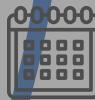


WELCOME TO

**Class: Power Automate (Advanced)
Intensive Workshop**



5 Nov 2025



9.00AM – 4.00PM

Instructor: K. Phakkaphong Krittawat

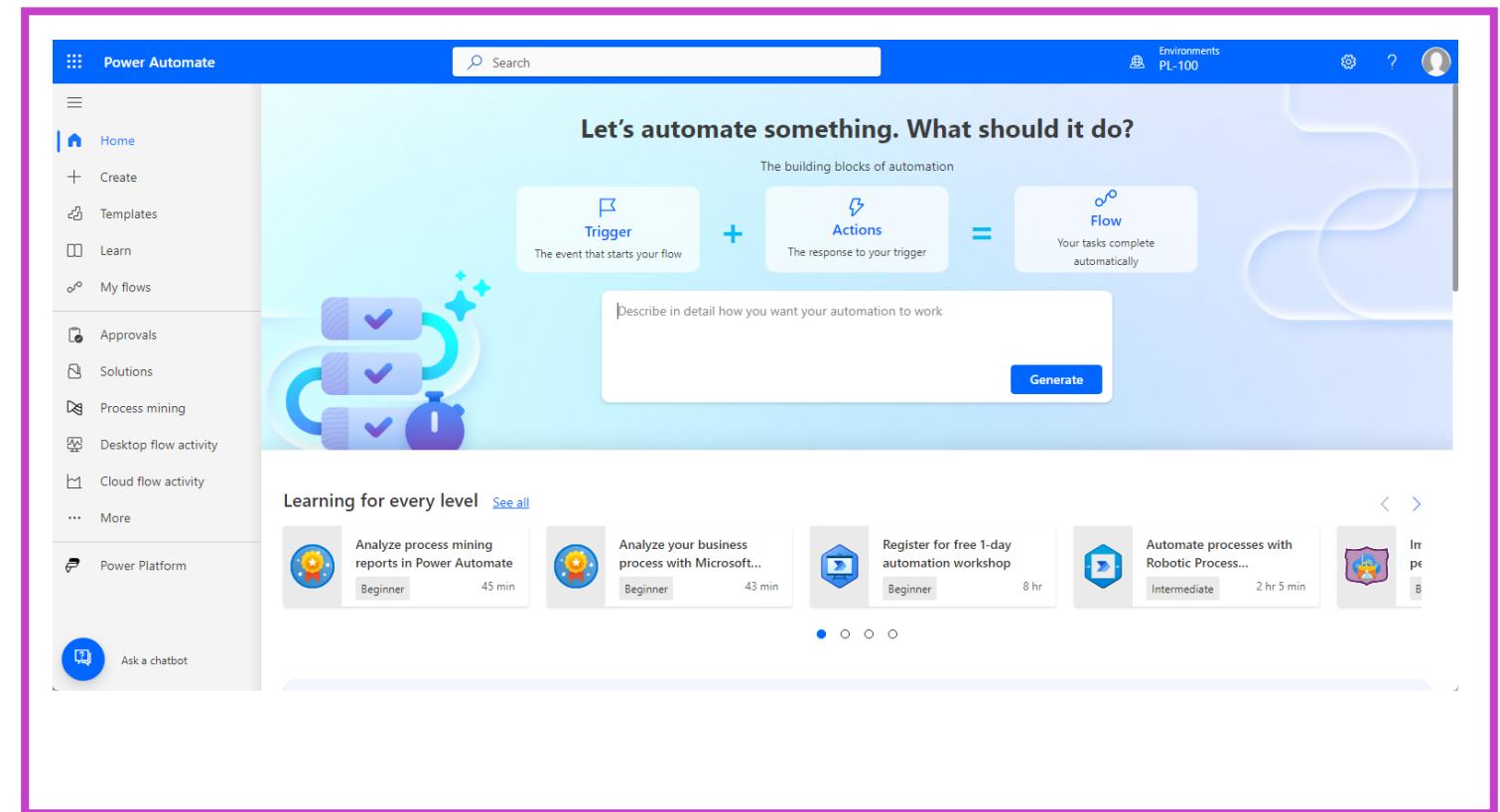


Module 1

Building Flows from Scratch

What is Power Automate?

Power Automate is an online workflow service that automates actions across the most common apps and services



What can you do with Power Automate?

Power Automate automates workflows between your favorite applications and services, sync files, get notifications, collect data, and more

For example, you can automate these tasks:

Instantly respond to high-priority notifications or emails

Capture, track, and follow up with new sales leads

Copy all email attachments to your OneDrive for Business account

Collect data about your business, and share that information with your team

Automate approval workflows



TRAINOCATE

Key concepts

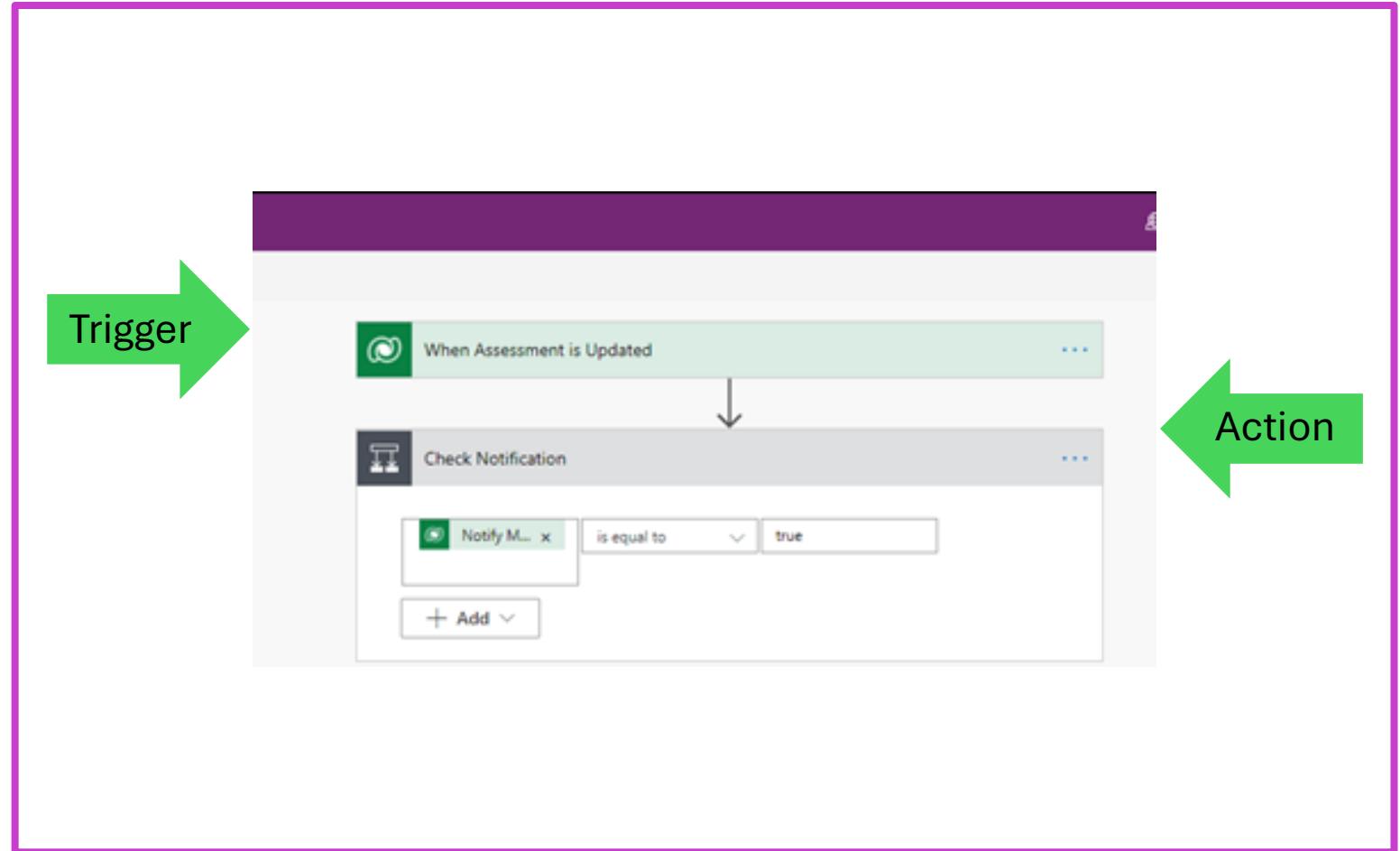
Every cloud flow has two main parts:

A Trigger

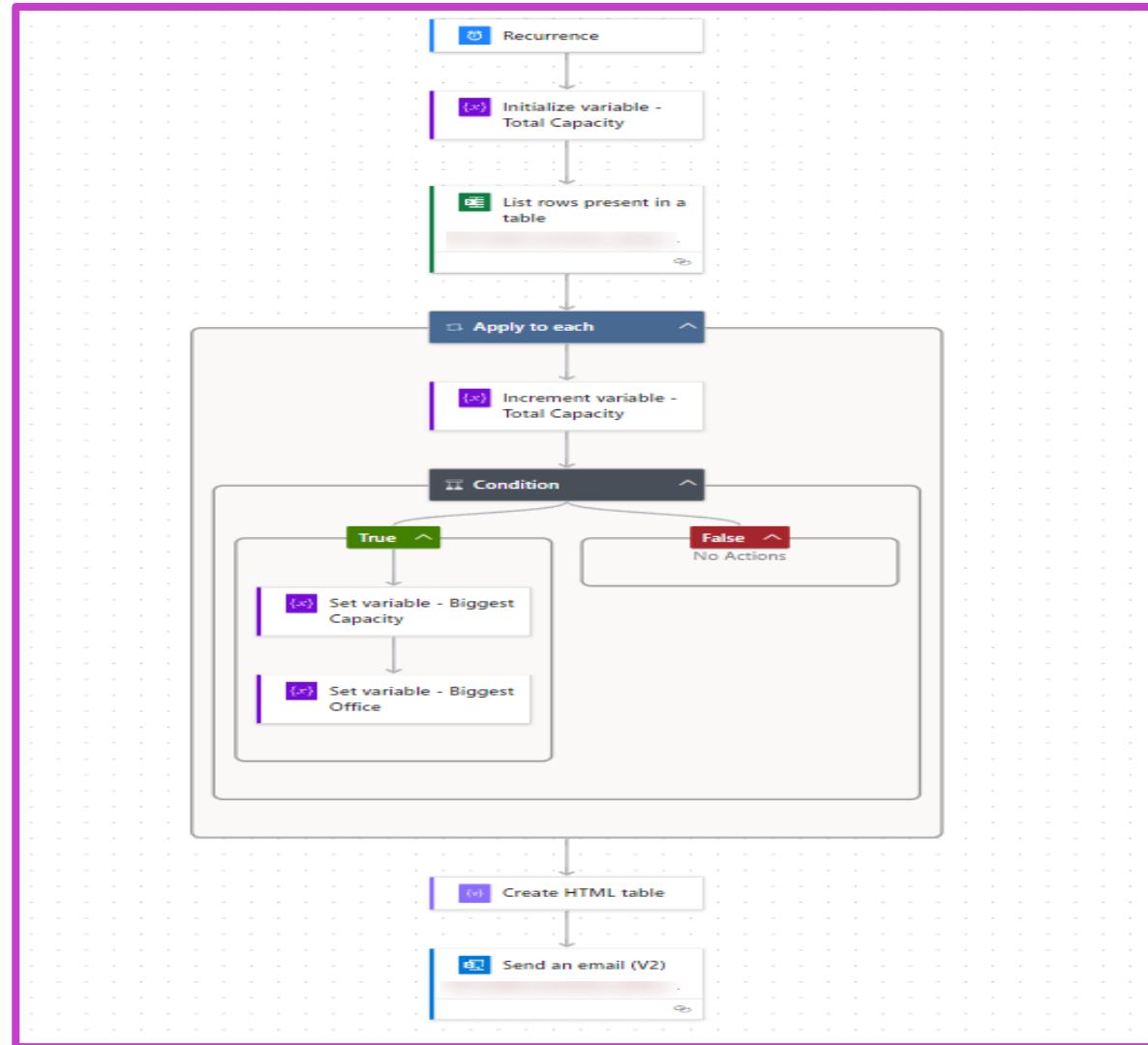
The starting action of the flow such as new email arriving in your inbox or a new item being added to a SharePoint list

One or more Actions

What you want to happen when a trigger is invoked such as start the action of creating a new file on OneDrive for Business



Example – Follow up on a message



Flow building fundamentals

Dynamic data

Variables

Loops

Error handling

Conditions

Expressions



Ways to start a cloud flow

With Copilot

From a template

From blank

Create a cloud flow

Start with Copilot

- Build flows through conversation

Let's automate something. What should it do?

Get started by selecting an example or describing your own automation idea.

Every month, copy all files from OneDrive folder to another OneDrive folder

Copy all rows from an Excel file to another excel file with a click of a button

When a new item is created in SharePoint, send me an email

Describe in detail how you want your automation to work

Generate

Create a cloud flow

Templates provide a quick start for common scenarios

Start from a template [\(i\)](#)

[Search all templates](#)

Top picks Remote work Email Notifications Save to cloud Approval



Follow up on a message
By Microsoft

Instant 25322



Create a task in Planner from a message
By Microsoft

Instant 24800



Notify a channel when the status of a task in Planner updates
By Microsoft

Automated 19616



Start an approval in Teams when a file is added to a SharePoint folder
By Microsoft

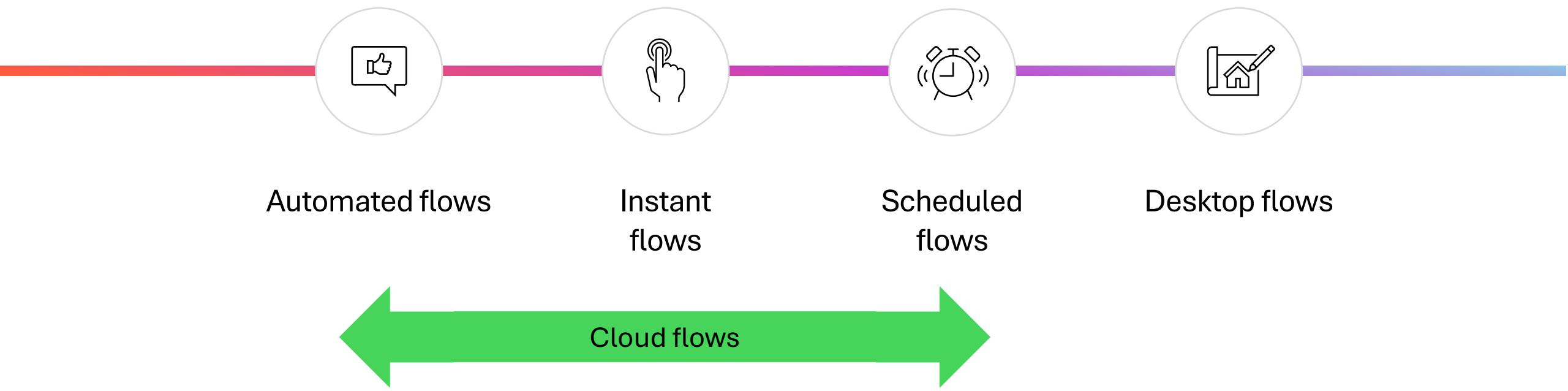
Automated 19200

>

[All templates →](#)

Power Automate

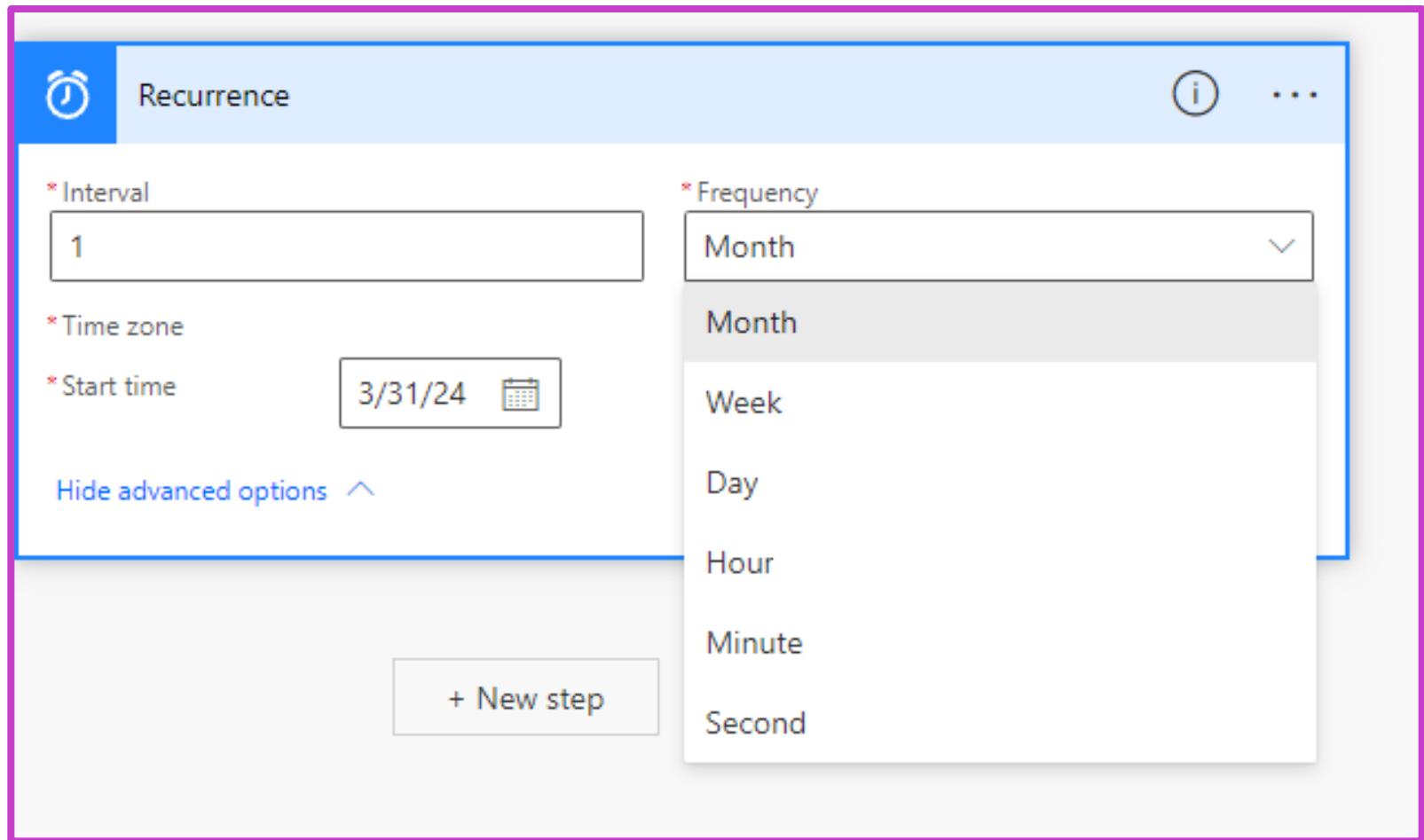
Types of Power Automate flow



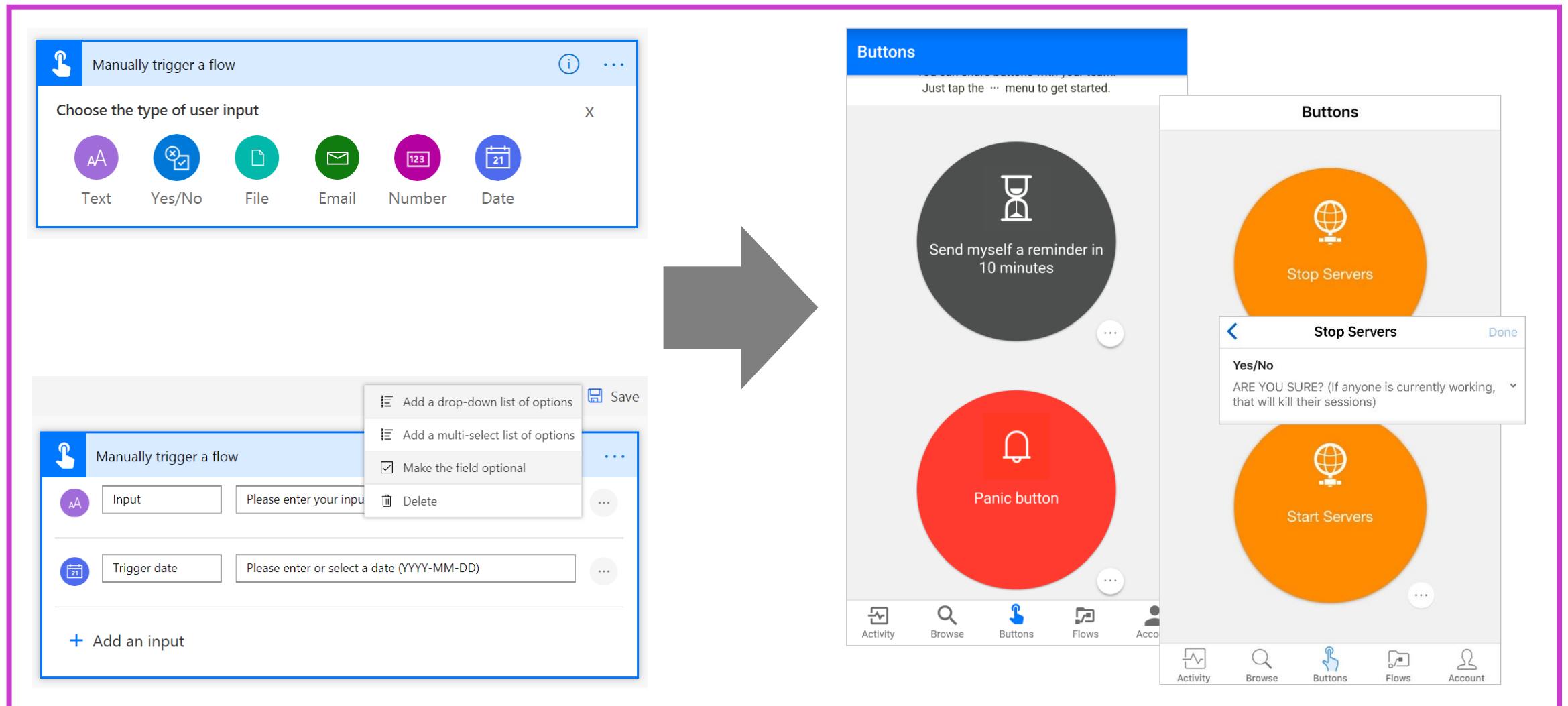
Create a scheduled flow

Recurrence trigger

- Runs on a pre-defined schedule
- Can also run manually from Power Automate portal



Create an instant flow



Create an automated flow

Event-based trigger

- Most connectors have triggers
- Tabular connectors
 - Created/Added
 - Updated/Modified
 - Read/Get
 - List/Guery
 - etc
- Function based triggers
 - Vary by connector

Triggers Actions

 When an item is created
SharePoint

 When an item is created or modified
SharePoint

 When a file is created in a folder (deprecated)
SharePoint

 For a selected file
SharePoint

 For a selected item
SharePoint

 When a file is classified by a Microsoft Syntex model
SharePoint

 When a file is created (properties only)
SharePoint

 When a file is created or modified (properties only)
SharePoint

 When a file is deleted
SharePoint

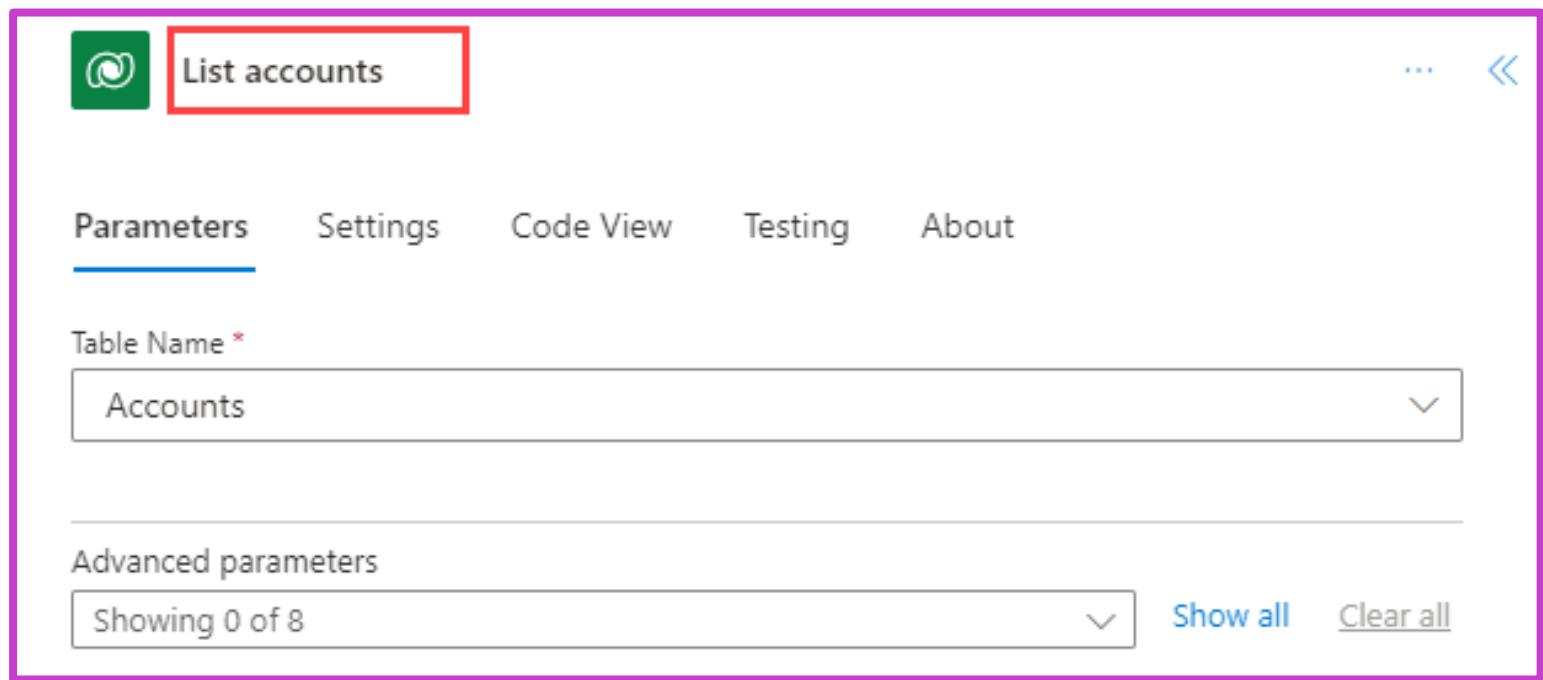
Naming

Naming flows

Most important for instant flows, since they will be selected by end users from a list of flows

All other flows will need clear names for administrators to tell what has run/failed

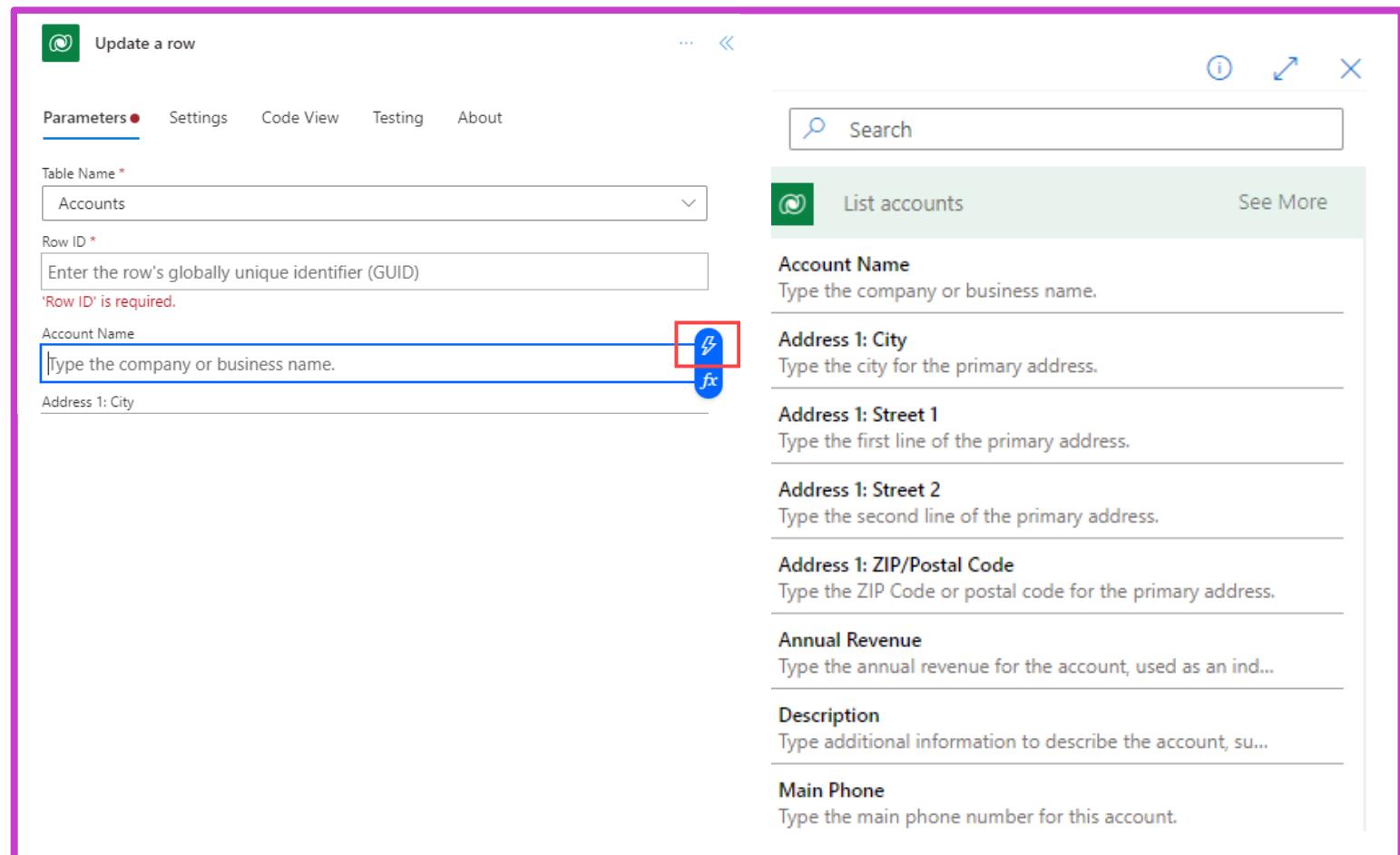
Naming flow actions
Every action gets default name



Dynamic data

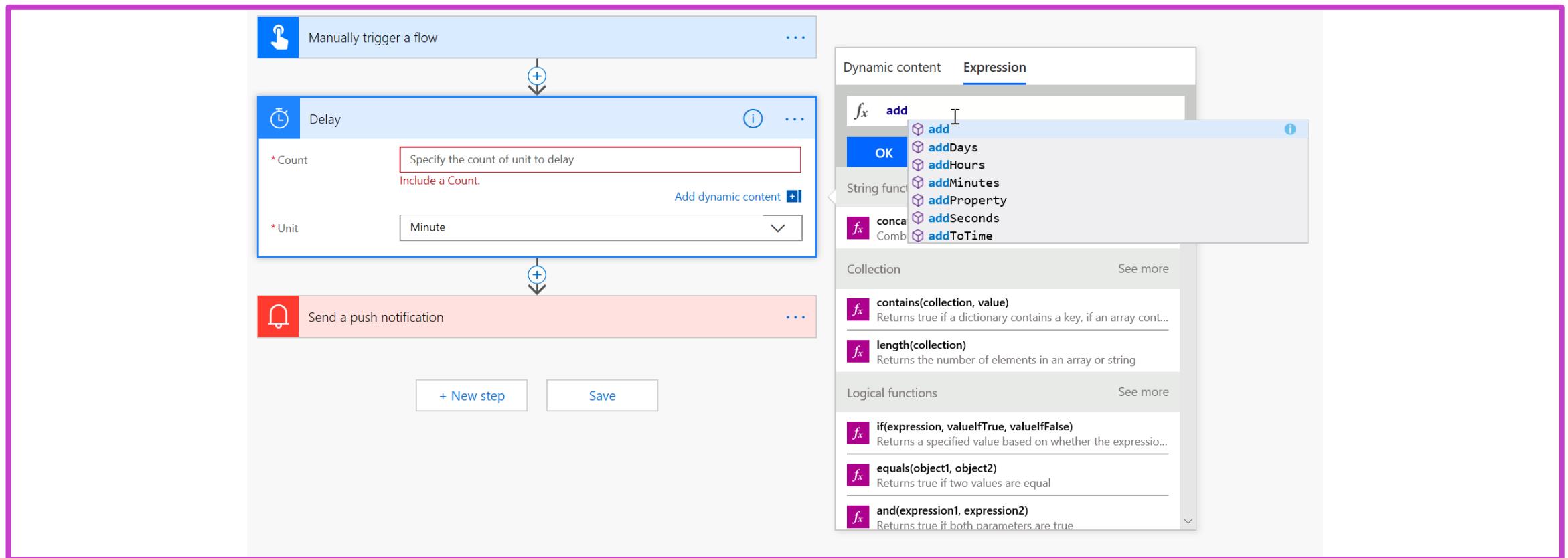
Use dynamic content panel to reference data from prior steps in a flow and dynamically bind it to current properties

Dynamic content panel shown when clicking in a property value on a step flow



Expressions

To write an expression in Power Automate, select a field to open the Dynamic content menu and then select the Expression tab.



Many reasons to use expressions



Change Data Type



Do Simple Math or Text Work



Create Dynamic Values (random, GUID,...)



Check for Empty Data



Make Decisions ("If" conditional statements)



Work with Data Sets (lists)



TRAINOCATE

Types of expressions

Expressions are grouped into 10 different categories like math and logic.

The image shows two side-by-side screenshots of a dynamic content editor interface. Both screenshots have a purple border.

Left Screenshot: A modal window titled "Add an expression to do basic things like access, convert, and compare values. [Learn more about dynamic content.](#)" is displayed. It has tabs for "Dynamic content" and "Expression". The "Expression" tab is selected. Below it is a search bar with "fx |" and an "OK" button. A red box highlights the "String functions" category, which contains the "concat" function. Other categories shown include "Collection" (with "contains" and "length" functions), "Logical functions" (with "if", "equals", and "and" functions), and "See more" buttons.

Right Screenshot: A larger window titled "Dynamic content" with the "Expression" tab selected. It shows a list of string functions under the heading "String functions". The "concat" function is highlighted with a blue background. Other functions listed are "substring", "replace", "guid", "toLowerCase", and "toUpperCase". Each function has a brief description and an "fx" icon.

Core expressions

- **outputs(stepName)** – Returns object with outputs from a step
- **body(stepName)** – Returns object with properties from a step
- **triggerBody()** – Like body() but for trigger output
- **item()** – Used to reference the current item in Apply to each
- **variables(variableName)** – Reference a variable
- **parameters(parameterName)** – Environment variable

Handling null values

Trying to use a property that's not set, "null", will cause the flow to fail

Two steps to handle

- Use the "?" character after selecting the property (this happens automatically)
- Then, use the coalesce() function to provide a default value

```
coalesce(body('Get_record')?['content'],'Default Value')
```

Data Types and Formats

Text – aka Strings

- Normal – ‘Example string’
- Email, URL, phone number, etc...
- Base64
- Binary content
- Data URI
- URI component

Non-text

- Floating-point number – 9.0
- Integer number – 137
- Boolean – True or false
- Array – A list of items
- Object – A record with key-value pairs
- XML Content

For strings use single quotes

Do *not* use single quotes for numbers, integers or Booleans: ‘true’ does *not* equal true

Working with Dates

- No date time datatype
- Date time is in ISO 8601 format (json)
 - 2022-11-30T20:46:27+10:00
- Date Time connector
- Or expressions
- Date parameter includes only date without time or timezone. To compare parameter dates
 - < ≤ ≥ > == ≠ all bad as you're comparing strings
 - Convert to ticks
 - Use substring to restrict what we are comparing, e.g., first 10 characters would give you the date only

Add an action X

[Return to search](#)

 Date Time

- Add to time [In App](#) i
- Convert time zone [In App](#) i
- Current time [In App](#) i
- Get future time [In App](#) i
- Get past time [In App](#) i
- Subtract from time [In App](#) i

 Time

-  [addToTime](#)
-  [convertTimezone](#)
-  [getPastTime](#)
-  [getFutureTime](#)
-  [parseDateTime](#)

Preparing Dates for Display

The screenshot shows a user interface for a "Convert time zone" tool. At the top left is a blue button with a gear icon labeled "Convert time zone". Below it is a navigation bar with tabs: "Parameters" (which is selected), "Settings", "Code View", and "About".

The main area contains several input fields:

- "Base Time *": A dropdown menu showing "Created On" with a delete icon.
- "Source Time Zone *": A dropdown menu showing "(UTC-08:00) Coordinated Universal Time-08" with a dropdown arrow.
- "Destination Time Zone *": A dropdown menu showing "(UTC-07:00) Mountain Time (US & Canada)" with a dropdown arrow.
- "Time Unit": A dropdown menu showing "Short date pattern - 6/15/2009 [d]" with a dropdown arrow.

To the right of these fields is a large, semi-transparent dropdown menu listing various date/time patterns:

- Full date/time pattern (long time) - Monday, June 15, 2009 1:45:30 PM [F]
- Full date/time pattern (short time) - Monday, June 15, 2009 1:45 PM [f]
- General date/time pattern (long time) - 6/15/2009 1:45:30 PM [G]
- General date/time pattern (short time) - 6/15/2009 1:45 PM [g]
- Long date pattern - Monday, June 15, 2009 [D]
- Long time pattern - 1:45:30 PM [T]
- Month/day pattern - June 15 [m]
- RFC1123 pattern - Mon, 15 Jun 2009 20:45:30 GMT [r]
- Round-trip date/time pattern - 2009-06-15T13:45:30.0000000-07:00 [o]
- Short date pattern - 6/15/2009 [d]** (This pattern is highlighted with a gray background)
- Short time pattern - 1:45 PM [t]
- Sortable date/time pattern - 2009-06-15T13:45:30 [s]
- Universal full date/time pattern - Monday, June 15, 2009 8:45:30 PM [U]
- Universal sortable date/time pattern - 2009-06-15 13:45:30Z [u]
- Year month pattern - June, 2009 [y]
- [Enter custom value](#)

Examples Using Expressions with Dates

Get days since account was created

```
int(  
    div(  
        sub(  
            ticks(utcNow()),  
            ticks(triggerBody()?['entity']?['createdon']))),  
        864000000000  
)
```

Check if date is between business hours

```
and(  
    greaterOrEquals(  
        formatdatetime(utcnow(), 'HH:ss'),  
        formatdatetime('7:00', 'HH:ss')),  
    lessOrEquals(  
        formatdatetime(utcnow(), 'HH:ss'),  
        formatdatetime('15:00', 'HH:ss'))  
)
```

Escaping Rules

- 1 When you are directly using the expression language you may need to escape certain characters
- 2 Using strings with a single quote character

```
@substring('It''s A Great Day!',1,5)
```

- 3 Using spaces in action names: @body('Name_with_spaces')
- 5 Building an object you need to escape '@'

- {
- “@odata.type”: “”
- }

Common expressions

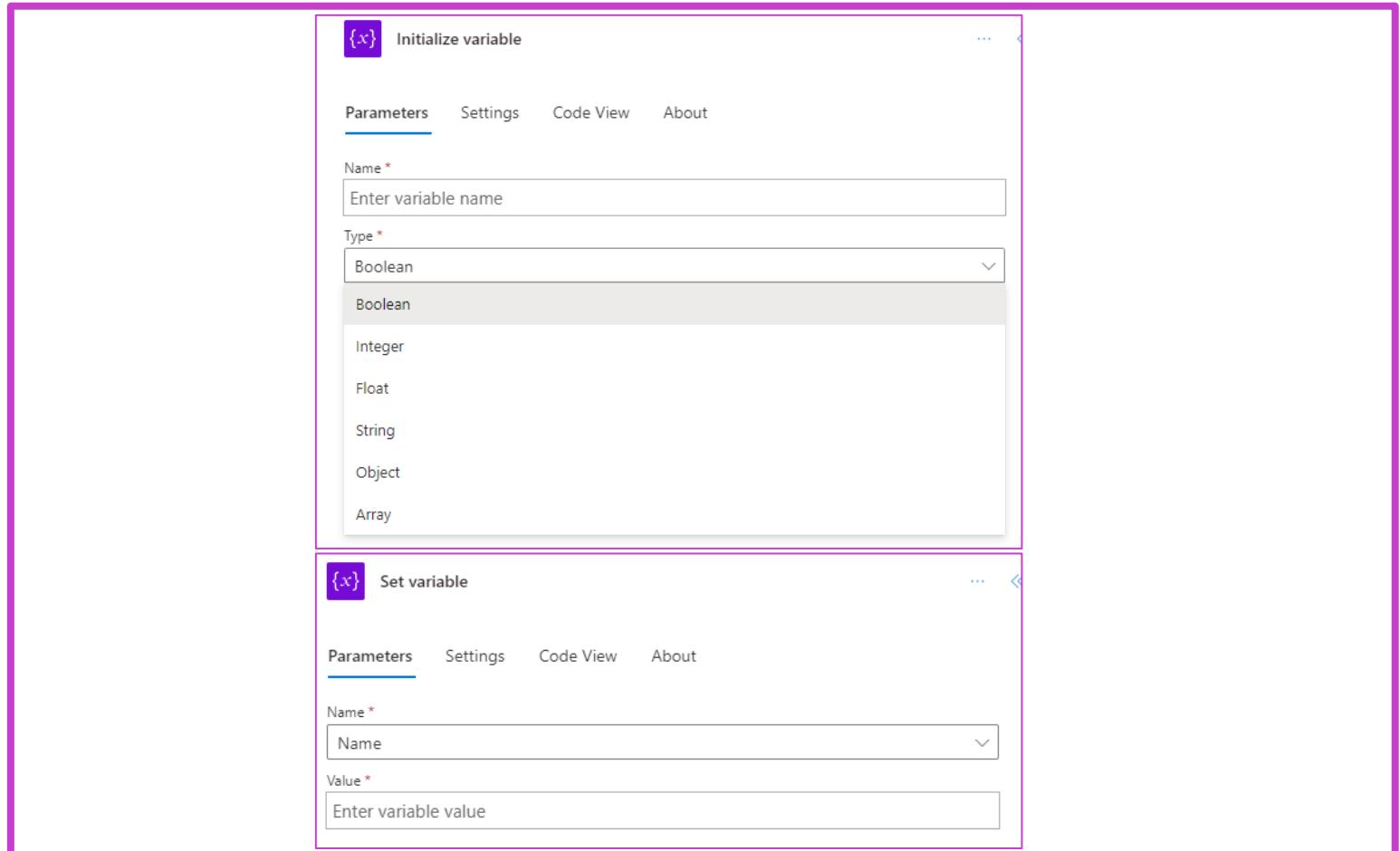
- **coalesce(<object_1>, <object_2>, ...)** – Return first non-null value
- **guid()** – Generate a GUID
- **replace(string, old, new)** – Replace old with new in string
- **equals(left, right)** – Returns true if left equals right
- **utcnow('yyyy-mm-dd')** – Generate a date/time
- **ticks('<timestampe>')** – Returns the number of ticks since January 1, 0001
- **string()** – Convert to plain/text
- **json()** – Convert to application/json – Can parse like JSON
- **xml()** – Convert to application/xml
- **xpath(<xml>, <expression>)** – Execute Xpath expression
- **if(<condition>, <true>, <false>)** – Set value based on condition
- **result(<scope>)** – Return the run result for a scope of actions

Variables

Initialize at top of flow and select data type

Set during flow

Reference using Dynamic data



Module 2: JSON in Workflows

The Core of Power Automate

JSON

- A data storage format that is easy for computers to understand.
- Comparable to an Excel table written in code.
- A programming language used for communication between various systems.
- Every **Action** in **Power Automate** uses **JSON**.

JSON Basics: Objects

- **Key-Value** Pair Storage
- **Example:**

```
{  
  "name": "สมชาย" ,  
  "age": 20,  
  "city": "กรุงเทพ"  
}
```

- "name" = **Key** (ชื่อของข้อมูล)
- "สมชาย" = **Value** (ค่าของข้อมูล)

JSON Basics: Arrays

- A list of multiple data items, arranged in a sequence
- **Simple Array :**

["แอปเปิล", "ส้ม", "กล้วย"]

- **Array of Objects:**

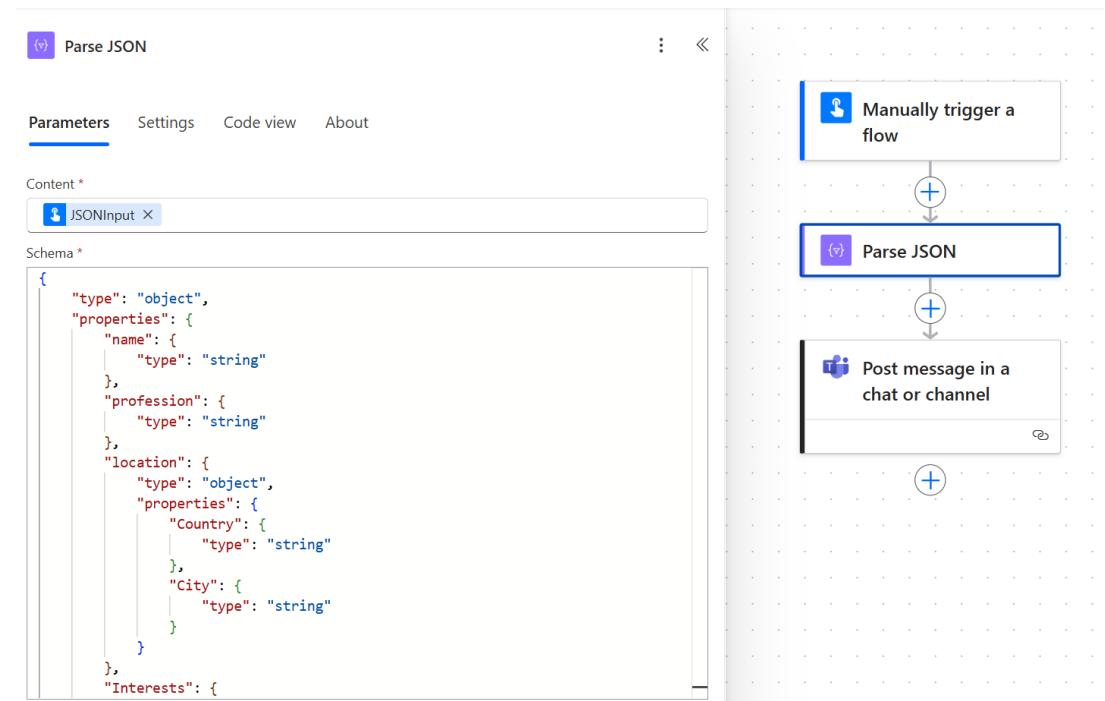
```
[  
  { "id": 1 , "name": "สมชาย" } ,  
  { "id": 2 , "name": "สมหญิง" }  
]
```

Key-Value ໃນ Power Automate

- Field Name = **Key**
- Dynamic Content = **Value**

Parse JSON Action

- **Common Issues:**
 - **JSON data is often a String (text)**
 - **It must be converted into an Object (data structure) before use**
- **The Method (in Power Automate):**
 1. Use the **Parse JSON Action**
 2. Input the **JSON String** to be converted
 3. Define the **Schema (Structure)**



Parse JSON (Array)

1. Steps (Configuration):

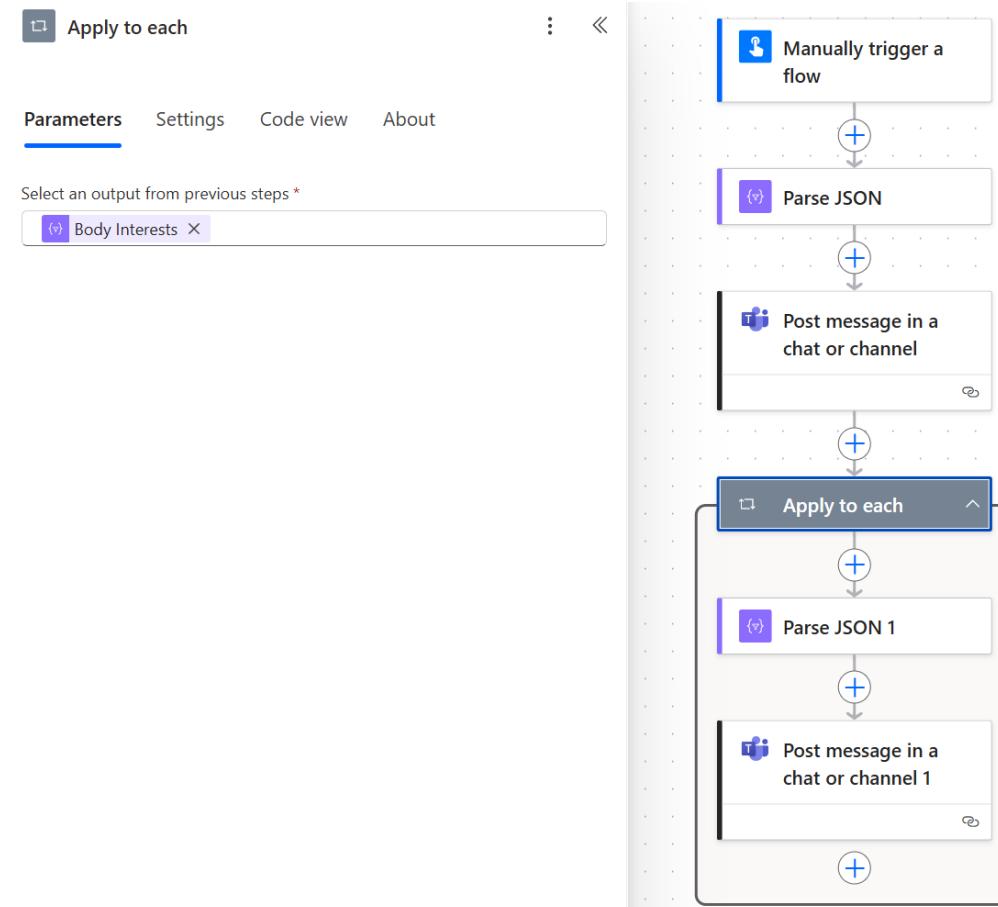
- Use the **Parse JSON Action**.
- Input the JSON String into the **Content field**.
- **Generate Schema** from a sample payload.

2. After Parse JSON:

- `body('Parse_JSON')?['items']`
(Array of items)

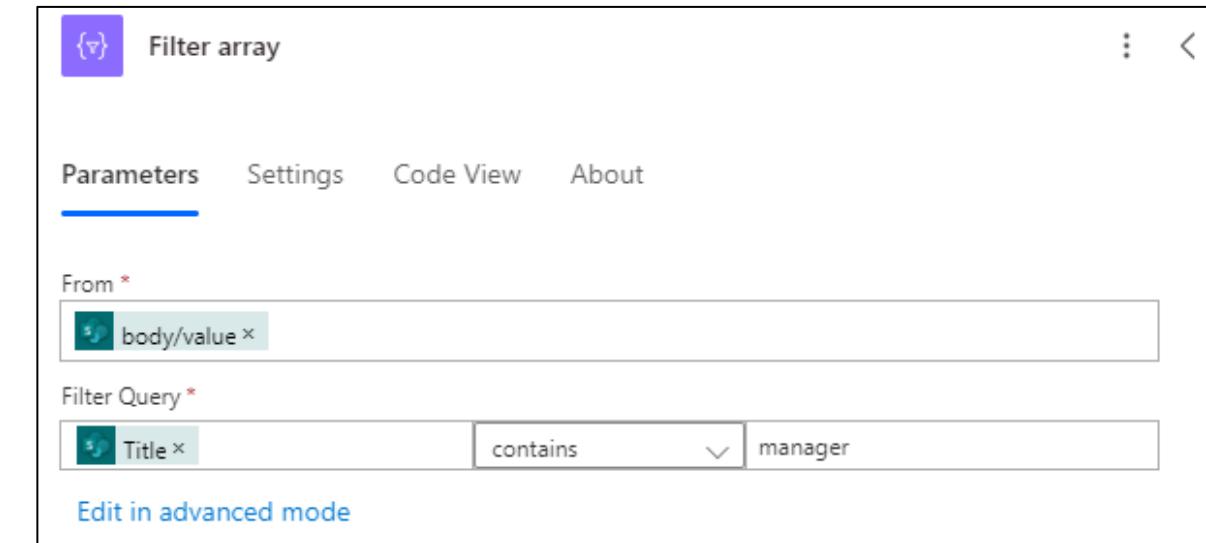
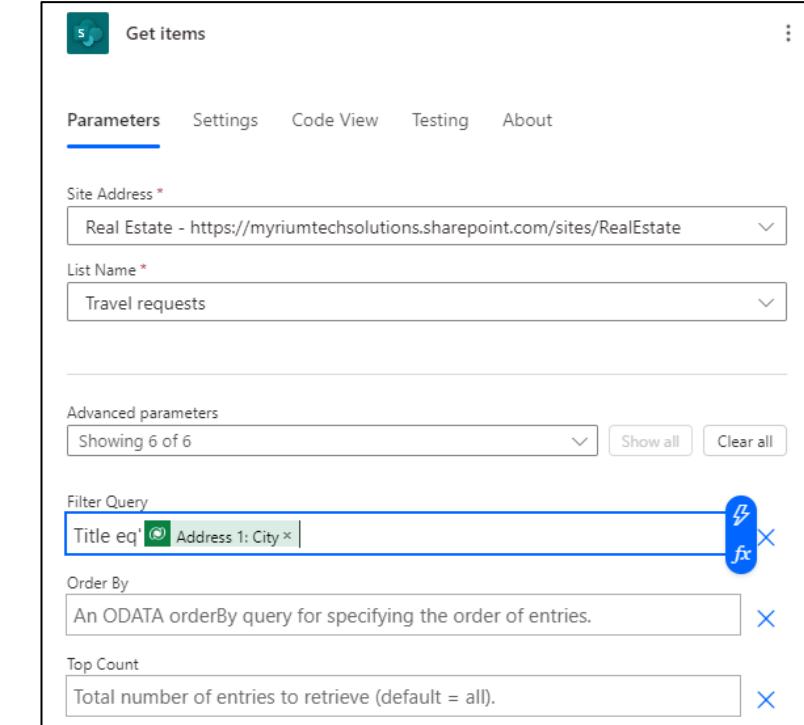
3. extraction Operation:

- Extract items



Filtering

- Have the connector do the filtering for you – **Recommended**
- Built-in flow action – Filter array
 - Select the array in the From field
 - Can either use simple or advanced mode just like for conditions
- If you need the first N items you can use take() or skip()



Selecting Arrays

Two input modes: Fill key-value pairs, or typing directly

Create an array of objects

The screenshot shows the 'Select' action configuration. The 'From' field is set to 'body/value'. The 'Map' field contains a table:

Title	Value
Travel Reason	Reason for travel
Enter key	Enter value

Useful for passing this array to another action

Create an array strings, numbers, Booleans etc.

The screenshot shows the 'Select' action configuration. The 'From' field is set to 'Body'. The 'Map' field contains the following JSON object:

```
{  
  "Title" : "Title"  
}
```

Useful for getting a simple list, for example, of email addresses

Convert the List to a String

- **Join** – Use join to get a simple list, for example, if you have a list of email addresses*
- **Create HTML/CSV table** – Convert a list of objects to a tabular display format, for example, for inclusion in the body

* **Note:** You must have a list of strings, not a list of objects

The image displays two separate instances of the Microsoft Power Automate 'Send an email' action configuration screen. Both instances show the 'Parameters' tab selected.

Top Screenshot (Join Example):

- To:** Specify email addresses separated by semicolons like someone@contoso.com
- Subject:** Specify the subject of the mail
- Body:** Specify the body of the mail
- Advanced mode checkbox:** Checked
- Advanced parameters:** Showing 1 of 7, with buttons for Show all and Clear all.
- Importance:** Normal

Bottom Screenshot (Create HTML table Example):

- To:** Enter part of a name or email address to find people
- Subject:** Specify the subject of the mail
- Body:** Contains the expression '=CreateHTMLTable(outputs('List group members'))'
- Advanced mode checkbox:** Unchecked
- Advanced parameters:** Showing 1 of 7, with buttons for Show all and Clear all.
- Importance:** Normal

Output pane (Left side of the interface):

- Search bar
- Join (highlighted in purple)
- Output
- List group members
- Display Name
- Given Name
- Job Title
- Mail
- Surname

Output pane (Right side of the interface):

- Search bar
- Create HTML table (highlighted in purple)
- Output
- List group members
- Display Name
- Given Name

Module 3

Taking Control with Flow Controls

Conditional logic

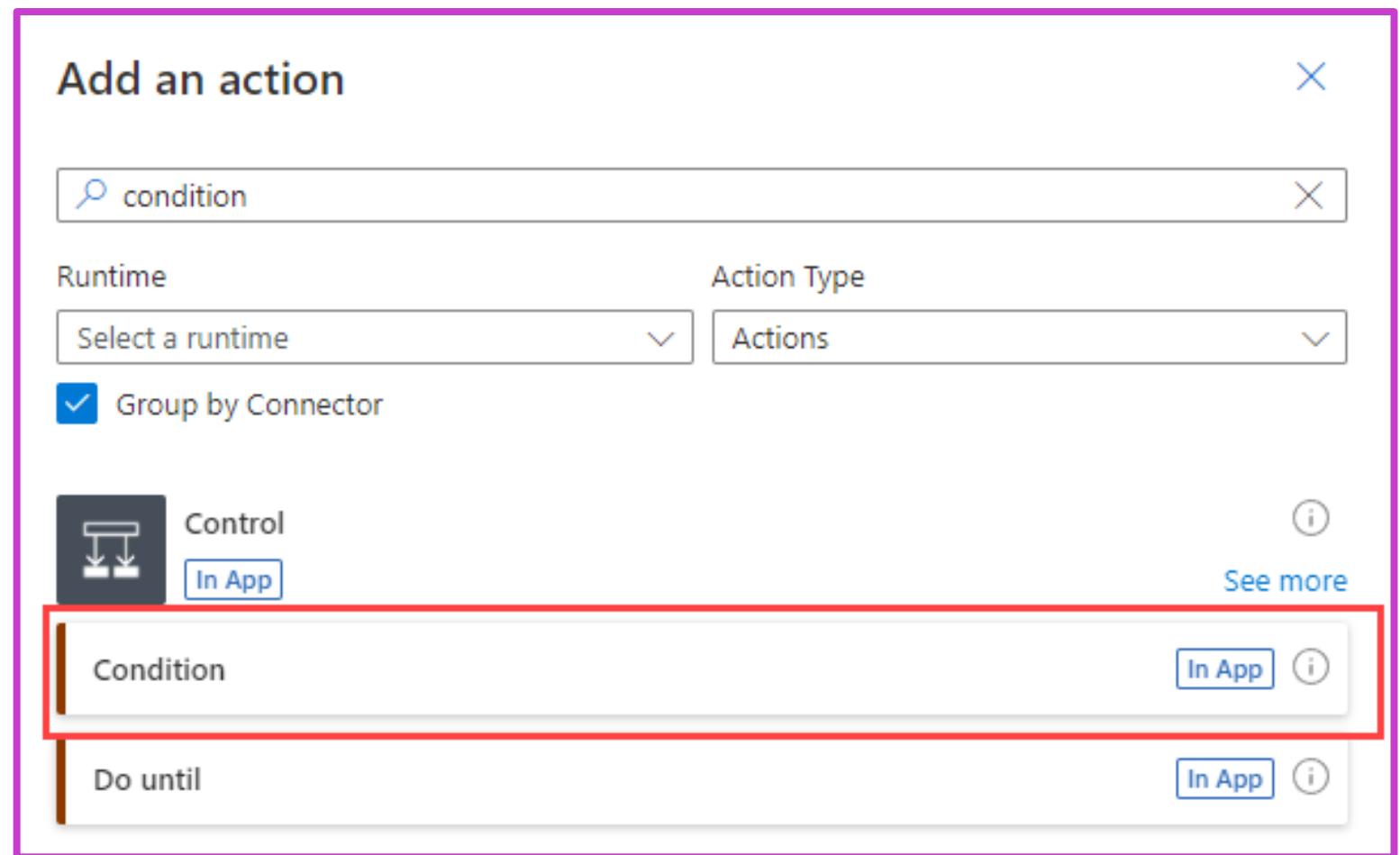
Conditions and switches offer ways to introduce conditional branching into a flow

Using conditions

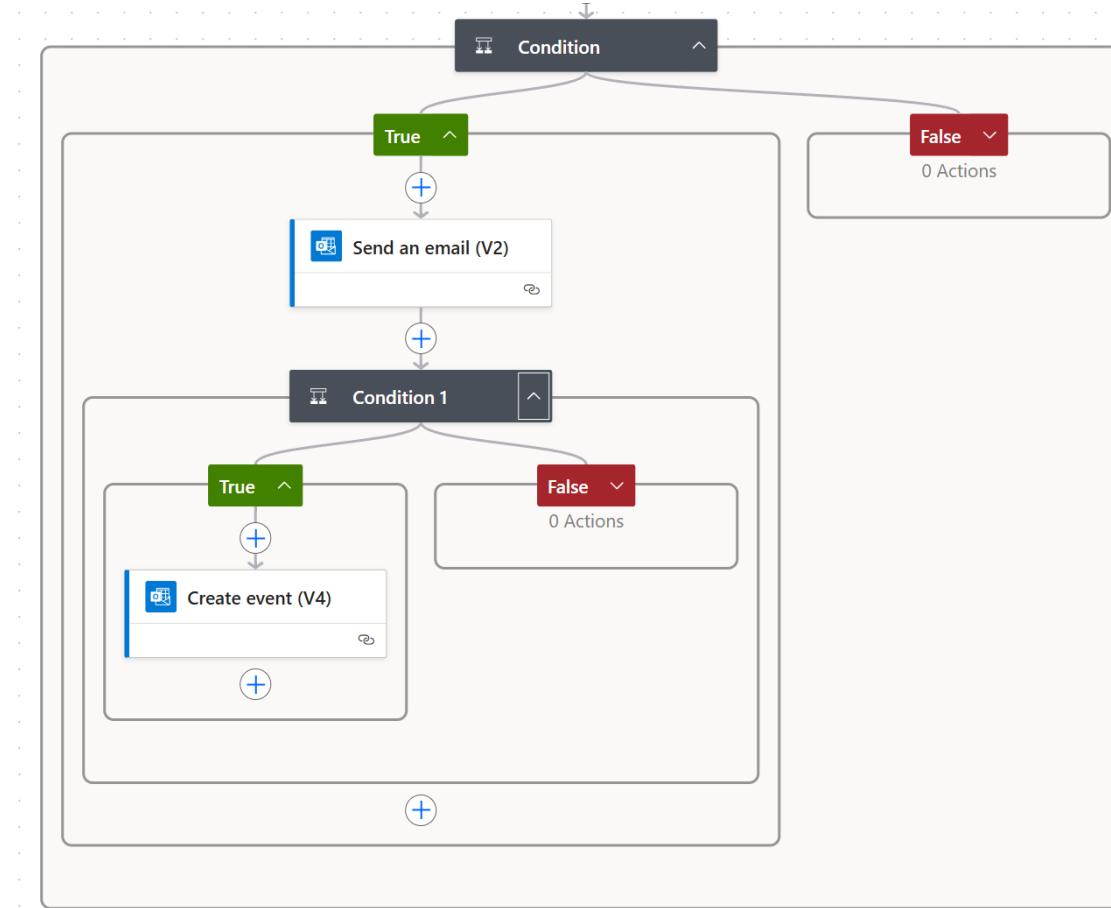
Search actions on “Condition” and select

Using switches

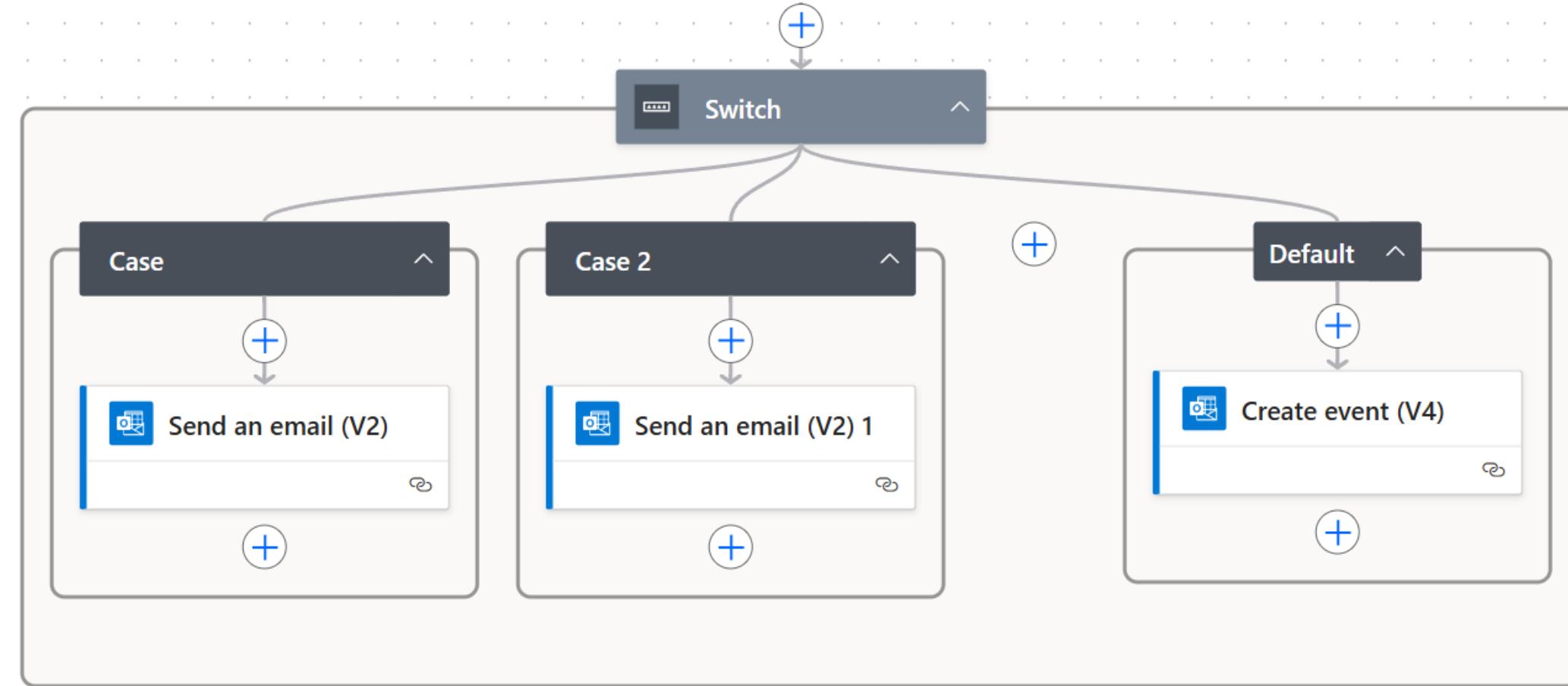
Searching on “Switch” and add to flow



Nest Condition



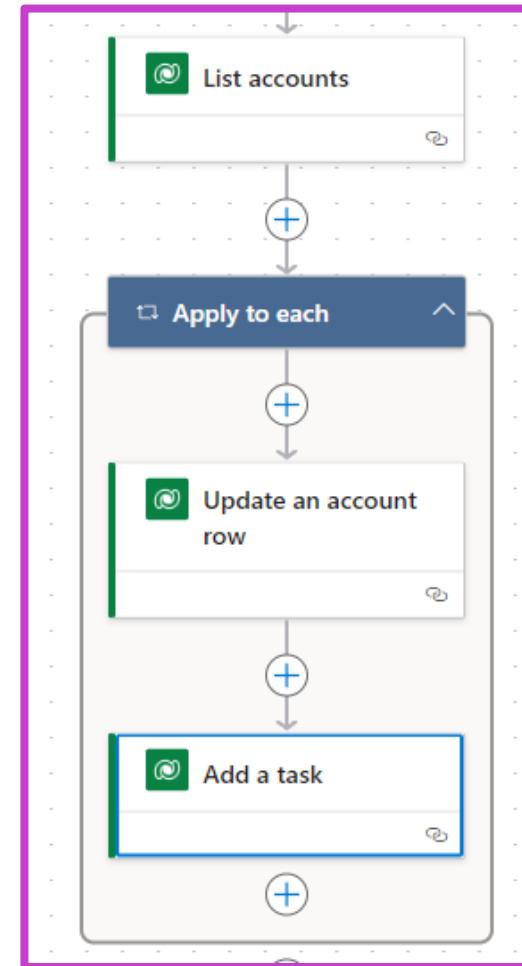
Switch



Loops

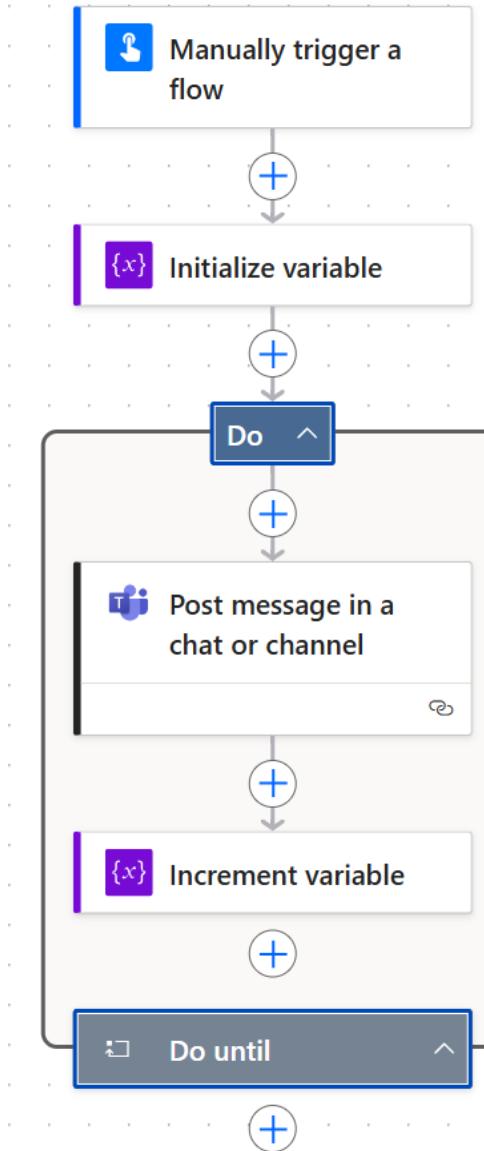
Apply for each

- Executes for each item in an array
- Often used after a query step



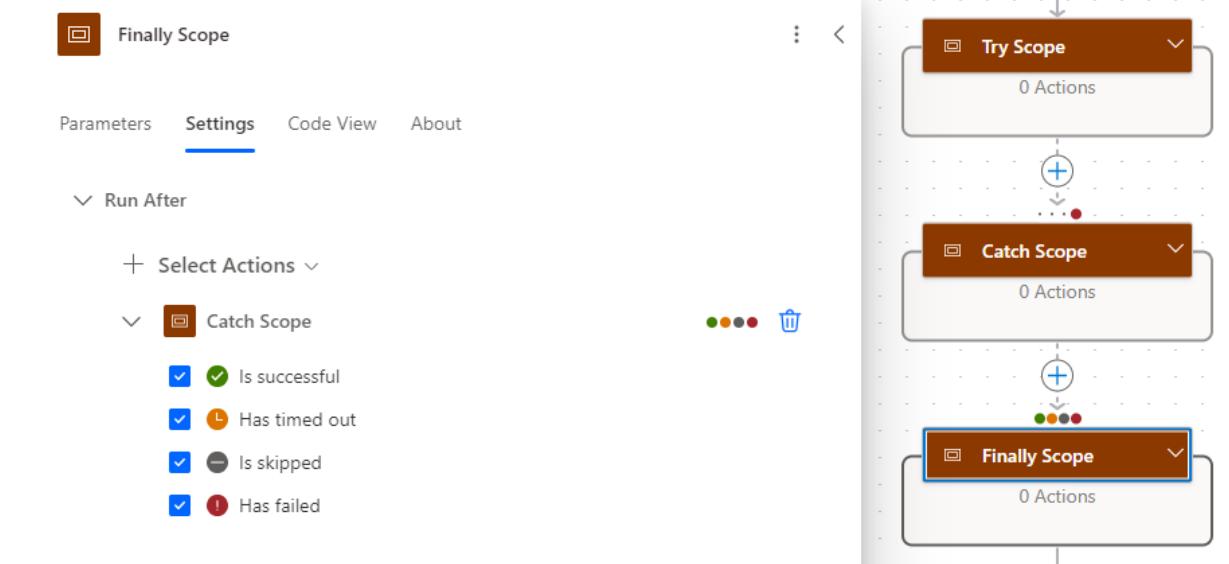
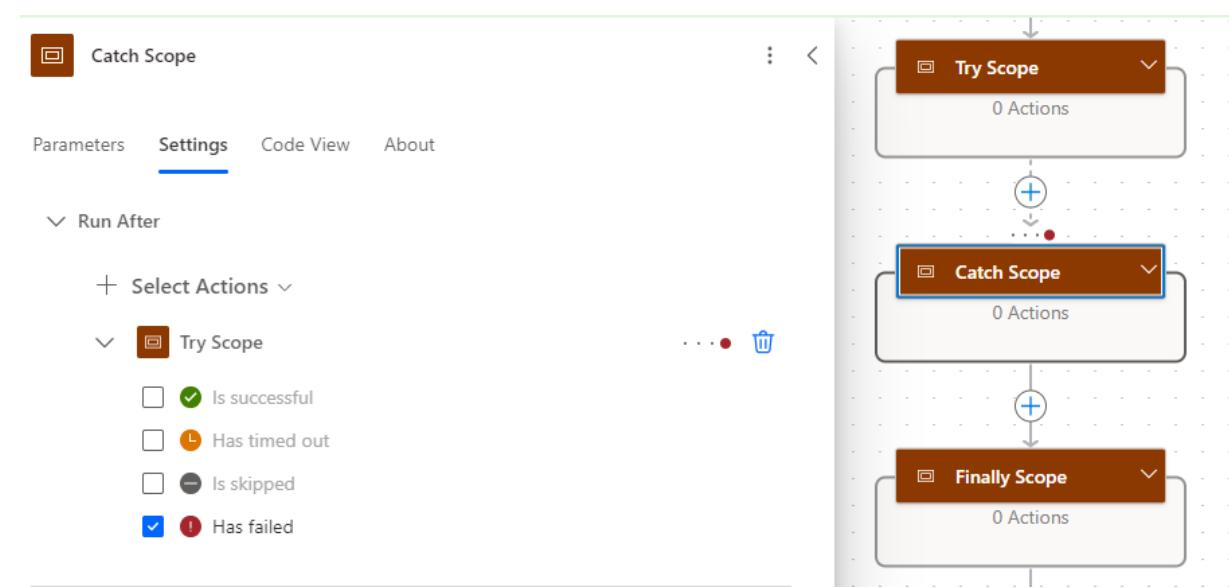
Loops Do-Until

Executes until a condition is met



Scope for Action Grouping

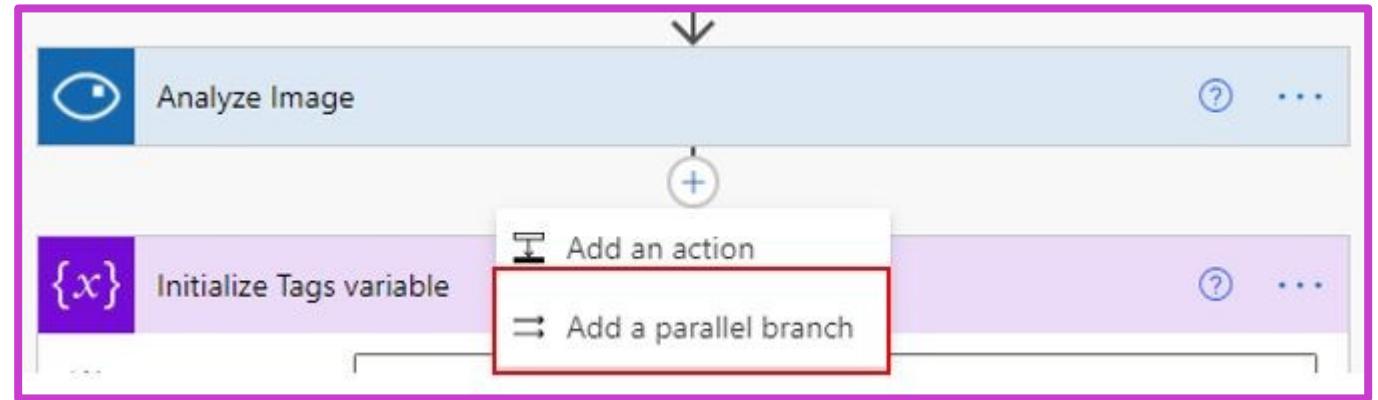
- Catch only runs when Try has failed
- When Try runs successfully, Catch scope is skipped
- Finally runs regardless of the Catch execution including when it is skipped



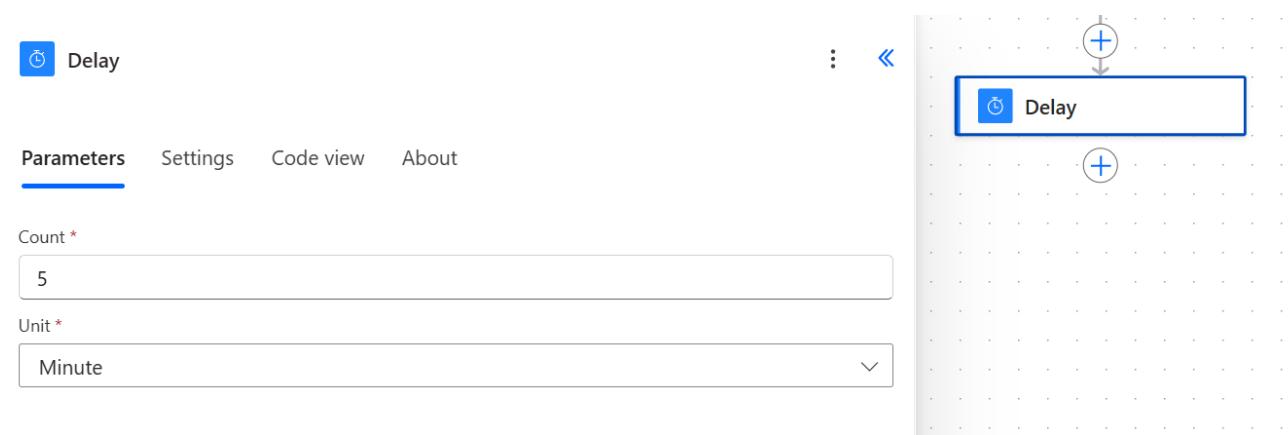
Branching

Parallel branches

- Error handling
- Parallel approvals



Delay / Delay Until / Timeout



Module 4

Advanced Approvals

Approvals actions

Actions

With the approvals capability in Power Automate, you can automate sign-off requests and combine human decision-making for workflows. Some popular cases where approvals can be used include:

- Approving vacation time requests
- Approving documents that need sign-off
- Approving expense reports



Approvals

Enables approvals in workflows.

Create an approval

Start and wait for an approval

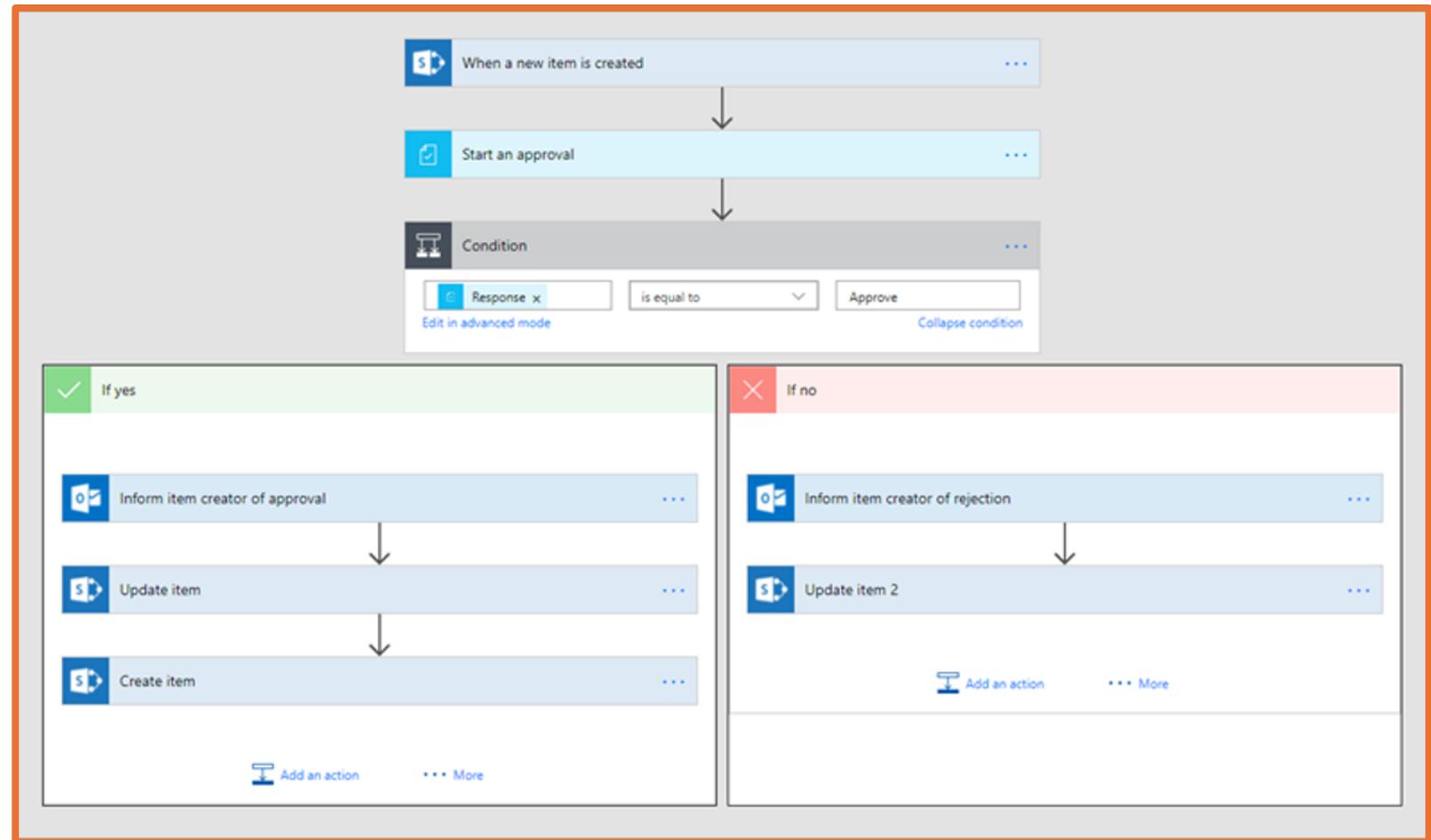
Start and wait for an approval of text

Wait for an approval

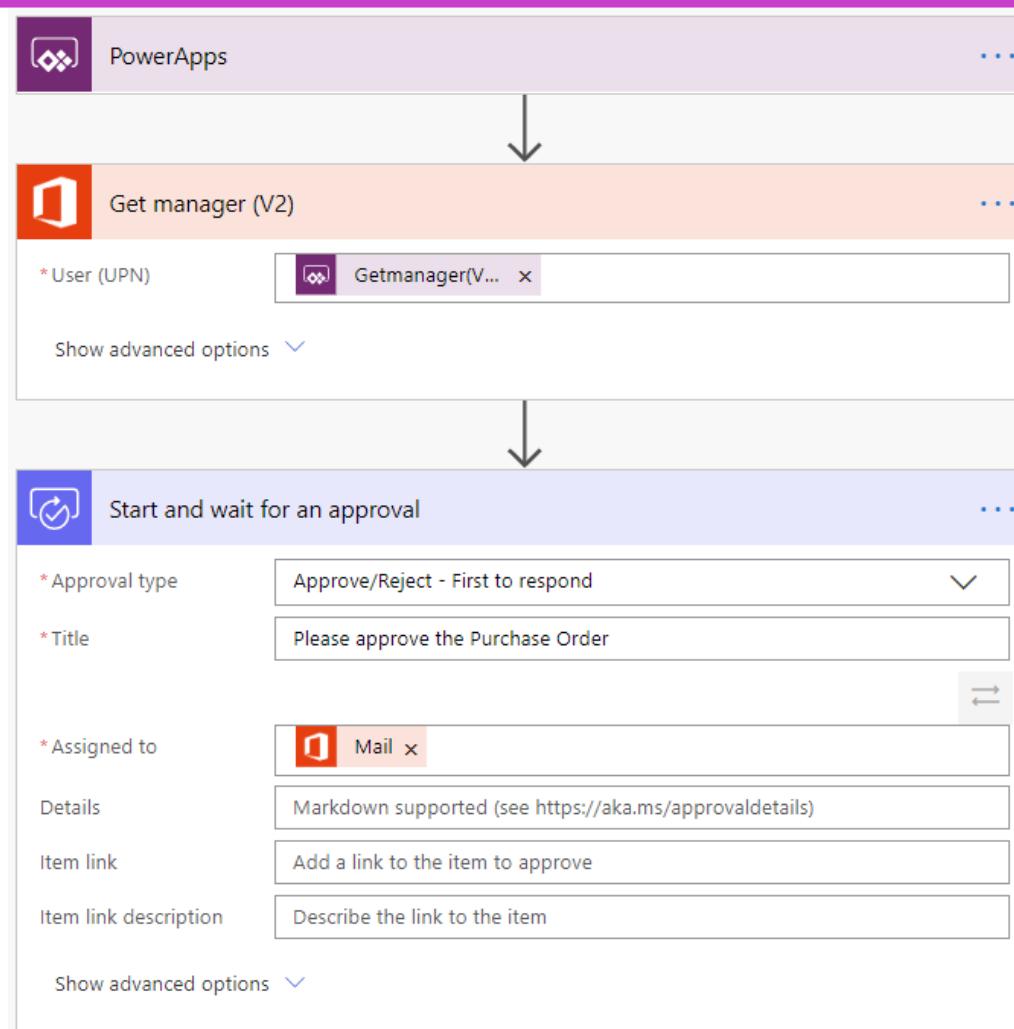
Approvals components

Components

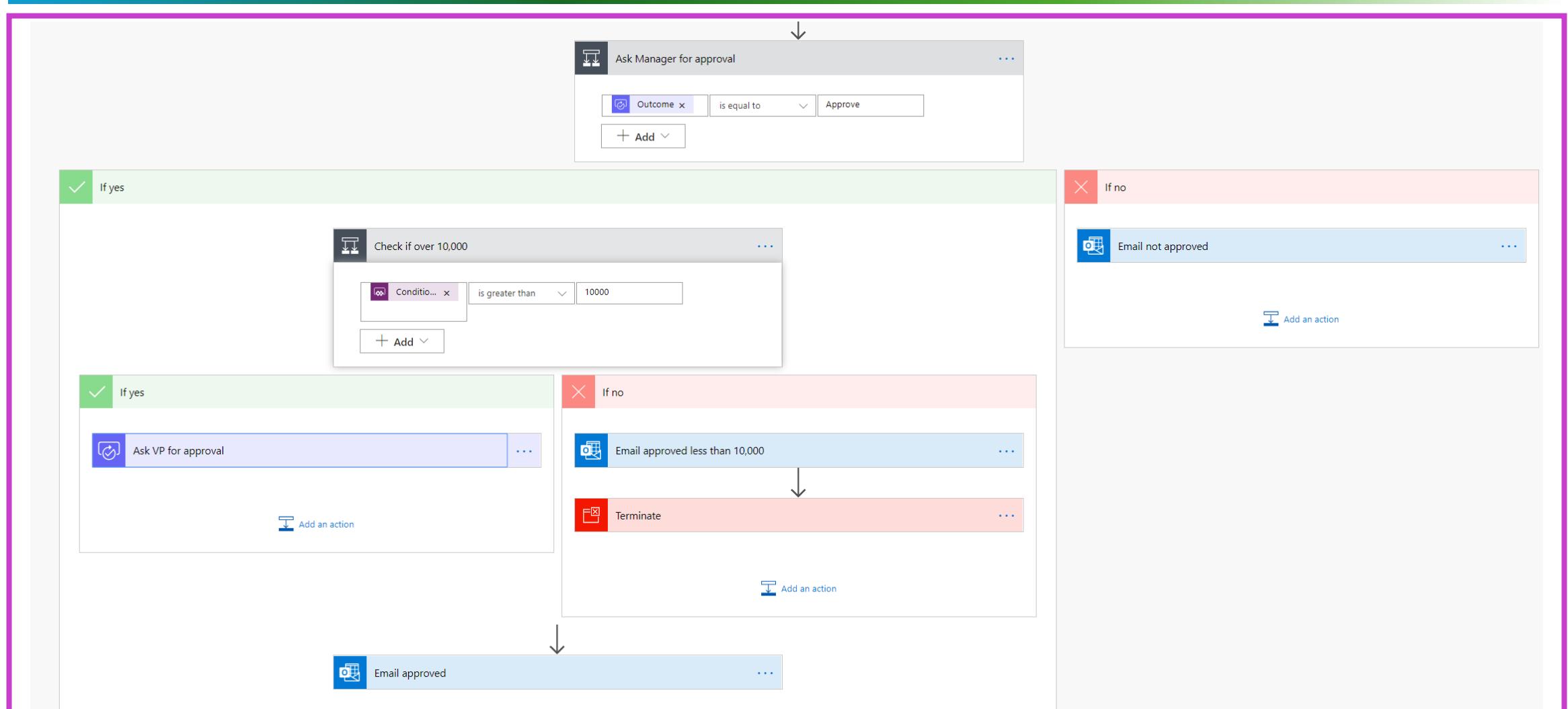
- Approval Action
- Approval Card
- Flow waits for Response
- Condition on Response



Example: Purchase order approval



Example: Purchase order approval



Approvals

Process approvals

- Power Automate portal
- Power Automate mobile
- Microsoft Teams

The screenshot shows the Microsoft Teams interface with the 'Approvals' app open. The left sidebar has a red box around the 'Approvals' icon. The main area shows a list of 'Received' approvals with 27 items. A red box highlights the 'Received' section header. The list includes various requests like 'Budget approval', 'New Social media post request', and 'A request for DocuSign', each with details such as status (Approved or Requested), creation date, requester, and recipient.

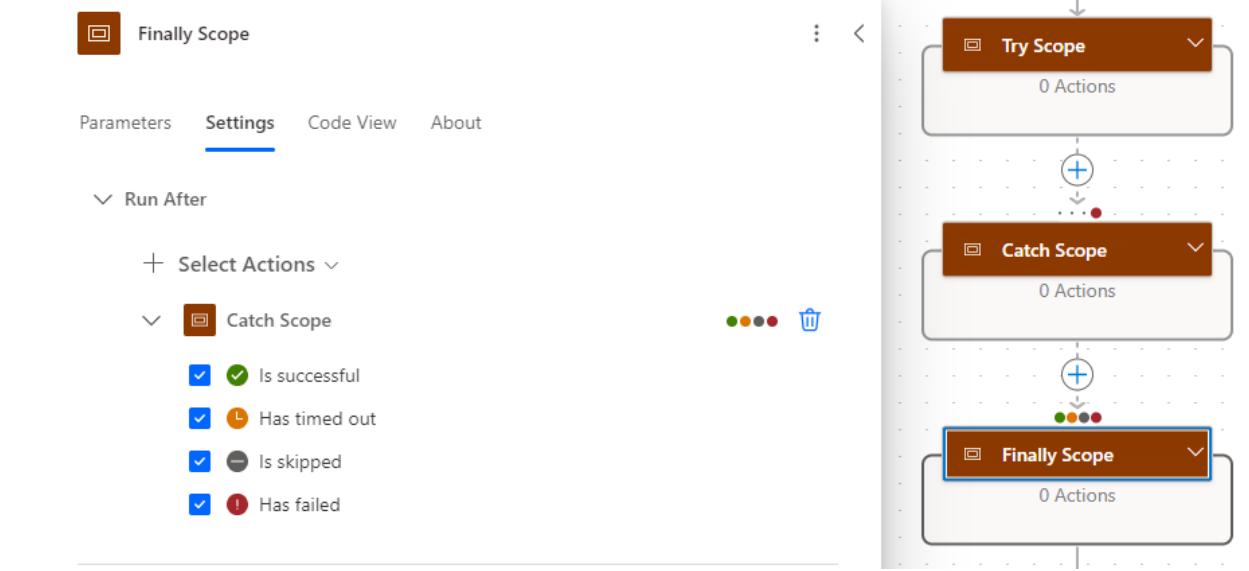
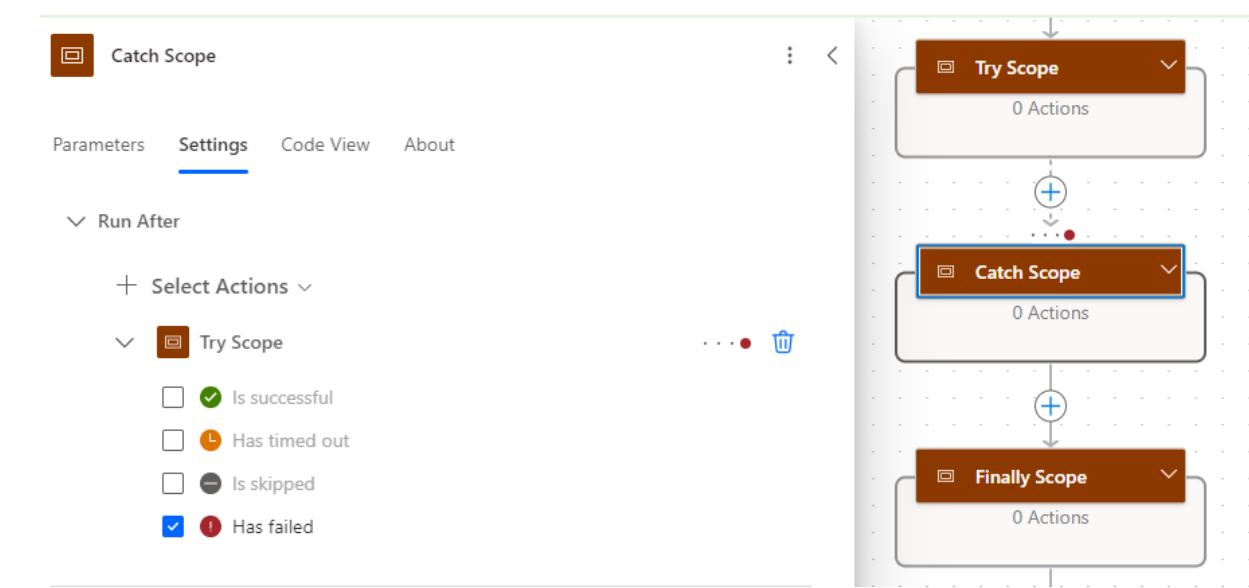
Request title	Status	Created	Requested by	Sent to
Budget approval	Approved	11/30/2020, 3:02:01 PM	Allan Deyoung	Alex Wilber
New Social media post request	Approved	11/30/2020, 11:44:22 AM	Allan Deyoung	Alex Wilber
A request for DocuSign	Approved	11/30/2020, 11:38:42 AM	Allan Deyoung	Alex Wilber
New Social media post request	Approved	11/30/2020, 10:30:49 AM	Allan Deyoung	Alex Wilber
New request for budget	Requested	11/30/2020, 7:29:07 AM	Allan Deyoung	Alex Wilber
New Social media post request	Approved	11/20/2020, 11:12:13 AM	Allan Deyoung	Alex Wilber
Hey approve this thing	Requested	11/20/2020, 11:07:29 AM	Allan Deyoung	Alex Wilber
Please approve: 'Luna'	Approved	11/17/2020, 4:36:16 PM	Allan Deyoung	Alex Wilber
New Social media post request	Approved	11/17/2020, 4:25:21 PM	Allan Deyoung	Alex Wilber
How would you rate it?	Amazing	11/17/2020, 4:14:33 PM	Allan Deyoung	Alex Wilber
Approvals v2	Awesome!	11/17/2020, 4:12:54 PM	Allan Deyoung	Alex Wilber
Demo with a file to boot	Approved	11/17/2020, 4:11:04 PM	Allan Deyoung	Alex Wilber

Module 5

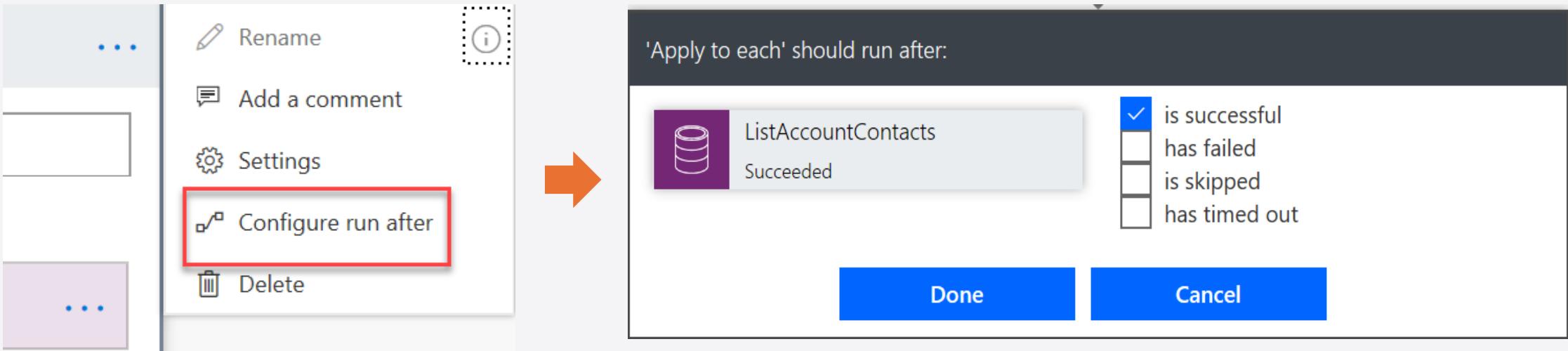
Best Practices and Error Handling

Try – Catch – Finally

- Catch only runs when Try has failed
- When Try runs successfully, Catch scope is skipped
- Finally runs regardless of the Catch execution including when it is skipped



Error handling



Configure run after

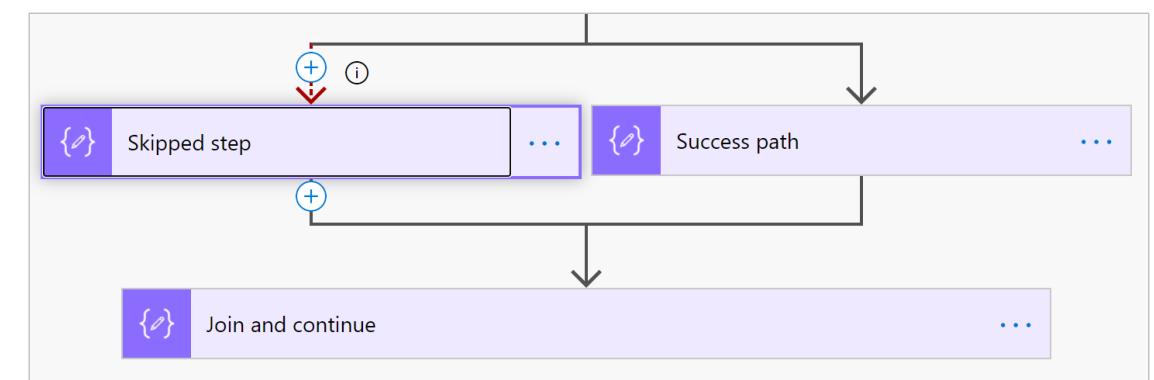
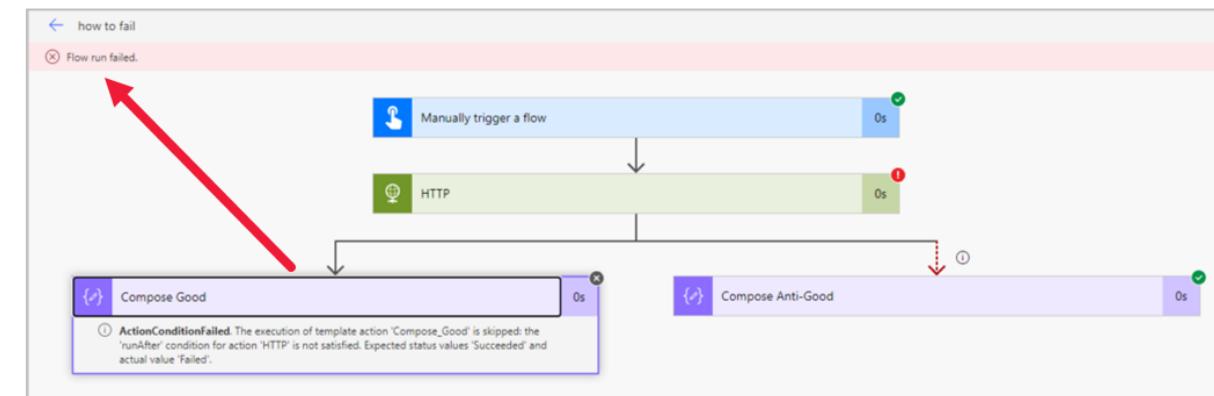
By default, flow will keep running through its step as long as prior actions have been successful

Errors

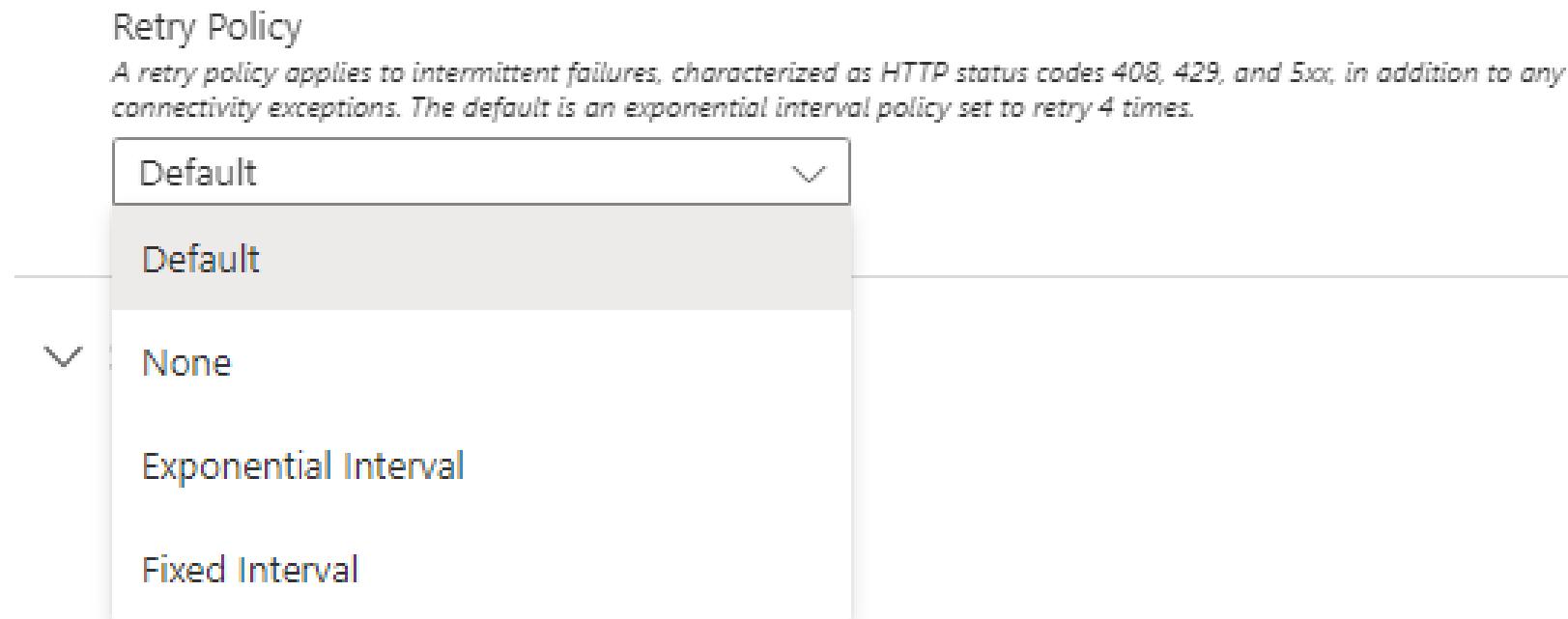
Failed
Skipped
Timed out

Errors and Flow Success

- For a flow run to be a success, all execution paths must complete
- That includes paths that contain skipped actions
- Solutions
 - Join the paths
 - Terminate the branch



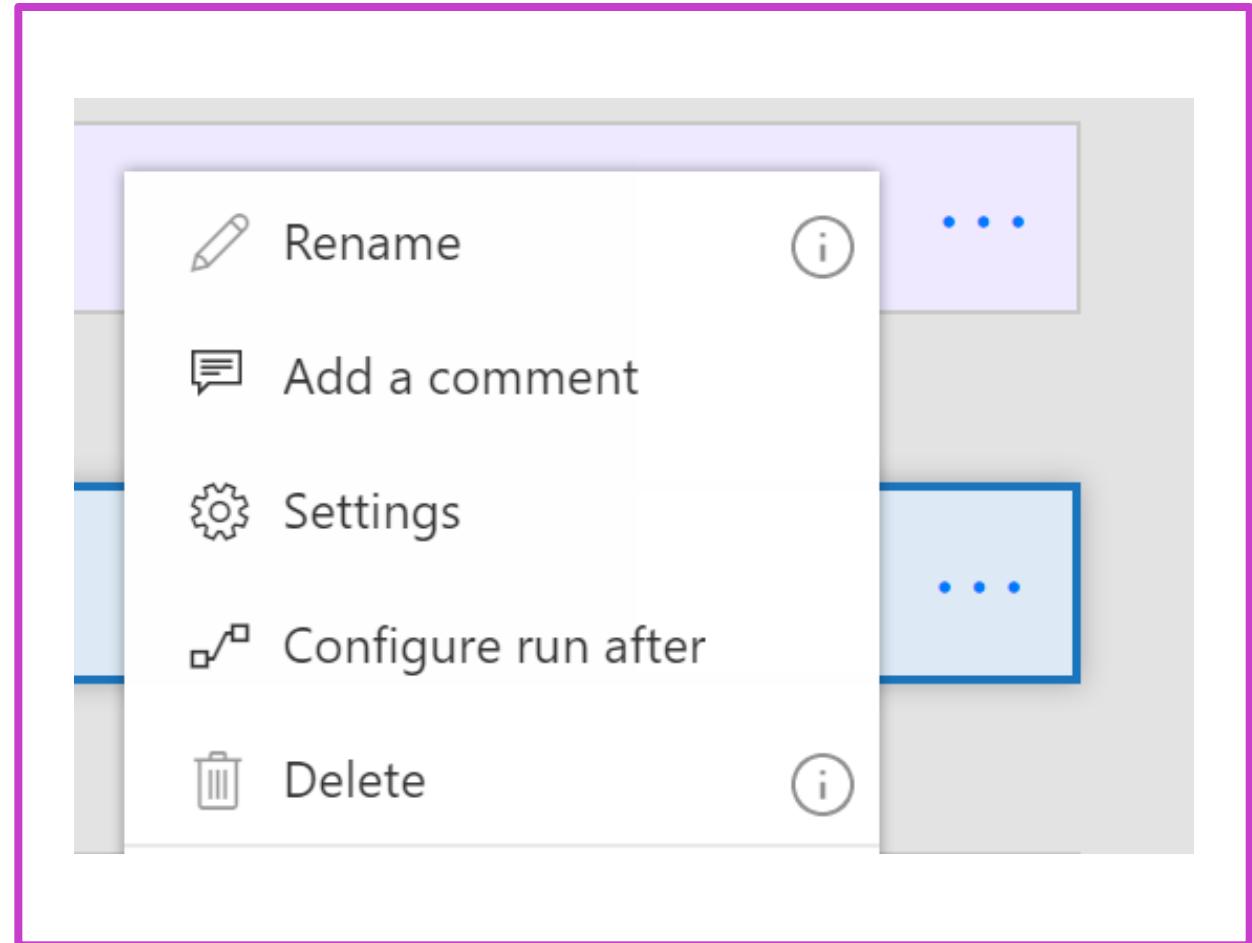
Changing Retry Policy



- Retry policy applies to intermittent failures, characterized as HTTP status codes 408, 429, and 5xx, in addition to any connectivity exceptions
- Adjust the retry policy to handle unique service scenarios

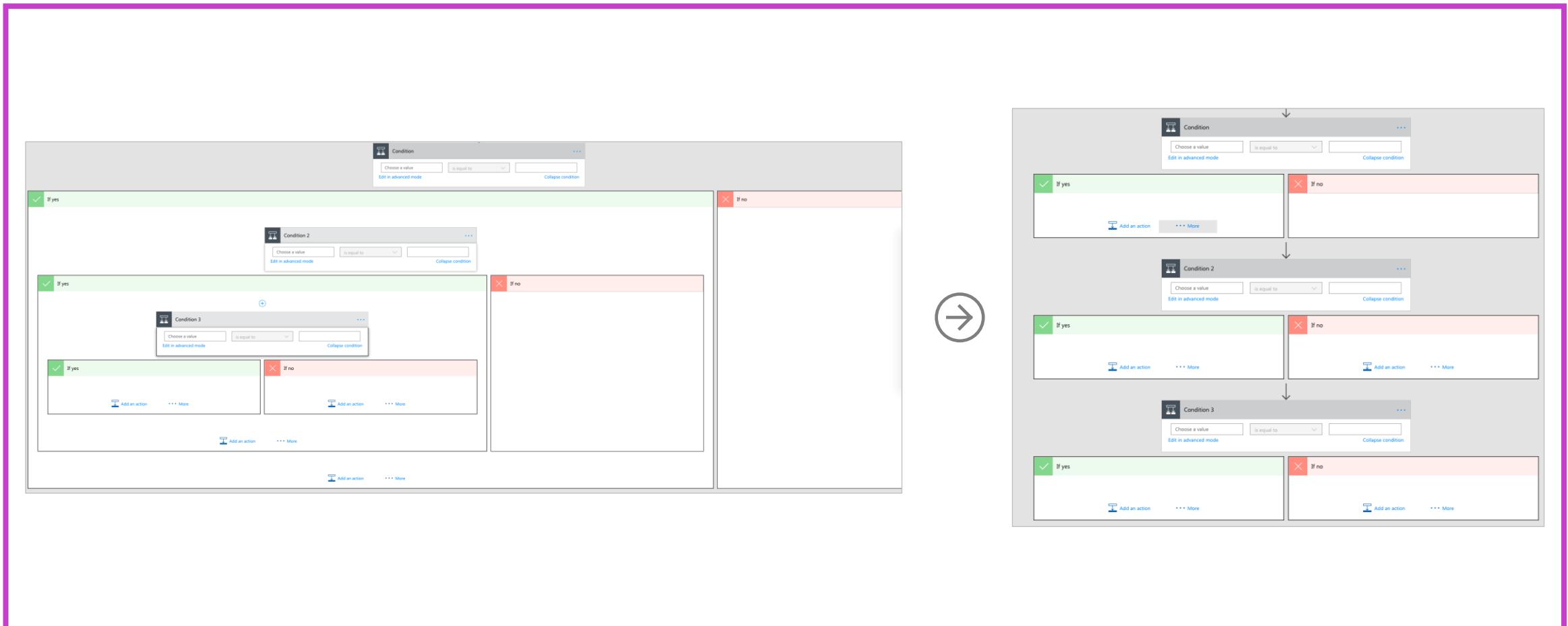
Action Settings

- The “...” menu contains entry points to Settings and Configure run after
- Settings lets you configure
 - Async actions
 - Timeout
 - Retry policy
 - Sequential
 - And more!

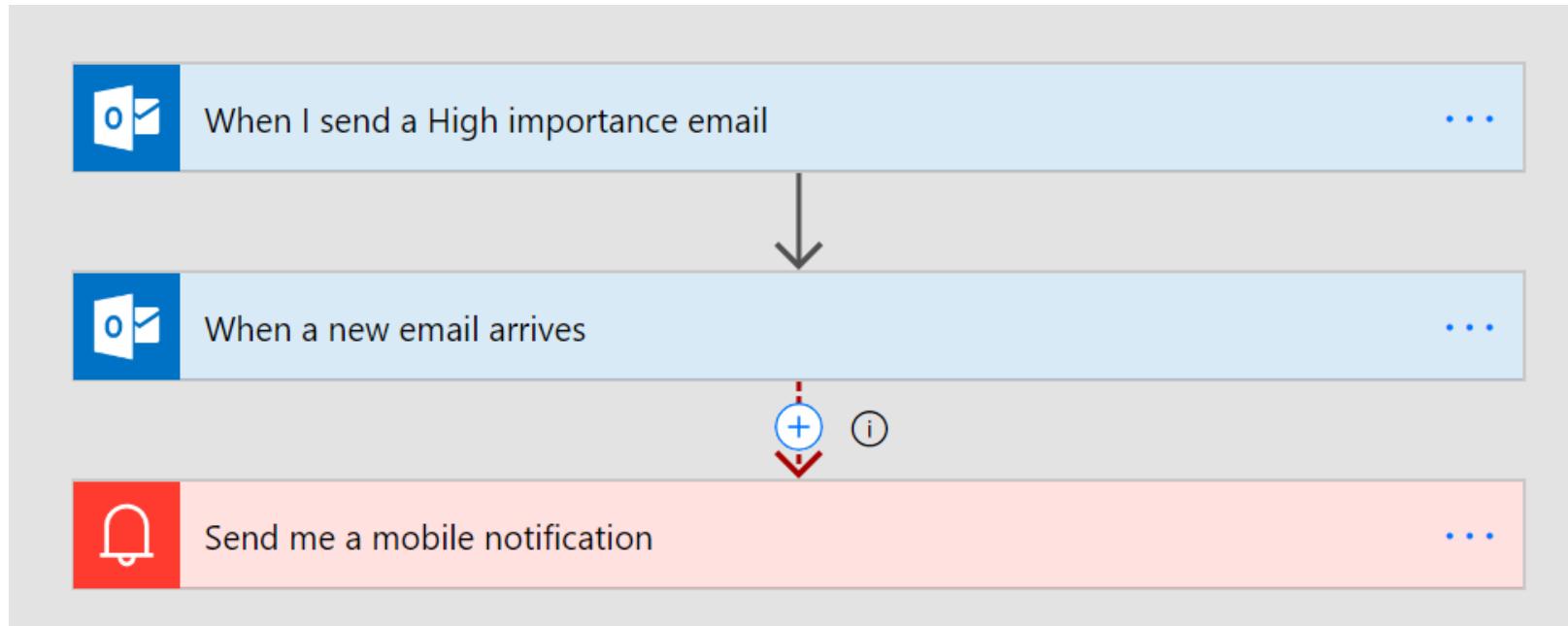


Terminate Flows

You can use this action to avoid nested conditions by ending in the middle



Asynchronous Actions

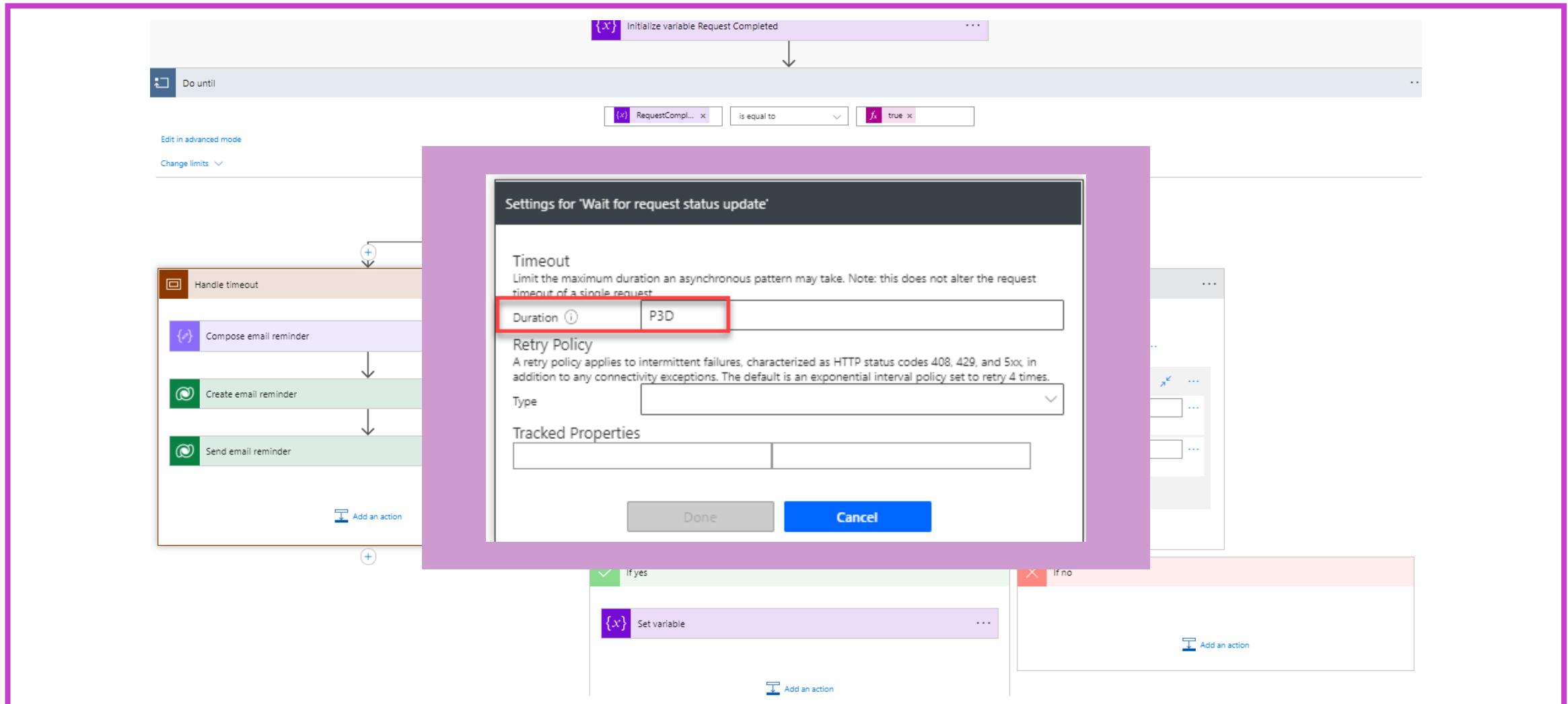


Any “trigger” can be in the middle of a flow as well

- For example, create a flow that waits for a reply to a certain thread in the middle

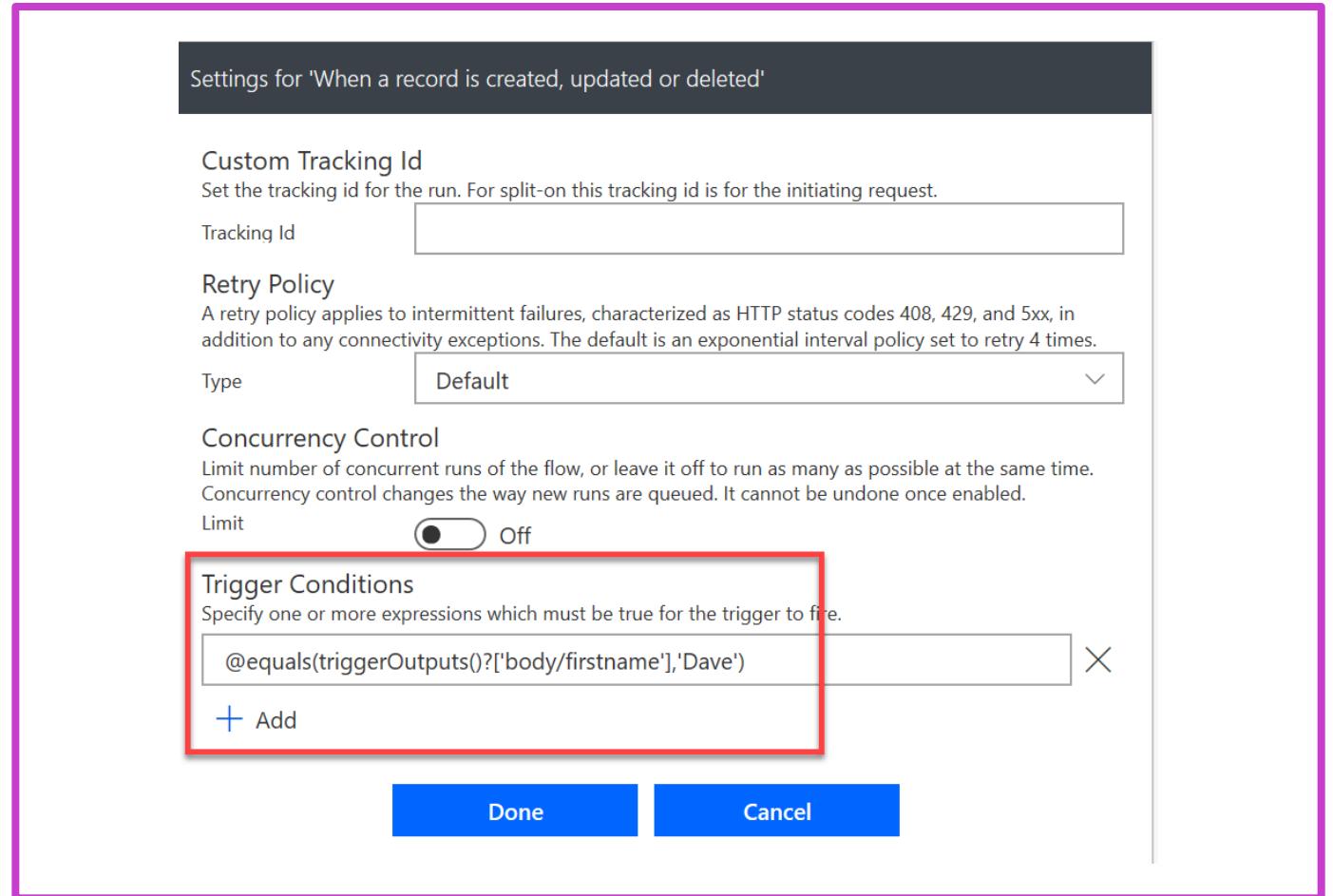
You probably want to configure a time out and handle that case

Wait Until Pattern



Using Settings → Trigger Conditions

- Use flow expressions for filter
- Works on create and update
- Not available if trigger is used later in flow

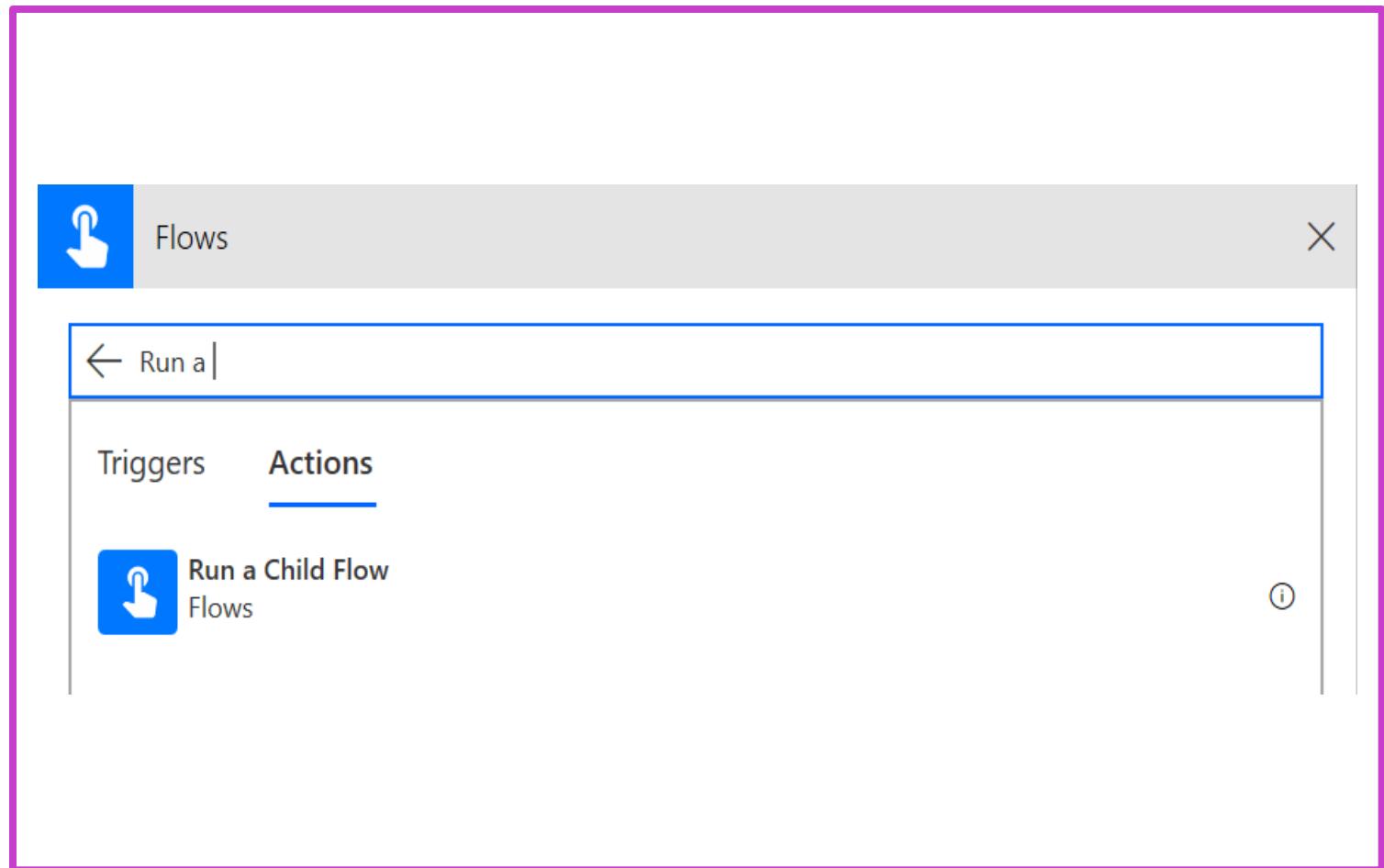


Module 6

Parent and Child Flows

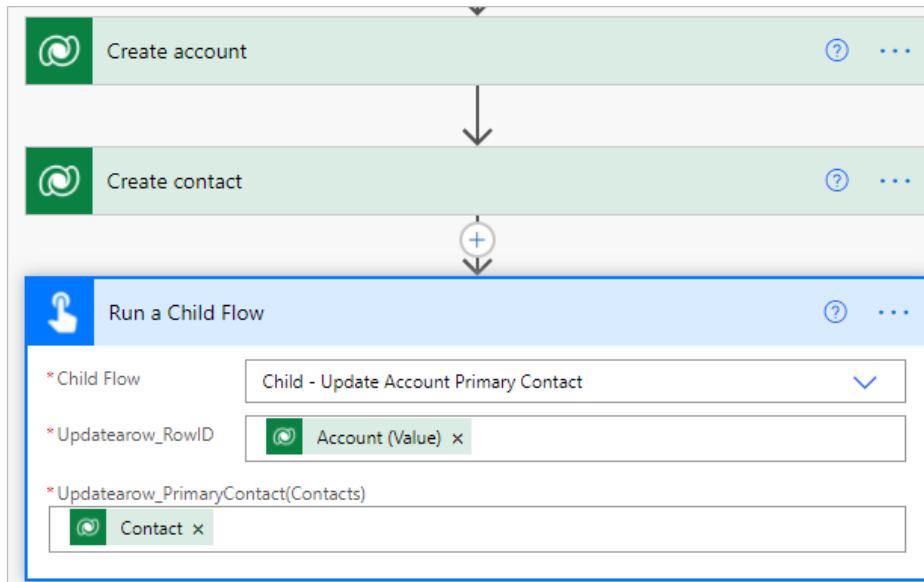
About Child Flows

- Allows breaking out parts of a flow into a reusable child flow
- Trigger support of child flows
 - Manually triggered button flow
 - Power Apps
 - HTTP request
- Respond with Power Apps or HTTP response
 - If no response action, parent won't wait
- Solutions are required for this to work

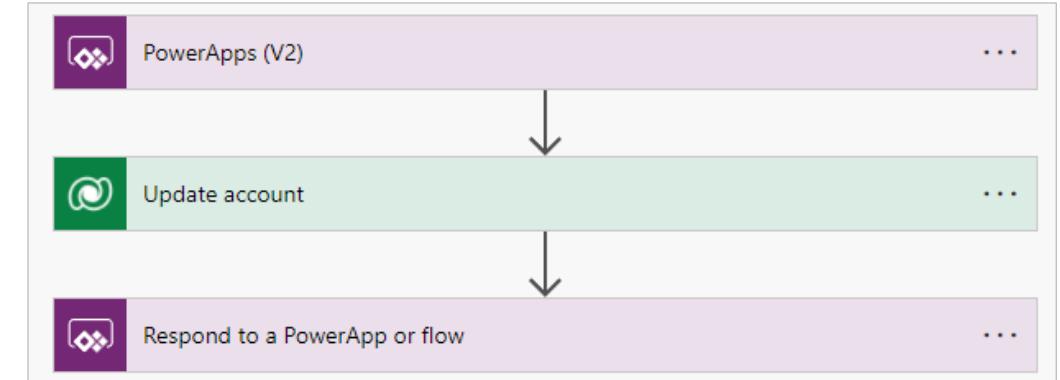


Example – Using a Child Flow

Parent flow



Child flow



Module 7

Governance, Security & Monitoring

Flow Naming Conventions

- **Standardizing Flow Names**
- **Format:** [Environment]-[Module]-[Function]-[Version]
- **Examples:**
 - PROD-Finance-PurchaseOrderApproval-v1
 - DEV-HR-LeaveRequestProcessing-v2
 - TEST-IT-UserProvisioning-v1
- **Best Practices:**
 - Use names that clearly describe the function
 - Avoid names that are too short or too long
 - Include a Version Number

Action Naming

- **Clearly Naming Actions**
- **Format:** [Action Type] - [Description]
- **Examples:**
 - HTTP Request - Get User Data
 - Parse JSON - Parse Order Response
 - Condition - Check Approval Status

- **Best Practices:**
 - Name the action to describe its operation.
 - Avoid default names like "Condition 2"

Variable Naming

- **Standardizing Variable Names**
- **Format:** [Prefix][Description]
- **Prefixes:**
 - str: String (e.g., strCustomerName)
 - int: Integer (e.g., intOrderCount)
 - bool: Boolean (e.g., boolIsApproved)
 - arr: Array (e.g., arrOrderItems)
 - obj: Object (e.g., objOrderData)

Environments

- **Environment Types:**
 - **Development:** For development and initial testing.
 - **Test/UAT:** For User Acceptance Testing.
 - **Production:** For live, operational usage.
- **Best Practices:**
 - Use separate Environments aligned with the SDLC (Software Development Life Cycle).
 - Limit Access based on Role.
 - Utilize Environment Variables for Configuration.

Solutions

- **Managing and Deploying Flows**
- **Purpose:** To manage and deploy Flows between Environments.
- **Solution Structure:**
 - **Components:** Flows, Connections, Variables.
 - **Dependencies:** External Dependencies.
 - **Versioning:** Version Management.
- **Deployment Process:**
 - Create Solution in Source Environment.
 - Add Components.
 - Export Solution.
 - Import to Target Environment.
 - Configure Connections and Variables.

Environment Variables

- **Storing Configuration Values**
- **Use Cases:**
 - API Endpoints
 - Connection Strings
 - Feature Flags
 - Threshold Values
- **Benefits:**
 - Environment-Specific Configuration.
 - Easy Updates.
 - Security (Sensitive Data).

Security Roles

- **Defining Security Roles**
- **Security Roles:**
 - **Flow Owner:** Flow owner with modification rights.
 - **Flow User:** User who can run the Flow.
 - **Environment Admin:** Environment maintainer/administrator.
 - **Solution Admin:** Solution maintainer/administrator.

Access Control

- **Setting Access Permissions**

- **Share Flow:** Share Flow with Users or Groups.
- **Permissions:**
 - **Can Edit:** Can modify the Flow.
 - **Can View:** Can view the Flow.
 - **Can Run:** Can execute the Flow.

- **Best Practices:**

- Apply the **Principle of Least Privilege**.
- Use **Groups** instead of Individual Users.
- Review Permissions regularly.

Optimization: Reduce Nested Loops

- Before

Apply to Each: items

 Apply to Each: subItems

 Apply to Each: details

 Process Detail

- After

Filter Array: items where status = "Active"

Select: Transform to required format

Apply to Each: filteredItems

 Process Item

Optimization: Reduce Nested Loops

- Before

Apply to Each: allItems

Condition: If item.status = "Active"

Process Item

- After

Filter Array: allItems where status = "Active"

Apply to Each: activeItems

Process Item

Before we finish
Today's class



Take Class Evaluation : [Click to Evaluation](#)

Scan now :





THANK YOU



The Offices at CentralWorld, 16 Floor
Pathumwan, Bangkok, 10330, Thailand



(+66) 2 613 1033



thailand@trainocate.com



www.trainocate.com