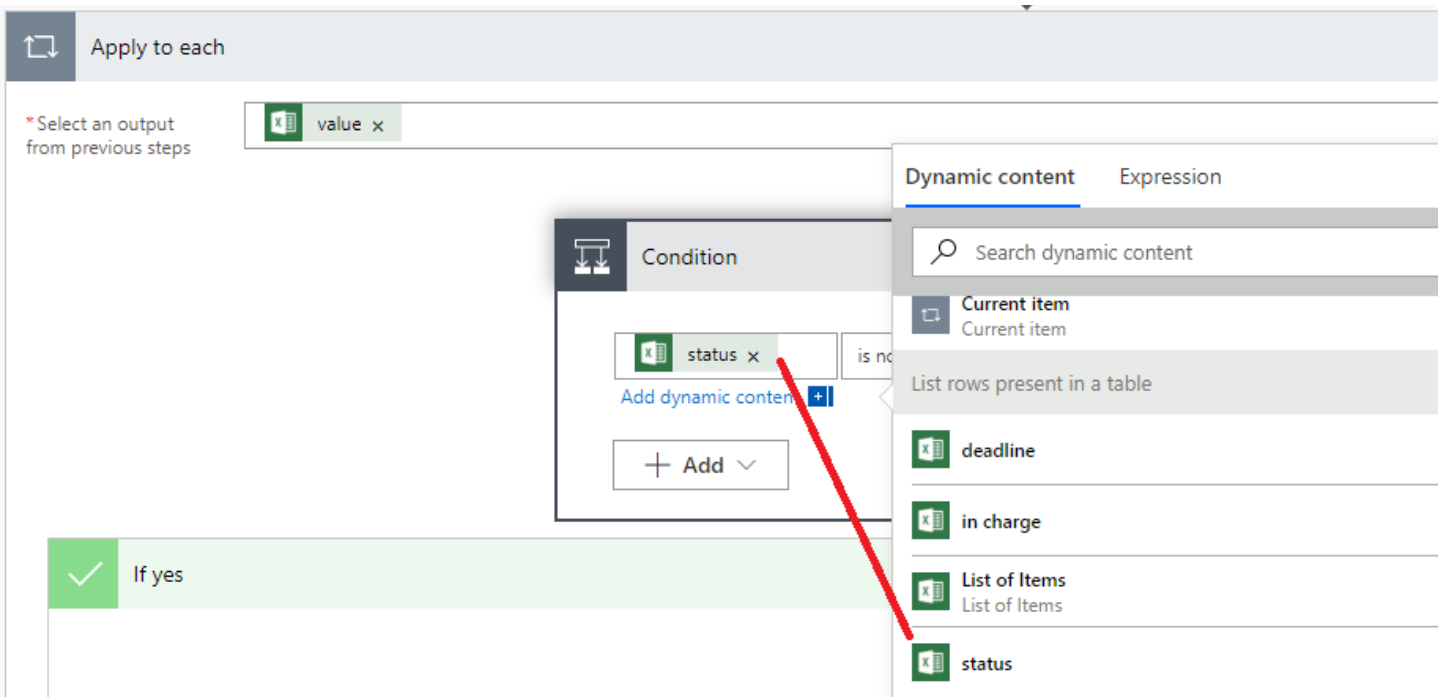
7. Let's loop through all tasks, so we need to add an **Apply to each** :



8. Select the value **Dynamic content** in the **Apply to each** action:



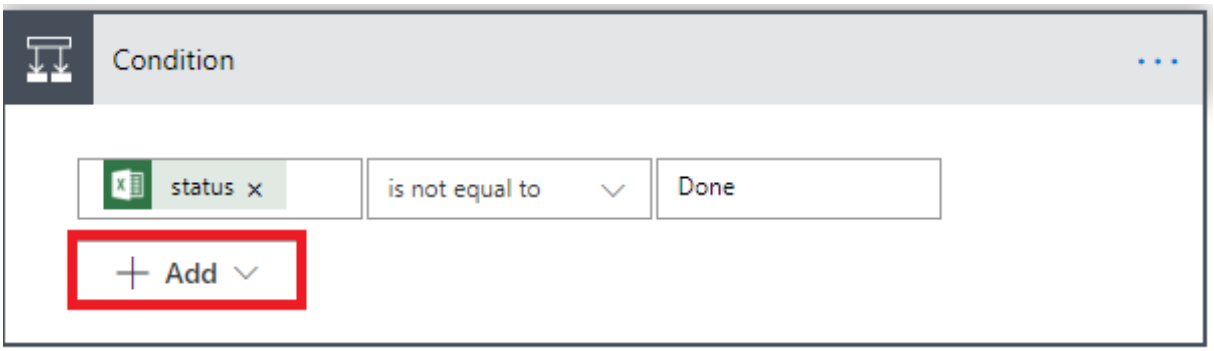
9. Add a **Condition** to filter the task where **Status** is not equal to **Done:**



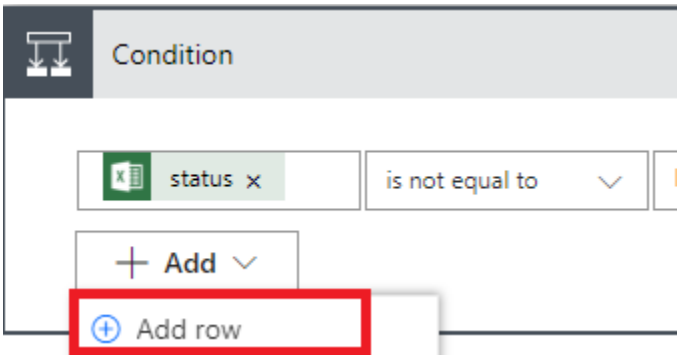
Make sure the operator used is "is not equal".

10. Add a new sub condition that will check if the task is overdue:

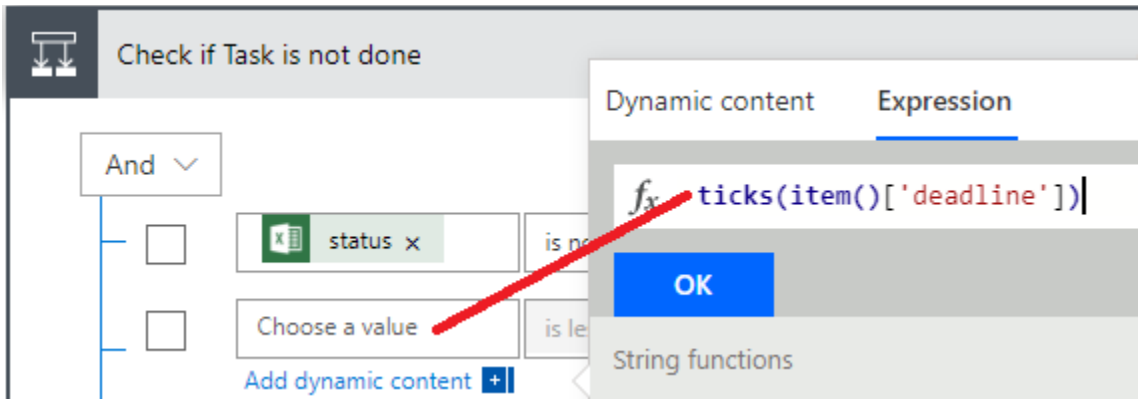
a) Click Add:



b) Select Add row:

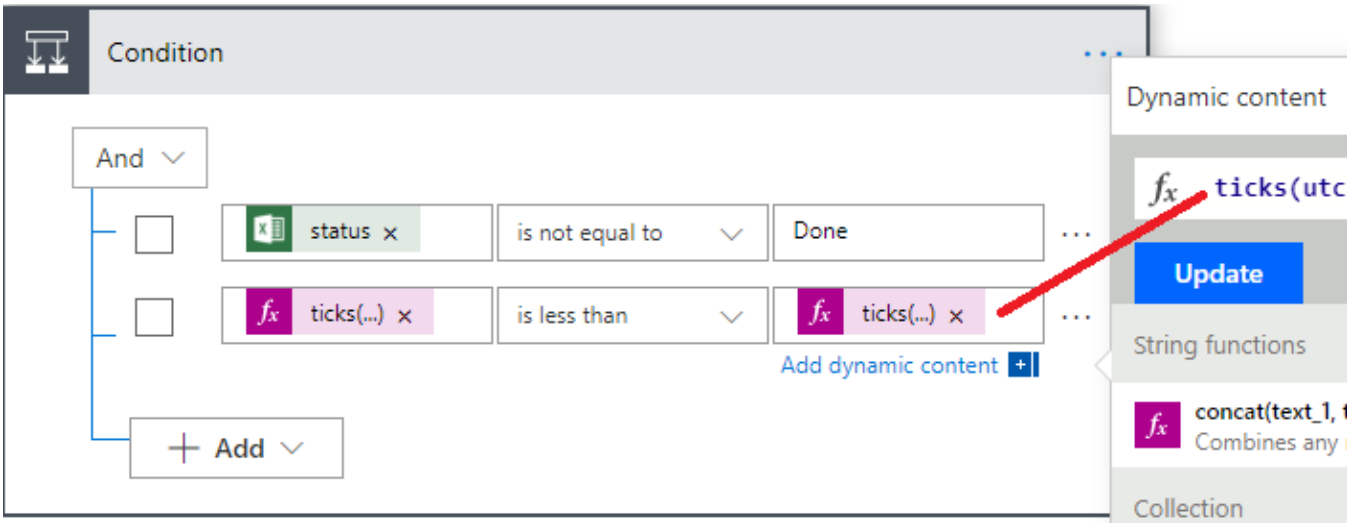


c) In "Choose a value", type the expression (and click Ok) :



Since a timestamp is in string format, the **ticks** expression returns the number of ticks (100 nanosecond intervals) since 1st January 1601. By using ticks, we can compare two different timestamp values.

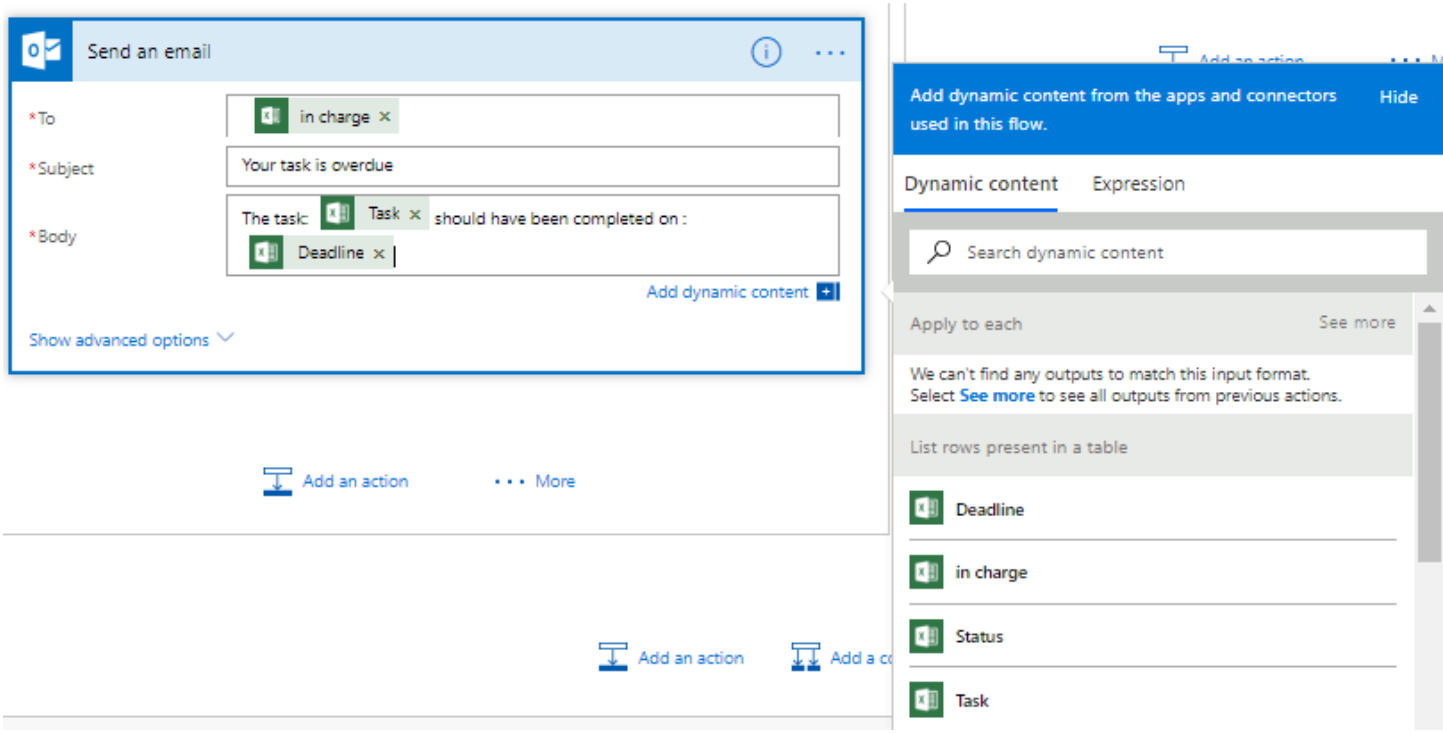
d) Select the operator is less than
e) Type the following expression (and click Ok):



11. In the **If yes branch**, add a new **condition** where we will check if the due date is overdue:

12. Rename the condition to **Check if overdue** to provide your flow with more clarity:

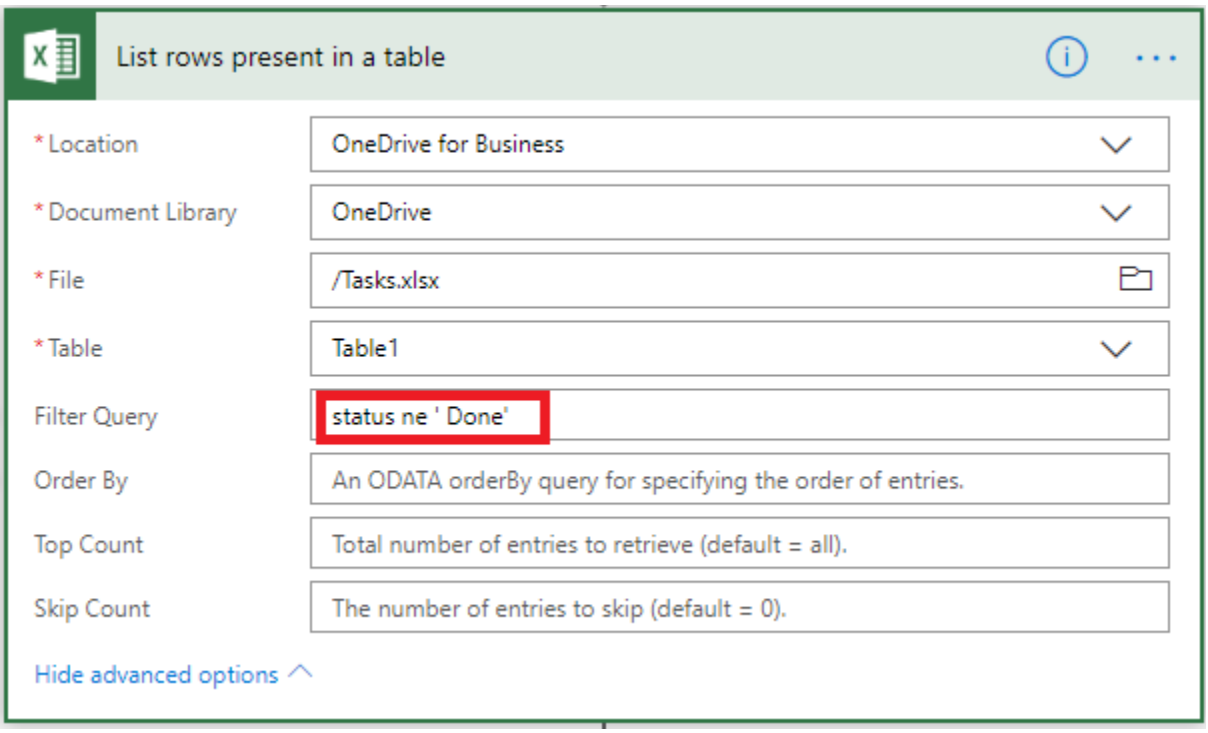
13. **Send an e-mail** to the person in charge of this task and fill-in the e-mail properties as following:
    a. The **To** field should get the Excel in charge value
    b. The **Subject** should be: "your task is overdue"
    c. The **Body** should be:



Note: If you cannot find the excel fields, click on **See more** link.

14. Run the flow and check your e-mail.

Note : instead of filtering the Tasks with the Status Done in a condition, you can also keep it simpler by using an oData Filer in the List rows present in a table action :



Lab 7. Creating a Business Process Flow

Learning Objectives: Be able to create a Business Process Flow and a Model driven app that will consume it.

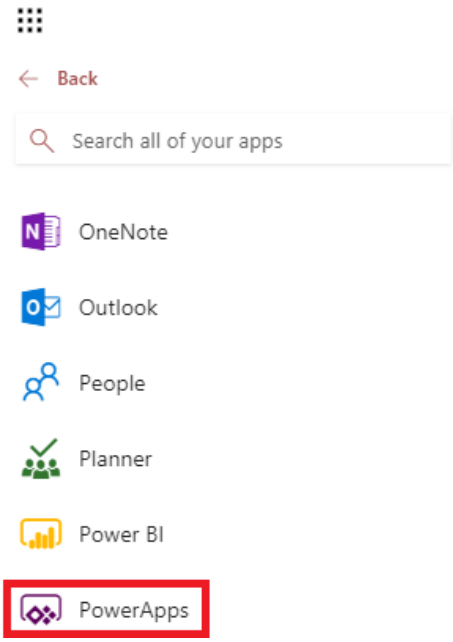
Scenario: You will create a Business Process Flow (BPF) that illustrates how to approve a loan. You will have to create business entities in CSD 2; these entities will be consumed by the BPF.

Prerequisites: PowerApps Plan 2 (trial). You can sign up for a PowerApps Plan 2 (trial) here: https://powerapps.microsoft.com/en-us/ with your current account. This step is necessary if you already have a PowerApps Office 365 Plan. You might have to sign-out and sign-in before seeing the new Plan 2.

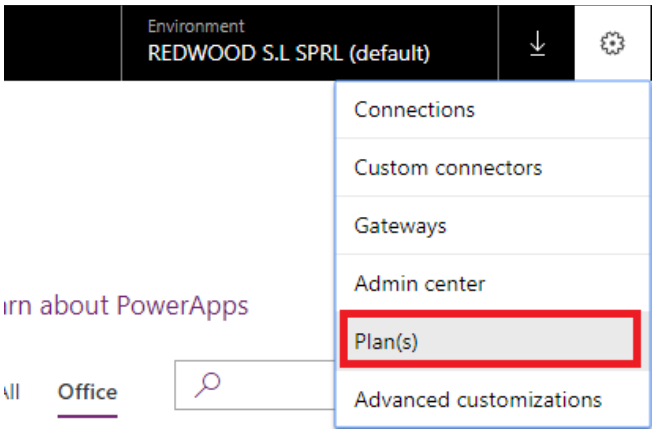
Duration: 35 minutes

Tasks:

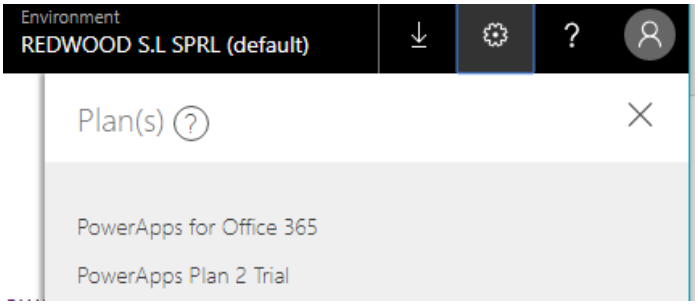
1. Go to the **PowerApps Studio**: *https://powerapps.microsoft.com or by clicking on the* **PowerApps app** in the **Office 365 App Launcher**:



2. If you are working on your own tenant, you first must have **PowerApps Plan 2** in order to be able to create **Model driven apps**. To check your PowerApps plan, go to the following PowerApps menu:

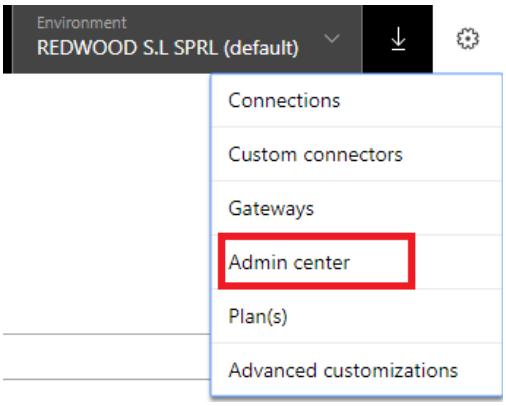


You should get **plan 2**:

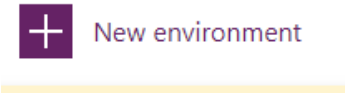


If that is not the case, move to the next step.

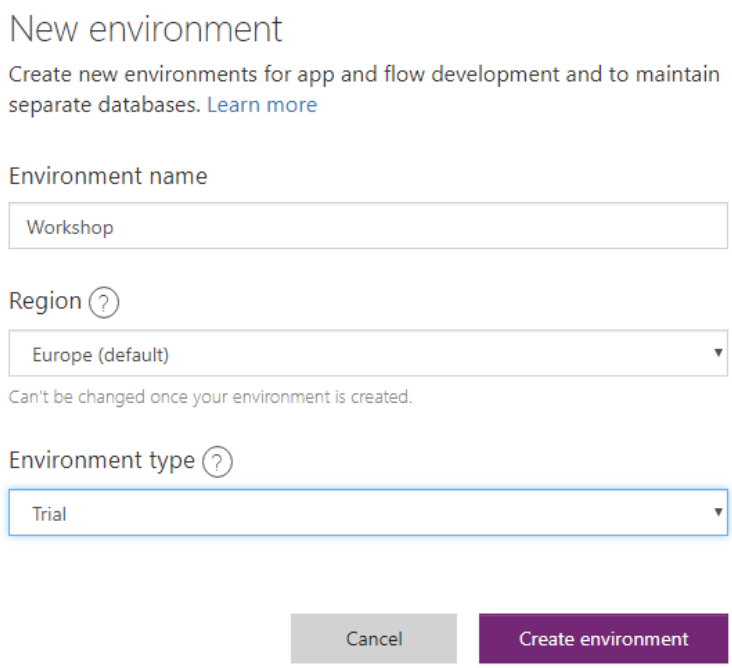
3. You can have a free trial version of PowerApps Plan 2. Connect to the [web site](https://powerapps.microsoft.com/en-us/pricing/) *https://powerapps.microsoft.com/en-us/pricing/* to start your trial. When the trial is generated, go to the **PowerApps Admin center**:



4. Click **New environment**:

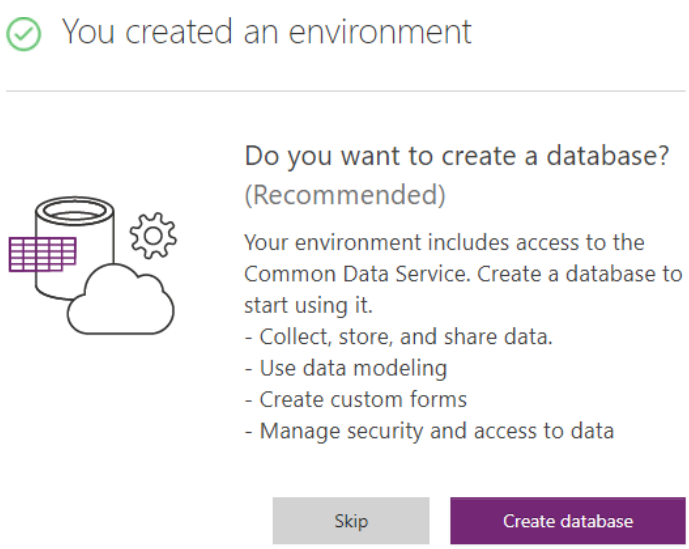


5. Fill in the required information:

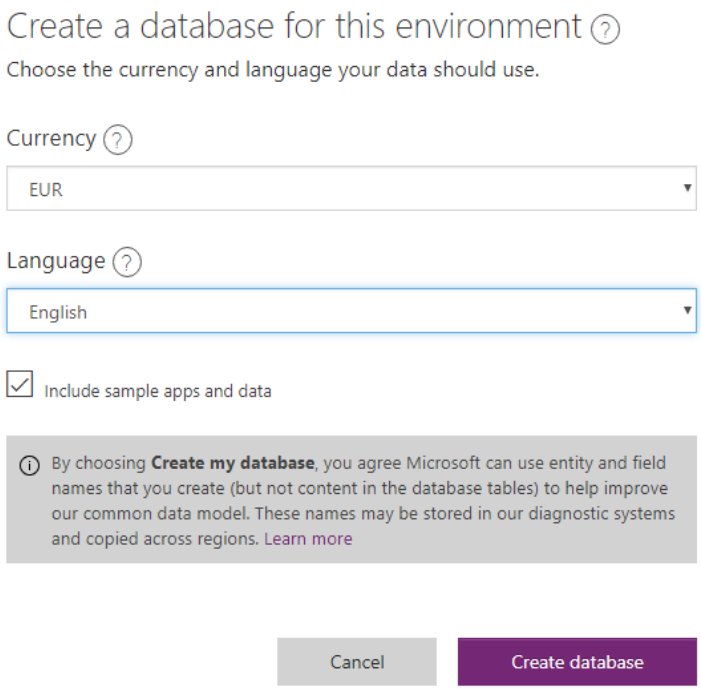


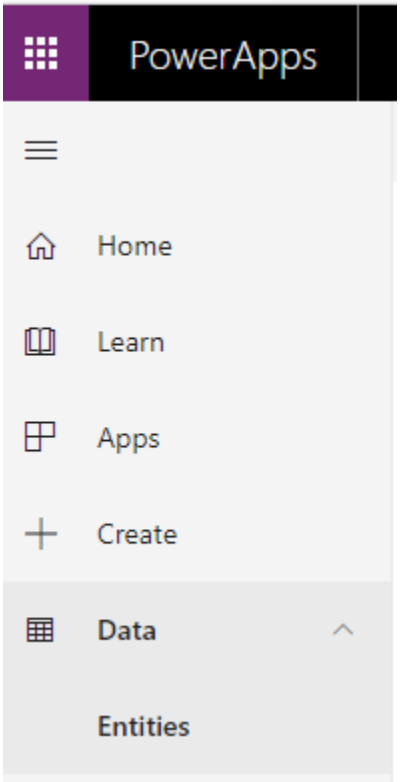
6. Click **Create environment**.
7. The next step (Creating a Database) should not be done if your PowerApps trial account is part of the tenant created by the trainer. You will use the shared database

When requested, click **Create Database**:



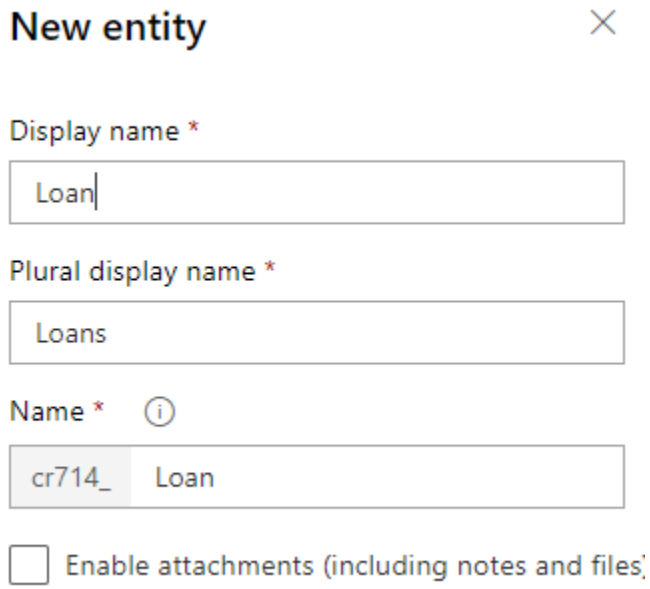
8. If the next screen you will provide more details on the **database settings**:



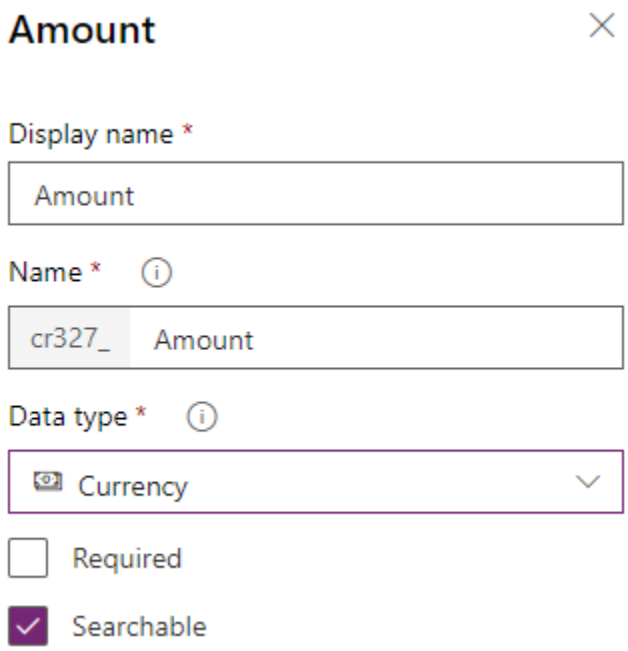
9. Click again on **Create database**.
10. Go to the list of environments and click on the newly created environment and you will see that the database is being built.
11. Go back to the PowerApps portal https://powerapps.microsoft.com and select the new environment:
12. Select the Data Menu and click on Entities:
13. 

14. Click new Entity and create a new Entity called Loan:

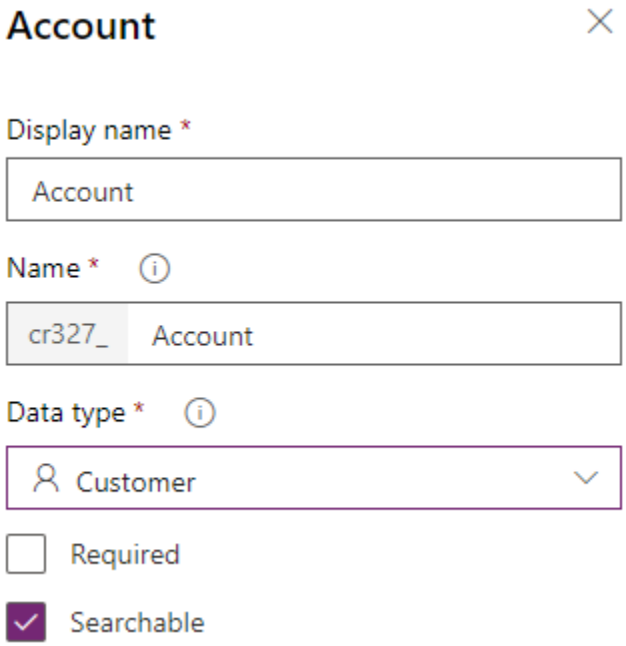
    Click Next. Click Add field to add a new field in this loan entity.



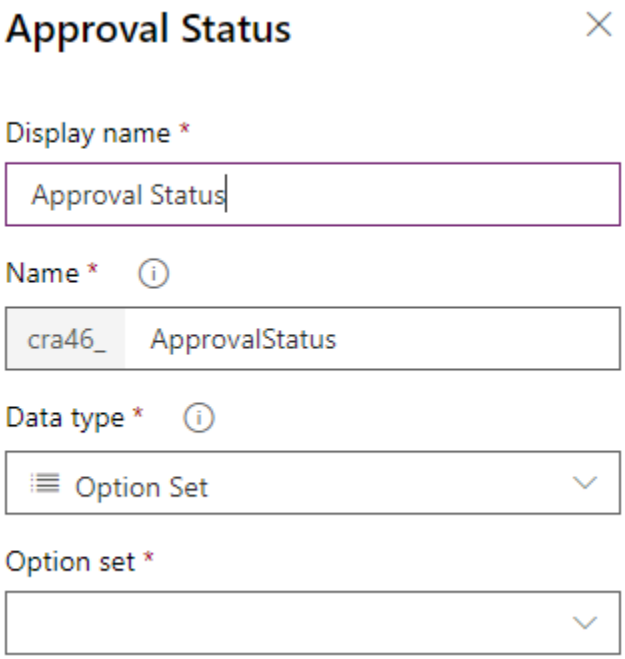
15. Create a field **Amount** (Menu + Add field) like shown here:



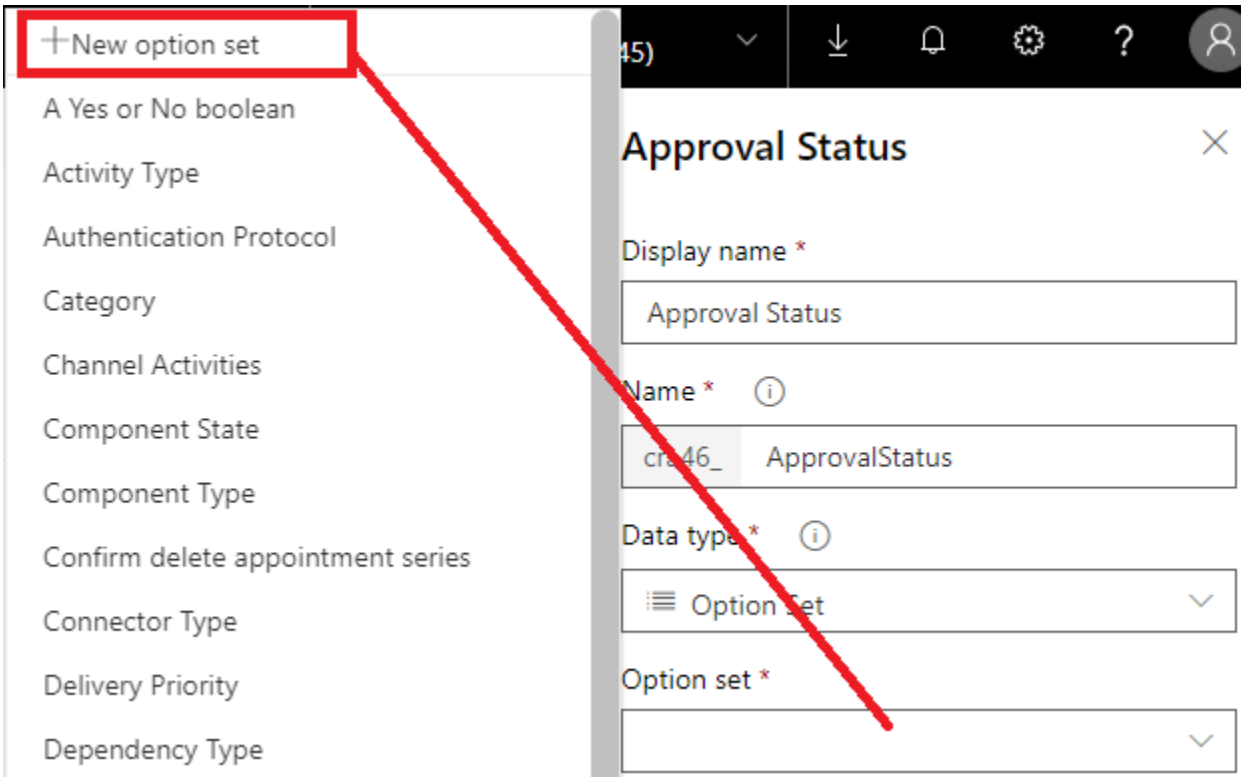
16. Create a field **Account** (Data type Customer):



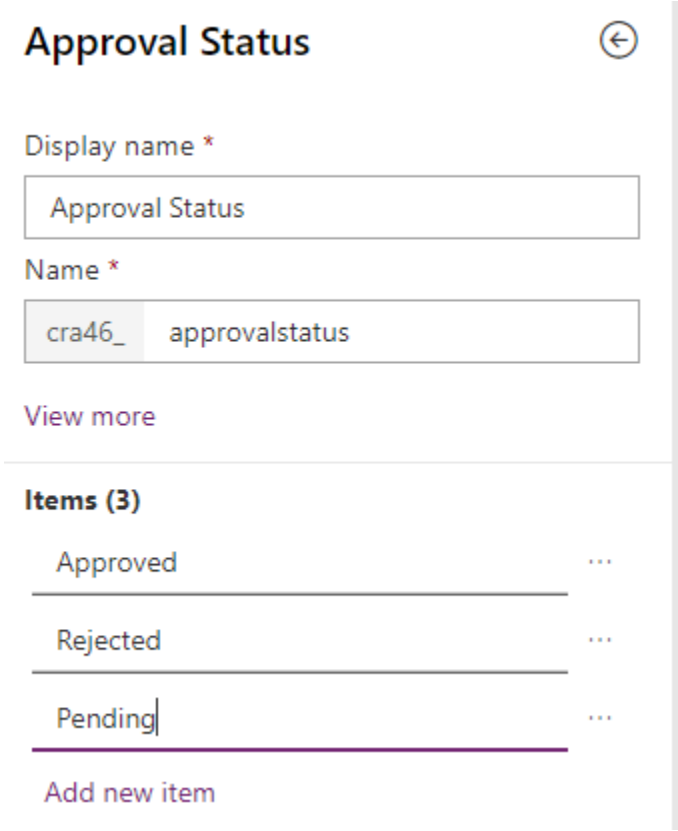
17. Create another field **Approval Status** based on the Option Set datatype

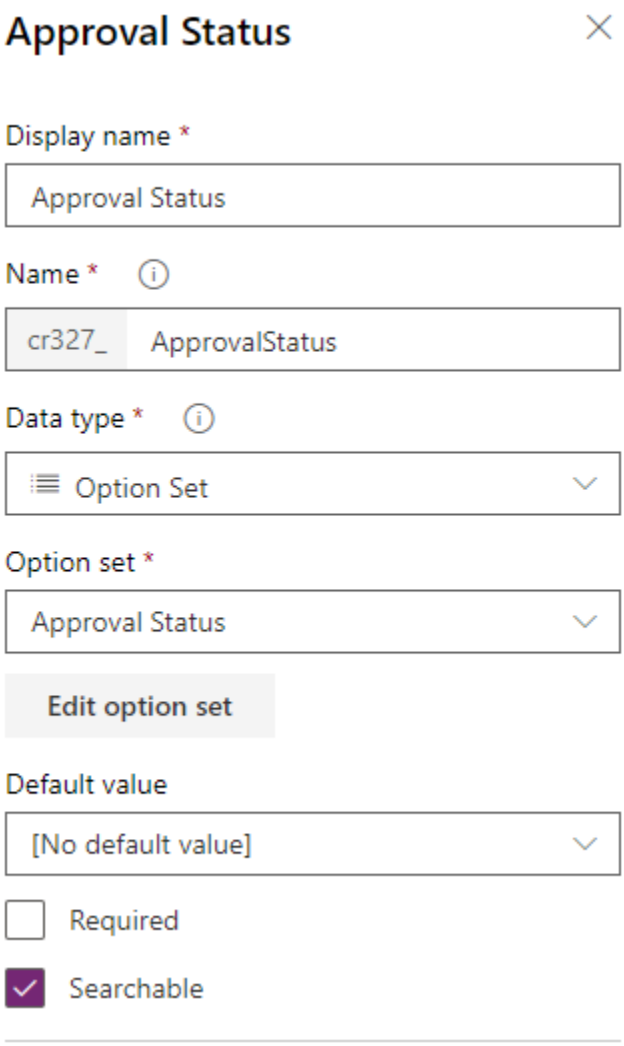


18. Click **Option set** and select **New option set** :

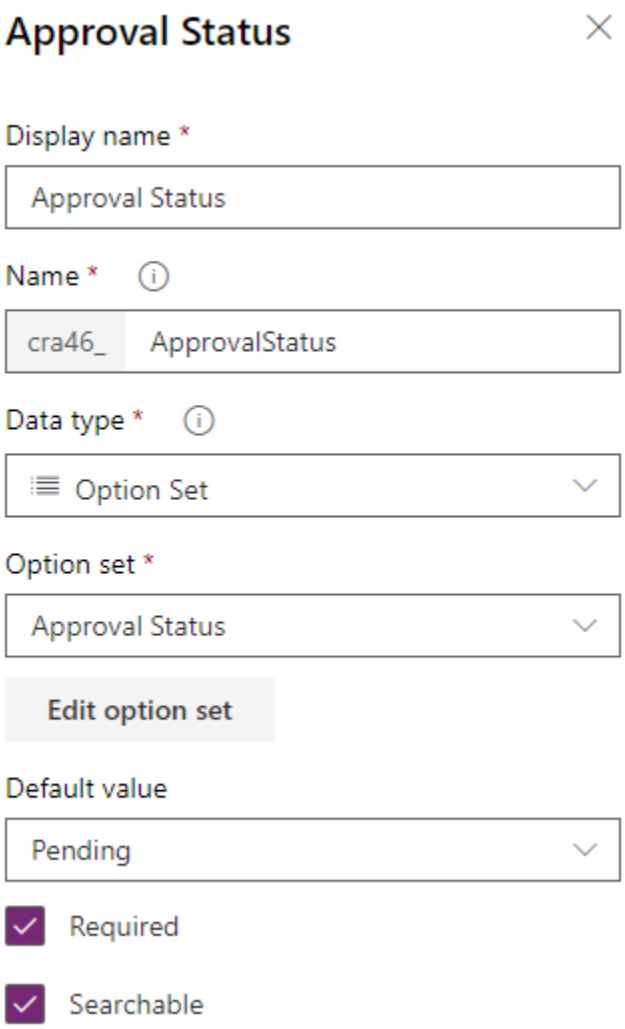


Provide 3 options: Approved, Rejected, Pending:



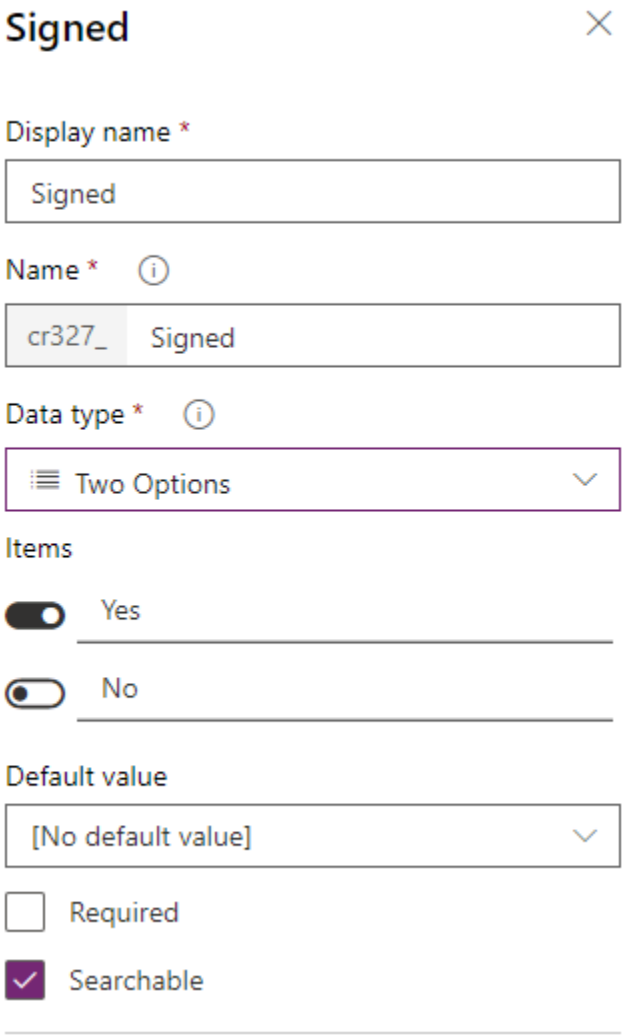


Click Save and Make Pending the default value:

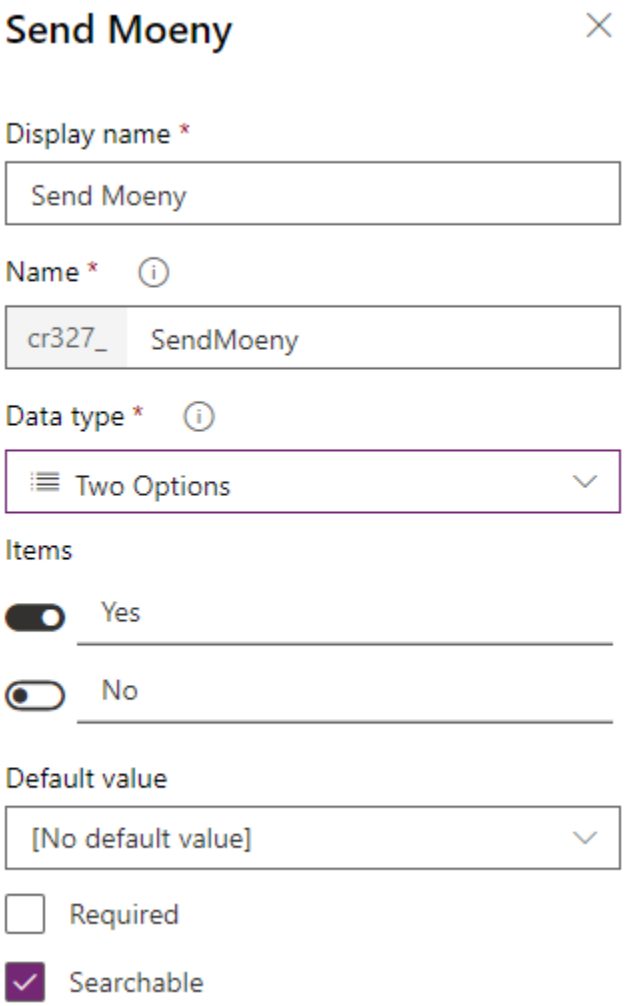


Click Done.

Create a field Signed based on the 2 options Data Types (Yes, No) and No should be the default value:



Create last field : Send Money



Save your field and save the entity.

The following table summarizes your entity Loan fields and settings=

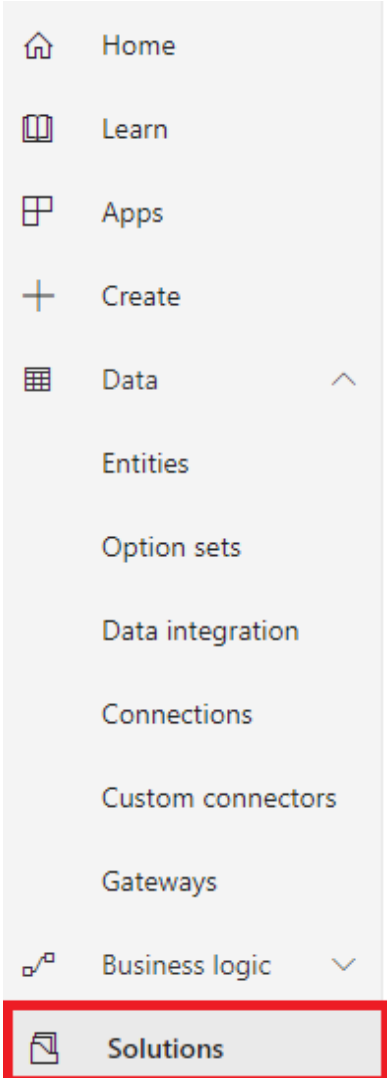
Solutions > Common Data Services Default Solution > Loan

Fields Relationships Business rules Views Forms Dashboards Charts Keys Data

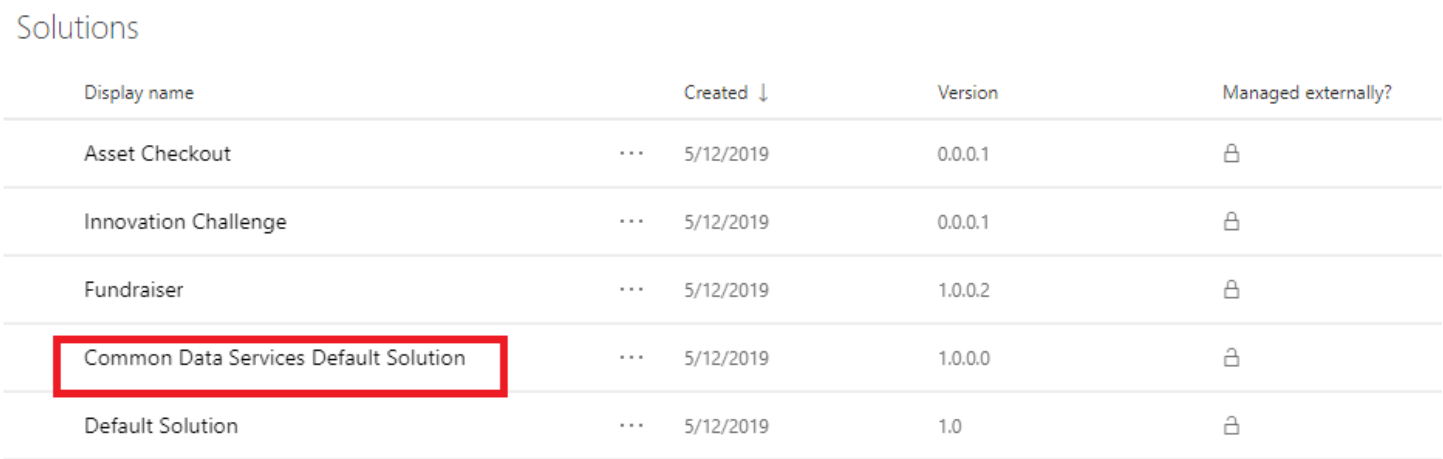
Display name	Name	Data type	Type	Required	Searchable
Account	cr327_account	Customer	Custom		✓
Amount	cr327_amount	Currency	Custom		✓
Amount (Base)	cr327_amount_base	Currency	Custom		✓
Approval Status	cr327_approvalstatus	Option Set	Custom		✓
Currency	transactioncurrencyid	Lookup	Standard		✓
(Deprecated) Traversed Path	traversedpath	Text	Standard		
Exchange Rate	exchangerate	Decimal Number	Standard		✓
Name	cr327_name	Text	Custom	✓	✓
Send Money	cr327_sendmoney	Two Options	Custom		✓
Signed	cr327_signed	Two Options	Custom		✓

Now we will define the loan entity as a Business process flow entity. To do so we need to move into classic mode.

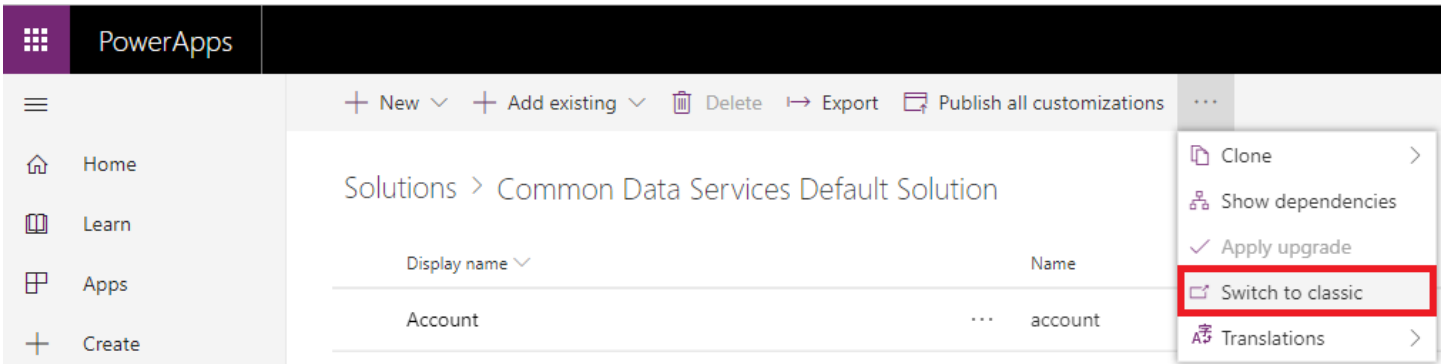
Click solutions in the left panel :



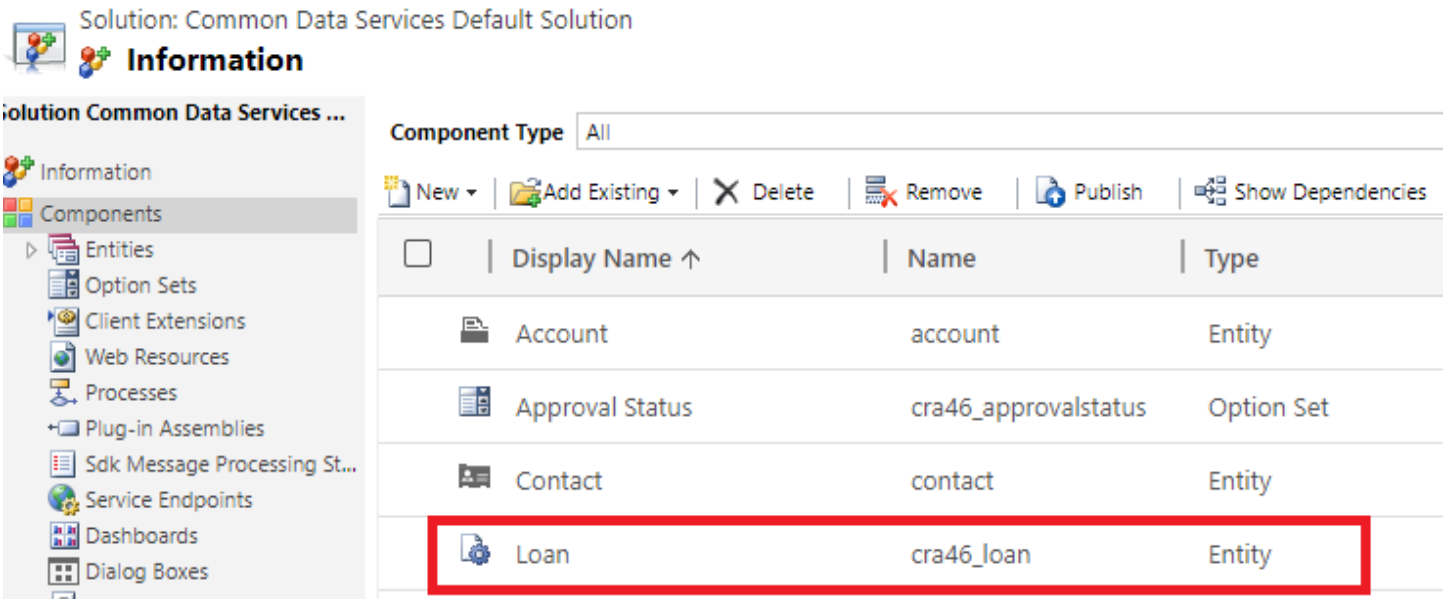
Click Common Data Services Default Solutions :



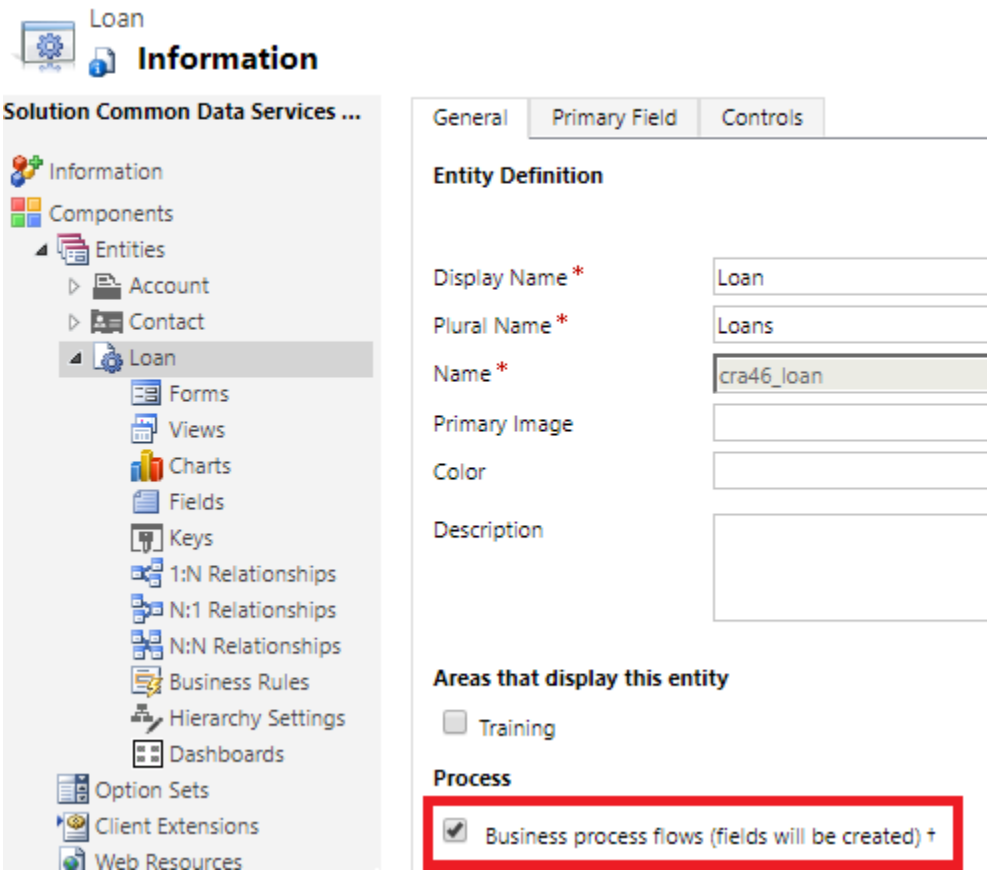
In the menu above, click Switch to classic :



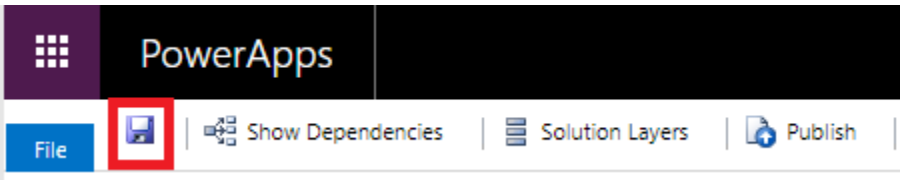
You should see your custom Loan entity:



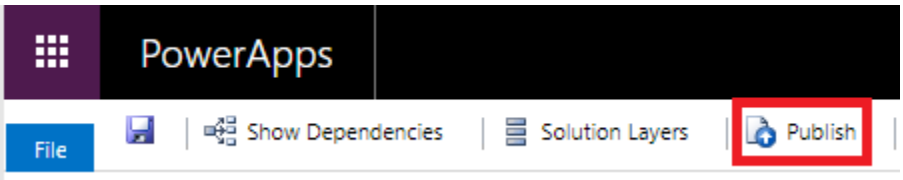
Select the Loan entity and double click on it; in the next scree, check the option Business process flows:



Click the Save icon.

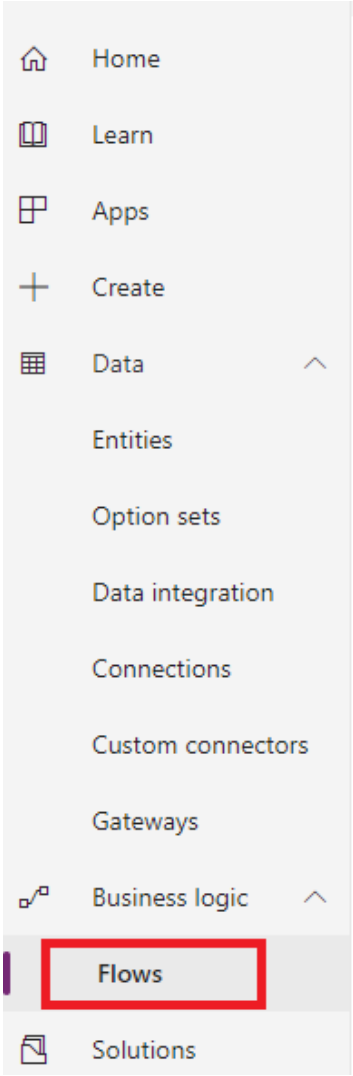


Click Publish.

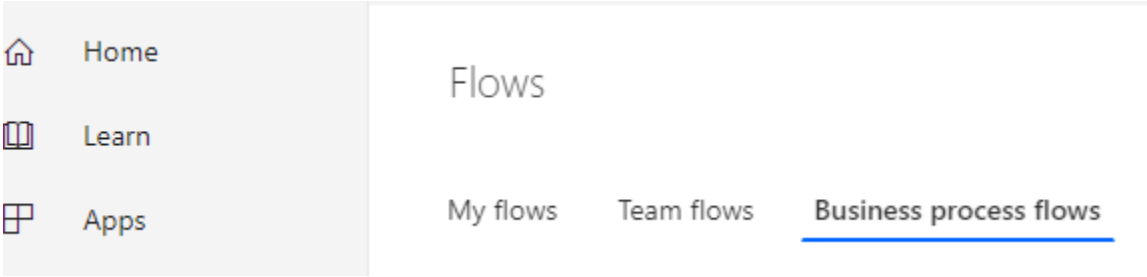


Let's get back to flow to create a new BPF flow. Make sure you are on the good environment.

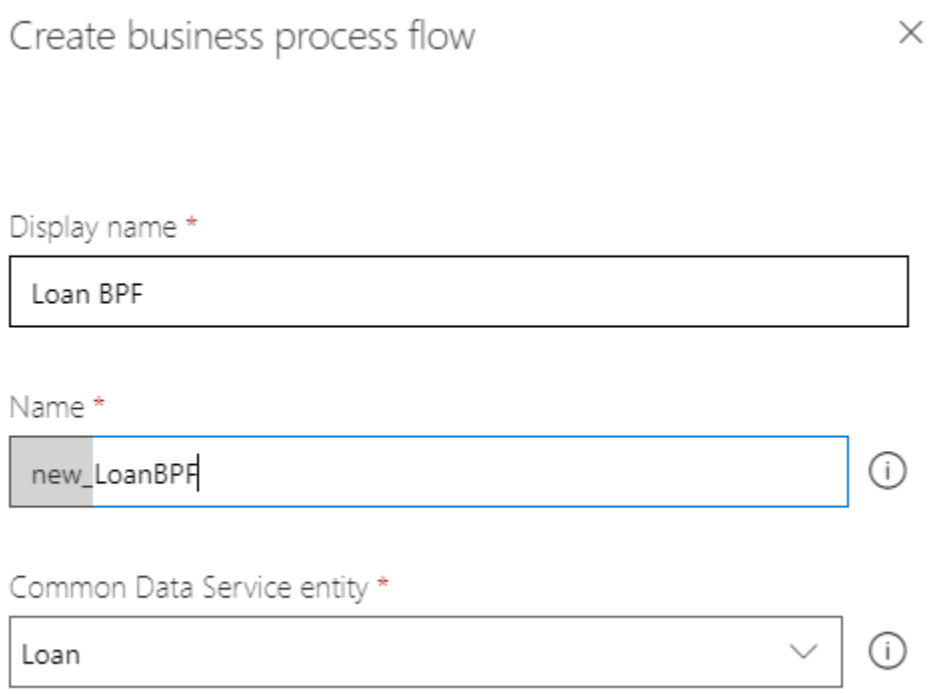
Click on Flows in the left panel.



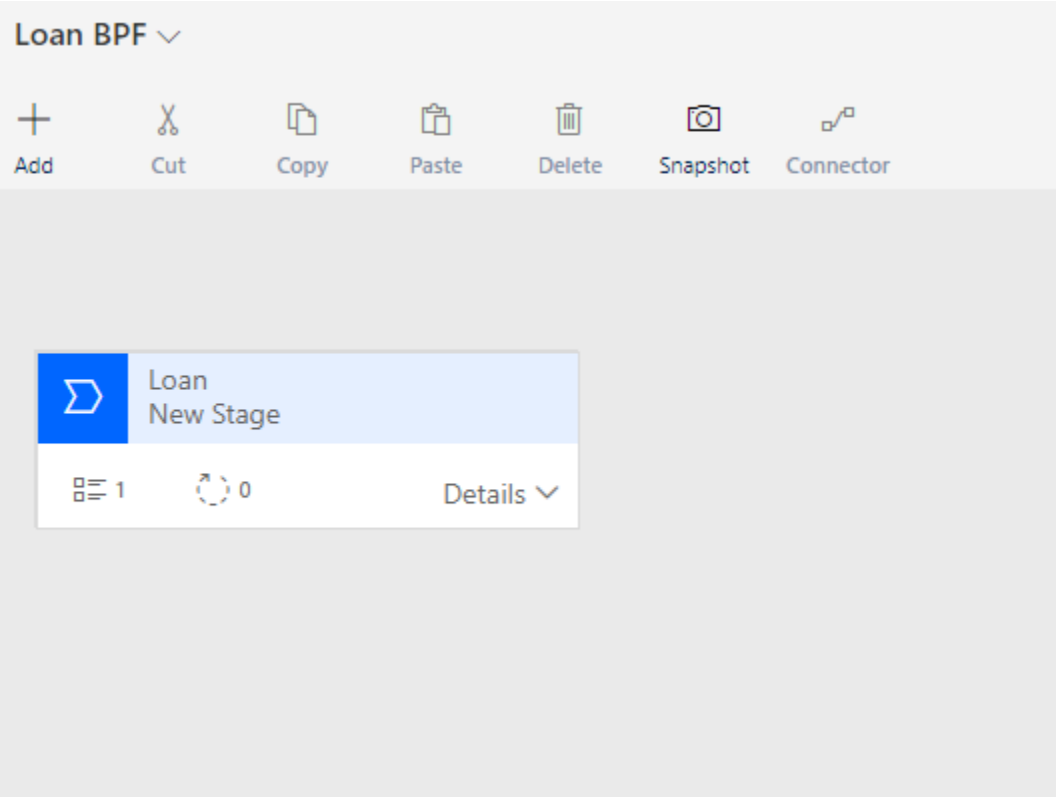
Click Business process flows:



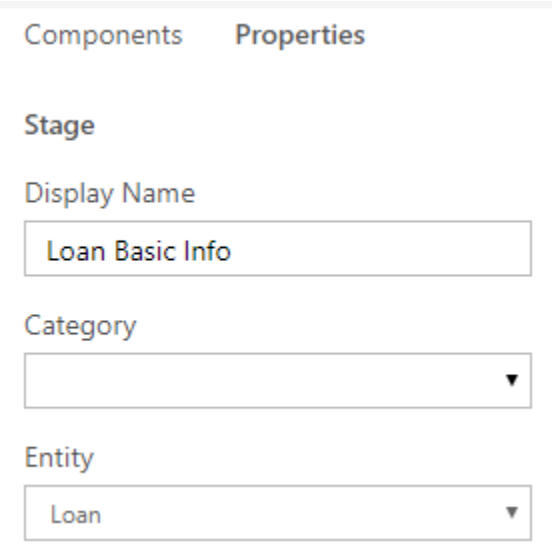
Fill in the bpf new and associate it with your loan entity in "Common Data Service entity:



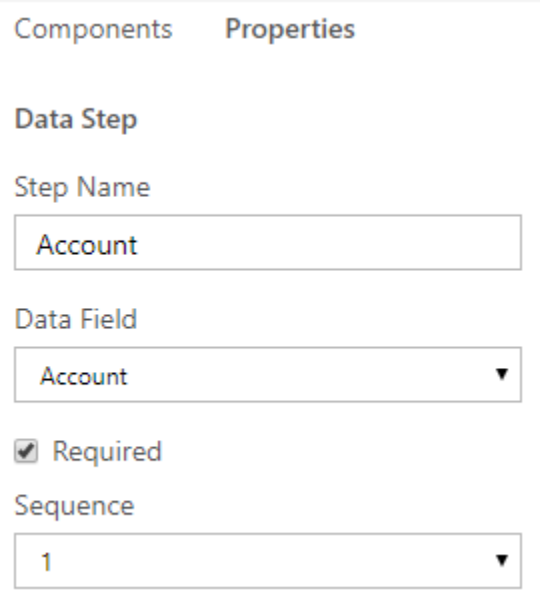
Clic Next and you bpf designer will show-up :



Click the Loan New stage and the stage property page, change the display name as Loan basic info and click on Apply.

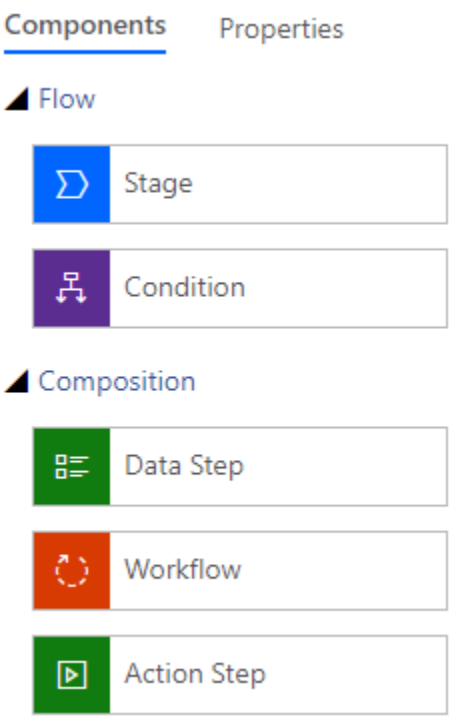


In the stage, click the Data Step, name the step Name “account”, select the Account data field and click apply.

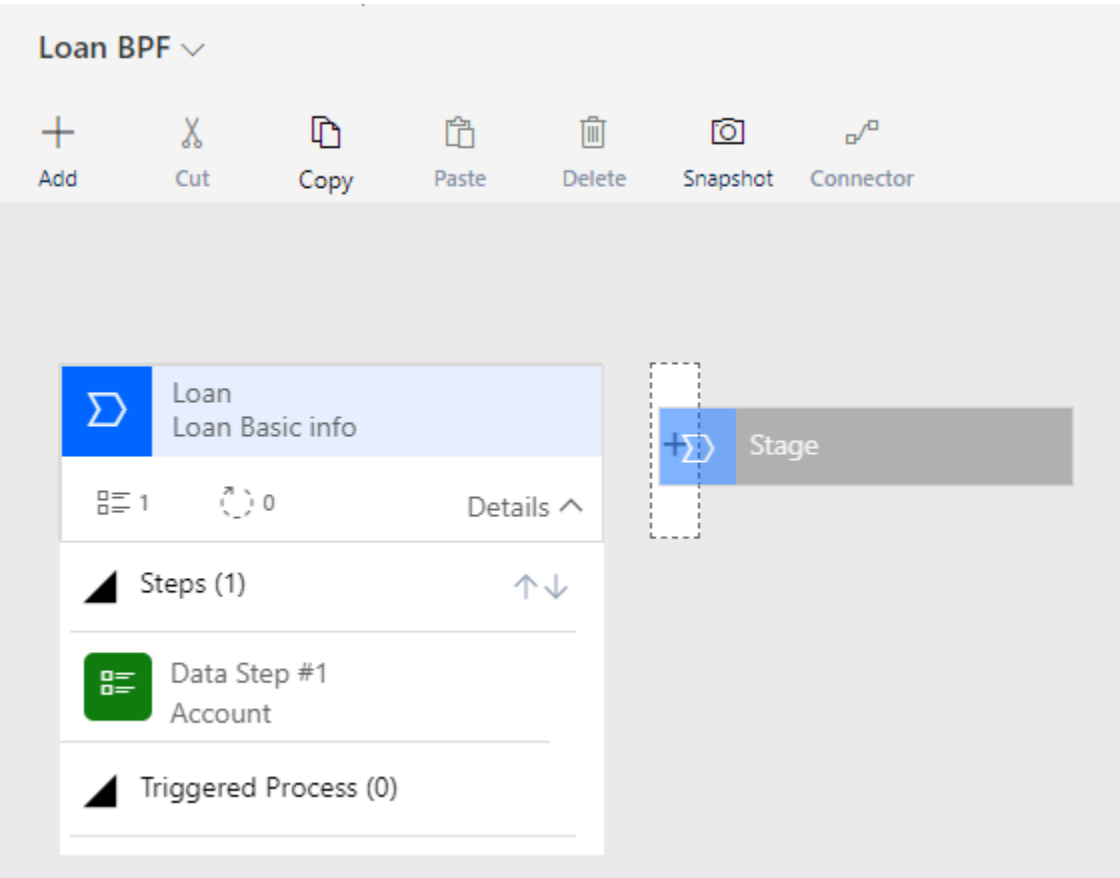


Click Apply.

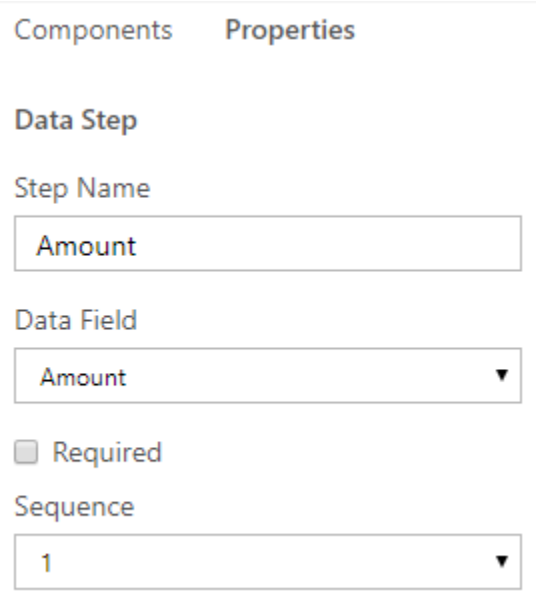
Click Components to add a new Stage



Click Stage and drag & drop a stage next to the existing stage:



Like you dit before, edit the data step of the new stage and link it to the Amount Data Field :



Click Apply.

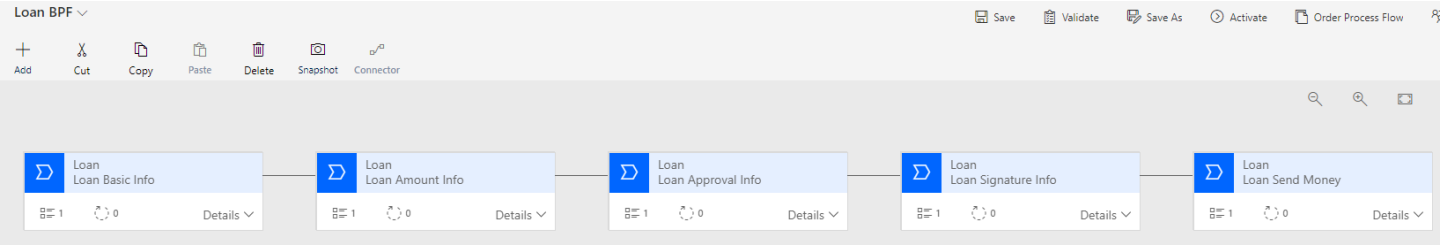
Name the stage Loan Amount info.

Add a new Stage and associate the data set with the field Approval status. Name the stage Loan Approval info

Add a new stage and associate the data set with the field signed. Name the Stage Loan Signature info.

Add a new stage and associate the data set with the field Send Money Name the Stage Loan Sign Money info.

The flow should look like this:

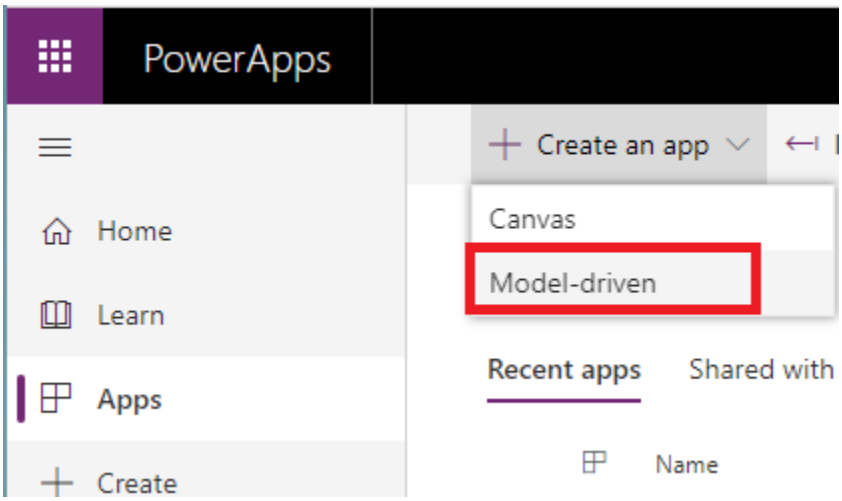


Click on Validate.

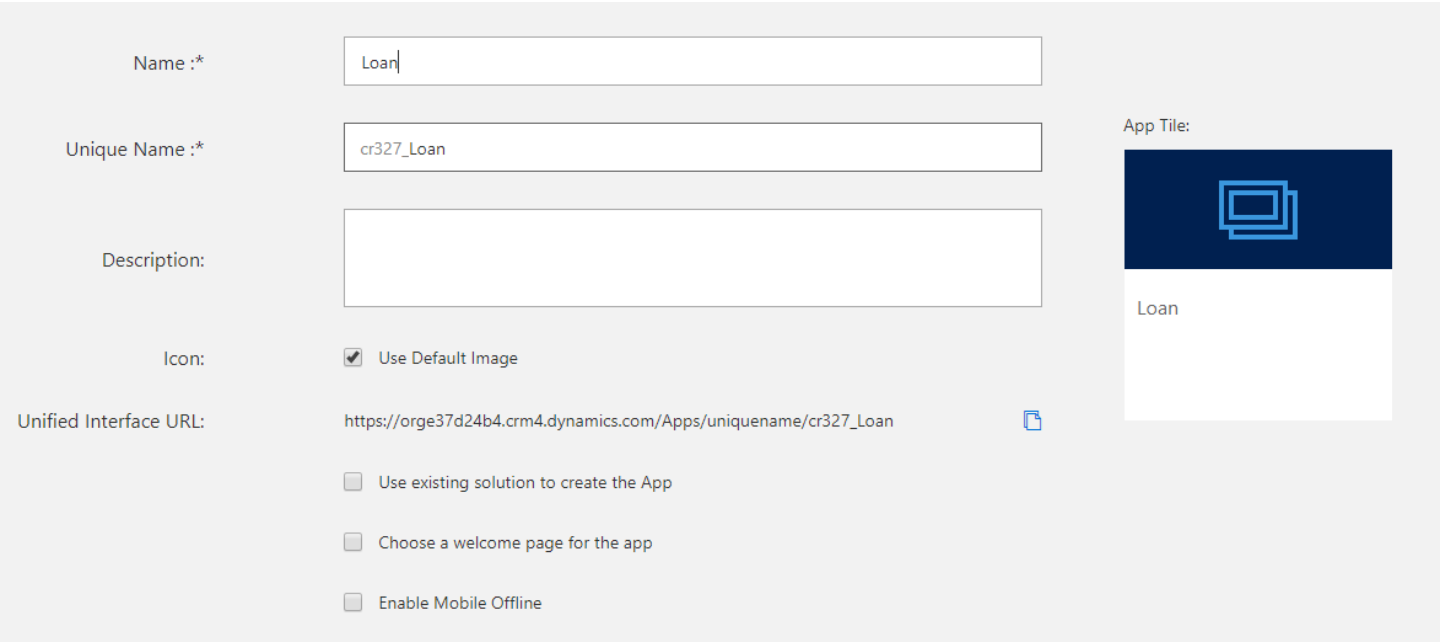
Click on Activate and confirm activation.

Now in order to use the Business Process flow we have to create a Model driven Apps associated with our loan Entity.

Go to the PowerApps portal and create a new Model Driven Apps:

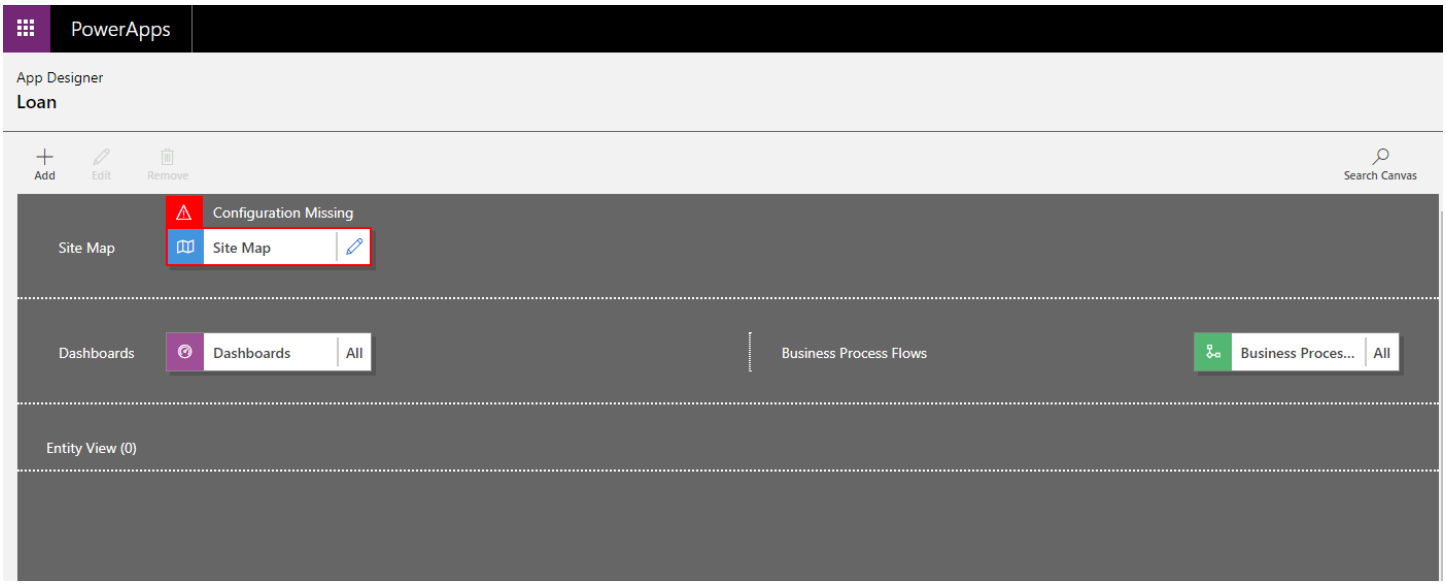


Name it Loan :



Click Done.

The App Designer will show-up :



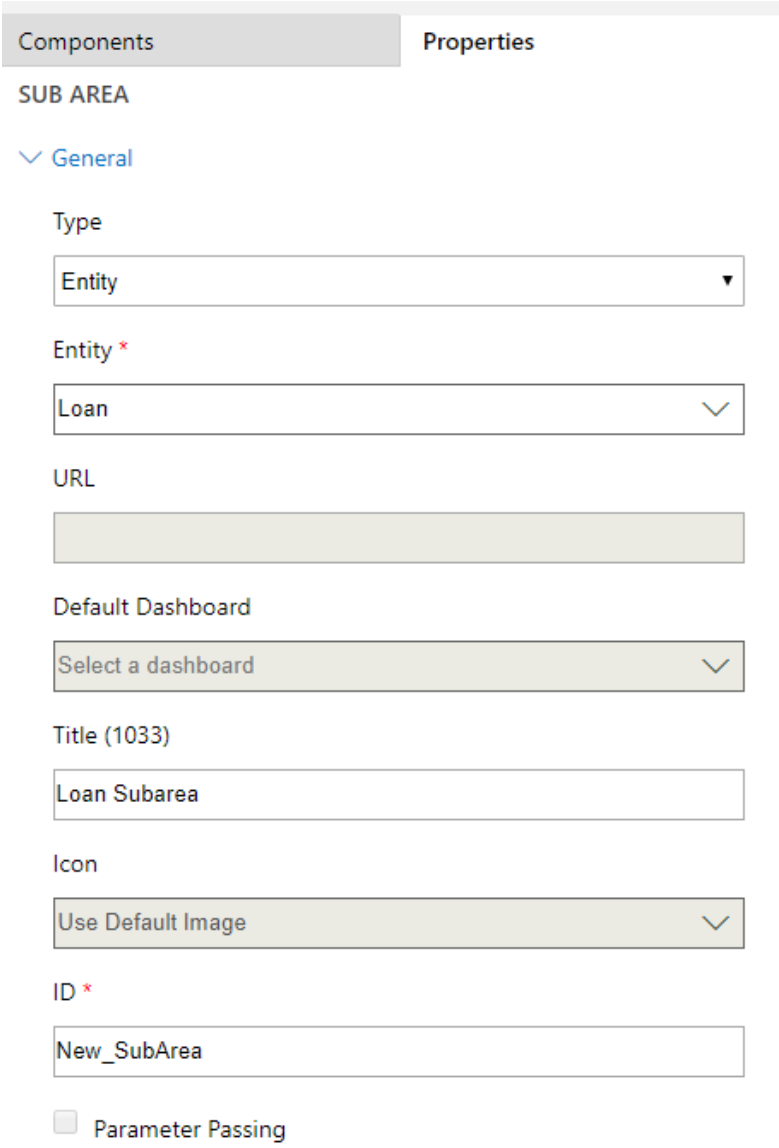
Edit the Site Map component.

Rename the New Area to Loan Area.

Rename the New Group to Loan Group.

Rename the Sub Area to Loan sub area.

Associate the Loan Subarea with the entity Loan



Click Save and Close, you will be redirected back to the App Designer screen.

Click Save

Click Publish

Click Play

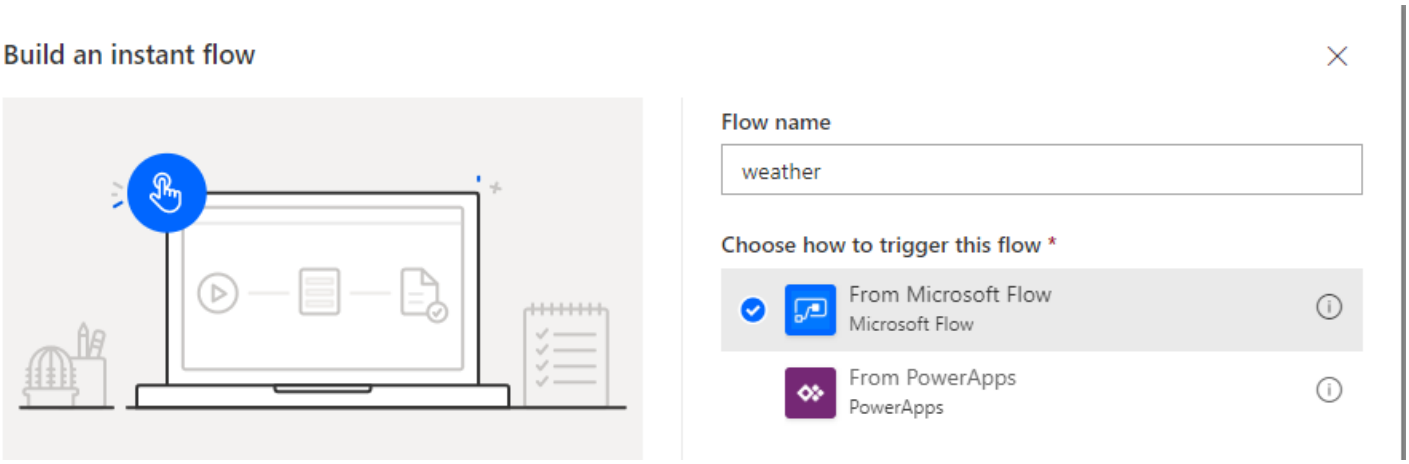
Lab 8. Call an external API from flow with the HTTP action

Learning objective: call an external REST api from flow

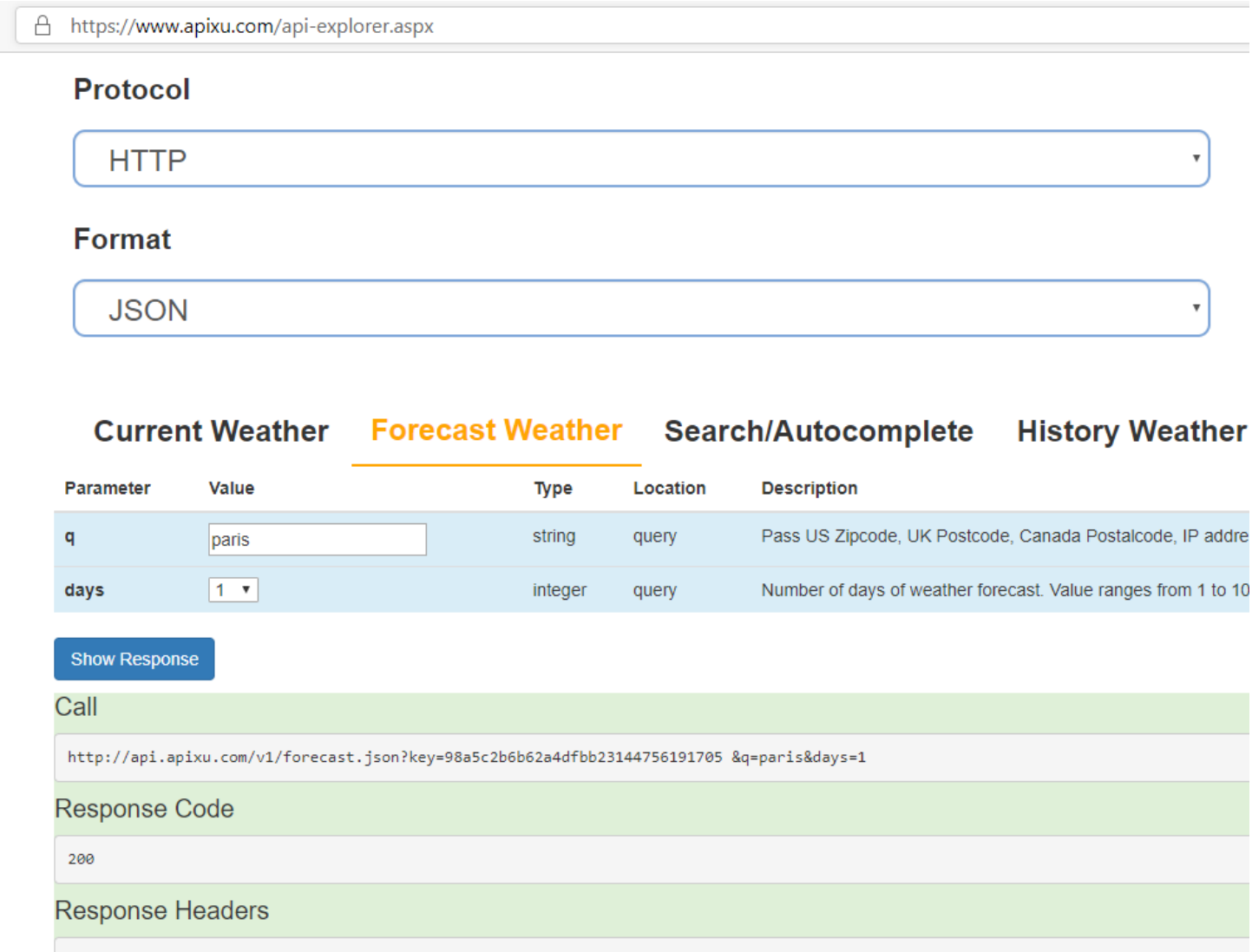
Duration: 15 minutes

Tasks:

1. Go to the site **apixu.com** and sign-up for a free account; you will get a key that you will use in Flow.
2. Create flow a started from a Button, name it **weather**:

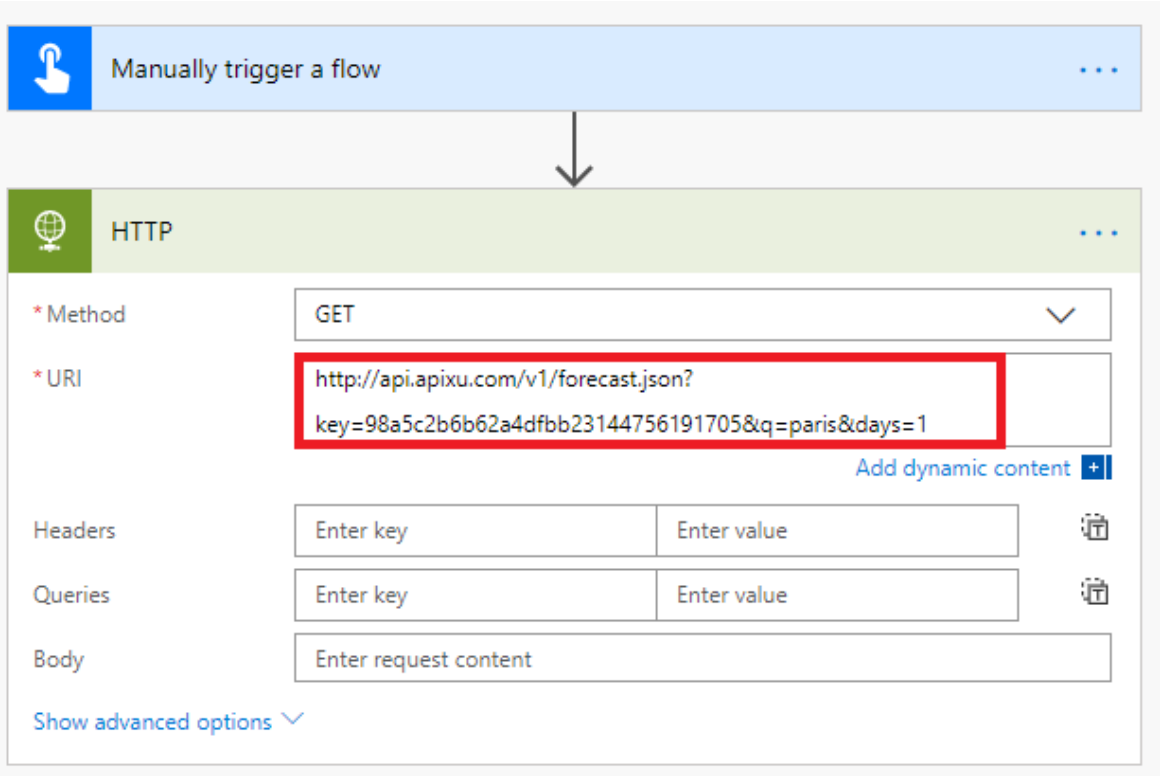


3. Add an HTTP action (this requires P1 licenses)
4. Go to the apixu explorer to test your query:



5. Copy and paste your query in the HTTP action; the value just after "key=" is your private key; the value after q= is the city.

```
http://api.apixu.com/v1/forecast.json?key=98a5c2b6b62a4dfbb23144756191705&q=paris&days=1
```



6. Save the flow and run it.
7. Check the output of the HTTP action; you should have something like this:

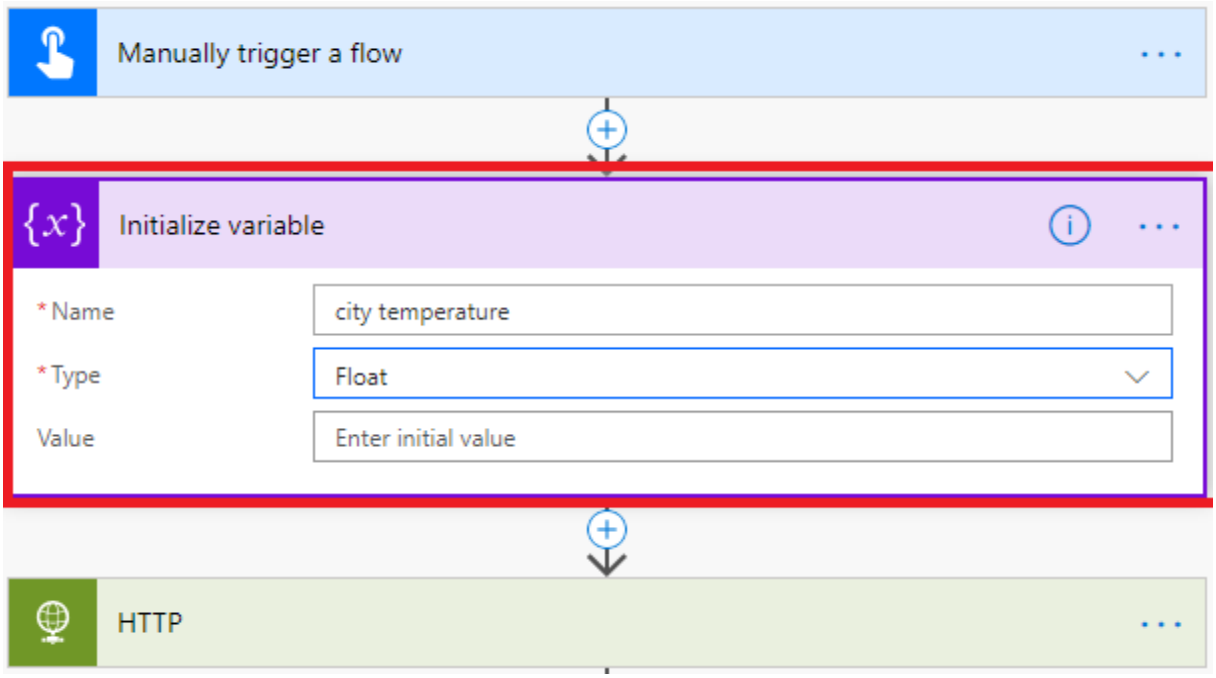

```
{
  "location": {
    "name": "Paris",
    "region": "Ile-de-France",
    "country": "France",
    "lat": 48.87,
    "lon": 2.33,
    "tz_id": "Europe/Paris",
    "localtime_epoch": 1558106441,
    "localtime": "2019-05-17 17:20"
  },
  "current": {
    "last_updated_epoch": 1558106108,
    "last_updated": "2019-05-17 17:15",
    "temp_c": 19,
    "temp_f": 66.2,
    "is_day": 1,
    "condition": {
      "text": "Sunny",
      "icon": "//cdn.apixu.com/weather/64x64/day/113.png",
      "code": 1000
    },
```

```
    "wind_mph": 6.9,
    "wind_kph": 11.2,
    "wind_degree": 40,
    "wind_dir": "NE",
    "pressure_mb": 1004,
    "pressure_in": 30.1,
    "precip_mm": 0.2,
    "precip_in": 0.01,
    "humidity": 49,
    "cloud": 0,
    "feelslike_c": 19,
    "feelslike_f": 66.2,
    "vis_km": 10,
    "vis_miles": 6,
    "uv": 5,
    "gust_mph": 8.1,
    "gust_kph": 13
  },
  "forecast": {
    "forecastday": [
      {
        "date": "2019-05-17",
        "date_epoch": 1558051200,
        "day": {
          "maxtemp_c": 19.7,
          "maxtemp_f": 67.5,
          "mintemp_c": 11.1,
          "mintemp_f": 52,
          "avgtemp_c": 15.2,
          "avgtemp_f": 59.4,
          "maxwind_mph": 7.2,
          "maxwind_kph": 11.5,
          "totalprecip_mm": 3.3,
          "totalprecip_in": 0.13,
          "avgvis_km": 13.9,
          "avgvis_miles": 8,
          "avghumidity": 64,
          "condition": {
            "text": "Moderate or heavy rain shower",
            "icon": "//cdn.apixu.com/weather/64x64/day/356.png",
            "code": 1243
          },
          "uv": 5.6
        },
        "astro": {
          "sunrise": "06:07 AM",
          "sunset": "09:28 PM",
```

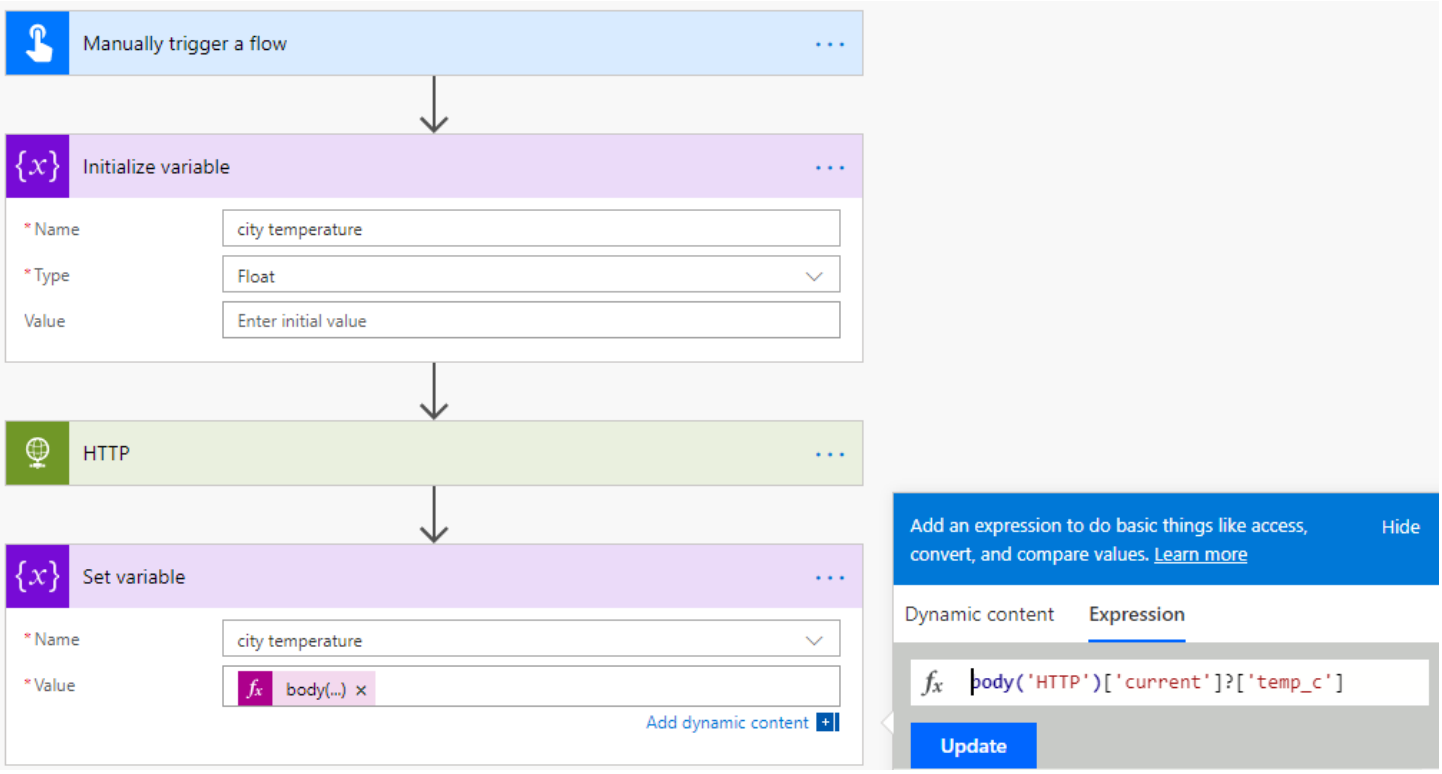


```
            "moonrise": "07:46 PM",
            "moonset": "05:47 AM"
          }
        }
      ]
    }
  }
```


8. Now we will store the current temperature (Celcius) in a variable. Create a string variable named **city temperature**:



9. Add a **Set variable** action that will grab the temperature:

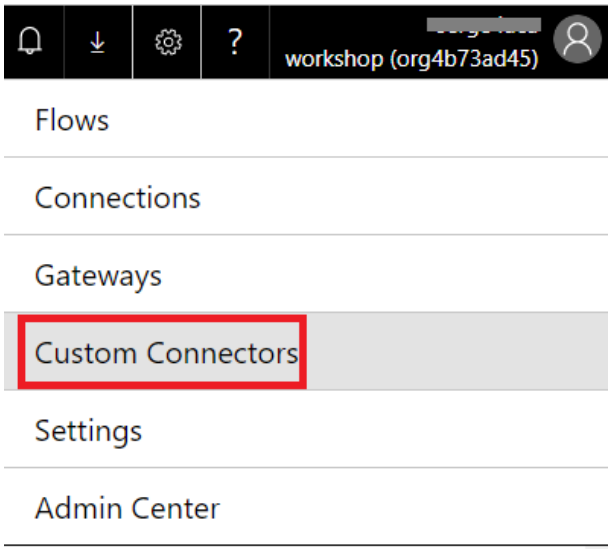


10. Run the flow and check the variable content.

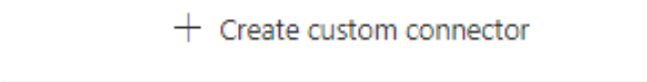
Lab 9. Create a Flow/PowerApps Custom Connector

Tasks:

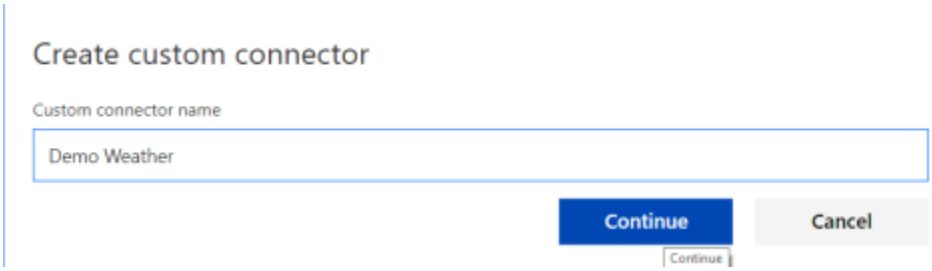
1. Go to the **Connector** menu and select **Custom Connectors**:



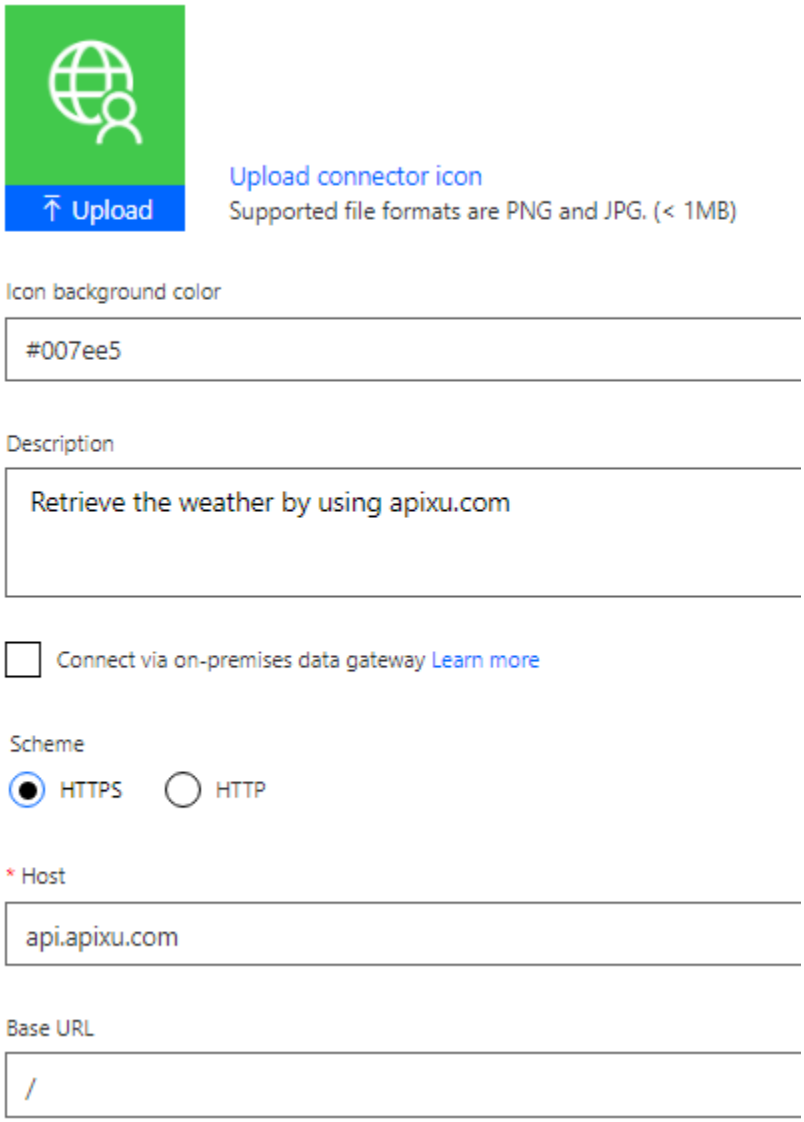
2. Click on **Create custom connector**:



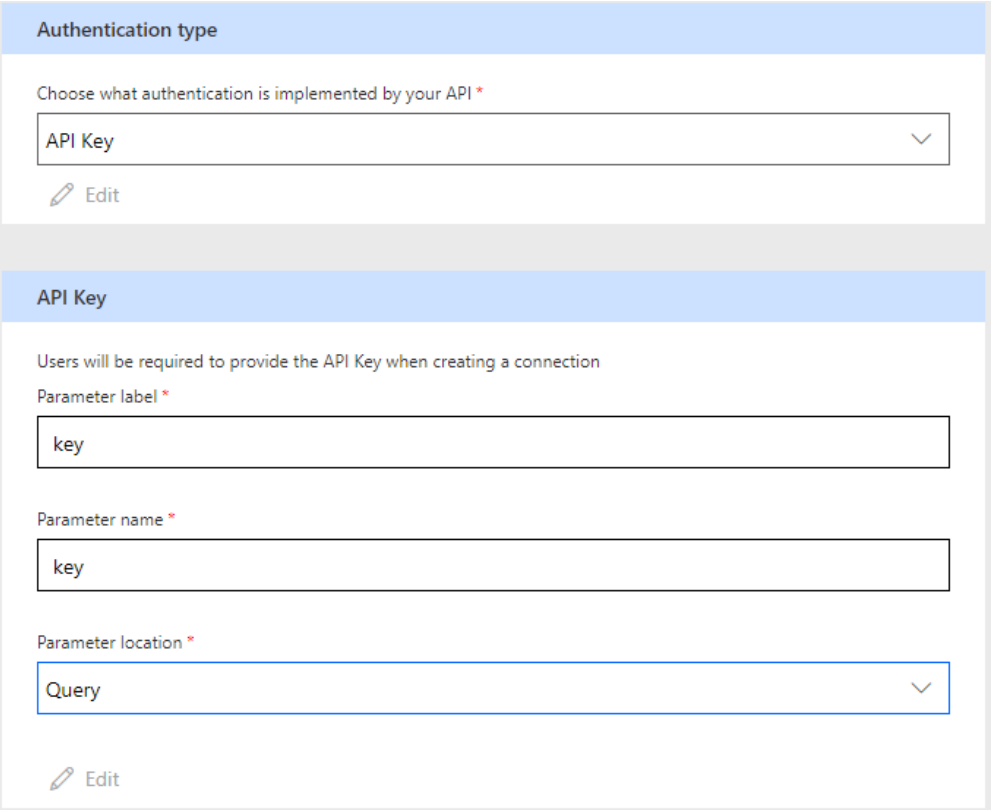
3. Select **Create from blank.**
4. Name the connector "Demo Weather" and click Continue



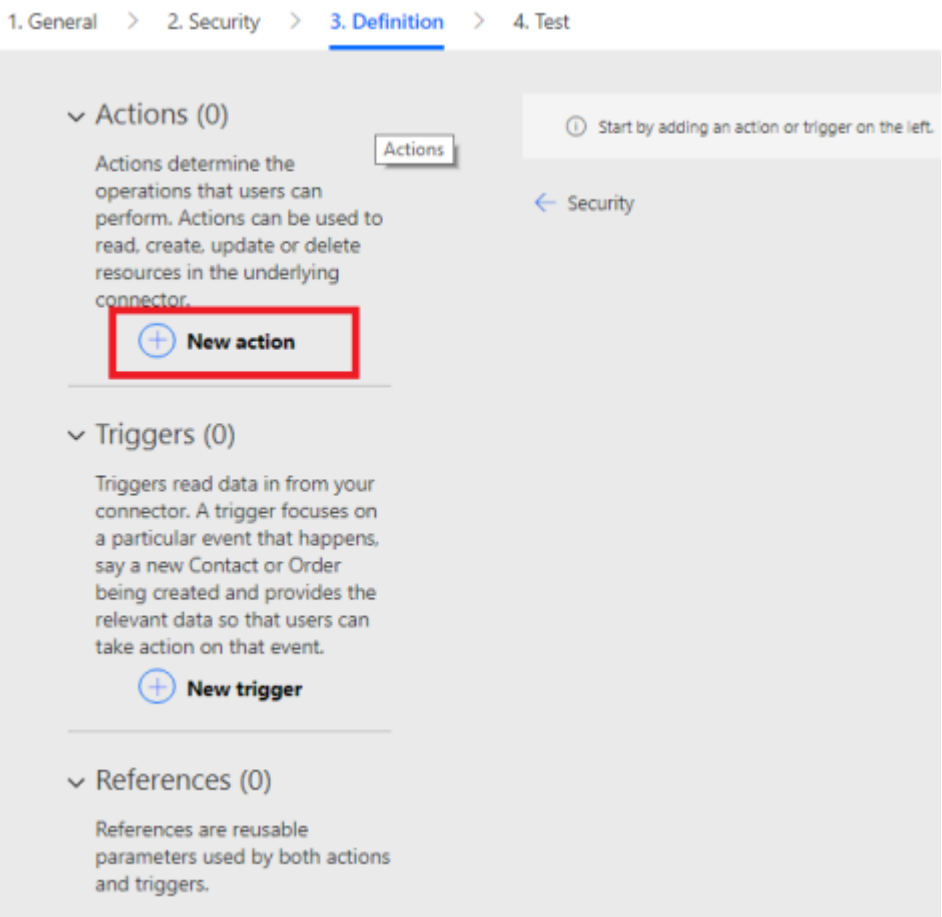
5. Click continue and in the next window, provide the host (api.apixu.com) and a short description of what your connector does:



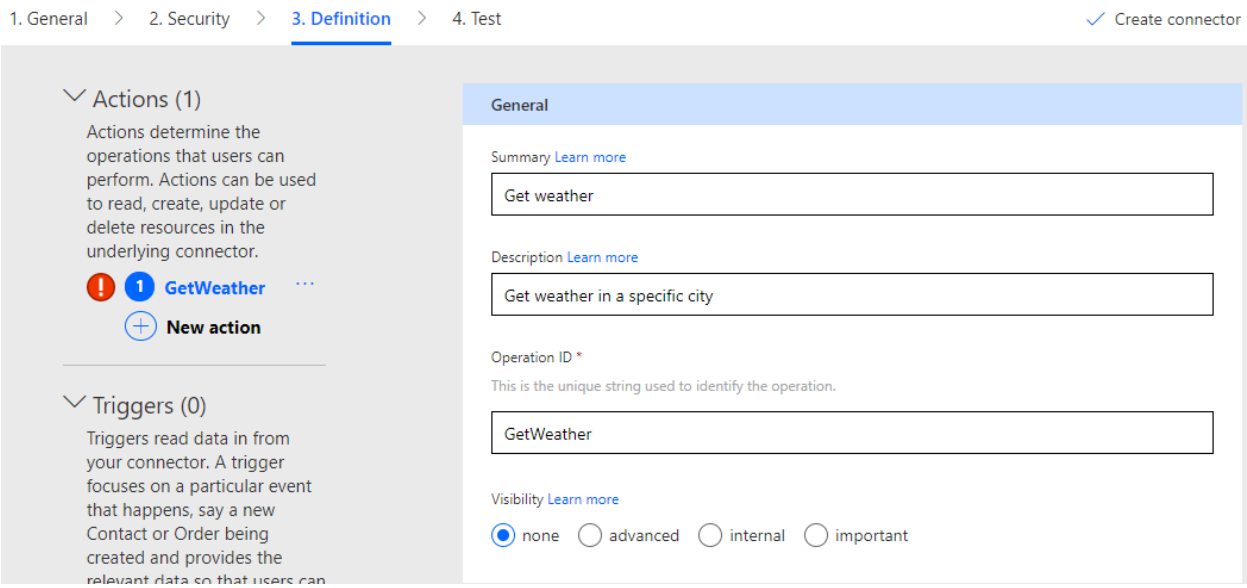
6. Click **Security** to move to the next screen.
7. The authentication type should be API key.
8. Since we want the Key parameter to be provided in the query string, create an api key with key as Parameter label and Parameter name; switch the parameter location to Query as illustrated in the following picture:



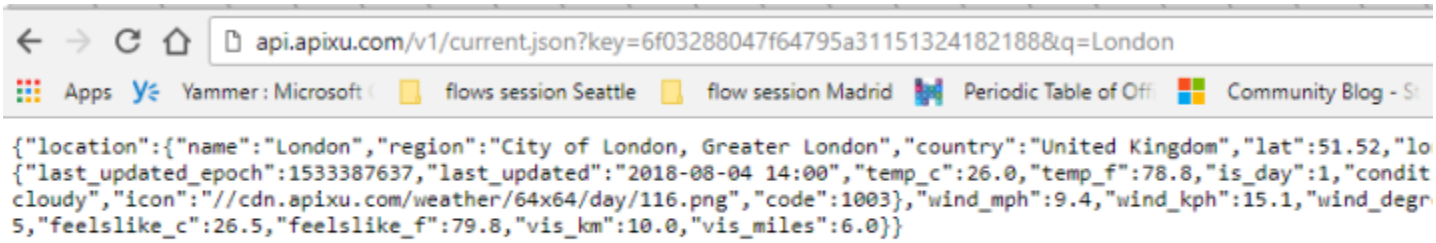
9. Click **Definition**:



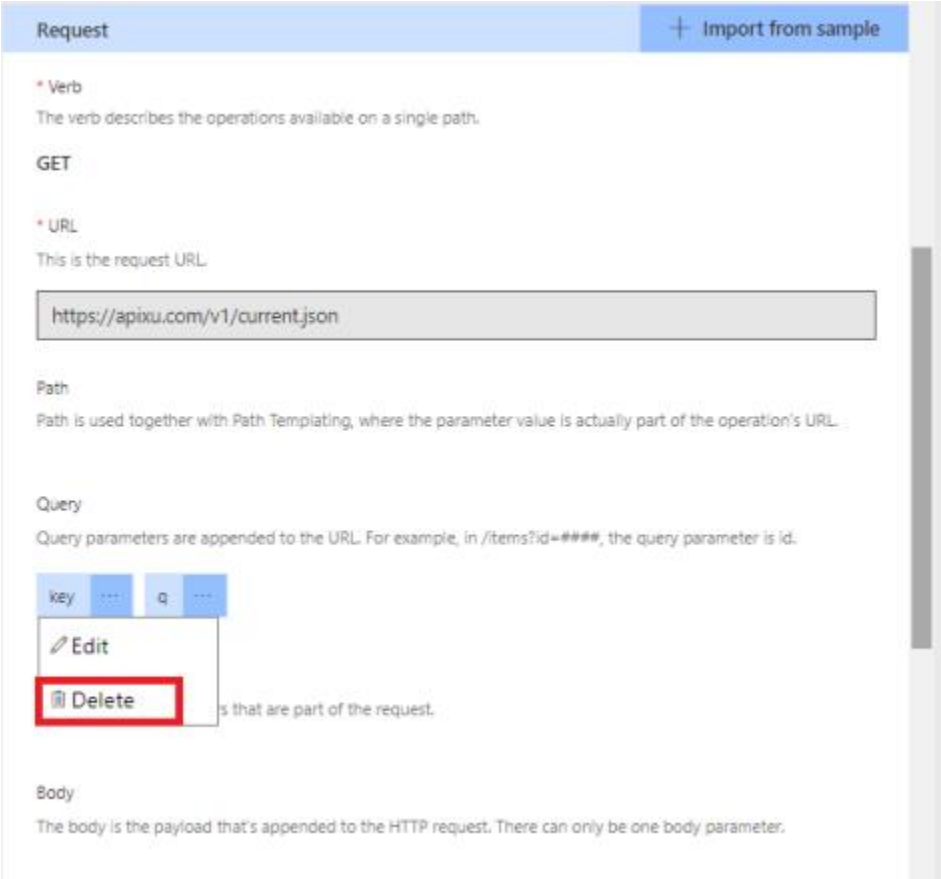
10. Click on **New action**:
11. Define the Action as follow:



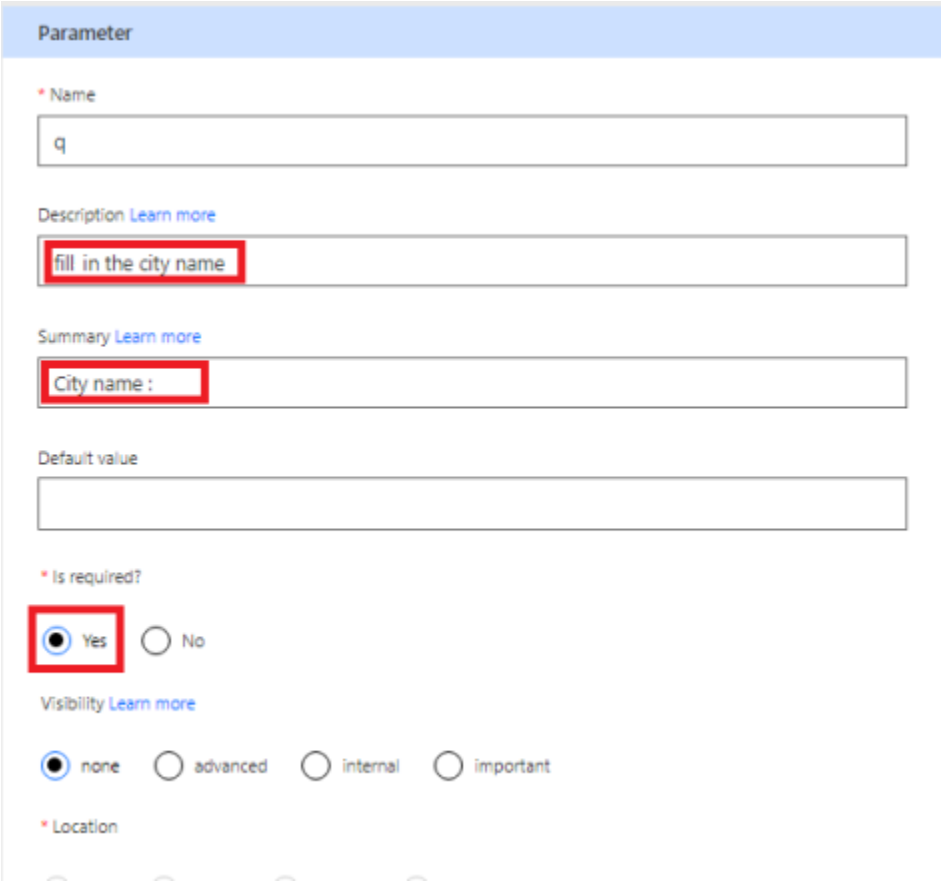
12. In a new browser tab, type a weather request to make sure it works fine and also to generate sample data that we will reuse in our connector definition (don't forget to pass your key as a parameter, as well as the city):



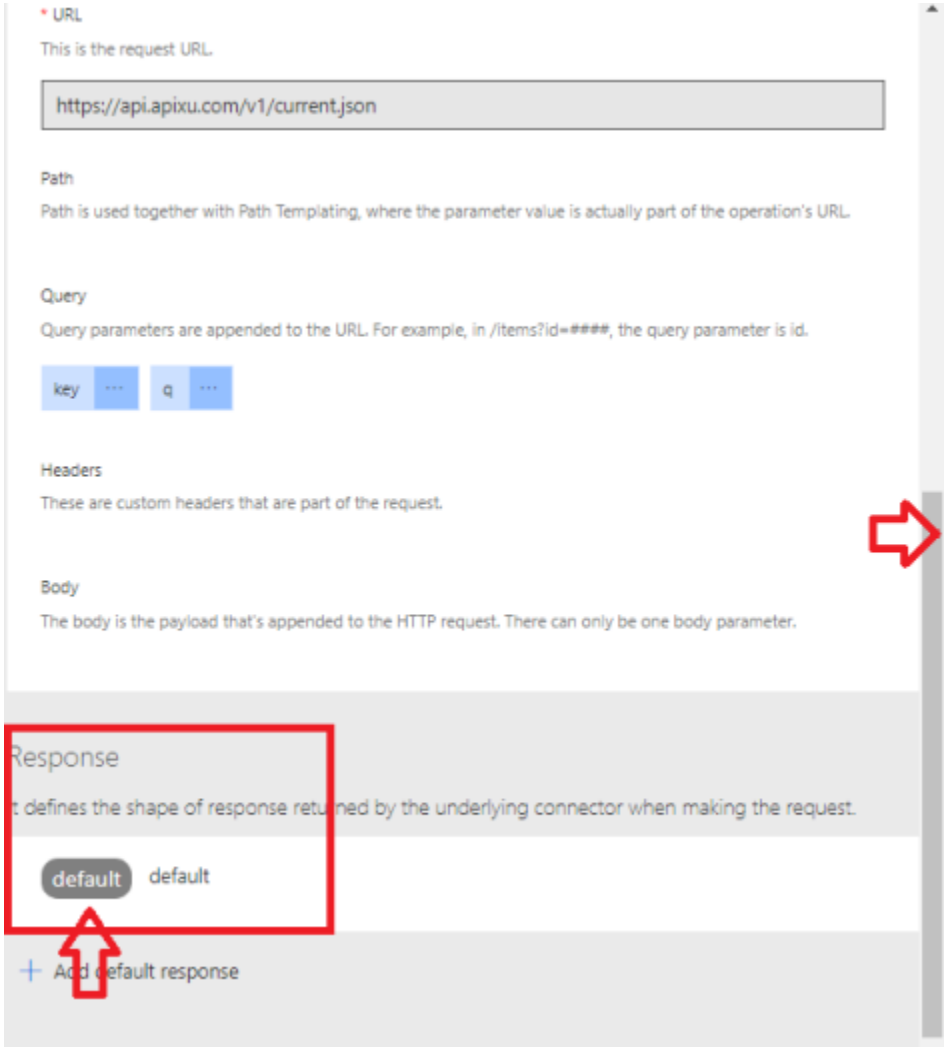
13. Keep this tab open and go back to the connector definition.
14. Down below in **Request**, click **Import from sample.**
15. Pass your querystring, set the verb to **Get** and click i**mport**:
16. 3 parameters will be visible in a Query : key, q and days. Remove days.
17. The Key will be registered in the connector, there is no need to pass the key for each query, so we can delete it:



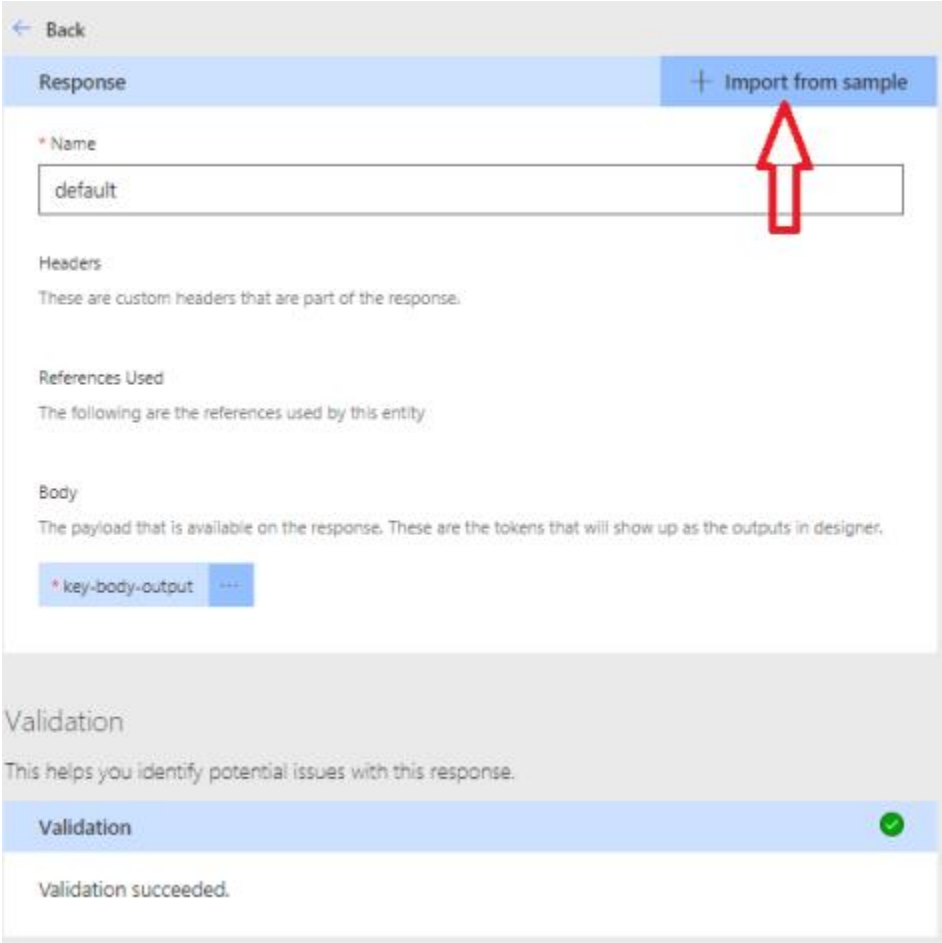
18. Edit the q parameter and fill in the **Description** as " Fill in the City Name" , **Summary** as "city : name and make the fields required :



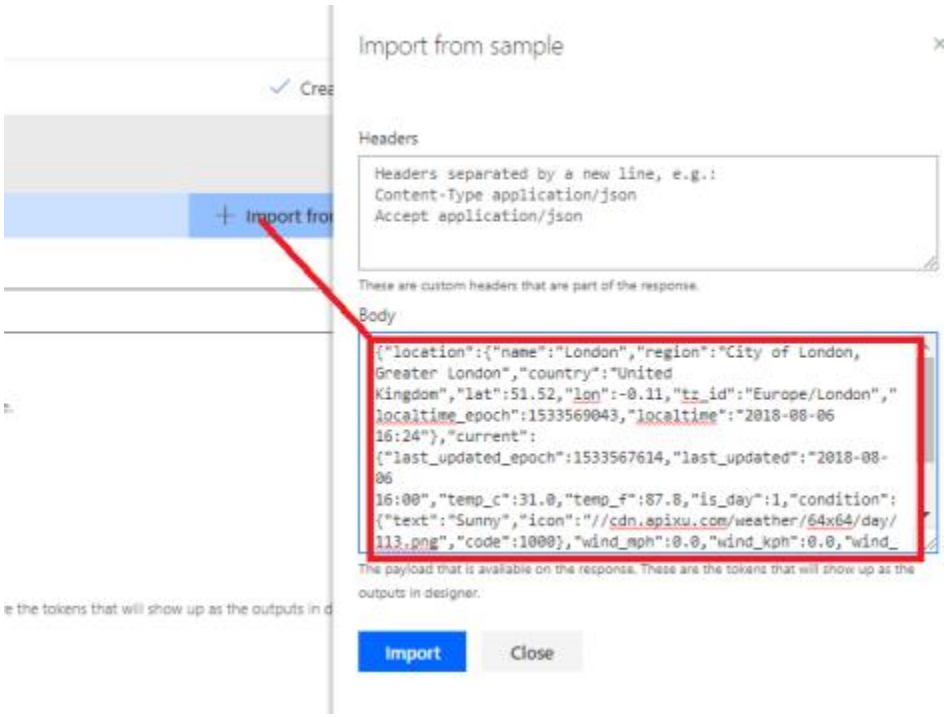
19. Click on **Back**, scroll to **Response**, click on **default** to import another sample ; the scroll bar is in the middle of the screen as illustrated in the picture:



20. In the next window, click on **Import from sample**:

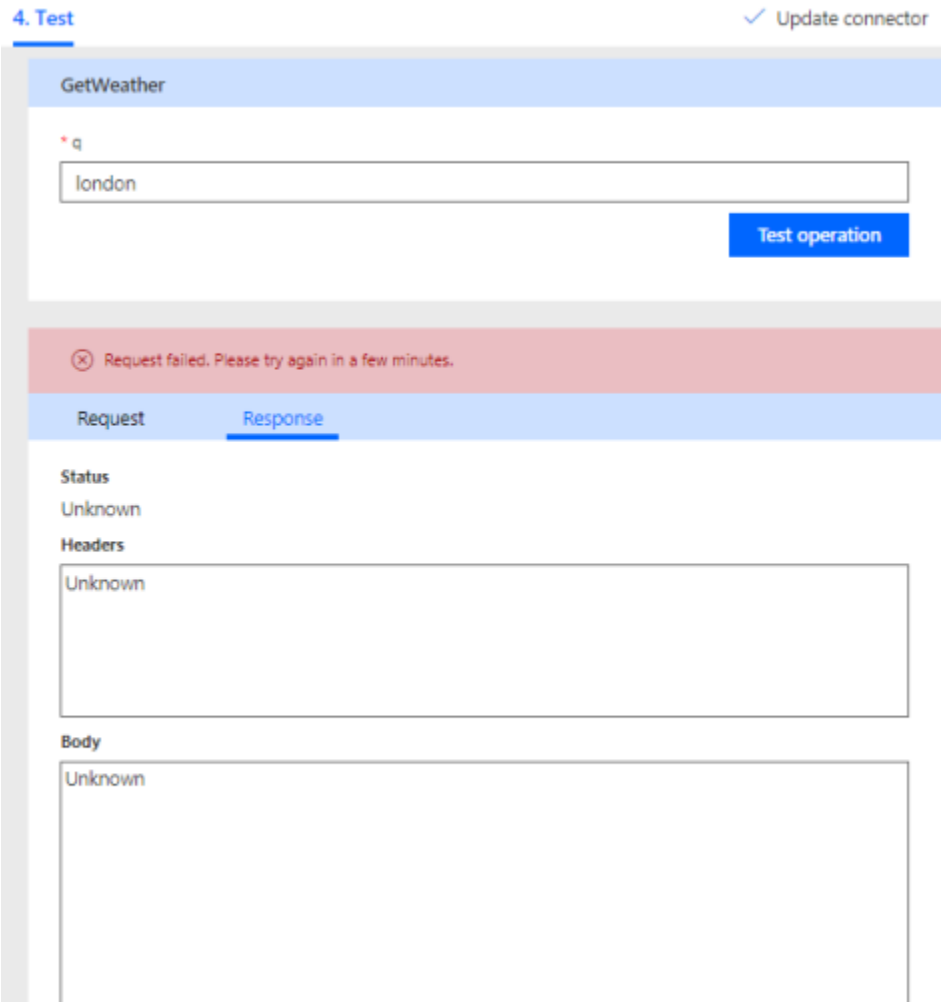


21. ...and in the next window, paste the JSON result grabbed from the browser:

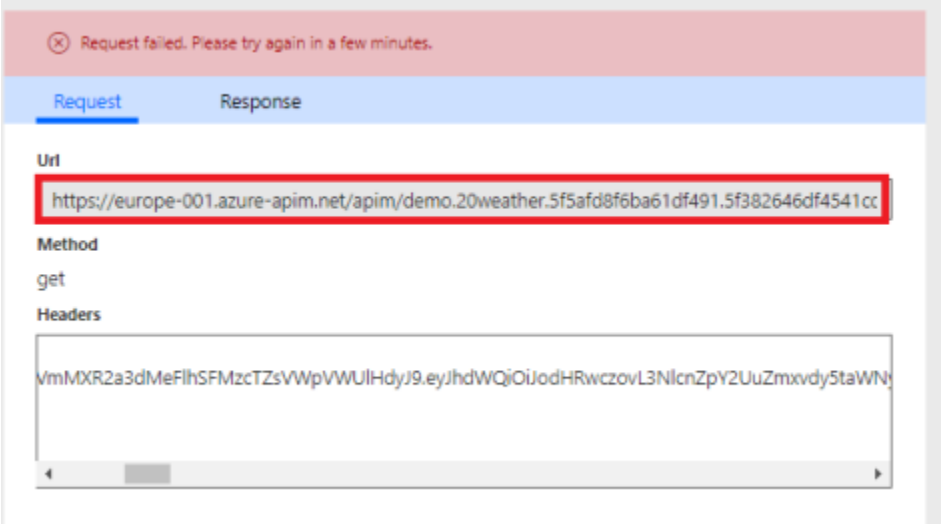


22. Click **Import**.
23. Click **Create Connector**.

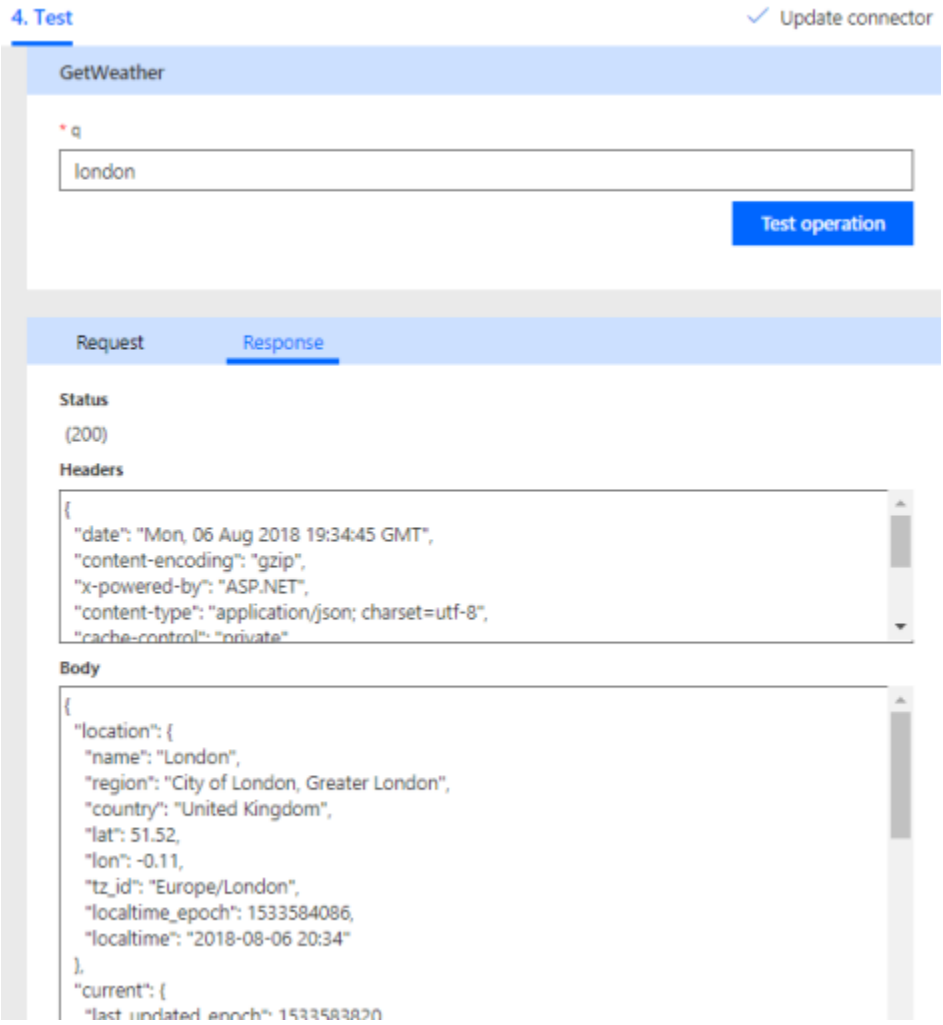
24. It can take a while before connector get created and deployed. Indeed, even though you specified the url http://api.apixu.com in order to call the REST, the connector will generate a public azure proxy that must be deployed and this can take several minutes.
25. So if you test it right away it might fail:



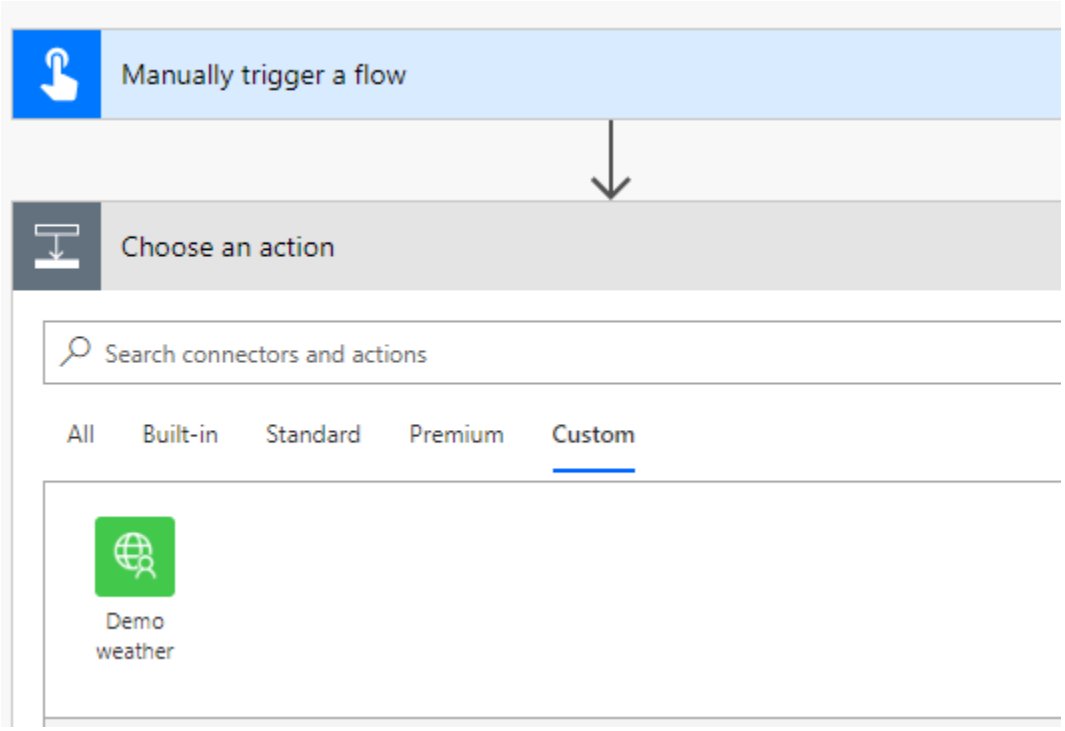
26. If it fails you will be able to see the Azure proxy by clicking in the (test) Request link :



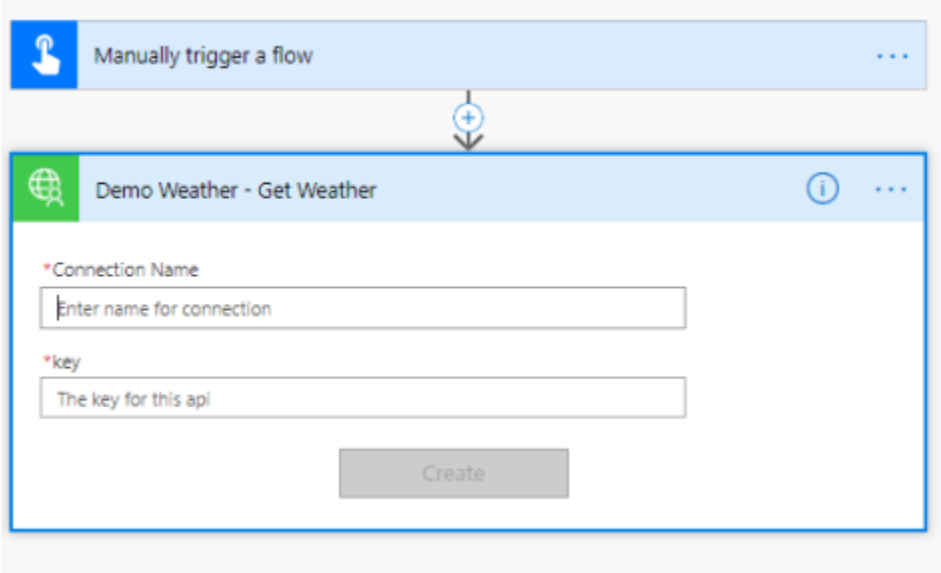
27. After a while, Test should be working



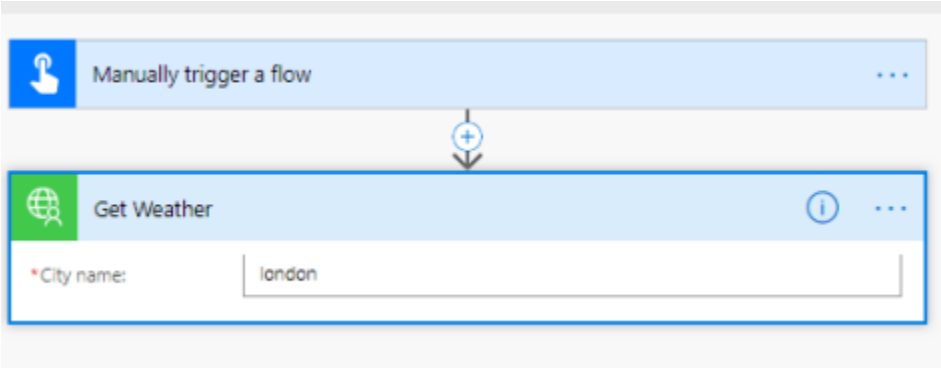
28. You can now create a new Empty flow started from a button.
29. Add an action from the Custom category; you should find you Demo Weather custom connector:

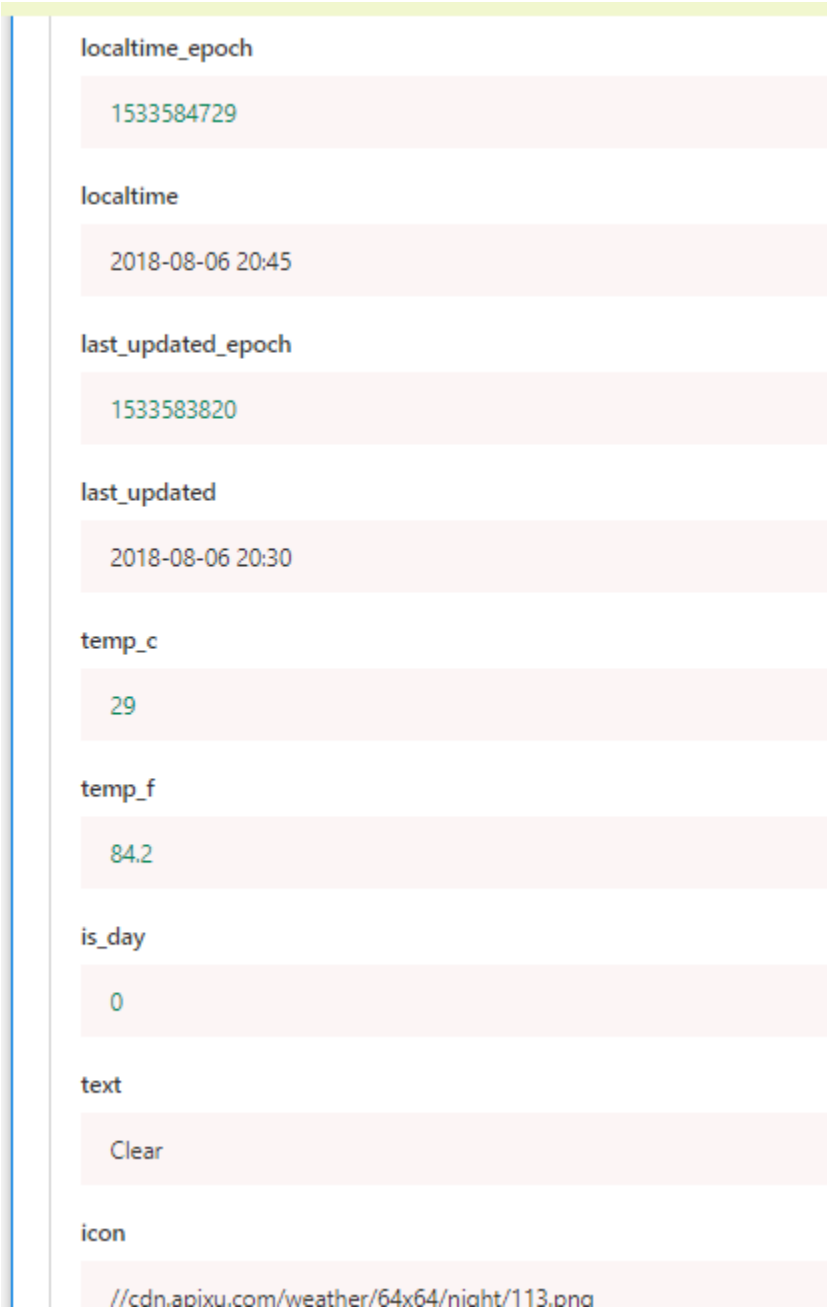


30. Select the **Get Weather** action
31. You will be prompted to create a new connection, that you will be able to reuse afterwards:

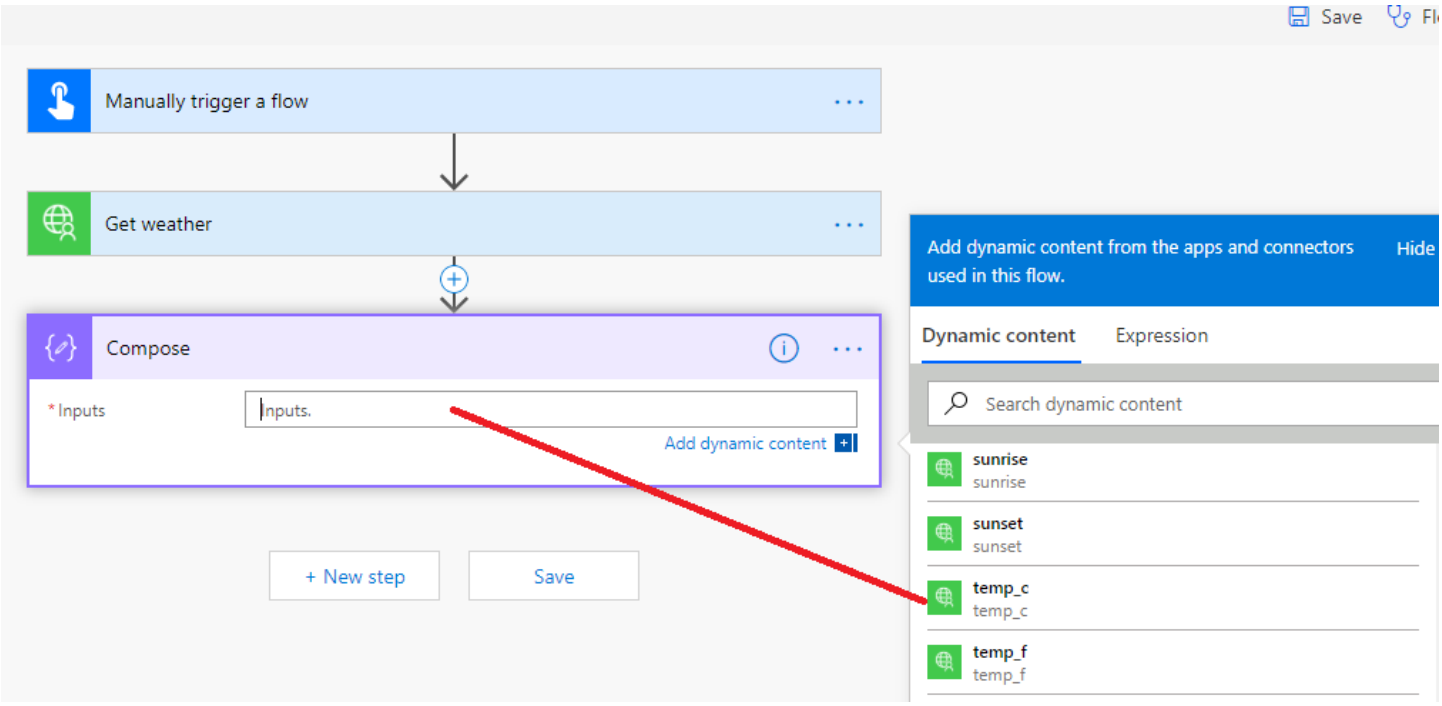


32. Provide a connection name, fill in the key and click Create.
33. Edit the action and hard code it 'London' as the city name.





34. Add an action like a variable or a **Compose** and store the **temp_c** value:



Lab 10. (Advanced) Implementing the flow controller pattern

Learning objective: Create state machines and have flows that can run beyond the 30 days limitation by using the Flow Controller pattern.

Duration: 30 minutes

Scenario: when a user submit a new expense in a SharePoint list, the expense will have to be approved by a first user (line manager); if this user doesn't react on time, another user (big boss) will react. If the big boss doesn't react on time, the system will ask the big boss to react again and again. The flow should work even if the whole process takes more than 30 days (current flow limitation).

Tasks :

This lab illustrates an implementation of the Controller pattern flow imagined by Serge Luca with the HTTP action.

This controller pattern flow provides more flexibility in the flow architecture design; we will apply this concept to have flows that can run beyond the 30 days limitation (flow run and approval), but also to implement state machines even though the current flow designer doesn't support state machines yet.

In this lab we will create 3 flows:

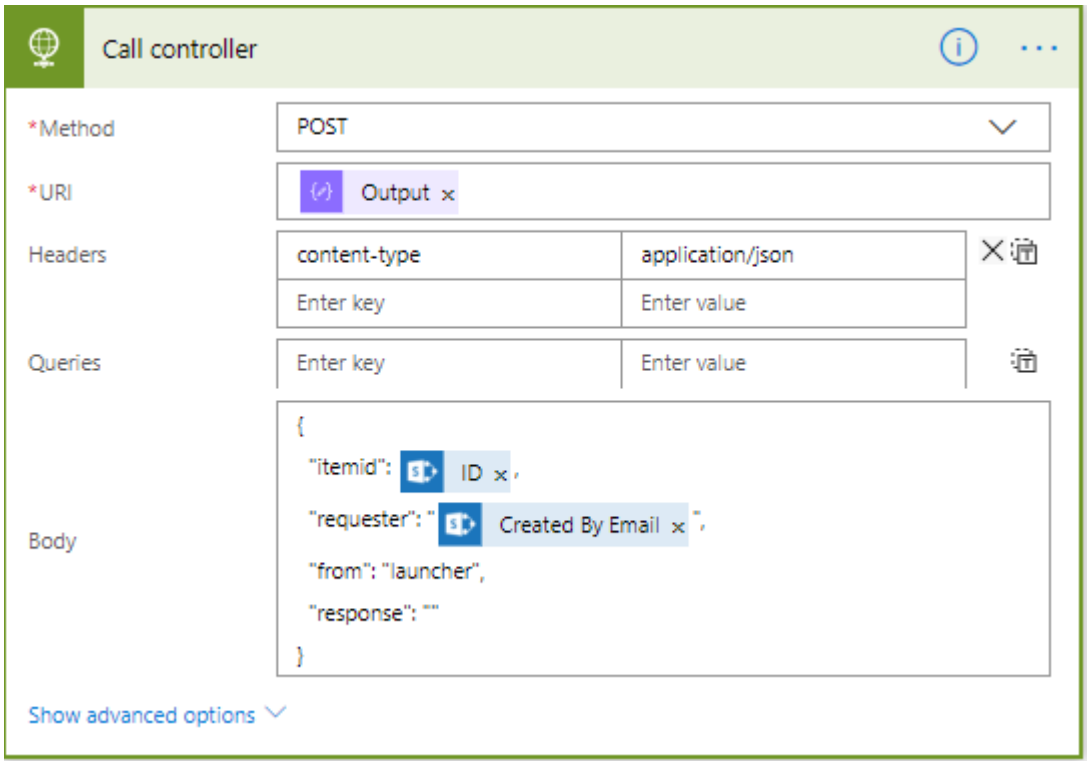
- The launcher flow
- The controller flow
- The generic approval flow

Setup

1. Create a SharePoint list named "Expenses" with 3 columns: the default column (title), a column **Amount** (type number) and column named **Status** (choice with the values In progress, Rejected, Accepted).

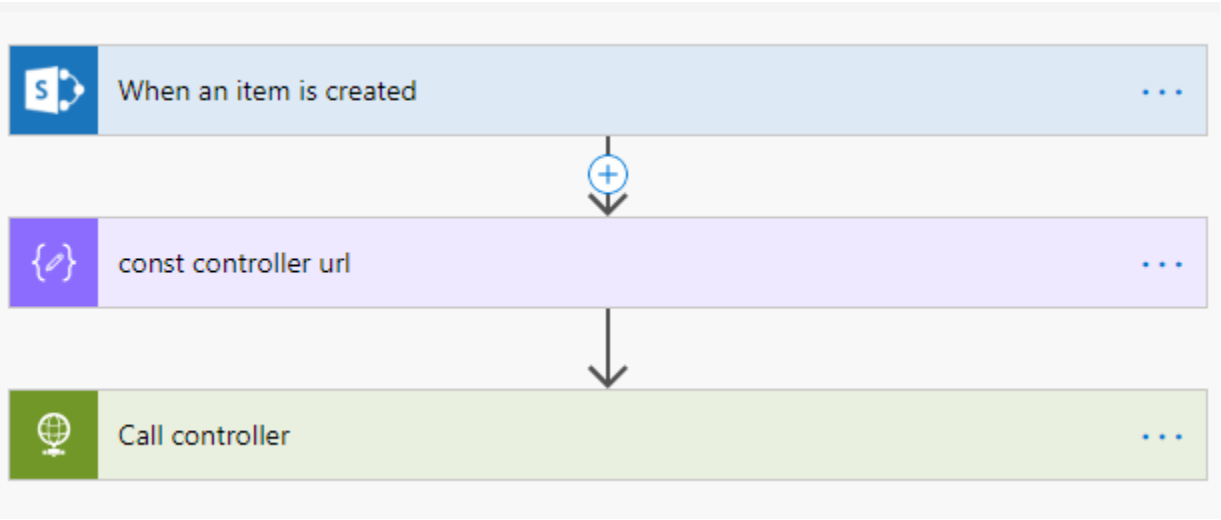
Launcher flow implementation

1. Create a Flow named **approval escalation launcher** that will start when a new expense is submitted in this list.
2. The trigger of this flow is **When an item is created** (SharePoint Connector)
3. Add a **compose** action, call it **const controller url** and set its value to "todo". We will update it afterwards.
4. Add an **HTTP action**, name it **Call Controller**
    a. Set the method POST
    b. Set the uri to the Output of **const controller url**
    c. In the Headers set **content-type** to **application/json**
    d. Set **Body** like as described below:



You can notice that ID is not surrounded by double quotes because it is an integer.

The flow looks like this:



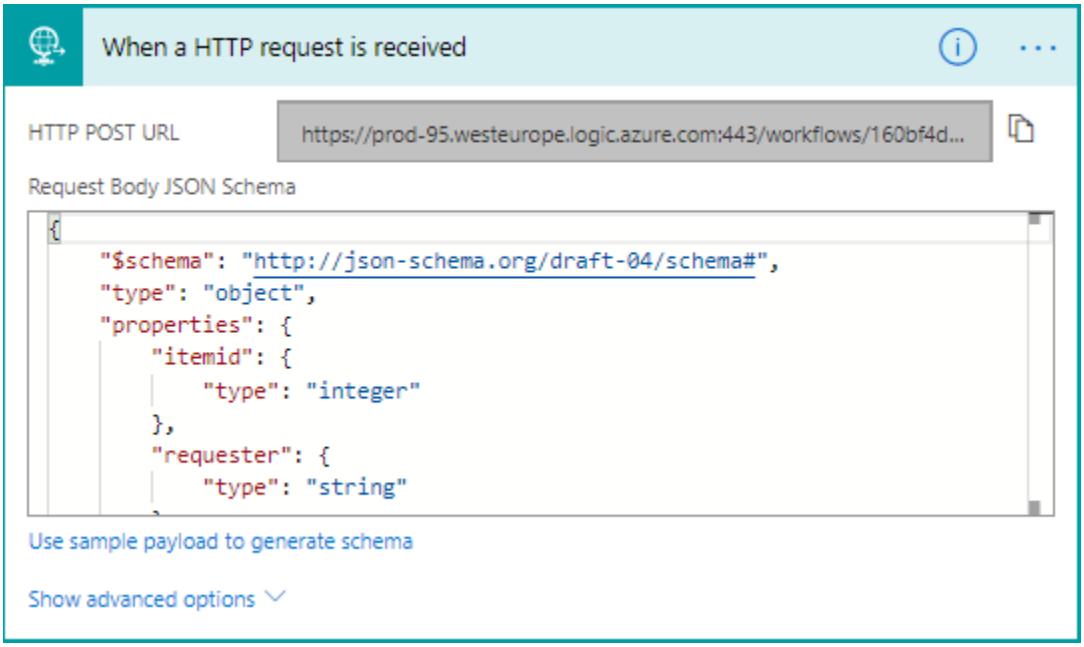
The flow controller will be implemented in the last place. Let's focus first on the Approval logic.

Approval generic Stage implementation

1. Create a new flow called **Approval generic stage** that starts with the trigger **When a HTTP request is received**.
2. In the Request body of the trigger, copy and paste the following JSON schema:

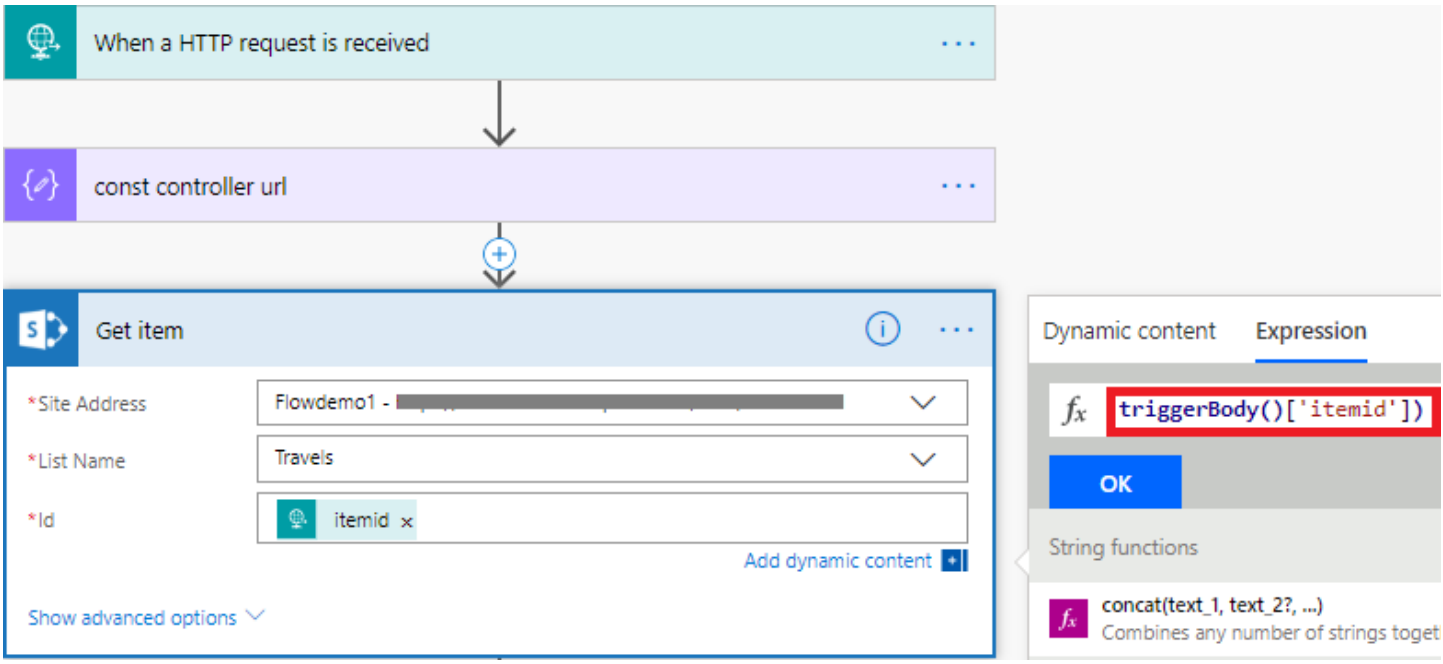
```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
```

```
    "type": "object",
    "properties": {
        "itemid": {
            "type": "integer"
        },
        "requester": {
            "type": "string"
        },
        "approver": {
            "type": "string"
        },
        "stagename": {
            "type": "string"
        }
    },
    "required": [
        "requester",
        "itemid",
        "approver",
        "stagename"
    ]
}
```

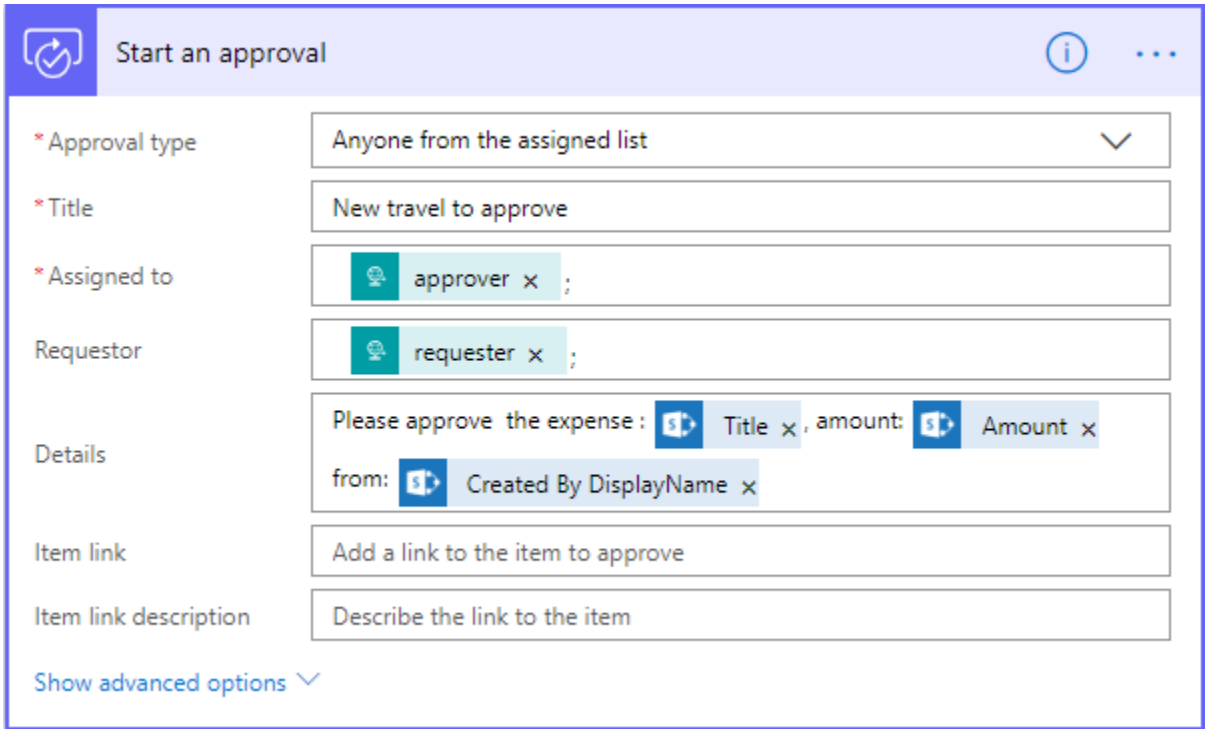


3. Add a Compose action named **const controller url**. We still have to generate the controller; in the meantime add a string "todo" in this action.

Add a SharePoint Get **item** action to retrieve you expense details; the id we must be grabbed from the trigger "itemid" parameter:



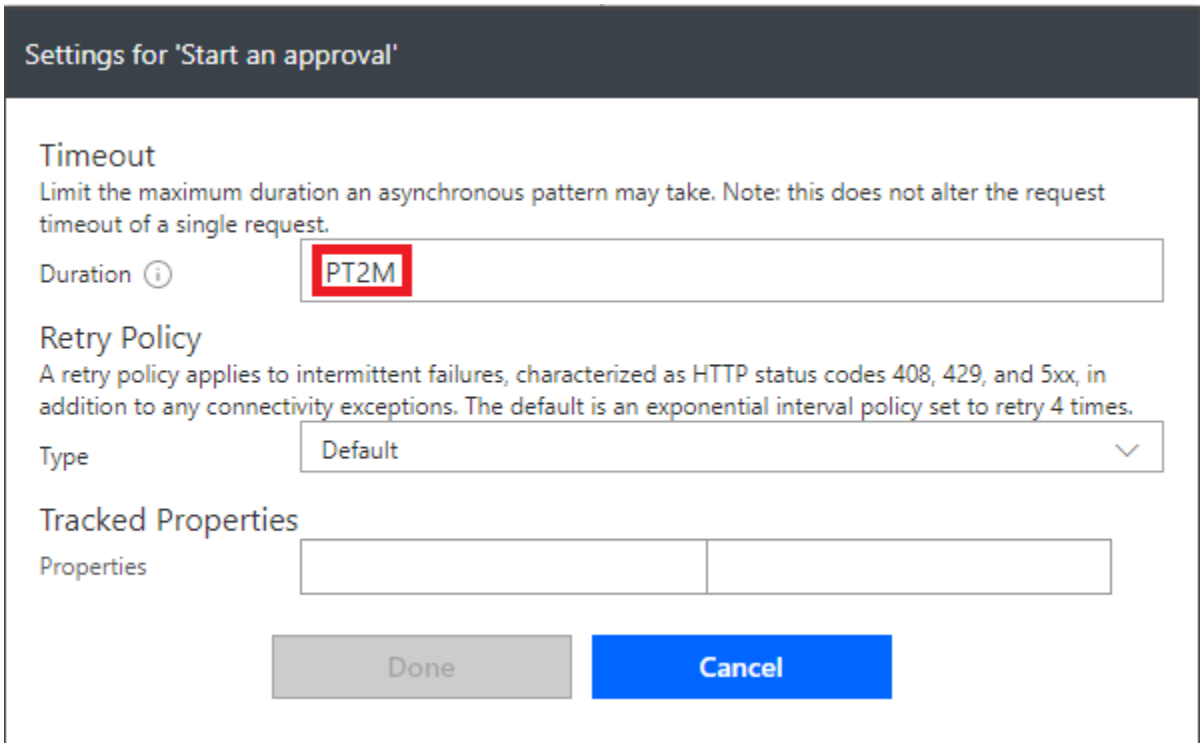
4. Add a **Start an approval action** and ask the line manager to approve:



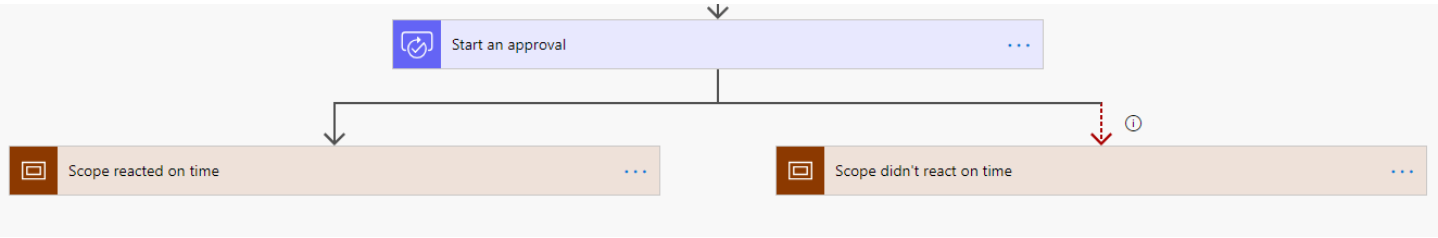
There are now 2 options: the approver reacts on time or he doesn't.

a) If he reacts on time, the process is completed;.
b) if he doesn't react the approval will be escalated to the big boss.

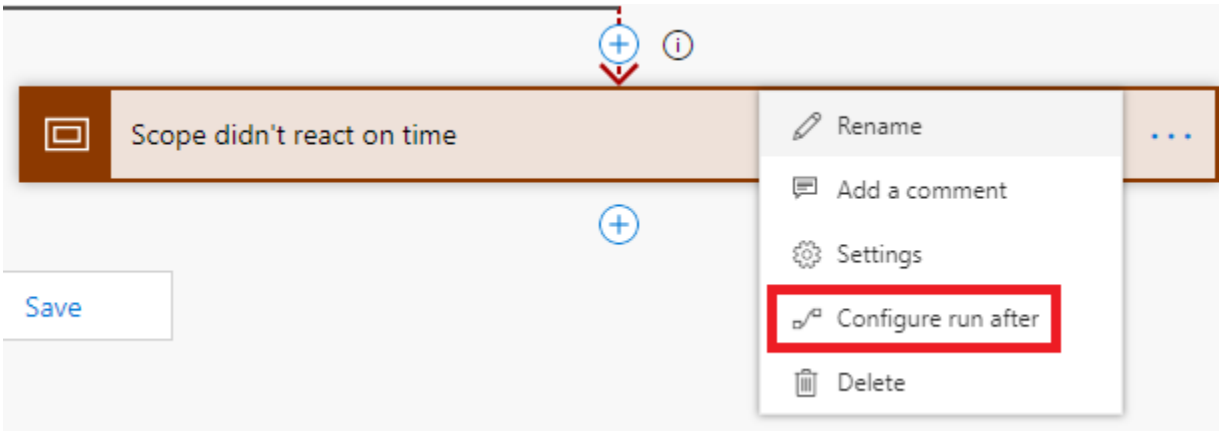
5. Select the **Start an approval** action setting and set the timeout to PT2M (2 minutes):

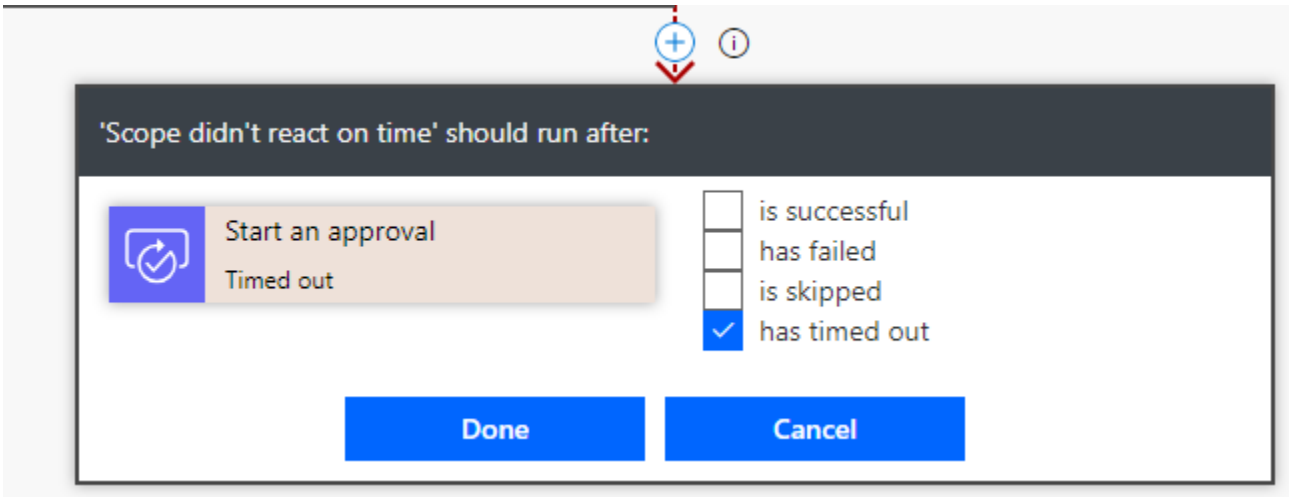


6. Add a parallel branch with 2 scopes and rename them accordingly:

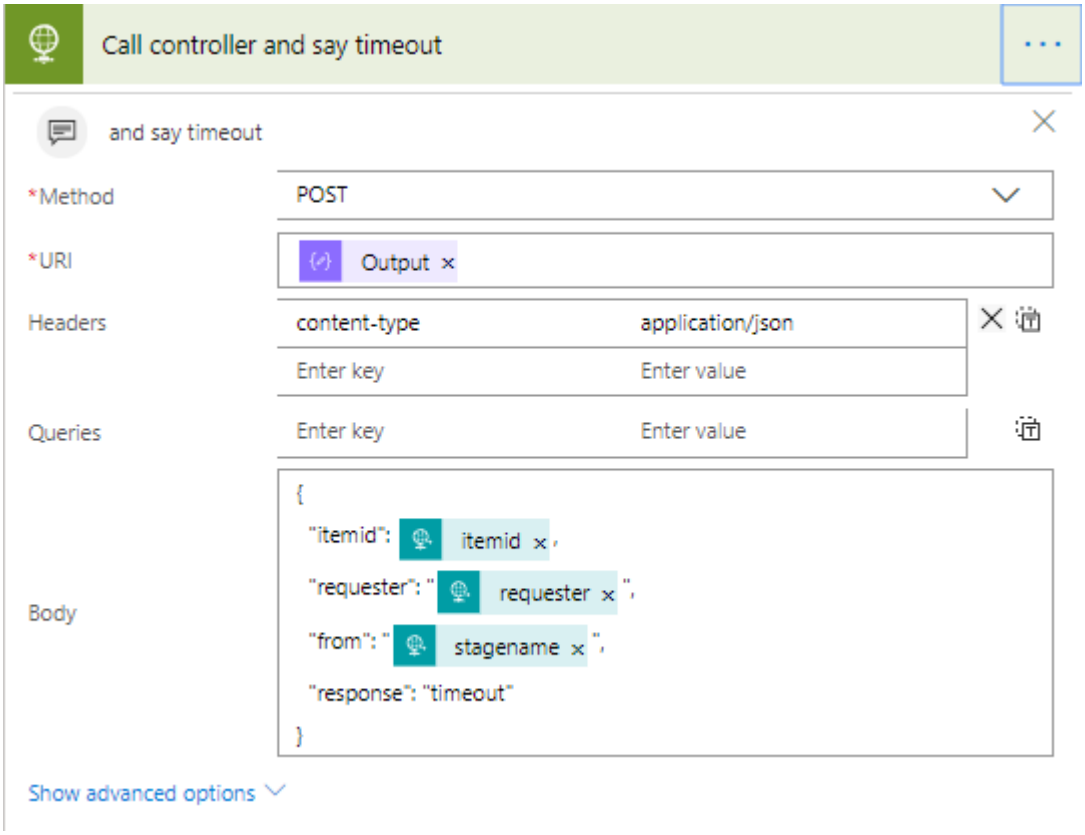


7. Select the "Scope didn't react on time" **Configure run after**:

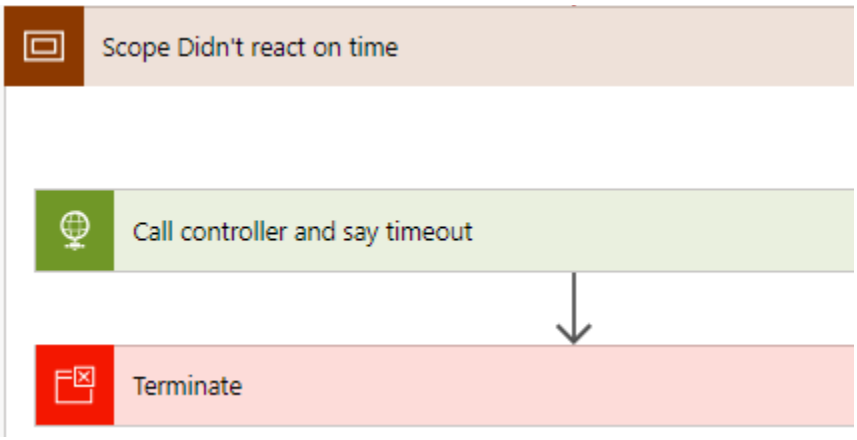




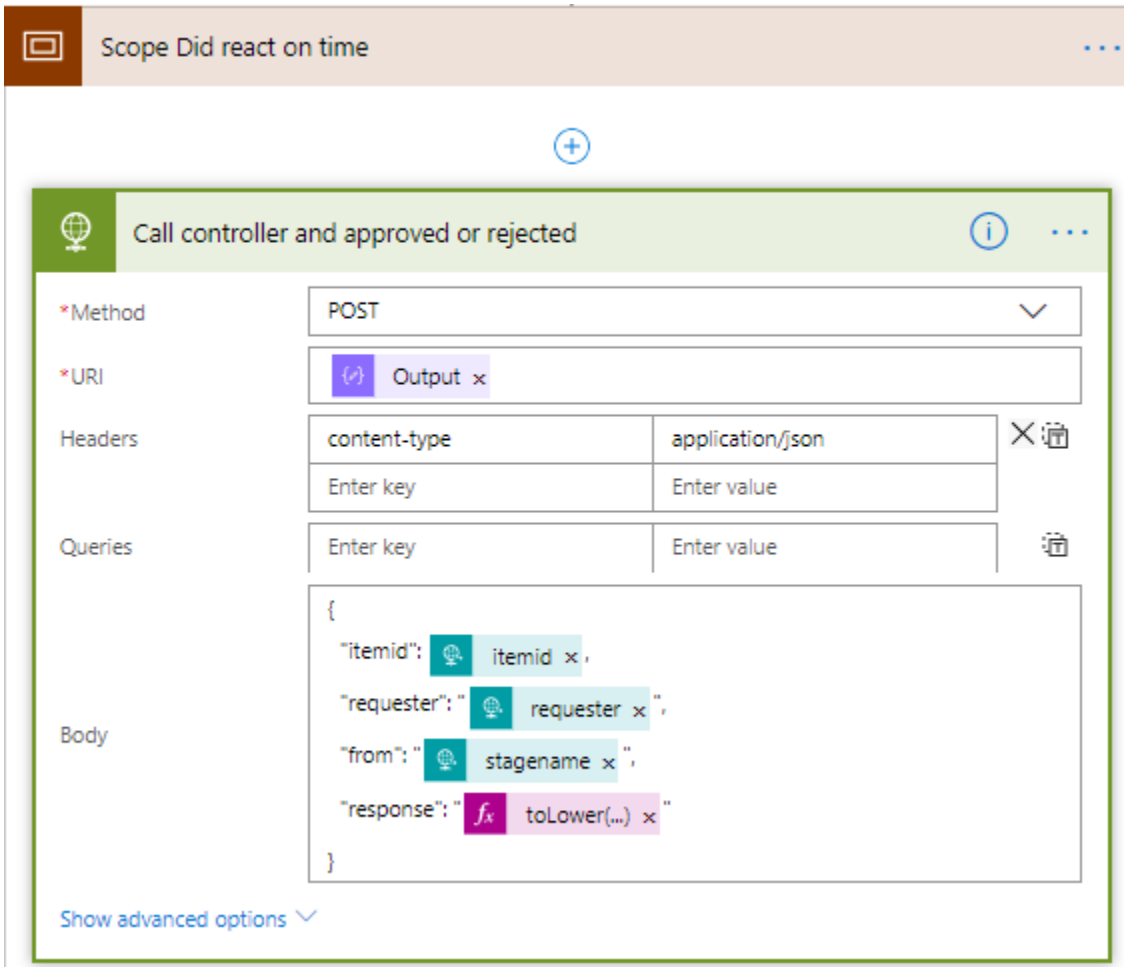
8. Check **has timed out**.
9. In the same scope add an **HTTP** action and set its settings as following:



10. Add a **Terminate** action with a Succeeded status:



11. In the other scope, add an **HTTP** action



The "response" parameter expression is the following:

```
toLower(body('Start_an_approval')?['response'])
```

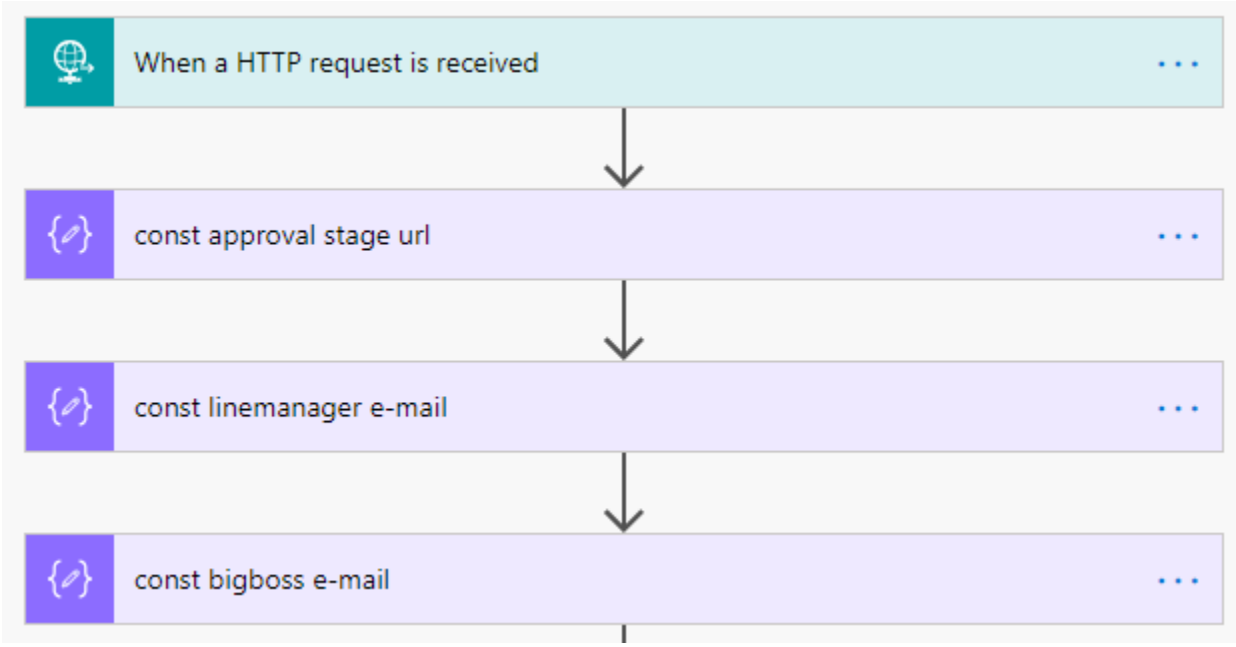
12. Save the flow.

Approval Controller flow implementation

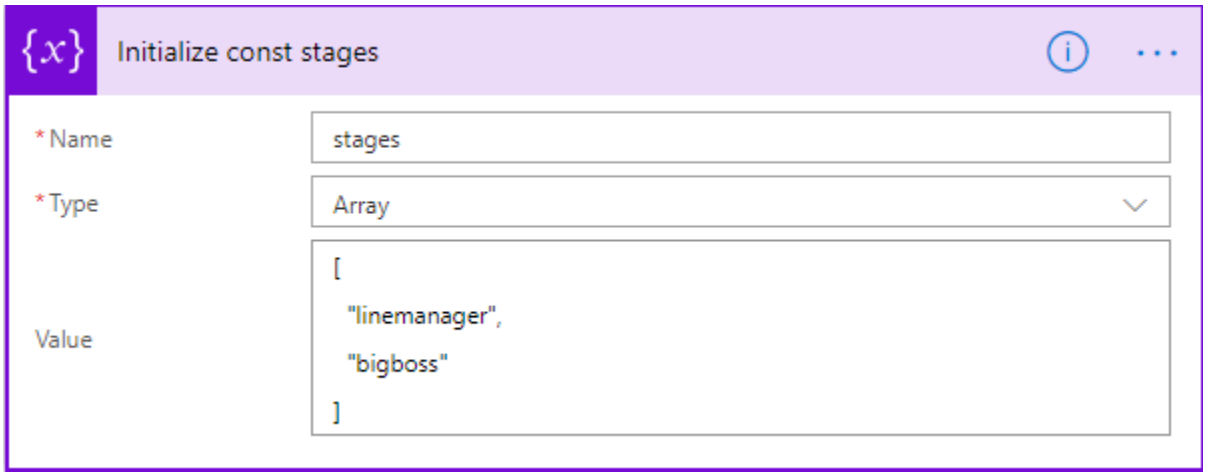
1. We will now create the flow controller flow. Create a flow called **Approval Controller**.
2. The trigger of this flow must be **When HTTP Request is received**
3. Copy and paste the following schema into the request trigger JSON schema:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "itemid": {
      "type": "integer"
    },
    "requester": {
      "type": "string"
    },
       "from": {
         "type": "string"
       },
       "response": {
         "type": "string"
       }
     },
     "required": [
       "requester",
       "itemid",
       "from"
     ]
   }
```

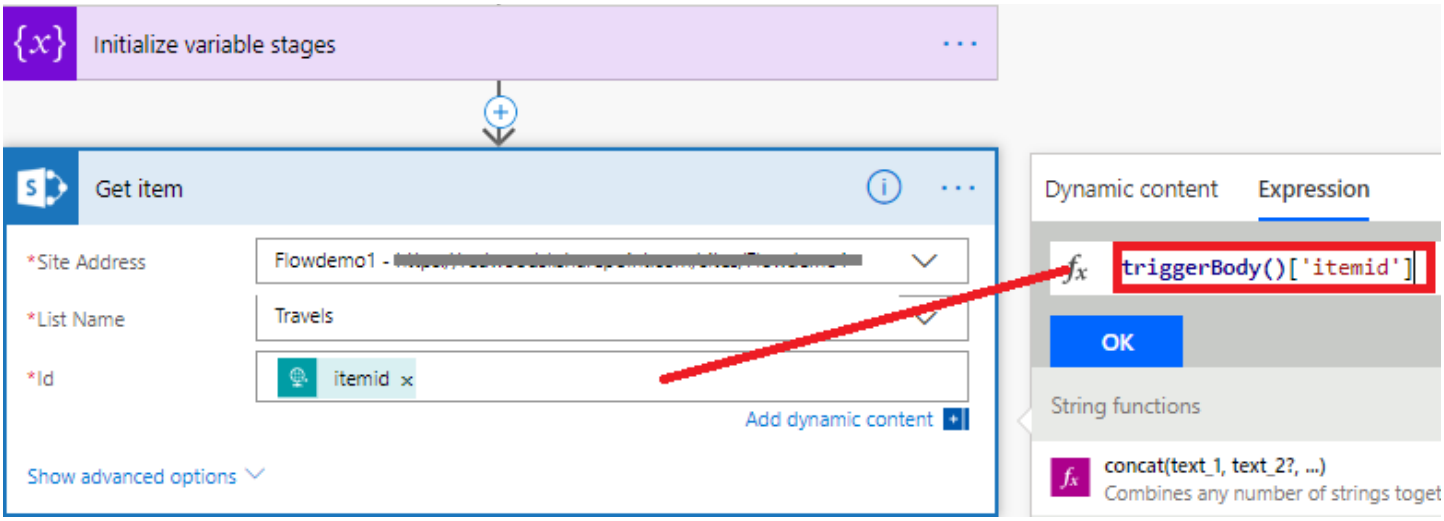
4. Add 3 **Compose** actions and name them as illustrated below:



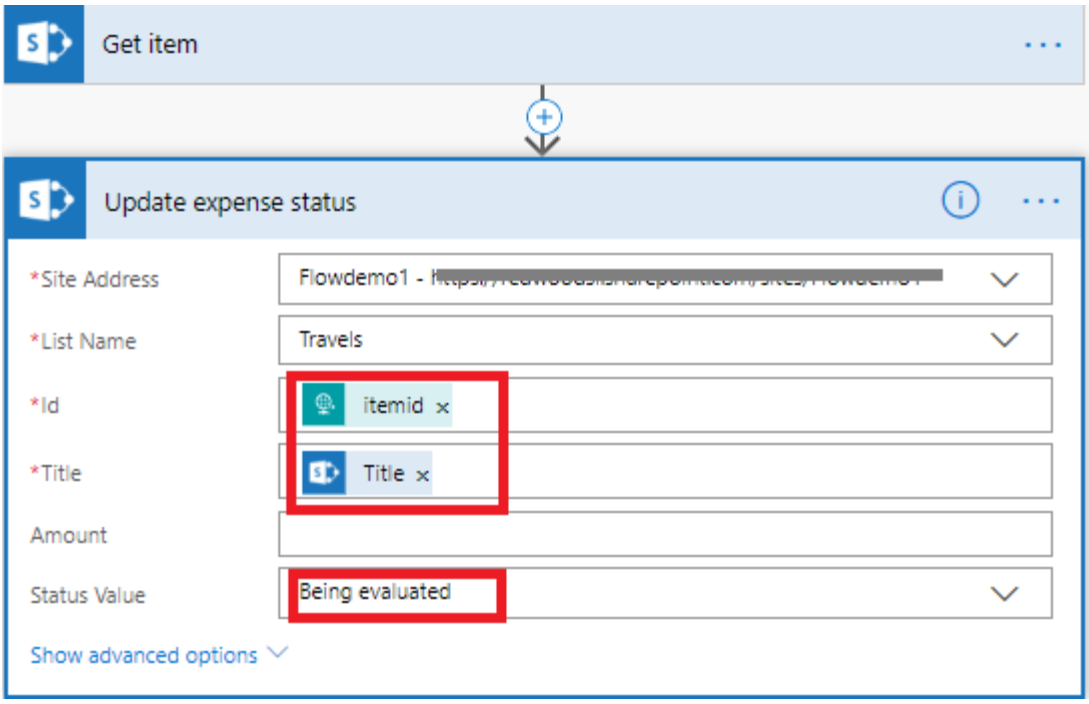
5. Store the e-mails addresses of the linemanager and of big boss in the 2 compose actions
6. Add "todo" in the **const approval stage url**.
7. Define a variable of type **Arra**y:



8. Add a SharePoint **Get item** action:



9. Add an **update item** action to change the status of the current expense and rename it **Update expense status**:

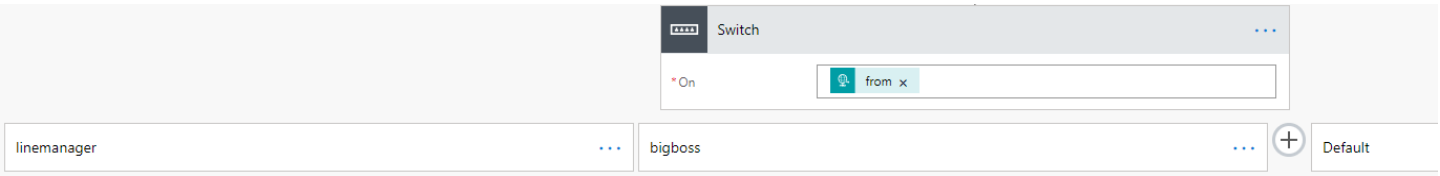


10. After the Update expense status, add a **Switch** action and check the **from** value:

11. Rename the switch **Check where the call comes from**.

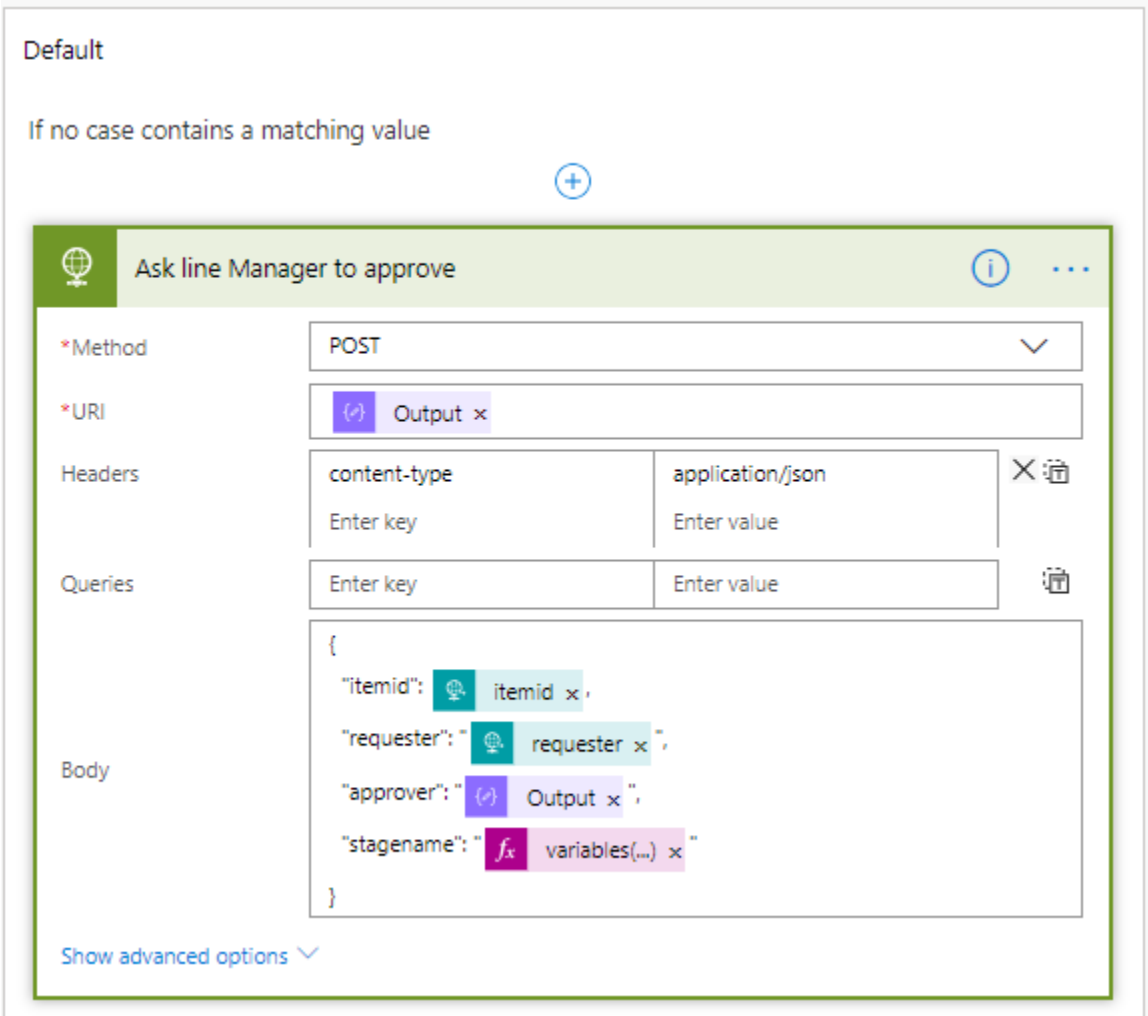
12. In the switch, we will analyze 3 scenarios:

    The from comes from the linemanager approval, from the bigboss approval or it is empty:



If **from** is empty we will go into the default branch where we will start the first approval (in this case line manager approval) :

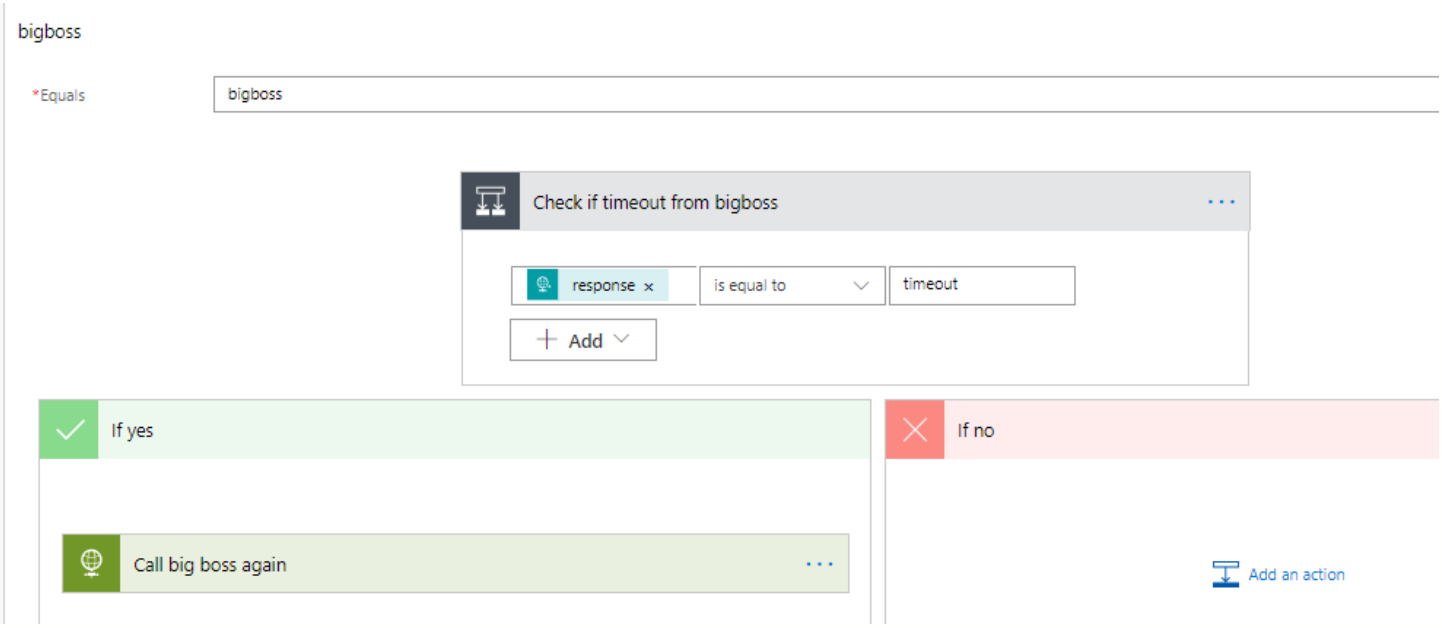
13. Add an **HTTP** request in the default branch:



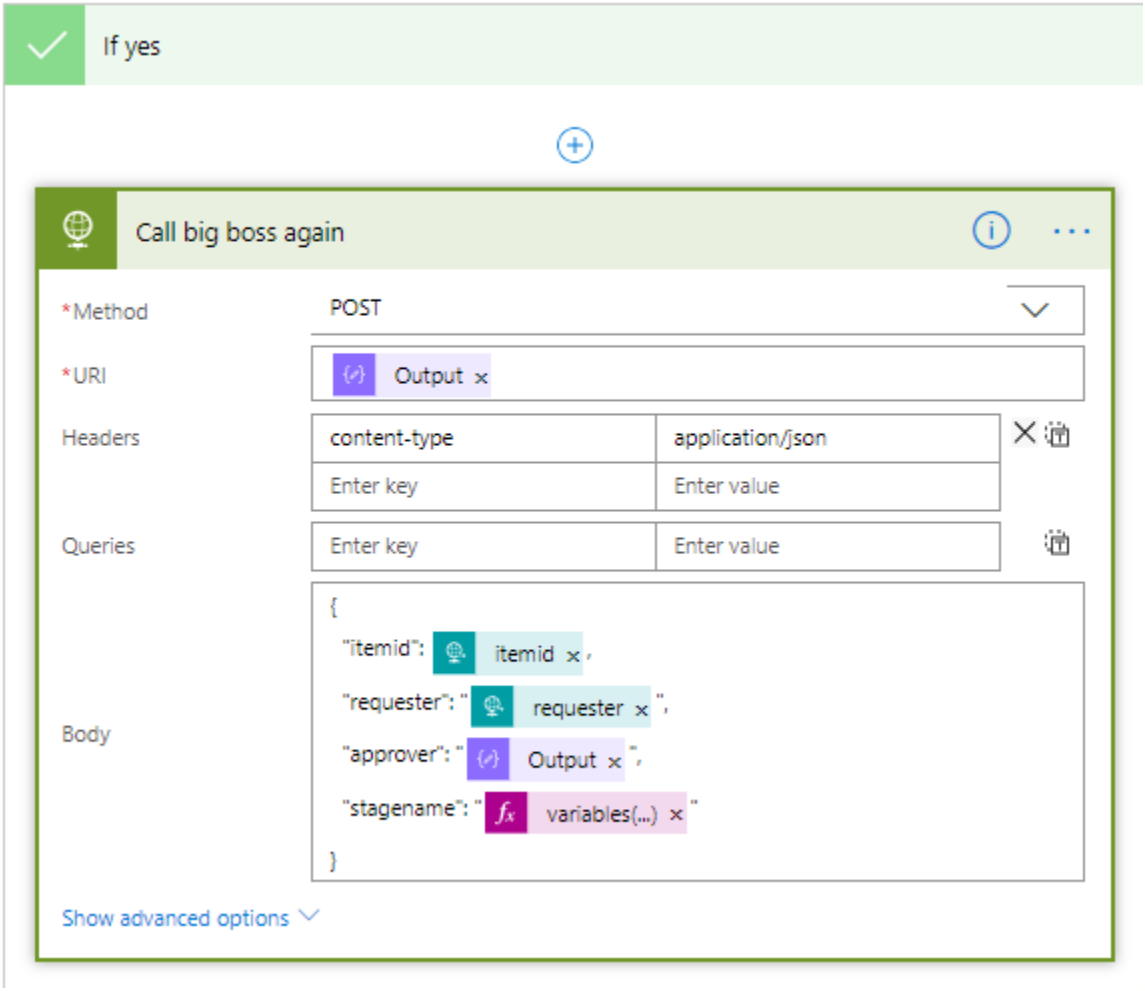
- In the URI : define the output value of the Compose **const approval stage url.**
- In the approver , define the output of the Compose **const linemanager email**.

- In the stagename, type the expression `variables('stages')[0]`

14. Here is what we must do if the value of from is "bigboss": (Big Boss Branch)
    We must check if the response is timeout; if that is the case then we need to call Big Boss again (new HTTP action):



15. The implementation of the HTTP action **Call big boss again** is the following:



Where the "stagename" value is the expression : `variables('stages')[1]`

16. Let's implement the LineManager branch . If the message is timeout, we must call the big boss again. This is pretty much what we have implemented in the big boss branch:

linemanager

*Equals linemanager

Check if timeout from linemanager

response × | is equal to | timeout

+ Add

If yes

Call Big boss

*Method POST

*URI Output ×

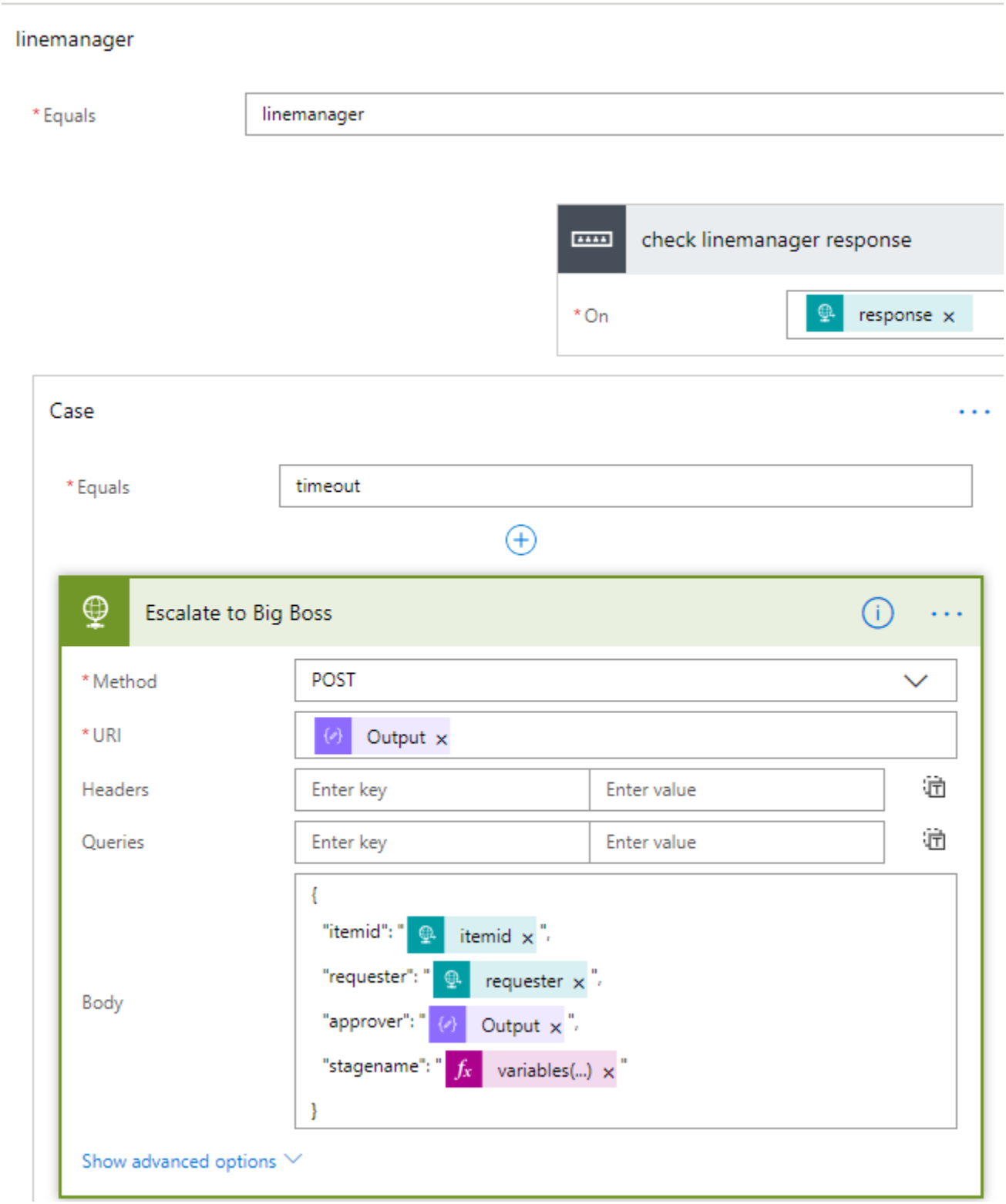
Headers
content-type | application/json
Enter key | Enter value

Queries
Enter key | Enter value

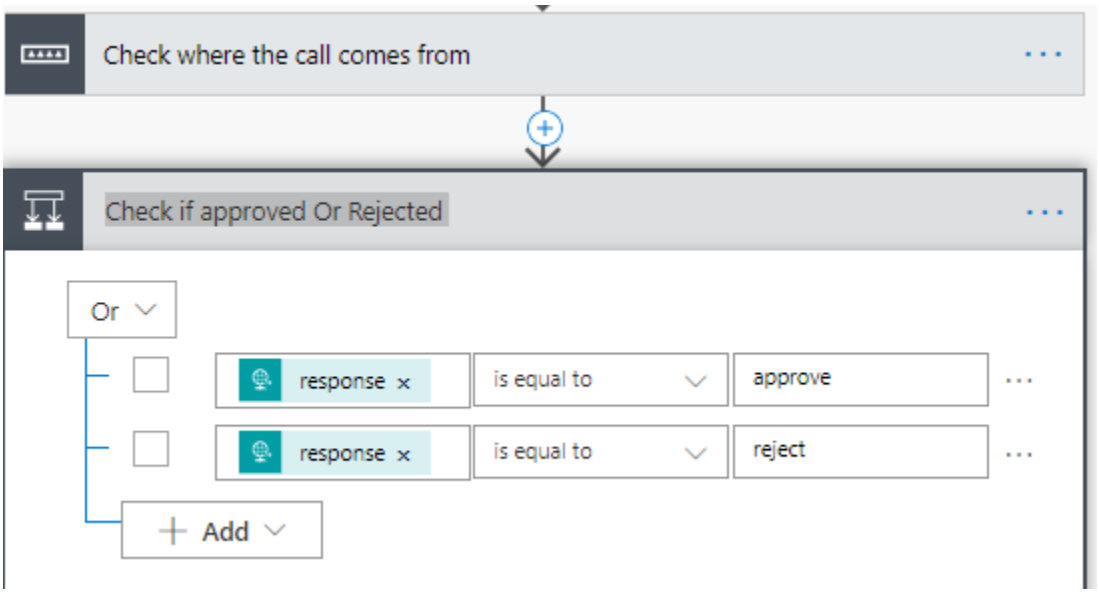
Body
```
{
 "itemid": itemid ×,
 "requester": " requester × ",
 "approver": " Output × ",
 "stagename": " variables(...) × "
}
```

Show advanced options

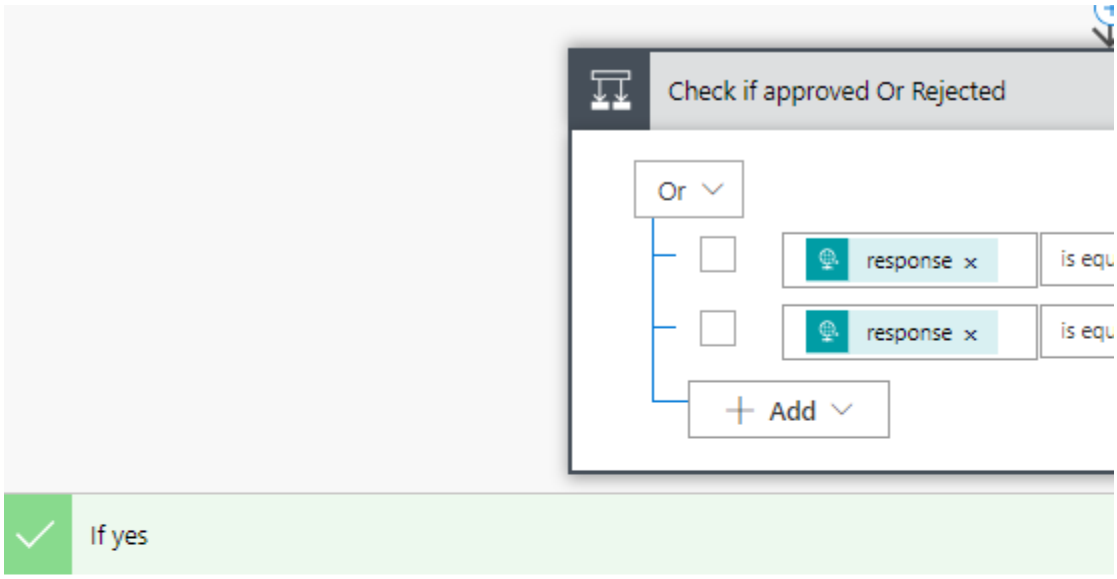
If no

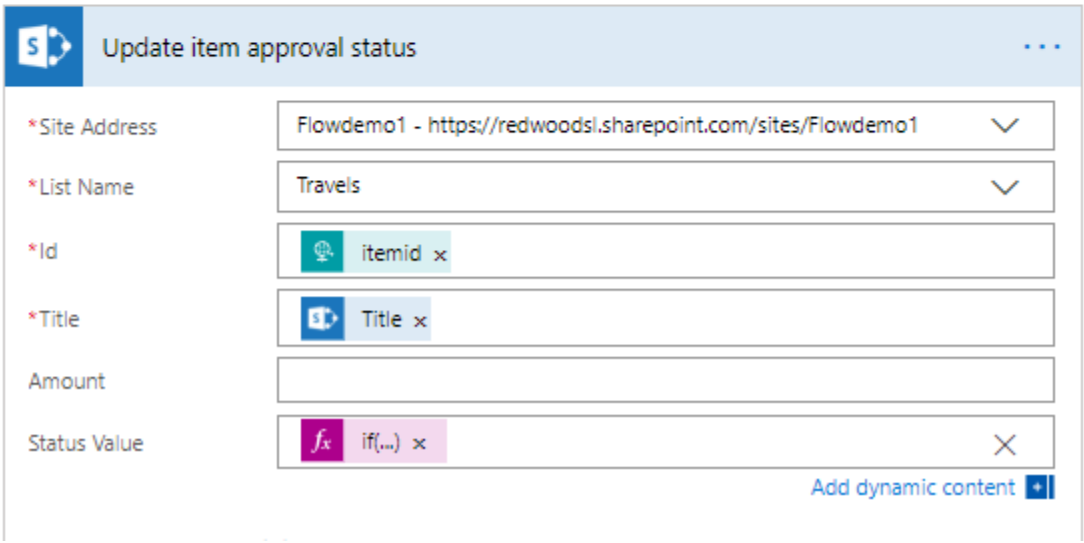


17. After the switch we test if the response was “approve” or “reject”: add a condition and name it **Check if approved Or Rejected**



18. In the left branch of the condition update the SharePoint item status:





The expression that updates the Status Value is the following:

```
if(equals(triggerBody()?['response'], 'approved'),'Approved','Rejected')
```

Your flow should look like this:

19. Save the flow and open it again the generate the associated public url in the request trigger (and copy the url):

20. Open the Generic Stage flow and paste this url in the **Compose const Controller url**.
21. Copy the Generac Stage public url and paste it in the controller flow (in the Compose const approval stage url).
22. Test you flow by adding a new expense in the SharePoint list.

Hackathon.

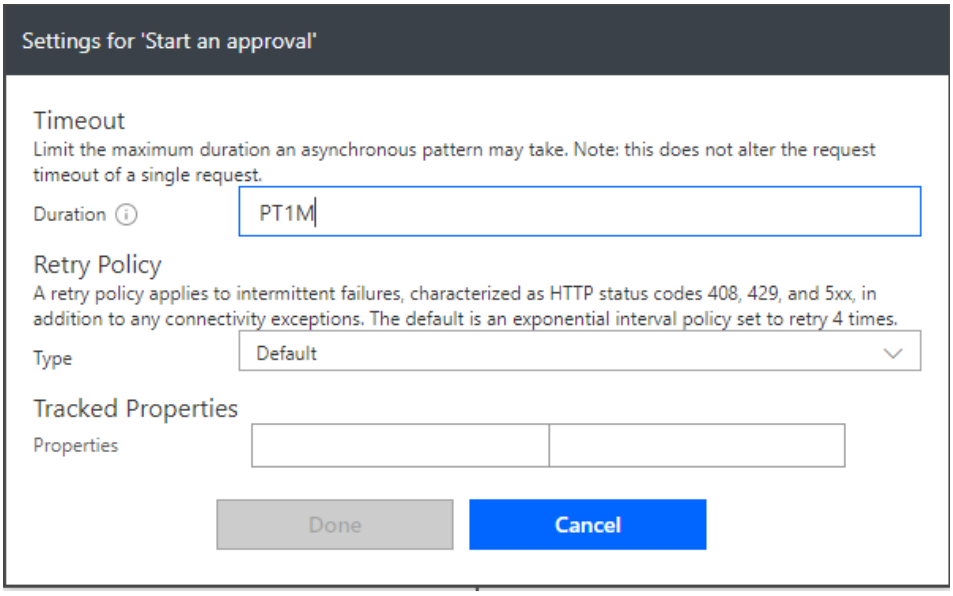
Theme 1. Timeout approval and escalation.

Complexity: easy

In the previous labs we illustrated a single approval process. In the real-world, approvals can be more complex.

1. If user, John, asks user, Linda, to approve an item, and Linda doesn't approve within a specific timeframe, you should send her a reminder.
2. If she doesn't react, you should escalate the approval to another user (typically the approver's boss).

Hint 1: You can define a timeout in an action if you go to the action settings. The duration is in format ISO 8601; if you can test with a 1-minute Duration: PT1M.



Additional references:

https://www.youtube.com/watch?v=U4iuVi1Vtgg&t=1s - Video Learn

https://sergeluca.wordpress.com/2017/10/22/microsoft-flow-approval-escalation-sharepoint/ -Read a Blog

Hint 2: You can use scope.

Hint 3: You can use Parallel actions https://flow.microsoft.com/en-us/blog/parallel-actions/.

Theme 2. Team bots -Request Teams channels

Complexity: easy

Scenario. Users can request new MS Team channel by creating a request in a SharePoint list. Create the SharePoint list with these requests. Create a flow that will send a daily e-mail containing the list of all Channel request. Call the flow from the MS Team bot. Watch this video : https://www.youtube.com/watch?v=T6qvb5B-r90&feature=youtu.be

Hint: you have to do a filter query and use an OData operator to filter the requests of the day; if you compare Date & Times in OData, use the datetime option: *When gt* ***datetime***'*<variable tomorrow>*'

There should be no space between datetime and ', otherwise you will get an error message at runtime.

Read this: https://flow.microsoft.com/en-us/blog/advanced-flow-of-the-week-filtering-with-odata/

Theme 3. Hot Dog, Not Hot dog

Complexity: medium

Scenario: You will create a flow that will take a picture of food and that will say if the food is a hot dog or not. This is inspired from the famous episode https://www.youtube.com/watch?v=ACmydtFDTGs. Here is an overview of the implementation of Hot Dog Not Hot Dog by the Flow team: https://www.youtube.com/watch?v=Q5Oe0Yjmu-k .

Set up the Vision project

- You can create a trial Azure Vison account by going to https://www.customvision.ai.
- In this site, create a new project, let's call it "Hot dog not hot dog"; select the food classification.
- Find 10 pictures of fruits and 10 pictures of hot dog and download them.
- In the project add the tag hotdog and the tag fruits.
- In the project add your pictures and tag them with hotdog or with fruits accordingly.
- Click the Train button to train the system.
- In the project click Performance, you should see iteration 1. Click Already default.

Create your flow

- The trigger must be a button
- In the button setting add an input of type File
- The picture you will upload will be stored in your OneDrive for Business, so add a OneDrive For Business Create file action
- Add a Custom Vision – Predict from image action where you will set
    - the image content field to the upload file content
    - the project ID that can be retrieved from your Vision project site
    - the iteration ID retrieved from your project Vision site
- The Predict from image custom action will generate a list of predictions that you will have to go through by using an Apply to each

- In this Apply to each, you can check if the tag prediction value of hotdog in great enough and much greater than the one of fruits; if that is the case send a notification with this is a hot dog, or this is not a hot dog.

Theme 4. Twitter Sentiment Analysis

Complexity: medium

Scenario:

Retrieve tweets from twitter and run them through a sentiment analysis engine in order to determine if their nature is positive or negative. Once you have captured this information you can post it Power BI.

1. Capture survey information about Flow and create an lively refreshed PowerBI Dashboard: read this: https://flow.microsoft.com/en-us/blog/forms-and-flow-and-powerbi/.
2. Send the comments to Azure Cognitive service and store the scores in the SharePoint list or to the PowerBI Dashboard : read this: https://flow.microsoft.com/en-us/blog/flow-of-the-week-2/.

Theme 5. Intelligent Customer Service

Complexity: high

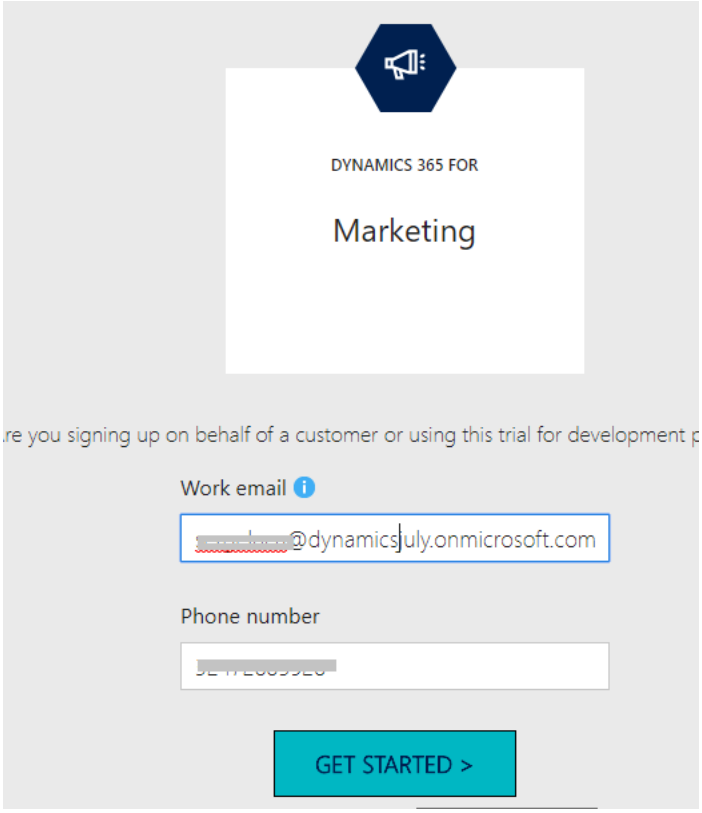
Scenario:

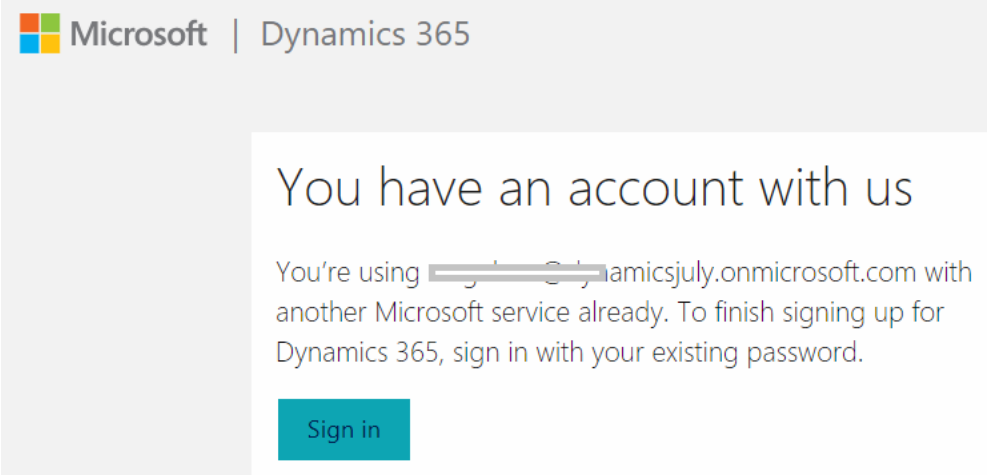
A power company, called Contoso Energy, would like to improve their customer service and wants to provide many channels for customers to solicit assistance. In addition to mobile, social and telephone conversations, they continue to receive customer support requests over email. Sifting through emails manually to log support cases inside of their CRM (Microsoft Dynamics 365) is a time consuming and low-value activity. Especially, when you consider the people that can help solve the customer's issue, are not immediately informed of the issue.

As a power company, there are many customer requests that can be made including reporting a power outage, billing queries, street light replacement and vegetation brushing to name a few. Some of these requests are higher priority than others so it is important we understand the user's intent when we log the support case. This allows us to prioritize their case correctly and ensure the right department can address high priority incidents asap!

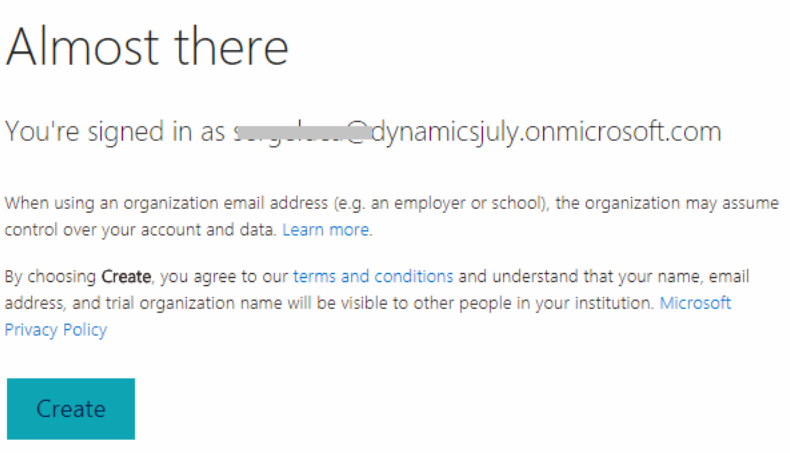
Read this https://flow.microsoft.com/en-us/blog/automating-intelligent-customer-service-using-microsoft-flow-luis-ai-and-dynamics-365/

You need to a free trial of LUIS.ai and a Dynamics 365 trial: https://trials.dynamics.com/ (use Dynamics 365 for Customer Service).





Sign-in, and click **Create**:



Additional Resources:

For More Microsoft Flow Resources Head over to AKA.MS/Flow-Resources

To Never Miss a Microsoft Flow Blog Post – Use THIS Template

For Documentation visit - Microsoft Flow Documentation

Visit the Microsoft Flow Community and Talk with other Flow Fans and Experts!

Special Thanks:

Special Thanks to Serge Luca of ShareQL for partnering with the Microsoft Flow Team to help create this material.