

WELCOME TO

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



5 Nov 2025



9.00AM – 4.00PM

Instructor: K. Phakkhaphong Krittawat

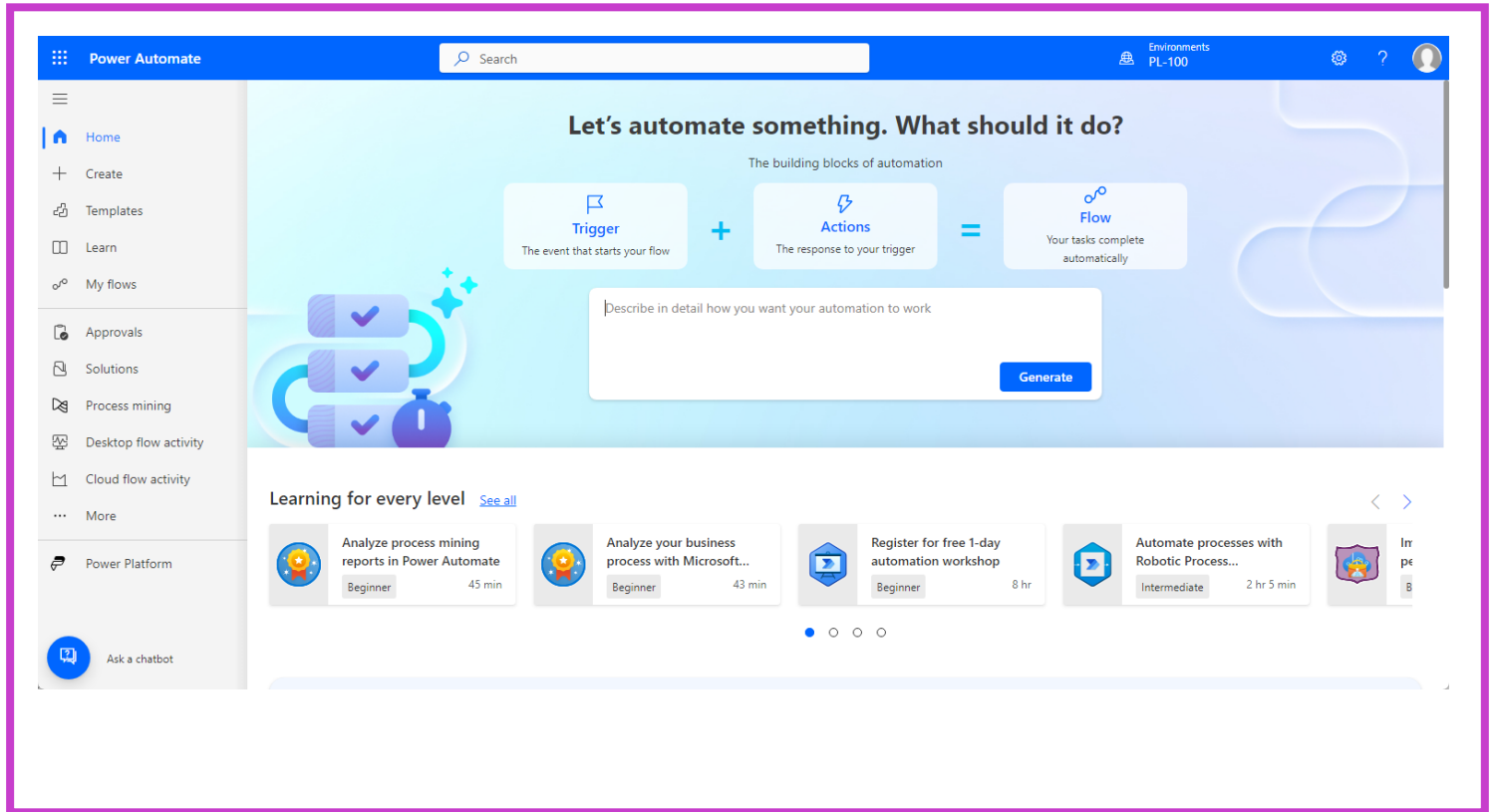


Module 1

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

Key concepts

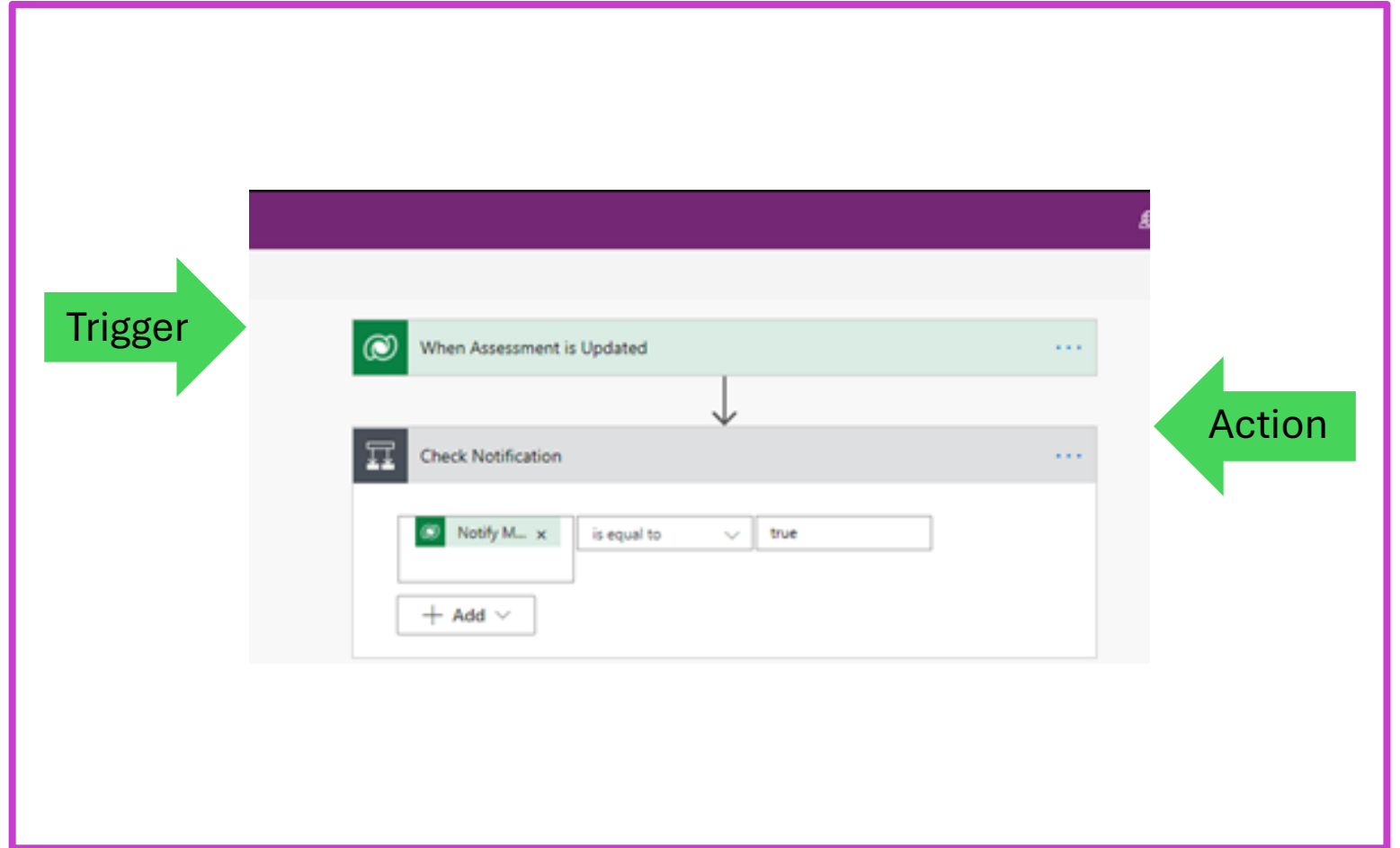
Every cloud flow has two main parts:

A Trigger

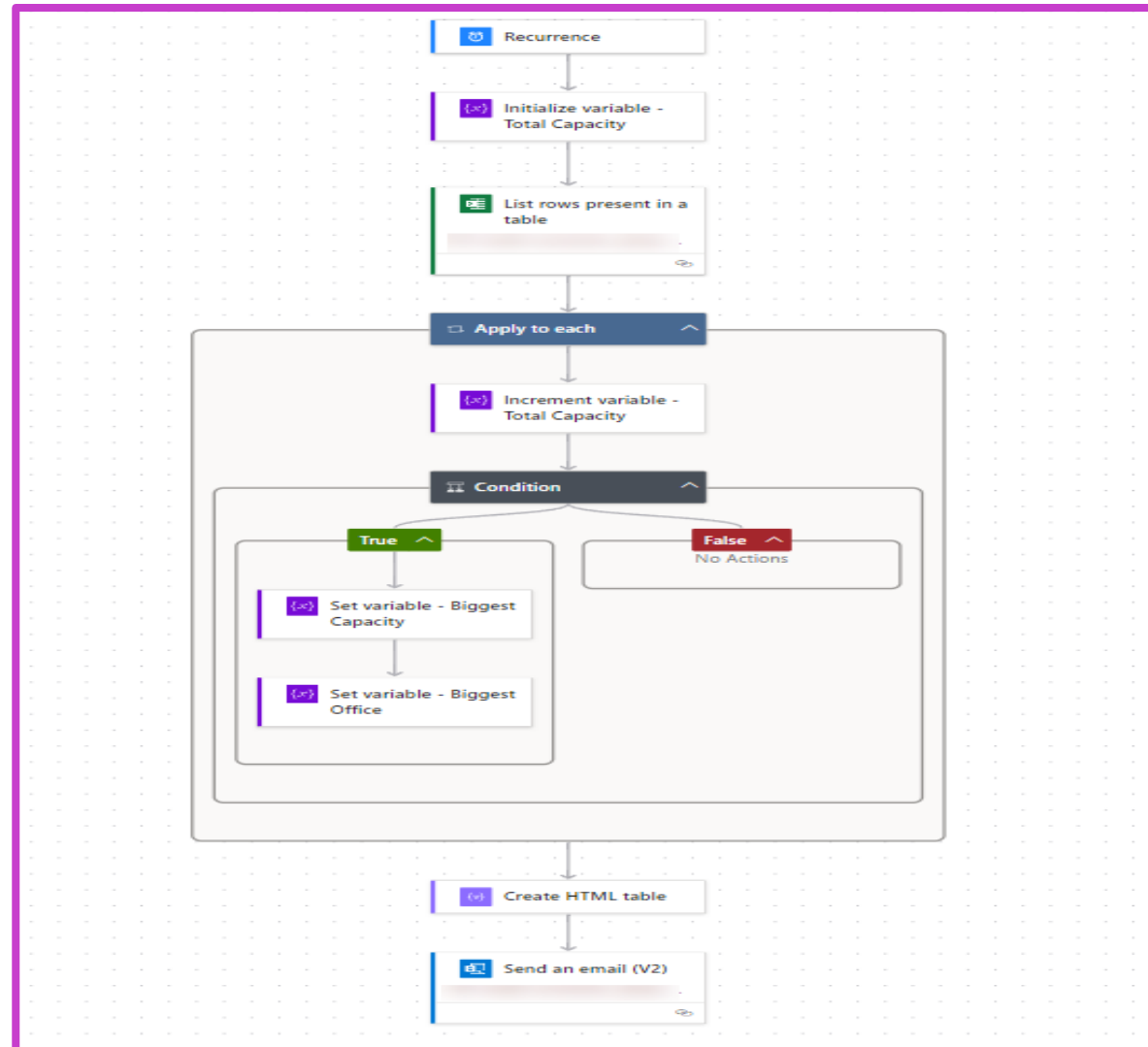
The starting action of the flow such as new email arriving in your inbox or a new item being added to a SharePoint list

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 ⓘ

Search all templates

Top picks


Remote work

Email

Notifications


Save to cloud

Approval




Follow up on a message
By Microsoft

Instant25322




Create a task in Planner from a message
By Microsoft

Instant24800



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

Automated19616



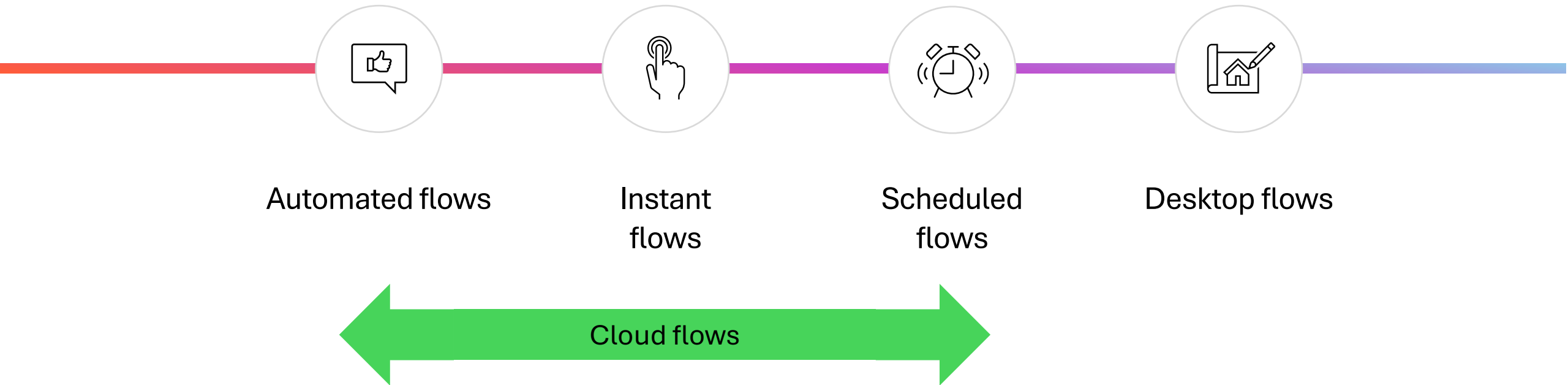
Start an approval in Teams when a file is added to a SharePoint folder
By Microsoft

Automated19200

All templates →

Power Automate

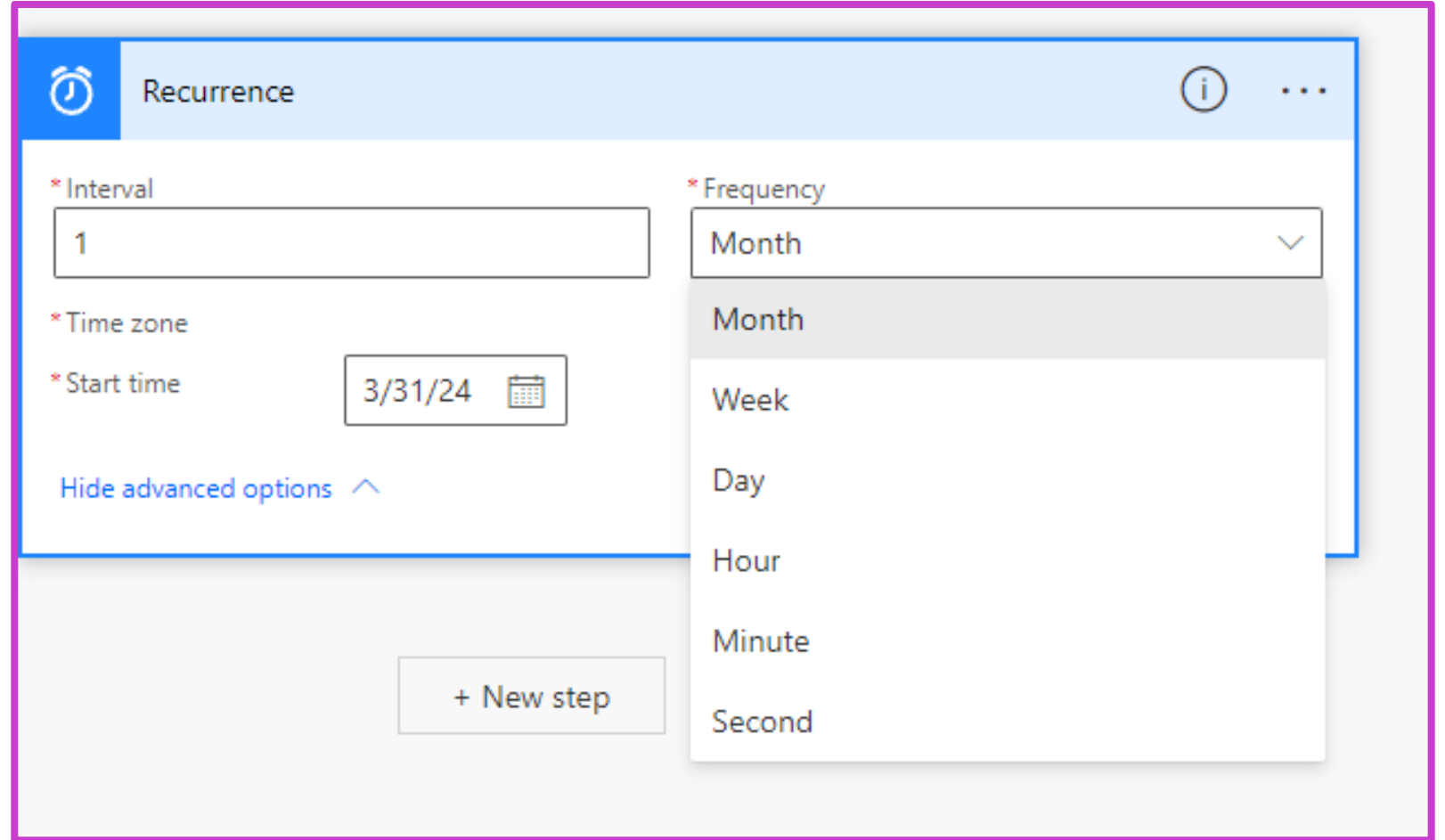
Types of Power Automate flow



Create a scheduled flow

Recurrence trigger

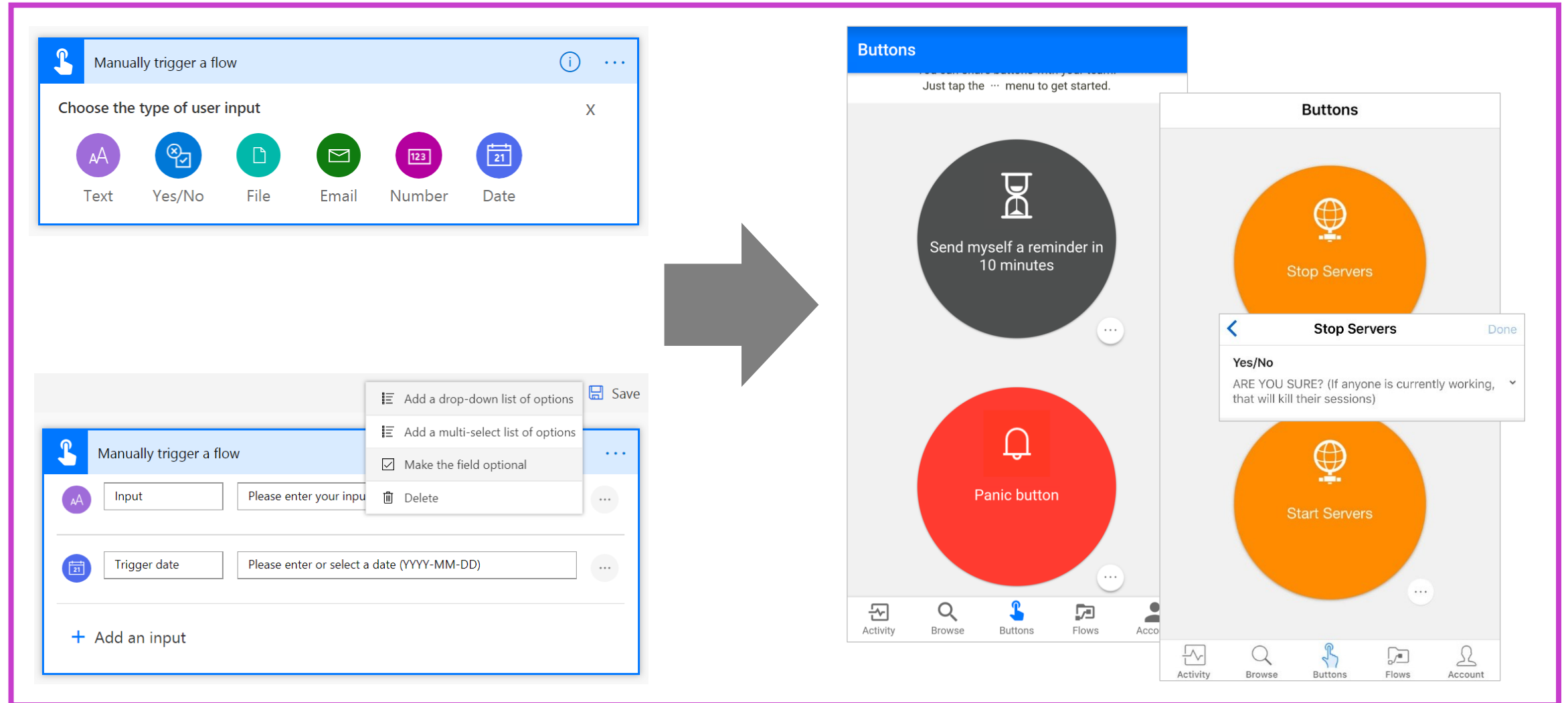
- Runs on a pre-defined schedule
- Can also run manually from Power Automate portal



The screenshot shows the 'Recurrence' configuration window in Power Automate. The window has a blue header with a clock icon and the title 'Recurrence'. Below the header, there are several fields for configuring the recurrence:

- * Interval:** A text box containing the number '1'.
- * Frequency:** A dropdown menu currently showing 'Month'. The dropdown is open, showing a list of options: 'Month', 'Week', 'Day', 'Hour', 'Minute', and 'Second'. 'Month' is highlighted.
- * Time zone:** A text box (partially visible).
- * Start time:** A text box containing '3/31/24' with a calendar icon to its right.
- Hide advanced options:** A link with an upward arrow.
- + New step:** A button at the bottom right.

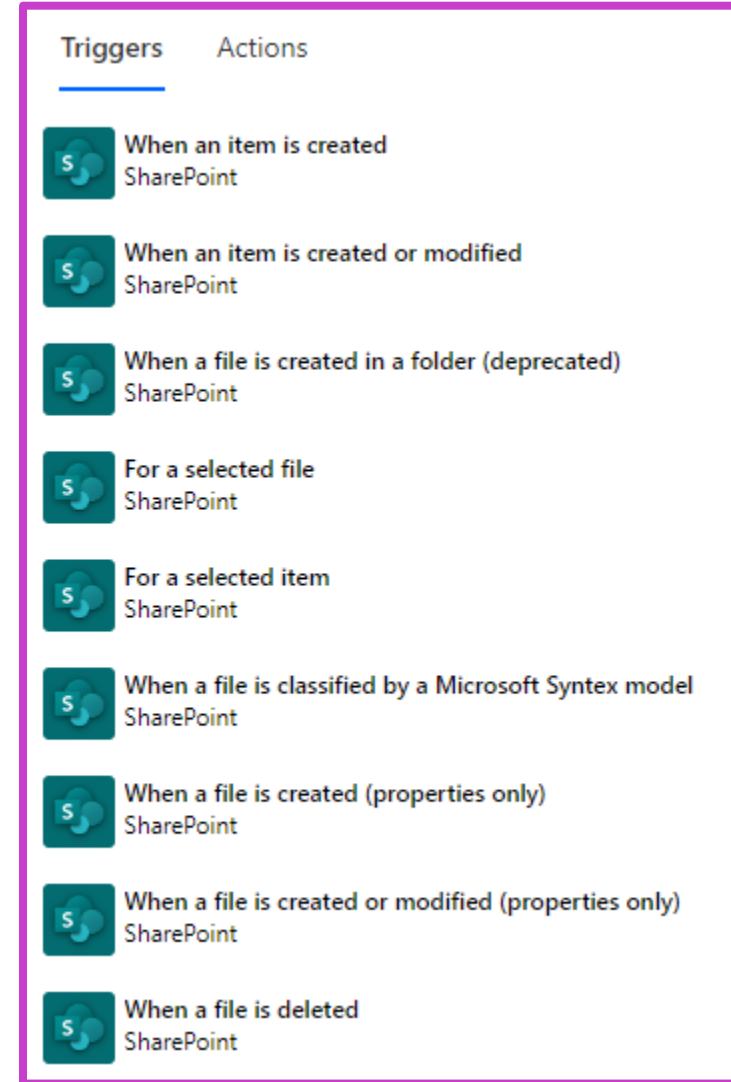
Create an instant flow



Create an automated flow

Event-based trigger

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



Naming

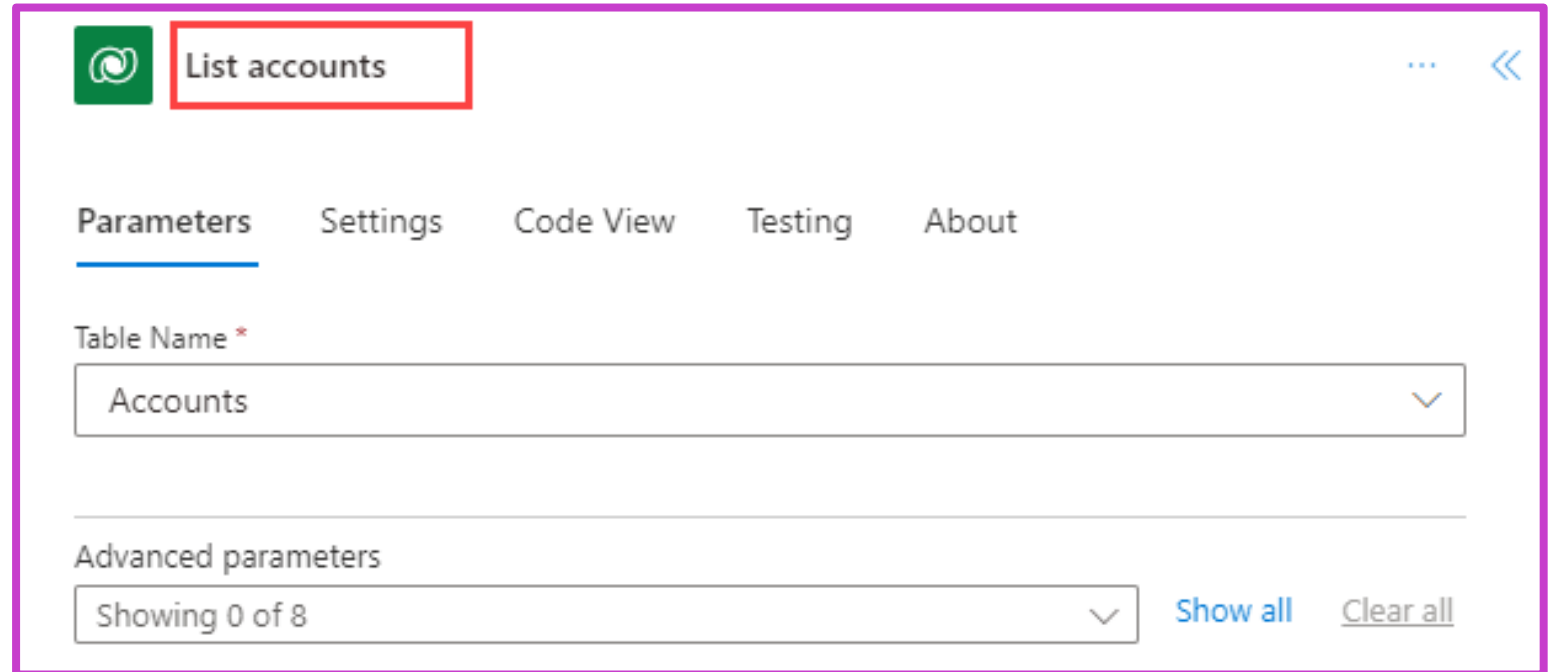
Naming flows

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



The screenshot shows a web interface for configuring a flow named "List accounts". The title bar includes a green icon with a white swirl and the text "List accounts", which is highlighted by a red rectangle. To the right of the title bar are three dots and a double-left arrow. Below the title bar is a tabbed interface with five tabs: "Parameters" (selected with a blue underline), "Settings", "Code View", "Testing", and "About". Under the "Parameters" tab, there is a label "Table Name *" followed by a text input field containing the word "Accounts" and a dropdown arrow. Below this is a section titled "Advanced parameters" with a text input field containing "Showing 0 of 8" and a dropdown arrow. To the right of this field are two links: "Show all" in blue and "Clear all" in grey.

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

The screenshot displays the 'Update a row' interface in the Trainocate application. The interface is divided into two main sections: a left sidebar for configuration and a right panel for data entry.

Left Sidebar (Configuration):

- Parameters:** The active tab, showing fields for 'Table Name' (set to 'Accounts') and 'Row ID' (with a red error message: 'Row ID' is required.).
- Settings:** A tab for configuring the step.
- Code View:** A tab for viewing the underlying code.
- Testing:** A tab for testing the step.
- About:** A tab for information about the step.

Right Panel (Data Entry):

- Search:** A search bar at the top.
- List accounts:** A section with a green header and a 'See More' link.
- Account Name:** A text input field with the placeholder 'Type the company or business name.'.
- Address 1: City:** A text input field with the placeholder 'Type the city for the primary address.'.
- Address 1: Street 1:** A text input field with the placeholder 'Type the first line of the primary address.'.
- Address 1: Street 2:** A text input field with the placeholder 'Type the second line of the primary address.'.
- Address 1: ZIP/Postal Code:** A text input field with the placeholder 'Type the ZIP Code or postal code for the primary address.'.
- Annual Revenue:** A text input field with the placeholder 'Type the annual revenue for the account, used as an ind...'.
- Description:** A text input field with the placeholder 'Type additional information to describe the account, su...'.
- Main Phone:** A text input field with the placeholder 'Type the main phone number for this account.'.

A red box highlights the 'Account Name' field in the left sidebar, and a blue box highlights the 'Account Name' field in the right panel, indicating the dynamic binding between them.

Expressions

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

The screenshot displays the Power Automate interface. A flow is shown with three steps: 'Manually trigger a flow', 'Delay', and 'Send a push notification'. The 'Delay' step is selected, and its configuration is visible. The 'Count' field is set to 'Specify the count of unit to delay' and the 'Unit' is set to 'Minute'. The 'Dynamic content' menu is open, showing the 'Expression' tab. The menu lists various functions under different categories: 'String functions' (add, addDays, addHours, addMinutes, addProperty, addSeconds, addToTime), 'Collection' (contains, length), and 'Logical functions' (if, equals, and). The 'add' function is highlighted in the 'String functions' category.

Many reasons to use expressions



Change Data Type



Do Simple Math or Text Work



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



Check for Empty Data



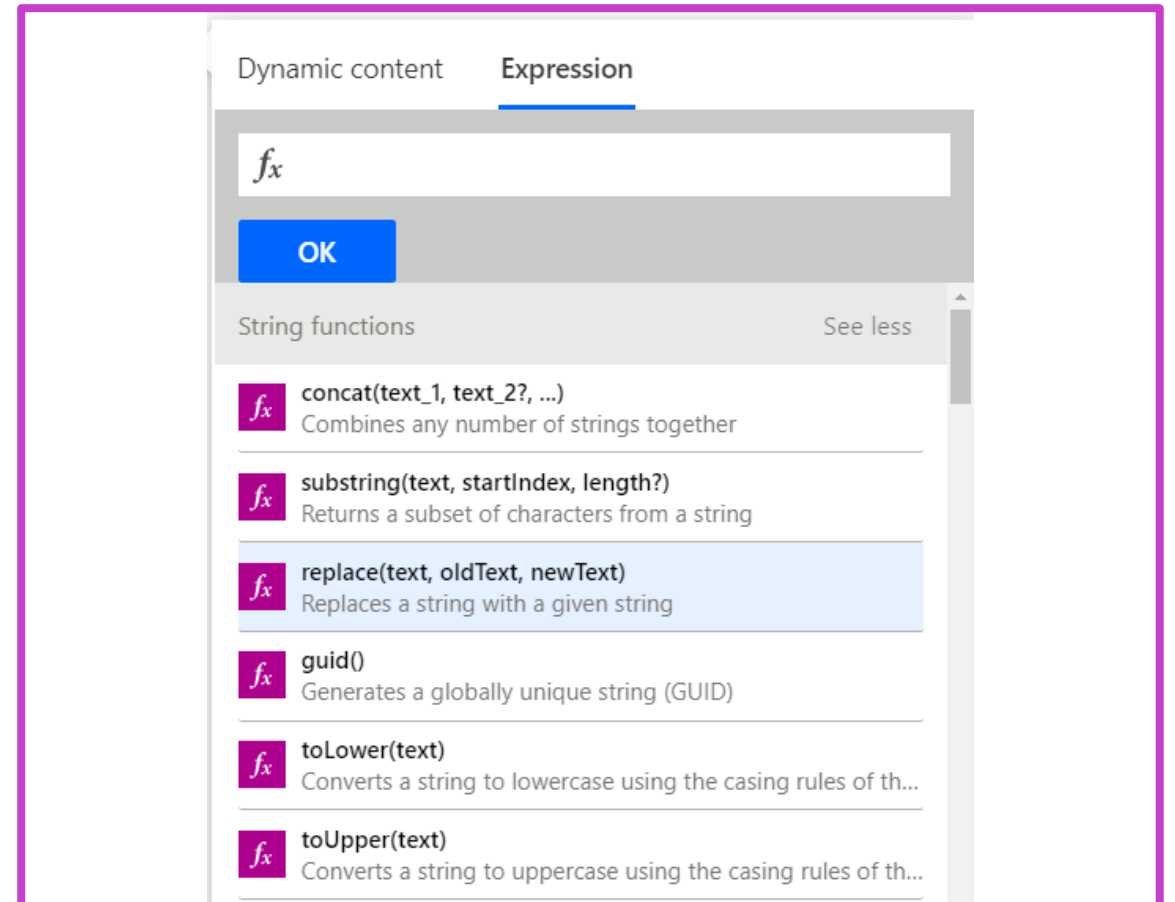
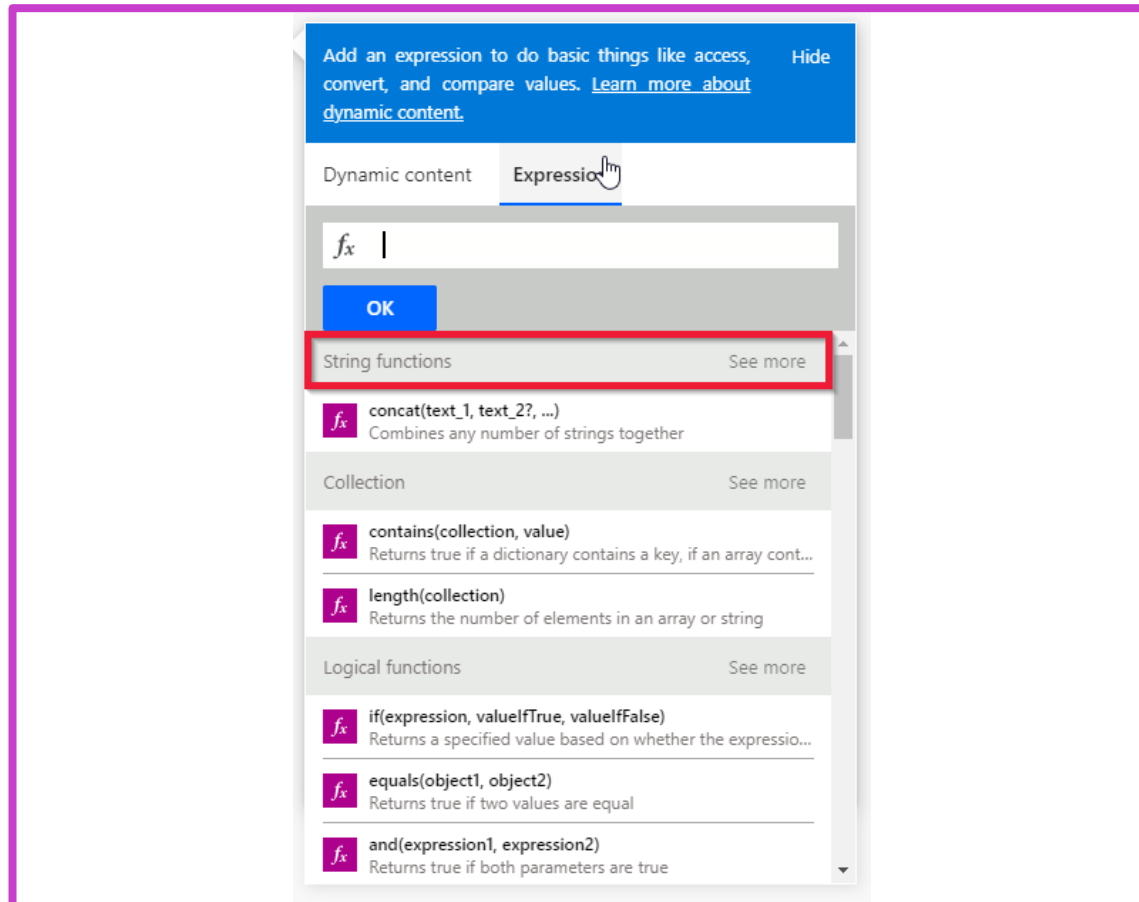
Make Decisions ("If" conditional statements)



Work with Data Sets (lists)

Types of expressions

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



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

For strings use single quotes

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

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
 - $< \leq \geq > == \neq$ all bad as you're comparing strings
 - Convert to ticks
 - Use substring to restrict what are we comparing, e.g., first 10 characters would give you the date only

Add an action



[← Return to search](#)



Date Time

Add to time

In App ⓘ

Convert time zone

In App ⓘ

Current time

In App ⓘ

Get future time

In App ⓘ

Get past time

In App ⓘ

Subtract from time

In App ⓘ



Time

- addToTime
- convertTimeZone
- getPastTime
- getFutureTime
- parseDateTime

Preparing Dates for Display

The screenshot shows the 'Convert time zone' application interface. The 'Parameters' tab is active, displaying the following settings:

- Base Time ***: Created On x
- Source Time Zone ***: (UTC-08:00) Coordinated Universal Time-08
- Destination Time Zone ***: (UTC-07:00) Mountain Time (US & Canada)
- Time Unit**: Short date pattern - 6/15/2009 [d]

The 'Time Unit' dropdown menu is open, showing a list of date and time patterns. The 'Short date pattern - 6/15/2009 [d]' option is highlighted.

Available patterns in the dropdown:

- 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]**
- 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'])),  
    8640000000000  
  ))
```

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('<timestamp>')** – 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

The image shows two configuration panels for variables in a workflow tool, both featuring a purple icon with a curly brace and an 'x'.

Initialize variable panel:

- Parameters tab:** Active tab.
- Name:** A text input field with the placeholder "Enter variable name".
- Type:** A dropdown menu with "Boolean" selected. The dropdown is open, showing a list of data types: Boolean, Integer, Float, String, Object, and Array.

Set variable panel:

- Parameters tab:** Active tab.
- Name:** A dropdown menu with "Name" selected.
- Value:** A text input field with the placeholder "Enter variable value".

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 :**

["แอปเปิ้ล", "ส้ม", "กล้วย"]

- **Array of Objects:**

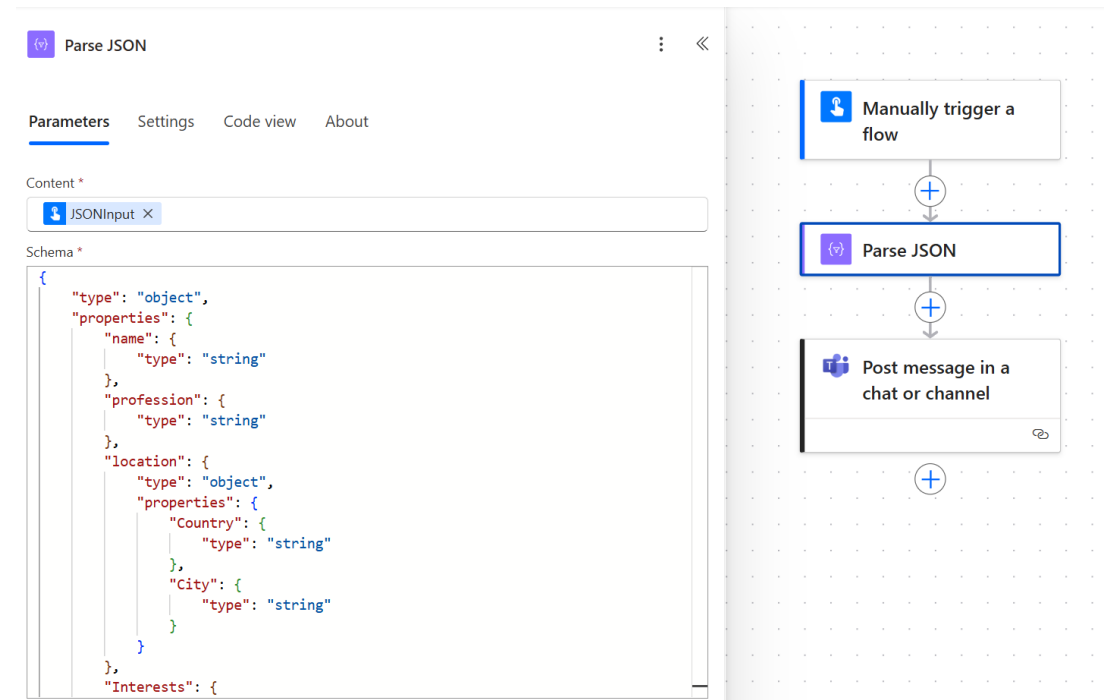
```
[  
  { "id": 1 , "name": "สมชาย" } ,  
  { "id": 2 , "name": "สมหญิง" }  
]
```


Key-Value ใน 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):

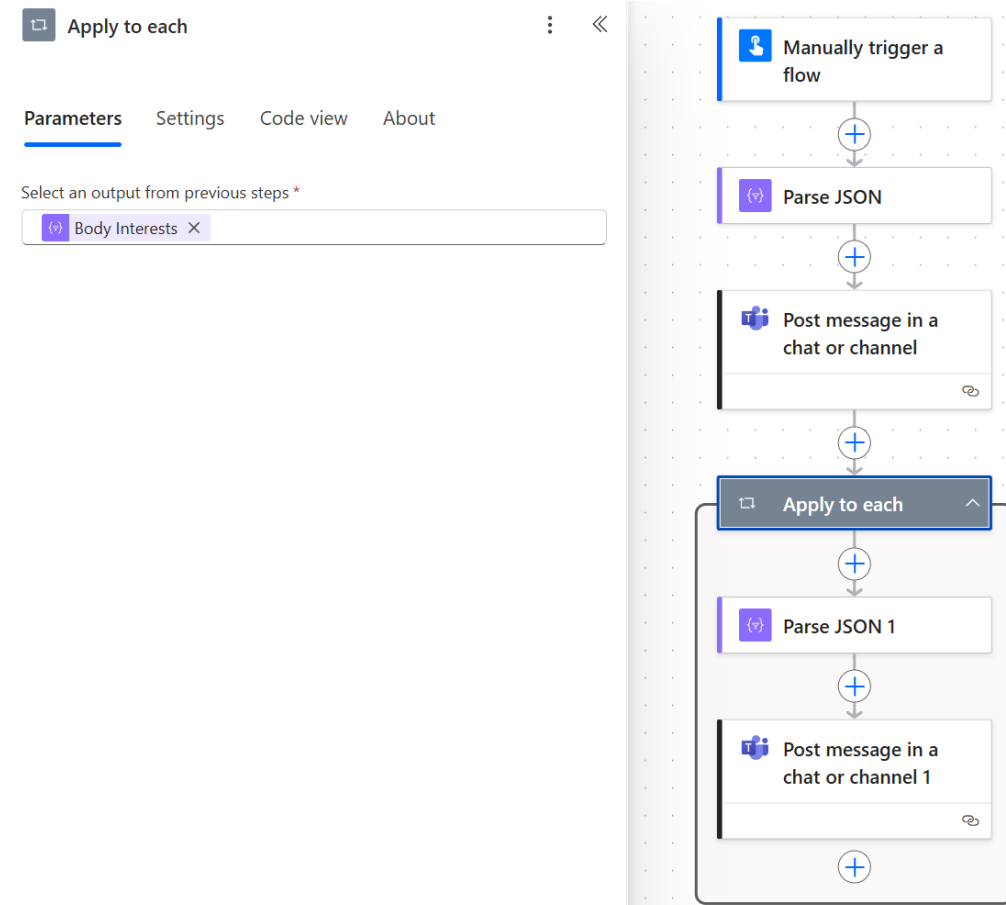
- Use the **Parse JSON Action**.
- Input the JSON String into the **Content** field.
- **Generate Schema** from a sample payload.

2. After Parse JSON:

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

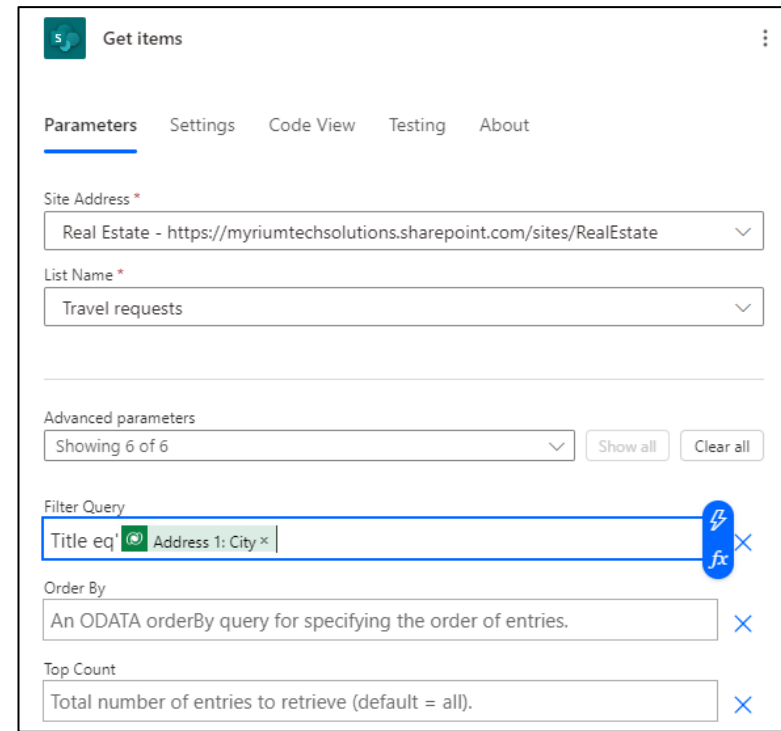
3. extraction Operation:

- Extract items



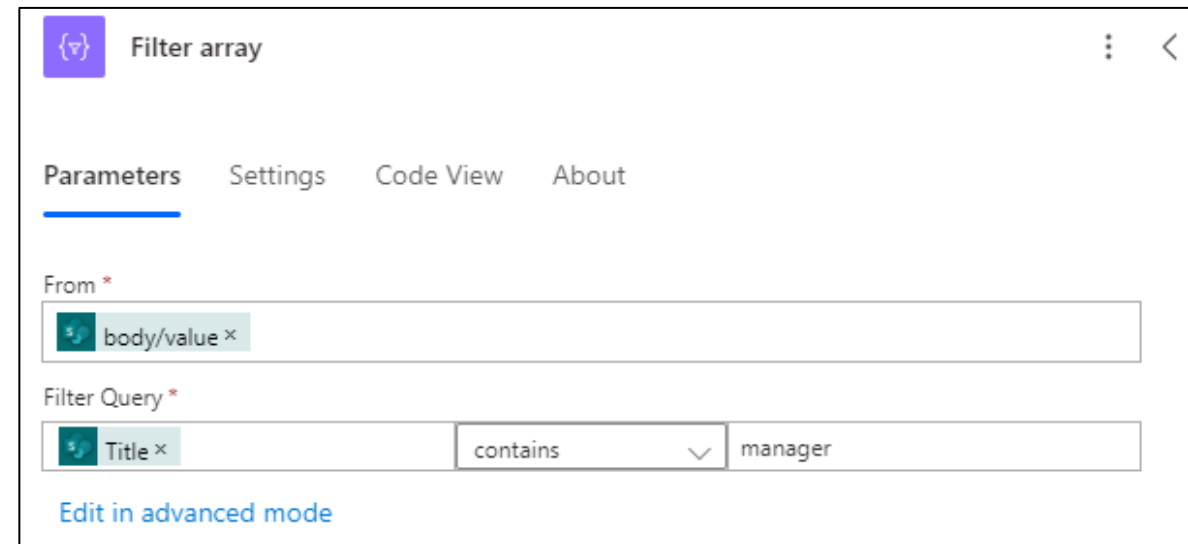
Filtering

- 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()



The screenshot shows the 'Get items' connector configuration window. It has tabs for Parameters, Settings, Code View, Testing, and About. The Parameters tab is active. It contains the following fields:

- Site Address ***: A dropdown menu showing 'Real Estate - https://myriumtechsolutions.sharepoint.com/sites/RealEstate'.
- List Name ***: A dropdown menu showing 'Travel requests'.
- Advanced parameters**: A section with a dropdown showing 'Showing 6 of 6', and 'Show all' and 'Clear all' buttons.
- Filter Query**: A text box containing 'Title eq' followed by a green icon and 'Address 1: City' followed by a red 'x' icon. To the right of the text box is a blue lightning bolt icon and a red 'x' icon.
- Order By**: A text box containing 'An ODATA orderBy query for specifying the order of entries.' with a red 'x' icon to its right.
- Top Count**: A text box containing 'Total number of entries to retrieve (default = all).' with a red 'x' icon to its right.



The screenshot shows the 'Filter array' connector configuration window. It has tabs for Parameters, Settings, Code View, and About. The Parameters tab is active. It contains the following fields:

- From ***: A dropdown menu showing 'body/value' with a red 'x' icon.
- Filter Query ***: A section with a dropdown menu showing 'Title' with a red 'x' icon, a dropdown menu showing 'contains' with a red 'x' icon, and a text box containing 'manager'.
- Edit in advanced mode**: A blue link at the bottom.

Selecting Arrays

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

Create an array of objects

The screenshot shows the 'Select' tool interface with the 'Parameters' tab selected. The 'From' field is set to 'body/value'. The 'Map' section is active, displaying a table with two rows: 'Title' mapped to 'Title' and 'Travel Reason' mapped to 'Reason for travel'. A third row is labeled 'Enter key' and 'Enter value'. A blue 'fx' icon is visible on the right side of the table.

Useful for passing this array to another action

Create an array strings, numbers, Booleans etc.

The screenshot shows the 'Select' tool interface with the 'Parameters' tab selected. The 'From' field is set to 'Body'. The 'Map' section is active, displaying a code editor with a JSON object:

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

. A blue 'fx' icon is visible on the right side of the code editor.

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 screenshots of the Microsoft Power Automate 'Send an email (V2)' action interface, illustrating how to format a list of data for an email body.

Top Screenshot: Using the 'Join' connector

- Action:** Send an email (V2) 2
- Parameters:** 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 parameters:** Showing 1 of 7, Importance: Normal.
- Body Field:** The 'Join' connector is selected, which concatenates the items from the 'List group members' action into a single string, separated by semicolons.
- Right Pane:** Shows the output of the 'List group members' action, including fields like Display Name, Given Name, Job Title, Mail, and Surname.

Bottom Screenshot: Using the 'Create HTML table' connector

- Action:** Send an email
- Parameters:** To (Enter part of a name or email address to find people), Subject (Specify the subject of the mail), Body (Specify the body of the mail).
- Advanced parameters:** Showing 1 of 7, Importance: Normal.
- Body Field:** The 'Create HTML table' connector is selected, which formats the output of the 'List group members' action into an HTML table structure.
- Right Pane:** Shows the output of the 'List group members' action, including fields like Display Name, Given Name, Job Title, Mail, and Surname.

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

Add an action ✕

🔍 condition ✕


Runtime

Select a runtime ▼

Action Type

Actions ▼

☒ Group by Connector

 Control

In App

Condition

In App ⓘ

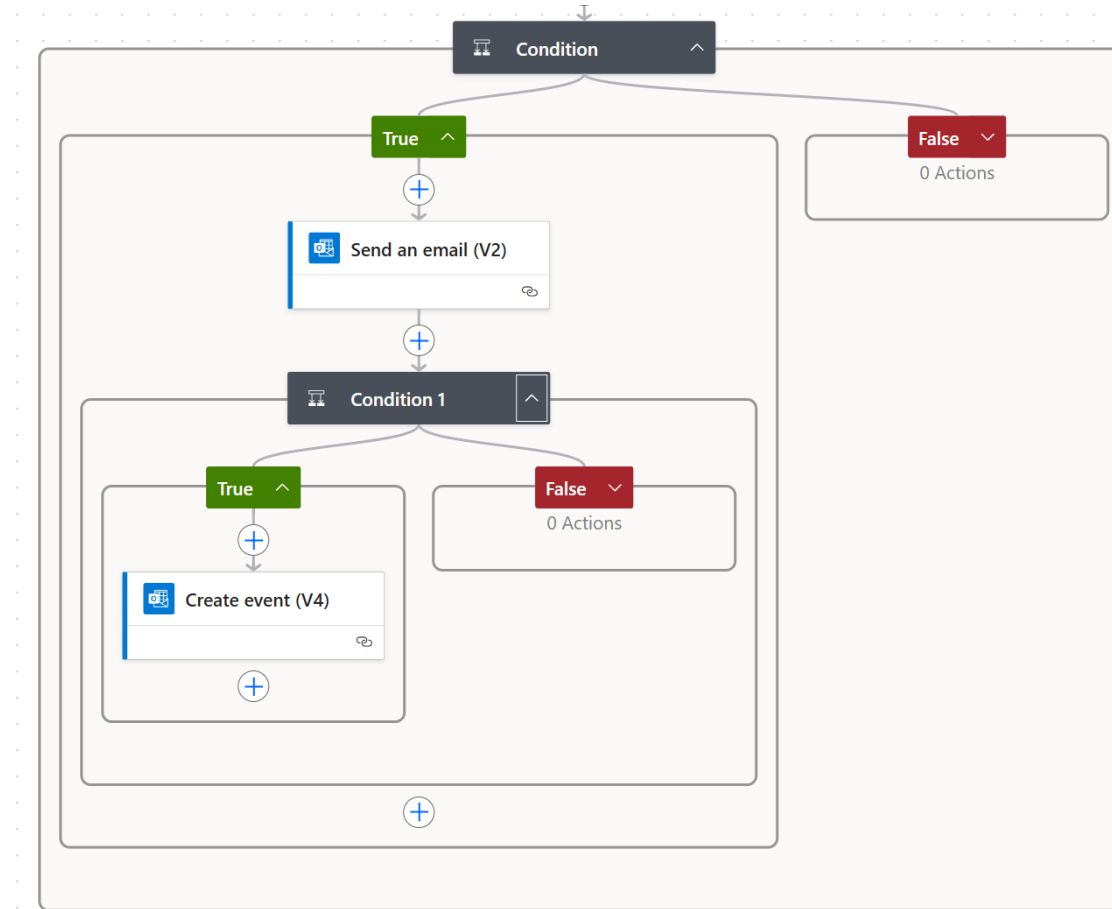
Do until

In App ⓘ

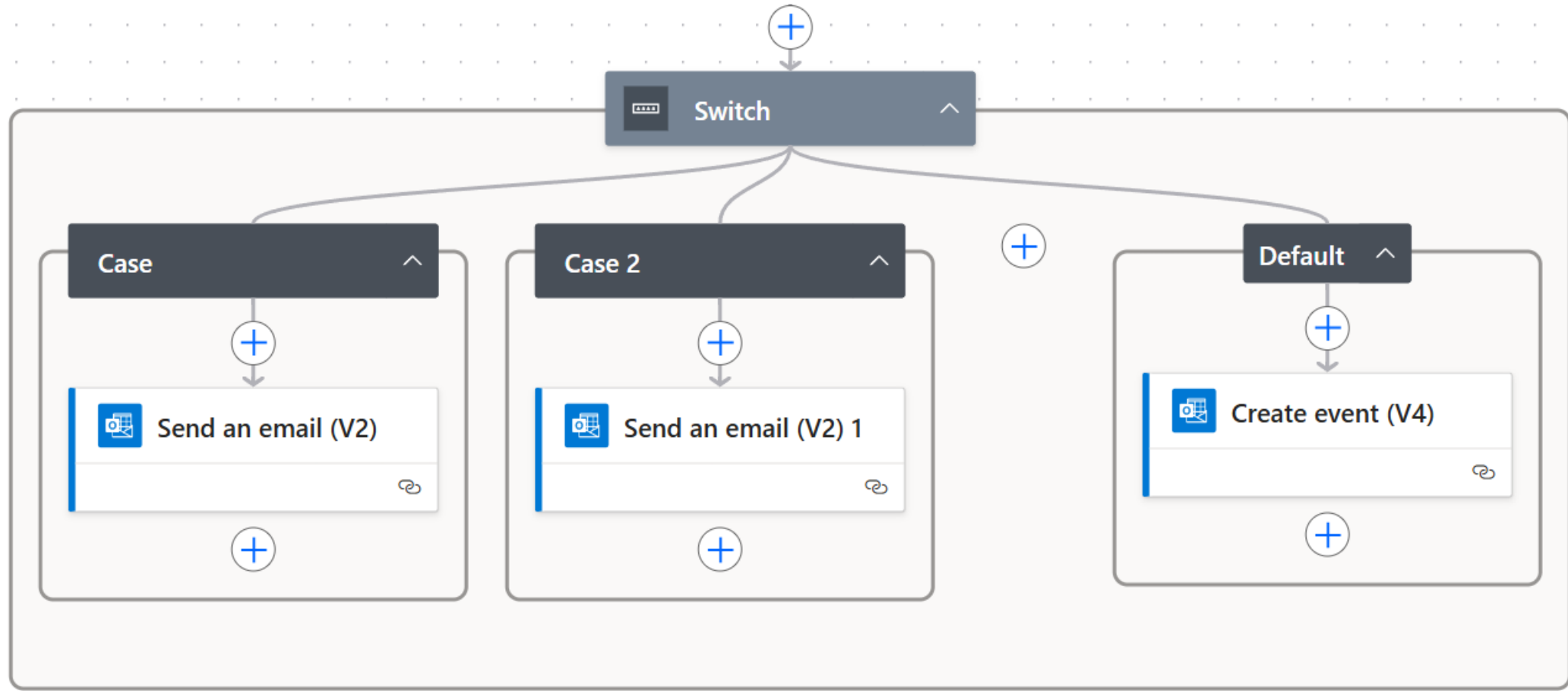
ⓘ

See more

Nest Condition



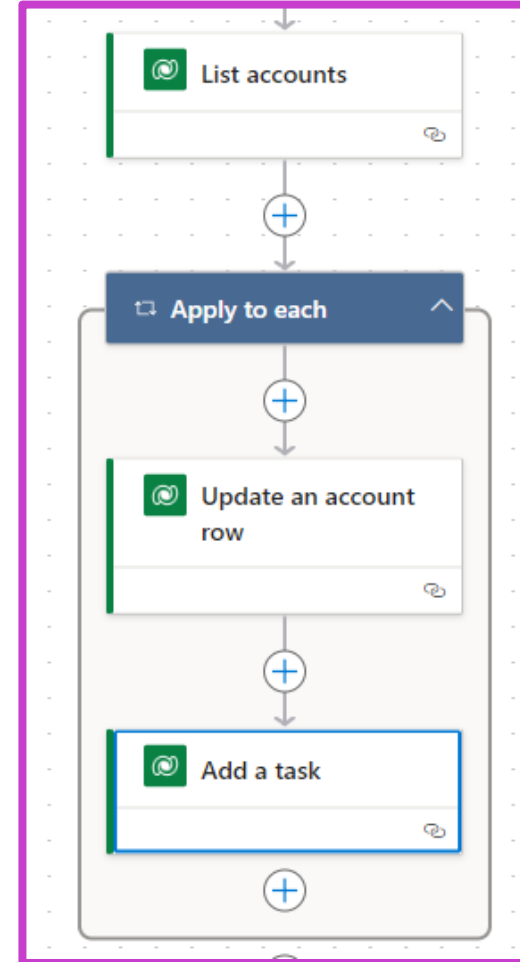
Switch



Loops

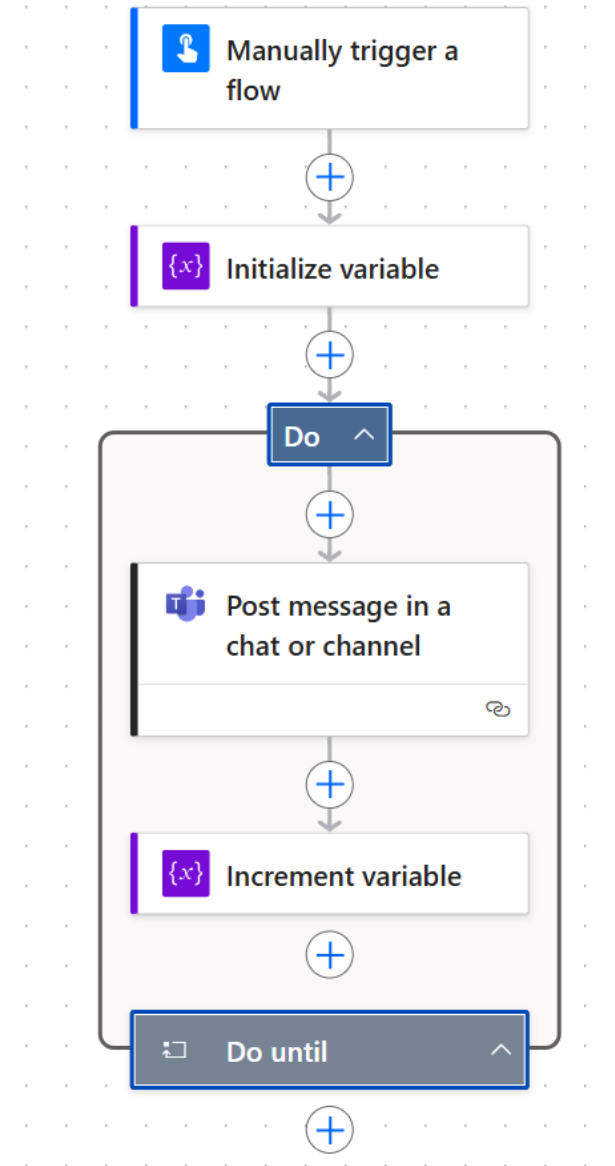
Apply for each

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



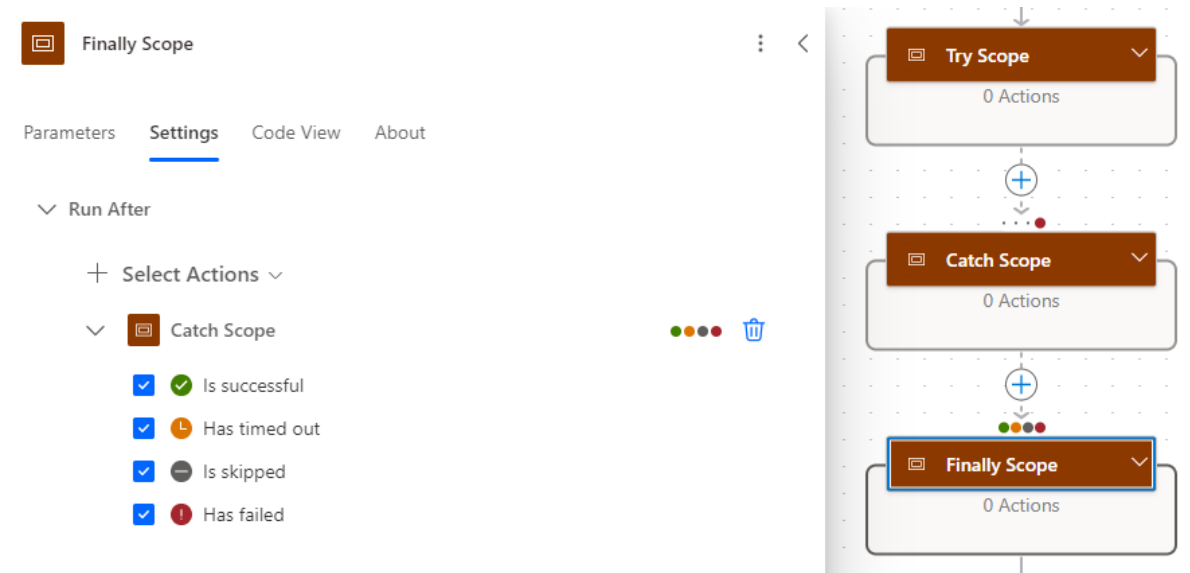
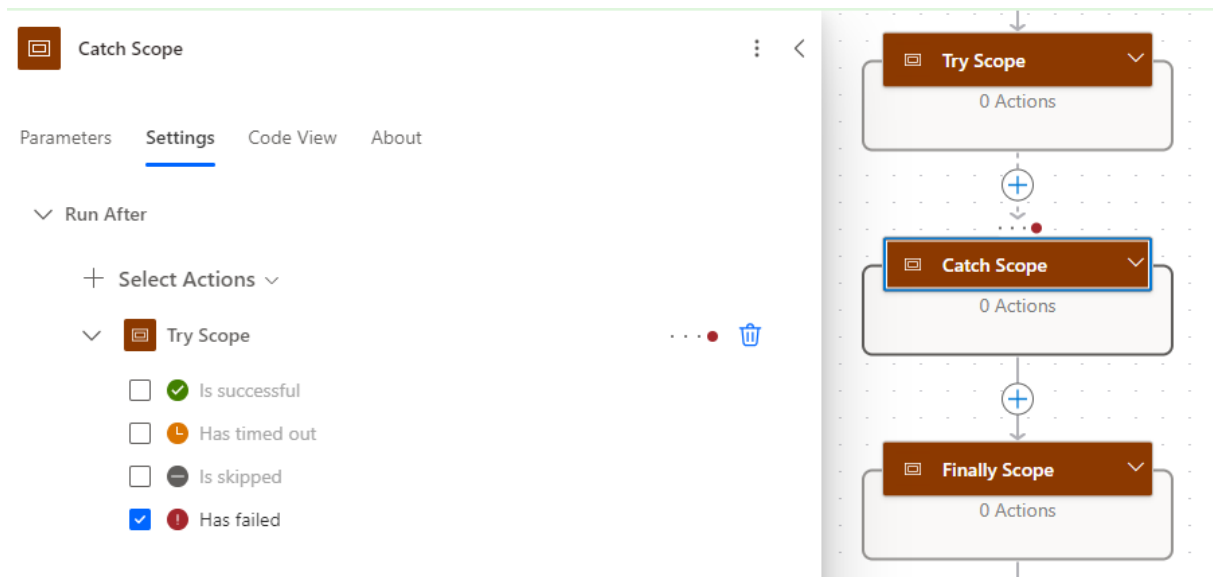
Loops Do-Until

Executes until a condition is met



Scope for Action Grouping

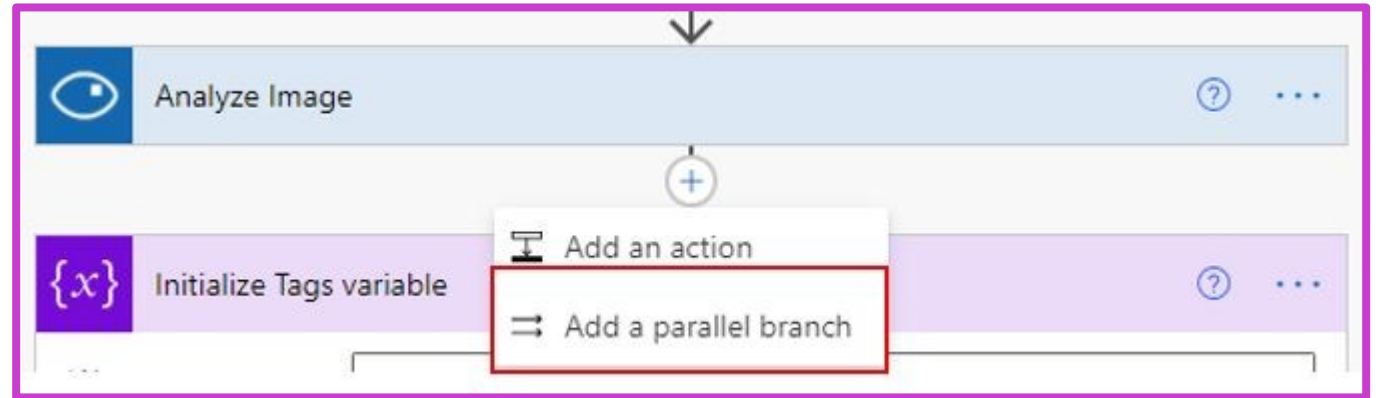
- Catch only runs when Try has failed
- When Try runs successfully, Catch scope is skipped
- Finally runs regardless of the Catch execution including when it is skipped




Branching

Parallel branches

- Error handling
- Parallel approvals



Delay / Delay Until / Timeout

 Delay

Parameters

Settings

Code view


About


Count ^{*}


Unit ^{*}

Minute

▼

 Delay





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

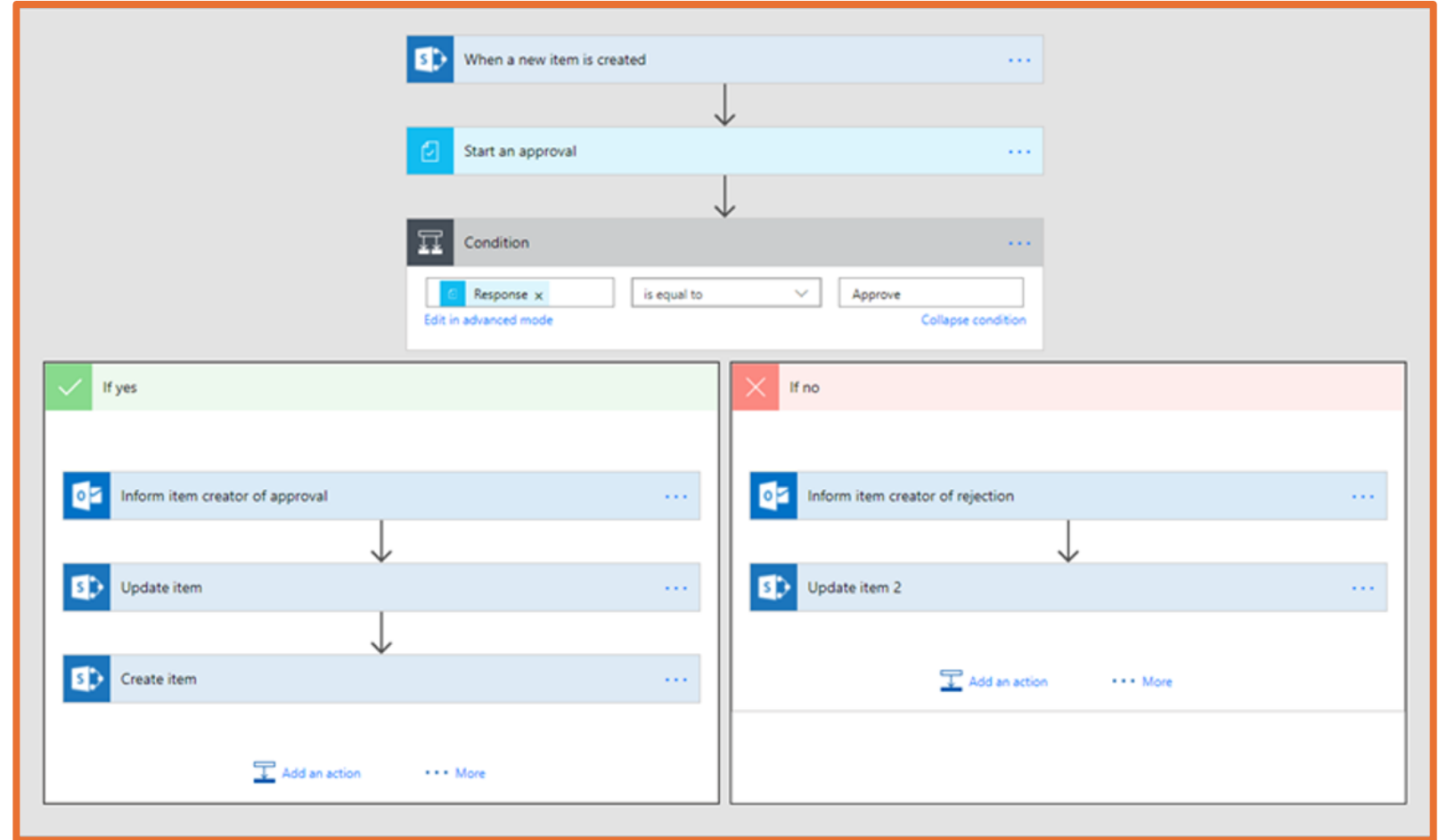
Wait for an approval



Approvals components

Components

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

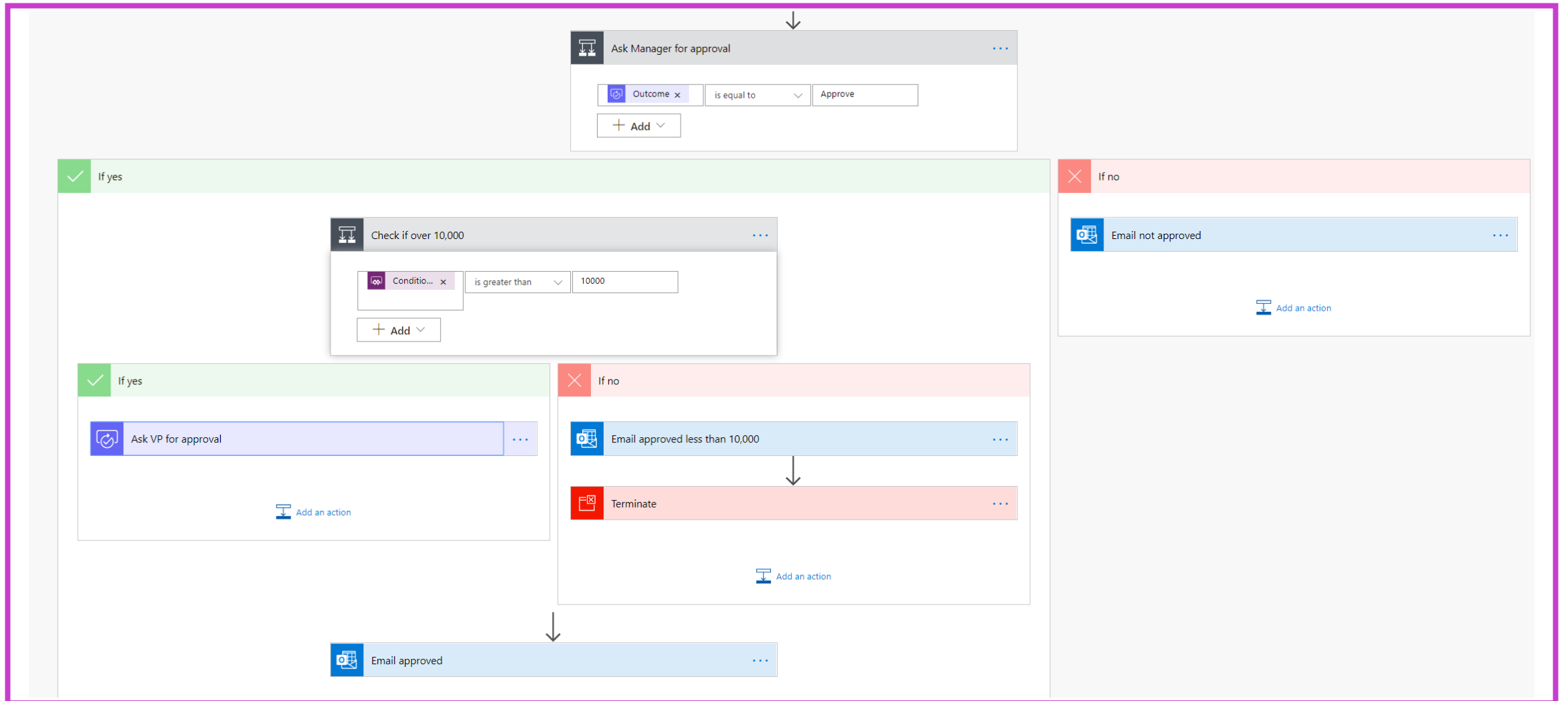


Example: Purchase order approval

The diagram illustrates a PowerApps flow for purchase order approval. It consists of three main steps connected by downward arrows:

- PowerApps**: The initial step, represented by a purple header bar.
- Get manager (V2)**: A step with an orange header bar. It includes a text input field for *** User (UPN)** containing the value `Getmanager(V...` and a **Show advanced options** link with a downward arrow.
- Start and wait for an approval**: A step with a blue header bar. It includes several input fields:
 - * Approval type**: A dropdown menu set to `Approve/Reject - First to respond`.
 - * Title**: A text input field containing `Please approve the Purchase Order`.
 - * Assigned to**: A text input field containing `Mail`.
 - Details**: A text input field containing `Markdown supported (see https://aka.ms/approvaldetails)`.
 - Item link**: A text input field containing `Add a link to the item to approve`.
 - Item link description**: A text input field containing `Describe the link to the item`.It also features a **Show advanced options** link with a downward arrow.

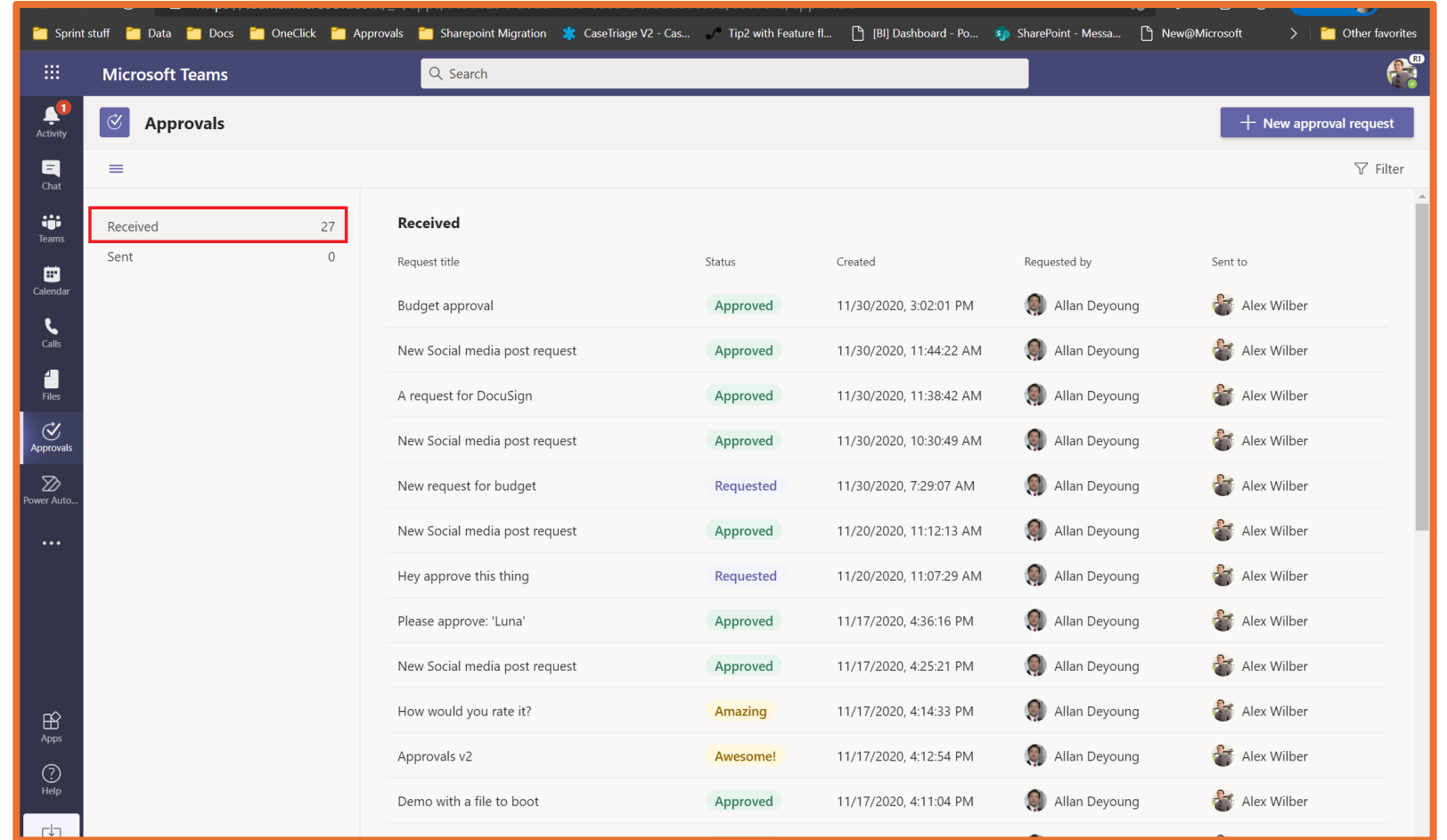
Example: Purchase order approval



Approvals

Process approvals

- Power Automate portal
- Power Automate mobile
- Microsoft Teams



The screenshot displays the Microsoft Teams interface with the 'Approvals' app open. The left sidebar shows the 'Approvals' tab selected, with a count of 27 for 'Received' and 0 for 'Sent'. The main area shows a list of approval requests with columns for Request title, Status, Created, Requested by, and Sent to.

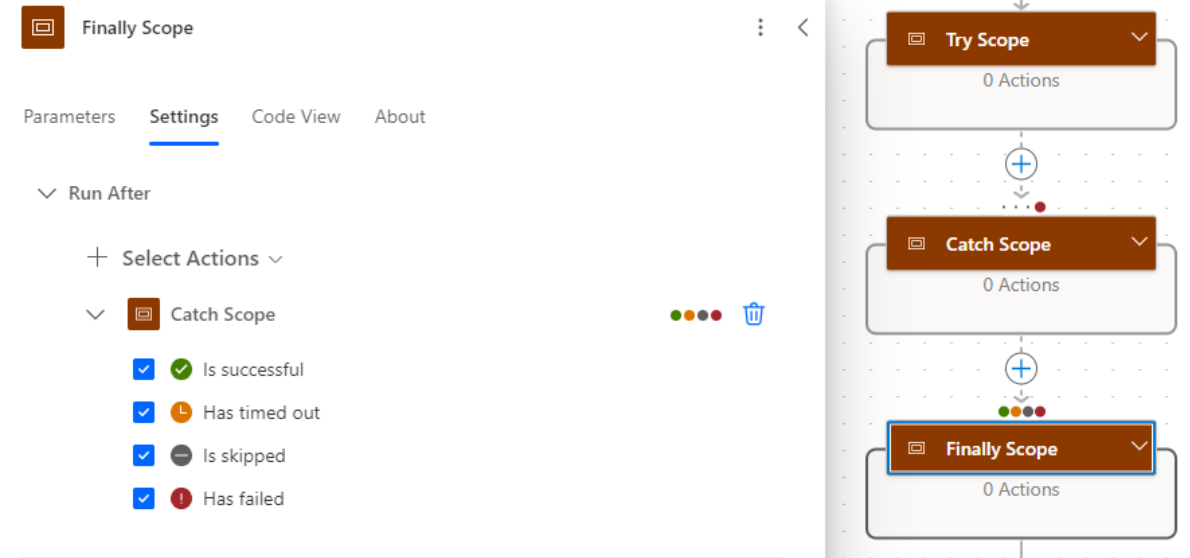
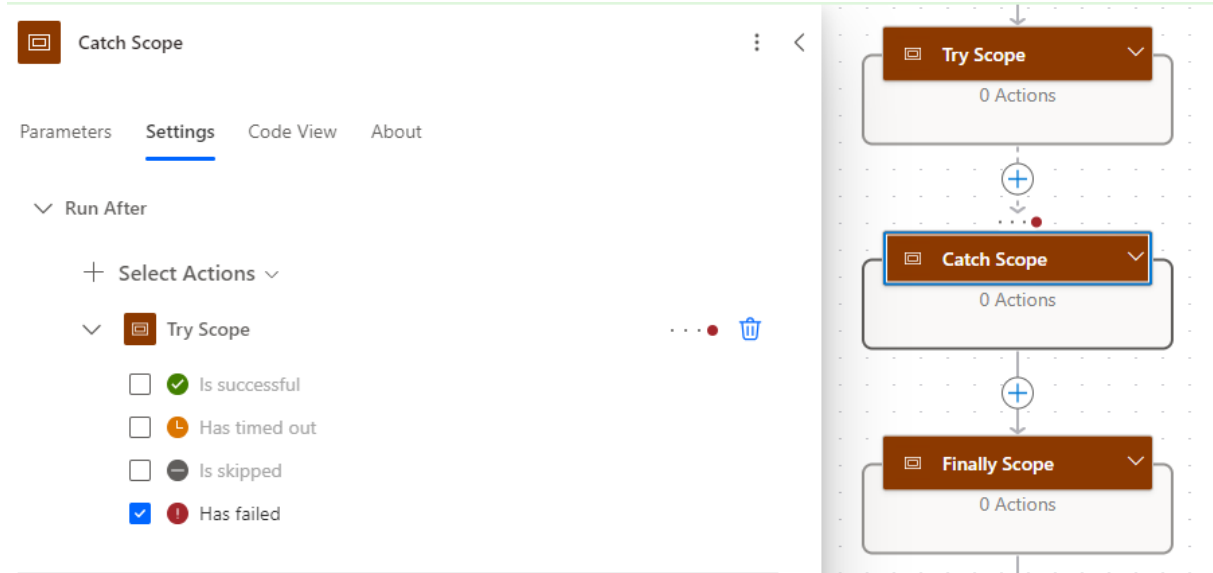
Request title	Status	Created	Requested by	Sent to
Budget approval	Approved	11/30/2020, 3:02:01 PM	Allan Deyoung	Alex Wilber
New Social media post request	Approved	11/30/2020, 11:44:22 AM	Allan Deyoung	Alex Wilber
A request for DocuSign	Approved	11/30/2020, 11:38:42 AM	Allan Deyoung	Alex Wilber
New Social 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

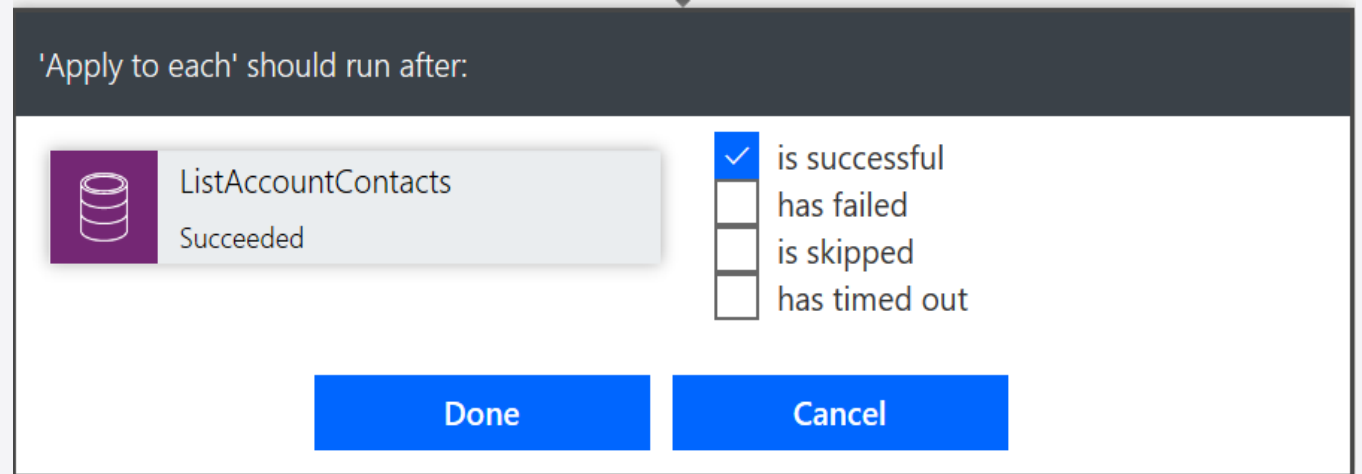
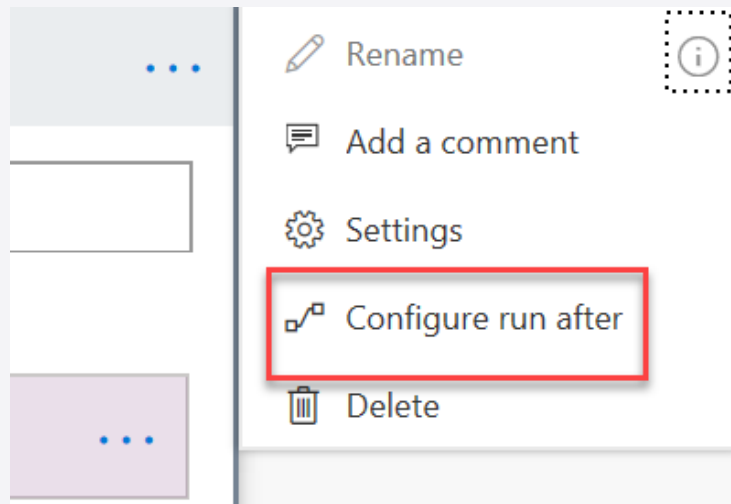
Best Practices and Error Handling

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

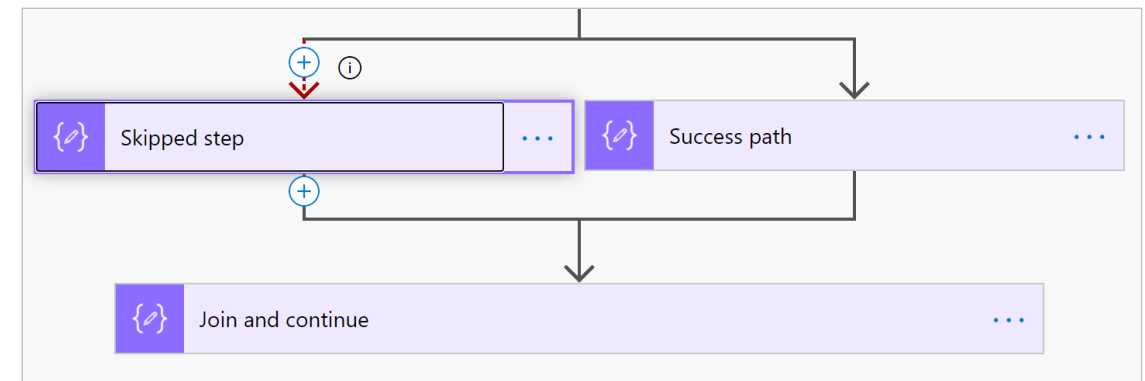
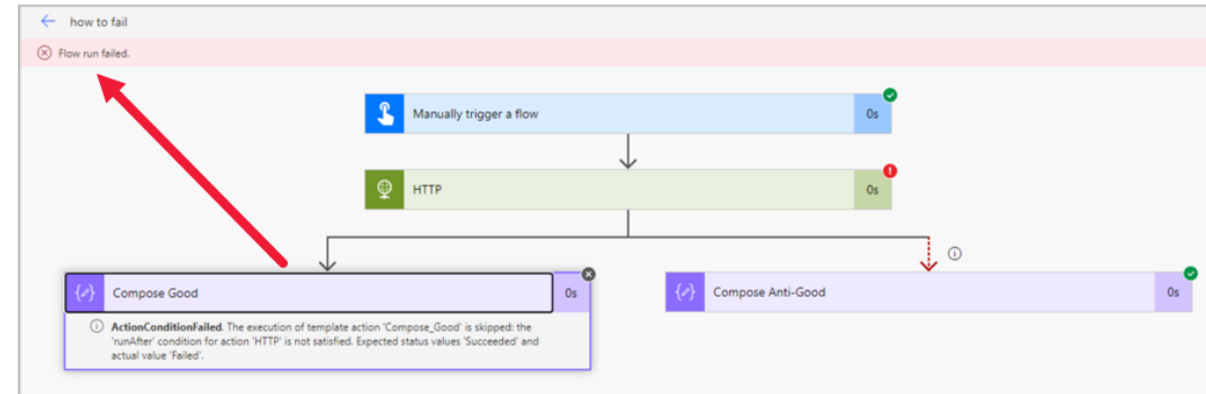
Failed

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

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.

▼

Default

▼

Default

None

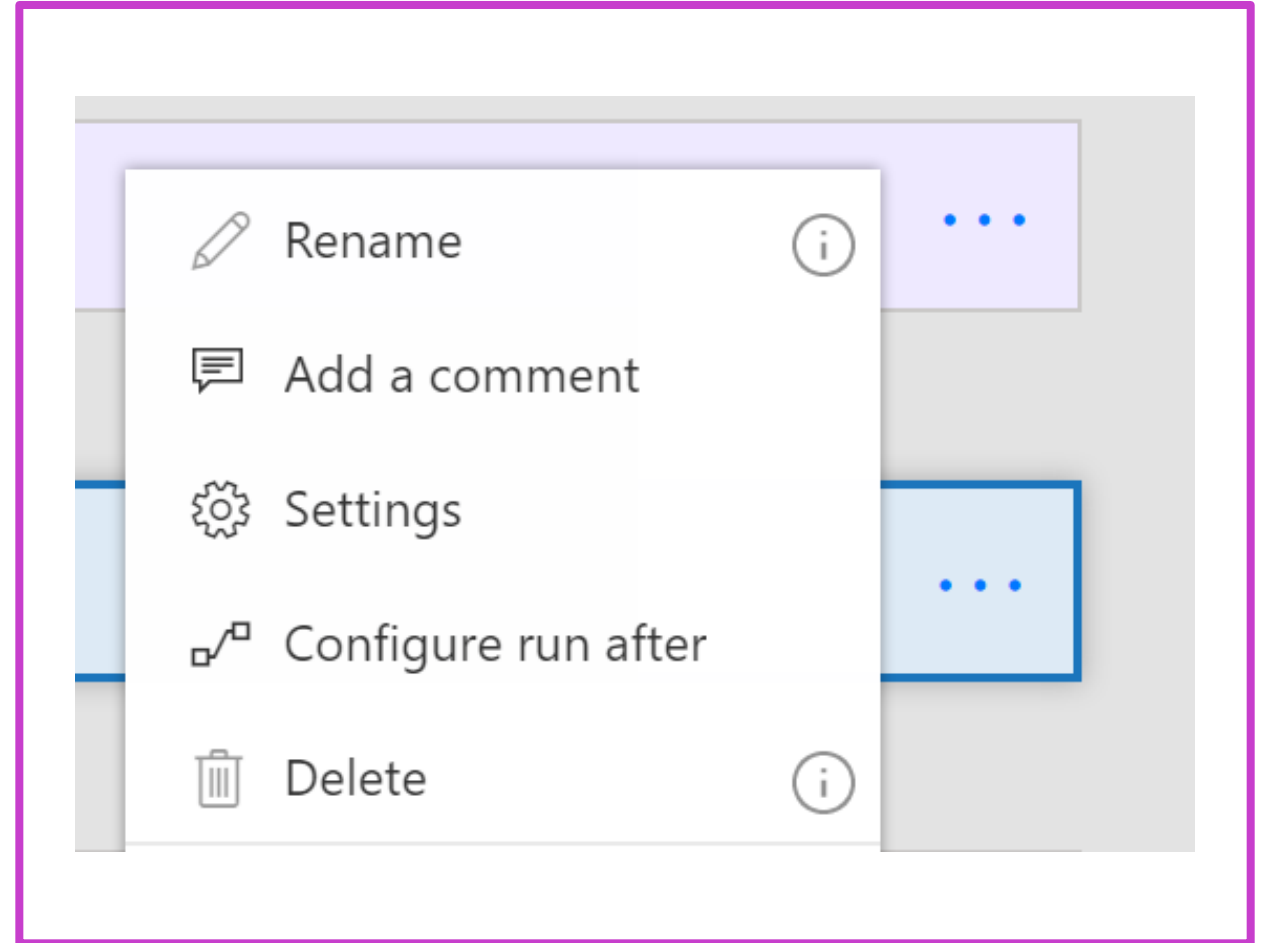
Exponential Interval

Fixed Interval

- 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

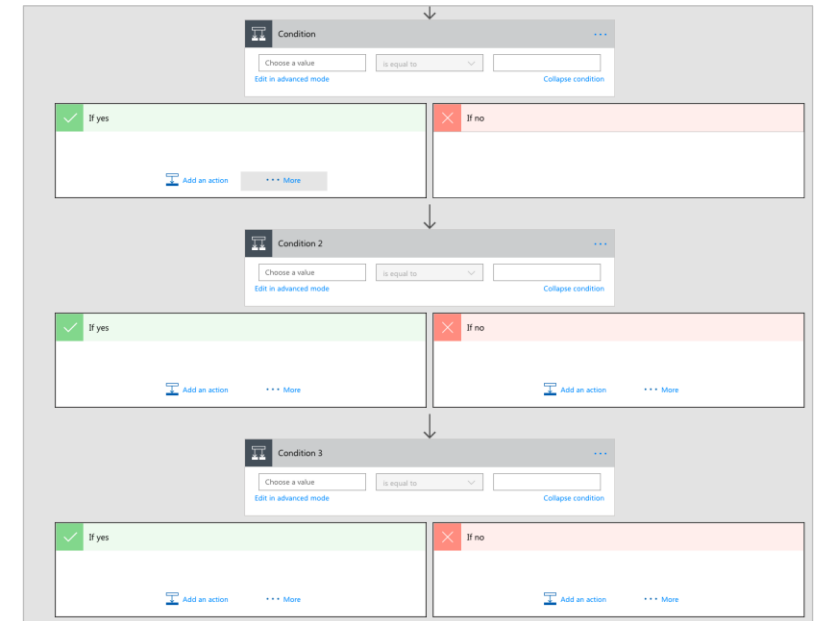
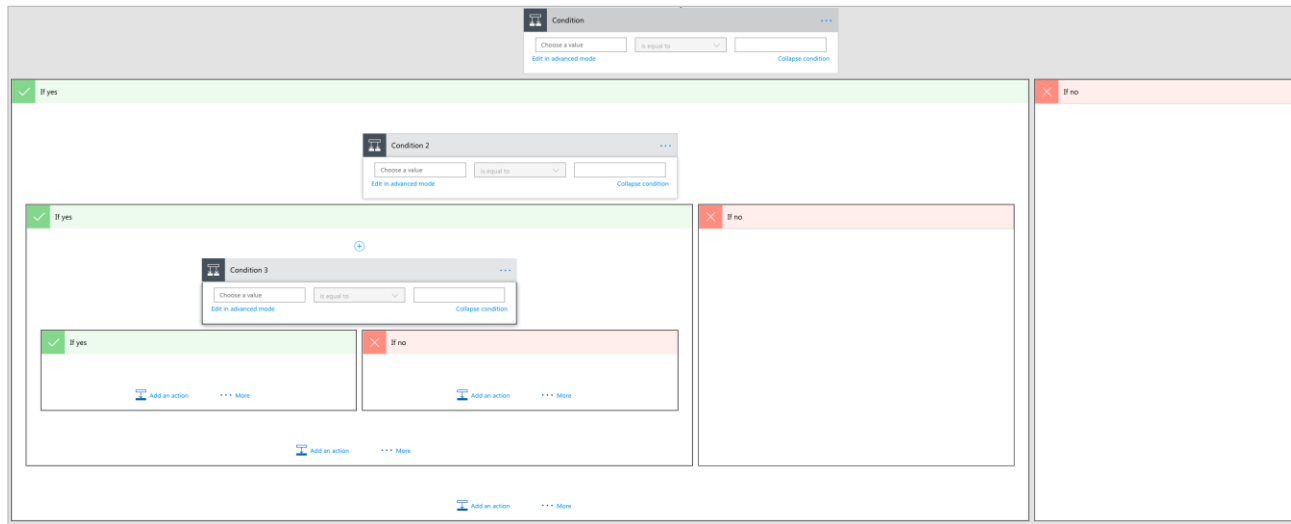
Action Settings

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

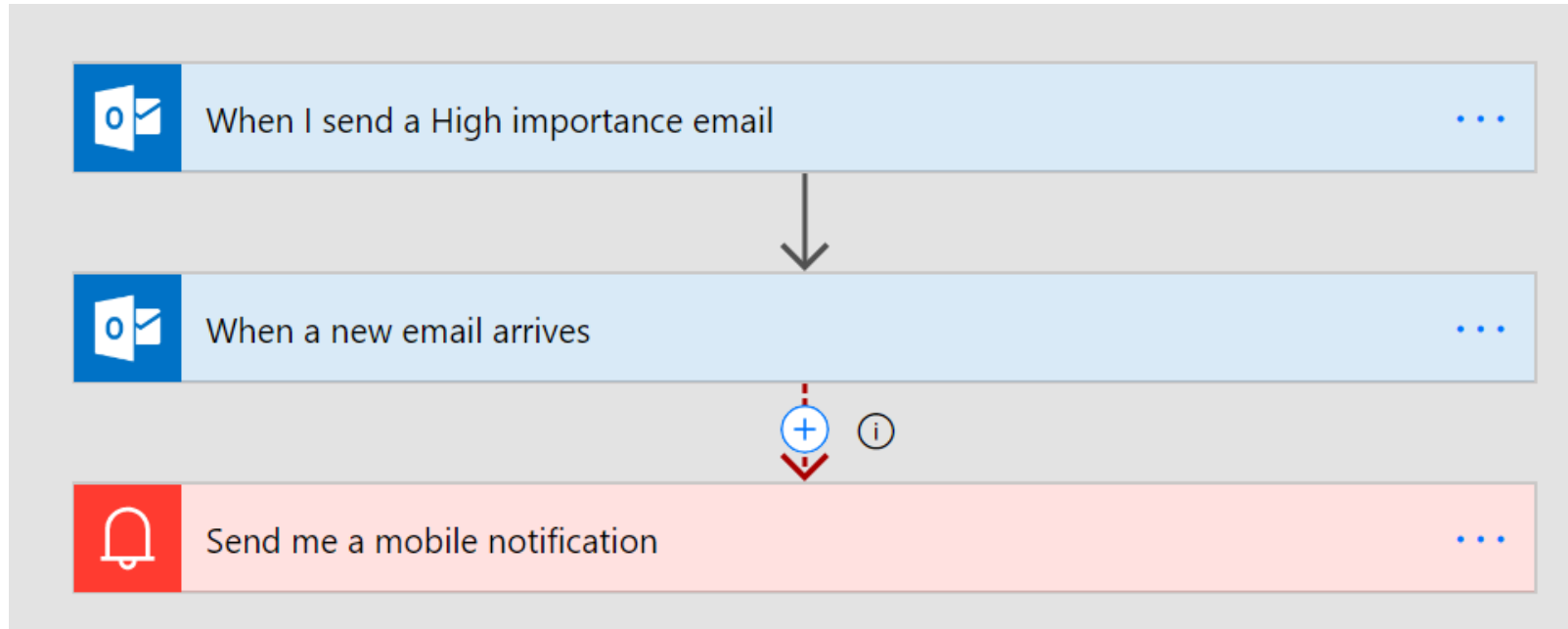


Terminate Flows

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

The screenshot displays the Trainocate workflow editor interface. At the top, a workflow step is labeled "Do until". Below it, a configuration bar shows the condition: `{x} RequestCompl...` is equal to `{x} true`. A modal window titled "Settings for 'Wait for request status update'" is open, showing the "Duration" field set to "P3D" (highlighted with a red box). The modal also includes sections for "Retry Policy" and "Tracked Properties". The background workflow shows a sequence of steps: "Handle timeout", "Compose email reminder", "Create email reminder", and "Send email reminder". Below the modal, a conditional flow is visible with "If yes" and "If no" branches, each containing a "Set variable" action.

Using Settings → Trigger Conditions

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

Settings for 'When a record is created, updated or deleted'

Custom Tracking Id
Set the tracking id for the run. For split-on this tracking id is for the initiating request.

Tracking Id

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

Concurrency Control
Limit number of concurrent runs of the flow, or leave it off to run as many as possible at the same time. Concurrency control changes the way new runs are queued. It cannot be undone once enabled.

Limit ☐ Off

Trigger Conditions
Specify one or more expressions which must be true for the trigger to fire.

+ Add

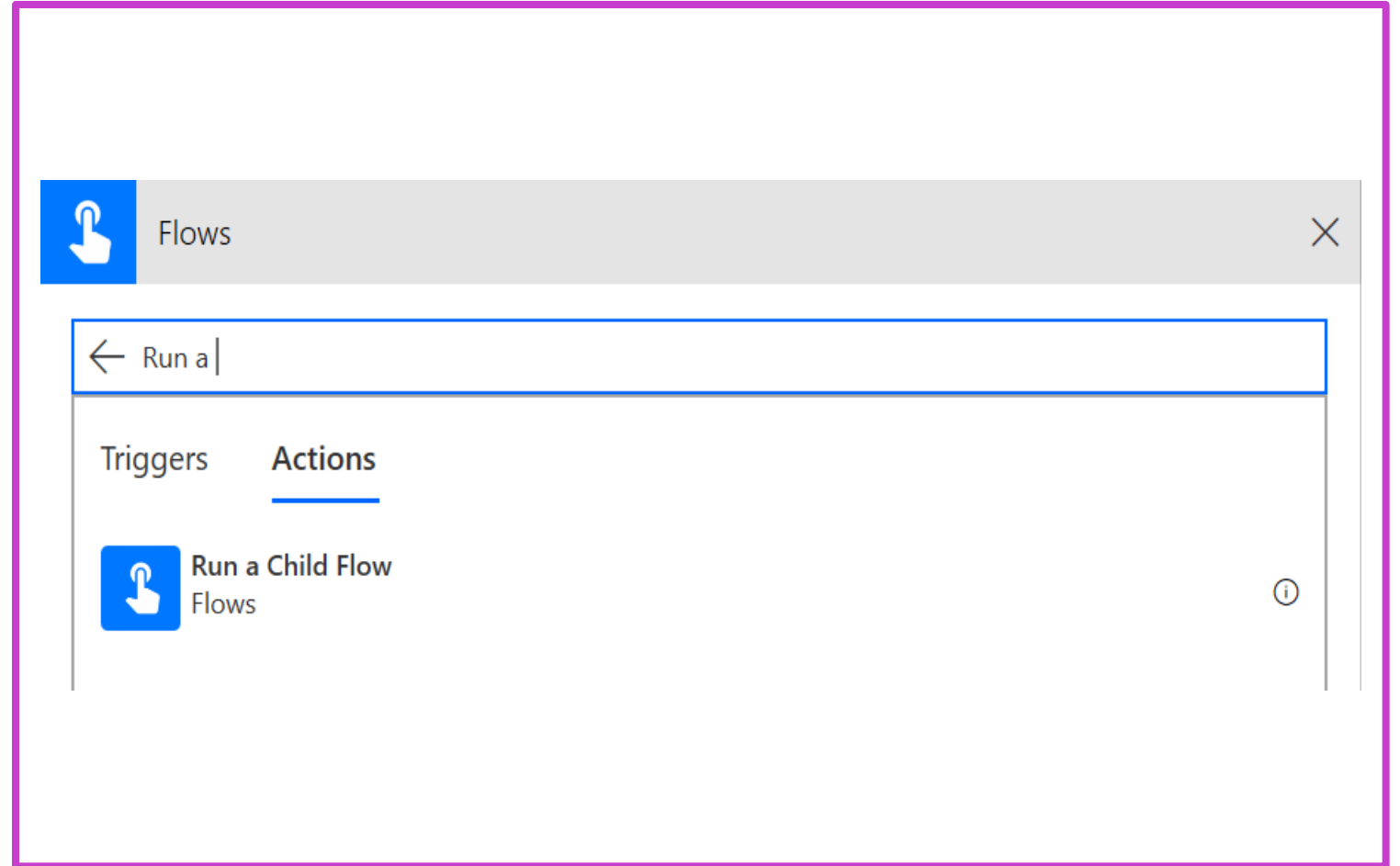
Done Cancel

Module 6

Parent and Child Flows

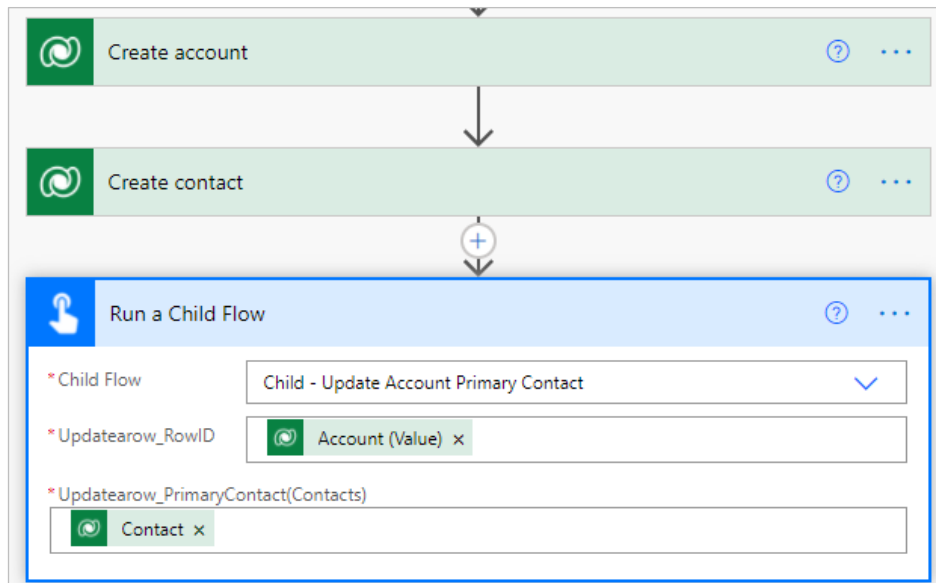
About Child Flows

- 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

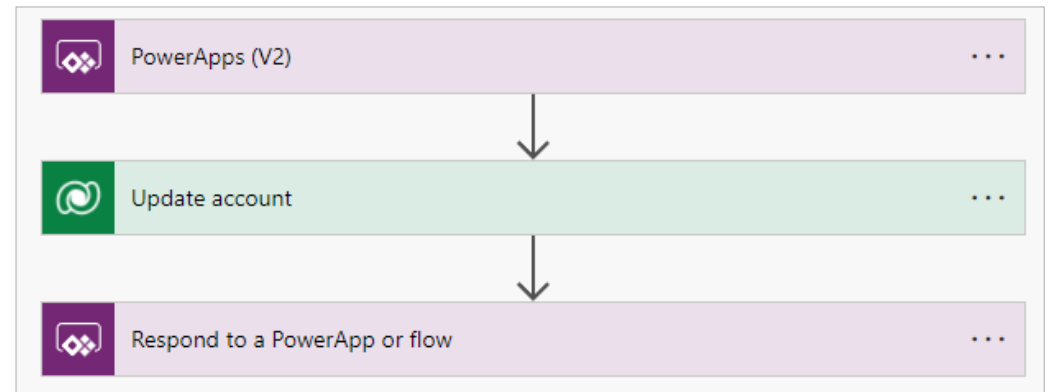


Example – Using a Child Flow

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 :

Evaluation: Power Automate
(Advanced) Intensive Workshop on
6-7 Aug 25



CATE



THANK YOU



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



(+66) 2 613 1033



thailand@trainocate.com



www.trainocate.com