

Getting Started with Power Automate



Agenda

- Power Automate Fundamentals
- Creating and Testing Flows
- Using Control-of-Flow Actions
- Writing Flow Expressions
- Automating Document Generation



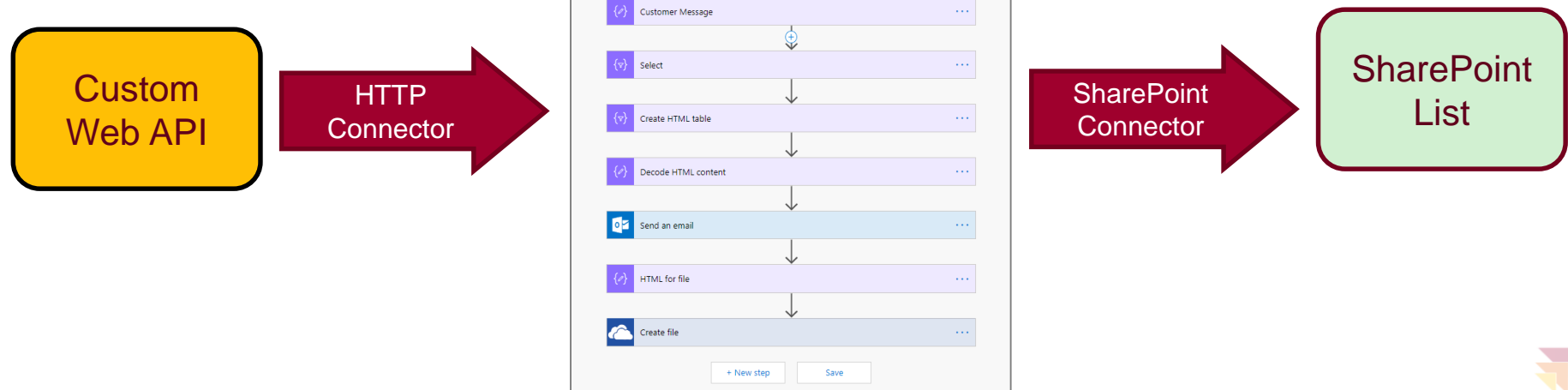
What is Power Automate?

- Service for automating business and workflow processes
 - You use Power Automate by creating and running "flows"
 - Power Automate provides browser-based flow editing experience
 - Power Automate originally went by the name of "Microsoft Flow"
- What can you automate by creating a flow?
 - Sending notifications
 - Importing and exporting data
 - Generating Microsoft Word documents
 - Automating approvals



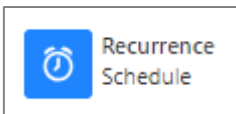
Building Blocks of Power Automate

- Flows are created using the following building blocks
 - **Triggers** - events that start a flow
 - **Actions** - tasks and operation executed by flow
 - **Services** - sources and destinations for data
 - **Connectors** - wrappers to communicate with service APIs

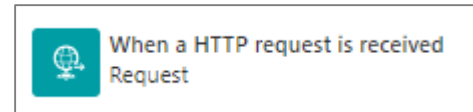
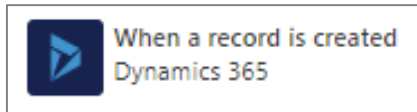
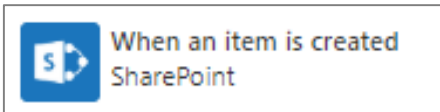


Flow Trigger Types

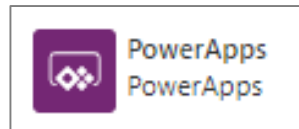
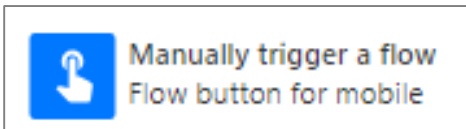
- Scheduled Flow Triggers
 - Runs periodically based on an interval



- Automated Flow Triggers
 - Runs when something happens

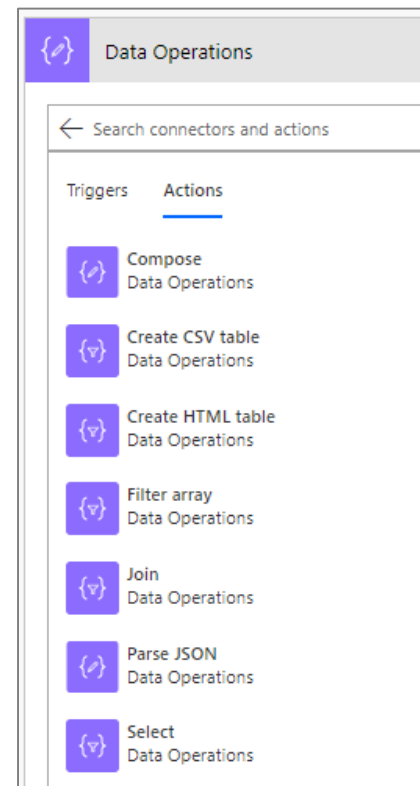
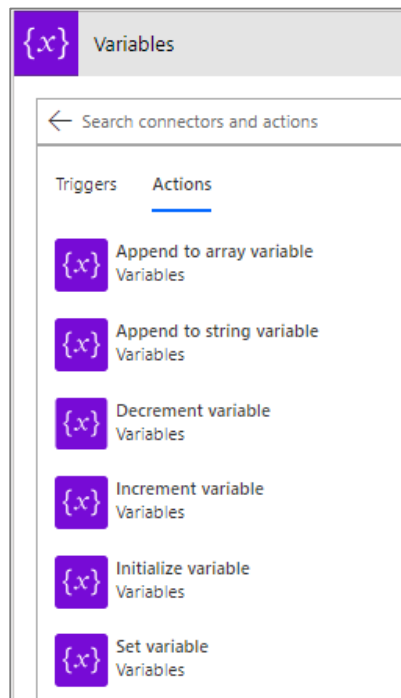
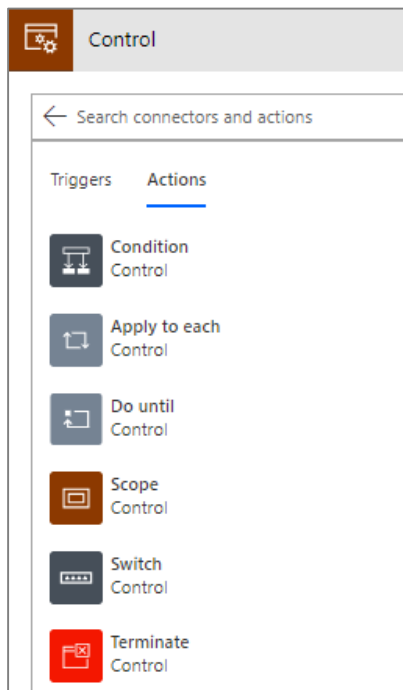


- On-demand Flow Triggers
 - Runs when a user clicks a button



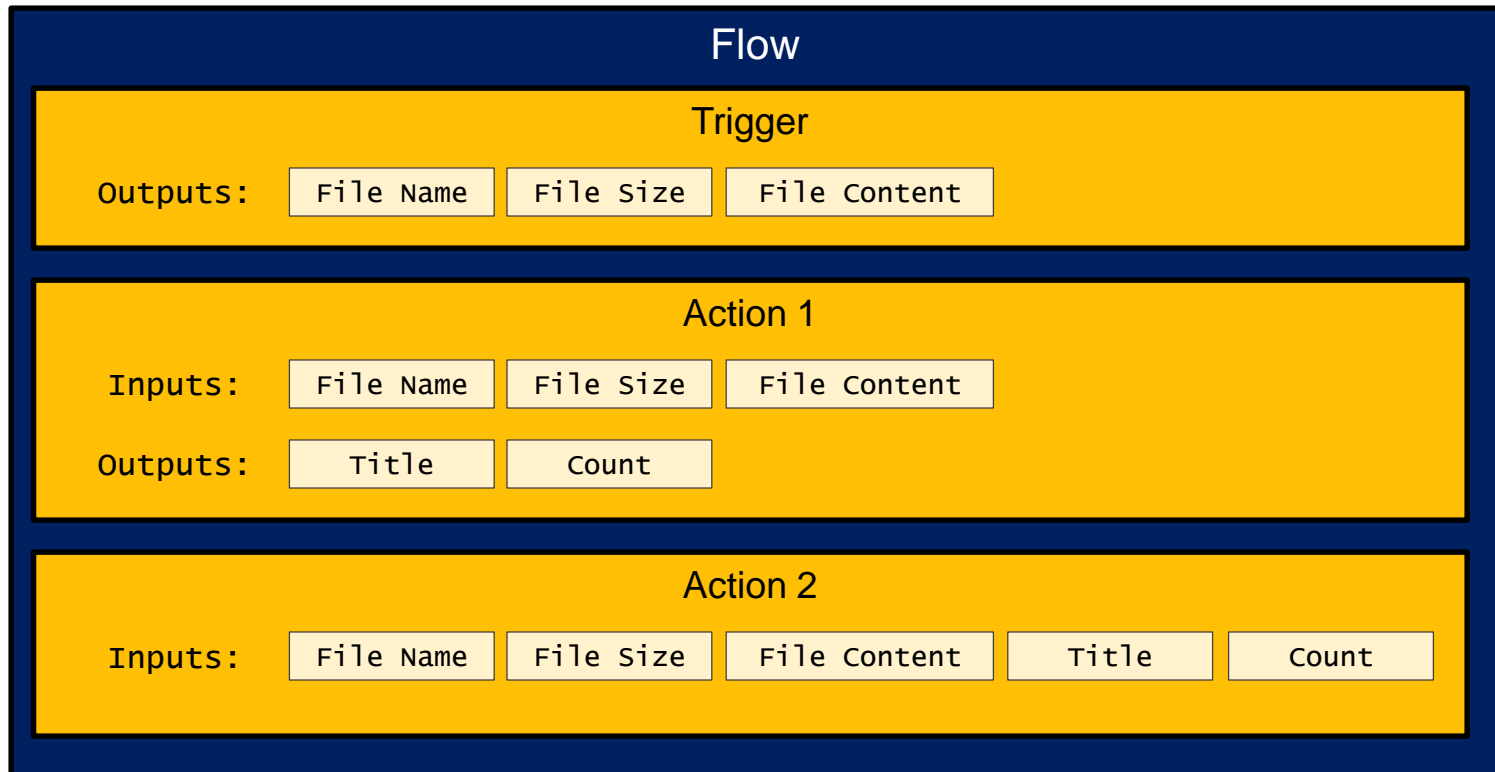
Core Action Categories

- **Control:** actions to provide control-of-flow
- **Variables:** actions to manage state within flow lifetime
- **Data operations:** action to process data & prepare content



Data Automatically Flows from Step to Step

- Data in flows added by step outputs
 - Data added in step output is available in later steps
 - It's easy to configure step input data using output data in previous steps
 - Certain outputs displayed/hidden based on types of input and output



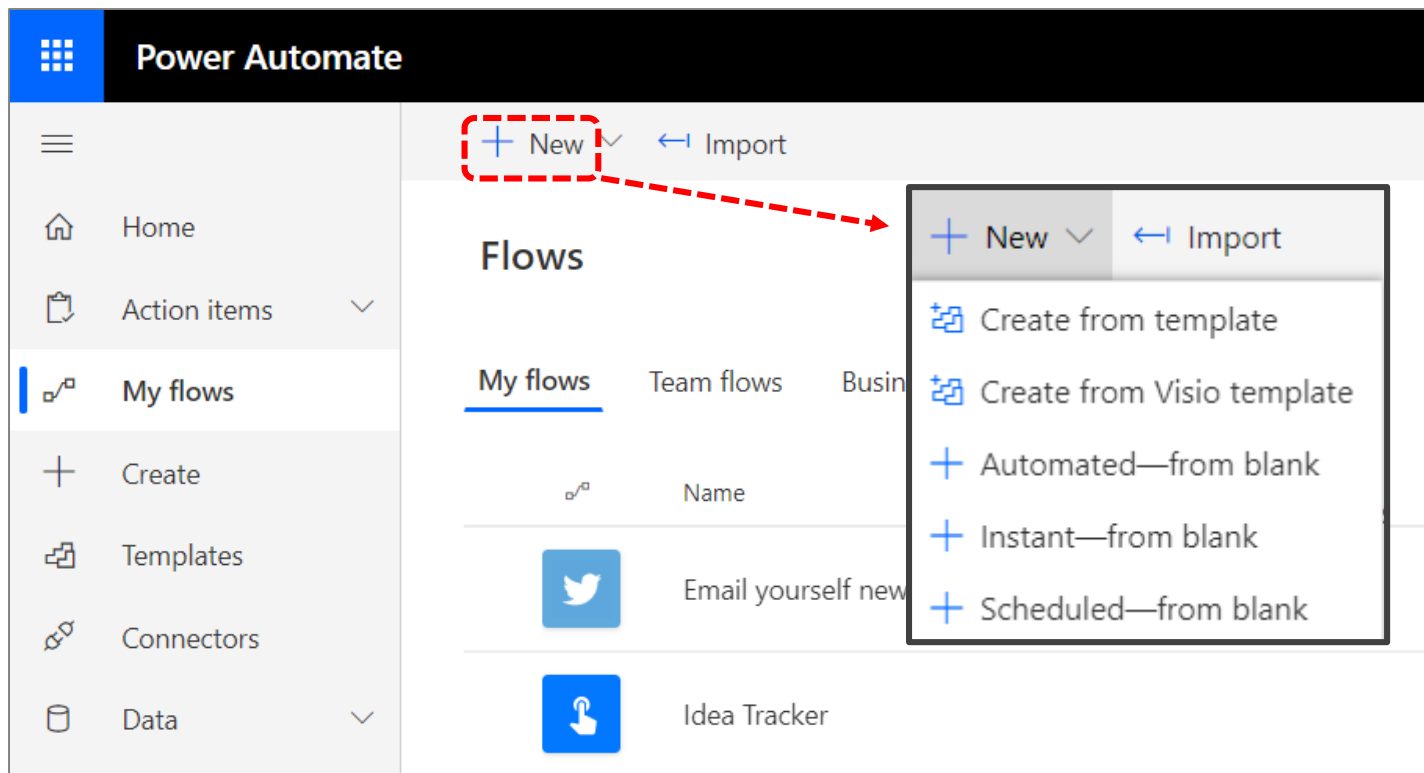
Agenda

- ✓ Power Automate Fundamentals
- Creating and Testing Flows
 - Using Control-of-Flow Actions
 - Writing Flow Expressions
 - Automating Document Generation



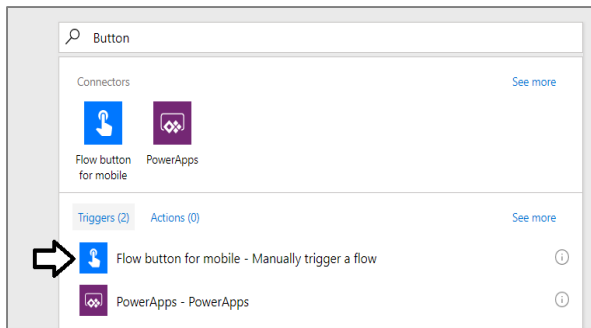
Creating and Managing Flows

- Power Automate user portal used to manage & edit flows
 - Accessible through <https://flow.microsoft.com>
 - Flow can be created from blank or from template

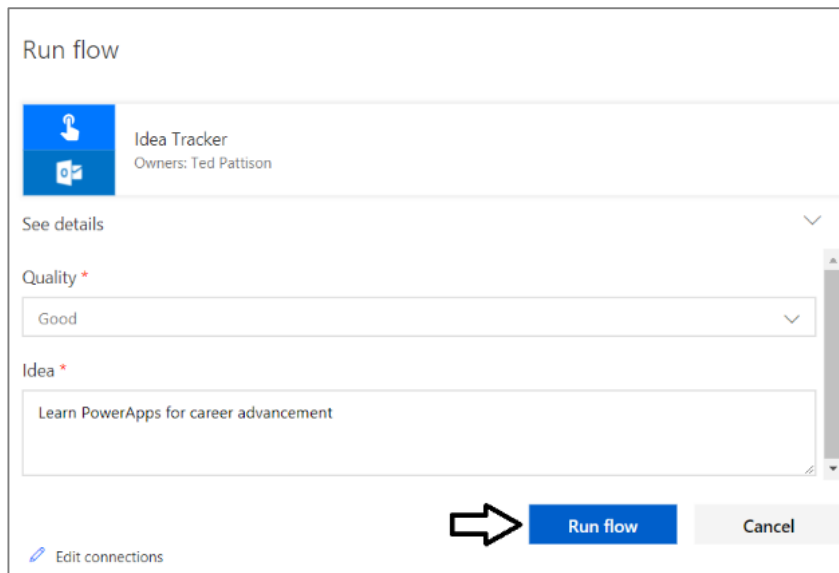


Creating a Flow Trigger by Manual Button

- Use Run button for Mobile as flow trigger



- Trigger can be extended with one or more input fields



This screenshot shows the 'Run flow' dialog box for the 'Idea Tracker' flow. The dialog box has a title bar that says 'Run flow'. Below the title bar, there is a section for the flow details, which includes the flow name 'Idea Tracker' and the owner 'Owners: Ted Pattison'. Below this, there is a 'See details' link. The dialog box also contains two input fields: 'Quality *' with a dropdown menu showing 'Good', and 'Idea *' with a text box containing 'Learn PowerApps for career advancement'. At the bottom of the dialog box, there is a blue 'Run flow' button and a grey 'Cancel' button. An arrow points to the 'Run flow' button.




Building Out The Idea Tracker Flow


Flow name

Idea Tracker

✓ Update flow

✕ Close

 Manually trigger a flow ...

 Quality ...

List Options


Lame


Good


Great


Brilliant


Enter another option












 Idea ...


 Add an input



 Send an email i ...


* To

* Subject


 Quality ✕

 Idea for consideration

* Body

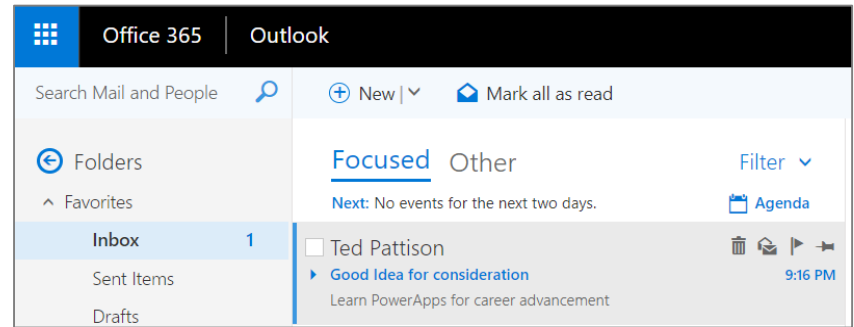
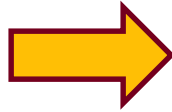
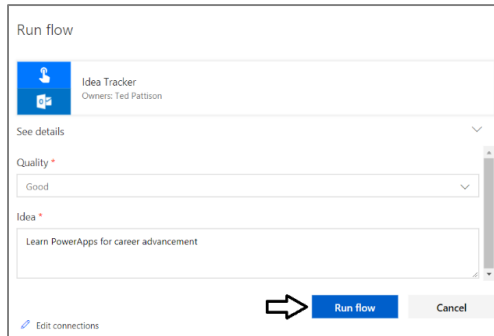
 Idea ✕

Show advanced options ▼

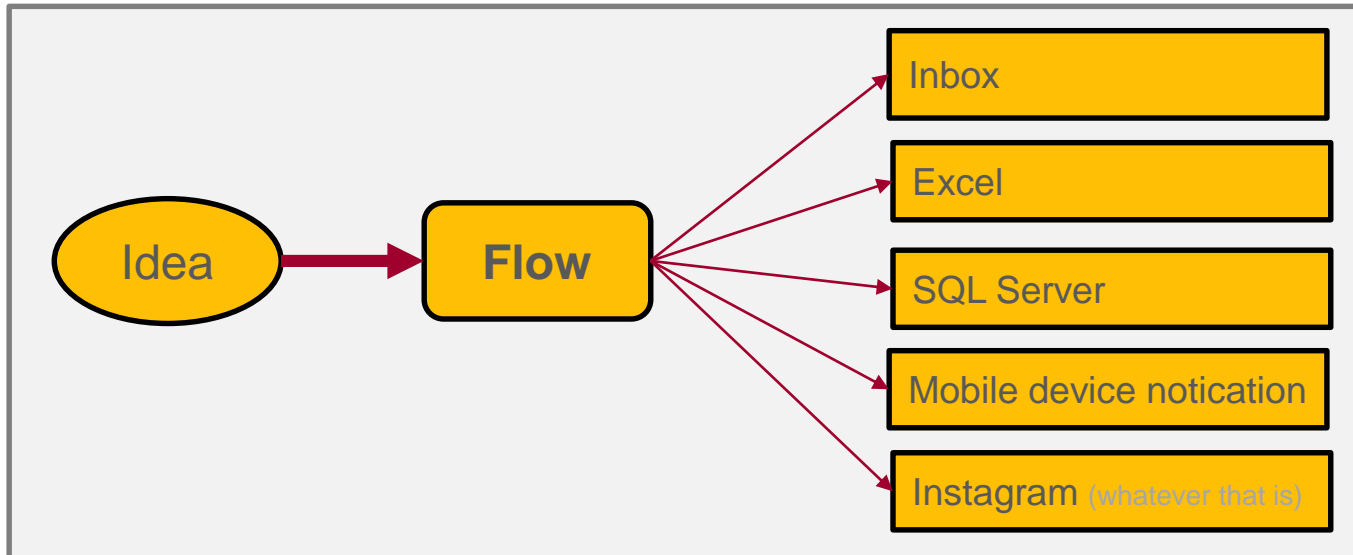


Where Should You Write the Idea?

- You can track the idea by sending an email

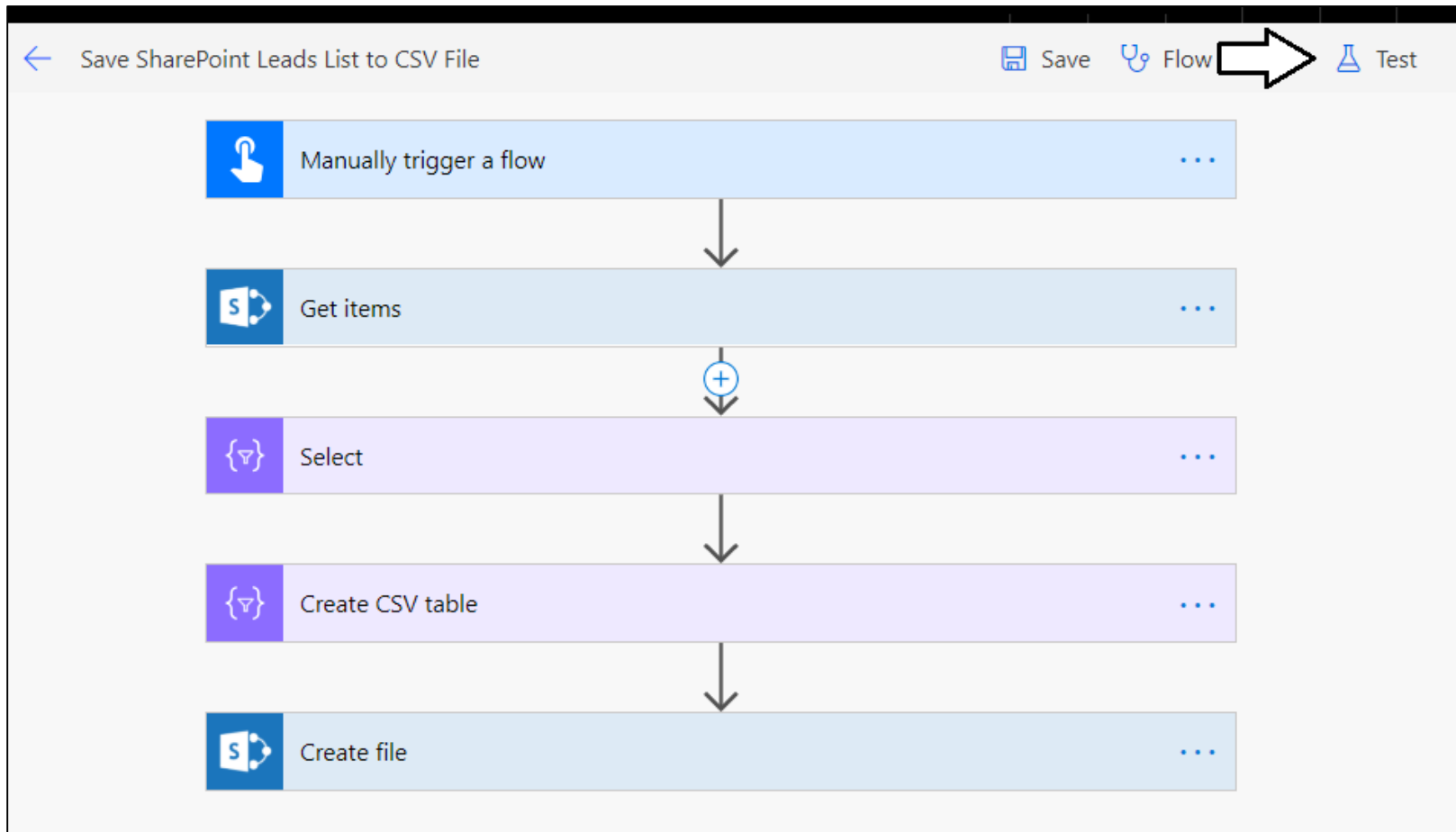


- Or get even more extravagant



Testing a Flow

- Flow can be run/tested from edit mode



Flow Checker

- Automatically checks flows for errors and omissions

The screenshot displays the Microsoft Flow Builder interface. The main workspace shows a flow titled "Send an email when a new item is created in SharePoint." with two steps:

- When a new item is created** (SharePoint connector):
 - * Site Address**: Example: `https://contoso.sharepoint.com/sites/sitename`. Below the input field is the error message: "Include a Site Address."
 - * List Name**: SharePoint list name. Below the input field is the error message: "Include a List Name."
- Send an email** (Outlook connector):
 - * To**: Specify email addresses separated by semicolons like `someone@contoso.com`. Below the input field is the error message: "Using the default values for the parameters. [Edit](#)"

A downward arrow indicates the flow sequence from the first step to the second.

The **Flow Checker** panel on the right side of the screen shows the following details:

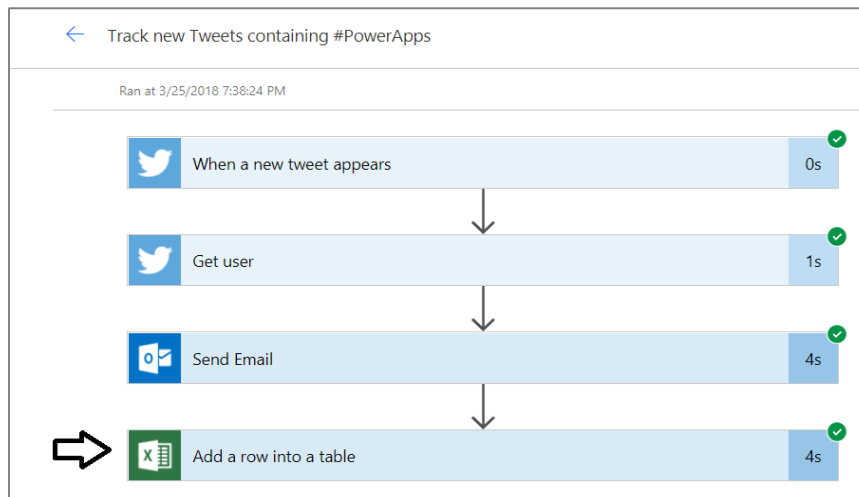
- Errors (2)** (expanded):
 - When a new item is created (2)** (expanded):
 - Include a Site Address.
 - Include a List Name.

Run History

- Flow provides history flows that have run

RUN HISTORY			
➡	✓ Succeeded	9 minutes ago	8 seconds
	✓ Succeeded	2 hours ago	0 seconds
	✓ Succeeded	3 hours ago	0 seconds

- Provides read-only view of data for auditing & monitoring





DEMO

Creating and Testing a Simple Flow

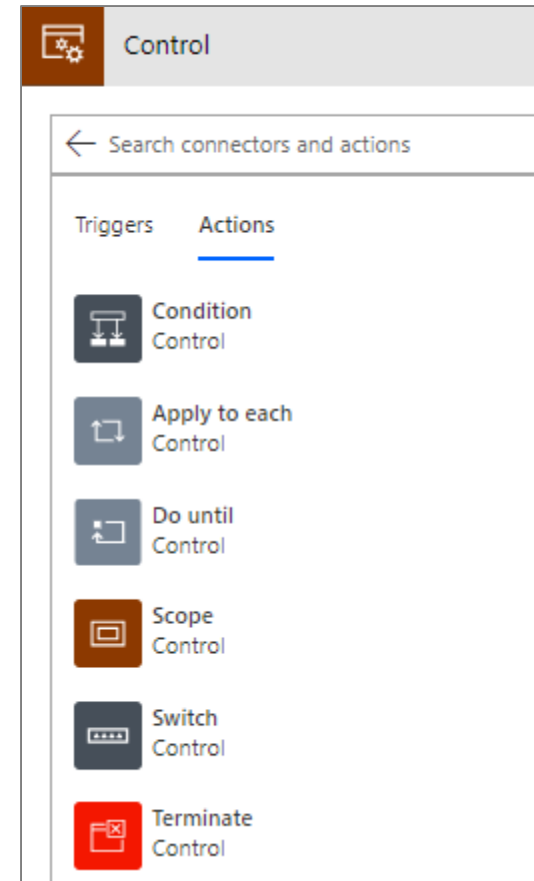
Agenda

- ✓ Power Automate Fundamentals
- ✓ Creating and Testing Flows
- Using Control-of-Flow Actions
 - Writing Flow Expressions
 - Automating Document Generation



Control of Flow

- Condition
 - Provides logical structure for If Then Else
- Apply to each
 - Enumerate through collection (e.g. list items)
- Do until
 - Repeat until condition changes
- Scope
 - Create an action container with a private scope
- Switch
 - Select Case flow
- Terminate
 - Completes a flow



Using a Condition

- Send alert email if expense amount greater than \$500
 - Condition runs test which returns true or false
 - Condition provide **If yes** branch and **If no** branch

Flow name Large Expense Alert ✓ Create flow ✕ Close

When an item is created

* Site Address

* List Name

Condition

is greater than or equal to

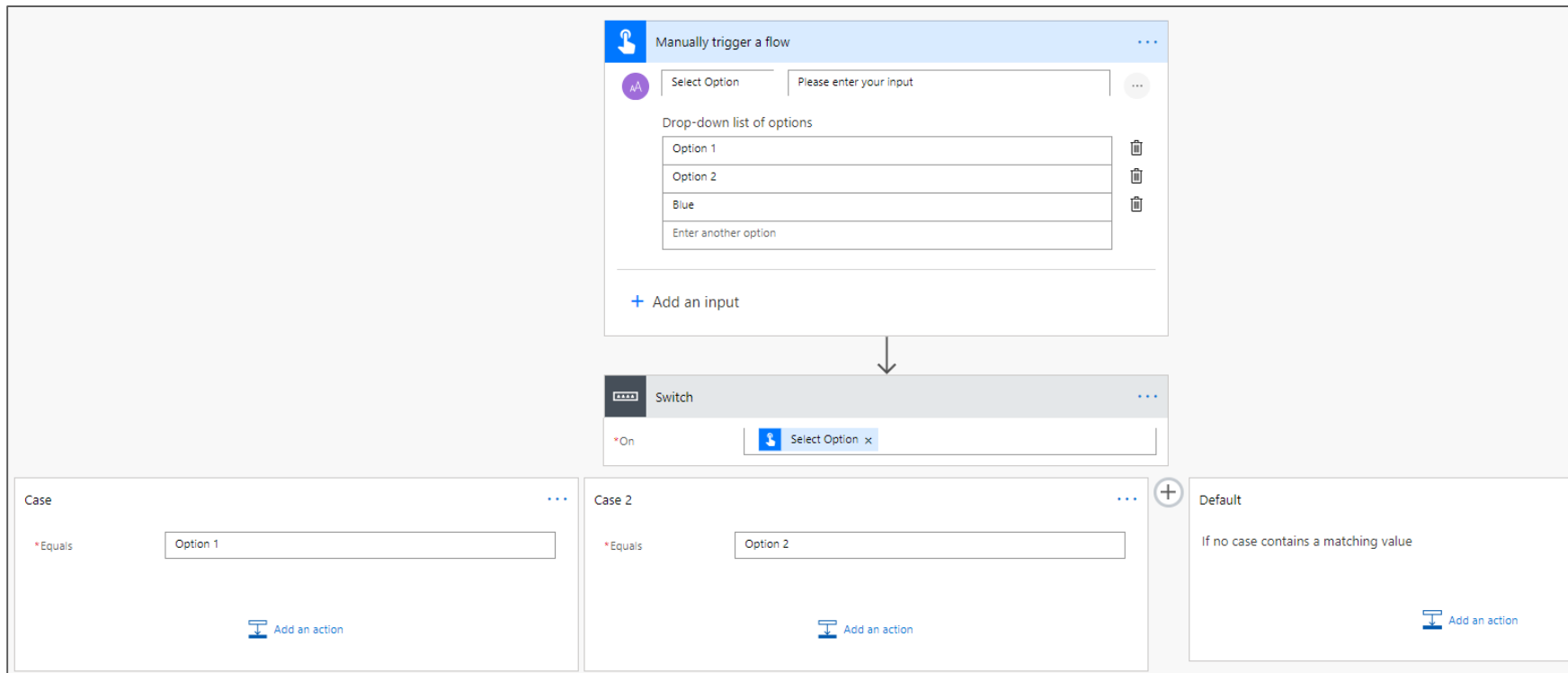
[Add dynamic content](#) [Edit in advanced mode](#) [Collapse condition](#)

✓ If yes ✕ If no



Switch Action

- Switch actions provides cases
 - Each case represents separate execution path
 - Only one execution path will execute
 - Default case executes when no other case is matched



Using Apply to Each

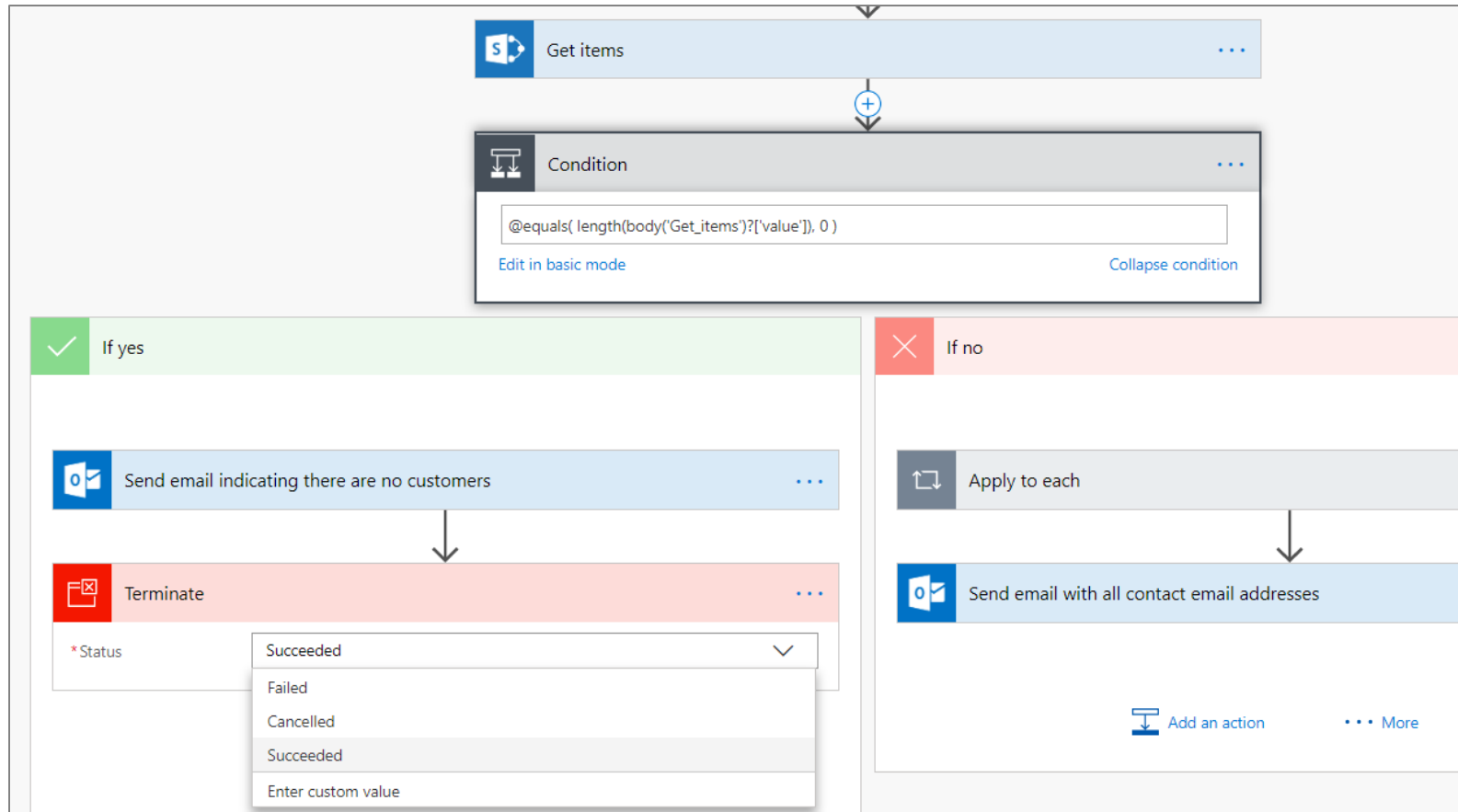
- Automatically added when list is used from output
- Destination step enumerates over list items

The screenshot displays a workflow editor interface. The first step is 'Get items', which is configured with 'Site Address' set to 'https://msd0910.sharepoint.com/' and 'List Name' set to '0769e0e8-aec5-4441-8c88-6283a6799718'. Below this step is an 'Apply to each' step, which is highlighted with a blue border. The 'Apply to each' step has a dropdown menu showing 'value x'. Below the 'Apply to each' step is a 'Delete item' step, which is configured with 'Site Address' set to 'Critical Path Training Labs Team Site - https://msd0910.sharepoint.com/', 'List Name' set to 'List1', and 'Id' set to 'ID x'. At the bottom of the editor, there are buttons for 'Add an action', 'Add a condition', and 'More'.



Terminate action

- Used to stop a flow at any point
 - Terminate status can be set to Succeeded, Cancelled, Failed



Agenda

- ✓ Power Automate Fundamentals
- ✓ Creating and Testing Flows
- ✓ Using Control-of-Flow Actions
- Writing Flow Expressions
 - Automating Document Generation



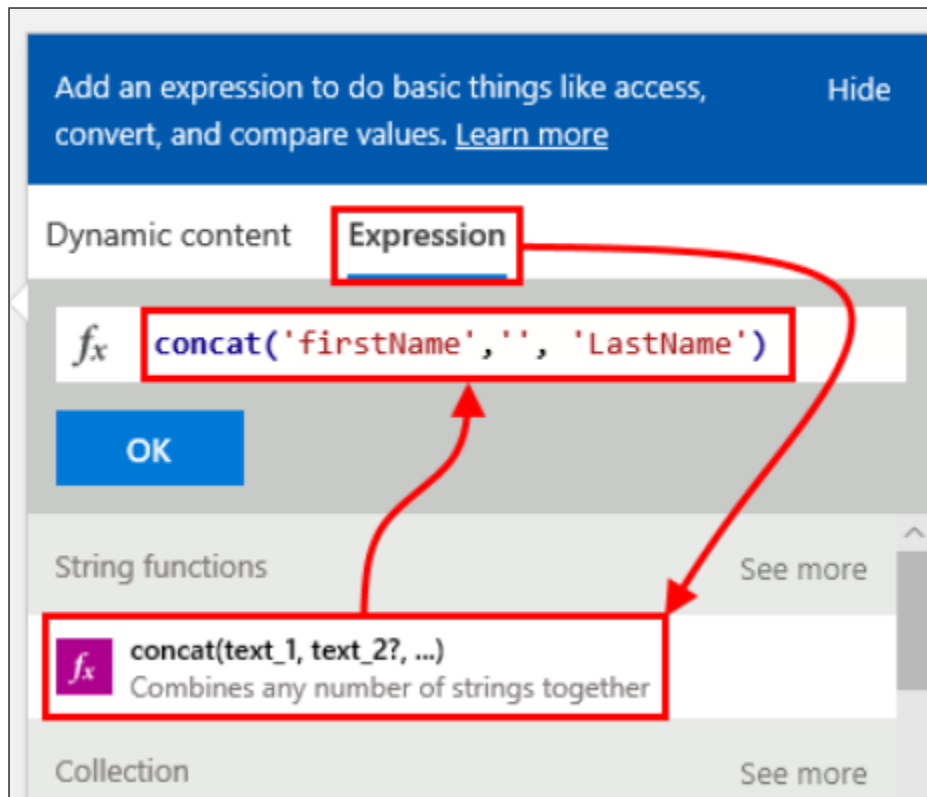
Writing Flow Expressions

- Scenarios for writing Flow expressions
 - Perform string manipulation
 - Generate a GUID or a random number
 - Convert types
 - Perform simple inline calculations
 - Handling optional values
 - Writing conditional statements using "If" statements
 - Working with arrays



Writing Expressions

- Expressions written in fx textbox
- Click OK to enter expressions



Workflow Definition Language (WDL)

- Flow expressions written in Workflow Definition Language
 - Same language used in Azure Logic Apps
 - WDL is more powerful yet more complicated than PowerApps
 - WDL does not overload operators like PowerApps does
 - WDL requires single quotes instead of double quotes

<code>fx "this is a test"</code>	Invalid: no double quotes
<code>fx 'this is a test'</code>	Valid
<code>fx 'this is a ' + 'test'</code>	Invalid : + operator not supported
<code>fx 'this is a ' & 'test'</code>	Invalid : & operator not supported
<code>fx concat('this is a ', 'test')</code>	Valid




Working with Strings

- Parse text together using **concat()**
- Parse out text using **substring()**
- Convert casing using **toLowerCase()** and **toUpperCase()**
- Search string using **indexOf** and **startsWith()**
- Create new GUID identifier using **guid()**

 **concat(text_1, text_2?, ...)**
Combines any number of strings together

 **substring(text, startIndex, length)**
Returns a subset of characters from a string

 **replace(text, oldText, newText)**
Replaces a string with a given string


 **guid()**
Generates a globally unique string (GUID)


 **toLowerCase(text)**
Converts a string to lowercase using the casing rules of the i...

 **toUpperCase(text)**
Converts a string to uppercase using the casing rules of the i...

 **indexOf(text, searchText)**
Returns the first index of a value within a string (case-insensi...

 **lastIndexOf(text, searchText)**
Returns the last index of a value within a string (case-insensit...

 **startsWith(text, searchText)**
Checks if the string starts with a value (case-insensitive, invar...

 **endsWith(text, searchText)**
Checks if the string ends with a value (case-insensitive, invari...



Performing Arithmetic Operations

- You cannot use standard arithmetic operators
 - No support for familiar operators such as **+**, **-**, *****, **/**
 - This does not work: **2 + 2**
 - This works: **add(2, 2)**

fx **min(collection or item1, item2?, ...)**
Returns the minimum value in the input array of numbers

fx **max(collection or item1, item2?, ...)**
Returns the maximum value in the input array of numbers

fx **rand(minValue, maxValue)**
Generates a random integer within the specified range (inclu...

fx **add(summand_1, summand_2)**
Returns the result from adding the two numbers

fx **sub(minuend, subtrahend)**
Returns the result from subtracting two numbers

fx **mul(multiplicand_1, multiplicand_2)**
Returns the result from multiplying the two numbers

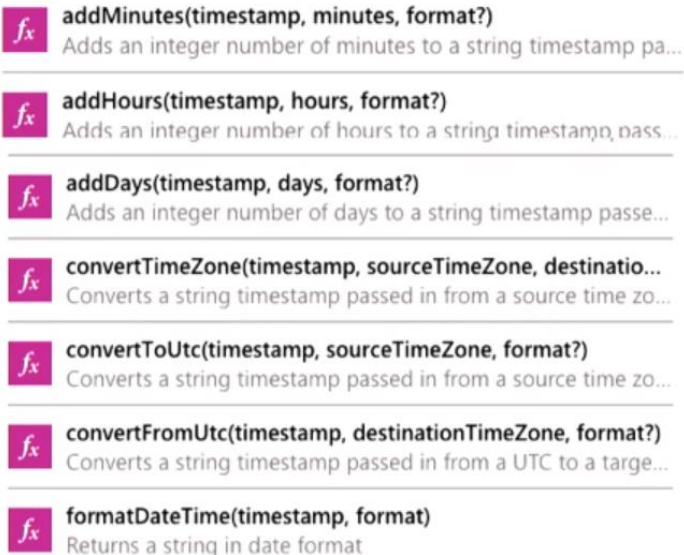
fx **div(dividend, divisor)**
Returns the result from dividing the two numbers

fx **mod(dividend, divisor)**
Returns the remainder after dividing the two numbers (mod...



Working with Dates and Time

- Get Greenwich Meantime using **utcnow()**
- Use **add*()** functions to move time back/forward
- **convertTimeZone()** used to handle local times
- **formatDateTime()** used to format



A screenshot of a code editor window displaying a list of date and time functions. Each function is preceded by a small icon of a pink square with a white 'fx' symbol. The functions listed are: **addMinutes(timestamp, minutes, format?)** (Adds an integer number of minutes to a string timestamp passed in), **addHours(timestamp, hours, format?)** (Adds an integer number of hours to a string timestamp passed in), **addDays(timestamp, days, format?)** (Adds an integer number of days to a string timestamp passed in), **convertTimeZone(timestamp, sourceTimeZone, destinationTimeZone, format?)** (Converts a string timestamp passed in from a source time zone to a destination time zone), **convertToUtc(timestamp, sourceTimeZone, format?)** (Converts a string timestamp passed in from a source time zone to UTC), **convertFromUtc(timestamp, destinationTimeZone, format?)** (Converts a string timestamp passed in from a UTC to a target time zone), and **formatDateTime(timestamp, format)** (Returns a string in date format).

fx **addMinutes(timestamp, minutes, format?)**
Adds an integer number of minutes to a string timestamp passed in

fx **addHours(timestamp, hours, format?)**
Adds an integer number of hours to a string timestamp passed in

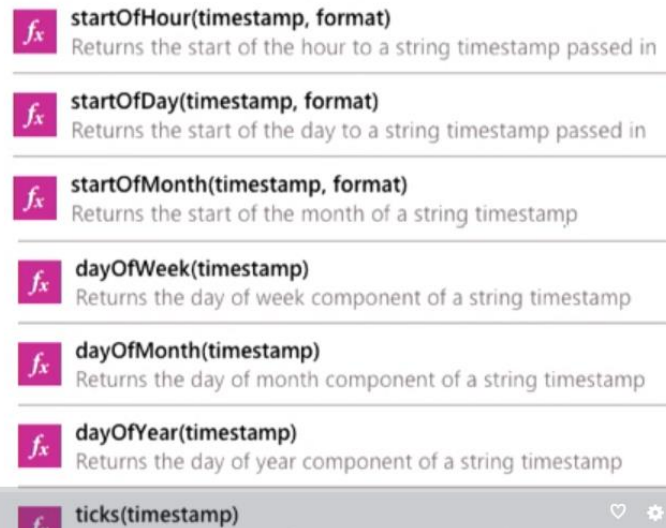
fx **addDays(timestamp, days, format?)**
Adds an integer number of days to a string timestamp passed in

fx **convertTimeZone(timestamp, sourceTimeZone, destinationTimeZone, format?)**
Converts a string timestamp passed in from a source time zone to a destination time zone

fx **convertToUtc(timestamp, sourceTimeZone, format?)**
Converts a string timestamp passed in from a source time zone to UTC

fx **convertFromUtc(timestamp, destinationTimeZone, format?)**
Converts a string timestamp passed in from a UTC to a target time zone

fx **formatDateTime(timestamp, format)**
Returns a string in date format



A screenshot of a code editor window displaying a list of date and time functions. Each function is preceded by a small icon of a pink square with a white 'fx' symbol. The functions listed are: **startOfHour(timestamp, format)** (Returns the start of the hour to a string timestamp passed in), **startOfDay(timestamp, format)** (Returns the start of the day to a string timestamp passed in), **startOfMonth(timestamp, format)** (Returns the start of the month of a string timestamp), **dayOfWeek(timestamp)** (Returns the day of week component of a string timestamp), **dayOfMonth(timestamp)** (Returns the day of month component of a string timestamp), **dayOfYear(timestamp)** (Returns the day of year component of a string timestamp), and **ticks(timestamp)**. At the bottom right of the editor, there are icons for a heart, a gear, a speaker, and a star.

fx **startOfHour(timestamp, format)**
Returns the start of the hour to a string timestamp passed in

fx **startOfDay(timestamp, format)**
Returns the start of the day to a string timestamp passed in

fx **startOfMonth(timestamp, format)**
Returns the start of the month of a string timestamp

fx **dayOfWeek(timestamp)**
Returns the day of week component of a string timestamp

fx **dayOfMonth(timestamp)**
Returns the day of month component of a string timestamp

fx **dayOfYear(timestamp)**
Returns the day of year component of a string timestamp

fx **ticks(timestamp)**



Understanding Arrays in Flow


- Flow arrays are zero-based
 - Primitive value arrays

0	Daugherty
1	Hernandez
2	Mack
3	Wiley

- Object arrays

	Last Name	First Name	Company	Business Phone	Home Phone
0	Daugherty	Cindy	Wonka Industries	1(337)111-4444	1(337)111-7777
1	Hernandez	Zane	Vandelay Industries	1(757)666-3333	1(757)777-1111
2	Mack	Chang	Wonka Industries	1(480)111-4444	1(480)777-0000
3	Wiley	Ramona	Ecumena	1(201)777-8888	1(201)777-2222

Accessing an Array using ['value']

 Get items ⓘ ⋮

*Site Address ✕

*List Name ✕

Show advanced options ▾



`body('Get_items')?['value']`

	Last Name	First Name	Company	Business Phone	Home Phone
0	Daugherty	Cindy	Wonka Industries	1(337)111-4444	1(337)111-7777
1	Hernandez	Zane	Vandelay Industries	1(757)666-3333	1(757)777-1111
2	Mack	Chang	Wonka Industries	1(480)111-4444	1(480)777-0000
3	Wiley	Ramona	Ecumena	1(201)777-8888	1(201)777-2222



Retrieving List Items

- Use **first()** and **last()** to get lead at head or tail
- Individual items retrieved using zero-based array syntax
 - SharePoint list item array - `body('Get_items')?['value']`
 - First list item in array - `body('Get_items')?['value'][0]`
 - Field value of first item - `body('Get_items')?['value'][0]['ID']`

The screenshot shows a workflow step titled 'Delete item'. It has three input fields: '* Site Address' with the value 'Critical Path Training Labs Team Site - https://msd0910.sharepoint.com/', '* List Name' with the value 'List1', and '* Id' with a dynamic content expression `body('Get_items')?['value'][0]['ID']`. The 'Id' field is highlighted with a blue border. There is an 'Add dynamic content' button at the bottom right of the form.

The screenshot shows the 'Expression' tab of a dynamic content editor. The expression `body('Get_items')?['value'][0]['ID']` is entered in the text box. Below the text box is a blue 'Update' button. The editor has tabs for 'Dynamic content' and 'Expression', with 'Expression' being the active tab.



Writing Expressions with item()

- You can access current item inside flow loops
 - Use `item()` function provided by WDL
 - Example below formats all company names in upper case
 - Requires using `toUpper(item()['Company'])` in Apply to each loop

The screenshot displays a Power Automate flow with the following steps:

- Get items**: The first step in the flow.
- Apply to each**: A loop step that iterates over the items returned by the 'Get items' step. The 'Select an output from previous steps' dropdown is set to 'value'.
- Update item**: An action within the 'Apply to each' loop. It is configured with the following fields:
 - Site Address**: Communication site - <https://summatime.sharepoint.com/>
 - List Name**: Customers
 - ID**: ID
 - Last Name**: Last Name
 - Company**: `toUpper(item()['Company'])`
 - First Name**: (Empty field)
 - E-Mail**: (Empty field)

The 'Update item' action's 'Company' field is highlighted with a pink background, indicating the use of a dynamic content expression. The expression `toUpper(item()['Company'])` is shown in the 'Dynamic content' pane on the right, which also includes a link to 'Learn more' and an 'Update' button.



The background of the slide is a close-up, low-angle shot of a server rack. The server units are illuminated with bright blue light, creating a strong sense of depth and perspective. The lights are arranged in a grid-like pattern, with some units appearing more brightly lit than others. The overall color scheme is dominated by blue and black, giving it a high-tech, digital feel.

DEMO

Enumerating through SharePoint List Items to Perform Cleanup Operations

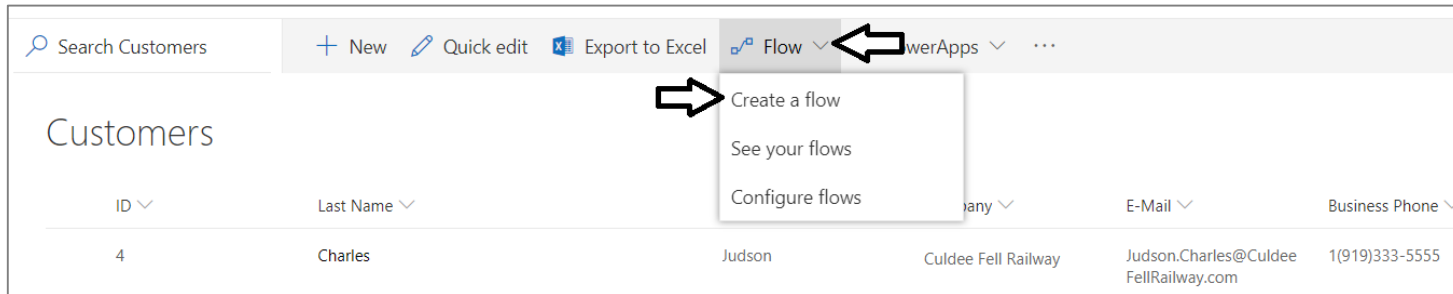
Agenda

- ✓ Power Automate Fundamentals
- ✓ Creating and Testing Flows
- ✓ Using Control-of-Flow Actions
- ✓ Writing Flow Expressions
- Automating Document Generation

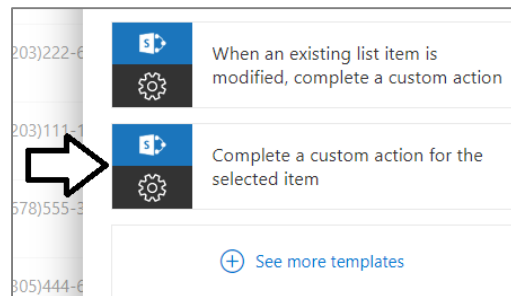
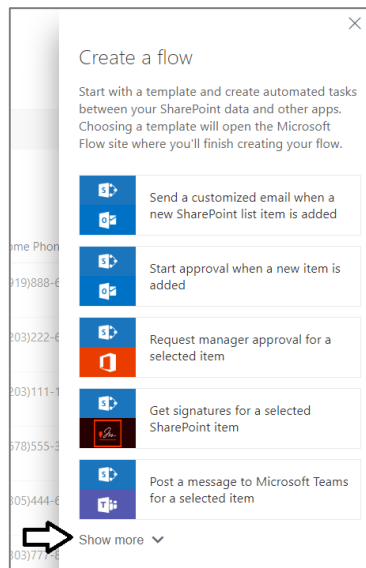


Adding Flows for SharePoint Selected Item

- You can create a flow for a specific SharePoint list



- Select option to **Complete a custom action for the selected item**



Custom Action for Selected Item Flow

- Flow is initially created with a trigger and an action
 - Flow created with a **For a selected item** trigger
 - Flow also contain **Get item** action to get list item field values

The screenshot displays the Microsoft Flow configuration interface. At the top, a navigation bar shows a back arrow and the text 'Complete a custom action for the selected item'. Below this, the 'For a selected item' trigger is configured with the following fields:

- * Site Address:
- * List Name:
- Show advanced options: ☐

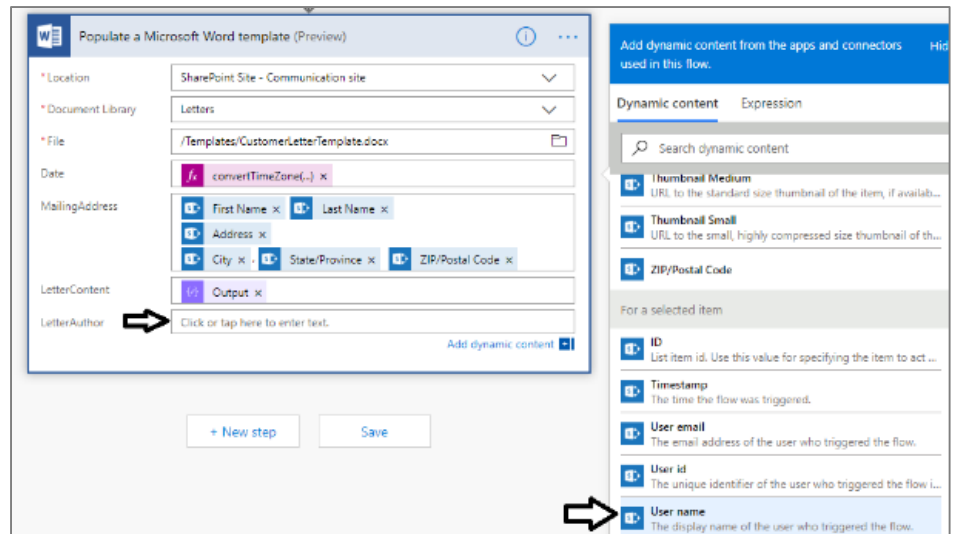
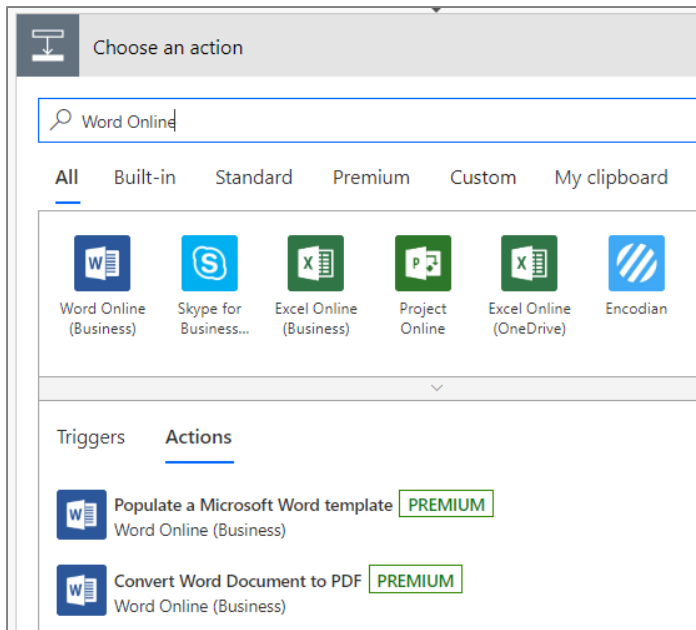
An arrow points down to the 'Get item' action, which is also configured with the following fields:

- * Site Address:
- * List Name:
- * Id:
- Show advanced options: ☐



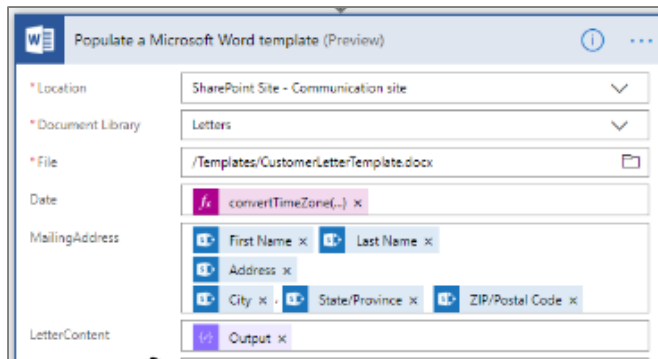
Word Online Connector

- Word Online connectors provides two valuable actions
 - Populate a Microsoft Word template
 - Convert Word Document to PDF

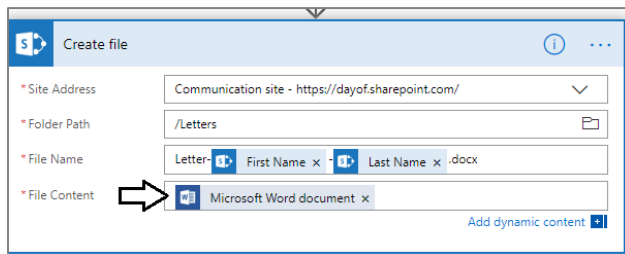


Populate a Microsoft Word template

- Using Populate a Microsoft Word template action
 - Select Word document template with named content controls
 - Dynamically add content into named content controls

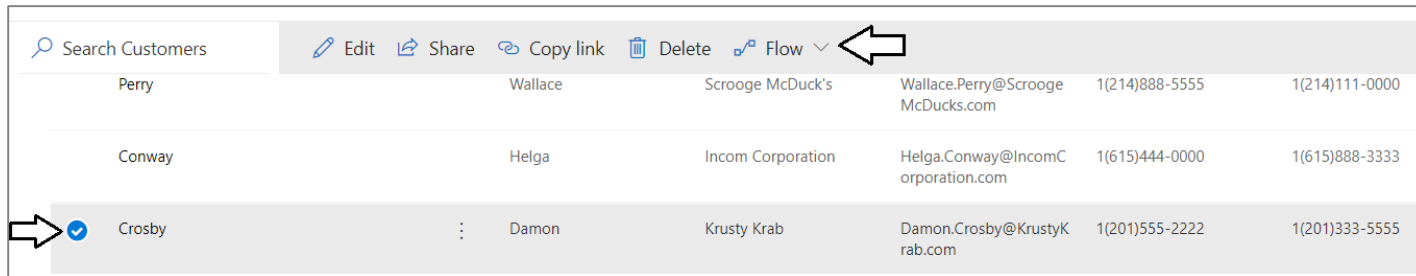


- Populate a Microsoft Word template doesn't create file
 - You must add another action to actually create the DOCX file



Running a For a Selected Item Flow

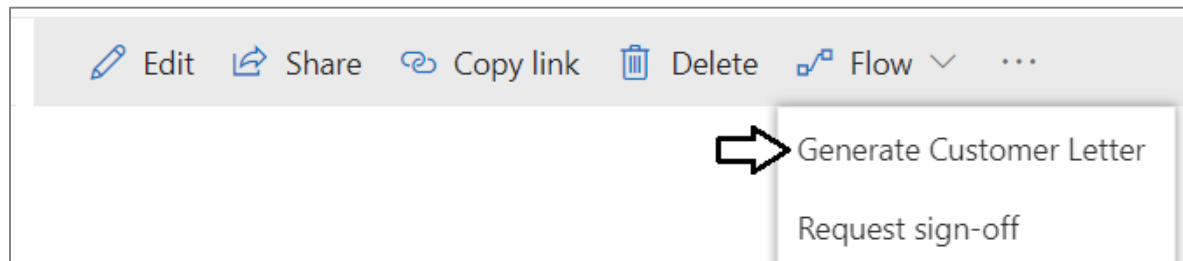
- For a selected item flow runs on one item at a time
 - Make sure just one item is selected then drop down Flow menu



The screenshot shows a table with customer information. The 'Flow' menu is open, and the 'Crosby' row is selected. An arrow points to the 'Flow' menu, and another arrow points to the 'Crosby' row.

Search Customers		Edit	Share	Copy link	Delete	Flow
Perry	Wallace	Scrooge McDuck's	Wallace.Perry@Scrooge McDucks.com	1(214)888-5555	1(214)111-0000	
Conway	Helga	Incom Corporation	Helga.Conway@IncomC corporation.com	1(615)444-0000	1(615)888-3333	
<input checked="" type="checkbox"/> Crosby	Damon	Krusty Krab	Damon.Crosby@KrustyK rab.com	1(201)555-2222	1(201)333-5555	

- For a select item flows should appear in dropdown menu





DEMO

Generating DOCX Files with the Word Online Connector

Summary

- ✓ Power Automate Fundamentals
- ✓ Creating and Testing Flows
- ✓ Using Control-of-Flow Actions
- ✓ Writing Flow Expressions
- ✓ Automating Document Generation

