

# 21st Century Background Services with Azure Logic Apps and Azure Functions

Kevin Griffin

@1kevgriff

# About Me



Kevin Griffin

10x Microsoft MVP

ASP.NET Core, Azure, and Web

[consultwithgriff.com](http://consultwithgriff.com)

[twitter.com/1kevgriff](https://twitter.com/1kevgriff)

[twitch.com/1kevgriff](https://twitch.com/1kevgriff)





Welcome to the Shedquarters

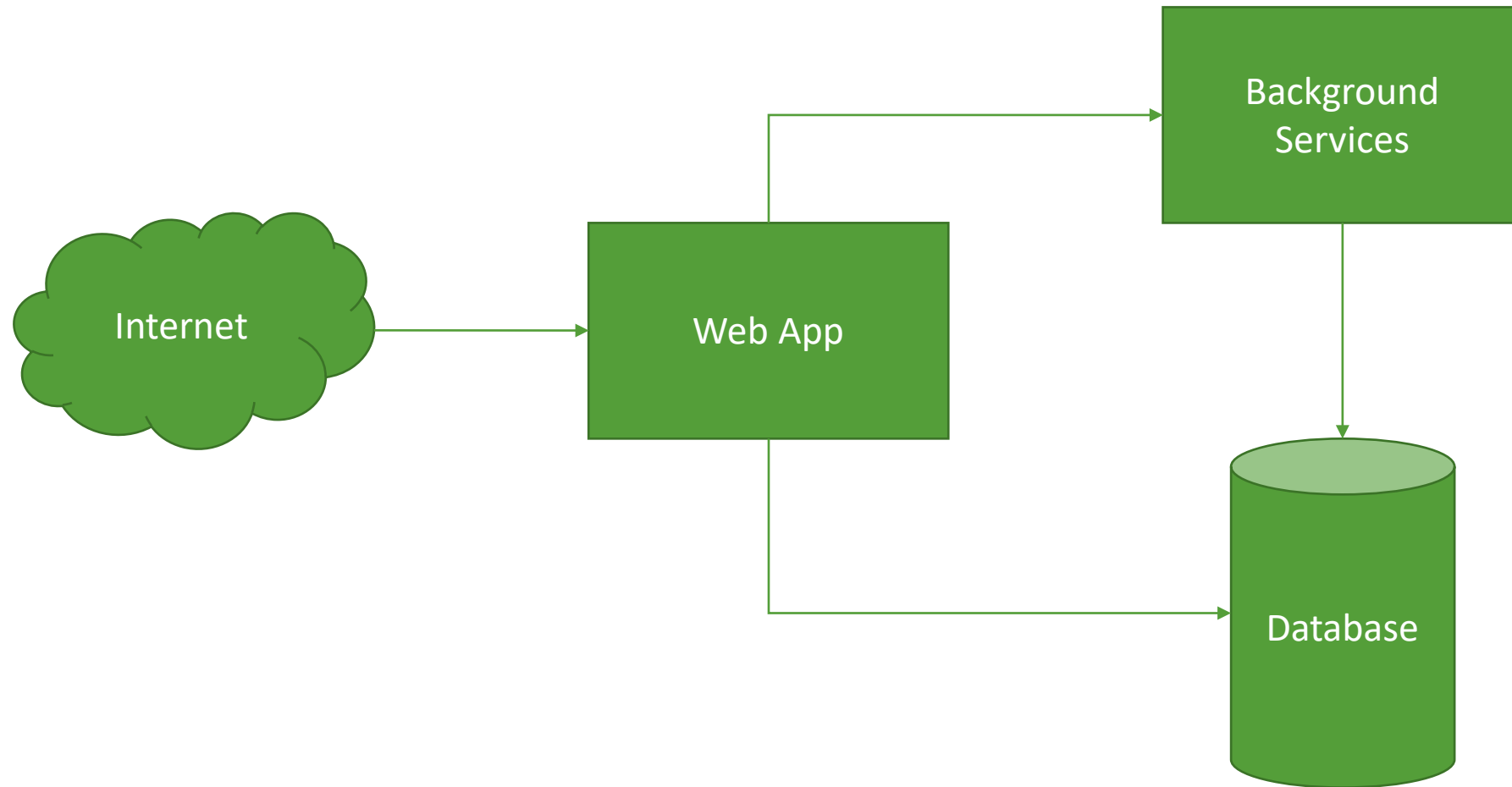
Say nice things about me on [twitter.com/likeygriff](#)



All details for THIS talk at:

<https://consultwithgriff.com/21st-century/>

# Background Services



# Background Services



Background Services

- Sending Email
- Processing Payments
- Cleaning up database/filesystem
- Recurring tasks
- And More!

# Background Services

## Hangfire

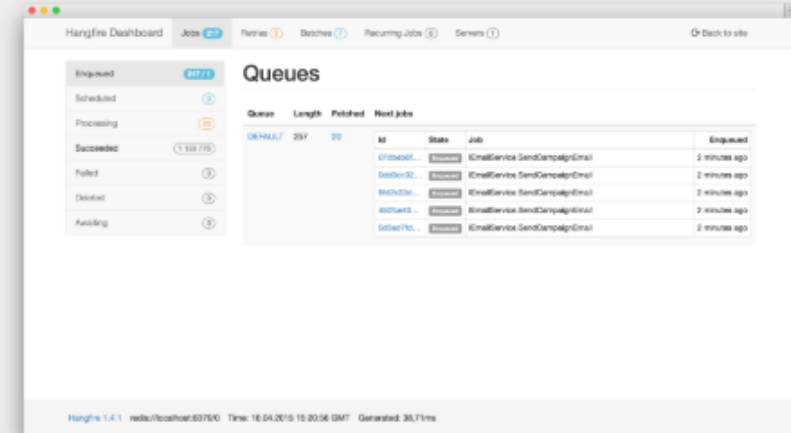
An easy way to perform background processing in .NET and .NET Core applications. No Windows Service or separate process required.

Backed by persistent storage. Open and free for commercial use.

release **v1.6.29**

downloads **17M**

Stars **5.7k**



# Background Services

## Background tasks with hosted services in ASP.NET Core

02/10/2020 • 14 minutes to read •  +2

### In this article

[Worker Service template](#)

[Package](#)

[IHostedService interface](#)

[BackgroundService base class](#)

[Timed background tasks](#)

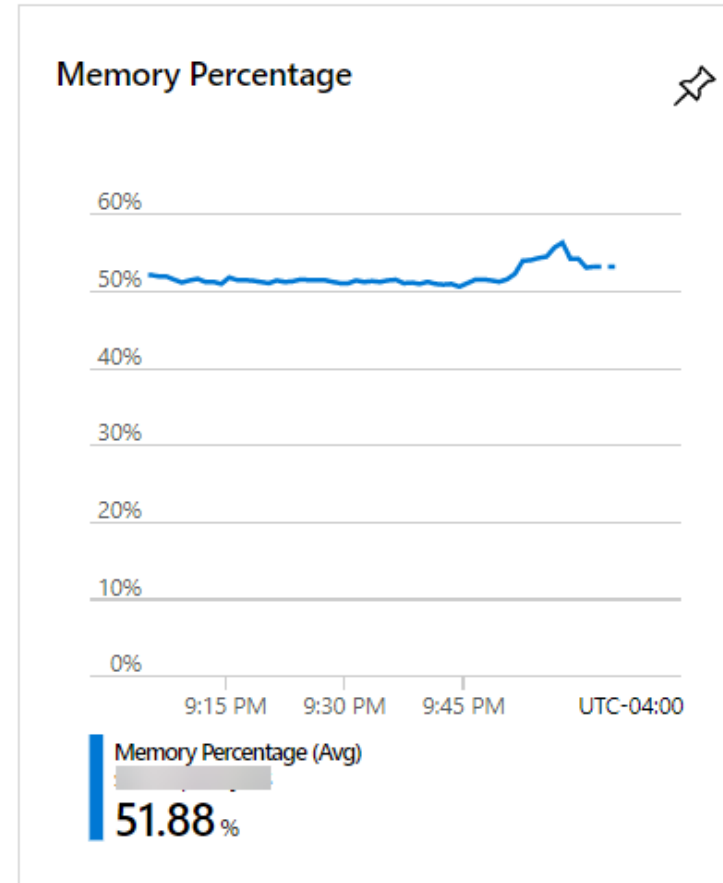
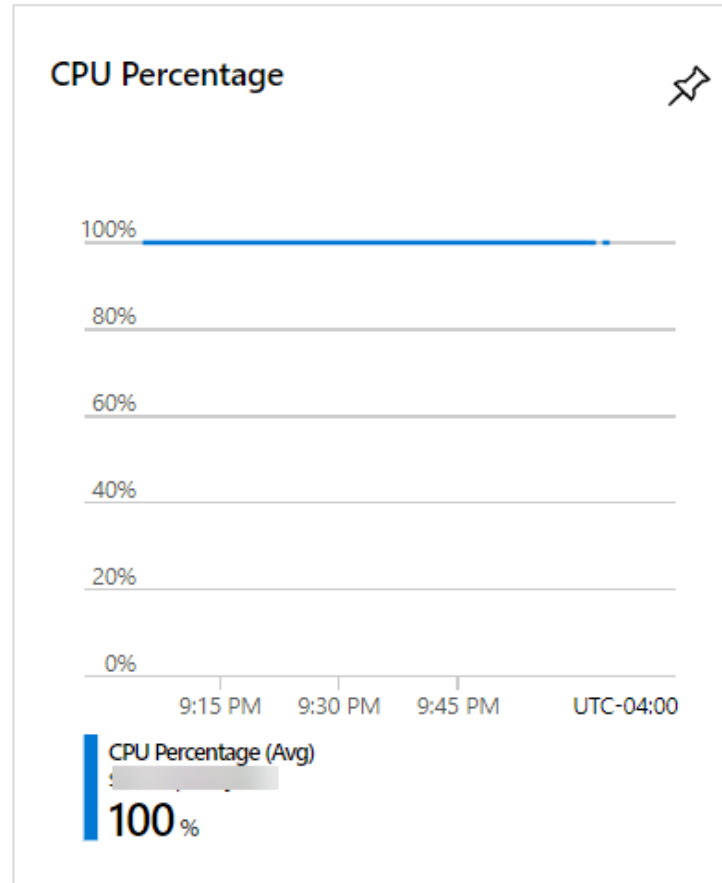
[Consuming a scoped service in a background task](#)

[Queued background tasks](#)

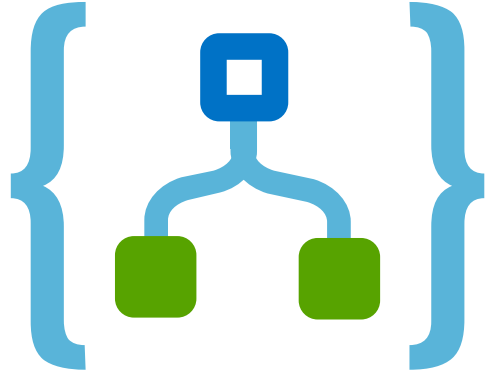
[Additional resources](#)



# Background Services

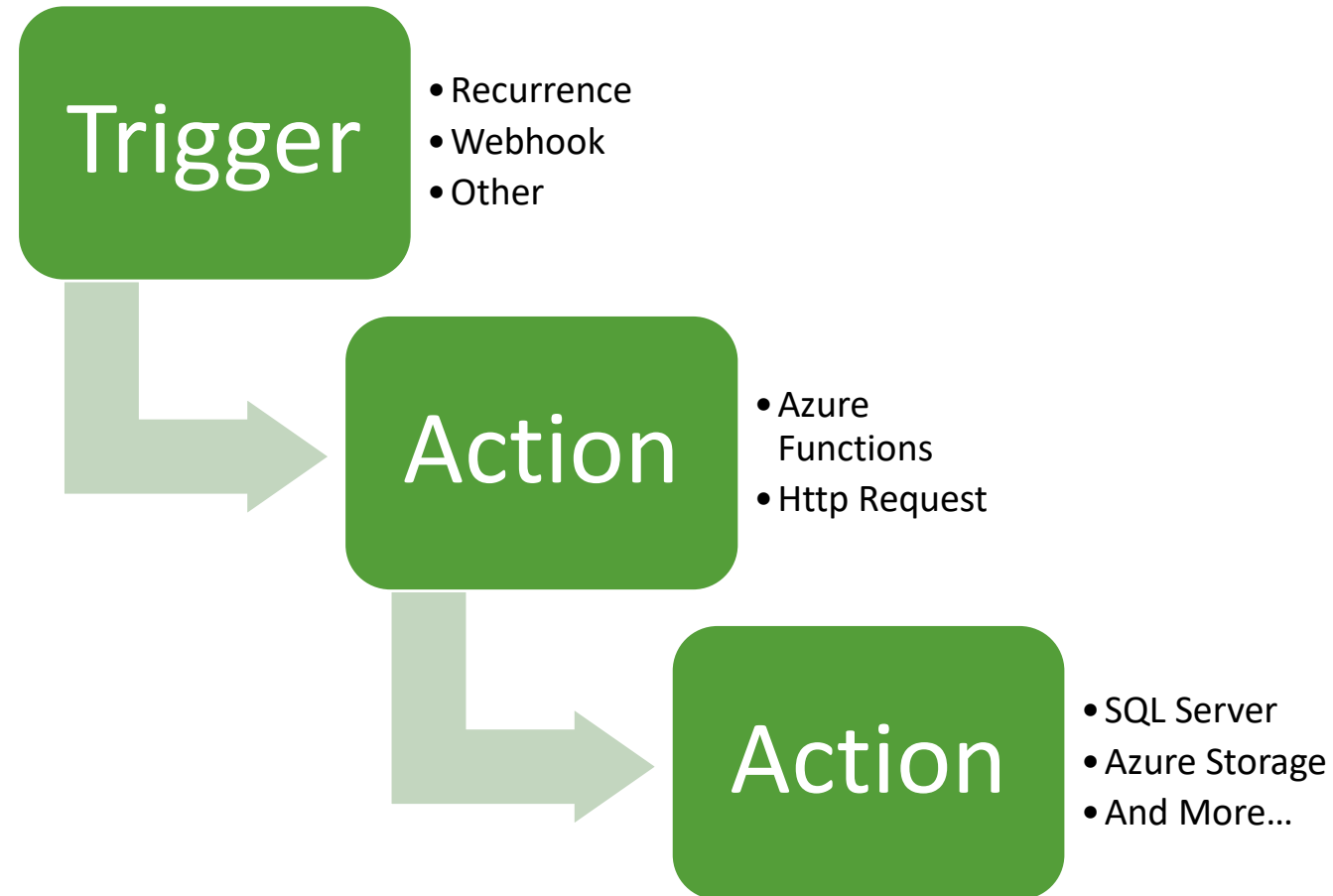


# Logic Apps



- Zapier or IFTTT for Developers and Business People
- Task oriented
- Low/No Code

# Logic Apps



# Consumption Based Pricing

Runs history



All

▼

Start time earlier than

▼

Pick a date

📅

Pick a time

🕒

Specify the run identifier to open monitor view directly

➔

Status	Start time	Identifier	Duration	Static Results
✔ Succeeded	5/12/2020, 3:00 AM	08586123400853071204422666722CU106	1 Seconds	
✔ Succeeded	5/11/2020, 3:00 AM	08586124264849485192013622591CU82	747 Milliseconds	
✔ Succeeded	5/10/2020, 2:59 AM	08586125128854935268132700819CU111	995 Milliseconds	
✔ Succeeded	5/9/2020, 3:00 AM	08586125992849812342315621400CU18	857 Milliseconds	
✔ Succeeded	5/8/2020, 3:00 AM	08586126856851585491642064090CU16	4 Seconds	
✔ Succeeded	5/7/2020, 3:00 AM	08586127720848185263530956377CU87	891 Milliseconds	
✔ Succeeded	5/6/2020, 2:59 AM	08586128584856705792364408982CU106	801 Milliseconds	
✔ Succeeded	5/5/2020, 8:47 AM	08586129240156197958951512827CU16	408 Milliseconds	



# Demo

Touring the Azure Logic Apps Editor

# Expression Editor

The screenshot shows the 'Expression Editor' for a workflow named 'Detect Sentiment (V2) (Preview)'. The editor is divided into three sections: 'documents Id - 1', 'documents Text - 1', and 'documents Language - 1'. Each section has a text input field with a Twitter icon and a close button. The 'documents Text - 1' field is currently selected. Below these sections is a '+ Add new item' button. At the bottom of the editor, it says 'Connected to devtalkcog. Change connection.' and there is a '+ Add an action' button. A dropdown menu is open, showing 'Add dynamic content from the apps and connectors' and 'Hide used in this flow.' options. The menu also has tabs for 'Dynamic content' and 'Expression'. Under 'Dynamic content', there is a search bar and a list of dynamic content items: 'When a new tweet is posted', 'Created at', 'Description', 'in\_reply\_to\_user\_id', and 'Location'. Each item has a Twitter icon and a description. The 'Created at' item is highlighted.

Detect Sentiment (V2) (Preview)

documents Id - 1

Tweet id x

Add dynamic content +

documents Text - 1

Tweet text x

documents Language - 1

en

+ Add new item

Connected to devtalkcog. Change connection.

+ Add an action

Add dynamic content from the apps and connectors Hide used in this flow.

Dynamic content Expression

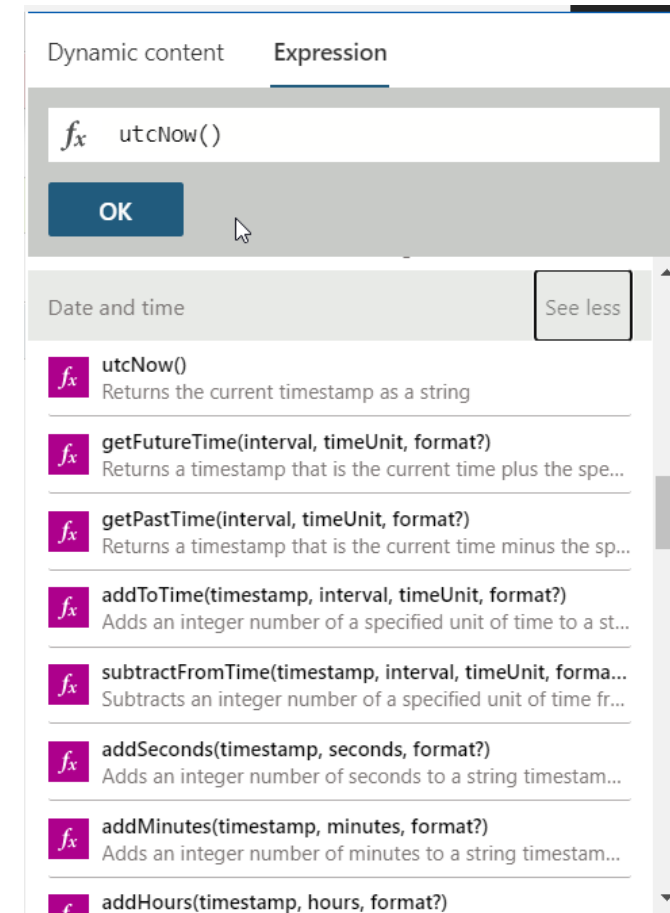
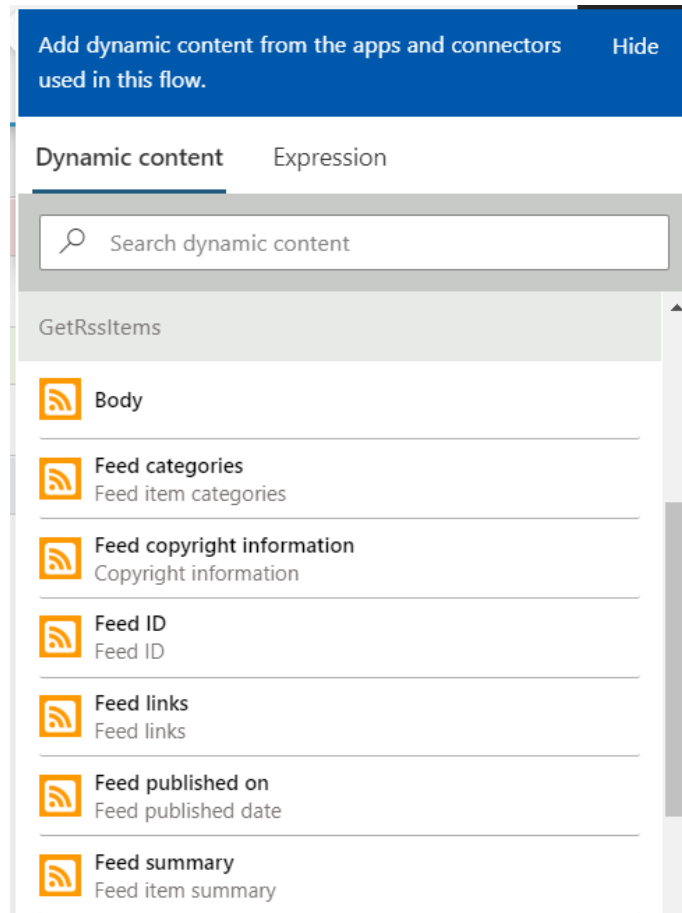
Search dynamic content

When a new tweet is posted See more

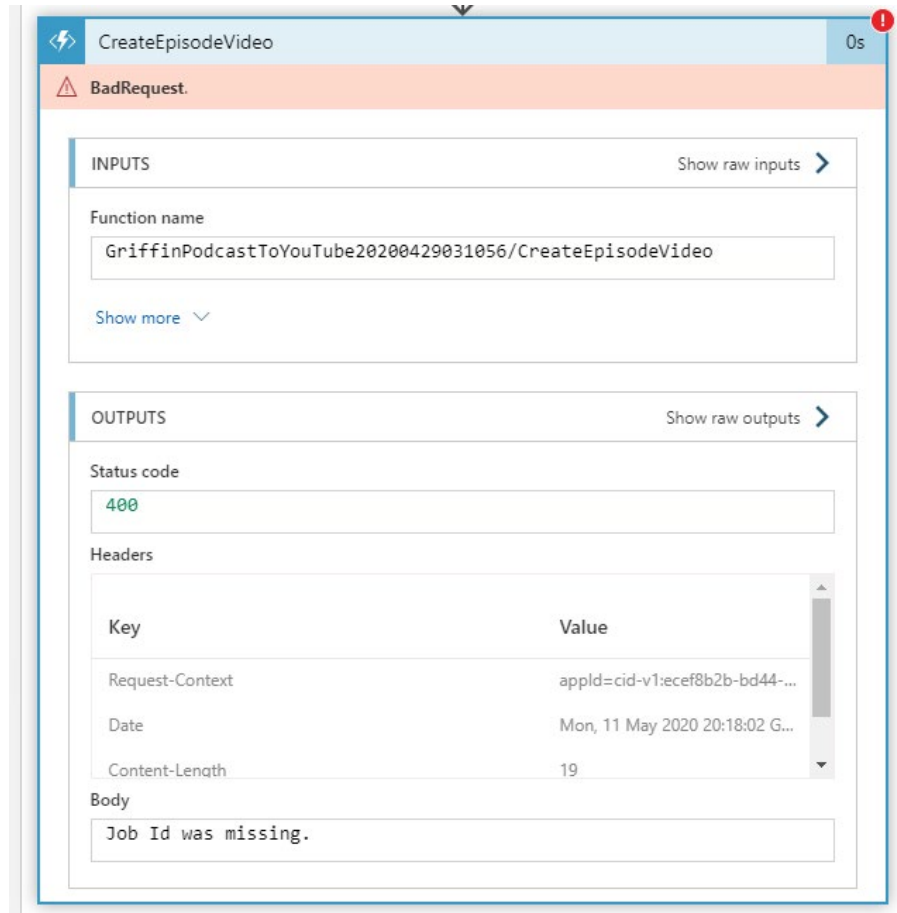
- Created at  
Time at which the tweet was posted
- Description  
User description
- in\_reply\_to\_user\_id  
User Id of the author of the tweet that the current tweet i...
- Location  
Location of the user

Say nice things about me on [twitter.com/1kevgriff](https://twitter.com/1kevgriff)

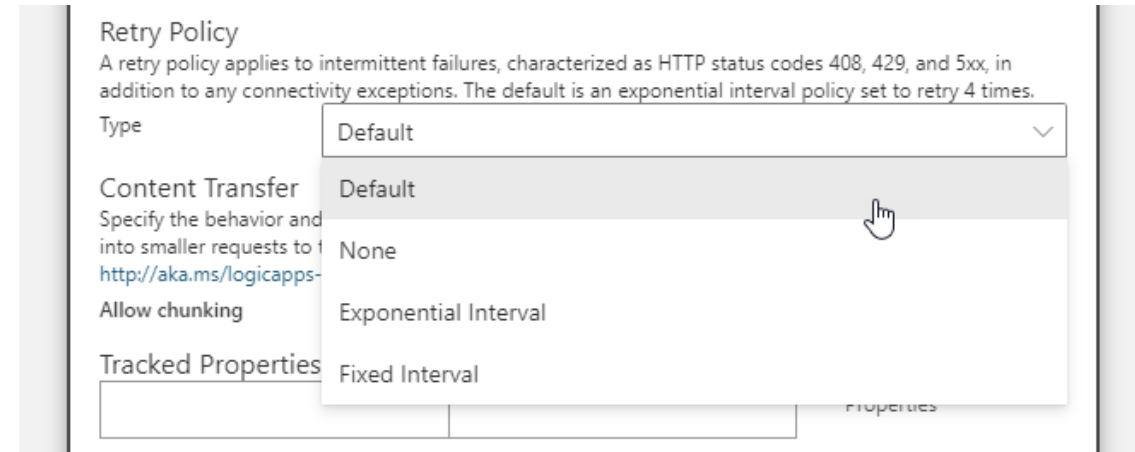
# Expression Editor



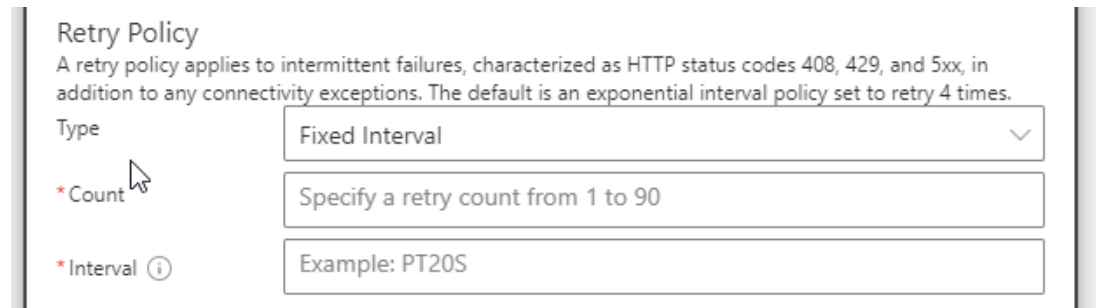
# Managing Dependency Errors



The screenshot shows the Azure Functions portal interface for a function named 'CreateEpisodeVideo'. At the top, there is a red banner indicating a 'Bad Request' error. Below this, the 'INPUTS' section shows the function name as 'GriffinPodcastToYouTube20200429031056/CreateEpisodeVideo'. The 'OUTPUTS' section shows a status code of '400' and a message in the body: 'Job Id was missing.' The headers section is also visible, showing request context, date, and content length.



The screenshot shows the 'Retry Policy' configuration interface. It includes a description: '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.' The 'Type' dropdown menu is open, showing options: 'Default', 'None', 'Exponential Interval', and 'Fixed Interval'. A mouse cursor is pointing at the 'Default' option.




The screenshot shows the 'Retry Policy' configuration form with 'Fixed Interval' selected in the 'Type' dropdown. It includes a description: '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.' The 'Count' field is set to 'Specify a retry count from 1 to 90' and the 'Interval' field is set to 'Example: PT20S'.



# Scenario 1

Processing Incoming Email

# Scenario 1: Processing Incoming Email






 When a new email arrives ...

Label	Inbox	✓	✕
Subject	Yesterday's Top 25		✕
Importance	All	✓	✕
Starred	All	✓	✕
Has Attachments	No	✓	✕
Include Attachments	Yes	✓	✕

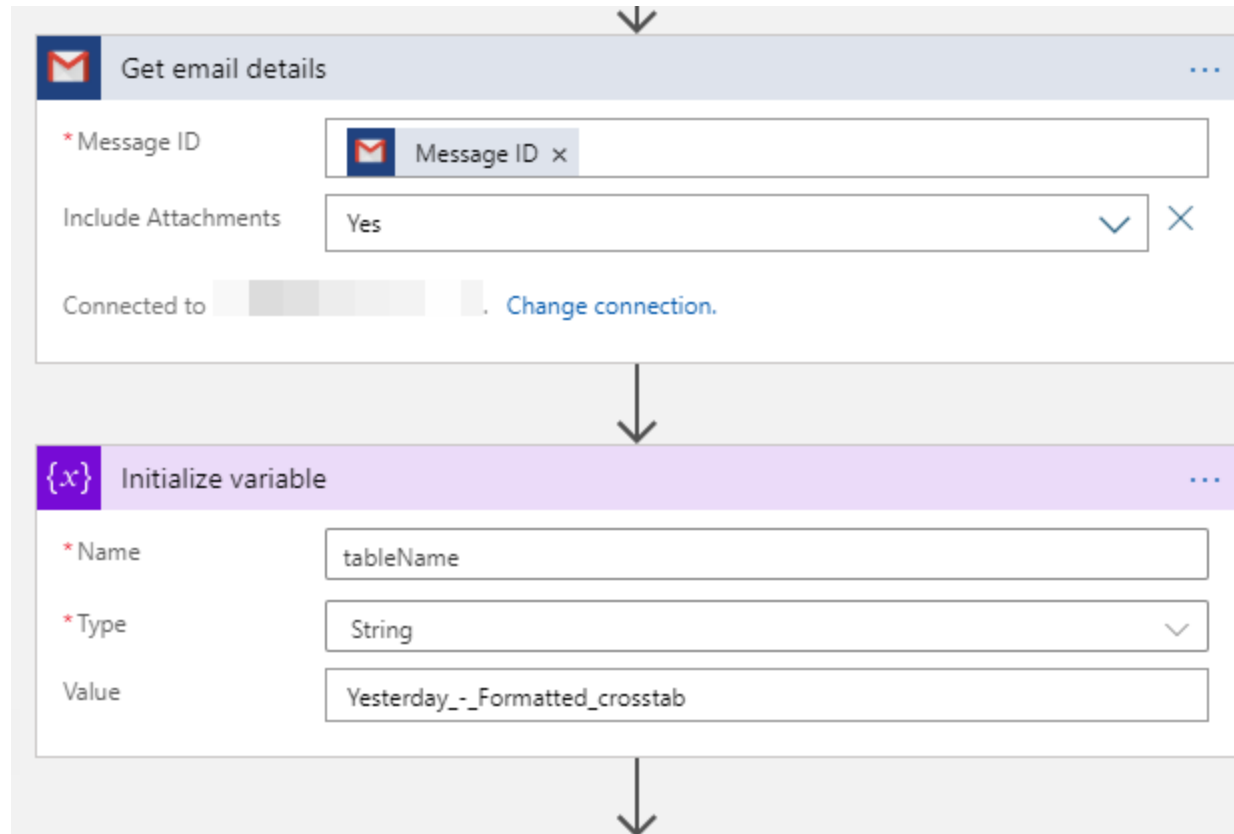
How often do you want to check for items?

15	Minute	✓
----	--------	---

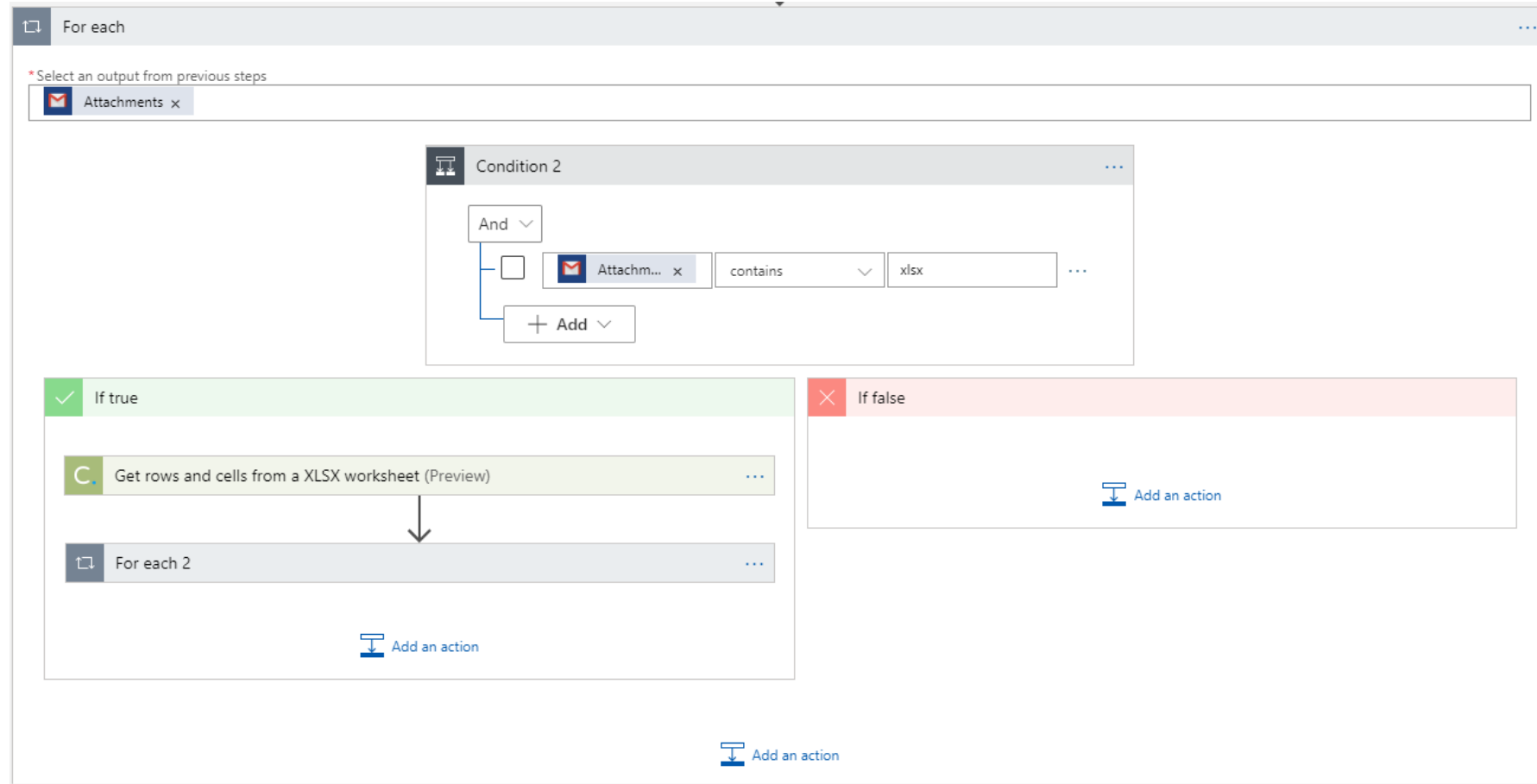
Add new parameter ✓

Connected to      [Change connection.](#)

# Scenario 1: Processing Incoming Email

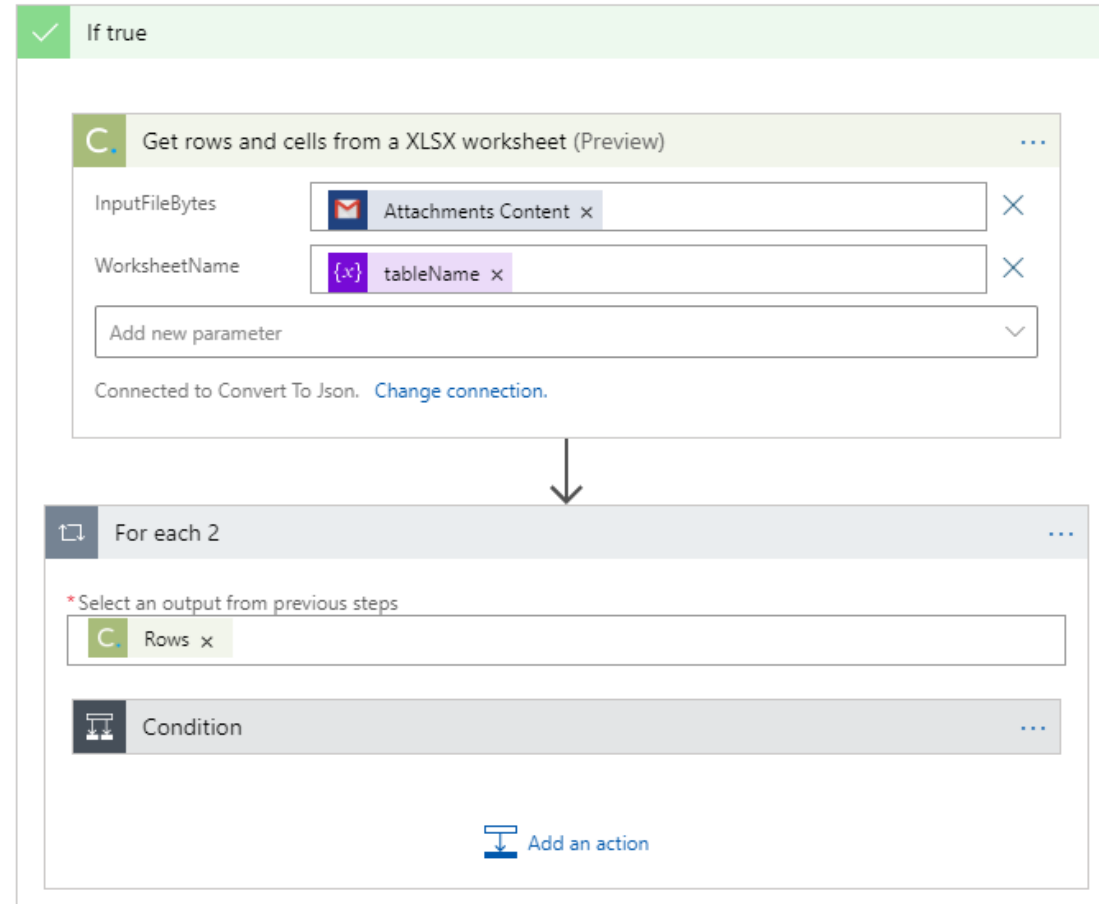


# Scenario 1: Processing Incoming Email

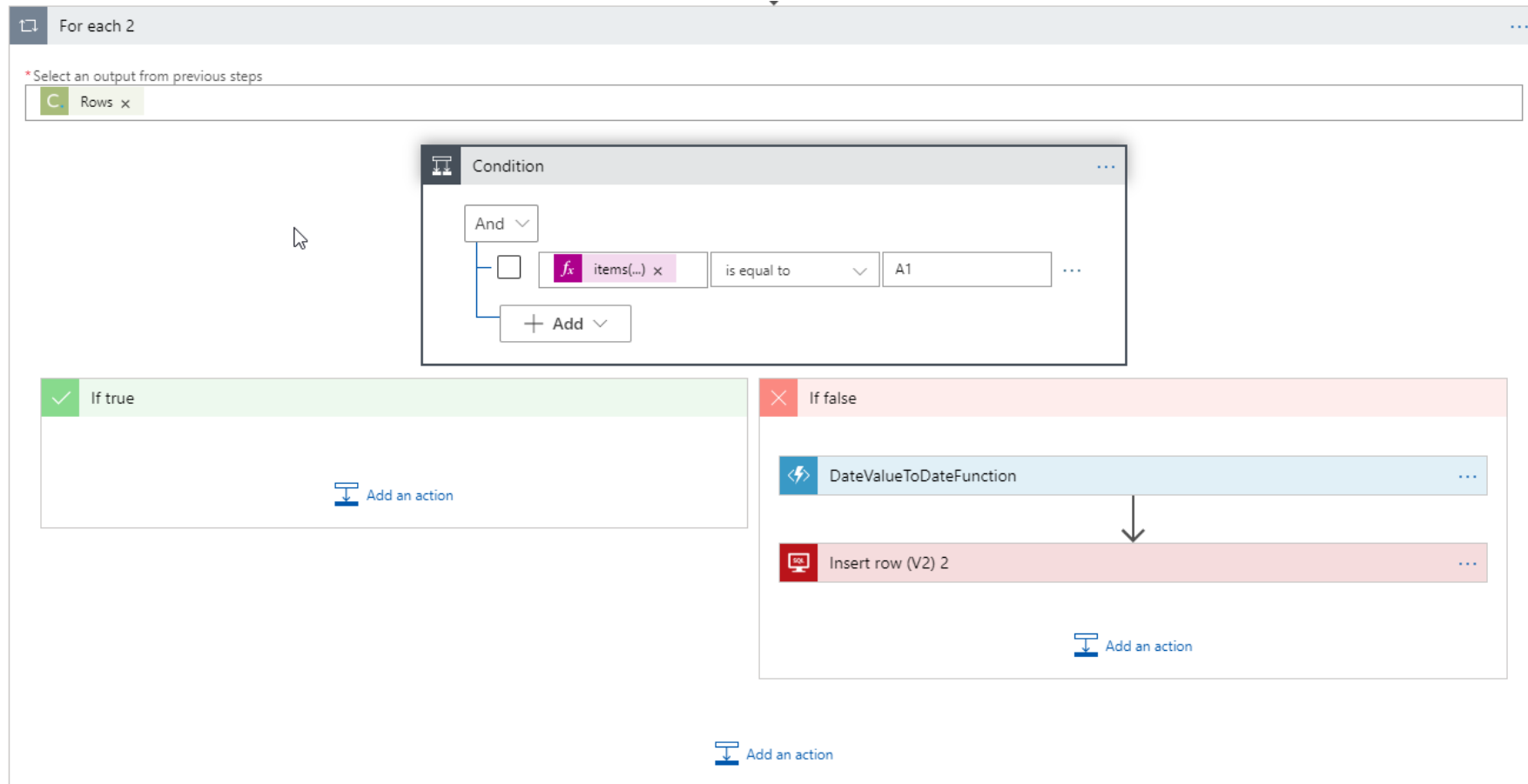






# Scenario 1: Processing Incoming Email



# Scenario 1: Processing Incoming Email



# Scenario 1: Processing Incoming Email


 DateValueToDateFunction 

Request Body


Context object to be passed to function: { .. }

Queries

```
{  
  "dateValue": "fx items(...) x"  
}
```



Add new parameter





# Scenario 1: Processing Incoming Email

Insert row (V2) 2

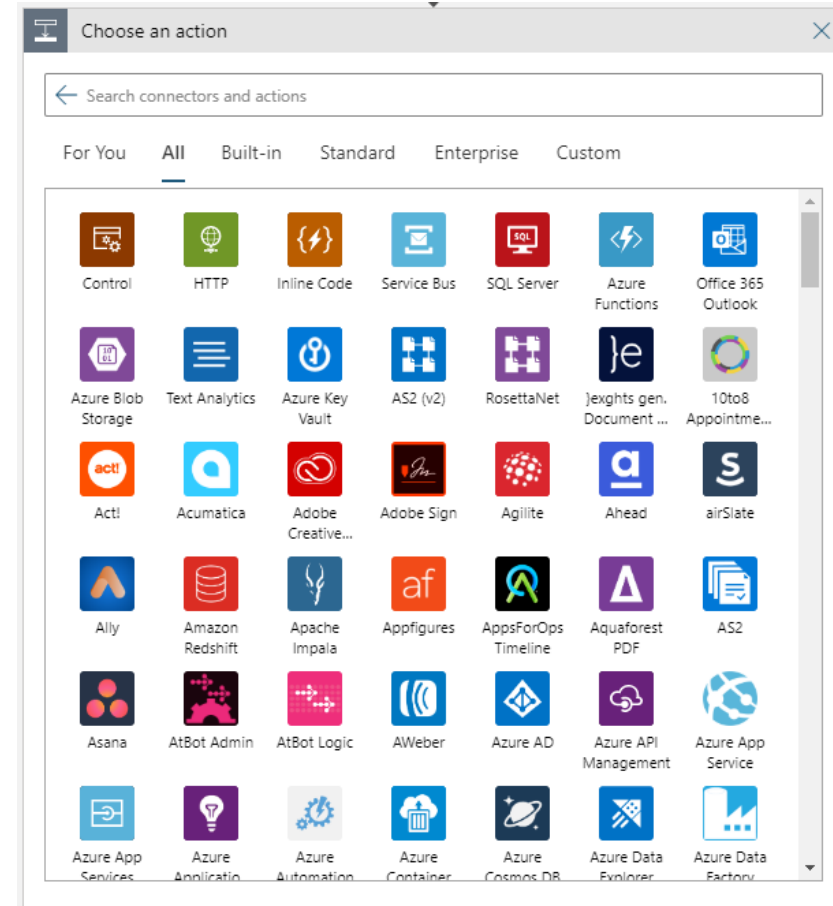
* Server name	Use connection settings ( )
* Database name	Use connection settings ( e)
* Table name	Top25
* EmailSubjectLine	Subject x
* EmailDate	Received Date-Time x
* FileName	Attachments Name x
* SalesRank	fx items(...) x
* EventName	fx items(...) x
* VenueName	fx items(...) x
* EventDate	Body x
* AvgTicketPrice	fx items(...) x

Connected to . Change connection.

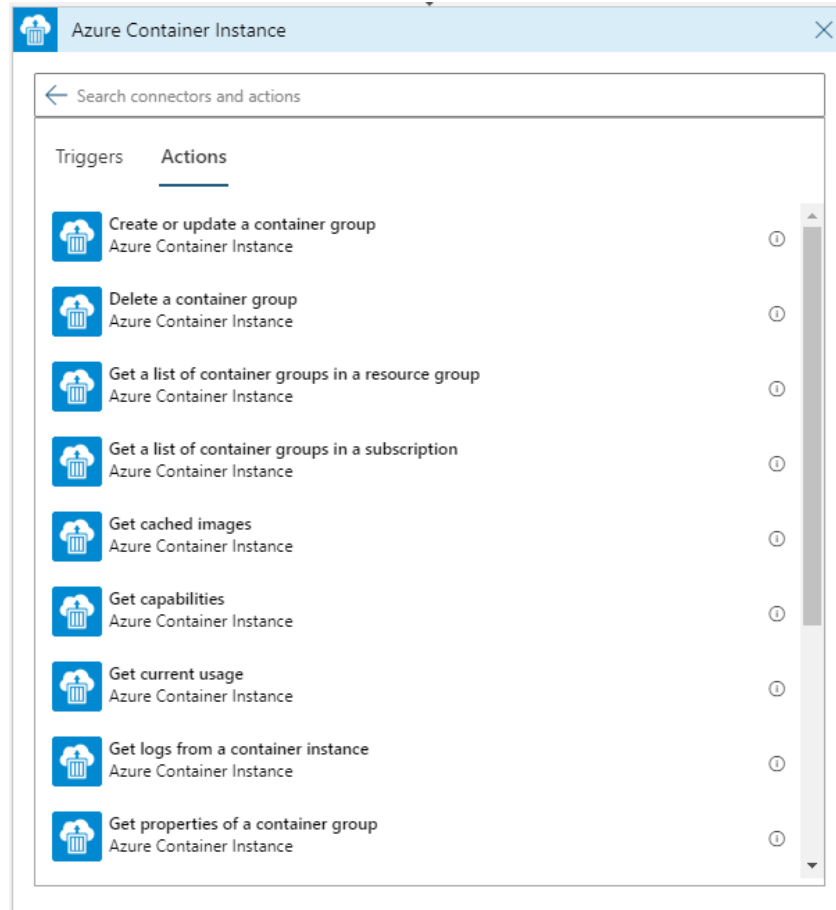


# Connectors

- Hundreds of connectors to external services
- Authorization is stored outside of Azure Logic Apps
- Reusable
- You can write custom connectors as well!



# Connectors



- Each connector has multiple triggers and actions

# Azure Functions as a Fallback



- Logic Apps does little on its own.
- Azure Functions are an amazing way to insert missing functionality.

**“If only I could write a line of code”**

# Azure Functions Example

```
[FunctionName("DateValueToDateFunction")]
0 references | Kevin Griffin, 26 days ago | 1 author, 2 changes
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
    ILogger log)
{
    log.LogInformation("DateValue to Date Function Running");

    if (!req.Query.ContainsKey("dateValue"))
    {
        return new BadRequestErrorMessageResult("Missing query string key 'dateValue'");
    }

    string name = req.Query["dateValue"];

    if (!double.TryParse(name, out var asDouble))
        return new BadRequestErrorMessageResult($"The input 'dateValue':'{name}' was invalid");

    var newDate = DateTime.Parse("1900-01-01").AddDays(asDouble);
    return new OkObjectResult(newDate);
}
```

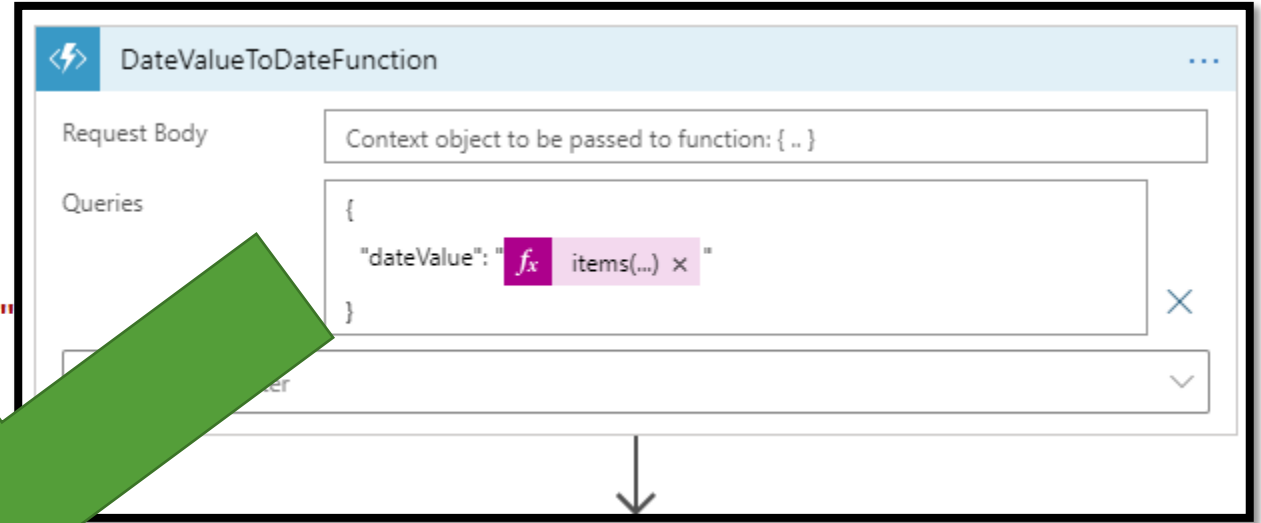
Say nice things about me on [twitter.com/1kevgriff](https://twitter.com/1kevgriff)

# Azure Functions Example

```
[FunctionName("DateValueToDateFunction")]
```

0 references | Kevin Griffin, 26 days ago | 1 author, 2 changes

```
public static async Task<IActionResult> Run(  
    [HttpTrigger(AuthorizationLevel.Function, "get",  
        ILogger log)  
    ]  
{  
    log.LogInformation("DateValue to Date Function");  
  
    if (!req.Query.ContainsKey("dateValue"))  
    {  
        return new BadRequestErrorMessageResult("Missing query string key 'dateValue'");  
    }  
  
    string name = req.Query["dateValue"];  
  
    if (!double.TryParse(name, out var asDouble))  
        return new BadRequestErrorMessageResult($"The input 'dateValue': '{name}' was invalid");  
  
    var newDate = DateTime.Parse("1900-01-01").AddDays(asDouble);  
    return new OkObjectResult(newDate);  
}
```

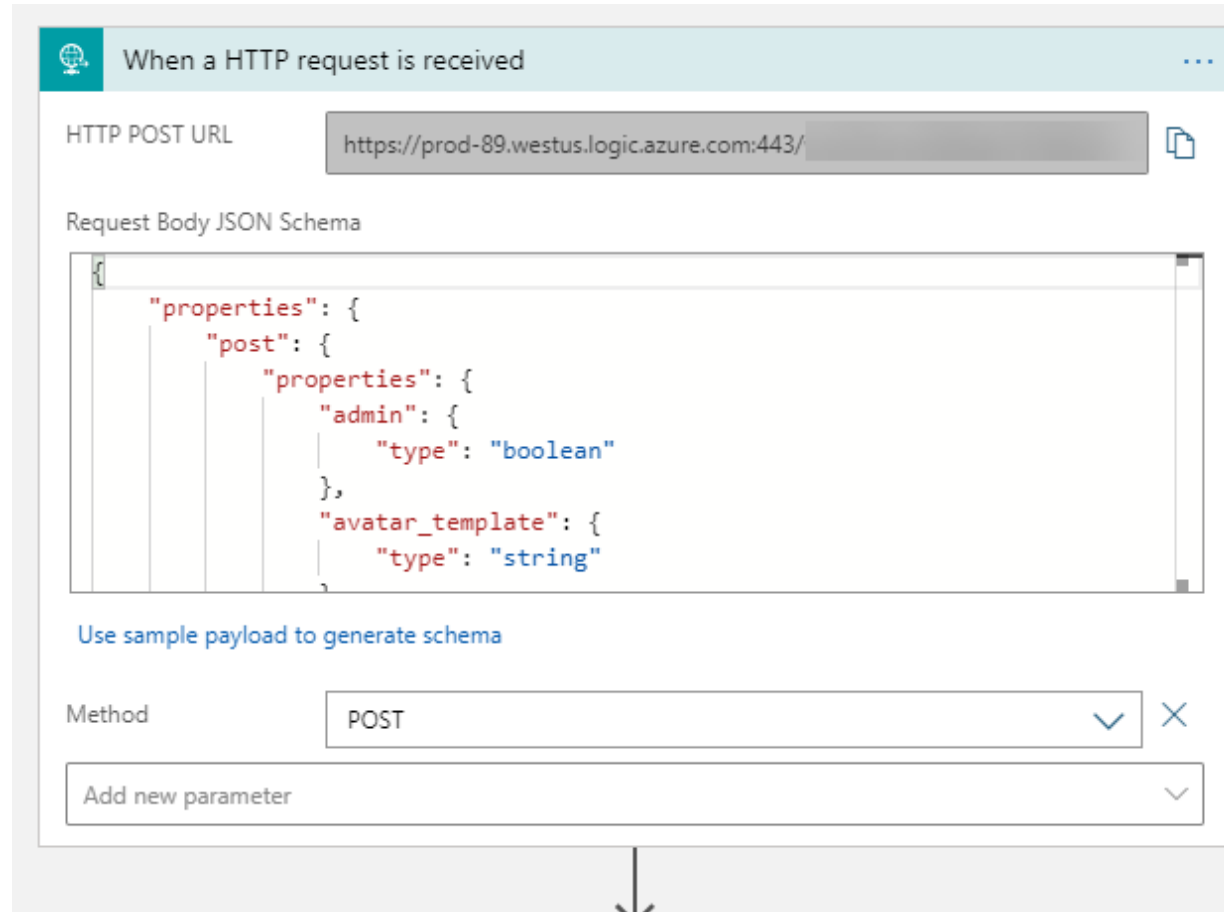


# Scenario 2

Webhooks



# Scenario 2: Webhook Processing



The screenshot shows the configuration for a Logic App trigger named "When a HTTP request is received". The "HTTP POST URL" is set to "https://prod-89.westus.logic.azure.com:443/". The "Request Body JSON Schema" is defined as a JSON object with nested properties: "properties" contains "post", which contains "properties" containing "admin" (type: "boolean") and "avatar\_template" (type: "string"). Below the schema, there is a link "Use sample payload to generate schema". The "Method" is set to "POST". At the bottom, there is a button "Add new parameter". A large downward arrow is positioned below the configuration box.

When a HTTP request is received

HTTP POST URL `https://prod-89.westus.logic.azure.com:443/`

Request Body JSON Schema

```
{
  "properties": {
    "post": {
      "properties": {
        "admin": {
          "type": "boolean"
        },
        "avatar_template": {
          "type": "string"
        }
      }
    }
  }
}
```

[Use sample payload to generate schema](#)

Method POST

Add new parameter

Say nice things about me on [twitter.com/1kevgriff](https://twitter.com/1kevgriff)

# Scenario 2: Webhook Processing

**{x}** Initialize variable ...

\* Name

\* Type

Value

↓

**{x}** Initialize variable 2 ...

\* Name

\* Type

Value

↓

**{x}** Initialize variable 3 ...

\* Name

\* Type

Value

↓

**{x}** Initialize variable 4 ...

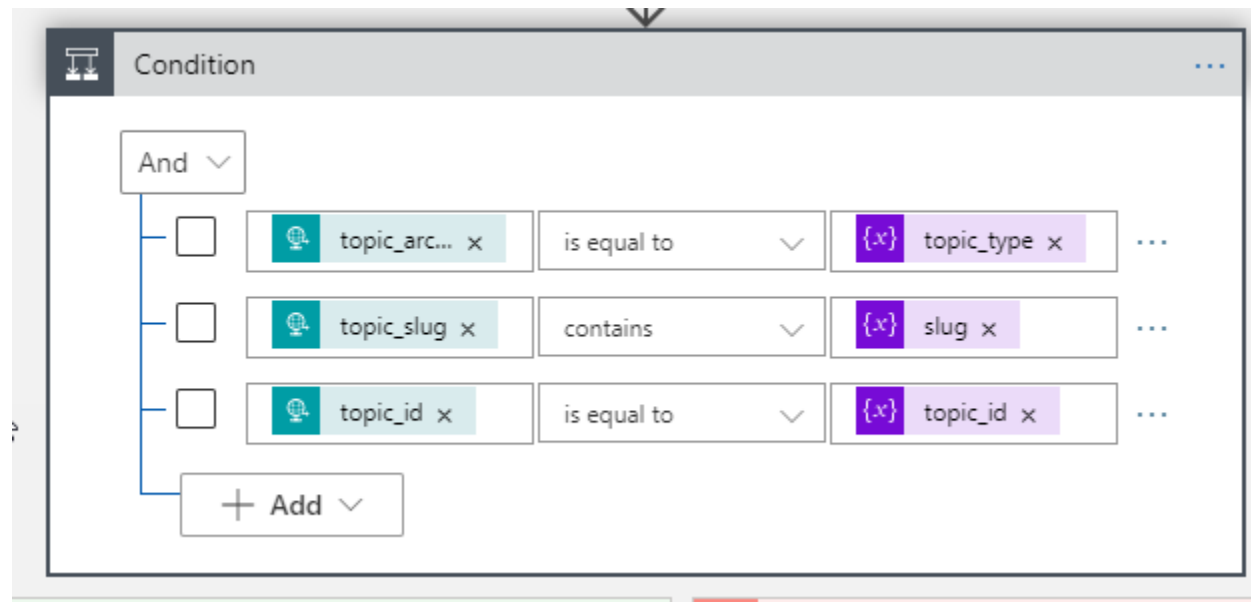
\* Name

\* Type

Value

↓

# Scenario 2: Webhook Processing

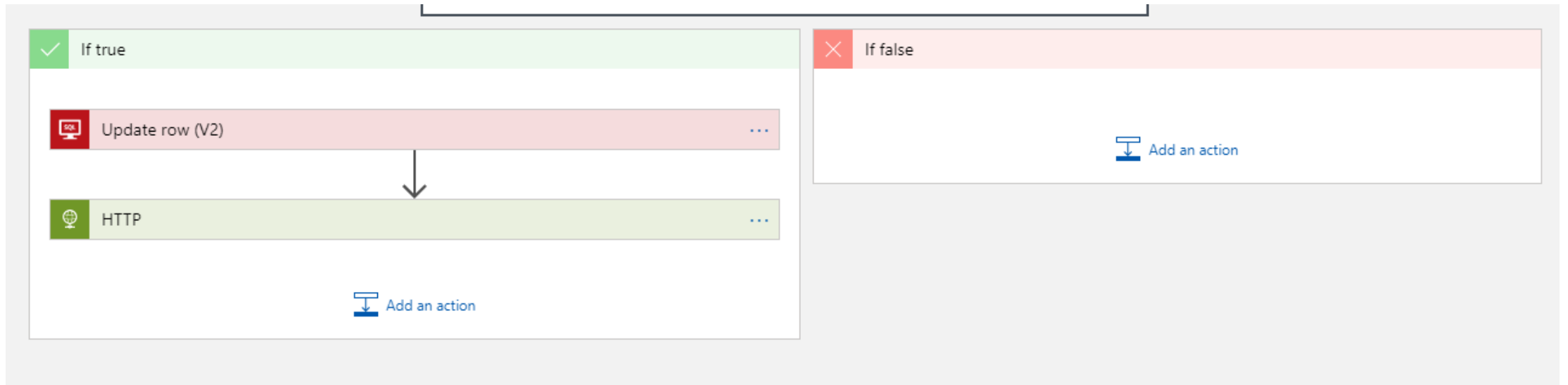


The screenshot shows a 'Condition' configuration window with a header bar containing a tree icon and the title 'Condition'. The main area displays a logical condition structure. At the top, there is a box labeled 'And' with a dropdown arrow. Below it, three conditions are listed, each preceded by an unchecked checkbox and connected to the 'And' box by a vertical line. Each condition consists of a field name, a comparison operator, and a value. The first condition is 'topic\_arc... x' is equal to '{x} topic\_type x'. The second is 'topic\_slug x' contains '{x} slug x'. The third is 'topic\_id x' is equal to '{x} topic\_id x'. Each condition has a three-dot menu to its right. At the bottom of the list is a button labeled '+ Add' with a dropdown arrow.

Condition	Field	Operator	Value
<input type="checkbox"/>	topic_arc... x	is equal to	{x} topic_type x
<input type="checkbox"/>	topic_slug x	contains	{x} slug x
<input type="checkbox"/>	topic_id x	is equal to	{x} topic_id x

+ Add

# Scenario 2: Webhook Processing



# Scenario 2: Webhook Processing

The screenshot displays a workflow editor interface. At the top, a green bar indicates a condition 'If true'. Below this, a red-bordered box highlights the 'Update row (V2)' action. This action is configured with the following fields:

- \* Server name: Use connection settings ( )
- \* Database name: Use connection settings ( )
- \* Table name: SitePreferences
- \* Row id: 2
- PreferenceKey: AA\_LastUpdated
- PreferenceValue: updated\_at

Below the configuration fields, it shows 'Connected to ' and a 'Change connection.' link. An arrow points down from the 'Update row (V2)' action to an 'HTTP' action, which is represented by a green bar. At the bottom of the editor, there is a button labeled 'Add an action'.

# Scenario 2: Webhook Processing

✓ If true

Update row (V2)

↓

HTTP

\* Method: POST

\* URI: {x} Webhook Uri x ?date={f\_x} utcNow() x

Headers: Enter key | Enter value

Queries: key | [blurred value]

Body: Enter request content

Cookie: Enter HTTP cookie

Add new parameter

Add an action

# Code View


```
1 {
2   "definition": {
3     "$schema": "https://schema.management.azure.com/providers/Microsoft.Logic/schemas/2016-06-01/workflowdefinition.json#",
4     "actions": {
5       "Execute_stored_procedure_(V2)": {
6         "inputs": {
7           "body": {},
8           "host": {
9             "connection": {
10              "name": "@parameters('$connections')['sql']['connectionId']"
11            }
12          },
13          "method": "post",
14          "path": "/v2/datasets/@{encodeURIComponent(encodeURIComponent('default'))},@{encodeURIComponent(encodeURIComponent('default'))}/procedures"
15        },
16        "runAfter": {},
17        "type": "ApiConnection"
18      },
19      "Execute_stored_procedure_(V2)_2": {
20        "inputs": {
21          "body": {},
22          "host": {
23            "connection": {
24              "name": "@parameters('$connections')['sql']['connectionId']"
25            }
26          },
27          "method": "post",
28          "path": "/v2/datasets/@{encodeURIComponent(encodeURIComponent('default'))},@{encodeURIComponent(encodeURIComponent('default'))}/procedures"
29        },
```

# Scenario 3

Blob Cleanup



# Scenario 3: Blob Cleanup

 Recurrence ...

\* Interval

1

\* Frequency

Week

▼

On these days

Sunday

▼

×

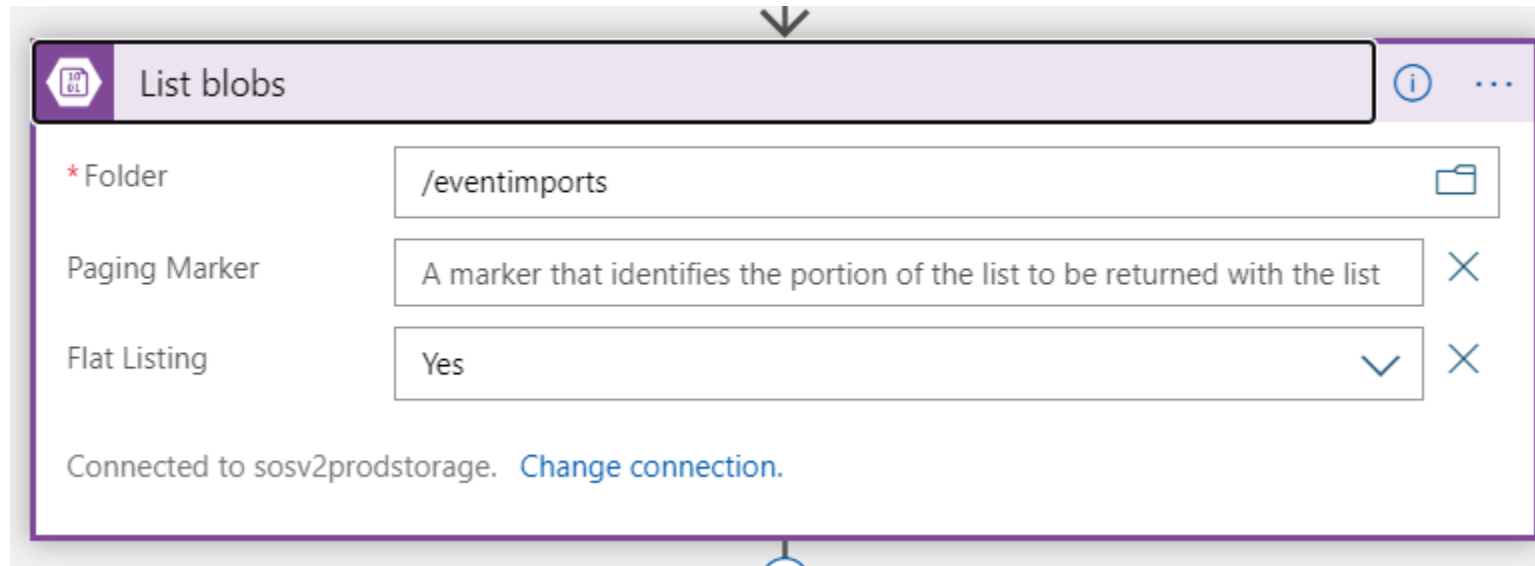
**Preview**

Runs on Sunday every week.





Add new parameter

▼

# Scenario 3: Blob Cleanup

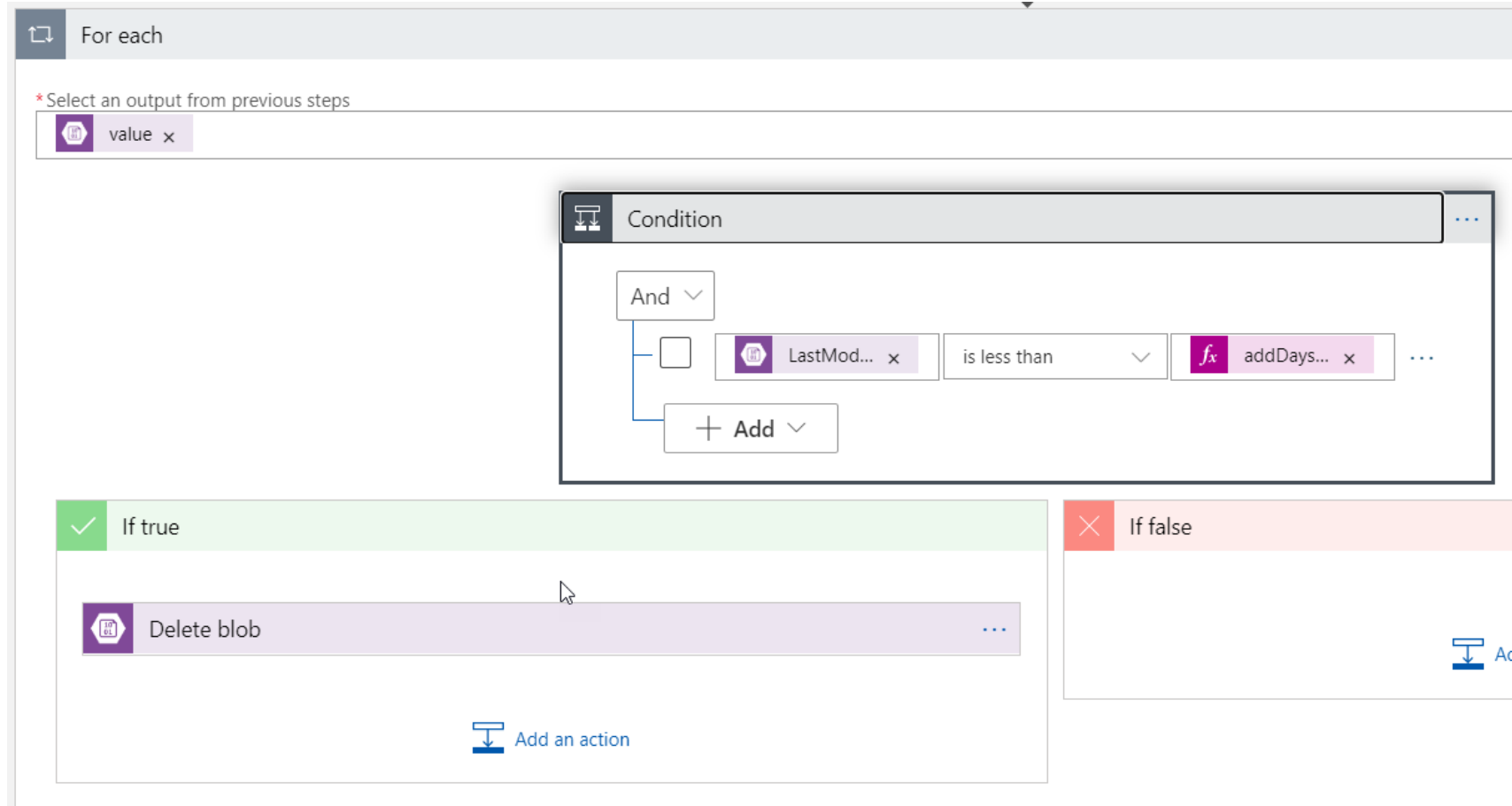


The screenshot shows a 'List blobs' dialog box with a purple header bar. The header bar contains a folder icon, the text 'List blobs', an information icon (i), and a menu icon (three dots). Below the header, there are three rows of settings:

* Folder	<input type="text" value="/eventimports"/>	
Paging Marker	<input type="text" value="A marker that identifies the portion of the list to be returned with the list"/>	
Flat Listing	<input type="text" value="Yes"/>	 

At the bottom of the dialog, it says 'Connected to sosv2prodstorage.' followed by a blue link 'Change connection.'.

# Scenario 3: Blob Cleanup



# Thanks!



Kevin Griffin

[twitter.com/1kevgriff](https://twitter.com/1kevgriff)

[consultwithgriff.com](http://consultwithgriff.com)