

# 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

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

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

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

© 2018 Microsoft Corporation. All rights reserved.

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

# Content

<b>Microsoft Flow in a Day step-by-step hands-on labs &amp; Hackathon.....</b>	<b>1</b>
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

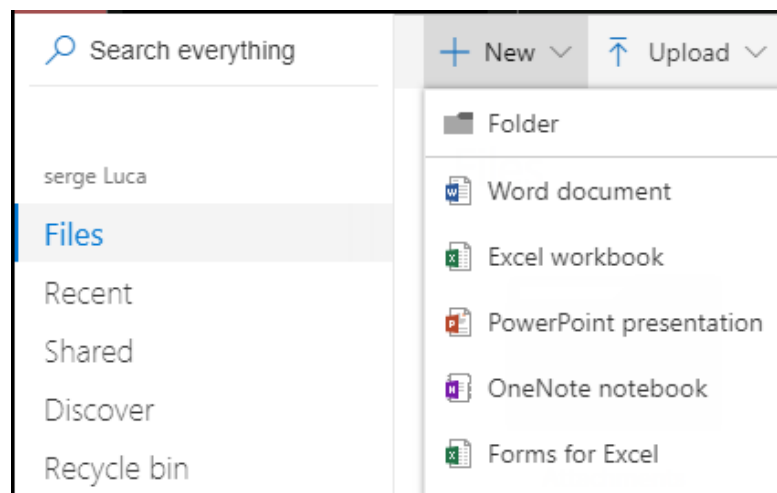
**Scenario:** When a user pushes a button, the current time and the current location will be saved in an Excel document stored in OneDrive for Business.

**Perquisites:** Ensure the timesheet.xls in your OneDrive

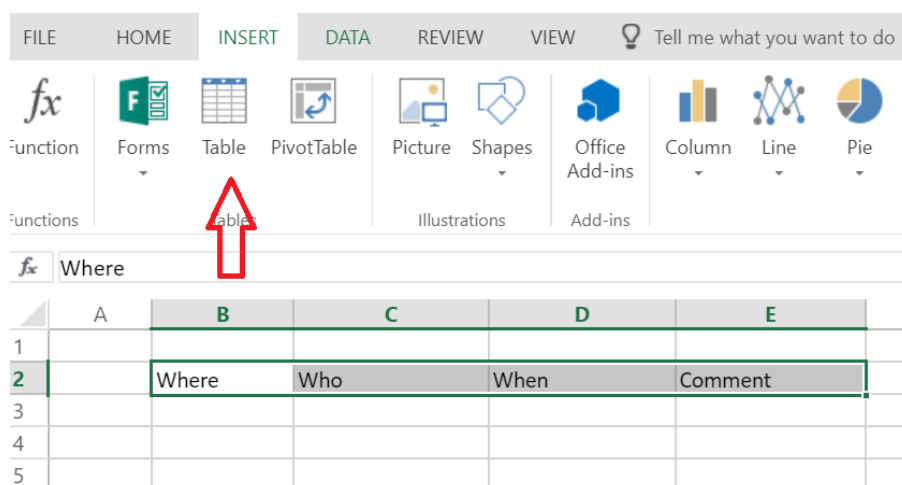
### Build this Flow from Blank by following the below steps!

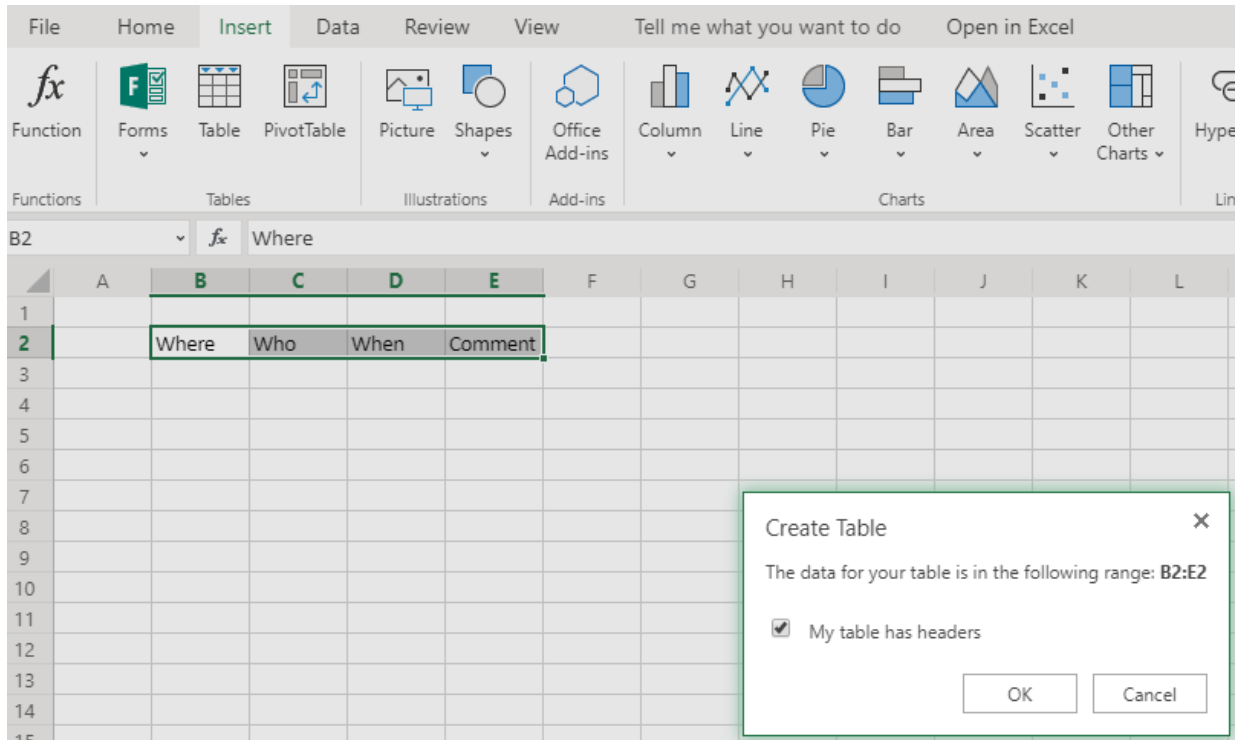
#### Tasks:

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

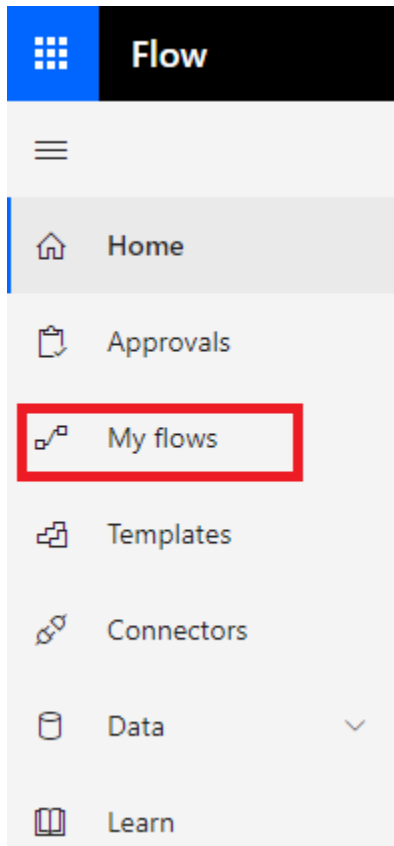


2. Create 4 columns (**where, who, when, comment**) and format these as a table (check the box, has headers):

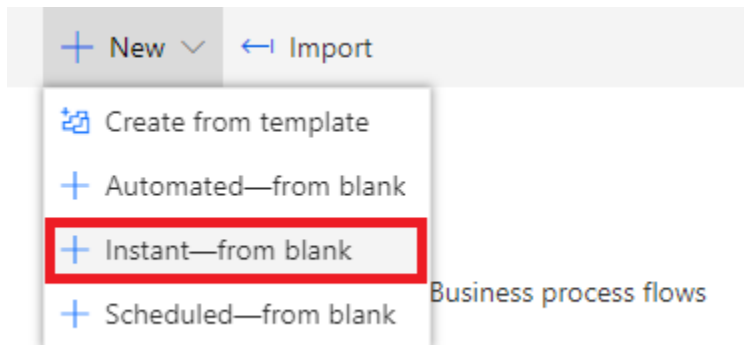




3. Save the file (Save as - Rename) with the name **Timesheet.xlsx**.
4. Let's create the **Track Time** flow: go to [flow.microsoft.com](https://flow.microsoft.com) and if requested, sign-in.
5. Go to **My flows**



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



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



## Build an instant flow



Flow name

Tack Time

Choose how to trigger this flow \*



From Microsoft Flow  
Microsoft Flow

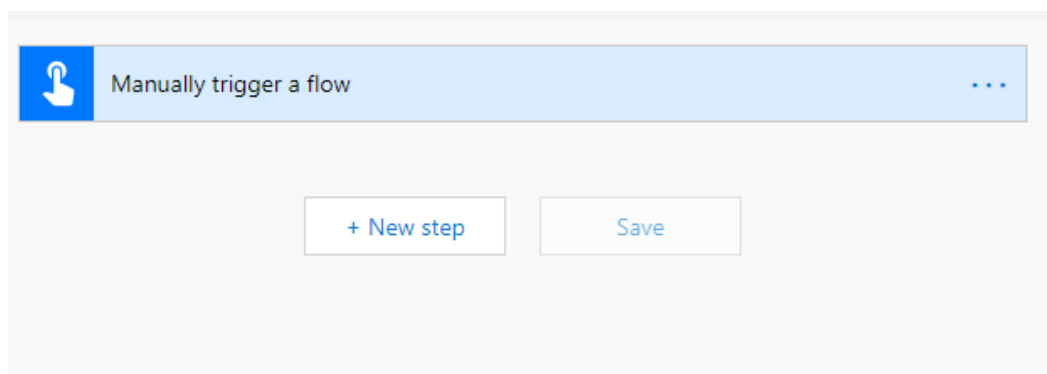


From PowerApps  
PowerApps



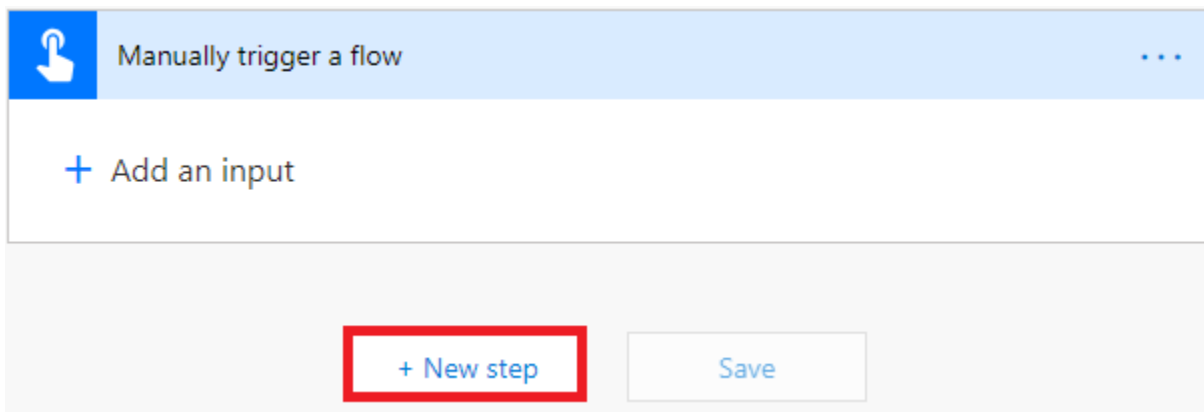
- Click Create.

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

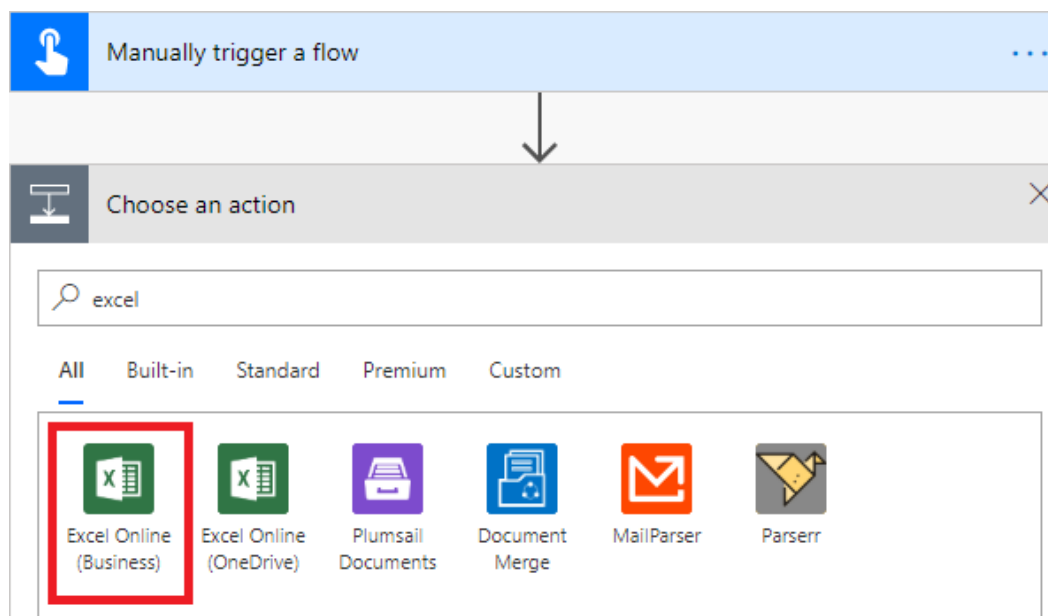


**Note:** This trigger will start the flow by pressing a button. The button can also be visible in the flow mobile app, including in the home screen of a smartphone. If needed, a form can show up when the user pushes the button. But in any case, some default information like the current user coordinates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

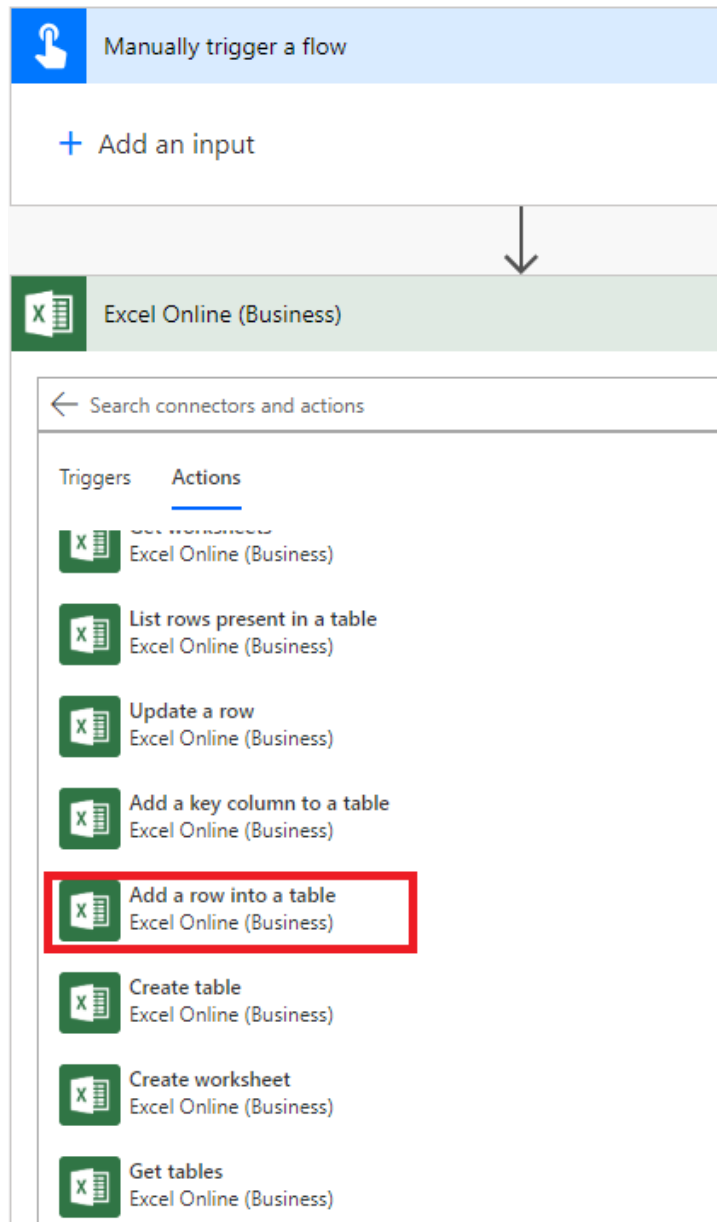
- Click on **New step**:



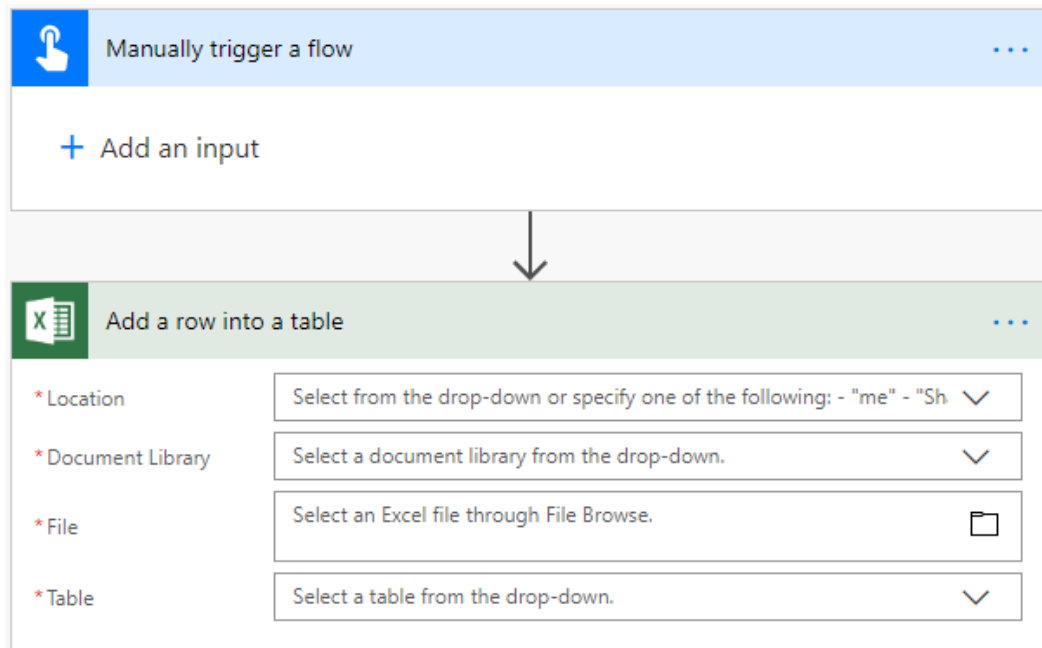
10. and then in **Choose an action**, type Excel to find the Excel Online (Business) connector (not Excel Online OneDrive; be careful here many users select the wrong connector); select it :



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



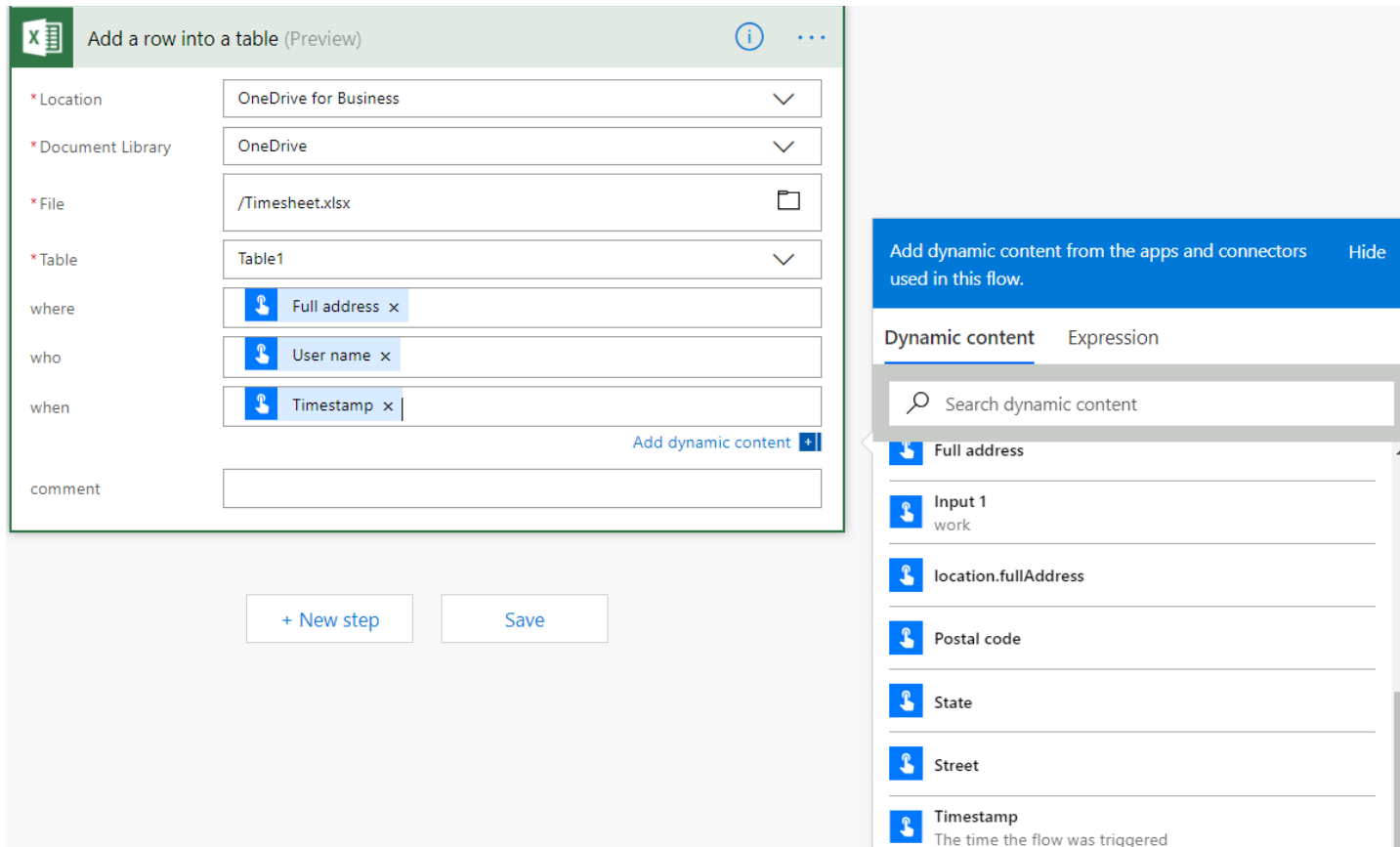
12. You should have this:



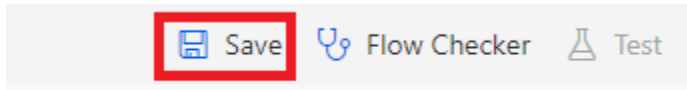
13. You need to configure this action; add a **Location**, **Document Library**, **File** and **Table** based upon the location of your spreadsheet in OneDrive for Business.

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:

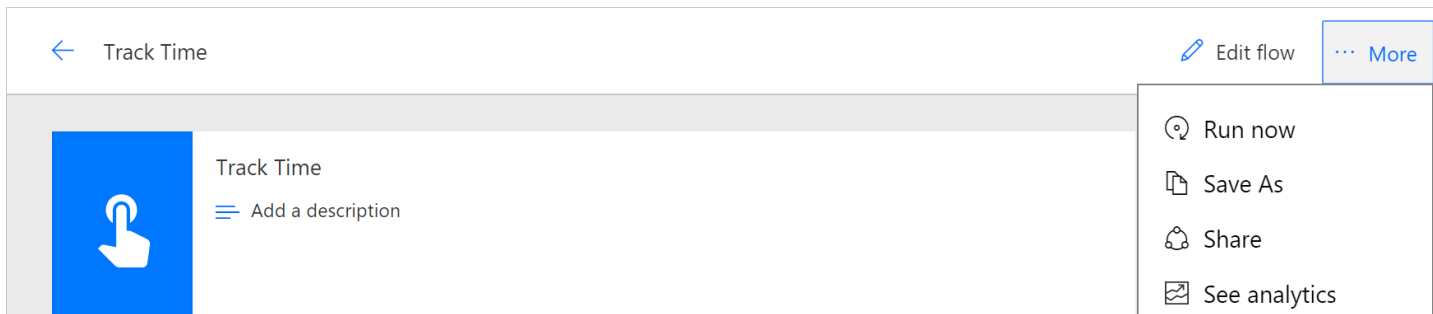


15. Save the flow (the Save menu in the upper right corner)



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:

The screenshot shows a Microsoft Flow workflow with three steps:

- Manually trigger a flow** (Trigger)
- Convert time zone** (Action)
  - \* Base time: Timestamp
  - \* Source time zone: (UTC) Coordinated Universal Time
  - \* Destination time zone: (UTC+01:00) Brussels, Copenhagen, Madrid, Paris
  - Format string: dd/MM/yyyy-T-hh:mm:ss
- Add a row into a table** (Action)

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

**Add a row into a table (Preview)**

\* Location: OneDrive for Business

\* Document Library: OneDrive

\* File: /Timesheet.xlsx

\* Table: Table1

where: Full address x

who: User name x

when: Converted time x

comment:

[Add dynamic content](#)

Add dynamic content from the apps and connectors used in this flow. [Hide](#)

**Dynamic content** Expression

Search dynamic content

Convert time zone

Converted time

Manually trigger a flow [See more](#)

City

comment

Country/Region

Date

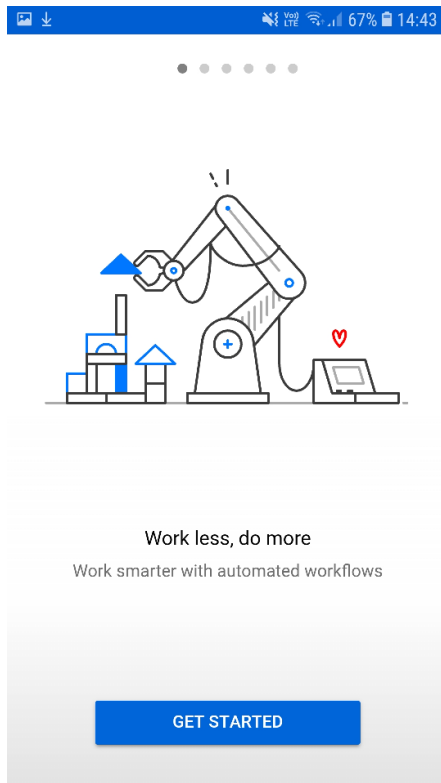
22. Save the flow, run it again and check your Excel sheet:

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

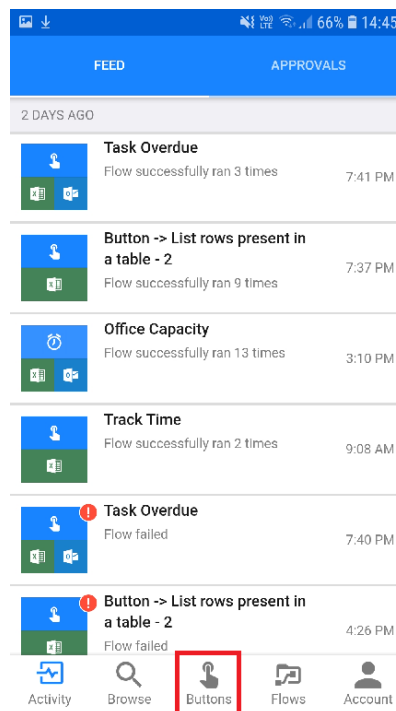
## Optional exercise: interacting with your flow on your mobile

**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**

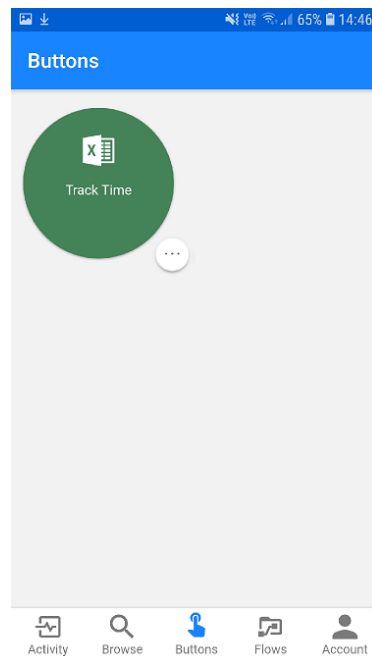


3. Sign in with your account.
4. Click **Buttons**:

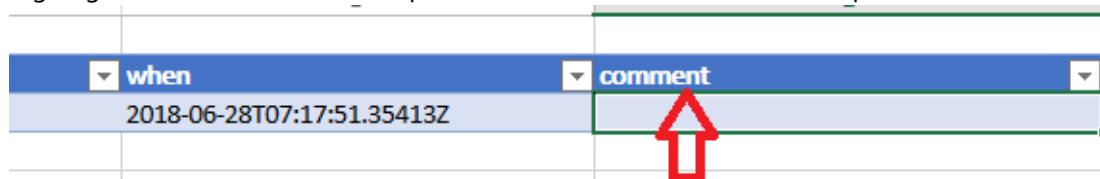


5. Your **Track Time button** should be displayed, and you can go ahead and click the button to execute the flow:

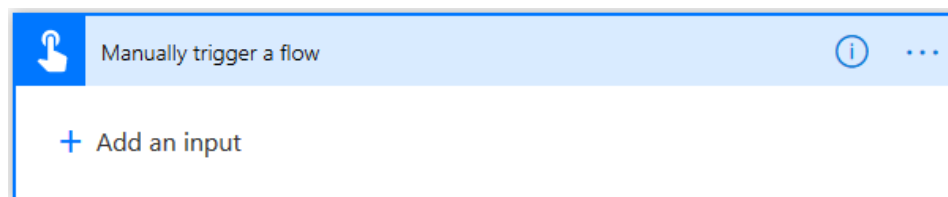




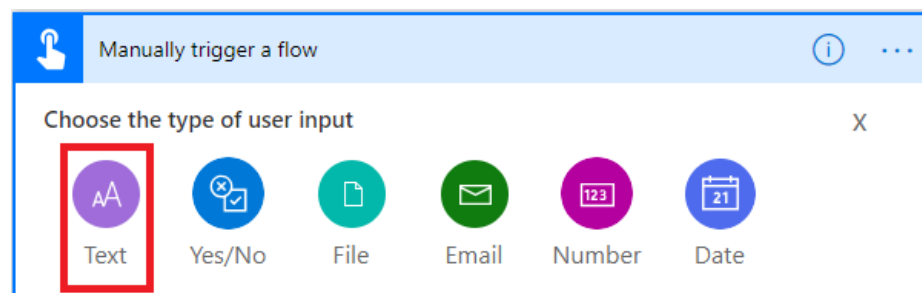
6. Now we are going to focus on how we can update the **comment** field within our spreadsheet:



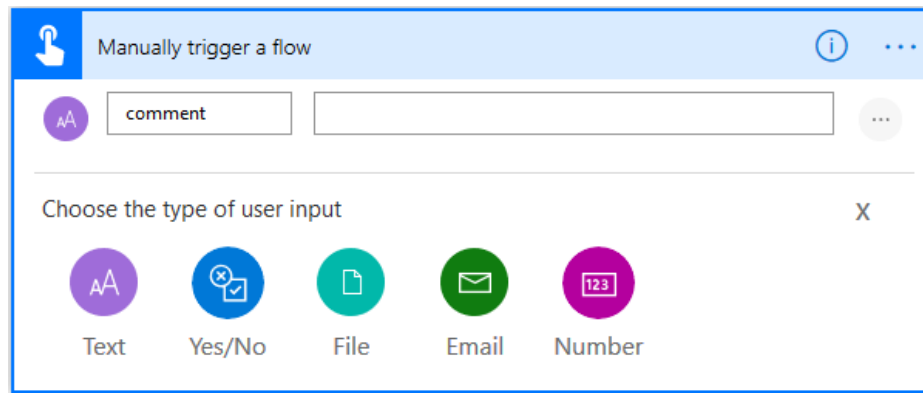
7. Edit the flow, click on the flow trigger (**Manually trigger a flow**) and click on **Add an input**:



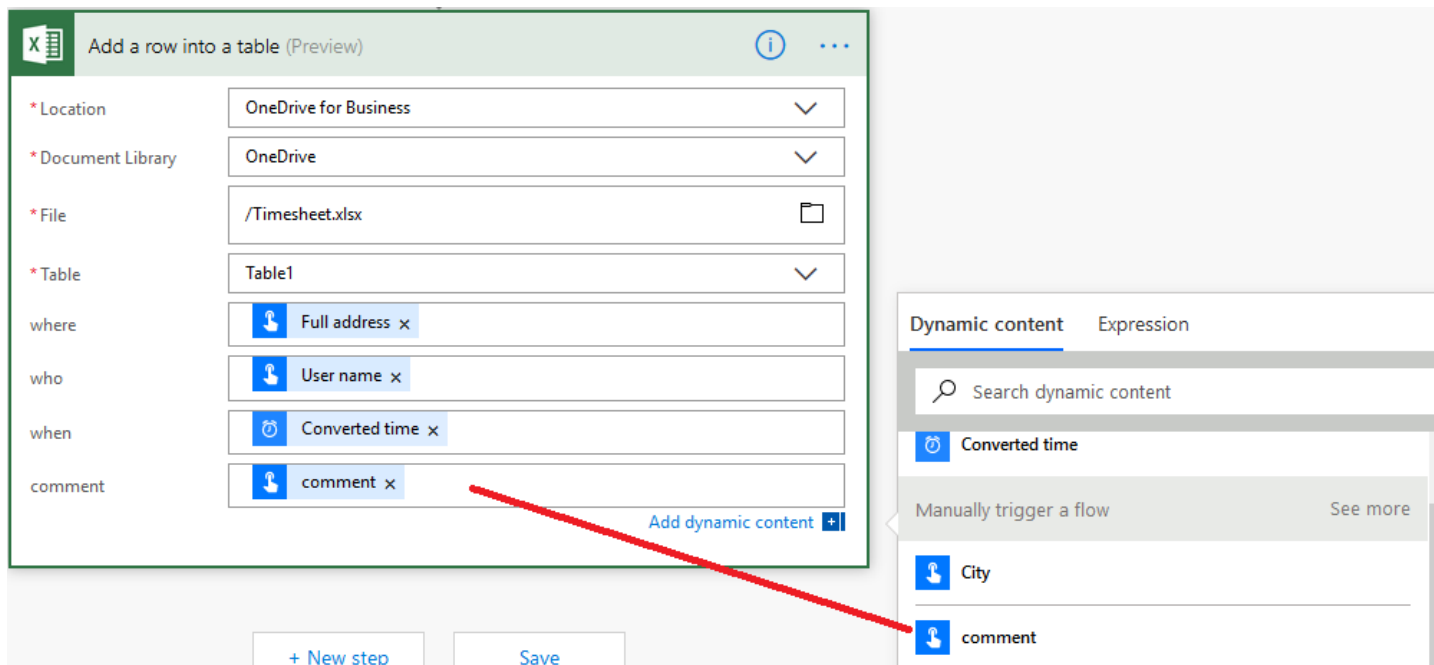
8. There are many options to provide greater interaction with users when they click on a Flow button. Select the **Text** option to add a text field:



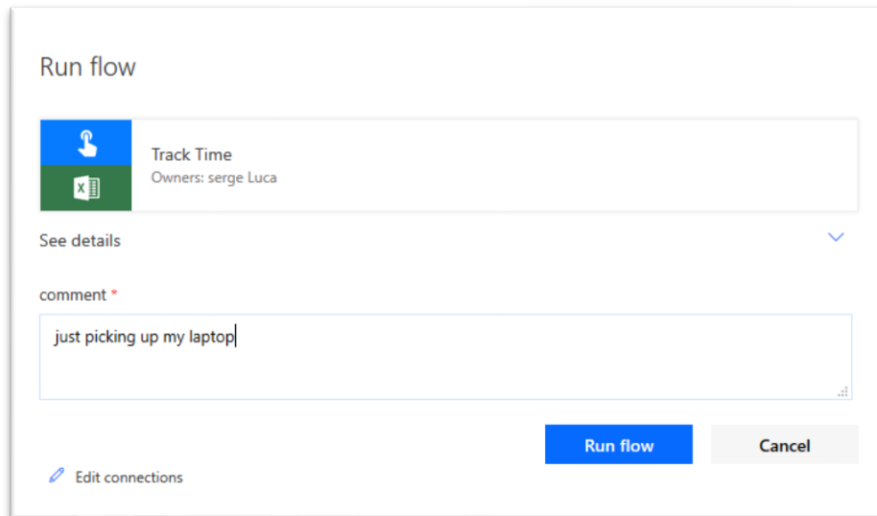
9. Replace **Input** text value with **comment** and provide a single space as default value:



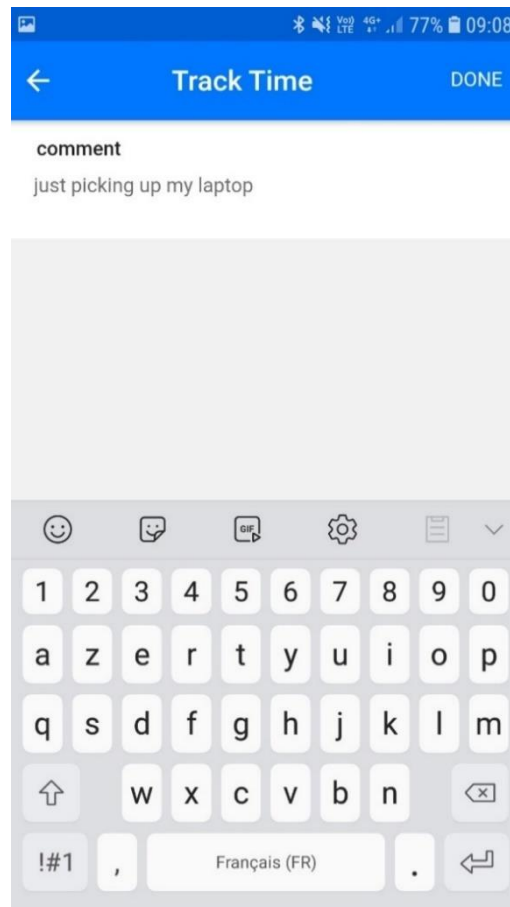
10. Click the action **Add row into a table**. We are going to store the comment field into the spreadsheet's **comment** column:



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).

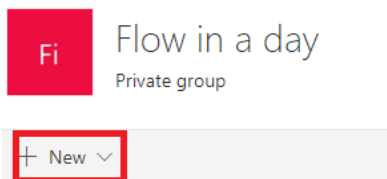
**If you want to take a look at the solution:** select the [Submit a travel request for approval](#) template.

**If you want to start the lab from scratch:**

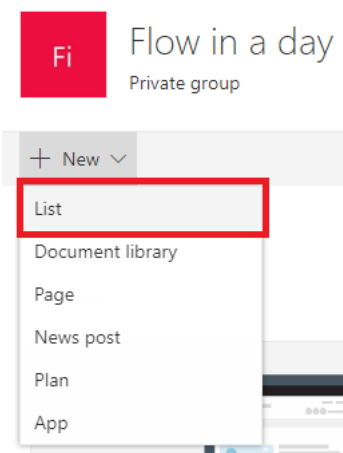
### Tasks:

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>**:

## Create list

Name \*

Travels\_AyrtonSenna

Description

☒ Show in site navigation

Create

Cancel

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

Travels\_AyrtonSenna

Title ▾

+ Add column

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

Close (X)

### Create a column

[Learn more about column creation.](#)

Name \*

Amount

Description

Type

Currency

Number of decimal places

Automatic

Currency format

\$123,456.00 (United States)

Default value

Enter a number

☐ Use calculated value ⓘ

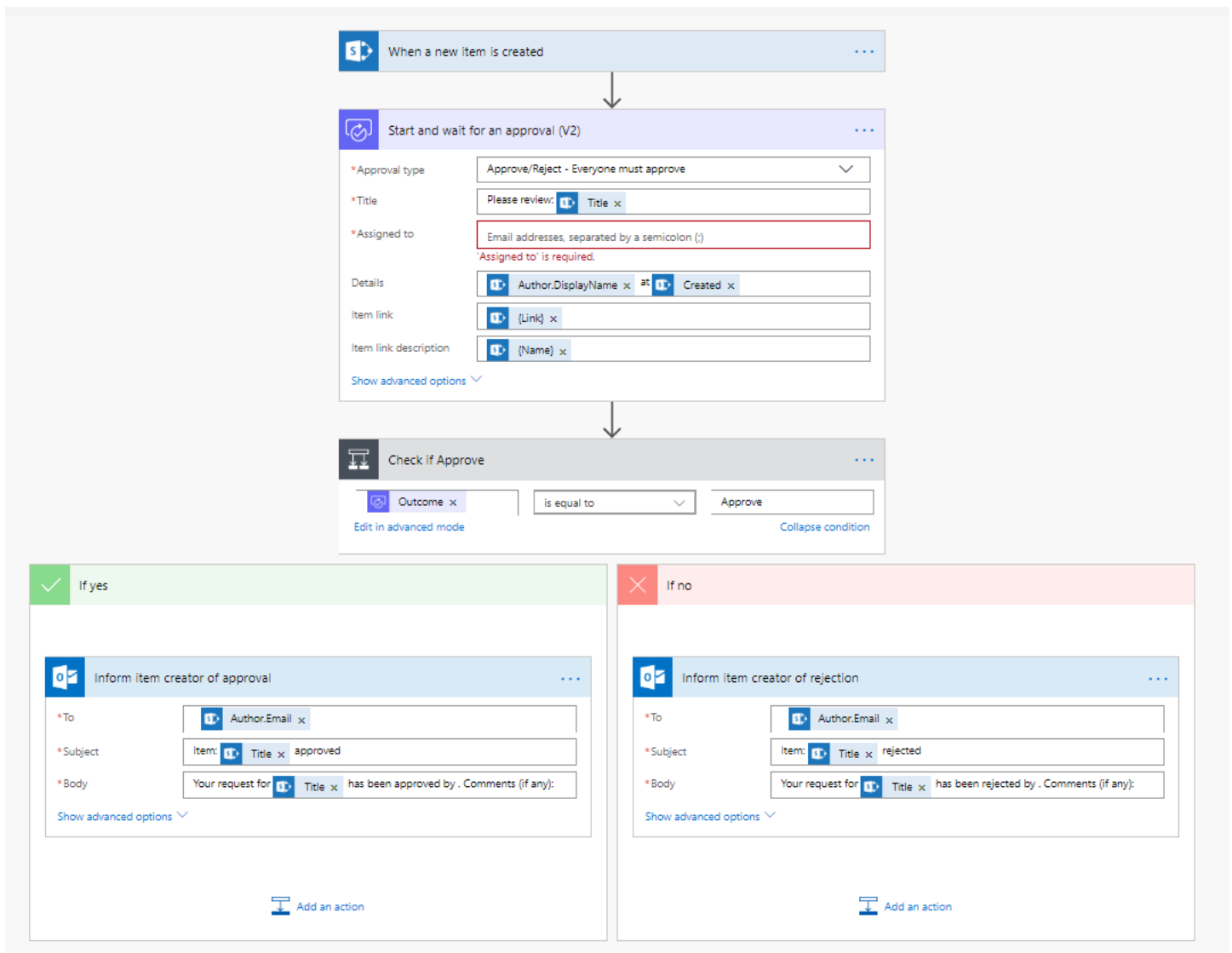
[More options](#)

Save Cancel

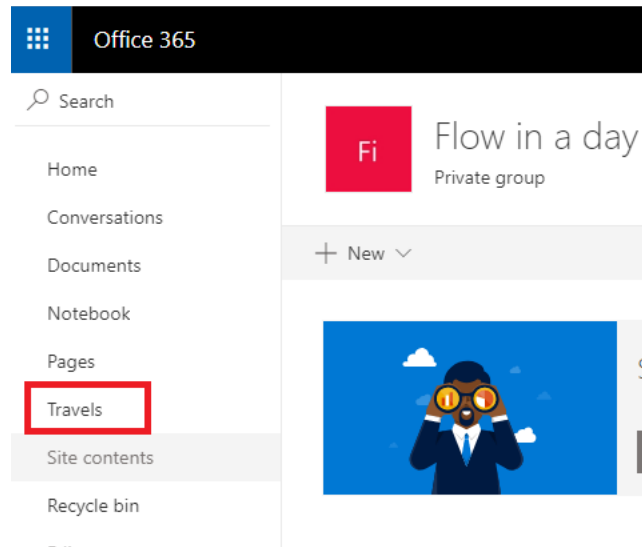
6. Click **Save**.
7. Now create a **Single line of text** column named **Status**.
8. Create a flow from the template [Start Approval when an item is added](#).

**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.

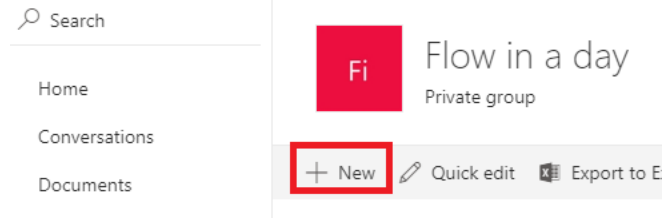
9. Once you have clicked Continue from the **Start approval when a new item is added**, the flow will look like this:



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):



 Save  Cancel  Copy link 

### New item

**Title \***

Paris

**Amount**

500

**Status**

Enter value here

**Attachments**




[Add attachments](#)

Save

Cancel


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


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



### Approve Travel


Use this template for processing approvals on SharePoint list items. The approver can view their approval requests in the Approvals Center and over email. Once an item is approved or rejected, the item creator is sent a confirmation email.

 Edit description

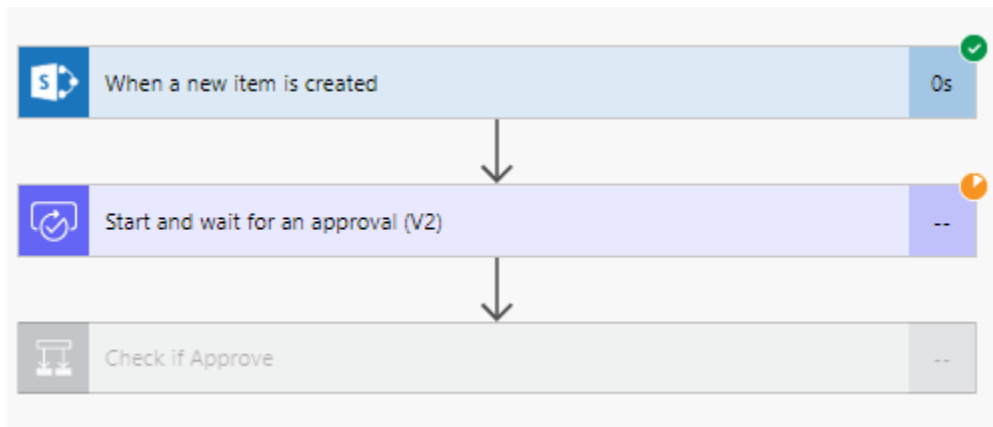
 See analytics

#### RUN HISTORY

See all >

 Running	19 seconds ago	19 seconds ago	>
---	----------------	----------------	---

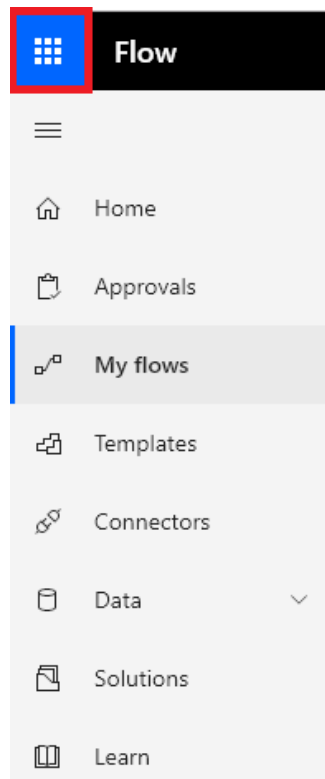
18. If you click the flow can you see where this flow instance is. Click Running and you will see this:



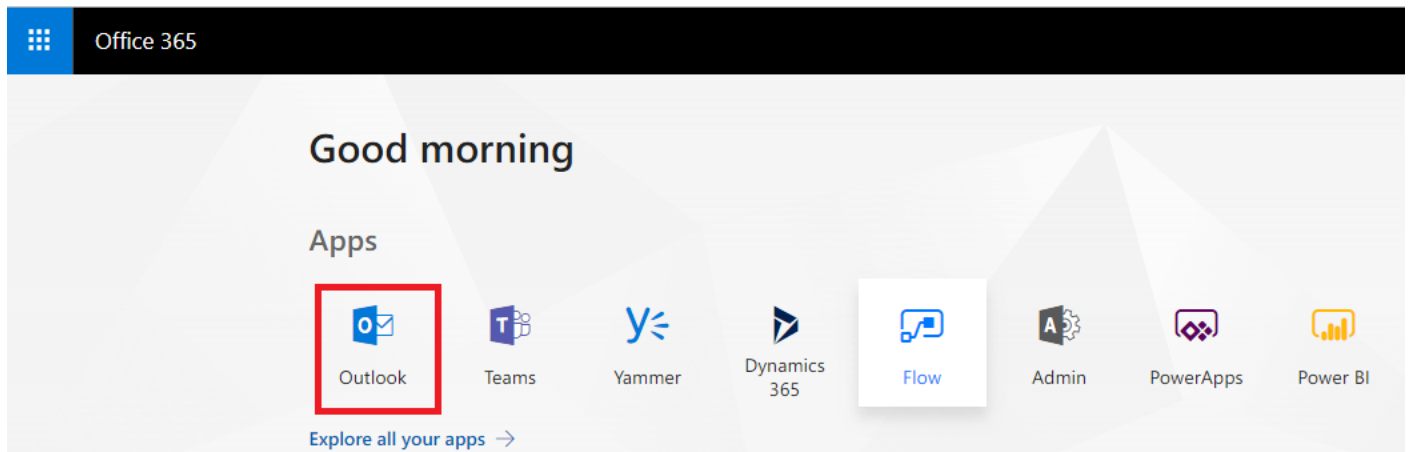
19. The flow is waiting for an approval. Approvals send an e-mail to the person(s) in charge of the approval.

20. Check your e-mail to display your approval e-mail:

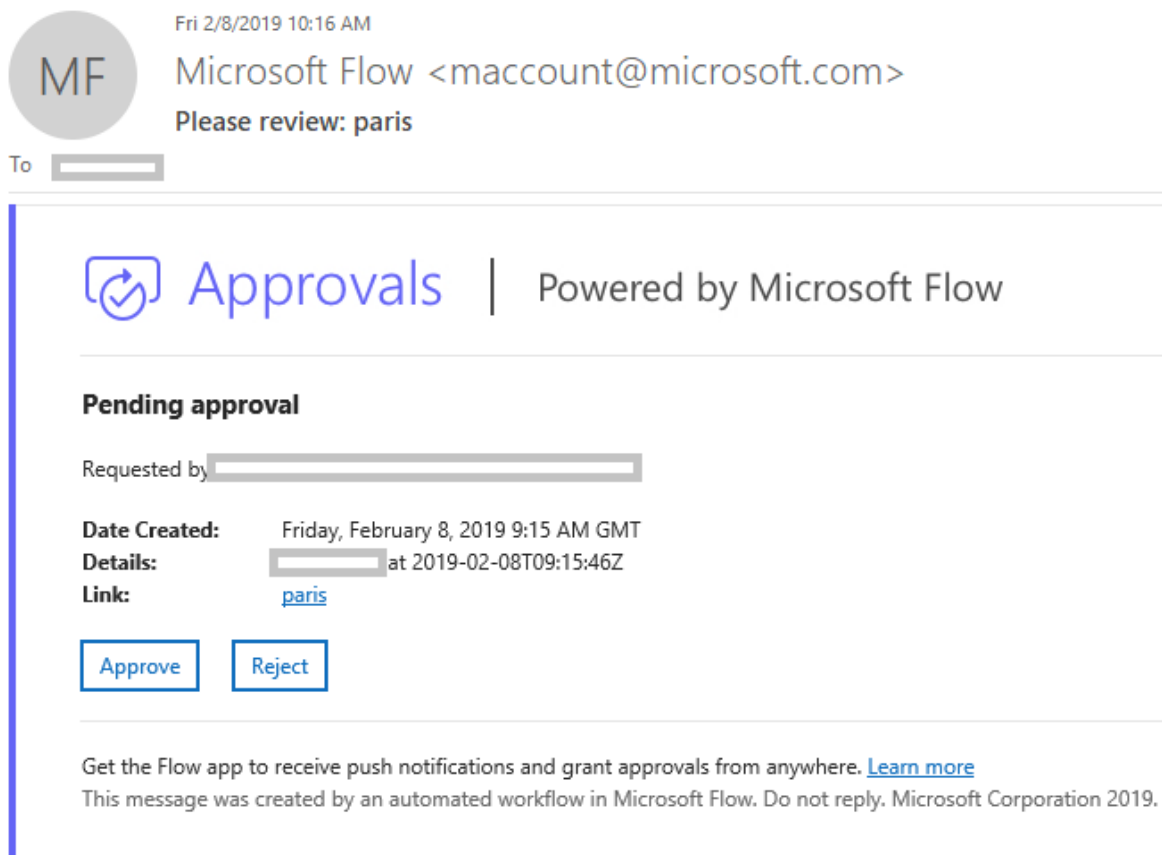
- a. To check your e-mails right click on the app launcher Flow button to open a link in a new tab



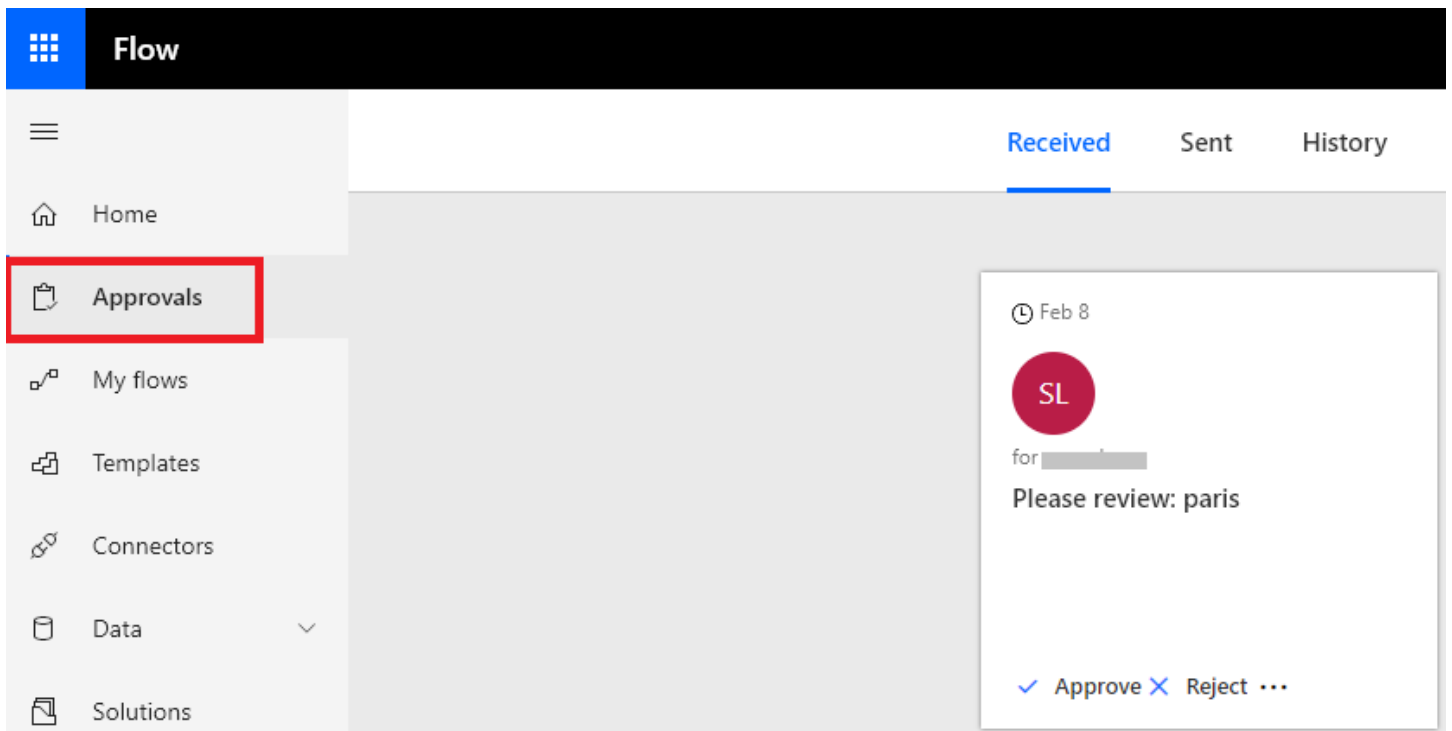
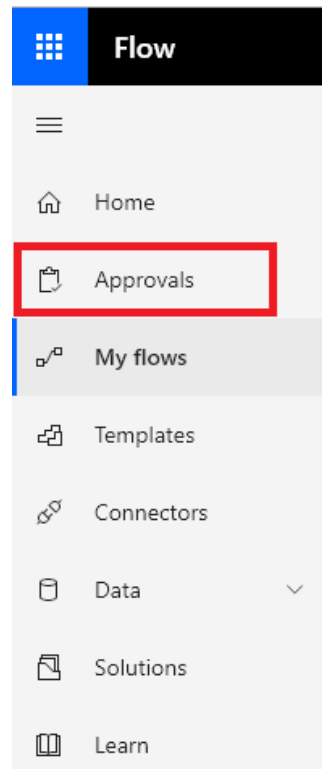
- b. In the list of Apps, click **Outlook**:



- 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...

Please review: paris

For



[Redacted Name]  
[Redacted Email]

Requested

Feb 8, 2019 at 10:15 AM

Details

~~Request~~ at 2019-02-08T09:15:46Z

Link

[paris](#)

✓ Reassign

Status

You have chosen to **Reject** this request. ...

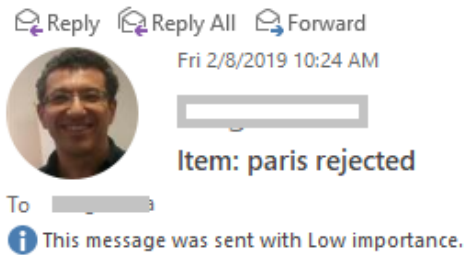
Add comments (optional)

Sorry Paris is not approved in the company policy

Confirm

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

## Microsoft Flow in a Day

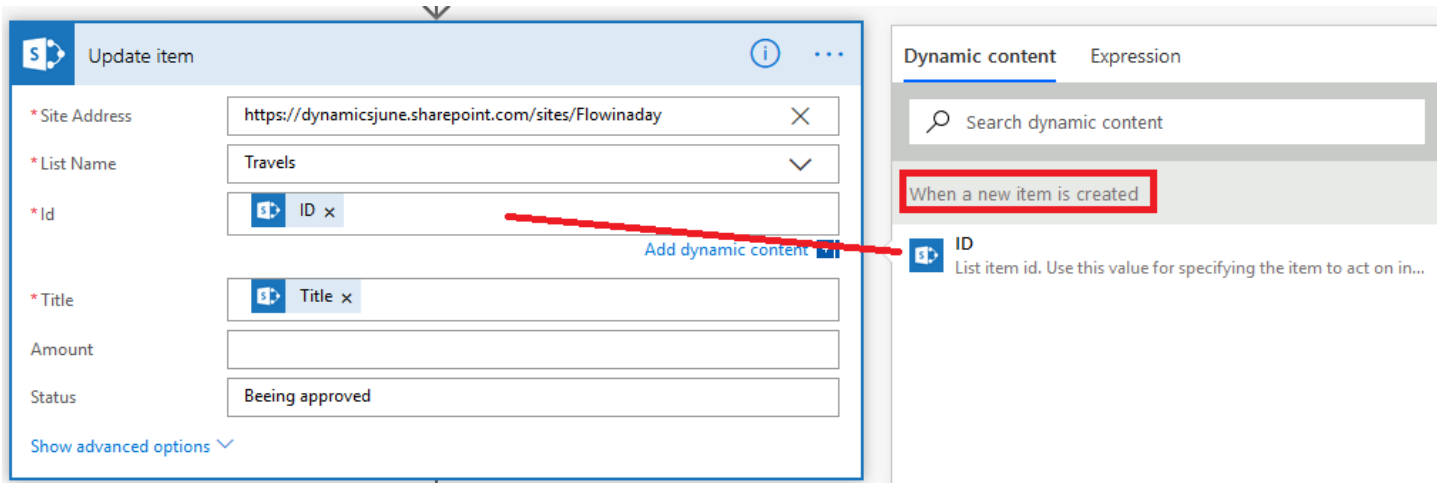


Your request for paris has been rejected by [redacted]. Comments (if any): Sorry Paris is not approved in the company policy

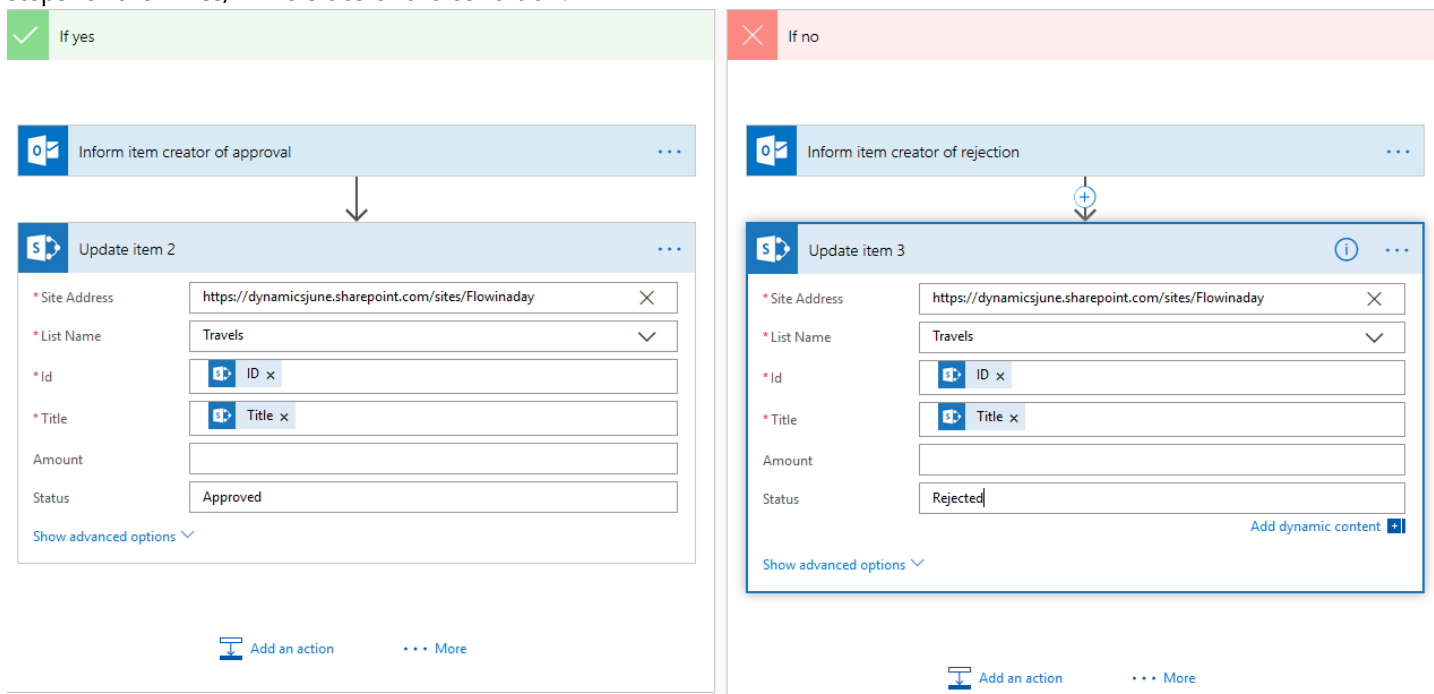
24. We will extend the flow by adding the text **Being approved** when the flow is waiting and **Approved** or **Rejected** when the approval is completed.
25. Click **Edit** in your flow details page
26. After the trigger, add a new **SharePoint Update Item** action and fill it with the following values:

When a new item is created	
↓	
Update item	
* Site Address	https://dynamicsjune.sharepoint.com/sites/Flowinaday
* List Name	Travels
* Id	ID
* Title	Title
Amount	
Status	Being approved
<a href="#">Add dynamic content</a>	
<a href="#">Show advanced options</a>	

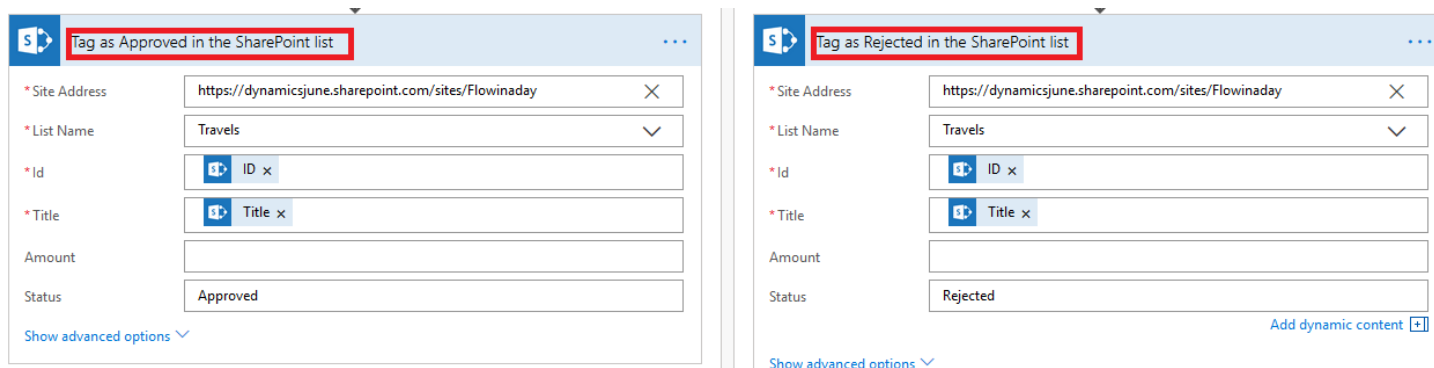
27. The **ID** Dynamic content comes from the trigger and identifies the corresponding SharePoint list item. **Title** also comes from our Trigger and we will provide a value of **Being approved** in the **Status** field:



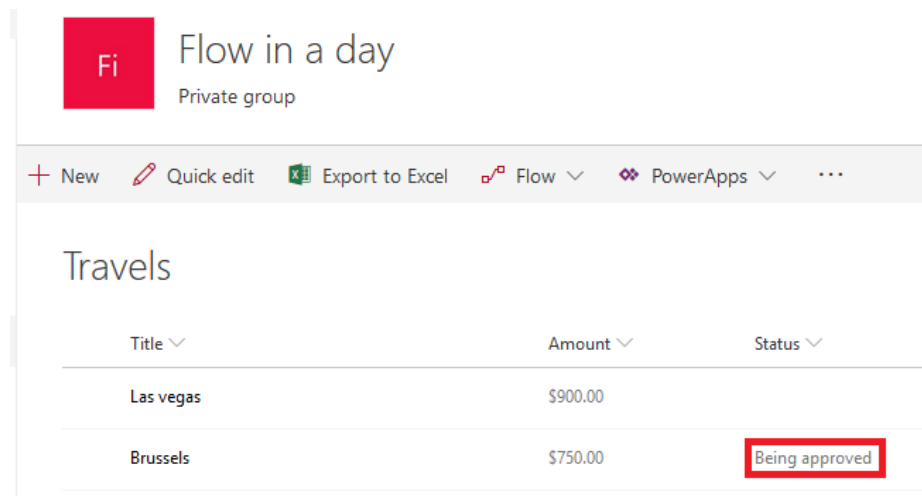
28. Provide similar updates in the **If** yes and **If** no branches, but type **Approved** or **Rejected** in the **Update Item** steps for the If Yes/ If No sides of the condition:



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):



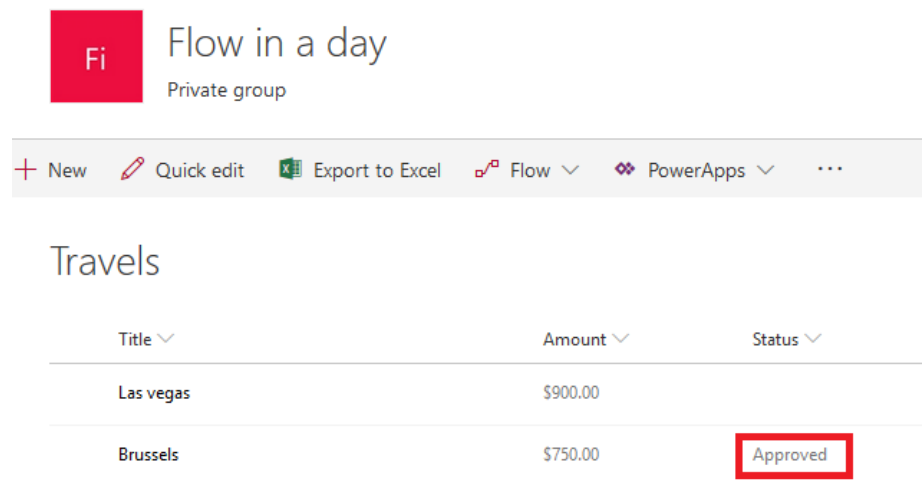
Flow in a day  
Private group

+ New   Quick edit   Export to Excel   Flow   PowerApps   ...

Travels

Title ▾	Amount ▾	Status ▾
Las vegas	\$900.00	
Brussels	\$750.00	Being approved

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



Flow in a day  
Private group

+ New   Quick edit   Export to Excel   Flow   PowerApps   ...

Travels

Title ▾	Amount ▾	Status ▾
Las vegas	\$900.00	
Brussels	\$750.00	Approved

**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.

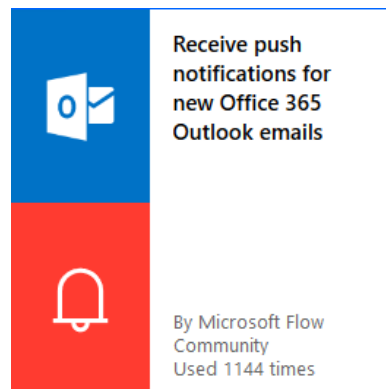
**If you want to take a look at the solution:** navigate to the [Important e-mails notification template](#).

**If you want to start the lab from scratch:**

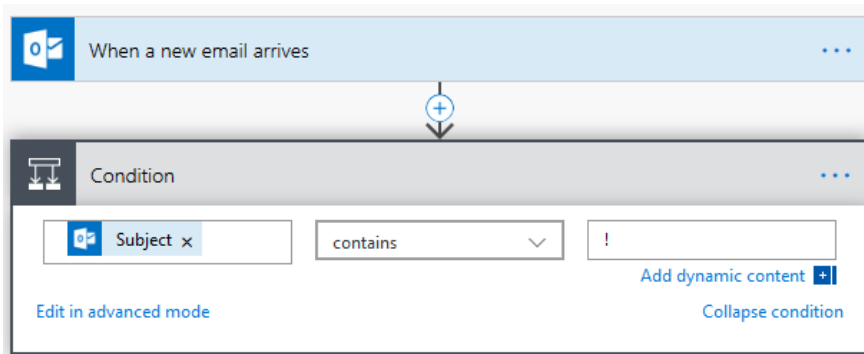
### Tasks:

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](#).

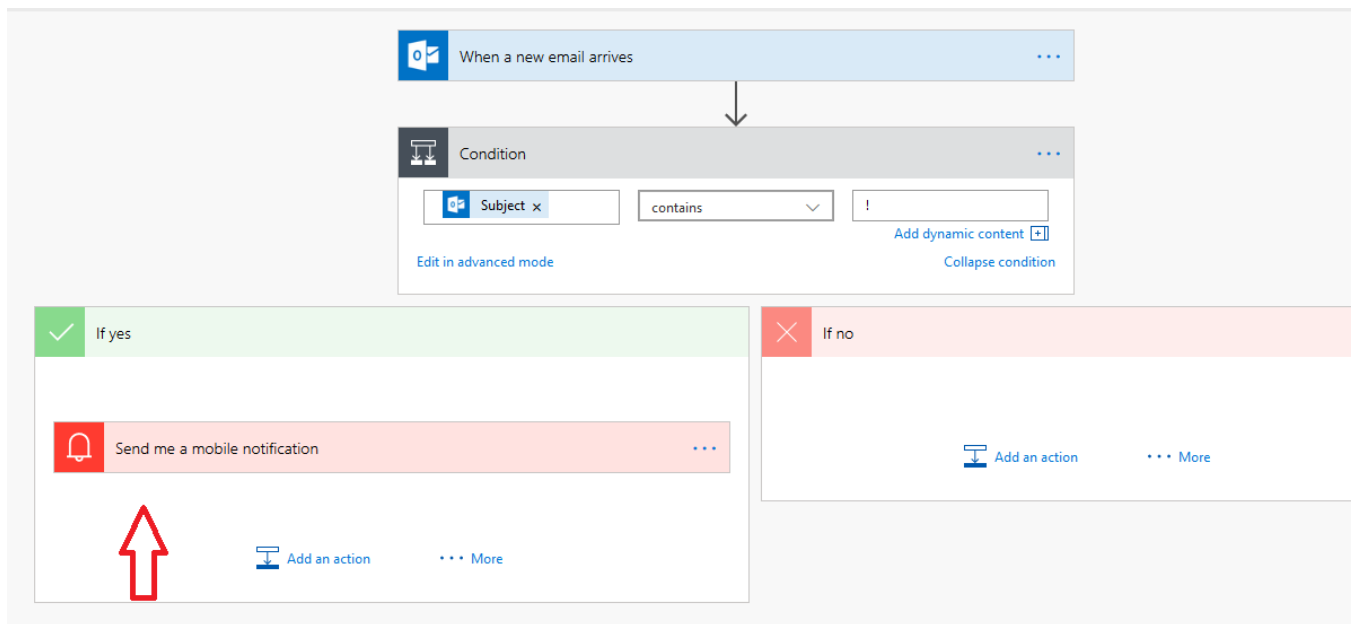
**Note:** If you have not created connections for Notifications, you will need to do so in order to provision the template.



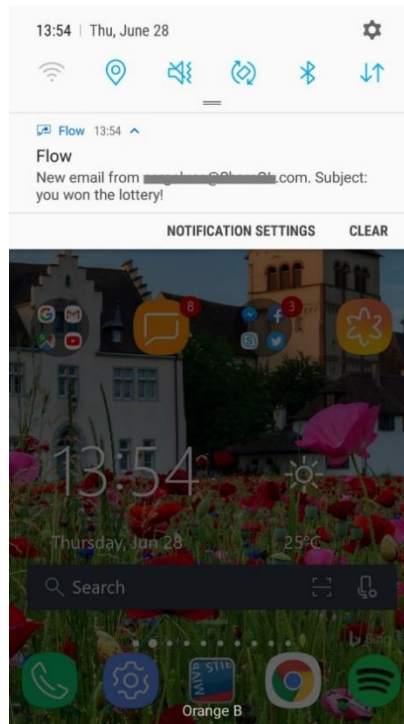
3. Save the flow as **Important e-mails notifications**.
4. Add a condition to check if the mail subject contains !.



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:

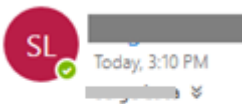
1. In your One Drive for Business, create an **Excel workbook** named **Offices.xlsx**.
2. Add two columns like below with all of the cities and capacities then format as a table with headers:

Contoso Offices	
city	Capacity
London	100
Brussels	250
Seattle	80
Vancouver	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:

## Office Capacity Report



This message was sent with low importance.

The biggest office is : Toronto

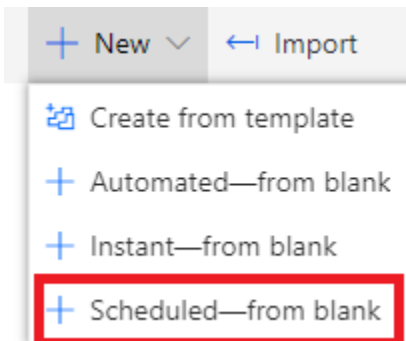
Its capacity is : 400

The total capacity is :

List of offices

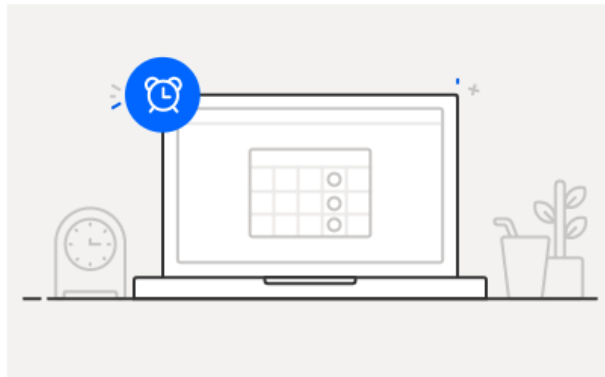
City	Capacity
London	100
Brussels	250
Seattle	80
Vancouver	200
Toronto	400
Antwerpen	15
Warsaw	300
Paris	54
Berlin	70
Amsterdam	60
Montreal	78

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:

### Build a scheduled flow



#### Flow name

Office Capacity

#### Run this flow \*


Starting  at

Repeat every

This flow will run:

Every month

The following flow will be generated:

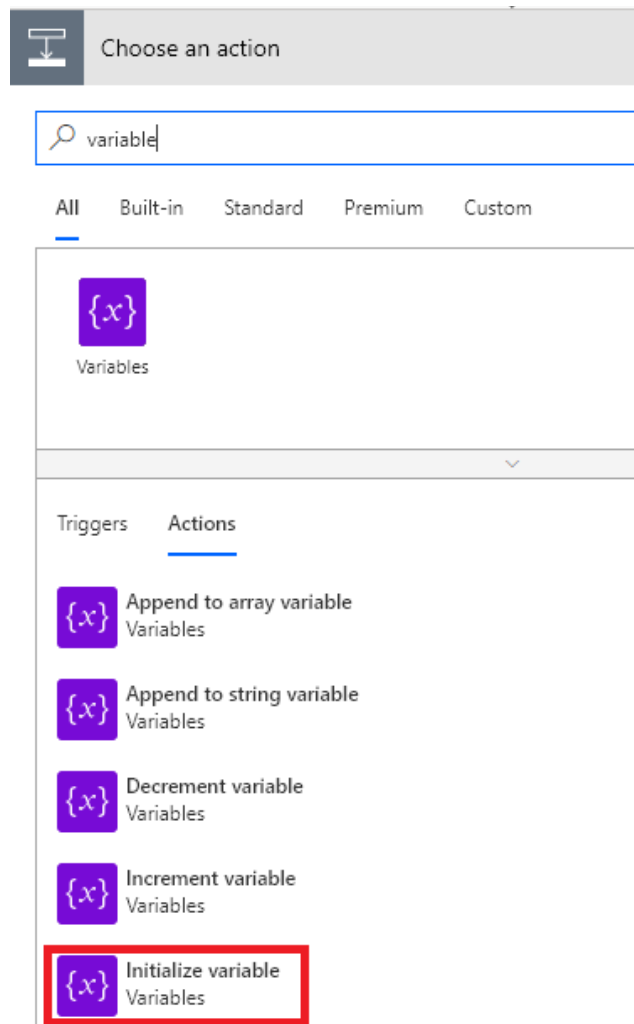
 Recurrence

\* Interval

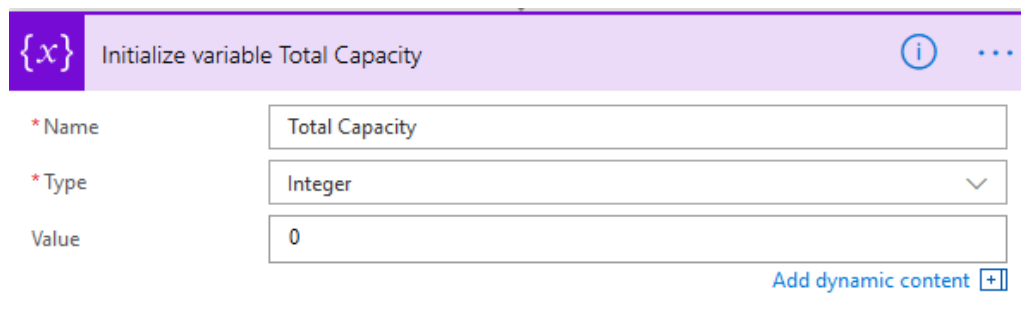
\* Frequency

Show advanced options

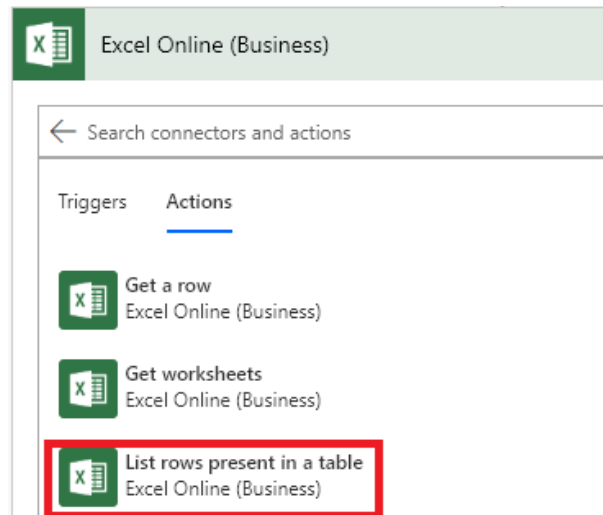
5. The first challenge will be to define the **Total Capacity**. As such, define a variable by adding the **Variable – Initialize variable** action:



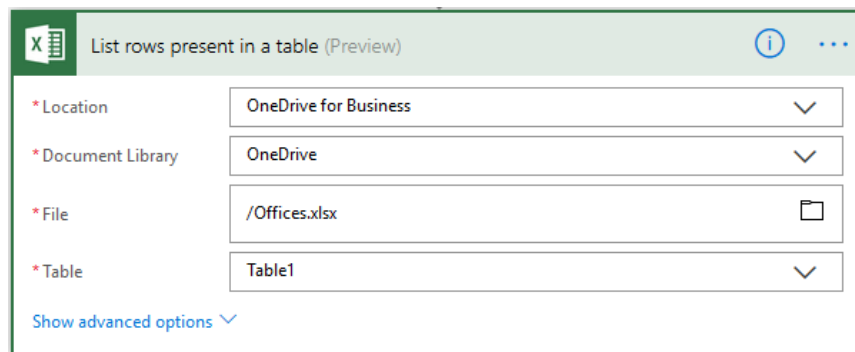
6. Rename this action to be **Initialize variable Total Capacity**, set the variable Name **Total Capacity**, and select **Integer** as the type and an initial **Value** of **0**:



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**:

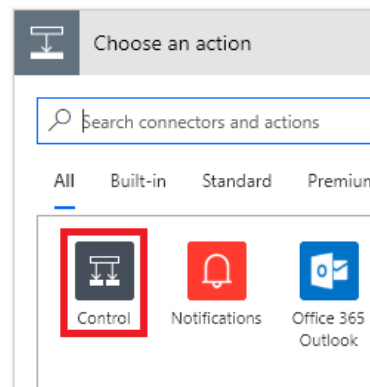


10. Set its properties according to the next picture:



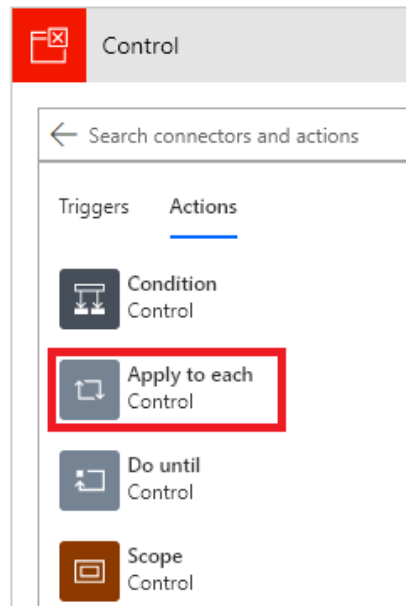
14. Add an **Apply to each** (from the Control connector) after the Excel action:

a. Click Control

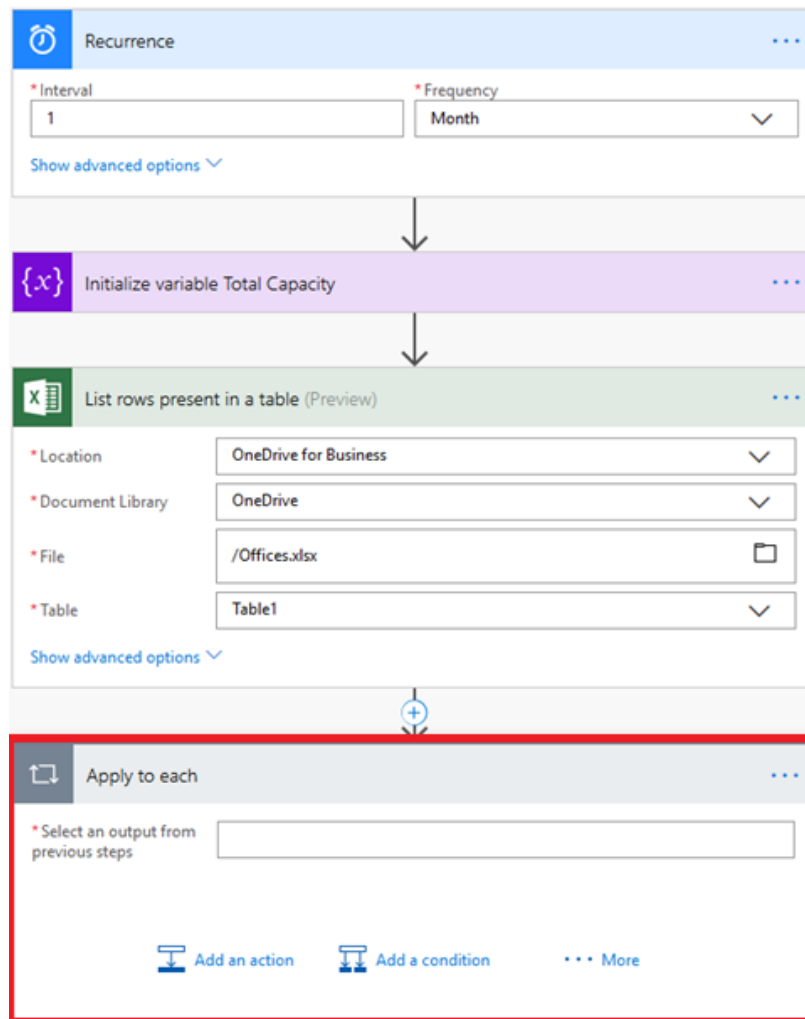


b. Click **Apply to each** :

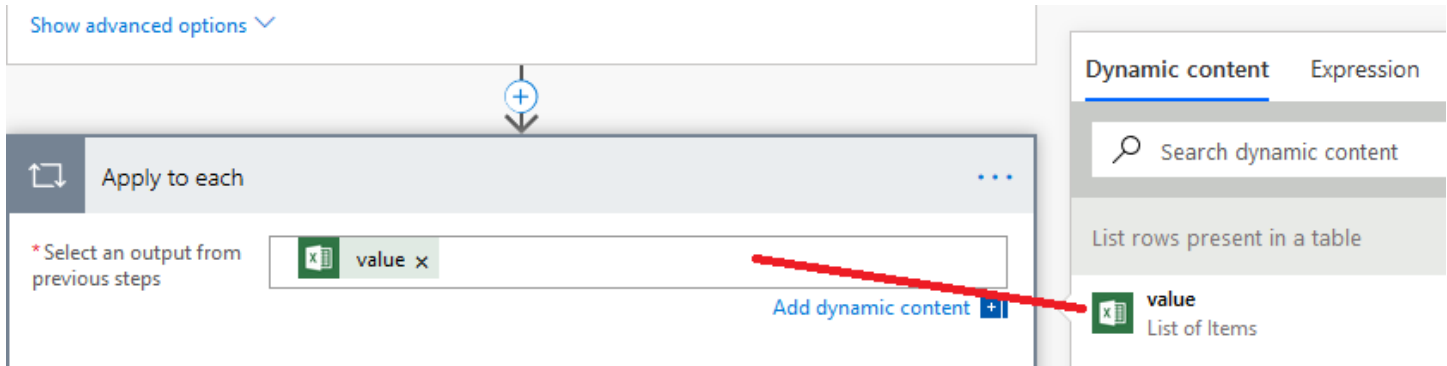




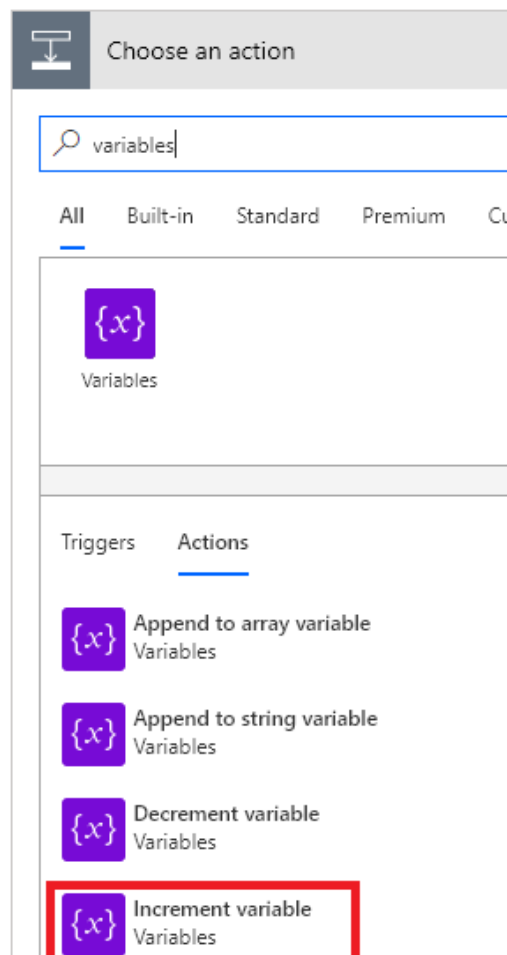
15. Your flow should now look like this:



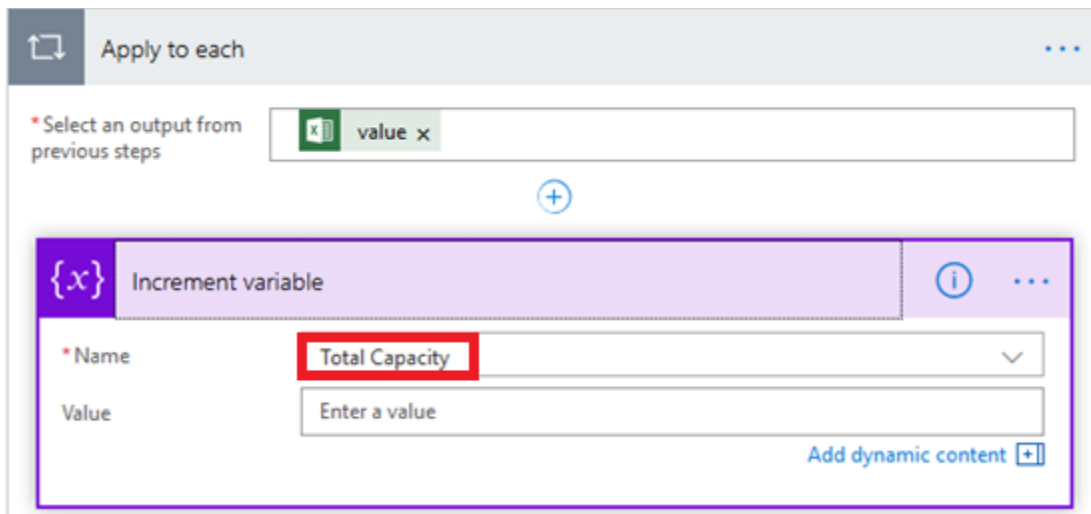
11. The **Apply to each** action expects a list of values, use the Dynamic content value of the **List rows present in a table** action:



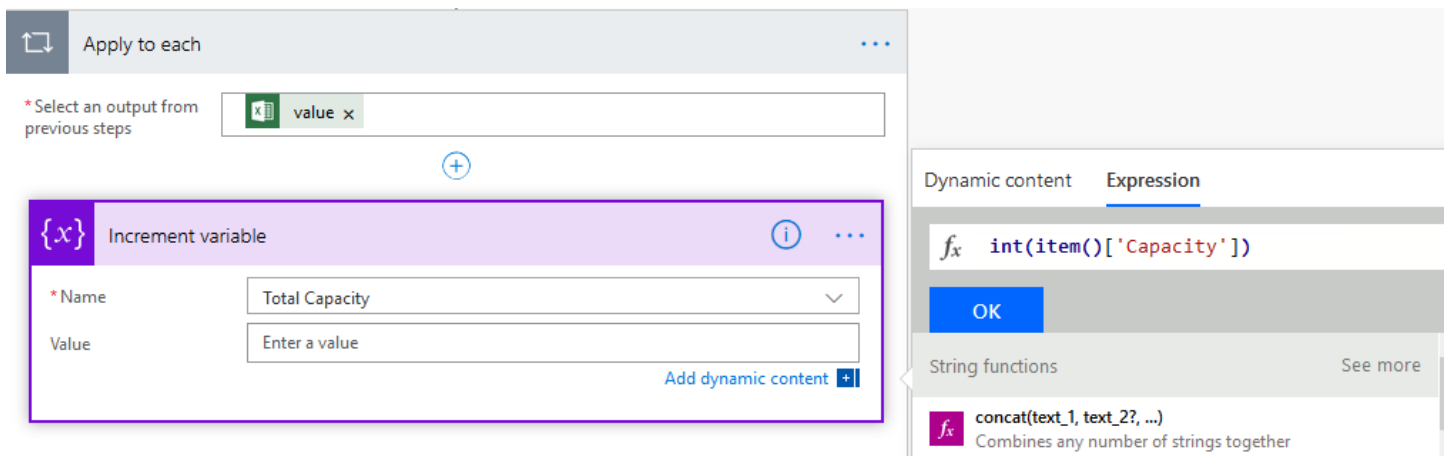
12. In the **Apply to each** action, click **Add an action**. We will compute the total capacity, so add the action **Variables – Increment variable** action:



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**:

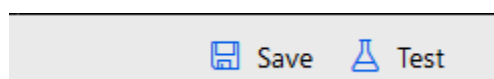


**Note:**

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:**

Test Flow


☒ I'll perform the trigger action


☐ Using data from previous runs  
Choose data from a list of previous runs:

**Test** Cancel

17. Click on **Run flow:**

Run flow


 Office Capacity  
Owners: serge Luca


See details 



This flow doesn't need additional input to run



**Run flow** Cancel



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



 Your flow ran successfully.





 Recurrence 0s 

 Initialize variable Total Capacity 0s 

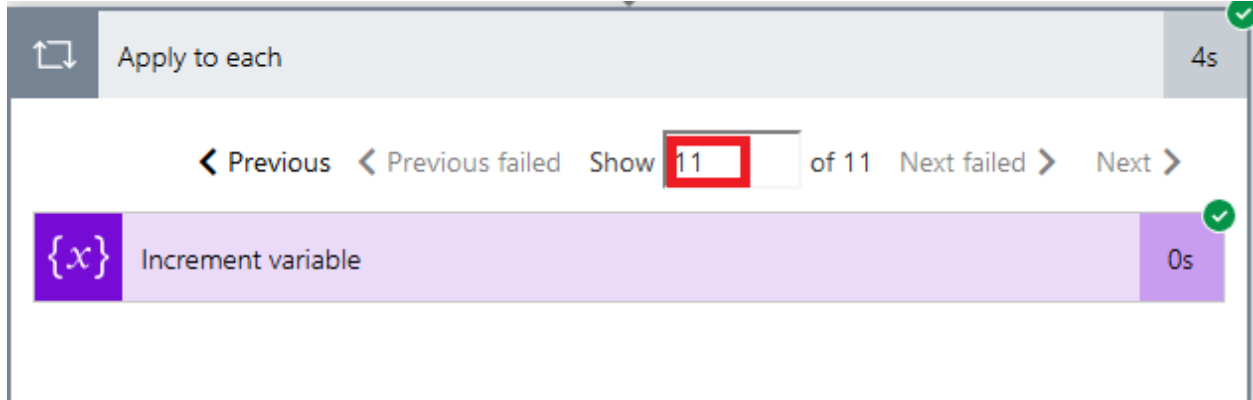
 List rows present in a table 3s 

 Apply to each 4s 

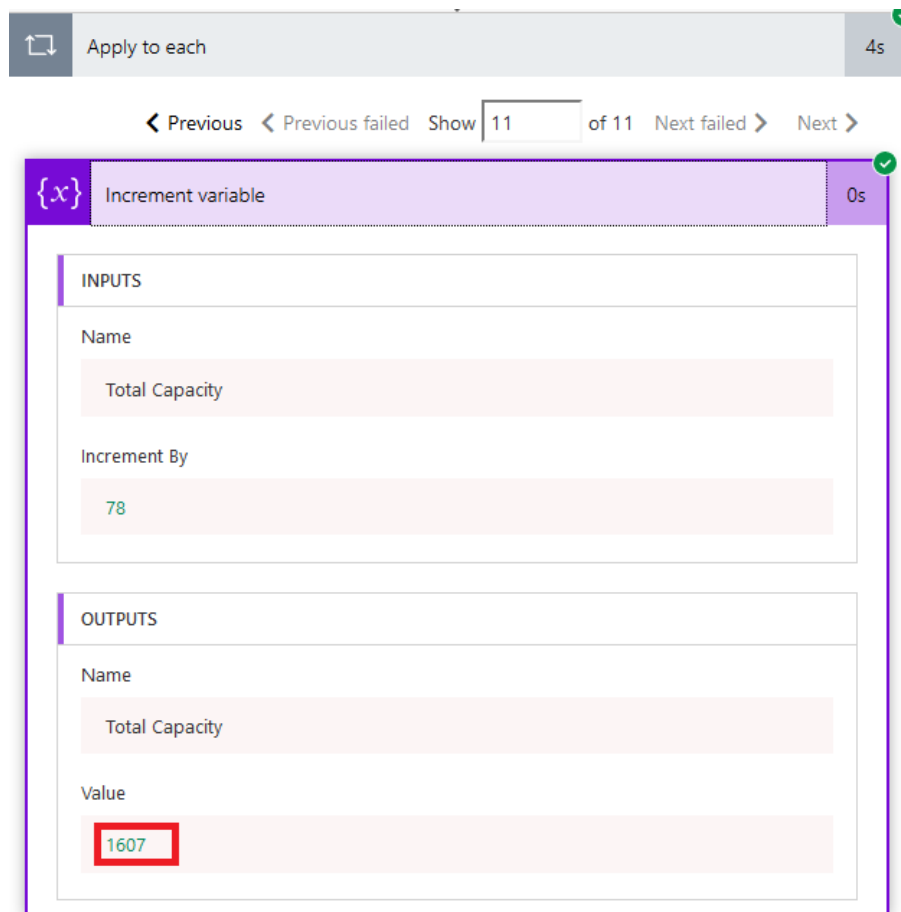
< Previous < Previous failed Show 1 of 11 Next failed > Next >

 Increment variable 0s 

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)

{x} Initialize variable Total Capacity

{x} Initialize variable Bigger office

\*Name Bigger office

\*Type String

Value Enter initial value

{x} Initialize variable MaxCapacity

\*Name MaxCapacity

\*Type Integer

Value Enter initial value

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:

{x} Increment variable

Condition

Choose a value is equal to

Add dynamic content +

+ Add

Dynamic content Expression

$f_x$  int(item()['Capacity'])

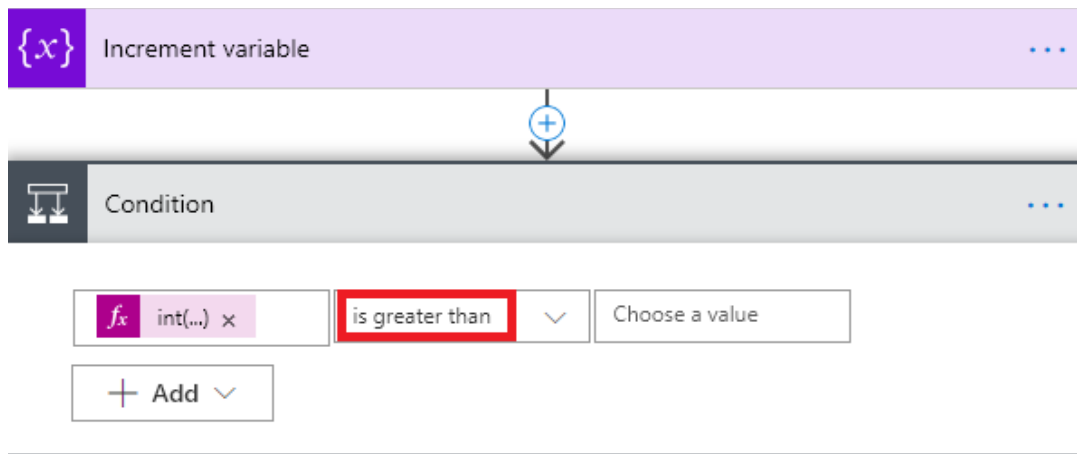
OK

String functions

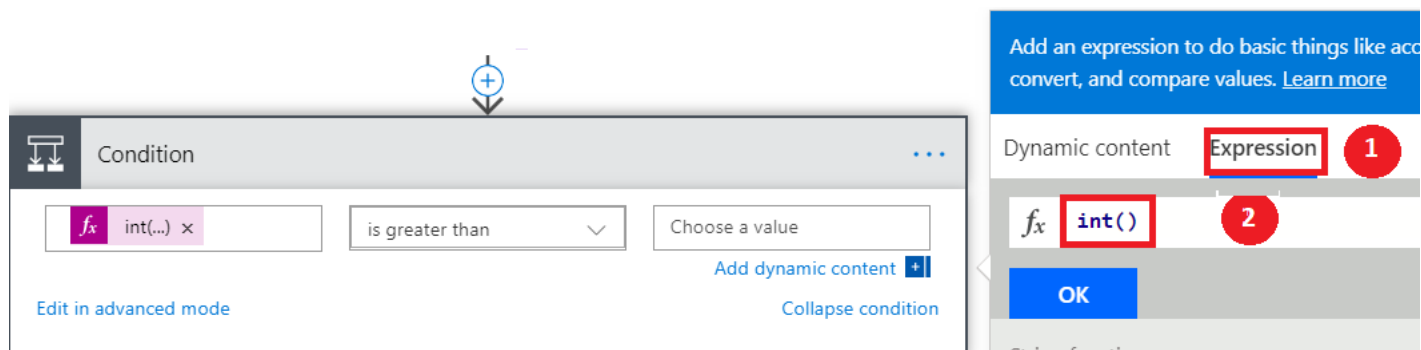
$f_x$  concat(text\_1, text\_2?, ...)  
Combines any number of strings together

24. Click **Ok**.

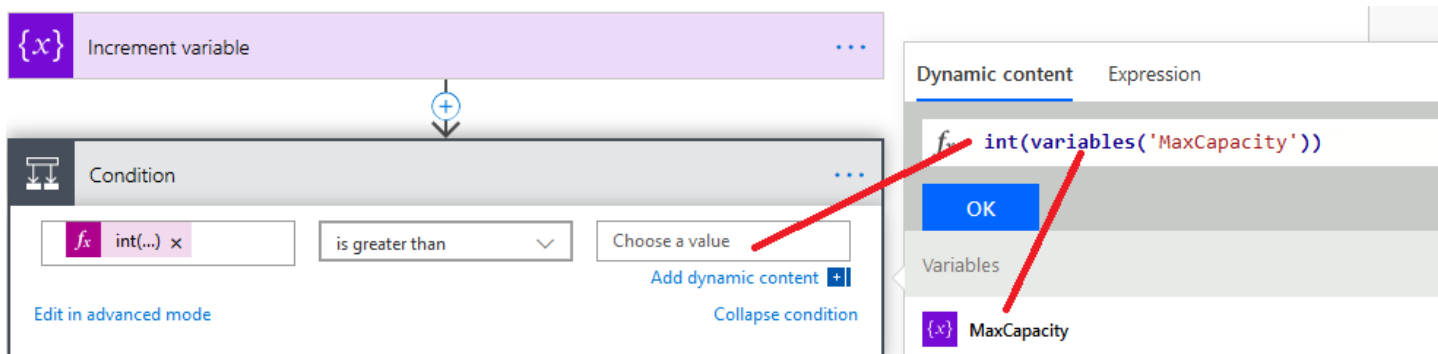
25. Select the comparison operator **is greater than**:



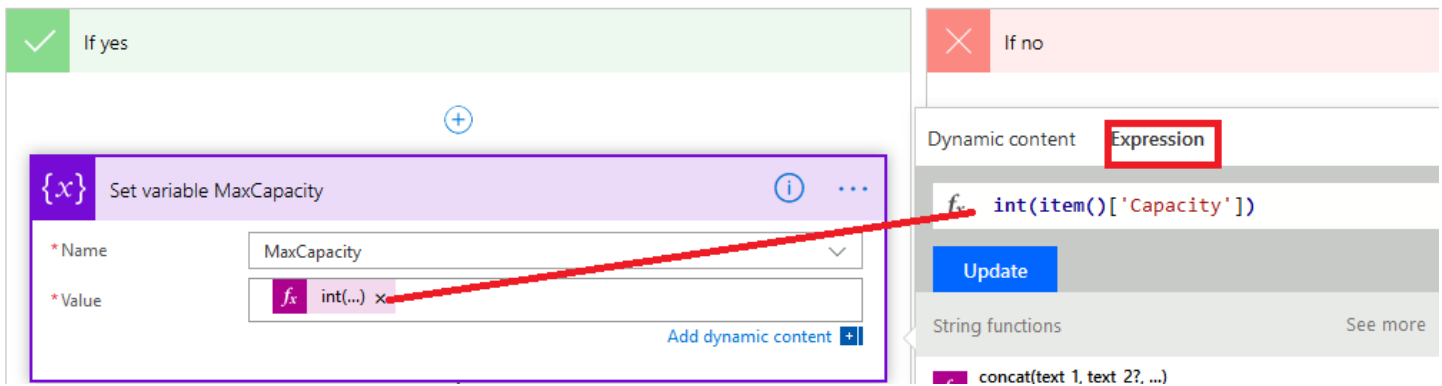
26. In the **Choose a value** textbox, we will include an expression much like we did before by using the **int()** expression, but with a small variation. Click choose a value, click on **Expression** and type **int()**



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.

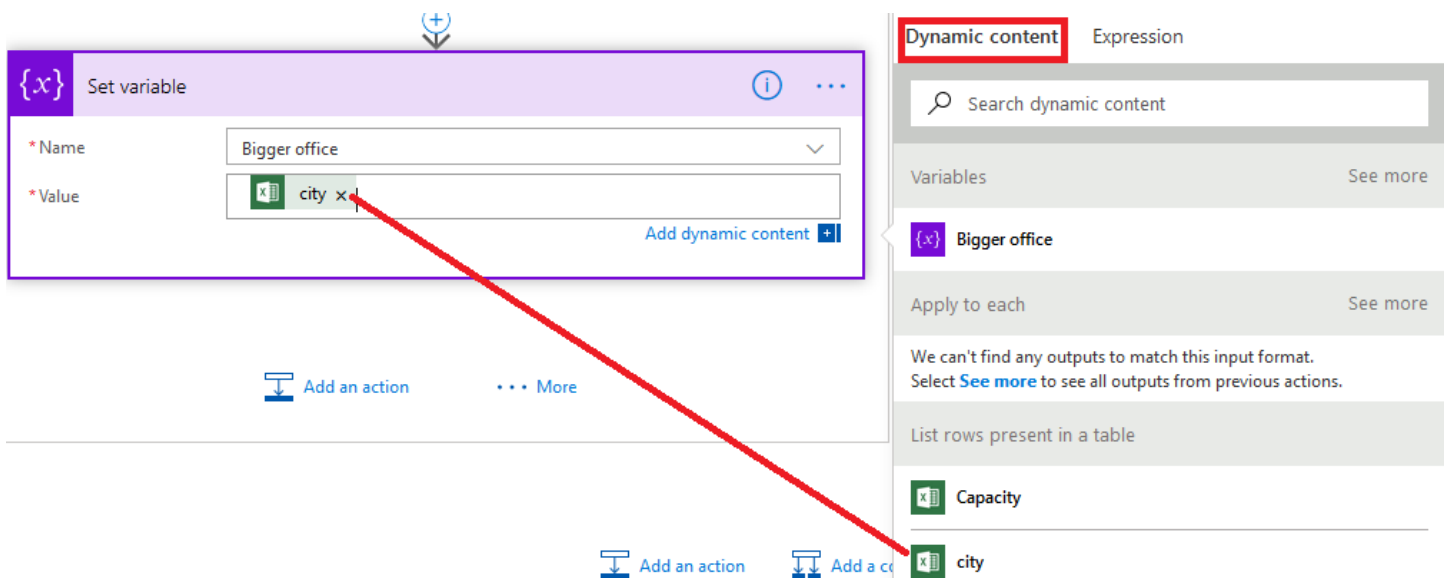


29. Now, in the left **If yes** branch, add a new action **Variables – Set variable** for our **MaxCapacity** variable and in the **Expression** panel type **int(item()['Capacity'])** as illustrated in the next picture.



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**.

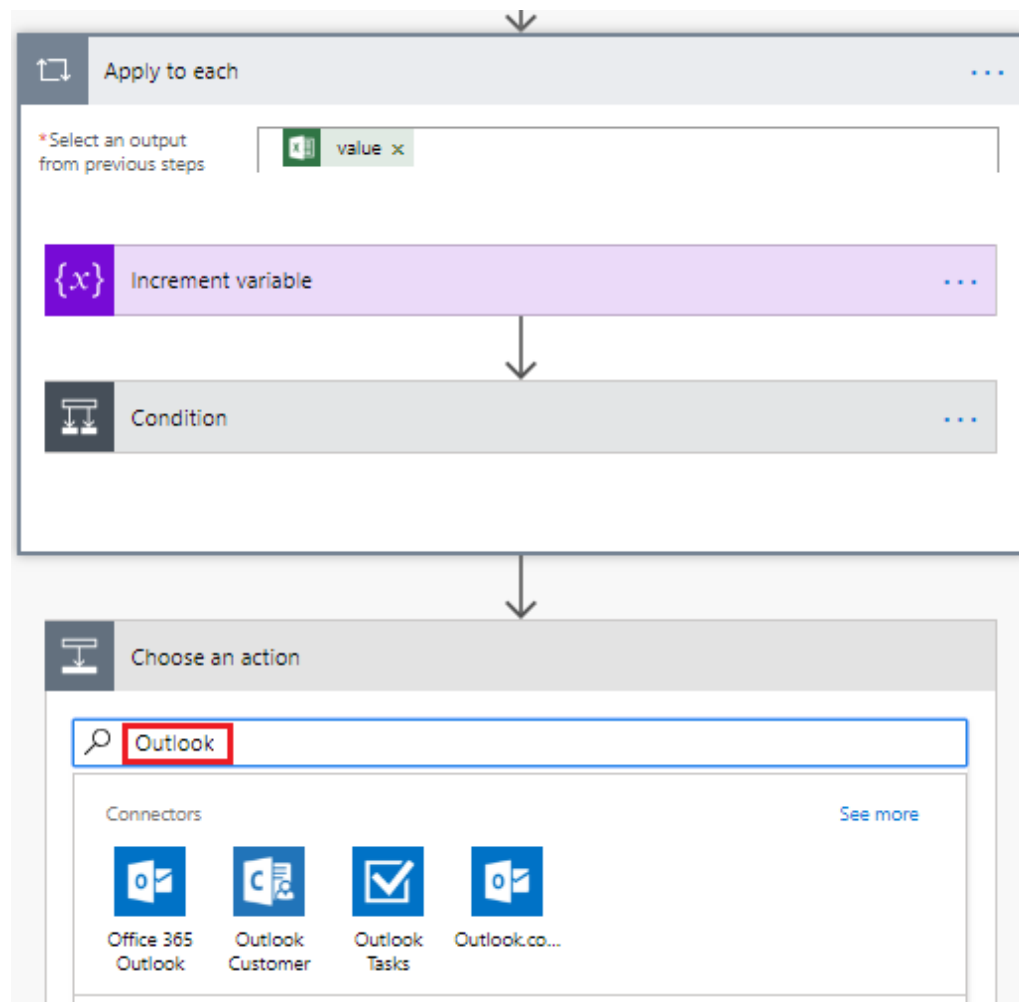


32. Save and test the flow to figure out which city has the bigger capacity (Toronto in our case). You can debug the flow or add a notification (or send an e-mail to yourself).

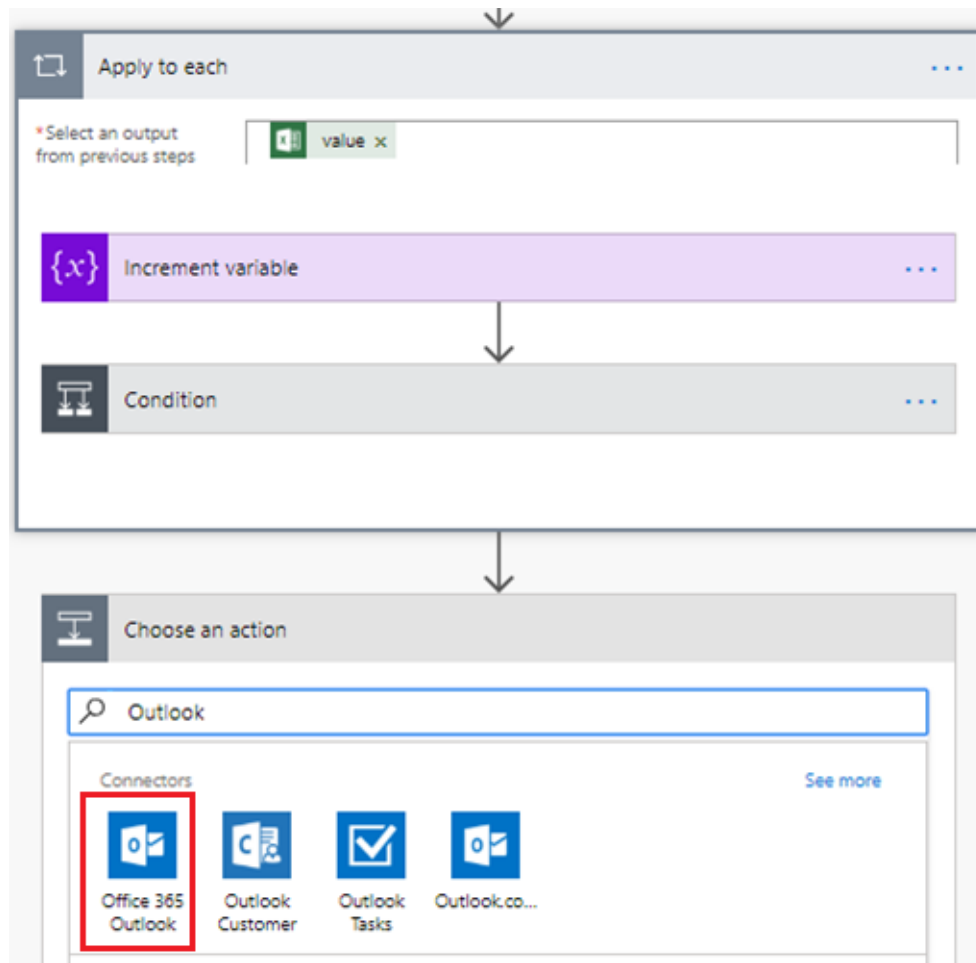
33. Next, let's send an e-mail by adding an **Outlook 365 Outlook - Send an e mail** action **after the Apply to each**:

- Find the action by typing Outlook:

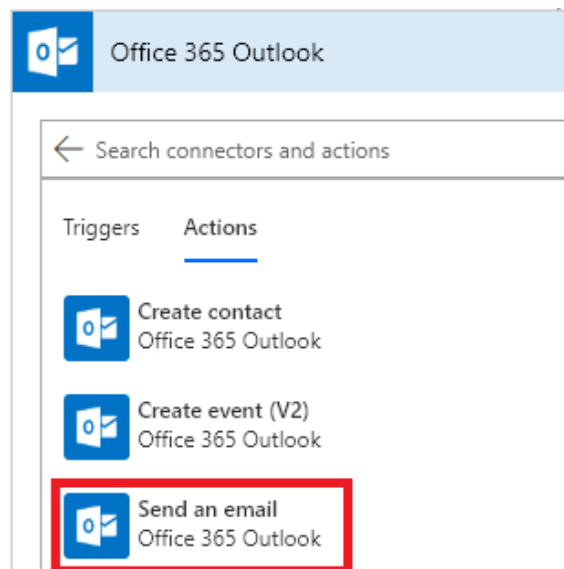




- b. In the Connectors list click **Office 365 Outlook**:

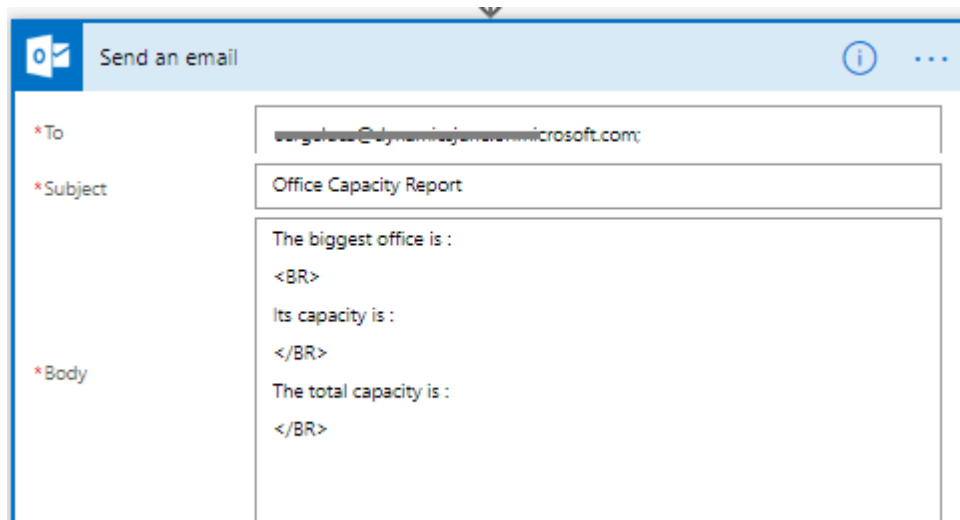


- c. Select the action **Office 365 Outlook – Send an email**:



- 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:



Send an email

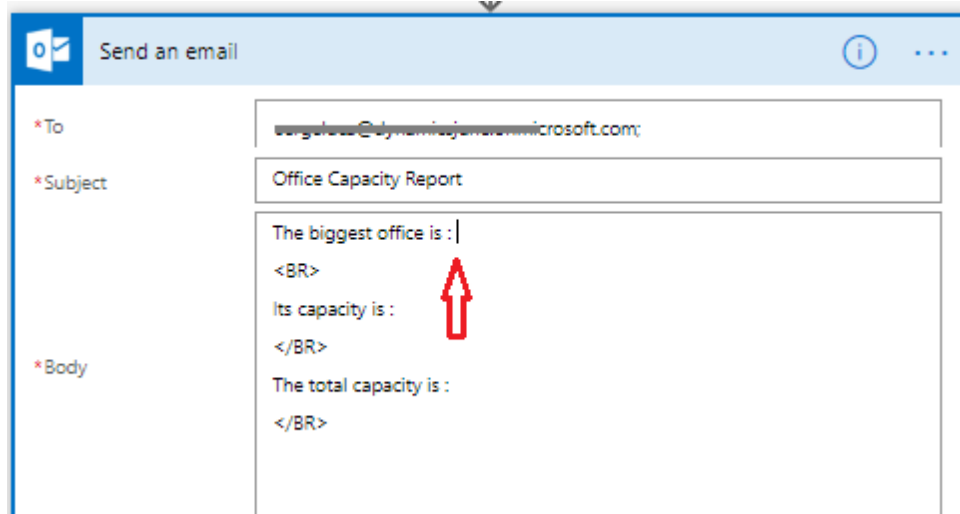
\*To: varghe...@dynamicjunction.microsoft.com;

\*Subject: Office Capacity Report

\*Body: The biggest office is :  
<BR>  
Its capacity is :  
</BR>  
The total capacity is :  
</BR>

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:*



Send an email

\*To: varghe...@dynamicjunction.microsoft.com;

\*Subject: Office Capacity Report

\*Body: The biggest office is :  
<BR>  
Its capacity is :  
</BR>  
The total capacity is :  
</BR>

- vi. The Dynamic content panel should show up:

**Send an email**

\*To:

\*Subject: Office Capacity Report

\*Body:
   
The biggest office is :
   
<BR>
   
Its capacity is :
   
</BR>
   
The total capacity is :
   
</BR>
   
[Add dynamic content](#)

From (Send as):

CC:

**Dynamic content** Expression

Variables

**Bigger office**

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

Send an email

\* To

\* Subject

Office Capacity Report

\* Body

The biggest office is :  
<BR>  
Its capacity is :  
</BR>  
The total capacity is :  
</BR>

From (Send as)

Email address to send mail from (requires "Send as" or "Send on behalf of" pe

CC

Specify email addresses separated by semicolons like someone@contoso.com

Add dynamic content

Add dynamic content from the apps and c  
used in this flow.

Dynamic content

Expression

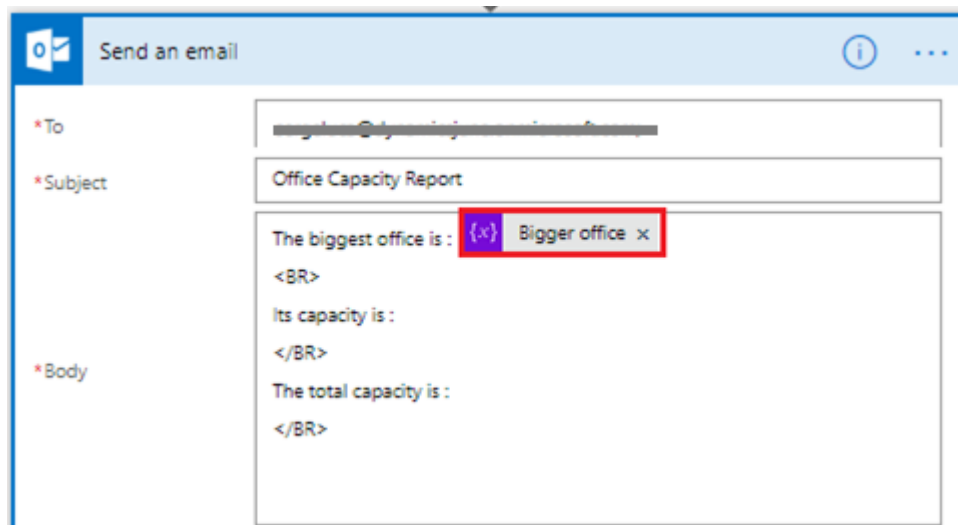
Search dynamic content

Variables

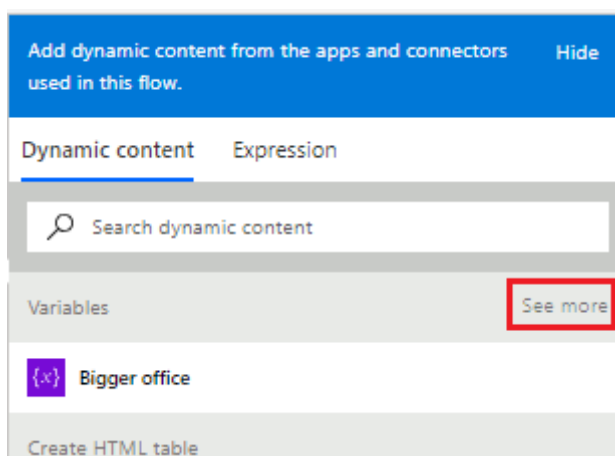
(x)

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:

Send an email

\*To: benjamin.d@dynamicjunction.com

\*Subject: Office Capacity Report

\*Body:

The biggest office is : {x} Bigger office x  
<BR>  
Its capacity is : {x} MaxCapacity x  
</BR>  
The total capacity is : {x} Total Capacity x  
</BR>  
|

Add dynamic content +

Show advanced options v

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

Send an email

\*To: benjamin.d@dynamicjunction.com

\*Subject: Office Capacity Report

\*Body:

The biggest office is : {x} Bigger office x  
<BR>  
Its capacity is : {x} MaxCapacity x  
</BR>  
The total capacity is : {x} Total Capacity x  
</BR>

Show advanced options v

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

**Send an email**

\*To: [Redacted]

\*Subject: Office Capacity Report

\*Body: The biggest office is : {x} Bigger office x  
<BR>Its capacity is : {x} MaxCapacity x</BR>The total capacity is : {x} Total Capacity x</BR>

From (Send as): Email address to send mail from (requires "Send as" or "Send on behalf of" permissions)

CC: Specify email addresses separated by semicolons like someone@contoso.com

BCC: Specify email addresses separated by semicolons like someone@contoso.com

Attachments Name - 1: Attachment name

Attachments Content - 1: Attachment content

+ Add new item

Importance: Importance

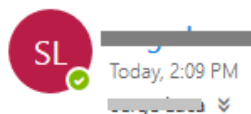
Is HTML: Yes

Hide advanced options ^

34. Save your flow and test it.

35. Check your e-mail, you should receive something like this:

## Office Capacity Report



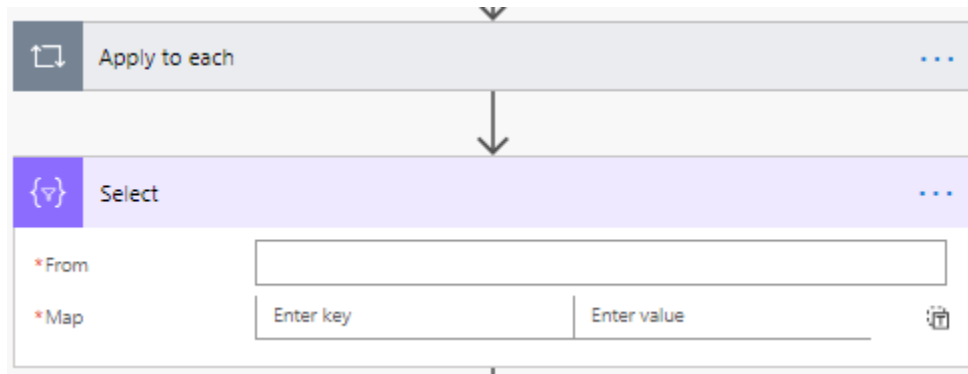
Inbox; Sent Items

This message was sent with low importance.

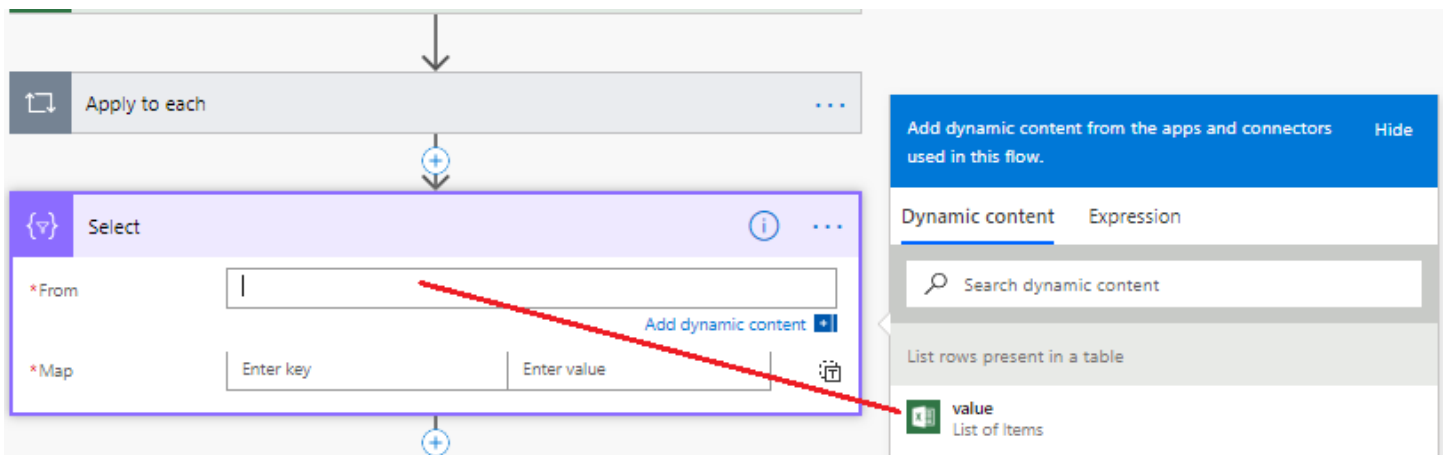
The biggest office is : Toronto  
Its capacity is : 400  
The total capacity is :1607

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.

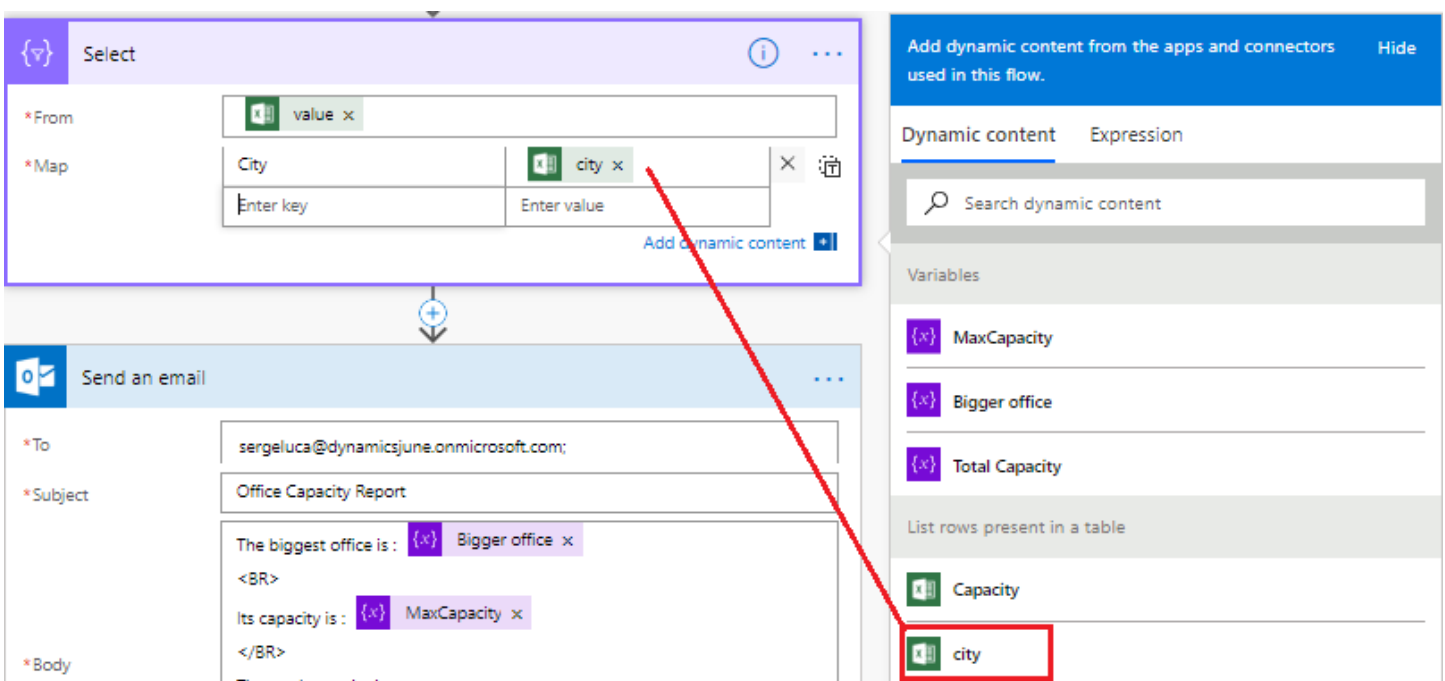
37. Let's define the list formatting logic. Before the **Send an email action** add a **Data Operations – Select** action:



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**:

The screenshot shows a Microsoft Flow canvas with two actions: 'Select' and 'Send an email'.

The 'Select' action has the following configuration:

- From:** value x
- Map:**

City	city x
Capacity	Capacity x
Enter key	Enter value

The 'Send an email' action has the following configuration:

- To:** sergeluca@dynamicjsune.onmicrosoft.com;
- Subject:** Office Capacity Report
- Body:**

The biggest office is : {x} Bigger office x

<BR>

Its capacity is : {x} MaxCapacity x

</BR>

A red arrow points from the 'Capacity' field in the 'Select' action's 'Map' section to the 'Dynamic content' panel on the right. The 'Dynamic content' panel shows a search bar and a list of variables: MaxCapacity, Bigger office, Total Capacity, Capacity, and city.

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

The screenshot shows a Microsoft Flow canvas with two actions: 'Select' and 'Create HTML table'.

The 'Select' action has the following configuration:

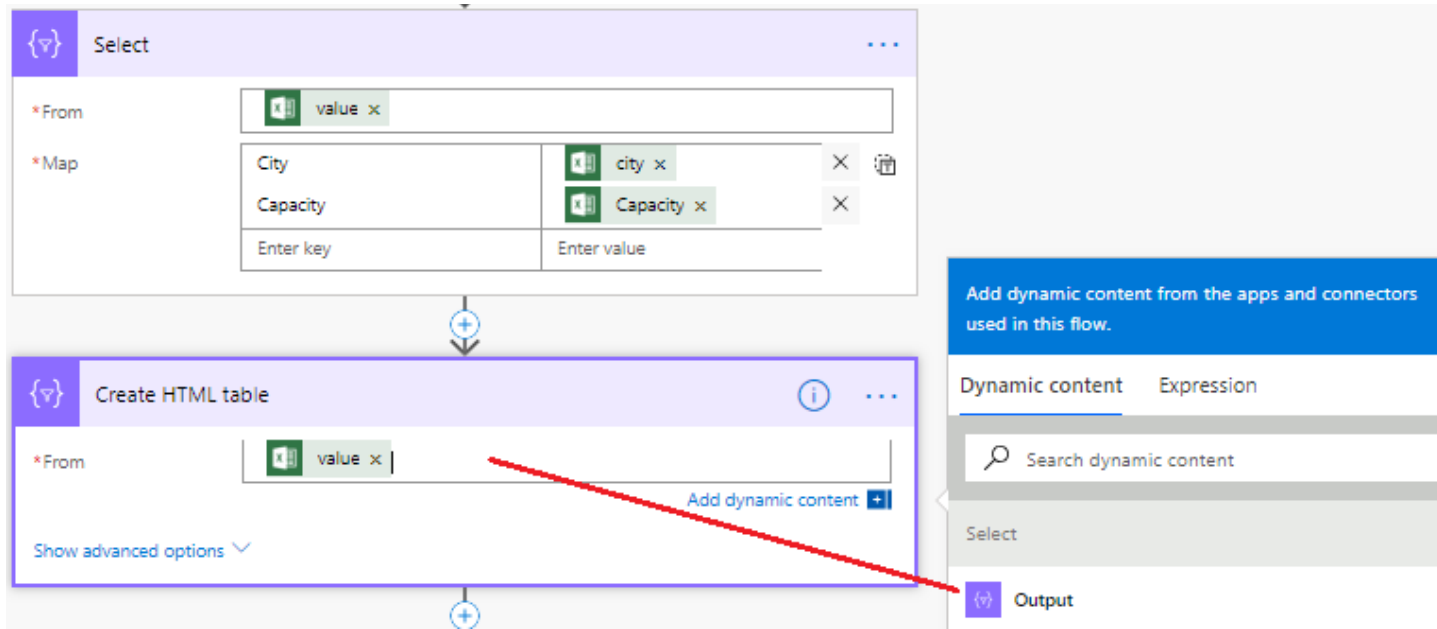
- From:** value x
- Map:**

City	city x
Capacity	Capacity x
Enter key	Enter value

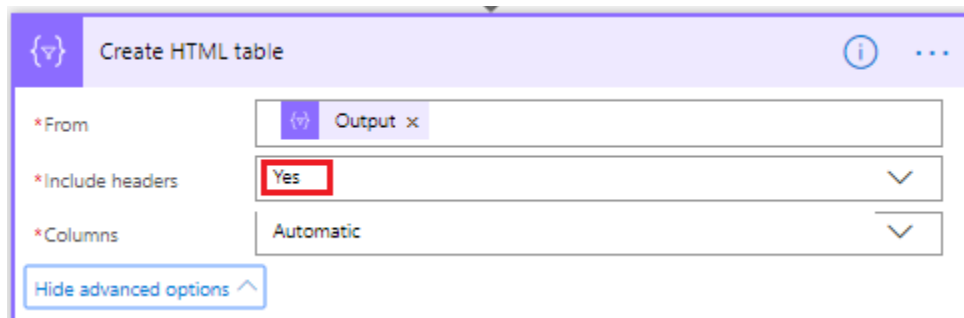
The 'Create HTML table' action is highlighted with a red box and has the following configuration:

- From:** |

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 :

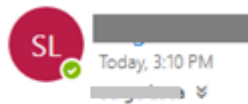
The screenshot shows the 'Send an email' step configuration in Microsoft Flow. The 'To' field is set to 'sergeluca@ynamicsjune.onmicrosoft.com', the 'Subject' is 'Office Capacity Report', and the 'Body' contains the following HTML-formatted text:

The biggest office is : {x} Bigger office x  
<BR>  
Its capacity is : {x} MaxCapacity x  
</BR>  
The total capacity is :  
</BR>  
<u>List of offices</u>  
{x} Output x

A red arrow points from the 'Output' dynamic content placeholder in the body to the 'Output' variable in the 'Dynamic content' pane on the right. The 'Dynamic content' pane shows a search bar and a list of variables: 'Bigger office' and 'Output'.

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

## Office Capacity Report



This message was sent with low importance.

The biggest office is : Toronto  
 Its capacity is : 400  
 The total capacity is :

### List of offices

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

**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:

The screenshot shows a workflow step titled "Initialize variable Max Office". The step is highlighted with a purple border. It contains the following fields:

- Name:** Max Office
- Type:** Object
- Value:**

```
{
  "Office Capacity": 0,
  "Office Name": ""
}
```

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

The screenshot shows a Microsoft Flow workflow. The first step is 'Increment variable Total Capacity'. Below it is a 'Condition' step. The condition is set to 'int(...) is greater than Choose a value'. A red line points from 'Choose a value' to an expression picker. The expression picker shows the expression `variables('Max Office')['Office Capacity']`. The picker also has tabs for 'Dynamic content' and 'Expression', and an 'OK' button.

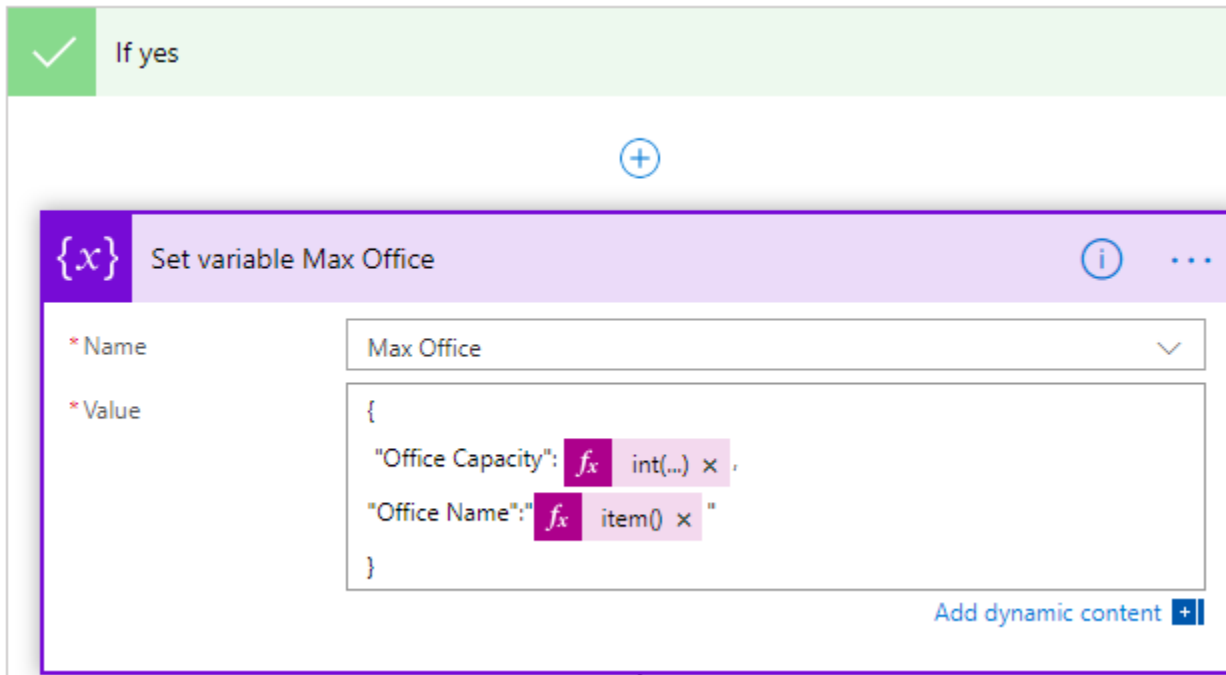
48. In the condition, create a Set variable that will update **Max Office** :

The screenshot shows the 'If yes' branch of a condition. The action is 'Set variable Max Office'. The variable name is 'Max Office'. The value is a JSON object: `{ "Office Capacity": , "Office Name": "" }`. A red line points from the comma after 'Office Capacity' to an expression picker. The expression picker shows the expression `int(item()['Capacity'])`. The picker also has tabs for 'Dynamic content' and 'Expression', and an 'OK' button.

The screenshot shows the 'If yes' branch of a condition. The action is 'Set variable Max Office'. The variable name is 'Max Office'. The value is a JSON object: `{ "Office Capacity": int(...) , "Office Name": "" }`. A red line points from the comma after 'Office Capacity' to an expression picker. The expression picker shows the expression `item()['city']`. The picker also has tabs for 'Dynamic content' and 'Expression', and an 'OK' button.

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

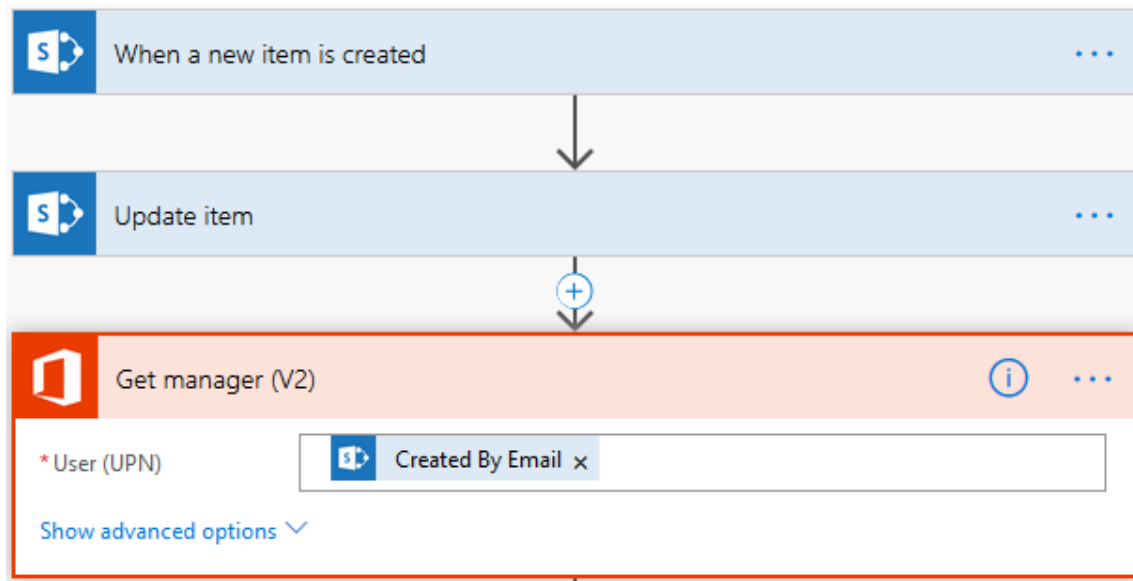
**Prerequisites:** The lab Approval (part 1) must be completed.

**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:



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

The screenshot shows the flow editor with the 'Dynamic content' pane open on the right. The 'When a new item is created' action is selected, and the 'Created By Email' dynamic property is highlighted with a red box. A red arrow points from this property to the 'User (UPN)' field in the 'Get manager (V2)' action.

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

The screenshot shows the 'Start an approval' step configuration in Microsoft Flow. The 'Assigned To' field is set to 'Mail'. The 'Details' field is set to 'Created By Dis...' at 'Created'. The 'Item Link' field is set to 'Link to item'. The 'Item Link Description' field is set to 'Name'. A red arrow points from the 'Add dynamic content' button to the 'Dynamic content' panel on the right, which shows 'Job Title' and 'Mail' as available content.

**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.

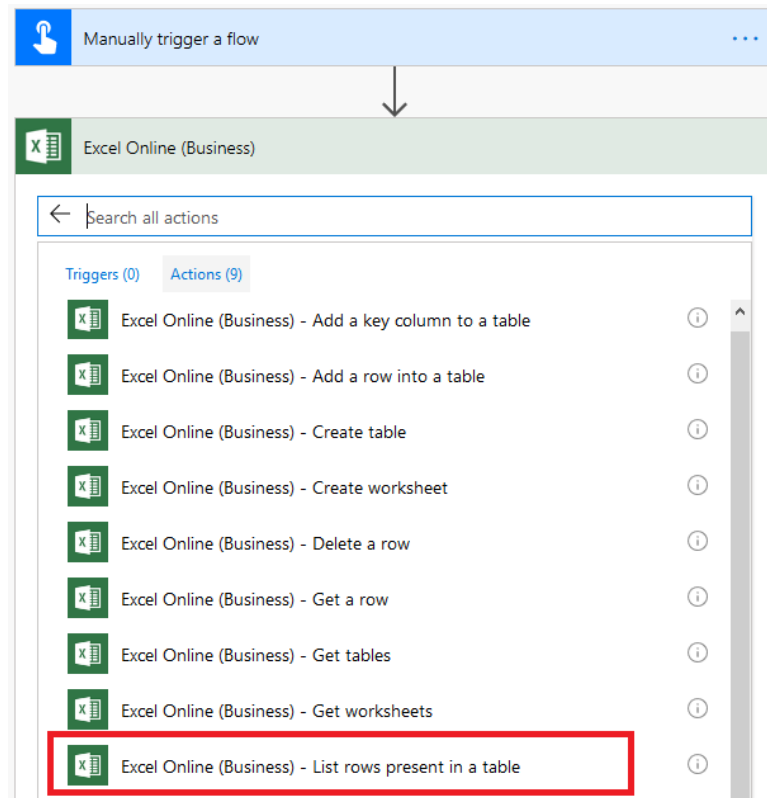
**If you want to start the lab from scratch:**

### Tasks :

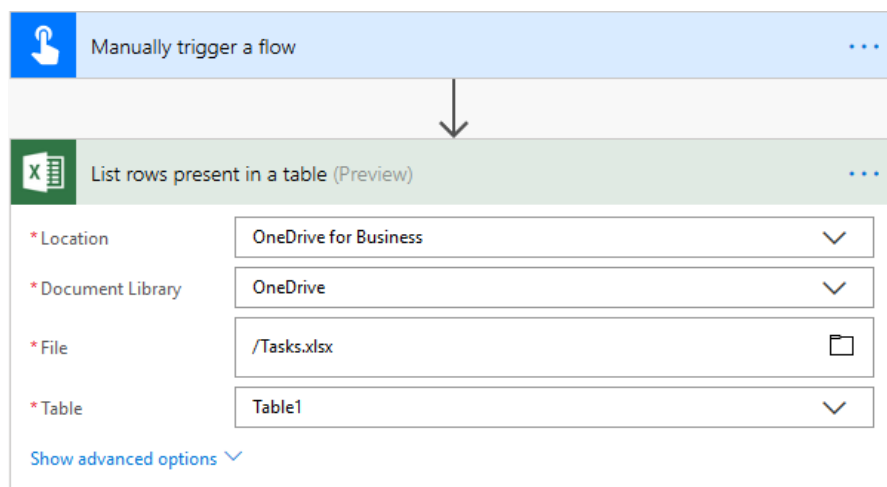
1. Create a new Excel file in your OneDrive for Business that looks like this (use the same columns). Before adding data in the Deadline column, make sure it is in **Text** format.

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

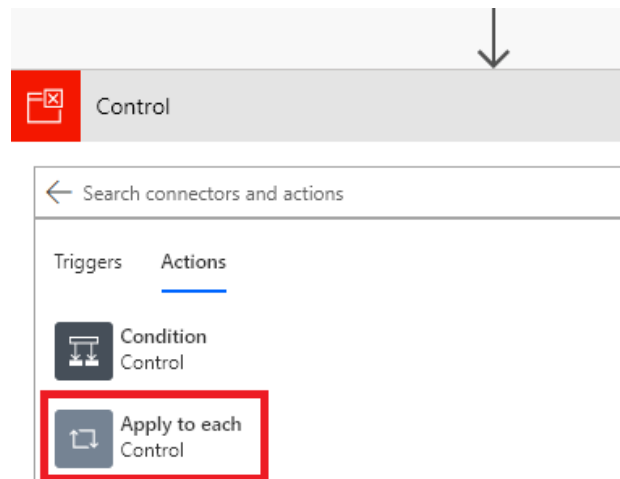
2. Name it **Tasks.xlsx**
3. Change the value of the **in charge column** to your e-mail address and adjust some deadline value.
4. We want to write a flow that will loop through all tasks and that will check if the task is overdue. To do so, **Create a new Instant flow from blank** and use a **Microsoft 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:



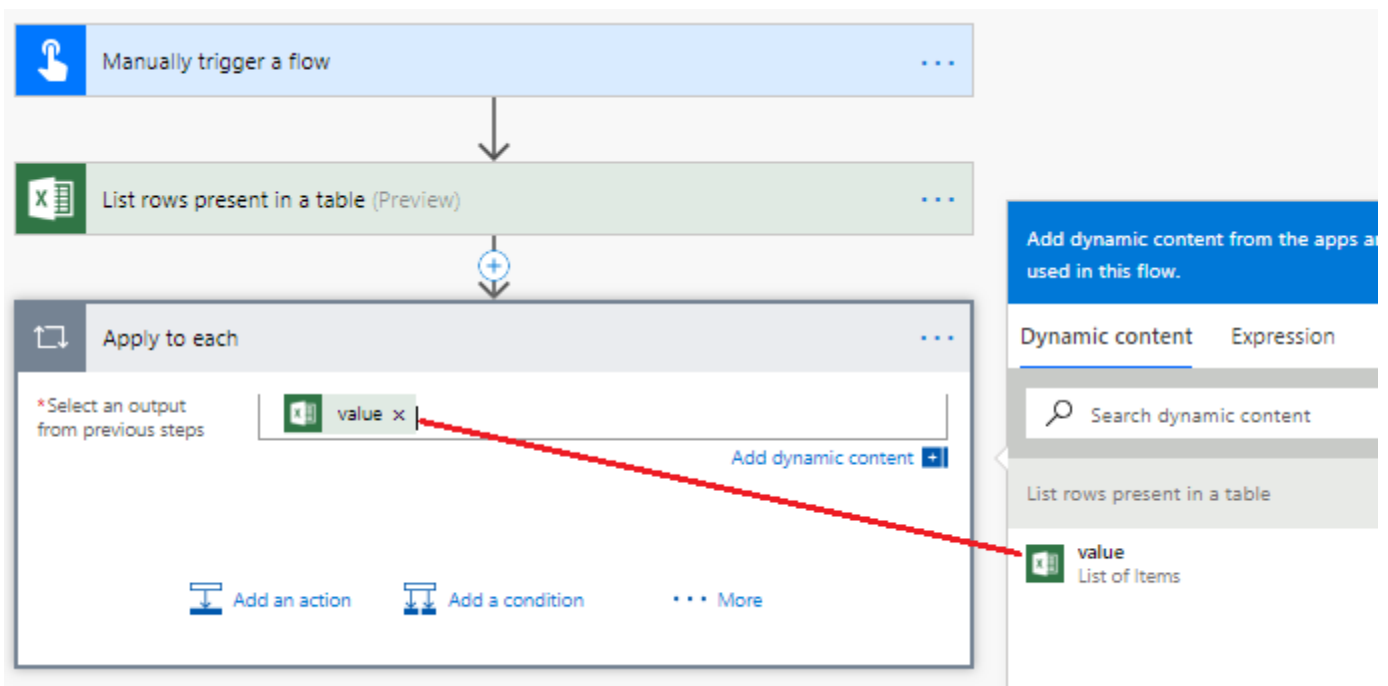
6. Set the action properties as illustrated in the next picture:



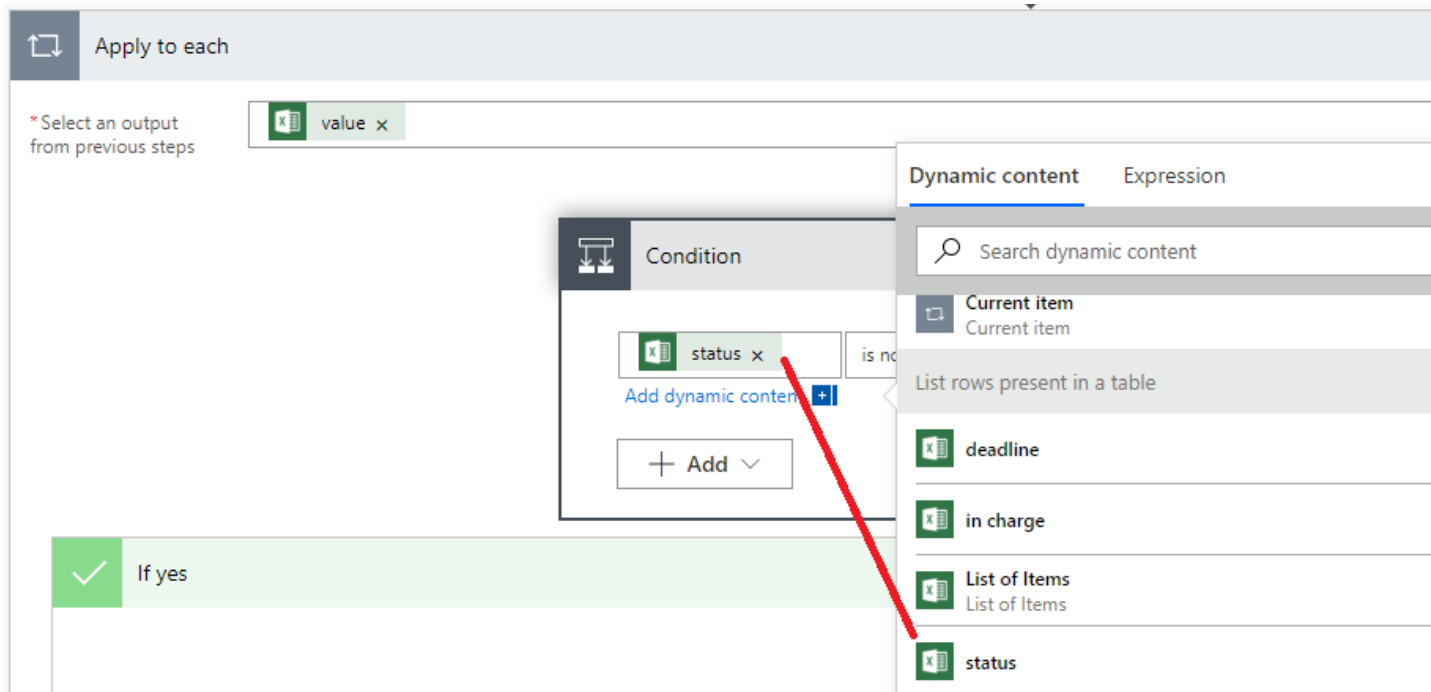
7. Let's loop through all tasks, so we need to add an **Apply to each** :



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



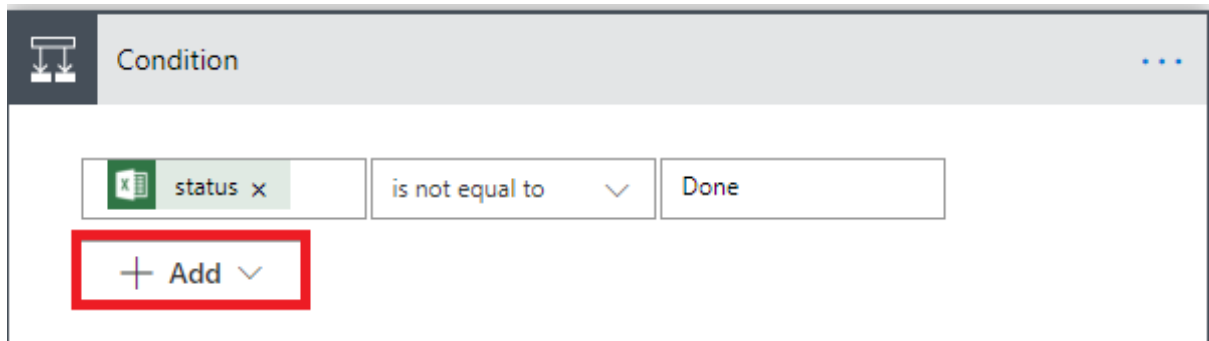
9. Add a **Condition** to filter the task where **Status** is not equal to **Done**:



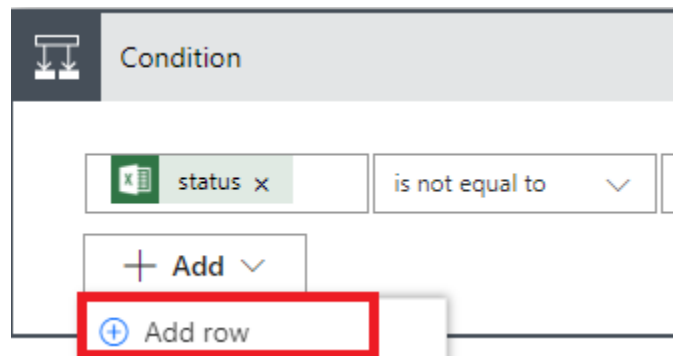
Make sure the operator used is "is not equal".

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

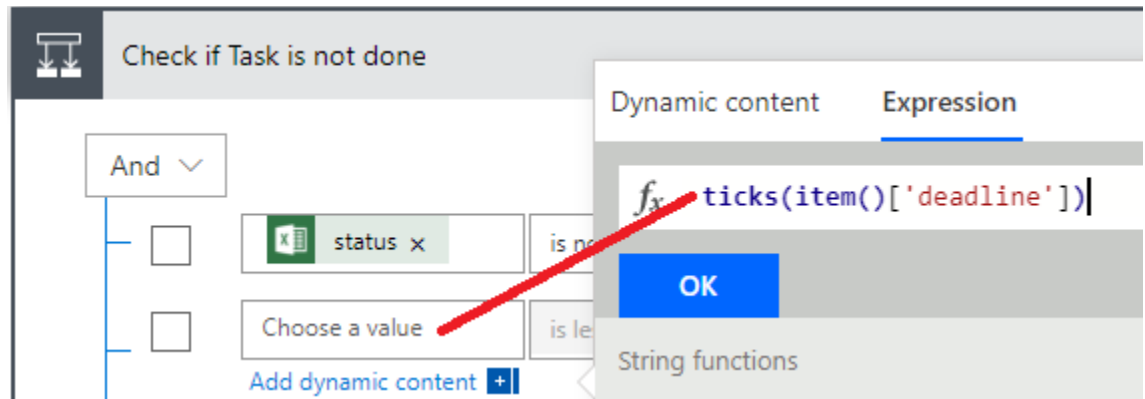
a) Click Add:



b) Select Add row:

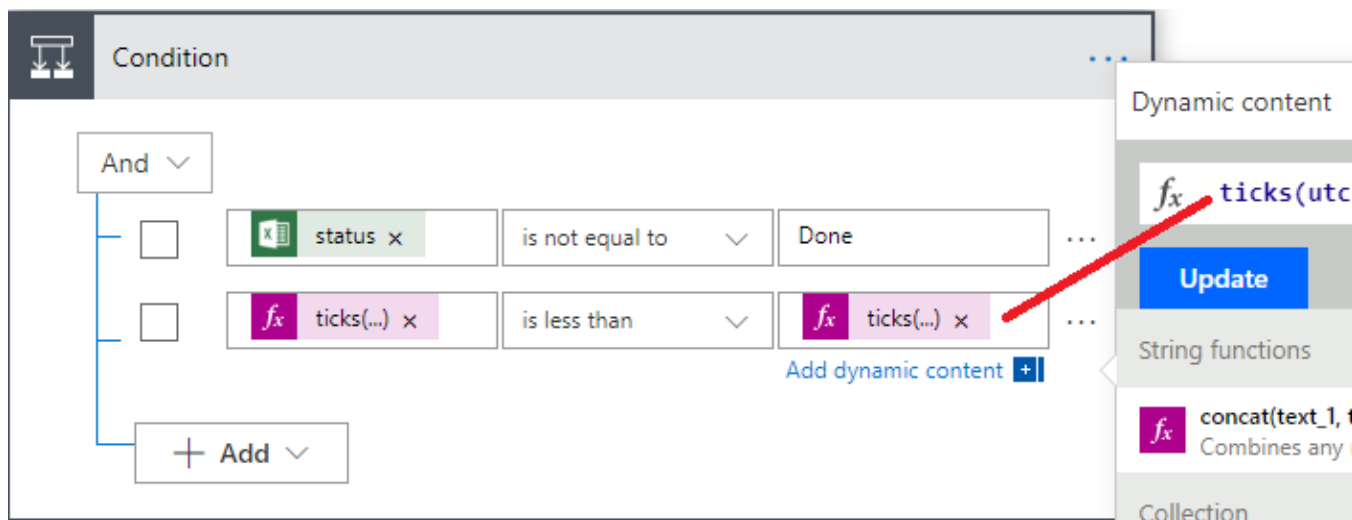


c) In "Choose a value", type the expression (and click Ok) :



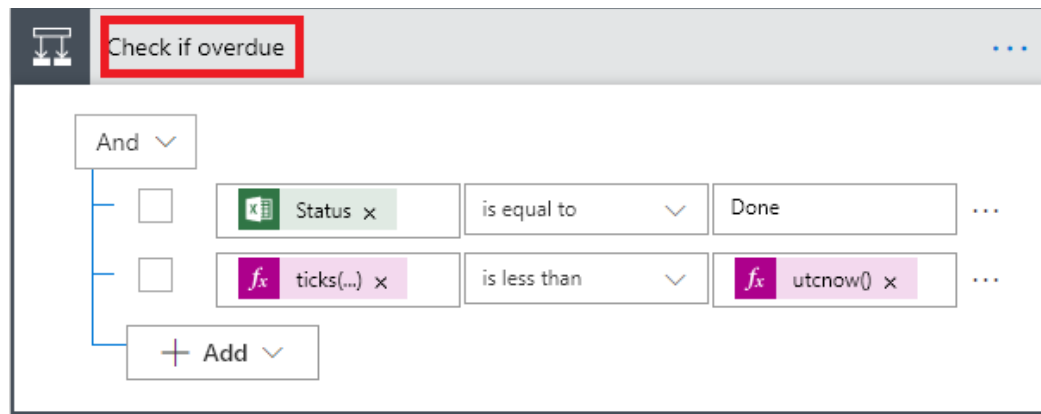
Since a timestamp is in string format, the **ticks** expression returns the number of ticks (100 nanosecond intervals) since 1<sup>st</sup> January 1601. By using ticks, we can compare two different timestamp values.

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

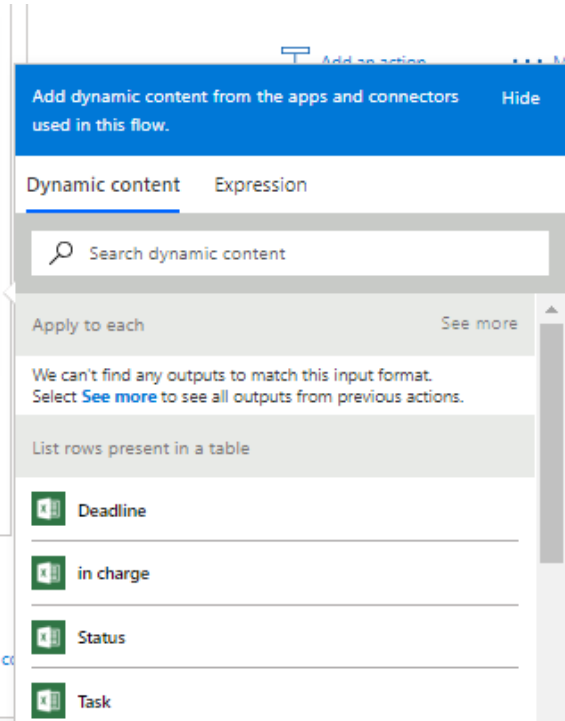
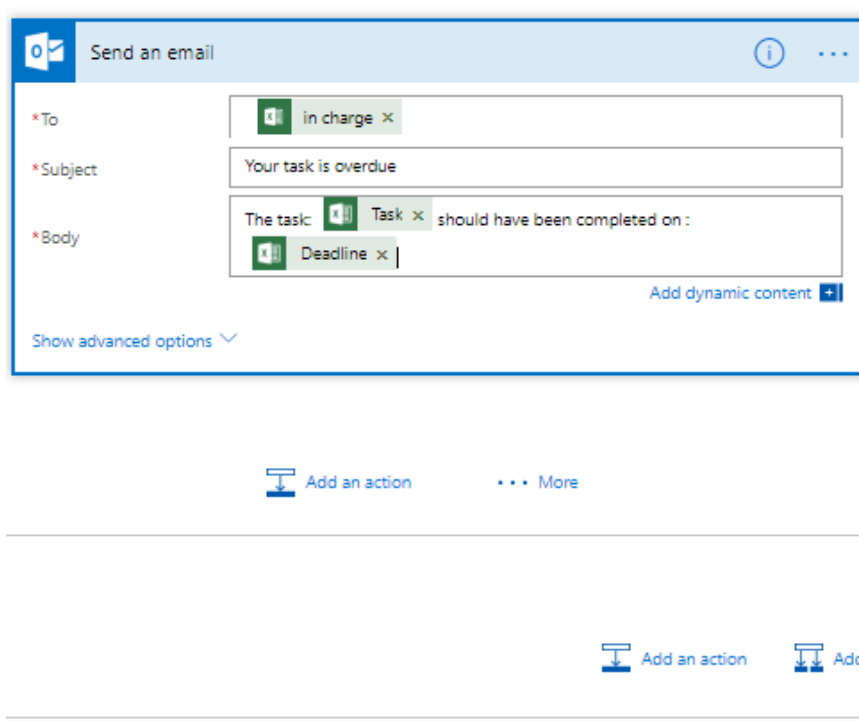


11. In the **If yes branch**, add a new **condition** where we will check if the due date is overdue:

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:
- The **To** field should get the Excel in charge value
  - The **Subject** should be: "your task is overdue"
  - 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 Filter in the List rows present in a table action :

 List rows present in a table ⓘ ⋮

* Location	OneDrive for Business	▼
* Document Library	OneDrive	▼
* File	/Tasks.xlsx	📁
* Table	Table1	▼
Filter Query	status ne 'Done'	
Order By	An ODATA orderBy query for specifying the order of entries.	
Top Count	Total number of entries to retrieve (default = all).	
Skip Count	The number of entries to skip (default = 0).	

[Hide advanced options](#) ^

## 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.

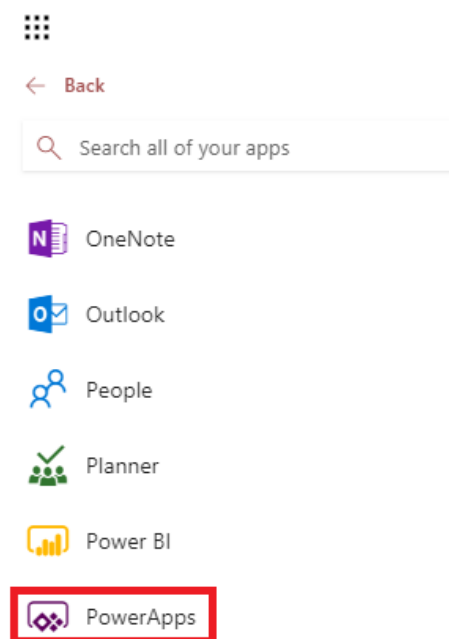
**Scenario:** You will create a Business Process Flow (BPF) that illustrates how to approve a loan. You will have to create business entities in CSD 2; these entities will be consumed by the BPF.

**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

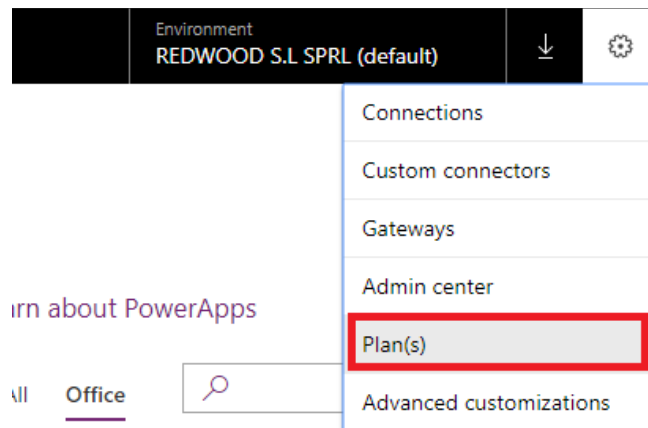
### Tasks:

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

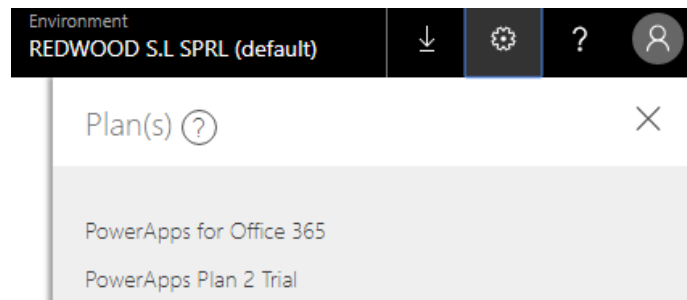


2. If you are working on your own tenant, you first must have **PowerApps Plan 2** in order to be able to create **Model driven apps**. To check your PowerApps plan, go to the following PowerApps menu:



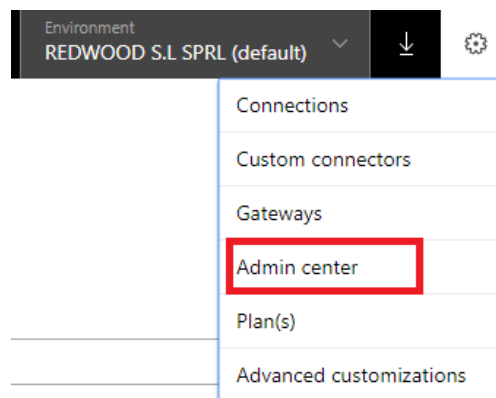


You should get **plan 2**:



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:

## New environment

Create new environments for app and flow development and to maintain separate databases. [Learn more](#)

Environment name

Workshop

Region (?)

Europe (default)

Can't be changed once your environment is created.

Environment type (?)

Trial

Cancel

Create environment

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**:

✓ You created an environment



Do you want to create a database?  
(Recommended)

Your environment includes access to the Common Data Service. Create a database to start using it.

- Collect, store, and share data.
- Use data modeling
- Create custom forms
- Manage security and access to data

Skip

Create database

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

## Create a database for this environment ?

Choose the currency and language your data should use.

Currency ?

EUR ▼

Language ?

English ▼

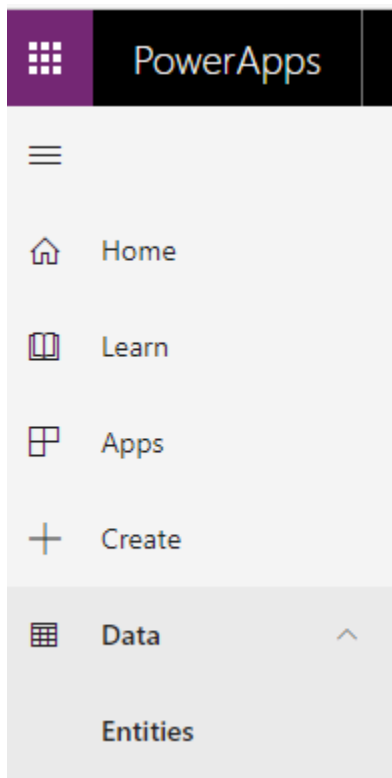
☒ Include sample apps and data

ⓘ By choosing **Create my database**, you agree Microsoft can use entity and field names that you create (but not content in the database tables) to help improve our common data model. These names may be stored in our diagnostic systems and copied across regions. [Learn more](#)

Cancel

Create database

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.

## New entity

×

Display name \*

Plural display name \*

Name \* ⓘ

cr714\_ Loan

☐ Enable attachments (including notes and files)

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

## Amount


×

Display name \*

Name \* ⓘ

cr327\_ Amount

Data type \* ⓘ

 Currency ▼

☐ Required

☒ Searchable

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

## Account



Display name \*

Account

Name \* ⓘ

cr327\_ Account

Data type \* ⓘ

Customer

☐ Required

☒ Searchable

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

## Approval Status



Display name \*

Approval Status

Name \* ⓘ

cra46\_ ApprovalStatus

Data type \* ⓘ

Option Set

Option set \*

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

The screenshot shows the 'New option set' dialog in Microsoft Flow. On the left, a list of options includes 'A Yes or No boolean', 'Activity Type', 'Authentication Protocol', 'Category', 'Channel Activities', 'Component State', 'Component Type', 'Confirm delete appointment series', 'Connector Type', 'Delivery Priority', and 'Dependency Type'. The 'Category' option is selected. On the right, the 'Approval Status' configuration panel is shown. It has fields for 'Display name \*' (set to 'Approval Status'), 'Name \*' (set to 'cra46\_ ApprovalStatus'), 'Data type \*' (set to 'Option Set'), and 'Option set \*' (a dropdown menu). A red box highlights the '+New option set' button, and a red arrow points from it to the 'Option set' dropdown.

Provide 3 options: Approved, Rejected, Pending:

The screenshot shows the 'Approval Status' configuration panel. It has fields for 'Display name \*' (set to 'Approval Status'), 'Name \*' (set to 'cra46\_ approvalstatus'), and a 'View more' link. Below the 'View more' link, there is a section titled 'Items (3)' which contains a list of three items: 'Approved', 'Rejected', and 'Pending'. Each item has a three-dot menu icon to its right. Below the list, there is a link to 'Add new item'.

## Approval Status ✕

Display name \*

Approval Status

Name \* ⓘ

cr327\_ ApprovalStatus

Data type \* ⓘ

☰ Option Set ▼

Option set \*

Approval Status ▼

Edit option set

Default value

[No default value] ▼

☐

Required

☒

Searchable

Click Save and Make Pending the default value:

## Approval Status

✕

Display name \*

Approval Status

Name \* ⓘ

cra46\_ ApprovalStatus

Data type \* ⓘ

☰ Option Set ▾

Option set \*

Approval Status ▾

Edit option set

Default value

Pending ▾

☒ Required

☒ Searchable

Click Done.

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



## Signed

×

Display name \*

Signed

Name \* ⓘ

cr327\_ Signed

Data type \* ⓘ

☰ Two Options ▾

Items

☒ Yes

☐ No

Default value

[No default value] ▾

☐ Required

☒ Searchable

Create last field : Send Money

Send Moeny

×

Display name \*

Send Moeny

Name \* ⓘ

cr327\_ SendMoeny

Data type \* ⓘ

☰ Two Options ▾

Items

Yes

No

Default value

[No default value] ▾

☐ Required

☒ Searchable

Save your field and save the entity.

The following table summarizes your entity Loan fields and settings=

## Microsoft Flow in a Day

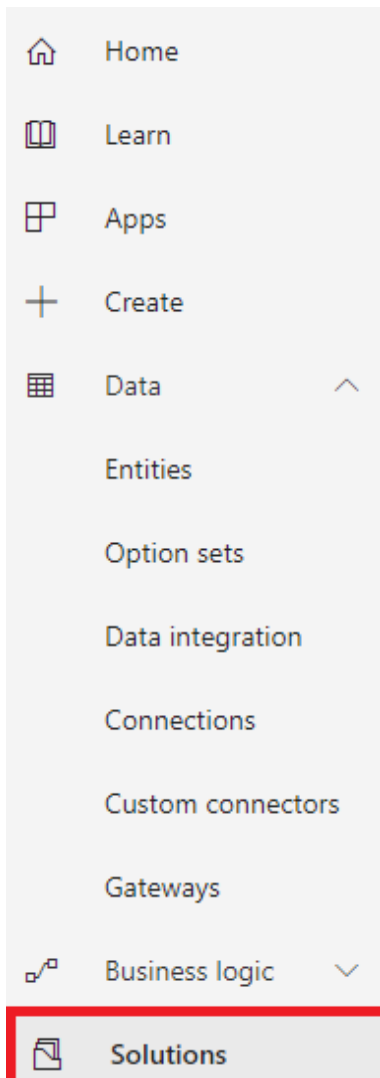
Solutions > Common Data Services Default Solution > Loan

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 :



## Microsoft Flow in a Day

Click Common Data Services Default Solutions :

## Solutions

Display name		Created ↓	Version	Managed externally?
Asset Checkout	...	5/12/2019	0.0.0.1	
Innovation Challenge	...	5/12/2019	0.0.0.1	
Fundraiser	...	5/12/2019	1.0.0.2	
Common Data Services Default Solution	...	5/12/2019	1.0.0.0	
Default Solution	...	5/12/2019	1.0	

In the menu above, click Switch to classic :

The screenshot shows the PowerApps interface. On the left is a navigation pane with icons for Home, Learn, Apps, and Create. The main area displays 'Solutions > Common Data Services Default Solution'. Below this is a table with columns 'Display name' and 'Name'. The table contains one row with 'Account' in the 'Display name' column and 'account' in the 'Name' column. A context menu is open over the 'Account' row, showing options: 'Clone', 'Show dependencies', 'Apply upgrade', 'Switch to classic' (highlighted with a red rectangle), and 'Translations'. At the top of the main area, there are buttons for '+ New', '+ Add existing', 'Delete', 'Export', and 'Publish all customizations'.

You should see your custom Loan entity:

Solution: Common Data Services Default Solution

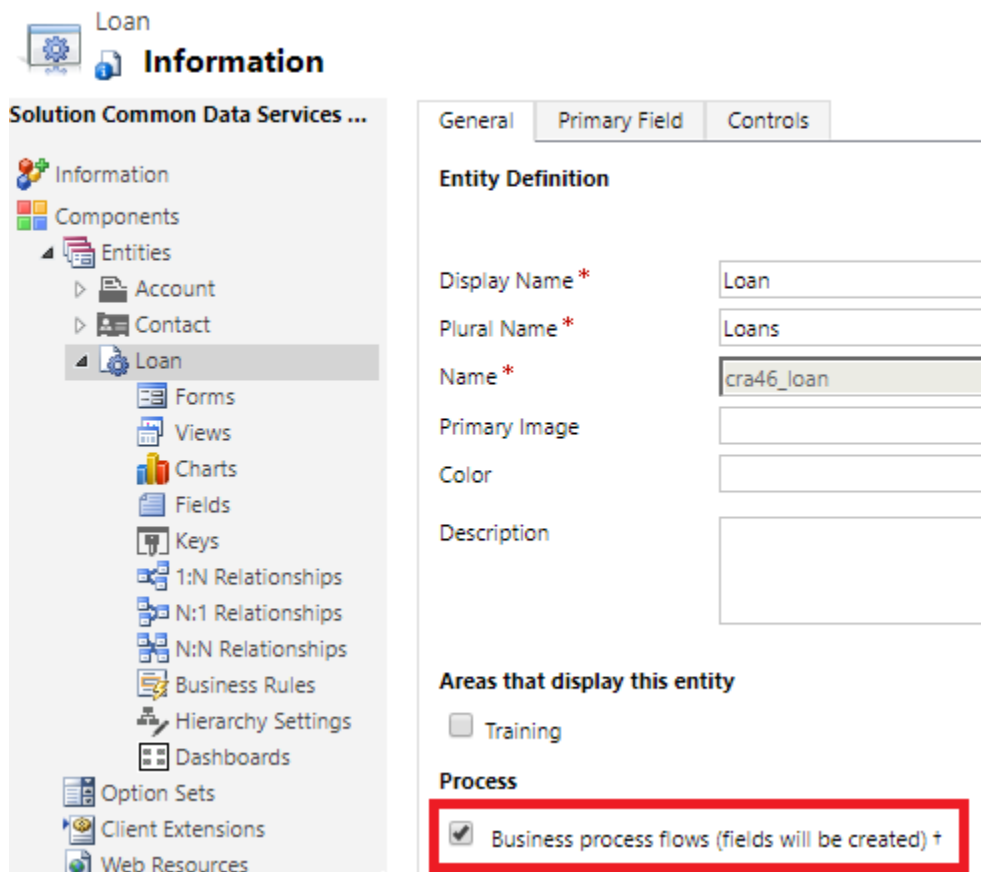
## Information

**Solution Common Data Services ...**

- Information
- Components**
  - Entities
  - Option Sets
  - Client Extensions
  - Web Resources
  - Processes
  - Plug-in Assemblies
  - Sdk Message Processing St...
  - Service Endpoints
  - Dashboards
  - Dialog Boxes

Component Type All			
New   Add Existing   Delete   Remove   Publish   Show Dependencies			
<input type="checkbox"/>	Display Name ↑	Name	Type
	Account	account	Entity
	Approval Status	cra46_approvalstatus	Option Set
	Contact	contact	Entity
	Loan	cra46_loan	Entity

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



Click the Save icon.

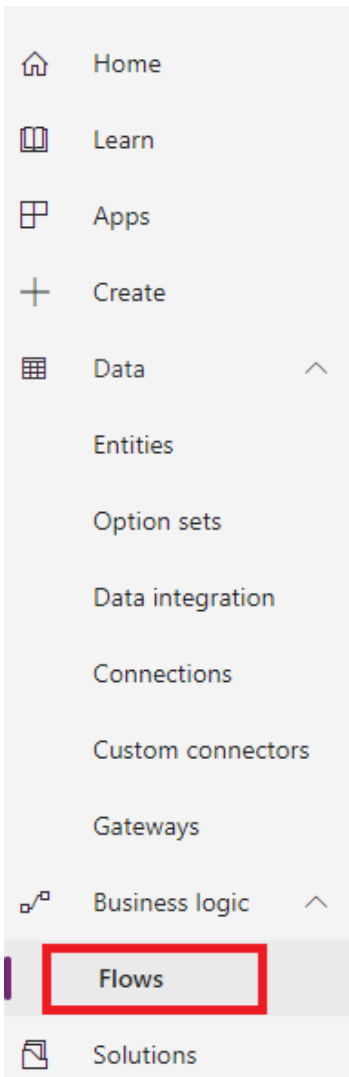


Click Publish.

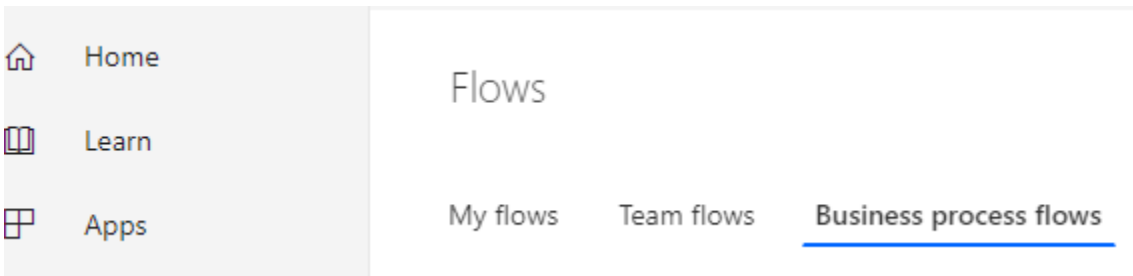


Let's get back to flow to create a new BPF flow. Make sure you are on the good environment.

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:

## Create business process flow ×

Display name \*

Loan BPF

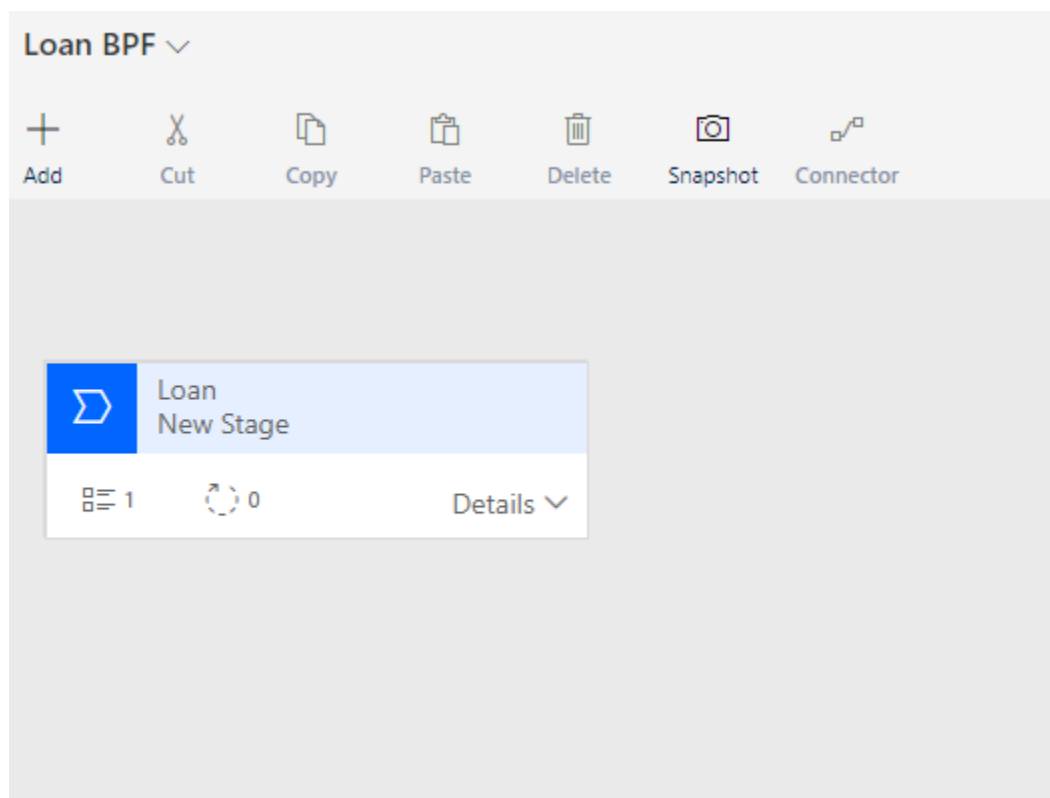
Name \*

new\_LoanBPF ⓘ

Common Data Service entity \*

Loan ⓘ

Click Next and your 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.

---

Components Properties

Stage

Display Name

Loan Basic Info

Category

Entity

Loan

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

---

Components Properties

Data Step

Step Name

Account

Data Field

Account

☒ Required

Sequence

1

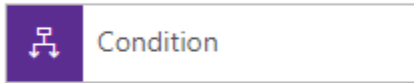
Click Apply.

Click Components to add a new Stage

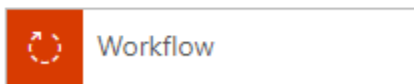


**Components** Properties

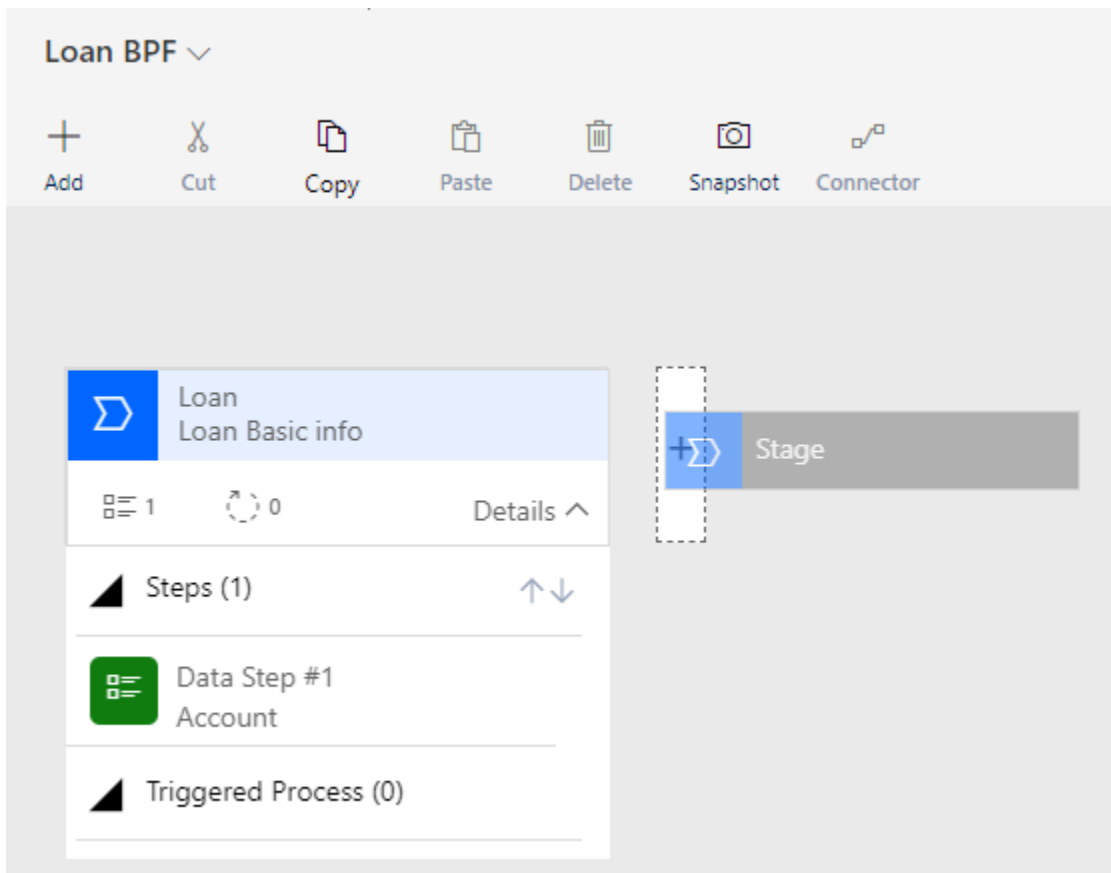
▲ Flow



▲ Composition



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



## Microsoft Flow in a Day

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

**Components** **Properties**

**Data Step**

Step Name

Amount

Data Field

Amount

☐ Required

Sequence

1

Click Apply.

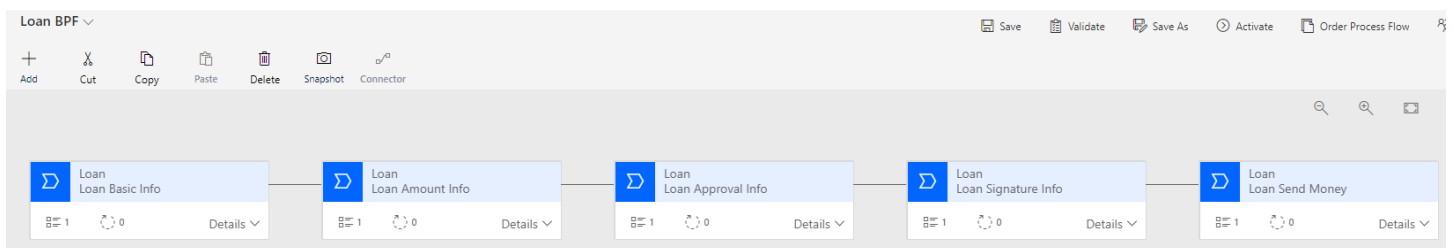
Name the stage Loan Amount info.

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:

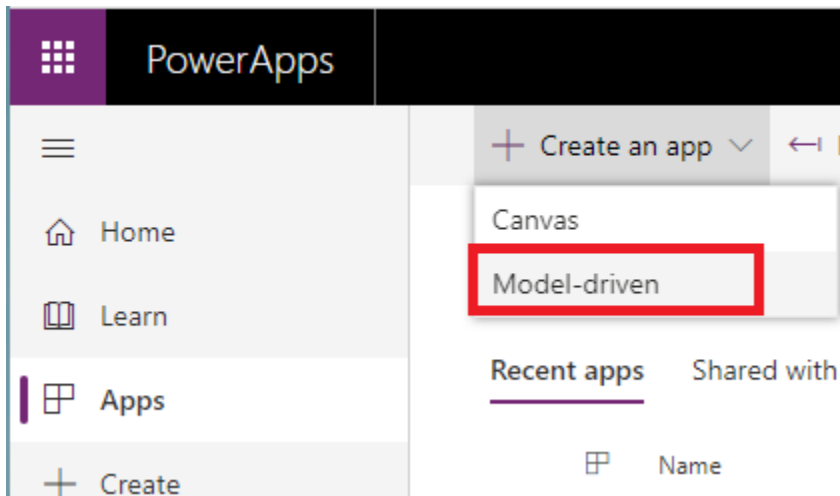


Click on Validate.

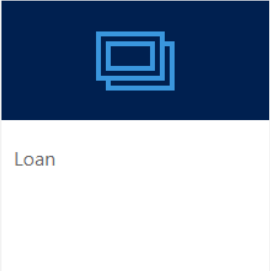

Click on Activate and confirm activation.

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:



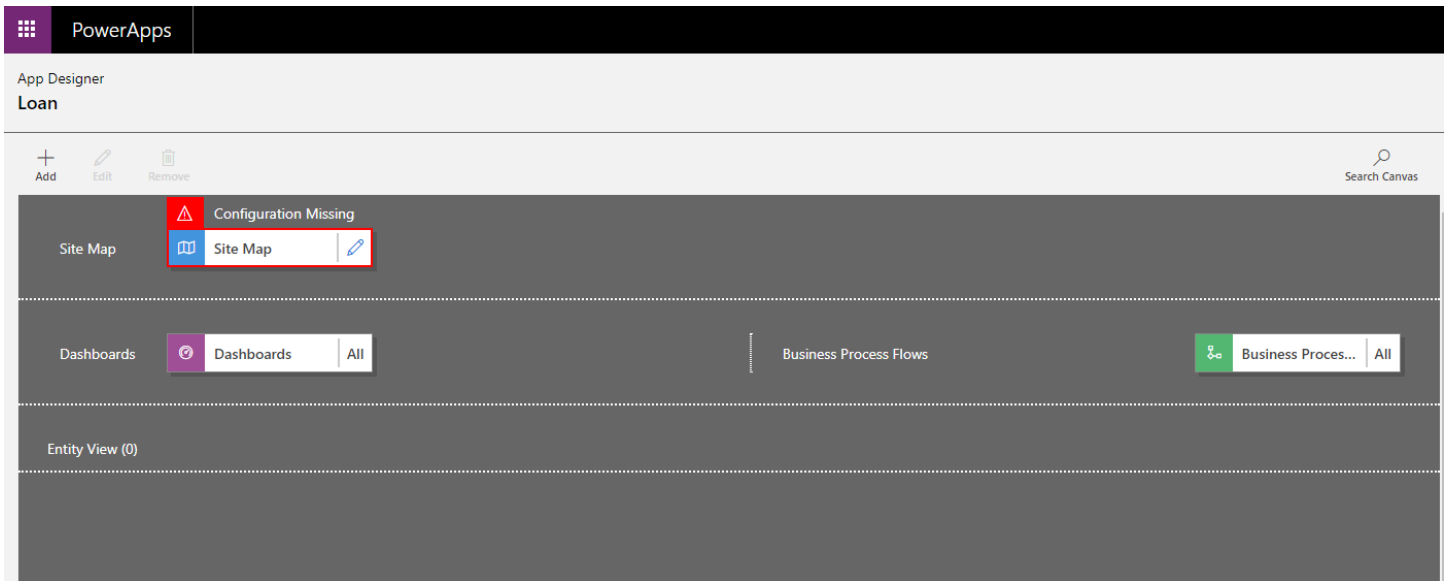
Name it Loan :

Name :*	<input type="text" value="Loan"/>	<b>App Tile:</b> 
Unique Name :*	<input type="text" value="cr327_Loan"/>	
Description:	<input type="text"/>	
Icon:	<input checked="" type="checkbox"/> Use Default Image	
Unified Interface URL:	<input type="text" value="https://orge37d24b4.crm4.dynamics.com/Apps/uniquename/cr327_Loan"/> 	
	<input type="checkbox"/> Use existing solution to create the App	
	<input type="checkbox"/> Choose a welcome page for the app	
	<input type="checkbox"/> Enable Mobile Offline	

Click Done.

The App Designer will show-up :

## Microsoft Flow in a Day



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

Components

Properties

SUB AREA

General

Type

Entity

Entity \*

Loan

URL

Default Dashboard

Select a dashboard

Title (1033)

Loan Subarea

Icon

Use Default Image

ID \*

New\_SubArea

☐ Parameter Passing

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

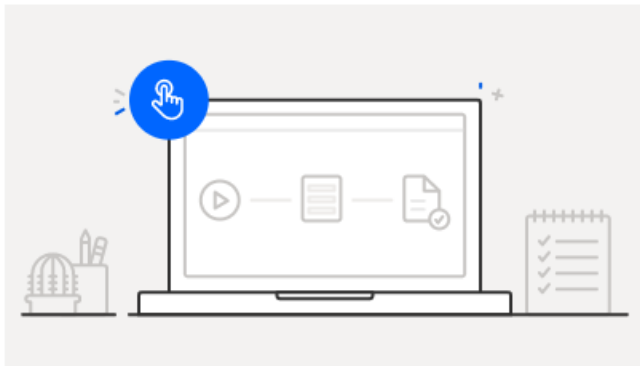
**Learning objective:** call an external REST api from flow

**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**:

#### Build an instant flow



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

#### Flow name

weather

#### Choose how to trigger this flow \*



From Microsoft Flow  
Microsoft Flow



From PowerApps  
PowerApps



<https://www.apixu.com/api-explorer.aspx>

## Protocol

HTTP

## Format

JSON

## Current Weather **Forecast Weather** Search/Autocomplete History Weather

Parameter	Value	Type	Location	Description
q	<input type="text" value="paris"/>	string	query	Pass US Zipcode, UK Postcode, Canada Postalcode, IP address
days	<input type="text" value="1"/>	integer	query	Number of days of weather forecast. Value ranges from 1 to 10

Show Response

### Call

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

### Response Code

200

### Response Headers

- 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
```

The screenshot shows a Microsoft Flow canvas with a 'Manually trigger a flow' step at the top. An arrow points down to an 'HTTP' step. The 'HTTP' step is configured with the following details:

- Method:** GET
- URI:** `http://api.apixu.com/v1/forecast.json?key=98a5c2b6b62a4dfbb23144756191705&q=paris&days=1` (This entire line is highlighted with a red border in the original image).
- Headers:** A table with 'Enter key' and 'Enter value' columns.
- Queries:** A table with 'Enter key' and 'Enter value' columns.
- Body:** A text box with 'Enter request content'.
- Buttons:** 'Add dynamic content' and 'Show advanced options'.

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"  
  }  
}  
]  
}  
}
```

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

The screenshot shows a Microsoft Flow Designer canvas with three actions:

- Manually trigger a flow**: The first action, with a blue icon of a hand clicking a button.
- Initialize variable**: The second action, highlighted with a red border. It has a purple icon of a variable {x}. The configuration fields are:
  - Name**: city temperature
  - Type**: Float (with a dropdown arrow)
  - Value**: Enter initial value
- HTTP**: The third action, with a green icon of a globe.

Arrows indicate the flow from the trigger to the variable initialization, and then to the HTTP action.

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

## Microsoft Flow in a Day

The screenshot displays a Microsoft Flow Designer interface with the following components:

- Manually trigger a flow**: The starting step of the flow.
- Initialize variable**: A step to create a new variable named `city temperature` of type `Float`. The value field is set to `Enter initial value`.
- HTTP**: A step to perform an HTTP request.
- Set variable**: A step to update the `city temperature` variable. The value is set to `body(...)`, which is linked to the output of the previous step.

The right sidebar shows the **Expression** tab with the following formula:

```
body('HTTP')['current']?['temp_c']
```

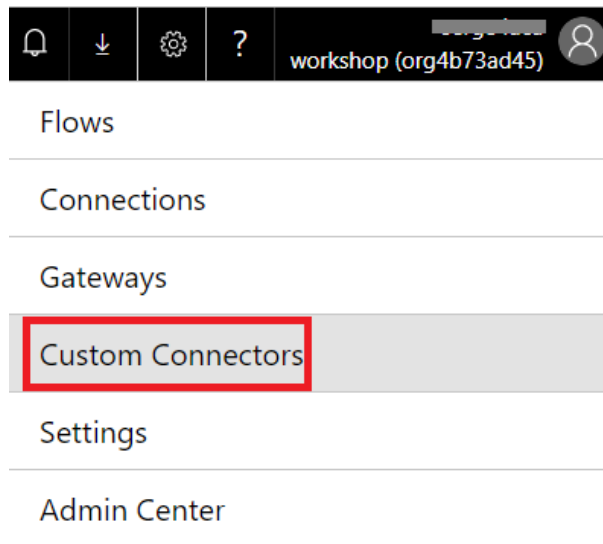
An **Update** button is visible at the bottom of the sidebar.

10. Run the flow and check the variable content.

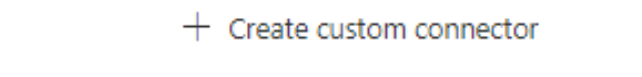
## Lab 9. Create a Flow/PowerApps Custom Connector

### Tasks:

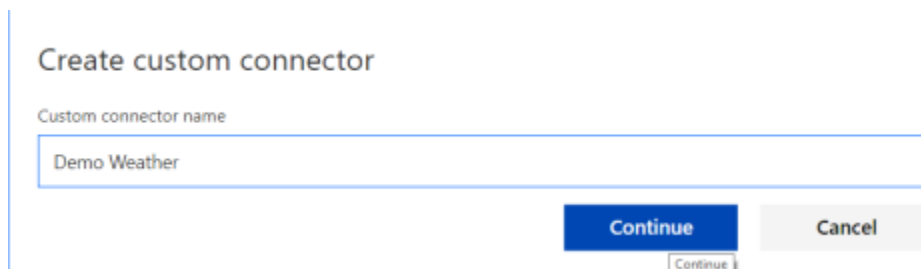
1. Go to the **Connector** menu and select **Custom Connectors**:



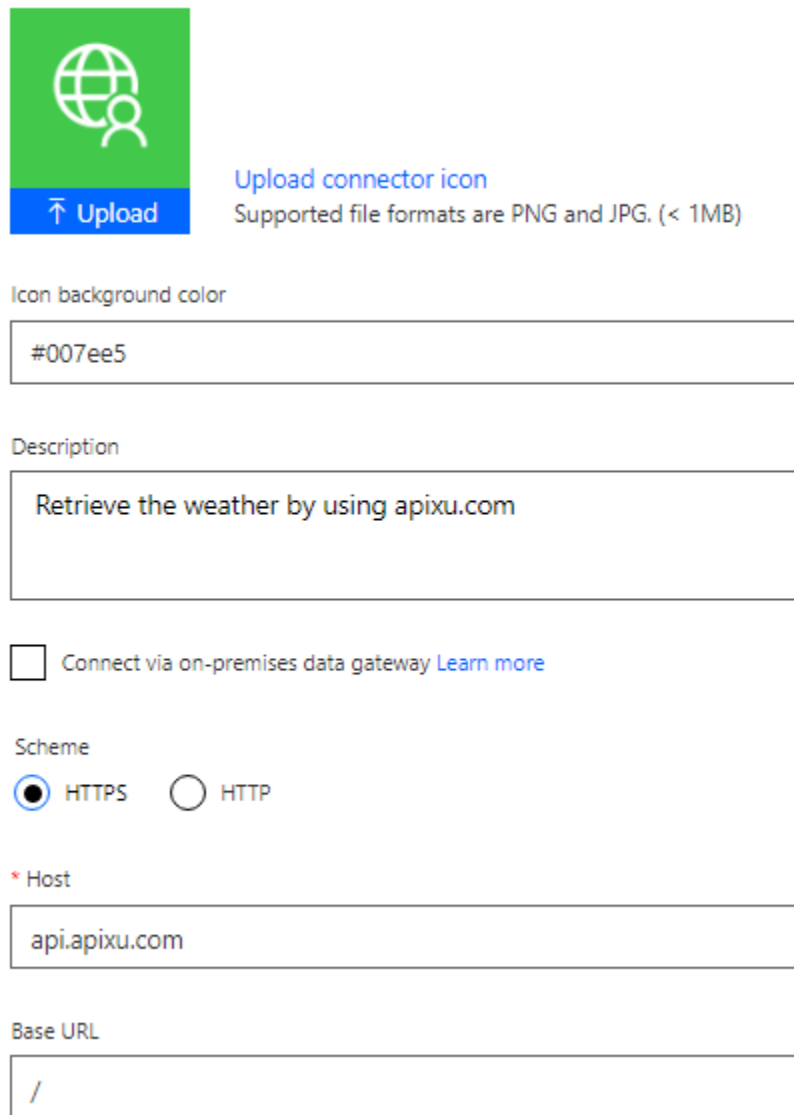
2. Click on **Create custom connector**:



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

A screenshot of a dialog box titled 'Create custom connector'. Inside the dialog, there is a label 'Custom connector name' above a text input field. The input field contains the text 'Demo Weather'. To the right of the input field are two buttons: 'Continue' (in blue) and 'Cancel' (in light gray). Below the 'Continue' button, there is a small, faint text label 'Continue'.

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



The image shows a web interface for uploading a connector icon. At the top left is a green square with a white globe icon and a person silhouette. Below it is a blue button with a white upward arrow and the word 'Upload'. To the right of the icon area, the text 'Upload connector icon' is followed by 'Supported file formats are PNG and JPG. (< 1MB)'. Below this is a section for 'Icon background color' with a text input field containing '#007ee5'. Next is a 'Description' section with a text input field containing 'Retrieve the weather by using apixu.com'. Below the description is a checkbox labeled 'Connect via on-premises data gateway' with a link 'Learn more'. Underneath is a 'Scheme' section with two radio buttons: 'HTTPS' (selected) and 'HTTP'. Below the scheme is a '\* Host' section with a text input field containing 'api.apixu.com'. Finally, there is a 'Base URL' section with a text input field containing '/'. The entire form is styled with a clean, modern look using a light gray background and blue accents.

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

Icon background color

#007ee5

Description

Retrieve the weather by using apixu.com

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

Scheme

☒ HTTPS ☐ HTTP

\* Host

api.apixu.com

Base URL

/

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:

**Authentication type**

Choose what authentication is implemented by your API \*

API Key

Edit

**API Key**

Users will be required to provide the API Key when creating a connection

Parameter label \*

key

Parameter name \*

key

Parameter location \*

Query

Edit

9. Click **Definition**:

1. General > 2. Security > **3. Definition** > 4. Test

Actions (0)

Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

Actions

+ New action

Triggers (0)

Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can take action on that event.

+ New trigger

References (0)

References are reusable parameters used by both actions and triggers.

Start by adding an action or trigger on the left.

← Security

10. Click on **New action**:

11. Define the Action as follow:

1. General > 2. Security > **3. Definition** > 4. Test ✓ Create connector

▼ Actions (1)

Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

! 1 **GetWeather** ...

+ **New action**

▼ Triggers (0)

Triggers read data in from your connector. A trigger focuses on a particular event that happens, say a new Contact or Order being created and provides the relevant data so that users can

**General**

Summary [Learn more](#)

Get weather

Description [Learn more](#)

Get weather in a specific city

Operation ID \*

This is the unique string used to identify the operation.

GetWeather

Visibility [Learn more](#)

☒ none ☐ advanced ☐ internal ☐ important

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):

```

{
  "location": {
    "name": "London",
    "region": "City of London, Greater London",
    "country": "United Kingdom",
    "lat": 51.52,
    "lon": -0.12
  },
  "last_updated_epoch": 1533387637,
  "last_updated": "2018-08-04 14:00",
  "temp_c": 26.0,
  "temp_f": 78.8,
  "is_day": 1,
  "condition": "cloudy",
  "icon": "http://cdn.apixu.com/weather/64x64/day/116.png",
  "code": 1003,
  "wind_mph": 9.4,
  "wind_kph": 15.1,
  "wind_degree": 110,
  "feelslike_c": 26.5,
  "feelslike_f": 79.8,
  "vis_km": 10.0,
  "vis_miles": 6.0
}
    
```

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 **import**:

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:

The screenshot shows the 'Request' connector configuration in Microsoft Flow. At the top, there's a blue header with the word 'Request' and a button labeled '+ Import from sample'. Below this, the 'Verb' is set to 'GET'. The 'URL' field contains 'https://apixu.com/v1/current.json'. Under the 'Query' section, there is a table with two columns: 'key' and 'q'. The 'q' column has a dropdown menu open, showing 'Edit' and 'Delete' options. The 'Delete' option is highlighted with a red rectangle. The 'Body' section is currently empty.

**Request** + Import from sample

\* Verb  
The verb describes the operations available on a single path.  
**GET**

\* URL  
This is the request URL.  
`https://apixu.com/v1/current.json`

Path  
Path is used together with Path Templating, where the parameter value is actually part of the operation's URL.

Query  
Query parameters are appended to the URL. For example, in `/items?id=###`, the query parameter is `id`.

key	...	q	...
<div>✎ Edit</div> <div><b>Delete</b></div>			

...s that are part of the request.

Body  
The body is the payload that's appended to the HTTP request. There can only be one body parameter.

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 :



Parameter

\* Name

q

Description [Learn more](#)

fill in the city name

Summary [Learn more](#)

City name :

Default value

\* Is required?

☒ Yes ☐ No

Visibility [Learn more](#)

☒ none ☐ advanced ☐ internal ☐ important

\* Location

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:

• URL

This is the request URL.

`https://api.apixu.com/v1/current.json`

Path

Path is used together with Path Templating, where the parameter value is actually part of the operation's URL.

Query

Query parameters are appended to the URL. For example, in `/items?id=###`, the query parameter is `id`.

key ... q ...

Headers

These are custom headers that are part of the request.

Body

The body is the payload that's appended to the HTTP request. There can only be one body parameter.

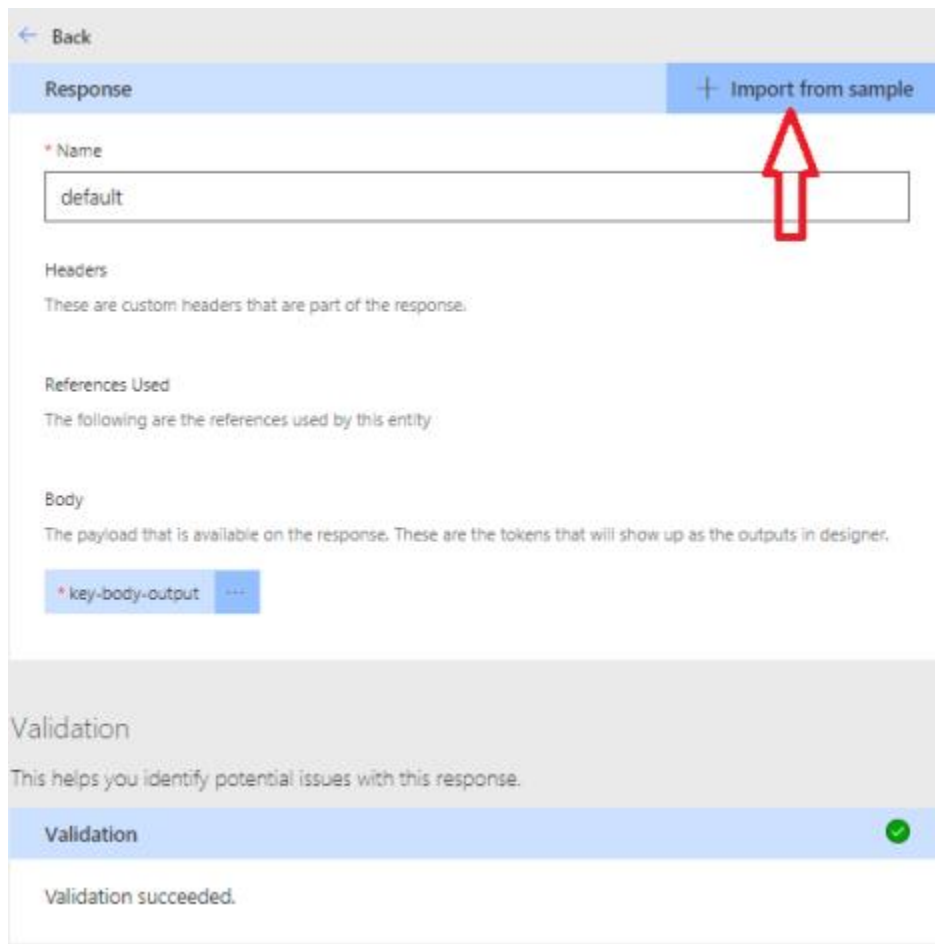
Response

It defines the shape of response returned by the underlying connector when making the request.

default default

+ Add default response

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



← Back

Response + Import from sample

\* Name  
default

Headers  
These are custom headers that are part of the response.

References Used  
The following are the references used by this entity

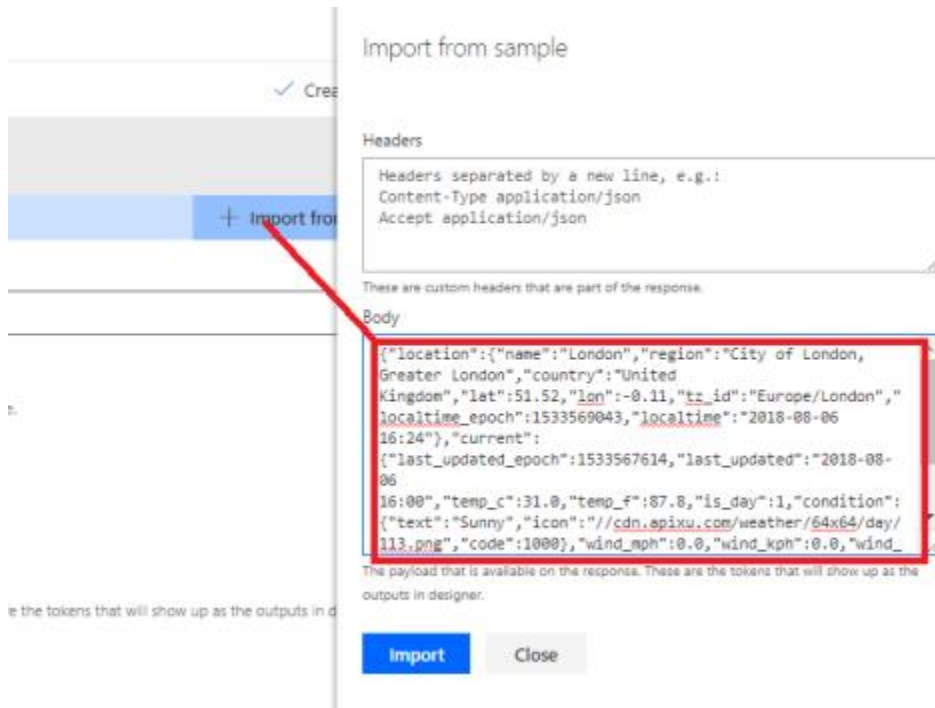
Body  
The payload that is available on the response. These are the tokens that will show up as the outputs in designer.

\* key-body-output

Validation  
This helps you identify potential issues with this response.

Validation ✓  
Validation succeeded.

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



Import from sample

Headers  
Headers separated by a new line, e.g.:  
Content-Type application/json  
Accept application/json

These are custom headers that are part of the response.

Body  
The payload that is available on the response. These are the tokens that will show up as the outputs in designer.

```
{
  "location": {
    "name": "London",
    "region": "City of London, Greater London",
    "country": "United Kingdom",
    "lat": 51.52,
    "lon": -0.11,
    "tz_id": "Europe/London",
    "localtime_epoch": 1533569043,
    "localtime": "2018-08-06 16:24",
    "current": {
      "last_updated_epoch": 1533567614,
      "last_updated": "2018-08-06 16:00",
      "temp_c": 31.0,
      "temp_f": 87.8,
      "is_day": 1,
      "condition": {
        "text": "Sunny",
        "icon": "http://cdn.epixu.com/weather/64x64/day/113.png",
        "code": 1000
      },
      "wind_mph": 0.0,
      "wind_kph": 0.0,
      "wind_
```

Import Close

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:

4. Test ✓ Update connector

GetWeather

\* q

london

Test operation

⊗ Request failed. Please try again in a few minutes.

Request Response

Status

Unknown

Headers

Unknown

Body

Unknown

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

⊗ Request failed. Please try again in a few minutes.

Request Response

Uri

https://europe-001.azure-apim.net/apim/demo.20weather.5f5afd8f6ba61df491.5f382646df4541cc

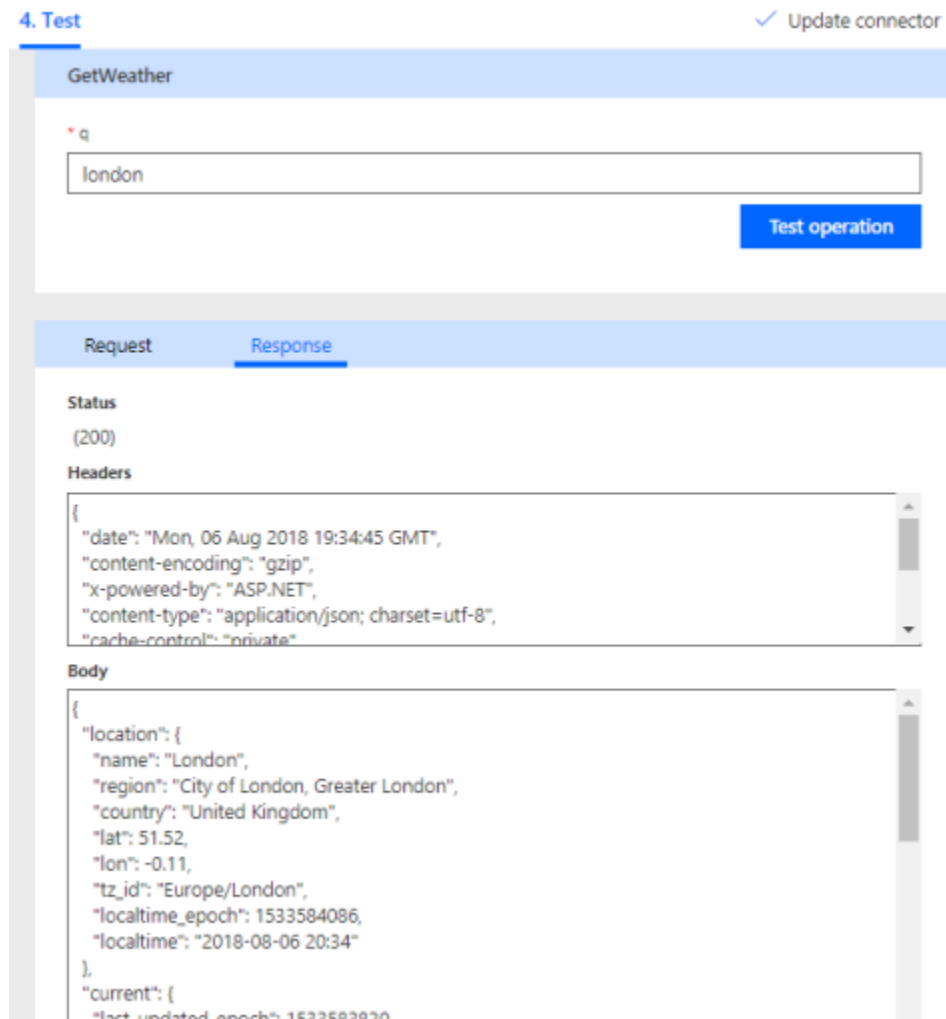
Method

get

Headers

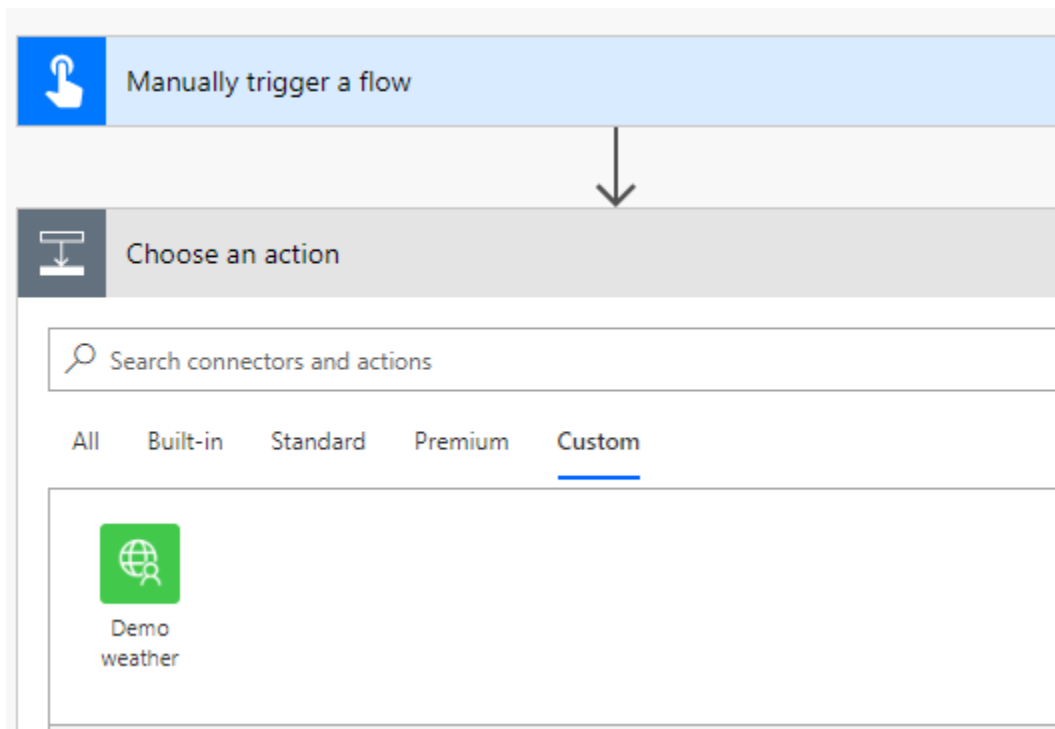
vmMXR2a3dMeFlhSFMzcTZsVWpVWUlhdyJ9.eYJhdWQjOiJodHRwcZovL3NlcnZpY2UuZmxvdy5taWNy

27. After a while, Test should be working



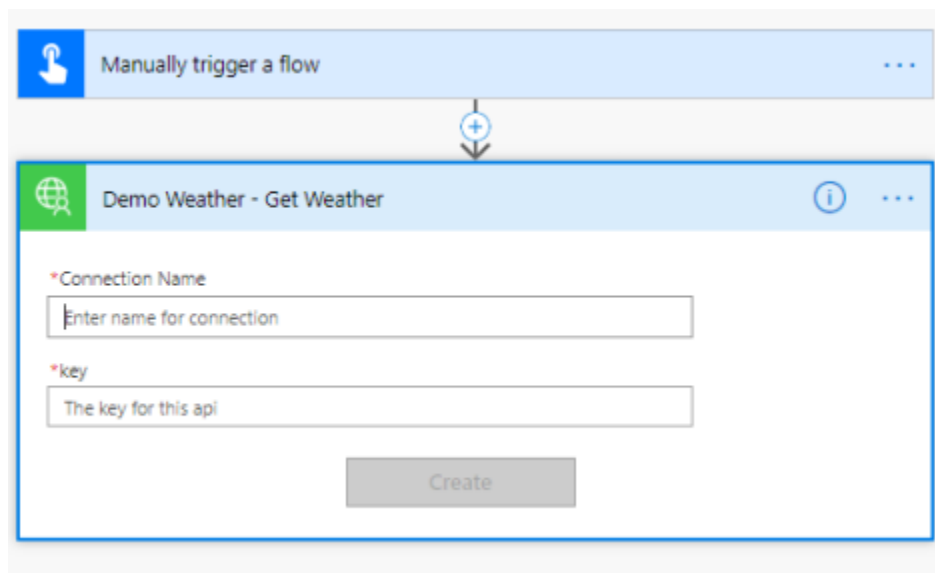
28. You can now create a new Empty flow started from a button.

29. Add an action from the Custom category; you should find you Demo Weather custom connector:



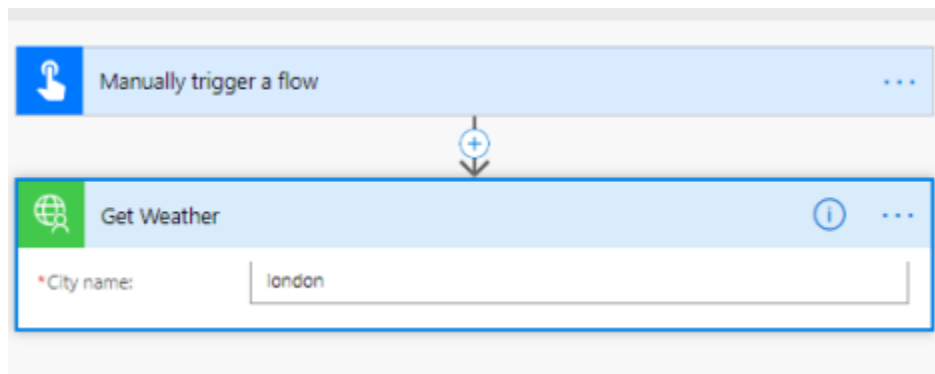
30. Select the **Get Weather** action

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.



localtime_epoch	1533584729
localtime	2018-08-06 20:45
last_updated_epoch	1533583820
last_updated	2018-08-06 20:30
temp_c	29
temp_f	84.2
is_day	0
text	Clear
icon	//cdn.apixu.com/weather/64x64/night/113.png

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

The screenshot shows the Microsoft Flow Designer interface. The flow consists of three steps: 'Manually trigger a flow', 'Get weather', and 'Compose'. The 'Compose' step is currently selected, and its 'Inputs' field is being edited. A red arrow points from the 'temp\_c' dynamic content option in the right-hand pane to the 'Inputs.' field in the 'Compose' step.

Dynamic content pane details:

Dynamic content	Expression
sunrise	sunrise
sunset	sunset
temp_c	temp_c
temp_f	temp_f



## 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:

**Call controller**

\*Method: POST

\*URI: Output

Headers: content-type: application/json

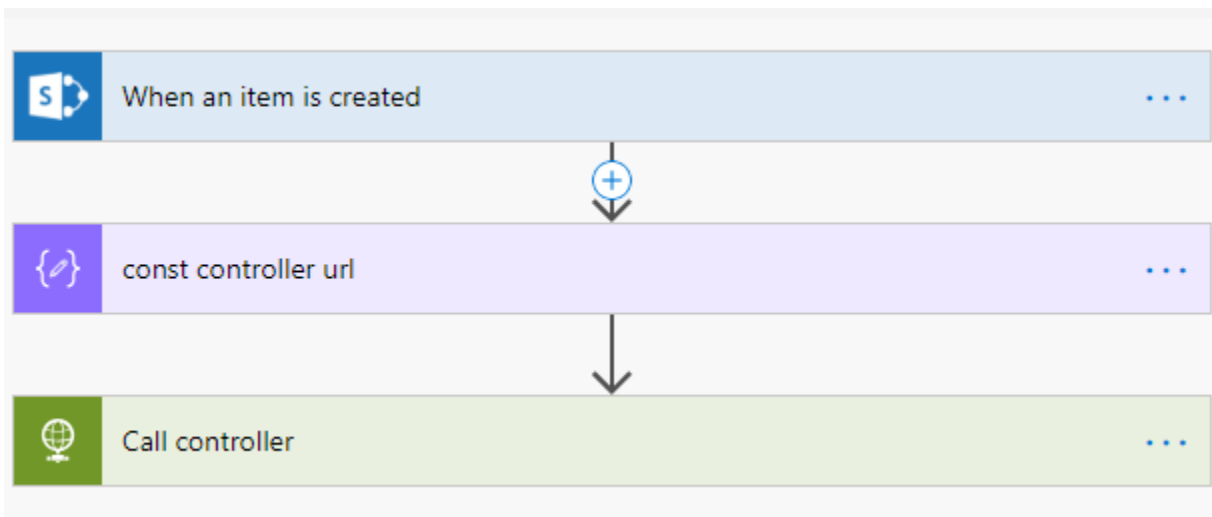
Queries: Enter key, Enter value

Body: {  
 "itemid": ID,  
 "requester": "Created By Email",  
 "from": "launcher",  
 "response": ""  
}

Show advanced options

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

The flow looks like this:



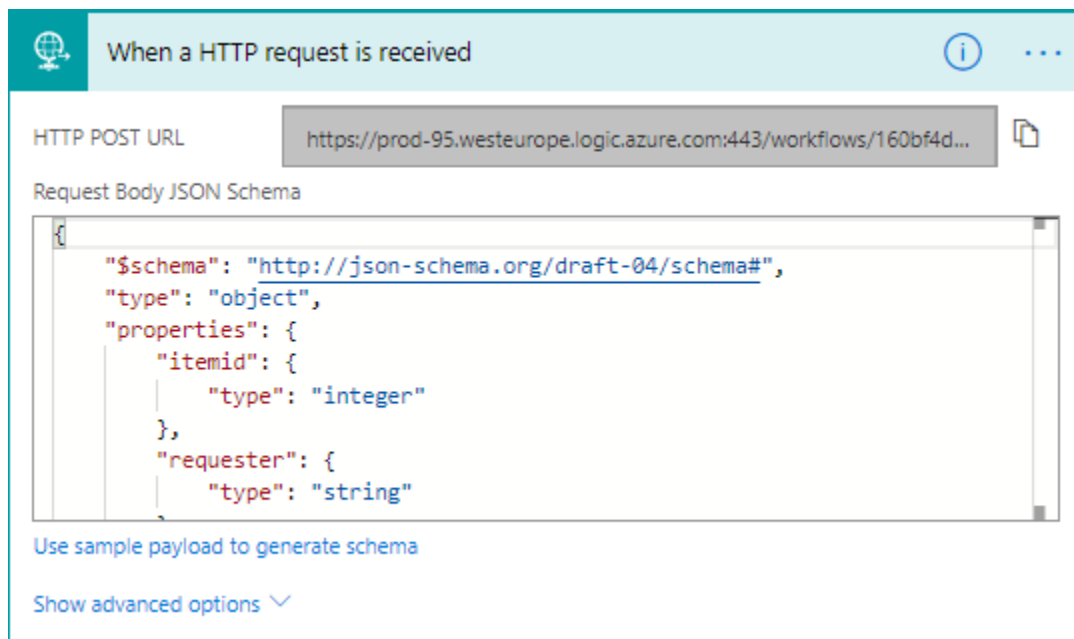
The flow controller will be implemented in the last place. Let's focus first on the Approval logic.

### 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:

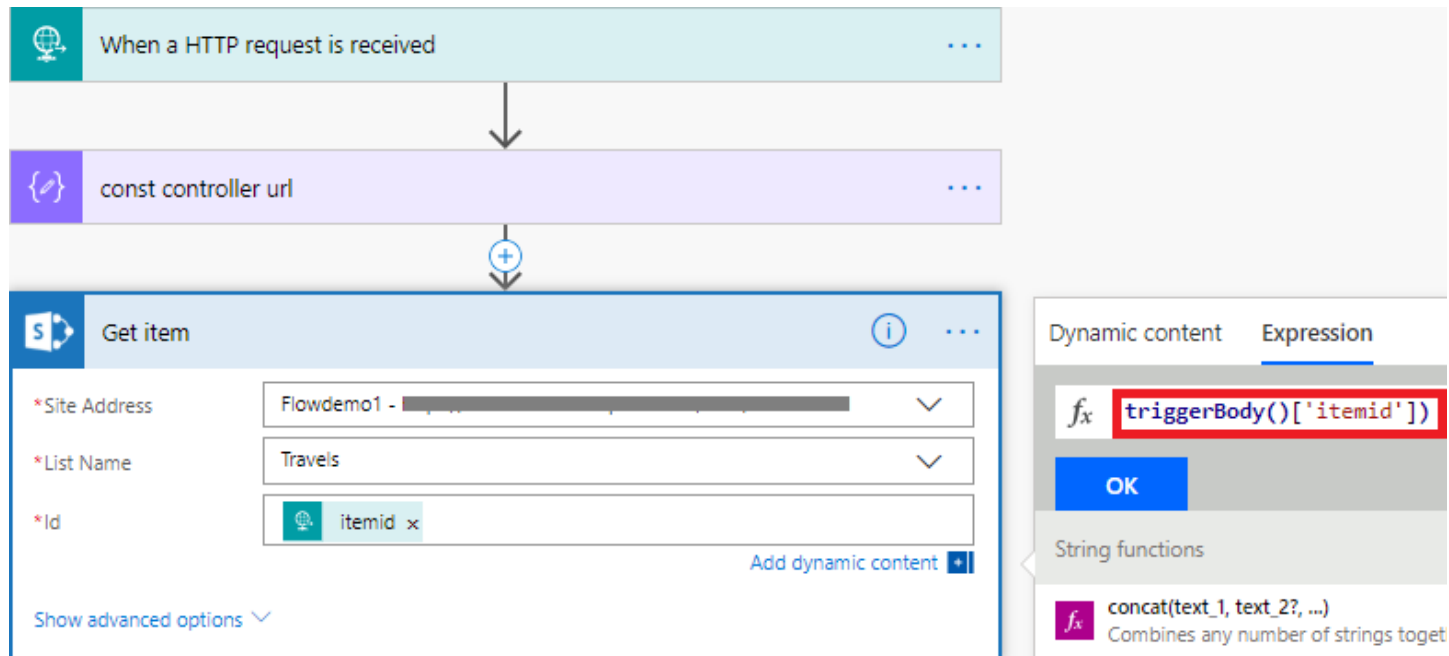
```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
```

```
"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:

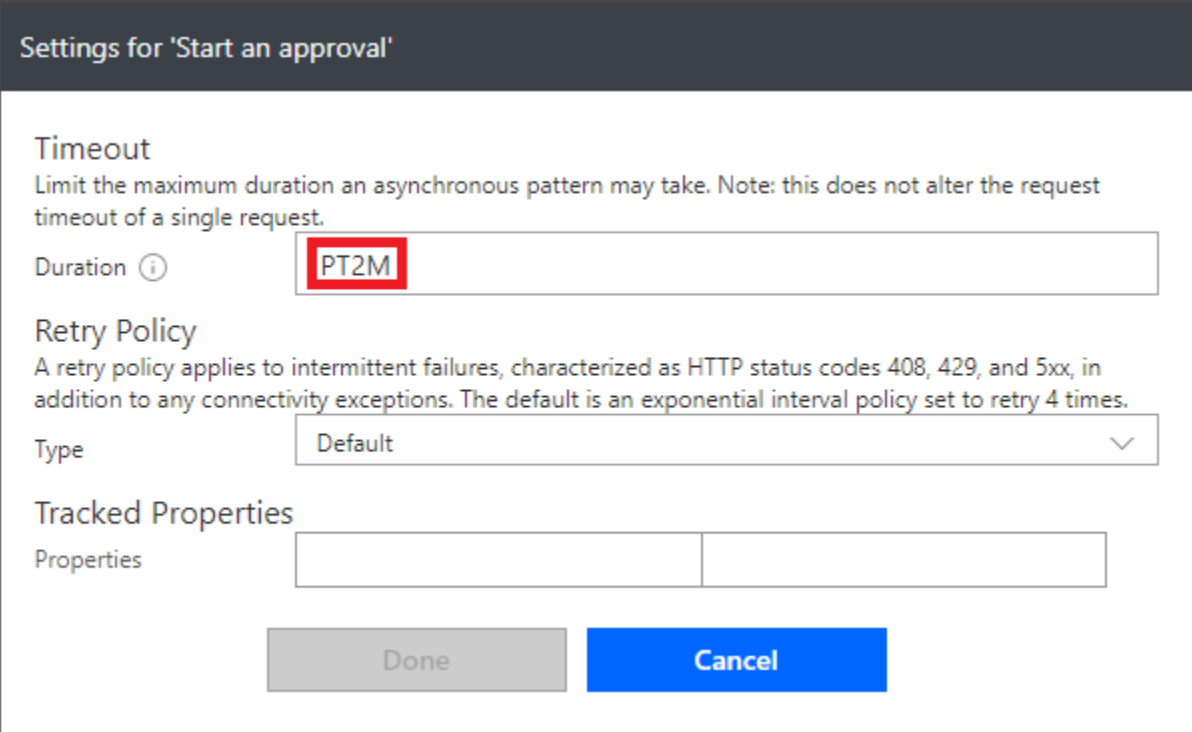
The 'Start an approval' action is configured with the following fields:

- \*Approval type:** Anyone from the assigned list
- \*Title:** New travel to approve
- \*Assigned to:** approver
- Requestor:** requester
- Details:** Please approve the expense: Title, amount: Amount, from: Created By DisplayName
- Item link:** Add a link to the item to approve
- Item link description:** Describe the link to the item

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):



Settings for 'Start an approval'

**Timeout**  
Limit the maximum duration an asynchronous pattern may take. Note: this does not alter the request timeout of a single request.

Duration ⓘ

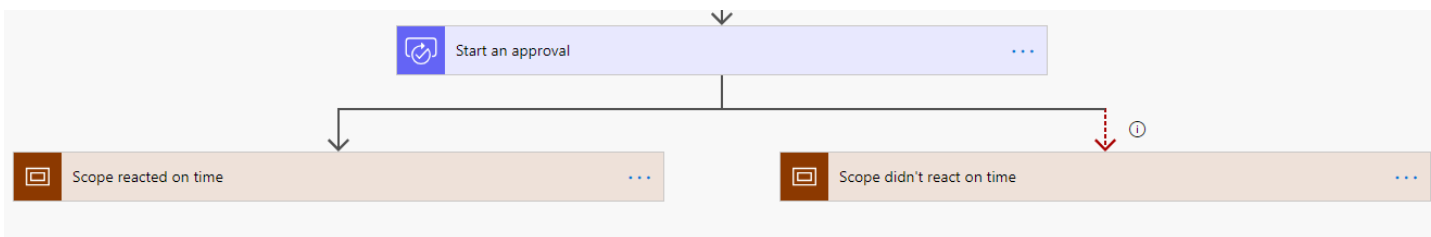
**Retry Policy**  
A retry policy applies to intermittent failures, characterized as HTTP status codes 408, 429, and 5xx, in addition to any connectivity exceptions. The default is an exponential interval policy set to retry 4 times.

Type

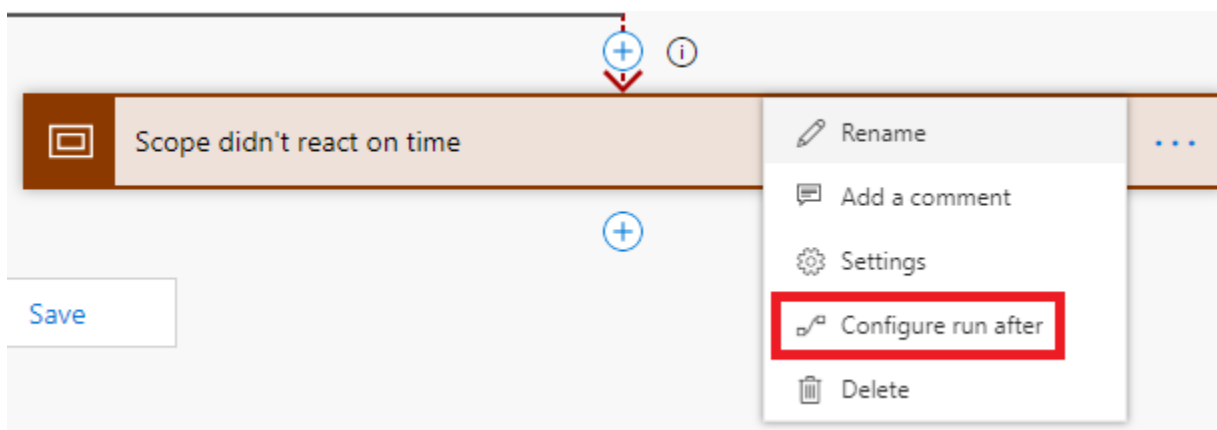
**Tracked Properties**

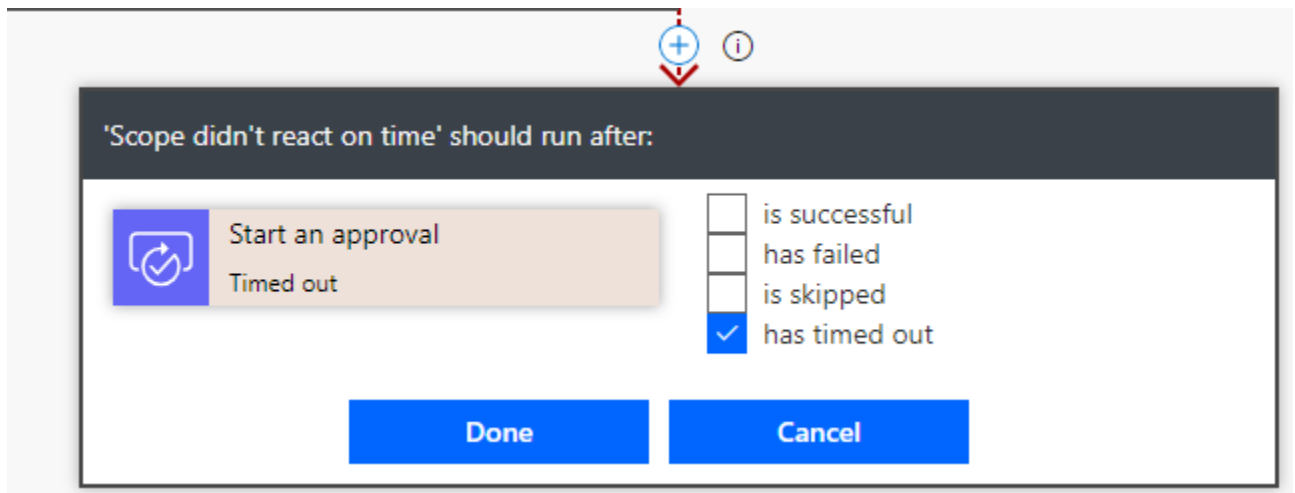
Properties

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

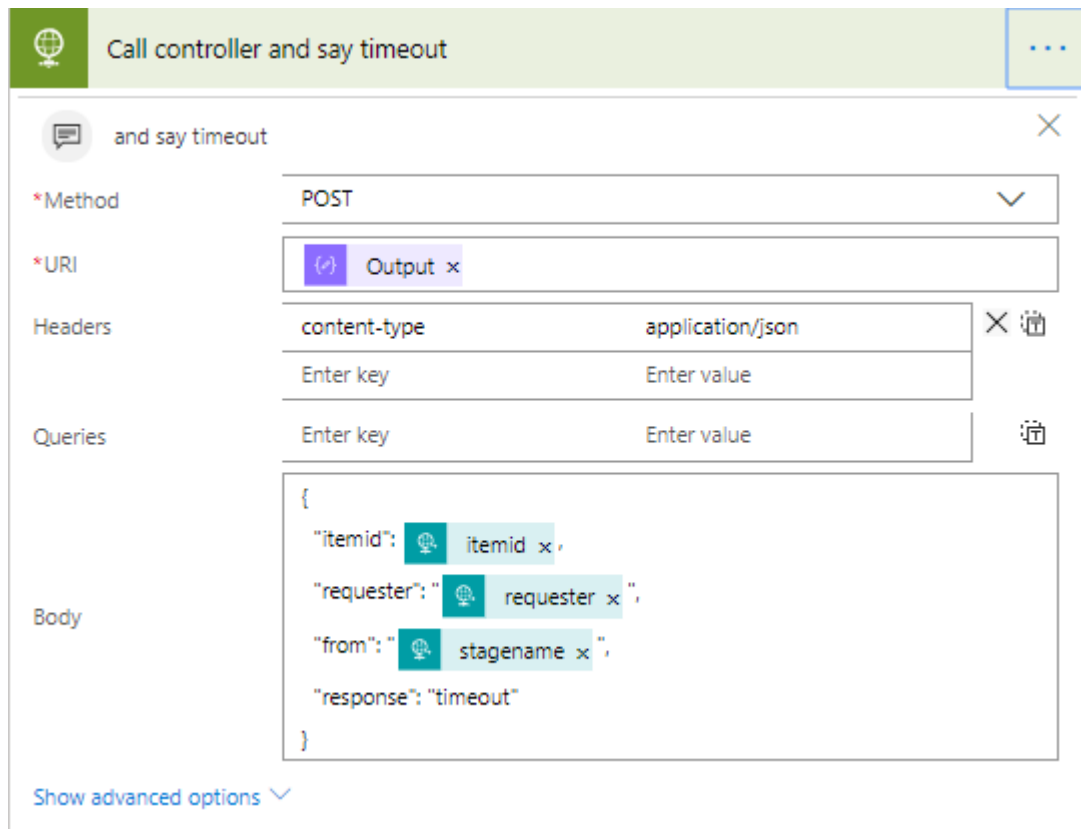


7. Select the "Scope didn't react on time" **Configure run after**:

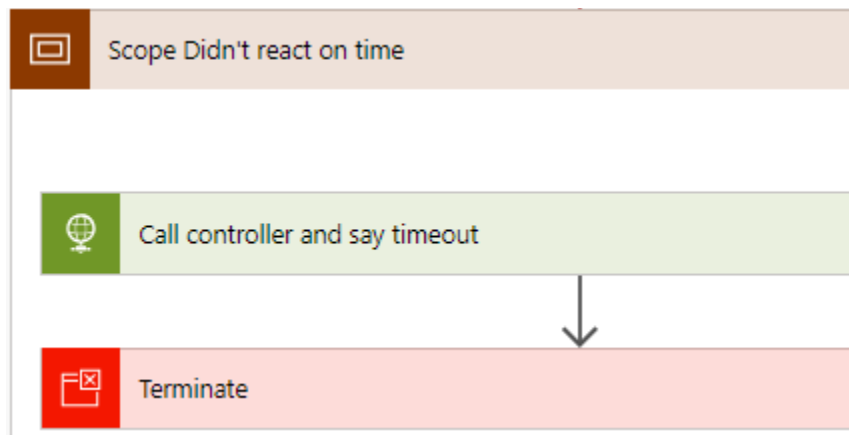




8. Check **has timed out**.
9. In the same scope add an **HTTP** action and set its settings as following:



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



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

The screenshot shows the configuration for an HTTP action named "Call controller and approved or rejected". The configuration is as follows:

- Method:** POST
- URI:** Output
- Headers:**
  - content-type: application/json
  - Enter key: Enter value
- Queries:**
  - Enter key: Enter value
- Body:**

```

{
  "itemid": itemid,
  "requester": requester,
  "from": stagename,
  "response": toLower(...)
}

```

At the bottom, there is a link to "Show advanced options".

The "response" parameter expression is the following:

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

12. Save the flow.

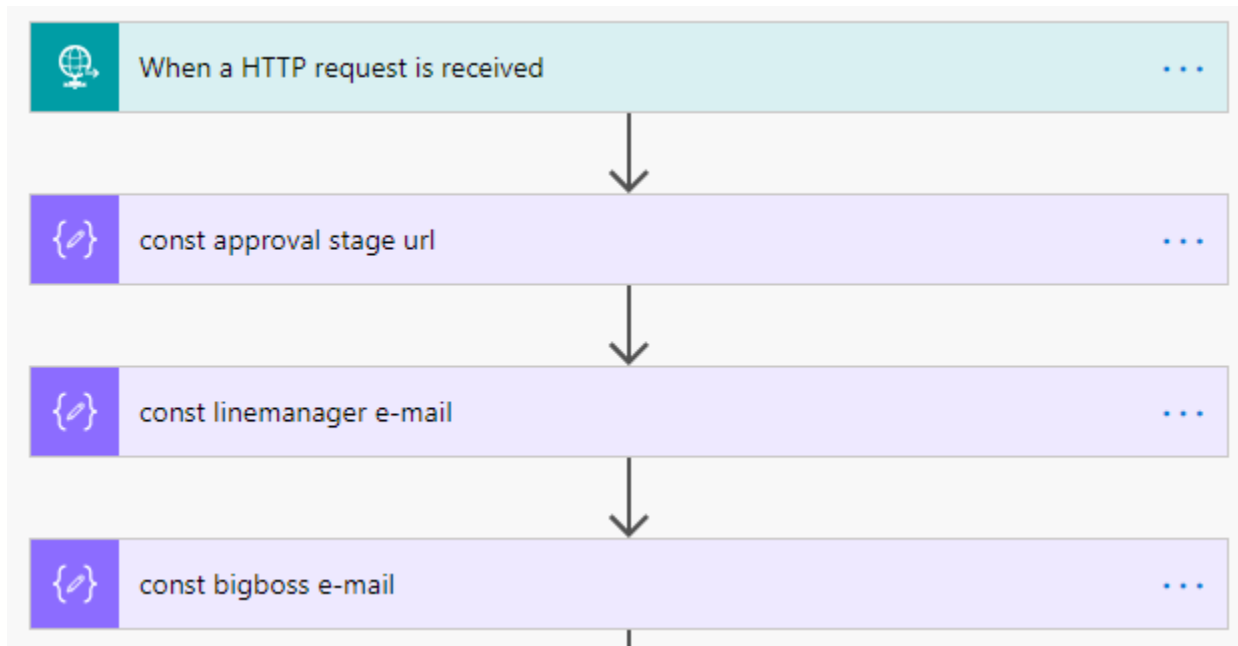
## Approval Controller flow implementation

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 **Array**:

Initialize const stages	
* Name	stages
* Type	Array
Value	<pre>[   "linemanager",   "bigboss" ]</pre>

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

The screenshot shows the 'Get item' action configuration. The 'Id' field is set to 'itemid'. A red arrow points from the 'triggerBody()['itemid']' expression in the 'Dynamic content' pane to the 'itemid' field.

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

The screenshot shows the 'Update expense status' action configuration. The 'Id' field is set to 'itemid', the 'Title' field is set to 'Title', and the 'Status Value' is set to 'Being evaluated'. Red boxes highlight the 'Id' and 'Title' fields, and the 'Status Value' dropdown.

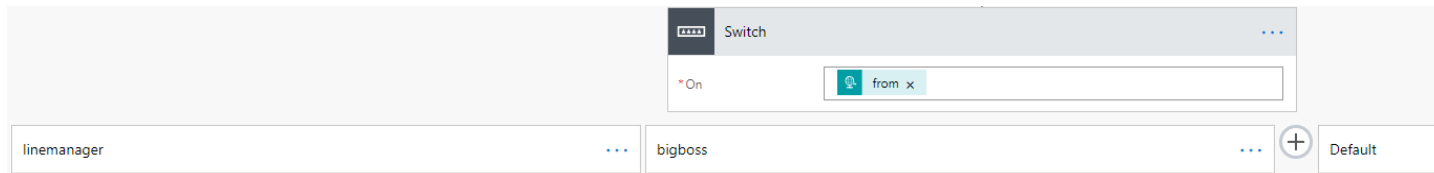
10. After the Update expense status, add a **Switch** action and check the **from** value:

The screenshot shows the 'Switch' action configuration. The 'On' field is set to 'from'.

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

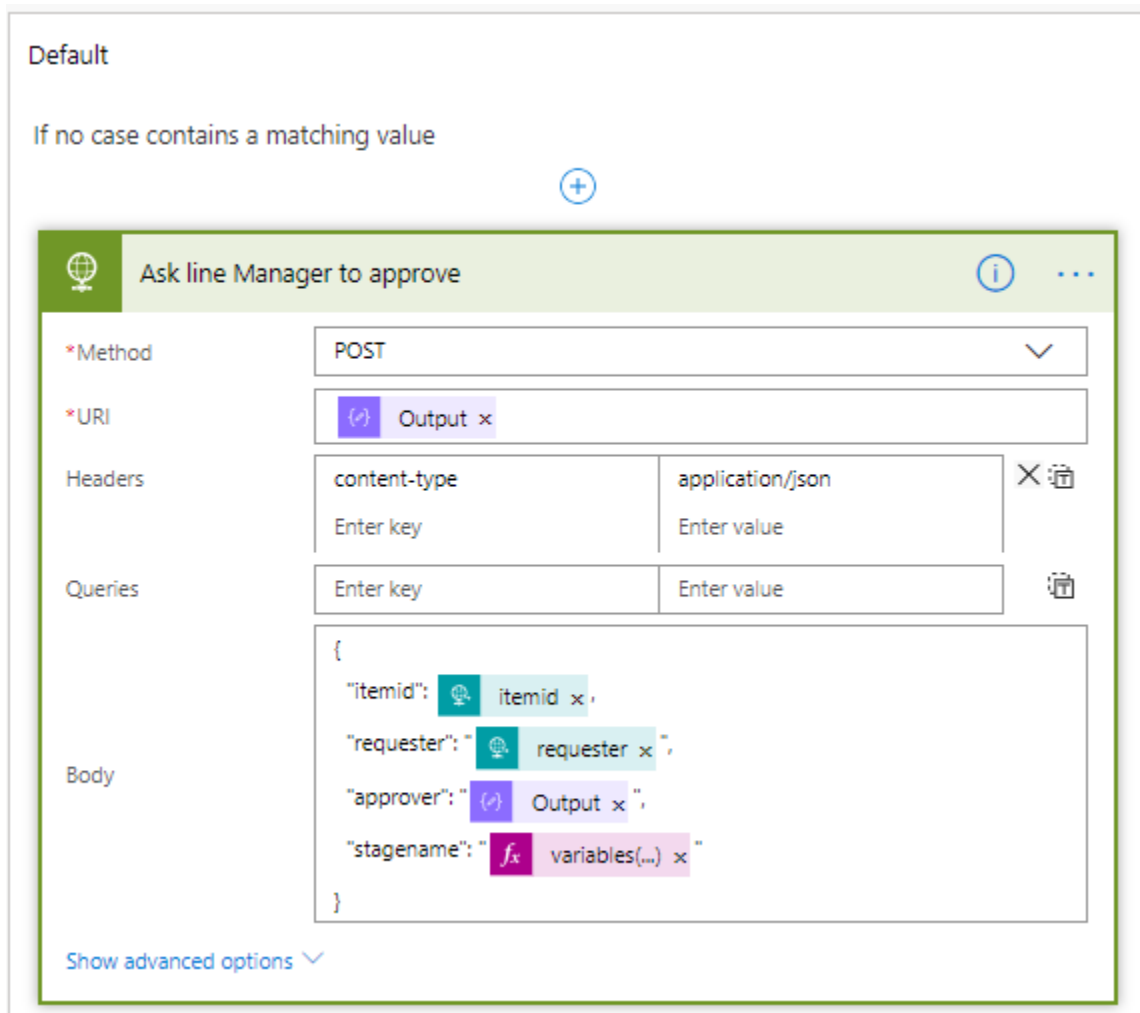
12. In the switch, we will analyze 3 scenarios:

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):

The screenshot shows a Microsoft Flow interface for a branch named 'bigboss'. The branch condition is set to 'Equals' with the value 'bigboss'. The branch has two paths: 'If yes' (green) and 'If no' (red). The 'If yes' path contains an action 'Call big boss again' (green). The 'If no' path is currently empty, with a button 'Add an action' visible. A configuration box for the condition is shown, displaying 'response' is equal to 'timeout'.

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

✓

If yes

+

🌐

Call big boss again

ⓘ ...

\*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": "fx variables(...) ×"
}
```

Show advanced options

▼

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

- 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 x

is equal to



timeout



Add



If yes



Call Big boss



\*Method

POST



\*URI



Output x

Headers

content-type

application/json



Enter key

Enter value

Queries

Enter key

Enter value



Body

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

Show advanced options



If no

linemanager

\* Equals

linemanager



check linemanager response

\* On



response x

Case



\* Equals

timeout



Escalate to Big Boss



\* Method

POST



\* URI



Output x

Headers

Enter key

Enter value



Queries

Enter key

Enter value

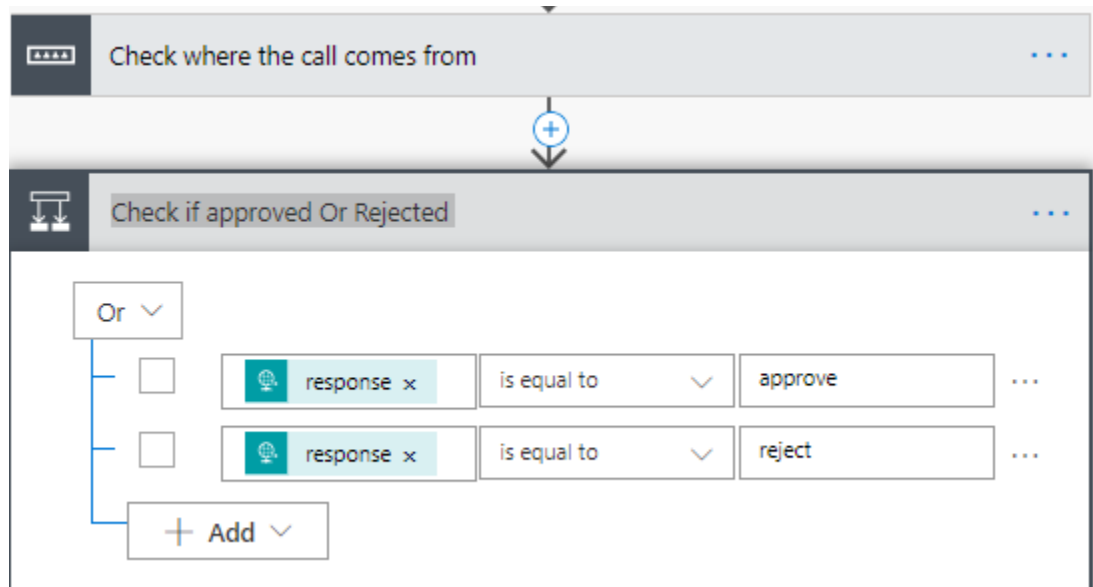


Body

```
{
  "itemid": "itemid x",
  "requester": "requester x",
  "approver": "Output x",
  "stagename": "fx variables(...) x"
}
```

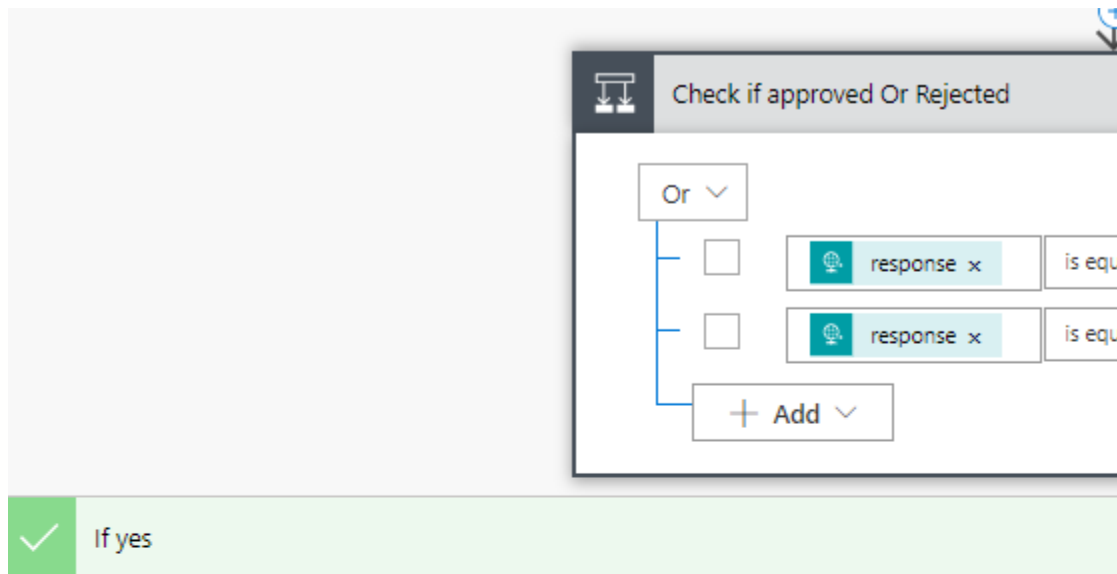
Show advanced options v

- 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:





**Update item approval status**

\*Site Address: Flowdemo1 - https://redwoods1.sharepoint.com/sites/Flowdemo1 ▾

\*List Name: Travels ▾

\*Id: itemid ×

\*Title: Title ×

Amount:

Status Value: if(...) ×

[Add dynamic content](#)

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.

Settings for 'Start an approval'

**Timeout**  
Limit the maximum duration an asynchronous pattern may take. Note: this does not alter the request timeout of a single request.

Duration ⓘ

**Retry Policy**  
A retry policy applies to intermittent failures, characterized as HTTP status codes 408, 429, and 5xx, in addition to any connectivity exceptions. The default is an exponential interval policy set to retry 4 times.

Type

**Tracked Properties**

Properties

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 Vision 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 is 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

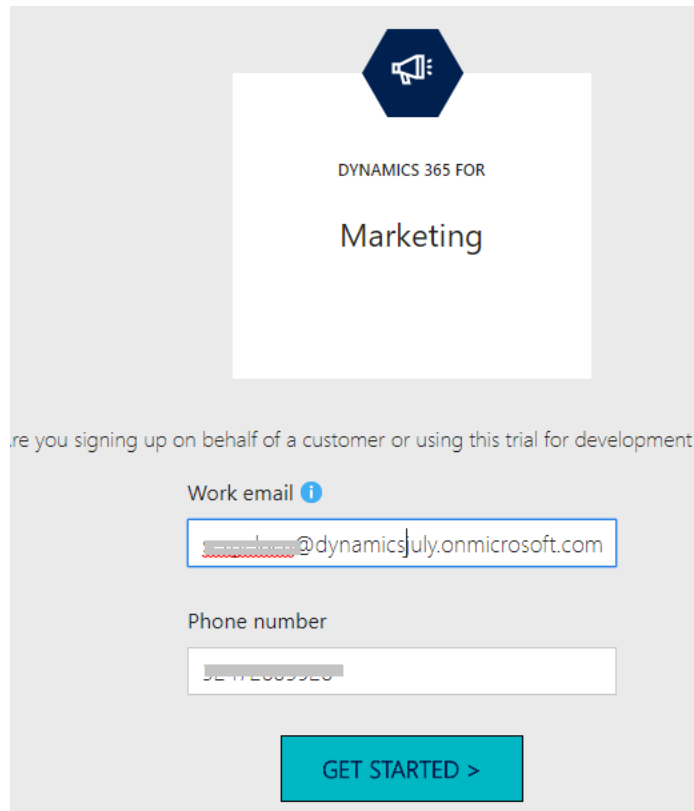
**Scenario:**

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.

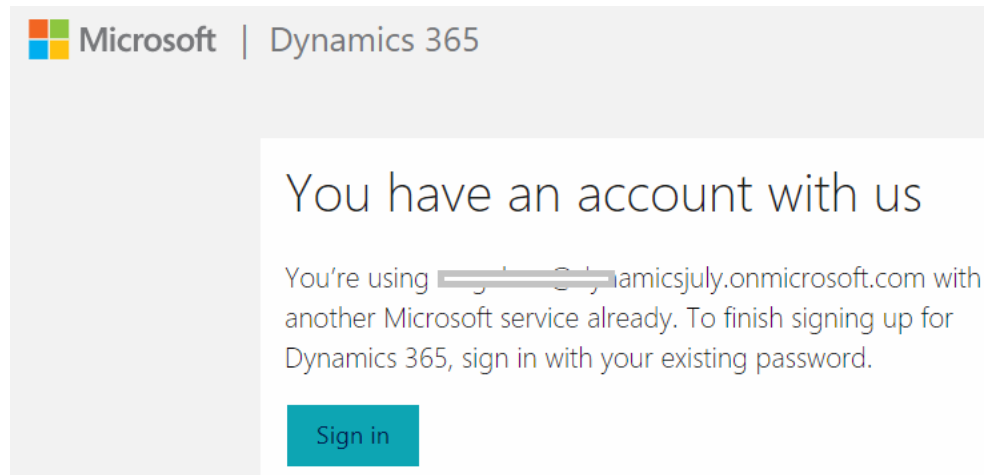
As a power company, there are many customer requests that can be made including reporting a power outage, billing queries, street light replacement and vegetation brushing to name a few. Some of these requests are higher priority than others so it is important we understand the user's intent when we 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](https://luis.ai) and a Dynamics 365 trial: <https://trials.dynamics.com/> (use Dynamics 365 for Customer Service).



The screenshot shows the Dynamics 365 Marketing sign-up page. At the top, there is a blue hexagonal icon with a white megaphone. Below it, the text "DYNAMICS 365 FOR" is displayed in a small, dark font, followed by "Marketing" in a larger, bold, dark font. Below this, there is a line of text: "Are you signing up on behalf of a customer or using this trial for development purposes?". Underneath, there are two input fields. The first is labeled "Work email" with a blue information icon to its right. The email address entered is "july@ynamicsjuly.onmicrosoft.com". The second input field is labeled "Phone number" and contains the text "321-234-5678". At the bottom of the form is a blue button with the text "GET STARTED >".



The screenshot shows the Dynamics 365 sign-in page. At the top, the Microsoft logo is followed by the text "Microsoft | Dynamics 365". Below this, the heading "You have an account with us" is displayed in a large, dark font. Underneath, there is a paragraph of text: "You're using july@ynamicsjuly.onmicrosoft.com with another Microsoft service already. To finish signing up for Dynamics 365, sign in with your existing password." Below this text is a blue button with the text "Sign in".

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

## Almost there

You're signed in as [serge.luca@shareql.com](#) on [dynamicsjuly.onmicrosoft.com](#)

When using an organization email address (e.g. an employer or school), the organization may assume control over your account and data. [Learn more.](#)

By choosing **Create**, you agree to our [terms and conditions](#) and understand that your name, email address, and trial organization name will be visible to other people in your institution. [Microsoft Privacy Policy](#)

Create

### Additional Resources:

For More Microsoft Flow Resources Head over to [AKA.MS/Flow-Resources](https://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.