

Introduction to Microsoft Flow



Agenda

- Microsoft Flow Fundamentals
- Creating and Testing Flows
- Using Control of Flow Actions
- Writing Flow Expressions
- Processing Data and Preparing Content



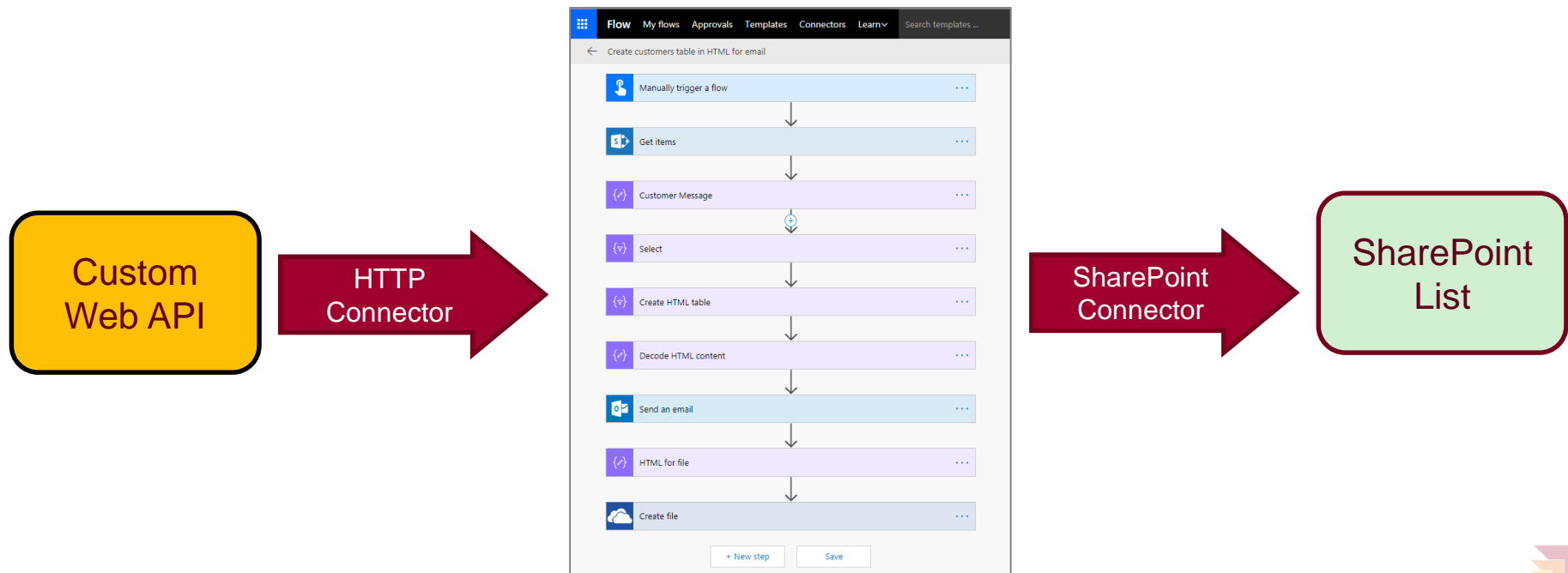
What is Flow?

- Service for automating workflows across other services
 - Designed by Microsoft for business users more than developers
- What can you do with Flow?
 - Get notifications
 - Copy files
 - Collect data
 - Automate approvals



Building Blocks of Flow

- **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



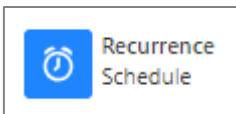
Examples Services to Use with Flow

- SharePoint Online
 - Trigger a flow when a new item is added
 - Trigger a flow when a new document is uploaded
 - Add an action to create or update a list item
- Twitter
 - Trigger a flow when a tweet contains a specific hashtag
 - Track tweet by sending email or write to SharePoint list
 - Add an action to reply to a tweet

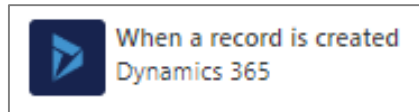
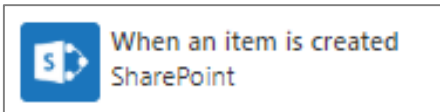


Flow Trigger Types

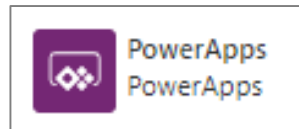
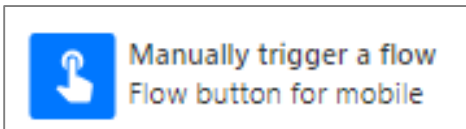
- Scheduled Flow Triggers
 - Runs periodically based on an interval



- Automated Flow Triggers
 - Runs when something happens

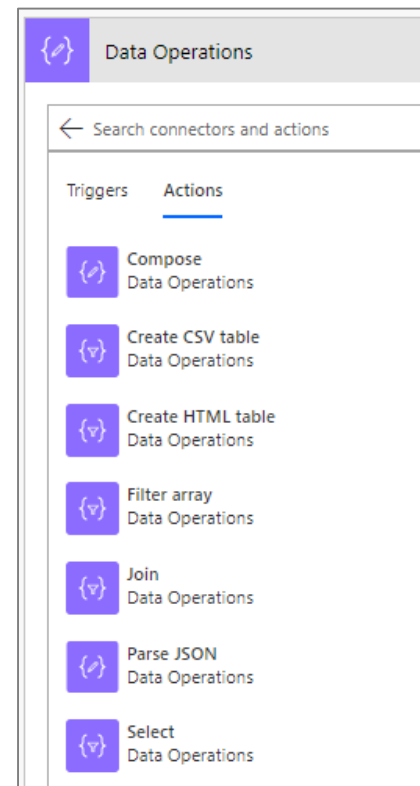
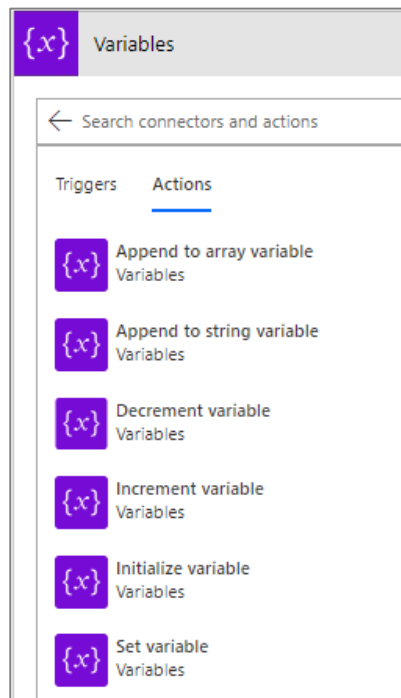
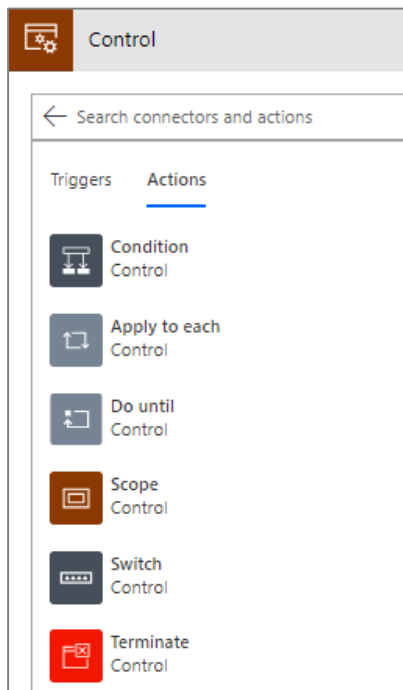


- On-demand Flow Triggers
 - Runs when a user clicks a button



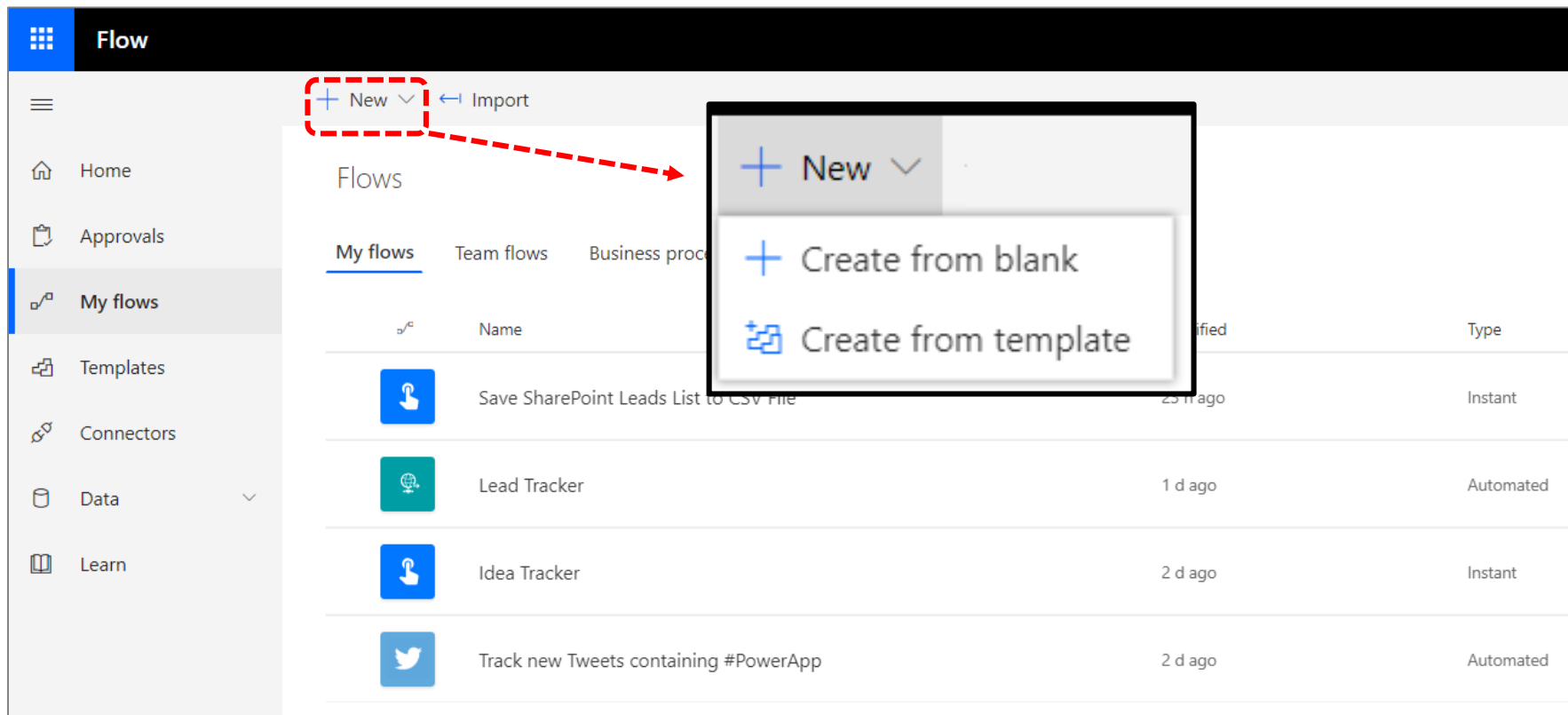
Core Action Categories

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



Creating and Managing Flows

- Flow user portal allows users to manage and edit flows
 - Accessible through <https://flow.microsoft.com>
 - Flow can be created from blank or from template

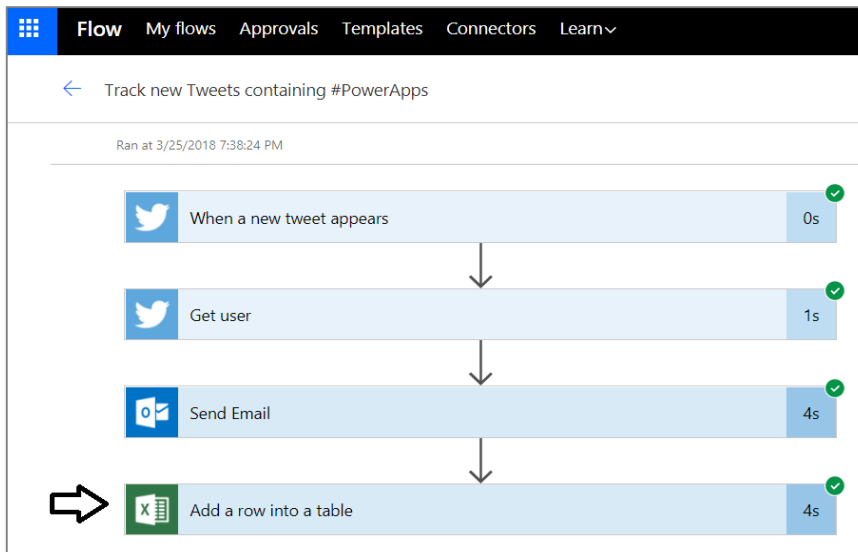


Run History

- 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



The background of the slide is a close-up, low-angle shot of a server rack. The rack is filled with numerous server units, each featuring a grid of small, glowing blue lights. The perspective is looking up the length of the rack, creating a sense of depth and scale. The lighting is predominantly blue, giving it a high-tech, digital feel.

DEMO

Demo 1

Creating and Testing a Simple Flow

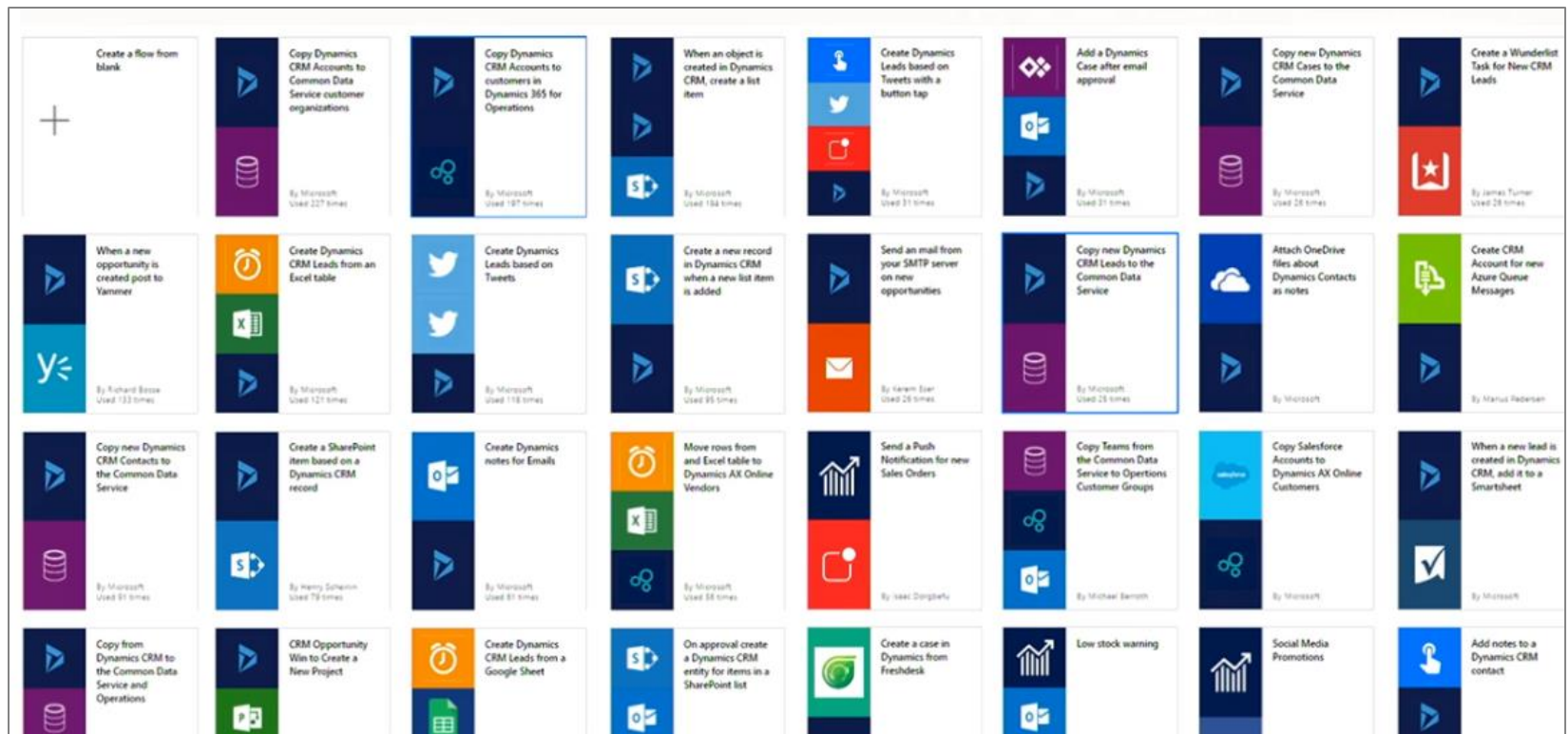
Agenda

- ✓ Microsoft Flow Fundamentals
- Creating and Testing Flows
 - Using Control of Flow Actions
 - Writing Flow Expressions
 - Processing Data and Preparing Content



Flow Templates

- There are 100s of templates
 - Flow is built with pre-configured trigger and actions

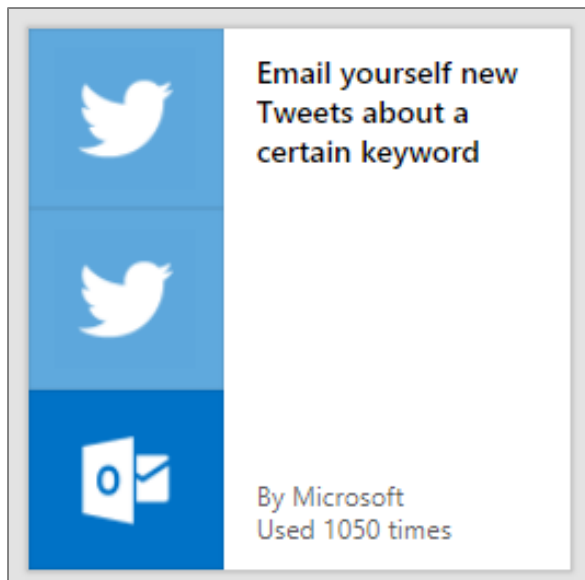


Creating a Flow from a Template

- Use Create from template link

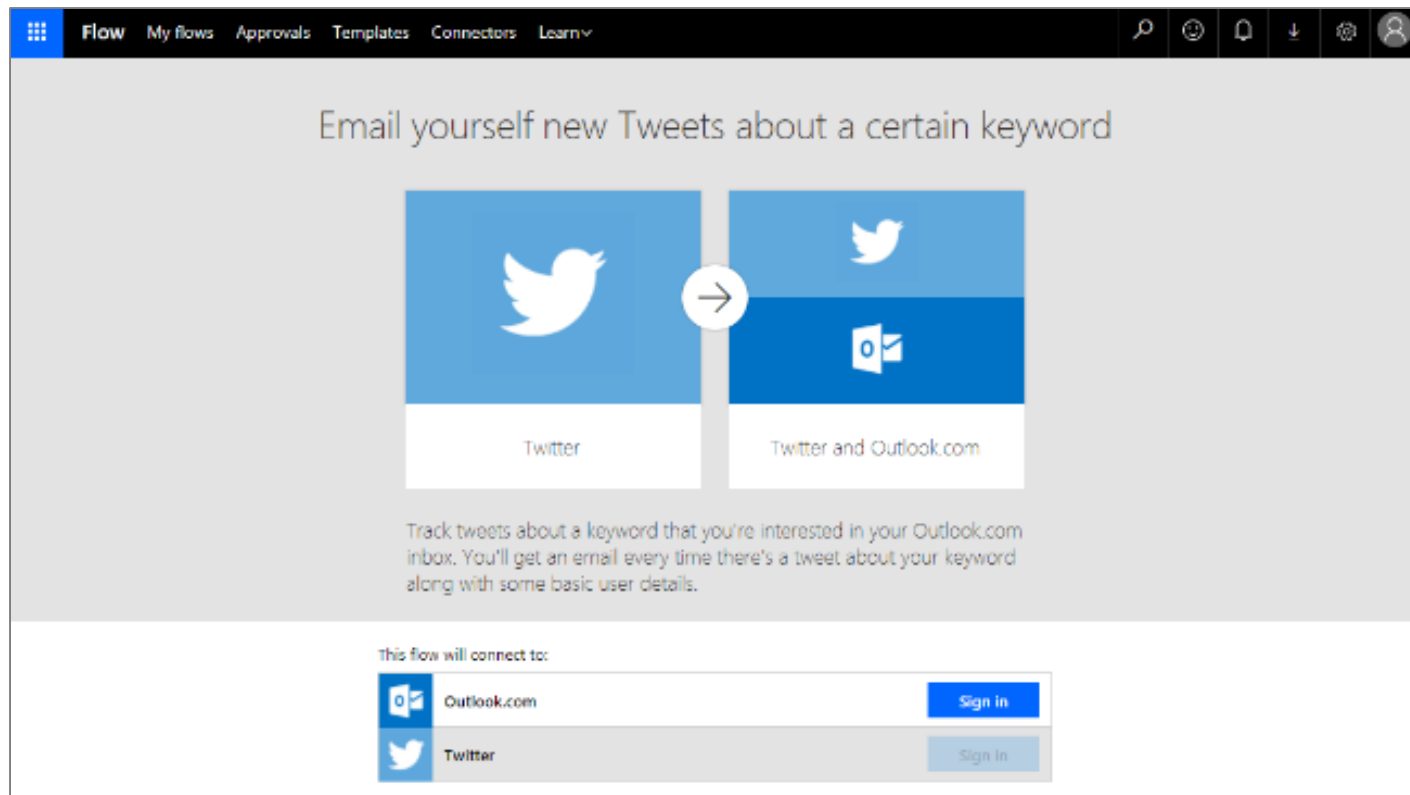


- Select a template




Many Flow Templates Require Connections


- This allows a flow to read and write data
 - Flow often requires user to authorize permissions



Authorizing an App to Connect to a Service

- User must authorize app to use service

 Microsoft

student@ddpaf.onmicrosoft.com 

PowerApps and Flow

Publisher's website: msmanaged-na.consent.azure-apim.net

This app would like to:

- ✓ Read and write access to your mail
- ✓ Send mail as you
- ✓ Have full access to your calendars
- ✓ Have full access of your contacts
- ✓ Access your data anytime

You should only accept if you trust the publisher and if you selected this app from a store or website you trust. Ask your admin if you're not sure.

Cancel Accept

Authorize Microsoft PowerApps and Flow to use your account?



Microsoft PowerApps and Flow
By Microsoft
www.powerapps.com

Microsoft PowerApps is a service for building and using custom business apps that connect to your data and work across the web and mobile - without the time and expense of custom software development.

☒ Remember me · [Forgot password?](#)

Authorize app Cancel

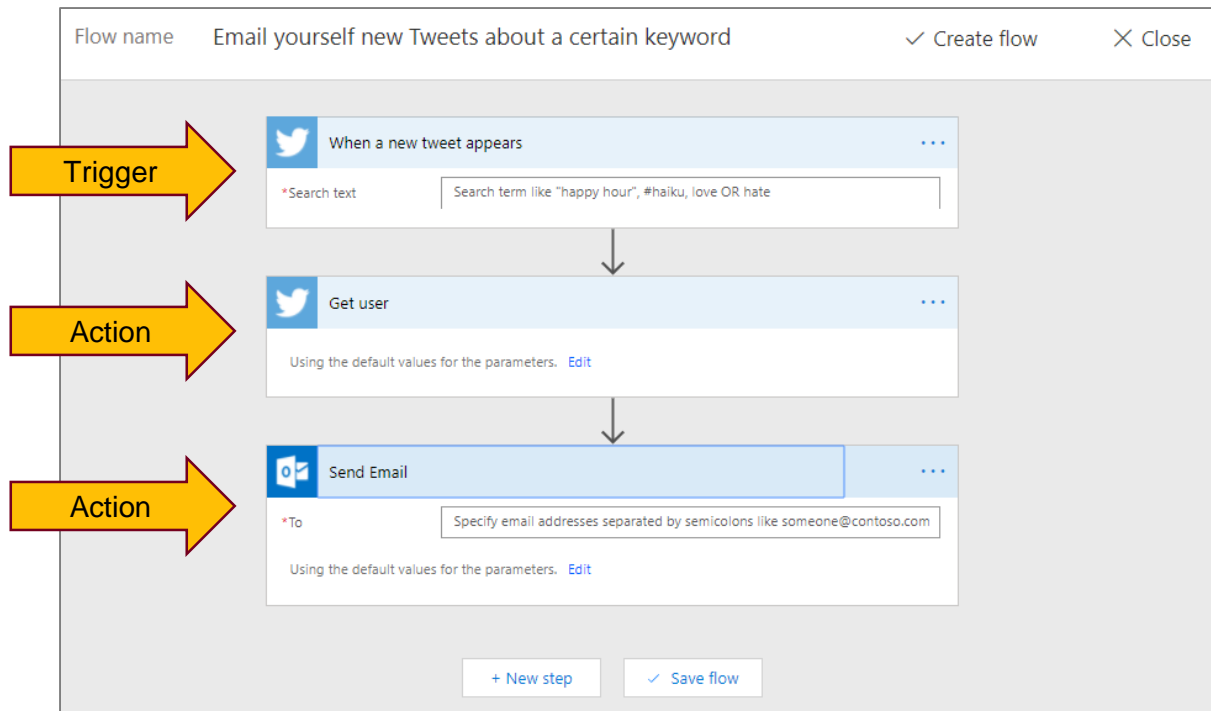
This application will be able to:

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.



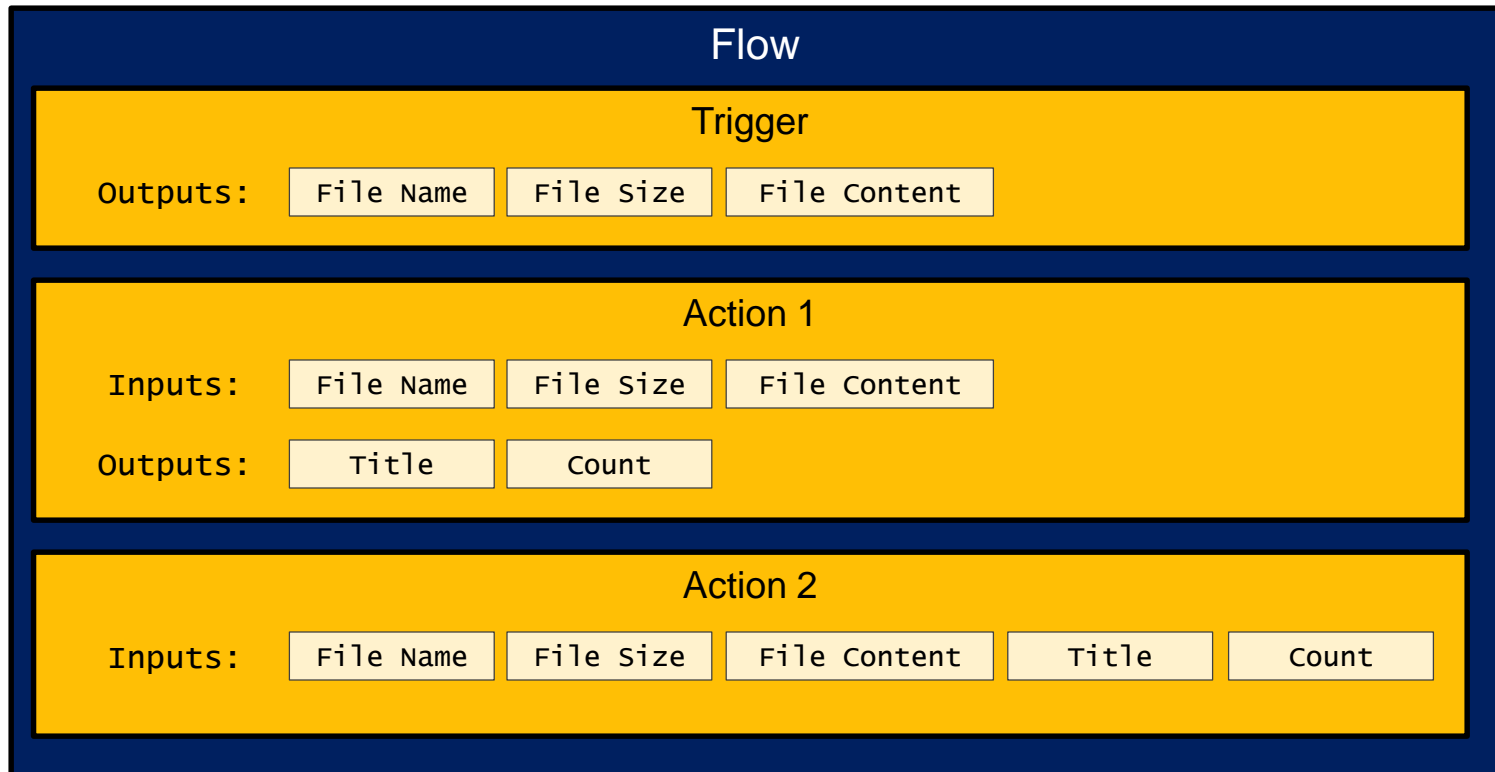
Working with the Flow Designer

- Flow Designer provides UI for building flows
 - Somewhat similar to SharePoint Designer workflow designer
 - You build flows by adding and configuring steps
 - There are 3 types of steps: triggers, actions or conditions



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



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](#)"

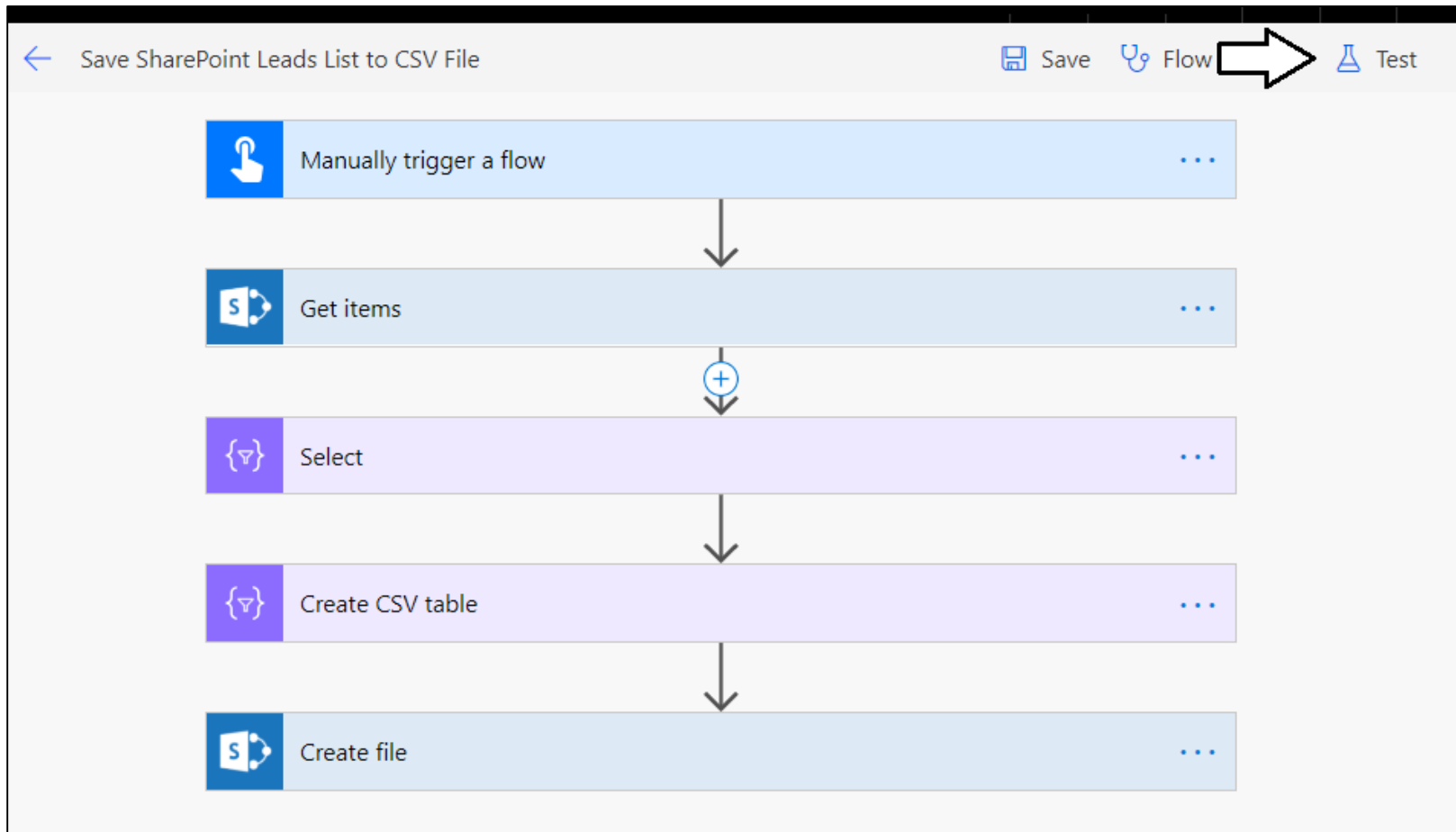
A downward arrow indicates the flow sequence from the first step to the second.

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

- Errors (2)**: A collapsed section header.
- When a new item is created (2)**: A collapsed section header for the first step.
- Include a Site Address.**: An error message associated with the Site Address parameter.
- Include a List Name.**: An error message associated with the List Name parameter.

Testing a Flow

- Flow can be run/tested from edit mode



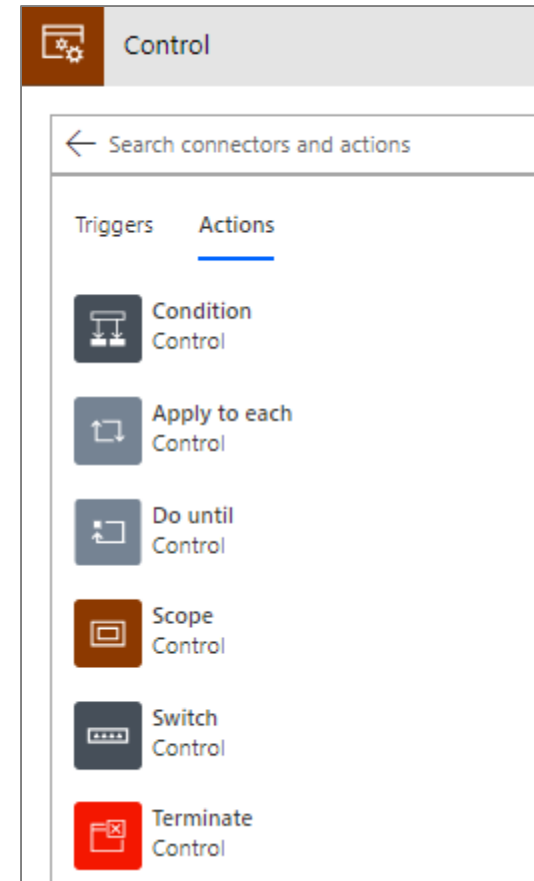
Agenda

- ✓ Microsoft Flow Fundamentals
- ✓ Creating and Testing Flows
- Using Control of Flow Actions
 - Writing Flow Expressions
 - Processing Data and Preparing Content

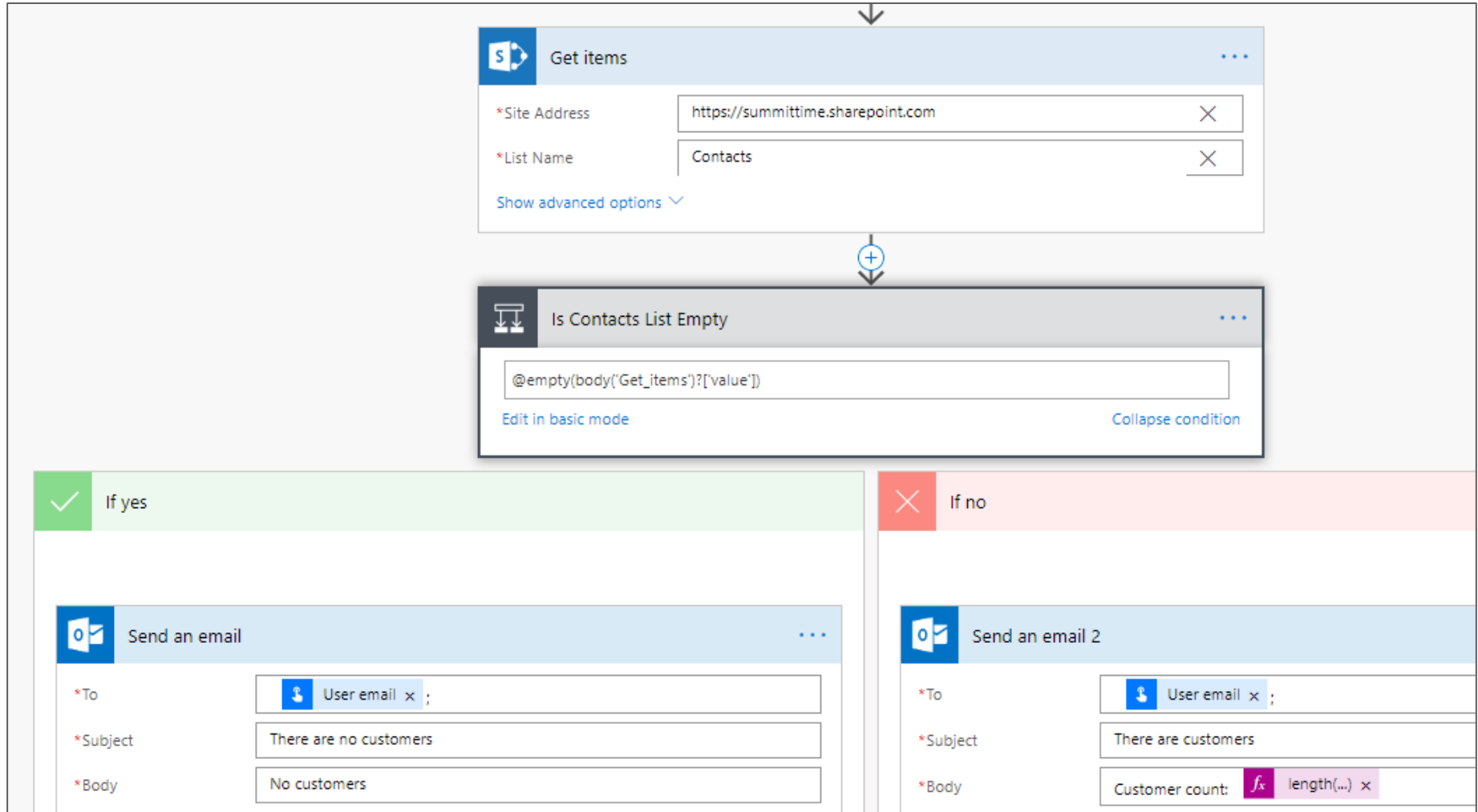


Control of Flow

- 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

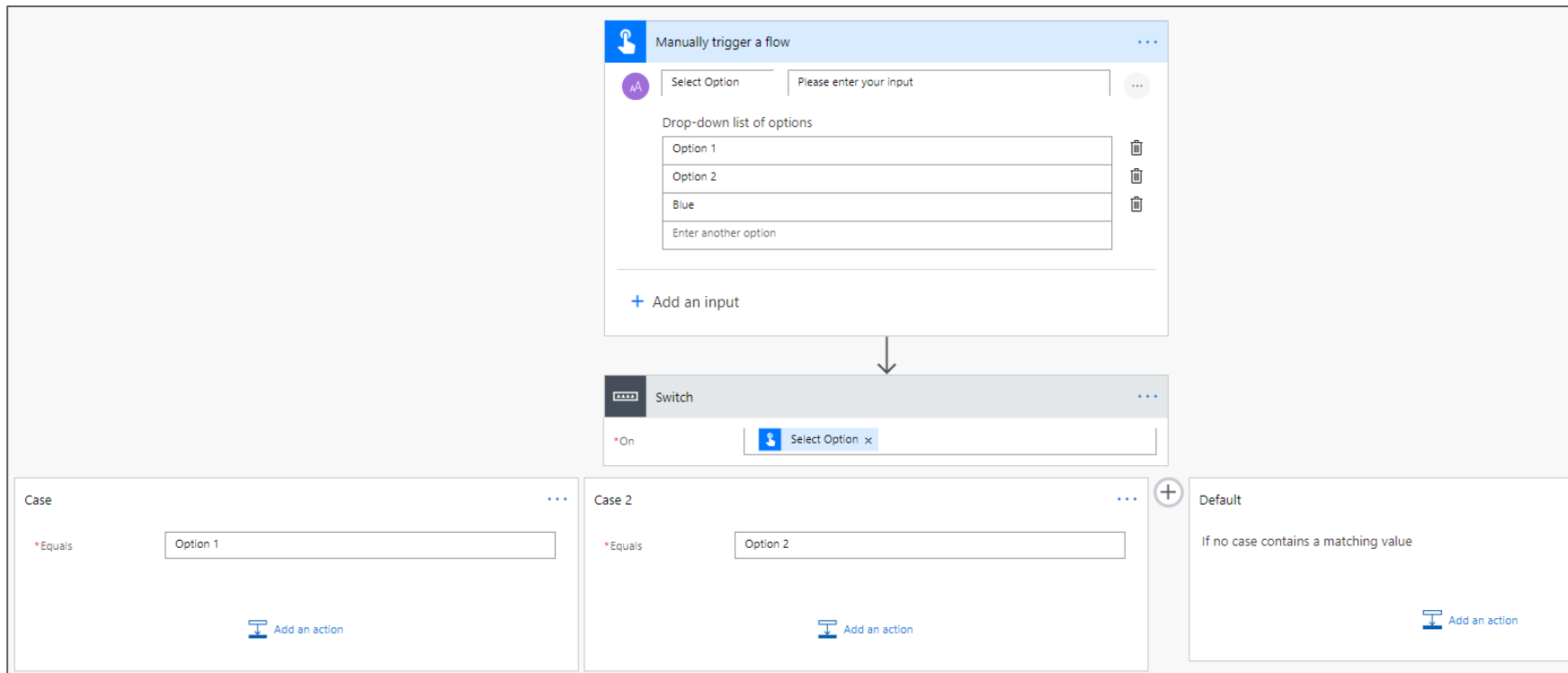


Condition Action



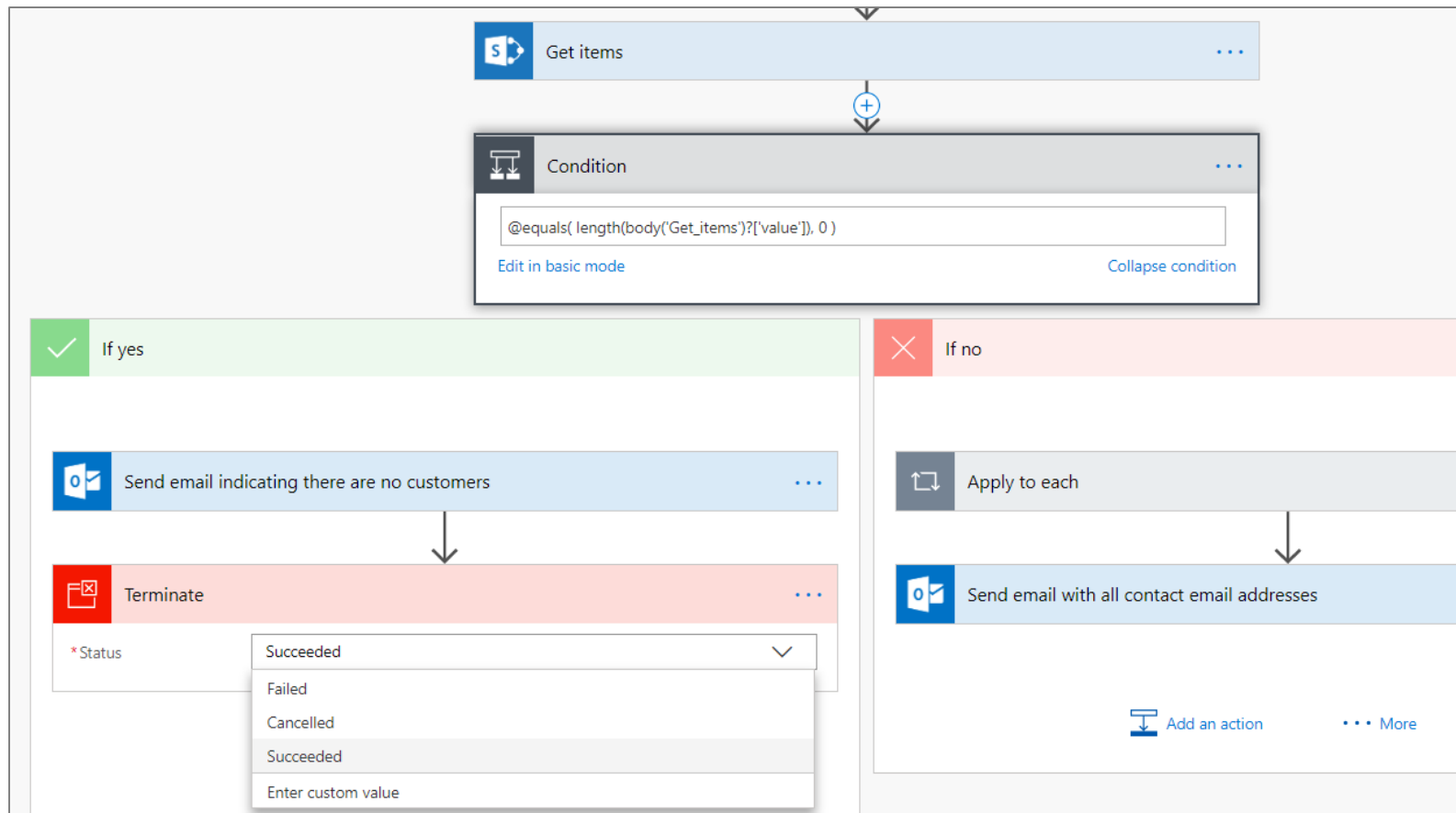
Switch Action

- 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



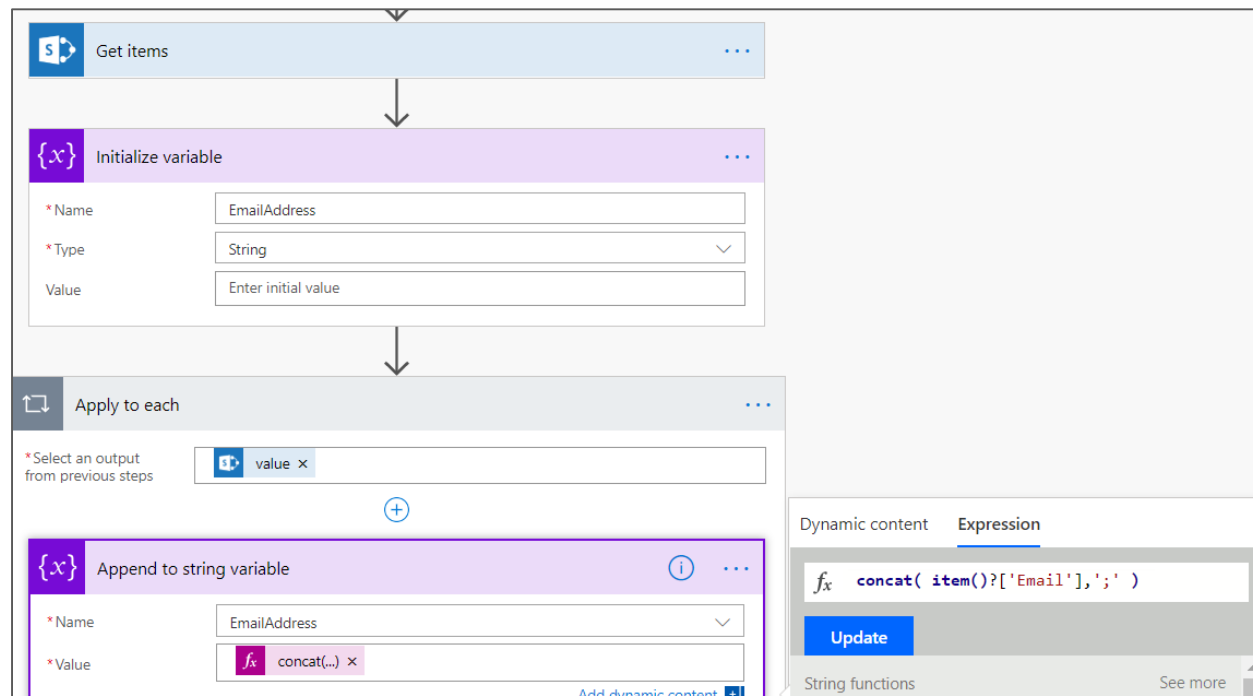
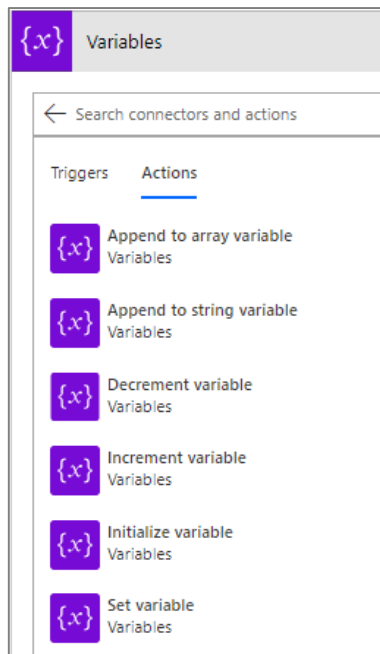
Terminate action

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

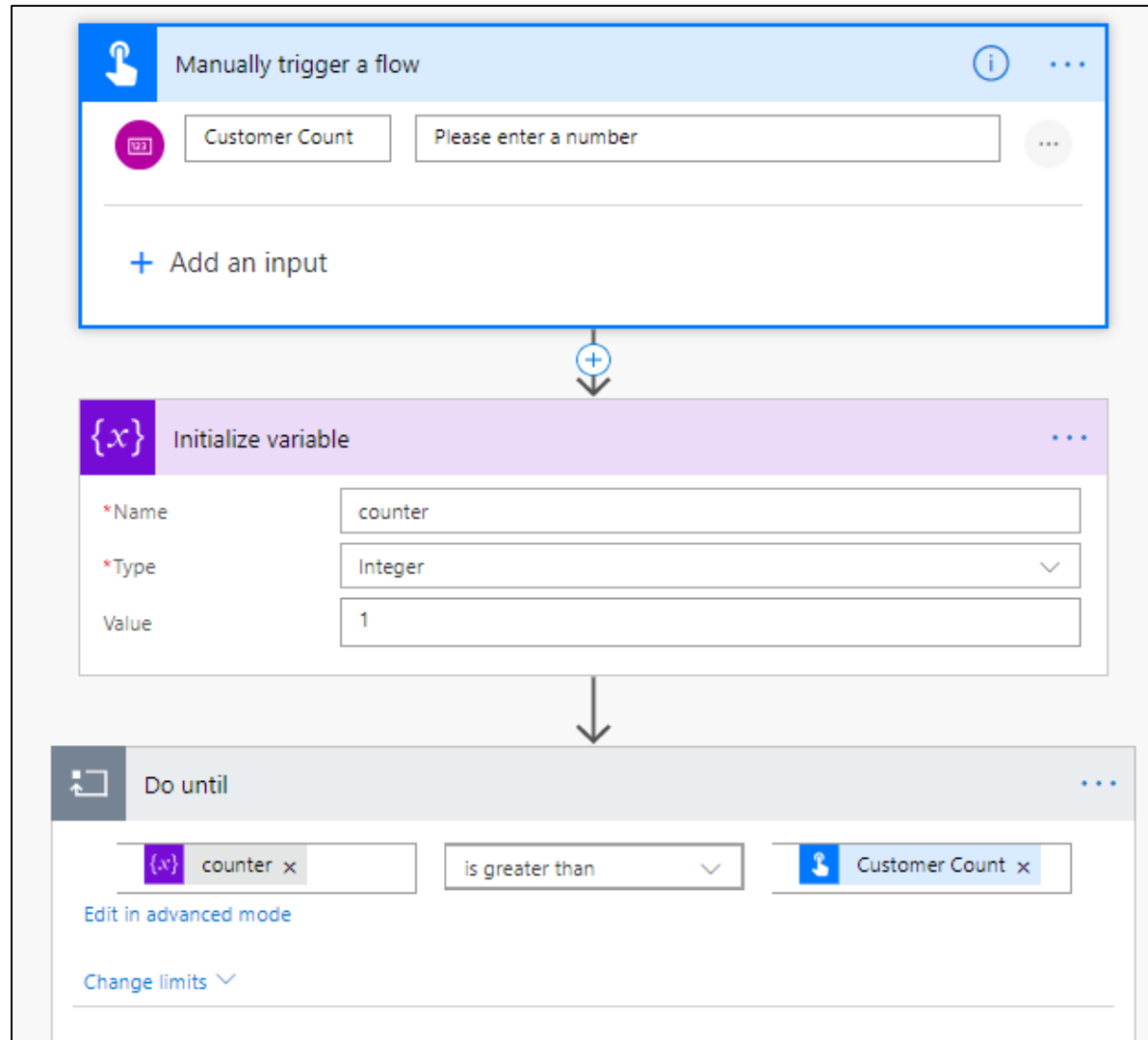


Tracking State using Variables

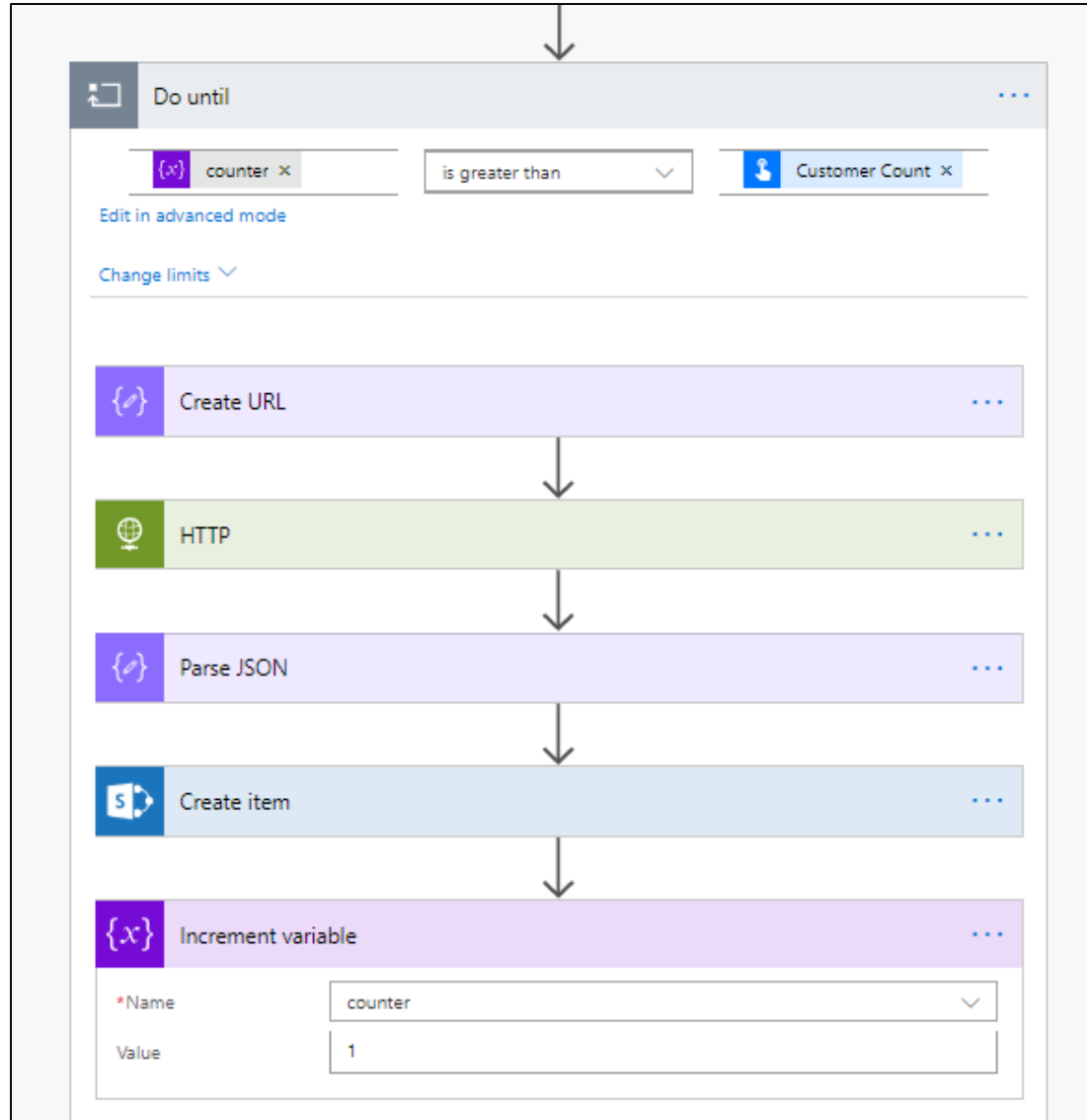
- Variables used to track state during flow lifetime
 - Initialize Variable used to create variable with Type and Value
 - Other variable actions uses to update variable values
 - By default, variable stored within flow until end of flow lifetime
 - Variables can be initialized inside Scope action to reduce lifetime



Do Until Action with Counter Variable



Executing Operations inside Do Until Loop



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, 'Get items', is highlighted with a blue border. It contains two dropdown menus: 'Site Address' with the value 'https://msd0910.sharepoint.com/' and 'List Name' with the value '0769e0e8-aec5-4441-8c88-6283a6799718'. Below these is a 'Show advanced options' link. A connector arrow points down to the 'Apply to each' step, which is highlighted with a grey border. This step has a dropdown menu for 'Select an output from previous steps' showing 'value'. Below this is another 'Get items' step, also highlighted with a blue border, titled 'Delete item'. It contains three dropdown menus: 'Site Address' with 'Critical Path Training Labs Team Site - https://msd0910.sharepoint.com/', 'List Name' with 'List1', and 'Id' with 'ID'. At the bottom of the editor are three buttons: 'Add an action', 'Add a condition', and 'More'.

Get items

* Site Address

* List Name

Show advanced options

Apply to each

* Select an output from previous steps

Delete item

* Site Address

* List Name

* Id

Add an action Add a condition More



Agenda

- ✓ Microsoft Flow Fundamentals
- ✓ Creating and Testing Flows
- ✓ Using Control of Flow Actions
- Writing Flow Expressions
 - Processing Data and Preparing Content



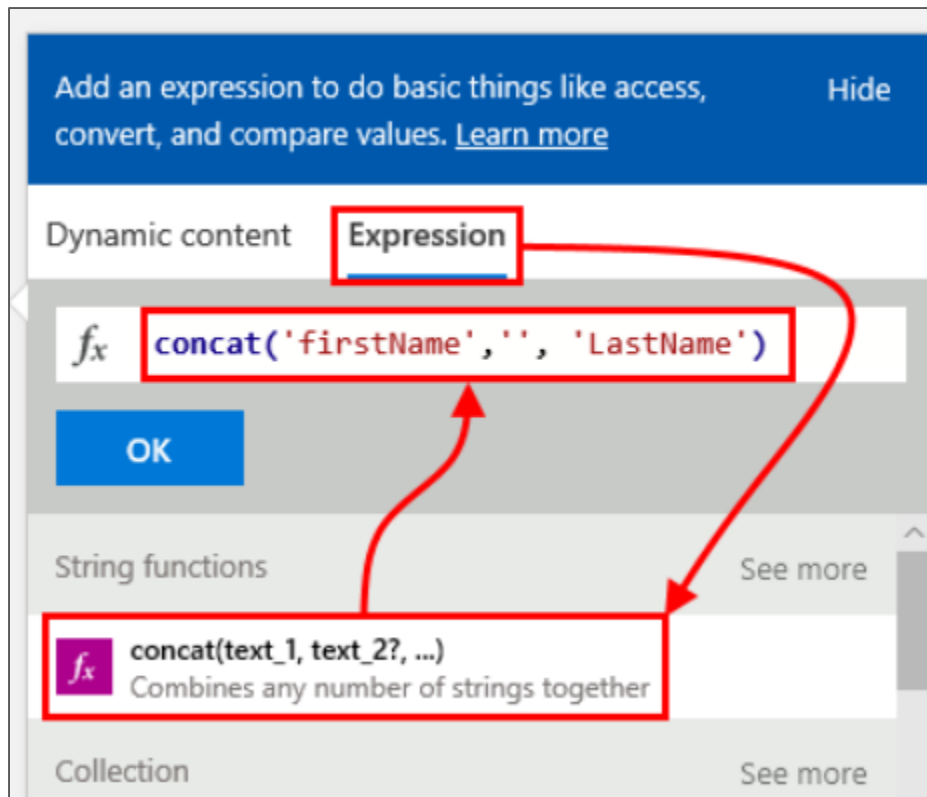
Writing Flow Expressions

- 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...



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 item field value - `body('Get_items')?['value'][0]['ID']`

The screenshot shows a Power Automate flow. The first action is 'Get items' (indicated by a downward arrow). The second action is 'Delete item' (indicated by a plus sign in a circle). The 'Delete item' action has the following configuration:

- *Site Address: Critical Path Training Labs Team Site - <https://msd0910.sharepoint.com/>
- *List Name: List1
- *Id: `body('Get_items')?['value'][0]['ID']` (with a dynamic content icon and a close button)

An 'Add dynamic content' button is visible at the bottom right of the 'Delete item' action card.

The screenshot shows the 'Dynamic content' pane in Power Automate. The 'Expression' tab is selected. The expression entered is:

```
body('Get_items')?['value'][0]['ID']
```

An 'Update' button is visible below the expression field.



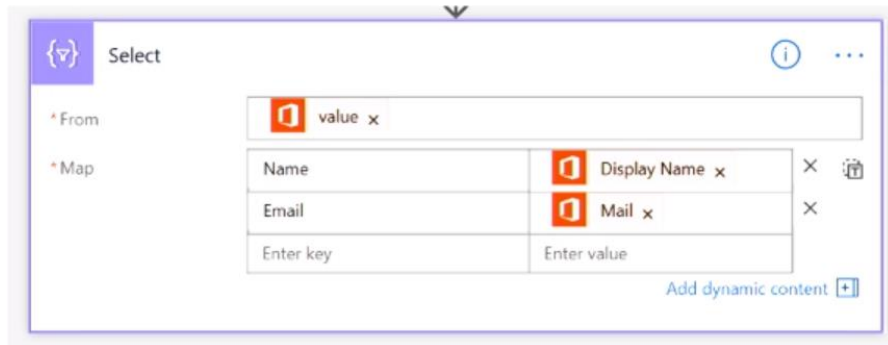
Agenda

- ✓ Microsoft Flow Fundamentals
- ✓ Creating and Testing Flows
- ✓ Using Control of Flow Actions
- ✓ Writing Flow Expressions
- Processing Data and Preparing Content



Transforming Arrays

- Use Select action
 - Two input modes: fill key-value pairs or typing directly
- Create array object objects
 - Useful for passing array to another action



- Create a simple array of strings, numbers, Booleans, etc
 - Useful for creating simple list (e.g. email addresses)



Converting an Array using Select

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

- Manually trigger a flow**: The starting trigger.
- Get items**: Retrieves data from a source.
- Email Array**: A variable of type Array. It contains:
 - *From**: A dynamic content field showing `value`.
 - *Map**: A dynamic content field showing `Email Address`.
- Get Parsed Email Addresses**: An action that uses the `join(...)` function to convert the array into a string.

The **Get Parsed Email Addresses** action is configured with the following inputs:

- *Inputs**: `join(...)`

The right-hand pane shows the **Expression** tab with the following code:

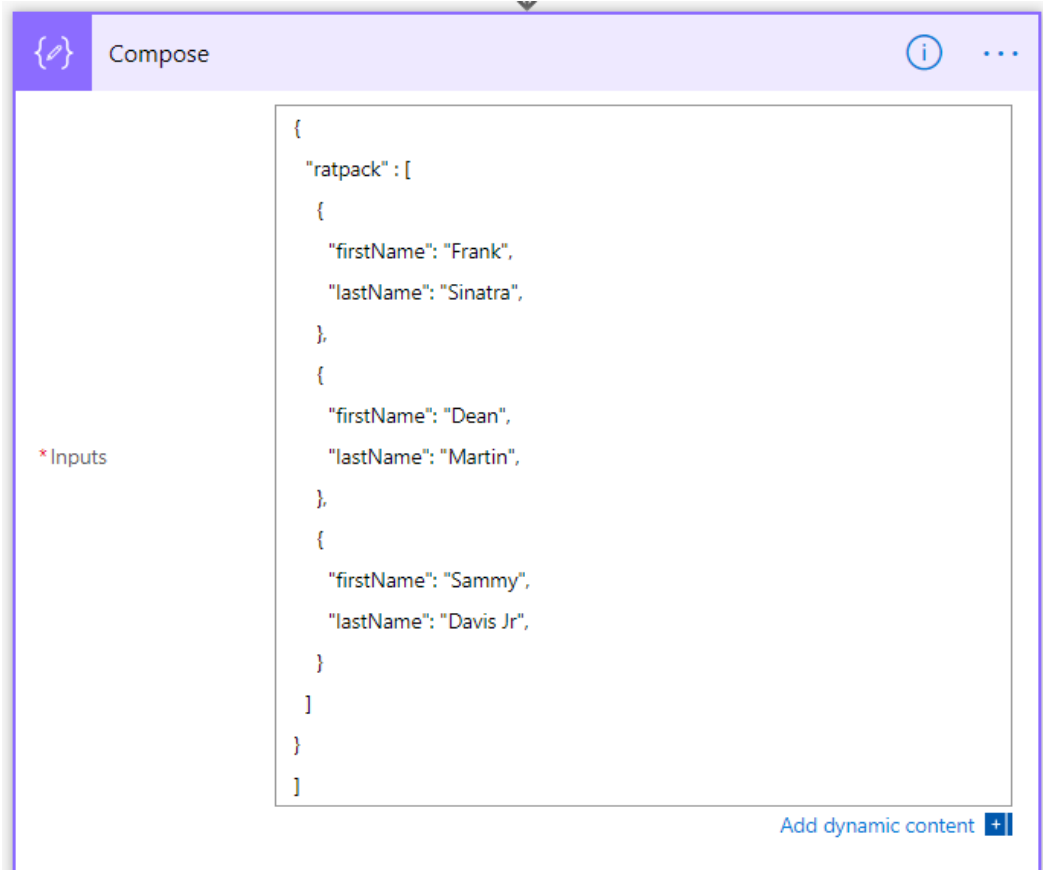
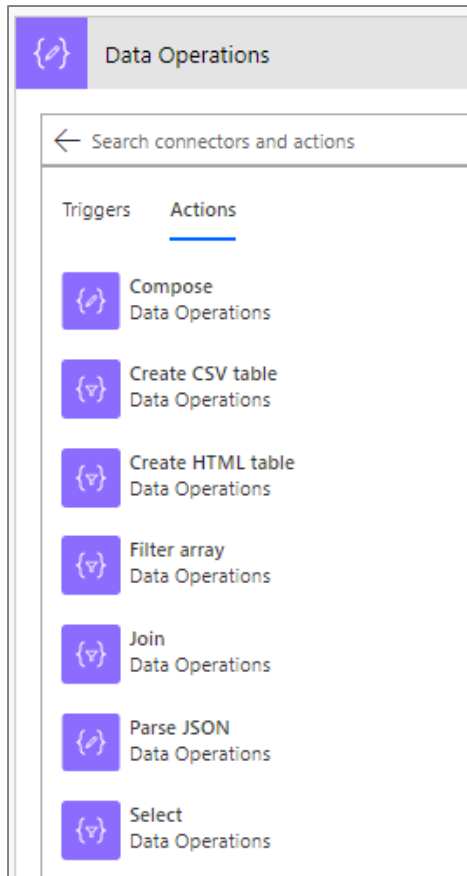
```
fx join(body('Email_Array'),';')
```

Below the code is an **Update** button. A tooltip at the top right of the pane reads: "Add an expression to do basic things like access, convert, and compare values. [Learn more](#)" with a **Hide** button.



Data Operations

- Used to process data and prepare content



Handling Type Conversion

- Some conversion is automatic
 - Sometimes conversions are performed for you
 - In other cases, you must explicitly convert between types

**string(value)**

Convert the parameter to a string

**float(value)**

Convert the parameter argument to a floating-point number

**bool(value)**

Convert the parameter to a Boolean

**base64(value)**

Returns the base 64 representation of the input string

**base64ToBinary(value)**

Returns a binary representation of a base 64 encoded string

**base64ToString(value)**

Returns a string representation of a base 64 encoded string

**binary(value)**

Returns a binary representation of a value

**dataUriToBinary(value)**

Returns a binary representation of a data URI

**dataUriToString(value)**

Returns a string representation of a data URI

**dataUri(value)**

Returns a data URI of a value

**uriComponent(value)**

Returns a URI encoded representation of a value

**uriComponentToBinary(value)**

Returns a binary representation of a URI encoded string

**uriComponentToString(value)**

Returns a string representation of a URI encoded string




Flow Type Conversion Matrix

To From	UI adds automatically					Floating-point	Integer	Bool.	Array	JSON Object	XML content
	String	Base 64	Binary content	Data URI	URI comp.						
String	Yes	base64()	binary()	dataUri()	uriComponent()	float()	int()	bool()	split() json()	json()	xml()
Base 64	base64ToString()	Yes	base64ToBinary()	*	*	*	*	*	*	*	*
Binary content	string()	base64()	Yes	dataUri()	uriComponent()	*	*	*	*	*	*
Data URI	dataUriToString()	*	dataUriToBinary()	Yes	*	*	*	*	*	*	*
URI comp.	uriComponentToString()	*	uriComponentToBinary()	*	Yes	*	*	*	*	*	*
Floating-point	Yes	base64()	binary()	dataUri()	uriComponent()	Yes	No	No	No	No	No
Integer	Yes	base64()	binary()	dataUri()	uriComponent()	Yes	Yes	No	No	No	No
Bool.	Yes	base64()	binary()	dataUri()	uriComponent()	No	No	Yes	No	No	No
Array	join() string()	*	*	*	*	No	No	No	Select Action	Select or Compose	xml()
JSON object	string()	*	*	*	*	No	No	No	Select or Compose	Compose Action	xml()
XML content	string()	*	*	*	*	No	No	No	xpath()	xpath()	Logic apps only



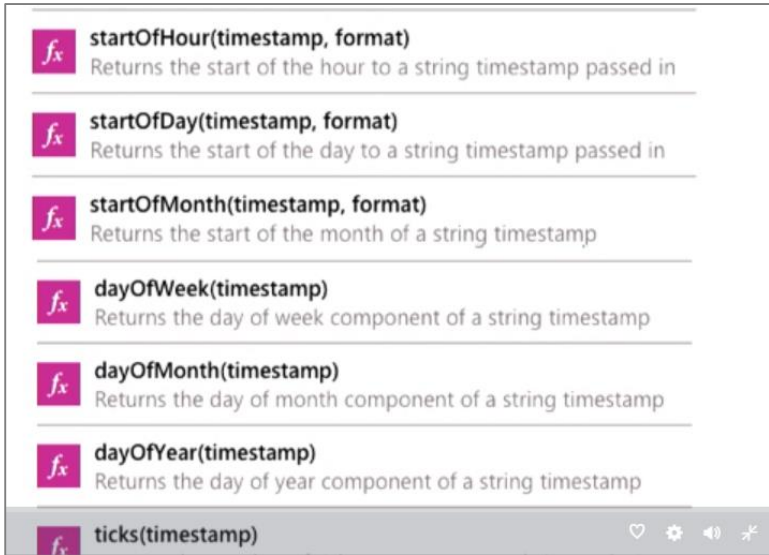
Working with Dates and Time

- 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
- 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
- ticks(timestamp)**

At the bottom right of the editor window, there is a toolbar with icons for a heart, a gear, a speaker, and a star.



dataUriToBinary()

- PowerApps photos require conversion
 - Allows you to upload photos to SharePoint
 - Accomplished using **dataUriToBinary()** function

```
dataUriToBinary(triggerBody()['Createfile_FileContent'])
```

The screenshot displays a PowerApps flow interface. The first step is 'Get File Name', which uses the 'concat(...)' function. The second step is 'Create file', which is configured with the following fields:

- Site Address:** Critical Path Training Labs Team Site - <https://msd0910.sharepoint.com/>
- Folder Path:** /My Photos
- File Name:** Output
- File Content:** dataUriToBinary(triggerBody()['Createfile_FileContent'])

On the right side, the 'Dynamic content' pane is open, showing the 'Expression' tab. It displays the function `dataUriToBinary(triggerBody()['Createfile_FileContent'])` and includes an 'Update' button. Below this, a list of 'String functions' is visible, including 'concat(text_1, text_2?, ...)' with a description: 'Combines any number of strings together.'

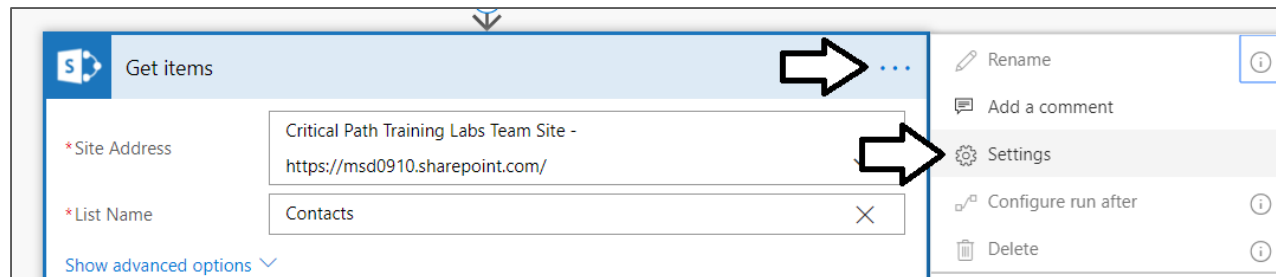
Normal action execution

- Standard behavior of a flow
 - Action steps execute in sequential order
 - Flow terminates if error occurs (failure or timeout)
- After flow runs, every action left in 1 of 4 possible states



Action Settings


- Settings let you configure
 - Async Actions
 - Timeouts
 - Retry Policy
 - Sequential Behavior
 - And more!




Error Handling

- Select the **Run after** option from action menu
 - Choose which error conditions, the arrow will turn dotted red
 - Use parallels for errors that are not at end of flow
 - Retry policy by default handles transient failures
 - Recommended to select exponential as they last a long time

'Handle duplicate file names' should run after:

 This step will only run if the flow couldn't create the file because there's already another one with the same name. It will add some numbers to the end of the file name to make it unique.

 **Create file**
Failed

☐ is successful
☒ has failed
☐ is skipped
☐ has timed out

Done **Cancel**

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 retry policy is to retry 4 times with a 20 second delay between each attempt.

Retry policy type

Interval ⓘ

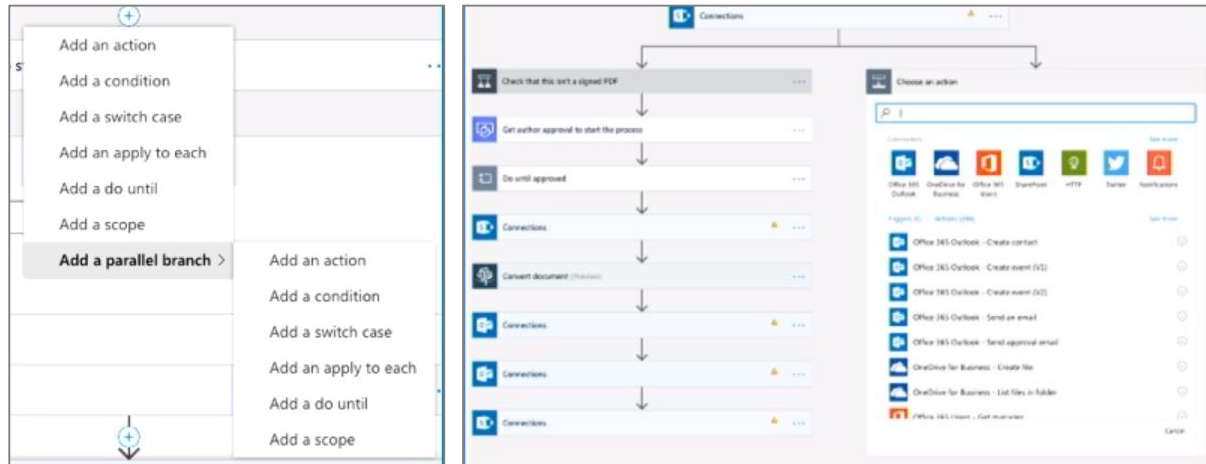
Count

Done **Cancel**

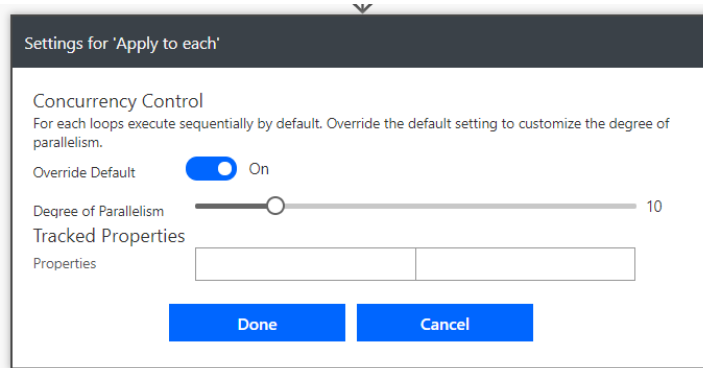


Parallel Execution

- Add parallel branch from above using ⊕

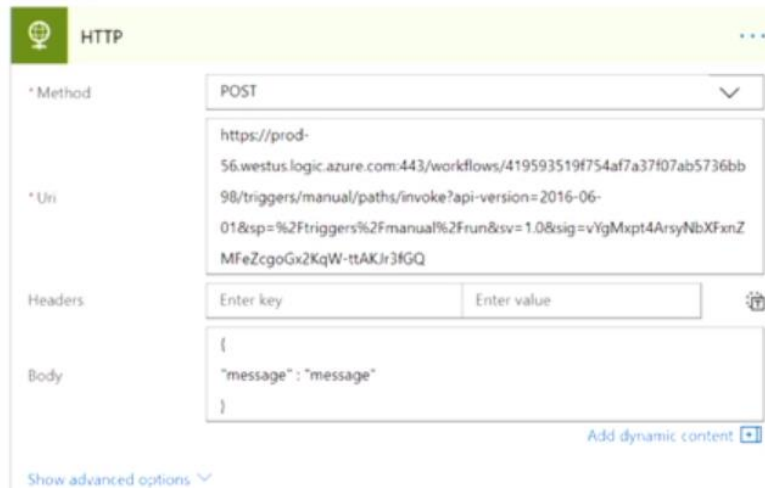


- Apply to each is sequential by default
 - Adding parallel execute to Apply to each



Calling Flows using the HTTP action

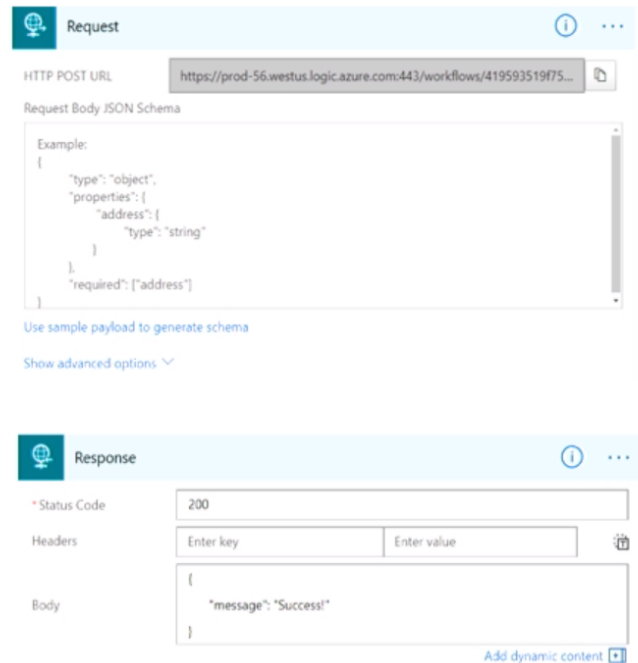
In the parent workflow:



The screenshot shows the configuration for an HTTP action in a parent workflow. The action is named "HTTP". The method is set to "POST". The URI is a long URL starting with "https://prod-56.westus.logic.azure.com:443/workflows/419593519f754af7a37f07ab5736bb98/triggers/manual/paths/invoke?api-version=2016-06-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=vYgMxpt4ArsyNbXfanZMFeZcgoGx2KqW-ttAKJr3fGQ". The headers section is empty, with input fields for "Enter key" and "Enter value". The body is a JSON object: {"message": "message"}. There is a link to "Show advanced options" and a button to "Add dynamic content".



In the child workflow:



The screenshot shows the configuration for a Request and Response action in a child workflow. The "Request" section is expanded, showing the "HTTP POST URL" as "https://prod-56.westus.logic.azure.com:443/workflows/419593519f754af7a37f07ab5736bb98/triggers/manual/paths/invoke?api-version=2016-06-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=vYgMxpt4ArsyNbXfanZMFeZcgoGx2KqW-ttAKJr3fGQ". The "Request Body JSON Schema" is shown with an example: {"type": "object", "properties": {"address": {"type": "string"}}, "required": ["address"]}. There are links to "Use sample payload to generate schema" and "Show advanced options". The "Response" section is also expanded, showing the "Status Code" as "200". The headers section is empty, with input fields for "Enter key" and "Enter value". The body is a JSON object: {"message": "Success!"}. There is a link to "Show advanced options" and a button to "Add dynamic content".




Summary

- ✓ Microsoft Flow Fundamentals
- ✓ Creating and Testing Flows
- ✓ Using Control of Flow Actions
- ✓ Writing Flow Expressions
- ✓ Processing Data and Preparing Content




Building a Flow

Flow name Track new Tweets containing #PowerApps ✓ Update flow ✕ Close


 When a new tweet appears ...

* Search text


↓

 Get user ...

* User name

 Tweeted by ✕


↓

 Send Email ...


* To

* Subject
 Name ✕"/>

* Body
Tweet:

 Tweet text ✕

Tweeted by:

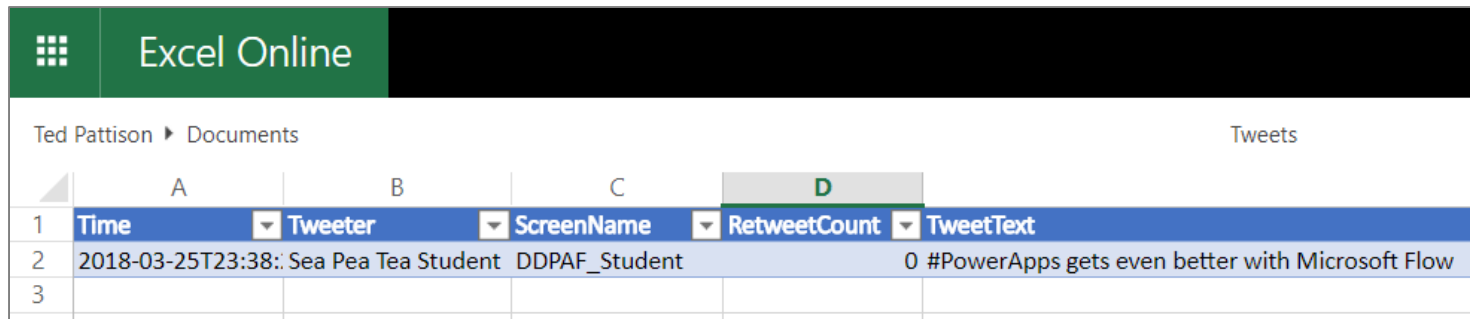
 Tweeted by ✕

Show advanced options ▾



Updating a Table in an Excel Workbook

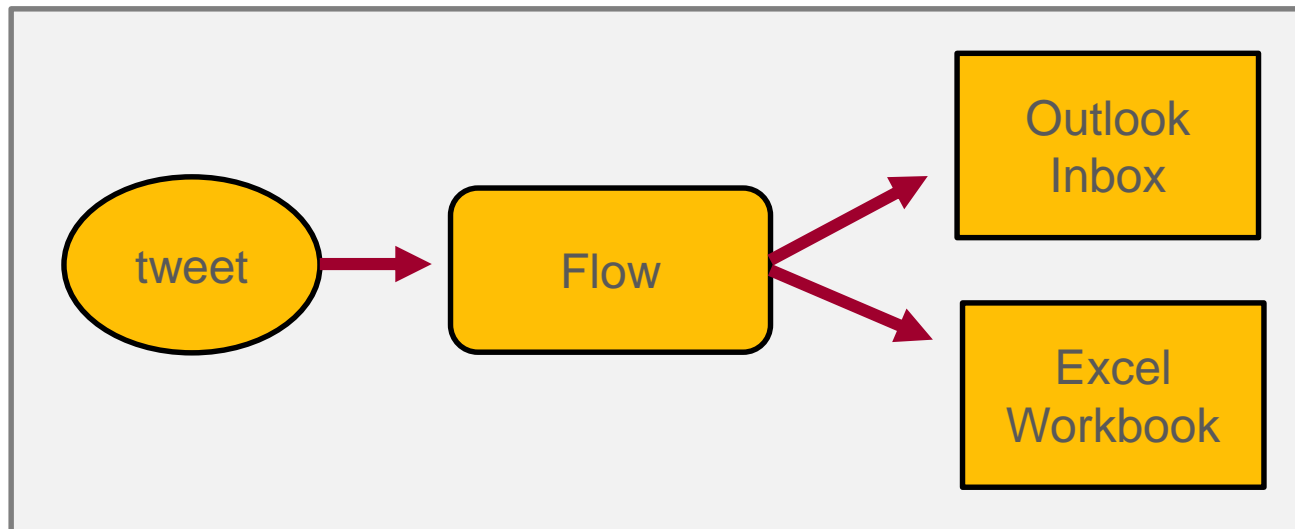
- Flow now writes tweets to Excel workbook as well as sending email





The screenshot shows the Excel Online interface. The top bar indicates 'Excel Online'. Below it, the user 'Ted Pattison' is logged in, and the document is named 'Documents'. The table is titled 'Tweets'. The table has five columns: 'Time', 'Tweeter', 'ScreenName', 'RetweetCount', and 'TweetText'. The first row of data shows a tweet from 'Sea Pea Tea Student' with a retweet count of 0. The second row is empty.

	A	B	C	D	
1	Time	Tweeter	ScreenName	RetweetCount	TweetText
2	2018-03-25T23:38:	Sea Pea Tea Student	DDPAF_Student	0	#PowerApps gets even better with Microsoft Flow
3					

- Observation: It's easy to design a flow to write data to multiple services



Run History Details

 Add a row into a table 4s 

INPUTS

Source

OneDrive for Business

Drive

OneDrive

File

/Tweets.xlsx

Table

Tweets

item

```
{  
  "Time": "2018-03-25T23:38:28.5422853Z",  
  "Tweeter": "Sea Pea Tea Student",  
  "ScreenName": "DDPAF_Student",  
  "RetweetCount": "0",  
  "TweetText": "#PowerApps gets even better with Microsoft Flow"  
}
```

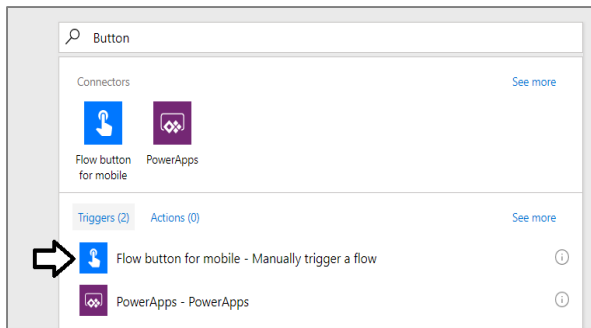
Time

2018-03-25T23:38:28.5422853Z



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




Building Out The Idea Tracker Flow


Flow name

Idea Tracker

✓ Update flow

✕ Close

 Manually trigger a flow

 Quality

Please enter your input

...

List Options


Lame


Good


Great


Brilliant


Enter another option










 Idea


Please enter your input

...

+

 Add an input




 Send an email

* To


Student@DDPAF.onmicrosoft.com;

* Subject

 Quality ✕

 Idea for consideration

* Body

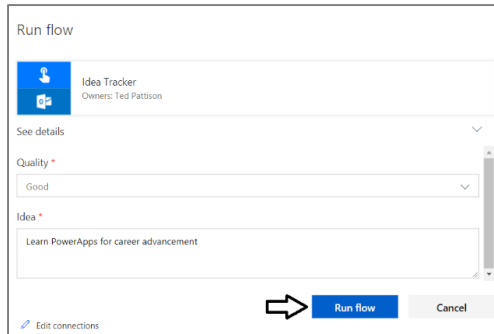
 Idea ✕

Show advanced options ▾

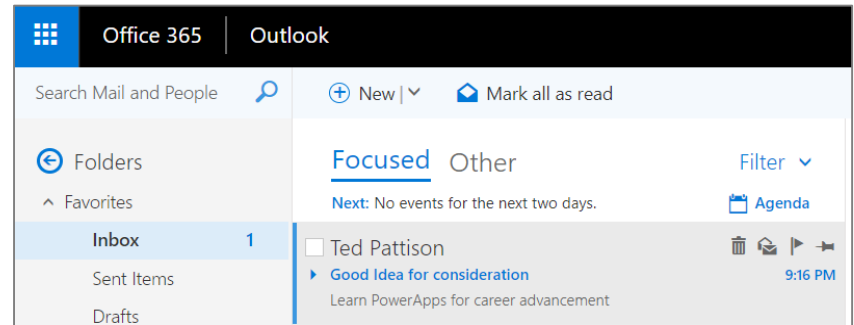
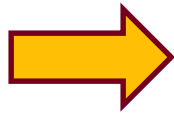


Where Should You Write the Idea?

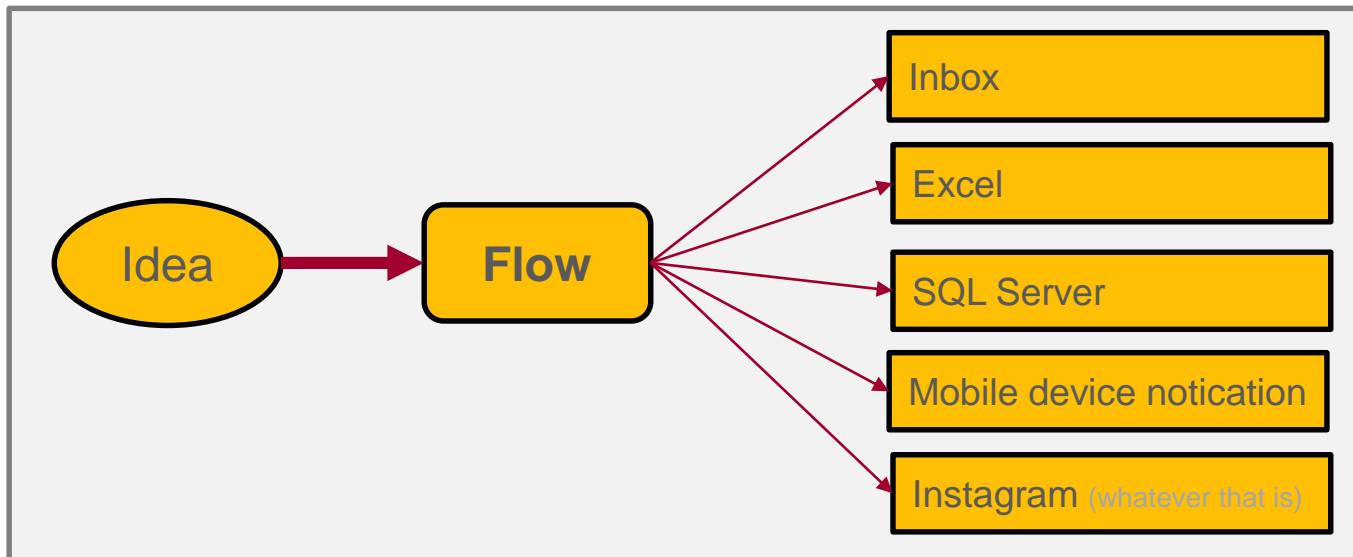
- You can track the idea by sending an email



The 'Run flow' dialog box for the 'Idea Tracker' flow (owned by Ted Pattison) is shown. It includes a 'See details' section with a 'Quality' dropdown set to 'Good' and an 'Idea' text box containing 'Learn PowerApps for career advancement'. At the bottom, there is a 'Run flow' button and a 'Cancel' button.



- Or get even more extravagant



Creating a Flow for a SharePoint List

CP

Critical Path Labs Team Site

+ New

Quick edit

Export to Excel

Flow

PowerApps

...

Expenses

Expense

Category

Date

Amount



Verizon - Telephone and Internet	Operations	1/1/2018	\$923.00
Electricity Bill	Operations	2/5/2018	\$338.00





Create a flow
See your flows

Create a flow



Start with a template and create automated tasks between your SharePoint data and other apps. Choosing a template will open the Microsoft Flow site where you'll finish creating your flow.





Send a customized email when a new SharePoint list item is added





Start approval when a new item is added



When a message is posted on a group, create a SharePoint list item



Request manager approval for a selected item



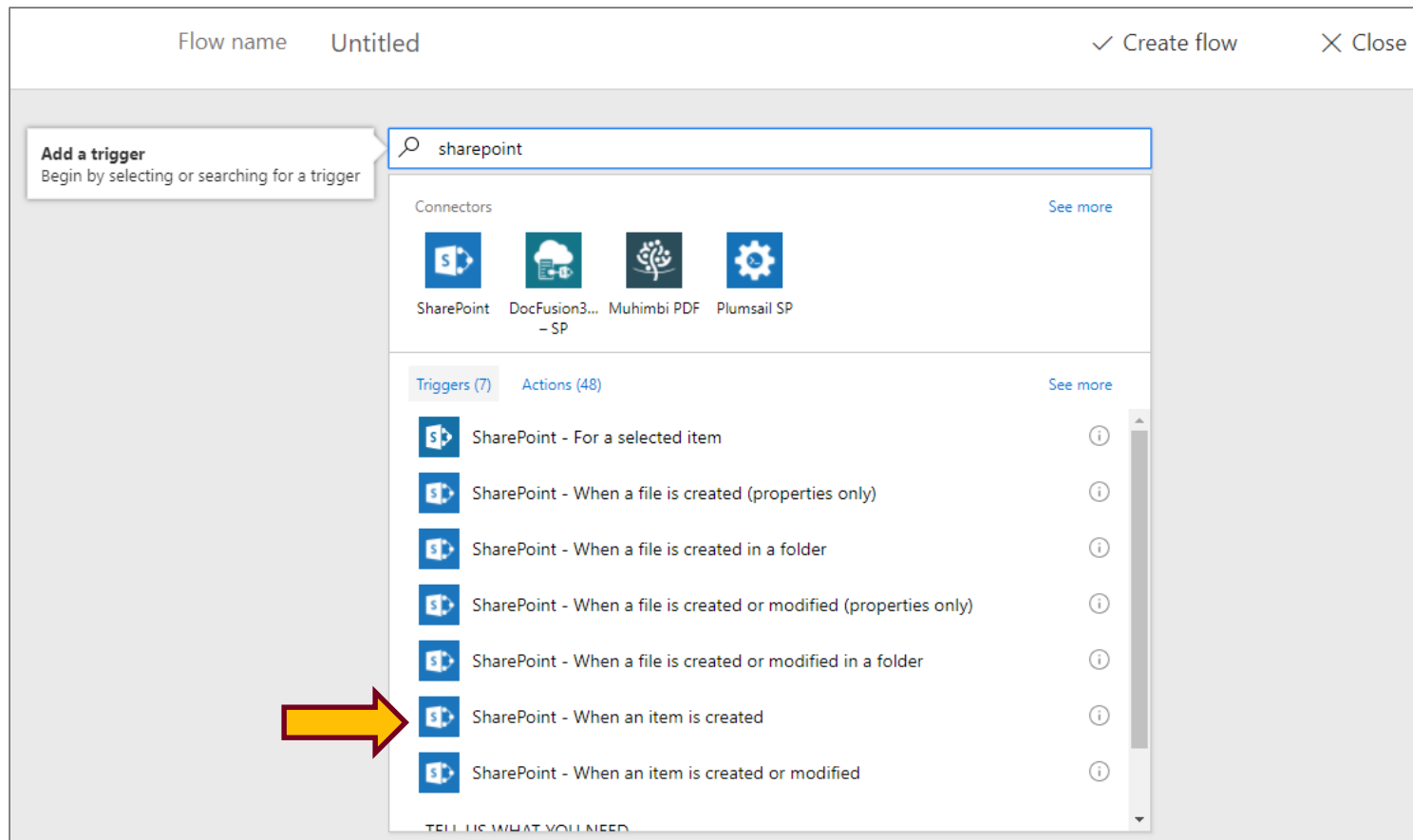
When an object is created in Dynamics 365, create a list item

Show more



Another Way to Create Flows for SharePoint

- Create the flow from blank using the Flow Designer
 - Then add trigger based on SharePoint item event



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



Build Out Branches using Actions

Flow name Large Expense Alert ✓ Create flow ✕ Close

When an item is created

Condition

Amount x is greater than or equal to 500

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

If yes

Send an email

*To: Ted Pattison x ;

*Subject: Large Expense Alert!

*Body:

Expense: Expense x

Category: Category Value x

Amount: Amount x

Link to item x

[Add dynamic content](#)

If no

[Add an action](#) [More](#)

