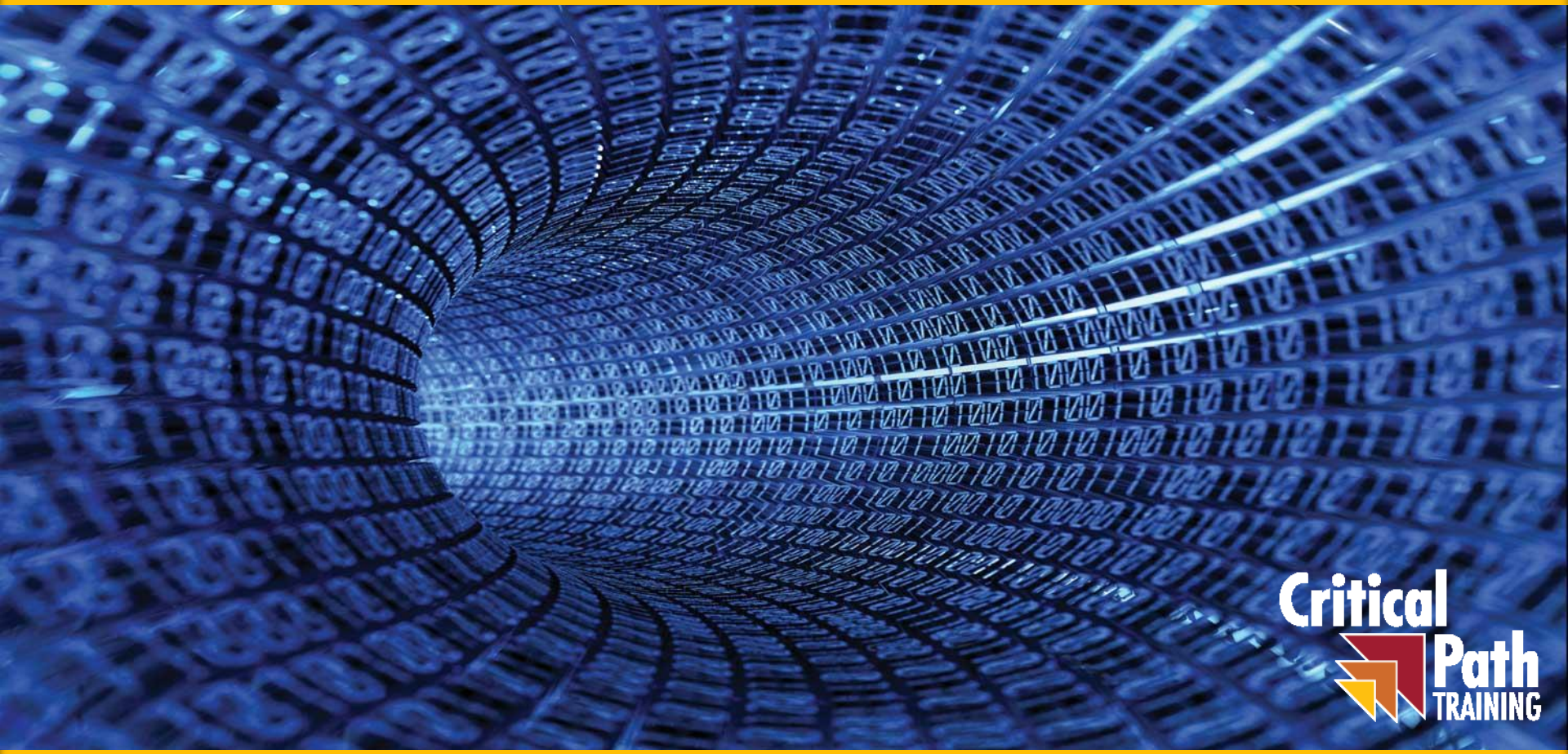
Introduction to Microsoft Flow



Agenda

- Flow Fundamentals
- Writing Flow Expressions
- Converting Between Types
- Working with Arrays
- Advanced Techniques

Thanks for coming up with the idea for this session

Stephen Siciliano, Principal Group PM Manager

Deep Dive into PowerApps and Flow

- Two action-packed days of building PowerApps and Flows
  1. Getting Started with PowerApps Studio
  2. Designing PowerApps using Advanced Techniques
  3. Building PowerApps for SharePoint Online
  4. Introduction to Microsoft Flow
  5. Designing Flows to Automate an Approval Process
  6. Building PowerApps and Flows for Power BI
  7. Working with the Common Data Service for Apps
  8. Managing Application Lifecycle with PowerApps and Flow

- More info
  - https://www.criticalpathtraining.com
  - info@criticalpathtraining.com

CriticalPath TRAINING

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
  - Create, copy, update and delete files
  - Collect and clean data
  - Automate approvals

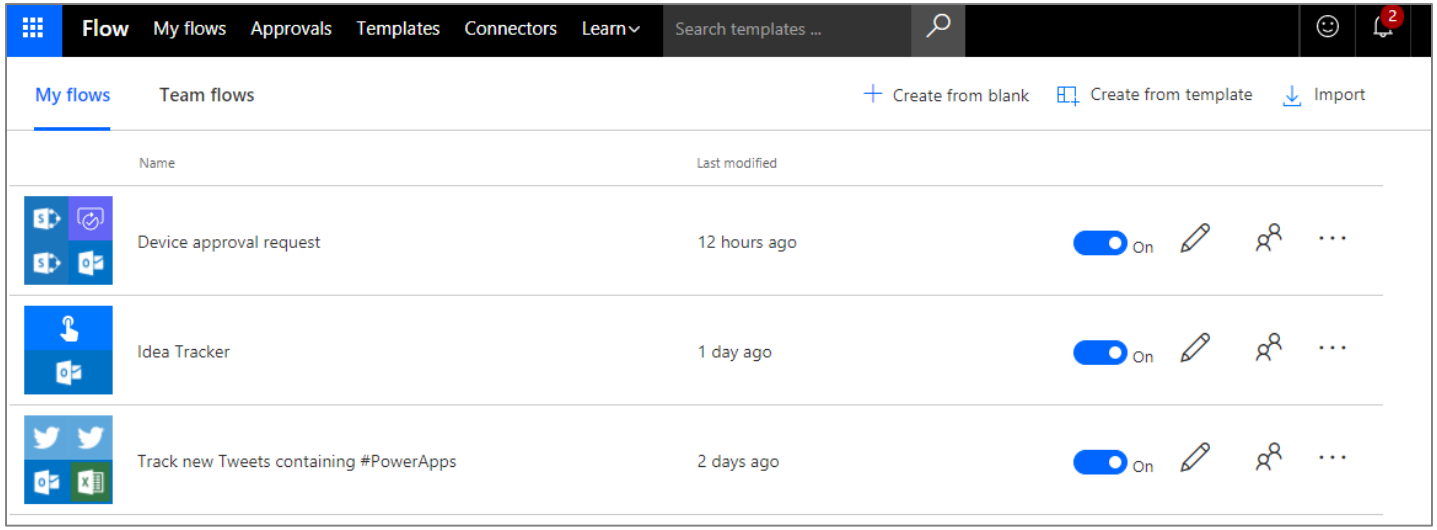
Building Blocks of Flow

- Flows whether created from a template or from scratch:
  - Will contain **building blocks** that work together in certain ways (much like a flowchart)

- Building blocks of Flow:
  - **Services** - sources and destinations of data in a flow
  - **Triggers** - events that start a flow
  - **Actions** - tasks accomplished by the flow
  - **Conditions** - allow for branching if/then logic in a flow
  - **Loops** - for iterating over actions more than once

Creating and Managing Flows

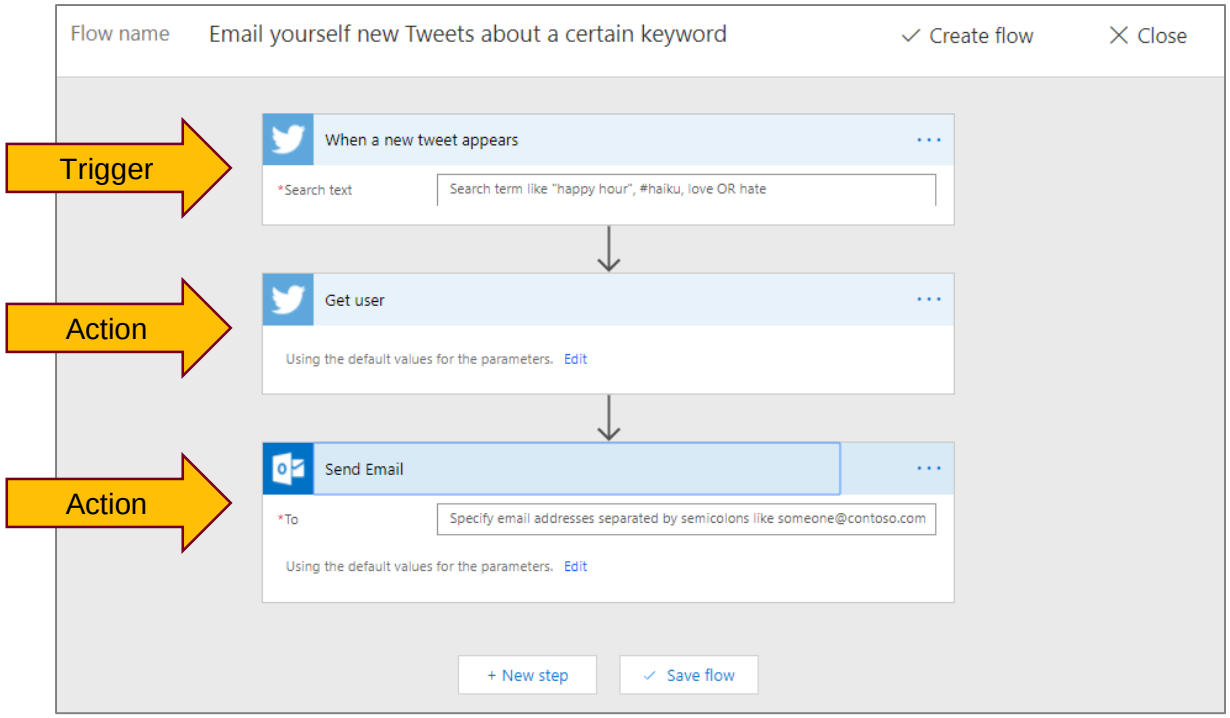
- Each user has a list of flows
  - Flows can be shared with or transferred to other users



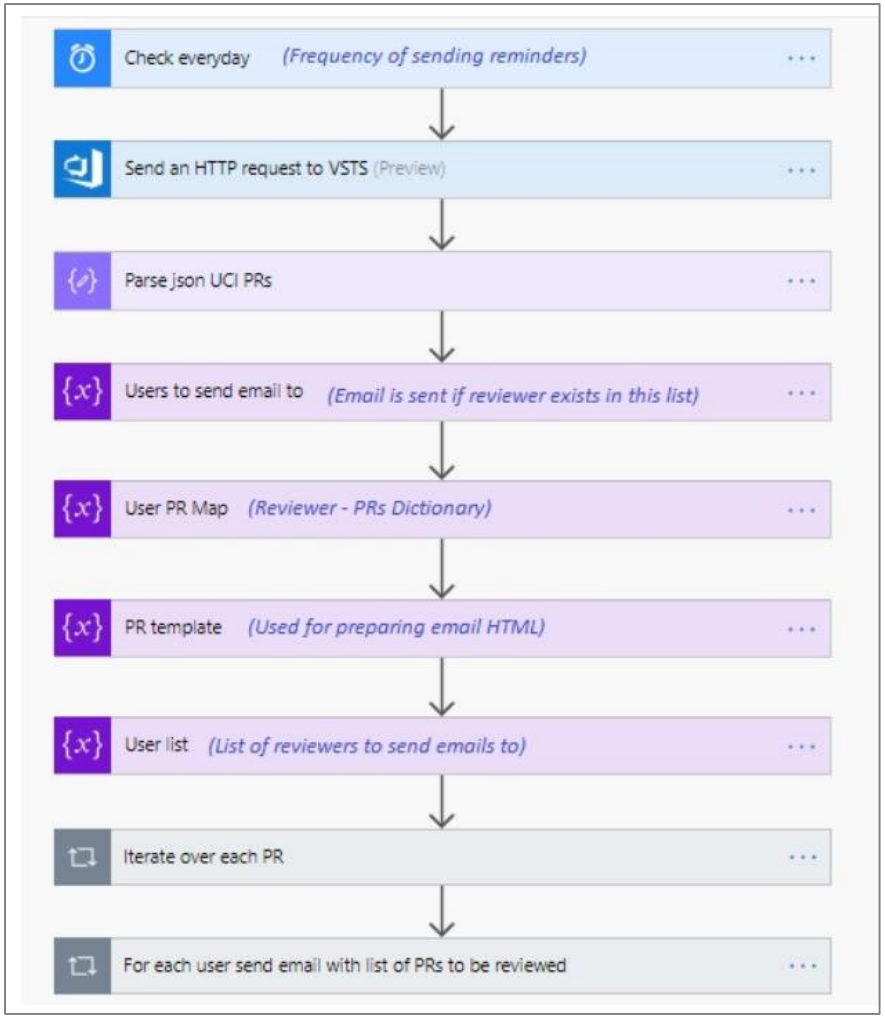
- Two ways to create a new flow
  - Create from template
  - Create from blank

Working with the Flow Designer

- Flow Designer provides UI for building flows
  - You build flows by adding and configuring steps
  - There are 3 types of steps: triggers, actions or conditions
  - Rename steps before referencing them in later steps

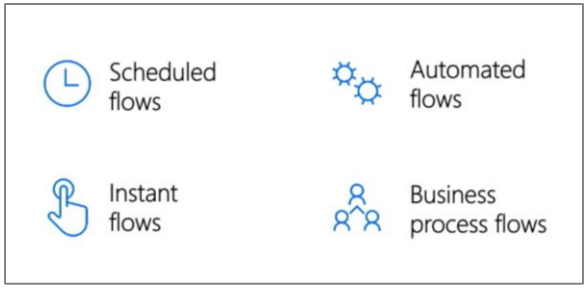


Example Award Winning Flow



Types of Flows

- Automated Flows
  - Runs when something happens (e.g. new list item)
- Scheduled Flows
  - Runs periodically based on an interval
- Instant Flows
  - Runs when a user clicks a button
- Business Process Flows
  - Integrated with CDS for Apps and Dynamics 365



DEMO

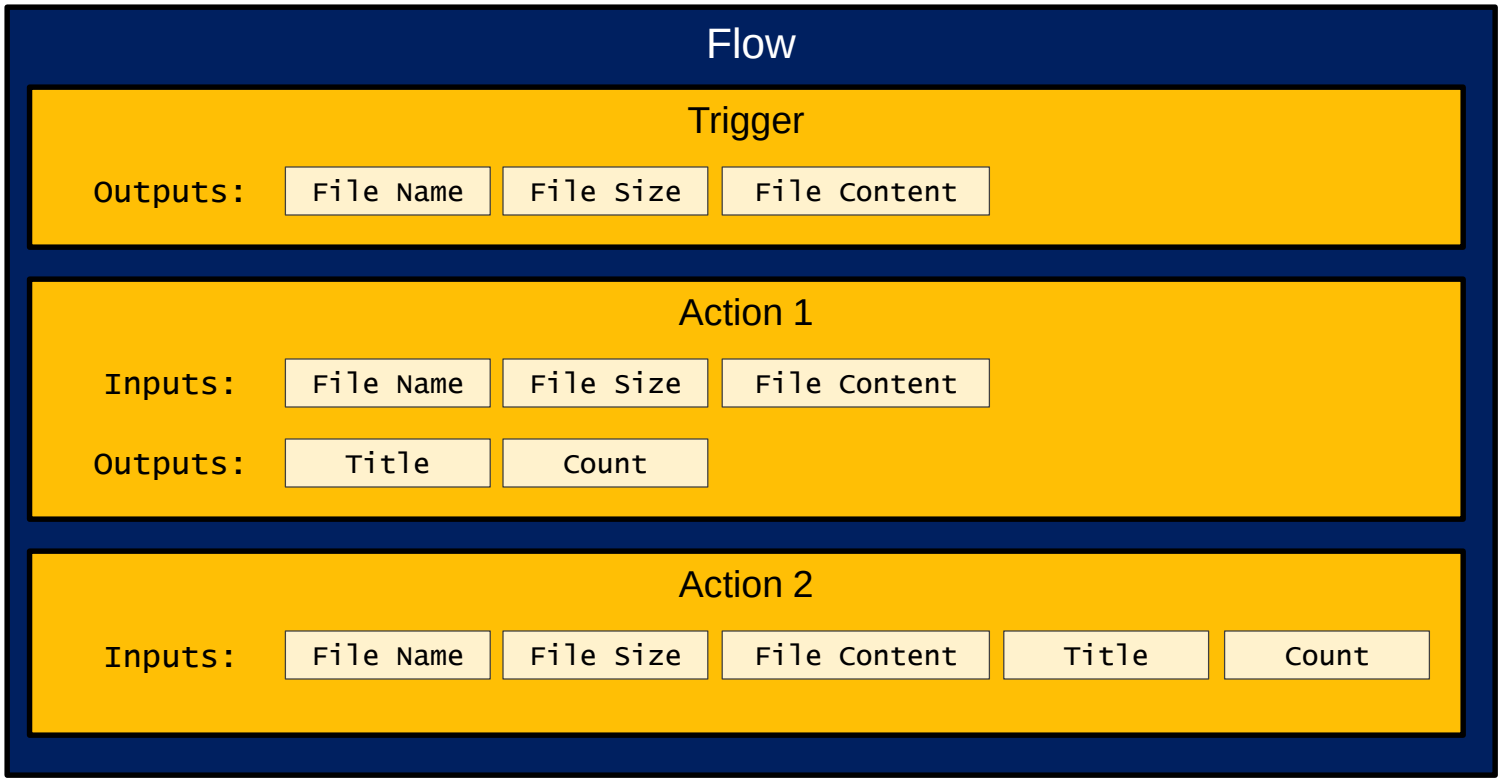
Changing the Flow Trigger Type

Agenda

- ✓ Flow Fundamentals
- ➢ Writing Flow Expressions
- Converting Between Types
- Working with Arrays
- Advanced Techniques

Flowing Data

- Data in flows added by steps
  - Data added in step is available in later steps
  - Add Dynamic Content select outputs from previous steps
  - Certain outputs shows up based on types of input and output



Writing Flow Expressions

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

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

```
fx "this is a test"
```

Invalid: no double quotes

```
fx 'this is a test'
```

Valid

```
fx 'this is a ' + 'test'
```

Invalid : **+** operator not supported

```
fx 'this is a ' & 'test'
```

Invalid : **&** operator not supported

```
fx concat('this is a ', 'test')
```

Valid

Using the Add Dynamic Content

How to create expressions

1. Add Dynamic Content

2. Dynamic content | Expression

   fx `replace(body('Get_record')?`

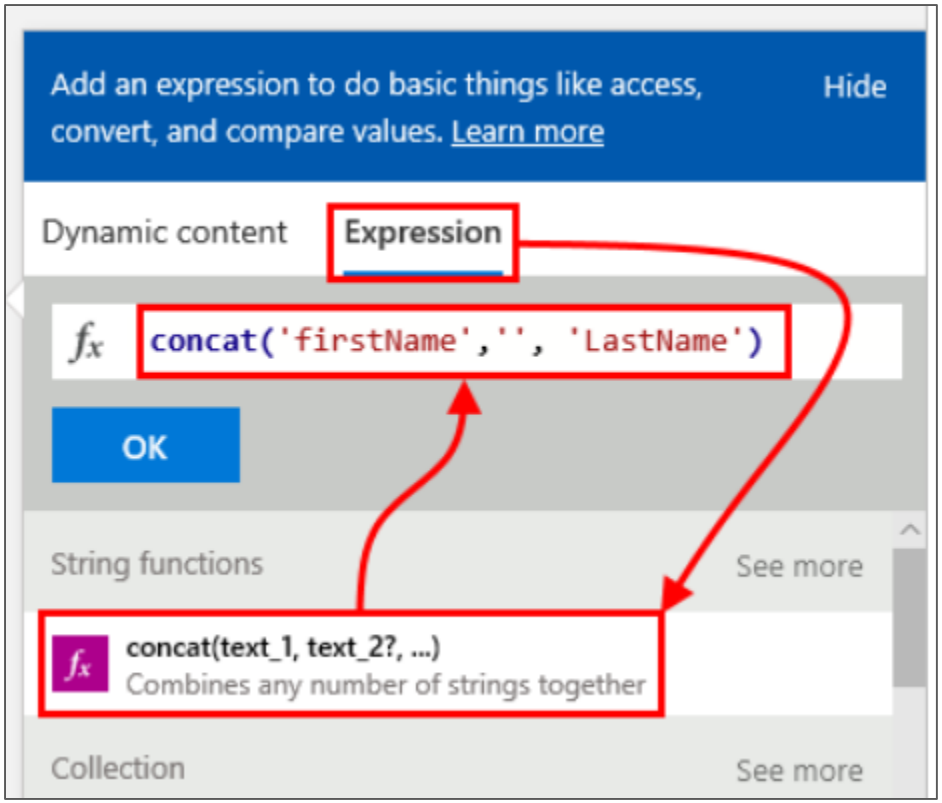
   OK

3. Action name

   Inputs: fx replace(...) ×

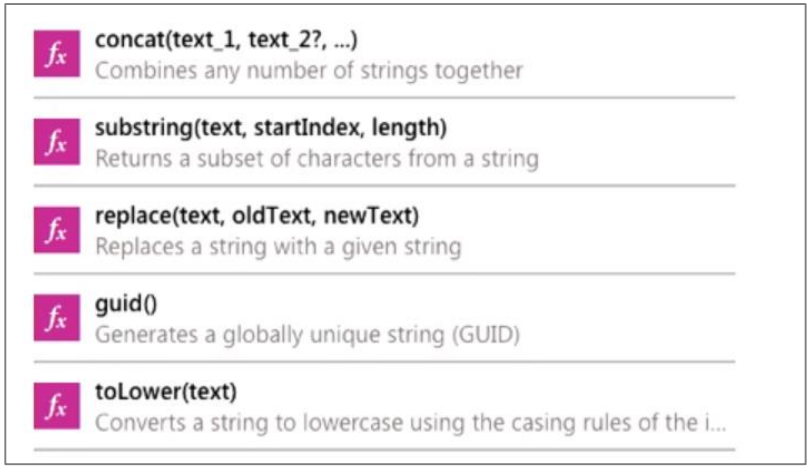
Writing Expressions

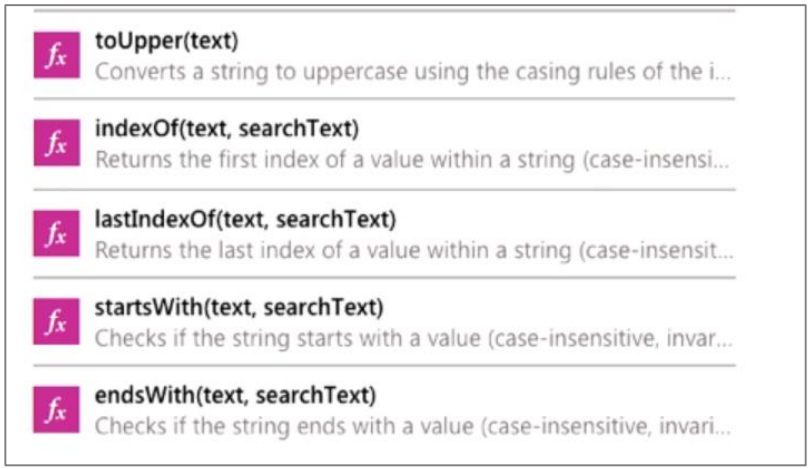
- Expressions written in fx textbox
- Click OK to enter expressions



Working with Strings

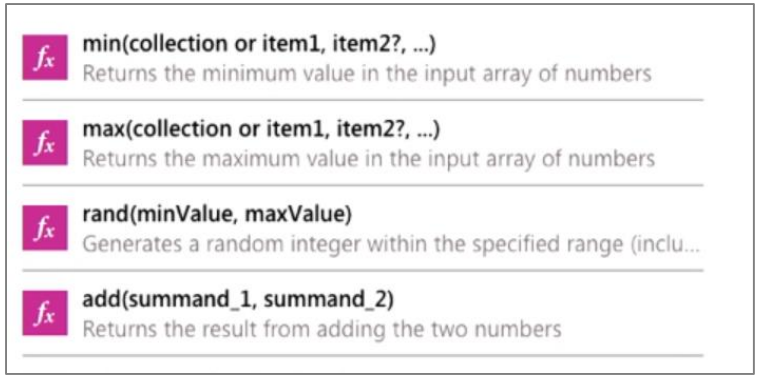
- Parse text together using **concat()**
- Parse out text using **substring()**
- Convert casing using **toLower()** and **toUpper()**
- Search string using **indexOf** and **startsWith()**
- Create new GUID identifier using **guid()**

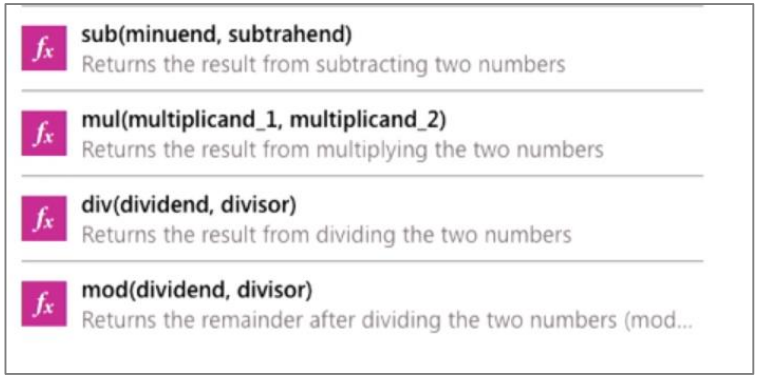




Performing Arithmetic Operations

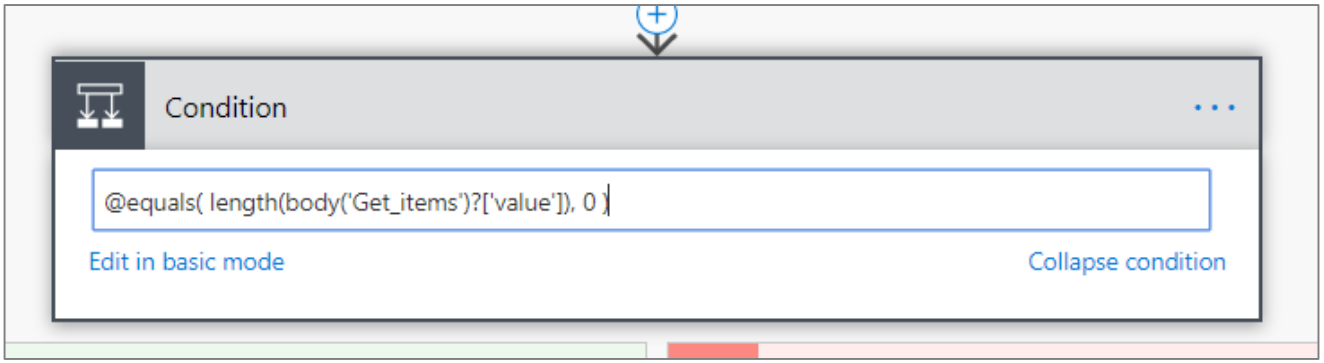
- You cannot use standard arithmetic operators
  - This includes familiar operators such as **+**, **-**, ***, **/**
  - This does not work: **2 + 2**
  - This works: **add(2, 2)**

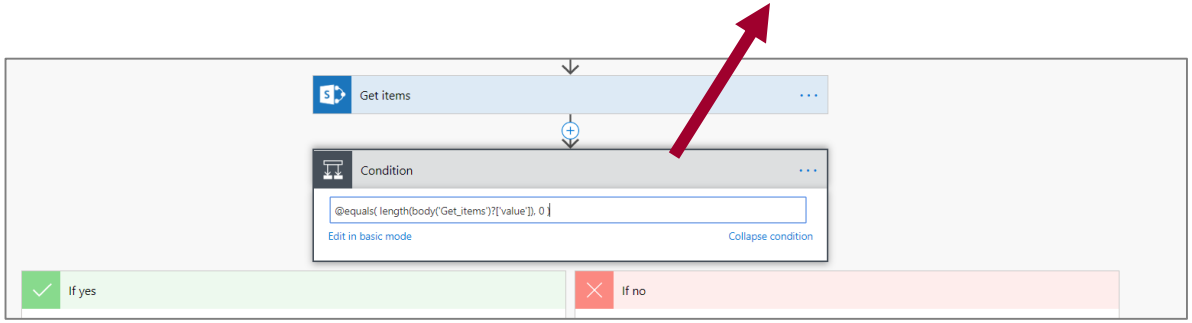




Using Boolean Expressions in a Condition

- Do as much as you can in basic mode
  - Then select Edit in advanced mode
  - Enter expression as Boolean expression
  - Unlike in Dynamic Content, you must begin with @





Escaping Rules

- Escaping often required when writing expressions

Using strings with a single quote character:

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

Using spaces in action names:

```
@body('Name_with_spaces')
```

Building an object you need to escape '@':

```
{
    "@@Odata.type" : ""
}
```

Agenda

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ➢ Converting Between Types
- Working with Arrays
- Advanced Techniques

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

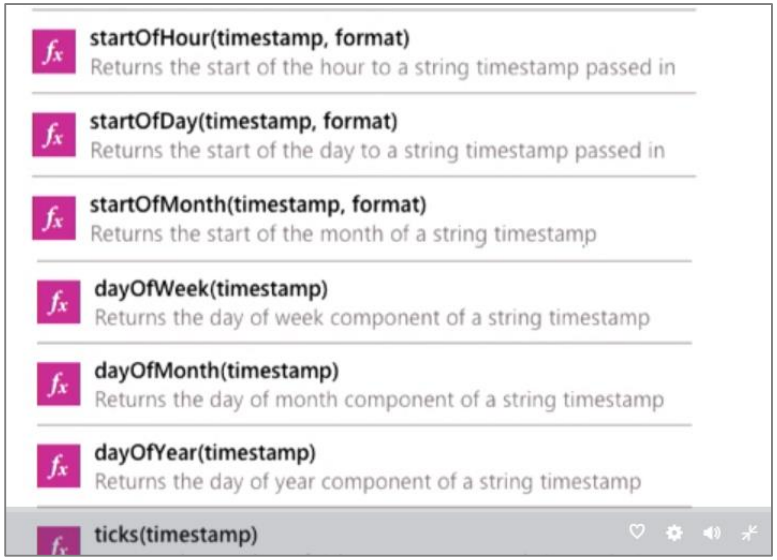
Flow Type Conversion Matrix

UI adds automatically

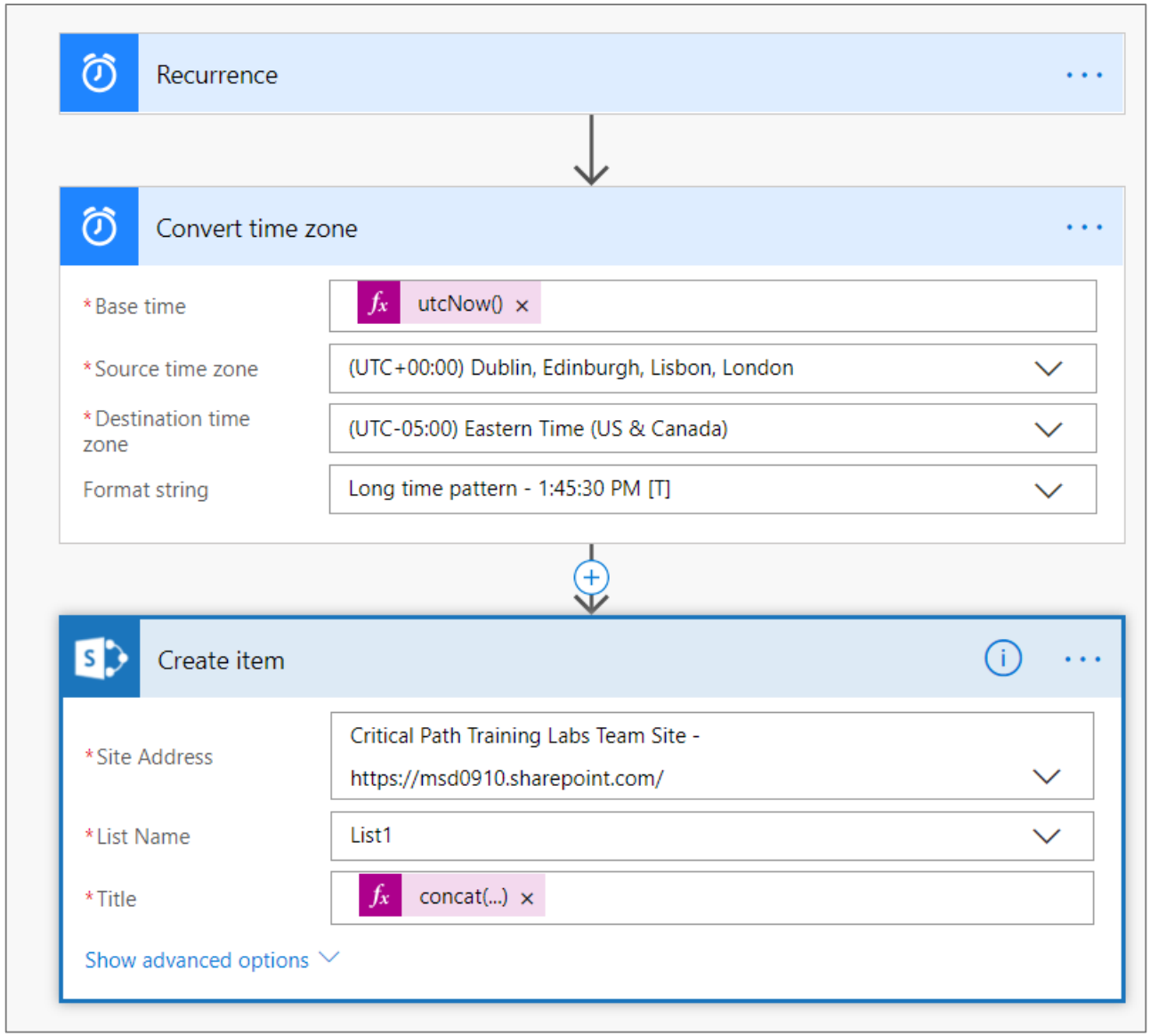
To → / From ↓	String	Base 64	Binary content	Data URI	URI comp.	Floating-point	Integer	Bool.	Array	JSON Object	XML content
String	Yes	base64()	binary()	dataUri()	uriComponent()	float()	int()	bool()	split() json()	json()	xml()
Base 64	base64ToString()	Yes	base64ToBinary()	*	*	*	*	*	*	*	*
Binary content	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



Using Convert time zone

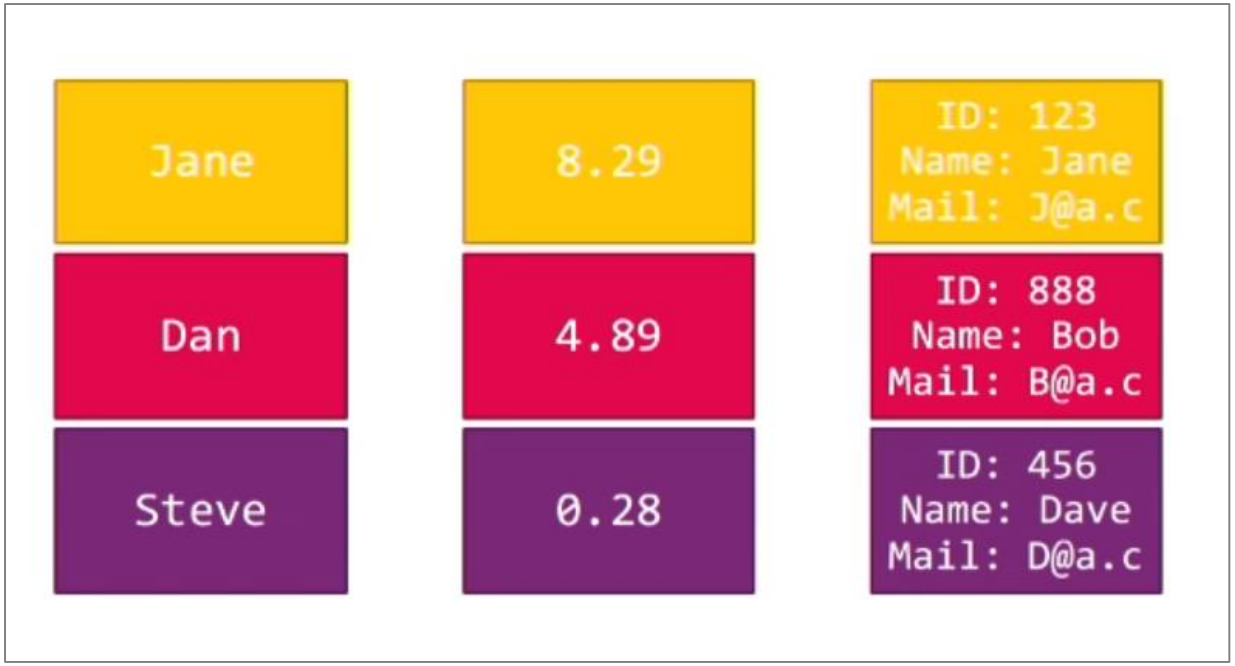


Agenda

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ✓ Converting Between Types
- ➢ Working with Arrays
- Advanced Techniques

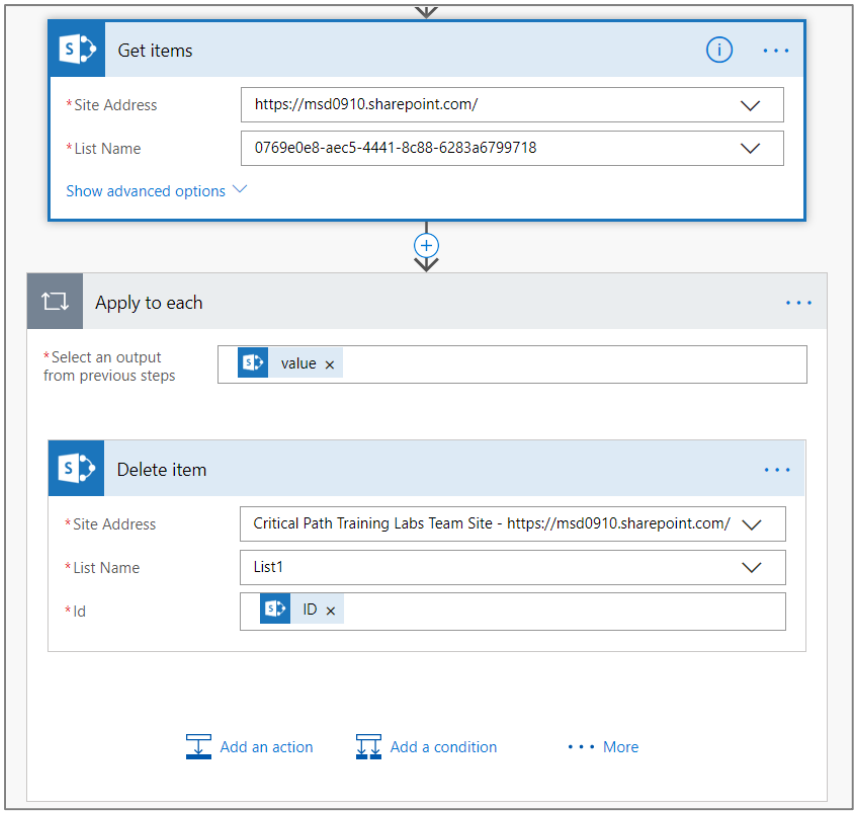
Working with Arrays in Flow

- Scenarios for working with arrays
  - Repeat action of each item in SharePoint list
  - Retrieve single item from a list
  - Filter list items



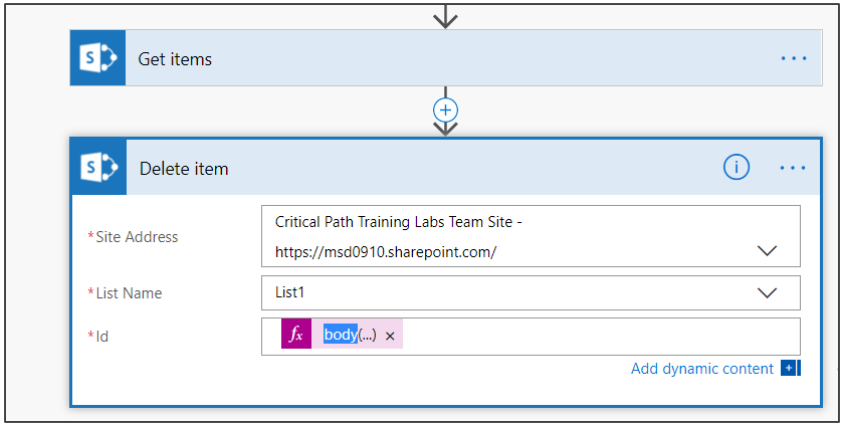
Using Apply to Each

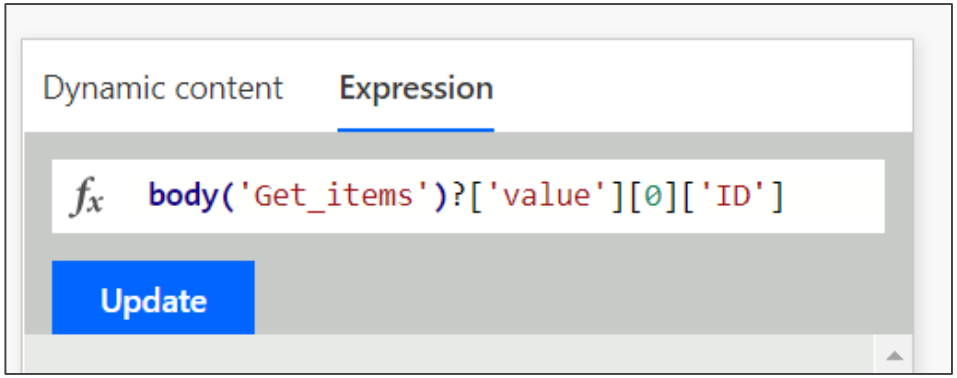
- Automatically added when list is used from output
- Destination step enumerates over list items



Retrieving List Items

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



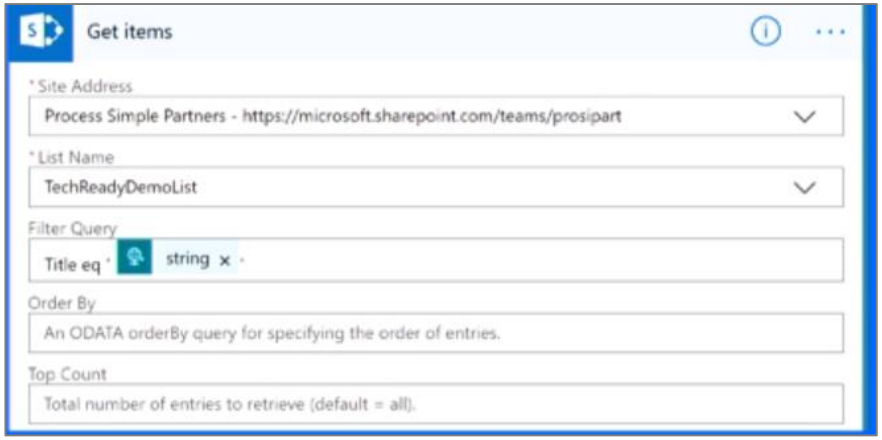


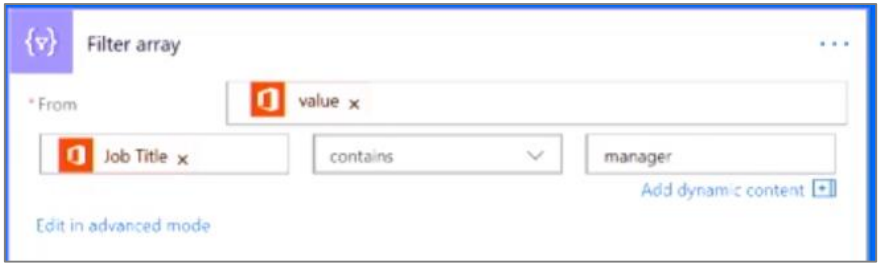
Convert a List to a String

- Join
  - - Use join for simple tasks such as combining email
    - Use must work with string array and not an object array
- Create HTML/CSV Table
  - Convert list of objects into tabular display format

Filtering

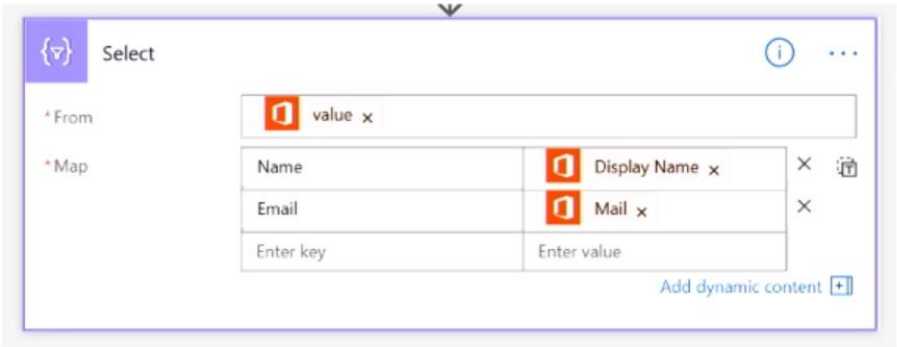
- Best to have connect perform filtering
  - Often requires use of OData query string parameters
- There is also a built-in Flow action named **Filter array**
  - Select the Array in the From field
  - Can use simple or advanced mode just like in condition
  - If you need the first N items, use **take()** and **skip()**



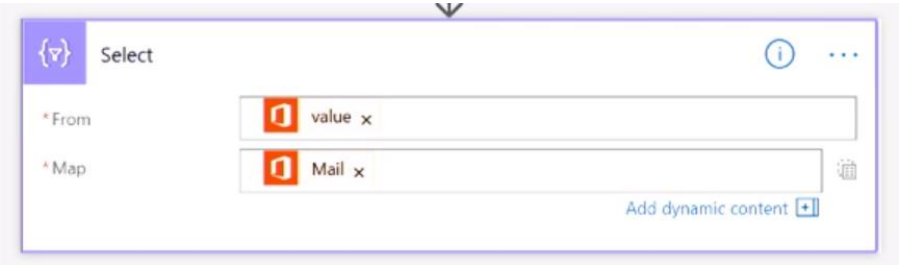


Transforming Arrays

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

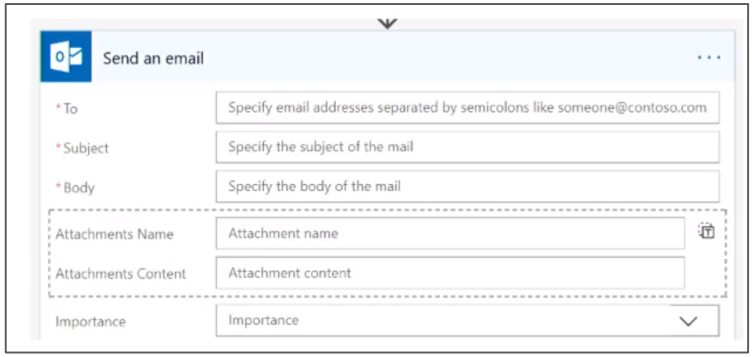


- Create a simple array of strings, numbers, Booleans, etc
  - Useful for creating simple list (e.g. email addresses)

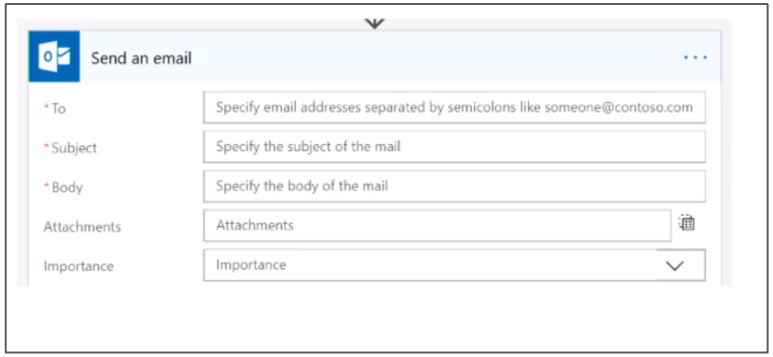


Passing Data into Arrays

- Two options to pass data to an array
  - Hardcode list into an action
  - Create list dynamically at runtime
- Fill in each property of object and get one item created



- Need to pass in an Array that exactly matches the fields



Checking List Containment or Duplicates

- You can use contains() to check if something is in array
- You can use union() with array twice to get unique items

Agenda

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ✓ Converting Between Types
- ✓ Working with Arrays
- ➢ Advanced Techniques

Building object directly

- You can use **Compose** action to create object or array

Compose

* Inputs

```
{
  "ratpack" : [
    {
      "firstName": "Frank",
      "lastName": "Sinatra",
    },
    {
      "firstName": "Dean",
      "lastName": "Martin",
    },
    {
      "firstName": "Sammy",
      "lastName": "Davis Jr",
    }
  ]
}
]
```

Add dynamic content

Calling Flows using the HTTP action

In the parent workflow:

HTTP

* Method: POST

* Uri: 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=vYgMxpt4ArsyNbXFxnZMFeZcgoGx2KqW-ttAKJr3fGQ

Headers: Enter key | Enter value

Body:
```
{
"message" : "message"
}
```

Add dynamic content

Show advanced options

In the child workflow:

Request

HTTP POST URL: https://prod-56.westus.logic.azure.com:443/workflows/419593519f75...

Request Body JSON Schema

Example:
```
{
    "type": "object",
    "properties": {
        "address": {
            "type": "string"
        }
    },
    "required": ["address"]
}
```

Use sample payload to generate schema

Show advanced options

Response

* Status Code: 200

Headers: Enter key | Enter value

Body:
```
{
    "message": "Success!"
}
```

Add dynamic content

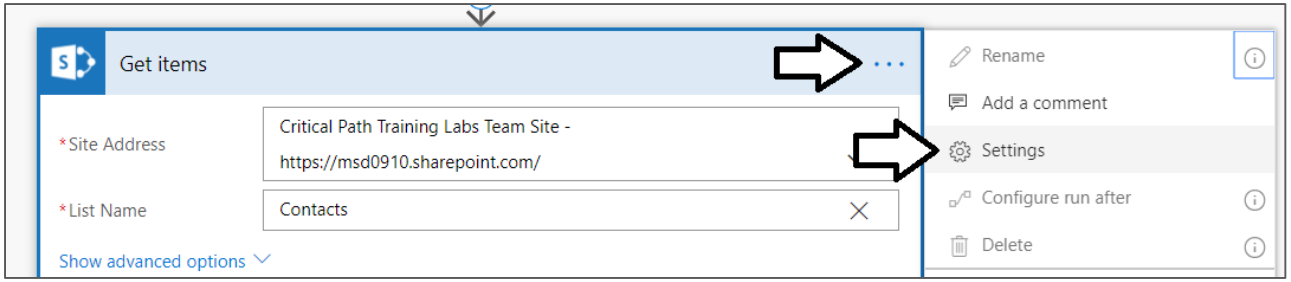
Normal action execution

- Standard behavior of a flow
  - Action steps execute in sequential order
  - Flow terminates if error occurs (failure or timeout)

- After flow runs, every action is in 1 of 4 possible states

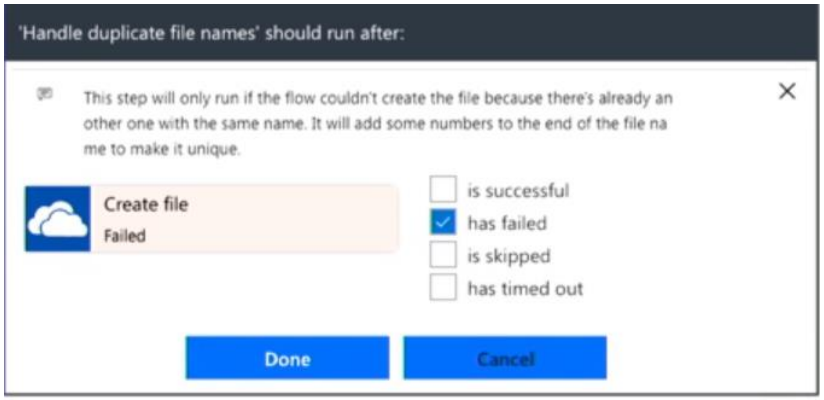
Action Settings

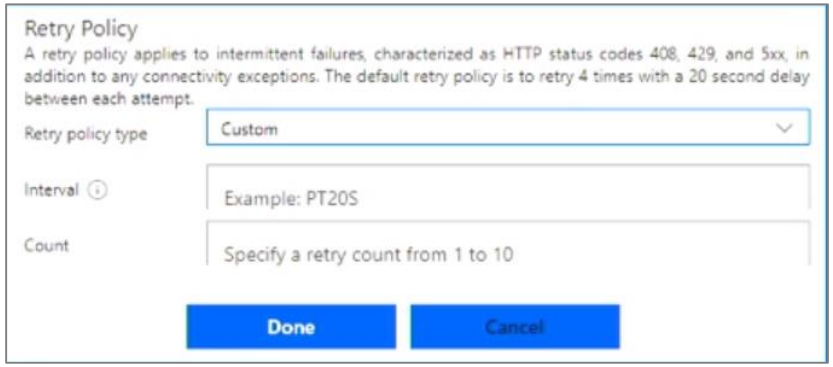
- Settings let you configure
  - Async Actions
  - Timeouts
  - Retry Policy
  - Sequential Behavior
  - And more!



Error Handling

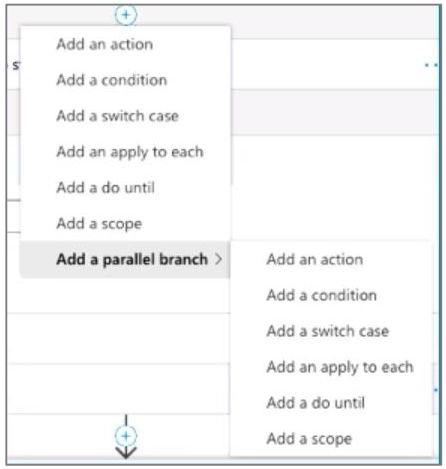
- Select the **Run after** option from action menu
  - Choose which error conditions, the arrow will turn dotted red
  - Use parallels for errors that are not at end of flow
  - Retry policy by default handles transient failures
  - Recommended to select exponential as they last a long time

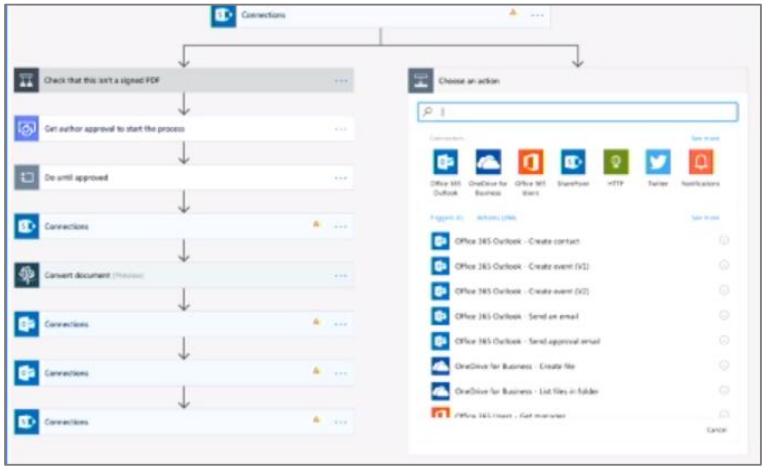




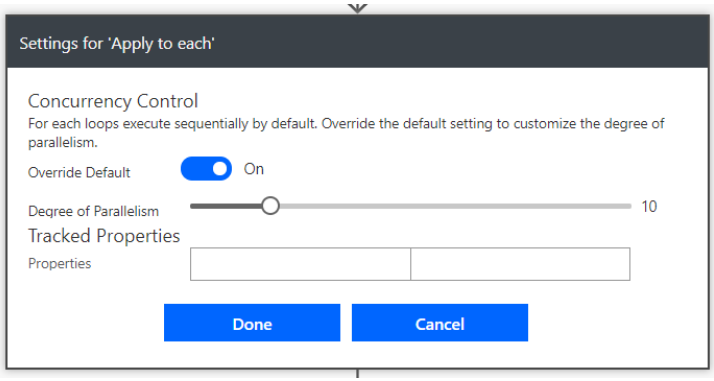
Parallel Execution

- Add parallel branch from above using ⊕



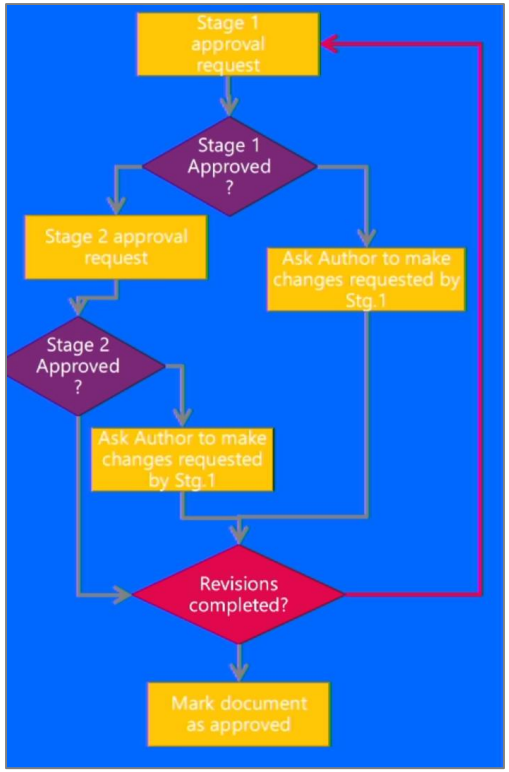


- Apply to each is sequential by default
  - Adding parallel execute to Apply to each



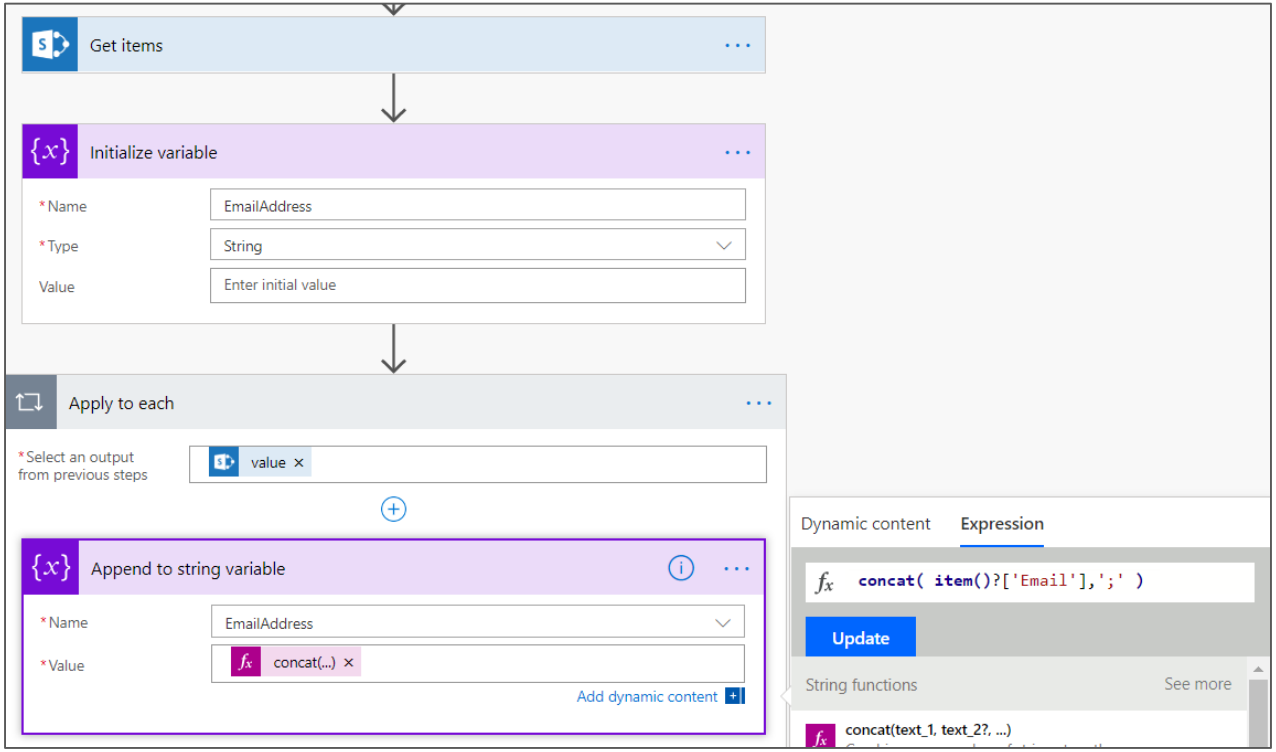
Looping

- Do until actions
  - Provides loop until a condition becomes true
  - Useful when implementing approval process



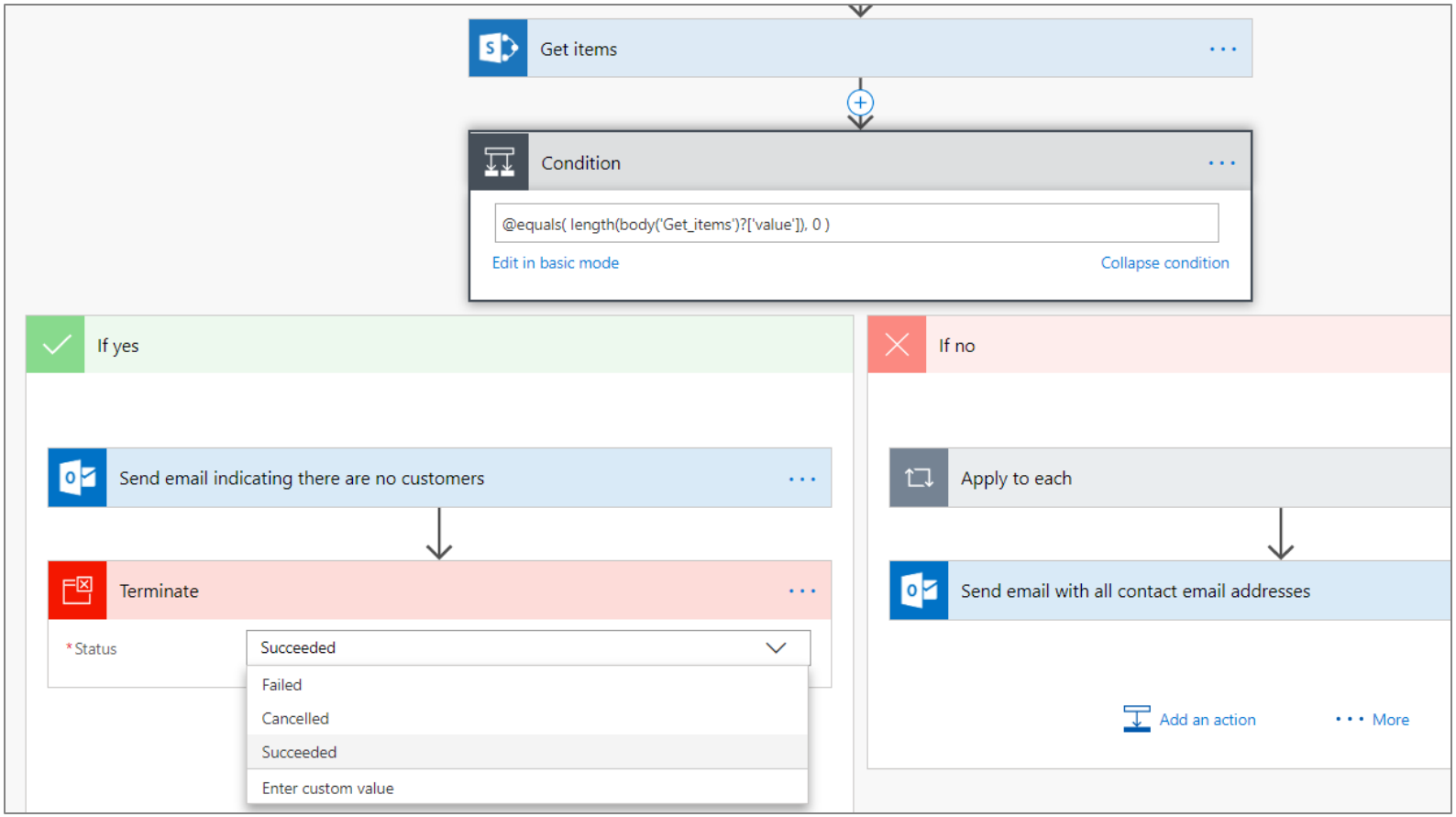
Variables

- Using variables
  - Always define the variable type and initialize it
  - Used for counters and Boolean conditions
  - Useful to append content in to strings & arrays



Terminate action

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



Summary

- ✓ Flow Fundamentals
- ✓ Writing Flow Expressions
- ✓ Converting Between Types
- ✓ Working with Arrays
- ✓ Advanced Techniques

Deep Dive into PowerApps and Flow

- Two action-packed days of building PowerApps and Flows
  1. Getting Started with PowerApps Studio
  2. Designing PowerApps using Advanced Techniques
  3. Building PowerApps for SharePoint Online
  4. Introduction to Microsoft Flow
  5. Designing Flows to Automate an Approval Process
  6. Building PowerApps and Flows for Power BI
  7. Working with the Common Data Service for Apps
  8. Managing Application Lifecycle with PowerApps and Flow

- More info
  - https://www.criticalpathtraining.com
  - info@criticalpathtraining.com

CriticalPath TRAINING