Provisioning a Microsoft Team with Approval Flow and Azure Runbook



Prepared by

**Jason Ortiz**

Premier Field Engineer

jason.ortiz@microsoft.com

Revision and Signoff Sheet

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 2/12/18 | Jason Ortiz | 1 |  |
| 4/4/2018 | Jason Ortiz | 1.1 |  |
|  |  |  |  |

Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction & Motivation 6](#_Toc510644848)

[2 Prerequisites 7](#_Toc510644849)

[3 Create a SharePoint List for Requests 9](#_Toc510644850)

[3.1 Introduction 9](#_Toc510644851)

[3.2 Creation Steps 9](#_Toc510644852)

[4 Create an Azure Automation Account and Azure Runbook 10](#_Toc510644853)

[4.1 Introduction 10](#_Toc510644854)

[4.2 Creation Steps 11](#_Toc510644855)

[4.2.1 Create an Azure Automation Account 11](#_Toc510644856)

[4.2.2 Create a Credential 12](#_Toc510644857)

[4.2.3 Create an Azure “Run as Account” 13](#_Toc510644858)

[4.2.4 Create an Azure Runbook 15](#_Toc510644859)

[5 Create the Microsoft Flow for Approval and Provision Kickoff 16](#_Toc510644860)

[5.1 Introduction 16](#_Toc510644861)

[5.2 Creation Steps 17](#_Toc510644862)

[5.2.1 Import the Microsoft Flow package 17](#_Toc510644863)

[5.2.2 Set up connections 18](#_Toc510644864)

[5.2.3 Update the Flow 21](#_Toc510644865)

[5.2.4 Trigger – When an item is created 21](#_Toc510644866)

[5.2.5 Action 1 – Initialize approver variable 21](#_Toc510644867)

[5.2.6 Action 2 – Send approval email 22](#_Toc510644868)

[5.2.7 Condition – Check if approved 22](#_Toc510644869)

[5.2.8 Condition Fail – Action 3B – Send a “request denied” email to requesting user [OPTIONAL] 23](#_Toc510644870)

[5.2.9 Action 3A – Get user profile (V1) 23](#_Toc510644871)

[5.2.10 Action 4A – Create O365 Group 24](#_Toc510644872)

[5.2.11 Action 5AB – Delete request item if creation failed [OPTIONAL] 24](#_Toc510644873)

[5.2.12 Action 6AB – Send a “creation failed” email to requesting user [OPTIONAL] 25](#_Toc510644874)

[5.2.13 Action 5AA – Update Approved Column to ‘Yes’ [OPTIONAL] 25](#_Toc510644875)

[5.2.14 Action 6AA – Add user to O365 Group 26](#_Toc510644876)

[5.2.15 Action 7AA – Delay 26](#_Toc510644877)

[5.2.16 Action 8AA – Create ProvisionMSTeam Runbook Job 27](#_Toc510644878)

[5.2.17 Action 9AA – Get job output 29](#_Toc510644879)

[5.2.18 Switch – Switch on Job Status 29](#_Toc510644880)

[5.2.19 Switch Case 1: Completed [OPTIONAL] 30](#_Toc510644881)

[5.2.20 Switch Case 1 – Action 1 – Update ‘Team Created’ Column to ‘Yes’ 30](#_Toc510644882)

[5.2.21 Switch Case 1 – Action 2 – Send creation confirmation email to requesting user 31](#_Toc510644883)

[5.2.22 Switch Case 2: Suspended [OPTIONAL] 31](#_Toc510644884)

[5.2.23 Switch Case 2 – Action 1 – Send an email mentioning a problem occurred [OPTIONAL] 32](#_Toc510644885)

[5.2.24 Switch Case 2 – Action 2 – Send an email to admin about suspended job 32](#_Toc510644886)

[5.2.25 Default Switch Case – Send an email to admin 33](#_Toc510644887)

[6 Recommended Testing & Debugging Tips 34](#_Toc510644888)

[7 Create a PowerApp for Provisioning Requests (Optional) 35](#_Toc510644889)

[8 Conclusion 36](#_Toc510644890)

1. Introduction & Motivation

This document outlines the steps necessary to create an *automated Microsoft Teams provisioning solution* which uses PowerApps, Microsoft Flow with Approval Request, Azure Automation, and Azure Runbooks with Microsoft Teams PowerShell. To implement the solution, review these four sections:

1. Prerequisites
2. Create a SharePoint List for Requests
3. Create an Azure Automation Account and Azure Runbook
4. Create the Microsoft Flow for Approval and Provision Kickoff

The motivation for this solution comes from the need to govern how an organization’s users utilize Office 365 Groups and Microsoft Teams. By default, any user in an organization can create an Office 365 Group and thus a Microsoft Team. It is reasonable to want to limit the creation of O365 Groups and Teams, so organizations may restrict this creation process to a limited number of select users. However, an organization may still want to allow their users to utilize Microsoft Teams, and that’s how this solution becomes useful. Allowing users to request the creation of a Microsoft Team, having that request reviewed, then approved or denied, and most importantly, having a Microsoft Team automatically provisioned upon approval, allows users to still be productive within Office 365 *and* still allow administrators the flexibility to govern their use and save themselves time from the provisioning process.

The flow chart on the next page diagrams the solution workflow:

User submits request for new Microsoft Team

Approval request email sent to specified user

Approve?

Create Office 365 Group

Add user as group member

Call Azure Runbook

Create new Microsoft Team using Office 365 Group ID

Add user as Office 365 Group Owner

Remove Team creator as Office 365 Group Owner

Email user and admin about provision success

**Key**

SharePoint Action

Microsoft Flow Action

Azure Automation Action

Yes

1. Prerequisites

Before implementing this solution, be sure to configure the following prerequisites:

1. [Configure who can create Office 365 Groups](https://support.office.com/en-us/article/Manage-who-can-create-Office-365-Groups-4c46c8cb-17d0-44b5-9776-005fced8e618)
   1. If every user is still able to create an Office 365 Group and thus a Microsoft Team still, then this solution has no purpose.
2. Have tenant admin permissions
3. Consider creating a “service account” to run the Flow and create the Team
   1. Actions performed by the solution can be done with accounts of your choice, such as a global admin, but in my opinion, it is unwise to tie these solutions to a real user. Instead, it is my recommendation to create a “service account” which will perform the actions in this solution. Specifically, the account will:
      1. Send emails to users and admins with provisioning statuses
      2. Create the Office 365 Group
      3. Create the Microsoft Team
   2. Using a service account is useful for several reasons:
      1. In Microsoft Flow, actions are performed with a configured connection. By default, this connection uses the account of the user that created the Flow. We may not want the emails sent to users by the Flow to appear as if they were sent by the person who created the Flow, i.e. you
      2. By adding a different connection and selecting that to be used within our Flow, actions will be performed by that connection/account instead.
      3. Actions like “create an Office 365 Group” and “New-Team” automatically add the chosen account as an owner to the group and team. Even though this solution removes the Group and Team creator at the end, a user can still see the Group and Team momentarily in OWA and the Teams application, which can be undesirable behavior.
   3. To create a service account, follow these steps:
      1. [Add a user individually to Office 365](https://support.office.com/en-us/article/Add-users-individually-or-in-bulk-to-Office-365-Admin-Help-1970f7d6-03b5-442f-b385-5880b9c256ec) 
         1. Give the account a UPN like [O365Workflow@domain.com](mailto:O365Workflow@domain.com)
         2. Make sure it has licenses for Exchange Online, SharePoint Online, Flow for Office 365, PowerApps for Office 365, and Microsoft Teams
         3. Assign it the “Global administrator” role
            1. **Note**: this assignment will only be temporary. Later, the admin role should be changed to “User management administrator.” The account will need global admin rights to properly approve of using the Azure AD connector in Microsoft Flow later.
      2. Add the account to the Azure AD security group of users that can create Office 365 Groups
         1. [Configure who can create Office 365 Groups](https://support.office.com/en-us/article/Manage-who-can-create-Office-365-Groups-4c46c8cb-17d0-44b5-9776-005fced8e618)
4. Create a SharePoint List for Requests
   1. Introduction

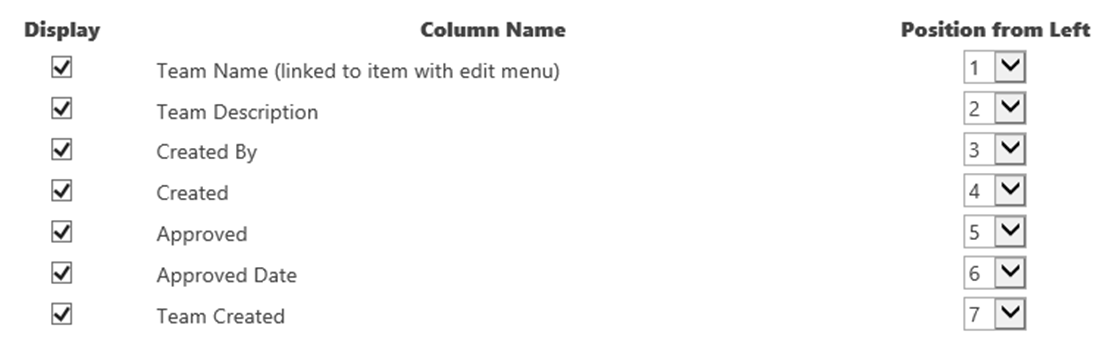
The requests for this solution will be stored in a SharePoint List. They could be stored elsewhere, like an on-premises database, a database in Azure, the Common Data Service, etc., but for our purposes, SharePoint lists are easy to create and manage. This list can be created on any existing SharePoint site or a new SharePoint site. However, it is worth considering ahead of time how and where users should make their requests:

* A private or public SharePoint site?
  + If private, any requesting user must have permissions to add items to the list
* Should a user be able to see other requests made?
  + If you want to allow users to only see their own items, advanced settings must be configured:  
    A screenshot of a cell phone

    Description generated with very high confidence
* Should requests be made via a SharePoint List or PowerApps?
  + A PowerApp could be created that allows for users to submit requests
  + The PowerApp could be added as a web part to a SharePoint site, deployed as an app across the organization, and used on mobile and desktop clients
  + Refer to Section 7 for more information on creating a PowerApp for this solution
  1. Creation Steps

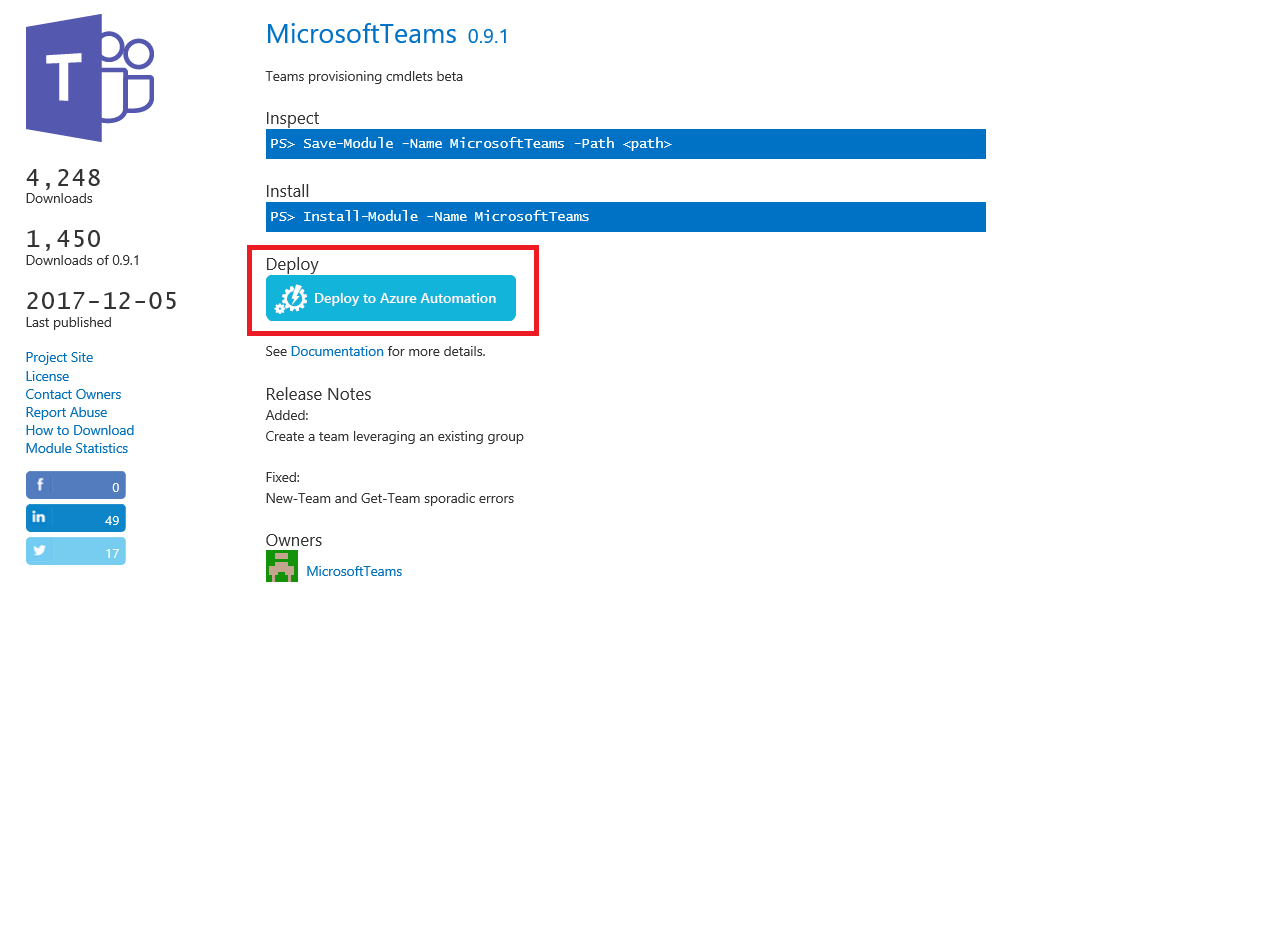
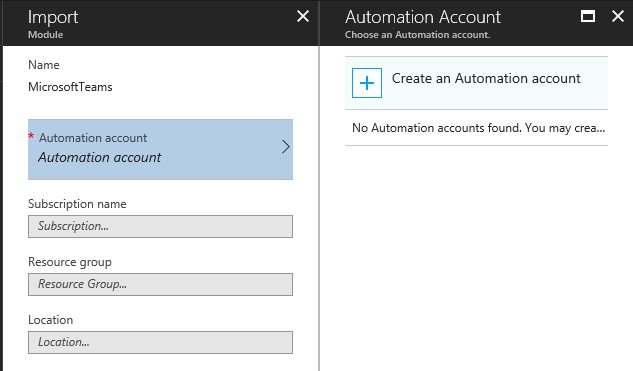
1. Create a custom SharePoint List. I named mine “Microsoft Teams Requests”
2. Go to List Settings, which can be found under the gear
3. Under **General Settings**, select **Advanced Settings**
4. Under **Allow Management of content types?**, choose **Yes,** then click **OK**
5. Under **Content Type**, select **Item**
   1. Select **Title**, then select **Edit Column**
   2. Rename the column to **Team Name**, click **OK**, then **OK** again
6. Select **Settings** in the breadcrumb near the top left of the page to get back to the **List Settings**
7. The list needs the following columns created (Select **Create Column** under the list of columns to create new ones):

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Column Type** | **Required?** | **Default Value** |
| Team Description | Multiple lines of text | No |  |
| Approved | Yes/No | No | No |
| Approved Date | Date and Time | No |  |
| Team Created | Yes/No | No | No |

1. Select **All Items** under **Views**
2. Select and order the columns like in this photo, then click **OK**:  
   
3. The list is ready to go!
4. Create an Azure Automation Account and Azure Runbook
   1. Introduction

To automate the Microsoft Teams provisioning process, we will use [Azure Automation](https://docs.microsoft.com/en-us/azure/automation/automation-intro) and [Azure Runbooks](https://docs.microsoft.com/en-us/azure/automation/automation-runbook-types). Azure Automation is a “cloud-based automation and configuration service” which we will use for “process automation” via Azure Runbooks. An Azure Runbook allows for PowerShell scripts to be run remotely. Runbooks can be started in [several ways](https://docs.microsoft.com/en-us/azure/automation/automation-starting-a-runbook), but for our solution, we will use Microsoft Flow. To utilize runbooks, an Azure Automation account must be created first, and then a PowerShell runbook can be created. Azure Automation has [existing PowerShell modules](https://docs.microsoft.com/en-us/azure/automation/automation-runbook-gallery#modules-in-powershell-gallery) from the [PowerShell Gallery](http://www.powershellgallery.com/) available for use. For our solution, we will need to use the latest version of the [Microsoft Teams PowerShell module](https://www.powershellgallery.com/packages/MicrosoftTeams/0.9.1https:/www.powershellgallery.com/packages/MicrosoftTeams/0.9.1) from the PowerShell Gallery. Follow the steps in the next section to create an Azure Automation Account, import the Microsoft Teams PowerShell Module, and create the Azure Runbook.

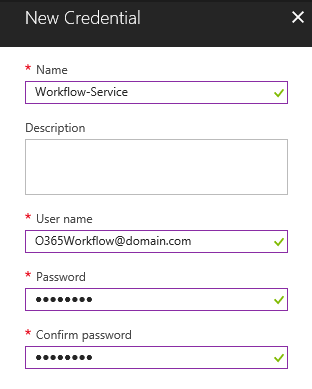
* 1. Creation Steps
     1. Create an Azure Automation Account

If you do not have an Azure Automation account already, you can simultaneously create an account and import the Microsoft Teams PowerShell module by clicking the **Deploy to Azure Automation** button on [this page](https://www.powershellgallery.com/packages/MicrosoftTeams/0.9.1) and going through the import wizard as seen below:  


* 1. Alternatively, learn how to [create an Azure Automation Account](https://docs.microsoft.com/en-us/azure/automation/automation-quickstart-create-account) separately
  2. **Note**: An Automation Account will require an Azure subscription, such as a free trial, Pay-As-You-Go, etc. As stated in the [pricing details](https://azure.microsoft.com/en-us/pricing/details/automation/), “billing for [runbook] jobs is based on the number of job run time minutes used in the month.” There are 500 minutes included per month. A charge of $0.002/minute will be incurred for every minute over 500. This runbook should run at most for about 30 seconds, which means about 1,000 runs included.

1. **OPTIONAL:** Pin Automation Accounts to services blade and dashboard for convenience and quickness later
   1. In the search bar at the top of the page in Azure, search for “automation,” and click on the automation account just created
   2. Click the pin in the top right to pin the blade to the dashboard
   3. In the leftmost blade, select **All Services**
   4. In the filter above, enter “automation”
   5. Click the star to add it to the services blade  
      
      1. Create a Credential

This credential will be used within the Azure Runbook to authenticate to Microsoft Teams, and thus the user chosen must be allowed to create Microsoft Teams. I chose to use the workflow service account mentioned in Section 2 for the credentials. This is useful since anyone that creates a Team is also temporarily added as an owner, and although the script will remove that user from the Team, it does not update immediately in the Teams application, which can add a lot of unwanted Teams in your application.

1. Find your Automation Account and select it
2. Under **Shared Resources**, select **Credentials**, then select **+ Add a credential**
   1. Enter a name for this credential (I chose “WorkflowService”), then enter the username and password of the user.  
      
      1. Create an Azure “Run as Account”

An Azure “Run as Account” is a service principal that can perform Azure AD operations. This account will be used to connect to Azure AD PowerShell. It will also be used later as a connection account in Microsoft Flow. Using this account is important so that an account tied to a real user is not built into the process.

1. Find your Automation Account and select it
2. Under **Account Settings**, select **Run as accounts**
3. Under **Azure Run As Account**, select **Create**, then go through the wizard to create a Run As Account
   1. Note: The Azure Run As Account will expire one year from when it was created, so be sure to renew the certificate before it expires
4. After creation, your account is ready to be used within the script.  
   A screenshot of a cell phone

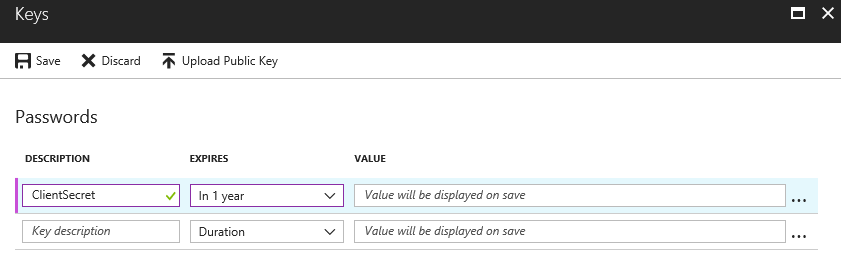
   Description generated with very high confidence

The service account still needs permissions to perform Azure AD operations such as getting user and group data, and to run the runbook.

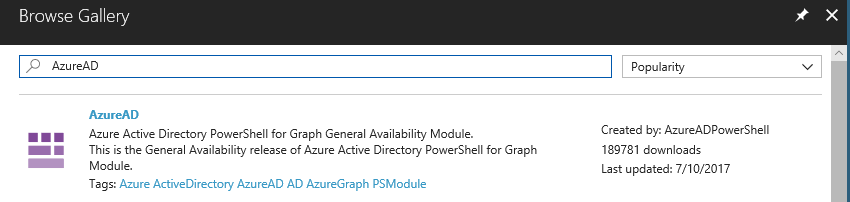
1. Select **Azure Active Directory** from the services list in the leftmost blade
2. Under **Manage**, select **App registrations**, make sure “All apps” is chosen from the dropdown, then select the app that starts with the name of your Automation Account followed by a long string of alphanumeric characters  
   
3. Select **Settings** near the top left of the new page that opens
4. Select **Required permissions**, **+ Add**, and under **Select an API**, select **Microsoft Graph**  
   A screenshot of a cell phone

   Description generated with very high confidence
5. For permissions, select **Read and write all groups**, press **Select,** then click **Done**
6. Click **+Add**, and under **Select an API**,select **Windows Azure Active Directory**
7. For permissions, select **Read and write directory data**, press **Select,** then click **Done**
8. Once added, click **Grant permissions** to finish the permission assignment process.

You will also need to create a client secret key, record it somewhere safe to be used later, and record the client/application ID and tenant/directory ID.

1. Under **API Access**, select **Keys**
2. Type a name for the key (e.g. “ClientSecret”)
3. Select an expiration time
4. Click the **Save** button
5. Copy the key value and record it somewhere safe. Do not share this key.
6. To get the client/application ID, go back to the screen where you selected the **Settings** button
7. Copy and record the **Application ID**
8. To get the tenant/directory ID, select the Azure Active Directory Service
9. Under **Manage**, select **Properties**
10. Copy and record the **Directory ID**

Finally, you will need to add the AzureAD PowerShell module for use by the runbook. This will let the script find and modify Azure AD groups and users.

1. Find your Automation account and select it
2. Under **Shared Resources**, select **Module Gallery**
3. Search for “AzureAD,” then select it
4. Click the **Import** button  
   * 1. Create an Azure Runbook

The Azure Runbook is the heart of this solution. Azure Automation allows us to call this runbook using REST APIs/webhooks, but in this solution, the process is made simpler by using Microsoft Flow to start the runbook. Here is an [introduction to Azure Automation](https://docs.microsoft.com/en-us/azure/automation/automation-intro) if interested. The PowerShell script that creates the Microsoft Team will be within a runbook.

1. In the Azure portal, open your Automation Account
2. From the Hub, select **Runbooks** to open the list of runbooks
3. Click on the **Add a runbook** button and then **Import**
   1. Download the ZIP from this link: <https://github.com/jason-ortiz/provision-ms-team/blob/master/Provision%20MS%20Team%20Files.zip>
   2. Extract the two files, one being the PowerShell script for the runbook
4. Click **Runbook file** and select the script just downloaded
5. If the **Name** field is enabled, then you have the option to change it. The runbook name must start with a letter and can have letters, numbers, underscores, and dashes
6. The [runbook type](https://docs.microsoft.com/en-us/azure/automation/automation-runbook-types) is automatically selected, but you can change the type after taking the applicable restrictions into account. The type should be **PowerShell**
7. The new runbook appears in the list of runbooks for the Automation Account

**Note**: If you chose a name for the credential other than “WorkflowService,” you must edit the runbook and update the “-Name” parameter on line 11, then click **Publish**

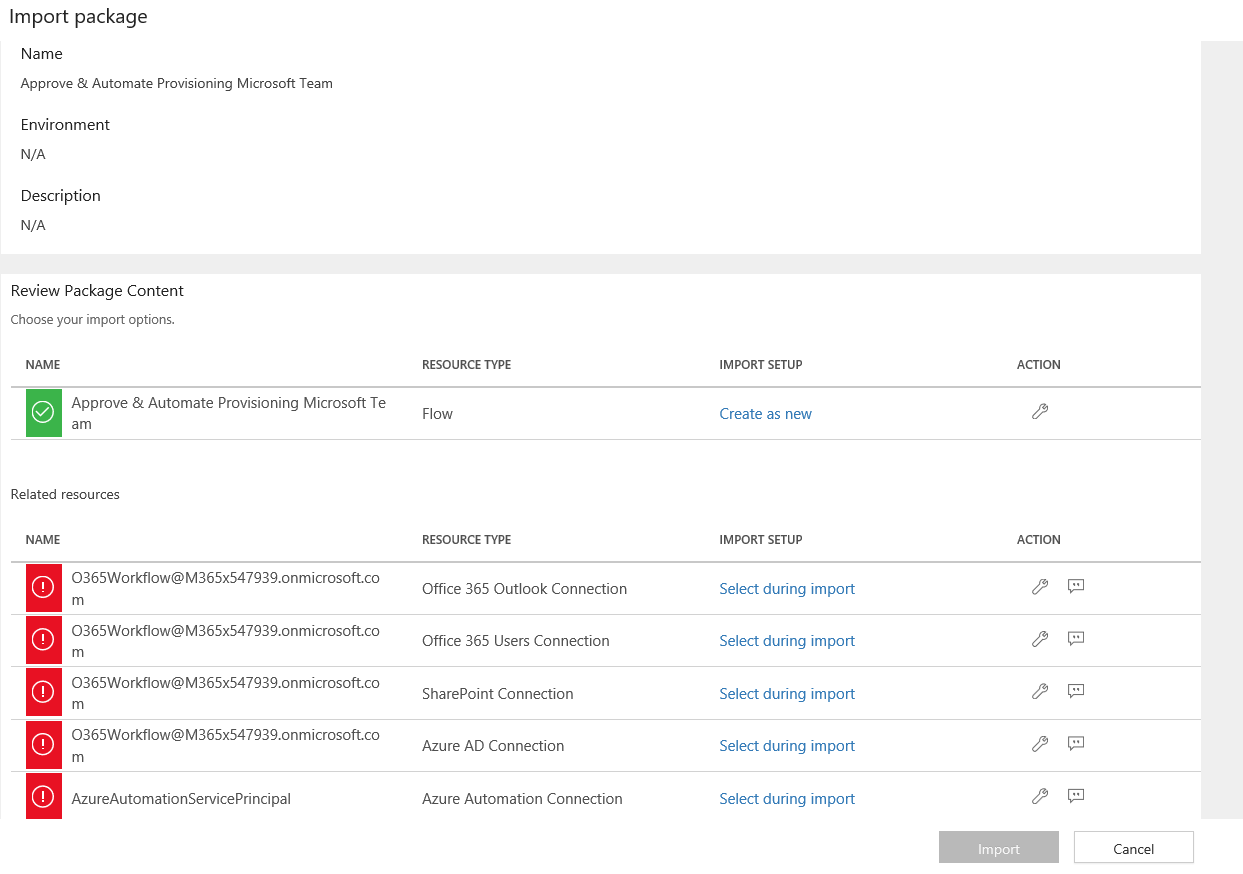
**A screenshot of a social media post

Description generated with very high confidence**

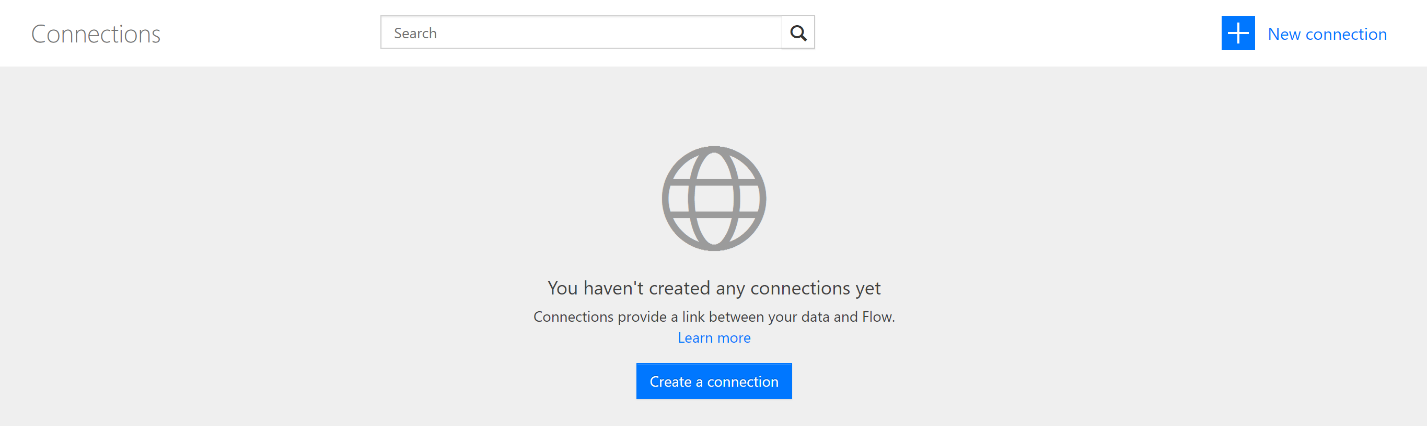
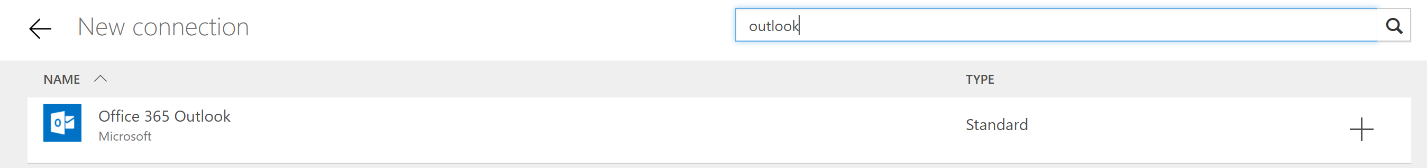
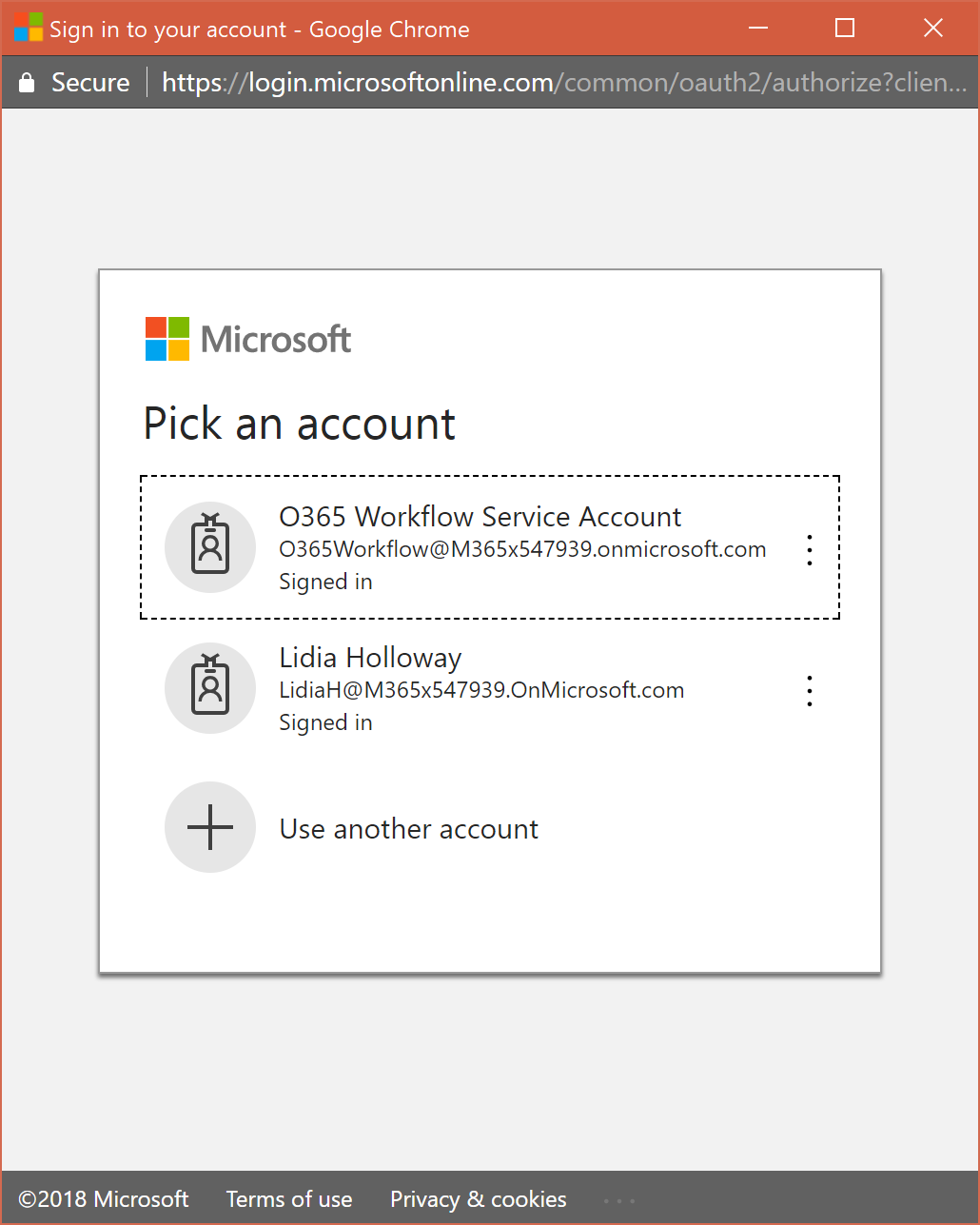
1. Create the Microsoft Flow for Approval and Provision Kickoff
   1. Introduction

From [MS Flow - Guided Learning](https://docs.microsoft.com/en-us/flow/guided-learning/get-started#step-1) documentation, “Microsoft Flow is an online **workflow service** that enables you to work smarter and more efficiently by **automating workflows** across the most common apps and services.” In this solution, our Flow will start whenever an item is added to the SharePoint list. It will then perform the actions necessary to request approval, and upon approval, call the Azure Runbook, which will do the heavy lifting of creating the Microsoft Team. Follow the steps below to create the Flow.

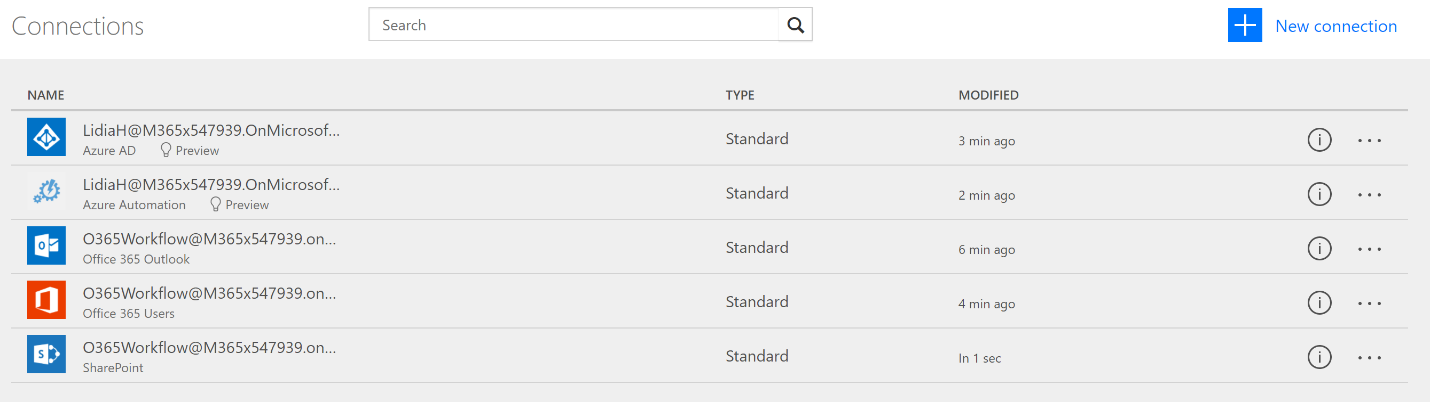
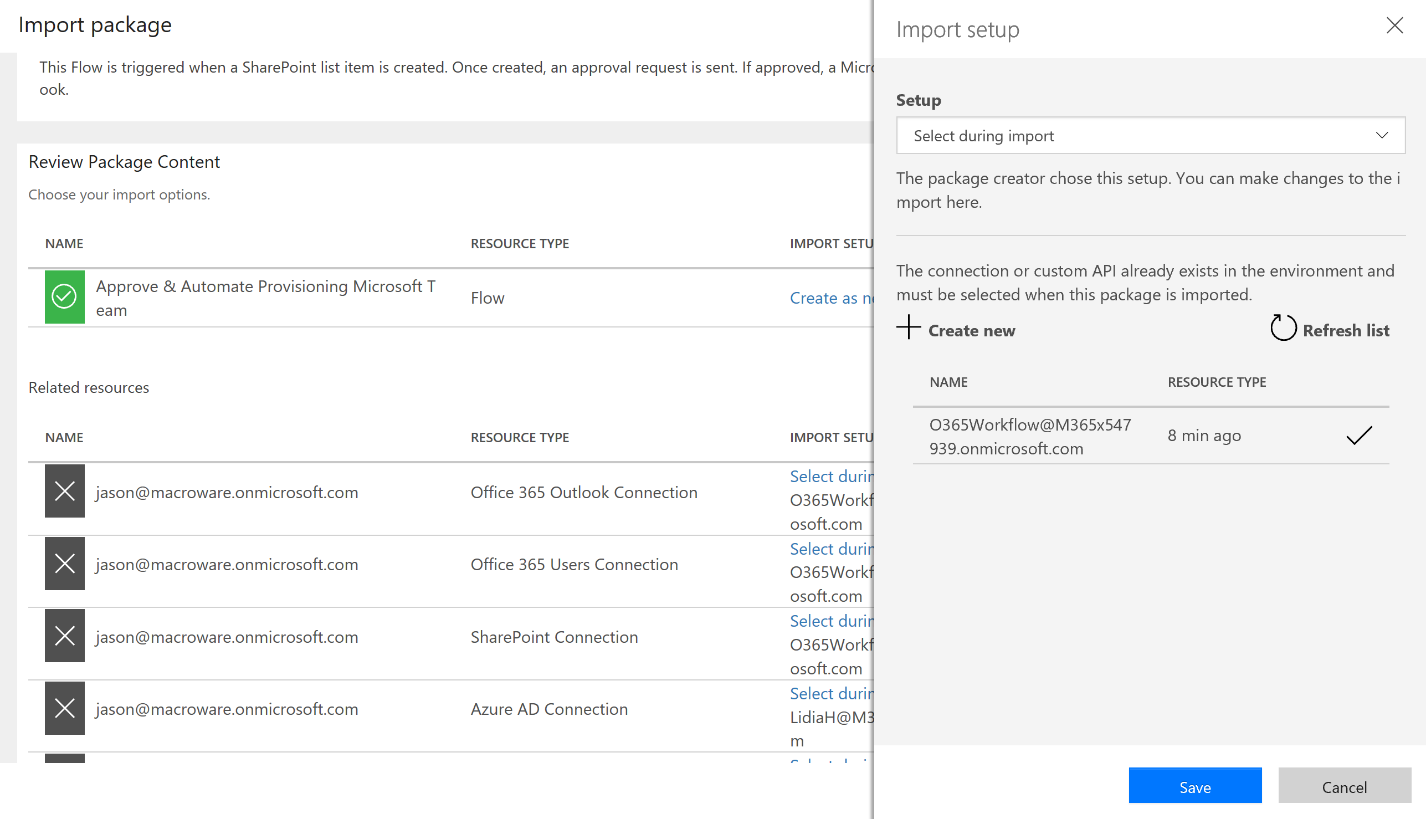
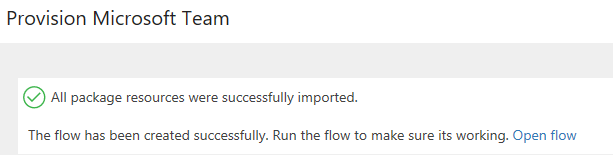
* 1. Creation Steps
     1. Import the Microsoft Flow package

1. Go to <https://flow.microsoft.com>
2. Select **My Flows** on the navigation bar
3. Click the **Import** button 
4. Find the .zip package downloaded from GitHub earlier. There is another .zip package inside name **ApproveAndAutomateProvisioningMicrosoftTeam\_Package.zip** - Upload that file
5. After upload, you should see the following:  
   
   * 1. Set up connections

Each trigger and action of a Microsoft Flow workflow requires a connection. A connection is essentially a configuration of permissions to access some resource (e.g. Outlook, SharePoint, Azure AD, etc.). Actions run in the context of the connection/user. With that said, it is important to decide early on whose context to use for these connections so that as the workflow operates, it does not seem like you are personally performing the actions, such as emailing users, modifying SharePoint list items, etc.

1. Select any of the **Select during import** links
2. Click **+ Create New**
3. You should see the following assuming you have no connections in Microsoft Flow  
   
4. Click **Create a connection** or **+ New Connection**
5. Search for “Outlook” and select the **Office 365 Outlook** connection  
   
6. Select **Create** and this window should appear  
   
7. For this connection, I recommend using the workflow service account created earlier within Office 365. Enter the credentials for that account. All emails sent in the workflow will use this account as the sender.
8. Repeat steps 4-6 for the other connections, searching by connection name. Refer to this table to decide which account to use for sign-in

|  |  |
| --- | --- |
| **Connection** **Name** | **Account to Use** |
| Azure AD | Flow service account |
| Azure Automation | Your admin account |
| Office 365 Users | Flow service account |
| SharePoint | Flow service account (also must be member of SharePoint site of the list being used) |

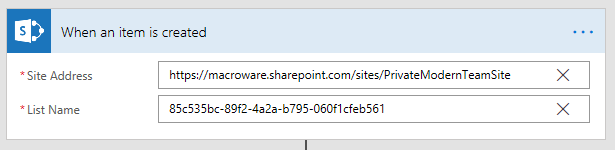
1. This is final list of connections you should have (Note: in my case, I chose my admin account for the Azure AD connection, but I still recommend using the service account)
2. Go back to the previous tab where the Flow import is still being configured
3. Click each **Select during import** link, refresh the list if necessary, and select the connection in the list, then click **Save  
   **
4. Once all the connections are selected, click the **Import** button at the bottom of the page. If successful, the result should be this:  
   
5. Click the **Open flow** link to start modifying the Flow
   * 1. Update the Flow

After import, it is essential to go through the workflow and review the trigger and each action for accuracy and errors. Configure them to suit your own needs if necessary. Not all the actions in the Flow are absolutely necessary, but they are still convenient. Any actions like this will say “**[OPTIONAL]**” in the section title and could be removed from the Flow if desired. In the next sections, the Flow trigger and actions’ behaviors are explained, and any actions that need remediation are discussed. Please review the **Action Needed** in each section.

* + 1. Trigger – When an item is created

**Description:** This workflow will trigger whenever an item is created in a SharePoint list of your choosing. For your workflow, you must select the site address and corresponding list being used for this solution that was created earlier.

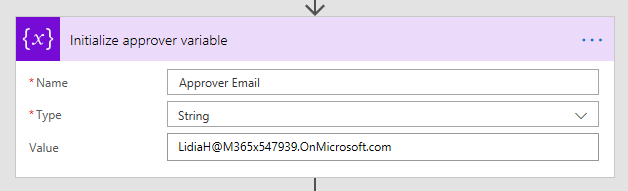
**Action Needed:** Clear the current site address and list name of the trigger and choose yours instead.



* + 1. Action 1 – Initialize approver variable

**Description:** This action creates a variable which should contain the email address of the mailbox where approval requests should go. The variable is used in many of the email actions later in the workflow, so this action is meant to serve as a convenient place to set the value once.

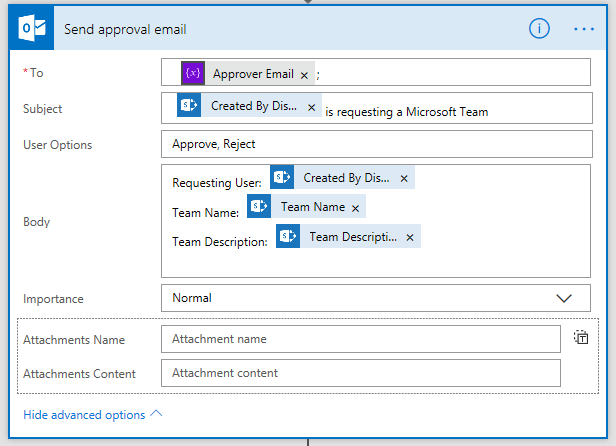
**Action Needed:** Change the value of this variable to an email address where you want approval requests to go.



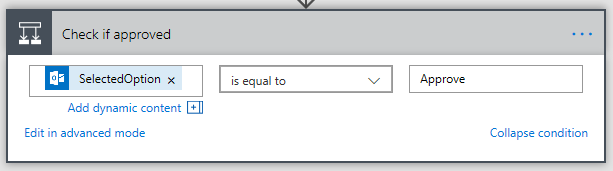
* + 1. Action 2 – Send approval email

**Description:** This action sends an email to the approver email listed earlier. The receiving user can choose “approve” or “reject.”

**Action Needed (Optional):** Update the subject and/or body of the email to read as you want.



* + 1. Condition – Check if approved

**Description:** This condition will check if the response from the approval email is equal to “Approve”  


**If yes:** continue the workflow

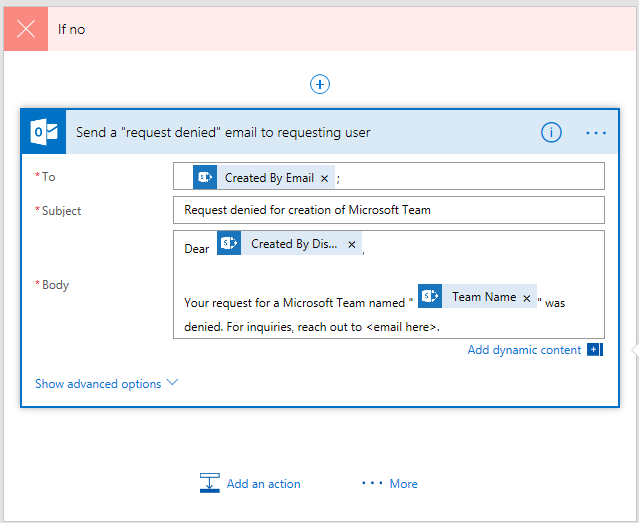
**If no:** Send an email to the requesting user

* + 1. Condition Fail – Action 3B – Send a “request denied” email to requesting user [OPTIONAL]

**Note:** This action is not absolutely necessary but is nice to have so a user can know if their request for a Team was denied.

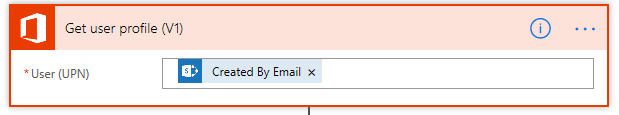
**Description:** This action will send an email to the user who created the original request list item. It will mention their request has been denied.

**Action Needed:** Within the body of the email, replace “<email here>” with an email of your choosing, or remove it entirely.

**Action Needed (Optional)**: Update the subject and body to read as you want.  


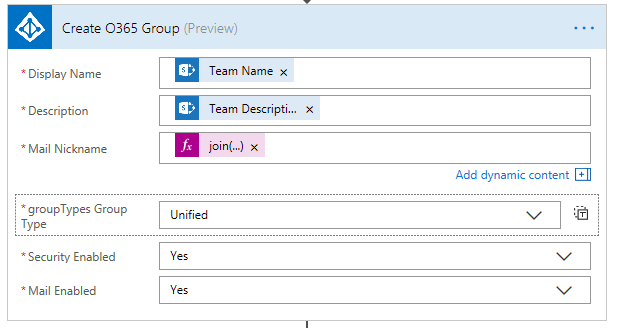
* + 1. Action 3A – Get user profile (V1)

**Description:** This action will get the user profile information for the user that created the request. It will use their email address to find their info. Their **User id** is used later.

****

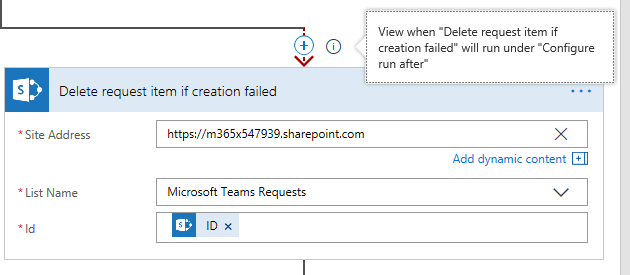
* + 1. Action 4A – Create O365 Group

**Description:** This action will create an Office 365 Group in Azure Active Directory. This group will later be referred to when creating the Microsoft Team. It will use the name and description supplied from the SharePoint list item. For the mail nickname, expressions, which are like functions, are used to remove any spaces from the given name to create a valid mail nickname.

**Action Needed:** Because the workflow service account now has a connection setup for Azure AD, it no longer needs Global Admin permissions. Changing the service account’s permissions to “User management administrator,” which can be done from the [O365 Admin Center](https://support.office.com/en-us/article/Assign-admin-roles-in-Office-365-for-business-eac4d046-1afd-4f1a-85fc-8219c79e1504), [PowerShell](https://technet.microsoft.com/en-us/library/mt736914.aspx), or [Azure AD](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-assign-admin-roles-azure-portal).  


* + 1. Action 5AB – Delete request item if creation failed [OPTIONAL]

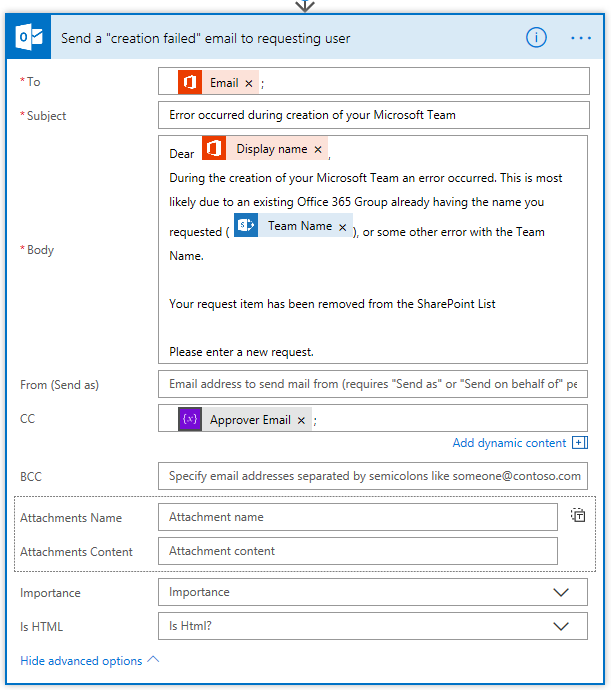
**Description:** This action will run only if the previous action (4A) failed, meaning it will run only if there was a problem creating the Office 365 Group. If there is a failure, the SharePoint list item will be deleted.

**Action needed:** Decide if this functionality is necessary for you. If so, then update the site address and list name to yours.  


* + 1. Action 6AB – Send a “creation failed” email to requesting user [OPTIONAL]

**Description:** This action will send an email to the user the made the request describing that there was a problem creating their Team and options for what to do next.

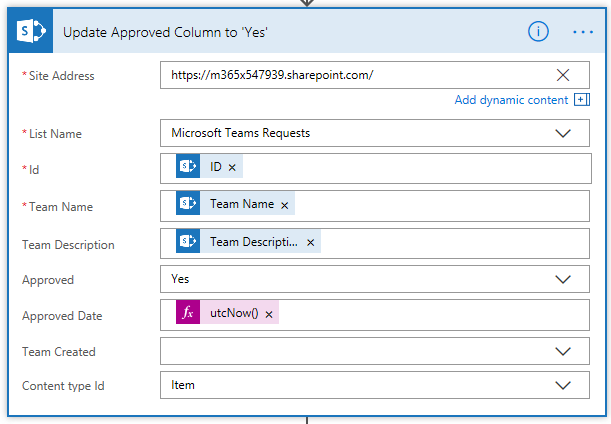
**Action needed:** Decide if this action is necessary. If so, review the subject and body and update as necessary.



* + 1. Action 5AA – Update Approved Column to ‘Yes’ [OPTIONAL]

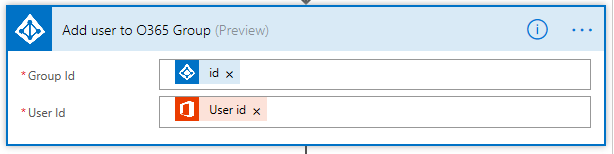
**Description:** This action will update the Approved Column to “Yes” in the SharePoint list and set the approved date to the current date.

**Action needed:** Decide if this functionality is necessary. If so, update the site address and list name to yours.



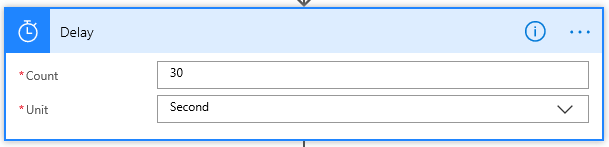
* + 1. Action 6AA – Add user to O365 Group

**Description:** This action will add the user requesting the Microsoft Team as a member to the Office 365 Group using their user ID.

No action needed.  


* + 1. Action 7AA – Delay

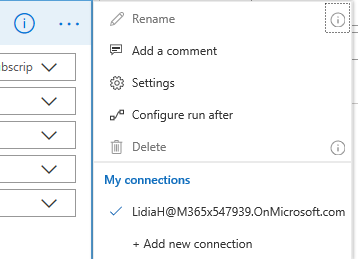
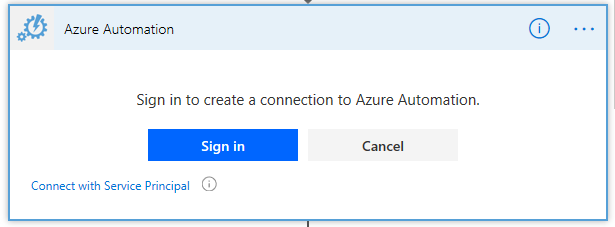
**Description:** This action will pause the workflow for 30 seconds. This allows the recently Office 365 Group to populate throughout the services so that when Microsoft Teams refers to it later, it can successfully find the group.

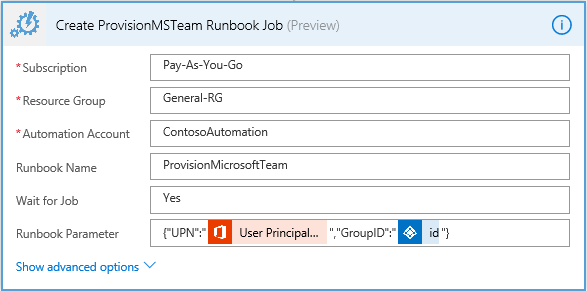
No action needed. Feel free to experiment with the delay time. 30 seconds was arbitrary but helped avoid errors later.  


* + 1. Action 8AA – Create ProvisionMSTeam Runbook Job

**Description:** This is the most important action of the Flow. The action will create a new job to run the Azure Runbook, which will in turn create the Microsoft Team. It will pass in as parameters the Office 365 Group ID and requesting user’s UPN.

**Action Needed:** When you first open the action, there will be an error because there is old information.

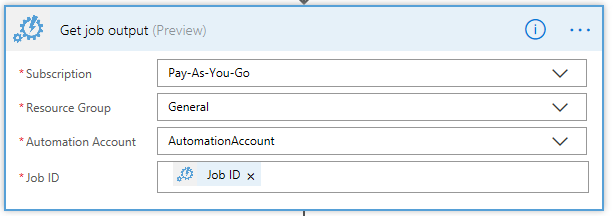
1. First, add a different connection. Specifically, use the Azure Run As account created earlier in Azure AD to perform the operations so that this workflow is not tied to the user creating it.
2. Click the ellipses in the top right, then click **+ Add new connection**  
   
3. Select **Connect with Service Principal**
4. Name the connection
5. Enter the Client/Application ID, Client Secret Key, and Tenant/Directory ID recorded earlier in Section 4.2.3
6. Click create
7. Update the fields to use the **Subscription**, **Resource Group**, **Automation Account**, and **Runbook Name** for your account
   1. For Runbook Parameter, paste in the following if nothing appeared:
8. The result should look something like this:  
   A screenshot of a cell phone

   Description generated with very high confidence  
   If it looks like the below picture instead, except without the data within “Runbook Parameter,” then paste in this value exactly: **{“UPN”:”body(‘Create\_O365\_Group’)?[‘id’])”,”GroupID”:”body(‘Get\_user\_profile\_(V1)’)?[‘UserPrincipalName’]”}**  
   
   * 1. Action 9AA – Get job output

**Description:** This action will get the runbook job output. In this workflow, the job status is used. The job status can be something like “completed,” “suspended,” or “failed.”

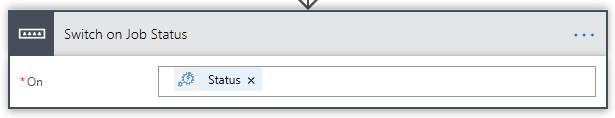
**Action needed:**

1. Select the ellipses in the top right
2. Select the service principal account connection created in the previous section
3. Update the **Subscription**, **Resource Group**, and **Automation Account** to your own

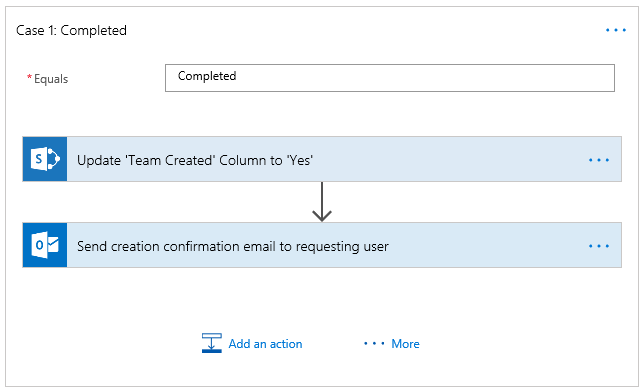


* + 1. Switch – Switch on Job Status

The switch will choose a path for the workflow based on its input. For this switch, the job status will determine how the workflow should proceed.

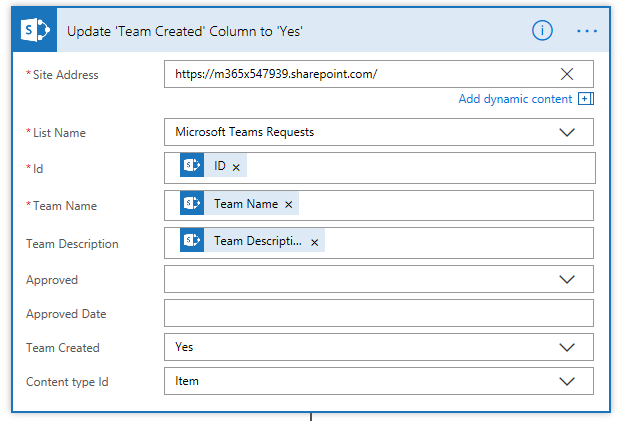


* + 1. Switch Case 1: Completed [OPTIONAL]



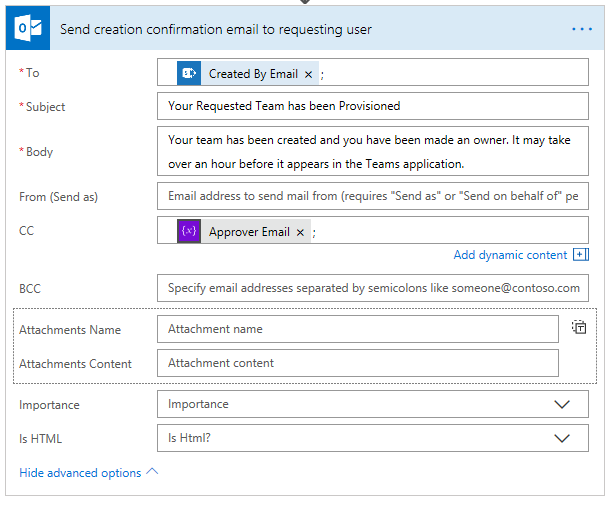
* + 1. Switch Case 1 – Action 1 – Update ‘Team Created’ Column to ‘Yes’

**Description:** This action will update the “Team Created” column in the SharePoint list to “Yes” if a Microsoft Team was successfully created.

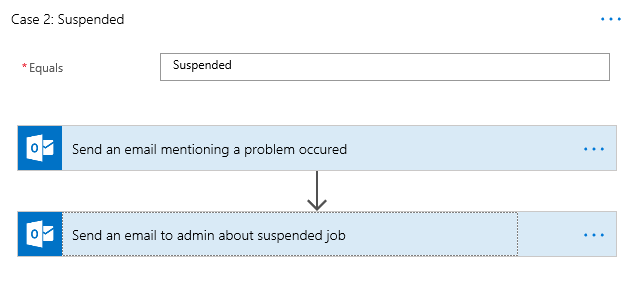
**Action needed:** Update the **Site Address** and **List Name** to your own.  
  


* + 1. Switch Case 1 – Action 2 – Send creation confirmation email to requesting user

**Description:** This action will send an email to the requesting user describing that their Microsoft Team was created and CC the approver email.

**Action Needed:** Review the subject and body of the email to make sure it reads as you want.  


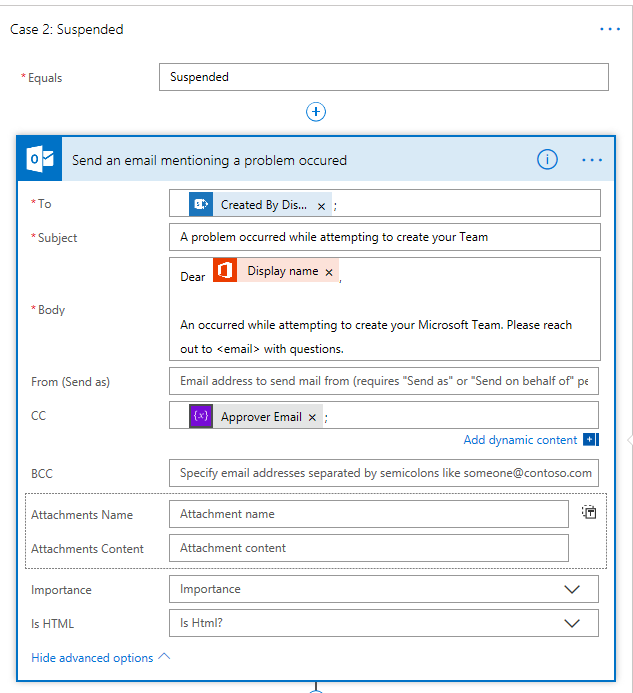
* + 1. Switch Case 2: Suspended [OPTIONAL]

If the job was suspended, this case will be chosen. A suspended job can be restarted or cancelled in Azure AD. Currently, the runbook should never suspend, but the case is here to cover all scenarios.  


* + 1. Switch Case 2 – Action 1 – Send an email mentioning a problem occurred [OPTIONAL]

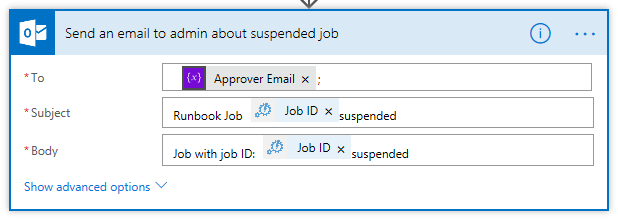
**Description:** This action will send an email to the requesting user and the approver describing that a problem occurred.

**Action needed:** Review the subject and body of the email and update the “<email>” in the body.



* + 1. Switch Case 2 – Action 2 – Send an email to admin about suspended job

**Description:** This action will email the approver/admin that an error occurred and share the job status and ID.

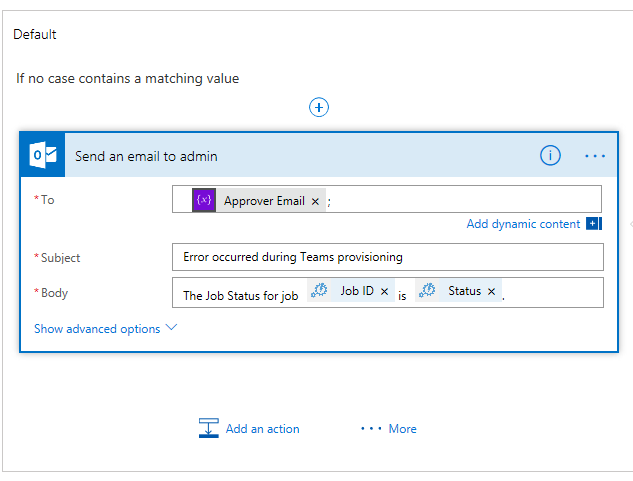
**Action needed (Optional):** It may be wise to change the email address in the “to” line to an admin that can review the runbook jobs instead of someone who is the approver if these are two different people.  


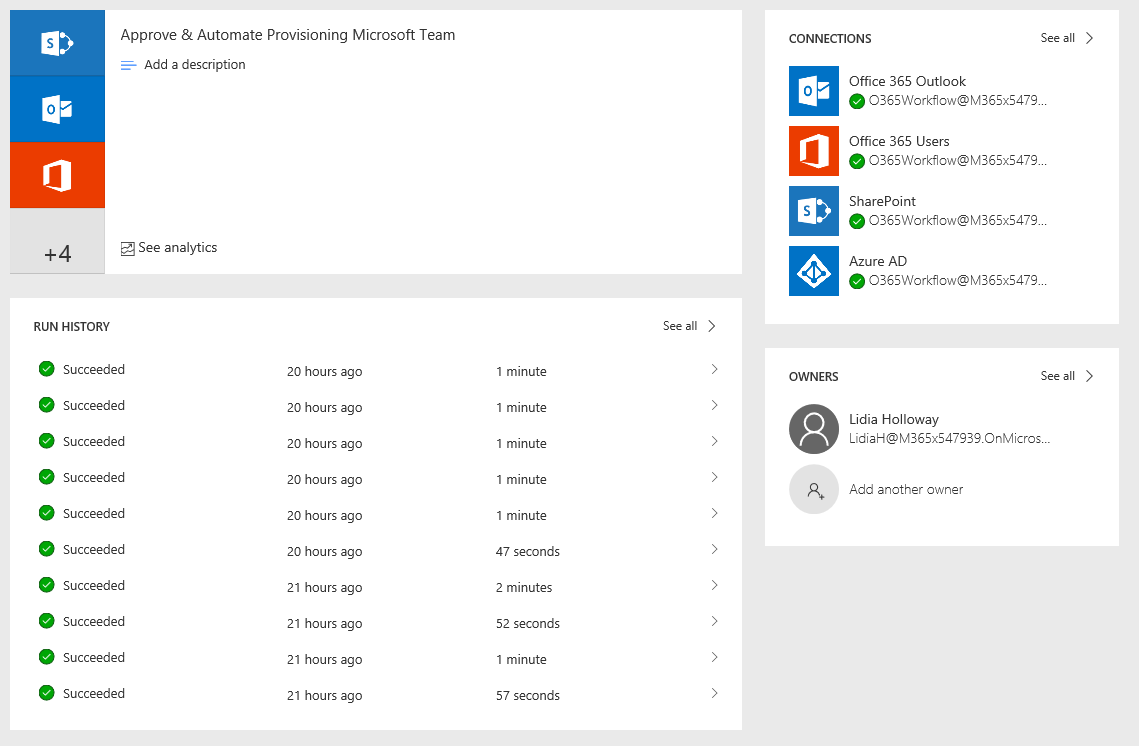
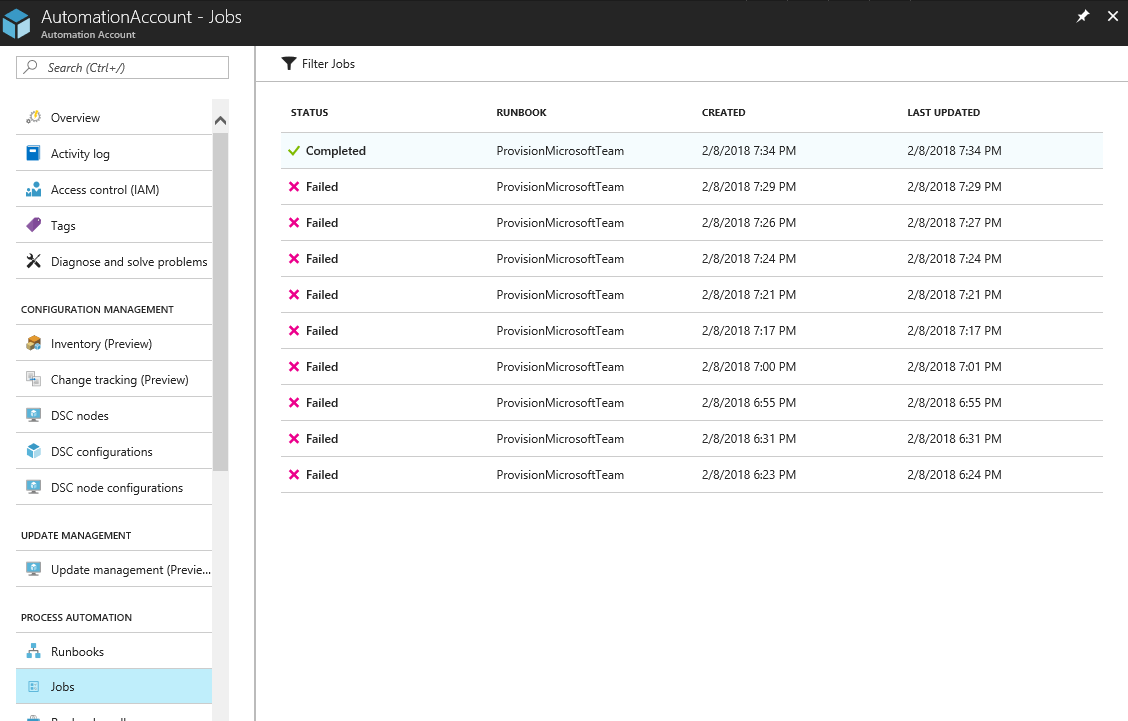
* + 1. Default Switch Case – Send an email to admin

As the default case, this will be chosen if the input does not match any other case. Oftentimes, this is chosen when the job status is “Failed.”

**Description:** This action will email the approver/admin the job status and ID.

**Action needed (Optional):** It may be wise to change the email address in the “to” line to an admin that can review the runbook jobs instead of someone who is the approver if these are two different people.



1. Recommended Testing & Debugging Tips
2. You can debug the workflow by reviewing the **Run History** for a specific run  
   
3. You can see the runbook job output by reviewing the **Jobs** (in your Automation account, look for “Process Automation->Jobs”

**Test 1**

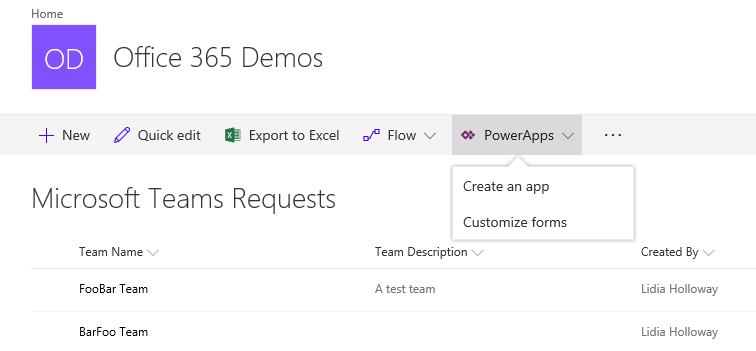
1. Create a new item in the SharePoint List with just a team name. Look for emails sent by the workflow. Check the Teams application after the confirmation email

**Test 2**

1. Choose a team name that already exists. Review the Flow run history and Azure AD groups to confirm a new Office 365 Group was not created.
2. Create a PowerApp for Provisioning Requests (Optional)

If you do not want users to use the SharePoint list as their interface for requesting a Microsoft Team, [PowerApps](https://docs.microsoft.com/en-us/powerapps/guided-learning/) can be a useful alternative. The form used to submit a request in the SharePoint list can also be a PowerApp. To create the PowerApp

1. Go to the SharePoint list
2. Select **PowerApps** in the toolbar, then select “Create an app” or “Customize forms”



Creating an app will allow you to create a standalone app that is tied to this SharePoint list. Customizing forms will change the default form used to create and edit items in the SharePoint list. Refer to the [guided learning](https://docs.microsoft.com/en-us/powerapps/guided-learning/) to learn more about PowerApps in general. Instructions for refining the PowerApp are outside the scope of this document.

1. Conclusion

If you have successfully gone through all the creation steps and now have a working solution, congratulations! I hope this solution proves to be useful and valuable. If you encountered issues during creation, feel free to create a new issue in the [GitHub repo](https://github.com/jason-ortiz/provision-ms-team/issues).