Activate – Microsoft Graph for O365

Proof of Concept Playbook
Microsoft Teams life-cycle with Microsoft Graph

POC Overview

Introduction

This playbook uses a fictitious company that you probably have heard about before, Contoso Airlines. You will be playing the role of a flight administrator for Contoso Airlines. In your role, you are responsible for the coordination of the different flights within the airline. This includes tasks like flight scheduling, assigning pilots and flight attendants to flights, and notifying the crew with the departure gate and time among others.

As part of Contoso Airline's digital transformation journey, you have decided to use Microsoft Teams as the modern collaboration tool. Microsoft Teams meet the flight crew's needs enabling a rich collaboration and communication experience. For each existing flight you will be creating a new Microsoft Team adding the necessary crew members so that they can collaborate and coordinate all the details pertaining to their flight.

Due to the large number of flights (and Teams) you have to manage, you will be using an automated solution to facilitate your tasks

The following activities are designed to help you learn how to leverage Azure and Office 365 services to manage the Microsoft Teams life-cycle in an automated way using **Microsoft Graph**.

> The POC referenced in this guide is a **modified** version of the one that is available on **GitHub**.
>
> **Your instructor will provide you with all the files needed to configure this solution.**

This POC is divided into different exercises that will guide you through the configuration of all the needed elements:

> **IMPORTANT:**
>
> Before jumping straight to the exercises, make sure you read the pre-requisites and the high-level architecture sections to fully understand the requirements and the elements involved with this solution.

🕮 Exercises

- **Exercise 1: Configure MS Teams**
- **Exercise 2: Configure SharePoint Online**
- **Exercise 3: Provision Azure resources**
- **Exercise 4: Configure .NET project**
- **Exercise 5: Create schema extensions**
- **Exercise 6: Deploy Azure Functions**
- **Exercise 7: Configure flows in Power Automate**
- **Exercise 8: Testing the solution**

Prerequisites

🛈 💬 Before you begin this journey...

💻 Bring Your Own Device

This playbook is not intended to be used against your production tenant. It requires a sandbox tenant configured and ready. If you have not procured a sandbox or trial tenant from Microsoft, please go through the steps detailed on ***Activate-MS_Graph_Dev_POC_Setup_Guide*** for instructions on how to procure a tenant, *free* for a limited time, from Microsoft. In the setup guide, you will also find details on how to enable Azure Function capabilities, as they are required for this POC.

🖳 Workstation Pre-Requisites

The solution you will be deploying needs that you have at least **Visual Studio 2017** with **Azure development** workload.

- For **Visual Studio 2017**, go through **this steps** to check the prerequisites and to make sure you are using the most recent version of the **Azure Function Tools**.
- For **Visual Studio 2019**, you can skip the steps mentioned in the above article.

🛈 User Accounts

You will be using your tenant administrator account to complete all the steps in this playbook. In order to test the solution, however, this playbook references specific user accounts. These user accounts **should** be available in the special demo tenant you are using (see **Exercise 8 - Testing the solution** for details). If you are using a blank tenant or your own test tenant, you will need to substitute these accounts for others existing in your tenant.

Solution Files

Besides this playbook, your instructor will provide you with the needed files to complete this POC. Inside **contoso-airlines-POC.zip** you will find the Visual Studio Solution. You will need to unzip the file to a desired location in your workstation.

High-level Architecture

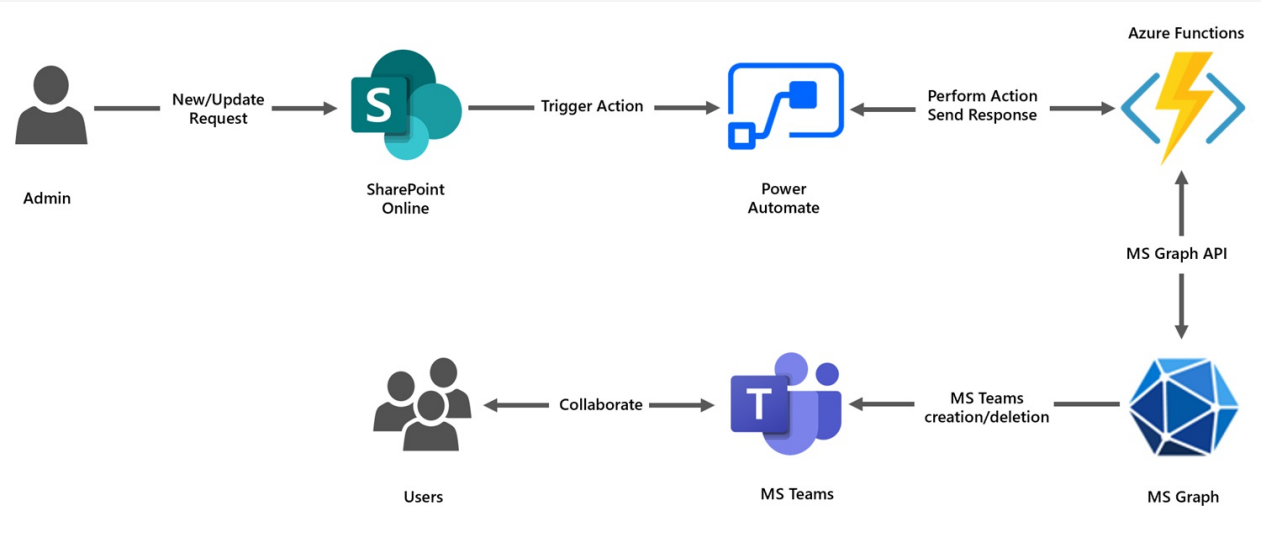
This POC will demonstrate how you can automate the life-cycle of Microsoft Teams teams (creation/update/deletion) using following services:

- Microsoft Graph
- Azure Functions
- SharePoint Online
- Microsoft Teams
- Power Automate with custom connectors

Specifically, Azure Functions are invoked via a Power Automate custom connector to provision a team within Microsoft Teams. Power Automate flows are triggered when the flight admin interacts (adds, modifies or deletes) with the Flight List hosted in a SharePoint Online site.

Components Diagram

The following diagram shows how these elements interact with each other:



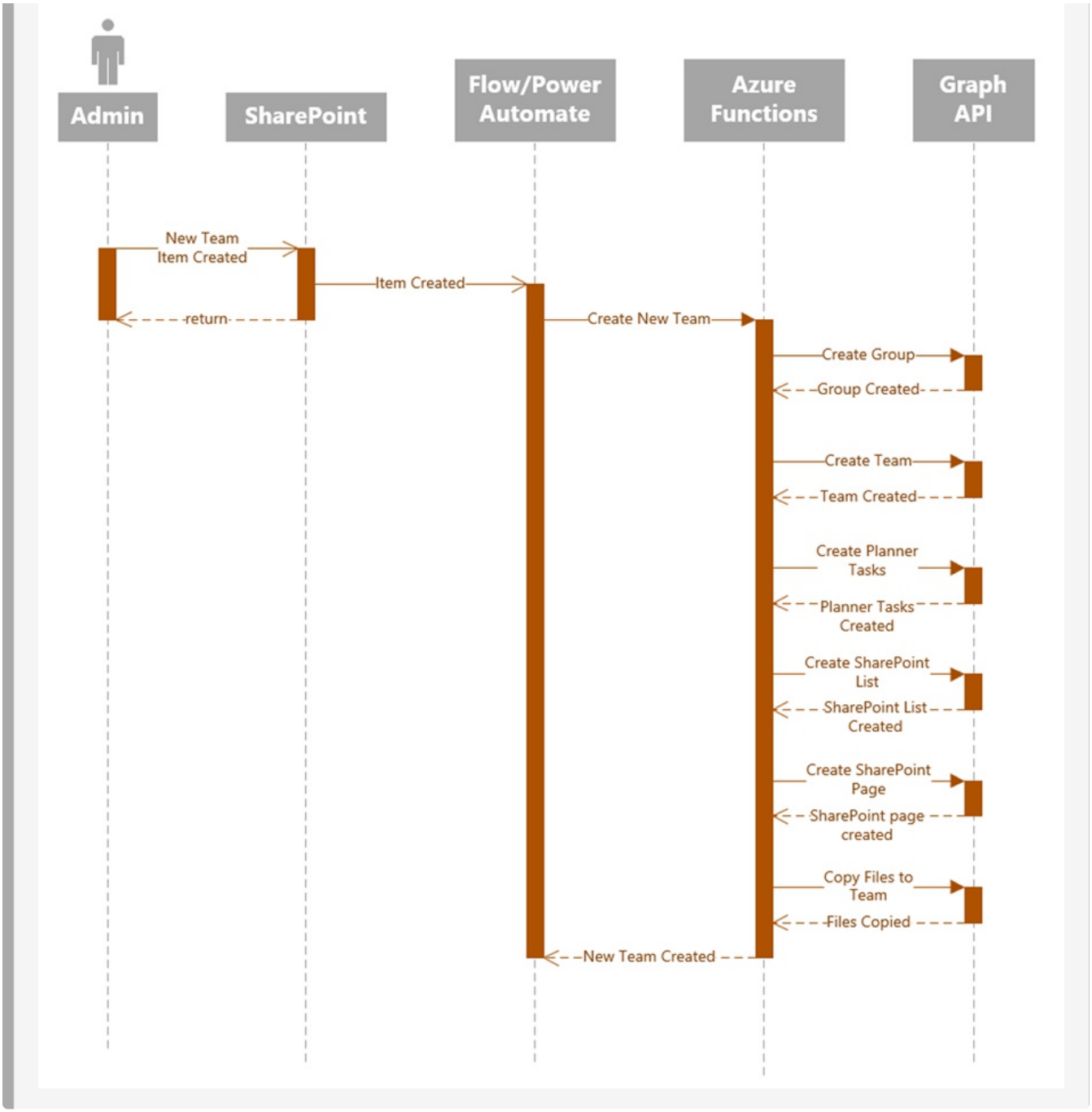
The Azure Functions use **Microsoft Graph** to handle the following provisioning tasks:

Microsoft Graph operations and resources used in this solution:

- Creates a unified **group** for the flight team, and initializes a **Team** for the group.
- Adds an **open extension** to the Group.
- **Creates invitation** to the Team.
- **Adds members** to Team.
- Creates **channels** in the team and adds a template flight log document to the team's files.
- **Creates chat message** in a Team Channel
- **Installs an app** (Polly) to the team.
- Creates a Planner **plan** with **bucket** and **tasks** for the team.
- Creates a custom SharePoint **page** and custom SharePoint **list** for the team.
- Adds **tabs** to the team's General channel for the planner plan and SharePoint page.
- **Archives the team** when the flight is deleted.

Provisioning operations

Taking a closer look at the teams provisioning piece, the below diagram shows a more detailed sequence of events:



Exercise 1: Microsoft Teams for guest access

Introduction

In this exercise you will enable guest access for Teams through the user interface. This is needed as the solution adds an external user to the team.

☑ Objectives

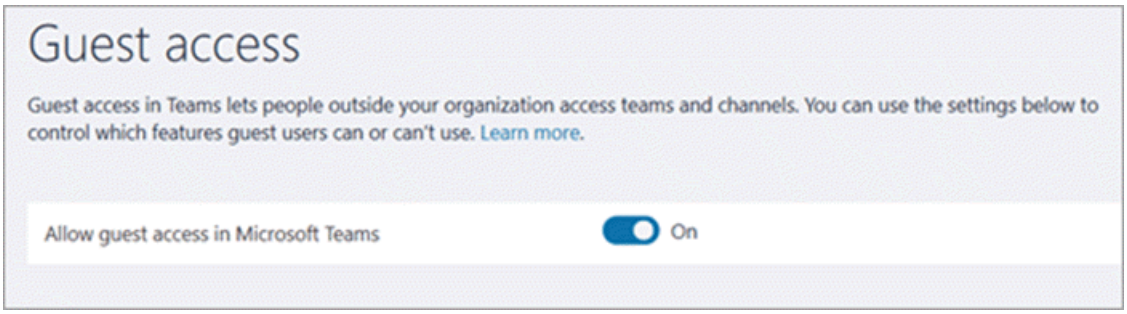
After completing this exercise, you will have gained the knowledge to enable guest access for Microsoft Teams.

☑ Prerequisites

- Office 365 Tenant with Microsoft Teams
- Office 365 Admin Account

☑ Tasks

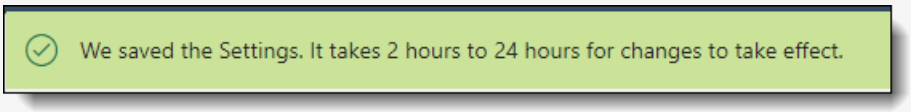
1. Sign-in to **Microsoft Teams admin center**.
2. On the left panel, expand **Org-wide settings** and click **Guest access**.
3. Set the **Allow guest access in Microsoft Teams** toggle switch to **On**.



4. Confirm the settings in the below table are set with the following values:

Setting	Value
Make private calls	On
Allow IP Video	On
Screen sharing mode	Entire screen
Allow Meet Now	On
Edit sent messages	On
Guest can delete sent messages	On
Chat	On
Use Giphy content rating	Moderate
Use Memes in conversations	On
Use Stickers in conversations	On
Allow immersive reader for viewing messages	On

5. At the bottom of the screen select **Save** to save the settings.

> You might see the following message as a top banner:
>
> 
>

Exercise 2: Create SharePoint Online Modern Team Site

Introduction

In this exercise you will create SharePoint Online site that will contain the list with all the flights information. This is the master list that will trigger the Power Automate flows when anything gets modified within that list.

Objectives

After completing this lab, you will be able to:

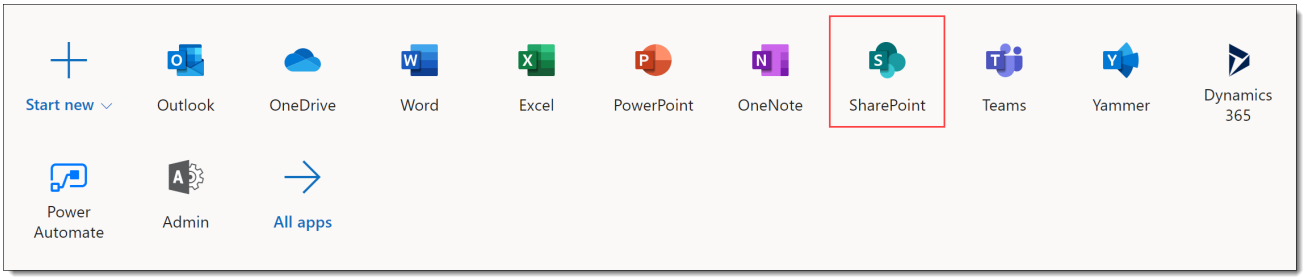
- Create a SharePoint Online modern team site.
- Create a list in SharePoint Online with custom columns/fields.
- Create documents in a SharePoint document library.

Prerequisites

- O365 Tenant
- O365 Admin Account

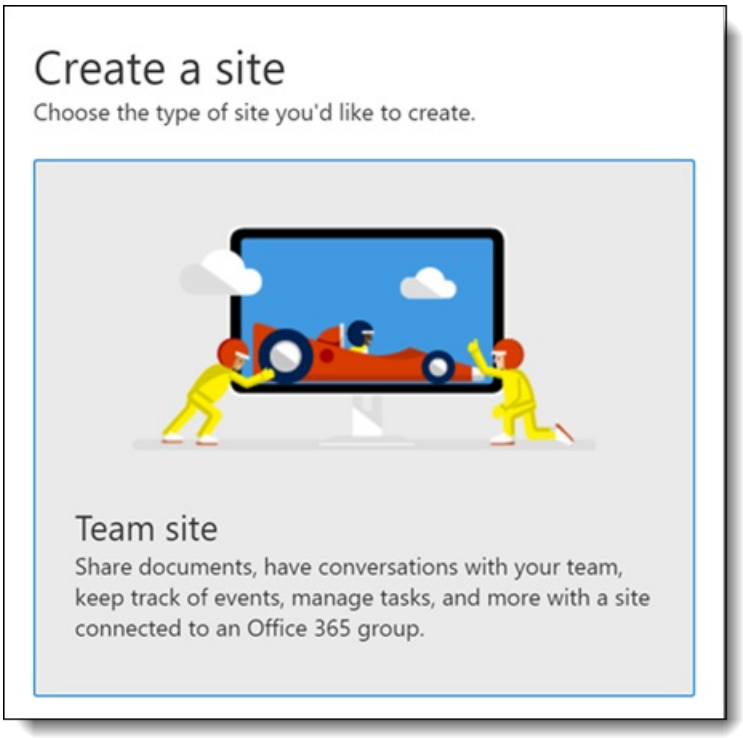
Tasks

1. Browse to **https://portal.microsoft.com** and log in using your tenant's admin account.
2. Click on the SharePoint icon to go to the Office 365 organization's root SharePoint site.

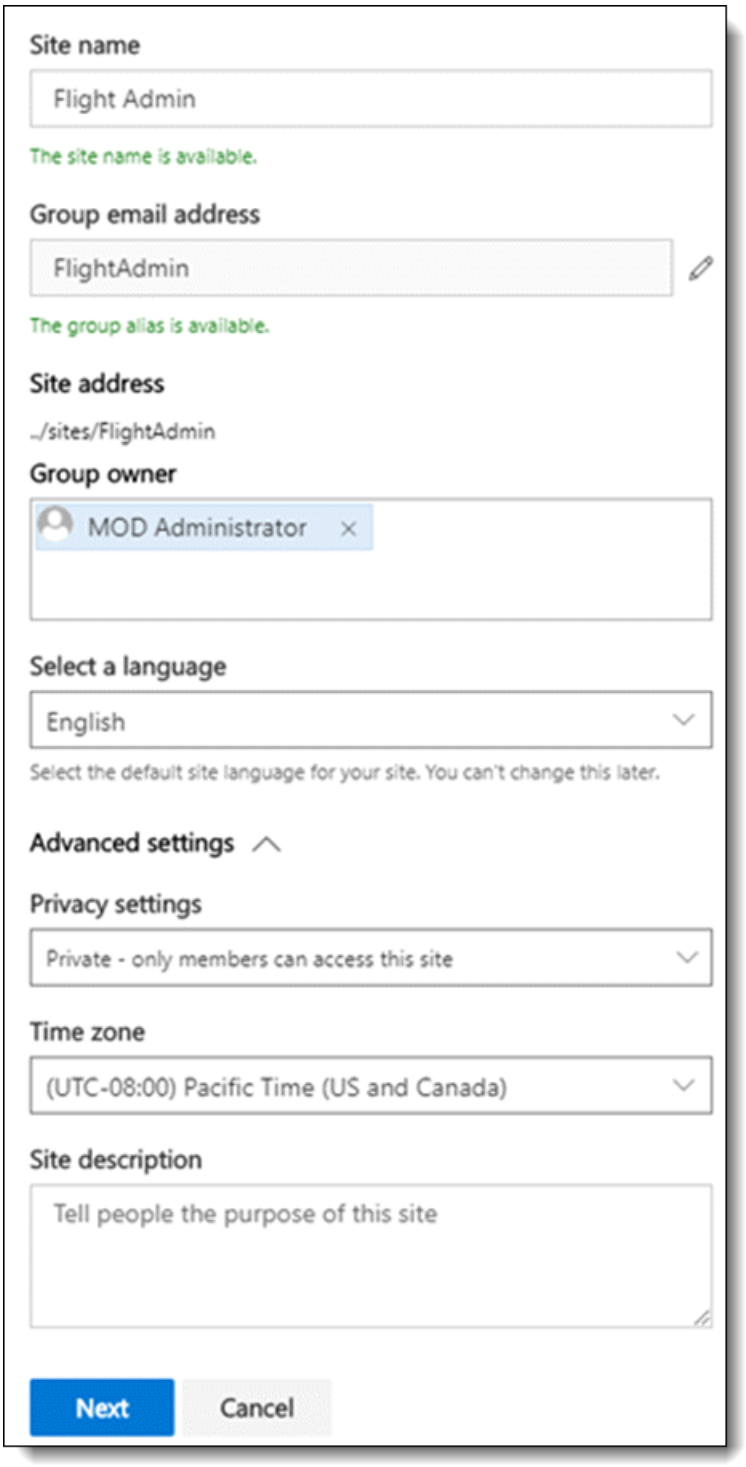


3. Once you are in the root site, click **Create Site** at the top.

4. Then choose **Team site**.

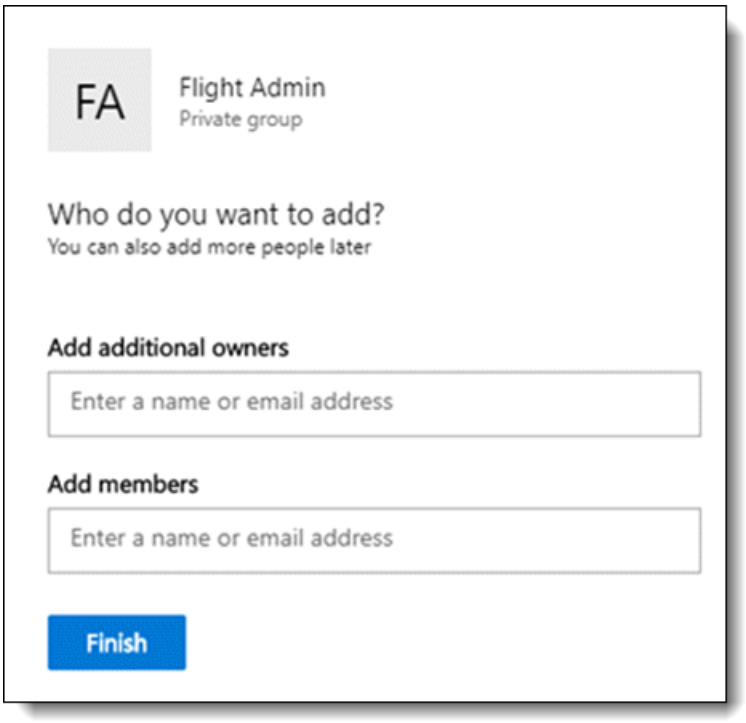


5. Name the site **Flight Admin** and choose **Next**.

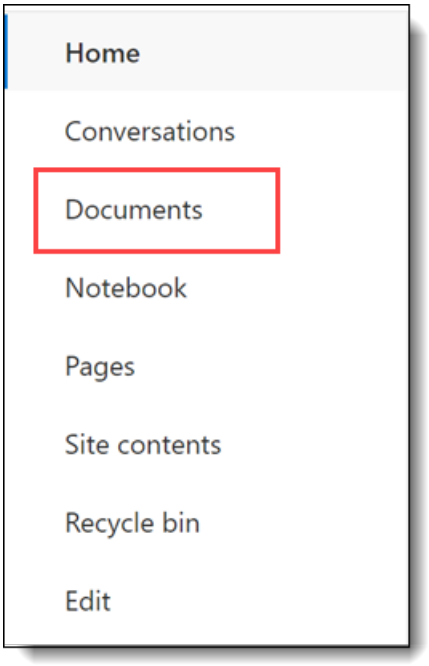


> **Note**: There is no need to add additional owners or members.

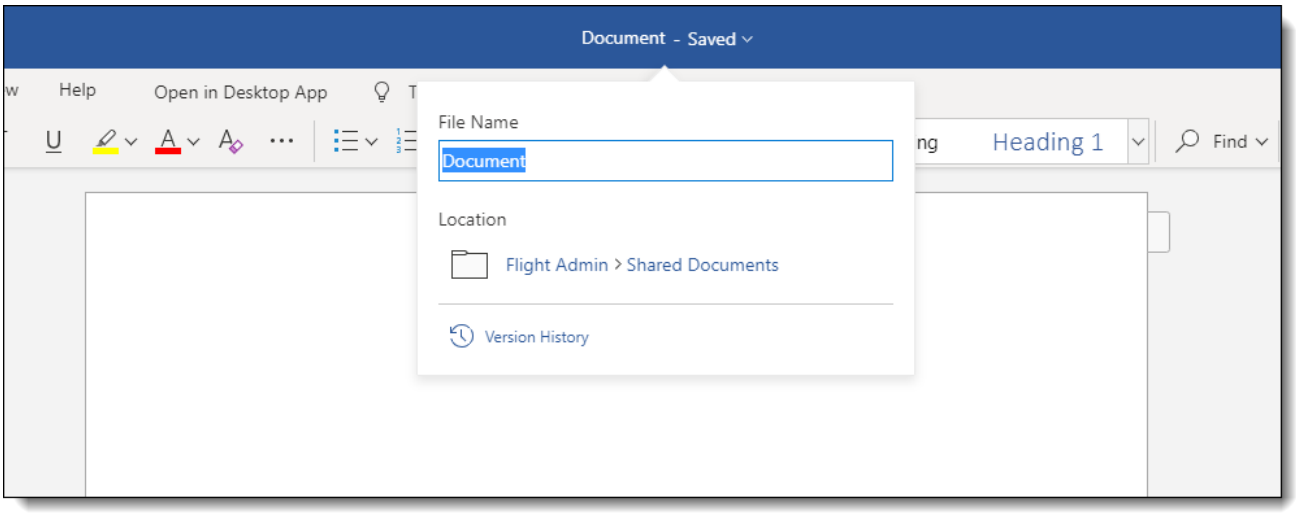
6. Click **Finish**.

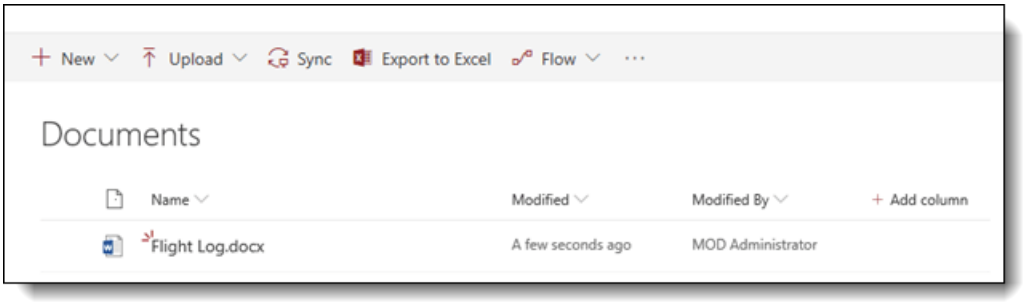


7. Your new site will load in the browser. On the left panel, click on **Documents** to access the site's default document library.

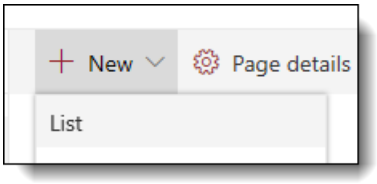


8. Click on **New** and click on create a **Word document**. Your document will open in a new tab. Click on the banner where it says “Document - Saved” to change the file name. Name your document **Flight Log**.

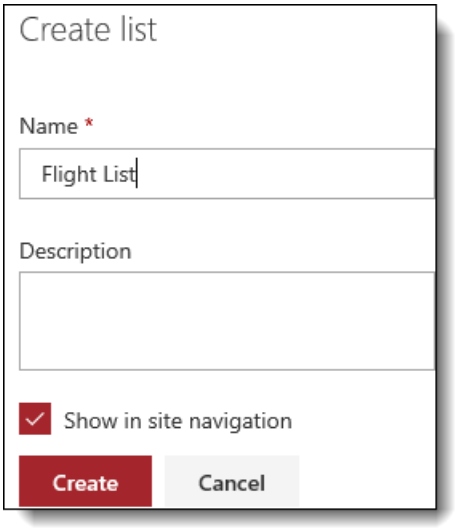




9. Navigate back to the home page. On the site's home page, choose **New** in the upper-left corner, then choose **List**.



10. Name the list **Flight List** and click **Create**.



11. When the new list loads in the browser, use the **Add column** button to add the following columns.

> **NOTE:**
>
> When configuring each column, make sure to select **More options** to expose all the column settings:

Name	Type	Require a value?	Other settings
Flight Number	Number	Yes	Number of decimal places: 0, Enforce unique values: Yes
Admin	Person	Yes	Allow selection of groups: No, Allow multiple selections: No
Pilots	Person	Yes	Allow selection of groups: No, Allow multiple selections: Yes
Flight Attendants	Person	Yes	Allow selection of groups: No, Allow multiple selections: Yes
Catering Liaison	Single line of text	Yes	
Departure Gate	Single line of text	Yes	
Departure Time	Date	Yes	Include time: Yes

When you're done, the list should look something like this:

Exercise 3: Provision Azure Resources

☑ Introduction

In this exercise you will create a new Function app as well as create and configure 2 app registrations: one for Azure function and another one for the Power Automate custom connector.

The Power Automate custom connector will interact with the Azure Functions, which will contact Microsoft Graph to create/delete teams within Microsoft Teams.

☑ Objectives

After completing this lab, you will be able to:

- Set up and configure a Function App.
- Crate 2 app registrations.
- Configure the app registrations:
    - set permissions to consume MS Graph resources.
    - create client secrets.
    - create custom scope.
- Connect the 2 app registrations so that one can consume an exposed API from the other.

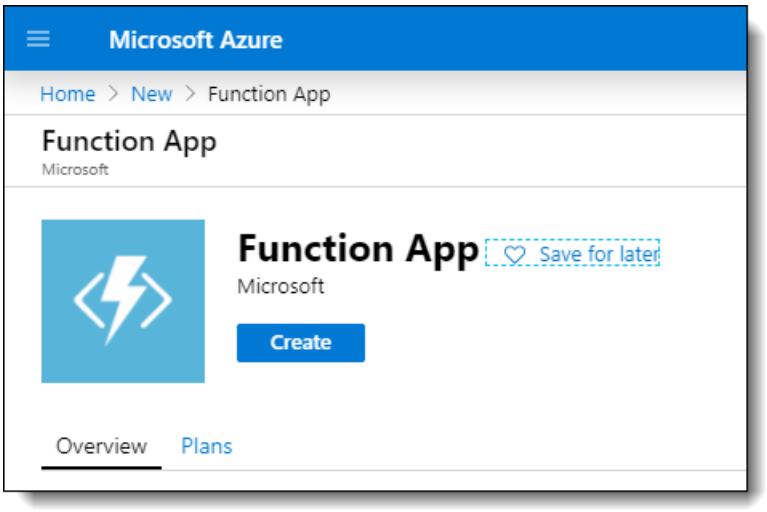
☑ Prerequisites

- Azure environment with the ability to use Azure Functions capabilities.
- Administrator Account

☑ Tasks

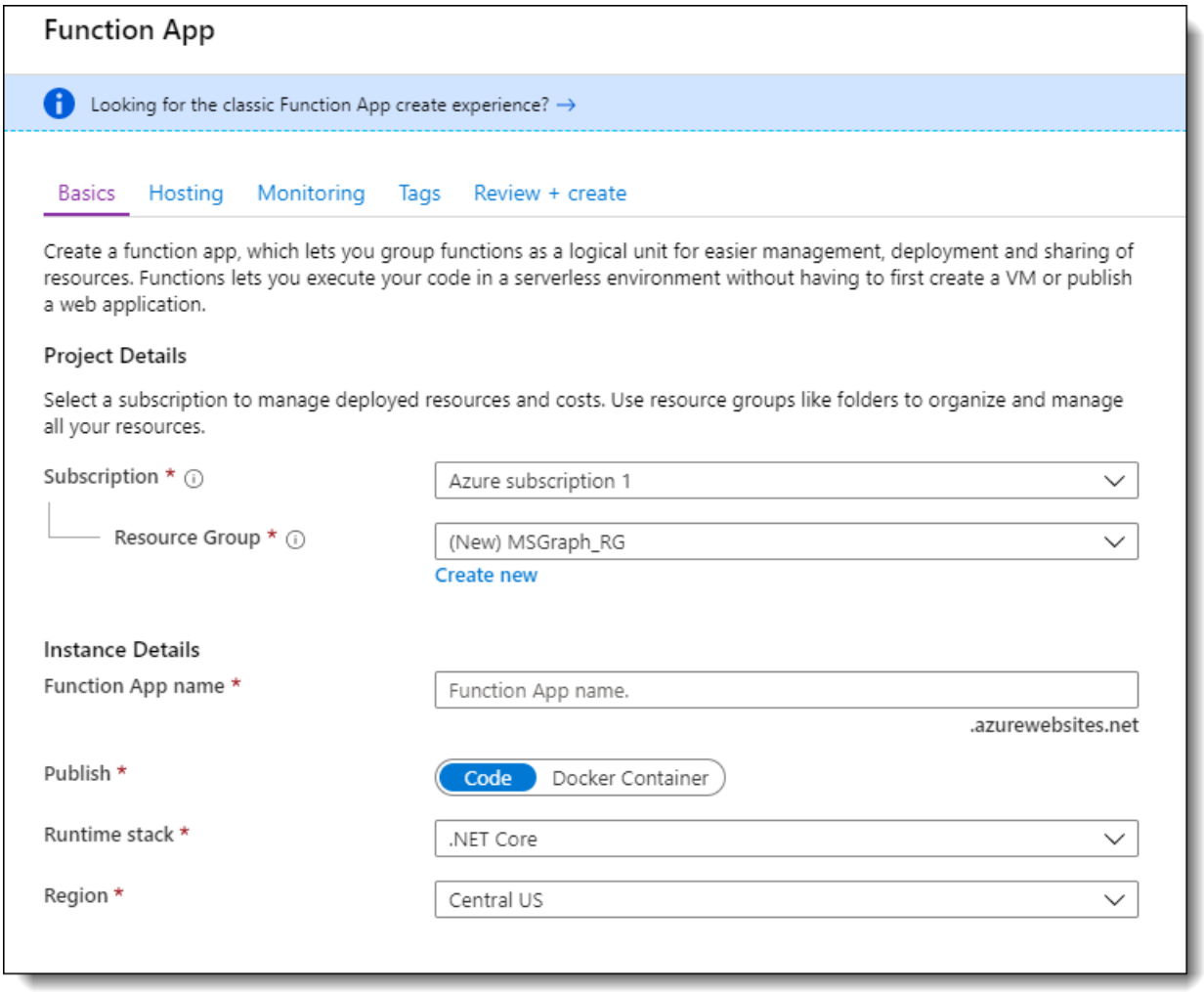
⚙ Create a new Function App

1. Go to the **Azure Portal** and sign in with your admin account. Use the **Create a resource** button to add a new Function App. Click **Create**

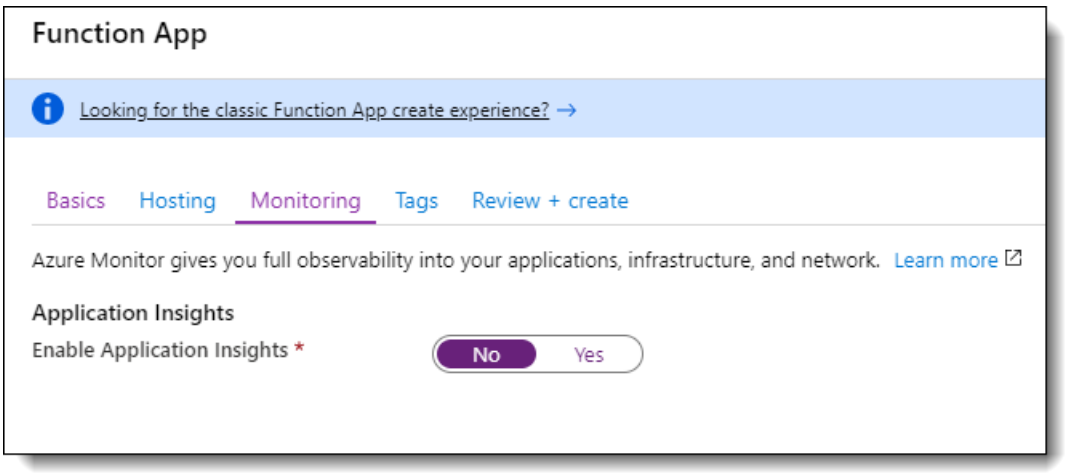


> **NOTE**: You should have followed the ***Activate-MS_Graph_Dev_POC_Setup_Guide*** as a pre-requisite for this playbook. If you did not, you create a free account that will provide you with $200 credit for 30 days. Follow the ***Activate-MS_Graph_Dev_POC_Setup_Guide*** for details on how to enable Azure Function capabilities, as they are required for this POC.

- On the **Basics** tab, select the Azure subscription you just created or select an existing one.
- You would need to provide an existing **Resource Group** or click on **Create** new to create a new one.
- Provide a Function App Name and select **.NET Core** Runtime Stack.



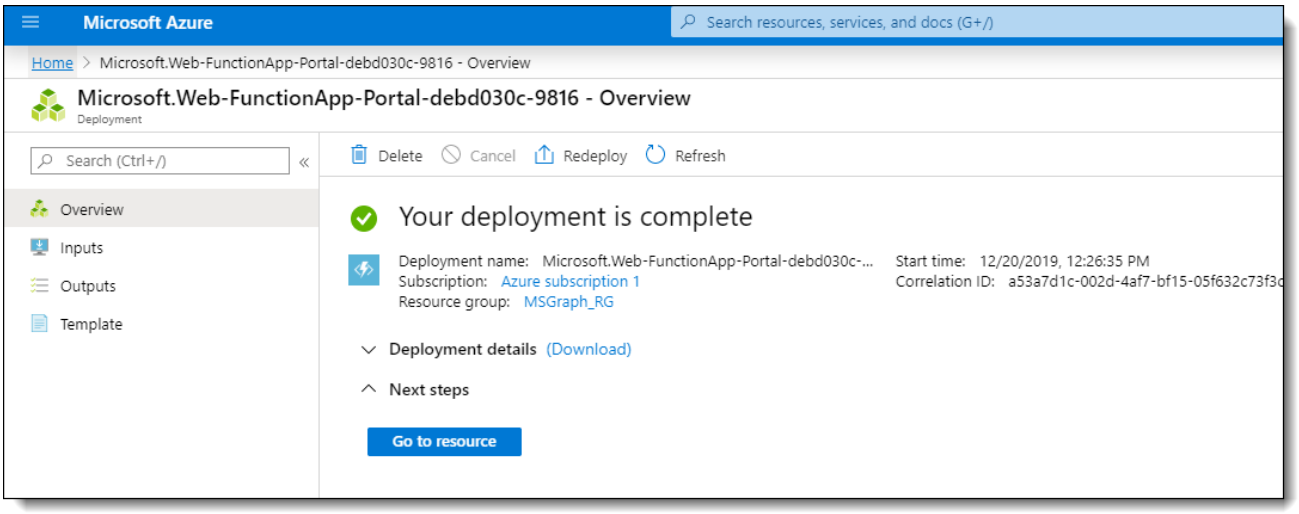
- In the **Monitoring** tab, Select **No** for Application Insights.



- Click **Review+Create** button at the bottom of the screen.
- Wait for the validation process to complete and click on **Create**.

> ℹ The creation of the function app can take several minutes (typically between 3-5 minutes).

2. Once the app is created, click on **Go to resource**.

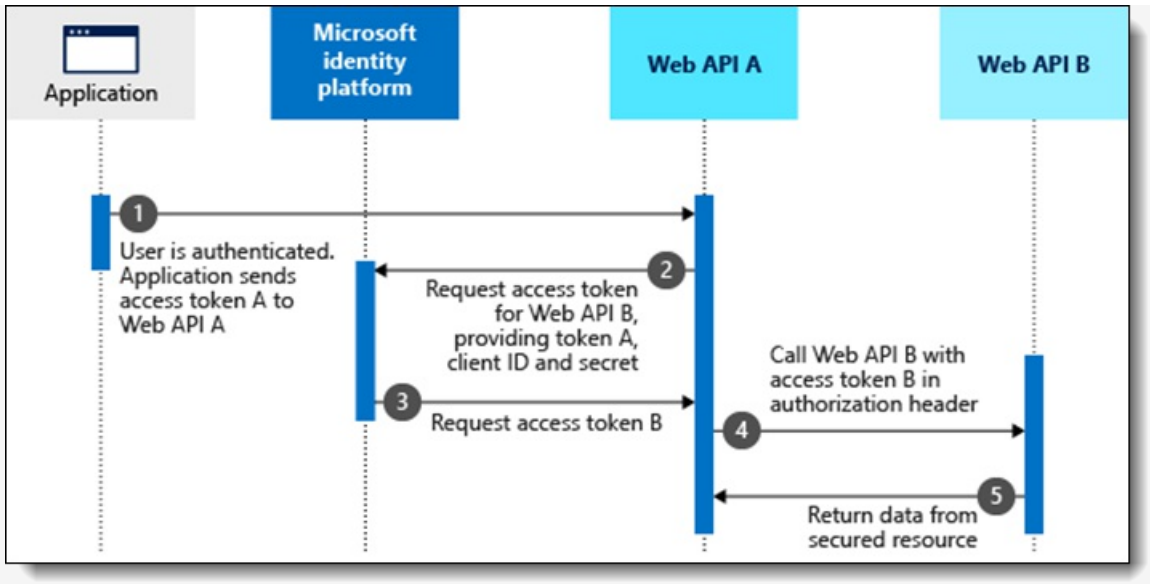


3. > **IMPORTANT**:
   > Copy the **URL** value of your function app, you'll need it later.

Create app registrations

You need to create two app registration in the Azure Portal. The reason for this is that we are going to be using the **On-Behalf-Of OAuth flow** when calling MS Graph.

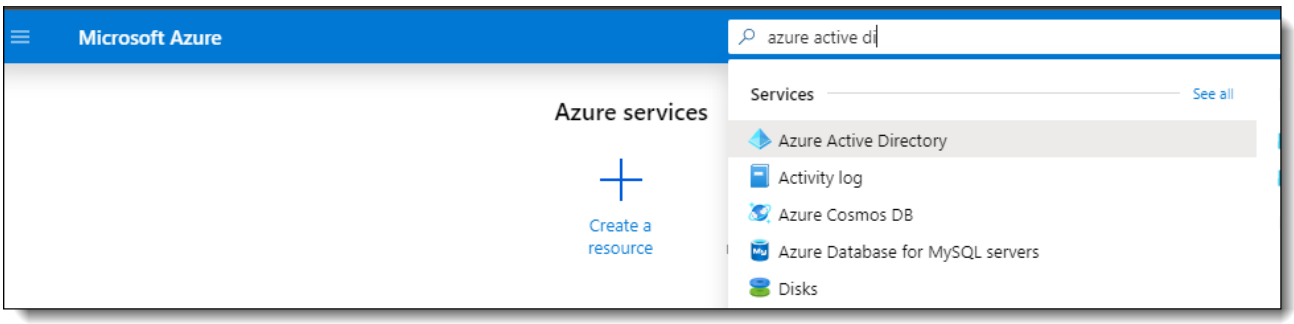
> ### On-Behalf-Of (OBO) OAuth 2.0 flow
>
> **The OAuth 2.0 On-Behalf-Of flow (OBO)** serves the use case where an application invokes a service/web API, which in turn needs to call another service/web API. The idea is to propagate the delegated user identity and permissions through the request chain. For the middle-tier service to make authenticated requests to the downstream service, it needs to secure an access token from the Microsoft identity platform, on behalf of the user.



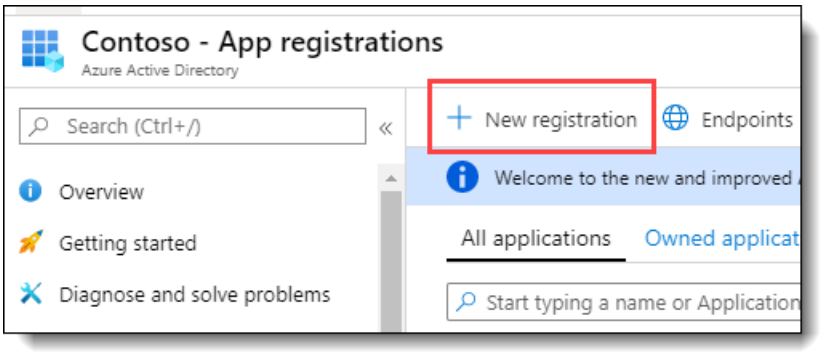
1. The client application makes a request to API A with token A (with an auth claim of API A).
2. API A authenticates to the Microsoft identity platform token issuance endpoint and requests a token to access API B.
3. The Microsoft identity platform token issuance endpoint validates API A's credentials with token A and issues the access token for API B (token B).
4. Token B is set in the authorization header of the request to API B. Data from the secured resource is returned by API B.

Register app for Azure Function

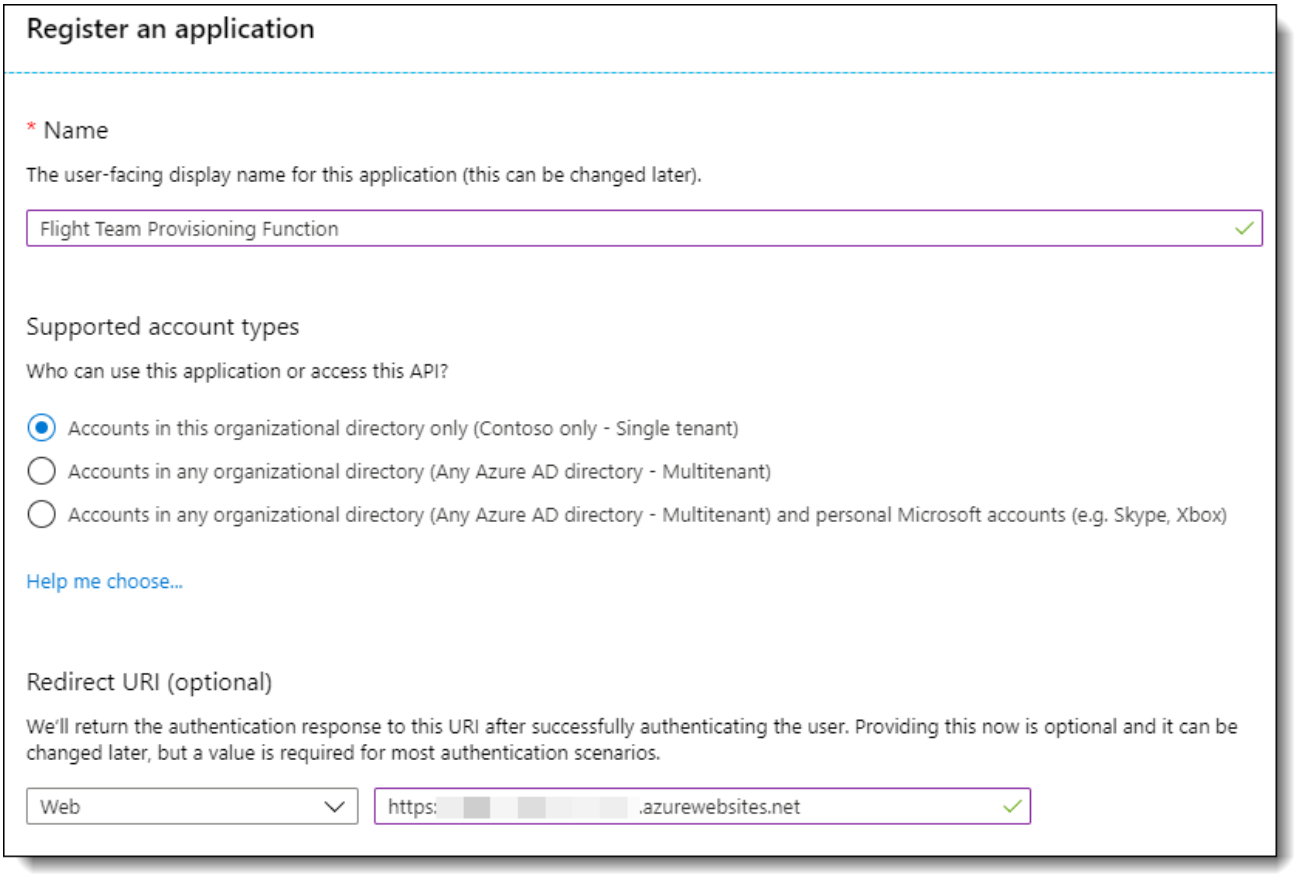
1. Navigate to **https://portal.azure.com** and search for **Azure Active Directory**. Click on it.



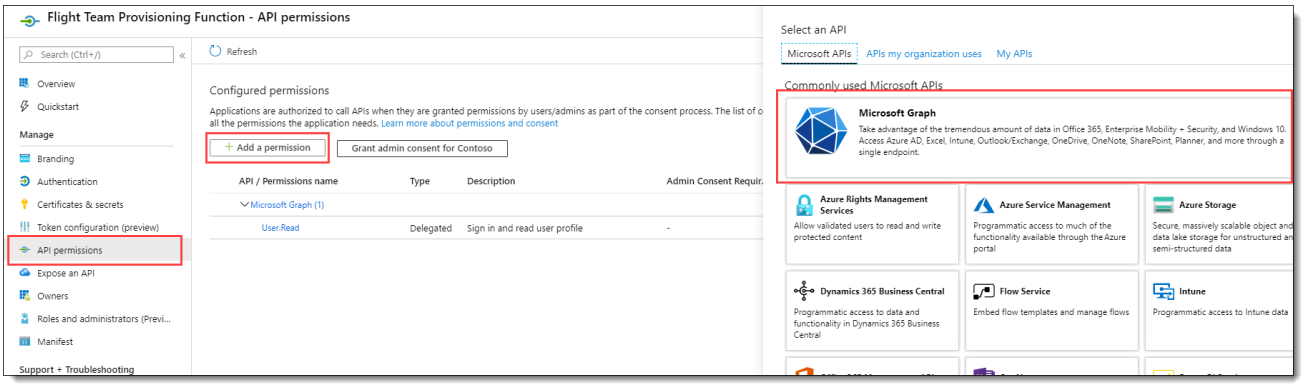
2. In the left pane, under **Manage**, click A**pp registrations**.
3. Click **New registration** on the top bar.



4. Set the **Name** to **Flight Team Provisioning Function**.
5. Set the **Supported account types** to **Accounts in this organizational directory only**.
6. Under **Redirect URI**, set the drop-down to **Web**, and set the value to the URL for the Azure Function app that you previously copied.



7. Click **Register**.
8. > **IMPORTANT:**
   > Copy the value of **Application (client) ID**, you will need it later.
9. On the left panel, click **API permissions**, then choose **Add a permission**.
10. Select **Microsoft Graph**.



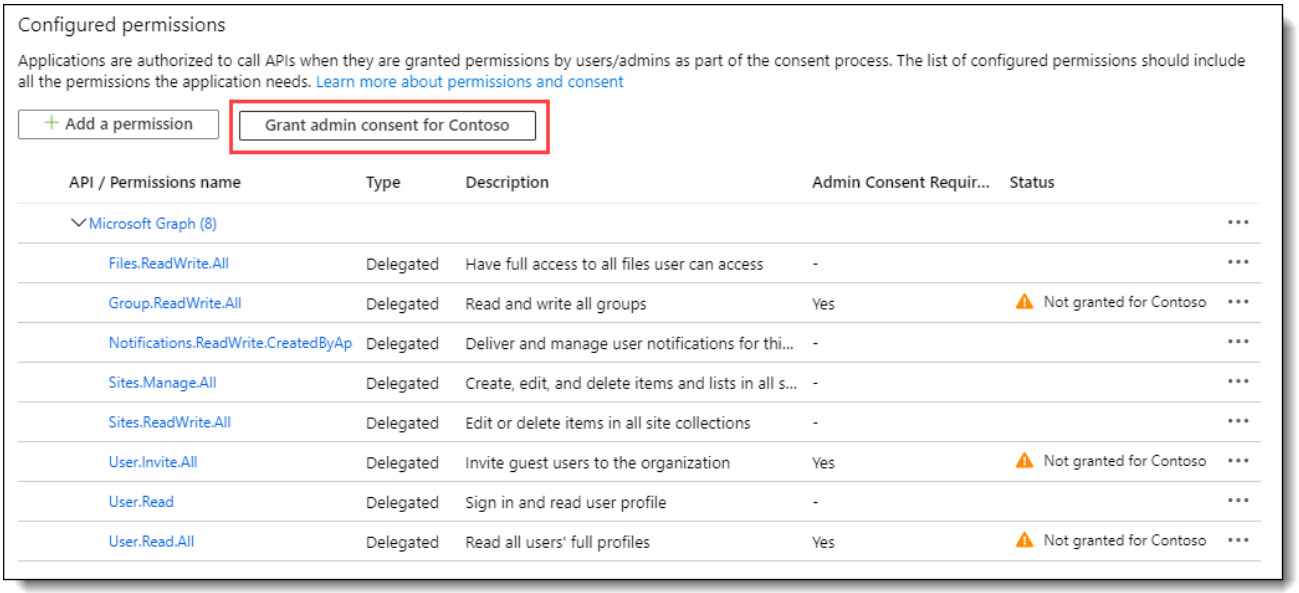
11. Click on **Delegated permissions**.

12. Select the following permissions under **Delegated permissions**:

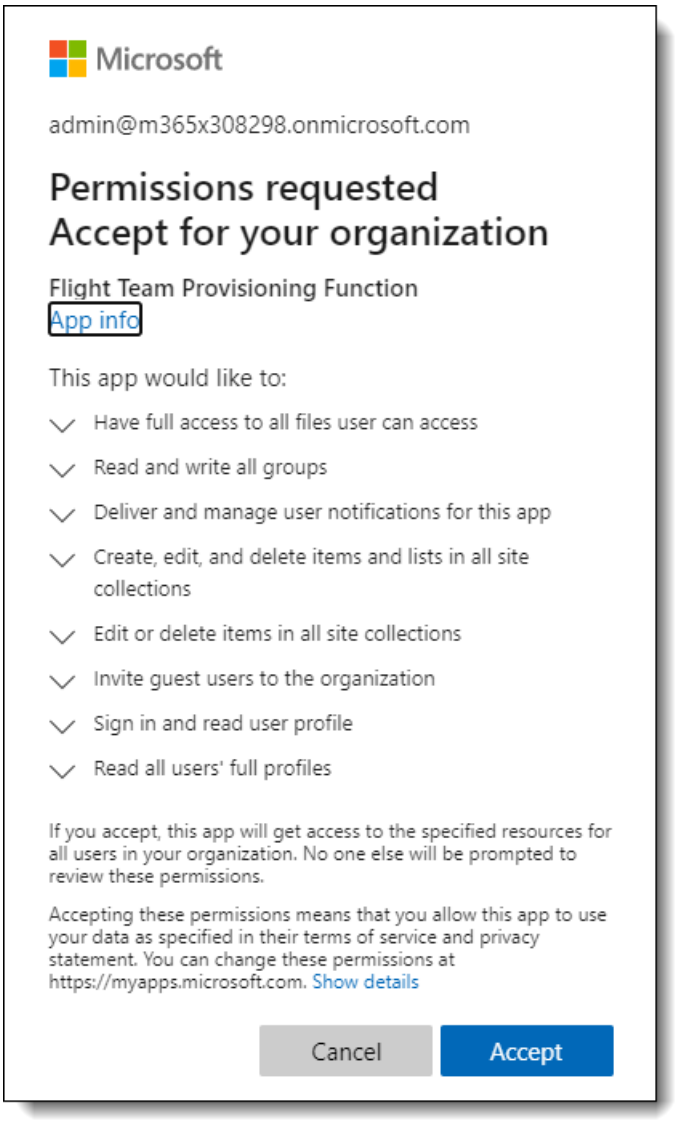
> **Delegated Permissions**
>
> Files.ReadWrite.All
> Group.ReadWrite.All
> Notifications.ReadWrite.CreatedByApp
> Sites.Manage.All
> Sites.ReadWrite.All
> User.Invite.All
> User.Read
> User.Read.All

11. Click **Add permissions**.

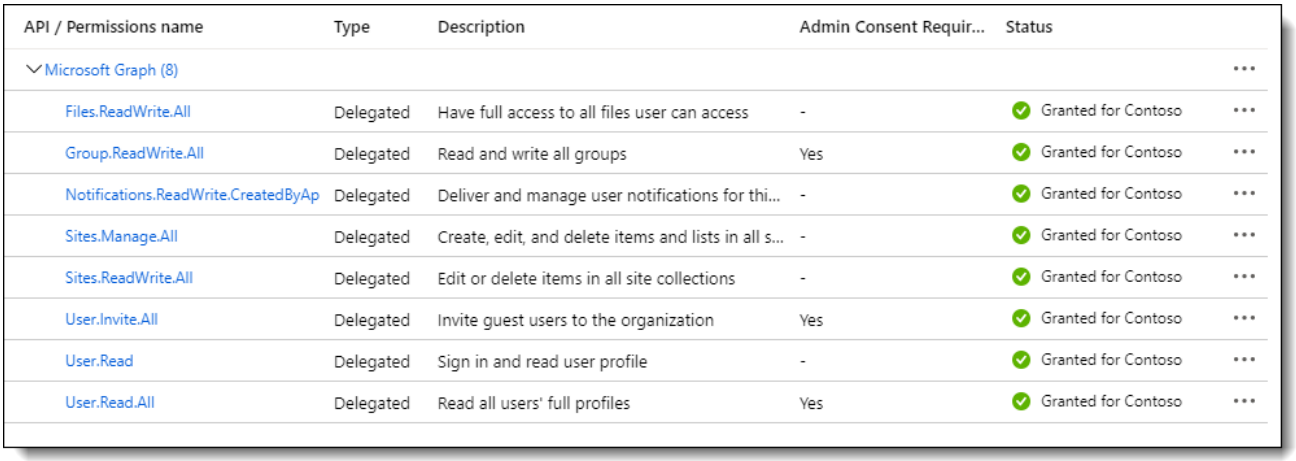
12. Some of the above permissions require admin consent. click on **Grant admin consent for** button to consent to the app.



You will be presented with a prompt so that you can grant consent. Use your tenant admin account and click on **Accept** when presented with the consent prompt window.



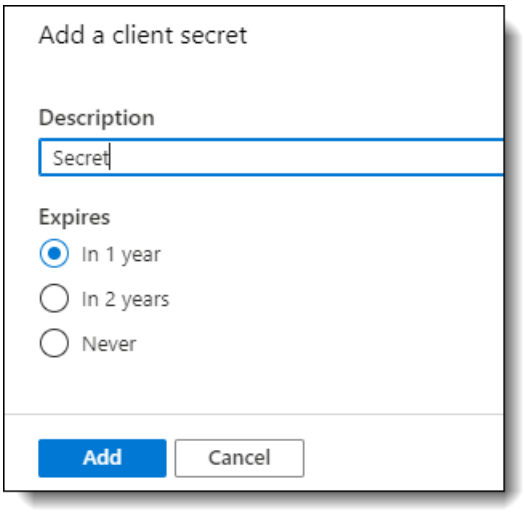
> ℹ You have to wait a couple of minutes before permissions are granted.



API / Permissions name	Type	Description	Admin Consent Requir...	Status
Microsoft Graph (8)				
Files.ReadWrite.All	Delegated	Have full access to all files user can access	-	Granted for Contoso
Group.ReadWrite.All	Delegated	Read and write all groups	Yes	Granted for Contoso
Notifications.ReadWrite.CreatedByAp	Delegated	Deliver and manage user notifications for thi...	-	Granted for Contoso
Sites.Manage.All	Delegated	Create, edit, and delete items and lists in all s...	-	Granted for Contoso
Sites.ReadWrite.All	Delegated	Edit or delete items in all site collections	-	Granted for Contoso
User.Invite.All	Delegated	Invite guest users to the organization	Yes	Granted for Contoso
User.Read	Delegated	Sign in and read user profile	-	Granted for Contoso
User.Read.All	Delegated	Read all users' full profiles	Yes	Granted for Contoso

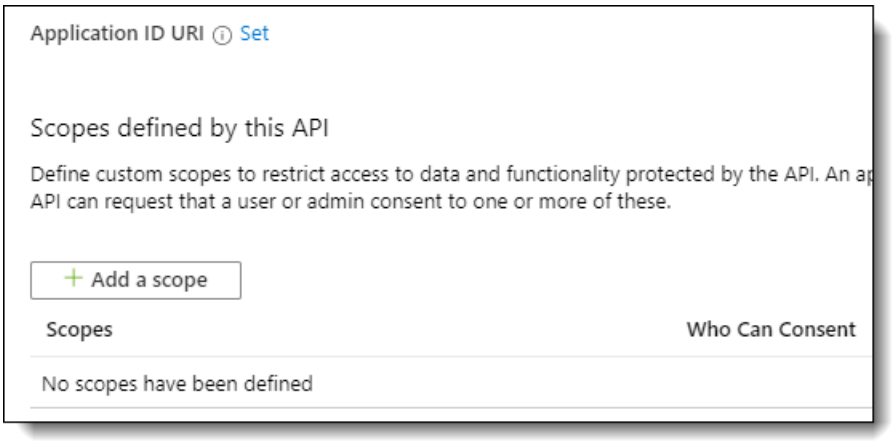
13. On the left panel, click **Certificates & secrets**. Under **Client Secrets**, then click **New client secret**.

14. Enter a description, select a duration, then choose **Add**.

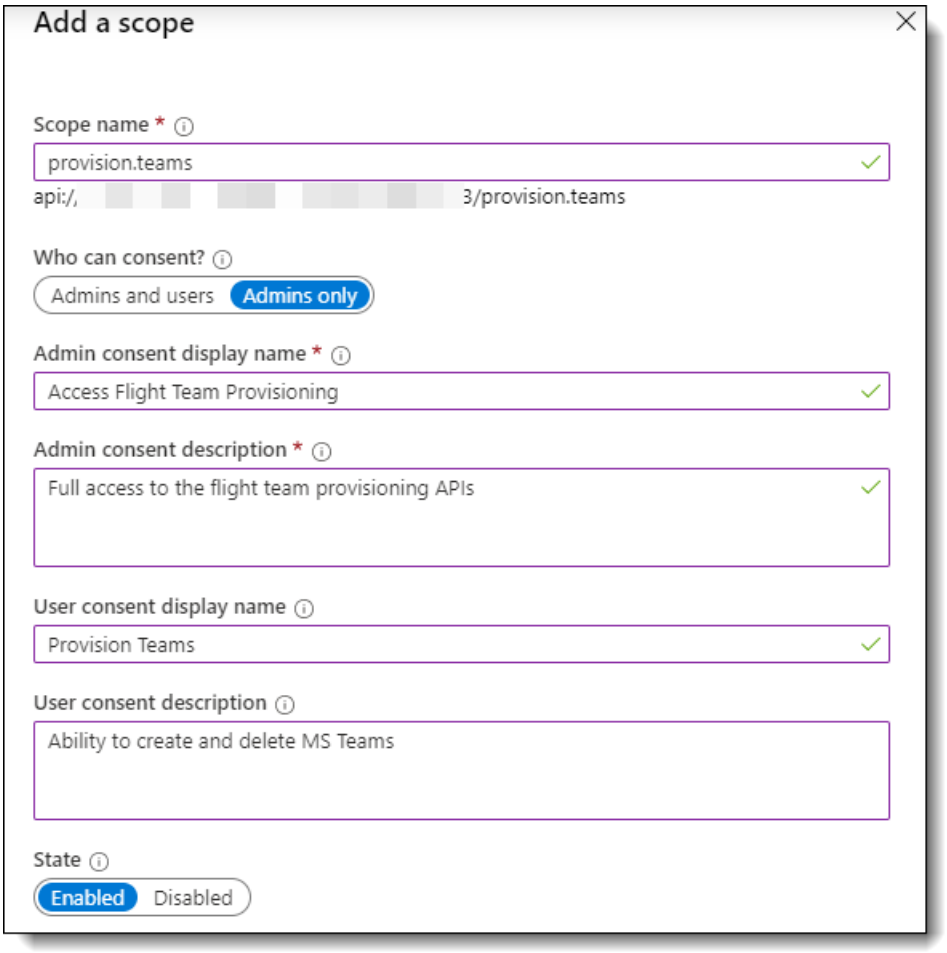


> **IMPORTANT**:
> Copy the value of the generated secret, you will need it later.
> **This is the only time that you will be able to obtain this secret.**

15. On the left panel, click **Expose an API**. Under Scopes defined by this API, click **Add a scope**.



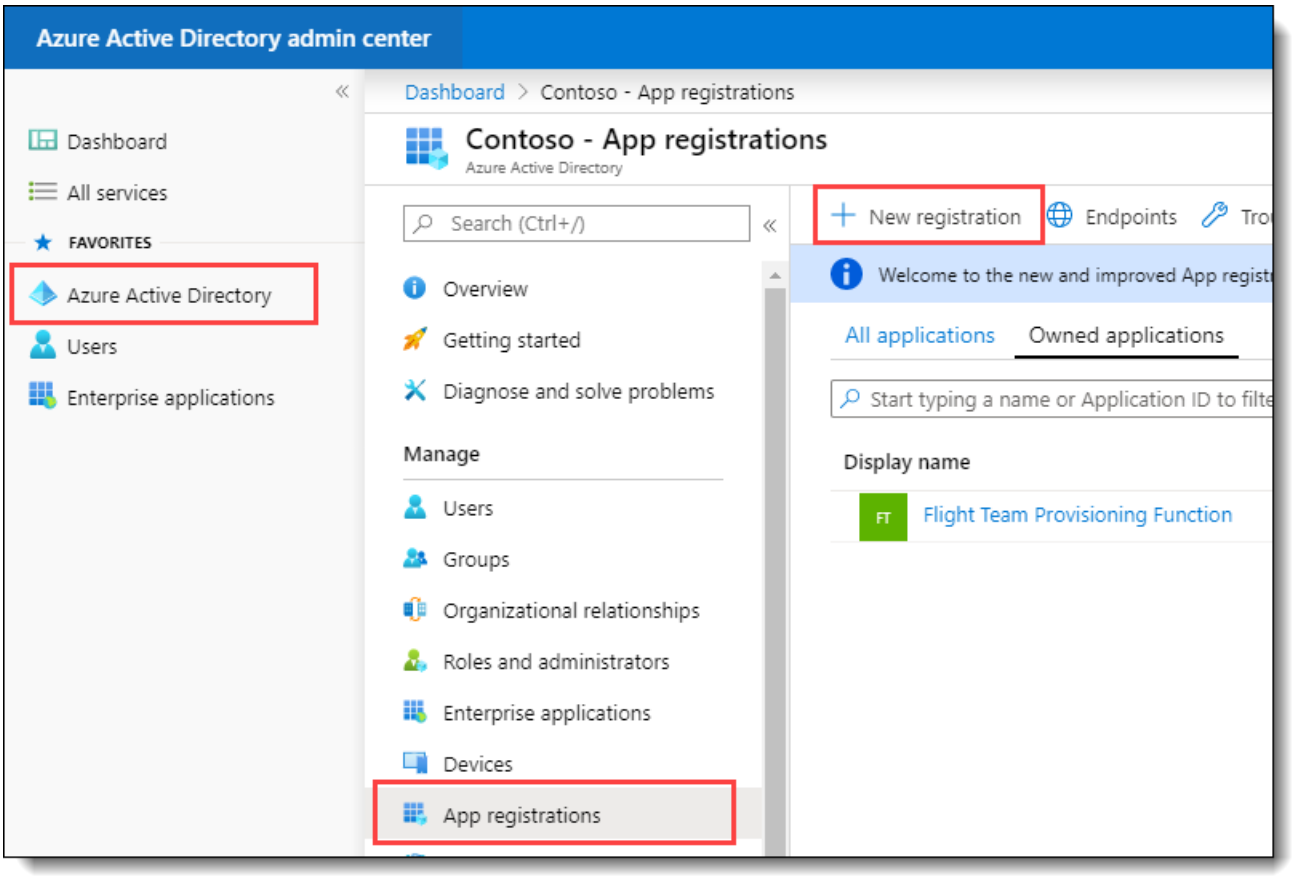
16. Leave the value for **Application ID URI** set to the generated value and choose **Save and continue**.
17. Set the Scope name to **provision.teams**.
18. Set **Who can consent?** to **Admins only**.
19. Set Admin consent display name to **Access Flight Team Provisioning**.
20. Set Admin consent description to "Have full access to the flight team provisioning APIs".
21. Set **User consent display name** and **description** as desired.
22. Set State to **Enabled**.



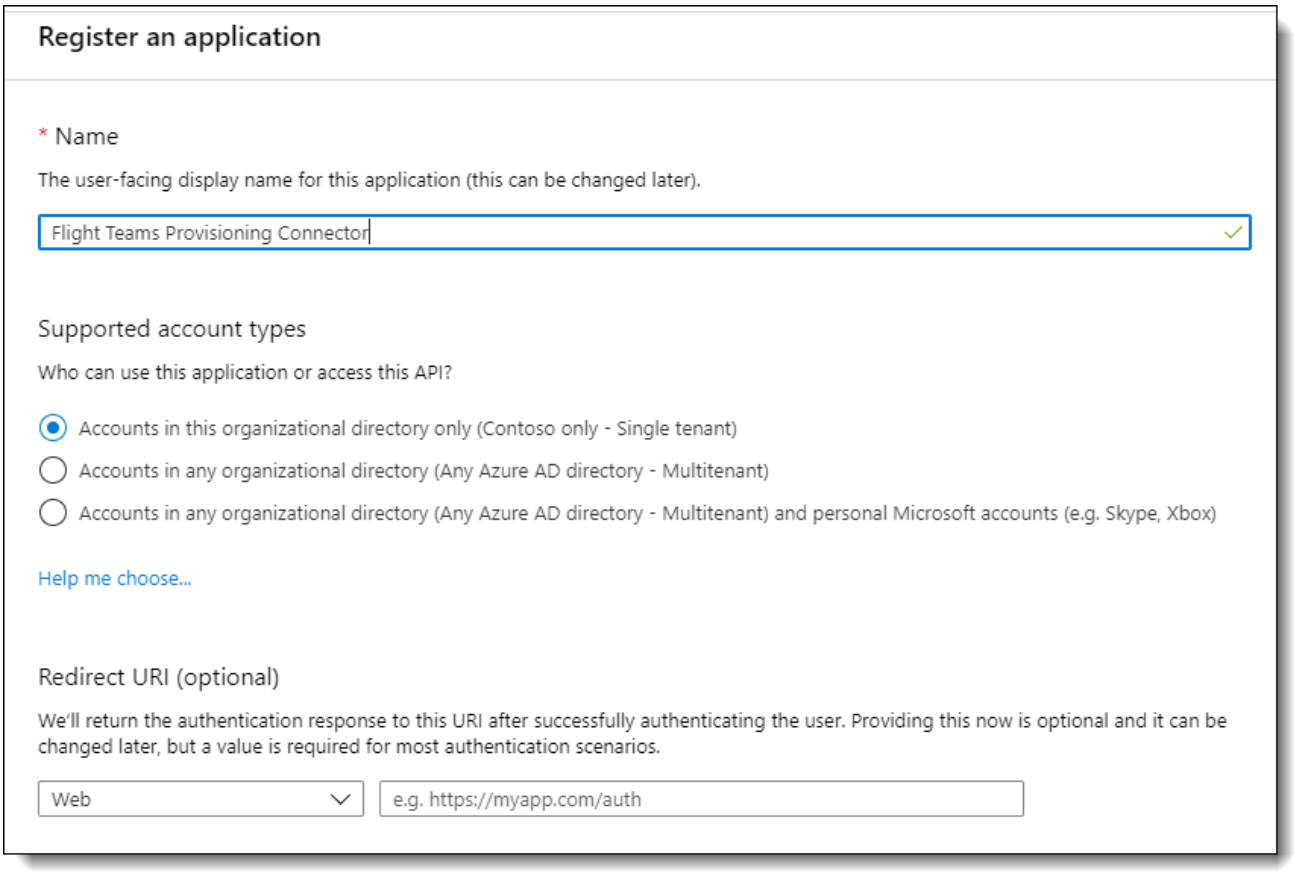
23. Choose **Add scope**.

Register app for Power Automate custom connector

1. In **Azure Active Directory** on the left panel, click **App registrations**.
2. Click on **New registration** in the top bar to register a new application.



3. Set the Name to **Flight Team Provisioning Connector**.
4. Set the **Supported account types** to **Accounts in this organizational directory only**.
5. Under **Redirect URI**, leave the value empty.

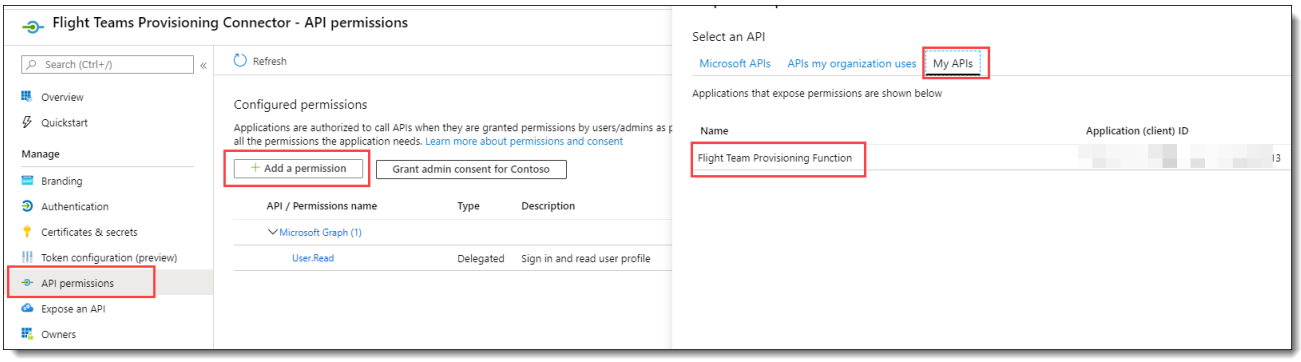


6. Choose **Register**.

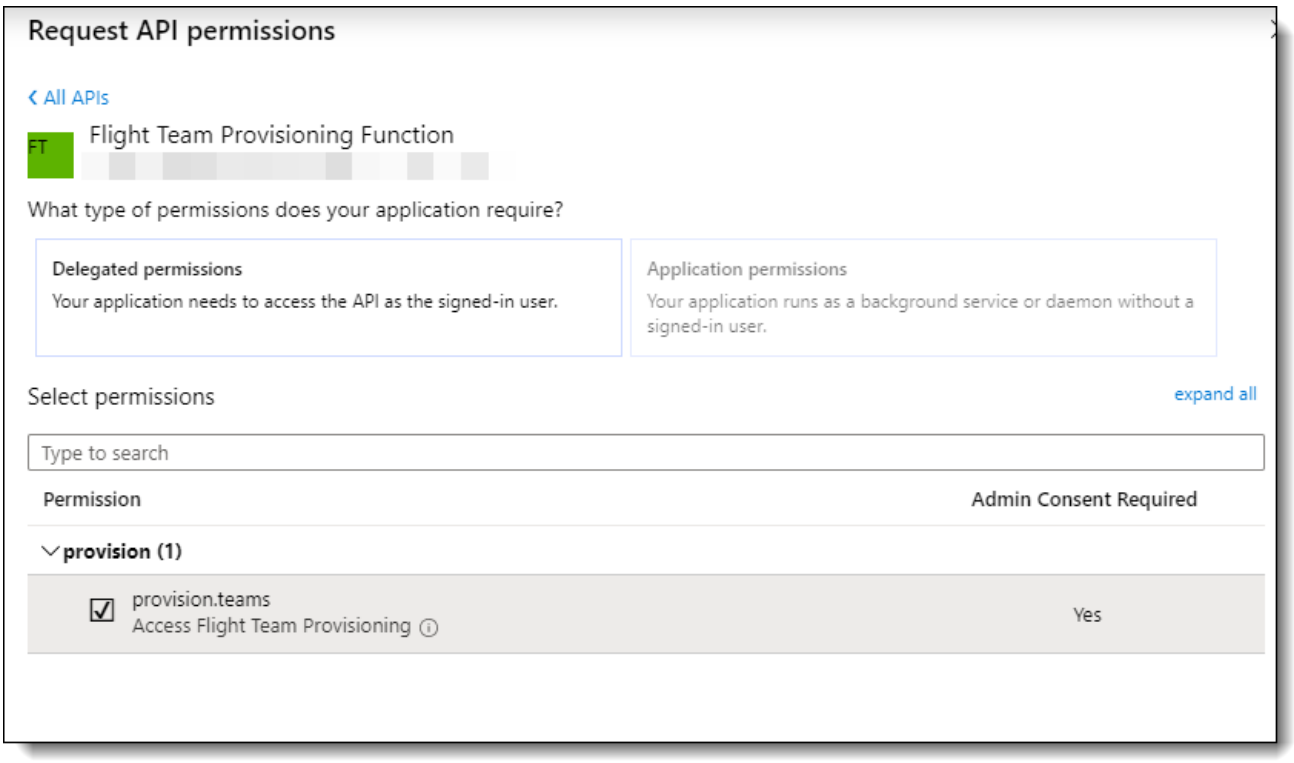
7. > **IMPORTANT:**
   > Copy the values of **Application (client) ID** and **Directory (tenant) ID**, you will need them later.

8. On the left panel, click **API permissions**, then choose **Add a permission**.

9. Select the **My APIs** tab, then select **Flight Team Provisioning Function**.



10. Select the **provision.teams** permission and choose **Add permissions** at the bottom.

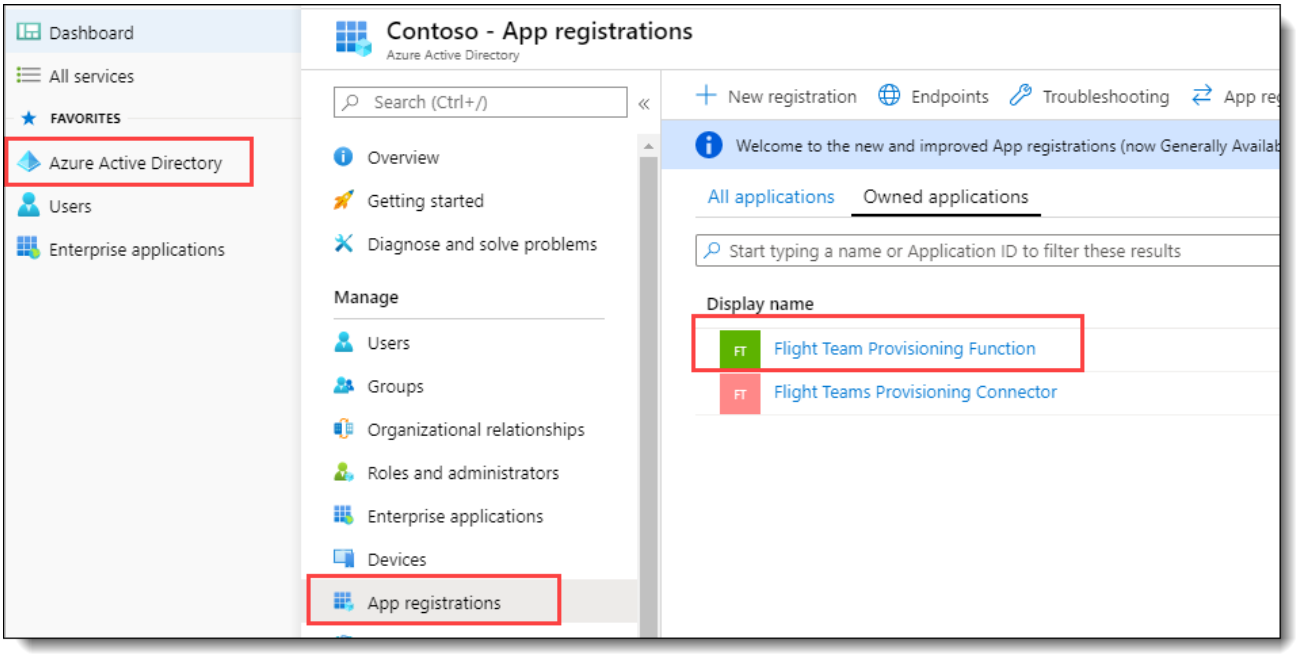


11. On the left panel, click **Certificates & secrets** under Client Secrets, then click **New client secret**.
12. Enter a description, select a duration, then choose **Add**.

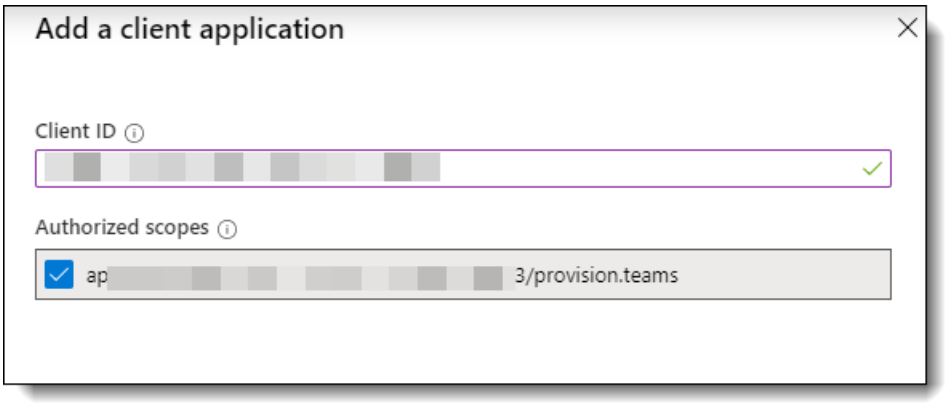
> **IMPORTANT:**
> Copy the value of the client secret, you will need them later.
> **This is the only time that you will be able to obtain this secret.**

Update the app registration for the Azure Function

1. Navigate to App Registrations in **Azure Active Directory** and click on the **Flight Team Provisioning Function** app registration.



2. On the left panel click **Expose an API** under Authorized client applications.
3. Click **Add a client application**.
4. Set **Client ID** to the application ID of the **Flight Team Provisioning Connector** app registration.
5. Select the **provision.teams** scope under Authorized scopes.



6. Chose **Add application**.

Exercise 4: Configure .Net project

Introduction

In this exercise you will configure the .Net project provided by your instructor. You will have to enter certain key

values regarding your tenant and registered applications for the solution to work.

☑ Objectives

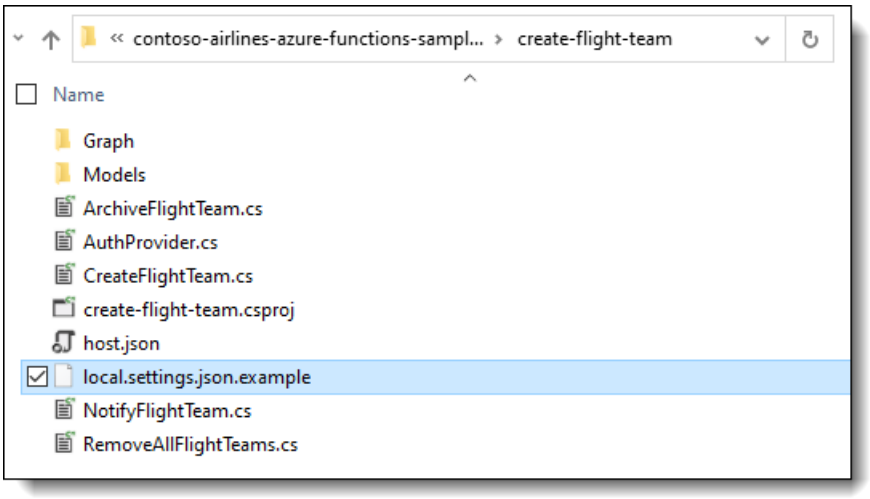
After completing this exercise, you will be able to configure .Net project for O365 consumption.

☑ Prerequisites

- Solution file provided by your instructor.

☑ Tasks

1. Extract the Zip file accompanying this lab manual (provided by your instructor), and browse to the extracted folder.
2. Rename the **create-flight-team\local.settings.json.example** file to **local.settings.json**.



3. Open the file with a text editor
4. Replace **YOUR TENANT ID HERE** with the **Directory (tenant) ID** value that you copied previously.
5. Replace **YOUR TENANT NAME HERE** with your tenant's domain name.
6. Replace **YOUR APP ID HERE** with the application ID for the **Flight Team Provisioning Function** app registration.
7. Replace YOUR **APP SECRET HERE** with the client secret for the **Flight Team Provisioning Function** app registration.

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "AzureWebJobsDashboard": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet",
    "TenantId": "YOUR TENANT ID HERE",
    "TenantName": "YOUR TENANT NAME HERE (contoso.onmicrosoft.com)",
    "AppId": "YOUR APP ID HERE",
    "AppSecret": "YOUR APP SECRET HERE",
    "TeamAppToInstall": "1542629c-01b3-4a6d-8f76-1938b779e48d",
    "FlightAdminSite": "FlightAdmin",
    "FlightLogFile": "Flight Log.docx",
    "NotificationAppId": "YOUR CROSS DEVICE EXPERIENCE HOST NAME"
  }
}
```

8. Save the file.

Exercise 5: Create schema extension

Introduction

In this exercise will create a schema extension with MS Graph Explorer.

This solution uses a MS Graph **schema extension** resource type to save the item ID of the SharePoint list item that initiated the team provisioning onto the O365 group. This is used to find the O365 Group later when archiving the team or notifying members of a change to the departure gate.

> **Creating schema extensions** is currently only supported with Delegated permissions. Therefore, we will be leveraging **Microsoft Graph Explorer** as it is the most convenient way to create the extension.

Objectives

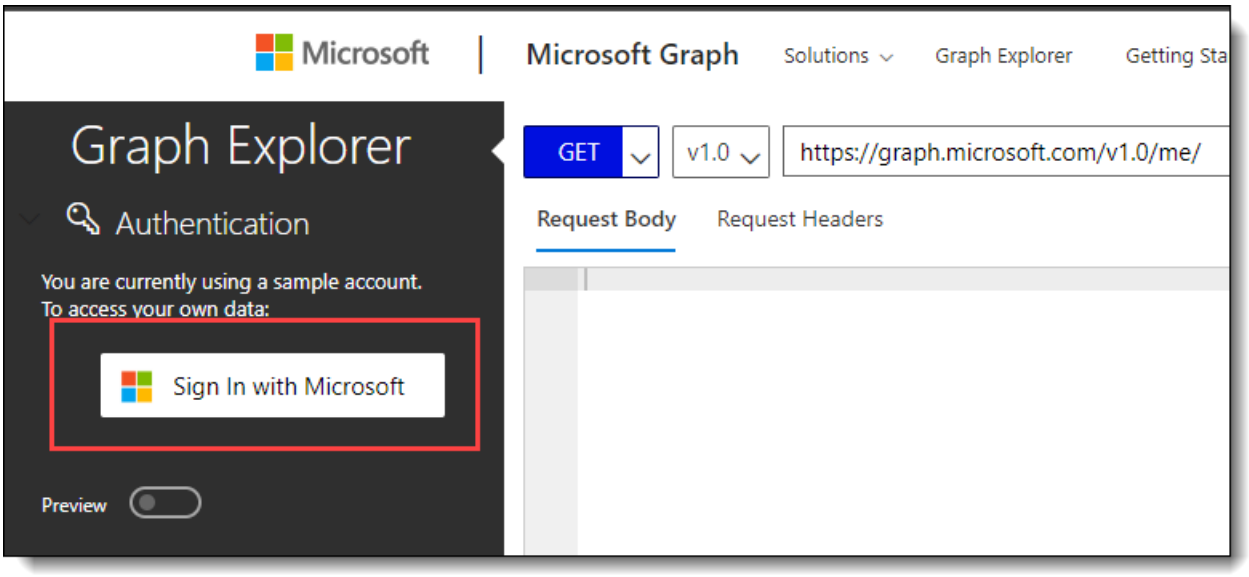
After completing this exercise, you will be able to: Create a schema extension using Microsoft Graph.

Prerequisites

- O365 Tenant
- O365 admin account
- Microsoft Graph Explorer

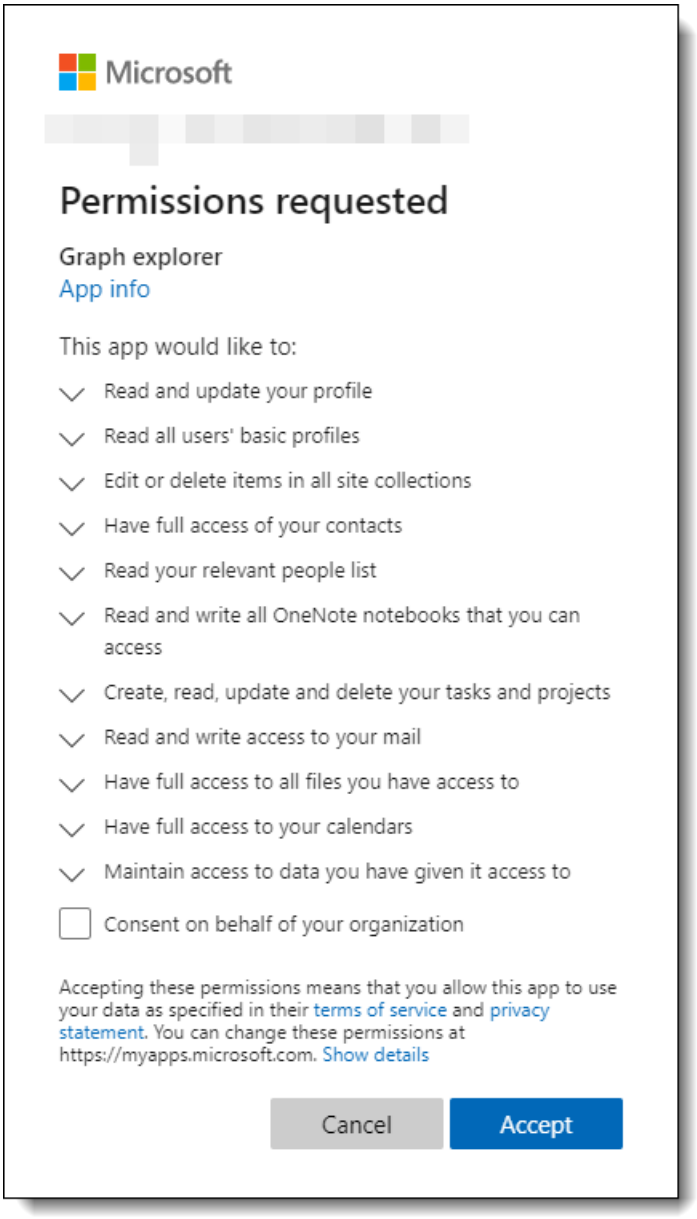
Tasks

1. Go to **Microsoft Graph Explorer** and click on **Sign In with Microsoft**.

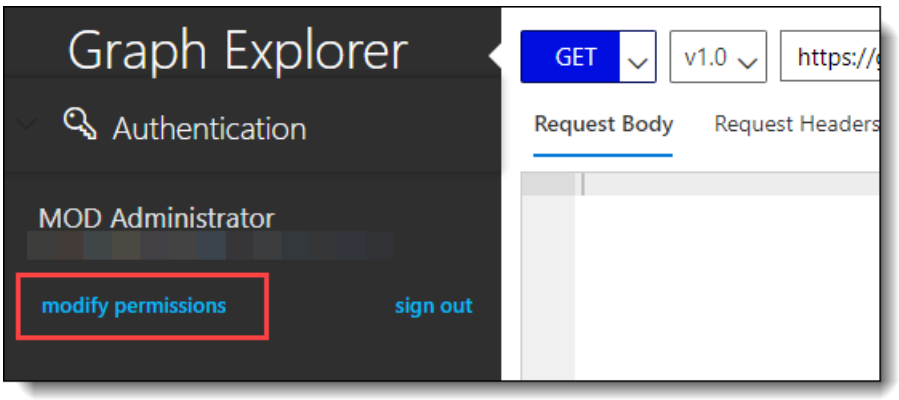


2. Log in with your tenant's administrator account (accept the consent prompt if presented with one)

> ℹ If you are using Microsoft Graph against a specific tenant for the first time, you will get a consent prompt.



3. Click the **modify permissions** link



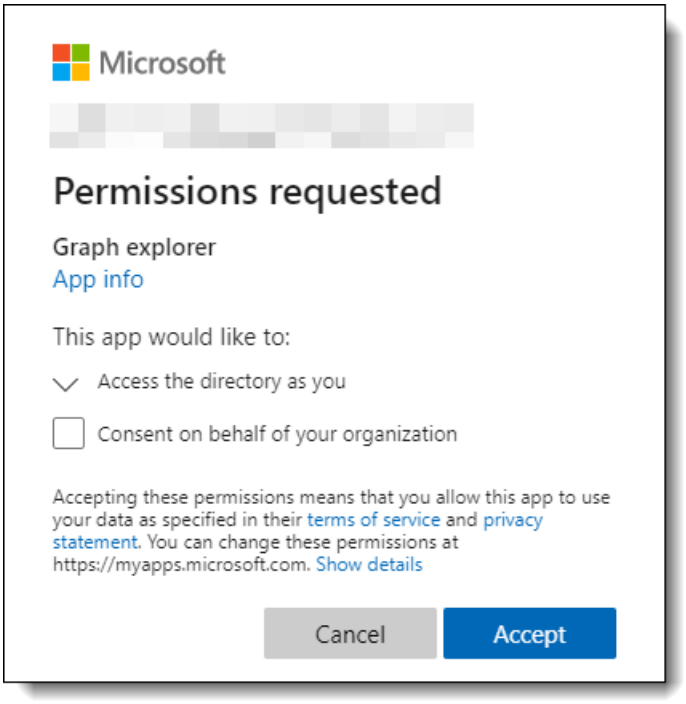
4. Select the **Directory.AccessAsUser.All** permission.

> **NOTE:** If the **Directory.AccessAsUser.All** permissions is disabled and displays **Consented**, you've

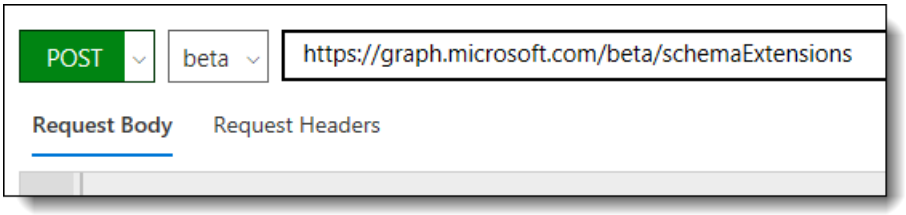
> already enabled this permission.

5. Select **Modify Permissions**. This might trigger a new admin consent prompt. If so, accept it.

   > ℹ Every time you add new permissions to Microsoft Graph, you will get a consent prompt.



6. Once in Microsoft Graph Explorer, change the method to **POST**, the version to **beta**, and set the URL to **https://graph.microsoft.com/beta/schemaExtensions**.



7. In the Request Body, add the following JSON, replacing **app-id-for-flight-team-provisioning-function** with the application ID for your **Flight Team Provisioning Function** app registration.

```
{
    "id":"contosoFlightTeam",
    "owner": "app-id-for-flight-team-provisioning-function",
    "description": "Contoso Flight Team Provisioning extensions",
    "targetTypes": [
        "Group"
    ],
    "properties": [
        {
            "name": "sharePointItemId",
            "type": "Integer"
        }
    ]
}
```

8. Click **Run Query**.
9. In the Response Preview, locate the value of the **id** property.

> **IMPORTANT:** Copy the **id** value. You will need it later.

Exercise 6: Deploy Function app sample solution

Introduction

In this exercise you will deploy the Function app solution to Azure. Although it's not the intention of this POC to analyze the code thoroughly, this lesson will highlight the key pieces of the solution so that you can understand key Microsoft Graph concepts like:

- Authentication
- OAuth flow
- Graph resources
- Calling MS Graph

Objectives

After completing this lab, you will be able to:

- Deploy the Function app solution to Azure.
- Understand key pieces of the code and how the different pieces of the solution interact with each other

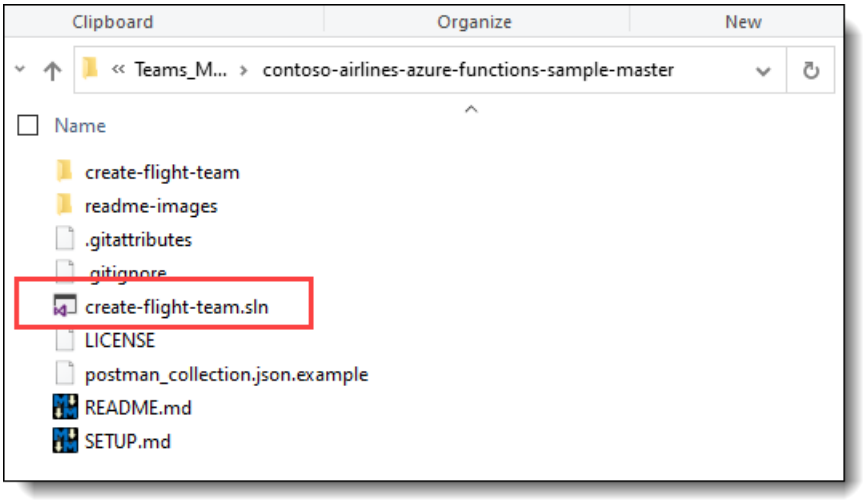
☑ Prerequisites

- Visual Studio solution provided by your administrator.
- Microsoft Visual Studio
- Azure subscription with a provisioned Function App

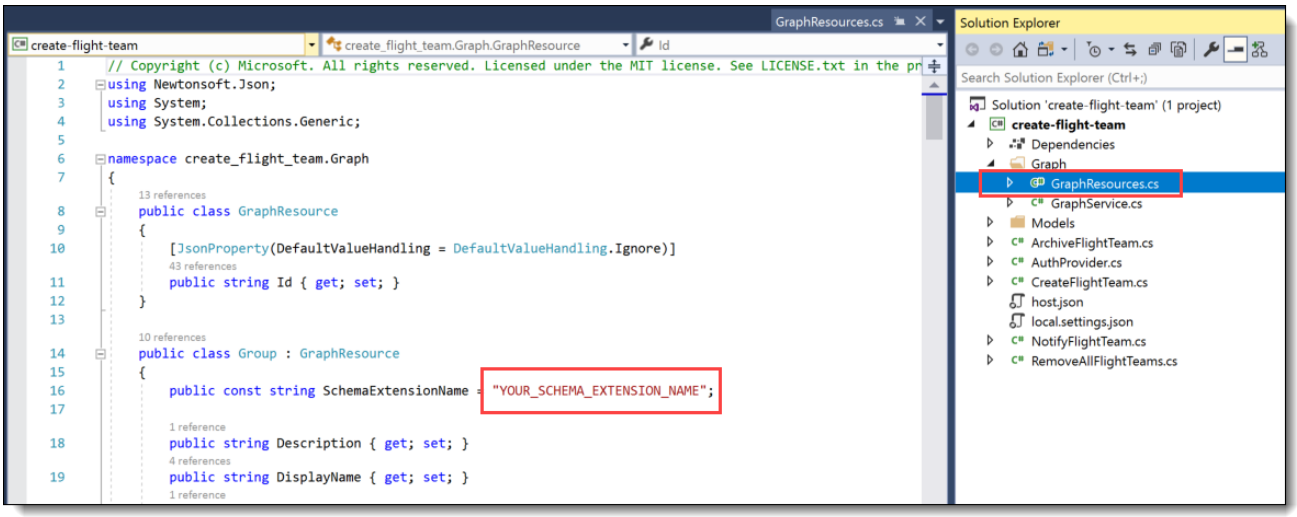
☑ Tasks

Deploy Azure Function solution to Azure

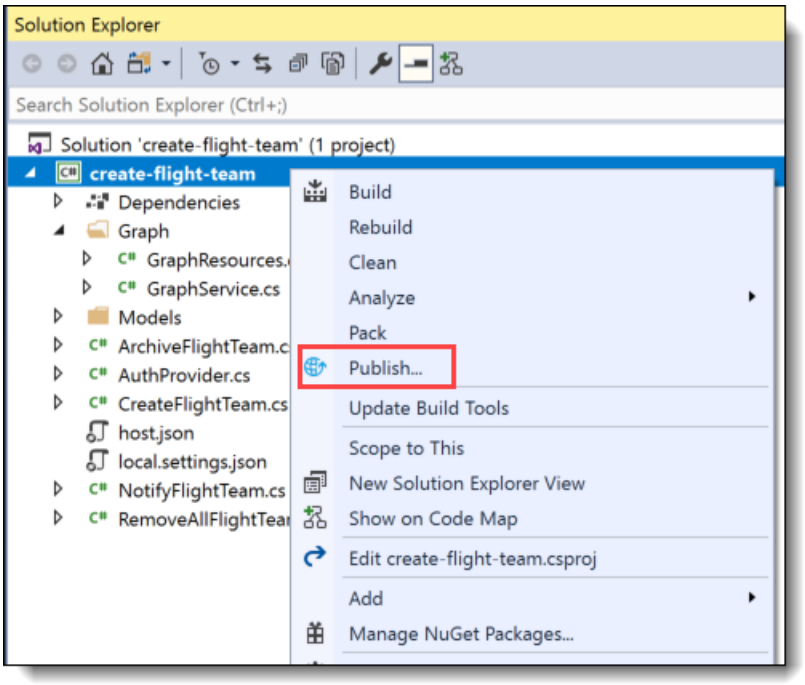
1. Open the sample solution provided by your instructor (**create-flight-team.sln**) in Visual Studio



2. Open the **create-flight-team/Graph/GraphResources.cs** file and replace **YOUR_SCHEMA_EXTENSION_NAME** with the **id** value of the schema extension you copied from last exercise.

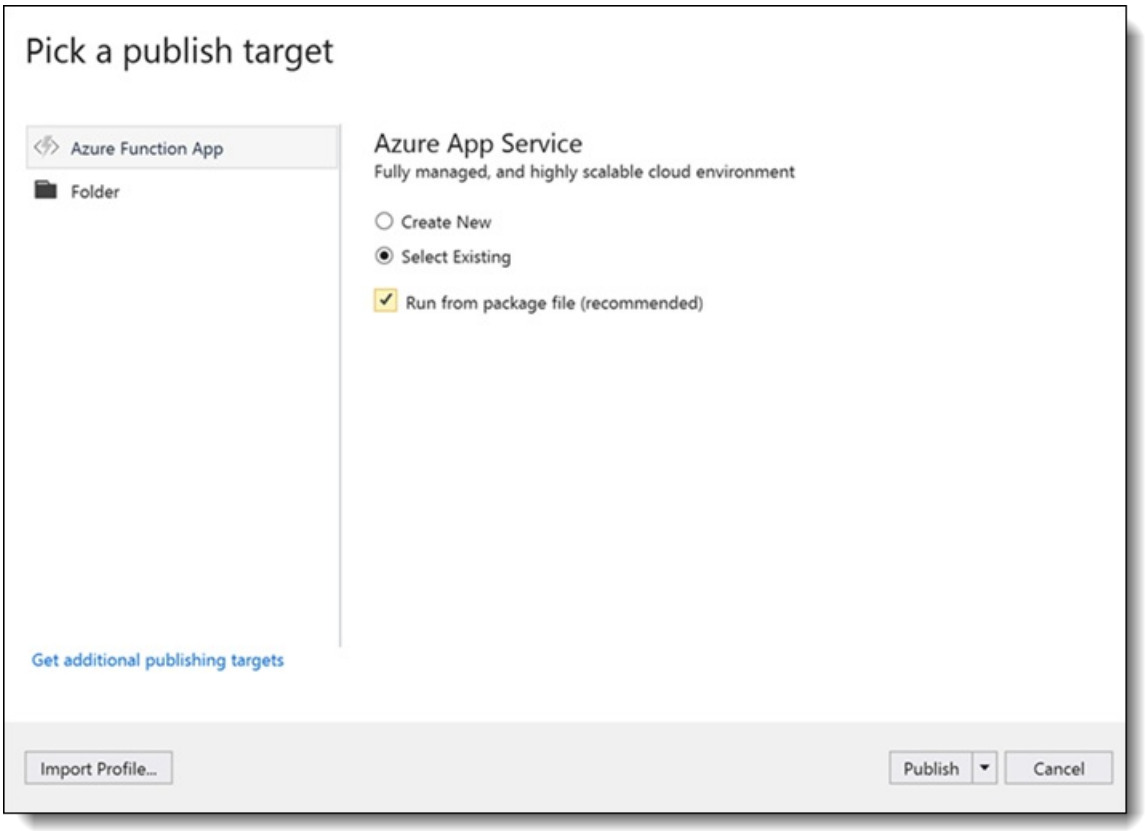


3. In Visual Studio solution explorer window, right click on the project and click **Publish...**

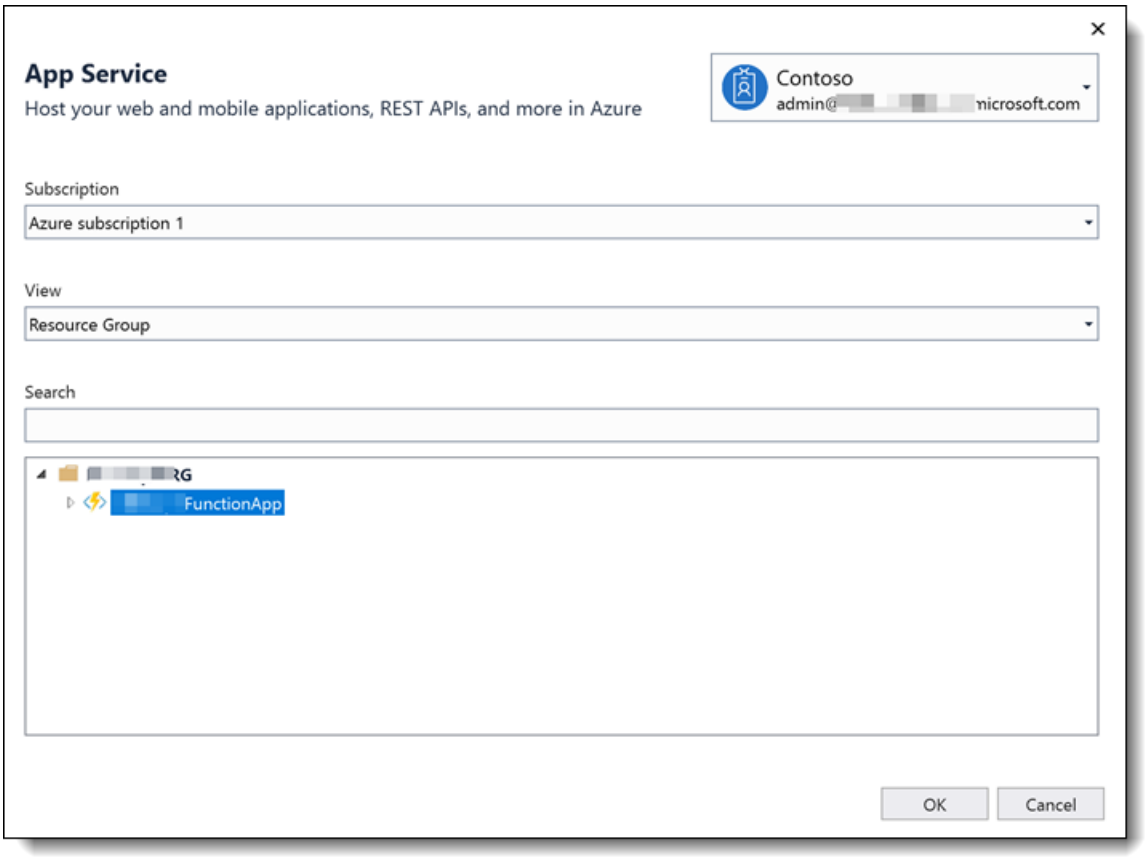


4. **Select existing** subscription and check **Run from package file (recommended)**.

> Login to your Azure tenant if required.



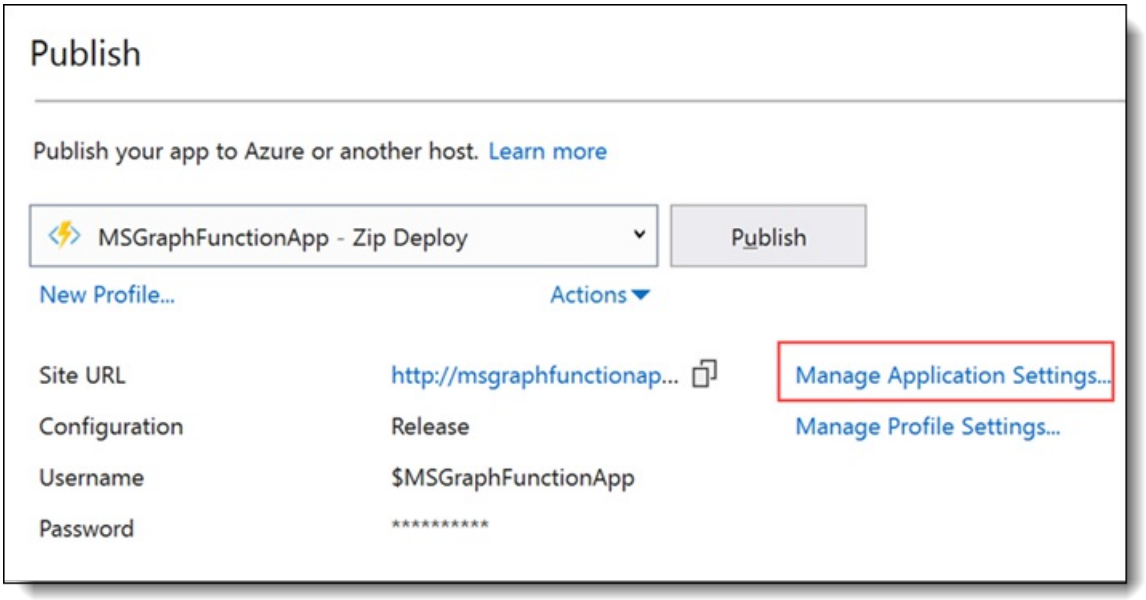
5. Select the Function App you created in Exercise 3 and click **OK**.



6. Once the solution has been published, you will need to upload the local settings to Azure.

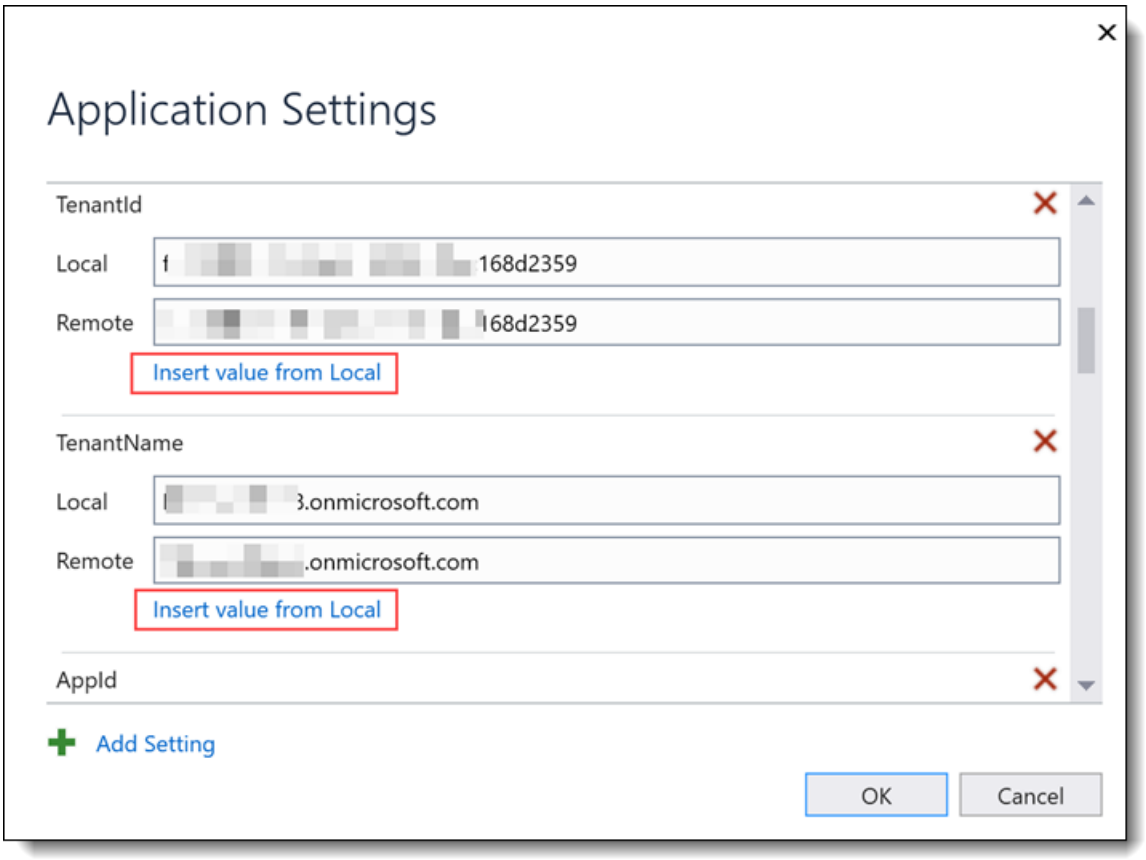
> ℹ The reason for this is that any settings you added in the **local.settings.json** file must be also added to the function app in Azure. These settings aren't uploaded automatically when you publish the project.

The easiest way to upload the required settings to your function app in Azure is to use the **Manage Application Settings...** link that is displayed after you successfully publish your project.



Click on **Insert value from Local** in each of the following settings to copy them to Azure.

- TenantID
- TenantName
- AppId
- AppSecret
- TeamAppToInstall
- FlightAdminSite
- FlightLogFile



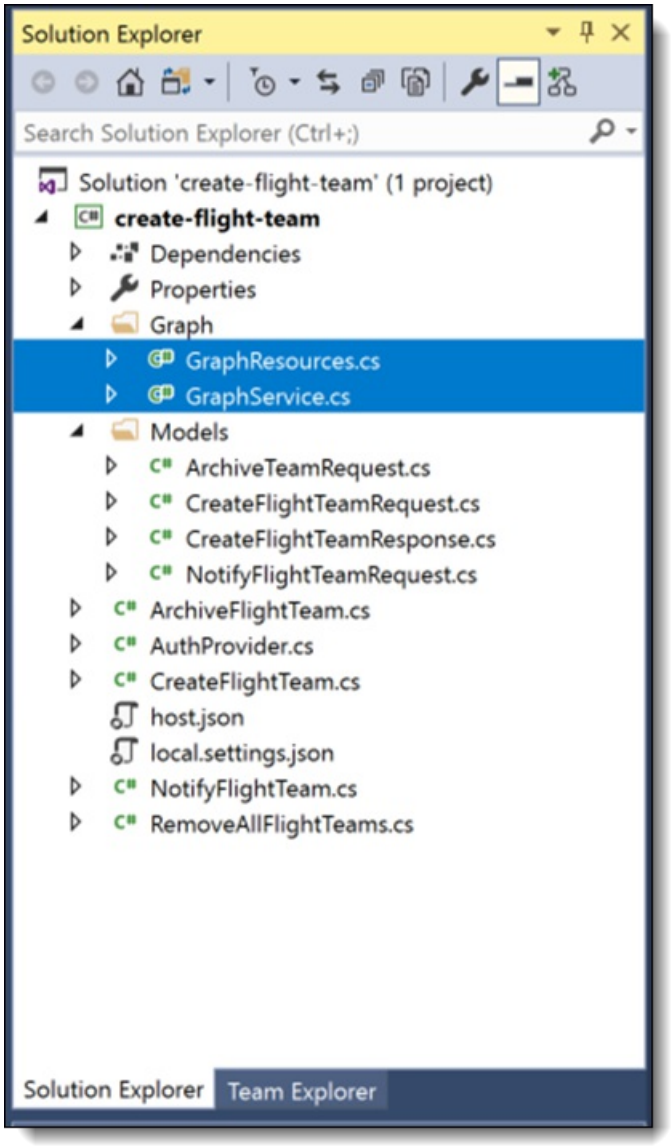
Click **OK** once you are done.

Visual Studio Solution walk-through

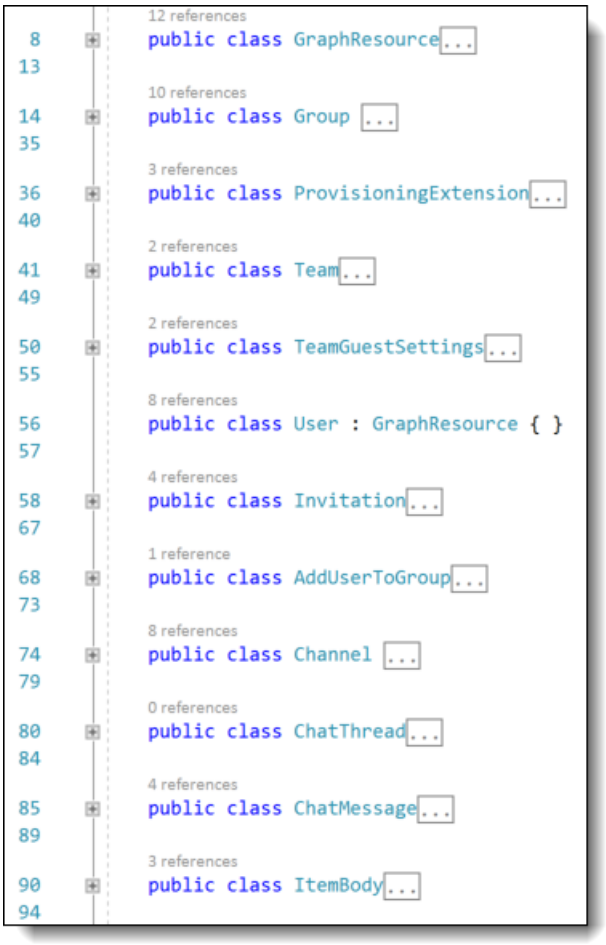
Now that you have deployed the Azure functions to you Azure environment let's walk through the key pieces of the solution so that you have a better understanding on how Microsoft Graph is being leveraged. This solution uses the **Microsoft Authentication Library (MSAL)** to help acquire the much needed access token when querying Microsoft Graph. However, it does not make use of **Microsoft Graph SDK**. Using MS Graph SDK will ease the development efforts but this particular solution aims to show how MS Graph can be called without the need of the Microsoft Graph SDK.

Take a look at the files within the solution. Inside **create-flight-team/Graph** you can see a couple of files:

GraphResources.cs and **GraphService.cs**.



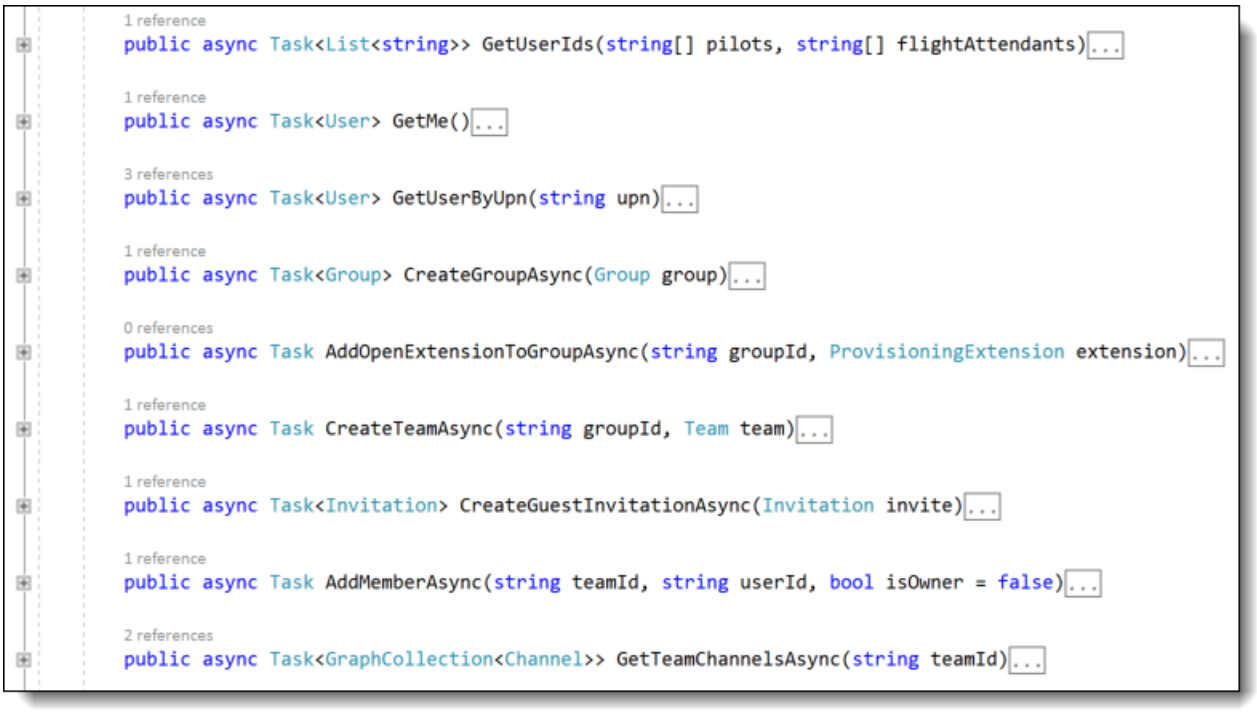
- **GraphResources.cs**: This file contains classes that define several Graph resources and it's properties. For example:



Here's an example of the **Group** resource and its properties:

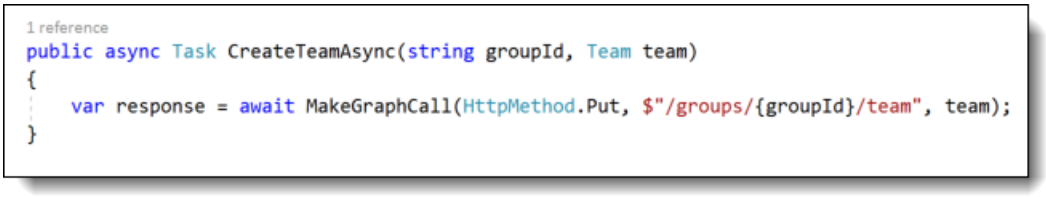
```
10 references
public class Group : GraphResource
{
    public const string SchemaExtensionName = "ext19scbxf9_contosoFlightTeam";

    1 reference
    public string Description { get; set; }
    4 references
    public string DisplayName { get; set; }
    1 reference
    public string[] GroupTypes { get; set; }
    1 reference
    public bool MailEnabled { get; set; }
    1 reference
    public string MailNickname { get; set; }
    1 reference
    public bool SecurityEnabled { get; set; }
    1 reference
    public string Visibility { get; set; }

    [JsonProperty(PropertyName = SchemaExtensionName, DefaultValueHandling = DefaultValueHandling.Ignore)]
    1 reference
    public ProvisioningExtension Extension { get; set; }

    [JsonProperty(PropertyName = "members@odata.bind", DefaultValueHandling = DefaultValueHandling.Ignore)]
    1 reference
    public List<string> Members { get; set; }

    [JsonProperty(PropertyName = "owners@odata.bind", DefaultValueHandling = DefaultValueHandling.Ignore)]
    1 reference
    public List<string> Owners { get; set; }
}
```

- **GraphService.cs**: This file contains a list of methods in charge of querying the different resources or entities of MS Graph:



```
1 reference
public async Task<List<string>> GetUserIds(string[] pilots, string[] flightAttendants)...

1 reference
public async Task<User> GetMe()...

3 references
public async Task<User> GetUserByUpn(string upn)...

1 reference
public async Task<Group> CreateGroupAsync(Group group)...

0 references
public async Task AddOpenExtensionToGroupAsync(string groupId, ProvisioningExtension extension)...

1 reference
public async Task CreateTeamAsync(string groupId, Team team)...

1 reference
public async Task<Invitation> CreateGuestInvitationAsync(Invitation invite)...

1 reference
public async Task AddMemberAsync(string teamId, string userId, bool isOwner = false)...

2 references
public async Task<GraphCollection<Channel>> GetTeamChannelsAsync(string teamId)...
```

Most of these methods will call the **MakeGraphCall** method that also exists in **GraphService.cs**. Here's an example of the **CreateTeamAsync** method:



```
1 reference
public async Task CreateTeamAsync(string groupId, Team team)
{
    var response = await MakeGraphCall(HttpMethod.Put, $"/groups/{groupId}/team", team);
}
```

The **MakeGraphCall** issues a standard http request through an **HttpClient** object (**System.Net.Http.HttpClient**) to the appropriate resource passed as a parameter.

> **❗ IMPORTANT:**
> Notice that this POC queries the **Beta** endpoint of MS Graph. Therefore this solution should not be implemented in production environments, as resources and entities in the **Beta** endpoint are subject to change.

```
31 references
private async Task<HttpResponseMessage> MakeGraphCall(HttpMethod method, string uri, object body = null, int retries = 0, string version = "beta")
{
    // Initialize retry delay to 3 secs
    int retryDelay = 3;

    string payload = string.Empty;

    if (body != null && (method != HttpMethod.Get || method != HttpMethod.Delete))
    {
        // Serialize the body
        payload = JsonConvert.SerializeObject(body, jsonSettings);
    }

    if (logger != null)
    {
        logger.LogInformation($"MakeGraphCall Request: {method} {uri}");
        logger.LogInformation($"MakeGraphCall Payload: {payload}");
    }

    do
    {
        // Create the request
        var request = new HttpRequestMessage(method, $"{graphEndpoint}{version}{uri}");


        if (!string.IsNullOrEmpty(payload))
        {
            request.Content = new StringContent(payload, Encoding.UTF8, "application/json");
        }

        // Send the request
        var response = await httpClient.SendAsync(request);

        if (!response.IsSuccessStatusCode)
```

In order to create a **GraphService** object, you need to pass an **access token** as a paramteer:

```
13 references
public class GraphService
{
    private static readonly string graphEndpoint = "https://graph.microsoft.com/";

    private readonly string accessToken = string.Empty;
    private HttpClient httpClient = null;
    private readonly JsonSerializerSettings jsonSettings =
        new JsonSerializerSettings { ContractResolver = new CamelCasePropertyNamesContractResolver() };

    private ILogger logger = null;

    4 references
    public GraphService(string accessToken, ILogger log = null)
    {
        this.accessToken = accessToken;
        httpClient = new HttpClient();
        httpClient.DefaultRequestHeaders.Authorization =
            new AuthenticationHeaderValue("Bearer", accessToken);
        httpClient.DefaultRequestHeaders.Accept.Add(
            new MediaTypeWithQualityHeaderValue("application/json"));

        logger = log;
    }
```

AuthProvider.cs handles the authentication piece and is in charge of obtaining valid access token (which will be used by the **GraphService** object). The **AuthProvider** class leverages **Microsoft.Identity.Client library (MSAL)** to instantiate this **ConfidentialClientApplication**:

```
4 references
public static class AuthProvider
{
    private static readonly string appId = Environment.GetEnvironmentVariable("AppId");
    private static readonly string tid = Environment.GetEnvironmentVariable("TenantId");
    private static readonly string authority = $"https://login.microsoftonline.com/{tid}";
    private static readonly ClientCredential clientCreds = new ClientCredential(
        Environment.GetEnvironmentVariable("AppSecret"));
    private static readonly string[] scopes = { "https://graph.microsoft.com/.default" };

    4 references
    public static async Task<string> GetTokenOnBehalfOfAsync(string authHeader)
    {
        if (string.IsNullOrEmpty(authHeader))
        {
            throw new MsalException("missing_auth", "Authorization header is not present on request.");
        }

        // Parse the auth header
        var parsedHeader = AuthenticationHeaderValue.Parse(authHeader);
        if (parsedHeader.Scheme.ToLower() != "bearer")
        {
            throw new MsalException("invalid_scheme", "Authorization header is missing the 'bearer' scheme.");
        }

        // Create an assertion based on the provided token
        var userAssertion = new UserAssertion(parsedHeader.Parameter);

        var confidentialClient = new ConfidentialClientApplication(appId, authority, clientCreds, null, null);

        // Exchange the provided token for a Graph token
        var result = await confidentialClient.AcquireTokenOnBehalfOfAsync(scopes, userAssertion, authority);

        return result.AccessToken;
    }
}
```

As you can see from the above code, and as discussed previously, we are using the **on-behalf-of** OAuth flow. For this, a **ConfidentialClientApplication** is needed. The access token is requested by calling the **AcquireTokenOnBehalfOfAsync** method.

Finally **CreateFlightTeam.cs**, **NotifyFlightTeam.cs** and **ArchiveFlightTeam.cs** are the files containing code in charge of creating, updating or deleting a MS Team depending on the action performed in the SharePoint flight list.

Let's focus on **CreateFlightTeam.cs** as it involves more operations.

> **It's encouraged that you take a look at the other classes and try to follow the code.**

When the flight admin enters the details of a new flight in the SharePoint flight list, it will cause the Power Automate **Create Team** flow to trigger (to be configured in the next exercise). This flow will trigger **CreateFlightTeam.cs** through a custom connector.

CreateFlightTeam.cs has the following methods and properties:

```
0 references
public static class CreateFlightTeam
{
    private static readonly string teamAppId = Environment.GetEnvironmentVariable("TeamAppToInstall");
    private static readonly string flightAdminSite = Environment.GetEnvironmentVariable("FlightAdminSite");
    private static readonly string flightLogFile = Environment.GetEnvironmentVariable("FlightLogFile");
    private static readonly string tenantName = Environment.GetEnvironmentVariable("TenantName");

    private static ILogger logger = null;

    [FunctionName("CreateFlightTeam")]
    0 references
    public static async Task<IActionResult> Run([HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = null)]HttpRequest req, IL

    1 reference
    private static async Task ProvisionTeamAsync(string accessToken, CreateFlightTeamRequest request)...

    1 reference
    private static async Task<Group> CreateUnifiedGroupAsync(GraphService graphClient, CreateFlightTeamRequest request)...

    1 reference
    private static async Task<Channel> InitializeTeamInGroupAsync(GraphService graphClient, string groupId, string welcomeMessage)...

    1 reference
    private static async Task CopyFlightLogToTeamFilesAsync(GraphService graphClient, string groupId)...

    1 reference
    private static async Task CreatePreflightPlanAsync(GraphService graphClient, string groupId, string channelId, DateTimeOffset de

    1 reference
    private static async Task<SharePointList> CreateChallengingPassengersListAsync(GraphService graphClient, string groupId, string

    1 reference
    private static async Task CreateSharePointPageAsync(GraphService graphClient, string groupId, float flightNumber)...

    1 reference
    private static string GetTimestamp()...
}
```

Here are the sequence of events:

- **AuthProvider.cs** is used to get an access token. If it succeeds, it calls **ProvisionTeamAsync**.
- **ProvisionTeamAsync** makes use of the **GraphService** class and then calls **CreateUnifiedGroupAsync**:
    - **CreateUnifiedGroupAsync**: This is where and O365 Group is provisioned. Several properties of the Group (DisplayName, Visibility, Members, Owners, etc.) are also configured here. This is where the schema extension that you previously configured is used. Invitation to a specified guest user is also sent.
    - **InitializeTeamGroupAsync:** This is used to create a Team leveraging the previously created O365 Group. This method is also in charge of creating several channels (Pilots and Flight Attendants), post an initial welcome message and to install a custom application (Polly) from the store.
    - **CreatePreflightPlanAsync**: This is the method in charge of creating the Planner plan, buckets and tasks.
    - **CreateChallengingPassengersListAsync**: This will create custom columns and the Challenging Passenger List in SharePoint Online. Also, this method is in charge of adding the mentioned list as a tab in the General channel.
    - **CreateSharePointPageAsync**: This method is in charge of creating a SharePoint custom page with webparts on it.
    - **CopyFlightLogToTeamFilesAsync**: This method is in charge of copying the Flight Log.docx present in the Flight Admin SharePoint site to the new team.

Exercise 7: Configure Power Automate

Introduction

In this exercise you will create 3 flows in Power Automate: **Create Team**, **Notify Team** and **Archive Team** flows. Each of those flows is will use a custom connector that you will configure first so that your flows can communicate with Azure Functions. When creating the custom connector you will use the Postman definition provided with the POC files.

Objectives

After completing this lab, you will be able to:

- Configure custom connector in Power Automate.
- Create a flow that uses the custom connector and integrates with Microsoft Teams.

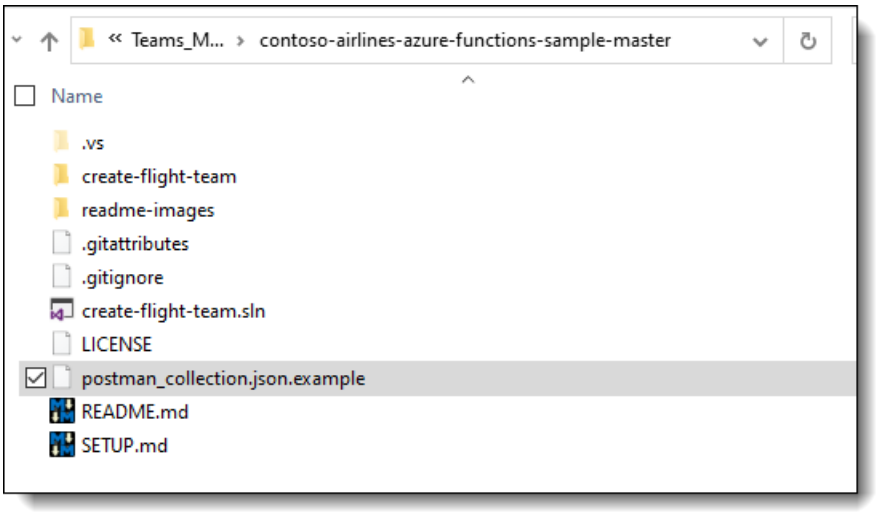
Prerequisites

- Power Automate environment
- Postman definition of the connectors (provided by your instructor)
- Azure Function App
- Microsoft Teams
- SharePoint Online Flight Admin site and list.

Tasks

Add custom connector for Azure functions

1. Go to the folder where you unzipped the provided files.
2. Rename the **postman_collection.json.example** to **postman_collection.json**, then open it in a text editor.



3. Replace all instances of **https://** `<your-azure-function-url>` with the URL to your Azure Function App (you previously copied this value in Exercise 3). There should be 3 instances to be replaced:

    i. https:// `<your-Azure-Function>` /api/CreateFlightTeam

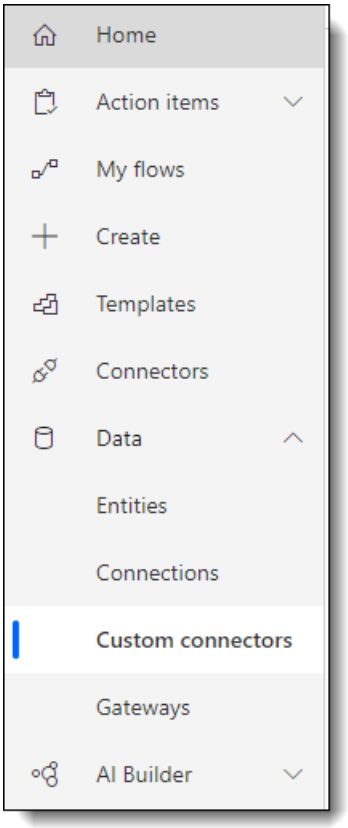
    ii. https:// `<your-azure-function-url>` /api/NotifyFlightTeam

    iii. https:// `<your-azure-function-url>` /api/ArchiveFlightTeam

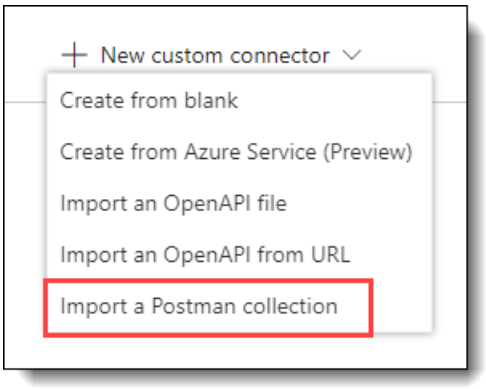
4. Save your changes.

5. Browse to **Power Automate** and sign in with your tenant's admin account.

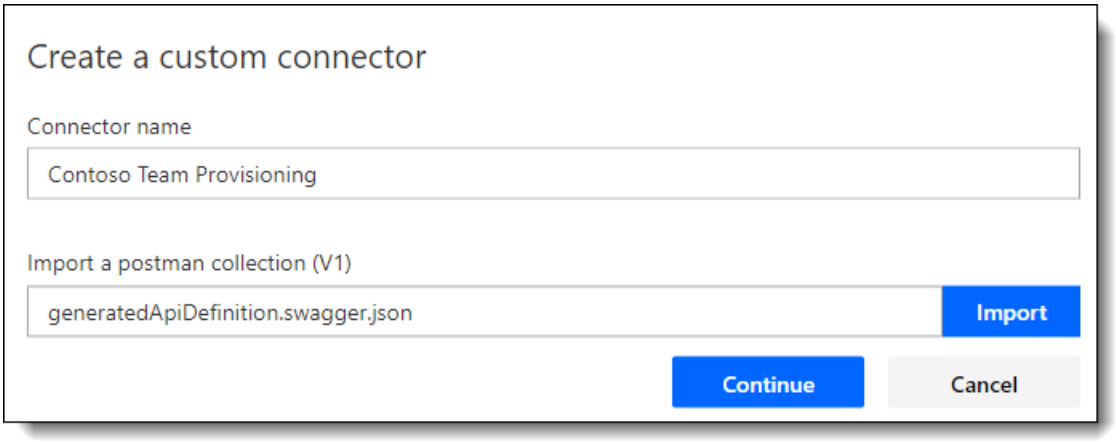
6. On the left panel, expand **Data** and click **Custom Connectors**.



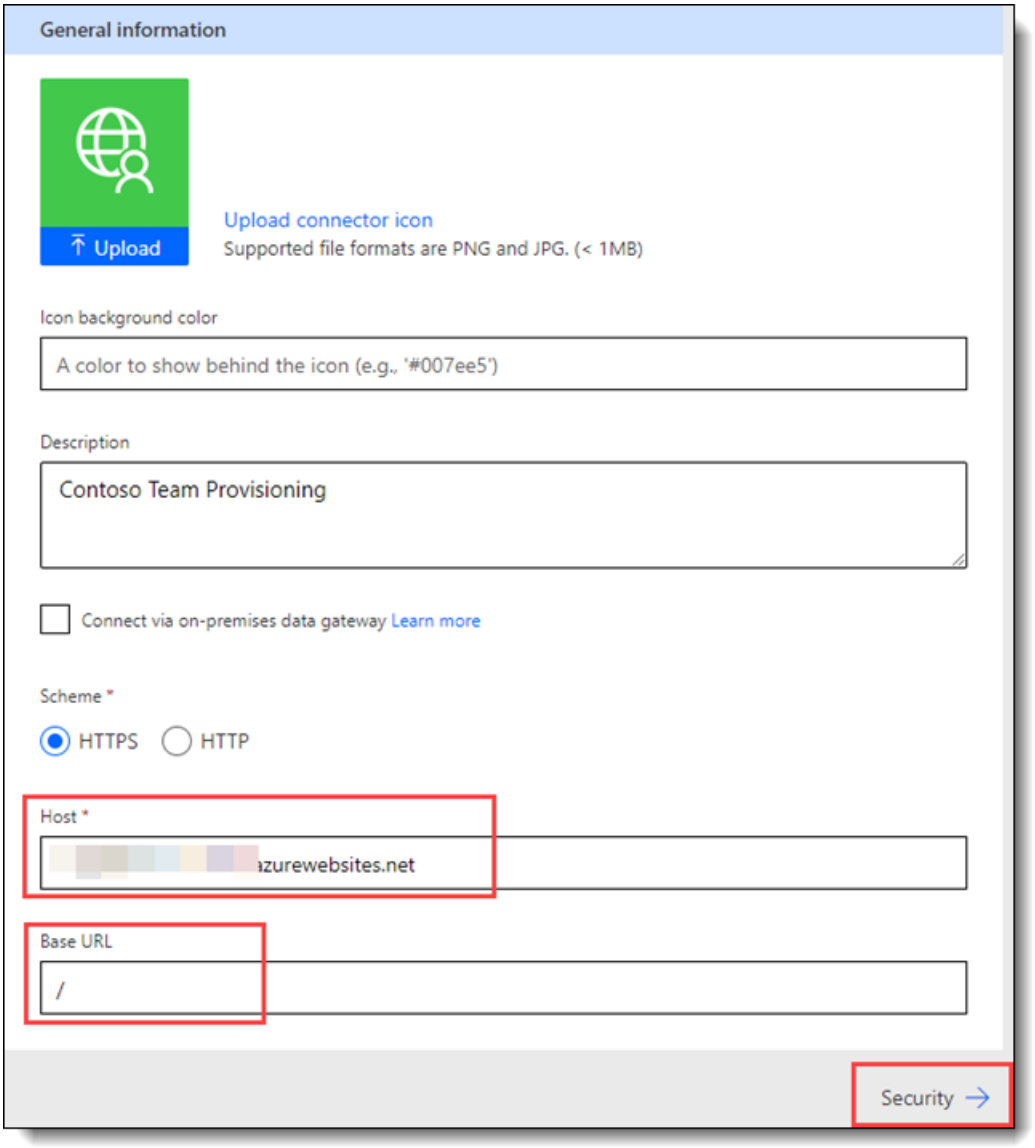
7. Click **+ New custom** connector and then choose **Import a Postman collection**.



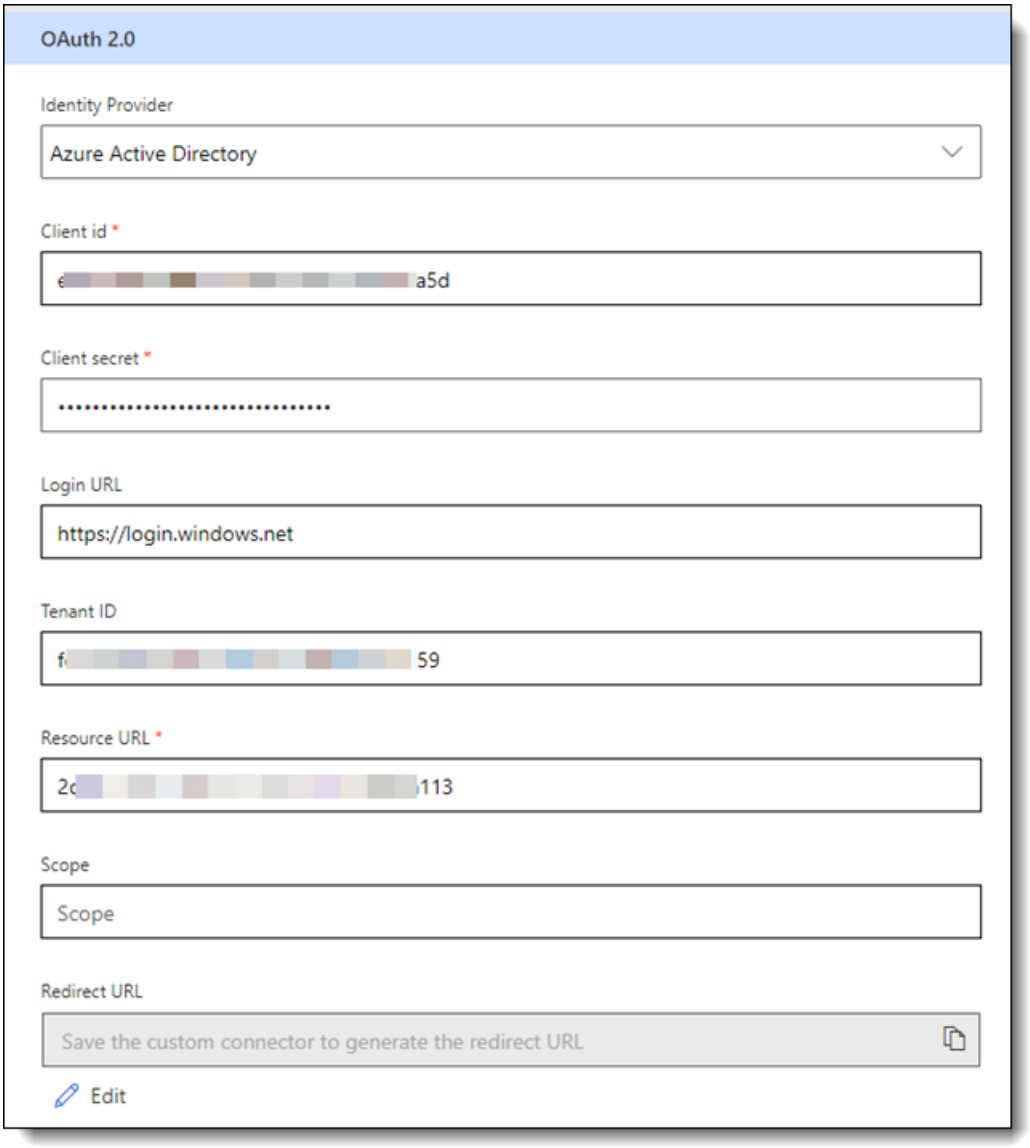
8. Enter **Contoso Team Provisioning** for Custom connector name. Click **Import** to select the **postman_collection.json** file in the **Upload a postman collection (V1)** field. Choose **Continue**.



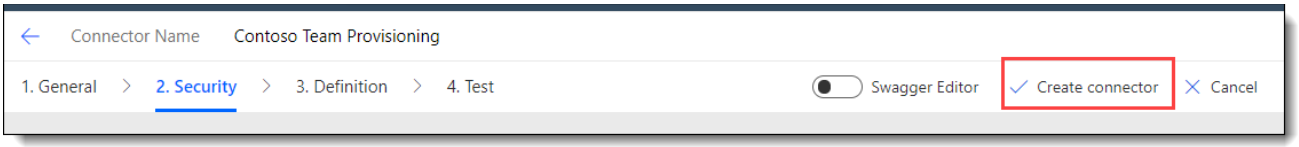
9. Under **General information**, verify that the **Host** field matches the URL of your Azure Function App, and the Base URL is set to **/**.



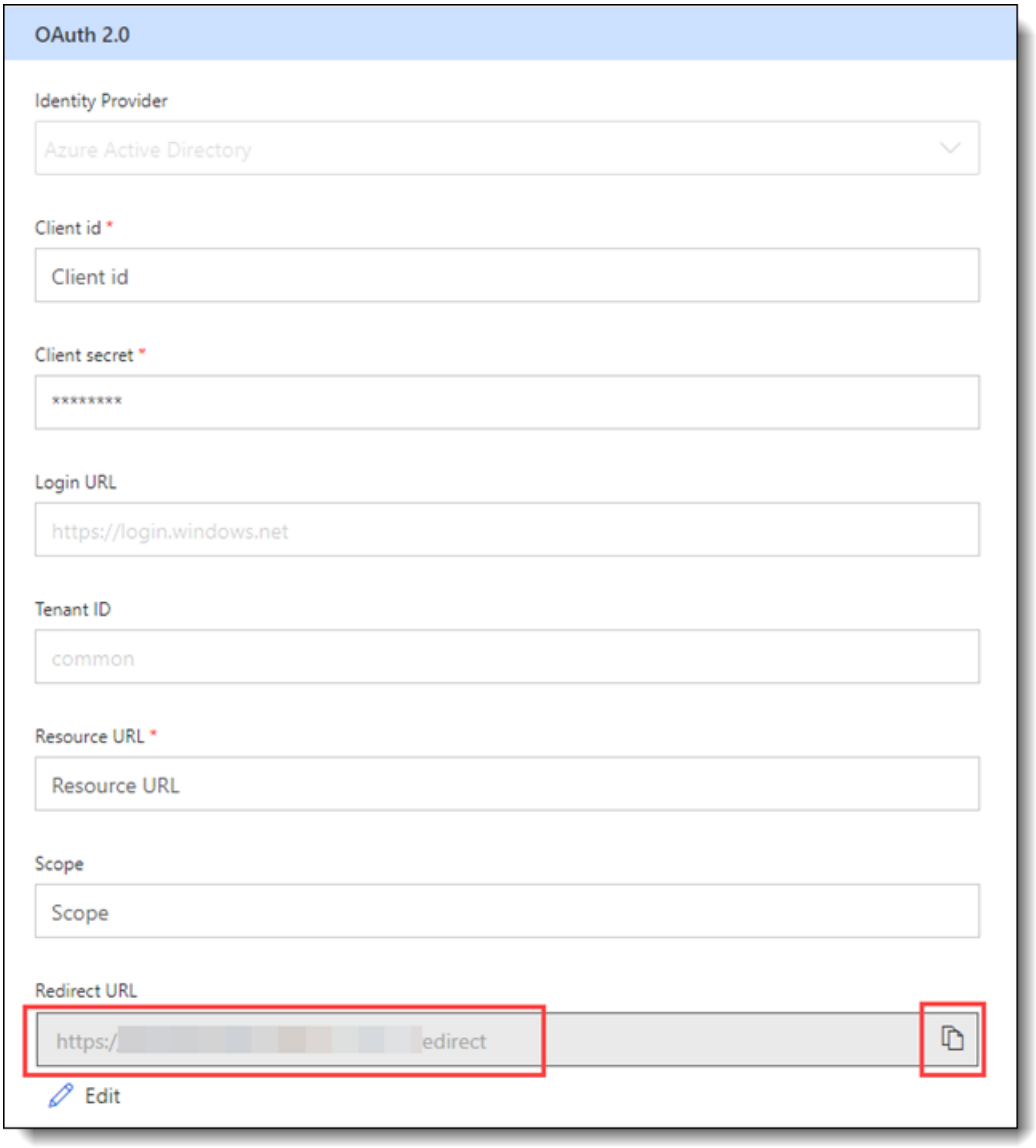
10. Click Security
11. Change Authentication type to **OAuth 2.0**, then fill in the fields as follows:
    i. Change Identity Provider to **Azure Active Directory**.
    ii. Enter the **Application (client) ID** for the **Flight Team Provisioning Connector** app registration in the **Client id** field.
    iii. Enter the **Client Secret** of the **Flight Team Provisioning Connector** app registration in the **Client secret** field.
    iv. Enter your **Directory (tenant) ID** in the **Tenant ID** field.
    v. Enter the **Application (client) ID** for the **Flight Team Provisioning Function** in the **Resource URL** field.
    vi. Leave all other fields as their default values.



12. In the top bar, click **Create connector**.

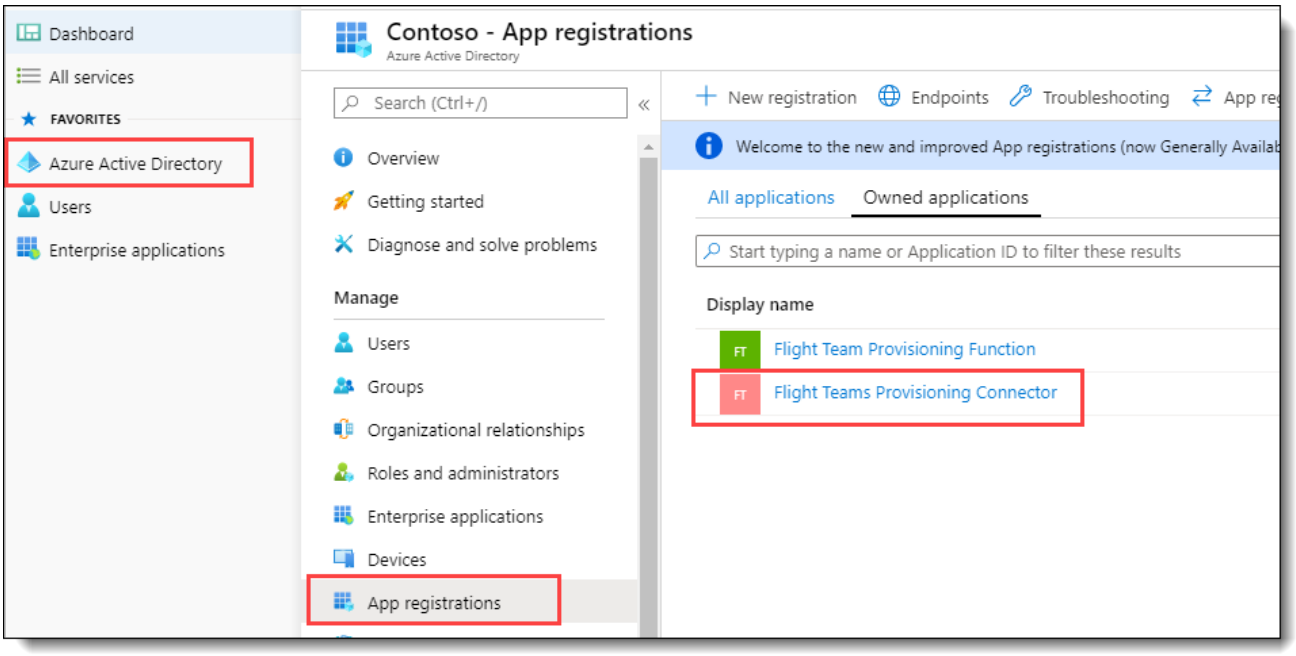


13. **IMPORTANT:**
Copy the value of the **Redirect URL**, you will need it later.

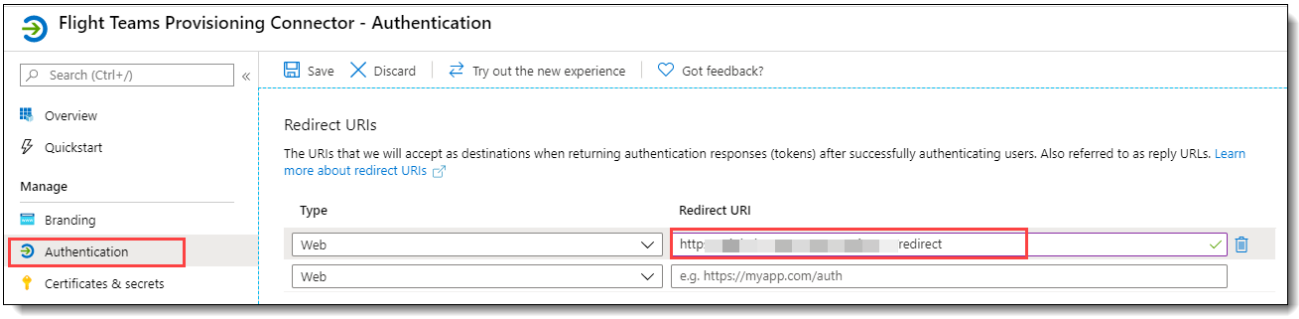


⚙ Update app registration for Microsoft Flow

1. Navigate to App Registrations in **Azure Active Directory** and click on the **Flight Team Provisioning Connector** app registration.



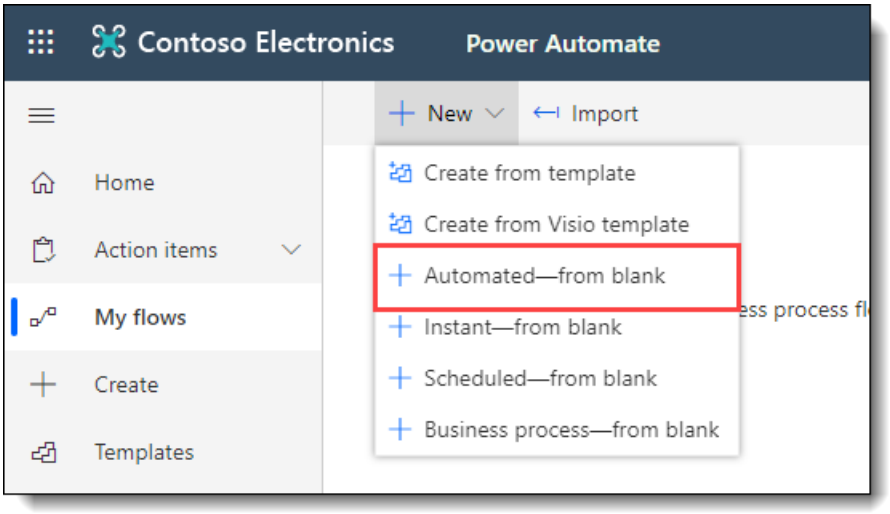
2. On the left panel, click **Authentication** under Manage.
3. Under **Redirect URIs**, make sure the Type is set to **Web**. Paste the value of Redirect URL you copied when creating the custom Flow connector in the **Redirect URI** field.



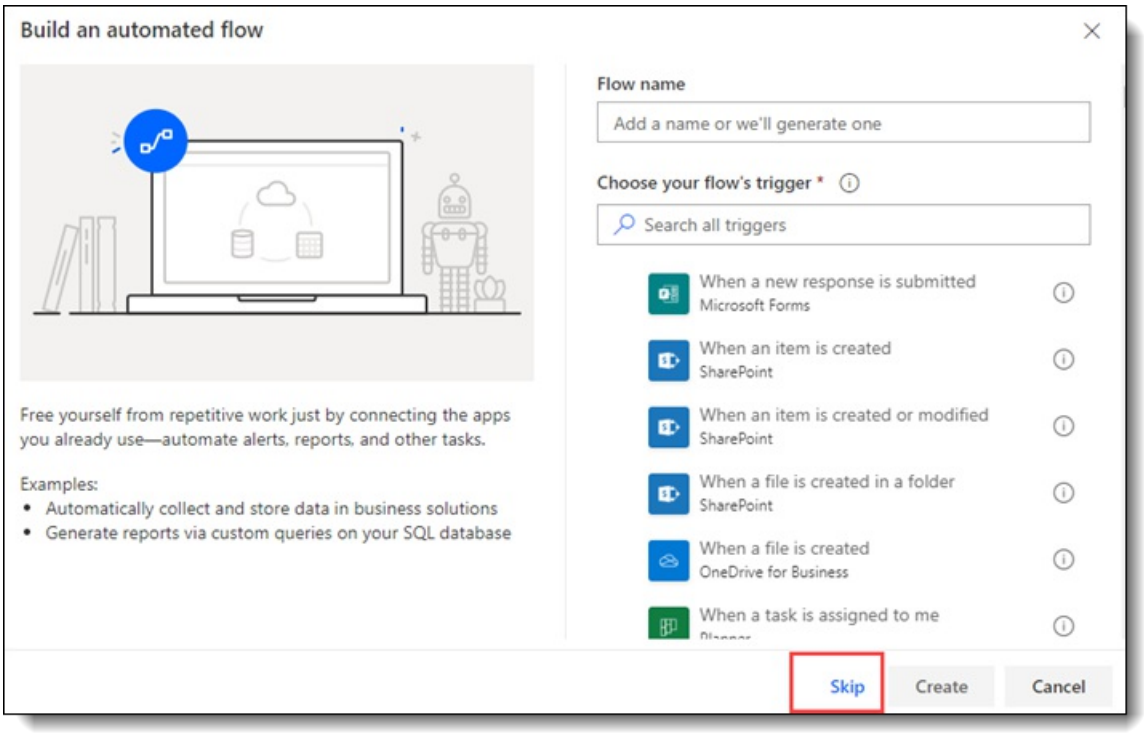
4. Click **Save** on the top bar.

⚙ Create the "Create Team" flow

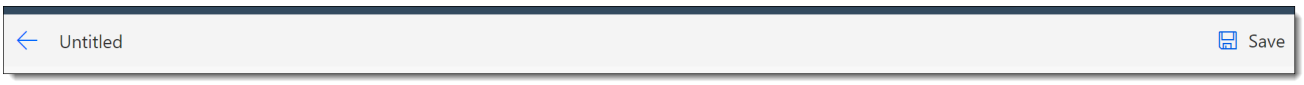
1. Browse to **Power Automate** and sign in with you tenant's admin account.
2. On the left panel, click **My flows**.
3. Click **New** at the top bar, then click **Automated - from blank**.



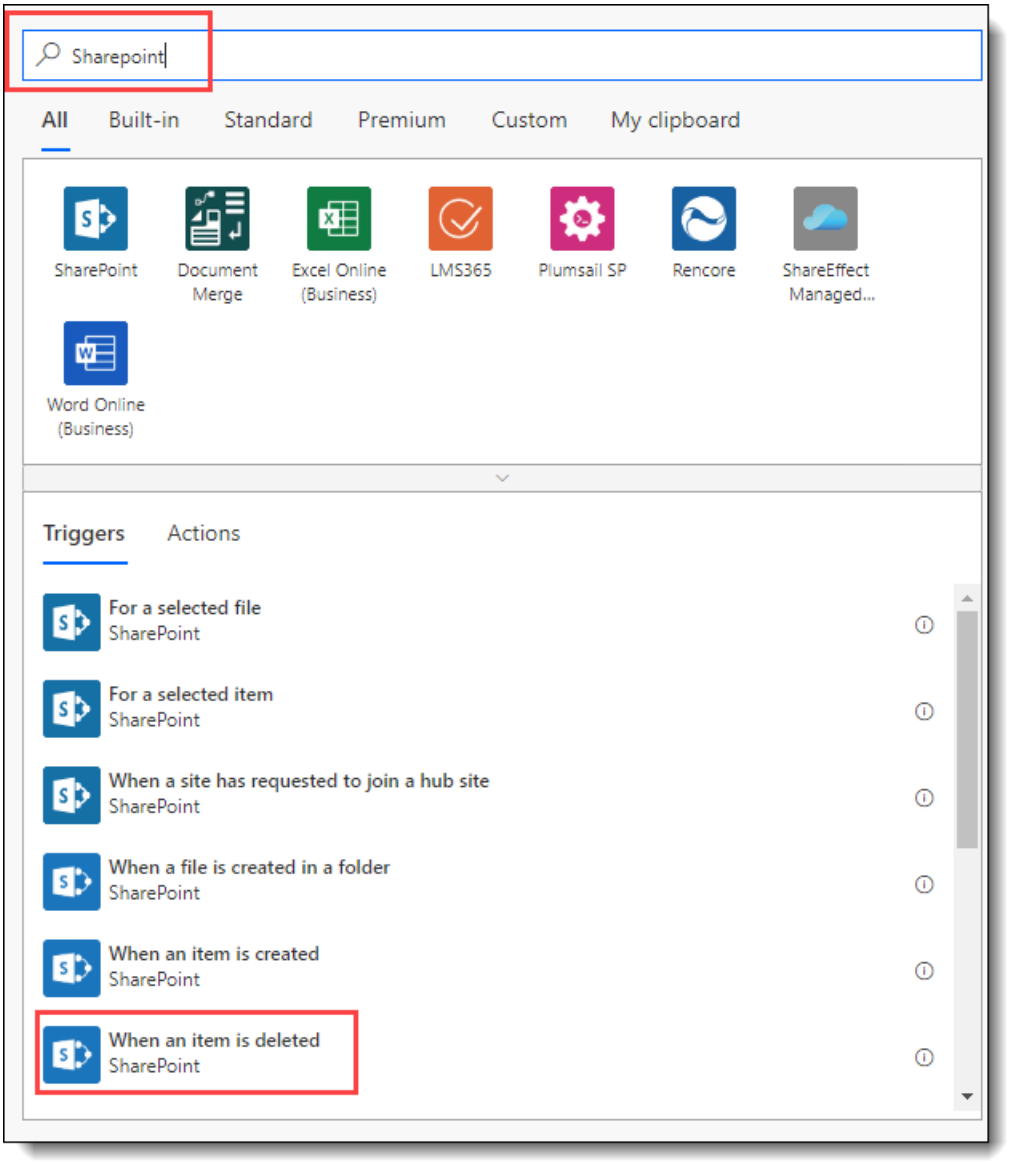
4. **Skip** the next screen.



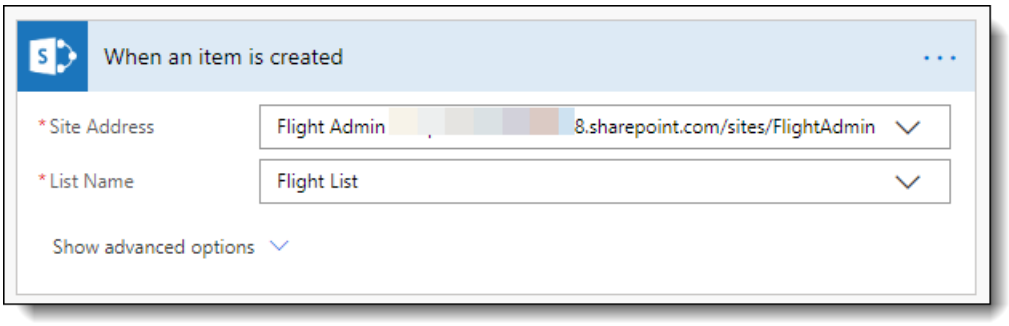
5. To rename the flow, click on **Untitled** (not the back arrow) at the top and enter **Create Team** as the flow name. Then click Save.



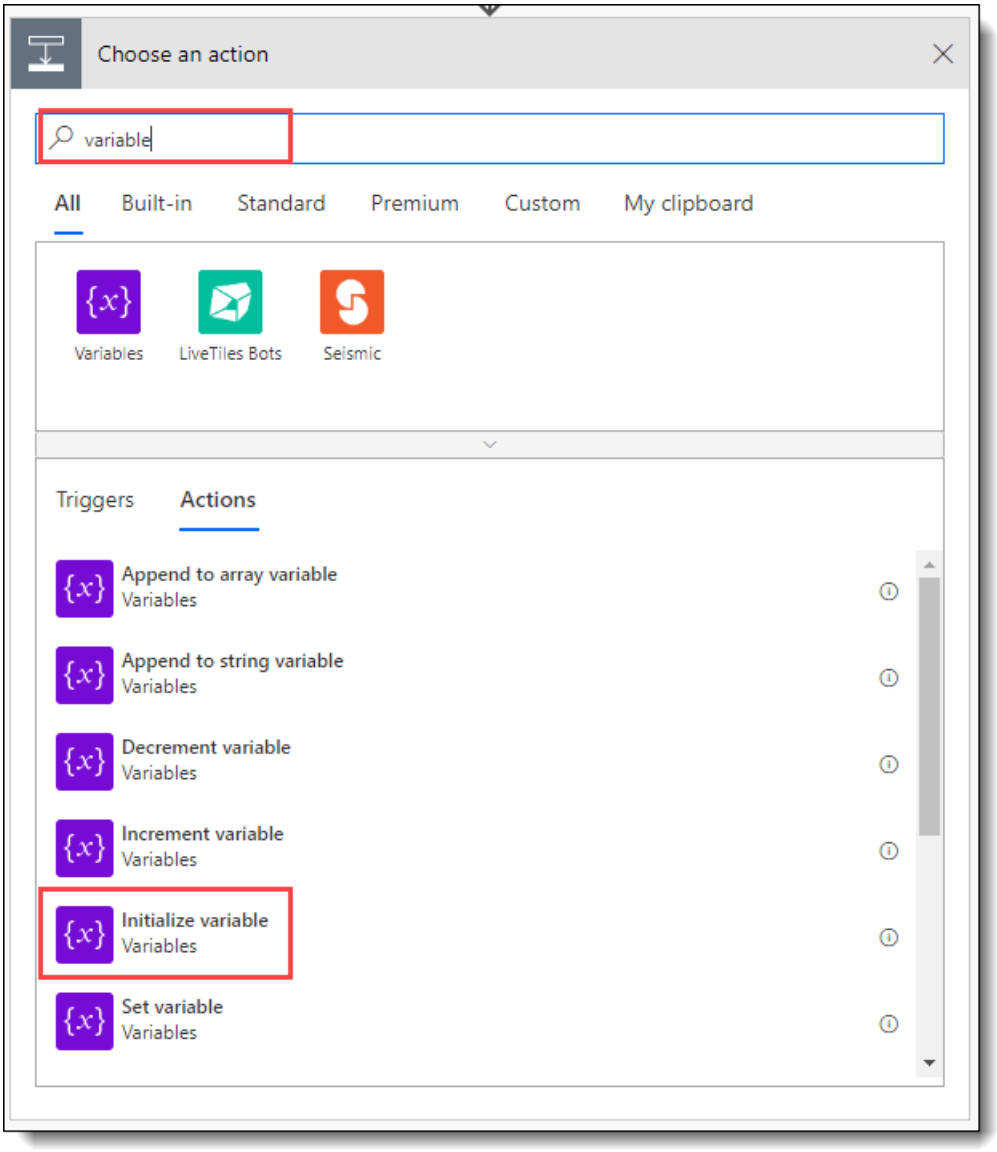
6. On the Search connectors and triggers search bar, enter **SharePoint** in the search box, then select **When an item is created** SharePoint trigger.



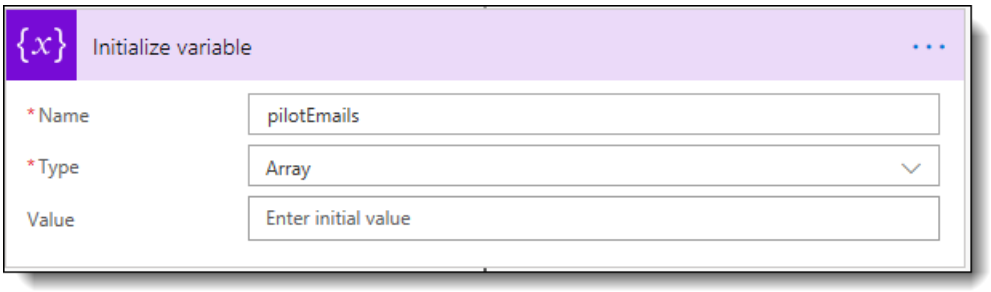
Enter the URL to your Flight Admin site you created earlier for **Site Address**, then select **Flight List** for **List Name**.



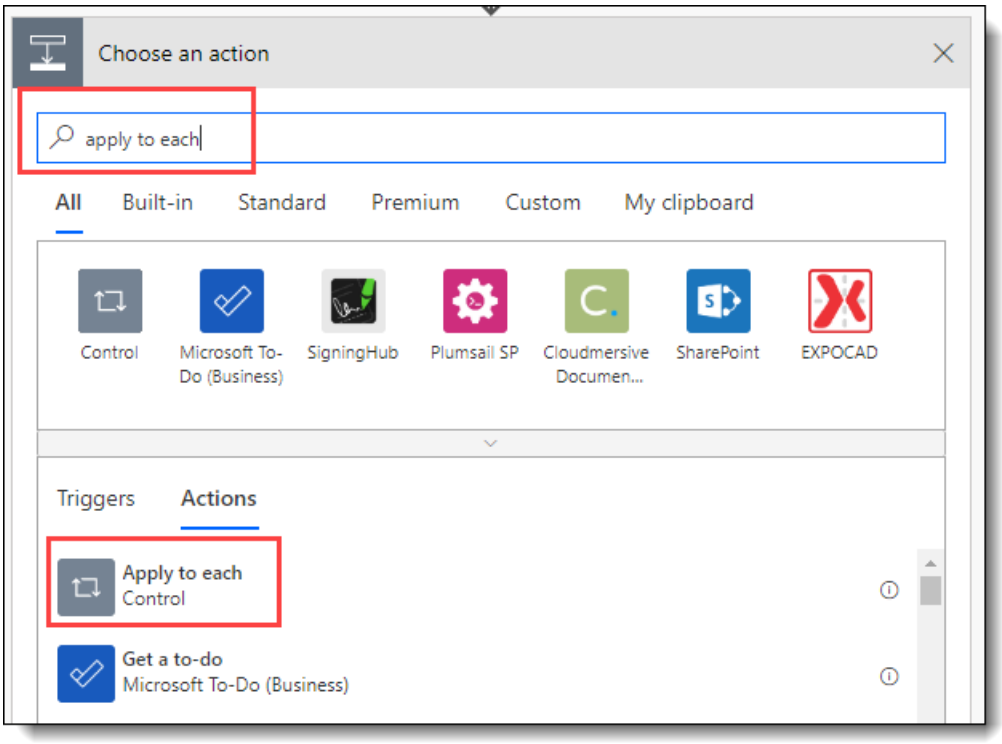
7. Click **New Step**.
8. Enter **variable** in the search box, then select **Initialize variable**.



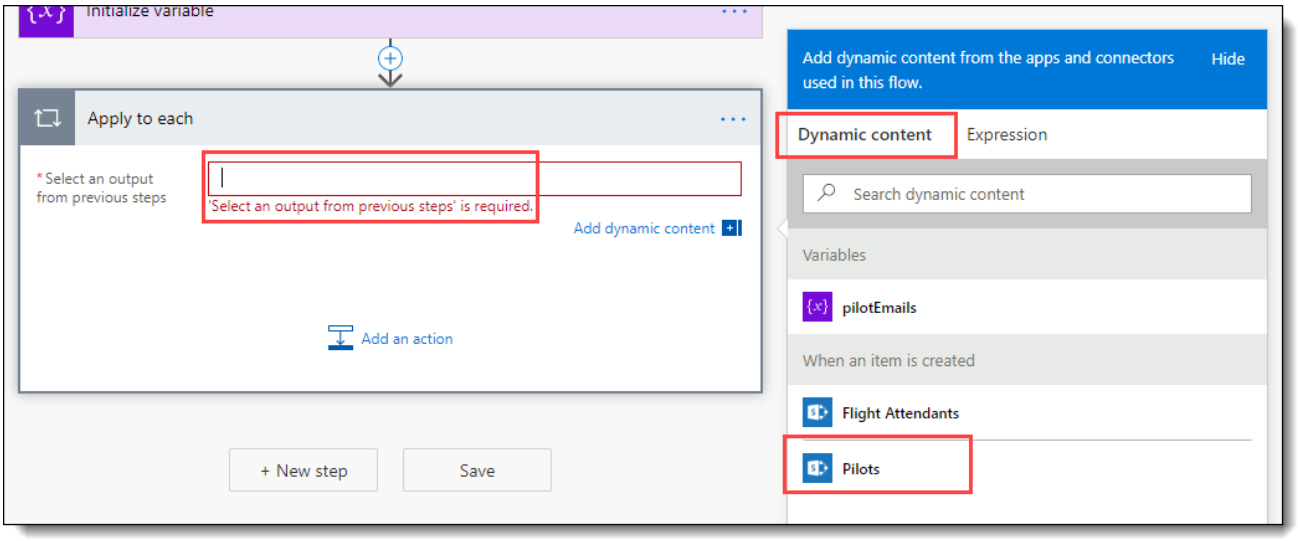
Enter **pilotEmails** for Name, change Type to **Array**, leave Value blank.



9. Choose **New Step**.
10. Enter **apply to each** in the search box, then select **Apply to each** control action.

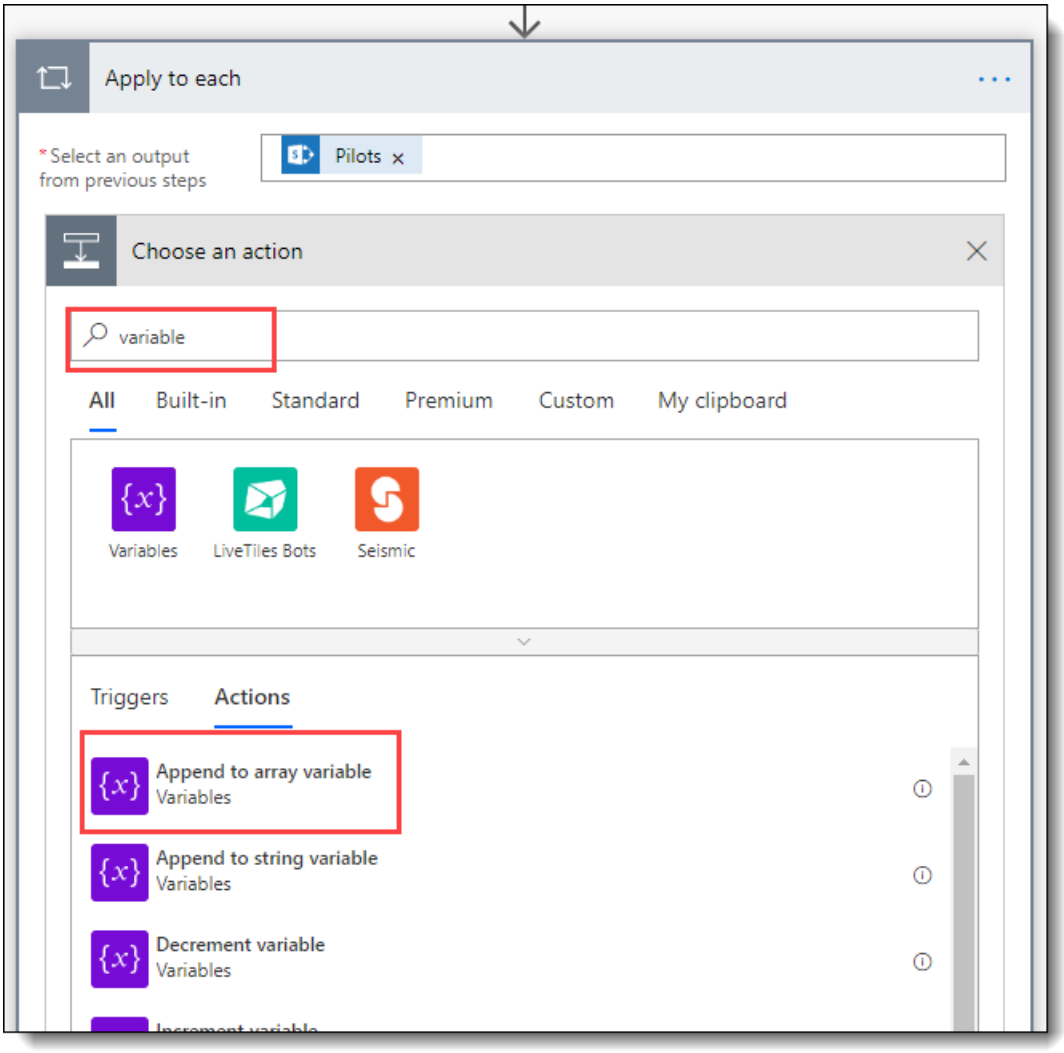


i. For **Select an output from previous steps**, under Dynamic content select **Pilots**.



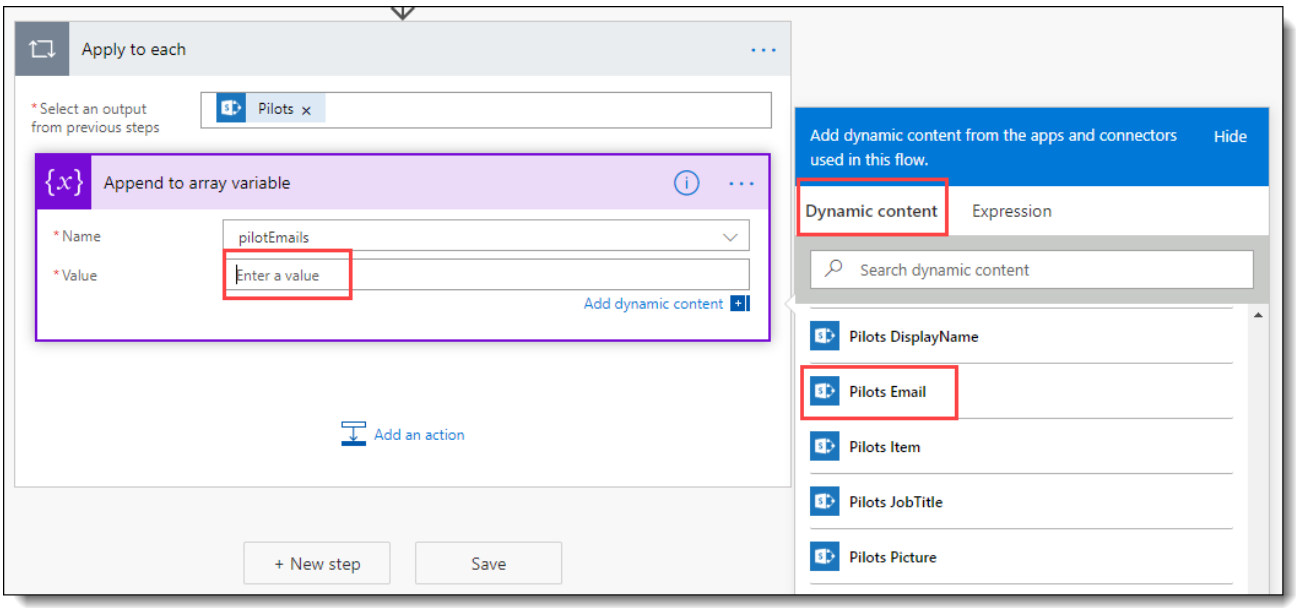
ii. Click **Add an action** within the Apply to each box

iii. Enter **variable** in the search box, then select **Append to array variable**.



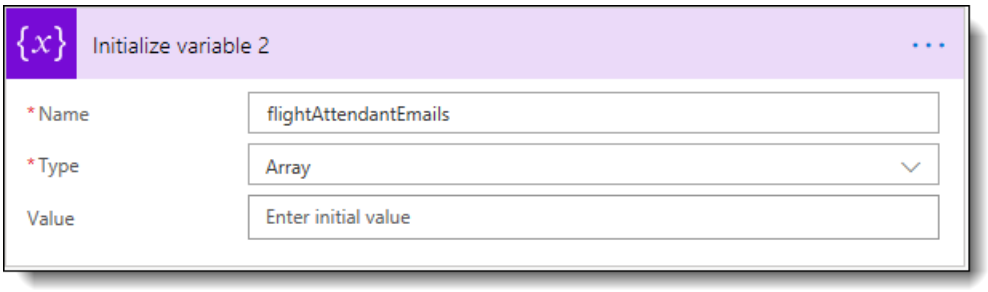
iv. Enter **pilotEmails** for Name.

v. For Value, under Dynamic content select **Pilots Email**.



11. Choose **New Step** (outside of the Apply to each box).

12. Enter **variable** in the search box, then select **Initialize variable**. Enter **flightAttendantEmails** for Name, change Type to **Array**, leave **Value** blank.



13. Choose **New Step**.

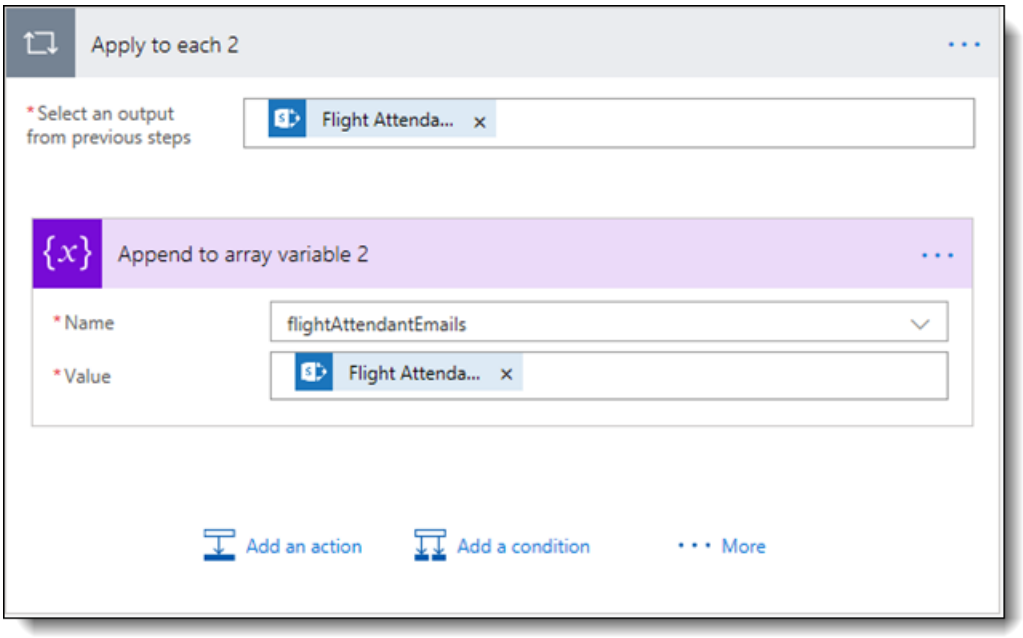
14. Enter **apply** in the search box, then select **Apply to each**.

    i. For **Select an output from previous steps**, under Dynamic content select **Flight Attendants**.

    ii. Click **Add an action**

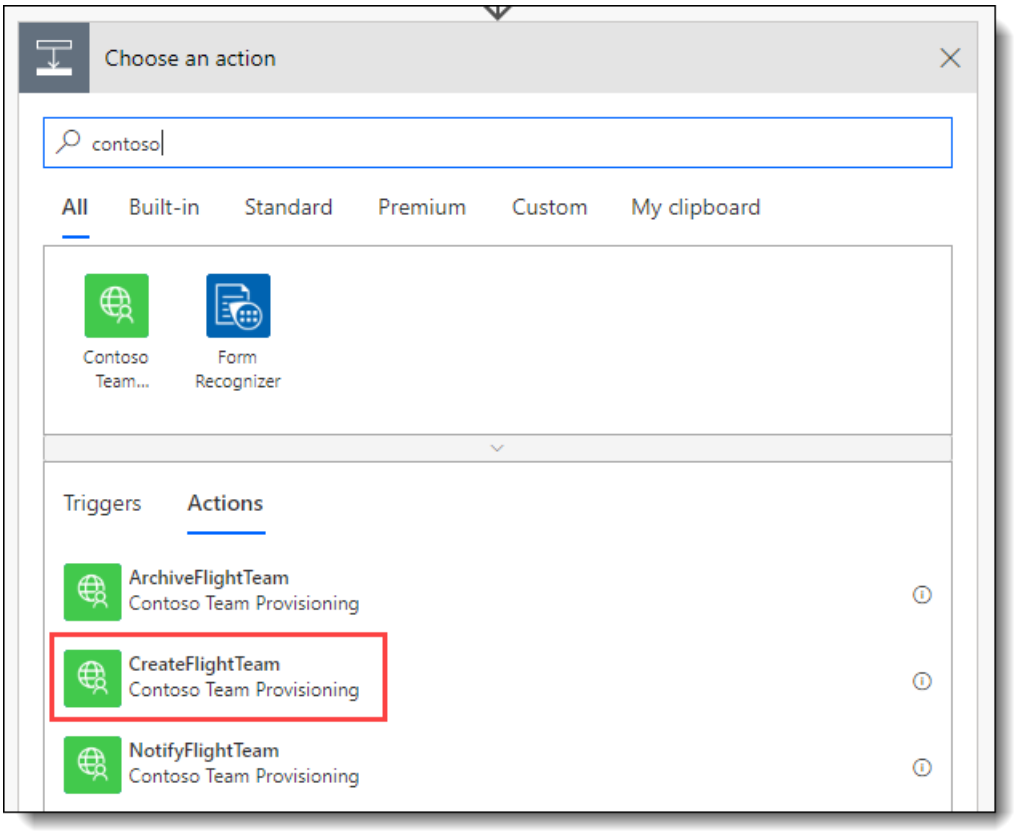
    iii. Enter **variable** in the search box, then select **Append to array variable**.

- Enter **flightAttendantEmails** for Name.
- For Value, under Dynamic content select **Flight Attendants Email**.



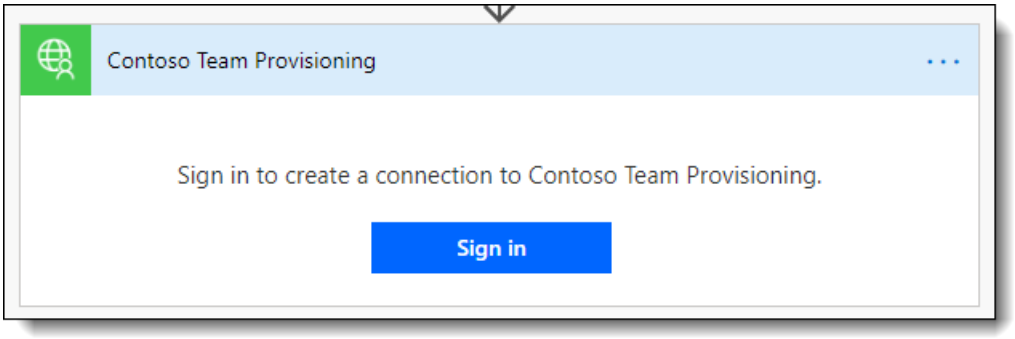
15. Click **New Step**.

16. Enter **Contoso** in the search box, then select **CreateFlightTeam**. This is the custom connector we previously created from the Postman collection.



17. Click **Sign in**.

> **NOTE:**
> Be sure to use an administrator account when you sign in to create the connection. The permissions required by the sample require administrator consent.



18. Once sign in and consent are complete, you should see a form for the action. Fill in the fields as follows:

    i. **sharePointItemId**: Under Dynamic content choose See more, then select **ID**.

    ii. **flightNumber**: Under Dynamic content choose See more, then select **Flight Number**.

    iii. **description**: Under Dynamic content select **Title**.

    iv. **admin**: Under Dynamic content select **Admin Email**.

    v. **pilots**: Choose the Switch to input entire array toggle to the right of the input box, then place your cursor

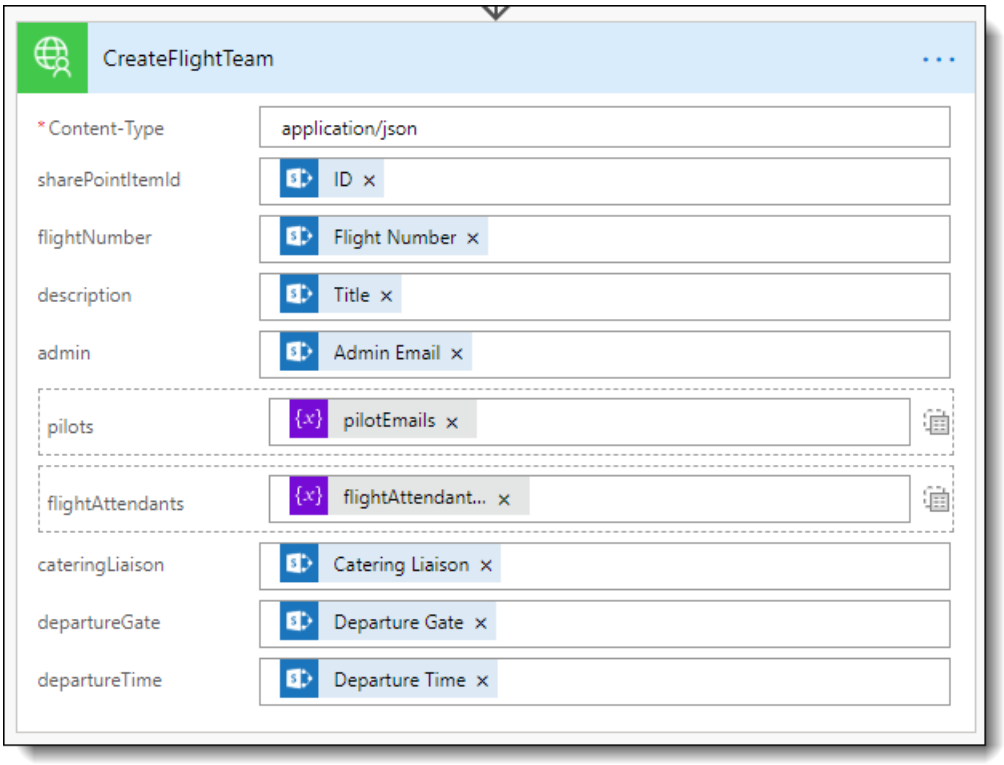
in the input box. Under Dynamic content, select **pilotEmails**.

vi. **flightAttendants**: Choose the Switch to input entire array toggle to the right of the input box, then place your cursor in the input box. Under Dynamic content, select **flightAttendantEmails**.

vii. **cateringLiaison**: Under Dynamic content select **Catering Liaison**.

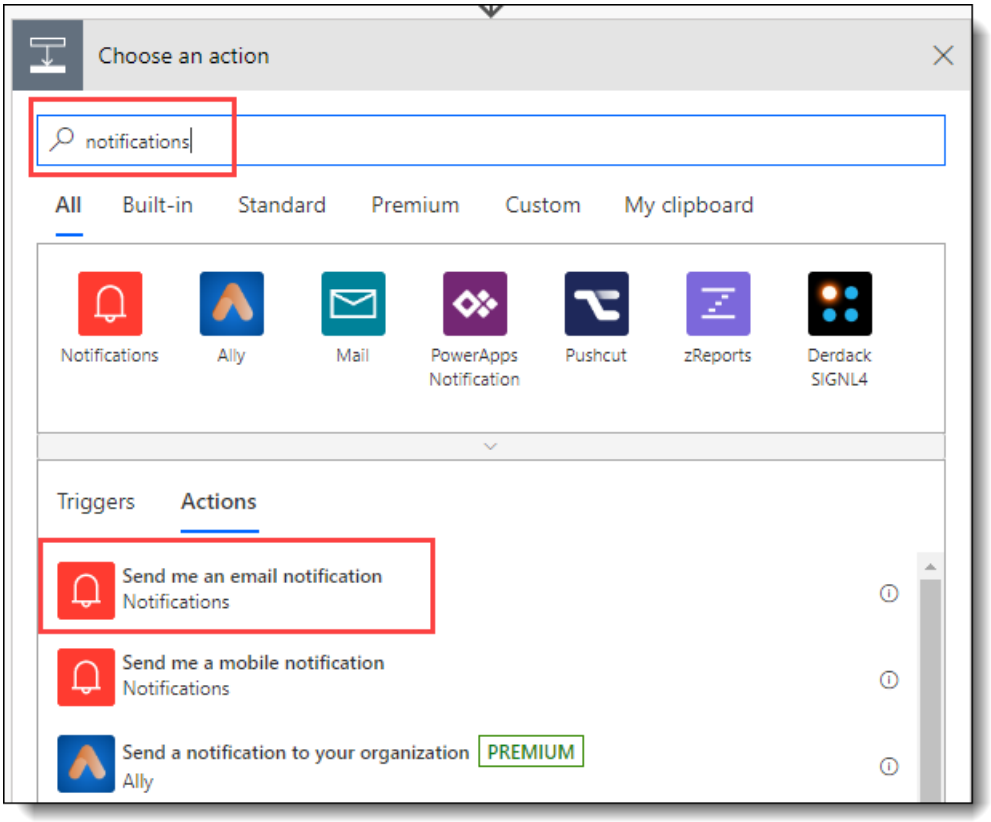
viii. **departureGate**: Under Dynamic content select **Departure Gate**.

ix. **departureTime**: Under Dynamic content select **Departure Time**.



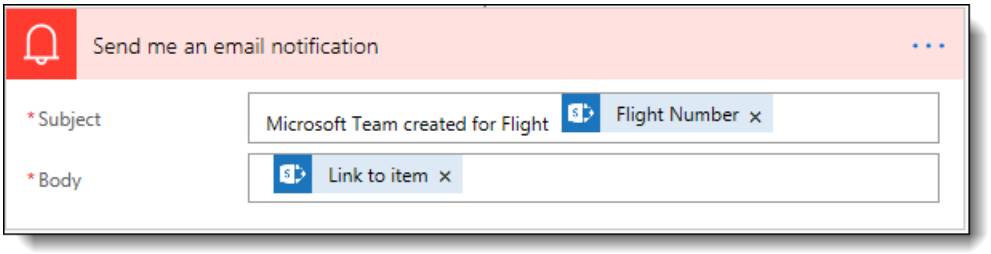
19. Choose **New Step**.

20. Enter **notification** in the search box, then select **Send me an email notification**.

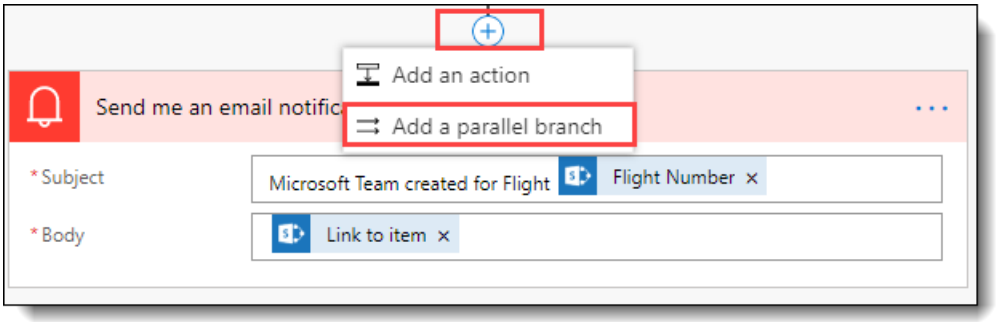


i. For **Subject**, enter **Microsoft Team created for Flight**, then under Dynamic content choose See more, then select **Flight Number**.

ii. For **Body**, under Dynamic content select **Link to item**.



21. Click on the **+** sign above the Send me an email notification, then **Add a parallel branch**.

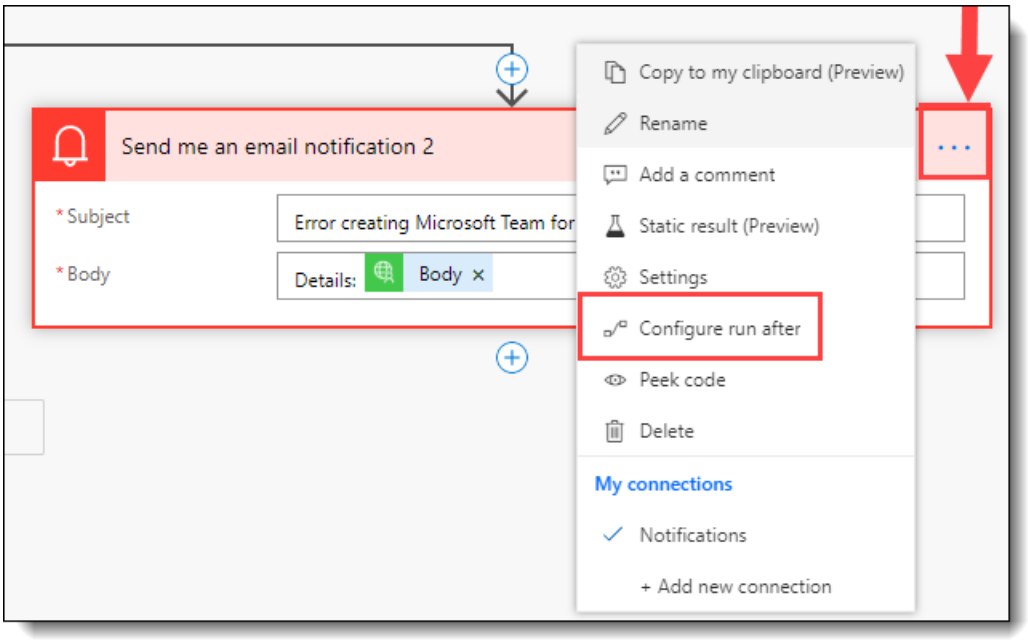


22. Enter **notification** in the search box, then select **Send me an email notification**.

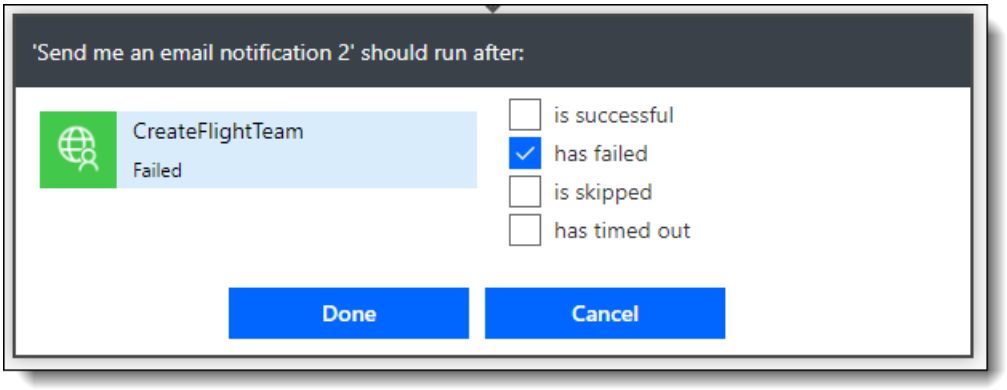
i. For Subject, enter **Error creating Microsoft Team for Flight**, then under Dynamic content choose See more, then select **Flight Number**.

ii. For Body, enter **Details**: then under Dynamic content select **Body**.

iii. Choose the **...** on the action and choose **Configure run after**.

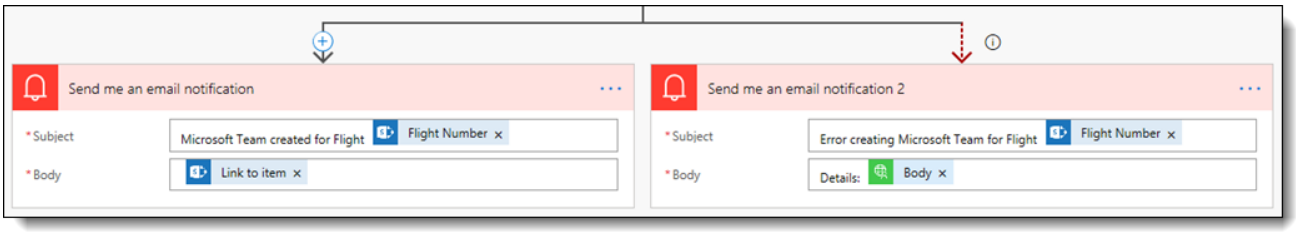


- Uncheck **is successful**
- Select **has failed**



- Click **Done**.

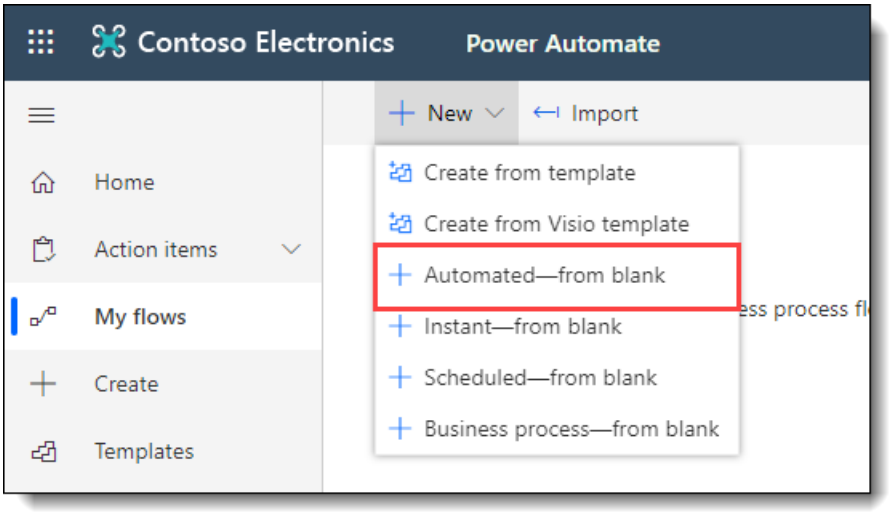
  The email notification actions should look like the following when you're done.



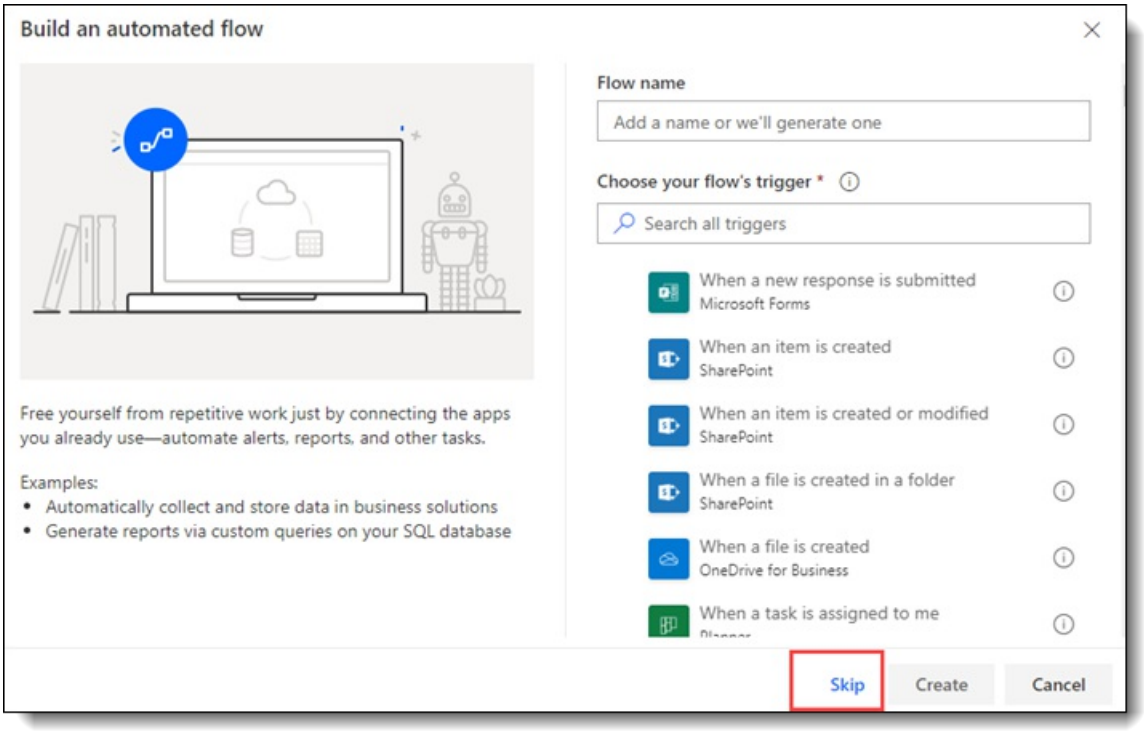
23. Select **Save** to save the flow.

Create the "Archive Team" flow

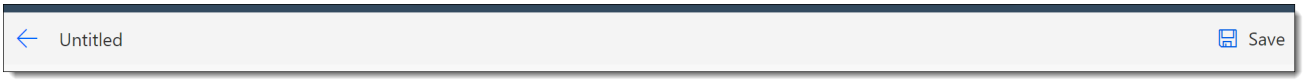
1. Browse to **Power Automate** and sign in with you tenant's admin account.
2. On the left panel, click **My flows**.
3. Click **New** at the top bar, then click **Automated - from blank**.



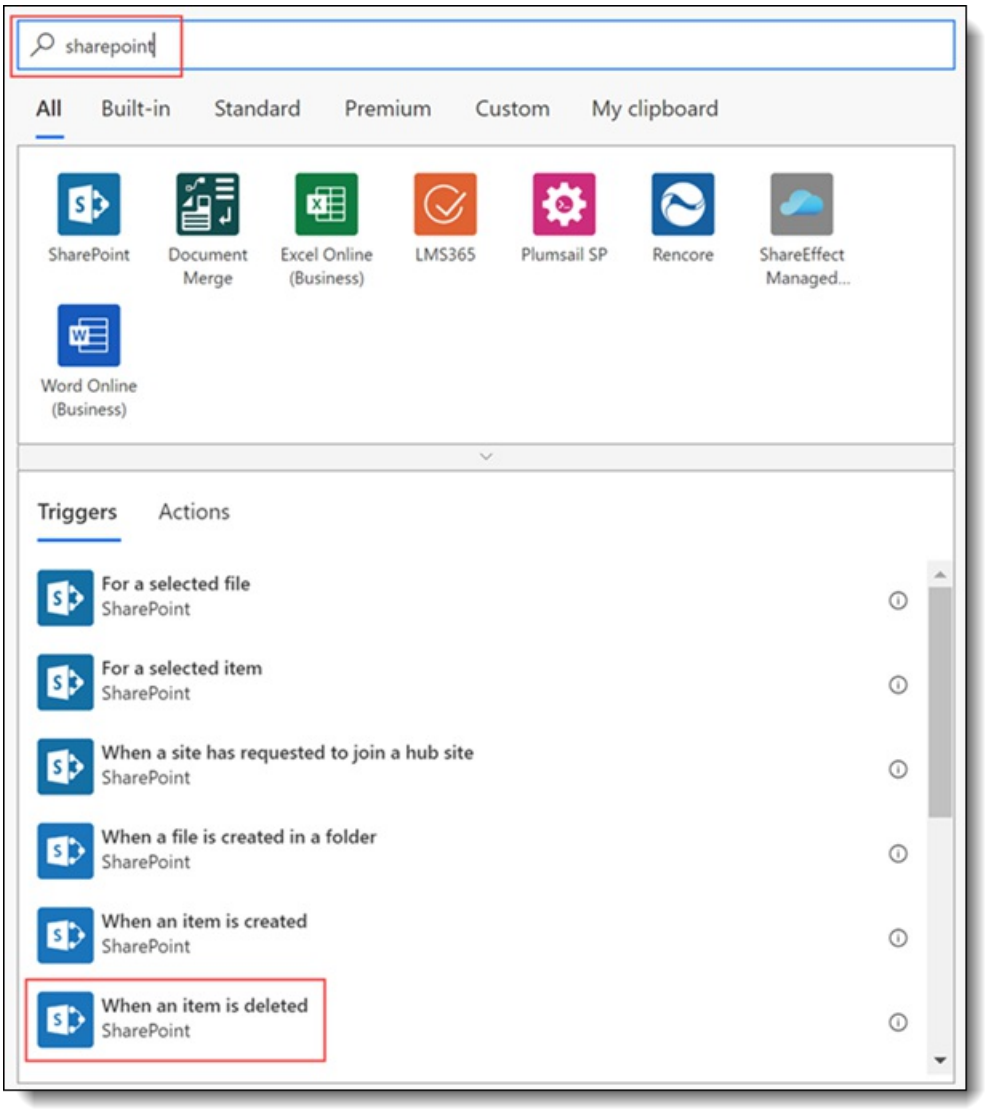
4. **Skip** the next screen.



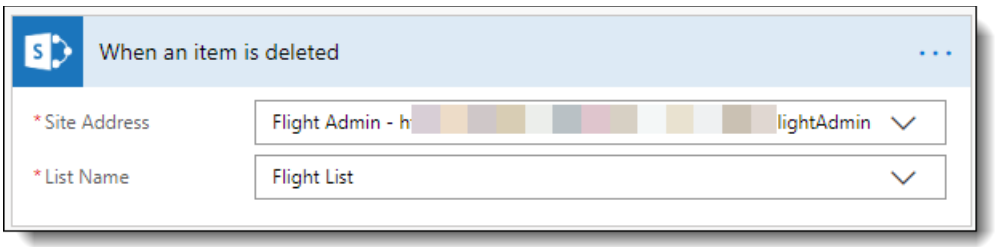
5. To rename the flow, click on **Untitled** (not the back arrow) at the top and enter **Archive Team** as the flow name. Then click **Save**.



6. On the Search connectors and triggers search bar, enter SharePoint in the search box, then select **SharePoint - When an item is deleted**.

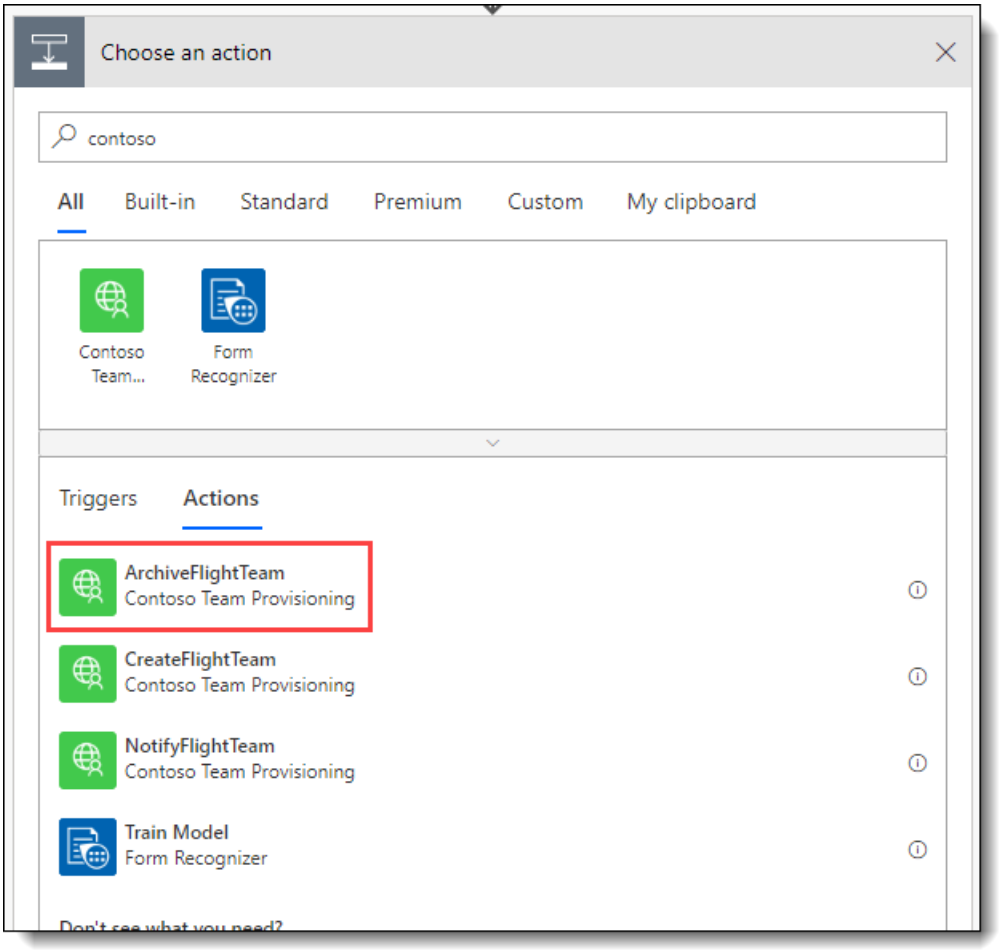


Enter the URL to your **Flight Admin SharePoint site** you created earlier for Site Address, then select **Flight List** for **List Name**.

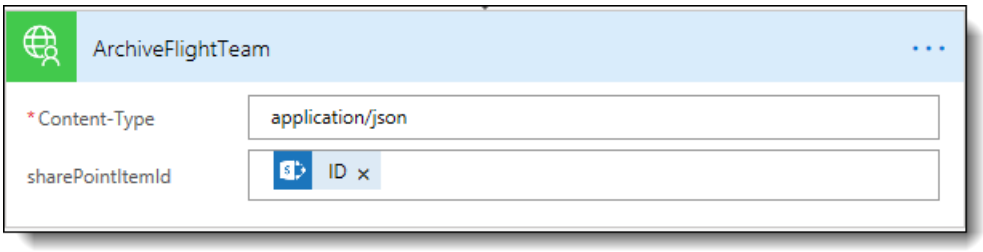


7. Click **New Step**, then **Add an action**.

8. Enter **Contoso** in the search box, then select **ArchiveFlightTeam**.



9. Fill in the fields as follows using the Flow UI. **sharePointItemId**: Under Dynamic content choose See more, then select **ID**.

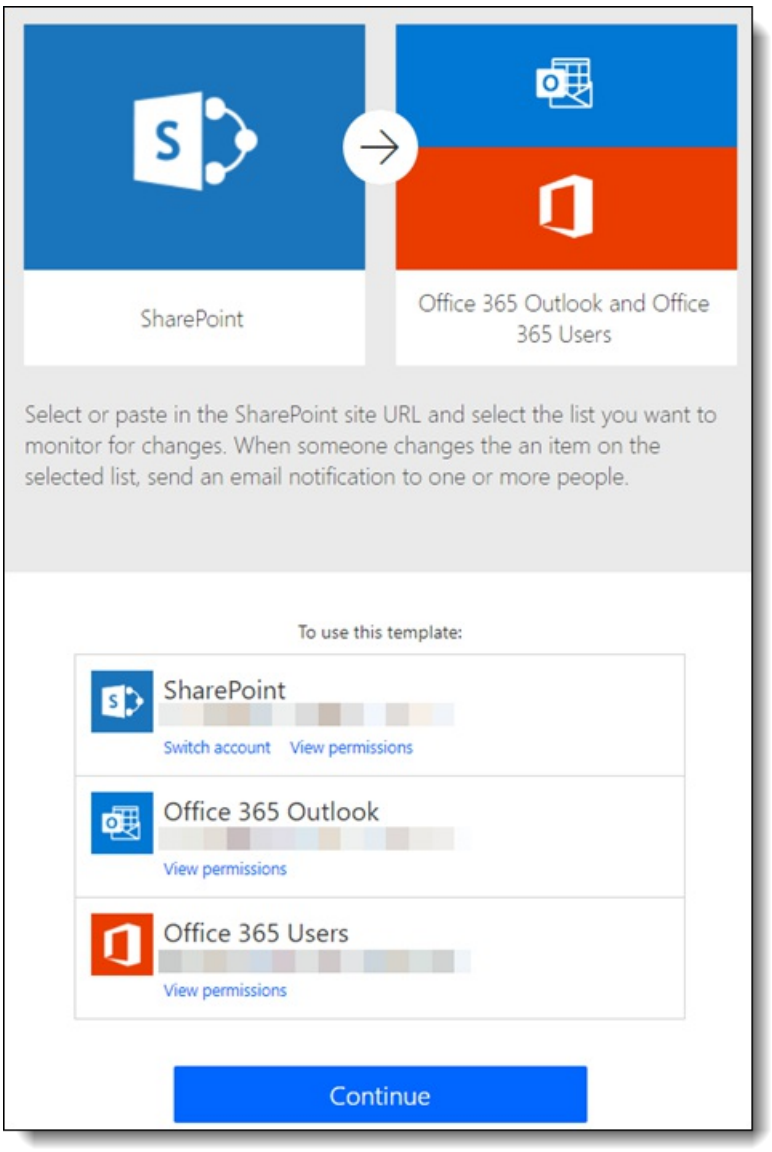


10. Select **Save** to save the flow.

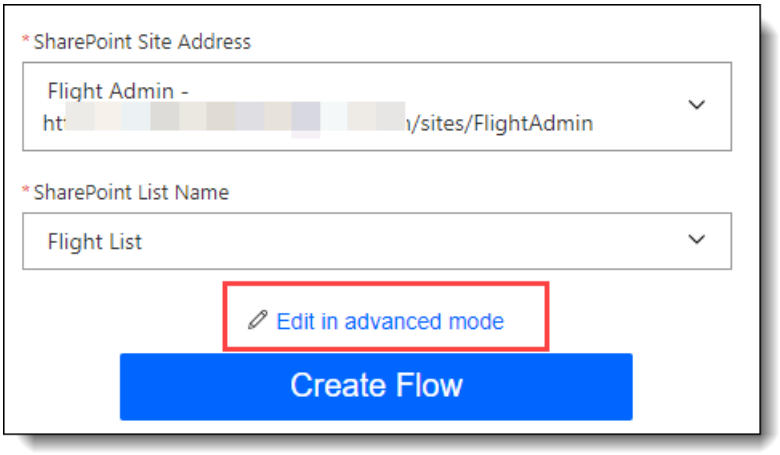
Create the "Notify Team" flow

> For this flow, we will start with an existing template and modify it to our use. Unfortunately, the Power Automate team removed the "When an existing item is modified" trigger for SharePoint from the UI when creating a blank flow. By using this template, we can still get access to it.

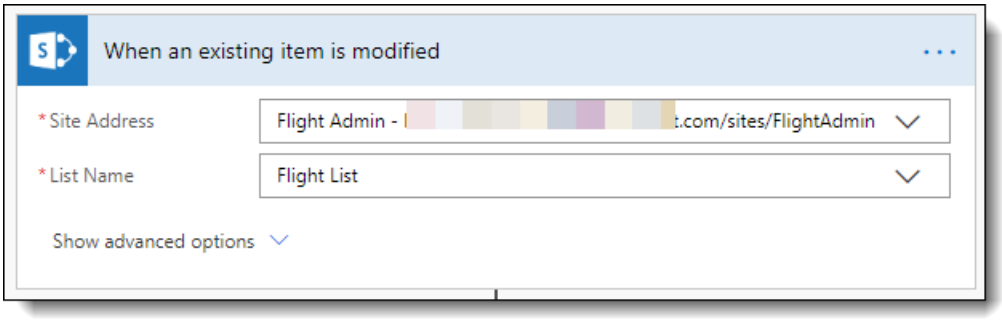
1. Browse to the following **Power Automate template**. Click **Continue**.



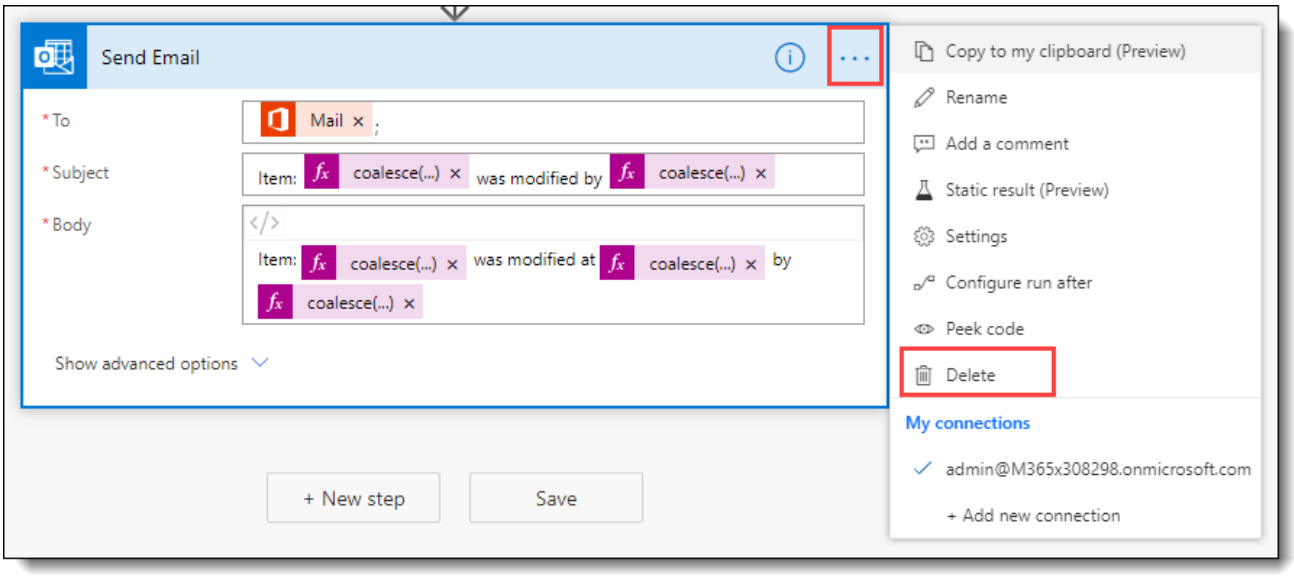
2. Specify the Site Url (SharePoint Flight admin site previously created) and the list name **Flight Name**. Click on **Edit in advance mode**.



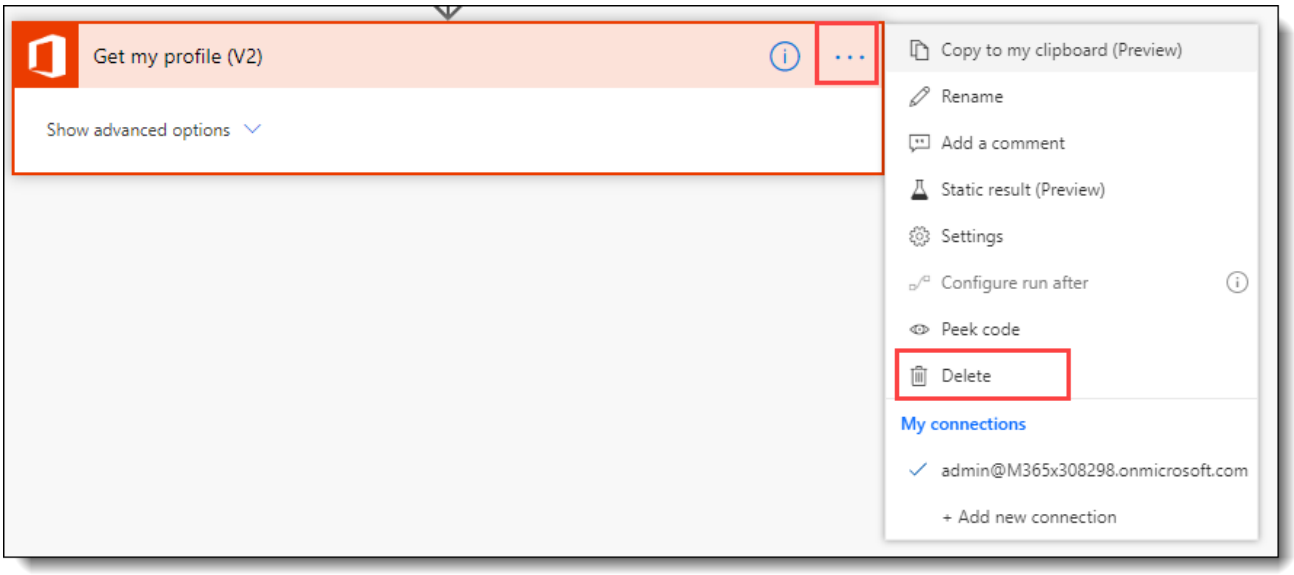
3. In the When an existing item is modified box, enter the URL to your **Flight Admin site** you created earlier for Site Address, then select **Flight List** for List Name.



4. Click the **...** on the **Get My Profile (V2)** box and choose **Delete**. Choose **OK** when prompted.

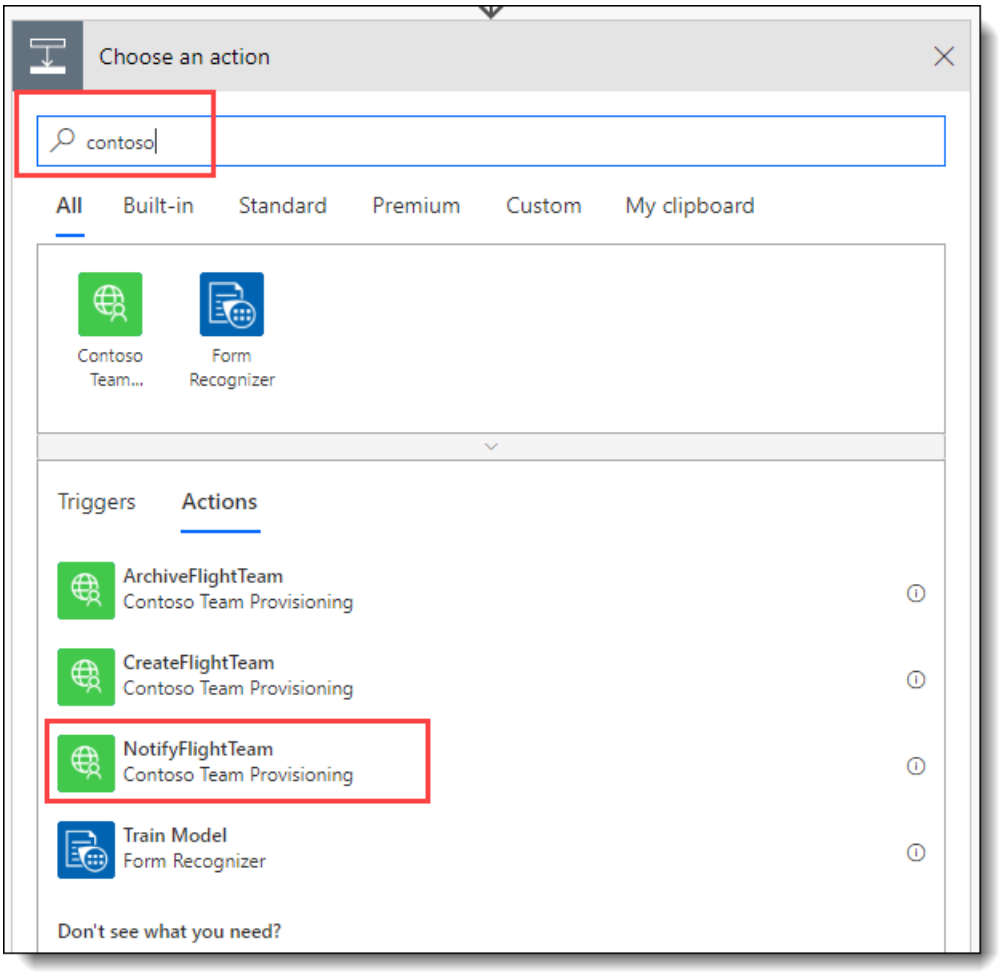


5. Choose the **...** on the **Send Email** box and choose **Delete**. Choose **OK** when prompted.



6. Click **New Step**, then **Add** an action.

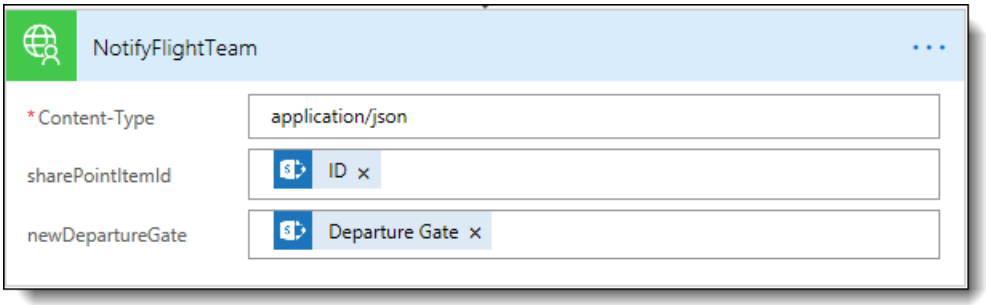
7. Enter **Contoso** in the search box, then select **NotifyFlightTeam**.



8. Fill in the fields as follows using the Flow UI.

i. **sharePointItemId**: Under Dynamic content choose See more, then select **ID**.

ii. **newDepartureGate**: Under Dynamic content select **Departure Gate**.



9. Select **Save** to save the flow.

Exercise 8: Testing the Solution

☑ Introduction

In this exercise you will be able to test the solution and see how all the elements work together.

☑ Objectives

After completing this exercise, you will be able to see the deployed solution in action.

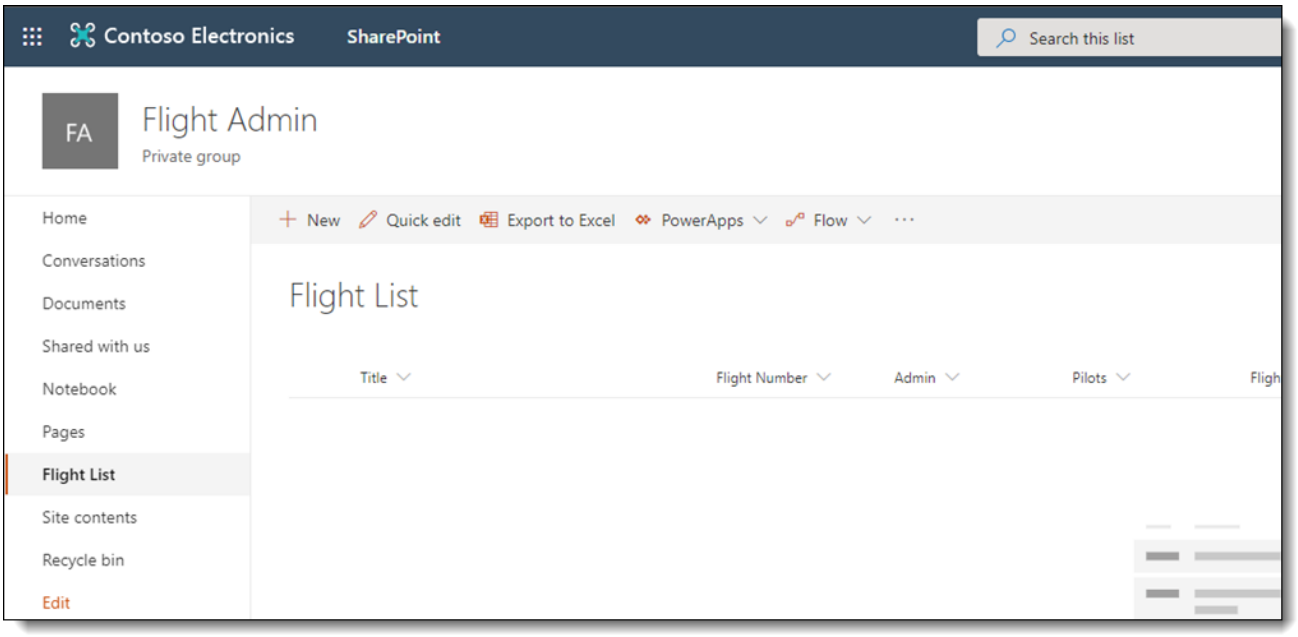
☑ Prerequisites

- Completion of Exercises 1 through 7.
- Demo user accounts.

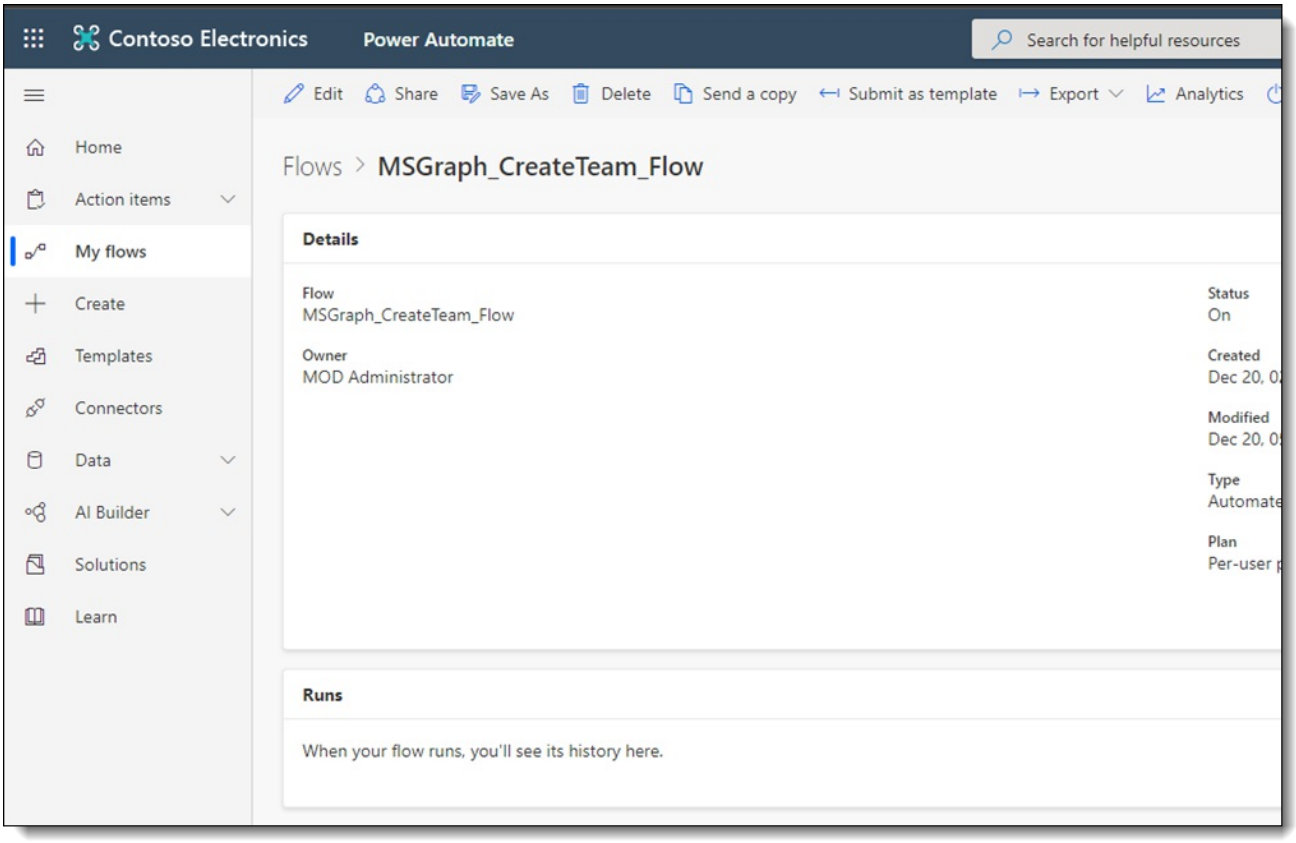
☑ Tasks

Now that all the pieces have been deployed and connected it's time to test the solution. To better see what's happening on the different services, let's open tabs within your browser:

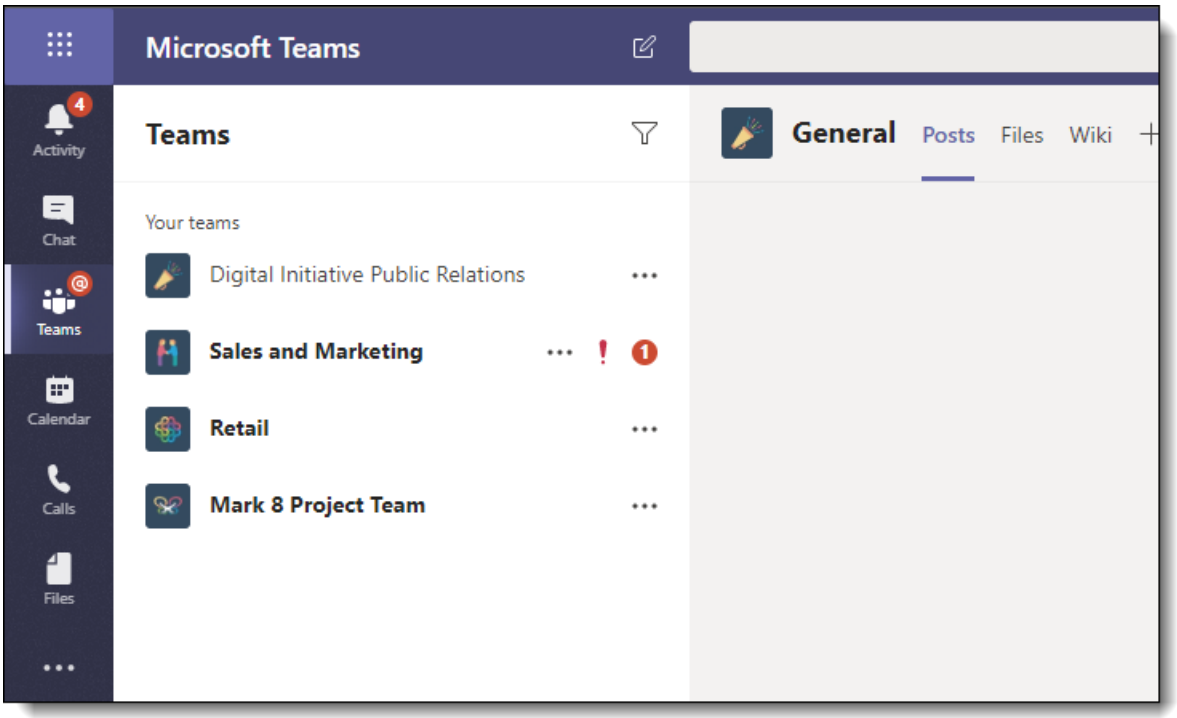
- **Tab #1:** Browse to the SharePoint Flight admin site that you created. Within that site, browse to the **Flight list**. In the next section, you will create an item on this list. This will cause the **Create Team** flow within Power Automate to trigger and start the provisioning of a Team for a specific flight.



- **Tab #2:** Browse to Power Automate and sign in as you tenant's admin. On the left panel, click on **My flows**. Depending on the operation (create/provision) you are testing you will click on a specific flow and be monitoring Runs window at the bottom of the screen.



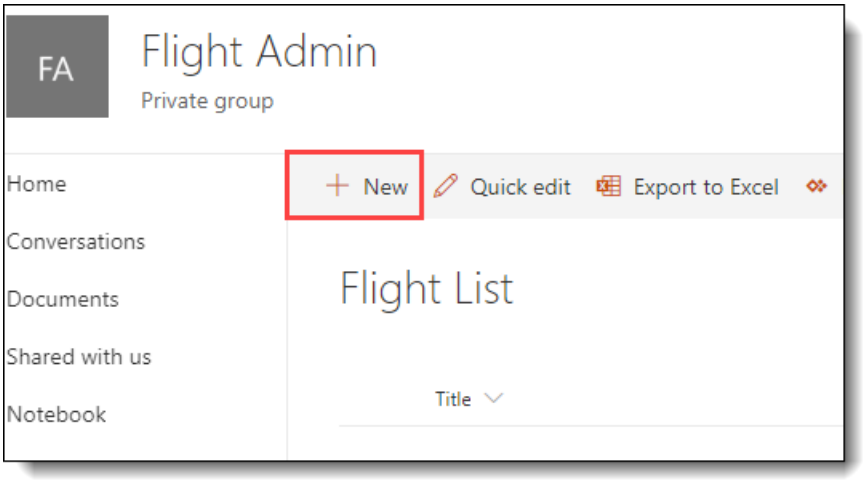
- **Tab #3**: Browse to **Microsoft Teams** and sign in to the web using you tenant's admin account. Click on **Teams**, to view all your existing Teams. Here we will confirm that new teams get provisioned and deleted



⚙ Provisioning a Team

In this section you will test the creation of a team for a specific flight. Follow the next steps:

1. On your SharePoint list (Tab #1) click on **New** at the list toolbar.

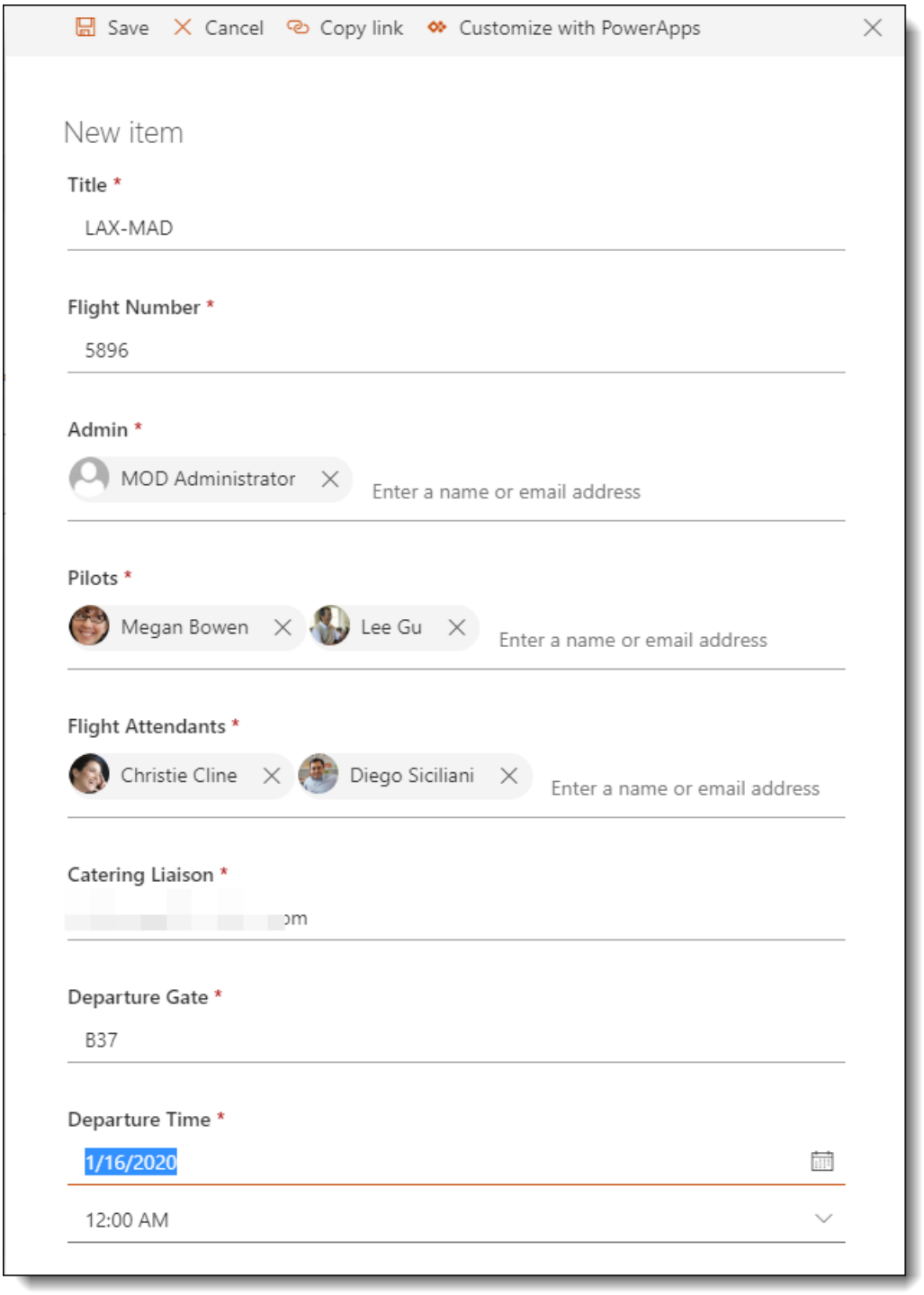


2. Fill in the data, for example: If you are using a demo tenant, you can use the following demo accounts to be placed as **Pilots** and **Flight** Attendants:

- Lee Gu
- Megan Bowen
- Christine Cline
- Diego Siciliani

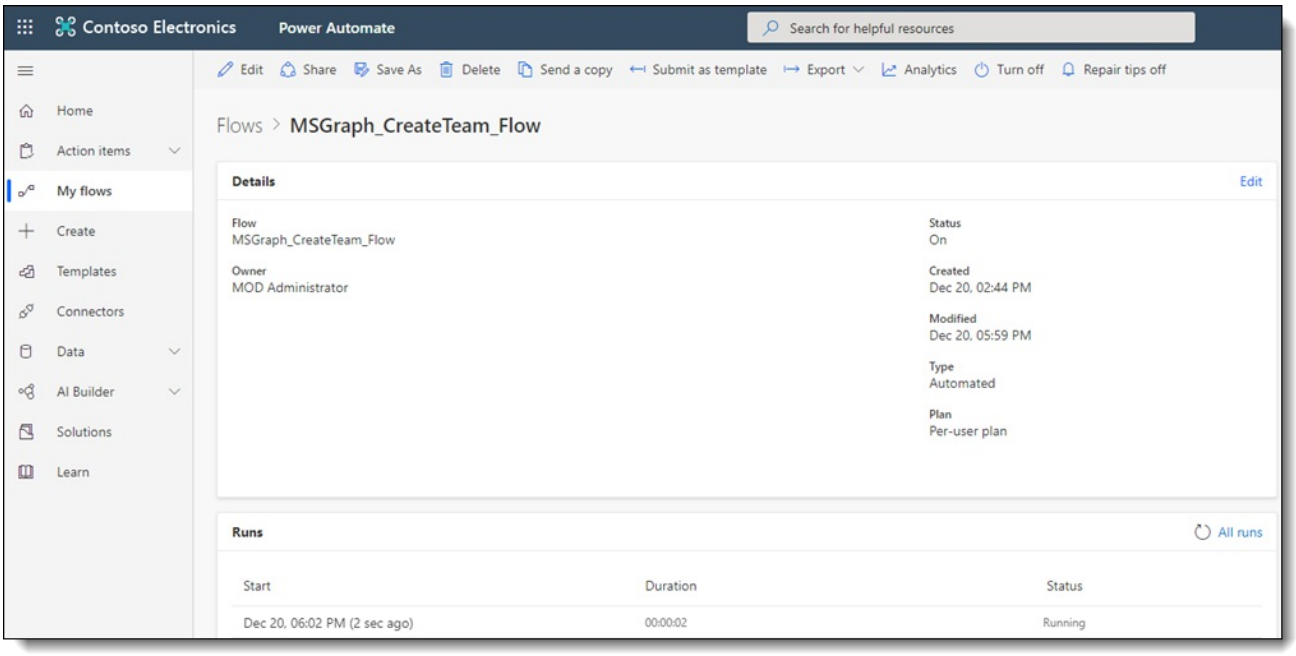
If you are using your own tenant or just provisioned a new M365 trial tenant, you would need to create and specify your own user accounts.

> **Note:** For the **Catering Liaison** field, insert a personal email account you can access. The solution will add your account as a guest member of the provisioned team. This is great for external access collaboration.

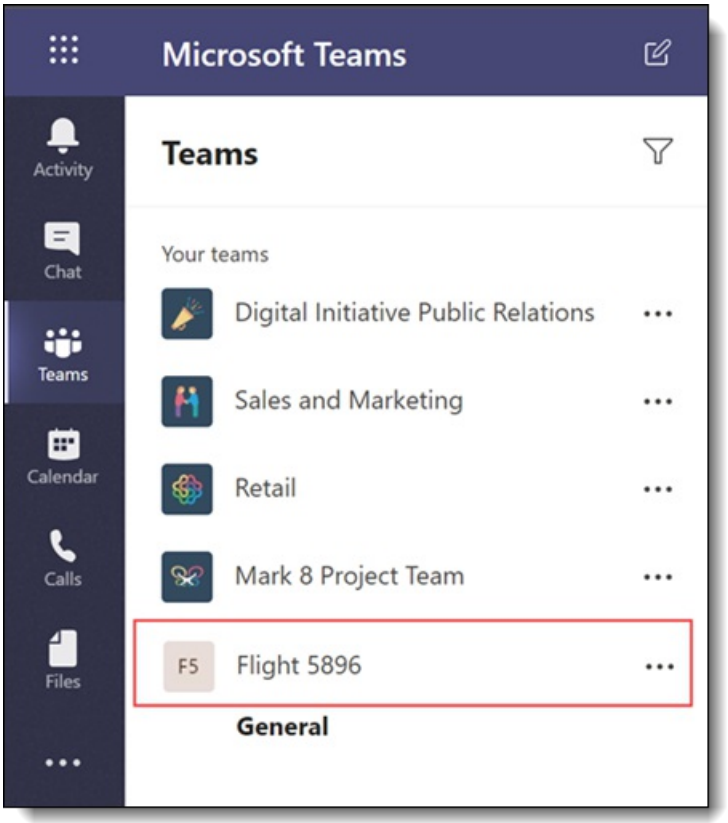


Click **Save** at the top when you are done.

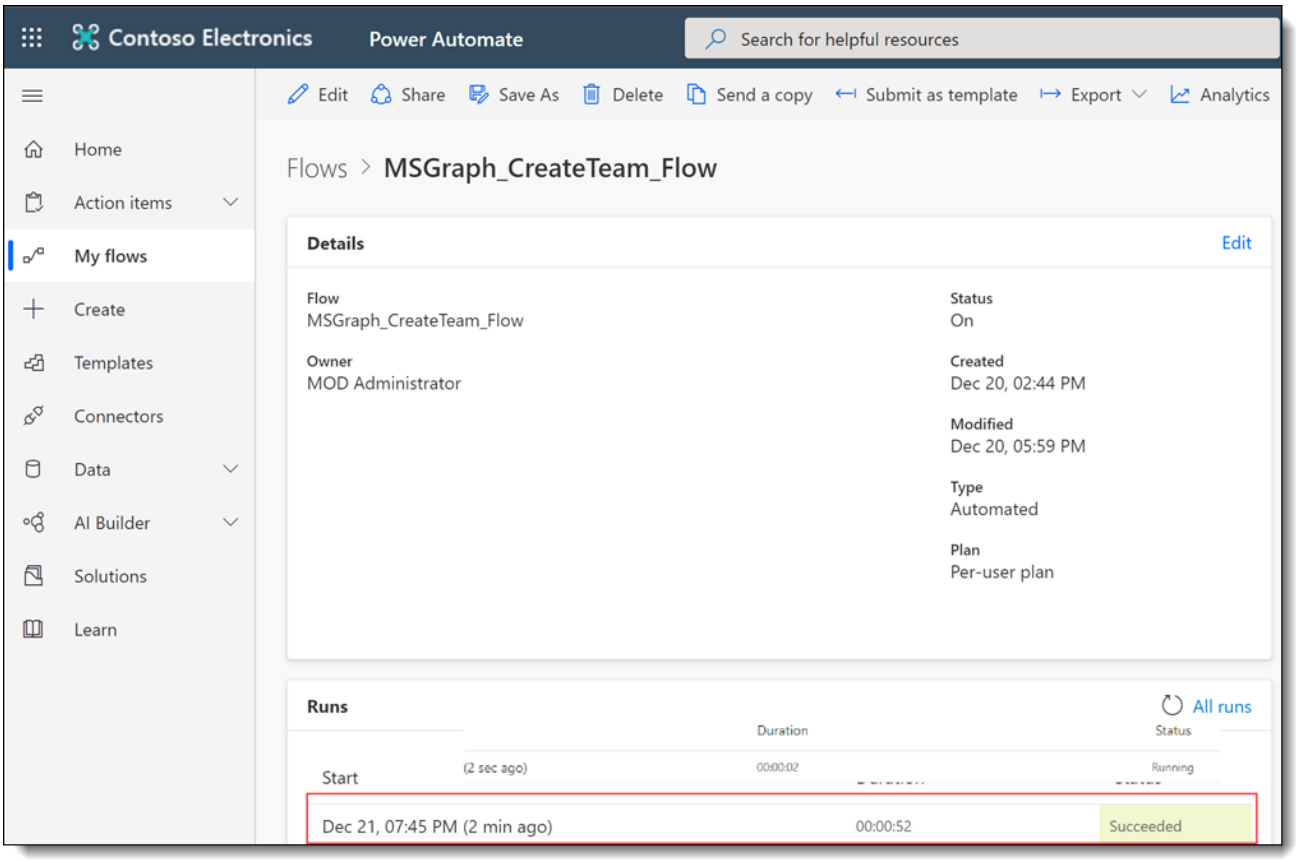
3. Go to **Power Automate** (tab #2) **> My Flows** and click on the **Create Team** flow. You should see a new instance of the flow with the **Running** status in the **Runs** section at the bottom of the screen.



4. Go to **MS Teams** (tab #3) while the flow is running, you should be able to see how the new team is getting provisioned:

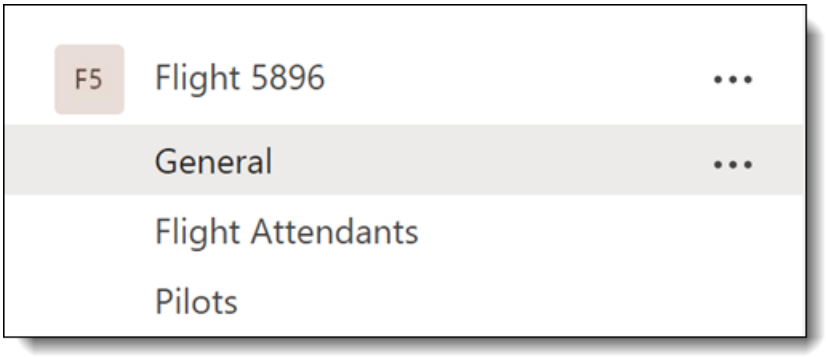


5. Wait for some minutes and verify that the flow to provision teams finished successfully

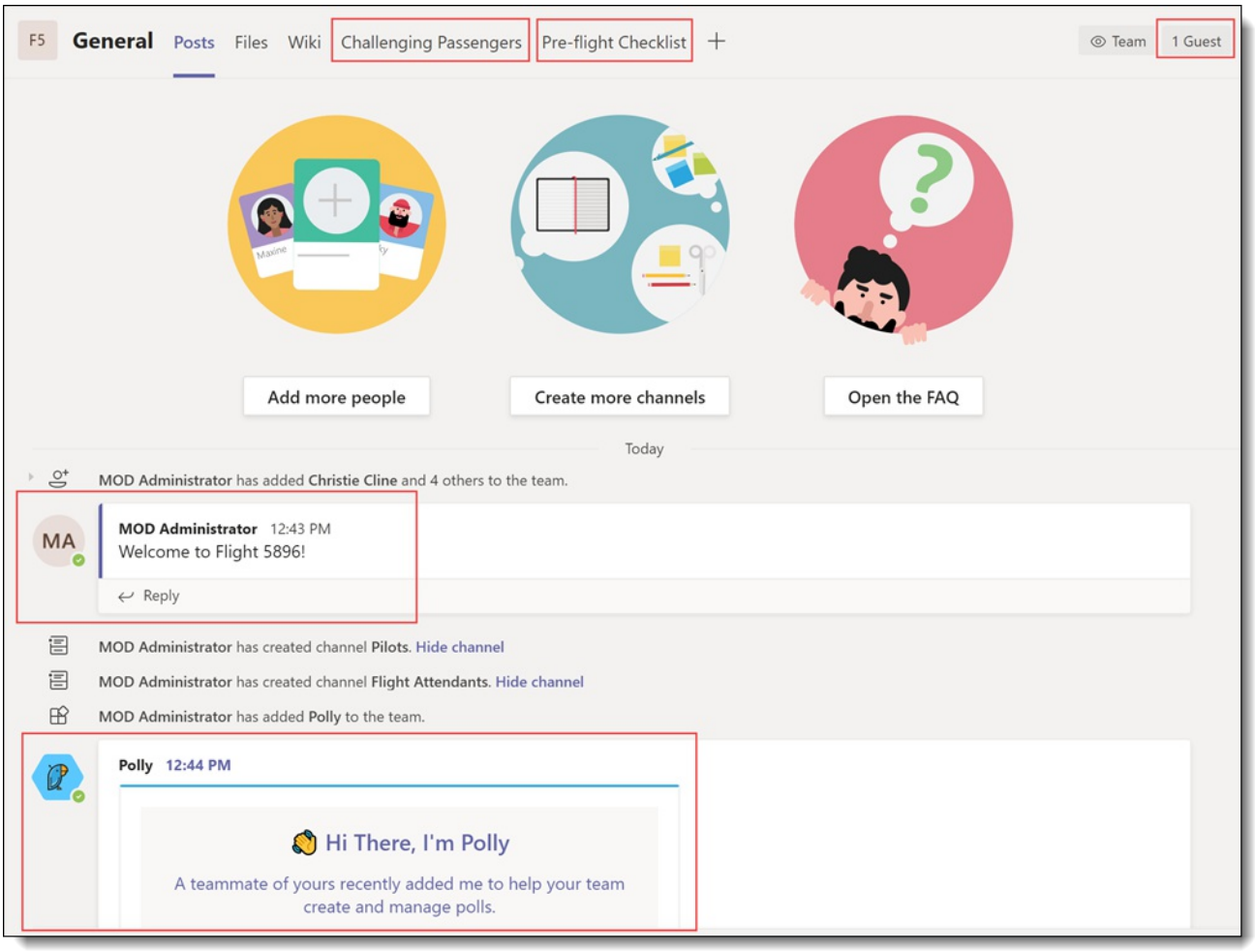


6. You should see a MS Team with the following elements:

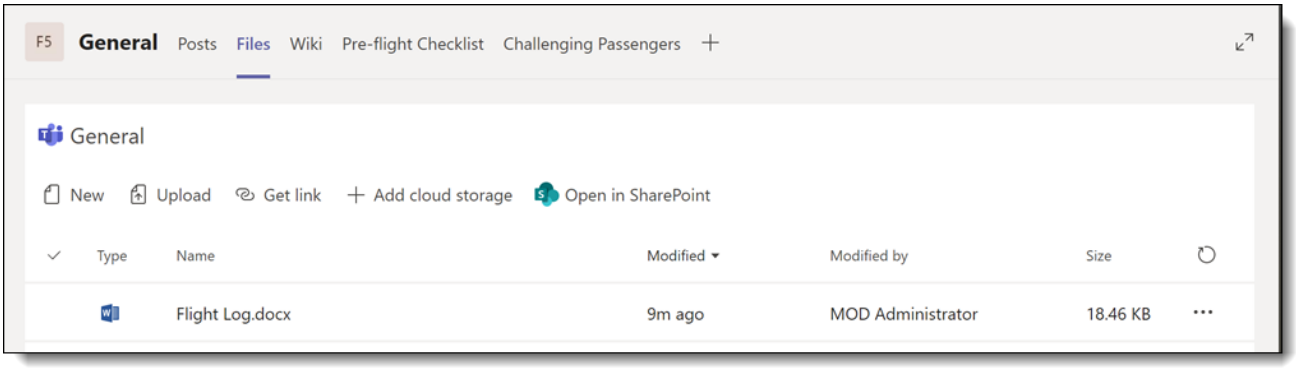
- 3 channels: **General**, **Flight Attendants** and **Pilots**:



- The **General Channel** (the remaining ones are out of the box with no customization) has the following elements:
    - Several **Tabs** on top
    - Some initial messages regarding channel creation and member addition
    - An installed App (Polly)
    - One guest user

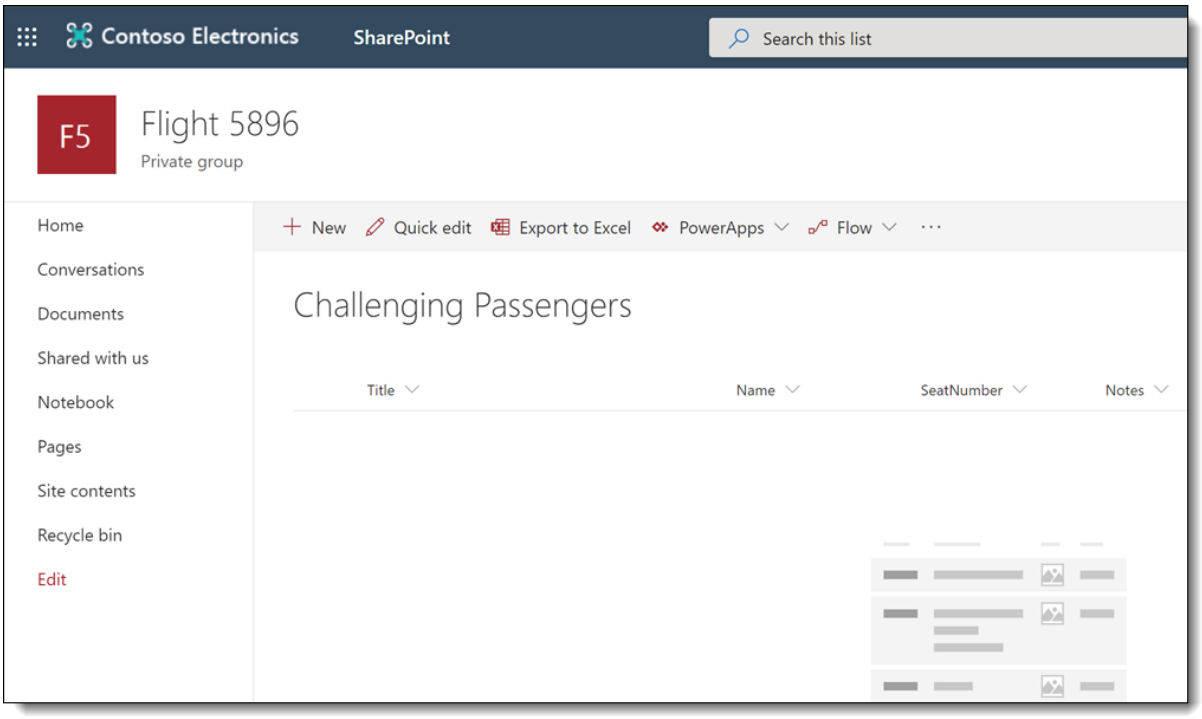


- Within the **Files** Tab, you will find a new document (**Flight Log.docx**) that automatically got created:

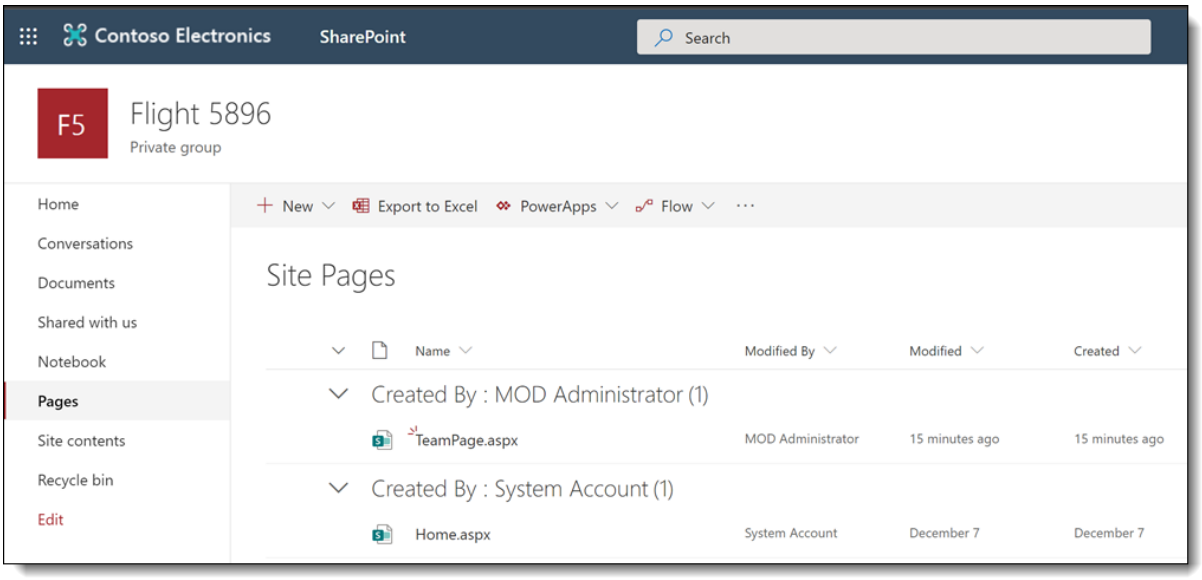


Click **Open in SharePoint**, you will also see that several elements got provisioned within the SharePoint site:

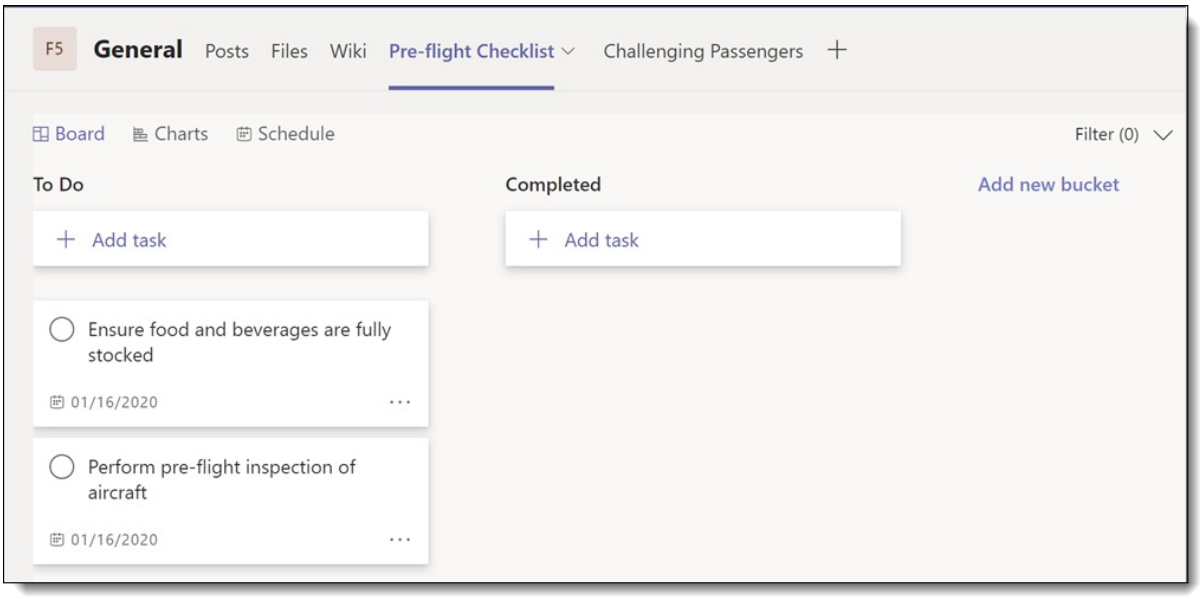
- **Challenging Passengers** List with some custom columns (Name, SeatNumber):



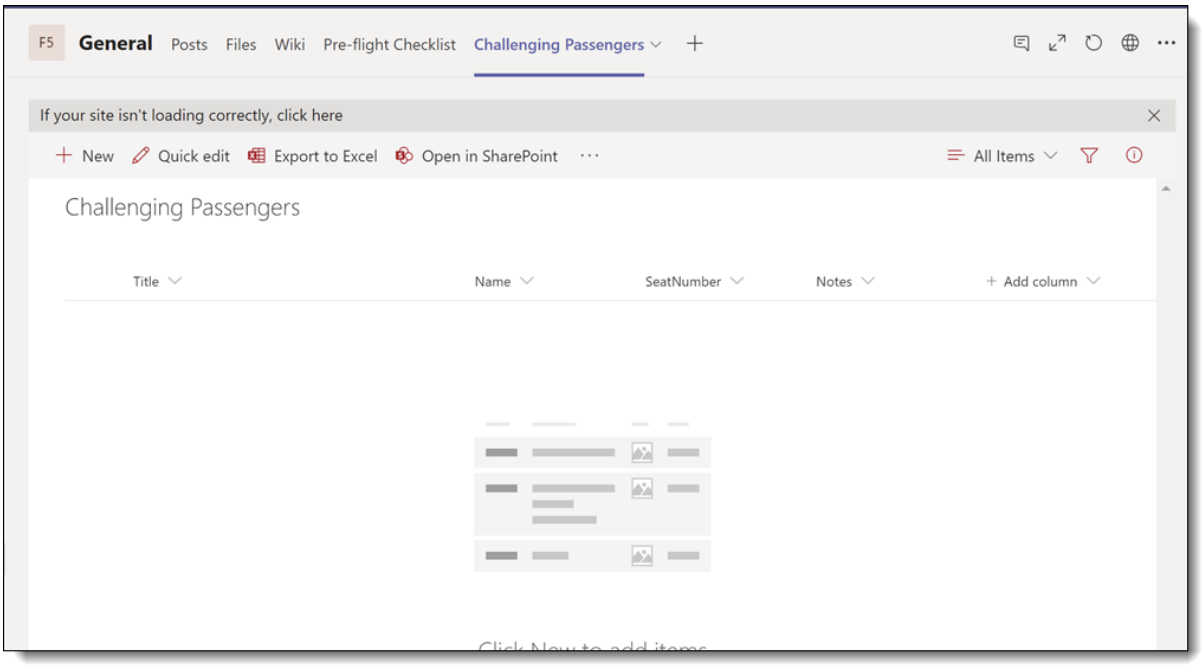
- An additional page (**TeamPage.aspx**) under Site Pages library. The page will also have a Documents webpart added:



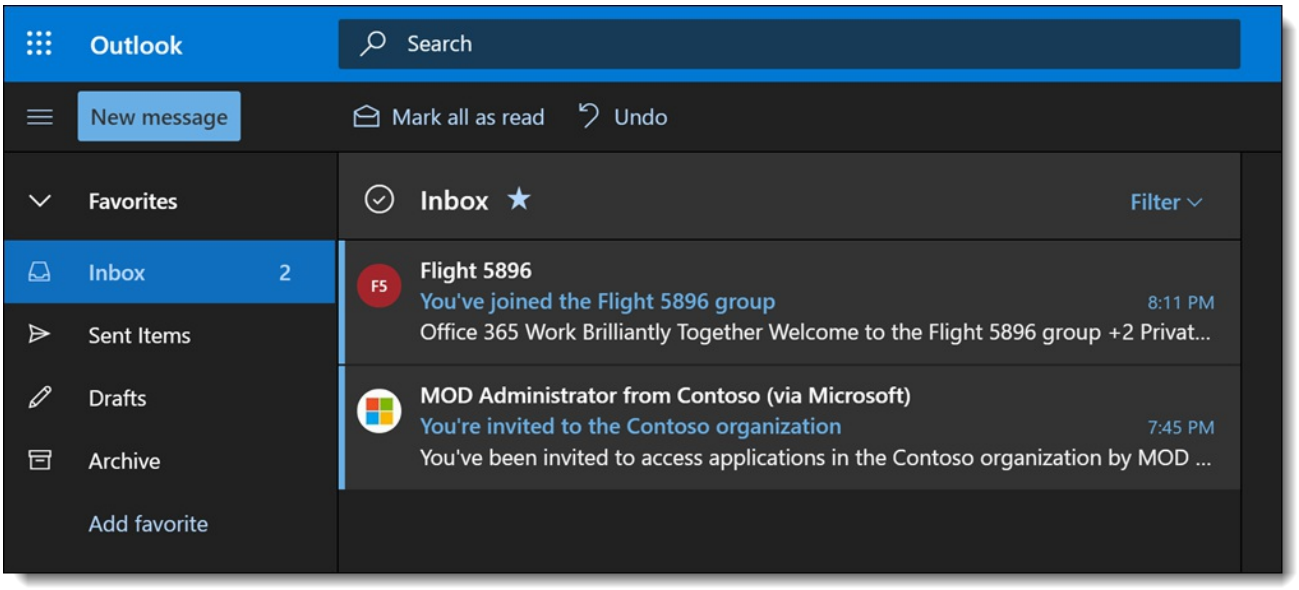
- A tab within the General channel called **Pre-flight checklist**. This will show a **Plan** within Planner with some **custom buckets** and **tasks** provisioned ready for the flight team to check.



- Finally, the General channel also exposes the **Challenging Passengers** list in the as a tab so that the team don't need to go to SharePoint to access that data.



- Additionally, if you access the personal email that you specified under **Catering Liaison** field in the **Flight Admin list**, you should have received an email stating that you were invited to the created Team as a guest user:

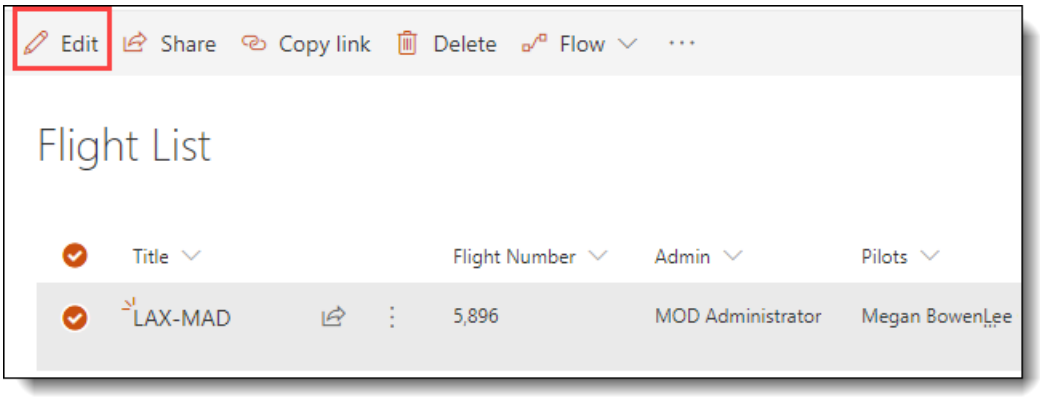


⚙ Updating the Departure Gate

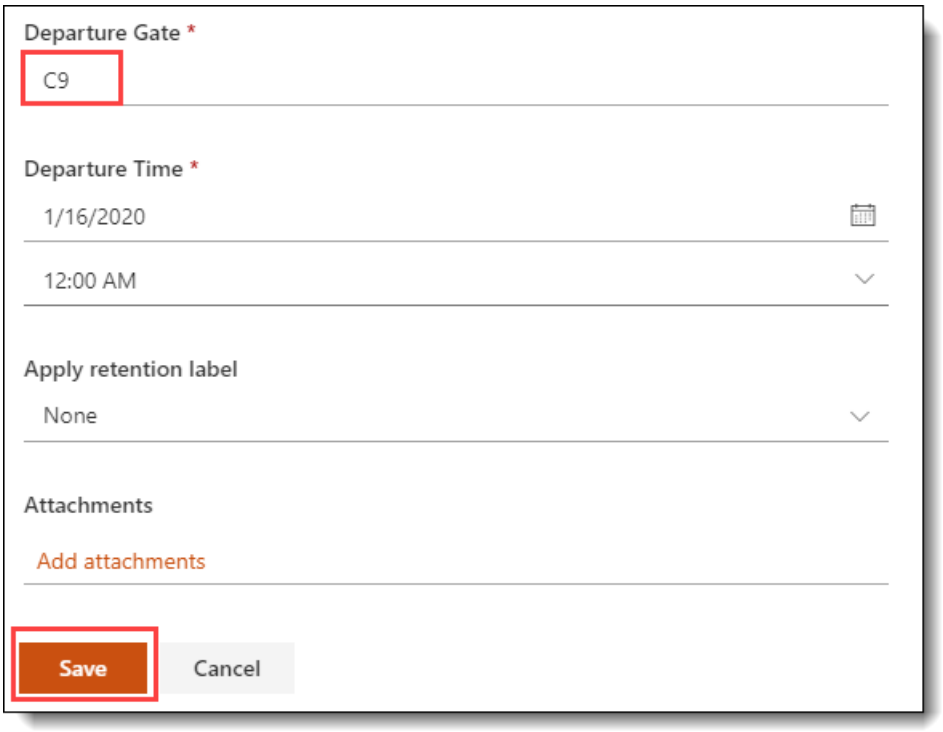
Departure Gate changes is a common scenario in flights. The solution posts a message to all of the created channels (General, Flight Attendants and Pilots) informing about Departure Gate changes so that the crew is up to date.

To test this behavior:

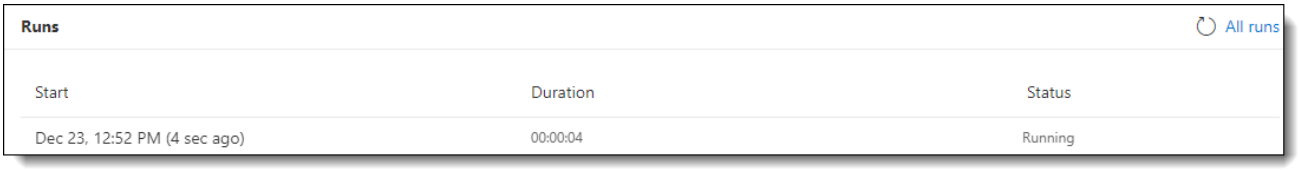
1. Browse to the **Flight List** on the Flight Admin SharePoint Online site.
2. Select and existing flight and click on **Edit**.



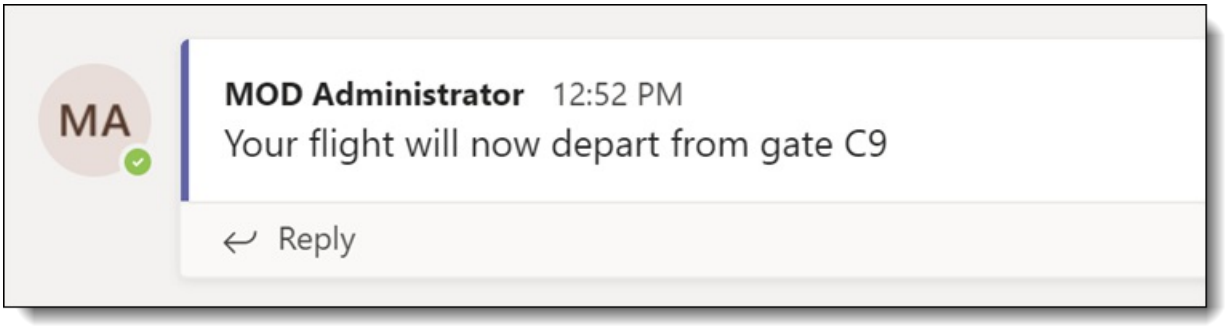
3. On the edit form, change the **Departure Gate** and click **Save**.



4. This should trigger the **Notify Team** flow in Power Automate.

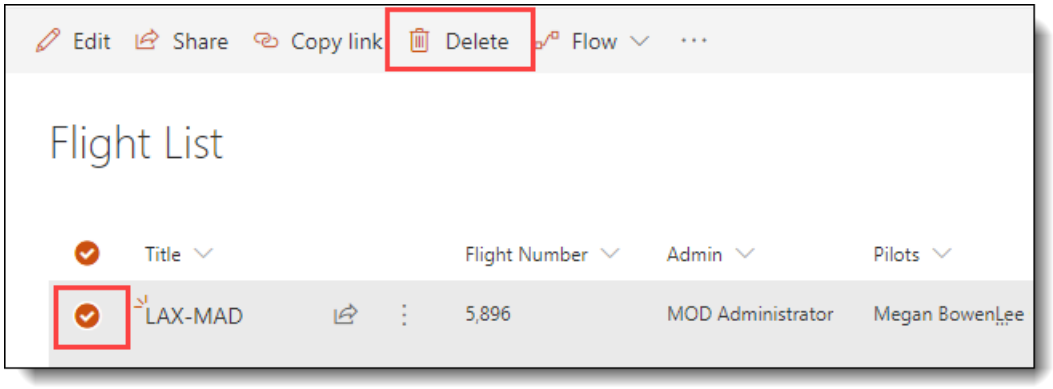


5. After the flow runs, you should see a new message posted on each of the channels of the Team linked to that specific flight informing about the new change.



Deleting a Team

Once a specific flight is over, the created Team needs to be deleted. Doing so is as simple as deleting the desired flight from the **Flight List** in the Flight Admin SharePoint site. The solution will respond to this deletion by triggering the **Archive Team** flow in Power Automate.



After the flow successfully runs, you will see that the team no longer appears in MS Teams as it has been archived.

⚠ You have completed this proof of concept!!!

Great job! You have successfully deployed and configured the Contoso Airlines solution to your test tenant.

This playbook was not meant to be a full-blown production-ready solution guide. However, it has provided with the knowledge on how you can use **Microsoft Graph** and integrate it with O365 and Azure to develop complex solutions. This POC can be further enhanced adding more functionality. Hopefully this serves as a starting point in your Microsoft Graph Journey and you can start crafting solutions that will help your company's digital transformation journey and will empower the employees within your organization to achieve more. Our goal with *Activate - Microsoft Graph for O365* is to show you what is possible and how to begin your own Pilot or Proof of Concept project within your organization. And, most importantly, we hope you had some fun during along the way too. □

If at any point in your Microsoft Graph journey, you find your team hitting a roadblock, have more questions, or need assistance, **please** do not hesitate to reach out to your Microsoft Account Team for our assistance.